Annals of Data Science (2022) 9:187–212
https://doi.org/10.1007/s40745-020-00253-5

A Comprehensive Survey of Loss Functions in Machine Learning

Qi Wang[1,2,3] · Yue Ma[1,2,3] · Kun Zhao[4] · Yingjie Tian[2,3,5]

Received: 2 March 2020 / Revised: 12 March 2020 / Accepted: 14 March 2020 / Published online: 12 April 2020

Abstract

As one of the important research topics in machine learning, loss function plays an important role in the construction of machine learning algorithms and the improvement of their performance, which has been concerned and explored by many researchers. But it still has a big gap to summarize, analyze and compare the classical loss functions. Therefore, this paper summarizes and analyzes 31 classical loss functions in machine learning. Specifically, we describe the loss functions from the aspects of traditional machine learning and deep learning respectively. The former is divided into classification problem, regression problem and unsupervised learning according to the task type. The latter is subdivided according to the application scenario, and here we mainly select object detection and face recognition to introduces their loss functions. In each task or application, in addition to analyzing each loss function from formula, meaning, image and algorithm, the loss functions under the same task or application are also summarized and compared to deepen the understanding and provide help for the selection and improvement of loss function.

Keywords Loss function · Machine learning · Deep learning · Survey

✉ Yingjie Tian
tyj@ucas.ac.cn

Qi Wang
wangqi173@mails.ucas.ac.cn

[1] School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing 100049, China

[2] Research Center on Fictitious Economy and Data Science, Chinese Academy of Sciences, Beijing 100190, China

[3] Key Laboratory of Big Data Mining and Knowledge Management, Chinese Academy of Sciences, Beijing 100190, China

[4] School of Logistics, Beijing Wuzi University, Beijing 101149, China

[5] School of Economics and Management, University of Chinese Academy of Sciences, Beijing 100190, China

1 Introduction

With the rapid development of computer technology and the emergence of large amounts of complex data, machine learning algorithm gradually tends to diversity to quickly and accurately mine data information. Generally speaking, the machine learning algorithm comes down to solving the optimization problem—structural risk minimization (SRM). Its concrete form is

$$\min_f \frac{1}{N}\sum_{i=1}^{N} L_\theta(f(x_i)) + \lambda R(f) \tag{1}$$

where N is the number of training set samples, the first term is the empirical risk, $L(\cdot)$ is the loss function, θ is the parameter vector, the second term $R(f)$ is the regularization item representing the model complexity, $\lambda \geq 0$ is the tradeoff to balance the empirical risk and the model complexity. It can be seen that loss function is the core part of the empirical risk as well as the important component of the structural risk function. The selection of inappropriate loss function will affect the effectiveness of the algorithm to some extent. Therefore, in recent years, loss function has become one of the hot spots in machine learning, which has profound theoretical significance and practical value. Recently, most of the researches on loss function focus on improving or constructing loss function for a specific algorithm or problem. It still has a big gap to summarize, analyze and compare the classical loss functions. However, the full understanding of classical loss functions in different fields not only enables us to select appropriate loss functions quickly when designing algorithms, but also provides directions and ideas for the improvement of loss functions according to the existing problems of algorithms. Therefore, this review paper proposes a new partition criterion of loss functions, then summarizes 31 important loss functions from several perspectives according to the partition criterion, such as formula, image, algorithm and so on. All loss functions in this paper are listed in Table 1.

In the rest of this paper, the partition criterion of loss functions in this paper will be introduced in Sect. 2, loss functions in traditional machine learning and in deep learning will be analyzed systematically respectively in Sects. 3 and 4, and the final conclusion will be drawn in Sect. 5.

2 Partition Criterion

The loss function corresponds to the machine learning algorithm, so the division of loss function usually adopts the division of machine learning. Next, we give two different partition methods of machine learning, and then give the partition criterion of loss function in this paper.

Machine learning can be divided into supervised learning and unsupervised learning according to whether the samples have label information, and supervised learning can be further divided into classification problem and regression problem. Specifically, the labels of classification problem is a finite number of discrete values,

Table 1 Loss functions

Category	Task/Application	Loss function
Traditional machine learning	Classification problem	0–1 loss, Perceptron loss, Logarithmic loss, Exponential loss, Sigmoid cross entropy loss, Softmax cross entropy loss, Hinge loss, Ramp loss, Pinball loss, Truncated pinball loss, Rescaled hinge loss
	Regression problem	Square loss, Absolute loss, Huber loss, Log-cosh loss, Quantile loss, ϵ-insensitive loss
	Unsupervised learning	Square error, Distance error, Reconstruction error, Negative variance
Deep learning	Object detection	IoU loss, Generalized IoU loss (GIoU loss), Smooth L1 loss, Focal loss
	Face recognition	Contrastive loss, Triplet loss, Center loss, Angular softmax loss (A-Softmax loss), Additive margin softmax loss (AM-Softmax loss), Additive angular margin loss (ArcFace loss)

the labels of regression problems are continuous values, unsupervised learning does not have the label information, which has two main tasks, clustering and dimension reduction.

According to different feature processing methods, machine learning can be divided into traditional machine learning and deep learning. In fact, there is no literature on the definition and division of the two types of machine learning methods at present. Due to their significant differences in the level of data processing and the way to construct features, people are used to distinguish different machine learning algorithms by such terms. The core of traditional machine learning is how to map the input features to the target space to complete the corresponding tasks. The process of feature extraction needs to be realized by some other methods (such as dimension reduction [1, 2], metric learning [3–5]) before using the algorithm to train the model. The traditional machine learning algorithms include linear regression [6], decision tree [7], random forest [8], Adaboost [9] and so on. Deep learning, as an important subclass of machine learning, derives its inspiration from the working mode of human brain and is a learning process that utilizes deep neural network to solve the feature expression. Different from traditional machine learning, deep learning does not need to focus on feature engineering [10], but directly extract features from data when training deep neural networks. That is, deep learning cuts down the work of designing feature extractors for each problem. Deep learning has been widely used in many fields, such as object detection [11–13], face recognition [14, 15], image segmentation [16, 17], machine translation [18, 19], text classification [20, 21], etc. There are significant differences between traditional machine learning and deep learning in many aspects. We can refer to the article 'Machine Learning versus Deep Learning'[1] for details.

From the above introduction to machine learning partitions, it can be found that no matter which way to introduce the loss functions is very confusing, so we propose a new partition criterion in this paper, so that readers can clearly understand each loss function. That is, we divide loss functions into the loss functions in traditional machine learning and in deep learning. The former is divided into those in classification problem, regression problem and unsupervised learning according to task type. The latter is subdivided according to application scenario, and this paper mainly selects two application scenarios, object detection and face recognition, to introduces their loss functions respectively.

In the following sections, $\{x_i\}_{i=1}^{N}(x_i \in R^m)$ stands for the training set, $\{y_i\}_{i=1}^{N}(y_i \in \{-1, 1\})$ stands for the labels in binary classification, $\{y_i\}_{i=1}^{N}(y_i \in \{1, 2, \ldots, K\})$ stands for the labels in multi-classification, $\{y_i\}_{i=1}^{N}(y_i \in R)$ stands for the labels in regression.

[1] Available at https://www.dzone.com/articles/comparison-between-deep-learning-vs-machine-learni.

3 Loss Functions in Traditional Machine Learning

This chapter introduces 21 loss functions in traditional machine learning algorithms, including 11 loss functions for classification problems, 6 loss functions for regression problems and 4 loss functions for unsupervised learning. The following will be introduced in detail.

3.1 Loss Functions in Classification

The formulas and corresponding algorithms of common loss functions in classification are shown in Tables 2 and 3, and their images are shown in Figs. 1 and 2. It should be noted that binary classification can be extended to multi-classification according to one-vs-rest, one-vs-one [26, 27] and so on [28–32]. Therefore, this chapter mainly introduces the loss functions of binary classification problem. Next, we will introduce each loss function in detail by combining the formula and the image, and finally make a comparison and summary from several aspects.

0–1 loss function In the classification problem, the 0–1 loss function means that if the predicted value of the sample has the same sign with the real label, the loss value is 0; otherwise, the loss value is 1. This loss function does not consider the error degree between the predicted value and the real value, that is, when the samples are misclassified, no matter how much error, the corresponding loss values are the same, thus the accuracy and efficiency of the model will be affected to some extent. In addition, it can be seen from the graph of 0–1 loss function that it's a non-convex function and there are many shortcomings in the process of solving the model. In fact, 0–1 loss is usually regarded as a standard in practical application, and the real loss function is its proxy functions, such as logarithmic loss, exponential loss, hinge loss, etc.

Perceptron loss function This loss function is also a piecewise function. When the predicted value of the sample has the same sign with the real label, the loss value is 0; otherwise, the loss value is the absolute value of the predicted value. From the perspective of geometric meaning, the former means that there is no loss for correctly classified samples, while the latter measures the distance from the predicted samples to the decision boundary $f(x) = 0$, and the larger the distance, the greater the error, so the greater the corresponding loss. Obviously, the idea of perceptron loss is vivid and easy to understand, and the loss function is continuous and differentiable with respect to variables so that it's easy to optimize. However, its goal is only to determine the sample category correctly, that is, the sample meets the requirement when it is located in the decision boundary. In this way, the model obtained has poor generalization performance and is not robust to noise data. What's more, its relative algorithm is Perceptron [33], which is a linear classification model of binary classification. For more information about perceptron, you can refer to [34].

Logarithmic loss function This loss function is a function of the sample prediction probability value, where the prediction probability value is obtained through the conditional probability distribution (see Table 2). Specifically, the greater the probability

Table 2 Loss functions in classification

0–1 loss	
Formula	$L(y,f(x)) = \begin{cases} 1, & \text{if} \quad yf(x) < 0 \\ 0, & \text{if} \quad yf(x) \geq 0 \end{cases}$
Explanation	In this table, $f(x) = w^{\top}x + b$.
Algorithm	–
Perceptron loss	
Formula	$L(y,f(x)) = \max\{0, -yf(x)\}$
Explanation	$\max\{a,b\} = \begin{cases} a, & \text{if} \quad a \geq b \\ b, & \text{if} \quad a < b \end{cases}$
Algorithm	Perceptron. The objective function is $L = -\sum_{x_i \in M} y_i(w^{\top}x_i + b)$, where, M is the misclassified sample set.
Logarithmic loss	
Formula	$L(y,\tilde{p}) = -\log \tilde{p}$, where, $\tilde{p} = \begin{cases} p, & \text{if} \quad y = 1 \\ 1-p, & \text{if} \quad y \neq 1 \end{cases}$.
Explanation	Conditional probability distribution: $p = P(y=1\|x) = \frac{1}{1+e^{-f(x)}}$, $1-p = P(y=-1\|x) = \frac{1}{1+e^{f(x)}}$
Algorithm	Logistic regression (LR). The objective function is $L = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{1}{1+e^{-y_i(w^{\top}x_i+b)}}$
Sigmoid cross entropy loss	
Formula	$L(y,\tilde{p}) = -\log \tilde{p}$, where, $\tilde{p} = \begin{cases} p, & \text{if} \quad y = 1 \\ 1-p, & \text{if} \quad y \neq 1 \end{cases}$
Explanation	$p = \sigma(f(x)) = \frac{1}{1+e^{-f(x)}}$, where, $\sigma(\cdot)$ is the sigmoid function
Algorithm	Neural network (NN). The objective function is the same as LR.
Softmax cross entropy loss	
Formula	$L(y, P(y\|x)) = -\log P(y\|x)$
Explanation	$P(y\|x) = \frac{e^{f_y(x)}}{\sum_k e^{f_k(x)}}$, where, $f_y(x)$ and $f_k(x)$ are the decision functions corresponding to the y and k class respectively
Algorithm	Neural network (NN). The objective function is $L = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{e^{w_{y_i}^{\top}x_i + b_{y_i}}}{\sum_k e^{(w_k^{\top}x_i + b_k)}}$
Exponential loss	
Formula	$L(y,f(x)) = e^{-yf(x)}$
Explanation	–
Algorithm	Adaboost. The objective function of the t-th iteration is $L = \sum_{i=1}^{N} \tilde{w}_{ti} e^{-y_i \alpha g(x_i)}$, where, $\tilde{w}_{ti}$can be obtained from the result of iteration t-1, $\alpha, g(\cdot)$ are the variables

of the sample being predicted as its label, the smaller the corresponding loss value; otherwise, the greater the loss value. In the actual calculation, the probability of being predicted as positive class is expressed as p, while the other probability is $1-p$. Fig. 1c, d show the change of logarithmic loss of the positive and negative sample with the probability p, respectively. It can be seen that the value of the logarithmic loss decreases slowly when the correct prediction probability is close to 1, and increases rapidly with the decrease of the correct prediction probability. This trend will make

Table 3 Loss functions in classification (Table 2 Continued)

Hinge loss	
Formula	$H_s(z) = \max\{0, s - z\}$, where, s is a constant
Explanation	Specially, in SVM, $L(y, f(x)) = H_1(yf(x)) = \max\{0, 1 - yf(x)\}$
Algorithm	Support vector machine (SVM). The objective function is $L = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \max\{0, 1 - y_i(w^\top x_i + b)\}$
Ramp loss	
Formula	$R_s(z) = H_1(z) - H_s(z)$, where, $s < 1$ is a constant
Explanation	Specially, in Ramp loss SVM, $L_r(y, f(x)) = H_1(yf(x)) - H_s(yf(x))$
Algorithm	Ramp loss SVM [22]. The objective function is $L = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} L_r(y_i, f(x_i))$
Pinball loss	
Formula	$L_\tau(u) = \max\{u, -\tau u\}$, where, $\tau \in [0, 1]$ is a constant
Explanation	Specially, in pin-SVM, $L_\tau(y, f(x)) = \max\{1 - yf(x), -\tau(1 - yf(x))\}$
Algorithm	pin-SVM [23]. The objective function is $L = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} L_\tau(y_i, f(x_i))$
Truncated pinball loss	
Formula	$L_{\tau,s}(u) = \begin{cases} \tau s, & \text{if} \quad u \le -s \\ -\tau u, & \text{if} \quad -s < u < 0, \\ u, & \text{if} \quad u \ge 0 \end{cases}$ where, $s > 0, \tau \in [0, 1]$ are the constants
Explanation	Specially, in $\overline{pin}$–SVM, $L_{\tau,s}(y, f(x)) = \begin{cases} \tau s, & \text{if} \quad yf(x) \ge 1 + s \\ -\tau(1 - yf(x)), & \text{if} \quad 1 < yf(x) < 1 + s \\ 1 - yf(x), & \text{if} \quad yf(x) \le 1 \end{cases}$
Algorithm	$\overline{pin}$–SVM [24]. The objective function is $L = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} L_{\tau,s}(y_i, f(x_i))$
Rescaled hinge loss	
Formula	$L_{rhinge}(z) = \beta(1 - e^{-\eta H_s(z)})$, where, $\eta > 0$ is a scaling constant, $\beta = \frac{1}{1-e^{-\eta}}$ is a normalizing constant
Explanation	Specially, in the improved SVM based on this loss function, $L_{rhinge}(y, f(x)) = \beta(1 - e^{-\eta H_1(yf(x))})$
Algorithm	[25]. The objective function is $L = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} L_{rhinge}(y_i, f(x_i))$

the prediction output closer to the label, which is conducive to the convergence of the algorithm. What's more, its relative algorithm is logistic regression (LR) [35], which is a logarithmic linear model. By using the maximum likelihood estimation (MLE) method to estimate the model parameters, the corresponding objective function and logarithmic loss can be derived. For the details, you can refer to [36].

Sigmoid cross entropy loss function From the formula of this loss function (see Table 2), it can be seen that sigmoid cross entropy loss and logarithmic loss are the same. The reason why the same losses are described by two names is that they are defined from different sources. The sigmoid cross entropy loss is obtained by seeing the probability value converted from the predicted value through the sigmoid activation function as the actual output of cross entropy, and the real label of the sample as the expected output of cross entropy. Cross entropy describes

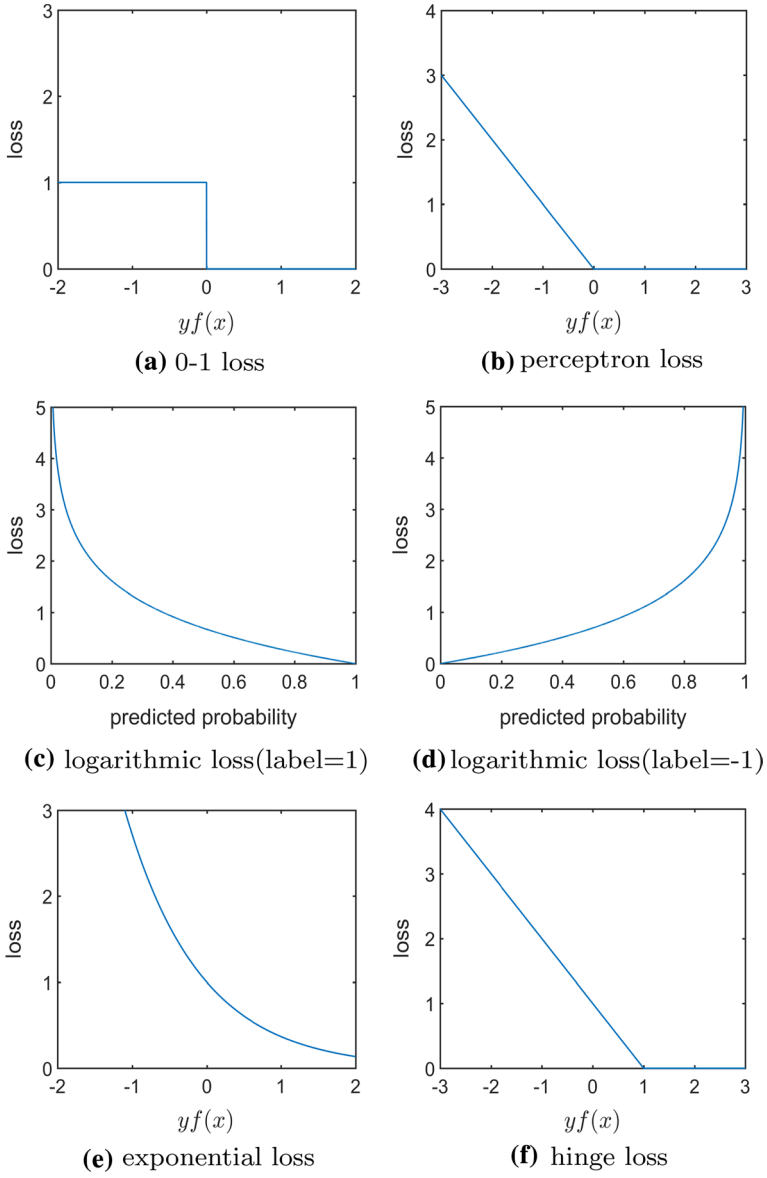
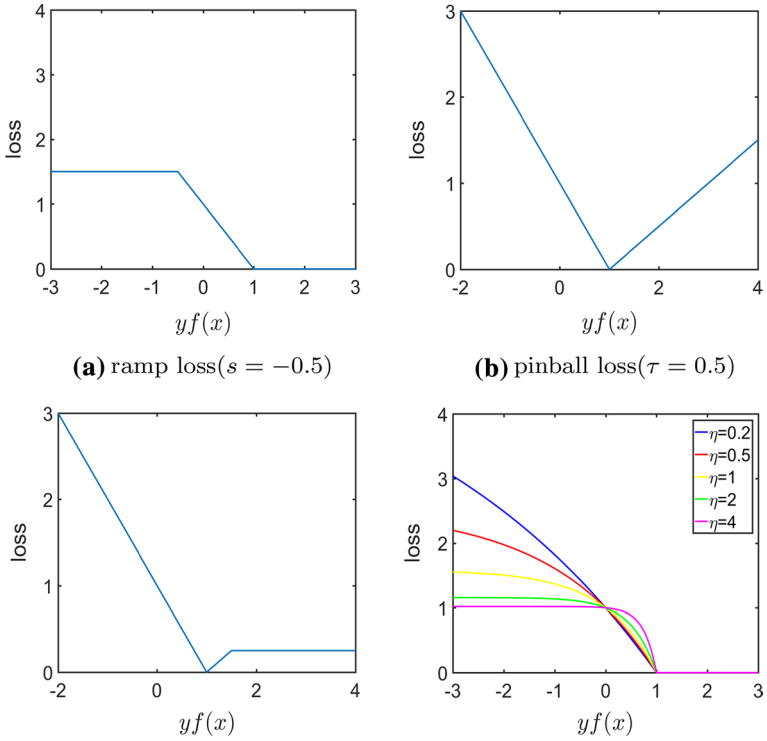


Fig. 1 The illustrations of loss functions in classification

the distance between the actual output and the expected output. The smaller the value of cross entropy is, the closer the two probability distributions are. Therefore, the meaning of sigmoid cross entropy loss is the same as that of logarithmic loss. What's more, sigmoid cross entropy loss function is used in neural networks (NN) [37] to avoid the slow learning rate of neurons in the output layer. Generally, in deep learning, this loss function is called cross entropy loss, and in logical



(a) ramp loss($s = -0.5$) **(b)** pinball loss($\tau = 0.5$)

(c) truncated pinball loss($\tau, s = 0.5$) **(d)** rescaled hinge loss

Fig. 2 The illustrations of loss functions in classification (Fig. 1 continued)

regression, it is called logarithmic loss. Of course, the logistic regression model can also be derived from the perspective of entropy, for details refer to [38].

Softmax cross entropy loss function The only difference between this loss function and sigmoid cross entropy loss function is that this loss function replaces the sigmoid function with the softmax function to solve the multi classification problem. Of course, we can also explain the expansion from the perspective of probability distribution, for details refer to [35]. Since many practical problems (such as semantic segmentation, text mining) are multi classification problems, softmax cross entropy loss has become the main loss function of deep learning. In many deep learning literatures, both sigmoid and softmax cross entropy loss are called cross entropy loss, which can be determined by the problem itself.

Exponential loss function This loss function is an approximation of the 0–1 loss function. For the disadvantages of 0–1 loss, exponential loss has the following two improvements. First, exponential loss function is continuous and differentiable, which is conducive to optimization. Secondly, for the misclassified samples, the penalty degrees of exponential loss are different, which are related to the error degrees, and there is also a cost to the correctly classified sample. From the

image Fig. 1e, it can be seen that as the error degree increases, the corresponding loss value increases faster, which is conducive to the convergence of the model; and as the probability of being predicted correctly increases, the corresponding loss becomes smaller and smaller, enhancing the generalization performance of the model. What's more, the representative algorithm of exponential loss function is AdaBoost [9], which is a binary classification model proposed by Yoav Freund and Robert schapire in 1995. One interpretation of Adaboost is that Adaboost is an addition model learned through a forward step algorithm, in which the optimal loss function during each iteration is an exponential loss. For the specific conversion process, you can refer to [39].

Hinge loss function In the binary classification problem, hinge loss function defines the margin (represented by the parameter s) near the decision boundary, and the correctly classified samples in the middle of the two margin boundaries and all misclassified samples have the cost. From the image Fig. 1f, compared with perceptron loss function, hinge loss function shifts s to the right, which not only requires the sample to be classified correctly, but also requires the loss to be 0 when the confidence is high enough (fully classified correctly). In addition, unlike exponential loss function, hinge loss function does not punish the correct classification samples of $|f(x)| \geq s$. It is considered that such samples have been learned well enough, so that the model is more focused on the overall classification error. It should be noted that s is generally set to 1 in applications and improvements of hinge loss function. What's more, support vector machines (SVMs) [40] are the first algorithm to use hinge loss function, which is a linear or nonlinear model to solve the binary classification task. This model can be formalized as an optimization problem of convex quadratic programming, and can also be equivalent to minimizing hinge loss with regularization. For details of this algorithm, you can refer to [41].

Ramp loss function This loss function [22] is an improved function of hinge loss. From the images of two loss functions (Figs. 1f and 2a), it can be seen that the hinge loss value of outlier is very large and outliers play a leading role in determining the decision boundary, so that the model will reduce the accuracy of normal samples to reduce such loss, and finally reduce the overall classification accuracy, resulting in low generalization ability of the model. However, ramp loss function limits the maximum loss value, which limits the influence of outliers to some extent, and the model is more robust to outliers. In addition, when ramp loss function is applied to SVM, the number of support vectors can be reduced and the training efficiency can be improved [42]; but at the same time, the objective becomes a nonconvex function, which can be solved by CCCP [43].

Pinball loss function This loss function was originally proposed for regression tasks, in which it is called quantile loss. It was not until 2013 that [23] applied pinball loss to SVM and proposed the pin-SVM classifier, and this loss function began to be used to solve the classification problem. Pinball loss function has good properties in solving classification problems. Compared with hinge loss function, pinball loss function penalizes all correctly classified samples, which not only does not increase the computational complexity and maintains the same classification error bound, but also makes the model less sensitive to noise near the decision boundary, have stronger resampling stability and more robust to outliers. However, since the

sub-gradient of pinball loss is not 0 almost everywhere, pin-SVM loses the sparsity of the standard SVM.

Truncated pinball loss function In order to make up for the shortcomings of pinball loss, [24] truncates pinball loss and proposes truncated pinball loss function. This loss function preserves the advantages of pinball loss and reduces the sensitivity to noise near the decision boundary. In addition, the SVM model $\overline{pin}$–SVM based on this loss function preserves the sparseness of the standard SVM, and the objective function of $\overline{pin}$–SVM is also non-convex, which can be solved by CCCP method.

Rescaled hinge loss function This loss function [25] is the improved loss of hinge loss function. From the image Fig. 2d, it can be found that rescaled hinge loss and ramp loss are very similar, both of which enhance the robustness to outliers by improving the function form of $yf(x) < 1$. Since outliers affect the sparsity of SVM, the SVM model based on rescaled hinge loss also improves the sparsity.

Comparison The above is the specific introduction of 11 loss functions in classification. Below, we will summarize and compare them briefly from several aspects. (1) Convexity. Perceptron loss, logarithmic loss (cross entropy loss), exponential loss, hinge loss, and pinball loss are all convex functions. 0–1 loss, ramp loss, truncated pinball loss, and rescaled hinge loss are all non-convex functions. (2) Robustness. Truncating pinball loss and pinball loss can reduce the sensitivity to noise near the decision boundary, and ramp loss and rescaled hinge loss are robust to outliers. (3) Generalization performance. Logarithmic loss (cross entropy loss), exponential loss and hinge loss are better than perceptron loss. (4) SVM sparsity. The SVM model based on ramp loss, truncated pinball loss and rescaled hinge loss can preserve or even improve the sparsity of the standard SVM (based on hinge loss).

3.2 Loss Functions in Regression

The formulas and corresponding algorithms or applications of common loss functions in regression are shown in Table 4 and their images are shown in Fig. 3. Next, we will introduce each loss function in detail by combining the formula and the image, and finally make a comparison and summary from several aspects.

Square loss function This loss function, one of the most common loss functions in regression problems, measures the square of the error between the true value y and the predicted value $f(x)$ of the sample x. As can be seen from the image Fig. 3a, the gradient of square loss is variable, the length of the gradient is large if the sample has large loss, and the length of the gradient decreases when the loss value is close to 0. These properties are beneficial to the fast convergence and high accuracy of the model. But square loss is sensitive to outliers. This is because square loss squares the error e ($e = y - f(x)$), and when $|e|$ gets larger, the loss increases faster. Therefore, the loss value of outlier is very large, which makes the model pay more attention to outliers, and the parameters will be adjusted constantly so that such loss decreases continuously. But correspondingly, the prediction accuracy of other normal samples will decline, ultimately reducing the overall prediction performance. What's more, when solving the

Table 4 Loss functions in regression

Square loss	
Formula	$L(y,f(x)) = (y-f(x))^2$
Algorithm	Linear regression. The objective function is $L = \sum_{i=1}^{N}(y_i - w^\top x_i - b)^2$
Absolute loss	
Formula	$L(y,f(x)) = \lvert y-f(x)\rvert$
Algorithm	Linear regression. The objective function is $L = \sum_{i=1}^{N}\lvert y_i - w^\top x_i - b\rvert$
Huber loss	
Formula	$L_\delta(y,f(x)) = \begin{cases} \frac{1}{2}(y-f(x))^2, & \text{if} \quad \lvert y-f(x)\rvert \le \delta \\ \delta\lvert y-f(x)\rvert - \frac{1}{2}\delta^2, & \text{otherwise} \end{cases}$, where, parameter $\delta > 0$
Algorithm	Robust statistics, additive model, deep learning.
Log-cosh loss	
Formula	$L(y,f(x)) = \log(\cosh(f(x)-y))$
Algorithm	XGBoost
Quantile loss	
Formula	$L_\gamma(y,f(x)) = \begin{cases} (\gamma-1)(y-f(x)), & \text{if} \quad y < f(x) \\ \gamma(y-f(x)), & \text{if} \quad y \ge f(x) \end{cases}$, where, $\gamma \in (0,1)$ is the given quantile
Algorithm	Quantile regression. The objective function is $L = \sum_{i:y_i<f(x_i)}(\gamma-1)\cdot(y_i - f(x_i)) + \sum_{i:y_i\ge f(x_i)}\gamma\cdot(y_i - f(x_i))$
ϵ-insensitive loss	
Formula	$L_\epsilon(y,f(x)) = \max\{0, \lvert y-f(x)\rvert - \epsilon\} = \begin{cases} 0, & \text{if} \quad \lvert y-f(x)\rvert \le \epsilon \\ \lvert y-f(x)\rvert - \epsilon, & \text{if} \quad \lvert y-f(x)\rvert > \epsilon \end{cases}$, where, parameter $\epsilon > 0$
Algorithm	Support vector regression (SVR). The objective function is $L = \frac{1}{2}\lVert w\rVert^2 + C\sum_{i=1}^{N} L_\epsilon(y_i,f(x_i))$

linear regression model [6], the most commonly used method is the least square method, and the core of the least square method is square loss. In addition, square loss can also be used for classification problems, usually measuring the difference between the sample label and the probability predicted as a certain class.

Absolute loss function This loss function, another common loss function in regression problems, measures the absolute value of the error between the true value y and the predicted value $f(x)$ of the sample x. As can be seen from the image Fig. 3b, absolute loss dose not increase rapidly with the increase of error. Therefore, absolute loss is more robust than square loss when there are outliers in the training set, and absolute loss should be selected when outliers are detected and have an impact on the learning of the model. However, since absolute loss function is not smooth when the error approaches 0, it is less used than square loss. In addition, the gradient of absolute loss is fixed and has nothing to do with the loss value, which will affect the efficiency of solving the model. What's more, when solving the linear regression model, the minimum absolute value error can also be taken as the objective function, which corresponds to absolute loss. But the most common method is still least squares.

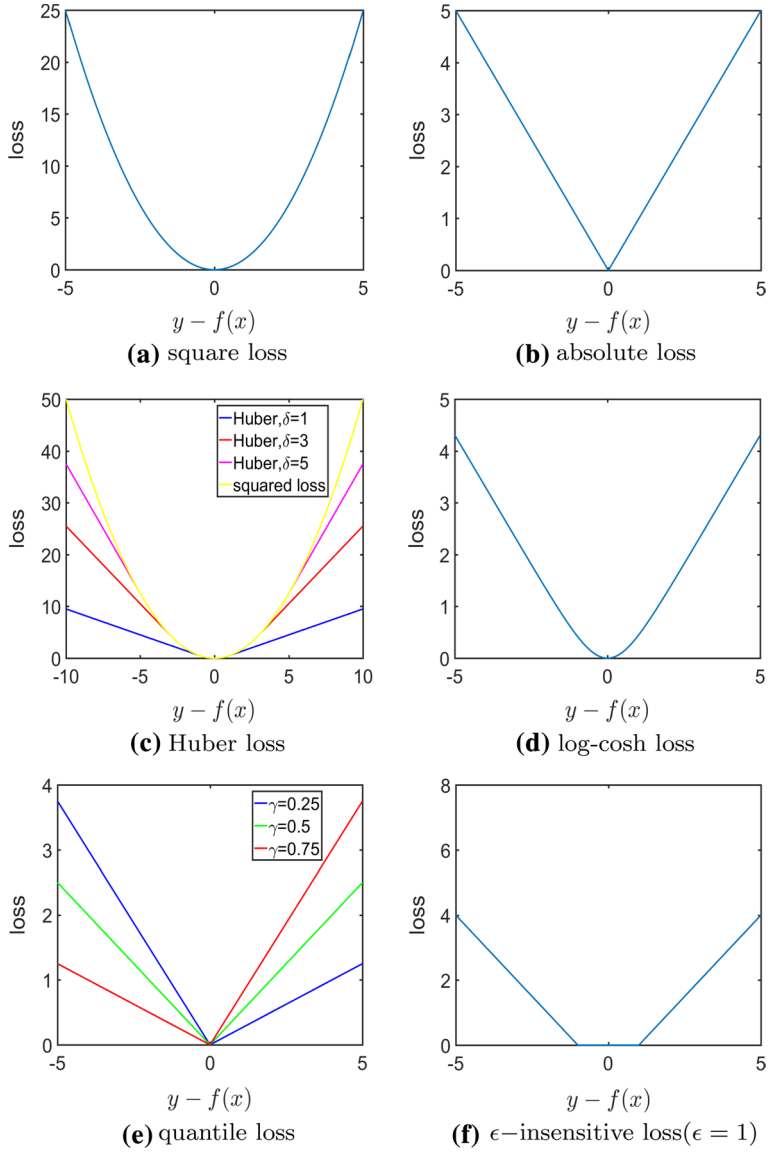


Fig. 3 The illustrations of loss functions in regression

Huber loss function It can be seen from the formula of Huber loss [44] that this loss is a piecewise function of square loss and absolute loss. Huber loss uses the parameter δ as the boundary to judge whether it is a more singular sample. The samples within this boundary use square loss, and the samples beyond this boundary use absolute loss, so as to reduce the weight of the loss of outliers in the total loss and avoid the model overfitting. In addition, the use of square loss

within the boundary can prevent the gradient from descending too fast and missing the optimum. Huber loss effectively combines the advantages of square loss and absolute loss and is differentiable everywhere. However, while the introduction of δ brings benefits, it also makes the loss function more complex and require constant iteration to train δ. In practice, Huber loss is often used in robust statistics [44], additive model [45]. It is also used in deep learning, such as smooth L1 loss of object detection, which is a special form of Huber loss.

Log-cosh loss function This loss function is the logarithm of the hyperbolic cosine of the error. When the error is small, the loss is approximately $\frac{1}{2}(y - f(x))^2$; when the error is large, the loss is approximately $|y - f(x)| - \log 2$. From the formula of log-cosh loss function, it can be seen that this loss function is very similar to Huber loss, so it has all the advantages of Huber loss, that is, it is robust to outliers and the gradient is variable. In addition, log-cosh loss is quadratic differentiable everywhere. Since many machine learning algorithms, such as XGBoost, use Newton's method to solve optimization problems, it is important to be second-order differentiable. However, log-cosh loss still has the problem of gradient and Hessian. For the samples with large error, the gradient and Hessian are also constant, which affects the efficiency of solution or has other effects, such as the lack of split points in XGBoost.

Quantile loss function This loss function is actually an extended form of absolute loss, which degenerates to absolute loss when γ takes the 50th percentile. Unlike the previous four loss functions, the regression model based on quantile loss can provide a reasonable prediction interval, and we get the range of the predicted value instead of just a predicted value. Quantile loss function adjusts the weight of each sample according to the selected quantile γ. The smaller γ, the more punishment will be given to those samples whose real value is less than the predicted value (i.e. the overestimated samples); on the contrary, the larger γ, the more punishment will be given to those samples whose real value is greater than the predicted value (i.e. the underestimated samples). What's more, its representative application is quantile regression, which is a regression analysis used in statistics and econometrics. For more information about quantile regression, you can refer to [46].

ϵ-insensitive loss function This loss function is commonly used in support vector regression (SVR) [47]. It can be seen from its image Fig. 3f that the difference with the above losses is that this loss does not punish the samples whose error does not exceed ϵ, making the model focus on the samples with large prediction error. But ϵ-insensitive loss function contains absolute term, so it is not differentiable.

Comparison The above is the specific introduction of 6 loss functions in regression. Below, we will summarize and compare them briefly from several aspects. (1) Derivative. Absolute loss, quantile loss and ϵ-insensitivity loss are not smooth, Huber loss is first derivative, and square loss and log-cosh loss are second derivative. (2) Robustness. Huber loss, log-cosh loss and absolute loss are robust to outliers.

3.3 Loss Functions in Unsupervised Learning

The two major learning tasks of unsupervised learning are clustering and dimension reduction. The objective of clustering is to divide the samples into different clusters according to the similarity index, so that the similarity within the cluster is high and the similarity between the clusters is low. Different clustering algorithms only use different indexes to describe similarity and adopt different strategies. For example, hierarchical clustering algorithm measures similarity by distance, while density-based clustering algorithm measures similarity by density. In this section, we take K-means, a typical clustering algorithm, as an example to introduce its loss function. And below, $\{C_1, C_2, \ldots, C_K\}$ stands for the cluster partition.

Dimension reduction refers to the transformation of the original high-dimensional attribute space into a low-dimensional subspace through a certain mathematical transformation, which can compress the data while preserving the data structure and usefulness. The classical linear dimension reduction methods include principal component analysis (PCA), and the nonlinear dimension reduction methods include manifold learning. In this section, we will summarize the loss functions in dimension reduction into three categories according to the ideas of different dimension reduction algorithms, and give the corresponding algorithms respectively. And below, h stands for the projection mapping from the original space R^m to the low-dimensional space $R^d (d \leq m)$, $\{z_i\}_{i=1}^N$ stands for the low-dimensional representation of the original samples $\{x_i\}_{i=1}^N$.

Next, we will introduce 4 loss functions, namely, square error of clustering, distance error, reconstruction error and negative variance of dimension reduction. The formulas and corresponding algorithms are shown in Table 5. It should be noted that these four names are given by ourselves according to the meaning of each loss.

Square error K-means [48] hope to obtain cluster partition by minimizing square error. Intuitively, square error describes the compactness of samples in the cluster around the mean vector of the cluster. The smaller the square error, the higher the similarity of samples within the cluster. There are two sets of variables $\{r_{ik}\}$ and $\{\mu_k\}(i = 1, \ldots, N; k = 1, \ldots, K)$, minimizing this loss is a NP hard problem [49], so k-means adopts greedy strategy and uses iterative optimization to approximate the solution.

Distance error A classical dimension reduction criterion is that the distance between samples in the original space is maintained in the low-dimensional space. Based on this idea, we define the loss function as the distance error. By minimizing the distance error, the representation $\{z_i\}_{i=1}^N$ can be obtained. In order to solve this objective function, the algorithm generally starts from the condition $\|z_i - z_j\| = dist_{ij}(1 \leq i, j \leq N)$. The most representative dimension reduction algorithm is multiple dimensional scaling (MDS) [50]. Since MDS is used in the isometric mapping (Isomap) [51] of manifold learning, Isomap is also one of the algorithms applying distance error.

Reconstruction error There are two reconstruction ideas in dimension reduction algorithms. One is that the distance between the original sample and the reconstructed sample based on the inverse projection mapping is close enough. The other is to try to keep a certain property of the original space in the low-dimensional

Table 5 Loss functions in unsupervised learning

Square error	
Formula	$L = \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik} \|x_i - \mu_k\|^2$, where, $r_k = \begin{cases} 1, & \text{if } x \in C_k \\ 0, & \text{otherwise} \end{cases}$, $\sum_{k=1}^{K} r_k = 1$, $\mu_k = \frac{1}{\|C_k\|} \sum_{x \in C_k} x$ is the mean vector of the cluster C_k
Explanation	Cluster partition $\{C_1, C_2, \ldots, C_K\}$
Algorithm	K-means
Distance error	
Formula	$L = \sum_{i,j=1}^{N} (\|z_i - z_j\| - dist_{ij})^2$, where, $dist_{ij}$ is the distance between x_i and x_j in the original space
Explanation	$h : R^m \mapsto R^d (d \leq m)$, $x_i \mapsto z_i (i = 1, \ldots, N)$
Algorithm	MDS, Isomap
Reconstruction error	
Formula	1. $L = \sum_{i=1}^{N} \|h^{-1}(z_i) - x_i\|$ 2. $L = \sum_{i=1}^{N} \|o(z_i) - z_i\|$
Explanation	$o : R^m \mapsto R^m$, $x_i \mapsto o(x_i) (i = 1, \ldots, N)$
Algorithm	1. PCA 2. LLE
Negative variance	
Formula	$L = -tr(\sum_{i=1}^{N} (z_i - \bar{z})(z_i - \bar{z})^\top)$, where, $\bar{z}$ is the mean vector
Explanation	$tr(\cdot)$ is the trace of matrix
Algorithm	PCA

space, so that the projected sample can be reconstructed with some samples in the low-dimensional space based on this property. Because the essence of these two ideas is to reconstruct the samples, we call both as reconstruction error. By optimizing the first reconstruction error, we can get the projection mapping h and the representation $\{z_i\}_{i=1}^N$. Its typical algorithm is principal component analysis (PCA) [52], which is a linear dimension reduction method. The specific algorithm flow can be found in [53]. By optimizing the second reconstruction error, we can get the low-dimensional vectors $\{z_i\}_{i=1}^N$. Its typical algorithm is locally linear embedding (LLE) [1] of manifold learning. In LLE, since the original samples are distributed on a manifold, they have local linearity, that is, each sample can be reconstructed by linear combination of its neighborhood samples. LLE hopes that this property can be maintained in the low-dimensional space, so the shared parameters of two spaces are the weight coefficients of the linear reconstruction. For more information about LLE, you can refer to [1].

Negative variance Another dimension reduction criterion is maximum separability, that is, all samples after projection are as separate as possible. This criterion can be measured by maximizing the variance of the samples after projection. Since optimizing loss function is to get the minimum, we name this loss function as negative variance. By optimizing the negative variance, we can get the projection mapping h and the low-dimensional vectors $\{z_i\}_{i=1}^N$. Its representative algorithm is PCA. The maximum separability is another interpretation of PCA. Two PCA objective functions based on reconstruction error and negative variance are equivalent.

4 Loss Functions in Deep Learning

This chapter introduces 10 loss functions of deep learning, including 4 loss functions of object detection and 6 loss functions of face recognition. It will be introduced in detail below.

4.1 Loss Functions in Object Detection

As one of the important applications of computer vision, object detection has been explored for a long time. Especially in recent years, the rapid development of deep learning has brought many new algorithms and techniques to object detection. Object detection is a multi-task learning problem, including object classification and object localization. For example, we not only need to use algorithms to determine whether the object in the picture is a car, but also mark its position accurately and circle the car with a bounding box.

Object classification can be a binary classification problem, such as detecting the cat in the picture, or a multi-classification problem, such as detecting the cat and dog in the picture respectively. In the object detection algorithms, the input image will generate thousands of candidate boxes. We need to classify these candidate boxes, but only a few of them contain real objects, which brings about the problem of class imbalance. Taking binary classification as an example, the positive samples (foreground samples) are far less than the negative samples (background samples), and most of the negative samples are useless samples which are easy to be classified. The total loss of these samples accounts for a large proportion of the total loss. Thus, the learning direction of the model will change, and the model will only distinguish the background, not the specific object. Many algorithms (such as OHEM [57]) to solve class imbalance cannot take into account both foreground-background class imbalance and easy-hard samples imbalance. Therefore, in this section, we introduce a loss function, focal loss, which can effectively solve two class imbalance problems.

Object localization is a regression problem, which is represented in the algorithm by measuring the offset between the candidate box and the ground truth box, and then modifying the bounding box. In fact, the measure is the loss function, and the modification is the optimization process of minimizing the loss function. In this section, we introduce several loss functions commonly used in object localization, including smooth L1 loss, IoU loss, GIoU loss and KL loss.

The formulas and corresponding algorithms of several loss functions are shown in Table 6. Next, we will combine the formulas to analyze and compare several loss functions.

Focal loss function This loss function [54] is modified on the basis of sigmoid cross entropy loss, $\tilde{\tau}$ is to solve the foreground-background class imbalance, and $(1-\tilde{p})^{\gamma}$ is to solve the imbalance of easy-hard samples. From its formula, it can be seen that the easy sample has the large $\tilde{p}$ and the small loss value, while the hard sample is just the opposite. Thus, the weights of easy samples are reduced, so that

Table 6 Loss functions in object detection

Focal loss	
Formula	$L = -\tilde{\tau}(1-\tilde{p})^{\gamma}\log(\tilde{p})$, where, $\tilde{\tau} = \begin{cases} \tau, & \text{if } y=1 \\ 1-\tau, & \text{otherwise} \end{cases}$, $\tilde{p} = \begin{cases} p, & \text{if } y=1 \\ 1-p, & \text{otherwise} \end{cases}$, $\tau \in [0,1]$ and $\gamma \geq 0$ are parameters, $p \in [0,1]$ is the probability of being predicted as the positive class
Algorithm	RetinaNet [54]
Explanation	–
Smooth L1 loss	
Formula	$smooth_{L_1}(a) = \begin{cases} 0.5a^2, & \text{if } \lvert a\rvert < 1 \\ \lvert a\rvert - 0.5, & \text{otherwise} \end{cases}$
Algorithm	Fast R-CNN [11], Faster R-CNN [12]. $L(B^p, B^g) = \sum_{i\in\{x,y,w,h\}} smooth_{L_1}(B_i^p - B_i^g)$, where, B^p is the predicted box, B^g is the ground truth box
Explanation	The box is defined as (x, y, w, h), where, (x, y) is its top-left corner, w is its weight, h is its height
IoU loss	
Formula	$L = -\ln(IoU) = -\ln\frac{I}{U}$, where, I is the intersection area of two boxes, U is the union area of two boxes.
Algorithm	UnitBox [55]. The specific calculation formulas of I and U can be found in [55]
Explanation	For each pixel (i, j), the predicted box is $B^p = (x_t^p, x_b^p, x_l^p, x_r^p)$, the ground truth box is $B^g = (x_t^g, x_b^g, x_l^g, x_r^g)$, where, (t, b, l, r) represent the distance between this pixel and the top, bottom, left, right bounds of ground truth, respectively
GIoU loss	
Formula	$L = 1 - GIoU$, where, $GIoU = IoU - \frac{A^c - U}{A^c}$, $IoU = \frac{I}{U}$, B^c is the smallest box enclosing both boxes, A^c is the area of B^c, I is the intersection area of two boxes, U is the union area of two boxes
Algorithm	[56]. The specific calculation formulas of I, U and A^c can be found in [56]
Explanation	The box is defined as (x_1, y_1, x_2, y_2), where, (x_1, y_1) is its top-left corner, (x_2, y_2) is its bottom-right corner

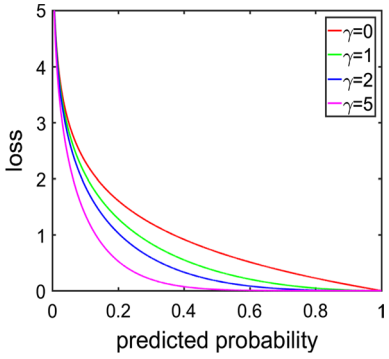


Fig. 4 The illustration of focal loss function

the model focuses more on the hard samples. In addition, γ is used to adjust the speed of loss reduction. When $\gamma = 0$, focal loss is sigmoid cross entropy loss. The image of the positive sample is shown in Fig. 4.

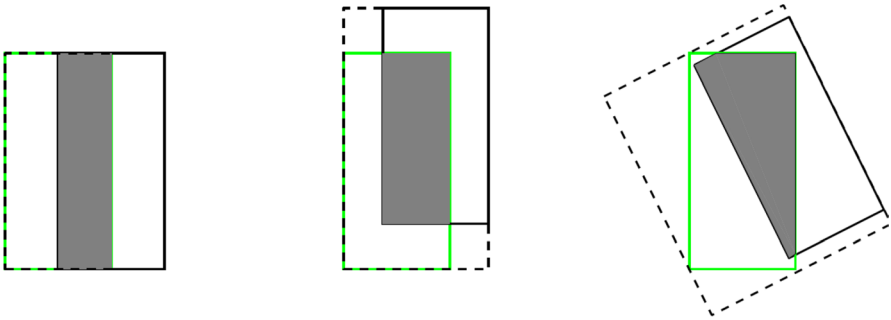


Fig. 5 Three different ways of overlap between two rectangles with the same IoU values and different GIoU values [56]

Smooth L1 loss function From its formula, we can see that smooth L1 loss is a Huber loss of $\delta = 1$, so it has all the properties of Huber loss. Square loss, used in R-CNN [58] and SPP-Net [59], has the problem of gradient explosion during optimization, so we need to carefully adjust the learning rate. Fast R-CNN [11] and Faster R-CNN [12] use smooth L1 loss function to effectively eliminate this problem. However, when using smooth L1 loss to train the regression model, the bounding box is regarded as four independent variables, which is inconsistent with the actual situation, so it will affect the accuracy of localization.

IoU loss function Intuitively, IoU loss [55] maximizes the coincidence between the predicted box and the ground truth box. From the formula point of view, when calculating the area of intersection and union of two boxes, four variables of measuring each box are used at the same time. Therefore, this loss function regards a box as a whole for training, and can get more accurate predicted box. In addition, regardless of the scale of the ground truth, IoU is normalized to [0, 1], which can prevent the model from focusing too much on large objects and ignoring small ones. [55] found that the use of IoU loss not only makes the location more accurate, but also speeds up the convergence rate.

GIoU loss function This loss function [56] is an extension of IoU loss. It not only retains the property of IoU loss, but also solves the problems of IoU loss effectively. As can be seen from its formula, by introducing the smallest box enclosing both the predicted box and the ground truth box, there is also the corresponding loss when two boxes do not intersect. Meanwhile, when two boxes intersect and there are the same IoU values under different overlap modes, the better the overlap mode is, the smaller the GIoU loss value will be. For example, Fig. 5 shows three different overlap modes with the same IoU values, i.e. $IoU = 0.33$, but different GIoU values, i.e. from the left to right $GIoU = 0.33, 0.24, -0.1$ respectively. GIoU value will be higher for the cases with better aligned orientation.

Comparison Compared with cross entropy loss, focal loss can solve both the imbalance of positive and negative classes and the imbalance of easy and hard samples. In the bounding box regression, smoothing L1 loss can eliminate the problem that square loss is sensitive to outliers. Compared with square loss and smooth L1 loss, IoU loss solves the influence of object scale, and improves the accuracy of

regression by learning the bounding box as a whole. GIoU loss not only preserves the property of IoU loss, but also gives the corresponding loss to two disjoint boxes, and can reflect which way of overlapping of two boxes is better through the GIoU loss value.

4.2 Loss Functions in Face Recognition

Face recognition is a kind of biometric technology based on facial feature information. It has experienced a long period of development, and the emergence of deep learning methods has further improved the accuracy of face recognition. The data set of face recognition is rather special. There are many classes (one person represents one class), but there are not many samples in each class. Only a few classes participate in the training, that is, the existing samples can not cover all classes, and the classes of test samples usually do not appear in the training.

Based on the above difficulties, face recognition methods need to extract key information (that is, learn distinguishing features) from face images, so that the samples within one class (different images of the same person) are similar, and the samples between classes (different images of different people) are different. One of the main challenges of feature learning in large-scale face recognition using deep convolutional neural network (DCNN) is to design appropriate loss function to improve the recognition ability. At present, the loss functions of face recognition can be roughly divided into two categories. One is to measure the difference between samples based on Euclidean space distance, and the representative algorithms include contrastive loss, triplet loss, center loss. The other is to measure the difference between samples based on angular space, and the representative algorithms include A-Softmax loss, AM-Softmax loss, ArcFace loss.

The formulas and corresponding algorithms of several loss functions are shown in Table 7. Since the input of several loss functions is the feature vector of image mapped to feature space, we use $\mathbf{x}$ to represent the feature of image x extracted by DCNN. Next, we will combine the formula of loss function to analyze and compare several loss functions.

Contrastive loss function Before using the contrastive loss function [60, 61] to train the model, we need to reconstruct the training set. The specific methods are as follows. The samples in the original training set are paired and given labels. When the two samples are from the same person, the label $Y = 1$ is given. Otherwise, the label $Y = 0$ is given. A new sample set consisting of P tuples (Y, x^1, x^2) is obtained. From the formula, it can be seen that the corresponding loss of $Y = 1$ is $L_S = \frac{1}{2}\|\mathbf{x}^1 - \mathbf{x}^2\|^2$, and the corresponding loss of $Y = 0$ is $L_D = \frac{1}{2}(\max\{m - \|\mathbf{x}^1 - \mathbf{x}^2\|, 0\})^2$. For one thing, the smaller the Euclidean distance of similar samples in the feature space is, the smaller the loss value is, which ensures the similarity of samples from one person. For another thing, the larger the Euclidean distance of different samples in the feature space is, the smaller the loss value is, which ensures the difference of samples from different persons. In addition, the threshold m defines a radius so that there is a cost if the distance of pair from different persons is within the radius. It makes

Table 7 Loss functions in face recognition

Contrastive loss	
Formula	$L = \frac{1}{2}\sum_{i=1}^{P} Y_i d_i^2 + (1 - Y_i)(\max\{m - d_i, 0\})^2$, where, $d_i = \|\mathbf{x}_i^1 - \mathbf{x}_i^2\|(i = 1, \ldots, P)$, $m > 0$ is a threshold
Explanation	In this table, $\mathbf{x}$ represents the feature vector of image x extracted by DCNN. In this loss, the new sample set $\{(Y_i, x_i^1, x_i^2)\}_{i=1}^{P}$, where, $Y = 1$ represents that x^1 and x^2 are from the same class, $Y = 0$ represents that x^1 and x^2 are from the different classes
Algorithm	[60, 61]
Triplet loss	
Formula	$L = \sum_{i=1}^{P} \max\{\|\mathbf{x}_i^a - \mathbf{x}_i^p\|^2 - \|\mathbf{x}_i^a - \mathbf{x}_i^n\|^2 + \alpha, 0\}$, where, α is a margin
Explanation	The new sample set $\{(x_i^a, x_i^p, x_i^n)\}_{i=1}^{P}$, where, x^a and x^p are from the same class, x^a and x^n are from the different classes
Algorithm	FaceNet [62]
Center loss	
Formula	$L_C = \frac{1}{2}\sum_{i=1}^{N} \|\mathbf{x}_i - c_{y_i}\|^2$, where, c_{y_i} is the feature center of the y_i class
Explanation	–
Algorithm	[63]
A-Softmax loss	
Formula	$L = \frac{1}{N}\sum_i -\log \frac{e^{\|\mathbf{x}_i\| \cos(m\theta_{y_i,i})}}{e^{\|\mathbf{x}_i\| \cos(m\theta_{y_i,i})} + \sum_{j \neq y_i} e^{\|\mathbf{x}_i\| \cos(\theta_{j,i})}}$, where, $m \geq 1$ and $m$is an integer
Explanation	In A-Softmax loss, AM-Softmax loss and ArcFace loss, $\theta_{a,b}$ is the angle between the weight W_a and the feature $\mathbf{x}_b$
Algorithm	SphereFace [64, 65]
AM-Softmax loss	
Formula	$L = \frac{1}{N}\sum_i -\log \frac{e^{s(\cos(\theta_{y_i,i}) - m)}}{e^{s(\cos(\theta_{y_i,i}) - m)} + \sum_{j \neq y_i} e^{s\cos(\theta_{j,i})}}$, where, s is the given $\|x\|$, $m > 0$
Explanation	–
Algorithm	CosFace [66, 67]
ArcFace loss	
Formula	$L = \frac{1}{N}\sum_i -\log \frac{e^{s(\cos(\theta_{y_i,i} + m))}}{e^{s(\cos(\theta_{y_i,i} + m))} + \sum_{j \neq y_i} e^{s\cos(\theta_{j,i})}}$, where, s is the given $\|x\|$, $m > 0$
Explanation	–
Algorithm	ArcFace [68]

the model pay more attention to the close pairs from different persons. Therefore, contrastive loss function can achieve the matching degree of pairs well, and also can be used to train the model of feature extraction effectively. However, the problem of this loss function is how to select the tuples, due to the extreme imbalance between negative tuples and positive tuples. How to select the appropriate negative tuples is the key and difficult point.

Triplet loss function The input of triple loss [62] is a set of triples $\{(x_i^a, x_i^p, x_i^n)\}_{i=1}^{P}$, where x^a is a randomly selected face image, x^p and x^a belong to the same class (the same person), x^n and x^a belong to different classes (different people). We hope that the learned features can satisfy $\|\mathbf{x}^a - \mathbf{x}^p\|^2 + \alpha \leq \|\mathbf{x}^a - \mathbf{x}^n\|^2$, where α represents the

minimum margin between the distance of samples from the same class and the distance of samples from different classes. Based on this idea, we get triple loss function. By optimizing the loss function, we can extract the features, which ensure that the similar samples are close and the different samples are far away. But the problem of this loss function is how to select triples. Triples that are easy to satisfy the above formula are not very helpful for feature extraction, and triples that are difficult to satisfy the above formula will result in local extremum, and the network may not be able to converge to the optimal value. [62] proposes a selection method, which can be consulted by oneself.

Center loss function Since contrastive loss and triplet loss are very sensitive to the construction and selection of tuples, [63] uses the extracted feature vector as the input of cross entropy loss to train the model through cross entropy loss. However, it is found that cross entropy loss can only separate the categories, but cannot constrain the in-class distance, so samples are scattered relatively sparsely among the separatable boundaries, and misjudgment will occur when cross entropy loss is applied to face recognition tasks. Therefore, [63] proposes center loss function to constrain the distance within a class. And the final loss function is a weighted combination of cross entropy loss and center loss.

A-Softmax loss function [64] uses cross entropy loss to extract and visualize features of a small data set, and find that the extracted features have obvious angular distribution. Thus, the authors thinks that combining cross entropy loss with Euclidean distance is unreasonable. Meanwhile, for the first time, the distance of angular space is used to describe the difference of features within and between classes, and A-Softmax loss function is proposed. Next, we take binary classification as an example to describe the derivation of A-Softmax loss function. In binary classification, the decision boundary of cross entropy loss function is $(W_1 - W_2)\mathbf{x} + b_1 - b_2 = 0$. After constraining $\|W_1\| = \|W_2\| = 1$ and $b_1 = b_2 = 0$, the decision boundary of modified cross entropy loss is $\|\mathbf{x}\|(\cos(\theta_1) - \cos(\theta_2)) = 0$, where $\theta_{a,b}$ is the angle between the weight W_a and the feature $\mathbf{x}_b$. Further, in order to make the intra class features similar and the inter class features different, the condition for class 1 becomes $\cos(m\theta_1) > \cos(\theta_2)$ from $\cos(\theta_1) > \cos(\theta_2)$, and the condition for class 2 becomes $\cos(m\theta_2) > \cos(\theta_1)$ from $\cos(\theta_2) > \cos(\theta_1)$, where $m(m \geq 1)$ is an integer. In this way, the decision boundary has changed from one to two, and a certain margin has been formed between the two decision boundaries, which has achieved the goal we want. By extending this idea to the multi classification problem, A-Softmax loss function is obtained. It should be noted that in order to ensure the monotonic decrease of loss function with angle and the model convergence, the applied A-Softmax loss is adjusted on the expression shown in Table 7. For details, please refer to [64, 65].

AM-Softmax loss function To make the model pay more attention to the angle information obtained from the data and ignore the value of the feature vector, [66, 67] fixed $\|\mathbf{x}\| = s$ and proposed AM-Softmax loss. Considering the binary classification, according to this loss function, the condition for class 1 becomes $\cos(\theta_1) > \cos(\theta_2) + m$, and the condition for class 2 becomes $\cos(\theta_2) > \cos(\theta_1) + m$, where $m > 0$. Similarly, there are two decision boundaries, and there is a certain margin between them.

Table 8 Decision boundaries for class 1 under binary classification case

Loss function	Decision boundary
Cross entropy loss	$(W_1 - W_2)\mathbf{x} + b_1 - b_2 = 0$
Modified cross entropy loss	$\|\mathbf{x}\|(\cos(\theta_1) - \cos(\theta_2)) = 0$
A-Softmax loss	$\|\mathbf{x}\|(\cos(m\theta_1) - \cos(\theta_2)) = 0$
AM-Softmax loss	$s(\cos(\theta_1) - m - \cos(\theta_2)) = 0$
ArcFace loss	$s(\cos(\theta_1 + m) - \cos(\theta_2)) = 0$

ArcFace loss function [68] proposed ArcFace loss function to further improve AM-Softmax loss. In the same case of binary classification, according to this loss function, the condition for class 1 becomes $\cos(\theta_1 + m) > \cos(\theta_2)$, and the condition for class 2 becomes $\cos(\theta_2 + m) > \cos(\theta_1) + m$, where $m > 0$. The decision boundaries become two, and a certain margin is formed between the two decision boundaries.

Comparison Contrastive loss and triplet loss need to construct tuples, which can cause combination explosion on large-scale data sets, further result in the increase of iteration and slow convergence. In addition, the selection of tuples is critical. Although some strategies can be adopted to select tuples, the computational complexity is also increased significantly. Center loss can only constrain the distance within one class, and it needs to be used together with cross entropy loss. As the features extracted by cross entropy loss show a certain angular distribution, A-Softmax loss, AM-Softmax loss and ArcFace loss increase different angular margins based on cross entropy loss. In the case of binary classification, the decision boundaries formed by the three kinds of losses for class 1 are shown in Table 8. A-Softmax loss requires some adjustments to ensure monotony of the function and convergence of the model. AM-softmax loss is easier to achieve, reduces the complex parameter calculation, simplifies the training process and ensures the convergence. Arcface loss is not only easy to realize, but also has better geometric properties. It forms a constant linear angular margin in the whole interval, while A-Softmax loss and AM-Softmax loss can only form a nonlinear angular margin [68].

5 Conclusion

In this paper, a new partition criterion of loss functions is proposed, and 31 loss functions are introduced from five aspects: classification, regression, unsupervised learning of traditional machine learning, object detection, face recognition of deep learning. These loss functions are commonly used in various fields or tasks. By analyzing their formulas, corresponding algorithms, images, and comparing their advantages and disadvantages, we hope to provide help for readers to apply and improve the loss functions.

Acknowledgements This work has been partially supported by Grants from: Science and Technology Service Network Program of Chinese Academy of Sciences (STS Program, No. KFJ-STS-ZDTP-060), National Natural Science Foundation of China (Nos. 71731009, 61472390, 71331005, 91546201), and Beijing Social Science Foundation Project (No.17GLB020).

References

1. Roweis ST, Saul LK (2000) Nonlinear dimensionality reduction by locally linear embedding. Science 290(5500):2323–2326
2. Van Der Maaten L, Postma E, Van den Herik J (2009) Dimensionality reduction: a comparative. J Mach Learn Res 10(66–71):13
3. Yang L, Jin R (2006) Distance metric learning: a comprehensive survey. Mich State Univ 2(2):4
4. Bellet A, Habrard A, Sebban M (2013) A survey on metric learning for feature vectors and structured data. arXiv: 1306.6709
5. Moutafis P, Leng M, Kakadiaris IA (2016) An overview and empirical comparison of distance metric learning methods. IEEE Trans Cybern 47(3):612–625
6. Seber GA, Lee AJ (2012) Linear regression analysis, vol 329. Wiley, Hoboken
7. Safavian SR, Landgrebe D (1991) A survey of decision tree classifier methodology. IEEE Trans Syst Man Cybern 21(3):660–674
8. Liaw A, Wiener M (2002) Classification and regression by randomForest. R News 2(3):18–22
9. Freund Y, Schapire RE (1995) A desicion-theoretic generalization of on-line learning and an application to boosting. In: European conference on computational learning theory. Springer, Berlin, Heidelberg, pp 23–37
10. Domingos P (2012) A few useful things to know about machine learning. Commun ACM 55(10):78–87
11. Girshick R (2015) Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision, pp 1440–1448
12. Ren S, He K, Girshick R, Sun J (2015) Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems, pp 91–99
13. He K, Gkioxari G, Dollár P, Girshick R (2017) Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision, pp 2961–2969
14. Ahonen T, Hadid A, Pietikainen M (2006) Face description with local binary patterns: application to face recognition. IEEE Trans Pattern Anal Mach Intell 28(12):2037–2041
15. Parkhi OM, Vedaldi A, Zisserman A (2015) Deep face recognition. In: Proceedings of the British Machine Vision Conference (BMVC), pp 41.1–41.12
16. Ronneberger O, Fischer P, Brox T (2015) U-net: convolutional networks for biomedical image segmentation. In: International conference on medical image computing and computer-assisted intervention. Springer, Cham, pp 234–241
17. Badrinarayanan V, Kendall A, Cipolla R (2017) Segnet: a deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans Pattern Anal Mach Intell 39(12):2481–2495
18. Bahdanau D, Cho K, Bengio Y (2014) Neural machine translation by jointly learning to align and translate. arXiv:1409.0473
19. Wu Y, Schuster M, Chen Z, Le Q V, Norouzi M, Macherey W, Klingner J (2016) Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144
20. Zhang X, Zhao J, LeCun Y (2015) Character-level convolutional networks for text classification. In: Advances in neural information processing systems, pp 649–657
21. Lai S, Xu L, Liu K, Zhao J (2015) Recurrent convolutional neural networks for text classification. In: Twenty-ninth AAAI conference on artificial intelligence
22. Collobert R, Sinz F, Weston J, Bottou L (2006) Trading convexity for scalability. In: Proceedings of the 23rd international conference on machine learning, pp 201–208
23. Huang X, Shi L, Suykens JA (2013) Support vector machine classifier with pinball loss. IEEE Trans Pattern Anal Mach Intell 36(5):984–997

24. Shen X, Niu L, Qi Z, Tian Y (2017) Support vector machine classifier with truncated pinball loss. Pattern Recognit 68:199–210
25. Xu G, Cao Z, Hu BG, Principe JC (2017) Robust support vector machines based on the rescaled hinge loss function. Pattern Recognit 63:139–148
26. Dietterich TG, Bakiri G (1994) Solving multiclass learning problems via error-correcting output codes. J Artif Intell Res 2:263–286
27. Allwein EL, Schapire RE, Singer Y (2000) Reducing multiclass to binary: a unifying approach for margin classifiers. J Mach Learn Res 1(12):113–141
28. Lee Y, Lin Y, Wahba G (2004) Multicategory support vector machines: theory and application to the classification of microarray data and satellite radiance data. J Am Stat Assoc 99(465):67–81
29. Weston J, Watkins C (1999) Support vector machines for multi-class pattern recognition. Esann 99:219–224
30. Crammer K, Singer Y (2001) On the algorithmic implementation of multiclass kernel-based vector machines. J Mach Learn Res 2(12):265–292
31. Liu Y, Yuan M (2011) Reinforced multicategory support vector machines. J Comput Graph Stat 20(4):901–919
32. Zhang C, Liu Y (2013) Multicategory large-margin unified machines. J Mach Learn Res 14(1):1349–1386
33. Rosenblatt F (1958) The perceptron: a probabilistic model for information storage and organization in the brain. Psychol Rev 65(6):386
34. Minsky M, Papert SA (2017) Perceptrons: an introduction to computational geometry. MIT Press, Cambridge
35. Kleinbaum DG, Dietz K, Gail M, Klein M, Klein M (2002) Logistic regression. Springer, New York
36. Gasso G (2019) Logistic regression
37. Kohonen T, Barna G, Chrisley R (1988) Statistical pattern recognition with neural networks: benchmarking studies. In: IEEE International Conference on Neural Networks, vol 1, pp 61–68
38. Berger AL, Pietra VJD, Pietra SAD (1996) A maximum entropy approach to natural language processing. Comput Linguist 22(1):39–71
39. Friedman J, Hastie T, Tibshirani R (2000) Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). Ann Stat 28(2):337–407
40. Cortes C, Vapnik V (1995) Support vector machine. Mach Learn 20(3):273–297
41. Deng N, Tian Y, Zhang C (2012) Support vector machines: optimization based theory, algorithms, and extensions. Chapman and Hall/CRC, Boca Raton
42. Steinwart I (2003) Sparseness of support vector machines. J Mach Learn Res 4(11):1071–1105
43. Yuille AL, Rangarajan A (2002) The concave-convex procedure (CCCP). In: Advances in neural information processing systems, pp 1033–1040
44. Huber PJ (1964) Robust estimation of a location parameter. Ann Math Stat 35(1):73–101
45. Friedman JH (2001) Greedy function approximation: a gradient boosting machine. Ann Stat pp 1189-1232
46. Koenker R, Hallock KF (2001) Quantile regression. J Econ Perspect 15(4):143–156
47. Drucker H, Burges CJ, Kaufman L, Smola AJ, Vapnik V (1997) Support vector regression machines. In: Advances in neural information processing systems, pp 155–161
48. Jain AK, Dubes RC (1988) Algorithms for clustering data. Prentice-Hall Inc, Upper Saddle River
49. Aloise D, Deshpande A, Hansen P, Popat P (2009) NP-hardness of Euclidean sum-of-squares clustering. Mach Learn 75(2):245–248
50. Cox TF, Cox MA (2000) Multidimensional scaling. Chapman and Hall/CRC, Boca Raton
51. Tenenbaum JB, De Silva V, Langford JC (2000) A global geometric framework for nonlinear dimensionality reduction. Science 290(5500):2319–2323
52. Wold S, Esbensen K, Geladi P (1987) Principal component analysis. Chemometr Intell Lab Syst 2(1–3):37–52
53. Abdi H, Williams LJ (2010) Principal component analysis. Wiley Interdiscip Rev Comput Stat 2(4):433–459
54. Lin TY, Goyal P, Girshick R, He K, Dollár P (2017) Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision, pp 2980–2988
55. Yu J, Jiang Y, Wang Z, Cao Z, Huang T (2016) Unitbox: An advanced object detection network. In Proceedings of the 24th ACM international conference on Multimedia, pp 516-520

56. Rezatofighi H, Tsoi N, Gwak J, Sadeghian A, Reid I, Savarese S (2019) Generalized intersection over union: a metric and a loss for bounding box regression. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 658–666
57. Shrivastava A, Gupta A, Girshick R (2016) Training region-based object detectors with online hard example mining. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 761–769
58. Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 580–587
59. He K, Zhang X, Ren S, Sun J (2015) Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Trans Pattern Anal Mach Intell 37(9):1904–1916
60. Chopra S, Hadsell R, LeCun Y (2005) Learning a similarity metric discriminatively, with application to face verification. In: 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05), vol 1. IEEE, pp 539–546
61. Hadsell R, Chopra S, LeCun Y (2006) Dimensionality reduction by learning an invariant mapping. In: 2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06), vol 2. IEEE, pp 1735–1742
62. Schroff F, Kalenichenko D, Philbin J (2015) Facenet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 815–823
63. Wen Y, Zhang K, Li Z, Qiao Y (2016) A discriminative feature learning approach for deep face recognition. In: European conference on computer vision. Springer, Cham, pp 499–515
64. Liu W, Wen Y, Yu Z, Li M, Raj B, Song L (2017) Sphereface: deep hypersphere embedding for face recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 212–220
65. Liu W, Wen Y, Yu Z, Yang M (2016) Large-margin softmax loss for convolutional neural networks. In: ICML vol 2(3), p 7
66. Wang H, Wang Y, Zhou Z, Ji X, Gong D, Zhou J, Liu W (2018) Cosface: Large margin cosine loss for deep face recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5265–5274
67. Wang F, Cheng J, Liu W, Liu H (2018) Additive margin softmax for face verification. IEEE Signal Process Lett 25(7):926–930
68. Deng J, Guo J, Xue N, Zafeiriou S (2019) Arcface: Additive angular margin loss for deep face recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4690–4699