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 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 create decision trees [3]. This algorithm was developed for situations where there are large amount of attributes and training sets which contain many attributes.

Initially, the algorithm as described by Mitchell [2] was implemented. The algorithm begins at the root node with a set of categorical, labeled data S. The attribute a which introduces the largest information gain is chosen and the node splits on that attribute. This is a recurssive process which continues until one of several terminating conditions are met. For example, this process immediately end if there are no possible subsets 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 multiclass classification problems, the SAMME AdaBoost algorithm [5] was implemented. This algorithm sees the addition of $log(|v| - 1)$, where $|v|$ is the number of classes, to the alpha value. Furthermore, the classification procedure calculates a map of the class and the total votes and returns the class with the highest vote.

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 may overfit if allowed to train to completion. Therefore, to implement AdaBoost on ID3, the depth of the trees was limited.

3.4 Random Forest

The 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 both the breast cancer dataset and mushroom dataset treated as another category. Attempts were initially made to replace the missing values; however, no improvements with accuracy were observed. Furthermore, no discretization occurred as all base algorithms are able to handle continuous data. For each dataset, 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

Table 1: The

Dataset	Minimum # of Samples
Breast Cancer	1
Car	1
E. Coli	2
Letter Recognition	1
Mushroom	1

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

5.2 Naïve Bayes

Dataset	Distribution
Breast Cancer	None
Car	None
E. Coli	Gaussian
Letter Recognition	Gaussian
Mushroom	None

5.3 AdaBoost on ID3

Dataset Proportion of Samples	Max Level	# of Learners
Breast Cancer 0.50	1	100
Car 0.70	1	100
E. Coli 0.50	1	100
Letter Recognition IDK	1	IDK
Mushroom 0.30	1	10

5.4 AdaBoost on Naïve Bayes

Dataset Proportion of Samples	**Distribution**	**# of Learners**
Breast Cancer 0.70	None	100
Car 0.40	None	100
E. Coli 0.30	Gaussian	150
Letter Recognition IDK	Gaussian	IDK
Mushroom 0.30	None	10

5.5 Random Forest

Dataset Proportion of Samples	**# of Learners**
Breast Cancer 0.70	120
Car 0.60	200
E. Coli 0.70	120
Letter Recognition 0.30	50
Mushroom 0.30	10

5.6 K-Nearest Neighbors

Dataset Distance	**K**
Breast Cancer Hamming	1
Car Hamming	1
E. Coli Euclidian	1
Letter Recognition Euclidian	1
Mushroom Hamming	1

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 and may be completely removed without affecting the functionality of the code.

6.1 jml

This is the root package of the project and contains several useful objects which are used through the various sub-packages.

Algorithm An interface that all algorithms implement. These objects fit a Model to a DataSet.

Dataset An object which abstracts away from the data and provides several useful behaviors such as splitting into subsets using both continuous or discrete attributes.

Model An abstract classes which every trained model implements. Objects which extend this class must implement the predict method.

KFold Implements the KFold cross validation algorithm. The object is initialized with set number of folds and is then able to generate reports given an algorithm and dataset.

Report An object which summarizes the results of a particular algorithm on a particular dataset. A report contains a list of accuracies and can calculate information such as the

mean accuracy of standard deviations.

Util This object contains several useful static methods. These include reading from CSV files, converting primitive arrays to Lists and calculating entropy amongst others.

6.2 tree

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

ID3 The implementation of the ID3 algorithm. Fits a DataSet to a tree of decision Nodes and creates a ID3Model. The algorithm is parameterized by the max level and minimum number of samples.

ID3Model A trained ID3 model. It uses the trained decision tree to make classification given a sample of new data.

Node A node within a decision tree. Contains the logic to choose the best continuous or discrete attribute to split on. Each node has an associated attribute

Children A Pluggable Object which both ContinuousChildren and DiscreteChildren implement. This allows for the abstraction of most of the discrete and continuous variable logic associated with splitting and classifying.

ContinuousChildren The children of a Node which split on a continuous attribute.

DiscreteChildren The children of a Node which split on a discrete attribute.

6.3 ensemble

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

AdaBoost The implementation of the SAMME AdaBoost algorithm. This algorithm is able to handle both binary and multiclass classification problems and is parameterized by the number of weak learners, proportion of samples and base learning algorithm.

AdaBoostModel A trained AdaBoost model. It classifies a new instance using a weighted vote mechanism and returns the most likely class.

RandomForest The implementation of the Random Forest algorithm. This algorithm is parameterized by the number of strong learners, proportion of samples and base learning algorithm.

RandomForestModel A trained Random Forest model. It classifies a new instances using a voting mechanism and returning the most common classification.

6.4 exceptions

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

AttributeException This exception is thrown when an error arises due to the attribute types (continuous, discrete).

DataException This exception is thrown when there is an error associated with the data.

FileException This exception is thrown when there is an issue with the data file, particularly while reading CSV files.

PredictionException This exception is thrown when an error occurs during classification.

6.5 math

This package contains objects associated with mathematics. For

MathException This exception occurs when an error occurs during a mathematical operation.

Matrix An object which representation of a matrix. It contains useful operations for manipulating matrices such as retrieving, modifying and adding rows and columns.

Tuple A tuple object which uses Generics to hold two objects. This object is useful for moving associated objects throughout the code.

Util A utility class that contains several useful mathematical operations such as logarithmic and exponential functions. For example, an *exp* function is implemented which is parameterized by a Vector and returns a Vector.

Vector is an object representation of a vector. Similar to the Matrix class, it contains several useful operation for manipulating its values. Furthermore, several vector arithmetic operations are implemented such as multiplication, addition and the dot product. The Vector class abstracts away from the underlying data representation.

6.5.1 distance

This package contains distance functions for the KNN classifier.

Distance is an interface which all distance functions must implement.

Euclidian An object which represents the Euclidian distance function $\sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$. This object can be used to calculate the Euclidian distance between two real valued Vectors while classifying an new instance in a KNN classifier.

Hamming An object which represents the Hamming distance function $\sum_{i=1}^{n} f_i(q, p)$ where $f_i(q, p) = 0$ if $q_i = p_i$ and 0 otherwise. This object can be used to calculate the Hamming distance between two discrete Vectors while classifying an new instance in a KNN classifier.

6.5.2 distribution

This package contains distribution implementations for the Naïve Bayes classifier.

Distribution is an interface which each distribution object must implement.

Gaussian An object which reprints a Gaussian distribution. It is used be the Naïve Bayes classifier for continuous attributes.

6.6 bayes

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

Attribute is an interface which the Continuous and Discrete attribute types must implement.

BayesException is an RuntimeException which is thrown when no Distribution is supplied and there are continuous attributes within the DataSet.

ClassSummary is an object created during the NaiveBayes learning process. It contains the class probability $P(v_k)$ and all of the necessary information to calculate either $p(a_i|v_k)$ or $P(a_j|v_k)$ given a new value.

Continuous represents a continuous attribute. It is parameterized by a mean, standard deviation and Distribution. Given a new value, the value of the probability density function of the given distribution is calculated.

Discrete represents a discrete attribute. It is parametrized by a map of conditional probabilities and the sum of the class length and probability of an unseen value. The parameters of the discrete attribute were carefully chosen for performance reasons.

NaïveBayes is the implementation of the Naïve Bayes algorithm. It is optionally parameterized by a Distribution. The algorithm create a ClassSummary for each distinct class.

Algorithm	Breast Cancer	Car	Letter	E. Coli	Mushroom
ID3	97.994%	90.634%	87.930%	79.678%	99.998%
Naïve Bayes	97.663%	81.940%	64.570%	80.556%	95.952%
AdaBoost on ID3	98.715%	88.822%	100000%	84.460%	99.446%
AdaBoost on Naïve Bayes	98.626%	91.367%	100000%	78.803%	99.744%
Random Forest	97.592%	91.175%	91.949%	84.021%	99.865%
K-Nearest Neighbor	98.050%	89.763%	93.923%	87.779%	99.826%

Table 2: Results of the six implemented algorithms on five UCI Machine Learning repositories. Accuracy calculated using 10 times 5-fold cross validation.

NaïveBayesModel is a trained Naïve Bayes model. It classifies new samples by calculating the probability of each class and returning the most likely classification.

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

6.7 neighbors

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

KNN is the implementation of the k-nearest neighbor algorithm. It is parameterized by the number of neighbors to check. The learning process simply consist of providing the DataSet and parameters to a KNNModel.

KNNModel is a trained KNN classifier. It classifies new samples by finding the k-closest neighbors and returning the most common classification.

KNNException is the exception thrown when k is greater than the data sample count.

7 Results

The ID3 algorithm generally outperformed the Naïve Bayes algorithm as seen in table 2. This is especially true when considering the letter recognition dataset. One possibility for this discrepancy is the type of distribution used for continuous attributes. No other probability density functions were experimented with. Furthermore, the letter recognition dataset likely expresses high correlation amongst the attributes. However, the two algorithms perform identically for the E Coli. and breast cancer datasets.

Algorithm	Breast Cancer	Car	Letter	E. Coli	Mushroom
ID3	1.465%	4.061%	3.392%	8.952%	0.017%
Naïve Bayes	1.678%	3.373%	1.357%	11.199%	2.128%
AdaBoost on ID3	1.216%	3.085%	10000%	8.246%	1.081%
AdaBoost on Naïve Bayes	1.394%	1.987%	10000%	8.802%	0.652%
Random Forest	1.597%	5.171%	2.622%	8.325%	0.702%
K-Nearest Neighbor	1.118%	3.967%	2.284%	8.089%	0.433%

Table 3: Results of the six implemented algorithms on five UCI Machine Learning repositories. Accuracy calculated using 10 times 5-fold cross validation.

The AdaBoost algorithm when using either ID3 or Naïve Bayes as a base model either improves or maintains the same accuracy as their base models. For example, AdaBoost on ID3 only improves the classification accuracy for E. Coli and breast cancer datasets. Additionally, AdaBoost on Naïve Bayes only improved the classification accuracy on the breast cancer and car datasets. The random forest algorithm also either improved or maintained the same accuracy as their base models. I believe the classification accuracy for each ensemble algorithm and dataset pair could be improved given more time for hyperparameter tuning. The larger datasets proved difficult for experimentation as they required much larger training times. In section 5, we see that both the mushroom and letter recognition datasets tend to have much lower learner counts than their counterparts. Optimization was considered during the algorithm implementations; however, training time was not drastically reduced.

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.

[5] Ji Zhu, Saharon Rosset, Hui Zou, and Trevor Hastie. Multi-class adaboost. 2006.