Comparison Study Naïve Bayes v/s K-NN

Aishwarya Vaidyanathan
University of Texas at Arlington
aishwarya.vaidyanathan@mavs.uta.edu

Jubin Sanghvi
University of Texas at Arlington
jubin.sanghvi@mavs.uta.edu

Abstract

Data Categorization (DC), also known as Data Classification, is the task of automatically classifying a set of data into different categories from a predefined set. On the other hand, Data Regression is the task of estimating relationship among the variables. If a data belongs to exactly one of the categories, it is a single-label classification task; otherwise, it is a multi-label classification task. Data analysis uses several tools from Information Retrieval (IR) and Machine Learning (ML). Text data analysis has received much attention in the last few years from both researchers in the academia and industry developers, as opposed to numerical data. In this paper, we focus mainly on numerical data for classification and regression with respect to speed and accuracy. We will focus on using two of the most popular machine learning algorithms, KNN and Naïve Bayes. The results of the algorithms are compared and analyzed for their performances.

Introduction

Data analysis is the process of obtaining usable and useful information. The analysis, irrespective of whether the data is qualitative or quantitative, may describe and summarize the data, identify relationships between variables, compare variables, identify the difference between variables, and forecast outcomes. The analysis of data includes, but not limited to, classification and regression. The task of automatically classifying a set of data into different categories from a predefined set is commonly referred as data classification, while the statistical task of estimating the relationships among variables is commonly referred as data regression. The existence of large amounts of data has urged scientists and large firms to develop models that analyze this data and get useful information from them to maintain pace with the ever-growing markets. Most of the researches and studies are focused on text data, however the data in real world also comprises of several numerical data that needs to be interpreted. In our project, we have thus focused on classification and regression of numerical data using the two popular machine learning algorithms, KNN and Naive Bayes and derived conclusions from the tests performed.

Methods & Data

This paper concerns methods for the classification and regression of high-dimensional numerical data, i.e. methods that, given a set of numerical training data with known categories or values and a set of test data will predict the unknown categories or values. It will also provide the accuracies and error rates for the two classification and regression methods being used.

Naïve Bayes

The Naive Bayesian classifier, assumes that numeric attributes are generated by a single Gaussian distribution. Although a Gaussian may provide a reasonable approximation to many real-world distributions, it is certainly not always the best approximation (John & Langley, 1995). The Naive Bayesian classifier provides a simple approach, with clear semantics, to representing, using, and learning probabilistic knowledge. The method is designed for use in supervised induction tasks, in which the performance goal is to accurately predict the class of test instances and in which the training instances include class information. One can view such a classifier, as a specialized form of Bayesian network, termed naive because it relies on two important simplifying assumptions. It assumes that the predictive attributes are conditionally independent given the class, and it posits that no hidden or latent attributes influence the prediction process. These assumptions support very efficient algorithms for both classification and learning (John & Langley, 1995). Let C be the random variable denoting the class of an instance and let X be a vector of random variables denoting the observed attribute values. Further, let c

represent a class label, and let x represent an observed attribute value vector. Given a test case x to classify, one simply uses Bayes' rule to compute the probability of each class given the vector of observed values for the predictive attributes, and then predicts the most probable class.

$$p(C = c|X = x) = \ p(C = c)\frac{p(X = x|C = c)}{p(X = x)}$$

Naïve Bayes treats discrete and numeric attributes somewhat differently. For each discrete attribute, p (X = x/C = c) is modeled by a single real number between 0 and 1 which represents the probability that the attribute X will take on the value x when the class is c. In contrast, each numeric attribute is modeled by some continuous probability distribution over the range of that attribute's values. A common assumption, not intrinsic to the Naïve Bayesian approach but often made nevertheless, is that, within each class, the values of numeric attributes are normally distributed. One can represent such a distribution in terms of its mean and standard deviation, and one can efficiently compute the probability of an observed value from such estimates. For continuous attributes, we can write,

$$p(X = x|C = c) = g(x; \mu_c; \sigma_c), \text{ where}$$

$$g(x; \mu; \sigma) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

the probability density function for a normal (or Gaussian) distribution (John & Langley, 1995).

Naive Bayes is also being used directly to regression problems, without discretizing the target value. For instance, say we are predicting a numeric target value Y, given an example E. E consists of m attributes X_1, X_2, . . ., X_m. Each attribute is either numeric, in which case it is treated as a real number, or nominal, in which case it is a set of unordered values. If the probability density function P(Y|E) of the target value were known for all possible examples E, we could choose Y to minimize the expected prediction error. However, P(Y|E) is usually not known, and has to be estimated from the data. Naive Bayes achieves this by applying Bayes' theorem and assuming independence of the attributes X_1, X_2, . . ., X_m given the target value Y. Bayes' theorem states that,

$$P(Y|E) = \frac{P(E,Y)}{\int P(E,Y)dY} = \frac{P(E|Y)P(Y)}{\int P(E|Y)P(Y)dY}$$

where the likelihood P(E|Y) is the probability density function (pdf) of the example E for a given target value Y , and the prior P(Y) is the pdf of the target value before any examples have been seen. Naive Bayes makes the key assumption that the attributes are independent given the target value, and hence the equation can be written (Kelly & Stutz, 1998).

k-Nearest Neighbors

In pattern recognition, the k-Nearest Neighbors algorithm (or k-NN) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression. In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor. In k-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors. 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. Both for classification and regression, it can be useful to assign weight to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of 1/d, where d is the distance to the neighbor. The neighbors are taken from a set of objects for which the class (for k-NN classification) or the object property value (for k-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required (Bijalwan, Kumari, Pascual & Semwa, 2014). A shortcoming of the k-NN algorithm is that it is sensitive to the local structure of the data.

In k-NN classification, the training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples. In the classification phase, k is a user-defined constant, and an unlabeled vector is classified by assigning the label which is most frequent among the k training samples nearest to that point. The distance metric used for continuous variables is Euclidean distance.

In k-NN regression, the k-NN algorithm is used for estimating continuous variables. One such algorithm

uses a weighted average of the k nearest neighbors, weighted by the inverse of their distance. This algorithm first computes the Euclidean distance from the point example to the labeled examples, then orders the labeled examples by increasing distance, finds a heuristically optimal number k of nearest neighbors, based on RMSE, which is done using cross validation. Finally, an inverse distance weighted average is calculated with the k-nearest multivariate neighbors (Vidya & Aghilla).

Implementation

As a part of pre-processing the all data files were converted into CSVs and the title rows were removed. Normalization was used to get the best out of the data and both algorithms. 10-fold cross validation was used on all the datasets for both the algorithms and both the machine learning tasks. For KNN, a validation set was used to estimate and use the best possible value for each dataset (Holmes & Witten, 2009). The error measure used to compare the algorithms for regression was Root Mean Square Error which is given by:

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n}(\hat{y} - y_t)^2}{n}}$$

All the tests were performed on an Elastic Compute server using the Amazon Web Services Cloud provider. The tests were performed on a raw server with no optimizations to understand the results of these algorithms without any external factors affecting them.

Data

The datasets used for the project were obtained from the University of California Irvine Machine Learning Repository. The chosen datasets were not confined to one particular domain or type. The datasets were chosen to be as general as possible so that they do not favor one algorithm over the other and that we get a broader picture of the comparison. Each dataset had between 10 to 100 attributes and between 100 to 1000 instances.

Results

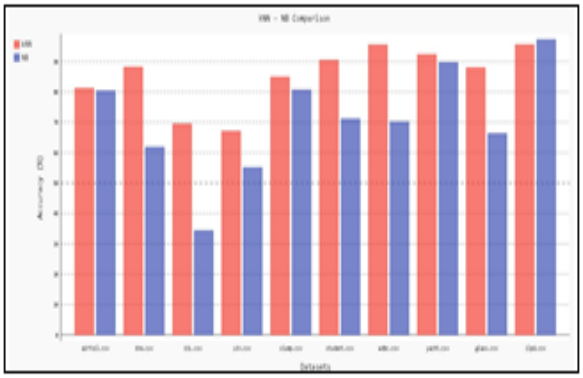
Below are the results and graphs for classification and regression tasks achieved by KNN and Naïve Bayes. We can clearly see that in most of the cases KNN is better than Naïve Bayes. But for classification datasets like ecoli.csv, popfailure.csv and wine.csv, the performance of Naïve Bayes is comparable to that of KNN. And for regression, datasets like ERA.csv, ESL.csv, student.csv and ilpd.csv, the difference between RMS errors of both these algorithms is quite less.

Classification Datasets	Accuracies (%)		Time (seconds)	
	KNN	NB	KNN	NB
ecoli	81.2	80.39	0.65	0.07
fertility	88.18	61.82	0.06	0.01
glass	69.49	34.32	0.3	0.05
ilpd	67.06	55.09	2.23	0.05
ionosphere	84.91	80.65	2.38	0.1
libras	90.48	71.11	3.91	1.69
parkinsons	95.5	70.17	0.49	0.03
popfailure	92.32	89.66	3.51	0.09
sonar	87.99	66.32	1.42	0.11
wine	95.56	97.22	0.28	0.03

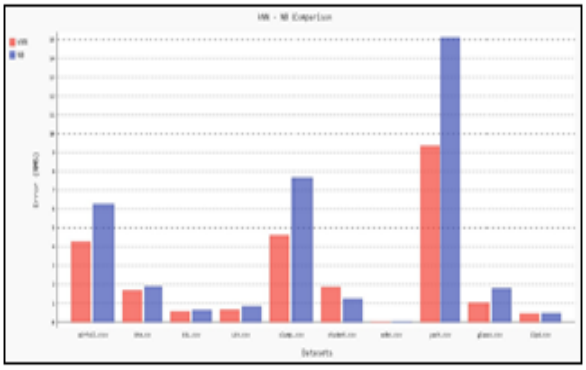
Classification Results for Time & Accuracy

Regression Datasets	Error (RMS)		Time (seconds)	
	KNN	NB	KNN	NB
airfoil	4.26	6.26	12.91	6.09
ERA	1.68	1.88	4.25	0.17
ESL	0.55	0.63	0.91	0.08
LEV	0.65	0.83	3.99	0.1
slump	4.6	7.66	0.08	0.07
student	1.85	1.23	1.16	0.33
wdbc	0.01	0.02	5.58	3.69
yacht	9.35	17.1	0.49	0.21
glass	1.02	1.79	0.29	0.05
ilpd	0.44	0.46	2.25	0.05

Regression Results for Time & Error Rate

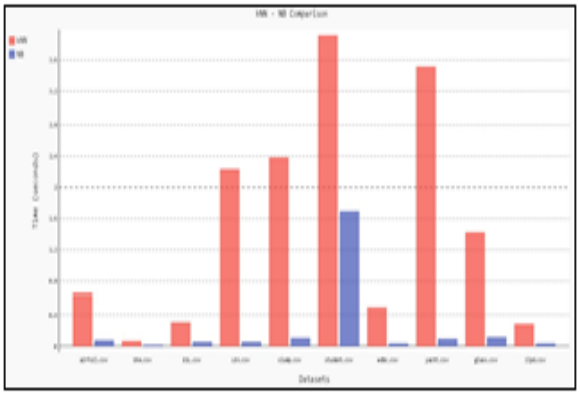


Classification Results Graph

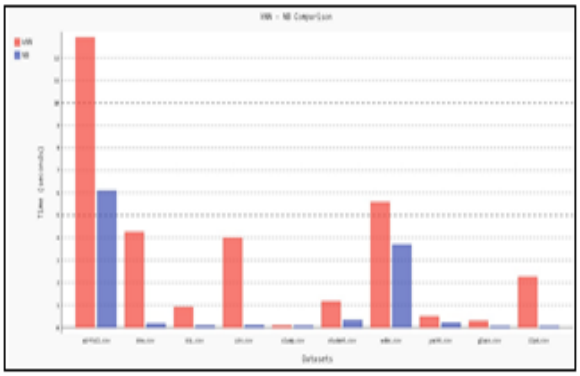


Regression Results Graph

But when we compare the classification and prediction time of both these algorithms we see a massive difference in favor of Naïve Bayes algorithm. It beats KNN in every dataset used for testing.



Classification Times Graph



Regression Times Graph

Conclusion

We conclude that KNN usually shows better accuracy as compared to Naïve Bayes. But datasets where attributes are close to Gaussian distribution we see Naïve Bayes performing on par if not better than KNN. Also, Naïve Bayes will always be faster in classification and prediction given that it stores information about the data and then uses it for prediction as opposed to KNN which calculates distances for every test instance. Thus, in applications where speed is of the utmost importance and where attributes are close to Gaussian distribution, we can use Naïve Bayes for classification or regression and get good results very quickly as compared to KNN.

References

[1]. George H. John, Pat Langley (1995): Estimating Continuous Distributions in Bayesian Classifiers.

[2]. Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W & Freeman, D. (1988), Autoclass: A Bayesian classification system, in "Machine Learning: Proceedings of the Fifth International Workshop", Morgan Kaufmann, pp. 54-64.

[3]. Vishwanath Bijalwan, Pinki Kumari, Jordan Pascual and Vijay Bhaskar Semwal (2014): Machine learning approach for text and document mining.

[4]. Murphy, P. M. & Aha, D. W. (1994), "UCI repository of machine learning databases", Available by anonymous ftp to ics.uci.edu in the pub/machine-learning-databases directory.

[5]. L. Wang, X. Zhao, “Improved knn Classification Algortihm Research in Text Categorization”, In the Proceedings of the 2nd International Conference on Communications and Networks (CECNet), pp. 1848-1852.

[6]. K. A. Vidhya and G. Aghila, “A Survey of Naïve Bayes Machine Learning approach in Text Document Classification”, International Journal of Computer Science and Information Security, vol. 7, no. 2, pp. 206-211.

[7]. Friedman, N., Geiger, D. & Goldszmidt, M. (1997). Bayesian network classifiers. Machine Learning, 29(2/3), 131–163.

[8]. Holte, R. C. (1993). Very simple classification rules perform well on most commonly used datasets. Machine Learning, 11, 63–91.

[9]. Eibe Frank, Leonard Trigg, Geoffrey Holmes, Ian H. Witten: Naive Bayes for Regression, Machine Learning, 000, 1–20.

[10]. Siddharth Deokar: Weighted K Nearest Neighbor, Machine Learning.

[11]. Yun-lei Cai, Duo Ji Dong-feng Cai: A KNN Research Paper Classification Method Based on Shared Nearest Neighbor, Proceedings of NTCIR-8 Workshop Meeting.