Online Kernel Selection: Algorithms and Evaluations

Tianbao Yang[1], **Mehrdad Mahdavi**[1], **Rong Jin**[1], **Jinfeng Yi**[1], **Steven C.H. Hoi**[2]
[1] Department of Computer Science and Engineering, Michigan State University, MI, 48824, USA
[2] School of Computer Engineering, Nanyang Technological University, 639798, Singapore
[1]{yangtia1, mahdavim, rongjin, yijinfen}@msu.edu, [2]chhoi@ntu.edu.sg

Abstract

Kernel methods have been successfully applied to many machine learning problems. Nevertheless, since the performance of kernel methods depends heavily on the type of kernels being used, identifying good kernels among a set of given kernels is important to the success of kernel methods. A straightforward approach to address this problem is cross-validation by training a separate classifier for each kernel and choosing the best kernel classifier out of them. Another approach is Multiple Kernel Learning (MKL), which aims to learn a single kernel classifier from an optimal combination of multiple kernels. However, both approaches suffer from a high computational cost in computing the full kernel matrices and in training, especially when the number of kernels or the number of training examples is very large. In this paper, we tackle this problem by proposing an efficient online kernel selection algorithm. It incrementally learns a weight for each kernel classifier. The weight for each kernel classifier can help us to select a good kernel among a set of given kernels. The proposed approach is efficient in that (i) it is an online approach and therefore avoids computing all the full kernel matrices before training; (ii) it only updates a single kernel classifier each time by a sampling technique and therefore saves time on updating kernel classifiers with poor performance; (iii) it has a theoretically guaranteed performance compared to the best kernel predictor. Empirical studies on image classification tasks demonstrate the effectiveness of the proposed approach for selecting a good kernel among a set of kernels.

Introduction

Kernel methods have attracted a significant amount of interest in machine learning, computer vision, and bioinformatics due to their superior empirical performance. For a specific task in hand, many kernels can be applied based on the special characteristics of certain objects. For example, in image classification, various types of features can be extracted (e.g. invariant SIFT features, bag-of-word features for images) and various types of kernels (e.g. chi-square, RBF, etc) can be used. As a result, the performance of kernel methods may depend on the kernel that is being used. It is empirically observed that in image classification, chi-square kernel is one of the most effective kernels (Zhang et al. 2007) and in text categorization, probability product kernel is a commonly used kernel which usually gives better performance than other kernels (Jebara, Kondor, and Howard 2004). Therefore, in the absence of prior experience, it is important to select an appropriate kernel for a specific task for the success of the kernel methods.

A straightforward approach for kernel selection is by cross-validation, i.e. training a separate kernel classifier for each individual kernel on a training data set and testing it on a separate validation data set, and then choosing the kernel classifier that gives the best performance on the validation data set. This approach wastes a lot of time on training, especially for those very bad kernels.

An approach to avoid training individual kernel classifier for each kernel is by MKL, which aims to efficiently learn a single kernel classifier from a combination of the multiple kernels, where the combination weights are learned by minimizing some empirical loss on a training data. However, MKL still suffers from a high computational cost in training, and current algorithms for MKL do not scale very well to a large number of kernels or a large number of training examples. In addition, both approaches need to compute all the full kernel matrices before training.

In this work, we present an efficient online approach for kernel selection. It incrementally learns a weight for each kernel, which measures the relative importance of each kernel classifier for prediction, and selects a good kernel based on the weights. The proposed approach is computationally efficient because it avoids computing all the full kernel matrices and updates a single kernel classifier each time by a sampling technique. The proposed approach also has a theoretic guarantee in performance for the selected kernel compared to the best kernel. We refer to the proposed online leaning approach for selecting a good kernel as **Online Kernel Selection (OKS)**.

Related Work

The present work is closely related to model selection problem. Model selection is the task of selecting a statistical model from a set of candidate models, given training data. Many criteria and approaches have been proposed for model selection. Examples include AIC (Akaike 1974), BIC (Schwarz 1978), and Mallows' Cp (Mallows 1973). Using these criteria for model selection, one needs to train a model for each function class, computes the value of the chosen criterion, and then selects the model that gives the best value. Another common practice for model selection is by cross-validation. The problem of these approaches is their high computational cost, i.e. one needs to learn a model completely for all function classes even though some classes may have very poor performance. Recently, Agarwal et al. (Agarwal et al. 2011) proposed an algorithm for model selection under a computational budget and proved its theoretical guarantee. However, it makes an explicit assumption on the hierarchical structure of the functional classes. In our setting, we are facing a set of Reproducing Kernel Hilbert Spaces (RKHS), which do not satisfy the hierarchical structure.

Our work is also related to kernel methods and kernel learning. Kernel methods have been used extensively in bioinformatics (Schölkopf, Guyon, and Weston 2000), machine learning (Schölkopf and Smola 2001), and computer vision (Lampert 2009). An interesting question is to find the best kernel among a set of kernels for a specific task. A common practice as we discussed before is by cross-validation. MKL is another approach to avoid training a separate model for each kernel. Many MKL algorithms have been proposed (Lanckriet et al. 2004; Rakotomamonjy et al. 2008; Xu et al. 2008; Cortes, Mohri, and Rostamizadeh 2010), to learn an optimal combination of kernels and the corresponding kernel classifier. However, these algorithms still suffer from high computational cost of computing the full kernel matrices and training as well.

The ideas used in the present paper draw on online learning. In online learning, one needs to update the predictor incrementally given the examples are coming sequentially. Staring from Perceptron algorithm (Rosenblatt 1958), many online learning algorithms have been proposed (Crammer et al. 2006; Freund and Schapire 1997; Auer et al. 2003). Recently, several online MKL algorithms have been proposed (Orabona, Luo, and Caputo 2010; Jin, Hoi, and Yang 2010). Our work differs from these works in that we are not aiming to optimize multiple kernel learning in an online fashion. Instead, we are interested in quickly finding a good kernel among a set of kernels for prediction. Finally, we note that the our online kernel selection algorithm for classification (i.e. Algorithm 2) is similar to the single stochastic update algorithm in (Jin, Hoi, and Yang 2010), however, our framework (i) updates the weight vector based on exponential weighted average algorithm, and (ii) uses an unbiased trick in updating the weight vector and the predictors, which together allow us to derive additive (rather than multiplicative) regret bound or mistake bound, and therefore to better compare with the performance of the best kernel.

We can summarize our contributions as follows: (i) we present a general online kernel selection algorithm for a prediction task with any convex loss function; (ii) we present an online kernel selection algorithm for classification with hinge loss; (iii) we present a theoretical analysis of the proposed algorithms for selecting a good kernel; (iv) we conduct empirical evaluations on the proposed online kernel selection algorithms on image classification.

Online Kernel Selection

In this section, we present several online learning algorithms for online kernel selection. We first present a general online algorithm for any convex loss function, state its regret bound, and analyze its performance in selecting the best kernel. Then we present an algorithm for classification by adapting the general algorithm to hinge loss.

Before jumping to detailed algorithms, we first introduce some notations that will be used throughout the paper. We let $(\mathbf{x}_t, y_t)$ denote the tth example with feature representation $\mathbf{x}_t \in \mathbb{R}^d$ and label y_t, and let $\mathcal{K}_m = \{\kappa_1, \cdots, \kappa_m\}$ denote the given m kernels. $\mathcal{H}_{\kappa_j}$ denotes the Reproducing Kernel Hilbert Space endowed by κ_j, $f_j^t \in \mathcal{H}_{\kappa_j}$ denotes the jth kernel predictor learned up to tth trial, $f_j^t(\mathbf{x}_t)$ denotes its evaluation on example $\mathbf{x}_t$, and $\mathbf{f}_t = (f_1^t, \cdots, f_m^t)$. We use $\mathbf{w}_t = (w_1^t, \cdots, w_m^t) \in \mathbb{R}_+^m$ to denote the weights for the m kernel predictors learned up to tth trial, $\mathbf{q}_t = \mathbf{w}_t / \sum_j w_j^t$ to denote the normalized vector, and $\mathbf{p}_t \in \mathbb{R}_+^m$ to denote a probability vector, i.e. $\sum_{j=1}^m p_j^t = 1$. Let $\ell(z, y)$ denote a convex loss function with respect to z, and $\nabla\ell(z, y)$ denote the (sub)gradient with respect to z. Finally, $[m]$ denotes the set $\{1, \cdots, m\}$, and $\mathsf{Multi}(\mathbf{p}_t)$ denotes multinomial distribution with parameters $\mathbf{p}_t$.

A General Algorithm and its Regret Bound

In this section, we present an online kernel selection algorithm for a general convex loss function, and state its regret bound. The basic steps of the proposed algorithm are shown in Algorithm 1. We explain the key steps of the algorithm in the following:

- **Step 1:** It takes as input a set of reproducing kernels $\mathcal{K}_m$, and parameters δ, η, λ. δ is a smoothing parameter as explained later. η, λ are step sizes for updating the weights $\mathbf{w}_t$ and the kernel predictors $\mathbf{f}_t$.
- **Step 5:** In trial t, after receiving one example $(\mathbf{x}_t, y_t)$, it samples one kernel out by following a multinomial distribution parameterized by $\mathbf{p}_t$.
- **Step 6:** It updates the weight of the sampled kernel by exponential weighted average algorithm, similar to prediction with expert advice (Cesa-Bianchi and Lugosi 2006). If the sampled kernel predictor suffers a large loss on the current example, its weight will be discounted by a large factor. The difference between Algorithm 1 and (Cesa-Bianchi and Lugosi 2006) is that in the exponential term, the loss is divided by the sampling probability corresponding to the sampled kernel. This trick is used to obtain an unbiased estimate of the loss vector, which has been used in many online learning algorithms, especially in the bandit settings (e.g. multi-armed bandit problems (Auer et al. 2003)), and is important for proving the regret bound.
- **Step 7:** It updates the predictor of the sampled kernel by gradient descent. Again, the gradient of the sampled predictor $f_{i_t}^t$ is divided by the sampling probability $p_{i_t}^t$ to obtain an unbiased gradient vector for $\mathbf{f}_t$.

Algorithm 1 Online Kernel Selection

```
1: INPUT: $\mathcal{K}_m$, $\delta \in (0,1)$, $\eta$, $\lambda$
2: Initialization: $\mathbf{f}_1 = \mathbf{0}, \mathbf{w}_1 = \mathbf{1}, \mathbf{p}_1 = \mathbf{1}/m$
3: for $t = 1, 2, \ldots, T$ do
4:     Receive an example $(\mathbf{x}_t, y_t)$
5:     Sample one kernel out $i_t \sim$Multi($\mathbf{p}_t$)
6:     Update $w_{i_t}^{t+1} = w_{i_t}^t \exp\left(-\eta \ell(f_{i_t}^t(\mathbf{x}_t), y_t)/p_{i_t}^t\right)$
7:     Update $f_{i_t}^{t+1} = f_{i_t}^t - \lambda \nabla \ell(f_{i_t}^t(\mathbf{x}_t), y_t)\kappa_{i_t}(\mathbf{x}_t, \cdot)/p_{i_t}^t$
8:     Update $\mathbf{q}_t = \mathbf{w}_t / \sum_j w_j^t$, $\mathbf{p}_t = (1-\delta)\mathbf{q}_t + \delta\mathbf{1}/m$
9: end for
10: OUTPUT: $\mathbf{q}_T$ or $\mathbf{w}_T$
```

- **Step 8:** It updates the sampling probabilities by combining the updated normalized weight vector $\mathbf{q}_t$ and the uniform probability $1/m$, with a smoothing parameter δ. On one hand, the kernel predictors with small weights, i.e. suffering large losses on the past data, will have less chance to be sampled for future updating. On the other hand, each kernel predictor has a chance (at least δ/m) to be updated. This effect is similar to the exploitation-exploration tradeoff in multi-armed bandit algorithms (Auer et al. 2003).

The following theorem shows the expected regret bound suffered by Algorithm 1. Its proof sketch is presented in the appendix.

Theorem 1. *Assuming the loss function is non-negative and $\ell(f_i^t(\mathbf{x}_t), y_t)$ is bounded by L for all kernel predictors over all trials, i.e. $\max_{t=1,\cdots,T} \ell(f_i^t(\mathbf{x}_t), y_t) \leq L$, its gradient is bounded by G, i.e. $\|\nabla \ell(z, y)\| \leq G$, we have the expected regret bound of Algorithm 1 given as follows:*

$$\mathrm{E}\left[\sum_{t=1}^{T}\sum_{i=1}^{m} q_i^t \ell(f_i^t(\mathbf{x}_t), y_t)\right] \leq \min_{i\in[m]} \min_{f \in \mathcal{H}_{\kappa_i}} \sum_{t=1}^{T} \ell(f(\mathbf{x}_t), y_t) + \frac{\|f\|_{\mathcal{H}_{\kappa_i}}^2}{2\lambda} + \frac{\lambda m T G^2}{2\delta} + \frac{\eta m T L^2}{2(1-\delta)} + \frac{\ln m}{\eta}$$

In particular, by assuming the optimal kernel predictor is bounded and setting $\eta, \lambda \propto T^{-1/2}$, we can obtain a regret bound in the order of $O(\sqrt{T})$, i.e.

$$\mathrm{E}\sum_{t=1}^{T}\sum_{i=1}^{m} q_i^t \ell(f_i^t(\mathbf{x}_t), y_t) \leq \min_{\substack{i\in[m] \\ f \in \mathcal{H}_{\kappa_i}}} \sum_{t=1}^{T} \ell(f(\mathbf{x}_t), y_t) + O(\sqrt{T})$$

Note that the regret is compared to not only the best kernel predictor $f \in \mathcal{H}_{\kappa_i}$, but also the best function classes among $\mathcal{H}_{\kappa_i}, i \in [m]$. And, the expected regret bound of Algorithm 1 is in the order of $O(\sqrt{T})$, the optimal regret bound of gradient-based online learning algorithms for general convex function (Abernethy et al. 2009).

Let us now discuss how the normalized weight vector $\mathbf{q}_T$ can help us to select a good kernel. We assume the examples $(\mathbf{x}_t, y_t)$ are i.i.d samples, and define the expected loss of a predictor f as

$$\overline{\ell}(f) = \mathrm{E}_{(\mathbf{x},y)}\left[\ell(f(\mathbf{x}), y)\right]$$

The following corollary shows that the kernel predictor f_k^T where $k \sim$ Multi($\mathbf{q}_T$) has a good performance guarantee.

Corrolary 2. *Under the condition in Theorem 1, and let $k \sim$ Multi($\mathbf{q}_T$), then with probability $1-\epsilon$, we have*

$$\overline{\ell}(f_k^T) \leq \min_{i\in[m]} \min_{f \in \mathcal{H}_{\kappa_i}} \overline{\ell}(f) + O\left(\frac{1}{\epsilon\sqrt{T}}\right)$$

Proof. By taking expectation over $(\mathbf{x}_t, y_t), t = 1, \cdots, T$ on the two sides in the last inequality in Theorem 1, we have

$$\mathrm{E}\left[\sum_{t=1}^{T}\sum_{i=1}^{m} q_i^t \overline{\ell}(f_i^t)\right] \leq \min_{i\in[m]} \min_{f \in \mathcal{H}_{\kappa_i}} T\overline{\ell}(f) + O(\sqrt{T})$$

Let τ be random index picked from $[T]$, and $k \sim \mathbf{q}_\tau$, we have

$$\mathrm{E}[\overline{\ell}(f_k^\tau)] \leq \min_{i\in[m]} \min_{f \in \mathcal{H}_{\kappa_i}} \overline{\ell}(f) + O\left(\frac{1}{\sqrt{T}}\right)$$

By Markov inequality, we have with probability $1-\epsilon$,

$$\overline{\ell}(f_k^\tau) \leq \min_{i\in[m]} \min_{f \in \mathcal{H}_{\kappa_i}} \overline{\ell}(f) + O\left(\frac{1}{\epsilon\sqrt{T}}\right)$$

By considering T as a random index from a large set $\{1, \cdots, \widehat{T}\}$, where $\widehat{T} > T$, similar to the analysis in (Shwartz, Singer, and Srebro 2007), we could also have

$$\overline{\ell}(f_k^T) \leq \min_{i\in[m]} \min_{f \in \mathcal{H}_{\kappa_i}} \overline{\ell}(f) + O\left(\frac{1}{\epsilon\sqrt{T}}\right)$$

where $k \sim$ Multi($\mathbf{q}_T$). □

We note that using that last vector $\mathbf{q}_T$ is more preferable than $\mathbf{q}_\tau$, where τ is a random index $\tau \in [T]$, because it has less variance by learning from more examples. Corrolary 2 provides us a way to select a good kernel. In a stochastic way, we can sample a kernel from Multi($\mathbf{q}_T$), or in a deterministic way we can select the kernel with the largest probability in $\mathbf{q}_T$.

Online Kernel Selection for Classification

In this section, we present an algorithm of online kernel selection for binary classification with a hinge loss function. The problem itself is interesting because in classification problem, we usually consider mistake bound rather than regret bound, and the hinge loss function allows us design more efficient algorithm than the one presented in previous section. It is also interesting to compare the mistake bound of the proposed algorithm to that of Perceptron algorithm that uses only one fixed kernel. The steps are presented in Algorithm 2. Different from Algorithm 1, Algorithm 2 first computes whether the sampled kernel classifier makes wrong prediction or not, indicated by $z_{i_t}^t$. If the answer is yes, i.e. $z_{i_t}^t = 1$, then it proceeds to update the weight and the classifier for the sampled kernel, along with the sampling probabilities; otherwise it proceeds to next example. In updating the weight and the classifier, the loss $\ell(f_{i_t}^t(\mathbf{x}_t), y_t)$ and the gradient $\nabla\ell(f_{i_t}^t(\mathbf{x}_t), y_t)$ in Algorithm 1 are replaced by the indicator variable z_{i_t}. It is therefore more efficient than Algorithm 1. The following theorem shows the mistake bound of Algorithm 2.

Algorithm 2 Online Kernel Selection for Classification

```
INPUT: $\mathcal{K}_m$, $\delta \in (0,1)$, $\eta$, $\lambda$
Initialization: $\mathbf{f}_1 = \mathbf{0}, \mathbf{w}_1 = \mathbf{1}, \mathbf{p}_1 = \mathbf{1}/m$
for $t = 1, 2, \ldots, T$ do
    Receive an example $(\mathbf{x}_t, y_t)$
    Sample one kernel out $i_t \sim$Multi($\mathbf{p}_t$)
    Set $z^t_{i_t} = I(y_t f^t_{i_t}(\mathbf{x}_t) \leq 0)$
    Update $w^{t+1}_{i_t} = w^t_{i_t} \exp(-\eta z^t_{i_t}/p^t_{i_t})$
    Update $f^{t+1}_{i_t} = f^t_{i_t} + \lambda y_t z^t_{i_t} \kappa_{i_t}(\mathbf{x}_t, \cdot)/p^t_{i_t}$
    Update $\mathbf{q}_t = \mathbf{w}_t / \sum_j w^t_j$, $\mathbf{p}_t = (1-\delta)\mathbf{q}_t + \delta \mathbf{1}/m$
end for
OUTPUT: $\mathbf{q}_T$ or $\mathbf{w}_T$
```

Theorem 3. *By running Algorithm 2 with T examples, we have the expected number of mistakes bounded as follows:*

$$\mathrm{E}\left[\sum_{t=1}^{T}\sum_{i=1}^{m} q_i^t z_i^t\right] \leq \min_{i\in[m]} \min_{f\in\mathcal{H}_{\kappa_i}} \sum_{t=1}^{T} \ell(f(\mathbf{x}_t), y_t) + \frac{\|f\|^2_{\mathcal{H}_{\kappa_i}}}{2\lambda} + \left(\frac{\lambda m T}{2\delta} + \frac{\eta m T}{2(1-\delta)} + \frac{\ln m}{\eta}\right)$$

where $\ell(\hat{y}, y) = \max(0, 1 - y\hat{y})$.

The proof of Theorem 3 is similar to that of Theorem 1. We omit the details due to the space limit. We next compare the mistake bound in Theorem 3 to the mistake bound of Perceptron using a single fixed kernel κ_i (Dekel, Shalev-Shwartz, and Singer 2005), which is given by

$$M_T \leq 2 \min_{f\in\mathcal{H}_{\kappa_i}} \sum_{t=1}^{T} \ell(f(\mathbf{x}_t), y_t) + R$$

where R is the bound on the norm of the optimal predictor $\|f\|^2_{\mathcal{H}_{\kappa_i}}$. We let $\lambda, \eta \propto T^{-1/2}$ in Algorithm 2, and we have the expected mistake bound in Theorem 3 given by

$$\mathrm{E}\left[M_T\right] \leq \min_{i\in[m]} \left(\min_{f\in\mathcal{H}_{\kappa_i}} \sum_{t=1}^{T} \ell(f(\mathbf{x}_t), y_t)\right) + O(\sqrt{T})$$

We can see that the constant term in the mistake bound of Perceptron is replaced by $O(\sqrt{T})$ in the mistake bound of Algorithm 2; however the tradeoff is that our mistake bound compares to the *best* function in the *best* function space among a set of function spaces, while Perceptron just compares to the best function in a *fixed* function space.

Empirical Studies

In this section, we present empirical studies to evaluate the proposed online kernel selection algorithms. We choose two image sets, i.e. Pascal07 and Corel5k for testingbed. We construct three binary classification tasks on each data set, i.e. animal vs. vehicle, animal vs. person, person vs. vehicle on Pascal07, and sky vs. people, water vs tree, sky vs. grass on Corel5k. These classes are the top classes (in terms of number of examples) in the two data sets, respectively. We use the INRIA features for both data sets, where 15 types of feature are extracted from each image (e.g. Gist desciptor, color histograms, etc). For more details about how to extract these features, please refer to (Guillaumin et al. 2009; Guillaumin, Verbeek, and Schmid 2010). We construct 4 kernels (i.e. linear, polynomial of order 2, chi-square, and Gaussian) for each type of feature, which results in 60 kernels in total. We use the default training/testing splitting in the data sets. For **baselines**, we compare to

- cross-validation approaches by running SVM and Perceptron for each kernel and select the one that gives the best performance on the validation data set, to which we refer as **MuSVM** and **MuTron**, respectively.
- a two-stage approach by first computing the alignment between each kernel matrix with the target kernel matrix based on the labels, and then training a kernel SVM using the selected kernel with the largest alignment, to which we refer as **AlSVM** (Cortes, Mohri, and Rostamizadeh 2010).
- a multiple kernel learning approach by Simple MKL (Rakotomamonjy et al. 2008), to which we refer as **SIPMKL**, a deterministic online multiple kernel learning algorithm (Algorithm 1) by Jin et al. (Jin, Hoi, and Yang 2010), to which we refer as **OM-1**, and an online multiple kernel learning algorithm by Jie et al. (Orabona, Luo, and Caputo 2010), to which we refer as **OM-2**.
- a stochastic online multiple kernel learning algorithm (Algorithm 3) by Jin et al. (Jin, Hoi, and Yang 2010), to which we refer as **SOM**. Comparing to SOM allows us to verify whether the proposed algorithm is better for online kernel selection.

About the implementation details:

- We use LibSVM[1] that is implemented in C language to train kernel SVM. We use the SimpMKL toolbox[2] for simple multiple kernel learning. We implement OKS, Perceptron, OM-1, OM-2, and SOM in Matlab.
- The parameters (e.g. the regularization parameter C in SVM, SIPMKL, the stepsize λ in OKS, Perceptron [3], and the discount factor β in OM-1 and SOM) are tuned in a range (e.g. $2^{[-10:1:10]}$ for C, λ, and $[0.1 : 0.04 : 0.9]$ for β) on a validation data set, which is a 10% random sample from the training data.
- We run OKS , SOM, and OM-2 with two epochs (i.e. two passes over the training data). The smoothing parameter δ of OKS is set to 0.5 in the first epoch and 0.2 in the second epoch. We run both MuTron and OM-1 with one epoch due to their high computational cost.
- The stepsize η in OKS is set to its optimal value (e.g. $\eta = \sqrt{2(1-\delta)\ln m/mT}$ in Theorem 3). The best kernel of OKS and SOM is selected deterministically, which gives the largest value in $\mathbf{q}_T$.

[1] `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

[2] `http://asi.insa-rouen.fr/enseignants/~arakotom`

[3] The original Perceptron used a fixed stepsize, we add a stepsize with tunning for fair comparision.

Table 1: Performance of Algorithms on Pascal07(h: hour(s))

		online			batch			online+batch	
task	Measure	MuTron	OM-1	OM-2	AlSVM	MuSVM	SIPMKL	SOM-SVM	OKS-SVM
animal vs vehicle (n=3434)	RT	7.2h	4.7h	2.5h	0.56h	1.3h	53h	0.03h+0.02h	0.013h+0.02h
	ACC	0.8502	0.8360	0.8279	0.8812	0.8818	0.8806	0.7603(±0.0732)	0.8675(±0.0155)
	opt-K	11	13	NA	11	15	34	1,6,14,20,48	10,10,11,11,13
animal vs person (n=2776)	RT	5.5h	3.6h	2.4h	0.45h	0.9h	36h	0.02h+0.02h	0.010h+0.02h
	ACC	0.8193	0.7783	0.8402	0.8285	0.8285	0.7435	0.7628(±0.0553)	0.8251(±0.0083)
	opt-K	15	11	NA	11	11	10	1,13,25,36,50	10,11,13,14,16
person vs vehicle (n=2584)	RT	4.1h	2.7h	1.4h	0.43h	0.8h	31h	0.015h+0.01h	0.008h+0.01h
	ACC	0.8773	0.7646	0.8562	0.9049	0.9064	0.8662	0.8678(±0.0332)	0.8954(±0.0098)
	opt-K	16	15	NA	11	15	14	9,9,9,29,51	9,11,13,16,33

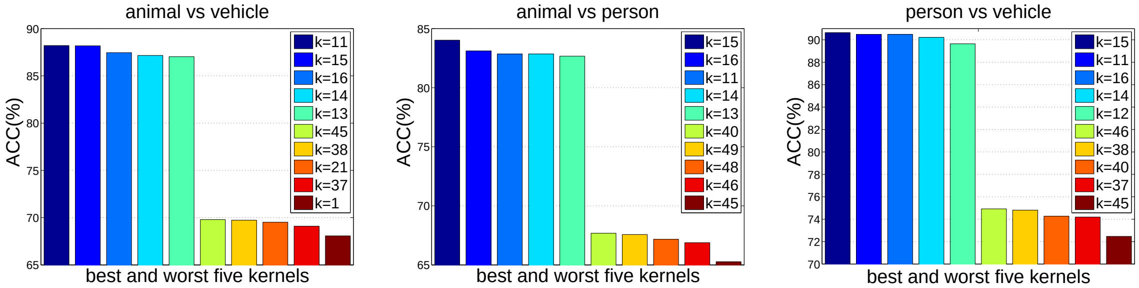

Figure 1: Accuracy of the best and the worst 5 kernels on the testing data of Pascal07 (Good kernels 9∼12 and 13∼16 are the 4 types of kernels defined on bag-of-words feature quantized from dense sift features at two layouts, respectively. Bad kernels 45∼48 are the 4 types of kernels defined on color histogram in LAB representation. Note that the best kernel 11 or 15 is chi-square kernel.)

We report the results on the two data sets of different algorithms in Table 1, and 2, respectively, where **SOM-SVM** and **OKS-SVM** refer to the approaches by running SOM and OKS first to select the best kernel and then running SVM with the selected kernel, respectively. We report three measures evaluated on different algorithms, i.e. the running time (RT)[4], the accuracy on testing data (ACC) of the returned kernel classifier[5], and the returned optimal kernel id (opt-K)[6]. Since SOM and OKS are stochastic algorithms, we therefore run both algorithms with 5 different random seedings, and report the selected best kernel in each trial and the averaged ACC over the 5 random trials. Note that the running time of SOM/OKS-SVM includes the running time of SOM/OKS for selecting one kernel plus the running time of LibSVM for training a kernel classifier using the selected kernel. We did not report the results using the corresponding kernel classifier for the selected kernel by OKS, since we are using OKS for selecting a good kernel. It is worth noting that by running OKS followed by Perceptron on the selected kernel we are able to obtain a good performance, which might be useful when training data is too huge to run batch kernel SVM. We also plot in Figure 1, 2 the accuracy of the best and the worst 5 kernel classifiers, which are trained by LibSVM and selected based on their performance on the testing data, which can be seen as the groundtruth to tell the performance of the best and the worst 5 kernels.

From these results, we can see that OKS can quickly identify a good kernel, and the performance of OKS-SVM is comparable to MuSVM and MKL. Compared to SOM, the proposed OKS can select better kernels, which verifies the important extensions we made to SOM for online kernel selection.

Conclusions

In this paper, we propose online kernel selection algorithms. The empirical studies on image classification demonstrate the effectiveness of the proposed algorithms. In the future, we plan to extend the work in two dimensions: in algorithmic dimension, we plan to derive algorithms that have high probability bounds in online setting, and in empirical dimension, we plan to evaluate the online kernel selection algorithms on more prediction tasks and data sets.

[4] The running time includes the cross validation time for tuning the parameters, and the preprocessing time for computing the kernel matrices for MuSVM, AlSVM.

[5] For MuTron, AlSVM, and MuSVM, the returned kernel classifier is the selected best kernel classifier, for OM-1, OM-2 and SIPMKL, the returned classifier is the combined kernel classifier.

[6] The optimal kernels for OM-1 and SIPMKL shown in the Tables are the ones that have the largest weight. OM-2 does not output any weight vector corresponding to kernels.

Table 2: Performance of Algorithms on Corel5k(s: second(s))

		online			batch			online+batch	
task	Measure	MuTron	OM-1	OM-2	AlSVM	MuSVM	SIPMKL	SOM-SVM	OKS-SVM
sky vs people (n=1471)	RT	1621s	997s	767s	924s	1187s	19493s	14s+18s	9s+18s
	ACC	0.7126	0.8563	0.8683	0.8563	0.8922	0.8503	0.7856(±0.0292)	0.8874(±0.0155)
	opt-K	10	16	NA	43	59	14	1,7,18,37,54	9,10,11,16,39
water vs tree (n=1572)	RT	2506s	1470s	915s	785s	1458s	25124s	17s+24s	12s+24s
	ACC	0.7351	0.7405	0.7676	0.7784	0.8108	0.8324	0.7589(±0.0293)	0.7892(±0.0220)
	opt-K	15	15	NA	11	16	14	6,12,41,44,52	9,11,13,15,16
sky vs grass (n=1187)	RT	817s	572s	447s	493s	725s	10808s	8s+12s	8s+12s
	ACC	0.8200	0.6933	0.8667	0.9067	0.8867	0.8800	0.8093(±0.0678)	0.8920(±0.0098)
	opt-K	13	11	NA	11	10	14	7,28,28,38,43	11,12,13,15,16

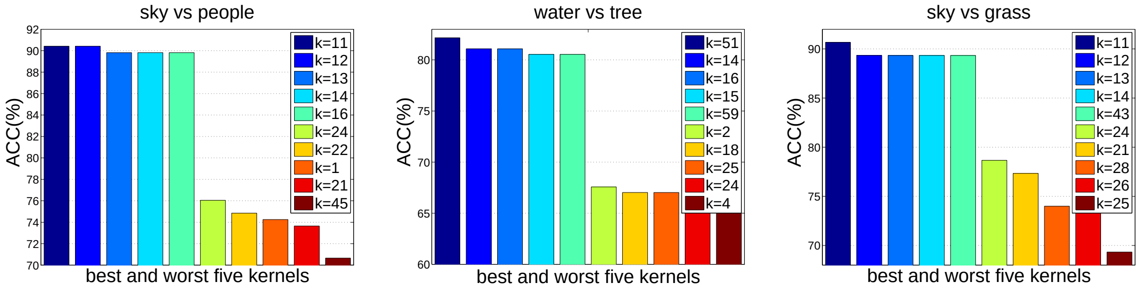

Figure 2: Accuracy of the best and the worst 5 kernels on the testing data of Corel5k (Bad kernels 21∼24 and 25∼28 are the 4 types of kernels defined on bag-of-words feature quantized from Hue descriptor extracted for Harris-Laplacian interest points at two layouts. Bad kernels 1∼4 are the 4 types of kernels defined on bag-of-words feature quantized from Hue descriptor extracted densely.)

Acknowledgments

This work was supported in part by National Science Foundation (IIS-0643494) and Office of Naval Research (ONR N00014-09-1-0663).

Appendix

Proof of Theorem 1

The proof is combining the proof of exponential weighted average algorithm and that of the gradient descent algorithm for online learning. Let us introduce the notations $m_i^t = I(i_t = i)$ and $\widetilde{m}_i^t = m_i^t/p_i^t$. It is easy to see that $\mathrm{E}_t[m_i^t] = p_i^t$ and $\mathrm{E}_t[\widetilde{m}_i^t] = 1$, where E_t is taken expectation on random variables i_t conditioned on all previous random variables. Following the standard analysis of exponential weighted average algorithm (e.g. (Auer et al. 2003)), we first bound $\sum_{t=1}^T \ln \frac{W_{t+1}}{W_t}$ from below and above. With $\widetilde{m}_i^t$, we can write the updating rule for the weights and the kernel predictors by $w_i^{t+1} = w_i^t e^{-\eta \widetilde{m}_i^t \ell(f_i(\mathbf{x}_t), y_t)}$, and $f_i^{t+1} = f_i^t - \lambda \nabla \ell(f_i^t(\mathbf{x}_t), y_t)\kappa_i(\mathbf{x}_t, \cdot)$. The summation is bounded from below by the updating rule of $\mathbf{w}_t$,

$$\sum_{t=1}^T \ln \frac{W_{t+1}}{W_t} = \ln \frac{W_{T+1}}{W_1} = \ln \frac{\sum_{i=1}^m w_i^{T+1}}{m}$$

$$\geq \ln w_i^{T+1} - \ln m = -\eta \sum_{t=1}^T \widetilde{m}_i^t \ell(f_i(\mathbf{x}_t), y_t) - \ln m$$

To bound the summation from above by using the inequality $e^{-x} \leq 1 - x + \frac{1}{2}x^2, \forall x \geq 0$ and $\ln(1+x) \leq x$,

$$\ln \frac{W_{t+1}}{W_t} = \ln \sum_{i=1}^m \frac{w_i^t \exp(-\eta \widetilde{m}_i^t \ell(f_i^t(\mathbf{x}_t), y_t))}{W_t}$$

$$\leq \ln \sum_{i=1}^m q_i^t \left(1 - \eta \widetilde{m}_i^t \ell(f_i^t(\mathbf{x}_t), y_t) + \frac{1}{2}\eta^2 (\widetilde{m}_i^t)^2 \ell^2(f_i^t(\mathbf{x}_t), y_t)\right)$$

$$\leq -\eta \sum_{i=1}^m q_i^t \widetilde{m}_i^t \ell(f_i^t(\mathbf{x}_t), y_t) + \frac{1}{2}\eta^2 \sum_{i=1}^m q_i^t (\widetilde{m}_i^t \ell(f_i^t(\mathbf{x}_t), y_t))^2$$

Then taking summation on both sides we have

$$\sum_{t=1}^{T} \ln \frac{W_{t+1}}{W_t} \leq -\eta \sum_{t=1}^{T} \sum_{i=1}^{m} q_i^t \widetilde{m}_i^t \ell(f_i^t(\mathbf{x}_t), y_t)$$

$$+ \frac{1}{2}\eta^2 \sum_{t=1}^{T} \sum_{i=1}^{m} q_i^t (\widetilde{m}_i^t \ell(f_i^t(\mathbf{x}_t), y_t))^2$$

Combing the lower bound and upper bound we have

$$\sum_{t=1}^{T} \sum_{i=1}^{m} q_i^t \widetilde{m}_i^t \ell(f_i^t(\mathbf{x}_t), y_t) \leq \sum_{t=1}^{T} \widetilde{m}_i^t \ell(f_i^t(\mathbf{x}_t), y_t)$$

$$+ \frac{1}{2}\eta \sum_{t=1}^{T} \sum_{i=1}^{m} q_i^t \frac{m_i^t}{(p_i^t)^2} L^2 + \frac{\ln m}{\eta}$$

where we use upper bound $|\ell(f_i^t(\mathbf{x}_t), y_t)| \leq L$ to bound the second order term $\ell^2(f_i^t(\mathbf{x}_t), y_t)$. Then we bound the first order term $\ell(f_i^t(\mathbf{x}_t), y_t)$ by $\ell(f(\mathbf{x}_t), y_t), \forall f \in \mathcal{H}_{\kappa_i}, i \in [m]$ using the standard analysis of the gradient descent algorithm (e.g. (Nesterov 2004)),

$$\begin{aligned}
&\widetilde{m}_i^t(\ell(f_i^t(\mathbf{x}_t), y_t) - \ell(f(\mathbf{x}_t), y_t)) \\
&\leq \langle f_i^t - f, \widetilde{m}_i^t \nabla \ell(f_i^t(\mathbf{x}_t), y_t) \kappa_i(\mathbf{x}_t, \cdot) \rangle \\
&\leq \frac{1}{2\lambda}\left(\|f_i^t - f\|^2 - \|f_i^{t+1} - f\|^2 + \frac{m_i^t \nabla \ell^2(f_i^t(\mathbf{x}_t), y_t)\lambda^2}{(p_i^t)^2} \right) \\
&\leq \frac{1}{2\lambda}\left(\|f_i^t - f\|^2 - \|f_i^{t+1} - f\|^2 + \frac{m_i^t G^2 \lambda^2}{(p_i^t)^2} \right)
\end{aligned}$$

By combining the above inequalities, taking expectation over randomness, and using simple algebra we can prove the theorem.

References

Abernethy, J.; Agarwal, A.; Bartlett, P. L.; and Rakhlin, A. 2009. A stochastic view of optimal regret through minimax duality. In *COLT*.

Agarwal, A.; Duchi, J. C.; Bartlett, P. L.; and Levrard, C. 2011. Oracle inequalities for computationally budgeted model selection. In *COLT*.

Akaike, H. 1974. A new look at the statistical model identification. *Automatic Control, IEEE Transactions on* 19(6):716–723.

Auer, P.; Cesa-Bianchi, N.; Freund, Y.; and Schapire, R. E. 2003. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.* 32:48–77.

Cesa-Bianchi, N., and Lugosi, G. 2006. *Prediction, Learning, and Games*. New York, NY, USA: Cambridge University Press.

Cortes, C.; Mohri, M.; and Rostamizadeh, A. 2010. Two-stage learning kernel algorithms. In *ICML*, 239–246.

Crammer, K.; Dekel, O.; Keshet, J.; Shalev-Shwartz, S.; and Singer, Y. 2006. Online passive-aggressive algorithms. *J. Mach. Learn. Res.* 7:551–585.

Dekel, O.; Shalev-Shwartz, S.; and Singer, Y. 2005. The forgetron: A kernel-based perceptron on a fixed budget. In *NIPS*.

Freund, Y., and Schapire, R. E. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.* 55:119–139.

Guillaumin, M.; Mensink, T.; Verbeek, J. J.; and Schmid, C. 2009. Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *ICCV*, 309–316.

Guillaumin, M.; Verbeek, J. J.; and Schmid, C. 2010. Multimodal semi-supervised learning for image classification. In *CVPR*, 902–909.

Jebara, T.; Kondor, R.; and Howard, A. 2004. Probability product kernels. *J. Mach. Learn. Res.* 819–844.

Jin, R.; Hoi, S. C. H.; and Yang, T. 2010. Online multiple kernel learning: Algorithms and mistake bounds. In *ALT*, 390–404.

Lampert, C. H. 2009. Kernel methods in computer vision. *Found. Trends. Comput. Graph. Vis.* 4:193–285.

Lanckriet, G.; Cristianini, N.; Bartlett, P.; and Ghaoui, L. E. 2004. Learning the kernel matrix with semidefinite programming. *JMLR* 5:27–72.

Mallows, C. L. 1973. Some comments on cp. *Technometrics* 15:661–675.

Nesterov, Y. 2004. *Introductory Lectures on Convex Optimization: A Basic Course (Applied Optimization)*. Springer Netherlands, 1 edition.

Orabona, F.; Luo, J.; and Caputo, B. 2010. Online-batch strongly convex multi kernel learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Rakotomamonjy, A.; Bach, F. R.; Canu, S.; and Grandvalet, Y. 2008. SimpleMKL. *JMLR* 9:2491–2521.

Rosenblatt, F. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review* 65(6):386–408.

Schölkopf, B., and Smola, A. J. 2001. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press.

Schölkopf, B.; Guyon, I.; and Weston, J. 2000. Statistical learning and kernel methods in bioinformatics. Technical report.

Schwarz, G. 1978. Estimating the dimension of a model. *The Annals of Statistics* 6(2):461–464.

Shwartz, S. S.; Singer, Y.; and Srebro, N. 2007. Pegasos: Primal estimated sub-GrAdient SOlver for SVM. In *Proceedings of the 24th international conference on Machine learning*, 807–814.

Xu, Z.; Jin, R.; King, I.; and Lyu, M. R. 2008. An extended level method for efficient multiple kernel learning. In *NIPS*, 1825–1832.

Zhang, J.; Marszalek, M.; Lazebnik, S.; and Schmid, C. 2007. Local features and kernels for classification of texture and object categories: A comprehensive study. *International Journal of Computer Vision* 213–238.