CS 475 Machine Learning: Homework 2
Supervised Classifiers 2

Name : Jin Yong Shin
JHED: Jshin44

October 4, 2016

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 (10 points) Consider a kernel logistic regression classifier. The classifier has regularization parameter λ, as well as a kernel parameter γ which is used by a non-linear kernel (the exact kernel is unimportant. These parameters can be determined by tuning on held-out development data, or by using cross validation.

(a) Suppose we use cross validation to determine λ and γ and find that the values (λ_1, γ_1) and (λ_2, γ_2) achieve the same cross validation error. What other factors should you consider in determining which are the appropriate parameters to select for the final trained model?

answer : The set (λ_1, γ_1) and (λ_2, γ_2) have same cross validation error. This means that their bias level are similar and project smoothly with good kernel value(Overfitting). The parameter that we have to consider will be γ parameter which will change the smoothness of the kernel projection and will change the loss. If we reduce the γ, we can get different error value for the loss of the kernel function. In addition, the reason getting the same cross validation error may come from the partition of the data set. If they are using same partition of the data set and similar parameter values, they may have same error. However, if we change the partition of the cross validation data set, then we can have different error.

(b) Just as with SVMs, we can fit a linear logistic regression classifier using either the primal or dual form. As we know, the primal form has m parameters to learn (number of features), while the dual form has n parameters to learn (number of examples). When $m \gg n$, will the dual form reduce over-fitting since it has fewer parameters? Explain your answer.

answer : In primal form, we can solve for the vector w which is the length of features, that is coming from variables. In here, we have m parameters for primal form. Additionally, dual form has n parameters to solve which we can solve for the vector a, denoted as length of examples. As problem has explained, when $m >> n$, it is better to use dual form because we are not free to choose a parameter for each feature. In a linear combination of inputs, we can only chose the parameters for a. Therefore, no matter how large our feature space projection, there are only n many as. Intuitively,

by transforming from primal to dual form, we can introduce slack variables to find maximum margin in the features. Therefore, by finding maximum margin, we can avoid overfitting (Generalization works).

2) Hinge Loss (12 points) Linear SVMs using a square hinge loss can be formulated as an unconstrained optimization problem:

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

where λ is the regularization parameter and $H(a) = \max(1-a, 0)^2$ is the square 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) Compared with the standard hinge loss function, what do you think are the the advantages and disadvantages of the square hinge loss function?

answer : (i) Advantages: As the prediction gets close to the correct answer, the loss will be reduced more than linear hinge loss. For example, if the prediction a $= \frac{1}{2}$, then squared hinge loss will give loss of $\frac{1}{4}$ while regular hinge loss will give $\frac{1}{2}$. Therefore, squared hinge loss will get more correct loss as the predictions get close to correct answer.
(ii) Disadvantages: This will penalize more if we predict wrong. This means that as it gets less correct, the outliers will get highly penalized compared to linear hinge loss.

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

answer: (i)To prove the convexity, we need to prove that $H(\frac{a+b}{2}) \leqslant \frac{H(a)+H(b)}{2}$. for $a, b \leqslant 0$. Here we are using $H(a) = max(1-a, 0)^2$. $H(\frac{a+b}{2}) = max(1 - \frac{a+b}{2}, 0)^2 = (\frac{2-a-b}{2})^2$. Now on the right hand side, $\frac{H(a)+H(b)}{2} = \frac{(1-a)^2+(1-b)^2}{2}$.
$(\frac{2-a-b}{2})^2 \leqslant \frac{(1-a)^2+(1-b)^2}{2}$.
(ii) There is another simple way of proving convexity. Take first derivative of the hinge loss.
$H'(a) = 2(a-1)$ for $a \leqslant 1$
$H'(a) = 0$ for $a \geqslant 1$
Then if we take second derivative, $H''(a) = 2 > 0$. From the derivative and convexity rule, if second derivative is bigger than 0, there is at least one global minimum, which means $H(a)$ is convex function of a.

(c) The function $L(a) = \max(-a, 0)^2$ can also approximate the 0/1 loss function. What is the disadvantage of using this function instead?

answer: The new function $H'(a) = max(-a, 0)^2$ has drawback in prediction and calculating correct loss. For example, at a = 0, standard squared loss function will give loss of 1 while new loss function will give 0. But a = 0 is not a correct prediction. Therefore, new loss function is not calculating loss correctly based on the prediction (poor performance).

(d) We can choose a different loss function $H'(a) = \max(0.5-a, 0)^2$. How will switching to H' from H effect the solution of the objective function? Specifically, the new

objective becomes:

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

Explain your answer in terms of the relationship between λ and λ'.

answer: The new hinge function $H'(a) = max(0.5 - a, 0)^2$ will shift the hinge function to the left from the original position when we use $H(a) = max(1 - a, 0)^2$. This means that new hinge function will sit below original and give less loss value than the original loss function. This will change our objective function value. Assuming that we are keeping weight vector constant with the objective function, we need to increase λ' from original λ to get same weights from the objective function. We need to put more regularization to keep it under-fit.

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

(a) Inverse Polynomial kernel: given $\|x\|_2 \leq 1$ and $\|x'\|_2 \leq 1$, we define $K(x, x') = 1/(d - x^\top x')$, where $d \geq 2$. Does increasing d make over-fitting more or less likely?

answer : With large d, the features in the set will vary less smoothly since it will cause lower kernel value (inner product of x features). By lowering the kernel, it will give SVM lower flexibility by restricting decision boundary. Low kernel will restrict decision boundary so that it will work just like linearly. (Less Sensitive) This means that it will have higher bias and lower variance. Therefore, it is less likely over-fitted.

(b) Chi squared kernel: Let x_j denote the j-th entry of x. Given $x_j > 0$ and $x'_j > 0$ for all j, we define $K(x, x') = \exp\left(-\sigma \sum_j \frac{(x_j - x'_j)^2}{x_j + x'_j}\right)$, where $\sigma > 0$. Does increasing σ make over-fitting more or less likely?

answer: With large σ, the features in the set will vary less smoothly. In kernel, estimating σ is critical since the behavior of the kernel is distinguished by σ. Therefore, if we have overestimated σ, which means high value of it, we will restrict the flexibility of kernel and make it act like linear kernel (Decision boundary less sensitive). This means that it will have higher bias and lower variance. Therefore, it is less likely over-fitted.

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.

answer: Let ϕ^1 and ϕ^2 (not squared, just denoting 2)be the feature vectors associated with K_1 and K_2, where $\phi = [\phi_1^1 \phi_2^1 ..., \phi_1^2 \phi_2^2 ...]$. In here we just need to show inner product of ϕ functions. Then $< \phi(x), \phi(x') >= \sum_{i=1}^{N} \phi_i(x)$ x $\phi_i(x') = \sum_{i=1}^{m} \phi_i^1(x)$ x $\phi_i^1(x') + \sum_{i=1}^{m} \phi_i^2(x)$ x $\phi_i^2(x')$. Then we will get $< \phi^1(x), \phi^1(x') > + < \phi^2(x), \phi^2(x') >$.This will equal to $K_1(x, x') + K_2(x, x') = K(x, x')$. Therefore, the proof is shown.

4) Predictions with Kernel (6 points) Let us compare primal linear SVMs and dual linear kernel SVMs in terms of 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? If each sample contains at most q nonzero features, what can we do to accelerate the prediction? What is the resulting computational complexity?

answer: Computational complexity of a linear SVM is depending on the number of the inputs. (Specifically input dimension). Linear model of the SVM will be stored and evaluated with the inner product of the example vectors and feature vectors. In here, we have m and n vectors respectively. Therefore, the computational complexity of a linear SVM is going to be $O(m * n)$. If every example contains at most q nonzero features, then the worst case of inner product of feature vector and example vector will be $O(q * n)$. This will be accelerating the prediction if we know the index set of the vector where it has 0 vector. Then we can make sparse vector to use quicker computation.

(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? If each sample contains at most q nonzero features, what can we do to accelerate the prediction? What is the resulting computational complexity?

answer: Dual kernel SVM computational complexity is proportional to support vector. As we have calculated in part(a), the linear SVM has $O(mn)$ computational complexity because it is depending on inner product of feature and example vectors. Therefore, computational complexity of dual kernel SVM will b $O(s * m * n)$ since it has s support vectors. Also same as the previous problem, we can make sparse vector of the feature vector since it has some zero vectors. Therefore, the computation complexity will be reduced to $O(s * q * n)$.

5) Dual Perceptron (6 points)

(c) You train a Perceptron classifier in the primal form on an infinite stream of data. This stream of data is not-linearly separable. Will the Perceptron have a bounded number of prediction errors?

answer: If we are training on a perceptron classifier and the stream of the data is not-linearly separable, then we can use kernel SVM to raise the dimension to separate the data stream. We usually raise the data stream into the dimension of data stream + 1 to separate. However, as the problem has mentioned, the data stream is infinite and we cannot raise to infinite + 1 dimension. Therefore, there is no bounded number in prediction error.

(c) Switch the primal Perceptron in the previous step to a dual Perceptron with a linear kernel. After observing T examples in the stream, will the two Perceptrons have learned the same prediction function?

answer: Yes, they are learning same prediction function. Primal perceptron and dual perceptron both are using support vector to construct hyperplane for classification model. Primal and dual formulations are complimentary even though their way

of learning is different. Therefore, learning prediction function will give same prediction function for the other. Primal function minimized objective function and use vector to minimize the function while dual formation will maximize objective function with dual variables. By meaning of linear programming, maximizing is another way of minimizing. Therefore, they will learn same prediction function.

(c) What computational issue will you encounter if you continue to run the dual Perceptron and allow T to approach ∞? Will this problem happen with the primal Percepton? Why or why not?

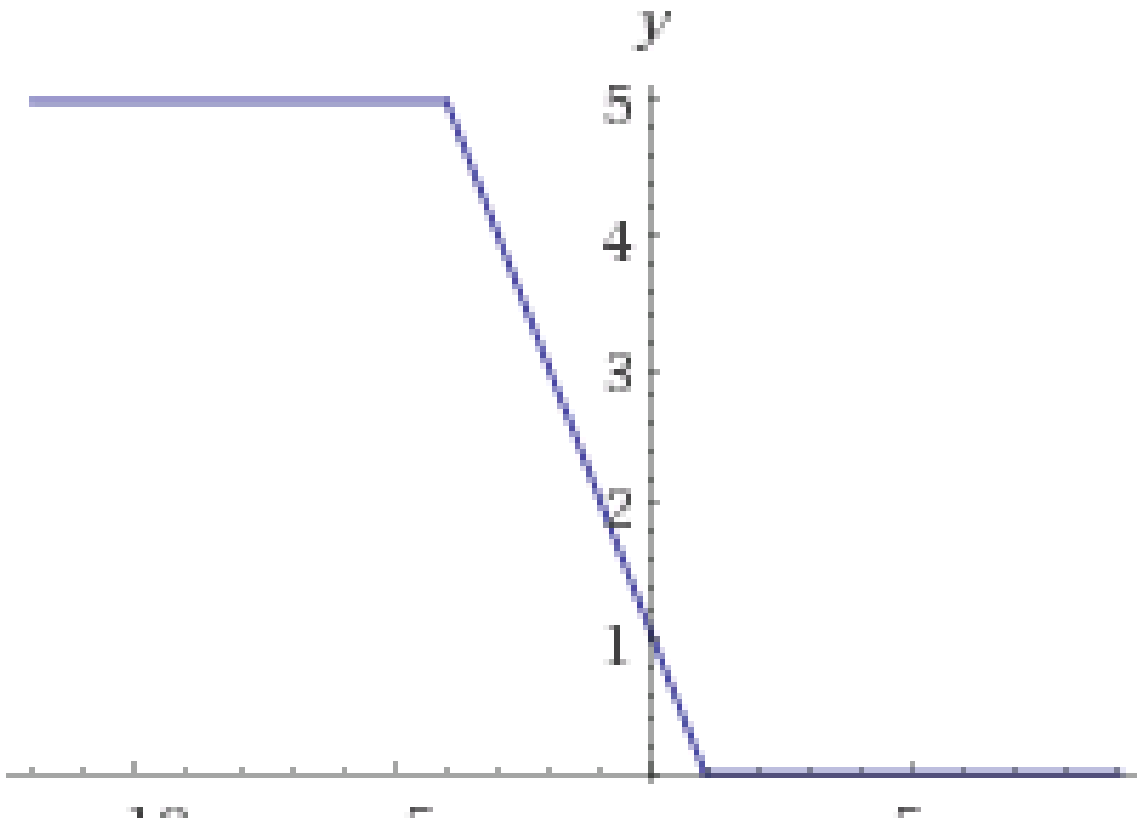
answer: Computational complexity associated with dual perceptron is $O(Td)$ where T is the size of the training set and d is the size of the parameter vector. Then if we raise T to infinite, then the computational complexity will be $O(Td) = \infty$, which we won't be able to compute the dual perceptron and the program will run forever. However, primal computational complexity does not depend on examples. Since primal perceptron complexity depends on features, we can raise T to ∞. Primal form would not have computational issue with large T

6) Robust SVM (6 points) For SVMs, the hinge loss can deal with some examples that are not linearly separable. However in some cases, these examples may be outliers: data points that are so inconsistent with the rest of the data that we suspect they are incorrect. A good classifier should be robust to those outliers. Is it possible for us to modify the hinge loss to achieve our goal? What is the disadvantage? (Hint: is it possible to choose a nonconvex function?)

answer: One of the drawbacks from SVM is that the classifier is very sensitive to the outlier so that the loss function tend to penalize a lot for examples that are not consistent. Therefore, the support vector has too many in numbers to mess up the projection. The classifier by using original hinge loss function can create poor classification performance. Rather than penalizing outlier, we can put restriction to bound the loss function and reduce noise value. If we add restriction on the maximum loss function (the threshold amount), we can reduce the problem of penalizing outliers and prevent poor prediction. The new function will be:

$$H'(a) = max(0, 1 - a) - max(0, r - a)$$

for $r \leqslant 0$. For this case, we can choose our threshold r and we can prevent outlier being penalized. This function will be robust to outliers. The function will look like below if we pick $r = -5$. However, this robust hinge loss also have some drawbacks. First is that some estimable examples (not outlier) will be overlapped with other examples in loss wise. Even though margins are different, they will have same loss which can create poor prediction as well. Another drawback is that they are containing convex function.


y
5
4
3
2
1