6.867: Homework 2

1. Logistic regression

In this section, we explore logistic regression with L1 and L2 regularization. We use gradient descent to compare the resulting weight vectors under different regularizers and regularization parameters, and we evaluate the effect of these choices in the context of multiple data sets.

1.1. L2 regularization

We first consider L2 regularization, in which the objective function to minimize is

$$E_{LR}(w, w_0) = \text{NLL}(w, w_0) + \lambda ||w||_2^2$$

where

$$\text{NLL}(w, w_0) = \sum_i \log(1 + \exp(-y^{(i)}(wx^{(i)} + w_0)))$$

and in the case of L2 regularization,

$$||w|| = |w|_2 = \sqrt{w_1^2 + ... + w_n^2}$$

Gradient descent was run with this objective function on the training dataset `data1_train.csv` with $\lambda = 0$. We observed how the weight vector changed as a function of the number of iterations of gradient descent across various initial guesses, step sizes, and convergence criterion. There were three key findings that we were able to make. First, we saw that the ultimate convergence weight was heavily dependent on the initial guess. This actually makes a good deal of sense because of the fact that there are infinitely many ways to perfectly separate a dataset that's linearly separable. The second observation was that w_0 and w_1 always decreased as number of iterations increased whereas w_2 also increased. Finally, we saw that the convergence weight values were all within the same order of magnitude as the initial guess. This makes intuitive sense since we know from class that in the case of linearly separable data, the optimization aims to make the weights as large as possible.

When $\lambda = 1$, we see drastically different behavior in how the weight vector changes throughout the course of the gradient descent. The primary difference is the fact that $w_0, w_1, and w_2$ all converge to much smaller values. For example, when we set the initial guesses to be 100 for all 3 components of the weight vector, the converged weights were two orders of magnitude less than the initial guesses. The reason for this phenomenon is due to the regularization penalty incurred when we set $\lambda = 1$

1.2. L1 regularization

In the case of L1 regularization, the objective function is the same except that $||w_2^2||$ becomes replaced with $||w_1||$, whereby $||w_1||$ is defined by:

$$|w| = |w|_1 = \sum_{i=1}^{n} |w_i|$$

We can evaluate the different regularization techniques under different values of λ in the context of the weight vectors, the decision boundary, and the classification error rate in each of the training data sets. Before we dive into details, we make the obvious observation that when $\lambda = 0$, the choice of regularizer (L_1 vs. L_2) makes no difference on anything since the entire term is just 0.

1.2.1. Weight vector

In terms of the weight vector, we discovered that depending on whether the particular $|w_n|$ was less than or greater to 1 either the L1 regularization resulted in a higher magnitude weight or the L2 regularization resulted in a higher magnitude weight. Specifically, when w_n was less than 1, L2 regularization resulted in a higher magnitude weight, while when w_n was greater than 1, L1 regularization resulted in a higher magnitude weight. This makes intuitive sense because squaring number larger than 1 results in larger numbers while squaring numbers less than 1 result in smaller numbers. We also discovered that a larger λ values resulted in smaller weight magnitudes for both L1 and L2.

1.2.2. Decision boundary

The decision boundary was able to perfectly separate the data for all values of λ as well as for both L1 and L2 regularization when the dataset is linearly separable.

Data	Best regularizer	Best λ	Test performance
1	both	all	1.0
2	both	all	0.805
3	L2	1	0.97
4	L1	1	0.5

Table 1. Optimal regularizer and λ for datasets

Generally, all the decision boundaries looked very similar to one another across values of λ as well as for both L1 and L2 regularization. The one slight difference we noticed is that when $\lambda = 0$ it appeared the algorithm purely tried to correctly classify the most amount of points which resulted in lines that seemed extremely close to a certain distribution while when $\lambda = 1$ the algorithm seemed to be a bit better in taking in to consideration the weights of the various points which resulted in lines that seemed divide the clusters more evenly.

1.2.3. Classification error rate

The classification error rate is always 0 for all values of λ as well as for both L1 and L2 regularization when the dataset is linearly separable. For datasets that were non-linearly separable we found that a larger λ generally resulted in slightly worse performance for the training sets (for example 0.9875 with $\lambda = 0$ and 0.965 with $\lambda = 1$ for training dataset 2 using L1 regularization.

1.3. Optimization

By using the training and validation data sets, we can identify the best regularizer and value for λ for each of the four data sets. These results are presented in Table 1 (above). Graphs of the decision boundaries for the optimal regularizers and λs are shown below.

2. Support Vector Machine

In this section, we explore various versions of the dual form of support vector machines, first with slack variables and then with generalized kernel functions.

2.1. Dual form with slack variables

We here implement a dual form of linear SVMs with slack variables. More specifically, we solve the following optimization problem with respect to α:

$$\max_{\alpha} -\frac{1}{2}|\sum_i \alpha_i y^{(i)} x^{(i)}|^2 + \sum_i \alpha_i$$

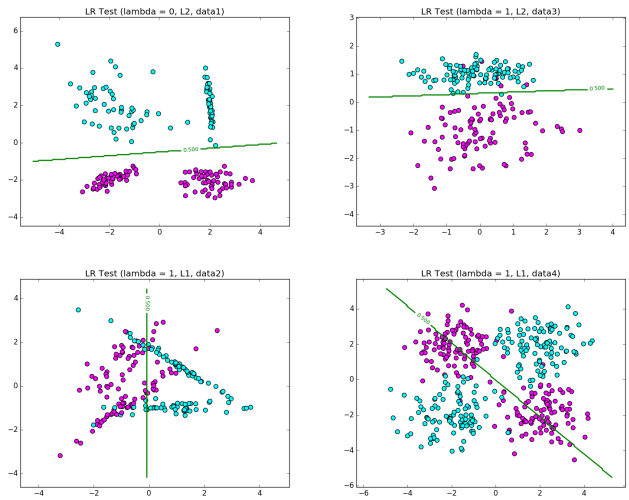

Figure 1. decision boundaries for test data

$$\text{s.t.} \sum_i \alpha_i y^{(i)} = 0$$

$$0 \leq \alpha_i \leq C, 1 \leq i \leq n$$

Written another way, this maximization problem can be framed as a minimization problem:

$$\min_{\alpha} \frac{1}{2} x^T P x + q^T x$$

$$\text{s.t.} Gx \leq h$$

$$Ax = b$$

where $b = 0$ and

$$x = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix}, q = \begin{bmatrix} -1 \\ -1 \\ \vdots \\ -1 \end{bmatrix}, A^T = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{bmatrix}, G = \begin{bmatrix} I \\ -I \end{bmatrix},$$

where I and $-I$ are the identity and negative identity matrix respectively. Furthermore,

$$P = \begin{bmatrix} x_0^2 y_0^2 & x_0 y_0 x_1 y_1 & \dots & x_0 y_0 x_{n-1} y_{n-1} \\ x_1 y_1 x_0 y_0 & x_1^2 y_1^2 & \dots & x_1 y_1 x_{n-1} y_{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n-1} y_{n-1} x_0 y_0 & x_{n-1} y_{n-1} x_1 y_1 & \dots & x_{n-1}^2 y_{n-1}^2 \end{bmatrix}$$

$$h^T = \begin{bmatrix} C & \dots & C & 0 & \dots & 0 \end{bmatrix}$$

In the context of the four-point 2D problem, we seek to solve the following optimization:

$$\min_{\alpha} \frac{1}{2} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}^T \begin{bmatrix} 16 & 24 & 0 & 24 \\ 24 & 36 & 0 & 36 \\ 0 & 0 & 0 & 0 \\ 24 & 36 & 0 & 36 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} + \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}^T \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}$$

Dataset	Training error rate	Validation error rate
1	0.0	0.0
2	0.1775	0.09
3	0.02	0.015
4	0.3	0.305

Table 2. Training and validation error rates

$$\text{s.t.} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} \leq \begin{bmatrix} C \\ C \\ C \\ C \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 3 \\ -1 \\ -2 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = 0$$

Solving for x when $C = 1$, we get that $\begin{bmatrix} \alpha_0 & \alpha_1 & \alpha_2 & \alpha_3 \end{bmatrix}^T = \begin{bmatrix} 0.1875 & 0 & 0.3750 & 0 \end{bmatrix}^T$. This indicates that the first and third samples are support vectors because $0 \leq \alpha_i \leq C$.

Our implementation of the dual form of SVMs with slack variables was run on each of the four 2D datasets provided. With $C = 1$, the decision boundaries and classification error rates were determined. This information is illustrated and summarized in Figure 2 and Table 2, respectively. As can be seen from the classification error rates in training and validation, depending on the values of the trained parameters and the spread of the datasets, the classification error could be greater for the training data (as in dataset 2), for the validation data (as in dataset 4), or equal (as in dataset 1).

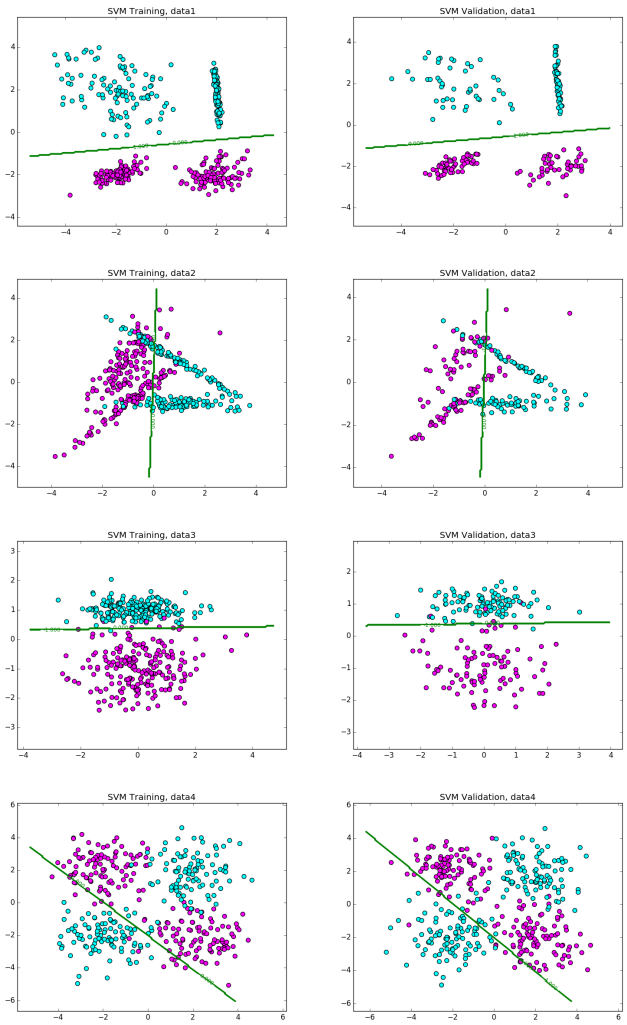

Figure 2. Training and validation decision boundary

It is interesting to note that due to numerical rounding and errors, the α values obtained were not exactly equal to 0 or C, but instead usually varied by a small threshold. In our case, we chose a threshold of 0.00001 in order to filter out α values extremely close to 0 or C and select for "true" support vectors. Varying this threshold however could change the classification of data as support vectors or not, and this could ultimately affect the accuracy and error rate of our classifier.

2.2. Dual form with kernels

We here extend our dual form SVM to take kernel functions and kernel matrices as input. This has important implications for a number of reasons including performance and minimizing the cost of operations, as well as space and storage. We here implemented the dual form of SVMs with kernels for two kernel functions: (1) linear kernels, and (2) Gaussian radial basis function (RBF) kernels, while varying the value of C.

As the value of C increased, on a conceptual level, the margin error should decrease. This is because

As the value C was increased, the number of support vectors generally did not show any noticeable or significant trends with a linear kernel. For all the data sets for varying values of C, the number of support vectors stayed roughly constant at around 3 or 4. However,

	$C = 0.01$	$C = 0.1$	$C = 1$	$C = 10$	$C = 100$
1	0	29	45	49	48
2	0	21	32	32	36
3	2	16	28	25	20
4	0	35	79	78	75

Table 3. Number of support vectors by C value

λ	2^1	2^{-1}	2^{-3}	2^{-5}	2^{-7}	2^{-10}
margin	∞	1.134	0.820	0.642	0.547	0.389

Table 4. SVM margin as a function of λ

for the Gaussian RBF kernel, the number of support vectors generally increased as the value of C increased until a certain point (generally around $C = 1$), after which the number of support vectors plateaued or even decreased. This data for the Gaussian RBF is summarized in Table 4.

3. Pegasos

In this section, we use the Pegasos algorithm to solve the following formulation of Soft-SVM:

$$min\frac{\lambda}{2}||w||^2 + \frac{1}{n}\sum_{i=1}^{N} max\{0, 1 - y_i(w^T x_i)\}$$

We choose to use Pegasos because it combines together multiple good ideas such as hinge loss, L2 regularization, stochastic (sub)-gradient descent, and a decaying step-size. The following table shows the impact of λ (regularization constant) on the margin. It indeed matches our understanding of the objective function because lower regularization means larger slack variables which lead to smaller margins. The reason for this is because $1 - \epsilon_i$ governs the width of the margin.

4. Handwritten digit recognition