6.867 Machine Learning Homework 2

1 LOGISTIC REGRESSION

Logistic Regression is a discriminative model used for classification. Given an input x, it finds the posterior of x belonging to one of the classes and then uses that probability to classify x. In the simplest case, it takes the dot product of x and w and uses that as an input to the sigmoid function, which outputs a number between 0 and 1. An advantage of logistic regressions is that they have few parameters, which allows them to be trained relatively quickly. One of the problems with logistic regressions is that they are very prone to overfitting to the training data. One way to prevent the overfitting is to add a regularization term, lambda, which penalizes the size of the weight vector. The size of the weight vector can be penalized using the L1 norm of the L2 norm. Here, we explore how different lambda values and the different norms affect several aspects of the logistic regression.

1.1 Optimizing with Gradient Descent

To investigate how l2 regularization affected the logistic regression we tried lambda values of 0 and 1. We decided not to penalize the bias term in the weight vector. We found that with a lambda value of 1 the weight vector decreased in every iteration of the algorithm until it converged to its optimal value. We believe this is because for most iterations, the quickest way to decrease the objective function is to decrease the norm of the weight vector, because it is penalized by a quadratic factor. With a lambda value of 0, the opposite happened. The norm of the weight increased in every iteration until it converged to its optimal value. Unregularized logistic regressions attempt to make the weight vector as large as possible because that makes the sigmoid function steeper, which in turn increases the log likelihood of the data. Our obersvations agree with our intuition that regularization makes the weight vector smaller.

1.2 Section1

You should be able to concisely represent trends for those configurations. For example, you could have one plot showing the effect of the error rate for different regularizers and lambdas, a good example of changing the decision boundary, and general observations, and maybe a plot about the weights. Perform your experiments and see what makes the most sense.

1.3 L1 vs L2 Norm

Two common metrics used to penalize the size of the weight vector are the L1 and L2 norm. The norms are defined as follows: (L1 is sum of absolute values). We tested the effect of each of these norms and different lambda values on the Classification Error Rate, decision boundary, and the weights of the weight vector. We used scikit-learn's implementation of Logistic Regression, which also penalizes the bias term. To allow for unpenalized bias, the logistic regression has an additional parameter, intercept$_{s}caling, whichisafactorbywhichtheinterceptismultipliedbytooffsettheeffectsoftheregularizatio$

Effect on Classification Error Rate: In general, we found that the regularization seemed to have a larger effect on the CER when using the L1 norm then when using the L2 norm. The L2 norm logistic regression was able to maintain low CER's even with high lambda values. The L1 norm, on the other hand, often drove the weight vector to zero, which increased the CER. For most data sets, the L2 norm achieved a better CER than the L1 norm when set to the same lambda values. When both weight vectors were nonzero, their CER's usually only differed by a percentage point. The difference between the CER's grew dramatically whenever the L1 norm drove one of the weights in the weight vector to zero. The two models generally behaved that way except for the last data set, in which the data was linearly inseparable. There, the L1 and L2 norm achieved very similar CER scores for every lambda value we tested.

Effect on Decision Boundary As mentioned in the previous section, the L1 norm tends to drive the weights of the weight vector to zero as lambda increases. As a result, there were several instances where the decision boundary was simply the x axis CHECK THIS. In general, we found that decreasing lambda changed the decision boundary in a direction where it could better separate the training data. This was true for all of the data sets where the data was linearly separable. For the data set in which the data was linearly inseparable, the decision boundary never changed, regardless of the lambda value.

Effect on Weights Higher lambda values decreased the weights of the weight vector in both models. Higher lambda values affected the L1 model a lot more, as it was more likely to have zero valued weights. The L2 model was more likely to have very small, but still nonzero weights.

1.4 Section2

2 SUPPORT VECTOR MACHINES

Support Vector Machines are supervised learning models that work by finding a dividing hyperplane between the training data while maximizing the gap between the training data and the decision boundary. This is to help the classifier generalize better and makes it more robust to noise. Assuming the data is linearly separable, finding this dividing hyperplane amounts to solving the quadratic program

$$\begin{aligned} \min \quad & \frac{1}{2}||w||^2 \\ \text{s. t.} \quad & y^i(w^T x^i + b) \geq 1, 1 \leq i \leq n \end{aligned} \tag{1}$$

where w is the vector perpendicular to the dividing hyperplane and $\frac{1}{||w||}$ is the size of the margin.

If the data is almost but not completely linearly separable, we can still model the data with an SVM by introducing slack variables when solving for a classifier. We allow the training points to be misclassified by some amount e and the goal is to maximize the margin while minimizing the slack. This formulation is called C-SVM and the separating hyperplane can be found by solving the quadratic program below.

$$\begin{aligned} \min \quad & \frac{1}{2}||w||^2 + C\sum_i e_i \\ \text{s. t.} \quad & y^i(w^T x^i + b) \geq 1 - e_i, 1 \leq i \leq n \\ & e_i \geq 0, 1 \leq i \leq n \end{aligned} \tag{2}$$

2.1 Dual Formulation of C-SVM

This problem as stated above is difficult to implement with kernels functions. Fortunately, we convert the problem from the primal formulation stated above and instead implement its dual formuation.

$$\begin{aligned} \min \quad & \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j [\phi(x_i)^T \phi(x_j)] - \sum_{t=1}^{n} \alpha_t \\ \text{s. t.} \quad & 0 \leq a_t \leq C, \sum_{t=1}^{n} \alpha_t y_t = 0 \end{aligned} \tag{3}$$

To solve this linear program and get the αs, we used the solver in a python library called cvxopt. The solver finds solutions to quadratic programs of the form:

$$\begin{aligned} \min \quad & \frac{1}{2}x^T P x + q^T x \\ \text{s. t.} \quad & Gx \leq h \\ & Ax = b \end{aligned} \tag{4}$$

In our case, our matrices are

$$P = Diag(\overrightarrow{y})XX^T Diag(\overrightarrow{y})$$

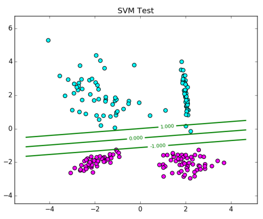
$$q = -1 * \overrightarrow{1}$$

$$G = [I| - I]$$

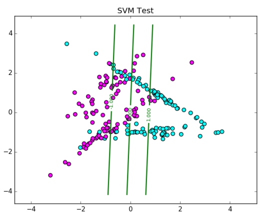
$$h = [C * \overrightarrow{1} \,|\, \overrightarrow{0}]$$

$$A = \overrightarrow{y}$$

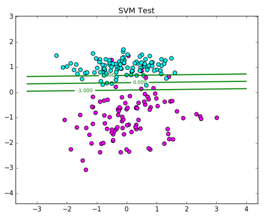
$$b = 0$$



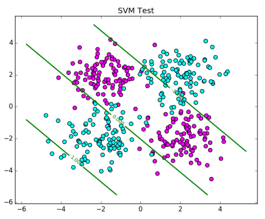
(a) A gull



(b) A gull2



(c) A tiger



(d) A mouse

Figure 1: Pictures of animals

Diag$(\overrightarrow{y})$ is a diagonal matrix with $y^{(i)}$s on the diagonal.
This solver returns $\overrightarrow{x}$ which is our $\overrightarrow{\alpha}$ vector. Testing this implementation,
Do the stupid example!

2.2 2D Dataset Results

2.3 2D Dataset Results with Kernel Functions

3 SUPPORT VECTOR MACHINE WITH PEGASOS

3.1 Section1?

3.2 Section2?

4 HANDWRITTEN DIGIT RECOGNITION WITH MNIST