CS 7641: Supervised Learning Assignment

Ymakram3

# Introduction

In this assignment we are analyzing a set of supervised machine learning algorithms on two different datasets to gain an intuition on how different algorithms perform. We are using Python, sickit-learn, and matplotlib libraries to perform the experiments. The experiments are performed in a set of Jupyter notebooks, a notebook per algorithm. The algorithms are Decision Trees, Neural Networks, Boosting, Support Vector Machines, and k-Nearest Neighbors.

We are using two datasets to evaluate the different algorithms. The first dataset is a tic-tac-toe end game dataset, with a board state as features, and win or loose as prediction. The second dataset is MNIST digits dataset, with a set of handwritten digits pixels as features, and numbers as classes.

# Datasets

## Tic-tac-toe End Game Dataset

<https://www.openml.org/d/50>

The dataset has 958 instances representing all valid end game configuration of tic tac toe games assuming x played first. Each instance has 9 features representing one tic-tac-toe square. Each feature can be one of {x, o, b} values. The value x means player x has taken the square, value o means player o has taken the square, and value b means a blank square. The target label represents win or loose configuration with two possible classes {positive, negative}. The value positive represents a win for x, and value negative represent a loss for x. The class distribution is 65.3% positive and 34.7% negative. This is a slight imbalance that need to be addressed either with sampling, or some precision-based scoring.

This dataset was selected for the following reasons:

* Simplicity: The relatively small number of features and instances allows faster iterations on algorithms and experiment with API, and algorithm configuration, while not being trivial for meaningful results.
* Categorical: The features have string labels, which requires preprocessing for different algorithms to work.
* Complete: The dataset represents the full problem space of the game.
* Relatively small number of instances: It is interesting to compare how different algorithms perform on relatively small number of instances.

## MNIST Database of Handwritten Digits

<http://yann.lecun.com/exdb/mnist/>

The MNIST database has a training set of 60,000 examples of handwritten digits, and a test set of 10,000 examples. The database has been preprocessed and normalized to 28x28 images with various grey levels generated by anti-aliasing. The number of features is 784 corresponding to pixel values in 28x28 images. The target classes are {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} representing single digit numbers.

|  |  |
| --- | --- |
| **Class** | **Number of Instances** |
| 0 | 6903 |
| 1 | 7877 |
| 2 | 6990 |
| 3 | 7141 |
| 4 | 6824 |
| 5 | 6313 |
| 6 | 6876 |
| 7 | 7293 |
| 8 | 6825 |
| 9 | 6958 |
| **Total** | 70000 |

The dataset was selected for the flowing reasons:

* High Dimensionality: The dataset has 784 features, which is interesting to evaluate how algorithms are performing on high dimensional set.
* Relatively large number of instances: Interesting to compare the performance of different algorithms compared to the small number of instances in the first dataset.
* Relatively large of number of classes: Interesting to compare the performance of classifying 10 different classes compared to the two classes in the first dataset.

# Decision Trees

## Tic-tac-toe Dataset

The default DecisionTreeClassifier of sickit-learn generated a tree with maximum depth of 12 with no pruning. Performing a grid search across the two algorithms “gini”, and “entropy” where entropy is the information gain showed that “entropy” is performing slightly better on this dataset. It also showed that pruning decreases the performance.

Figure 1 shows the error rate against number of samples. The training loss is zero suggesting overfitting, while the validation score is increasing suggesting no convergence. The learning curve suggests a good validation accuracy at depth of 9, but when tried the test set performed a little bit worse with accuracy of 83.5% vs. 85.4 with no pruning.

Figure Learning Curve (no pruning)

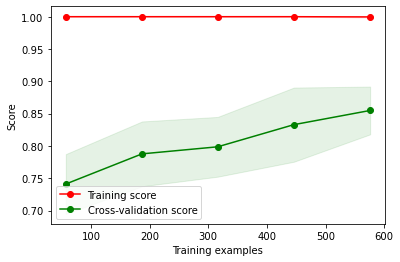


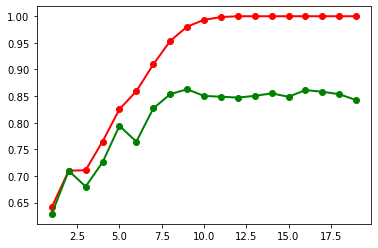
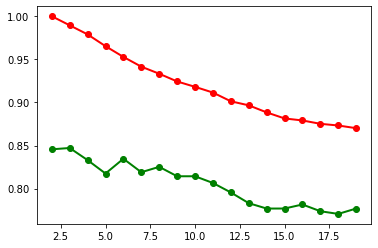
Figure Validation curve (max\_depth) 

Figure Validation curve (min\_samples\_split)



## MNIST Dataset

The default sickit-learn classifier yielded a tree with depth of 44. With a randomized hyper parameter search RandomizedSearchCV on criterion {gini, entropy} and a max\_depth between {1, 44} recommended a tree with depth 22.

The validation curve shows convergence at depth of 12. The tuned learner with entropy criterion and depth of 22 yielded 87.5% prediction accuracy.

Figure Learning curve (max\_depth=22) pruning

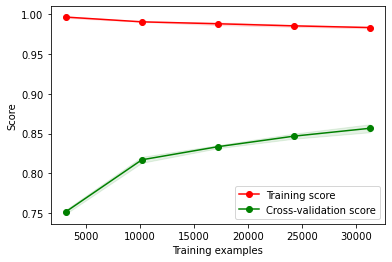


Figure Validation curve (max depth)

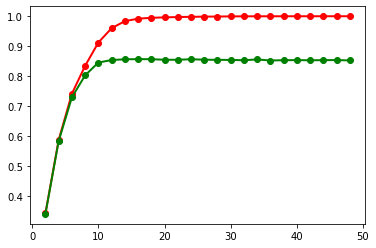
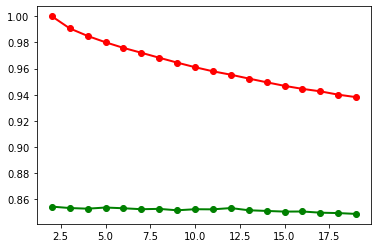
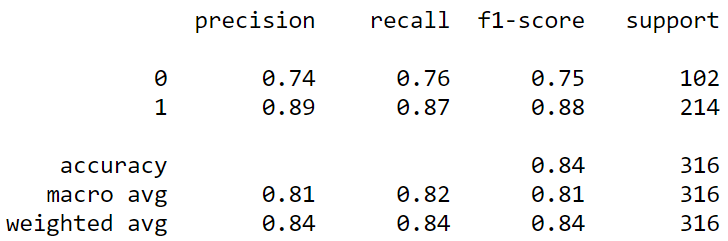


Figure Validation curve (min samples split)

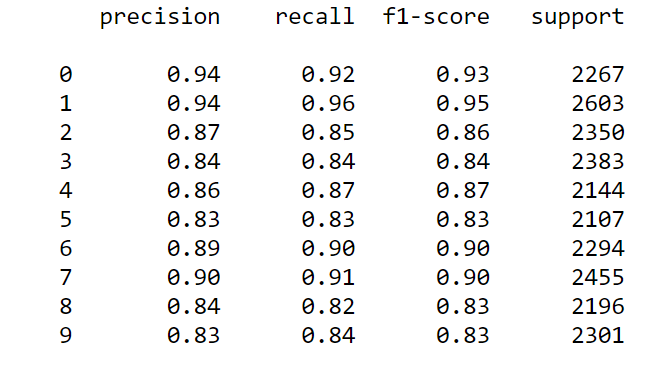


## Analysis

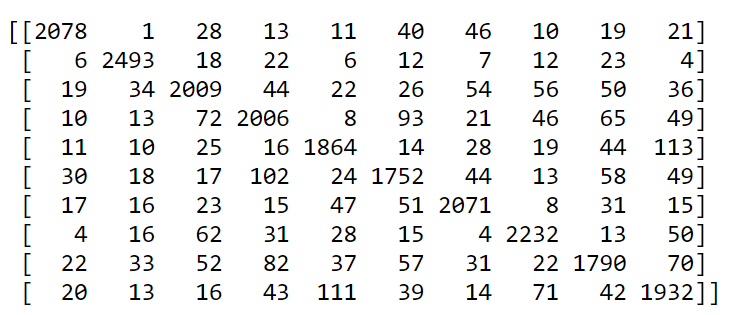
The decision tree classifier is performing poorly on the tic-tac-toe dataset. This was surprising since the data seems to fit a decision tree, but the classifier was overfitting and generalizing poorly. Pruning did not help improve the prediction accuracy. The classification report shown below that the precision and recall on the 0 class is worse than the positive class.



On MNIST dataset, the decision tree classifier seems to overfit as well. Even with pruning the overfitting is starting at lower tree depths and the accuracy seems to converge at 87.5% prediction accuracy. The precision and recall showing better performance on some digits like 0, 1 and poor performance on digits like 5, 9.



The confusion matrix further shows which classes are mis-predicted, for example it shows that the digit 9 is frequently confused with 4.



# Neural Networks

## Tic-tac-toe Dataset

The default multi-layer perceptron implementation with stochastic gradient descent, and logistic activation performed moderately on the dataset with accuracy close to decision trees.

A grid search showed Adam solver with RELU activation performing much better.

Figure MLP classifier with SGD, and logistic activation

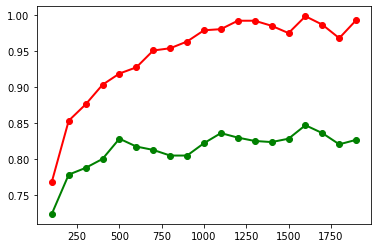
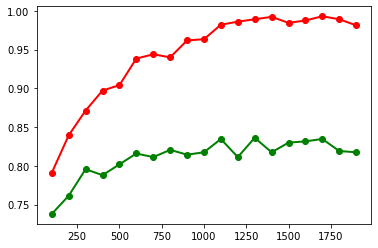
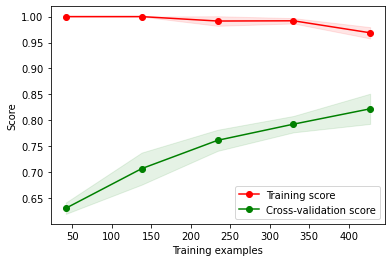


Figure MLP classifier with Adam and RELU



Increasing the maximum iterations is causing overfitting in both solvers with slight improvement in validation accuracy. The Adam solver is learning faster, with more stable validation curve.

Figure Learning curve Adam with RELU, max\_iter 900



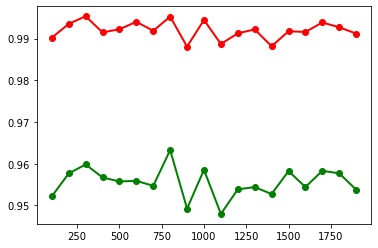
The grid search as well as the validation curve yielded a tuned max iteration of 900. The learning curve is showing improvement with more data and suggests the learner can still improve with more data.

## MNIST Dataset

The default MLP classifier with stochastic gradient descent solver and logistic regression performed fine on the MNIST dataset with 95% cross validation score. A grid search showed that Adam solver with RELU activation, and 800 maximum iteration is performing best.

The validation curve of the SGD showing instability in validation accuracy beyond 800 iterations suggesting overfitting, with the maximum iteration can be used for regularization.

Figure Validation curve SGD with max iterations



The validation curve with Adam solver and RELU has a more stable validation curve, but the performance is still best at 800 iterations.

Figure Validation curve Adam with max iterations

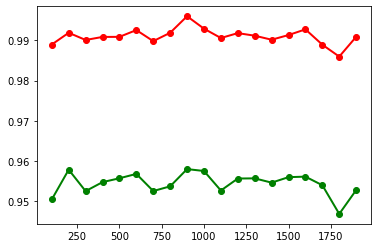
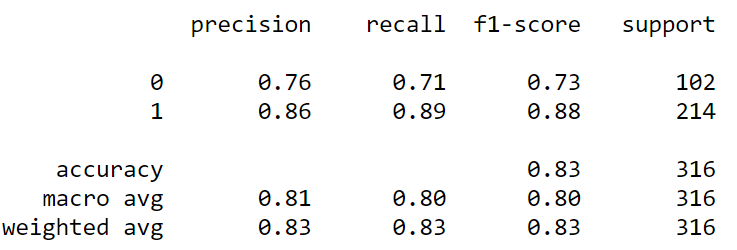


Figure Learning curve Adam, RELU, max iter 800

## 

## Analysis

The MLP is performing poorly on tic-tac-toe dataset. The learning curve is suggesting overfitting, with slight improvement as the learner gets more instances. This suggests the algorithm is not dealing well with small datasets. The accuracy, precision, and recall is performing slightly less than the decision tree.



The multi-layer perceptron, while still a simple network is performing well on the MNIST dataset. The network is learning the data fast, with slight overfitting that could be regularized with limiting the maximum iterations, and early stopping. There is still room for improvement with more complex networks but was beyond the time available for this exercise. The class precision and recall variance is much lower than decision trees. The confusion matrix is showing that the maximum confusion with 44 errors is between 9 classified as 4.

