CSE 446 (Spring 2015) 1st Assignment

Instructions Submit your homework on the *dropbox*.

1 Decision trees

Problem statement For this mini-project, we'll be using the *Letter Recognition Data Set* for classifying hand-written characters from the Latin alphabet. The dataset contains about $20,000$ examples, spanning 26 classes; each example having 16 attributes taking integral values. The attributes are explained on the UCI page.

Your task will be to generalize the ID3 algorithm [1] (with split-stopping) for the multiclass setting, implement the same, and to apply it to the given dataset.

The binary version of the ID3 is described in Chapter 8 of Duda et al. or Chapter 3 of Mitchell. The following general definitions ought to be useful in aiding your task.

Conditional Entropy Recall that in the original ID3 algorithm, the attribute to split on is chosen to be one which maximizes Information gain (or equivalently minimizes the conditional entropy).

Suppose the (current) dataset $\mathcal{D} = \{(x_j, l_j)\}$, contains $|\mathcal{D}|$ examples belonging to k different classes. Let the count for class 'i' in a set S be given by $\nu_i(S) := |\{(x,l) \in S \mid l = c_i\}|$. The estimated probability-of-occurence for each class is then given by, $\hat{p}_i = \frac{\nu_i(\mathcal{D})}{|\mathcal{D}|}$, and the (estimated) entropy of $\mathcal{D}$ is given by,

$$\hat{H}(\mathcal{D}) := \sum_{i=1}^{k} -\hat{p}_i \log_2(\hat{p}_i).$$

If we choose to split on an attribute 'A' taking m values, then the conditional entropy for the same is given by,

$$\hat{H}(\mathcal{D}|A) := \sum_{i=1}^{m} \hat{p}(A = a_i|\mathcal{D}) H(\mathcal{D}[A = a_i]), \text{ where}$$

$$\mathcal{D}[A = a_i] := \{(x,l) \in \mathcal{D} \mid x[A] = a_i\}, \quad \hat{p}(A = a_i|\mathcal{D}) := \frac{|\mathcal{D}[A = a_i]|}{|\mathcal{D}|}.$$

χ^2 test If the attribute A is irrelevant to $\mathcal{D}$ - read as "if the null-hypothesis holds" - then it is approximately true that,

$$S = \sum_{j=1}^{m} \sum_{i=1}^{k} \frac{(\nu_i(\mathcal{D}[A = a_j]) - e_i^j)^2}{e_i^j} \sim \chi^2[(m-1) \times (k-1)],$$

where,

$$e_i^j := |\mathcal{D}[A = a_j]| \frac{\nu_i(\mathcal{D})}{|\mathcal{D}|},$$

is the expected count for the class 'i' when $A = a_j$; and $\chi^2((m-1) \times (k-1))$ is the Chi-squared distribution with $(m-1) \times (k-1)$ degrees of freedom. The p-value for some observation $S = s$ is defined to be,

$$\text{p-value} := \Pr[S \geq s] = \int_{[s,\infty)} \chi^2[(m-1) \times (k-1)] \mathtt{d}x.$$

If the p-value is small - or smaller than a given threshold - then it is likely that the attribute is not irrelevant (yes, that's a double negative). For a more mathematical explanation as to why this is, see [2]. (Python users: You may want to look at the functions `st.chi2.cdf` & `scipy.stats.itemfreq`. Java users: Use the method `critchi(double p, int df)` $\in$ `Chi.java`, which returns a threshold for S given the p-value & degrees-of-freedom. Others: Make a table of `critchi`.)

Dataset Download the dataset the data/source from *here*. The archive contains files `features.txt` & `labels.txt`. The latter encodes the alphabet as integers $0, \ldots, 25$ (there is only one case). The format of the files is not complicated: just rows of integers separated by empty spaces.

The first 12000, samples form your training set, and the latter 8000 the test data set.

Implementation You may program in C/C#/C++, Java, Python or R (or gasp Lisp!). If you are not proficient in any of these, please contact the TAs. For initial debugging, it is recommended that you construct a very simple data set (e.g., based on a boolean formula) and test your program on it. Your program should use the chi-squared split stopping criterion with the p-value threshold given as a parameter.

1.1 Questions

Use your implementation with the threshold for the criterion set to $0.05, 0.01$ and 1. Remember, 1 corresponds to growing the full tree.

Answer the following questions:

1. For each value of threshold, what is your tree's -training & test- accuracy and size (size equals number of internal nodes and leaves) ? What do you observe? If all your accuracies are low, tell us what you have tried to improve the accuracies and what you suspect is failing.

2. Explain which options work well and why.

3. For the best performing tree, pick the top-5 paths that explain a large fraction of the test data. What are the number of test examples correctly classified by each of these paths ? List the class associated with each of the leaf node on these paths.

4. **Extra Credit** (10%): Try to improve your predictive accuracy. There are many approaches you could try. You could construct new attributes from the existing ones and augment the examples with them. You could also try alternative classification techniques, or modify one that you used above. Explain.

Your report should also contain a good high level description of how your code works. Document well any complicated parts you might have. You are not required to provide code that is extremely well optimized, but you should look for any obvious, gross optimizations that are possible.

2 Rule Induction

Mitchell 10.1: Consider a sequential covering algorithm such as CN2 and a simultaneous covering algorithm such as ID3. Both algorithms are to be used to learn a target concept defined over instances represented by conjunctions of n boolean attributes. If ID3 learns a balanced decision tree of depth d, it will contain $2^d - 1$ distinct decision nodes, and therefore will have made $2^d - 1$ distinct choices while constructing its output hypothesis. How many rules will be formed if this tree is re-expressed as a disjunctive set of rules? How many preconditions will each rule possess? How many distinct choices would a sequential covering algorithm have to make to learn this same set of rules? Which system do you suspect would be more prone to overfitting if both were given the same training data?

References

[1] *Induction on Decision trees*, J Ross Quinlan

[2] *Chi-squared goodness-of-fit test*, Course Notes - “Statistics for Applications”, Prof. Dmitry Panchenko