CSCI 5521: Pattern Recognition

Fall 2009, Prof. Schrater

Problem set 3:

11/12/09 Due: Sat. 11/26/09

We will be using a reduced MNIST handwritten digit database for this assignment. The data has already been brought into MATLAB. Please load the files: digitimagesred.mat and digitlabels.mat from the HW3 directory. These files contain images for the digits 8, 9 and 0 in a matrix: **digitims,** that is 28x28x3000, which is a stack of 3000 28x28 images of handwritten digits. To view the *imnumber*th one of these, use

imagesc(digitims(:,:,imnumber)); colormap(gray(256));

The vector **labels** encodes the actual class labels for each of these images. This data consistutes the **training data** for the task.

1) ***Visualizing High Dimensional Data (10%):*** Use Fisher's LDA to help visualize the data. Recall that Fisher's linear discriminant results in an equation $S_w^{-1} S_B w = \lambda w$, for finding a w that optimizes Fisher's error criterion. This equation can be solved for the best *w*, resulting in the equation: $w = S_w^{-1}(\mu_1 - \mu_2)$.

Do the following.

Start with classes 8 vs. 0. Reshape the data:

```
% number of positive and negative examples
num_ω1 = length(find(labels==8));
num_ω2 = length(find(labels==0));

% stretch out images into vectors
X_ω1 = reshape(digitims(:,:,find(labels==8)),[28^2  num_ω1])';
X_ω2 = reshape(digitims(:,:,find(labels==0)),[28^2  num_ω2])';
% rescale values for numerical stability
X_ω1 = X_ω1/256;
X_ω2 = X_ω2/256;
```

Compute the means and *scatter* matrices for X_ω1 and X_ω2 {scatter is equal to S1 = num_ω1*cov(X_ω1,1)}. Form the average within class scatter

Sw = 1/(num_ω1+num_ω2 - 2)*(S1 + S2)

Similarly, compute the means m1 & m2

We need to form the vector ***w = Sw***$^{-1}$***(m1-m2).*** Try inverting ***Sw.*** What happens? The matrix is singular: Try: `rank(Sw)`

Why is the matrix singular? Look at the mean images (reshape to 28x28 and use imagesc). There are regions around the border for which all the images have zero intensity. Because each pixel forms an axis, this means that there are many axes in which the data do not live. Thus the data do not span the entire space, which results in the matrix being singular.

One simple solution is to remove the set of boundary pixels from our analysis and work in a reduced space. To visualize the boundary pixels:

```
imagesc(reshape((m1==0 | m2==0),28,28))
```

However, regularizing the inverse works better:

```
Id = 0.2*eye(size(Sw));
% small multiple of the identity
Sinv = inv(Sw+Id);
```

a) Compute the best *w* vector for another class comparison as well. (0 vs 9 or 8 vs 9). Project all the images into the 2-D space given by the two class comparisons (e.g. $w_{0\text{vs}8}$ and $w_{0\text{vs}9}$). Hint: Form a matrix W of the two vectors, and perform a matrix multiplication of W on the image data for each class, resulting in 2x1000 matrices for 0, 8, and 9. Plot these projected points using plot or scatter so that all three classes can be simultaneously viewed using different colors for each class. Label the axes using xlabel, ylabel according to the discriminant used (e.g. 0 vs. 8). You should see three distributions of points with some degree of overlap.

2) Logistic Regression (25%)

Write a program to implement Logistic regression. Apply logistic regression to the 8 and 0 digit data. Use the first 900 points from each class for training. Test the classifier on the remaining 100. You will need to transform your data. Augment each data vector to include a 1 (for the bias). Compute test error rates and compare to the average probability of error you can compute by evaluating the logistic formula on the test set.

3) SVM (30%):

a) Train a SVM linear classifier for the digit problem 8 vs. 0. Use the first 900 points from each class for training. Test the classifier on the remaining 100.

b) Kernel SVM: For kernel SVMs, the problem is modified to: maximize $-\frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) + \sum_i \alpha_i$. Train the SVM with the kernel function $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^4$. Test as in a).

c) Reshape and display the support vectors.

HINTS AND GUIDANCE

You could use the standard Matlab quadratic programming solver, which I will discuss first, or you can download the free software given below, which I recommend.

Solving Quadratic Programming problems using Matlab's toolbox functions,
Matlab like many other QP solvers, requires that the objective function and constraint

functions are expressed as canonical matrix equations in the following way:

$$\text{minimize} \quad \frac{1}{2}\alpha^T\mathbf{H}\alpha + \mathbf{f}^T\alpha$$

$$\text{such that} \quad \mathbf{A}\alpha \le \mathbf{b} \quad \text{and} \quad \mathbf{C}\alpha = \mathbf{d}$$

Using Matlab to solve the SVM dual problem.

```
epsilon = (choosesomesmallnumber);
num = size(X,2);
H = (X' * X) .* (y * y') + epsilon*eye(num);
f = -ones(1,num);
A = -eye(num);
b = zeros(num,1);
C = y';
d = 0;
alpha = quadprog(H,f,A,b,C,d);
```

Recall the Dual Problem:

$$\text{maximize} \quad -\frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i^T \mathbf{x}_j) + \sum_i \alpha_i$$

$$\text{such that} \quad \alpha_i \ge 0, \text{ for all } i$$

$$\text{and} \quad \sum_{i=1}^{n} y_i \alpha_i = 0$$

Putting this in canonical form yields the correspondences:

$$\mathbf{H} = \begin{bmatrix} h_{11} & \cdots & h_{1n} \\ \vdots & \vdots & \vdots \\ h_{n1} & \cdots & h_{nn} \end{bmatrix} \qquad h_{ij} = y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

$$\mathbf{f} = \begin{bmatrix} -1 \\ \vdots \\ -1 \end{bmatrix}$$

$$\mathbf{A} = -\mathbf{I_n} \quad (n\text{x}n \text{ Identity matrix})$$

$$\mathbf{b} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} y_1 & \cdots & y_n \end{bmatrix}$$

$$\mathbf{d} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$

Unfortunately, this straightforward approach is very inefficient, and the algorithm may not terminate well. Instead, I suggest you use the following free SVM package:

http://ida.first.fraunhofer.de/~anton/svm_v251.tar.gz

URL http://ida.first.fraunhofer.de/~anton/software.html

Download and unpack the software. Within matlab, either add the created directory to matlab's search path, or use the 'Current Directory' toolbar to make Matlab's working directory the new directory. Within matlab, type

```
mex –v loqo.c pr_loqo.c
```

Matlab will then compile c-code for a fast quadratic programming solver.

Next we need to set up our problem for the solver. Type

help svm, help svmtrain, help help svmfwd

You will need to stack all of your training data into a single matrix X that will be 1800x784. In addition, rescale your data by dividing X by 256. Create a column vector Y of 1800 class labels using 1 for the first class and –1 for the second. The calls you need to make for the non-linear kernel case will look something like:

```
polysvm = svm(28^2, ‘poly’, 4, 1, 0, 'loqo');
polysvm = svmtrain(polysvm , X, Y,[],1);
[labels, outputs] = svmfwd(net,testX);
```

I was able to train in about 2 minutes, resulting in 216 support vectors, and a test error rate of 1%.

4) **Regression (35%)**:

For the regression problem, we will be working with a dataset called "galaxy", described in Hastie, Tibshirani and Friedman. I have constructed a tutorial script that shows how to run simple linear regression, ridge regression, support vector regression and lasso regression. You will run regression in several feature spaces, including a radial basis function space, and we will assess performance using cross validation. Run galaxyData.m, which will load into memory a 2x323 predictor matrix **X** and **y:** a 1x323 vector of output values. Use

```
plot3(X(:,1),X(:,2),y,'k.')
```

to visualize the data set. The data set is difficult to predict due to the way the data was sampled (along radial arms-see the plot).

First run regressionTut.m, to learn how to use the methods. Answer the questions that follow for credit.

1) Fit a general cubic equation to the galaxy data using least squares.
2) Fit a general cubic equation to the galaxy data using ridge regression. Vary the regularization parameter and compute the leave-one-out cross validation error (CV error) on for a series of regularization parameter values. Plot CV error against the regularization parameter values. Find a value that (approximately) minimizes error.
3) Fit radial basis functions to the galaxy data using three methods –

   a. Ordinary least squares
   b. Lasso
   c. Support vector regression

4) Compare CV error for each of the three methods in problem 3). For both the Lasso and SV regression, search for values of the regularization parameter C that perform well. (Basically I am asking you to experiment with these values to see their effects, however SV regression takes a long time, so I don't expect you to try to systematically minimize CV error).

Tutorial - Read about how to map the lasso to a quadratic program below.

Given a matrix Z of with m observations (rows) and n predictors,

the lasso regression estimate is the solution to:

$$\hat{w} = \min_{w} (y - Zw)^T (y - Zw)$$

subject to

$$\sum_{j=1}^{n} |w_j|$$

Lets rewrite this as a quadratic program

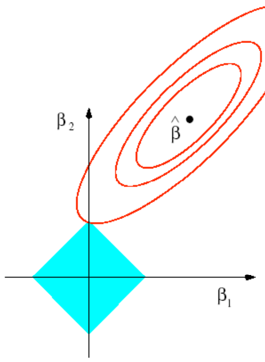
$$\begin{aligned} \hat{w} &= \min_{w} (y - Zw)^T (y - Zw) \\ &= \min_{w} \left[\tfrac{1}{2} 2 \left(y^T y - 2 y^T Z w + w^T Z^T Z w \right) \right] \\ &= \min_{w} \left[w^T H w + f^T w \right] \end{aligned}$$

where $H = 2Z^T Z$,

$f^T = -4 y^T Z$,

and $y^T y$ doesn't affect the minimization

The sum of absolute values can be converted into a set of linear inequalities. For example, **in a 2D problem**, we have a quadratic loss function, and a constraint that the coefficient vectors have to lie inside a diamond around the origin.



Note that
$|w_1|+|w_2|<c$ is the same as:

$$\begin{array}{r} w_1+w_2<c \\ w_1-w_2<c \\ -w_1+w_2<c \\ -w_1-w_2<c \end{array} \quad \begin{cases} \text{if } w_1>0 \;\&\; w_2>0 \\ \text{if } w_1>0 \;\&\; w_2<0 \\ \text{if } w_1<0 \;\&\; w_2>0 \\ \text{if } w_1<0 \;\&\; w_2<0 \end{cases}$$

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} < c$$

In general we can find a set of linear constraints equivalent to the sum of absolute values and express them as a linear inequality similar to the example above, and there are other tricks for doing this conversion. For large dimensional problems, the number of inequalities gets large, and it is computationally simpler to use a different optimization procedure, like the one included in the homework (lasso.m).

EXTRA CREDIT-10%.
There are many good Pattern Recognition Toolboxes out there. Download and install one of the toolboxes below. Apply a method to the digit data. For example, download SPIDER (below) and run the multi-class svm. Email code and results figures for credit.

Here are a few:
MATLAB based
Netlab functions, a free Pattern Recognition toolbox available here:
http://www.ncrg.aston.ac.uk/netlab/index.php
SPIDER, a free machine learning toolbox available here:
http://www.kyb.tuebingen.mpg.de/bs/people/spider/
Non-MATLAB based
Rapidminer
This is a java-based data mining GUI with a large set of methods implemented. It is relatively easy to use