

Machine Learning 6.867 - Pset 2

October 18, 2015

1 Logistic Regression

1.1 Implementation

We implemented L_2 -regularized logistic regression using gradient descent.

1.2 Testing in data with $\lambda = 0$

We test the logistic regression

Table 1: Estimated logistic regrecoefficients

Data	w_0	w_1	w_2
stdev1	-66.3378	95.2461	101.1527
stdev2	-0.0466	0.7636	1.1148
stdev4	-0.0093	0.2363	0.2034
nonsep	0.0006	-0.0247	-0.0237

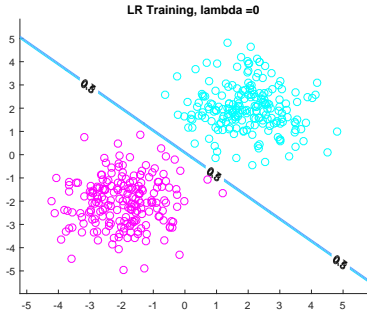
2 Support Vector Machine

Support Vector Machines are a popular classification method to construct linear or nonlinear decision boundaries by solving a convex optimization problem. There are two common forms of the optimization problem considered for SVM, which we refer to as the primal and dual. In this paper, we only consider the dual form, because it is computationally more tractable for many problems, and this method has the ability to generalize to different choices of kernel. The dual form of SVM for a general kernel function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is as follows:

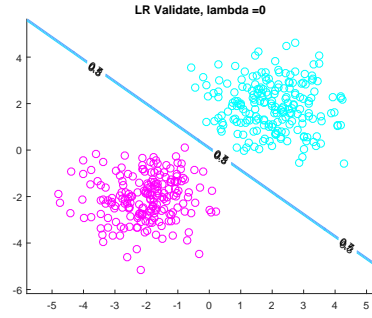
$$\begin{aligned} \max_{\alpha \in \mathbb{R}^n} \quad & \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y^{(i)} y^{(j)} k(x^{(i)}, x^{(j)}) \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, \quad i = 1, \dots, n, \\ & \sum_{i=1}^n \alpha_i y^{(i)} = 0. \end{aligned} \tag{1}$$

2.1 Implementation

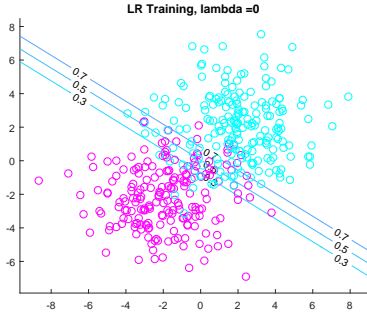
First, we implemented the dual form of the SVM with a linear kernel, where k is the usual dot product $k(x, z) = \langle x, z \rangle$ for all $x, z \in \mathcal{X}$. In MATLAB, we created a function with inputs: data $X \in \mathbb{R}^{n \times p}$, labels $Y \in \{-1, 1\}$, and cost parameter $C \in \mathbb{R}^+$. Within the function, we use the quadratic solver `quadprog` to solve the SVM dual problem (1) with these parameters to find the optimal α 's. Since `quadprog` requires that the problem fit into a certain functional form, we reformulate the problem (1) as follows:



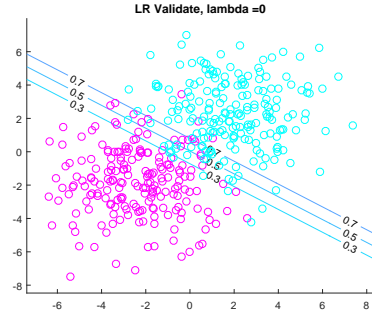
(a) Data with $\sigma = 1$, training



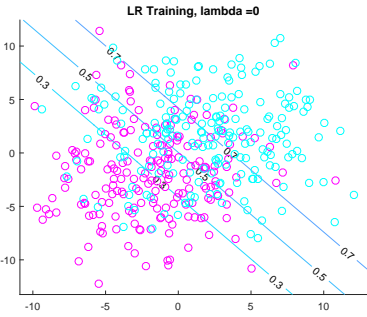
(b) Data with $\sigma = 1$, validation



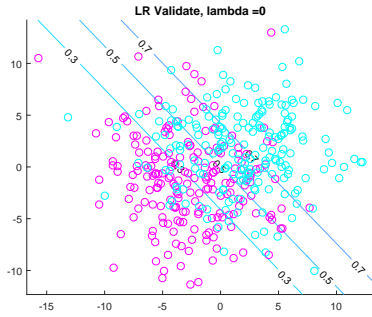
(c) CData with $\sigma = 2$, training



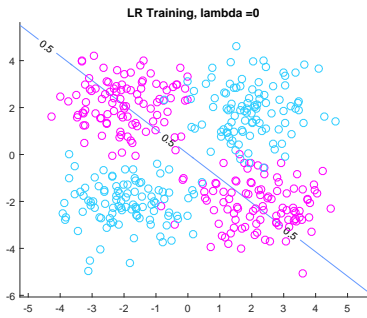
(d) CData with $\sigma = 2$, validation



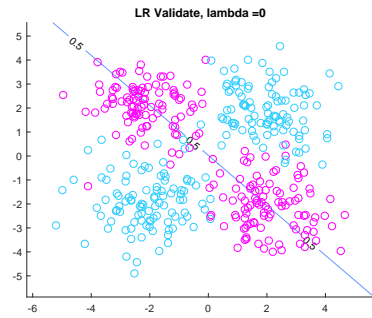
(e) Data with $\sigma = 4$, training



(f) Data with $\sigma = 4$, validation



(g) Non-seperable data, training



(h) Non-seperable data, validation

Figure 1

$$\begin{aligned}
 & - \min_{\alpha \in \mathbb{R}^n} \quad \frac{1}{2} \alpha^T H \alpha - \sum_{i=1}^n \alpha_i \\
 & \text{s.t.} \quad 0 \leq \alpha_i \leq C, \quad i = 1, \dots, n, \\
 & \quad \sum_{i=1}^n \alpha_i y^{(i)} = 0,
 \end{aligned} \tag{2}$$

where: $H \in \mathbb{R}^{n \times n}$ is a matrix with $(i, j)^{th}$ entry $H_{ij} = y^{(i)}y^{(j)}k(x^{(i)}, x^{(j)})$. Given the optimal solution $\alpha \in \mathbb{R}^n$ for the SVM problem with a linear kernel, the chosen linear decision boundary $\theta^T x + \theta_0 = 0$ is given by:

$$\theta = \sum_{i=1}^n \alpha_i y^{(i)} x^{(i)} \quad (3)$$

$$\theta_0 = \frac{1}{\mathcal{M}} \left(\sum_{j \in \mathcal{M}} \left(y^{(j)} - \sum_{i \in \mathcal{S}} \alpha_i y^{(i)} (x^{(j)})^T x^{(i)} \right) \right) \quad (4)$$

The output of our linear SVM function is $[\theta, \theta_0]$. We tested our function on the 2D example $X = \{(1, 2), (2, 2), (0, 0), (-2, 3)\}$, $Y = \{1, 1, -1, -1\}$. For this problem, the objective function generated for problem (2) is:

$$\frac{1}{2} \alpha^T H \alpha - \sum_{i=1}^4 \alpha_i, \quad (5)$$

where:

$$H = \begin{bmatrix} 5 & 6 & 0 & -4 \\ 6 & 8 & 0 & -2 \\ 0 & 0 & 0 & 0 \\ -4 & -2 & 0 & 13 \end{bmatrix}$$

The constraints are:

$$0 \leq \alpha_i \leq C, \quad i = 1, \dots, 4, \quad (6)$$

$$\alpha_1 + \alpha_2 - \alpha_3 - \alpha_4 = 0. \quad (7)$$

2.2 Performance on datasets

We tested our linear SVM function on the same 2D datasets from the previous section, with parameter $C = 1$.

2.3 Kernel SVM

We extended our SVM implementation in MATLAB to operate with more general kernels, taking the kernel function or kernel matrix as input.

Questions:

- As C increases, the geometric margin $1/\|\mathbf{w}\|$ decreases. If the data is not linearly separable, then the geometric margin $1/\|\mathbf{w}\|$ decreases strictly monotonically as C increases. However, if the data is linearly separable, then this does not always happen as we increase C . In this case, once the margin is sufficiently small such that all points are correctly classified, then it will not decrease further even as C approaches infinity.
- As C decreases, the number of support vectors decreases. This is because the larger penalty on misclassified points leads to a decision boundary with fewer misclassifications on the training data. However, the number of the support vectors is bounded below by two as C approaches infinity, because there will always be at least one support vector on each side of the decision boundary.
- Maximizing the geometric margin $1/\|\mathbf{w}\|$ on the training data is not an appropriate criterion for selecting C because this leads to a classifier which is overfit to the training set. To obtain a classifier which generalizes well on test data, we should use out-of-sample data to select an appropriate value for C . To do this, we can train the SVM model with $C = \{0.01, 0.1, 1, 10, 100\}$, and then select the value for C which yields the classifier with the highest accuracy on the validation set.

3 Titanic Data