Regression and SVM

October 27, 2015

Abstract

Previously, we discussed various methods for polynomial regression and minimizing error in our objective function, and explored famous examples, namely ridge regression and other norms (i.e. LAD for the L1 norm). Here, we will extend those concepts to a few classification problems, through logistic regression (a natural adaptation of the polynomial regression we've been doing) and then more formally through the concept of support vector machines (SVMs).

1 Logistic Regression

In past discussions, we've often encountered the objective function of our MLE estimate, or of some other estimate that we've established for some parameter θ that we'd like to minimize so as to minimize the value of our error function. This is called **regression**. Here we will discuss **logistic regression**, which involves fitting the data to a curve of the form

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

This is known as the **sigmoid** function, and it's often used when the data we would like to fit is well-modeled as a classification problem (with labels +1 and -1). Given d-dimensional data $x^{(1)}, x^{(2)}, ..., x^{(n)}$ and corresponding scalar labels $y^{(1)}, y^{(2)}, ..., y^{(n)}$, we can define our LR objection function as:

$$\text{NLL}(w) = \sum_{i=0}^{n} \log(1 + e^{-y^{(i)}(x^{(i)} \cdot w + w_0)})$$

given the parameters w, w_0. To reduce overfitting, we can apply an $L2$ regularization term to our equation, as in ridge regression. As a result, we are simply minimizing

$$E_{LR}(w) = \text{NLL}(w) + \lambda w^\intercal w$$

so that

$$w^* = \text{argmin}_w \left[\sum_{i=0}^{n} \log(1 + e^{-y^{(i)}(x^{(i)} \cdot w + w_0)}) + \lambda w^\intercal w \right]$$

We can estimate w numerically using stochastic gradient descent; if we let $\lambda = 0$, then we have no regularization and we end up with this data, as well as figures 1-8:

Table 1: $\lambda = 0$ (step = .001, threshold = .008)

	`stdev1`		`stdev2`		`stdev4`		`data_nonsep`	
w_{init}	#it	Loss	#it	Loss	#it	Loss	#it	Loss
$[1, 1, 1]$	255	0	91	0.09	49	0.32	37	0.48
$[4, 4, 4]$	466	0	259	0.12	91	0.36	64	0.44
$[1, 10, 0]$	408	0	354	0.11	39	0.33	32	0.53
$[-5, -5, -5]$	299	0	199	0.1	89	0.33	73	0.49

However, if we vary λ, then we obtain more stable solutions with respect to w, and the solutions are more constant regardless of the starting paramters. Figure 9-12 show the decision bounday for $\lambda = 20$.

Figures 1-4 show the training data, and figures 5-8 show the validation data for $\lambda = 0$. The predictor is reasonably effective and seems to seek to minimize the loss. When we increase λ, however, we instead increase the loss as we try to minimize the $\lambda w^\intercal w$ term in the error function. This means that the model is more inaccurate in this case, which is because with two parameters, w wasn't likely going to overfit to the data anyways. But note that the effect is not too large, even for larger λ ($\lambda = 40$) - our main goal is still to minimize the total error.

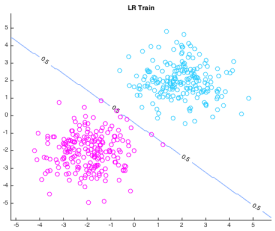

Figure 1: `stdev = 1`

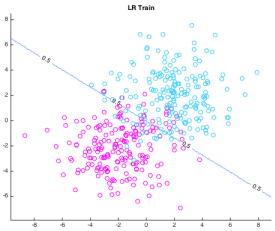

Figure 2: `stdev = 2`

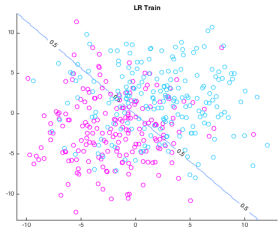

Figure 3: `stdev = 4`

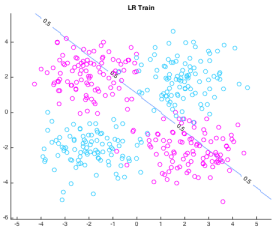

Figure 4: `unseparable`

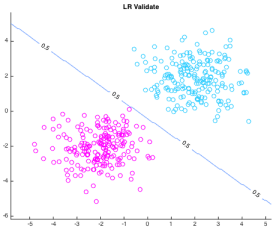

Figure 5: `stdev = 1`

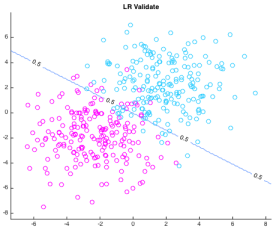

Figure 6: `stdev = 2`

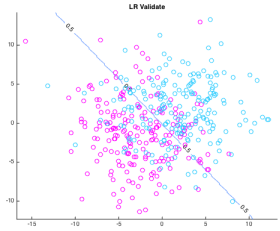

Figure 7: `stdev = 4`

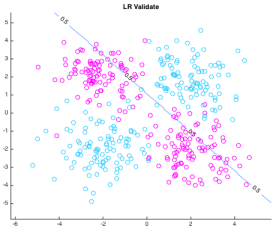

Figure 8: `unseparable`

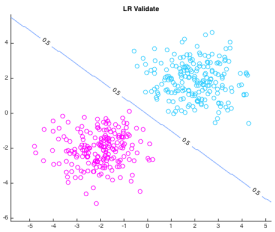

Figure 9: `stdev = 1`

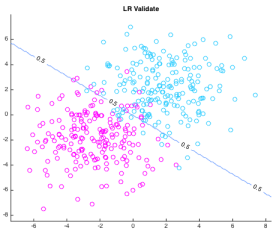

Figure 10: `stdev = 2`

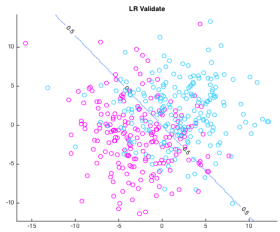

Figure 11: `stdev = 4`

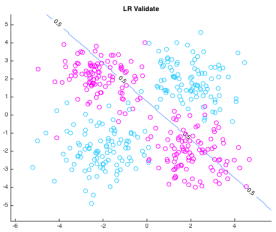

Figure 12: `unseparable`

2 Support Vector Machines

Support Vector Machines are a machine-learning technique that is used to classify binary sets of data. We define θ as the normal to the decision hyperplane, and classify data points by $\text{sgn}(\theta^\intercal x + \theta_0)$. In order to train the classifier, we initially set up the minimization problem (known as **Hard-SVM**):

$$\max_{\theta,\theta_0} \frac{1}{||\theta||} \min_{1\leq i\leq n} y^{(i)}(\theta^\intercal x^{(i)} + \theta_0) \tag{1}$$

However, we cannot satisfy this minimization if the training set is not linearly separable. Therefore, we add a "slack variable" to each constraint, which is a measure of the "wrongness" of the decision boundary when classifying that point. We call these slack variables ξ_i. We then desire to minimize

$$\min_{\theta,\theta_0,\xi} \frac{\lambda}{2}||\theta||^2 + \frac{1}{n}\sum_{i=1}^{n}\xi_i, \tag{2}$$

$$s.t. \quad y^{(i)}(\theta^\intercal x^{(i)} + \theta_0) \geq 1 - \xi_i, \tag{3}$$

$$\xi_i \geq 0, i \in n \tag{4}$$

However, we find it more computationally efficient to solve the dual form of the **Soft-SVM**:

$$\max_{\alpha \in \mathbb{R}^n} \left[\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} (x^{(i)})^\intercal x^{(j)} \right] \tag{5}$$

$$s.t. \quad 0 \leq \alpha_i \leq C \tag{6}$$

$$\sum_i \alpha_i y^{(i)} = 0 \tag{7}$$

To provide some intuition: α_i is the weight of each training point on the final decision boundary. It is 0 for all $x^{(i)}$ that are farther from the decision boundary than the margin, so that only the "important" $x^{(i)}$ are "support vectors". C is the maximum value of α, and determines the size of the margin. **ADD SOMETHING ABOUT WHICH WAY THAT WORKS HERE**. To return from α to the more familiar θ and θ_0, we plug in:

$$\theta = \sum_{i=1}^{n} \alpha_i y^{(i)} x^{(i)} \tag{8}$$

$$\theta_0 = \frac{1}{\mathcal{M}} \left[\sum_{j \in \mathcal{M}} \left(y^{(j)} - \sum_{i \in \mathcal{S}} \alpha_i y^{(i)} (x^{(j)})^\intercal x^{(i)} \right) \right] \tag{9}$$

Where $\mathcal{M} = \{i : 0 < \alpha_i < C\}$ and $\mathcal{S} = \{i : 0 < \alpha_i\}$.

For example, given the data $X = [1, 2;\ 2, 2;\ 0, 0;\ -2, 3], Y = [1;\ 1;\ -1;\ -1]$, we generate the following objective function:

$$\min_{\alpha \in \mathbb{R}^n} \left[\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} (x^{(i)})^\intercal x^{(j)} - \sum_{i=1}^{n} \alpha_i \right] \tag{10}$$

This can be reformulated to be plugged into a standard quadratic programming solver, with **blah blah blah**

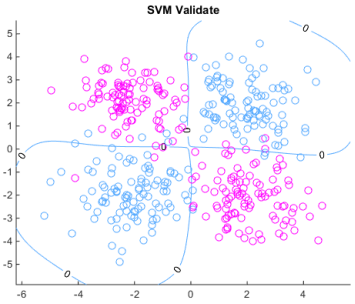

Figure 13: $C = 0.01, \sigma = 1$

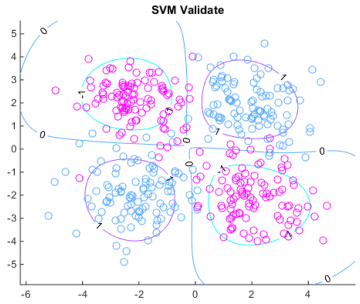

Figure 14: $C = 0.1, \sigma = 1$

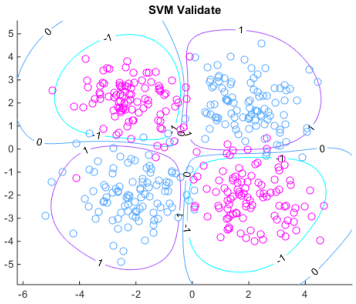

Figure 15: $C = 1, \sigma = 1$

We also note that as C increases, the number of boundary points decreases. Thus, the number of support vectors decrease. For the nonseparable example, for $C = (.01, .1, 1, 10, 100)$, we see that $\mathcal{M} = (400, 324, 133, 68, 59)$.

As per the usual procedure with the approximately equivalent λ, to select the correct C, we first compute α for various values of C using a training set, then determine the validation error using a validation set. We select the C which produces the lowest validation error in order to maximize the generality of our solution.

3 Titanic Data

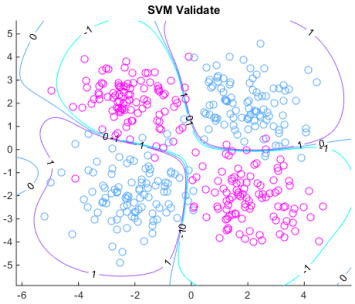

Figure 16: $C = 10, \sigma = 1$

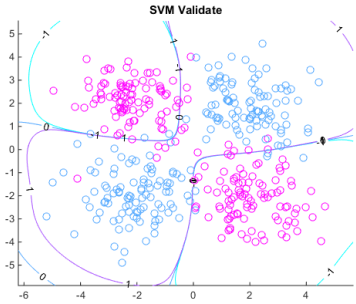

Figure 17: $C = 100, \sigma = 1$