ml-skill-proj3

---

1. You are provided with a training set of examples. Which feature will you pick first to split the data as per the ID3 decision tree learning algorithm? Show all your work: compute the information gain for all the four attributes and pick the best one.

Figure 1: Table with training examples. Each row corresponds to a single training example. There are four features, namely, outlook, temperature, humidity, and wind. "PlayTennis" is the class label.

Find the feature with highest gain

Total entropy = Entropy (Decision) = - p(Yes) * log2p(Yes) - p(No) * log2p(No)
Entropy (Decision) = - (9/14) * log2(9/14) - (5/14) * log2(5/14) = 0.940

1. Outlook p (Yes | $Outlook = Overcast$) = 4/4 = 1

p (No | $Outlook = Overcast$) = 0/4 = 0

p (Yes | $Outlook = Sunny$) = 2/5 = 0.4

p (No | $Outlook = Sunny$) = 3/5 = 0.6

p (Yes | $Outlook = Rain$) = 3/5 = 0.6

p (No | $Outlook = Rain$) = 2/5 = 0.4

Entropy(Decision| $Outlook = Overcast) = -(0/4)*log2(0/4)-(4/4)*log2(4/4) =$ 0

Entropy(Decision| $Outlook = Sunny) = -(3/5) * log2(3/5) - (2/5) * log2(2/5) =$ 0.97

Entropy(Decision| $Outlook = Rain) = -(2/5) * log2(2/5) - (3/5) * log2(3/5) =$ 0.97

Gain(Decision, Outlook) = Entropy(Decision) - (p(Decision| $Outlook = Overcast)* Entropy(Decision \mid Outlook = Overcast)) - (p(Decision \mid Outlook = Sunny) * Entropy(Decision \mid Outlook = Sunny)) - (p(Decision \mid Outlook = Rain) * Entropy(Decision \mid Outlook = Rain)) = 0.940 - ((4/14) * 0) - ((5/14) * 0.97) - ((5/14) * 0.97) = 0.247$

2.Temperature p (Yes | $Temperature = Hot) = 2/4p(No \mid Temperature =$

$Hot) = 2/4 p(Yes \mid Temperature = Mild) = 4/6 p(No \mid Temperature = Mild) = 2/6 p(Yes \mid Temperature = Cool) = 3/4 p(No \mid Temperature = Cool) = 1/4$

Entropy (Decision| $Temperature = Hot) = -(2/4)*log2(2/4)-(2/4)*log2(2/4) = 1$

Entropy (Decision| $Temperature = Mild) = -(2/6)*log2(2/6)-(4/6)*log2(4/6) = 0.918$

Entropy (Decision| $Temperature = Cool) = -(1/4)*log2(1/4)-(3/4)*log2(3/4) = 0.81$

Gain (Decision, Outlook) = Entropy(Decision) - (p(Decision| $Temperature = Hot)*Entropy(Decision \mid Temperature = Hot))-(p(Decision \mid Temperature = Mild)*Entropy(Decision \mid Temperature = Mild))-(p(Decision \mid Temperature = Cool) * Entropy(Decision \mid Temperature = Cool)) = 0.940 - ((4/14) * 1) - ((6/14) * 0.918) - ((4/14) * 0.81) = 0.029$

3.Humidity p (Yes | $Humidity = High) = 3/7 p(No \mid Humidity = High) = 4/7$

p (Yes | $Humidity = Normal) = 6/7 p(No \mid Humidity = Normal) = 1/7$

Entropy(Decision| $Humidity = High) = -(4/7) * log2(4/7) - (3/7) * log2(3/7) = 0.985 Entropy(Decision \mid Humidity = Normal) = -(1/7) * log2(1/7) - (6/7) * log2(6/7) = 0.59$

Gain (Decision, Humidity) = Entropy(Decision) - (p(Decision| $Humidity = High) * Entropy(Decision \mid Humidity = High)) - (p(Decision \mid Humidity = Normal) * Entropy(Decision \mid Humidity = Normal)) = 0.940 - ((7/14) * 0.985) - ((7/14) * 0.59) = 0.153$

4.Wind

1- Entropy(Decision| $Wind = Weak) = -p(No)*log2p(No)-p(Yes)*log2p(Yes)2- Entropy(Decision \mid Wind = Weak) = -(2/8) * log2(2/8) - (6/8) * log2(6/8) = 0.8111 - Entropy(Decision \mid Wind = Strong) = -p(No) * log2p(No) - p(Yes) * log2p(Yes)2 - Entropy(Decision \mid Wind = Strong) = -(3/6)*log2(3/6)-(3/6)* log2(3/6) = 1$

Gain(Decision, Wind) = Entropy(Decision) - (p(Decision| $Wind = Weak).Entropy(Decisi$ $Wind = Weak)) - (p(Decision \mid Wind = Strong).Entropy(Decision \mid Wind = Strong)) = 0.940 - ((8/14) * 0.811) - ((6/14) * 1) = 0.048$

1- Gain(Decision, Outlook) = 0.246 2- Gain(Decision, Temperature) = 0.029 3- Gain(Decision, Humidity) = 0.153

2. We know that we can convert any decision tree into a set of if-then rules, where there is one rule per leaf node. Suppose you are given a set of rules $R = \{r_1, r_2, \ldots, r_k\}$, where r_i corresponds to the i^{th} rule. Is it possible to convert the rule set R into an equivalent decision tree? Explain your construction or give a counterexample.

   The hierarchical structure of the tree ensures that the rules in the set are nonoverlapping, that is, each example can only be covered by a single rule. So if the rule is conflict to each other or overlapping, that will be difficult to convert to a decision tree.

3. Suppose $\boldsymbol{x} = [x_1, x_2, \ldots, x_d]$ and $\boldsymbol{z} = [z_1, z_2, \ldots, z_d]$ be two points in a high-dimensional space (i.e., d is very large).

   (a) Try to prove the following, where the right-hand side quantity represent the standard Euclidean distance.

   $$\left(\frac{1}{\sqrt{d}}\sum_{i=1}^{d} x_i - \frac{1}{\sqrt{d}}\sum_{i=1}^{d} z_i\right)^2 \leq \sum_{i=1}^{d} (x_i - z_i)^2$$

   Hint: Use Jensen's inequality – If X is a random variable and f is a convex function, then $f(E[X]) \leq E[f(X)]$.

   (b) We know that the computation of nearest neighbors is very expensive in the high-dimensional space. Discuss how we can make use of the above property to make the nearest neighbors computation efficient?

   (a)

   $$\sqrt{\mid x - z \mid} \leq \sqrt{\mid x - r \mid + \mid r - z \mid} \leq \sqrt{\mid x - r \mid} + \sqrt{\mid r - z \mid}$$

   (b) We can find the most correlated feature to reduce the feature. Also, we can

   use PCA, so the dimension of the feature can be decreased.

4. **Fortune Cookie Classifier**: You will build a binary fortune cookie classifier. This classifier will be used to classify fortune cookie messages into two classes: messages that predict what will happen in the future (class 1) and messages that just contain a wise saying (class 0). For example, "Never go in against a Sicilian when death is on the line" would be a message in class 0. "You will get an A in Machine learning class" would be a message in class 1.

Files Provided: There are three sets of files. All words in these files are lower case and punctuation has been removed.

1) The training data: traindata.txt. This is the training data consisting of fortune cookie messages. trainlabels.txt: This file contains the class labels for the training data.

2) The testing data: testdata.txt. This is the testing data consisting of fortune cookie messages. testlabels.txt: This file contains the class labels for the testing data.

3) A list of stopwords: stoplist.txt

There are two steps: the pre-processing step and the classification step. In the pre-processing step, we convert fortune cookie messages into features to be used by your classifier. We use a bag of words representation. The following steps outline the process involved:

Form the vocabulary. The vocabulary consists of the set of all the words that are in the training data with stop words removed (stop words are common, uninformative words such as "a" and "the" that are listed in the file stoplist.txt). The vocabulary will now be the features of your training data. Keep the vocabulary in alphabetical order to help you with debugging.

Then, we convert the training data into a set of features. Let M be the size of your vocabulary. For each fortune cookie message, we convert it into a feature vector of size M. Each slot in that feature vector takes the value of 0 or 1. For these M slots, if the ith slot is 1, it means that the ith word in the vocabulary is present in the fortune cookie message; otherwise, if it is 0, then the ith word is not present in the message. Most of these feature vector slots will be 0. Since you are keeping the vocabulary in alphabetical order, the first feature will be the first word alphabetically in the vocabulary.

(a) Implement the ID3 decision tree learning algorithm that we discussed in the class. The key step in the decision tree learning is choosing the next feature to split on. Implement the information gain heuristic for selecting the next feature. Please see lecture notes or `https://en.wikipedia.org/wiki/ID3_algorithm` for more details.

(b) Implement the decision tree pruning algorithm discussed in the class (via validation data).

(c) Compute the accuracy of decision tree and pruned decision tree on validation examples and testing examples. List your observations by comparing the performance of decision tree with and without pruning.

My code didn't work well. In my opinion, after pruning, the testing accuracy will increase because we prevent the over-fitting. However, for the training example, the tree without pruning will have higher accuracy.