Introduction to Machine Learning (67577)

Problem Set 5 – Decision Trees and Nearest Neighbors

Due: 18.6.2017

Submission guidelines

- The assignment is to be submitted via Moodle only.
- Files should be zipped to ex_N_First_Last.zip (In this case N stands for 5, First is your first name, Last is your last name. In English, of course).
- The .zip file size should not exceed 10Mb.
- All answers should be submitted as a single pdf file. We encourage you to typeset your answers (either LaTeX, Lyx or Word), but it is not mandatory. Note that if you choose to scan a handwritten solution, then answers written in incomprehensible handwriting or scanned in poor quality will be counted as wrong answers. Moreover, the scans should be integrated with the plots to a single coherent pdf file.
- All of your Python code files used in the practical question should be attached in the zip file. There should be one .py file, other then that there are no special requirements or guidelines for your code. Note that if your code doesn't run on the CS environment you will not get points for this question.

1 Suboptimality of ID3

Consider the following training set, where $\mathcal{X} = \{0,1\}^3$ and $\mathcal{Y} = \{0,1\}$:

$$((1,1,1),1)$$
$$((1,0,0),1)$$
$$((1,1,0),0)$$
$$((0,0,1),0)$$

Suppose we wish to use this training set in order to build a decision tree of depth 2 (i.e. we are allowed to ask two questions of the form "$x_i = 0$?" before deciding on the label)

1. (15 points) Suppose we run the ID3 algorithm we have learned in class, up to depth 2 (namely, we pick the root node and its children according to the algorithm, but instead of keeping on with the recursion, we stop

and pick leaves according to the majority label in each subtree). Assume that the subroutine used to measure the quality of each feature is based on the entropy function (so we measure the and that if two features get the same score, one of them is picked arbitrarily). See question 3 of this exercise for a reminder of the information gain. Show that the training error of the resulting decision tree is at least $\frac{1}{4}$.

2. (15 points) Find a decision tree of depth 2, which attains zero training error.

2 k-nearest neighbor

In the following questions you will consider a k-nearest neighbor decision rule using the Euclidean distance metric on a binary classification task. We assign the class of the test point to be the class of the majority of the k-nearest neighbors. Note that a point can be its own neighbor.

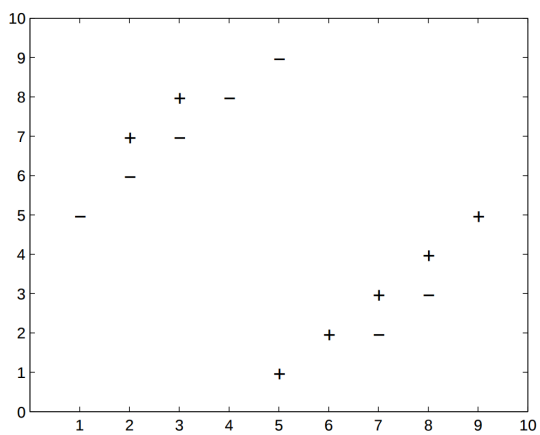

Figure 1: Dataset for k-NN binary classification task

1. (5 points) What value of k minimizes the training error for the dataset containing the points in figure 1? What is the resulting training error?

2. (5 points) What value of k minimizes leave-one-out cross-validation error for this dataset? What is the resulting error?

3. (5 points) Why might using too large values k be bad in this dataset? Why might too small values of k be bad?

4. (5 points) In Figure 1, sketch the 1-nearest neighbor decision boundary for this dataset.

3 Practical part

In this part you will implement the ID3 algorithm learned in class. From the course site you can download two data sets (train and validation) of 16 questions asked to different USA senators (i.e. our features). Each feature has 3 options yes (y), no (n) and unanswered question (u). Your goal is to create a decision tree which predicts whether the senator is a Republican or a Democrat (i.e. our label). Here are the exact instructions:

1. (15 points) Growing the tree: Implement the ID3 algorithm seen in class. You may limit your implementation to the case where features have finite range and the label is binary. Prepare code for (at least) one of the three gain functions listed below:

   - **Information gain**. Information gain measures the expected reduction in entropy - the uncertainty of the distribution - after a split according to some specific feature A. We shall want to choose the feature which will maximize the expected reduction in the uncertainty (entropy) of the resulting distribution. If our label set has $[1, ..., c]$ different values the entropy of the sample set is defined as $H(S) = \sum_{i=1}^{c} -p_i log p_i$ Where p_i is the fraction of the samples that were labeled "i".
     The information gain, $Gain(S; A)$ of a feature A, relative to a collection of samples S, is defined as $Gain(S; A) = H(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} H(S_v)$
     Where $values(A)$ is the set of all possible values for feature A, and S_v is the subset of S for which feature A has value v (i.e. $S_v = \{s \in S \ : \ s_A = v\}$where s_A is the value of feature A in the specific data point s). Notice that $\frac{|S_v|}{|S|}$ is just the empirical distribution of the attribute A according to the sample.
   - **Train Error**. The simplest definition of gain is the decrease in training error. Formally, if we denote $C(x) = min\{x, 1-x\}$, and denote P_S to be the empirical distribution defined by the sample S. Then, in the case all labels are binary then the training error before splitting on feature A is $C(P_S(y=1))$. The expected error after splitting on feature A can be obtained by:

     $$\sum_{v \in values(A)} \frac{|S_v|}{|S|} C(P_S(y = 1 \mid s_A = v)) = \sum_{v \in values(A)} \frac{|S_v|}{|S|} C(P_{S_v}(y = 1))$$

     Where $P_{S_v}$is the empirical distribution according to the subset of the sample S_v.
     Therefore, we can denote $Gain$ to be the difference between the two, namely:

     $$Gain(S, A) = C(P_S(y = 1)) - \left(\sum_{v \in values(A)} \frac{|S_v|}{|S|} C(P_{S_v}(y = 1)) \right)$$

- **Gini Index**. The Gini coefficient is a measure of the inequality of a distribution, a value of 0 expressing total equality and a value of 1 maximal inequality. A feature will be considered good if it promotes equality in the distributions conditioned on the features value, i.e. the expected gini index after the split shall be small. The formal definition:

$$Gini\left(S\right) = 1 - \sum_{i \in L} P_S\left(L = i\right)^2$$

and for binary labels:

$$Gini\left(S\right) = 1 - P_S\left(L = 0\right)^2 - P_S\left(L = 1\right)^2$$

In order to use the gini index for the gain function in the binary labels scenario simply plug in the gini function instead of $C\left(\right)$ in the former bullet.

2. (10 points) Run the algorithm d+1 (where d is the number of features, $d = 16$) times - each time with a different number of iterations (i.e each run should produce a tree of different height, from 0 to d). Display the resulting trees and produce graphs of the train and validation errors as a function of the height of the tree (all should be handed in. If you have trouble with implementing a function which draws the tree elegantly, it suffices to describe in words: number of nodes, number of levels, etc).

3. (15 points) Recall that the generalization error of a sample S and a distribution (by which it was generated i.i.d) $\mathcal{D}$ on hypothesis h is $|L_{\mathcal{D}}\left(h\right) - L_S\left(h\right)|$. Implement and Run the pruning algorithm (on the tree of height d from the previous part). For an estimate of the generalization error, use the difference between the validation error and train error (use the validation and training data you downloaded). Hand in a sketch of the resulting tree including the generalization error achieved for it according to the validation set. You may find pseudo-code for the pruning algorithm in Shay's book, chapter 18.

4. (10 points) According to the previous sections, we can assess which height should be used in order to minimize validation error $L_V\left(h\right)$ without overfitting. Since our data is limited, a better approach would be cross validation. Using 8-fold cross validation, unite the validation+train datasets into 1 dataset; split this dataset into 8 equally-sized groups; leave each of those groups aside while reconstructing all trees from sections 2,3. For each height and for the pruned tree, calculate the average validation error and plot in a graph. Now, you should be able to decide what is the optimal architecture (height/pruned) to use for this task. After this procedure, is it possible (without additional data) to give a good estimation for the true error of the best tree? Leaving the pruned architecture aside, is the best

height according to cross validation and simple validation (section 2) the same?