AI2611 Project Report

Jiude Wei 521030910418

June 1, 2023

1 Introduction

We manually constructed a support vector machine (SVM) in this project. First of all, we will discuss about the mathmatical principle of SVM and introduce how kernel works and how to solve a convex optimization problem with multi mutually constrained lagrange multipliers by SMO algorithm. We then evaluated our SVM and compared it with the SVM provided by *sklearn*. We also processed some ablation study about the parameters involved in our model and kernels in order to find a combination of parameters that bests our model performance. We attempted to build a multi kernel by linearly combining current kernels in our model and evaluated its performance.

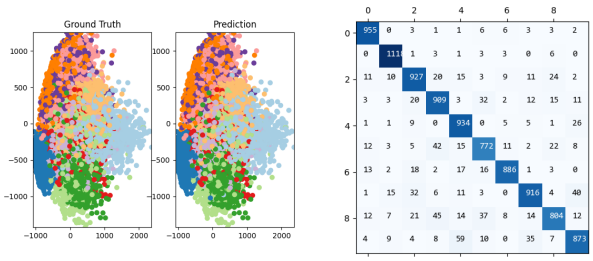

Figure 1: The image left shows the distribution of all instances. Each color refers to a certain class label. We use principle component analysis (PCA) to downsize the dimension of instances to 2 and then painted this image. The image right is the confusion matrix of ten labels. Both images depicts the evaluation of model with polynomial kernel, evaluated on MNIST.

2 SVM

2.1 Task

We are given several images of certain labels. We need to fit an support vector machine to classify the images according to their features.

Formally, we define x an n-dimensional vector that features an image and y its ground truth label where there are N labels totally. Given a training set $\mathcal{S} = \{(x, y) \mid x \in \mathbb{R}^n, y \in [N]\}$. We propose to fit a parameterized function $f_{w,b} : \mathbb{R}^n \to \mathcal{P}^N$ where $\mathcal{P}^N$ represents an n-dimensional probability distribution. The function takes an image as input and returns a probability distribution that represents its probability of being a certain label. The classification problem can be described as

$$\arg\min_{w,b} \sum_{(x,y)\in\mathcal{S}} \mathcal{L}(y, f_{w,b}(x))$$

where $\mathcal{L}$ is a loss function.

2.2 Principle

2.2.1 Binary SVM

In a binary classification task, we assume the label of the classes as $y = \pm 1$. We use parameters w and b to describe the classification margin

$$w^T x + b = 0$$

Assume our training set is linear seperable for ease. We define geometric margin from line to instances $(x^{(i)}, y^{(i)})$

$$\gamma = \min_{i=1,\cdots,M} \frac{1}{\|w\|}(w^T x^{(i)} + b)y^{(i)}$$

Our goal is to find a straight line that maximizes the minimum margin from line to instances with scaling $\|w\|$.

$$\begin{aligned} \max \quad & \gamma \\ s.t. \quad & (w^T x^{(i)} + b)y^{(i)} \geq \gamma \qquad i = 1, \cdots, M \\ & \|w\| = 1 \end{aligned}$$

We transform the optimization problem into a convex one by rescaling $\frac{\gamma}{\|w\|} = 1$

$$\begin{aligned} \min \quad & \frac{1}{2}\|w\|^2 \\ s.t. \quad & (w^T x^{(i)} + b)y^{(i)} \geq 1 \qquad i = 1, \cdots, M \end{aligned}$$

Now we consider a more complicated and common case where the training set is not linear seperable. We need to soften the margin constraint and hence we can derive an convex optimization problem

$$\begin{aligned} \min \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{M} \xi_i \\ s.t. \quad & (w^T x^{(i)} + b)y^{(i)} \geq 1 - \xi_i \qquad i = 1, \cdots, M \\ & \xi_i > 0 \qquad\qquad\qquad\qquad i = 1, \cdots, M \end{aligned}$$

Given the strong duality, we can solve its dual problem. By driving Lagrange on original problem, we can

derive the dual problem

$$\begin{aligned} \max \quad & \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_i \alpha_j y^{(i)} y^{(j)} \langle x^{(i)} x^{(j)} \rangle \\ s.t. \quad & \sum_{i=1}^{M} \alpha_i y^{(i)} = 0 \\ & \alpha \geq 0 \end{aligned}$$

Detailed discussion of binary SVM is shown in Appendix A.1.

2.2.2 Kernel

Before solving the dual problem, we would like to introduce kernel. In most cases we cannot linear seperate the intances in original space. We have to ascent dimension to create a hyperplane for classification.

We denote ϕ a mapping from original feature to dimensional ascented feature and replace the original features in the optimization problem. We define kernel to replace the inner products in optimization problem

$$\mathcal{K}(x^{(i)}, x^{(j)}) = \langle \phi(x^{(i)}), \phi(x^{(j)}) \rangle = \phi(x^{(i)})^T \phi(x^{(j)})$$

For instance, consider

$$\phi(x) = [x_1 x_1, \cdots, x_1 x_n, \cdots, x_n x_1, \cdots, x_n x_n]^T$$

A trivial way to compute the inner product based on kernel mapping is

$$\begin{aligned} \mathcal{K}(x^{(i)}, x^{(j)}) &= \phi(x^{(i)})^T \phi(x^{(j)}) \\ &= \sum_{k,l=1}^{n} (x_k^{(i)} x_l^{(i)})(x_k^{(j)} x_l^{(j)}) \\ &= \sum_{k=1}^{n} \sum_{l=1}^{n} x_k^{(i)} x_l^{(i)} x_k^{(j)} x_l^{(j)} \\ &= \left(\sum_{k=1}^{n} x_k^{(i)} x_k^{(j)} \right) \left(\sum_{l=1}^{n} x_l^{(i)} x_l^{(j)} \right) \\ &= (x^{(i)T} x^{(j)})^2 \end{aligned}$$

The second row in above statement describes a $O(n^2)$ computation method. We can ease the time complexity to $O(n)$ like the last row with certain derivation. Therefore, although kernel is time costly, we can ease the time complexity with some computational tricks.

We will discuss specific expressions and functions of some kernels in following experiments.

2.2.3 Regression: SMO

Consider the dual problem of a 2-classification SVM with kernel,

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_i \alpha_j y^{(i)} y^{(j)} \mathcal{K}\left(x^{(i)}, x^{(j)}\right) \\ s.t. \quad & \sum_{i=1}^{M} \alpha_i y^{(i)} = 0 \\ & \alpha \geq 0 \end{aligned}$$

The first constraint makes the components of parameter α not independent. Therefore we can not extract a single α_i and solve a QP only with respect to α_i.

The SMO algorithm decomposes the problem with M parameters into several problems with 2 parameters. We randomly choose a pair (α_i, α_j), fix other components as constants and optimize the problem only with respect to the pair for each epoch. In the following steps, we are going to optimize α_i.

Notice that our classification bound can be written by kernels and Lagrange multipliers

$$f(x) = w^T x + b = \sum_{i=1}^{M} \alpha_i y^{(i)} \mathcal{K}(x^{(i)}, x) + b$$

We use following notations to ease our expression

$$\begin{aligned} E_i &= f(x^{(i)}) - y_i \\ E_j &= f(x^{(j)}) - y_j \\ \eta &= \mathcal{K}(x^{(i)}, x^{(i)}) + \mathcal{K}(x^{(j)}, x^{(j)}) - 2\mathcal{K}(x^{(i)}, x^{(j)}) \end{aligned}$$

By setting the partial of the optimization function to partial α_i to zero, we can derive the updating rule on α_i

$$\alpha_i^{new} = \alpha_i^{old} + \frac{y^{(i)}(E_i - E_j)}{\eta}$$

Detailed derivation of SMO algorithm is shown in Appendix A.2.

2.2.4 Multi SVM: one-against-all

One-to-all method is a common idea to solve multi-classification task based on a well-defined binary SVM. We traverse each class and map the label of instances which belongs to this class to 1 and labels of others to -1. Then we can cast a binary SVM on instances with mapped labels.

$$\sigma_i(y^{(k)}) = \begin{cases} 1, & y^{(k)} = i \\ -1, & y^{(k)} \neq i \end{cases}, \quad i \in [n]$$

According to the derivation of binary SVM, an instance lies 'above' the classification boundary geometrically if its label is 1 and vice versa. Therefore an instance may belong to a certain class i if

$$w_i^T x + b_i > 0$$

However, an instance lies above multiple classification boundary and hence seems to belong to multiple classes. Remember our goal is to maximize the minimum margin between instances and classification boundary. We propose that an instance is more likely to belong to a class if its distance from the classification boundary of this class is greater. We rewrite our classifier in following term.

$$h(x) = \arg\max_{i \in [n]} w_i^T x + b_i$$

2.2.5 Multi SVM: one-against-one

Another common way to solve multi-classification task is one-to-one method. We traverse $\binom{n}{2}$ class pairs. For each pair (i, j), we only pick instances that belong to both labels and map their labels

$$\sigma_{ij}(y^{(k)}) = \begin{cases} 1, & y^{(k)} = i \\ -1, & y^{(k)} = j \end{cases}, \quad i, j \in [n]$$

Then we cast binary SVM on instances with mapped labels for $\binom{n}{2}$ times. In prediction process, each class pair will vote for an instance. A pair (i, j) votes an instance to belong to class i if

$$w_{ij}^T x + b_{ij} > 0$$

Otherwise it votes an instance to belong to class j. An instance is predicted to belong to a certain class if this class is voted most.

$$g_{ij}(x) = \begin{cases} 1, & w_{ij}^T x + b_{ij} \geq 0 \\ 0, & w_{ij}^T x + b_{ij} < 0 \end{cases}$$

$$h(x) = \arg\max_{i \in [n]} \sum_{j \in [n], j \neq i} g_{ij}(x)$$

To ease the time complexity of one-to-one classification, we can only cast binary SVM on $\binom{n}{2}/2$ class pairs. Each pair (i, j) satisfy $i < j$.

2.3 Implementation

In this project, we build our SVM in Python. Computation is mainly based on NumPy matrices.

We solve optimization problem with offline kernel values when the input size is not enormous so that huge amount of kernel computation in optimization process can be avoided and the time efficiency is increased. Our SVM also accept live update kernel computation to ease storage stress at the cost of time efficiency.

We regularize the value of input pixels into range $[0, 1]$ and hence we can evaluate our model's performance on various combination of hyperparameters and benchmark.

We use totally 5000 images to fit our model as well as sklearn SVM model in all evaluation below. For both MNIST and CIFAR-10 dataset, each label among the their ten labels respectively constitutes of 500 images.

Without further explanation, the multi SVM strategies in following experiments are one-against-all.

3 Experiment: Single kernel

3.1 Baseline

We evaluate our SVM model's basic performance in prediction accuracy and execution time and compare it with *sklearn* SVM model. The evaluation is seperately with three basic kernels: linear, rbf and polynomial.

Here we fit hyperparameters with $C = 1$, $\varepsilon = 10^{-3}$, $\gamma = 0.05$(for rbf), $c = 1$ and $d = 2$(for poly). The accuracy of our model with rbf kernel and polynomial kernel are respectively their best accuracy among various hyperparameters. The accuracy of sklearn SVM is the average accuracy evaluated with default hyperparameters.

		Ours		sklearn	
		Acc.	Time	Acc.	Time
MNIST	linear	0.834	310.6	**0.876**	0.7
	rbf	**0.898**	82.9	0.871	2.7
	poly	**0.909**	97.2	0.865	0.9
CIF'10	linear	0.231	718.0	**0.298**	2.2
	rbf	0.284	27.4	**0.351**	3.3
	poly	0.309	515.2	**0.325**	2.4

Table 1: Base evaluation of our model and comparation with sklearn SVM model. Linear stands for linear kernel. Rbf stands for Gaussian kernel. Poly stands for polynomial kernel.

We can observe from Table 1 that our model takes much longer time to fit itself than *sklearn* does, especially the execution time of model with linear kernel. But our model has better performance with rbf kernel and polynomial kernel. The accuracy of our model is able to match the one of *sklearn*

3.2 Ablation: regularization critic

In this section we focus on the parameter C in the optimization problem

$$\begin{aligned} \min \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{M}\xi_i \\ s.t. \quad & (w^T x^{(i)} + b)y^{(i)} \geq 1 - \xi_i \qquad i = 1, \cdots, M \\ & \xi_i > 0 \qquad\qquad\qquad\qquad i = 1, \cdots, M \end{aligned}$$

We evaluate the performance of our model with various value of C. The evaluation is with hyperparameter $\varepsilon = 10^{-3}$, maximum iteration $= 100$. The evaluation on rbf kernel is with $\gamma = 0.05$. The evaluate on polynomial kernel is with $c = 1$ and $d = 2$. We choose MNIST as dataset.

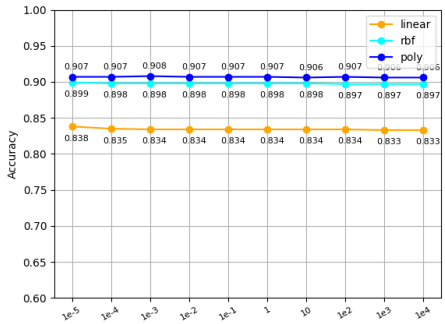

Figure 2: Ablation study on C. The x-axis stands for log scale value of C.

Among the three kernels, the accuracy drops slightly as C increases. But the drop is so tiny that we can neglect the effective of C in our model. A possible explanation that C shows no importance in our model is that we only consider C when we need to clip the value of α. In most cases, the value of α derived in SMO algorithm is in a range that clipping is not triggered since we cast regularization on input value. In conclusion, we can choose C with more freedom in our following evaluation and in most cases we do not need to worry about its influence on results.

3.3 Ablation: γ in rbf kernel

The expression of rbf kernel is

$$\mathcal{K}(x^{(i)}, x^{(j)}) = \exp\left(-\gamma \left\|x^{(i)} - x^{(j)}\right\|^2\right)$$

where γ is a positive real.

We reshape rbf kernel and cast Taylor's expansion on it

$$\begin{aligned} & \mathcal{K}(x^{(i)}, x^{(j)}) \\ &= \exp\left(-\gamma \left\|x^{(i)} - x^{(j)}\right\|^2\right) \\ &= \exp\left(-\gamma \sum_{k=1}^{n} x_k^{(i)} x_k^{(i)} - 2x_k^{(i)} x_k^{(j)} + x_k^{(j)} x_k^{(j)}\right) \\ &= \sum_{s=0}^{+\infty} \frac{1}{s!}\left(-\gamma \sum_{k=1}^{n} x_k^{(i)} x_k^{(i)} - 2x_k^{(i)} x_k^{(j)} + x_k^{(j)} x_k^{(j)}\right)^s \end{aligned}$$

Exponentially, rbf kernel is able to project an original feature with finite dimension into a infinity dimensional feature.

$$\phi_{\text{rbf}}(x) = \begin{bmatrix} 1 \\ -\gamma x_1 \\ \vdots \\ \sqrt{2\gamma} x_1 \\ \vdots \\ \sqrt{2}\gamma x_1 x_1 \\ \sqrt{2}\gamma x_1 x_2 \\ \vdots \\ 2\gamma x_1^3 \\ 2\gamma x_1^2 x_2 \\ \vdots \end{bmatrix}$$

We wonder how γ will influence our model with rbf kernel. This evaluation is with hyperparameter $\varepsilon = 10^{-3}$, maximum iteration $= 100$, $C = 1$.

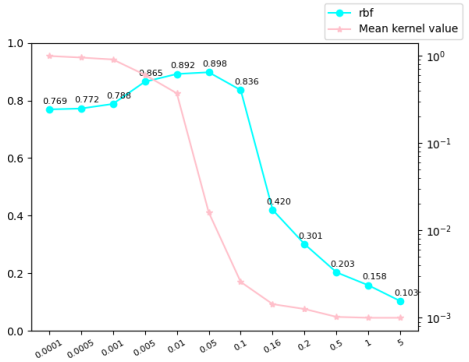

Figure 3: Ablation study on γ for rbf kernel. Cyan line stands for the accuracy of model with rbf kernel under various γ, whose y-axis is on the left. Pink line stands for the mean value of rbf kernels under various γ, whose y-axis is on the right.

The study shows that there exists a peak of rbf kernel model performance. Our model reaches its peak at about $\gamma = 0.05$. We also print the value of rbf kernels under various γ and expect to find its relationship with model performance. Notice that rbf kernel regularizes the features into $(0, 1)$ and it depicts the difference between two features. A reasonable explanation is that,

when γ is tiny, most kernels are close to 1 and hence our model cannot distinguish them effectively. When γ is large, most kernels are close to 0, which means even two images that belong to a same class are recognized as two images with huge differences. In such situation, our model cannnot construct a concept of a class or even extract the common feature of a class. Therefore its performance is poor.

3.4 Ablation: c, d in polynomial kernel

The expression of polynomial kernel is

$$\mathcal{K}(x^{(i)}, x^{(j)}) = \left(x^{(i)T}x^{(j)} + c\right)^d$$

Polynomial kernel is able to project an original feature into a higher dimension. For instance, let $d = 2$ here.

$$\begin{aligned}\mathcal{K}(x^{(i)}, x^{(j)}) &= \left(x^{(i)T}x^{(j)} + c\right)^2 \\ &= \sum_{s,t=1}^{n} (x_s^{(i)}x_t^{(i)})(x_s^{(j)}x_t^{(j)}) \\ &\quad + \sum_{s=1}^{n} (\sqrt{2c}x^{(i)})(\sqrt{2c}x^{(j)}) + c^2\end{aligned}$$

This eqivalents to a projection to a finite higher dimension

$$\phi_{\text{poly},d=2}(x) = \begin{bmatrix} c \\ x_1x_1 \\ x_1x_2 \\ \vdots \\ x_nx_n \\ \sqrt{2c}x_1 \\ \vdots \\ \sqrt{2c}x_n \end{bmatrix}$$

A greater d can project the original feature to a higher dimension, which occasionally results in a better classification performance.

We execute a discrete grid search on c and d to find the best match for our model with polynomial kernel.

We can observe from Table 2 and Table 3 that when d is not huge, c can only influence the performance slightly, given that the kernels is mainly dominant by d. As d becomes huge, a huge c can severely influence the value of kernels, resulting in the difference of model performance. A competitive combination is $d = 2$ or $d = 2.5$ with arbitrary c for our model.

d \ c	0.001	0.01	0.1	1	10
0.1	0.656	0.656	0.657	0.656	0.656
0.25	0.664	0.664	0.664	0.664	0.664
0.5	0.664	0.664	0.664	0.664	0.665
0.75	0.698	0.698	0.698	0.698	0.700
1	0.825	0.826	0.829	0.834	0.849
1.25	0.886	0.886	0.886	0.891	0.892
1.5	0.899	0.899	0.900	0.900	0.902
1.75	0.907	0.907	0.907	0.907	0.907
2	**0.909**	**0.909**	**0.909**	**0.909**	**0.909**
2.5	**0.909**	**0.909**	**0.909**	**0.909**	**0.909**
3	0.907	0.907	0.907	0.907	0.899
3.5	0.901	0.901	0.901	0.901	0.893
4	0.891	0.891	0.891	0.891	0.862

Table 2: Accuracy of model with polynomial kernel. Each row stands for a certain d. Each column stands for a certain c

d \ c	0.001	0.01	0.1	1	10
0.1	1.4	1.4	1.4	1.4	1.5
0.25	2.4	2.4	2.4	2.4	2.5
0.5	5.7	5.7	5.7	5.8	6.5
0.75	13.9	13.9	13.9	14.2	17.0
1	34.3	34.3	34.4	35.3	44.3
1.25	85.9	85.9	86.2	88.8	116.6
1.5	217.5	217.6	218.4	226.1	309.5
1.75	557.6	557.8	560.0	582.1	827.8
2	1445.3	1445.9	1452.1	1514.8	2231.2
2.5	1.0E4	1.0E4	1.0E4	1.1E4	1.7E4
3	7.2E4	7.2E4	7.3E4	7.7E4	1.3E5
3.5	5.4E4	5.4E4	5.5E4	5.8E4	1.0E6
4	4.2E6	4.2E6	4.3E6	4.5E6	8.1E6

Table 3: Mean value of polynomial kernels under various combination of c and d. Each row stands for a certain d. Each column stands for a certain c

4 Experiment: Multi kernel

4.1 Principle

We propose to construct our multi-kernel function by linearly overlaying the current kernel functions. Formally, suppose we have some kernel functions that maps feature to another dimension,

$$\Phi(x) = \begin{bmatrix} \phi_1(x) \\ \phi_2(x) \\ \vdots \\ \phi_t(x) \end{bmatrix}$$

We mutiply the feature vector by a vector p where $\|p\| = 1$. Hence the combined kernel can be expressed as

$$\hat{\Phi}(x) = p^T\Phi(x)$$

To simplify the computation, we also rewrite the kernel function.

$$\begin{aligned}&\hat{\Phi}(x^{(i)})^T\hat{\Phi}(x^{(j)})\\=&\sum_{k=1}^{t} p_k\Phi(x^{(i)})_k \cdot p_k\Phi(x^{(j)})_k\\=&\sum_{k=1}^{t} p_k^2\Phi(x^{(i)})_k\Phi(x^{(j)})_k\end{aligned}$$

We propose that the size of p is same as the number of basic kernel functions we have. We multiply the kernels in block for further simplification. We can reshape the multi kernel function as

$$\hat{\mathcal{K}}(x^{(i)}, x^{(j)}) = \sum_{k=1}^{t} p_k^t \mathcal{K}(x^{(i)}, x^{(j)})$$

4.2 Result

We evaluate the performance of our model with multi kernel based on various combination weight p. Our multi kernel involves three basic kernels: linear, rbf and polynomial kernel. We traverse the value of each component of p as well as the parameters corresponding to the kernels respectively to find the best kernel combination for our model. For each combination of kernel, we will list the best performance and the middle performance among all hyperparameters we have evaluated.

	Top Acc.	Mid Acc.
linear	0.834	0.834
linear + rbf	0.834	0.834
linear + poly	0.909	0.892
rbf + poly	0.909	0.909
linear + rbf + poly	**0.910**	0.908

Table 4: Evaluation of multi kernel based on various kernel weight and hyperparameters. Top Acc stands for the best performance among hyperparameters that we have searched. The evaluation is implmented on MNIST.

We can observe from Table 4 that the performance of model with any combination involving polynomial kernel is rather high. Furthermore, the accuracy of model with kernel combination involving polynomial kernel is similar to the accuracy of model with polynomial kernel itself. Meanwhile rbf kernel seems to be inaffective with the model performance. A reasonable explanation is that the values of polynomial kernel is enormous and hence even with weight about 10^{-3} the polynomial kernel still dominates the value of multi kernel. Similarly, the value of rbf kernel is tiny compared with linear kernel and polynomial kernel. Therefore the value of rbf kernel is always neglected in multi kernel.

We attempt to rescale the value of all kernels to $[0, 1]$ respectively and then evaluate the model with multi kernel which constitutes of the linear overlay of kernels. We fit $\gamma = 0.05$ for rbf kernel and $c = 1$, $d = 2$ for polynomial kernel here. $c = 1$

weight of rbf	Top Acc.	Mean Acc.
0	0.203	0.203
0.1	0.204	0.203
0.2	0.202	0.202
0.3	0.427	0.417
0.4	0.618	0.602
0.5	0.802	0.794
0.6	0.813	0.796
0.7	0.845	0.833
0.8	0.865	0.860
0.9	0.889	0.886

Table 5: Evaluation of multi kernel with rescaled kernel, based on various kernel weight. We mainly focus on the ratio of rbf. Top Acc and Mean Acc respectively stand for the best performance and average performance of model with certain weight of rbf kernel.

The experiments about polynomial kernel above have shown that rescaling the value of polynomial kernel will sigificantly destroy the structure in values of polynomial kernels. Therefore when rbf kernel is with less weight, the performance is poor where it is dominated by linear kernel and polynomial kernel. When the rbf kernel is with most weight, the performance reaches the performance of model with only rbf kernel but still cannot overtake it.

	Ours (multi kernel)		sklearn	
	Acc.	Time	Acc.	Time
MNIST	**0.910**	310.6	0.876	0.7
CIF'10	0.262	402.5	**0.351**	3.3

Table 6: Base evaluation of our model with multi kernel and comparation with sklearn SVM model with single kernel. *Sklearn* stands for the best performance among all single kernels evaluated with SVM provided by *sklearn*.

The best performance of our model with multi kernel costs much longer time than sklearn does and is similar to the result of model with single kernel. There is no sigificant evidence that the multi kernel strategy that we proposed can overtake single kernel.

5 Conclusion

We have constructed an SVM manually with kernel. We used SMO algorithm to solve the optimization problem in the SVM. With single kernel, our model can keep up with the performance provided by *sklearn* but its efficiency is much worse.

We executed ablation studies in order to search for the best hyperparameter for each kernel. Based on the

result of ablation studies, we confirmed the best hyperparameters responding to our model and dataset. We also discussed about the relationship between the value of kernels and the performance of model. Each kernel has its own explanation of differences between features and we should respect them when scaling the kernel value.

We then proposed a multi kernel which is the linear overlay of basic kernels. We searched for the best kernel weight for our model and compared it with the performance of SVM with single kernel provided by *sklearn*. The result shows that our proposal failed to further enhance the overall performance of SVM.

6 Acknowledgement

Here I would like to appreciate *Prof. Bingbing Ni* for providing us with a great amount of machine learning basic theories in this semester. I also appreciate *T.A. Qiaoqiao Jin* and *T.A. Yihan Li* for negotiating the requirements of project with us, providing computation power and answering our questions about this project and machine learning.

Appendix

A.1 Detailed derivation of binary SVM

We consider a linear classifier for a binary classification problem with features x and labels y. We can assume $y = \pm 1$ for further function. We use parameters w and b to describe our classifier

$$h(x) = \begin{cases} 1, & w^T x + b \geq 0 \\ -1, & w^T x + b < 0 \end{cases}$$

Given an instance $(x^{(i)}, y^{(i)})$, we define the functional margin of parameters of linear classifier w, b with respect to the instance

$$\hat{\gamma}_i = (w^T x^{(i)} + b) y^{(i)}$$

Here if $y = 1$, then the line should lie under the instance graphically in order that $\hat{\gamma}$ be a positive number. Conversely, the line should lie above the instance graphically. Given the training set S, we define the functional margin of w, b with respect to S as the minimal functional margin among the instances,

$$\hat{\gamma} = \min_{i=1,\cdots,M} \hat{\gamma}_i$$

Now suppose we have found appropriate parameters w, b. The classification boundary is straight line l : $w^T x + b = 0$. With instance $A : x^{(i)}$, we would like to find B where $AB \perp l$. We define $|AB| = \gamma_i$. B can be represented as

$$B : x^{(i)} - \gamma_i y^{(i)} \frac{w}{\|w\|}$$

Notice that B is exactly on classification boundary,

$$w^T \left(x^{(i)} - \gamma_i y^{(i)} \frac{w}{\|w\|} \right) + b = 0$$

Solving for γ_i indicates that

$$\gamma_i = \frac{1}{\|w\|} (w^T x^{(i)} + b) y^{(i)}$$

Here we derive γ_i the geometric margin of w, b with respect to instance $(x^{(i)}, y^{(i)})$. Notice $\gamma_i = \frac{\hat{\gamma}_i}{\|w\|}$, which means geometric margin equals to functional margin if $\|w\| = 1$. Similarly, we derive the geometric margin with respect to training set S

$$\gamma = \min_{i=1,\cdots,M} \gamma_i$$

Now we assume the training set S is linear seperable, which means $\gamma > 0$. Our goal is to find a classification boundary with maximum margin with respect to all instances.

$$\begin{aligned} \max \quad & \gamma \\ s.t. \quad & (w^T x^{(i)} + b) y^{(i)} \geq \gamma \qquad i = 1, \cdots, M \\ & \|w\| = 1 \end{aligned}$$

Given that γ and $\hat{\gamma}$ are synchronous while only one scalling factor $\|w\|$ differs them, we can rewrite the optimization problem as

$$\begin{aligned} \max \quad & \frac{\hat{\gamma}}{\|w\|} \\ s.t. \quad & (w^T x^{(i)} + b) y^{(i)} \geq \hat{\gamma} \qquad i = 1, \cdots, M \end{aligned}$$

By this transformation, we have got ridden of the non-convex constraint $\|w\| = 1$. Moreover, we rescale $\hat{\gamma} = 1$ by switching the scale of w and b.

$$\begin{aligned} \max \quad & \frac{1}{\|w\|} \\ s.t. \quad & (w^T x^{(i)} + b) y^{(i)} \geq 1 \qquad i = 1, \cdots, M \end{aligned}$$

However, optimization problem with form $\frac{1}{\|w\|}$ is difficult to solve. Therefore we transform the optimization object and optimal value of the derivation should keep the same.

$$\begin{aligned} \min \quad & \frac{1}{2}\|w\|^2 \\ s.t. \quad & (w^T x^{(i)} + b) y^{(i)} \geq 1 \qquad i = 1, \cdots, M \end{aligned}$$

Now we consider a more complicated and common task where the training set is not linear seperable. We ease the constraint that each instance $(x^{(i)}, y^{(i)})$ can live beyond the support vector with distance $\xi_i > 0$. Also, we want to minimize such cost.

$$\begin{aligned} \min \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{M} \xi_i \\ s.t. \quad & (w^T x^{(i)} + b) y^{(i)} \geq 1 - \xi_i \qquad i = 1, \cdots, M \\ & \xi_i > 0 \qquad\qquad\qquad\qquad i = 1, \cdots, M \end{aligned}$$

Here we introduce a critic factor C. We increase the cost to $C\xi_i$ to further constrain the instances beyond the margin for better classification performance as well as shrinking $\|w\|^2$.

Now we can solve the convex optimization problem by forming its Lagrange.

$$\mathcal{L}(w, b, \xi, \alpha, \beta) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{M} \xi_i - \sum_{i=1}^{M} \beta_i \xi_i - \sum_{i=1}^{M} \alpha_i \left[(w^T x^{(i)} + b) y^{(i)} - 1 + \xi_i \right]$$

Notice $\mathcal{L}$ is a scalar, then $\mathcal{L}^T = \mathcal{L}$, by which we reform $w^T x^{(i)} + b$ in the lagrange into $x^{(i)T} w + b$ for solving the optimal value. Let The partial difference on w be zero,

$$\frac{\partial \mathcal{L}}{\partial w} = w - \sum_{i=1}^{M} \alpha_i y^{(i)} x^{(i)} = 0$$

Obviously we get the optimal value for w

$$w^* = \sum_{i=1}^{M} \alpha_i y^{(i)} x^{(i)}$$

By solving partial difference on b we can not get the optimal value of b. We consider derive b^* geometrically. We assume that w^* have been found. The nature of our goal requests that line $w^T x + b = 0$ maximize the minimum of margins to the instances. Hence the line should be located right at the medium of two instances with minimum margin to the line that belong to different classes.

$$b^* = -\frac{1}{2}\left(\max_{y^{(i)}=1} w^{*T} x^{(i)} + \min_{y^{(i)}=-1} w^{*T} x^{(i)} \right)$$

Now we have derived the optimal value w^* and b^*. They are dependent on the Lagrange multipliers α. Given strong duality, we can solve its dual problem.

$$\begin{aligned} \max \quad & \sum_{i=1}^{M} \alpha_i - \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M} \alpha_i \alpha_j y^{(i)} y^{(j)} \langle x^{(i)} x^{(j)} \rangle \\ s.t. \quad & \sum_{i=1}^{M} \alpha_i y^{(i)} = 0 \\ & C = \alpha_i + \beta_i \qquad\qquad i = 1, \cdots, M \\ & \alpha_i \geq 0 \qquad\qquad i = 1, \cdots, M \\ & \beta_i \geq 0 \qquad\qquad i = 1, \cdots, M \end{aligned}$$

Given that β and C do not exist in the optimization object, we rewrite the dual problem more compactly.

$$\begin{aligned} \max \quad & \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_i \alpha_j y^{(i)} y^{(j)} \langle x^{(i)} x^{(j)} \rangle \\ s.t. \quad & \sum_{i=1}^{M} \alpha_i y^{(i)} = 0 \\ & \alpha \geq 0 \end{aligned}$$

The dual problem stated above is excatly the problem we need to optimize and discuss later.

A.2 SMO algorithm

John Platt proposed the SMO algorithm which provides us with an efficient method of solving an optimization problem with multiple Lagrange multipliers.

Consider the dual problem of binary SVM with kernel,

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_i \alpha_j y^{(i)} y^{(j)} \mathcal{K} \left(x^{(i)}, x^{(j)} \right) \\ s.t. \quad & \sum_{i=1}^{M} \alpha_i y^{(i)} = 0 \\ & \alpha \geq 0 \end{aligned}$$

The first constraint makes the components of parameter α not independent. Therefore we can not extract a single α_i and solve a QP only with respect to α_i.

The SMO algorithm decomposes the problem with M parameters into several problems with 2 parameters. We randomly choose a pair (α_i, α_j), fix other components as constants and optimize the problem only with respect to the pair for each iteration.

$$\begin{aligned} \max_{\alpha_i, \alpha_j} \quad & \alpha_i + \alpha_j - \frac{1}{2} \alpha_i^2 y^{(i)2} \mathcal{K}(x^{(i)}, x^{(i)}) - \frac{1}{2} \alpha_j^2 y^{(j)2} \mathcal{K}(x^{(j)}, x^{(j)}) - y^{(i)} y^{(j)} \alpha_i \alpha_j \mathcal{K}(x^{(i)}, x^{(j)}) \\ & - y^{(i)} \alpha_i \sum_{k \neq i,j} y^{(k)} \alpha_k \mathcal{K}(x^{(k)}, x^{(i)}) - y^{(j)} \alpha_j \sum_{k \neq i,j} y^{(k)} \alpha_k \mathcal{K}(x^{(k)}, x^{(j)}) + C \\ s.t. \quad & \alpha_i y^{(i)} + \alpha_j y^{(j)} = - \sum_{k \neq i,j} \alpha_k y^{(k)} \end{aligned}$$

We denote $\zeta = \sum_{k \neq i,j} \alpha_k y^{(k)}$ and $K_{ij} = \mathcal{K}(x^{(i)}, x^{(j)})$ for ease. We take the constraint into the optimization object and eliminate α_j. The optimization function that only related to α_i is

$$\begin{aligned} F(\alpha_i) = & \alpha_i - \alpha_i y^{(i)} y^{(j)} - y^{(j)} \zeta - \frac{1}{2} \alpha_i^2 y^{(i)2} K_{ii} - \frac{1}{2} K_{jj} \left(\alpha_i y^{(i)} + \zeta \right)^2 + \alpha_i y^{(i)} K_{ij} (\alpha_i y^{(i)} + \zeta) \\ & - \alpha_i y^{(i)} \sum_{k \neq i,j} \alpha_k y^{(k)} K_{ik} + (\alpha_i y(i) + \zeta) \sum_{k \neq i,j} \alpha_k y^{(k)} K_{jk} + C \end{aligned}$$

Let the partial difference to α_i be zero

$$\frac{\partial F}{\partial \alpha_i} = \alpha_i (K_{ii} + K_{jj} - 2K_{ij}) - \left(1 - y^{(i)} y^{(j)} + y^{(i)} \zeta (K_{ij} - K_{jj}) + y^{(i)} \sum_{k \neq i,j} \alpha_k y^{(k)} (K_{jk} - K_{ik}) \right) = 0$$

Notice the classification bound can be written as

$$f(x) = w^T x + b = \sum_{i=1}^{M} \alpha_i y^{(i)} \mathcal{K}(x^{(i)}, x) + b$$

We can rewrite that

$$\sum_{i\neq i,j} \alpha_k y^{(k)} K_{ik} = \sum_{i\neq i,j} \alpha_k y^{(k)} \mathcal{K}(x^{(i)}, x^{(k)}) = f(x^{(i)}) - \alpha_i y^{(i)} K_{ii} - \alpha_j y^{(j)} K_{ij} - b$$
$$\sum_{i\neq i,j} \alpha_k y^{(k)} K_{jk} = \sum_{i\neq i,j} \alpha_k y^{(k)} \mathcal{K}(x^{(j)}, x^{(k)}) = f(x^{(j)}) - \alpha_i y^{(i)} K_{ij} - \alpha_j y^{(j)} K_{jj} - b$$

Notice α_i and α_j in the equation above is not updated and treated as constant. To distinguish them with the α_i which we want to update by letting $\frac{\partial F}{\partial \alpha_i} = 0$, we write the equation in following status.

$$\frac{\partial F}{\partial \alpha_i} = \alpha_i^{new}(K_{ii} + K_{jj} - 2K_{ij}) - \alpha_i^{old}(K_{ii} + K_{jj} - 2K_{ij}) + y^{(i)} \left(\left(f(x^{(j)}) - y^{(j)} \right) - \left(f(x^{(i)}) - y^{(i)} \right) \right) = 0$$

We denote

$$E_i = f(x^{(i)}) - y_i$$
$$E_j = f(x^{(j)}) - y_j$$
$$\eta = K_{ii} + K_{jj} - 2K_{ij}$$

Therefore we derive the updating rule of SMO algorithm

$$\alpha_i^{new} = \alpha_i^{old} + \frac{y^{(i)}(E_i - E_j)}{\eta}$$