Assignment 2

Tutors: (list all your tutors)
Group members: (list all your group members with name, ID, and Unikey)
Dehong Liang: 470188761
Tianzuo Zhang: 470085460
Shanshan Luo: 470349580

Abstract

In recent years, machine learning has become more and more popular. It has a remarkable effect in some fields, e.g. image processing, object recognition and prediction trends etc. In this experiment, our duty is classifying the item into certain category from the dataset of "Fashion-MNIST", which consisting training set of grayscale image with a label from 10 classes. We choose Random forest, adaptive boost, support vector machine to implement our experiment, Finally, we analyze the performance of the three algorithms and compare their results respectively.
Keywords: classification, Fashion-MNIST, Random forest, Adaptive boost, SVM

1. Introduction

Applying different algorithm to classify datasets in machine learning can gain different performance, which leads us to find the appropriate algorithm to deal with different targets classification. It's an interesting process because we are able to compare the different classic classification algorithms throughout analysis of the accuracy of output and the running time cost.

In our experiment, considering about the practical situation and the suitability, we finally choose the "Fashion-MNIST" dataset, it is a dataset of images—consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. So, we try to test different algorithms on this dataset by classifying the testing set after using the training set to gain a propriate classifier.

In last assignment, we chose logistic regression algorithm and KNN method in experiment. In this time, we want to try tree distinct algorithms in order to compare the various classification methods, at the same time, we can get a better understanding about the practical implementation of certain method we learned from lecture or research. We also chose two Algorithms that are not referred in lecture.

Firstly, a method that's similar to Logistic Regression comes to our mind, the support vector machine algorithm, But SVM has many differences like: the loss function of SVM is maximum distance and SVM is unable to produce probability but LR could. Secondly, A team member propose an algorithm, Random forest, which is simple and efficient. Simply put, the Random forest is an algorithm that brings together multiple decision trees by integrating the idea of learning, For a Random forest, its basic building block is the decision quite tree, Thirdly, we choose to use the Adaptive boosting algorithm, Its adaptation lie in that the weight of the sample that was misclassified by the previous basic classifier will increase, and the weight of the correctly classified sample will decrease, and it will be used again to train the next basic classifier. At the same time, in each iteration, a new weak classifier is added until a predetermined sufficiently small error rate is reached, or a pre-specified maximum number of iterations is reached to determine the final strong classifier.

This study is very important for us to have a good understanding of different kinds of algorithm and how to apply it to make classification in practical application process. By analyzing different algorithms, we can find the appropriate algorithm which is used with the result of high accuracy rate and less running time. What's more, our experiment can be used by others, which can provide a detailed description of basis classification for people and help them to have a better understanding of Random forest, SVM, Adaptive boosting algorithms.

2. Previous work

Before the experiment, we did many researches around this area including image preprocessing and classification in order to figure out the how the scholars use different methods to solve analogous problem. The

dataset is can be colorful images or gray scale images, we prefer to concentrate on the classification algorithm since our dataset is not that complicated. We want to get some inspiration from the research papers before experience.

2.1 First article

This first article implements the classification using a special way by combining the generic search and cluster method. In this article, the author proposes a GA based clustering technique, GCUK-clustering, which can automatically evolve the appropriate clustering of a data set. And they utilize this classifier to the image classification. [1]

Actually, generic algorithm belongs to a class of search techniques that mimic the principles of natural selection to develop solutions of large optimization problems. Generic algorithm operates by maintaining and manipulating a population of potential solutions called chromosomes. Each chromosome has an associated fitness value which is a qualitative measure of the goodness of the solution encoded in it. This fitness value is used to guide the stochastic selection of chromosomes which are then used to generate new candidate solutions through crossover and mutation.

Clustering is a well-known exploratory data analysis tool where the objective is to partition the data into a number of clusters. It's an unsupervised pattern classification technique which partitions the input space into K regions based on some similarity or dissimilarity metric. The number of clusters may or may not be known a priori. Let the input space S be represented by n points $\{x_1, x_2, \dots, x_n\}$, and the K clusters be represented by $\{C_1, C_2, \dots, C_K\}$. Then

$$C_i \neq \emptyset \; For \; i = 1, \dots K$$

$$C_i \cap C_j \neq \emptyset \; For \; i = 1, \dots, K; j = 1, \dots, K \; and \; i \neq j$$

The genetic clustering technique is subsequently referred to as the genetic clustering for unknown K (GCUK-clustering), where K denotes the number of clusters.

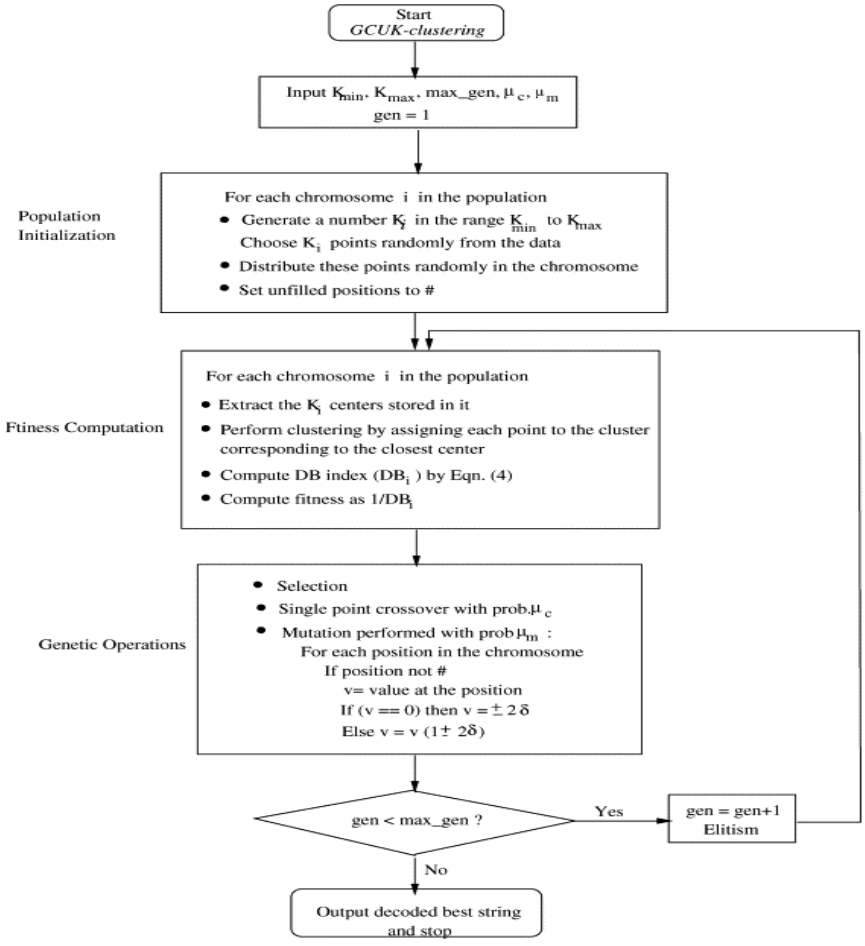


Figure 1: flowchart of the method

2.2 Second article

Another article introduces a new algorithm random forest. The objective of this article is image classification – classifying an image by the object category that it contains. The advantage of such classifiers, over multi-way SVM for example, is the ease of training and testing. [2]

In order to process the dataset easily, an image is represented using the spatial pyramid scheme proposed by Lazebnik et al, which is based on spatial pyramid matching, but here applied to both appearance and shape. They follow the approach of Bosch et al. SIFT descriptors are computed at points on a regular grid with spacing M pixels.

Figure 2: Appearance and shape spatial representation

At each grid point the descriptors are computed over four circular support patches with different radii, consequently each point is represented by four SIFT descriptors. Meanwhile, Local shape is represented by a histogram of edge orientations gradients within an image sub region quantized into K bins. Next is to image matching. The similarity between a pair of images I and J is computed using a kernel function between their PHOG (or PHOW) histogram descriptors D_I and D_J, with appropriate weightings for each level of the pyramid:

$$K(D_{I,}D_J) = exp\left\{\frac{1}{\beta}\sum_{l\epsilon l,} \alpha_l d_l(D_I, D_J)\right\}$$

We mainly focus on the classification problem, A random forest multi-way classifier consists of a number of trees, with each tree grown using some form of randomization. The leaf nodes of each tree are labeled by estimates of the posterior distribution over the image classes. Each internal node contains a test that best splits the space of data to be classified. An image is classified by sending it down every tree and aggregating the reached leaf distributions. Randomness can be injected at two points during training: in subsampling the training data so that each tree is grown using a different subset; and in selecting the node tests.
According to how the tree is growing. The trees here are binary and are constructed in a top-down manner. The binary test at each node can be chosen in one of two ways: first randomly, i.e. data independent; or by a greedy algorithm which picks the test that best separates the given training examples. “Best” here is measured by the information gain caused by partitioning the set Q of examples into two subsets Q_i according the given test.

$$\Delta E = -\sum_i \frac{|Q_i|}{|Q|} E(Q_i)$$

Suppose that T is the set of all trees, C is the set of all classes and L is the set of all leaves for a given tree. During the training stage the posterior probabilities for each class $c \in C$ at each leaf node $l \in L$, are found for each tree $t \in T$. These probabilities are calculated as the ratio of the number of images I of class c that reach l to the total number of images that reach l. $Y(I)$ is the class-label c for image I. In classification process, the test image is passed down each random tree until it reaches a leaf node. All the posterior probabilities are then averaged and the arg max is taken as the classification of the input image.
Particularly, this article adds Random Ferns Classifier to increase the speed of the random forest. But what’s the “Ferns”, Ferns are nonhierarchical structures where each one consists of a set of binary tests which in our case is 0 if $nT_x + b > 0$ or 1 if $nT_x + b \leq 0$. During training there are an ordered set of tests S applied to the whole training data set. This is in contrast to random forests where only the data that falls in a child is taken into account in the test. As in random forests “leaves” store the posterior probabilities. During testing the probability that an image belongs to anyone of the classes that have been learned during training is returned. The result of each test and the ordering on the set defines a binary code for accessing the “leaf” node.

2.3 Third article

The third article use Flexible, High Performance Convolutional Neural Networks for Image Classification, CNN is one of the neural network algorithms. Deep hierarchical neural models roughly mimick the nature of mammalian visual cortex. The most successful hierarchical object recognition systems all extract localized features from input images, convolving image patches with filters. Filter responses are then repeatedly sub-sampled and re-filtered, resulting in a deep feed-forward network architecture whose output feature vectors are eventually classified [3]
A convolutional layer is parametrized by the size and the number of the maps, kernel sizes, skipping factors, and the connection table. Each layer has M maps of equal size (M_x, M_y). A kernel (blue rectangle in Fig 1) of size (K_x, K_y) is shifted over the valid region of the input image (i.e. the kernel has to be completely inside the

image). The skipping factors (S_x, S_y)define how many pixels the filter/kernel skips in x- and y-direction between subsequent convolutions.

$$M_x^n = \frac{M_x^{n-1} - K_x^n}{S_x^n + 1} + 1$$
$$M_y^n = \frac{M_y^{n-1} - K_y^n}{S_y^n + 1} + 1 (1)$$

The biggest architectural difference between our implementation and the CNN of is the use of a max-pooling layer instead of a sub-sampling layer. No such layer is used by who simply skips nearby pixels prior to convolution, instead of pooling or averaging. found that max-pooling can lead to faster convergence, select superior invariant features, and improve generalization. A theoretical analysis of feature pooling in general and max-pooling in particular is given by. The output of the max-pooling layer is given by the maximum activation over nonoverlapping rectangular regions of size $((K_x, K_y)$.

Kernel sizes of convolutional filters and max-pooling rectangles as well as skipping factors are chosen such that either the output maps of the last convolutional layer are down sampled to 1 pixel per map, or a fully connected layer combines the outputs of the topmost convolutional layer into a 1D feature vector. The top layer is always fully connected, with one output unit per class label.

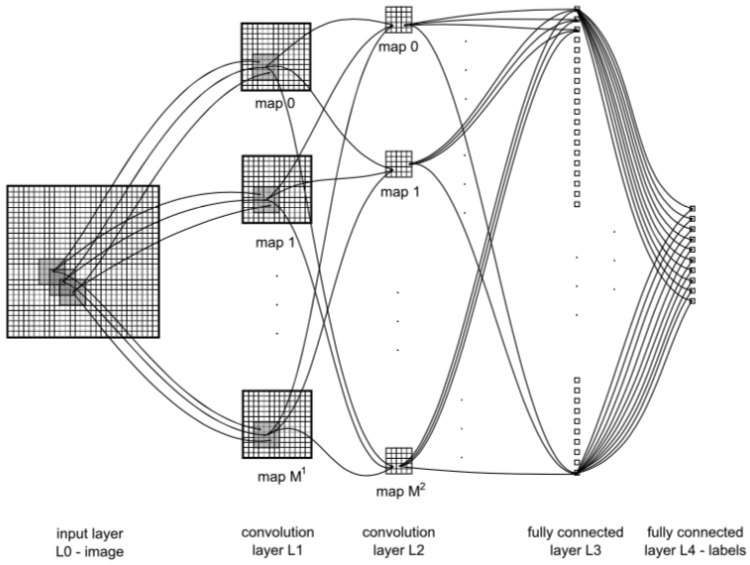


Figure 3: Architecture of a convolutional neural network with fully connected layers

Both outputs y and deltas δ of layer Ln are 2D stride. Their original size is (M_x, M_y), but they are horizontally strided with a pitch of 32 floats (we use this stride for all 2D data), resulting in coalesced memory accesses. The vertical stride avoids additional bounding tests in CUDA kernels. For BP and FP, analogous information about connections is needed. Therefore, they store backward connections in CBP. AW requires a list of all map connections, stored as an array of map index pairs. Dealing with biases in BP kernel requires to know where the weights of particular connections start; this information is stored in a 2D array WIDXBP of size (M_n, M_{n-1}).

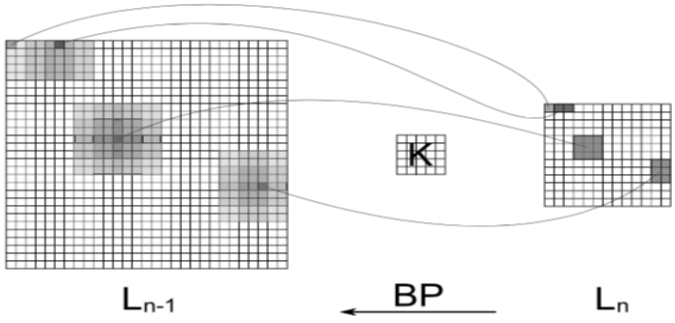


Figure 4: Back propagating deltas

The biggest advantage of CNN is in feature extraction. Since the feature detection layer of CNN learns through the training data, the feature extraction of the display is avoided, but the learning data is implicitly learned; and since the weights of the neurons on the same feature mapping surface are the same, the network can finish parallelized learning, this is also another major advantage of convolutional networks compared to the network of neurons that connected to each other. It is mainly used to identify two-dimensional graphics of displacement, scaling and other forms of distortion invariance.

3. Methods

We choose the dataset we want to use and three main method to analyze data during this experiment. General descriptions of those three methods are demonstrated below:

3.1 Random forest and decision tree

The first method we applied in our experiment is Random forest, which is a common and practical classification method in machine learning. The classification of random forest is relatively simple, and the classification effect on some problems is considerable. The theory of Random forest can be described in simple terms: by applying ensemble learning [4], random forest integrate several decision tree together to make the classification decision. so the basic component for random forest algorithm is decision tree [5], which will be described below for a better understanding of random forest.

3.1.1 Decision tree

Decision tree is a method of making decisions by using tree-shaped data structures [6]. It is an intuitive representation of knowledge and an efficient classifier, which relies on information theory to abstract complex classification decision problems into easy-to-understand and express judgments. The symbolic meaning of different component inside the tree is different. To be specific, each node inside the tree represents a test of a feature, the branch of the tree represents each test result of the feature, and each leaf node of the tree represents a category. The highest level of the tree is the root node.

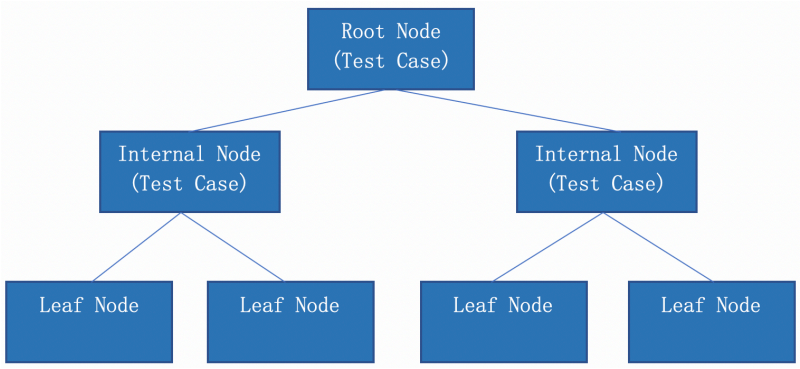


Figure 5: Decision tree

As mentioned above, the key element for classifying datasets in decision tree structure is the node, so focusing on optimal node is the core to construct a decision tree. For finding the optimal node to achieve classification, we need to find the samples with purity as high as possible, which can be measured by a special term—entropy [7].

$$H = -\sum_{i=1}^{n} P_i \log_2 P_i$$

And P_i is the corresponding probability of different sample types after passing the current node. $\log_2 P_i$ is used to punish the node with high entropy value or high degrees of confusion.

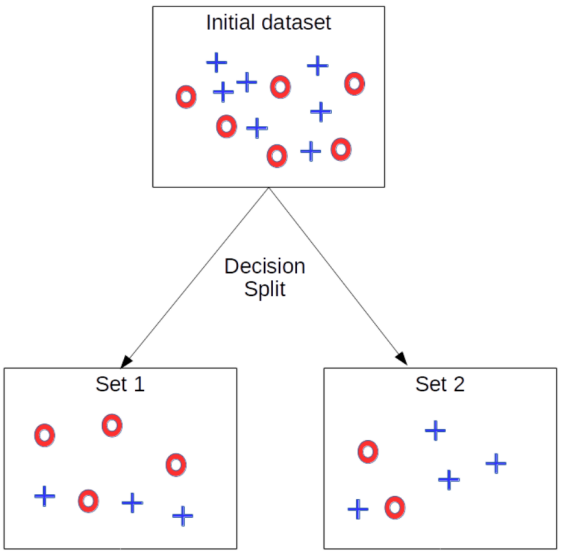


Figure 6: Dataset before and after Node

Selecting the optimal node with high purity needs to find the entropy as small as possible, because entropy is an amount which measures the degrees of confusion of passing current samples. From the image, it can be known that if the sample is highly chaotic, its probability value correspond to different samples can be small and its entropy value can be large.
After getting the basis for selecting the node, the next step is to construct the decision tree.
the basic idea of constructing decision tree is that as the depth of the tree increases, the entropy of the node decreases rapidly. we aim to construct a decision tree with shallow depth because this kind of model can gain high classification efficiency. to gain this aim, we seek for nodes with faster decreasing rate.

3.1.2 C4.5

The earliest and most influential decision tree algorithm is the ID3 algorithm. However, for the ID3 algorithm, it has many defects. For example, the ID3 algorithm uses all the current training examples at each step of the search, which greatly reduces the sensitivity to faults in individual training examples. inherent biasing exists when measuring the information divergence, which can lead to in favors of attributes with more values. The ID3 algorithm grows the depth of each branch of the tree until it is perfectly categorized for training samples, and it can cause decision tree overfitting problem.

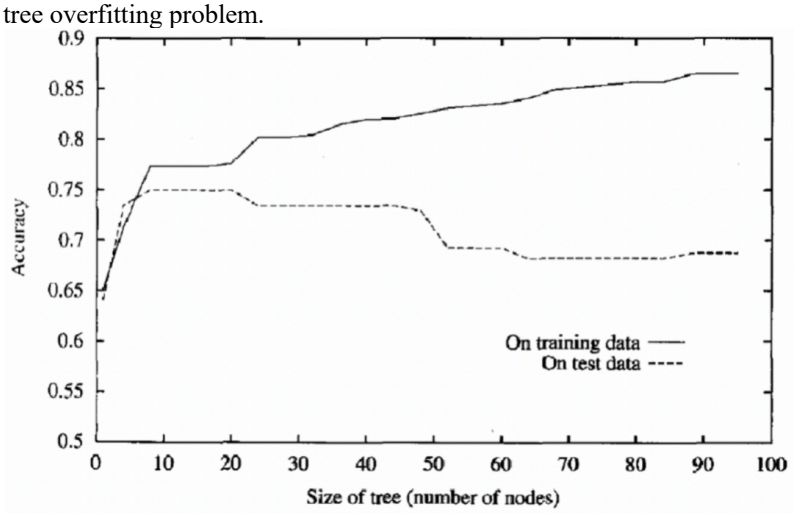


Figure 7: Accuracy of model in different size of tree

With the continuous research, the decision tree algorithm has been continuously improved. The arise of C4.5 method trim to solve the shortcoming of ID3 decision tree, which can be detailed described below:
When the attribute can be assigned with a large number of values, there may be only one or a few samples under the corresponding attribute. For this circumstance, the information divergence and purity are very high. If using the ID3 algorithm, the decision tree will treat this attribute as a suitable one for division. While the problem of this kind of attributes is the comparison weak of its generalization ability, which unable to predict new samples effectively and efficiently.

For the C4.5 decision tree, it does not directly use the information divergence as the main basis for dividing the sample, but proposes another concept, the [8].
Before calculating the Information Gain Rate, we should understand so concepts. Firstly, it is Information Gain that is used to indicate the difference between the entropy value and the original entropy value after the sample passes through the node:

$$\text{Information Gain} = \text{Entroy(before this node)} - \text{Entroy(after this node)}$$

Fromula:

$$Gain(D, A) = H(D) - H(A)$$

And

$$\text{Information Gain Rate} = \text{Penalty parameter} * \text{Information Gain}$$

So, the formula of Information Gain Rate, $G_R(D, A)$ is:

$$G_R(D, A) = \frac{Gain(D, A)}{H_A(D)}$$

And $H_A(D)$ is the empirical entropy of sample set D, which uses the current feature A as a random variable.

$$H_A(D) = -\sum_{i=1}^{n} P_i \log_2 P_i$$

Penalty parameter is the reciprocal of the entropy of the data set D with the feature A as the random variable.

$$\text{Penalty parameter} = \frac{1}{H_A(D)} = -\frac{1}{\sum_{i=1}^{n} P_i \log_2 P_i}$$

3.1.3 Random forest:

In 2001, Breiman et al. combined decision trees into random forests, and this is the appearance of random forest for the first time.
The basic principle of random forest depends on integrated learning [9]. Integrated learning is a machine learning method that uses a series of learners to learn and uses some rules to integrate the results of each learner to achieve better learning effect than a single learner.

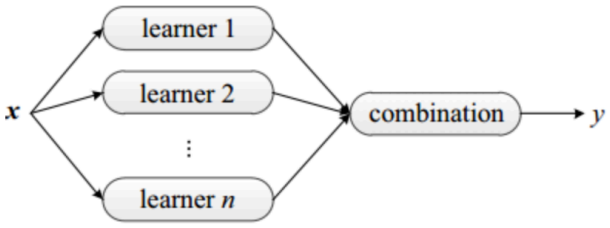


Figure 8: Integrated Learning

The accuracy of the classifier, the classification accuracy of each individual classifier must be greater than 0.5. As shown in the figure below, it can be seen that if p<0.5, the classification accuracy will decrease as the integration scale increases, but if it is greater than 5, the final classification accuracy can be 1.

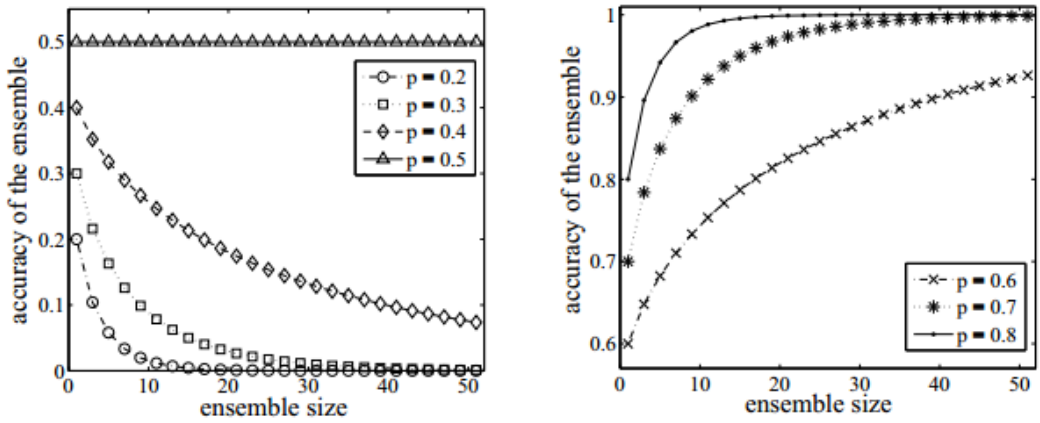


Figure 9: Accuracy of the ensembles with different ensemble size

Random forests use this idea to combine multiple decision trees together to improve the accuracy of decision making. But for it, its internal decision tree combination is not irregular. It uses the bootstrap resampling [10] technique to randomly extract and put back k samples from the original training sample set N to generate a new

training sample set, and then generate k classification tree sets in new training sample set to form a random Forests, the classification results of new data are determined by the number of votes cast by the decision tree. The essence of random forest is an improvement of the decision tree algorithm combined with multiple decision trees. Constructing each tree depends on an independently extracted sample. Each tree in the forest has the same distribution, and the classification error depends on the classification ability of each tree and the correlation between them. Feature selection uses a random method to split [11] each node and then compare the errors produced in different circumstance. The number of selected features can be determined by the inherent estimation errors, classification capabilities, and correlations. A single tree's classification ability may be small, while a test sample can select the most likely classification according to the classification results of each tree after randomly generating a large number of decision trees.

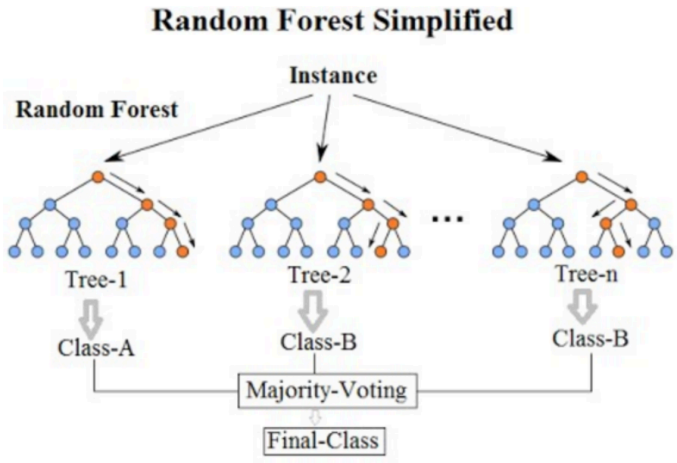


Figure 10: Random Forest

There are two advantages for this randomization process. The first advantage is the ability of not easy falling into overfitting problem or being affected by noise. The second advantage is there is no correlation between each tree, which means that they have no influence on other tree's decision making. in summary, the random forest can improve the classification accuracy under the premise that the computational complexity is not significantly improved.
Each classification tree in the random forest is a binary tree, and its generation follows the principle of top-down recursive splitting. In the binary tree, the root node contains all training data, which can be split into left and right nodes according to the information divergence principle. Each node contains a subset in training samples. Divided nodes continue to split until meeting the branch stop rule.

3.2 SVM

3.2.1 introduction

In this part, the method that we describe is SVM, Support Vector Machine. In the field of machine learning, it is a supervised learning model that is commonly used for pattern recognition, classification, and regression analysis. For the case of linear inseparability, it uses a nonlinear mapping algorithm to transform a linearly indivisible sample of low-dimensional input space into a high-dimensional feature space to make it linearly separable. Therefore, it is possible to linearly analyse the nonlinear characteristics of the sample using a linear algorithm in the high-dimensional feature space.
To detailed describe the classification process, two perspectives can be demonstrated. The core attribute in both perspectives is margin. for the first perspective, probability model $h_\theta = g(\theta^T x)$ can be used to predict by labeling 0 or 1 in logistic regression. In this way, $h_\theta(x) = p(y = 1|x; w, b)$ reflect the degree of confidence which means the correctness of this classification. For another perspective, a decision boundary arises towards the figure below:

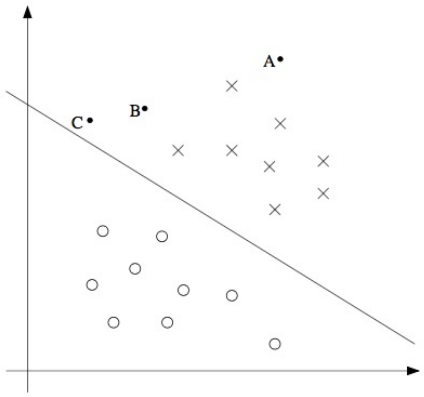


Figure 11: decision boundary for classification

Decision boundary which is the boundary to classifier positive training examples and negative training examples is also called separating hyperplane. Hyperplane in the feature space based on the structural risk minimization theory, so that the learner is globally optimized. By applying separating hyperplane, prediction can also made which can be derived as the further distance between training samples and decision boundary, the more confident of making this prediction.
In this way, to determine the decision boundary, notations should be applied for classification. The notation is shown where b is the intercept term:

$$h_{w,b}(x) = g(w^T x + b)$$

3.2.2 Define functional and geometric margins

As the notation of classification is given, functional and geometric margins should be represented based on it. Functional margin can be represented by notation below:

$$\hat{\gamma}^{(i)} = y^{(i)}(w^T x + b)$$

This notation is transition of the logistic regression transition, and this transition ensures the intercept term b separately from the other parameters. The relationship between functional margin and the correctness is that the larger functional margin is, the more confident and correct the prediction is.
It is defined that the function margin of (w, b) with respect to S should be smallest where S is the individual training samples. This can be represented the notation:

$$\hat{\gamma} = \min_{i=1,\ldots,m} \hat{\gamma}^{(i)}$$

the notation is listed below. In the same way, the geometric margins should also be minimized, which is linked with functional margin by unit-vector.
Corresponding geometric margin can be shown in picture and the notation is shown below.

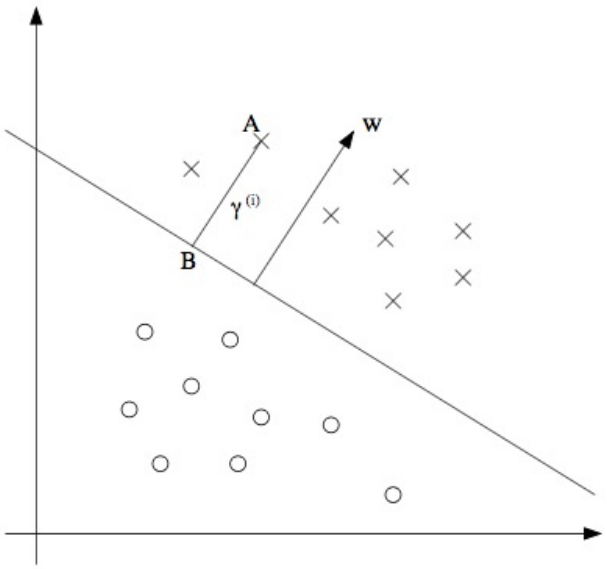


Figure 12: Geometric margin for classification

$$w^T\left(x^{(i)}-\gamma^{(i)}\frac{w}{\|w\|}\right)+b=0$$

$x^{(i)}$ is the input of training samples which labels is 1. The distance between this input and decision margin can be the line segment AB, w is the vector which is orthogonal to the separating hyperplane.

By using unit-vector $\frac{w}{\|w\|}$, functional margin and geometric margin can be linked, especially when $w^Tx+b=0$, the values are equal. The explanation is that x can be replaced by $x^{(i)}-\gamma^{(i)}\frac{w}{\|w\|}$, which means this is the point which lies on the decision boundary that satisfy the equation $w^Tx+b=0$ [12]. There exists the formation deformation:

$$w^T\left(x^{(i)}-\gamma^{(i)}\frac{w}{\|w\|}\right)+b=0$$

The characteristic of logistic regression defines that $\{-1,1\}$ is not a good measure of confidence, but classifier can be described as no change will happen if rescaling or change the magnitude of (w,b), for the reason that $h_{w,b}(x)$ keep unchanged. This invariant feature help to solve formula problems in the next chapter.

3.2.3 The optimal margin classifier

The process of finding optimal margin classifier which has mentioned ahead is to find the maximum margin of the decision boundary, as maximized margin can provide the confident prediction. so the formation shows below:

$$\max_{\gamma,w,b}\gamma$$
$$s.t.\ y^{(i)}\left(w^Tx^{(i)}+b\right)\geq\gamma, i=1,\dots,m$$
$$\|w\|=1$$

It is assumed that the geometric margin is at least the value of functional margin ($\|w\|=1$), while it is difficult to calculate the value of maximum margin. To get the value, we need to apply the invariant characteristic which we mentioned before. By changing the scaling constraint of w and b to keep the value of training set equal to $1(\hat{\gamma}=1)$, we can get the transition of formation:

$$\max_{\gamma,w,b}\frac{1}{2}\|w\|^2$$
$$s.t.\quad y^{(i)}\left(w^Tx^{(i)}+b\right)\geq 1, i=1,\dots,m$$

In this way, the difficulty of finding the maximum margin can be transferred to solving the equation which contains a convex quadratic objective and only linear constraints, and this problem is simpler than before.

3.2.4 Lagrange duality

The introduction of Lagrange duality is necessary as it relate to optimization problem's dual form, which is a key element for using kernels to get the classification optimization in a high dimension with efficiency.

Our primal optimization problem can be expressed below:

$$\min_w f(w)$$
$$s.t.\quad h_i(w)=0, i=1,\dots,l$$

Lagrangian can be defined below, where l's partial derivatives to 0:

$$\mathcal{L}(w,\beta)=f(w)+\sum_{i=1}^{l}\beta_ih_i(w)$$

Generalized Lagrangian can be defined to solve the primal optimization problem [13], where α_i,β_i are Lagrange multipliers:

$$\min_w f(w)$$
$$s.t.\quad g_i(w)\leq 0, i=1,\dots,k$$
$$h_i(w)=0, i=1,\dots,l$$

$$\theta_{\mathcal{P}}(w)=\max_{\alpha,\beta:\alpha_i\geq 0}\mathcal{L}(w,\alpha,\beta)$$

The formation that should be verified is shown below:

$$\theta_{\mathcal{P}}(w)=\max_{\alpha,\beta:\alpha_i\geq 0}f(w)+\sum_{i=1}^{k}\alpha_ig_i(w)+\sum_{i=1}^{l}\beta_ih_i(w)=\infty$$

So, by judging whether the multiplier satisfy the primal constraints, we can get the formations in two conditions:

$$\theta_{\mathcal{P}}(w)=\begin{cases}f(w)\\ \infty\end{cases}$$

For considering the minimization problem, we define the optimal value of the objective as the value of primal problem:

$$p^* = \min_w \theta_{\mathcal{P}}(w)$$

Another problem which is dual problem, and the max and min of dual problem is exchanged compared with primal problem:

$$\max_{\alpha,\beta:\alpha_i \geq 0} \theta_{\mathcal{D}}(\alpha,\beta) = \max_{\alpha,\beta:\alpha_i \geq 0} \min_w \mathcal{L}(w,\alpha,\beta)$$

The relation of dual problem and primal problem is below, and there is an equation when under certain conditions:

$$d^* = \max_{\alpha,\beta:\alpha_i \geq 0} \min_w \mathcal{L}(w,\alpha,\beta) \leq \min_w \max_{\alpha,\beta:\alpha_i \geq 0} \mathcal{L}(w,\alpha,\beta) = p^*$$

$$d^* = p^*$$

The certain condition can be that there exists some w so that $g_i(w) < 0$ for all i, where f and g_i's is convex, and h_i are affine. The constraints g, are feasible. So there should exist w^*, α^*, β^* where is the primal problem's solution, α^*, β^* are dual problems solution. All multipliers can meet the condition of KKT, and this condition is shown below:

$$\frac{\partial}{\partial w_i}\mathcal{L}(w^*,\alpha^*,\beta^*) = 0, i = 1,\dots,n$$

$$\frac{\partial}{\partial \beta_i}\mathcal{L}(w^*,\alpha^*,\beta^*) = 0, i = 1,\dots,l$$

$$\alpha_i^* g_i(w^*) = 0, i = 1,\dots,k$$

$$g_i(w^*) \leq 0, i = 1,\dots,k$$

$$\alpha^* \geq 0, i = 1,\dots,k$$

There exists an important condition called KKT dual complementarily condition, which we can apply to show the number of support vectors in SVM and the convergence test for SMO algorithm which support the optimization of SVM.

3.2.5 optimal margin classifiers

Support vectors can be shown on the graph which are the closest points to the decision boundary, as those points represent the optimal solution for optimization.

Now several formations of the derivative and indication of the Lagrangian can be used to get only the inner product$\langle x^{(i)}, x^{(j)} \rangle$ which support the applying of Kernels.

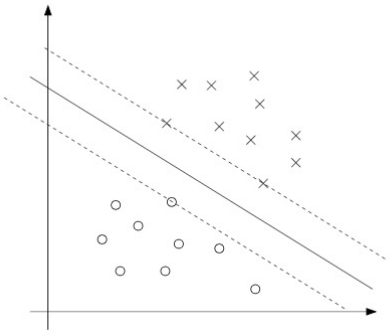


Figure 13: The closest points to decision boundary

The transition of Lagrangian lay the foundation of using Kernels by only figuring out certain values. So the transitioned formation can be listed as the dual problem:

$$\max_\alpha W(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i,j=1}^m y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle$$

$$s.t.\ \alpha_i \geq 0, i = 1,\dots,m$$

$$\sum_{i=1}^m \alpha_i y^{(i)} = 0$$

The optimal value for intercept term b can also be listed:

$$b^* = -\frac{\max_{i:y^{(i)}=-1} w^{*T} x^{(i)} + \min_{i:y^{(i)}=-1} w^{*T} x^{(i)}}{2}$$

So the quantity which need to calculate and depend on the inner product can also be written:

$$w^T x + b = \left(\sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}\right)^T x + b = \sum_{i=1}^{m} \alpha_i y^{(j)} \langle x^{(i)}, x \rangle + b$$

3.2.6 Kernels

Kernel is the algorithm which can help the classification in high dimensions. The character of kernel can be explained below for the reason why it is an efficient algorithm.

At the beginning, several terms should be mentioned. Input attribute is the attribute which has not been processed. By passing the input attributes by learning algorithm, input feature can be gain. The learning algorithm can be represented by feature mapping. it is the procedure of kernel algorithm.

In this way, feature mapping can be applied in the original formation, which replace x with $\phi(x)$, the corresponding kernel formation can be shown:

$$K(x, z) = \phi(x)^T \phi(z)$$

Examples can be given to support the efficiency of this replace. To detail demonstrate it, the calculating process of kernel formation can be much simpler and more inexpensive than the calculation spent on $\phi(x)$ itself, as it is easy to get the value of inner product. that is the reason why kernel algorithm is efficient.

Another property is that despite the high dimensions of classification, kernel can take only linear time which means no high dimensional feature needed to be represented by feature vectors. What's more, the value of kernel algorithm can also represent the similarity of $\phi(x)$ and $\phi(z)$ [14]. So to calculate the similarity of x and z, we can apply the special kernel algorithm which called Gaussian kernel with infinite dimensional feature mapping.

The target is to get valid kernel. The kernel matrix which is the representation of all examples with the premise of valid kernel should be symmetric positive semidefinite. This is the elaboration of Mercer theorem, and this theorem can help the testing of whether it is a valid kernel.

The formation of this kernel matrix property can be shown below:

$$K_{ij} = K\left(x^{(i)}, x^{(j)}\right) = \phi\left(x^{(i)}\right)^T \phi\left(x^{(j)}\right) = \phi\left(x^{(i)}\right)^T \phi\left(x^{(j)}\right) = K\left(x^{(i)}, x^{(j)}\right)$$

$$z^T K z = \sum_i \sum_j z_i K_{ij} z_j$$

$$= \sum_i \sum_j z_i \phi(x^{(i)})^T \phi\left(x^{(j)}\right) z_j$$

$$= \sum_i \sum_j z_i \sum_k \phi_k\left(x^{(i)}\right) \phi_k\left(x^{(j)}\right) z_j$$

$$= \sum_k \sum_i \sum_j z_i \phi_k\left(x^{(i)}\right) \phi_k\left(x^{(j)}\right) z_j$$

$$= \sum_k \left(\sum_i z_i \phi_k\left(x^{(i)}\right)\right)^2$$

$$\geq 0$$

There z can be any vector, so K is positive semi-definite in this formation.

3.2.7 Regularization and the non-separable case

As all mentioned before, we can understand the main concept of SVM, while optimization should be applied to make the SVM less sensitive and more efficient.

Regularization should point out to deal with the drawback of SVM, which is the difficulty of figuring out the boundary of non-linear data samples and keep the stability of decision boundary. the formation can also be optimized as below:

$$\min_{\gamma, w, b} \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{m} \xi_i$$

$$s.t. \quad y^{(i)}\left(w^T x^{(i)} + b\right) \geq 1 - \xi_i, i = 1, \ldots, m$$

$$\xi_i \geq 0, i = 1, \ldots, m$$

This formation can be explained as C is the variable to keep functional margin of examples at least 1, while the right side of this formation ensure the functional margin less than 1. The main change of this optimization process is that the constraint can be transferred from $0 \leq \alpha_i$ to $0 \leq \alpha_i \leq C$. By applying and simplifying Lagrangian and the derivative targeted to dual problem, we can get the final formation and solve the optimization problem by solving this formation. The algorithm we use to optimize and solve problem is SMO algorithm which is based on coordinate ascent algorithm.

3.2.8 The SMO algorithm

Using the optimal margin classifier which apply to functional and geometric margins, combined with kernel algorithm which lay on lagrange duality and the regularization, we can find the core element for this SVM classification process is the solution of dual problem, and this problem can be addressed by SMO algorithm. SMO algorithm is the algorithm which combine the theory of coordinate ascent, but the difference is that the constraint α can not be easily addressed because not all other constraints are fixed, which means that the change of this constraint can have an influence on the violation of other constraints [15]. So SMO apply the method which is the transition of coordinate ascent:

Repeat till convergence {

1. Select some pair α_i and α_j to update next (using a heuristic that tries to pick the two that will allow us to make the biggest progress towards the global maximum).
2. Reoptimize $W(\alpha)$ with respect to α_i and α_j, while holding all the other α_k's $(k \neq i, j)$ fixed.

}

Figure 14: SMO method

This process can good address the dual problem and keep the efficiency of computing. The convergence tolerance parameter and KKT condition can be used for the testing of algorithm convergence.
So the solution of the dual problem can be attributed to two constraints. Some formation can be applied for the solution:

$$\alpha_1 = \left(\varsigma - \alpha_2 y^{(2)}\right) y^{(1)}$$

Objectives can also be shown:

$$W\left(\alpha_1, \alpha_{2,} \dots, \alpha_m\right) = W\left(\left(\varsigma - \alpha_2 y^{(2)}\right) y^{(1)}, \alpha_{2,} \dots, \alpha_m\right)$$

The formation of one constraint is shown:

$$\alpha_2^{new} = \begin{cases} H, if\ \alpha_2^{new,unclipped} > H \\ \alpha_2^{new,unclipped}, if\ L \leq \alpha_2^{new,unclipped} \leq H \\ L, if\ \alpha_2^{new,unclipped} < L \end{cases}$$

So the dual problem can be addressed by SMO. The whole classification process of SVM is described and the detailed derivative process and the solution is given.

3.3 Adaboost

In the third part, the method we explained is Adaboost.
When making classification decisions, the effect of a single classifier will always have its limitations, which is unavoidable. At this time, if an algorithm can be used to integrate the decision results of different classifiers, the decision effect of the classifier can be improved to a large extent. This is the core idea of the meta-algorithm [16]. The meta-algorithm combines different classifiers, to improve the effect of classification decision. It has multiple implementations. For example, it can be the integration of different algorithms, or the integration of different algorithms under different settings. What is more, it can be the integration of different parts of the data set assigned to different classifiers.
Adaboost [17], "Adaptive Boosting", proposed by Yoav Freund and Robert Schapiro in 1995, is one of the most popular meta-algorithms. Adaboost is an improved algorithm optimized by Bagging [18].
The Bagging algorithm is based on bootstrap sampling. Given a data set containing m samples, we first randomly take a sample into the sampling set, and then put the sample back into the initial data set, so that the sample may still be selected at the next sampling.

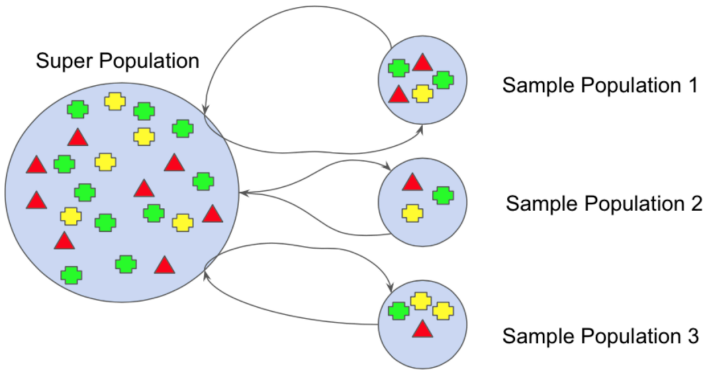


Figure 15: Bootstrap sampling

After m times of randomization, we obtained a sample set with m samples. Thus, T new data sets are obtained after selecting T times from the original data set, and the size of each new data set is equal to the size of the original data set, with m samples.
After the T new data sets are built, a classification is applied to each data set to obtain T classifiers. When we want to classify new data sets, we can apply these T classifiers for classification. At the same time, the category with the largest number of classifier voting results is selected as the final classification result.

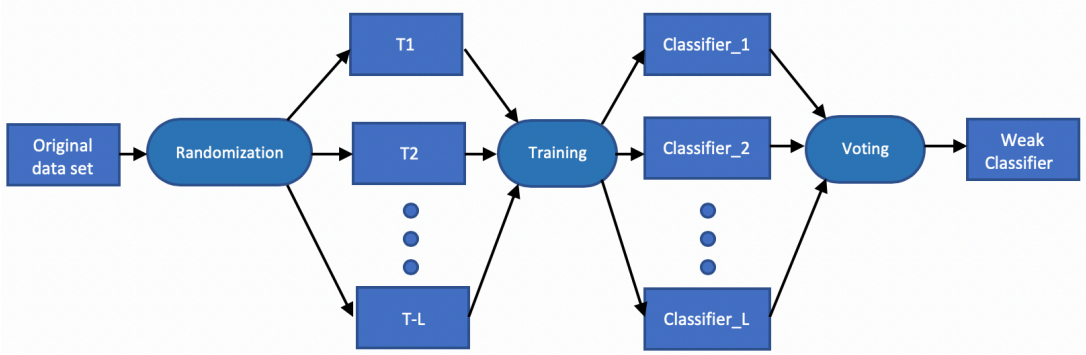


Figure 16: Bagging

In the bagging algorithm, the voting weights of each classifier trained by the randomly selected sample data set are consistent, and thus there is a case where the data of the misclassified data is insensitive [19]. In order to optimize the classification decision for misclassified samples, the Adaboost algorithm makes the following modifications on the basis of bagging. The weight of the sample that is misclassified by the previous basic classifier will be increased, and the weight of the sample that is correctly classified will be reduced. And, these updated weights will be used again to train the next basic classifier. At the same time, a new weak classifier is added in each iteration, until it reached a predetermined sufficiently small error rate, or a pre-specified maximum number of iterations. At the same time, it determines the final strong classifier.

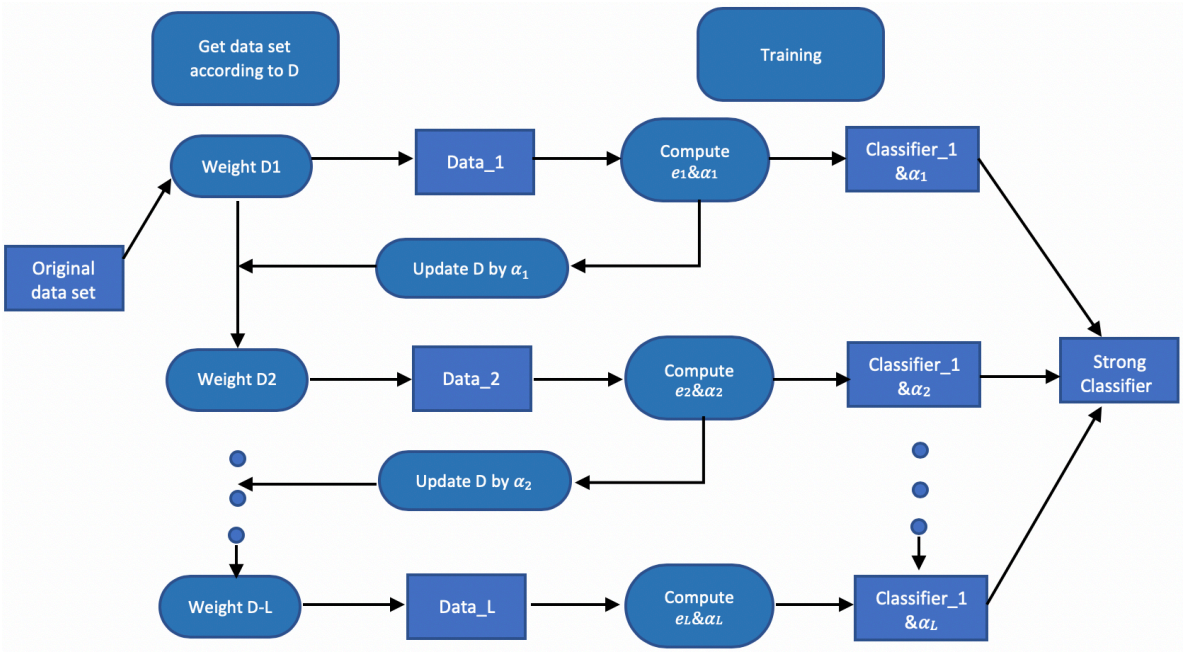


Figure 17: Adaboost

The implementation process of Adaboost algorithm is as follows:
The data set that we used in this method is:

$$T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$$

And x_i is the input data, $x_i \in X \subseteq R^n$; $y_i$is the label of the input data, $y_i \in Y = \{-1, +1\}$
Initialize the probability distribution of the training data, starting with a uniform distribution.

$$D_1 = (w_{11}, w_{12}, \dots, w_{1N}), and\ w_{1i} = \frac{1}{N}, i = 1,2,3, \dots, N$$

w is the weight of the sample.
Adaboost algorithm can be understood as an algorithm based on "additive model"， as the linear combination of base classifier.

$$H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

And $h_t(x)$is the base classifier, e.g. Decision tree, SVM; $\alpha_t$is the coefficient of the base classifier, which is used to minimize the exponential loss function [20].

$$L(f(x), H(x)) = \exp[-f(x)H(x)]$$

And $f(x)$ is the correct label of the train data, whose value should be -1 or 1; $H(x)$ is the result of the classification, $H(x) \in \{-1,1\}$.
So, in order to get the best classification, we should minimize the exponential loss function.

$$L(f(x), H(x)) = \mathrm{P}[f(x) = h_t(x)]\, e^{-\alpha_t} + \mathrm{P}[f(x) \neq h_t(x)]\, e^{\alpha_t} = (1 - \varepsilon_t)e^{-\alpha_t} + \varepsilon_t e^{\alpha_t}$$

And $\mathrm{P}[f(x) = h_t(x)]$ is the possibility of correctly classified samples; $\mathrm{P}[f(x) \neq h_t(x)]$ is the possibility of incorrectly classified samples.
To find the partial derivative of $\alpha_t$for the $L(f(x), H(x))$:

$$\frac{\partial L(f(x), H(x))}{\partial \alpha_t} = -(1 - \varepsilon_t)e^{-\alpha_t} + \varepsilon_t e^{\alpha_t}$$

And ε_t is the incorrect rate of base classifier, $h_t(x)$.
To get the minimum of the loss function:

$$\frac{\partial L(f(x), H(x))}{\partial \alpha_t} = 0$$

So,

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right)$$

First training a basic classifier on the training classifier and calculate the error rate of the class, then training the basic classifier again on the same data set.

In the second training of the classifier, the weight of each sample will be re-adjusted. The weight of the sample that is misclassified by the previous basic classifier will be increased, and the weight of the sample that is correctly classified will be reduced. To get the final classification result from all basic classifiers, Adaboost assigns each classifier a weight value w_{mi}, which is calculated basic on the error rate of each basic classifier. D_m represents the probability distribution (or weight distribution) of the training data before the start of the m-th iteration, and w_{mi} represents the weight of the i-th sample.

$$D_{\mathrm{m+1}} = \left(w_{\mathrm{m+1,1}}, w_{\mathrm{m+2,1}}, \dots, w_{\mathrm{m+1},N}\right),$$

And

$$w_{\mathrm{m+1},i} = \frac{w_{mi}}{Z_m}\exp\left(-\alpha_m y_i H_m(x_i)\right), i = 1,2,3, \dots, N$$

Z_m is the normalization factor,

$$Z_m = \sum_{i=1}^{N} w_{mi}\exp\left(-\alpha_m y_i H_m(x_i)\right), \text{ and } \sum_{i=1}^{N} w_{m+1,i} = 1,$$

If the sample is classified in a correct label,

$$w_{\mathrm{m+1},i} = \frac{w_{mi}}{Z_m}e^{-\alpha_m}$$

If the sample is classified in an incorrect label,

$$w_{\mathrm{m+1},i} = \frac{w_{mi}}{Z_m}e^{\alpha_m}$$

After calculating $D_{\mathrm{m+1}}$, the Adaboost algorithm begins to move to the next iteration.
When it reached a predetermined sufficiently small error rate, or a pre-specified maximum number of iterations, it will stop. At the same time, it determines the final strong classifier.

3.3.1 Softmax

The above describes in detail how the Adaboost algorithm implements classification optimization. As a kind of integrated learning, its classification effect is naturally affected by the effect of the basic classifier. After comparing several different base classifiers, we used two kinds of basic classifiers: decision tree and softmax [21].
Softmax is a kind of logistic regression for multi-classification. For general logistic regression algorithms, it is used to solve the two-class problem. This means that for the samples of the dataset, there are only two types of labels for their values. However, softmax makes some modification on the algorithm of logistic regression, so that it can solve the multi-class problem, that is, for m samples in the data set, the value of the label can be from 1 to k, $y^{(i)} \in \{1,2, \dots, k\}$.
As a classifier, it can output its corresponding label when inputting the characteristic value of the test data. Similar to logical regression, it outputs the probability of the input sample as a different classification:

$$g_\theta(x) = \begin{bmatrix} p\left(y^{(i)}\right) = 1|x^{(i)}; \theta \\ p\left(y^{(i)}\right) = 2|x^{(i)}; \theta \\ \dots \\ p\left(y^{(i)}\right) = k|x^{(i)}; \theta \end{bmatrix}$$

Similar to logical regression, it uses the sigmoid function(引用) as the activity function.
The input of this function is $\theta^T X$.
So,

$$g_\theta(x) = \frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x^i}}\begin{bmatrix} e^{\theta_1^T x^{(i)}} \\ e^{\theta_2^T x^{(i)}} \\ \dots \\ e^{\theta_k^T x^{(i)}} \end{bmatrix}$$

And $\theta_1, \theta_2, \dots, \theta_k$ is the parameter of the model.
So, for a sample, the possibility of the label $y^{(i)}$:

$$P(y^{(i)} = j|x^{(i)};\ \theta) = \frac{e^{\theta_j^T x^{(i)}}}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}}$$

Its loss function is:

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} 1\{y^{(i)}\}\log\frac{e^{\theta_j^T x^{(i)}}}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}}\right]$$

And m is the number of the train data set.
The function $1\{*\}$ means:

$$1\{* = ture\} = 1$$

$1\{* = false\} = 0$
In order to find the optimal solution, we use Gradient Descent method in the loss function.

$$\nabla_{\theta_j} J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[x^{(i)}\left(1\{y^{(i)} = j\} - P(y^{(i)} = j|x^{(i)};\ \theta)\right)\right]$$

4. Experiments and Discussion

4.1 Experiments, comparisons and evaluation

Before we really choose the algorithm and start our experiment for this assignment, we tested the KNN and logistic regression algorithm that we used in first assignment with the help of "sklearn" library. For KNN algorithm, take the time cost into consideration, we add a PCA operation to reduce its dimensions before, which largely increase the efficiency of the KNN and bring a little bit loss in accuracy. As for the logistic regression, we set the arguments like this: the max iteration number is 50, the range of error for putting an end to iteration is 0.001, the classification method is "one-over-rest", and use "Stochastic gradient descent" to reduce the time to find the minimum loss function. Both of the two results are satisfactory, the accuracy of the 10000 test examples both exceed 80%. And the running time is less than 3minute.

4.1.1 Random forest

Random forest is a fast way for classification, we used "RandomForestClassifier" in sklearn library to build the classifier. In order to make our classifier suit our training examples, we need careful adjustment on the arguments. The "n_estimators" means the maximum number of the weak learning machine, if it's pretty small, it could cause Underfitting. In contrast, if it's pretty larger, it increases the volume of work. Meanwhile, when n_estimators reach a certain number, the sequent increase won't improve the accuracy much, here we choose 20 as n_estimators. The" criterion" is the criteria when partition CART tree, we choose "Gini" index [22]. Next we need to choose Decision Tree arguments, we limited the maximum depth of tree as 30, and remain the rest arguments as default. The result of this classifier is fantastic, the correct rate can reach 86% and the running time is around 18s.

4.1.2 SVM algorithm

Secondly, we experimented the SVM algorithm, there're three relevant functions, which are SVC, NuSVC, LinearSVC, because the SVC only support the training example that is less than 10000, we have to abandon it, the NuSVC is Similar to SVC but uses a parameter to control the number of support vectors.
LinearSVC's implementation is based on libsvm. Similar to SVC with parameter kernel='linear', but implemented in terms of liblinear rather than libsvm, so it has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples. So, we finally chose this function and set the maximum iteration time to 500, because the further augment wouldn't improve the performance, instead, the accuracy dropped. We choose the standard penalty 'l2'.

4.1.3 adaptive boosting

In respect to third experience, we used adaptive boosting as our algorithm. AdaBoost can be used to boost the performance of any machine learning algorithm. For example, if use a decision tree, it can use only those important features and the integration of multiple weak classifiers can improve the predictive ability of the model.
For comparison, we chose the default weak classifier: Decision Tree and other one: Logistic regression using softmax for multiple classification. Firstly, we test the performance of Decision tree, we set the number of classifiers to 10 and the depth of decision tree to 30, minimum sample split to 10. The performance is acceptable, it costed 6 minutes with accuracy of 83%
Then, we test the Adaboost with softmax, we set it to stochastic gradient descent and limit the maximum iteration to 50, build 10 weak classifiers. As the result, the performance wasn't very good, not to mention the running time reach to 15 minutes, the accuracy reduced to 65%.

4.2 Meaningful discussion of results and design choices

4.2.1 analysis method

In the field of machine learning, confusion matrices are widely used analytical tables. It visualizes the performance of the algorithm by present its result for each class. Its effect on reviewing the performance of the classifier is excellent because it makes it easy to see if the classifier confuses different classes.

Table 1: Confusion matrix

	Actual class	

		Positive	Negative	Total
Predicted class	Positive	a(TP)	b(FP)	a + b
	Negative	c(FN)	d(TN)	c + d
	Total	a + c	b + d	N

Table 2: Meaning of TP, TN, FP and FN

TP	true positive
TN	true negative
FP	false positive
FN	false negative

Some Cost-Sensitive Measures as:

positive predictive value:

$$\text{Precision(p)} = \frac{a}{a+c}$$

true positive rate:

$$\text{Recall(r)} = \frac{a}{a+b}$$

F1 score is the harmonic mean of precision and sensitivity:

$$\text{F} - \text{measure(F)} = \frac{2a}{2a+b+c}$$

accuracy (ACC)

$$\text{ACC} = \frac{a+d}{N}$$

4.2.2 discussion over result

After the first experience of Random forest, we decide to reform the performance of this algorithm. As we already mentioned, Random Forest is a collection of Decision Trees, the Random Forest algorithm randomly selects observations and features to build several decision trees and then averages the results. From the basic idea of integrated learning, a combination of multiple weak learners is used as a new model to improve the prediction effect. As a kind of bagging model, random forest contains several independent sub-decision tree models (usually using CART decision tree as the base model), which is determined based on the results of each sub-model in the prediction process. So, we finally found the relatively balanced arguments, the number of decision tree is set to 50 and the max depth of tree is set to 60 after a serial test. The result is excellent with correct rate of 87 percent and running time within 48s

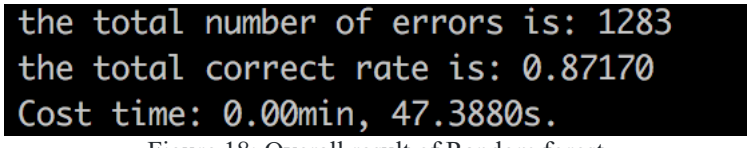
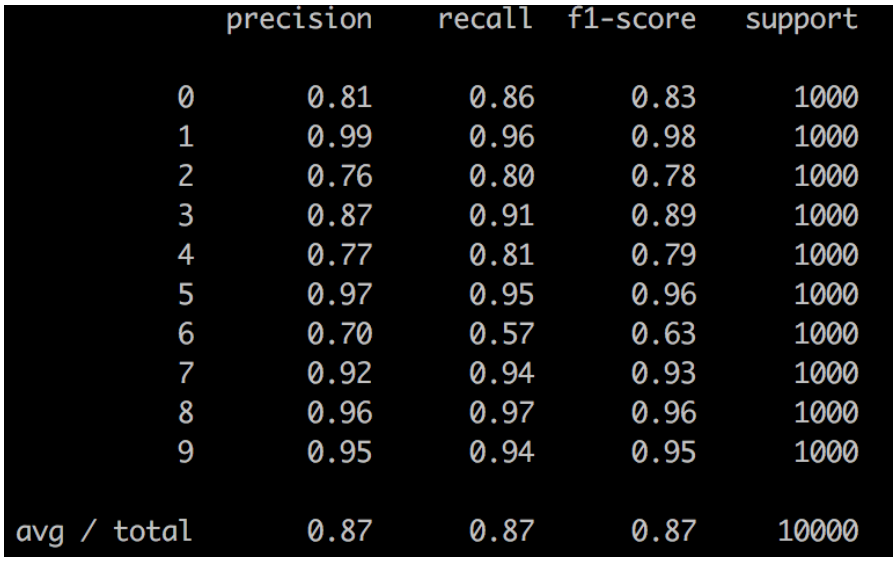


Figure 18: Overall result of Random forest

	pre1	pre2	pre3	pre4	pre5	pre6	pre7	pre8	pre9	pre10
class1	857	0	11	32	4	1	83	0	12	0
class2	2	959	1	24	6	0	6	0	2	0
class3	11	1	802	9	109	0	66	0	2	0
class4	21	3	10	906	30	0	26	0	3	1
class5	1	1	97	32	812	0	56	0	1	0
class6	0	0	0	1	0	955	0	32	1	11
class7	165	1	127	31	86	0	569	0	21	0
class8	0	0	0	0	0	19	0	942	0	39
class9	1	2	6	4	5	1	5	3	972	1
class10	0	0	0	0	0	11	0	43	3	943

Figure 19: The confusion matrix of Random forest



	precision	recall	f1-score	support
0	0.81	0.86	0.83	1000
1	0.99	0.96	0.98	1000
2	0.76	0.80	0.78	1000
3	0.87	0.91	0.89	1000
4	0.77	0.81	0.79	1000
5	0.97	0.95	0.96	1000
6	0.70	0.57	0.63	1000
7	0.92	0.94	0.93	1000
8	0.96	0.97	0.96	1000
9	0.95	0.94	0.95	1000
avg / total	0.87	0.87	0.87	10000

Figure 20: The precision recall and F-measure value for each class

The objective of the SVM algorithm is to find a hyperplane in an N-dimensional space (N—the number of features) that distinctly classifies the data points. The study rate is changed to 0.8, and loss function is set as "squared hinge", the main method for multi-class is "one-over-rest". The tolerance of the stopping criteria we set is 1e-4.

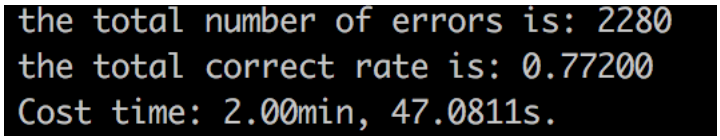


Figure 21:Overall result of SVM

	pre1	pre2	pre3	pre4	pre5	pre6	pre7	pre8	pre9	pre10
class1	896	5	56	4	2	0	12	0	23	2
class2	22	935	23	11	1	0	2	1	3	2
class3	26	2	950	1	9	0	5	0	6	1
class4	218	29	180	478	13	1	61	2	12	6
class5	27	0	768	1	154	1	42	0	7	0
class6	0	1	3	0	0	912	0	29	6	49
class7	263	1	456	5	33	0	211	0	31	0
class8	0	0	0	0	0	54	0	879	0	67
class9	18	2	34	2	1	14	3	7	918	1
class10	0	0	1	0	0	17	1	16	0	965

Figure 22: The confusion matrix of SVM

	precision	recall	f1-score	support
0	0.62	0.92	0.74	1000
1	0.93	0.95	0.94	1000
2	0.48	0.91	0.63	1000
3	0.80	0.80	0.80	1000
4	0.82	0.16	0.27	1000
5	0.96	0.87	0.91	1000
6	0.75	0.30	0.43	1000
7	0.90	0.94	0.92	1000
8	0.92	0.92	0.92	1000
9	0.90	0.95	0.93	1000
avg / total	0.81	0.77	0.75	10000

Figure 23: The precision recall and F-measure value for each class

The “adaptive boosting” is time consuming algorithm, because it likes a voting system with several weak classifier and combine them to a strong classifier. In terms of the choice of the weak classifier for adaptive boosting algorithm, we had a discussion about it, the first choice is the default classifier: Decision tree, the second choice is logistic regression, because the algorithm we chose is “Samme.R”, which has a faster iteration speed, so the weak classifier must support sample weight and KNN doesn’t support sample weight, so we ignore it .In matter of speed, the decision tree could faster with no doubt, but in matter of accuracy, there’s need for further tests. we tried to use two different kinds of weak classifier to compare which one is better.

```
the total number of errors is: 1679
the total correct rate is: 0.83210
Cost time: 5.00min, 29.4505s.
```

Figure 24: Overall result of Adaptive boosting

	pre1	pre2	pre3	pre4	pre5	pre6	pre7	pre8	pre9	pre10
class1	803	0	14	38	7	1	130	0	7	0
class2	4	945	2	33	6	0	8	0	2	0
class3	22	0	717	6	114	0	138	0	3	0
class4	41	4	19	850	39	0	45	0	1	1
class5	3	0	125	48	715	0	103	0	6	0
class6	0	0	0	1	0	923	0	46	7	23
class7	160	0	126	30	96	1	570	0	17	0
class8	0	0	0	0	0	25	0	922	2	51
class9	3	1	12	3	3	3	23	5	946	1
class10	0	0	0	0	0	16	2	49	3	930

Figure 25: The confusion matrix of Adaptive boosting

	precision	recall	f1-score	support
0	0.78	0.80	0.79	1000
1	0.99	0.94	0.97	1000
2	0.71	0.72	0.71	1000
3	0.84	0.85	0.85	1000
4	0.73	0.71	0.72	1000
5	0.95	0.92	0.94	1000
6	0.56	0.57	0.56	1000
7	0.90	0.92	0.91	1000
8	0.95	0.95	0.95	1000
9	0.92	0.93	0.93	1000
avg / total	0.83	0.83	0.83	10000

Figure 26: The precision, recall and F-measure value for each class

4.3 Relevant personal reflection

First of all, one of our team members suggest that we can use random forest algorithm with the following reason: It can handle very high dimensional data without having to make feature selection, because its feature subsets are randomly selected, Secondly, Insensitive to missing values, accuracy can be maintained if a significant portion of the features are lost. Thirdly, trees are independent of each other during training, and the training speed is fast, and it is easy to make a parallelization method. So that's why we choose this algorithm to implement our test. Besides, since it's a fast method, and can deal with the high dimension dataset easily, we decided not to use any normalization or simplification process before, if the result didn't perform well, we can change.

According to the flection from our team member, SVC function is time consuming without preprocessing, and the probable time cost could exceed twenty minus, and the official document deprecate the using of SVC if the dataset over 10000, so we turned to LineSVC to solve our task. During the experiment, the setting of the iteration argument is sensitive, when this argument reaches a certain point, it becomes a relatively good, and continuous augment of it won't increase accuracy, in contrast, accuracy dropped.

One of our team members proposed that we can use Adaptive boosting as finally algorithm, it focuses on classification problems and aims to convert a set of weak classifiers into a strong one. The Adaboost algorithm is one of the Boosting ensembles. It is called Adaptive boosting because in each weak classifier, the sampled faults are used to train the next weak classifier. These weak classifiers may be weak, but as long as the classification effect is better than random, the model effect can be improved.

5. Conclusion and future work

To sum up, the random forest is relatively better comparing with the other two algorithms, compared with a single model such as a decision tree, random forests have the characteristics of integrated learning. So, it's fast and accuracy, technically, its running time cost is lowest among the three algorithm and get a highest accurate result.

The SVM algorithm is still an efficient method to deal with classification, it seems the SVM has a not pretty amazing performance in this dataset since the accuracy of it only reach 77 percent, it's a little dissatisfactory thinking of the logistic regression can own 87 percent accuracy.

AdaBoost is adaptive and not too sensitive to parameters, even though it's a time-consuming algorithm. It can avoid "dimension disaster" to a certain extent. It only needs to construct weak classifiers. The reason why the accuracy of using softmax as weak classifier is not dissatisfactory, we guess, is the number of estimators is too small, we didn't build much estimators in order to reduce the running time, but this algorithm is depending on enough weak classifiers. That's why the performance is far away from using single softmax classifier.

Compared with using decision tree, because the decision tree is fast and if we set the tree with short depth and increase the number of estimators, the performance returned normal with 83 percent correctness rate using 5 minutes

5.1 Meaningful future work suggested

In this test, we haven't tested any algorithm from deep-learning, In the future work, we may introduce some new method like Neural Network to solve the classification problem from different respect, Artificial neural network is a simulation and approximation of biological neural networks. It is an adaptive nonlinear dynamic network system composed of a large number of neurons connected through each other.

The basic structure of CNN consists of an input layer, a convolution layer, a sampling layer, a fully connected layer, and an output layer. There are generally several convolutional and sampling layers, which are alternately arranged with a convolutional layer and a sampling layer, that is, one convolutional layer is connected to one sampling layer, the sampling layer is connected to a convolutional layer, and so on.

In recent years, CNN's excellent features such as weight sharing, less trainable parameters, and strong robustness have attracted the attention of many researchers. CNN reduces the number of weights required for training by weight sharing, reduces the computational complexity of the network, and makes the network's local transformation to the input have certain invariance such as translation invariance and scaling invariance through pooling operations. Improve the generalization ability of the network.

References:

[1] U. M. Sanghamitra Bandyopadhyay, “Genetic clustering for automatic evolution of clusters and application to image classification,” 2000.

[2] A. Bosch, A. Zisserman annd X. Munoz, “Image Classification using Random Forests and Ferns,” 2007 IEEE 11th International Conference on Computer Vision, 10 2007.

[3] D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella 和 J. ¨. Schmidhuber, “Flexible, High Performance Convolutional Neural Networks for Image Classification,” pp. 1237-1242, 7 2011.

[4] Y. M. Cha Zhang, “Ensemble Machine Learning,” Springer, 2012.

[5] K. B. I. Usama M. Fayyad, “On the handling of continuous-valued attributes in decision tree generation,” Machine Learning, Vol. 8, no. 1, pp. 87-102, January 1992.

[6] S. A.Bejan, “Constructal tree-shaped flow structures,” Applied Thermal Engineering, Vol. 27, no. 4, March 2007.

[7] Z. W. Changming Zhu, “Entropy-based matrix learning machine for imbalanced data sets,” Pattern Recognition Letters, Vol. 88, pp. 72-80, March 2017.

[8] J. J. S. N. T. B. Abdullah Akce, “An SSVEP-Based Brain–Computer Interface for Text Spelling With Adaptive Queries That Maximize Information Gain Rates,” IEEE Transactions on Neural Systems and Rehabilitation Engineering, Vol. 23, no. 5, 2015.

[9] J. K. A. B. B. B. G. K. S. W. M. A. Michael Kommenda, “Optimization Networks for Integrated Machine Learning,” International Conference on Computer Aided Systems Theory, pp. 392-399, January 2018.

[10] C. Soguero-Ruiz, F. J. Gimeno-Blanes, I. Mora-Jiménez, M. P. Martínez-Ruiz 和 J. L. Rojo-Álvarez, “Deal effect curve and promotional models: Using machine learning and bootstrap resampling test,” Vol. 2, pp. 537-540, 2012.

[11] H. Ishwaran, “The effect of splitting on random forests,” Machine Learning, Vol. 99, no. 1, pp. 75-118, April 2015.

[12] Z. L. Wang Jianhua, “Fault detection for sensor of network control system by dual coordinate descent margin-based SVM classier,” Vol. 10, no. 6, pp. 2257-7447.

[13] A. L. Andreas H. Hamel, “Lagrange Duality in Set Optimization,” Journal of Optimization Theory and Applications, Vol. 161, no. 2, pp. 368-397, 2013.

[14] B. S. a. A. J. S. Thomas Hofmann, “Kernel Methods in Machine Learning,” Vol. 36, no. 3, pp. 1171-1220, 2008.

[15] Z. X. S. X. H. T. J. Yueguo Luo, “Classification noise detection based SMO algorithm,” Optik - International Journal for Light and Electron Optics, Vol. 127, no. 17, pp. 7021-7029, 2016.

[16] D. R. E. E. D. E. S. Mustafa A. Kocak, “SafePredict: A Meta-Algorithm for Machine Learning That Uses Refusals to Guarantee Correctness,” 2017.

[17] M. Q.-G. L. J.-C. G. L. Cao Ying, “Advance and prospects of AdaBoost algorithm,” Vol. 39, no. 6, pp. 745-758, 2013.

[18] C. D. L. Yu Wang, “Learning by Bagging and Adaboost based on Support Vector Machine,” June 2007.

[19] S. K. R. W. Jingjing Cao, “A noise-detection based AdaBoost algorithm for mislabeled data,” Pattern Recognition, Vol. 45, no. 12, pp. 4451-4465, December 2012.

[20] L. G. L. Y. W. C. W. X. Hu Jinhai, “AdaBoost algorithm for multi-class classification based on exponential loss function and its application,” Vol. 29, no. 4, pp. 811-816, 2008.

[21] P. K. P. A. Warren Hoburg, “Data fitting with geometric-programming-compatible softmax functions,” Optimization and Engineering, Vol. 17, no. 4, pp. 897-918, December 2016.

[22] G. Z. G. R. Xiaofeng Lv, “Gini index estimation for lifetime data,” Lifetime Data Analysis, Vol. 23, no. 2, pp. 275-304, January 2016.