Propaedeutics and Overview of Deep Learning

www.huawei.com

HUAWEI

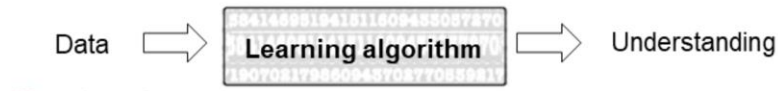
Objectives

- After completing this course, you will be able to:
  - Understand learning algorithms and common machine learning algorithms.
  - Understand the concepts of the hyperparameter and validation set.
  - Master maximum likelihood estimation and Bayes estimation.
  - Understand the definition and development of neural networks.
  - Master the types of neural networks.

Contents

Learning Algorithms (1)

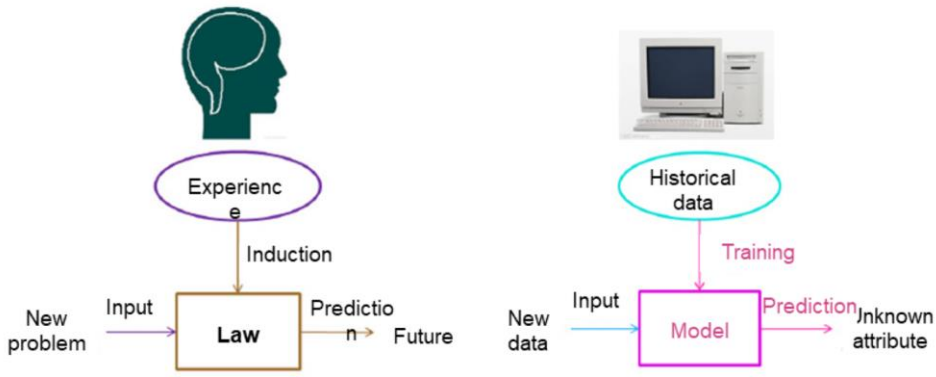
- Machine learning (including deep learning) is a study of learning algorithms. A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E.



- Deep learning is a specific kind of machine learning. To understand deep learning well, we should have a solid understanding of the basic principles of machine learning.
  - Task T: how the machine learning system should process a sample. A sample is a collection of features that have been quantitatively measured from some object or event that we want the machine learning system to process, such as classification, regression, and machine translation.
  - Performance measure P: evaluates the abilities of a machine learning algorithm, such as accuracy and error rate.
  - Experience E: Most learning algorithms can be understood as being allowed to experience an entire dataset. Some machine learning algorithms do not just experience a fixed dataset. For example, reinforcement learning algorithms interact with an environment, so there is a feedback loop between the learning system and its experiences. Machine learning algorithms can be broadly categorized as unsupervised or supervised by what kind of experience they are allowed to have during the learning process.
- To learn Go:
  - Experience E_1: playing with itself — unsupervised and indirect learning
  - Experience E_2: inquiring humans when playing with itself — semi-supervised learning

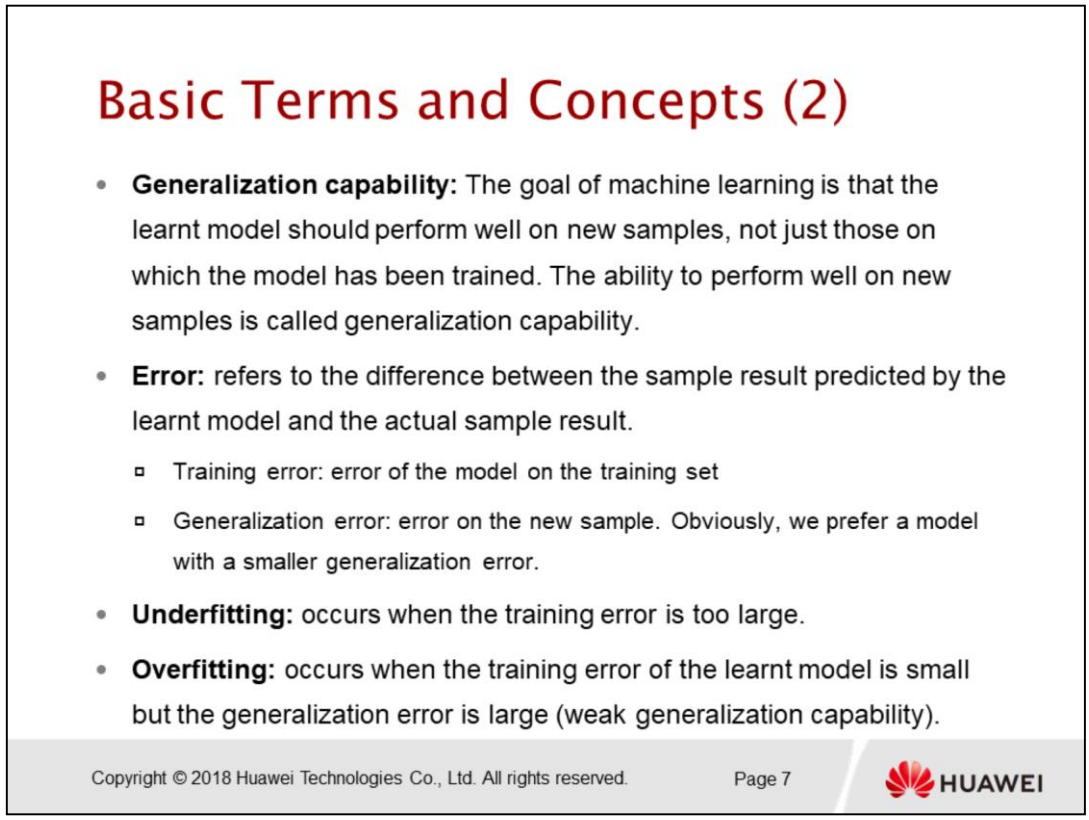
    - Experience E_3: historical human games — supervised and direct learning
- Handwriting recognition issue: Task T: Identify handwriting text. Performance measure P: classification accuracy. Experience E: classified example library (supervised and direct learning).
- Robots' desire to advance: Look for new games and practice their skills through tiny changes in the same situation, enriching their training examples.

Learning Algorithms (2)



Basic Terms and Concepts (1)

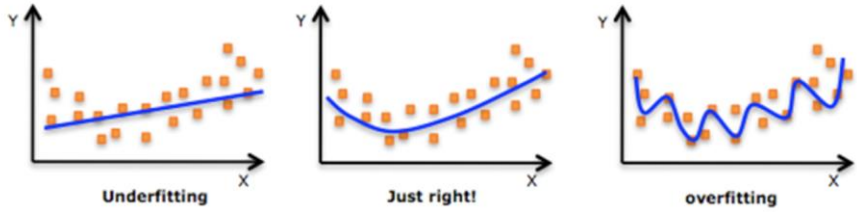
- **Dataset:** refers to a set of data used in machine learning tasks. Each piece of data is called a sample. The event or attribute that reflects the performance or nature of a sample in a certain aspect is called a **feature**.
- **Training set:** refers to a dataset used in the training process, where each sample is referred to as a training sample. The process of learning a model from data is called **learning (training)**.
- **Test set:** Test refers to the process of using the learnt model for prediction. The dataset used is called a test set, and each sample is called a test sample.



- Once the form of a problem's hypothesis is given, all possible functions constitute a space, which is hypothesis space. The problem of machine learning is to search for a suitable fitting function in the hypothetical space.
- Overfitting: It occurs frequently in complex mathematical models. To prevent overfitting, we can simplify mathematical models, end training before overfitting, or use dropout/weight decay methods.
- Underfitting: Underfitting occurs if the mathematical model is too simple or the training time is too short. To solve the former issue, use a more complex model. To solve the latter issue, extend the training time.

Basic Terms and Concepts (3)

- **Capacity of a model:** refers to the ability to fit a wide variety of functions. Machine learning algorithms will generally perform best when their capacity is appropriate for the true complexity of the task they need to perform and the amount of training data they are provided with. Models with an insufficient capacity are unable to solve complex tasks. Models with a high capacity can solve complex tasks, but when their capacity is higher than that is needed to solve the present task, they may overfit.



Failing to learn features

Learning noise

- The effective capacity is restricted by algorithms, parameters, and regularization.

Basic Terms and Concepts (4)

- Many kinds of tasks can be solved with machine learning. The following describes the most typical and common types.
  - **Classification:** The computer program is asked to specify which of k categories some input belongs to. To solve this task, the learning algorithm is usually asked to produce a function $f: R^n \rightarrow (1,2, \ldots, k)$. For example, the image classification algorithm in computer vision is to solve a classification task.
  - **Regression:** In this type of task, the computer program is asked to predict a numerical value given some input. The learning algorithm is usually asked to produce a function $f: R^n \rightarrow R$. An example of a regression task is the prediction of the expected claim amount that an insured person will make (used to set insurance premiums), or the prediction of future prices of securities.
- **Classification and regression are two major types of prediction tasks. The outputs of classification are discrete type values while the outputs of regression are consecutive values.**

Basic Terms and Concepts (5)

- Terms:
  - P: positive examples, involving major interesting classes
  - N: negative examples (other examples)
  - TP: true positive examples (positive examples that are correctly classified by the classifier)
  - TN: true negative examples (negative examples that are correctly classified by the classifier)
  - FP: false positive examples (negative examples that are incorrectly marked as positive examples)
  - FN: false negative examples (positive examples that are incorrectly marked as negative examples)

Predicted / Observed	Yes	No	Total
Yes	TP	FN	P
No	FP	TN	N
Total	P'	N'	$P+N$

Confusion matrix

- Confusion matrix: $CM_{i,j}$ element of the first i-th line and j-th column is the number of examples known to be in class i but marked as j by the classifier.
  - Ideally, for a highly accurate classifier, most data samples should be represented by the elements on the diagonal from $CM_{1,1}$ and $CM_{m,m}$. Other elements are 0 or close to 0. In other words, FP and FN are close to 0.

Basic Terms and Concepts (6)

Measure	Formula
Accuracy and correct classification rate	$\frac{TP + TN}{P + N}$
Error rate and false classification rate	$\frac{FP + FN}{P + N}$
Sensitivity, true positive rate, and $recall$	$\frac{TP}{P}$
Specificity and true negative rate	$\frac{TN}{N}$
$Precision$	$\frac{TP}{TP + FP}$
F score: harmonic mean of precision and recall	$\frac{2 \times precision \times recall}{precision + recall}$
F_β (β is a non-negative real number)	$\frac{(1 + \beta^2) \times precision \times recall}{\beta^2) \times precision + recall}$

Example of Machine Learning Performance Assessment

- We have trained a machine learning model to identify that whether the object in an image is a cat. Now we use 200 images to test its performance. Among 200 images, 170 contain cats and 30 do not contain cats. The identification result of the model is that 160 images contain cats and 40 do not contain cats.

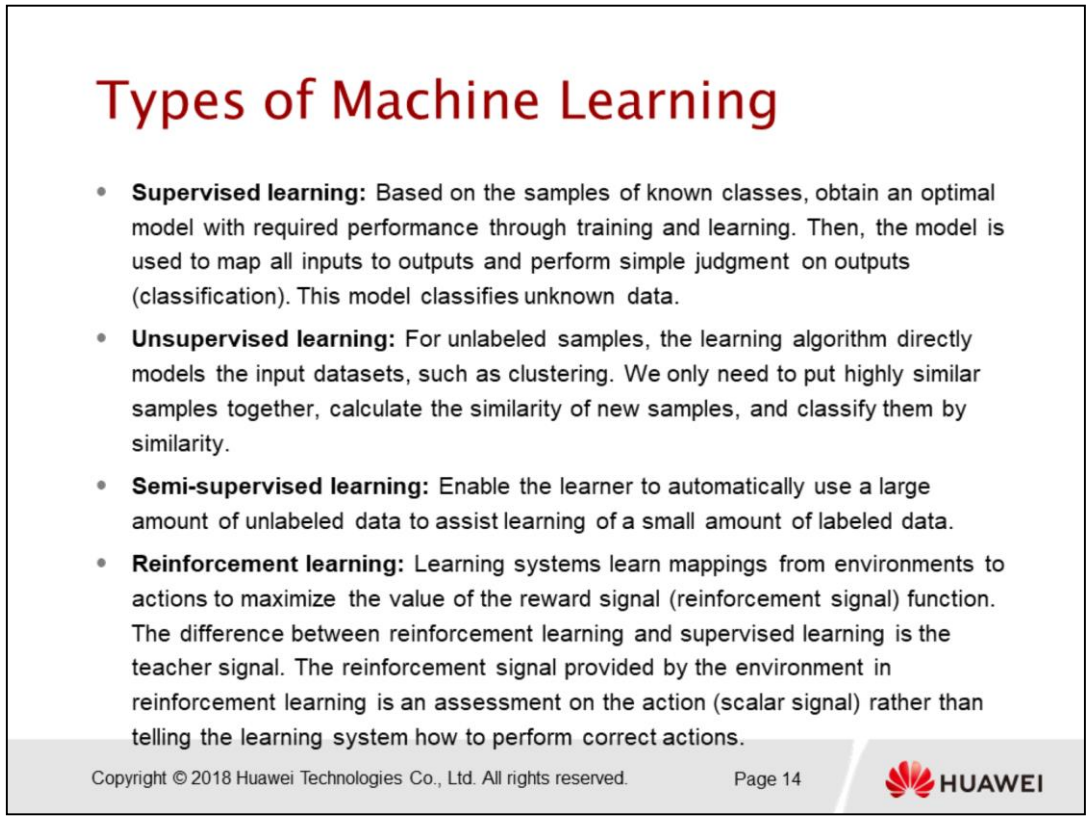
Precision: $P = \frac{TP}{TP+FP} = \frac{140}{140+20} = 87.5.\%$

Recall: $R = \frac{TP}{P} = \frac{140}{170} = 93.3\%.$

Accuracy: $ACC = \frac{TP+TN}{P+N} = \frac{140+10}{170+30} = 85\%$

Predicted / Observed	*Yes*	*No*	Total
Yes	140	30	170
No	20	10	30
Total	160	40	200

Contents

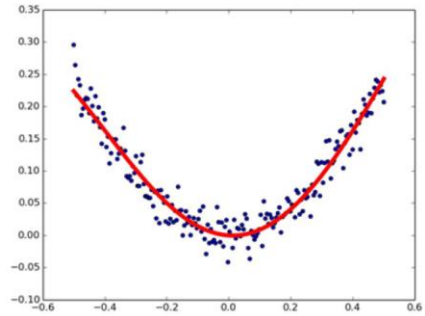


- Supervised learning: We give a computer a bunch of choice questions (training samples) and provide standard answers. The computer tries to adjust its model parameters to make predictions closer to standard answers. In this way, it learns how to deal with this type of problem. Then it helps us solve choice questions whose answers are not given (test samples).
- Non-supervised learning: We give a computer a bunch of choice questions (training samples), but do not provide standard answers. The computer tries to analyze the relationships between these questions and classify them. It does not know the answers to these questions, but it thinks that the answers to the questions in the same class should be the same.
- Semi-supervised learning: Traditional supervised learning uses a large number of labeled training samples to establish a model for predicting the labels of new samples. For example, in a classification task, a label is the type of a sample while in a regression task, a label is a real-valued output of the sample. As our data collection and storage capabilities are developing, we have a large amount of unlabeled data in many practical tasks. Labeling the data is labor-consuming and time-consuming. For example, for web page recommendation, we need users to mark web pages they like, but only a few users are willing to spend a lot of time doing this. Then we only get limited labeled web page data and a large amount of unlabeled web page data.
- Reinforcement learning: We give a computer a bunch of choice questions (training samples), but do not provide standard answers. It tries to solve these questions, and

we judge whether the answers are correct as teachers. If the computer generates more correct answers, we offer more rewards. The computer adjusts its model parameters to make its predictions correct and obtain more rewards. Not strictly speaking, reinforcement learning can be understood as non-supervised learning plus supervised learning.

Common Machine Learning Algorithm: Linear Regression

- Linear regression: a statistical analysis method to determine the quantitative relationships between two or more variables through regression analysis in mathematical statistics.



- Unary linear regression analysis only involves one independent variable and one dependent variable and their relationship can be approximately represented by a straight line. Multivariable linear regression analysis involves two or more independent variables and the relationship between independent and dependent variables are linear. The learning result is not necessarily a straight line. It is a straight line when the variable x is one-dimensional and a hyperplane when the variable is high-dimensional. For example, the price of an apartment is determined by a variety of factors such as the area, layout, and location. Prediction of the apartment price based on these factors can be abstracted into a linear regression problem.

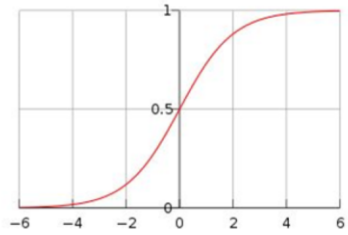
Common Machine Learning Algorithm: Logistic Regression

- Logistic regression: The logistic regression model is used to solve classification problems. Model definition:

$$P(Y = 1|x) = \frac{e^{wx+b}}{1 + e^{wx+b}}$$

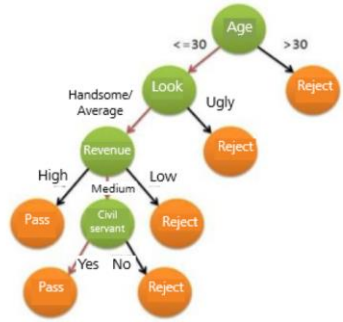
$$P(Y = 0|x) = \frac{1}{1 + e^{wx+b}}$$

where w is the weight, b is the bias, $wx + b$ is the linear function of x. Compare the two probability values. x belongs to the class with a larger probability value.



Common Machine Learning Algorithm: Decision Tree

- Decision tree: A decision tree is a tree-like structure (a binary tree or a non-binary tree). Each non-leaf node represents a test on a feature attribute. Each branch represents the output of a feature attribute at a certain value range, and each leaf node stores a class. To use the decision tree, start from the root node, test the feature attributes of the items to be classified, select the output branches, and use the class stored on the leaf node as the final result.

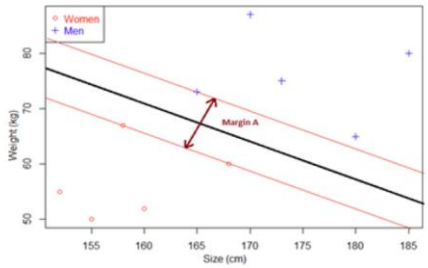


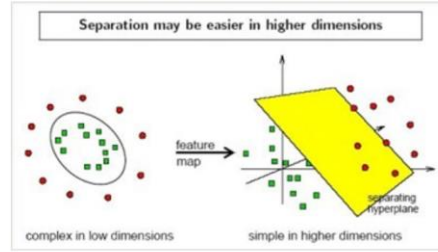
HUAWEI

- How to construct a decision-tree is very important. We should determine the topological structure of feature attributes by selecting attributes based on quantitative values. The key step is to split attributes. That is, different branches are constructed based on the differences of a feature attribute on a node.
- The decision tree learning algorithm is used to generate decision trees. Common learning algorithms include ID3, C4.5, and CART.

Common Machine Learning Algorithm: Support Vector Machine

- A support vector machine (SVM) is a two-class classification model, and its basic model is a linear classifier defined in the feature space with the largest interval. SVM also includes kernel tricks that make it a non-linear classifier. The SVM learning algorithm is the optimum solution to convex quadratic programming.





HUAWEI

- The main ideas of SVM include two points:
  - In case of linear inseparability, non-linear mapping algorithms are used to moving the linearly inseparable samples of a low-dimensional space into a high-dimensional feature space. In this way, samples become linearly separable. Then the linear algorithm can be used to analyze the non-linear features of samples.
  - Based on the structural risk minimization principle, an optimal hyperplane is constructed in the feature space, so that the learner is optimized globally, and the expectation of the whole sample space satisfies an upper boundary with a certain probability.

Common Machine Learning Algorithm: Naive Bayes

- **Naive Bayes algorithm:** A classification method based on **Bayes' theorem** and **attribute conditional independence assumption**. For a given training set, the algorithm learns the joint probability distribution of inputs/outputs based on the attribute conditional independence assumption, and then based on this model, obtain the output y with the maximum posterior probability for a given input x using Bayes' theorem. According to Bayes' theorem, in a classification problem, for a given sample feature X, the probability that the sample belongs to the class H is:

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}$$

  X is the data sample and generally described with the measurement value of n attribute sets. H is a hypothesis, for example, data sample X belonging to one certain class C. $P(H \mid X)$ is the posterior probability or H's posterior probability under the condition X. $P(H)$ is the prior probability or H's prior probability and is independent of X. $P(X)$ is the prior probability of X.

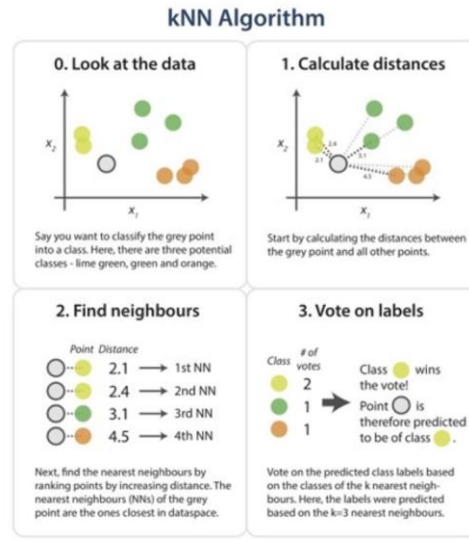
- Based on the naive assumption, namely attribute conditional independence assumption, and the law of total probability, the basic formula of Naive Bayes classification is as follows:

$$P(C_k|X) = \frac{P(X|C_k)P(C_k)}{P(X)} = \frac{\prod_{k=1}^{M} P(X_i|C_k)P(C_k)}{\sum_k P(C_k)\prod_{k=1}^{M} P(X_i|C_k)}$$

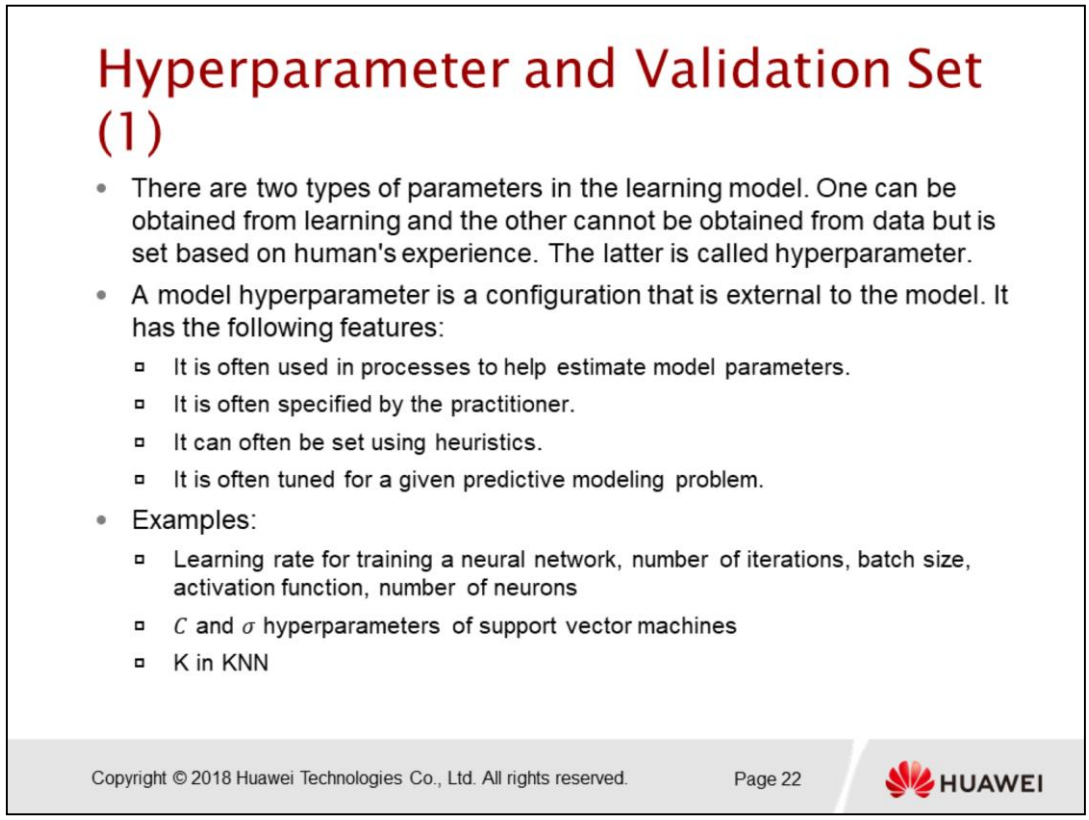
- **Class conditional independence:** The Bayes classifier assumes that the effect of an attribute value on a given class is independent of the values of other attributes. This assumption is made to simplify the calculation and becomes "naive" in this sense.
- Bayes classifier can be applied to large databases, featuring high accuracy and a fast speed.

Common Machine Learning Algorithm: KNN

- The k-nearest neighbors (KNN) classification algorithm is a theoretically mature method and one of the simplest machine learning algorithms. If the majority of k samples that are most similar to one sample (nearest neighbors in the feature space) belongs to one class, the sample also belongs to this class.



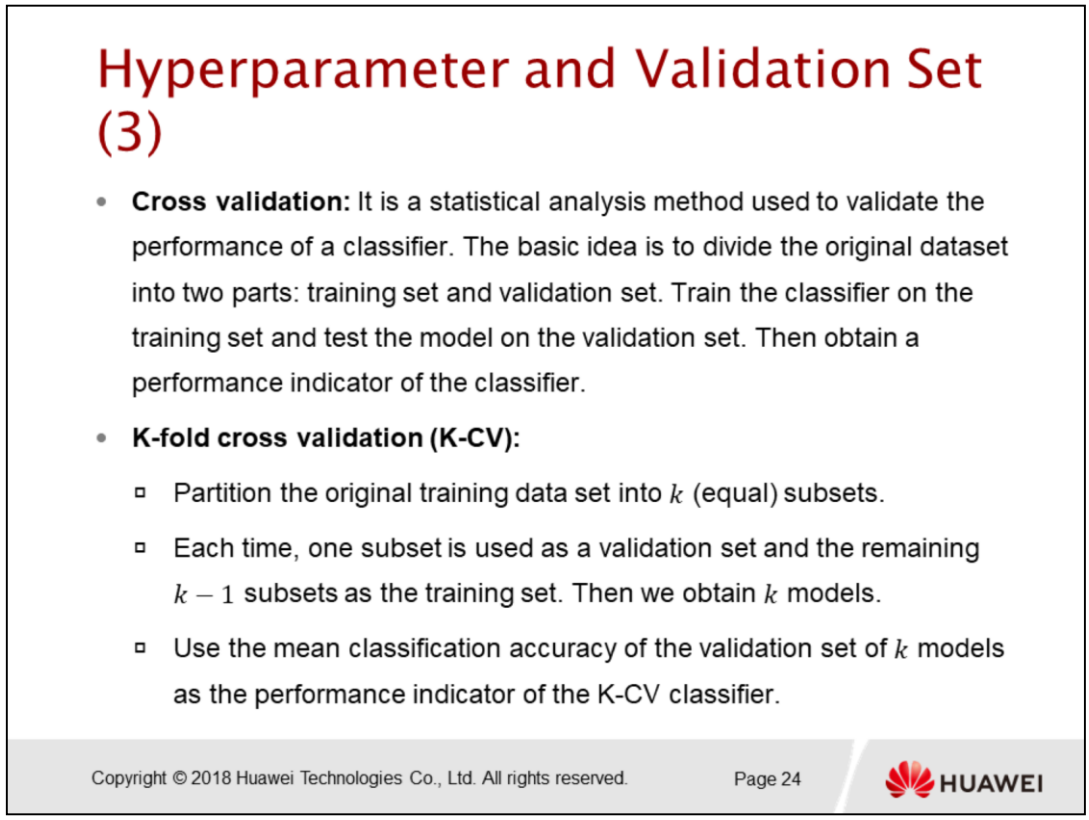
Contents



- A parameter is the part of the model that is learnt from historical training data and key to machine learning algorithms. It has the following features:
  - It is required by the model when making predictions.
  - Its value defines the skill of the model on your problem.
  - It is estimated or learnt from data.
  - It is often not set manually by the practitioner.
  - It is often saved as part of the learnt model.
- Examples:
  - Weights in an artificial neural network
  - Support vectors in a support vector machine
  - Coefficients in a linear regression or logistic regression

Hyperparameter and Validation Set (2)

- The procedure for searching for hyperparameters:
  - Divide a dataset into a training set, verification set, and test set.
  - Optimize the model parameters in the training set based on the performance indicators of the model.
  - Search for the model hyperparameters in the training set based on the performance indicators of the model.
  - Step 2 and step 3 are performed alternately. Finally, obtain parameters and hyperparameters of the model and assess the model in the test set.
- The search process in step 3 requires search algorithms, including:
  - Grid search
  - Random search
  - Heuristic intelligent search
  - Bayesian search

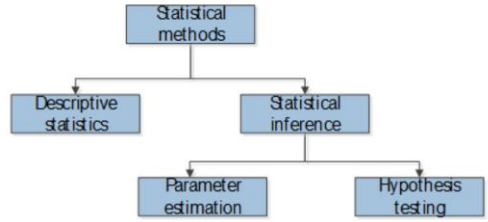


- Dividing the dataset into a fixed training set and a fixed test set can be problematic if it results in the test set being small. A small test set implies statistical uncertainty around the estimated average test error, making it difficult to claim that algorithm A works better than algorithm B on the given task. When the dataset has hundreds of thousands of examples or more, this is not a serious issue. When the dataset is too small, alternative procedures enable one to use all the examples in the estimation of the mean test error, at the price of increased computational cost.
- K-CV: Generally, the value of k is greater than or equal to 2. In practice, the value is greater than or equal to 3. The value 2 is used only when the original data set is small. $K-CV$ can effectively avoid over-learning and under-learning, and the final result is also persuasive.

Contents

Parameter Estimation

- **Parameter estimation:** Suppose that there is a statistical population and its distribution function is $F(x, \theta)$, where θ is an unknown parameter. Now we sample the population and obtain the sample $X_1, X_2, \ldots, X_n$. We need to find an estimator $\hat{\theta}$ of the parameter θ or estimate θ's function $g(\theta)$. This problem is called **parameter estimation**, including **point estimation** and **interval estimation**.
- **The position of parameter estimation in statistical methods:**



HUAWEI

- Point estimation: μ = 7; interval estimation: μ within [5,12]

Point Estimator

- In one area, the fetal weight is $X \sim N(\mu, \sigma^2)$ (μ and σ^2 are unknown). Randomly select 100 babies and obtain their weight data

  10,7,5,6,6,5.2, ...

  We have 100 figures and how do we estimate μ and σ?
- **Point estimator:** We need to construct a function of an appropriate sample: $T(X_1, X_2, \ldots, X_n)$. Once we have a sample, we can input the value into the function to obtain a result which can be used as the estimator of μ. $T(X_1, X_2, \ldots, X_n)$ is the parameter's point estimator.
- Problems: Which estimator can be used to estimate μ? How do we asset the estimators?
  - Mean of the samples
  - Median of the samples
  - Other statistics

- Estimator with normal distribution: $\hat{\mu} = \bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$, $\widehat{\sigma^2} = S^2 = \frac{1}{n-1}\left(\sum_{i=1}^{n} X_i^2 - n\bar{X}^2\right)$.
- Methods to search for estimators:
  - Maximum likelihood estimation
  - Bayes method
  - Method of moments

Statistics Assessment Standard: Unbiased

- **Unbiased:** The bias of an estimator is defined as

$$bias(\hat{\theta}) = E(\hat{\theta}) - \theta$$

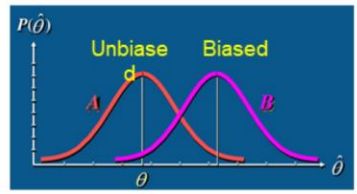
An estimator $\hat{\theta}$ is said to be unbiased if $bias(\hat{\theta}) = 0$, which implies that $E(\hat{\theta}) = \theta$.

An estimator $\hat{\theta}$ is said to be asymptotically unbiased if $\lim bias(\hat{\theta}) = 0$, which implies that $\lim E(\hat{\theta}) = \theta$.

- Example: Suppose $(X_1, X_2, \dots, X_n)$ is a sample from the population X. $EX = \mu$, $DX = \sigma^2$. Prove:

(1) $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ is an unbiased estimator of μ.

(2) $S^2 = \frac{1}{n-1}\left(\sum_{i=1}^{n} X_i^2 - n\bar{X}^2\right)$ is an unbiased estimator of σ^2.



- The practical meaning of unbiased estimators is that they do not have system errors.

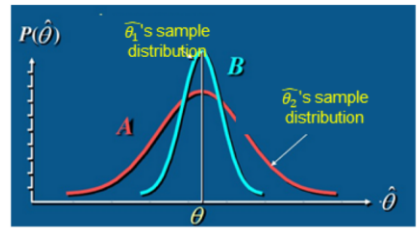
Statistics Assessment Standard: Valid

- Example: X_1, X_2, and X_3 is an sample of the population X. Are the following statistics the unbiased estimators of the mean μ? Which one is optimal?

$$\widehat{\mu_1} = \frac{2}{5}X_1 + \frac{1}{10}X_2 + \frac{1}{2}X_3,\ \widehat{\mu_2} = \frac{1}{3}X_1 + \frac{3}{4}X_2 - \frac{1}{12}X_3,$$

$$\widehat{\mu_3} = \frac{1}{2}X_1 + \frac{1}{3}X_2 + \frac{1}{6}X_3,\ \widehat{\mu_4} = \frac{1}{5}X_1 + \frac{1}{10}X_2 + \frac{7}{10}X_3.$$

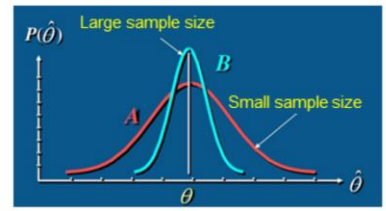
- **Valid:** Suppose $\widehat{\theta_1}$ and $\widehat{\theta_2}$ are two unbiased estimators of θ. If $D(\widehat{\theta_1}) < D(\widehat{\theta_2})$, $\widehat{\theta_1}$ is more valid than $\widehat{\theta_2}$.



- Only unbiased estimators can be regarded as valid or invalid.
- $D(\widehat{\mu_1}) = D\left(\frac{2}{5}X_1 + \frac{1}{10}X_2 + \frac{1}{2}X_3\right) = \frac{4}{25}D(X_1) + \frac{1}{100}D(X_2) + \frac{1}{4}D(X_3) = \frac{21}{50}DX$. Similarly, $D(\widehat{\mu_2}) = \frac{49}{72}DX$, $D(\widehat{\mu_3}) = \frac{7}{18}DX$, $D(\widehat{\mu_4}) = \frac{27}{50}DX$. Therefore, $\widehat{\mu_3}$ is the most valid one.

Statistics Assessment Standard: Consistent

- **Consistent:** With probability P, $\hat{\theta} \to \theta(n \to \infty)$, i.e. for $\forall \epsilon > 0$, $\lim_{n\to\infty} P(|\hat{\theta} - \theta| > \varepsilon)=0$.
- Example: $EX = \mu$, $DX = \sigma^2$, $X_1 \quad X_2, \ldots, X_n$ is a sample from X, $\widehat{\mu_1}=\frac{1}{n}\sum_{i=1}^{n} X_i$, $\widehat{\mu_2}=\frac{1}{k}\sum_{i=1}^{k} X_i$, $k < n$. Prove that $\widehat{\mu_1}$ and $\widehat{\mu_2}$ are unbiased estimators of μ and $\widehat{\mu_1}$ is more valid than $\widehat{\mu_2}$.



PROOF: $E(\widehat{\mu_1}) = E\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \frac{1}{n} \times n\mu = \mu.$

$E(\widehat{\mu_2}) = E\left(\frac{1}{k}\sum_{i=1}^{k} X_i\right) = \frac{1}{k} \times k\mu = \mu.$

$D(\widehat{\mu_1}) = D\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \frac{1}{n^2}\sum_{i=1}^{n} D(X_i) = \frac{\sigma^2}{n}$, $D(\widehat{\mu_2}) = \frac{\sigma^2}{k}$.

Since $n > k$, $D(\widehat{\mu_1}) < D(\widehat{\mu_2})$ and $\widehat{\mu_1}$ is more valid.

HUAWEI

- Consistency ensures that the estimator bias will decrease as the number of data samples increases. The consistent estimator is superior when the sample size n is large enough.

Contents

Maximum Likelihood Estimation (1)

- Suppose that an urn contains a number of black balls and a number of white balls. It is known that the ratio is 1:3. However, it is not known whether the urn has more black balls or more white balls. If sampling is performed for n times, the probability of the number of black balls as x is:

$$p(x;p) = \binom{n}{x} p^x q^{n-x},$$

where $q = 1 - p$. According to the assumption, $p = \frac{1}{4}$ or $\frac{3}{4}$. If $n = 3$, x's probability under p is:

x	0	1	2	3
$P(x, \frac{3}{4})$	$\frac{1}{64}$	$\frac{9}{64}$	$\frac{27}{64}$	$\frac{27}{64}$
$P(x, \frac{1}{4})$	$\frac{27}{64}$	$\frac{27}{64}$	$\frac{9}{64}$	$\frac{1}{64}$

$$\hat{p}(x) = \begin{cases} \frac{1}{4}, x = 0,1 \\ \frac{3}{4}, x = 2,3 \end{cases}.$$

- Example: A student went out hunting with a hunter, and a hare ran in front of them. One shot was fired and the hare fell down. Do you think who shot the hare? You might think that the chance of a hunter hitting the hare by one shot is usually greater than that of the student. So the hare may be shot by the hunter. This is the basic idea of the maximum likelihood estimation.

Maximum Likelihood Estimation (2)

- **Maximum likelihood estimation principle:** Suppose that $X_1, X_2, \ldots, X_n$ is a sample from the population X. Its joint density or joint distribution function is $f(x_1, x_2, \cdots, x_n; \theta)$. Define the likelihood function as:

$$L(\theta) = f(x_1, x_2, \cdots, x_n; \theta),$$

  where $x_1, x_2, \cdots, x_n$ is the observed value of the sample. $L(\theta)$ is a function of parameter θ and can be used to assess the possibility of θ generating $x_1, x_2, \cdots, x_n$.

- Maximum likelihood estimation uses $\hat{\theta}$ that maximizes $L(\theta)$ to estimate θ.

$$L(\hat{\theta}) = \max_{\theta} L(\theta),$$

  $\hat{\theta}$ is the maximum likelihood estimate of θ. The statistic $\hat{\theta}(X_1, X_2, \ldots, X_n)$ is the maximum likelihood estimator of θ.

- The probability of A is related to $\theta \in \Theta$. If θ changes, $P(A)$ also changes. Therefore $P(A|\theta)$ denotes the probability of A. If A occurs, θ from Θ maximizes $P(A|\theta)$. This is the idea of maximum likelihood estimation.

Maximum Likelihood Estimation Example

- Example: Suppose that X_1, X_2,..., X_n is a sample from the population $X \sim B(1, p)$. Find the maximum likelihood estimator of the parameter p.

  SOLUTION: Likelihood function:

$$L(p) = f(x_1, x_2, \ldots, x_n; p) = \prod_{i=1}^{n} p^{x_i}(1-p)^{1-x_i} = p^{\sum_{i=1}^{n} x_i}(1-p)^{n-\sum_{i=1}^{n} x_i}.$$

  Log likelihood function:

$$\ln L(p) = \sum_{i=1}^{n} x_i \ln(p) + (n - \sum_{i=1}^{n} x_i)\ln(1-p).$$

  Derive the function with respect to p and let it be 0:

$$\frac{d \ln L(p)}{dp} = \frac{1}{p}\sum_{i=1}^{n} x_i - \frac{1}{1-p}(n - \sum_{i=1}^{n} x_i) = 0.$$

  $\hat{p} = \bar{x}$. Therefore the maximum likelihood estimator of p is:

$$\hat{p}(X_1, X_2, \ldots, X_n) = \frac{1}{n}\sum_{i=1}^{n} X_i = \bar{X}.$$

- Common procedure:
  - Obtain the joint distribution (or joint density) function based on the population distribution.
  - Regard the independent variable in the joint distribution function as a known constant and parameter θ as the independent variable and then obtain the likelihood function $L(\theta)$.
  - Find the maximum point of the likelihood function $L(\theta)$ (generally find the maximum point of $lnL(\theta)$), namely θ's maximum likelihood estimator.
  - In the maximum point expression, input the sample value to obtain the maximum likelihood estimator of the parameter.

Contents

Bayes' Theorem

- Bayes' theorem calculates the posterior probability $P(h|D)$ with $P(D)$, $P(h)$, and $P(D|h)$. The formula is as follows:

$$P(h \mid D) = \frac{P(D \mid h)P(h)}{P(D)}$$

D is the data sample and generally described using measurement values of n attribute sets. h is a hypothesis, for example, data sample D belonging to one certain class C. $P(h \mid D)$ is the posterior probability or h's posterior probability given that D is true. $P(h)$ is the prior probability or h's prior probability and is independent of D. $P(D)$ is D's prior probability.

Maximum a Posteriori Hypothesis

- $P(h|D)$ is the posterior probability of the hypothesis h. The hypothesis h that maximizes $P(h|D)$ is the maximum a posteriori (MAP) hypothesis.

  Formally, MAP supposes that h_{MAP} satisfies: $h_{MAP} = \underset{h \in H}{\operatorname{argmax}} P(h|D) = \underset{h \in H}{\operatorname{argmax}} \frac{P(D|h)P(h)}{P(D)} = \underset{h \in H}{\operatorname{argmax}} P(D|h)P(h)$

- In practice, we usually cannot obtain the prior probability of each hypothesis. We only assume that all hypotheses in the hypothesis space are equally likely.$P(h)$ is a constant and $h_{MAP} = \underset{h \in H}{\operatorname{argmax}} P(D|h)P(h) = \underset{h \in H}{\operatorname{argmax}} P(D|h) = h_{ML}$

  where $P(D \mid h)$ is the likelihood of data D given h. Therefore the hypothesis that maximizes $P(D|h)$ is the maximum likelihood hypothesis h_{ML}.

- If hypotheses in the hypothesis space are equally likely, h_{MAP} = h_{ML}.

Challenges Motivating Deep Learning

- Curse of dimensionality
- Local constancy and smoothness regularization
- Manifold learning

Contents

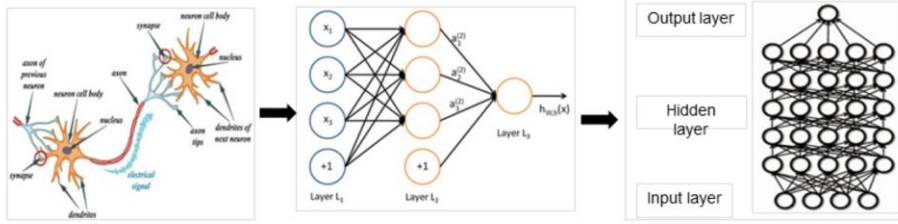
HUAWEI

Neural Network Definition

- Currently, the definition of the neural network is not determined yet. Hecht Nielsen, a neural network researcher in the US, defines a neural network as "a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs."
- Based on the origin, features, and interpretations of the neural network, it can be simply defined as **an information processing system designed to simulate human brain's structure and functions**.
- **Artificial neural network (neural network for short):** refers to a network composed of artificial neurons. It abstracts and simplifies a human brain based on its microscopic structure and functions. It is an important way to simulate human intelligence and reflects some basic features of human brain functions, such as parallel information processing, learning, association, pattern classification, and memory.

Deep learning

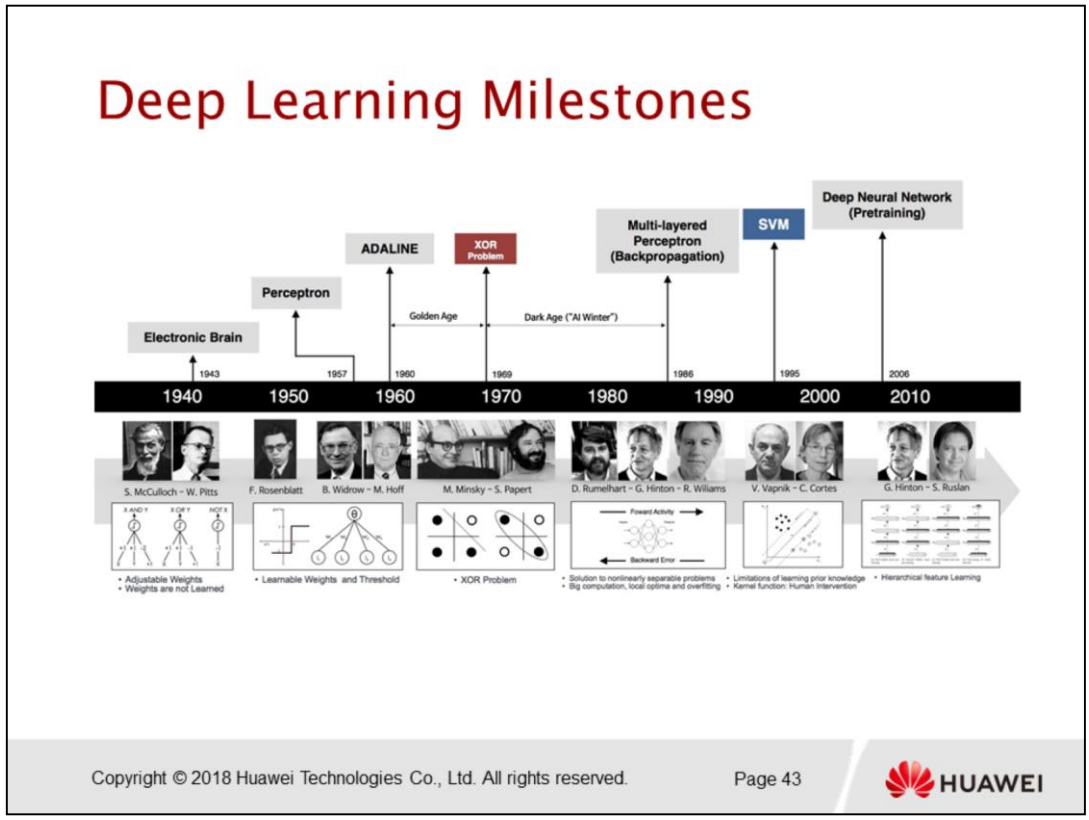
- Deep learning generally involves a deep neural network, where the depth refers to the number of layers of the neural network.



Artificial neural network | Perceptron | Deep neural network

HUAWEI

- In the design and application of the artificial neural network, three aspects need to be considered: neuron function, connection form between neurons, and network learning (training).



- Budding neural network (1958–1969)
  - In 1958, Frank Rosenblatt invented the perceptron algorithm.
  - In 1969, Marvin Minsky, pioneer of artificial intelligence in the US, questioned that the perceptron could only handle linear classification problems and failed to classify even the simplest XOR problems. The research on the perceptron was doomed to failure.
- Developing neural network (1986–1998)
  - The second-generation neural network: In 1986, G. E. Hinton, a deep learning expert, developed a BP algorithm suitable for multilayer perceptron (MLP) and used Sigmoid for non-linear mapping, which solved the problem of non-linear classification and learning.
  - Universal approximation theorem: In 1989, Robert Hecht-Nielsen proved that a continuous function f in any closed interval could be approximated by a BP network containing a hidden layer.
- Rising neural network (2006–now)
  - In 2006, deep learning came into being. Hinton proposed a solution to the vanishing gradient problem in deep network training: weight initialization for unsupervised learning and slight weight adjustment for supervised learning.
  - In 2012, at ImageNet, a top image recognition contest, Hinton's team won the

title using the convolutional neural network (CNN), boosting deep learning development.

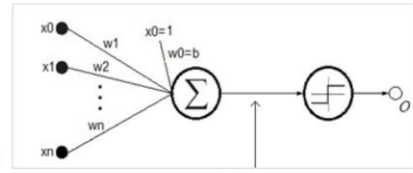
- In 2016, AlphaGo, Google's AI program based on deep learning, defeated Lee Sedol, a professional Go player of 9 dan rank, which further promoted deep learning.

Contents

HUAWEI

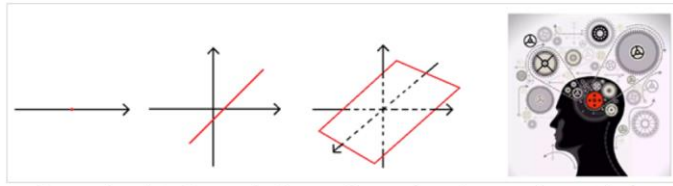
Perceptron

- **Input vector:** $X = [x_0, x_1, \dots, x_n]^T$.
- **Weight:** $W = [\omega_0, \omega_1, \dots, \omega_n]^T$, where ω_0 is the bias.
- **Activation function:** $O = sign(net) = \begin{cases} 1, net > 0, \\ -1, otherwise. \end{cases}$



$$net = \sum_{i=0}^{n} \omega_i x_i = \boldsymbol{W^T X}$$

- The perceptron is equivalent to a classifier. Its input is the high-dimensional vector X and it performs binary classification on input samples in the high-dimensional space. If $\boldsymbol{W^T X} > 0$, $o = 1$ and the sample is classified into one class. Otherwise, $o = -1$ and the sample is classified into the other class. What is the boundary? $\boldsymbol{W^T X} = 0$. This is a high-dimensional hyperplane.



Segmentation point	Segmentation line	Segmentation surface	Segmentation hyperplane
$Ax + B = 0$	$Ax + By + C = 0$	$Ax + By + Cz + D = 0$	$W^T X + b = 0$

Perceptron Training Rules

- Perceptron training rules: For each training sample $< X, t >$
  - Use the current weights to calculate the perceptron output o.
  - Each weight is updated as follows:

$$\omega_i \leftarrow \omega_i + \Delta\omega_i$$
$$\Delta\omega_i = \eta[t - o]x_i$$

  where X is the input vector, t is the target value, o is the output under the current weights, η is the learning rate, x_i and ω_i are the i-th elements of vectors X and W.
- When the training sample is linearly separable, the preceding method is repeatedly used. After finite times of training, the perceptron converges to a classifier that can correctly classify all training samples.
- When the training sample is not linearly separable, the training may be unable to converge. Therefore, another rule is designed to overcome this deficiency, namely the **delta rule**. It uses the **gradient descent** method to search for the possible weight vectors in the hypothesis space to find the weight vectors that best fit the training example.

The idea of gradient descent runs through the whole neural network theory, such as fully connected network (FCN) and CNN. Together with BP algorithm, it is the cornerstone of the neural network training theory, and we should master this. The next lessons begin with gradient descent. We start with a perceptron (linear unit) without an activation function, that is, o = $\boldsymbol{W^T X}$, and then explain why this is also valid.

Gradient Descent and Loss Function

- For the multivariable function $o = f(x) = f(x_0, x_1, \dots, x_n)$, its gradient at $X' = [x_0', x_1', \dots, x_n']^T$ is:

$$\nabla f(x_0', x_1', \dots, x_n') = [\frac{\partial f}{\partial x_0}, \frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n}]^T |_{X=X'}.$$

Direction of the gradient vector is the steepest ascent direction of the function. Therefore, the negative gradient vector $-\nabla f$ points to the steepest descent direction of the function.

- When training samples are not linearly separable, we cannot find a hyperplane to enable the perceptron to perfectly classify training samples. However, we can classify them approximately and allow some minor classification errors. To minimize the errors, we need to first parameterize the error using the **loss function (error function)**. The function reflects the error between the target output and the actual output of the perceptron. The most common error function is **L2 loss function:**

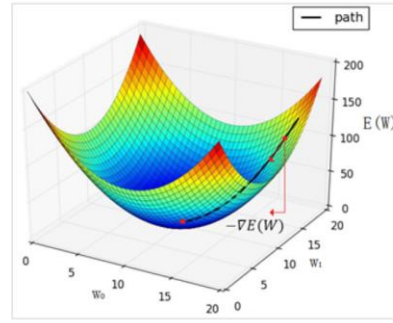
$$E(w) = \frac{1}{2}\sum_{d \in D}(t_d - o_d)^2,$$

where d is the training example, D is the training example set, t_d is the target output , and o_d is the actual output.

- This loss function has the following features:
  - Uses the weight vector as the independent variable.
  - Uses the sum of squares of deviations between the target output t_d and the actual output o_d of training examples as the subject.
  - There is a coefficient ½.
- We see that once a training sample is given, its input and target output values are constants and the actual output changes with W, so the independent variable of the error function is W.
- The coefficient ½ is difficult to understand. When deriving E with respect to the independent variable, the coefficient ½ offsets the coefficient to 1, which will be seen later.

Extrema of the Loss Function

- The independent variable of the loss function $E(W)$ is a weight so the function is defined in a weight space. Then we need to search for a weight vector W that minimizes $E(W)$ in the weight space. Unfortunately, $E(W) = \frac{1}{2}\sum_{d \in D}(t_d - o_d)^2$ defines a very complex high-dimensional curved surface and **there is no effective mathematical method to calculate the extrema of a high-dimensional curved surface**. Considering that the negative gradient direction is the steepest descent direction, we can start from a point and go along the $-\nabla E(W)$ direction to find the minimum point of $E(W)$. This is the core idea of the gradient descent method.



Gradient descent on the binary paraboloid

Global Gradient Descent Algorithm for Linear Units

- Each sample in the training sample set D is denoted as $< X, t >$. X is the input vector, t is the target output, and η is the learning rate.
  - Initialize each w_i to a random value with a small absolute value.
  - Before the termination condition is met, do:
    - Initialize each Δw_i to 0.
    - For each $< X, t >$ in D, do:
      - Enter X in the unit and calculate the output o.
      - For each w_i in the unit, do Δw_i+= η(t-o) x_i
    - For each w_i in the unit, do w_i+= Δw_i
- This gradient descent algorithm is not commonly used because it has the following defects:
  - The convergence process is very slow because all training samples need to be calculated every time the weight is updated.
  - If the error surface has multiple local minima, the process is easily trapped in local extrema.

SGD Algorithm and Online Learning

- To address the defects of the original gradient descent algorithm, a common variant, incremental gradient descent algorithm, is used, which is also called stochastic gradient descent (SGD) algorithm. One implementation is called online learning, which updates the gradient based on each sample:

$$\Delta w_i = \eta \sum_{d \in D} (t_d - o_d) x_{id} \Longrightarrow \Delta w_i = \eta (t_d - o_d) x_{id}.$$

- Online-Gradient-Descent (D, η)
  - Initialize each w_i to a random value with a small absolute value.
  - Before the termination condition is met, do:
    - For each $< X, t >$ in D, do:
      - Enter X in the unit and calculate the output o.
      - For each w_i in the unit, do Δw_i+= η(t-o) x_i.

- This gradient descent algorithm goes to another extreme, that is, updating the weight for each sample. Because the training sample usually contains noise, the gradient is difficult to converge to the extrema when approaching the extrema.

Mini-Batch Gradient Descent

- To address the defects of the previous two gradient descent algorithms, another algorithm, mini-batch gradient descent, has been proposed. It is the most widely used one. The idea is to use samples of a fixed batch size (BS) to calculate Δw_i and update the weight.
- Batch-Gradient-Descent (D, η, BS)
  - Initialize each w_i to a random value with a small absolute value.
  - Before the termination condition is met, do:
    - Initialize each Δw_i to 0.
    - Select samples (BS) from D. For each $< X, t >$ among these samples, do:
      - Enter X in the unit and calculate the output o.
      - For each w_i in the unit, do Δw_i+= η(t-o) x_i.
    - For each w_i in the unit, do w_i += Δw_i
    - If it is the last batch, disrupt the training sample sequence.

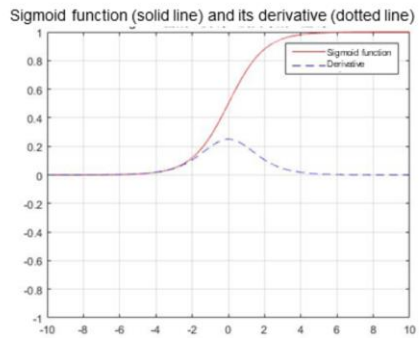
- This gradient descent algorithm considers both the efficiency and gradient stability. It is easy to overshoot the local minimum and is the most commonly used gradient descent algorithm in actual work. The value of BS varies with specific problems. Generally, the value of is 128.

Contents

Activation Function: Sigmoid

- Sigmoid function:

$$f(x) = \frac{1}{1 + e^{-x}}$$



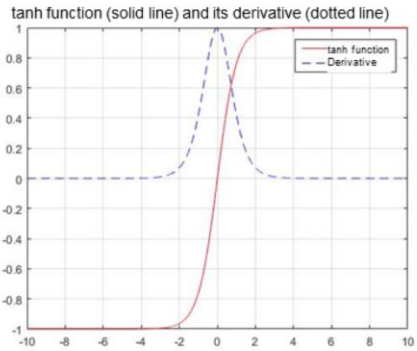
HUAWEI

- The Sigmoid function is monotonic and continuous and easy to derive. The output is bounded, and the network is easy to converge. However, we see that its derivative is close to 0 at the point away from the central point. When the network is very deep, more and more backpropagation gradients fall into the saturation area, so that the gradient model becomes smaller. Generally, if the Sigmoid network has five or fewer layers, the gradient is degraded to 0, which is difficult to train. This phenomenon is called gradient vanishing. In addition, the output of the Sigmoid is not zero-centered.

Activation Function: tanh

- tanh function:

$$\tanh x = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

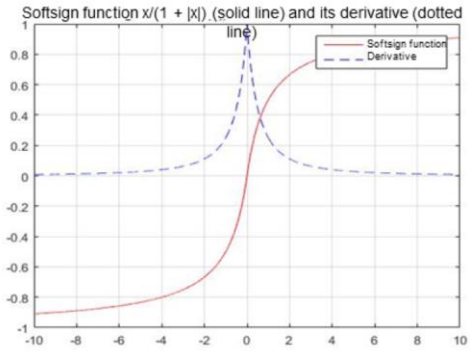


- The tanh function also has the similar defect, that is, the derivative of the point far away from the center approaches 0. Because the tanh function is zero-symmetric, its average output is closer to 0 than that of the Sigmoid function. Its SGD is more closer to the natural gradient, reducing the iteration times.

Activation Function: Softsign

- Softsign function:

$$f(x) = \frac{x}{|x| + 1}$$

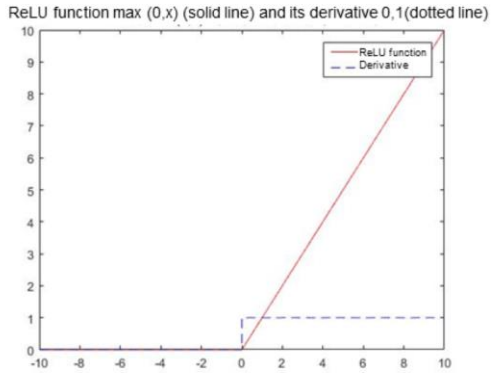


- This function saturates more slowly than the tanh function.

Activation Function: ReLU

- Rectified Linear Unit (ReLU) function:

$$y = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$$



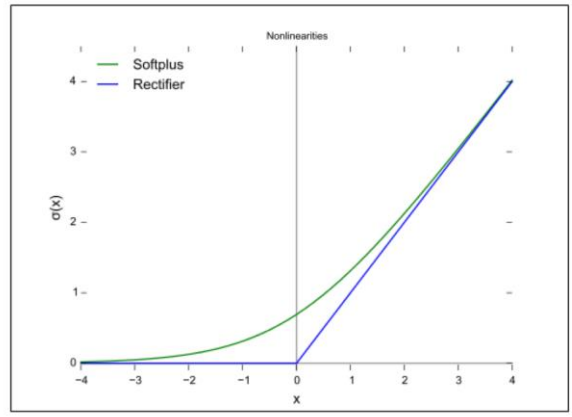
HUAWEI

- Advantages:
  - Compared with Sigmoid and tanh, ReLU supports faster convergence in SGD.
  - Sigmoid and tanh involve exponential calculation while ReLU has simpler calculation.
  - Effectively alleviates the vanishing gradient problem.
  - Delivers good performance without unsupervised pre-training.
- Disadvantages:
  - There is no upper bound, and training is relatively easy to diverge.
  - Not differentiable at 0 and forcibly defines a derivative.
  - The curved surface of the turning point is also "angular", which is not smooth enough in some regression problems.

Activation Function: Softplus

- Softplus function:

$$f(x) = \ln(e^x + 1)$$



HUAWEI

- Compared with ReLU, this function has more complex computation. However, it has a continuous derivative and defines a smooth curved surface.

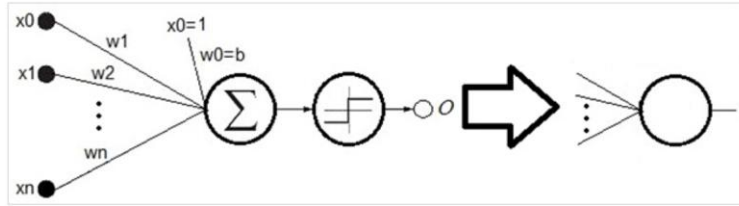
Factors to Be Considered in Activation Function Design

- **Non-linear:** When the activation function is non-linear, a two-layer neural network can be proved to be a common function approximation. If the function is linear, the entire network is equivalent to a single-layer linear model.
- **Continuously differentiable:** This attribute is desirable for enabling gradient-based optimization methods. If some functions that are locally non-differentiable are selected, the derivative must be defined forcibly.
- **Range:** When the range of the activation function is finite, gradient-based training methods tend to be more stable. When the range is infinite, training is generally more efficient but it is easy to diverge. Smaller learning rates are typically necessary.
- **Monotonicity:** When the activation function is monotonic, the loss function associated with a single-layer model is convex.
- **Smooth:** A smooth function with a monotonic derivative has been proved to generalize better in some cases.
- **Approximating identity near the origin:** When activation functions have this property, the neural network will learn efficiently when its weights are initialized with small random values. When the activation function does not approximate identity near the origin, special care must be used when initializing the weights.

Contents

Multi-layer Fully Connected Artificial Neural Network

- A single perceptron has limited representation capability and can only represent the linear decision surface (hyperplane). If we connect many perceptrons like a human brain does and then replace the activation function with a non-linear function, we can express a wide range of non-linear surfaces.

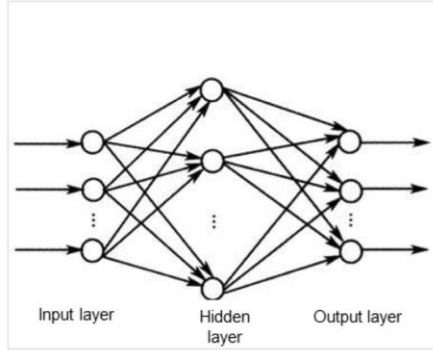


HUAWEI

- Impact of the sign activation function:
  - When discussing training rules, we said that the impact of the softsign activation function could be temporarily ignored. Why could we do this?
  - For the target output, the value is 1 or -1. If the linear unit without an activation function can perfectly fit the target output, the target output can still be perfectly fitted after the softsign function is added because sign (1) is 1 and sign (-1) is -1. If the linear unit fails to perfectly fit the target output, the target output can be then perfectly fitted after the sign mapping as long as the linear unit outputs correct symbols.
  - So far, we have introduced the training rules of a single perceptron. Next, we need to interconnect multiple perceptrons to form a real artificial neural network.
- Figure: To facilitate the drawing of a multi-layer neural network, we simplify the perceptron diagram. The SUM and activation functions are expressed in circles, and the input, weight, and output symbols are omitted.

Feedforward Neural Network

- **The feedforward neural network** is one of the simplest neural networks in which neurons are arranged hierarchically. It is the most widely neural network with the fastest development.
- The input node does not support computation, but is merely used to represent each element value of the input vector.
- Each node represents a neuron that supports computation, which is called a computational unit. Each neuron is connected only to the neurons at the previous layer.
- A hidden layer receives the output from the previous layer and sends the results to the next layer. A unidirectional multi-layer structure is adopted. Each layer contains several neurons. The neurons at the same layer are not connected to each other. Inter-layer information is transmitted only in one direction.



- Feedforward: The weight is not returned to the input unit or the output unit at the previous layer.
- Fully connected. Each unit provides the input for each unit at the next layer.

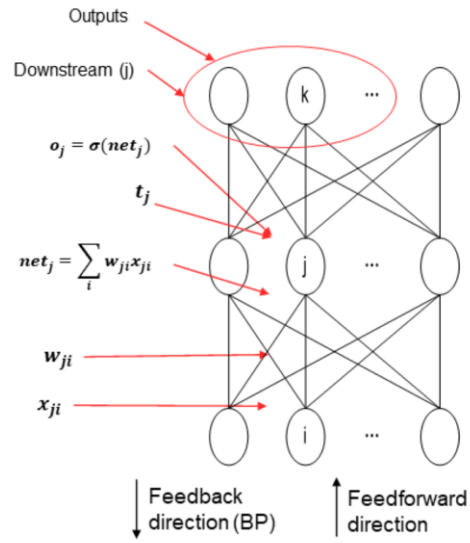
Backpropagation Algorithm

- We can train the neural network with this formula:

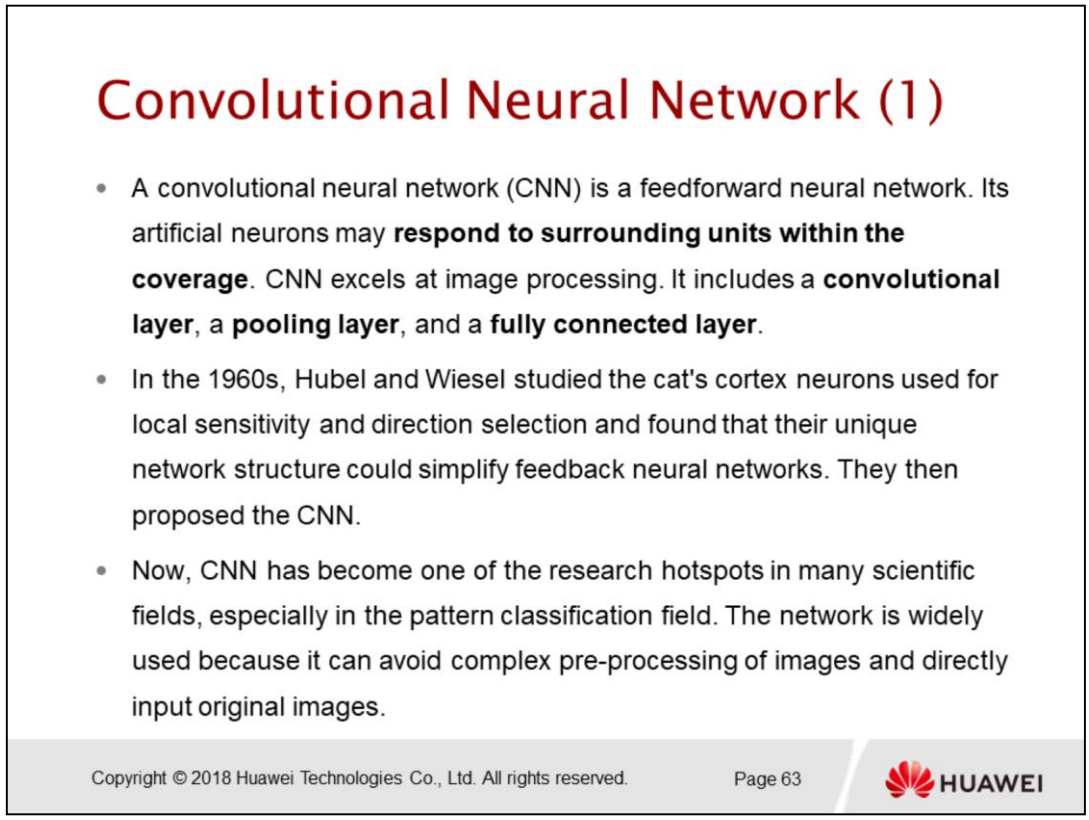
$$\Delta w_{ji} = \eta \delta_j x_{ji}$$

$$\delta_j = \begin{cases} o_j(1-o_j)(t_j - o_j), & j \in outputs \quad (1) \\ o_j(1-o_j)\sum_{k \in DS(j)} \delta_k w_{kj}, & otherwise \quad (2) \end{cases}$$

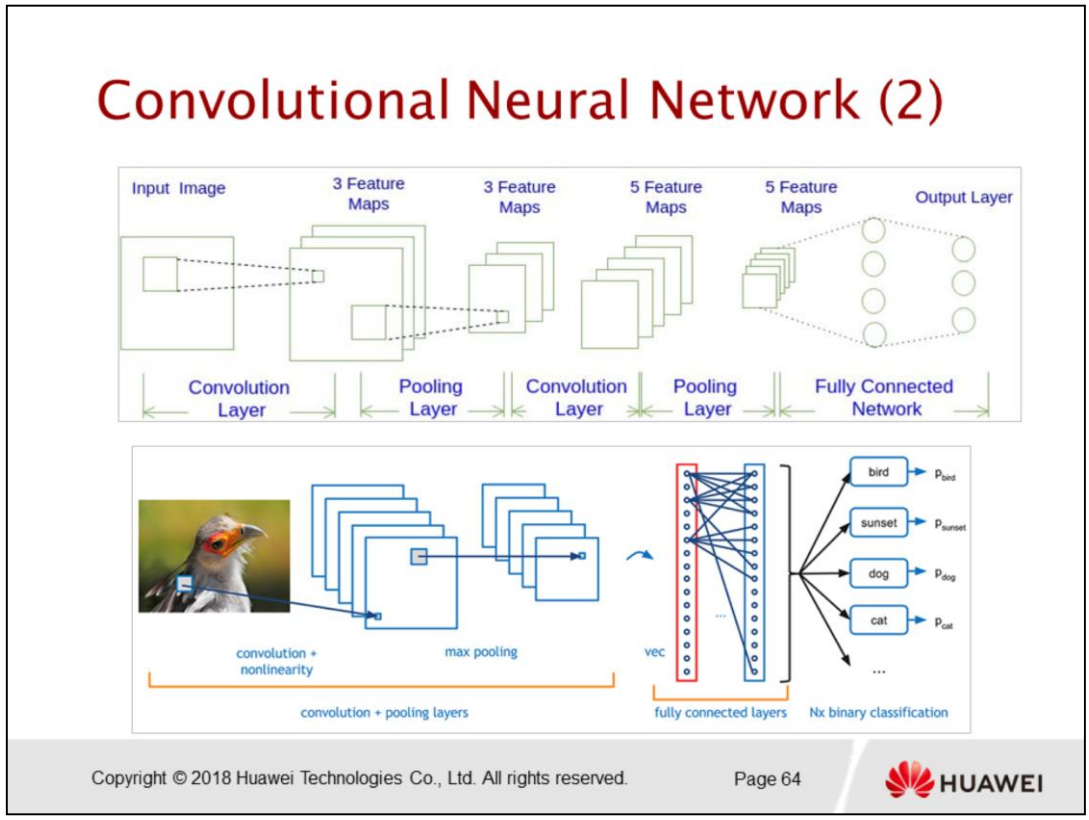
- The procedure for using the BP algorithm to train the network is as follows:
  1. Take out the next training sample $< X, T >$, and enter X to the network to obtain the actual output O.
  2. Calculate δ of the output layer based on the output layer error formula (1) and update the weight.
  3. For the hidden layer, according to the hidden layer error propagation formula (2), the δ of each layer is calculated from the output direction to the input direction, layer by layer with iteration. Once the δ of a layer is calculated, update the weight of the layer until all weights are updated.
  4. Return to 1 to continue.



- In step 3, the error is calculated from the output direction to the input direction, layer by layer with iteration. Therefore, the signal is propagated forward, and the error is propagated back. This is the origin of the name of backpropagation algorithm.



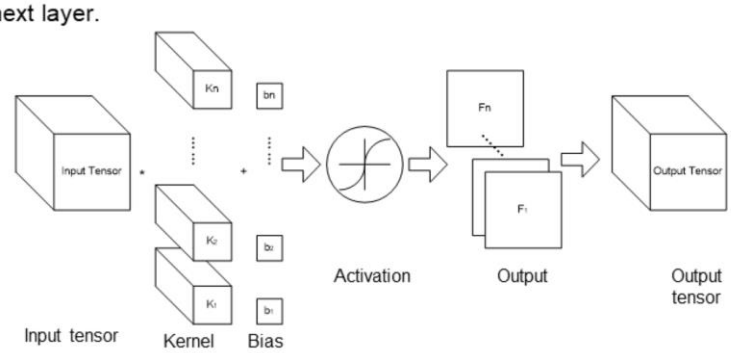
- A filter matrix is a set of fixed weights and can be seen as a constant filer (kernel). The convolution (adding each element, weighted by the kernel) is performed between an image (data from different data windows) and a kernel. This type of network is called CNN.
- Local receptive field: Generally, people get to know the outside world from local to global. For image space association, a pixel is closely related to local pixels and weakly related to remote pixels. Therefore, each neuron does not need to know the global image. It only needs to know the local image and then the local information is combined at a higher level to generate global information. The idea of local network connection is also inspired by the biological visual system structure. The neurons in the visual cortex receive local information (respond to stimuli of certain regions).
- Parameter sharing: One or more filters can be used to scan the input images. The parameters of the filter are weights. At the layers scanned by the same filter, each filter uses the same parameter to perform weighted calculation. Weight sharing means that the parameter value of each filter does not change when the filter scans the entire image. For example, we have three feature filters and each filter scans the entire image. During the scanning process, the parameter values of the filters do not change. In other words, all elements of the image share the same weights.



- Input layer: data input
- Convolutional layer: composed of several convolutional units. The parameters of each convolutional unit are optimized by the backpropagation algorithm. The purpose of the convolution operation is to extract different features of the input. The first convolutional layer may extract only basic features such as edges, lines, and angles. A multi-layer network can extract more complex features from the basic features.
- ReLU layer: uses ReLU f (x) = max (0, x) as the activation function.
- Pooling layer: partitions features obtained from the convolutional layer into some areas and outputs the maximum or minimum value, generating new features with a smaller spatial size.
- Fully connected layer: integrates all local features into global features to calculate the final score for each type.
- Output layer: outputs the final result.

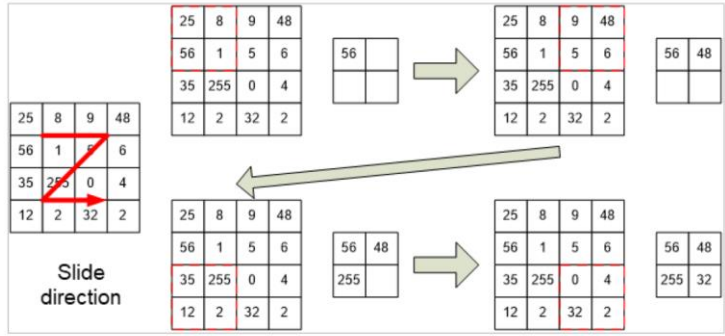
Convolutional Layer

- The basic structure of the convolutional neural network is the convolutional layer, namely multi-channel convolution that has been mentioned before. The output of the previous layer (or the original image of the first layer) is used as the input of the current layer. It is then convolved with the convolution kernel of the layer and serves as the output of this layer. The convolution kernel of each layer is the weight to be learnt. Similar to FCN, after the convolution is complete, the result should be biased and activated through activation functions before being input to the next layer.



Pooling Layer

- Pooling combines the nearby units to reduce the size of the input for the next layer. It includes max pooling and average pooling. The former selects the maximum value in a small square area as the representative of this area while the latter uses the mean. The side of this small area is the pool window size. The following figure shows the max pooling operation whose pooling window size is 2.

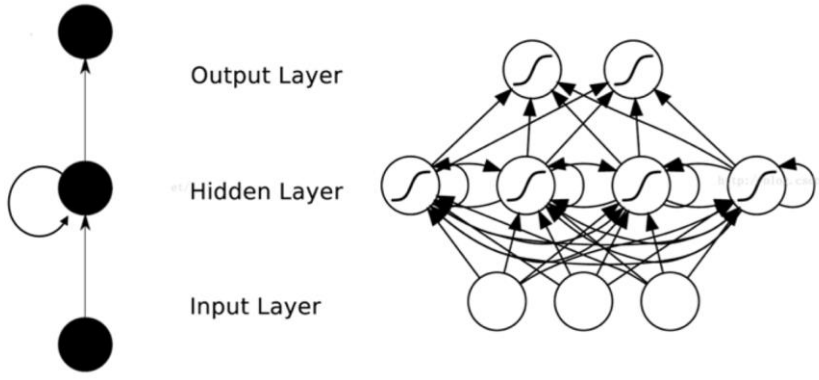


- The actual classification networks are feedforward networks that are formed by interconnected convolutional and pooling layers. The pooling layer has the following functions:
  - Invariance: Max pooling ensures invariance within a certain range, because the maximum value of a region is the final output value regardless of where the value is.
  - Reducing the input size for the next layer: Pooling effectively reduces the size of the input data for the next layer, the number of parameters, and computation workload.
  - Obtaining fixed-length data: By properly setting the pooling window size and stride, we can obtain fixed-length outputs from variable-length inputs.
  - Increasing the scale: The features of the previous layer can be extracted from a larger scale.
  - Preventing overfitting: Pooling simplifies the network and reduces the fitting precision. Therefore, it can prevent overfitting (pay attention to the possible underfitting).

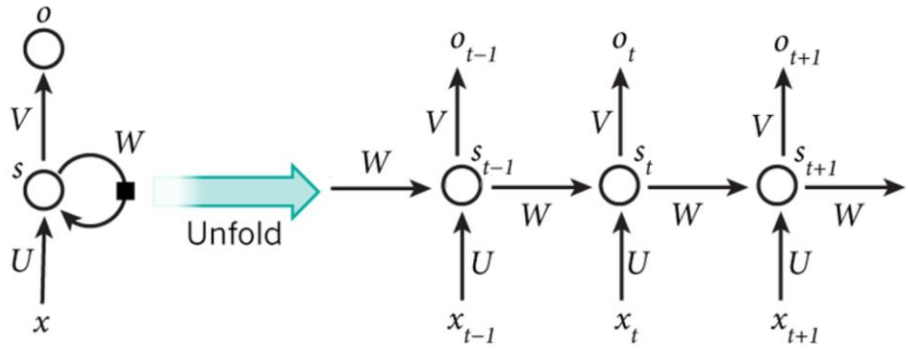
Recurrent Neural Network (1)

- The recurrent neural network (RNN) is a neural network that captures dynamic information in sequential data through periodical connections of hidden layer nodes. It can classify sequential data.
- Unlike other forward neural networks, the RNN can keep a context state and even store, learn, and express related information in context windows of any length. Different from traditional neural networks, it extends in space and time sequences. In other words, the hidden layers of the current moment and the next moment are related.
- The RNN is widely used in scenarios related to sequences, such as videos consisting of image frames, audio consisting of clips, and sentences consisting of words.

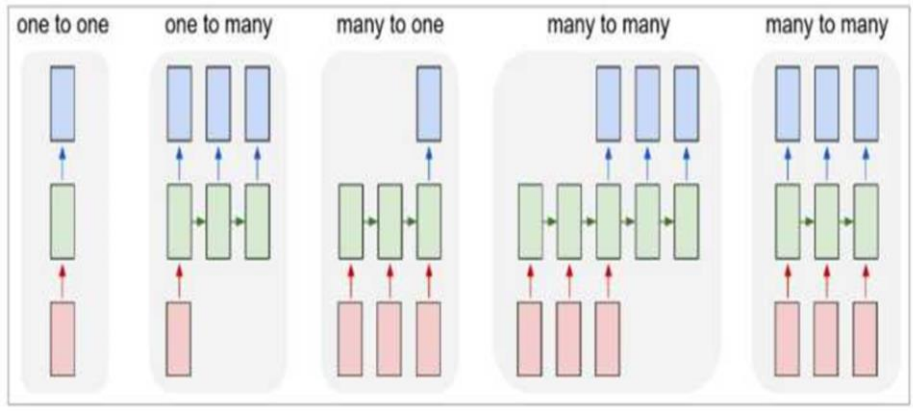
Recurrent Neural Network (2)

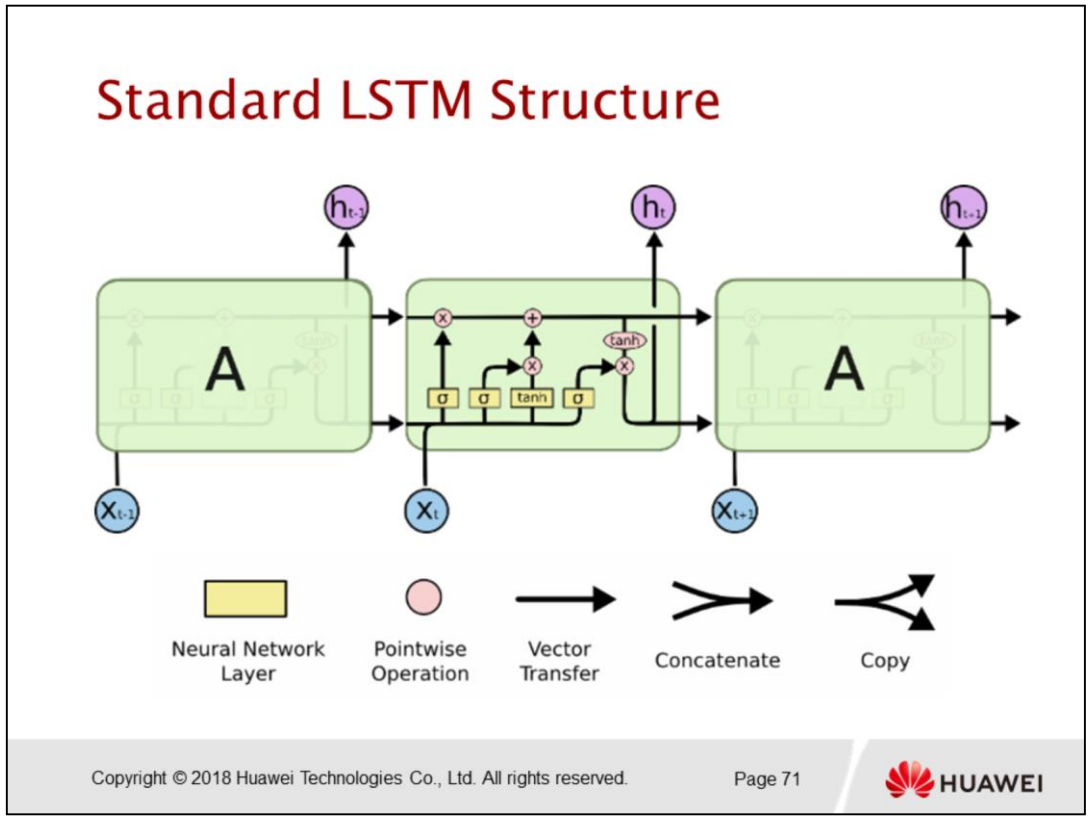


Unfolded Recurrent Neural Network



Types of Recurrent Neural Networks





- Long short-term memory (LSTM) applies to the scenario with a large gap between the relevant information and the point where it is needed. It can connect previous information for long periods of time to the present task.

Contents

Regularization in Deep Learning

- Regularization is a very important and effective technology to reduce generalization errors in machine learning. It is especially useful for deep learning models which tend to have overfitting due to diverse parameters. Researchers have also proposed many effective technologies to prevent overfitting, including:
  - Adding constraints to parameters, such as L_1 and L_2 norms
  - Expanding the training set, such as adding noise and transforming data
  - Dropout

Parameter Penalties

- Many regularization approaches add a parameter penalty $\Omega(\theta)$ to the objective function J, limiting the learning capability of models. We denote the regularized objective function as $\tilde{J}$:

$$\tilde{J}(\theta; X, y) = J(\theta; X, y) + \alpha\Omega(\theta),$$

where $\alpha \epsilon [0, \infty)$ is a hyperparameter that weights the relative contribution of the norm penalty term, Ω, relative to the standard objective function $J(X; \theta)$. Setting α to 0 results in no regularization. Larger values of α correspond to more regularization.

- Generally, in deep learning, we only add constraints to affine parameter w instead of biases. The main reason is that biases usually require less data to fit accurately. Adding constraints often leads to underfitting.

L_2 Regularization

- Add L_2 parameter norm penalty to prevent overfitting.

$$\tilde{J}(w; X, y) = J(w; X, y) + \frac{1}{2}\alpha\|w\|^2,$$

The parameter optimization method can be inferred using the optimization technology (such as gradient correlation method):

$$w = (1 - \varepsilon\alpha)\omega - \varepsilon\nabla J(w),$$

where ε is the learning rate. This multiplies parameters by a reduction factor compared with the common gradient optimization function.

- If J is a secondary optimization problem, the model parameter may be further represented as $\widetilde{w}_i = \frac{\lambda_i}{\lambda_i + \alpha} w_i$, that is, adding a control factor to the original parameter, where λ is an eigenvalue of the parameter Hessian matrix. Therefore:
  - When $\lambda_i \gg \propto$, the penalty factor has a small effect.
  - When $\lambda_i \ll \propto$, the corresponding parameter is reduced to 0.

L_1 Regularization

- Add L_1 norm constraint to model parameters:

$$\tilde{J}(w; X, y) = J(w; X, y) + \alpha \|w\|_1,$$

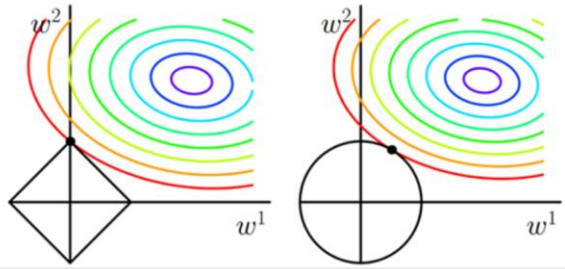
- If the gradient method is used to solve the problem, the parameter gradient is

$$\nabla \tilde{J}(w) = \propto sign(w) + \nabla J(w).$$

- In special cases, for the secondary optimization problem and assuming that the corresponding Hessian matrix is a diagonal matrix, the parameter recursive formula is $w_i = sign(w_i^*)\max(|w_i^*| - \frac{\alpha}{\lambda_i}, 0)$. When $|w_i^*| < \frac{\alpha}{\lambda_i}$, the value of this parameter is reduced to 0, which is different from L_2 regularization. L_2 regulation does not directly reduce the parameter to 0, but makes a value close to 0.

L_2 VS L_1

- Major differences between L_2 and L_1:
  - According to the preceding analysis, L_1 can generate a more sparse model than L_2. When the parameter w is small, L_1 regularization can directly reduce the parameter to 0, which can be used for feature selection.
  - From the perspective of probability, many norm constraints are equivalent to adding prior distribution to parameters. L_2 norm equals to Gaussian distribution while L_1 norm equals to Laplace distribution.

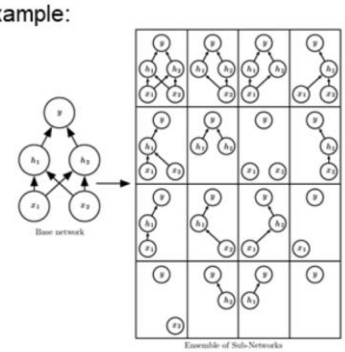


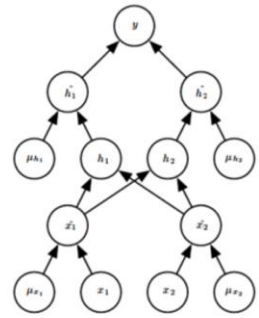
Dataset Expansion

- The most effective way to prevent overfitting is to add a training set. A larger training set has a smaller overfitting probability. Dataset expansion is a time-saving method, but it varies in different fields.
  - A common method in the object recognition field is to rotate or scale images. (The prerequisite to image transformation is that the type of the image cannot be changed through transformation. For example, for handwriting digit recognition, category 6 and 9 can be easily changed after rotation).
  - Random noise is added to the input data in speech recognition.
  - The common idea of NLP is to replace words with their synonyms.
  - Noise injection can add noise to the input or to the hidden layer or output layer. For example, for the softmax classification problem, noise can be added by using the label smoothing technology. If noise is added to the 0-1 category, the corresponding probability is changed to $\frac{\varepsilon}{k}$ and $1-\frac{k-1}{k}\varepsilon$.

Dropout

- Dropout is a common and simple regularization method, which has been widely used since 2014. Simply put, dropout randomly discards some inputs during the training process. In this case, the parameters corresponding to the discarded inputs are not updated. Dropout is an integration method. It combines all sub-network results and obtains sub-networks by randomly dropping inputs. For example:





- The sampling probability of each entry is 0.8 for the input and 0.5 for the hidden layers.
- Advantages:
  - Compared with weight decay and norm constraints, this strategy is more effective.
  - It is computationally cheap and simple and can be used in other non-deep-learning models.
  - However, it is less effective when the training data is insufficient.
  - Stochasticity is not necessary or sufficient to achieve the regularizing effect of dropout. The invariant shielding parameters can be constructed to obtain good solutions.
- In addition to the preceding methods, we can also use semi-supervised learning, multi-task learning, early stopping, parameter sharing, ensemble methods, and adversarial training.

Contents

Optimizer

- There are various improved versions of gradient descent algorithms. In object-oriented language implementation, different gradient descent algorithms are often encapsulated into an object which is called an **optimizer**.
- The **purpose** of algorithm improvement includes but is not limited to:
  - Accelerates algorithm convergence.
  - Avoids or overshoots local extrema.
  - Simplifies manual parameter setting, especially the learning rate.
- Common optimizers: common GD optimizer, momentum optimizer, Nesterov, **Adagrad**, **Adadelta**, **RMSprop**, Adam, AdaMax, and Nadam

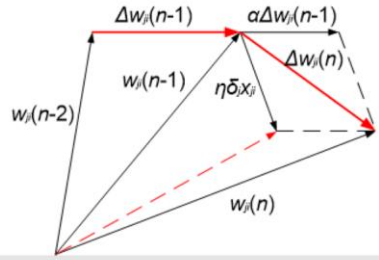
Momentum Optimizer

- One basic improvement is to add the momentum term for Δw_{ji}. Denote the weight correction of n-th iteration as $\Delta w_{ji}(n)$ and then the weight correction rule is:

$$\Delta w_{ji}(n) = \eta \delta_j x_{ji} + \alpha \Delta w_{ji}(n-1)$$

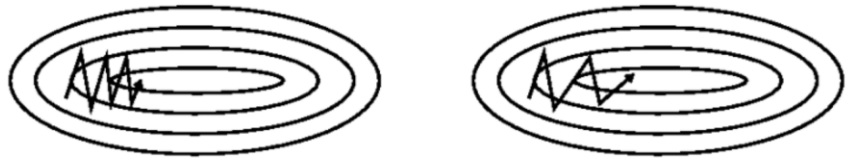
  where α is a constant ($0 \leq \alpha < 1$) and called momentum and $\alpha \Delta w_{ji}(n-1)$ is a momentum term.

- Imagine a small ball rolls down the error surface starting from a random point. The introduction of the momentum term is equivalent to giving the small ball inertia.



Advantages and Disadvantages of Momentum Optimizer

- Advantages:
  - Enhances the stability of the gradient correction direction and reduces mutations.
  - In areas where the gradient direction is stable, the ball rolls faster and faster (there is a speed upper limit because $\alpha < 1$), which helps the ball quickly overshoot the flat area and accelerates convergence.
  - A small ball with inertia is more likely to roll over some narrow local extrema.
- Disadvantage:
  - The learning rate η and momentum α need to be manually set, which often requires more experiments to determine the appropriate value.



Adam Optimizer (1)

- Adaptive Moment Estimation (Adam): Developed based on Adagrad and Adadelta, Adam maintains two additional variables m_t and v_t for each variable to be trained:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

where t represents t-th iteration and g_t is the calculated gradient. m_t and v_t are moving averages of the gradient and the squared gradient. From the statistical perspective, m_t and v_t are estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients respectively, hence the name of the method.

- Adam attempts to compute adaptive learning rates for each parameter. This is very useful in complex network structures because different parts of the network have different sensitivity to weight adjustment. A very sensitive part usually requires a smaller learning rate. It is difficult or complex to manually identify the sensitive part and set a learning rate. It may be the best optimizer at present.

Adam Optimizer (2)

- As m_t and v_t are initialized as vectors of 0's, they are biased towards 0, especially during the initial steps, and especially when β_1 and β_2 are close to 1. To solve this problem, we use $\hat{m}_t$ and $\hat{v}_t$:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

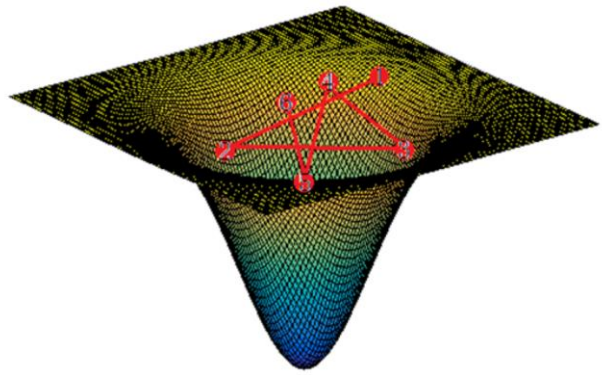
- Adam's weight update rule:

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t$$

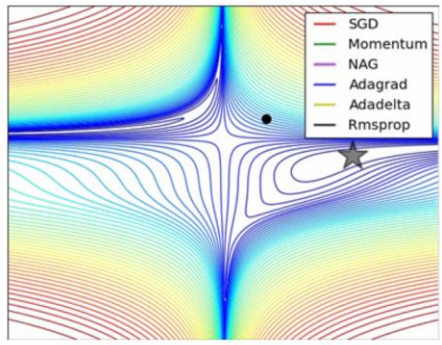
- Although the rule involves manual setting of η, β_1, and β_2, this is much simpler. According to experiments, the default setting are $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and η = 0.001. In practice, Adam will converge quickly. For convergence saturation, reduce η. After several times of reduction, the satisfying local extrema will be converged. Other parameters do not need adjustment.

Adam Optimizer (3)

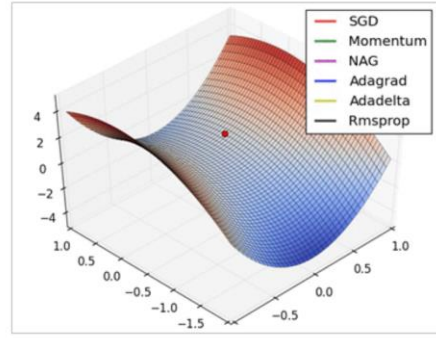
- $\widehat{m}_t$ is the current gradient with a momentum term and $\frac{\eta}{\sqrt{\hat{v}_t+\epsilon}}$ is equivalent to the current learning rate. If the gradient model is too large, the weight cannot jump out of the extrema or converge. Then we need to reduce the learning rate to facilitate convergence.



Visualized Comparison of Optimizers



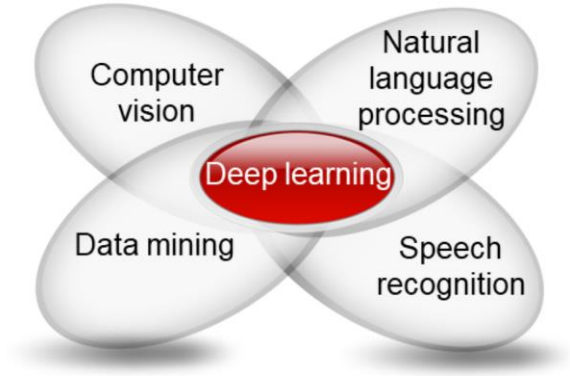
Comparison of optimization algorithms in contour maps of loss functions



Comparison of optimization algorithms at the saddle point

Contents

Applications of Deep Learning



Computer vision

- Image classification
- Scene recognition
- Face detection and recognition
- Object detection and recognition
- Semantic segmentation
- Stylization
- Super-resolution
- OCR

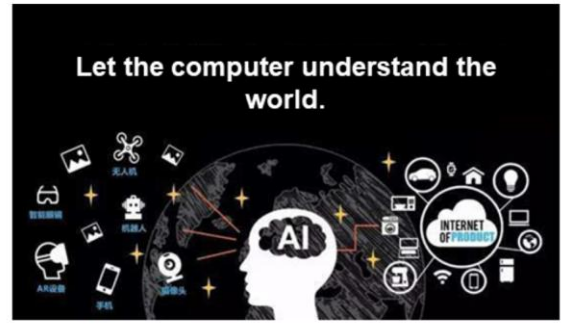


Image Classification

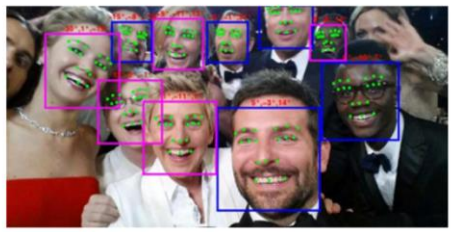
- Maps images to different class sets, which can be used for image retrieval and image archiving.

Scene Recognition

- Classifies images based on scenes and environments, which can be used for situation awareness and intelligent 3A for photographing.

Face Detection and Recognition

- Face detection: discovers and locates faces and facial features in images, which can be used for face focusing, polishing up, and augmented reality.
- Facial recognition: differentiates people, which can be used for identity authentication and privacy protection.

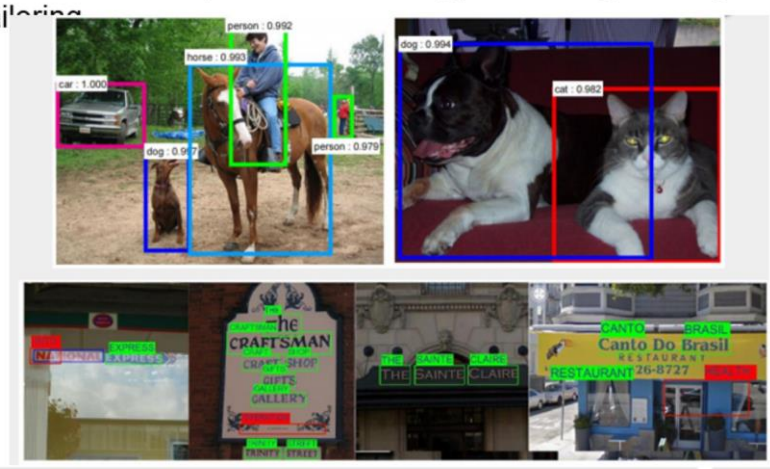


Face detection

Facial recognition

Object Detection and Recognition

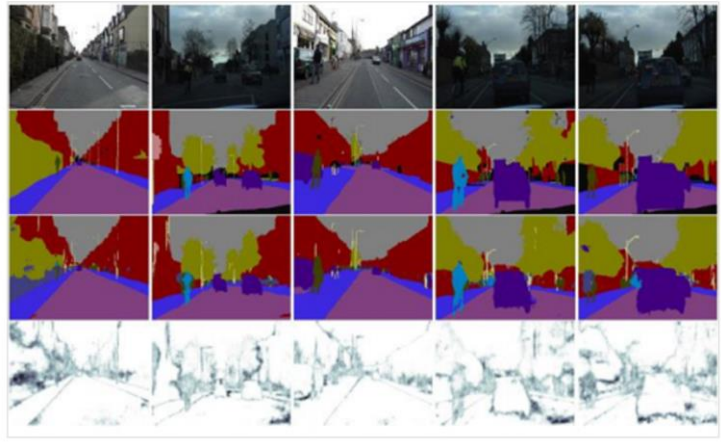
- Detects, locates, and identifies different objects in images, including digital, text, and pedestrian detection. This function can be used for OCR, unmanned driving, and intelligent image tailoring.



HUAWEI

Semantic Segmentation

- Predicts the label of each pixel in the image, including segmentation and recognition. It can be used for unmanned driving, augmented reality, and situation awareness.

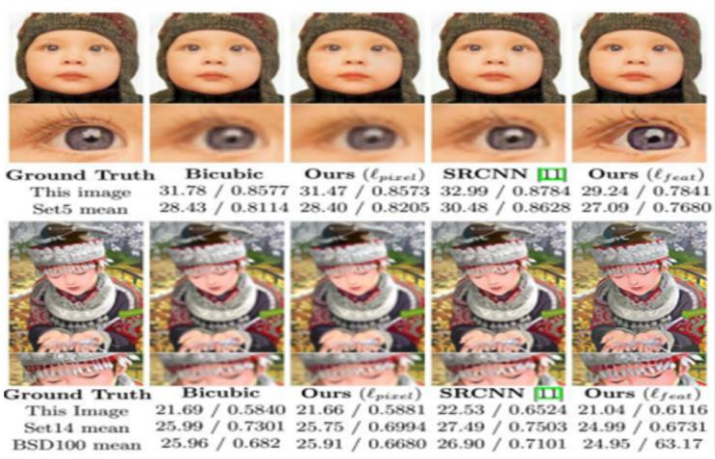


Stylization

- Changes the image style while retaining the content of the image, which can be used for stylized image processing.

Super-resolution

- Generates high-resolution images from low-resolution images, which can be used for image processing, security surveillance, and medical imaging.



Ground Truth	Bicubic	Ours (ℓ_{pixel})	SRCNN [11]	Ours (ℓ_{feat})
This image	31.78 / 0.8577	31.47 / 0.8573	32.99 / 0.8784	29.24 / 0.7841
Set5 mean	28.43 / 0.8114	28.40 / 0.8205	30.48 / 0.8628	27.09 / 0.7680

Ground Truth	Bicubic	Ours (ℓ_{pixel})	SRCNN [11]	Ours (ℓ_{feat})
This Image	21.69 / 0.5840	21.66 / 0.5881	22.53 / 0.6524	21.04 / 0.6116
Set14 mean	25.99 / 0.7301	25.75 / 0.6994	27.49 / 0.7503	24.99 / 0.6731
BSD100 mean	25.96 / 0.682	25.91 / 0.6680	26.90 / 0.7101	24.95 / 63.17

OCR

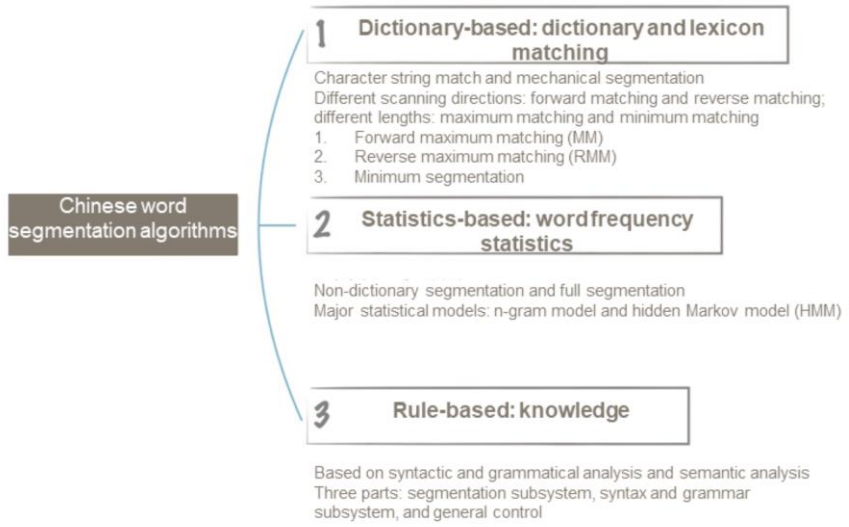
- Identifies information such as numbers and texts in images to digitize images or paper documents, which can be used to automatically identify business cards or scan invoices.

Natural Language Processing

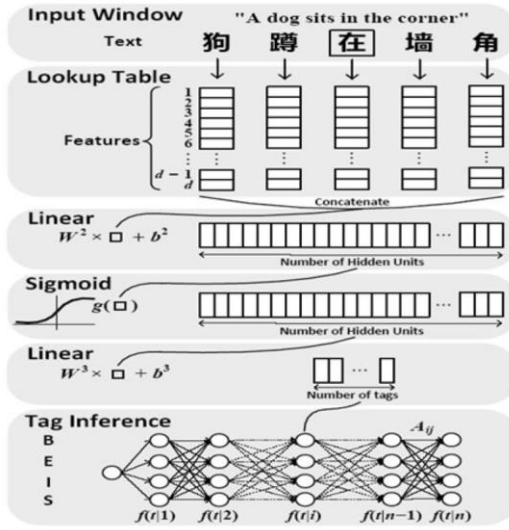
- Chinese word segmentation
- Knowledge exploration
- Machine translation
- Sentiment analysis

HUAWEI

Chinese Word Segmentation Algorithms



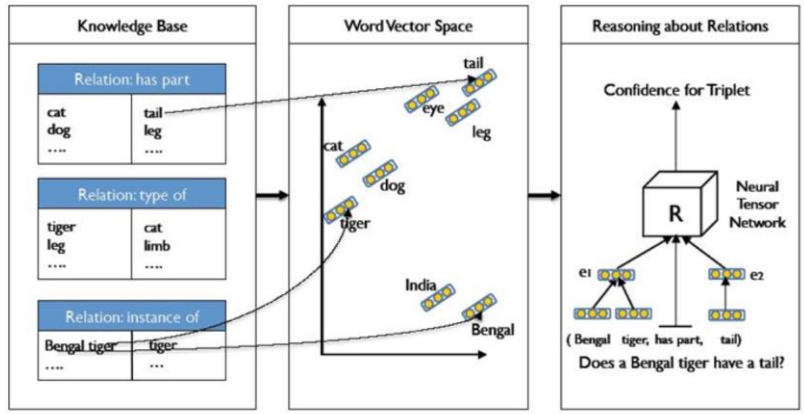
Chinese Word Segmentation



Knowledge Exploration

- Two types of problems
  - New knowledge reasoning in the existing knowledge base
    - CYC, WordNet, FreeNet, etc.
    - Current studies use similar approaches.
      - Known entities are represented by word embeddings.
      - The entity relationship is modeled using tensor networks.
      - Backpropagation + SGD training
  - Mining structured knowledge from free text
    - Text mining
    - TF-IDF

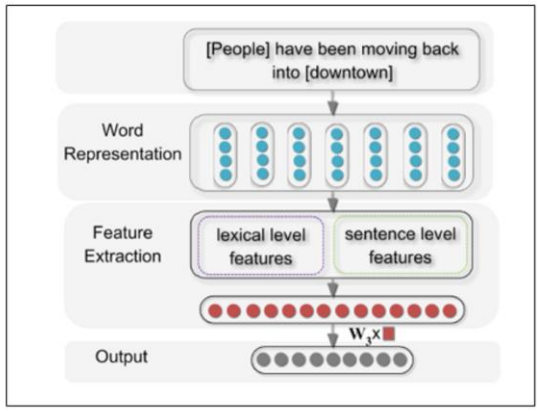
New Knowledge Reasoning in the Existing Knowledge Base



HUAWEI

- **Knowledge reasoning** means that a computer or an intelligent system simulates human's intelligent reasoning and uses formal knowledge for thinking and solving problems based on reasoning control strategies.
- The reasoning method mainly solves the logical relationship between the premise and the conclusion in the reasoning process and the transfer of uncertainty in the fuzzy reasoning.
- According to different standards, the reasoning methods are classified into the following three types:
  - Deductive reasoning and inductive reasoning by form
  - Precise reasoning and fuzzy reasoning by certainty
  - Monotonic reasoning and non-monotonic reasoning by monotonicity

Mining Structured Knowledge from Free Text (1)

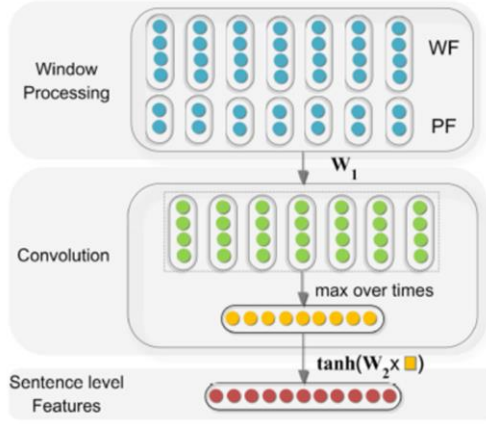


Overall structure

Features	Remark
L1	Noun 1
L2	Noun 2
L3	Left and right tokens of noun 1
L4	Left and right tokens of noun 2
L5	WordNet hypernyms of nouns

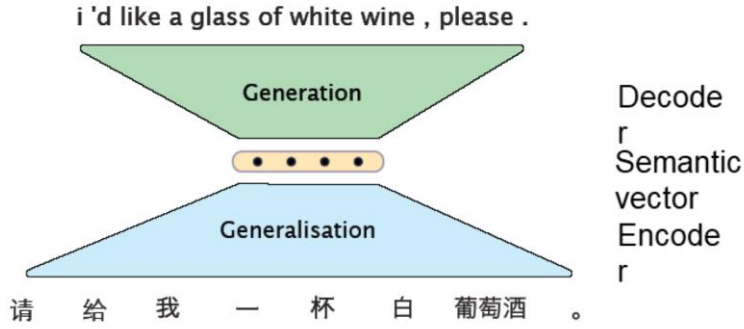
Lexical feature

Mining Structured Knowledge from Free Text (2)



Sentence-level feature extraction: convolutional network

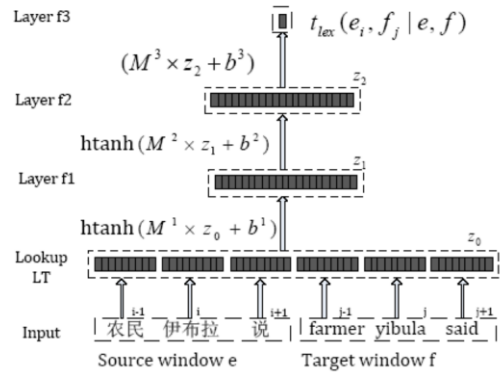
Machine Translation (Common Model)



The most common model: encoder-decoder model

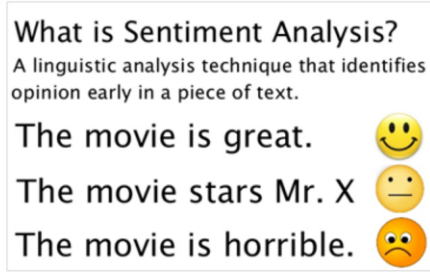
Machine Translation: Deep Learning Applied to Multiple Areas

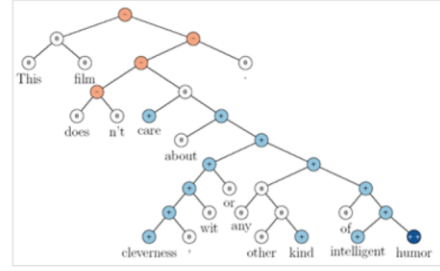
- Word alignment
- Phrase alignment
- Phrase reordering
- Language model
- Translation model
- Conjunctive model
- Translation result reordering
- …



Sentiment Analysis

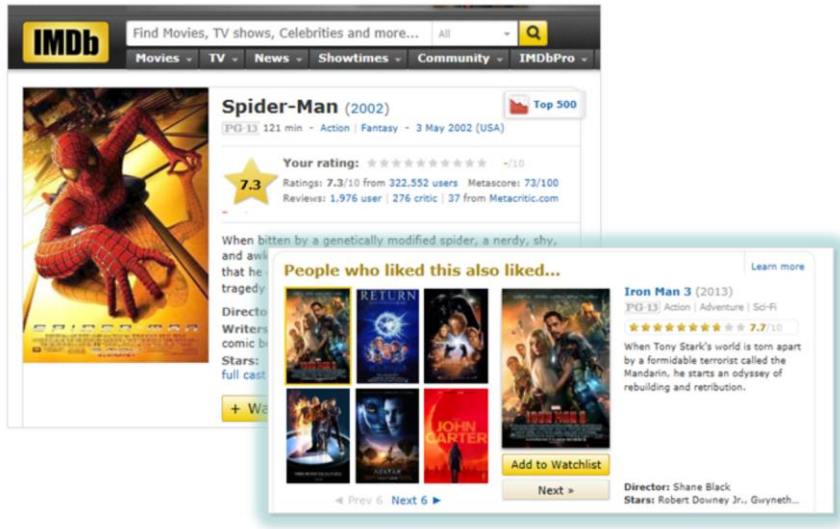
- Two core issues:
  - Sentence-level word embedding representation
  - How to encode emotional tendencies to word embeddings at all levels
    - Semi-supervised or supervised learning: The emotion tendency is encoded into the WE structure through the training process.





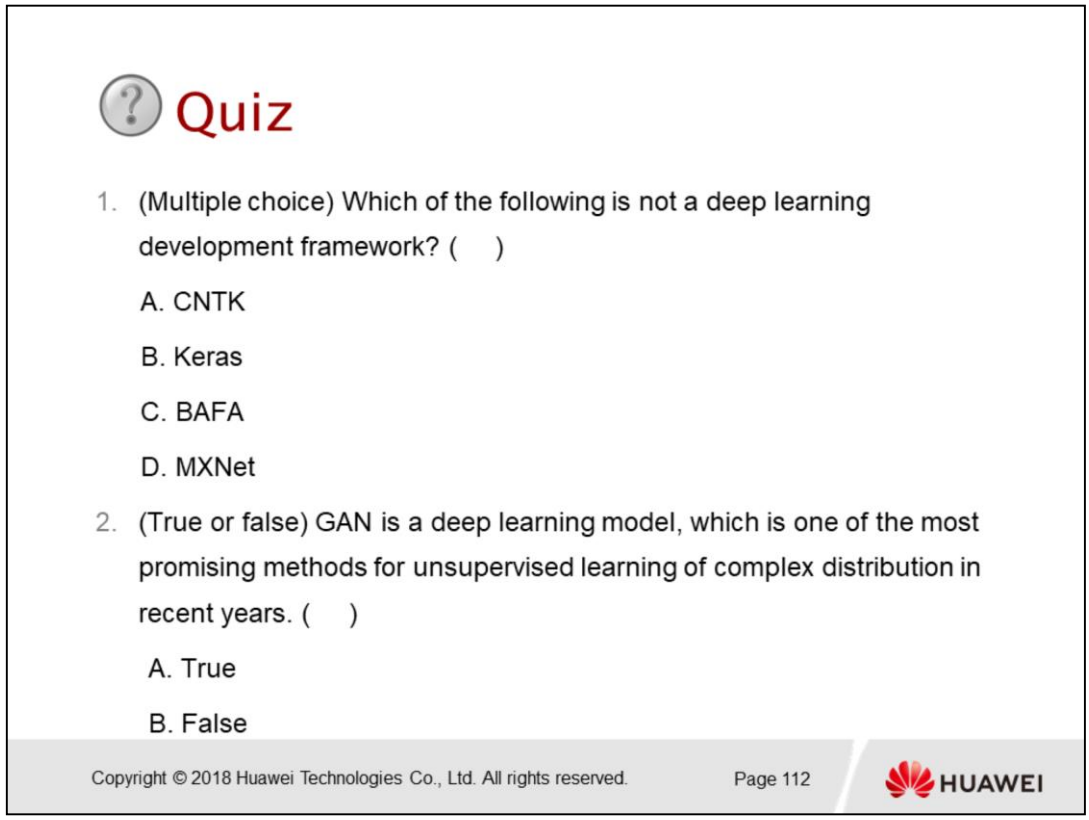
Speech Recognition

Recommendation System

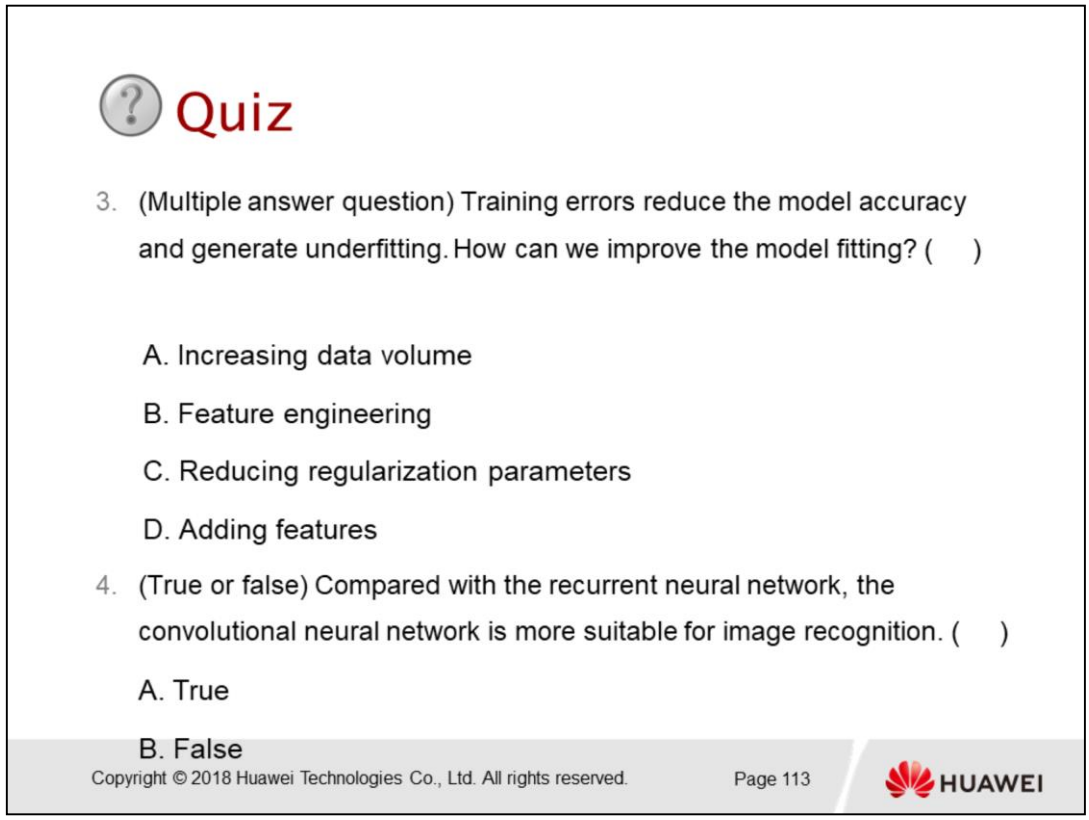


Summary

- This chapter introduces the propaedeutics of deep learning, including learning algorithms, common machine learning algorithms, hyperparameters, verification sets, parameter estimation, and estimation methods. Then it illustrates the definition, types, and development of neural networks as well as perceptron and its training rules.



- Answers: 1.C 2.A



- Answers: 3. ABD 4.A

Recommendations

- Huawei E-learning website:
  - http://support.huawei.com/learning/Index!toTrainIndex
- Huawei Support case library:
  - http://support.huawei.com/enterprise/servicecenter?lang=zh

Thank You

www.huawei.com