1. Logistic Regression

We used our gradient descent code from Homework 1 to solve this problem. In particular we supplied the optimizer with analytic gradients. If we define $\tilde{x}$ to be $[1, x]^T$ then the gradient of the regularized NLL objective can be written as

$$\frac{\partial E_{LR}(w)}{\partial w} = \sum_i -y^{(i)}\tilde{x}^{(i)} \frac{\exp(-y^{(i)}(\tilde{x}^{(i)}w + w_0))}{1 + \exp(-y^{(i)}(\tilde{x}^{(i)}w + w_0))} \tag{1}$$

As was shown in lecture, the objective function for logistic regression is convex. Thus gradient descent should, with the appropriate step size and convergence threshold, converge to the global minimum. For almost all the subsequent optimizations we used a step size of $\eta = 1e-4$ and an initial guess of $w_{initial} = [0, 0, 0]$. A few optimizations (in particular those for very large λ) required a smaller step size, approximately $\eta = 1e-6$.

1.1. $\lambda = 0$

First we present the results for $\lambda = 0$ in Table 1.1

dataset	misslassification rate
stdev1 train	0
stdev1 val	0
stdev2 train	0.0925
stdev2 val	0.08
stdev4 train	0.26
stdev4 val	0.25
nonsep train	0.485
nonsep val	0.5025

Table 1. Results on different dataset for $\lambda = 0$

Define $\tilde{w} = [w_0, w]$ to be the full weight vector. For the datasets stdev1, stdev2, stdev4 the gradient descent method converges to solution. The stdev1 dataset is linearly separable. This presents a problem for the optimizer since given a weight $\tilde{w}$ that separates the data, we can arbitrarily increase the value of the likelihood by considering new weights $c\tilde{w}$ where $c > 1$ is a constant. In practice our gradient descent method always terminates because either the termination criterion is reached (e.g. difference in function values on successive steps is smaller than a threshold) or we reach the maximum number of function calls. As an illustration if we run the optimizer with termination criterion $\epsilon = 1e-4$ we get $||\tilde{w}|| = 7.9$. If we increase the tolerance to $\epsilon = 1e-10$ then the norm of the solution weights increases to $||\tilde{w}|| = 14.7$. This illustrates the weights "running off to infinty". Since the training data is linearly separable the classification error rate is zero, see Figure 1.1. Also as can be seen from Figure 1.1 the classification error rate on the validation data is also zero.

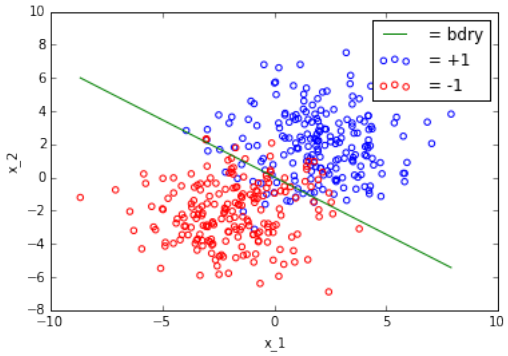

Figure 1. $\lambda = 0$, stdev1 train dataset

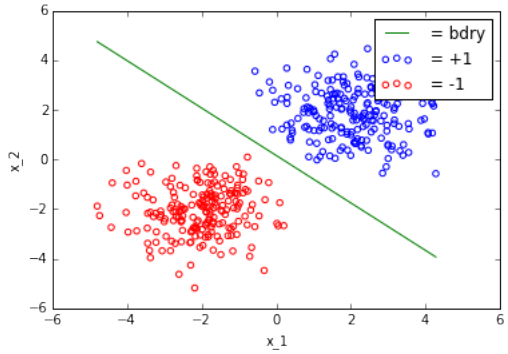

Figure 2. $\lambda = 0$, stdev1 validation dataset

The other datasets (other than stdev1) are not linearly separable. Hence we don't have the "weights running off to infinity" problem that we had with stdev1. Since the stdev of the points in these datasets is larger (making them nonseparable) we also expect the classfication error rate to be larger as well. The results for the stdev4 dataset are shown in Figures 1.1 and 1.1. The classifier still exihibits relatively good performance in separating the data. Even though the data is noisy, the decision boundary is very similar to the one in the less noisy case, the stdev1 data.

In contrast the performance on the nonseparable dataset is quite poor. This makese sense since the data is not linearly separable.

Interestingly, even with the regularizer λ set to 0 the performance across the training and validation sets for a given dataset (e.g. stdev2) are comparable. This can be seen by the similar missclassification rates across corresponding training and validation sets in Table 1.1. This means that our logistic regression classifier is generalizing fairly well.

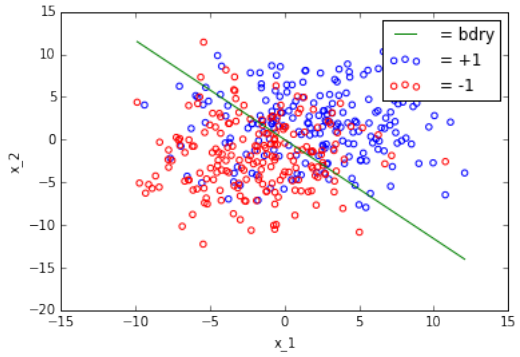

Figure 3. $\lambda = 0$, stdev4 train dataset

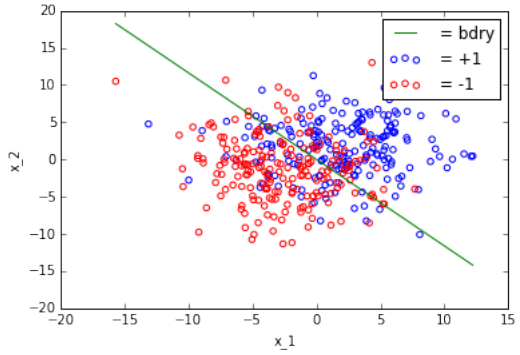

Figure 4. $\lambda = 0$, stdev4 validation dataset

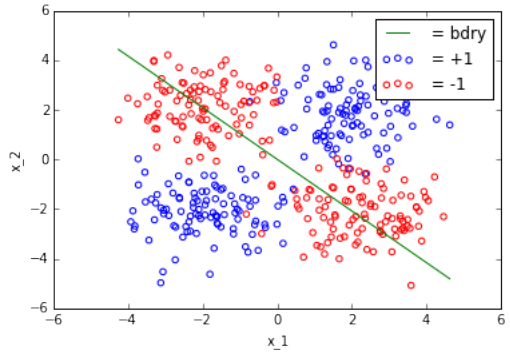

Figure 5. $\lambda = 0$, nonsep train dataset

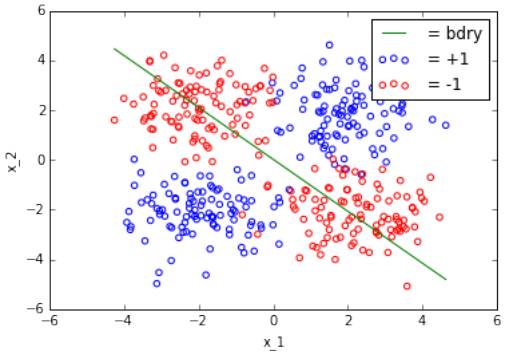

Figure 6. $\lambda = 10$, nonsep train dataset

1.2. $\lambda > 0$

Now we analyze the results for different values of the regularizer λ. Figures 1.2, 1.2, 1.2 shows the result of running logistic regression on the nonsep dataset for different values of λ. The thing to note is that for different values of λ the actual decision boundary, plotted in green, looks very similar. In fact what seems to be happening is that as λ increases the optimal weight $w(\lambda)$ are just scaled down to zero. In fact if we define $\overline{w}(\lambda) = \frac{w(\lambda)}{||w(\lambda)||}$ then $\overline{w}(\lambda)$ is very similar across different values of λ. Thus the regularizer λ is forcing the weights toward zero, i.e. $||w(\lambda)||$ is decreasing in λ, but the scaled weights $\overline{w}(\lambda)$ are staying essentially the same. This explains why the decision boundaries look the same across the three plots in Figure since $w_0 + w_1 x_1 + w_2 x_2 = 0$ and $cw_0 + cw_1 x_1 + cw_2 x_2 = 0$ (where $c > 0$ is a constant) define the same hyperplane. Thus in this case the regularizer λ just serves to drive the weights toward zero, but does so in a way that doesn't significantly affect the classification boundary. Table 1.2 shows how $||\tilde{w}(\lambda)|| \to 0$ as we increase λ.

include reference

λ	$\|\|\tilde{w}(\lambda)\|\|$
0	0.034
10	0.0329
100	0.0242
1000	0.00067

Table 2. Norm of $\tilde{w}(\lambda)$ for different values of λ on nonsep train dataset.

For the nonseparable data the results look fairly identical. Not sure if we should expect something different . . .

2. Support Vector Machine (SVM)

We implemented the dual form of Soft-SVMs. Our implementation is general and capable of handling linear, Gaussian, or other user-specified kernels or Gram matri-

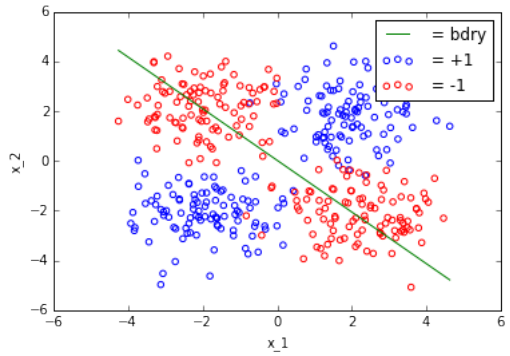

Figure 7. $\lambda = 100$, nonsep train dataset

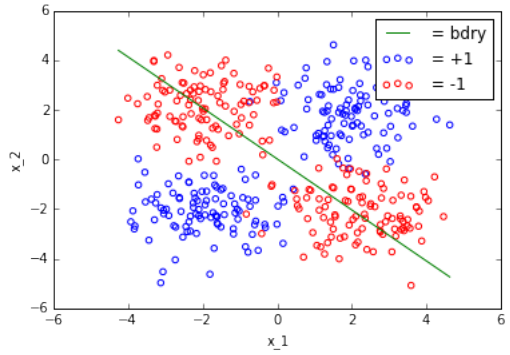

Figure 8. $\lambda = 1000$, nonsep train dataset

ces, and allows the user to additionally specify C (the variable that adjusts the relative penalty on the slack variables) and any special kernel-dependent variables (including the bandwidth for the Gaussian kernel).

A description of the implementation is as follows. Given a kernel function, $k(\cdot, \cdot)$, a set of N data points $\{x^{(i)}, y^{(i)}\}$ where each each $\mathbf{x} \in \mathbb{R}^d$ is paired with a a target label $t \in \{-1, 1\}$, and C as mentioned above, we can solve the dual form Soft-SVM optimization problem:

$$\begin{aligned} \underset{\alpha \in \mathbb{R}^N}{\text{maximize}} \quad & \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m) \\ \text{subject to} \quad & 0 \leq a_n \leq C \\ & \sum_{n=1}^{N} a_n t_n = 0 \end{aligned} \tag{2}$$

The decision variables in the optimization are the elements a_i of $\mathbf{a} \in \mathbb{R}^N$. As the objective is quadratic and the constraints are linear inequalities and a linear equality, we solve the problem as a QP (Quadratic Program), using CVXOPT, a free software package for convex optimization. The QP interface to CVXOPT accepts problems in the form:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} a^T P a + q^T a \\ \text{subject to} \quad & Ga \leq h \\ & Aa = b \end{aligned} \tag{3}$$

A correct formulation of our Soft-SVM for the CVXOPT interface is accordingly: $P = \mathbf{t}\mathbf{t}^T K$ (where K is the Gram matrix for the kernel), q is a vector of length N with each element -1, G is a vertically stacked block matrix $[G_0; G_C]$ where G_0 and G_C are both $N \times N$ diagonal matrices with, respectively, -1 and 1 down the diagonals, H is a vector of length $2N$ where the first N elements are zero and the last N elements are each C, A is $\mathbf{t}^T$, and $b = 0$.

The additional important details of the implementation are in transforming the optimal vector $\mathbf{a}$ into a form that nicely represents the decision boundary. The first step is to compute b as follows:

$$b = \frac{1}{N_{\mathcal{S}}} \sum_{n \in \mathcal{S}} \left(t_n - \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) \right) \tag{4}$$

where $\mathcal{S}$ denotes the set of indices of the support vectors, and $N_{\mathcal{S}}$ is the number of support vectors. The support vector indices can be determined by finding the indices of $\mathbf{a}$ that are non-zero. With b, the predictor function is then

What should I write down for the constraints and objective for the 2D problem given? Should I actually show each $P, q, G, h, A,$

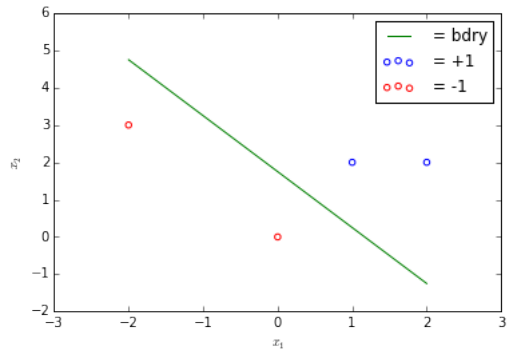

Figure 9. Soft-SVM solution for decision boundary of simple 2D dataset with $C = 1$

simply:

$$y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b \tag{5}$$

and $y(\cdot) = 0$ is the decision boundary. It is also useful to compute the weight vector $\mathbf{w} \in \mathbb{R}^d$, which for the linear kernel is $\mathbf{w} = \sum_{n=1}^{N} a_n t_n \mathbf{x_n}$ (not including w_0) and $w_0 = b$.

We now examine the performance of the SVM classifiers on the same datasets analyzed in Part 1 for logistic regression. The results of the misclassification rates, with a linear kernel $k(x, x') = x^T x'$ are shown in Table 2.

Dataset	Misslassification rate
stdev1 train	0
stdev1 val	0
stdev2 train	0.0925
stdev2 val	0.085
stdev4 train	0.265
stdev4 val	0.2375
nonsep train	0.49
nonsep val	0.515

Table 3. SVM misclassification rates on different datasets for C=0.

Comparing the results of Tables 1.1 and 2, we see that the performance of the logistic regression and SVM classifiers are almost identical on these 2D datasets. The plots of the decision boundary together with the data also closely match those from Part 1. **What else can be said? It says to report/explain the decision boundary and classification error rate??**

The Gaussian kernel offers an interesting alternative to the linear kernel for classification. The formulation of the SVM is exactly the same as the linear kernel except the

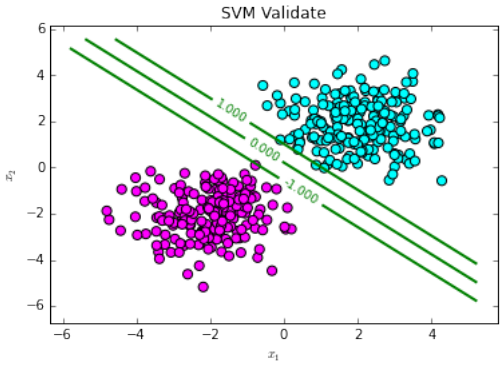

Figure 10. Soft-SVM solution for decision boundary of "stdev1" validation dataset with $C = 1$ and a linear kernel

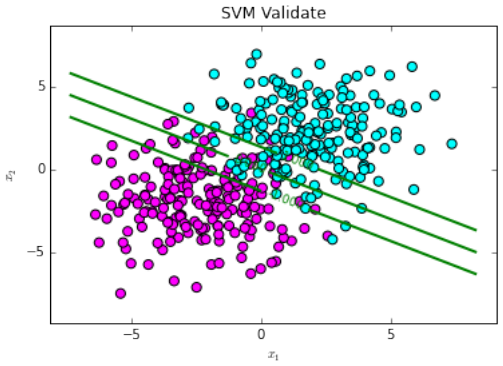

Figure 11. Soft-SVM solution for decision boundary of "stdev2" validation dataset with $C = 1$ and a linear kernel

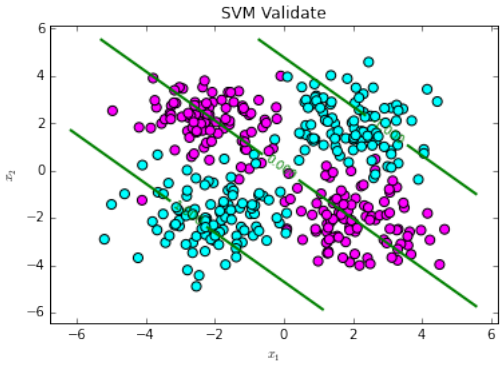

Figure 12. Soft-SVM solution for decision boundary of "nonsep" validation dataset with $C = 1$ and a linear kernel

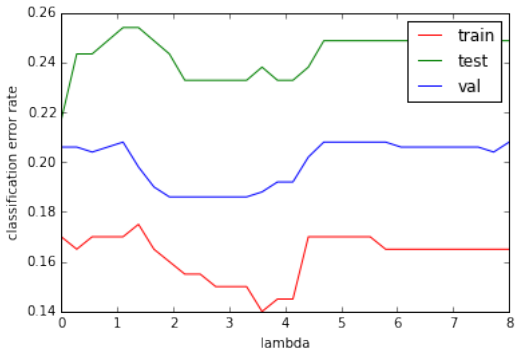

Figure 13. Results of logistic regression classifier on Titanic dataset for different value of λ.

kernel is now:

$$k(x, x') = \exp\big(-\frac{1}{2\sigma^2}||x - x'||_2^2\big) \tag{6}$$

where σ is termed the bandwidth. We can adjust both the C (relative penalty on the slack variables) and bandwidth, σ, in order to tune our classifier to the data.

The values of the geometric margin, $\frac{1}{||\mathbf{w}||}$, for Soft-SVMs with the Gaussian kernel are shown for the different datasets and for $C = \{0.01, 0.1, 1, 10, 100\}$.

Dataset	$C = 0.01$	0.1	1	10	100
stdev1	0.09	0.0875	0.0825	0.075	0.07
stdev2	0	0	0	0	0
stdev4	0	0	0	0	0
nonsep	0	0	0	0	0

Table 4. Training data misclassification rates for various values of C, for each of the provided 2D datasets.

Dataset	$C = 0.01$	0.1	1	10	100
stdev1	0.07	0.07	0.0775	0.0875	0.0925
stdev2	0	0	0	0	0
stdev4	0	0	0	0	0
nonsep	0	0	0	0	0

Table 5. Validation data misclassification rates for various values of C, for each of the provided 2D datasets.

3. Titanic

1. We perform "interval" type feature rescaling. Namely we rescale all the features so that they lie in $[0, 1]$. The performance is shown in Figure 1. Performing model selection leads us to choose $\lambda = 2.5$.

2.

3. The classifiers (resulting from model selection in parts 1 and 2) for logistic regression (LR) and SVM are detailed in Table 6. For LR we see that the most important features are which class you are in, your sex, and your age. Sex is by far the most important of those.

Meaning	LR	SVM
w_0	-0.854	
1st Class	0.35	
2nd Class	0.33	
3rd Class	-0.69	
Sex	1.65	
Age	-0.267	
# Siblings/Spouses	-0.004	
# Parents/Children	0.23	
Fare	0.16	
Southampton	-0.255	
Cherbourg	0.23	
Queenstown	0.026	

Table 6. Classification weights