10-701 Machine Learning (Spring 2012)
Solution for problem set 1

1. Basic probability and statistics

1.1 Probability distribution

Distinction between probability density (continuous) and mass (discrete).

1.2 Mean and variance

$$\mathrm{E}(\mathrm{X}) = \int \mathrm{x}\mathrm{p}(\mathrm{x})\mathrm{dx} = \int_0^{\frac{1}{2}} 2\mathrm{x}\mathrm{dx} = \frac{1}{4}$$

$$\mathrm{Var}(\mathrm{X}) = \mathrm{E}(\mathrm{X}^2) - [\mathrm{E}(\mathrm{X})]^2 = \int_0^{\frac{1}{2}} 2\mathrm{x}^2\mathrm{dx} - \left(\frac{1}{4}\right)^2 = \frac{1}{12} - \frac{1}{16} = \frac{1}{48}$$

1.3 Sampling

Let $\mathrm{X} = \frac{\mathrm{Y}}{2}$, one can sample from $\mathrm{Y} \sim \mathrm{Uniform}(0,1)$ and divide the sampled value into 2.

$$\mathrm{E}(\mathrm{X}) = \mathrm{E}\left(\frac{\mathrm{Y}}{2}\right) = \int_0^1 \frac{\mathrm{y}}{2} 1\mathrm{dy} = \frac{1}{4}$$

$$\mathrm{Var}(\mathrm{X}) = \mathrm{E}\left((\frac{\mathrm{Y}}{2})^2\right) - [\mathrm{E}(\mathrm{X})]^2 = \int_0^1 \left(\frac{\mathrm{y}}{2}\right)^2 1\mathrm{dx} - \left(\frac{1}{4}\right)^2 = \frac{1}{12} - \frac{1}{16} = \frac{1}{48}$$

1.4 More on expectations

$$\mathrm{E}(\text{roll a 6}) = \frac{1}{6}1 + \frac{5}{6}[\mathrm{E}(\text{roll a 6}) + 1] \Rightarrow \mathrm{E}(\text{roll a 6}) = 6$$

1.5 Detective Bayes

Let B be the event biased towards 3, F be the event fair (die) and D be data. Judging fairness boils down to comparing the log odds of the posteriors of two distinct hypotheses.

$$\ln\frac{\mathrm{P}(\mathrm{B}|\mathrm{D})}{\mathrm{P}(\mathrm{F}|\mathrm{D})} = \ln\frac{\mathrm{P}(\mathrm{D}|\mathrm{B})\mathrm{P}(\mathrm{B})}{\mathrm{P}(\mathrm{D}|\mathrm{F})\mathrm{P}(\mathrm{F})} = \ln\frac{(0.1)^7(0.5)^3 0.05}{\left(\frac{1}{6}\right)^{10} 0.95} = -3.22$$

Since the log odds is tiny, i.e. $-3.22 - \ln(.5) = -2.53$, it is quite certain that the die is fair given the observations.

2. Linear regression

2.1 Probabilistic interpretation

The likelihood function conditioned on the input-output observations is in the form of a Gaussian where $y_i - wx_i \sim \text{Normal}(0,1)$.

$$L(Y|X,W) = \ln\left(\prod P(y_i|x_i,w)\right) = \ln\left(\prod \frac{1}{\sqrt{2\pi 1^2}}\exp\left(-\frac{(y_i - wx_i)^2}{2}\right)\right)$$
$$\propto -\sum(y_i - wx_i)^2$$

Thus minimizing the sum of squared error is equivalent to maximizing the log likelihood.

2.2 Geometric interpretation

Minimizing sum of squared error between estimates and target values is equivalent to minimizing the (squared) Euclidean distance between WX and Y.

2.3 Regularization

$$\frac{\partial}{\partial w}(y - Xw)^t(y - Xw) + 2\lambda w^t w$$
$$= \frac{\partial}{\partial w}(y^t - w^t X^t)(y - Xw) + 2\lambda w^t w$$
$$= \frac{\partial}{\partial w} y^t y - y^t Xw - w^t X^t y + w^t X^t Xw + 2\lambda w^t w$$
$$= -2y^t X + 2w^t X^t X + 2\lambda w^t$$

If we set the derivative to zero, we obtain an expression for the weight vector

$$-2y^t X + 2w^t X^t X + 2\lambda w^t = 0 \Rightarrow (X^t X + \lambda I)w = X^t y \Rightarrow w = (X^t X + \lambda I)^{-1} X^t y$$

Note the weight decreases as the penalty increases by increasing λ.

3. Density estimation

3.1 When $k = n$, the training error should have an expected value of a half. When $k = 1$, there will be no training error since every point "predicts" its own label which is always true. The error should lie between these two values as k varies.

3.2 The sketch of generalization error looks roughly convex where there should be an optimal choice of k that lies in between n and 1. At $k = 1$, the generalization error should be high because the model is over complex and is unlikely to predict well. At $k = n$, the model has a poor structure and is also unlikely to perform well in prediction.

3.3 Many options here. A typical answer could be cross-validation where k is chosen such

that the generalization error can be minimized.

3.4 Many options here. A typical answer could be to weight the contribution of neighbors in proportion to some functional form of their distances to the target point, e.g. a Gaussian radial basis. This way the closer neighbors receives more trust than distant ones and smooth out the estimate.

3.5 A) computational overheads in computing high-dimensional pairwise distances. B) the points are much further apart in a high dimensional space and the number of points needed to provide a reliable estimate for KNN grows exponentially (curse of dimensionality).

4. Naïve Bayes and classification

4.1 In a full Gaussian model, one needs to estimate N parameters in the mean and $\frac{\mathrm{N}^2-\mathrm{N}}{2}$ unique parameters in the covariance matrix, totaling $\frac{\mathrm{N}^2+3\mathrm{N}}{2}$ parameters, as opposed to $\mathrm{N} \times 2$ parameters for unit-dimensional Gaussian means and variances in a Naïve Bayes model.

4.2 The plots here should include the mean of the cross-validation error (or accuracy) and the standard deviation (or that of the mean) across folds. Depending on the data set one generates, the trends may look different in each case. But almost certainly under parameter set 2, the full Gaussian should perform (significantly) better at all data sizes.

4.3 A good answer for prediction could relate the patterns of data under two parameterizations to your understanding of the NB and full Gaussian assumptions. For example, since the two dimensions are highly dependent under parameter set 2, you would expect the full Gaussian to perform much better because it captures the covariation of the features, whereas the NB can't. Another possible point to mention is that as the data size grows, you would expect both classifiers to perform better. However, as you could see in this particular dataset it isn't the case because 1) the configuration of the two classes are relatively simple (no intricate overlap) 2) the dimension is low at 2.

See sample code in Appendix for reference (permission from *Max Korein)*.

5. Bonus question

Lots of ways you can tackle this question. An appropriate classifier in this case would be Naïve Bayes (either with a Gaussian likelihood function or Bernoulli) and Logistic regression. One can also consider first reduce the dimension of the images say via principal components or SVD and then apply classification on the data to reduce the amount of computation.

5 Appendix

```
function [classes, params] = ...
    fullGaussianClassify(train0, train1, test)
%A full Bayes classifier for
%binary classification.
%
%train0, train1, and test1 are
%all sets of data with features as columns
%and observations as rows.
%
%train0 and train1 are training
%data from two different classes.
%
%test is the testing data.
%
%Returns the classifications of the
%testing data and the parameters of the
%models used.

var0 = cov(train0);
mean0 = mean(train0);
probs0 = mvnpdf(test, mean0, var0);

var1 = cov(train1);
mean1 = mean(train1);
probs1 = mvnpdf(test, mean1, var1);

classes = probs1 > probs0;

params = {mean0, var0, mean1, var1};
```

```
function [classes, params] = ...
    naiveBayesClassify(train0, train1, test)
%A naive Bayes classifier
%for binary classification.
%
%train0, train1, and test1 are
%all sets of data with features as columns
%and observations as rows.
%
%train0 and train1 are training
%data from two different classes.
%
%test is the testing data.
%
%Returns the classifications of the
%testing data and the parameters of the
%models used.

var0 = var(train0);
mean0 = mean(train0);

probs0 = zeros(size(test));
for i = 1:size(test, 1)
    probs0(i, :) = normpdf(test(i, :), ...
        mean0, var0);
end
probs0 = prod(probs0, 2);

var1 = var(train1);
mean1 = mean(train1);

probs1 = zeros(size(test));
for i = 1:size(test, 1)
    probs1(i, :) = normpdf(test(i, :), ...
        mean1, var1);
end
probs1 = prod(probs1, 2);

classes = probs1 > probs0;

params = {mean0, var0, mean1, var1};
```

```
function [accuracy, errorBar, parameters] ...
    = crossValTest(classFunction, data0, data1, K)
%Performs K-fold crossvaliation
%(using K=10 by default)
%
%classFunction is a function
%handle to use for classifying
%(such as naiveBayesClassify or
%fullGaussianClassify). The function
%must take three parameters: two
%training data sets from different classes,
%and one set of test data.
%
%Returns the mean accuracy, the
%standard deviation of that accuracy (to use
%as an error bar in a graph), and
%the mean parameters of the models
%learned.

if nargin < 4
    K = 10;
end

inds0 = crossvalind('KFold', size(data0, 1), K);
inds1 = crossvalind('KFold', size(data1, 1), K);

accuracies = zeros(K, 1);

for i = 1:K
    train0 = data0(inds0 ~= i, :);
    test0 = data0(inds0 == i, :);
    train1 = data1(inds1 ~= i, :);
    test1 = data1(inds1 == i, :);
    test = [test0; test1];

    truthClasses = [zeros(size(test0, 1),...
        1); ones(size(test1, 1), 1)];
    [guessClasses params] = ...
        classFunction(train0, train1, test);
    numCorrect = length(find...
        (truthClasses == guessClasses));
    accuracies(i) = numCorrect...
        /(size(truthClasses, 1));

    if i == 1
        parameters = params;
    else
        for j = 1:length(parameters)
            parameters{j} = ...
                parameters{j} + params{j};
        end
    end
end

accuracy = mean(accuracies);
errorBar = std(accuracies);

for i = 1:length(parameters)
    parameters{i} = parameters{i}/K;
end
```

```
%Simulates the data sets for Assignment 1

numSamples = [20, 40, 100, 300, 1200];
mu0 = [-2, 0];
mu1 = [2, 0];
sigma_0 = [2, 0; 0, 3];
sigma_1 = [2, -1.8; -1.8, 3];

class0 = cell(1, 10);
class1 = cell(1, 10);

for i = 1:5
    class0{i} = mvnrnd(mu0, sigma_0, numSamples(i));
    class1{i} = mvnrnd(mu1, sigma_0, numSamples(i));
end
for i = 6:10
    class0{i} = mvnrnd(mu0, sigma_1, numSamples(i - 5));
```

```
    class1{i} = mvnrnd(mu1, sigma_1, numSamples(i - 5));
end
```