Massachusetts Institute of Technology
6.867 Machine Learning, Fall 2006

Problem Set 1 Solutions

Section B

1. (a) The functions are on the course website hw1/solutions. The major cause of variation among the solutions was the choice of the order in which points were picked for the update step. This was also the cause of a few subtle mistakes as well. The most straightforward strategy for picking points (in the order of their occurrence in the dataset), produces $\theta = 2.3799$ radians or $\theta = 136.3582°$.

   The number of updates, as per the above strategy, was 10. The analysis described in the lectures indicated that the number of perceptron updates (mistakes) was necessarily bounded by $\left(R/\gamma^*_{geom}\right)^2$ where γ^*_{geom} is the maximum geometric margin for this problem. A small number of updates therefore suggests that γ^*_{geom} is reasonably large in comparison to the radius R of the enclosing sphere. In other words, the two class populations appear to be well-separated.

   (b) The most straightforward strategy for picking points (in the order of their occurrence in the dataset), produces $\theta = 2.3566$ radians or $\theta = 136.3582°$. We will accept answers in the range $(2.3552, 2.3592)$ radians or $(134.9454°, 135.1730°)$. The latter ranges corresponds to decision boundaries going through points at the margins of the max-margin classifier (see Prob 2).

   The number of updates, as per the above strategy, was 152. The bounding sphere is about the same for these points, however. We would therefore expect that the geometric margin γ^*_{geom} is larger for this problem.

   (c) We can also evaluate γ_{geom}, the margin actually achieved by the perceptron algorithm. This is not the maximum margin but may nevertheless be indicative of how hard the problem is. Given X and θ, γ_{geom} can be calculated in MATLAB as follows:

```
gamma_geom = min(abs(X*theta / norm(theta)))
```

   We get $\gamma^a_{geom} = 1.6405$ and $\gamma^b_{geom} = 0.0493$, again with some variation due to the order in which one selects the training examples. These margins appear to be consistent with our analysis, at least in terms of their relative magnitude. The bound on the number of updates holds for any margin, maximum or not, but gives the tightest guarantee with the maximum margin.

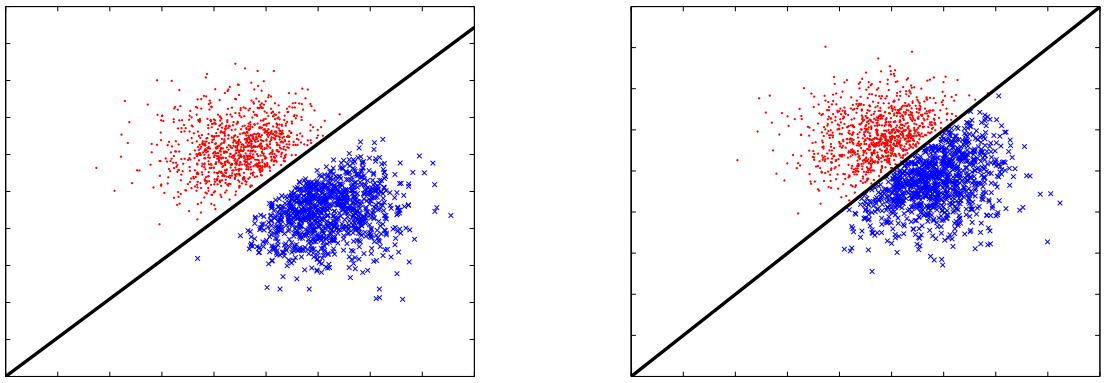
   (d) Given X, R can be calculated in MATLAB as

```
R = max(sqrt(sum(X.^2,2)))
```

   Then, $R_a = 200.561$ and $R_b = 196.3826$. Using these, and ${\gamma^*}^a_{geom} = 5.5731$ and ${\gamma^*}^b_{geom} = 0.3267$ evaluated below, the theoretical bounds on the number of perceptron updates for the two problems are

$$k_a \leq \left(R_a/{\gamma^*}^a_{geom}\right)^2 \approx 1295 \quad (1)$$

$$k_a \leq \left(R_b/{\gamma^*}^b_{geom}\right)^2 \approx 361333 \quad (2)$$



(a) Dataset A

(b) Dataset B

Figure 1: The decision boundary $\theta^T x$ for the two datasets is show in black. The two classes are indicated by the colors red and blue, respectively.

The bounds are not very tight. This is in part because they must hold for any order in which you chose to go through the training examples (and in part because they are simple theoretical bounds omitting a lot of specifics of the two problems).

(e) The plots are shown in Fig 1.

2. (a) The functions are on the course website. The distance must not be used, since it is magnitude-dependent. Measuring the angle between them is the most appropriate way. The angle will vary, depending upon the ordering strategy used in Prob 1 and, consequently, the θ found in Prob 1. However, the answer corresponding to the strategy where points are picked in their order in the input, the answer for data set 'A' was 0.0221 radians or 1.2672°. We will accept answers < 0.0351 radians (or 2.011°). Similarly, for data set 'B', the most straigthforward strategy produced the difference as 0.0019 radians or 0.1114°. We will accept answers < 0.002 radians (or 0.1146°). These latter limits are based on the decision boundaries going through points at the margins of the max-margin classifier.

   (b) $\gamma^a_{geom} = 5.5731$ and $\gamma^b_{geom} = 0.3267$. These are the maximum margins achievable with any linear classifier through origin.

3. (a) The correct invocation of SVM$_{light}$ in training is had by setting C to infinity:

```
% svm_learn -c +inf train-01-images.svm
Scanning examples...done
Reading examples into memory... <snip> ..OK. (12665 examples read)
Optimizing... <snip> .done. (687 iterations)
Optimization finished (0 misclassified, maxdiff=0.00100).
Runtime in cpu-seconds: 1.30
Number of SV: 84 (including 0 at upper bound)
L1 loss: loss=0.00000
Norm of weight vector: |w|=0.00924
Norm of longest example vector: |x|=4380.65657
Estimated VCdim of classifier: VCdim<=890.06377
Computing XiAlpha-estimates...done
Runtime for XiAlpha-estimates in cpu-seconds: 0.02
```

```
XiAlpha-estimate of the error: error<=0.65% (rho=1.00,depth=0)
XiAlpha-estimate of the recall: recall=>99.48% (rho=1.00,depth=0)
XiAlpha-estimate of the precision: precision=>99.30% (rho=1.00,depth=0)
Number of kernel evaluations: 164632
Writing model file...done
```

Since this is a hard-margin SVM, if a solution exists then the training error will be 0 (and you get away without submitting an image). With the new model file, we can evaluate our model on the test set:

```
$ svm_classify test-01-images.svm svm_model
Reading model...OK. (84 support vectors read)
Classifying test examples..<snip>..done
Runtime (without IO) in cpu-seconds: 0.00
Accuracy on test set: 99.91% (2113 correct, 2 incorrect, 2115 total)
Precision/recall on test set: 99.91%/99.91%
```

Now, for interest's sake, we'd like to extract the two test images that were misclassified. This is how it's done using unix power tools:

```
$ cat test-01-images.svm | awk '{ print $1 }' | paste - svm_predictions \
  | nl | awk '{ if ($2 * $3 < 0) { print $1 } }'
1665
2032
```

It can also be done somewhat less easily by using a spreadsheet utility. Here are the images:


test-001665.png


test-002032.png

We also accepted training with the default value of C:

```
$ svm_learn train-01-images.svm
Scanning examples...done
Reading examples into memory...<snip>..OK. (12665 examples read)
Setting default regularization parameter C=0.0000
Optimizing...<snip>...done. (163 iterations)
Optimization finished (9 misclassified, maxdiff=0.00089).
Runtime in cpu-seconds: 0.60
Number of SV: 141 (including 77 at upper bound)
L1 loss: loss=35.80801
Norm of weight vector: |w|=0.00365
Norm of longest example vector: |x|=4380.65657
Estimated VCdim of classifier: VCdim<=142.17389
Computing XiAlpha-estimates...done
Runtime for XiAlpha-estimates in cpu-seconds: 0.01
XiAlpha-estimate of the error: error<=0.92% (rho=1.00,depth=0)
XiAlpha-estimate of the recall: recall=>99.12% (rho=1.00,depth=0)
XiAlpha-estimate of the precision: precision=>99.15% (rho=1.00,depth=0)
Number of kernel evaluations: 135983
Writing model file...done
```

In this case, nine training examples are misclassified. Again, there are many ways of extracting the misclassified images. Here is how it is done with unix power tools:

```
$ svm_classify train-01-images.svm svm_model
Reading model...OK. (141 support vectors read)
Classifying test examples..<snip>..done
Runtime (without IO) in cpu-seconds: 0.00
Accuracy on test set: 99.93% (12656 correct, 9 incorrect, 12665 total)
Precision/recall on test set: 99.93%/99.94%
$ cat train-01-images.svm | awk '{ print $1 }' | paste - svm_predictions \
  | nl | awk '{ if ($2 * $3 < 0) { print $1 } }'
315
2824
4496
4612
6031
6627
8185
8547
9327
```

Here are the images:


train-000315.png


train-002824.png


train-004496.png


train-004612.png


train-006031.png

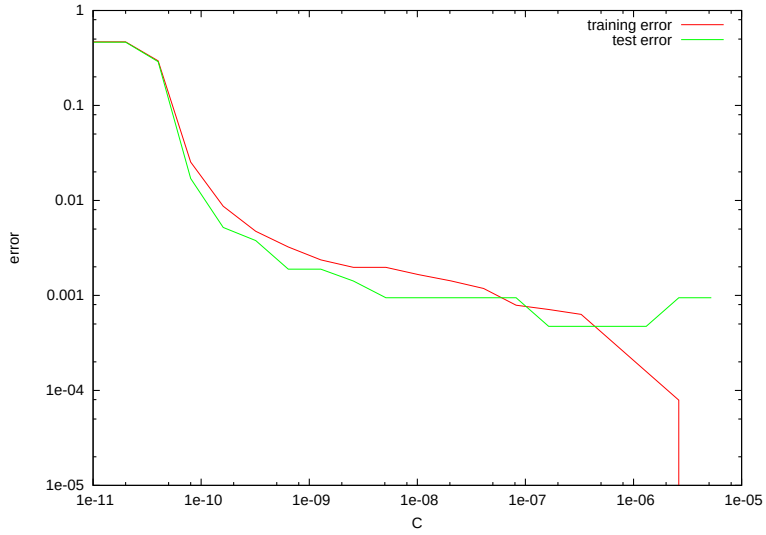

train-006627.png


train-008185.png


train-008547.png

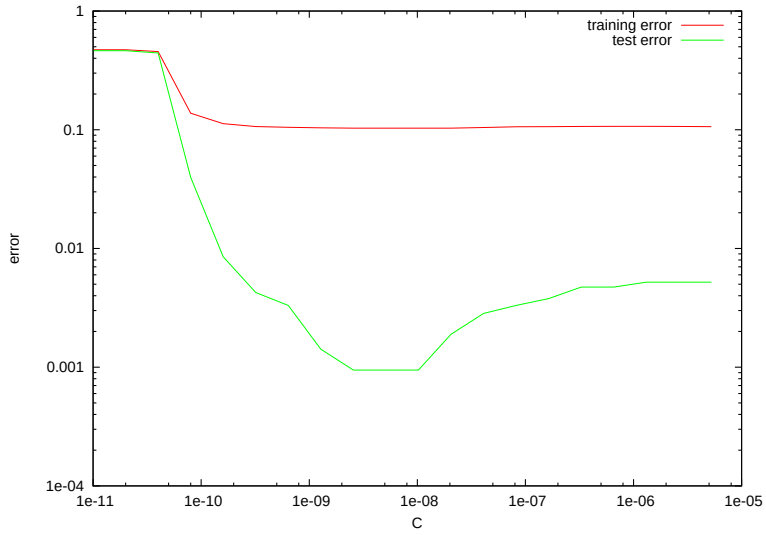

train-009327.png

(b) I trained the SVM with 20 values of C. The training error and test error are depicted below:



Notice that, except for exceptionally small values of C in the figure, the training error is 0 for this data set (it is easily separated by a hyper-plane). Observe also that the training error simply decreases as C increases; consequently, the best training error is achieved for very large C (in this case, it is zero training error). In contrast, if we could choose C with hindsight, i.e., measure the test error before having to select C, we would not pick a very large value.

(c) We produce an analogous graph for SVMs trained on the partially mislabeled data.



Clearly now, having an intermediate value of C very slightly helps the training error and significantly reduces the test error. Again, though we could not legitimately select C based on the test error, we can expect similar results from cross-validation.