CS7641 Supervised Learning HW1

Aleksandr Plokhikh (aplokhikh3@gatech.edu)

Abstract

The purpose of this assignment is to explore the performance of 5 supervised learning algorithms (Decision Tree, Boosting, KNN, Neural Networks, and SVM) on various interesting classification tasks.

Data set selection

For this task I took two classification datasets from Kaggle which I found interesting.

First is the fruits dataset [1]. I am a person who one time bought cilantro instead of parsley, and my wife was made on me for that. After this incident, I started to pay closer attention to fruits, and vegetables that I buy in a grocery store because I do not want my wife to be mad at me once again I think the ask of fruits classification is pretty important for many fellas. Also, I found that the fruits dataset contains fruits I have never seen in my life, and it would be handy if I could utilize ML algorithms to classify them. The data set contains 34 features, which is a lot. I do not think that I will do 34 measurements of each fruit in a grocery store but find the task fun anyway. It is also an interesting way to learn about new fruits.

The second dataset is the smartphone price classification table[2]. It contains features of a smartphone and one of 4 price ranges as a target for classification. This task is important for various reasons. First, it helps manufacturers to set the right prices for their phones to maximize their revenue, and profits. They do not want to sell phones with minimal margin but they do not want to let their phones sit on the shelf either. Consumers may also find the task handy as they can evaluate the fair price for the smartphone they want to buy because obviously most of them may not want to overpay for a phone.

Data preprocessing

I did a preprocessing for my data sets. First of all, for the fruits dataset, I changed the names of the fruits on numbers from 0 to 6 to facilitate the classification work. Also, I scaled all features linearly to range from 0 to 1 where 0 is the minimal feature value, and 1 is the maximum feature value. I did it in order to facilitate the KNN algorithm. Without proper scaling, KNN may not work properly since some features will be more important than others since they have a higher variance. The implemented scaling may not be ideal but it is the best neutral option that I can offer without extra domain knowledge about the data sets.

All datasets were shaffled and split into two sets: training (80% of data), and the testing one (20% of data) which I put aside, and never used for training, or validation purposes only for final testing of the ML algorithms performances evaluation.

Important note

I made a few steps to make the report more concise and save you time as the person who evaluates your report. I hope that all my measures will not be counted against me, I just want to save your valuable time. There are 5 methods, 2 data sets, and only 10 page limit. Thank you for understanding.

I used only one performance metric to be concise, and consistent. All learning curves are plotted based on the best estimators which are found with the grid search for best estimator. Not all the methods are evaluated with the very best estimators because in some cases the run of the best estimator with the best f-score is impractical because it takes too much time (it is true for boosting, and NN and will be described later). Accuracy-time to train- time to query is an important trade-off, and must be respected, and tuned. Also, I plotted learning, and validation curves to save space, and the full performance data is available in a separate pdf file (“output data.pdf” in the folder) and the program prints out.

Decision tree, Fruits dataset

For the task, I did a grid search for the best ccp alpha pruning coefficient, and the best criteria: gini or entropy. The best ccp alpha was pretty low – 0.014 which says that the tree is not very prone to overfit, and we can use a pretty deep tree for the task. A separate test says that entropy works almost as well as gini here, and there is not a sufficient difference between them.

Fig 1 shows learning and validation curves for the task. Cross-validation f-score steadily increases upon increasing the train set so it under fits, and extra training may help here.

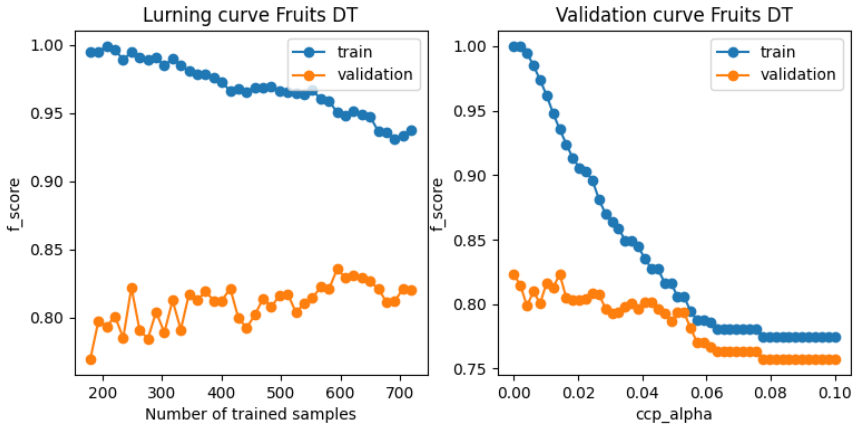

Fig.1 Decision tree learning, and validation curve on fruits dataset

The validation curve with respect to ccp alpha pruning coefficient shows that the coefficient below 0.1 leads to too aggressive pruning which reduces the f-score, but the performance significantly increases at ccp alpha below 0.1, and stays the same till values of 0.06 which may say that pruning does not change much in that here, Than bellow 0.06 the f -score rises for cross-validation sample and there is no significant change bellow value of 0.02 while the metric of the train set rises. So there is no sense to use ccp below 0.02 in order to reduce the size of the tree, but anyway, even with alpha=0.003 the tree takes only 18ms seconds to train, and less than 1ms to query.

The weighted f-score on the test set is 0.83 which is close to the f-score on the cross-validation. Our model is just right. The DT struggled with Deglet fruit (f-score -0.65) but shined with Rotana(0.93 f-score), and Savafi -0.98 f-score.

Decision tree, Fruits dataset

The best parameters in the grid search are entropy and alpha=0, which gives us a hint that the decision tree is not prone to overfit on the phone dataset as it does not require any pruning for the best performance. This guess is proved by the learning curve (Fig. 2) which shows the slow gradual rise of the f-score for the cross-validation sample with the increase of the sample size while the f-score for the training sample is 1 which tells us that the underfit of the model even with alpha=0.

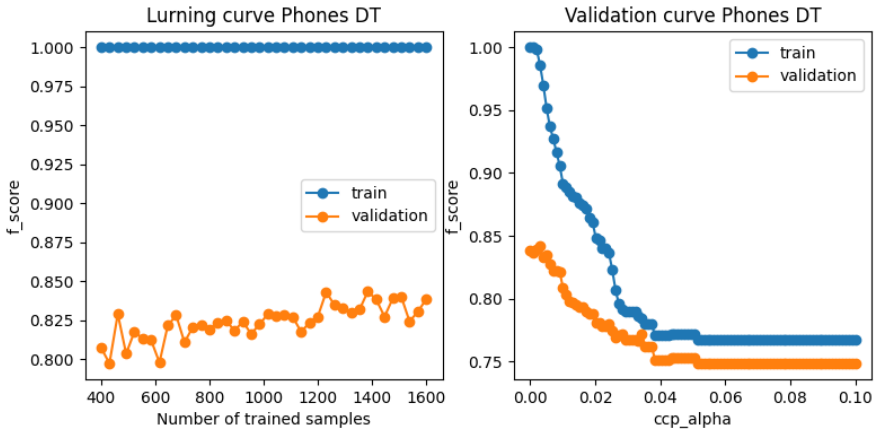

Fig.2 Decision tree learning, and validation curve on phones dataset

The alpha coefficient validation curve shows that changes in alpha in a wide range of relatively large alpha does not change the f-score of the model at all which most likely says that it does not affect on the decision tree pruning. Reducing the alpha bellow value of 0.04 gradually increases the f-score for the train, and validation sample, which says that now more deep tree is allowed, and it is not prone to overfit. The validation curve has a barely seen maxima in the area of very low values of alpha so we may say that here we have some sort of bias-variance trade-off but it is not substantial, and we can easily allow the largest tree with no pruning because it does not have other substantial penalties since the decision tree training takes 10 ms, and tree query takes less than 1 ms.

The weighted f-score of decision tree performance on the test sample with the best parameters is 0.875, which is even better than on the cross-validation sample so it means that our model is definitely not overfit. The model has the best f-score on the price range 0, and 3- the edge values (f-score -0.92). Most likely these categories correspond to the cheapest and most expensive phones, and the

decision tree easily finds the most expensive, and the least expensive ones, but the phones which are in the middle are a bit harder to classify which is expected for the decision tree model.

Boosting, fruits data set

Boosting learning can be done with any classification, so I decided to do it on decision trees. The grid search was done on the list of parameters: DT criteria (entropy or ginny), estimator splitting (best or random), DT max depth up to 10, number of trees up to 50 (usually the bigger forest is better, and we do not anticipate overfitting on this parameter, but forest learning takes time so I decided to be sensible and limit the parameter to 50).

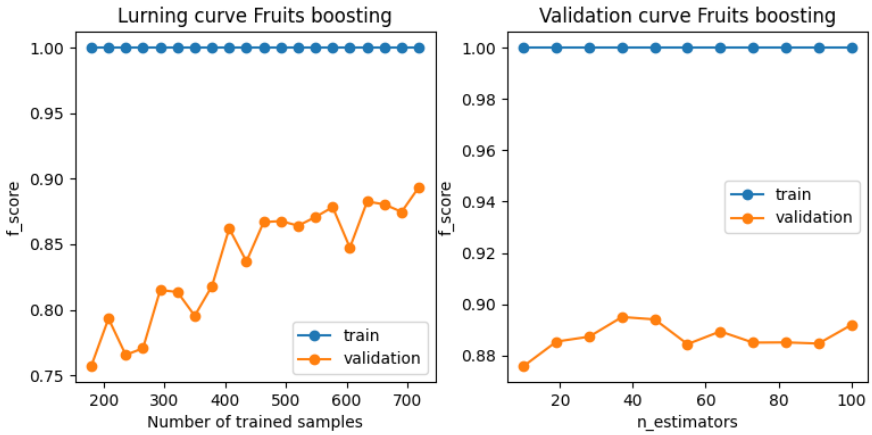

Fig.3 Boosting learning, and validation curve on fruits dataset

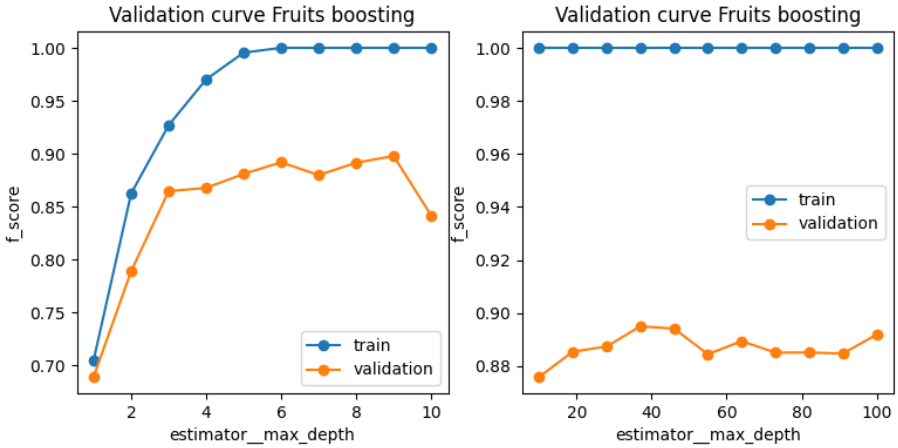

Fig.4 Boosting validation curves on fruits dataset

For the fruits data set the best DT criterion is entropy which is expected as the most popular one. The best estimator split is random, the max DT depth is 9 which also says that we may experience the bias-variance trade-off on this parameter, and the best size of the forest is 50 (the max size) which is expected since it is hard to make overfitting with a very large number of trees in the forest, and we underfit here on this hyperparameter.

The learning curve in Fig 3 has a strong uprising f-score trends for the validation set while the f-score for the training sample is 1 which tells us that the model is definitely undertrained and may show significantly better results in the larger database, we are far from the learning plateau here.

The validation curve on the number of trees hyperparameter (fig 3, and 4) shows that the performance of the forest increases once we have around 20 trees in the forest, and further increase of the tree number does not significantly increase the performance, so we may stick with the smaller forests since the learning of the best forest with 50 trees, and max depth of 9 now takes 223ms, and a query is 7ms which is the order of magnitude higher than for the simple decision tree from the previous chapter (18ms to train and less than 1ms to query with the minor boost in terms of performance (0.89 vs 0.85 of f-score for forest, and DT on cross validation respectively). The validation curve has maxima around 40-50, 100 trees but it mainly variance due to a relatively small cross-validation split of 5 for faster program run. No forest larger than 20-30 trees is needed here.

The validation curve on the max tree depth parameter (fig 4) shows that the performance of the forest significantly increases when we increase the depth to 3, then the rise of the performance is gradual up to 9, and we have a sharp decline at 10 which indicates the start of the overfitting the learner become strong, and ruin the idea of weak learners set. The best performance is at a depth of 9 but the sweet spot is around 6 since it must reduce the total train and query time for the forest which is substantial.

The weighted average f-1 score on the test sample is 0.93 which is substantially better than DT performance (0.83), and even better than on cross-validation step. The worse classification type is still Degled but the f-score of its classification is very significantly increased to 0.82 vs 0.65 in DT. The classification of Savati is 100% accurate. So the boost in performance is mainly due to the better classification of the types that were poor classified by DT.

Boosting, phones data set

According to the grid search the best criteria on the phone dataset is entropy, and the best splitting is “best”, so the forest gravitates to a sophisticated model, the best depth of the tree is 8 which hits that there can be an early and prominent bas-variance trade-off in this hyperparameter, and the best tree depth is 42 which should be read as manly variance of the data due to limited cross-validation splitting.

The learning curve in Fig 5 has a strong uprising f-score trends for the validation set while the f-score for the training sample is 1 which tells us that the model is definitely undertrained, and may show significantly better results in the larger database, we are far from the learning plateau here. As in the case of boosting on the fruits dataset. More data will lead to better performance.

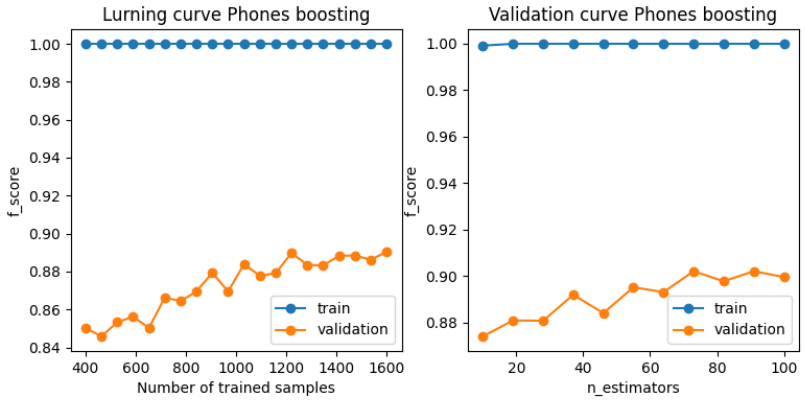

Fig.5 Boosting learning, and validation curve on phones dataset

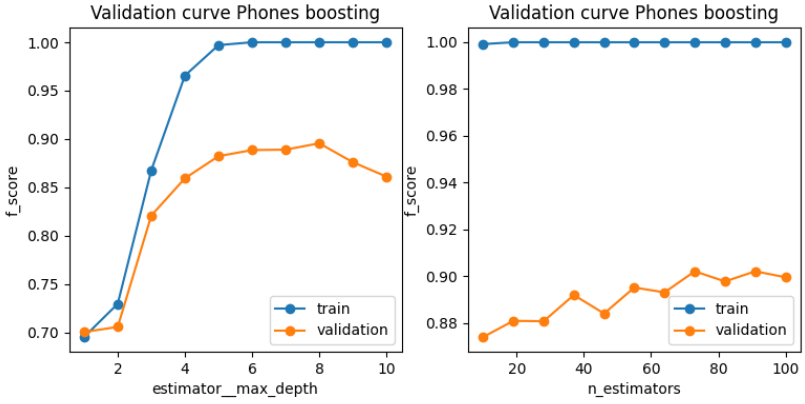

Fig.6 Boosting validation curves on phones dataset

The validation curve on the number of trees hyperparameter (Fig. 5, and 6) gives no surprises. The f-score on validation sets increases with the increase in the number of trees but forests larger than 70 do not have a significant performance increase and are not practical to pursue since the forest with 42 trees train takes 759 ms which is almost two orders of magnitude longer than the training of the DT optimized on alpha (10s) and query of the forest takes 7ms vs less than 1ms for DT.

The validation curve on the max tree depth (fig 6) shows a very prominent bias-variance trade-off and the best value for this hyperparameter. The f-score of the validation set rapidly rises when we increase the max depth to 5, then it does not substantially change, reaches the maximum at 8, and then steadily declines indicating that our learners are not weak here anymore and are prone

to overfitting. The best value is either 8 for the best f-score or 5-6 which provides shorter trees. Faster learning, and query which is very important for the very time-consuming boosting classifier.

The test set weighted f-score is 0.9, and only slightly better than the result of the DT on the same data set with the f-score of 0.875. So it is hard to justify boosted trees over a decision tree for this problem since the boosted forest learning is 75 times longer than training a single best tree, and the query is at least 7 times longer as well. Significantly larger train sets may change the state of things but we do not possess it. The best f-scores here as in the case of the simple DT for price rang 0, and 3 -for the cheapest, and most expensive phones.

KNN, fruits dataset

The grid search is based on the algorithm, leaf size, max number of neighbors (up to 20 for fruits database), and weight function uniform or distance.

The best number of neighbors for the fruits database is 3- it is the smallest K not have some variance in the neighbors and effectively chooses the answer best on the majority. The best weight function is distance. Most likely the distance weight function helps to effectively make a tie-break if there are 3 different classes in 3 neighbors. The closest example is the most probable which does make sense.

The learning curve in Fig. 7 has a strong uprising f-score trends for the validation set while the f-score even increases with a higher pace once we reach the limit if the database for the training sample is 1 which tells us that the model is definitely undertrained, and behavior of cross-validation curve says us that we may vary significantly increase the performance of the model if we add even a few more hundreds of samples in our database but of course, it will tank our query performance which very slow for KNN by design. KNN method here with 3 neighbors takes 10ms to query which is like the boosted trees method (7ms), and the training (here it is mainly data set load) takes 4ms as well (the fastest method so too much faster than decision tree which takes 18ms to train).

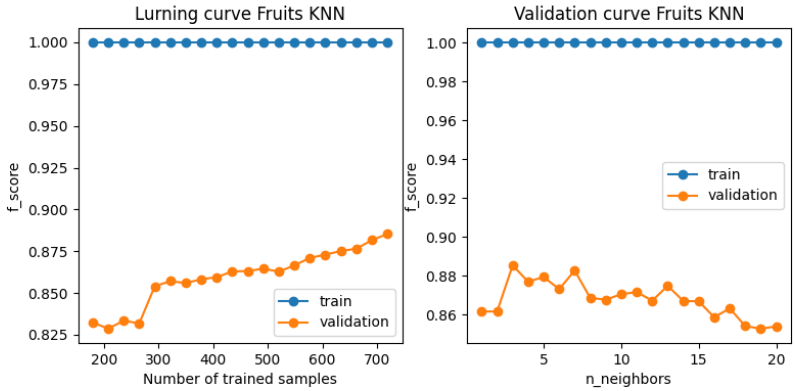

Fig.7 KNN learning, and validation curve on fruits dataset

Fig. 7 also contains a validation graph on the K parameter. The performance for 1, and 2 neighbors is the same as we expect with the distance metric since only the closest neighbor determines the classification. But when we increase K up to 3 the f-score rises. At K=3 we allow some reasonable neighbors set and the majority rules classification. Further increase of K monotonically decreases the performance of the module and at K>15 the performance of the module is worse than with only 1 closest neighbor. This phenomenon can be explained by the fact that some classes have a low feature variance, make large clusters, and make miss classification of queries somewhere on the edge of the cluster because the true class has a lower sample density, and lower number of entities with the right class are present in the KNN query sampling.

The weighted f-score on the best-performing model on the test set is 0.9 which is slightly less than the f-score of the boosted trees (0.93), but the KNN model takes no time to train while the training of the boosted trees takes hundreds of milliseconds. Boosting tree's query time is lower but insignificantly (10ms vs 7ms). The worst classification is on Deglet (f-score-0.77 lower than the boosted tree score of 0.82). Savati is classified with 100% accuracy as in the case of boosted trees.

KNN, phones dataset

First grid searches on the database showed the tendency of the model to take the largest K available with distance weighting. Eventually, I limited the K on the grid search to 300, and the optimizer took the value of 200 as the best one. I did not check K values higher than 300 because even at K=300 the query time is already 70ms and 7 times higher than even training of the decision tree on

the dataset (10ms), while the query of the decision tree is less than 1ms. So there is no practical point in exploring K values higher than 200.

The learning curve in Fig.8 shows that the f-score on the validation set increases up to a train size of around 700 samples. Further increase in the training size provides a slower rise of the f-score. So increase in the training size set may increase the performance further but it is hardly necessary – the f-score of the KNN on the data set is significantly lower than in other methods (around 0.6) and an increase in the training size set will lead to increase in the query time.

The validation curve in Fig.8 shows that the f-score of the cross-validation set rises when we increase the K value to 300, any further increase of K does not change the performance. There is no start of overfitting but there is no point in using high K ether since it raises the query time.

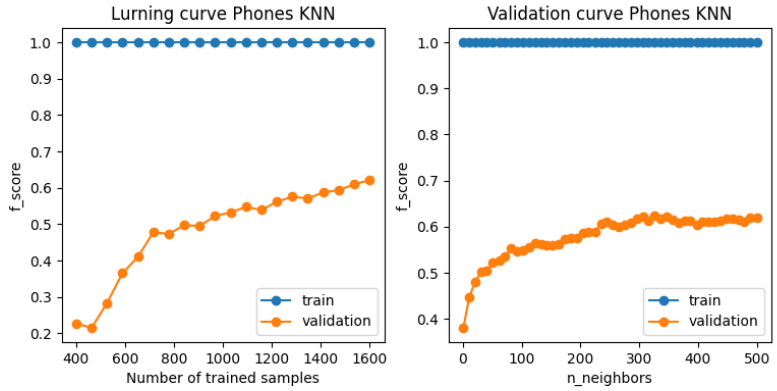

Fig.8 KNN learning, and validation curve on phones dataset

The performance of the model on the test sample is remarkably low (f-score is only 0.57) which is slightly less than the cross-validation performance of around 0.6. Classification of the cheapest and most expensive phones are higher with f-scores of 0.73, and 0.72 respectively, but the classification of group 1 is the worst f-score only 0.36 it is especially bad because the dataset contains only 4 classes while there are 8 fruit classes in other data set, add KNN performance quite good on the fruits dataset.

The most reasonable explanation for KNN's poor performance on the phone database is that not all features are equally important in the determination of the phone price. The right features weigh, and scaling does matter here, in the data preprocessing step we deliberately used the most basic scaling which does not work great on the phone's dataset.

Neural networks, fruits dataset

The grid search for the best classificatory was performed over further parameters: activation function ("identity", "logistic", "tanh", or "relu"), hidden layers structure (number of layers, number of neurons) ([10,10],[15,15],[20,20],[25,25],[30,30],[40,40]), initial learning rate from 0.05 to 0.5, and learning rate change method ('constant', "invscaling", or "adaptive").

Medium-size neural network 25x25 won, with 0.01 starting learning rate, adaptive learning rate, and tanh activation function.

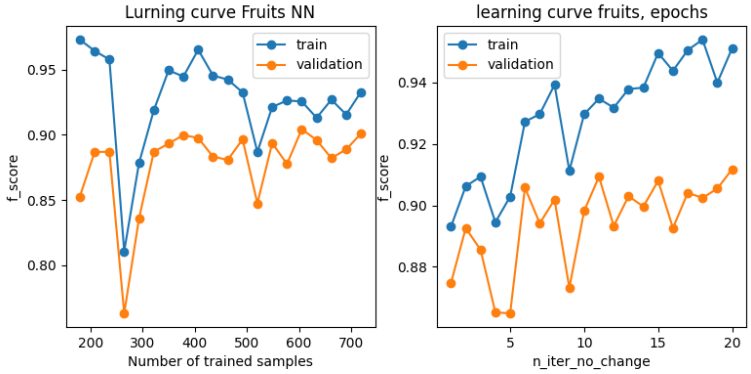

Fig.9 Neural network learning curves on fruits dataset

Fig. 9 shows two types of learning functions (in terms of train sample size, and in terms of the epoch number). The training set sample size variation learning curve shows gradual increases of the f-score on the cross-validation sample with the increase of the training

sample size, but further increase of the training data set may not increase the performance significantly since the performance of cross-validation sample affine the performance of the training set which does not really changes with the variation of the training sample size. The f-score of 0.94 is the natural limit here. It is also worth pointing out two very significant dips in the learning curve in the area of relatively low training data sets. It can be explained by the fact that a neural network is a complex mechanism with hundreds of neurons, and thousands of links (in our case) which dynamically affect each other and may result in variations of performance on undertrained neural networks.

The second learning curve shows that there is not much sense of training the neural network for more than 6 epochs, further training leads to only a marginal increase of the f-score on the training set.

Fig 10 shows validation curves for the NN in which I independently vary the number of hidden layers, and number of neurons in hidden layers. The performance of the neural network decreases with the increase in the number of neurons in the net. This is true for both the test and validation sets. Overcomplexity does not help here and the smallest layers perform the best. The change in the number of hidden layers does not have a profound impact on the model performance and all the changes are within the margin of variation. Probably the lower number of layers and less complex model works better. So in general smaller NNs perform the best on the fruits database, and there is no point in a more complex model that takes more time to train. Even a relatively simple 25 x25 model takes 255ms to train (similar to boosted trees), and 2 ms to query.

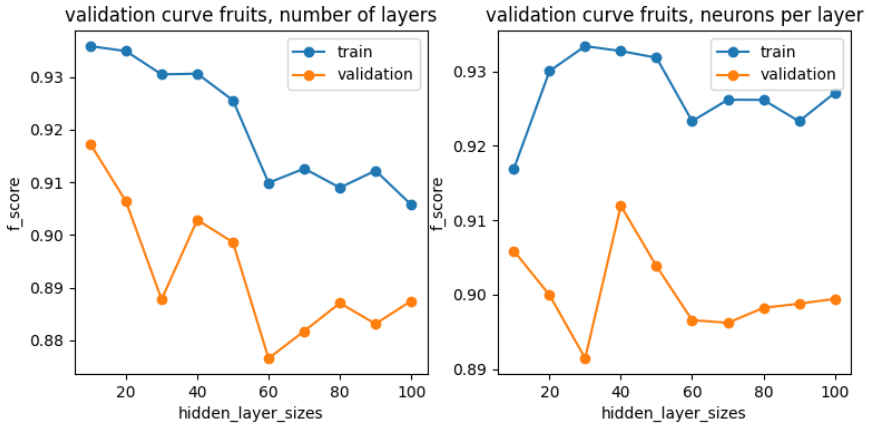

Fig.10 Neural network validation curves on fruits dataset

The weighted f-score on the best-performing model on the test set is 0.92 which is very slightly less than the f-score of the boosted trees (0.93). The worst classification is on Deglet (f-score-0.83 the same as the score for the boosted trees of 0.82). Savati is classified with 100% accuracy as in the case of boosted trees. In general, the performance of NN matches the one of boosted trees but query time is significantly low for neural networks 2ms versus 7 ms for boosted trees.

Neural networks, fruits dataset

The midsize 25x25 network performed the best on the phones dataset with an initial learning rate of 0.01, constant learning rate, and logic activation function.

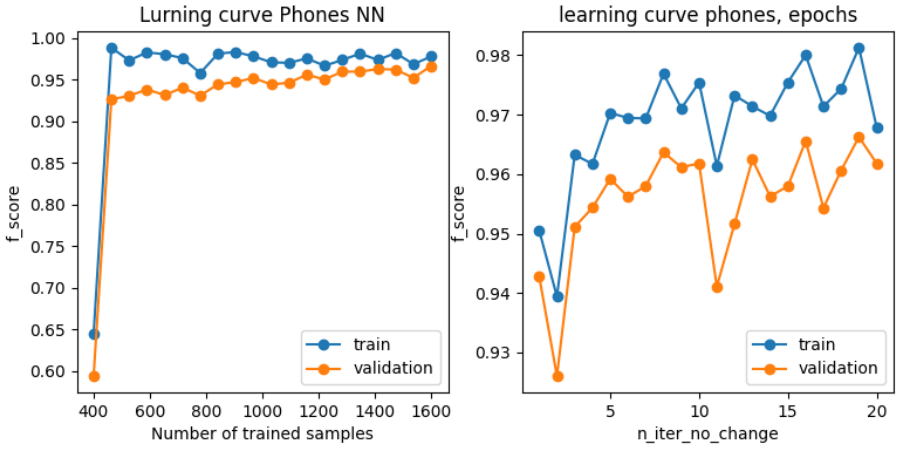

Fig.11 Neural network learning curves on fruits dataset

As in the case of the fruits dataset, the neural network learns very fast and reaches almost its full potential after running only on 500 samples (see Fig 11). The performance increases steadily and monotonically with the increase of the data set size and the f-score reaches values over 0.95 for the cross-validation sample, the further training on new data may not significantly increase the performance of NN since the performance of it is very close to the one of the training sample. The neural network is trained just right. The performance of the network also steadily rises with the increase of the epoch number for both train and cross-validation samples and reaches the f-score of 0.96 for the validation set. The neural network does not exhibit any signs of overfitting, just a steady increase in performance which is remarkable. The network took 595ms to learn which is comparable with the time needed to train boosted trees forest (759ms) but this time can be cut in about 3 times if we will train the network only on 500 samples, the resulting performance will exceed the ones of all the previous methods described above for the phone's dataset. The query takes 2ms the same as the neural network on the fruit dataset which does make sense since the sizes of the nets are the same 25x25.

Validation curves in Fig.12 show that the performance remains more or less the same when we increase the number of the hidden layers from 10 to 70, but then the performance deteriorates for both the train, and validation samples, which says the network becomes too complex for the task, and the dataset. On the contrary, the performance of the network greatly increases when we increase the number of neurons up to 40 neurons per layer, and then the f-score does not change that much. This positive effect may be related to the fact that the dataset has 35 features in it, and nets with a high number of neurons in hidden layers perform better.

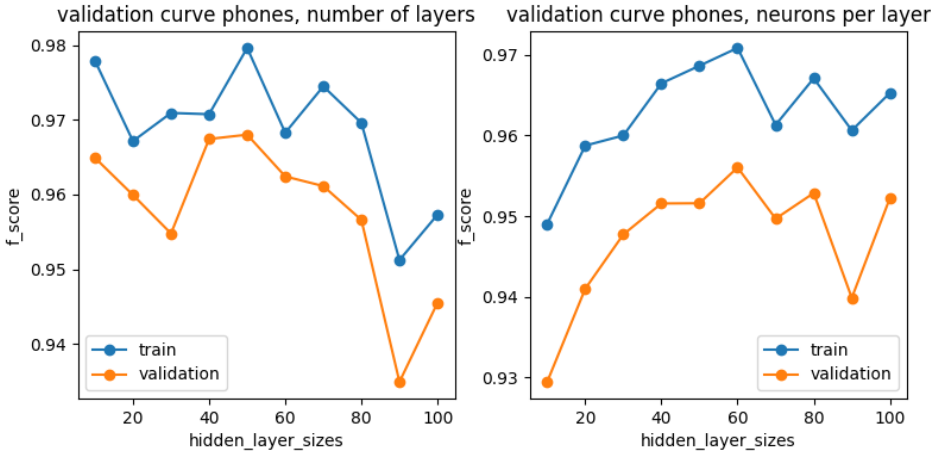

Fig.12 Neural network validation curves on phones dataset

The NN performed well on the test sample as well with a weighted f-score of 0.9675, which is the record so far for all classifiers and all databases described above. The NN as all other algorithms performs a bit better on the edges of the total price range but for NN classification of the middle-price phones is very effective with an f-score higher than 0.956. There is almost no difference between classes in turns of performance.

SVM, fruits dataset

The grid search for the SVM classifier was performed on the following parameters: kernel type ('linear', 'poly', and 'rbf'), C parameter (from 0.2 to 2), degree for the polynomial kernel (from 2 to 5), and gamma.

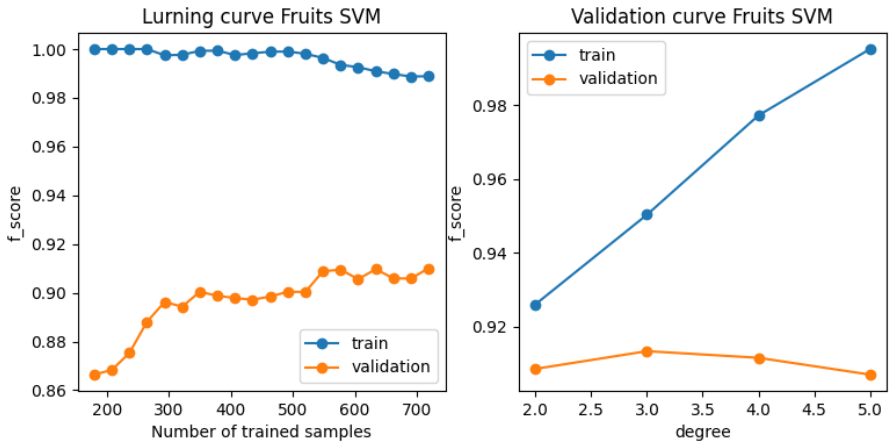

Fig.13 KNN learning, and validation curve on phones dataset

A cubic polynomial kernel showed the best performance and was used for learning curve construction. The learning curve in Fig. 13 shows the f-score steadily increases with the increase of the training sample, while performance on the training sample remains very high with an f-score of around 0.99, which means that the model is undertrained, and additional data can significantly enhance the performance of the model.

The validation curve shows that the f-score on the training set rises with the increase of the polynomial degree, while the performance on the cross-validation sample goes through the maximum ad degree equal to 3. We definitely have a very prominent overfiring at polynomial degrees more than 3, and the best bias-variance trade-off is simple quadratic, and cubic kernels, and we should choose them in order to avoid overfitting.

The weighted f-score on the best-performing model on the test set is 0.93 which is the same as for the neural network, and boosted trees learner, but SVM took only 11ms to train, and 1ms to query which is an order of magnitude lower than the corresponding values for boosting trees, and NN learners. The worst classification is on Deglet (f-score-0.77 which is worse than the result of boosting trees, and neural network). Savati is classified with an f-score of 0.98 (as in the case of a simple decision tree -only one mistake for the entire class in the test sample). The SVM seems to be the best all-around learner for the fruits dataset especially taking into consideration the fact that there is still room for additional training there while for example neural net exhausted all its learning potential well before the end of the training set.

SVM, phones dataset

Quadratic polynomial kernel showed the best performance on the dataset.

The f-score of the cross-validation sample quickly rises on the learning curve (see Fig. 14) while the corresponding value for the training set does not change much and rests around 0.96. The model is undertrained and can quickly increase its performance with additional training without savior penalties since the model training takes only 49ms to train and 4ms to query. The training time is an order of magnitude faster than boosted trees, and neural network learners.

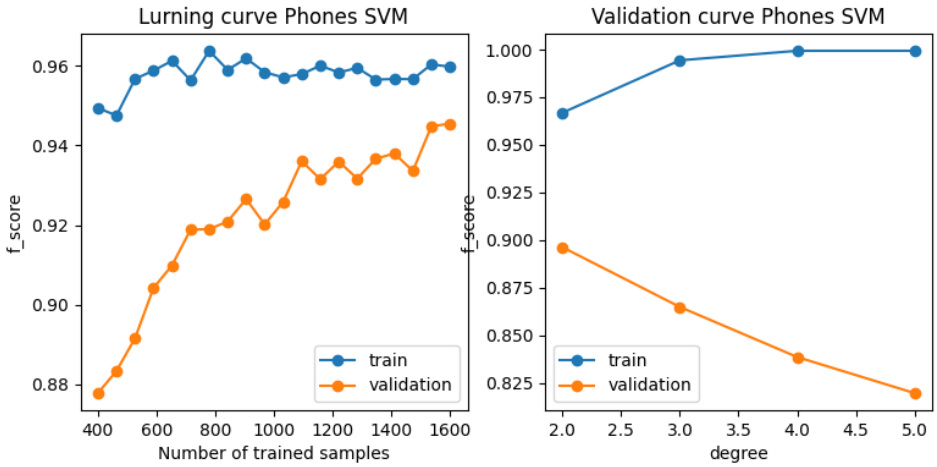

Fig.14 KNN learning, and validation curve on phones dataset

The validation curve shows that the f-score on the training set rises with the increase of the polynomial degree, while the performance on the cross-validation sharply goes down right from the beginning. We are on the very prominent overfitting regime and should restrain our model to the most simple quadratic kernel. More complex models are not only wasteful but also provide significantly worse performance.

The SVM performed well on the test sample as well with a weighted f-score of 0.96, which is which is just a bit lower than the performance of the neural network. The NN as all other algorithms performs a bit better on the edges of the total price range but for NN classification of the middle-price phones is very effective with an f-score higher than 0.94. There is a difference between classes in turns of performance. The SVM seems to be the best all-around learner for the phones dataset especially taking into consideration the fact that there is still room for additional training there while for example neural net exhausted all its learning potential well before the end of the training set.

Conclusions

Both selected datasets are treatable by all five types of supervised learning algorithms and provide a good understanding of the benefits and drawbacks of each method. The performance of the all methods and all test datasets where no worse than the

performance of the same methods on cross-validation samples which says that dataset preprocessing, and split of the dataset on train, and test were performed correctly.

NN, SVM, and boosted trees showed identically good performance on the fruits dataset with an average f-score of 0.92-0.93. The KNN had almost as good performance – f-score=0.9 which is remarkable considering that there is no actual learning in the algorithm, The decision tree showed a worse but still acceptable result of 0.83 in terms of f-score. It is still good taking into account that the dataset contains 8 distinct fruit classes, and it is also worth mentioning that a simple decision tree has the fastest query (less than 1ms even the "time" python module cannot measure the actual query time) The worst fruit for classification was Deglet, while Savati was identified with 100% accuracy for all the method except of simple decision tree, and SVM which made only one mistake in the classification of the fruit. The SVM can be considered the best classifier here since it comprises one of the shortest learning, and query times while having one of the highest prediction accuracy, and potential for further training of the model without the risk of overfitting, or losing time.

SVM and neural network were the best on the phones dataset with the f-score of 0.96-0.98. Between the two methods, we should choose SVM since it trains and queries a few times faster than neural networks even if we will train the net only on 1/3 of the dataset. The performance of the KNN method was remarkably low (f-score=0.57 on the test sample), and the issue here is the fact that not all the features have the same impact on the price of the phone, and our nail linear min-max scaling was clearly not the best here but we did not possess any domain knowledge to chose any other scaling technique over the very basic one. The most cheap, and the most expensive phones are determined with the highest accuracy in all learning models, while middle-prized phones are a bit harder to classify, which does make sense. The closer analysis of confusion matrices (see program printout, a "print out.pdf") reveals if the classifier makes a mistake then it assigns the phone to a neighbor class, and never makes a mistake when phones are misclassified for 2 or more price tiers, which is quite in testing, and remarkable. The only exception is the KNN method which has the worst performance. And even KNN never mixed up the most expensive phones with the cheapest ones.

Algorithm-wise the fastest to train was the KNN method since there is no actual training happening, and we consume time only on the creation of the classifier, and passing the database to it which takes 4-10ms depending on the size of the database here. But it is the slowest algorithm to query and may take up to 70ms to query which does make sense since we need to scan the entire database for the closest neighbors, some other methods like decision tree, or SVM take less time to query the KNN model. KNN also completely failed the phone database.

Decision trees were the fastest to query (less than 1ms) which again does make sense especially if we have aggressive pruning. There is only one supervised learning algorithm that is faster – linear regression, but it is not appropriate for classification problems. Decision trees were also one of the fastest to train which also does make sense since boosted trees, and neural networks need much more calculations to build a model. It is also interesting that the decision tree was not prone to overfitting on the alpha pruning parameter. Seems like the fruits, and phones datasets are very overfitting prone by itself.

Boosted trees consistently outperformed simple decision trees in terms of accuracy on test samples but they required orders of magnitude higher time to train, and query, since we boosted forests contained 50, and 300 trees for fruits, and phones datasets respectively.

Neural networks showed consistently very good performance on both datasets and had one of the lowest query times around 2ms. They also came to one optimal performance much quicker than other learners. They needed only 1/3 of the training set for an f-score of more than 0.9, and 0.94 for the fruits, and phones datasets respectively. So Neural networks could be the best candidates for the learner if we had a much shorter database.

The best learner for both datasets was SVM with a simple polynomial kernel it comprises lower training query time with very high consistent performance, and it still has room to improve the performance and learn more if the data sets were just a bit larger, it could make SVM an overall champion of the learners on this two databases, but anyway, SVM is the best all-arounder here.

P.S.

Now I need an algorithm to distinguish parsley from cilantro. Because I am struggling. :)

Citations

[1] https://www.kaggle.com/datasets/muratkokludataset/date-fruit-datasets

[2] https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification