

6.867 Homework 2

Logistic Regression

1.1 Unregularized Logistic Regression

We started by implementing logistic regression with stochastic gradient descent and plotting the norm of the weight vector over time for both the unregularized ($\lambda = 0$) and L2 regularized ($\lambda = 1$) objective.

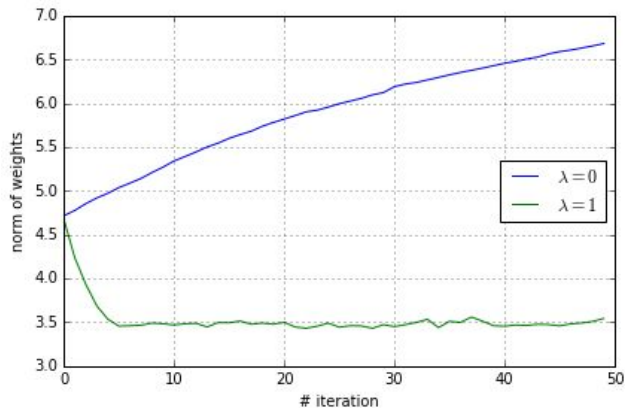


Figure 1. TODO: SOME DESCRIPTION.

As shown in Figure 1, the unregularized weight vector grows indefinitely while the regularized weight vector converges. This can be explained by the fact that larger magnitude weight vectors will result in more confident predictions, moving the prediction closer to 1.0 or 0.0 for positive and negative samples respectively.

Note, however, that a more confident predictor is not necessarily better. In fact, in many cases such as medical applications, a model that is both confident and incorrect is worse than a model that is uncertain on some inputs.

1.2 Hyperparameters for Logistic Regression

We examined each of the four datasets with various regularizers and λ values using the sklearn implementation of logistic regression. In our code, we explicitly adjust the intercept scaling so as not to regularize the bias term.

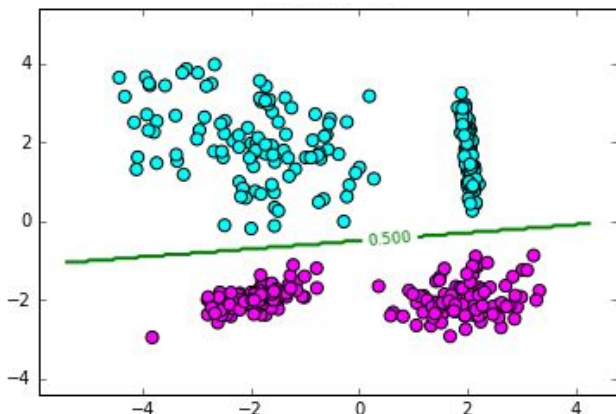


Figure 2. TODO: SOME DESCRIPTION.

The first dataset, shown in figure 2, is linearly separable and every combination of regularizer and λ between 0.0 and 20.0 produced a decision boundary which perfectly separated the two classes on the training, validation, and test set. As the λ constant increases, the norm of the weight vector shrinks.

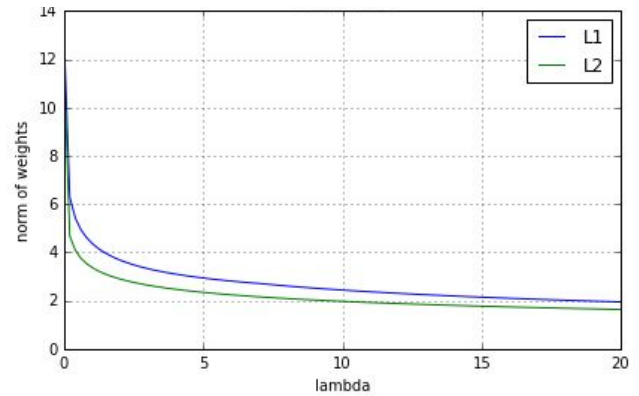


Figure 3. TODO: SOME DESCRIPTION.

On this particular dataset, we can completely ignore the first dimension of the input vector and still separate the two classes. This is reflected by the fact that with a large regularization constant, the L1 regularizer produces a sparse weight vector while the L2 regularizer produces a weight vector where the first dimension is near-zero.

The second dataset, however, is not linearly separable and there is no linear decision boundary capable of perfectly discriminating between positive and negative samples.

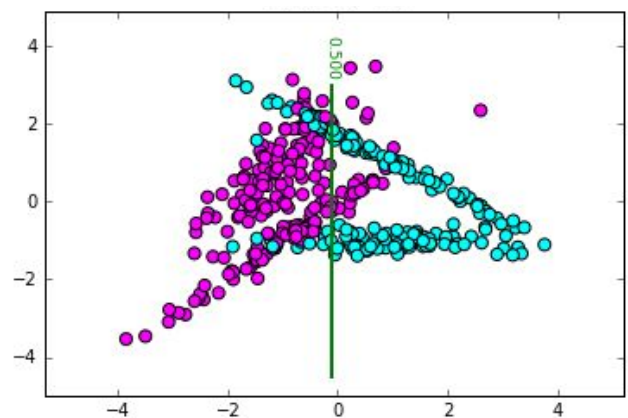


Figure 4. TODO: SOME DESCRIPTION.

As we increased λ , we observed that the norm of the weights behaved exactly like it did for the first dataset. On the other hand, the training and validation accuracy changed significantly as we adjusted λ and the regularizer.

A subset of the regularizer and λ values we examined is shown below in figure 5, where the bolded entry indicates a pair of values which minimizes the validation error.

L	λ	Train Error	Validation Error	Test Error
-	0	0.1700	0.1750	0.1950
L1	1	0.1675	0.1750	0.1950
L2	1	0.1675	0.1750	0.1950
L1	2	0.1700	0.1800	0.1900
L2	2	0.1650	0.1750	0.1950

Figure 5. TODO: SOME DESCRIPTION

The third dataset was nearly linearly separable. We determined that L2 regularization with a λ of 0.5 produced the smallest validation error.

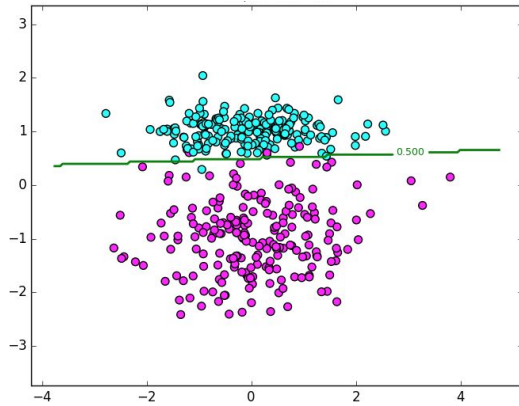


Figure 6. TODO: SOME DESCRIPTION.

L	λ	Train Error	Validation Error	Test Error
-	0	0.0125	0.035	0.050
L1	0.5	0.0200	0.030	0.045
L2	0.5	0.0125	0.025	0.400
L1	1	0.0175	0.035	0.045
L2	1	0.0225	0.030	0.030

Figure 7. TODO: SOME DESCRIPTION.

The fourth dataset reminiscent of the XOR problem and is not linearly separable. Any linear decision boundary is guaranteed to be wrong on half (or more) of the inputs.

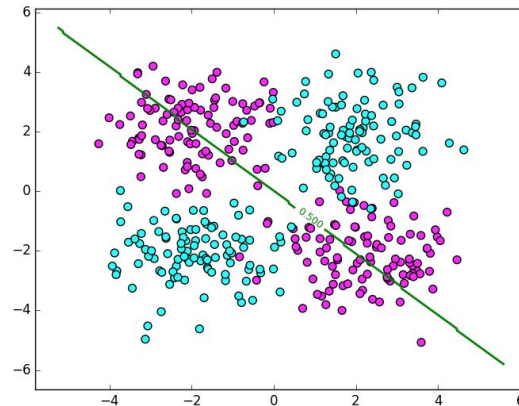


Figure 8. TODO: SOME DESCRIPTION.

L	λ	Train Error	Validation Error	Test Error
-	0	0.4850	0.5075	0.5025
L1	1	0.4825	0.5025	0.5000
L2	1	0.4850	0.5075	0.5025
L1	3	0.4825	0.5050	0.5000
L2	3	0.4850	0.5075	0.5025

Figure 9. TODO: SOME DESCRIPTION.

Support Vector Machine

We tested our support vector machine on a toy dataset consisting of (2, 2), (2, 3), (0, -1), and (-3, -2) with two positive and two negative examples. Our implementation produced the below objective function and constraints.

Objective Function									
$\frac{1}{2}$	$[\alpha_0]$	[8.00	0.10	2.00	0.10]	$[\alpha_0]$	+	$[-1]$	$[\alpha_0]$
	$[\alpha_1]$	[0.10	0.13	3.00	0.12]	$[\alpha_1]$		$[-1]$	$[\alpha_1]$
	$[\alpha_2]$	[2.00	3.00	1.00	2.00]	$[\alpha_2]$		$[-1]$	$[\alpha_2]$
	$[\alpha_3]$	[0.10	0.12	2.00	0.13]	$[\alpha_3]$		$[-1]$	$[\alpha_3]$

Constraints				
[-1.00 -0.00 -0.00 -0.00]	$[\alpha_0]$ $[\alpha_1]$ $[\alpha_2]$ $[\alpha_3]$	<	[0.00]	
[-0.00 -1.00 -0.00 -0.00]			[0.00]	
[-0.00 -0.00 -1.00 -0.00]			[0.00]	
[-0.00 -0.00 -0.00 -1.00]			[0.00]	
[1.00 0.00 0.00 0.00]			[C]	
[0.00 1.00 0.00 0.00]			[C]	
[0.00 0.00 1.00 0.00]			[C]	
[0.00 0.00 0.00 1.00]			[C]	
[1.00]	$[\alpha_0]$	=	[0.00]	
[1.00]	$[\alpha_1]$			
[-1.00]	$[\alpha_2]$			
[-1.00]	$[\alpha_3]$			

By solving this quadratic programming problem, we get the following decision boundary which only depends on two support vectors, (2,2) and (0, -1).

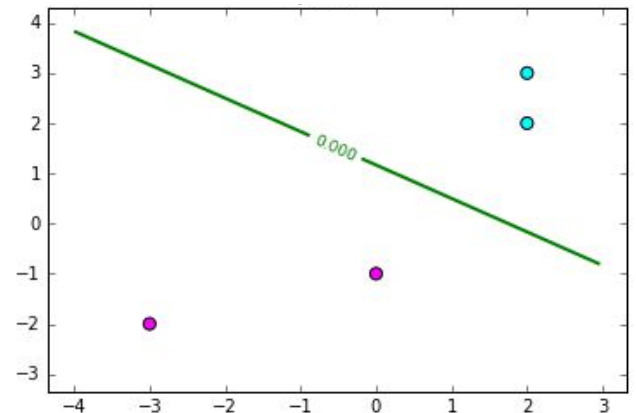


Figure 10. TODO: SOME DESCRIPTION.

Dataset 1

For the first dataset, we noted the various C values in the set $\{0.01, 0.1, 1, 10, 100\}$ did not have a significant impact on the linear decision boundary; in fact, just like when applying logistic regression, every C value produced a boundary which perfectly separated the two classes.

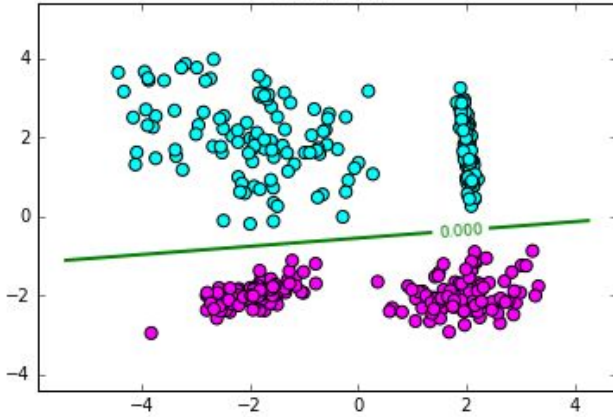


Figure 11. TODO: SOME DESCRIPTION.

When using the radial basis function kernel, we were able to achieve a reasonable decision boundary for all values of C in the above set. By increasing the bandwidth, we were able to produce decision boundaries that ranged from a nearly straight line (on the local scale) to a circular boundary which completely encircled one of the classes.

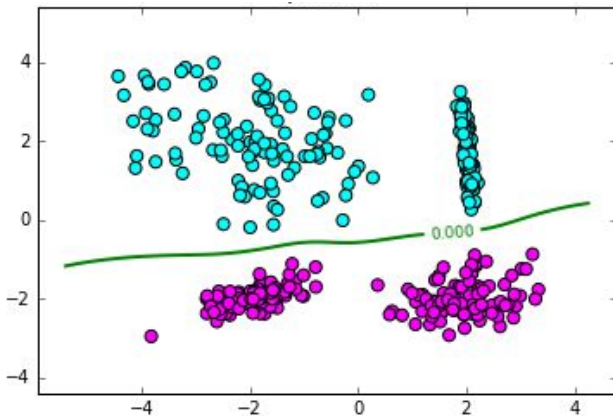


Figure 12. TODO: SOME DESCRIPTION.

Due to the linearly separable nature of the first dataset, we observed that as we increased C , the geometric margin and number of support vectors both decreased. This can be explained by the fact that the C value controls the tradeoff between a wider margin and the number of mistakes. By increasing C , the support vector machine penalizes mistakes more heavily.

Dataset 2

The second dataset helped provide a better understanding of the relationship between the C value and the decision

boundary. We started by establishing a baseline using the linear support vector machine, achieving error rates of 0.165 on the validation set and 0.19 on the test set.

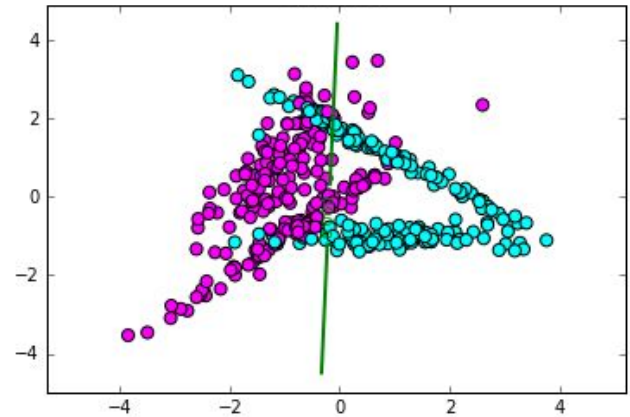


Figure 13. TODO: SOME DESCRIPTION.

For small values of C such as $C = 0.01$, we observed that the decision boundary was nearly linear, achieving similar accuracy rates. As we increased the C value to $C = 10.0$, we observed a decision boundary with validation error 0.099 and test error 0.045, significantly outperforming the linear support vector machine.

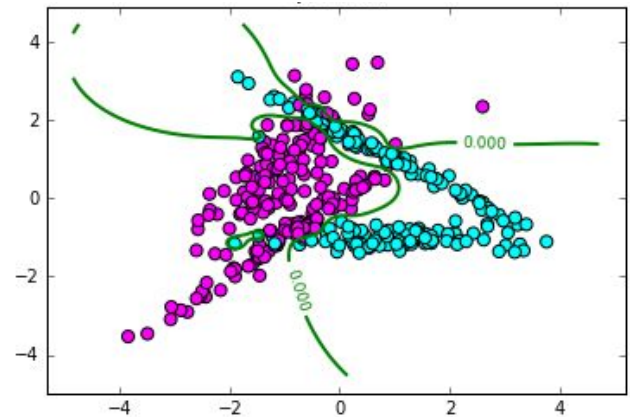
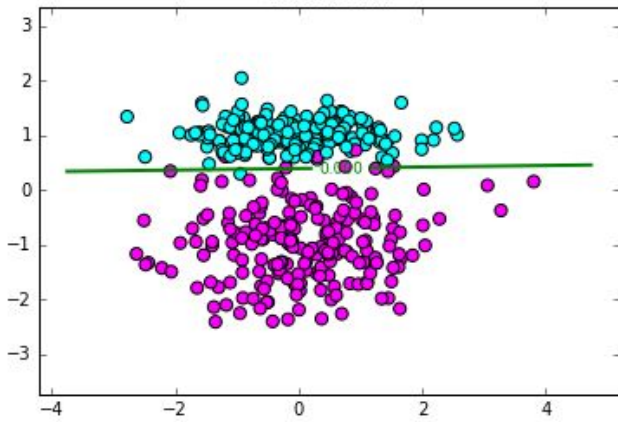


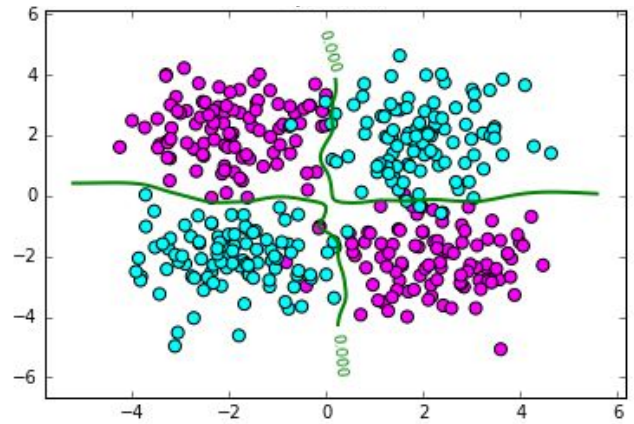
Figure 14. TODO: SOME DESCRIPTION.

Although our goal for this dataset was not to find the optimal decision boundary, we noted that choosing C values by maximizing the geometric margin on the training set is not as effective as using the validation set.

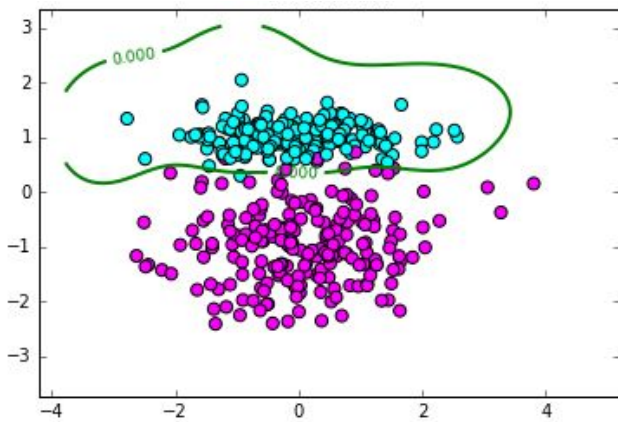
Dataset 3



('quadSVM val', array([0.035]))
('quadSVM test', array([0.05]))

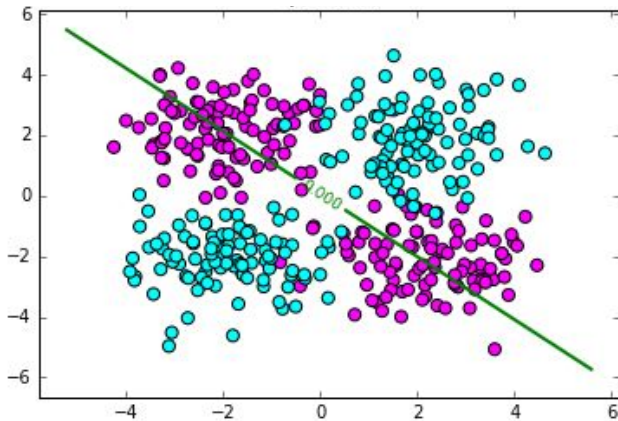


('sklearnSVM val', array([0.05]))
('sklearnSVM test', array([0.045]))



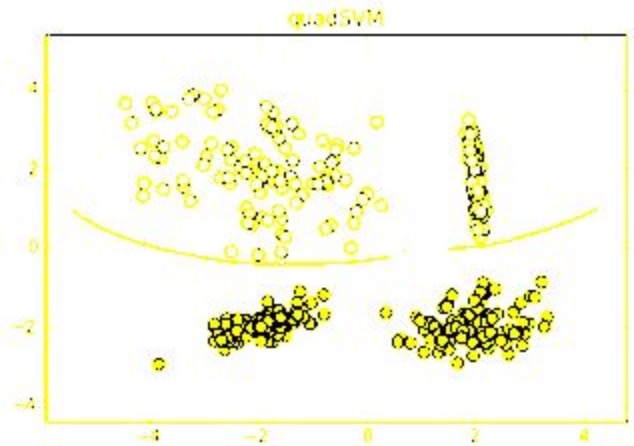
('quadSVM val', array([0.06]))
('quadSVM test', array([0.03]))

Dataset 4



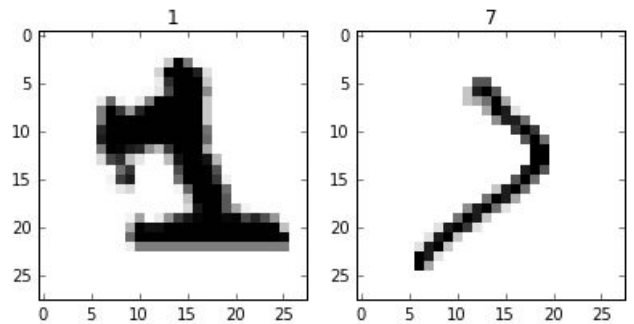
('quadSVM val', array([0.515]))
('quadSVM test', array([0.51]))

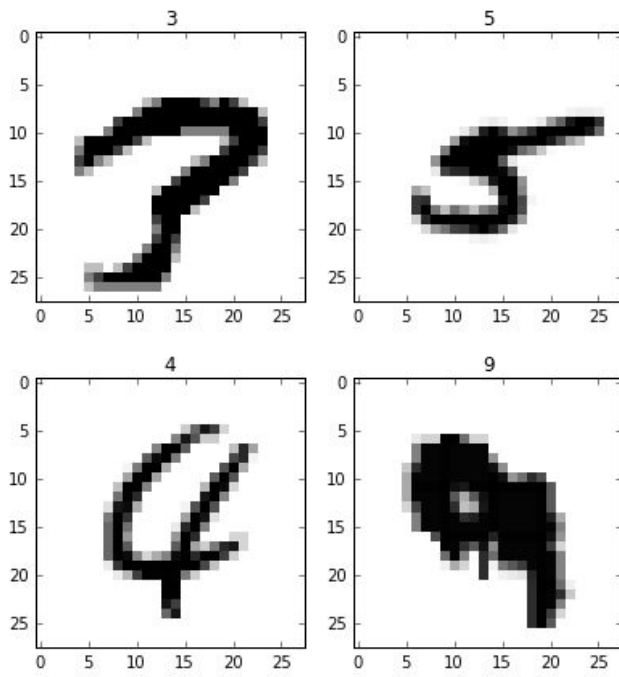
Pegasos



MNIST

Examples that are misclassified by the linear support vector machine.





With a large enough bandwidth, the radial basis function kernel outperformed the linear support vector machine.