CS178 Homework #4 Solution

Machine Learning & Data Mining: Winter 2015

Problem 1: Decision Trees

You can most easily do this by hand, but since I have to type it I will put it in Matlab format:

```
Xy = [0 0 1 1 0 -1
      1 1 0 1 0 -1
      0 1 1 1 1 -1
      1 1 1 1 0 -1
      0 1 0 0 0 -1
      1 0 1 1 1  1
      0 0 1 0 0  1
      1 0 0 0 0  1
      1 0 1 1 0  1
      1 1 1 1 1 -1 ];
ep=1e-12;            % move values away from log(0)=-infty
X = Xy(:,1:end-1); Y=Xy(:,end);
```

(a) Calculate the entropy of the class variable y

```
p = mean(Y>0);
Hy = -(p*log2(p) + (1-p)*log2(1-p));
% .971 bits
```

(b) Calculate the information gain for each feature x_i. Which feature should I split on first?

```
for i=1:5,
 idx = X(:,i)>0;
 if (sum(idx)==0 || sum(~idx)==0) IG(i)=0; continue; end;
 p1 = mean( Y(idx)>0 )+ep; p0 = mean(Y(~idx)>0)+ep; a = mean(idx)+ep;
 IG(i) = Hy + a*(p1*log2(p1)+(1-p1)*log2(1-p1))+(1-a)*(p0*log2(p0)+(1-p0)*log2(1-p0));
end;
IG,
% IG = 0.0464    0.6100    0.0058    0.0913    0.0058
% Pick feature # 2
```

(c) Draw the complete decision tree that will be learned from these data.

```
% Splitting on feature 2 divides the data:
Xy = [1 1 0 1 0 -1
      0 1 1 1 1 -1
      1 1 1 1 0 -1
      0 1 0 0 0 -1
      1 1 1 1 1 -1];
% On this branch we can always predict "-1"
Xy = [0 0 1 1 0 -1
      1 0 1 1 1  1
      0 0 1 0 0  1
      1 0 0 0 0  1
      1 0 1 1 0  1 ];
% On this branch, we'll need to split; repeating, we find that the next best is feature #1
% You can pretty much see this by inspection; or you can compute it if you prefer.

Xy = [1 0 1 1 1  1
      1 0 0 0 0  1
```

```
      1 0 1 1 0  1 ];
% On this branch we just predict "1"

Xy = [0 0 1 1 0 -1
      0 0 1 0 0  1 ];
% On this branch we'll need to split again; by inspection split on feature #4 and predict 1 or -1

% So the final rule is:
% if (long) discard
% else
%   if (known) read
%   else
%     if (has 'grade') discard
%     else read
```

Problem 2: Decision Trees in Kaggle

```
X   = load('kaggle.X1.train.txt');        % load the data
Y   = load('kaggle.Y.train.txt');         %   and target values
[Xt Xv Yt Yv] = splitData(X,Y,.8);        % split out some validation data
```

```
% (a) Train a decision tree and compute its accuracy
dt = treeRegress(Xt,Yt, 'maxDepth',20);   % train a decision tree w/ default params
[mse(dt,Xt,Yt), mse(dt,Xv,Yv)],           % compute training and validation errors
% ans =   0.0361    0.7140
```

```
for d=0:15
  dt = treeRegress(Xt,Yt,'maxDepth',d);
  [d, mse(dt,Xt,Yt), mse(dt,Xv,Yv)],
end;
% ans =       0    0.6908    0.7183       % Simplest model
% ans =  1.0000    0.5579    0.5741       % (Underfitting...)
% ans =  2.0000    0.5051    0.5198
% ans =  3.0000    0.4727    0.4837
% ans =  4.0000    0.4520    0.4656
% ans =  5.0000    0.4343    0.4555
% ans =  6.0000    0.4180    0.4466
% ans =  7.0000    0.3973    0.4355       % Best validation
% ans =  8.0000    0.3776    0.4392
% ans =  9.0000    0.3533    0.4473
% ans = 10.0000    0.3252    0.4675
% ans = 11.0000    0.2920    0.4903       % (Overfitting...)
% ans = 12.0000    0.2558    0.5131
% ans = 13.0000    0.2170    0.5431
% ans = 14.0000    0.1788    0.5880
% ans = 15.0000    0.1434    0.6236       % Most complex model
```

```
for n=2.^[3:12],
  dt = treeRegress(Xt,Yt,'minParent',n, 'maxDepth',20);
  [log2(n), mse(dt,Xt,Yt), mse(dt,Xv,Yv)],
end;
% ans = 3.0000    0.0685    0.6887        % Most complex model
% ans = 4.0000    0.1145    0.6447
```

```
% ans = 5.0000    0.1746    0.5871
% ans = 6.0000    0.2358    0.5261        % (Overfitting...)
% ans = 7.0000    0.2959    0.4763
% ans = 8.0000    0.3482    0.4407
% ans = 9.0000    0.3818    0.4302        % Best model
% ans = 10.0000    0.4067    0.4355
% ans = 11.0000    0.4226    0.4438       % (Underfitting...)
% ans = 12.0000    0.4471    0.4647       %
%    (at n=2^16 you will get the same as depth 1)
```

```
% (d) I'll pick minParent = 9
dt = treeRegress(X,Y,'minParent', 9 );        % build classifier on full data
Xe = load('kaggle.X1.test.txt');
Ye = predict(dt, Xe);

% Now, output to a file for upload
fh = fopen('kaggle_dtree.csv','w');
fprintf(fh,'ID,Prediction\n');
for i=1:length(Ye), fprintf(fh,'%d,%d\n',i,Ye(i)); end;
fclose(fh);
% and upload the file to see how it does on Kaggle.
```

Problem 3-1: Random Forests

```
% (a) Let's train all our bagged decision trees first, then test with different #s
rf = cell(1,25);
YtHat = zeros(size(Yt,1),25);                % we'll just make the predictions at
YvHat = zeros(size(Yv,1),25);                %   the same time to save looping

for l=1:25,                    % (This can take a while!)
  [Xi Yi] = bootstrapData(Xt,Yt,size(Xt,1)); % bootstrap sample for this learner
  rf{l} = treeRegress(Xi,Yi, 'maxDepth',15, 'nFeatures',60); % train & save learner
  YtHat(:,l) = predict(rf{l},Xt);            % predict on training
  YvHat(:,l) = predict(rf{l},Xv);            %   and validation
end;

% Now predict using various numbers of bagged learners:
for l=[1 5 10 25],
  mseT = mean( (Yt - mean(YtHat(:,1:l),2)).^2 );   % compute MSE of ensemble avg
  mseV = mean( (Yv - mean(YvHat(:,1:l),2)).^2 );   %   on training & validation data
  [l, mseT, mseV],
end;
% ans = 1.0000    0.3275    0.6567           % has overfit to a subset of the data...
% ans = 5.0000    0.1722    0.4295
% ans = 10.0000    0.1535    0.4029
% ans = 25.0000    0.1439    0.3823          % validation error is still low!
%  Notice the validation error is lower than we got for any single tree!
```

```
(b) Now let's do it on the full data
nEnsemble = 25;
Ye = zeros(size(Xe,1),1);
for l=1:nEnsemble,                     % (This can take a while!)
  [Xi Yi] = bootstrapData(X,Y,size(X,1));      % bootstrap sample for this learner
  rf{l} = treeRegress(Xi,Yi, 'maxDepth',20, 'nFeatures',60);  % train next tree
```

```
  Ye = Ye + predict( rf{l}, Xe);      % build and predict
end;
Ye = Ye / nEnsemble;                            % take average value

% Now, output to a file for upload
fh = fopen('kaggle_rforest25.csv','w');
fprintf(fh,'ID,Prediction\n');
for i=1:length(Ye), fprintf(fh,'%d,%d\n',i,Ye(i)); end;
fclose(fh);
% and upload the file to see how it does on Kaggle.
```

Problem 3-2: Gradient Boosting

```
% (a) Let's train all our boosted trees first, then test with different #s
rf = cell(1,25);
YtHat = zeros(size(Yt,1),25);                % we'll just make the predictions at
YvHat = zeros(size(Yv,1),25);                %   the same time to save looping

mn = mean(Yt);
Yi = Yt - mn;
for l=1:25,                   % (This is faster than the bagging loop)
  rf{l} = treeRegress(Xt,Yi, 'maxDepth',3);  % train & save learner
  YtHat(:,l) = predict(rf{l},Xt);            % predict on training
  YvHat(:,l) = predict(rf{l},Xv);            %   and validation
  Yi = Yi - YtHat(:,l);                      % keep residual for next member
end;

% Now predict using various numbers of bagged learners:
for l=[1 5 10 25],
  mseT = mean( (Yt - mn - sum(YtHat(:,1:l),2)).^2 );   % compute MSE of ensemble total
  mseV = mean( (Yv - mn - sum(YvHat(:,1:l),2)).^2 );   %   on training & validation data
  [l, mseT, mseV],
end;
% ans = 1.0000    0.4727    0.4837           % underfitting (simple learner, depth 3)
% ans = 5.0000    0.4178    0.4327
% ans = 10.0000    0.3967    0.4179
% ans = 25.0000    0.3612    0.4089          % validation error getting low!
%  Notice the validation error is already about as low as our best single tree!

% You can repeat the training procedure on the full data, and try it on Kaggle
```