CS60050 Machine Learning - Weekly Report

Nisarg Upadhyaya (19CS30031)

Week 3: 25-27 August

1 Topics covered

- Remarks on Candidate Elimination and Version Spaces
- Inductive Bias and Unbiased Learners
- Computational Complexity of VS
- Probably Approximately Correct learning model
- Some general notes for a learning problem
- Decision Trees

2 Summary

2.1 Remarks on Candidate Elimination and Version Spaces

2.1.1 Convergence of CE Algorithm

The candidate elimination algorithm converges only if

1. There are no errors in training examples
2. There is some hypothesis in H that describes the target concept

2.1.2 Ability to request training examples

With the version space bounded by G and S using the CE algorithm, it is now possible for the learner to ask for specific training examples which will help it converge towards a correct target concept. Queries which are satisfied by half of the version space generated so far are optimum queries as the size of the version space is reduced by half with every example and we can converge in logarithmic time. However, this might not always be possible.

2.1.3 Voting procedure

With the current version space in hand, if the learner is asked to classify a new instance a voting procedure can be run on the VS. If more hypothesis in the VS classify it as positive it is declared positive else negative. The percentage of hypothesis in the VS which classify the instance as positive/negative can be used as a confidence score of the prediction.

2.1.4 Inability to converge

It may be possible that the target concept is not present in the given hypothesis space. We say the hypothesis space is restricted/biased. In such cases we require a more expressive hypothesis space.

2.2 Inductive Bias and Unbiased Learners

An unbiased learner is a one with a hypothesis space capable to represent every teachable concept. While expressibility is not a concern in this space, the ability to generalise beyond training examples surely is. It needs to use all the instances to learn the concept. The voting procedure fails on any instance not in the training examples. The version space so generated will always return a 50/50 vote on such examples. This leads to the following property of inductive inference: *a learner that makes no a priory assumptions regarding the identity of the target concept has no rational basis for classifying any unseen instances.*

2.2.1 Inductive Bias

The key idea is that there is a policy by which the learner generalizes beyond the training examples in order to classify new instances. Thus inductive bias is a base set of assumptions B which allow the learning algorithm to classify instances deductively. Thus we have

1. For rote learner: $B = \{\}$, no bias
2. For CE algorithm: $B = \{$The target concept c is present in H$\}$
3. For Find-S algorithm: $B = \{$The target concept c is present in H, All instances that are not positive are negative$\}$

2.3 Computational Complexity of VS

The set S grows linearly in case of conjunctive feature vectors while the set G grows exponentially. In more expressive languages both S and G can grow exponentially.

2.4 Probably Approximately Correct learning model

Learner gives a hypothesis h which classifies new instances within a specific error bound, $error(h) = P(c(x) \neq h(x))$. A concept class C defined over a set of instances X of length n is called PAC learnable if $\forall c \in C$ a learner L using hypothesis space H with probability at least $(1-\delta)$ outputs a hypothesis $h \in H$ such that $error(h) \leq \epsilon$, $0 < \epsilon, \delta < 1/2$ in time that is polynomial in $1/\epsilon, 1/\delta, n$, and $size(c)$.

2.4.1 Sample complexity of a PAC learner

To get a rough estimate of the number of training examples required so that the version space consists of any consistent hypothesis bounded by an error of ϵ. The version space for a target concept c having such property is called ϵ-exhausted. If the maximum probability for providing a consistent hypothesis with $error > \epsilon$ is δ then the count of training examples m is

$$m \geq (1/\epsilon)(ln(|H|) + ln(1/\delta))$$

2.4.2 Sample complexity of infinite H

For infinite H we use Vapnik-Chervonenkis dimension. It is the size of the largest finite subset in X shattered by H. $VC(h) \leq log_2|H|$. Shattering by a hypothesis space H is when there exists a consistent h for every dichotomy on a set of instances S.

2.5 Some general notes for a learning problem

It is important to handle noise in data and allow tolerance in the model for this. A balanced model should be preferred which is not highly complex or overly simple as these may lead to overfitting and underfitting. Higher the proportion of training samples in the input space, better will be the model fitting. A trade off needs to be done between the complexity of the hypothesis, amount of training data and the generalization error on new examples.

2.6 Decision Trees

While the hypothesis space previously considered during concept learning was a conjunction of attributes, decision trees represent a disjunction of conjunction of attributes. This is a larger hypothesis space. Each path from tree root to a leaf is a conjunction of attribute tests and the different paths are a disjunction of these conjunctions.

2.6.1 Applications

Decision trees are useful when:

1. Instances are described by a fixed set of attributes and their values
2. Target function is discrete valued
3. The hypothesis space is disjunctive
4. Training data is noisy

2.6.2 Construction

One greedy approach is by initially treating the entire dataset as a single box. Our target is to split this box into compartments which are *pure*. The sense of *pure* is that each compartment finally contains the same class label. Each compartment will then correspond to some conjunction of attributes. For each box we choose the split with an attribute that reduces its impurity (in terms of class labels) by the maximum amount and split the box having highest reduction in impurity. Stop when all boxes are pure.

2.6.3 How to choose the best attribute for split?

For this we use the concept of entropy and information gain. Entropy is a measure for (im)purity of an arbitrary collection of examples S. In the most general sense, if the target function can take on c different values, then the entropy of S relative to this c-wise distribution is

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2(p_i)$$

where p_i is the proportion of S belonging to class i. For selecting the best attribute we calculate $Gain(S, A)$ which is the expected reduction in entropy after selecting attribute A for a collection of examples S. It is defined as

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

where $Values(A)$ is the set of all possible values of attribute A and S_v is the subset of S for which attribute A has value v. We choose the attribute which provides the maximum gain.

3 Challenging concepts

This week was slightly more math oriented than previous two weeks and it took some extra reading to properly understand the various inequalities and equations discussed in the class.

4 Interesting concepts

In the first go it seems like just taking the hypothesis space large enough to include every teachable concept might solve the problem faced by CE and Find-S algorithms but then getting to know the importance of Inductive Bias and the role it plays in providing generalization beyond the training examples was very interesting.

5 Concepts not understood

None

6 Ideas

The height of the decision tree plays an important role while trying to classify new instances. So is the branching. High branching and height means more data storage. It will be interesting to know that if we can compromise on the accuracy for some threshold ϵ then how can we improve the speed and reduce the tree size by approximating after a certain level. This will also include calculating confidence scores of such approximations.