Pattern Recognition: Assignment 6

Due on Tuesday, June 12 2012, 23:59

Bernhard X. Kausler (TA), Summer Term 2012

patternrecognition@hci.iwr.uni-heidelberg.de

`http://hci.iwr.uni-heidelberg.de/MIP/Teaching/pr/`

Random Forests

The focus of this exercise is on Decision Trees and Random Forest classification. You will start with investigating the feature space partitioning of a single classification tree. Then you will calculate the optimal split value using Gini impurity. Finally you will apply Random Forests to the MNIST digits dataset.

Some parts of this exercise require proofs and derivations. You can either type them in Latex/LibreOffice and submit a pdf (preferred), or write them up by hand and give them to your teaching assistant.

Prob. 1: Decision Trees

In this problem we will use the `ClassficationTree` code contained in the Matlab statistics toolbox to gain insight into the partitioning of feature space by a decision tree.
We work with the–by now–well known student scores dataset, differently formatted for your convenience (`student-scores.mat`).

(a) Plot the feature space partitioning (3 points)

Train a ClassificationTree on the training data. Set both options MinLeaf and MinParent to 1. This will encourgae pure leaf nodes.
Implement a function `plot_partitioning( tree, datax, datay )` that plots the data points together with regions corresponding to the leaf nodes of the decision tree. (Don't confuse these areas with the decision boundaries of the tree!) See Fig. 1 for an example. Use the jet colormap. (suggested commands: ClassificationTree.train, predict, imagesc)

(b) Variance of decision trees (2 points)

Now, split the training set into two subsets comprising the first 106 samples and the remaining 107. Train a decision tree on both and plot both partitionings. What do you learn about the variance of decision trees? (hint: read up on the bias-variance trade-off of classifiers)

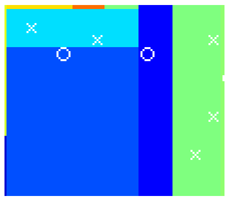

Figure 1: sample partitioning of feature space

(c) A clearer visualization (3 points)

We want to improve the visualization of the feature space partitioning. Instead if plotting colored areas, we will display the border lines of the partitioning (cf. Fig. 2). Implement a simple edge detector for horizontal resp. vertical lines (use a convolution kernel together with the `conv2` command). Plot the partitioning of the noise-free data again. Further reading: `http://www.imageprocessingbasics.com/image-convolution-filters/`

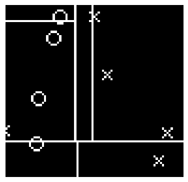

Figure 2: sample partitioning of feature space: visualization using edges

Prob. 2: Splitting Criterion

(a) Compute the Gini impurity (2 points)

Implement a Matlab function for computing the Gini impurity: `gi = get_gini_impurity(labels)`. `gi` is a scalar and `labels` is a vector of 1 and 2 (suggested command: unique).

(b) Find the best split value (2 points)

Given that the input feature space is 1D, implement a Matlab function `[giDelta] = get_gini_impurity_decrease (feature, labels, splitValue)` that computes the decrease of the Gini impurity at a given split value `splitValue`. `giDelta` and `splitValue` are scalars; `feature` and `labels` are vectors of integers or doubles. Load an example 1D dataset from `data_gini_1d.mat`. Its first column contains the 1D feature and its second column the labels. Create a plot to visualize the decrease of the Gini impurity depending on the splitting value. Let the x axis represent the feature space and the y axis the decrease of the Gini impurity and plot all data points with respect to all suitable split values (suggested command: sort).

Prob. 3: Random Forests

In this exercise, you will investigate some theoretical properties of Random Forests and apply them to real life data sets. First of all, install the Random Forests toolbox for Matlab which contains two functions: vigraLearnRF, vigraPredictProbabilitiesRF. Read the documentation of those functions, e. g. type help vigraLearnRF in the Matlab console. Note that this library is for 64-bit Linux system only (as running in the IWR computer pool). The Random Forests toolbox is included as `rf_vigra_matlab_a64.zip`.

(a) Variance of decision trees (3 points)

The average of B independent and identically distributed (i.i.d.) random variables, each with variance σ^2, has variance σ^2/B. If the variables are simply i.d. (identically distributed, but not necessarily independent) with positive pairwise correlation ρ, show that the variance of their average is $\rho\sigma^2 + (1-\rho)\sigma^2/B$.
This formula depicts the underlying idea of Random Forests: a single decision tree shows a low bias, but a large variance. By reducing the correlation between the decision trees (built on bootstrap samples), the overall variance decreases for a sufficiently large value of B.

(b) Out-of-bag error estimate (3 points)

An important aspect of Random Forests is the use of out-of-bag (oob) samples: For each observation, the Random Forest predictors are computed by averaging only those trees which correspond to bootstrap samples not containing this observation. The oob error estimate is almost identical to that obtained by N-fold cross-validation; however, the oob error estimate is performed on-the-fly within the learning process. What is the approximate probability that an observation is out-of-bag? Explain why.

(c) Random Forests for classification (2 points)

Now using the MNIST digits dataset (`digit.mat`), train a Random Forest classifier with a varying number of trees (up to 250) on the training dataset (again five vs. non-five). Plot the oob error and test error[1] (on the test dataset) with respect to the number of trees. For training a Random Forests classifier, use the function vigraLearnRF and for predicting the class assignment probabilities of new data points, use the function vigraPredictProbabilitiesRF. Note that you need to set a proper probability cutoff to transform the predicted probabilities into labels.

Regulations

Please hand in the matlab code, figures and explanations (describing clearly which belongs to which). Non-trivial sections of your code should be explained with short comments, and variables should have self-explanatory names. Plots should have informative axis labels, legends and captions. Please enclose everything into a single PDF document (e.g. use the publish command of MATLAB for creating a LaTeX document and run latex, dvips and ps2pdf or copy and paste everything into an office document and convert to PDF). Please email the PDF to patternrecognition@hci.iwr.uni-heidelberg.de before the deadline specified below. You may hand in the exercises in teams of two people, which must be clearly named on the solution sheet (one email is sufficient). Discussions between different teams about the exercises are encouraged, but the code must not be copied verbatim (the same holds for any implementations which may be available on the WWW). Please respect particularly this rule, otherwise we cannot give you a passing grade.

[1] 1 minus the correct classification rate