Cost-sensitive Dynamic Feature Selection

He He HHE@CS.UMD.EDU
Hal Daumé III HAL@UMIACS.UMD.EDU
Dept. of Computer Science, University of Maryland, College Park, MD

Jason Eisner JASON@CS.JHU.EDU
Dept. of Computer Science, Johns Hopkins University, Baltimore, MD

Abstract

We present an instance-specific test-time dynamic feature selection algorithm. Our algorithm sequentially chooses features given previously selected features and their values. It stops the selection process to make a prediction according to a user-specified accuracy-cost trade-off. We cast the sequential decision-making problem as a Markov Decision Process and apply imitation learning techniques. We address the problem of learning and inference jointly in a simple multiclass classification setting. Experimental results on UCI datasets show that our approach achieves the same or higher accuracy using only a small fraction of features than static feature selection methods.

1. Introduction

In a practical machine learning task, features are usually acquired at a cost with unknown discriminative powers. In many cases, expensive features often imply better performance. For example, in medical diagnosis, some tests can be very informative (e.g., X-ray, electrocardiogram) but are expensive to run or have side-effects on human body. Oftentimes, while at training time we can devote large amounts of time and resources to collecting data and building models, at test time we may not afford to obtain a complete set of features for all instances. This leaves us the cost-accuracy trade-off problem.

We consider the setting where a pretrained model using a complete set of features is given and each feature has a known cost. At *test time*, we would like to dynamically select a subset of features for *each instance* and be able to explicitly specify the cost-accuracy trade-off. This can be naturally framed as a sequential decision-making problem. At each step, based on the instance's current feature set, we decide whether to stop acquiring features and make a prediction; if not, which feature(s) to purchase next.

A direct solution is to cast this as a Markov Decision Process. This allows us to search for an optimal purchasing policy under a reward function that combines cost and accuracy (Section 2). We propose to decompose inference into a sequence of simple classification tasks and learn the classifiers using imitation learning methods (Section 3). A typical approach to imitation learning is to define an oracle that executes the optimal policy based on the reward function; using the oracle-generated examples as supervised data, one can learn a classifier/regressor to mimic the oracle's behavior. However, sometimes the optimal actions can be too good for the agent to imitate due to limitation of the learning policy space. In such cases, instead of labeling a state with the maximum-reward action, we label it with a possibly suboptimal action that has a fairly high reward *and* already scores fairly well under the current model (McAllester et al., 2010; Chiang et al., 2009) (Section 4). These labels may change during learning as we improve the model parameters. Intuitively, this allows the learner to move towards a better action without much effort, and to gradually achieve the best action it can, instead of aiming at an impractical goal from the beginning.

Our main contribution is developing a novel imitation learning framework for test-time dynamic feature selection. Our model does not have any constraint on the type of features or the pretrained model; and we allow users to explicitly specify the trade-off between accuracy and cost.

Presented at the International Conference on Machine Learning (ICML) workshop on *Inferning: Interactions between Inference and Learning*, Edinburgh, Scotland, UK, 2012.

2. Dynamic Feature Selection as an MDP

In a typical supervised classification setting, we have a training set $\{(\boldsymbol{x}_1, y_1) \dots (\boldsymbol{x}_n, y_n)\}$ and have access to all the feature values. We assume that we are provided with a classifier that has been trained to work well on instances for which all features are known. We will refer to this pretrained classifier as the *data classifier*. We assume that there exists a (possibly empty) set of free features, whose values are known up front for each test instance, while other features have to be obtained at a cost. The precise definition of cost is problem-dependent, for instance the computation time or the expense of running an experiment. Our goal is to achieve high accuracy without spending too much on acquiring features.

We represent the dynamic feature selection process as a Markov Decision Process (MDP). We allow the agent to select more than one feature at a time. A selectable bundle of one or more features is called a *factor*; such a bundle might be defined by a feature template, for example, or by a procedure that acquires several features at once. The *state* includes the set of factors selected so far. Thus on any fixed input we have an exponentially large state space of size 2^D, where D is the number of available factors. The *action* space includes all factors that have not been selected yet, as well as a termination action (stop adding more features and make a prediction). An agent follows a memoryless *policy* π that determines which action to choose in state s, i.e. $\pi(s) \rightarrow a$.

In the MDP setting, achieving an accuracy-cost tradeoff corresponds to finding the optimal *policy* under a *reward function*. The reward function should allow us to explicitly specify the trade-off. When considering a single instance, we use the margin given by the data classifier to reflect accuracy. Let $\mathcal{Y}$ be the set of labels/classes. We denote $\text{score}(s, y)$ the score of class y using features in state s. Given an instance $(\boldsymbol{x}_i, y_i)$, we define the margin in state s as $\text{score}(s, y_i) - \max_{y \in \mathcal{Y} \setminus \{y_i\}} \text{score}(s, y)$. At each time step t, we define the immediate reward r in state s_t after taking action a_t as

$$r(s_t, a_t) = \text{margin}(s_t, a_t) - \lambda \cdot \text{cost}(s_t, a_t) \quad (1)$$

Here $\text{margin}(s_t, a_t)$ and $\text{cost}(s_t, a_t)$ denote the margin and cost after adding the factor given by a_t respectively; λ is the trade-off parameter. When classifying using an incomplete feature set, we set values of non-selected features to be zero. Using a sparse feature vector also improves classification efficiency at test time.

3. Imitation Learning for Dynamic Feature Selection

A typical approach to imitation learning is to predict the oracle's action by solving a sequence of multiclass classification problems. To apply supervised classification methods, we define a forward-selection oracle that generates labels and a feature map that describes the state.

3.1. Imitation Learning via Classification

In a typical imitation learning task, at training time we have an oracle to demonstrate optimal actions that maximize the reward. Then we collect a set of trajectories generated by the oracle. The agent attempts to imitate the oracle's behavior without any notion of the reward function. Thus maximizing the expected reward is reduced to minimizing a *surrogate loss* with respect to the oracle's policy .

To mimic the oracle's behavior, we train a multiclass classifier to predict the oracle action. Let s_π denote states visited by π. We collect training examples $\{(\phi(s_{\pi^*}), \pi^*(s_{\pi^*}))\}$ by running the oracle, where ϕ is a feature map describing the state.

We denote Π the policy space and $\ell(s, \pi)$ the surrogate loss (classification loss) of π with respect to π^*. Using any standard supervised learning algorithm, we can learn a policy (*action classifier*)

$$\hat{\pi} = \arg\min_{\pi \in \Pi} \mathbb{E}_{s_{\pi^*}}[\ell(s, \pi)] \quad (2)$$

Here $\ell(s, \pi)$ can be any loss function used by the chosen classifier, for example, hinge loss in SVM. Let $J(\pi)$ be the task loss (negative reward) that we actually want to minimize. Denote T the task horizon. We have the following guarantee:

Theorem 1. *(Ross & Bagnell, 2010) Let* $\mathbb{E}_{s_{\pi^*}}[\ell(s, \pi)] = \epsilon$, *then* $J(\pi) \leq J(\pi^*) + T^2\epsilon$.

This theorem shows that we can bound the task loss by how well the agent mimics the oracle.

3.2. Oracle Actions

Ideally, an oracle action should lead to a subset of features having the maximum reward. However, we have too large a state space to search for the optimal subset of features exhaustively. In addition, given a state, the oracle action may not be unique since the optimal subset of features does not have to be selected in a fixed order.

We address the problem by using a greedy forward-selection oracle. At time step t, the oracle iterates

through the action space A_t and calculates each action's reward $r(s_t, a)$ $(a \in A_t)$ in state s_t; it then chooses the action that yields the maximum immediate reward. To identify the stop point, the oracle continues adding factors until all are selected. It then set the action in the maximum-reward state to be stop. Formally, let $a_t^* = \arg\max_{a \in A_t} r(s_t, a)$ and $r_t^* = r(s_t, a_t^*)$. This gives us a trajectory $\tau = \langle s_0, a_0^*, r_0^*, \ldots, s_T, a_T^*, r_T^* \rangle$. Let $r_{\max}$ be the maximum reward in T step. We define the oracle's policy as

$$\pi^*(s_t) = \begin{cases} a_t^* & \text{if} \quad r(s_t, a_t^*) < r_{\max} \\ \text{stop} & \text{otherwise} \end{cases} \tag{3}$$

In other words, the oracle stops in the maximum-reward state. Adding factors after the stop action will decrease the reward.

3.3. Policy Features

We define $\phi(s)$ as concatenation of features in the current state and meta-features that provide information about previous classification results and cost. More specifically, we have the following meta-features: confidence score given by the data classifier; change in confidence score after adding the previous factor; boolean bit indicating whether the prediction changed after adding the previous factor; cost of the current feature set; change in cost after adding the previous factor; cost divided by confidence score; current guess of the model. As $\phi(s)$ can contain first-order history information along the trajectory, predicting each action in turn allows the learner to learn dependencies between actions implicitly.

4. Iterative Policy Learning

One drawback of the above approach is that it ignores difference between state distribution of the oracle and the agent. When it cannot mimic the oracle perfectly (i.e. classification error occurs), the wrong action will change the following state distribution. Thus the learned policy is not able to handle situations where the agent follows a wrong path that is never chosen by the oracle. In fact in the worst case, performance can approach random guessing, even for arbitrarily small ϵ (Kääriäinen, 2006).

This problem can be alleviated by iteratively learning a policy trained under states visited by both the oracle and the agent. For example, during learning one can use a "mixture oracle" that at times takes an action given by the previous learned policy (Daumé III et al., 2009). Alternatively, at each iteration one can learn a policy from trajectories generated by all previous policies (Ross et al., 2011).

4.1. Dataset Aggregation

In its simplest form, the Dataset Aggregation (DAgger) algorithm (Ross et al., 2011) works as follows. In the first iteration, we initialize π_1 to π^* and collect training set $\mathcal{D}_1 = \{(\phi(s_{\pi^*}), \pi^*(s_{\pi^*}))\}$ from the oracle to learn a policy π_2. In the next iteration, we collect trajectories by executing π_2 and label $\phi(s_{\pi_2})$ with the oracle action, i.e. $\mathcal{D}_2 = \{(\phi(s_{\pi_2}), \pi^*(s_{\pi_2}))\}$; π_3 is then learned on $\mathcal{D}_1 \bigcup \mathcal{D}_2$. We repeat this process for several iterations. At each iteration the policy is trained on datasets collected from all previous policies. Intuitively, this enables it to make up for past failures to mimic the oracle. Algorithm 1 shows the training process.

Let $Q_t^{\pi'}(s, \pi)$ denote the t-step cost of executing π in the initial state and then running π'. We assume that if π picks a different action from π^*, it results in at most loss u along the trajectory. Suppose $\ell(s, \pi)$ is a convex loss upper bounding the 0-1 loss, which is common for most classification algorithms. We can generalize Theorem 1 to policy running under its own induced state distribution:

Theorem 2. *(Ross et al., 2011) Let* $\mathbb{E}_{s_\pi}[\ell(s, \pi)] = \epsilon$ *and* $Q_{T-t+1}^{\pi^*}(s, \pi) - Q_{T-t+1}^{\pi^*}(s, \pi^*) \leq u$*, then* $J(\pi) \leq J(\pi^*) + uT\epsilon$*.*

Let $\epsilon_N = \min_{\pi \in \Pi} \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{s_{\pi_i}}[\ell(s, \pi)]$ be the minimum loss we can achieve in the policy space Π. We denote the sequence of learned policies $\pi_1, \pi_2, \ldots, \pi_N$ by $\pi_{1:N}$. Ross *et al.* showed that for DAgger, there exists a policy $\pi \in \pi_{1:N}$ such that $\mathbb{E}_{s_\pi}[\ell(s, \pi)] \leq \epsilon_N + O(1/T)$. More specifically, applying Theorem 2, in the infinite sample case we have

Theorem 3. *(Ross et al., 2011) For DAgger, if* $Q_{T-t+1}^{\pi^*}(s, \pi) - Q_{T-t+1}^{\pi^*}(s, \pi^*) \leq u$ *and* N *is* $\tilde{O}(uT)$*, there exists a policy* $\pi \in \pi_{1:N}$ *s.t.* $J(\pi) \leq J(\pi^*) + uT\epsilon_N + O(1)$*.*

This theorem holds in the finite sample case as well. Readers are referred to (Ross et al., 2011) for detailed analysis.

4.2. DAgger with Coaching

In most cases, our oracle can achieve high accuracy with rather small cost. Considering a linear classifier, as the oracle already knows the correct class label of an instance, it can simply choose, for example, a positive feature that has a positive weight to correctly classify a positive instance. In addition, in the start state even when $\phi(s_0)$ are almost the same for all instances, the oracle may tend to choose factors that favor the instance's class. Since the optimal policy space is far

Algorithm 1 DAgger for Feature Selection

Input: $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$
Initialize $\mathcal{D} \leftarrow \emptyset$
Initialize $\pi_1 \leftarrow \pi^*$
for $i = 1$ **to** N **do**
  $\mathcal{D}_i \leftarrow \emptyset$
  for $j = 1$ **to** n **do**
    Remove factors from $\boldsymbol{x}_j$
    Sequentially add factors to $\boldsymbol{x}_j$ until stop
    $\mathcal{D}_i = \mathcal{D}_i \bigcup \{(\phi(s_{j\pi_i}), \pi^*(s_{j\pi_i}))\}$
  end for
  $\mathcal{D} = \mathcal{D} \bigcup \mathcal{D}_i$
  Train classifier π_{i+1} on $\mathcal{D}$
end for
Return best π evaluated on validation set

from the learning policy space and some environment information known by the oracle cannot be sufficiently represented by the policy feature, the oracle's behavior is too good to imitate for the learner. In the experiment, we observe a substantial gap between the oracle's performance and the agent's.

We address this problem by defining a *coach* $\tilde{\pi}^*$ in place of the oracle. The coach demonstrates suboptimal actions that are not much worse than the oracle action but are easier to learn within the learner's ability. Let $\text{score}_\pi(a)$ be a measure of how likely π chooses action a, such as confidence level given by the action classifier. Similar to Chiang et al. (2009), we define a *hope action* that combines the task loss and score given by the current policy.

$$\tilde{a}_t^* = \arg\max_{a \in A_t} \eta \cdot \text{score}_{\pi_i}(a) + r(s_t, a) \tag{4}$$

Our intuition is that when the learner has difficulty following the teacher, instead of being authoritative, the teacher should lower the goal properly. We use $\tilde{a}_t^*$ that the current policy prefers *and* has a relatively high reward, because a_t^* may not be achievable within the agent's learning ability. The parameter η specifies how permissive the coach is for allowing the agent to follow its will if this helps increase the reward. We gradually shrink η to let the coach approach the oracle. In this way we avoid the situation where an oracle action is far from what the model prefers that causes drastic change to the policy. It is hoped that gradually the learner can achieve the original goal in a more stable way.

5. Experimental Results

We perform experiments on three UCI datasets: radar signal (binary), digit recognition (10 classes) and image segmentation (7 classes). Our baselines are two static incremental feature selection methods. Both use a fixed queue of features and add them one by one. The first ranks features according to standard forward feature selection algorithm without any notion of the cost. The second uses a cost-sensitive ranking criteria: w_f/cost, where w_f is the weight of a factor f given by the data classifier. The weight is defined by the maximum absolute value of its features.

5.1. Experiment Setting

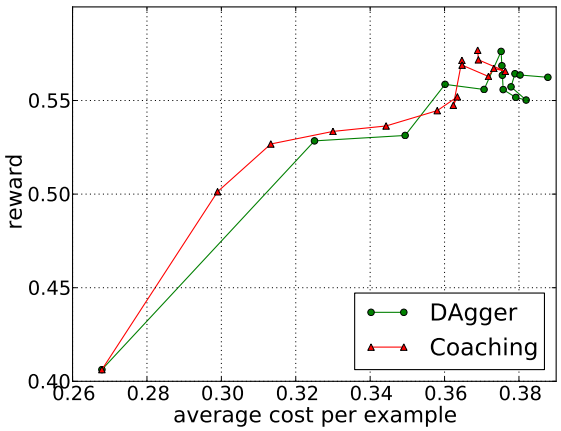
For all datasets, the data classifier are trained using MegaM (Daumé III, 2004). However, since we assume the provided classifier is to be used at test time, using it at training time may cause difference in the distribution of training and test data for feature selection. For example, the confidence level in $\phi(s)$ during training can be much higher that that during testing. Therefore, similar to cross validation, we split the training data into 10 folds. We collect trajectories on each fold using a data classifier trained on the other 9 folds. This provides a better simulation of the environment at test time.

For the digit dataset, we split the 16×16 image into non-overlapping 4×4 blocks and each factor contains the 16 pixel values in a block. For the other two datasets, each factor contains one feature.

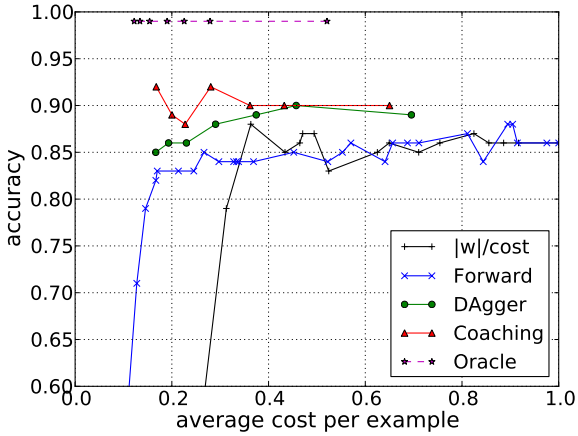
We choose 7 values (0, 0.1, 0.25, 0.5, 1, 1.5, 2) for the trade-off parameter λ. The base classifier in is a linear SVM trained by Liblinear (Fan et al., 2008). We run for 15 iterations and use the best policy tested on a development set. For coaching, we set the initial η to be 0.5 and decrease it by e^{-t} in each iteration.

5.2. Result Analysis

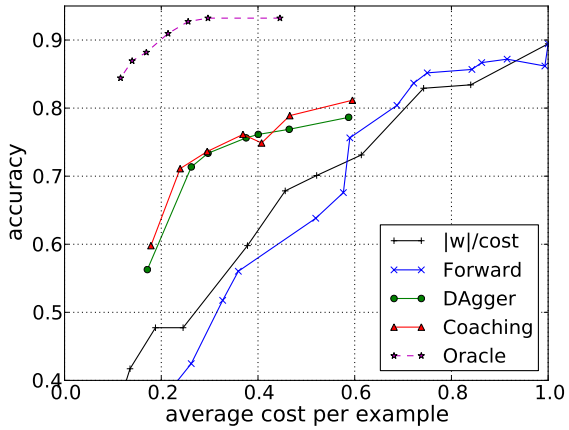
We first compare the learning curve of DAgger and Coaching over 15 iterations on the digit dataset with $\lambda = 0.5$ in Figure 1(a). We can see that DAgger makes a big improvement in the second iteration, while Coaching takes smaller steps but achieves higher reward gradually. In addition, the reward of Coaching changes smoothly and grows stably, which means it avoids drastic change of the policy. Figure 1(b) to Figure 1(d) show the accuracy-cost curves. We can see that our methods achieve comparable or even higher classification accuracy than using a complete set of features at a small cost. This can be explained by the dynamic selection scheme: for easy examples, we can make a decision with a small number of factors; only for hard examples do we need to acquire expensive factors. We also notice that there is a substantial gap between the learned policy's performance and the ora-



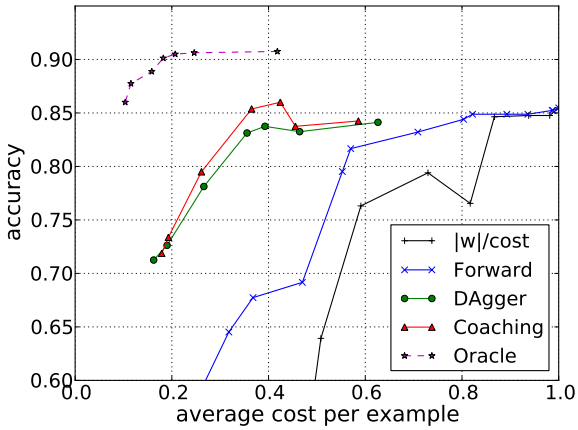
(a) Reward of DAgger and Coaching



(b) Radar dataset (32 factors).



(c) Digit dataset (16 factors).



(d) Segmentation dataset (19 factors).

cle's, however, in almost all settings Coaching achieves a higher reward.

6. Related Work

The work that has a problem setting most similar to ours is a recent study on active classification (Gao & Koller, 2010) in multiclass classification tasks. Based on value of information, they defined value of classifier to learn a probabilistic model that sequentially chooses which classifier to evaluate for each instance at test time. Our work is also related to budgeted learning. Kapoor & Greiner (2005) considered the problem of active model selection via standard reinforcement learning techniques. However, their results showed that it is inferior to simple and intuitive policies. Recently, Reyzin (2011) approached the problem by training an ensemble classifier consisting of base learners trained on each feature. This method is constrained to binary classification though.

7. Conclusion and Future Work

We propose a dynamic feature selection algorithm that automatically trades off feature cost and accuracy. We formalize it as an imitation learning problem and propose a coaching scheme when the optimal action is too good to learn. Experimental results show that our method achieves high accuracy with significant cost savings. One future direction is to explicitly include feature dependency and train the feature weights jointly with the selection policy. We are also interested in applying our method to structured prediction problems where policy features may require inference under selected features and cost may not be known until run time.

Acknowledgements

We thank Jiarong Jiang, Adam Teichert and Tim Vieira for helpful discussions that improves this paper.

References

Chiang, David, Knight, Kevin, and Wang, Wei. 11,001 new features for statistical machine translation. In *NAACL-HLT*, 2009.

Daumé III, Hal. Notes on cg and lm-bfgs optimization of logistic regression. 2004. Software available at http://www.cs.utah.edu/~hal/megam/.

Daumé III, Hal, Langford, John, and Marcu, Daniel. Search-based structured prediction. *Machine Learning Journal (MLJ)*, 2009.

Fan, Rong-En, Chang, Kai-Wei, Hsieh, Cho-Jui, Wang, Xiang-Rui, and Lin, Chih-Jen. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

Gao, Tianshi and Koller, Daphne. Active classification based on value of classifier. In *NIPS*, 2010.

Kääriäinen. Lower bounds for reductions. In *Atomic Learning Workshop*, 2006.

Kapoor, A. and Greiner, R. Reinforcement learning for active model selection. In *Proceedings of the 1st international workshop on Utility-based data mining*, pp. 17–23. ACM, 2005.

McAllester, D., Hazan, T., and Keshet, J. Direct loss minimization for structured prediction. In *NIPS*, 2010.

Reyzin, Lev. Boosting on a budget: sampling for feature-efficient prediction. In *ICML*, 2011.

Ross, Stéphane and Bagnell, J. Andrew. Efficient reductions for imitation learning. In *AISTATS*, 2010.

Ross, Stéphane., Gordon, Geoffrey J., and Bagnell, J. Andrew. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.