Decision Tree and Genetic Algorithm

Pattern Recognition/ECE 471

William Willie Wells
wwells4@utk.edu

Abstract

Decision tree learning sprang up from the desire to classify data that is not easily or naturally represented quantitatively or where distance between features has no intuitive meaning. This is one of two non-statistical approaches to pattern classification. Tree construction involves decisions at each node that have a finite number of responses with the binomial case being the most popular and easiest to implement. Overfitting can occur if tree splitting is not terminated before each training sample has its own leaf, but if splitting is terminated too soon the accuracy of the classifier is reduced. Chi-squared statistics, impurity functions, and minimum description length are some algorithms used to determine when to terminate the splitting procedure. Though developed for nonmetric data decision trees are used to classify metric data as well. Data sets with one or more missing feature value can still result in a classification using decision tree learning by using a subset (consisting of the features with values) of the split path that is most probable given the known features and assigning the unknown features the most probable value. Using Pima data sets, the Pima data set with missing feature values resulted in higher accuracies values that the Pima data set denoted as the test set. With variations in decision tree parameters such as split criterion, classification cost function, and prior probability the Pima data set with missing feature values have an average accuracy of 83.84% whereas the Pima data set denoted as the test set have an average accuracy of 73.87%.

Introduction:

Feature spaces where there is no natural notion of similarity or ordering within a single feature make it difficult to create a measure of distance between such unlike features. Thus a different classification approach is necessary to classify such samples. The two common non-statistical approaches are decision trees and syntactic approach. Classification trees and regression trees are a further division of decision trees. If the response variable of a feature space consists of a binary classification such as that of the Pima data set a standard classification tree is usually used. Starting at the parent node a decision tree compares one feature to a discrete relational value such as npreg < 3 or color = green, if the statement evaluates to true you proceed to traditionally the left hand child node and otherwise to the right hand child node where another comparison is made using another feature until a conclusion of what class membership the sample belongs to is ultimately made. These final concluding branches end in leaf nodes with a stated classification. Several mathematical algorithms have been developed to decide when to make the classification conclusion including various impurity functions, minimum description length, and chi-squared statistics. For deficient samples containing missing data the highest decision nodes that contain available data from the sample are used to make decisions about the sample and parent missing nodes are approximated to the value most strongly correlated with the sample. The performance of a decision tree is measured using its accuracy, how many samples are accurately classified. [(1)(2)]

Technical Approach:

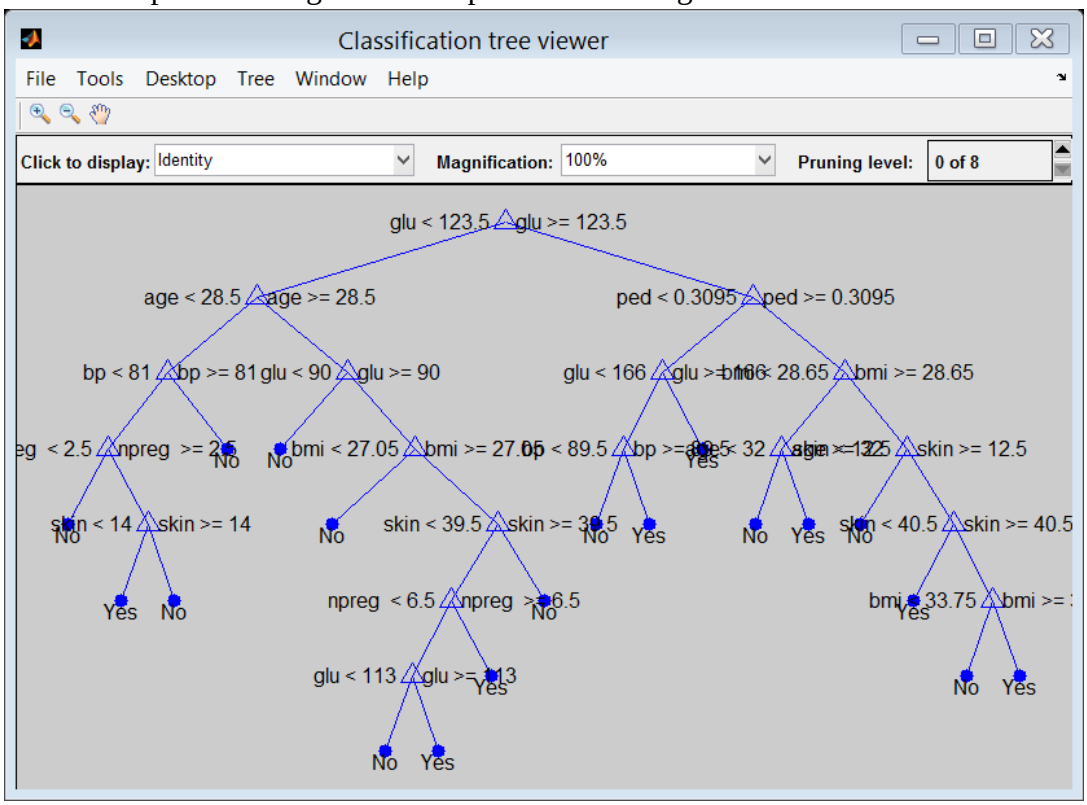
The Matrix Laboratory numerical computing software produced by MathWorks includes a function labeled classregtree which performs either classification or regression as determined by the type of data or by the programmer. All trees produced by classregtree are binary trees. There are 14 different parameters that can be modified for classification trees some of which are interrelated. The value choices for the parameter *splitcriterion* are gdi (Gini's diversity index, the default value), twoing, and deviance. This parameter was changed to deviance for the second decision tree used.

For the parameter *cost* the value choices are square matrices with each diagonal composed of 0's. A zero-one square matrix is the default value. This parameter was changed to cost = [0 1.4; 0.6 0] for the third decision tree utilized. The value choices for the parameter *priorprob* are empirical (the default value) determined from class frequencies in the training set, equal, or as a vector of probabilities. This parameter was changed to equal for the fourth decision tree created.

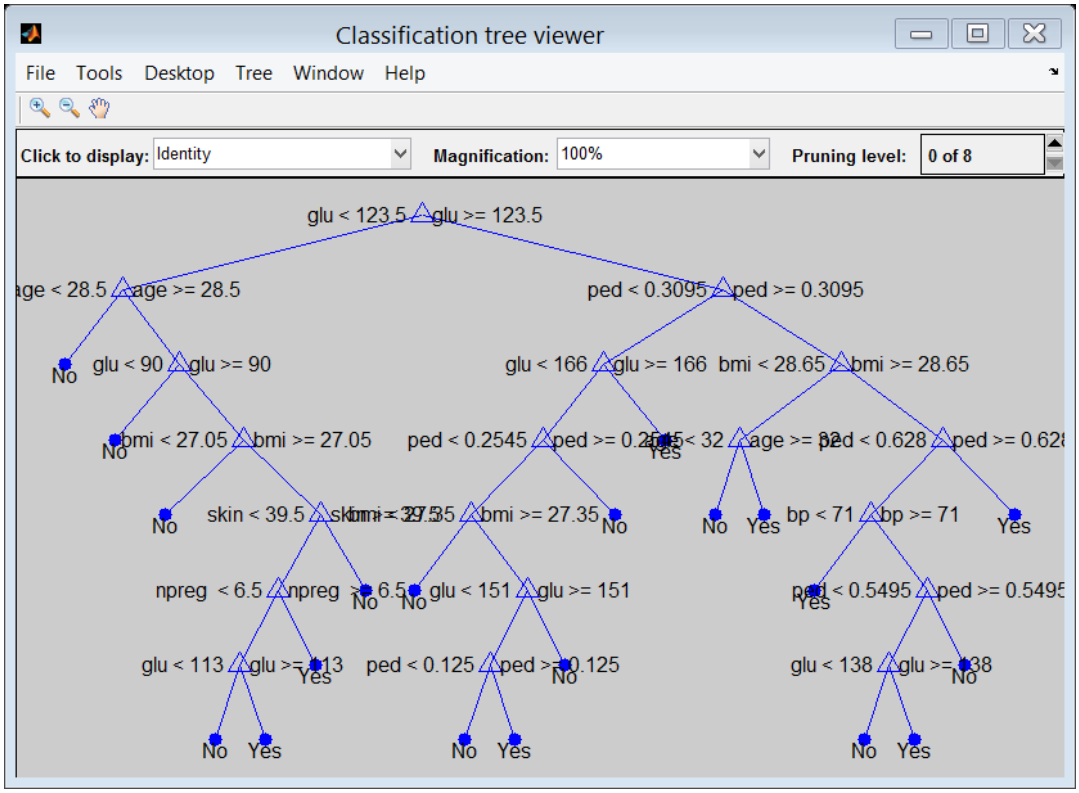
After the respective trees were produced the MATLAB function eval was used to classify the pima.tr, the pima.te, and the pima.tr2 data sets. The MATLAB functions strcmp and mean were used to compute the accuracy of each classification.

Experiments and Results

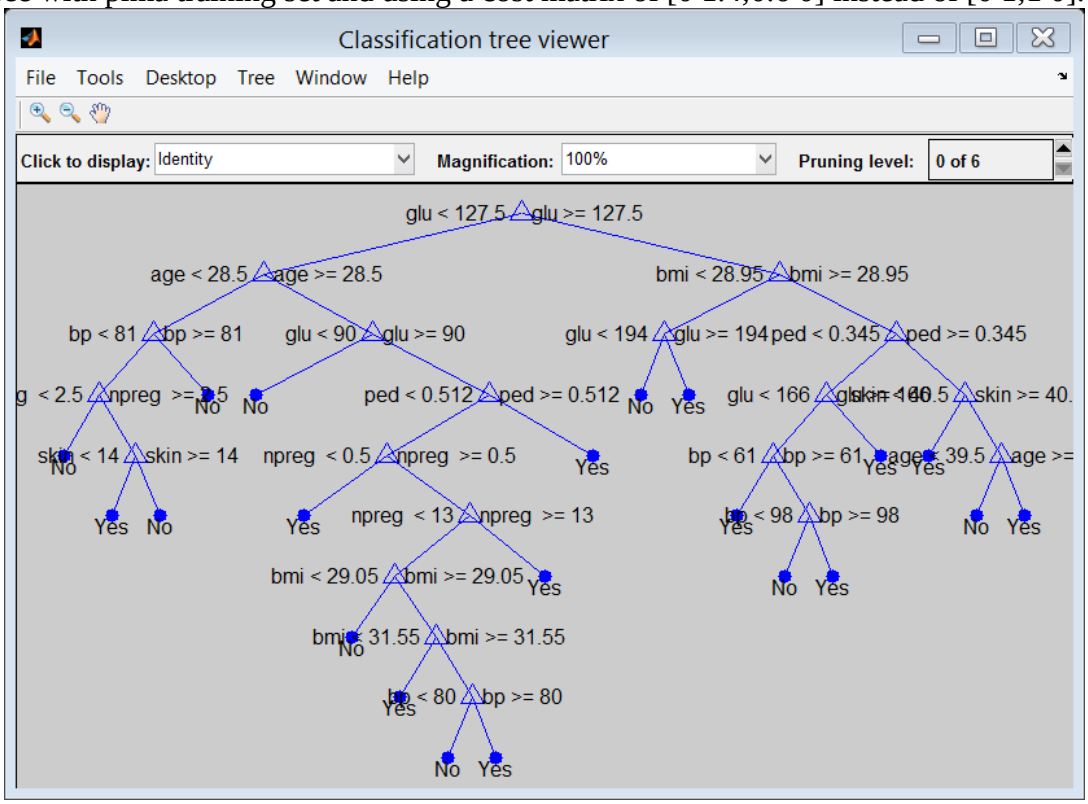
Decision tree with pima training set and no parameters changed:



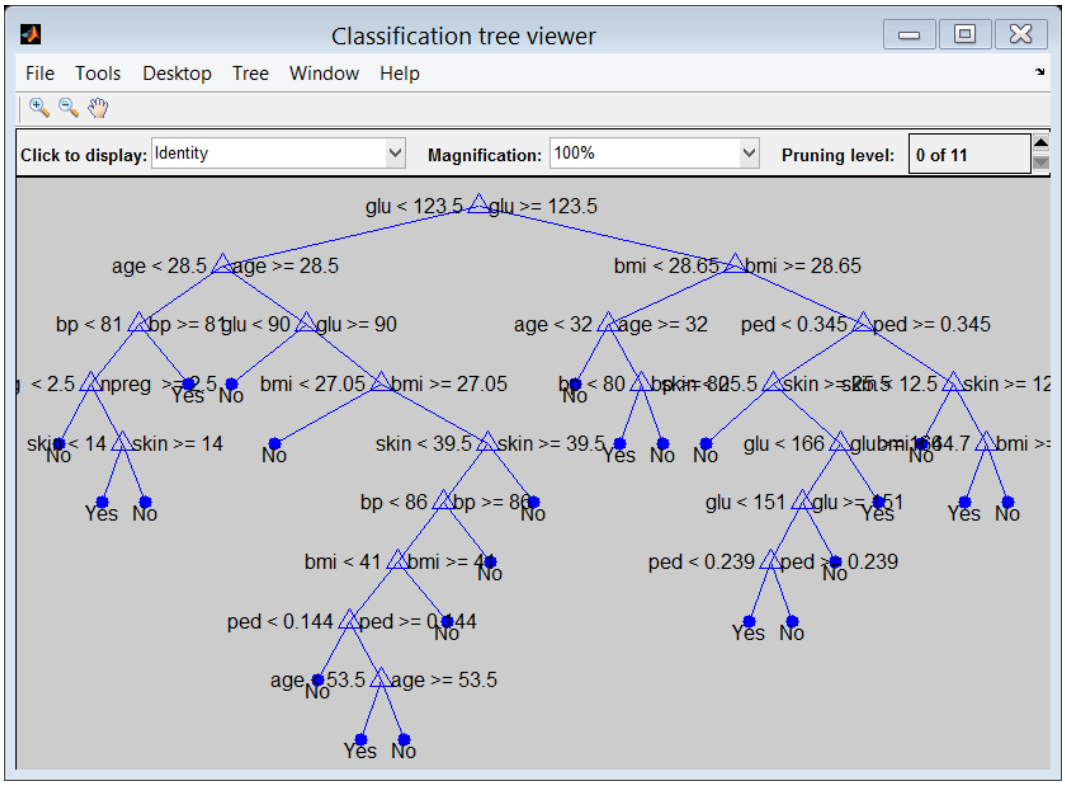
Decision tree with pima training set and with maximum deviance reduction split criterion instead of Gini's diversity index:



Decision tree with pima training set and using a cost matrix of [0 1.4;0.6 0] instead of [0 1;1 0]:



Decision tree with pima training set and with equal prior probability instead of distribution based prior probabilities:



Classification accuracies:

classifier/data	tree1	tree2	tree3	tree4
pima1	0.905	0.925	0.9	0.93
pima2	0.7651	0.7711	0.7319	0.6867
pima3	0.84	0.8467	0.8167	0.85

Conclusion:

The decision tree with equal prior probabilities produced a higher accuracy for the set it was trained with and the set with missing data but performed the worse of all the variant trees for the set designated as the test set. The change of deviant split criterion produced the highest accuracy for the test set and the second highest accuracy for the other two sets and is clearly the best parameter change of the changes that were selected. The change of cost function resulted in a lower accuracies for all sets. Though there may be a cost value that would perform better than zero-one cost for all sets. The decision trees resulted in higher accuracies for the original test set but none produced a 100% accuracy which would be a result of overfitting. The set with missing data resulted in higher accuracies than the test set.

References:

1 R. O. Duda, P. E. Hart, D. G. Stork, *Pattern Classification*, 2nd Edition, John Wiley, 2001.

2 http://web.eecs.utk.edu/~qi/ece471-571/lecture22_syntactic.pdf

3 http://www.mathworks.com/help/stats/classregtree.html

Appendix:

```
% proj5.m - Purpose: Implement decision tree classification
%                    on Pima data sets
%
% Author: William Willie Wells
%
% Created: November 2013
%

% feature names
feature={'npreg ' 'glu' 'bp' 'skin' 'bmi' 'ped' 'age'};

   % Read in data from pima.tr, pima.te, and pima.tr2 %
% read in individual columns of pima.tr and reconstruct n-by-7 feature space
[f1,f2,f3,f4,f5,f6,f7,class1]=textread('pima.tr','%f %f %f %f %f %f %f %s','headerlines',1);
pima=[f1 f2 f3 f4 f5 f6 f7];

% read in individual columns of pima.te and reconstruct n-by-7 feature space
[f1,f2,f3,f4,f5,f6,f7,class2]=textread('pima.te','%f %f %f %f %f %f %f %s','headerlines',1);
pima2=[f1 f2 f3 f4 f5 f6 f7];

% read in individual columns of pima.tr2
[f1,f2,f3,f4,f5,f6,f7,class3]=textread('pima.tr2','%f %f %s %s %s %f %f %s','headerlines',1);

% convert cells to symbols of columns with 'NA' for comparison
f3=sym(f3);
f4=sym(f4);
f5=sym(f5);

% convert 'NA' to nan for evaluation
for ci=1:1:length(f3)
   if f3(ci)=='NA'
      f3(ci)=nan;
   end
   if f4(ci)=='NA'
      f4(ci)=nan;
   end
   if f5(ci)=='NA'
      f5(ci)=nan;
   end
end

% convert symbols to double for evaluation
f3=double(f3);
f4=double(f4);
f5=double(f5);

% reconstruct n-by-7 feature  space
pima3=[f1 f2 f3 f4 f5 f6 f7];

   % Train Decision Trees %
% using pima.tr train first decision tree and display the tree
Tree1=classregtree(pima,class1,'method', 'classification','names', feature);
view(Tree1)

% using pima.tr train decision tree with different split criterion and display the tree
Tree2=classregtree(pima,class1,'method', 'classification','names', feature,'splitcriterion','deviance');
view(Tree2)

% using pima.tr train decision tree with different cost function and display the tree
C=[0 1.4;.6 0];
Tree3=classregtree(pima,class1,'method', 'classification','names', feature,'cost',C);
view(Tree3)

% using pima.tr train decision tree with different prior probability and display the tree
```

```
Tree4=classregtree(pima,class1,'method', 'classification','names', feature,'priorprob','equal');
view(Tree4)

    % Classify using decision trees %
% evaluate and classify using decision trees
pima1Tree1=eval(Tree1,pima);
pima2Tree1=eval(Tree1,pima2);
pima3Tree1=eval(Tree1,pima3);
pima1Tree2=eval(Tree2,pima);
pima2Tree2=eval(Tree2,pima2);
pima3Tree2=eval(Tree2,pima3);
pima1Tree3=eval(Tree3,pima);
pima2Tree3=eval(Tree3,pima2);
pima3Tree3=eval(Tree3,pima3);
pima1Tree4=eval(Tree4,pima);
pima2Tree4=eval(Tree4,pima2);
pima3Tree4=eval(Tree4,pima3);

  % Compute Accuracies %
% calculate classification accuracies
cTrue=zeros(3,4);
cTrue(1,1)=mean(strcmp(pima1Tree1,class1));
cTrue(1,2)=mean(strcmp(pima1Tree2,class1));
cTrue(1,3)=mean(strcmp(pima1Tree3,class1));
cTrue(1,4)=mean(strcmp(pima1Tree4,class1));
cTrue(2,1)=mean(strcmp(pima2Tree1,class2));
cTrue(2,2)=mean(strcmp(pima2Tree2,class2));
cTrue(2,3)=mean(strcmp(pima2Tree3,class2));
cTrue(2,4)=mean(strcmp(pima2Tree4,class2));
cTrue(3,1)=mean(strcmp(pima3Tree1,class3));
cTrue(3,2)=mean(strcmp(pima3Tree2,class3));
cTrue(3,3)=mean(strcmp(pima3Tree3,class3));
cTrue(3,4)=mean(strcmp(pima3Tree4,class3));
cTrue
```