Project 3

Classification Algorithms

Sagar Dhamija (sagardha : 50169364)

Varun Khandelwal (varunkha : 50168936)

Contents

Classification

Classification is a process of creating models(functions) based on the training data to describe and distinguish the concepts and classes for future prediction. There are different methods that can be utilized to perform classification, a few of them are listed below:

- K-Nearest Neighbor
- Decision Trees
- Naïve Bayes
- Rule-Based classification
- Logistic Regression
- Support Vector Machine
- Ensemble Methods

K-Nearest Neighbor or K-NN classification

In K-NN method of classification, the output is a class membership. In this method, the classification of the object is based on the majority of votes obtained by its neighbors, resulting in the object being assigned to the most common class among its k nearest neighbors where k is a positive integer, typically small. The value k = 1 simply results in the object being assigned to the class of that single nearest neighbor.

Implementing K-NN classification requires three things:

- The set of stored records
- Distance Metric to compute distance between records
- The value of k, the number of nearest neighbors to retrieve

In order to classify an unknown record, we need to follow the steps as mentioned below:

- Compute distance to other training records
- Identify k nearest neighbors
- Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

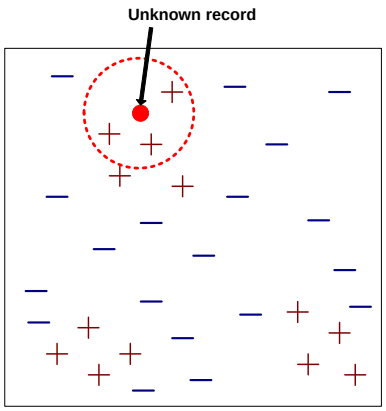
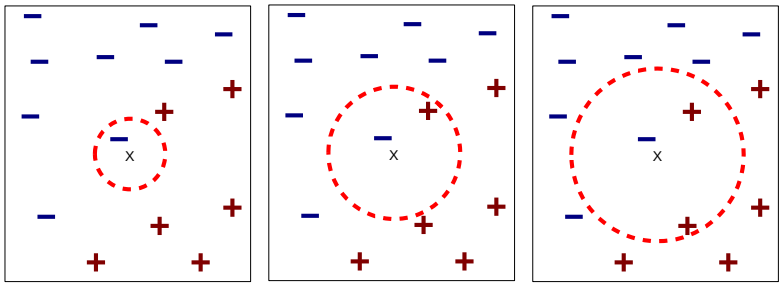


Fig.1



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

Fig.2

The above figure explains what exactly K-NN means. K-nearest neighbors of a record x are data points that have the k smallest distance to x.

To compute the distance between two points we can make use of the Euclidian distance:

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

Now, in order to determine the class from nearest neighbor list:

- take the majority vote of class labels among the k-nearest neighbors
- Weigh the vote according to distance
    - weight factor, w = 1/d2

Advantages:

- K-NN classification algorithm is robust to noisy training data (especially if we use inverse square of the weighted distance as the distance)
- It is effective for even very large data sets

Disadvantages

- Choosing the value of k:
- If k is too small, sensitive to noise points
- If k is too large, neighborhood may include points from other classes
- Scaling Issues:
    - In order to prevent the distance measure being dominated one of the attributes, the attributes need to be scaled
- K-NN classifiers are Lazy learners:
    - It doesn't build models explicitly
    - Also, classifying unknown records is relatively expensive

Our Implementation:

1. We have implemented our code in Python.
2. Our code is divided into many functions:
    a. calculate_distance()

Calculates the distance of the test data points against every training data point

b. find_label()
Finds k nearest neighbours for each test data point and then assigns the maximum appearing label to the test data point

c. calculate_measure()
Calculates various measures like accuracy, precision, recall and f-measure

d. Main Code
We have shuffled the data and applied k-fold to get k sets of testing and training data. We then called the corresponding functions and then displayed the measures.

Choice Description:

1. K: We chose k as 5. If k is too small, it is sensitive to noise and if k is too large, it may include points from other classes. Also, we prefer k to be odd so that there is no voting issue. However, in case someone prefers an even value of k, we have used distance to weigh the vote to break the tie.
2. Distance: We chose Euclidean distance for continuous and 0-1 loss for categorical data. However, we have normalized each individual column distance before adding to each other so that every column gets same preference (no bias)
3. Categorical Data: We have used 0-1 loss function to calculate distance for categorical data. If both data points have the same category, then their distance from each other is 0 else it is 1.
4. Continuous data: We have used Euclidean distance for continuous data points.

Decision Trees

Decision Tree- a sort of decision support tool utilizes a tree-like graph or model of decision with their possible consequences, including chance event outcomes, resource costs, and utility. It's more or less a way to display an algorithm

Decision trees usually find their application in operational research specifically to aid decision analysis, ease the process of identifying a strategy resulting in goal completion. But most importantly they are a popular tool in machine learning.

In other words, a decision tree represents a flowchart like structure where every internal node represents a "test" on an attribute (e.g. A 'refund' attribute can have two possibilities- 'Yes' or 'No'), every branch represents the outcome of the test and every leaf node represents a class label (decision taken after computing all attributes). The classification rules are represented by the path taken from root till the leaf.

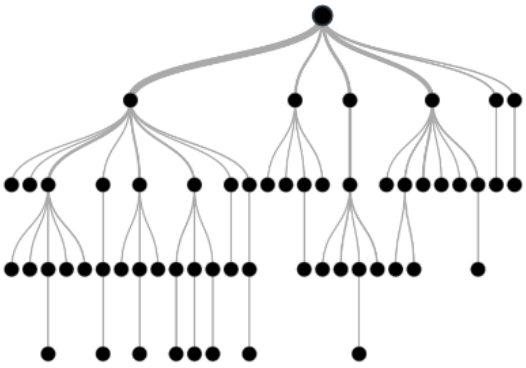
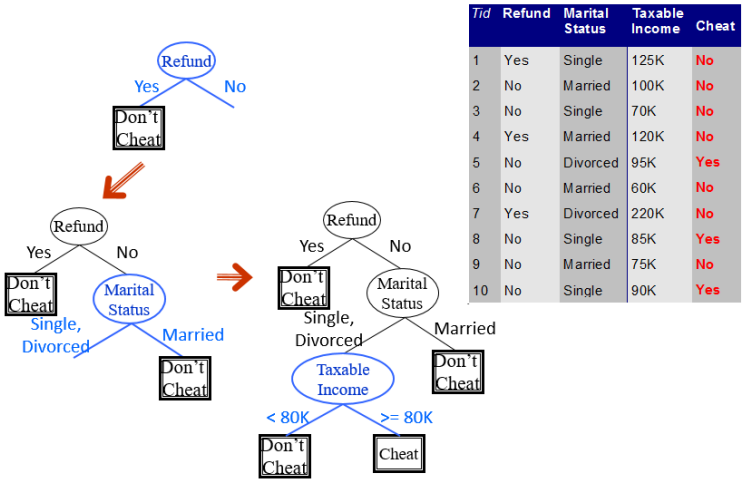


Fig.3 A Decision Tree structure

A decision tree and the closely related diagram in the process of decision analysis serves as a visual and analytical decision support tool, where the expected values of alternatives are calculated.

A decision tree consists of 3 types of nodes:

1. Decision nodes - commonly represented by squares
2. Chance nodes - represented by circles
3. End nodes - represented by triangles



Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Fig.4 A Decision Tree example

Advantages

- Decision trees are simple to understand and interpret.
- Important insights can be generated about the data with just a little explanation.
- Help in determining best, worst and expected values for different scenarios.
- Allows the addition of other possible scenarios.

Disadvantages

- Calculations can get very complex if many values are uncertain and if many outcomes are linked.
- Information gain in decision trees is biased towards attributes with more levels for a data containing categorical values with different number of levels.

Our Implementation:

1. We have implemented our code in Python.
2. Our code is divided into many modules:
   a. Class TreeNode
      We have created a class to represent the tree data structure. It has 3 functions:
   b. init()
      This is the constructor.
   c. setLeaf()
      Used to mark a node as leaf node and set the truth value
   d. setChild()
      Use to mark a node as parent and store its child pointers
   e. create_branches_continous()
      Divides the dataset on the basis of '<=' operator
   f. create_branches_categorical()
      Divides the dataset on the basis of '==' operator
   g. calculate_gini()
      Returns the gini index for the training labels at each node
   h. build_tree()
      Builds a tree for a given training dataset
   i. print_tree()
      Prints the tree built using the previous function
   j. find_label()
      Finds the label for each test data point by traversing the tree
   k. calculate_measure()
      Calculates various measures like accuracy, precision, recall and f-measure
   l. Main Code
      We have shuffled the data and applied k-fold to get k sets of testing and training data. We then called the corresponding functions and then displayed the measures.

Choice Description:

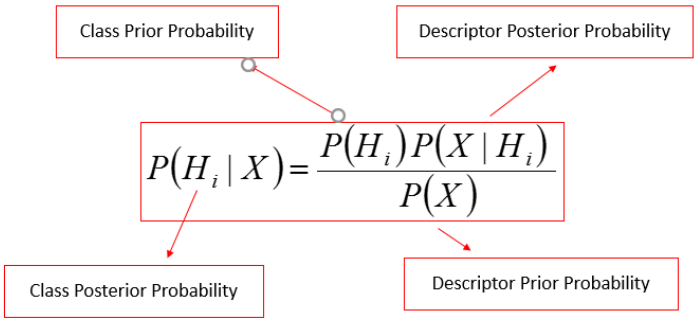
1. Categorical Data: We have used binary split for categorical data. So the left tree will contain data that has value1 in the column and the remaining data that has some other data in that column are sent to the right tree.
2. Continuous data: We have used '<=' for continuous data. Data having value less than equal to value1 are sent to the left tree while the remaining data is sent to the right tree.
3. Best feature: Best feature is the one which minimizes Gini Index the most.
4. Post-processing: We have performed pruning on our tree. Whenever the left and right node have the same truth value, we have eliminated the children and passed the truth value to the parent.

Naïve Bayes Classification

Naïve Bayes classifiers, a family of simple probabilistic classifiers, are based on applying Bayes' theorem with strong (naive) independence assumptions between the features. Unlike the decision trees that predict class labels, Bayesian classifiers are statistical classifiers that predict class membership probabilities.

Bayes Theorem

Bayes' theorem also known as Bayes' Law or Bayes' rule describes the probability of an event, based on prior knowledge of conditions that might be related to the event.



Advantages

- Naïve Bayes is fast to train and fast to classify.
- It is not sensitive to irrelevant features
- It is capable of handling real, discrete and streaming data well.

Disadvantages

- The algorithm assumes independence of features
- Another issue is that if you have no occurrences of a class label and certain attribute value together then the frequency based probability estimate will be zero.

In a nutshell:

- Naïve Bayesian classifier is a simple classifier that assumes attribute independence.
- Naïve Bayesian classifier is efficient when applied to large datasets.
- In terms of performance it is comparable to decision trees.

Our Implementation:

1. We have implemented our code in Python.
2. Our code is divided into many functions:
   a. calculate_probability()
      Calculates the probability of the test data points. For categorical data, we have calculated probability based on the counts. For continuous data, we have fitted the training data to a normal distribution and then calculated the pdf for the test data points.
   b. find_label()

Assigns the label having maximum probability to each test data point

c. calculate_measure()
Calculates various measures like accuracy, precision, recall and f-measure

d. Main Code
We have shuffled the data and applied k-fold to get k sets of testing and training data. We then called the corresponding functions and then displayed the measures.

Choice Description:

1. Continuous data: For continuous data, we have fitted the training data to a normal distribution and then calculated the pdf for the test data points.
2. Zero Probability: Laplacian correction is used to correct the zero probability problem. We simply add 1 to each case so that none of the probabilities becomes 0.

Random Forest

Random Forest, a type of ensemble learning method, where a group of weak models combine to form a powerful model, is capable of performing regression as well as classification tasks. This classification algorithm takes into consideration the dimensional reduction methods, outlier values and treats missing values and yet does a fairly good job.

Advantages

- The algorithm does a fairly good job in solving both classification and regression tasks.
- The algorithm is capable of handling large data set with high dimensionality i.e. it can handle thousands of input variables and yet identify most significant variables and thus is considered as one of the dimensionality reduction methods.
- The algorithm has an efficient method for estimating missing data and even maintains accuracy when the large data is missing.
- It has methods of balancing errors in the dataset where classes are imbalanced.
- Random Forest involves sampling the input data with replacement called bootstrap sampling. About one third of the input data can be utilized as test data, referred to as 'out-of-bag' samples. The error obtained on out-of-bag samples suggests that out-of-bag estimate is as accurate as using a test set of the same size as the training set. Therefore, using the out-of-bag error estimate removes the need for a set aside test set.

Disadvantages

- The algorithm doesn't give precise nature predictions for regression problems i.e. it doesn't predict beyond the range of the training data and that may cause over-fitting problem in case of noisy data.
- Random forest at times may feel like black-box approach as you have little control over the model, at best all you can do is just use different parameters and seeds.

Our Implementation:

1. We have implemented our code in Python.
2. Our code is divided into many modules:
   a. Class TreeNode
   We have created a class to represent the tree data structure. It has 3 functions:

b. init()
   This is the constructor.
c. setLeaf()
   Used to mark a node as leaf node and set the truth value
d. setChild()
   Use to mark a node as parent and store its child pointers
e. create_branches_continous()
   Divides the dataset on the basis of '<=' operator
f. create_branches_categorical()
   Divides the dataset on the basis of '==' operator
g. calculate_gini()
   Returns the gini index for the training labels at each node
h. build_tree()
   Builds a tree for a given training dataset
i. print_tree()
   Prints the tree built using the previous function
j. find_label()
   Finds the label for each test data point by traversing the tree
k. calculate_measure()
   Calculates various measures like accuracy, precision, recall and f-measure
l. Main Code
   We have shuffled the data and applied k-fold to get k sets of testing and training data. For each set of testing and training data, we created n trees. Different trees were possible as we changed the training set using bagging(random selection with replacement) and by selecting 20% of the columns randomly. We then called the corresponding functions and then displayed the measures.

Boosting

The term boosting belongs to a family of algorithms that involves converting weak learners to strong learners. In other words, boosting combines a base learner to form a strong rule.

The identification of weak rule involves applying the base (ML) learning algorithms with a different distribution. It's an iterative process which results in a new weak prediction rule every time base learning algorithm is applied. After sufficient number of iterations, the algorithm converges and combines these weak rules into a strong prediction rule.

Advantages:

1. Can use several weak classifiers to create a powerful classifier
2. Gives better accuracy than if an individual classifier was used.
3. An individual classifier is much easier to show/represent/explain than a Boosting model which has several classifiers

Disadvantages:

1. It is computationally expensive as compared to any of the classifiers used to build it
2. It can overfit

Our Implementation:

1. We have implemented our code in Python.
2. Our code is divided into many modules:
   a. Class TreeNode
      We have created a class to represent the tree data structure. It has 3 functions:
   b. init()
      This is the constructor.
   c. setLeaf()
      Used to mark a node as leaf node and set the truth value
   d. setChild()
      Use to mark a node as parent and store its child pointers
   e. create_branches_continous()
      Divides the dataset on the basis of '<=' operator
   f. create_branches_categorical()
      Divides the dataset on the basis of '==' operator
   g. calculate_gini()
      Returns the gini index for the training labels at each node
   h. build_tree()
      Builds a tree for a given training dataset
   i. print_tree()
      Prints the tree built using the previous function
   j. find_label()
      Finds the label for each test data point by traversing the tree
   k. calculate_measure()
      Calculates various measures like accuracy, precision, recall and f-measure
   l. Main Code
      We have shuffled the data and applied k-fold to get k sets of testing and training data. We assigned equal probability to each of the training data points and then divide it into train and validation. We then test the validation set and increase the probability of the records which were incorrectly labelled. Then we sample the training data into train and validation and continue this process for n trees. We then create a combined model by assigning weight to each models output based on the accuracy.

Result Analysis:

K-NN

Dataset	Accuracy	Precision	Recall	F-measure
1	0.964285714286	0.357000728133	0.920316350765	0.513168796598
2	0.673913043478	0.204972120738	0.384091970121	0.263437228234

Naïve-Bayes

Dataset	Accuracy	Precision	Recall	F-measure
1	0.933928571429	0.360723898537	0.898456216298	0.513907109003

2	0.695652173913	0.343326273488	0.689094053951	0.456627771924

Decision Tree

Dataset	Accuracy	Precision	Recall	F-measure
1	0.941071428571	0.354769158826	0.892099542334	0.503383664353
2	0.65652173913	0.234578979084	0.463068467574	0.30533252214

Random Forest

Dataset	Accuracy	Precision	Recall	F-measure
1	0.946428571429	0.344427187941	0.878813331217	0.492224390354
2	0.663043478261	0.0556681206752	0.107229225023	0.0726657649576

Boosting

Dataset	Accuracy	Precision	Recall	F-measure
1	0.951785714286	0.360991155695	0.921066586922	0.517394113371
2	0.639130434783	0.190652513556	0.35001616031	0.244788715554

From the results thus obtained, we can see that K-NN and Boosting are the best in terms of Accuracy, Precision, Recall and F-measure. However, Boosting is computationally complex as it creates several models and then uses them to predict. Thus, we should prefer KNN over Boosting if both give comparable results.

Choosing a classifier depends on the data that you are working with. One must start with the least computationally expensive classifier and progressively move towards stronger classifiers unless you know which classifier will give the best result.

Cross- validation

In order to assess how the statistical analysis results will generalize into an independent data set, a rotation estimation technique called Cross-validation is used. The primary use of this technique is in prediction and to estimate how accurately a predictive model will perform in practice.

K-fold cross validation

The K-fold cross-validation technique involves randomly partitioning the original sample into k equal sized subsamples. Now, out of the K subsamples thus obtained, a single subsample is retained as the validation data for testing the model and the remaining k-1 subsamples are utilized as training data. This cross-validation process is then repeated with each of the K subsample being used only once as the validation data and the k results thus obtained from the folds can be averaged to produce a single estimation.

An advantage of such an iterative cross-validation technique is that it allows every observation to be used both as training and validation data and each observation is used only once as validation data.

References

- https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
- https://en.wikipedia.org/wiki/Decision_tree
- https://en.wikipedia.org/wiki/Naive_Bayes_classifier
- https://en.wikipedia.org/wiki/Random_forest
- http://stats.stackexchange.com/questions/18891/bagging-boosting-and-stacking-in-machine-learning
- https://www.quora.com/Whats-the-difference-between-boosting-and-bagging
- https://en.wikipedia.org/wiki/Cross-validation_(statistics)