Learning Kernels: Formulation, Algorithms and Performance Guarantee

Xinyu Wu
Department of Aeronautics and Astronautics, MIT

Abstract

In this report we present a review of the kernel learning problem, whose goal is to learn the suitable kernel matrix to classify data hard to be linearly classified in the original space. We primarily focus on kernel learning in Support Vector Machine (SVM), while the methodology can be naturally extended to any problem related to kernel selection. We introduce the basic problem formulation of this problem, mainly associated with semi-definite programming, and discuss its variants. We then summarize the generalization bound in terms of Rademacher complexity. Finally, we do simulation on small training set, and observe that it does not perform as well as in prior works where datasets are large. We posit opinions on why this happens, including data scarcity, the selection of candidate kernels, and overfitting.

1 Kernel Learning

Kernel learning has developed for around two decades, which aims to obtain the optimal kernel or optimal combination of a set of candidate kernels that can best characterize data attributes, including classification [1] and clustering [2]. Its related research has incorporated both transduction and induction learning. In transduction learning, the unlabeled test data can also be used in training while in induction learning, this is not allowed. We primarily discuss the transduction learning on classification based on SVM in this report. Due to the limitation of pages, we put necessary background knowledge in the appendix.

1.1 Basic Problem Formulation

Let $S_{tr} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{n_{tr}}, y_{n_{tr}})\}$ are the labeled training data, and $S_{te} = \{(\mathbf{x}_{n_{tr}+1}, y_{n_{tr}+1}), \ldots, (\mathbf{x}_{n_{tr}+n_{te}}, y_{n_{tr}+n_{te}})\}$ are the unlabeled test data, where $\{y_{n_{tr}+i}\}_{i=1}^{n_{te}}$ are unknown in prior. In transduction learning, the kernel matrix K can be written as $K = \begin{bmatrix} K_{tr} & K_{tr,te} \\ K_{tr,te}^T & K_{te} \end{bmatrix}$, where K_{tr} represents the block related to training data, K_{te} is related to test data, and $K_{tr,te}$ is the mixed block. Note that K must be symmetric and positive semidefinite. We denote the set of all symmetric and positive semidefinite matrices as $\mathcal{Q}$, then we need to find K that belongs to some convex subset $\mathcal{K} \subseteq \mathcal{Q}$ such that the classification margin achieves the optimum. The most common $\mathcal{K}$ is the affine space spanned by some fixed candidate kernels: $\{K_i\}_{i=1}^m$, and we restrict the trace of matrix K to be fixed since scaling does not affect the optimizer, that is

$$\mathcal{K} = \left\{ K \mid K = \sum_{i=1}^{m} \mu_i K_i,\ K \succeq 0,\ \text{trace}(K) = c \right\},$$

MIT 9.520/6.860: Statistical Learning Theory and Applications (Fall 2019).

where $\{\mu_i\}_{i=1}^m$ are decision variables. A more restricted way is that $\mu_i \geq 0, \forall i$, which we will discuss later. Hence, under SVM with hard margin, the kernel learning problem can be formulated as

$$K^* = \arg\min_{K\subseteq\mathcal{K}} f(K) = \arg\min_{K\subseteq\mathcal{K}} \max_{\boldsymbol{\alpha}\in\mathbb{R}^n, \boldsymbol{\alpha}\geq 0, \boldsymbol{\alpha}^T\mathbf{y}_{tr}=0} -\frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n \alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j) + \sum_{i=1}^n \alpha_i. \tag{1}$$

The cases under L_1-norm and L_2-norm soft margin SVM will be discussed later in comparison.

1.2 SDP Transformation under Hard Margin

To solve (1), we observe that $f(K)$ is a convex function of K, therefore this problem can be transformed into an SDP based on the following theorem.

Theorem 1 *[3]Consider S_{tr} and S_{te}. Let $\mathbf{y}_{tr} = [y_1, \ldots, y_{n_{tr}}]^T$ denote the vector containing labels of training data and let $\mathbf{Y}_{tr} = diag(\mathbf{y}_{tr})$ denote the corresponding diagonal matrix, then the kernel K^* that minimizes* (1) *can be found by solving the following SDP,*

$$\begin{aligned} \min_{K,\lambda,t,\boldsymbol{\gamma}} \quad & t \\ \textit{s.t.} \quad & \begin{bmatrix} \mathbf{Y}_{tr}K_{tr}\mathbf{Y}_{tr} & \mathbf{1}+\boldsymbol{\gamma}+\lambda\mathbf{y}_{tr} \\ (\mathbf{1}+\boldsymbol{\gamma}+\lambda\mathbf{y}_{tr})^T & t \end{bmatrix} \succeq 0 \\ & \textit{trace}(K) = c,\ K\in\mathcal{K},\ \boldsymbol{\gamma}\geq 0. \end{aligned} \tag{2}$$

If we further restrict that K should be a nonnegative combination of $\{K_i\}_{i=1}^m$, then we denote $\mathcal{K}^+ = \{K\in\mathcal{K} | K = \sum_{i=1}^m \mu_i K_i, \mu_i \geq 0\}$ and replace $K\in\mathcal{K}$ by $K\in\mathcal{K}^+$ in (2). The restriction is non-trivial, as it can transform (1) into the following second-order cone programming (SOCP) problem, a special case of SDP but enjoys lower time complexity:

Theorem 2 *[3] If we replace $\mathcal{K}$ by $\mathcal{K}^+$ in* (1)*, then solving* (1) *is equivalent to solve the following problem and taking its dual solution.*

$$\begin{aligned} \max_{\boldsymbol{\alpha},t} \quad & \boldsymbol{\alpha}^T\mathbf{1} - \frac{1}{2}ct \\ \textit{s.t.} \quad & K = \sum_{i=1}^m \mu_i K_i,\ t \geq \frac{1}{\textit{trace}(K_i)}\boldsymbol{\alpha}^T\mathbf{Y}_{tr}K_{i,tr}\mathbf{Y}_{tr}\boldsymbol{\alpha}, i = 1,2,\ldots,m. \\ & \textit{trace}(K) = c,\ \boldsymbol{\alpha}\geq 0,\ \boldsymbol{\alpha}^T\mathbf{y}_{tr} = 0. \end{aligned} \tag{3}$$

In terms of complexity, (2) is an SDP that can be solved with worst-case complexity $O((m+n_{tr})^2 n^{2.5})$, while (3) is an SOCP yielding worst-case complexity $O(mn_{tr}^3)$. There is $O(\min\{m, n_{tr}\}^{0.5})$ gap between these two methods, showing that the positive restriction leads to higher efficiency. On the opposite, theoretically the restriction will induce suboptimal solution compared with non-restricted version, however, as we show in experiments, such case may not be as expected. Both can be solved by the general program SeDuMi [4].

1.3 Variants

In this part we discuss several variants, including SVM under soft margin, the induction learning version, and kernel alignment, an alternative way to think over this problem.

1.3.1 Soft margin version

Recall the SVM with soft margin ((7) in appendix) which overcomes the problem of no feasible solutions in the original hard margin version. Here we summarize the results when we take L_1-norm and L_2-norm soft margin SVM.

For L_1-norm, the dual version of the objective under soft margin SVM is similar to the hard margin version except the constraint becomes $\alpha_i \in [0, C], \forall i$, which changes the positive semidefinite

constraint into $\begin{bmatrix} \mathbf{Y}_{tr}K_{tr}\mathbf{Y}_{tr} & \mathbf{1}+\boldsymbol{\gamma}-\boldsymbol{\beta}+\lambda\mathbf{y}_{tr} \\ (\mathbf{1}+\boldsymbol{\gamma}-\boldsymbol{\beta}+\lambda\mathbf{y}_{tr})^T & t-2C\boldsymbol{\beta}^T\mathbf{1} \end{bmatrix} \succeq 0$with $\boldsymbol{\beta} \geq 0$. For L_2-norm, the dual version is similar to the hard margin version except the objective function

$$f(K_{tr}) = \max_{\boldsymbol{\alpha}\in\mathbb{R}^n, \boldsymbol{\alpha}\geq 0, \boldsymbol{\alpha}^T\mathbf{y}_{tr}=0} -\frac{1}{2}\boldsymbol{\alpha}^T\left(\mathbf{Y}_{tr}K_{tr}\mathbf{Y}_{tr}+\frac{1}{C}I_{n_{tr}}\right)\boldsymbol{\alpha}+\boldsymbol{\alpha}^T\mathbf{1},$$

which changes the positive semidefinite constraint into $\begin{bmatrix} \mathbf{Y}_{tr}K_{tr}\mathbf{Y}_{tr}+C^{-1}I_{n_{tr}} & \mathbf{1}+\boldsymbol{\gamma}+\lambda\mathbf{y}_{tr} \\ (\mathbf{1}+\boldsymbol{\gamma}+\lambda\mathbf{y}_{tr})^T & t \end{bmatrix} \succeq 0$. Both cases can be solved following the transformation into SDP or SOCP in Section 1.2. L_1-norm offers a hard bound on dual variables $\boldsymbol{\alpha}$ while L_2-norm tends to limit the value of the square value of $\boldsymbol{\alpha}$, intuitively the variance among different components in $\boldsymbol{\alpha}$.

1.3.2 Induction learning

For induction learning, unlabeled test data can not be used in the training process, therefore the main difference in this setting for kernel learning exists in (3). First, only K_{tr} remains in the formulation, under the constraint $K_{tr} = \sum_{i=1}^m \mu_i K_{i,tr}$. According to [3], to make the problem more elegant, normalization is conducted by each K_i: $K_i(j,k) \leftarrow K_i(j,k)/\sqrt{K_i(j,j)K_i(k,k)}$, which makes trace$(K_i) = 1$ for all i. Therefore the first inequality constraint in (3) can be written in

$$t \geq \frac{1}{n_{tr}+n_{te}}\boldsymbol{\alpha}^T\mathbf{Y}_{tr}K_{i,tr}\mathbf{Y}_{tr}\boldsymbol{\alpha}, \forall i = 1,\ldots,m,$$

under n_{tr} training samples and n_{te} test samples, with the objective and other constraints unchanged. Note that the normalization avoids the difficulty to evaluation the trace of K_{te} in this constraint in (3).

1.3.3 Kernel Alignment

Despite the formulation based on classification error, another track of idea arises from measuring the closeness between a kernel $K \in \mathcal{K}$ and the kernel induced directly by the labels of the training samples $\mathbf{y}_{tr}\mathbf{y}_{tr}^T$ [5]. If they are close, then it indicates that K can well classify these training samples. The alignment between kernel matrices K_1 and K_2 w.r.t the training samples is defined as

$$A(K_1,K_2) := \frac{\langle K_1,K_2\rangle_F}{\sqrt{\langle K_1,K_1\rangle_F\langle K_2,K_2\rangle_F}} = \frac{\text{trace}(K_1^TK_2)}{\text{trace}(K_1^TK_1)\text{trace}(K_2^TK_2)}.$$

To maximize the alignment, akin to the SDP formulation, when all $K \in \mathcal{K}$ are normalized by trace$(K) = 1$, it is equivalent to solve

$$\min_{X,K} \quad \langle K_{tr}, \mathbf{y}_{tr}\mathbf{y}_{tr}^T\rangle_F \qquad \text{s.t.} \qquad \begin{bmatrix} X & K^T \\ K & I_n \end{bmatrix} \succeq 0,\ \text{trace}(X) \leq 1,\ K \in \mathcal{K}. \tag{4}$$

1.4 Summary over the Development of Algorithms

The conventional way to solve (1) is based on wrapper methods, which comes from [6]. The core idea is to solve the minimax problem iteratively between solving SVM and selecting kernel parameters. Specifically, it involves different types of numerical methods including Semi-Infinite Linear Program (SILP) [7], reduced gradient [8], Newton's method [9], and mirror descent [10]. However the above algorithms involves a large amount of optimization problems to be solved iteratively. To enhance the efficiency, the idea based on SDP/SOCP was proposed by [3] serving as the foundation of the latter studies, whose core idea is reviewed in this work. Besides, Cortes et.al. [11] proposed to utilize project gradient while in [12] the SMO algorithms were applied to tackle this problem.

2 Generalization Bounds

In this section, we discuss the generalization bound of the learned kernel $K^* \in \mathcal{K}$ on the test data. Recall that Rademacher complexity is applied to upper bound the out-of-sample error. The core issue is to determine the Rademacher complexity of the admissible kernel set $\mathcal{K}$ w.r.t. the training samples,

denoted as $\hat{\mathcal{R}}_{S_{tr}}(\mathcal{K})$. Note that each kernel K is generated by a feature mapping ϕ_K, hence, based on [1], it is equivalent to consider the $\hat{\mathcal{R}}_{S_{tr}}(\mathcal{H}_{m,q})$, where

$$\mathcal{H}_{m,q} = \{h(\mathbf{x}) = \phi_K(\mathbf{x}), \mathbf{x} \in \mathcal{X} | K = \sum_{k=1}^{m} \mu_k K_k, \boldsymbol{\mu} \in \Delta_q\}, \text{ with } \Delta_q = \{\boldsymbol{\mu} | \boldsymbol{\mu} \geq 0, \sum_{i=1}^{m} \mu_i^q = 1\}.$$

Note that such formulation adds the constraint $\sum_{i=1}^{m} \mu_i^q = 1$, which eliminates the normalization constraint over trace$(K) = c$ in Section 1. In terms of $\mathcal{H}_{m,q}$, we have the following result:

Theorem 3 *[1] Denote* $\mathbf{u} = (trace(K_1), \ldots, trace(K_m))^T$*, then for any* S_{tr} *with size* n_{tr} *we have*

$$\hat{\mathcal{R}}_{S_{tr}}(\mathcal{H}_{m,1}) \leq \frac{1}{m}\sqrt{\frac{23}{22} r ||\mathbf{u}||_r}, \ \forall r \in \mathbb{N}, r \geq 1; \ \hat{\mathcal{R}}_{S_{tr}}(\mathcal{H}_{m,q}) \leq \frac{1}{m}\sqrt{\frac{23}{22} r ||\mathbf{u}||_r}, \ r \in \mathbb{N}, \ \frac{1}{q} + \frac{1}{r} = 1.$$

From this theorem, we can analyze that when the $K_k(x, x) \leq C$ is bounded for $\forall x \in \mathcal{X}, \forall k = 1, \ldots, K$, then $\hat{\mathcal{R}}_{S_{tr}}(\mathcal{H}_{m,1}) = O\left(\left(\frac{\log m}{n_{tr}}\right)^{1/2}\right)$ and $\hat{\mathcal{R}}_{S_{tr}}(\mathcal{H}_{m,q}) = O\left(\left(\frac{r m^{1/r}}{n_{tr}}\right)^{1/2}\right)$. This result shows the interesting conclusion that the upper bound of $\mathcal{H}_{m,1}$ depends slightly on m, which indicates that the number of kernels does not have a large impact on increasing the complexity, thus not affecting much on generalization bound. However, this is not the case when q becomes larger: As q increases, r decreases, while $rm^{1/r}$ will decrease in most cases (as the unique extreme point is at $r = \log m$), and when $q = 1$, $rm^{1/r}$ leads to infinity bound! This is not meaningful. Hence refining the bound of $\hat{\mathcal{R}}_{S_{tr}}(\mathcal{H}_{m,q})$ is an open direction.

Furthermore, we can obtain the following generalization bound:

Theorem 4 *[1] Denote* $L(h) = \mathbb{E}[\mathbf{1}_{yf(\mathbf{x})<0}]$ *and* $\hat{L}_\gamma(h) = \frac{1}{m}\sum_{i=1}^{m} \min(1, [1 - y_i h(\mathbf{x}_i)/\gamma]_+)$ *as the* γ*-empirical margin loss, when* $K_k(x, x) \leq C$*, we have with probability at least* $1 - \delta$*,*

$$L(h) \leq \hat{L}_\gamma(h) + 2\sqrt{\log m}\sqrt{\frac{23eC/\gamma^2}{22n_{tr}}} + 3\sqrt{\frac{\log(2/\delta)}{2n_{tr}}} \text{ when } q = 1.$$

$$L(h) \leq \hat{L}_\gamma(h) + 2m^{\frac{1}{2r}}\sqrt{\frac{23eC/\gamma^2}{22n_{tr}}} + 3\sqrt{\frac{\log(2/\delta)}{2n_{tr}}} \text{ when } q \geq 2.$$

We can observe that $q = 1$ yields better bound regarding the value of m. Moreover, when we increase γ, $\hat{L}_\gamma(h)$ is larger while the second term previously related to Rademacher complexity becomes smaller, forming a balance.

Besides, it also has been proved that the dependence of $\sqrt{\log m}$ cannot be further enhanced due to the lower bound over the out-of-sample error $L(h)$: $\Omega(\sqrt{\log m}/n_{tr})$ derived by VC dimension in [3]. Meanwhile this results can be extended to $q \geq 2$. However, I am still interested in this tightness result, as it seems that the upper bound of $\hat{\mathcal{R}}_{S_{tr}}(\mathcal{H}_{m,q})$ when $q \geq 2$ is inconsistent with the bound when $q = 1$, and the fact that when $q = 1$, $rm^{1/r}$ goes to infinity as mentioned before. I suppose that more efforts should be placed in the tightness study to tackle with this issue.

Some other bounds, including additive bound (rather than multiplicative bound discussed above) in terms of $\log m$ [13], and the bound in more general cases without $\boldsymbol{\mu} \geq 0$ [14, 15], are shown in recent literature. However, it is also mentioned that even in $q = 1$, such good theoretical guarantee does not perform well in practical experiments [1].

3 Simulation

In this section we do several experiments over kernel learning. The experiments involve: (i) comparison between kernel learning over $\mathcal{K}$ and $\mathcal{K}^+$; (ii) comparison between hard margin, L_1-norm soft margin and L_2-norm soft margin. In each case, we consider a set of 3 candidate kernels $\{K_i\}_{i=1}^3$, where $K_1(i, j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^2$ is polynomial, $K_2(i, j) = e^{-1/2||\mathbf{x}_i - \mathbf{x}_j||^2}$ is Gaussian, and $K_3(i, j) = \mathbf{x}_i^T \mathbf{x}_j$ is linear. We compare the performance under five settings: (i) purely K_1; (ii) purely K_2; (iii) purely K_3; (iv) affine combination $\sum_{i=1}^3 \mu_i K_i$; (v) nonnegative combination $\sum_{i=1}^3 \mu_i^+ K_i$ where $\mu_i^+ \geq 0$.

We consider the synthetic dataset where each sample has 2 dimensions, and the sampled value at each dimension follows the standard normal distribution independently, i.e., $X_i = (X_{i1}, X_{i2})^T$ where $X_{i1} \sim \mathcal{N}(0,1)$, $X_{i2} \sim \mathcal{N}(0,1)$, $X_{i1} \perp\!\!\!\perp X_{i2}$. We take 3 different labeling criteria: (i) For X_i, if its $X_{i1}^2 + X_{i2}^2 < 1$ (Euclidean length less than 1), then its label $y_i = 1$, otherwise $y_i = -1$; (ii) For X_i, if $|-X_{i1} + X_{i2}^2| < 1$, then $y_i = 1$, otherwise $y_i = -1$; (iii) For X_i, if $X_{i1} + \frac{1}{X_{i2}} < 2$, then $y_i = 1$, otherwise $y_i = -1$. These three settings represent different classification boundaries.

In each trial, I consider 100 training samples and test over 40 samples, and take the average considering different selections of initial points in optimization. The reason I did not test larger dataset is due to the severe complexity increase when training samples accumulate ($O(m + n_{tr})^2 n^{2.5}$), which indicates one weakness of SDP-based kernel learning. Besides, as in previous works [3, 6, 7], the involved datasets are large enough, in most cases more than 1000, hence I am curious to see the performance under small amount of data. We use MATLAB to run the experiments.

Results are shown in Table 1, 2, and 3. It is interesting that none of the 5 kernel selections can always perform better than others, which shows that under different labeling distributions, kernel selection really matters. Furthermore, in contrast to previous work, the combination of candidate kernels cannot beat the best single kernel in most cases. For example, under labeling technique 1, simply selecting K_1 keeps better than either $\sum_{i=1}^3 \mu_i K_i$ and $\sum_{i=1}^3 \mu_i^* K_i$. This, from my perspectives, may result from the lacking of training data. Although in our kernel learning we also try to maximize the margin as that in single kernel cases, the scarceness of training samples may not ensure that such maximization is the best. Therefore unlike in prior arts directly utilizing the kernel learning framework, the kernel learning can not achieve the good results with a small number of samples, where the candidate kernel selection becomes more important and we need to make multiple trials on different (combinations of) kernels to select the best one.

Table 1: Prediction Accuracy under Labeling Technique 1

	K_1	K_2	K_3	$\sum_{i=1}^3 \mu_i K_i$	$\sum_{i=1}^3 \mu_i^* K_i$
Hard Margin	100%	97.5%	52.5%	87.5%	92.5%
L_1 **Soft Margin**	92.5%	90.0%	37.5%	37.5%	87.5%
L_2 **Soft Margin**	92.5%	85.0%	37.5%	20.5%	42.5%

Table 2: Prediction Accuracy under Labeling Technique 2

	K_1	K_2	K_3	$\sum_{i=1}^3 \mu_i K_i$	$\sum_{i=1}^3 \mu_i^* K_i$
Hard Margin	12.5%	82.5%	32.5%	77.5%	92.5%
L_1 **Soft Margin**	92.5%	92.5%	35.0%	35.0%	92.5%
L_2 **Soft Margin**	80.0%	90.0%	35.0%	15.0%	65.0%

Table 3: Prediction Accuracy under Labeling Technique 3

	K_1	K_2	K_3	$\sum_{i=1}^3 \mu_i K_i$	$\sum_{i=1}^3 \mu_i^* K_i$
Hard Margin	30.0%	25.0%	67.5%	25%	22.5%
L_1 **Soft Margin**	30.0%	65.0%	77.5%	77.5%	75%
L_2 **Soft Margin**	30.0%	65.0%	77.5%	72.5%	77.5%

The reason why K_1 performs extremely well under labeling technique 1 may result from the similarity between the circle classification boundary and the quadratic K_1, which embeds a proper feature that maps the circle boundary to an approximately linear boundary in the feature space. The reason why K_2 performs well under labeling technique 2 may also come from the similarity, as the classification boundary is a parabola where both linear and quadratic polynomials are not so akin to it.

Besides, by comparing $\sum_{i=1}^3 \mu_i K_i$ and $\sum_{i=1}^3 \mu_i^+ K_i$, we can observe that in most cases, $\sum_{i=1}^3 \mu_i^+ K_i$ is better, which is somewhat counterintuitive as $\sum_{i=1}^3 \mu_i^+ K_i$ is under more constraints. The lacking of training data may explain this, however this phenomenon partly coincides the results in [3] with large datasets. From my perspectives, the freedom of μ_i to be selected in negative numbers may cause overfitting over the data as such freedom enhances model complexity. Moreover, we can observe that the choice of hard or soft margin also needs to be determined under different classification techniques.

In all, under small training datasets, the effect of kernel learning does not appear as good as in previous works when larger datasets were involved. The selection of optimal kernels and hard/soft margin depends on the distribution of data and labels. Under our settings, the constraint on nonnegative kernel weights may be conducive to the generalizability of classification.

4 Conclusion

In this paper, we make a concise summary over the formulation and development of the kernel learning problem. We also concentrate on the generalization bound of this problem, where the number of candidate kernels does not play an important role. Simulation results show that under more complex classification bound, the kernel learning solution tends to behave better relatively to a single kernel. However, the kernel learning may not perform as well as single candidates under a small number of samples. The low efficiency of SDP also inhibits us to consider large-scale datasets within limited time. For future directions, the refinement of the generalization bound, and how to better design the kernel learning under small number of sampled data promise to be proper topics.

References

[1] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. New generalization bounds for learning kernels. *arXiv preprint arXiv:0912.3309*, 2009.

[2] Mark Girolami. Mercer kernel-based clustering in feature space. *IEEE Transactions on Neural Networks*, 13(3):780–784, 2002.

[3] Gert RG Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine learning research*, 5(Jan):27–72, 2004.

[4] Jos F Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization methods and software*, 11(1-4):625–653, 1999.

[5] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Algorithms for learning kernels based on centered alignment. *Journal of Machine Learning Research*, 13(Mar):795–828, 2012.

[6] Jason Weston, Sayan Mukherjee, Olivier Chapelle, Massimiliano Pontil, Tomaso Poggio, and Vladimir Vapnik. Feature selection for svms. In *Advances in neural information processing systems*, pages 668–674, 2001.

[7] Sören Sonnenburg, Gunnar Rätsch, Christin Schäfer, and Bernhard Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7(Jul):1531–1565, 2006.

[8] Alain Rakotomamonjy, Francis R Bach, Stéphane Canu, and Yves Grandvalet. Simplemkl. *Journal of Machine Learning Research*, 9(Nov):2491–2521, 2008.

[9] Marius Kloft, Ulf Brefeld, Pavel Laskov, Klaus-Robert Müller, Alexander Zien, and Sören Sonnenburg. Efficient and accurate lp-norm multiple kernel learning. In *Advances in neural information processing systems*, pages 997–1005, 2009.

[10] Arash Afkanpour, András György, Csaba Szepesvári, and Michael Bowling. A randomized mirror descent algorithm for large scale multiple kernel learning. In *International Conference on Machine Learning*, pages 374–382, 2013.

[11] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Learning non-linear combinations of kernels. In *Advances in neural information processing systems*, pages 396–404, 2009.

[12] Zhaonan Sun, Nawanol Ampornpunt, Manik Varma, and Svn Vishwanathan. Multiple kernel learning and the smo algorithm. In *Advances in neural information processing systems*, pages 2361–2369, 2010.

[13] Zakria Hussain and John Shawe-Taylor. Improved loss bounds for multiple kernel learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 370–377, 2011.

[14] Nathan Srebro and Shai Ben-David. Learning bounds for support vector machines with learned kernels. In *International Conference on Computational Learning Theory*, pages 169–183. Springer, 2006.

[15] Yiming Ying and Colin Campbell. Generalization bounds for learning the kernel. 2009.

5 Appendix: Background Knowledge

5.1 Kernel Matrix

In SVM, when data are not linearly separable, we need to search a feature mapping to embed each input data into a feature space where they can be separated by a linear classifier with high accuracy. Kernel-based method is proposed to solve such problem, without the need to explicitly specify the feature mapping. The essence of kernel is the inner product of any two feature vectors, representing their relative position. Specifically, we use $\phi(\cdot)$ to denote the feature mapping and $K(\cdot,\cdot)$ to denote the kernel, hence for any input vector $\mathbf{x}_i$ and $\mathbf{x}_j$, $K(\mathbf{x}_i,\mathbf{x}_j) := \langle\phi(\mathbf{x}_i),\phi(\mathbf{x}_j)\rangle = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$.

Common kernels include: (i) Polynomial kernel: $K(\mathbf{x}_i,\mathbf{x}_j) = (c + \mathbf{x}_i^T\mathbf{x}_j)^d$, where $c > 0$ and d are adjustable. (ii) Gaussian kernel: $K(\mathbf{x}_i,\mathbf{x}_j) = \exp\left(-\frac{||\mathbf{x}_i-\mathbf{x}_j||^2}{2\sigma^2}\right)$, where $\sigma \neq 0$ is adjustable. (iii) Sigmoid kernel: $K(\mathbf{x}_i,\mathbf{x}_j) = \tanh(a(\mathbf{x}_i^T\mathbf{x}_j) + b)$ where $a, b \geq 0$ are adjustable. Each kernel implicitly specifies a feature mapping. For example for polynomial kernel $K(\mathbf{x}_i,\mathbf{x}_j) = (1 + \mathbf{x}_i(1)\mathbf{x}_j(1) + \mathbf{x}_i(2)\mathbf{x}_j(2))^2$, we can verify that when $\phi(\mathbf{x}) = [(\mathbf{x}(1))^2, (\mathbf{x}(2))^2, \sqrt{2}(\mathbf{x}(1)\mathbf{x}(2)), \sqrt{2}\mathbf{x}(1), \sqrt{2}\mathbf{x}(2), 1]^T$, then $K(\mathbf{x}_i,\mathbf{x}_j) = \phi(\mathbf{x}_i)\phi(\mathbf{x}_j)$.

Suppose we have n input vectors $\{\mathbf{x}_i\}_{i=1}^n$, then we can define the corresponding kernel matrix $K = \{K(\mathbf{x}_i,\mathbf{x}_j)\}_{i,j=1\ldots n} \in \mathbb{R}^{n\times n}$. According to [1], K is a symmetric and positive semidefinite matrix, i.e., $K \succeq 0$, and in fact, if a matrix $A \succeq 0$ and is symmetric, then A is a kernel matrix.

5.2 Support Vector Machine

Recall in linearly separable case, the set of all linear classifiers is $H = \{\mathbf{x} \to \text{sgn}(\mathbf{w}^T\mathbf{x} + b) : \mathbf{w} \in \mathbb{R}^N, b \in \mathbb{R}\}$ where the label for each input belongs to $\{\pm 1\}$. We can formulate the problem maximizing the margin as

$$\min_{w,b} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} \qquad \text{s.t.} \qquad y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \geq 0 \tag{5}$$

By Lagrangian function and duality, the above problem is equivalent to find α such that

$$f = \max_{\alpha\in\mathbb{R}^n_{\geq 0}, \alpha^T y=0} -\frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n \alpha_i\alpha_j y_i y_j(\mathbf{x}_i^T\mathbf{x}_j) + \sum_{i=1}^n \alpha_i. \tag{6}$$

In the duality form we can incorporate the kernel by replacing $\mathbf{x}_i^T\mathbf{x}_j$ with $K(\mathbf{x}_i,\mathbf{x}_j)$. By learning the optimal α^*, the linear classifier in the feature space satisfies $w^* = \sum_{i=1}^N \alpha_i^* y_i \mathbf{x}_i$ and $b^* = y_j - \sum_{i=1}^N \alpha_i^* y_i(\mathbf{x}_i^T\mathbf{x}_j)$. If the data in the original space cannot be linearly separated, then we apply the soft-margin version of SVM as follows.

$$\min_{\mathbf{w}\in\mathbb{R}^N, b\in\mathbb{R}} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C||\xi|| \qquad \text{s.t.} \qquad y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i \geq 0, \quad \xi_i \geq 0, \tag{7}$$

where $||\cdot||$ denotes any norm.

5.3 Semidefinite Programming

Semidefinite programming (SDP) focuses on the optimization of convex functions over the convex cone of symmetric positive semidefinite matrices. It has a linear objective, and constraints including a linear matrix inequality and linear equality constraints. Specifically,

$$\min_u \quad c^T u \qquad \text{s.t.} \qquad M_0^i + \sum_{j=1}^q u_j M_j^i \succeq 0,\ i = 1, 2, \ldots, L. \quad Au = b, \tag{8}$$

where M_j^i are given matrices. The advantage of SDP is that the interior-point algorithm can solve it efficiently and with good theoretical guarantee. The core technique in kernel learning is to formulate the problem into a SDP, with the powerful tool Schur Complement Lemma:

Lemma 1 (Schur Complement Lemma [3]) *Consider the partitioned symmetric matrix* $X = X^T = \begin{bmatrix} A & B; \\ B^T & C \end{bmatrix}$, *where* A *and* C *are square and symmetric. If* A *is invertible, then if* $A \succ 0$ *then* $X \succeq 0$ *if and only if* $S = C - B^T A^{-1} B \succeq 0$.