CS 7641 Machine Learning

Chen Peng, cpeng78

c.peng@gatech.edu

Assignment 1: Supervised Learning

1. Datasets Introduction

Chronic diseases and their expensive treatment costs are the most important factors driving the surge in medical expenditures in the global healthcare industry. Diabetes and obesity are among the top ten chronic diseases in the United States. According to the data from CDC website, in 2017, the Americans spend on diabetes treatment was $327 billion. The total estimated spend on obesity treatment was $147 billion yearly. More than 34 million Americans have diabetes, 1 in 4 of them don't know they have it, and another 88 million have prediabetes. And the prevalence of obesity was more than 42.4%. One of the key methods to controlling and reducing medical expenses for the diabetes and obesity is early prediction and detection the diseases.

Here I found two datasets from the UCI Machine Learning Repository, Dataset 1 is an early stage diabetes risk prediction dataset. The dataset contains 16 attributes (Age, Gender, Polyuria, Polydipsia, sudden weight loss, weakness, Polyphagia, Genital thrush, visual blurring, Itching, Irritability, delayed healing, partial paresis, muscle stiffness, Alopecia, Obesity) for 520 diabetic patients. The target labels are 320 positives and 200 negatives. Dataset 2 is an estimation of obesity levels based on eating habits and physical condition. The dataset contains 16 attributes (Gender, Age, Height, Weight, family_history_with_overweight, FAVC, FCVC, NCP, CAEC, SMOKE, CH2O, SCC, FAF, TUE, CALC, MTRANS) of 2111 records. The target labels are 7 different weight levels, each of the class has approximately 300 instances. The label distribution are almost evenly distributed, which is good to train.

2. Preprocessing of the datasets

Both two datasets have no missing data. Each class of target labels have about same number of instances. These made the datasets easy to use. Figure 1 shows the distribution of the target classes for the two datasets.

For dataset 1, one of the features, Age, is integer. The other 15 features and the target are binary classes. Therefore, I encode the categorical features as an integer array and encode target labels as

2 classes. For dataset 2, 8 features are continuous value, which do not need pre-encoding. The other 8 features are categorical features, and each of the features are degree levels. Therefore, I encode the 8 features with ordinal encoder instead of one-hot encoder. The target of dataset 2 are 7 degree classes, I encode them with label encoder.

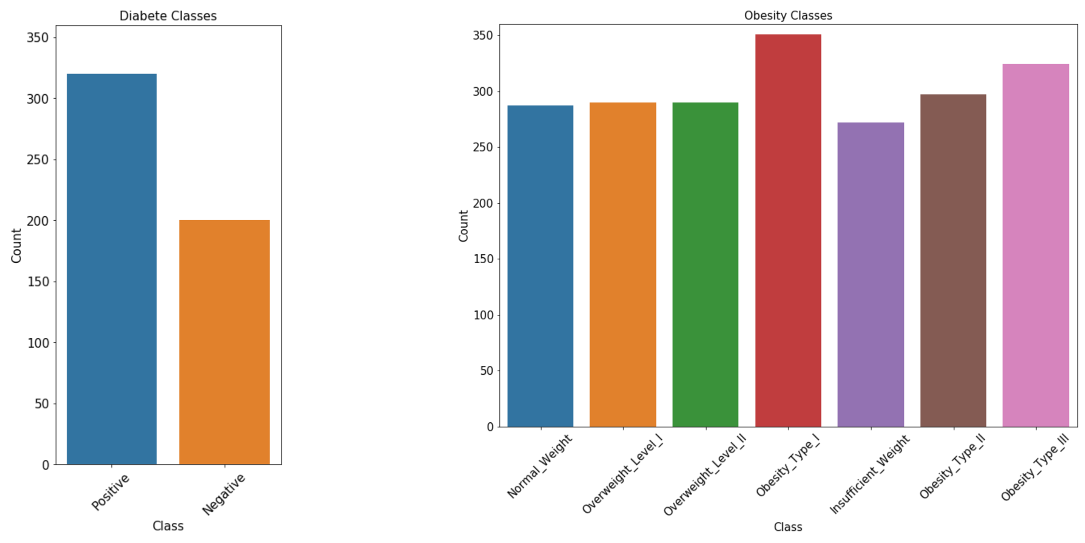


Figure 1. Class distributions for datasets 1 and 2.

After encode the 2 datasets, the data was randomly split into two parts, 80% of the data was assigned as training data, including training set and validation set if the cross validation is used. The other 20% of the data was assigned as test data, which was only used to evaluation the model after the training process is done.

3. Decision tree results and analysis

The first model I use is a decision tree classifier. A decision tree is a tree structure in which each internal node represents a judgment on an attribute, each branch represents the output of a judgment result, and finally each leaf node represents a classification result.

Decision tree generation algorithms include ID3, C4.5 and CART, etc. Here, I use the Decision Tree Classifier in scikit-learn package, which is a kind of CART algorithm that proposed by Breiman et al. in 1984. The CART algorithm assumes that the decision tree is a binary tree, in the generation stage, the Gini index is used to select the most characteristic feature, and at the same time determine the optimal binary cut point of the feature. The Gini index is defined as:

$$Gini(p)\sum_{k=1}^{K} p_k(1-p_k) = 1-\sum_{k=1}^{K} p_k^2$$

where p_k is the probability of k class.

In order to avoid overfitting (while the tree is too large or deep), the CART pruning algorithm subtracts some subtrees from the bottom of the 'fully grown' decision tree until pruning to the root node. Then, test the subtree sequences on an independent verification dataset by cross-validation, and select the best subtree. The loss function of the subtree is defined as:

$$C_\alpha(T) = C(T) + \alpha|T|$$

Where T is the subtree, $C(T)$ is the prediction error of training data, $|T|$ is the number of the nodes, α is a positive value that represents the complexity parameter. Larger α obtains smaller tree.

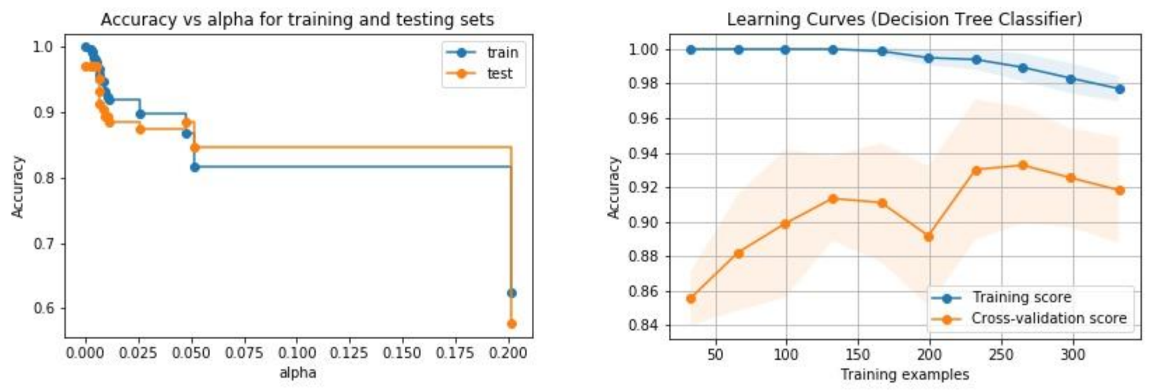


Figure 2, Accuracy of decision tree classifier v.s. model complexity (left) and training data size (right) for dataset 1.

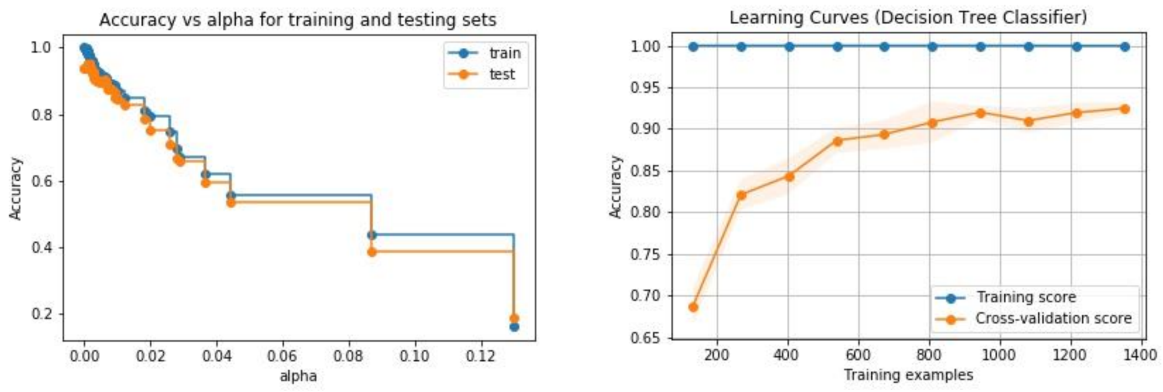
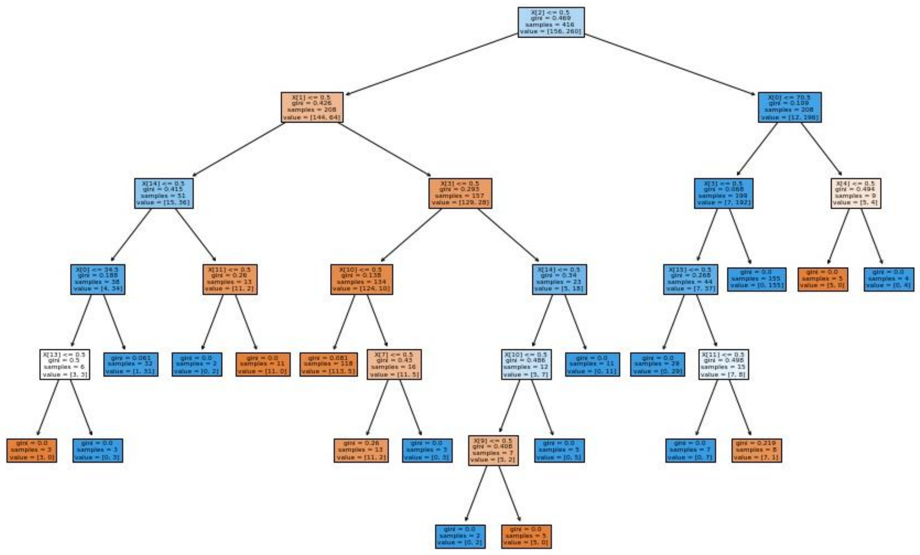


Figure 3, Accuracy of decision tree classifier v.s. model complexity (left) and training data size (right) for dataset 2.

Here, Figure 2 and Figure 3 shows the results of the two datasets. Here, a 5-fold cross validation method is used. As we can see in the two left figures, for diabetes dataset, the accuracy on both train and validation datasets are declined with α increases. It means when the decision tree gets bigger, the accuracy increase and does not show overfitting on this dataset. This is due to the small size of the data. On the obesity dataset, the accuracy of test dataset first increase then decrease with increasing of α, while the accuracy of training set keeps decreasing. This shows the overfitting effect on the obesity dataset, which has a larger size. After this test, In order to avoid overfitting and obtain higher accuracy, I choose the largest α when the validation accuracy gets highest. For diabetes data, $\alpha = 0.00493097$. For obesity data, $\alpha = 0.00058358$, according to Occam's razor principle. Figure-DTree shows the decision tree for the diabetes dataset.



X[2] <= 0.5 gini = 0.469 samples = 416 value = [156, 260]
X[1] <= 0.5 gini = 0.426 samples = 208 value = [144, 64]
X[0] <= 70.5 gini = 0.109 samples = 208 value = [12, 196]
X[14] <= 0.5 gini = 0.415 samples = 51 value = [15, 36]
X[3] <= 0.5 gini = 0.293 samples = 157 value = [129, 28]
X[3] <= 0.5 gini = 0.068 samples = 199 value = [7, 192]
X[4] <= 0.5 gini = 0.494 samples = 9 value = [5, 4]
X[0] <= 34.5 gini = 0.188 samples = 38 value = [4, 34]
X[11] <= 0.5 gini = 0.26 samples = 13 value = [11, 2]
X[10] <= 0.5 gini = 0.135 samples = 134 value = [124, 10]
X[14] <= 0.5 gini = 0.34 samples = 23 value = [5, 18]
X[15] <= 0.5 gini = 0.268 samples = 44 value = [7, 37]
gini = 0.0 samples = 155 value = [0, 155]
gini = 0.0 samples = 5 value = [5, 0]
gini = 0.0 samples = 4 value = [0, 4]
X[13] <= 0.5 gini = 0.5 samples = 6 value = [3, 3]
gini = 0.061 samples = 32 value = [1, 31]
gini = 0.0 samples = 2 value = [0, 2]
gini = 0.0 samples = 11 value = [11, 0]
gini = 0.081 samples = 118 value = [113, 5]
X[7] <= 0.5 gini = 0.43 samples = 16 value = [11, 5]
X[10] <= 0.5 gini = 0.486 samples = 12 value = [5, 7]
gini = 0.0 samples = 11 value = [0, 11]
gini = 0.0 samples = 29 value = [0, 29]
X[11] <= 0.5 gini = 0.498 samples = 15 value = [7, 8]
gini = 0.0 samples = 3 value = [3, 0]
gini = 0.0 samples = 3 value = [0, 3]
gini = 0.26 samples = 13 value = [11, 2]
gini = 0.0 samples = 3 value = [0, 3]
X[9] <= 0.5 gini = 0.408 samples = 7 value = [5, 2]
gini = 0.0 samples = 5 value = [0, 5]
gini = 0.0 samples = 7 value = [0, 7]
gini = 0.219 samples = 8 value = [7, 1]
gini = 0.0 samples = 2 value = [0, 2]
gini = 0.0 samples = 5 value = [5, 0]

Figure-DTree, The decision tree build for diabetes dataset.

I also test the effect of data size. The training process uses the 10% to 100% of the training data with the parameter I have chosen. Right figures in Figure 2 and Figure 3 shows the result. As we can see, the accuracy of test dataset are generally increase with the growth of size of the data. For diabetes data, the test accuracy shows a large standard deviation, this is probably due to the small size of data. For Obesity data, the accuracy score on training set is keep 100%, while the test accuracy grows from approximate 68% to 92%, the model performs well when the training size reach to 1000 instances on the obesity dataset. The overall results shows the importance of data size for decision tree classification.

4. Neural networks results and analysis

The neural network is based on artificial neurons, and each neuron is connected to each other to complete signal transmission, reception and processing. In the implementation of ANN, the signals transmitted between artificial neurons are real numbers, and each neuron corresponds to a threshold. When the total signal is higher than the threshold, the signal calculation is completed with the aid of the excitation function. Using multilayer neural networks can construct very complex estimation functions with simple neurons.

The activation function of the neuron affects the presentation ability of the whole neural network. For example, if a non-linear function is not applied to each neuron, the neural network will also be a linear function, so it is not more powerful than a single neuron. Another important thing that affects the presentation ability of the network is the structure complexity. In general, the more complex network structures can represent more complex functions, thereby realizing complex applications.

In this assignment, I use Multi-layer Perceptron classifier in scikit-learn package to implement the neural network algorithm. I choose two different activation function, ReLU and tanh, with 1 to 10 hidden layers. Each hidden layer has 10 neurons. The ReLU activation function is a linear function while tanh is non-linear. I choose adam solver to optimize the loss function, and the learning rate is 0.0001. For this algorithm, the validation fraction is 0.2.

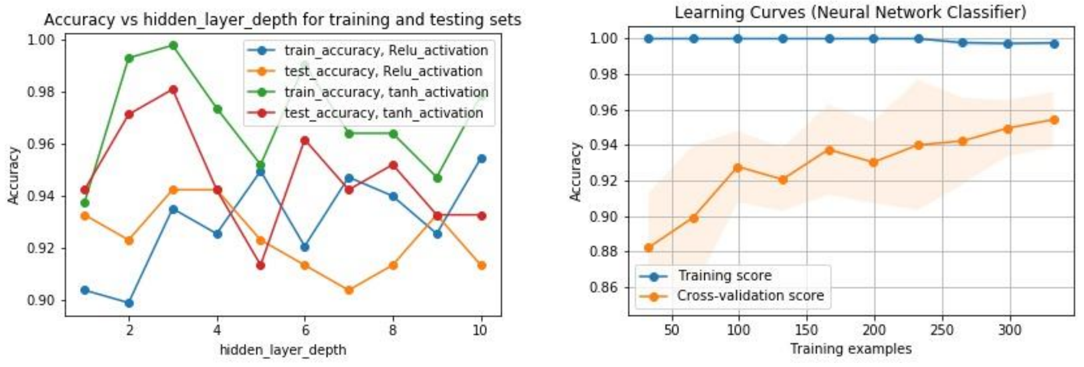


Figure 4, Accuracy of neural networks classifier v.s. model complexity (left) and training data size (right) for dataset 1.

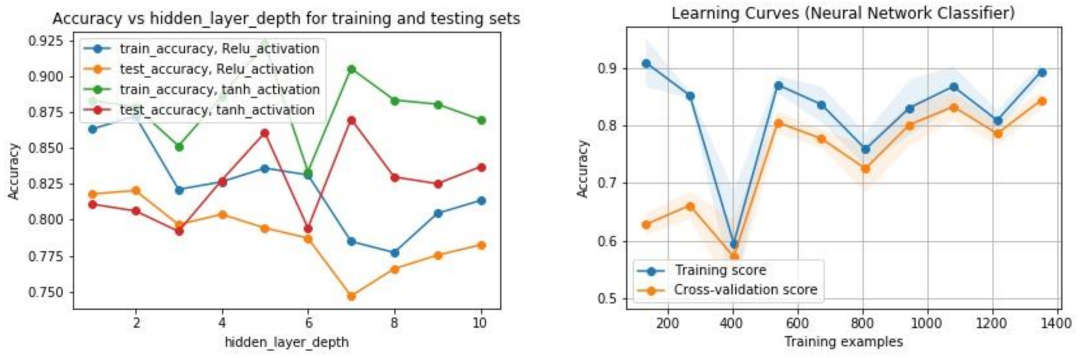


Figure 5, Accuracy of neural networks classifier v.s. model complexity (left) and training data size (right) for dataset 2.

Figure 4 and Figure 5 shows the results of neural network classifier on the two datasets. The left two figure compare the accuracy with hidden layer depth for the two activation functions. As we can see. For both datasets, the tanh activation function works better than the ReLU function. This is due to the limitation of the representation ability of the linearity function. Another finding is that theoretically complex networks have stronger expressive power, but in practical, more hidden layers did not obtain higher accuracy here. This is because, first, a feedforward network with a hidden layer in theory has the ability to approximate any continuous function. Second, training a network with multiple hidden layers requires complex learning algorithms to prevent it from falling into a divergent state.

I also compared the accuracy versus the training size of the data. According to the previous results, I use 3-hidden-layer network with tanh activation function on diabetes dataset, and 5-hidden-layer network with tanh function on obesity dataset. For the diabetes set, the test accuracy increases from 88% to 95% with the growth of data size, this means the network works well for this dataset, we can improve the result with adding more instances. For the obesity data, the accuracy of the training dataset is around 90%, which means the model has not converge yet. This is because the model is to complex to converge before reach the iteration limit, which I set to 2000 here.

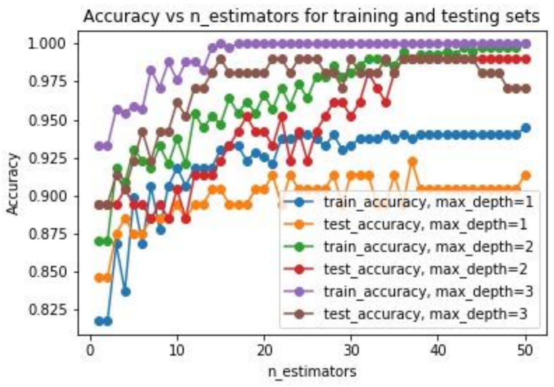
Overall, the neural network algorithm has strong expression ability and equation approximation ability, we can use deep neural networks to implement very complex applications. On the other hand, neural network is difficult to converge to a stable state if it’s too complex. Besides the network structure and neuron function, there are still a lot of parameters we can tune, such as the learning rate, the optimization method, etc. We should balance among the computational cost and the efficiency, and carefully tune the parameters when apply.

5. Boosting results and analysis

Boosting is a method that can promote a weak classifier to a strong learner. Generally speaking, a weak learner can only get slightly better results than random guessing, while a strong learner can be very close to the optimal learner. The boosting method trains a series of classifiers serially, so that the samples that the previous base classifier did wrong receive more attention in the follow-up, and combine these classifiers to obtain a strong classifier with perfect performance.

In this assignment, the boosting algorithm is implemented with the AdaBoost Classifier in scikit-learn package. In order to implement the multi-class classifier, the algorithm of the boosting is SAMME, which minimizes the multi-class index loss function. The base estimator is the decision tree classifier that I use in part 3. I tried three different depths of the decision tree as the the base estimator, and compared the accuracy with different number of estimators.

Figure 6 and Figure 7 shows the results of boosting decision trees with two datasets. The left two figures show the accuracy change with the increase of number estimators. For both of the two datasets, the AdaBoosting of decision tree with max depth of three achieved the best overall performance. This tells us that the boosting with better base estimators performs better than the weaker base learner even the boosting algorithm can improve the performance. For the diabetes dataset, the boosting can increase the accuracy before the number of estimators reach 15, when with max depth 3 decision trees. For the obesity dataset, the accuracy keeps increase until 50 estimators. We can also observed that for the decision tree with max depth 1, the boosting works not so well and oscillates on the obesity dataset, this probably because the dataset contains some noise data.



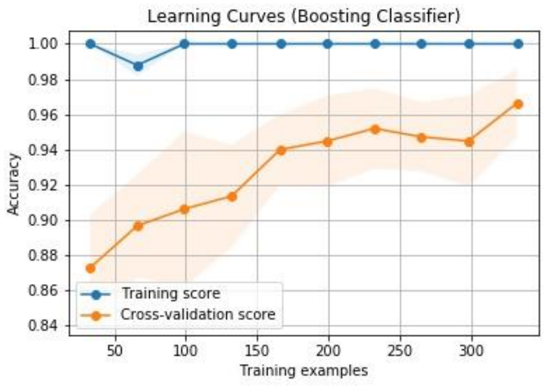


Figure 6, Accuracy of boosting classifier v.s. model complexity (left) and training data size (right) for dataset 1.

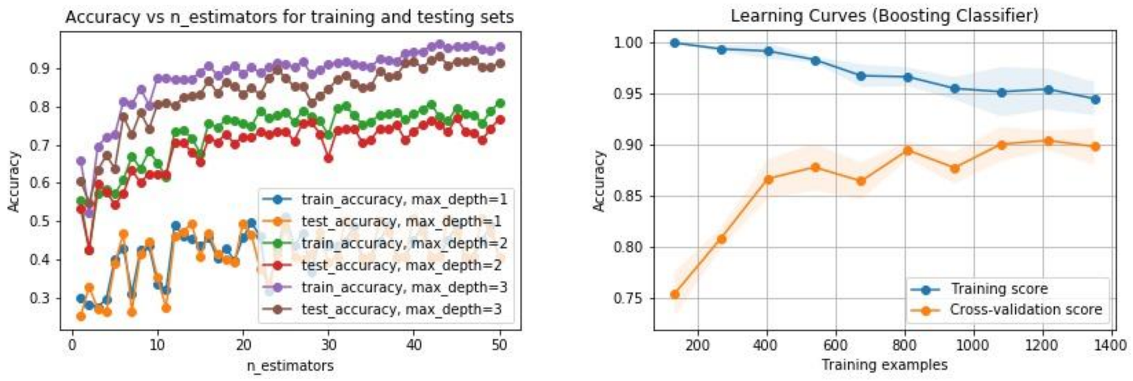


Figure 7, Accuracy of boosting classifier v.s. model complexity (left) and training data size (right) for dataset 2.

I also compare the accuracy versus the dataset size. For diabetes dataset, I chose the number of estimators to be 20, while for the obesity dataset is 40. The max depth of the base decision tree learner is 3. Both of the datasets shows the increase of the test accuracy when the size increase. For diabetes dataset, we can add more data to improve the performance. For the obesity dataset, the accuracy increase smoothly after the size of training samples reach 800.

In general, the boosting algorithm combines weak learners into strong learners. Theoretically, the boosting method does not appear over fitting. However, the AdaBoost algorithm is designed for clean data sets. In practical applications, the data is often noisy. The AdaBoost algorithm is very sensitive to noise in the application. The sensitivity of AdaBoost to noise comes from the exponential loss function. According to the existing tags, if an instance is not correctly classified, its weight will be increased. In this way, when the labels of the training samples contain noise, AdaBoost will still try to fit these noises, thereby reducing the predictive ability of the classifier.

6. Support vector machines results and analysis

Support vector machine is a large interval classifier used to solve two classification problems. It tries to find the hyperplane with the largest interval to distinguish different types of samples. Among them, the interval is defined as the distance from samples of different categories to the classification hyperplane. For nonlinear classification problems, support vector machines use kernel function to extend linear support vector machines to nonlinear support vector machines.

In this assignment, I implement the support vector machine classifier with the C-Support Vector Classification in scikit-learn package. I compared three different kernel functions: linear function, polynomial function, and radial basis function. I also chose 6 different regularization parameters from 10^-3 to 10^2 to add a squared L2 penalty to avoid overfitting.

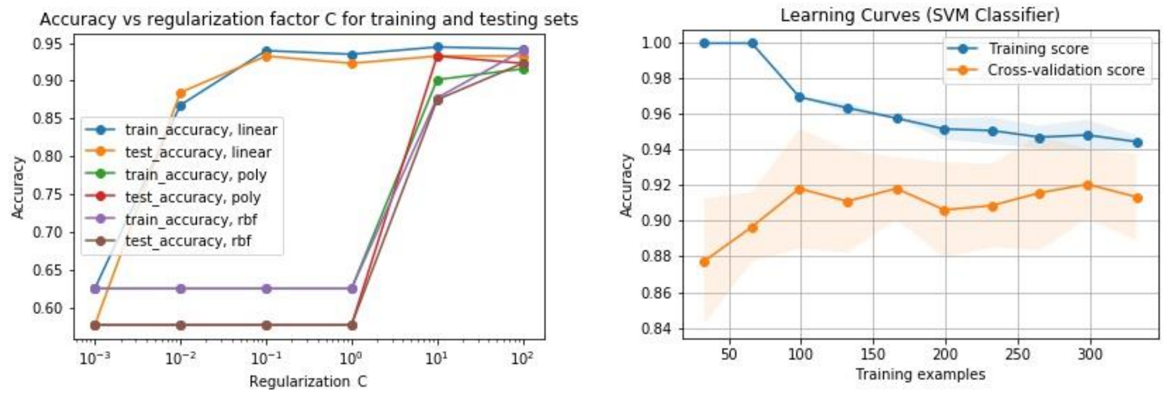


Figure 8, Accuracy of support vector machines classifier v.s. model complexity (left) and training data size (right) for dataset 1.

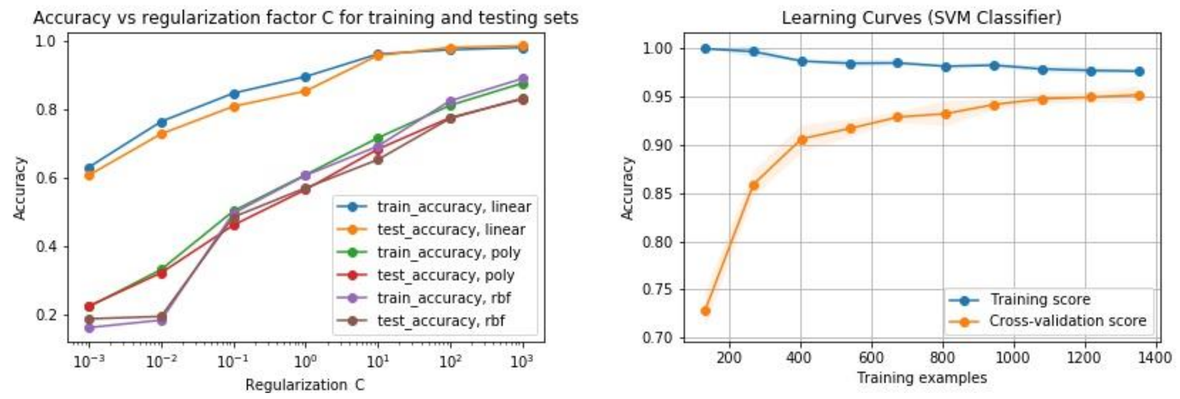


Figure 9, Accuracy of support vector machines classifier v.s. model complexity (left) and training data size (right) for dataset 2.

Figure 8 and Figure 9 shows the results of the support vector machines. As we can see in the left two figures, for both of the two datasets, the linear kernel function works better than the polynomial function and radial basis function. This is probably because the problem with two datasets are more like linear problem. Also, the accuracy of train and validation data are increase when we use a larger regularization parameter.

I also compared the impact of dataset size on results accuracy. Here I use linear kernels for both datasets, and the regularization parameter for diabetes dataset is 10 and for obesity dataset is 100. As we can see in the right two figures, for the diabetes dataset, the accuracy increases quickly until the size reach 100. After this, the accuracy has no obvious increase while increasing training data size. The similar phenomenon appears in the obesity dataset, the accuracy increases rapidly until the size of the training instances reach 400. After this, the accuracy increases smoothly. This shows that the support vector machines can work well with even a small set of data. This is great when we must face the lack of data some time.

In general, the support vector machines use the kernel function to make the estimation function non-linear, it has the ability to learn non-linearity on a small amount of data. For the linear problem, the linear kernel function performs better.

7. k-nearest neighbors results and analysis

The basic assumption of the k-nearest neighbor algorithm is that similar samples in the input space should be similar in the output space. The algorithm has no explicit training process, only needs to store all training samples. It is a lazy learning method. During the test, for the test sample x, the k-nearest neighbor algorithm searches the training sample for k-nearest neighbors to the test sample. For the classification task, the test sample is classified into the category with the highest vote among the k nearest neighbor samples.

K value selection, distance measurement, classification decision rules, are the three basic elements of the k nearest neighbor algorithm. In this assignment, I compare the different k values from 1 to 50, and compared the Euclidean distance and Manhattan distance measurement. For the classification decision rules, I chose kd-tree with uniform weight. The kd-tree is a binary tree, which represents a division of the k-dimensional space, each node of which corresponds to a super rectangular area in the k-dimensional space.

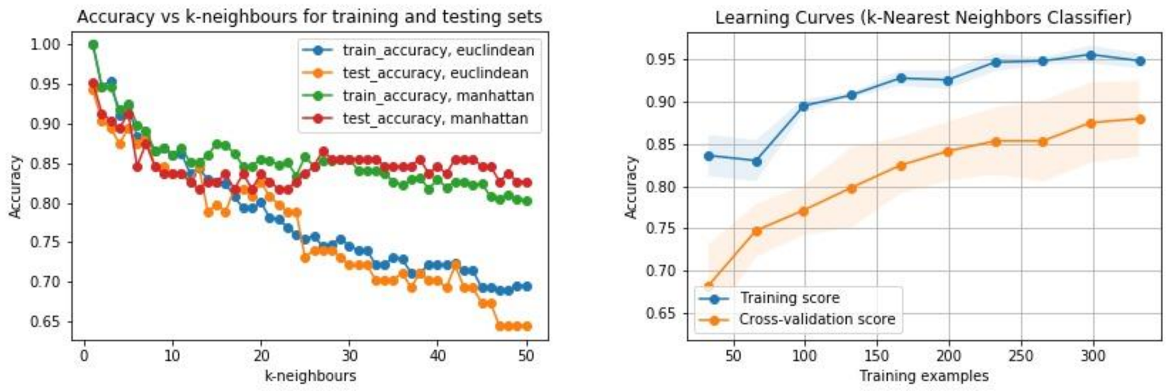


Figure 10, Accuracy of k-nearest-neighbors classifier v.s. model complexity (left) and training data size (right) for dataset 1.

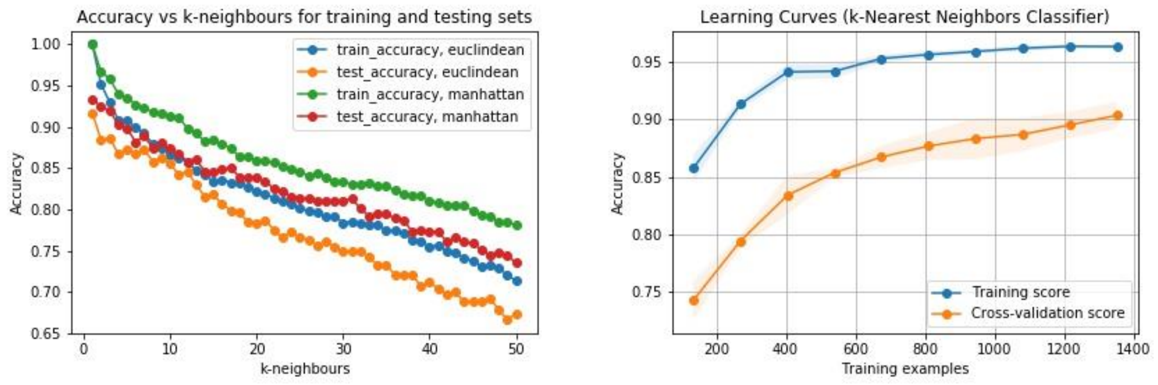


Figure 11, Accuracy of k-nearest-neighbors classifier v.s. model complexity (left) and training data size (right) for dataset 2.

Figure 10 and Figure 11 shows the results of k-nearest neighbor algorithm. The left two figures show the accuracy of the validation dataset. As we can see, the accuracy of test data for both datasets decrease with the increase of k values. This means, the size of the two dataset is not large enough to show the overfitting effect during the decreasing of k value from 50 to 1. We can improve the results with adding more data to the training set while using k-nearest neighbor algorithm for these two data. We can also find that the Manhattan distance works better than the Euclidean distance on the two datasets.

I also studied the effect of data size on results. For both datasets, I use Manhattan distance and k value is 2. The right two figures show the results. As we can see, the accuracy for the test set increase while the size grow. This tells us the k-nearest neighbor algorithm needs large dataset if we want a good prediction.

The k-nearest neighbor algorithm is simple and intuitive. When the K value, distance measurement, classification decision rules is determined, the result is uniquely determined. One thing I noticed during training and testing is that the training speed of the k-nearest neighbor algorithm is quite rapid, but the query speed is slow. This is because when the training set is large (the number of features or the number of samples is large), the prediction process cost much time to query tree and calculate the prediction results.

8. Conclusions

In this assignment, I use five different supervised learning algorithms, decision trees, neural networks, boosting, support vector machines, to classify two datasets, diabetes dataset and obesity dataset. The different algorithms have different effectiveness and performance.

The boosting algorithm obtains the maximum test accuracy (97%) on diabetes dataset, and the support vector machines obtains the maximum test accuracy (95%) on obesity dataset. The main problems that I need to face are small size of data and overfitting.

Reference

Islam, MM Faniqul, et al. 'Likelihood prediction of diabetes at early stage using data mining techniques.' Computer Vision and Machine Intelligence in Medical Image Analysis. Springer, Singapore, 2020. 113-125.

Palechor, F. M., & de la Hoz Manotas, A. (2019). Dataset for estimation of obesity levels based on eating habits and physical condition in individuals from Colombia, Peru and Mexico. Data in Brief, 104344.

Codesource:

All of the code that use in the assignment is available on my dropbox. Here is the downloading link below. In case of improper transmission, I set a security code, which is my GT account: cpeng78

https://www.dropbox.com/sh/76vv9akhzravakk/AAANxYp0MhmEg11Jwk0Av59Ea?dl=0