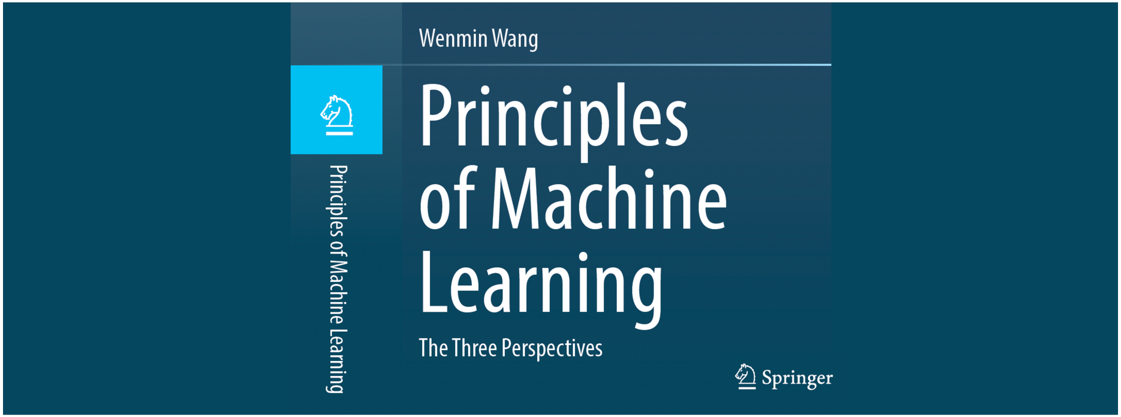


Wenmin Wang

Principles of Machine Learning

The Three Perspectives

Springer

Principles of Machine Learning

https://link.springer.com/book/10.1007/978-981-97-5333-8

Part IV Tasks

12 Classification Task

12 Classification Task

Problem and Definition

Classification is the process of categorizing something.

Definition: (Classification)

Classification in machine learning is the task that trains a classifier with the labeled samples whose categories are known, and then uses it to identify which categories the other data belongs to.

The three elements of classification: 1) identifying data into categories; 2) the categories are known; and 3) training with samples.

In short, classification is identifying data into known categories by training.

12 Classification Task

Formal Description

Let $\mathcal{X} \subseteq \mathbb{R}^m$ be input space, and $\mathcal{C} = \{c_j \mid j = 1, \ldots, k\}$ be output space consisting of k known categories.

Through $P(\boldsymbol{x})$ we obtain n independent and identically distributed (i.i.d.) data:

$$D = \{\boldsymbol{x}_i \mid \boldsymbol{x}_i \in \mathcal{X} \text{ and } i = 1, \ldots, n\}.$$

Let the target function of classification $f : \mathcal{X} \to \mathcal{C}$ be $P(c|\boldsymbol{x})$, where $\boldsymbol{x} \in \mathcal{X}$, and $c \in \mathcal{C}$.

Based on $P(\boldsymbol{x}, c) = P(c|\boldsymbol{x})\, P(\boldsymbol{x})$, each data $\boldsymbol{x}_i$ is labeled with its target category c_j by $P(c|\boldsymbol{x})$, resulting in a set of training samples S with n elements, where each element is represented as a data pair $(\boldsymbol{x}_i, c_j) \in \mathcal{X} \times \mathcal{C}$, i.e.,

$$S = \{(\boldsymbol{x}_i,\ c_j) \mid i = 1, \ldots, n \ \text{ and } \ j = 1, \ldots, k\}.$$

Formal Description

Let $H : \mathcal{X} \to \mathcal{C}$ be a set of hypothesis functions for classification. The goal is to obtain $h \in H$ through training on S, and $h(\boldsymbol{x}) = \hat{c}$ has small expected risk with $f(\boldsymbol{x}) = c$:

$$\underset{h \in H}{\arg\min}\, R(h) = \underset{h \in H}{\arg\min}\, \mathbb{E}\left[\mathcal{L}\left(h(\boldsymbol{x}), f(\boldsymbol{x})\right)\right] = \underset{h \in H}{\arg\min}\, \mathbb{E}\left[\mathcal{L}\left(\hat{c}, c\right)\right],$$

where $\hat{c}$ is predicted category, c is target category, and $\mathcal{L}(\cdot, \cdot)$ is loss function.

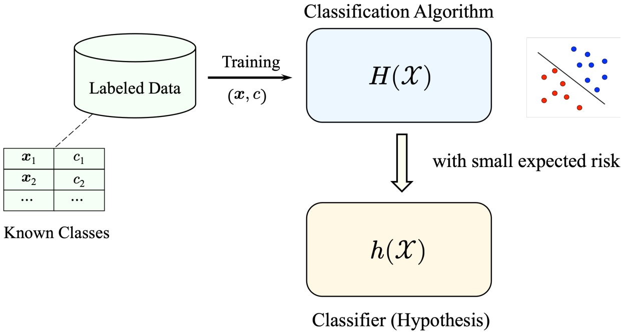
Given n' test data $\boldsymbol{x}_i \in \mathcal{X}$ with unknown categories:

$$T = \left\{\boldsymbol{x}_i \mid i = 1, \ldots, n'\right\} \subseteq \mathcal{X}.$$

Use $h(\boldsymbol{x})$ to classify the n' test data: $h(\boldsymbol{x}_i) = \hat{c}_j \in \mathcal{C}$, the result of classification is:

$$T_{\text{Output}} = \left\{(\boldsymbol{x}_i, \hat{c}_j) \mid i = 1, \ldots, n' \text{ and } j = 1, \ldots, k\right\}.$$

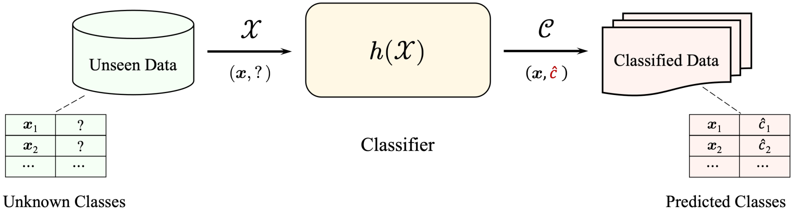
Illustrated Description



Labeled dataset $\{(\boldsymbol{x}, c)\}$ are used to train H, and $h(\mathcal{X}) \in H$ is obtained, which satisfies small expected risk between $h(\boldsymbol{x})$ and labeled category c.

Illustrated Description

The classifier hypothesis $h(\boldsymbol{x})$ is used for some input data where the category is unknown (also called unseen data) to get the predicted classification result $\hat{c}$.



Classification Functions

Probabilistic Classification: Its classification result is the probability of which category the unknown data belongs to.

Let $\boldsymbol{x} \in \mathcal{X}$ be input data, $\mathcal{C} = \{c_j \mid j = 1, \ldots, k\}$ be known categories, then probabilistic classification function is:

$$h_j(\boldsymbol{x}) = \arg\max_j P\left(c = c_j | \boldsymbol{x}\right).$$

In other words, probabilistic classification first measures the probability that the input data belongs to, and then obtains the category with highest probability.

In machine learning, the algorithms such as logistic regression, naive Bayes, and softmax use probabilistic classification functions.

Classification Functions

Statistical classification: Its result of classification is what category the unknown data belongs to.

Let $\boldsymbol{x} \in \mathcal{X}$ be input data, $\mathcal{C} = \{c_j \mid j = 1, \ldots, k\}$ be known categories, $\boldsymbol{w}_j$ be weight, and b denote bias, then the statistical classification function is:

$$h_j(\boldsymbol{x}) = \arg\max_j c\left(\boldsymbol{w}_j^\intercal \cdot \boldsymbol{x} + b\right),$$

where the $c(\cdot)$ is a linear or nonlinear function for statistical classification.

In machine learning, the algorithms such as linear discriminant analysis, k-nearest neighbor, and support vector machine (SVM) use statistical classification functions.

12 Classification Task

One-Class Classification

One-class classification (OCC), is also known as unary classification.

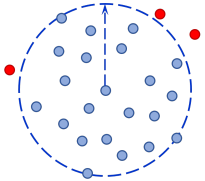
Let $f : \mathcal{X} \to \mathcal{C}$ be the OCC function, then $\mathcal{C} = \{c_j \mid j = 1\}$.

OCC is mainly used to identify the data points on the outside a specific category in the input data.

Its corresponding subtasks include outlier detection, anomaly detection, and novelty detection.

Classic methods of OCC include:

- Density methods, such as the Gaussian model, and Parzen density estimation;
- Boundary methods, such as k-center algorithm.



Two-Class Classification

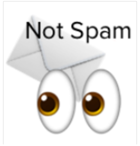
Two-class classification (TCC), is also known as binary classification.

Let $f : \mathcal{X} \to \mathcal{C}$ be the TCC function, then $\mathcal{C} = \{c_j \mid j = 1, 2\}$.

TCC is used to predict whether input data has a certain specific property and qualitative characteristics.

Its applications include email detection (spam or non-spam), disease diagnosis (the disease or not), and credit fraud detection (fraud or legal).

Commonly used algorithms of TCC include logistic regression, k-nearest neighbors, decision trees, support vector machine (SVM), and naive Bayes.



Multi-Class Classification

Multi-class classification (MCC) is the method of dividing unknown input data into multiple categories through training.

Let $f : \mathcal{X} \to \mathcal{C}$ be the MCC function, then $\mathcal{C} = \{c_j \mid j = 1, ..., k\}$ and $k > 2$.

There are several ways to solve MCC problems, such as:

1) transforming MCC into existing TCC;
2) extending existing TCC to MCC;
3) hierarchical classification.

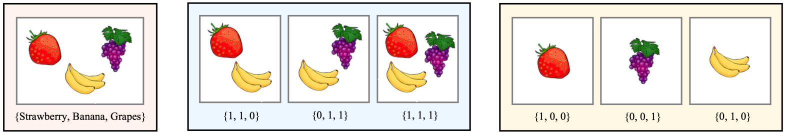
The softmax function is an effective method for solving MCC.

Multi-label Classification

Multi-label classification (MLC) is a variant of classification problems, for each unknown data, two or more class labels may be predicted.

Let $f : \mathcal{X} \to \{0, 1\}^{\mathcal{C}}$ be the MLC function, then $\mathcal{C} = \{c_j \mid j = 1, \ \ldots, k\}$ and $k > 1$.

Therefore, MLC is to construct a model that maps $\boldsymbol{x} \in \mathcal{X}$ to $y \in \{0, 1\}^{\mathcal{C}}$.



(a) Samples (b) Multi-Label (c) Multi-Class

MLC algorithms include multi-label k-nearest neighbor, multi-label decision tree, and ANN. Ensemble learning can combine several MCC into a MLC.

Imbalanced Classification

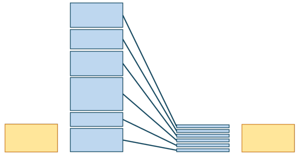
Imbalanced classification is the classification of imbalanced data.

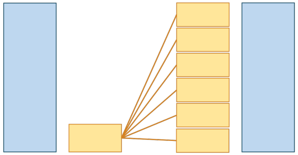
The imbalanced data refers to a dataset where the distribution of categories is extremely imbalanced.

E.g., the ratio of data between two categories is 1:10, 1:100, or even 1:1000.

An effective way is to use resampling techniques to transform imbalanced data into balanced one.

1) Down-sampling for data with a high ratio.
2) Up-sampling for data with a low ratio.
3) A combination of down- and up-sampling.


(a) Samples of majority class


(b) Copies of minority class

12 Classification Task

Linear and Nonlinear

Linear Separability: It refers to the data points in binary classification problems which can be separated using linear decision boundary.

Definition: (Linear Separable)

Let $\mathbb{R}^m$ be a set of m-dimensional real-valued vectors, $\mathcal{X} \subseteq \mathbb{R}^m$ be an input space, $D \subset \mathcal{X}$ and $D' \subset \mathcal{X}$ be two sets of data points, $\boldsymbol{w} \in \mathbb{R}^m$, and $b \in \mathbb{R}$. Then D and D' are called linear separable, if there exists a linear function $u(\boldsymbol{x}) = \boldsymbol{w}^\intercal \boldsymbol{x} - b$ such that every $\boldsymbol{x} \in D$ satisfies $u(\boldsymbol{x}) = \boldsymbol{w}^\intercal \boldsymbol{x} - b > 0$ and every $\boldsymbol{x}' \in D'$ satisfies $u(\boldsymbol{x}') = \boldsymbol{w}^\intercal \boldsymbol{x}' - b < 0$.

Mathematically, D and D' are two closed convex sets, and $D \cap D' = \emptyset$.

Linear Classification

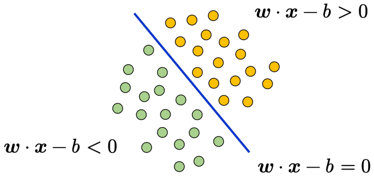
If the data is linearly separable, it can be classified using linear classification, the function of which is called linear classification function.

A linear classification for two-class classification can be expressed as:

$$f(\boldsymbol{x}) = \operatorname{sign}\left(u(\boldsymbol{x})\right) = \operatorname{sign}(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x} - b),$$

where $\operatorname{sign}(\cdot)$ is a sign function. And the $f(\boldsymbol{x})$ is defined as follows:

$$f(\boldsymbol{x}) = \begin{cases} +1 & \boldsymbol{w}^{\mathsf{T}}\boldsymbol{x} - b > 0 \\ 0 & \boldsymbol{w}^{\mathsf{T}}\boldsymbol{x} - b = 0 \\ -1 & \boldsymbol{w}^{\mathsf{T}}\boldsymbol{x} - b < 0. \end{cases}$$


$\boldsymbol{w} \cdot \boldsymbol{x} - b > 0$
$\boldsymbol{w} \cdot \boldsymbol{x} - b < 0$
$\boldsymbol{w} \cdot \boldsymbol{x} - b = 0$

Nonlinear Classification

If the data is nonlinearly separable, it needs to be classified by a nonlinear classification function that is a nonlinear combination of parameters:

$$f(\boldsymbol{x}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_m x^m = w_0 + \sum_{i=1}^{m} w_i x^i.$$

After normalizing, we get the following nonlinear classification expression:

$$f(\boldsymbol{x}) = w_0 + \sum_{i=1}^{m} w_i \phi_i(\boldsymbol{x}) = \boldsymbol{w}^\top \Phi(\boldsymbol{x}) + b.$$

Some nonlinear classification functions may have several nonlinear decision boundaries discontinuously.

Hard and Soft Classification

For the problem in medical data classification, given n samples:

$$S = \{(\boldsymbol{x}_i,\ c_j) \mid i = 1, \ldots, n \text{ and } j = 1, \ldots, k\},$$

where $\boldsymbol{x}_i$ is symptom of the disease, and c_j is category of the disease. .

Hard classification:

classification function $f_c(\boldsymbol{x})$ explicitly divides the new symptom $\boldsymbol{x}_i'$ into some disease class c_j.

Soft classification:

classification function $p(c|\boldsymbol{x})$ gives the probability of new symptom $\boldsymbol{x}_i'$ being a disease class c_j.

Lazy and Eager Classification

Lazy classification:

During training phase, it only stores the training data. Entering testing phase, it checks whether the input is most relevant object in training data, and then classifies it. Feature: less training time, but more time in classification. Example classifier: *k*-nearest neighbor.

Eager classification:

During the training phase, it undergoes rigorous debugging and training. Entering the testing phase, it directly classifies the input data. Feature: long time for training, but less time for classification. Example classifiers: decision trees, naive Bayes.

12 Classification Task

Logistic Regression

Logistic regression is a statistical classification method based on the relationship between independent and dependent variables.

Classification is essentially a regression problem with discrete output, the algorithm of logistic regression reflects this nature.

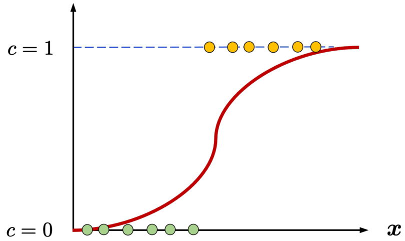
Let $\mathcal{X}$ denote input space, and $\mathcal{C} = \{0, 1\}$ denote output space. For $\boldsymbol{x} \in \mathcal{X}$, the logistic regression predicting its category $c \in \mathcal{C}$ is as follows:

$$P\left(c|\boldsymbol{x}\right) = \sigma\left(z\right) = \frac{1}{1 + e^{-z}},$$

where $\sigma\left(\cdot\right)$ is logistic function, also known as Sigmoid function.

Logistic Regression

Logistic regression has the following property: $P(c=0|\boldsymbol{x}) = 1 - P(c=1|\boldsymbol{x})$, because the probability distribution of a two-class classification should satisfy $P(c=0|\boldsymbol{x}) + P(c=1|\boldsymbol{x}) = 1$.



Let $z = \boldsymbol{w}^\top \boldsymbol{x} - b$, we have:

$$P(c=1|\boldsymbol{x}) = \sigma(\boldsymbol{w}^\top \boldsymbol{x} - b) = \frac{1}{1+e^{-(\boldsymbol{w}^\top \boldsymbol{x} - b)}},$$

$$P(c=0|\boldsymbol{x}) = 1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x} - b)$$

$$= 1 - \frac{1}{1+e^{-(\boldsymbol{w}^\top \boldsymbol{x} - b)}} = \frac{e^{-(\boldsymbol{w}^\top \boldsymbol{x} - b)}}{1+e^{-(\boldsymbol{w}^\top \boldsymbol{x} - b)}}.$$

Naive Bayes

Naive Bayes belongs to probability classification based on Bayes theorem.

Why called naive Bayes is its features should have strong independence.

Given input data $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathcal{X}$, output $\mathcal{C} = \{c_j \mid j = 1, \ \ldots, k\}$, and classifier $P\left(c_j|\boldsymbol{x}\right)$. According to Bayes theorem:

$$P\left(c_j|\boldsymbol{x}\right) = \frac{P\left(\boldsymbol{x}|c_j\right) P\left(c_j\right)}{P(\boldsymbol{x})} = \frac{P\left(\boldsymbol{x}|c_j\right) P\left(c_j\right)}{\sum_k P\left(\boldsymbol{x}|c_j\right) P\left(c_j\right)}.$$

And, using the chain rule of joint probability distribution, we have:

$$P\left(c_j, \boldsymbol{x}\right) = P\left(c_j, x_1, \ldots, x_n\right) = P\left(c_j\right) \prod_{i=1}^{n} P\left(x_i|x_{i-1}, \ldots, , c_j\right).$$

Naive Bayes

Based on the strong independence, $P\left(x_i|x_{i-1},...,x_1,c_j\right)=P\left(x_i|c_j\right)$, hence:

$$P(c_j,x_1,\ldots,x_n)=P\left(c_j\right)\prod_{i=1}^{n}P\left(x_i|c_j\right).$$

Therefore, the following proportional relationship exists, expressed as:

$$P\left(c_j|x_1,\ldots,x_n\right)\propto P\left(c_j\right)\prod_{i=1}^{n}P\left(x_i|c_j\right).$$

$$P\left(c_j|x_1,\ldots,x_n\right)=\frac{1}{Z}P\left(c_j\right)\prod_{i=1}^{n}P\left(x_i|c_j\right),\ \ \text{where}\ \ Z=P(\boldsymbol{x})=\sum_{j}P\left(\boldsymbol{x}|c_j\right)P\left(c_j\right).$$

Adaptive Boosting

Let $\mathcal{X}$ denote input space, and $c_i \in \mathcal{C} = \{-1, +1\}$ denote output space, training samples $S = \{(\boldsymbol{x}_i,\ c_i) \mid i = 1, \ldots, n\}$, where $\boldsymbol{x}_i \in \mathcal{X}$, and $c_i \in \mathcal{C} = \{-1, +1\}$.

Given T weak classifiers, $h_t : \mathcal{X} \rightarrow \mathcal{C}$, $(t = 1, \ldots, T)$.

Adaptive Boosting (AdaBoost) iteratively maintains a set of weighted distributions of training samples, $D = \{d_t\,(i) \mid t = 1, \ldots, T$ and $i = 1, \ldots, n\}$. $\forall i$, its initial is $d_1\,(i) = \frac{1}{n}$.

First, it calculates the weighted distribution error ϵ_t of the training process:

$$\epsilon_t = \sum_{i=1}^{n} d_t(i)\mathbb{I}(h_t(\boldsymbol{x}_i) \neq c_i).$$

Adaptive Boosting

Next, calculate its adaptive parameter:

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

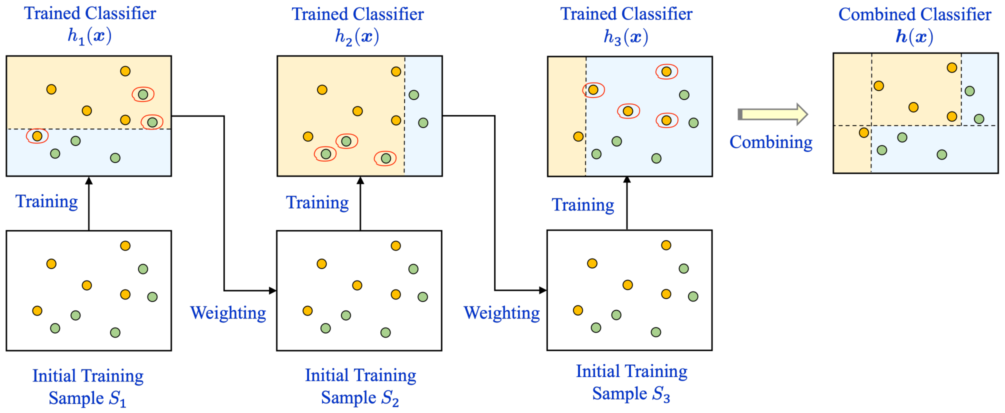
Then, $\forall i$, the weighted distribution is updated:

$$d_{t+1}(i) = \frac{1}{Z_t} d_t(i)\exp\left(-\alpha_t c_i h_t(\boldsymbol{x}_i)\right).$$

After T iterations, we obtain the following combined strong classifier:

$$\boldsymbol{h}_T(\boldsymbol{x}) = \operatorname{sgn}\left(\sum_{t=1}^{T}\alpha_t h_t(\boldsymbol{x})\right).$$

Adaptive Boosting



Trained Classifier $h_1(\boldsymbol{x})$
Trained Classifier $h_2(\boldsymbol{x})$
Trained Classifier $h_3(\boldsymbol{x})$
Combined Classifier $\boldsymbol{h}(\boldsymbol{x})$
Combining
Training
Training
Training
Weighting
Weighting
Initial Training Sample S_1
Initial Training Sample S_2
Initial Training Sample S_3

Schematic diagram of the AdaBoost algorithm.

Support Vector Machine

Support vector machine (SVM) is originally designed for linear separable data. Given a set of linearly separated training samples:

$$S = \{(\boldsymbol{x}_i,\ c_i) \mid i = 1, \ldots, n \text{ and } c_i \in \{-1, +1\}\},$$

A hyperplane $\mathcal{H}$ is a set of points that satisfy the following expression:

$$\mathcal{H} = \{\boldsymbol{x} \mid \boldsymbol{w}^\intercal \boldsymbol{x} - b = 0\}.$$

The parameter $\frac{b}{\|\boldsymbol{w}\|}$ is the offset of $\mathcal{H}$ from origin along normal vector $\boldsymbol{w}$. We can choose $\mathcal{H}_+$ and $\mathcal{H}_-$ parallel to $\mathcal{H}$ to distinguish the data:

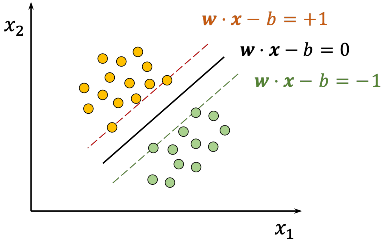
$$\begin{aligned} \mathcal{H}_+ &= \{\boldsymbol{x} \mid \boldsymbol{w}^\intercal \boldsymbol{x} - b = +1\}, \\ \mathcal{H}_- &= \{\boldsymbol{x} \mid \boldsymbol{w}^\intercal \boldsymbol{x} - b = -1\}. \end{aligned}$$

Support Vector Machine

Each data point on hyperplanes $\mathcal{H}_+$ or $\mathcal{H}_-$ is called the support vector, the distance between $\mathcal{H}_+$ and $\mathcal{H}_-$ is $\frac{2}{\|\boldsymbol{w}\|}$, called margin. Therefore, maximizing the margin is equivalent to minimizing $\|\boldsymbol{w}\|$.

Use the S to train the SVM, so that it satisfies following constraints:

$$\boldsymbol{w}^\top \boldsymbol{x}_i - b \geq +1, \text{ if } c_i = +1,$$
$$\boldsymbol{w}^\top \boldsymbol{x}_i - b \leq -1, \text{ if } c_i = -1.$$



x_2
$\boldsymbol{w} \cdot \boldsymbol{x} - b = +1$
$\boldsymbol{w} \cdot \boldsymbol{x} - b = 0$
$\boldsymbol{w} \cdot \boldsymbol{x} - b = -1$
x_1

The above constraints ensure that each data point is on correct side, can be written:

$$c_i \left(\boldsymbol{w}^\top \boldsymbol{x}_i - b\right) \geq +1, \ \ \forall i \in 1, \ldots, n.$$

Support Vector Machine

For nonlinear separable data, the SVM uses kernel function to map the data into a linear separable high-dimensional space, and then find the hyperplane with maximum margin for linear classification.

The kernel function introduced in Section 4.7 performs $\varphi\left(\boldsymbol{x}_i\right)$ transformation:

$$\kappa\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \left\langle \varphi\left(\boldsymbol{x}_i\right), \varphi\left(\boldsymbol{x}_j\right) \right\rangle = \varphi\left(\boldsymbol{x}_i\right) \cdot \varphi\left(\boldsymbol{x}_j\right)$$

The value of $\boldsymbol{w}$ is in the transformed space, and the inner product of $\boldsymbol{w}$ can be calculated using kernel trick.

$$\boldsymbol{w} = \sum_i a_i c_i \varphi\left(\boldsymbol{x}_i\right), \quad \boldsymbol{w} \cdot \varphi\left(\boldsymbol{x}_i\right) = \sum_{i=1}^{n} a_i c_i \kappa\left(\boldsymbol{x}_i, \boldsymbol{x}\right).$$

Artificial Neural Networks

For multi-class classification in ANNs, softmax function is one of effective methods.

The softmax function normalizes a vector with k real numbers into a probability distribution composed of k probability values.

That is to say, each real number in this vector is mapped to a value in the range of $\{0, 1\}$, and the sum of k values equals 1.

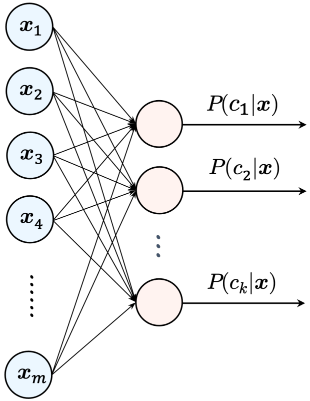
Let $\boldsymbol{c} = (c_1, \ldots, c_k) \in \mathbb{R}^k$, and $j = 1, \ldots, k$, then calculate the jth expression of the softmax function:

$$f_{c_j \in \boldsymbol{c}}(c_j) = \frac{e^{c_j}}{\sum_{i=1}^{k} e^{c_i}}.$$

Artificial Neural Networks

Subsequently, let $c_j = \boldsymbol{x}^\top \boldsymbol{w}_j$, and $\boldsymbol{x} \in \mathbb{R}^m$, and if softmax function is expressed in conditional probability, then it k different outcomes:

$$P_{c_j \in \mathcal{C}}\left(c_j | \boldsymbol{x}\right) = \frac{e^{\boldsymbol{x}^\top \boldsymbol{w}_j}}{\sum_{i=1}^{k} e^{\boldsymbol{x}^\top \boldsymbol{w}_i}}.$$



Softmax function also plays an important role in other multi-class classification algorithms, such as:

- multinormal logistic regression,
- multi-class linear discriminant analysis,
- multi-class naive Bayes.

12 Classification Task

About Loss Functions

A loss functions for classification is used to measure the error between the predicted class by classifier and the target class labeled in training samples.

We will introduce several commonly used loss functions for classification:

0-1 loss, logistic loss, exponential loss, hinge loss, and cross-entropy loss.

We will use following notations:

$\boldsymbol{x} \in \mathcal{X}$ denotes input data, $c \in \mathcal{C}$ denotes labeled target class, $h(\boldsymbol{x})$ denotes classification hypothesis, $\hat{c} = h(\boldsymbol{x})$ is predicted class, and loss function is denoted as $\mathcal{L}\left(h(\boldsymbol{x}), c\right) = \mathcal{L}\left(\hat{c}, c\right)$.

0-1 Loss

It is the most commonly used for two-class classification, defined as:

$$\mathcal{L}_{0-1}\left(\hat{c}, c\right) = \sum_{i=1}^{n} \mathbb{I}\left(h\left(\boldsymbol{x}_i\right) \neq c_i\right),$$

where $\mathbb{I}\left(\omega\right)$ is an indicator function, that equals 1 if ω is true, otherwise 0.

It means, if predicted class is different from target class, it is 1, otherwise 0.

For the two-class classification, the cost of false positives and false negatives is same, the 0-1 loss is naturally chosen as loss function.

However, the issue of 0-1 loss is non-differentiable, cannot be used for the methods such as gradient descent.

Logistic Loss

It is mainly used for the training of logistic regression classifiers.

The hypothesis of logistic regression is represented as $\hat{c} = P\left(h(\boldsymbol{x})|\boldsymbol{x}\right)$, the labeled target output is $c \in \mathcal{C} = \{0, 1\}$, then the expression of its logical loss function is as follows:

$$\mathcal{L}_{\text{Log}}\left(\hat{c}, c\right) = -\left(c \log\left(h(\boldsymbol{x})\right) + (1 - c)\log\left(1 - h(\boldsymbol{x})\right)\right).$$

That is:

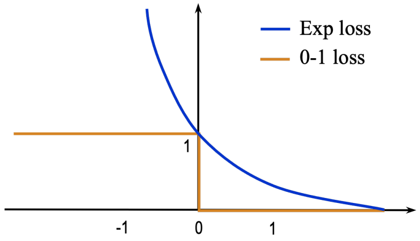
$$\mathcal{L}_{\text{Log}}\left(\hat{c}, c\right) = \begin{cases} -\log\left(h(\boldsymbol{x})\right) & \text{if } c = 1 \\ -\log\left(1 - h(\boldsymbol{x})\right) & \text{if } c = 0. \end{cases}$$

Exponential Loss

The exponential loss function is mainly used for the training of AdaBoost classifier.

Let the AdaBoost classifier be $\hat{c} = h_T(\boldsymbol{x})$,
the labeled target output is $c \in \mathcal{C}$,
then its loss function is as follows:

$$\begin{aligned}\mathcal{L}_{\text{Exp}}(\hat{c}, c) &= \exp\left(-c \cdot \hat{c}\right) \\ &= \exp\left(-c \cdot h_T(\boldsymbol{x})\right).\end{aligned}$$



Hinge Loss

It is a specific loss function, used for training two-class classifier of SVM.

Hinge loss takes into account the distance of data points from classification decision boundary, if the distance is not large enough, it will be also considered as a loss.

For a linear SVM classifier, $\hat{c} = h(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} + b$, target output $c \in \{-1, +1\}$, then its hinge loss is defined as:

$$\mathcal{L}_{\text{Hinge}}(\hat{c}, c) = \max\left(0,\ 1 - c \cdot h(\boldsymbol{x})\right).$$

According to the above equation: if $c \cdot h(\boldsymbol{x}) \geq 1$ then $\mathcal{L}_{\text{Hinge}}(\hat{c}, c) = 0$, else if $c \cdot h(\boldsymbol{x}) < 1$ then $\mathcal{L}_{\text{Hinge}}(\hat{c}, c)$ increases linearly with the change of $c \cdot h(\boldsymbol{x})$.

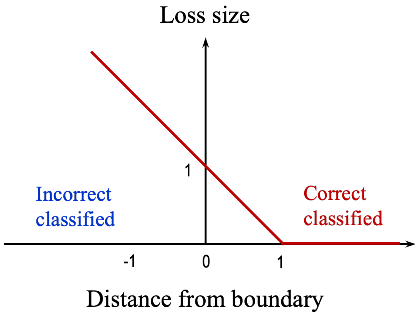
Hinge Loss

In the figure of right side, horizontal axis is the distance between data points and boundary, red line is $\mathcal{L}_{\text{Hinge}}(\hat{c}, c) = \max(0,\ 1 - c \cdot h(\boldsymbol{x}))$.

The left side of horizontal axis is the part of misclassification, $c \cdot h(\boldsymbol{x}) \leq 0$.

The right side of horizontal axis is the correctly classified part:

- if $c \cdot h(\boldsymbol{x}) \geq 1$, the $h(\boldsymbol{x})$ predicts correct category, its hinge loss is 0;
- if $0 < c \cdot h(\boldsymbol{x}) < 1$, it has a certain loss since the distance is not large enough.



Cross-Entropy Loss

It is also known as logarithmic loss, mainly used for training softmax classifier.

For discrete probability distributions $P(\boldsymbol{x})$ and $Q(\boldsymbol{x})$, their cross-entropy is:

$$H(P,Q) = \mathbb{E}_{\boldsymbol{x}\sim Q}[-\log P(\boldsymbol{x})] = -\sum_{\boldsymbol{x}\in\mathcal{X}} Q(\boldsymbol{x}) \log P(\boldsymbol{x}).$$

In machine learning, cross-entropy loss is as follows:

$$\mathcal{L}_{\text{Cross}}\left(\hat{c}, c\right) = -\sum_{i=1}^{n} c_i \log h\left(\boldsymbol{x}_i\right), \quad \text{where } \; n \geq 2.$$

if $n = 2$, it is equivalent to logistic loss, used to train a two-class classifier.

12 Classification Task

Terminology

Confusion matrix of the terms for classifier evaluation

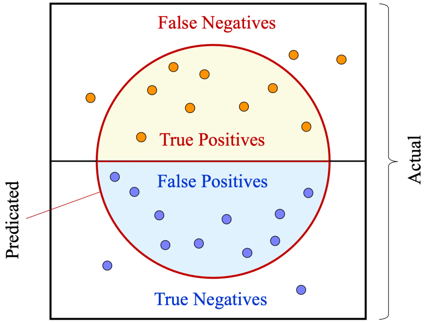
		Actual		
		Positive	Negative	
Predicted	Positive	True Positive (TP)	False Positive (FP)	Predicted Positive (TP+FP)
	Negative	False Negative (FN)	True Negative (TN)	Predicted Negative (FN+TN)
		Actual Positive (TP+FN)	Actual Negative (FP+TN)	

Actual	the labeled actual samples.
Predicted	the predicted results of the classification model.
Positive	the positive cases in the actual samples or predicted results.
Negative	the negative cases in the actual samples or predicted results.

Terminology

Physical view of the terms for classifier evaluation

Actual	the samples in black square.
Predicted	the samples in red circle.
True Positives	the samples in red up half circle.
False Positives	the samples in red down half circle.
False Negative	the samples in black up half square.
True Negatives	the samples in black down half square.



Evaluation Metrics

$$\text{Accuracy} = \frac{\text{True Positive (TP)} + \text{TrueNegative (TN)}}{\text{Actural Positive (AP)} + \text{Actural Negative (AN)}}.$$

$$\text{Precision} = \frac{\text{True Positive (TP)}}{\text{True Positive (TP)} + \text{False Positive (FP)}}.$$

$$\text{Recall} = \frac{\text{True Positive (TP)}}{\text{True Positive (TP)} + \text{False Negative (FN)}}.$$

- Accuracy is useful when all classes are of equal importance.
- Precision assesses how many of predicted positive cases are actual positive.
- Recall assesses how many of true positives are predicted.

Evaluation Metrics

$$\text{Sensitivity} = \frac{\text{True Positive (TP)}}{\text{True Positive (TP)} + \text{False Negative (FN)}}.$$

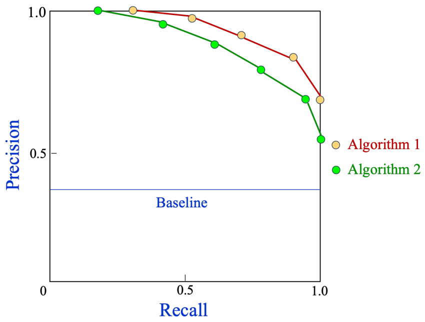
$$\text{Specificity} = \frac{\text{True Negatives (TN)}}{\text{True Negatives (TN)} + \text{False Positives (FP)}}.$$

$$\text{F1 Score} = \frac{\text{True Positive (TP)}}{\text{True Positive (TP)} + \frac{1}{2}\left(\text{False Positive (FP)} + \text{False Negative (FN)}\right)}.$$

- Sensitivity assesses the true positive rate identified correctly, equivalent to Recall.
- Specificity assesses the true negative rate identified correctly.
- F1 Score is the harmonic mean of recall and precision.

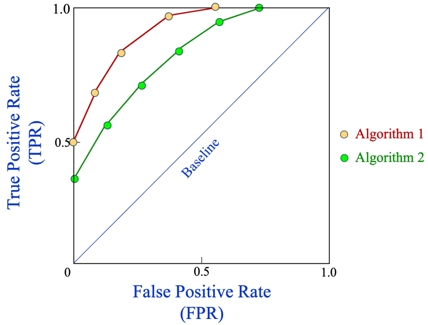
Evaluation Metrics

P-R curve



Precision-recall curve

ROC Curve



Receiver operating characteristic curve

Evaluation Metrics

Matthew's correlation coefficient (MCC)

$$\text{MCC} = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP}+\text{FP})(\text{TP}+\text{FN})(\text{TN}+\text{FP})(\text{TN}+\text{FN})}}.$$

TP: true positives, FN: false negatives, FP: false positives, and TN: true negatives.

MCC will only score high when all four metrics achieve good results, which is proportional to the number of positive elements and negative elements in the data sample.

Thank You