CSE 881: Data Mining (Fall 2016 Homework 5)
Due date: Oct 11, 2016 (before midnight).

A soft copy of your homework must be submitted via handin. All submitted homework must be your own work.

1. Consider the two decision trees shown in Figure 1 for a binary classification problem. Assume the classes are denoted as + and −, respectively. Suppose there are four binary attributes in the data, A, B, C, and D. The counts shown in the leaf nodes of the tree correspond to the number of training examples assigned to the nodes. Assume that the decision tree classifier assigns the majority class of training examples as the class label of each leaf node.

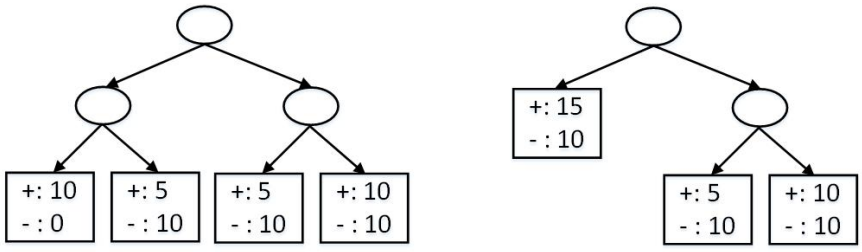

Figure 1: Two candidate decision trees

   (a) Draw the confusion matrix for both trees on the training data. A confusion matrix is a table that summarizes the number of examples correctly or incorrectly predicted by the model. For example:

		Predicted	
		+	−
Actual	+	n_{++}	n_{+-}
	−	n_{-+}	n_{--}

   In the above table, n_{+-} is the number of positive examples incorrectly predicted as negative class.

   (b) Calculate the training error rate of both decision trees. Which tree has a lower training error?

   (c) Apply the minimum description length principle to determine which tree should be preferred.

2. A linear classifier in SVM can be mathematically expressed as $f(\mathbf{x}) = \sum_i w_i \Phi_i(\mathbf{x})$, where each $\Phi_i(\mathbf{x})$ is a feature function of the original feature

set $\mathbf{x}$. Note that the transformed feature set $\{\Phi_i(\mathbf{x})\}$ could be infinite-dimensional. The predicted class for a test instance $\mathbf{x}$ is determined as follows:

$$\hat{y} = \begin{cases} +1, & \text{if } f(\mathbf{x}) \geq 0; \\ -1, & \text{otherwise.} \end{cases}$$

For each binary classification data set described below, state whether it can be perfectly classified by a linear classifier by choosing appropriate feature functions. Restrict the feature functions to polynomial expansions of the original attributes, e.g., $\Phi_1(\mathbf{x}) = x_1 x_2^2$ or $\Phi_2(\mathbf{x}) = x_2 x_3 x_4$, where x_1, x_2, x_3, and x_4 are part of the original attributes. If the answer is yes, write the mathematical expression for the linear classifier $f(\mathbf{x})$. Identify the feature functions $\Phi_i(\mathbf{x})$ as well as the parameters $\mathbf{w}$ in your expression.

(a) A data set with 4 continuous-valued features x_1, x_2, x_3, and x_4. The class label is $+1$ if the product of the x_1 and x_2 is greater than or equal to the product of x_3 and x_4; otherwise, it is -1.

(b) A data set with 4 continuous-valued features x_1, x_2, x_3, and x_4. The class label is $+1$ if at least one of the features is greater than 10; otherwise, it is -1.

(c) A data set with 2 continuous-valued features x_1 and x_2. The class label is $+1$ if the exponential value of the difference between $x_1 - x_2$ is greater than 100 (i.e., $\exp[x_1 - x_2]$); otherwise, it is -1.

(d) A data set with 2 binary features, x_1 and x_2, whose class label y is determined as follows (this is similar to the exclusive OR binary operator using -1 instead of 0):

x_1	x_2	y
1	1	-1
1	-1	1
-1	1	1
-1	-1	-1

3. In this exercise, you will apply both linear and nonlinear classifiers to the email spam classification problem. First, you need to download the email spam data from `https://archive.ics.uci.edu/ml/datasets/Spambase`. The data set is stored in a file named `spambase.data`. You can load the file directly into matlab as follows:

```
A = load('spambase.data');
N = size(A, 1);
seed = 10;                % seed for random number generator
rng(seed);                % for repeatability of your experiment
A = A(randperm(N),:);     % this will reshuffle the rows in matrix A
```

The reshuffling of rows is needed since the first 1813 rows in $\mathbf{A}$ belong to class 1 (spam) and the remaining rows belong to class 0 (non-spam). After

the data has been successfully reshuffled, you should split the predictor attributes and the class label into a separate matrix $\mathbf{X}$ and a vector $\mathbf{y}$. Note that the class label is given in the last column of the matrix $\mathbf{A}$. You need to compare the performance of the linear and non-linear support vector machine classifiers on this data set using nested cross-validation. Use k=10 for the outer loop (classifier evaluation) and k=5 for the inner loop (model selection) of the nested cross-validation procedure. For the outer loop cross-validation, you can use Matlab's `crossvalind` function[1] to create the training and test sets for different folds.

(a) **Linear support vector machine[2].** Linear SVM is designed to solve the following constrained optimization problem:

$$\min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2} + \lambda \sum_{i=1}^{N} \xi_i$$
$$\text{s.t.} \quad \forall i : y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i$$

You can use the `fitclinear` function in Matlab to construct a linear SVM model. You need to tune the hyper-parameter λ for linear SVM using 5-fold cross-validation, which can be done as follows:

```
lambda = logspace(-4,3,11);     % create a set of candidate lambda values
SVMmodel = fitclinear(Xtrain, Ytrain, 'Kfold', 5, 'Learner', 'svm', 'Lambda', lambda);
foldNumber = 3;             % to examine the model created for the 3rd fold
model.Trained{foldNumber}
ce = kFoldLoss(model)       % to examine the classification error for each lambda
bestIdx = min(ce);          % identify the index of lambda with smallest error
bestLambda = Lambda(bestIdx);
```

Once you have learned the best lambda, re-train the model using the entire training set and apply it to the test set. This can be done as follows:

```
SVMmodel = fitclinear(Xtrain, Ytrain, 'Learner', 'svm', 'Lambda', bestLambda);
pred = predict(SVMmodel, Xtest);
```

You need to repeat this for each pair of training and test sets created in the outer fold of the nested cross-validation procedure (for classifier evaluation). Report the classification error and confusion matrix obtained by linear SVM. You can use `classperf` to obtain the confusion matrix and calculate the classification error. For example:

```
cp = classperf(Ytest);
classperf(cp,pred);
cp.DiagnosticTable      % to show the confusion matrix
cp.ErrorRate            % to show the classification error
```

[1] See the example given in `https://www.mathworks.com/help/bioinfo/ref/crossvalind.html`.

[2] `https://www.mathworks.com/help/stats/fitclinear.html`

Note that you should calculate the error and confusion matrix for the entire data set after the nested cross-validation.

(b) **Nonlinear support vector machine**[3]**.** You can use the `fitcsvm` function in Matlab to construct a nonlinear SVM model. For this exercise, you will use Gaussian radial basis function as the kernel function and nested cross-validation to evaluate the classifier as well as determining its hyper-parameters. The Gaussian kernel function is defined as follows:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left[-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma}\right]$$

The hyper-parameters that need to be tuned when building the model include the kernel scale (width), σ, and the box constraint, λ (see the preceding linear SVM formulation).

```
lambda = logspace(-3,3,11);     % create a set of lambda values
sigma = logspace(-3,3,11);      % create a set of kernel scale values
cvloss = zeros(11,11);          % stores the CV error for each (lambda,sigma) pair
for i=1:11
    for j=1:11
        SVMmodel = fitcsvm(Xtrain, Ytrain, 'KernelFunction','RBF',...
                    'KernelScale',sigma(j),'BoxConstraint',lambda(i),'Kfold', 5);
        cvloss(i,j) = = kfoldLoss(SVMmodel);
    end;
end;
```

Use the cvloss matrix to identify the best pair of lambda and sigma hyper-parameters, (bestLambda,bestSigma), to retrain the model on the entire training set and apply it to the test (for each given outer fold).

```
SVMmodel = fitcsvm(Xtrain, Ytrain, 'KernelFunction','RBF',...
                   'KernelScale',bestSigma,'BoxConstraint',bestLambda);
pred = predict(SVMmodel, Xtest);
```

Similar to part (a), you need to repeat this for each test set Xtest created in the outer fold of the nested cross-validation procedure (for classifier evaluation). Report the classification error and confusion matrix obtained by nonlinear SVM and compare it against linear SVM. Which method is more effective?

In addition to the solution for each question above, you should attach your Matlab script for the linear and nonlinear SVM.

[3] `http://www.mathworks.com/help/stats/fitcsvm.html`