Classification of Pima Indians Diabetes Dataset using AdaBoost with MLE as Base Classifier

Sesha Phani Deepika Vadlamudi

8th May 2020

This Final Project report is submitted to the Stevens Institute of Technology towards partial fulfilment of the Machine Learning (CS559) course in MS in Computer Science degree program.

Appeal

Due to time constraint, many of the sentences in this report are reproduced as it is from the references given in this report. So, it may come under plagiarism. We request you to please be considerate and exempt us from any kind of punishment.

1. Why Machine Learning?

It is not possible to write a program exhausting all possible scenarios of a problem. For example, it is hard to compute the probability that a credit card transaction is fraudulent. This is largely because the relationship among multiple features may be quite complex or may not even be possible. Also, since fraud scenario changes at every instance, the program that estimates the probability of occurrence of a fraud needs to be adaptive and changing to be fairly accurate in its estimation. Machine Learning is an area that addresses this kind of problems. It allows programs to automatically learn and make accurate predictions based on past observations. That is, Machine learning is concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data [Hadi Hormozi, et al.]. So, a major focus of machine learning research is to automatically learn to recognize complex patterns and make intelligent decisions based on data; the difficulty lies in the fact that the set of all possible behaviors given all possible inputs is too large to be covered by the set of observed examples (training data). Hence the learner must generalize from the given examples, to be able to produce a useful output in new cases. So, the goal of various machine learning algorithms is to devise learning algorithms that learns automatically without any human intervention or assistance. The emphasis of machine learning is on automatic methods.

Machine Learning algorithms can broadly be classified into two categories: Supervised Learning and Unsupervised Learning.

2. Supervised Learning: An Overview

In Supervised learning, the machine is provided with a given set of inputs with their desired outputs. The machine needs to study those given sets of inputs and outputs and find a general function that maps inputs to desired outputs. That is these techniques attempt to find out the relationship between input attributes (independent variables) and a target attribute (dependent variable) [Aized Amin Soofi et al.]. In essence, a supervised learning algorithm supervises the training data and produces a general rule (function), which can be used for mapping new inputs. So, in supervised machine learning, we search for algorithms that reason from externally supplied instances to produce general hypotheses. These hypotheses then make predictions about future instances. In other words, the goal of supervised learning is to build a concise model of the distribution of class labels in terms of predictor features. The resulting classifier is then used to assign class labels to the testing instances where the values of the predictor features are known, but the value of the class label is unknown.

Supervised learning techniques in Machine Learning can broadly be classified into Regression and Classification. In regression output variable takes continuous values while in classification output variable takes class labels [Aized Amin Soof et al.]. Classification categorizes our data into a desired and distinct number of classes where we can assign a label to each class. It is used to predict group membership for data instances. Spam detection in email service is an example of a classification problem. Other applications of classification problem include credit approval, medical diagnosis, target marketing, speech recognition, hand writing recognition, biometric identification, document classification, soil classification, crop classification, intrusion detection, etc. Although there are variety of

available techniques for machine learning but classification is most widely used technique [Aized Amin Soof et al.]. Classification is an admired task in machine learning especially in future planning and knowledge discovery [Aized Amin Soof et al.]. Supervised classification is one of the tasks most frequently carried out by the intelligent systems. There are several classification techniques that can be used for classification purpose.

For the moment we can loosely state the learning task as follows: given the value of an input vector X, make a good prediction of the output Y, denoted by $\acute{Y}$. If Y takes values in $\mathbb{R}$ then so should $\acute{Y}$; likewise for categorical outputs, $\hat{G}$ should take values in the same set $\mathcal{G}$ associated with G.

3. Classification

The goal in classification is to take an input vector X and to assign it to one of K discrete classes C_k where k = 1, 2, …, K. in the most common scenario, the classes are taken to be disjoint, so that each input is assigned to one and only one class. The input space is thereby divided into decision regions whose boundaries are called decision boundaries or decision surfaces.

Classification can be either binary or multiclass. Formally, the binary classification problem is defined as follows: Given a set of m examples (x^j, y^j), j = 1, 2, …, m (the learning set) sampled from some distribution D, where $x^j \in R^n$, $y^j \in \{-1, +1\}$, the i^{th} component of x^j, x^j_i, which is termed feature i, find a function $f : R^n \longrightarrow \{-1, +1\}$ that classifies "well" additional samples $\{x^k\}$ sampled from the same distribution D.

Different classification algorithms differ in the form of function they are learning. Some of the common classification algorithms include Naïve Bayes, kNN, Logistic regression, Maximum Likelihood, Decision Trees, SVM, Fisher Linear Discriminant, Neural Networks, and ensemble methods such as Bagging and Boosting. What follows now is a brief discussion of some of these methods:

3.1. Various Classifiers: Overview

3.1.1 Naïve Bayes Classifier

Naïve Bayes Classifier: It is based on Bayes theorem and it is a conditional probability model; wherein the attributes (features) are assumed to be independent. That is, given a problem instance to be classified, represented by a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ representing some n features (independent variables), the classifier assigns probabilities $p(C_k, x_1, x_2, \ldots, x_n)$ to this instance for each of the K possible classes C_k by computing $(1/Z)*p(c_k)*p(x_1|c_k)*p(x_2|c_k)*\ldots*p(x_n|c_k)$, where $Z = \Sigma p(c_k)*p(\mathbf{x}|c_k)$. The class label then to the instance is

$$\hat{y} = \underset{k \in \{1,2,\ldots,K\}}{\operatorname{argmax}} p(c_k) \prod_{i=1}^{n} p(x_i|c_k)$$

3.1.2. kNN Classifier

kNN Classifier: The k nearest neighbors (kNN) classification scheme employs a more local method of classification. The method requires no initial processing of the training data, and new samples are classified based on their k nearest neighbors in the training set. It works by determining the k nearest neighbors and classify the sample based on the class labels of its k nearest neighbors. Since there are several metrics, the concept of proximity can be defined in several ways. Also, several decision rules can be proposed for assigning the class to a sample. In light of these choices, the kNN classifier has numerous variants. That is the choice of the metric and that of the decision rule determines that particular classifier. For example, a simple variant of kNN would find the neighbors based on Euclidian distance and use the majority rule to classify the new sample. In another variation each neighbor is given a weight according to its distance from the sample. In kNN, k is the only hyper-parameter of the algorithm. This kind of classifier can learn a relatively complex separation function. A drawback of this method is that in some practical problems, the Euclidian distance is inappropriate, and the "correct" distance metric is difficult to define.

3.1.3. Logistic Regression

Logistic Regression: It is a binary classification algorithm. The idea of Logistic Regression is to find a relationship between features and probability of a particular outcome. That is, it gives out the probability for something to be true or false. So it builds a regression model to predict the probability that a given data entry belongs to the category numbered as "1". Just as Linear Regression assumes that the data follows a linear function, Logistic Regression models the data using the logistic function or sigmoid function

$$g(z) = \frac{1}{1 + e^{-z}}$$

Logistic regression becomes a classification technique only when a decision threshold is brought into the picture. The setting of the threshold value is an especially important aspect of Logistic regression and is dependent on the classification problem itself.

3.1.4. Maximum Likelihood Estimate (MLE) Classifier

Maximum Likelihood Classifier: it is one of the most popular methods of classification in remote sensing, in which a pixel with the maximum likelihood is classified into the corresponding class.

The main idea of Maximum Likelihood Classification is to predict the class label y that maximizes the likelihood of our observed data x. Likelihood is defined as the posterior probability of x belonging to class labeled y. We will consider x as being a random vector and y as being a parameter (not random) on which the distribution of x depends. At first, we need to assume about the distribution of x (usually a Gaussian distribution).

Then, the learning of our data consists of the following [Dorian Lazar]:

- We split our dataset into subsets corresponding to each label y.
- For each subset, we estimate the parameters of our assumed distribution for x using only the data inside that subset.

When making a prediction on a new data vector x [Dorian Lazar]:

- We evaluate the probability density function (PDF) of our assumed distribution using our estimated parameters for each label y.
- Return the label y for which the evaluated PDF had the maximum value.

The maximum likelihood method has an advantage from the view point of probability theory, but care must be taken with respect to the following items [JARS].

(1) Sufficient ground truth data should be sampled to allow estimation of the mean vector and the variance-covariance matrix of population.

(2) The inverse matrix of the variance-covariance matrix becomes unstable in the case where there exists remarkably high correlation between two bands or the ground truth data are very homogeneous. In such cases, the number of bands should be reduced by a principal component analysis.

(3) When the distribution of the population does not follow the normal distribution, the maximum likelihood method cannot be applied.

3.2 Linearly Separable and Kernel Trick

Most classifiers such as Perceptron, Logistic Regression and the Support Vector Machines without kernels use linear functions to separate classes. If the data are not linearly separable, a linear classification cannot perfectly distinguish the two classes. Linearly separable data is data that can be classified into different classes by a line or a hyper-plane. In many datasets that are not linearly separable, a linear classifier will still be "good enough" and classify most instances correctly. In cases where a linear classifier is not good enough for classification, a nonlinear function can be used to separate instances that are not linearly separable. Two such nonlinear classifiers are k-nearest neighbors (kNN) and Kernel SVM. In Kernel method data is transformed using some nonlinear function so the resulting

transformed data points become linearly separable. That is a Kernel SVM is implicitly learning a linear separator in a higher dimensional space, but the separator is non-linear in the original feature space.

3.3 Ensemble Methods:

In Machine Learning, a model with low bias and low variance is considered a good model. To be able to solve a problem, we would like our model to have enough degrees of freedom to be able to resolve the underlying complexity of the data we are working with. But, at the same time, we want it not to have too many degrees of freedom to avoid high variance [Joseph Rocca]. Most of the time, we have models that have either high bias (models that have less number of degrees of freedom) or high variance (models that have considerable number of degrees of freedom). So, we combine several of these models to have an ensemble model that has better performance than any of these individual (component) models.

An ensemble of classifiers is a set of classifiers whose individual decisions are combined in some way to classify new data points. That is ensemble methods are meta-algorithms that combine several models into one single predictive model [Vadim Smolyakov]. This approach allows for better predictive performance compared to a single model [Vadim Smolyakov]. In line with this, it is observed that ensembles are often much more accurate than the individual classifiers which make them up [Thomas G. Dietterich, Ambika Choudhury]. Several methods of constructing good ensembles of classifiers exist in the literature. The original ensemble method is Bayesian averaging. This is followed by other methods such as Bagging and Boosting. Ensemble methods can be classified into sequential methods, where base learners are generated sequentially and parallel methods, where base learners are generated in parallel [Vadim Smolyakov]. AdaBoost and Random Forest are the respective examples of these methods. Most ensemble methods use a single base learning algorithm to produce base learners of the same type leading to homogeneous ensembles [Vadim Smolyakov]. Other methods use learners of diverse types; thereby leading to heterogeneous ensembles. For ensemble methods to be more accurate than any of its individual members, base learners must be as accurate as possible and at the same time as diverse as possible [Vadim Smolyakov]. Also, our choice of base learners should be coherent with the way we aggregate these models. If we choose base models with low bias but high variance, it should be with an aggregating method that tends to reduce variance whereas if we choose base models with low variance but high bias, it should be with an aggregating method that tends to reduce bias [Joseph Rocca].

4. Problem Statement

Here as part of our project we chose to classify Pima Indians Diabetes Data set by implementing Boosting technique (AdaBoost), wherein the weak (base) classifier is taken to be maximum likelihood estimation (MLE).

4.1 Boosting:

It is one of the most widely used technique in data science applications. It often considers homogeneous weak learners, learns them sequentially in a very adaptive way (a base model depends on the previous ones) and combines them following a deterministic strategy. That is, it refers to a family of algorithms that convert weak learners to a strong learner. It grants power to machine learning models to improve their accuracy of prediction [Sunil Ray]. In general, boosting tries to produce less biased models than its components even while reducing the variance [Joseph Rocca].

As discussed, boosting combines base learner to form a strong rule. A weak learner can be found by applying a base learner with a different distribution [Sunil Ray]. Each time base learning algorithm is applied to generate a new weak prediction rule [Sunil Ray]. While generating new weak rules, higher weights are assigned to previously mislabeled examples to arrive at the right distribution. This is an iterative process. After many iterations, the boosting algorithm combines these weak rules into a single strong prediction rule. Concise summary of the boosting algorithm is as follows [Sunil Ray]:

1. Apply base learning algorithm by assigning equal weight to each observation
2. Assign higher weights to those observations (examples) that are misclassified by previous learning algorithms
3. Apply base learning algorithm with the modified weight distribution computed in Step 2.
4. Iterate steps 2 and 3 until either the limit of the base learning algorithm is reached or the desired accuracy is achieved.
5. Now classify an observation by combining the outputs from the weak learners.

5. Results and Discussion

We have applied the algorithm on the Pima-Indians_diabetes dataset. We have analyzed the results by picking the 3rd feature first. Since the accuracy is a bit low, we have tried to select another feature and compare our results with the same. Also, we have tried splitting our data in two different ways to analyze and compare our results. Firstly, we have split the data equally into training and testing datasets. Secondly, we have given more data (66% of the original dataset) to the training phase and have used the remaining for the testing phase. We have also included the results that are obtained by performing the normal and basic Maximum Likelihood Estimation.

	Active_feature = 3						**Active_feature = 2**					
	Split = 0.5			Split = 0.66			Split = 0.5			Split = 0.66		
M	5	10	Single MLE	5	10	Single MLE	5	10	Single MLE	5	10	Single MLE
1	254	253	240	172	161	88	289	284	250	202	195	83
2	254	254	232	165	172	84	290	289	242	200	198	84
3	259	248	253	157	179	91	289	295	244	196	195	82
4	260	249	238	162	161	82	275	295	246	193	192	94
5	249	257	244	167	168	86	290	293	243	195	203	86
6	237	243	246	171	186	88	282	283	256	195	195	90
7	251	251	250	163	160	83	280	287	259	190	195	90
8	262	260	255	177	161	81	291	290	257	198	195	84
9	248	255	239	174	165	85	286	281	243	200	204	87
10	249	249	250	172	163	89	279	286	244	198	201	86
Mean	252.3	251.9	244.7	168	167.6	85.7	285.1	288.3	248.4	196.7	197.3	86.6
Acc	65.7%	65.5%	63.7%	64.3%	64.2%	32.8%	74.2%	75.0%	64.6%	75.3%	75.5%	33.1%

The above table is obtained by performing each case ten times to find out the mean of the tuples that are correctly classified and to estimate the accuracy of the classifier under the given conditions. M denotes the number of base classifiers whereas 'acc' denotes the accuracy of a certain classifier.

Observations:

- As we can see from the above table, Boosting clearly gives a better performance and has improved the accuracy of the basic MLE by at least 2%. A major difference in accuracy can however be seen when the data is split by a ratio of 0.66.
- We can also conclude that 10 base classifiers did not give any better performance when compared to 5 base classifiers.
- There is a trivial difference in accuracy of the boosting algorithm when compared against splitting of the dataset. However, when it comes to the case of applying MLE alone to the dataset, a noteworthy difference in accuracy can be seen when compared against the split of the dataset.
- Another crucial observation one can make is that of results when compared against the active feature. Clearly, selecting the 2nd feature gave better results over the 3rd feature.

Future Work

- A modification that we would like to do is to apply dimensionality reduction before we begin boosting so as to achieve maximum performance.
- We would like to extend this project to try and include multiple features.
- Also, we would like to experiment and find out if there is any optimal number of classifiers that one should use to give better results.

References

1. Sunil Ray, Quick Introduction to Boosting Algorithms in Machine Learning, https://www.analyticsvidhya.com/blog/2015/11/quick-introduction-boosting-algorithms-machine-learning/

2. Ambika Choudhury, Basics of Ensemble Learning in Classification Explained, https://analyticsindiamag.com/basics-of-ensemble-learning-in-classification-techniques-explained/, 23/05/2019

3. Joseph Rocca, Ensemble Methods: Bagging, Boosting, and Stacking: Understanding the key concepts of ensemble learning, https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205, 23/04/2019

4. Dorian Lazar, Maximuum Likelihood Classification: Implementing a Maximum Likelihood Classifier and using it to predict heart disease, https://towardsdatascience.com/maximum-likelihood-classification-4b129971ea92.

5. Aized Amin Soofi and Arshad Awan, Classification Techniques in Machine Learning: Applications and Issues, Journal of Basic & Applied Sciences, 2017, 13, 459-465

6. S. B. Kotsiantis, Supervised Machine Learning: A Review of Classification Techniques, Informatica 31 (2007) 249-268.

7. Ariruna Dasgupta Asoke Nath, Classification of Machine Learning Algorithms, International Journal of Innovative Research in Advanced Engineering (IJIRAE), Issue 03, Volume 3 (March 2016).

8. Hadi Hormozi, Elham Hormozi, and Hamed Rahimi Nohooji, The Classification of the Applicable Machine Learning Methods in Robot Manipulators, International Journal of Machine Learning and Computing, Vol. 2, No. 5, October 2012,

9. Alexander Vezhnevets, Vladimir Vezhnevets, 'Modest AdaBoost' – Teaching AdaBoost to Generalize Better, Graphicon-2005, Novosibirsk Akademgorodok, Russia, 2005.

10. JARS, Maximum Likelihood Classifier, Remote Sensing Notes, Edited by Japan Association of Remote Sensing, http://sar.kangwon.ac.kr/etc/rs_note/rsnote/cp11/cp11-7.htm

11. Vadim Smolyakov, Ensemble Learning to Improve Machine Learning Results: How Ensemble Methods Work: bagging, boosting, and stacking, https://blog.statsbot.co/ensemble-learning-d1dcd548e936

12. Thomas G. Dietterich, Ensemble Methods in Machine Learning, in Josef Kittler and Fabio Roli (Eds.), Multiple Classifier Systems, Proc. Of First International Workshop, Cagliari, Italy, June 2000, https://web.engr.oregonstate.edu/~tgd/publications/mcs-ensembles.pdf

APPENDIX

MatLab code implementing AdaBoost with Maximum Likelihood Estimate as the base classifier. This code is a modified version of the code given in the lecture 4 slides. That is the MatLab code implementing Maximum Likelihood Estimation classifier is modified to implement the AdaBoost classifier; wherein the base classifier is the MLE classifier. Number of base classifiers is denoted by M.

```
%{
Author - Sesha Phani Deepika Vadlamudi
Project - CS 559 Classification using Adaboost with MLE as the weak
clasifier
Version 2
Notes - Adding comments
%}


dataset = readmatrix('pima-indians-diabetes')
[m,~]=size(dataset);
%Splitting data into Training data and Testing data
P = 0.66 ;
idx = randperm(m)  ;
Trainingset = dataset(idx(1:round(P*m)),:);
Testingset = dataset(idx(round(P*m)+1:end),:);
% All the observations of training data are assigned equal weight
D = repmat((1/length(Trainingset)), [length(Trainingset), 1])
D0 = D(Trainingset(:,9)==0);
D1 = D(Trainingset(:,9)==1);
%for the purpose of implementation and understanding we have used only one
%feature
active_feat = 2;
%--------Training--------%
%M denotes the Number of base classifiers chosen
M=5;
for k=1:M
   % Initially each data point (observation) is assumed to be correctly classified by the weak classifier
   data_1 = [Trainingset, repmat(0,length(Trainingset),1)];
```

```
TrainingData0 = transpose(Trainingset(Trainingset(:,9)==0,active_feat));
TrainingData1 = transpose(Trainingset(Trainingset(:,9)==1,active_feat));
% Estimate parameters of the distributions of each class corresponding to kth weak classfier
mean1(k)= (TrainingData0*D0)/sum(D0) ;
mean2(k) = (TrainingData1*D1)/sum(D1) ;
var1(k) = var(Trainingset(Trainingset(:,9)==0,active_feat),D0) ;
var2(k) = var(Trainingset(Trainingset(:,9)==1,active_feat),D1) ;
% Computation of prior probabilities of the data based on the weights assigned to the data points
% These prior probabilities correspond to the kth classifier
prior1tmp = sum(D0);
prior2tmp = sum(D1);
prior1(k) = prior1tmp/(prior1tmp+prior2tmp) ;
prior2(k) = prior2tmp/(prior1tmp+prior2tmp);
correct=0;
wrong=0;
% Computation of likelihood estimates and hence posterior probabilities of all the training examples
% Also classify the training examples using the kth weak classifier based on the posterior probabilities

invvar1(k) = inv(var1(k));
invvar2(k) = inv(var2(k));
for i = 1:length(Trainingset)
   lklhood1 = exp(-(Trainingset(i,active_feat)-mean1(k))^2/(2*var1(k))) /sqrt(var1(k));
   lklhood2 = exp(-(Trainingset(i,active_feat)-mean2(k))^2/(2*var2(k)))/sqrt(var2(k));
   post1 = lklhood1*prior1(k);
   post2 = lklhood2*prior2(k);
   % Determine the data points which are misclassified by the kth weak classifier
   if(post1>post2 && Trainingset(i,9)==0)
      correct = correct+1;
      data_1(i,10)=1;
   elseif(post1 < post2 && Trainingset(i,9)==1)
      correct = correct+1;
      data_1(i,10)=1;
   else
      wrong = wrong +1;
```

```
    end
  end

% Modify the weights assigned to the data points based on whether the data point is correctly classified
% or not. Also determine the weight of the kth weak classifier while aggregating the results of all the
% weak classifier to have a strong classifier

%% Epsilon Calculation

  Eps_num = 0;
  Eps_den = 0;
  for i =1:length(Trainingset)
    Eps_den = Eps_den+D(i);
    if(data_1(i,10)==0)
      Eps_num = Eps_num +D(i);
    end
  end
  Eps = Eps_num/Eps_den;
  %% Alpha Calculation

  if(Eps>0&&Eps<1)
    alpha(k)= 0.5*log((1-Eps)/Eps);%% In fact, it should be assigned largest number. Here, we consider 2 to be
largest

  elseif(Eps==1)
    alpha(k) = -2;%% In fact, it should be assigned smallest number. Here, we consider -2 to be smallest
  end
  %% Computation of Weight distribution
  for i = 1:length(Trainingset)
    if(data_1(i,10)==1)
      D(i) = D(i)*exp(-alpha(k));
    else
      D(i)=D(i)*exp(alpha(k));
    end
```

```
    end
    z = sum(D);
    D = D/z;
    % New weights of data points of class “0”
    D0 = D(Trainingset(:,9)==0);
    % New weights of data points of class “1”
    D1 = D(Trainingset(:,9)==1);
end



%-----Testing-----%
correct = 0;
wrong = 0;
% Classify each data point in the test data
for i = 1:length(Testingset)
    post1 =0;
    post2 = 0;
    % Classify the ith data point in the test data by all the M weak classifiers, aggregate these decisions with
    % the corresponding weights, and come up with the final classification of this data point.
    for k = 1:M
        lklhood1 = exp(-(Testingset(i,active_feat)-mean1(k))^2/(2*var1(k)))/sqrt(var1(k))
        lklhood2 = exp(-(Testingset(i,active_feat)-mean2(k))^2/(2*var2(k)))/sqrt(var2(k))
        post1 = post1 +alpha(k)*lklhood1*prior1(k);
        post2 = post2 +alpha(k)*lklhood2*prior2(k);
    end
    if(post1>post2 && Testingset(i,9)==0)
        correct = correct+1;
    elseif(post1<post2&&Testingset(i,9)==1)
        correct = correct+1;
    else
        wrong = wrong+1;
    end
end
```