Georgia Institute of Technology

Master of Computer Science

CS7641 - Machine Learning

Supervised Learning

Author:
Alexis Durocher
(adurocher3)

Supervisors:
Prof. Charles Isbell &
Prof. Michael Littman

February 4, 2018

1 Introduction

In this assignment, the goal was to study the efficiency, differences and properties of different algorithms in supervised learning. In order to do so, we will use 2 datasets carefully chosen with properties emphasizing (as much as possible) the algorithms' pros and cons. The main judgment will be based on an error metric calculated on the testing set by cross-validation. Indeed, regardless of its properties, the main argument for choosing an algorithm more than another remains its final performance on real data.

The 5 algorithms studied here are classifiers. On top of a given dataset, a classifier builds a model so it can classify a new data (i.e give the class it belongs). To do so, classifiers use different techniques which performance and accuracy depend widely on the data characteristics. Indeed, the science of Machine Learning (and especially when it comes to classify) follows the famous theory of **'No-Free-Lunch'** implying that there is not ONE algorithm that works best for one type of problem. Many criteria have to be considered and different algorithms should always be tested and compared to find out the best option. This is our task for this assignment : finding the best option for each of our two classification problems and explaining why.

In this first report, I will describe the two datasets used (nature, volume, type of data, dimensions ...) before applying the 5 classifiers and comparing their performance.

2 Datasets

Those datasets have been chosen following criteria such as:

- volume : a relatively big amount of data usually provides more accurate models than few data. Also, if the dataset is too small, its properties are not visible enough.
- type of features : binary, numerical or categorical.
- dimensions : data should be described with enough features for the classifiers to learn an accurate model from. However datasets with 'too high' dimensions could also lead to more complex and less accurate classifiers.

Finally, the distribution of our data will also be taken into account since it is usually is a good indicator for explaining performances and failures.

2.1 Dataset Pima

This first dataset contains information about Indian women and whether they are diabetic or not. The first 7 columns corresponds to characteristics (medical attributes, number of times pregnant or age) describing each instance (woman). The 8th column is the label (diabetic or not).

This dataset isn't big in volume (532 instances recorded) but has a relatively high dimension (7) and no null values (already cleaned). This classification problem is binary since there is only two classes (diabetic or not).

Features are all numerical (continuous and discrete). Both classes are not equally represented but the difference remains acceptable ($\pi_1 = 1/3$ and $\pi_2 = 2/3$). By comparing the covariance matrix of each class, we can draw some similarities. Both classes seem to have the same distribution in their space and the features are not independent respectively to the class (covariance matrix are not diagonals). Some features are especially highly variated (ex : the glu feature) in both classes ($varGlu_1 = 589.85$ and $varGlu_2 = 977.50$). Finally the mean of each class is almost equal :

$$\mu_1 = \begin{pmatrix} npreg = 2.92 \\ glu = 110.02 \\ bp = 69.91 \\ skin = 27.29 \\ bmi = 31.43 \\ ped = 0.44 \\ age = 29.22 \end{pmatrix} \quad \mu_2 = \begin{pmatrix} 4.70 \\ 143.12 \\ 74.70 \\ 32.98 \\ 35.82 \\ 0.62 \\ 36.41 \end{pmatrix}$$

Visualization is impossible on dimension ¿ 3. However similar means and highly variated features in both classes makes us believe that both classes are not linearly separable. Intersection between classes seems relatively high.

2.2 Dataset Wine (quality)

This second dataset contains information about wines and the 'quality classes' they belong to. The first 11 columns corresponds to each instance's (wine) chemical characteristics and the 12th column to the quality label (3 to 8).

This dataset is bigger in volume (1599 instances recorded) but has a higher dimension (11). This classification problem is multi-classes since there are 6 classes of quality value.

Features are all numerical (continuous and discrete). Classes are very imbalanced ($\pi_5 = 0.42$ and $\pi_6 = 0.4$, the other classes are much less represented) which may impact the accuracy of our results (if not taken into account). For

this multi-classes classification problems we will use the algorithms to get insight about the linear separability of our classes (rather than comparing all covariance matrix and mean respectively to each class).

3 Algorithms

For each of our algorithms we will use a cross-validation based error metric for testing their performance.

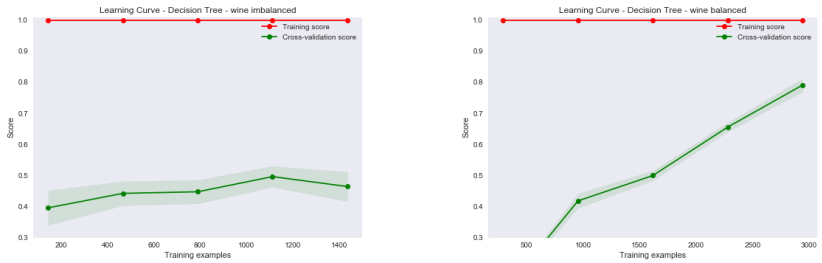
This will help us tuning our parameters if needed and comparing the efficiency of the algorithm on the 2 datasets.

We will use different curves to showcase our results. Depending of the algorithm, we will draw learning curves function of number of iterations or function of number of training example. To visualize the tuning of a single parameter and to analyze its influence (ex : maximum depth of node in Decision Tree), then we will draw validation curves.

3.1 Decision Tree

Decision trees do not require special data preparations. However such an algorithm produces biased decision trees if the data set is imbalanced in terms of classes.

Balance wine dataset Our second dataset is highly Imbalanced (two classes out of 6, have more than 80 % of all the instances). Therefore, we first start by balancing it by up-sampling minority classes with replacement, setting the number of samples to approximately match that of the majority class. We first use a row (i.e not tuned yet) decision trees to see the impact of balancing the dataset.

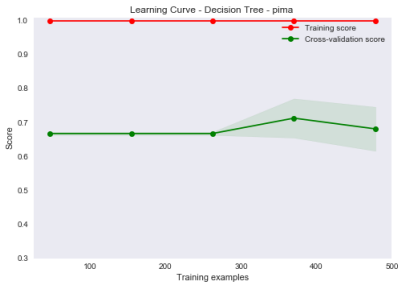


a b

Figure 1: Row Decision Tree performances on wine dataset
a. Imbalanced *b.* Balanced

Analysis fig 1 : The difference is clear. The cross-validation score on balanced dataset tends to increase constantly while the score on imbalanced is keeping a 'constant' low value (it even tends to decrease after a high number of training samples).
Hence, we will always use the balanced version of wine dataset for Decision Tree classifiers or Ensemble learners using Decisions Trees.

How to deal with Over-fitting ? Decision trees are very famous classifiers because they perform usually well and are easily interpretable (i.e the don't act as a black box that one runs without understanding it). The one we used here are based on CART Algorithm. Such a tree is based on binary nodes that asks if/else question on features' values. To choose the best feature to split on, it searches for the feature and split value that minimizes the entropy (here our metric is 'gini') to produce the purest following node (the one that will best split the dataset). This search for optimality makes this very flexible classifier, sensible to over-fitting.



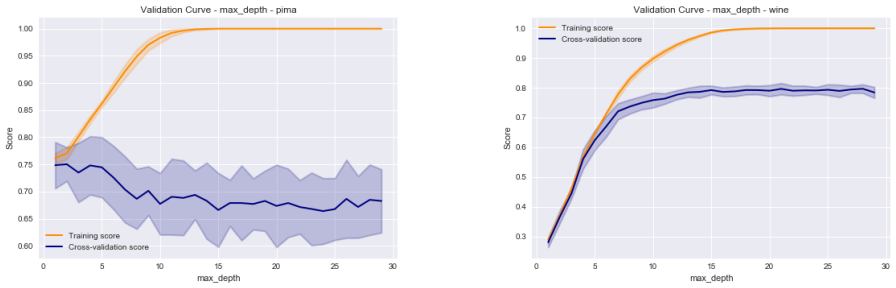
a

Figure 2: Row Decision Tree performances on pima dataset

The figure 2 is a good example of over-fitting. Indeed, the training score is constant and equal to 1 because the Decision Tree has effectively built pure leaves to split the training data on. However after a certain value of training examples (here around 380), the cross validation error tends to decrease which mean that the tree isn't generalized enough to be efficient on other than training samples.

To overcome this issue, a solution is pruning. Pruning is the idea of extending a tree until it becomes perfectly optimal and then progressively removing splits until we found a more robust Decision Tree.

In our case, we will do a manual sort of pruning. Our 'pruning' method will consist in limiting the growth of our tree by fixing a 'max depth' parameter in our classifier. We used a grid search to iterate over all the value of 'max depth', so we can compare the performance of cross-validation.



a b

Figure 3: Validation curve for maximum depth parameter
a. Pima Dataset *b.* Wine Dataset

Analysis fig 3 : First on both dataset, we can see that, according to the depth of our decision tree, the performances change widely.

Decision Tree on Pima dataset is sensible to over-fitting as we could have noticed in the learning curve (cf fig 2). Hence it makes sense that higher performance be provided for a more robust tree with very low 'max-depth' parameter (pima dataset : max depth = 1 and score = 0.750).

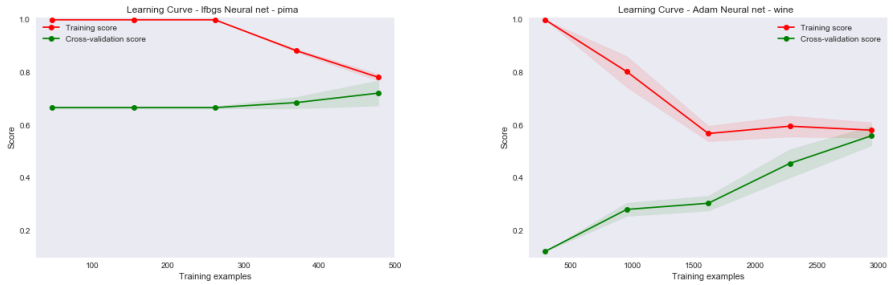
Decision Tree on Wine dataset was already providing a greater performance thanks to the help of balancing the dataset before hands. (max depth = 27 and score = 0.797). However, on the validation curve we can notice that the cross validation error reaches a 'plateau' for a value of max-depth around 15 and for a performance of around 0.795. Hence, a better (i.e more robust) decision would be to keep the smallest max-depth parameter and choose max-depth = 15 rather then 27. We will discuss further about this decision on the Boosting section.

3.2 Neural Networks

Here, we will use a Multi Layer Perceptron (MLP) Classifier. As defines by its name, this classifier builds multi different layers of Perceptron classifier to compute the most accurate output. A perceptron corresponds to one node and use an 'activation function' that computes a sum of weighted inputs to produce (or not) an output (based on a treshold). This model is usually flexible for complex boundaries and thus, can quickly become very complex and sensible to over-fit.

The optimization method used by the MLP should be adjusted regarding the dataset that we use (especially regarding its size for this parameter). A quasi-newton-like optimization method is known to perform better on smaller dataset but gradient descent is faster and better on larger dataset (no need to compute the Hessian matrix). After a study of both methods on our datasets, it turns

out indeed, that quasi Newton-like method ('lbfgs') provided best results on the smallest (pima) dataset and gradient descent ('adam') provided best results on the biggest (wine) dataset.



a b

Figure 4: Learning curve for Neural networks
a. Pima Dataset / lbfgs method *b.* Wine Dataset / adam method

Analysis fig 4 : On pima dataset : score = 0.722 and on wine dataset : score = 0.560.

Score is much better on pima dataset. The distribution of wine dataset led to a very complex neural network, due to the high dimension and high variance. Even if the trend of the curve tends to increase drastically proportionally to the number of training samples, we may believe that it won't reach a value higher than the score on the training set. Increasing the number of training samples would not help for this dataset and classifier.

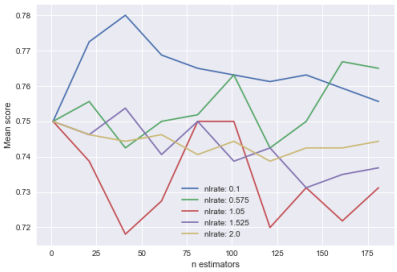
3.3 Ensemble learner - Boosting

Boosting is an ensemble learner classifier. It includes an ensemble of 'weak learners' (learners that perform, at least, better than random : ¿ 0.5 score). We use ADABoost algorithm. This algorithm trains iteratively weak learners by weighting the samples regarding the previous error : The error-prone samples will be weighted higher than others so the next weak learners will insist on them while being trained on.

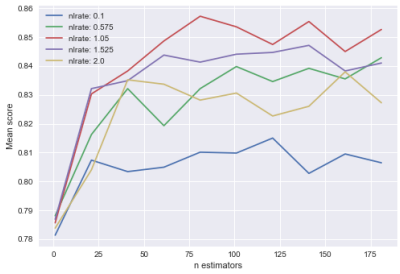
ADABoost supposes a score higher than those of every weak learners independently.

However, it is important to tune the algorithm's parameters to find an optimal trade off between the number of estimators (max of weak learners considered at which the algorithm stops) and the learning rate (defining the contribution of each classifier)

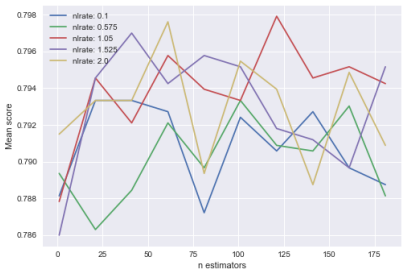
For this purpose of tuning, we use a grid search and plot the results showing the evolution of our boosting score for different values of learning rate and function of the number of estimators.



a



b.15



b.21

Figure 5: Learning rate and Number of estimators tuning
a. Pima Dataset *b.15.* Wine Dataset (decision tree with max-depth = 15)
b.21. Wine Dataset (decision tree with max-depth = 21)

Remark on individual estimators' parameters : Here, we use a Decision Tree Classifier as a weak learner. We want the most optimal DTree to be used for each ensemble learner. A solution would be to reuse the parameter setup previously found for optimal Decision Trees (cf. subsection 3.1). For Pima Dataset : max-depth = 1 For Wine Dataset : max-depth = 21.

However, one strength of Boosting remains in the variety of the ensemble learners it uses and it has to remain consistent with the dataset.

Increasing variety on Pima : A max-depth of 1 corresponds to a single tree with two leaves : it reduces drastically the freedom for decision trees to be diverse. To inject more freedom, thus more variety to the ensemble learners, we will use a DTree with max-depth = 2 for Pima Dataset. (the performance was very close to max-depth = 1 on Pima Dataset).

Increasing consistency on Wine : Regarding the Wine Dataset, the optimal Decision Tree was tuned with a max-depth = 21. We already mentioned the choice of considering a max-depth = 15 for a more robust classifier. As we can see on on figure 5 this decision makes all of its sense for the Boosting Algorithm.

For decision trees with max-depth = 21 (graph b.21), the ensemble learner has highly variated result. The wine dataset seems to small for having so deep DTrees aligned in an ensemble learner : inconsistent score. It is hard to draw a trend or any convergence because the ensemble learners are here 'too much variated' for the dataset.

Instead, a max-depth = 15 is a great trade off : similar very high performance on single decision tree and results are increasing smoothly and converging towards 1.

Analysis fig 5 :
On Pima Dataset, the trend remains unusual. Indeed, a boosting algorithm should not over-fit. Here, the blue line (learning rate = 0.10) reaches a top score for n estimators = 41, and then decreases. This might be explained by the fact that Pima is a really small dataset. Hence, increasing the number of estimators (41 aligned estimators is already a a lot for small decision tree of max-depth = 2) might lead to useless complexity and over-fitting in this particular case.

On Wine Dataset, the plot b.15 shows how robust boosting is. Indeed, it provides very great performance on a complex dataset and is not sensible to over-fit. Here the red line (lrate = 1.05) seems to attain a 'plateau' with small ups and downs. At this point (n-estimators = 81), increasing the number of estimators doesn't affect the score(no over-fit) but affects positively the confidence of our results. Adding different weak learners increase the confidence one has in the results but increases the computational time.

Finally we can say that the best (score = 0.780) ADABoost using DTree on Pima Dataset is setup with max-depth = 2 for the weak learner, n estimators = 41 and learning rate = .10 On Wine Dataset, the best (score = 0.857) ADABoost using DTree is setup with a max-depth = 15 for the weak learner, n estimators = 81 and learning rate = 1.05.

3.4 Support Vector Machine - SVM

Support Vector Machine is a classifier that uses similarity metrics (kernels) to compute and try to maximize a boundary margin between samples considered as support vectors.

First, SVM works and takes decision via a distance-based metrics. Therefore, it is crucial to normalize our dataset beforehand so we consider each feature within the same range (even if the variance is high). Indeed, a row dataset with highly different ranges of features' values would lead the SVM to be much more biased on certain features and will reduce its accuracy.

Second, it is impossible to visualize both datasets and thus get an insight of the data distribution so we could know which kernel and parameters use. A solution here is then to use a grid search to evaluate the different choices.

Finally, we have no clear idea of 'how much' each support vector should be considered by the model, and how complex the boundary line is. Our grid search will also have to look for the best C and gamma parameters.

Results The grid search has too much parameters for us to plot the results in a visible dimensions. The result obtained are the following :

On Pima Dataset:
Best score: 0.789
Best parameters: 'C': 100.0, 'degree': 1, 'gamma': 10.0, 'kernel': 'rbf'

On Wine DataSet
Best score: 0.578
Best parameters: 'C': 100.0, 'degree': 1, 'gamma': 10.0, 'kernel': 'sigmoid'

Analysis on the results
On pima Dataset, the resulting optimal SVM fits our first expectation (cf section 2.2 Dataset Pima) : The pima dataset is not linearly separable. Instead the best model uses a rbf (radial basis function) to maximise the margin between the classes.

Rmk : If we could visualize this multidimensional dataset we would eventually see a boundary with a 'round shape' between the two classes.

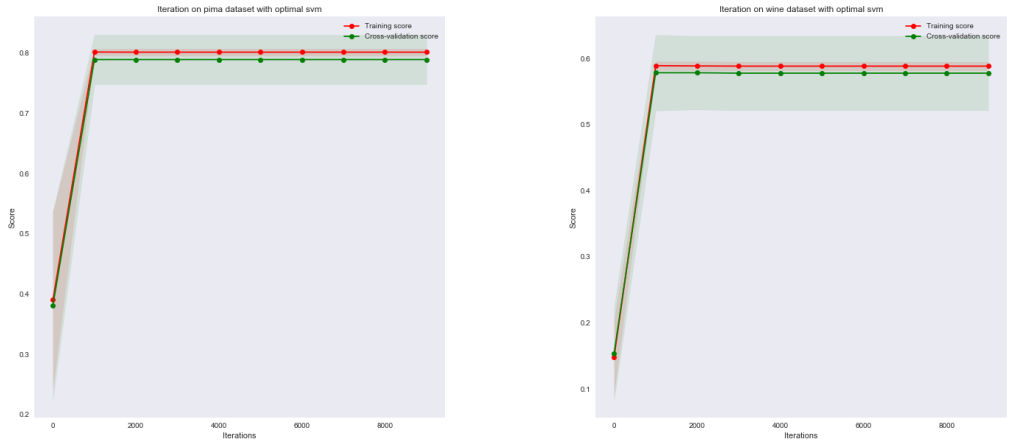
The High C and High gamma also corresponds to our expectations : Intersections between the classes is relatively frequent. This explains why the boundary has a complexe shape and why the best SVM needs to consider more instance as support vectors (high C) and gives more influence to the instances close to the boundary (high gamma).

Finally, on this dataset, the score is very high. This might be due to the fact that SVMs are classifiers which perform well on small datasets even if they are in high dimension (case of Pima).

On wine Dataset, the result answers also our first question : is wine data set linearly separable ? No. Instead, a sigmoid kernel provides a better result.

Also, we notice that a high C and high gamma is needed to provide the best score. Recalling that the Wine problem is a multi-classes classification problems, with highly variated features in big dimension : the boundary between the classes is very complex. This explains again the need for high C and high Gamma and the low score obtained.

Iteration SVM are iterative and very-long-to-compute models. An interesting information could be the minimum number of iteration necessary to get the best result in order to reduce the SVM's computational time. Here we will use an iterative learning curve.



a b

Figure 6: Iterative learning curve for SVM
a. Pima Dataset *b.* Wine Dataset

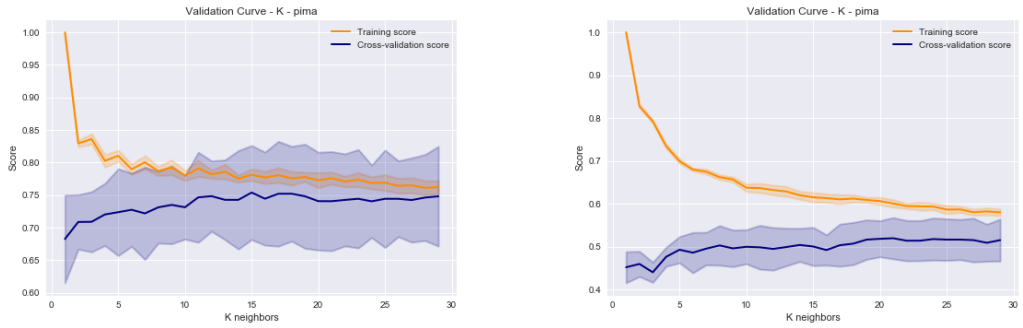
In both case, we can see that each SVM could set a maximum iteration to around 1800 to reduce computational time without reducing the model's accuracy.

3.5 K Nearest Neighbors - KNN

The KNN classifier is a lazy learner. It uses all the training dataset to compare a new data (test) and find the K closest (with a given metric distance) neighbors so it can makes a decision based on the neighbors' labels. It is a lazy learner in a sense that it doesn't build a model, but instead, uses all training data set at every queries to compute the output. Rmk : Again, it is very important to use normalize dataset for KNN (distance metric based classifier).

A very important hyper-parameters for KNN is the number of K (number of neighbors) to consider. Indeed, a 'too-small' K might lead to over-fitting and a 'too-high' K might lead to a lack in the accuracy of the model's decision making.

We will use a validation curve to see the influence of our hyper-parameter K on the classifier's score.



a b

Figure 7: Validation curve for KNN on K parameter
a. Pima Dataset *b.* Wine Dataset

Analysis fig 7 : Pima : score = 0.754, K = 14 and Wine : score = 0.519, K = 20. On both plot a and b, we can see that for a K = 1, the training score is equal to 1. This makes sense since, for each training sample queried, the model uses the sample itself as 1st neighbor. Hence, the label is always right. However this implies a very high over-fitting (very low score for test data in cross validation).

For both dataset, the cross validation score reaches a 'plateau' after a certain value of k and the training score continue to decrease (under-fitting).

On Pima dataset, the score is relatively high and best K corresponds to the beginning of the 'plateau'.

On Wine dataset, the score is low and best K is high. This emphasizes one weakness of KNN while applied on highly variated dataset. Indeed, the K needs to be high to consider further neighbors (because of high variance) but this lower the accuracy as it can also consider neighbors that have a different label. This is the case for Wine Dataset because there are many different labels (multi class problems) in a high dimension space where samples are represented with high variance.

3.6 Comparison

Most of our analysis and reasons for failure or performance have been described on each sections above. This comparative table helps to sum up.

For Pima Dataset , all the classifiers provided relatively good score (sup to 0.7). The Pima Dataset corresponds to a binary classification problem which is usually easier to solve regarding the size of our datasets. The SVM classifier provided the best cross validation score since SVM are very good classifiers on small datasets and it has managed to find a great boundary (rbf) between the two classes. It is interesting to point out that without tuning, results could have been completely different : validation and grid curves showed how diverse the results were, depending on the parameters tested.

For the Wine Dataset , we notice the performance of DTree or DTree-based classifiers (Boosting is the best on wine). One of the main reason is that DTree requires almost no data preparation and are particularly more efficient on multi outputs problems (multi classes problems). The wine dataset was used without being prepared (almost), thus it could have been interesting to apply some feature engineering and see how it impacts the performance of the other classifiers.

Finally the wine dataset also faced the 'curse of dimension' since it has very few samples comparing to its high dimension. Having at least twice more data (not only up-sampled ones) could have eventually helped in having better classifying performances.

	pima-1	vine-2
D Tree	0.750375	0.796703
Nr Nets	0.735029	0.543544
Boosting	0.780075	0.857230
SVM	0.789474	0.577861
KNN	0.753901	0.519758

Figure 8: Comparison of each cross validation score on the 2 datasets by the 5 algorithms