Discovering Compact and Informative Structures through Data Partitioning

Madalina Fiterau

September 2015
CMU-ML-15-105

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Artur Dubrawski, Chair
Geoff Gordon
Alex Smola
Andreas Krause, ETH Zürich

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This work supported by: National Science Foundation grant numbers IIS0911032 and IIS1320347; National Institutes of Health grant number R01391201; Air Force Research Laboratory contract numbers FA875010C0210 and FA87501220324; Department of Energy contract number DEAC5207NA27344; Department of the Treasury contract number TIRNO-06-D-00020; Department of the Interior contract number GS35F0273L; Department of the Navy contract numbers N6600107D0015 and N6600112D0048; the University of Pittsburgh contract numbers 0026043 and 0038431; and a Highmark Research Award.

Keywords: informative projection recovery, cost-based feature selection, ensemble methods, data partitioning, active learning, clinical data analysis, artifact adjudication, nuclear threat detection

Abstract

In many practical scenarios, prediction for high-dimensional observations can be accurately performed using only a fraction of the existing features. However, the set of relevant predictive features, known as the sparsity pattern, varies across data. For instance, features that are informative for a subset of observations might be useless for the rest. In fact, in such cases, the dataset can be seen as an aggregation of samples belonging to several low-dimensional sub-models, potentially due to different generative processes. My thesis introduces several techniques for identifying sparse predictive structures and the areas of the feature space where these structures are effective. This information allows the training of models which perform better than those obtained through traditional feature selection.

We formalize Informative Projection Recovery, the problem of extracting a set of low-dimensional projections of data which jointly form an accurate solution to a given learning task. Our solution to this problem is a regression-based algorithm that identifies informative projections by optimizing over a matrix of point-wise loss estimators. It generalizes to a number of machine learning problems, offering solutions to classification, clustering and regression tasks. Experiments show that our method can discover and leverage low-dimensional structure, yielding accurate and compact models. Our method is particularly useful in applications involving multivariate numeric data in which expert assessment of the results is of the essence. Additionally, we developed an active learning framework which works with the obtained compact models in finding unlabeled data deemed to be worth expert evaluation. For this purpose, we enhance standard active selection criteria using the information encapsulated by the trained model. The advantage of our approach is that the labeling effort is expended mainly on samples which benefit models from the hypothesis class we are considering. Additionally, the domain experts benefit from the availability of informative axis aligned projections at the time of labeling. Experiments show that this results in an improved learning rate over standard selection criteria, both for synthetic data and real-world data from the clinical domain, while the comprehensible view of the data supports the labeling process and helps preempt labeling errors.

Acknowledgments

This research was done in collaboration with:
Karen Chen, CMU, Auton Lab
Gilles Clermont, University of Pittsburgh
Artur Dubrawski, CMU, Auton Lab
Nick Gisolfi, CMU, Robotics
Geoff Gordon, CMU, MLD
Mathieu Guillaume-Bert, CMU, Auton Lab
Marilyn Hravnak, University of Pittsburgh
Michael R. Pinsky, University of Pittsburgh
Alex Smola, CMU, MLD
Donghan Wang, CMU, Auton Lab

Contents

List of Figures

List of Tables

Chapter 1

Introduction

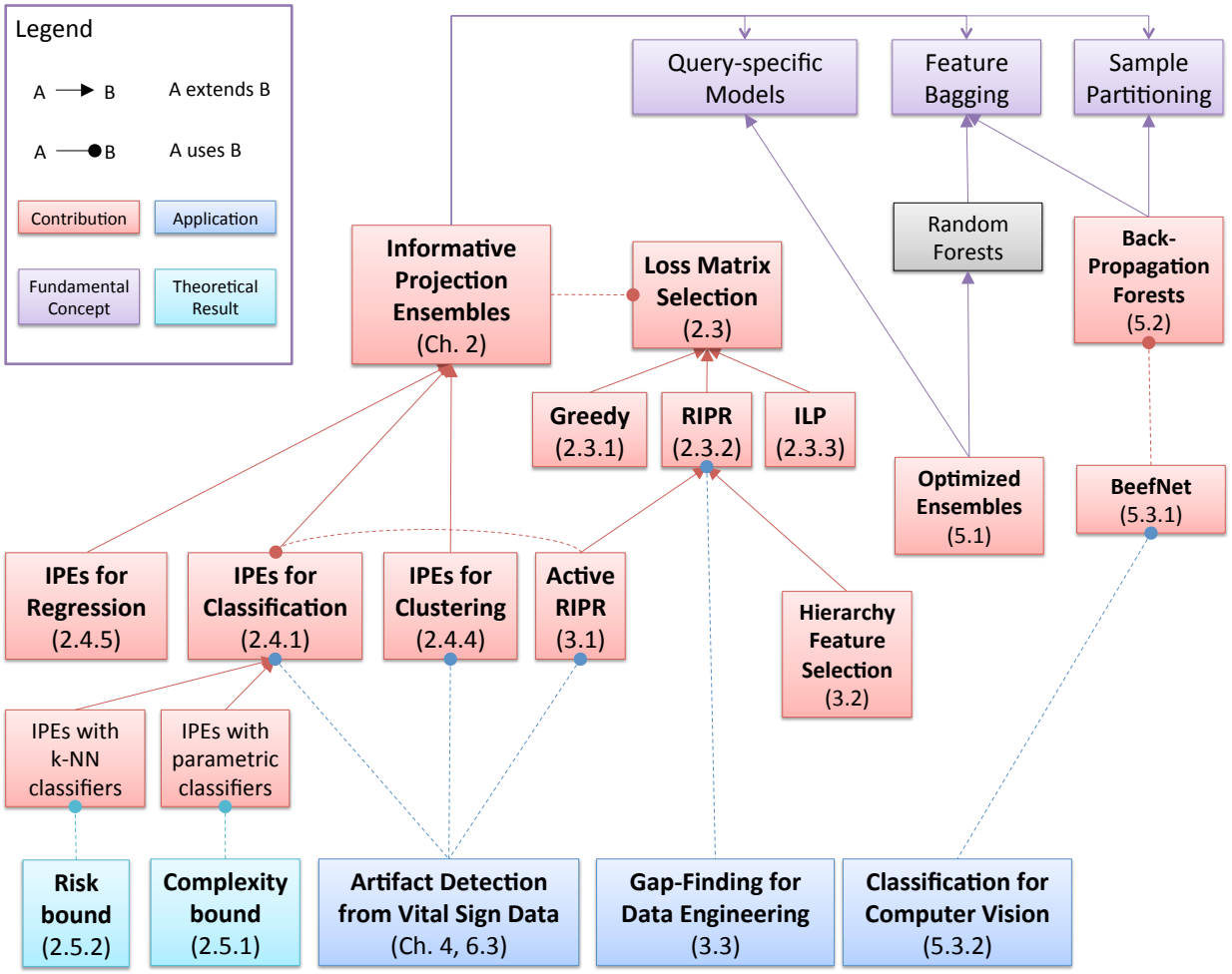
As we all know, the past decade has brought incredible advancements in terms of data acquisition. Despite its immense benefits, Big Data is complex, noisy and utterly unusable by the general public in its original form. The ML community has brought in a plethora of powerful, complex and highly accurate models to handle the data. Sadly, while ML experts have a keen understanding of the models, they are, too often, not accessible to regular users. Thus, the gap between users and data is compounded by an ever increasing gap between users and models.

In this context, my research focuses on finding important aspects of the data and make them accessible to many more users than currently possible. The methods we introduce build compact, interpretable models that, through simple visualizations, put the data back in the hands of the users. There are a number of applications where this is useful, in particular, for decision support systems, where interpretability is key. In applications such as border control and medical diagnostics, a design requirement is that the models should be understood by users, in order to aid them in everyday decision making. The thesis which we put forth states that *it is possible to identify low dimensional structures in complex high-dimensional data, if such structures exist, and leverage them to construct compact interpretable models for various machine learning tasks.*

1.1 Thesis overview

The first part of the thesis focuses on Informative Projection Ensembles (IPEs), their construction and their applicability to several practical problems. IPEs are compact models designed to obtain high performance for a learning task such as classification or clustering, while ensuring that users gain an understanding of the data. IPEs consist of several low-dimensional models obtained by leveraging data partitioning and use query-specific information to handle samples. Next, we show that query-specific handling of data can improve the accuracy of tree ensembles. Finally, we introduce a tree structure which allows the dynamic re-allocation of samples and features, through the use of back-propagation. The resulting ensemble, Back-Propagation Forests, uses both feature bagging, subset selection and data partitioning, illustrating how the same concepts used to extract informative projections can be leveraged toward improving the state of the art when combined with a powerful representation learning mechanism.

Assume we have a heterogeneous dataset that is an aggregation of samples from a multitude of sources. For instance clinical data coming from different patients. The typical approach to deter-



mining sparse predictive structures is to learn a set of features that are globally informative. This could be difficult because the sparsity pattern can change across the input space. At the other end of the spectrum, there are local models which estimate the relevant features in the neighborhood of different samples. However, there might be insufficient training data in the neighborhood of the sample. We propose a tradeoff between the two: compact data partitioning models. The methods that we introduce automatically split the data into several groups based on the existence of low-dimensional structure. The application requirements for compact, interpretable models led to our formulation of the Informative Projection Retrieval (IPR) problem, which is used to train what we call Informative Projection Ensembles, presented in Chapter 2. The IPR problem is relevant to applications where both the model and its handling of queries need to be understood by users. Given a learning task, informative projection retrieval is the problem of selecting a few subspaces in which queries can be confidently resolved while maintaining low expected risk. Our methods focus on axis-aligned projections, or sets of features. The methods presented are applicable to

other low-dimensional models, but making use of the informative features makes the models more interpretable. In general, the datasets we are targeting have a multitude of redundant features, which contain little useful information. We work under the assumption that one or several sets of features, however, do contain structure. For each of these informative projections, the structure may span only part of the samples. Jointly, the informative projections provide structure for all the data.

The framework we introduce fulfills the design requirements for a broad range of learning tasks semi-supervised classification, regression and clustering. The framework accepts a query point, selects a low-dimensional subspace of the features on which to project the point, then applies a task solver on the subspace. Finally, the outcome is shown in the context of the low-dimensional projection. Given the structure we are attempting to learn, the training process needs to resolve two issues. First, the samples are split across the informative projections. This of course includes the task of narrowing down the set of projections. Second, there is the training of solvers on each projection. There are of course many ways to assign the points to the projections, so determining the best k set of projections by considering all possible point assignments is very difficult. Once the set of projections and solvers is selected, they will be used to handle queries. There is a limit to the number of projections we can pick, the difficulty being that there are many possible combinations of projections. To keep the model simple, we keep the number of projections small. Clearly, an ensemble of low loss projections guarantees good performance. We may include a projection using all features for the difficult-to-classify points.

We express the problem of learning Informative Projection Ensembles as a combinatorial problem over elements of a loss matrix, as explained in Section 2.2. The loss matrix L quantifies how well the solver on each projection handles every sample point. One element L_ij is the loss estimator of projection j for point i. These estimators are specific to the learning task. The point is that we need to decrease the loss while selecting one projection for every sample point. We could select the best k projections by using the marginal loss for the decisions. However, this solution might be very far from the optimum. Instead, our optimization procedures find a set of projections for which the loss estimators are as close to the optimum as possible. To indicate the point-to-projection assignment, we introduce B, a binary selection matrix of the same dimensionality as L. B_{ij} is 1 iff projection j is to be used to handle point i. Given L, B induces a loss over the training data. We learn B to minimize the difference between the induced loss and the optimal loss, putting a constraint on the number of non-zero columns. While the selection matrix indicates how training points are assigned to submodels, determining the appropriate submodel for testing data is done by either training a selection function based on the selection matrix or by selecting, for each test point the submodel with the lowest estimated loss.

We present three solutions to learning the selection matrix. The first one, presented in Section 2.3.1, relies on an integer linear program which directly minimizes the induced loss. The column constraints of the ILP ensure that the selection matrix has up to k non-zero columns. The ILP finds the best k sub-models for the training data, however, it has the highest computational cost. The second method, called Regression for Informative Projection Recovery (RIPR), presented in 2.3.2, uses a convex optimization procedure, related to the adaptive lasso. Through regularization, it provides a solution that trades-off between getting close to the minimum loss and using a small number of submodels. The third solution, introduced in Section 2.3.3 is a greedy procedure which iteratively adds submodels to the ensemble such that the loss decrease is maximal at each iteration. The greedy procedure offers a guarantee on how close the loss is to the optimal k-submodel loss

obtained through the ILP.

We can apply either of the three procedures for classification, clustering, regression and other tasks by simply replacing the loss function and appropriately computing the loss matrix. For k-NN classification, presented in Section 2.4.1 we use a local estimator for conditional entropy, which quantifies the heterogeneity of the output in the neighborhood of a point i on the projection c_j. If there are unlabeled samples in the dataset, we'll select the lowest loss value across all possible label assignments. We similarly devise losses for parametric classification, for instance, in the case of SVM classifiers we use the hinge loss. Tests on the three selection procedures for k-NN classification have shown that in terms of accuracy, the ILP performs the best, RIPR second-best, Greedy version, though fast, yields least-accurate models. Performance is close to (or better than) that of a global model although the Informative Projection Ensembles use considerably fewer features.

For clustering, described in Section 2.4.4 we use a loss that is lower for densely-packed regions, namely the negative KL divergence between the current data distribution and a uniform distribution over the same space. This loss function has the advantage that dimensionality issues due to the different subspace sizes are eliminated. Our experiments show that, while standard k-means fails to discover the low-d structure, applying the RIPR procedure reveals the hidden structure in data and the clusters are clearly visible and homogeneous in the low-dimensional projections. Presenting clusters that are clearly distinct in low-dimensions facilitates understanding by human users. We evaluate the clusters by computing the distortion (mean distance to cluster centers) and log-cluster-volume, which measures compactness in the fully dimensional feature space. Tested on UCI data, RIPR clearly outperforms k-means in terms of compactness, obtaining drastic compression according to both metrics.

The loss function for regression is the MSE of the regressor computed in the neighborhood of the sample point, as shown in Section 2.4.5. We compared the MSE of SVMs with the MSE of IPEs using SVM regressors which were obtained with RIPR. In many cases, the low-d model performed better. Low-d structures are less apparent when data is more noisy, however, for low-noise settings, the underlying patterns are detected.

Regardless of the learning task or the selection matrix learning procedure, once the optimal set of projections projections is determined, test samples are assigned to the appropriate projection by selecting the submodel in the ensemble that has the best chance of succeeding in classifying the point. The query handling process is shown in Section 2.3.4. This means we pick the submodel which yields the lowest possible estimated loss for the point.

We have also extended the IPE learning framework to work in an active learning setting, as described in Section 3.1. We typically use RIPR to learn the selection matrix in this case, calling the resulting procedure ActiveRIPR, as it provides the best trade-off between speed and accuracy, though the ILP or the Greedy method can also be used. As far as the sample selection is concerned, any of the popular scoring functions can be used, although we found Information Gain to be the best performing in practice. Empirical results show faster convergence rates when compared to random forests with active learning as only the relevant dimensions of the feature space are explored. In addition, informative projections make adjudications easier and intuitive for users.

The theoretical results we obtained for IPE models are presented in Section 2.5. For IPEs using parametric classifiers from hypothesis classes with finite VC dimensions, using selection functions of finite complexity are also upper bounded in complexity. The growth function of the ensemble class is upper bounded by the product of the growth functions of the hypothesis classes

for the submodels. Since the latter are low-dimensional, the complexity of the ensemble is, in many cases, lower than that of a single classifier using all the dimensions. In addition, IPEs using k-NN classifiers are consistent under certain assumptions concerning the existence of a predictive sparsity pattern. The risk of the ensemble converges at faster rates compared to that of a k-NN classifier trained on the entire feature space.

In addition to learning informative projections, we have also enhanced out framework to take into account feature hierarchies in order to decrease the feature acquisition cost. For several of our applications, complex features are designed and derived from base features, with the cost of a complex feature depending on the cost of the features it was derived from. In Section 3.2, we introduce special penalties generated based on hierarchical structures, which result in the selection of informative projections composed of features that have low acquisition cost overall.

We have applied several of the techniques, including classification IPEs, clustering IPEs and the active learning framework to a medical application. These case studies are the focus of Chapter 4. Consider a vital sign monitoring system which issues healthcare alerts whenever one of the patient's vital signs is out of its normal range. The alerts are quite frequent, with a typical SDU having one alert go off every 90 seconds. Luckily, some of them are false alerts, due to limitations of the monitoring equipment. We have computed features from the vital signal time series and, based on these, we attempted to find models that adjudicate the true alerts and that characterize the artifacts. We trained classification models for each type of alerts, obtaining high accuracy. At the same time the models can be visualized and were used by domain experts to derive artifact filtering rules. Moreover, by applying RIPR with k-means to clustering artifactual alerts, we identified human interpretable archetypes (patterns) of false alerts as a preliminary step to corrective action plans. In a separate experiment, two experts independently annotated 80 alerts (40 due to the respiratory rate and 40 due to the oxygen saturation levels) which were automatically selected by our ML system. Then these experts adjudicated the same alerts using the available chart time series. We summarized the results to observe the consistency of adjudication.

We have applied our Informative Projection Retrieval framework to the problem of identifying discrepancies between training and test data that are responsible for the reduced performance of a classification system. Intended for use when data acquisition is an iterative process controlled by domain experts, our method exposes insufficiencies of training data and presents them in a user-friendly manner, through low-dimensional projections of data. The proposed process, introduced in Section 3.3 begins with the construction of a classifier. Any plausible type of a classification model can be used, though we employ the random forest method primarily due to its scalability to high-dimensional feature spaces and the computable on-the-fly metrics that diagnose the reliability of predictions being made. We use two metrics that characterize reliability of predictions made by our random forest classifier: Dot-Product-Sum (DPS) and In-Bounds Score (IBS). The Gap-Finding Module identifies where the original classification model experiences considerably low accuracy. We extended the RIPR algorithm to facilitate improvements in training data generation, primarily by leveraging its ability to detect low-dimensional patterns of low performance areas. As a result of executing the Gap-Finding, the resulting low-dimensional subspaces are visualized. The domain experts and data engineers gain intuition as to what data may be missing from the training set and decide which parts of the feature space would most benefit from additional samples. We have applied Gap-Finding to a radiation threat detection system, parts of which contain signatures of threats that were synthesized by domain experts and injected into non-threat data. The data is subject to iterative refinements in order to ensure that the training set is shaped into a faithful

reflection of the test set, optimizing the performance of the threat adjudication system.

So far, we presented the concept of using query-specific classifiers selected from a predefined pool to handle individual data points in classification tasks. We leveraged this concept for ensembles of generic classifiers, rather than just IPEs. The idea is rather straightforward: to train meta-classifiers which determine which submodels should be used for each data point. We apply this idea to improve the performance of existing forest ensembles, without modifying the actual ensemble, in Section 5.1.

The informative projection models as well as the optimized ensembles show that query specific handling and subspace selection can be leveraged to improve the performance of classification systems. We aim to allow data shift across submodels in the ensemble. Allowing flexible assignment of both features and samples can be done by training tree ensembles which admit a differentiable global loss. To achieve this goal, we introduce a new structure called *Back-Propagation Trees* (Section 5.2, which makes the the random forest ensemble amenable to back-propagation. Back-propagation (BP) trees use soft splits, such that a sample is probabilistically assigned to all the leaves. Also, the leaves assign a distribution across the labels. A BP tree has two types of parameters: the splitting parameters θ which determine how a sample is routed in each non-leaf node, and the leaf parameters π which specify the label distribution in each leaf. The θs are obtained through SGD. Θ optimizes the log loss over the entire tree, which is differentiable but non-convex. After each update of the θs the optimal πs are computed exactly as the maximizers of a log concave procedure. Results on public vision datasets show that back-propagation forests improve over random forests and other tree ensembles.

We used BackPropagation Forests to develop Deep Convolutional Neural Decision Forests, as illustrated in Section 5.3. Our novel approach unifies classification trees with the representation learning functionality known from deep convolutional networks by training them in a joint manner. To combine these two worlds, we introduce a stochastic and differentiable decision tree model, which steers the representation learning usually conducted in lower level layers of a (deep) convolutional network. Our model differs from conventional deep networks because a decision forest provides the final predictions and it differs from conventional decision forests because we propose a principled, joint and global optimization of split and leaf node parameters. We show experimental results on benchmark machine learning datasets like MNIST and ImageNet and find on-par or superior results when compared to state-of-the-art deep models. Most remarkably, we obtain a Top5-Error of only 7.84% on ImageNet challenge data when integrating our forest in a single-model GoogLeNet architecture, without any form of training data set augmentation. The resulting BeefNet architecture is described in Section 5.3.2.

My thesis introduces methods that extract intelligible models from data, thus helping turn machine learning models into an extension of human knowledge. The compact models extracted by RIPR were useful in practical application such as artifact detection in clinical alerts. The low dimensional views facilitate decision support and can make data annotation faster. In addition, the framework presented can be easily used for any learning task, such as the data acquisition guidance system for nuclear threat detection. Finally, the compact models presented fulfill our objective of making data accessible to human users.

1.2 Motivation and application requirements

Feature selection is an essential part of model learning for high-dimensional data, especially when few samples are available. Standard approaches to feature selection do not always yield concise models which accurately reflect the underlying structure of the data, mainly because they target the selection of a globally-useful set of features without accounting for the characteristics of individual samples. At the other end of the spectrum, recent advances into query-specific models with feature selection such as localized feature selection and locally-linear embeddings leverage neighborhood information in order to generate a plethora of models, each tailored to a diminutive portion of the feature space.

There are cases in which neither of these two extremes provides a satisfactory solution. On one hand, shoehorning the entire dataset into the same low-dimensional model through techniques such as the lasso runs the risk of bringing unnecessary features into the prediction process for some of the samples, which could hurt accuracy. On the other hand, local models are prone to overfitting, have limited applicability and risk introducing needless complexity. All the while, neither captures a compact but comprehensive picture of the dataset, as sought by domain experts. My thesis explores the idea of building small ensembles of low-dimensional components (sub-models) which are applicable to significant subsets of data.

To exemplify, consider a medical application where existing vital sign readings, signals derived from them and a number of other contextual features are used to predict a potentially multivariate output signal such as diagnostics or health-status change alerts. The input space is extensive, containing, at the very least, the readings computed within a window of a few minutes with their corresponding statistics. Each event of interest needs to be manually labeled by clinicians, which requires considerable time and effort, yielding a short supply of labeled data. Given the high feature-to-sample ratio (the problem could even be underdetermined), feature selection is necessary. However, we expect that patients suffer from different underlying conditions and have different characteristics, which is why having several sparse models which are used alternatively, rather than a single generic one, makes more sense. Standard feature selection could pinpoint that blood sugar level is relevant to predicting heart failures. In contrast, a small ensemble model can also identify the conditions under which the feature affects the prediction. For instance, we might find that blood sugar level is only a factor in heart failure prediction when an affine combination of the blood pressure, heart rate and risk of diabetes is above a certain threshold.

As an added incentive, small ensembles of low-dimensional models are also amenable to visualization. This is particularly appealing for applications where human operators have to gain an understanding of the data, and/or quickly validate the system-made predictions. An example of such an application is the detection of nuclear threats at border control points based on vehicle characteristics and measured characteristics of emitted radiation. The automated threat detection system assigns a threat/non-threat label to each vehicle, but it is ultimately up to the border control agents to permit/deny entry or submit the vehicle to further verification. Establishing confidence in the system's decision, if possible, is an important aspect of this application, and can be achieved by providing a visual representation of the classification process. To our ensemble-building methods, this translates as an upper bound on the dimensionality of the components.

Our proposed family of methods works under the assumption that groups of samples can be classified with different small subsets of features. The aim is to uncover the informative sparsity patterns across the feature space, provided that the changes in feature relevance can also be

characterized through sparse functions. We propose to achieve this by training ensembles of low-dimensional components such that every sample can be handled using one of these sub-models or using a sparse mixture of them. We assume no prior knowledge of the sample groups, which could overlap. The assignment of samples to sub-models and the dimensionality reduction for the learners on the sub-models are performed jointly, avoiding the pitfalls of EM-like approaches.

1.3 Scope and novelty of proposed approach

To address the demand for concise, interpretable and visualizable models, we develop a framework which recovers compact ensembles, consisting of solvers (which can be regressors or classifiers) trained on what we call 'informative projections' [34, 35]. An Informative Projection is a low-dimensional transformation of the features, where the learning task can be accurately and reliably solved for a group of samples. We obtain these models through convex procedures, avoiding the issues typically encountered with mixture formulations by estimating the performance of low-dimensional solvers on the training data. The low-dimensional projections responsible for each part of the feature space are selected through an optimization which factors in the appropriate sparsity, smoothness and cost constraints over the parameters. Conceptually, we are combining the flexibility of hierarchical latent variable models [11] and sparse mixture models [54] with the convex formulations and the theoretical guarantees inherent to sparse structured learning [2, 56].

One of the novel aspects of our approach to building compact ensembles is the computation of a matrix which estimates the performance of the low-dimensional solvers at each sample, typically using some non-parametric divergence-based estimator. Once this loss matrix is obtained, it is used in a convex program which optimizes the empirical risk given the established model class. The procedure is detailed in the following chapter. A prerequisite for this type of method is that the learning task admits risk-consistent loss estimators. The only other established methods which learn models resembling those we seek involve non-convex learning procedures to obtain sparse mixtures, such as the method introduced by Larsson and Ugander [68] for MAP inference with a sparsity-inducing generalization of the Dirichlet prior.

Since the overall objective is to obtain a compact representation of the data, the size of the ensemble should be constrained. Determining the number of sub-models intrinsic to a dataset is a key model-selection challenge, which we address through regularization by adding component-wise sparsity penalties. To further compress the model, each component in the ensemble will be low-dimensional, with sparsity being the most favorable option. Regularization is also used to reduce component dimensionality, with the caveat that, in some scenarios, additional restrictions will be imposed. For instance, if human-interpretable visualization is desired, each component would only use up to three features. The components learned with our method will differ significantly either in terms of their sparsity patterns or their parameters, with the discrepancies increasing as the number of sub-models becomes more limited. The range varies between ensembles with few, very different components and larger ensembles where some characteristics (features) can be shared across the components.

During the ensemble learning process, samples are assigned to the components as the sub-models are being built. Each sample can be allocated to one sub-model, thus achieving a partitioning of the feature space, or to a very small number of them, similar to sparse mixture models. Conceptually, the partitioning variant makes it easier for human users to understand the trained

model and to follow the handling of test queries. However, enforcing a hard division of samples across sub-models could be contrary to the realities of the data. We explore and compare these two design options, choosing the appropriate one depending on the application and dataset characteristics.

1.4 Challenges in learning data partitioning ensembles

One of the main computational issues characteristic to this type of model is the 'chicken and egg' problem associated with assigning training samples to sub-models. This happens because the sub-models themselves are built based on their assigned samples. While traditional methods would rely on expectation-maximization, our methods avert this complication by formulating the learning problem as a convex program. The constants in the program are non-parametric estimates that assess the benefit of different candidate models in the neighborhood of each sample of the dataset. The parameters that need to be learned are the assignments of the samples to sub-models. A consensus across the samples is reached concerning which set of models is most useful overall. The benefit of this procedure is that it evaluates whole models rather than individual features. This technique is inherently robust in that, in the neighborhood of any sample point, the model is less sensitive to changes elsewhere in the feature space. The process of finding the models we are targeting raises some issues, a notable one being identifiability. Namely, there could be several very different, albeit accurate, alternatives which solve the learning problem under the settings described above. While our methods work by formulating an objective function and selecting the best performing one, we also take steps to ensure robustness of the selected model and derive necessary conditions for identifiability. A related issue is the use of regularization and the trade-offs between ensemble size and component complexity, which we investigate (so far, only empirically) in order to determine how to best set the parameters (and constraints) to obtain optimal performance for a given dataset.

Our method of learning ensembles of compact solvers improve on existing non-specialized models, at least for data which complies with the assumption that any given query can be handled using only a subset of the initial features. We primarily target classification, although the basic concepts also apply to regression and clustering. We showed experimentally and are we looking to prove that the models obtain faster learning rates, in terms of sample size, than (1) non-specialized models with solvers from the same hypothesis space using all the features and (2) non-specialized models with solvers from the same hypothesis space using the same number of features as the ensemble. In the latter case, we also expect to obtain higher limiting accuracy (3).[1]

1.5 Related work

Extensive research in dimensionality reduction has resulted in a number of techniques which we use in the development and analysis of our algorithms. The problem we address is related to structured sparse learning [57] and compressed sensing [17]. Our method has an advantage over

[1]Points (2) and (3) are straightforward to show since, for partitions, the ensemble is a more generic class, implying that it will fit the data better, but will take longer to train. The elimination of spurious features reduces the amount of needed training samples.

them as it partitions the data, as opposed to building a universal model. Specifically, the analysis of our methods relies on existing theoretical results in structured sparsity [49, 77, 87, 110, 111], as well as the optimization methods that make this type of learning possible [3, 4, 76]. Also, low-dimensional ensemble components can be learned under the assumption that subsets of the given samples can be written as sparse signals in some basis and thus admit a compressed representation (in the form of basis/matching pursuit), which can be determined through existing techniques [6, 44].

We also note some conceptual similarities to hierarchical latent variable models [11, 70] and sparse mixture models [32, 98] – the notion of several underlying processes that determine the output signal. However, our methodology remains very different from standard algorithms on these topics, as we avoid non-convexity by directly operating on the feature space, without the use of intermediaries such as latent variables or mixture components.

Currently, our approaches use axis-aligned subspaces (through lasso penalties) or linear combinations of features (via compressed sensing), but if these fail to deliver the required compact ensemble, we will approach the problem from a nonlinear perspective [69, 92]. Given the multi-model characteristics of the data we target, we use techniques which explicitly learn several manifolds before training the set of solvers [107] or, alternatively, employ multiple kernel learning [48]. Either way, these techniques assume that all data falls under the same model and extra mechanisms are required to assign groups of samples to manifolds/kernels.

Currently, there exist several ensemble-based methods to which we can relate our work [26, 28, 50, 103]. Most of these are, however, purely empirical and not accommodating of theoretical analysis. Our approach not only provides a model which is more representative of the underlying processes and more communicative to the domain experts, but it does so in a manner that makes it possible to obtain theoretical guarantees.

Chapter 2

Informative Projection Ensembles (IPEs)

Intelligent decision support systems often require human involvement because of data limitations, such as the absence of contextual information, as well as due to the need for accountability. The stringency of the requirement usually escalates with the stakes of decisions being made. Notable examples include medical diagnosis or nuclear threat detection, but the benefits of explainable analytics are universal. To meet these requirements, the output of a regression, clustering, or a classification system must therefore be presented in a form that is comprehensible and intuitive to humans, while offering the users insight into how the learning task was accomplished. A desirable solution consists of a small number of low-dimensional (not higher than 3D) projections of data, selected from among the original dimensions, that jointly provide good accuracy while exposing the processes of inference and prediction to visual inspection by humans.

Extracting compact and communicative models is fundamental to decision support systems. Specifically, the use of *Informative Projections* has been shown to facilitate the decision process, making automatic classification transparent and providing domain experts useful views of the data. In this chapter, we provide guarantees for the previously introduced ensembles of classifiers trained on low-dimensional Informative Projections. We analyze the theoretical properties of such ensembles, presenting three methods of training them. We provide optimality guarantees for our algorithms, under the assumption that the sample-specific information dictates the use of the classifiers in the ensembles. Our experiments demonstrate that high classification accuracy can be obtained using low-dimensional models extracted by our methods. Finally, we show how the query-specific solver selection procedure can be applied to other ensembles, improving the performance of random forest classifiers.

Predictive systems designed to keep human operators in the loop rely on the existence of comprehensible classification models, which present an interpretable view of the data [40, 81, 91]. Often, the domain experts require that the test data be represented using a small number of the original features, which serves to validate the classification outcome by highlighting the similarities to relevant training data [108]. The user typically interacts with the system by providing a query (test) point that needs to be labeled; the system then selects a submodel which can accurately classify the query using a small subset of the features; finally, the decision is presented to the user together with a representation of how the classification label was assigned in the projected subspace.

Informative Projection Ensembles (IPEs), first introduced in [34], alternatively use one of several compact submodels that ensure compliance with the stringent requirement on model size,

while also attaining high performance through the use of specialized models. This concept is related to mixtures of experts [10, 59, 63], with the notable difference that the final outcome for any given query is due to only one of the 'experts', making it easier for users to understand and validate.

2.1 Model class

Assume we are given a dataset $X \in \mathbb{R}^{a \times n}$, $Y \in \mathcal{Y}$, typically, $\mathcal{Y} = \{0,1\}^n$, where a is the number of features and n is the sample size. Let $\mathcal{H}_r$ represent hypothesis classes for classifiers on $\mathbb{R}^r$, for some $r \in \mathbb{N}$. We now introduce $\mathcal{M}_{d,k}$, the model class for Informative Projection Ensembles of k axis-aligned projections that have up to d features.

Definition 2.1.1. Given $d, k \in \mathbb{N}$, an *Informative Projection Ensemble* belongs to the following class

$$\begin{aligned} \mathcal{M}_{d,k} = \{M = (C, H, g) \mid\ & C = \{c_i | c_i \subseteq \{1 \dots a\}, a_i \overset{def}{=} |c_i|, a_i \leq d, i \in \{1 \dots k\}\}, \\ & H = \{h_i | h_i \in \mathcal{H}_{a_i}, h : \mathbb{R}^{a_i} \to \{0,1\}, i \in \{1 \dots k\}\}, \\ & g : \mathbb{R}^a \to \{1 \dots k\}\} \end{aligned} \tag{2.1}$$

Above, for an informative projection model M, C is the set which determines the features used by the submodels in the ensemble. Each set c_i for $i \in \{1 \dots k\}$, represents the indices of the a_i features which constitute the *informative projection* of the i^{th} submodel. Each discriminator $h_i \in H$ classifies the samples assigned to it based on the features in the set c_i. The model uses g, which we will refer to as the *selection function*, to determine which discriminator needs to be applied to a given sample. Under the IPE model M, the assigned label of a sample x is $\hat{y} = h_{g(x)}(x_{c_{g(x)}})$.

IPEs have proven useful for applications where data is heterogeneous in nature. In such cases, the vast amount of data available belies the relatively small percentage of it which is actually useful in learning, as the samples typically come from several distributions and are affected by noise [35].

For instance, consider a patient monitoring system built for the classification of health alerts. As the risk of false negatives is high, the predictions have to be ultimately validated by clinicians, requiring comprehensible models. Moreover, this is a typical example of multi-source data as it was collected under different circumstances, from different patients and using multiple sensing modalities [36]. In such cases, frequently encountered in the medical domain, the iid assumption rarely holds, and nor do other typical suppositions concerning the sample distribution or noise. However, there exist groups of samples which exhibit similarities, with the outcome that models tuned to one group will behave poorly on samples from other groups. The use of Informative Projection Ensembles constitutes a solution to both of these issues (i.e. the requirement for simplicity and the heterogeneous nature of the data) because the model partitions the data and uses different low-dimensional projections to classify the points within the context of their group [38].

The clear utility of IPEs in many practical instances, including the clinical alert classification [37] and nuclear threat detection [46, 47], prompts us to analyze the theoretical characteristics of such models in order to highlight their discriminative capabilities and illustrate scenarios in which they are superior to contending models. We also present several ways of learning them from data, with different guarantees in terms of optimality and sample complexity, essentially showing that, under a set of non-stringent assumptions, it is possible to efficiently train near-optimal models.

We make two assertions: (1) that IPEs are the right choice of model in many cases; (2) that informative IPE models can be learned successfully through several techniques. To make an analogy to statistical machine learning: (1) is the 'modeling' error; we show that this model class is rich enough to capture most aspects of data; (2) is the 'approximation' error; we 'approximate' the optimal model training through a greedy/suboptimal procedure.

One could apply decision trees or rule learning to the IPE problem [15]. However, while these models make interpretable classification possible, visualization is difficult without a hard constraint on the number of features to be used. Standard feature selection techniques could also be used [12, 78] though visualization could be difficult for any set of data with more than 3-5 features. It is also unlikely that the same set of features will be relevant for different groups of samples. Instead, our model accounts for the possible existence of several underlying patterns in data. Our approach partitions the data using a specific and dedicated low-dimensional projection to classify each point. This does not only facilitate comprehension, but it can also increase classification accuracy compared to standard feature selection due to reduced model complexity. The solution, be it exact – if finding it is feasible – or approximate, is applicable without the need for post-processing.

In addition to the theoretical findings, our experiments show that the methods we introduce can discover and leverage low-dimensional structure in data, if it exists, yielding accurate and compact models. Our method is particularly useful in applications involving multivariate numeric data in which expert assessment of the results is of the essence.

2.2 Construction of Informative Projection Ensembles

This section describes how the ensemble construction can be reduced to a combinatorial problem by optimizing over a matrix of loss estimators computed for every data point. We introduce three ways of solving this combinatorial problem. First, we formulate an integer linear program which computes the optimal point-to-projection allocation for the training sample given a limited number of projections. An alternative is a two-step procedure, similar to the adaptive lasso, which replaces the constraint on the number of projections with a ℓ_1 penalty with adaptive weights. Finally, we consider greedy projection selection, which is a great option in this case because of the supermodularity of the loss.

We formulate Informative Projection Recovery (IPR) as the problem of identifying IPEs which encapsulate enough information to allow learning of well-performing models. Each such feature group, equivalent to a low-dimensional axis-aligned projection, handles a different subset of data with a specific model. The resulting set of projections, jointly with their corresponding models, form a solution to the IPR problem. We have previously proposed such a solution tailored to non-parametric classification. Our RIPR algorithm [33] employs point estimators for conditional entropy to recover a set of low-dimensional projections that classify queries using non-parametric discriminators in an alternate fashion – each query is classified using one specific projection from the retrieved set.

Solving the IPR problem is relevant in many practical applications. For instance, consider a nuclear threat detection system installed at a border check point. Vehicles crossing the border are scanned with sensors so that a large array of measurements of radioactivity and secondary contextual information is being collected. These observations are fed into a classification system

that determines whether the scanned vehicle may carry a threat. Given the potentially devastating consequences of a false negative, a border control agent is requested to validate the prediction and decide whether to submit the vehicle for a costly further inspection. With the positive classification rate of the system under strict bounds because of limitations in the control process, the risk of false negatives is increased. Despite its crucial role, human intervention should only be withheld for cases in which there is reason to doubt the validity of classification. In order for a user to attest the validity of the decision, the user must have a good understanding of the classification process, which happens more readily when the classifier only uses the original dataset features rather than combinations of them and when the discrimination models are low-dimensional.

2.2.1 Formulation of IPE learning

Intuitively, the aim is to minimize the expected classification error over $\mathcal{M}_d$, however, a notable modification is that the projection and, implicitly, the discriminator, are chosen according to the data point that needs to be classified. Given a query x in the space $\mathcal{X}$, $g(x)$ will yield the subspace $c_{g(x)}$ onto which the query is projected and the discriminator $h_{g(x)}$ for it. Distinct test points can be handled using different combinations of subspaces and discriminators. We consider models that minimize the loss function ℓ. For a sample x, the label is $\hat{y} = h_{g(x)}(x_{c_{g(x)}})$, which we can use to express the risk for a loss $\ell(\hat{y}, y)$.

$$\begin{aligned}\mathbb{R}(M) &= \mathbb{E}\ell(\hat{y}, y) = \mathbb{E}\ell(h_{g(x)}(x_{c_{g(x)}}), y) \\ &= \sum_{i=1}^{k} \mathbb{E}\ell(h_i(x_{c_i}), y)\end{aligned}$$

The estimated risk can be expressed in terms of the losses of the individual solvers, evaluated at the data points assigned to them. Below, $x_{.,i}$ is a vector representing the i^{th} sample in the dataset, while x_{i,c_j} represents the projection of this sample on the set of variables c_j.

$$\hat{\mathbb{R}}(M) = \frac{1}{n} \sum_{j=1}^{k} \sum_{i=1}^{n} I(g(x_i) = j)\ell(h_j(x_{i,c_j}), y_i)$$

Trivially, if $\forall j, \hat{\mathbb{R}}(h_j) \to 0$, then $\hat{\mathbb{R}}(M) \to 0$, since

$$\hat{\mathbb{R}}(M) \leq \sum_{j=1}^{k} \frac{1}{n} \sum_{i=1}^{n} \ell(h_j(x_{i,c_j}), y_i) \leq \sum_{j=1}^{k} \hat{\mathbb{R}}(h_j)$$

Hence, the IPR problem for learning classification IPEs can be stated as follows:

$$M^* = \underset{M \in \mathcal{M}_d}{\arg\min}\, \mathbb{E}_{\mathcal{X},\mathcal{Y}} \ell(h_{g(x)}(c_{g(x)}(x)), y) \tag{2.2}$$

There are limitations on the type of selection function g that can be learned. A simple example for which g can be recovered is a set of signal readings x for which, if one of the readings x_i exceeds a threshold t_i, the label can be predicted just based on x_i. A more complex one is a dataset containing regulatory variables, that is, for x_i in the interval $[a_r, b_r]$ the label only depends on $(x_r^1 \dots x_r^{n_r})$. Datasets that fall into the latter category fulfill what we call the Subspace-Separability Assumption.

2.2.2 Classifier selection as a combinatorial problem

There are several problems which need to be addressed in order to learn the IPE models. First, a set of candidate models need to be established, then, k of them are selected and assigned to individual data points. Typically, the set of projections is limited by utility requirements. For instance, in the case of visualization, combinations of 2 or 3 features are considered, so $d \leq 3$. If larger submodels are acceptable, then informative projections can be built through more traditional techniques such as forward or backward feature selection or separate feature evaluation. Although determining the projections of the candidate models is relatively straightforward, there is the additional requirement of estimating their classification performance in order to select the appropriate k-subset. This can be done by either training classifiers and evaluating their performance or computing how much information the projection encodes about the output in the neighborhood of each training data point.

We consider the existence of $\mathbf{m}$ candidate submodels, out of which $\mathbf{k}$ will be selected. We can re-write the estimated risk in terms of two matrices, the *loss matrix* $L \in \mathbb{R}^{n \times m}$ and the *selection matrix* $B \in \{0,1\}^{n \times m}$. The loss matrix quantifies the loss of each candidate submodel for each of the data points. The selection matrix represents the assignment of data points to submodels, $B_{i,j} = 1$ if point i is assigned to submodel j and 0 otherwise.

$$B_{i,j} \stackrel{def}{=} I(g(x_i) = j) \qquad \forall i \in \{1 \dots n\}, \quad j \in \{1 \dots m\} \tag{2.3}$$

$$L_{i,j} \stackrel{def}{=} \ell(h(x_{i,c_j}), y_i) \tag{2.4}$$

The loss matrix can be computed whereas B needs to be learned. We propose several methods to learn B, and thus determine the best set of submodels, as well as the training points assignment. For the test data, we select the classifier in the model with the lowest estimated loss. The procedure to estimate loss depends on the hypothesis classes being considered. For SVM classifiers, for instance, the margin is an appropriate estimator, whereas for nearest-neighbor classification, it could be a ratio between distances to data points of different classes. Since submodel selection is typically not problematic once an informative set of submodels is captured, we first focus on learning the selection matrix B.

For each training point, we compute the best loss amid all the projections, which is simply $T_i = \min_{j \in [m]} L_{ij}$.

The objective can be then further rewritten as a function of the elements of the loss matrix:

$$\min_{M \in \mathcal{M}_d} \sum_{i=1}^{n} \sum_{j=1}^{m} I[g(x_i) = j] L_{ij} \tag{2.5}$$

From the definition of T, it is also clear that

$$\min_{M \in \mathcal{M}_d} \sum_{i=1}^{n} \sum_{j=1}^{m} I[g(x_i) = j] L_{ij} \geq \sum_{i=1}^{n} T_i \,. \tag{2.6}$$

Considering form (2.5) of the objective, and given that the estimates L_{ij} are constants, depending only on the training set, the projection retrieval problem is reduced to finding g for all training points, which will implicitly select the subset of projections to be contained by the model.

Naturally, one might assume the best-performing classification model is the one containing all the axis-aligned subspaces. This model achieves the lower bound (2.6) for the training set. However, the larger the set of projections, the more values the function g takes, and thus the problem of selecting the correct projection becomes more difficult. It becomes apparent that the number of projections should be somehow restricted to allow generalization. Assuming a hard threshold of at most k projections, the optimization (2.5) becomes an entry selection problem over matrix L where one value must be picked from each row under a limitation on the number of columns that can be used. This problem cannot be solved exactly in polynomial time. Instead, it can be formulated as an optimization problem under ℓ_1 constraints.

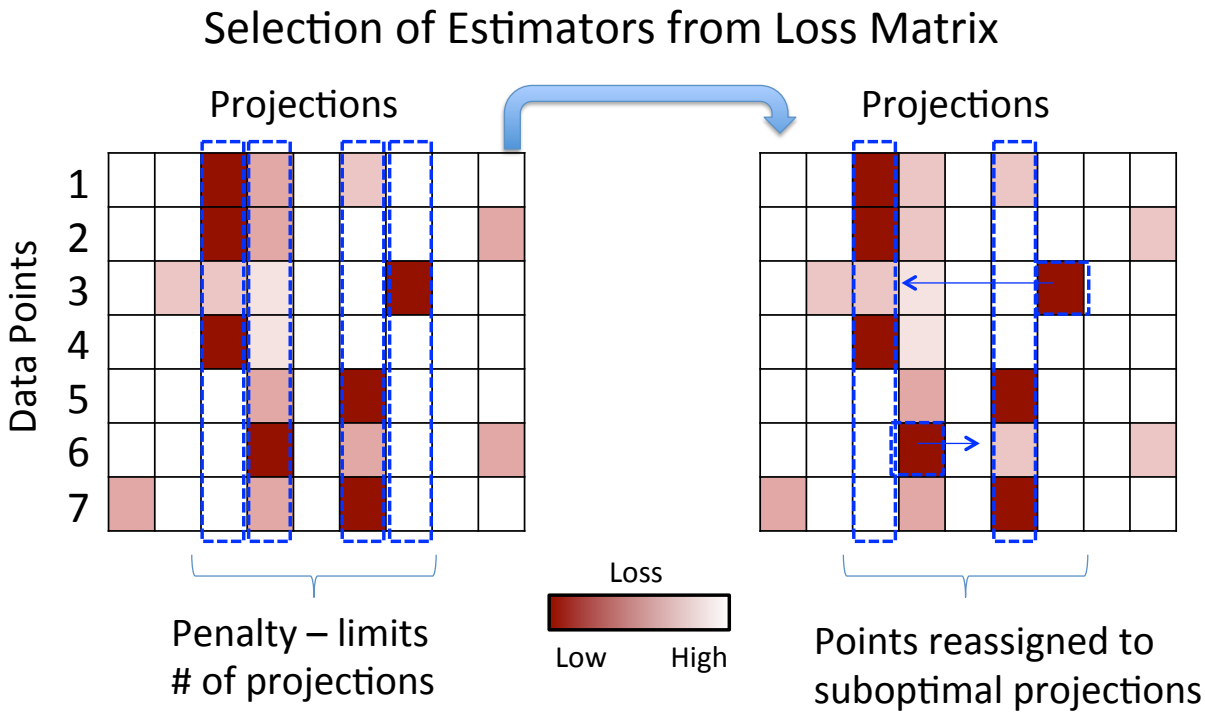


Figure 2.1: Using the loss matrix for projection selection.

Figure 2.1 shows how the projection selection works using the loss matrix. Without regularization, the minimum over each row would be selected. However, by imposing a constraint on the number of columns used, some points have to be re-assigned to sub-optimal projections, though still maintaining the loss as small as possible.

2.3 Learning the selection matrix

2.3.1 Optimal submodel selection through an Integer Linear Program

Learning B is simply a combinatorial problem of selecting an element from each row of the loss matrix and only from at most k of the columns, while minimizing the sum of the losses over

Algorithm 2.3.1 Learning IPEs through an Integer Linear Program

$$
\begin{aligned}
\text{maximize} \quad & -\sum_{i=1}^{n} \mathbf{b_{i,.}^{T}} \ell_{i,.} \\
\text{subject to} \quad & \sum_{j=1}^{m} \mathbf{b_{i,j}} = 1, \qquad \forall i \in \{1 \dots n\}, \\
& \sum_{i=1}^{n} \mathbf{b_{i,j}} \leq n \mathbf{p_j}, \qquad \forall j \in \{1 \dots m\}, \\
& \sum_{j=1}^{m} \mathbf{p_j} \leq k, \\
& 0 \leq \mathbf{b_{i,j}} \leq \mathbf{p_j} \leq 1, \qquad \forall i \in \{1 \dots n\}, \quad \forall j \in \{1 \dots m\}, \\
\text{and} \quad & \mathbf{p_j}, \mathbf{b_{i,j}} \in \mathbb{Z}
\end{aligned}
$$

Algorithm 2.3.2 Learning IPEs through an Integer Linear Program which determines k

$$
\begin{aligned}
\text{maximize} \quad & -\sum_{k=1}^{k_{max}} \sum_{i=1}^{n} \mathbf{b_{i,.,k}^{T}} \ell_{i,.} \\
\text{subject to} \quad & \sum_{j=1}^{m} \mathbf{b_{i,j,k}} = 1, \qquad \forall i \in \{1 \dots n\}, \qquad \forall k \in \{1 \dots k_{max}\} \\
& \sum_{i=1}^{n} \mathbf{b_{i,j,k}} \leq n \mathbf{p_{j,k}}, \qquad \forall j \in \{1 \dots m\}, \\
& \sum_{j=1}^{m} \mathbf{p_{j,k}} \leq k, \qquad \forall k \in \{1 \dots k_{max}\} \\
& 0 \leq \mathbf{b_{i,j,k}} \leq \mathbf{p_{j,k}} \leq \mathbf{s_k} \leq 1, \qquad \forall i \in \{1 \dots n\}, \quad \forall j \in \{1 \dots m\}, \\
& \sum_{k=1}^{k_{max}} \mathbf{s_k} = 1, \\
\text{and} \quad & \mathbf{p_{j,k}}, \mathbf{b_{i,j,k}}, \mathbf{s_k} \in \mathbb{Z}
\end{aligned}
$$

the selected elements. We formulate an ILP, shown in Algorithm 2.3.1, that finds the optimal matrix B for a computed matrix L, given that the final IPE can have at most k submodels. For the purpose of the ILP, $\ell_{i,j}$, the elements of L, are constant. The variables of the LP are $\mathbf{b_{i,j}}$, the elements of the selection function, and $\mathbf{p_j}$, which specifies whether the j^{th} candidate submodel is used in the IPE.

The ILP ensures that the optimal loss is obtained for an ensemble with k submodels. Setting an appropriate value for k is not always straightforward, and one potential solution is to run the

ILP for different values of k and select the best tradeoff between loss and model size. On the other hand, we can ensure that, if there are several models with the same value of the objective function, the model with the smallest number of projections is selected. We slightly modify the ILP as presented in Algorithm 2.3.2, by maintaining one set of variables $\mathbf{b}$ and $\mathbf{p}$ for each k and introducing the binary variables $\mathbf{s_k}$, $\forall k \in \{1 \dots k_{max}\}$. $k_{max} \leq m$ is the maximum number of projections in the IPE.

We will refer to the solution B^* of the ILP (whichever of the two suits the problem) as the *k-optimal selection matrix* given the dataset X and it will serve as a point of comparison with the other methods of learning the selection matrix.

2.3.2 Convex formulations for submodel selection

The problem of learning the selection matrix, and implicitly the informative projection ensemble, can be transformed to a regression problem. We consider T, the minimum obtainable value of the entropy estimator for each data point, as the output which a egress ion model needs to predict. We typically use an entropy estimator as the loss function L, with each row i of the parameter matrix B representing whether the entropy estimates on each projection contribute to the total entropy estimator for the data point x_i. The regression model that selects the optimal submodel for each data point, given the matrix of loss estimates, is

$$\hat{B} = \arg\min_{B} ||T - (B. * L)\mathbf{1}||_2^2, \quad \text{where } T_i \stackrel{def}{=} \min_{1 \leq j \leq m} B_{i,j}, \text{ and } (B. * L)_{i,j} = B_{i,j} L_{i,j}. \quad (2.7)$$

Typically, only one element in each row of B is 1, corresponding to the submodel that is assigned to the data point. Clearly, the regression procedure above, used without regularization would select all the submodels that are optimal for at least one data point. The appropriate regularization term to be used would be a penalty on the total number of non-zero columns in B under the constraint that the sum over each row of B is 1. The sum of ℓ_0 norms of each column is $\lambda_{\ell_0}(B) = \lambda \sum_{j=1}^{m} ||B_{.,j}||_0$.

$$\hat{B} = \arg\min_{B} ||T - (B. * L)\mathbf{1}||_2^2 + \lambda_{\ell_0}(B) \qquad \text{subject to } ||B_{i,.}||_1 = 1, \forall 1 \leq i \leq n \quad (2.8)$$

The challenge with this optimization problem is that it is not convex. A typical work-around for this issue is to use a convex relaxation, the ℓ_1 norm. This would transform the penalized term:

$$\lambda_{\ell_1}(B) = \sum_{j=1}^{m} ||B_{.,j}||_1.$$

However, under the row constraints,

$$\sum_{j=1}^{m} ||B_{.,j}||_1 = \sum_{i=1}^{n} ||B_{i,.}||_1 = n,$$

which means that the penalty has no effect when the row constraints are enforced.

Two-step convex submodel selection

An alternative is to bias the non-zero elements in B towards a small number of columns, such that fewer submodels are used overall. The mechanism we propose to achieve this is similar to the adaptive lasso. Adding a penalty of the form $B\delta$, where δ is an m-sized column vector with each element representing the penalty for a column in B. The penalty is lower for submodels that are useful for a large subset of data, which correspond to denser columns in B, and higher for less useful submodels. To determine submodel usefullness, we get an initial estimator for B using only the row constraints, which themselves ensure a bound on the ℓ_1 norm. Next, we refine the selection by adapting the regularization weights for each of the submodels according to the previously determined B. The results in this section refer to the 2-step procedure presented in Algorithm 2.3.3.

Algorithm 2.3.3 Two-Stage IPE learning (Regression for Informative Projection Recovery [35])

$$\tilde{B} = \arg\min_{B} ||T - (B. * L)\mathbf{1}||_2^2 \qquad \text{subject to } ||B_{i,.}||_1 = 1, \forall 1 \leq i \leq n \tag{2.9}$$

$$\hat{B} = \arg\min_{B} ||T - (B. * L)\mathbf{1}||_2^2 + \lambda||B\delta||_1 \tag{2.10}$$

$$\text{where } \delta_j = \frac{1}{||\tilde{B}_{.,j}||_1} \tag{2.11}$$

Iterative convex submodel selection – the RIPR framework

The process could be iterated until the convergence of δ. With no prior information about which subspaces are more informative, δ starts as an all-1 vector. An initial value for B is obtained through the optimization (2.9). Since our goal is to handle data using a small number of projections, δ is then updated such that its value is lower for the denser columns in B. The matrix B itself is updated, and this 2-step process continues until convergence of δ. Once δ converges, the projections corresponding to the non-zero columns of B are added to the model. The procedure is shown in Algorithm 2.3.4. In theory, there are no guarantees so far with respect to whether the process converges or the time to convergence. In practice, however, the method has converged for every dataset on which we tested it.

Algorithm 2.3.4 Framework for Informative Projection Recovery

$\delta = [1 \dots 1]$
repeat
$B = \arg\min_B ||T - (L. * B)\mathbf{1}||_2^2 + \lambda_1 \sum_{j=1}^{m} |B_{.,j}|_1 + \lambda_2 |B\delta|_1$
subject to $|B_{i,.}|_1 = 1, \quad \forall i \in \{1 \dots n\}$
$B_{i,j} \geq 0$
$\delta_j = |B_{.,j}|_1 \qquad j = 1 \dots d^*$ (update multiplier)
$\delta = (|\delta|_1 - \delta)/|\delta|_1$
until δ converges
return $C = \{c_j; \quad |B_{.,j}|_1 > 0 \quad \forall j = 1 \dots d^*\}$

2.3.3 Greedy submodel selection

For a model $M = (C, H, g)$, the simplest selection function is the one which assigns a sample x the projection with the lowest estimated loss for it.

$$g_{min}(x) \stackrel{def}{=} \arg\min_{j \in \{1 \dots k\}} \hat{\ell}(h_j, x) \tag{2.12}$$

$$\hat{\ell}(M, x) = \hat{\ell}(H, x) \stackrel{def}{=} \min_{h_j \in H} \hat{\ell}(h_j, x) \tag{2.13}$$

$$\hat{\ell}(M, X) \stackrel{def}{=} \sum_{x \in X} \hat{\ell}(M, x) = \sum_{x \in X} \min_{h_i \in H} \hat{\ell}(h_i, x) \tag{2.14}$$

Below, we show that the loss we are minimizing under g_{min} is supermodular. This, in turn, means that greedy selection can be applied to construct an ensemble, resulting in a near-optimal ensemble.

As a reminder, a set function F is submodular if, $\forall$ sets A, B s.t. $A \subseteq B$ and for every set element x

$$F(A \cup \{x\}) - F(A) \geq F(B \cup \{x\}) - F(B). \tag{2.15}$$

Let $M_1 = (C_1, H_1, g_m)$, $M_2 = (C_2, H_2, g_m)$ with $C_1 \subseteq C_2$ and $H_1 \subseteq H_2$. Since there are more sub models in $\mathcal{H}_2$, we have that, for any sample $x \in X$

$$\min_{h_i \in H_2} \hat{\ell}(h_i, x) \leq \min_{h_i \in H_1} \hat{\ell}(h_i, x) \qquad \text{and} \tag{2.16}$$

$$I[\hat{\ell}(h, x) < \min_{h_i \in H_2} \hat{\ell}(h_i, x)] \leq I[\hat{\ell}(h, x) < \min_{h_i \in H_1} \hat{\ell}(h_i, x)]. \tag{2.17}$$

We now study the behavior of the loss function when a new classifier h is added to the ensemble. First, we show that the min function over a set is supermodular.

Proposition 2.3.1 (Supermodularity of min function)**.** *For all sets A, B with $A \subseteq B$ and elements x*

$$\min(A \cup \{x\}) - \min(A) \leq \min(B \cup \{x\}) - \min(B). \tag{2.18}$$

Proof. From $A \subseteq B$, we have that $\min(B) \leq \min(A)$. We show that the proposition holds for all values of x. If $x \leq \min(B) \leq \min(A)$, then

$$\min(A \cup \{x\}) - \min(A) = x - \min(A) \leq x - \min(B) = \min(B \cup \{x\}) - \min(B).$$

If $\min(B) \leq x \leq \min(A)$, then

$$\min(A \cup \{x\}) - \min(A) = x - \min(A) \leq 0 = \min(B \cup \{x\}) - \min(B).$$

If $\min(B) \leq \min(A) \leq x$, then

$$\min(A \cup \{x\}) - \min(A) = 0 = \min(B \cup \{x\}) - \min(B).$$

□

Adding a classifier h to an ensemble H_1 yields the following loss difference

$$\hat{\ell}(H_1 \cup \{h\}, x) - \hat{\ell}(H_1, x) = \min_{h_i \in H_1 \cup \{h\}} \hat{\ell}(h_i, x) - \min_{h_i \in H_1} \hat{\ell}(h_i, x), \tag{2.19}$$

for which, by applying Proposition 2.3.1 we get the following lemma:

Lemma 2.3.2 (Supermodularity of ensemble loss)**.** *Given a model class for which the selection function assigns each sample to the submodel with the lowest loss, the loss over the IPEs is super modular for any dataset X, in other words, for any $M_1 = (C_1, H_1, g_m)$, $M_2 = (C_2, H_2, g_m)$ with $C_1 \subseteq C_2$ and $H_1 \subseteq H_2$, we have that*

$$\hat{S}(H_1 \cup \{h\}, X) - \hat{S}(H_1, X) \geq \hat{S}(H_2 \cup \{h\}, X) - \hat{S}(H_2, X). \tag{2.20}$$

Proof. The proof follows by simply applying the proposition to the loss function, given its equivalence to a min function over a set expressed in (2.19), to obtain

$$\hat{\ell}(H_1 \cup \{h\}, x) - \hat{\ell}(H_1, x) \leq \hat{\ell}(H_2 \cup \{h\}, x) - \hat{\ell}(H_2, x). \tag{2.21}$$

The result follows since the loss is additive w.r.t the samples in X, according to (2.14). □

Since the loss is supermodular, the opposite of the loss, which we call the score $\hat{S} \stackrel{def}{=} 1 - \hat{\ell}$ is submodular.

$$\hat{S}(H_1 \cup \{h\}, x) - \hat{S}(H_1, x) \geq \hat{S}(H_2 \cup \{h\}, x) - \hat{S}(H_2, x) \tag{2.22}$$

According to the result of Nemhauser et al. [82], by applying a greedy selection procedure to obtain k submodels from m candidates, we have that

$$\hat{S}(M_k^{greedy}, X) \geq (1 - 1/e) \max_{\{M_k^* = (C, H, g_{min}) || H| < k\}} \hat{S}(M_k^*, X). \tag{2.23}$$

2.3.4 Query handling

Once the projections are selected, the second stage of the algorithm deals with assigning the projection with which to classify a particular query point. An immediate way of selecting the correct projection starts by computing the local entropy estimator for each subspace with each class assignment. Then, we may select the label/subspace combination that minimizes the loss.

$$(j^*, y^*) = \arg\min_{j,y} \left(\frac{\nu_k(c_j(x), c_j(X_y))}{\nu_k(c_j(x), c_j(X_{\neg y}))} \right)^{|c_j|(1-\alpha)} \qquad j = 1 \dots m \quad , \quad \alpha \approx 1 \tag{2.24}$$

Figure 2.2 shows the procedure of labeling a test sample given a RIPR model with $k \stackrel{def}{=} |C|$ projections. The framework accepts a query point x, selects the low-dimensional subspace of the features $c_{g(x)}$ on which to project the point, then applies the classifier $h_{g(x)}$ of the subspace. Finally, the classification outcome is shown in the context of the low-dimensional projection, highlighting the projection $c_{g(x)}(x)$ of the test point as well as its neighbors.

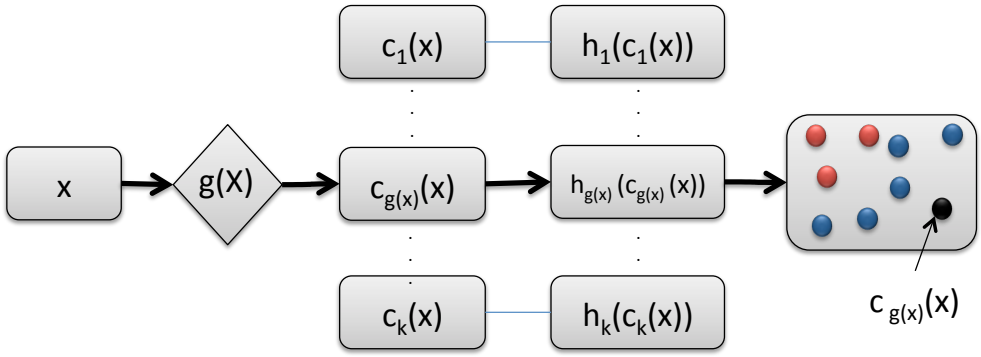


Figure 2.2: The sample labeling procedure.

2.4 Customized IPE construction for different learning tasks

2.4.1 Classification IPEs using conditional entropy

To solve the IPR problem for classification, we need means by which to ascertain which projections are useful in terms of discriminating data from the two classes. Since our model allows the use of distinct projections depending on the query point, it is expected that each projection would potentially benefit different areas of the feature space. $\mathcal{A}_i$ refers to the area of the feature space where the projection c_i is selected.

$$\mathcal{A}_i = \{x \in \mathcal{X} : g(x) = i\} \tag{2.25}$$

The objective becomes

$$\min_{M \in \mathcal{M}_d} E_{\mathcal{X},\mathcal{Y}} \left[y \neq h_{g(x)}(c_{g(x)}(x)) \right] = \min_{M \in \mathcal{M}_d} \sum_{c_i \in C} p(\mathcal{A}_i) \mathbb{E}_{x \in \mathcal{A}_i} \left[y \neq h_{g(x)}(c_{g(x)}(x)) \right] \quad . \tag{2.26}$$

The expected classification error over $\mathcal{A}_i$ is linked to the conditional entropy of $Y|X$. Fano's inequality provides a lower bound on the error while Feder and Merhav [30] derive a tight upper bound on the minimal error probability in terms of the entropy. This means that conditional entropy characterizes the potential of a subset of the feature space to separate data, which is more generic than simply quantifying classification accuracy for a specific discriminator. In view of this connection between classification accuracy and entropy, we adapt the objective to

$$\min_{M \in \mathcal{M}_d} \sum_{c_i \in C} p(\mathcal{A}_i) H(Y|c_i(X); X \in \mathcal{A}_i). \tag{2.27}$$

The method we propose optimizes an empirical analog of (2.27) which we develop below and for which we will need the following result.

Proposition 2.4.1. *Given a continuous variable $X \in \mathcal{X}$ and a binary variable Y, where X is sampled from the mixture model.*

$$f(x) = p(y=0) f_0(x) + p(y=1) f_1(x) = p_0 f_0(x) + p_1 f_1(x)\,,$$

$$\textit{then} \quad H(Y|X) = -p_0 \log p_0 - p_1 \log p_1 - D_{KL}(f_0||f) - D_{KL}(f_1||f)$$

Next, we will use the nonparametric estimator presented in [83] for Tsallis α-divergence. Given samples $u_i \sim U$, with $i = 1 \ldots n$ and $v_j \sim V$ with $j = 1 \ldots m$, the divergence is estimated as follows:

$$\hat{T}_\alpha(u||v) = \frac{1}{1-\alpha}\left[\frac{1}{n}\sum_{i=1}^{n}\left(\frac{(n-1)\nu_k(u_i, U \setminus u_i)^d}{m\nu_k(u_i, V)^d}\right)^{1-\alpha} B_{k,\alpha} - 1\right], \tag{2.28}$$

where d is the dimensionality of the variables U and V and $\nu_k(z, Z)$ represents the distance from z to its k^{th} nearest neighbor of the set of points Z. For $\alpha \approx 1$ and $n \to \infty$, $\hat{T}_\alpha(u||v) \approx D_{KL}(u||v)$.

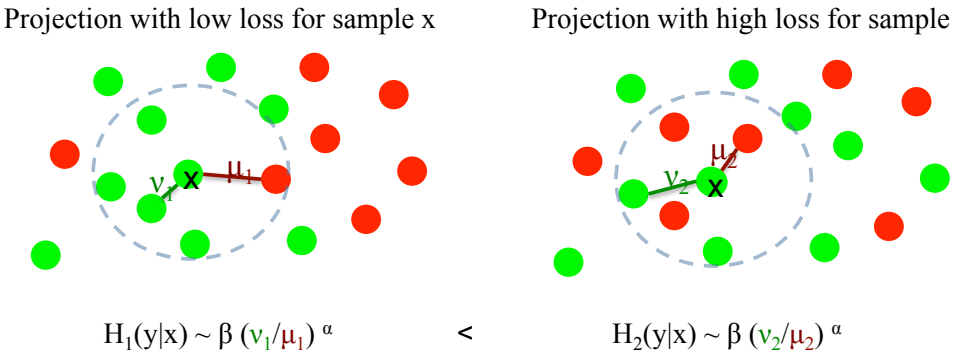


Figure 2.3: Estimating entropy through distance ratios.

We use the divergence estimator to estimate the conditional entropy at every sample point. The entropy estimation is based on the ratio of the distances to the k-nearest point of the same class and the k-nearest point of the opposite class. An illustration of this is presented in Figure 2.3. Intuitively, the smaller the ratio, the closer the point is to other samples of the same class, which

means that the neighborhood is homogeneous. If the ratio is close to or higher than 1, it means that the points of different classes are about the same distance with respect to the sample considered, so the neighborhood is heterogeneous.

We will now plug (2.28) in the formula obtained by Proposition 2.4.1 to estimate the quantity (2.27). We use the notation X_0 to represent the n_0 samples from X which have the labels Y equal to 0, and X_1 to represent the n_1 samples from X which have the labels set to 1. Also, $X_{y(x)}$ represents the set of samples that have labels equal to the label of x and $X_{\neg y(x)}$ the data that have labels opposite to the label of x.

$$\hat{H}(Y|X; X \in \mathcal{A}) = -C(p_0) - C(p_1) - \hat{T}(f_0^x || f^x) - \hat{T}(f_1^x || f^x) \qquad \alpha \approx 1 \tag{2.29}$$

$$\begin{aligned}
\hat{H}(Y|X; X \in \mathcal{A}) &\propto \frac{1}{n_0} \sum_{i=1}^{n_0} I[x_i \in \mathcal{A}] \Big(\frac{(n_0 - 1)\nu_k(x_i, X_0 \setminus x_i)^d}{n\nu_k(x_i, X \setminus x_i)^d} \Big)^{1-\alpha} \\
&+ \frac{1}{n_1} \sum_{i=1}^{n_1} I[x_i \in \mathcal{A}] \Big(\frac{(n_1 - 1)\nu_k(x_i, X_1 \setminus x_i)^d}{n\nu_k(x_i, X \setminus x_i)^d} \Big)^{1-\alpha} \\
&\propto \frac{1}{n_0} \sum_{i=1}^{n_0} I[x_i \in \mathcal{A}] \Big(\frac{(n_0 - 1)\nu_k(x_i, X_0 \setminus x_i)^d}{n\nu_k(x_i, X_1 \setminus x_i)^d} \Big)^{1-\alpha} \\
&+ \frac{1}{n_1} \sum_{i=1}^{n_1} I[x_i \in \mathcal{A}] \Big(\frac{(n_1 - 1)\nu_k(x_i, X_1 \setminus x_i)^d}{n\nu_k(x_i, X_0 \setminus x_i)^d} \Big)^{1-\alpha} \\
&\propto \frac{1}{n} \sum_{i=1}^{n} I[x_i \in \mathcal{A}] \Big(\frac{(n - 1)\nu_k(x_i, X_{y(x_i)} \setminus x_i)^d}{n\nu_k(x_i, X_{\neg y(x_i)} \setminus x_i)^d} \Big)^{1-\alpha}
\end{aligned}$$

As expected, an important aspect of the Informative Projection model is the selection of the area of applicability for each projection. Clearly, the risk increases drastically if the projection allocated to a certain neighborhood of the feature space does does not contain all the relevant features for accurate classification in that neighborhood. The active projection is chosen separately for each point in the training set, thus yielding the empirical area of applicability. There are several possible ways to generalize the selection for test data. A first option is to estimate the loss of each submodel for the sample, then select the projection/classifier pair that corresponds to the lowest loss, as shown in Section 2.3.4. An alternative is use the training set assignment in order to build an additional classifier that determines, for each sample, the submodel to which it should be assigned. Such a strategy is used in Section **??**.

By applying the previous estimator to all activation areas, we obtain an estimator for the entropy of the data classified with submodel j (projection c_j, classifier h_j):

$$\hat{H}(Y|c_j(X); X \in \mathcal{A}_j) \propto \frac{1}{n} \sum_{i=1}^{n} I[x_i \in \mathcal{A}_j] \Big(\frac{(n - 1)\nu_k(c_j(x_i), c_j(X_{y(x_i)}) \setminus c_j(x_i))^d}{n\nu_k(c_j(x_i), c_j(X_{\neg y(x_i)} \setminus x_i))^d} \Big)^{1-\alpha} \tag{2.30}$$

From (2.30) and using the fact that $I[x_i \in \mathcal{A}_j] = I[g(x_i) = j]$, we estimate the objective as

$$\min_{M \in \mathcal{M}_d} \sum_{c_j \in C} \frac{1}{n} \sum_{i=1}^{n} I[g(x_i) = j] \Big(\frac{(n - 1)\nu_k(c_j(x_i), \pi(X_{y(x_i)}) \setminus c_j(x_i))^d}{n\nu_k(c_j(x_i), c_j(X_{\neg y(x_i)} \setminus x_i))^d} \Big)^{1-\alpha} \tag{2.31}$$

Therefore, the contribution of each data point x to the objective corresponds to a distance ratio on the projection $c_{g(x)}$ where the class of the point is obtained with the highest confidence (data is separable in the neighborhood of the point). We start by computing the distance-based metric of each point on each projection of size up to d - there are d^* such projections.

This procedure yields an extended set of features L, which we name local entropy estimates:

$$L_{ij} = \Big(\frac{\nu_k(c_j(x_i), c_j(X_{y(x_i)}) \setminus c_j(x_i))}{\nu_k(c_j(x_i), c_j(X_{\neg y(x_i)} \setminus x_i))}\Big)^{d(1-\alpha)} \qquad \alpha \approx 1 \quad j \in \{1 \dots d^*\} \tag{2.32}$$

2.4.2 Generalized IPE models

We now substantially extend the Informative Projection Recovery (IPR) problem using a formalization applicable to any learning task for which a consistent estimator of the loss function exists. To solve the generalized IPR problem, we introduce the Regression-based Informative Projection Recovery (RIPR) algorithm. It is applicable to a broad variety of machine learning tasks such as semi-supervised classification, clustering, or regression, as well as to various generic machine learning algorithms that can be tailored to fit the problem framework. RIPR is useful when (1) There exist low-dimensional embeddings of data for which accurate models for the target tasks can be learned; (2) It is feasible to identify a low-dimensional model that can correctly process given queries. We formulate loss functions that can be used to implement IPR solutions for common learning problems, and we introduce additive estimators for them. We empirically show that RIPR can succeed in recovering the underlying structures. For synthetic data, it yields a very good recall of known informative projections. For real-world data, it reveals groups of features confirmed to be relevant by domain experts. We observe that low-dimensional RIPR can perform at least as well as models using learners from the same class, trained using all features in the data.

Assume we are given a dataset $X = \{x_1 \dots x_n\} \in \mathcal{X}^n$ where each sample $x_i \in \mathcal{X} \subseteq \mathbb{R}^a$ and a learning task on the space $\mathcal{X}$ with output in a space $\mathcal{Y}$ such as classification, clustering or regression. The learner for the task is selected from a class $\mathcal{T} = \{f : \mathcal{X} \to \mathcal{Y}\}$, where the risk for the class $\mathcal{T}$ is defined in terms of the loss ℓ as

$$\mathcal{R}(t, \mathcal{X}) = \mathbb{E}_{\mathcal{X}} \ell(x, t) \quad \forall t \in \mathcal{T}. \tag{2.33}$$

The optimal learner for the task is $t^* \stackrel{def}{=} \arg\min_{t \in \mathcal{T}} \mathcal{R}(t, \mathcal{X})$. We indicate by $t_{\{X\}}$ the learner from class $\mathcal{T}$ obtained by minimizing the empirical risk over the training set X.

$$t_{\{X\}} \stackrel{def}{=} \arg\min_{t \in \mathcal{T}} \hat{\mathcal{R}}(\mathcal{T}, X) = \arg\min_{t \in \mathcal{T}} \frac{1}{n} \sum_{i=1}^{n} \ell(x_i, t) \tag{2.34}$$

The class $\mathcal{M}$ of models constructed by our IPR framework is formalized as having a set C of projections with dimension at most d, a set T of learners and a selection function g:

$$\begin{aligned} \mathcal{M} = \{ & C = \{c_i : c_i \subseteq \{1 \dots a\}, |c_i| \leq d\}, \\ & T = \{t_i : t_i \in \mathcal{T}_{a_i}, t : c_i \to \mathcal{Y}, \quad \forall i \in \{1 \dots |C|\}\}, \\ & g \in \{f : \mathcal{X} \to \{1 \dots |C|\}\} \quad \}\,. \end{aligned} \tag{2.35}$$

$2^{\{1...a\}}$ contains all axis-aligned projections; the subset $C \subseteq 2^{\{1...a\}}$ in $\mathcal{M}$ contains only projections with at most d features. The value d is application-specific; usually 2 or 3, to permit users to view the projections. Function g selects the adequate projection c_j and its corresponding learner t_j to handle a given query x.

Based on this model, we derive a composite learner which combines the learners operating on the individual low-dimensional projections. The loss of this learner can be expressed in terms of the component losses: $t_{\mathcal{M}}(x) = t_j(c_j(x)), \quad \ell(x, t_{\mathcal{M}}) = \ell(c_j(x), t_j)$, where $g(x) = j$ represents the index of the learner which handles data point x and $c_j(x)$ is the projection of x onto c_j. Optimizing over the model class $\mathcal{M}$, the IPR problem for learning task $\mathcal{T}$ can be formulated as a minimization of the expected loss:

$$M^* = \arg\min_{\mathcal{M}} \mathbb{E}_{\mathcal{X}} \ell(c_{g(x)}(x), t_{g(x)}) \tag{2.36}$$

Thus, every sample data x_i can be dealt with by just one projection c_j. Recall that $g(x_i) = j$. We model this as a binary matrix B: $\quad B_{ij} = I[g(x_i) = j]$.
The minimizers of the risk and empirical risk are:

$$\begin{aligned} M^* &= \arg\min_{\mathcal{M}} \mathbb{E}_{\mathcal{X}} \sum_{j=1}^{|C|} I[g(x) = j] \ell(c_j(x), t_j) \\ \hat{M}^* &= \arg\min_{\mathcal{M}} \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{|C|} I[g(x_i) = j] \ell(c_j(x_i), t_j) \end{aligned} \tag{2.37}$$

Assume now that we can consistently estimate the loss of a task learner τ at each available sample, that is

$$\exists \hat{\ell} \text{ s.t.} \quad \forall x \in \mathcal{X}, t \in \mathcal{T} \quad plim_{n \to \infty} \hat{\ell}(x, \tau) = \ell(x, t) \tag{2.38}$$

Plugging (2.38) into (2.37) yields the final form used to obtain the estimated model:

$$\begin{aligned} \hat{M} &= \arg\min_{\mathcal{M}} \sum_{i=1}^{n} \sum_{j=1}^{|C|} I[g(x_i) = j] \hat{\ell}(c_j(x_i), t_i) \\ &= \arg\min_{\mathcal{M}, |C| \leq m} \sum_{i=1}^{n} \sum_{j=1}^{m} B_{ij} L_{ij}, \quad L_{ij} = \hat{\ell}(c_j(x_i), t_j) \end{aligned}$$

The loss estimators L_{ij} are computed for every data point on every subspace of up to the user-specified dimensionality d. B is learned through a regularized regression procedure that penalizes the number of projections $|C|$ used in the model. This translates to an ℓ_0 penalty on the number of non-zero columns in B, relaxed to ℓ_1. The ℓ_0 penalty is written as $I[|B_{\cdot,j}| \neq 0]$, while its relaxation is $||B||_{1,1}$.

$$\hat{B} = \arg\min_{B} ||T - L \odot B||_2^2 + \lambda \sum_{j=1}^{m} I[B_{\cdot,j} \neq 0] \tag{2.39}$$

where m is the number of candidate projections, $T_i \stackrel{def}{=} \min_j L_{ij}$ and the operator $\odot$ is defined as

$$\odot : \mathbb{R}^{n,m} \times \mathbb{R}^{n,m} \to \mathbb{R}^n, \quad (L \odot B)_i = \sum_{j=1}^{m} L_{ij} B_{ij}$$

The basic optimization procedure remains the same one shown in Algorithm 2.3.4 for all learning tasks, the key difference here is in the computation of the loss matrix L. The technique resembles the adaptive lasso. It gradually reduces the number of non-zero columns in B until convergence to a stable set of projections. As illustrated in Algorithm 2.3.4, the procedure uses the multiplier δ to gradually bias selection towards projections that not only perform well but also suit a large number of data points.

Next, we show how to compute the loss function for different learning tasks. When the aim is to find informative projections without knowing the class of learners to be used, we employ nonparametric estimators of loss. The performance of the algorithm will depend on their rates of convergence.

2.4.3 Semi-supervised classification IPEs

While the case of classification has been handled in the previous section, RIPR does allow an extension to semi-supervised classification. Consider a problem with labeled samples X_+ and X_- and unlabeled samples X_u, where each sample belongs to $\mathbb{R}^a$. The objective is to find a discriminator in a low-dimensional sub-space of features that correctly classifies the labeled samples and simultaneously allows substantial separation for unlabeled data, i.e., very few unlabeled data points remain between the clusters of data from different classes. We choose a loss function that penalizes unlabeled data according to how ambivalent they are to the label assigned. This is equivalent to considering all possible label assignments and assuming the most 'confident' one – the label with the lowest loss – for unlabeled data. The estimator for labeled data is the same as for supervised classification. The estimators use distances to the k^{th} nearest neighbors of each sample. The score for a projection is computed by using the same estimator for KL divergence between class distributions, to which we add a metric for unlabeled data which penalizes samples that are about equidistant from the point-clouds of each class: $\hat{\mathcal{R}}(X_u, t_c^k)$. We use the notation $c(X)$ to represent the projections of a set of data points X:

$$\begin{aligned}
\hat{\mathcal{R}}(X, t_c^k) = &\sum_{x \in X_+} \Big(\frac{\nu_{k+1}(c(x), c(X_+))}{\nu_k(c(x), c(X_-))} \Big)^{(1-\alpha)|c|} \\
&+ \sum_{x \in X_-} \Big(\frac{\nu_{k+1}(c(x), c(X_-))}{\nu_k(c(x), c(X_+))} \Big)^{(1-\alpha)|c|} \\
&+ \sum_{x \in X_u} \min \Big(\frac{\nu_k(c(x), c(X_-))}{\nu_k(c(x), c(X_+))}, \frac{\nu_k(c(x), c(X_+))}{\nu_k(c(x), c(X_-))} \Big)^{(1-\alpha)|c|}
\end{aligned}$$

In these learning tasks, typical convergence issues encountered with nearest-neighbor estimators can often be remedied thanks to low dimensionality of the projections.

2.4.4 Clustering IPEs

It is not always straightforward to devise additive point estimators of loss for clustering since some methods rely on global as well as local information. Distribution-based and centroid-based clustering fit models on the entire sets of data. This is an issue for the IPR problem because it is not known upfront how data should be assigned to the submodels. To go around this, we first learn a RIPR model for density-based clustering, and then cluster each projection using only data assignment provided by it. Of course, that is not required if density-based clustering is the method of choice. To solve IPR for density-based clustering, we consider the negative divergence, in the neighborhood of each sample, between the distribution from which the sample X is drawn and the uniform distribution on $\mathcal{X}$. Let U be the size n sample drawn uniformly from $\mathcal{X}$. Again, we use the nearest-neighbor estimator converging to the KL divergence. t_i^{clu} is some clustering technique such as k-means.

$$\hat{\mathcal{R}}_{clu}(c_i(x), t_i^{clu}) \to -KL(c_i(X)||c_i(U))$$
$$\hat{\ell}_{clu}(c_i(x), t_i^{clu}) \approx \Big(\frac{d(c_i(x), c_i(X))}{d(c_i(x), U)}\Big)^{|c_i|(1-\alpha)}$$

We now illustrate how RIPR clustering with k-means can improve over applying k-means to the entire set of features. Synthetic data used has 20 numeric features, and contains three Gaussian clusters on each of its informative projections. The informative projections comprise the following sets of feature indices: $\{17, 12\}$, $\{10, 20, 1\}$ and $\{4, 6, 9\}$. Clusterings obtained by k-means shown in those projections are depicted in the left part of Figure 2.4. The right part of it shows results obtained with RIPR. Every cluster is colored differently, with black representing data not assigned to that projection. The number of clusters is selected with cross-validation for both k-means and RIPR. The clustering obtained with k-means on all dimensions looks very noisy when projected on the actual informative features. The explanation is that the clustering might look correct in the 20-dimensional space, but when projected, it no longer makes sense. On the other hand, RIPR recovers the underlying model enabling the correct identification of the clusters. Naturally, recovery is only possible as long as the number of incoherent data points (that do not respect the low-dimensional model) stays below a certain level.

2.4.5 Regression IPEs

Our intent is to enable projection retrieval independently of the type of a regressor used, so the natural choice for a loss metric is a non-parametric estimator. We consider k-NN regression - computing the value at a query point by averaging the values at the k-nearest neighbors of the query. To factor in spatial placement, we weigh the values by their inverse distance from query, then estimate predicted value as normalized weighted average of the neighbor values.

$$\hat{\ell}_{reg}(c_i(x), t_i(c_i(x))) = (\hat{t}(c_i(x)) - y)^2 \qquad \hat{\ell}_{reg} \to 0$$
$$\hat{t}_i(c_i(x)) = \frac{\sum_{i=1}^{k} w_{(i)} y_{(i)}}{\sum_{i=1}^{k} w_{(i)}}, \qquad \text{where } w_{(i)} = \frac{1}{||x - x_{(i)}||_2}$$

Concerning the selection function, we identify two possible approaches. The first is to label each training data point according to the projections in the set used to solve it, then train a classifier using

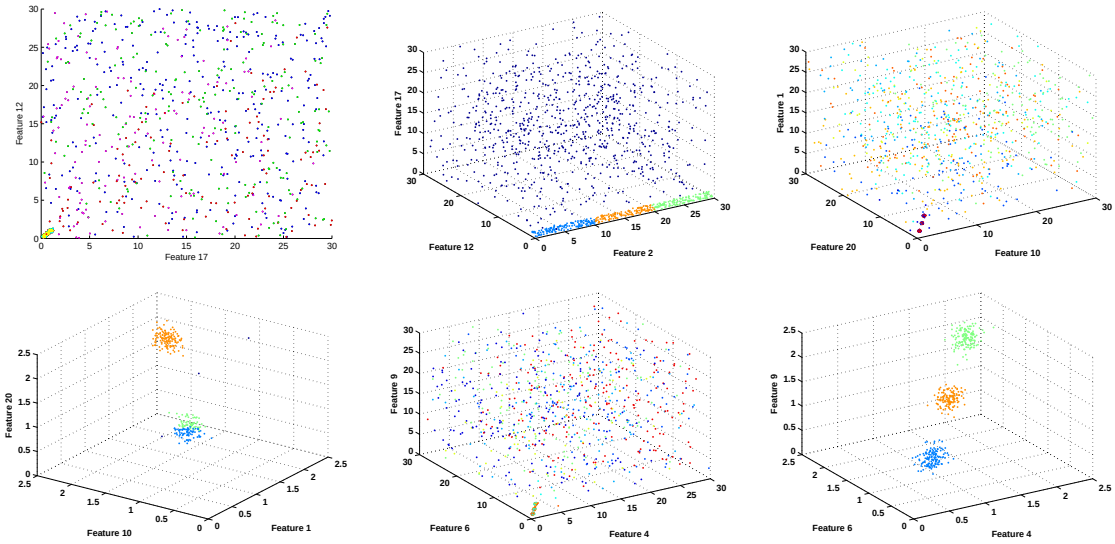


Figure 2.4: Projections of k-means clusters on the informative features and RIPR low-dimensional clusters induced from synthetic data.. Each cluster determined by the algorithm is shown in a different color.

these labels. The second is to simply estimate, based on the regressor accuracy at neighboring data, the probability that the regressor is appropriate for this data point. We opt for the latter because it avoids the issues with an additional training step and it is consistent with the regressors themselves in the usage of neighborhood information.

$$\hat{g}(x) = \underset{j \in \{1 \ldots |C|\}}{\arg\min} \frac{\sum_{i=1}^{k} w_{(i)} B_{(i)j}}{\sum_{i=1}^{k} w_{(i)}}, \qquad w_{(i)} = \frac{1}{||x - x_{(i)}||_2}$$

Interestingly, because of the consistency properties of the nearest-neighbor methods [25], the composite regressor is also consistent under the assumption of existence of embedding.

2.5 Properties of Informative Projection Ensembles

The flexibility of IPEs stems from the choice of the hypothesis class $\mathcal{H}$, as well as from the properties of the selection function. We first analyze the VC dimension and Rademacher complexity of the informative projection ensemble model class. These results apply in the case when the hypothesis classes considered for the classifiers and the selection function have finite VC dimension, for instance, for parametric models such as half-spaces. Next, we provide consistency results for IPEs which use kNN classifiers. In both cases (parametric and non-parametric), we compare the ensemble to a single classifier of that type and show that: (1) the complexity of the ensemble can be controlled as easily as the complexity of a single predictor; (2) an ensemble using sparse linear solvers has comparable sample complexity bounds to a single sparse linear solver; (3) an ensemble using kNN classifiers preserves the consistency properties of a single kNN classifier using all features.

2.5.1 VC Dimension of classification IPEs

In order to obtain bounds on the VC dimension of the informative projection ensemble, the VC dimension of the submodel classifiers as well as that of the selection function needs to be finite. To argue why the latter is required, assume that there were no bounds on the complexity of the selection function and k, the number of submodels, was greater or equal to $|\mathcal{Y}|$, the number of classes. Under these conditions, we could construct a model with $|\mathcal{Y}|$ classifiers such that $h_i(z) = v_i, \forall z \in \mathbb{R}^{a_i}$, where $v_i \in \mathcal{Y}$, in other words, each submodel assigns a specific label to its corresponding samples. Since there is no constraint on the complexity of g, any given set $\{(x_1, y_1) \dots (x_n, y_n)\}$, where n can be infinitely large, can be shattered by setting $g(x_j) = i$, where $y_j = v_i$. Thus, one cannot bound the VC dimension of IPEs without assuming a complexity bound on g.

We will impose a restriction that the selection function splits the feature space into convex subsets. We refer to the set of all the functions that fulfill this requirement as the *Convex Partition* class, formally introduced in Def. 2.5.1. Examples of selection functions that fulfill this criterion are linear classifiers, rectangular bounding boxes and Voronoi diagrams based on the Euclidean distance.

Definition 2.5.1. The Convex Partition class corresponding to $\mathcal{M}_{d,k}$ is defined as

$$\mathcal{G} = \{g : \mathcal{X} \subseteq \mathbb{R}^a \to \{1 \dots k\} \text{ s.t. } \forall X \subseteq \mathcal{X}, \forall i, j \in \{1 \dots k\}, i \neq j, \\ \mathcal{C}(\{x \in X; g(x) = i\}) \cap \mathcal{C}(\{x \in X \text{s.t. } g(x) = j\}) = \emptyset\}, \tag{2.40}$$

where $\mathcal{C}(A)$ represents the convex hull of the set A.

Lemma 2.5.2. *Under the following assumptions for a model from the class $\mathcal{M}_{d,k}$, which has base classifiers on $\mathbb{R}^{a_i}$, with*

(A1) The classifiers and the selection functions are limited in complexity, so for $v_i \stackrel{def}{=} VC(\mathcal{H}^{a_i}) < \infty$, $v_g \stackrel{def}{=} VC(g) < \infty$

(A2) The base classifiers, from the hypothesis classes $\mathcal{H}_r$, are affine invariant

(A3) The selection function is a convex partition, in other words $g \in \mathcal{G}$

It can be shown that $VC(\mathcal{M}) \geq \sum_{i=1}^{k} VC(\mathcal{H}^{a_i})$ and

$$VC(\mathcal{M}_k) \leq (\sum_{i=1}^{k} VC(\mathcal{H}^{a_i}) + v_g)^{2(k-1)} \tag{2.41}$$

Proof. First, we need to show that there exists a set of size $v \stackrel{def}{=} \sum_{i=1}^{k} v_k$ that can be shattered by functions from the model class $\mathcal{M}_{d,k}$. From (A1), we know that there exist sets $X_1 \dots X_k$, of size $v_1 \dots v_k$ respectively, which can be shattered by solvers from the base classes. These sets can be translated such that their convex hull do not overlap and according to (A2), base classifiers with the same shape as the original ones can be constructed. Since the sets are non-overlapping, the partition function simply assigns data points from each set to its corresponding classifier.

To prove the second part, let us first consider the case when $k = 2$. We will consider the growth function, that is the number of ways (configurations) in which the model assigns the n point sample set. Since h_1 has the VC dimension v_1, according to Sauer's Lemma [90], it can provide up to $\left(\frac{en}{v_1}\right)^{v_1}$ configurations, where n is the number of samples and e is the base of the

natural logarithm. Similarly, h_2 provides $\left(\frac{en}{v_2}\right)^{v_2}$ configurations. Now, we consider g and the placement of samples in either of the sets corresponding to h_1 or h_2. Since g also has a finite VC dimension and it is capable of fully splitting only v_g points, and, the set of n points can be split into, at most, $\left(\frac{en}{v_g}\right)^{v_g}$ configurations. By combining these ways of splitting the samples, we have an upper bound on the splitting capability of a 2-classifier model in terms of the number of configurations as

$$\Pi_{\mathcal{M}_2} \leq \left(\frac{en}{v_1}\right)^{v_1}\left(\frac{en}{v_2}\right)^{v_2}\left(\frac{en}{v_g}\right)^{v_g} \tag{2.42}$$

Since this is an upper bound growth function, we then obtain an upper bound on the VC dimension of the 2-classifier model by using the definition of the VC dimension:

$$\begin{aligned}
VC(\mathcal{M}_2) &= \max\left\{n > 0 | \Pi_{\mathcal{M}_2}(n) = 2^n\right\} && (2.43)\\
&\leq \max\left\{n > 0 | \Pi_{\mathcal{M}_2}(n) = 2^n\right\} && (2.44)\\
&\leq \max\left\{n > 0 | \left(\frac{en}{v_1}\right)^{v_1}\left(\frac{en}{v_2}\right)^{v_2}\left(\frac{en}{v_g}\right)^{v_g} = 2^n\right\} && (2.45)\\
&\leq \max\left\{n > 0 | v_1 \log_2\left(\frac{en}{v_1}\right) + v_2 \log_2\left(\frac{en}{v_2}\right) + v_g \log_2\left(\frac{en}{v_g}\right) = n\right\} && (2.46)\\
&\leq \max\left\{n > 0 | (v_1 + v_2 + v_g)\log_2(en) - \log_2(v_1) - \log_2(v_2) - \log_2(v_g) = n\right\} && (2.47)\\
&\leq \max\left\{n > 0 | B\ln(n) + A \geq n\right\} && (2.48)\\
&\leq \max\left\{n > 0 | Bq(n^{1/q} - 1) + A \geq n\right\} && (2.49)\\
&\leq \max\left\{n > 0 | 2B(\sqrt{n} - 1) + A \geq n\right\} && (2.50)\\
&\leq A - 2B + 2B^2 && (2.51)\\
&\leq \frac{v^2}{\ln^2 2} - \frac{v}{\ln 2} \qquad \text{where } v = v_1 + v_2 + v_g && (2.52)\\
&\leq \left(\frac{v}{\ln 2} - 1/4\right)^2 \leq \left(\frac{v}{\ln 2}\right)^2 \leq (v_1 + v_2 + v_g)^2 && (2.53)
\end{aligned}$$

where $\mathcal{M}_2$ is the 2-classifier IPE model class and Π is its growth function.

For a model using more than k classifiers, we can consider a multi-stage selection process, working as a decision list. At each stage i, the selection function g_i' picks either the current classifier h_i or an alternative model consisting of classifiers $h_1 \ldots h_{i-1}$. Because $g_i'(x) = 1$ iff $g(x) = i$ and $g_i'(x) = 0$ otherwise, we have that $VC(g_i') < VC(g)$. An upper bound on the VC dimension of such a multi-stage selection model would be an upper bound on the VC dimension of the original model. By applying 2.53, we have that $VC(\mathcal{M}_k) \leq (v_g + v_k + VC(\mathcal{M}_{k-1}))^2$. We can use this formula recursively, which yields the result.

□

Additionally, we can use the growth function as an upper bound for the Rademacher complexity

by applying the formula in (2.42).

$$\mathcal{R}(\mathcal{M}_{d,k}) \leq \Pi_{\mathcal{M}_{d,k}} \leq \Pi_{\mathcal{G}}^{k-1} \prod_{i=1}^{k} \Pi_{\mathcal{H}^{a_i}} \leq \left(\frac{en}{v_g}\right)^{kv_g} \prod_{i=1}^{k} \left(\frac{en}{v_i}\right)^{v_i} \tag{2.54}$$

The bound in Lemma 2.5.2 is not tight, at least in the case of linear classifiers and rectangular bounding-box classifiers. However, when the submodels are associated with interval classifiers and the partitioning of the samples can be done based on a single dimension, the upper bound matches the lower bound and we have that $VC(\mathcal{M}) = \sum_{i=1}^{k} VC(\mathcal{H}^{a_i})$.

2.5.2 Consistency of ensembles using nearest-neighbor classifiers

An alternative to ensembles of parametric classifiers, such as linear separators, is to use nearest-neighbor predictors. Asymptotic consistency of nearest-neighbor classification has been studied in [20, 24, 39, 99]. Rates of convergence and finite sample guarantees have been introduced in [19, 43, 67, 101, 106].

In order to study the consistency of neighbor-based IPE models, we start with a result obtained by Kulkarni et al. [67]. Assume that our samples $x_1 \dots x_n$ are iid from a distribution μ, with $\mathcal{X} \in \mathbb{R}^a$. The result requires that $(\mathcal{X}, \rho)$ be a metric space with totally bounded support $\mathcal{K}(\mu)$. We also assume that the output y is conditionally independent of all other outputs, given the sample x:

$$\forall i, \forall S \subseteq \mathcal{Y} \qquad \mathbb{P}(y_i \in S | x_1 \dots x_n, y_1 \dots y_i, y_{i+1} \dots y_n) = P(y_i \in S | x_i). \tag{2.55}$$

Given $X = x$, Y is drawn from a conditional distribution $F(y|X = x)$. The risk bounds we provide are based on squared error loss, with the conditional Bayes risk r_B^* and Bayes risk R_B^* defined as

$$r_B^*(x) = \mathbb{E}[||Y - Y^*(x)||^2 | X = x] \tag{2.56}$$

$$R_{B,\mu}^* = \int r^*(x)\mu(x)dx \tag{2.57}$$

We introduce the conditional mean and conditional variance of Y given $X = x$ as

$$m(x) = \mathbb{E}[Y|X = x] \tag{2.58}$$

$$\sigma^2(x) = \mathbb{E}[||Y||^2 | X = x] - ||m(x)||^2 \tag{2.59}$$

The Bayes estimator is actually the conditional mean, $Y^*(x) = m(x)$, which yields

$$r_B^* = \sigma(x)^2 \quad \text{and} \quad R_B^* = \mathbb{E}\sigma(x)^2.$$

An additional assumption is that $F(y|x)$ satisfies the following Lipschitz conditions: there exist $C_1, C_2 > 0$ and $0 < \alpha < 1$ such that $\forall x_1, x_2 \in X$,

$$||m(x_1) - m(x_2)|| \leq \sqrt{C_1}\rho(x_1, x_2)^{\alpha}, |\sigma^2(x_1) - \sigma^2(x_2)| \leq C_2\rho(x_1, x_2)^{2\alpha}. \tag{2.60}$$

For a sample x_0 with label y_0, $x^{(\kappa)}$ is the κ-nearest neighbor of x_0, selected from the set $\{x_1 \dots x_n\}$ with label $y^{(\kappa)}$. We refer to the loss and the risk of the nearest-neighbor predictor as

$$r_n(x_0) = \mathbb{E}_{x^{(\kappa)}}[||y_0 - y^{(\kappa)}||^2 | x_0] \quad \text{and} \quad R_n = \mathbb{E}r_n(x_0) \tag{2.61}$$

The limiting value of the risk is defined as $R_\infty \stackrel{def}{=} \lim_{n\to\infty} R_n$.

The risk also depends on the total volume of the support or, more specifically, on how easily it can be covered with ϵ-balls. To quantify this concept, we need the *metric covering radius*, $\mathcal{N}^{-1}(p, \mathcal{K})$, which is the smallest radius such that there exist p balls of this radius which cover the set.

$$\mathcal{N}^{-1}(p, \mathcal{K}) = inf\{\epsilon : \exists\ b_1 \dots b_p \in \mathcal{X} \text{ s.t. } \mathcal{K} \subseteq \bigcup_{i=1}^{p} \mathcal{B}(b_i, \epsilon)\}, \tag{2.62}$$

where $\mathcal{B}(b, \epsilon)$ is the ϵ-ball centered at b.

Theorem 2.5.3 (Kulkarni and Posner)**.** *Under the assumptions (2.55) and (2.60), for $\alpha < a/2$, we have that*

$$R_n \leq R_\infty + (C_1 + C_2)\frac{8^\alpha}{n}\sum_{i=1}^{n-1}[\mathcal{N}^{-1}(i, \mathcal{K}(\mu))]^{2\alpha} \tag{2.63}$$

$$R_n \leq R_\infty + (C_1 + C_2)\frac{a(8\mathcal{N}^{-1}(1, \mathcal{K}(\mu)))^\alpha}{a - 2\alpha} n^{-2\alpha/a} \tag{2.64}$$

$$R_\infty = 2R_B^* \tag{2.65}$$

It is possible to provide a finite sample bound, in terms of the Bayes optimal risk, on the risk of IPE models which use nearest-neighbor predictors. We will first define the *activation support* of a sub model as the subset of the input space where the sub model is used for classification.

Definition 2.5.4. The activation support of a submodel i of an ensemble $\mathcal{M}_{d,k}$, is defined as

$$\mathcal{A}_i \stackrel{def}{=} \{x : x \in \mathcal{X}, g(x) = i\} \tag{2.66}$$

We introduce the quantities n_i which specify the number of samples in each submodel i. Given the selection function $g_{n,i}$, learned based on a finite sample, we have the corresponding activation support $\mathcal{A}_{n,i}$. Thus, the loss and the risk for the submodel i with support $\mathcal{A}_{n,i}$ are

$$r_{n,i}(x_0) = \mathbb{E}_{x^{(\kappa)}}[||y_0 - y^{(\kappa)}||^2 | x_0] \quad \text{and} \quad R_{n,i} = \int_{x\in\mathcal{A}_{n,i}} r_{n,i}(x) d\mu x, \tag{2.67}$$

where submodel i uses a κ-nearest-neighbor classifier and $x^{(k)}$ is the neighbor of x_0 on the projection c_i of the IPE model, while $y^{(k)}$ is its label. The risk of a model $M \in \mathcal{M}_{d,k}$ will then be

$$R_n = \sum_{i=1}^{k} \frac{Vol(\mathcal{A}_i)}{Vol(\mathcal{X})} R_{n,i} = \sum_{i=1}^{k} \frac{Vol(\mathcal{A}_i)}{Vol(\mathcal{X})} \int_{x\in\mathcal{A}_{n,i}} r_{n,i}(x) d\mu x \tag{2.68}$$

In order to obtain bounds on the risk we have to consider the predictive capabilities of the informative projection of a submodel on its activation support. Specifically, we introduce the notions of *predictive feature subset* and *sparsity pattern* which delineate the features that are required to appropriately predict the output variable Y when X belongs to a compact set or neighborhood.

Definition 2.5.5. For a given input X with features $\{1 \dots a\}$, drawn from a set $\mathcal{A} \in \mathcal{X}$, a set $c \subseteq \{1 \dots a\}$ is a predictive feature subset for the output variable Y if the variables X_c contain all the predictive information for Y. Let $X_{\neg c}$ be all the features of X that are not in c. If c is a predictive feature subset, then Y is independent of $X_{\neg c}$ given X_c and that X takes values in $\mathcal{A}$.

$$Y \perp X_{\neg c} | X_c, X \in \mathcal{A} \tag{2.69}$$

$$\forall i, \forall V \subseteq \mathcal{Y} \qquad P(Y \in V | X, X \in \mathcal{A}) = P(y_i \in V | X_c, X \in \mathcal{A}). \tag{2.70}$$

Definition 2.5.6. The sparsity pattern for the prediction of a variable Y based on a variable X taking values in a set $\mathcal{A}$ is defined as the predictive feature subset of minimal size.

$$sp_{\mathcal{A}} \stackrel{def}{=} \underset{c \subseteq \{1 \dots a\}}{\arg\min} |c| \qquad \text{s.t.} \quad Y \perp X_{\neg c} | X_c, X \in \mathcal{A} \tag{2.71}$$

In order for the risk of the IPE model $M = (C, H, g)$ to approach the Bayes risk, the submodels must jointly encapsulate all the predictive information. For each neighborhood in the feature set, at least one of the submodels contains all the variables that encapsulate information about the samples in that neighborhood. In order words, the sparsity pattern of each neighborhood needs to be included in at least one of the projections of the model.

$$\forall \mathcal{B} = \{(x_c, \epsilon) | 0 < \epsilon, \epsilon \text{ is small}\} \subset \mathcal{X}, \ \exists\, c \in C \text{ s.t. } sp_{\mathcal{B}} \subseteq c \tag{2.72}$$

This assumption can be easily fulfilled by including a classifier that uses all the features as one of the candidate sub models, though a higher submodel dimensionality will result in a lower rate.

Lemma 2.5.7 (Consistency of IPE models using nearest-neighbor predictors)**.** *Assume that (2.72) holds for IPE models using nearest-neighbor classifiers, with a selection function that always picks the submodel with the lowest estimated risk also computed based on nearest-neighbors.*

If the Lipschitz conditions over $F(y|x)$ hold for each of the submodels with constants $C_{1,i}, C_{2,i}$, then, for a sufficiently large n, where $|\mathcal{A}_{n,i}| = n_i$ and $\sum_{i=1}^{k} n_i = n$ we have that

$$R_n \leq R_\infty + \sum_{i=1}^{k} \Big(\frac{Vol(\mathcal{A}_i)}{Vol(\mathcal{X})} (C_{1,i} + C_{2,i}) \frac{a_i (8 \mathcal{N}^{-1}(1, \mathcal{A}_{n,i}))^\alpha}{a_i - 2\alpha} |\mathcal{A}_{n,i}|^{-2\alpha/a_i} \Big). \tag{2.73}$$

Proof. The risk of the IPE model depends on the risks of the individual classifiers on their respective activation supports. In some neighborhoods within the activation support of the submodel i, the set c_i is a predictive subset for Y. In other words, over these neighborhoods, the Bayes risk of only the features in c_i, R^*_{B,c_i}, is the same as the Bayes risk of all the features, R^*_B.

$$\mathbb{B}_i \stackrel{def}{=} \{\mathcal{B} \subseteq \mathcal{A}_i; Y \perp X_{\neg c_i} | X_{c_i}, X \in \mathcal{B}\} \tag{2.74}$$

$$\mathcal{B}_i \stackrel{def}{=} \bigcup_{\mathcal{B} \in \mathbb{B}_i} \mathcal{B} \qquad \bar{\mathcal{B}}_i = \mathcal{A}_i \setminus \mathcal{B}_i \tag{2.75}$$

$$R^*_B(\mathcal{B}_i) = R^*_{B,c_i}(\mathcal{B}_i) \qquad (c_i \text{ is a predictive subset over } \mathcal{B}_i) \tag{2.76}$$

The set $\mathcal{B}_i$ covers the neighborhoods of $\mathcal{A}_i$ where c_i is a predictive subset, while $\bar{\mathcal{B}}_i$ covers everything else in $\mathcal{A}_i$. Clearly, the risk of submodel i decomposes over the two sets.

$$R_{n,i} = \frac{Vol(\mathcal{B}_i)}{Vol(\mathcal{A}_i)} R_{n,i}(\mathcal{B}_i) + \frac{Vol(\bar{\mathcal{B}}_i)}{Vol(\mathcal{A}_i)} R_{n,i}(\bar{\mathcal{B}}_i) \tag{2.77}$$

We now point out that if $F(y|x)$ is Lipschitz continuous with constants $C_{1,i}$ and $C_{2,i}$ over $\mathcal{B}_i$, the conditions for Theorem 2.5.3 are met. Thus, we can apply the bound in equation (2.64) as follows

$$R_{n,i}(\mathcal{B}_i) - R^*_{B,c_i}(\mathcal{B}_i) = R_{n,i}(\mathcal{B}_i) - R^*_B(\mathcal{B}_i) \leq (C_{1,i} + C_{2,i}) \frac{a_i (8\mathcal{N}^{-1}(1, \mathcal{B}_i))^\alpha}{a_i - 2\alpha} |\mathcal{B}_i|^{-2\alpha/a_i} \quad (2.78)$$

This leaves the risk on the part of $\mathcal{A}_i$ for which c_i is not a predictive subset, $R_{n,i}(\bar{\mathcal{B}}_i)$. However, according to assumption (2.72), for each neighborhood in $\bar{\mathcal{B}}_i$, there exists a least one classifier j in the model for which c_i is a predictive subset. For classifier j, the risk converges to the Bayesian risk at the rate in (2.64). Function g selects the submodel based on the minimum risk estimator, which means that, for each neighborhood, the selection risk (and thus the risk of misclassification) is also subject to the bound. Let $\bar{\mathbb{B}}_i$ be some disjoint set of neighborhoods of $\bar{\mathcal{B}}_i$. Then for each $B \in \bar{\mathbb{B}}_i$, we have that

$$R_n(B) - R^*_B(B) \leq \min_{j \in \{1 \ldots k\}} (C_{1,j} + C_{2,j}) \frac{a_j (8\mathcal{N}^{-1}(1, B))^\alpha}{a_j - 2\alpha} |B|^{-2\alpha/a_j} \quad (2.79)$$

$$\leq (C_{1,i} + C_{2,i}) \frac{a_i (8\mathcal{N}^{-1}(1, B))^\alpha}{a_i - 2\alpha} |B|^{-2\alpha/a_i} \quad (2.80)$$

Summing over the previous equation yields the risk bound over $\bar{\mathcal{B}}_i$

$$R_{n,i}(\bar{\mathcal{B}}_i) - R^*_B(\bar{\mathcal{B}}_i) \leq \sum_{B \in \bar{\mathbb{B}}_i} (C_{1,i} + C_{2,i}) \frac{a_i (8\mathcal{N}^{-1}(1, B))^\alpha}{a_i - 2\alpha} |B|^{-2\alpha/a_i} \quad (2.81)$$

$$\leq (C_{1,i} + C_{2,i}) \frac{a_i 8^\alpha}{a_i - 2\alpha} \Big(\sum_{B \in \bar{\mathbb{B}}_i} \mathcal{N}^{-1}(1, B)^\alpha |B|^{-2\alpha/a_i} \Big) \quad (2.82)$$

$$\leq (C_{1,i} + C_{2,i}) \frac{a_i 8^\alpha}{a_i - 2\alpha} \Big(\mathcal{N}^{-1}(1, |\bar{\mathcal{B}}_i|) \Big)^\alpha \Big(\sum_{B \in \mathbb{B}_i} |B|^{-2\alpha/a_i} \Big) \quad (2.83)$$

$$\leq (C_{1,i} + C_{2,i}) \frac{a_i 8^\alpha}{a_i - 2\alpha} \Big(\mathcal{N}^{-1}(1, |\bar{\mathcal{B}}_i|) \Big)^\alpha \Big(\sum_{B \in \mathbb{B}_i} |B|^{-2\alpha/a_i} \Big) \quad (2.84)$$

$$\leq (C_{1,i} + C_{2,i}) \frac{a_i 8^\alpha}{a_i - 2\alpha} \Big(\mathcal{N}^{-1}(1, |\bar{\mathcal{B}}_i|) \Big)^\alpha |\bar{\mathcal{B}}_i|^{-2\alpha/a_i} \quad (2.85)$$

The last equation holds since (2.84) is true for every partition $\bar{\mathbb{B}}_i$ of $\bar{\mathcal{B}}_i$, is is also true for the case when $\mathcal{B}_i$ contains a single set. By combining (2.78) and (2.85), which offer bounds of the risk over subsets of $\mathcal{A}_i$, we obtain (2.73). □

2.6 Experiments

2.6.1 Comparison of classification IPEs

Table 2.1 shows the performance of IPEs using K-NN classifiers and standard K-NN on UCI datasets. We present the performance of IPEs trained through each of the three techniques. We also test the methods on a Cell dataset containing a set of measurements such as the area and perimeter of the cell and a label which specifies whether the cell has been subjected to treatment or not. In the

Vowel dataset, a nearest-neighbor approach works exceptionally well, even beating random forests (0.94 accuracy), which is an indication that all features are jointly relevant. For some d lower than the number of features, our models pick projections of only one feature, but if there is no such limitation, the space of all the features is selected as informative. For the MiniBOONE dataset, also from the UCI repository, none of the IPE learning methods were able to extract an ensemble of low-dimensional k-NN classifiers that performs as well as the fully dimensional k-NN. This is mainly because we explored models with a very small number of features and, on the other hand, the dataset contains sufficient samples for k-NN to achieve high accuracy.

Table 2.1: Accuracy of K-NN and K-NN IPEs

Dataset	***KNN***	***IPE*** (optimal)	***IPE*** (2-stage)	***IPE*** (greedy)
Breast Cancer Wis	**0.8415**	**0.8415**	0.8275	0.8275
Breast Tissue	1.0000	1.0000	1.0000	1.0000
Cell	0.7072	**0.7640**	**0.7640**	0.7260
MiniBOONE	**0.7896**	0.7728	0.7396	0.7268
Spam	0.7680	0.7680	0.7680	0.7680
Vowel	0.9839	0.9839	0.9839	0.9839

We also compared the performance IPEs with k-NN on larger datasets, with multi-class outputs. The results are reported in Table 2.2. We also compared against other feature selection methods (PCA, lasso, forward feature selection and backward feature selection). For each dataset, we only report the top performing result in the column titled FS + k-NN.

Experiments on artificial data, showing that our procedures are effective are recovering informative projections, are shown in the Appendix. This essentially shows that, when low-dimensional structures exist, our procedures will recover them, which means that the conditions necessary for Lemma 2.5.3 are satisfied and accuracy can be improved.

Table 2.2: Accuracy (%) of k-NN models on letter data.

Dataset	Features	Samples	Classes	k-NN	FS + k-NN	IPE Greedy
Chars74k	85	3410	62	34.31	35.2	**35.78**
G50C	50	550	2	70	76	**84**
Letter	16	16000	26	94.37	95.08	**95.55**
MNIST	784	60000	10	60.11	60.11	**62.36**
USPS	256	11000	10	96.2	86.4	**96.8**

2.6.2 RIPR framework applied to clustering

RIPR can be wrapped around virtually any existing clustering, regression, or classification algorithm, maintaining their high performance while satisfying the requirement of working with only a few dimensions of data at a time. Below we show that RIPR combined with k-means, which

we informally call Ripped k-means, performs better than the standard k-means by leveraging the low-dimensional structure in data.

We trained RIPR and k-means models and evaluated their performance on datasets from the UCI repository. Meta-parameters for both methods were optimized via cross-validation. The data was scaled to $[0, 1)$ before clustering. We used distortion as the evaluation metric as it is native to k-means. We opt against using Rand index since in its standard form it requires the actual labels that are unavailable in most real-world clustering data sets. As shown in Table 2.3, the distortion results for the RIPR model are better than for plain k-means.

Table 2.3: Results of clustering on real-world datasets.

UCI	*Avg Dist RIPR*	*Avg Dist k-means*	*LogVol RIPR*	*LogVol k-means*
Seeds	16	107	7.68	9.70
Libras	9	265	-5.80	7.26
Boone	125	1.15e6	240.00	248.15
Cell	40,877	8.18e6	54.69	67.68
Concrete	1,370	55,594	49.24	52.75

The resulting cluster dimensionalities vary as well, which is why we also considered another metric of success: the volume of the resulting clusters measured in full feature space. This comparison is fair because the volumes are computed in the same dimensionality. For k-means, we approximated the volume of each cluster by its enclosing hyper-ellipsoid. For RIPR, the approximation for each cluster used its enclosing cylinder, the base of which was the ellipsoid corresponding to the actual identified low-dimensional cluster. This comparison is also provided in Table 2.3. It is apparent that RIPR obtains slightly more compact models than k-means, but has the advantage that only a fraction of the features are used by it. The total number of centroids is roughly the same for k-means and RIPR, so the difference in volume is genuinely due to the improvement fidelity of clustering.

We present the RIPR clustering models obtained from two datasets from the UCI repository to demonstrate how patterns in data can be mined with our approach. Figure 2.5 shows the model recovered from the Seeds dataset. The clustering that RIPR constructs uses the size and shape of seeds to achieve their placement into three categories, clearly visually separated in the figure. The separation according to their aspect ratio is something that one might intuitively expect.

Figure 2.6 shows the two informative projections mined from the Concrete dataset. Here, different concrete mixtures are grouped by their content. While the first projection generates clusters according to the high/low contents of cement and high/low contents furnace residue, the second projection singles out the mixtures that have (1) No fly ash, (2) No furnace residue or (3) Equal amounts of each. The clusters seem to capture what an experimenter might manually label.

2.6.3 RIPR framework applied to regression

As with clustering, RIPR regression is meant to complement existing regression algorithms. We exemplify by enhancing SVM and comparing it with the standard SVM. The synthetic data we use contains 20 features generated uniformly with Gaussian noise. The first feature and q pairs of other

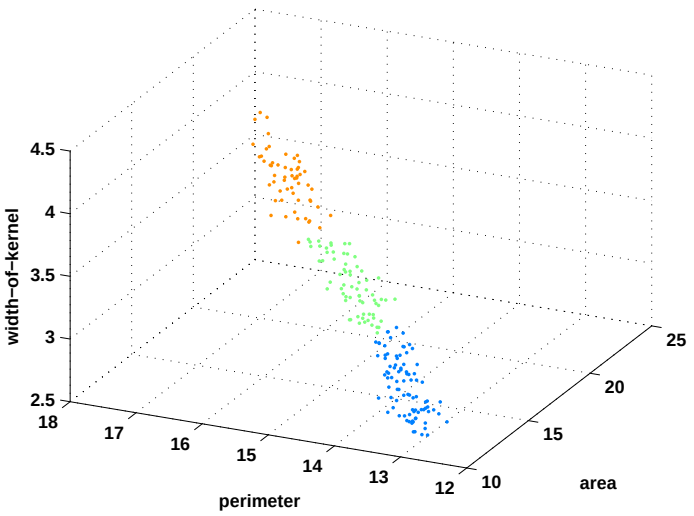


Figure 2.5: Clusters from the Seeds dataset

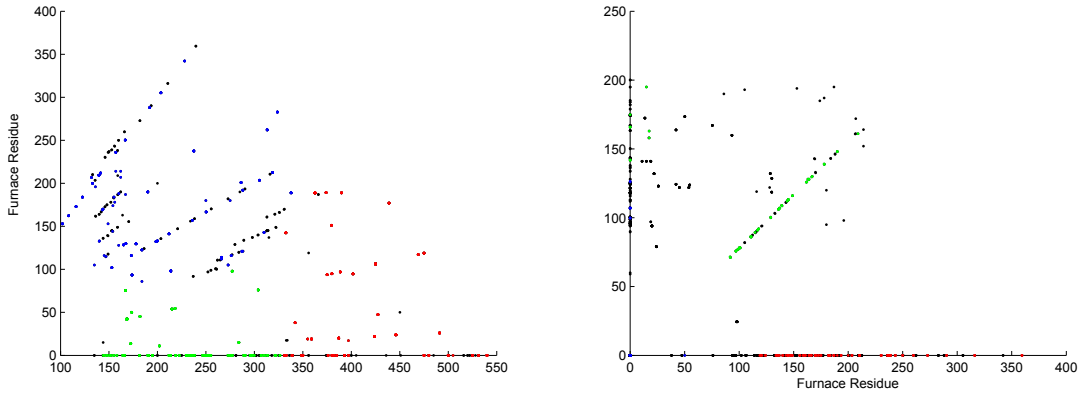


Figure 2.6: Clusters induced from the Concrete dataset.

features (j_1, j_2) determine the regression function as follows:

$$f(x) = \sum_{j=1}^{q} I[j <= x_1 < j+1] f_j(x_{j_1}, x_{j_2}) + \epsilon \qquad \forall j \in 1 \ldots q$$

Table 2.4 shows that 'Ripped Kernel SVM' achieves better accuracy that Kernel SVM trained on all features. The explanation is that RIPR actively identifies and ignores noisy features and useless data while learning each submodel. Additionally, we tested whether the underlying projections are correctly recovered by computing precision and recall metrics. Recall is always high, while precision is high as long as the projections do not overlap significantly in the feature space. It is because partially-informative projections can also be recovered if feature overlaps exist. This behavior can be controlled by adjusting the extent of regularization.

Table 2.4: RIPR SVM and standard SVM compared on synthetic data

IP #	***2***	***3***	***5***	***7***	***10***	***2***	***3***	***5***	***7***	***10***
	MSE RIPR					***MSE SVM***				
0	0.05	0.27	0.05	0.02	0.23	0.27	1.16	0.11	0.1	0.43
100	0.42	1.26	0.34	1.45	0.52	0.8	1.02	0.6	2.99	0.94
200	0.5	0.86	0.8	0.33	0.99	0.97	1.27	0.29	0.68	1.44
400	0.63	1.47	1.34	1.61	0.11	0.4	1.26	1.64	1.71	0.08
800	0.69	0.38	1.12	0.68	1.1	0.52	0.06	0.91	0.9	1.16
	RIPR Precision for IPR					***RIPR Recall for IPR***				
0	1	1	0.4	0.43	0.3	0.67	1	0.67	1	1
100	1	0.67	0.6	0.43	0.2	0.67	0.67	1	1	0.67
200	1	1	0.6	0.43	0.3	0.67	1	1	1	1
400	1	1	0.6	0.43	0.1	0.67	1	1	1	0.33
800	1	0.67	0.4	0.29	0.3	0.67	0.67	0.67	0.67	1

2.7 Discussion of IPE learning efficiency

2.7.1 Computational complexity of IPE learning

The IPE learning methods rely on the construction of the loss matrix, that is estimating the loss for every data point, for every combination of features. For nonparametric loss functions, this requires finding the k$^{\text{th}}$ nearest neighbors. We use k-d trees [42] for every projection of size d. The time required to build the tree is $O(dn \log n)$, where n is number of training samples. The time needed to find the neighbors of one sample point is $O(\log n)$. Thus, for all $m = O(a^d)$ candidate projections of up to size d, the total time required to compute the loss matrix is $O(m(d+1)n \log n)$, or, in terms of the number of features a, $O(da^d n \log n)$.

The ILP uses $m(n+1)$ variables, one integer variable for each element in the matrix and an additional variable for each column in the matrix. It contains n equality constraints, each summing over m variables. There are also m inequality constraints over n+1 variables each, one inequality constraint summing over m variables, $m * n$ inequality constraints over two variables each and $m * (n+1)$ inequality constraints over one variable each. Overall, the ILP is quite complex and, since it yields an exact solution, in the worst case scenario it must consider all possible assignments. In total, there are $\binom{m}{k}$ ways of selecting the sub-models. For each configuration, each of the n samples can be assigned to one of k projections, which yields on the order of k^n possibilities. Therefore, there are $\binom{m}{k}k^n$ assignments to be considered in the worst case. While the average run time encountered in practice is a lot more reasonable, as shown in the experiments, the ILP remains the slowest of the methods, albeit conferring the best solution. To reduce runtime, it is possible to relax the ILP into an LP and then apply rounding. However, we have observed that, for artificial data, the solutions obtained in this way are less precise (i.e. further from the underlying low-dimensional patterns) than the ones obtained with the two-step convex procedure.

For the complexity of Algorithm 2.3.4, we use the bounds in [5]. The optimization is over a matrix of size $N = mn$. Computing the values and derivatives of the objective and the constraints requires $M = O(mn)$ operations. The upper and lower bound on the number of operations needed to obtain a solution ϵ away from the optimum are $O(NM)ln(\frac{1}{\epsilon})$ and $O(N(N^3+M))ln(\frac{1}{\epsilon})$

respectively. Thus, the worst case runtime for the optimization is $O\Big(a^{4d}n^4\Big)\ln\Big(\frac{1}{\epsilon}\Big)$. Although the complexity increases exponentially with d, for the applications we consider d is typically 2, resulting in a worst case runtime of

$$O\Big(a^8n^4\Big)\ln\Big(\frac{1}{\epsilon}\Big)+O\Big(a^2n\log n\Big). \tag{2.86}$$

In the adaptive lasso procedure, we can discount projections that are not informative for any of the sample data points so the dimensionality of the optimization problem is reduced from $n\times m$ to $n\times\min(m,n)$. When $a^d>n$, the runtime depends largely on n (2.87), which is beneficial for datasets that are underdetermined (small sample size but large number of features) – a frequent case in computational biology, for instance. In this case, the worst case runtime is

$$O\Big(n^8\Big)\ln\Big(\frac{1}{\epsilon}\Big)+O\Big(dn^2\log n\Big). \tag{2.87}$$

The greedy selection procedure is the fastest and most scalable. At each of k iterations, we must compute the loss decrease due to each projection, which can be done in $O(mn)$. Therefore, the runtime of the greedy method is just $O(mnk)$, making it the most advantageous procedure to use when the sample size is high. This is the case because, if there are enough training samples, the greedy procedure

2.7.2 Comparison of methods in terms of running time

We have evaluated the running time of each of the three procedures, varying the number of features and the number of samples. The data was generated artificially, to allow control of the experiment. There are two underlying informative projections, each affecting roughly half of the data. The IPE learning procedures we used for this experiment are Matlab prototypes. We used the cvx package for the convex optimization. We also used a python script to generate the ILP from the loss matrix. The ILP was then solved using the publicly available tool Lpsolve. We trained IPE models, timing each matrix selection procedure. The experiments were ran on a MacBook with 2.6 GHz Intel i5 and 8 GB 1600 MHz DDR3 and the times, averaged over 5 runs, are reported in Table 2.5. It must be noted that the purpose of this experiment is to provide a comparison between the three methods, rather than to establish a baseline on the size/complexity of the datasets for which these techniques can be applied. Providing a faster implementation and running the experiments on a machine with more computational power is conceptually possible, but outside the scope of the thesis.

According to the table, the greedy method is fast enough for all cases and, in fact, scales up to much larger datasets. The precision and recall of the greedy method for this simulated data are perfect as long as the noisy samples represents less than 50% of the data. To compare, the ILP recovers the low-dimensional patterns even when the noise represents 75% of the data. The RIPR method provides a more robust solution that the greedy version and faster than the ILP. The ILP method only works for tiny datasets, or in the case that feature selection and/or subsampling have already been performed. RIPR only works for small datasets on the order of tens/hundreds of features and up to thousands of samples. The applicability of RIPR can be widened by combining it with other feature selection methods and by reducing the set of candidate projections to only the ones using informative features. We have applied such optimizations to obtain the 3-D projections presented in Chapter 4.

Table 2.5: Running time (seconds) of selection matrix learning procedures.

Features	**100 samples**			**1000 samples**		
	ILP	RIPR	greedy	ILP	RIPR	greedy
5	0.2955	2.4443	0.0051	13.2807	16.1068	0.0405
10	1.4064	3.0487	0.0084	233.6535	21.6976	0.0589
20	10.74	6.0532	0.0155	1637.5	56.0764	0.1217
50	2299.8	36.1182	0.0630	$>$ 2hrs	587.5157	0.6433
100	$>$ 2hrs	281.103	0.2317	$>$ 2hrs	3926	2.3132

Chapter 3

Extensions to the RIPR Framework

3.1 Learning IPEs in an active learning setting

We adapt standard active learning sample selection heuristics to work directly with the RIPR models and introduce new heuristics that find unlabeled data worth expert evaluation based on their appearance in low-dimensional subspaces. We also modify the RIPR optimization to find contradictory patterns in data, which is useful in the active learning context when the intent is to prompt the domain experts into disambiguating samples which are difficult to classify automatically. This method is part of the annotation system which doctors used to label a subset of alerts as real or artifactual.

3.1.1 Overview of active learning with dimensionality reduction

We introduce an approach which recovers informative projections in the more challenging *active learning* setting. Our framework selects samples to be labeled based on the relevant dimensions of the current classification model, trained on previously annotated data. The effort is thus shifted to labeling samples that *specifically target performance improvement* for the class of low-dimensional models we are considering. An important outcome is that *high accuracy is achieved faster* than with standard sampling techniques, reducing the data annotation effort exerted by domain experts. An added benefit is that the *compact models are available to experts during labeling*, in addition to the full-featured data. The informative projections[1] highlight structure that experts should be aware of during the labeling process, which helps prevent user errors, as illustrated in a case study. Moreover, our active learning framework selects the most controversial, most informative and/or most uncertain data yet unlabeled (depending on the selected sampling technique), presenting it to the human experts in an intuitive and comprehensible manner, typically using 2 or 3-dimensional projections, which further simplifies the annotation process.

We have previously formulated *Informative Projection Retrieval* (IPR) as the problem of finding query-specific models using ensembles of classifiers trained on small subsets of features. The *Regression for Informative Projection Retrieval* (RIPR) algorithm [35] provides a solution to this problem in the form of compact models consisting of low-dimensional projections. We will call

[1] In this chapter, the focus is exclusively on axis aligned projections (sets of features), since domain experts have no difficulty interpreting them.

them *RIPR models*. This chapter presents a framework, called *ActiveRIPR*, which enables active selection of yet unlabeled data which specifically targets the construction of accurate RIPR models. For this purpose, we adapt established active learning query criteria to the IPR task. Our contributions are: (i) we solve the Informative Projection Retrieval problem in the active learning setting; (ii) we compare various querying strategies under different noise models; (iii) we apply ActiveRIPR to alert adjudication leading to considerable reduction of labeling effort.

Active learning is an intensely-studied branch of machine learning, with many successful sampling methods currently available [94]. Adding to established methods such as uncertainty sampling, information gain and query by committee, are recent developments such as the Kernel Query by Committee [45], sampling based on mutual information [51] and the use of importance weighting in a scheme which works with general loss functions to correct sampling bias [7]. Our sample selection criteria take into account the utility of the samples for each of the projections in our ensemble. Previous work considering ensembles include the approach of Körner and Wrobel [65], who compare different approaches that use ensemble disagreement adapted to the problem of multiclass learning and show that margins are the best performing for the purpose. Donmez et al. [27] consider the existence of an ensemble of labeling sources and investigate how to jointly learn their accuracy and obtain the most informative labels while minimizing labeling effort. Examples of structured prediction being enhanced by active learning include the work by Culotta and McCallum [22], introducing a selective sampling framework which considers not just the number of samples requested for active learning in structured prediction, but also the effort required in labeling them. Liang et al. [74] also investigate the interplay between structured learning and model enhancement using contextual features, using unlabeled data to shift predictive power between models. The algorithm they present interleaves labeling features and samples, which improves the active learning performance. Bilgic proposes dynamic dimensionality reduction for active learning [8], a method which, during the query selection process, performs PCA on the data, selecting the features with the largest eigenvalues and performing L_2 regularization on them. There are some notable differences to their approach, the most important of which is that, in their setup, the allowed number of features is increased as more samples become available. The method of Raghavan et al. [84] directly incorporates human feedback in the feature selection procedure through feature weighting, while Rashidi and Cook [85] introduce a method that reduces the effort needed for labeling by requesting, in each iteration, labels for all samples matching a rule.

A notable method focused on improving models from a specific class is Tong and Koller's approach [104] to Bayesian Network structure learning. Their interventional active learning setup is different from the one we are considering in that the learner has the freedom of requesting samples rather than selecting them from a batch.

Our main improvement over related work is that our framework is designed to train accurate intelligible models which domain experts can use during the labeling process. ActiveRIPR not only queries the samples which improve model accuracy, but also considers human involvement and targets compact, user-friendly models, such that, at every step in the active learning procedure, the experts can consult the current informative model. Access to this visualizable model can make expert adjudication faster and more reliable. Also, clinicians can observe the classification model in action and be better prepared to decide whether it is mature enough for deployment.

3.1.2 Active informative projection recovery framework

Active learning iteratively selects samples for labeling until the model meets some accuracy criteria. Assume now that, at iteration k, the samples X_ℓ^k are labeled as Y_ℓ^k and the samples X_u^k are available for labeling. Also let the RIPR model built so far be M^k, with its components C^k, T^k and g^k. The problem of selecting samples for IPR is reduced to finding a scoring function $s : \mathcal{M} \times \mathcal{X} \to \mathbb{R}$, used to select the next sample to be labeled:

$$x^{k+1} = \arg\min_{x \in X_u^k} s(M^k, x)$$

The expected error of a model $M^k = \{C^k, T^k, g^k\}$ is

$$Err(M^k) = \mathbb{E}_{x \in \mathcal{X}}[I(t^k_{g^k(x)}(c^k_{g^k(x)}(x)) \neq y)]$$

We use the notation M_s^k to refer to a model obtained after k iterations of labeling, using the scoring function s. If the labeled samples are picked adequately, the training error will decrease (or at least not increase) with each iteration: $Err(M_s^{k+1}) \leq Err(M_s^k)$. Given the maximum acceptable error ϵ, and a set $\mathcal{S}$ of scoring functions, selecting the optimal strategy can be expressed as follows:

$$s^* = \arg\min_{s \in \mathcal{S}} \min_k \{k \text{ s.t. } Err(M_s^k) \leq \epsilon\} \tag{3.1}$$

ActiveRIPR starts by requesting the labels of a set of r_0 randomly selected samples. It then builds a RIPR model from these samples. Using a function which scores yet-unlabeled data considering the current model, ActiveRIPR selects the next set of samples to be labeled. The next section describes several such scoring functions. New models are trained as additional samples are added to the pool. While it is possible to efficiently update the current model using the new samples, we currently re-train from scratch, both for simplicity and to avoid any possibility of bias. The Active RIPR procedure is shown in Algorithm 3.1.1. X_u are the unlabeled samples, X_ℓ are the samples for which labels have been requested and Y_ℓ are their provided labels. X_t and Y_t represent a separate set of samples used for testing. M_s^k is the model trained at iteration k, based on samples queried using scoring function s. Err_s^k is the error of model M_s^k.

3.1.3 Active sample selection

Extensive research in the domain of active learning has led to a variety of algorithms which determine which points should be labeled next. We do not seek to supplant these, but rather adapt a subset of them to work with the class of model we target. The intuition is that, for data where most of the features are spurious, adapting the scoring function to consider only the significant features for each sample has the potential to improve the learning rate.

Uncertainty sampling

This score is used to pick the unlabeled data for which the label is the most uncertain, typically this translates to selecting the samples with the highest conditional entropy of the output given the features. Under the RIPR assumption, the label of a sample depends only on the projection to

Algorithm 3.1.1 Active RIPR with scoring function s

X_u (unlabeled samples), X_t (test samples), Y_t (test labels), $k = 0$ (iterations)
X_ℓ = SelectRandom(X_u)
Y_ℓ = LabelSamples(X_ℓ)
repeat
  $k = k + 1$
  M_s^k = TrainRiprModel(X_ℓ, Y_ℓ, X_u)
  Err_s^k = EvaluateRiprModel(M_s^k, X_t, Y_t)
  $x_i = \arg\min_{x_i \in X_u} s(M_s^k, x_i)$
  y_i = LabelSample(x_i)
  $X_u = X_u \setminus \{x_i\}$, $X_\ell = X_\ell \cup \{x_i\}$, $Y_\ell = Y_\ell \cup \{y_i\}$
until $Err_s^k \leq \epsilon$ or $|X_u| = 0$
return M_s^k

which the point is assigned. Using a RIPR model $M^k = \{\Pi^k, \tau^k\}$, the corresponding projection for a sample x and its label $\hat{y}(x)$ are determined as follows:

$$g^k(x) := \underset{(c,t)\in(\Pi^k,\tau^k)}{\arg\min}\ \hat{h}(t(c(x))|c(x))$$
$$\hat{y}(x) := t^k_{g^k(x)}(x),$$

where $\hat{h}$ denotes the conditional entropy estimator for a label given a subset of the features and $\hat{y}(x)$ is the prediction made for a sample x. The score for ActiveRIPR using uncertainty sampling simply considers the lowest conditional entropy on the projections of the model M^k_{uncrt}:

$$s_{uncrt}(x) = \min_{c\in C^k_{uncrt}, t\in T^k} \hat{h}(t(c(x))|c(x)) \tag{3.2}$$

Query by committee

Query by committee selects the samples on which the classifiers in an ensemble disagree. For a RIPR model M^k_{qbc}, this is simply obtained by comparing the labels assigned by each of the classifiers in T^k_{qbc}.

$$s_{qbc}(x) = \max_{t_i,t_j\in T^k_{qbc}} I(t_i(c_i(x)) \neq t_j(c_j(x))) \tag{3.3}$$

Information gain

The information gain criterion sorts unlabeled data according to the expected reduction in conditional entropy upon labeling each point. We use the notation $\hat{H}^k_{X_0,Y_0}(X_1)$ to represent the estimated conditional entropy of the samples X_1 given the samples X_0 and their labels Y_0. Assuming that, at iteration k, ActiveRIPR based on Information Gain has selected samples $X^k_{\ell,ig}$ while samples $X^k_{u,ig}$

are available for labeling, the information gain score can be expressed as follows:

$$\begin{aligned}\forall x \in X^k_{u,ig}, \quad s_{ig}(x) =& \hat{H}^k_{X_\ell,Y_\ell}(X^k_{u,ig}) \\ &- p(y=0)\hat{H}^k_{X_\ell\cup\{x\},Y_\ell\cup\{0\}}(X^k_{u,ig}) \\ &- p(y=1)\hat{H}^k_{X_\ell\cup\{x\},Y_\ell\cup\{1\}}(X^k_{u,ig})\end{aligned}$$

Low conditional entropy

Selecting samples with high uncertainty makes sense when there are aspects of the model not yet discovered – in the case of RIPR models, there might be projections that are informative, but are only relevant for a small subset of the data. However, once the informative projections have been discovered, selecting samples with high uncertainty often leads to the selection of purely noisy samples. In this case, selecting the data for which the classification is the most confident improves the model, as it is more likely that these points satisfy the model assumptions and can be used in the classification of their neighboring samples. This claim is verified experimentally, and the score for this query selection criteria is simply the opposite of the uncertainty sampling score:

$$s_{mc}(x) = 1 - \min_{c\in C^k_{mc}, t\in T^k_{mc}} \hat{h}(t(c(x))|c(x))$$

3.2 Informative Projection learning for feature hierarchies

We improve budget-constrained feature selection by leveraging the structure of the feature dependency graph and information about the cost required to compute each feature. We consider the process used to generate the features, as well as their cost, reliability and interdependence. Typically, our applications rely on a core set of features obtained through expensive measurements, enhanced using transformations derived (cheaply) from one or several core features. Also, some measurements can be obtained through more than one procedure. This structure, which is not considered in our previous work, could make our classifiers more powerful for the same total cost. Our proposed method works by generating, based on the feature dependencies, a regularizer which ensures that, once the cost for a feature is paid, all the features it depends on add no extra penalty. Thus, we leverage the cost and the redundancy of the features by generating penalties according to the structure of the dependency graph. This improves accuracy compared to a model obtained using the lasso at no increase in cost.

3.2.1 Cost-sensitive feature selection

We are given a dataset $(X \in \mathbb{R}^{n\times m}, Y \in \mathbb{R}^n)$ with features $A = \{a_1 \dots a_m\}$, a cost function $c : A \to \mathbb{R}$ and information about feature dependencies in the form of the directed graph (A, D), where $(a_i, a_j) \in D$ iff feature j depends on feature i. Learning the set of parameters $w \in \mathbb{R}^m$ involves minimizing a convex loss function f with a regularizer g which penalizes according to the feature cost.

$$w^* = \arg\min_w \sum_{i=1}^{n} f(w, x_i, y_i) + g(w) \tag{3.4}$$

A standard way of using the cost in performing feature selection is the weighted lasso $g_{\ell_1}(w) = \sum_{i=1}^{m} c(a_i)|w_i|$. The issue with this procedure is that it considers only the total cost for each feature ignoring the manner in which the cost decomposes across the dependency graph, which results in a potentially suboptimal selection of the sparsity pattern for a fixed cost in terms of accuracy, since some features that are virtually free are ignored.

3.2.2 Exploiting the feature dependency graphs through ℓ_1 and ℓ_2 penalties

Our procedure links each feature to their children in a dependency graph through ℓ_2 norms instead of penalizing them separately. Define the index set of children of a feature a_i as

$$\phi(a_i) = \{1 \leq j \leq m | (a_i, a_j) \in D\}\,. \tag{3.5}$$

The modified regularizer becomes $g_{c,D}(w) = \sum_{i=1}^{m} c(a_i)||w_{i,\phi(i)}||_2\,.$

For features that have no children, the term simply equals the ℓ_1 norm. For the rest, however, the ℓ_2 penalty decreases the weight magnitude, but only actually encourages sparsity on the parent feature to be 0 when the weights of all child features are 0. In this case, the ℓ_2 norm simply becomes an ℓ_1 norm and the feature is penalized as in the standard lasso case.

In some applications, the information can be relayed through several different sources, resulting in highly correlated – or even identical – features in the dataset. An example when this situation may occur is health monitoring. For many vital signs, there exist multiple means of obtaining measurements: invasive, non-invasive and computed indirectly from other vitals. Such correlated features are also present in data which holds responses to queries sent to several servers. Although features in the same series are all informative, it is clear that only one of them is needed at a time, including in the construction of child features. This leads to an 'OR' constraint – the presence of one of the features is necessary and sufficient to derive child features.

We enforce this constraint through a penalty which distributes the weight across the redundant features. Assume that $a_i^1 \ldots a_i^r$ is a series of features, either of which can be used to obtain a_i. The parameter w_i corresponding to a_i decomposes into the auxiliary components $w_i^1 \ldots w_i^r$, only one of which is non-zero. Let $\phi(i)$ denote any child features of a_i. The additional penalty for w_i is

$$g_{OR}(w_i) = c(a_i)||w_{i,\phi(i)}||_2 + \sum_{j=1}^{r}\sum_{k\neq j}^{r} c(a_i^j)||\bar{w}_i^j, w_i^k||_2\,, \quad \text{where } \bar{w}_i^j = \max\left(\frac{1}{w_i^j + 0.5} - 0.5, 0\right), \tag{3.6}$$

with the following constraint added to the optimization procedure: $\sum_{j=1}^{r} w_i^j = w_i$.

3.2.3 Leveraging feature hierarchies in vital sign monitoring

We applied our method to a classification problem involving clinical data obtained from a cardio-respiratory monitoring system. The system is designed to process multiple vital signs indicative of

the current health status of a critical care patient and issue an alert whenever some form of instability requires medical attention. In practice, a substantial fraction of these alerts are not due to real emergencies (true alerts), but instead are triggered by malfunctions or inaccuracies of the sensing equipment (artifacts). Each system-generated alert is associated with the vital sign that initiated it: heart rate (HR), respiratory rate (RR), blood pressure (BP), or peripheral arterial oxygen saturation (SpO_2). We extracted multiple temporal features independently for each vital sign over the duration of each alert and a window of 4 minutes preceding its onset. The 150 interdependent features included metrics of data density, as well as common moving-window statistics computed for each of the vital timeseries. Here, the cost of all base features is a unit, and one cost unit is added for each additional operation which needs to be performed to obtained derived features. The dataset has a total of 812 samples (alerts). Our type of regularization increases performance for the same cost when compared to the lasso.

Table 3.1: Comparison of our procedure against the lasso on the clinical data.

Cost	MSE (CFS)	MSE (lasso)		Cost	MSE (CFS)	MSE (lasso)
0	0.777094	0.777094		4	**0.244362**	0.250995
1	**0.343564**	0.435285		6	**0.244267**	0.250995
2	**0.245647**	0.250995		12	0.243772	0.243772

3.3 Projection-based gap-finding for data engineering

3.3.1 Guided data acquisition

We consider the problem of identifying discrepancies between training and test data which are responsible for the reduced performance of a classification system. Intended for use when data acquisition is an iterative process controlled by domain experts, our method exposes insufficiencies of training data and presents them in a user-friendly manner. The system is capable of working with any classification system which admits diagnostics on test data. We illustrate the usefulness of our approach in recovering compact representations of the revealed gaps in training data and show that predictive accuracy of the resulting models is improved once the gaps are filled through collection of additional training samples.

Consider an incident classification task in a radiation threat detection and adjudication system. As vehicles travel across international borders, they may be scanned for sources of harmful radiation, such as improperly contained medical or industrial isotopes, or nuclear devices. A substantial number of potential threats flagged by radiation measurement devices that may be used in such applications are actually non-threatening artifacts due to naturally occurring radioactive materials (e.g. ceramics, marble or cat litter). We have been using machine learning methodology to dismiss alerts that are confidently explainable by non-threatening natural causes, without increasing the risk of neglecting actual threat [29].

A robust alert adjudication system must be trained and validated on data that includes the actual threats. However, such data is (luckily) hard to come by. Therefore, it is practical and common to place the bulk of the available empirically gathered positive incident examples into a testing data set, and create training data using benign measurements mixed with a carefully chosen selection of

simulated threat. Nonetheless, the volumes and complexities of the feature space in data typically encountered in radiation measurement applications makes synthesizing a robust, sufficiently large, and (most importantly) comprehensive set of training data difficult and prone to omissions.

3.3.2 Finding meaningful gaps with RIPR

We present an engineering framework that facilitates data quality audits by automatically detecting gaps in training data coverage. These gaps denote differences in distributions of select variables between training and testing samples or point to areas of the feature space where the observed performance of the threat adjudication system appears suboptimal. The findings are presented in the form of human-readable, low-dimensional projections of data, in order to ensure interpretability of results and to simplify planning of corrective actions. The resulting iterative data improvement procedure boosts threat adjudication accuracy while reducing the workload of data engineers and application domain experts, when compared to using uninformed data gathering process.

The proposed process involves: (1) Building a threat classifier (any plausible type of a classification model can be used, we employ the random forest method primarily due to its scalability to highly-dimensional feature spaces, but also because of the computable on-the-fly metrics that diagnose reliability of predictions being made, which it provides), (2) Gap Retrieval Module (GRM), and (3) Human-driven procedure of addressing the identified gaps. Of particular relevance are two metrics that attempt to characterize reliability of predictions made by our random forest classifier: Dot-Product-Sum (DPS) and In-Bounds Score (IBS). DPS measures consistency of predictions made independently by the individual trees in the forest. IBS is perfect if for each node of a classification tree, the query fits within the range of the bounding box of the training data. Otherwise, it returns the value proportional to the fraction of nodes where the query was in-bounds. The GRM identifies where the original threat classification model performs well and where it performs poorly. It does this in one of two ways: (a) By finding low-dimensional projections where the testing and training data distributions differ significantly, and (b) By finding low-dimensional regions of data space where the original classifier experiences considerably low accuracy. The GRM leverages a previously published algorithm called Regression for Informative Projection Retrieval (RIPR) [35]. This algorithm discovers a small set of low-dimensional projections of possibly highly multivariate data which reveal specific low-dimensional structures in data, if such structures exist. RIPR's primary application is to improve understandability of classification, regression, or clustering tasks by explaining their results in a human-readable form. Here we extend the algorithm to facilitate improvements in training data generation, primarily by leveraging its ability to detect low-dimensional patterns of unexpected discrepancy between training and testing data, as well as low-dimensional structures of low performance areas. As a result of executing the GRM, the resulting low-dimensional subspaces are visualized and the domain experts and data engineers gain intuition as to what data may be missing from the training set and decide which parts of the feature space would most benefit from additional samples. The expanded training data will reflect these changes in the next machine learning iteration, and the process can continue until the training set is shaped into a faithful reflection of the test set, and the performance of the threat adjudication system is optimized.

3.3.3 Experimental Results

To find data gaps directly, our algorithm simply looks for mismatches between the training and testing data distributions in all 2 or 3 dimensional projections of data, to enable visually interpretable output. In this scenario, the algorithm returns the most prominent gap, even if it is located in a projection that yields relatively little information to support model predictions. Our results show that GRM is able to identify potentially irregularly shaped areas of mismatch between the training and test sets. The set up of our experiments involves the selection of two random samples: one of an arbitrary number of data points in the training set composed of semi-synthetic data and another similarly sized sample of data from the testing set. By taking these uniformly random samples, any mismatch we find is representative of the entire dataset with high probability, as the process does not change the training and testing data distributions. The leftmost graph in Figure 3.1 shows an overlay mismatch where the test set seems to simply be a shifted version of the training set. After conferring with the data engineers who built the data, we determined that the cause of the overlap is actually a single scalar parameter that was changed between two successive artificial data builds. Visualization provided by our framework allows data engineers to easily gather succinct information about the variations of the underlying structure of data.

Next, we tied in a cost function that determines which gaps are more meaningful in terms of the impact they may have on the threat classification performance. We can achieve this by incorporating diagnostic measures resulting from the original classifier performance evaluation, as observed on the test samples. The middle graph in Figure 3.1 shows a projection retrieved using a nonparametric loss estimator. We see that our random forest makes the most confident predictions (high DPS) for blue points which occupy a densely packed T-shaped space in the projection. Red points, which correspond to predictions which were not fully consistent among the trees in the forest (low DPS), indicate test data which may benefit from additional nearby training samples. They are far more spread out within the projection, and often reside near the edges of the gray point cloud which represents all of the training data.

Humans are good at understanding how to fill in gaps in low dimensional projections that retain some sort of a regular structure (i.e. a box or triangle), which is why we also devised a parametric loss estimator. It enables extraction of projections that contain regularly-shaped gaps which may cause considerable loss of threat classification performance. In the rightmost graph in Figure 3.1, we use linear Support Vector Machine model to separate high- and low-performance areas. Our goal here is to find projections of data where misclassified queries occupy one side of the classification boundary, while correctly classified queries occupy the other side. This is a useful type of a gap to look at because it identifies sets of features that jointly emphasize a controversy on how test data should be classified.

To show our framework increases model accuracy, we train random forest models using different subsets of training data. We start by taking our original data set and removing samples which fall within a certain region of a 2D projection, thus creating an artificial hole in the data. The random forest trained from this data set achieves 75.0% classification accuracy. We then run RIPR which identifies this gap and we add excluded samples back to the training set, which fills the gap that RIPR identified. Now training a new random forest, we achieve 75.7% accuracy. This shows we are able to improve model performance by filling in gaps that the GRM identifies. Additionally, we trained a model with a random subset of the original training set and obtain 75.2% accuracy. This shows us that filling in gaps in the training set is more efficient at improving model accuracy

than just adding more samples which may or may not help fill the gap.

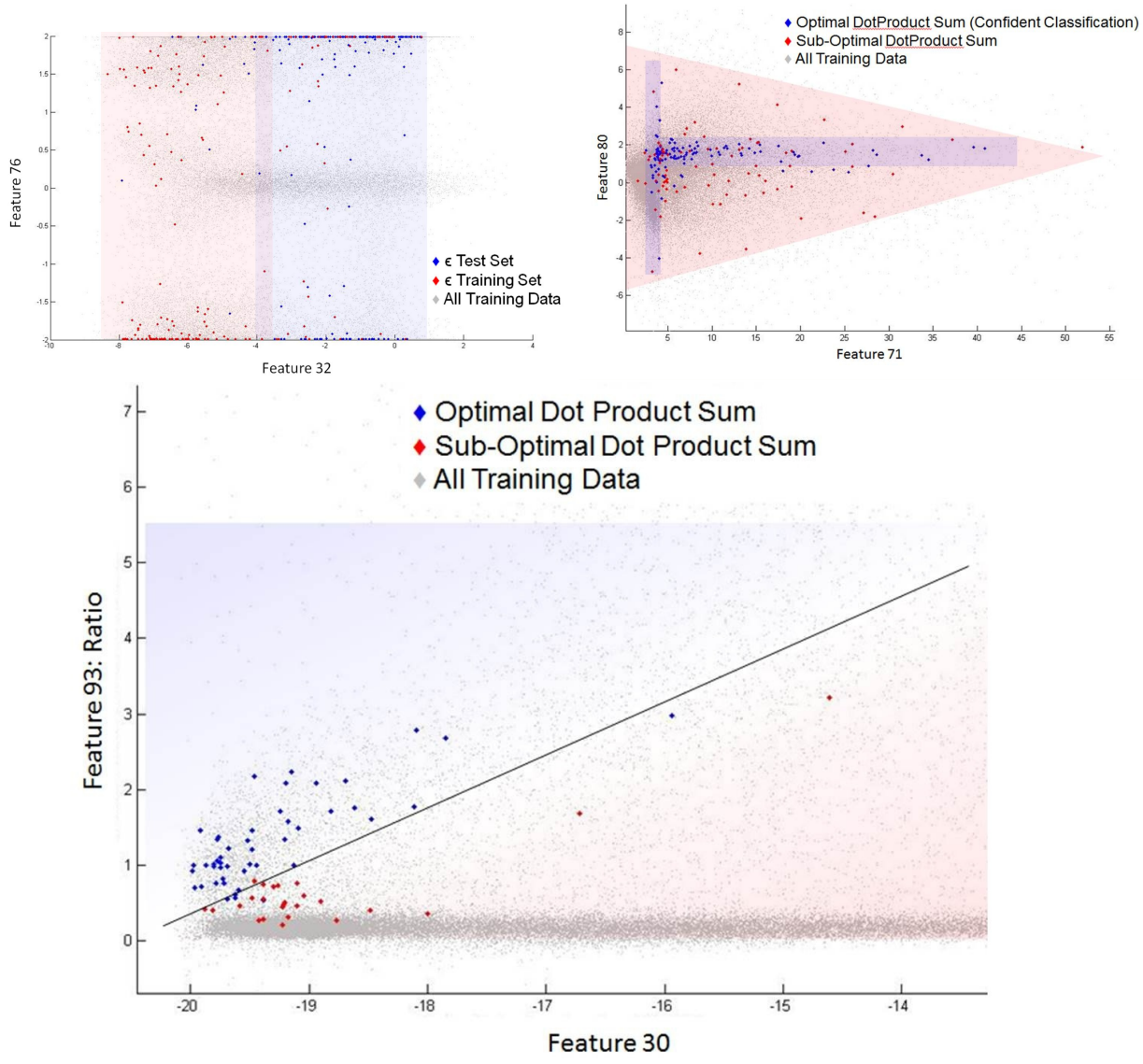


Figure 3.1: Example projections retrieved using direct (top left), and diagnostic nonparametric (top right) and parametric (bottom) approaches. Features names were obscured as the dataset is classifier.

Chapter 4

Detection of artifacts in clinical alerts from vital sign data

We outline a novel approach to distinguish correct alerts from artifacts in multivariate non-invasive vital signs data collected at the bedside of critical care patients. The framework selects informative low-dimensional projections of data that allow easy identification and interpretation of artifacts by humans. The results enable the construction of reliable decision rules which can be used to identify and ignore false alerts in real time. The proposed approach aims at reducing the tedious effort of expert clinicians who must annotate training data to support development of decision support systems. Using our method, the expert intervention is reduced to simply validating the outcome produced by an automated system using a small part of the available data. The bulk of the data can then be labeled automatically. The framework we present makes the decision process to aid the expert adjudication transparent and comprehensible. The projections jointly form a solution to the learning task. The method works under the assumption that each projection addresses a different subset of the feature space. The purpose is to determine which of the subsets of data correspond to genuine clinical alerts and which are artifacts due to particularities of the monitoring devices or data acquisition processes. We show how artifacts can be separated from real alerts in feature space using a small amount of labeled samples and present our system's utility in identifying patterns in data that are informative to clinicians.

4.1 Clinical alert adjudication

Clinical monitoring systems are designed to process multiple sources of information about the current health condition of a patient and issue an alert whenever a change of status, typically an onset of some form of instability, requires the attention of medical personnel. In practice, a substantial fraction of these alerts are not truly reflective of the important health events, but instead they are triggered by malfunctions or inaccuracies of the monitoring equipment. Accidentally disconnected ECG electrodes, poorly positioned blood oxygenation probe, and many other such problems unrelated to the patient's clinical condition may in practice yield instability alerts. Frequency of such false detections may cause the "alert fatigue" syndrome, pervasive among medical personnel, particularly in critical care environments. Alert fatigue may have adverse effects on the quality of care and patient outcomes. To maintain and enhance effectiveness of care, it is important to reliably

identify and explain these non-consequential artifacts. We outline a novel approach to distinguish correct alerts from artifacts in multivariate non-invasive vital signs data collected at the bedside of critical care patients. The approach selects informative low-dimensional projections of data that allow easy identification and interpretation of artifacts by humans. The results enable designing reliable decision rules that can be used to identify and ignore false alerts on-the-fly. They can also reduce data review and annotation efforts by expert clinicians.

The outlined problem can be generalized to any system designed to provide decision support to human users. Typically, this involves automating tasks such as grouping or classification while offering the experts insight into how the learning task was solved and how the model is applied to new data. An ideal scenario for a multitude of practical applications is the following: a domain expert provides the system with preliminary training data for some learning task; the system learns a model for the task (which uses only simple projections); the user provides queries (test points); for a given query point, the system selects the projection that is expected to be the most informative for this point; the system displays the outcome as well as a representation of how the task was performed within the selected projection.

We have previously formalized the problem of recovering simple projections for classification. The RIPR algorithm proposed there uses point estimators for conditional entropy and recovers a set of low-dimensional projections that classify queries using non-parametric discriminators in an alternate fashion - each query point is classified using one of the projections in the retrieved set. It can retrieve projections for any task that can be expressed in terms of a consistent loss function. RIPR is designed to work with any type of model or algorithm suitable to the particular task. For the application discussed here, we consider linear classifiers (SVM) and nonparametric clustering models (K-means). A classifier or a clustering model is trained for every recovered projection and used for the subset of data assigned to that projection.

The focus of this chapter is the application of RIPR to artifact identification. We illustrate the projections recovered for the task of discriminating artifacts from genuine clinical alerts. Since the types of alerts we focus on are triggered by excessive values of one of the vital signals at a time, we build separate artifact discrimination models for alerts on respiratory rate, blood pressure, and oxygen saturation. We evaluate the performance of these models at annotating unlabeled data. We also show, through case studies, how the models can help physicians identify outliers and abnormalities in the vital sign signals. Finally, we outline an active learning procedure meant to reduce the effort of clinicians in adjudicating vital sign data as healthy signal, artifact or genuine alarm.

4.2 Description of SDU patient vital sign data

A prospective longitudinal study recruited admissions across 8 weeks to a 24-bed trauma and vascular surgery stepdown unit. Non-invasive vital sign (VS) monitoring consisted of 5-lead electrocardiogram to determine heart rate (HR) and respiratory rate (RR; bioimpedance), non-invasive blood pressure (oscillometric) to determine systolic (SBP) and diastolic (DBP) blood pressure, and peripheral arterial oxygen saturation by finger plethysmography (SpO2). Noninvasive continuous monitoring data were downloaded from bedside monitors and analyzed for vital signs beyond local stability criteria: HR $\leq$ 40 or $\geq$140 beatsmin-1, RR$\leq$ 8 or $\geq$ 36 breathsmin-1, systolic BP $\leq$ 80 or $\geq$ 200, diastolic BP $\geq$ 110 mmHg, SpO2 $\leq$ 85% persisting for at least 4 out of 5 minutes of

continuous data. Each alert is active for the duration of the signal abnormality. The alert is associated with the first VS that is out of the normal range. VS time plots of patients whose vital sign parameters crossed the instability thresholds for any reason were visually assessed to judge them as patterns consistent with physiologically plausible instability, or as physiologically implausible and therefore artifactual. Artifact adjudication is challenging, even ground truth elicitation requiring the input of several expert clinicians.

Each alert is associated with a category indicating the type of the chronologically first VS signal that exceeds its stability limits. As a result, an alert with labeled as 'respiratory rate' may also include other VS outside of the bounds that have escalated shortly after the abnormal respiratory rate is recognized. We extracted a number of features to characterize each of the 813 alert events found in our data. The features are computed for each VS signal independently during the duration of each alert and a short window (of 4 minutes) preceding its onset. The list of features includes common statistics of each VS signal such as mean, standard deviation, minimum, maximum and range of values. It also includes features that are thought to be relevant (by domain experts) in discriminating between artifacts and true alerts.

There are a total of 147 features, shown in Table 6.5 derived from all VS as follows: The data density or duty cycle is the normalized count of signal readings during the alert period. A low value of this metric indicates the temporal sparseness of the data, while a value of zero simply means there was no data captured in that period. We also record the minimum and maximum of the first order difference of VS value during alert window. Extreme values of these statistics typically indicate a sharp increase/decrease of the VS value. The difference of means of VS values for the 4-minute window before and after the alert is also used, as is the value of the slope which results from fitting linear regression to the VS values versus the time index.

4.3 Performance of classification IPEs for artifact adjudication

We now show the classification models obtained to distinguish between artifacts and alerts corresponding to different VS. A description of the collected data, including hours of monitoring and number of patients is shown in Table 6.4. We considered alerts associated with different VS as separate classification tasks. Out of the 813 labeled alert samples, expert clinicians have identified 181 artifacts. The artifacts were adjudicated separately by two expert clinicians and a consensus was reached with respect to the labeling. Aside from the 813 labeled samples, there is a large amount of data, roughly 8000 samples, that remain unlabeled. The goal now is to train a separate model for each alert type such that other potential artifacts can be detected in the unlabeled data. Since domain experts will review the classification results, we rely on the RIPR framework to extract simple and intuitive projections, which will make it easy for clinicians to validate the results.

The majority of labeled alerts in our data are associated with the respiratory rate (RR). There are 362 such cases and a significant proportion of these (132 samples) are actually artifacts. Figure 4.1 shows the set of 2-dimensional projections retrieved by RIPR for the true alarm vs. artifact classification task. All the data points are represented in the plot as dots - the true alerts are shown in blue while the artifacts are shown in red. Recall that each point is only classified using one projection. To illustrate this, we plotted the data assigned for each projection with red circles (for artifacts) and blue triangles (for true alerts).

We apply the same procedure for alerts related to BP signals. There are 96 labeled examples

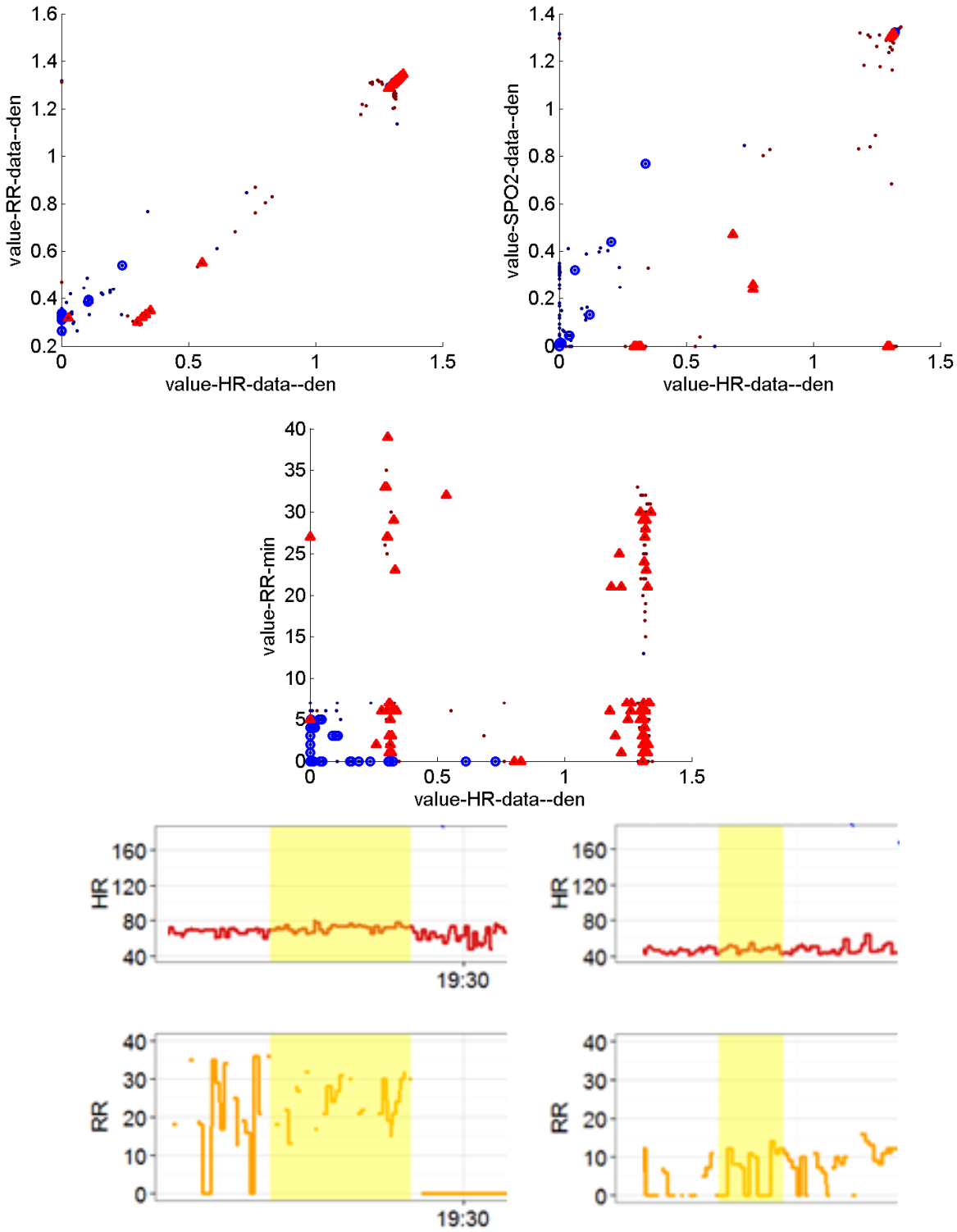


Figure 4.1: 2-D informative projections (top, middle) and sample vital signs (bottom) for RR (Respiratory Rate) alerts. Artifacts are represented with blue circles, while the true alerts are red triangles.

of such alerts out of which 24 are artifacts. The 2-D projections are displayed in Figure 4.2. Since in this case using two-dimensional projections appears insufficient to provide a convincing model, we also identified informative 3-dimensional projections. The figure shows the model resulting from this procedure. Only the alerts assigned to the specific projection were shown, in order to avoid overloading the figure.

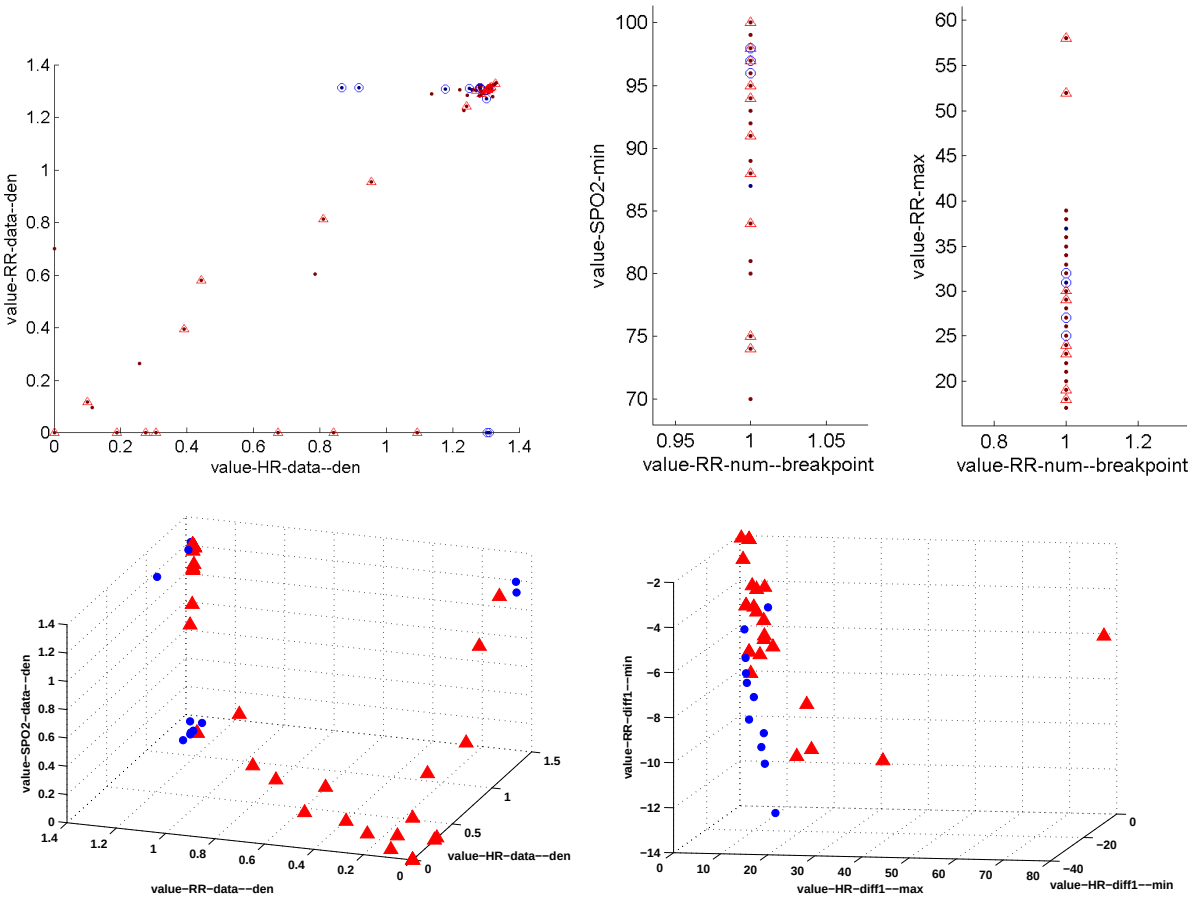


Figure 4.2: 2-D (top) and 3-D (bottom) informative projections for BP (Blood Pressure) alerts. Artifacts are represented with blue circles, while the true alerts are red triangles.

The training set for SpO2 consists of 259 samples out of which only 24 are labeled as artifacts. Figure 4.3 shows the 2-dimensional projections recovered for this problem. As there is substantial class overlap, we also trained 3-D models.

The remaining 96 alarms are associated with the HR, though the reviewers adjudicated only one of these an artifact. Predictive accuracy of the presented RIPR models is summarized in Table 4.1. The results are obtained through leave-one-out cross-validation.

The plots in Figure 4.1 show a good separation between artifacts and true alerts, which was one of our objectives. Also, the projections retrieved use data density features for the RR, SpO2 and HR signals as well as the minimum value RR. The use of these features is consistent with human intuition about what may constitute a respiratory rate artifact. For instance, a lot of missing data often signifies that the probe was removed from the patient for a period of time. The same can

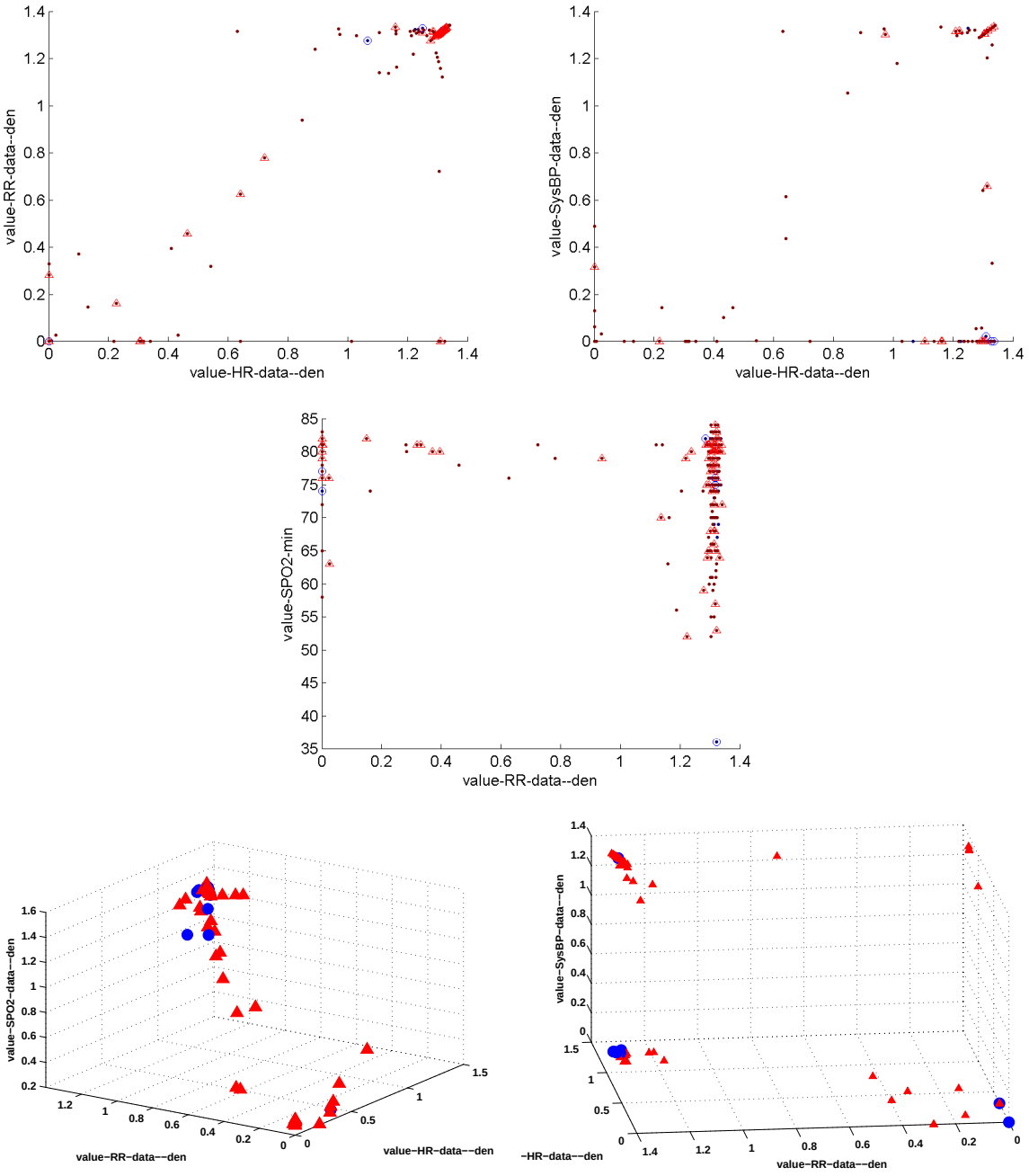


Figure 4.3: 2-D (top) and 3-D (bottom) informative projections for Oxygen Saturation (SPO2) alerts. Artifacts are represented with blue circles, while the true alerts are red triangles.

be said about minimum values for a VS - the measuring device could have been disconnected or misplaced. A good indication of the invalidity of a RR alert is the lack of HR data. So a simple decision rule - as stated by the clinicians - would be just to see whether there is HR data, if there is HR data, then the RR alert is an artifact, otherwise, it could be real. In classifying RR-based alerts, the algorithm correctly picked HR data density as the most important dimension.

The top right of the second graph in Figure 4.1 contains two blue circles representing samples

Table 4.1: Classification accuracy of RIPR models. Precision and recall in recovering the genuine alerts.

Alarm Type	RR	BP		SPO2	
	2D	2D	3D	2D	3D
Accuracy	0.98	0.833	0.885	0.911	0.9151
Precision	0.979	0.858	0.896	0.929	0.9176
Recall	0.991	0.93	0.958	0.945	0.9957

that would be classified as non-artifact according to the projection. Both of them have continuous stream of data, but the RR signals are irregular. This is a different type of artifact. Because there are very few alerts with this type of artifact, and the algorithm is designed to retrieve a small set of projections, they end up being misclassified. The vital signs corresponding to these two samples are presented in the bottom graph as Figure 4.1. Further investigation showed that variance of the signal values provides a reliable way to detect these outliers. Thus, expert attention was focused on this more problematic type of artifact rather than on the type that represents the majority of cases and is relatively easy to handle automatically. On the other hand, some samples were classified by the system as artifacts while the domain experts considered them true alerts. On closer inspection, they seemed to exhibit artifact-like features - with little or no recorded values in the HR signal. When we drilled down to look at the data, we found that the samples were actually labeled incorrectly in the training set. Therefore, the RIPR approach can also be useful in detecting inconsistencies due to human error.

For BP alert classification, for which the model is shown in Figure 4.2, though the features used are known to be informative, the class separation is not very clear. This is visible especially in the top right corner of the first plot, where we can observe a substantial overlap between artifacts and true alerts. In some cases, the algorithm provides two-dimensional projections, as required, but only one of the features is informative. This happens because that single feature has more discriminative capacity than other 2-dimensional projections where features might be correlated. Such occurrences are represented in plots the second and third plots of Figure 4.2. It is also noticeable from Figure 4.2 that the addition of the third dimension greatly improves the class separation. Again, the sparsity of data readings is an important feature, though this time the data density of three different VS needs to be considered for the subset of data presented in the first projection of Figure 4.2. The second 3-D projection uses the maximum and minimum values of HR and RR to classify artifacts and there exists a hyperplane separating the two classes.

The alerts based on SpO2 are more difficult to classify. Figure 4.3 confirms this, since both 3-D projections of the model use data sparsity features to isolate artifacts, though we must note that the separation is still somewhat noisy.

4.4 Clustering IPEs for identifying artifact archetypes

Additionally, by applying RIPR with K-means to clustering artifactual alerts, we identified human interpretable archetypes of false alerts as a preliminary step to corrective action plans. The intuition derived from these patterns is presented in the Discussion section.

4.5 Annotation framework for the classification of clinical alerts

Recovery of meaningful, explainable models is fundamental for the clinical decision-making process. We work with cardio-respiratory monitoring systems designed to process multiple vital signs indicative of the current health status of a critical care patient. The Step Down Unit (SDU) patients are connected to monitors, which continuously track the variability of multiple vital signs over time. The system issues an alert whenever some form of instability requires attention, that is, when any of the vitals exceeds pre-set control limits. Typically, such deviations indicate serious decline in patient health status. In practice, a substantial fraction of the issued alerts are not due to real emergencies (true alerts), but instead are triggered by malfunctions such as probe dislocation or inaccuracies of the sensing equipment (artifacts). Each system-generated alert is associated with the vital sign that initiated it: heart rate (HR), respiratory rate (RR), blood pressure (BP), or peripheral arterial oxygen saturation (SpO_2).

In order to reduce alarm fatigue in clinical staff, the ideal monitoring system would dismiss artifactual alerts on-the-fly and allow interpretable validation of true alerts by human experts when they are issued. As expected, the preparation of a high-quality and comprehensive sample of data needed to train an effective artifact adjudication system could be a tedious process in which important parts of the feature space are easy to neglect. This strenuous effort is often compounded by the sheer complexity of the involved feature space. Without a framework similar to the one presented here, precious expert time would be spent primarily navigating the dimensions of the data to establish grounds for labeling specific instances. We propose to not only select the minimal set of unlabeled data for human adjudication, but to also concurrently determine and present the informative small projections of this otherwise high-dimensional data.

We use ActiveRIPR to predict oxygen saturation alerts, treating the existing labeled data as the pool of samples available for active learning. There are 50 features in total. Roughly 10% of the data has been manually labeled and the aim is to use that subset to determine which of of the unlabeled samples are worth the experts' attention. We performed 10-fold cross validation, training the ActiveRIPR model on 90% of the labeled samples and using the remainder to calculate the learning curve. Table 4.2 shows the number of samples required to reach an accuracy of 0.85 (a value deemed acceptable by clinicians) and 0.88 (the maximum achievable accuracy). Information Gain performs considerably better than the rest and uncertainty sampling, despite having performed poorly in simulations, is also competitive. The results indicate that an accuracy of 0.88 can be achieved by labeling less than 25% of the total samples using the InfoGain scoring function.

Table 4.2: Percentage of samples needed by ActiveRIPR and ActiveRIPRssc to achieve accuracies of 0.85 and 0.88 in oxygen saturation alert adjudication.

	ActiveRIPR		ActiveRIPRssc	
Target Accuracy	**0.85**	**0.88**	**0.85**	**0.88**
Score Function				
Uncertainty	18.33	18.33	36.67	50.00
QbC	46.67	46.67	86.67	86.67
InfoGain	21.67	25.00	25.00	51.67
CondEntropy	43.33	46.67	48.33	63.33

Table 4.2 summarizes the proportion of samples needed by ActiveRIPR and ActiveRIPRssc to

achieve 0.85 or 0.88 accuracy on the hold out test data of the oxygen saturation alert dataset.

Given the success of ActiveRIPR using the InfoGain selection criterion for the oxygen saturation alert adjudication, we proceeded to apply it to detecting blood pressure alerts. This time, we compared it against other classification methods using uncertainty sampling. This type of sampling differs from the uncertainty score used by ActiveRIPR in that it considers the entire feature space as opposed to only low-dimensional projections when making the selection. Also, the classifiers are trained on all features as opposed to only a subset, so it is expected that they would perform well. Random Forests and KernelSVM are some of the well-performing classifiers, which we selected because we aim to assess how accurate the system can be when there are no restrictions on model dimensionality.

Samples	K-nn	K-SVM	RF	ActiveRIPR
20	0.61	0.64	**0.65**	**0.65**
50	0.58	0.66	**0.71**	0.70
75	0.6	0.63	0.71	**0.75**

Table 4.3: Active learning for blood pressure alerts

Table 4.5 presents the mean leave-one-out accuracy of after 20, 50 and 75 labels. ActiveRIPR's performance approaches that of Random Forests and, at times, outperforms KernelSVM, while maintaining compactness of representation and performing drastic feature reduction. The RIPR models used, at any time, at most two 3-dimensional projections, so 6 features in total. ActiveRIPR wins by a sizeable margin over K-NN which we tested because of its potential for interpretability.

Applying clustering on Informative Projections has yielded the archetypes presented in Figure 4.4. Clinician review of the patterns aided the clinicians in formulating the following conjectures. A cluster of RR artifacts on features RR mean (0,4) and RR standard deviation (2,5) likely suggests a loose ECG lead, while a pattern RR mean (33,40) and RR standard deviation of (0,10) is likely due to insufficient bioimpedance. For SpO2 artifacts, features SpO2 min and SpO2 slope expose patterns suggesting motion, sensor reattachment and loose lead or low perfusion. The patterns identified for RR artifacts (HR-DD =0, SPO2-DD bimodal with peaks at 0.1 and 0.01) suggest the lack of ECG electrode integrity. For SpO2, decreases in both HR-DD and RR-DD appear associated with artifact and suggest an overall problem with signal pickup in both SpO2 and the ECG/RR sensors. These artifact archetypes agree with clinical intuition and can potentially be used to guide corrective actions in practice.

4.6 Studies of expert labeling using time series and informative projections

In order to ascertain the effectiveness of the informative projection models in assisting domain experts, we have performed a user study in which two expert clinicians were asked to adjudicate alerts based on the projection models and, separately, based on vital signals. The experiment was performed in two stages, each of the stages dealing with 20 RR alerts and 20 SpO2 alerts. In the first stage, the projections display each query against the background of the entire sample space. In the second stage, the projection provides a zoomed-in view of the area close to the query. In

total, 80 samples were labeled, each sample being assigned four scores, two by each clinician, one based on the projection as well as one the vital sign time series for the alert. The scores range from -3, indicating high reviewer confidence that the alert constitutes an artifact, to 3, indicating high reviewer confidence that the alert is real. Based on the scores assigned by each reviewer, the alert falls into one of three confidence categories, represented in Table 4.4. If there is disagreement between reviewers, or a reviewer is uncertain, the sample is marked as ambiguous and no label can be assigned.

Table 4.4: Annotation scoring matrix. Category and label assignment based on reviewer scores.

Category	C1	Strong agreement
	C2	Weak agreement, need 3rd reviewer
	C3	Disagreement
Label	A	Artifact
	R	Real alert
	-	Ambiguous artifact. No label assigned.

		Reviewer 1 Confidence						
	Category	**3**	**2**	**1**	**0**	**-1**	**-2**	**-3**
	3	R	R	R	-	-	-	-
	2	R	R	R	-	-	-	-
	1	R	R	R	-	-	-	-
	0	-	-	-	-	-	-	-
Reviewer 2 Confidence	**-1**	-	-	-	-	A	A	A
	-2	-	-	-	-	A	A	A
	-3	-	-	-	-	A	A	A

An example of the visual representations shown to clinicians for adjudication is in Figure 4.5. The RR alert that needs to be adjudicated, identified with a star symbol, is projected on the features value_RR_max and value_RR_median, amid previously labeled data. It is located in a cluster of data that were labeled as artifacts. Based on this informative projection, it was labeled as a real alert by both clinicians. Based on the time series corresponding to this alert, which is also represented at the top of Figure 4.5, the alert was also labeled as real. In this case, the outcome of using the compact representation using Informative Projection is the same as that of using the full time series representation with the added benefit that adjudication can be performed faster/easier by domain experts and that the labels can be automatically assigned by the system.

By merging the reviewer scores as shown in Table 4.4., each of the samples is assigned a label and confidence category from the projection-assisted annotation, and a separate one from the adjudication based on vital signs. The latter is considered the ground truth. Table 4.5 shows the extent of overlap between the confidence categories. Out of the 80 samples, 36 are labeled with the same confidence (and label) irrespective of the manner of annotation, 3 samples that could not be annotated based on the trace were annotated by analyzing the projections, 10 of the samples were annotated with more confidence based on the VS, while the remaining 31 (38.75% of total) could not be annotated based on the projected representation, but could be adjudicated using VS. This experiment points out that projection-assisted annotation was useful in obtained labels for 35 samples, 53.03% of the non-ambiguous cases, reducing the need for VS adjudication to 31

samples, 46.97% of the non-ambiguous cases.

Table 4.5: Categories of Projection-assisted labeling and VS-based labeling.

Number of samples		**Category of vital sign -based labeling**			***Total***
		C1	**C2**	**C3**	
Category of projection-assisted labeling	**C1**	19	0	1	*20*
	C2	10	3	*2*	*15*
	C3	24	7	14	*45*
Total		*53*	*10*	*17*	*80*

We have evaluated the success of the IP-assisted annotation by comparing the resulting labels with the ground-truth obtained through the VS analysis. The comparison, shown in Table 4.6, was performed separately for the sets of labels of the two experts and for the final labels obtained by combining their scores. 27 of the samples were correctly classified using the projections, 31 could not be classified due to either expert disagreement or at least one of the experts being uncertain, 4 artifacts could not be filtered out using the projections, while only one alert was missed. The remaining 17 samples could not be adjudicated even through the use of the time series. As shown in Table 4.6, combining the predictions of the two experts results in a more conservative estimate of the label, but it also decreases the number of mistakes compared to single-expert prediction.

Table 4.6: Success of Informative Projection-assisted labeling compared to the ground-truth obtained by VS-based labeling.

Number of samples	**Correct IP-assisted classification**[1]	**Inconclusive IP-assisted classification**	**Incorrect IP-assisted classification**	**Sample is ambiguous**[2]
Reviewer 1	31	34	10 (4FN [3], 5 FP)	5
Reviewer 2	43	25	11 (1 FN, 7 FP)	1
Final	27	31	5 (1 FN, 4 FP)	17

As previously described, the projections in the second stage of the annotation experiment, showed a zoomed-in view of the area around a query as opposed to the entire set of samples. This was done to avoid misleading predictions due to the imbalance of true alerts and artifacts in the labeled data. Tables 4.7 and 4.8 show the confidence and success statistics for each of the two stages. The annotation is both more confident and more accurate in the second stage of the labeling, showing that the change in resolution was effective.

We have presented the use of the RIPR algorithm to support annotation of clinical data. We have shown the models that our RIPR framework produces for automatic data labeling and how the retrieved low-dimensional projections make it possible for domain experts to quickly validate the assigned labels. We also illustrated how RIPR models can be used to find special cases and incomplete or invalid data. Thus, the proposed framework promises to be useful to clinicians by partially automating annotation of medical data in a human understandable and intuitive manner.

Table 4.7: Comparison of first stage and second stage annotations.

Number of samples	Correct IP-assisted classification	Inconclusive IP-assisted classification	Incorrect IP-assisted classification	Sample is ambiguous
First Stage				
Reviewer 1 C	14	21	4	1
Reviewer 2 H	23	11	5	1
Final	11	18	3	8
Second Stage				
Reviewer 1 C	17	13	6	4
Reviewer 2 H	20	14	6	0
Final	16	13	2	9

Table 4.8: Comparison of first stage and second stage annotations.

Number of samples		**Category of vital sign -based labeling**			
(first stage)		**C1**	**C2**	**C3**	
Category of projection-assisted labeling	**C1**	10	0	0	***Total***
	C2	3	1	0	*10*
	C3	15	3	8	*4*
Total		*28*	*4*	*8*	*26*
					40
(second stage)		**C1**	**C2**	**C3**	
Category of projection-assisted labeling	**C1**	9	0	1	***Total***
	C2	7	2	2	*10*
	C3	9	4	6	*11*
Total		*25*	*6*	*9*	*19*
					40

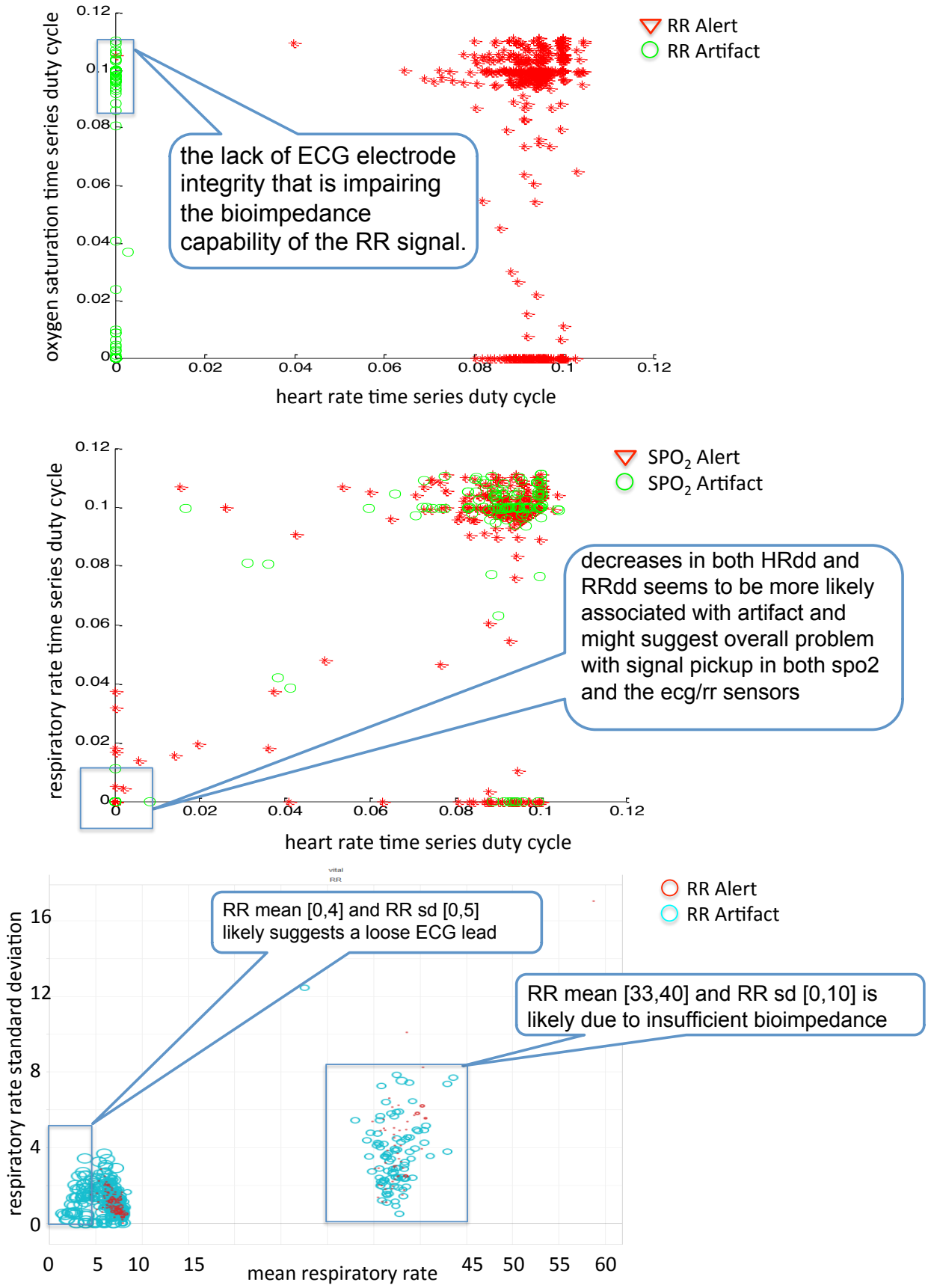


Figure 4.4: Artifact Archetypes.

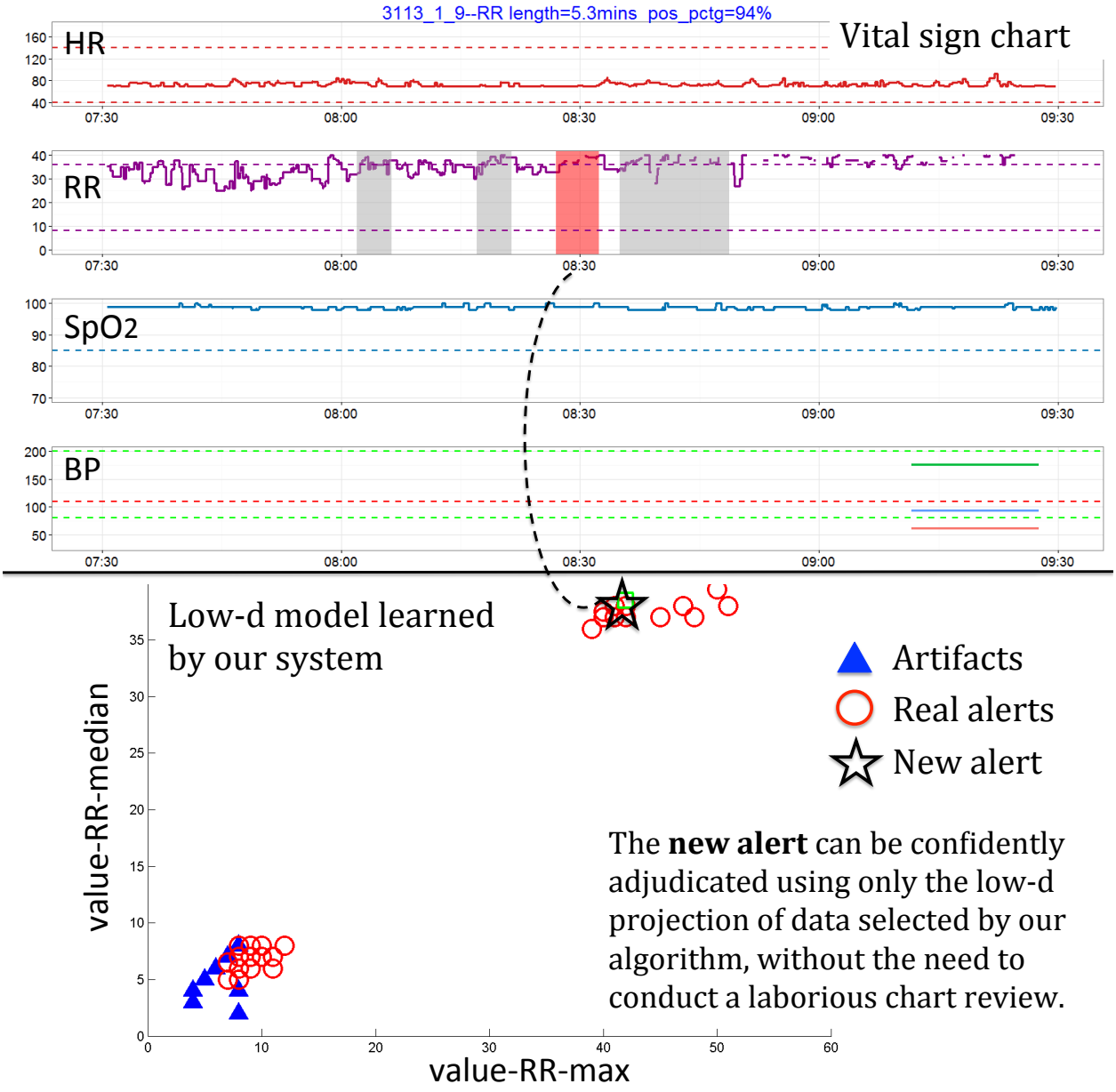


Figure 4.5: Example of projection-assisted annotation. Original vital sign chart (top) and informative projection (bottom). The test point that needs to be labeled is identified with a star symbol.

Chapter 5

Ensembles for Large-scale Data Partitioning

5.1 Optimizing tree ensembles

We presented the concept of using query-specific classifiers selected from a predefined pool to handle individual data points in classification tasks. We leverage this concept for ensembles of generic classifiers, rather than just IPEs. The idea is rather straightforward: to train meta-classifiers which determine which submodels should be used for each data point.

In the experiment below, we first train an ensemble of classification trees as described in [15], using feature bagging. Then, for each tree, we train a meta-classifier which predicts 1 if the tree is deemed accurate for the data point x and 0 otherwise. When a test query is provided to the optimized ensemble, we first evaluate each classifier, to determine if it should be used. Finally, we average over the predictors of the selected classifiers. This simple procedure typically improves performance of forests, as shown in Table 5.1.

Table 5.1: Performance improvement of tree ensembles through the use of query-specific selection. For each of the experiments, we report the number of features in the dataset a, the number of samples n, the number of classes $|\mathcal{Y}|$, the error of the tree ensemble and the error of the optimized ensemble.

Dataset	**a**	**n**	$\lvert\mathcal{Y}\rvert$	**Ensemble Err (%)**	**Opt Ensemble Err (%)**
Chars74k	85	3410	62	48.39	**48.09**
G50C Avg	50	550	2	7.81	**6.18**
Letter	16	16000	26	6.75	**5.50**
USPS	256	11000	10	2.6	**2.3**

5.2 Backpropagation forests

Up to this point, we have applied the submodel optimization paradigm to projections and random forests. While successful, the accuracy is limited by the compactness imposted by application-related restrictions. In some applications, however, there are no constraints on model size, the single performance metric being accuracy. Even in this case, training models that are specialized

for a subset of the samples, using a subset of the features can bring considerable benefits. The models we previously presented, namely the IPEs and the optimized ensembles, show that query specific handling and subspace selection can be leveraged to improve the performance of classification systems. We aim to allow data shift across submodels in the ensemble. Allowing flexible assignment of both features and samples, as shown in Figure 5.1, can be done by training tree ensembles which admit a differentiable global loss.

Figure 5.1: Online data shift, in terms of both features and samples, across ensemble submodels.

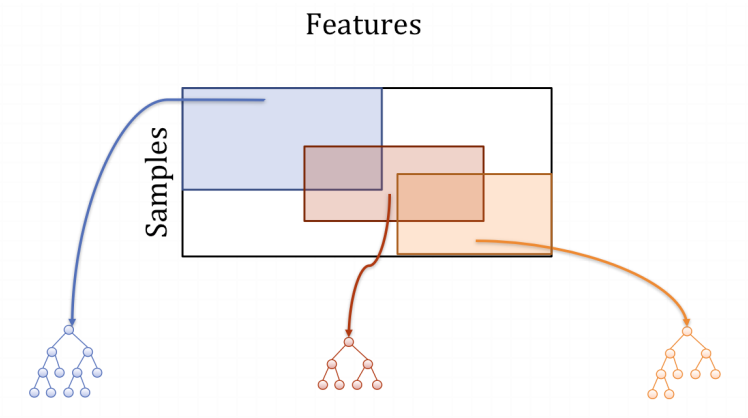


Figure 5.2: Before shift

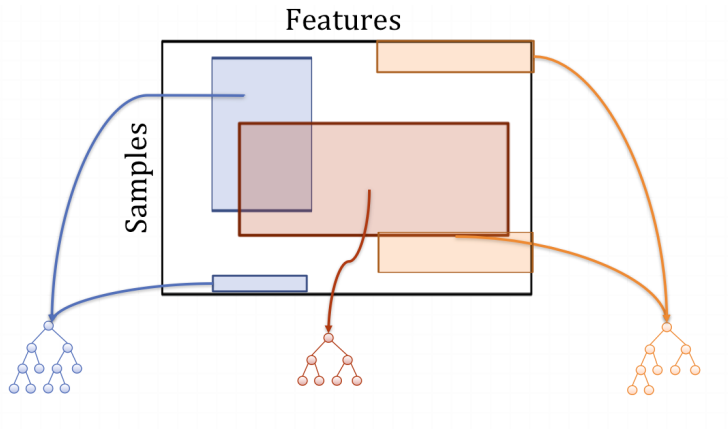


Figure 5.3: After shift

We combine sample partitioning and subspace selection for high performing forest classifiers with representation learning. Thus, we introduce Deep Convolutional Neural Decision Forests – a novel approach that unifies classification trees with the representation learning functionality known from deep convolutional networks, by training them in a joint manner. To combine these two worlds, we introduce a stochastic and differentiable decision tree model, which steers the representation learning usually conducted in lower level layers of a (deep) convolutional network. Our model differs from conventional deep networks because a decision forest provides the final predictions and it differs from conventional decision forests since we propose a principled, joint and

global optimization of split and leaf node parameters. We show experimental results on benchmark machine learning datasets like MNIST and ImageNet and find on-par or superior results when compared to state-of-the-art deep models. Most remarkably, we obtain a Top5-Error of only 7.84% on ImageNet validation data when integrating our forest in a single-model GoogLeNet architecture, without any form of training data set augmentation.

5.2.1 Related work

Random forests [1, 14, 21] have a rich and successful history in machine learning in general and the computer vision community in particular. Their performance has been empirically demonstrated to outperform most state-of-the-art learners when it comes to handling high dimensional data problems [18], they are inherently able to deal with multi-class problems, are easily distributable on parallel hardware architectures while being considered to be close to an ideal learner [52]. These facts and many (computationally) appealing properties made them attractive for various research areas and commercial products. In such a way, random forests could be used as out-of-the-box classifiers for many computer vision tasks such as image classification [13] or semantic segmentation [16, 95], where the input space (and corresponding data representation) they operated on was typically predefined and left unchanged.

One of the consolidated findings of modern, (very) deep learning approaches [66, 75, 102] is that their joint and unified way of learning feature representations together with their classifiers greatly outperforms conventional feature descriptor & classifier pipelines, whenever sufficient amounts of training data and compute capabilities are available. In fact, the recent work in [53] demonstrated that deep networks could even outperform humans on the task of image classification. Similarly, the success of deep networks extends to speech recognition [109] and automated generation of natural language descriptions of images [31].

Addressing random forests to learn both, proper representations of the input data and the final classifiers in a joint manner is an open research field which received only little attention in the literature so far. Notable but limited exceptions are [64, 80] where random forests were trained in an entangled setting, stacking intermediate classifier outputs with the original input signal. The approach in [88] introduced a way to integrate multi-layer perceptrons as split functions, however, representations were learned only locally at split node level and independently among split nodes. While these attempts can be considered early forms of representation learning in random forests, they still delivered suboptimal results with prediction accuracies well below the state-of-the-art.

In this work we present *Deep Convolutional Neural Decision Forests* – a novel approach to unify appealing properties from representation learning as known from deep architectures with the divide-and-conquer principle of decision trees. We introduce a stochastic, differentiable, and therefore back-propagation compatible version of decision trees, guiding the representation learning in lower layers of deep convolutional networks. Thus, the task for representation learning is to reduce the uncertainty on the routing decisions of a sample taken at the split nodes, such that a globally defined loss function is minimized. Additionally, we show how to obtain globally optimal predictions for all leaves of our trees, which we pose as a convex formulation of a minorization-maximization problem, without the need for tedious step-size selection. Therefore, at test time we can take the optimal decision for a sample ending up in the leaves, with respect to all the training data and the current state of the network.

Our realization of backpropagation trees is modular and we discuss how they can be easily

implemented in terms of *back-prop-tree - routing* and *-loss* layers in existing deep learning frameworks such as Caffe [61], MatConvNet [105], Minerva [1], etc., respectively. Of course we also maintain the ability to use back-propagation trees as (shallow) stand-alone classifiers. We demonstrate the efficacy of our approach on a range of datasets, including MNIST and ImageNet, showing superior or on-par performance with state-of-the-art methods.

The main contributions of our work relate to enriching decision trees with the capability of representation learning, which requires tree training concepts departing from the prevailing greedy, local optimization procedures typically employed in the literature [21]. To this end, we will present the parameter learning task in the context of empirical risk minimization. Related approaches of tree training via global loss function minimization were e.g. introduced in [93] where during training a globally tracked weight distribution guides the optimization, akin to concepts used in boosting. The work in [60] introduced regression tree fields for the task of image restoration, where leaf parameters were learned to parametrize Gaussian conditional random fields, providing different types of interaction. In [100], fuzzy decision trees were presented, including a training mechanism similar to backpropagation in neural networks. Despite sharing some properties in the way parent-child relationships are modeled, our work differs as follows: i) We provide a globally optimal strategy to estimate predictions taken in the leaves (whereas [100] simply uses histograms for probability estimation). ii) The aspect of representation learning is absent in [100] and iii) We do not need to specify additional hyper-parameters which they used for their routing functions (which would potentially account for millions of additional hyper-parameters needed in the ImageNet experiments). The work in [79] investigated the use of sigmoidal functions for the task of differentiable information gain maximization. In [9], a Bayesian approach using priors over all parameters is introduced, where also sigmoidal functions are used to model splits, based on linear functions on the input (the non-Bayesian work from Jordan [62]). Other hierarchical mixture of expert approaches can also be considered as tree-structured models, however, lacking both, representation learning and ensemble aspects.

5.2.2 Decision trees with stochastic routing

Consider a classification problem with input and (finite) output spaces given by $\mathcal{X}$ and $\mathcal{Y}$, respectively. A *decision tree* is a tree-structured classifier consisting of decision nodes (*a.k.a.* split nodes) and prediction nodes. Decision nodes indexed by $\mathcal{N}$ are internal nodes of the tree, while prediction nodes indexed by $\mathcal{L}$ are the terminal (*a.k.a.* leaf) nodes of the tree. Each prediction node $\ell \in \mathcal{L}$ is assigned a probability distribution $\boldsymbol{\pi}_\ell$ over $\mathcal{Y}$. Each decision node $n \in \mathcal{N}$, instead, is assigned a decision function $d_n(\cdot;\Theta) : \mathcal{X} \rightarrow [0,1]$ parametrized by Θ, which is responsible for the routing of samples along the tree. When a sample $\boldsymbol{x} \in \mathcal{X}$ reaches a decision node it will be routed in the left or right subtree based on the output of $d_n(\boldsymbol{x};\Theta)$. In standard decision forests, d_n is binary and the routing is deterministic. Here, we consider rather a probabilistic routing, *i.e.* the routing direction is the output of a Bernoulli random variable with mean $d_n(\boldsymbol{x};\Theta)$. Once a sample ends in a leaf node ℓ, the related tree prediction is given by the class-label distribution $\boldsymbol{\pi}_\ell$. In the case of stochastic routings, the leaf predictions will be averaged by the probability of reaching the leaf. Accordingly, the final prediction for sample $\boldsymbol{x}$ from tree t with decision nodes parametrized

[1] `https://github.com/dmlc/minerva`

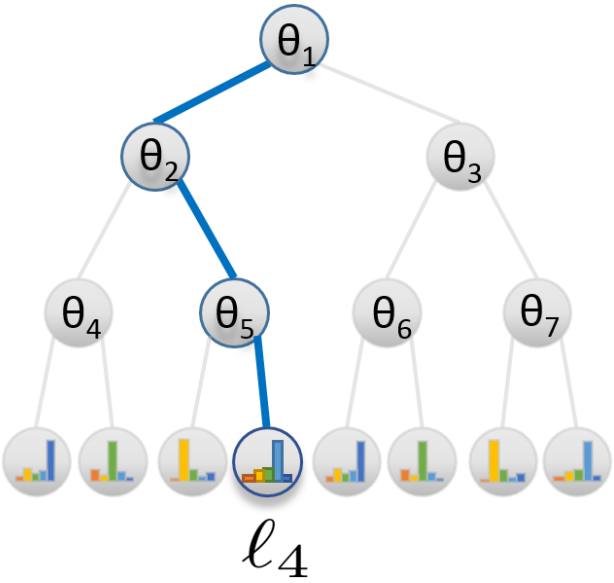


Figure 5.4: Exemplary routing of a sample $\boldsymbol{x}$ along a tree to reach ℓ_4, using $f_n(\boldsymbol{x};\Theta) = \boldsymbol{x}^\top\theta_n$. Here, $\mu_{\ell 4} = d_1(\boldsymbol{x}^\top\theta_1)\bar{d}_2(\boldsymbol{x}^\top\theta_2)\bar{d}_5(\boldsymbol{x}^\top\theta_5)$

by Θ is given by

$$\mathbb{P}_T[y|\boldsymbol{x},\Theta,\boldsymbol{\pi}] = \sum_{\ell\in\mathcal{L}} \pi_{\ell y}\mu_\ell(\boldsymbol{x}|\Theta) \tag{5.1}$$

where $\boldsymbol{\pi} = (\boldsymbol{\pi}_\ell)_{\ell\in\mathcal{L}}$ and $\pi_{\ell y}$ denotes the probability of a sample reaching leaf ℓ to take on class y, while $\mu_\ell(\boldsymbol{x}|\Theta)$ is regarded as the *routing function* that provides the probability that sample $\boldsymbol{x}$ will reach leaf ℓ. Clearly,

$$\sum_\ell \mu_\ell(\boldsymbol{x}|\Theta) = 1$$

for all $\boldsymbol{x}\in\mathcal{X}$.

In order to provide an explicit form to the routing function we introduce the following binary relations that depend on the tree's structure: $\ell \swarrow n$, which is true if ℓ belongs to the left subtree of node n, and $n \searrow \ell$, which is true if ℓ belongs to the right subtree of node n. We can now exploit these relations to express μ_ℓ as follows:

$$\mu_\ell(\boldsymbol{x}|\Theta) = \prod_{n\in\mathcal{N}} d_n(\boldsymbol{x}|\Theta)^{\mathbb{1}_{\ell\swarrow n}}\bar{d}_n(\boldsymbol{x}|\Theta)^{\mathbb{1}_{n\searrow\ell}}\,, \tag{5.2}$$

where

$$\bar{d}_n(\boldsymbol{x}|\Theta) = 1 - d_n(\boldsymbol{x}|\Theta)\,,$$

and $\mathbb{1}_P$ is an indicator function for the truth value of P. Although the product in (5.2) runs over all nodes, only decision nodes along the path from the root node to the leaf ℓ contribute to μ_ℓ, because for all other nodes $\mathbb{1}_{\ell\swarrow n}$ and $\mathbb{1}_{n\searrow\ell}$ will be both 0 (see Fig 5.4 for an illustration).[2]

[2]we assume $0^0 = 1$.

Decision nodes For the remainer of the chapter, We consider decision functions delivering a stochastic routing with decision functions being defined as follow:

$$d_n(\cdot;\Theta) = \sigma \circ f_n(\cdot;\Theta)\,, \tag{5.3}$$

where

$$\sigma(x) = (1 + e^{-x})^{-1}$$

is the sigmoid function, and $f_n(\cdot;\Theta) : \mathcal{X} \to \mathbb{R}$ is a real-valued function depending on the sample and the parametrization Θ. Further details about the functions f_n can be found in Section 5.2.4, but intuitively depending on how we choose these functions we can model trees having shallow decisions (*e.g.* such as in oblique forests [55]) as well as deep ones.

Forests of decision trees. A forest is an ensemble of decision trees $\mathcal{F} = \{T_1, \ldots, T_{\mathsf{k}}\}$, which delivers a prediction for a sample $\boldsymbol{x}$ by averaging the output of each tree, *i.e.*

$$\mathbb{P}_{\mathcal{F}}[y|\boldsymbol{x}] = \frac{1}{\mathsf{k}} \sum_{k=1}^{\mathsf{k}} \mathbb{P}_{T_k}[y|\boldsymbol{x}]\,, \tag{5.4}$$

where we omitted the trees' parameters for notational convenience.

5.2.3 Learning backpropagation trees

Learning a decision tree modeled as in Section 5.2.2 requires estimating both the decision node's parametrization Θ and the leaf predictions $\boldsymbol{\pi}$. For this estimation we adhere to the minimum empirical risk principle with respect to a training set $\mathcal{T} \subset \mathcal{X} \times \mathcal{Y}$ under log-loss, *i.e.* we search for the minimizers of the following risk term:

$$R(\Theta, \boldsymbol{\pi}; \mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{(\boldsymbol{x},y)\in\mathcal{T}} L(\Theta, \boldsymbol{\pi}; \boldsymbol{x}, y)\,, \tag{5.5}$$

where $L(\Theta, \boldsymbol{\pi}; \boldsymbol{x}, y)$ is the log-loss term for the training sample $(\boldsymbol{x}, y) \in \mathcal{T}$, which given by

$$L(\Theta, \boldsymbol{\pi}; \boldsymbol{x}, y) = -\log(\mathbb{P}_T[y|\boldsymbol{x}, \Theta, \boldsymbol{\pi}])\,,$$

and $\mathbb{P}_T$ is defined as in (5.1).

We consider a two-step optimization strategy, where we alternate updates of Θ with updates of $\boldsymbol{\pi}$ in a way to minimize (5.5). The next two subsections describe the two update steps, while subsection 5.2.4

Learning Decision Nodes

All decision functions depend on a common parameter Θ, which in turn parametrizes each function f_n in (5.3). Since we made no assumption thus far about the very nature of the functions f_n, the optimization of the risk with respect to Θ, assuming $\boldsymbol{\pi}$ fixed, could be in general a difficult and large-scale optimization problem. As an example, Θ could absorb all the parameters of a deep neural network having f_n as one of its output units. For this reason, we will employ a Stochastic

Gradient Descent (SGD) approach to minimize the risk with respect to Θ, as usually done in the context of deep neural networks:

$$\begin{aligned}\Theta^{(t+1)} &= \Theta^{(t)} - \eta \frac{\partial R}{\partial \Theta}(\Theta^{(t)}, \boldsymbol{\pi}; \mathcal{B}) \\ &= \Theta^{(t)} - \frac{\eta}{|\mathcal{B}|} \sum_{(\boldsymbol{x},y)\in\mathcal{B}} \frac{\partial L}{\partial \Theta}(\Theta^{(t)}, \boldsymbol{\pi}; \boldsymbol{x}, y)\end{aligned} \tag{5.6}$$

where $0 < \eta$ is the learning rate and $\mathcal{B} \subseteq \mathcal{T}$ is a random subset (*a.k.a.* mini-batch) of samples from the training set. Although we do not show it, we consider also an additional momentum term to smooth the variations of the gradient term.

The gradient term with respect to L can be decomposed by the chain rule in

$$\frac{\partial L}{\partial \Theta}(\Theta, \boldsymbol{\pi}; \boldsymbol{x}, y) = \sum_{n\in\mathcal{N}} \frac{\partial L(\Theta, \boldsymbol{\pi}; \boldsymbol{x}, y)}{\partial f_n(\boldsymbol{x}; \Theta)} \frac{\partial f_n(\boldsymbol{x}; \Theta)}{\partial \Theta} \,. \tag{5.7}$$

Here, the gradient term that depends on the decision tree is given by

$$\frac{\partial L(\Theta, \boldsymbol{\pi}; \boldsymbol{x}, y)}{\partial f_n(\boldsymbol{x}; \Theta)} = d_n(\boldsymbol{x}; \Theta) A_{n_r} - \bar{d}_n(\boldsymbol{x}; \Theta) A_{n_l} \,, \tag{5.8}$$

where n_l and n_r indicate the left and right child of node n, respectively, and

$$A_m = \frac{\sum_{\ell\in\mathcal{L}_m} \pi_{\ell y} \mu_\ell(\boldsymbol{x}|\Theta)}{\mathbb{P}_T[y|\boldsymbol{x}, \Theta, \boldsymbol{\pi}]} \,,$$

Here, $\mathcal{L}_m \subseteq \mathcal{L}$ denotes the subset of leaves of the subtree rooted in m. Detailed derivations of (5.8) can be found in the Appendix. Moreover, we describe in Section 5.2.4 how it can be efficiently computed with a single pass through the tree.

As a final remark, we considered also an alternative optimization procedure to SGD, namely Resilient Back-Propagation [86], which automatically adapts a specific learning rate for each parameter based on the sign change of its risk partial derivative over the last iteration.

Learning prediction nodes

We consider the problem of minimizing (5.5) with respect to $\boldsymbol{\pi}$ when Θ is fixed, *i.e.*

$$\min_{\boldsymbol{\pi}} R(\Theta, \boldsymbol{\pi}; \mathcal{T}) \,. \tag{5.9}$$

This is a *convex* optimization problem and a global solution can be easily recovered. A similar problem has already been encountered in the context of decision trees in [88], but at the level of a single node. In our case, however, the whole tree is taken into account, for we jointly estimate all the leaf predictions.

In order to compute a global minimizer of (5.9) we propose the following iterative scheme:

$$\pi_{\ell y}^{(t+1)} = \frac{1}{Z_\ell^{(t)}} \sum_{(\boldsymbol{x},y')\in\mathcal{T}} \frac{\mathbb{1}_{y=y'} \pi_{\ell y}^{(t)} \mu_\ell(\boldsymbol{x}|\Theta)}{\mathbb{P}_T[y|\boldsymbol{x}_i, \Theta, \boldsymbol{\pi}^{(t)}]} \,, \tag{5.10}$$

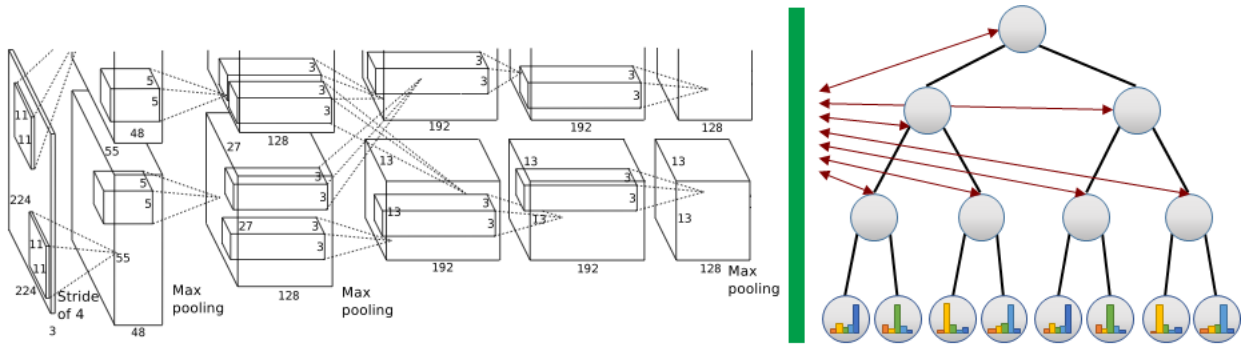


Figure 5.5: Schematic illustration of building blocks for representation learning (left Figure, taken from [66]) and proposed, differentiable decision tree. In the example the fully connected layers (+softmax) are replaced by a tree. Please note that split nodes can be attached to any part of the network.

for all $\ell \in \mathcal{L}$ and $y \in \mathcal{Y}$, where $Z_\ell^{(t)}$ is a normalizing factor ensuring that $\sum_y \pi_{\ell y} = 1$. The starting point $\boldsymbol{\pi}^{(0)}$ can be arbitrary as long as every element is positive. A typical choice is to start from the uniform distribution in all leaves, *i.e.* $\pi_{\ell y} = |\mathcal{Y}|^{-1}$. It is interesting to note that the update rule in (5.10) is step-size free and it guarantees a strict decrease of the risk at each update until a fixed-point is reached (see proof in supplementary material).

As opposed to the update strategy for Θ, which is based on mini-batches, we adopt an offline learning approach to obtain a more reliable estimate of $\boldsymbol{\pi}$, because suboptimal predictions in the leaves have a strong impact on the final prediction. Moreover, we interleave the update of $\boldsymbol{\pi}$ with a whole epoch of stochastic updates of Θ as described in the previous subsection.

5.2.4 Learning backpropagation forests (BPFs)

Thus far we have assumed to deal with a single decision tree. Now, we consider an ensemble of trees $\mathcal{F}$, where all trees can possibly share the same parameter Θ, but each tree can have a different structure with a different set of decision functions (yet defined as in (5.3)), and independent leaf predictions $\boldsymbol{\pi}$.

Since each tree in the forest $\mathcal{F}$ has its own set of parameters $\boldsymbol{\pi}$, we can update the prediction nodes of each tree independently as described in Subsection 5.2.3, given the actual estimate of Θ.

As for Θ, instead, we randomly select a tree in $\mathcal{F}$ for each mini-batch and then we proceed as detailed in Subsection 5.2.3 for the SGD update. This strategy resembles some similarities with the way Dropout works [97], where each SGD update is potentially applied to a different network topology, which is sampled according to a specific distribution.

During test time, as shown in (5.4), the prediction delivered by each tree is averaged to produce the final outcome.

The BPF learning procedure

The learning procedure is summarized in Algorithm 5.2.1. We start with a random initialization of the decision nodes' parameter Θ and iterate the learning procedure for a pre-determined number of

epochs, given a training set $\mathcal{T}$. At each epoch, we initially obtain an estimation of the prediction nodes' parameters $\boldsymbol{\pi}$ given the actual value of Θ by running the iterative scheme in (5.10), starting from the uniform distribution in each leaf, *i.e.* $\pi_{\ell y}^{(0)} = |\mathcal{Y}|^{-1}$. Then we split the training set into a random sequence of mini-batches and we perform for each mini-batch a SGD update of Θ as in (5.6). After each epoch we might eventually change the learning rate according to pre-determined schedules.

More details about the computation of some tree-specific terms are given in the next section.

Algorithm 5.2.1 Learning DT by backpropagation

Require: $\mathcal{T}$: training set, nEpochs

1: random initialization of Θ
2: **for all** $i \in \{1, \ldots, \text{nEpochs}\}$ **do**
3: Compute $\boldsymbol{\pi}$ by iterating (5.10)
4: break $\mathcal{T}$ into a set of random mini-batches
5: **for all** $\mathcal{B}$: mini-batch from $\mathcal{T}$ **do**
6: Update $\boldsymbol{\Theta}$ by SGD step in (5.6)
7: **end for**
8: **end for**

BPF implementation notes

Decision nodes We have defined decision functions d_n in terms of real-valued functions $f_n(\cdot; \Theta)$, which are not necessarily independent, but coupled through the shared parametrization Θ. Our intention is to endow the tree with feature learning capabilities by letting the functions f_n to be embedded within a deep convolutional neural network with parameters Θ. In the specific, we can regard each function f_n as a linear, output unit of a deep network that will be turned into a probabilistic routing decision by the action of d_n, which embeds a sigmoid activation to obtain a response in the $[0, 1]$ range. Figure 5.5 reports a schematic illustration of this idea, where the units of the output layer of a deep network is connected to the decision functions of a decision tree, which are arranged into a hierarchy. During the forward pass, the sample produces soft activations of the routing decisions of the tree that induce via the routing function a mixture of leaf predictions as per (5.1).

Note finally that by assuming the functions $f_n(\boldsymbol{x}; \boldsymbol{\theta}_n) = \boldsymbol{\theta}_n^\top \boldsymbol{x}$ linear and independent (by having separate parametrizations), we recover a model that resembles an oblique forests [55].

Routing function μ_ℓ

The computation of the routing function can be carried out by traversing the tree once. Let $\top \in \mathcal{N}$ be the root node and for each node $n \in \mathcal{N}$ let n_l and n_r denote its left and right child, respectively. We start from the root by setting $\mu_\top = 1$ and for each node $n \in \mathcal{N}$ that we visit in breath-first order we set

$$\mu_{n_l} = d_n(\boldsymbol{x}; \Theta)\mu_n$$

and

$$\mu_{n_r} = \bar{d}_n(\boldsymbol{x}; \Theta)\mu_n \, .$$

At the end, we can read from the leaves the desired values of the routing function.

Learning Decision Nodes

The forward pass of the backpropagation algorithm precomputes the values of the routing function $\mu_\ell(\boldsymbol{x}; \Theta)$ and the value of the tree prediction $\mathbb{P}_T[y|\boldsymbol{x}, \Theta, \boldsymbol{\pi}]$ for each sample $(\boldsymbol{x}, y)$ in the mini-batch $\mathcal{B}$. The backward pass requires the computation of the gradient term in (5.8) for each sample $(\boldsymbol{x}, y)$ in the mini-batch. This can be carried out by a single, bottom-up tree traversal. We start by setting

$$A_\ell = \frac{\pi_{\ell y}\mu_\ell(\boldsymbol{x}; \Theta)}{\mathbb{P}_T[y|\boldsymbol{x}, \Theta, \boldsymbol{\pi}]}$$

for each $\ell \in \mathcal{L}$. The we visit the tree in reversed breadth-first order (bottom-up). Once in a node $n \in \mathcal{N}$, we can compute the partial derivative in (5.8) since we can read A_{n_l} and A_{n_r} from the children, and we set

$$A_n = A_{n_l} + A_{n_r} \, ,$$

which will be required by the parent node.

Learning Prediction Nodes

Before starting the iterations in (5.10), we precomputed $\mu_\ell(\boldsymbol{x}; \Theta)$ for each $\ell \in \mathcal{L}$ and for each sample $\boldsymbol{x}$ in the training set, as detailed in Subsection 5.2.4. The iterative scheme requires few iterations to converge to a solution with an acceptable accuracy (20 iterations were enough for all our experiments).

5.2.5 Comparison of BPFs to conventional forest classifiers

Our experiments illustrate both the performance of back-propagation forests (BPFs) as standalone classifiers, as well as their usefulness in improving the classification performance of an end-to-end classification pipeline on top of a deep, convolutional neural network. To this end, we evaluate our proposed methods on diverse datasets, covering a broad range of classification tasks (ranging from simple binary classification of synthetically generated data up to large-scale image recognition on the 1000-class ImageNet dataset).

We first compared BPFs against the state-of-the-art in terms of stand-alone, off-the-shelf forest ensembles. We used the 5 datasets in [93] to compare the accuracy of backprop forests to that of Alternating Decision Forests. For ADF, we provide results reported in their paper and we use their maximal tree depth and forest size as an upper bound on the size of our models. Essentially, for each of the datasets, all our trees are less deep and there are fewer of them than in the corresponding ADF models. We used ensembles of different sizes depending on the size of the dataset and the complexity of the learning tasks. Though all the trees are allowed to become non-balanced (which we control via monitoring the loss in the leaves and dub *flexible* in what follows), the percentage of nodes that are split at each iteration varies by the dataset - the default is around 5%. In all cases, we use RPROP and the recommended hyper-parameters of the original publication for split

node parameter optimization. We reported the means and standard deviations resulting from 10 repetitions of the experiment. We have not used random subspace selection in the split nodes as this did not yield considerable performance improvements.

	G50C	***Letter***	***USPS***	***MNIST***	***Char74k***
ADF mean	18.71	**3.52**	**5.59**	2.71	**16.67**
ADF stdev	1.27	0.12	0.16	0.1	0.21
BPF mean	17.4	**2.92**	**5.01**	2.8	**16.04**
BPF stdev	1.52	0.17	0.24	0.12	0.2
Num Train Samples	50	16000	7291	60000	66707
Num Test Samples	500	4000	2007	10000	7400
Num Classes	2	26	10	10	62
Num Input dimensions	50	16	256	784	64
Tree input features	10 (random)	8 (random)	10x10 patches	15x15 patches	10 (random)
Depth	5	10	10	10	12
Flexible	yes	yes	yes	yes	yes
% Increase in nodes	5%	1%	5%	5%	5%
Number of trees	50	70	100	80	200
Regularization	rprop	rprop	rprop	rprop	rprop
Batch size	25	500	250	1000	1000

Table 5.2: Comparison to other forest-based classifiers

The results of this experiment are summarized in Table 5.2 The $G50c$ dataset, used by [96], is the smallest and deals with just a two-class problem, so training trees of depth 5 was sufficient for this task. Each tree uses only a randomly selected set of 10 out of the 50 features. For the 26-class letter data [41] we used an ensemble of 70 trees of depth 10, 8 randomly selected features per tree and larger batches of 500 samples. We only split 1% of the nodes. For the USPS [58] and MNIST [71] datasets, we use randomly selected patches of sizes 10x10 and 15x15, trees of depth 10, ensembles of 100 and 80 trees and batches of 250 and 1000, respectively. Finally, since the Char74k dataset [23] presents a more difficult 62-class problem, we allow a maximum depth of 12. We achieved high performance with a 200 tree ensemble, which is smaller than the 300 tree ADF ensemble. Overall, we outperform ADF trees, though significant results, with p-values less than 0.05, were obtained for the Letter, USPS and Char74k datasets.

MNIST

We used the same dataset as described in the previous section, however, this time we used a convolutional network architecture in conjunction with our BPF. The baseline architecture implementation uses a softmax as output layer, yielding an error of 0.9%. Using our proposed BPF on top, we can reduce the classification error to 0.7%. The ensemble size was fixed to 10 trees, each with a depth of 5.

5.3 Deep Convolutional Neural Decision Forests

5.3.1 Improving performance using DNNs + BPFs

We used the MatConvNet library [105] for building an end-to-end image classification pipeline to test on MNIST, replacing the conventionally used softmax layer by our back-propagation forests as classifiers. Our listed baseline results are the scores we obtained from re-running the provided example architectures including the respective settings for optimization, hyper-parameters, *e.t.c.* . Please note the positive effect on MNIST performance compared to Section 5.2.5 when spending additional layers on representation learning.

5.3.2 ImageNet experiments

We used BackPropagation forests to improve the performance of the GoogLeNet network described in [102]. This network has a reported error of 10.07% when using only a single model and only the center crop for training. The starting point for our experiment was the GoogLeNet 'baseline', implemented using the Distributed (Deep) Machine Learning Common (DMLC) library [73][3]. We used just one crop and a single model. The data was scaled and centered, but no augmentation was used. By default, the model uses 3 softmax layers at different stages of the network to encourage the construction of informative features early on. Each of these softmax layers gets their input from a fully connected (FC) layer, built on top of an Average Pool layer, which in turn is built on top of a corresponding Concat layer. Let DC0, DC1 and DC2 be the concat layers preceding each of the softmax layers in GoogLeNet. Let AvgPool0, AvgPool1 and AvgPool2 be the average pool layers preceding these softmax layers. In order to avoid problems with propagation of gradients given the depth of the network and in order to provide the final classification layers with the features obtained in the early stages of the pipeline, we have also supplied DC0 as input to AvgPool1 and AvgPool2 and DC1 as input to AvgPool2. We call this network 'Base2' and its Top5 error is 10.02%.

To achieve higher performance, we follow the same procedure as before (and simply replace all softmax layers with a BPF), obtaining a structure which we call the *BeefNet*. As in the previous experiments, the BPF layers have one neuron fused with every node of every tree in the ensemble. These tree neurons have inputs from all the nodes of the previous layers. For this experiment, each BPF layer uses an ensemble of 10 trees and each tree is a balanced depth 15 tree. BeefNet obtained a test performance on the validation data of 7.84%, which is notable since we are using a single crop and a single model only. Please mind that the only difference to 'Base2' is installing our BPF forests. We offer more details concerning the Base2 and BeefNet architecture in the supplementary material. We have not used regularization, subspace selection of dropout for this network, this being subject of future research, as is the use of additional crops and models.

In this chapter we have shown how to model and train stochastic, differentiable decision trees and enriched them with the ability to account for representation learning, akin to (deep) convolutional neural networks. Prevailing approaches for decision tree training typically operate in a greedy and local (mostly split node-specific) way, making it impossible to be used for representation learning. To overcome this problem, we presented a way to support stochastic routing in

[3] https://github.com/dmlc/cxxnet.git

decision trees and performing split node parameter learning via back-propagation. Moreover, we introduced a globally optimal procedure for learning leaf node prediction parameters, *i.e.* we populate leaf nodes with their optimal predictors, given the state of the network and all available training data. We have successfully validated our new decision forest model as stand-alone classifier on standard machine learning datasets. Finally, integrating our new decision tree/forest model as classifier in the publicly available implementation of the GoogLeNet deep network architecture (single model setting) on ImageNet has given a Top5 error of only 7.84% *without any form of data set augmentation*. There are several directions to be explored in order to gain further improvements and more practical insights, such as using multi-network architectures and data augmentation.

Chapter 6

Summary

In this thesis, we have shown that it is possible to identify low-dimensional structures in complex high-dimensional data, if such structures exist. We have leveraged these underlying structures to construct compact interpretable models for various machine learning tasks.

We have formalized the problem of finding Informative Projections from data and constructing ensembles of low-dimensional solvers, such that points are assigned to different solvers based on a selection function. We have studied the properties of such ensembles and shown faster consistency for an ensemble of low-dimensional k-nn classifiers than achievable with the standard k-nn model. We have phrased informative projection retrieval as a combinatorial problem through the computation of a matrix of point-wise loss estimators. The result of the optimization problem is the assignment of points to the projections which can most successfully handle them. We have also introduced several ways of solving the combinatorial problem and building the ensembles: first, an ILP which obtains the optimal solution to the combinatorial problem, second, a two-step convex procedure, which can also be applied iteratively, that obtains a solution through ℓ_1 norm regularization and finally, a greedy solution which comes with optimality guarantees due to the submodularity of the objective. As shown by our experiments, all three methods yield accurate but compact models when the subsets of the data are separable in low-dimensional spaces.

Although our main focus is classification, the technique has also proved useful when adapted to regression problems or clustering. In addition, we have implemented an extension which performs active learning by leveraging low-dimensional models. We applied the set of informative projection tools to the clinical problem of adjudicating artifacts from alerts issued by a vital sign monitoring system. The retrieved patterns were not only deemed informative by clinicians, but also served as a starting point for rules of filtering artifacts. We have done studies involving expert clinicians which demonstrate that the low-dimensional models can be used to automatically classify a large part of the samples, leaving only a handful for human annotation, resulting in a considerable decrease in labeling effort. Also successful was the use of projections to guide data acquisition for a radiation classification system. Our method identified several areas where more data was required to improve the accuracy of random forest classifiers. Once the data was collected, the performance of the system did increase considerably more than it would have with random collection.

Furthermore, we have leveraged the notion of query-specific models to improve the performance of random forests and introduced the notion of back-propagation forests. The latter can be coupled with deep network architectures, where each tree in the forest is associated with a different set of learned representations and samples. Thus, we have obtained a highly accurate classification system, improving over the the state-of-the-art in vision tasks and other ML problems.

Bibliography

[1] Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *(NC)*, 9 (7):1545–1588, 1997. 5.2.1

[2] Francis Bach, Rodolphe Jenatton, Julien Mairal, and Guillaume Obozinski. Structured sparsity through convex optimization. *Statistical Science*, 27(4):450–468, 11 2012. doi: 10.1214/12-STS394. URL http://dx.doi.org/10.1214/12-STS394. 1.3

[3] Francis Bach, Rodolphe Jenatton, Julien Mairal, and Guillaume Obozinski. Structured sparsity through convex optimization. *Statistical Science*, 27(4):450–468, 2012. 1.5

[4] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009. 1.5

[5] Aharon Ben-Tal and Arkadi Nemirovski. *Lectures on Modern Convex Optimization*. Society for Industrial and Applied Mathematics (SIAM), 2001. 2.7.1

[6] Radu Berinde, Piotr Indyk, and Milan Ruzic. Practical near-optimal sparse recovery in the l1 norm. In *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, pages 198–205. IEEE, 2008. 1.5

[7] Alina Beygelzimer, Sanjoy Dasgupta, and John Langford. Importance weighted active learning. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, pages 49–56. ACM, 2009. 3.1.1

[8] Mustafa Bilgic. Combining active learning and dynamic dimensionality reduction. In *SDM*, pages 696–707, 2012. 3.1.1

[9] C. M. Bishop and M. Svensén. Bayesian hierarchical mixtures of experts. In *Proc. of Conference on Uncertainty in Artificial Intelligence*, page 5764, 2003. 5.2.1

[10] Christopher M Bishop and Markus Svenskn. Bayesian hierarchical mixtures of experts. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 57–64. Morgan Kaufmann Publishers Inc., 2002. 2

[11] Christopher M Bishop and Michael E Tipping. A hierarchical latent variable model for data visualization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20 (3):281–293, 1998. 1.3, 1.5

[12] Avrim L. Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(12):245 – 271, 1997. ISSN 0004-3702. doi: http://dx.doi.org/10.1016/S0004-3702(97)00063-5. URL http://www.sciencedirect.com/science/article/pii/S0004370297000635. Relevance. 2.1

[13] A. Bosch, A. Zisserman, and X. Muñoz. Image classification using random forests and

ferns. In *(ICCV)*, pages 1–8, 2007. 5.2.1

[14] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. 5.2.1

[15] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984. 2.1, 5.1

[16] Gabriel J. Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. Segmentation and recognition using structure from motion point clouds. In *(ECCV)*, pages 44–57. 2008. 5.2.1

[17] Emmanuel J Candes, Justin K Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on pure and applied mathematics*, 59(8):1207–1223, 2006. 1.5

[18] R. Caruana, N. Karampatziakis, and A. Yessenalina. An empirical evaluation of supervised learning in high dimensions. In *(ICML)*, pages 96–103, 2008. 5.2.1

[19] Thomas Cover. Rates of convergence for nearest neighbor procedures. *Proceedings of The Hawaii International Conference on System Sciences*, 1967. 2.5.2

[20] Thomas M Cover and Peter E Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27, 1967. 2.5.2

[21] A. Criminisi and J. Shotton. *Decision Forests in Computer Vision and Medical Image Analysis*. Springer, 2013. 5.2.1

[22] Aron Culotta and Andrew McCallum. Reducing labeling effort for structured prediction tasks. In *AAAI/IAAI*, pages 746–751, 2005. 3.1.1

[23] T. E. de Campos, B. R. Babu, and M. Varma. Character recognition in natural images. In *Proceedings of the International Conference on Computer Vision Theory and Applications, Lisbon, Portugal*, February 2009. 5.2.5

[24] Luc Devroye, Laszlo Gyorfi, Adam Krzyzak, and Gabor Lugosi. On the strong universal consistency of nearest neighbor regression function estimates. *The Annals of Statistics*, pages 1371–1385, 1994. 2.5.2

[25] Luc Devroye, László Györfi, and Gábor Lugosi. *A probabilistic theory of pattern recognition*. Springer, 1996. 2.4.5

[26] Pedro Domingos. Knowledge discovery via multiple models. *Intelligent Data Analysis*, 2: 187–202, 1998. 1.5

[27] Pinar Donmez, Jaime G. Carbonell, and Jeff Schneider. Efficiently learning the accuracy of labeling sources for selective sampling. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 259–268, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-495-9. doi: 10.1145/1557019.1557053. URL `http://doi.acm.org/10.1145/1557019.1557053`. 3.1.1

[28] Eulanda M. Dos Santos, Robert Sabourin, and Patrick Maupin. A dynamic overproduce-and-choose strategy for the selection of classifier ensembles. *Pattern Recogn.*, 41:2993–3009, October 2008. ISSN 0031-3203. doi: 10.1016/j.patcog.2008.03.027. URL `http://dl.acm.org/citation.cfm?id=1385702.1385963`. 1.5

[29] Artur Dubrawski, Saswati Ray, Peter Huggins, Simon Labov, and K Nelson. Diagnosing machine learning-based nuclear evaluation system. In *Proceedings of the IEEE NSS*, 2012.

3.3.1

[30] Meir Feder and Neri Merhav. Relations between entropy and error probability. *IEEE Transactions on Information Theory*, 40(1):259–266, January 1994. doi: 10.1109/18.272494. URL http://dx.doi.org/10.1109/18.272494. 2.4.1

[31] Andrej Karpathy Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *(CVPR)*, 2015. 5.2.1

[32] Mario A. T. Figueiredo and Anil K. Jain. Unsupervised learning of finite mixture models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(3):381–396, 2002. 1.5

[33] Madalina Fiterau and Artur Dubrawski. Projection retrieval for classification. In *Advances in Neural Information Processing Systems 25 (NIPS)*, pages 3032–3040, 2012. 2.2, 6.2

[34] Madalina Fiterau and Artur Dubrawski. Projection retrieval for classification. In *Advances in Neural Information Processing Systems 25*, pages 3032–3040, 2012. 1.3, 2

[35] Madalina Fiterau and Artur Dubrawski. Informative projection recovery for classification, clustering and regression. In *International Conference on Machine Learning and Applications*, volume 12, 2013. 1.3, 2.1, 2.3.3, 3.1.1, 3.3.2

[36] Madalina Fiterau, Artur Dubrawski, Lujie Chen, Marilyn Hravnak, Clermont Gilles, and Michael R. Pinsky. Automatic identification of artifacts in monitoring critically ill patients. In *Annual Congress of the European Society of Intensive Care Medicine*, volume 39, page S470, 2013. 2.1

[37] Madalina Fiterau, Artur Dubrawski, Lujie Chen, Marilyn Hravnak, Clermont Gilles, Eliezer Bose, and Michael R. Pinsky. Artifact adjudication for vital sign step-down unit data can be improved using active learning with low-dimensional models. In *Annual Congress of the European Society of Intensive Care Medicine*, 2014. 2.1

[38] Madalina Fiterau, Artur Dubrawski, Lujie Chen, Marilyn Hravnak, Eliezer Bose, Clermont Gilles, and Michael R. Pinsky. Archetyping artifacts in monitored noninvasive vital signs data. In *Society of Critical Care Medicine Annual Congress*, 2015. 2.1

[39] Evelyn Fix and Joseph L Hodges Jr. Discriminatory analysis-nonparametric discrimination: consistency properties. Technical report, DTIC Document, 1951. 2.5.2

[40] Alex A. Freitas. Comprehensible classification models: A position paper. *SIGKDD Explor. Newsl.*, 15(1):1–10, March 2014. ISSN 1931-0145. doi: 10.1145/2594473.2594475. URL http://doi.acm.org/10.1145/2594473.2594475. 2

[41] Peter W. Frey and David J. Slate. Letter recognition using holland-style adaptive classifiers. *Mach. Learn.*, 6(2):161–182, March 1991. ISSN 0885-6125. doi: 10.1023/A:1022606404104. URL http://dx.doi.org/10.1023/A:1022606404104. 5.2.5

[42] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, September 1977. ISSN 0098-3500. doi: 10.1145/355744.355745. URL http://doi.acm.org/10.1145/355744.355745. 2.7.1

[43] Jozsef Fritz. Distribution-free exponential error bound for nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 21(5):552–557, 1975. 2.5.2

[44] Rahul Garg and Rohit Khandekar. Gradient descent with sparsification: an iterative algorithm for sparse recovery with restricted isometry property. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 337–344. ACM, 2009. 1.5

[45] Ran Gilad-Bachrach, Amir Navot, and Naftali Tishby. Query by committee made real. In *Advances in Neural Information Processing Systems 18 (NIPS)*, 2005. 3.1.1

[46] Nicholas Gisolfi, Madalina Fiterau, and Artur Dubrawski. Finding gaps in data to guide development of a radiation threat adjudication system. In *Symposium on Radiation Measurements and Applications*, 2014. 2.1

[47] Nicholas Gisolfi, Madalina Fiterau, and Artur Dubrawski. Finding meaningful gaps to guide data acquisition for a radiation adjudication system. In *Conference of the Association for the Advancement of Artificial Intelligence*, volume 29, 2015. 2.1

[48] Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *The Journal of Machine Learning Research*, 999999:2211–2268, 2011. 1.5

[49] Pinghua Gong, Jieping Ye, and Changshui Zhang. Multi-stage multi-task feature learning. *Journal of Machine Learning Research*, 14:2979–3010, 2013. 1.5

[50] Quanquan Gu, Zhenhui Li, and Jiawei Han. Joint feature selection and subspace learning, 2011. URL `https://www.aaai.org/ocs/index.php/IJCAI/IJCAI11/paper/view/2910`. 1.5

[51] Yuhong Guo and Russell Greiner. Optimistic active-learning using mutual information. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 823–829, 2007. 3.1.1

[52] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, 2009. 5.2.1

[53] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015. 5.2.1

[54] Xiaofei He, Deng Cai, Yuanlong Shao, Hujun Bao, and Jiawei Han. Laplacian regularized gaussian mixture model for data clustering. *IEEE Transactions on Knowledge and Data Engineering*, 2010. 1.3

[55] David Heath, Simon Kasif, and Steven Salzberg. Induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2(2):1–32, 1993. 5.2.2, 5.2.4

[56] Junzhou Huang, Tong Zhang, and Dimitris Metaxas. Learning with structured sparsity. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 417–424, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-516-1. doi: 10.1145/1553374.1553429. URL `http://doi.acm.org/10.1145/1553374.1553429`. 1.3

[57] Junzhou Huang, Tong Zhang, and Dimitris Metaxas. Learning with structured sparsity. *The Journal of Machine Learning Research*, 12:3371–3412, 2011. 1.5

[58] Jonathan J. Hull. A database for handwritten text recognition research. *(PAMI)*, 16(5): 550–554, 1994. 5.2.5

[59] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991. 2

[60] Jeremy Jancsary, Sebastian Nowozin, and Carsten Rother. Loss-specific training of non-parametric image restoration models: A new state of the art. In *(ECCV)*, 2012. 5.2.1

[61] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 5.2.1

[62] Michael I. Jordan. Hierarchical mixtures of experts and the em algorithm. *(NC)*, 6:181–214, 1994. 5.2.1

[63] Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994. 2

[64] P. Kontschieder, P. Kohli, J. Shotton, and A. Criminisi. GeoF: Geodesic forests for learning coupled predictors. In *(CVPR)*, pages 65–72, 2013. 5.2.1

[65] Christine Körner and Stefan Wrobel. Multi-class ensemble-based active learning. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *Proceedings of the 17th European Conference on Machine Learning (ECML)*, volume 4212, pages 687–694. Springer, 2006. 3.1.1

[66] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. Imagenet classification with deep convolutional neural networks. In *(NIPS)*, 2012. (document), 5.2.1, 5.5

[67] Sanjeev R Kulkarni and Steven E Posner. Rates of convergence of nearest neighbor estimation under arbitrary sampling. *Information Theory, IEEE Transactions on*, 41(4):1028–1039, 1995. 2.5.2

[68] Martin O Larsson and Johan Ugander. A concave regularization technique for sparse mixture models. In *Advances in Neural Information Processing Systems*, pages 1890–1898, 2011. 1.3

[69] Martin HC Law and Anil K Jain. Incremental nonlinear dimensionality reduction by manifold learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(3): 377–391, 2006. 1.5

[70] Neil D. Lawrence. Hierarchical gaussian process latent variable models. In *In International Conference in Machine Learning*, 2007. 1.5

[71] Yann Lecun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998. 5.2.5

[72] Mu Li, Li Zhou, Zichao Yang, Aaron Li, Fei Xia, David G Andersen, and AJ Smola. Parameter server for distributed machine learning. In *Big Learning NIPS Workshop*, 2013. 6.7.2

[73] Mu Li, David G. Andersen, Jun Woo Park, Alexander J. Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J. Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pages 583–598, Broomfield, CO, October 2014. USENIX Association. ISBN 978-1-931971-16-4. URL `https://www.usenix.org/`

conference/osdi14/technical-sessions/presentation/li_mu. 5.3.2, 6.7.2

[74] Percy Liang, Hal Daumé III, and Dan Klein. Structure compilation: trading structure for features. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 592–599. ACM, 2008. 3.1.1

[75] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *CoRR*, abs/1312.4400, 2013. 5.2.1

[76] Jun Liu, Lei Yuan, and Jieping Ye. An efficient algorithm for a class of fused lasso problems. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 323–332. ACM, 2010. 1.5

[77] Karim Lounici, Massimiliano Pontil, Alexandre B Tsybakov, and Sara van de Geer. Taking advantage of sparsity in multi-task learning. *arXiv preprint arXiv:0903.1468*, 2009. 1.5

[78] Luis Carlos Molina, Lluís Belanche, and Àngela Nebot. Feature selection algorithms: A survey and experimental evaluation. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 306–313. IEEE, 2002. 2.1

[79] A. Montillo, J. Tu, J. Shotton, J. Winn, J. E. Iglesias, D. N. Metaxas, and A. Criminisi. Entangled forests and differentiable information gain maximization. In *Decision Forests in Computer Vision and Medical Image Analysis*. 2013. 5.2.1

[80] Albert Montillo, Jamie Shotton, John Winn, Juan Eugenio Iglesias, Dimitri Metaxas, and Antonio Criminisi. Entangled decision forests and their application for semantic segmentation of ct images. In *(IPMI)*, pages 184–196, 2011. 5.2.1

[81] Mark A Musen, Blackford Middleton, and Robert A Greenes. Clinical decision-support systems. In *Biomedical informatics*, pages 643–674. Springer, 2014. 2

[82] G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14(1):265–294, 1978. ISSN 0025-5610. doi: 10.1007/BF01588971. URL http://dx.doi.org/10.1007/BF01588971. 2.3.3

[83] Barnabás Poczós and Jeff G. Schneider. On the estimation of alpha-divergences. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 609–617, 2011. 2.4.1

[84] Hema Raghavan, Omid Madani, and Rosie Jones. Active learning with feedback on features and instances. *Journal of Machine Learning Research*, 7:1655–1686, 2006. 3.1.1

[85] Parisa Rashidi and Diane J. Cook. Ask me better questions: active learning queries based on rule induction. In Chid Apt, Joydeep Ghosh, and Padhraic Smyth, editors, *KDD*, pages 904–912. ACM, 2011. ISBN 978-1-4503-0813-7. 3.1.1

[86] M. Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *IEEE Conf. on Neural Networks*, 1993. 5.2.3

[87] Alessandro Rinaldo. Properties and refinements of the fused lasso. *The Annals of Statistics*, 37(5B):2922–2952, 2009. 1.5

[88] Samuel Rota Bulò and Peter Kontschieder. Neural decision forests for semantic image

labelling. In *(CVPR)*, 2014. 5.2.1, 5.2.3

[89] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 2014. doi: 10.1007/s11263-015-0816-y. 6.7

[90] Norbert Sauer. On the density of families of sets. *Journal of Combinatorial Theory, Series A*, 13(1):145–147, 1972. 2.5.1

[91] Vicki L Sauter. *Decision Support Systems for business intelligence*. John Wiley & Sons, 2014. 2

[92] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *Artificial Neural NetworksICANN'97*, pages 583–588. Springer, 1997. 1.5

[93] Samuel Schulter, Paul Wohlhart, Christian Leistner, Amir Saffari, Peter M. Roth, and Horst Bischof. Alternating decision forests. In *(CVPR)*, 2013. 5.2.1, 5.2.5

[94] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009. 3.1.1

[95] Jamie Shotton, Ross Girshick, Andrew Fitzgibbon, Toby Sharp, Mat Cook, Mark Finocchio, Richard Moore, Pushmeet Kohli, Antonio Criminisi, Alex Kipman, and Andrew Blake. Efficient human pose estimation from single depth images. *(PAMI)*, 2013. 5.2.1

[96] Vikas Sindhwani, Partha Niyogi, and Mikhail Belkin. Beyond the point cloud: From transductive to semi-supervised learning. In *(ICML)*, pages 824–831. ACM, 2005. 5.2.5

[97] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. 5.2.4

[98] Nicolas Städler, Peter Bühlmann, and Sara Van De Geer. l 1-penalization for mixture regression models. *Test*, 19(2):209–256, 2010. 1.5

[99] Charles J Stone. Consistent nonparametric regression. *The annals of statistics*, pages 595–620, 1977. 2.5.2

[100] Alberto Suárez and James F. Lutsko. Globally optimal fuzzy decision trees for classification and regression. *(PAMI)*, 21(12):1297–1311, 1999. 5.2.1

[101] CE Sundberg, T Aulin, N Rydbeck, et al. The rate of convergence of k,-nn regression estimates and classification rules. *IEEE Trans. Commun*, 20:429–435, 1972. 2.5.2

[102] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. (document), 5.2.1, 5.3.2, 6.7, 6.7.1, 6.6

[103] Kai Ming Ting, Jonathan R. Wells, Swee Chuan Tan, Shyh Wei Teng, and Geoffrey I. Webb. Feature-subspace aggregating: ensembles for stable and unstable learners. *Machine Learning*, 82(3):375–397, 2011. ISSN 0885-6125. URL `http://dx.doi.org/10.1007/s10994-010-5224-5`. 1.5

[104] Simon Tong and Daphne Koller. Active learning for structure in bayesian networks. In

Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI), pages 863–869, 2001. 3.1.1

[105] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. *CoRR*, abs/1412.4564, 2014. 5.2.1, 5.3.1

[106] Terry J Wagner. Convergence of the nearest neighbor rule. *Information Theory, IEEE Transactions on*, 17(5):566–571, 1971. 2.5.2

[107] Yong Wang, Yuan Jiang, Yi Wu, and Zhi-Hua Zhou. Spectral clustering on multiple manifolds. *Neural Networks, IEEE Transactions on*, 22(7):1149–1161, 2011. 1.5

[108] Liyang Wei, Yongyi Yang, and Robert M. Nishikawa. Microcalcification classification assisted by content-based image retrieval for breast cancer diagnosis. *Pattern Recognition*, 42(6):1126 – 1132, 2009. ISSN 0031-3203. doi: http://dx.doi.org/10.1016/j.patcog.2008.08.028. URL `http://www.sciencedirect.com/science/article/pii/S0031320308003415`. Digital Image Processing and Pattern Recognition Techniques for the Detection of Cancer. 2

[109] D. Yu and L. Deng. *Automatic Speech Recognition: A Deep Learning Approach*. Springer, 2014. 5.2.1

[110] Lei Yuan, Jun Liu, and Jieping Ye. Efficient methods for overlapping group lasso. 2013. 1.5

[111] Tong Zhang. Analysis of multi-stage convex relaxation for sparse regularization. *The Journal of Machine Learning Research*, 11:1081–1107, 2010. 1.5

Appendix A

6.1 RIPR results on artificial data for supervised classification

Table 6.2 shows the classification accuracy for the standard RECIP method obtained for synthetic data. As expected, the observed performance is initially high when there are few known informative projections in data and it decreases noise and ambiguity of the injected patterns increase.

Most types of ensemble learners would use a voting scheme to arrive at final classification of a testing sample, rather than use a model selection scheme. For this reason, we have also compared predictive accuracy revealed by RECIP against a method based on majority voting among multiple candidate subspaces. Table **??** shows that the accuracy of this technique is lower than the accuracy of RECIP, regardless of whether the informative projections are recovered by the algorithm or assumed to be known a priori. This confirms the intuition that a selection-based approach can be more effective than voting for data which satisfies the subspace separability assumption.

For reference, we have also classified the synthetic data using K-Nearest-Neighbors algorithm using all available features at once. The results of that experiment are shown in Table **??**. Since RECIP uses neighbor information, K-NN is conceptually the closest among the popular alternatives. Compared to RECIP, K-NN performs worse when there are fewer synthetic patterns injected in data to form informative projections. It is because some features used then by K-NN are noisy. As more features become informative, the K-NN accuracy improves. This example shows the benefit of a selective approach to feature space and using a subset of the most explanatory projections to support not only explanatory analyses but also classification tasks in such circumstances.

6.2 RIPR results on artificial data for semi-supervised classification

To evaluate RIPR semi-supervised classification, we use the same type of synthetic data as in [33], but we obscure some labels before training to see if the projection recovery performance is maintained. The synthetic data for this section contains $P = 2$ informative projections and $M = 10$ features. Every projection has $N = 1,000$ data points which it can classify. There are also R noisy data points that cannot be classified by any projection; this parameter varies between experiments. Also variable is the proportion of unlabeled data. We start with fully labeled data, then for every u points in the training set we obscure one label, so for smaller u, the larger proportion of unlabeled data, and the harder the task.

Table 6.3 summarizes the accuracy of RIPR for semi-supervised classification using k-NN models on each of the projections. We call this method Ripped k-NN. We have included the

Table 6.1: Projection Recovery for Artificial Datasets with $1 \ldots 7$ informative features and noise level $0 \ldots 0.2$ in terms of mean and variance of *Precision* and *Recall*. Mean/var obtained for each setting by repeating the experiment with datasets with different informative projections.

PRECISION										
	Mean					**Variance**				
	0	***0.02***	***0.05***	***0.1***	***0.2***	***0***	***0.02***	***0.05***	***0.1***	***0.2***
1	1	1	1	0.9286	0.9286	0	0	0	0.0306	0.0306
2	1	1	1	1	1	0	0	0	0	0
3	1	1	1	1	1	0	0	0	0	0
4	1	1	1	1	1	0	0	0	0	0
5	1	1	1	1	1	0	0	0	0	0
6	1	1	1	1	1	0	0	0	0	0
7	1	1	1	1	1	0	0	0	0	0
RECALL										
	Mean					**Variance**				
	0	***0.02***	***0.05***	***0.1***	***0.2***	***0***	***0.02***	***0.05***	***0.1***	***0.2***
1	1	1	1	1	1	0	0	0	0	0
2	1	1	1	1	1	0	0	0	0	0
3	1	1	0.9524	0.9524	1	0	0	0.0136	0.0136	0
4	0.9643	0.9643	0.9643	0.9643	0.9286	0.0077	0.0077	0.0077	0.0077	0.0128
5	0.7714	0.7429	0.8286	0.8571	0.7714	0.0163	0.0196	0.0049	0.0082	0.0278
6	0.6429	0.6905	0.6905	0.6905	0.6905	0.0113	0.0113	0.0272	0.0113	0.0113
7	0.6327	0.5918	0.5918	0.5714	0.551	0.0225	0.02	0.0258	0.0233	0.02

performance of a k-NN model trained using all features. As expected, RIPR outperforms the high-dimensional model. Even though noise impacts RIPR performance, our technique performs better than k-NN even for $R = 1,000$. This improvement is not limited to k-NN classifiers: Similar results are obtained when comparing SVM regressors to their Ripped version. RIPR achieves very good precision and recall for all values of R, despite the noise and unlabeled data.

6.3 Active RIPR case study: Artifact Detection from Partially-Observed Vital Signals of Intensive Care Unit Patients

A feature of the RIPR algorithm is its tolerance to missing data. For a data point x, the values of the loss estimators are set to ∞ for all projections that involve missing values for x. This ensures that data tends to be explained using projections that have a full description for it, while projections with some missingness are not prefereable though not ignored. This new capability expands practical applicability of RIPR. The set of relevant examples includes a medical informatics application.

Recovery of meaningful, explainable models is fundamental for the clinical decision-making process. We work with a cardio-respiratory monitoring system designed to process multiple vital signs indicative of the current health status of a patient. The system issues an alert whenever some form of instability requires attention. In practice, a substantial fraction of these alerts are not due to real emergencies (true alerts), but instead are triggered by malfunctions or inaccuracies of the sensing equipment (artifacts). Each system-generated alert is associated with a vital sign that

Table 6.2: RECIP Classification Accuracy on Artificial Data

CLASSIFICATION ACCURACY										
	Mean					**Variance**				
	0	***0.02***	***0.05***	***0.1***	***0.2***	***0***	***0.02***	***0.05***	***0.1***	***0.2***
1	0.9751	0.9731	0.9686	0.9543	0.9420	0.0000	0.0000	0.0000	0.0008	0.0007
2	0.9333	0.9297	0.9227	0.9067	0.8946	0.0001	0.0001	0.0001	0.0001	0.0001
3	0.9053	0.8967	0.8764	0.8640	0.8618	0.0004	0.0005	0.0016	0.0028	0.0007
4	0.8725	0.8685	0.8589	0.8454	0.8187	0.0020	0.0020	0.0019	0.0025	0.0032
5	0.8113	0.8009	0.8105	0.8105	0.7782	0.0042	0.0044	0.0033	0.0036	0.0044
6	0.7655	0.7739	0.7669	0.7632	0.7511	0.0025	0.0021	0.0026	0.0025	0.0027
7	0.7534	0.7399	0.7347	0.7278	0.7205	0.0034	0.0040	0.0042	0.0042	0.0045
CLASSIFICATION ACCURACY - KNOWN PROJECTIONS										
	Mean					**Variance**				
	0	***0.02***	***0.05***	***0.1***	***0.2***	***0***	***0.02***	***0.05***	***0.1***	***0.2***
1	0.9751	0.9731	0.9686	0.9637	0.9514	0.0000	0.0000	0.0000	0.0001	0.0000
2	0.9333	0.9297	0.9227	0.9067	0.8946	0.0001	0.0001	0.0001	0.0001	0.0001
3	0.9053	0.8967	0.8914	0.8777	0.8618	0.0004	0.0005	0.0005	0.0007	0.0007
4	0.8820	0.8781	0.8657	0.8541	0.8331	0.0011	0.0011	0.0014	0.0014	0.0020
5	0.8714	0.8641	0.8523	0.8429	0.8209	0.0015	0.0015	0.0018	0.0019	0.0023
6	0.8566	0.8497	0.8377	0.8285	0.8074	0.0014	0.0015	0.0016	0.0023	0.0021
7	0.8429	0.8371	0.8256	0.8122	0.7988	0.0015	0.0018	0.0018	0.0021	0.0020

Table 6.3: Accuracy of semi-supervised RIPR on synthetic data compared to a k-NN model on all features and projection recovery.

	no u	***u=7***	***u=5***	***u=3***	***no u***	***u=7***	***u=5***	***u=3***
R	**Accuracy RIPR SSC**				**Accuracy k-NN**			
0	0.928	0.931	0.918	0.928	0.722	0.713	0.714	0.707
30	0.923	0.919	0.931	0.928	0.726	0.724	0.717	0.714
50	0.904	0.896	0.898	0.886	0.726	0.701	0.701	0.699
100	0.893	0.882	0.878	0.877	0.717	0.711	0.698	0.715
1000	0.688	0.687	0.693	0.705	0.627	0.621	0.612	0.607

initiated it: either heart rate (HR), respiratory rate (RR), blood pressure (BP), or peripheral arterial oxygen saturation (SpO_2). Here, we show as an example the analysis of respiratory rate alerts, i.e. we consider episodes when this vital sign was the first to exceed its control limits, triggering an alert. A modest subset of data was manually reviewed and labeled by clinicians, and true alerts were distinguished from apparent artifacts. Our aim was to learn an artifact-identification model and to apply it to data not yet labeled. The objective was to identify artifact alerts that can be dismissed on-the-fly to reduce the impact of alert fatigue among medical personnel and to enable improvements of the quality of care. We extracted multiple temporal features for each vital sign independently over duration of each alert and a window of 4 minutes preceding its onset. These features included metrics of data density, as well as common moving-window statistics computed for each of the vital timeseries.

Figure 6.1 shows the RIPR semi-supervised classification model obtained for the RR artifact detection. The features used are the data densities for HR, RR and SpO_2 and the minimum value of RR over a time window of observation. These retrieved models are consistent with the intuition of seasoned clinicians. The accuracy of the model is 97.8%, precision and recall for genuine

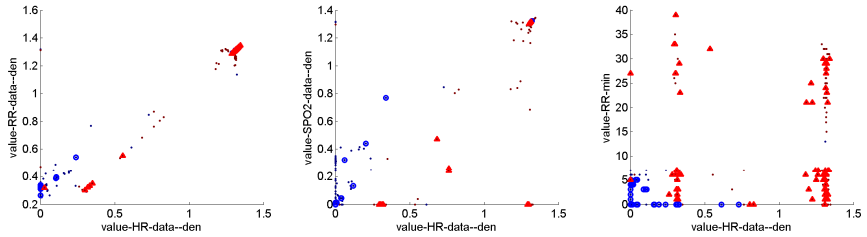


Figure 6.1: RIPR for Respiratory Rate alerts. Artifacts: Blue circles. True instability: Red triangles.

alert recovery are 97.9% and 99.1% respectively, all computed with leave-one-out cross-validation. Some instances were classified by the system as artifacts while domain experts initially considered them to be true alerts. Yet, on a closer visual inspection made possible by the low-dimensional RIPR projections, they were found to exhibit artifact-like characteristics. Further validation shown these instances to be labeled incorrectly in the original data.

Comparison of ActiveRIPR scoring functions on artificial data

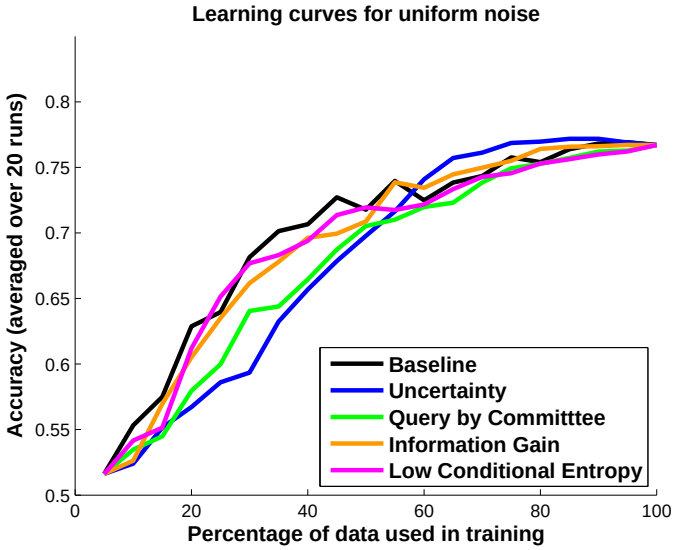


Figure 6.2: ActiveRIPR on artificial data with uniform noise.

At each iteration, a batch of 30 training data points is selected for expert labeling. We track the accuracy of the models at each iteration using hold-out test data. For this setup, model-conforming points will improve the model accuracy when selected for labeling. Each can be classified correctly using one of the informative projections, and thus the placement of the low losses they incur pinpoints the appropriate set of projections. On the other hand, non-conforming (noisy) data do not follow this pattern and tend to confuse the model as their labels are random. In view of this, we consider a baseline strategy of requesting labels for conforming points first. Clearly, for non-artificial data we would not be able to apply this since we would have no prior knowledge of which

data have noisy labels, but this baseline is an indicator of the upper limits of performance. When noise is distributed uniformly, this strategy is optimal, since all the samples that are labeled can actually be useful to the model.

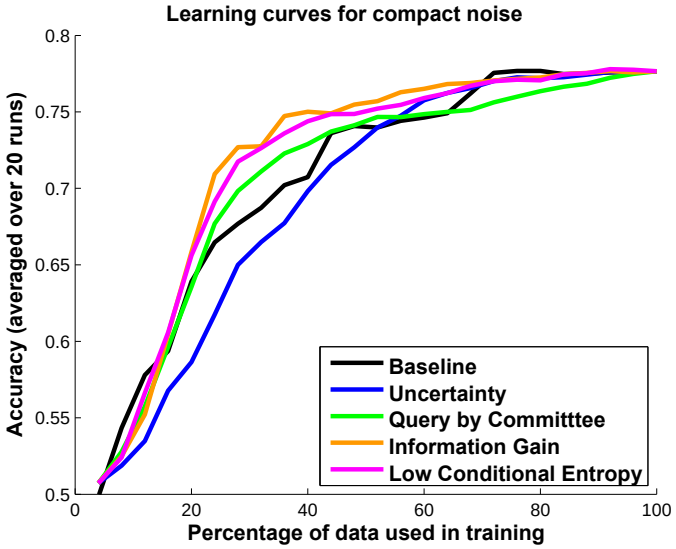


Figure 6.3: ActiveRIPR on artificial data with compact noise.

Figure 6.2 shows learning curves for ActiveRIPR using different scoring functions when the non-conforming points in the artificial data are distributed uniformly. All methods rebuild RIPR models from scratch after each batch of data is labeled. We do this to mitigate any model bias from previous iterations. The results confirm our intuition about the noisy samples: we can see that as long as model-conforming data is available for labeling, the baseline performs, overall, slightly better than the rest, while its performance saturates once only noisy data is available. Sampling by low conditional entropy and information gain perform well. Uncertainty sampling seems to pick out the non-conforming samples, unhelpful to the models.

It is apparent that little improvement can be brought to this type of data if the noise is distributed randomly. In fact, random sample selection does not perform significantly worse than either of the sampling methods used by ActiveRIPR. We now turn our attention to the case when the noise is distributed in more compact areas of the feature space. This time, the scoring functions we previously introduced prove useful, as shown in Figure 6.3. We keep the same baseline as in the previous experiment: the model-conforming samples are to be selected first. However, for compact noise, this strategy is no longer optimal as the model-conforming samples differ in their proximity to, or overlap with, the noisy part of the feature space. Although using model-conforming data helps briefly at first, the baseline is soon outperformed by information gain and low conditional entropy-based selection. On the other hand, uncertainty sampling performs poorly and query by committee is also not competitive.

These results are averaged over twenty executions of the algorithm with the same generated data, but with different starting samples. It turns out that the steepness of the curve differs considerably depending on the starting sample. For this reason, it is difficult to establish confidence bounds on accuracy with respect to data permutations. As a reminder, our algorithm is deterministic once the initial sample is set as this best answers the needs of our application. Nevertheless, we can determine whether the relative performance of the scoring techniques is consistent. We

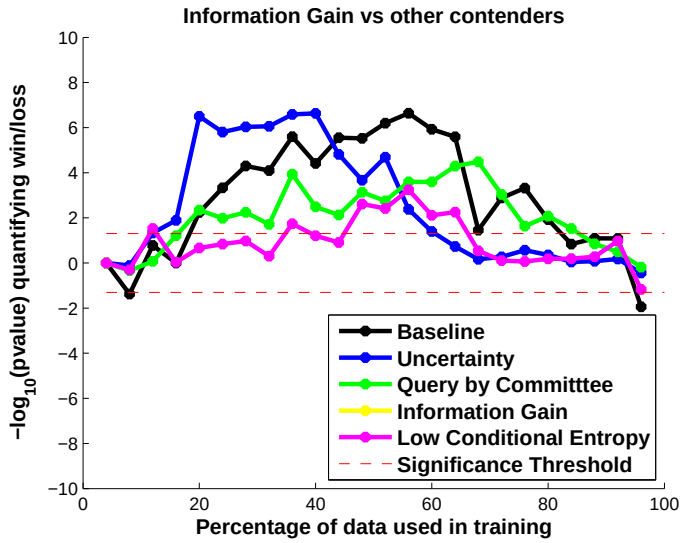


Figure 6.4: ActiveRIPR comparison significance for Information Gain scoring against other contenders. Significant wins/losses are above/below the red dash coresponding to a p-value of 0.05. Artificial data with compact noise.

perform paired t-tests for pair-wise comparison between alternative methods, after each batch of training data. Thus, at each active learning iteration, for any scoring function s_0 and any of its competitors $s_{\bar{0}}$, we obtain a p-value indicating the significance of the win/loss of s_0 over $s_{\bar{0}}$.

For each considered method, we plot the negative decimal logarithm of the p-value in a *win/loss graph*, such as the one shown in Figure 6.4 for the InfoGain scoring method. Each line corresponds to a set of p-values obtained when comparing InfoGain to another contender. The p-value in the case of a win – i.e. when InfoGain outperforms the contender – is placed in the positive interval of the y-axis. On the other hand, if the method loses, the p-value is reversed. The two dashed lines distinguish significant wins/losses from insignificant ones. The top dashed line corresponds to $y = -\log_{10}(0.05)$, whereas the bottom one corresponds to $y = \log_{10}(0.05)$. Thus, anything above the top line is a significant win, anything between the dashed lines is not significant, and anything below the bottom line shows a significant loss. Also, we are mainly interested in significant results in the first and the middle part of the x-axis. In the first few iterations, we do not expect considerable difference between scoring functions since the initial sample is the same. With more iterations, the well performing scoring methods may achieve significant wins. Finally, when all useful data is labeled, all methods begin to converge to the same accuracy, typically with no significant wins/losses. The plot for InfoGain scoring in Figure 6.4, for compact noise artificial data, shows that InfoGain obtains significant wins over all other methods. For conditional entropy-based scoring this only starts after 50% of the data has been labeled. For all other scoring functions, this begins to happen after only 20% of the data has been requested for labeling.

Figure 6.5 displays the win/loss graphs for the other scoring functions. We may conclude that uncertainty scoring loses consistently against other methods until after 60% of the data has been labeled, that query-by-committee has no significant wins and that the baseline is outperformed by both InfoGain and LowCondEntropy. In fact, LowCondEntropy seems the second-best perfomer after InfoGain in terms of significant wins, while being the cheapest to compute.

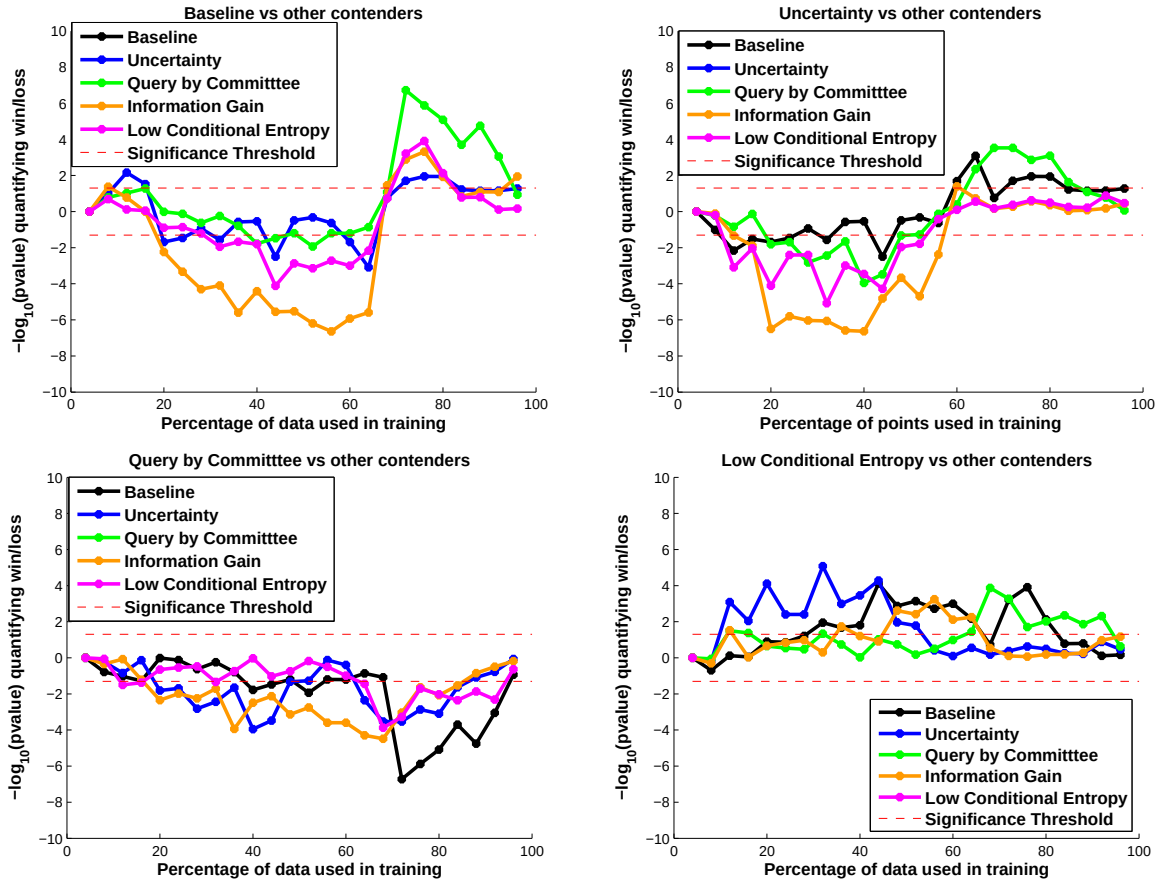


Figure 6.5: ActiveRIPR comparison significance for the baseline (top left), uncertainty (top right), query-by-committee (bottom left) and conditional entropy (bottom right) scoring against their respective contenders. Significant wins/losses are above/below the red dash coresponding to a p-value of 0.05. Artificial data with compact noise.

The *computational efficiency* of InfoGain with ActiveRIPR has a linear (not a high-order) dependency on the InfoGain selection because the training and sample selection are performed sequentially. Moreover, only features selected by RIPR are used by the InfoGain selection, making the procedure *less time-consuming than computing information gain over the full-dimensional space*.

Appendix B

6.4 Summary of SDU data

6.5 Features derived from vital sign time-series

Table 6.4: Summary of step-down unit (SDU) patients, monitoring and annotation outcome of alerts.

	Training and Validation Set
	N=279 patients; 294 admissions
Patients	
Total	279
% male (n %)	180 (58.4%)
Age (mean years $\pm$ SD)	57.42 $\pm$ 20.2
Race (N %)	
White	221 (71.8%)
Black	43(14%)
Other	44(14.3%)
Charlson Deyo Index (mean $\pm$ SD)	0.94 + 1.45
SDU length of stay (mean days $\pm$ SD)	4.70 $\pm$ 4.21
Hospital length of stay (mean days $\pm$ SD	9.17 $\pm$ 9.8
Monitoring	
Total monitoring hours	22,980
Mean hours per patient	82
Median hours per patient	62
Raw VA alerts (Alerts without a continuity or persistence requirement)	
Total	271,288
Total by subtype	
HR	11,952 (4%)
RR	101,589 (38%)
SpO2	61,242 (23%)
Systolic BP	54,679 (20%)
Diastolic BP	41,826 (15%)
Vital sign alert events (VSAE; at least 2 consecutive alerts 40s of each other constitute the same event)	
Total	38,286
Total by subtype	
HR	2,308 (6%)
RR	24,477(63%)
SpO2	11,353(30%)
BP *	148(1%)
Vital sign alert event epochs (VSAE with additional persistence requirement, length $\geq$ 180 s, duty cycle $\geq$ 60%)	
Total	1,582
Total by subtype	
HR	149(9%)
RR	700 (44%)
SpO2	585 (37%)
BP*	148 (10%)
Vital sign alert events with additional persistence requirement annotated by experts	
Total	576
Total by subtype	
Real	418
Artifact	158
Real alerts by subtype	
HR	60
RR	132
SpO2	181
BP	45
Artifact by subtype	
HR	0
RR	25
SpO2	93

Table 6.5: List of feature categories, the aspect of the vital signs signals those features were meant to capture, and the feature names and descriptions.

Feature Category	**Aspect of VS Signals Features Meant to Capture**	**Feature Name**	**Feature Description**
Basic statistics	Basic profile of the alert	Mean	mean
		Sd	standard deviation
		Cv	coefficiency of variance (sd /mean)
		Mad	median of absolute deviation from median [median (abs(x-abs(x))]
		N	number of data points within alert period
		Min	min value
		Max	max value
		Median	median
		Range	max-min
		range_ratio	range/median
Data density	Sparseness of signals	data_den	number of valid data points /length of alert period
		data_den_trail	trailing data density at the beginning of alert, i.e. for 15 min before the start of alert [# data points/(timestamp for last data point - timestamp for first data point)]
		data_den_trail2	trailing data density at the beginning of alert i.e. for 15 min before the start of alert [# data points/(15*60)]
		delta_t	time elapsed since the end of last chunk of data before alert start
		max_gap	max gap between data points (also indicator of data density)
Spectrum features	Frequency and power	spec_ratio	power of high frequency/lower frequency
		max_spec	max power frequency
Change features	Change of VS in current alert period from period immediately before alert	delta_mean	difference of mean (current alert period vs. 4 mins before start of alert)
		delta_sd	difference of sd (current alert period vs. 4 mins before start of alert)
		MW_stat	statistics from Wilcox test to compare distribution within alert period and those from 4 mins before
		MW_pvalue	pvalue from wilcox test to compare distribution within alert period and those from 4 mins before
		KS_stat	statistics from Kolmogorov-Smirnov test to compare distribution within alert period and those from 4 mins before
		KS_pvalue	pvalue from Kolmogorov-Smirnov test to compare distribution within alert period and those from 4 mins before
		t_stat	statistics from t test to compare mean within alert period and those from 4 mins before
		t_pvalue	pvalue from t test to compare mean within alert period and those from 4 mins before
		F_stat	statistics from F test to compare sd within alert period and those from 4 mins before
		F_pvalue	pvalue from F test to compare sd within alert period and those from 4 mins before
Variance features	Signal smoothness	Slope	linear slope during alert period
		Rslope	robust slope during alert period
		slope1	slope for the period before change point
		slope2	slope for the period after change point
		num_breakpoint	number of break points detected
		diff1_max	max first order difference
		diff1_min	min first order difference
		max_grad	max of slope for a series of moving window during alert period
		min_grad	min of slope for a series of moving window during alert period
		quad_rsq	rsquared from a quadratic fitting
		quad_coef1	first order coefficient of quadratic fitting
		quad_coef2	2nd order coefficient of quadratic fitting

Appendix C

6.6 Derivation for the gradient term in (5.8)

$$\begin{aligned}\frac{\partial L(\Theta,\boldsymbol{\pi};\boldsymbol{x},y)}{\partial f_n(\boldsymbol{x};\Theta)} &= \sum_{\ell\in\mathcal{L}} \frac{\partial L(\Theta,\boldsymbol{\pi};\boldsymbol{x},y)}{\partial \mu_\ell(\boldsymbol{x};\Theta)} \frac{\partial \mu_\ell(\boldsymbol{x};\Theta)}{\partial f_n(\boldsymbol{x};\Theta)} \\ &= -\sum_{\ell\in\mathcal{L}} \frac{\pi_{\ell y}}{\mathbb{P}_T[y|\boldsymbol{x},\Theta,\boldsymbol{\pi}]} \frac{\partial \mu_\ell(\boldsymbol{x};\Theta)}{\partial f_n(\boldsymbol{x};\Theta)} \\ &= -\sum_{\ell\in\mathcal{L}} \frac{\pi_{\ell y}\mu_\ell(\boldsymbol{x};\Theta)}{\mathbb{P}_T[y|\boldsymbol{x},\Theta,\boldsymbol{\pi}]} \frac{\partial \log \mu_\ell(\boldsymbol{x};\Theta)}{\partial f_n(\boldsymbol{x};\Theta)}\,,\end{aligned}$$

where

$$\begin{aligned}\frac{\partial \log \mu_\ell(\boldsymbol{x};\Theta)}{\partial f_n(\boldsymbol{x};\Theta)} &= \mathbb{1}_{\ell\swarrow n} \frac{\partial \log d_n(\boldsymbol{x};\Theta)}{\partial f_n(\boldsymbol{x};\Theta)} \\ &\quad + \mathbb{1}_{n\searrow\ell} \frac{\partial \log \bar{d}_n(\boldsymbol{x};\Theta)}{\partial f_n(\boldsymbol{x};\Theta)} \\ &= \mathbb{1}_{\ell\swarrow n}\bar{d}_n(\boldsymbol{x};\Theta) - \mathbb{1}_{n\searrow\ell} d_n(\boldsymbol{x};\Theta)\,.\end{aligned}$$

By substituting the latter in the previous formula we get

$$\begin{aligned}\frac{\partial L(\Theta,\boldsymbol{\pi};\boldsymbol{x},y)}{\partial f_n(\boldsymbol{x};\Theta)} &= -\sum_{\ell\in\mathcal{L}} \mathbb{1}_{\ell\swarrow n} \frac{\pi_{\ell y}\mu_\ell(\boldsymbol{x};\Theta)}{\mathbb{P}_T[y|\boldsymbol{x},\Theta,\boldsymbol{\pi}]} \bar{d}_n(\boldsymbol{x};\Theta) \\ &\quad + \sum_{\ell\in\mathcal{L}} \mathbb{1}_{n\searrow\ell} \frac{\pi_{\ell y}\mu_\ell(\boldsymbol{x};\Theta)}{\mathbb{P}_T[y|\boldsymbol{x},\Theta,\boldsymbol{\pi}]} d_n(\boldsymbol{x};\Theta) \\ &= -\sum_{\ell\in\mathcal{L}_{n_l}} \frac{\pi_{\ell y}\mu_\ell(\boldsymbol{x};\Theta)}{\mathbb{P}_T[y|\boldsymbol{x},\Theta,\boldsymbol{\pi}]} \bar{d}_n(\boldsymbol{x};\Theta) \\ &\quad + \sum_{\ell\in\mathcal{L}_{n_r}} \frac{\pi_{\ell y}\mu_\ell(\boldsymbol{x};\Theta)}{\mathbb{P}_T[y|\boldsymbol{x},\Theta,\boldsymbol{\pi}]} d_n(\boldsymbol{x};\Theta) \\ &= d_n(\boldsymbol{x};\Theta)A_{n_r} - \bar{d}_n(\boldsymbol{x};\Theta)A_{n_l}\,.\end{aligned}$$

6.7 Details of ImageNet experiment

This section provides additional description of the ImageNet dataset [89] and the changes we made to GoogLeNet [102] to obtain our proposed architecture using back-propagation forests (BPF). In addition, we provide more detailed analyses about our model by *e.g.* visualizing statistics about split node decision evolution or error plots, both as a function of training epochs.

6.7.1 ImageNet Dataset

Originally, ImageNet was designed for large-scale image recognition, *i.e.* images were annotated with a single ground truth label and have to be classified into one of 1000 target object categories. The dataset consists of approximately 1.2M training images, 50.000 validation images and 100.000 test images with average dimensionality of 482x415 pixels. Training and validation data is publicly available and we followed the commonly agreed protocol, using validation data for testing and therefore as a proxy for the official (but non-available) test data set. Since categorization to a single ground-truth label might be too restrictive (an image might contain different object categories resulting in label-ambiguity), the *Top5-Error* is used as evaluation metric. Consequently, the classification result for a test image is considered successful if the ground truth label is among 5 possible predictions provided by the classifier. The GoogLeNet architecture we have used as basis for our experiments (see left illustration in Fig. 6.6) has a reported Top5-Error of 10.07%, when used in a single-model, single-crop setting (see first row in Table 3 in [102]). We adhere to this setting, *i.e.* all further presented scores are produced in a single-model, single-crop setup.

6.7.2 The BeefNet Architecture

Figure 6.6 (right illustration) shows that we have introduced two different modifications with respect to the original GoogLeNet architecture. First, we have connected the outputs of the Depth-Concat layers to the inputs of the AveragePool layers (as described in the main submission), visualized by red arrows in the plot. The resulting, modified baseline network is dubbed 'Base2' and achieves a Top5-Error of 10.02% when using conventional SoftMaxLoss layers as in the original network. The implementation yielding this score was realized in the Distributed (Deep) Machine Learning Common (DMLC) library [72, 73][1], using resized images with dimensionality 100x100 as described in [72]. The training used the standard settings for GoogLeNet, stochastic gradient descent with 0.9 momentum, fixed learning rate schedule, decreasing the learning rate by 4% every 8 epochs. We trained 'Base2' with minibatches composed of 50 images.

In order to obtain a *Deep Convolutional Neural Decision Forest* model, we have replaced each SoftMaxLoss-layer from 'Base2' with a forest consisting of 10 trees, resulting in a total number of 30 trees. For our architecture, which we implemented in DMLC as well, we trained the network for 1000 epochs using minibatches composed of 100.000 images. This is feasible due to massive distribution of the computational load to a cluster of 52 CPUs and 12 hosts, where each host is equipped with a NVIDIA Tesla K40 GPU.

We term the resulting BackPropagation forests as BPF_0, BPF_1 and BPF_2, where BPF_0 is closest to the input layer and BPF_2 is the final (last) layer in the architecture. Each tree is a balanced and

[1] https://github.com/dmlc/cxxnet.git

fixed depth 15 tree, which means that the total number of split nodes is $2^{15} - 1 = 32.767$ and the number of leaf nodes is $2^{15} = 32.768$.

The way we have actually realized the individual forests is illustrated in Fig. **??** and described next: We use a standard fully connected (or inner-product) layer FC to model functions $f_n(\cdot; \Theta)$ described in Equation (3) of the main submission. For BeefNet, each f_n receives as input 500 randomly selected output dimensions of the respectively preceding layers in Base2 (or GoogLeNet). In such a way, a single FC layer can be defined that provides all the split node inputs for one back-propagation forest (a total number of #trees $\times$ #split nodes/tree). Next, each of the resulting split function inputs is passed through a sigmoid layer such that $d_n(\boldsymbol{x}) = \sigma(f_n(\boldsymbol{x}))$. Using only a randomly selected subset for generating the split node inputs f_n reduces computational load and helps to improve decorrelation of trees. Another strategy aiming for tree decorrelation is related to bagging and defines how we actually update the individual tree parameters: During training, each minibatch is only forwarded to a single tree per forest. We keep a simple strategy and update trees in a sequential fashion, using the randomly compiled minibatches. In such a way, each tree will only be updated based on a subset of all samples and we only have to compute the gradients for a single tree at a time, which helps to reduce computational load for both, forward and backward passes.

At this place we would like to correct the description of the main paper and clarify that dropout was integrated between the two fully connected layers preceding BPF_0 and BPF_1 and also before the fully connected layer of BPF_2. Dropout was also present in the two baselines at the same levels of the network. As a final remark on Fig. **??** we want to mention that the assignment of the FC output units to the split nodes can be chosen arbitrarily since the actual hierarchy of the trees is constructed when the $\mu_\ell(\boldsymbol{x}|\cdot)$ quantities (Equation (2) of main submission) are computed.

Regarding the posterior learning, we only update the leaf node predictions for the tree where we also compute the gradients for the split node updates, *i.e.* the tree which was selected for the minibatch feedforward-pass. In order to improve computational efficiency, we considered only the samples of the current minibatch for posterior learning, while *all* the training data could be used in principle. However, since each of our minibatches is composed of 100.000 samples, we can approximate the training set sufficiently well while simultaneously introducing a positive, regularizing effect.

6.7.3 Evaluation of split nodes

Next, we provide insights on some of the operational characteristics of our model. Intuitively, a tree-structured classifier follows the divide-and-conquer principle and aims to produce 'pure' leaf node distributions. This means that our training task is to partition the input space such that we improve on the predictions about the training samples, which happens when the input space is split in a way that it correlates with classes.

We can partially monitor this behavior by analyzing the split node responses (their outputs), which inform us about the routing uncertainties for a given sample $\boldsymbol{x}$ as it traverses the tree(s). In Figure 6.7 we show histograms of all available split node outputs of our three forests for all samples of the validation set after running for 100, 500 and 1000 epochs over the training data. The leftmost histogram (after 100 training epochs) shows the highest uncertainty about the routing direction, *i.e.* the split decisions are not yet very crisp such that a sample will be routed to many leaf nodes. As training progresses, (middle and right plots after 500 and 1000 epochs) we can see

how the distributions get very peaked at 0 and 1 (*i.e.* samples are routed either to the left or right child with low uncertainty), respectively. As a result, the samples will only be routed to a small subset of available leaf nodes with reasonably high probability. In other words, most available leaves will never be reached from a sample-centric view and therefore only a small number of overall paths needs to be evaluated at test time.

6.7.4 Evaluation of model performance

In Figure 6.8 we show the development of Top5-Errors for each backpropagation-forest in BeefNet as well as their ensemble performance. In particular, we show how the individual errors decrease as a function of the training epochs. The left plot shows the performance over the entire number of training epochs (1000) while the right plot shows the error development in a zoomed window from epochs 500 to 1000 and Top5-Errors 0–12%. As expected, BPF_0 (which is closest to the input layer) performs worse than BPF_2, which constitutes the final layer of BeefNet, however, only by 1.34%. This means, all the computational load between BPF_0 and BPF_2 can in principle be traded for a degradation of only 1.34% in accuracy. Conversely, using all three back-propagation forests as an ensemble yields the lowest Top5-Error of 7.84% after 1000 epochs over training data. Please note that we observe the lowest Top5-Error on validation data with 7.32% after 953 epochs, however, to ensure a fair comparison to the baseline networks we evaluated them in a way such that they have seen the same amount of training data.

6.7.5 Evaluations of leaf nodes

Finally, we show three randomly chosen leaf node distributions in Figure 6.9 as obtained from the global posterior learning process. As can be seen, the histograms are quite sparsely populated and therefore show clear preferences for certain object categories.

To get a better idea of the quality of the resulting leaf posterior distributions obtained from the global optimization procedure, we provide a plot how the mean leaf entropy develops as a function of training epochs (Fig. 6.10). To this end, we randomly selected 1024 leaves from all available ones per tree and computed their mean entropy after each epoch. The highest entropy would result from a uniform distribution and is approximately 9.96 bits for a 1000-class problem. Instead, we want to obtain highly peaked distributions for the leaf predictors, leading to low entropy values. The plot in Fig. 6.10 illustrates how the mean entropy gracefully degrades as the number of training epochs increases and therefore confirms the efficacy of our proposed leaf node parameter learning approach.

6.8 Proof of update rule for $\boldsymbol{\pi}$

Theorem 6.8.1. *Consider a tree with parameters Θ and $\boldsymbol{\pi}$ and let*

$$\hat{\pi}_{\ell y} = \frac{1}{Z_\ell} \sum_{(\boldsymbol{x},y')\in\mathcal{T}} \mathbb{1}_{y=y'} \frac{\pi_{\ell y}\mu_\ell(\boldsymbol{x};\Theta)}{\mathbb{P}_T[y|\boldsymbol{x};\Theta,\boldsymbol{\pi}]}\,, \qquad \textit{for all } (\ell,y)\in\mathcal{L}\times\mathcal{Y}\,, \tag{6.1}$$

where Z_ℓ is the normalizing factor ensuring that $\hat{\boldsymbol{\pi}}_\ell = (\hat{\pi}_{\ell y})_{y\in\mathcal{Y}}$ is a probability distribution. In other terms, we assume $\hat{\pi}_{\ell y}$ to be the result of an update step as per (10) of our ICCV contribution.

The following holds:

$$R(\Theta, \boldsymbol{\pi}; \mathcal{T}) \geq R(\Theta, \hat{\boldsymbol{\pi}}; \mathcal{T})$$

with equality if and only if $\hat{\boldsymbol{\pi}} = \boldsymbol{\pi}$, where R is the risk defined in (5) of our ICCV contribution, and $\boldsymbol{\pi} = (\boldsymbol{\pi}_\ell)_{\ell \in \mathcal{L}}$.

Proof. Consider the following auxiliary function:

$$\phi(\boldsymbol{\pi}, \overline{\boldsymbol{\pi}}) = R(\Theta, \overline{\boldsymbol{\pi}}; \mathcal{T}) - \frac{1}{|\mathcal{T}|} \sum_{(\boldsymbol{x}, y) \in \mathcal{T}} \sum_{\ell \in \mathcal{L}} \xi_\ell(\overline{\boldsymbol{\pi}}; \boldsymbol{x}, y) \log \left(\frac{\pi_{\ell y}}{\overline{\pi}_{\ell y}} \right) ,$$

where

$$\xi_\ell(\boldsymbol{\pi}; \boldsymbol{x}, y) = \frac{\pi_{\ell y} \mu_\ell(\boldsymbol{x}; \Theta)}{\mathbb{P}_T[y | \boldsymbol{x}; \Theta, \boldsymbol{\pi}]} .$$

and $\mathbb{P}_T[y | \boldsymbol{x}, \Theta, \boldsymbol{\pi}]$ is defined as per (1) of our ICCV contribution. Note that $\phi(\boldsymbol{\pi}, \boldsymbol{\pi}) = R(\Theta, \boldsymbol{\pi}; \mathcal{T})$ holds for any $\boldsymbol{\pi}$, for the logarithm term in ϕ nullifies. Moreover, $\phi(\boldsymbol{\pi}, \overline{\boldsymbol{\pi}}) \geq R(\Theta, \boldsymbol{\pi}; \mathcal{T})$ holds for any $\boldsymbol{\pi}$ and $\overline{\boldsymbol{\pi}}$. This can be seen by applying Jensen's inequality and with few algebraic manipulations:

$$\begin{aligned}
R(\Theta, \boldsymbol{\pi}; \mathcal{T}) &= -\frac{1}{|\mathcal{T}|} \sum_{(\boldsymbol{x}, y) \in \mathcal{T}} \log \left(\sum_{\ell \in \mathcal{L}} \pi_{\ell y} \mu_\ell(\boldsymbol{x}; \Theta) \right) \\
&\leq -\frac{1}{|\mathcal{T}|} \sum_{(\boldsymbol{x}, y) \in \mathcal{T}} \sum_{\ell \in \mathcal{L}} \xi_\ell(\overline{\boldsymbol{\pi}}; \boldsymbol{x}, y) \log \left(\frac{\pi_{\ell y} \mu_\ell(\boldsymbol{x}; \Theta)}{\xi_\ell(\overline{\boldsymbol{\pi}}; \boldsymbol{x}, y)} \right) \qquad \text{(by Jensen's inequality)} \\
&= -\frac{1}{|\mathcal{T}|} \sum_{(\boldsymbol{x}, y) \in \mathcal{T}} \sum_{\ell \in \mathcal{L}} \xi_\ell(\overline{\boldsymbol{\pi}}; \boldsymbol{x}, y) \left[\log \left(\frac{\pi_{\ell y}}{\overline{\pi}_{\ell y}} \right) + \log \mathbb{P}_T[y | \boldsymbol{x}, \Theta, \overline{\boldsymbol{\pi}}] \right] \\
&= R(\Theta, \overline{\boldsymbol{\pi}}; \mathcal{T}) - \frac{1}{|\mathcal{T}|} \sum_{(\boldsymbol{x}, y) \in \mathcal{T}} \sum_{\ell \in \mathcal{L}} \xi_\ell(\overline{\boldsymbol{\pi}}; \boldsymbol{x}, y) \log \left(\frac{\pi_{\ell y}}{\overline{\pi}_{\ell y}} \right) = \phi(\boldsymbol{\pi}, \overline{\boldsymbol{\pi}}) .
\end{aligned}$$

We can now show that $\hat{\boldsymbol{\pi}}$ is a global minimizer of $\phi(\cdot, \boldsymbol{\pi})$ for any value of $\boldsymbol{\pi}$. We start rewriting $\hat{\boldsymbol{\pi}}$ in terms of ξ_ℓ as follows:

$$\hat{\pi}_{\ell y} = \frac{1}{Z_\ell} \sum_{(\boldsymbol{x}, y') \in \mathcal{T}} \mathbb{1}_{y = y'} \xi_\ell(\boldsymbol{\pi}; \boldsymbol{x}, y) ,$$

where Z_ℓ is the normalizing factor. Then, we have that

$$\begin{aligned}
&\phi(\hat{\boldsymbol{\pi}}, \boldsymbol{\pi}) - \phi(\overline{\boldsymbol{\pi}}, \boldsymbol{\pi}) \\
&\quad = -\frac{1}{|\mathcal{T}|} \sum_{(\boldsymbol{x}, y) \in \mathcal{T}} \sum_{\ell \in \mathcal{L}} \xi_\ell(\boldsymbol{\pi}; \boldsymbol{x}, y) \log \left(\frac{\hat{\pi}_{\ell y}}{\overline{\pi}_{\ell y}} \right) \\
&\quad = -\frac{1}{|\mathcal{T}|} \sum_{\ell \in \mathcal{L}} \sum_{y \in \mathcal{Y}} \sum_{(\boldsymbol{x}, y') \in \mathcal{T}} \mathbb{1}_{y = y'} \xi_\ell(\boldsymbol{\pi}; \boldsymbol{x}, y) \log \left(\frac{\hat{\pi}_{\ell y}}{\overline{\pi}_{\ell y}} \right) \\
&\quad = -\frac{1}{|\mathcal{T}|} \sum_{\ell \in \mathcal{L}} \sum_{y \in \mathcal{Y}} Z_\ell \hat{\pi}_{\ell y} \log \left(\frac{\hat{\pi}_{\ell y}}{\overline{\pi}_{\ell y}} \right) \\
&\quad = -\frac{1}{|\mathcal{T}|} \sum_{\ell \in \mathcal{L}} Z_\ell D_{KL}(\hat{\boldsymbol{\pi}}_\ell \| \overline{\boldsymbol{\pi}}_\ell) \leq 0 ,
\end{aligned}$$

holds for all values of $\overline{\boldsymbol{\pi}}$, where $D_{KL}(\cdot\|\cdot)$ is the Kullback-Leibler divergence. Note that the last inequality yields equality if and only if $\hat{\boldsymbol{\pi}} = \overline{\boldsymbol{\pi}}$, for the Kullback-Leibler divergence yields zero if and only if the two distributions in input coincide. Accordingly, $\hat{\boldsymbol{\pi}}$ is a *strict* global minimizer of $\phi(\cdot, \boldsymbol{\pi})$ for any $\boldsymbol{\pi}$.

As a consequence of the previous derivations we have

$$R(\Theta, \boldsymbol{\pi}; \mathcal{T}) = \phi(\boldsymbol{\pi}, \boldsymbol{\pi}) > \phi(\hat{\boldsymbol{\pi}}, \boldsymbol{\pi}) \geq R(\Theta, \hat{\boldsymbol{\pi}}; \mathcal{T}) \,,$$

where equality holds if and only if we have a fixed point of the update rule (6.1), *i.e.* if $\hat{\boldsymbol{\pi}} = \boldsymbol{\pi}$. $\square$

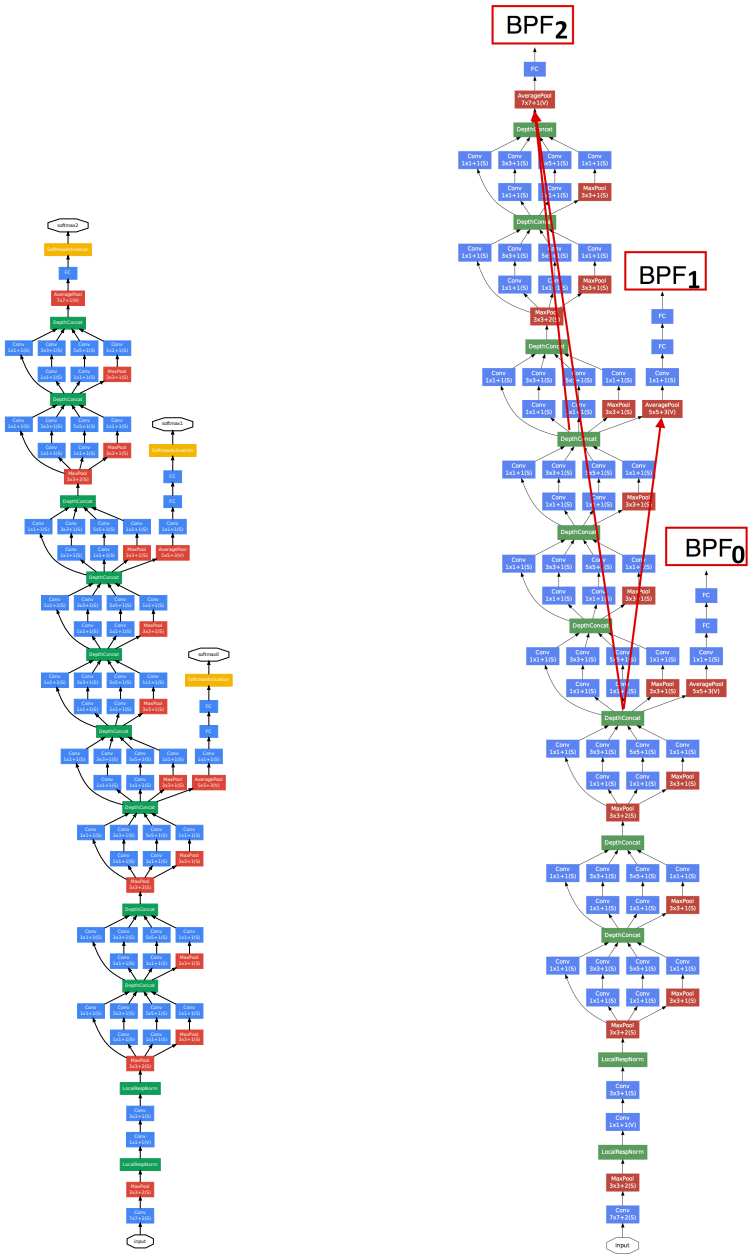


Figure 6.6: Left: Original GoogLeNet architecture proposed in [102]. Right: The modifications we brought to the GoogLeNet architecture resulting in *BeefNet* – our proposed model using BPFs as final classifiers. Best viewed with digital zoom.

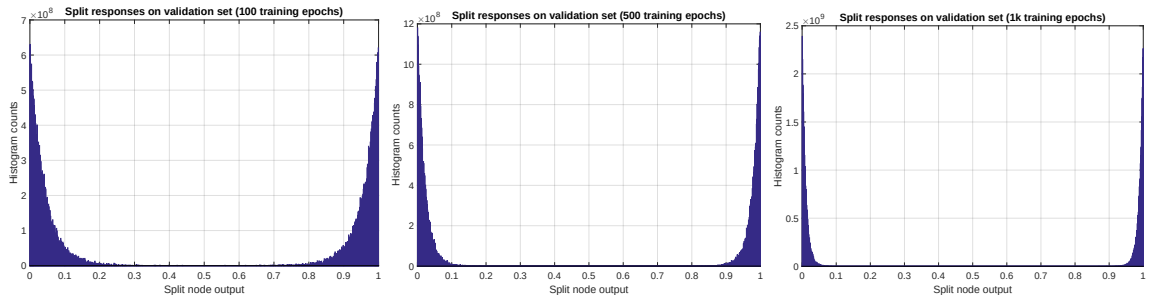


Figure 6.7: Histograms over all split node responses of all three forests in BeefNet on validation data after accomplishing 100 (left), 500 (middle) and 1000 (right) epochs over training data. As training progresses, the split node outputs approach 0 or 1 which corresponds to eliminating routing uncertainty of samples when being propagated through the trees.

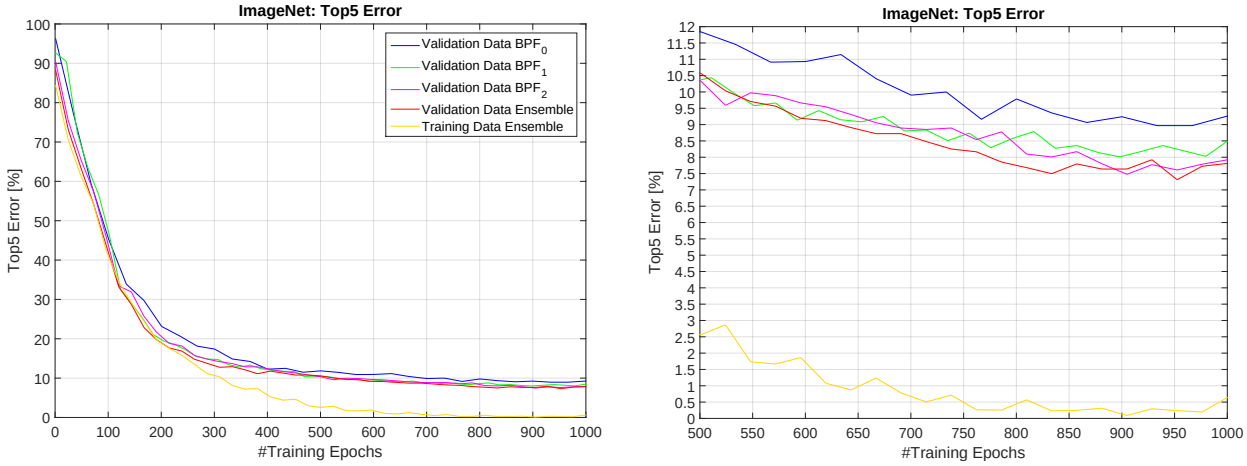


Figure 6.8: Top5-Error plots for individual BPFs used in BeefNet as well as their ensemble errors. Left: Plot over all 1000 training epochs. Right: Zoomed version of left plot, showing Top5-Errors from 0–12% between training epochs 500-1000

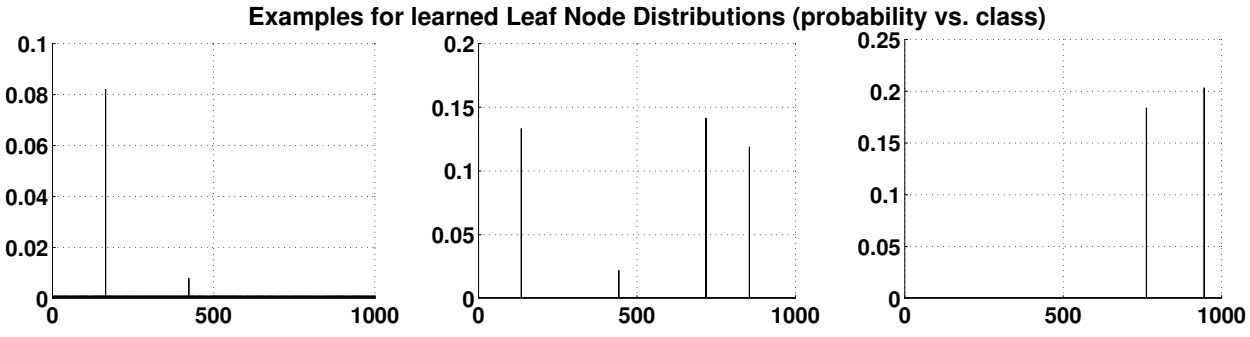


Figure 6.9: Exemplary leaf node distributions that are obtained by solving the convex optimization problem defined in Equation (9) of the main submission.

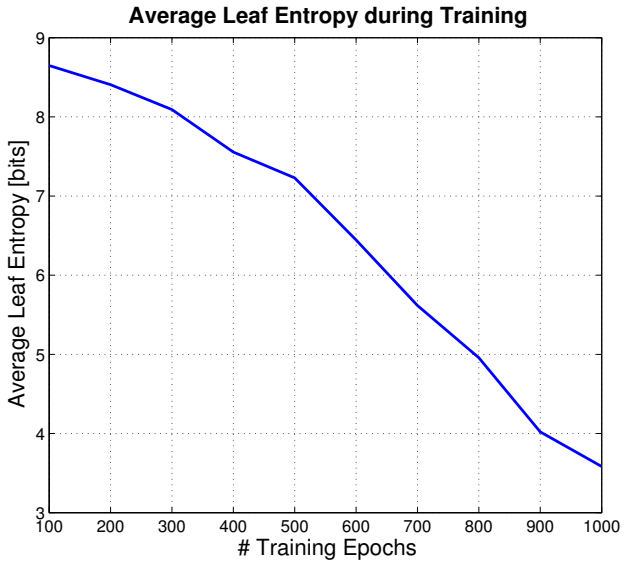


Figure 6.10: Average leaf entropy development as a function of training epochs.