6867 Problem Set 2

October 10, 2016

1. Logistic Regression

Linear regression on feature values X to predict discrete variable Y might produce predictions that larger than Y's maximum or less than Y's minimum, which prevents a mathematically feasible mapping to predicted probabilities. To facilitate this mapping, Logistic Regression transforms Linear Regression's output using the sigmoid function, so that it the predictions are bounded on $[0, 1]$.

$$\sigma(z) = \frac{1}{1 + e^z} \tag{1}$$

Using this formulation, we find weights tha minimize:

$$\min_w \sum_i \log\left(1 + e^{-y^{(i)}(w_0 + x^{(i)} \cdot w)}\right) + \lambda w^T w \tag{2}$$

where the left side is negative log likelihood of the model and where $\lambda$is a regularization penalty to prevent overfitting.

When we train logistic regression on the 2D datasets, with a decision boundary at .5, we get predictable results

Table 1: TODO: Logistic Regression on 2D Datasets

When the classes are linearly separable in dataset 1, the classifier performs very well. In the nonseparable case, it performs no better than a coin flip.

We also explore the effect of adding L1 and L2 regularization The figures show that for the cases that are not trivial or impossible, the classficattion error on the validation set decreases as we increase λ and L1 regularization performs slightly better than L2 regularization. In sample, less regularization is associated with less loss since overfitting is fine.

$/emphTODO : plot$

2. Support Vector Machines

Below. we show the equations for linear SVM with slack variables for an example problem with positive examples (1, 2), (2, 2) and negative examples (0, 0), (-2, 3) for linear SVM with slack variables.

$$\min_{\alpha_1,\alpha_2,\alpha_3,\alpha_4} \frac{1}{2} \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \end{bmatrix} \begin{bmatrix} 5 & 6 & 0 & -4 \\ 6 & 8 & 0 & -2 \\ 0 & 0 & 0 & 0 \\ -4 & -2 & 0 & 13 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} + \begin{bmatrix} -1 & -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix}$$

s.t.

Table 2: Classification Performance for Different SVM Parameters and Kernels

Table 3: Soft SVM Performance for Different Parameters and Kernels

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ C \\ C \\ C \\ C \end{bmatrix},$$

$$\begin{bmatrix} 1 & 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} = 0$$

1. soft SVM, in this case, uses a simple linear kernel, so the decision boundaries that are generated are still straight lines in $\mathbb{R}^2$. However, the slack provided by the soft SVM allows the algorithm to classify data and create a "margin", even when the data is not linearly separable. As expected, in general the algorithm does slightly better on the training set than the validation set.

$/emphTODO : plotboundaries$

However, note that as the data becomes less and less linearly separable, our soft SVM with a linear kernel does increasingly poorly. This motivates our decision to explore the use of a Gaussian (or RBF) kernel. Table 4 includes the performance of our soft SVM algorithm on various training sets when we change kernels (for both linear and Gaussian kernels), and also when we modify the bandwidth of our Gaussian kernel (where applicable) and change our regularization parameter, C.

Geometric margin tends to increase as the parameter C decreases, and the larger the geometric margin, the more support vectors. The more slack we allow, the more data points we will observe on or inside of our margin. As C goes to 0, the descision boundary will approach the hard SVM decision boundary, and the geometric margin will stop increasing.

Choosing C by maximizing the geometric margin encourages tiny values of C, which are likely to overfit the training data. A better procedure would select the value of C that minimizes classification error on the validation set.

3. Pegasos SVM

4.MNIST Classification

Table 4: MNIST Classification Performance