CS 475 Machine Learning: Homework 3
Supervised Classifiers 2
Due: Friday October 16, 2015, 11:59pm
100 Points Total Version 1.1

Li-Yi Lin / llin34@jhu.edu

1 Analytical (50 points)

The following problems consider a standard binary classification setting: we are given n observations with m features, $x_1, ..., x_n \in \mathbb{R}^m$.

1) Overfitting (8 points) SVMs using nonlinear kernels usually have two tuning parameters (regularization parameter C and kernel parameter γ), which are usually determined by cross validation.

(a) Suppose we use cross validation to determine the non-linear kernel parameter and slack variable for an SVM. We find that classifiers using parameters (c_1, γ_1) and (c_2, γ_2) achieve the same cross validation error, but (c_1, γ_1) leads to fewer support vectors than (c_2, γ_2). Explain which set of parameters should we choose for the final model?
Ans:
When two sets of parameters achieve the same cross validation error, we should choose the model with lower variance, which is more general, for the final model. The reason is that, under the same training data, if one model has more support vectors, it means the model uses more features (higher dimension) for the kernel. And a model using more features (higher dimension) for the kernel means it is more likely to be over-fitting. Therefore, we should choose the model with fewer support vectors.

(b) The optimization problem of linear SVMs can be in either primal or dual form. As we know, the primal form has m parameters to learn, while the dual form has n parameters to learn. If $m \gg n$, is it true that the dual form reduces over-fitting since it has fewer parameters? Explain.
Ans:
It is false that if $m \gg n$, the dual form reduces over-fitting since it has fewer parameters. The reason is that the key factors of avoiding over-fitting when using kernel are the regularization (c) and kernel (γ) parameters. When c becomes larger, it will make the model more likely to be over-fitting. γ also have the similiar effect as c does, when c becomes larger, it will make the model more likely to be over-fitting.

2) Hinge Loss (12 points) Linear SVMs can be formulated in an unconstrained optimization problem

$$\min_w \sum_{i=1}^{n} H(y_i(w^T x_i)) + \lambda \|w\|_2^2, \tag{1}$$

where λ is the regularization parameter and $H(a) = \max(1-a, 0)$ is the well known hinge loss function. The hinge loss function can be viewed as a convex surrogate of the 0/1 loss function $I(a \leq 0)$.

(a) Prove that $H(a)$ is a convex function of a.

Ans:

If $H(a)$ function is convex, then it must satisfy

$$H(ta_1 + (1-t)a_2) \leq tH(a_1) + (1-t)H(a_2)$$

$$\forall t \in [0, 1]$$

We substitute $H(a) = \max(1-a, 0)$ to the above equation, then we have to prove the following equation

$$\max(ta_1 + (1-t)a_2, 0) \leq t \max(a_1) + (1-t)\max(a_2)$$

Since the value of $H(a) = \max(1-a, 0)$ function will be 0 if $a \geq 1$ and $1-a$ if $a < 1$, we will discussion three situations: (1) $1 \leq a_1 \leq a_2$; (2) $a_1 \leq a_2 \leq 1$; (3) $a_1 < 1 < a_2$.
(1) when $1 \leq a_1 \leq a_2 => 1 \leq ta_1 + (1-t)a_2$

$$H(ta_1 + (1-t)a_2) = 0$$

$$H(a_1) + H(a_2) = 0 + 0 = 0$$

Since $0 = 0$, we have proved that

$$H(ta_1 + (1-t)a_2) \leq tH(a_1) + (1-t)H(a_2)$$

(2) when $a_1 \leq a_2 < 1 => ta_1 + (1-t)a_2 < 1$

$$H(ta_1 + (1-t)a_2) = 1 - (ta_1 + (1-t)a_2) = 1 - ta_1 - a_2 + ta_2$$
$$tH(a_1) + (1-t)H(a_2) = t(1-a_1) + (1-t)(1-a_2) = 1 - ta_1 - a_2 + ta_2$$

Since $1 - ta_1 - a_2 + ta_2 = 1 - ta_1 - a_2 + ta_2$, we have proved that

$$H(ta_1 + (1-t)a_2) \leq tH(a_1) + (1-t)H(a_2)$$

(3) if $a_1 < 1 < a_2$:
(3.1) when $ta_1 + (1-t)a_2 \geq 1 => 1 - (ta_1 + (1-t)a_2) \leq 0$

$$H(ta_1 + (1-t)a_2) = \max(1 - (ta_1 + (1-t)a_2), 0) = 0$$

$$tH(a_1) + (1-t)H(a_2) = t(1-a_1) + 0 = t(1-a_1)$$

Since $t(1-a_1) \geq 0$, we have proved that

$$H(ta_1 + (1-t)a_2) \leq tH(a_1) + (1-t)H(a_2)$$

(3.2) when $ta_1 + (1-t)a_2 < 1 => 1 - (ta_1 + (1-t)a_2) > 0$

$$H(ta_1 + (1-t)a_2) = \max(1 - (ta_1 + (1-t)a_2), 0) = 1 - (ta_1 + (1-t)a_2)$$

$$tH(a_1) + (1-t)H(a_2) = t(1-a_1) + (1-t)0 = t(1-a_1)$$

Since

$$\begin{aligned} &tH(a_1) + (1-t)H(a_2) - H(ta_1 + (1-t)a_2) \\ &= t(1-a_1) - (1 - (ta_1 + (1-t)a_2)) \\ &= t - ta_1 - 1 + ta_1 + a_2 - ta_2 \\ &= t(1-a_2) - (1-a_2) \\ &= (t-1)(1-a_2) \geq 0 \end{aligned}$$

we have $tH(a_1) + (1-t)H(a_2) - H(ta_1 + (1-t)a_2) \geq 0$. Therefore, we have proved that $H(ta_1 + (1-t)a_2) \leq tH(a_1) + (1-t)H(a_2)$. Thus, $H(a)$ is a convex function.

(b) The function $L(a) = \max(-a, 0)$ can also approximate the 0/1 loss function. What is the disadvantage of using this function instead?

Ans:

When using the function $L(a) = \max(-a, 0)$, the width of the margin for the SVM model will be 0 since the original Hinge loss will become $\max(-yi(wTxi), 0)$. This situation will make the dicision boundary become more likely close to some training instances because it will not adjust the dicision boundary until one training instance is acrossing the decision boundary. So, it will become more likely to classify future instance incorrectly. Therefore, the disadvantage of this kind of hinge loss function will make the SVM have less "confidence" in classifying future instances.

(c) If $H'(a) = \max(0.5 - a, 0)$, show that there exists λ' such that (2) is equivalent to (1). Hint: think about the geometric interpretation of hinge loss.

$$\min_w \sum_{i=1}^{n} H'(y_i(w^T x_i)) + \lambda' \|w\|_2^2. \tag{2}$$

Ans:

Since the meaning will not change even if we scale the w, we can scale w by multiplying it with 0.5. Then we get the following objective function:

$$\begin{aligned} &\min_w \sum_{i=1}^{n} H'(y_i(0.5w^T x_i)) + \lambda' \|0.5w\|_2^2 \\ &= \min_w \sum_{i=1}^{n} \max(0.5 - y_i(0.5w^T x_i)) + \lambda' \|0.5w\|_2^2 \\ &= \min_w \sum_{i=1}^{n} 0.5 \max(1 - y_i(w^T x_i)) + 0.25\lambda' \|w\|_2^2 \\ &= 0.5 \min_w \sum_{i=1}^{n} \max(1 - y_i(w^T x_i)) + 0.5\lambda' \|w\|_2^2 \end{aligned}$$

When $0.5\lambda' = \lambda => \lambda' = 2\lambda$, we have found a λ' such that (2) is equivalent to (1) because the solution of w will be the same.

3) Kernel Trick (10 points) The kernel trick extends SVMs to handle with nonlinear data sets. However, an improper use of a kernel function can cause serious over-fitting. Consider the following kernels.

(a) Polynomial kernel: $K(x, x') = (1 + (x^T x'))^d$, where $d \in \mathbb{N}$. Does increasing d make over-fitting more or less likely?
Ans:
The polynomial kernel $K(x, x') = (1+(x^T x'))^d$ can be written as $\sum_{n=0}^{d} \binom{d}{n} 1^n (x^T x')^{d-n}$. As we can see, when d increases, the kernel will have more dimensions and more dimensions will make the model over-fitting more likely. Therefore, increasing d will make over-fitting more likely.

(b) Gaussian kernel: $K(x, x') = \exp(-\|x - x'\|^2/2\sigma^2)$, where $\sigma > 0$. Does increasing σ make over-fitting more or less likely?
Ans:
Increasing σ will make over-fitting less likely because when σ increases to a very large (infinite) number, the Gaussian kernel, $K(x, x') = \exp(-\|x-x'\|^2/2\sigma^2) = \exp(-0) = 1$, will always have the same value for every instance. Therefore this will make the model have higher bias, in other words, less variance and less over-fitting.

We say K is a kernel function, if there exists some transformation $\phi : \mathbb{R}^m \to \mathbb{R}^{m'}$ such that $K(x_i, x_{i'}) = \langle \phi(x_i), \phi(x_{i'}) \rangle$.

(c) Let K_1 and K_2 be two kernel functions. Prove that $K(x_i, x_{i'}) = K_1(x_i, x_{i'}) + K_2(x_i, x_{i'})$ is also a kernel function.
Ans:

$$\begin{aligned} K_1(x_i, x_{i'}) + K_2(x_i, x_{i'}) &= \langle \phi_1(x_i), \phi_1(x_{i'}) \rangle + \langle \phi_2(x_i), \phi_2(x_{i'}) \rangle \\ &= \sum_{n=1}^{m_1} \phi_{1n}(x_i)\phi_{1n}(x_{i'}) + \sum_{n=1}^{m_2} \phi_{2n}(x_i)\phi_{2n}(x_{i'}) \\ &= [\phi_1(x_i), \phi_2(x_i)]^T [\phi_1(x_{i'}), \phi_2(x_{i'})] \end{aligned}$$

where $[\phi_1(x_i), \phi_2(x_i)]$ means a vector that concatenats $\phi_1(x_i)$ and $\phi_2(x_i)$.
We let $\phi_3(x_i) = [\phi_1(x_i), \phi_2(x_i)]$ and $\phi_3(x_{i'}) = [\phi_1(x_{i'}), \phi_2(x_{i'})]$. In addition, we let $K(x_i, x_{i'}) = \langle \phi_3(x_i), \phi_3(x_{i'}) \rangle$. Then we have

$$[\phi_1(x_i), \phi_2(x_i)]^T [\phi_1(x_{i'}), \phi_2(x_{i'})] = \langle \phi_3(x_i), \phi_3(x_{i'}) \rangle$$

Hence, we have proved that

$$K(x_i, x_{i'}) = K_1(x_i, x_{i'}) + K_2(x_i, x_{i'})$$

4) Prediction using Kernel (8 points) One of the differences between primal linear SVMs and dual kernel SVMs concerns computational complexity at prediction time.

(a) What is the computational complexity of prediction of a primal linear SVM in terms of the numbers of the training samples n and features m?
Ans:
When using primal linear SVM, it has to compute the value of $w^T\phi(x) + b$ for each instance. So the complexity of predicting one instance is related to the dimension of w, namely the feature number m. Hence, the complexity of using primal linear SVM to predict one instance is $O(m)$.

(b) What is the computational complexity of prediction of a dual kernel SVM in terms of the numbers of the training samples n, features m, and support vectors s?
Ans:
When using dual kernel SVM, it has to compute the value of $\sum_{i\in\text{support vectors}} \alpha_i y_i K(x, x_i)$ for each instance. So the complexity of predicting one instance is related to the number of support vectors, s, and the complexity of kernel. Assume we use Gaussian kernel. The kernel function is $\exp(-\|x - x'\|^2/2\sigma^2)$ and its complexity is related to the dimension of x, which is m. Therefore, the totla complexity of using Gaussian kernel SVM to predict one instance is $O(ms)$.

5) Stochastic Gradient Algorithm (12 points) The stochastic gradient algorithm is a very powerful optimization tool to solve large-scale machine learning problems. Instead of computing the gradient over the entire data set before making an update, the stochastic gradient algorithm computes the gradient over a single sample, then updates the parameters. By passing over the entire data set of n samples in this fashion we can converge to the optimal parameters.

A single iteration of stochastic gradient considers a single example. While it takes many more iterations, each iteration is much faster, both in terms of memory and computation.

Consider a ridge regression problem with n samples:

$$\hat{\beta} = \arg\min_{\beta} \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i^T\beta)^2 + \lambda\|\beta\|_2^2. \tag{3}$$

In each iteration, instead of using only one example, we randomly choose k out of n samples and obtain $(x_{1'}, y_{1'}), ..., (x_{k'}, y_{k'})$.

(a) What is the computational complexity of computing the mini-batch stochastic gradient or gradient at each iteration (using k and n samples respectively)?
Ans:
We let $F_k = \frac{1}{k}\sum_{i=1}^{k}(y_i - x_i^T\beta)^2 + \lambda\|\beta\|_2^2$.
For finding the gradient descent at each iteration, the equation will be:

$$\beta' = \beta + \eta \bigtriangledown F_k$$

$$\bigtriangledown F_k = \frac{\partial F}{\partial \beta} = \frac{2}{k}\sum_{i=1}^{k}(y_i - x_i^T\beta)(-x_i) + 2\lambda\beta$$

As we can see, the complexity at each iteration is related to computing $\bigtriangledown F_k$. So the complexity is $O(km)$ due to computing $x_i^T \beta$ for k times. If using gradient (n samples at each iteration), then the complexity is $O(nm)$ at each iteration.

(b) What are the advantages/disadvantages of increasing k in terms of computational complexity (using k and n samples respectively)? What is traded-off by increasing/decreasing k?

Ans:

The advantage of increasing k in terms of computational complexity is that the parameters will walk toward a direction that is more like the direction to the optimal point. In other words, it has more confidence in knowing which direction the gradient descent update should go because it consider more data in one iteration. So that the total number of iteration might be reduced and might reduce the total computational complexity. However, when increasing k, the total computational complexity of each stochastic gradient descent will also increase because it has to consider more instances in one iteration. If the number of k is very large, then it have to wait for a long time to get a single update for the parameters.

(c) Give one advantage and one disadvantage of using stochastic gradient descent with $k = 1$ for a possibly nonconvex optimization problem, ignoring computational considerations. Explain.

Ans:

When using the stochastic grdient descent with $k = 1$ for a possibly nonconvex problem, it will have the chance to go out from a local optimal because it use one instance for updating the parameters so that it will not "walk" toward the (local) optimal directly. On the other hand, it might stay at a local optimal if a gradient descent is used. Hence, the advantage of using stochastic gradient is that it will have a higher chance to go out from a local optimal and find the global optimal in a nonconvex problem.

However, using stochastic descent usually needs more "steps (iterations)" than gradient descent because it will not walk toward the optimal point directly. In addition, because it updates the parameters using one instance for each iteration, stochastic gradient descent might walk around the global optimal but not reach the global potimal.