Solutions Homework 3 – CS478

April 11, 2000

1 Problem 10.1 (10pt)

2pt In a balanced tree with $2^d - 1$ decision nodes, 2^{d-1} nodes are at the bottom of the tree. Only one leaf can be attached to any of these *final* nodes. The number of rules required to *emulate* the tree with a rule-based system is equal with the number of nodes in this last layer (number of leaves if you want), namely 2^{d-1}.

2pt The number of preconditions for each of the rules is $d - 1$, the depth of the tree with decision nodes only.

3pt As mentioned in the text, the number of distinct decisions a *sequential* algorithm must do to learn this rules is $(d - 1) * 2^{d-1}$.

3pt Since the *sequential* algorithm takes more decisions when compared with the decision tree ($2^d - 1$ decisions), it is more prone to *overfitting*.

Grading I considered the solution with 2^d rules, d preconditions and $d2^d$ decisions for the *sequential* algorithm also correct. If you had something like 2^d rules and $d - 1$ preconditions or the other way around I took off 1pt. For the last part I took off 2pt for no justification at all, 1pt for a decently argued wrong solution (that ID3 is more prone to overfitting). Some of you suggested that CN2 has some anti-overfitting protection. I considered this solution correct if sufficient arguments were present.

2 Problem 10.2 (10pt)

The only thing that has to be changed is the way *All Constraints* are generated. To accommodate constraints of the form $a < v$ one, as in the case of decision trees (following the hint), has to sort all the possible values of the attribute a present in the training data, pick threshold values τ as the average of consecutive values for which there are examples with different classifications, and add constraints $a < \tau$ to the list of constraints.

Grading If you just suggested that *All Constraints* part has to be modified and that you will add constraints of the form $a < v$ where v is a value seen in training data you got 7pt.

If you mentioned that you can do better by sorting the values and taking midpoints where classification changes you got all the 10pt.

Where you haven't clearly specified what has to be modified or you made confusions I took 1-4pt depending on the particular mistake.

3 Problem 10.3 (10pt)

Like in the previous problem, to accommodate constraints of the form $a \in V$ one has to modify only the way the set of all constraints are built. For each attribute a take all the non-null subsets V of the set of all values seen in training data and add constraints $a \in V$ to the set of constraints.

Noting else should be modified. The Performance function is specified in a general way and can accommodate the new constraints.

Grading Again I took off points if you haven't clearly specified how the modifications have to be made. Some of you tried to emulate the new constraints with constraints of the form $a = v$, which is not exactly possible (I took off about 5pt).

4 Problem 10.4 (25pt)

4pt the learn-one-rule algorithm (Table 10.2) is using generate-and-test, general-to-specific and sequential cover

7pt generate-and-test vs. data-driven: generate-and-test is less prone to noise since every decision is based on all the available data, data-driven is faster since it considers a smaller number of choices

7pt general-to-specific vs. specific-to-general: is clear how to start general-to-specific (with the most general hypothesis), specific-to-general might be preferable if the most specific solution is desirable and works faster in practice.

7pt sequential cover vs. simultaneous cover: sequential cover might produce independent rules, works well with a lot of data, simultaneous cover works faster and has redundancy in rules (can be desirable in certain circumstances).

Grading I accepted any reasonable advantages and disadvantages. If you just mentioned an advantage for one side and repeated the symmetric disadvantage for the other side I took off 3pt (you were supposed to discuss positive and negative aspects for teh alternative not only negative).

If you didn't mention which choices are made by the learn-one-rule algorithm I took off 4pt (you were explicitly required to talk about this).

5 Problem 8.3 (20pt)

I considered two versions of the lazy-ID3 correct (plus the small variations). In both approaches the training phase just stores the training examples. At classification time:

- Start building the tree like in ID3(use information gain to decide on which attribute to split) but build the tree only on the branch that has the value of the attribute in the query. In this way you construct only one path in the tree, the one that can answer your query (the other ones are irrelevant for this query anyway). The big advantage is that you do not build parts of the tree you don't need (by moving the computation from training phase to classification phase but saving exponential time in this way) and if the number of queries is small (when compared to the number of attributes for example) you can answer them faster. This becomes not so advantageous over a big number of queries. Since this

version of lazy-ID3 gives exactly the answer the original ID3 gives there are no other disadvantages (other than speed in some circumstances).

- Pick k closest examples to the query and build a tree using ID3. Use this tree to classify the query. Since k is usually much smaller than the number of training examples, the obtained tree is smaller and can be learned faster. Since finding the closest k neighbors can be quite computational intensive in itself is not clear if this is faster than learning with ID3 a tree on the training data, but using only similar examples can improve the final result. As in the previous case doing this repeatedly on multiple queries can be quite expensive.

Grading 10pt for the algorithm (the English description is fine), 10pt for mentioning at least an advantage and a disadvantage (5pt for each).

Some of you have proposed algorithms that are not lazy, in the sense that you do more work when compared to normal ID3 even when you know the query in advantage. In these cases I subtracted 5-10pt depending on the solution.

6 Problem 6 (25pt)

Let d be the number of attributes, n the number of rules in CN2, N the number of training examples ($N > n$ if the CN2 solution is not overfitting the data).

10pt The classification time for ID3 is $O(d)$ (the depth of the tree that is about the number of attributes), for CN2 is $O(d * n)$ (every rule has about d preconditions and we have n of them) and for k_{nn} is $O(d * N)$ (to compute the distance between the query and a training example it takes $O(d)$ and we have to compute N of them). Looking at this analysis is clear that ID3 is superior (except degenerated cases with $n = 1$ and $N = 1$).

Grading I gave about 3pt per case for analyzing it and 1pt for the final verdict. If your analysis was not precise (you just explained in words without clear cost analysis) I took 2-3pt off. If you made some precise statements I took only 1pt off.

5pt As mentioned in the class, IF–THEN rules are easier to understand for humans than pretty much anything else. Some of you argued that k_{nn} solution gives a better explanation but I disagree since humans tend to have a similarity notion that is quite different from the one used in these systems, that seems bizarre. You got 2-3pt for a reasonably argued solution like this.

10pt When more data is available ID3 and CN2 are in a very bad position since we have to retrain the system from scratch. This is due to the fact that both algorithms use sophisticated greedy techniques that involve entropy calculation that are hard to adjust to the new data.

k_{nn} is in a very good position since in the training phase we just need to store the training examples, so we just need to add the new data to the old one.

Grading 3pt for specifying the behavior of each algorithm, 1pt for mentioning the best. Some of you argued that you can incrementally update ID3 and/or CN2. If the explanation was reasonable I considered it correct.