Preprint typeset using LaTeX style AASTeX6 v. 1.0

SUPERVISED LEARNING

Shanshan Jiang
Geogia Institute of Technology

1. INTRODUCTION

One of the classifications of machine learning is to divide machine learning into supervised learning and unsupervised learning. Alpaydin (2009)And my article is about supervised learning. Supervised learning is a method of machine learning that can be used to learn or build a model of functions and learning models, and to speculate on new instances in this mode. The training data consists of the input object and the expected output. For example, you want your model to be able to identify the bike from a given set of data. Training is done by providing a set of inputs and outputs that help the system understand the basic functions of defining bicycles. These characteristics can include saddles, wheels, alloys, etc. Now, when the new data is entered into the model, the model can recognize the bike. Each output provided by the model, as well as new data that facilitates the output, becomes a new input-output combination that is fed into the model as training data for learning. Thus, The task of a supervised learner after observing some of the training examples (input and expected output) predict the output of this function to any possible input values. To achieve this, learners must generalize from the existing material in a "reasonable" way. And the output of a function can be a continuous value (Regression) or a categorical label (Classification). So Supervised learning is divided into regression and classification. In my article, I am going to talk about Classification in five different Algorithms.

2. ALGORITHMS

2.1. *Decision Trees*

In machine learning, a decision tree is a decision support tool that uses a tree or decision model and its possible consequences, including opportunity event results, resource costs, and utility.Alpaydin (2009) For example, Time for decision trees, you are making plans for the weekend and find that your good friends may come to town. You want to make arrangements, but some unknown factors will determine what you can and cannot do. A decision tree is a predictive model that represents a mapping between object properties and object values. Usually, the decision trees can be used in both classification trees and regression trees.

In the classification setting, it has several ways to measure the quality of a split. $\hat{\pi}_c = \frac{1}{|D|}\sum_{i\in\mathcal{C}}\mathbb{I}(y_i = c)$(D is the data in the leaf), It is estimating the class-conditional probilities to the data in the leaf, and it is fitting multinoulli model and satisfying the test $X_j < t$(Murphy 2012). According to this, several common error measure can be evaluating a proposed partition as follow:

- Misclassification rate. For here, the most probable class label as $\hat{y}_c = \text{argmax}_c\hat{\pi}_c$, and it has corresponding error rate is:$\frac{1}{|D|}\sum_{i\in D}\mathbb{I}(y_i \neq \hat{y}) = 1 - \hat{\pi}_{\hat{y}}$
- Entropy, Deviance: $\mathbb{H}(\hat{\pi}) = -\sum_{c=1}^{C}\hat{\pi}_c \log\hat{\pi}_c$Minimizing the entropy is equivalent to maximizing the information gain:infoGain$(X_j < t, Y) = (-\sum_c p\,(y = c)\log p\,(y = c)) + (\sum_c p\,(y = c \mid X_j < t)\log p\,(c \mid X_j < t))$
- Gini index: It is the expected error rate. $\hat{\pi}_c$is the probability a random entry belongs to class c in the leaf, and $(1 - \hat{\pi}_c)$is the probility that would be misclassified: $\sum_{c=1}^{C}\hat{\pi}_c(1 - \hat{\pi}_c) = \sum_c\hat{\pi}_c - \sum_c\hat{\pi}_c^2 = 1 - \sum_c\hat{\pi}_c^2$

2.2. *Neural networks*

Artificial Neural Network (ANN) is a computational system which is fuzzily inspired by the biological neural network constituting animal brains.In a common ANN implementation, the signal at the junction between the artificial neurons is real, and the output of each artificial neuron is computed by some nonlinear function of its input. Murphy (2012)The

<sjiang086@gatech.edu>

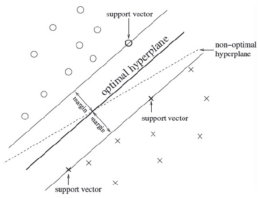

Figure 1. Support Vector Mahcine Model

link between the artificial neurons is called the "edge". Artificial neurons and edges often have weights that are adjusted as learning progresses. Weight increases or decreases the strength of the signal at the junction. Artificial neurons can have thresholds that send signals only when the aggregated signal exceeds the threshold. Usually, artificial neurons accumulate into layers.

Different layers can perform different types of conversions on their inputs. The signal may propagate from the first layer (input layer) to the last layer (the output layer) after iterating through the layers multiple times.We can computer layer l+1's activations a$^{(l+1)}as$:$z^{(l+1)} = W^{(l)}a^{(l)} + b^{(l)}$, $a = f(z^{(l+1)})$

2.3. *K nearest neighbors*

The K nearest neighbors algorithm is the simplest of all machine learning algorithms. KNN could used in many application besides classification, for example, it could be used to synthesize new samples as shown in Zhang et al. (2016). In the KNN classification, the output is class membership. An object is classified by a majority vote of its neighbors, where the object is assigned to the most common class of its k nearest neighbors (k is a positive integer, usually a small integer). If k = 1, the object is simply assigned to the nearest single neighbor class. That is, where ther ith nearest neighbors is assigned a weight W_{ni} with $\sum_{i=1}^{n} w_{ni} = 1$

2.4. *Boosting*

Boosting is a machine learning set meta-algorithm used primarily to reduce bias, as well as to monitor variance in learning, and a series of machine learning algorithms that convert weak learners into strong learners. it is traning the next learner on the mistakes of the previous leaners, and trying to genrate complementary base-learners. (Quinlan 1986)For all base-learners j=1,...,L; Given x, calculate d$_j(x), j = 1, ..., L; Calculate$ class outputs, i=1,...,K: $y_i = \sum_{j=1} L(log\frac{1}{\beta_j})d_{ji}(x)$

2.5. *Support Vector Machine*

In the case of support vector machines, there is a data point viewed as a p-dimensional vector. And the learners try to separate such points with a (p-1)-dimensional hyperplane, which is called a linear classifier. The best hyperplane is the one to represent the largest separation(margin) between two categories, as shown in Figure 1.

3. EVALUATION

3.1. *Benchmark*

- **Datasets:** I have two dataset, *Letter Image Recognition Data* and *Sloan Digital Sky Survey.*

1. **Letter Imagine Recognition Data**: It designs to recognize a large number of black and white rectangular pixel displays as one of the 26 uppercase letters in the English alphabet. The character image is based on 20 different fonts, each of which is randomly deformed in each of the 20 fonts, producing 20,000 uniquely stimulating files. Each stimuli is converted to 16 primitive numeric attributes (Statistical moment and Edge count) and then scaled to accommodate an integer value range from 0 to 15. We usually train the top 16000 items and then use the resulting model to predict the remaining 4000 of the letter categories.

2. **Sloan Digital Sky Survey**: It consists of 10,000 records of observations of space taken by the Sloan Digital Sky Survey. There are 17 feature columns and 1 target colum to indentify the observation to be a star, a galaxy or quasar for each observation.

For both letter and sky survey dataset, 2/3 of the data is used for training and remaining 1/3 is used for testing.

- **Metrics**: Severl metrics are computed for evaluation purpose including precision: $Precision = \frac{tp}{tp+fp}$, recall: $Recall = \frac{tp}{tp+fn}$, F1-score: $F1 = 2 * \frac{precision*recall}{precision+recall}$, and accuracy: Accuracy $Accuracy = \frac{tp+tn}{p+n}$;

K	Acc(Tr)	Acc(Tst)	Prec(Tr)	Prec(Tst)	Rec(Tr)	Rec(Tst)	F1(Tr)	F1(Tst)
1	0.9184	0.8997	0.8498	0.8184	0.9184	0.8997	0.8817	0.8556
3	0.9909	0.9897	0.9908	0.9897	0.9909	0.9897	0.9908	0.9896
5	0.9931	0.9882	0.9931	0.9881	0.9931	0.9882	0.9931	0.9881
7	0.9954	0.9885	0.9954	0.9884	0.9954	0.9885	0.9954	0.9885
9	0.9981	0.9864	0.9981	0.9863	0.9981	0.9864	0.9981	0.9863
11	0.9993	0.9870	0.9993	0.9870	0.9993	0.9870	0.9993	0.9870
13	1.0000	0.9861	1.0000	0.9860	1.0000	0.9861	1.0000	0.9860

Table 1. Evaluation results of tree depth pruning for sky survey dataset.

3.2. *Decision Tree Experiments*

In this section, an experiment testing different magnitudes of decision trees pruning is conducted.

3.2.1. *Experiment One – Pruning*

In this experiment, I will evaluate experiment results of different levels of pruning. The pruning in my code (scikit-learn) is achieved by setting different depth of trees that are going to be generated. Usually, people do Pruning to prevent overfitting for Decision Trees algrithoms. If the reduction in errors is not sufficient to prove the additional complexity of adding an extra subtree, people can stop growing the trees.My code will be using first colum to do pruning test, which is the Depth. I change the depth for each, then get the data of Accuracy, Precision, Recall and F1-score.

Results of two datasets are discussed respectively.

- **Sky survey**

My purpose is to use SkySurvey data to test the results of pruning. I change the depth range 1-13 to do pruning test. In Table1, what I can get is a result of the training always getting better, but the result of testing gets better gradually and then goes down again, this is because the data is too complicated at the beginning, so that a short tree (simple) is not able to split data clearly enough. But as the tree growing taller, the tree will split the data using more features at different node. This is why the training results are always getting better as the tree grows. However, overfitting may happen when tree grows to some certain level. Because when three is very deep, for some nodes that are very close to leaf node, the threshold of each split is computed using only a few samlpes left in those nodes, which will possibly lead to an overfiting dureing testing.

So, from Table1, results of pruning are presented in a bottom to up order in the table. As you can see, pruning really helps preventing overfitting, as K become smaller when $K = 5,7$, testing results got better.

- **Letter**

My purpose is to use Letter data to test the results of pruning. I change the depth range 1-19 to do pruning test. In Table 2, I also get a very similar result: the result of testing gets better gradually and then goes down again, even if the result of training always got better, which is because the data is too complicated at the beginning. I did the pruning operation to avoid overfitting then the results got better, but when I continued pruning, the data became too simple, and the results got worse.

3.2.2. *Experiment Two – Different Portion of Training Data*

One of way of evaluating a machine learning algorithm is to check its inference performance by training the same model using different portion of training data. This test will also help us understand for the underlying test data and chosen model, whether we need more training data or the current model is already saturated to existing training data. So for all classification algorithm in this report, I run this test for all of them. To run the test, I fix the test sets in all tests which are 1/3 of the original datasets. Then I used remaining 2/3 data as the training data. Then 10 training processes are conducted each of which will use a gradually added portion of training data. The trained model will be evaluated on the same testing set.

As can be see in Table 3.2.2, For both datasets, decision trees have already achieved very high performance in F1-score of testing in the beginning when there is only small portion of training data . That means decision tree has a very good property to generality. As more training data being added, decision trees are able to use more data and

K	Acc(Tr)	Acc(Tst)	Prec(Tr)	Prec(Tst)	Rec(Tr)	Rec(Tst)	F1(Tr)	F1(Tst)
1	0.0719	0.0677	0.0061	0.0055	0.0719	0.0677	0.0111	0.0101
3	0.2366	0.2414	0.0830	0.0876	0.2366	0.2414	0.1198	0.1251
5	0.5028	0.5039	0.5161	0.5176	0.5028	0.5039	0.4911	0.4916
7	0.6766	0.6709	0.7138	0.7010	0.6766	0.6709	0.6850	0.6775
9	0.7977	0.7752	0.8133	0.7908	0.7977	0.7752	0.8001	0.7779
11	0.9027	0.8432	0.9054	0.8474	0.9027	0.8432	0.9034	0.8441
13	0.9663	0.8691	0.9666	0.8712	0.9663	0.8691	0.9664	0.8696

Table 2. Evaluation results of tree depth pruning for letter dataset.

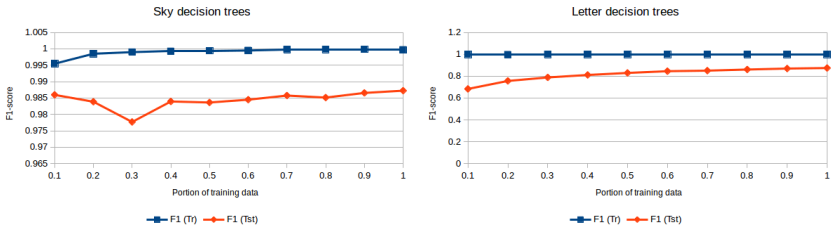

Figure 2. Experiment of using different portion of training data in decision trees on two datasets.

more feature to compute a more accurate threshold of features during node splitting, that explains why the F1-score of both datasets are getting better along with the use of more data.

3.3. *Neural Networks Experiments*

Neural network has been widely used in many different areas Zhang et al. (2015). It becomes even popular these days due to the blossom of deep learning research. The simpliest form of neural network is a multi-layer perceptron (MLP). MLP is compose of many layers in which contains a number of neurons. As can be seen ni Equation **??** and Equation **??**, Each neuron will apply a calculation using weight parameters associated to the neuron to the incoming vectors. Many factors can affect the neuron's performance which in turn will affect MLP's final performance. To test neural network on my two datasets, I looked into three factors of a neural network which are configuration of layers and number of neurons, regularization term and activation function. Exerpiemtn of tuning these three parameters and configurations are show in the following subsections.

3.3.1. *Experiment One – Network Layers*

Figure different configurations set up in this experiment starting with a MLP containing only a single layer up to a MLP containing five layers. In this experiment I fixed activation function to use Relu activation and I set regularization parameter alpha to 0.0001.

- **Sky survey**

In the Table 3, my expectation in the beginning is to see results will get better while more and more layers being used in a MLP. However, it got worse since the forth layer. I did some research and got that it is the vanishing gradient problem. As shown in He et al. (2016), the problem is that in some cases the gradient will be very small, effectively preventing a change in weight. In the worst case, this may completely stop the neural network from further training. As an example of the cause of the problem, the traditional activation function, such as the hyperbolic tangent function, has a gradient in the range (0,1), and the reverse propagation calculates the gradient through the chain rule.

- **Letter**

In the Table4, by looking at F1-score, the result of training start getting better until four layers configuration is used. It also comes with a phenomenon that is the vanishing gradient problem. The problem is that in some cases the gradient will be very small, effectively preventing a change in weight. In the worst case, this may completely stop the neural network from further training. As an example of the cause of the problem, the traditional activation function, such as the hyperbolic tangent function, has a gradient in the range (0,1), and the reverse propagation calculates the gradient through the chain rule.

	Acc(Tr)	Acc(Tst)	Prec(Tr)	Prec(Tr)	Rec(Tr)	Rec(Tst)	F1(Tr)	F1(Tst)
l1-128	0.7079	0.6997	0.6672	0.6522	0.7079	0.6997	0.6805	0.6669
l1-128-l2-64	0.6804	0.6739	0.7532	0.7375	0.6804	0.6739	0.6592	0.6485
l1-128-l2-64-l3-32	0.7987	0.7864	0.7499	0.7261	0.7987	0.7864	0.7650	0.7457
l1-128-l2-64-l3-32-l4-16	0.7431	0.7376	0.7702	0.7613	0.7431	0.7376	0.7245	0.7150
l1-128-l2-64-l3-32-l4-16-l5-8	0.4457	0.4415	0.5578	0.5545	0.4457	0.4415	0.3153	0.3170

Table 3. Evaluation results of different layer setups for sky survey dataset.

	Acc(Tr)	Acc(Tst)	Prec(Tr)	Prec(Tst)	Rec(Tr)	Rec(Tst)	F1(Tr)	F1(Tst)
l1-128	0.9455	0.9273	0.9463	0.9282	0.9455	0.9273	0.9456	0.9273
l1-128-l2-64	0.9520	0.9320	0.9536	0.9338	0.9520	0.9320	0.9523	0.9322
l1-128-l2-64-l3-32	0.9704	0.9333	0.9708	0.9348	0.9704	0.9333	0.9704	0.9334
l1-128-l2-64-l3-32-l4-16	0.9469	0.9185	0.9493	0.9217	0.9469	0.9185	0.9473	0.9190
l1-128-l2-64-l3-32-l4-16-l5-8	0.4457	0.4415	0.5578	0.5545	0.4457	0.4415	0.3153	0.3170

Table 4. Evaluation results of different layer setups for letter dataset.

	Acc (Tr)	Acc(Tst)	Prec(Tr)	Prec(Tr)	Rec(Tr)	Rec(Tst)	F1(Tr)	F1(Tst)
Reg-0.0001	0.7987	0.7864	0.7499	0.7261	0.7987	0.7864	0.7650	0.7457
Reg-0.001	0.8012	0.7879	0.7559	0.7308	0.8012	0.7879	0.7673	0.7470
Reg-0.01	0.8018	0.7885	0.7562	0.7312	0.8018	0.7885	0.7679	0.7476
Reg-0.1	0.4187	0.4082	0.1753	0.1666	0.4187	0.4082	0.2471	0.2366

Table 5. Evaluation results of different regularization for sky survey dataset.

3.3.2. *Experiment Two – Regularization*

Its purpose is to observe their results by training and to test two sets of data to use Regularization. In computer science, especially in the field of machine learning problem, regularization is the process of introducing additional information to address ill-posed issues or prevent overfitting. In scikit-learn, MLP uses a L_2regularization which is as known as ridge regression and also known as weight decay. Weight decay will cause values of weight parameters gradually being reduced to zero. Getting some weights value close to zeros has some advantages. One of advantages is that since some weights are close to zero, we can consider that the resulting MLP model becomes simpler since we will have less connection between neurons by having zero values for weights connecting neurons. Model's generality actually will get improved in this case.

- **Sky survey**

In the Table5, the greater the value of regularization, the faster the weight decay, which means getting better. But the result is slowly getting worse, because after controlling overfitting, the model becomes more and more simple, so the result becomes worse.

- **Letter**

In the Table6 , by looking at F1, the result of training get better gradually as the result of testing.

3.3.3. *Experiment Three – Activation Function*

In an artificial neural network, the activation function of a node defines the output of the node in the case of a given input or input set. A standard computer chip circuit can be considered an active digital network, which can be "on" (1) or "off" (0), depending on the input. It is similar to the behavior of a linear perceptron in a neural network. However, only non-linear activation functions allow such networks to use only a few nodes to compute nontrivial problems.

- **Sky survey**

	Acc (Tr)	Acc(Tst)	Prec(Tr)	Prec(Tst)	Rec(Tr)	Rec(Tst)	F1(Tr)	F1(Tst)
Reg-0.0001	0.9195	0.8855	0.9227	0.8891	0.9195	0.8855	0.9200	0.8859
Reg-0.001	0.9543	0.9289	0.9549	0.9306	0.9543	0.9289	0.9542	0.9290
Reg-0.01	0.9668	0.9348	0.9672	0.9362	0.9668	0.9348	0.9668	0.9348
Reg-0.1	0.9751	0.9386	0.9755	0.9404	0.9751	0.9386	0.9751	0.9386

Table 6. Evaluation results of different regularization for letter dataset.

	Acc (Tr)	Acc(Tst)	Prec(Tr)	Prec(Tr)	Rec(Tr)	Rec(Tst)	F1(Tr)	F1(Tst)
Identity	0.6434	0.6258	0.6238	0.5932	0.6434	0.6258	0.6125	0.5899
Relu	0.7596	0.7521	0.7771	0.7619	0.7596	0.7521	0.7499	0.7369
Logistic	0.8021	0.7897	0.7558	0.7316	0.8021	0.7897	0.7682	0.7488
Tanh	0.8012	0.7879	0.7559	0.7308	0.8012	0.7879	0.7673	0.7470

Table 7. Evaluation results of different activation function for sky survey dataset.

	Acc (Tr)	Acc(Tst)	Prec(Tr)	Prec(Tst)	Rec(Tr)	Rec(Tst)	F1(Tr)	F1(Tst)
Identity	0.9681	0.9305	0.9693	0.9333	0.9681	0.9305	0.9682	0.9307
Relu	0.7581	0.7592	0.7611	0.7624	0.7581	0.7592	0.7575	0.7583
Logistic	0.9196	0.9033	0.9220	0.9055	0.9196	0.9033	0.9199	0.9033
Tanh	0.9904	0.9559	0.9905	0.9568	0.9904	0.9559	0.9905	0.9560

Table 8. Evaluation results of different activation function for letter dataset.

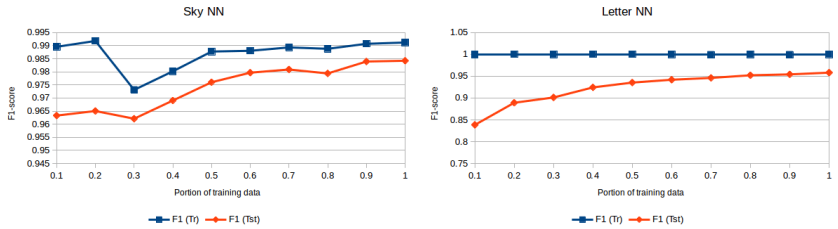

Figure 3. Experiment of using different portion of training data in neural network on two datasets.

From the Table 7, I used SkyServey data to do Traning and testing. The result of F1-score is gradually increased in both Traning and testing. That is because of the Identity is the Linear function: A=cx, where activation is proportinal to input.Tanh is a scaled sigmoid function and nonlinear in nature. Its gradient is stronger than sigmoid. Relu function A(x)=max(0,x),the computational cost of Relu is lower than that of the Tanh and the sigmoid because it involves simpler mathematical operations.

- **Letter**

From Table 8, I made use of Letter data set to get the result of F1-score. For here, Relu has the least data of result.

3.3.4. *Experiment Four – Different Portion of Training Data*

3-layer configuration of MLP is used in this test for both datasets. The performance of a MLP is usually related to amount of data used in training process. Because a MLP usually contains much more parameters than other 4 models evaluated in this report. So, the more complex the MLP is , the more data it will need to gaurantee a good performance. As expected, it can be observer from Figure 3.3.4, the F1-score of testing will go up as more data is used for training. For Sky dataset, performance of both traininga and testing go down a bit when 30% of training data is used, I think this may because that some noisy data get added to the training set.

3.4. *K nearest neighbors experiments*

KNN classifier is the most special algorithm among five classifiers of this project. The specialty of the KNN lays in the factor that the KNN algorithm does not learn any discriminative or generative function from the training

K	Acc(Tr)	Acc(Tst)	Prec(Tr)	Prec(Tst)	Rec(Tr)	Rec(Tst)	F1(Tr)	F1(Tst)
5	0.8306	0.7806	0.8254	0.7544	0.8306	0.7806	0.8111	0.7508
10	0.8169	0.7882	0.8130	0.7844	0.8169	0.7882	0.7892	0.7553
15	0.8143	0.7955	0.8022	0.8010	0.8143	0.7955	0.7830	0.7579
20	0.8143	0.7924	0.7988	0.7733	0.8143	0.7924	0.7834	0.7540
25	0.8137	0.7955	0.7989	0.7726	0.8137	0.7955	0.7828	0.7575
30	0.8130	0.7955	0.8053	0.7909	0.8130	0.7955	0.7798	0.7555

Table 9. Evaluation results of different activation function for sky survey dataset.

K	Acc(Tr)	Acc(Tst)	Prec(Tr)	Prec(Tst)	Rec(Tr)	Rec(Tst)	F1(Tr)	F1(Tst)
5	0.9719	0.9453	0.9721	0.9463	0.9719	0.9453	0.9719	0.9454
10	0.9578	0.9371	0.9584	0.9393	0.9578	0.9371	0.9578	0.9372
15	0.9473	0.9250	0.9483	0.9272	0.9473	0.9250	0.9475	0.9251
20	0.9346	0.9189	0.9362	0.9218	0.9346	0.9189	0.9349	0.9192
25	0.9281	0.9117	0.9303	0.9151	0.9281	0.9117	0.9285	0.9121
30	0.9191	0.9011	0.9219	0.9052	0.9191	0.9011	0.9196	0.9016

Table 10. Evaluation results of different activation function for letter dataset.

data but just remember the training dataset instead. KNN thus is also known as an algoirthm with lazy learning. KNN could have many different variant such as the one with delegates, weight adjusted KNN, adaptive KNN, and so on. In this section, I only conduct experiments to explore the properties of the most basic KNN algorithm. I used Euclidean distance as the distance metric when finding the nearest neighbore. Euclidean distance is very sensitive to unnormalized data. Therefore all data is normalized in this experiment before running KNN.

3.4.1. *Experiment One – Number of Neighbors*

- **Sky survey**

In KNN classification algorithm K stands for number of neighbors to look at when doing classification, different value used for K in my experiments are shown in Table9. By looking at results of F1-score, the overall trend of training result is gradually declining. But the results of the testing were not consistent with the results of the training. That's because when the number of K is approaching 1, there is an 'overfitting' situation. The result of the training is too accurate so that the KNN trys to compromise the decision boundary to each sample near the boundary of two sets during training, so it may lead a bad result of the testing. is not the best. And when k=30, the result of training is the worst, but the test result is not necessarily the worst. KNN's performance on sky dataset is much worse than other four classifiers. This could because the data distribution in feature space is very complex that by using Euclidean distance metric, KNN failed to find true actual nearest neighbors of each sample. Thus, KNN will result in many inaccurate classification results.

- **Letter**

In Table 10, I am using Letter data set, and also setting K as the number of neighbors. By looking at the results of F1-score, the overall trend of training result is gradually declining as the results of the testing. That's because when the number of K is approaching 1, the result is going to be more accurate. It also produces a good result of testing.

3.4.2. *Experiment Two – Different Portion of Training Data*

As can be seen from Figure 3.4.2, for sky dataset, KNN almost have the same performance for all experiments. Adding more training data does not help improve KNN's performance. In think this could because in original feature space of sky dataset where the KNN is executed, the distribution of data has no clear decison boundary. In other words, different class of data in sky dataset may be mixed up together so that the amound of training data in this case does not help.

3.5. *Boosting experiments*

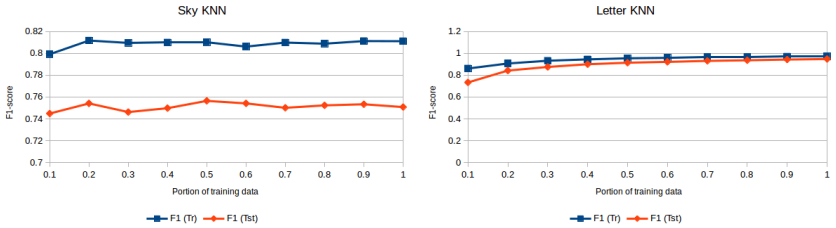

Figure 4. Experiment of using different portion of training data in KNN on two datasets.

N	Acc(Tr)	Acc(Tst)	Prec(Tr)	Prec(Tst)	Rec(Tr)	Rec(Tst)	F1(Tr)	F1(Tst)
20	0.9969	0.9836	0.9969	0.9838	0.9969	0.9836	0.9969	0.9837
40	1.0000	0.9873	1.0000	0.9872	1.0000	0.9873	1.0000	0.9872
60	1.0000	0.9909	1.0000	0.9909	1.0000	0.9909	1.0000	0.9908
80	1.0000	0.9894	1.0000	0.9894	1.0000	0.9894	1.0000	0.9893
100	1.0000	0.9906	1.0000	0.9906	1.0000	0.9906	1.0000	0.9905

Table 11. Evaluation results of different number of base classifier for sky survey dataset.

N	Acc(Tr)	Acc(Tst)	Prec(Tr)	Prec(Tst)	Rec(Tr)	Rec(Tst)	F1(Tr)	F1(Tst)
20	0.4937	0.4917	0.6155	0.6051	0.4937	0.4917	0.5056	0.5003
40	0.5364	0.5317	0.6128	0.6075	0.5364	0.5317	0.5345	0.5301
60	0.6203	0.6167	0.6581	0.6540	0.6203	0.6167	0.6209	0.6178
80	0.6354	0.6314	0.6686	0.6664	0.6354	0.6314	0.6362	0.6314
100	0.6716	0.6608	0.6965	0.6895	0.6716	0.6608	0.6730	0.6623

Table 12. Evaluation results of different number of base classifier for letter dataset.

Different from other classifier in this project, boosting is the only algorithm contains more than one classifier. To my understanding, there are two reasons leads to boosting's superior performance. First boosting is doing a hard negative mining during training. By setting different weight to different samples, boosing will increase the probablity of getting training for samples that are misclassified in the previous classifier. Thus, the classifier will be trained more on these hard cases. Second, each weak classifier will have a high variance on its performance. However, as long as we can make sure each weak classifier could have a performance better than random guess, by using numerous weak classifiers, by average, the system will result in a high performance at the end.

3.5.1. *Experiment One – Number of Base Classifiers*

I did the first evaluation by changing the number of the base classifier.

- **Sky survey**

In the Table 11, The first column N as the number of base classifiers.The first column N as the number of base classifiers. Usually, the result of both F1(Traning) and F1(Testing) should be gradually better, then becoming stable. By looking at the result of both F1(Traning) and F1(Testing), I have first F(Training) as 0.9969, and F1(Testing) as 0.9837. Then the data doesn't change anymore, which met my prediction.

- **Letter**

In the Table 12, By looking at the result of both F1(Traning) and F1(Testing), the result of F1(Traning) does not stop changing, and the result is getting better, which means we can still change the N's number to let the F1(Traning) becoming stable.

3.5.2. *Experiment Two – Pruning of Base Classifiers*

I did the second evaluation by doing pruning.

- **Sky survey**

Depth	Acc(Tr)	Acc(Tst)	Prec(Tr)	Prec(Tst)	Rec(Tr)	Rec(Tst)	F1(Tr)	F1(Tst)
1	0.9613	0.9645	0.9619	0.9649	0.9613	0.9645	0.9614	0.9646
3	1.0000	0.9906	1.0000	0.9906	1.0000	0.9906	1.0000	0.9905
5	1.0000	0.9906	1.0000	0.9906	1.0000	0.9906	1.0000	0.9905
7	1.0000	0.9897	1.0000	0.9897	1.0000	0.9897	1.0000	0.9896
9	1.0000	0.9894	1.0000	0.9894	1.0000	0.9894	1.0000	0.9893
11	1.0000	0.9915	1.0000	0.9915	1.0000	0.9915	1.0000	0.9914

Table 13. Evaluation results of different depth base classifier for sky survey dataset.

Depth	Acc(Tr)	Acc(Tst)	Prec(Tr)	Prec(Tst)	Rec(Tr)	Rec(Tst)	F1(Tr)	F1(Tst)
1	0.4254	0.4127	0.4926	0.4859	0.4254	0.4127	0.4128	0.4005
3	0.6716	0.6608	0.6965	0.6895	0.6716	0.6608	0.6730	0.6623
5	0.8696	0.8348	0.8760	0.8443	0.8696	0.8348	0.8709	0.8367
7	0.9865	0.9274	0.9867	0.9303	0.9865	0.9274	0.9865	0.9280
9	0.9994	0.9517	0.9994	0.9533	0.9994	0.9517	0.9994	0.9520
11	1.0000	0.9658	1.0000	0.9665	1.0000	0.9658	1.0000	0.9659
13	1.0000	0.9682	1.0000	0.9689	1.0000	0.9682	1.0000	0.9683

Table 14. Evaluation results of different depth base classifier for letter dataset.

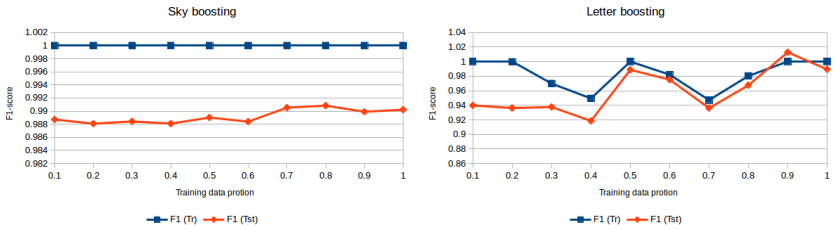

Figure 5. Experiment of using different portion of training data in boosting on two datasets.

In the Table 13, the first column Depth as the pruning of base classifiers. Usually, the result of both F1(Traning) and F1(Testing) should be gradually better, then becoming stable. By looking at the result of both F1(Traning) and F1(Testing), I have first F(Training) equals 0.9614, and F1(Testing) equals 0.9646. Then the result of F1(Training) always equals 1.0000. By here, we can see the difference between F1(Traning) and F1(testing) is getting bigger, which means overfitting.

- **Letter**

By looking at the result of both F1(Traning) and F1(Testing), the result of F1(Traning) becomes stable until Depth=1.0000. But overfitting still happens because the difference between F1(Traning) and F1(testing) is getting bigger.

3.5.3. *Experiment Three – Different Portion of Training Data*

Because boosting actually conducts a hard negative mining during trianing, it is not that sensitive to the amound of training data. In fact, as long as existing data contains samples that are hard to be classified, boosting algorithm can learn a reasonable decision boundary using these samples and results in a performance which is no bad. Results of two datasets in this experiment can be found in Figure 3.5.3

3.6. *Support vector machine experiments*

Support vector machine classifys underlying data using a boundary and margin supported by support vectors. One of advantage of using support vector machine is that the complexity of optimizing SVM is not depending on the dimension of training data, instead, the complexity is $O(n^3)$in terms of the number of training data n. Also, since the

C	Acc(Tr)	Acc(Tst)	Prec(Tr)	Prec(Tst)	Rec(Tr)	Rec(Tst)	F1(Tr)	F1(Tst)
0.0001	0.8452	0.8394	0.8667	0.8648	0.8452	0.8394	0.8415	0.8351
0.0010	0.8591	0.8685	0.8837	0.8850	0.8690	0.8685	0.8674	0.8670
0.0100	0.9221	0.9227	0.9253	0.9260	0.9221	0.9227	0.9219	0.9226
0.1000	0.9596	0.9506	0.9598	0.9509	0.9596	0.9506	0.9595	0.9506
1.0000	0.9748	0.8970	0.9750	0.9013	0.9748	0.8970	0.9748	0.8954

Table 15. Evaluation results of different parameter C for sky survey dataset.

C	Acc(Tr)	Acc(Tst)	Prec(Tr)	Prec(Tst)	Rec(Tr)	Rec(Tst)	F1(Tr)	F1(Tst)
0.0001	0.2687	0.2511	0.3500	0.3340	0.2687	0.2511	0.2218	0.2092
0.0010	0.6981	0.7015	0.7125	0.7163	0.6981	0.7015	0.6943	0.6959
0.0100	0.8512	0.8455	0.8583	0.8538	0.8512	0.8455	0.8526	0.8461
0.1000	0.9701	0.9545	0.9707	0.9558	0.9701	0.9545	0.9702	0.9545
1.0000	0.9999	0.9374	0.9999	0.9504	0.9999	0.9374	0.9999	0.9407

Table 16. Evaluation results of different parameter C for letter survey dataset.

boundary and margin is only defined by a few support vectors, which make using SVM in classification problem very efficient.

One of difficulty I faced when using SVM in this project is to choosing correct parameters. I looked in to two parameters in my experiments. One is parameter C which is a slack variable and work as a regularization term in objective functiona. Another parameter is parameter Gamma, which controls variance of RBF kernel.

I started with a default setting of these two parameters and ended up with very low performance of SVM on my two datasets. I then did some research online and figure out that I have to run grid search to search for a combination of these two parameters. The best result I obtained in my experiment is when I set $C = 0.1$and $Gamma = 0.1$.

There are three experiments being conducted in this section. The first experiment will fix the parameter $Gamma = 0.1$ and I will tune parameter C only. In the second experiment, I fixed $C = 0.1$ and change $Gamma$ to be different values. The third experiment will test the influence of using different kernel functions.

3.6.1. *Experiment One – Parameter C*

The parameter C tells the SVM how much we want to avoid misclassifying each training example. For small values of C, a hyperplane is with the largest minimum margin, and For large values of C, a hyperplane correctly separates as many instances as possible, but it is also a sign of overfitting in some cases. In this experiment, I use different value for parameter C in SVM. The value starts with 0.0001and I increase its balue by 10 times for an experiment in the next.

- **Sky survey**

In the Table 15, I change the value of C to evaluate the Sky survey dataset. By looking at the result of both F1(Tr) and F1(Tst), The F1 traning is gradully getting better, becuase when parameter C gets larger, the generated boundary becomes more bented to be compromised to each samples near the boundary. Thus it could be seen from Table 15 that the F1-score of training data is getting better along using bigger C. However, since using bigger C will force SVM algorithm to fit every detail of samples near the boundary, the algorithm is possibly overfit the training data, which is actually the case in my experiment with Sky survey dataset. So, this explains why the test F1-score goes down when $C = 1$.

- **Letter**

In the Table 16, the same situation happened to letter dataset when we change the value of parameter C to be larger. F1-score of training process gradually getting close to 1, however, the SVM overfitted to training data while using large C, that's why it could be observed from the Table that, the F1-score of testing lower down a bit when $C = 1$.

3.6.2. *Experiment Two – Parameter Gamma*

Gamma	Acc(Tr)	Acc(Tst)	Prec(Tr)	Prec(Tst)	Rec(Tr)	Rec(Tst)	F1(Tr)	F1(Tst)
0.0001	0.8452	0.8394	0.8667	0.8648	0.8452	0.8394	0.3358	0.8351
0.0010	0.8690	0.8685	0.8837	0.8850	0.8690	0.8685	0.8674	0.8674
0.0100	0.9221	0.9227	0.9253	0.9260	0.9221	0.9227	0.9219	0.9226
0.1000	0.9596	0.9506	0.9598	0.9509	0.9596	0.9506	0.9595	0.9506
1.0000	0.9748	0.8970	0.9750	0.9013	0.9748	0.8970	0.9748	0.8954

Table 17. Evaluation results of different parameter gamma for sky survey dataset.

Gamma	Acc(Tr)	Acc(Tst)	Prec(Tr)	Prec(Tst)	Rec(Tr)	Rec(Tst)	F1(Tr)	F1(Tst)
0.0001	0.2687	0.2511	0.3500	0.3340	0.2687	0.2511	0.2218	0.2092
0.0010	0.6981	0.7015	0.7125	0.7163	0.6981	0.7015	0.6943	0.6959
0.0100	0.8512	0.8455	0.8583	0.8538	0.8512	0.8455	0.8526	0.8461
0.1000	0.9701	0.9545	0.9707	0.9558	0.9701	0.9545	0.9702	0.9545
1.0000	0.9999	0.9374	0.9999	0.9504	0.9999	0.9374	0.9999	0.9407

Table 18. Evaluation results of different parameter gamma for letter survey dataset.

In this section, I will use Parameter Gamma to do evaluation.Gamma are the parameters for a nonlinear support vector machine (SVM) with a Gaussian radial basis function kernel. A small gamma means a Gaussian with a large variance; A large gamma leads to high bias and low variance models, and vice-versa.

- **Sky survey**

In the Table 17, as its data increase, its results gradually get better. For the testing result of the data, in the beginning, it also gradually increase. But the last one dropped abruptly, which is because when Gamma is big enough, the overfitting happenned.

- **Letter**

In the Table 18, both results of training and testing are gradually getting better, which met my prediction.

3.6.3. *Experiment Three – Parameter Kernel*

SVM without using kernel trick will only able to work well on linearly separable datasets. But in most cases, the data is not linearly separable in it original space. A trick could be done in this case is to transform data to a space with higher dimension in which the distribution of data could be possibly separable. Using kernel will achieve this goal implicitylyin a more efficient way. I am testing three most popular kernel used by people in SVM which are radius based function kernel, linear kernel and polynomial kernel of degree 3.

- **Sky survey**

In the Table 19, traning result and testing are very similar. The linear got the best result. I think this is probably because the distribution of the datasets in higher dimension is linear separable. I actually expect RBF kernel achieves the best results. But since RBF kernel will depends on searching good value for parameter *Gamma*. I have tried many values for *Gamma* though.

- **Letter**

In the Table 20, the results on letter dataset makes more sense to me in which RBF kernel performs the best. I think this may because that the data distribution of letter dataset is more complex than sky survey dataset so, by just transforming data to slightly higher dimension space, both linear and polynomial kernel still couldn't split the data very well. However, since RGB kernel will transform data to infinit high dimension space, it is more likely to separate data in that space using hyperplane.

3.6.4. *Experiment Four – Different Portion of Training Data*

Like before, I conducted another experiment to test how performance of SVM is affected by using different amount of training data. It could be seen from Figure 3.6.4 that for both dataset, the F1-scores between training and testing process are highly correlated that they almost have the same trend. Both training and testing performances are getting better and better along increasing the portion of training data.

Kernel	Acc(Tr)	Acc(Tst)	Prec(Tr)	Prec(Tst)	Rec(Tr)	Rec(Tst)	F1(Tr)	F1(Tst)
rbf	0.9560	0.9512	0.9563	0.9516	0.9560	0.9512	0.9559	0.9512
linear	0.9830	0.9785	0.9830	0.9786	0.9830	0.9785	0.9830	0.9785
Poly-3	0.9131	0.9082	0.9192	0.9152	0.9131	0.9082	0.9127	0.9077

Table 19. Evaluation results of different kernel for sky survey dataset.

Kernel	Acc(Tr)	Acc(Tst)	Prec(Tr)	Prec(Tst)	Rec(Tr)	Rec(Tst)	F1(Tr)	F1(Tst)
rbf	0.9451	0.9329	0.9470	0.9358	0.9451	0.9329	0.9454	0.9330
linear	0.8708	0.8535	0.8724	0.8562	0.8708	0.8535	0.8709	0.8535
Poly-3	0.8537	0.8398	0.9096	0.8923	0.8537	0.8398	0.8709	0.8548

Table 20. Evaluation results of different kernel for sky survey dataset.

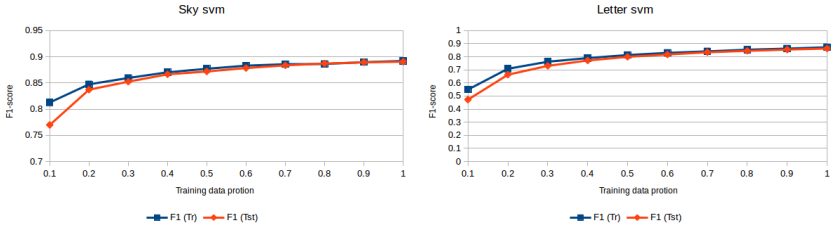

Figure 6. Experiment of using different portion of training data in SVM on two datasets.

4. CONCLUSIONS

According to my evaluation, decision trees is one of the most efficient algorithms. Its advantage is that performance is guaranteed as long as there is enough data, and even non-numerical data can be used. The disadvantage is that it requires a lot of data, and it needs to be balanced. At the same time, a large amount of data is required for doing pruning.

Neural Network can change different parameters. Each had good results. And the algorithm can do a lot of computation. It can be used in a vast field. But there is always a bit of tricky between its layers and nodes.

In my opinion, K nearest neighbor is the most straightforward algorithm of these algorithms. It is very sensitive to the distribution of data and picking the 'neighbors,' so the whole algorithm runs smoothly. But if the data gets large, it becomes very slow.

Boosting is the fastest algorithm and it gets the best results.

Support vector machine has the most average efficiency. I can do the operation by only a small number of Support vector. But adjusting its hyperparameter is a tough part, once the correct hyperparameter is found, the result will be very good.

REFERENCES

Ethem Alpaydin. *Introduction to Machine Learning (Adaptive Computation and Machine Learning series)*. The MIT Press, 2009.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning)*. The MIT Press, 2012. ISBN 0262018020.

J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

Xi Zhang, Yanwei Fu, Shanshan Jiang, Leonid Sigal, and Gady Agam. Learning from synthetic data using a stacked multichannel autoencoder. In *Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on*, pages 461–464. IEEE, 2015.

Xi Zhang, Di Ma, Lin Gan, Shanshan Jiang, and Gady Agam. Cgmos: Certainty guided minority oversampling. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1623–1631. ACM, 2016.