CSCI 567–Homework 3

Chengliang Dong (Meng Liu, Kan Wang, Ye Feng and Peehoo Dewan) Wednesday 15th October, 2014

1 Kernelized perceptron

1.1 Linear combination

We will use induction to show this.

Base case: When $k = 0$, we have, for weight vector w_0,

$$\begin{aligned} w_0 &:= 0 \\ &= \sum_{i=1}^{m} \alpha_i \phi(x_i) \end{aligned}$$

where $\alpha_i = 0, \forall i$

Inductive step: If there exists n s.t. for $k = n > 0$, we have,

$$w_n = \sum_{i=1}^{m} \alpha_i \phi(x_i)$$

Then, for $k = n + 1$, i.e. $(n + 1)^{th}$ iteration, we have, for weight vector w_{n+1},

$$\begin{aligned} w_{n+1} &= w_n + sign(w^T \phi(x_i))\phi(x_j) \\ &= \sum_{i=1}^{m} \alpha_i \phi(x_i) \pm \phi(x_j) \\ &= \sum_{i=1, i \neq j}^{m} \alpha_i \phi(x_i) + (\alpha_j \pm 1)\phi(x_j) \\ &= \sum_{i=1}^{m} \alpha'_i \phi(x_i) \end{aligned}$$

Therefore, $\forall k \geq 0$, we have,

$$w_k = \sum_{i=1}^{m} \alpha_i \phi(x_i)$$

1.2 Inner product

Prediction can be written as,

$$\begin{aligned} y_i &= sign(w^T \phi(x_i)) \\ &= sign((\sum_{j=1}^{m} \alpha_j \phi(x_j))^T \phi(x_i)) \\ &= sign(\sum_{j=1}^{m} \alpha_j \phi(x_j)^T \phi(x_i)) \\ &= sign(\sum_{j=1}^{m} \alpha_j < \phi(x_j), \phi(x_i) >) \end{aligned}$$

1.3 Algorithm

From above two questions, we propose algorithm as follows.
Initialization: initialized all α_i to 0. Iteration: For each training instance i, we calculate,

$$\hat{y}_i = sign(\sum_{j=1}^{m} \alpha_i < \phi(x_j), \phi(x_i) >)$$

If $\hat{y}_i > y_i$, then update α_i by incrementing it by 1.

$$\alpha_i' = \alpha_i + 1$$

If $\hat{y}_i < y_i$, then update α_i by decrementing it by 1.

$$\alpha_i' = \alpha_i - 1$$

Repeat above process till all $\hat{y}_i = y_i$ or number of iterations reached certain limit.

2 SVM without bias term

2.1 Introduce slack variable

For i^{th} instance, we have,

$$\xi_i = max(0, 1 - y^{(i)}(w^T\phi(x)^{(i)}))$$

2.2 Lagrangian primal form

$$L(w, \xi_i, \alpha, \beta) = \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{m}\xi_i + \sum_{i=1}^{m}\alpha_i(1 - y^{(i)}(w^T\phi(x)^{(i)}) - \xi_i) + \sum_{i=1}^{m}\beta_i(-\xi_i)$$

2.3 Link to dual form, primal variable

$$\begin{aligned}
\frac{\partial L(w, \xi_i, \alpha, \beta)}{w_j} &= w_j - \sum_{i=1}^{m}\alpha_i y^{(i)}\phi(x)_j^{(i)} := 0, \forall i \\
\frac{\partial L(w, \xi_i, \alpha, \beta)}{\xi_i} &= C - (\alpha_i + \beta_i) := 0, \forall i
\end{aligned} \tag{1}$$

2.4 Link to dual form, dual variable

$$\begin{aligned}
g(\alpha, \beta) &= \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{m}\xi_i + \sum_{i=1}^{m}\alpha_i(1 - y^{(i)}(w^T\phi(x)^{(i)}) - \xi_i) - \sum_{i=1}^{m}\beta_i\xi_i \\
&= \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}(\alpha_i + \beta_i)\xi_i + \sum_{i=1}^{m}\alpha_i(1 - y^{(i)}(w^T\phi(x)^{(i)})) \\
&= \frac{1}{2}\|w\|_2^2 + \sum_{i=1}^{m}\alpha_i - \|w\|_2^2 \\
&= \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\|w\|_2^2 \\
&= \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{k=1}^{n}\sum_{i=1}^{m}\alpha_i y^{(i)}\phi(x)_k^{(i)}\sum_{j=1}^{m}\alpha_j y^{(j)}\phi(x)_k^{(j)}
\end{aligned}$$

2.5 Dual form

Therefore, we can write dual form as,

$$max_\alpha \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{k=1}^{n}\sum_{i=1}^{m} \alpha_i y^{(i)} \phi(x)_k^{(i)} \sum_{j=1}^{m} \alpha_j y^{(j)} \phi(x)_k^{(j)}$$
$$\text{s.t. } 0 < \alpha_i <= C$$

2.6 Difference with/without bias term

Due to the absence of bias term, current dual form lacks $\sum_{i=1}^{m} \alpha_i y_i = 0$ in the constraint term.

2.7 Show dual form is a convex optimization problem (concave maximization/convex minimization)

Since the dual form is the summation of a linear function $\sum_{i=1}^{m} \alpha_i$, which is obviously convex and concave functions and a negative L_2 norm $\frac{1}{2}\sum_{k=1}^{n}\sum_{i=1}^{m} \alpha_i y^{(i)} \phi(x)_k^{(i)} \sum_{j=1}^{m} \alpha_j y^{(j)} \phi(x)_k^{(j)} = \frac{1}{2}\|w\|_2^2$, which is also a strictly concave function (if not all x and y are 0 proved in homework 1 and 2 many times). By the properties of function, sum of concave function with a strictly concave function is still strictly concave function, we have the dual problem as a convex optimization problem, with unique solution.

3 Sample questions

3.1 Regularization

First, define weight vector and feature vector (for each instance) as follows,

$$\theta = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{pmatrix} \in \mathbb{R}^{n+1}$$
$$x = \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{R}^{n+1}$$

Second, we minimize objective function as follows,

$$min_w - L + \lambda\|\theta\|_2^2$$
$$=min_w logP(D) + \lambda\|\theta\|_2^2$$

Where $\lambda > 0$.

3.2 Logistic regression

3.2.1 Posterior probability

First, define weight vector and feature vector (for each instance) as follows,

$$(w)_0 = \begin{pmatrix} w_0^(0) \\ w_1^(0) \\ \vdots \\ w_n^(0) \end{pmatrix} \in \mathbb{R}^{n+1}$$

$$(w)_1 = \begin{pmatrix} w_0^(1) \\ w_1^(1) \\ \vdots \\ w_n^(1) \end{pmatrix} \in \mathbb{R}^{n+1}$$

$$x = \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{R}^{n+1}$$

Second, using softmax function, we have,

$$P(y=0|x) = \frac{exp(\mathbf{w}_0^T x)}{exp(\mathbf{w}_0^T x) + exp(\mathbf{w}_1^T x)}$$

$$P(y=1|x) = \frac{exp(\mathbf{w}_1^T x)}{exp(\mathbf{w}_0^T x) + exp(\mathbf{w}_1^T x)}$$

3.2.2 Invariance by adding constant

$$\begin{aligned} P(y=0|x) &= \frac{exp((\mathbf{w}_0 + \mathbf{b})^T x)}{exp((\mathbf{w}_0 + \mathbf{b})^T x) + exp((\mathbf{w}_1 + \mathbf{b})^T x)} \\ &= \frac{exp(\mathbf{w}_0^T x) exp(\mathbf{b}^T x)}{exp(\mathbf{w}_0^T x) exp(\mathbf{b}^T x) + exp(\mathbf{w}_1^T x) exp(\mathbf{b}^T x)} \\ &= \frac{exp(\mathbf{w}_0^T x)}{exp(\mathbf{w}_0^T x) + exp(\mathbf{w}_1^T x)} \\ P(y=1|x) &= \frac{exp((\mathbf{w}_1 + \mathbf{b})^T x)}{exp((\mathbf{w}_0 + \mathbf{b})^T x) + exp((\mathbf{w}_1 + \mathbf{b})^T x)} \\ &= \frac{exp(\mathbf{w}_1^T x) exp(\mathbf{b}^T x)}{exp(\mathbf{w}_0^T x) exp(\mathbf{b}^T x) + exp(\mathbf{w}_1^T x) exp(\mathbf{b}^T x)} \\ &= \frac{exp(\mathbf{w}_1^T x)}{exp(\mathbf{w}_0^T x) + exp(\mathbf{w}_1^T x)} \end{aligned}$$

3.2.3 Regularization

We minimize objective function defined as follows, with respective to w_0, w_1,

$$\begin{aligned} & -L + \lambda_0 \|w_0\|_2^2 + \lambda_1 \|w_1\|_2^2 \\ = & -logP(D) + \lambda_0 \|w_0\|_2^2 + \lambda_1 \|w_1\|_2^2 \\ = & \sum_{i=1}^{m} (y_i logP(y_i = 0|x) + (1 - y_i) logP(y_i = 1|x)) + \lambda_0 \|w_0\|_2^2 + \lambda_1 \|w_1\|_2^2 \end{aligned}$$

Where $P(y_i = 0|x), P(y_i = 0|x)$ are defined above in 3.2.1 and $\lambda_0, \lambda_1 > 0$. Now we show that this objective function is strictly convex. As shown in class, -L (i.e. cross entropy) is a convex function (i.e. Hessian matrix is positive semidefinite). Moreover, from previous two homework, we have already proved that the regularization terms $\lambda_0||w_0||_2^2$, $\lambda_1||w_1||_2^2$ are strictly convex (i.e. Hessian matrix is positive definite). Therefore, by the properties of convex function, adding strictly convex functions with convex functions results in a strictly convex function. Therefore, the objective function is a strictly convex function, with unique global minimum. In other words, adding a constant vector **b** to the optimal w_0, w_1 can only results in increase of objective function and thus can never be the other plausible optimal solution.

4 Programming

4.1 Implementation of SVM

Done.

4.2 Cross validation of SVM

C	4^{-6}	4^{-5}	4^{-4}	4^{-3}	4^{-2}
Accuracy	0.7940	0.8030	0.8080	0.8090	0.8170
Speed (second)	183.0481	153.6179	127.0685	120.7289	115.3971

C	4^{-1}	1	4^{1}	4^{2}
Accuracy	0.8100	0.8130	0.8079	0.8069
Speed (second)	117.2368	120.6426	121.4721	114.7466

Increase of C is associated with decrease of time cost, by decreasing the scale of w_i and thus more speeding up quadratic operations of w_i. Increase of C results in increase of accuracy and when it reaches certain plateau, it decreases its accuracy. Indeed, putting more weight on slack variables results in over-fitting and thus lower accuracy, whereas, putting more weight on $||w||_2^2$ term results in over-regularization and thus also lower accuracy.

4.3 Linear SVM in libsvm

4.3.1 Accuracy

Accuracy = 0.791 for all parameter settings.

4.3.2 Speed

Average speed = 5.1576 seconds.

4.4 Kernel SVMs in libsvm

4.4.1 Accuracy for polynomial SVM

Accuracy = 0.793 for all costs settings with d=1.
Accuracy = 0.714 for all costs settings with d=2.

Accuracy = 0.806 for all costs settings with d=3.

4.4.2 Speed for polynomial SVM

Average speed = 0.511 seconds for all costs settings with d=1.

Average speed = 0.6224 seconds for all costs settings with d=2.

Average speed = 0.5955 seconds for all costs settings with d=3.

4.4.3 Accuracy for polynomial SVM

Accuracy = 0.551 for all parameter settings.

4.4.4 Speed for polynomial SVM

Average speed = 1.0521 seconds for all costs settings with $gamma = 4^{-7}$.

Average speed = 0.9069 seconds for all costs settings with $gamma = 4^{-6}$.

Average speed = 0.9181 seconds for all costs settings with $gamma = 4^{-5}$.

Average speed = 0.9172 seconds for all costs settings with $gamma = 4^{-4}$.

Average speed = 0.9444 seconds for all costs settings with $gamma = 4^{-3}$.

Average speed = 0.9378 seconds for all costs settings with $gamma = 4^{-2}$.

Average speed = 0.9268 seconds for all costs settings with $gamma = 4^{-1}$.

4.4.5 Chose one kernel

I would chose polynomial kernel, with d=3 parameter setting, based only on current observation.