Guides to a Series of Experiments for “Pattern Recognition and Machine Learning”

School of Automation Science and Electrical Engineering

Contents

Experiment 1 Perceptron Learning toward Linear Classification

1.1 Introduction

Linear perceptron is one of the simplest learning algorithms for a two-class classifier. Given a set of data points in d-dimensions, belonging to two classes, ω_1 and ω_2, the algorithm tries to find a linear separating hyper-plane between the samples of the two classes. If the samples are in one, two or three dimensions, the separating hyperplane would be a point, line or a plane respectively. The specific algorithm that we look into is a special case of a class of algorithms that uses gradient descent on a carefully defined objective function to arrive at a solution.

1.2 Principle and Theory

Assume that the samples of the two classes are linearly separable in the feature space. i.e., there exists a plane $\mathrm{G(X)} = \mathrm{W^T X} + \mathrm{w_{n+1}} = 0$, where $\mathrm{W} \in \mathrm{R^n}$ and $\mathrm{X} \in \mathrm{R^n}$, such that all samples belonging to the first class are on one side of the plane, and all samples of the second class are on the opposite side. If such planes exist, the goal of the perceptron algorithm is to learn any one such plane, given the data points. Once the learning is completed and the plane is determined, it will be easy to classify new points in the future, as the points on one side of the plane will result in a positive value for $\mathrm{G(X)} = \mathrm{W^T X} + \mathrm{w_{n+1}}$, while points on the other side will give a negative value.

According to the principle of perceptron learning, the weight vector $\mathrm{W} \in \mathrm{R^n}$ can be extended to $\widehat{\mathrm{W}} = (\mathrm{w_1\, w_2 \cdots w_n\ w_{n+1}})^\mathrm{T} \in \mathrm{R^{n+1}}$ and the feature vector may be extended to $\widehat{\mathrm{X}} = (\mathrm{x_1\, x_2 \cdots x_n\ 1})^\mathrm{T} \in \mathrm{R^{n+1}}$ also, thus the plane of classification can be expressed as $G(X) = \widehat{W}^{T}\widehat{X} = 0$. The learning rule (algorithm) for the update of weights is designed as

$$\widehat{\mathbf{W}}(\mathbf{t}+\mathbf{1}) = \widehat{\mathbf{W}}(\mathbf{t}) + \frac{\mathbf{1}}{\mathbf{2}}\boldsymbol{\eta}\{\widehat{\mathbf{X}}(\mathbf{t}) - \widehat{\mathbf{X}}(\mathbf{t})\mathbf{sgn}[\widehat{\mathbf{W}}^{\mathbf{T}}(\mathbf{t})\widehat{\mathbf{X}}(\mathbf{t})]\}$$

$$= \begin{cases} \mathbf{W}(\mathbf{t}) & \widehat{W}^{T}(t)\widehat{X}(t) > 0 \\ \mathbf{W}(\mathbf{t}) + \boldsymbol{\eta}\widehat{\mathbf{X}}(\mathbf{t}) & \text{otherwise} \end{cases}$$

where $\boldsymbol{\eta}$ is the learning rate which may be adjusted properly to improve the convergence efficiency of the learning course.

1.3 Objective

The goals of the experiment are as follows:

(1) To understand the working of linear perceptron learning algorithm.

(2) To understand the effect of various parameters on the learning rate and convergence of the algorithm.

(3) To understand the effect of data distribution on learnability of the algorithm.

1.4 Contents and Procedure

Stage 1:

(1) According to the above principle and theory in section 1.2, design and compile the programme codes of perceptron learning toward the linear classification of two classes.

(2) Create a linearly separable pattern dataset with more than 50 samples for each one of two classes.

(3) Initialize the weight vector $\widehat{W}(0)$ and select a proper value for the learning rate $\boldsymbol{\eta} \in (\mathbf{0}, \mathbf{1}]$.

(4) Run your programme with your dataset and record the final result, pay attention to the number of iterations for convergence.

Stage 2:

Repeat the above procedure (2) ~ (4) in stage 1 for the different datasets with varying amount of separation between two classes of patterns. Note down your observations.

Stage 3:

Study the effect of varying the learning rate $\boldsymbol{\eta}$ with different amounts of separability.

Stage 4：

Explore the questions in the previous section and design experiments to answer these questions. Complete and submit an experiment report about all experiment results with comparative analysis and a summary of experiences in this experiment study.

Experiment 2　Synthetical Design of Bayesian Classifier

2.1　Introduction

Linear perceptrons allow us to learn a decision boundary that would separate two classes. They are very effective when there are only two classes, and they are well separated. Such classifiers are referred to as discriminative classifiers.

In contrast, generative classifiers consider each sample as a random feature vector, and explicitly model each class by their distribution or density functions. To carry out the classification, the likelihood function should be computed for a given sample which belongs to one of candidate classes so as to assign the sample to the class that is most likely. In other words, we need to compute $p(\omega_i|X)$ for each class ωi. However, the density functions provide only the likelihood of seeing a particular sample, given that the sample belongs to a specific class. i.e., the density functions can be provided as $p(X|\omega_i)$. The Bayesian rule provides us with an approach to compute the likelihood of the class for a given sample, from the density functions and related information.

2.2　Principle and Theory

The essence of the Bayesian approach is to provide a mathematical rule explaining how you should change your existing beliefs in the light of new evidence. In other words, it allows us to combine new data with their existing knowledge or expertise. The canonical example is to imagine that a precocious newborn observes his first sunset, and wonders whether the sun will rise again or not. He assigns equal prior probabilities to both possible outcomes, and represents this by placing one white and one black marble into a bag. The following day, when the sun rises, the child places another white marble in the bag. The probability that a marble plucked randomly from the bag will be white (i.e., the child's degree of belief in future sunrises) has thus gone from a half to two-thirds. After sunrise the next day, the child adds another white marble, and the probability (and thus the degree of belief) goes from two-thirds to three-quarters. And so on. Gradually, the initial belief that the sun is just as likely as not to rise each morning is modified to become a near-certainty that the sun will always rise.

In terms of classification, the Bayesian theorem allows us to combine prior probabilities, along with observed evidence to arrive at the posterior probability. More or less, conditional probabilities represent the probability of an event occurring given evidence. According to the Bayesian Theorem, if $P(\omega_i)$, $P(X|\omega_i)$, $\mathrm{i} = 1,2, \cdots, \mathrm{c}$, and X are known or given, the posterior probability can be derived as follows

$$P(\omega_i|X) = \frac{P(X|\omega_i)P(\omega_i)}{\sum_{j=1}^{c} P(X|\omega_j)P(\omega_j)} \qquad \text{i=1,…, c} \qquad (1)$$

Let the series of decision actions as $\{\mathrm{a}_1, \mathrm{a}_2, \cdots, \mathrm{a}_\mathrm{c}\}$, the conditional risk of decision action a_i can be computed by

$$R(a_i|X) = \sum_{j=1, j\neq i}^{c} \lambda(a_i, \omega_j)P(\omega_j|X), \qquad \text{i=1,…, c} \qquad (2)$$

Thus the minimum risk Bayesian decision can be found as

$$a_\mathrm{k}^* = \mathrm{Arg\ min_i}\ R(a_i|X), \qquad \text{i=1,…, c} \qquad (3)$$

2.3 Objective

The goals of the experiment are as follows:

(1) To understand the computation of likelihood of a class, given a sample.

(2) To understand the use of density/distribution functions to model a class.

(3) To understand the effect of prior probabilities in Bayesian classification.

(4) To understand how two (or more) density functions interact in the feature space to decide a decision boundary between classes.

(5) To understand how the decision boundary varies based on the nature of density functions.

2.4 Contents and Procedure

Stage 1:

(1) According to the above principle and theory in section 2.2, design a Bayesian classifier for the classification of two classes of patterns which are subjected to Gaussian normal distribution and compile the corresponding programme codes.

(2) In view of the normal cell class ω_1 , the corresponding data of sample features are extracted as

Ω_1={ -3.9847，-3.5549，-1.2401，-0.9780，-0.7932，-2.8531，-2.7605，-3.7287，-3.5414，-2.2692，-3.4549，-3.0752，-3.9934， -0.9780，-1.5799，-1.4885，-0.7431，-0.4221，-1.1186，-2.3462，-1.0826，-3.4196，-1.3193，-0.8367，-0.6579，-2.9683},

and the sample features of abnormal cell class ω_2 are listed as

Ω_2= { 2.8792， 0.7932，1.1882，3.0682，4.2532，0.3271,0.9846,2.7648,2.6588}

The prior probabilities of both ω_1 and ω_2 are known as

$p(\omega_1)$ = 0.9, $p(\omega_2)$ = 0.1

The loss parameters for different decision action are given as table 1

Table 1 the loss parameters for different decision

real class		ω_1	ω_2	
decision action	a_1	0	1	loss parameters
	a_2	6	0	

Suppose the conditional probability distributions are Gaussian, find the conditional probability density functions $p(X|\omega_1)$ and $p(X|\omega_2)$ and complete the design of Bayessian classifier with minimum risk, and then give a comparative analysis with the situation without considering decision loss.

Draw the curves of prior and posterior probability density functions, $p(X|\omega_1)$, $p(X|\omega_2)$, $p(\omega_1|X)$, and $p(X|\omega_2)$, give the classifying decision boundary function and illustration of classification result.

Stage 2 (towards an in depth study):

(1) Create a pattern dataset of multiple classes and high dimension with more than 50 samples for each class. Then design a Bayesian classifier and complete the corresponding experiments and comparative analysis by using self-supposed prior probabilities and loss parameters for the same terms as stage 1.

(2) Think and analyse the intrinsic relationship between the classifier of two classes and the one of multiple classes, give your comments.

Stage 3：

Explore the questions in the previous section and design experiments to answer these questions. Complete and submit an experiment report about all experiment results with comparative analysis and a summary of experiences about this experiment study.

Experiment 3 Decision Tree Learning for Classification

3.1 Introduction

Decision tree induction is one of the simplest and yet most successful learning algorithms. A decision tree (DT) consists of internal and external nodes and the interconnections between nodes are called branches of the tree. An internal node is a decision-making unit to decide which child nodes to visit next depending on different possible values of associated variables. In contrast, an external node also known as a leaf node, is the terminated node of a branch. It has no child nodes and is associated with a class label that describes the given data. A decision tree is a set of rules in a tree structure, each branch of which can be interpreted as a decision rule associated with nodes visited along this branch.

3.2 Principle and Theory

Decision trees classify instances by sorting them down the tree from root to leaf nodes. This tree-structured classifier partitions the input space of the data set recursively into mutually exclusive spaces. Following this structure, each training data is identified as belonging to a certain subspace, which is assigned a label, a value, or an action to characterize its data points. The decision tree mechanism has good transparency in that we can follow a tree structure easily in order to explain how a decision is made. Thus interpretability is enhanced when we clarify the conditional rules characterizing the tree.

Entropy of a random variable is the average amount of information generated by observing its value. Consider the random experiment of tossing a coin with probability of heads equal to 0.9, so that P(Head) = 0.9 and P(Tail) = 0.1. This provides more information than the case where P(Head) = 0.5 and P(Tail) = 0.5.

Entropy is used to evaluate randomness in physics, where a large entropy value indicates that the process is very random. The decision tree is guided heuristically according to the information content of each attribute. Entropy is used to evaluate the information of each attribute; as a means of classification. Suppose we have *m* classes, for a particular attribute, we denoted it by pi by the proportion of data which belongs to class C_i where *i = 1, 2, ... m.*

The entropy of this attribute is then:

$$Entropy = \sum_{i=1}^{m} -p_i \cdot \log_2 p_i$$

We can also say that entropy is a measurement of the impurity in a collection of training examples: larger the entropy, the more impure the data is. Based on entropy, Information Gain (IG) is used to measure the effectiveness of an attribute as a means of discriminating between classes.

$$IG(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

where all examples *S* is divided into several groups (i.e. S_v for $v \in Values(A)$) according to the value of *A*. It is simply the expected reduction of entropy caused by partitioning the examples according to this attribute.

3.3 Objective

The goals of the experiment are as follows:

(1) To understand why we use entropy-based measure for constructing a decision tree.

(2) To understand how Information Gain is used to select attributes in the process

of building a decision tree.

(3) To understand the equivalence of a decsion tree to a set of rules.

(4) To understand why we need to prune the tree sometimes and how can we prune? Based on what mesure we prune a decsion tree.

(5) To understand the concept of Soft Decsion Treees and why they are imporant extensions to classical decision trees.

3.4 Contents and Procedure

Stage 1:

(1) According to the above principle and theory in section 3.2, implement the code to calculate the information entropy of each attribute.

(2) Select the most informative attribute from a given classification problem (e.g., we will be given the Iris Dataset from the UCI Machine Learning Repository)

(3) Find an appropriate data structure to represent a decion tree. Building the tree from the root to leaves based on the principal discussed in section 3.2 by using Information Gain guided heuritics.

Stage 2:

(1) Now consider the case of with continuous attributes or mixed attributes (with both continuous and discrete attributes), how can we deal with the decision trees? Can you propose some approaches to do discretization?

(2) Is there a tradeoff between the size of the tree and the model accuracy? Is there existing an optimal tree in both compactness and performance?

(3) For one data element, the classical decsion tree gives a hard bounday to

decide which branch to follow, can you propose a “soft approach” to increase the robustness of the decision tree?

(4) Compare to the Naïve Bayes, what are the advantages and disvantages of the decision tree learning?

Stage 3：

Explore the questions in the previous section and design experiments to answer these questions. Complete and submit an experiment report about all experiment results with comparative analysis and a summary of experiences about this experiment study.

Experiment 4 Neural Networks and Back Propagation

4.1 Introduction

The learning model of Artificial Neural Networks (ANN) (or just a neural network (NN)) is an approach inspired by biological neural systems that perform extraordinarily complex computations in the real world without recourse to explicit quantitative operations. The original inspiration for the technique was from examination of bioelectrical networks in the brain formed by neurons and their synapses. In a neural network model, simple nodes (called variously “neurons” or “units”) are connected together to form a network of nodes, hence the term “neural network”.

Each node has a set of input lines which are analogous to input synapses in a biological neuron. Each node also has an “activation function” that tells the node when to fire, similar to a biological neuron. In its simplest form, this activation function can just be to generate a ‘1’ if the summed input is greater than some value, or a ‘0’ otherwise. Activation functions, however, do not have to be this simple - in fact to create networks that can do useful things, they almost always have to be more complex, for at least some of the nodes in the network. Typically there are at least three layers to a feed-forward network - an input layer, a hidden layer, and an output layer. The input layer does no processing - it is simply where the data vector is fed into the network. The input layer then feeds into the hidden layer. The hidden layer, in turn, feeds into the output layer. The actual processing in the network occurs in the nodes of the hidden layer and the output layer.

4.2 Principle and Theory

The goal of any supervised learning algorithm is to find a function that best maps a set of inputs to its correct output. An example would be a simple classification task, where the input is an image of an animal, and the correct output would be the name of the animal. For an intuitive example, the first layer of a Neural Network may be responsible for learning the orientations of lines using the inputs from the individual pixels in the image. The second layer may combine the features learned in the first layer and learn to identify simple shapes such as circles. Each higher layer learns more and more abstract features such as those mentioned above that can be used to classify the image. Each layer finds patterns in the layer below it and it is this ability to create internal representations that are independent of outside input that gives multi-layered networks their power. The goal and motivation for developing the back-propagation algorithm was to find a way to train a multi-layered neural network such that it can learn the appropriate internal representations to allow it to learn any arbitrary mapping of input to output.

Mathematically, a neuron's network function *f(x)* is defined as a composition of other functions $g_i(x)$ which can further be defined as a composition of other functions. This can be conveniently represented as a network structure, with arrows depicting the dependencies between variables. A widely used type of composition is the nonlinear weighted sum, where:

$$f(x) = K\left(\sum_i w_i g_i(x)\right)$$

where *K* (commonly referred to as the activation function) is some predefined function, such as the hyperbolic tangent. It will be convenient for the following to refer to a collection of functions g_i as simply a vector $g = (g_1, g_2, \ldots, g_n)$. Back-propagation requires a known, desired output for each input value in order to calculate the loss function gradient. It is therefore usually considered to be a supervised learning method

The squared error function is:

$$E = \frac{1}{2}(t - y)^2$$

where E is the squared error, t is the target output for a training sample, and

y is the actual output of the output neuron. For each neuron j, its output o_j is defined as

$$o_j = \varphi(net_j) = \varphi\left(\sum_{k=1}^{n} w_{kj}x_k\right)$$

The input net to a neuron is the weighted sum of outputs o_k of previous neurons. If the neuron is in the first layer after the input layer, the o_k of the input layer are simply the inputs x_k to the network. The number of input units to the neuron is n. The

variable w_{ij} denotes the weight between neurons i and j.

The activation function φ is in general non-linear and differentiable. A commonly used activation function is the logistic function, e.g.:

$$\varphi(z) = \frac{1}{1 + e^{-z}}$$

which has a nice derivative of:

$$\frac{\partial \varphi}{\partial z} = \varphi(1 - \varphi)$$

Calculating the partial derivative of the error with respect to a weight w_{ij} is done using the chain rule twice:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}}$$

We can finally yield:

$$\frac{\partial E}{\partial w_{ij}} = \delta_j x_i$$

with

$$\delta_j = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial net_j} = \begin{cases} (o_j - t_j)\varphi(net_j)\left(1 - \varphi(net_j)\right) & if\ j\ is\ an\ output\ neuron \\ (\sum_{l \in L} \delta_l w_{jl})\varphi(net_j)\left(1 - \varphi(net_j)\right) & if\ j\ is\ an\ inner\ neuron \end{cases}$$

4.3 Objective

The goals of the experiment are as follows:

(1) To understand how to build a neural network for a classification problem.

(2) To understand how the back-propagation algorithm is used for training a given a neural network.

(3) To understand the limitation of the neural network model (e.g., the local minimum).

(4) To understand how to use back-propagation in Autoencoder.

4.4 Contents and Procedure

Stage 1：

(1) Given a dataset for classification, (E.g., Iris, Pima Indian and Wisconsin Cancer from the UCI ML Repository). Build a multi-layer neural network

(NN) and train the network using the BP algorithm. We can start with constructing a NN with only one hidden layer.

(2) Compare the NN results to the results of Perceptron, which model is better in terms of efficiency and accuracy?

(3) Whether the performance of the model is heavily influenced by different parameters settings (e.g., the learning rate α)?

Stage 2：

(1) How the efficiency and accuracy will be influenced by different activation functions and more hidden layers.

(2) Do you think that biological neural networks work in the same way as our NN model? Can you provide any discussions to support your opinions?

(3) Using the BP algorithm to train an autoencoder.

(We will train the simplest autoencoder which is a feedforward, non-recurrent neural net with an input layer, an output layer and one or more hidden layers connecting them. The difference is that, for an autoencoder, the output layer has equally many nodes as the input layer, and instead of training it to predict some target value y given inputs x, an autoencoder is trained to reconstruct its own inputs x.

(4) For an autoencoder with only one hidden layer, how the number of nodes in the hidden layer influence the model performance?

Stage 3：

Explore the questions in the previous section and design experiments to answer these questions. Complete and submit an experiment report about all experiment results with comparative analysis and a summary of experiences about this experiment study.