FAST MARGIN-BASED COST-SENSITIVE CLASSIFICATION

Feng Nan, Joseph Wang, Kirill Trapeznikov, Venkatesh Saligrama

Boston University

ABSTRACT

We present a novel classification algorithm for learning with test time budgets. In this setting, the goal is to reduce feature acquisition cost while maintaining classification accuracy. For every decision, our approach dynamically selects features based on previously observed information. Once a desired confidence of a decision is achieved, the acquisition stops and the test instance is classified. Our approach can be used in conjunction with many popular margin based classification algorithms. We use margin information from training data in the partial feature neighborhood of a test point to compute a probability of correct classification. This estimate is used to either select the next feature or to stop. We compare our algorithm to other cost-sensitive methods on real world datasets. The experiments demonstrate that our algorithm provides an accurate estimate of classification confidence and outperforms other approaches while being significantly more efficient in computation.

Index Terms**—** cost-sensitive, learning with test time budget, dynamic feature selection

1. INTRODUCTION

Most of traditional machine learning has focused on improving accuracy of classification algorithms. However, in a growing number of applications, high classification accuracy has to be balanced with the cost of using an algorithm. This cost arises due to the computational complexity of acquiring features during test time. Often some decisions can be made reliably with several cheap features while others require the full feature set. For example, in a web search ranking application, an algorithm has to return relevant results in milliseconds. It is impossible to compute complex features of each document in order to rank them within such a short time. In fact, many documents can be quickly eliminated using cheap pre-computed features such as PageRank, and the remaining documents need to be scored with more expensive features ([1]). In an object recognition application, many features are costly to compute and detection cascades ([2, 3, 4]) use a sequence of stages that are tuned to make majority decisions using fast features and compute more expensive features only when necessary.

This work is partially supported by NSF Grants 0932114 and 1330008

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 ([5]). 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 3 and Experiments in Section 4.

Related Work: Classification under budgets has become an increasingly active research topic in recent years. One direction has been in learning a fixed order in which features are to be acquired for every test point. For instance, the greedy miser algorithm ([6]) performs cost sensitive boosting of regression trees by balancing the choice of feature with its measurement cost. However, the resulting decision system is not adaptive and requests the same order of features for every decision. In contrast, our approach dynamically determines which features will improve classification for every test instance. In a partially adaptive setting, sequential classifiers ([7, 8]) and detection cascade ([2, 3, 4]) consist of several stages of decisions. Early stages utilize cheap/fast features and pass the difficult examples to later stages to acquire more expensive features. However, each stage is limited to either classifying or requesting the next feature in the predetermined order while in our method, the order is adaptive to every decision.

Other methods formulate this problem as a Markov Decision Process. However, classical MDP solutions becomes intractable as the number of features and the state space size grow. Authors in [9] use standard linear approximation of Q factors for learning policies for timely image classification. Other approaches utilize imitation learning from the robotics community. Here, researchers assume access to an oracle policy that outputs the optimal decision given current state and the goal is to train a policy that imitates this oracle [10]. However, this approach needs to iterate through a quickly growing set of training instances which is computationally expensive.

Authors in [11] present a feature selection scheme related to test time budgeted learning. The method uses Gaussian mixture to model the likelihood of an unmeasured feature given observed measurements. At each time, this likelihood model is used to estimate expected value of acquiring each feature, and the one with the highest value is chosen. In contrast, our method does not estimate the unmeasured feature distribution but estimates the probability of correct classification directly. In fact, our method is purely data-based and does not assume a probability model. We compare this method in Section 4.

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 $\phi(x^{(l)}) \in \Re^d$, 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, $f(x) = w^T\phi(x)$, 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([12]) 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 first feature randomly or according to some simple rules since no information about the test point is available. Let $\phi_{\mathcal{O}}(x)$ denote the measurement values obtained about the test point and we set the unmeasured feature values to be 0 [1], $\phi_{\overline{\mathcal{O}}}(x) = 0$. If a classification is needed with the current measurements we can simply compute

$$y = w_{\mathcal{O}}^T \phi_{\mathcal{O}}(x), \tag{1}$$

and decide based on its sign.

Given any measured feature set $\mathcal{O}$, it is not clear how $w_{\mathcal{O}}^T\phi_{\mathcal{O}}(x)$ relates $w^T\phi(x)$ (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 (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 $\phi_{\mathcal{O}}(x)$ on the index set $\mathcal{O}$. We define $N(\mathcal{O})$ to contain the K nearest neighbors (with respect to Euclidean distance) of $\phi_{\mathcal{O}}(x)$ in the training set, where K is a positive natural number.[2]

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

$$\eta_k^{\mathcal{O}} = y^{(k)}(w_{\mathcal{O}}^T\phi_{\mathcal{O}}(x^{(k)})), k \in N(\mathcal{O}). \tag{2}$$

If $\eta_k^{\mathcal{O}}$ is positive then (1) will give correct classification based on the measured feature set $\mathcal{O}$. Similarly, define the one-step-ahead partial margin of the kth training point in the neighborhood $N(\mathcal{O})$ based on the current measurement feature set $\mathcal{O}$ and feature i as

$$\eta_{i,k}^{\mathcal{O}} = y^{(k)}(w_{\mathcal{O}}^T\phi_{\mathcal{O}}(x^{(k)}) + w_i\phi_i(x^{(k)})), k \in N(\mathcal{O}), i \in \overline{\mathcal{O}}. \tag{3}$$

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

[1] Note that this is a missing feature classifier. While there has been some work (see [13]) on learning classifiers robust to missing features, this is outside the scope of this paper.

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

of correct classification to the total number of training points within the neighborhood:

$$p^{\mathcal{O}} = \frac{\#\{k : \eta_k^{\mathcal{O}} > 0\}}{|N(\mathcal{O})|}. \tag{4}$$

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

$$p_i^{\mathcal{O}} = \frac{\#\{k : \eta_{i,k}^{\mathcal{O}} > 0\}}{|N(\mathcal{O})|}. \tag{5}$$

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.

3. ALGORITHM

We present our algorithm in Algorithm 1. Notice that feature cost can be factored in to the decision process naturally. In Line 13 of the algorithm α represents a trade-off parameter of the cost with accuracy.

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

```
1: Train classifier $y = w^T \phi(x)$ on the entire training data
2: Given a test point $x$:
3: Measure feature $i$
4: for $t = 1 \to d$ do                ▷ Iterate through the total number of features
5:     if $p^{\mathcal{O}} >$ threshold then
6:         Stop, make classification
7:     else
8:         Compute neighborhood $N(\mathcal{O})$
9:         for $i \in \overline{\mathcal{O}}$ do
10:            Update partial margins $\eta_{i,k}^{\mathcal{O}}$ for $k \in N(\mathcal{O})$ according to (3)
11:            Compute $p_i^{\mathcal{O}}$ according to (5)
12:        end for
13:        Select feature $i_{\max} = \text{argmax}_i p_i^{\mathcal{O}} - \alpha c_i$ 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)
15:        Compute $p^{\mathcal{O}}$ according to (4)
16:    end if
17: end for
```

Illustrative Example: We demonstrate our algorithm on a synthetic example in order to show the effectiveness of our approach. Here we assume the feature extraction returns the corresponding components of the data point; in other word, $\phi(x) = x$. Suppose there are 8 training data points as shown in Figure 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.

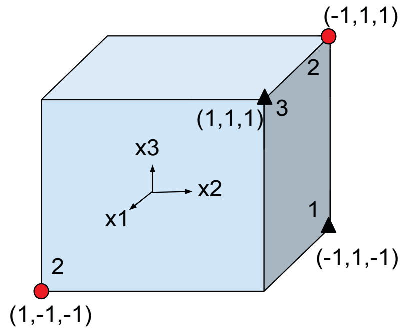

Fig. 1. An example of cost sensitive learning. Given 8 training points, each is binary with 3 features: $x^{(1)} = x^{(2)} = (1, -1, -1), x^{(3)} = x^{(4)} = (-1, 1, 1), x^{(5)} = (-1, 1, -1), x^{(6)} = x^{(7)} = x^{(8)} = (1, 1, 1)$, with labels $y^{(1)}, \ldots, y^{(4)} = -1, y^{(5)}, \ldots, y^{(8)} = 1$. 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 (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). 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 (3). Therefore we obtain $p_1^{\mathcal{O}} = \frac{5}{6}$ using (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 (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 (4) is the sample mean of the actual classification accuracy according to (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 extremely low test time complexity. In fact, it scales linearly in sample size. There are at most d iterations (the total number of features) and each iteration involves only $\mathbf{O}(nd^2K)$ operations, where K is a constant neighborhood size.

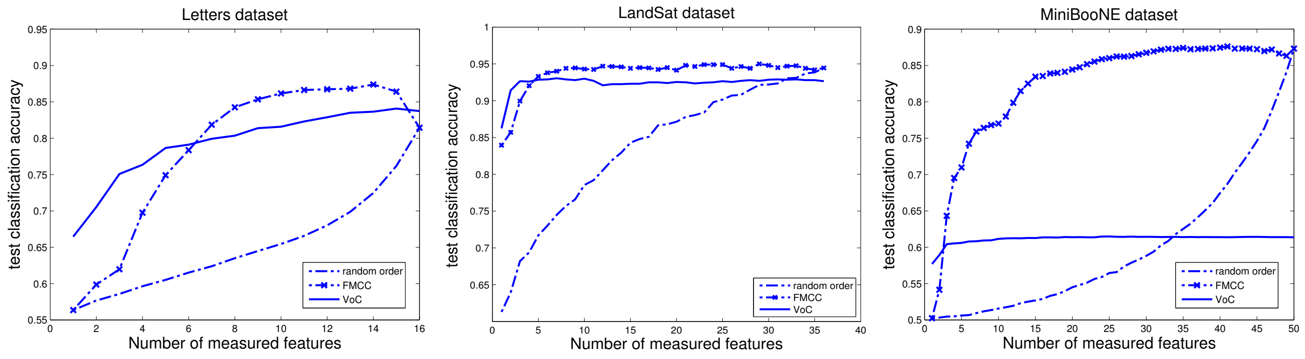

Fig. 2. Experiment result of classification accuracy vs number of features measured on Letters, LandSat MiniBooNE datasets. FMCC is consistent across all datasets while the VoC does not perform well on the MiniBooNE dataset.

4. EXPERIMENTS

We evaluate our algorithm on three UCI data sets [14]. To demonstrate the wide applicability of our algorithm, we base our algorithm on Boosting for the first two data sets 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 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 [11], 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 (1). We see from Fig. 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 is much faster than VoC, which requires solving a locally weighted least square problem at each stage.

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

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 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 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 1 we also compare the partial probability of correct classification $p^{\mathcal{O}}$ (See Eq(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.

5. REFERENCES

[1] Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan, "Expected reciprocal rank for graded relevance," in *Proceedings of the 18th ACM conference on Information and knowledge management*, New York, NY, USA, 2009, CIKM '09, pp. 621–630, ACM.

[2] Paul A. Viola and Michael J. Jones, "Rapid object detection using a boosted cascade of simple features.," in *CVPR (1)*. 2001, pp. 511–518, IEEE Computer Society.

[3] M. Chen, Z. Xu, K.˜Q. Weinberger, O. Chapelle, and D. Kedem, "Classifier cascade: Tradeoff between accuracy and feature evaluation cost," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012, pp. 235–242.

[4] C. Zhang and Z. Zhang, "A Survey of Recent Advances in Face Detection," Tech. Rep., Microsoft Research, 2010.

[5] Corinna Cortes and Vladimir Vapnik, "Support-vector networks," in *Machine Learning*, 1995, pp. 273–297.

[6] Zhixiang Eddie Xu, Kilian Q. Weinberger, and Olivier Chapelle, "The greedy miser: Learning under test-time budgets," in *ICML*, 2012.

[7] K Trapeznikov, V Saligrama, and D A Castañon, "Mulit-Stage Classifier Design," in *Machine Learning*, 2013, pp. 1–24.

[8] K Trapeznikov and V Saligrama, "Supervised sequential classification under budget constraints," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2013.

[9] Sergey Karayev, Tobias Baumgartner, Mario Fritz, and Trevor Darrell, "Timely object recognition," in *NIPS*, 2012, pp. 899–907.

[10] Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11)*, Geoffrey J. Gordon and David B. Dunson, Eds. 2011, vol. 15, pp. 627–635, Journal of Machine Learning Research - Workshop and Conference Proceedings.

[11] T. Gao and D. Koller, "Active classification based on value of classifier," in *Advances in Neural Information Processing Systems (NIPS 2011)*, 2011.

[12] Yoav Freund and Robert E Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119 – 139, 1997.

[13] Laurens Maaten, Minmin Chen, Stephen Tyree, and Kilian Q. Weinberger, "Learning with marginalized corrupted features," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, Sanjoy Dasgupta and David Mcallester, Eds. 2013, vol. 28, pp. 410–418, JMLR Workshop and Conference Proceedings.

[14] K. Bache and M. Lichman, "UCI machine learning repository," 2013.