MAHINE LEARNING

HW2

YUN LI 6035800416

Problem1 Kernelized Perceptron

(a) When $y_i \neq \text{sign}(w^T \varphi(x_i))$. We update w by the following rules:

$$w = w + y_i \varphi(x_i)$$

Since $y_i \in \{1, -1\}$, w will always be the linear combination of $\varphi(x_i)$. We can write the conclusion in the following formula:

$$w = \sum_i \alpha_i \varphi(x_i)$$

Where $\alpha_i = \begin{cases} 0 & \textit{if } y_i = \text{sign}(w^T \varphi(x_i)) \\ y_i & \textit{otherwise} \end{cases}$ and $w_0 = 0$

(b) In this case, we make prediction by

$$\begin{aligned} \hat{y} &= \text{sign}(w^T \varphi(x)) \\ &= \text{sign}((\sum_j \alpha_j \varphi(x_j))^T \varphi(x)) \\ &= \text{sign}(\sum_j \alpha_j (\varphi(x_j)^T \varphi(x))) \\ &= \text{sign}(\sum_j \alpha_j K(x_j, x)) \end{aligned}$$

This prediction depends only on the inner products of feature vector x

(c) When $y_i \neq \text{sign}(w^T \varphi(x_i))$. We update w by the following rules:

$$\alpha_i = \alpha_i + y_i$$

At first, vector $\alpha_i = 0$ for length of n.

Problem2 Support Vector Machine without Bias Term

(a) primal form:

$$\begin{aligned} &\min_{w,\{\xi_i\}} C \sum_i \xi_i + \frac{1}{2} \|w\|_2^2 \\ &s.t.\ 1 - y_i[w^T \varphi(x_i)] \leq \xi_i, \forall i \\ &\xi_i \geq 0, \forall i \end{aligned}$$

(b) add lagrange multipliers β_i, α_i to each term :

$$L(\beta_i,\xi_i,\alpha_i,w)=C\sum_i \xi_i+\frac{1}{2}\|w\|_2^2-\sum_i \alpha_i(y_i[\mathrm{w}^T\varphi(\mathrm{x}_i)]+\xi_i-1)-\sum_i \beta_i\xi_i$$

(c) take derivatives

$$L(\beta_i,\xi_i,\alpha_i,w)=C\sum_i \xi_i+\frac{1}{2}\|w\|_2^2+\sum_i \alpha_i(y_i[\mathrm{w}^T\varphi(\mathrm{x}_i)]+\xi_i-1)+\sum_i \beta_i\xi_i$$

$$\frac{\partial L}{\partial w}=w-\sum_i \alpha_i y_i\varphi(\mathrm{x}_i)=0$$

$$\frac{\partial L}{\partial \xi_i}=C-\alpha_i-\beta_i=0, i=1,2....N$$

Thus we have:

$$w=\sum_i \alpha_i y_i\varphi(\mathrm{x}_i)$$

$$C=\alpha_i+\beta_i,\ i=1,2....N$$

(d) Substituting the above equations into Lagrangian.

$$\begin{aligned}L(\beta_i,\alpha_i)&=C\sum_i \xi_i+\frac{1}{2}\|w\|_2^2-\sum_i \alpha_i(y_i[\mathrm{w}^T\varphi(\mathrm{x}_i)]+\xi_i-1)-\sum_i \beta_i\xi_i\\&=\frac{1}{2}(\sum_i \alpha_i y_i\varphi(\mathrm{x}_i))^2-\sum_i \alpha_i y_i(\sum_i \alpha_i y_i\varphi(\mathrm{x}_i))^T\varphi(\mathrm{x}_i)+\sum_i \alpha_i\\&=\sum_i \alpha_i-\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j\varphi(\mathrm{x}_i)\varphi(\mathrm{x}_j)\end{aligned}$$

(e) Dual form:

$$\max_\alpha \sum_i \alpha_i-\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j\varphi(\mathrm{x}_i)^T\varphi(\mathrm{x}_j)$$
$$s.t.\ \ 0\le\alpha_i\le C$$

(f) The main difference is the condition constrains. Here, we only have $s.t.\ \ 0\le\alpha_i\le C$, while in the lecture, it is $\begin{array}{l}s.t.\ \ 0\le\alpha_i\le C\\ \sum_i a_i y_i=0\end{array}$

(g) compute the second order derivative:

$$L=\sum_i \alpha_i-\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j\varphi(\mathrm{x}_i)^T\varphi(\mathrm{x}_j)$$

$$\frac{dL}{d\alpha^2}=-\frac{1}{2}\sum_{i,j}y_i y_j\varphi(\mathrm{x}_i)^T\varphi(\mathrm{x}_j)=-\frac{1}{2}\sum_{i,j}y_i y_j\,\mathrm{k}(\mathrm{x}_i,\mathrm{x}_j)$$

Because $\sum_{i,j} y_i y_j \mathrm{k}(\mathrm{x}_i, \mathrm{x}_j)$ is a positive definite matrix (kernel function), we have $-\frac{1}{2}\sum_{i,j} y_i y_j \mathrm{k}(\mathrm{x}_i, \mathrm{x}_j) \le 0$. Thus the dual form is a concave function.

In conclusion, we need to maximize the dual function.

Problem 3 Sample Questions

3.1

The object function is:

$$L(\mathrm{D}) = \mathrm{p}(\mathrm{D};\theta) + \frac{1}{2}\|\theta\|_2^2$$

We need to maximize the object function

3.2

(a) Posterior probabilities are:

$$p(\mathrm{y}=1\,|\,\mathrm{x}) = \frac{e^{w_1^T x}}{e^{w_0^T x} + e^{w_1^T x}}$$
$$p(\mathrm{y}=0\,|\,\mathrm{x}) = \frac{e^{w_0^T x}}{e^{w_0^T x} + e^{w_1^T x}}$$

(b)stay unchanged:

$$p(\mathrm{y}=1\,|\,\mathrm{x}) = \frac{e^{w_1^T x}}{e^{w_0^T x} + e^{w_1^T x}} = \frac{1}{e^{(w_0^T - w_1^T)x} + 1} = \sigma((w_1^T - w_0^T)\mathrm{x})$$
$$p(\mathrm{y}=0\,|\,\mathrm{x}) = \frac{e^{w_0^T x}}{e^{w_0^T x} + e^{w_1^T x}} = \frac{1}{e^{(w_1^T - w_0^T)x} + 1} = \sigma((w_0^T - w_1^T)\mathrm{x})$$

If we add a constant vector b to both weight vectors, the above equations will stay unchanged because $(w_0^T - w_1^T)x$ is constant.

(c) suppose:

$$y_{ik} = \begin{cases} 1 & \mathit{if}\ y_i = k \\ 0 & \mathit{otherwise} \end{cases}$$

$$logP(D) = \sum_n logP(\mathrm{y}_n\,|\,\mathrm{x}_n) = \sum_n \log \prod_i p(\mathrm{c}_k\,|\,\mathrm{x}_i)^{y_{ik}} = \sum_n \sum_i \log y_{ik} p(\mathrm{c}_k\,|\,\mathrm{x}_i)$$

After Adding the regularization, we get

$$L(\mathrm{w}_0,\mathrm{w}_1)=\sum_i\sum_k \log y_{ik}p(\mathrm{c}_k\,|\,\mathrm{x}_i)+\frac{1}{2}\lambda_1\|w_0\|_2^2+\frac{1}{2}\lambda_2\|w_1\|_2^2$$
$$=\sum_i(\log y_{i0}p(\mathrm{c}_0\,|\,\mathrm{x}_i)+\log y_{i1}p(\mathrm{c}_1\,|\,\mathrm{x}_i))+\frac{1}{2}\lambda_1\|w_0\|_2^2+\frac{1}{2}\lambda_2\|w_1\|_2^2$$
$$=\sum_i(y_{i0}\log\sigma((w_0^T-w_1^T)\mathrm{x}_i)+\log y_{i1}\sigma((w_1^T-w_0^T)\mathrm{x}_i)+\frac{1}{2}\lambda_1\|w_0\|_2^2+\frac{1}{2}\lambda_2\|w_1\|_2^2$$
$$\textit{where } \lambda_1>0 \textit{ and } \lambda_2>0$$

Let $D=\sum_i(y_{i0}\log\sigma((w_0^T-w_1^T)\mathrm{x}_i)+\log y_{i1}\sigma((w_1^T-w_0^T)\mathrm{x}_i)$

Then
$$\frac{\partial D}{\partial w_1^{\,2}}=y_{i0}\sigma(-w_1^{\,T}\,\mathrm{x})(1-\sigma(-w_1^{\,T}\,\mathrm{x}))+y_{i1}\sigma(w_1^{\,T}\,\mathrm{x})(1-\sigma(w_1^{\,T}\,\mathrm{x}))$$
$$\frac{\partial D}{\partial w_0^{\,2}}=y_{i0}\sigma(w_0^{\,T}\,\mathrm{x})(1-\sigma(w_0^{\,T}\,\mathrm{x}))+y_{i1}\sigma(-w_0^{\,T}\,\mathrm{x})(1-\sigma(-w_0^{\,T}\,\mathrm{x}))$$

Both of them are convex (Hessian Metrix).

L is a strictly convex function, because the first term is convex, the second term and third term is strictly convex. Thus it has a unique solution.

Problem4 Programming

4.3 Implement linear SVM

(a) Report the cross-valiation accuracy and average training time on different C

result\C	4^{-6}	4^{-5}	4^{-4}	4^{-3}	4^{-2}	4^{-1}	4^{0}	4^{1}	4^{2}
Average accuracy	0.5300	0.7860	0.8010	0.8000	0.7950	0.7870	0.7880	0.7900	0.7910
Average Training time	0.9781	0.6813	0.7250	0.8625	1.2250	1.1281	1.1875	1.1469	1.1969

When C increases, the averaged accuracy first increases, achieving the maximum, then decreases, and then increases again; the averaged time first decreases, achieving the minimum, then increases, then decreases and increases again. But when optimal averaged accuracy achieved, the averaged training time is also near optimal.
To explain this, C controls the trade-off between errors of the SVM on training data and margin maximization. For large values of C, the optimizer will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. It may overfit and the complexity of the model increases. That is why accuracy increases and training time also increases. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points. It may underfit and the

complexity of this model is smaller. In this case, the accuracy and training time also decreases.

(b) C=4^-4

(c) 0.8543

4.4 Use linear SVM in LIBSVM

result\C	4^{-6}	4^{-5}	4^{-4}	4^{-3}	4^{-2}	4^{-1}	4^{0}	4^{1}	4^{2}
averaged accuracy	0.5170	0.7830	0.8040	0.8030	0.8030	0.8030	0.8030	0.7980	0.7960
Averaged Training time	0.3593	0.3281	0.2500	0.2500	0.2343	0.3281	0.8593	2.8437	14.5468

(a) The accuracy is different from mine

(b) Generally, it is much faster.

4.5 Use kernel SVM in LIBSVM

(a) Polynomial Kernel

Averaged accuracy:

degree\C	4^{-4}	4^{-3}	4^{-2}	4^{-1}	4^{0}	4^{1}	4^{2}	4^{3}	4^{4}	4^{5}	4^{6}	4^{7}
1	0.5170	0.5170	0.7820	0.8010	0.8050	0.8000	0.8020	0.8030	0.7980	0.7960	0.7970	0.7960
2	0.5170	0.5170	0.5170	0.6530	0.7340	0.7130	0.7090	0.7090	0.7090	0.7090	0.7090	0.7090
3	0.5170	0.5170	0.5170	0.6180	0.7920	0.8010	0.7990	0.7990	0.7990	0.7990	0.7990	0.7990

Training time(cputime, win8, ie7, 64):

degree\C	4^{-4}	4^{-3}	4^{-2}	4^{-1}	4^{0}	4^{1}	4^{2}	4^{3}	4^{4}	4^{5}	4^{6}	4^{7}
1	0.3594	0.3438	0.3281	0.2500	0.2188	0.2500	0.3750	1.0469	2.6250	13.3750	43.5938	72.7344
2	0.3750	0.3750	0.3750	0.3750	0.3594	0.4688	0.4531	0.4219	0.4375	0.4375	0.4375	0.4375
3	0.3750	0.3750	0.3750	0.3594	0.3906	0.4531	0.4063	0.4063	0.4531	0.4063	0.3906	0.4219

(b)RBF kernel

Averaged accuracy

C\gamma	4^{-7}	4^{-6}	4^{-5}	4^{-4}	4^{-3}	4^{-2}	4^{-1}
4^{-4}	0.5170	0.5170	0.5170	0.5170	0.5170	0.5170	0.5170
4^{-3}	0.5170	0.5170	0.5170	0.5170	0.5170	0.5170	0.5170
4^{-2}	0.5170	0.5170	0.5170	0.5400	0.6080	0.5170	0.5170
4^{-1}	0.5170	0.5170	0.7400	0.8020	0.8390	0.5170	0.5170
4^{0}	0.5170	0.7600	0.7950	0.8230	0.8580	0.7620	0.5700
4^{1}	0.7680	0.7900	0.8110	0.8470	0.8800	0.7780	0.5740
4^{2}	0.7900	0.8050	0.8240	0.8790	0.8770	0.7780	0.5740
4^{3}	0.8000	0.8090	0.8420	0.8770	0.8770	0.7780	0.5740
4^{4}	0.8030	0.8110	0.8790	0.8770	0.8770	0.7780	0.5740
4^{5}	0.8100	0.8400	0.8750	0.8770	0.8770	0.7780	0.5740
4^{6}	0.8070	0.8750	0.8750	0.8770	0.8770	0.7780	0.5740
4^{7}	0.8410	0.8740	0.8750	0.8770	0.8770	0.7780	0.5740

Training time (cputime, win8, ie7, 64):

C\gamma	4^{-7}	4^{-6}	4^{-5}	4^{-4}	4^{-3}	4^{-2}	4^{-1}
4^{-4}	0.3906	0.3906	0.4063	0.3906	0.4063	0.4063	0.4219
4^{-3}	0.3906	0.3906	0.4219	0.3906	0.4063	0.4063	0.4375
4^{-2}	0.3906	0.3906	0.4219	0.3906	0.4219	0.4219	0.4219
4^{-1}	0.3906	0.3906	0.4063	0.3438	0.3594	0.4063	0.4531
4^{0}	0.3906	0.4219	0.3281	0.2813	0.3125	0.4219	0.4375
4^{1}	0.4063	0.3281	0.2969	0.2500	0.3906	0.4688	0.5000
4^{2}	0.3281	0.2656	0.2500	0.2969	0.4063	0.4844	0.4688
4^{3}	0.2656	0.2344	0.2813	0.4063	0.3906	0.4375	0.4375
4^{4}	0.2500	0.2656	0.5000	0.4063	0.4531	0.4844	0.4531
4^{5}	0.2813	0.4688	0.6094	0.4063	0.3906	0.4531	0.4531
4^{6}	0.4844	1.1406	0.5938	0.4063	0.3906	0.4375	0.4531
4^{7}	1.1250	1.5469	0.5938	0.4219	0.3906	0.4375	0.4531

I will choose the RBF kernel, C=4 and gamma= 4^{-3}
Test accuracy is 90.4368%