Machine Learning Lab 1

$$\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{m}, \boldsymbol{C}),\ \boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \implies \boldsymbol{y} \sim \mathcal{N}(\boldsymbol{A}\boldsymbol{m},\ \boldsymbol{A}\boldsymbol{C}\boldsymbol{A}^T)$$

1. Familiarize yourself with `MATLAB`. Work through the examples in the document `http://users.ecs.soton.ac.uk/mn/MatlabIntroduction.pdf`, which are notes accompanying a textbook on `MATLAB`. Use of the `help` and `lookfor` commands help you learn a broad range of the features of the language. The documents `http://users.ecs.soton.ac.uk/mn/MatlabProgramming.pdf` and `http://users.ecs.soton.ac.uk/mn/MatlabStyle.pdf` are also worth going through, but not essential to get started.

2. Generate 1000 uniform random numbers and plot a histogram. Here are the useful commands in `MATLAB` to do this.

```
> x = rand(1000,1);
> hist(x,40);
> help hist
> [nn, xx] = hist(x);
> bar(nn);
```

   Repeat the above with 1000 random numbers drawn from a Gaussian distribution of mean 0 and standard deviation 1 using `x = randn(1000,1);`. You now see how data drawn from two different probability densities are distributed. Change the number of bins into which data is split (*i.e.* the 40 in `hist(x,40)`) and note the differences.
   Now try the following

```
> N = 1000;
> x1 = zeros(N,1);
> for n=1:N
>   x1(n,1) = sum(rand(12,1))-sum(rand(12,1));
> end
> hist(x1,40);
```

   What do you observe? Is there a theorem that explains your observation?
   Note: Do not type the `MATLAB` statements one at a time into command line; type them into a text file `labone.m` and invoke the script by `> labone` against the prompt. Look up `> help path`.

3. Consider the covariance matrix $\boldsymbol{C} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$.
   Factorize this into $\boldsymbol{A}^t\boldsymbol{A} = \boldsymbol{C}$ using `> A = chol(C)`.
   Confirm the factorization is correct by multiplying. Generate 1000 bivariate Gaussian random numbers by `X=randn(1000,2);`
   Transform each of the two dimensional vectors (rows of `X`) by `> Y=X*A`.
   Now draw a scatter plot of `X` and `Y`.
   `> plot(X(:,1),X(:,2),'c.', Y(:,1),Y(:,2),'mx');`
   What do you observe?

   Construct a vector $\boldsymbol{u} = [\sin\theta\ \cos\theta]$, parameterized by the variable θ and compute the variance of projections of the data in $\boldsymbol{Y}$ along this direction:

```
> theta = 0.25;
> u = [sin(theta); cos(theta)]
> yp = Y*u;
```

```
> var_empirical = var(yp)
> var_theoretical = u'*C*u;
```

In the above, the variance of projections has been calculated in two ways (theoretical, *i.e.* from a formula and empirical, *i.e.* by simulating data). Explore how the difference between the two changes with the number of data points used (at 10, 100, 1000 and 10000).
Plot how this projected variance changes as a function of θ:

```
> N = 50;
> plotArray = zeros(N,1);
> thRange = linspace(0,2*pi,N);
> for n=1:N
> ...
> ...
> end
> plot(plotArray)
```

Explain what you observe by calculating the eigenvectors of the covariance matrix.

Derive an expression for the projected variance analytically for this two-dimensional case (*i.e.* the variance of projected data as a function of θ) and confirm that the plot you have drawn is correct.

How does what you have done above differ for $\boldsymbol{C} = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$.

Export the figures for inclusion in a report `> print -depsc f1.eps`.

Machine Learning Lab 2

1. Draw a line: A straight in two dimensional coordinate geometry is defined as $a_1x + b_1y + c_1 = 0$. It intersects the x and y axes at $-c/b_1$ and $-c/a_1$ respectively. Therefore, points $(-c/b_1, 0)$ and $(0, -c/a_1)$ are points on this line. So, if we have $3x + 4y + 12 = 0$, we can plot this in `MATLAB` by:

```
plot([0 -4], [-3 0], 'b', 'LineWidth', 2);
axis([-5 5 -5 5]); grid on;
```

   The perpendicular distance from the origin to the straight line above is $-c/\sqrt{a_1^2 + b_1^2}$. Confirm this is correct on the plot you have drawn (simple visual inspection on the graph is fine).

   (In vector notation, we will write the equation of the straight line as $\boldsymbol{w}^t\,\boldsymbol{x} + b = 0$ in which the vector $\boldsymbol{w}$ contains information about the slope and the *bias* term b relates to the offset of the line. The distance from the origin to the line is thus $-b/|\boldsymbol{w}|$. We will return to this when we study support vector machine later in the module.)

2. Generate 100 samples each from two bi-variate Gaussian densities with distinct means $\boldsymbol{m}_1 = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$ and $\boldsymbol{m}_2 = \begin{bmatrix} 1.5 \\ 0.0 \end{bmatrix}$, and identical covariance matrix $\boldsymbol{C} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$.

   Hints:

   - See work in `Lab One` in which we used cholesky decomposition (`chol`) of the covariance matrix and turned samples from an isotropic distribution to samples from a distribution with the desired covariance matrix.
   - If `X` is an `N` by `2` array variable containing zero mean bivariate samples, one way of shifting the mean of the distribution is (also see `repmat`):

```
> m1 = [0 2];
> X1 = X + kron(ones(N,1), m1);
> help kron
```

   Plot the data as a scatter plot.

3. Compute the Bayes' optimal class boundary (see slides of lecture in Week 2) between the two distributions and draw it on the same graph (Hint: Use `> hold on` to retain a plot and display a second graph on top of it).

4. Implement a perceptron learning algorithm to classify this data and compare its solution with the Bayes' optimal boundary. Hint: please see Appendix for illustrative snippet of code (with no guarantees of correctness) which might help you get started. Explore how convergence of the learning algorithm changes with (a) different values of the learning rate; and (b) separation of the two classes.

Appendix

Perceptron Learning: Code snippet

```
> % Set up your own data in the following form
> % X: N by 2 matrix of data
> % y: class labels -1 or +1
> % include column of ones for bias
> X = [X ones(N,1)];
>
> % Separate into training and test sets (check: >> doc randperm)
> ii =  randperm(N);
> Xtr = X(ii(1:N/2),:);
> ytr = X(ii(1:N/2),:);
> Xts = X(ii(N/2+1:N),:);
> yts = y(ii(N/2+1:N),:);
>
> % initialize weights
> w = randn(3,1);
>
>
> % Error correcting learning
> eta = 0.001;
> for iter=1:500
>     j = ceil(rand*N/2);
>     if ( ytr(j)*Xtr(j,:)*w  < 0 )
>          w = w + eta*ytr(j)*Xtr(j,:)';
>     end
> end
>
> % Performance on test data
> yhts = Xts*w;
> disp([yts yhts])
> PercentageError = 100*sum(find(yts .* yhts < 0))/Nts;
>
```

Machine Learning Lab 3

1. Define a two class pattern classification problem in two dimensions, in which the two classes are Gaussian distributed with means $\boldsymbol{m}_1 = [0 \;\; 2]^t$ and $\boldsymbol{m}_1 = [1.7 \;\; 2.5]^t$, and have a common covariance matrix

$$\boldsymbol{C}_1 = \boldsymbol{C}_2 = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}.$$

   Plot contours on the density by adapting the code snippet below.

```
numGrid = 50;
xRange = linspace(-6.0, 6.0, numGrid);
yRange = linspace(-6.0, 6.0, numGrid);
P1 = zeros(numGrid, numGrid);
P2 = P1;
for i=1:numGrid
    for j=1:numGrid;
        x = [yRange(j) xRange(i)]';
        P1(i,j) = mvnpdf(x', m1', C1);
        P2(i,j) = mvnpdf(x', m2', C2);
    end
end
Pmax = max(max([P1 P2]));
figure(1), clf,
contour(xRange, yRange, P1, [0.1*Pmax 0.5*Pmax 0.8*Pmax], 'LineWidth', 2);
hold on
plot(m1(1), m1(2), 'b*', 'LineWidth', 4);
contour(xRange, yRange, P2, [0.1*Pmax 0.5*Pmax 0.8*Pmax], 'LineWidth', 2);
plot(m2(1), m2(2), 'r*', 'LineWidth', 4);
```

2. Draw 200 samples from each of the two distributions and plot them on top of the contours.

```
N  = 200;
X1 = mvnrnd(m1, C1, N);
X2 = mvnrnd(m2, C2, N);
plot(X1(:,1),X1(:,2),'bx', X2(:,1),X2(:,2),'ro');grid on
```

3. Compute the Fisher Linear Discriminant direction using the means and covariance matrices of the problem, and plot the discriminant direction:

```
wF = inv(C1+C2)*(m1-m2);
xx = -6:0.1:6;
yy = xx*wF(2)/wF(1);
plot(xx,yy,'r', 'LineWidth', 2);
```

4. Project the data onto the Fisher discriminant directions and plot histograms of the distribution of projections:

```
p1 = X1*wF;
p2 = X2*wF;

plo = min([p1; p2]);
phi = max([p1; p2]);
[nn1, xx1] = hist(p1);
[nn2, xx2] = hist(p2);
hhi = max([nn1 nn2]);
figure(2),
subplot(211), bar(xx1, nn1);
```

```
axis([plo phi 0 hhi]);
title('Distribution of Projections', 'FontSize', 16)
ylabel('Class 1', 'FontSize', 14)
subplot(212), bar(xx2, nn2);
axis([plo phi 0 hhi])
ylabel('Class 2', 'FontSize', 14)
```

5. Compute a Receiver Operating Characteristic (ROC) curve, by sliding a decision threshold, and computing the True Positive and False Positive rates:

```
thmin = min([xx1 xx2]);
thmax = max([xx1 xx2]);

rocResolution = 50;
thRange = linspace(thmin, thmax, rocResolution);
ROC = zeros(rocResolution,2);
for jThreshold = 1: rocResolution
    threshold = thRange(jThreshold);
    tPos = length(find(p1 > threshold))*100 / N;
    fPos = length(find(p2 > threshold))*100 / N;
    ROC(jThreshold,:) = [fPos tPos];
end
figure(3), clf,
plot(ROC(:,1), ROC(:,2), 'b', 'LineWidth', 2);
axis([0 100 0 100]);
grid on, hold on
plot(0:100, 0:100, 'b-');
xlabel('False Positive', 'FontSize', 16)
ylabel('True Positive', 'FontSize', 16);
title('Receiver Operating Characteristic Curve', 'FontSize', 20);
```

6. Compute the area under the ROC curve (Hint: try `>help trapz`)

7. For a suitable choice of decision threshold, compute the classification accuracy.

8. Plot the ROC curve (on the same scale) for

   - A random direction (instead of the Fisher discriminant direction).
   - Projections onto the direction connecting the means of the two classes.

   Compute the area under the ROC curve for these two cases.

   (Optional: Since you need to call the ROC computing code several times, tidy up your code by writing this part as a function to which you pass the data as parameters and have the ROC curve returned as answer.)

9. Implement a nearest neighbour classifier (1-NN) on this data, and compare its accuracy with that of the Fisher Discriminant Analyzer.

```
% Nearest neighbour classifier
% (Caution: The following code is very inefficient -- why?)
X = [X1; X2];
N1 = size(X1, 1);
N2 = size(X2, 1);
y = [ones(N1,1); -1*ones(N2,1)];
d = zeros(N1+N2-1,1);
nCorrect = 0;
for jtst = 1:(N1+N2)
    % pick a point to test
```

```
    %
    xtst = X(jtst,:);
    ytst = y(jtst);

    % All others form the training set
    %
    jtr = setdiff(1:N1+N2, jtst);
    Xtr = X(jtr,:);
    ytr = y(jtr,1);

    % Compute all distances from test to training points
    %
    for i=1:(N1+N2-1)
        d(i) = norm(Xtr(i,:)-xtst);
    end

    % Which one is the closest?
    %
    [imin] = find(d == min(d));

    % Does the nearest point have the same class label?
    %
    if ( ytr(imin(1)) * ytst > 0 )
        nCorrect = nCorrect + 1;
    else
        disp('Incorrect classification');
    end
end

% Percentage correst
%
pCorrect = nCorrect*100/(N1+N2);
disp(['Nearest neighbour accuracy:  ' num2str(pCorrect)]);
```

10. For the dataset you have generated, construct a distance-to-mean classifire using (a) Euclidean distance and (b) Mahalanobis distance as distance measures and compare their classification accuracies.

11. For the above classification problem, compute and plot a three dimensional graph of the posterior probability of one of the two classes for the Bayes optimal classifier. Does the graph match your expectations from theory?

12. What do we expect the Baye's optimal class boundary to be, if $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ are not identical? Write out the algebra to show this. Change $\boldsymbol{C}_2$ to a different covariance matrix, *e.g.* $\boldsymbol{C}_2 = 1.5\boldsymbol{I}$ and illustrate the theoretical prediction.

Machine Learning Lab 4

In this task we will use the convex optimization package `CVX`. Download the appropriate version of the package from `http://cvxr.com/cvx/download/`, store it in a convenient place in your filespace, uncompress it and run the script `cvxsetup.m` that comes with it to set paths correctly.

Linear Least Squares Regression:

Download the `Boston Housing` dataset from the `UCI Machine Learning` repository [1]; this comes in two files: `housing.data`, which contains the data and `housing.names`, which describes the different variables and other uses of the dataset. Load the data into `MATLAB` and normalize the variables as follows:

```
% Load Boston Housing Data from UCI ML Repository
% into an array housing_data; Normalize the data to have
% zero mean and unit standard deviation
%
[N, p1] = size(housing_data);
p = p1-1;
Y = [housing_data(:,1:p) ones(N,1)];
for j=1:p
    Y(:,j)=Y(:,j)-mean(Y(:,j));
    Y(:,j)=Y(:,j)/std(Y(:,j));
end
f = housing_data(:,p1);
f = f - mean(f);
f = f/std(f);
```

You can predict the response variable (output variable) f, the house price, from the covariates (input variable) by estimating a linear regression:

```
% Least squares regression as pseudo inverse
%
w = inv(Y'*Y)*Y'*f;
fh = Y*w;
figure(1), clf,
plot(f, fh, 'r.', 'LineWidth', 2),
grid on
xlabel('True House Price', 'FontSize', 14)
ylabel('Prediction', 'FontSize', 14)
title('Linear Regression', 'FontSize', 14)
```

Split the data into a training set and a test set, estimate the regression model ($\boldsymbol{w}$) on the training set and see how training and test errors differ.
Implement 10-fold cross validation on the data and quantify an average prediction error and an uncertainty on it.

Regression using the CVX Tool:

The least squares regression you have done in the above section can be implemented as follows in the `cvx` tool:

```
cvx_begin quiet
   variable w1( p+1 );
   minimize norm( Y*w1 - f )
cvx_end
fh1 = Y*w1;
```

Check if the two methods produce the same results.

```
figure(2), clf,
plot(w, w1, 'mx', 'LineWidth', 2);
```

Sparse Regression:

Let us now regularize the regression: $w_2 = \min_{\boldsymbol{w}} |Y\boldsymbol{w} - \boldsymbol{f}| + \gamma |\boldsymbol{w}|_1$. You can implement this as follows:

```
gama = 8.0;
cvx_begin quiet
      variable w2( p+1 );
      minimize( norm(Y*w2-f) + gama*norm(w2,1) );
cvx_end
fh2 = Y*w2;
plot(f, fh1, 'co', 'LineWidth', 2),
legend('Regression', 'Sparse Regression');
```

You can find the non-zero coefficients that are not switched off by the regularizer:

```
[iNzero] = find(abs(w2) > 1e-5);
disp('Relevant variables');
disp(iNzero);
```

Find out from `housing.names.txt` which of the variables are selected as relevant to the house price prediction problem.

The amount of regularization is controlled by γ, for which I have selected a convenient value. Write a program to change this parameter over the range $0.01 \rightarrow 40$ in 100 steps and plot a graph of how the number of non-zero coefficients changes with increasing regularization.

Machine Learning Lab 5

Construct a radial basis functions (RBF) model to predict house prices using the "Boston Housing Data," used in `Lab 4`. The RBF model is given by

$$g(\boldsymbol{x}) = \sum_{k=1}^{K} \lambda_k \phi(||\boldsymbol{x} - \boldsymbol{c}_k||).$$

This is a model in which the nonlinear part is fixed and only the weights λ_k are estimated in a manner similar to linear regression. The nonlinear part is fixed in some sensible way (we will use $K-$means clustering to do this). We will use a Gaussian RBF $\phi(\alpha) = \exp(-\alpha/\sigma^2)$.

1. Load the data, normalize it as done in `Lab 4` and get random partitions of traing and test sets. Say variable `Xtr`, a matrix of `Ntr` $\times$ `p` is inputs of your training set and `ytr`, the corresponding outputs (targets).

2. Set the widths of the basis functions to a sensibel scale; here I use the distance between two randomly chosen items of data:

```
sig = norm(Xtr(ceil(rand*Ntr),:)-Xtr(ceil(rand*Ntr,:));
```

3. Perform $K-$means clustering to find centres $\boldsymbol{c}_k$ for the basis functions. Use $K = Ntr/10$.

```
help kmeans
[Idx, C] = kmeans(Xtr, round(Ntr/10))
```

4. Construct the *design matrix*

```
for i=1:Ntr
    for j=1:K
        A(i,j)=exp(-norm(Xtr(i,:) - C(j,:))/sigma^2);
    end
end
```

5. Solve for the weights

```
lambda = A \ ytr;
```

6. Compute what the model predict at each of the training data:

```
yh = zeros(Ntr,1);
u  = zeros(Ntr,1);
for n=1:Ntr
   for j=1:K
      u(j)  = exp(-norm(Xtr(n,:) - C(j,:))/sigma^2);
   end
   yh(n) = lambda'*u;
end
plot(y, yh, 'rx', 'LineWidth', 2), grid on
title('RBF Prediction on Training Data', 'FontSize', 16);
xlabel('Target', 'FontSize', 14);
ylabel('Prediction', 'FontSize', 14);
```

7. Adapt the above to calculate what the model predicts at the unseen data (test data) and draw a similar scatter plot. How do the training and test errors compare? Compute the difference between training and test errors at different values of the number of basis functions, K. Briefly comment on any observation you make.

8. Compare your results with the linear regression model of `Lab 4`. Does the use of a nonlinear model improve predictions?

9. Carry out a similar comparison between linear and nonlinear (RBF) models on a different dataset of your choice taken from the `UCI` Machine Learning repository.

10. Upload a two-page report on your work before the stated deadline.

Note: When comparing performances of the linear and RBF models, partition the data into training / test sets multiple times (say 20), evaluate the test set prediction errors, and present your results as two boxplots drawn side by side, as in Fig. .

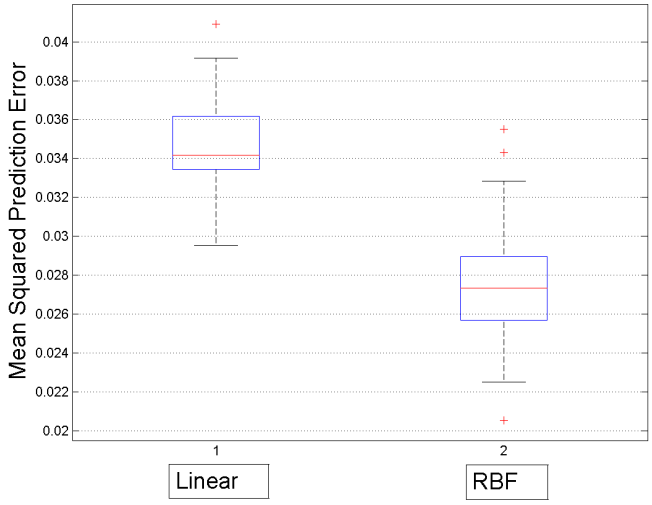

Figure 1: Comparison of linear and radial basis functions (RBF) models on predicting house prices. The mean squared error on test data is obtained from 20 random partitionings.

Mahesan Niranjan January 2018

References

[1] K. Bache and M. Lichman, "UCI machine learning repository." `http://archive.ics.uci.edu/ml`, 2013.