Feature Cost Sensitive Random Forest

Anna Thomas THOMASAT@STANFORD.EDU
Stanford University

Abstract

In many applications, it is necessary to consider not only the predictive power of a machine learning model, but also its computational cost at test time. Here we explore greedy methods for feature cost sensitive random forest training. We also consider the case where different features share common subroutines or other dependencies such that computing one reduces the computational cost of others. We formulate the resulting problem as submodular minimization, therefore admitting a polynomial time solution. We demonstrate that for diverse applications, test time cost can be reduced without significantly compromising accuracy.

1. Introduction

In recent years, controlling test time computational cost has emerged as a key challenge in the application of machine learning. If it is possible to achieve similar prediction accuracy while, for example, significantly reducing the load time of a web page or the cost of a medical test, then it will likely be beneficial to incorporate knowledge of this tradeoff into model training.

Here we focus on the costs induced by the process of extracting features at test time. As some features may be more expensive to compute than others, we explore strategies for incorporating this information into model training, with a tunable parameter to allow user control over the tradeoff between prediction time accuracy and feature cost.

In this project we focus on the random forest (Breiman et al. 2001), which is a powerful and widely used model in machine learning. It has been used successfully for body parts recognition, hand tracking, biological pathway analysis, ecological classification, and many other applications (Sharp et al. 2015; Shotton et al. 2013; Cutler et al. 2007; Diaz Urte et al. 2006).

However, Breiman's random forest algorithm does not incorporate the cost of feature acquisition in training. This may result in situations where for two features which are near identical in information gain (the metric used to choose node splits) but widely different in acquisition cost, the more expensive feature is chosen. Aggregated over many features and the various trees of the forest, choices such as these may significantly increase the overall prediction time cost of the forest without corresponding improvements in performance.

The goal in this work will thus be to achieve classification accuracy as similar as possible to the standard random forest algorithm, while reducing the cost as much as possible. The input is a labeled training set, feature costs, and a user determined tradeoff parameter between computational cost and prediction accuracy (denoted λ in following sections), and the output is the trained forest.

2. Related Work

Early work in this area focused on building cascades of classifiers in order to reduce test time cost. For example, Viola and Jones (2002) develop a widely used object detection cascade which allows background or other low probability regions of an image to be discarded quickly.

Xu et al. (2012) develop an extension to stagewise regression in which a budget exists for the total cost. They solve a relaxation of their optimization problem via coordinate descent, and show that it outperforms prior work. Gao and Koller (2011) introduced an ensemble method in which features are chosen dynamically based on their expected classification gain in addition to its computational cost, thereby creating an instance-specific decision path for every test example. They show that their method is able to meet the classification accuracy of traditional methods while significantly reducing the computatonal cost.

Karayev et al. (2013) do not focus on a specific classifier, but rather develop a general method for feature selection based on learning a Markov decision process. They learn a feature selection policy via policy iteration, and exceed the performance of Xu et al. (2012) on the Scene-15 dataset. One recent paper, Nan et al. (2015), explored a random forest model in which a budget exists for the total feature cost. They grow trees using minimax cost-weighted impurity splits and show that their algorithm outperforms the then state of the art in tree-based cost-sensitive classification.

In this paper we explore a different but related problem, in which we do not have an explicit budget, but rather aim to maximize the weighted difference of a function which reflects the quality of the features used in the forest and a function which reflects the cost. Section 4 of this work also differs from previous work in that we formulate cost-sensitive feature selection, in the case where we have access to a dependency graph between features which represents shared computational subroutines, as a submodular optimization problem.

3. Greedy Forest Construction

3.1. Problem formulation

Similar to the formulation in Nan et al. 2015, our goal is to learn a classifier F from a family of functions $\mathcal{F}$ that minimizes the sum of the expected errors and the computational cost of the final feature set, assuming that test instances are drawn from a probability distribution $(x, y) \sim D$:

$$\min_{F \in \mathcal{F}} E_{xy}[L(y, f(x))] + \lambda E_x[C(f, x)] \tag{1}$$

where $L(y, \hat{y})$ is a loss function and $C(f, x)$ is the cost of evaluating the function of f on example x.

Our formulation differs from that of Nan et al. (2015) in that we do not have a constraint on the feature costs, but rather incorporate the cost minimization into the objective itself.

Since in practice we are given a training set sampled from a distribution, not the distribution itself, we will instead solve the following problem:

$$\min_{F \in \mathcal{F}} \sum_{i=1}^{N} L(y^{(i)}, f(x^{(i)})) + \lambda \sum_{j=1}^{|F_S|} C_j \tag{2}$$

3.2. Modified splitting criterion

We first test a simple randomized greedy algorithm. When $\lambda = 0$, this is equivalent to Breiman's original random forest algorithm. Following the original algorithm, we use the information gain, or reduction in entropy induced by splitting on a feature, as a measure of the quality of a split. We modify the split function used in tree construction to reflect the feature cost in addition to the information gain. Thus, if S denotes the set of splits that has already been chosen, and $C(i|S)$ denotes the cost of computing feature i given that we have already chosen the splits in S, then the value of a split s_{ij}, i.e. splitting at value j for feature i, is given by:

$$F(s_{ij}|S) = H(N) - H(N|s_{ij}) - \lambda C(i|S) \tag{3}$$

where $H(N)$ is the entropy of the current node.

Here, the greedy objective used for computing splits is weighted sum of the information gain and the feature cost. Once a feature has been used in any tree in the forest, its cost of reuse is 0, as shown in Algorithms 1 and 2; the vector $C \in \mathbb{R}^n$ is the vector of feature costs. We apply this modified splitting criterion in Algorithm 2.

Algorithm 1 Cost Sensitive Random Forest

```
Input: $X \in \mathbb{R}^{mxn}$, $y \in \mathbb{Z}^m$, $C \in \mathbb{R}^n$, $N \in \mathbb{Z}$, $\lambda \in \mathbb{R}$
$\mathcal{T} \leftarrow \emptyset$;
for each tree i=1:N do
    Randomly sample training data to form $X^i$ and $y^i$
    $T, C' \leftarrow$ GreedyTree($X^i, y^i, C, \lambda$)
    $C := C'$
    $\mathcal{T} \leftarrow \mathcal{T} \cup T$
return $\mathcal{T}$
```

Algorithm 2 Cost Sensitive Greedy Tree

```
for each attribute i=1:M do
    Randomly sample splits $s_{ij}$ and compute $F(s_{ij})$ as described in (3)
$\hat{s} \leftarrow \text{argmin}_s F(s)$
$C_i := 0$
Create new node using feature $i$ and split value $j$.
for each child node do
    GreedyTree($(X^i)_{\hat{s}}, (y^i)_{\hat{s}}, C, \lambda$)
return $T, C$
```

3.3. Complementary forest training

In order to achieve very low feature costs, it is necessary to consider the full set of features at each split. Selecting a random set of $\sqrt{n}$ features, as is recommended in the original random forest paper (Breiman et al. 2001), would prevent the repeated reuse of a few features that is required to achieve very low cost.

However, considering the full set of features at each split is likely to increase the correlation of the trees in the forest, as informative features may be used repeatedly. One recent study of random forests found that, for many applications, low correlation among trees is a key factor in prediction accuracy of the forest as a whole (Bernard et al. 2010).

Thus, in order to account for increased intra-forest correlation of the resulting model when computing splits based on the full set of features, we incorporate a boosting like strategy, based on the work of Bernard et al. (2012) into Algorithm 1 to reduce this effect.

Algorithm 3 Complementary Cost Sensitive Random Forest

```
Input: $X \in \mathbb{R}^{mxn}$, $y \in \mathbb{Z}^m$, $C \in \mathbb{R}^n$, $N \in \mathbb{Z}$, $\lambda \in \mathbb{R}$
$\mathcal{T} \leftarrow \emptyset$;
for each training example j=1:M do
    $W_j = \frac{1}{M}$
for each tree i=1:N do
    Randomly sample training data to form $X^i$ and $y^i$
    $T, C' \leftarrow$ GreedyTree($X^i, y^i, C, \lambda$)
    $C := C'$
    for each training example j=1:M do
        $\epsilon_j = \frac{1}{|\mathcal{T}|} \sum_{T_i \in \mathcal{T}} I(h_i(x_j) = y_j)$
        $W_j = \epsilon_j$
    $\mathcal{T} \leftarrow \mathcal{T} \cup T$
return $\mathcal{T}$
```

Feature	Cost
Num. Times Pregnant	1.00
Glucose Tolerance	17.61
Diastolic Pb	1.0
Triceps	1.0
Insulin	22.78
Mass Index	1.0
Pedigree	1.0
Age	1.0

Table 1. Example of feature costs provided in the Pima Indian diabetes dataset.

3.4. Datasets and features

3.4.1. Pima Indian diabetes diagnosis

This binary classification task has 8 features. It is located at https://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes. The cost of each of the 8 medical tests or patient health records used to diagnose diabetes is included and used as input to Algorithms 1 and 3.

3.4.2. Spam filtering

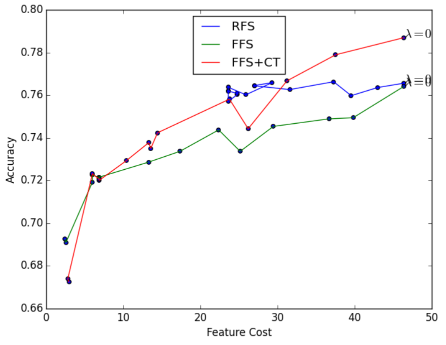
This dataset, also a binary classification task, is located at https://archive.ics.uci.edu/ml/datasets/Spambase. The features, which had already been extracted, correspond to word frequencies, character frequencies, average length of uninterrupted sequences of capital letters, length of longest uninterrupted sequence of capital letters, and the total number of capital letters in the email. Feature costs were not provided, so we set $C(S) = |S|$.

3.4.3. Digit recognition

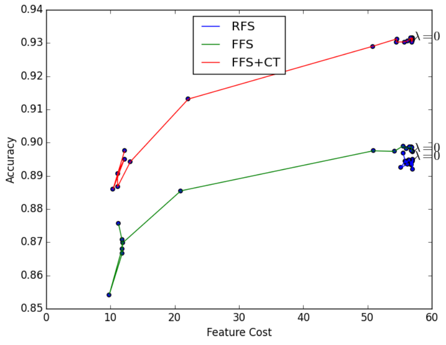
This is a multiclass classification task. The dataset is located at https://archive.ics.uci.edu/ml/datasets/Multiple+Features. The features, which had already been extracted, correspond to pixel intensity averages in 2 x 3 windows. Feature costs were not provided, so we set $C(S) = |S|$.

3.5. Results

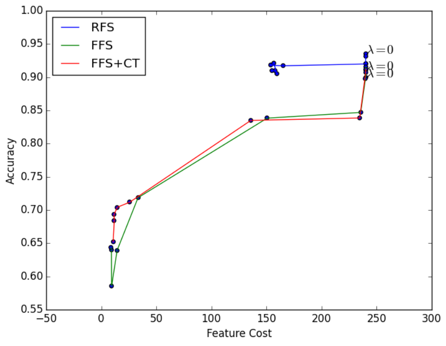
We evaluate the greedy forest training method on three datasets. For spam classification and digit recognition costs were not provided, so we use $C(S) = |S|$. In general, it appears that for these three problems it is possible to reduce feature cost while not substantially reducing accuracy. For the diabetes dataset, we are able to reduce cost, for example, by an average of 23% while decreasing accuracy by an average of 0.9%, or 51% and 2.4% respectively. For the spam classification dataset, it is possible to reduce cost by 77% while reducing accuracy by 3.6%. For the digit recognition dataset, both methods which use the full feature set to compute splits produce a significant accuracy reduction as cost decreases, but computing a random subset of size $\sqrt{n}$, where n is the number of features, allows for a cost reduction of 55.8% while accuracy decreases by 2.6%. We additionally note that complementary training



(a)



(b)



(c)

Figure 1. Cost versus accuracy tradeoff for three datasets: a) diabetes diagnosis, b) spam classification, c) digit recognition. FFS - full feature set considered at each split; RFS - randomized feature set of size $\sqrt{n}$ considered at each split; FFS + CT - full feature set and complementary training (Algorithm 3). Results are the average of 20 randomized trials. Each point represents a different value of λ.

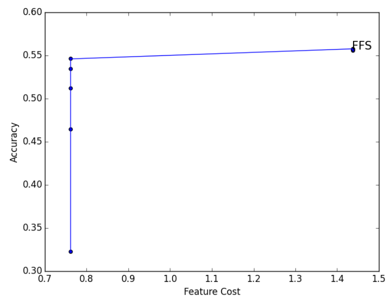


Figure 2. Cost versus accuracy tradeoff for the Scene-15 dataset, using cost-sensitive feature selection before applying Breiman's random forest algortihm. Results are the average of 20 randomized trials.

appears to improve performance in two of the three datasets, suggesting the importance of low correlation among trees. As expected, the randomized subset selection algorithm is limited in the extent to which cost can be reduced. See Figure 2 for the full tradeoff curves.

4. Global Optimization of Feature Set

4.1. Submodular set functions

In addition to greedy tree training, we may also want to explore a global strategy for choosing the best subset of features to use as input to the random forest algorithm. This section is applicable to the more general problem of cost sensitive feature selection, and not confined to the random forest model. Here we formulate this problem as a discrete optimization problem in which we aim to identify the best possible subset of features in order to balance feature cost and prediction accuracy. We then use this subset of features to train the random forest, using the original algorithm from Breiman et al. (2001).

A submodular set function is one that satisfies the property of diminishing marginal gains: for every $X, Y \subseteq \Omega$ with $X \subseteq Y$ and every $x \in \Omega \backslash Y$:

$$f(X \cup \{x\}) - f(X) \geq f(Y \cup \{x\}) - f(Y) \qquad (4)$$

Submodular functions have the appealing property that greedy maximization of submodular functions is guaranteed to obtain near optimal results, and submodular minimization can be solved exactly in polynomial time (Nemhauser et al. 1978).

4.2. Cost dependency graph

We define a weighted hypergraph $H = (V, E)$ where each vertex $v \in V$ corresponds to a feature. For a subset of features $S \in F$, $S \in E$ if computing some feature $f \in S$ reduces the cost of computing all other features $S \backslash f$ by $W(S) \in \mathbb{R}$. Thus, hyperedges correspond to groups of features which share a common subroutine. We also have a vector $C \in \mathbb{R}^n$ of the original feature costs.

Thus, to assess the cost of a subset S of features, we compute:

$$C(S; H) = \sum_{f \in S} (C_f - \sum_{T \in E : f \in T, |T \cap S| > 1} W(T)) \qquad (5)$$

4.3. Mutual information

In order to estimate the quality of each feature X we use the mutual information between the feature and the class variable Y, defined as:

$$MI(X; Y) = \sum_{y \in Y} \sum_{x \in X} p(x, y) \log \frac{p(x, y)}{p(x)\, p(y)} \qquad (6)$$

In the case of continuous features, we discretize into bins before computing (6). We use 50 bins for all features.

Although it would have been better to take into account complementary relationships among features via the joint mutual information, due to computational constraints we did not explore this. This would then become a problem of minimizing the difference between submodular functions, for which good approximation guarantees are not currently available.

4.4. Problem formulation

Thus our final objective for cost-sensitive feature selection is to compute $S^* \subseteq F$, defined as:

$$S^* = \text{argmin}_{S \in F} C(S; H) - \lambda \sum_{f \in S} MI(f; Y) \qquad (7)$$

where $C(S; H)$ and $MI(f; Y)$ are defined in 5 and 6 respectively, and $\lambda \in \mathbb{R}$ is a user determined parameter controlling the tradeoff between computational cost and accuracy.

This objective is submodular. See 7 for details.

4.5. Optimization

Several polynomial time algorithms for submodular minimization exist (Orlin 2009; Schrijver 2000; Iwata 2002). However, in practice it has been found that the minimum-norm point algorithm, proposed by Wolfe (1976) and extended to submodular minimization by Fujishige et al. (2005), is faster, even though the worst case time complexity is currently unknown. Thus we use the minimum-norm point algorithm, as implemented in the Submodular Function Optimization Toolbox (Krause 2010). Wolfe's minimum-norm point algorithm solves the following problem: given a finite set P of points in $\mathbb{R}^n$, find the minimum norm point in the convex hull $\hat{P}$ of these points. It is initialized with a random point, and then repeatedly updates an affinely independent subset of points (a simplex) by solving a linear optimization problem over $\hat{P}$. This is repeated until convergence. For a general polytope, the minimum-norm point algorithm can be exponential in the runtime. However, for a submodular polytope, Fujishige showed that the update can be computed efficiently.

Thus, in order to train the random forest, we solve the problem posed in (7) via the minimum-norm point algorithm, and

Figure 3. Examples of images in the Scene-15 dataset.

Dataset	**Training**	**Test**	**Num. Feat.**
Diabetes Diagnosis	400	368	8
Spam Filtering	2000	2601	57
Digit Recognition	1000	1000	240
Scene-15 Classification	1500	2685	800

Table 2. Datasets used in this work. The first three were obtained from the UCI Machine Learning Repository located at `http://archive.ics.uci.edu/ml/`, and the Scene-15 dataset was obtained from Matt Kusner.

then apply the standard random forest algorithm (Breiman et al. 2001) on the resulting set of features.

4.6. Dataset and Features

The Scene-15 dataset (Lazebnik et al. 2006) contains 4485 images, each belonging to one of 15 scene classes. We used the same features as Xu et al. 2012, namely GIST, spatial HOG, local binary patterns, self-similarity, texton histogram, geometric textons, geometric color, and Object Bank (Li et al. 2010). The original dataset has 76187 features total; due to computational constraints (the typical runtime for the minimum-norm algorithm we use to find the optimal feature subset was estimated at $O(n^7)$ by Jegelka et al. (2011)) we randomly sample 100 features from each descriptor to arrive at 800 features total. The costs associated with these features are submodular because once a feature associated with a descriptor is used, there is no additional cost to using another feature from the same descriptor.

4.7. Results

We then evaluate the submodular cost-sensitive feature selection method on the Scene-15 dataset, using the same train/test split as Xu et al. (2012). As shown in Figure 2, we are able to reduce feature cost by 48.5% while only reducing accuracy by 0.12%. However, beyond this initial reduction we unable to further reduce cost, even though the set of features returned by the feature selection algorithm grows progressively smaller as λ decreases (data omitted due to space). This suggests that combining the cost-sensitive feature selection algorithm with cost-sensitive splits, as in Algorithms 1 and 2 may yield further improvements.

Depending on the value of λ, the time to minimize 7 can be 10 minutes or more, suggesting that approximate methods may be worthwhile to explore in the future.

5. Conclusion and Future Work

In this work two general strategies for reducing test-time cost of the random forest algorithm have been explored: greedy tree construction with the split selection modified to incorporate the feature cost, and a cost sensitive feature selection method which considers the dependencies among features. In the first case, we show that a simple modification to the random forest algorithm allows for user control over the computational cost of the trained classifier. In order to account for the increased intra-forest correlation when more variables are considered at each node split, we also test a boosting-like iterative sample reweighting strategy (based on the work of Bernard et al. (2010)), which generally improves performance. In the second case, we show that cost sensitive feature selection can be formulated as a submodular minimization problem, for which an exact solution can be found in polynomial time. This feature selection method can be applied to any model and is not specific to the random forest.

Overall, these results indicate that for several real world problems it is possible to significantly reduce test time cost with minimal effects on accuracy. In the future, it would be interesting to test approximate methods for submodular minimization in order to improve training time, combine the two proposed methods, and consider model evaluation time in addition to feature extraction time. It would also be interesting to use the joint mutual information in assessing feature quality and apply approximate methods for minimizing the difference of submodular functions.

6. Acknowledgments

Thank you to Junjie Qin for valuable feedback, and to Matthew Kusner for providing the Scene-15 dataset and feature costs.

7. Appendix

Proof of submodularity. We need to show that for every $X, Y \subseteq \Omega$ with $X \subseteq Y$ and every $x \in \Omega \backslash Y$:

$$F(X \cup \{x\}) - F(X) \geq F(Y \cup \{x\}) - F(Y)$$

$$C(X \cup \{x\}; H) - \lambda \sum_{f \in X \cup \{x\}} MI(f;Y) - C(X;H)$$

$$+\lambda \sum_{f \in X} MI(f;Y) \geq C(Y \cup \{x\}; H) - \lambda \sum_{f \in Y \cup \{x\}} MI(f;Y) - C(Y;H) + \lambda \sum_{f \in Y} MI(f;Y)$$

$$C(X \cup \{x\}; H) - C(X;H) - \lambda MI(x;Y) \geq C(Y \cup \{x\}; H) - \lambda MI(x;Y) - C(Y;H)$$

$$C(X \cup \{x\}; H) - C(X;H) \geq C(Y \cup \{x\}; H) - C(Y;H)$$

Since $X \subseteq Y$ and the marginal cost of a fixed feature can only decrease as more features added, F is thus submodular.

8. References

1. Bernard, Simon, Sbastien Adam, and Laurent Heutte. ”Dynamic random forests.” Pattern Recognition Letters

33.12 (2012): 1580-1586.

2. Bernard, Simon, Laurent Heutte, and Sbastien Adam. ”A study of strength and correlation in random forests.” Advanced Intelligent Computing Theories and Applications. Springer Berlin Heidelberg, 2010. 186-191.

3. Blake, C.L. and C. J. Merz. UCI repository of machine learning databases, 1998. https://archive.ics.uci.edu/ml

4. Breiman, Leo. ”Random forests.” Machine learning 45.1 (2001): 5-32.

5. Cutler, D. Richard, et al. ”Random forests for classification in ecology.” Ecology 88.11 (2007): 2783-2792.

6. Daz-Uriarte, Ramn, and Sara Alvarez De Andres. ”Gene selection and classification of microarray data using random forest.” BMC bioinformatics 7.1 (2006): 3.

7. Fujishige, Satoru. Submodular functions and optimization. Vol. 58. Elsevier, 2005.

8. Gao, Tianshi, and Daphne Koller. ”Active classification based on value of classifier.” Advances in Neural Information Processing Systems. 2011.

9. Iwata, Satoru. ”A fully combinatorial algorithm for submodular function minimization.” Journal of Combinatorial Theory, Series B 84.2 (2002): 203-212.

10. Jegelka, Stefanie, Hui Lin, and Jeff A. Bilmes. ”On fast approximate submodular minimization.” Advances in Neural Information Processing Systems. 2011.

11. Karayev, Sergey, Mario J. Fritz, and Trevor Darrell. ”Dynamic feature selection for classification on a budget.” International Conference on Machine Learning (ICML): Workshop on Prediction with Sequential Models. 2013.

12. Krause, Andreas. ”SFO: A toolbox for submodular function optimization.” The Journal of Machine Learning Research 11 (2010): 1141-1144.

13. Lazebnik, Svetlana, Cordelia Schmid, and Jean Ponce. ”Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories.” Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on. Vol. 2. IEEE, 2006.

14. Li, Li-Jia, et al. ”Object bank: A high-level image representation for scene classification and semantic feature sparsification.” Advances in neural information processing systems. 2010.

15. Nan, Feng, Joseph Wang, and Venkatesh Saligrama. ”Feature-Budgeted Random Forest.” arXiv preprint arXiv:1502.05925 (2015).

16. Nemhauser, George L., Laurence A. Wolsey, and Marshall L. Fisher. ”An analysis of approximations for maximizing submodular set functionsI.” Mathematical Programming 14.1 (1978): 265-294.

17. Orlin, James B. ”A faster strongly polynomial time algorithm for submodular function minimization.” Mathematical Programming 118.2 (2009): 237-251.

18. Ren, Shaoqing, et al. ”Global refinement of random forest.” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.

19. Schrijver, Alexander. ”A combinatorial algorithm minimizing submodular functions in strongly polynomial time.” Journal of Combinatorial Theory, Series B 80.2 (2000): 346-355.

20. Sharp, Toby, et al. ”Accurate, Robust, and Flexible Real-time Hand Tracking.” Proc. CHI. Vol. 8. 2015.

21. Shotton, Jamie, et al. ”Real-time human pose recognition in parts from single depth images.” Communications of the ACM 56.1 (2013): 116-124.

22. Viola, P. and Jones, M. Robust real-time object detection. International Journal of Computer Vision, 57(2):137154, 2002.

23. Xu, Zhixiang, et al. ”Cost-Sensitive Tree of Classifiers.” Proceedings of The 30th International Conference on Machine Learning. 2013.

24. Xu, Zhixiang, Kilian Weinberger, and Olivier Chapelle. ”The greedy miser: Learning under test-time budgets.” arXiv preprint arXiv:1206.6451 (2012).

25. Wang, Joseph, Kirill Trapeznikov, and Venkatesh Saligrama. ”An LP for Sequential Learning Under Budgets.” Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics. 2014.

26. Wolfe, Philip. ”Finding the nearest point in a polytope.” Mathematical Programming 11.1 (1976): 128-149.