Project 4: Supervised Learning through Decision Trees and Artificial Neural Networks

Luke Runnels

April 18, 2023

1 Introduction

1.1 Supervised Learning

Supervised learning is one of three major paradigms for machine learning models. It consists of a dataset of attribute vectors $\mathbf{X} = (x_1, x_2, ..., x_n)$ and corresponding label vectors $\mathbf{Y} = (y_1, y_2, ..., y_n)$, where according to the explanation provided by Russell et. al, are generated from an unknown function $y = F(x)$. [4] The goal of supervised learning is to build a model that corresponds to the truth function $F(x)$ from the hypothesis space H, or all possible functions.

To find a good function for $F(x)$, the model needs a couple of considerations. First, according to Russell et. al, a good function choice is one that generalizes well to the testing dataset. [4] Second, the model needs to be closely consider the bias-variance tradeoff in the hypothesis space. This consideration often leads to the notion of underfitting and overfitting a model to the training dataset, which is not the goal of a supervised learning model.

1.2 Decision Trees

Decision trees are supervised learning models, where, according to Russell et al, are representations of the truth function are significant attributes are mapped to a single decision. [4] As the name suggests, this mapping is represented in an N-ary tree structure, where each internal node represents a chosen attribute, each branch represents an attribute value, and each leaf node represents a "decision" or prediction.

To build a good truth function in a decision tree while handling the bias-variance tradeoff, the tree needs to split the training data into small, homogeneous groups. That is, the splitting of the training dataset based on attributes should lead to cases where predictions are mutually exclusive while splitting as few attributes as possible. To test which attributes lead to this goal, the notion of information gain is applied when choosing an attribute to expand upon. It is the difference between the current attribute impurity and a possible reduced impurity.

1.3 Artificial Feedforward Neural Networks

A feedforward neural network is a type of deep learning model. It consists of a bunch of interconnected nodes, or deep layers, between the attribute vector and the output vector, which are named the input layer and output layer. Each node's significant is represented as a weighted sum between the connected nodes closer to the input layer.

To build a good truth function in the neural network while handling bias-variance tradeoff, the network needs to adjust the weights between each node to represent the training dataset. Adjusting weights requires three components. First, there needs to be a loss function between the initial prediction and the truth vector. Second, there needs to be activation function that roughly maps the weighted sums between the nodes into a "classification" type. Then, a backpropagation scheme is needed calculate the errors between all of the deep layers and the output layer. These components are combined to adjust the weights between the nodes through gradient descent.

2 Datasets used

2.1 Artificial Dataset

This dataset was originally provided for the first project of CS 4253-01 at the University of Tulsa [3]. It simply consists of 1000 datapoints. Each datapoint has two attribute types and an associated classification type. The two attributes are real values inbetween 0 and 1, while the classification type are the integers 0 and 1.

2.2 EMG Physical Action Dataset

This dataset was compiled by Theo Theodoridis at the University of Essex [5]. The attribute represent body segments from the left arm, right arm, left leg, and right leg. Each body segment consists of 8 muscles, which total 8 integer value attributes on each data point, that range from -4000 to 4000. There are 20 different classification types, which represent the type of physical activity that is happening from the muscle values.

2.3 Dataset of Optical Handwritten Digits

This dataset was originally compiled by E. Alpaydin and C. Kaynak at the University of Istanbul [1]. It consists of 64 attributes that originally representing the pixel strength on an 8x8 picture. Each attribute are real values inbetween 0 and 16. The associated classification types are simply the integers 0 through 9.

3 Algorithm Explanations

3.1 Decision Trees

Decision trees an N-ary tree data structure that keep track of the most significant attributes from a collection of attribute vectors $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_n)$ as the split points. The significance of an attribute is measured in terms of information gain. That is, given the jth attribute x_j^i from $\mathbf{X}_i$, the information gain is calculated as

$$Gain(\mathbf{X}, x_j) = Impurity(\mathbf{X}) - Remainder(\mathbf{X}, x_j) \tag{1}$$

The remainder term from (1) is calculated as

$$Remainder(\mathbf{X}, x_j) = \sum_{k=1}^{d} \frac{\text{num of Xs in } \mathbf{X}\backslash x_j^k}{\text{num of Xs in } \mathbf{X}} Impurity(\mathbf{X}\backslash x_j^k) \tag{2}$$

where the phrase "num of Xs in $\mathbf{X}$" means the total number of attributes in $\mathbf{X}$. It should be noted that the remainder term is essentially summation of all possible attribute values x_j^k for the attribute x_j and the resulting vector $\mathbf{X}\backslash x_j$ that does not contain x_j.

There are numerous implementations for the impurity function in (1), but the most common implementations are Entropy and the Gini Index, which are calculated as

$$Impurity(\mathbf{X}) = \begin{cases} -\sum_{l=1}^{k} p_l(\mathbf{X}) log_2(p_l(\mathbf{X})) & \text{Entropy} \\ \sum_{l=1}^{k} p_l(\mathbf{X})(1 - p_l(\mathbf{X})) & \text{Gini Index} \end{cases} \tag{3}$$

Given the number of possible classification types for $\mathbf{X}$ is represented as the integers $l \in [1, k]$, then $p_l(\mathbf{X})$ denotes the fraction of points in $\mathbf{X}$ that belong to class l.

Given the information gain calculation between $\mathbf{X}$ and $\mathbf{X}\backslash x_j$, the decision tree model is built by first choosing the significant attribute using (1), which is the attribute that maximizes (1). Then, every possible value for the chosen attribute is "branched" out from the current node. This action will lead from the vector $\mathbf{X}$ to $\mathbf{X}\backslash x_j$. This algorithm is recursively applied to $\mathbf{X}\backslash x_j$ and so on.

There are three cases to consider when $\mathbf{X}$ runs out of attributes to split upon. The three cases are

$$Base\_Case(\mathbf{X}, \mathbf{X}\backslash x_j) = \begin{cases} Plurality(\mathbf{X}) & \text{if } \mathbf{X}\backslash x_j = \emptyset \\ Plurality(\mathbf{X}\backslash x_j) & \text{if } \mathbf{X} = \emptyset \\ class & \text{if } \mathbf{X} \text{ all have the same class.} \end{cases} \tag{4}$$

The plurality function is simply the most common classification in $\mathbf{X}$.

As a reminder, the built decision tree is an N-ary where each internal node represents a chosen attribute, each branch represents an attribute value, and each leaf node represents a

"decision" or prediction.

A prediction for $\mathbf{X}$ is obtained by utilizing its attribute values to traverse down the tree. Assuming that a given branch b is labelled $x_j = v$ and all of the branches are sorted from left to right in ascending order based on v, the tree will traverse down b if $\mathbf{X}_k <= v$. This is recursively performed until a leaf node is reached.

3.2 Artificial Feedforward Neural Networks

3.2.1 Gradient Descent

Artificial Feedforward Neural Networks consists of a layer of nodes for the input attribute vector $\mathbf{X}$ and the output vector $\mathbf{Y}$. Given that there are L layers in the network, which includes the input and output layer, the jth node in the lth layer of the network is represented as

$$a_j^l = g(\sum_{i=1}^{n} W_{ij}^l * a_i^{l-1}) \tag{5}$$

where g is an activation function and W_{ij}^l is a weighted value inbetween the ith node in the previous layer and the jth node in the current layer.

For the network to develop a good approximation for the truth function $\mathrm{F}(\mathbf{X}) = \mathbf{Y}$, it needs to adjust the weights W_{ij}^l to accuracy map $\mathbf{X}$ to $\mathbf{Y}$. This is accomplished through three components. Given the truth output vector $\hat{\mathbf{Y}}$, the approximate loss between the predicted vector $\mathbf{Y}$ and $\hat{\mathbf{Y}}$ needs to be found. Although there are different loss functions, this project will use the Mean-Squared-Error(MSE) as the loss function. This is written as

$$Loss(Y, \hat{Y}) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{6}$$

or, per attribute in $\mathbf{Y}$ and $\hat{\mathbf{Y}}$

$$Loss(y_i, \hat{y}_i) = \frac{1}{2}(y_i - \hat{y}_i)^2 \tag{7}$$

The weights are adjusted through gradient descent, which is written as

$$W_{ij}^l = W_{ij}^l + \alpha\frac{\partial Loss(y_i, \hat{y}_i)}{\partial W_{ij}^l} \tag{8}$$

where α is called the learning rate and l is the current layer.

For the output layer, the loss gradient can be calculated by first substituting (5) into y_i then taking the derivative of (7) in order to obtain an intermediate error vector of

$$\Delta[L]_i = g^{'}(a_i^{L-1}) \times (y_i - \hat{y}_i) \tag{9}$$

which extends to

$$\frac{\partial Loss(y_i, \hat{y}_i)}{\partial W_{ij}^L} = a_i^L \times \Delta[L]_j \tag{10}$$

For the deep layers, the intermediate error vector is calculated by recursively applying the derivative of (7) to all of the layers through the chain rule. For any lth layer, this leads to the expression

$$\Delta[l]_i = g^{'}(a_i^l) \sum_{j=1}^{n} W_{ij} \Delta[l+1]_j \tag{11}$$

where extends to

$$\frac{\partial Loss(y_i, \hat{y}_i)}{\partial W_{ij}^l} = a_i^l \times \Delta[l]_j \tag{12}$$

3.2.2 L2 Regularization

Although gradient descent is good starting place for training neural networks, it suffers from some problems. For example, gradient descent can especially vulnerable to overfitting the training dataset. The best solution for reducing the effects of overfitting is applying a regularization term to the cost function. Although there are different types of regularization, this project will focus on L2 regularization.

Based on Andrew Ng's Coursea machine learning course in lecture 7 [2], the regularization term can be applied to the loss function as

$$Loss(Y, \hat{Y}) = \frac{1}{2} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{l=1}^{L} \sum_{i=1}^{l} \sum_{j=1}^{i} (W_{ij}^l)^2 \tag{13}$$

where λ is a simple regularization constant in between 0 and 0.1.

Regularization is applied to gradient descent via

$$W_{ij}^l = (1 - \alpha\lambda) * W_{ij}^l + \alpha \frac{\partial Loss(y_i, \hat{y}_i)}{\partial W_{ij}^l} \tag{14}$$

4 Performance & Analysis

4.1 Performance & Analysis of Decision Trees

The following figures show the performance of the decision trees for a different **X** sizes and impurity functions. For the artifical dataset, the different sizes include 50, 100, 200, 400, 600, 800, and 1000. It should be noted that the maximum amount of datapoints in the artificial dataset is 1000. The EMG and optical digit dataset have larger sizes, with 50, 100,

500, 1000, 2500, 3500, and 4500. The training and testing accuracies were evaluated using 5-fold cross validation.

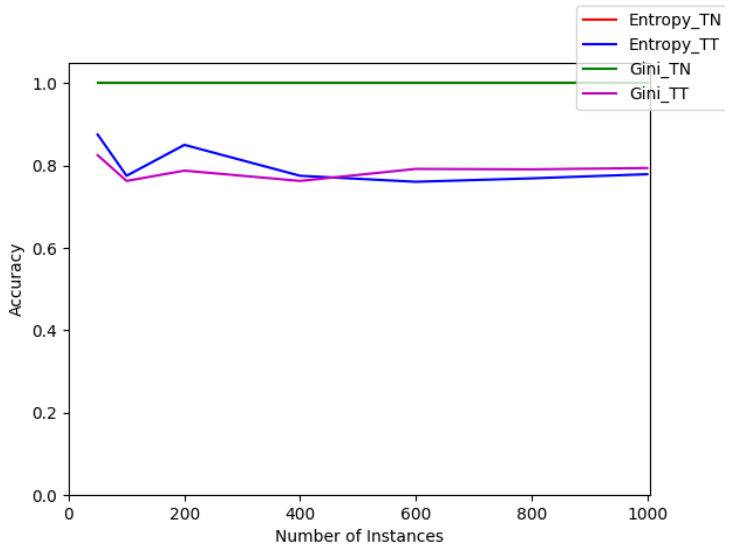

Figure 1: Performance of the artificial dataset with various instances and two impurity functions

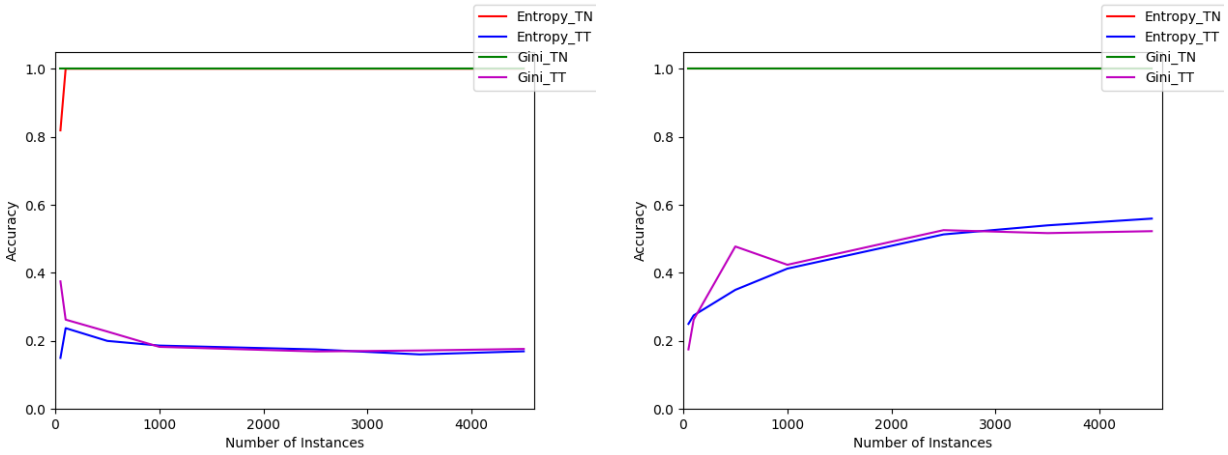

Figure 2: Performance of the EMG dataset(left) and optical digit dataset(right) with various instances and two impurity functions

For all three of the datasets, the training accuracy 100%, which for the decision trees, is a good observation as it proves the tree was properly built. However, both the EMG and optical digit dataset had disappointing testing accuracies.

It should be noted that decision trees are extremely vulnerable to overfitting the training dataset. The digit dataset has 10 classifications, while the EMG dataset had 20 classifications. In addition, both datasets from a large range for their attribute values. Therefore, the

testing accuracies from Figure 2 most likely reflects the complexity of both datasets. That is, the higher number of attribute values and classification leads to dataset that is vulnerable to overfitting. In contrast, the artificial dataset only has two attributes that range between 0 and 1 as well as two possible classifications, which means that it is not as vulnerable to overfitting.

One final note is the relative performance of the Entropy and Gini Index functions. For larger sizes, Entropy performed better on the optical digit dataset while the Gini Index performed better on the artifical dataset. For the EMG dataset, the performance better than impurities seem relatively indistinguishable. Entropy provides complex loss compared to the Gini Index. So, the relatively simple structure of the artificial dataset most likely utilized the Entropy function most effectively for smaller sizes. However, the complex stmore ructure of the optical dataset seemed to utilize the simpler nature of the Gini Index effectively.

4.2 Performance & Analysis of Feedforward Neural Networks

The following tables show the training and testing accuracy of a feedforward neural network for the datasets described in section 2. For all three datasets, the network had a single deep layer with roughly five times the amount of nodes(node_size) compared to the number of attributes. The sigmoid activation function was utilized in combination with one-hot encoding for the output layer. The total size of the training and testing sets combined was set to 100. The training and testing sets are split via 5-fold cross validation.

For the EMG and optical digit datasets, L2 regularization was integrated with the weight updates. In addition, the input vectors were normalized for these datasets.

Performance		
Alpha	Train. Accu	Test Accu
0.01	0.678125	0.675
0.025	0.712499	0.725
0.05	0.756251	0.75
0.075	0.765625	0.75
0.1	0.774999	0.75
0.25	0.759375	0.7875
0.5	0.746875	0.775

Table 1: Performance on the artifical dataset. $\lambda = 0$ and node_size = 25

Performance		
Alpha	Train. Accu	Test Accu
0.1	0.60625	0.3625
0.25	0.8	0.587499
0.5	0.88125	0.725001
0.75	0.915625	0.75

Table 2: Performance on EMG dataset. $\lambda = 0.0001$ and node_size = 40

Performance		
Alpha	Train. Accu	Test Accu
0.1	0.909375	0.775
0.25	0.71875	0.6
0.5	0.675	0.675
0.75	0.596874	0.48750004

Table 3: Performance on Optical Digit dataset. $\lambda = 0.0001$ and node_size = 320

For the artificial and EMG dataset, a higher learning rate seemed to provide the best accuracy. However, a lower learning rate provided the best accuracy for the dataset of optical digits. This phenomeon is most likely being caused by the number of attributes, classifications, and their effects on the gradients. The artificial and EMG datasets have 2 and 8 attributes in the input vector respectively, compared to the optical digit dataset's 64 attributes. This relatively low number of attributes most likely causes a lower magnitude in the weight gradients. So, the network needs to compensate with a higher learning rate to converge to a good truth function. This is not needed in the optical digits dataset. In fact, a higher learning rate most likely overestimates the truth function, which prevents it from converging.

The learning rate performances are not the only interesting takeaway from these results. For the artificial dataset, there is very little difference inbetween the training and testing accuracies. However, there is substantial difference inbetween the training and testing accuracies for the EMG and optical digit datasets. The artifical dataset not only has a low number of attributes, or it also has a low number of classifications. The nature of this dataset most likely causes low magnitude in the weight gradients, which consequently, provides small amounts of learning.

5 Conclusion

Supervised learning is a powerful for classification and regression prediction. There are numerous types of supervised learning models. However the performance of these models depend greatly on both the datasets they are trained on and the hyperparameters for each model.

As shown in this paper, a decision tree is a generally sufficient option for datasets with a low number of attributes and classifications. However, its vulnerability to overfitting becomes prevalent for more complex datasets. To address this problem, a neural network can be used instead to fit more complex datasets. Even then, a good choice of hyperparameters needs to be made in order to maximize the performance. For example, applying L2 regularization is good idea for complex datasets to reduce overfitting. In addition, the complexity of the dataset, or the number of attributes and classifications, can hint at a possible learning rate value.

References

[1] E Alpaydin and C Kaynak. Optical recognition of handwritten digits data set. *UCI Machine Learning Repository*, 9(11), 1998.

[2] Andrew Ng. Coursera: Machine learning, 2017.

[3] Luke Runnels. Project 1: K-nearest-neighbors, simulated annealing, and genetic algorithms. *URL: https://github.com/lar9482/CS-4253–Project-1*, 2023.

[4] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearsons, 4 edition, 2021.

[5] Theo Theodoridis. Emg physical action data set. *UCI Machine Learning Repository*, 2011.