Homework 2 Analysis Section

Introduction to Machine Learning

Jonathan Liu

jliu118

Contents

Problem 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?

Jon's Answer:

Another thing we should consider is the shape of the kernel function. If the kernel function is smoother then that means two vectors that are farther away from each other would be counted as more similar than the same two vectors being compared with a less smooth kernel function. So more vectors would be counted as similar, even if they actually aren't which is the definition of increasing bias which goes hand in hand with lowering variance, which decreases overfitting. And the opposite effect happens when the the kernel function is less smooth and more abrupt.

(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.

Not necessarily because there are a bunch of other factors that affect over-fitting and variance.

Problem 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?

One disadvantage to the square hing loss function is that outliers are given a really large loss, making convergence a lot slower. One advantage to using a square hinge loss is that the function becomes smooth, making differential calculations easier.

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

A linear function is convex because a line segment between any two points on the graph of the function lies above or on the graph, and the maximum of two convex functions is therefore convex, and the square function is also convex, so $H(a)$ is also convex.

(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?

Then examples when a is just barely above 0 will have a loss of 0 and counted as correctly classified, while it is actually ¡ 1 so it should be incorrect.

(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:

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

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

If the loss function is changed to H', then the margin is essentially reduced, so there are more examples counted as incorrect, which means that there is a higher chance of overfitting, so λ' would increase to counteract the overfitting.

Problem 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?

Increasing d makes overfitting less likely because as d increases, K approaches 0 and the kernel function becomes smoother, which reduces variance and thus reduces the chance of overfitting.

(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?

If you look at the general graph of the kernel function, as σ increases, the kernel function varies more abruptly, i.e. is less smooth and has a smaller influence area, so more support vectors are needed to cover boundaries, which can lead to overfitting.

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.

$K_1(x_i, x_{i'}) + K_2(x_i, x_{i'}) = [\phi(x_i)]\,[\phi(x_{i'})]^T + [\theta(x_i)]\,[\theta(x_{i'})]^T$

$= \begin{bmatrix} \phi(x_i) \\ \theta(x_i) \end{bmatrix} \cdot \begin{bmatrix} \phi(x_{i'})^T \\ \theta(x_{i'})^T \end{bmatrix} = \langle \omega(x_i), \alpha(x_{i'}) \rangle = K(x_i, x_{i'})$

Problem 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?

The computational complezity of prediction for a primal linear SVM is just $O(m)$ because when you're making a prediction, you take the dot product of x and w, and w which takes $O(m)$ time because w has m elements. If each sample contains at most q nonzero features then we can use a feature vector sparse vector so it only takes $O(q)$

(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?

$O(sm)$ because you use each support vector in the calculation of each kernal, so if there are s support vectors and m kernels, the time would be $O(sm)$. If each sample contains at most q nonzero features then we can use a sparse vector so we only need to iterate through the nonzero features so it only takes $O(q)$

Problem 5: Dual Perceptron (6 points)

(a) 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?

If the data is not linearly separable then the Perceptron algorithm won't converge because the algorithm will keep making mistakes because it's not linearly separable, therefore if there are is an infinite stream of data, then there will also be an unbounded number of prediction errors.

(b) 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?

Because solving one algorithm will give the solution for the other, the primal perceptron and the dual perceptron are have the 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?

We will run into the issue that we will have to store each an every support vector to make each consecutive prediction, which is impossible because then we would need an infinite amount of memory.

Problem 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?)

We should add a large slack variable to our hinge loss function, so that outliers within our large slack won't affect our SVM hypothesis. The disadvandtage to increasing slack is that more of the training examples will be wrong, so there will be more bias in our output.