CSC 242 Learning Writeup

PRIKSHET SHARMA

CSC Concepts: • **Artificial Intelligence, Supervised Learning**

KEYWORDS

Decision Trees, Linear Regression, Gradient Descent, Neural Networks

0 COMPILATION INSTRUCTIONS

Go to the directory CSC242-project-3-ferguson/src and run the following commands:

For WillWaitProblem decision tree: make decisionTreeAIMA

For discrete Iris decision tree: make decisionTreeIris

For perceptron linear classifier: make perceptron

For logistic linear classifier: make logistic

For single layer neural net: python neural\_net.py

I want to claim extra credit for implementing all three learners.

1 INTRODUCTION

1.1 Decision Trees

“A decision tree represents a function that takes as input a vector of attribute values and returns a “decision”—a single output value. The input and output values can be discrete or continuous… A decision tree reaches its decision by performing a sequence of tests. Each internal node in the tree corresponds to a test of the value of one of the input attributes, Ai, and the branches from the node are labeled with the possible values of the attribute, . Each leaf node in the tree speciﬁes a value to be returned by the function.”[[1]](#footnote-2)

1.2 Linear Classifier

“A **decision boundar**y is a line (or a surface, in higher dimensions) that separates the two classes… A linear decision boundary is called a linear separator and data that admit such a separator are called linearly separable.”[[2]](#footnote-3)

“Using the convention of a dummy input , we can write the classiﬁcation hypothesis as:”[[3]](#footnote-4)

“Alternatively, we can think of *h* as the result of passing the linear function through a **threshold function**:”[[4]](#footnote-6)

1.3 Neural Network

“Neural networks are composed of nodes or units… connected by directed links. A link from unit i to unit j serves to propagate the activation ai from i to j.8 Each link also has a numeric weight wi,j associated with it, which determines the strength and sign of the connection. Just as in linear regression models, each unit has a dummy input a0 =1with an associated weight w0,j. Each unit j ﬁrst computes a weighted sum of its inputs:”[[5]](#footnote-8)

“Then it applies an activation function g to this sum to derive the output:”[[6]](#footnote-10)

“The activation function g is typically either a hard threshold, in which case the unit is called a **perceptron**, or a logistic function, in which case the term **sigmoid perceptron** is sometimes used.”[[7]](#footnote-12)

2 EXPERIMENTAL AND COMPUTATIONAL DETAILS

2.1 Decision Trees

“The greedy search used in decision tree learning is designed to approximately minimize the depth of the ﬁnal tree. The idea is to pick the attribute that goes as far as possible toward providing an exact classiﬁcation of the examples… We will use the notion of information gain, which is deﬁned in terms of entropy. Entropy is a measure of the uncertainty of a random variable; acquisition of information corresponds to a reduction in entropy. A random variable with only one value—a coin that always comes up heads—has no uncertainty and thus its entropy is deﬁned as zero; thus, we gain no information by observing its value. A ﬂip of a fair coin is equally likely to come up heads or tails, 0 or 1, and we will soon show that this counts as “1 bit” of entropy.”[[8]](#footnote-13)

Entropy is defined as:

“[D]eﬁne B(q) as the entropy of a Boolean random variable that is true with probability q:”

“the expected entropy remaining after testing attribute A is”

“The **information grain** from the attribute test on A is the expected reduction in entropy.”

The importance function given to us is exactly this gain function.

Moreover, the most important algorithm we had to implement was the actual decision tree learning algorithm, which was implemented as follows using the pseudocode given in AIMA Fig 18.5:

**protected** DecisionTree learn(Set<Example> examples, List<Variable> attributes, Set<Example> parent\_examples) {  
 **if** (examples.isEmpty()) **return new** DecisionTree(pluralityValue(parent\_examples));  
 *// if all examples have the same classification* **else if** (uniqueOutputValue(examples) != **null**) {  
 System.***out***.println(**"printing unique output value: "** + uniqueOutputValue(examples));  
 **return new** DecisionTree(uniqueOutputValue(examples));  
 } **else if** (attributes.isEmpty()) {  
 **return new** DecisionTree(pluralityValue(examples));  
 } **else** {  
 Variable A = argMaxAttr(attributes, examples);  
 DecisionTree tree = **new** DecisionTree(A);  
 **for**(String vk : A.getDomain()) {  
 Set<Example> exs = examplesWithValueForAttribute(examples, A, vk);  
 DecisionTree subtree = learn(exs, subtract(attributes, A), examples);  
 tree.**children**.add(subtree);  
 }  
 **return** tree;  
 }  
 }

2.1 Linear Classifier

“In multivariate linear regression, our hypothesis space is the set of functions of the form:”[[9]](#footnote-18)

“The term, the intercept, stands out as different from the others. We can ﬁx that by inventing a dummy input attribute, xj,0, which is deﬁned as always equal to 1. Then h is simply the dot product of the weights and the input vector (or equivalently, the matrix product of the transpose of the weights and the input vector):”[[10]](#footnote-19)

The update rule for multivariate linear regression, which is required for linear classification, is given by

2.3 Single Layer Feed Forward Neural Network

“A network with all the inputs connected directly to the outputs is called a **single-layer neural network**, or a **perceptron network**.”[[11]](#footnote-22) I implemented the single-layer neural network in python that learns a simple Boolean function.

3 ANALYSIS AND COMPARISON OF IMPLEMENTATIONS

Decision tree

One of the sample outputs from the Iris dataset:

C:\Users\Prikshet\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Capture.png

Fig. 1. Sample output for the example query given in the Project description when run with Rejection Sampling Algorithm

3.1 RAW DATA ANALYSIS

|  |  |  |
| --- | --- | --- |
| Proportion of data | Percent correct | Time to train \* 105 ns |
| 0.2 | 33.33 | 32.53 |
| 0.4 | 66.67 | 57.53 |
| 0.5 | 66.67 | 74.20 |
| 0.6 | 66.67 | 72.00 |
| 0.8 | 95.33 | 99.55 |
| 1.0 | 95.33 | 110.00 |

Table 1: Decision Tree accuracy compared to ratio of examples used for learning vs testing from the Iris dataset

|  |  |  |
| --- | --- | --- |
| Proportion of data | Percent correct | Time to train \* 105 (ns) |
| 0.2 | 83.33 | 38.48 |
| 0.4 | 66.67 | 31.34 |
| 0.5 | 66.67 | 42.81 |
| 0.6 | 83.33 | 39.26 |
| 0.8 | 91.67 | 49.10 |
| 1.0 | 100 | 54.87 |

Table 2: Decision Tree accuracy compared to ratio of examples used for learning vs testing from the AIMA dataset

|  |  |  |
| --- | --- | --- |
| Proportion of data | Time to train \* 109 ns | Accuracy |
| 0.2 | 5.85 | 0.99 |
| 0.4 | 6.25 | 0.99 |
| 0.5 | 6.26 | 0.99 |
| 0.6 | 6.43 | 0.97 |
| 0.8 | 6.43 | 0.95 |
| 1.0 | 6.53 | 0.93 |

Table 3: Perceptron classifier learn time compared to proportion of total data used for learning.

|  |  |  |
| --- | --- | --- |
| Proportion of data | Time to train \* 109 ns | Accuracy |
| 0.2 | 6.43 | 1 |
| 0.4 | 5.81 | 1 |
| 0.5 | 6.03 | 1 |
| 0.6 | 6.25 | 1 |
| 0.8 | 5.54 | 0.96 |
| 1.0 | 5.94 | 0.916 |

Table 4: Perceptron classifier learn time compared to proportion of total data used for learning.

3.3 Graphs (Processed Data)

Graph 1: Decision Tree accuracy compared to ratio of examples used for learning vs testing from the Iris dataset.

Graph 2: Decision Tree accuracy compared to ratio of examples used for learning vs testing from the AIMA dataset.

Graph 3: Perceptron classifier learn time compared to proportion of total data used for learning

3.4 Analysis of Processed Data

For the Iris dataset, the graph follows a linear trajectory for both learning time and percent correct. However, this trend is not apparent in the AIMA dataset. This is because the data is small.

As the data increases, we see a linear growth in learn time for perceptron classifier, whereas we see a speedup for the logistic classifier.

REFERENCES

|  |  |
| --- | --- |
| [1] | RUSSELL, STUART NORVIG PETER. *ARTIFICIAL INTELLIGENCE: a Modern Approach*. PEARSON, 2018. |
| [2] | FERGUSON, GEORGE. CSC242 Lecture Slides. |

1. RUSSELL, STUART NORVIG PETER. *ARTIFICIAL INTELLIGENCE: a Modern Approach*. PEARSON, 2018. [↑](#footnote-ref-2)
2. Ibid. [↑](#footnote-ref-3)
3. Ibid. [↑](#footnote-ref-4)
4. Ibid. [↑](#footnote-ref-6)
5. Ibid. [↑](#footnote-ref-8)
6. Ibid. [↑](#footnote-ref-10)
7. Ibid. [↑](#footnote-ref-12)
8. Ibid. [↑](#footnote-ref-13)
9. Ibid. [↑](#footnote-ref-18)
10. Ibid. [↑](#footnote-ref-19)
11. Ibid. [↑](#footnote-ref-22)