

Boston University

OpenBU

<http://open.bu.edu>

Theses & Dissertations

Boston University Theses & Dissertations

2018

Learning to predict under a budget

<https://hdl.handle.net/2144/30726>

Boston University

BOSTON UNIVERSITY
COLLEGE OF ENGINEERING

Dissertation

LEARNING TO PREDICT UNDER A BUDGET

by

FENG NAN

B.S., National University of Singapore, 2008
M.S., Massachusetts Institute of Technology, 2009

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

2018

© 2018 by
FENG NAN
All rights reserved

Approved by

First Reader

Venkatesh Saligrama, PhD
Professor of Electrical and Computer Engineering
Professor of Systems Engineering
Professor of Computer Science

Second Reader

David A. Castañón, PhD
Professor of Electrical and Computer Engineering
Professor of Systems Engineering

Third Reader

Lorenzo Orecchia, PhD
Assistant Professor of Computer Science

Fourth Reader

Alexander Olshevsky, PhD
Assistant Professor of Electrical and Computer Engineering
Assistant Professor of Systems Engineering

Acknowledgments

I have been fortunate to receive much support for the work of this thesis.

First, I am happy to give thanks to God and the Lord Jesus Christ for His love. He opened the door for me to pursue doctorate study and has carried me through.

I would also like to thank my advisor Professor Venkatesh Saligrama for his guidance and encouragement. The breath and depth of his intellect will continue to be my aspiration. In addition, I have learned what doing good research is about from seeing him at work. It is about being optimistic and enjoying the process even when the goal seems far; it is also about never giving up an idea without knowing why it does not work.

I also had the privilege of working with Professor Yannis Paschalidis, who kindly mentored me for the first year at BU and introduced me to topics in computational biology. I would also like to acknowledge Professor David Castañón and Professor Lorenzo Orecchia for their availability to share their insights with me on optimization. I also thank Professor Alexander Olshevsky for agreeing to be on my thesis committee.

I must also thank my colleagues and friends in BU for their numerous help and encouragement. They made my doctorate study a lot more enjoyable.

Last but not least, I would like to thank my family, especially my wife Yanping. She gave up much to support me through my study and gave birth to our two beautiful girls. Her sacrifice and support are indispensable to my success. I also thank our parents for their selfless love and unconditioned support.

LEARNING TO PREDICT UNDER A BUDGET

FENG NAN

Boston University, College of Engineering, 2018

Major Professor: Venkatesh Saligrama, PhD

Professor of Electrical and Computer Engineering

Professor of Systems Engineering

Professor of Computer Science

ABSTRACT

Prediction-time budgets in machine learning applications can arise due to monetary or computational costs associated with acquiring information; they also arise due to latency and power consumption costs in evaluating increasingly more complex models. The goal in such budgeted prediction problems is to learn decision systems that maintain high prediction accuracy while meeting average cost constraints during prediction-time.

In this thesis, I will present several learning methods to better trade-off cost and error during prediction. The conceptual contribution of this thesis is to develop a new paradigm of bottom-up approaches instead of the traditional top-down approaches. A top-down approach attempts to build out the model by selectively adding the most cost-effective features to improve accuracy. It leads to fundamental combinatorial issues in multi-stage search over all feature subsets. In contrast, a bottom-up approach first learns a highly accurate model and then prunes or adaptively approximates it to trade-off cost and error. We show that the bottom-up approach has several benefits.

To develop this theme, we first propose two top-down methods and then two bottom-up methods. The first top-down method uses margin information from train-

ing data in the partial feature neighborhood of a test point to either select the next best feature in a greedy fashion or to stop and make prediction. The second top-down method is a variant of random forest (RF) algorithm. We grow decision trees with low acquisition cost and high strength based on greedy minimax cost-weighted impurity splits. Theoretically, we establish near-optimal acquisition cost guarantees for our algorithm.

The first bottom-up method we propose is based on pruning RFs to optimize expected feature cost and accuracy. Given a RF as input, we pose pruning as a novel 0-1 integer program and show that it can be solved exactly via LP relaxation. We further develop a fast primal-dual algorithm that scales to large datasets. The second bottom-up method is adaptive approximation, which significantly generalizes the RF pruning to accommodate more models and other types of costs besides feature acquisition cost. We first train a high-accuracy, high-cost model. We then jointly learn a low-cost gating function together with a low-cost prediction model to adaptively approximate the high-cost model. The gating function identifies the regions of the input space where the low-cost model suffices for making highly accurate predictions.

We demonstrate empirical performance of these methods and compare them to the state-of-the-arts. Finally, we study adaptive approximation in the on-line setting to obtain regret guarantees and discuss future work.

Contents

1	Introduction	1
1.1	Resource-constrained Machine Learning: Motivation	1
1.2	Problem Definition	3
1.2.1	Feature Acquisition Cost	4
1.2.2	Computational Cost	5
1.2.3	Communication/Latency Cost	5
1.3	Challenges	5
1.4	Contribution	7
1.5	Related Work	8
1.5.1	Non-adaptive methods	9
1.5.2	Fixed feature acquisition	9
1.5.3	Myopic feature acquisition	10
1.5.4	Non-myopic feature acquisition	11
1.5.5	Other methods	12
1.6	Organization	13
2	Margin-based Nearest Neighbor Approach	15
2.1	Related Work	16
2.2	Problem Setup	16
2.3	Algorithm	19
2.4	Experiments	21

3	Feature-Budgeted Random Forest	25
3.1	Related Work	26
3.2	Problem Setup	27
3.3	Algorithm	28
3.4	Bounding the Cost of Each Tree	30
	3.4.1 Admissible Impurity Functions	35
	3.4.2 Discussions	38
3.5	Experiments	40
4	Pruning Random Forests	49
4.1	Related Work	51
4.2	Problem Setup	52
	4.2.1 Pruning with Costs	53
4.3	Theoretical Analysis	56
	4.3.1 A Naive Pruning Formulation	59
4.4	Algorithm	61
4.5	Experiments	65
	4.5.1 Baseline Comparison	65
	4.5.2 Additional Experiments	70
	4.5.3 Discussion and Conclusion	73
5	Adaptive Approximation	75
5.1	Related Work	78
5.2	Problem Setup	79
5.3	Algorithms	84
5.4	Experiments	87
6	On-line Adaptive Approximation	95
6.1	Problem Setup	95

6.2	Upper Bound	97
6.3	Lower Bound	100
7	Future Work	103
7.1	Distributed Prediction	103
7.1.1	Adaptive Sparse Regression	104
7.1.2	Local Convexity - Hessian Computation	105
7.1.3	Optimization	106
7.1.4	Theorems to be proved	107
7.1.5	Algorithms	108
7.1.6	Experiments	109
7.2	Extending the Regret Lower Bound	112
7.2.1	Non-uniform sampling of patterns	114
8	Conclusions	120
A	Appendix	122
A.1	ADAPT-LSTSQ for Chapter 5	122
A.2	Experimental Details for Chapter 5	124
A.2.1	Synthetic-1 Experiment	124
A.2.2	Synthetic-2 Experiment:	124
A.2.3	Letters Dataset (Dheeru and Karra Taniskidou, 2017)	124
A.2.4	MiniBooNE Particle Identification and Forest Covertype Datasets (Dheeru and Karra Taniskidou, 2017):	126
A.2.5	Yahoo! Learning to Rank(Chapelle et al., 2011):	127
A.2.6	CIFAR10 (Krizhevsky, 2009):	128
	References	129
	Curriculum Vitae	135

List of Tables

2.1	The actual and estimated probabilities of correct classification for neighborhood sizes 65, 125 and 185.	22
4.1	Typical feature usage in a 40 tree RF before and after pruning (our algorithm) on the MiniBooNE dataset. Columns 2-4 list percentage of test examples that do not use the feature, use it 1 to 7 times, and use it greater than 7 times, respectively. Before pruning, 91% examples use the feature only a few (1 to 7) times, paying a significant cost for its acquisition; after pruning, 68% of the total examples no longer use this feature, reducing cost with minimal error increase. Column 5 is the average feature cost (the average number of unique features used by test examples). Column 6 is the test error of RFs. Overall, pruning dramatically reduces average feature cost while maintaining the same error level.	50
5.1	Dataset Statistics	90

List of Figures

2·1	An example of cost sensitive learning. Given 8 training points, each is binary with 3 features: $x^{(1)} = x^{(2)} = (1, -1, -1), x^{(3)} = x^{(4)} = (-1, 1, 1), x^{(5)} = (-1, 1, -1), x^{(6)} = x^{(7)} = x^{(8)} = (1, 1, 1)$, with labels $y^{(1)}, \dots, y^{(4)} = -1, y^{(5)}, \dots, y^{(8)} = 1$. They are linearly separable with optimal SVM solution $y = w' * x + b = (0.9995, 1.4998, -0.5002)x - 0.9997$	20
2·2	Experiment result of classification accuracy vs number of features measured on Letters, LandSat MiniBooNE datasets. FMCC is consistent across all datasets while the VoC does not perform well on the Mini-BooNE dataset.	24
3·1	A synthetic example to show max-cost of GREEDYTREE can be “smoothed” to approach the expected-cost. The left and right figures above show the classifier outcomes of feature t_1 and t_2 , respectively.	39
3·2	The error-cost trade-off plot of the subroutine GREEDYTREE using threshold-Pairs on the synthetic example. 0.39% error can be achieved using only a depth-2 tree but it takes a depth-10 tree to achieve zero error.	40
3·3	Comparison of BUDGETRF against ASTC (Kusner et al., 2014) and CSTC (Xu et al., 2013) on 4 real world datasets. BUDGETRF has a clear advantage over these state-of-the-art methods as it achieves high accuracy/low error using less feature costs.	42

3.4	Comparison of classification error vs. max-cost for the Powers impurity function in (3.13) for $l = 2, 3, 4, 5$ and the threshold-Pairs impurity function. Note that for both House Votes and WBCD, the depth 0 tree is not included as the error decreases dramatically using a single test. In many cases, the threshold-Pairs impurity function outperforms the Powers impurity functions for trees with smaller max-costs, whereas the Powers impurity function outperforms the threshold-Pairs function for larger max-costs.	48
4.1	An ensemble of two decision trees with node numbers and associated feature in subscripts	58
4.2	Turning pruning to equivalent shortest path problems.	64
4.3	Comparison of BUDGETPRUNE against CCP, BUDGETRF with early stopping, GREEDYPRUNE and GREEDYMISER on 4 real world datasets. BUDGETPRUNE (red) outperforms competing state-of-art methods. GREEDYMISER dominates ASTC (Kusner et al., 2014), CSTC (Xu et al., 2013) and DAG (Wang et al., 2015) significantly on all datasets. We omit them in the plots to clearly depict the differences between competing methods.	66
4.4	Comparing BUDGETPRUNE and CCP with uniform and non-uniform feature cost on MiniBooNE dataset. BUDGETPRUNE is robust when the feature cost is non-uniform.	71
4.5	Comparisons of various pruning methods based on entropy and Pairs splitting criteria on MiniBooNE and Forest datasets	72

4.6	Comparing various pruning approaches on RF built with $k=20$ and $k=120$ on Scene15 dataset. The initial RF has higher accuracy and higher cost for $k=20$. GREEDYPRUNE performs very well in $k=20$ but very poorly in $k=120$	72
5.1	Left: single stage schematic of our approach. We learn low-cost gating g and a LPC model to adaptively approximate a HPC model. Right: Key insight for adaptive approximation. x-axis represents feature space; y-axis represents conditional probability of correct prediction; LPC can match HPC's prediction in the input region corresponding to the right of the gating threshold but performs poorly otherwise. Our goal is to learn a low-cost gating function that attempts to send examples on the right to LPC and the left to HPC.	76
5.2	Synthetic-1 experiment without feature cost. (a): input data. (d): decision contour of RBF-SVM as f_0 . (b) and (c): decision boundaries of linear g and f_1 at initialization and after 10 iterations of ADAPT-LIN. (e) and (f): decision boundaries of boosted tree g and f_1 at initialization and after 10 iterations of ADAPT-GBRT. Examples in the beige areas are sent to f_0 by the g	88
5.3	A 2-D synthetic example for adaptive feature acquisition. On the left: data distributed in four clusters. The two features correspond to x and y coordinates, respectively. On the right: accuracy-cost trade-off curves. Our algorithm can recover the optimal adaptive system whereas a L1-based approach cannot.	89

5.4	Comparison of ADAPT-GBRT against GREEDYMISER and BUDGET-PRUNE on four benchmark datasets. RF is used as f_0 for ADAPT-GBRT in (a-c) while an RBF-SVM is used as f_0 in (d). ADAPT-GBRT achieves better accuracy-cost tradeoff than other methods. The gap is significant in (b) (c) and (d). Note the accuracy of GREEDYMISER in (b) never exceeds 0.86 and its precision in (c) slowly rises to 0.138 at cost of 658. We limit the cost range for a clearer comparison.	91
5.5	Compare the L1 baseline approach, ADAPT-LIN and ADAPT-GBRT based on RBF-SVM and RF as f_0 's on the Letters dataset.	93
6.1	Feedback graph for four experts. ξ_1 and ξ_2 request for label and receives full feedback; ξ_3 and ξ_4 classify using h and receives no feedback. . . .	96
7.1	A distributed mixture of expert model for	104
7.2	Sigmoid parameter is the constant multiplier inside the exponential of the sigmoid function; noise is the noise level; ss is the step size	119

List of Abbreviations

DAG	Directed Acyclic Graph
GBRT	Gradient Boosted Regression Tree
RF	Random Forest
\Re^K	the K-dimensional Euclidean space

Chapter 1

Introduction

1.1 Resource-constrained Machine Learning: Motivation

Machine learning plays an increasingly important role in many scientific and engineering problems. It includes problems such as classification, regression, ranking, clustering and so on. Much of machine learning research has focused on improving accuracy. But more recently, as the scale and complexity of machine learning applications grow, costs in both training and test time have gained importance. To limit scope, we consider exclusively supervised learning in this thesis. Training time thus typically involves the cost of collecting labeled data and the computational cost of processing the collected data to learn the model. In many applications such as health care, labeled data is scarce and expensive. The area of active learning (Settles, 2009) is devoted to efficiently using fewer labeled examples to train models. Once a model is trained, it is used for prediction of new examples. Prediction-time costs can arise due to monetary costs associated with acquiring information or computation time (or delay) involved in extracting features and running the algorithm; they can also arise in mobile computing due to limited memory, battery and communication.

In many machine learning applications training can be carried out off-line, separate from the production system. On the other hand, prediction typically occurs in production and is subject to more stringent budget constraints. Therefore, this thesis focuses primarily on reducing costs incurred during prediction or test time. Only toward the end (Chapter 6) we will discuss an on-line learning scenario where

we bring together training and test time costs. Consider the following applications as motivation for prediction time budget constraints.

- **Automated medical diagnosis:** This is a classification task. During training, an algorithm is given medical records of diagnosed patients as input features and the diagnosis as labels. The goal is to learn a model to automatically diagnose new patients based on the outcome of their medical test results. Some of these medical tests are simple and inexpensive such as blood pressures, vitals. Others are more expensive and could potentially be harmful to the human body such as X-ray, MRI. The prediction time cost consists of the monetary cost of each medical test as well as its associated risk. When a new patient is presented to the system, it is thus undesirable to require him or her to undertake all possible medical tests and then make a prediction. Instead, we aim to learn a system that recommends only the necessary medical tests to reduce cost while maintaining high diagnosis accuracy.
- **Document ranking:** (Chapelle et al., 2011) This is a ranking task. During training, an algorithm is given a set of queries as well as a set of documents associated with each query ranked according to the relevance to the query. The goal is to learn a model so that given a new query and a set of documents, it can rank the documents according to the relevance to the query. To achieve this, features of each query-document pair must be extracted. Some features are cheap to extract, such as key word search; other features are computationally more expensive, such as textual similarity and proximity measures. Each of these features require CPU time to extract, yet the ranking has to be done in milliseconds to be displayed to the user. This precludes extraction of computationally expensive features for all query-document pairs. We aim to learn a system that extracts the expensive features only if it is necessary to reduce

cost while maintaining high ranking accuracy.

- **Deep neural networks (DNNs):** DNNs have been successfully applied in many application including visual object recognition, speech recognition and machine translation. They achieve the state of the art accuracy yet require considerable computational budget during prediction due to their increasing complexity. For example, the Resnet152 (He et al., 2016) architecture with 152 layers has 4.4% accuracy gain in top-5 performance over GoogLeNet (Szegedy et al.,) on the large-scale ImageNet dataset (Russakovsky et al., 2015) but is 14X slower at test-time(Bolukbasi et al., 2017). We aim to learn systems that can reduce the computational cost while maintaining high accuracy.
- **Mobile computing, Internet of Things (IoT):** Smart devices include phones, watches, cameras and sensors (known as edge devices) have been widely used to gather and process information for tasks such as activity recognition and surveillance. Such devices typically have limited battery, memory and computational power. Machine learning models that run on such devices are constrained by these physical limitations. For real-time applications, there is also a communication cost in terms of latency whenever the edge devices communicate with the server(cloud). We aim to develop machine learning systems that are suitable to be deployed on such edge devices and use small budget to achieve high accuracy.

1.2 Problem Definition

In this section, we introduce some basic notations and present the general problem of learning with prediction-time costs similar to the formulation in (Trapeznikov and Saligrama, 2013; Wang et al., 2014b). We focus on the supervised setting where we assume fully annotated datasets are available for training. We seek to learn decision systems that maintain high-accuracy while meeting average resource constraints

during prediction-time.

Suppose an example-label pair (x, y) is drawn from distribution \mathcal{P} . The goal is to learn a prediction function f from a family of functions \mathcal{F} that minimizes expected prediction error subject to a budget constraint:

$$\min_{f \in \mathcal{F}} \mathbb{E}_{(x,y) \sim \mathcal{P}} [\text{err}(y, f(x))], \text{ s.t. } \mathbb{E}_{x \sim \mathcal{P}_x} [C(f, x)] \leq B, \quad (1.1)$$

where $\text{err}(y, \hat{y})$ is the error function; $C(f, x)$ is the cost of evaluating the function f on example x and B is a user specified budget constraint.

In practice, we are not given the distribution but instead are given a set of training data $(x^{(1)}, y^{(1)}), \dots, (x^{(n)}, y^{(n)})$ drawn i.i.d. from distribution \mathcal{P} . We can then minimize an empirical approximation of the expected error function:

$$\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n L(y^{(i)}, f(x^{(i)})), \text{ s.t. } \frac{1}{n} \sum_{i=1}^n C(f, x^{(i)}) \leq B, \quad (1.2)$$

where $L(y, \hat{y})$ is a loss function. Note our budget constraint is on prediction costs averaged over the examples. This allows the flexibility to spend the budget in an example-dependent manner.

The definition of $C(f, x)$ is application specific as seen in the motivation examples in Section 1.1. We shall focus on the *feature acquisition cost* in this thesis while addressing other types of costs such as *computational* and *communication/latency* costs as well.

1.2.1 Feature Acquisition Cost

Features (or covariates in statistics) are the numerical attributes associated with an input example. They provide information about the examples as a basis for prediction. There is often a cost associated with acquiring or extracting these feature values. Suppose $x \in \mathbb{R}^K$ is the feature vector with an acquisition cost $c_\alpha \geq 0$ assigned to each

of the features $\alpha = 1, \dots, K$.¹

For a given example x , we assume that once it pays the cost to acquire a feature, its value can be efficiently cached; and subsequent use of the feature value does not incur additional cost. Thus, the cost of utilizing a particular prediction function, denoted by $C(f, x)$, is computed as the sum of the acquisition cost of *unique* features required by f for x .

1.2.2 Computational Cost

$C(f, x)$ can also measure the amount of computation required to compute $f(x)$. In a decision tree f , for example, it is proportional to the number of internal nodes x traverses. In a neural network, it is proportional to the number of layers and the number of connections between the layers.

1.2.3 Communication/Latency Cost

In mobile applications, prediction $f(x)$ may involve communication between the edge device and the server (cloud). $C(f, x)$ can capture such costs in terms of communication/latency cost.

1.3 Challenges

The problem of learning to prediction under a budget might appear well-studied as formulated in Eq.(1.2), which consists of an empirical loss minimization subject to a constraint. Indeed, the sparse learning or feature selection problem is an instance of learning to predict under a budget. Consider each feature element carries a unit acquisition cost and \mathcal{F} is the space of linear regressors. Each $f \in \mathcal{F}$ can be parame-

¹Note that our algorithms can be adapted to handle group-structured features where several elements in x may be associated with one feature acquisition cost. In other words, several elements in the x vector can be obtained together by paying the acquisition cost for one feature. We avoid it in the exposition for clarity purpose.

terized by $w \in \mathbb{R}^K$. The cost $C(f, x)$ is equal to the number of non-zero elements in w : $C(f, x) = \|w\|_0$. The budget constraint on the prediction-time feature acquisition cost thus reduces to a sparsity constraint on w . The sparse linear regression problem is

$$\min_{w \in \mathbb{R}^K} \frac{1}{n} \sum_{i=1}^n (y^{(i)} - w^T x^{(i)})^2, \text{ s.t. } \|w\|_0 \leq B. \quad (1.3)$$

Algorithms including LASSO (Tibshirani, 1996) and other subset selection methods have been well established (Miller, 2002). Yet we highlight that the goal of traditional sparse learning or feature selection is to identify a subset of the features to be used for all the examples. The assumption is that there exists a common subset of features that are useful for predicting *all* examples. But in practice, different examples may benefit from different subsets of features. Consider the medical diagnosis example, it makes sense to recommend different subsets of medical tests for different patients, depending on their individual conditions. In other words, the decision functions that we seek to learn are more general, able to *adapt* to different input examples.

The key idea in our budgeted prediction framework is to recognize that in many machine learning tasks not all input examples are created equal. There are “easy” examples that can be predicted at low cost (e.g. using a few low cost features or going through a small number of layers in a neural network). Only the “difficult” examples require more cost (e.g. using more features or going through many layers in a neural network). Since the budget constraint is on the average prediction cost over the examples, we can achieve high prediction accuracy by allocating less budget on the “easy” examples and more budget on the “difficult” ones.

In some sense, the family of decision functions \mathcal{F} in Eq. (1.2) that we optimize over is a family of adaptive decision rules, or decision policies, rather than static models such as a linear predictor. We highlight several challenges this entails.

- **Distinguish “easy” V.s. “difficult” examples:** Given a training dataset, it is not clear how to partition the examples into “easy” V.s. “difficult” ones. The partition function itself is a classifier that needs to be learned. Furthermore, how the dataset is partitioned impacts the data distribution for the downstream prediction models. In other words, the partition function should be learned jointly with a “cheap” prediction model that handles the “easy” examples as well as an “expensive” prediction model that handles the “difficult” ones. This interdependency translates to products of indicator functions in the optimization objective and leads to non-convexity.
- **Combinatorial state space:** With feature acquisition costs, the adaptive decision rule can be represented by a Directed Acyclic Graph (DAG) (Wang et al., 2015). The internal nodes correspond to feature subsets and decision functions at each node choose whether to acquire a new feature or predict using the already acquired features. The edges correspond to acquiring new features, transitioning from one feature subset to another. The number of states, or feature subsets, is 2^K , where K is the number of features. Learning decision functions for each state becomes intractable when the number of features is large.

1.4 Contribution

We develop several novel algorithmic approaches to the learning under prediction budget problem, improving the state of the art performance with each method. More importantly, through these methods, we develop a new *bottom-up* paradigm for the learning under prediction budget problem. Here we give a summary of the contributions made in this thesis.

- We propose a nearest neighbor approach to feature selection that incorporates

margins in classification.

- We propose to learn the adaptive decision rule as random forests. We propose a family of impurity measures and a splitting criteria so that the decision trees we grow are guaranteed to have near-optimal feature acquisition costs.
- Given any random forest, we propose to prune it to optimize expected feature cost & accuracy. We pose pruning RFs as a novel 0-1 integer program and establish total unimodularity of the constraint set to prove that the corresponding LP relaxation solves the original integer program. We further exploit connections to combinatorial optimization and develop an efficient primal-dual algorithm that scales to large problems. This *bottom-up* pruning approach circumvents the need for combinatorial search faced by the *top-down* approaches.
- We develop an adaptive approximation framework as a general *bottom-up* approach. The framework incorporates general machine learning models such as RF, boosting, SVM and neural networks. It also accounts for various types of costs such as feature acquisition, computational and communication/latency costs.
- We propose an on-line learning framework for the adaptive approximation and provide regret analysis.

1.5 Related Work

The problem of learning from full training data for prediction-time cost reduction (MacKay, 1992) has been extensively studied. We summarize related work according to the key properties in their approaches. We focus on the feature acquisition costs first.

1.5.1 Non-adaptive methods

The non-adaptive methods reduce prediction-time cost by identifying a common sparse subset of features that are used by all examples. Some of these methods include subset selection (Miller, 2002) and L1 regularization (Tibshirani, 1996). The Greedy Miser (Xu et al., 2012) is a non-linear method in this category. It is an adaptation of gradient boosted regression trees in the setting of feature acquisition costs. The algorithm iteratively adds weak learners (low-depth regression trees) to the ensemble by trading off the goodness fit to the current gradient and the additional feature cost introduced. The typical trees are limited to low-depth (4 or 5 levels) to avoid overfitting. As a result, we consider it a non-adaptive method because all the examples typically encounter the same set of features. Furthermore, the training algorithm does not consider feature usage at a per-example basis and it bears more similarity to a stepwise feature selection process. In contrast, the methods we propose in this thesis are adaptive in the sense that different examples can be routed differently in the decision rule and incur different costs.

1.5.2 Fixed feature acquisition

Among the adaptive methods, some assume a feature acquisition graph is given a priori, which is fixed by domain experts or enumerated in the case of just a few features. The task reduces to learning reject functions as well as end classifiers in the case of detection cascades (Viola and Jones, 2001). (Trapeznikov and Saligrama, 2013) generalize detection cascades to classifier cascades to handle balanced and/or multi-class scenarios. They solve a stage-wise empirical minimization problem and use cyclic optimization to iterate over the stages. (Wang et al., 2014b) extends the cascade to tree structures and formulated a convex surrogate that bounds the global empirical risk. (Wang et al., 2015) extend the tree structure to directed acyclic graphs

(DAGs) and trains decision functions for each node of the graph according to a child-to-parent order. In contrast to these methods based on fixed feature acquisition graph, ours proposed in this thesis aim to learn such sequential feature acquisition graphs.

1.5.3 Myopic feature acquisition

Due to the combinatorial search space for sequential feature acquisition, many methods resort to greedy/myopic strategies based on utility of acquiring each feature. (Ji and Carin, 2007) model the sequential decision process for feature acquisition and classification as a POMDP. But due to the difficulty associated with a POMDP formulation such as high computational cost, the need to quantize features and the lack of mechanism to rule out repeated actions, the authors proposed a myopic approximation. The utility of an action is evaluated by the difference between its cost and the reduction in the Bayes risk, as computed from the probability model. On the other hand, (Kanani and Melville, 2008) propose to define utility as difference in unlabeled margin divided by feature acquisition cost. Without assuming probability model, they propose to estimate feature value distribution by discretizing the feature values and learning classifiers based on available features to predict its distribution. (Gao and Koller, 2011) propose a locally weighted regression method during test time and assume a Gaussian model to myopically select features based on information gain of unknown features. The above methods tend to have high computational cost during prediction-time and require generative assumptions.

(Sheng and Ling, 2006) propose a sequential batch test algorithm to minimize total cost of acquiring features and misclassification. Used with decision trees, it selects features to split the internal nodes based on their utility values. This heuristic utility is defined as the difference of the expected cost before and after acquiring the feature. In Chapter 3, we propose a different utility measure as well as splitting criteria that lead to near-optimal cost guarantee.

1.5.4 Non-myopic feature acquisition

In general, learning non-myopic feature acquisition rules is computationally intractable. However, under certain assumptions and for specialized settings, this is achievable. (Busa-Fekete et al., 2012) formulate the decision process as an MDP. The features they consider are ordered base learners obtained from AdaBoost. At each base learner, the actions to take are to Evaluate, Skip or Quit. The Quit action leads to a final classification. The state is composed of the sum of evaluated weak learner outputs so far as well as the index of current base learner. By associating each Evaluate action with a cost, the MDP reward is to minimize the final classification loss plus the total cost. The formulation in (Busa-Fekete et al., 2012) is primarily based on the fixed weak learner order, which helps reduce the action space. (Bilgic and Getoor, 2007) introduce a novel data structure called Value of Information Lattice (VOILA) to calculate value of information for subsets of features. VOILA is a directed graph where each node represents a unique subset of the features and each edge represents a subset relationship between its nodes. In order to reduce the exponential number of feature subsets, a Bayesian network over the features as well as the class variable is assumed given. (Karayev et al., 2012) formulates a problem of object recognition under time constraint as an MDP. The actions correspond to running different detectors or classifiers. A state includes current estimates of the distribution of class presence, the history of actions taken together with the resulting observations, as well as the time costs so far and the time budget left. The state-action pair is featurized by concatenating the prior distribution of the classes for the action, the distribution of the classes as well as the entropies for all classes conditioned on observations so far. The long term reward function Q is modeled as a inner product of the state-action pair and a vector parameterizing the policy. A convenient property in the above problem is that the evaluation function is additive per action, as computed by the change in

average precision introduced by the action. (Zubek and Dietterich, 2002) also formulates the problem of classification with feature acquisition costs as an MDP. They use heuristic search algorithms to reduce the search space for the optimal policy. But this approach requires discretization of the feature values and is inefficient when the state space is large.

Another line of methods aims to learn feature acquisition rules in a discriminative framework through empirical risk minimization. (Chen et al., 2012) aim to re-order a set of pre-training base learners to reduce prediction costs. It optimizes the parameters of the stages in cycles. Multiple levels of relaxations are proposed to make the optimization objective continuous and differentiable. Still, the proposed algorithm faces computational difficulty as it needs to solve a non-convex optimization problem during each cycle.

(Xu et al., 2013) propose to learn a tree of classifiers. The tree structure is a limited-depth balanced binary tree. Each path of the tree requires a different set of features so as to reduce test-time feature cost. Similar to (Chen et al., 2012), they propose to minimize the sum of losses at all internal nodes plus a weighted cost term. Several relaxations are used to make computation tractable. Cyclic optimization is used to learn the classifier at each node while fixing all other nodes. During test time, an example is routed at each internal node by a linear classifier to determine the probability of going left or right and the branch is taken with respect to these probabilities. (Kusner et al., 2014) recognize that the method in (Xu et al., 2013) is hard to train and requires involved optimization hyperparameter tuning. They propose a simpler training procedure based on greedy selection.

1.5.5 Other methods

Besides feature acquisition costs, many researchers have considered reducing computational costs as well as memory usage during prediction-time. The distillation

framework (Hinton et al., 2015; Lopez-Paz et al., 2016) aims to compress a complex teacher model into a smaller student model without losing much accuracy. (Kumar et al., 2017) propose a compact tree model called Bonsai that achieves high accuracy with small model size. (Gupta et al., 2017) propose PtotoNN as a compressed K-Nearest Neighbor algorithm. The main idea is to learn a small number of prototypes to represent the entire training set and jointly learn a projection matrix to reduce dimensionality as well. Reducing the prediction costs of deep neural networks has also been an area of recent interest (Bolukbasi et al., 2017; Lin et al., 2017).

1.6 Organization

The rest of this thesis is organized in the following manner:

Chapter 2 will describe a myopic algorithm for feature acquisition based on margin and nearest neighbors. We will explain the intuitive advantage of this simple algorithm and also show numerical experiments.

Chapter 3 will describe a new decision tree growing algorithm that incorporates feature acquisition cost. Even though the algorithm is myopic, we will provide theoretical guarantee to show that it achieves near-optimal cost. We then expand to ensembles of such decision trees and illustrate performance with numerical experiments.

Chapter 4 will describe a novel method to prune random forests to optimize feature acquisition costs and accuracy. We will provide detailed formulation and theoretical guarantee that the pruning optimization problem can be solved in polynomial time. We will further provide a specialized primal-dual algorithm that can scale to large datasets. Finally, we will evaluate the performance with numerical experiments.

Chapter 5 will motivate and describe a novel framework of adaptive approximation of general models for prediction-time cost reduction. We will formulate an

optimization problem and point out the computational advantages compared to previous approaches. This general framework is then specialized into linear and gradient boosted models. Again, we will evaluate the performance with numerical experiments.

Chapter 6 will study the adaptive approximation problem in an on-line setting with limited feedback. We provide theoretical analysis of the regret.

Chapter 7 will discuss future directions. We will introduce the problem of distributed prediction and explain our formulation. We will also show some preliminary experimental results. We will also consider extensions of the regret analysis of the on-line adaptive approximation problem.

Chapter 2

Margin-based Nearest Neighbor Approach

We introduce a novel algorithm to dynamically select features for every test instance until we reach a desired classification accuracy. We assume we have access to a training set with full features and corresponding class labels. For every test point, there is a cost associated with measuring or computing each feature. Our system acquires one feature at a time, adaptively deciding which feature to request next or when to stop and classify. We learn such a policy by utilizing training examples within a neighborhood of a test point. The key challenge in learning such a decision system is to correctly determine the neighborhood. After acquiring a partial set of features, we can not infer the true distance from a test point to training points in the full feature space. In other words, the nearest neighbor based on partial feature measurement may not be a true neighbor in the full feature space. We call this difficulty partial neighborhood confusion. Algorithms that try to learn the label of a test point based on the labels of training points in the partial neighborhood tend to perform poorly due to this difficulty.

In contrast, to make our approach more robust to such partial neighborhood confusion, we incorporate classification margins in our system. In binary classification, a margin of an example is typically an output of a decision functions times the label ($+1/-1$) of an example. Margins are widely used as a measure of classification confidence. A large positive margin indicates high confidence, while a negative margin indicates an incorrect decision. Maximizing margins has led to many powerful tools

in machine learning such as SVM, boosting, etc ((Cortes and Vapnik, 1995)). We use margins to estimate the probability of correct classification and sequentially maximize this probability at each stage of the decision making process. Since the label of a test point is unknown, its margin cannot be computed directly. To overcome this problem, our algorithm learns the unknown test margin from the training data in the partial neighborhood of this test point. Recall that feature values and labels are known for the training data, hence margins are also fully known. Since our algorithm learns margin information instead of class label from nearest neighbors based on partial feature measurement, we are more robust to the partial neighborhood confusion problem. Intuitively, points far from each other in the full feature space are unlikely to share the same label but may produce the same sign margins on the same feature. We will illustrate this point further through an example in Section 2.3 and Experiments in Section 2.4.

The work presented in this chapter is published in (Nan et al., 2014).

2.1 Related Work

The method we propose in this chapter involves myopic feature acquisition as discussed in Section 1.5. Different from (Ji and Carin, 2007; Gao and Koller, 2011), we do not assume specific probability distribution. Different from (Kanani and Melville, 2008), we use labeled margin of nearest neighbors rather than unlabeled margin.

2.2 Problem Setup

Given the training set of N data points and corresponding labels $(x^{(l)}, y^{(l)}), l = 1, \dots, n$, each point has d features $x^{(l)} \in \mathfrak{R}^d$, and we assume all features are known for training. Given an unknown test point, a feature j can be measured or acquired for a cost $c_j, j = 1, \dots, d$. We assume we are given a linear classifier, $f(x) = w^T x$, trained

on the entire training set. Note we omit the offset term in our discussion because it can be considered as an additional (constant) feature of the data point.

Remark: We assume a linear classifier is used for the entire data set. This is not as restrictive as it may appear. In fact, kernel SVM is linear in the transformed feature space and Boosting (Freund and Schapire, 1997) is linear once we consider weak learners as transformed features. We will show in our experiment that our algorithm works with both SVM and Boosting.

In the rest of this section, we explain our dynamic feature selection approach for a new test point, x . Let \mathcal{O} be the index set of measured features and $\overline{\mathcal{O}}$ be the index set of the remaining features. We use $w_{\mathcal{O}}$ to denote the elements of w indexed by \mathcal{O} . For ease of notation we use i to denote the index of the next potential feature to be measured. Initially $\mathcal{O} = \emptyset$. We can choose the most discriminative feature as the first feature. Let $x_{\mathcal{O}}$ denote the measurement values obtained about the test point and we set the unmeasured feature values to be 0¹, $x_{\overline{\mathcal{O}}} = 0$. If a classification is needed with the current measurements we can simply compute

$$y = w_{\mathcal{O}}^T x_{\mathcal{O}}, \quad (2.1)$$

and decide based on its sign.

Given any measured feature set \mathcal{O} , it is not clear how $w_{\mathcal{O}}^T x_{\mathcal{O}}$ relates $w^T x$ (a decision when all features are measured). However, assume we choose the features in \mathcal{O} to produce positive margins on the neighboring training points. In this scenario, these features will most likely also produce positive margins on the test point and result in accurate classification based on (2.1). To be more concrete, we define a partial neighborhood $N(\mathcal{O})$ of the test point as the index set of those training points that are close to $x_{\mathcal{O}}$ on the index set \mathcal{O} . We define $N(\mathcal{O})$ to contain the K nearest neighbors

¹Note that this is a missing feature classifier. While there has been some work (see (Maaten et al., 2013)) on learning classifiers robust to missing features, this is outside the scope of this paper.

(with respect to Euclidean distance) of $x_{\mathcal{O}}$ in the training set, where K is a positive natural number.²

Next, define the partial margin of the k th training point in the neighborhood $N(\mathcal{O})$ based on the current measurement feature set \mathcal{O} as

$$\eta_k^{\mathcal{O}} = y^{(k)}(w_{\mathcal{O}}^T x_{\mathcal{O}}^{(k)}), k \in N(\mathcal{O}). \quad (2.2)$$

If $\eta_k^{\mathcal{O}}$ is positive then (2.1) will give correct classification based on the measured feature set \mathcal{O} . Similarly, define the one-step-ahead partial margin of the k th training point in the neighborhood $N(\mathcal{O})$ based on the current measurement feature set \mathcal{O} and feature i as

$$\eta_{i,k}^{\mathcal{O}} = y^{(k)}(w_{\mathcal{O}}^T x_{\mathcal{O}}^{(k)} + w_i x_i^{(k)}), k \in N(\mathcal{O}), i \in \overline{\mathcal{O}}. \quad (2.3)$$

To estimate classification accuracy, we define the partial probability of correct classification of the test point based on current measurement feature set \mathcal{O} as the ratio of the number of correct classifications to the total number of training points within the neighborhood:

$$p^{\mathcal{O}} = \frac{\#\{k : \eta_k^{\mathcal{O}} > 0\}}{|N(\mathcal{O})|}. \quad (2.4)$$

Similarly we define the one-step-ahead partial probability of correct classification of the test point based on current measurement feature set \mathcal{O} and feature i as

$$p_i^{\mathcal{O}} = \frac{\#\{k : \eta_{i,k}^{\mathcal{O}} > 0\}}{|N(\mathcal{O})|}. \quad (2.5)$$

At each step, we can decide to measure the next feature or to stop based on the accuracy estimate $p^{\mathcal{O}}$. And $p_i^{\mathcal{O}}$ provides an estimate of how much accuracy we can get by measuring i as the next feature. We can thus choose the i that gives the best accuracy-cost trade-off.

²While there are many ways to define a neighborhood (i.e. based on thresholding a distance metric between $x_{\mathcal{O}}$'s), we focus on KNN in this paper

2.3 Algorithm

We present our algorithm in Algorithm 1.

Algorithm 1 Fast Margin-based Cost-sensitive Classification (FMCC)

- 1: Train classifier $y = w^T x$ on the entire training data
 - 2: Fix accuracy-cost tradeoff parameter α
 - 3: Given a test point x :
 - 4: Measure feature i
 - 5: **for** $t = 1 \rightarrow d$ **do** *▷ Iterate through the total number of features*
 - 6: **if** $p^\mathcal{O} > \text{threshold}$ **then**
 - 7: Stop, make classification
 - 8: **else**
 - 9: Compute neighborhood $N(\mathcal{O})$
 - 10: **for** $i \in \overline{\mathcal{O}}$ **do**
 - 11: Update partial margins $\eta_{i,k}^\mathcal{O}$ for $k \in N(\mathcal{O})$ according to (2.3)
 - 12: Compute $p_i^\mathcal{O}$ according to (2.5)
 - 13: Select feature $i_{\max} = \operatorname{argmax}_i p_i^\mathcal{O} - \alpha c_i$ to measure next, $\mathcal{O} \leftarrow (\mathcal{O}, i_{\max})$.
 - 14: Update partial margins $\eta_k^\mathcal{O}$ for $k \in N(\mathcal{O})$ according to (2.2)
 - 15: Compute $p^\mathcal{O}$ according to (2.4)
-

Suppose there are 8 training data points as shown in Figure 2-1. The class labels are indicated in red disks (label -1) and black triangles (label 1), with the weight (number of repeated training examples) shown besides them. For each training point, we also display the coordinates. By inspection, to locate an unknown test point (assuming it follows the distribution of the training data), the optimal strategy would be to measure x_2 , then x_1 and lastly x_3 . We show that our algorithm indeed follows this strategy by simply computing the margins.

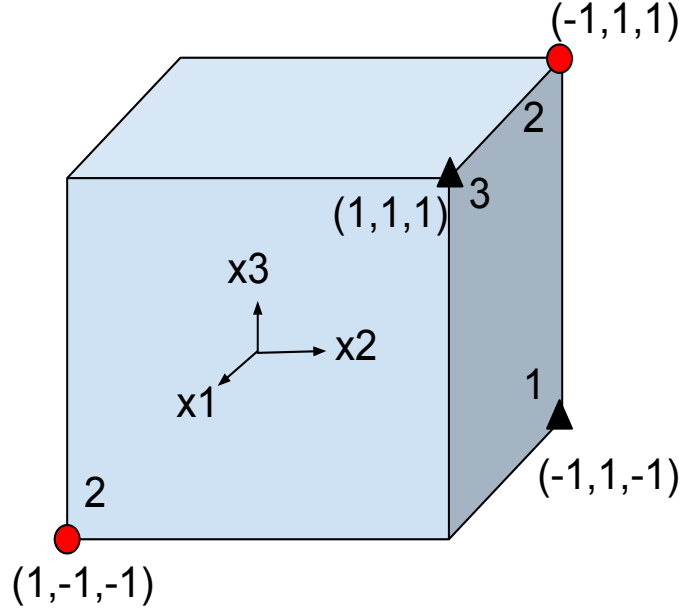


Figure 2.1: An example of cost sensitive learning. Given 8 training points, each is binary with 3 features: $x^{(1)} = x^{(2)} = (1, -1, -1)$, $x^{(3)} = x^{(4)} = (-1, 1, 1)$, $x^{(5)} = (-1, 1, -1)$, $x^{(6)} = x^{(7)} = x^{(8)} = (1, 1, 1)$, with labels $y^{(1)}, \dots, y^{(4)} = -1, y^{(5)}, \dots, y^{(8)} = 1$. They are linearly separable with optimal SVM solution $y = w' * x + b = (0.9995, 1.4998, -0.5002)x - 0.9997$

Suppose all the features carry the same measurement cost. And the partial neighborhood is defined to be those training points having exactly the same feature values as the test point on $x_{\mathcal{O}}$. We apply our algorithm sequentially as follows. Step 1. Measuring x_1, x_2, x_3 will give $2+2=4$, $2+1+3=6$, $2+2=4$ correct classifications based on (2.1), respectively. So measuring x_2 will result in higher accuracy. Step 2. Suppose x_2 has been measured and it's equal to -1, there are 2 points in $N(\mathcal{O}) = \{1, 2\}$. Compute $\eta_1^{\mathcal{O}} = \eta_2^{\mathcal{O}} > 0$ using (2.2). So the estimated accuracy $p^{\mathcal{O}} = 1$. Stop and classify, giving the correct classification. Suppose x_2 is measured to be 1, there are 6 points in $N(\mathcal{O}) = \{3, 4, 5, 6, 7, 8\}$. As we contemplate on measuring x_1 next, compute $\eta_{1,3}^{\mathcal{O}} = \eta_{1,4}^{\mathcal{O}} > 0, \eta_{1,5}^{\mathcal{O}} < 0, \eta_{1,6}^{\mathcal{O}} = \eta_{1,7}^{\mathcal{O}} = \eta_{1,8}^{\mathcal{O}} > 0$ using (2.3). Therefore we obtain $p_1^{\mathcal{O}} = \frac{5}{6}$ using (2.5). Similarly we obtain $p_3^{\mathcal{O}} = \frac{3}{6}$. This suggests measuring x_1 next will results

in higher accuracy, which agrees with the optimal strategy.

Analysis: When the full set of features are measured, $p^{\mathcal{O}}$ in (2.4) is unbiased estimator of the correct classification probability. The training points in the neighborhood $N(\mathcal{O})$ obey the probability distribution of the training points. And (2.4) is the sample mean of the actual classification accuracy according to (2.1). And we assume all data points are i.i.d hence we get the result. We can regard $p^{\mathcal{O}}$ as a good estimate of the probability of correct classification in the (finite) limit sense (when the number of measured features increases to the maximum). We can also show our algorithm has test time complexity that scales linearly in sample size. There are at most d iterations (the total number of features) and each iteration involves only $\mathbf{O}(ndK)$ operations, where K is a constant neighborhood size. Thus, our algorithm is well suited when the number of training examples are not too large or the test time computation budget is large. In contrast, the VoC algorithm requires solving a locally weighted least square problem at each iteration and the per iteration complexity is $\mathbf{O}(n^2d^3)$, which is much higher.

2.4 Experiments

We evaluate our algorithm on three UCI data sets (Dheeru and Karra Taniskidou, 2017). To demonstrate the wide applicability of our algorithm, we base our algorithm on Boosting for the first two data set and linear SVM for the third data set.

Performance Metric: A natural way to evaluate performance of budged learning is compare accuracy vs the number of features acquired. The objective is to achieve high classification accuracy while acquiring as few features as possible. We assume acquisition cost for all features is uniform.

Letter Recognition Data Set: This is a multi-class data set with the goal of distinguishing 26 capital letters in the English alphabet from a large number of

		Number of measured features										
		1	5	10	15	20	25	30	35	40	45	50
65	Act	.503	.663	.746	.847	.881	.888	.895	.895	.895	.890	.895
	Est	.513	.657	.763	.868	.908	.914	.918	.920	.917	.905	.910
125	Act	.503	.672	.769	.870	.883	.887	.888	.891	.891	.882	.895
	Est	.514	.653	.778	.884	.902	.907	.910	.910	.907	.895	.909
185	Act	.503	.676	.779	.875	.879	.883	.884	.889	.892	.879	.895
	Est	.513	.647	.783	.887	.897	.901	.904	.904	.901	.888	.909

Table 2.1: The actual and estimated probabilities of correct classification for neighborhood sizes 65, 125 and 185.

black-and-white rectangular pixel displays. Each data point consists of 16 features. We randomly draw 200 examples from each letter as training points and 100 examples as test points and assign the first 13 letters to one class and the other 13 letters to the other class. Results are shown from 50 randomly drawn sets of data to prevent sampling bias. We train a boosted collection of 1000 stumps on the training set, where each stump thresholds a single feature. Evaluating the stumps for each feature yields a set of margins. For comparison, we implement the probabilistic model-based Value of Classifier (VoC) algorithm (Gao and Koller, 2011), with the negative of classification loss as the reward function. We set $\lambda = 0.5$ in the locally weighted regression and the bandwidth parameter β is set to be the median of the distances from the test point to the training points at each step. We also compare to a random order scheme, where the next feature to measure is chosen at random and classification at each step is computed by (2.1). We see from Fig. 2-2 that our FMCC is close to VoC and outperforms the baseline. For small budgets (few observed features), VoC achieves higher accuracy than FMCC, however after measuring 6 features, FMCC outperforms VoC. This behavior is expected, as the estimated neighborhood of each example are unreliable when few features have been observed. Additionally, the FMCC algorithm has lower computational complexity than VoC; at each stage of feature acquisition, FMCC computes the partial neighborhood of the test example and the estimated probability of correct classification. which

Landsat Satellite Data Set: The Landsat data set contains 4435 and 2000 training and test points, respectively. Each data point is 36-dimensional satellite image features and belongs to 1 of 6 classes of soil. We consider binary classification by assigning the first 3 classes to class -1 and the other 3 classes to class 1 . We train a boosted collection of 1000 stumps on the training set, where each stump thresholds a single feature. Fig. 2.2 shows that FMCC outperforms VoC after 4 feature measurements and outperforms the baseline, which gives similar result as in the Letters dataset.

MiniBooNE Particle Identification Data Set: The MiniBooNE data set is a binary classification task, with the goal of distinguishing electron neutrinos (signal) from muon neutrinos (background). Each data point consists of 50 experimental particle identification variables (features). We train a linear SVM on 1000 training points randomly chosen, with an equal number drawn from each class. The test classification accuracy is evaluated by randomly drawing 300 data points from each class and the results are averaged over 50 cross validations. Fig. 2.2 shows that FMCC achieves higher classification accuracy than VoC and the random schemes for any given number of measured features (cost) greater than 2. We believe that VoC performs poorly on this data set as the features are poorly modeled by a mixture of Gaussian distributions. In Table 2.1 we also compare the partial probability of correct classification $p^{\mathcal{O}}$ (See Eq(2.4)), against the actual test classification accuracy across different neighborhood sizes. We observe that $p^{\mathcal{O}}$ in our algorithm provides a good estimate of the true probability of correct classification thus it can be used reliably for accuracy-cost trade-off. Furthermore, it also shows our algorithm is robust to the neighborhood $N(\mathcal{O})$ definition.

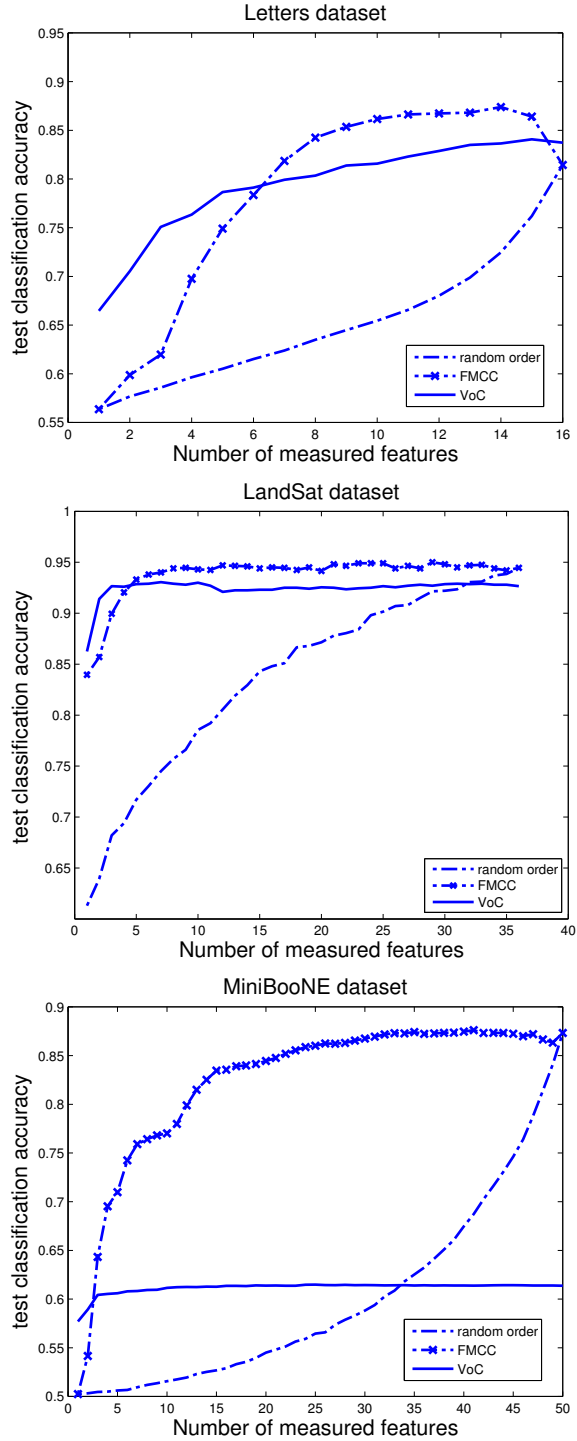


Figure 2.2: Experiment result of classification accuracy vs number of features measured on Letters, LandSat MiniBooNE datasets. FMCC is consistent across all datasets while the VoC does not perform well on the MiniBooNE dataset.

Chapter 3

Feature-Budgeted Random Forest

In this chapter we propose a novel random forest learning algorithm to minimize prediction error for a user-specified average feature acquisition budget. Random forests (Breiman, 2001) construct a collection of trees, wherein each tree is grown by random independent data sampling and feature splitting, producing a collection of independent identically distributed trees. The resulting classifiers are robust, are easy to train, and yield strong generalization performance.

Although well suited to unconstrained supervised learning problems, applying random forests in the case of prediction-time budget constraints presents a major challenge. First, random forests do not account for feature acquisition costs. If two features have similar utility in terms of power to classify examples but have vastly different costs, random forest is just as likely to select the high cost feature as the low cost alternative. This is obviously undesirable. Second, a key element of random forest performance is the diversity among trees (Breiman, 2001). Empirical evidence suggest a strong connection between diversity and performance, and generalization error is bounded not only with respect to the strength of individual trees but also the correlation between trees (Breiman, 2001). High diversity among trees constructed without regard for acquisition cost results in trees using a wide range of features, and therefore a high acquisition cost (See Section 3.5).

Thus, ensuring a low acquisition cost on the forest hinges on growing each tree with high discriminative power and low acquisition cost. To this end, we propose to learn

decision trees that incorporates feature acquisition cost. Our random forest grows trees based on greedy minimax cost-weighted-impurity splits. Although the problem of learning decision trees with optimally low-cost is computationally intractable, we show that our greedy approach outputs trees whose cost is closely bounded with respect to the optimal cost. Using these low cost trees, we construct random forests with high classification performance and low prediction-time feature acquisition cost.

Abstractly, our algorithm attempts to solve an empirical risk minimization problem subject to a budget constraint. At each step in the algorithm, we add low-cost trees to the random forest to reduce the empirical risk until the budget constraint is met. The resulting random forest adaptively acquires features during prediction time, with features only acquired when used by a split in the tree. In summary, our algorithm is greedy and easy to train. It can not only be parallelized, but also lends itself to distributed databases. Empirically, it does not overfit and has low generalization error. Theoretically, we can characterize the feature acquisition cost for each tree and for the random forest. Empirically, on a number of benchmark datasets we demonstrate superior accuracy-cost curves against state-of-the-art prediction-time algorithms.

The work presented in this chapter is published in (Nan et al., 2015).

3.1 Related Work

Supervised learning approaches with prediction-time budgets have previously been studied under an empirical risk minimization framework to learn budgeted decision trees (Xu et al., 2013; Kusner et al., 2014; Trapeznikov and Saligrama, 2013; Wang et al., 2014b; Wang et al., 2014a). In this setting, construction of budgeted decision cascades or trees has been proposed by learning complex decision functions at each node and leaf, outputting a tree of classifiers which adaptively select sensors/features

to be acquired for each new example. Common to these systems is a decision structure, which is a priori fixed. The entire structure is parameterized by complex decision functions for each node, which are then optimized using various objective functions. In contrast we build a random forest of trees where each tree is grown greedily so that global collection of random trees meets the budget constraint. (Xu et al., 2012) incorporates feature acquisition cost in stage-wise regression during training to achieve prediction-time cost reduction.

Construction of simple decision trees with low costs has also been studied for discrete function evaluation problems (Cicalese et al., 2014; Moshkov, 2010; Bellala et al., 2012). Different from our work these trees operate on discrete data to minimize function evaluations, with no notion of test time prediction or cost.

As for Random forests despite their widespread use in supervised learning, to our knowledge they have not been applied to prediction-time cost reduction.

3.2 Problem Setup

As discussed in Section 1.2, we minimize the empirical loss subject to a budget constraint according to (1.2), copied below for easy reference.

$$\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n L(y^{(i)}, f(x^{(i)})), \text{ s.t. } \frac{1}{n} \sum_{i=1}^n C(f, x^{(i)}) \leq B, \quad (3.1)$$

In our context the classifier f is a random forest, \mathcal{T} , consisting of m random trees, D_1, D_2, \dots, D_m , that are learned on training data. Consequently, the expected cost for an instance x during prediction-time can be written as follows:

$$E_f [E_x [C(f, x)]] \leq \sum_{j=1}^m E_{D_j} [E_x [C(D_j, x)]] \quad (3.2)$$

where, in the RHS we are averaging with respect to the random trees. As the trees in a random forest are identically distributed the RHS scales with the number of trees.

This upper-bound captures the typical behavior of a random forest due to the low feature correlation among trees.

As a result of this observation, the problem of learning a budgeted random forest can be viewed as equivalent to the problem of finding decision trees with low expected evaluation cost and error. This motivates our algorithm BUDGETRF, where greedily constructed decision trees with provably low feature acquisition cost are added until the budget constraint is met according to validation data. The returned random forest is a feasible solution to (1.2) with strong empirical performance.

3.3 Algorithm

During Training: As shown in Algorithm 2, there are seven inputs to BUDGETRF: impurity function F , prediction-time feature acquisition budget B , a cost vector $C \in \mathbb{R}^m$ that contains the acquisition cost of each feature, training class labels y_{tr} and data matrix $X_{tr} \in \mathbb{R}^{n \times K}$, where n is the number of samples and K is the number of features, validation class labels y_{tv} and data matrix X_{tv} . Note that the impurity function F needs to be *admissible*, which essentially means monotone and supermodular. We defer the formal definition and theoretical results to Section 3.4. For now it is helpful to think of an impurity function F as measuring the heterogeneity of a set of examples. Intuitively, F is large for a set of examples with mostly different labels and small for a set with mostly the same label.

BUDGETRF iteratively builds decision trees by calling GREEDYTREE as a subroutine on a sampled subset of examples from the training data until the budget B is exceeded as evaluated using the validation data. The ensemble of trees are then returned as output. As shown in subroutine GREEDYTREE, the tree building process is greedy and recursive. If the given set of examples have zero impurity as measured by F , they are returned as a leaf node. Otherwise, compute the *risk* $R(t)$ for each

Algorithm 2 BUDGETRF

```

1: procedure BUDGETRF( $F, B, C, y_{tr}, X_{tr}, y_{tv}, X_{tv}$ )
2:    $\mathcal{T} \leftarrow \emptyset$ .
3:   while Average cost using validation set on  $\mathcal{T} \leq B$  do
4:     Randomly sample  $n$  training data with replacement to form  $X^{(i)}$  and  $y^{(i)}$ .
5:     Train  $T \leftarrow \text{GREEDYTREE}(F, C, y^{(i)}, X^{(i)})$ .
6:      $\mathcal{T} \leftarrow \mathcal{T} \cup T$ .
7:   return  $\mathcal{T} \setminus T$ .

```

Subroutine - GREEDYTREE

```

8: procedure GREEDYTREE( $F, C, y, X$ )
9:    $S \leftarrow (y, X)$  ▷ the current set of examples
10:  if  $F(S) = 0$  then return
11:  for each feature  $t = 1$  to  $K$  do
12:    Compute  $R(t) := \min_{g_t \in \mathcal{G}_t} \max_{i \in \text{outcomes}} \frac{c(t)}{F(S) - F(S_{g_t}^i)}$ , ▷ risk for feature  $t$ 
13:    where  $S_{g_t}^i$  is the set of examples in  $S$  that has outcome  $i$  using classifier  $g_t$ 
    with feature  $t$ .
14:   $\hat{t} \leftarrow \text{argmin}_t R(t)$ 
15:   $\hat{g} \leftarrow \text{argmin}_{g_{\hat{t}} \in \mathcal{G}_{\hat{t}}} \max_{i \in \text{outcomes}} \frac{c(\hat{t})}{F(S) - F(S_{g_{\hat{t}}}^i)}$ 
16:  Make a node using feature  $\hat{t}$  and classifier  $\hat{g}$ .
17:  for each outcome  $i$  of  $\hat{g}$  do
18:    GREEDYTREE( $F, C, y_{\hat{g}}^i, X_{\hat{g}}^i$ ) to append as child nodes.

```

feature t , which involves searching for a classifier g_t among the family of classifiers \mathcal{G}_t that minimizes the maximum impurity among its outcomes. Intuitively, a feature with the least $R(t)$ can uniformly reduce the impurity among all its child nodes the most with the least cost. Therefore such a feature \hat{t} is chosen along with the corresponding classifier \hat{g} . The set of examples are then partitioned using \hat{g} to different child nodes at which GREEDYTREE is recursively applied. Note that we allow the algorithm to reuse the same feature for the same example in GREEDYTREE.

During Prediction: Given a test example and a decision forest \mathcal{T} returned by BUDGETRF, we run the example through each tree in \mathcal{T} and obtained a predicted label from each tree. The final predicted label is simply the majority vote among all the trees.

Different from random forest, we incorporate feature acquisition costs in the tree building subroutine GREEDYTREE with the hope of reducing costs while maintaining low classification error. Our main theoretical contribution is to propose a broad class of *admissible* impurity functions such that on any given set of n' examples the tree constructed by GREEDYTREE will have max-cost bounded by $O(\log n')$ times the optimal max-cost tree.

3.4 Bounding the Cost of Each Tree

Given a set of examples S with features and corresponding labels, a classification tree D has a feature-classifier pair associated with each internal node. A test example is routed from the root of D to a leaf node directed by the outcomes of the classifiers along the path; the test example is then labeled to be the majority class among training examples in the leaf node it reaches. The feature acquisition cost of an example $s \in S$ on D , denoted as $cost(D, s)$, is the sum of all feature costs incurred along the root-to-leaf path in D traced by s . Note that if s encounters a feature

multiple times in the path, the feature cost contributes to $\text{cost}(D, s)$ only once because subsequent use of a feature already acquired for the test example incurs no additional cost. We define the total max-cost as

$$\text{Cost}(D) = \max_{s \in S} \text{cost}(D, s).$$

We aim to build a decision tree for any given set of examples such that the max-cost is minimized. Note that the max-cost criterion bounds the expected cost criterion of Eq. 3.2. While this bound could be loose we show later (see Sec. 3.4.2) that by parameterizing a suitable class of impurity functions, the max-costs of our GREEDYTREE solution can be “smoothed” so that it approaches the expected-cost.

First define the following terms: n' is the number of examples input to GREEDYTREE and K is the number of features, each of which has (a vector of) real values; F is a given impurity function; $F(S)$ is the impurity on the set of examples S ; D_F is the family of decision trees with $F(L) = 0$ for any of its leaf L ; each feature has a cost $c(t)$; a family of classifiers \mathcal{G}_t is associated with feature t ; $\text{Cost}_F(S)$ is the max-cost of the tree constructed by GREEDYTREE using impurity function F on S ; and assume no feature is used more than once on the same example in the *optimal* decision tree among D_F that achieves the minimum max-cost, which we denote as $\text{OPT}(S)$ for the given input set of examples S . Note the assumption here is a natural one if the complexity of \mathcal{G}_t is high enough. This assumption is used in the proof of Lemma 3.4.1 to lower bound the cost of the optimal tree. We show the $O(\log n')$ approximation holds for the max-cost of the optimal testing strategy using the GREEDYTREE subroutine if the impurity function F is admissible.

Definition A function F of a set of examples is *admissible* if it satisfies the following five properties: (1) Non-negativity: $F(G) \geq 0$ for any set of examples G ; (2) Purity: $F(G) = 0$ if G consists of examples of the same class; (3) Monotonicity: $F(G) \geq$

$F(R), \forall R \subseteq G$; (4) Supermodularity: $F(G \cup j) - F(G) \geq F(R \cup j) - F(R)$ for any $R \subseteq G$ and example $j \notin R$; (5) $\log(F(S)) = O(\log n')$, where n' is the number of examples in S .

In Section 3.4.1 we provide several natural impurity functions that satisfy the above definition.

Since the set S is always finite, by scaling F we can assume the smallest non-zero impurity of F is 1. Let τ and \hat{g}_τ be the first feature and classifier selected by GREEDYTREE at the root and let $S_{\hat{g}_\tau}^i$ be the set of examples in S that has outcome i using classifier \hat{g}_τ . Note the optimization of classifier in Line (12) of Algorithm 2 needs not to be exact. We say GREEDYTREE is λ -greedy if \hat{g}_τ is chosen such that

$$\max_{i \in \text{outcomes}} \frac{c(\tau)}{F(S) - F(S_{\hat{g}_\tau}^i)} \leq \min_{g_t \in \mathcal{G}_t} \max_{i \in \text{outcomes}} \frac{\lambda c(t)}{F(S) - F(S_{g_t}^i)},$$

for some constant $\lambda \geq 1$. By definition of max-cost,

$$\frac{\text{Cost}_F(S)}{\text{OPT}(S)} \leq \frac{c(\tau) + \max_i \text{Cost}_F(S_{\hat{g}_\tau}^i)}{\text{OPT}(S)},$$

because feature τ could be selected multiple times by GREEDYTREE along a path and the feature cost $c(\tau)$ contributes only once to the cost of the path.

Let q be such that $\text{Cost}_F(S_{\hat{g}_\tau}^q) = \max_i \text{Cost}_F(S_{\hat{g}_\tau}^i)$. We first provide a lemma to lower bound the optimal cost, which will later be used to prove a bound on the cost of the tree.

Lemma 3.4.1 *Let F be monotone and supermodular; let τ and \hat{g}_τ be the first feature and classifier chosen by GREEDYTREE λ -greedily on the set of examples S , assume no feature is used more than once on any path of the optimal tree, then*

$$c(\tau)F(S)/(F(S) - F(S_{\hat{g}_\tau}^q)) \leq \lambda \text{OPT}(S).$$

Proof Let $D^* \in D_F$ be a tree with optimal max-cost. Let v be an arbitrarily chosen

internal node in D^* , let γ be the feature associated with v and g_γ^* the corresponding classifier. Let $R \subseteq S$ be the set of examples associated with the leaves of the subtree rooted at v . Let i be such that $c(\tau)/(F(S) - F(S_{\hat{g}_\tau}^i))$ is maximized. Let $g_\gamma^{min} = \operatorname{argmin}_{g_\gamma \in \mathcal{G}_\gamma} \max_{i \in \text{outcomes}} \frac{c(\gamma)}{F(S) - F(S_{g_\gamma}^i)}$. Let w be such that $c(\gamma)/(F(S) - F(S_{g_\gamma}^w))$ is maximized; similarly let j be such that $c(\gamma)/(F(S) - F(S_{g_\gamma}^j))$ is maximized. We then have:

$$\begin{aligned} \frac{c(\tau)}{F(S) - F(S_{\hat{g}_\tau}^i)} &\leq \frac{c(\tau)}{F(S) - F(S_{\hat{g}_\tau}^i)} \leq \frac{\lambda c(\gamma)}{F(S) - F(S_{g_\gamma}^w)} \\ &\leq \frac{\lambda c(\gamma)}{F(S) - F(S_{g_\gamma}^j)} \leq \frac{\lambda c(\gamma)}{F(R) - F(R_{g_\gamma}^j)}. \end{aligned} \quad (3.3)$$

The first inequality follows from the definition of i . The second inequality follows from the λ -greedy choice at the root. The third inequality follows from the minimization over classifiers given feature γ . To show the last inequality, we have to show $F(S) - F(S_{g_\gamma}^j) \geq F(R) - F(R_{g_\gamma}^j)$. This follows from the fact that $S_{g_\gamma}^j \cup R \subseteq S$ and $R_{g_\gamma}^j = S_{g_\gamma}^j \cap R$ and therefore $F(S) \geq F(S_{g_\gamma}^j \cup R) \geq F(S_{g_\gamma}^j) + F(R) - F(R_{g_\gamma}^j)$, where the first inequality follows from monotonicity and the second follows from the definition of supermodularity.

For a node v , let $S(v)$ be the set of examples associated with the leaves of the subtree rooted at v . Let v_1, v_2, \dots, v_p be a root-to-leaf path on D^* as follows: v_1 is the root of the tree, and for each $i = 1, \dots, p-1$ the node v_{i+1} is a child of v_i associated with the branch of j that maximizes $c(t_i)/(F(S) - F(S_{g_{t_i}}^j))$, where t_i is the test associated with v_i . It follows from (3.3) that

$$\frac{[F(S(v_i)) - F(S(v_{i+1}))]c(\tau)}{\lambda(F(S) - F(S_{\hat{g}_\tau}^i))} \leq c_{t_i}. \quad (3.4)$$

Since the cost of the path from v_1 to v_p is no larger than the max cost of the D^* , we

have that

$$\begin{aligned}
OPT(S) &\geq \sum_{i=1}^{p-1} c_{t_i} \\
&\geq \frac{c(\tau)}{\lambda(F(S) - F(S_{\hat{g}_\tau}^q))} \sum_{i=1}^{p-1} (F(S(v_i)) - F(S(v_{i+1}))) \\
&= \frac{c(\tau)(F(S) - F(S(v_p)))}{\lambda(F(S) - F(S_{\hat{g}_\tau}^q))} = \frac{c(\tau)F(S)}{\lambda(F(S) - F(S_{\hat{g}_\tau}^q))},
\end{aligned}$$

where the first inequality follows by the assumption that no feature is used more than once on any path of the optimal tree.

The main theorem of this section is the following.

Theorem 3.4.2 *GREEDYTREE constructs a decision tree achieving $O(\log n')$ -factor approximation of the optimal max-cost in D_F on the set S of n' examples if F is admissible and no feature is used more than once on any path of the optimal tree.*

Proof This is an inductive proof:

$$\frac{Cost_F(S)}{OPT(S)} \leq \frac{c(\tau) + Cost_F(S_{\hat{g}_\tau}^q)}{OPT(S)} \quad (3.5)$$

$$\leq \frac{c(\tau)}{OPT(S)} + \frac{Cost_F(S_{\hat{g}_\tau}^q)}{OPT(S_{\hat{g}_\tau}^q)} \quad (3.6)$$

$$\leq \lambda \frac{F(S) - F(S_{\hat{g}_\tau}^q)}{F(S)} + \frac{Cost_F(S_{\hat{g}_\tau}^q)}{OPT(S_{\hat{g}_\tau}^q)} \quad (3.7)$$

$$\leq \lambda \log\left(\frac{F(S)}{F(S_{\hat{g}_\tau}^q)}\right) + \lambda \log(F(S_{\hat{g}_\tau}^q)) + 1 \quad (3.8)$$

$$= \lambda \log(F(S)) + 1 = O(\log(n')). \quad (3.9)$$

The inequality in (3.6) follows from the fact that $OPT(S) \geq OPT(S_{\hat{g}_\tau}^q)$. (3.7) follows from Lemma 3.4.1. The first term in (3.8) follows from the inequality $\frac{x}{x+1} \leq \log(1+x)$ for $x > -1$ and the second term follows from the induction hypothesis that for each $G \subset S$, $Cost_F(G)/OPT(G) \leq \lambda \log(F(G)) + 1$. If $F(G) = 0$ for some set of examples G , we define $Cost_F(G)/OPT(G) = 1$.

We can verify the base case of the induction as follows. if $F(G) = 1$, which is the smallest non-zero impurity of F on subsets of examples S , we claim that the optimal decision tree chooses the feature with the smallest cost among those that can reduce the impurity function F :

$$OPT(G) = \min_{t|\exists g_t, \text{s.t. } F(G_{g_t}^i)=0, \forall i \in \text{outcomes}} c(t).$$

Suppose otherwise, the optimal tree chooses first a feature t with a child node G' such that $F(G') = 1$ and later chooses another feature t' such that all the child nodes of G' by $g_{t'}$ has zero impurity, then t' could have been chosen in the first place to reduce all child nodes of G to zero impurity by supermodularity of F . On the other hand, $R(t) = \infty$ in GREEDYTREE for the features that cannot reduce impurity and $R(t) = c(t)$ for those features that can. So the algorithm would pick the feature among those that can reduce impurity and have the smallest cost. Thus, we have shown that $Cost_F(G)/OPT(G) = 1 \leq \lambda \log(F(G)) + 1$ for the base case.

3.4.1 Admissible Impurity Functions

A wide range of functions falls into the class of admissible impurity functions. We employ a particular function called threshold-Pairs defined as

$$F_\alpha(G) = \sum_{i \neq j} [[n_G^i - \alpha]_+ [n_G^j - \alpha]_+ - \alpha^2]_+, \quad (3.10)$$

where n_G^i denotes the number of objects in G that belong to class i , $[x]_+ = \max(x, 0)$ and α is a threshold parameter.

Lemma 3.4.3 $F_\alpha(G)$ is admissible.

Before showing admissibility of the threshold-Pairs function in the multi-class setting, we first show $F_\alpha(G)$ is admissible for the binary setting. Consider the binary

classification setting, let

$$F_\alpha(G) = [[n_G^1 - \alpha]_+ [n_G^2 - \alpha]_+ - \alpha^2]_+.$$

All the properties are obviously true except supermodularity. To show supermodularity, suppose $R \subseteq G$ and object $j \notin R$. Suppose j belongs to the first class. We need to show

$$F_\alpha(G \cup j) - F_\alpha(G) \geq F_\alpha(R \cup j) - F_\alpha(R). \quad (3.11)$$

Consider 3 cases:

- (1) $F_\alpha(R) = F_\alpha(R \cup j) = 0$: The right hand side of (3.11) is 0 and (3.11) holds because of monotonicity of F_α .
- (2) $F_\alpha(R) = 0, F_\alpha(R \cup j) > 0, F_\alpha(G) = 0$: (3.11) reduces to $F_\alpha(G \cup j) \geq F_\alpha(R \cup j)$, which is true by monotonicity.
- (3) $F_\alpha(R) = 0, F_\alpha(R \cup j) > 0, F_\alpha(G) > 0$: Note that $F_\alpha(G) > 0$ implies that $[n_G^1 - \alpha]_+ [n_G^2 - \alpha]_+ - \alpha^2 > 0$ which further implies $n_G^1 > \alpha, n_G^2 > \alpha$. Thus the left hand side is

$$\begin{aligned} F_\alpha(G \cup j) - F_\alpha(G) &= (n_G^1 - \alpha + 1)(n_G^2 - \alpha) - \alpha^2 - ((n_G^1 - \alpha)(n_G^2 - \alpha) - \alpha^2) \\ &= n_G^2 - \alpha. \end{aligned}$$

The right hand side is

$$F_\alpha(R \cup j) = (n_R^1 - \alpha + 1)(n_R^2 - \alpha) - \alpha^2 = (n_R^1 - \alpha)(n_R^2 - \alpha) - \alpha^2 + (n_R^2 - \alpha).$$

If $n_R^1 \geq \alpha$, $F_\alpha(R) = \max((n_R^1 - \alpha)(n_R^2 - \alpha) - \alpha^2, 0) = 0$ because $F_\alpha(R \cup j) > 0$ implies $n_R^2 > \alpha$. So $F_\alpha(R \cup j) \leq n_R^2 - \alpha \leq n_G^2 - \alpha = F_\alpha(G \cup j) - F_\alpha(G)$.

- (4) $F_\alpha(R) > 0$: We have

$$F_\alpha(G \cup j) - F_\alpha(G) = n_G^2 - \alpha \geq n_R^2 - \alpha = F_\alpha(R \cup j) - F_\alpha(R).$$

This completes the proof for the binary classification setting. To generalize to the multiclass threshold-Pairs function, again, all properties are obviously true except supermodularity, which follows from the fact that each term in the sum is supermodular according to the proof for binary setting.

The following polynomial impurity function is also admissible.

Lemma 3.4.4 *Suppose there are k classes in G . Any polynomial function of n_G^1, \dots, n_G^k with non-negative terms such that n_G^1, \dots, n_G^k do not appear as singleton terms is admissible. Formally, if*

$$F(G) = \sum_{i=1}^M \gamma_i (n_G^1)^{p_{i1}} (n_G^2)^{p_{i2}} \dots (n_G^k)^{p_{ik}}, \quad (3.12)$$

where γ_i 's are non-negative, p_{ij} 's are non-negative integers and for each i there exists at least 2 non-zero p_{ij} 's, then F is admissible.

Proof Properties (1),(2),(3) and (5) are obviously true. To show F is supermodular, suppose $R \subset G$ and object $\hat{j} \notin R$ and \hat{j} belongs to class j , we have

$$\begin{aligned} & F(R \cup \hat{j}) - F(R) \\ &= \sum_{i \in I_j} \gamma_i [(n_R^1)^{p_{i1}} \dots (n_R^j + 1)^{p_{ij}} \dots (n_R^k)^{p_{ik}} - (n_R^1)^{p_{i1}} \dots (n_R^j)^{p_{ij}} \dots (n_R^k)^{p_{ik}}] \\ &\leq \sum_{i \in I_j} \gamma_i [(n_G^1)^{p_{i1}} \dots (n_G^j + 1)^{p_{ij}} \dots (n_G^k)^{p_{ik}} - (n_G^1)^{p_{i1}} \dots (n_G^j)^{p_{ij}} \dots (n_G^k)^{p_{ik}}] \\ &= F(G \cup \hat{j}) - F(G), \end{aligned}$$

where the first summation index set I_j is the set of terms that involve n_R^j . The inequality follows because $(n_R^j + 1)^{p_{ij}}$ can be expanded so the negative term can be canceled, leaving a sum-of-products form for R , which is term-by-term dominated by that of G .

Another family of *admissible* impurity functions is the Powers function.

Corollary 3.4.5 *Powers function*

$$F(G) = \left(\sum_{i=1}^k n_G^i \right)^l - \sum_{i=1}^k (n_G^i)^l \quad (3.13)$$

is admissible for $l = 2, 3, \dots$.

Neither entropy nor Gini index satisfy the notion of admissibility because they are not monotonic set functions, that is a subset of examples does not necessarily have a smaller entropy or Gini index compared to the entire set. Therefore traditional decision tree learning algorithms do not incorporate feature costs and have no guarantee on the max-cost as stated in our paper.

3.4.2 Discussions

Before concluding the BUDGETRF algorithm and its analysis, we discuss further various design issues as well as their implications.

Choice of threshold α . In subroutine GREEDYTREE, each tree is greedily built until a minimum leaf impurity is met, then added to the random forest. The threshold α can be used to trade-off between average tree depth and number of trees. A lower α results in deeper trees with higher classification power and acquisition cost. As a result, fewer trees are added to the random forest before the budget constraint is met. Conversely, a higher α yields shallower trees with poorer classification performance, however due to the low cost of each tree, many are added to the random forest before the budget constraint is met. As such, α can be viewed as a bias-variance trade-off. In practice, it is selected using validation dataset.

Minimax-splits. The splitting criterion in the subroutine GREEDYTREE is based on the worst case impurity among child nodes, we call such splits minimax-splits as opposed to expected-splits, which is based on the expected impurity among child nodes. Using minimax-splits, our theoretical guarantee is a bound on the max-cost of

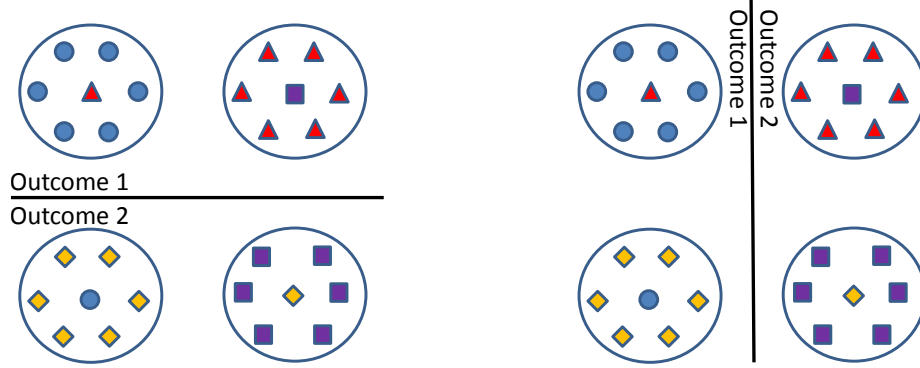


Figure 3-1: A synthetic example to show max-cost of GREEDYTREE can be “smoothed” to approach the expected-cost. The left and right figures above show the classifier outcomes of feature t_1 and t_2 , respectively.

individual trees. Note such minimax-splits have been shown to lead to expected-cost bound as well in the setting of GBS (Nowak, 2008); an interesting future research direction is to show whether minimax-splits can lead to a bound on the expected-cost of individual trees in our setting.

Smoothened Max-Costs. We emphasize that by adjusting α in threshold-Pairs function - essentially allowing some error, the max-costs of the GREEDYTREE solution can be “smoothened” so that it approaches the expected-cost. Consider the synthetic example as shown in Figure 3-1. Here we consider a multi-class classification example to demonstrate the effect of “smoothened” max-cost of the tree approaching the expected-cost. Consider a data set composed of 1024 examples belonging to 4 classes with 10 binary features available. Assume that is no two examples that have the same set of feature values. Note that by fixing the acquisition order of the features, the set of feature values maps each example to an integer in the range $[0, 1023]$. From this mapping, we give the examples in the ranges $[1, 255]$, $[257, 511]$, $[513, 767]$, and $[769, 1023]$ the labels 1, 2, 3, and 4, respectively, and the examples 0, 256, 512, and 768 the labels 2, 3, 4, and 1, respectively (Figure 3-1 shows the data projected to the first two features). Suppose each feature carries a unit cost.

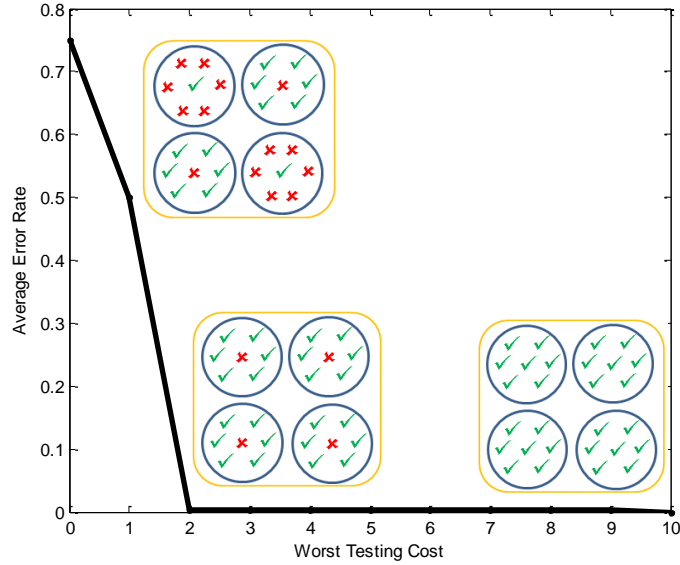


Figure 3.2: The error-cost trade-off plot of the subroutine GREEDYTREE using threshold-Pairs on the synthetic example. 0.39% error can be achieved using only a depth-2 tree but it takes a depth-10 tree to achieve zero error.

By Kraft’s Inequality (Cover and Thomas, 1991), the optimal max-cost in order to correctly classify every object is 10, however, using only t_1 and t_2 as selected by the greedy algorithm, leads to a correct classification of all but 4 objects, as shown in Figure 3.2. Thus, the max-cost of the early stopped tree is only 2 - much closer to the expected-cost.

3.5 Experiments

For establishing baseline comparisons we apply BUDGETRF on 4 real world benchmarked datasets. The first one has varying feature acquisition costs in terms of computation time and the purpose is to show our algorithm can achieve high accuracy during prediction while saving massive amount of feature acquisition time. The other 3 datasets do not have explicit feature costs; instead, we assign a unit cost to each feature uniformly. The purpose is to demonstrate our algorithm can achieve low test error using only a small fraction of features. Note our algorithm is

adaptive, meaning it acquires different features for different examples during testing. So the feature costs in the plots should be understood as an average of costs for all test examples. We use GreedyMiser (Xu et al., 2012), CSTC (Xu et al., 2013) and ASTC (Kusner et al., 2014) for comparison because they have been shown to have state-of-the-art cost-error performance. For comparison purposes we use the same configuration of training/validation/test splits as in ASTC/CSTC. The algorithm parameters for ASTC are set using the same configuration as in (Kusner et al., 2014). We report values for CSTC from (Kusner et al., 2014). We use the code provided by the authors for GreedyMiser, tuning the learning rate and loss-cost trade-off parameter λ using grid search. In all our experiments we use the threshold-Pairs (3.10) as impurity function. We use stumps as the family of classifiers \mathcal{G}_t for all features t . The optimization of classifiers in line 12 of Algorithm 2 is approximated by randomly generating 80, 40 and 20 stumps if the number of examples exceeds 2000, 500 and less than 500, respectively and select the best among them. All results from our algorithm were obtained by taking an average of 10 runs and standard deviations are reported using error bars.

Yahoo! Learning to Rank: (Chapelle et al., 2011) We evaluate BUDGETRF on a real world budgeted learning problem: Yahoo! Learning to Rank Challenge ¹. The dataset consists of 473,134 web documents and 19,944 queries. Given a set of training query-document pairs together with relevance ranks of documents for each query, the Challenge is to learn an algorithm which takes a new query and its set of associated documents and outputs the rank of these documents with respect to the new query. Each example x_i contains 519 features of a query-document pair. Each of these features is associated with an acquisition cost in the set $\{1, 5, 20, 50, 100, 150, 200\}$, which represents the units of time required for extraction and is provided by a Yahoo! employee. The labels are binarized so that $y_i = 0$ means the document is unrelated

¹<http://webscope.sandbox.yahoo.com/catalog.php?datatype=c>

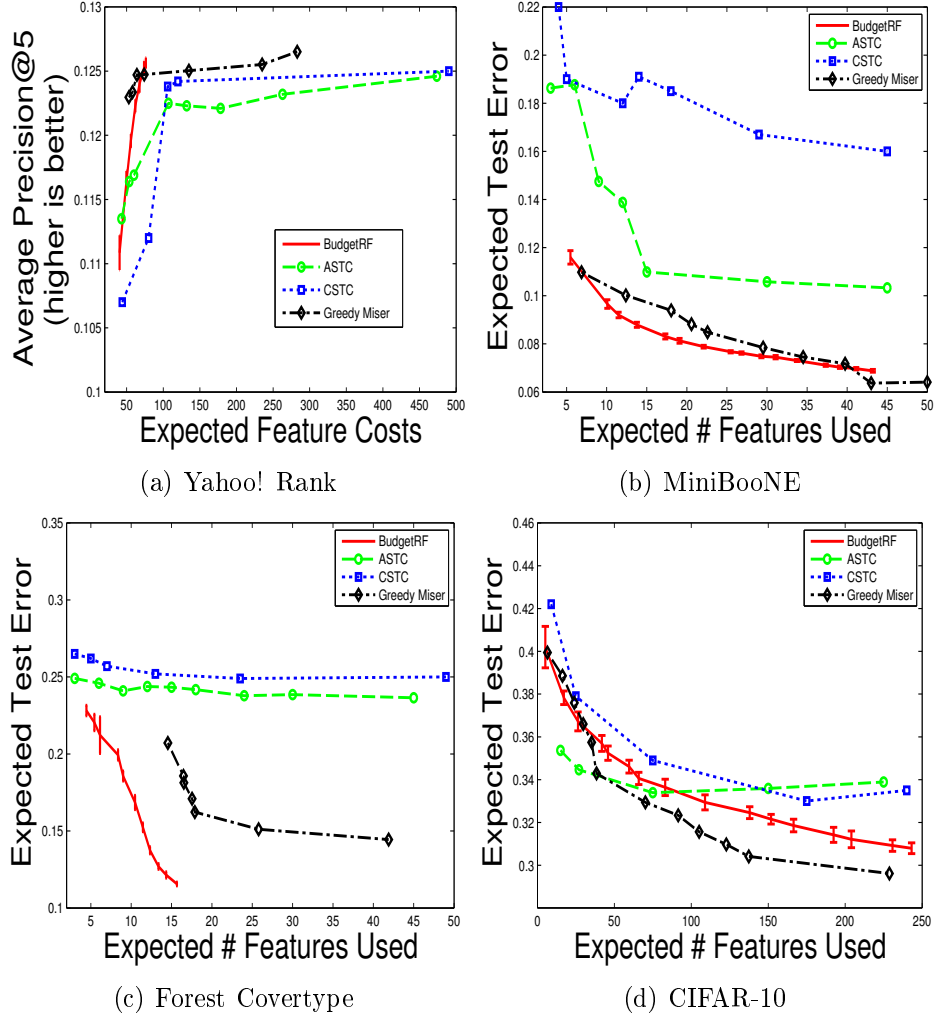


Figure 3.3: Comparison of BUDGETRF against ASTC (Kusner et al., 2014) and CSTC (Xu et al., 2013) on 4 real world datasets. BUDGETRF has a clear advantage over these state-of-the-art methods as it achieves high accuracy/low error using less feature costs.

to the query in x_i whereas $y_i = 1$ means the document is relevant to the query. There are 141,397/146,769/184,968 examples in training/validation/test sets. We use the Average Precision@5 as performance metric, same as that used in (Kusner et al., 2014). To evaluate a predicted ranking for a test query, first sort the documents in decreasing order of the predicted ranks - that is, the more relevant documents predicted by the algorithm come before those that are deemed irrelevant. Take the top 5 documents in this order and reveal their true labels. If all of the documents

are indeed relevant ($y = 1$), then the precision score is increased by 1; otherwise, if the first unrelated document appears in position $1 \leq j \leq 5$, increase the precision score by $\frac{j-1}{5}$. Finally, the precision score is averaged over the set of test queries. We run BUDGETRF using the threshold $\alpha = 0$ for the threshold-Pairs impurity function. To incorporate prediction confidence we simply run a given test example through the forest of trees to leaf nodes and aggregate the number of training examples at these leaf nodes for class 0 and 1 separately. The ratio of class 1 examples over the sum of class 1 and 0 examples gives the confidence of relevance. The comparison is shown in plot (a) of Figure 3-3. The precision for BUDGETRF rises much faster than ASTC and CSTC. At an average feature cost of 70, BUDGETRF already exceeds the precision that ASTC/CSTC can achieve using feature cost of 450 and more. GreedyMiser has an initial precision higher than BUDGETRF but rises more slowly. In this experiment the maximum number of trees we build is 140; the precision is set to rise even higher if we were to use more trees. BUDGETRF thus represents a better ranking algorithm requiring much less wait time for users of the search engine.

MiniBooNE Particle Identification Data Set: (Dheeru and Karra Taniskidou, 2017) The MiniBooNE data set is a binary classification task, with the goal of distinguishing electron neutrinos (signal) from muon neutrinos (background). Each data point consists of 50 experimental particle identification variables (features). There are 45,523/19,510/65,031 examples in training/validation/test sets. We apply BUDGETRF with a set of 10 values of $\alpha = [0, 2, 4, 6, 8, 10, 15, 25, 35, 45]$. For each α we build a forest of maximum 40 trees using BUDGETRF. Each point on the BUDGETRF curve in (b) of Figure 3-3 corresponds to a α setting and the number of trees that meet the budget level. The final α is chosen using validation set. Our algorithm clearly achieves lower test error than both ASTC and CSTC on every point of the budget level. Indeed, using just about 6 features on average out of 50, BUDGETRF

achieves lower test error than what can be achieved by ASTC or CSTC using any number of features. GreedyMiser achieves similar performance with BUDGETRF.

Forest Covertypes Data Set: (Dheeru and Karra Taniskidou, 2017) The Forest data set contains cartographic variables to predict 7 forest cover types. Each example contains 54 (10 continuous and 44 binary) features. There are 36,603/15,688/58,101 examples in training/validation/test sets. We use the same α values as in MiniBooNE. The final α is chosen using validation set. In (c) of Figure 3-3, ASTC and CSTC struggles to decrease test error even at high feature budget whereas the test error of BUDGETRF decreases rapidly as more features are acquired. GreedyMiser again does better than ASTC/CSTC but uses much more features than BUDGETRF for similar test error. We believe this dramatic performance difference is partly due to the distinct advantage of BUDGETRF in handling mixed continuous and discrete (categorical) data where the optimal decision function is highly non-linear.

CIFAR-10: (Krizhevsky, 2009) CIFAR-10 data set consists of 32x32 colour images in 10 classes. 400 features for each image are extracted using technique described in (Coates and Ng, 2011). The data are binarized by combining the first 5 classes into one class and the others into the second class. There are 19,761/8,468/10,000 examples in training/validation/test sets. As shown in (d) of Figure 3-3 BUDGETRF and GreedyMiser initially have higher test error than ASTC when the budget is low; from a budget about 90 onward BUDGETRF and GreedyMiser outperform ASTC while they outperform CSTC on the entire curve. An important trend we see is that the errors for both ASTC and CSTC start to increase after some budget level. This indicates an issue of overfitting with these methods. We do not see such an issue with BUDGETRF and GreedyMiser.

As a general comment, we observe that in low-cost regions using higher α achieves lower test error whereas setting $\alpha = 0$ leads to low test error at a higher cost. This

is consistent with our intuition that setting a high value for α terminates the tree building process early and thus saves on cost, as a consequence more trees can be built within the budget. But as budget increases, more and more trees are added to the forest, the prediction power does not grow as fast as setting α to low values because the individual trees are not as powerful.

Comments on standard Random Forest Cost is not incorporated in the standard random forest (RF) algorithm. One issue that arises is how to incorporate budget constraint. Our strategy was to limit the number of trees in the RF to control the cost. But this does not work well even if the acquisition costs are uniform for all features. We run Breiman’s RF as implemented in Matlab’s TreeBagger using default settings: fraction of input data to sample with replacement from the input data for growing each new tree is 1; number of features to select at random for each decision split is square root of the total number of features; minimum number of observations per tree leaf is 1. We report our findings in Table 3.1. Each entry in the table contains the cost as percentage of total number of features used, together with test error in parenthesis. All results are averaged over 10 runs. The test errors of the 3 methods are quite close in all 3 datasets with increasing number of trees. But BUDGETRF using threshold-Pairs impurity with $\alpha = 0$ has significantly lower costs than RF(Gini) and RF(entropy) uniformly across all datasets. For example in Forest dataset, after building 40 trees, RF(Gini) uses 76.63% of total number of features for an average test example whereas BUDGETRF uses only 29.01%. In terms of test error BUDGETRF achieves 0.1156, slightly better than 0.1196 obtained by RF(Gini). For Yahoo! Rank dataset, RF does even worse because some features have very high cost and yet RF still uses them just like the less expensive features, resulting in high cost.

	Num. Trees	1	10	20	40
MiniB	RF(Gini)	27.32(0.1278)	85.06(0.0803)	94.83(0.0730)	98.42(0.0693)
	RF(entropy)	20.67(0.1230)	75.54(0.0775)	90.35(0.0713)	97.54(0.068)
	BudgetRF	16.4(0.1241)	57.8(0.0786)	73.57(0.0721)	86.78(0.0689)
Forest	RF(Gini)	22.57(0.2107)	63.04(0.1307)	70.76(0.1238)	76.63(0.1196)
	RF(entropy)	21.68(0.2045)	63.56(0.1313)	72.37(0.1239)	79.34(0.1200)
	BudgetRF	11.37(0.2122)	23.21(0.1364)	25.92(0.1232)	29.01(0.1156)
CIFAR	RF(Gini)	3.8(0.4284)	30.35(0.3604)	49.85(0.3349)	72.20(0.3106)
	RF(entropy)	3.46(0.4267)	28.34(0.3561)	46.45(0.3296)	68.47(0.3080)
	BudgetRF	2.62(0.4264)	21.09(0.3600)	35.45(0.3332)	54.24(0.3125)

Table 3.1: Percentage of average number of features used together with test error in parenthesis for different number of trees.

Threshold-Pairs V.s. Powers functions: Finally, we compare the threshold-Pairs with various α values against the Powers function to study the effect of them on the tree building subroutine GREEDYTREE. We compare performance using 9 data sets from the UCI Repository in Figure 3-4. We assume that all features have a uniform cost. For each data set, we replace non-unique objects with a single instance using the most common label for the objects, allowing every data set to be complete (perfectly classified by the decision trees). Additionally, continuous features are transformed to discrete features by quantizing to 10 uniformly spaced levels. For trees with a smaller cost (and therefore lower depth), the threshold-Pairs impurity function outperforms the Powers impurity function with early stopping (higher α leads to earlier stopping), whereas for larger cost (and greater depth), the Powers impurity function outperforms threshold-Pairs. If α is set to 0, the difference between threshold-Pairs and Powers function is small.

Figure 3-4 shows that the Powers function does not offer significant advantage over the threshold-Pairs used in this paper.

Details of Data Sets The house votes data set is composed of the voting records for 435 members of the U.S. House of Representatives (342 unique voting records) on 16 measures, with a goal of identifying the party of each member. The sonar data set contains 208 sonar signatures, each composed of energy levels (quantized to 10 levels) in 60 different frequency bands, with a goal of identifying The ionosphere data set has 351 (350 unique) radar returns, each composed of 34 responses (quantized to 10 levels), with a goal of identifying if an event represents a free electron in the ionosphere. The Statlog DNA data set is composed of 3186 (3001 unique) DNA sequences with 180 features, with a goal of predicting whether the sequence represents a boundary of DNA to be spliced in or out. The Boston housing data set contains 13 attributes (quantized to 10 levels) pertaining to 506 (469 unique) different neighborhoods around Boston, with a goal of predicting which quartile the median income of the neighborhood the neighborhood falls. The soybean data set is composed of 307 examples (303 unique) composed of 34 categorical features, with a goal of predicting from among 19 diseases which is afflicting the soy bean plant. The pima data set is composed of 8 features (with continuous features quantized to 10 levels) corresponding to medical information and tests for 768 patients (753 unique feature patterns), with a goal of diagnosing diabetes. The Wisconsin breast cancer data set contains 30 features corresponding to properties of a cell nucleus for 569 samples, with a goal of identifying if the cell is malignant or benign. The mammography data set contains 6 features from mammography scans (with age quantized into 10 bins) for 830 patients, with a goal of classifying the lesions as malignant or benign.

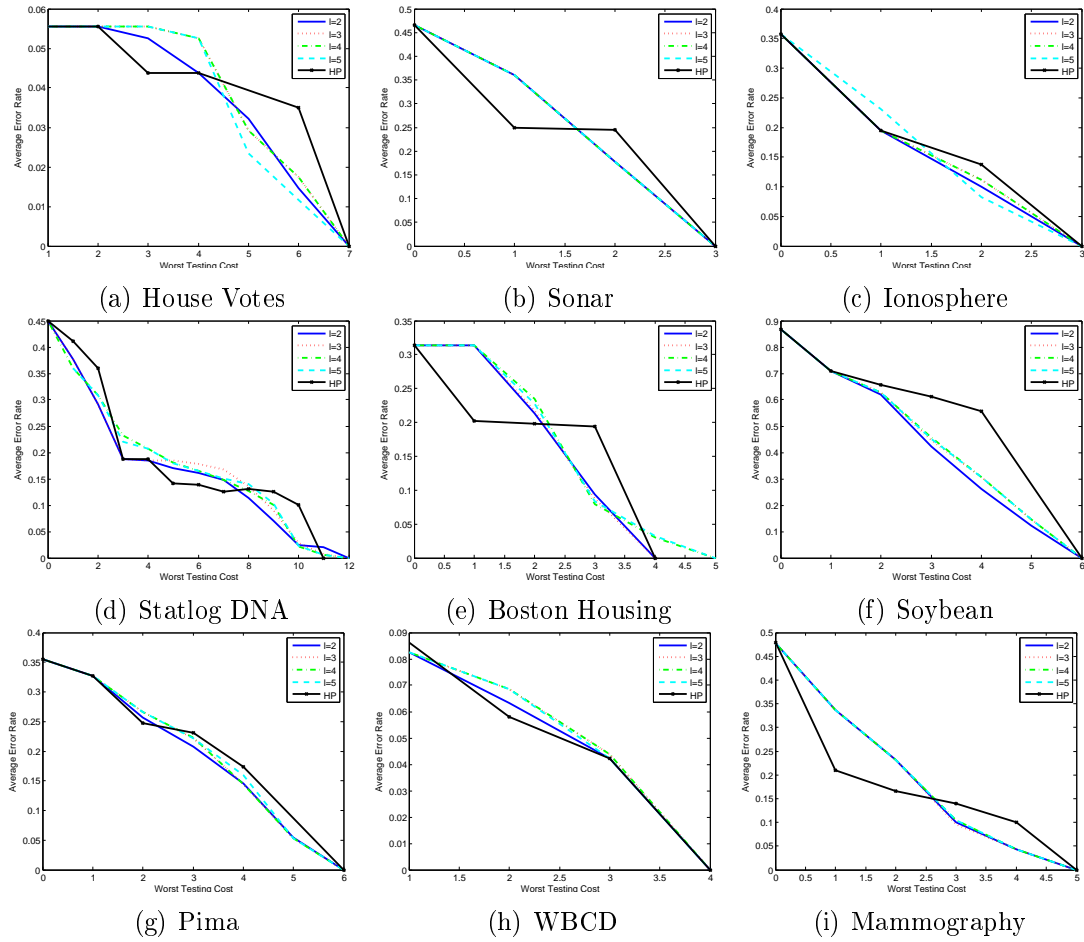


Figure 3.4: Comparison of classification error vs. max-cost for the Powers impurity function in (3.13) for $l = 2, 3, 4, 5$ and the threshold-Pairs impurity function. Note that for both House Votes and WBCD, the depth 0 tree is not included as the error decreases dramatically using a single test. In many cases, the threshold-Pairs impurity function outperforms the Powers impurity functions for trees with smaller max-costs, whereas the Powers impurity function outperforms the threshold-Pairs function for larger max-costs.

Chapter 4

Pruning Random Forests

We propose a two-stage algorithm. In the first stage, we train a random forest (RF) of trees using an impurity function such as entropy or more specialized cost-adaptive impurity (Nan et al., 2015). Our second stage takes a RF as input and attempts to jointly prune each tree in the forest to meet global resource constraints. During prediction-time, an example is routed through all the trees in the ensemble to the corresponding leaf nodes and the final prediction is based on a majority vote. The total feature cost for a test example is the sum of acquisition costs of *unique* features¹ acquired for the example in the entire ensemble of trees in the forest.²

We derive an efficient scheme to learn a *globally optimal pruning* of a RF minimizing the empirical error and incurred average costs. We formulate the pruning problem as a 0-1 integer linear program that incorporates feature-reuse constraints. By establishing total unimodularity of the constraint set, we show that solving the linear program relaxation of the integer program yields the optimal solution to the integer program resulting in a *polynomial time algorithm for optimal pruning*. We develop a primal-dual algorithm by leveraging results from network-flow theory for scaling the linear program to large datasets. Empirically, this pruning outperforms state-of-the-art resource efficient algorithms on benchmarked datasets. Our approach

¹When an example arrives at an internal node, the feature associated with the node is used to direct the example. If the feature has never been acquired for the example an acquisition cost is incurred. Otherwise, no acquisition cost is incurred as we assume that feature values are stored once computed.

²For time-sensitive cases such as web-search we parallelize the implementation by creating parallel jobs across all features and trees. We can then terminate jobs based on what features are returned.

	No Usage	1-7	> 7	Cost	Error
Unpruned RF	7.3%	91.7%	1%	42.0	6.6%
BudgetPrune	68.3%	31.5%	0.2%	24.3	6.7%

Table 4.1: Typical feature usage in a 40 tree RF before and after pruning (our algorithm) on the MiniBooNE dataset. Columns 2-4 list percentage of test examples that do not use the feature, use it 1 to 7 times, and use it greater than 7 times, respectively. Before pruning, 91% examples use the feature only a few (1 to 7) times, paying a significant cost for its acquisition; after pruning, 68% of the total examples no longer use this feature, reducing cost with minimal error increase. Column 5 is the average feature cost (the average number of unique features used by test examples). Column 6 is the test error of RFs. Overall, pruning dramatically reduces average feature cost while maintaining the same error level.

is motivated by the following considerations:

(i) RFs are scalable to large datasets and produce flexible decision boundaries yielding high prediction-time accuracy. The sequential feature usage of decision trees lends itself to adaptive feature acquisition. (ii) RF feature usage is superfluous, utilizing features with introduced randomness to increase diversity and generalization. Pruning can yield significant cost reduction with negligible performance loss by selectively pruning features sparsely used across trees, leading to cost reduction with minimal accuracy degradation (due to majority vote). See Table 4.1. (iii) Optimal pruning encourages examples to use features either a large number of times, allowing for complex decision boundaries in the space of those features, or not to use them at all, avoiding incurring the cost of acquisition. It enforces the fact that once a feature is acquired for an example, repeated use incurs no additional acquisition cost. Intuitively, features should be repeatedly used to increase discriminative ability without incurring further cost. (iv) Resource constrained prediction has been conventionally viewed as a top-down (tree-growing) approach, wherein new features are acquired based on their utility value. This is often an intractable problem with combinatorial (feature subsets) and continuous components (classifiers) requiring several relaxations

and heuristics. In contrast, ours is a bottom-up approach that starts with good initialization (RF) and prunes to realize optimal cost-accuracy tradeoff. Indeed, while we do not pursue it, our approach can also be used in conjunction with existing approaches.

The work presented in this chapter is published in (Nan et al., 2016).

4.1 Related Work

Learning decision rules to minimize error subject to a budget constraint during prediction-time is an area of recent interest, with many approaches proposed to solve the prediction-time budget constrained problem (Gao and Koller, 2011; Xu et al., 2012; Trapeznikov and Saligrama, 2013; Wang et al., 2015; Kusner et al., 2014). These approaches focus on learning complex adaptive decision functions and can be viewed as orthogonal to our work. Conceptually, these are top-down “growing” methods as we described earlier (see (iv)). Our approach is bottom-up that seeks to prune complex classifiers to tradeoff cost vs. accuracy.

Our work is based on RF classifiers (Breiman, 2001). Traditionally, feature cost is not incorporated when constructing RFs, however we have shown in Chapter 3 an approximation of budget constraints to learn budgeted RFs (Nan et al., 2015). The tree-growing algorithm in Chapter 3 does not take feature re-use into account. Rather than attempting to approximate the budget constraint during tree construction, our work focuses on pruning ensembles of trees subject to a budget constraint. Methods such as traditional ensemble learning and budgeted random forests can be viewed as complementary.

Decision tree pruning has been studied extensively to improve generalization performance; we are not aware of any existing pruning method that takes into account the feature costs. A popular method for pruning to reduce generalization error is

Cost-Complexity Pruning (CCP), introduced by Breiman et al. (Breiman et al., 1984). CCP trades-off classification ability for tree size, however it does not account for feature costs. As pointed out by Li et al. (Li et al., 2001), CCP has undesirable “jumps” in the sequence of pruned tree sizes. To alleviate this, they proposed a Dynamic-Program-based Pruning (DPP) method for binary trees. The DPP algorithm is able to obtain optimally pruned trees of all sizes; however, it faces the curse of dimensionality when pruning an ensemble of decision trees and taking feature cost into account. (Zhang and Huei-chuen, 2005; Sherali et al., 2009) proposed to solve the pruning problem as a 0-1 integer program; again, their formulations do not account for feature costs that we focus on in this chapter. The coupling nature of feature usage makes our problem much harder. In general pruning RFs is not a focus of attention as it is assumed that overfitting can be avoided by constructing an ensemble of trees. While this is true, it often leads to extremely large prediction-time costs. Kulkarni and Sinha (Kulkarni and Sinha, 2012) provide a survey of methods to prune RFs in order to reduce ensemble size. However, these methods do not explicitly account for feature costs.

4.2 Problem Setup

In this chapter, we consider solving the Lagrangian relaxed problem of learning under prediction-time resource constraints, also known as the error-cost tradeoff problem:

$$\min_{f \in \mathcal{F}} E_{(x,y) \sim \mathcal{P}} [\text{err}(y, f(x))] + \lambda E_{x \sim \mathcal{P}_x} [C(f, x)], \quad (4.1)$$

where example/label pairs (x, y) are drawn from a distribution \mathcal{P} ; $\text{err}(y, \hat{y})$ is the error function; $C(f, x)$ is the cost of evaluating the classifier f on example x ; λ is a tradeoff parameter. A larger λ places a larger penalty on cost, pushing the classifier to have smaller cost. By adjusting λ we can obtain a classifier satisfying the budget

constraint. The family of classifiers \mathcal{F} in our setting is the space of RFs, and each RF f is composed of T decision trees $\mathcal{T}_1, \dots, \mathcal{T}_T$.

Our approach: Rather than attempting to construct the optimal ensemble by solving Eqn. (4.1) directly, we instead propose a two-step algorithm that first constructs an ensemble with low prediction error, then prunes it by solving Eqn. (4.1) to produce a pruned ensemble given the input ensemble. By adopting this two-step strategy, we obtain an ensemble with low expected cost while simultaneously preserving the low prediction error.

There are many existing methods to construct RFs, however the focus of this chapter is on the second step, where we propose a novel approach to prune RFs to solve the tradeoff problem Eqn.(4.1). Our pruning algorithm is capable of taking any RF as input, offering the flexibility to incorporate any state-of-the-art RF algorithm.

4.2.1 Pruning with Costs

In this section, we treat the error-cost tradeoff problem Eqn. (4.1) as an RF pruning problem. Our key contribution is to formulate pruning as a 0-1 integer program with totally unimodular constraints.

We first define notations used throughout the chapter. A training sample $S = \{(\mathbf{x}^{(i)}, y^{(i)}) : i = 1, \dots, N\}$ is generated i.i.d. from an unknown distribution, where $\mathbf{x}^{(i)} \in \mathbb{R}^K$ is the feature vector with a cost assigned to each of the K features and $y^{(i)}$ is the label for the i^{th} example. In the case of multi-class classification $y \in \{1, \dots, M\}$, where M is the number of classes. Given a decision tree \mathcal{T} , we index the nodes as $h \in \{1, \dots, |\mathcal{T}|\}$, where node 1 represents the root node. Let $\tilde{\mathcal{T}}$ denote the set of leaf nodes of tree \mathcal{T} . Finally, the corresponding definitions for \mathcal{T} can be extended to an ensemble of T decision trees $\{\mathcal{T}_t : t = 1, \dots, T\}$ by adding a subscript t .

Pruning Parametrization: In order to model ensemble pruning as an optimization problem, we parametrize the space of all prunings of an ensemble. The process

of pruning a decision tree \mathcal{T} at an internal node h involves collapsing the subtree of \mathcal{T} rooted at h , making h a leaf node. We say a pruned tree $\mathcal{T}^{(p)}$ is a valid pruned tree of \mathcal{T} if (1) $\mathcal{T}^{(p)}$ is a subtree of \mathcal{T} containing root node 1 and (2) for any $h \neq 1$ contained in $\mathcal{T}^{(p)}$, the sibling nodes (the set of nodes that share the same immediate parent node as h in \mathcal{T}) must also be contained in $\mathcal{T}^{(p)}$. Specifying a pruning is equivalent to specifying the nodes that are leaves in the pruned tree. We therefore introduce the following binary variable for each node $h \in \mathcal{T}$

$$z_h = \begin{cases} 1 & \text{if node } h \text{ is a leaf in the pruned tree,} \\ 0 & \text{otherwise.} \end{cases}$$

We call the set $\{z_h, \forall h \in \mathcal{T}\}$ the node variables as they are associated with each node in the tree. Consider any root-to-leaf path in a tree \mathcal{T} , there should be exactly one node in the path that is a leaf node in the pruned tree. Let $p(h)$ denote the set of predecessor nodes, the set of nodes (including h) that lie on the path from the root node to h . The set of valid pruned trees can be represented as the set of node variables satisfying the following set of constraints: $\sum_{u \in p(h)} z_u = 1 \quad \forall h \in \tilde{\mathcal{T}}$. Given a valid pruning for a tree, we now seek to parameterize the error of the pruning.

Pruning error: As in most supervised empirical risk minimization problems, we aim to minimize the error on training data as a surrogate to minimizing the expected error. In a decision tree \mathcal{T} , each node h is associated with a predicted label corresponding to the majority label among the training examples that fall into the node h . Let S_h denote the subset of examples in S routed to or through node h on \mathcal{T} and let Pred_h denote the predicted label at h . The number of misclassified examples at h is therefore $e_h = \sum_{i \in S_h} \mathbb{1}_{[y^{(i)} \neq \text{Pred}_h]}$. We can thus estimate the error of tree \mathcal{T} in terms of the number of misclassified examples in the leaf nodes: $\frac{1}{N} \sum_{h \in \tilde{\mathcal{T}}} e_h$, where $N = |S|$ is the total number of examples.

Our goal is to minimize the expected test error of the trees in the random forest,

which we empirically approximate based on the aggregated probability distribution in Step (6) of Algorithm 3 with $\frac{1}{TN} \sum_{t=1}^T \sum_{h \in \tilde{\mathcal{T}}_t} e_h$. We can express this error in terms of the node variables: $\frac{1}{TN} \sum_{t=1}^T \sum_{h \in \mathcal{T}_t} e_h z_h$.

Pruning cost: Assume the acquisition costs for the K features, $\{c_k : k = 1, \dots, K\}$, are given. The feature acquisition cost incurred by an example is the sum of the acquisition costs of unique features acquired in the process of running the example through the forest. This cost structure arises due to the assumption that an acquired feature is cached and subsequent usage by the same example incurs no additional cost. Formally, the feature cost of classifying an example i on the ensemble $\mathcal{T}_{[T]}$ is given by $C_{\text{feature}}(\mathcal{T}_{[T]}, \mathbf{x}^{(i)}) = \sum_{k=1}^K c_k w_{k,i}$, where the binary variables $w_{k,i}$ serve as the indicators:

$$w_{k,i} = \begin{cases} 1 & \text{if feature } k \text{ is used by } \mathbf{x}^{(i)} \text{ in any } \mathcal{T}_t, t = 1, \dots, T \\ 0 & \text{otherwise.} \end{cases}$$

The expected feature cost of a test example can be approximated as

$$\frac{1}{N} \sum_{i=1}^N \sum_{k=1}^K c_k w_{k,i}.$$

In some scenarios, it is useful to account for computation cost along with feature acquisition cost during prediction-time. In an ensemble, this corresponds to the expected number of Boolean operations required running a test through the trees, which is equal to the expected depth of the trees. This can be modeled as $\frac{1}{N} \sum_{t=1}^T \sum_{h \in \mathcal{T}_t} |S_h| d_h z_h$, where d_h is the depth of node h .

Putting it together: Having modeled the pruning constraints, prediction performance and costs, we formulate the problem of pruning using the relationship between the node variables z_h 's and feature usage variables $w_{k,i}$'s. Given a tree \mathcal{T} , feature k , and example $\mathbf{x}^{(i)}$, let $u_{k,i}$ be the first node associated with feature k on the root-to-leaf path the example follows in \mathcal{T} . Feature k is used by $\mathbf{x}^{(i)}$ if and only if none of

the nodes between the root and $u_{k,i}$ is a leaf. We represent this by the constraint $w_{k,i} + \sum_{h \in p(u_{k,i})} z_h = 1$ for every feature k used by example $x^{(i)}$ in \mathcal{T} . Recall $w_{k,i}$ indicates whether or not feature k is used by example i and $p(u_{k,i})$ denotes the set of predecessor nodes of $u_{k,i}$. Intuitively, this constraint says that either the tree is pruned along the path followed by example i before feature k is acquired, in which case $z_h = 1$ for some node $h \in p(u_{k,i})$ and $w_{k,i} = 0$; or $w_{k,i} = 1$, indicating that feature k is acquired for example i . We extend the notations to ensemble pruning with tree index t : $z_h^{(t)}$ indicates whether node h in \mathcal{T}_t is a leaf after pruning; $w_{k,i}^{(t)}$ indicates whether feature k is used by the i^{th} example in \mathcal{T}_t ; $w_{k,i}$ indicates whether feature k is used by the i^{th} example in any of the T trees $\mathcal{T}_1, \dots, \mathcal{T}_T$; $u_{t,k,i}$ is the first node associated with feature k on the root-to-leaf path the example follows in \mathcal{T}_t ; $K_{t,i}$ denotes the set of features the i^{th} example uses on tree \mathcal{T}_t . We arrive at the following integer program.

$$\begin{aligned}
\min_{\substack{z_h^{(t)}, w_{k,i}^{(t)}, \\ w_{k,i} \in \{0,1\}}} & \quad \overbrace{\frac{1}{NT} \sum_{t=1}^T \sum_{h \in \mathcal{T}_t} e_h^{(t)} z_h^{(t)}}^{\text{error}} + \lambda \left(\overbrace{\sum_{k=1}^K c_k \left(\frac{1}{N} \sum_{i=1}^N w_{k,i} \right)}^{\text{feature acquisition cost}} + \overbrace{\frac{1}{N} \sum_{t=1}^T \sum_{h \in \mathcal{T}_t} |S_h| d_h z_h}_{\text{computational cost}} \right) \quad (\text{IP}) \\
\text{s.t.} & \quad \sum_{u \in p(h)} z_u^{(t)} = 1, \quad \forall h \in \tilde{\mathcal{T}}_t, \forall t \in [T], \quad (\text{feasible prunings}) \\
& \quad w_{k,i}^{(t)} + \sum_{h \in p(u_{t,k,i})} z_h^{(t)} = 1, \quad \forall k \in K_{t,i}, \forall i \in S, \forall t \in [T], \quad (\text{feature usage/ tree}) \\
& \quad w_{k,i}^{(t)} \leq w_{k,i}, \quad \forall k \in [K], \forall i \in S, \forall t \in [T]. \quad (\text{global feature usage})
\end{aligned}$$

4.3 Theoretical Analysis

Even though integer programs are NP-hard to solve in general, we show that (IP) can be solved exactly by solving its LP relaxation. We prove this in two steps: first, we examine the special structure of the equality constraints; then we examine the inequality constraint that couples the trees. Recall that a network matrix is one with each column having exactly one element equal to 1, one element equal to -1 and the remaining elements being 0. A network matrix defines a directed graph with the

nodes in the rows and arcs in the columns. We have the following lemma.

Lemma 4.3.1 *The equality constraints in (IP) can be turned into an equivalent network matrix form for each tree.*

Proof We observe the first constraint $\sum_{u \in p(h)} z_u^{(t)} = 1$ requires the sum of the node variables along a path to be 1. The second constraints $w_{k,i}^{(t)} + \sum_{h \in p(u_{t,k,i})} z_h^{(t)} = 1$ has a similar sum except the variable $w_{k,i}^{(t)}$. Imagine $w_{k,i}^{(t)}$ as yet another node variable for a fictitious child node of $u_{t,k,i}$ and the two equations are essentially equivalent. The rest of proof follows directly from the construction in Proposition 3 of (Sherali et al., 2009).

Figure 4.1 illustrates such a construction. There are two decision trees with the nodes numbered 1 to 12. The subscript at each node number is the feature index used in the node. For simplicity we consider only one example being routed to nodes 4 and 11 respectively on the two trees. The equality constraints in (IP) can be separated based on the trees and put in matrix form:

$$\begin{matrix} & z_1 & z_2 & z_3 & z_4 & z_5 & w_{1,1}^{(1)} & w_{2,1}^{(1)} \\ \begin{matrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \end{matrix} & \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{matrix},$$

for tree 1 and

$$\begin{matrix} & z_6 & z_7 & z_8 & z_9 & z_{10} & z_{11} & z_{12} & w_{2,1}^{(2)} & w_{3,1}^{(2)} \\ \begin{matrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \end{matrix} & \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{matrix},$$

for tree 2. Through row operations they can be turned into network matrices, where

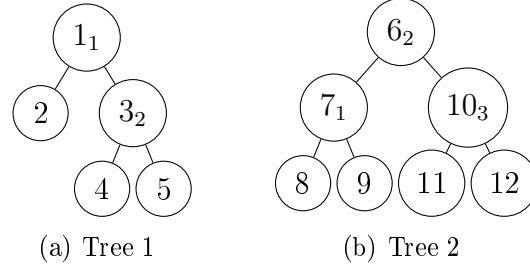


Figure 4.1: An ensemble of two decision trees with node numbers and associated feature in subscripts

there is exactly two non-zeros in each column, a 1 and a -1 .

$$\begin{array}{c}
 -r_1 \\
 r_1 - r_2 \\
 r_2 - r_3 \\
 r_3 - r_4 \\
 r_4 - r_5 \\
 r_5
 \end{array}
 \begin{pmatrix}
 z_1 & z_2 & z_3 & z_4 & z_5 & w_{1,1}^{(1)} & w_{2,1}^{(1)} \\
 -1 & -1 & 0 & 0 & 0 & 0 & 0 \\
 0 & 1 & -1 & -1 & 0 & 0 & 0 \\
 0 & 0 & 0 & 1 & -1 & 0 & 0 \\
 0 & 0 & 0 & 0 & 1 & 0 & -1 \\
 0 & 0 & 1 & 0 & 0 & -1 & 1 \\
 1 & 0 & 0 & 0 & 0 & 1 & 0
 \end{pmatrix},$$

for tree 1 and

$$\begin{array}{c}
 -r_1 \\
 r_1 - r_2 \\
 r_2 - r_3 \\
 r_3 - r_4 \\
 r_4 - r_5 \\
 r_5 - r_6 \\
 r_6
 \end{array}
 \begin{pmatrix}
 z_6 & z_7 & z_8 & z_9 & z_{10} & z_{11} & z_{12} & w_{2,1}^{(2)} & w_{3,1}^{(2)} \\
 -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\
 0 & 1 & 0 & 1 & -1 & -1 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\
 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 1 \\
 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
 \end{pmatrix}$$

for tree 2. Note the above transformation to network matrices can always be done as long as the leaf nodes are arranged in a pre-order fashion in binary tree traversal.

Next, we deal with the inequality constraints and obtain our main result.

Theorem 4.3.2 *The LP relaxation of (IP), where the 0-1 integer constraints are relaxed to interval constraints $[0, 1]$ for all integer variables, has integral optimal solutions.*

Proof The main idea is to show the constraints are still *totally unimodular* even after

adding the coupling constraints and the LP relaxed polyhedron has only integral extreme points (Nemhauser et al., 1978). Denote the equality constraints of (IP) with index set J_1 . They can be divided into each tree. Each constraint matrix in J_1 associated with a tree can be turned into a network matrix according to Lemma 4.3.1. Stacking these matrices leads to a larger network matrix. Denote the $w_{k,i}^{(t)} \leq w_{k,i}$ constraints with index set J_2 . Consider the constraint matrix for J_2 . Each $w_{k,i}^{(t)}$ only appears once in J_2 , which means the column corresponding to $w_{k,i}^{(t)}$ has only one element equal to 1 and the rest equal to 0. If we arrange the constraints in J_2 such that for any given k, i $w_{k,i}^{(t)} \leq w_{k,i}$ are put together for $t \in [T]$, the constraint matrix for J_2 has interval structure such that the non-zeros in each column appear consecutively. Finally, putting the network matrix from J_1 and the matrix from J_2 together. Assign J_1 and the odd rows of J_2 to the first partition Q_1 and assign the even rows of J_2 to the second partition Q_2 . Note the upper bound constraints on the variables can be ignored as this is an minimization problem. We conclude that the constraint matrix of (IP) is totally unimodular according to Theorem 2.7, Part 3 of (Nemhauser and Wolsey, 1988) with partition Q_1 and Q_2 . By Proposition 2.1 and 2.2, Part 3 of (Nemhauser and Wolsey, 1988) we can conclude the proof.

As a result, solving the LP relaxation results in the optimal solution to the integer program **(IP)**, allowing for polynomial time optimization. The nice result of totally unimodular constraints is due to our specific formulation. We illustrate an alternative formulation that does not have such a property.

4.3.1 A Naive Pruning Formulation

The nice property of totally unimodular constraints in Theorem 4.3.2 is due to our specific formulation. Here we present an alternative integer program formulation and

show its deficiency. Recall we defined the following node variables

$$z_h = \begin{cases} 1 & \text{if node } h \text{ is a leaf in the pruned tree,} \\ 0 & \text{otherwise.} \end{cases}$$

and indicator variables of feature usage:

$$w_{k,i} = \begin{cases} 1 & \text{if feature } k \text{ is used by } \mathbf{x}^{(i)} \text{ in any } \mathcal{T}_t, t = 1, \dots, T \\ 0 & \text{otherwise.} \end{cases}$$

First, note that if $z_h = 1$ for some node h , then the examples that are routed to h must have used all the features in the predecessor nodes $p(h)$, excluding h . We use $k \sim p(h)$ to denote feature k is used in any predecessor of h , excluding h . Then for each feature k and example i , we must have $w_{k,i} \geq z_h$ for all nodes h such that $i \in S_h$ and $k \sim p(h)$. Combining these constraints with the pruning constraints we formulate pruning as a 0-1 integer program for an individual tree:

$$\begin{aligned} \min_{\substack{z_h \in \{0,1\} \\ w_{k,i} \in \{0,1\}}} \quad & \frac{1}{N} \sum_{h \in \mathcal{N}} e_h z_h + \lambda \sum_{k=1}^K c_k \left(\frac{1}{N} \sum_{i=1}^N w_{k,i} \right) \\ \text{s.t.} \quad & z_h + \sum_{u \in p(h)} z_u = 1 \quad \forall h \in \tilde{\mathcal{T}}, \\ & w_{k,i} \geq z_h \quad \forall h : i \in S_h \wedge k \sim p(h), \\ & \quad \quad \quad \forall k \in [K], \forall i \in S. \end{aligned}$$

To solve the integer program, a common heuristic is to solve its linear program relaxation. Unfortunately, the constraint set in the above formulation has fractional extreme points, leading to possibly fractional solutions to the relaxed problem. It is not clear how to perform rounding to obtain good prunings. Consider the first tree in Figure 4-1. Feature 1 is used at the root node and feature 2 is used at node 3. There are 7 variables (assuming there is only one example and it goes to leaf 4):

$z_1, z_2, z_3, z_4, z_5, w_{1,1}, w_{2,1}$. The LP relaxed constraints are:

$$z_1 + z_3 + z_4 = 1, z_1 + z_3 + z_5 = 1, z_1 + z_2 = 1,$$

$$w_{1,1} \geq z_4, w_{1,1} \geq z_3, w_{2,1} \geq z_4, 0 \leq z \leq 1.$$

The following is a basic feasible solution:

$$z_1 = 0, z_2 = 1, z_3 = z_4 = z_5 = 0.5, w_{1,1} = w_{2,1} = 0.5,$$

because the following set of 7 constraints are active:

$$z_1 + z_3 + z_4 = 1, z_1 + z_3 + z_5 = 1,$$

$$w_{1,1} \geq z_4, w_{1,1} \geq z_3, w_{2,1} \geq z_4, z_1 = 0, z_2 = 1.$$

Even if we were to interpret the fractional solution of z_h as probabilities of h being a leaf node, we see an issue with this formulation: the example has 0.5 probability of stopping at node 3 or 4 ($z_3 = z_4 = 0.5$). In both cases, feature 1 at the root node has to be used, however $w_{1,1} = 0.5$ indicates that it is only being used half of the times. This solution is not a feasible pruning and fails to capture the cost of the pruning.

Attempting to use an LP relaxation of this formulation fails to capture the desired behavior of the integer program.

4.4 Algorithm

Even though we can solve **(IP)** via its LP relaxation, the resulting LP can be too large in practical applications for any general-purpose LP solver. In particular, the number of variables and constraints is roughly $O(T \times |\mathcal{T}_{\max}| + N \times T \times K_{\max})$, where T is the number of trees; $|\mathcal{T}_{\max}|$ is the maximum number of nodes in a tree; N is the number of examples; K_{\max} is the maximum number of features an example uses

in a tree. The runtime of the LP thus scales $O(T^3)$ with the number of trees in the ensemble, limiting the application to only small ensembles. In this section we propose a primal-dual approach that effectively decomposes the optimization into many sub-problems. Each sub-problem corresponds to a tree in the ensemble and can be solved efficiently as a shortest path problem. The runtime per iteration is $O(\frac{T}{p}(|\mathcal{T}_{\max}| + N \times K_{\max}) \log(|\mathcal{T}_{\max}| + N \times K_{\max}))$, where p is the number of processors. We can thus massively parallelize the optimization and scale to much larger ensembles as the runtime depends only linearly on $\frac{T}{p}$. To this end, we assign dual variables $\beta_{k,i}^{(t)}$ for the inequality constraints $w_{k,i}^{(t)} \leq w_{k,i}$ and derive the dual problem.

$$\begin{aligned}
\max_{\substack{\beta_{k,i}^{(t)} \geq 0 \\ z_h^{(t)} \in [0,1] \\ w_{k,i}^{(t)} \in [0,1] \\ w_{k,i} \in [0,1]}} \min_{\substack{z_h^{(t)} \\ w_{k,i}^{(t)} \\ w_{k,i}}} & \frac{1}{NT} \sum_{t=1}^T \sum_{h \in \mathcal{T}_t} \hat{e}_h^{(t)} z_h^{(t)} + \lambda \left(\sum_{k=1}^K c_k \left(\frac{1}{N} \sum_{i=1}^N w_{k,i} \right) \right) + \sum_{t=1}^T \sum_{i=1}^N \sum_{k \in K_{t,i}} \beta_{k,i}^{(t)} (w_{k,i}^{(t)} - w_{k,i}) \\
\text{s.t.} & \sum_{u \in p(h)} z_u^{(t)} = 1, \quad \forall h \in \tilde{\mathcal{T}}_t, \forall t \in [T], \\
& w_{k,i}^{(t)} + \sum_{h \in p(u_{t,k,i})} z_h^{(t)} = 1, \quad \forall k \in K_{t,i}, \forall i \in S, \forall t \in [T],
\end{aligned}$$

where for simplicity we have combined coefficients of $z_h^{(t)}$ in the objective of **(IP)** to $\hat{e}_h^{(t)}$. The primal-dual algorithm is summarized in Algorithm 3. It alternates between updating the primal and the dual variables. The key is to observe that given dual variables, the primal problem (inner minimization) can be decomposed for each tree in the ensemble and solved in parallel as shortest path problems due to Lemma 4.3.1. Figure 4-2 illustrates such a construction based on the network matrices shown above. The nodes in the graphs correspond to rows in the network matrices and the arcs correspond to the columns, which are the primal variables $z_h, w_{k,i}^{(t)}$'s. There is a cost associated with each arc in the objective of the minimization problem. The task is to find a path from the first node (source) to the last node (sink) such that the sum of arc costs is minimized. Note each path from source to sink corresponds to a feasible pruning. For example, in (a) of Figure 4-2, consider the path of 1-2-5-6, the

Algorithm 3 BUDGETPRUNE

During Training: input - ensemble($\mathcal{T}_1, \dots, \mathcal{T}_T$), training/validation data with labels, λ

- 1: initialize dual variables $\beta_{k,i}^{(t)} \leftarrow 0$.
 - 2: update $z_h^{(t)}, w_{k,i}^{(t)}$ for each tree t (shortest-path algo). $w_{k,i} = 0$ if $\mu_{k,i} > 0$, $w_{k,i} = 1$ if $\mu_{k,i} < 0$.
 - 3: $\beta_{k,i}^{(t)} \leftarrow [\beta_{k,i}^{(t)} + \gamma(w_{k,i}^{(t)} - w_{k,i})]_+$ for step size γ , where $[\cdot]_+ = \max\{0, \cdot\}$.
 - 4: go to Step 2 until duality gap is small enough.
-

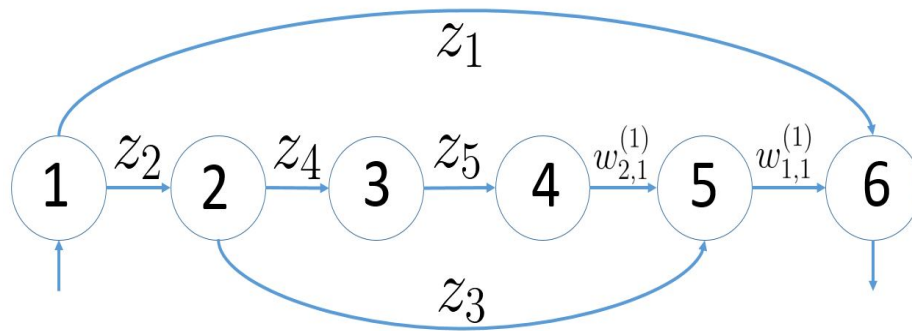
During Prediction: input - test example \mathbf{x}

- 5: Run \mathbf{x} on each tree to leaf, obtain the probability distribution over label classes \mathbf{p}_t at leaf.
 - 6: Aggregate $\mathbf{p} = \frac{1}{T} \sum_{t=1}^T \mathbf{p}_t$. Predict the class with the highest probability in \mathbf{p} .
-

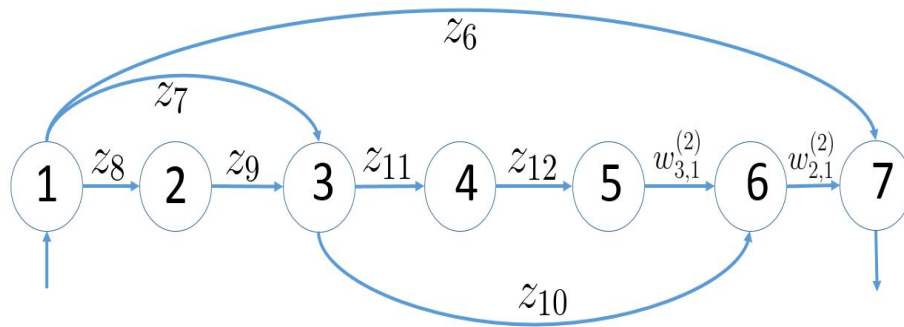
active arcs are z_2, z_3 and $w_{1,1}^{(1)}$. Setting these variables to 1 and others to 0, we see that it corresponds to pruning Tree 1 at node 3 in Figure 4-1. (Note the nodes in Figure 4-2 and Figure 4-1 are not to be confused - they do not have a relation with each other.)

The primal variables $w_{k,i}$ can be solved in closed form: simply compute $\mu_{k,i} = \lambda c_k / N - \sum_{t \in T_{k,i}} \beta_{k,i}^{(t)}$, where $T_{k,i}$ is the set of trees in which example i encounters feature k . So $w_{k,i}$ should be set to 0 if $\mu_{k,i} > 0$ and $w_{k,i} = 1$ if $\mu_{k,i} < 0$.

Note that our prediction rule aggregates the leaf distributions from all trees instead of just their predicted labels. In the case where the leaves are pure (each leaf contains only one class of examples), this prediction rule coincides with the majority vote rule commonly used in random forests. Whenever the leaves contain mixed classes, this rule takes into account the prediction confidence of each tree in contrast to majority voting. Empirically, this rule consistently gives lower prediction error than majority voting with pruned trees.



(a) Tree 1



(b) Tree 2

Figure 4.2: Turning pruning to equivalent shortest path problems.

4.5 Experiments

4.5.1 Baseline Comparison

We test our pruning algorithm BUDGETPRUNE on four benchmark datasets used for prediction-time budget algorithms. The first two datasets have unknown feature acquisition costs so we assign costs to be 1 for all features; the aim is to show that BUDGETPRUNE successfully selects a sparse subset of features on average to classify each example with high accuracy.³ The last two datasets have real feature acquisition costs measured in terms of CPU time. BUDGETPRUNE achieves high prediction accuracy spending much less CPU time in feature acquisition.

For each dataset we first train a RF and apply BUDGETPRUNE on it using different λ 's to obtain various points on the accuracy-cost tradeoff curve. We use in-bag data to estimate error probability at each node and the validation data for the feature cost variables $w_{k,i}$'s. We implement BUDGETPRUNE using CPLEX (cpl, 2010) network flow solver for the primal update step. The running time is significantly reduced (from hours down to minutes) compared to directly solving the LP relaxation of (IP) using standard solvers such as Gurobi (Gurobi Optimization, 2015). Furthermore, the standard solvers simply break trying to solve the larger experiments whereas BUDGETPRUNE handles them with ease. We run the experiments for 10 times and report the means and standard deviations.

Competing methods: We compare against four other approaches. (i) BUDGETRF(Nan et al., 2015): the recursive node splitting process for each tree is stopped as soon as node impurity (entropy or Pairs) falls below a threshold. The threshold is a measure of impurity tolerated in the leaf nodes. This can be considered as a naive pruning method as it reduces feature acquisition cost while maintaining low

³In contrast to traditional sparse feature selection, our algorithm allows adaptivity, meaning different examples use different subsets of features.

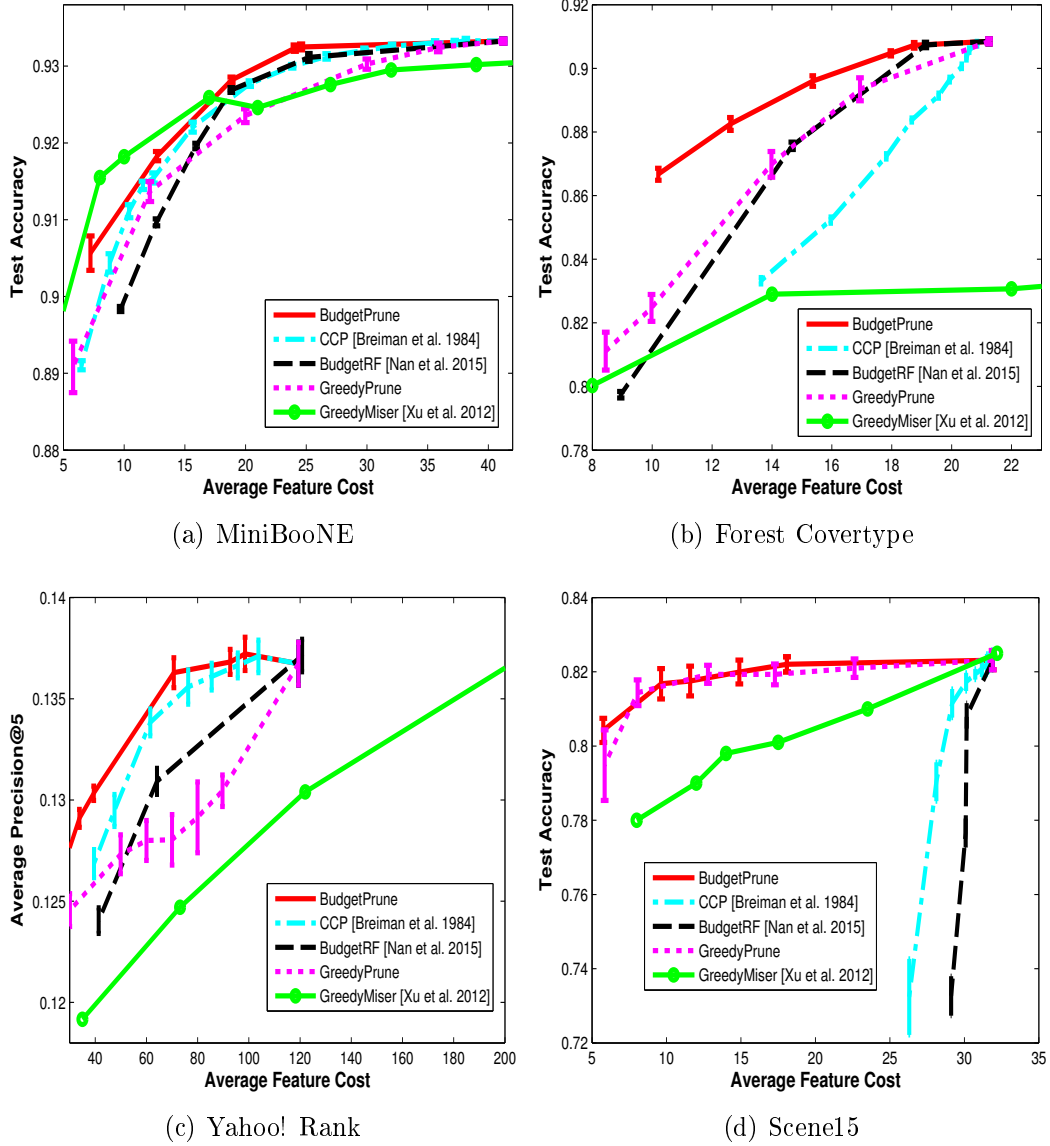


Figure 4-3: Comparison of BUDGETPRUNE against CCP, BUDGETRF with early stopping, GREEDYPRUNE and GREEDYMISER on 4 real world datasets. BUDGETPRUNE (red) outperforms competing state-of-art methods. GREEDYMISER dominates ASTC (Kusner et al., 2014), CSTC (Xu et al., 2013) and DAG (Wang et al., 2015) significantly on all datasets. We omit them in the plots to clearly depict the differences between competing methods.

impurity in the leaves. (ii) Cost-Complexity Pruning (CCP) (Breiman et al., 1984): it iteratively prunes subtrees such that the resulting tree has low error and small

size. We perform CCP on individual trees to different levels to obtain various points on the accuracy-cost tradeoff curve. CCP does not take into account feature costs.

(iii) GREEDYPRUNE: is a greedy global feature pruning strategy that we propose; at each iteration it attempts to remove all nodes corresponding to one feature from the RF such that the resulting pruned RF has the lowest training error and average feature cost. The process terminates in at most K iterations, where K is the number of features. The idea is to reduce feature costs by successively removing features that result in large cost reduction yet small accuracy loss. We also compare against the state-of-the-art methods in budgeted learning (iv) GREEDYMISER (Xu et al., 2012): it is a modification of gradient boosted regression tree (Friedman, 2000) to incorporate feature cost. Specifically, each weak learner (a low-depth decision tree) is built to minimize squared loss with respect to current gradient at the training examples plus feature acquisition cost. To build each weak learner the feature costs are set to zero for those features already used in previous weak learners. Other prediction-time budget algorithms such as ASTC (Kusner et al., 2014), CSTC (Xu et al., 2013) and cost-weighted l -1 classifiers are shown to perform strictly worse than GREEDYMISER by a significant amount (Kusner et al., 2014; Nan et al., 2015) so we omit them in our plots. Since only the feature acquisition costs are standardized, for fair comparison we do not include the computation cost term in the objective of (IP) and focus instead on feature acquisition costs.

MiniBooNE Particle Identification and Forest Covertypes Datasets

(Dheeru and Karra Taniskidou, 2017): The MiniBooNE data set is a binary classification task to distinguish electron neutrinos from muon neutrinos. There are 45523/19510 /65031 examples in training/validation/test sets. Each example has 50 features, each with unit cost. The Forest data set contains cartographic variables to predict 7 forest cover types. There are 36603/15688/58101 examples in train-

ing/validation/test sets. Each example has 54 features, each with unit cost. Our base RF consists of 40 trees using entropy split criteria and choosing from the full set of features at each split. We use 1000 trees for GREEDYMISER and search over learning rates in $[10^{-5}, 10^2]$ for MiniBooNE and Forest. As shown in (a) and (b) of Figure 4-3, BUDGETPRUNE (in red) achieves the best accuracy-cost tradeoff. The advantage of BUDGETPRUNE is particularly large in (b). GREEDYMISER has lower accuracy in the high budget region compared to BUDGETPRUNE in (a) and significantly lower accuracy in (b). The gap between BUDGETPRUNE and other pruning methods is small in (a) but much larger in (b). This indicates large gains from globally encouraging feature sharing in the case of (b) compared to (a). In both datasets, BUDGETPRUNE successfully prunes away large number of features while maintaining high accuracy. For example in (a), using only 18 unique features on average instead of 40, we can get essentially the same accuracy as the original RF.

Yahoo! Learning to Rank:(Chapelle et al., 2011) This ranking dataset consists of 473134 web documents and 19944 queries. Each example in the dataset contains features of a query-document pair together with the relevance rank of the document to the query. There are 141397/146769/184968 examples in the training/validation/test sets. There are 519 features for each example; each feature is associated with an acquisition cost in the set $\{1, 5, 20, 50, 100, 150, 200\}$, which represents the units of CPU time required to extract the feature and is provided by a Yahoo! employee. The labels are binarized so that the document is either relevant or not relevant to the query. The task is to learn a model that takes a new query and its associated set of documents to produce an accurate ranking using as little feature cost as possible. We use 3000 trees for GREEDYMISER and search over learning rates in $[10^{-5}, 1]$. As in (Nan et al., 2015), we use the Average Precision@5 as the performance metric, which gives a high reward for ranking the relevant documents on

top. Our base RF consists of 140 trees using cost weighted entropy split criteria as in (Nan et al., 2015) and choosing from a random subset of 400 features at each split. As shown in (c) of Figure 4-3, BUDGETPRUNE achieves similar ranking accuracy as GREEDYMISER using only 30% of its cost.

Scene15 (Lazebnik et al., 2006): This scene recognition dataset contains 4485 images from 15 scene classes (labels). Following (Xu et al., 2012) we divide it into 1500/300/2685 examples for training/validation/test sets. we use a diverse set of visual descriptors varying in computation time: GIST, spatial HOG, Local Binary Pattern, self-similarity, texton histogram, geometric texton, geometric color and 177 object detectors from the Object Bank (Li et al., 2010). We treat each individual detector as an independent descriptor so we have 184 different visual descriptors in total. The acquisition costs of these visual descriptors range from 0.0374 to 9.2820. For each descriptor we train 15 one-vs-rest kernel SVMs and use the output (margins) as features. The best classifier based on individual descriptors achieves an accuracy of 77.8%. Note the features are grouped based on the visual descriptors. Once any feature corresponding to a visual descriptor is used for a test example, an acquisition cost of the visual descriptor is incurred and subsequent usage of features from the same group is free for the test example. Our base RF consists of 500 trees using entropy split criteria and choosing from a random subset of 20 features at each split. As shown in (d) of Figure 4-3, BUDGETPRUNE and GREEDYPRUNE significantly outperform other competing methods. BUDGETPRUNE has the same accuracy at the cost of 9 as at the full cost of 32. BUDGETPRUNE and GREEDYPRUNE perform similarly, indicating the greedy approach happen to solve the global optimization in this particular initial RF.

4.5.2 Additional Experiments

We perform additional experiments to evaluate BUDGETPRUNE with different costs, input RFs.

Non-uniform cost on MiniBooNE We observe that CCP performs similarly to BUDGETPRUNE on MiniBooNE when the costs are uniform in the case of entropy splitting criteria, indicating little gain from global optimization with respect to feature usage. We suspect that uniform feature costs work in favor of CCP because there's no loss in treating each feature equally. To confirm this intuition we assign the features non-uniform costs and re-run prunings on the same RF. We first normalize the data so that the data vectors corresponding to the features have the same l_2 norm. We then train a linear SVM on it and obtain the weight vector corresponding to the learned hyperplane. We around the absolute values of the weights and make them the costs for the features. Intuitively the feature with higher weight tends to be more relevant for the classification task so we assign it a higher acquisition cost. The resulting costs lie in the range of $[1, 40]$ and we normalize them so that the sum of all feature costs is 50 - the number of features. We plot BUDGETPRUNE and CCP for uniform cost as well as the non-uniform cost described above in Figure 4-4. BUDGETPRUNE still achieves similar performance as uniform cost while CCP performance drops significantly with non-uniform feature cost. This shows again the importance of taking into account feature costs in the pruning process.

Entropy Vs Pairs How does BUDGETPRUNE depend on the splitting criteria used in the underlying random forest? On two data sets we build RFs using the popular entropy splitting criteria and the mini-max Pairs criteria used in (Nan et al., 2015) and the results are shown in Figure 4-5. We observe that entropy splitting criteria lead to RFs with higher accuracy while the Pairs criteria lead to RFs with lower cost.

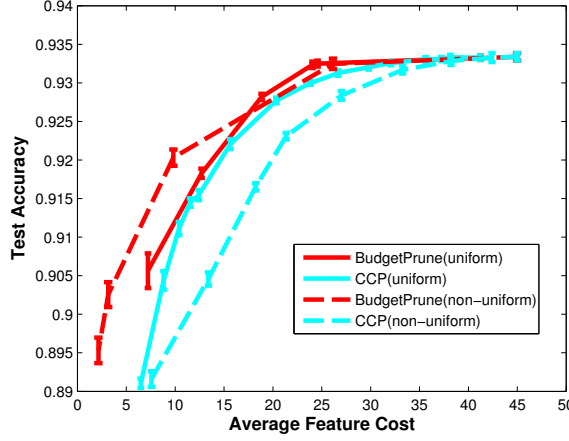


Figure 4-4: Comparing BUDGETPRUNE and CCP with uniform and non-uniform feature cost on MiniBooNE dataset. BUDGETPRUNE is robust when the feature cost is non-uniform.

This is expected as using Pairs biases to more balanced splits and thus provably low cost (Nan et al., 2015). In (a) of Figure 4-5 we observe that as more of the RF is pruned away BUDGETPRUNE and CCP results for entropy and Pairs coincide. This suggests that the two criteria actually lead to similar tree structures in the initial tree-building process. However, as the trees are built deeper their structures diverge. Plot (b) in Figure 4-5 shows that pruning based on the RFs from the Pairs criteria can achieve higher accuracy in the low cost region. But if high accuracy in the high cost region is desirable then the entropy criteria should be used.

Size of random feature subset at each split At each split in RF building, it is possible to restrict the choice of splitting feature to be among a random subset of all features. Such restriction tends to further reduce correlation among trees and gain prediction accuracy. The drawback is that test examples tend to encounter a diverse set of features, increasing feature acquisition cost. For illustration purpose, we plot various pruning results on Scene15 dataset for feature subset sizes $k = 20$ and $k = 120$ in Figure 4-6. The initial RF has higher accuracy and higher cost for

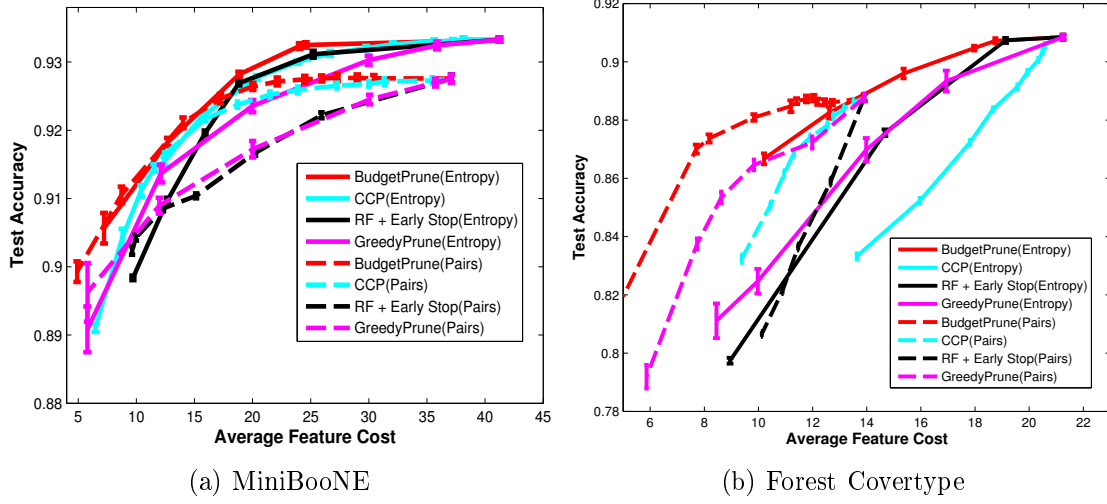


Figure 4-5: Comparisons of various pruning methods based on entropy and Pairs splitting criteria on MiniBooNE and Forest datasets

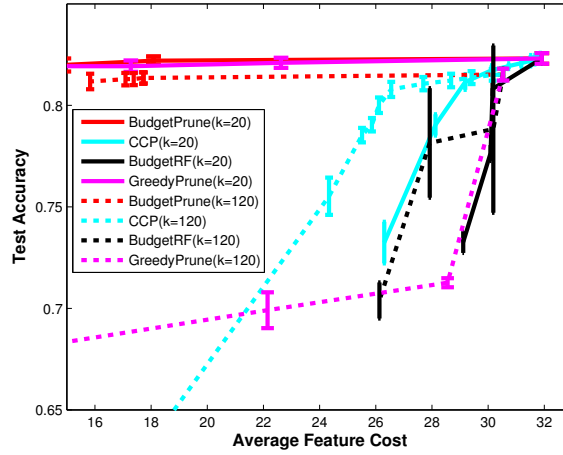


Figure 4-6: Comparing various pruning approaches on RF built with $k=20$ and $k=120$ on Scene15 dataset. The initial RF has higher accuracy and higher cost for $k=20$. GREEDYPRUNE performs very well in $k=20$ but very poorly in $k=120$.

$k = 20$ as expected. BUDGETPRUNE achieves slightly better accuracy in $k = 20$ than $k = 120$. Note also how GREEDYPRUNE performance drops significantly for $k = 120$ so it is not robust. In our main experiments k is chosen on validation data to achieve highest accuracy for the initial RF.

4.5.3 Discussion and Conclusion

We have empirically evaluated several resource constrained learning algorithms including BUDGETPRUNE and its variations on benchmarked datasets. We highlight key features of our approach below. (i) STATE-OF-THE-ART METHODS. Recent work has established that GREEDYMISER and BUDGETRF are among the state-of-the-art methods dominating a number of other methods (Kusner et al., 2014; Xu et al., 2013; Wang et al., 2015) on these benchmarked datasets. GREEDYMISER requires building class-specific ensembles and tends to perform poorly and is increasingly difficult to tune in multi-class settings. RF, by its nature, can handle multi-class settings efficiently. On the other hand, as we described earlier, (Kusner et al., 2014; Wang et al., 2015; Xu et al., 2013) are fundamentally "tree-growing" approaches, namely they are top-down methods acquiring features sequentially based on a surrogate utility value. This is a fundamentally combinatorial problem that is known to be NP hard (Chakravarthy et al., 2011; Xu et al., 2013) and thus requires a number of relaxations and heuristics with no guarantees on performance. In contrast our pruning strategy is initialized to realize good performance (RF initialization) and we are able to globally optimize cost-accuracy objective. (ii) VARIATIONS ON PRUNING. By explicitly modeling feature costs, BUDGETPRUNE outperforms other pruning methods such as early stopping of BUDGETRF and CCP that do not consider costs. GREEDYPRUNE performs well validating our intuition (see Table. 1) that pruning sparsely occurring feature nodes utilized by large fraction of examples can improve test-time cost-accuracy tradeoff. Nevertheless, the BUDGETPRUNE outperforms GREEDYPRUNE, which is in-

dicative of the fact that apart from obvious high-budget regimes, node-pruning must account for how removal of one node may have an adverse impact on another downstream one. (iii) SENSITIVITY TO IMPURITY, FEATURE COSTS, & OTHER INPUTS. We experiment BUDGETPRUNE with different impurity functions such as entropy and Pairs (Nan et al., 2015) criteria. Pairs-impurity tends to build RFs with lower cost but also lower accuracy compared to entropy and so has poorer performance. We also explored how non-uniform costs can impact cost-accuracy tradeoff. An elegant approach has been suggested by (Benbouzid, 2014), who propose an adversarial feature cost proportional to feature utility value. We find that BUDGETPRUNE is robust with such costs. Other RF parameters including number of trees and feature subset size at each split do impact cost-accuracy tradeoff in obvious ways with more trees and moderate feature subset size improving prediction accuracy while incurring higher cost.

Chapter 5

Adaptive Approximation

So far in this thesis we have focused on feature acquisition cost. Other costs such as communication and latency costs pose a key challenge in the design of mobile computing, or the Internet-of-Things(IoT) applications, where a large number of sensors/camera/watches/phones (known as edge devices) are connected to a cloud.

Adaptive System: Rather than having the edge devices constantly transmit measurements/images to the cloud where a centralized model makes prediction, a more efficient approach is to allow the edge devices make predictions locally (Kumar et al., 2017), whenever possible, saving the high communication cost and reducing latency. Due to the memory, computing and battery constraints, the prediction models on the edge devices are limited to low complexity. Consequently, to maintain high-accuracy, adaptive systems are desirable. Such systems identify easy-to-handle input instances where local edge models suffice, thus limiting the utilization cloud services for only hard instances. We propose to learn an adaptive system by training on fully annotated training data. Our objective is to maintain high accuracy while meeting average resource constraints during prediction-time.

There have been a number of promising approaches that focus on methods for reducing costs while improving overall accuracy as discussed in Section 1.5. Many of these methods train models in a top-down manner, namely, attempt to build out the model by selectively adding the most cost-effective features to improve accuracy.

In contrast we propose a novel bottom-up approach. We train adaptive models on

annotated training data by selectively identifying parts of the input space for which high accuracy can be maintained at a lower cost. The principal advantage of our method is twofold. First, our approach can be readily applied to cases where it is desirable to reduce costs of an existing high-cost legacy system. Second, training top-down models in case of feature costs leads to fundamental combinatorial issues in multi-stage search over all feature subsets (see Sec. 5.1). In contrast, we bypass many of these issues by posing a natural adaptive approximation objective to partition the input space into easy and hard cases.

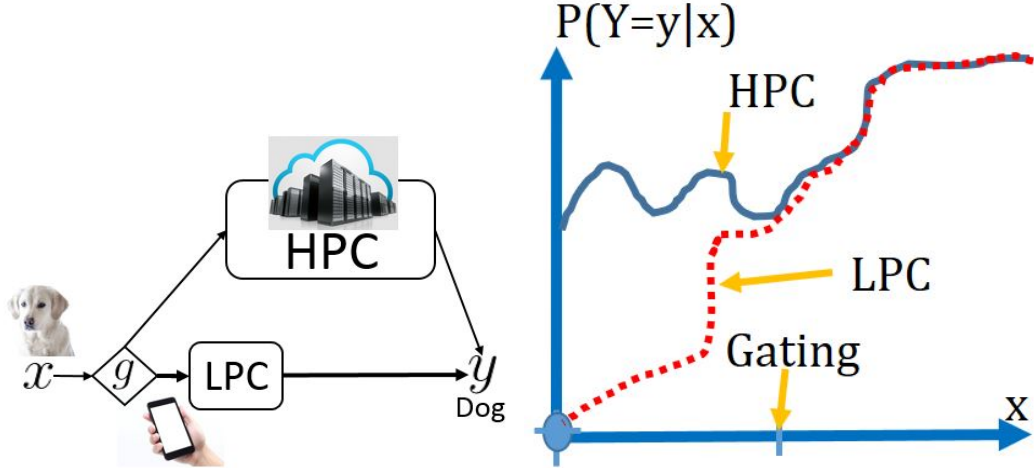


Figure 5.1: **Left:** single stage schematic of our approach. We learn low-cost gating g and a LPC model to adaptively approximate a HPC model. **Right:** Key insight for adaptive approximation. x-axis represents feature space; y-axis represents conditional probability of correct prediction; LPC can match HPC’s prediction in the input region corresponding to the right of the gating threshold but performs poorly otherwise. Our goal is to learn a low-cost gating function that attempts to send examples on the right to LPC and the left to HPC.

In particular, when no legacy system is available, our method consists of first learning a high-accuracy model that minimizes the empirical loss regardless of costs. The resulting high prediction-cost model (HPC) can be readily trained using any of the existing methods. For example, this could be a large neural network in the cloud

that achieves the state-of-the-art accuracy. Next, we jointly learn a low-cost gating function as well as a low prediction-cost (LPC) model so as to *adaptively approximate* the high-accuracy model by identifying regions of input space where a low-cost gating and LPC model are adequate to achieve high-accuracy. In IoT applications, such low-complexity models can be deployed on the edge devices to perform gating and prediction. At test-time, for each input instance, the gating function decides whether or not the LPC model is adequate for accurate classification. Intuitively, “easy” examples can be correctly classified using only an LPC model while “hard” examples require HPC model. By identifying which of the input instances can be classified accurately with LPCs we bypass the utilization of HPC model, thus reducing average prediction cost. The left part of Figure 5-1 is a schematic of our approach, where x is feature vector and y is the predicted label; we aim to learn g and an LPC model to adaptively approximate the HPC. The key observation as depicted in the lower figure is that the probability of correct classification given x for a HPC model is in general a highly complex function with higher values than that of a LPC model. Yet there exists regions of the input space where the LPC has competitive accuracy (as shown to the right of the gating threshold). Sending examples in such regions (according to the gating function) to the LPC results in no loss of prediction accuracy while reducing prediction costs.

The problem would be simpler if our task were to primarily partition the input space into regions where LPC models would suffice. The difficulty is that we must also learn a low-cost gating function capable of identifying input instances for which LPC suffices. Since both prediction and gating account for cost, we favor design strategies that lead to shared features and decision architectures between the gating function and the LPC model. We pose the problem as a discriminative empirical risk minimization problem that jointly optimizes for gating and prediction models in

terms of a joint margin-based objective function. The resulting objective is separately convex in gating and prediction functions. We propose an alternating minimization scheme that is guaranteed to converge since with appropriate choice of loss-functions (for instance, logistic loss), each optimization step amounts to a probabilistic approximation/projection (I-projection/M-projection) onto a probability space. While our method can be recursively applied in multiple stages to successively approximate the adaptive system obtained in the previous stage, thereby refining accuracy-cost trade-off, we observe that on benchmark datasets even a single stage of our method outperforms state-of-art in accuracy-cost performance.

The work presented in this chapter is published in (Nan and Saligrama, 2017).

5.1 Related Work

Top-Down Methods: For high-dimensional spaces, many existing approaches focus on learning complex adaptive decision functions top-down (Gao and Koller, 2011; Xu et al., 2012; Kusner et al., 2014; Wang et al., 2015). Conceptually, during training, top-down methods acquire new features based on their utility value. This requires exploration of partitions of the input space together with different combinatorial low-cost feature subsets that would result in higher accuracy. These methods are based on multi-stage exploration leading to combinatorially hard problems. Different novel relaxations and greedy heuristics have been developed in this context.

Bottom-up Methods: The work in this Chapter is somewhat related to that in Chapter 4, where we propose to prune a fully trained random forests (RF) to reduce costs. Nevertheless, in contrast to the adaptive system, the RF pruning perspective is to compress the original model and utilize the pruned forest as a stand-alone model for test-time prediction. Furthermore, the work in 4 is specifically tailored to random forests whereas the adaptive approximation framework proposed in this chapter is for

general models.

The teacher-student framework (Lopez-Paz et al., 2016) is also related to our bottom-up approach; a low-cost student model learns to approximate the teacher model so as to meet test-time budget. However, the goal there is to learn a better stand-alone student model. In contrast, we make use of both the low-cost (student) and high-accuracy (teacher) model during prediction via a gating function, which learns the limitation of the low-cost (student) model and consult the high-accuracy (teacher) model if necessary, thereby avoiding accuracy loss. Our composite system is also related to HME (Jordan and Jacobs, 1994), which learns the composite system based on max-likelihood estimation of models. A major difference is that HME does not address budget constraints. A fundamental aspect of budget constraints is the resulting asymmetry, whereby, we start with an HPC model and sequentially approximate with LPCs. This asymmetry leads us to propose a bottom-up strategy where the high-accuracy predictor can be separately estimated and is critical to posing a direct empirical loss minimization problem.

5.2 Problem Setup

We consider the standard learning scenario of resource constrained prediction with feature costs. A training sample $S = \{(x^{(i)}, y^{(i)}) : i = 1, \dots, N\}$ is generated i.i.d. from an unknown distribution, where $x^{(i)} \in \mathbb{R}^K$ is the feature vector with an acquisition cost $c_\alpha \geq 0$ assigned to each of the features $\alpha = 1, \dots, K$ and $y^{(i)}$ is the label for the i^{th} example. In the case of multi-class classification $y \in \{1, \dots, M\}$, where M is the number of classes. Let us consider a single stage of our training method in order to formalize our setup. The model, f_0 , is a high prediction-cost (HPC) model, which is either a priori known, or which we train to high-accuracy regardless of cost considerations. We would like to learn an alternative low prediction-cost (LPC) model

f_1 . Given an example x , at test-time, we have the option of selecting which model, f_0 or f_1 , to utilize to make a prediction. The accuracy of a prediction model f_z is modeled by a loss function $\ell(f_z(x), y), z \in \{0, 1\}$. We exclusively employ the logistic loss function in binary classification: $\ell(f_z(x), y) = \log(1 + \exp(-yf_z(x)))$, although our framework allows other loss models. For a given x , we assume that once it pays the cost to acquire a feature, its value can be efficiently cached; its subsequent use does not incur additional cost. Thus, the cost of utilizing a particular prediction model, denoted by $c(f_z, x)$, is computed as the sum of the acquisition cost of *unique* features required by f_z .

Oracle Gating: Consider a general gating likelihood function $q(z|x)$ with $z \in \{0, 1\}$, that outputs the likelihood of sending the input x to a prediction model, f_z . The overall empirical loss is:

$$\mathbb{E}_{S_n} \mathbb{E}_{q(z|x)} [\ell(f_z(x), y)] = \mathbb{E}_{S_n} [\ell(f_0(x), y)] + \underbrace{\mathbb{E}_{S_n} [q(1|x) (\ell(f_1(x), y) - \ell(f_0(x), y))]}_{\text{Excess Loss}}$$

The first term only depends on f_0 , and from our perspective a constant. Similar to average loss we can write the average cost as (assuming gating cost is negligible for now):

$$\mathbb{E}_{S_n} \mathbb{E}_{q(z|x)} [c(f_z, x)] = \mathbb{E}_{S_n} [c(f_0, x)] - \underbrace{\mathbb{E}_{S_n} [q(1|x) (c(f_0, x) - c(f_1, x))]}_{\text{Cost Reduction}},$$

where the first term is again constant. We can characterize the optimal gating function (see (Trapeznikov and Saligrama, 2013)) that minimizes the overall average loss subject to average cost constraint:

$$\underbrace{\ell(f_1, x) - \ell(f_0, x)}_{\text{Excess loss}} \underset{q(1|x)=1}{\overset{q(1|x)=0}{\geq}} \underbrace{\eta (c(f_0, x) - c(f_1, x))}_{\text{Cost reduction}}$$

for a suitable choice $\eta \in \mathbb{R}$. This characterization encodes the important principle that if the marginal cost reduction is smaller than the excess loss, we opt for the HPC model. Nevertheless, this characterization is generally infeasible. Note that the LHS depends on knowing how well HPC performs on the input instance. Since this information is unavailable, this target can be unreachable with low-cost gating.

Gating Approximation: Rather than directly enforcing a low-cost structure on q , we decouple the constraint and introduce a parameterized family of gating functions $g \in \mathcal{G}$ that attempts to mimic (or approximate) q . \mathcal{G} can be the family of linear classifiers, gradient boosted trees or neural networks. By passing the output of g through the sigmoid function $\sigma(s) = 1/(1 + e^{-s})$, we obtain a probability distribution over the gating output classes (sending the example to f_0 or f_1). To ensure the gating distribution approximates q , we can minimize some distance measure $D(q(\cdot|x), g(x))$. A natural choice for an approximation metric is the Kullback-Leibler (KL) divergence although other choices are possible. The KL divergence between q and g is given by $D_{KL}(q(\cdot|x)||g(x)) = \sum_z q(z|x) \log(q(z|x)/\sigma(\text{sgn}(0.5 - z)g(x)))$. Besides KL divergence, we have also proposed another symmetric metric fitting g directly to the log odds ratio of q . See Appendix A for details.

Budget Constraint: With the gating function g , the cost of predicting x depends on whether the example is sent to f_0 or f_1 . Let $c(f_0, g, x)$ denote the feature cost of passing x to f_0 through g . As discussed, this is equal to the sum of the acquisition cost of unique features required by f_0 and g for x . Similarly $c(f_1, g, x)$ denotes the cost if x is sent to f_1 through g . In many cases the cost $c(f_z, g, x)$ is independent of the example x and depends primarily on the model being used. This is true for linear models where each x must be processed through the same collection of features. For these cases $c(f_z, g, x) \triangleq c(f_z, g)$. The total budget simplifies to: $\mathbb{E}_{S_n}[q(0|x)]c(f_0, g) + (1 - \mathbb{E}_{S_n}[q(0|x)])c(f_1, g) = c(f_1, g) + \mathbb{E}_{S_n}[q(0|x)](c(f_0, g) - c(f_1, g))$. The budget thus

depends on 3 quantities: $\mathbb{E}_{S_n}[q(0|x)]$, $c(f_1, g)$ and $c(f_0, g)$. Often f_0 is a high-cost model that requires most, if not all, of features so $c(f_0, g)$ can be considered a large constant.

Thus, to meet the budget constraint, we would like to have (a) low-cost g and f_1 (small $c(f_1, g)$); and (b) small fraction of examples being sent to the high-accuracy model (small $\mathbb{E}_{S_n}[q(0|x)]$). We can therefore split the budget constraint into two separate objectives: (a) ensure low-cost through penalty $\Omega(f_1, g) = \gamma \sum_{\alpha} c_{\alpha} \|V_{\alpha} + W_{\alpha}\|_0$, where γ is a tradeoff parameter and the indicator variables $V_{\alpha}, W_{\alpha} \in \{0, 1\}$ denote whether or not the feature α is required by f_1 and g , respectively. Depending on the model parameterization, we can approximate $\Omega(f_1, g)$ using a group-sparse norm or in a stage-wise manner as we will see in Algorithms 4 and 5. (b) Ensure only P_{full} fraction of examples are sent to f_0 via the constraint $\mathbb{E}_{S_n}[q(0|x)] \leq P_{\text{full}}$.

Putting Together: We are now ready to pose our general optimization problem:

$$\begin{aligned} \min_{f_1 \in \mathcal{F}, g \in \mathcal{G}, q} \mathbb{E}_{S_n} \sum_z \overbrace{[q(z|x)\ell(f_z(x), y)]}^{\text{Losses}} + \overbrace{D(q(\cdot|x), g(x))}^{\text{Gating Approx}} + \overbrace{\Omega(f_1, g)}^{\text{Costs}} \quad (\text{OPT}) \\ \text{subject to: } \mathbb{E}_{S_n}[q(0|x)] \leq P_{\text{full}}. \quad (\text{Fraction to } f_0) \end{aligned}$$

The objective function penalizes excess loss and ensures through the second term that this excess loss can be enforced through admissible gating functions. The third term penalizes the feature cost usage of f_1 and g . The budget constraint limits the fraction of examples sent to the costly model f_0 .

Remark 1: Directly parameterizing q leads to non-convexity. Average loss is q -weighted sum of losses from HPC and LPC; while the space of probability distributions is convex, a finite-dimensional parameterization is generally non-convex (e.g. sigmoid). What we have done is to keep q in non-parametric form to avoid non-convexity and only parameterize g , connecting both via a KL term. Thus, (OPT)

is now convex with respect to the f_1 and g for a fixed q . It is again convex in q for a fixed f_1 and g . Otherwise it would introduce non-convexity as in prior work. For instance, in (Chen et al., 2012) a non-convex problem is solved in each inner loop iteration (line 7 of their Algorithm 1).

Remark 2: We presented the case for a single stage approximation system. However, it is straightforward to recursively continue this process. We can then view the composite system $f_0 \triangleq (g, f_1, f_0)$ as a black-box predictor and train a new pair of gating and prediction models to approximate the composite system.

Remark 3: To limit the scope of this chapter, we focus on reducing feature acquisition cost during prediction as it is a more challenging (combinatorial) problem. However, other prediction-time costs such as computation cost can be encoded in the choice of functional classes \mathcal{F} and \mathcal{G} in (OPT).

Surrogate Upper Bound of Composite System: We can get better insight for the first two terms of the objective in (OPT) if we view $z \in \{0, 1\}$ as a latent variable and consider the composite system $\Pr(y|x) = \sum_z \Pr(z|x; g) \Pr(y|x, f_z)$. A standard application of Jensen's inequality reveals that, $-\log(\Pr(y|x)) \leq \mathbb{E}_{q(z|x)} \ell(f_z(x), y) + D_{KL}(q(z|x) \| \Pr(z|x; g))$. Therefore, the conditional-entropy of the composite system is bounded by the expected value of our loss function (we overload notation and represent random-variables in lower-case format):

$$H(y | x) \triangleq \mathbb{E}[-\log(\Pr(y|x))] \leq \mathbb{E}_{x \times y}[\mathbb{E}_{q(z|x)} \ell(f_z(x), y) + D_{KL}(q(z|x) \| \Pr(z|x; g))].$$

This implies that the first two terms of our objective attempt to bound the loss of the composite system; the third term in the objective together with the constraint serve to enforce budget limits on the composite system.

Group Sparsity: Since the cost for feature re-use is zero we encourage feature re-use among gating and prediction models. So the fundamental question here is:

How to choose a common, sparse (low-cost) subset of features on which both g and f_1 operate, such that g can effective gate examples between f_1 and f_0 for accurate prediction? This is a hard combinatorial problem. The main contribution of this chapter is to address it using the general optimization framework of (OPT).

5.3 Algorithms

To be concrete, we instantiate our general framework (OPT) into two algorithms via different parameterizations of g, f_1 : ADAPT-LIN for the linear class and ADAPT-GBRT for the non-parametric class.

Algorithm 4 ADAPT-LIN

Input: $(x^{(i)}, y^{(i)}), P_{\text{full}}, \gamma$
 Train f_0 . Initialize g, f_1 .
repeat
 Solve (OPT1) for q given g, f_1 .
 Solve (OPT2) for g, f_1 given q .
until convergence

Algorithm 5 ADAPT-GBRT

Input: $(x^{(i)}, y^{(i)}), P_{\text{full}}, \gamma$
 Train f_0 . Initialize g, f_1 .
repeat
 Solve (OPT1) for q given g, f_1 .
 for $t = 1$ **to** T **do**
 Find f_1^t using CART to minimize (5.1).
 $f_1 = f_1 + f_1^t$.
 For each feature α used, set $u_\alpha = 0$.
 Find g^t using CART to minimize (5.2).
 $g = g + g^t$.
 For each feature α used, set $u_\alpha = 0$.
until convergence

Both of them use the KL-divergence as distance measure. We also provide a third algorithm ADAPT-LSTSQ that uses the symmetric distance in the Appendix.

All of the algorithms perform alternating minimization of (OPT) over q, g, f_1 . Note that convergence of alternating minimization follows as in (Ganchev et al., 2008). Common to all of our algorithms, we use two parameters to control cost: P_{full} and γ . In practice they are swept to generate various cost-accuracy tradeoffs and we choose the best one satisfying the budget B using validation data.

ADAPT-LIN: Let $g(x) = g^T x$ and $f_1(x) = f_1^T x$ be linear classifiers. A feature is used if the corresponding component is non-zero: $V_\alpha = 1$ if $f_{1,\alpha} \neq 0$, and $W_\alpha = 1$ if $g_\alpha \neq 0$. The minimization for q solves the following problem:

$$\begin{aligned} \min_q \quad & \frac{1}{N} \sum_{i=1}^N [(1 - q_i)A_i + q_i B_i - H(q_i)] \\ \text{s.t.} \quad & \frac{1}{N} \sum_{i=1}^N q_i \leq P_{\text{full}}, \end{aligned} \tag{OPT1}$$

where we have used shorthand notations $q_i = q(z = 0|x^{(i)})$, $H(q_i) = -q_i \log(q_i) - (1 - q_i) \log(1 - q_i)$, $A_i = \log(1 + e^{-y^{(i)} f_1^T x^{(i)}}) + \log(1 + e^{g^T x^{(i)}})$ and $B_i = -\log p(y^{(i)}|z^{(i)} = 0; f_0) + \log(1 + e^{-g^T x^{(i)}})$. This optimization has a closed form solution: $q_i = 1/(1 + e^{B_i - A_i + \beta})$ for some non-negative constant β such that the constraint is satisfied. This optimization is also known as I-Projection in information geometry because of the entropy term (Ganchev et al., 2008). Having optimized q , we hold it constant and minimize with respect to g, f_1 by solving the problem (OPT2), where we have relaxed the non-convex cost $\sum_\alpha c_\alpha \|V_\alpha + W_\alpha\|_0$ into a $L_{2,1}$ norm for group sparsity and a tradeoff parameter γ to make sure the feature budget is satisfied. Once we solve for g, f_1 , we can hold them constant and minimize with respect to q again. ADAPT-LIN is summarized in Algorithm 4.

$$\begin{aligned} \min_{g, f_1} \quad & \frac{1}{N} \sum_{i=1}^N \left[(1 - q_i) \left(\log(1 + e^{-y^{(i)} f_1^T x^{(i)}}) + \log(1 + e^{g^T x^{(i)}}) \right) + q_i \log(1 + e^{-g^T x^{(i)}}) \right] \\ & + \gamma \sum_{\alpha} \sqrt{g_\alpha^2 + f_{1,\alpha}^2}. \end{aligned} \tag{OPT2}$$

ADAPT-GBRT: We can also consider the non-parametric family of classifiers

such as gradient boosted trees (Friedman, 2000): $g(x) = \sum_{t=1}^T g^t(x)$ and $f_1(x) = \sum_{t=1}^T f_1^t(x)$, where g^t and f_1^t are limited-depth regression trees. Since the trees are limited to low depth, we assume that the feature utility of each tree is example-independent: $V_{\alpha,t}(x) \cong V_{\alpha,t}, W_{\alpha,t}(x) \cong W_{\alpha,t}, \forall x$. $V_{\alpha,t} = 1$ if feature α appears in f_1^t , otherwise $V_{\alpha,t} = 0$, similarly for $W_{\alpha,t}$. The optimization over q still solves (OPT1). We modify $A_i = \log(1 + e^{-y^{(i)} f_1(x^{(i)})}) + \log(1 + e^{g(x^{(i)})})$ and $B_i = -\log p(y^{(i)} | z^{(i)} = 0; f_0) + \log(1 + e^{-g(x^{(i)})})$. Next, to minimize over g, f_1 , denote loss:

$$\ell(f_1, g) = \frac{1}{N} \sum_{i=1}^N \left[(1 - q_i) \cdot \left(\log(1 + e^{-y^{(i)} f_1(x^{(i)})}) + \log(1 + e^{g(x^{(i)})}) \right) + q_i \log(1 + e^{-g(x^{(i)})}) \right],$$

which is essentially the same as the first part of the objective in (OPT2). Thus, we need to minimize $\ell(f_1, g) + \Omega(f_1, g)$ with respect to f_1 and g . Since both f_1 and g are gradient boosted trees, we naturally adopt a stage-wise approximation for the objective. In particular, we define an impurity function which on the one hand approximates the negative gradient of $\ell(f_1, g)$ with the squared loss, and on the other hand penalizes the initial acquisition of features by their cost c_α . To capture the initial acquisition penalty, we let $u_\alpha \in \{0, 1\}$ indicates if feature α has already been used in previous trees ($u_\alpha = 0$), or not ($u_\alpha = 1$). u_α is updated after adding each tree. Thus we arrive at the following impurity for f_1 and g , respectively:

$$\frac{1}{2} \sum_{i=1}^N \left(-\frac{\partial \ell(f_1, g)}{\partial f_1(x^{(i)})} - f_1^t(x^{(i)}) \right)^2 + \gamma \sum_{\alpha} u_\alpha c_\alpha V_{\alpha,t}, \quad (5.1)$$

$$\frac{1}{2} \sum_{i=1}^N \left(-\frac{\partial \ell(f_1, g)}{\partial g(x^{(i)})} - g^t(x^{(i)}) \right)^2 + \gamma \sum_{\alpha} u_\alpha c_\alpha W_{\alpha,t}. \quad (5.2)$$

Minimizing such impurity functions balances the need to minimize loss and re-using the already acquired features. Classification and Regression Tree (CART) (Breiman et al., 1984) can be used to construct decision trees with such an impurity function. ADAPT-GBRT is summarized in Algorithm 5. Note that a similar impurity is used in

GREEDYMISER (Xu et al., 2012). Interestingly, if P_{full} is set to 0, all the examples are forced to f_1 , then ADAPT-GBRT exactly recovers the GREEDYMISER. In this sense, GREEDYMISER is a special case of our algorithm. As we will see in the next section, thanks to the bottom-up approach, ADAPT-GBRT benefits from high-accuracy initialization and is able to perform accuracy-cost tradeoff in accuracy levels beyond what is possible for GREEDYMISER.

5.4 Experiments

BASLINE ALGORITHMS: We consider the following simple L1 baseline approach for learning f_1 and g : first perform a L1-regularized logistic regression on all data to identify a relevant, sparse subset of features; then learn f_1 using training data restricted to the identified feature(s); finally, learn g based on the correctness of f_1 predictions as pseudo labels (i.e. assign pseudo label 1 to example x if $f_1(x)$ agrees with the true label y and 0 otherwise). We also compare with two state-of-the-art feature-budgeted algorithms: GREEDYMISER(Xu et al., 2012) - a top-down method that builds out an ensemble of gradient boosted trees with feature cost budget; and BUDGETPRUNE(Nan et al., 2016) - a bottom-up method that prunes a random forest with feature cost budget. A number of other methods such as ASTC (Kusner et al., 2014) and CSTC (Xu et al., 2013) are omitted as they have been shown to underperform GREEDYMISER on the same set of datasets (Nan et al., 2015). Detailed experiment setups can be found in the Appendix.

We first visualize/verify the adaptive approximation ability of ADAPT-LIN and ADAPT-GBRT on the Synthetic-1 dataset without feature costs. Next, we illustrate the key difference between ADAPT-LIN and the L1 baseline approach on the Synthetic-2 as well as the Letters datasets. Finally, we compare ADAPT-GBRT with state-of-the-art methods on several resource constraint benchmark datasets.

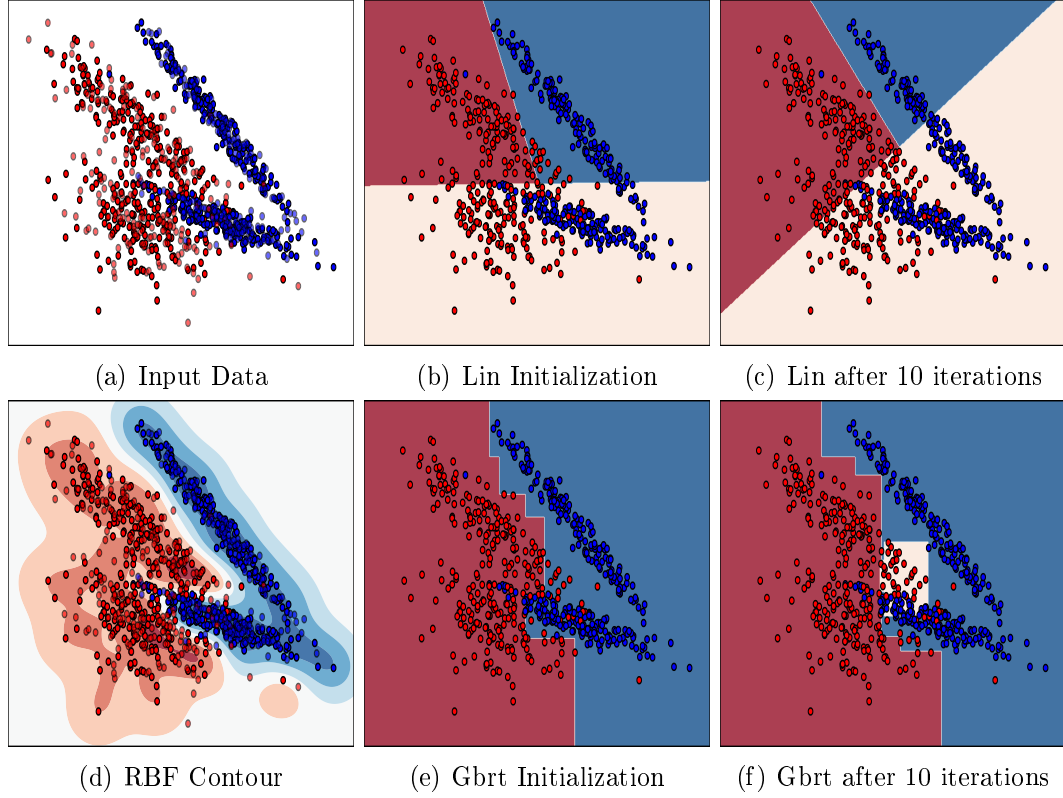


Figure 5-2: Synthetic-1 experiment without feature cost. (a): input data. (d): decision contour of RBF-SVM as f_0 . (b) and (c): decision boundaries of linear g and f_1 at initialization and after 10 iterations of ADAPT-LIN. (e) and (f): decision boundaries of boosted tree g and f_1 at initialization and after 10 iterations of ADAPT-GBRT. Examples in the beige areas are sent to f_0 by the g .

POWER OF ADAPTATION: We construct a 2D binary classification dataset (Synthetic-1) as shown in (a) of Figure 5-2. We learn an RBF-SVM as the high-accuracy classifier f_0 as in (d). To better visualize the adaptive approximation process in 2D, we turn off the feature costs (i.e. set $\Omega(f_1, g)$ to 0 in (OPT)) and run ADAPT-LIN and ADAPT-GBRT. The initializations of g and f_1 in (b) results in wrong predictions for many red points in the blue region. After 10 iterations of ADAPT-LIN, f_1 adapts much better to the local region assigned by g while g sends about 60% (P_{full}) of examples to f_0 . Similarly, the initialization in (e) results in wrong predictions in

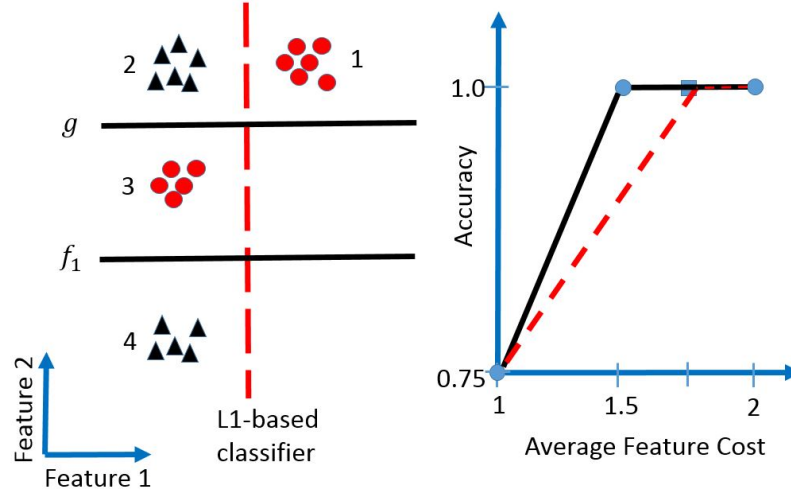


Figure 5-3: A 2-D synthetic example for adaptive feature acquisition. On the left: data distributed in four clusters. The two features correspond to x and y coordinates, respectively. On the right: accuracy-cost tradeoff curves. Our algorithm can recover the optimal adaptive system whereas a L1-based approach cannot.

the blue region. ADAPT-GBRT is able to identify the ambiguous region in the center and send those examples to f_0 via g . Both of our algorithms maintain the same level of prediction accuracy as f_0 yet are able to classify large fractions of examples via much simpler models.

POWER OF JOINT OPTIMIZATION: We return to the problem of prediction under feature budget constraints. We illustrate why a simple L1 baseline approach for learning f_1 and g would not work using a 2D dataset (Synthetic-2) as shown in Figure 5-3 (left). The data points are distributed in four clusters, with black triangles and red circles representing two class labels. Let both feature 1 and 2 carry unit acquisition cost. A complex classifier f_0 that acquires both features can achieve full accuracy at the cost of 2. In our synthetic example, clusters 1 and 2 are given more data points so that the L1-regularized logistic regression would produce the vertical red dashed line, separating cluster 1 from the others. So feature 1 is acquired for both g and f_1 . The best such an adaptive system can do is to send cluster 1 to f_1 and

Table 5.1: Dataset Statistics

Dataset	#Train	#Validation	#Test	#Features	Feature Costs
Letters	12000	4000	4000	16	Uniform
MiniBooNE	45523	19510	65031	50	Uniform
Forest	36603	15688	58101	54	Uniform
CIFAR10	19761	8468	10000	400	Uniform
Yahoo!	141397	146769	184968	519	CPU units

the other three clusters to the complex classifier f_0 , incurring an average cost of 1.75, which is sub-optimal. ADAPT-LIN, on the other hand, optimizing between q, g, f_1 in an alternating manner, is able to recover the horizontal lines in Figure 5-3 (left) for g and f_1 . g sends the first two clusters to the full classifier and the last two clusters to f_1 . f_1 correctly classifies clusters 3 and 4. So all of the examples are correctly classified by the adaptive system; yet only feature 2 needs to be acquired for cluster 3 and 4 so the overall average feature cost is 1.5, as shown by the solid curve in the accuracy-cost tradeoff plot on the right of Figure 5-3. This example shows that the L1 baseline approach is sub-optimal as it doesnot optimize the selection of feature subsets *jointly* for g and f_1 .

REAL DATASETS: We test various aspects of our algorithms and compare with state-of-the-art feature-budgeted algorithms on five real world benchmark datasets: Letters, MiniBooNE Particle Identification, Forest Covertypes datasets from the UCI repository (Dheeru and Karra Taniskidou, 2017), CIFAR-10 (Krizhevsky, 2009) and Yahoo! Learning to Rank(Chapelle et al., 2011). Yahoo! is a ranking dataset where each example is associated with features of a query-document pair together with the relevance rank of the document to the query. There are 519 such features in total; each is associated with an acquisition cost in the set $\{1,5,20,50,100,150,200\}$, which represents the units of CPU time required to extract the feature and is provided by a Yahoo! employee. The labels are binarized into relevant or not relevant. The task is to learn a model that takes a new query and its associated documents and produce a

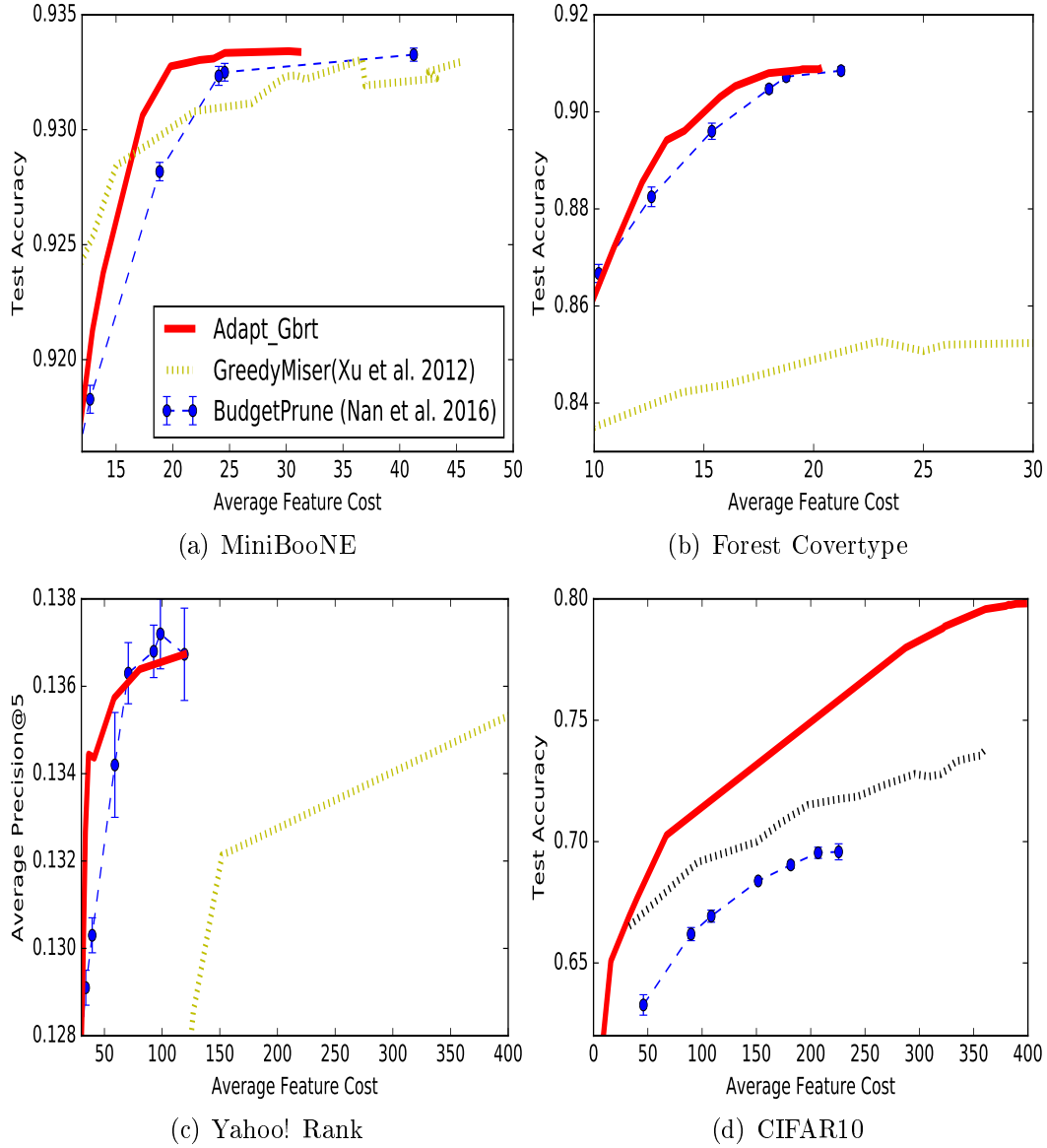


Figure 5-4: Comparison of ADAPT-GBRT against GREEDYMISER and BUDGETPRUNE on four benchmark datasets. RF is used as f_0 for ADAPT-GBRT in (a-c) while an RBF-SVM is used as f_0 in (d). ADAPT-GBRT achieves better accuracy-cost tradeoff than other methods. The gap is significant in (b) (c) and (d). Note the accuracy of GREEDYMISER in (b) never exceeds 0.86 and its precision in (c) slowly rises to 0.138 at cost of 658. We limit the cost range for a clearer comparison.

relevance ranking so that the relevant documents come on top, and to do this using as little feature cost as possible. The performance metric is Average Precision @ 5 following (Nan et al., 2016). The other datasets have unknown feature costs so we assign costs to be 1 for all features; the aim is to show ADAPT-GBRT successfully selects *sparse* subset of “usefull” features for f_1 and g . We summarize the statistics of these datasets in Table 5.1. Next, we highlight the key insights from the real dataset experiments.

Generality of Approximation: Our framework allows approximation of powerful classifiers such as RBF-SVM and Random Forests as shown in Figure 5-5 as red and black curves, respectively. In particular, ADAPT-GBRT can well maintain high accuracy while reducing cost. This is a key advantage for our algorithms because we can choose to approximate the f_0 that achieves the best accuracy.

ADAPT-LIN Vs L1: Figure 5-5 shows that ADAPT-LIN outperforms L1 baseline method on real dataset as well. Again, this confirms the intuition we have in the Synthetic-2 example as ADAPT-LIN is able to iteratively select the common subset of features jointly for g and f_1 .

ADAPT-GBRT Vs ADAPT-LIN: ADAPT-GBRT leads to significantly better performance than ADAPT-LIN in approximating both RBF-SVM and RF as shown in Figure 5-5. This is expected as the non-parametric non-linear classifiers are much more powerful than linear ones.

ADAPT-GBRT Vs BUDGETPRUNE: Both are bottom-up approaches that benefit from good initializations. In (a), (b) and (c) of Figure 5-4 we let f_0 in ADAPT-GBRT be the same RF that BUDGETPRUNE starts with. ADAPT-GBRT is able to maintain high accuracy longer as the budget decreases. Thus, ADAPT-GBRT improves state-of-the-art bottom-up method. Notice in (c) of Figure 5-4 around the cost of 100, BUDGETPRUNE has a spike in precision. We believe this is because the

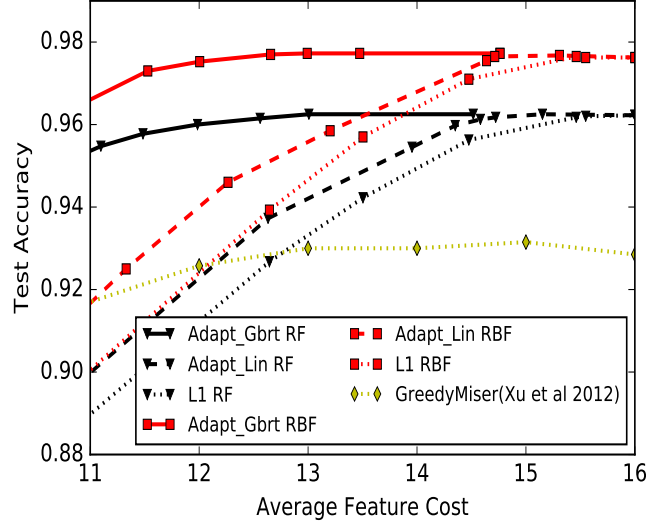


Figure 5.5: Compare the L1 baseline approach, ADAPT-LIN and ADAPT-GBRT based on RBF-SVM and RF as f_0 's on the Letters dataset.

initial pruning improved the generalization performance of RF.

But in the cost region of 40-80, ADAPT-GBRT maintains much better accuracy than BUDGETPRUNE. Furthermore, ADAPT-GBRT has the freedom to approximate the best f_0 given the problem. So in (d) of Figure 5.4 we see that with f_0 being RBF-SVM, ADAPT-GBRT can achieve much higher accuracy than BUDGETPRUNE.

ADAPT-GBRT Vs GREEDYMISER: ADAPT-GBRT outperforms GREEDYMISER on all the datasets. The gaps in Figure 5.5, (b) (c) and (d) of Figure 5.4 are especially significant.

Significant Cost Reduction: Without sacrificing top accuracies (within 1%), ADAPT-GBRT reduces average feature costs during test-time by around 63%, 32%, 58%, 12% and 31% on MiniBooNE, Forest, Yahoo, Cifar10 and Letters datasets, respectively.

In summary, we presented an adaptive approximation approach to account for prediction costs that arise in various applications. At test-time our method uses a gating function to identify a prediction model among a collection of models that is

adapted to the input. The overall goal is to reduce costs without sacrificing accuracy. We learn gating and prediction models by means of a bottom-up strategy that trains low prediction-cost models to approximate high prediction-cost models in regions where low-cost models suffice. On a number of benchmark datasets our method leads to an average of 40% cost reduction without sacrificing test accuracy (within 1%). It outperforms state-of-the-art top-down and bottom-up budgeted learning algorithms, with a significant margin in several cases.

Chapter 6

On-line Adaptive Approximation

As discussed in Chapter 5, we aim to learn a local predictor h together with a gating function g given the remote model f , such that as many predictions as possible are made locally, sending only “difficult” examples to the remote server. In particular, for a new example x , if $g(x) \leq 0$ then it is sent to the remote model f ; otherwise it is predicted using the local model $h(x)$. In this chapter we would like to study this problem in an on-line setting.

6.1 Problem Setup

Suppose the environment generates an arbitrary sequence of examples: $(x^t, y^t) \in \mathbb{R}^m \times \{0, 1\}, t = 1, 2, \dots$. At each round, the player receives x^t , and chooses a gating function g_t together with a local predictor h_t . if $g_t(x^t) \leq 0$ (the example is sent to f), an overhead cost c is incurred and a ground truth label y^t is obtained; if $g_t(x^t) > 0$ (the example is sent to h_t), only the prediction $\hat{y}^t = h_t(x^t)$ is obtained. The loss at round t is

$$l_t(g_t, h_t) = \mathbb{1}_{[g_t(x^t) > 0]} \cdot \mathbb{1}_{[\hat{y}^t \neq y^t]} + \mathbb{1}_{[g_t(x^t) \leq 0]} \cdot c. \quad (6.1)$$

The player gets full feedback if $g_t(x^t) \leq 0$ and no feedback otherwise. Our goal is to minimize regret:

$$\mathbb{E}\left[\sum_{t=1}^T l_t(g_t, h_t)\right] - \min_{g \in \mathcal{G}, h \in \mathcal{H}} \sum_{t=1}^T l_t(g, h),$$

where the expectation is with respect to the internal randomness of the algorithm.

Discretizing g and h Suppose there are only a finite number of g and h 's as experts, denoted as $\Xi = \{\xi_j = (g_j, h_j) : j = 1, \dots, K\} \subset \mathcal{G} \times \mathcal{H}$. Then we can cast the problem as an online learning problem with feedback graph (Alon et al., 2015). The feedback graph changes in each round because of x^t . One instance of such a graph is illustrated in Figure 6.1.

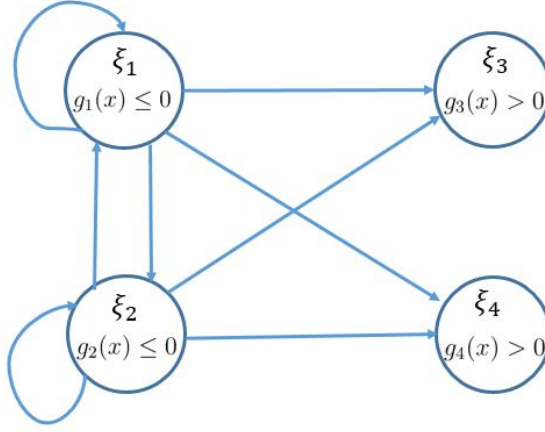


Figure 6.1: Feedback graph for four experts. ξ_1 and ξ_2 request for label and receives full feedback; ξ_3 and ξ_4 classify using h and receives no feedback.

We can apply the algorithm EXP3.G (Alon et al., 2015). The algorithm is copied here for easy reference.

Algorithm 6 EXP3.G

Input: Feedback graph $G = (V, E)$, learning rate $\eta > 0$, exploration set U , exploration rate $\gamma \in [0, 1]$.

Let u be the uniform distribution over U ;

Initialize q_1 to the uniform distribution over V ;

for round $t = 1, 2, \dots$ **do**

 Compute $p_t = (1 - \gamma)q_t + \gamma u$;

 Draw $I_t \sim p_t$, play I_t and incur loss $\ell_t(I_t)$;

 Observe $\{(i, \ell_t(i)) : i \in N^{\text{out}}(I_t)\}$;

 Update:

$$\forall i \in V, \hat{\ell}_t(i) = \frac{\ell_t(i)}{P_t(i)} \mathbf{1}_{[i \in N^{\text{out}}(I_t)]}, P_t(i) = \sum_{j \in N^{\text{in}}(i)} p_t(j)$$

$$\forall i \in V, q_{t+1}(i) = \frac{q_t(i) \exp(-\eta \hat{\ell}_t(i))}{\sum_{j \in V} q_t(j) \exp(-\eta \hat{\ell}_t(j))}$$

Remark 1: The context the player receives at each round (x_t) is used to construct the feedback graph. Once the graph is constructed, the problem turns into a non-contextual problem.

Remark 2: The discretization of g, h 's into different experts ignores the underlying relations among g, h 's.

6.2 Upper Bound

We show that the upper bound of Theorem 2 of (Alon et al., 2015) can hold even for small T 's in our setting. We assume that the feedback graph is always observable in each round. We apply EXP3.G with the exploration set in each round $U_t = \{k_t : g_{k_t}(x^t) \leq 0\}$. Note the exploration set has only one element, as the dominating number of our feedback graphs is always 1.

Theorem 6.2.1 *Let $G_t = (V, E_t)$ be the feedback graphs over the set of experts $\Xi = \{\xi_j = (g_j, h_j), j = 1, \dots, K\}$ for $t = 1, \dots, T$. Assume the graph is observable in each round. Then the expected regret against any loss sequence is $\mathcal{O}((\log K)^{1/3} T^{2/3})$ for $T \geq \log K$.*

Proof By the standard second-order regret bound of Hedge,

$$\sum_{t=1}^T \sum_{i \in V} q_t(i) \ell_t(i) - \sum_{t=1}^T \ell_t(i^*) \leq \frac{\log K}{\eta} + \eta \sum_{t=1}^T \sum_{i=1}^K q_t(i) \ell_t(i)^2. \quad (6.2)$$

We apply the loss estimates and take expectations and arrive at:

$$\mathbb{E} \left[\sum_{t=1}^T \sum_{i \in V} q_t(i) \mathbb{E}_t \hat{\ell}_t(i) \right] - \sum_{t=1}^T \mathbb{E}_t \hat{\ell}_t(i^*) \leq \frac{\log K}{\eta} + \eta \mathbb{E} \left[\sum_{t=1}^T \sum_{i=1}^K q_t(i) \mathbb{E}_t [\hat{\ell}_t(i)^2] \right]. \quad (6.3)$$

Since $\mathbb{E}_t \hat{\ell}_t(i) = \ell_t(i)$ and $\mathbb{E}_t [\hat{\ell}_t(i)^2] = \frac{\ell_t(i)^2}{P_t(i)}$, the above inequality can be written as

$$\mathbb{E} \left[\sum_{t=1}^T \sum_{i \in V} q_t(i) \ell_t(i) \right] - \sum_{t=1}^T \ell_t(i^*) \leq \frac{\log K}{\eta} + \eta \mathbb{E} \left[\sum_{t=1}^T \sum_{i=1}^K \frac{q_t(i)}{P_t(i)} \right]. \quad (6.4)$$

Since our feedback graph has only two types of nodes: full observation nodes and no observation nodes, the probability of observing the loss of expert i is equal for all $i = 1, \dots, K$ for any t . Thus, we can denote the probability of observation at t under the distribution p_t as:

$$P_t^o = \sum_{i: g_i(x^t) \leq 0} p_t(i). \quad (6.5)$$

We further have $P_t^o \geq \gamma$ because $p_t = (1 - \gamma)q_t + \gamma u_t$, where u_t is the probability concentrated on a single full observation expert k_t . Thus, we bound the regret as:

$$\mathbb{E}\left[\sum_{t=1}^T \sum_{i \in V} p_t(i) \ell_t(i)\right] - \sum_{t=1}^T \ell_t(i^*) \quad (6.6)$$

$$\leq \mathbb{E}\left[\sum_{t=1}^T \sum_{i \in V} q_t(i) \ell_t(i)\right] - \sum_{t=1}^T \ell_t(i^*) + \gamma T \quad (6.7)$$

$$\leq \frac{\log K}{\eta} + \eta \mathbb{E}\left[\sum_{t=1}^T \sum_{i=1}^K \frac{q_t(i)}{P_t^o(i)}\right] + \gamma T \quad (6.8)$$

$$\leq \frac{\log K}{\eta} + \eta \sum_{t=1}^T \frac{1}{P_t^o} + \gamma T \quad (6.9)$$

$$\leq \frac{\log K}{\eta} + \frac{\eta}{\gamma} T + \gamma T \quad (6.10)$$

$$= 3(\log K)^{1/3} T^{2/3}, \quad (6.11)$$

where the first inequality follows from $p_t \leq q_t + \gamma u_t$ and the last equality follows by setting $\eta = \gamma^2 = (\frac{\log K}{T})^{2/3}$.

Cost of full observation expert: If the cost of the full observation expert is c , then we can modify the above regret bound as:

$$\mathbb{E}\left[\sum_{t=1}^T \sum_{i \in V} p_t(i) \ell_t(i)\right] - \sum_{t=1}^T \ell_t(i^*) \quad (6.12)$$

$$\leq \mathbb{E}\left[\sum_{t=1}^T \sum_{i \in V} q_t(i) \ell_t(i)\right] - \sum_{t=1}^T \ell_t(i^*) + c\gamma T \quad (6.13)$$

$$\leq \frac{\log K}{\eta} + \eta \mathbb{E}\left[\sum_{t=1}^T \sum_{i=1}^K \frac{q_t(i)}{P_t(i)}\right] + c\gamma T \quad (6.14)$$

$$\leq \frac{\log K}{\eta} + \eta \sum_{t=1}^T \frac{1}{P_t^o} + c\gamma T \quad (6.15)$$

$$\leq \frac{\log K}{\eta} + \frac{\eta}{\gamma} T + c\gamma T. \quad (6.16)$$

Set $\eta = c^{-\frac{1}{3}}\left(\frac{\log K}{T}\right)^{2/3}$ and $\gamma = c^{-\frac{2}{3}}\left(\frac{\log K}{T}\right)^{1/3}$, then the regret upper bound becomes

$$3c^{\frac{1}{3}}(\log K)^{1/3}T^{2/3}.$$

Relation to the label efficient learning of partial monitoring Our problem is related to the partial monitoring with a revealing action (Bianchi et al., 2006). The algorithm in Figure 2 of (Bianchi et al., 2006) resembles our adaptation of EXP3.G. Furthermore, the upper bounds in Theorem m4.1 of (Bianchi et al., 2006) matches ours shown above. Although in our problem we may have many revealing actions (experts), they are all equivalent in terms of the loss and feedback they provide at any round. In other words, any one of the revealing experts can be picked for full feedback without affecting the loss or feedback info if the player wants to explore.

VC Dimension In case of hypothesis classes containing infinitely many g, h 's and finite VC dimension, we can evoke Sauer's Lemma that says the growth function $\tau(T) \leq (eT/d)^d$, where T is the number of examples and d is the VC dimension of

the hypothesis class. Thus, given the number of rounds T , the effective number of g, h 's are less than or equal to $K = (eT/d)^d$ restricted to the examples.

6.3 Lower Bound

Here we try to incorporate the $\log K$ dependency in the lower bound. Given a set of experts $\Xi = \{\xi_j = (g_j, h_j) : j = 1, \dots, K\}$, assume the environment can choose the samples such that the experts $1, \dots, K-1$ never request for the true label whereas the K th expert always requests for the true label for the rounds $t = 1, \dots, T$. We further assume the environment can generate $K-1$ loss function distributions $\mathbb{P}_j, j = 1, \dots, K-1$ such that under $\mathbb{P}_j, L_t(i) \sim \mathbf{Ber}(\mu_i)$ be a Bernoulli random variable with parameter

$$\mu_i = \begin{cases} \frac{1}{2} - \epsilon & \text{if } i = j, \\ \frac{1}{2} & \text{if } 1 \leq i \leq K-1, i \neq j \\ 1 & \text{if } i = K. \end{cases}$$

In words, the loss of each expert is i.i.d. Bernoulli random variable with the i th expert having a slightly lower mean under \mathbb{P}_i and the last expert has a deterministic loss of 1. Consider for any deterministic player algorithm, the regret lower bound is as follows:

$$\sup_{L_t(i)} \left[\sum_{t=1}^T L_t(I_t) - \min_j \sum_{t=1}^T L_t(j) \right] \geq \frac{1}{K-1} \sum_{i=1}^{K-1} \mathbb{E}_i \left[\sum_{t=1}^T L_t(I_t) - \min_j \sum_{t=1}^T L_t(j) \right] \quad (6.17)$$

$$\geq \frac{1}{K-1} \sum_{i=1}^{K-1} \left[\mathbb{E}_i \sum_{t=1}^T L_t(I_t) - \min_j \mathbb{E}_i \sum_{t=1}^T L_t(j) \right] \quad (6.18)$$

$$\geq \frac{1}{K-1} \sum_{i=1}^{K-1} \mathbb{E}_{M_i} \left[\epsilon \mathbb{E}_{\mathbb{Q}_1^{M_i}} \sum_{t=1}^T \mathbb{1}_{[I_t \neq i, I_t < K]} + \frac{1}{2} M_i \right] \quad (6.19)$$

where M_i is the number of times the true label is requested under \mathbb{P}_i and Let ϕ be the random variable with uniform distribution between 1 and $K-1$, representing the choice of i for \mathbb{P}_i . Let Z_1^M be the collection of losses for all experts for the M times

when true label is requested. Let $e = \mathbb{Q}_1^M[I_t \neq i, I_t < K]$ be the probability of making wrong prediction after observing the losses Z_1^M . Then we have the following Markov Chain: $\phi \rightarrow Z_1^M \rightarrow I_t$. By Fano's inequality we have

$$p(e) \log(K-1) + H(e) \geq H(\phi|I_t) \geq H(\phi|Z_1^M). \quad (6.20)$$

So we need to bound the conditional entropy of ϕ given the observed losses.

$$\begin{aligned} H(\phi|Z_1^M) &= H(\phi) - I(\phi; Z_1^M) \\ &\geq \log(K-1) - \frac{M}{(K-1)^2} \sum_{i=1}^{K-1} \sum_{j=1}^{K-1} \mathbb{KL}(\mathbb{P}_i(Z), \mathbb{P}_j(Z)) \\ &= \log(K-1) - \frac{M}{(K-1)^2} 2(K-2)(K-1) \mathbb{KL}(\mathbf{Ber}(1/2), \mathbf{Ber}(1/2 - \epsilon)) \\ &\geq \log(K-1) - 8M\epsilon^2. \end{aligned}$$

Combining the above with (6.20), we have a lower bound on $p(e)$, which we will use next.

Continuing with (6.19), We have the regret:

$$\frac{1}{K-1} \sum_{i=1}^{K-1} \mathbb{E}_{M_i} \left[\epsilon(T - M_i)p(e) + \frac{1}{2}M_i \right] \quad (6.21)$$

$$\geq \frac{1}{K-1} \sum_{i=1}^{K-1} \mathbb{E}_{M_i} \left[\epsilon(T - M_i) \left(1 - \frac{8\epsilon^2 M_i + 1}{\log(K-1)}\right) + \frac{1}{2}M_i \right] \quad (6.22)$$

$$\begin{aligned} &= \frac{1}{K-1} \sum_{i=1}^{K-1} \mathbb{E}_{M_i} \left[\frac{8\epsilon^3}{\log(K-1)} M_i^2 + \left(\frac{\epsilon}{\log(K-1)} - \frac{8\epsilon^3 T}{\log(K-1)} + \frac{1}{2} - \epsilon \right) M_i \right. \\ &\quad \left. + T\epsilon \left(1 - \frac{1}{\log(K-1)}\right) \right] \quad (6.23) \end{aligned}$$

$$\begin{aligned} &\geq \frac{1}{K-1} \sum_{i=1}^{K-1} \left[\frac{8\epsilon^3}{\log(K-1)} \bar{M}_i^2 + \left(\frac{\epsilon}{\log(K-1)} - \frac{8\epsilon^3 T}{\log(K-1)} + \frac{1}{2} - \epsilon \right) \bar{M}_i \right. \\ &\quad \left. + T\epsilon \left(1 - \frac{1}{\log(K-1)}\right) \right], \quad (6.24) \end{aligned}$$

where we make use of the Fano's inequality in (6.22) and re-arrange terms into a quadratic form in (6.23) and (6.24) follows because we set $\bar{M}_i = \mathbb{E}M_i$ and $\mathbb{E}[(M_i)^2] \geq (\mathbb{E}M_i)^2$.

Next, we show that no matter what the value of \bar{M}_i , (6.24) is always

$$O(T^{2/3}(\log(K-1))^{1/3}).$$

The approach we take is the following. We first minimize (6.24) with respect to \bar{M}_i and then select ϵ to match the claimed lower bound.

The minimizer of the quadratic expression inside the sum in (6.24) is

$$\bar{M}_i^* = \frac{T}{2} - \frac{\epsilon + (1/2 - \epsilon)\log(K-1)}{16\epsilon^3}.$$

(Note we do not need to consider the boundary case of $\bar{M}_i^* \leq 0$ because in that case ($M_i = 0$) the regret is large.) Substitute the above value into (6.24) we get

$$\begin{aligned} & -\frac{\frac{1}{\log(K-1)} + \log(K-1) - 2}{32\epsilon} - \epsilon^3 T^2 \frac{2}{\log(K-1)} - \frac{\log(K-1)}{128\epsilon^3} \\ & + T\epsilon \frac{1 - \frac{1}{\log(K-1)}}{2} - \frac{1 - \log(K-1)}{32\epsilon^2} + \frac{T}{4}. \end{aligned}$$

Finally, set $\epsilon = (16T)^{-1/3}(\log(K-1))^{1/3}$ and we obtain

$$\begin{aligned} & - (16T)^{1/3}(\log(K-1))^{-1/3} \left(\frac{1}{\log(K-1)} + \log(K-1) - 2 \right) / 32 \\ & + 16^{-1/3} T^{2/3} (\log(K-1))^{1/3} \left(1 - \frac{1}{\log(K-1)} \right) \\ & - 16^{2/3} T^{2/3} (\log(K-1))^{-2/3} \frac{1 - \log(K-1)}{32} \\ & = O((T^{2/3}(\log(K-1))^{1/3}). \end{aligned}$$

Chapter 7

Future Work

The first part of our future work deals with the communication cost in a distributed prediction setting, where a number of edge devices with sensors cooperate to make prediction while trying to reduce communication. The second part of our future work is to further develop the lower bound in Chapter 6 for function classes.

7.1 Distributed Prediction

Inspired by the sparsely-gated mixture of experts structure in (Shazeer et al., 2017), we propose a novel architecture designed for Edge/Cloud computing as shown in Figure 7.1.

Suppose there are K edge devices: s_1, \dots, s_K . For example $x^{(i)}$, we denote the measurements by the K edge devices as $x_1^{(i)}, \dots, x_K^{(i)}$.

Each edge device k has an embedded gating function g_k and low-cost predictor f_k . For each example $x^{(i)}$, an edge device k transmits its output $a_k^{(i)}$ to the Cloud:

$$a_k^{(i)} = \begin{cases} f_k(x^{(i)}) & \text{if } g_k(x^{(i)}) = 1, \\ 0 & \text{if } g_k(x^{(i)}) = 0. \end{cases}$$

The Cloud, having received $a_k^{(i)}$ from all edge devices, predicts using a centralized predictor f_0 to get the final prediction.

The communication savings in this setup occurs when $g_k(x^{(i)}) = 0$ and the edge device only needs to transmit a binary bit 0, saving the communication cost. Our goal is therefore to minimize prediction loss while limiting the number of “participation”

edge devices averaged over the examples.

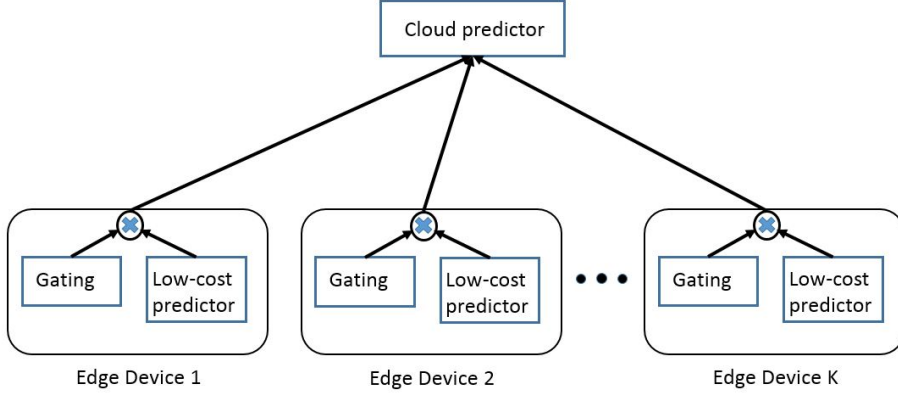


Figure 7.1: A distributed mixture of expert model for

7.1.1 Adaptive Sparse Regression

Consider the following regression problem:

$$y = \sum_{k=1}^K v_k x_k \phi(w_k x_k + b_k), \quad (7.1)$$

where $x \in \mathbb{R}^K$ is sampled from the multivariate Gaussian distribution $N(0, I_K)$ and ϕ is an activation function taking values in $[0, 1]$. We first consider the noise-free case. Given a dataset of n samples $\{(x^{(i)}, y^{(i)})\}, i = 1, \dots, n$, we would like to recover the weights v, w, b . Motivated by the budget constraint, we would like to perform the following constrained optimization with the number of active features on average to be less than or equal to B .

$$\begin{aligned} \min_{v, w, b} \quad & \mathbb{E}_{x \sim N(0, I_K)} \left(\sum_{k=1}^K v_k x_k \phi(w_k x_k + b_k) - y \right)^2 \\ \text{subject to :} \quad & \mathbb{E}_{x \sim N(0, I_K)} \sum_{k=1}^K \mathbb{1}_{[\phi(w_k x_k + b_k) > 0.5]} \leq B. \end{aligned} \quad (7.2)$$

7.1.2 Local Convexity - Hessian Computation

We show that the objective is locally convex around the ground truth parameters. Let $L(w, v) = \mathbb{E} \left[(y - \sum_{k=1}^K v_k x_k \phi(w_k x_k))^2 \right]$. Note we have absorbed the offset term into w by augmenting x with a constant 1. We can thus compute the partial derivatives:

$$\begin{aligned} \frac{\partial L(w, v)}{\partial v_i} &= \left(\sum_{k=1}^K v_k x_k \phi(w_k x_k) - y \right) x_i \phi(w_i x_i). \\ \frac{\partial L(w, v)}{\partial w_i} &= \left(\sum_{k=1}^K v_k x_k \phi(w_k x_k) - y \right) v_i x_i^2 \phi'(w_i x_i). \\ \frac{\partial L(w, v)}{\partial v_i \partial v_j} &= x_i x_j \phi(w_i x_i) \phi(w_j x_j). \\ \frac{\partial L(w, v)}{\partial w_i \partial w_j} &= x_i^2 v_i x_j^2 v_j \phi'(w_i x_i) \phi'(w_j x_j) + \left(\sum_{k=1}^K v_k x_k \phi(w_k x_k) - y \right) 1_{\{i=j\}} v_i x_i^3 \phi''(w_i x_i). \\ \frac{\partial L(w, v)}{\partial v_i \partial w_j} &= x_i x_j^2 v_j \phi(w_i x_i) \phi'(w_j x_j) + \left(\sum_{k=1}^K v_k x_k \phi(w_k x_k) - y \right) 1_{\{i=j\}} x_i^2 \phi'(w_i x_i). \end{aligned}$$

The Hessian evaluated at w^*, v^* is positive semi-definite as it can be written as an outer product form.

$$H(v^*, w^*) = \psi \psi^T,$$

where

$$\psi = \left[x_1 \phi(w_1 x_1), \dots, x_K \phi(w_K x_K), v_1 x_1^2 \phi'(w_1 x_1), \dots, v_K x_K^2 \phi'(w_K x_K) \right]^T.$$

7.1.3 Optimization

Given samples $(x^{(i)}, y^{(i)}), i = 1, \dots, n$, we can formulate the empirical optimization problem as

$$\begin{aligned} \min_{v, w, b} \quad & \sum_{i=1}^n \left(\sum_{k=1}^K v_k x_k^{(i)} \phi(w_k x_k^{(i)} + b_k) - y^{(i)} \right)^2 \\ \text{subject to :} \quad & \frac{1}{n} \sum_{i=1}^n \sum_{k=1}^K \mathbb{1}_{[w_k x_k^{(i)} + b_k > 0]} \leq B. \end{aligned} \quad (7.3)$$

We can use projected gradient descent to solve the above problem. The projection step involves

$$\begin{aligned} \min_{w, b} \quad & \sum_{k=1}^K (w_k - \hat{w}_k)^2 + (b_k - \hat{b}_k)^2 \\ \text{subject to :} \quad & \frac{1}{n} \sum_{i=1}^n \sum_{k=1}^K \mathbb{1}_{[w_k x_k^{(i)} + b_k > 0]} \leq B, \end{aligned} \quad (7.4)$$

where \hat{w} and \hat{b} are the parameters to be projected. We would like to solve the following related problem.

$$\begin{aligned} \min_z \quad & \sum_{i=1}^n \sum_{k=1}^K (z_k^{(i)} - \hat{z}_k^{(i)})^2 \\ \text{subject to :} \quad & \frac{1}{n} \sum_{i=1}^n \sum_{k=1}^K \mathbb{1}_{[z_k^{(i)} > 0]} \leq B, \end{aligned} \quad (7.5)$$

where $z_k^{(i)} = w_k x_k^{(i)} + b_k$ and $\hat{z}_k^{(i)} = \hat{w}_k x_k^{(i)} + \hat{b}_k$. The intuition behind going from (7.4) to (7.5) is that

$$\begin{aligned} \sum_{k=1}^K \sum_{i=1}^n (z_k^{(i)} - \hat{z}_k^{(i)})^2 &= \sum_{k=1}^K \sum_{i=1}^n (w_k - \hat{w}_k)^T x_k^{(i)} x_k^{(i)T} (w_k - \hat{w}_k) \\ &\quad + (b_k - \hat{b}_k)^2 - 2(b_k - \hat{b}_k)(w_k - \hat{w}_k)^T x_k^{(i)} \\ &\approx \sum_{k=1}^K (w_k - \hat{w}_k)^T (w_k - \hat{w}_k) + (b_k - \hat{b}_k)^2, \end{aligned}$$

where we assume $\mathbb{E}[x_k^{(i)} x_k^{(i)T}] = I$ and $\mathbb{E}x_k^{(i)} = 0$. Thus we hope to solve (7.5) for z 's and then recover w, b from z 's using least square regression. Note the solution to (7.5) simply involves ordering non-negative $z_k^{(i)}$'s in increasing order and setting those small ones to 0 until the constraint is satisfied.

7.1.4 Theorems to be proved

Let the ground truth parameters in (7.1) be v^*, w^*, b^* . Let the optimal solution to the population optimization (7.2) be v^{c*}, w^{c*}, b^{c*} , and the optimal objective value be opt^{c*} , where the superscript c stands for constrained. Let the optimal solution to the empirical optimization (7.3) be $v^{cs*}, w^{cs*}, b^{cs*}$, and the optimal objective value be opt^{cs*} , where the superscript cs stands for constrained and sampled optimization problem.

Our first theorem relates opt^{c*} to opt^{cs*} in a PAC learning way.

Theorem 7.1.1 *Under certain conditions, there exists $m(\epsilon, \delta) = ?$ such that when the sample size $n > m(\epsilon, \delta)$ we have $|opt^{c*} - opt^{cs*}| \leq \epsilon$ with probability at least $1 - \delta$.*

Our second theorem guarantees that the proposed projected gradient descend converges to $v^{cs*}, w^{cs*}, b^{cs*}$.

Theorem 7.1.2 *The v^t, w^t, b^t obtained after t steps of projected gradient descend for (7.3) converge to $v^{cs*}, w^{cs*}, b^{cs*}$ provided we can have a good initialization v^0, w^0, b^0 .*

To prove the above theorem, we must show that the Hessian of the objective of (7.3) around $v^{cs*}, w^{cs*}, b^{cs*}$ is PSD.

Lemma 7.1.3 *The Hessian of the objective of (7.3) around $v^{cs*}, w^{cs*}, b^{cs*}$ is PSD.*

Note in Section 7.1.2 we only showed the Hessian of the objective of (7.3) at v^*, w^*, b^* is PSD.

7.1.5 Algorithms

Direct Parameterization We can approximately solve Eq. (7.3) by directly parameterizing the gating indicator function with the sigmoid activation function and minimize the Lagrangian form:

$$\min_{v,w,a,b} \frac{1}{n} \sum_{i=1}^n \log \frac{[\exp(\sum_{k=1}^K h_k^{(i)} g_k^{(i)})]_{y^{(i)}}}{\sum_j [\exp(\sum_{k=1}^K h_k^{(i)} g_k^{(i)})]_j} + \lambda \frac{1}{nK} \sum_{i=1}^n \sum_{k=1}^K g_k^{(i)}, \quad (\text{OPT-Direct-Clf})$$

where $h_k^{(i)} = v_k^T x_k^{(i)} + a_k \in \mathfrak{R}^M$ with M being the number of classes and $g_k^{(i)} = \sigma(w_k^T x_k^{(i)} + b_k) = \frac{1}{1+\exp(-w_k^T x_k^{(i)} - b_k)} \in \mathfrak{R}$ is the activation value of the k th device for the i th example. Different budget levels are achieved by varying λ . We use stochastic gradient descend to minimize this non-convex objective.

Relaxation Rather than directly parameterizing the gating probability, we can relax it by introducing intermediate quantities $q_k^{(i)}$ and minimize the KL distance:

$$\min_{v,w,b,q} \sum_{i=1}^n \left(\sum_{k=1}^K v_k x_k^{(i)} q_k^{(i)} - y^{(i)} \right)^2 + \lambda \sum_{i=1}^n \sum_{k=1}^K \mathbb{KL}(q_k^{(i)}, \phi(w_k x_k^{(i)} + b_k)). \quad (\text{OPT-Relax-Reg})$$

For classification problems, the objective becomes

$$\min_{v,w,b,q} \sum_{i=1}^n \log \frac{[\exp(\sum_{k=1}^K h_k^{(i)} q_k^{(i)})]_{y^{(i)}}}{\sum_j [\exp(\sum_{k=1}^K h_k^{(i)} q_k^{(i)})]_j} + \lambda \sum_{i=1}^n \sum_{k=1}^K \mathbb{KL}(q_k^{(i)}, \phi(w_k x_k^{(i)} + b_k)), \quad (\text{OPT-Relax-Clf})$$

where $h_k^{(i)} = v_k^T x_k^{(i)} + a_k$. The constraints on $q_k^{(i)}$ are the following: $0 \leq q_k^{(i)} \leq 1, \forall i, k$ and $\frac{1}{nK} \sum_{i=1}^n \sum_{k=1}^K q_k^{(i)} \leq B$.

We can perform alternating minimization over v, w, b and q to minimize the above objective. To solve for q 's, we use the Frank-Wolfe algorithm due to the simplicity of the constraint set. In particular, let $\frac{\partial L}{\partial q_k^{(i)}}$ be the gradient of the loss ((OPT-Relax-Reg)

or (OPT-Relax-Clf)) with respect to $q_k^{(i)}$. Starting from an initial q^0 , the Frank-Wolfe algorithm chooses the point \hat{q}^1 in the constraint set to minimize the inner product between $\frac{\partial L}{\partial q}$ and \hat{q}^1 . The next q is updated according to $q^1 = \eta q^0 + (1 - \eta)\hat{q}^1$. This process continues until convergence. Note the \hat{q} 's can be efficiently solved by sorting the elements of $\frac{\partial L}{\partial q}$ in ascending order and setting the top $\text{Floor}(BnK)$ elements to 1, the $\text{Floor}(BnK) + 1$ th element to $BnK - \text{Floor}(BnK)$ and the rest to 0.

7.1.6 Experiments

Adaptive Sparse Regression

We first test the direct minimization via SGD of the adaptive sparse regression formulation without the budget constraint. The purpose is to test the parameter recovery ability of our algorithm with different activation functions and different initializations.

Data generation We generate $n = 2000$ sample data points $x^{(i)}, i = 1, \dots, n$, each consists of features from $K = 2$ devices and each device has 2 dimensional features: $x_k^{(i)} \in \mathbb{R}^2, k = 1, 2$. The features are generated from an i.i.d. standard normal distribution. We then generate the ground truth parameters V, W, b with each element drawn from i.i.d. standard normal distribution. Finally, we generate the regression targets based on:

$$y^{(i)} = \sum_{k=1}^K v_k x_k^{(i)} \phi(w_k x_k^{(i)} + b_k)$$

Initialization We perform iterative algorithm to minimize the objective with different initialization of V, W, b by simply adding a scaled version of Gaussian noise to the ground truth parameters: $V_0 = V + \text{noise level} \times N_V$, where noise level is a scalar value controlling the magnitude of the noise and N_V is drawn from standard normal distribution with the same dimension as V . Likewise we initialize W_0 and b_0 .

Algorithm We perform alternating minimization of the following objective.

$$\min_{v,w,b} \sum_{i=1}^n \left(\sum_{k=1}^K v_k x_k^{(i)} \phi(w_k x_k^{(i)} + b_k) - y^{(i)} \right)^2$$

, where we have a choice of using ReLu or Sigmoid activation function for ϕ . The algorithm proceeds as follows: first fix W, b , perform 50 steps of gradient descend for V ; then fix V and perform the same number of gradient descend steps for W, b ; repeat for 1000 times.

To summarize the result shown in Figure 7-2, we observe the following:

- Decreasing initialization noise for Sigmoid shifts the loss curve down along the y-axis. Ref. subplots (a,b,c).
- Decreasing initialization noise for ReLU does not impact the convergence of loss. This indicates that ReLU is more robust for loss minimization. Ref. subplots(g,h,i).
- The parameter difference in the Cosine metric show that ReLU activation converges much faster to the ground truth parameters than Sigmoid activation. Ref. subplots(j,k,l,m).

Alt-Min Vs Direct Parameterization

Next, we compare the direct and the relaxed parameterizations in classification tasks. We test on the Wearable Action Recognition Database (WARD) (Yang et al., 2009). A human subject wears 5 wireless motion sensors on different parts of the body and performs 13 types of activities including walking, lying down, etc.. Each sensor records a time series of 5-dimensional readings from a triaxial accelerator and a biaxial gyroscope. Each example in the dataset consists of readings from these sensors in a 8-time-step window and a label indicating one of the 13 activities.

On the WARD activity recognition dataset, we apply the alternating minimization algorithm with the parameters: 1) batch size: 100, 2) learning rate for ADAM in minimizing w, v, a, b : $1e-3$, 3) number of epochs in minimizing w, v, a, b : 50, 4) number of iterations in minimizing q : 20, 5) number of alternating steps: 30, 6) step size in minimizing q : $1e-2$, 7) KL term weight α : $1e-1$, 8) budget: 0.26. We obtain the following result:

```
e_update mean(q) 0.2607, max 1.0000, min 0.0000
Alt-min iter 29
Epoch 0, train loss 1.8263, train acc 0.8323, test acc 0.7006
Epoch 5, train loss 1.6878, train acc 0.8256, test acc 0.7480
Epoch 10, train loss 1.6693, train acc 0.8285, test acc 0.7284
Epoch 15, train loss 1.6641, train acc 0.8159, test acc 0.7232
Epoch 20, train loss 1.6619, train acc 0.8122, test acc 0.7133
Epoch 25, train loss 1.6612, train acc 0.8108, test acc 0.7219
Epoch 30, train loss 1.6606, train acc 0.8098, test acc 0.7133
Epoch 35, train loss 1.6603, train acc 0.8185, test acc 0.7178
Epoch 40, train loss 1.6597, train acc 0.8137, test acc 0.7235
Epoch 45, train loss 1.6595, train acc 0.8055, test acc 0.7164
e_update mean(q) 0.2606, max 1.0000, min 0.0000
```

whereas direct parameterization gives a test accuracy of 0.8 and training accuracy of 0.979 with the test budget of 0.254. Preliminary experiments show that the direct parameterization achieves higher accuracy than the relaxed parameterization for the same budget.

Using ReLU activation, the direct parameterization approach gives the following cost-accuracy trade-off:

```
alpha = 0:
train loss 0.0259, train accu 0.9930, budget 0.9034, test accu 0.822069
alpha = 0.1:
train loss 0.1476, train accu 0.9840, budget 0.7451, test accu 0.824594
alpha = 1:
```

train loss 0.3534, train accu 0.9615, budget 0.6124, test accu 0.796326
 alpha = 2:
 train loss 0.4767, train accu 0.9450, budget 0.4844, test accu 0.761428

The future direction is to adjust the architecture and the algorithm so that we can improve this trade-off. Our baseline target is the 1-nearest neighbor method that achieves the test accuracy of 0.662226, 0.669200, 0.729655, 0.786623, 0.805743, respectively using individual sensors and 0.8865 using majority vote of all the sensors.

7.2 Extending the Regret Lower Bound

Here we aim to derive a regret lower bound when the gating function g and the local predictor h come from function classes \mathcal{G} and \mathcal{H} , respectively. Let \mathcal{Z} denote the space of examples $\mathcal{X} \times \mathcal{Y}$.

We would like to show that there exists an outcome space \mathcal{Z} , a loss function $\ell : \mathcal{G} \times \mathcal{H} \times \mathcal{Z} \rightarrow [0, 1]$, such that we can lower bound the cumulative expected regret:

$$\sup_{z_1, \dots, z_n \in \mathcal{Z}} \left(\mathbf{E} \left(\sum_{t=1}^n \ell(g_t, h_t, z_t) \right) - \min_{g \in \mathcal{G}, h \in \mathcal{H}} \sum_{t=1}^n \ell(g, h, z_t) \right) \quad (7.6)$$

To construct a probability distribution over \mathcal{Z} , we first choose n distinct data points $X^n = \{x^1, \dots, x^n\} \subset \mathcal{X}$. For each $x \in \mathcal{X}$, a (g, h) -pair produces a prediction from the set $\{0, 1, r\}$; r means g decides to send x to the cloud whereas 0 or 1 means g decides to use the local predictor h which then gives the binary prediction. For a fixed set of examples X^n , we use $S \in \{r, 0, 1\}^{N \times n}$ to denote the pattern matrix where N is the number of different patterns $g \in \mathcal{G}$ and $h \in \mathcal{H}$ can produce on X^n .

For each pattern $i = 1, \dots, N$, we define the probability distribution P_i over $\mathcal{X} \times \mathcal{Y}$ as follows. x_t is drawn from X^n uniformly at random: $x_t = x^{J_t}$, $J_t \sim$

uniform $[1, \dots, n]$. The label y_t is such that if $S_{i,J_t} = r$ then

$$Y_t = \begin{cases} 1 & \text{with prob. } c + \epsilon, \\ 0 & \text{with prob. } 1 - c - \epsilon. \end{cases} \quad (7.7)$$

If $S_{i,J_t} = 1$ then

$$Y_t = \begin{cases} 1 & \text{with prob. } 1 - c + \epsilon, \\ 0 & \text{with prob. } c - \epsilon. \end{cases} \quad (7.8)$$

If $S_{i,J_t} = 0$ then

$$Y_t = \begin{cases} 1 & \text{with prob. } c - \epsilon, \\ 0 & \text{with prob. } 1 - c + \epsilon. \end{cases} \quad (7.9)$$

Having defined the distributions P_i , we have

$$\sup_{z_1, \dots, z_n \in \mathcal{Z}} \left(\mathbf{E}_A \left(\sum_{t=1}^n \ell(g_t, h_t, z_t) \right) - \min_{g \in \mathcal{G}, h \in \mathcal{H}} \sum_{t=1}^n \ell(g, h, z_t) \right) \quad (7.10)$$

$$\geq \max_{i=1, \dots, N} \mathbf{E}_i \left(\mathbf{E}_A \left(\sum_{t=1}^n \ell(g_t, h_t, Z_t) \right) - L_{i,n} \right) \quad (7.11)$$

$$= \max_{i=1, \dots, N} \sum_{d=1}^D \alpha_d \mathbf{E}_i \left(\sum_{t=1}^n \sum_{k=1}^N \mathbb{1}_{[I_t^d=k|Z_1^{t-1}]} \ell(k, Z_t) - L_{i,n} \right) \quad (7.12)$$

$$(7.13)$$

Under P_i , for any j th example, if the prediction is different from $S_{i,j}$, an expected regret of at least ϵ is incurred. Since the patterns differ from each other for at least some $x \in X^n$, and x is chosen uniformly at random, whenever $I_t^d \neq i$, the expected regret is at least ϵ/n .

Define the error set as

$$e_i = \{k | \text{Hamming}(S_i, S_k) > \delta n\}, \quad (7.14)$$

where S_i is the i th row of S . Then in the event $I_t^d \in e_i$, the expected regret is at least $\frac{\epsilon}{n}\delta n = \epsilon\delta$.

$$\max_{i=1,\dots,N} \sum_{d=1}^D \alpha_d \mathbf{E}_i \left(\sum_{t=1}^n \sum_{k=1}^N \mathbb{1}_{[I_t^d=k|Z_1^{t-1}]} \ell(k, Z_t) - L_{i,n} \right) \quad (7.15)$$

$$\geq \max_{i=1,\dots,N} \sum_{d=1}^D \alpha_d \sum_{t=1}^n \epsilon \delta P_i[I_t^d \in e_i] \quad (7.16)$$

$$\geq \sum_{d=1}^D \alpha_d \sum_{t=1}^n \epsilon \delta \mathbf{E}_i P_i[I_t^d \in e_i] \quad (7.17)$$

$$\geq \sum_{d=1}^D \alpha_d \sum_{t=1}^n \epsilon \delta \left(1 - \frac{I(\Phi; Z_1^M) + \log 2}{\log \frac{N}{N_t^{\max}}} \right), \quad (7.18)$$

where Φ denotes the random choice of the patterns and the last inequality comes from Duchi's note.

We can upper bound the mutual information as

$$I(\Phi; Y_1^M) \leq \frac{M}{(K-1)^2} \sum_{i=1}^{K-1} \sum_{j=1}^{K-1} \mathbb{KL}(\mathbb{P}_i(Z), \mathbb{P}_j(Z)) \quad (7.19)$$

$$\leq \frac{M}{(K-1)^2} 2(K-2)(K-1) \mathbb{KL}(\mathbf{Ber}(c-\epsilon), \mathbf{Ber}(1-c+\epsilon)) \quad (7.20)$$

7.2.1 Non-uniform sampling of patterns

We can arrange the N patterns into two groups: $i = 1, \dots, N_1$ corresponding to the patterns with no r appearing in them; $i = N_1 + 1, \dots, N$ corresponding to the patterns

with at least one r in them.

$$\begin{aligned}
& \sup_{z_1, \dots, z_n \in \mathcal{Z}} \left(\mathbf{E}_A \left(\sum_{t=1}^n \ell(g_t, h_t, z_t) \right) - \min_{g \in \mathcal{G}, h \in \mathcal{H}} \sum_{t=1}^n \ell(g, h, z_t) \right) \\
& \geq c \max_{i=1, \dots, N_1} \mathbf{E}_i \left(\mathbf{E}_A \left(\sum_{t=1}^n \ell(g_t, h_t, Z_t) \right) - L_{i,n} \right) \\
& + (1-c) \max_{i=N_1+1, \dots, N} \mathbf{E}_i \left(\mathbf{E}_A \left(\sum_{t=1}^n \ell(g_t, h_t, Z_t) \right) - L_{i,n} \right).
\end{aligned}$$

The idea is that if $c = 1$, then the regret suffered should be at least that of learning the local predictor h . If $c = 0$, the regret should be 0 as the optimal action is to request label for all the examples. Following previous analysis, the second term is lower-bounded by

$$(1-c) \sum_{d=1}^D \alpha_d \sum_{t=1}^n \epsilon \delta \left(1 - \frac{I(\Phi; Z_1^M) + \log 2}{\log \frac{N-N_1}{N_t^{\max}}} \right) \quad (7.21)$$

$$\geq (1-c) \sum_{d=1}^D \alpha_d \sum_{t=1}^n \epsilon \delta \left(1 - \frac{2M\mathbb{KL}(\mathbf{Ber}(c-\epsilon), \mathbf{Ber}(1-c+\epsilon)) + \log 2}{\log \frac{N-N_1}{N_t^{\max}}} \right). \quad (7.22)$$

As for the first term, we can re-define the probability distribution P_i over $\mathcal{X} \times \mathcal{Y}$ as follows. x_t is drawn from X^n uniformly at random: $x_t = x^{J_t}$, $J_t \sim \text{uniform}[1, \dots, n]$. The label y_t is such that if $S_{i,J_t} = 1$ then

$$Y_t = \begin{cases} 1 & \text{with prob. } 1/2 + \epsilon, \\ 0 & \text{with prob. } 1/2 - \epsilon. \end{cases} \quad (7.23)$$

If $S_{i,J_t} = 0$ then

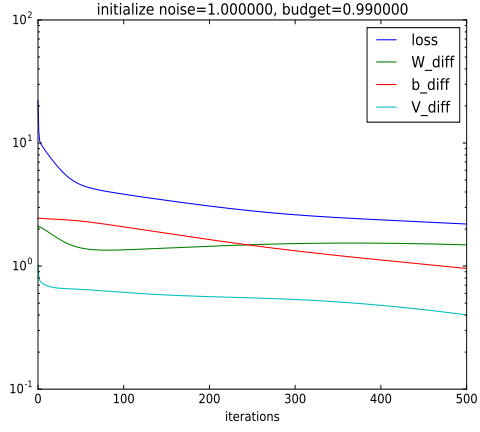
$$Y_t = \begin{cases} 1 & \text{with prob. } 1/2 - \epsilon, \\ 0 & \text{with prob. } 1/2 + \epsilon. \end{cases} \quad (7.24)$$

Similar to the previous analysis, the first term can be bounded by

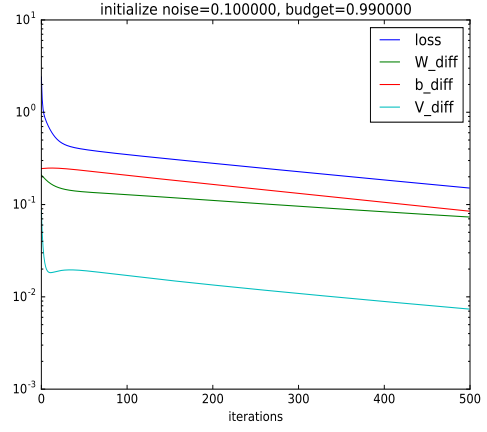
$$c \sum_{d=1}^D \alpha_d \sum_{t=1}^n \epsilon \delta \left(1 - \frac{I(\Phi; Z_1^M) + \log 2}{\log \frac{N_1}{N_t^{\max}}} \right) \quad (7.25)$$

$$\geq c \sum_{d=1}^D \alpha_d \sum_{t=1}^n \epsilon \delta \left(1 - \frac{2M\mathbb{KL}(\mathbf{Ber}(1/2 - \epsilon), \mathbf{Ber}(1/2 + \epsilon)) + \log 2}{\log \frac{N_1}{N_t^{\max}}} \right). \quad (7.26)$$

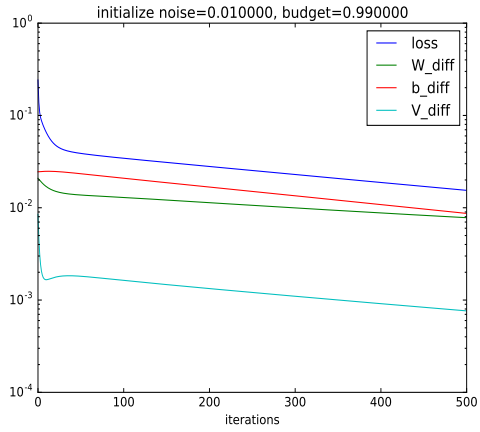
We would like to further simplify the expression so that the lower bound is in terms of VC dimensions of \mathcal{G}, \mathcal{H} .



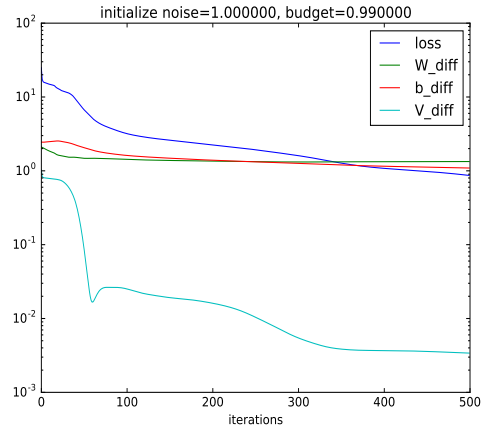
(a) Sig 1, noise 1, ss 1e-2



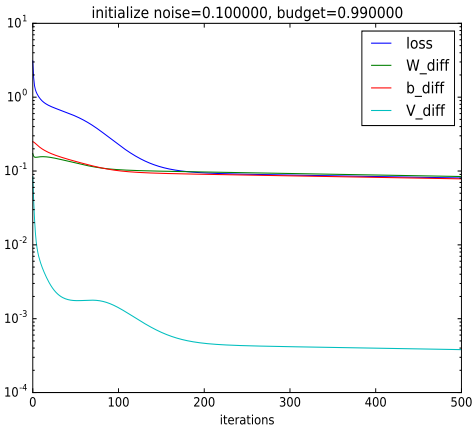
(b) Sig 1, noise 0.1, ss 1e-2



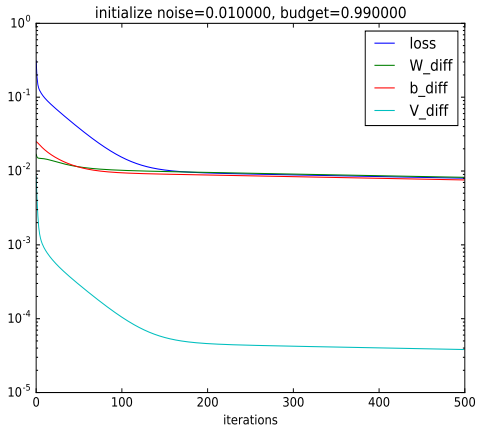
(c) Sig 1, noise 0.01, ss 1e-2



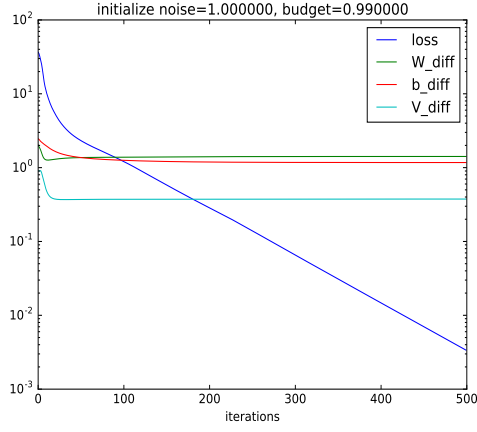
(d) Sig 10, noise 1, ss 1e-2



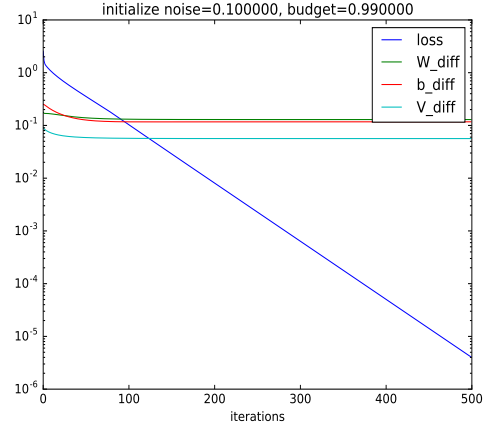
(e) Sig 10, noise 0.1, ss 1e-2



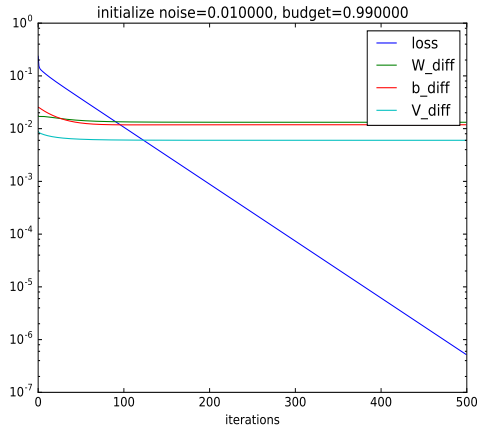
(f) Sig 10, noise 0.01, ss 1e-2



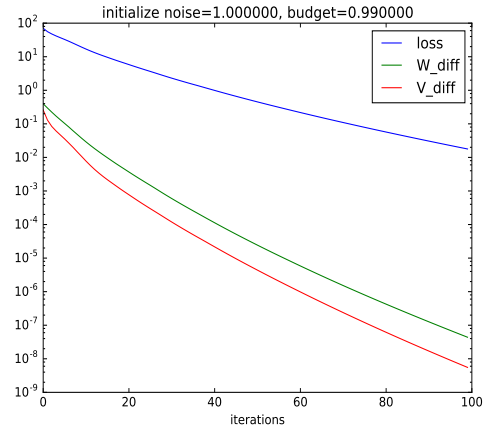
(g) ReLu, noise 1, ss 1e-2



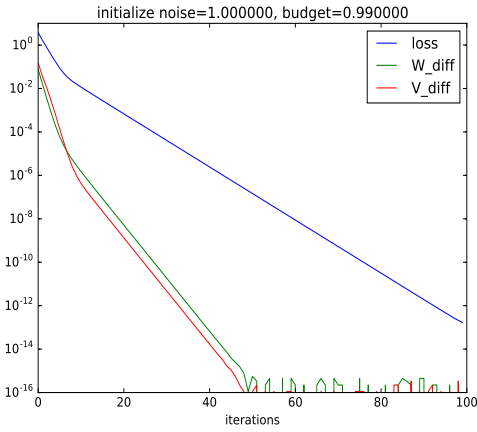
(h) ReLu, noise 0.1, ss 1e-2



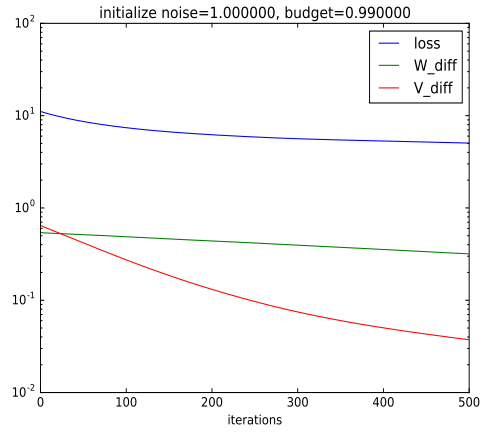
(i) ReLu, noise 0.01, ss 1e-2



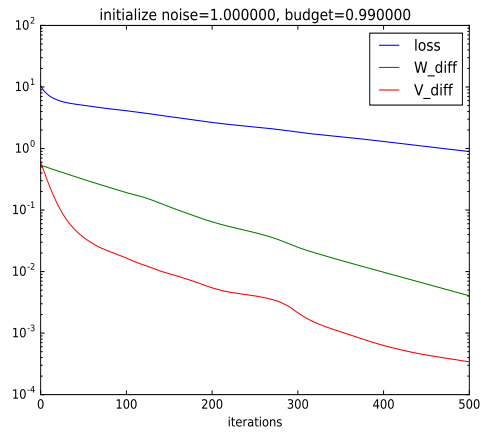
(j) ReLu, noise 1, ss 1e-4, 100 dim, Cos



(k) ReLu, noise 1, ss 5e-3, 10 dim, Cos



(l) Sig 1, noise 1, ss 1e-4, 100 dim, Cos



(m) Sig 1, noise 1, ss 1e-3, 100 dim, Cos

Figure 7.2: Sigmoid parameter is the constant multiplier inside the exponential of the sigmoid function; noise is the noise level; ss is the step size

Chapter 8

Conclusions

In this thesis we proposed four learning methods for the problem of budgeted prediction. The two methods presented in Chapters 2 and 3 follow a top-down approach, where the most cost-effective features are acquired sequentially to improve accuracy. The two methods presented in Chapters 4 and 5 form a new paradigm of bottom-up approach. These methods start with a high accuracy model as initialization and then reduce prediction cost while maintaining accuracy. The benefits of the bottom-up approach are as follows.

- Computationally it circumvents the difficulty of multi-stage search over all feature subsets faced by the top-down approach. In the RF pruning method we showed that the global optimization problem can be efficiently solved; in the adaptive approximation method we showed that the original non-convex problem can be relaxed via alternating minimization where each step is solving a convex problem.
- Statistically it takes advantage of a good initialization. In some cases the best accuracy of top-down methods cannot match that of the bottom-up methods because the former are restricted by the model choices whereas the latter can have much more general initialization models.

We also studied adaptive approximation in the on-line setting and provide matching upper and lower bounds for the regret. For each method we empirically show that it

improves the state-of-the-art performance.

For the future work, we aim to study the computational and statistical properties of the distributed prediction model. Also, we aim to extend the regret lower bound in the on-line adaptive approximation problem to accommodate the fact that gating and low-cost models are chosen from some function classes.

Appendix A

Appendix

A.1 ADAPT-LSTSQ for Chapter 5

Other Symmetric metrics: KL divergence is not symmetric and leads to widely different properties in terms of approximation. We also consider a symmetric metric:

$$D(r(z), s(z)) = \left(\log \frac{r(0)}{r(1)} - \log \frac{s(0)}{s(1)} \right)^2$$

This metric can be viewed intuitively as a regression of

$$g(x) = \log \left(\frac{\Pr(1|g; x)}{\Pr(0|g; x)} \right)$$

against the observed log odds ratio of $q(z|x)$.

The main advantage of using KL is that optimizing w.r.t. q can be solved in closed form. The disadvantage we observe is that in some cases, the loss for minimizing w.r.t. g , which is a weighted sum of log-losses of opposing directions, becomes quite flat and difficult to optimize especially for linear gating functions. The symmetric measure, on the other hand, makes the optimization w.r.t. g better conditioned as the gating function g fits directly to the log odds ratio of q . However, the disadvantage of using the symmetric measure is that optimizing w.r.t. q no longer has closed form solution; furthermore, it is even non-convex. We offer an ADMM approach for q optimization.

We still follow an alternating minimization approach. To keep the presentation simply, we assume g, f_1 to be linear classifiers and there is no feature costs involved.

Algorithm 7 ADAPT-LSTSQ

Input: $(x^{(i)}, y^{(i)}), B$
 Train a full accuracy model f_0 .
 Initialize g, f_1 .
repeat
 Solve (OPT5) for q given g, f_1 .
 Solve (OPT6) for g given q .
 Solve (OPT7) for f_1 given q .
until convergence

To minimize over q , we must solve

$$\begin{aligned} \min_{q_i \in [0,1]} \quad & \frac{1}{N} \sum_{i=1}^N \left[(1 - q_i) A_i + \left(\log \frac{q_i}{1 - q_i} - g(x^{(i)}) \right)^2 \right] \\ \text{s.t.} \quad & \frac{1}{N} \sum_{i=1}^N q_i \leq P_{\text{full}}, \end{aligned} \tag{OPT5}$$

where $q_i = q(z = 0|x^{(i)})$, $A_i = \log(1 + e^{-y^{(i)} f_1^T x^{(i)}}) + \log p(y^{(i)}|z^{(i)} = 1; f_0)$. Unlike (OPT3), this optimization problem no longer has a closed-form solution. Fortunately, the q_i 's in the objective are decoupled and there is only one coupling constraint. We can solve this problem using an ADMM approach (Boyd et al., 2011). To optimize over g , we simply need to solve a linear least squares problem:

$$\min_g \frac{1}{N} \sum_{i=1}^N \left(\log \frac{q_i}{1 - q_i} - g^T(x^{(i)}) \right)^2. \tag{OPT6}$$

To optimize over f_1 , we solve a weighted logistic regression problem:

$$\min_{f_1} \frac{1}{N} \sum_{i=1}^N (1 - q_i) \log(1 + e^{-y^{(i)} f_1^T x^{(i)}}). \tag{OPT7}$$

We shall call the above algorithm ADAPT-LSTSQ, summarized in Algorithm 7. However, on a number of datasets, we found that ADAPT-LSTSQ is comparable to ADAPT-GBRT thus we did not include it in the main plots.

A.2 Experimental Details for Chapter 5

We provide detailed parameter settings and steps for our experiments here.

A.2.1 Synthetic-1 Experiment

We generate the data in Python using the following command:

```
X, y = make_classification(n_samples=1000, flip_y=0.01, n_features=2,
n_redundant=0, n_informative=2, random_state=17, n_clusters_per_class=2)
```

For ADAPT-GBRT we used 5 depth-2 trees for g and f_1 .

A.2.2 Synthetic-2 Experiment:

We generate 4 clusters on a 2D plane with centers: (1,1), (-1,1), (-1,-1), (-1, -3) and Gaussian noise with standard deviation of 0.01. The first two clusters have 20 examples each and the last two clusters have 15 examples each. We sweep the regularization parameter of L1-regularized logistic regression and recover feature 1 as the sparse subset, which leads to sub-optimal adaptive system. On the other hand, we can easily train a RBF SVM classifier to correctly classify all clusters and we use it as f_0 . If we initialize g and f_1 with Gaussian distribution centered around 0, ADAPT-LIN with can often recover feature 2 as the sparse subset and learn the correct g and f_1 . Or, we could initialize $g = (1, 1)$ and $f_1 = (1, 1)$ then ADAPT-LIN can recover the optimal solution.

A.2.3 Letters Dataset (Dheeru and Karra Taniskidou, 2017)

This letters recognition dataset contains 20000 examples with 16 features, each of which is assigned unit cost. We binarized the labels so that the letters before "N" is class 0 and the letters after and including "N" are class 1. We split the examples 12000/4000/4000 for training/validation/test sets. We train RBF SVM and RF (500

trees) with cross-validation as f_0 . RBF SVM achieves the higher accuracy of 0.978 compared to RF 0.961.

To run the greedy algorithm, we first cross validate L1-regularized logistic regression with 20 C parameters in logspace of $[1e-3, 1e1]$. For each C value, we obtain a classifier and we order the absolute values of its components and threshold them at different levels to recover all 16 possible supports (ranging from 1 feature to all 16 features). We save all such possible supports as we sweep C value. Then for each of the supports we have saved, we train a L2-regularized logistic regression only based on the support features with regularization set to 1 as f_1 . The gating g is then learned using L2-regularized logistic regression based on the same feature support and pseudo labels of $f_1 - 1$ if it is correctly classified and 0 otherwise. To get different cost-accuracy tradeoff, we sweep the class weights between 0 and 1 so as to influence g to send different fractions of examples to the f_0 .

To run ADAPT-LIN, we initialize g to be 0 and f_1 to be the output of the L2-regularized logistic regression based on all the features. We then perform the alternative minimization for 50 iterations and sweep γ between $[1e-4, 1e0]$ for 20 points and P_{full} in $[0.1, 0.9]$ for 9 points.

To run ADAPT-GBRT, we use 500 depth 4 trees for g and f_1 each. We initialize g to be 0 and f_1 to be the GreedyMiser output of 500 trees. We then perform the alternative minimization for 30 iterations and sweep γ between $[1e-1, 1e2]$ for 10 points in logspace and P_{full} in $[0.1, 0.9]$ for 9 points. In addition, we also sweep the learning rate for GBRT for 9 points between $[0.1, 1]$.

For fair comparison, we run GREEDYMISER with 1000 depth 4 trees so that the model size matches that of ADAPT-GBRT. The learning rate is swept between $[1e-5, 1]$ with 20 points and the λ is swept between $[0.1, 100]$ with 20 points.

Finally, we evaluate all the resulting systems from the parameter sweeps of all

the algorithms on validation data and choose the efficient frontier and use the corresponding settings to evaluate and plot the test performance.

A.2.4 MiniBooNE Particle Identification and Forest Covertypes Datasets (Dheeru and Karra Taniskidou, 2017):

The MiniBooNE data set is a binary classification task to distinguish electron neutrinos from muon neutrinos. There are 45523/19510/65031 examples in training/validation/test sets. Each example has 50 features, each with unit cost. The Forest data set contains cartographic variables to predict 7 forest cover types. There are 36603/15688/58101 examples in training/validation/test sets. Each example has 54 features, each with unit cost.

We use the unpruned RF of BUDGETPRUNE (Nan et al., 2016) as f_0 (40 trees for both datasets.) The settings for ADAPT-GBRT are the following. For MiniBooNE we use 100 depth 4 trees for g and f_1 each. We initialize g to be 0 and f_1 to be the GreedyMiser output of 100 trees. We then perform the alternative minimization for 50 iterations and sweep γ between $[1e-1, 1e2]$ for 20 points in logspace and P_{full} in $[0.1, 0.9]$ for 9 points. In addition, we also sweep the learning rate for GBRT for 9 points between $[0.1, 1]$. For Forest we use 500 depth 4 trees for g and f_1 each. We initialize g to be 0 and f_1 to be the GreedyMiser output of 500 trees. We then perform the alternative minimization for 50 iterations and sweep γ between $[1e-1, 1e2]$ for 20 points in logspace and P_{full} in $[0.1, 0.9]$ for 9 points. In addition, we also sweep the learning rate for GBRT for 9 points between $[0.1, 1]$.

For fair comparison, we run GREEDYMISER with 200 depth 4 trees so that the model size matches that of ADAPT-GBRT for MiniBooNE. We run GREEDYMISER with 1000 depth 4 trees so that the model size matches that of ADAPT-GBRT for Forest.

Finally, we evaluate all the resulting systems from the parameter sweeps on val-

validation data and choose the efficient frontier and use the corresponding settings to evaluate and plot the test performance.

A.2.5 Yahoo! Learning to Rank(Chapelle et al., 2011):

This ranking dataset consists of 473134 web documents and 19944 queries. Each example is associated with features of a query-document pair together with the relevance rank of the document to the query. There are 519 such features in total; each is associated with an acquisition cost in the set $\{1, 5, 20, 50, 100, 150, 200\}$, which represents the units of CPU time required to extract the feature and is provided by a Yahoo! employee. The labels are binarized into relevant or not relevant. The task is to learn a model that takes a new query and its associated documents and produce a relevance ranking so that the relevant documents come on top, and to do this using as little feature cost as possible. The performance metric is Average Precision @ 5 following (Nan et al., 2016).

We use the unpruned RF of BUDGETPRUNE (Nan et al., 2016) as f_0 (140 trees for both datasets.) The settings for ADAPT-GBRT are the following. we use 100 depth 4 trees for g and f_1 each. We initialize g to be 0 and f_1 to be the GREEDYMISER output of 100 trees. We then perform the alternative minimization for 20 iterations and sweep γ between $[1e-1, 1e3]$ for 30 points in logspace and P_{full} in $[0.1, 0.9]$ for 9 points. In addition, we also sweep the learning rate for GBRT for 9 points between $[0.1, 1]$.

For fair comparison, we run GREEDYMISER with 200 depth 4 trees so that the model size matches that of ADAPT-GBRT for Yahoo.

Finally, we evaluate all the resulting systems from the parameter sweeps on validation data and choose the efficient frontier and use the corresponding settings to evaluate and plot the test performance.

A.2.6 CIFAR10 (Krizhevsky, 2009):

CIFAR-10 data set consists of 32x32 colour images in 10 classes. 400 features for each image are extracted using technique described in (Coates and Ng, 2011). The data are binarized by combining the first 5 classes into one class and the others into the second class. There are 19,761/8,468/10,000 examples in training/validation/test sets. BUDGETPRUNE starts with a RF of 40 trees, which achieves an accuracy of 69%. We use an RBF-SVM as f_0 that achieves a test accuracy of 79.5%. The settings for ADAPT-GBRT are the following. we use 200 depth 5 trees for g and f_1 each. We initialize g to be 0 and f_1 to be the GREEDYMISER output of 200 trees. We then perform the alternative minimization for 50 iterations and sweep γ between $[1e-4, 10]$ for 15 points in logspace and P_{full} in $[0.1, 0.9]$ for 9 points. In addition, we also sweep the learning rate for GBRT for 10 points between $[0.01, 1]$.

For fair comparison, we run GREEDYMISER with 400 depth 5 trees so that the model size matches that of ADAPT-GBRT.

Finally, we evaluate all the resulting systems from the parameter sweeps on validation data and choose the efficient frontier and use the corresponding settings to evaluate and plot the test performance.

References

- (2010). IBM ILOG CPLEX Optimizer.
<http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/>.
- Alon, N., Cesa-Bianchi, N., Dekel, O., and Koren, T. (2015). Online learning with feedback graphs: Beyond bandits. In *Proceedings of The 28th Conference on Learning Theory, COLT 2015, Paris, France, July 3-6, 2015*, pages 23–35.
- Bellala, G., Bhavnani, S. K., and Scott, C. (2012). Group-based active query selection for rapid diagnosis in time-critical situations. *IEEE Transactions on Information Theory*, 58(1):459–478.
- Benbouzid, D. (2014). *Sequential prediction for budgeted learning : Application to trigger design*. Theses, Université Paris Sud - Paris XI.
- Bianchi, N. C., Lugosi, G., and Stoltz, G. (2006). Regret Minimization Under Partial Monitoring. *Mathematics of Operations Research*, 31:562–580.
- Bilgic, M. and Getoor, L. (2007). Voila: Efficient feature-value acquisition for classification. In *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 2, AAAI’07*, pages 1225–1230. AAAI Press.
- Bolukbasi, T., Wang, J., Dekel, O., and Saligrama, V. (2017). Adaptive neural networks for efficient inference. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 527–536, International Convention Centre, Sydney, Australia. PMLR.
- Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122.
- Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.
- Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. CRC press.
- Busa-Fekete, R., Benbouzid, D., and Kégl, B. (2012). Fast classification using sparse decision dags. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*.

- Chakaravarthy, V. T., Pandit, V., Roy, S., Awasthi, P., and Mohania, M. K. (2011). Decision trees for entity identification: Approximation algorithms and hardness results. *ACM Transaction on Algorithms*, 7(2):15:1–15:22.
- Chapelle, O., Chang, Y., and Liu, T., editors (2011). *Proceedings of the Yahoo! Learning to Rank Challenge, held at ICML 2010, Haifa, Israel, June 25, 2010*.
- Chen, M., Xu, Z. E., Weinberger, K. Q., Chapelle, O., and Kedem, D. (2012). Classifier cascade for minimizing feature evaluation cost. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2012, La Palma, Canary Islands, April 21-23, 2012*, pages 218–226.
- Cicalese, F., Laber, E. S., and Saettler, A. M. (2014). Diagnosis determination: decision trees optimizing simultaneously worst and expected testing cost. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China*.
- Coates, A. and Ng, A. G. (2011). The importance of encoding versus training with sparse coding and vector quantization. In *Proceedings of the 28th International Conference on Machine Learning*. ACM.
- Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.
- Cover, T. M. and Thomas, J. A. (1991). *Elements of Information Theory*. Wiley-Interscience, New York, NY, USA.
- Dheeru, D. and Karra Taniskidou, E. (2017). UCI machine learning repository. <http://archive.ics.uci.edu/ml>, University of California, Irvine, School of Information and Computer Sciences.
- Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119 – 139.
- Friedman, J. H. (2000). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232.
- Ganchev, K., Taskar, B., and Gama, J. (2008). Expectation maximization and posterior constraints. In Platt, J. C., Koller, D., Singer, Y., and Roweis, S. T., editors, *Advances in Neural Information Processing Systems 20*, pages 569–576. Curran Associates, Inc.
- Gao, T. and Koller, D. (2011). Active classification based on value of classifier. In *Advances in Neural Information Processing Systems (NIPS 2011)*.

- Gupta, C., Suggala, A. S., Goyal, A., Simhadri, H. V., Paranjape, B., Kumar, A., Goyal, S., Udupa, R., Varma, M., and Jain, P. (2017). ProtoNN: Compressed and accurate kNN for resource-scarce devices. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1331–1340, International Convention Centre, Sydney, Australia. PMLR.
- Gurobi Optimization, I. (2015). Gurobi optimizer reference manual. <http://www.gurobi.com>.
- He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
- Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.
- Ji, S. and Carin, L. (2007). Cost-sensitive feature acquisition and classification. *Pattern Recognition*, 40(5):1474–1485.
- Jordan, M. I. and Jacobs, R. A. (1994). Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6(2):181–214.
- Kanani, P. and Melville, P. (2008). Prediction-time Active Feature-Value Acquisition for Cost-Effective Customer Targeting. In *Advances In Neural Information Processing Systems (NIPS)*.
- Karayev, S., Baumgartner, T., Fritz, M., and Darrell, T. (2012). Timely object recognition. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 890–898. Curran Associates, Inc.
- Krizhevsky, A. (2009). Learning Multiple Layers of Features from Tiny Images. Master’s thesis, University of Toronto.
- Kulkarni, V. and Sinha, P. (2012). Pruning of random forest classifiers: A survey and future directions. In *International Conference on Data Science Engineering (ICDSE)*, pages 64–68.
- Kumar, A., Goyal, S., and Varma, M. (2017). Resource-efficient machine learning in 2 KB RAM for the internet of things. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1935–1944, International Convention Centre, Sydney, Australia. PMLR.

- Kusner, M., Chen, W., Zhou, Q., Zhixiang, E., Weinberger, K., and Chen, Y. (2014). Feature-cost sensitive learning with submodular trees of classifiers. In *AAAI Conference on Artificial Intelligence*.
- Lazebnik, S., Schmid, C., and Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2169–2178.
- Li, L.-J., Su, H., Xing, E. P., and Fei-Fei, L. (2010). Object Bank: A High-Level Image Representation for Scene Classification and Semantic Feature Sparsification. In *Advances in Neural Information Processing Systems*. MIT Press.
- Li, X.-B., Sweigart, J., Teng, J., Donohue, J., and Thombs, L. (2001). A dynamic programming based pruning method for decision trees. *INFORMS Journal on Computing*, 13(4):332–344.
- Lin, J., Rao, Y., Lu, J., and Zhou, J. (2017). Runtime neural pruning. In *Advances in Neural Information Processing Systems*, pages 2178–2188.
- Lopez-Paz, D., Schölkopf, B., Bottou, L., and Vapnik, V. (2016). Unifying distillation and privileged information. In *International Conference on Learning Representations*.
- Maaten, L., Chen, M., Tyree, S., and Weinberger, K. Q. (2013). Learning with marginalized corrupted features. In Dasgupta, S. and Mcallester, D., editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 410–418. JMLR Workshop and Conference Proceedings.
- MacKay, D. J. C. (1992). Information-based objective functions for active data selection. *Neural computation*, 4(4):590–604.
- Miller, A. (2002). *Subset selection in regression*. CRC Press.
- Moshkov, M. J. (2010). Greedy algorithm with weights for decision tree construction. *Fundamenta Informaticae*, 104(3):285–292.
- Nan, F. and Saligrama, V. (2017). Adaptive classification for prediction under a budget. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 4727–4737. Curran Associates, Inc.
- Nan, F., Wang, J., and Saligrama, V. (2015). Feature-budgeted random forest. In Blei, D. and Bach, F., editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1983–1991. JMLR Workshop and Conference Proceedings.

- Nan, F., Wang, J., and Saligrama, V. (2016). Pruning random forests for prediction on a budget. In *Advances in Neural Information Processing Systems 29*, pages 2334–2342. Curran Associates, Inc.
- Nan, F., Wang, J., Trapeznikov, K., and Saligrama, V. (2014). Fast margin-based cost-sensitive classification. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, Florence, Italy, May 4-9, 2014*.
- Nemhauser, G. L. and Wolsey, L. A. (1988). *Integer and Combinatorial Optimization*. Wiley-Interscience, New York, NY, USA.
- Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L. (1978). An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14(1):265–294.
- Nowak, R. (2008). Generalized binary search. In *In Proceedings of the 46th Allerton Conference on Communications, Control, and Computing*, pages 568–574.
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.
- Settles, B. (2009). Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.
- Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.
- Sheng, V. S. and Ling, C. X. (2006). Feature value acquisition in testing: A sequential batch test algorithm. In *Proceedings of the 23rd International Conference on Machine Learning, ICML '06*, pages 809–816, New York, NY, USA. ACM.
- Sherali, H. D., Hobeika, A. G., and Jeenanunta, C. (2009). An optimal constrained pruning strategy for decision trees. *INFORMS Journal on Computing*, 21(1):49–61.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., Rick Chang, J.-H., et al. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288.

- Trapeznikov, K. and Saligrama, V. (2013). Supervised sequential classification under budget constraints. In *International Conference on Artificial Intelligence and Statistics*, pages 581–589.
- Viola, P. and Jones, M. (2001). Robust Real-time Object Detection. *International Journal of Computer Vision*, 4:34–47.
- Wang, J., Bolukbasi, T., Trapeznikov, K., and Saligrama, V. (2014a). Model selection by linear programming. In *European Conference on Computer Vision*, pages 647–662.
- Wang, J., Trapeznikov, K., and Saligrama, V. (2014b). An lp for sequential learning under budgets. In *International Conference on Artificial Intelligence and Statistics*.
- Wang, J., Trapeznikov, K., and Saligrama, V. (2015). Efficient learning by directed acyclic graph for resource constrained prediction. In *Advances in Neural Information Processing Systems 28*, pages 2143–2151. Curran Associates, Inc.
- Xu, Z., Kusner, M., Chen, M., and Weinberger, K. Q. (2013). Cost-sensitive tree of classifiers. In *Proceedings of the 30th International Conference on Machine Learning*.
- Xu, Z. E., Weinberger, K. Q., and Chapelle, O. (2012). The greedy miser: Learning under test-time budgets. In *Proceedings of the 29th International Conference on Machine Learning, ICML*.
- Yang, A. Y., Jafari, R., Sastry, S. S., and Bajcsy, R. (2009). Distributed recognition of human actions using wearable motion sensor networks. *Journal of Ambient Intelligence and Smart Environments*, 1(2):103–115.
- Zhang, Y. and Huei-chuen, H. (2005). Decision tree pruning via integer programming. Working paper, <https://pdfs.semanticscholar.org/d8ce/ddf3335dc9b7898e92fcc3ca823bd582397a.pdf>.
- Zubek, V. B. and Dietterich, T. G. (2002). Pruning improves heuristic search for cost-sensitive learning. In *Proceedings of the Nineteenth International Conference on Machine Learning, ICML '02*, pages 19–26, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

CURRICULUM VITAE

