MACHINE LEARNING TOOLBOX

Dr. Pankaj Agarwal Professor, Akshay Bhasin, Rohit Keshwani, Aman Verma and Suryansh Kaushik

B.Tech CS 4th Yr Department of Computer Science, IMS Engineering College, Ghaziabad, U.P, India

ABSTRACT

This paper presents a review & performs a comparative evaluation of few known machine learning algorithms in terms of their suitability & code performance on any given data set of any size. In this paper, we describe our Machine Learning ToolBox that we have built using python programming language. The algorithms used in the toolbox consists of supervised classification algorithms such as Naïve Bayes, Decision Trees, SVM, K-nearest Neighbors and Neural Network (Backpropagation). The algorithms are tested on iris and diabetes dataset and are compared on the basis of their accuracy under different conditions. However using our tool one can apply any of the implemented ML algorithms on any dataset of any size. The main goal of building a toolbox is to provide users with a platform to test their datasets on different Machine Learning algorithms and use the accuracy results to determine which algorithms fits the data best. The toolbox allows the user to choose a dataset of his/her choice either in structured or unstructured form and then can choose the features he/she wants to use for training the machine We have given our concluding remarks on the performance of implemented algorithms based on experimental analysis.

KEYWORDS

Machine Learning, Neural Networks, Backpropagation, Decision Tree (C4.5), SVM (Support Vector Machines), K-means Clustering, K-nearest Neighbors.

1. INTRODUCTION

Machine Learning is simply the ability of a machine to learn from some set of data and make inferences on the basis of that data. As explained by Tom Mitchell **[1]** machine learning can be best defined as:

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance measure at tasks in T, as measure by P, improves with experience E."

The beauty of machine learning models is that when they are exposed to new data, they are able to independently adapt. The need for fast and quick way to try different datasets on different algorithms with different parameters is still seen in the market. The use of this kind of tool can be important to researchers, students, teachers or professionals. The goal of the machine learning toolbox is to provide an easy way to upload datasets in text file format and apply various machine learning algorithms. The machine learning toolbox will provide options such as choosing both training and testing dataset to be the same or splitting the training and testing

DOI:10.5121/mlaij.2016.3303

datasets in some proportions. The toolbox will provide suitable options in different algorithms such as providing a field for knowing the number of hidden neurons and a field for knowing the learning rate in case of neural network. This way the user can interact with the toolbox easily, try out different patterns, and see which configuration of an algorithms suits the dataset well.

Machine Learning Toolbox is a research project that is basically meant to provide the user with a complete package where he/she can use it to know how different machine learning approaches and algorithms would work on a particular dataset. The toolbox would allow the user to provide a dataset either in structured or unstructured form and one can choose the features he/she wants to use for training the machine. In this manner, the user will have a complete control in the comparison process.

2. SUPERVISED MACHINE LEARNING APPROACHES

Machine Learning ToolBox includes the usage of various supervised learning techniques. The search for algorithms that can reason from externally supplied instances to product a general hypothesis is known as supervised machine, which then make predictions about future instances. In other words, the main aim of supervised learning is to build a model of the distribution of class labels by using predictor features. The resulting classifier model is then used to assign class labels to the testing datasets where the values of the predicting data are known, but the value of the class label is unknown. In the present paper, we have focused on the algorithms necessary to do this. In particular, this work is concerned with classification problems in which the output of instances acknowledges only continuous, real values **[2]**. Table 1 gives an example of iris dataset format with four features expressed in real values & three possible output classes expressed numerically.

Table 1 Example of Iris Data Set for applying supervised learning algorithm

Feature 1	Feature 2	Feature 3	Feature 4	Known output class
sepal length in cm	sepal width in cm	petal length in cm	petal width in cm	Iris Setosa / Iris Versicolour/ Iris Virginica

Inductive machine learning is the process of creating a classifier model that can later be used to predict for new datasets. The process of applying supervised machine learning to a real-world problem is described in Figure 1.

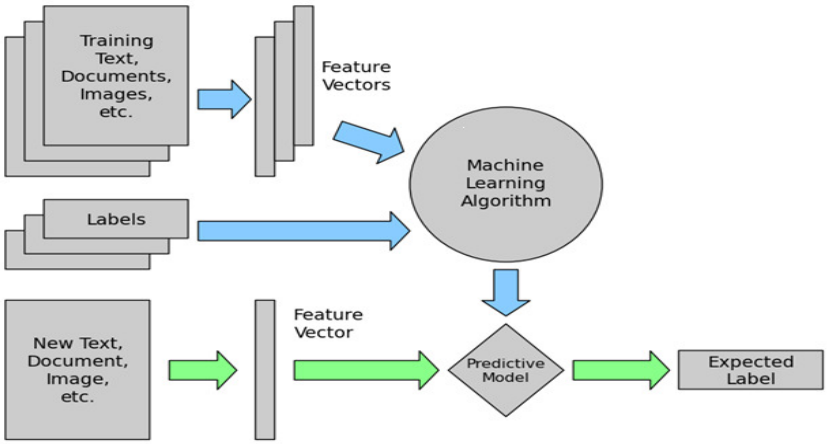

Figure 1 Supervised Learning Flow Diagram

2.1 Naïve Bayes Classifier

Naive Bayes techniques are a set of supervised learning algorithms based on Bayes' theorem with the assumption of independence between every pair of features. Naive Bayes is a simple method for constructing classifiers: the models assign class labels to problem instances, represented as vectors of feature values, where the output class labels predicted from some finite set[3]. Naive Bayes requires a small amount of training data to estimate the parameters necessary for classification

Given a class variable y and a dependent feature vector x_1 through x_n, Bayes' theorem states the following relationship:

$$P(y \mid x_1 \ldots\ldots x_n) = \frac{P(y)P(x_1 \ldots\ldots x_n \mid y)}{P(x_1 \ldots\ldots x_n)}$$

The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i \mid y)$. Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods.

2.2 K-nearest Neighbors Classifier

KNN is a very popular classification algorithm that produces good performance characteristics and takes lesser time for training the datasets. *k*-Nearest Neighbors algorithm (or *k*-NN for short) is a non-parametric method used for classification and regression. *k*-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The *k*-NN algorithm is among the simplest of all machine-learning algorithms.

However, there are a few disadvantages of KNN algorithm. First, all the neighbors are treated equally in the standard KNN algorithm. Second, the imbalanced data problem can occur in KNN algorithms. Large classes always have a better chance to win [4].

Step 1) Search the K training instances that are closest to unknown instance.

Step 2) Pick the most commonly occurring class label for these K instances.

In the classification phase, *k* is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the *k* training samples nearest to that query point. A commonly used distance metric for continuous variables is Euclidean distance. For discrete variables, such as for text classification, another metric can be used, such as the **overlap metric** (or Hamming distance)

2.3 Decision Tree Classifier

Decision Tree Classifier is a simple and widely used classification technique. It applies a simple approach to solve the classification problem. Each time it receives an answer, a follow-up question is asked until a conclusion about the class label of the record is reached. A decision tree is a tree like structure where in decisions are taken based on values of some attributes and thus further branching is done. Decision tree can classify a data to some class label by moving from the root node and comparing the attribute values as specified in the nodes until a leaf node is reached. The leaf node contains the class label. It aims is the partition of a dataset into groups as homogeneous as possible in terms of the variable to be predicted **[5]**.

Creating a optimal decision tree is a key problem in decision tree classifier. In general, for many cases a decision tree can be constructed from a given set of attributes. While in some cases, finding the optimal tree is computationally infeasible because of the exponential size of the search space.

However, various efficient algorithms have been developed to construct a reasonably accurate, suboptimal decision tree in a reasonable amount of time. These algorithms usually employ a greedy strategy that grows a decision tree by making a series of locally optimum decisions about which attribute to use for partitioning the data. For example, Hunt's algorithm, ID3, C4.5, CART, SPRINT are greedy decision tree induction algorithms.

2.4 SVM Classifier

The support vector machine (SVM) is a training algorithm for classification rule from the data set which trains the classifier; it is then used to predict the class of the new sample. SVM is based on the concept of decision planes that define decision boundary and point to form the decision boundary between the classes called support vector threat as parameter **[6]**. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on. More formally, a support vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks as shown below.

.

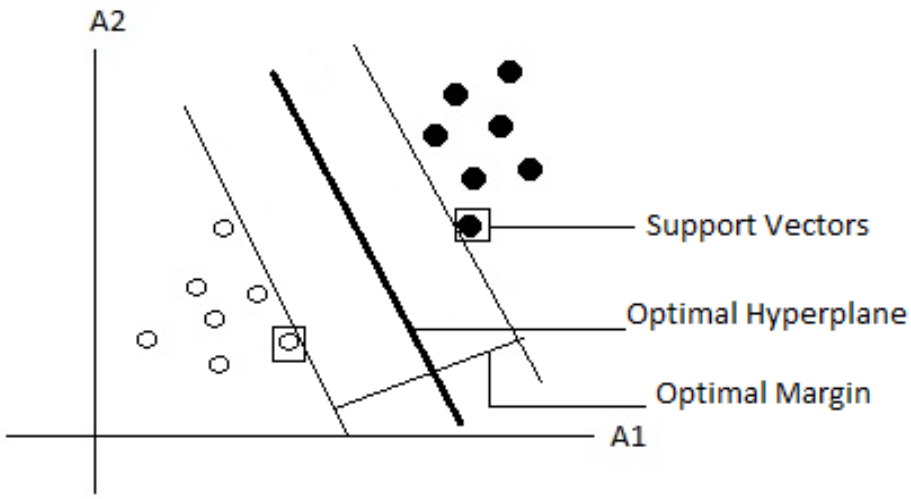

Figure 2 SVM Illustration

2.5 Artificial Neural Network Classifier

Artificial Neural Network is a network where nodes are interconnected and they represent some value and are connected to each other via some weighted edges. This network is parallel in the sense that the neurons are collected at different layers and adjacent layer neurons are connected. ANN was made to mimic the working of a biological nervous system where each neuron contributes some amount to the final output.

Neural networks are made up of layers, which are in turn made up of a number of interconnected nodes which contain an activation function. The nodes from input layer are connected to the nodes of the hidden layer with weighted edges. The hidden layers are linked to an output layer where the answer is received. It is represented as:

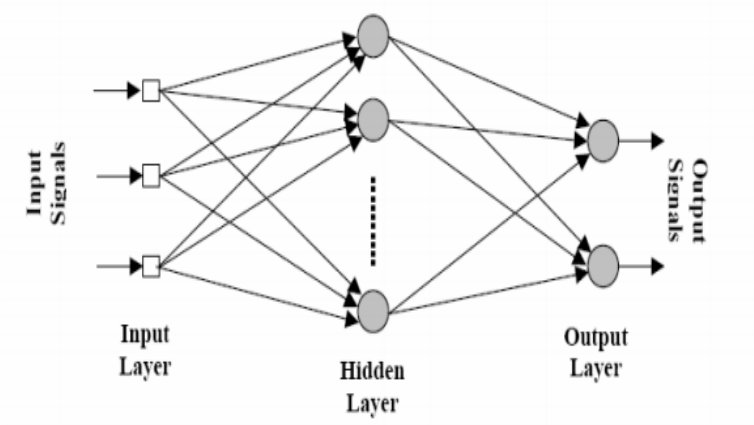

Figure 3 Neural Network Illustration

These networks have been established to be universal approximates of continuous/discontinuous functions and therefore they are suitable for usage where we have some information about the

input-output map to be approximated. A set of data (Input-Output information) is used for training the network. Any input can be given once the network has been trained and it will give an output, which would correspond to the expected output from the approximated mapping.

The activation function used is the log-sigmoid function as given in can be expressed as: -

$$\Phi(a) = \frac{1}{1 + e^{-a}}$$

Where

$$a = \sum_{i=1}^{N} \mathbf{wx}$$

w array contains the weights of the edges and x array contains the outputs of the previous layer. For the hidden layer x correspond to the input of the network while for the output layer x correspond to the output of the hidden layer. The network is trained using the error back propagation algorithm. The weight update rule as given in can be expressed as: -

$$\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \eta \delta_j(n) y_i(n)$$

where α is called the momentum constant, η is the learning rate, $\Delta w_{ji}(n)$ is the correction applied to the synaptic weight connecting the output of neuron i to the input of neuron j at iteration n, δ_j (n) is the local gradient at nth iteration, y_i (n) is the function signal appearing at the output of neuron i at iteration n. The percentage error is specified beforehand according to a problem and thus the network tries to converge to that error by using back propagation technique. The advantage of using neural network approach is that they are robust with noisy data, adaptable, nonparametric, robust with respect to node failure and empirically shown to work well for many problem domains.

3. PROPOSED MACHINE LEARNING TOOLBOX

The machine learning toolbox is developed using python as the core programming language and the main aim was to provide a code free environment to a user so that he/she can easily try out different algorithms with different parameters and thus can choose the algorithm that fits the datasets best.

It takes in the dataset in arff format. An ARFF (Attribute-Relation File Format) file is an ASCII text file that describes a list of instances sharing a set of attributes. Two datasets that were used here for comparative analysis are Iris dataset and a Pima Indian Diabetes dataset.

One of them is Iris dataset or Fisher's Iris data set. It is a data set produced by Ronald Fisher that he used in his paper “The use of multiple measurements in taxonomic problems” as an example of linear discriminant analysis. There are 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the

length and the width of the sepals and petals, in centimeters. Based on the combination of these four features, the task is to distinguish the species from each other. **[8]**

The other dataset is Pima Indian Diabetes dataset, a determination of whether a given woman is diabetic**[9]**. There are 7 predictors, including personal history and medical measurements and of the women were Pima Indians.

The toolbox provides algorithms like Naïve Bayes, SVM, Neural Network, Decision Tree and K-nearest neighbors. It provides an easy to use GUI, which takes in input basic parameters for each algorithm. For example, in neural network, there are field to enter learning rate, momentum, number of hidden neurons and the maximum epochs up to which it should be trained.

4. RESULTS

Accuracy of Results

Naïve Bayes Algorithm

Naïve Bayes was run on iris dataset having 150 training values and was tested on the same 150 test cases. The accuracy was seen to be 96%. Naive Bayes was also run on diabetes dataset having about 750 training values and was tested 150 new values and accuracy between was 70-76% was observed

K-nearest Neighbors Algorithm

K-nearest neighbors was run on iris dataset and diabetes datasets and showed the following results for various training and testing datasets partition:

Table 2 Accuracy results for iris dataset on K-nearest neighbors

K	**Split(TRAIN:TEST)**		
	100:100	80:20	70:30
1	100%	96.66%	97.77%
3	96%	96.66%	95.55%

Table 3 Accuracy results for diabetes dataset on knn

K	**Split(TRAIN:TEST)**		
	100:100	80:20	70:30
1	100%	65.58%	66.23%
3	85.41%	70.12%	70.56%

SVM Algorithm

SVM classifier was run on iris dataset and diabetes datasets and showed the following results for various training and testing datasets partition:

Table 4 Accuracy results for iris and diabetes dataset on SVM

Dataset	Split(TRAIN:TEST)		
	100:100	80:20	70:30
Iris	82%	93.33%	95.55%
Diabetes	74.60%	67.53%	62.77%

Decision Tree Algorithm

Decision Tree classifier was run on iris dataset and diabetes datasets and showed the following results for different training and testing datasets partition:

Table 5 Accuracy results for iris and diabetes dataset on Decision Tree

Dataset	Split(TRAIN:TEST)		
	100:100	80:20	70:30
Iris	100%	70.97%	73.91%
Diabetes	100%	53.55%	53.45%

Neural Network Backpropagation Algorithm

Neural Net classifier was run on iris dataset and diabetes datasets and showed the following results for various training and testing datasets partition:

Table 6 Accuracy results for iris and diabetes dataset on Neural Network

Dataset	Hidden Layers	Split(TRAIN:TEST)		
		100:100	80:20	70:30
Iris	4	76%	75%	70.47%
	5	97.33%	75%	92.35%
Diabetes	4	65.10%	65.31%	64.05%
	5	65.10%	65.53%	65.41%

Summarized Result

Running different classifiers on different datasets with different parameters show a lot of variation in the accuracy results on iris dataset. This can be summarized as:

Table 7 Comparison of results on iris and diabetes datasets on different algorithms

Algorithm	Average Accuracy	Remarks
Naïve Bayes	88%	The fastest of all algorithms with a higher accuracy rate.
K-nearest Neighbors	96%	This algorithm worked best on iris dataset.
SVM	84%	Slower than most algorithms and low accuracy due to random choice of attributes.
Decision Tree	80%	Moderate time and accuracy results.
Neural Network	78%	Slowest of all algorithms and works well with low number of hidden neurons.

5. CONCLUSIONS AND FUTURE SCOPE

We can easily apply implemented ML algorithms on any data set with set of numerical features and setting the appropriate parameters through our GUI. This can be surely a useful tool for researchers/students who wish to apply a ML algorithm on his/her dataset and understand which algorithm works better.

We are currently working on adding many more ML algorithms and adding graphical interpretations to compare the effectiveness of algorithms. It is expected that a fully developed toolbox available on web will also do commercially well. At least we were unable to find similar kind of toolbox on web. We will also be including unsupervised methods in the upcoming versions.

REFERENCES

1. T. M. Mitchell, “Machine Learning,” McGraw Hill, 1997.
2. Ch. M. Bishop, “Pattern Recognition and Machine Learning,” Springer, 2006.
3. P. Langley, W. Iba., and K. Thompson. Decision making using probabilistic inference methods. In Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence (UAI’92), pages 399–406, San Mateo, CA, 1992.
4. Nitin Bhatia, vandana” Survey on nearest neighbor techniques” IJCSIS, Vol 80, no 2(2010).
5. HSSINA, Badr et al. "A Comparative Study Of Decision Tree ID3 And C4.5". International Journal of Advanced Computer Science and Applications 4.2 (2014): n. pag. Web.
6. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). Support Vector Machines. In An Introduction to Statistical Learning (pp. 337-372). Springer New York.
7. S.B. Kotsiantis, Supervised Machine Learning: A Review of Classification Techniques, Informatica 31(2007) 249-268, 2007.

8. Wikipedia (2012). Iris flower data set. Retrieved 18 April, 2016, https://en.wikipedia.org/wiki/Iris_flower_data_set.
9. https://archive.ics.uci.edu/ml/datasets.html

ACKNOWLEDGEMENT

Dr. Pankaj Agarwal is current associate as Professor & Head, Department of Computer Science & Engineering, IMS Engineering College,Ghaziabad, U.P, India. He has more than 15 years of teaching & more than 8 years of research experience. Dr. Agarwal has published research publications in more than 30 International Journals & Conferences of repute. He has also authored five books on subjects including Algorithm Analysis, Software Project Management, .NET, Cyber Laws, Computer Programming. His current areas of interest includes softcomputing, Machine learning & data analytics. I would like to appreciate my team of students including Akshay, Rohit, Aman & Suryansh, studying in B.Tech 4th year who took up this project and completed the assigned tasks as per the desired expectations. We plan to add many new features to this toolbox.