Generative Adversarial Networks

Jacob Smith

December 12, 2017

1 Introduction

This report details the experimental study conducted for the CS6735 programming project. Six machine learning algorithms were implemented, including both regular and ensemble learning algorithms. Each was evaluated on five datasets from the UCI Machine Learning data repository using 10 times 5-fold cross validation. The technical details of the implementation will be described in this report and the results will be presented and analyzed.

2 Objectives

1. Implement the following machine learning algorithms:
   (a) ID3
   (b) Adaboost on ID3
   (c) Random Forest
   (d) Naïve Bayes
   (e) Adaboost on Naïve Bayes
   (f) K-nearest neighbors
2. Evaluate the algorithms on the following datasets:
   (a) Brest Cancer Data
   (b) Car Data
   (c) Ecoli Data
   (d) Mushroom Data

3. Create report of the experimental study including:
   (a) Description of the learning algorithms you implement.
   (b) Description of the datasets you use (number of examples, number of attribute, number of classes, type of attributes, etc.).
   (c) Technical details of your implementation: pre-processing of data sets (discretization, etc.), parameter setting, etc.
   (d) Design of your programming implementation (data structures, overall program structure).
   (e) Report and analysis of your experimental results using using 10 times 5-fold cross-validation (include mean, standard deviation of accuracies).
   (f) Comparison and discussion of the algorithms with respect to the experimental results.

3 Algorithms

For this study, six different algorithms; however, only the five unique algorithms are described in this section. Both the ID3 decision tree and Naïve Bayes algorithms were applied to the adaboost ensemble algorithm as weak learners.

3.1 ID3

The Iterative Dichotomiser 3 (ID3) algorithm is a decision tree algorithm used to efficiently develop decision trees [3]. This algorithm was developed for situations where there are large amount of attributes and training sets which contain many attributes, but there was the need to be able to efficiently develop relatively well performing decision trees.

The algorithm as described by Mitchell [2] was implmented for this assignment. The operations begin at the root node with a set of catigorical, labled data S. The attribute A which introduces the largest information gain is chosen to split on. This is an exhaustive process which checks all remaining attributes. This process imediatly stops if there are no possible subsets (identical samples have same class label) or the split is not statistically significant (positive information gain). If these stopping conditions are not met, the data S is split using attribute A into subsets $S_1, ..., S_n$ where n is the amount of different categories of attribute A. A new node is then created for each subset. If the entropy of a new node is 0 or the sample count is 1, then the node is turned into a leaf and the splitting operation

ends. Furthermore, two additional early stopping conditions are implemented. Max tree level and minimum number of samples attributes are initially set at the start of training.

After the initial ID3 algorithm was implemented, furthure enhancements were made to allow for continuous data to be used. This enhancement was taken from the C4.5 as developed by Ross Quinlan [1]. Given a continuous attribute A, we temporarily sort the data S. After the data is sorted, we iteratively determine a threshhold to split the data as the maximize the information gain. To test potential threshholds, we find two sequential numbers which are not of the same category and split at the midway point of the two values. This enhancements saves an a large amount of time for each continuous attribute found within the datasets as descritization does not need to occur. Furthermore, this algorithm chooses the optimal threshhold to split the data which improves classification accuracy.

3.2 Naïve Bayes

The Naïve Bayes algorithm is implemented as described in [2]. This algorithm is named due its assumption of conditional independence amongst each attribute:

$$P(a_1, ..., a_n|v_j) = \prod_{i=1}^{n} P(a_i|v_j)$$

and its application of Bayes thereon:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Naïve Bayes is a practical learning algorithm, receives relatively good results given its preference bias and is easy to implement. This algorithm naturally handles both categorical and continuous data (given a prior is assumed). For each continuous attribute, a Gaussian distribution was assumed. The final algorithm implemented was:

$$\hat{y} = \text{argmax}_{k \in \{1,..K\}} P(v_k) \prod_{i=1}^{m} p\left(a_i|v_k\right) \prod_{j=m}^{n} P\left(a_j|v_k\right)$$

where P is the probability, p is the gaussian probability density function and K is the number of distinct targets. In this equation, the first m attributes are continuous whereas the later are catigorical. During training, the data S is split by lables into subsets $S_1, ..., S_n$ where n is the number of different classes. For each subset, we estimate the class probability $P(v_k)$. Subsequently, for each attribute value a_i of each attribute a, we estimate $P(a_i|v_k)$ or p. We are not able to immediately calculate $p(a_i|v_k)$ as attribute a is a continous attribute.

3.3 Adaboost

Adaboost is a boosting algorithm that was first introduced in 1995 by Freunb Schapire. This type of ensemble learning uses a set of weak learners to reduce bias and become a strong learner. It solves many of the difficulties associated with previous boosting algorithms [4].

The primary part of the algorithm is to maintain a weight value w for every sample within the data S. This value increases when the sample is misclassified and decreases when the sample is correctly classified. The target values are assumed be to in the set $+1, -1$. The Adaboost algorithm is easily implemented on weak learners which are parameterized by sample weights. Neither Naïve Bayes or ID3 natively implement weighted training. Therefore, an alternative implementation was used where a subset of the training data is selected with replacement based on the weights. For example, if the weight vector for two samples is $1, 2$, the second sample is twice as likely to be picked each time.

Initially, the weights are set to $\frac{1}{|S|}$. For a predefined number of iterations, a weak learner is trained on the a subset of the traning data. Predictions are made for each training sample and the error is calculated. Using the accuracy, a value α is calculated which represents the accuracy of the weak learner. This α value is used to update the weights and the weak learner, α pair is added to the list of previous weak learners. To make classification, there is a weighted vote between using the weak learners and their associated α values and the sign is used to make the final classification.

To implement multiclass adaboost, K adaboost models are trained where K is the total number of distant classes. For each adaboost model, one target is selected as the $+1$ adaboost target and the other become the -1 target. The target value associated with the most positive prediction becomes the predicted class.

3.3.1 Adaboost on Naïve Bayes

Naïve Bayes is an inherently weak learner due to its assumption of conditional independence amongst attributes. This fact introduces a large amount of preference bias which makes a Naïve Bayes and ideal candidate for the Adaboost algorithm.

3.3.2 Adaboost on ID3

The ID3 algorithm does not introduce a large amount of preference bias and often overfitting if allowed to train to completion. Therefore, to implement Adaboost on ID3, the depth of the trees were limited to one level. This type of tree is known as a *decision stump*.

3.4 Random Forest

Random forest algorithm is a bagging algorithm which uses an ensemble of strong learners to reduce overfitting to the training set. This method contains many similarities to the adaboost algorithm as described in section 3.3; however, no weights are used for random forests. For a predefined number of iterations, a subset of the of the training data is selected with replacement. The new training set is then used to train a decision tree to completion and added to tree is added to a list. To make a classification, each tree in the forest votes and the most common target value becomes the predicted class. As an extension of the base random forest algorithm, weights may be given to each decision tree based on their classification error during training; however, this extension was not added for this implementation.

3.5 K-Nearest Neighbors

The K-Nearest Neighbor algorithm is an instance based learning algorithm which makes predictions by finding the k nearest neighbors of a sample x and using the most common class as the predicted class for that sample. This algorithm can be used with both categorical and continuous data. For this implementation, Euclidian distance was used for continuous variables and Hamming distance was used for categorical data.

4 Data

Each algorithm was implemented on five different datasets from the UCI Machine Learning Repository.

4.1 Breast Cancer Data

Title: Wisconsin Diagnostic Breast Cancer

Date: November 1995

Number of Samples: 569

Number of Attributes: 10

Attribute Information:

Description	Values	Type
Sample Code Number	id number	Discrete
Clump Thickness	1 - 10	Discrete
Uniformity of Cell Size	1 - 10	Discrete
Uniformity of Cell Shape	1 - 10	Discrete
Marginal Adhesion	1 - 10	Discrete
Single Epithelial Cell Size	1 - 10	Discrete
Bare Nuclei	1 - 10	Discrete
Bland Chromatin	1 - 10	Discrete
Normal Nucleoli	1 - 10	Discrete
Mitoses	1 - 10	Discrete

Class Target: Neoplasm

Class Information:

Description	Value	Count	Percentage
Benign	2	357	65.5%
Malignant	4	212	34.5%

Missing Attributes: 16

4.2 Car Data

Title: Car Evaluation Database

Date: June 1997

Number of Samples: 1728

Number of Attributes: 6

Attribute Information:

Description	Values	Type
buying	v-high, high, med, low	Discrete
maint	v-high, high, med, low	Discrete
doors	2, 3, 4, 5-more	Discrete
persons	2, 4, more	Discrete
lug_boot	small, med, big	Discrete
safety	low, med, high	Discrete

Class Target: Car Quality

Class Information:

Description	Value	Count	Percentage
Unacceptable	unacc	1210	70.023%
Acceptable	acc	384	22.222%
Good	good	69	3.993%
Very Good	v-good	65	3.762%

Missing Attributes: none

4.3 E. Coli Data

Title: Protein Localization Sites

Date: September 1997

Number of Samples: 336

Number of Attributes: 8

Attribute Information:

Description	Values	Type
Sequence Name	Accession Number	Discrete
mcg	0.00 - 0.89	Continuous
gvh	0.16 - 1.00	Continuous
lip	0.48 - 1.00	Discrete
chg	0.50 - 1.00	Discrete
aac	0.00 - 0.88	Continuous
alm1	0.03 - 1.00	Continuous
alm2	0.00 - 0.99	Continuous

Class Target: Localization Site

Class Information:

Description	Value	Count	Percentage
Cytoplasm	cp	143	42.56%
Inner Membrane	im	77	22.92%
Perisplasm	pp	52	15.48%
Inner Membrane Uncleavable	imU	35	10.42%
Outer Membrane	om	20	5.95%
Outer Membrane	omL	5	1.49%
Inner Membrane Lipoprotein	imL	2	0.59%
Inner Membrane Cleavable	imS	2	0.59%

Missing Attributes: none

4.4 Letter Recognition Data

Title: Letter Image Recognition Data

Date: January 1991

Number of Samples: 20000

Number of Attributes: 16

Attribute Information:

Description	Values	Type
x-box	0 - 9	Continuous
y-box	0 - 9	Continuous
width	0 - 9	Continuous
high	0 - 9	Continuous
onpix	0 - 9	Continuous
x-bar	0 - 9	Continuous
y-bar	0 - 9	Continuous
x2bar	0 - 9	Continuous
y2bar	0 - 9	Continuous
xybar	0 - 9	Continuous
x2ybr	0 - 9	Continuous
xy2br	0 - 9	Continuous
x-ege	0 - 9	Continuous
xegvy	0 - 9	Continuous
y-ege	0 - 9	Continuous
yegvx	0 - 9	Continuous

Class Target: Letter

Class Information:

Description	Value	Count	Percentage
Letter A	A	789	0.04%
Letter B	B	766	0.04%
Letter C	C	736	0.04%
Letter D	D	805	0.04%
Letter E	E	768	0.04%
Letter F	F	775	0.04%
Letter G	G	773	0.04%
Letter H	H	734	0.04%
Letter I	I	755	0.04%
Letter J	J	747	0.04%
Letter K	K	739	0.04%
Letter L	L	761	0.04%
Letter M	M	792	0.04%
Letter N	N	783	0.04%
Letter O	O	753	0.04%
Letter P	P	803	0.04%
Letter Q	Q	783	0.04%
Letter R	R	758	0.04%
Letter S	S	748	0.04%
Letter T	T	796	0.04%
Letter U	U	813	0.04%
Letter V	V	764	0.04%
Letter W	W	752	0.04%
Letter X	X	787	0.04%
Letter Y	Y	786	0.04%
Letter Z	Z	734	0.04%

Missing Attributes: none

4.5 Mushroom Data

Title: Mushroom Database

Date: 27 April 1987

Number of Samples: 8124

Number of Attributes: 22

Attribute Information:

Description	Values	Type
cap-shape	b,c,x,f,k,s	Discrete
cap-surface	f,g,y,s	Discrete
cap-color	n,b,c,g,r,p,u,e,w,y	Discrete
bruises?	t,f	Discrete
odor	a,l,c,y,f,m,n,p,s	Discrete
gill-attachment	a,d,f,n	Discrete
gill-spacing	c,w,d	Discrete
gill-size	b,n	Discrete
gill-color	k,n,b,h,g,r,o,p,u,e,w,y	Discrete
stalk-shape	e,t	Discrete
stalk-root	b,c,u,e,z,r,?	Discrete
stalk-surface-above-ring	f,y,k,s	Discrete
stalk-surface-below-ring	f,y,k,s	Discrete
stalk-color-above-ring	n,b,c,g,o,p,e,w,y	Discrete
stalk-color-below-ring	n,b,c,g,o,p,e,w,y	Discrete
veil-type	p,u	Discrete
veil-color	n,o,w,y	Discrete
ring-number	n,o,t	Discrete
ring-type	c,e,f,l,n,p,s,z	Discrete
spore-print-color	k,n,b,h,r,o,u,w,y	Discrete
population	a,c,n,s,v,y	Discrete
habitat	g,l,m,p,u,w,d	Discrete

Class Target: Edibility

Class Information:

Description	Value	Count	Percentage
Edible	e	4208	51.80%
Poisonous	p	3916	48.20%

Missing Attributes: 2480 (all for attribute #11)

5 Technical Details

Missing attribute values within the breast cancer dataset were replaced with the mode of the samples of the same class. For the mushroom dataset, as the amount of missing values for attribute 11 was very large, the missing values themselves were simply treated as another category. No discretization occurred as all base algorithms are able to handle continuous data. For all datasets, any sort of identification number or code attributes were removed. Within the E. Coli dataset, the lip and chg categorical attributes were

also removed as they provided minimal information. Furthermore, any string values were converted to integers. For example, the "v-high", "high", "med" and "low" attribute values would have been converted to 0, 1, 2, 3 respectively.

5.1 ID3

Dataset	Max Level	Minimum # of Samples
Breast Cancer	None	SOMETHING
Car Cancer	None	SOMETHING
Letter Recognition	None	SOMETHING
E. Coli	None	SOMETHING
Mushroom	None	SOMETHING

5.2 Naïve Bayes

gaussian

5.3 Adaboost on Naïve Bayes

number of learners

5.4 Adaboost on ID3

number of learenrs

5.5 Random Forest

number of trees

5.6 K-Nearest Neighbors

k distance

6 Project Design

A Java machine learning framework was first created to ease the implementation of the six learning algorithms. This framework uses JUnit for unit tests and slfj4 to control logging. Both work independently of the machine learning algorithms. The package hiarchy was modeled off ...

Algorithm:

Dataset:

Model:

KFold: Implements the KFold cross validation algorithm.

Report: The object returned from a k-fold cross validation test....

Util: Utility functions used through the machine learning project.

6.1 dt

This package contains all of the objects associated with decision trees.

Children:

ID3:

ID3Model:

Node:

6.2 ensemble

This package contains all of the objects associated with enseble learners.

Adaboost:

AdaboostModel:

Multi-Adaboost:

Multi-AdaboostModel:

RandomForest:

RandomForstModel:

6.3 exceptions

This package contains the common exceptions which occur the learning process.

AttributeException:

DataException:

FileException:

PredictionException:

6.4 math

Data was particularly difficult to handle. Whereas Java is a statically typed language, difficulties arose in handling both floating point and integer values. To manage this, a math package was created which implements useful data structures such as vectors and matrices. These object were used to abstract away from the underlying data structures. Furthermore, the 'Vector' object provided many useful methods for vector arithmetic such as element-wise multiplication, division and exponents. These methods also support broadcasting, where a vector can be multiplied by a single number. These objects are used throughout the library in place of primitive arrays and ArrayLists and have were essential to the code quality. Often, methods it was necessary to have two associated objects held in one data structure. For example, there exists a method within the DataSet object which splits itself into two parts based on a pivot point. This method would preferably return a pair of datasets; however, many common data structures such as arrays, lists and maps seemed did not seem to adequate. To accomplish this task, a simple Tuple data structure was created. Several other useful objects were implemented within the math package such as distance function objects, distribution objects and an object containing useful static methods. The purpose of these objects will be elaborated within Naïve Bayes package design section. The utility object implemented useful functions not available within the Java math package.

MathException:

Matrix:

Tuple:

Util:

Vector:

6.4.1 distance

This package contains distance functions for the KNN classifier.

DistanceFunction:

Euclidian:

Hamming:

6.4.2 distribution

This package contains distribution implementations for the Naïve Bayes classifier.

Distribution:

GaussianDistribution:

6.5 nb

This package contains all of the objects associated with Naïve Bayes.

Attribute:

BayesException:

ClassSummary:

ContinuousAttribute:

DiscreteAttribute:

NaïveBayes:

NaïveBayesModel:

6.6 neighbors

This package contains all of the objects associated with k-nearest neighbors.

KNN:

KNNModel:

KNNException:

Breast Cancer	Car	Letter	E. Coli	Mushroom
Adaboost on ID3	100.00%	100.00%	100.00%	100.00%
ID3	100.00%	100.00%	100.00%	100.00%
Random Forest	100.00%	100.00%	100.00%	100.00%
Naïve Bayes	100.00%	100.00%	100.00%	100.00%
Adaboost on Naïve Bayes	100.00%	100.00%	100.00%	100.00%
K-Nearest Neighbor	100.00%	100.00%	100.00%	100.00%

Table 1: Results of the six implemented algorithms on five UCI Machine Learning repositories. Accuracy calculated using 10 times 5-fold cross validation.

7 Results

graph of hyperparameter search?

References

[1] Badr HSSINA, Abdelkarim MERBOUHA, Hanane EZZIKOURI, and Mohammed ERRITALI. A comparative study of decision tree id3 and c4.5. *International Journal of Advanced Computer Science and Applications(IJACSA)*, 2014.

[2] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[3] J. R. Quinlan. Induction of decision trees. *Mach. Learn.*, 1(1):81–106, March 1986.

[4] Robert E. Schapire. A brief introduction to boosting. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'99, pages 1401–1406, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.