Machine Learning – Assignment 2 (Parts A&B)

Due date: 23rd May 2019

Part A: SVMs and Bayes Classifiers [45 Marks]

Support Vector Machines [30 Marks]

We have povided the following Matlab routines for constructing svm classifiers:
`Klinear.m, Kpoly.m, Kgaussian.m` A linear, polynomial, and Gaussian kernel. Besides the two vectors, the kernel takes an extra parameter. In the case of the polynomial kernel, the extra parameter is the degree. In the case of the Gaussian, the extra parameter is the standard deviation. The linear kernel ignores this parameter. The name of the kernel must be prefixed by '`@`' when passed to `svm train.m`.

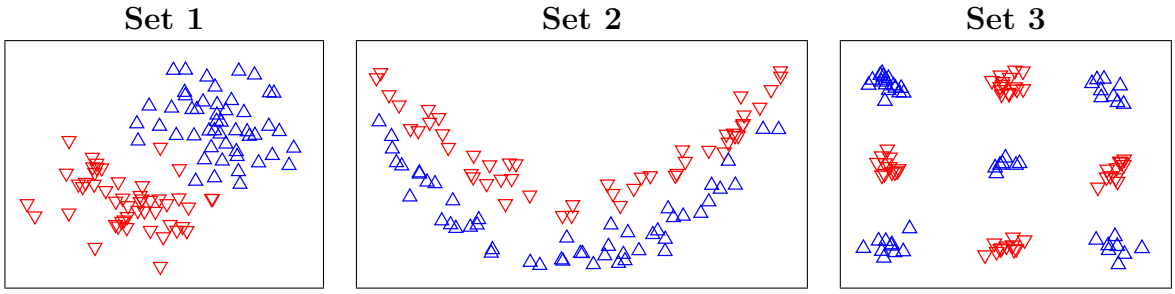
`svm_train.m` Trains a SVM given training data, a kernel, kernel parameter, and the C penalty on training errors.

`svm_discrim_func.m` Evaluates the SVM hyperplane on a set of test points. Data is classified according to the sign of this evaluation.

`svm_plot.m` Plots the SVM decision boundary and the supplied labeled datapoints.

`svm_test.m` Runs an SVM experiment by training the SVM on the supplied training data, and testing it on the supplied test data. Plots the SVM decision boundary, and the test errors.

We have also provided four train/test sets of data. The first three are artificially generated 2-dimensional data, while the last one is the 64-dimensional digit dataset. You can load all data with `load data ps3 2.mat`. The 2-dimensional datasets are pictured below:



1. For the first three datasets, consider the linear, second order polynomial, and Gaussian of standard deviation 1 kernels. For each dataset provide a rationale for which kernel should be the best for training a SVM classifier on that dataset. Choose the best kernel for each dataset and plot the decision boundary and test errors with svm test.m. Hand in the three plots. (for consistency, everybody should use C = 1000). [15 marks]

2. For the digit dataset (set4 train and set4 test), train and test SVM's with a linear, polynomial of degree 2, and Gaussian of standard deviation 1.5 kernels. Report the test errors.

   (Note: you will have to use svm train.m and svm discrim func.m directly, because svm test.m is for 2D data only). [15 marks]

Bayes Classifiers [15 Marks]

In this problem you will use bayes rule,

$$p(y|x) = p(x|y)p(y)/p(x)$$

to perform classification.

Suppose that we observe the following training data, with two binary-valued features x_1 and x_2 and a binary-valued class label y:

x_1	x_2	y
0	0	0
0	0	1
0	0	1
0	0	1
0	1	0
0	1	1
0	1	1
0	1	1
1	0	0
1	0	0
1	0	0
1	1	0
1	1	0
1	1	0
1	1	1
1	1	1

We will classify a small test set,

x_1	x_2	y
0	1	1
1	0	1
1	1	0

(a) What probabilities do you need to create a *joint* Bayes classifier (i.e., a Bayes classifier that uses the joint distribution over all features), and what are their estimated values given the training data? Use them to classify the test set; report both the class prediction and the estimated probability $p(y = C|x_1, x_2)$ of each class C given the feature values x_1, x_2.

(b) What probabilities do you need to create a *naïve* Bayes classifier (a Bayes classifier that uses a conditional independence assumption), and what are their estimated values given the training data? Use them to classify the test set and report both the class prediction and estimated probability $p(y = C|x_1, x_2)$ of each class.

Part B: PCA & Clustering [45 marks]

EigenFaces [25 Marks: 5 marks for each section (a) to (e)]

PCA has been applied to faces, and we showed some of the results in the class. Here, you'll explore this representation yourself. First, load the data and display a few faces to make sure you understand the data format:

```
X = load('data/faces.txt');          % load face dataset
img = reshape(X(i,:),[24 24]);  % convert vectorized datum to 24x24 image patch
imagesc(img); axis square; colormap gray;    % display an image patch
```

(a) Subtract the mean of the face images ($X_0 = X - \mu$) to make your data zero-mean. Take

the SVD of the data, so that

$$X_0 \approx U \cdot S \cdot V^T$$

Note that since the number of data is larger than the number of dimensions, S will be zero on its lower part; I suggest tak ing $X_0 \approx W \cdot V$ where $W = U \cdot S$. You may also prefer to use `svds`, which can return only the top K singular values and their associated columns of U and V.

(b) For $K = 1 \ldots 10$, compute the app roximation to X_0 given by $\hat{X}_0 = W(:, 1:K) \cdot V(:, 1:K)^T$, and compute the mean squared error in the SVD's approximation, `mean(mean( (`$X_0 - \hat{X}_0$`).^2 ))`. Plot these values as a function of K.

(c) Display the first few principal directions of the data, by computing μ+ α V(:,j)' and μ-α V(:,j)', where α is a scale factor (I suggest, for example, `2*median(abs(W(:,j)))`, to get a sense of the scale found in the data). These should be vectors of length 24^2, so you can reshap e them and view them as "face images" similar to the original data.

(d) These are often called "latent space" metho ds, as the coeficients can be interpreted as a new geometric space in which the data are being described. To visualize this, choose a few faces at random (say, about 15–25), and display them as images with the coordinates given by their coeficients on the first two principal components:

```
idx = ...      % pick some data at random or otherwise
figure; hold on; axis ij; colormap(gray);
range = max(W(idx,1:2)) - min(W(idx,1:2)); % find range of coordinates to be plotted
scale = [200 200]./range;                  % want 24x24 to be visible
for i=idx, imagesc(W(i,1)*scale(1),W(i,2)*scale(2), reshape(X(i,:),24,24)); end;
```

This can often help you get a "feel" for what the latent representation is capturing.

(e) Choose two faces and reconstruct them using only K principal directions, for $K = 5, 10, 50$.

Clustering [20 Marks: 5 marks for each section (a) to (d)]

Basic exploration of the clustering techniques.

(a) Load the usual Iris data restricted to the first two features, and ignore the class / target variable. Plot the data and see for yourself how "clustered" you think it looks. (You don't have to report on this.)

(b) Run k-means on the data, for $k = 5$ and $k = 20$. For each, turn in a plot with the data, colored by assignment, and the cluster centers. (You can do this yourself manually, using `plotClassify2D([],X,z)`, or just use the plotting options in the k-means code.) Try a few different initializations and check to see whether they find the same solution; if not, pick the one with the best score.

(c) Run agglomerative clustering on the data, using *single linkage* and then again using *complete linkage*, each with 5 and then 20 clusters. Again, plot with color the final assignment of the clusters, and describe their similarities and differences from each other and k-means. (This algorithm has no initialization issues; so you do not have to try multiple initializations.)

(d) Run the EM Gaussian mixture model with 5 and 20 components. Using the internal plotting functions will show you the evolution of the mixture components' locations and shapes (the magnitude coefficient isn't shown). As with k-means, you may want to try several initializations. Again, compare / discuss differences with the other clusterings. Which do you think is most reasonable?

As a side note: Clustering is often a useful element of other predictive tasks, like supervised learning. To be used properly, you need to be able to define the "out of sample" cluster assignments, but this is very easy for k-means and EM (a bit less so for agglomerative); for k-means, say:
`crule = knnClassify( clusters, (1:K)', 1 ); z = predict( crule, X );`
Then, you can then use these cluster assignments as a feature in a classifier:
`Phi = @(x) to1ofK( predict(crule, x) , 1:K );` will create K new binary features indicating which of the clusters is closest to a new point x.

References:
[1] Codes&Practices: Courtesy of Alex Ihler. Used with permission.

Deliverables

You should submit via Blackboard a zip file containing

1. A report in pdf format including
    - data analysis and reporting on methods and results using Matlab/Python code
    - enough detail about how you solved the problem
2. Your Matlab/Python files

Draft Marking Scheme

- Report and Code quality (readability, in line documentation): 10 Points
- Part A: 45 Points.
  Part B: 45 Points.
- Total: 100 Points.
  Contribution to final mark: 15%

Final Remarks

- You are encouraged to ask questions during the practical sessions to check that you are on the right track.