EE 769 - Report for Course Project

Introduction to Machine Learning

Aman Butoliya
Electrical Engineering
IIT Bombay

Bhagyasree M
Aerospace Engineering
IIT Bombay

Shubham Namdev Kamble
Chemical Engineering
IIT Bombay

***Abstract*—Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention. The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly. Main two types of ML is 'Supervised' and 'Unsupervised'. For this work we have taken supervised learning into consideration. Three models; random forest, support vector machines and neural networks have been studied intensively and implemented using Python language without the usage of build-in functions. Implemented methods for a given data set have been validated by checking accuracy for each of the models.**

Key Words: Machine learning, random forest, SVM, neural network.

I. Introduction

Machine Learning is a branch of Artificial Intelligence. Machine Learning algorithms allow us to provide some data input to the computer and extract useful data or patterns from the data set, based on which we can make a better decision.Our goal is to implement Machine learning algorithms by just using simple mathematical operations. We have implemented Random Forest, support Vector Machine and Neural network on data set.

The report has been arranged in following manner, section 1 provides introduction about the problem statement, dataset used and its preprocessing. Section 2 provides background information about each models. Section 3 provides implementation steps used which is followed by result and then conclusion.

A. Objective

The objective of the project is to build Machine learning models from scratch without importing libraries in python. The models to train data were build by the authors from scratch based on the key concepts learnt and explored. The chosen models were Random forest, Support vector machines and Neural networks. The models were trained for a given data set and their accuracy was compared with each other and also with corresponding scikit-learn libraries in python which has inbuilt models. Based on the observations, conclude which model is better and how the efficiency of others can be improved.

B. Data preprocessing

The data set is about the churn Model of customers, and the end goal is to predict which feature make more impact on deciding whether customers should exit or not. It is a classification problem. The given data set is preprocessed to make it compatible for the ML model. Initially the correlation matrix was formed for dropping less correlated features. The features which doesn't contribute to the output were also dropped intuitively. Categorical feature values were converted to numerical type. The given data set was normalized, for the training to be more efficient.The preprocessing and the code development for the three models were done using python in google colaboratory. Entropy is used as the measurement for constructing binary decision trees which is a measurement of uncertainty.

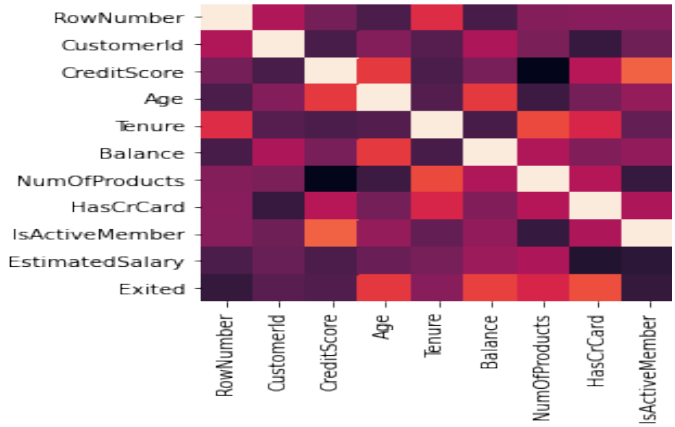

Fig. 1: Correlation Matrix

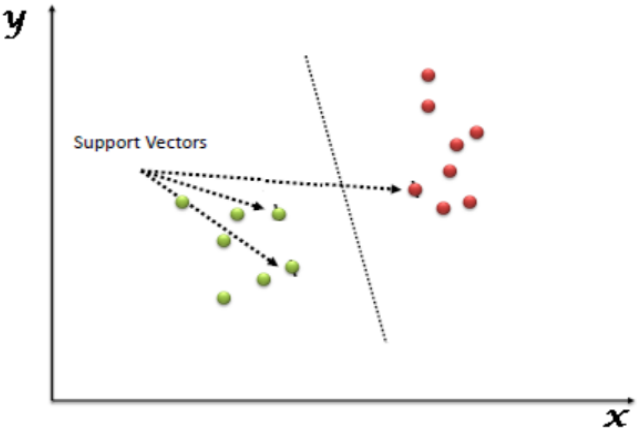

Fig. 2: Hyperplane

II. Background Information

This section focuses on the key concepts used in developing the models and also the theory pertaining to it.

A. Random Forest

Random forest is an ensemble learning method used in classification and regression problems. The model is used to train for the given data set which is a classification problem. This model works as a set of decision tress where each fit on a sub sample of data. The main advantage of random forest is that it is non-parameteric and doesn't require parameter tuning. It also doesn't require huge matrix operations since they are not gradient based. The model works with binary decision trees for a classification problem and in the case of regression they output mean prediction.

B. Support Vector Machine

Support Vector machine (SVM) is supervised Machine learning (ML) algorithm. It can be used for both classification and regression. However, it is widely used for classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well as shown below.

1) How does it work?: The main problem is to segregate the two classes using hyperplane. Lets look at how to identify right hyperplane?.

- **Identify the right hyper-plane:**
  In this case, we have three hyperplane and to choose right hyperplane we have to maximize the distance between nearest data point (either class) and hyper-plane. This distance is called as **Margin**.

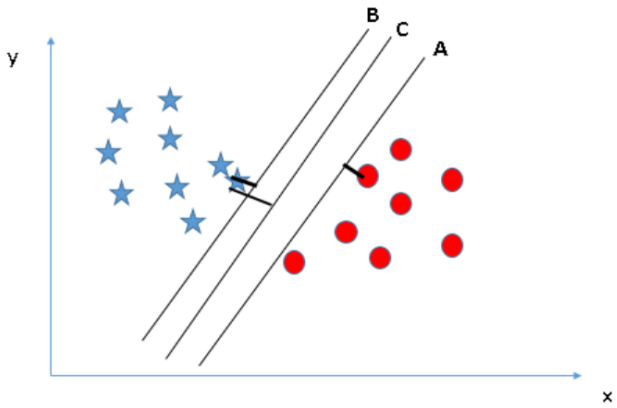

Fig. 3: SVM

As can be seen from above figure, margin for hyperplane C is high as compared to both hyperplane B and A. Hence, we choose C.

Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.

- **Support Vectors:**
  Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM.

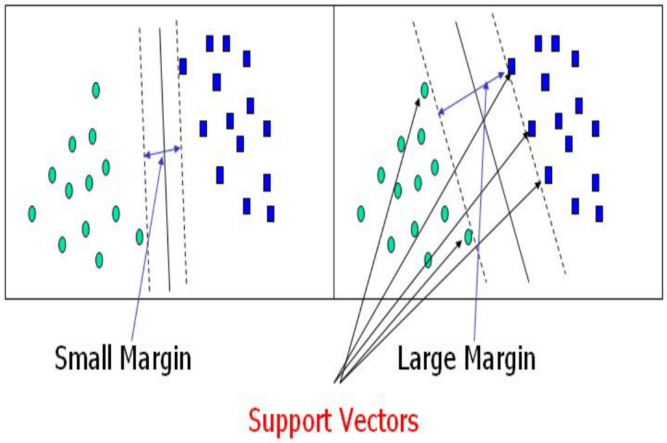

Fig. 4: Support Vectors

C. Neural Network

Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. This patterns are numerical, contained in vectors, into which all real-world data, be it images, sound, text or time series, must be translated. Neural networks can also extract features that are fed to other algorithms for clustering and classification. Network consists of several layers i.e. 'Stacked Neural Network' is called as 'Deep Learning network'.

These layers are made up of nodes. A node is just a place where computation happens. A node combines input from the data with a set of coefficients, or weights, that either amplify or dampen that input, thereby assigning significance to inputs with regard to the task the algorithm is trying to learn; e.g. which input is most helpful is classifying data without error? These input-weight products are summed and then the sum is passed through a node's so-called activation function, to determine whether and to what extent that signal should progress further through the network to affect the ultimate outcome, say, an act of classification. If the signals passes through, the neuron has been "activated". Below diagram best illustrate what one node might look like:

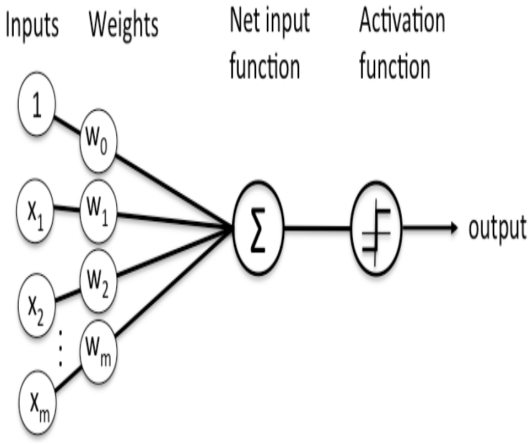

Fig. 5: A node in neural network

III. Execution

Here, we have explained about the steps we have followed to execute selected problem statement.

A. Random Forest

The first step is to preprocess the data as mentioned previously and to split the data for training and predicting respectively. The next step would be to quantify the uncertainty through entropy which takes the following form in a binary classification problem.The algorithm works such that it finds a value that minimizes entropy.

$$H(x) = -plog_2p - (1-p)log_2(1-p)$$

The entropy before and after the split is compared to obtain the information gain. Another reason random forest is so powerful is because of bootstrapping. Each decision tree will be constructed on a bootstrapped subset of our data.i.e. some points in our data will be selected more than once and some

will not be selected at all. The other feature of random forest is the process of bagging where each node of a growing tree looks at every value in the bootstrapped sample in every feature to find the best split in the data at the particular node. This is done for all trees. The random forest algorithm is depicted below.

Suppose we have the following data $\{(x_1, y_1), (x_2, y_2), .., (x_n, y_n)\}$ where each xi represents a feature vector $[x_{i1}, x_{i2}, ..., x_{im}]$ and let B be the number of trees we want to construct in our forest. We will do the following,

1) for b=1 to B:
   - Draw a bootstrap sample of size n from the data.
   - Grow a decision tree T_b from our bootstrapped sample, by repeating the following steps until the each node consists of 1 class only or until we've reached the minimum node size min_{size} specified beforehand.
     a) Sample $m = \sqrt{p}$ features (where p is the number of features in our data set).
     b) Compute the information gain for each possible value among the bootstrapped data and m features.
     c) Split the node into 2 children nodes.
2) Output the ensemble of trees T_1^B.

B. Support Vector Machine

The SVM (Support Vector Machine) is a supervised machine learning algorithm typically used for binary classification problems. the algorithm finds a hyper-plane (or decision boundary) which should ideally have the following properties:

- It creates separation between examples of two classes with a maximum margin.
- Its equation $(w.x + b = 0)$ yields a value ≥ 1 for examples from +ve class and ≤ -1 for examples from -ve class.

The very first step is to perform data pre-processing as explained in section 1.2. Our objective is to find a hyper-plane that separates +ve and -ve examples with the largest margin while keeping the misclassification as low as possible. To achieve this, we will minimize the cost/objective function shown below.

$$J(w) = \frac{1}{2}||w||^2 + C\left[\frac{1}{N}\sum_i^n max(0, 1 - y_i * (\mathbf{w} \cdot x_i + b))\right]$$

In the training phase, Larger C results in the narrow margin (for infinitely large C the SVM becomes hard margin) and smaller C results in the wider margin.

We have used another version of cost function that looks like this:

$$J(w) = \frac{\lambda}{2}||w||^2 + \frac{1}{N}\sum_i^n max(0, 1 - y_i * (\mathbf{w} \cdot x_i + b))$$

Larger λ gives a wider margin and smaller λ results in the narrow margin (for infinitely small λ the SVM becomes hard margin). Now, we will move on to gradient which will be used in the training phase to minimize cost function.

Here, we have used 'Stochastic Gradient Descent (SGD)' to achieve our objective i.e. to minimize cost function. For this we have to:

- Minimize $||w||^2$ which maximizes margin $\left(\frac{2}{||w||}\right)$.
- Minimize the sum of hinge loss which minimizes misclassifications i.e.(Minimize Hinge loss function).

$$max(0, 1 - y_i * (\mathbf{w} \cdot x_i + b))$$

To implement SGD, we can run loop many number of times but each iteration would cost us time and extra computation. So, We have terminated the loop when our stoppage criterion is met which is that We will stop the training when the current cost hasn't decreased much as compared to the previous cost.

After training the model using SGD we finally got the optimal weights $\mathbf{w}^*$ which defines the best possible hyper-plane separating two classes. The accuracy is shown later in the result section.

C. Neural Network

Here, we are implementing a two layer neural network (i.e. one input layer, one output layer and one hidden layer). To implement this, first we have created a neural network class and then during initialization,some variables to hold intermediate calculations. Implemented architecture accepts 13 input features, uses 8 nodes in the hidden layer, and finally uses 1 node in the output layer.

- **The Weights and Biases:**
  Weights and biases are the learn-able parameters that help a neural network correctly learn a function. We created a function (init_weights) to initialize the weights and biases as random numbers. These weights are initialized from a uniform random distribution and saved to a dictionary called params. One thing to notice is that, The first weight array (W1) will have dimensions of 13 by 8—this is because we have 13 input features and 8 hidden nodes, while the first bias (b1) will be a vector of size 8 because we have 8 hidden nodes. The second weight array (W2) will be a 10 by 1-dimensional array because we have 10 hidden nodes and 1 output node, and finally, the second bias (b2) will be a vector of size 1 because we have just 1 output.
- **The Activation Function:**
  Now that we have initialized the weights and biases, its time to consider 'Activation Function' into account. There are many types of activation functions for examples Sigmoid, ReLU, tanh, Leaky ReLU, and so on. Each activation function has its pros and cons, but the ReLU function has been shown to perform very well, so for our work, we used the **ReLU** function.

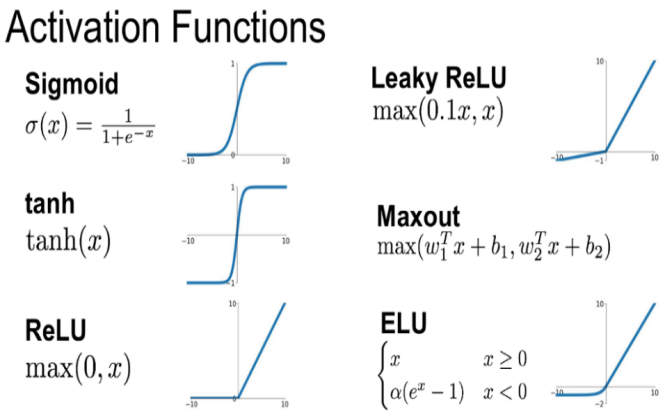

Fig. 6: Different activation functions used in deep learning

ReLU (Rectified Linear Unit) is a simple function that compares a value with zero. That is, it will return the value passed to it if it is greater than zero; otherwise, it returns zero.
In summary, the hidden layer receives values from the input layer, calculates a weighted sum, adds the bias term, and then passes each result through an activation function—in our case a ReLU. The result from the ReLU is then passed to the output layer, where another weighted sum is performed using the second weights and biases. But then instead of passing the result through another activation function, it is passed through the output function. Since, it is binary classification problem we have used 'Sigmoid Function' as output function.

- **The Loss Function:**
  The most important thing to consider is the 'Loss Function' of neural network. The loss function is a way of measuring how good a model's prediction is so that it can adjust the weights and biases. A loss function must be properly designed because it tells you whether a prediction made is far or close to the true prediction. The choice of the loss function is dependent on the task and for classification problems, we have used cross-entropy loss given by equation below:

$$CE = -\sum_{i}^{C} y_i \log(\hat{y_i})$$

  Where C is the number of classes, y is the true value and y_hat is the predicted value. For a binary classification task (i.e. C=2), the cross-entropy loss function becomes:

$$CE = -\sum_{1}^{2} y_i \log(\hat{y_i}) = -y_1 \log(\hat{y_1}) - (1-y_1) \log(1-\hat{y_1})$$

- **Going Forward: Forward Propagation:**
  Forward propagation is the name given to the series of computations performed by the neural network before a prediction is made. In our two-layer network, following computations were performed for forward propagation:
  - Compute the weighted sum between the input and the first layer's weights and then add the bias: **Z1 = (W1 * X) + b**.
  - Pass the result through the ReLU activation function: **A1 = Relu(Z1)**.
  - Compute the weighted sum between the output (A1) of the previous step and the second layer's weights—also add the bias: **Z2 = (W2 * A1) + b2**.
  - Compute the output function by passing the result through a sigmoid function: **A2 = sigmoid(Z2)**.

- And finally, compute the loss between the predicted output and the true labels: **loss(A2, Y)**.

- **A Step Backward: Backpropagation:**
  Backpropagation is the name given to the process of training a neural network by updating its weights and bias. A neural network learns to predict the correct values by continuously trying different values for the weights and then comparing the losses. If the loss function decreases, then the current weight is better than the previous, or vice versa. This means that the neural net has to go through many training (forward propagation) and update (backpropagation) cycles in order to get the best weights and biases. This cycle is what generally refer to as the training phase, and the process of searching for the right weights is called **Optimization**. To perform backpropagation in neural network, we have followed the steps listed below:
  - calculate the derivative of the loss with respect to the output yhat as:

$$\frac{dl}{\hat{y}} = \frac{-y}{\hat{y}} + \frac{(1-y)}{(1-\hat{y})}$$

  - calculate the derivative of sigmoid activation with respect to the loss: After calculation the end result that we get is

$$d(sigmoid) = sigmoid(x) * (1 - sigmoid(x))$$

  - Then, calculate the derivative of the loss wrt Z2:

$$\frac{dl}{z2} = \frac{dl}{\hat{y}} * d(sigmoid)$$

  - But $Z2 = (W2 * A1) + b2$. Therefore,

$$\frac{dl}{A1} = w2 * \frac{dl}{z2},$$
$$\frac{dl}{w2} = A1 * \frac{dl}{A1},$$
$$\frac{dl}{b2} = \frac{dl}{z2}$$

  - And A1=Relu(z1). Therefore,

$$\frac{dl}{Relu} = d(Relu)$$
$$\frac{dl}{z1} = \frac{dl}{A1} * d(Relu)$$

  - Z1 = (W1 * X) + b. Therefore,

$$\frac{dl}{w1} = X * \frac{dl}{Z1}$$
$$\frac{dl}{b1} = \frac{dl}{Z1}$$

- **Optimization and Training of the Neural Network:** Previously we used calculus to compute the derivatives of the weights and biases with respect to the loss. The model now knows how to change them. To automatically use this information to update the weights and biases, a neural network must perform hundreds, thousands, and even millions of forward and backward propagations. That is, in the training phase, the neural network must perform the following:
  - Forward propagation
  - Backpropagation
  - Weight updates with calculated gradients
  - Repeat

  What we're basically doing here is subtracting the derivative multiplied by a small value called the learning rate. The learning rate is a value that tells our neural network how big the update should be.

IV. Results

We implemented Random forest Classifier, Support vector Machine and Neural Network on Churn Modelling data set from scratch and following are the accuracy we obtained for each of the model used respectively.

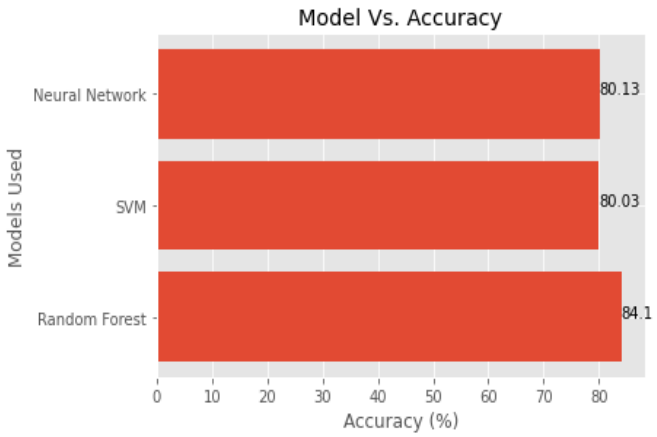

Fig. 7: Accuracy for different models used

We obtained higher accuracy for Random Forest Classifier as expected than other two classifier models, as it builds multiple decision trees and merge them together to get more accurate prediction.As we have used neural network with two 1 hidden layer and last output layer, that's why maybe it has lower accuracy than Random Forest. If we form a deep neural network then it might give more accuracy. So Random Forest Classifier turns out to be best classifier for churn modelling.

V. Conclusion

The three machine learning models i.e. random forest, support vector machines and neural networks were built from scratch and were used for training the given data set which is a classification problem. The accuracy of the three models were compared with each other and were also compared with the corresponding scikit-learn libraries to validate the results. From this we can conclude that Random forest classifier always give higher accuracy than Support Vector Machine and Two layer neural network.Basic reason for this is a collection of decision tree, these multiple decision trees merges together to give more accurate and more stable prediction.

VI. Limitation and future work

Lack of computation power is the major limitation that we faced during implementation. Implemented model was taking more than 2 hours for training. We think chosen dataset is the reason for this problem. For simplicity and speed one can go for simpler dataset or some advanced techniques should be used for the training and optimization process. For such a large dataset we could use CUDA which is parallel computing platform used for processing large blocks of data in Machine Learning.

Acknowledgment

The Authors would like to express his gratitude towards Professor Amit Sethi who provided the necessary help and guidance required to summarize this report. They would also like to thank him for providing clarifications wherever required.

VII. References

- Class Notes
- Lecture Videos
- https://fordcombs.medium.com/svm-from-scratch-step-by-step-in-python-f1e2d5b9c5be