Model Selection by Linear Programming

Joseph Wang, Tolga Bolukbasi, Kirill Trapeznikov, and Venkatesh Saligrama

Boston University, USA

Abstract. Budget constraints arise in many computer vision problems. Computational costs limit many automated recognition systems while crowdsourced systems are hindered by monetary costs. We leverage wide variability in image complexity and learn adaptive model selection policies. Our learnt policy maximizes performance under average budget constraints by selecting "cheap" models for low complexity instances and utilizing descriptive models only for complex ones. During training, we assume access to a set of models that utilize features of different costs and types. We consider a binary tree architecture where each leaf corresponds to a different model. Internal decision nodes adaptively guide model-selection process along paths on a tree. The learning problem can be posed as an empirical risk minimization over training data with a non-convex objective function. Using hinge loss surrogates we show that adaptive model selection reduces to a linear program thus realizing substantial computational efficiencies and guaranteed convergence properties.

Keywords: test-time budget, adaptive model selection, cost-sensitive learning.

1 Introduction

Image recognition often relies on expensive intermediate visual processing tasks that can hinder test-time applicability. In automated systems, low-level representations (e.g., histograms of oriented gradients) typically incur a high computation cost and impact test time tractability. In crowdsourced systems, humans are paid to identify intermediate visual cues/attributes and can be prohibitively expensive for test-time.

On the other hand, we can leverage the fact that images exhibit wide diversity in complexity. Indeed, recognition for many typical instances can be performed to desired accuracy with relatively cheap models that utilize computationally inexpensive features or only a few expensive attributes. This key insight motivates our model selection policies that adapts to problem difficulty. We learn decision rules from training data, which when presented with a new example selects the most informative and cost-effective model for that example.

We describe our work in the context of handwriting recognition and scene categorization. In handwriting recognition the objective is to predict a word given a sequence of letter images. While a more complex model, that uses several feature types or processing at multiple resolutions yields better predictive

D. Fleet et al. (Eds.): ECCV 2014, Part II, LNCS 8690, pp. 647–662, 2014.

performance, the system suffers from the prohibitively slow computation time [24]. Scene recognition– another scenario where budget constraints arise–is a difficult task due to the large number of classes and interclass similarity [15]. Low-level features are often insufficient for acceptable performance; and high-level attributes crowdsourced by Amazon Mechanical Turk (AMT) are often used in predictive models incurring monetary costs. Due to the wide diversity of images, high-cost attributes/features are often unnecessary for many images to meet acceptable performance. Indeed "cheap" models can often be used for typical cases. The goal of this paper is to learn policies that adaptively utilizes cost-effective models while ensuring desired performance. If we represent an input data instance as $\mathbf{x}$, its unknown response as y and our adaptive selection system as $\mathbf{g}(\mathbf{x})$ then the high level objective is to minimize the average prediction error subject to an average budget B.

$$\min_{\mathbf{g}} \ \mathbf{E}\left[\ \text{error}(\mathbf{g}(\mathbf{x}), y)\ \right] \quad s.t. \ \ \mathbf{E}\left[\ \text{cost}(\mathbf{g}(\mathbf{x}))\ \right] \leq B$$

Several researchers have explored similar problems ([8,10,11,24]) which we will describe later.

The novel contribution that differentiates our work is a convex formulation for learning an adaptive model selector. We assume we are given a collection of precomputed models. Each model operates on features with different costs. Our decision system is described by a binary tree (see Fig.1). Each leaf corresponds to a particular model. Due to this structure, models can share features/attributes. The internal decision nodes route examples along the paths in a tree culminating in a model that is cost-effective while meeting desired accuracy levels.

Learning decision functions at each node of such a tree can be posed as an empirical risk minimization(ERM) problem that balances acquisition cost and misclassification error. We express ERM as an extremal(maxima) point of sums of indicator functions. This key transformation enables us to introduce convex surrogates for the indicator functions and, in turn, results in a convex objective. Without our transformation, direct substitution of surrogates in the original empirical risk results in a non-convex multi-linear formulation which is known to be NP-complete [14].

Next, by choosing a hinge loss for upper-bounding surrogate, we reduce the objective to a linear program (LP): a very well studied problem with strong convergence guarantees and efficient optimization algorithms. However, other convex surrogates are also possible and our formulation carries all the advantage of convex programming such as repeatability, global convergence and computation efficiency. In contrast, alternating non-convex optimization approaches [18,2,21] applied to similar problems do not have such guarantees.

1.1 Related Work

Our work is broadly related to detection cascades (see [20,27,5] and references therein) and the more recent work on classifier cascades [18]. Detection cascades

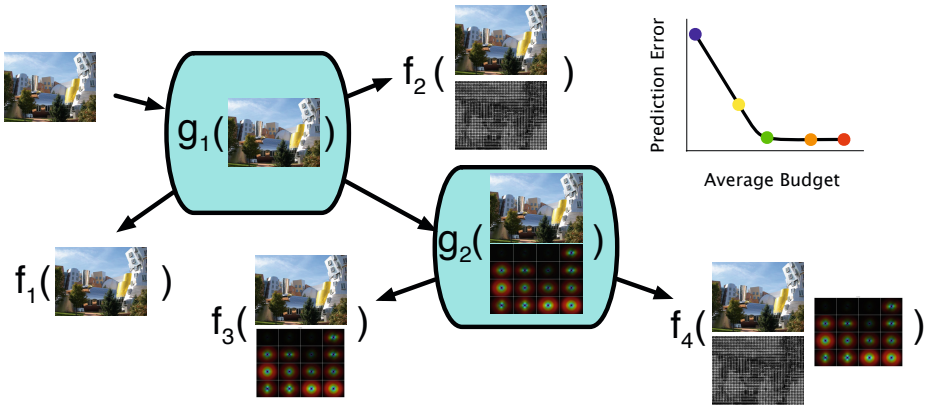


Fig. 1. An illustration of our model selection tree. We have access to four models: $f_1, \ldots f_4$. The models use a different combination of three representations: rgb, hog, and gist. The system has two internal decisions nodes. $g_1(rgb)$ uses the raw pixel values to either select a low cost model $f_1(rgb)$, medium cost model $f_2(rgb, hog)$ or delay the decision. The last decision node, $g_2(rgb, gist)$, acquired the gist feature selects between predicting with available information, $f_3(rgb, gist)$, or processing *hog* and predicting with the most expensive model, $f_4(rgb, gist, hog)$. The performance of an adaptive model selection can be represented by an budget vs error curve in the upper right corner. The colors correspond to different operating points as we vary the trade off between cost and error. The overall goal is to operate close the performance of the most complex model (red) with much lower budgets. The green point will an example of a desired system.

have been used for highly skewed problems for object detection and realize efficiency by using simpler models to reject examples as negative without needing to evaluate the more complex models farther along the cascade. Classifier cascades generalize this to multi-class scenarios with a series of increasingly informative models that adapt to problem difficulty. More recently, these ideas have also been generalized to cost-sensitive tree classifiers for web page ranking [26]. Our work differs from these contributions in several ways. First, our architecture is flexible and account for tree structures unlike cascades. Second, our approach can deal with a wider variety of prediction tasks including structured learning and sequence prediction with combinatorial output spaces in contrast to [20,27,5,18,26]. Finally, unlike much of this existing work that involve non-convex objectives and resort to alternative minimization schemes we formulate a globally convex objective with guaranteed convergence properties. We generalize the work on convex classifier cascades in [23] to more flexible architectures and broader range of prediction problems. Also, related convex optimization techniques were used by [22] in local learning problems.

Our work can be placed within the broader context of MDP approaches as well. MDP methods unlike ours do not assume fixed architectures. [10,9] apply an imitation learning (IL) algorithm introduced by [16] to the problem of feature

selection. IL learns decision functions that mimics an *oracle* on training data. Many issues arise in this context. We do not have access to an oracle in our setting of model selection. Furthermore, IL [9] requires generating arbitrary collection of states (candidate feature subsets) from training data to ensure a sufficiently rich collection of state-actions to mimic. Nevertheless this idea applied to our setting entails models that can take any arbitrary subset of features as inputs, which is not tractable. In contrast by employing a fixed acquisition architecture we only need a relatively small number of models that can be readily trained. Related to the IL approach is another direction based on reinforcement learning [12,4,7]. In lieu of an oracle the authors linearly parameterize a reward function and estimate it with standard RL techniques. However, the need for models that are customized to arbitrary subset of attributes remains as in IL.

Our work is closely related to dynamic model selection for structured prediction of [24]. There the authors combine the architecture of detection cascades with decision structure of RL. The authors define a value for selecting a more complex model to make predictions, and approximate the selection policy as a linear combination of meta-features computed on previous model outputs. The goal is to improve inference accuracy while satisfying a budget on a batch of test data. Our approach is more general. Instead of being limited to cascades, we have the ability to construct a binary tree architecture. Also, instead of a single policy that controls model selection at every step, we learn a separate decision function for every internal node of the tree. These advantages produce a more cost-effective model selection policy as we demonstrate in our experiments.

2 Empirical Risk Problem

In a typical learning problem, a data instance, $\mathbf{x} \in \mathcal{X}$ has a corresponding response $y \in \mathcal{Y}$. The goal is to learn a model $f(\mathbf{x}) \in \mathcal{Y}$ that correctly predicts the response variable y. For notational purposes we let $\mathcal{D}$ denote the unknown joint distribution for $(\mathbf{x}, y)$.

For example, in scene categorization, the objective is to predict a scene category, y, in an image $\mathbf{x}$. Here, the response space $\mathcal{Y}$ consists of L possible classes, $\{1, \ldots, L\}$. In structured prediction, the input, $\mathbf{x}$, is a sequence of handwritten letter images. The goal is predict the written word. Here, $\mathcal{Y}$ is a combinatorial output space consisting of all admissible letter sequences.

Each instance $\mathbf{x}$ is composed of M different vector-valued feature/attribute components. The mth feature component has an associated cost c_m. We assume we have access to K prediction models: $f_1(\mathbf{x}), \ldots, f_K(\mathbf{x})$ that are a priori given and fixed. The input to each model, $f_k(\cdot)$, is a sub-collection, S_k, of the M attributes or features. Each model has an associated cost of prediction: $\sum_{m \in S_k} c_m$. In addition, each model's prediction performance is evaluated with a loss function given the ground truth response variable: $\mathcal{L}(f(\mathbf{x}), y) \in \mathbb{R}^+$. For instance, in classification, the loss is simply a 0/1 error, $\mathcal{L}(f(\mathbf{x}), y) = \mathbb{1}_{[f(\mathbf{x}) \neq y]}$.

Our goal is to learn a decision system that dynamically selects one of these models for every instance $\mathbf{x}$. We represent our system as a binary tree. The

binary tree is composed of K leafs and $K-1$ internal nodes. At each internal node, $j = 1, \ldots, K-1$, is a binary decision function, $\text{sign}[g_j(\mathbf{x})] \in \{+1, -1\}$. This function determines which action should be taken for a given example. The binary decisions, $g_j(\mathbf{x})$'s, represent actions from the following set: stop and predict with the model that uses the current set of features or choose which feature to request next. Each leaf node, $k = 1, \ldots, K$, corresponds to a terminal decision of predicting with the model $f_k(\mathbf{x})$ based on the available information. For notational simplicity, we denote applying a decision node and a leaf model as $g_j(\mathbf{x})$ and $f_k(\mathbf{x})$ respectively. Note the functions implicitly operate only on the feature sets that have been acquired along the associated path to each node.

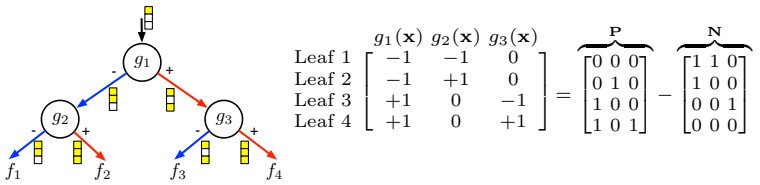


Fig. 3. An example decision system of depth two: node $g_1(x_1)$ selects either to acquire feature 2 for a cost c_2 or 3 for a cost c_3. Node $g_2(x_1, x_2)$ selects either to stop and predict with features $\{1, 2\}$ or to acquire 3 for c_3 and then terminate. Node $g_3(x_1, x_3)$ selects to predict with $\{1, 3\}$ or with $\{1, 2, 3\}$.

The objective is to learn the internal decision functions: $g_j(\mathbf{x})$'s. We define the system risk:

$$R(\mathbf{g}, \mathbf{x}, y) = \sum_{k=1}^{K} R_k(f_k, \mathbf{x}, y) G_k(\mathbf{g}, \mathbf{x}) \tag{1}$$

Here, $\mathbf{g} = \{g_1, \ldots g_{K-1}\}$ is the set of decision functions. $R_k(f_k, \mathbf{x}, y)$ is the risk of making a decision at a leaf k. It consists of two terms: loss of the model at the leaf and the cost of features corresponding to the sub-collection of attributes, S_k, acquired along the path from the root node to the leaf; and α is a parameter that controls trade-off between acquisition cost and model performance.

$$R_k(f_k, \mathbf{x}, y) = \mathcal{L}(f_k(\mathbf{x}), y) + \alpha \sum_{m \in S_k} c_m \tag{2}$$

$G_k(\mathbf{g}, \mathbf{x}) \in \{0, 1\}$ is a binary state variable indicating whether or not an instance $\mathbf{x}$ is terminated at the kth leaf. As illustrated in Fig. 3 we compactly encode the path from the root to every leaf in terms of internal decisions, $g_j(\mathbf{x})$'s, by two auxiliary binary matrices: $\mathbf{P}$, $\mathbf{N} \in \{0, 1\}^{K \times K-1}$. If $\mathbf{P}_{k,j} = 1$ then, on the path to leaf k, a decision node j must be positive: $g_j > 0$. If $\mathbf{N}_{k,j} = 1$ then on the path to leaf k, a decision at node j must be negative: $g_j \leq 0$. A kth row in $\mathbf{P}$ and $\mathbf{N}$ jointly encode a path from the root node to a leaf k. The sign pattern

for each path is obtained by $\mathbf{P} - \mathbf{N}$. Using this path matrix, the state variable can be defined:

$$G_k(\mathbf{g}, \mathbf{x}) = \prod_{j=1}^{K-1} [\mathbb{1}_{g_j(\mathbf{x})>0}]^{\mathbf{P}_{k,j}} [\mathbb{1}_{g_j(\mathbf{x})\leq 0}]^{\mathbf{N}_{k,j}} \tag{3}$$

Our goal is to learn decision functions $g_1, \ldots, g_{K-1}$ that minimize the expected system risk:

$$\min_{\mathbf{g}} \mathbb{E}_{\mathcal{D}} \left[R(\mathbf{g}, \mathbf{x}, y) \right] \tag{4}$$

However, the probability distribution $\mathcal{D}$ is assumed to be unknown and cannot be estimated reliably due to potential high-dimensionality of attributes. Instead, we are given a set of N training examples with full features, $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$. We approximate the expected risk by a sample average over the data and construct the following empirical risk minimization (ERM) problem:

$$\min_{\mathbf{g}} \sum_{i=1}^{N} R(\mathbf{g}, \mathbf{x}_i, y_i) = \sum_{i=1}^{N} \sum_{k=1}^{K} \overbrace{R_k(f_k, \mathbf{x}_i, y_i)}^{\text{risk of leaf } k} \underbrace{\prod_{j=1}^{K-1} [\mathbb{1}_{g_j(\mathbf{x}_i)>0}]^{\mathbf{P}_{k,j}} [\mathbb{1}_{g_j(\mathbf{x}_i)\leq 0}]^{\mathbf{N}_{k,j}}}_{G_k(\cdot) = \text{ state of } \mathbf{x}_i \text{ in a tree}} \tag{5}$$

Note that by the definition of risk in (1), the ERM problem can be viewed as a minimization over a function of indicators with respect to decisions: $g_1, \ldots, g_{K-1}$.

3 Model Selection by Linear Programming

A popular approach to solving ERM problems is to substitute indicators with convex upper-bounding surrogates, $\phi(z) \geq \mathbb{1}_{[z]}$ and then to minimize the resulting surrogate risk. However, this strategy leads to a non-convex, multi-linear optimization problem in our setting. Previous attempts to solve problems of this form have focused on computationally costly alternating minimization approaches [18,2,21] with no guarantees on optimality. A key point of this paper is that rather than attempting to solve this non-convex surrogate problem, we instead reformulate the indicator empirical risk in (5) as a maximization over sums of indicators before introducing convex surrogate. Our approach yields a globally convex upper-bounding surrogate of the empirical loss function.

3.1 Convex Risk Objective

In reformulating the risk, it is useful to define the "savings" for an example. The *savings*, π_k^i, for an example i, represents the difference between the worst case outcome, R_{max} and the risk $R_k(f_k, \mathbf{x}_i, y_i)$ for terminating at the kth leaf.

Intuitively R_{max} is the cost of incorrectly predicting with the most expensive model (the model that uses all the features): $R_{max} = \max_{y'} \mathcal{L}(y, y') + \alpha \sum_m c_m$.

$$\pi_k^i = R_{max} - R_k(f_k, \mathbf{x}_i, y_i) \tag{6}$$

Note that the savings do not depend on the decisions, $g_j's$, that we are interested in learning.

For a binary tree, $\mathcal{T}$, composed of $K-1$ internal nodes and K leaves, it turns out that the risk in (5) can be rewritten as a maxima of K terms. Each term is a weighted linear combination of indicators, and each weight corresponds to the *savings lost* if the decision inside the indicator argument is true. Before stating the result, we define the weights for the linear combination in each term of the max. For an internal node j, we denote C_j^n as the set of leaf nodes in a subtree corresponding to a negative decision $g_j(\mathbf{x}) \leq 0$. And C_j^p is the set of leaf nodes in a subtree corresponding to a positive decision. For instance in Fig. 1, $C_1^p = \{Leaf\ 3, Leaf\ 4\}$.

For a compact representation, recall that the kth rows in matrices $\mathbf{P}$ and $\mathbf{N}$ define a path to leaf k in terms of $g_1, \ldots, g_{K-1}$, and a non-zero $\mathbf{P}_{k,j}$ or $\mathbf{N}_{k,j}$ indicates if $g_j \lessgtr 0$ is on the path to leaf k. So for each $\mathbf{x}_i$ and each leaf k, we introduce two positive weight row vectors of length $K-1$:

$$\mathbf{w}_{n,k}^i = \mathbf{N}_{k,1} \sum_{l \in C_1^p} \pi_l^i, ..., \mathbf{N}_{k,K-1} \sum_{l \in C_{K-1}^p} \pi_l^i, \quad \mathbf{w}_{p,k}^i = \mathbf{P}_{k,1} \sum_{l \in C_1^n} \pi_l^i, ..., \mathbf{P}_{k,K-1} \sum_{l \in C_{K-1}^n} \pi_l^i \tag{7}$$

Using these weight definitions, the empirical risk in (5) can be rewritten as:

Lemma 31. *The empirical risk of tree $\mathcal{T}$ is:*

$$R(\mathbf{g}, \mathbf{x}_i, y_i) = R_{max} - \sum_{k=1}^{K} \pi_k^i + \max_{k \in \{1,\ldots,K\}} \mathbf{w}_{p,k}^i \begin{bmatrix} \mathbb{1}_{g_1(\mathbf{x}_i)>0} \\ \vdots \\ \mathbb{1}_{g_{K-1}(\mathbf{x}_i)>0} \end{bmatrix} + \mathbf{w}_{n,k}^i \begin{bmatrix} \mathbb{1}_{g_1(\mathbf{x}_i)\leq 0} \\ \vdots \\ \mathbb{1}_{g_{K-1}(\mathbf{x}_i)\leq 0} \end{bmatrix} \tag{8}$$

The proof of this lemma is included in the Supplementary Material

The jth component of $\mathbf{w}_{n,k}^i$ multiplies $\mathbb{1}_{[g_j(\mathbf{x}_i)\leq 0]}$ in the term corresponding to the kth leaf. For instance in our four leaf example in Fig. 1, the weight multiplying $\mathbb{1}_{[g_1(\mathbf{x}_i)\leq 0]}$ is the sum of these savings for leaves 3 and 4 (i.e. savings lost if $g_1 \leq 0$). $\left(\mathbf{w}_{n,1}^i\right)_1 = \pi_3^i + \pi_4^i$. Therefore, sets C_j^p, C_j^n define which π_k^i's contribute to a weight for a decision term. If $\mathbf{P}_{k,j}$ or $\mathbf{N}_{k,j}$ is zero then decision $g_j \gtrless 0$ is not on the path to leaf k and the weight is zero.

Intuitively, the empirical risk in (8) represents a scan over the paths to each leaf ($k = 1, \ldots, K$), and each term in the maximization encodes a path to one of the K leaves. The active term in the maximization corresponds to the leaf to which an observation is assigned by the decision functions $g_1, \ldots, g_{K-1}$. Additionally, the weights on the indicators represent the *savings lost* if the argument of the indicator is active. In our example, if the decision function $g_1(\mathbf{x}_i)$ is negative, leaves 3 and 4 cannot be reached by $\mathbf{x}_i$, and therefore π_3^i and π_4^i, the savings associated with leaves 3 and 4, cannot be realized and are lost.

An important observation is that each term in the max in (8) is a linear combination of indicators instead of a product as in (5). This transformation enables us to upper-bound each indicator function with a convex surrogate, $\phi(z)$: $\phi[g_j(\mathbf{x})] \geq \mathbb{1}_{[g_j(\mathbf{x})>0]}$, $\phi[-g_j(\mathbf{x})] \geq \mathbb{1}_{[g_j(\mathbf{x})\leq 0]}$. And the result is a novel convex upper-bound on the empirical risk in (8). We denote this risk as $R_\phi(\mathbf{g})$. And the optimization problem over a set of training examples, $\{\mathbf{x}_i, y_i\}_{i=1}^N$ and a family of decision functions $\mathcal{G}$:

$$\max_{\mathbf{g}\in\mathcal{G}} \sum_{i=1}^{N} R_\phi\left(\mathbf{g}, \mathbf{x}_i, y_i\right) \tag{9}$$

3.2 Linear Programming

There are many valid choices for the surrogate $\phi(z)$. However, if a hinge loss is used as an upper bound and $\mathcal{G}$ is a family of linear functions of the data then the optimization problem in (9) becomes a linear program (LP).

Proposition 32. *For* $\phi(z) = \max(1-z, 0)$ *and linear decision functions* $g_1, \ldots, g_{K-1}$*, the minimization in* (9) *is equivalent to the following linear program:*

$$\min_{\substack{g_1,\ldots,g_{K-1},\gamma^1,\ldots,\gamma^N \\ \alpha_1^1,\ldots,\alpha_{K-1}^N,\beta_1^1,\ldots,\beta_{K-1}^N}} \sum_{i=1}^{N} \gamma^i \quad \textit{subject to:} \tag{10}$$

$$\gamma^i \geq \mathbf{w}_{p,k}^i \begin{bmatrix} \alpha_1^i \\ \vdots \\ \alpha_{K-1}^i \end{bmatrix} + \mathbf{w}_{n,k}^i \begin{bmatrix} \beta_1^i \\ \vdots \\ \beta_{K-1}^i \end{bmatrix}, i \in [N],\ k \in [K]$$

$$1 + g_j(\mathbf{x}_i) \leq \alpha_j^i, \quad 1 - g_j(\mathbf{x}_i) \leq \beta_j^i, \qquad \alpha_j^i \geq 0, \quad \beta_j^i \geq 0,$$

$$j \in [K-1], i \in [N]$$

We introduce the variable γ^i for each example $\mathbf{x}_i$ to convert from a maximization over leaves to a set of linear constraints. Similarly, the maximization within each hinge loss is converted to a set of linear constraints. The variables α_j^i upper-bound the indicator $\mathbb{1}_{g_j(x_i)>0}$ and the variables β_j^i upper-bound the indicator $\mathbb{1}_{g_j(x_i)\leq 0}$. Additionally, the constant terms in the risk are removed for notational simplicity, as these do not effect the solution to the linear program. For details please refer to Suppl. materials.

Complexity: Linear programming is a relatively well-studied problem, with efficient algorithms previously developed. Specifically, for K leaves, N training points, and a maximum feature dimension of D, we have $O(KD+KN)$ variables and $O(KN)$ constraints. The state of the art primal-dual methods for LP are fast in practice, with an expected number of iterations $O(\sqrt{n}\log n)$, where n is the number of variables [1].

Kernelization: Our formulation can handle more complex decision functions $\mathbf{g}(\mathbf{x})$ by kernelization. The observations x_i are replaced in the LP by $\psi(\mathbf{x}_i)$ for

some expanded basis function $\psi(\cdot)$. For expanded basis functions, a natural solution is to add ℓ_2 regularization on the decision functions, converting the LP to a quadratic program. Addition of ℓ_2 regularization removes non-unique solutions, with solution of the regularized problem equal to the minimum norm solution of the unregularized problem (for a sufficiently small regularization parameter value). Furthermore, the ℓ_2 regularization allows for the problem to be kernelized, as the optimization can be expressed with respect to expanded basis inner products of the form $\psi(\mathbf{x}_i)^T\psi(\mathbf{x}_j)$ in the dual problem. While this is possible, yielding a quadratic optimization problem in place of the proposed LP, empirical evidence indicates that on real-world data the family of linear and low-order polynomial decision functions is sufficiently rich and therefore we do not explore kernelization in the experimental section.

Algorithm 1. Model Selection by LP

INPUT: $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_K(\mathbf{x})$ {Models}; $S_1, S_2, \ldots, S_K$ {Features used by each model}; $\mathbf{P}, \mathbf{N}$ {Tree structure}; $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$ {Training Data}; α {Trade-off parameters}
for $(i, k) = \{1, \ldots, N\} \times \{1, \ldots, K\}$ **do**
  Compute savings in (6): $\pi_k^i \leftarrow R_{max} - R_k(f_k, \mathbf{x}_i, y_i)$
  Compute weight vectors in (7): $\mathbf{w}_{n,k}^i, \mathbf{w}_{p,k}^i$
end for
Solve linear program in (10): $[g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_{K-1}(\mathbf{x})] \leftarrow LPsolver(\{\mathbf{w}_{n,k}^i, \mathbf{w}_{p,k}^i\})$
OUTPUT: Model Selection Tree: $\mathbf{g}(\mathbf{x})$

4 Experiments

We demonstrate our LP model selection approach in Algorithm 1 on two important prediction tasks in computer vision. First, we apply our method to the problem of structured prediction. We use the handwriting dataset for word prediction and compare our method to the RL based model selection ([24]). Here, the cost is computation time for processing HOG transforms of different scales. For the second experiment, we apply our method to the SUN scene categorization [15] dataset. Here instead of using image processing features, we use human generated descriptor as inputs to a classifier. In this set-up, the cost of feature acquisition is the monetary value paid to Amazon Mechanical Turk workers.

Performance Metric: Our goal is to train a set of decision functions for a fixed tree that minimizes prediction loss subject to an average budget constraint. We examine average acquisition cost vs. average prediction loss to compare performance of the proposed LP approach. We sweep over values of the tradeoff parameter α in order to learn systems of varying average budget, resulting in a series of learned trees of differing prediction rates and average budgets. Increasing the value of α biases the system to learn decisions with low average

acquisition cost with an increased system error, while decreasing the value α yields systems with smaller error at the expense of an increase in cost. Although a system may not be learned that exactly matches a desired budget, any point in the convex hull of budget/error points learned is achievable by weighted randomization over learned systems. As a result, we take the lower convex hull of points in the space of average error vs. average cost to learn a decision system for any average budget. Note that in the experimental results, a convex hull over the training points is taken, with the corresponding policies applied to unseen test data, and therefore the resulting curve is not necessarily a convex hull. In all experiments, we first divide the data into 10 training/test folds. Within each fold, we further divide the training data of each fold into 10 sub-folds. In these sub-folds, we use all but one subfold to train the models, and apply this learned predictor to estimate the losses for the unused subfold. These sub-folds are used to more accurately represent the prediction ability of the models for learning our adaptive system.

Leaf Models: Each individual leaf model, $f_1, \ldots, f_K$, operates on a subset of the features acquired on the path to that leaf. We assume f_k's are pre-computed prior to learning the decision system. The goal of our paper is to demonstrate the advantage of an adaptive selection system therefore we do not seek to learn the most accurate models. We simply illustrate the gain in relative performance: same level of accuracy as the most complex model achieved with lower budgets.

4.1 Model Selection in Structured Learning

Structured Learning Problem: In structured learning, the goal is to learn a model from a set of training samples that maps inputs $x \in \mathcal{X}$ to the outputs $y \in \mathcal{Y}$. In a typical structured prediction setup [19], the response space $\mathcal{Y}$ is not simply a discrete label but instead a more complex structured output. In particular, we focus on the problem of predicting words from handwritten characters, where the output space, $\mathcal{Y}$, is a string of letters of varying length. The goal is to learn a scoring function, $\Psi_w(\mathbf{x}, y)$, over training data such that the prediction model, $f(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} \Psi_w(\mathbf{x}, y)$ matches the given training structure.

We use a function $\Psi_w(\mathbf{x}, y) = \mathbf{w} \cdot \mathbf{h}[\mathbf{x}, y]$ which is linear in the score features $\boldsymbol{h} : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^p$. In general, there are exponentially many outputs and solving this inference is computationally infeasible. However, $\boldsymbol{h}$ is usually constructed to decompose over subsets of y that enables this problem to be solved efficiently. In our experiment, we adopted a first-order linear conditional random field model which is commonly used in optical character recognition (OCR) tasks [24,13]. In this model, the score features decompose into sum of unary and pairwise terms. Given an input character image sequence of $\mathbf{x} = \{x(1), ..., x(l)\}$, the score of output sequence can be written as, $\Psi_w(\mathbf{x}, y) = \sum_{j=1}^{\ell} \boldsymbol{h}[x(j), y(j)] \cdot \boldsymbol{w} + \sum_{j=2}^{\ell} \boldsymbol{h}[\boldsymbol{x}, y(j-1), y(j)] \cdot \boldsymbol{w}$, where $y(1), ..., y(\ell)$ are labels for individual characters in the word y. Given the weight vector $\boldsymbol{w}$, we use max-sum algorithm [3] to solve the inference problem. To learn the weight vector

$\boldsymbol{w}$, we solve maximum conditional likelihood using stochastic gradient descent. We used the implementation in [13] for this purpose.

Dataset and Simulation Details: We used the OCR data set from [17]. This data set has 6,877 handwritten words where each word is represented as sequence of 16x8 binary letter images. There are 55 unique words, 26 unique characters and 55,152 letters.

Following [17], we use three sets of features: raw images, histogram of gradients (hog1) [6] computed in 3x3 bins and a finer HoG computed on 2x2 bins (hog2). We train three CRF models: $f_{rgb}, f_{rgb,hog1}, f_{rgb,hog2}$. Note that once hog2 is computed, hog1 does not add additional information. The computational cost of processing the raw images is assumed to be negligible, while the computational cost of the 2x2 and 3x3 HOG features are assumed to be equal and proportional to the length of the word.

The goal is to learn a system to minimize character recognition error subject to an average computational cost constraint per letter. We train two architectures: a two stage cascade and a three decision node binary tree as illustrated in Fig. 4a. Note the tree allows greater flexibility by allowing us to acquire hog2 directly from rgb while a cascade has to acquire hog1 before processing hog2.

Following the framework presented in [24], the decision functions in our LP tree also act on meta features as opposed to the raw features. These meta features reflect the fit of the structured predictor $f_{(\cdot)}$ to the training set population. The meta-features used are the difference in the score for the top two sequence predictions, the average of the min/max and mean entropies of the marginal distributions as predicted at each position in the word by the predictor at that stage. Additional meta-features count the number of times a 3-gram 4-gram and 5-gram are predicted but never occur in the training set.

Dynamic Model Selection Baseline: We compare our approach to dynamic structured model selection method (DMS) in [24]. There the authors employ a cascade architecture with models arranged sequentially in the order of increasing cost, and learn a policy that controls whether an example should be predicted using the current model or rejected to the next more expensive model. For their DMS architecture, we use the same cascade as for our approach.

The authors define the value of delaying a decision as a decrease in the loss when a sample is moved from stage i to $i+1$. This value function is modeled as a linear combination of the meta-features. The policy then sends the instance that suffer the maximum predicted loss reduction to the next stage until a predetermined budget limit is hit. The value of skipping a stage is defined as: $V(f_i, \boldsymbol{x}, \boldsymbol{y}) = L(f_{i-1}(\boldsymbol{x}), \boldsymbol{y}) - L(f_i(\boldsymbol{x}), \boldsymbol{y})$. The policy parameter β is found by ridge regression, $\arg\min_\beta \lambda||\beta||_2^2 + \sum_{i=2}^{3}\sum_{j=1}^{n}(V(f_i, \boldsymbol{x}_j, y_j) - \beta^T\phi(\boldsymbol{x}_j, f_{1:i-1}))^2$, where ϕ denotes the meta-features for given sample and stage predictors. The test time value is then defined as $J(\tau_1, ..., \tau_n, \eta) = \sum_{j=1}^{n}\sum_{i=2}^{\tau_j}\beta^T\phi(x^j, f_{1:i-1})$, where τ_j denotes how many features are computed for example j. During test time, the total value is greedily maximized until the budget constraint B prevents any other features from being computed.

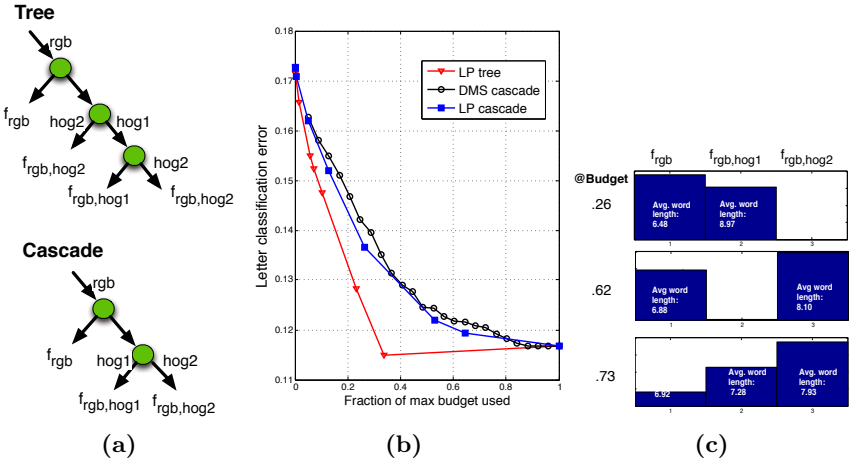


Fig. 4. (a) shows two system structures used in the OCR experiment: a two stage cascade and a three node tree. In (b), we display the budget vs error plot for three methods: cascade and tree architecture of our LP model selection system and a DMS cascade system. While performance of LP and DMS are on par in the cascade structure, LP Tree has a significant advantage over DMS. LP tree achieves same accuracy with a significant speedup and lower computation cost($\geq$70% savings). Panel (c) displays the distribution of examples that end up at 3 different stages/models in the LP cascade at three budget operating points. For the tree, this is not illustrative since the best accuracy is achieved at .33 budget point.

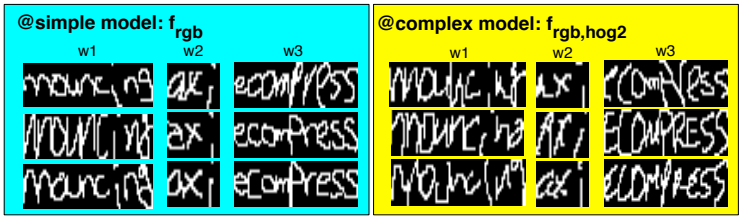


Fig. 5. Here, we examine the histogram at .62 budget from Fig. 4c. We provide examples of three different words being classified at the cheapest/simplest model (f_{rgb}) and at the complex/expensive model ($f_{rgb,hog2}$). As expected, more obscure and rotated words require the complex model.

Discussion: We report average error for different values of average budget (see Fig. 4b). For simplicity, we normalize the units to the fraction of the maximum budget allowed. For example, if the system operates at budget 1 then every example is routed to the most expensive model $f_{rgb,hog2}$ (the best accuracy). For budget 0, every example remains with the cheapest model, f_{rgb}. An adaptive system with a budget between 0 and 1, utilized the cheap model for some examples and the expensive model for others resulting in a lower budget but accuracy equivalent to the expensive model.

The experiments clearly highlight the advantages of our approach. Our LP cascade performance matches the accuracy for all budget values of a DMS cascade. However, when we introduce a more flexible tree architecture instead of a cascade, the performance dramatically improves. Our LP tree exhibits significant computational-savings($\geq 70\%$) and speedup to achieve similar accuracy as a DMS cascade. In a cascade, an example cannot go directly to the most complex model, $f_{rgb,hog2}$ while in a tree this decision is possible and results in higher cost efficiency. In addition, our approach learns a separate decision for every internal node in the tree allowing for more complex selection functions. In contrast in DMS, the same policy function is used at every stage of the cascade limiting the discriminative power of the decision system. Note that DMS does not generalize to trees in an obvious way since it is in essence an early stopping policy.

In addition to the error vs budget performance, we explore the distribution of examples that are being routed to the three models in our LP cascade architecture. We examine systems corresponding to budgets: .26,.62 and .73. As expected at a budget of .26, model utilization is evenly distributed between the cheapest, f_{rgb} and the medium complexity model, $f_{rgb,hog1}$. At the other end of the spectrum, at a higher budget of .73, most examples are being routed to the most expensive/complex model, $f_{rgb,hog2}$. However, in the middle of the spectrum at the .62 budget system, for half of the examples, f_{rgb} is being utilized and the rest are routed to the last model $f_{rgb,hog2}$. This however may not be that surprising. Since the performance of hog1 and hog2 are similar, the system decides to use the more expensive feature. We do not explore the distributions for the LP Tree since the best performance is already achieved at a .33 budget.

We also report the average word length that each model sees. As expected longer (presumably harder to classify) end up at a later more complex model. We next look at the actual images being classified at the cheapest (simplest) model (f_{rgb}) and at the most expensive, $f_{rgb,hog2}$ levels. We look at different instances of the same word. Fig. 5 illustrates more obscure instances of the same word are routed to the last stage (the most complex/expensive model).

4.2 Scene Recognition

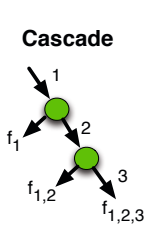
Next, we apply our system to another challenging task in computer vision: scene recognition. The problem can be posed as multi-class classification problem, where $\mathbf{x}$ is an image of a scene, and y is one of L scene categories. We focus on the popular scene dataset SUN [25].

The difficulty in this problem is due to several factors. First, the number of classes, L, is very large, $L > 700$, and the number of examples per class is small, 20. Partly due to this data limitation and the difficulty of the task itself, automatic visual recognition features such as HoG or GIST do not achieve suitably high accuracy rates. In an attempt to improve performance, authors in [15] proposed exploring human annotated attributes. Amazon Mechanical Turk workers were asked to vote whether images fit certain descriptions such as: camping,

cluttered space, fire. For each attribute, an average of three votes is reported, with a total of 102 attributes. The attributes are then grouped into three sets: (1), functions (camping, hiking, biking...), (2), materials (trees, clouds, grass,...), and (3) surface/spatial properties (dirty, glossy, rusty,open area, warm,...).

To simplify our experiment, we use the second level of the class hierarchy, which groups the scenes into more general categories, and then we discard the indoor categories, resulting in only 10 classes. From this data, we randomly construct an even training/test split, resulting in around 400 training and 400 test points per class.

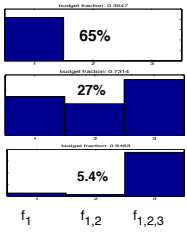
We then train 3 models: $f_1, f_{1,2}, f_{1,2,3}$, with the subscripts indicating which attribute groups are used to construct the model. Since attributes are acquired by paying AMT workers, the goal is to make accurate predictions while using the smallest number of total attributes. Additionally, due to the nature of the system, dynamic model selection must be performed in a streaming test data setting as opposed to collecting data for all test examples before acting.



(a)

	Cost Savings (%)		
Accuracy Loss	LP	DMS	Fixed
0%	**5.4**	3	0
1%	**27**	12	6
2%	**32**	**32**	24
10%	**65**	**65**	**65**

(b)



(c)

Fig. 6. SUN Scene Categorization Results. (a) shows cascade structure used in the experiment. In (b), we report cost savings for four accuracy levels. Loss is the difference between the accuracy of the most complex model and the dynamic model at different budget points. Cost savings is the percent saved from the most expensive model. In (c), we display example distribution among stages.

Fig. 7. Sample images from the railroad category sent to different models

We compare the performance of our system to a DMS cascade and non-adaptive fixed-length systems. Note that the DMS cascade cannot be applied to individual test examples, as the system ceases to dynamically select the models. To accommodate for this shortcoming, we randomly partition the test data into subsets of 10 examples, with performance of the learned DMS cascades averaged over all subsets. In contrast, the system learned using the proposed LP operates on single examples allowing for streaming/parallel application as opposed to a batch/centralized strategy.

Table 6b compares change in classification accuracy vs. cost reduction for the three approaches. For all 4 changes in classification accuracy, the proposed LP approach matches or exceeds the performance of the fixed-length systems or the adaptive DMS system. In particular, the proposed LP approach produces an adaptive system that reduces the budget by 27% while reducing accuracy by only 1%. In comparison, the DMS cascade is only able to reduce the budget by 12% when maintaining a classification performance within 1% of the full system.

References

1. Anstreicher, K.M., Ji, J., Potra, F.A., Ye, Y.: Probabilistic analysis of an infeasible-interior-point algorithm for linear programming. Math. Oper. Res. 24(1), 176–192 (1999)
2. Bennett, K.P., Mangasarian, O.L.: Bilinear separation of two sets in n-space. Computational Optimization and Applications 2 (1993)
3. Bishop, C.M., et al.: Pattern recognition and machine learning, vol. 1. Springer, New York (2006)
4. Busa-Fekete, R., Benbouzid, D., Kégl, B.: Fast classification using sparse decision dags. In: 29th International Conference on Machine Learning (ICML) (2012)
5. Chen, M., Xu, Z., Weinberger, K.Q., Chapelle, O., Kedem, D.: Classifier cascade: Tradeoff between accuracy and feature evaluation cost. In: International Conference on Artificial Intelligence and Statistics (AISTATS), pp. 235–242 (2012)
6. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005, vol. 1, pp. 886–893. IEEE (2005)
7. Dulac-Arnold, G., Denoyer, L., Preux, P., Gallinari, P.: Datum-wise classification: a sequential approach to sparsity. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011, Part I. LNCS, vol. 6911, pp. 375–390. Springer, Heidelberg (2011)
8. Gao, T., Koller, D.: Active classification based on value of classifier. In: NIPS, vol. 24, pp. 1062–1070 (2011)
9. He, H., Daume III, H., Eisner, J.: Imitation learning by coaching. In: Advances In Neural Information Processing Systems (NIPS), pp. 3158–3166 (2012)
10. Jiang, J., Teichert, A.R., Daumé III, H., Eisner, J.: Learned prioritization for trading off accuracy and speed. In: NIPS, pp. 1340–1348 (2012)
11. Karayev, S., Baumgartner, T., Fritz, M., Darrell, T.: Timely object recognition. In: NIPS, pp. 899–907 (2012)
12. Karayev, S., Fritz, M., Darrell, T.: Dynamic feature selection for classification on a budget. In: International Conference on Machine Learning (ICML): Workshop on Prediction with Sequential Models (2013)

13. Maaten, L., Welling, M., Saul, L.K.: Hidden-unit conditional random fields. In: International Conference on Artificial Intelligence and Statistics, pp. 479–488 (2011)
14. Megiddo, N.: On the complexity of polyhedral separability. Discrete & Computational Geometry 3(1) (1988)
15. Patterson, G., Hays, J.: Sun attribute database: Discovering, annotating, and recognizing scene attributes. In: Proceeding of the 25th Conference on Computer Vision and Pattern Recognition (CVPR) (2012)
16. Ross, S., Bagnell, D.: Efficient reductions for imitation learning. In: International Conference on Artificial Intelligence and Statistics (AISTATS), pp. 661–668 (2010)
17. Taskar, B., Guestrin, C., Koller, D.: Max-margin markov networks. In: NIPS (2003)
18. Trapeznikov, K., Saligrama, V.: Supervised sequential classification under budget constraints. In: International Conference on Artificial Intelligence and Statistics (AISTATS) (2013)
19. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y., Singer, Y.: Large margin methods for structured and interdependent output variables. Journal of Machine Learning Research 6(9) (2005)
20. Viola, P., Jones, M.: Robust Real-time Object Detection. International Journal of Computer Vision 4, 34–47 (2001)
21. Wang, J., Saligrama, V.: Local supervised learning through space partitioning. Advances in Neural Information Processing Systems 25 (2012)
22. Wang, J., Saligrama, V.: Locally-Linear Learning Machines (L3M). In: Asian Conference on Machine Learning, pp. 451–466 (2013)
23. Wang, J., Trapeznikov, K., Saligrama, V.: An LP for Sequential Learning Under Budgets. In: Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, pp. 987–995 (2014)
24. Weiss, D., Sapp, B., Taskar, B.: Dynamic structured model selection. In: ICCV (2013)
25. Xiao, J., Hays, J., Ehinger, K.A., Oliva, A., Torralba, A.: Sun database: Large-scale scene recognition from abbey to zoo. In: 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3485–3492. IEEE (2010)
26. Xu, Z., Kusner, M., Chen, M., Weinberger, K.Q.: Cost-sensitive tree of classifiers. In: Proceedings of the 30th International Conference on Machine Learning (ICML 2013), pp. 133–141 (2013)
27. Zhang, C., Zhang, Z.: A Survey of Recent Advances in Face Detection. Tech. rep., Microsoft Research (2010)