

信息检索与数据挖掘

第10章 文本分类

part1: 文本分类及朴素贝叶斯方法

part2: 基于向量空间的文本分类

part3: 支持向量机及机器学习方法

回顾：基于向量空间模型的文本分类的思路

- 向量空间模型

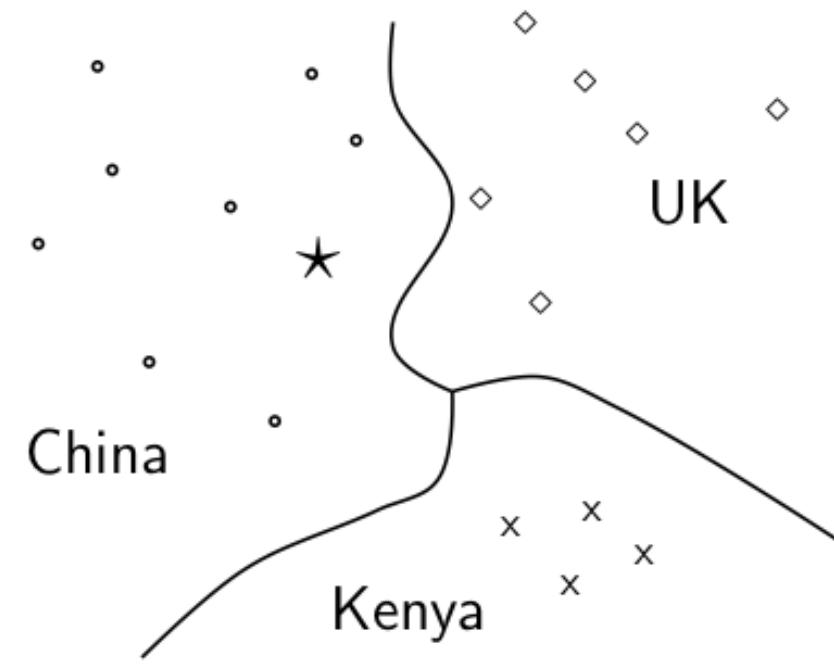
- 词项-文档矩阵：二值→计数→权重矩阵（tf-idf值）
- 相关性=向量距离：欧氏距离→夹角→余弦相似度

利用向量空间模型进行文本分类的思路主要基于邻近假设

(contiguity hypothesis) :

①同一类的文档会构成一个邻近区域，②而不同类的邻近区域之间是互不重叠的。

核心问题是如何找到分类面
决策边界 (decision boundary)



回顾： Rocchio分类方法

$$\vec{\mu}(c) = \frac{1}{|D_c|} \sum_{d \in D_c} \vec{v}(d)$$

其中， D_c 是文档集 D 中属于类别 c 的文档子集： $D_c = \{d : \langle d, c \rangle \in D\}$ 。这里将归一化的文档向量记为 $\rightarrow v(d)$

• 算法步骤

- (1)计算每个类的中心向量
- (2)将每篇测试文档分到离它最近的那个中心向量

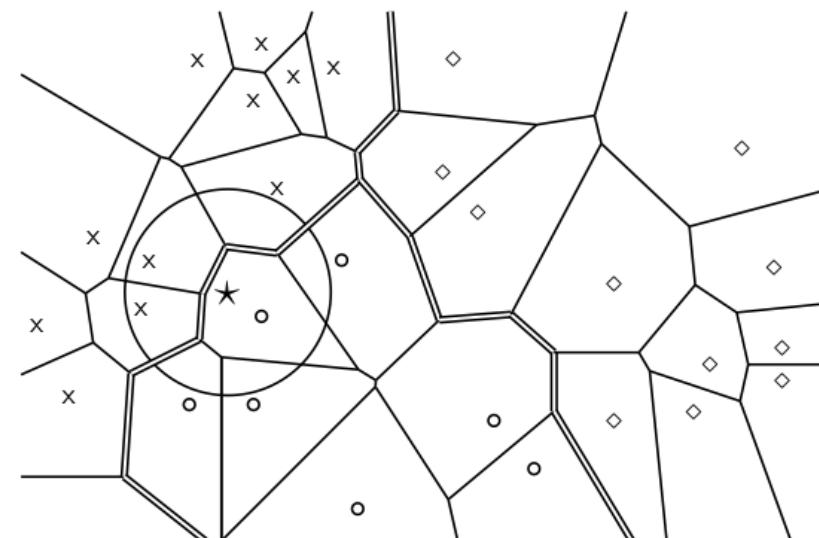
• 特性

- Rocchio 分类方法类的边界由那些到两个类质心等距的点集组成（超平面）。
- Rocchio 分类中的每个类别一定要近似球形，并且它们之间具有相似的球半径。当某类的内部文档并不近似分布在半径相近的球体之内时，其分类精度并不高。
- Rocchio算法的时间复杂度与NB方法在训练上具有相同的时间复杂度

小结：kNN（k近邻）方法

- 思路：将每篇测试文档分到训练集中离它最近的k篇文档所属类别中最多的那个类别
- kNN 的基本依据：根据邻近假设，一篇测试文档d将和其邻域中的训练文档应该具有相同的类别。

- 当训练集非常大的时候，kNN分类的精度很高
- 如果训练集很小， kNN可能效果很差。

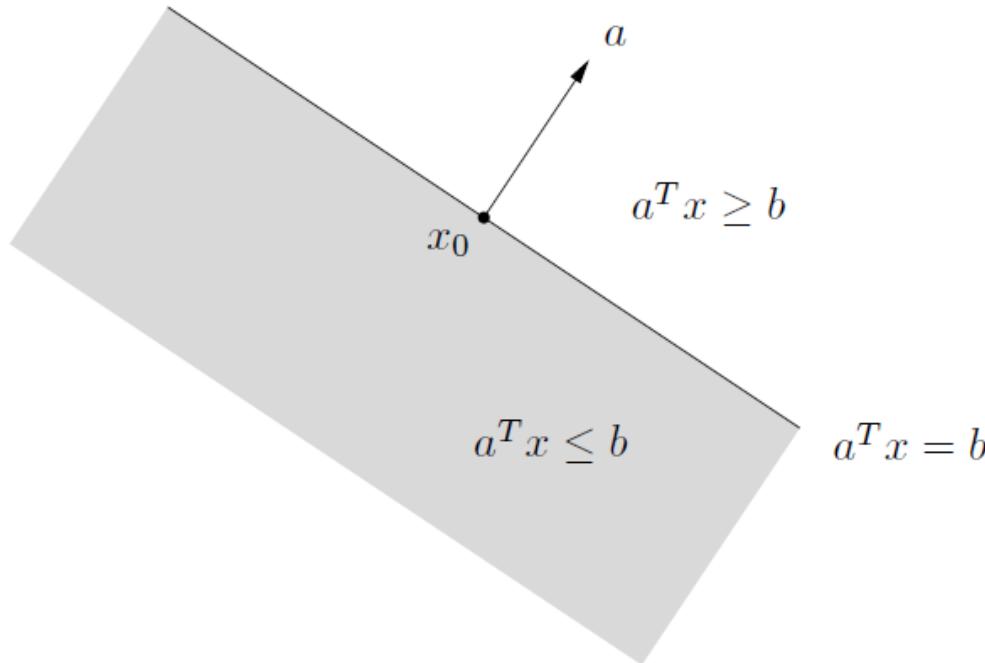


回顾：线性分类器

- 线性分类器：超平面 $\sum_i w_i x_i > \theta?$
- Two-class Rocchio as a linear classifier
- Naive Bayes is a linear classifier
- kNN不是线性分类器
- Linear / nonlinear classifiers
 - Noise documents
 - Fisher's linear discriminant
- single label problem → multilabel classification

回顾：单个超平面应对二分类问题

- A hyperplane divides R^n into **two halfspaces**. A (closed) halfspace is a set of the form $\{\mathbf{x} \mid \mathbf{a}^T \mathbf{x} \leq b\}$,



超平面只能将
空间分成两类

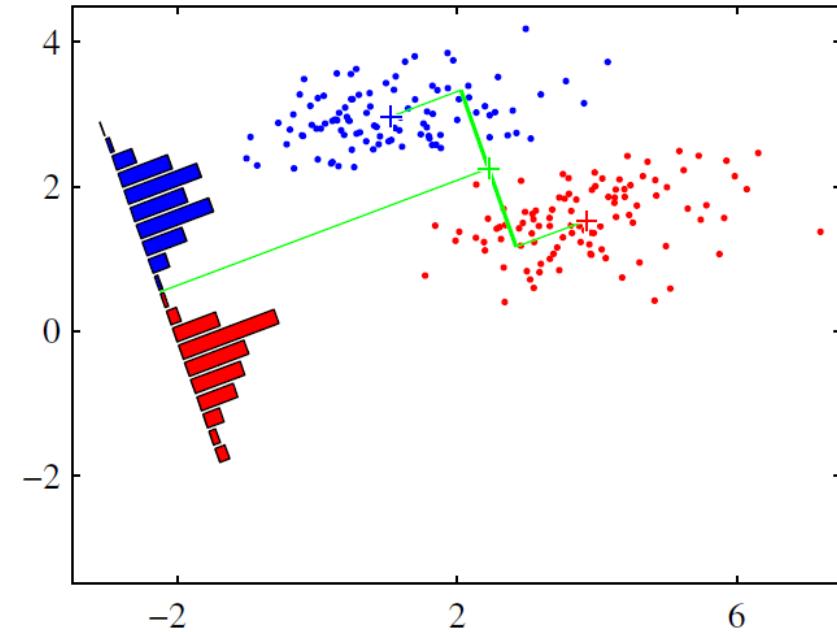
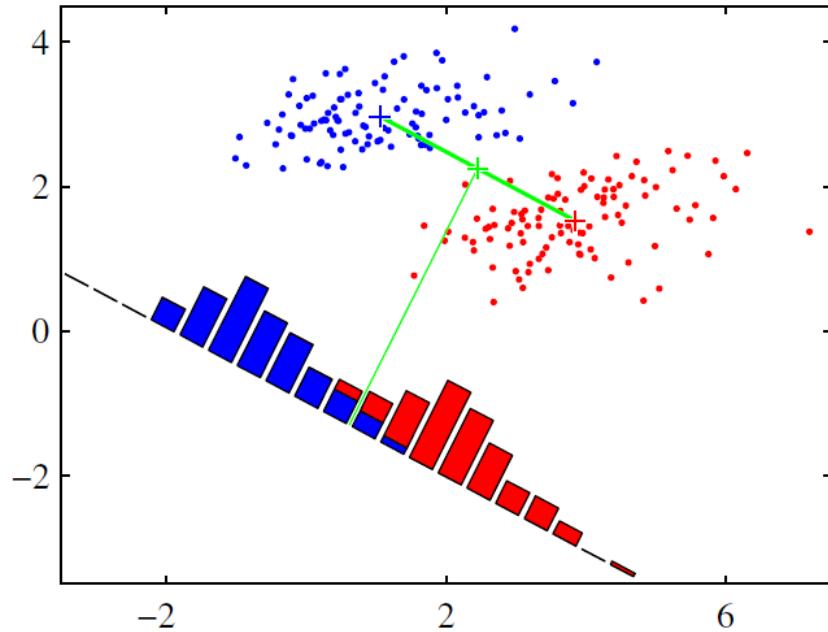
$n=2 \rightarrow$ 直线、 $n=3 \rightarrow$ 平面、 $n>3 \rightarrow$ 超平面

回顾：高维非线性分类→一维线性分类

Fisher's linear discriminant

Christopher M. Bishop

《Pattern Recognition and Machine Learning》 Chapter4



Fisher判别的基本思路就是投影。对P维空间中的某点 $\mathbf{x}=(x_1, x_2, \dots, x_p)$ 寻找一个能使它降为一维数值的线性函数 $y(\mathbf{x})= \sum C_j x_j$ 。 $y(\mathbf{x})$ 既能最大限度地缩小同类中各个样本点之间的差异，又能最大限度地扩大不同类别中各个样本点之间的差异，这样才可能获得较高的判别效率。

Fisher判别中最佳投影方向的获取即 C_j 的求解过程

Fisher's linear discriminant

最佳投影方向的求解

样本在d维X空间

(1) 各类样本均值向量 m_i

$$m_i = \frac{1}{n_i} \sum_{x_k \in X_i} x_k, \quad i = 1, 2$$

(2) 样本类内离散度矩阵 S_i 与总类内离散度矩阵 S_w

$$S_i = \sum_{\mathbb{X} \in S_i} (\mathbb{X} - m_i)(\mathbb{X} - m_i)^T, \quad i = 1, 2$$

$$S_w = S_1 + S_2$$

(3) 样本类间离散度矩阵 S_b

$$S_b = (m_1 - m_2)(m_1 - m_2)^T$$

S_B is the between-class covariance matrix
 S_W is the total within-class covariance matrix

样本在一维Y空间

(1) 各类样本均值 \tilde{m}_i

$$\tilde{m}_i = \frac{1}{N_i} \sum_{y \in Y_i} y, \quad i = 1, 2$$

(2) 样本类内离散度 \tilde{S}_i^2 和总类内离散度 \tilde{S}_w

$$\tilde{S}_i = \sum_{y \in Y_i} (y - \tilde{m}_i)^2, \quad i = 1, 2$$

$$\tilde{S}_w = \tilde{S}_1 + \tilde{S}_2$$

$$\tilde{S}_b = (\tilde{m}_1 - \tilde{m}_2)^2$$

Fisher准则函数

$$J_F(w) = \frac{\tilde{S}_b}{\tilde{S}_1 + \tilde{S}_2} = \frac{\mathbf{w}^T S_b \mathbf{w}}{\mathbf{w}^T S_w \mathbf{w}}$$

Fisher最佳投影方向求解

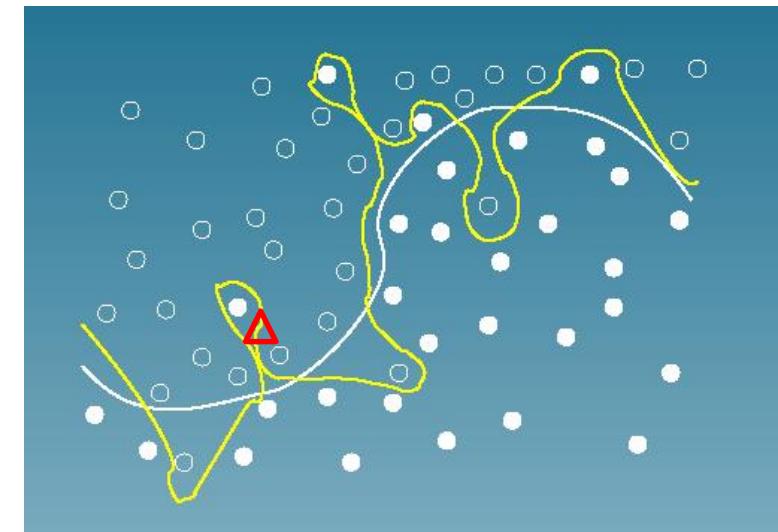
$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmax}} J_F(\mathbf{w})$$

回顾：学习方法的偏差、方差

对于某固定学习方法，训练集改变使分类函数变化，不同的分类函数产生的决策结果如果基本一致，我们说该学习方法的方差不大，如果不同分类函数的决策结果差异性很大，我们说该学习方法的方差大

白色分类边界：偏差大（一直存在错分）；但方差小（不怎么受零星出现在某一类别中的另一类别文档的影响）

黄色分类边界：偏差小，但是方差大（大部分情况下正确，但如果有文档出现在三角形所示位置，容易出现错分。故总体判决表现为时好时坏）



偏差-方差折衷准则：选择不同的权重对偏差和方差进行加权求和

过学习 (overfitting)：若学习样本不充分以及学习机器设计不合理，会导致训练误差过小，泛化能力下降。₉

课程内容

- 第1章 绪论
- 第2章 布尔检索及倒排索引
- 第3章 词项词典和倒排记录表
- 第4章 索引构建和索引压缩
- 第5章 向量模型及检索系统
- 第6章 检索的评价
- 第7章 相关反馈和查询扩展
- 第8章 概率模型
- 第9章 基于语言建模的检索模型
- **第10章 文本分类**
 - 文本分类及朴素贝叶斯方法
 - 基于向量空间的文本分类
 - 支持向量机及机器学习方法
- 第11章 文本聚类
- 第12章 Web搜索
- 第13章 多媒体信息检索
- 第14章 其他应用简介

Information Retrieval(IR): 从大规模非结构化数据（通常是文本）的集合（通常保存在计算机上）中找出满足用户信息需求的资料（通常是文档）的过程

数据挖掘（Data Mining）从大量的、不完全的、有噪声的、模糊的、随机的实际应用数据中，提取隐含在其中的、人们事先不知道的、但又是潜在有用的信息和知识的过程

本讲内容： 支持向量机及机器学习方法

- 支持向量机

- 二元线性SVM
- SVM用于非线性分类

- 机器学习方法

- 人工神经网络 (Artificial Neural Network, ANN)
- 深度学习 (Deep Learning) 现状
- 经典的深度学习模型/算法
 - 卷积神经网络 Convolutional Neural Networks (CNN)
 - 自动编码器 AutoEncoder
 - 受限玻尔兹曼机 Restricted Boltzmann Machine, RBM
 - 深度置信网络 (Deep Belief Nets, DBN)

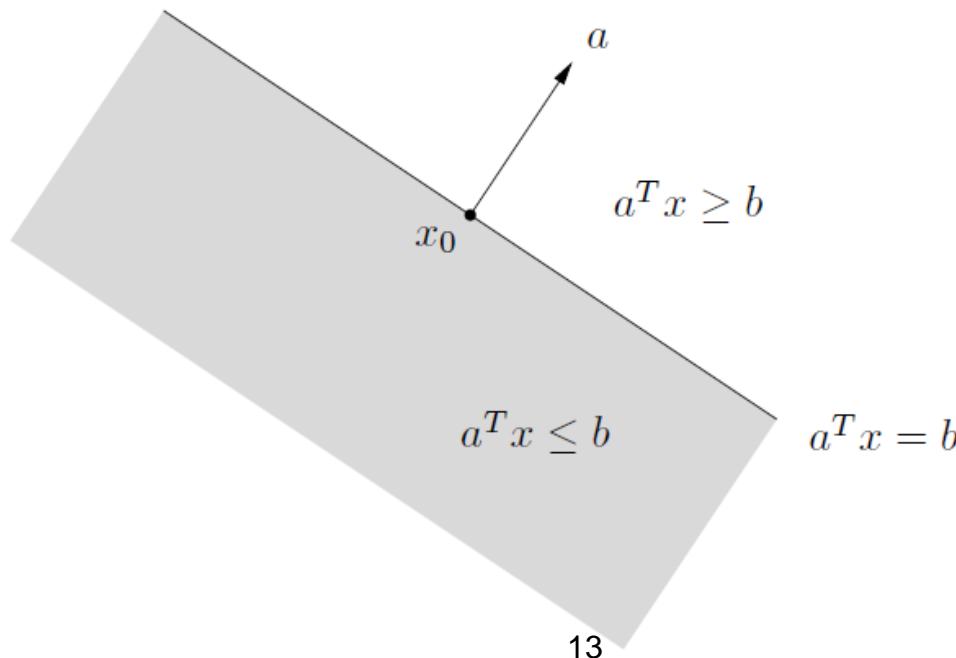
近20 年机器学习的研究产生了一系列的高效分类器，如支持向量机、提升式（boosted）决策树、正则化logistic回归、神经网络和随机森林（random forest）等。

本讲内容： 支持向量机及机器学习方法

- 支持向量机
 - 二元线性SVM
 - SVM用于非线性分类
- 机器学习方法
 - 人工神经网络 (Artificial Neural Network, ANN)
 - 深度学习 (Deep Learning) 现状
 - 经典的深度学习模型/算法
 - 卷积神经网络 Convolutional Neural Networks (CNN)
 - 自动编码器 AutoEncoder
 - 受限玻尔兹曼机 Restricted Boltzmann Machine, RBM
 - 深度置信网络 (Deep Belief Nets, DBN)

用超平面来分割多维空间

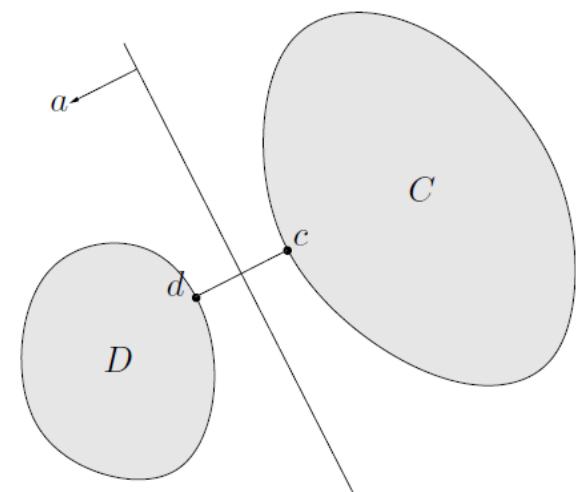
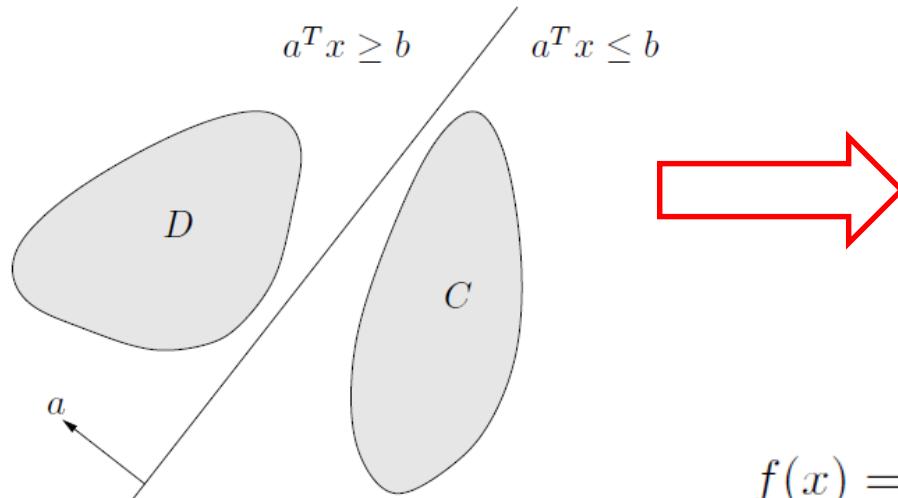
- A *hyperplane* is a set of the form $\{\mathbf{x} \mid \mathbf{a}^T \mathbf{x} = b\}$, where $\mathbf{a} \in \mathbb{R}^n$, $\mathbf{a} \neq 0$, and $b \in \mathbb{R}$.
- A hyperplane divides \mathbb{R}^n into **two halfspaces**. A (closed) halfspace is a set of the form $\{\mathbf{x} \mid \mathbf{a}^T \mathbf{x} \leq b\}$,



separating hyperplane theorem

- ***separating hyperplane theorem***: Suppose C and D are two convex sets that do not intersect, i.e., $C \cap D = \emptyset$. Then there exist $a \neq 0$ and b such that $a^T x \leq b$ for all $x \in C$ and $a^T x \geq b$ for all $x \in D$.

$$\text{dist}(C, D) = \inf\{\|u - v\|_2 \mid u \in C, v \in D\}$$



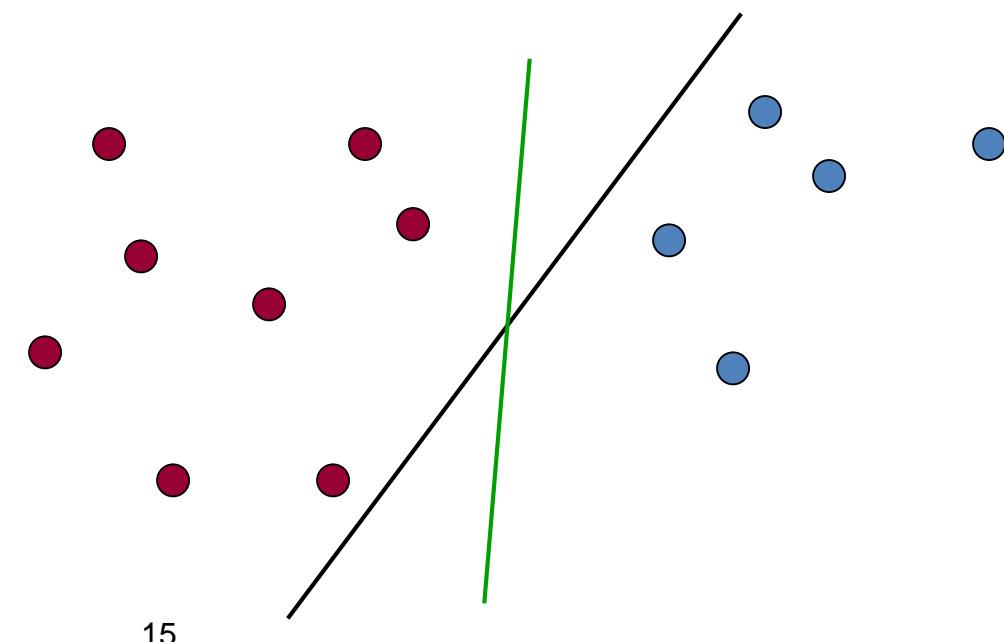
$$f(x) = a^T x - b = (d - c)^T (x - (1/2)(d + c))$$

应该选哪个超平面？

- 对于线性可分的训练集而言，肯定存在无穷多个分类面可以将两类完全正确地分开，但是不同的分类面在测试集的表现完全迥异...
- 对于新数据，有些分类器的错误率很高，有一些却很低。如感知机：通常很差；朴素贝叶斯、Rocchio：一般；线性SVM：好

对于二类线性可分问题存在无穷多个能区分两类的超平面

直观的感觉是绿色的更好



Distance between polyhedra

The (Euclidean) distance between the polyhedra $\mathcal{P}_1 = \{x \mid A_1x \preceq b_1\}$ and $\mathcal{P}_2 = \{x \mid A_2x \preceq b_2\}$ in \mathbf{R}^n is defined as

$$\text{dist}(\mathcal{P}_1, \mathcal{P}_2) = \inf\{\|x_1 - x_2\|_2 \mid x_1 \in \mathcal{P}_1, x_2 \in \mathcal{P}_2\}.$$

If the polyhedra intersect, the distance is zero.

To find the distance between \mathcal{P}_1 and \mathcal{P}_2 , we can solve the QP

$$\begin{aligned} & \text{minimize} && \|x_1 - x_2\|_2^2 \\ & \text{subject to} && A_1x_1 \preceq b_1, \quad A_2x_2 \preceq b_2, \end{aligned}$$

with variables $x_1, x_2 \in \mathbf{R}^n$. This problem is infeasible if and only if one of the polyhedra is empty. The optimal value is zero if and only if the polyhedra intersect, in which case the optimal x_1 and x_2 are equal (and is a point in the intersection $\mathcal{P}_1 \cap \mathcal{P}_2$). Otherwise the optimal x_1 and x_2 are the points in \mathcal{P}_1 and \mathcal{P}_2 , respectively, that are closest to each other. (We will study geometric problems involving distance in more detail in chapter 8.)

分类器的间隔

- 有些学习方法（如感知机）只需找到任一线性分界面即可，而另一些方法（如NB）则需按照某个准则找到最优的线性分界面。
- SVM是**最大间隔分类器**的一种，它是基于向量空间的机器学习方法，其目标是找到两个类别之间的一个决策边界，使之尽量远离训练集上的任意一点（当然一些离群点和噪音点可能不包括在内）。
- 对于SVM而言，它定义的准则是寻找一个离数据点最远的决策面。从决策面到最近数据点的距离决定了**分类器的间隔（margin）**。

支持向量机 (SVM, Support Vector Machines)

- SVM的目标是最大化决策超平面处的分类器间隔 (**maximize the margin**)
- SVM的决策函数通过训练集中的支持向量 (**support vectors**) 获取

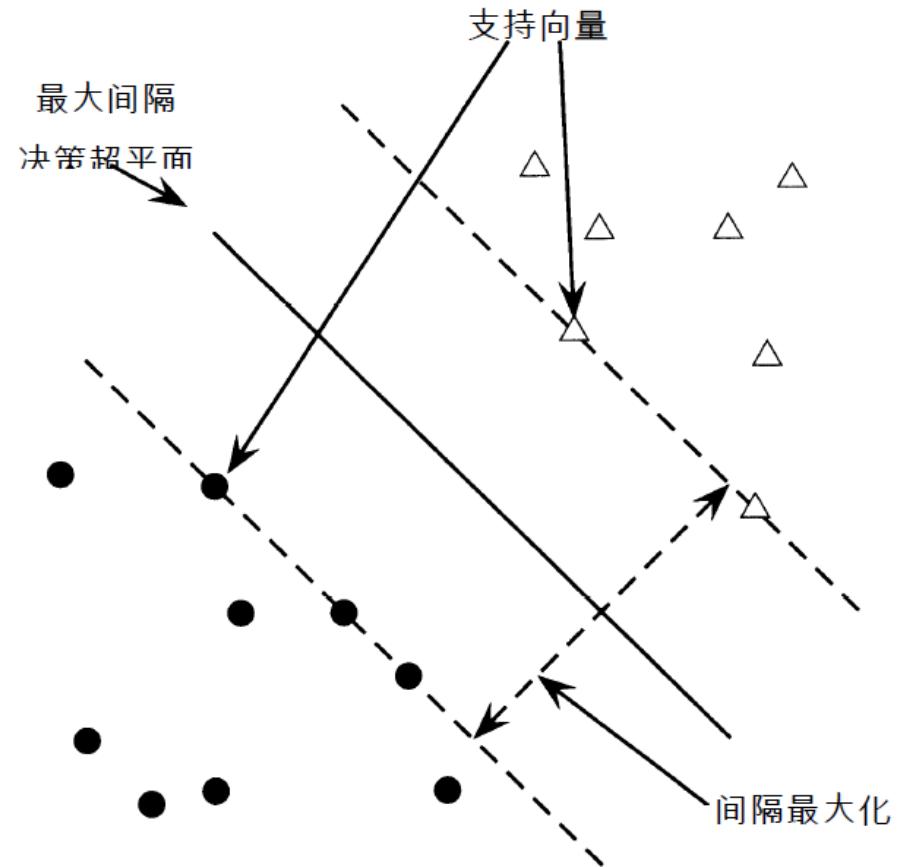


图 15-1 分类器间隔两端的 5 个点是支持向量

Margin: 分类器的间隔

一个SVM的例子：几何求解

- 最大间隔权重向量将和两类中距离最短的那条线段(直线)平行，即与连接点 $(1, 1)$ 和 $(2, 3)$ 的直线平行，这可以得到权重向量 $(1, 2)$ 。最优的分类直线与上述线段垂直并相交与其中点(中垂线)，因此它经过点 $(1.5, 2)$ 。

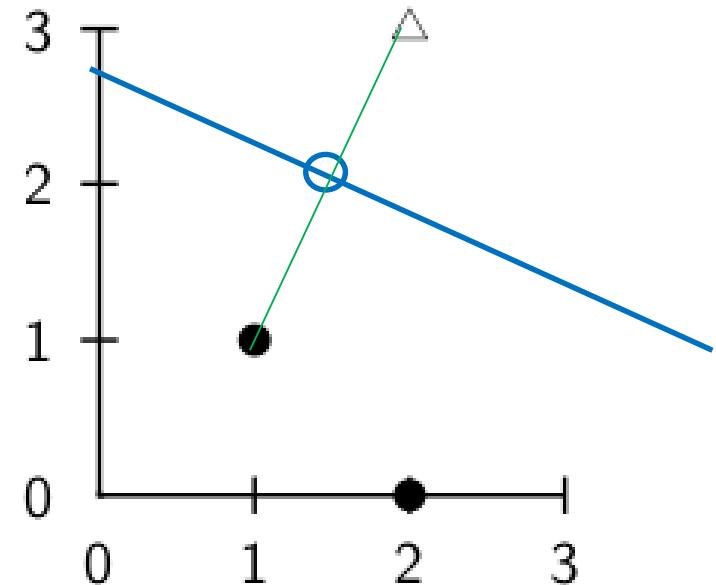
$$x_2 = -0.5x_1 + 2.75$$

于是，SVM的决策方程为：

$$y = x_1 + 2x_2 - 5.5$$

$y > 0$ 属于一类

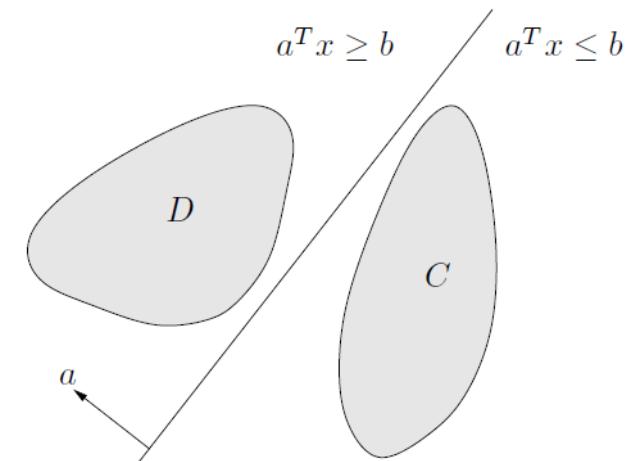
$y < 0$ 属于另一类



二元线性分类器的形式化定义

- w : decision hyperplane normal vector
- x_i : data point i
- y_i : class of data point i (+1 or -1)
- Classifier is: $f(x_i) = \text{sign}(w^T x_i + b)$

w : 决策超平面的法向量, 下图中a
 b : 决策超平面的截距



- 函数间隔
- Functional margin of x_i is: $y_i(w^T x_i + b)$
- Functional margin of dataset is twice the minimum functional margin for any point

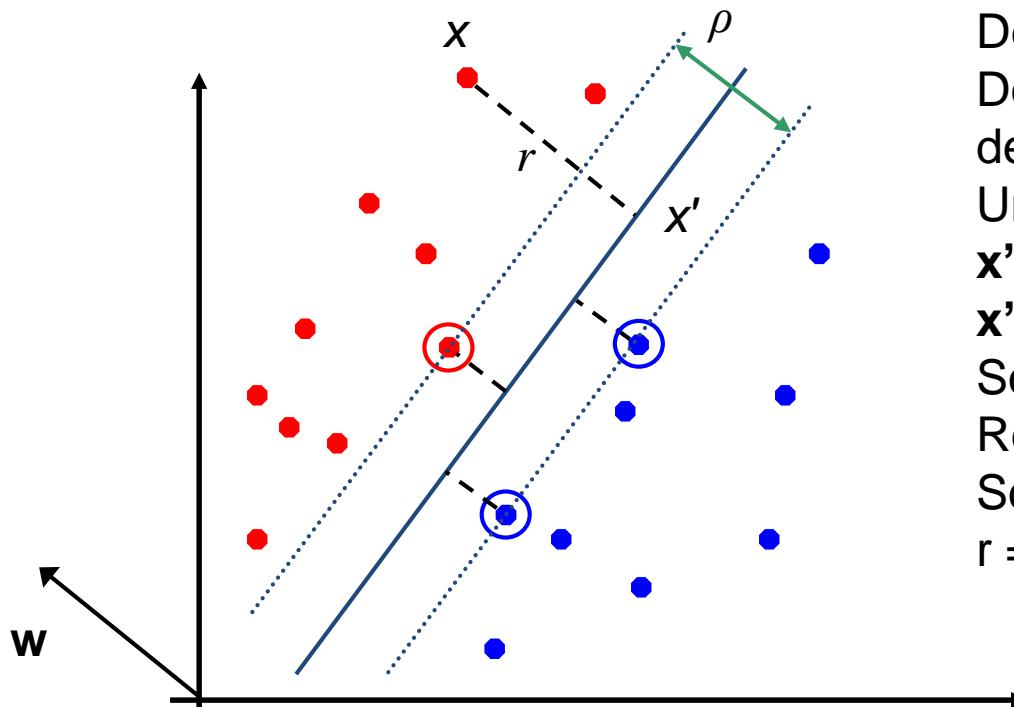
- The factor of 2 comes from measuring the whole width of the margin

$$\text{dist}(C, D) = \inf \{\|u - v\|_2 \mid u \in C, v \in D\}$$

Geometric Margin

分类器的几何间隔: 中间空白带的最大宽度,
该空白带可以用于将两类支持向量分开

- Distance from example to the separator is $r = y \frac{\mathbf{w}^T \mathbf{x} + b}{\|\mathbf{w}\|}$ 点 \mathbf{x} 到决策平面的距离 r : \mathbf{x} 向量在法向量 \mathbf{w} 上的投影
- Examples closest to the hyperplane are *support vectors*.
- *Margin ρ* of the separator is the width of separation between support vectors of classes.



Derivation of finding r :

Dotted line $\mathbf{x}' - \mathbf{x}$ is perpendicular to decision boundary so parallel to \mathbf{w} .

Unit vector is $\mathbf{w}/|\mathbf{w}|$, so line is $r\mathbf{w}/|\mathbf{w}|$.

$$\mathbf{x}' = \mathbf{x} - yr\mathbf{w}/|\mathbf{w}|.$$

$$\mathbf{x}' \text{ satisfies } \mathbf{w}^T \mathbf{x}' + b = 0.$$

$$\text{So } \mathbf{w}^T(\mathbf{x} - yr\mathbf{w}/|\mathbf{w}|) + b = 0$$

$$\text{Recall that } |\mathbf{w}| = \sqrt{\mathbf{w}^T \mathbf{w}}.$$

So, solving for r gives:

$$r = \frac{y(\mathbf{w}^T \mathbf{x} + b)}{|\mathbf{w}|}$$

Linear SVM Mathematically

The linearly separable case

- Assume that all data is at least distance 1 from the hyperplane, then the following two constraints follow for a training set $\{(x_i, y_i)\}$

$$\mathbf{w}^T \mathbf{x}_i + b \geq 1 \quad \text{if } y_i = 1$$

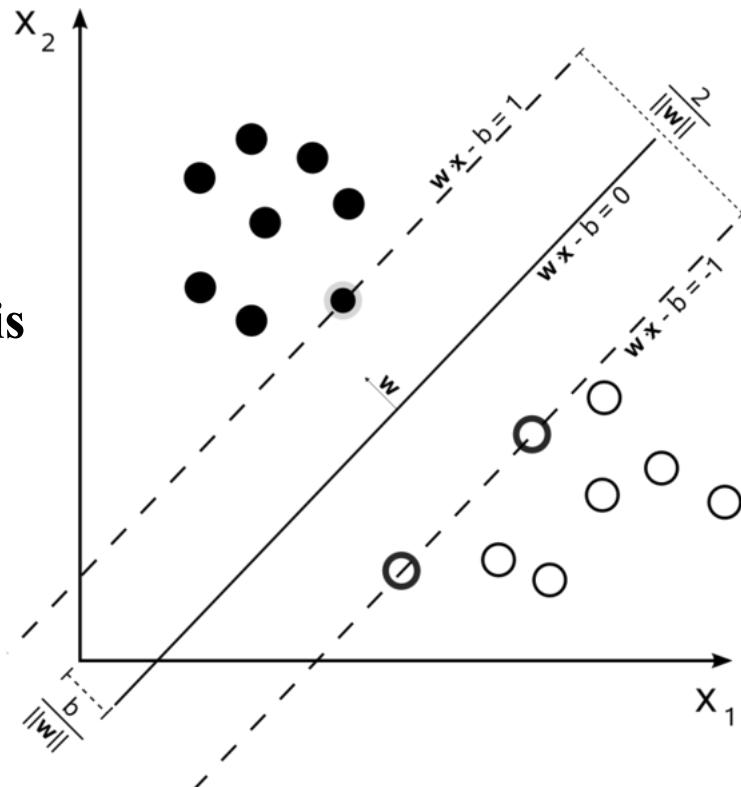
$$\mathbf{w}^T \mathbf{x}_i + b \leq -1 \quad \text{if } y_i = -1$$

- For support vectors, the inequality becomes an equality. Then, since each example's distance from the hyperplane is

$$r = y \frac{\mathbf{w}^T \mathbf{x} + b}{\|\mathbf{w}\|}$$

- The margin is:

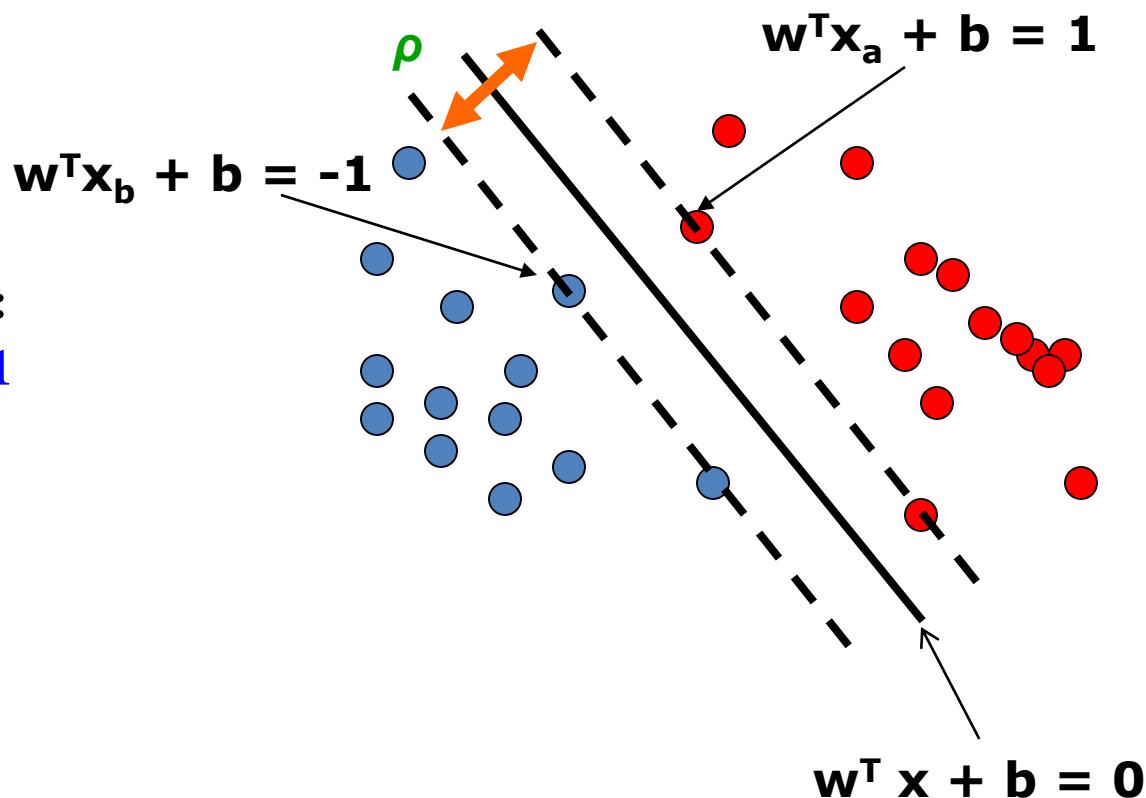
$$\rho = \frac{2}{\|\mathbf{w}\|}$$



Linear Support Vector Machine (SVM)

- **Hyperplane**

$$\mathbf{w}^T \mathbf{x} + b = 0$$



- **Extra scale constraint:**

$$\min_{i=1,\dots,n} |\mathbf{w}^T \mathbf{x}_i + b| = 1$$

- **This implies:**

$$\mathbf{w}^T (\mathbf{x}_a - \mathbf{x}_b) = 2$$

$$\rho = \|\mathbf{x}_a - \mathbf{x}_b\|_2 = 2/\|\mathbf{w}\|_2$$

Linear SVMs Mathematically (cont.)

- Then we can formulate the *quadratic optimization problem*:

Find \mathbf{w} and b such that

$$\rho = \frac{2}{\|\mathbf{w}\|} \text{ is maximized; and for all } \{(\mathbf{x}_i, y_i)\}$$

$$\mathbf{w}^T \mathbf{x}_i + b \geq 1 \text{ if } y_i = 1; \quad \mathbf{w}^T \mathbf{x}_i + b \leq -1 \quad \text{if } y_i = -1$$

- A better formulation ($\min \|\mathbf{w}\| = \max 1/\|\mathbf{w}\|$):

Find \mathbf{w} and b such that

$$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} \text{ is minimized;}$$

$$\text{and for all } \{(\mathbf{x}_i, y_i)\}: y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

上述问题实际上是在线性约束条件下的**二次函数优化**问题。该问题是数学中的基本问题之一，存在很多解决算法（比如，有一些二次规划库）

优化问题 (Optimization problems)

- 目标函数
- 不等式约束
- 等式约束

$$\begin{aligned} f_0(x) \\ f_i(x), i=1, \dots, m \\ h_i(x), i=1, \dots, p \end{aligned}$$

Source: 《Convex Optimization》, Stephen Boyd
Chapter 4 Convex optimization problems

We use the notation

$$\begin{aligned} & \text{minimize} && f_0(x) \\ & \text{subject to} && f_i(x) \leq 0, \quad i = 1, \dots, m \\ & && h_i(x) = 0, \quad i = 1, \dots, p \end{aligned} \tag{4.1}$$

to describe the problem of finding an x that minimizes $f_0(x)$ among all x that satisfy the conditions $f_i(x) \leq 0$, $i = 1, \dots, m$, and $h_i(x) = 0$, $i = 1, \dots, p$. We call $x \in \mathbf{R}^n$ the *optimization variable* and the function $f_0 : \mathbf{R}^n \rightarrow \mathbf{R}$ the *objective function* or *cost function*. The inequalities $f_i(x) \leq 0$ are called *inequality constraints*, and the corresponding functions $f_i : \mathbf{R}^n \rightarrow \mathbf{R}$ are called the *inequality constraint functions*. The equations $h_i(x) = 0$ are called the *equality constraints*, and the functions $h_i : \mathbf{R}^n \rightarrow \mathbf{R}$ are the *equality constraint functions*. If there are no constraints (*i.e.*, $m = p = 0$) we say the problem (4.1) is *unconstrained*.

QP问题

- *linear program (LP)*

$$\begin{aligned} & \text{minimize} && c^T x + d \\ & \text{subject to} && Gx \leq h \\ & && Ax = b \end{aligned}$$

$$G \in \mathbf{R}^{m \times n} \text{ and } A \in \mathbf{R}^{p \times n}$$

- *quadratic program (QP)*

$$\begin{aligned} & \text{minimize} && (1/2)x^T Px + q^T x + r \\ & \text{subject to} && Gx \leq h \\ & && Ax = b, \end{aligned}$$

$$P \in \mathbf{S}_+^n, G \in \mathbf{R}^{m \times n}, \text{ and } A \in \mathbf{R}^{p \times n}$$

Find \mathbf{w} and b such that

$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w}$ is minimized;

and for all $\{(\mathbf{x}_i, y_i)\}$: $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$



拉格朗日对偶

Source: 《Convex Optimization》 ,Stephen Boyd
Chapter 4 Convex optimization problems

$$\begin{aligned} & \text{minimize} && f_0(x) \\ & \text{subject to} && f_i(x) \leq 0, \quad i = 1, \dots, m \\ & && h_i(x) = 0, \quad i = 1, \dots, p, \end{aligned}$$

- The basic idea in Lagrangian duality is to take the constraints into account by augmenting the objective function with a weighted sum of the constraint functions.

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^m \lambda_i f_i(x) + \sum_{i=1}^p \nu_i h_i(x)$$

- The vectors λ and ν are called the *dual variables or Lagrange multiplier vectors* associated with the Problem.

$$g(\lambda, \nu) = \inf_{x \in \mathcal{D}} L(x, \lambda, \nu) = \inf_{x \in \mathcal{D}} \left(f_0(x) + \sum_{i=1}^m \lambda_i f_i(x) + \sum_{i=1}^p \nu_i h_i(x) \right)$$

Solving the Optimization Problem

不等式约束

Find w and b such that

$\Phi(w) = \frac{1}{2} w^T w$ is minimized;

and for all $\{(x_i, y_i)\}$: $y_i (w^T x_i + b) \geq 1$

上述问题实际上是在线性约束条件下的二次函数优化问题。该问题是数学中的基本问题之一，存在很多解决算法（比如，有一些二次规划库）

The solution involves constructing a *dual problem* where a *Lagrange multiplier* α_i is associated with every constraint in the primary problem:

每个拉格朗日因子 α_i 对应一个
不等式约束 $y_i (w^T x_i + b) \geq 1$

Find $\alpha_1 \dots \alpha_N$ such that

$Q(\alpha) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j x_i^T x_j$ is maximized and

- (1) $\sum \alpha_i y_i = 0$
- (2) $\alpha_i \geq 0$ for all α_i

等式约束

QP (Quadratic Programming, 二次规划)

Find w and b such that

$\Phi(w) = \frac{1}{2} w^T w$ is minimized;

and for all $\{(x_i, y_i)\}$: $y_i (w^T x_i + b) \geq 1$

- The solution involves constructing a *dual problem* where a *Lagrange multiplier* α_i is associated with every constraint in the primary problem:
- The solution has the form:

$$w = \sum \alpha_i y_i x_i \quad b = y_k - w^T x_k \text{ for any } x_k \text{ such that } \alpha_k \neq 0$$

- Each non-zero α_i indicates that corresponding x_i is a support vector.
- Then the **classifying function** will have the form:

$$f(x) = \sum \alpha_i y_i x_i^T x + b$$

求解拉格朗日对偶问题得到 α_i
进一步得到 w 和 b
故得到了分类函数

小结：SVM要点

$$f(\vec{x}) = \text{sign} \left(\sum_i \alpha_i y_i \vec{x}_i \cdot \vec{x} + b \right)$$

书上的公式

- 线性SVM的结果分类器为：

$$f(\mathbf{x}) = \sum_i \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$$

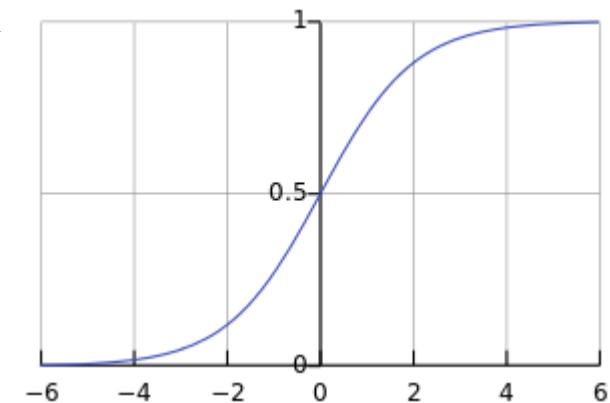
- SVM基本过程

- 基于给定训练数据集，通过二次优化过程寻找最佳的分类超平面
- 对于待分类的新数据点，利用分类函数计算该点到超平面的距离
- 距离的正负（分类函数的符号）决定了该数据点类别的归属
- 如果该点在分类器的间隔之内，分类器可以在原来的两个类之外，返回“类别未知”

- 分类函数值可以也转化为一个分类概率

- 通过Sigmoid函数拟合转化是一种常规做法

$$S(t) = \frac{1}{1 + e^{-t}}$$

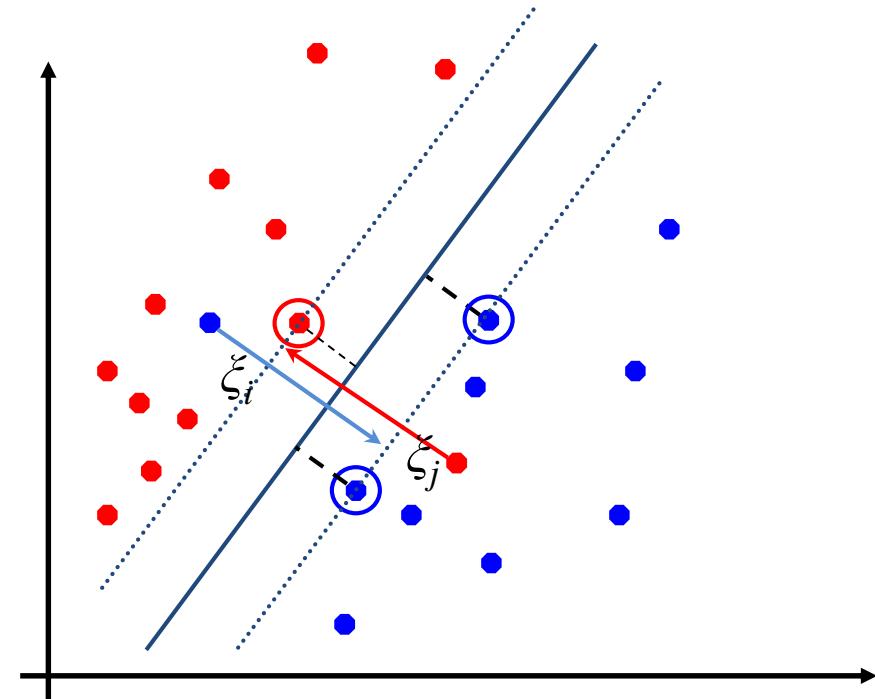


本讲内容：支持向量机及机器学习方法

- 支持向量机
 - 二元线性SVM
 - SVM用于非线性分类
- 机器学习方法
 - 人工神经网络 (Artificial Neural Network, ANN)
 - 深度学习 (Deep Learning) 现状
 - 经典的深度学习模型/算法
 - 卷积神经网络 Convolutional Neural Networks (CNN)
 - 自动编码器 AutoEncoder
 - 受限玻尔兹曼机 Restricted Boltzmann Machine, RBM
 - 深度置信网络 (Deep Belief Nets, DBN)

软间隔(Soft margin)分类

- If the training data is not linearly separable, *slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples.
- Allow some errors
 - Let some points be moved to where they belong, at a cost
- Still, try to minimize training set errors, and to place hyperplane “far” from each class (large margin)



Soft Margin Classification Mathematically

- The old formulation:

Find \mathbf{w} and b such that

$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w}$ is minimized and for all $\{(\mathbf{x}_i, y_i)\}$

$$y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

- The new formulation incorporating slack variables:

Find \mathbf{w} and b such that

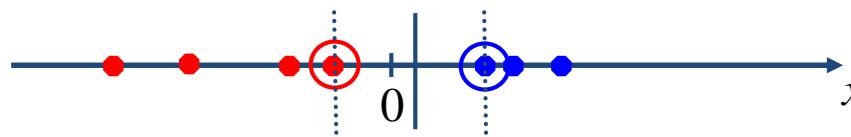
$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum \xi_i$ is minimized and for all $\{(\mathbf{x}_i, y_i)\}$

$$y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0 \text{ for all } i$$

- Parameter C can be viewed as a way to control overfitting – a regularization term

Non-linear SVMs

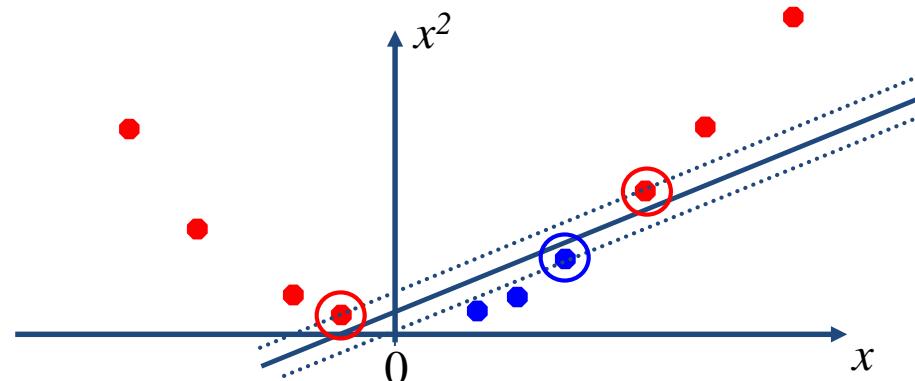
- Datasets that are linearly separable (with some noise) work out great:



- But what are we going to do if the dataset is just too hard?

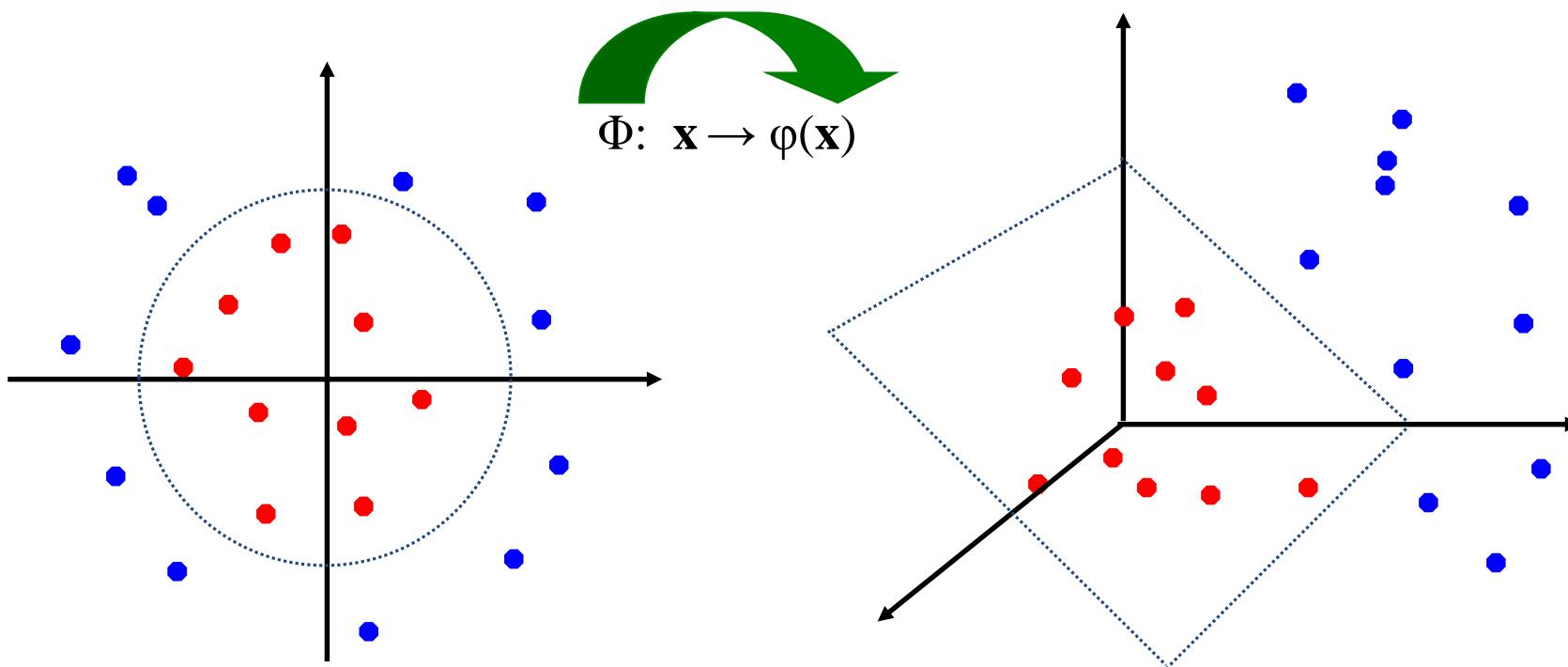


- How about ... mapping data to a higher-dimensional space:



Non-linear SVMs: Feature spaces

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:



利用核技巧 (Kernel Trick) 直接计算高维空间的分类函数

- 将原始的特征空间映射到某个更高维的线性可分的特征空间上去 $\Phi: \vec{x} \mapsto \Phi(\vec{x})$

$$f(\vec{x}) = sign\left(\sum_i \alpha_i y_i x_i^T \vec{x} + b\right) \quad \Rightarrow \quad f(\vec{x}) = sign\left(\sum_i \alpha_i y_i K(\vec{x}_i, \vec{x}) + b\right)$$

$K(\vec{x}_i, \vec{x}_j) = \vec{x}_i^T \vec{x}_j$

低维空间

高维空间

$$K(\vec{x}_i, \vec{x}_j) = \Phi(\vec{x}_i)^T \Phi(\vec{x}_j)$$

The ‘Kernel Trick’

$$f(\vec{x}) = \text{sign} \left(\sum_i \alpha_i y_i K(\vec{x}_i, \vec{x}) + b \right)$$

- SVM 线性分类器依赖于数据点之间的内积操作。令 $K(x_i, x_j) = x_i^T x_j$
- If every datapoint is mapped into high-dimensional space via some transformation Φ : $x \rightarrow \phi(x)$, the inner product becomes:

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

- A *kernel function* is some function that corresponds to an inner product in some expanded feature space.
- 假设我们通过某个映射函数 ϕ 可以将原始空间的点映射到新空间，即 $\phi : x \rightarrow \phi(x)$ 。那么，新空间下的点积计算为 $\phi(x_i)^T \phi(x_j)$ 。如果能够证明该点积（也就是一个实数）能够通过原始数据点简单高效地计算出来，那么我们就不必真的要将 $x \rightarrow \phi(x)$ 。相反，我们可以直接通过 $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ 来计算。核函数 K 实际上对应于新特征空间上的内积。

二维空间上一个二次核函数的例子

- Example:

2-dimensional vectors $\mathbf{x} = [x_1 \ x_2]$; let $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^2$,

下面我们说明这是一个核函数，也就是说，
对于某个函数 ϕ 有 $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$:

$$\text{考虑 } \phi(\mathbf{x}) = [1 \ x_1^2 \ \sqrt{2}x_1x_2 \ x_2^2 \ \sqrt{2}x_1 \ \sqrt{2}x_2]$$

$$\begin{aligned} K(\mathbf{x}_i, \mathbf{x}_j) &= (1 + \mathbf{x}_i^T \mathbf{x}_j)^2 \\ &= 1 + x_{i1}^2 x_{j1}^2 + 2 x_{i1} x_{j1} x_{i2} x_{j2} + x_{i2}^2 x_{j2}^2 + 2 x_{i1} x_{j1} + 2 x_{i2} x_{j2} \\ &= [1 \ x_{i1}^2 \ \sqrt{2}x_{i1}x_{i2} \ x_{i2}^2 \ \sqrt{2}x_{i1} \ \sqrt{2}x_{i2}]^T [1 \ x_{j1}^2 \ \sqrt{2}x_{j1}x_{j2} \ x_{j2}^2 \ \sqrt{2}x_{j1} \ \sqrt{2}x_{j2}] \\ &= \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \end{aligned}$$

SVM是一个效果非常好的文本分类器

表15-2 SVM分类算法的正确率-召回率等值点上的 F_1 值 (Joachims 2002a, p. 114)。下面给出了Reuters - 21578语料中10个最大类上的 F_1 结果和所有90个类别的微平均值 (rbf指径向基核函数)

	NB	Rocchio	dec. Trees	kNN	Linear SVM		rbf-SVM
					C=0.5	C=1.0	$\sigma \approx 7$
earn	96.0	96.1	96.1	97.8	98.0	98.2	98.1
acq	90.7	92.1	85.3	91.8	95.5	95.6	94.7
money-fx	59.6	67.6	69.4	75.4	78.8	78.5	74.3
grain	69.8	79.5	89.1	82.6	91.9	93.1	93.4
crude	81.2	81.5	75.5	85.8	89.4	89.4	88.7
trade	52.2	77.4	59.2	77.9	79.2	79.2	76.6
interest	57.6	72.5	49.1	76.7	75.6	74.8	69.1
ship	80.9	83.1	80.9	79.8	87.4	86.5	85.8
wheat	63.4	79.4	85.5	72.9	86.6	86.8	82.4
corn	45.2	62.2	87.7	71.4	87.5	87.8	84.6
microavg.	72.3	79.9	79.4	82.6	86.7	87.5	86.4

小结：非线性SVM与Kernels

- **Soft Margin Classification**

Find \mathbf{w} and b such that

$$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum \xi_i \quad \text{is minimized and for all } \{(\mathbf{x}_i, y_i)\}$$
$$y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0 \text{ for all } i$$

- **Non-linear SVMs**

- 通过空间映射将原始空间映射到新空间，为避免显式的映射函数，引入核函数(定义在原始空间下但是结果是新空间下的内积函数)

- **Common kernels**

- Linear

- Polynomial (多项式核)

- Radial basis function (径向基核)

$$K(\vec{x}, \vec{z}) = (1 + \vec{x} \cdot \vec{z})^d$$

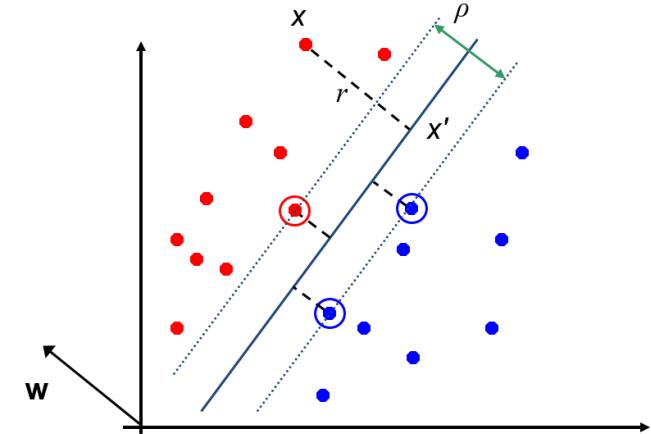
$$K(\vec{x}, \vec{z}) = e^{-(\vec{x}-\vec{z})/(2\sigma^2)}$$

- **Haven't been very useful in text classification**

小结：SVM

- 线性SVM

Find \mathbf{w} and b such that 通过拉格朗日对偶问题求解
 $\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w}$ is minimized and for all $\{(\mathbf{x}_i, y_i)\}$
 $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$



- SVM用于非线性问题

- Soft Margin

Find \mathbf{w} and b such that 原优化问题进行松弛
 $\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum \xi_i$ is minimized and for all $\{(\mathbf{x}_i, y_i)\}$
 $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

- Non-linear SVMs

$$K(\vec{x}_i, \vec{x}_j) = \vec{x}_i^T \vec{x}_j$$

$$K(\vec{x}_i, \vec{x}_j) = \Phi(\vec{x}_i)^T \Phi(\vec{x}_j)$$

$$f(\vec{x}) = \text{sign} \left(\sum_i \alpha_i y_i K(\vec{x}_i, \vec{x}) + b \right)$$

核技巧使高维特征空间中内积计算简单化

本讲内容：支持向量机及机器学习方法

- 支持向量机
 - 二元线性SVM
 - SVM用于非线性分类
- 机器学习方法
 - 人工神经网络（Artificial Neural Network, ANN）
 - 深度学习（Deep Learning）现状
 - 经典的深度学习模型/算法
 - 卷积神经网络 Convolutional Neural Networks (CNN)
 - 自动编码器 AutoEncoder
 - 受限玻尔兹曼机 Restricted Boltzmann Machine, RBM
 - 深度置信网络 (Deep Belief Nets, DBN)

人工神经网络

Artificial Neural Network, ANN

- 人工神经网络算法模拟生物神经网络，是一类模式匹配算法。通常用于解决分类和回归问题。人工神经网络是机器学习的一个庞大的分支，有几百种不同的算法（其中深度学习就是其中的一类算法）。

- 神经网络的缺点：

- (1)需要很长的训练时间
- (2)需要大量的参数
- (3)可解释性差

- 神经网络的优点：

- (1)对噪声数据的高承受能力
- (2)对未经训练的数据的模式分类能力

Multi-Layer Neural Network

神经元、激励函数

- 输入

x_1, x_2, x_3 (and a +1 intercept term)

- 输出

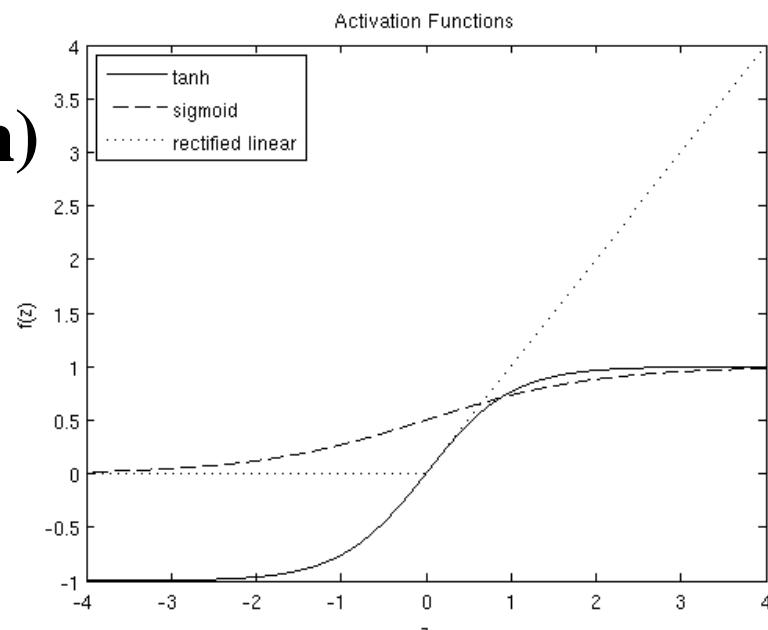
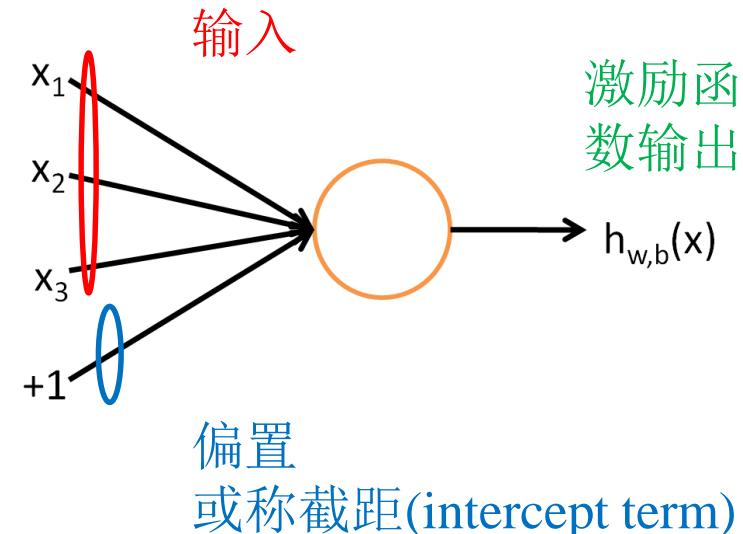
$$h_{W,b}(x) = f(W^T x) = f(\sum_{i=1}^3 W_i x_i + b)$$

- 激励函数(activation function)

$$f(z) = \frac{1}{1 + \exp(-z)}$$

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

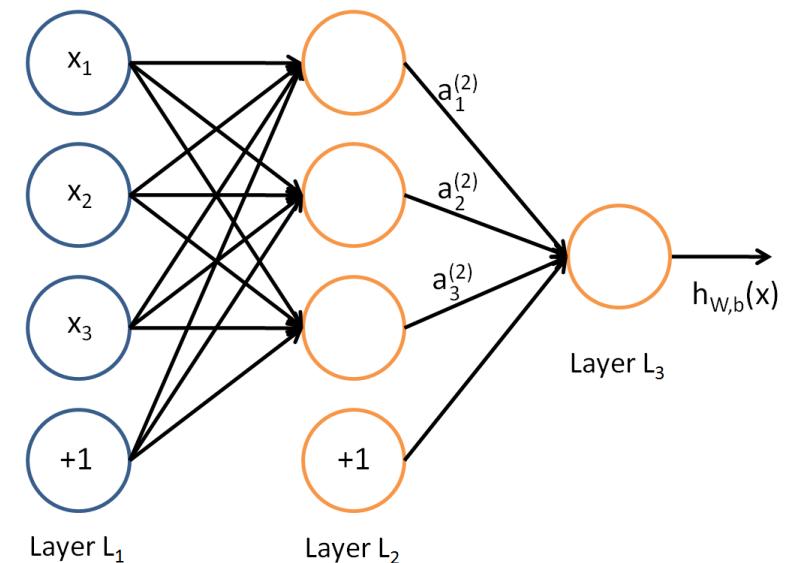
$$f(z) = \max(0, z)$$



Neural Network model

神经网络由神经元组成

- x_i 输入
- $+1$ 偏置
- input layer
- hidden layer
- output layer



$a_i^{(l)}$ 第 l 层第*i*单元激励函数输出
 $b_i^{(l)}$ 第 l 层第*i*单元的bias(偏置)

训练：获取各层的权重参数 $W^{(l)}$ 和偏置 $b^{(l)}$

$$a_1^{(2)} = f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)})$$

$$a_2^{(2)} = f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)})$$

$$a_3^{(2)} = f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)})$$

$$h_{W,b}(x) = a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})$$

训练过程可用梯度下降法来迭代求解 batch gradient descent

$h(\theta)$ 是要拟合的函数， $J(\theta)$ 代价（损失）函数。 θ 是参数，即要迭代求解的值， θ 求解出来了那最终要拟合的函数 $h(\theta)$ 就出来了。其中 m 是训练集的记录条数， j 是参数的个数

$$h(\theta) = \sum_{j=0}^n \theta_j x_j$$

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (y^i - h_\theta(x^i))^2$$

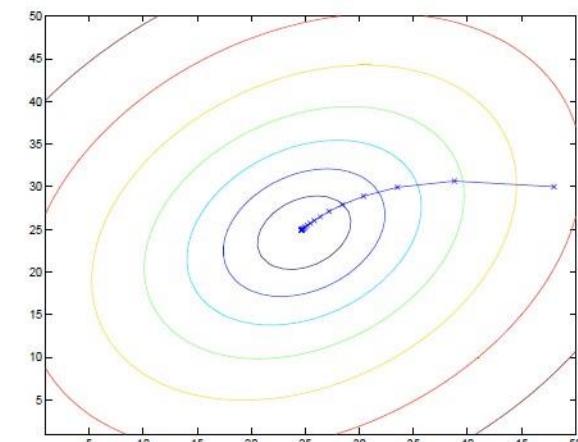
批量梯度下降的求解思路为：

(1) 将 $J(\theta)$ 对 θ 求偏导，得到每个 θ 对应的梯度

$$\frac{\partial J(\theta)}{\partial \theta_j} = -\frac{1}{m} \sum_{i=1}^m (y^i - h_\theta(x^i)) x_j^i$$

(2) 按每个参数 θ 的梯度负方向，来更新每个 θ

$$\theta_j' = \theta_j + \frac{1}{m} \sum_{i=1}^m (y^i - h_\theta(x^i)) x_j^i$$



forward propagation

$$a_1^{(2)} = f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)})$$

$$a_2^{(2)} = f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)})$$

$$a_3^{(2)} = f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)})$$

$$h_{W,b}(x) = a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})$$

$z_i^{(l)}$ denote total weighted sum of inputs to unit i in layer l

By organizing our parameters in matrices and using matrix-vector operations, we can take advantage of fast linear algebra routines to quickly perform calculations in our network.

$$\begin{aligned} z^{(l+1)} &= W^{(l)}a^{(l)} + b^{(l)} \\ a^{(l+1)} &= f(z^{(l+1)}) \end{aligned}$$

写成矢量形式

$$z_i^{(2)} = \sum_{j=1}^n W_{ij}^{(1)}x_j + b_i^{(1)}$$

$$a_i^{(l)} = f(z_i^{(l)})$$

$$z^{(2)} = W^{(1)}x + b^{(1)}$$

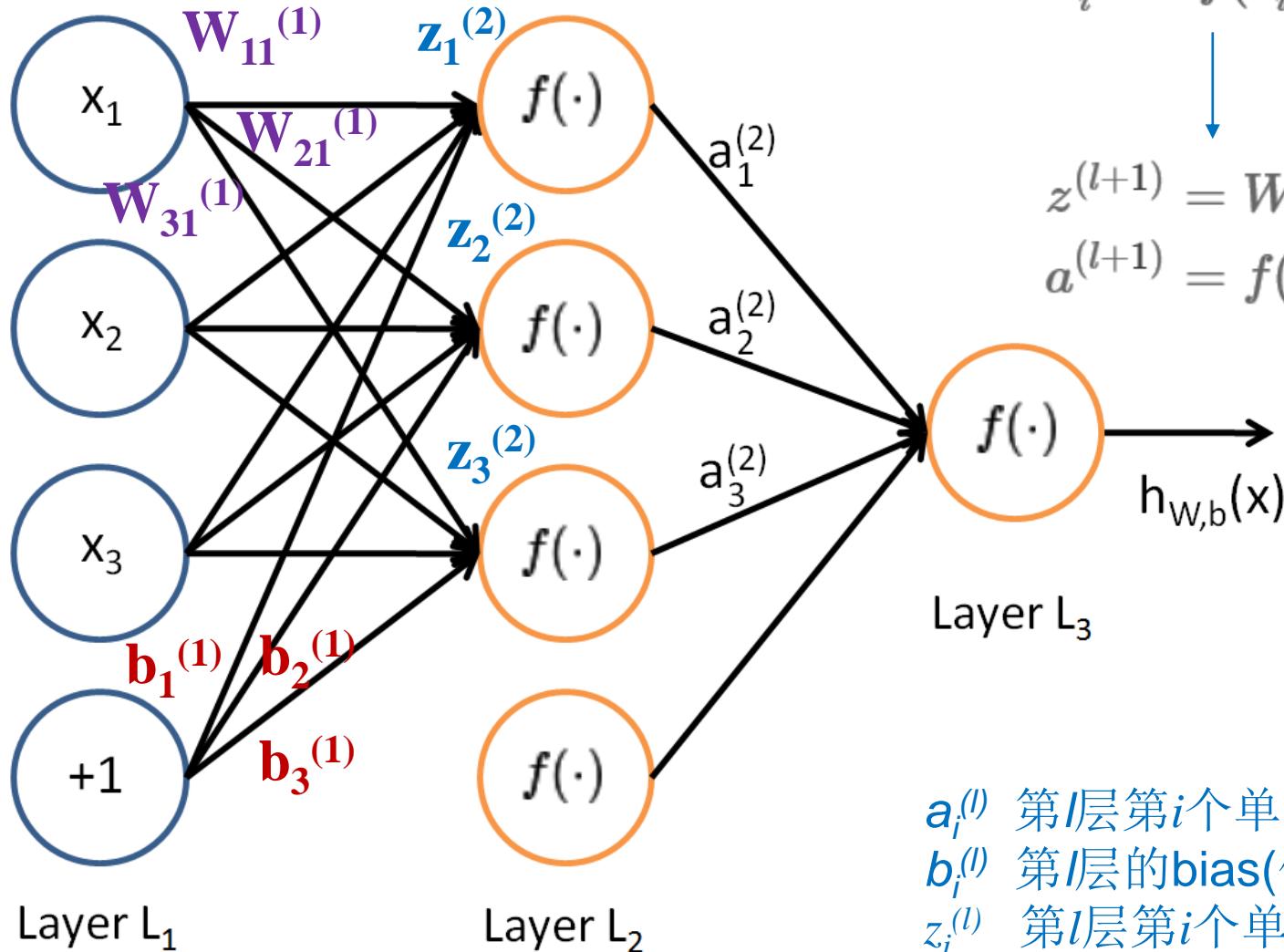
$$a^{(2)} = f(z^{(2)})$$

$$z^{(3)} = W^{(2)}a^{(2)} + b^{(2)}$$

$$h_{W,b}(x) = a^{(3)} = f(z^{(3)})$$

forward propagation

符号定义示例



$$z_i^{(2)} = \sum_{j=1}^n W_{ij}^{(1)} x_j + b_i^{(1)}$$

$$a_i^{(l)} = f(z_i^{(l)})$$

矢量形式

$$z^{(l+1)} = W^{(l)} a^{(l)} + b^{(l)}$$

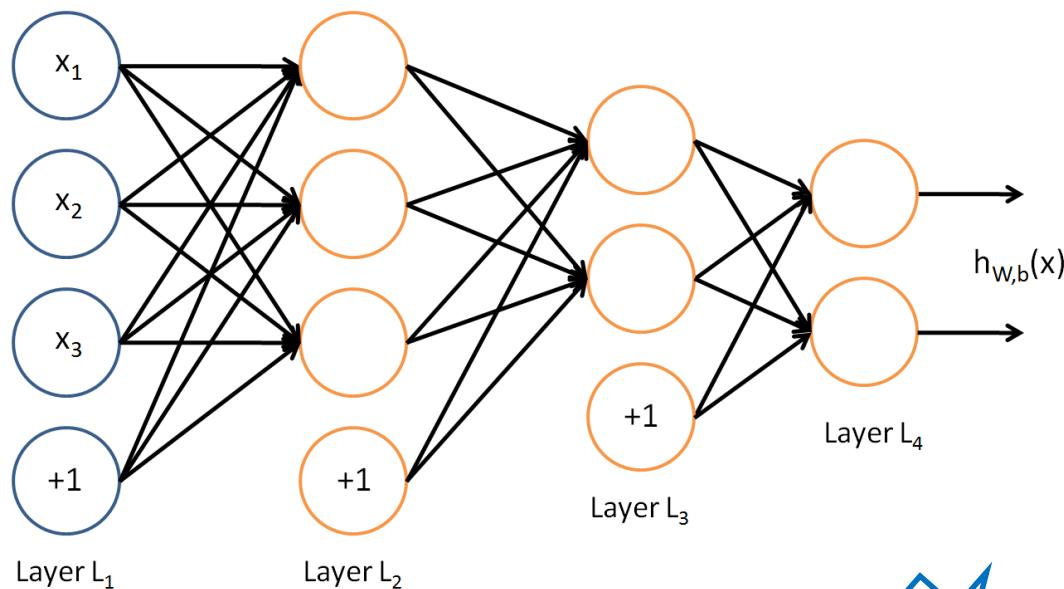
$$a^{(l+1)} = f(z^{(l+1)})$$

$$h_{W,b}(x)$$

Layer L₃ $a_i^{(l)}$ 第 l 层第 i 个单元激励函数输出 $b_i^{(l)}$ 第 l 层的bias(偏置) $z_i^{(l)}$ 第 l 层第 i 个单元的总输入

更复杂结构的神经网络

多个输出



$a_i^{(l)}$ 第 l 层第*i*个单元激励函数输出

$b_i^{(l)}$ 第 l 层的bias偏置

$z_i^{(l)}$ 第 l 层第*i*个单元的总输入

$a^{(l)}$ 第 l 层激励函数输出

$b^{(l)}$ 第 l 层的偏置

$z^{(l)}$ 第 l 层的总输入



To train this network,
we would need training
examples $(x^{(i)}, y^{(i)})$
where $y^{(i)} \in \mathbb{R}^2$.

训练：获取各层的权
重参数 $W^{(l)}$ 和偏置 $b^{(l)}$

$$z^{(l+1)} = W^{(l)} a^{(l)} + b^{(l)}$$

$$a^{(l+1)} = f(z^{(l+1)})$$

如何完成神经网络的训练？

求出使代价函数 cost function 最小的参数

- m 个训练样本 $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$
- 针对单个训练样本定义代价函数

$$J(W, b; x, y) = \frac{1}{2} \|h_{W,b}(x) - y\|^2$$

- m 个训练样本的总体代价函数为

$$\begin{aligned} J(W, b) &= \left[\frac{1}{m} \sum_{i=1}^m J(W, b; x^{(i)}, y^{(i)}) \right] + \frac{\lambda}{2} \sum_{l=1}^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (W_{ji}^{(l)})^2 \\ &= \left[\frac{1}{m} \sum_{i=1}^m \left(\frac{1}{2} \|h_{W,b}(x^{(i)}) - y^{(i)}\|^2 \right) \right] + \frac{\lambda}{2} \sum_{l=1}^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (W_{ji}^{(l)})^2 \end{aligned}$$

sum-of-squares error

weight decay

迭代法求解使代价函数最小的参数W^(l)和b^(l)

batch gradient descent

$$J(W, b) = \left[\frac{1}{m} \sum_{i=1}^m \left(\frac{1}{2} \|h_{W,b}(x^{(i)}) - y^{(i)}\|^2 \right) \right] + \frac{\lambda}{2} \sum_{l=1}^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} \left(W_{ji}^{(l)} \right)^2$$

Our goal is to minimize $J(W, b)$ as a function of W and b . To train our neural network, we will initialize each parameter $\underline{W_{ij}^{(l)}}$ and each $\underline{b_i^{(l)}}$ to a small random value near zero (say according to a $Normal(0, \epsilon^2)$) distribution for some small ϵ , say 0.01), and then apply an optimization algorithm such as batch gradient descent. Since $J(W, b)$ is a non-convex function, gradient descent is susceptible to local optima; however, in practice gradient descent usually works fairly well.

训练：参数的学习过程

$$\begin{aligned}
 J(W, b) &= \left[\frac{1}{m} \sum_{i=1}^m J(W, b; x^{(i)}, y^{(i)}) \right] + \frac{\lambda}{2} \sum_{l=1}^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (W_{ji}^{(l)})^2 \\
 &= \left[\frac{1}{m} \sum_{i=1}^m \left(\frac{1}{2} \|h_{W,b}(x^{(i)}) - y^{(i)}\|^2 \right) \right] + \frac{\lambda}{2} \sum_{l=1}^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (W_{ji}^{(l)})^2
 \end{aligned}$$

learning rate

$$\begin{aligned}
 \frac{\partial}{\partial W_{ij}^{(l)}} J(W, b) &= \left[\frac{1}{m} \sum_{i=1}^m \frac{\partial}{\partial W_{ij}^{(l)}} J(W, b; x^{(i)}, y^{(i)}) \right] + \lambda W_{ij}^{(l)} \\
 \frac{\partial}{\partial b_i^{(l)}} J(W, b) &= \frac{1}{m} \sum_{i=1}^m \frac{\partial}{\partial b_i^{(l)}} J(W, b; x^{(i)}, y^{(i)})
 \end{aligned}$$

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha \frac{\partial}{\partial W_{ij}^{(l)}} J(W, b)$$

$$b_i^{(l)} = b_i^{(l)} - \alpha \frac{\partial}{\partial b_i^{(l)}} J(W, b)$$

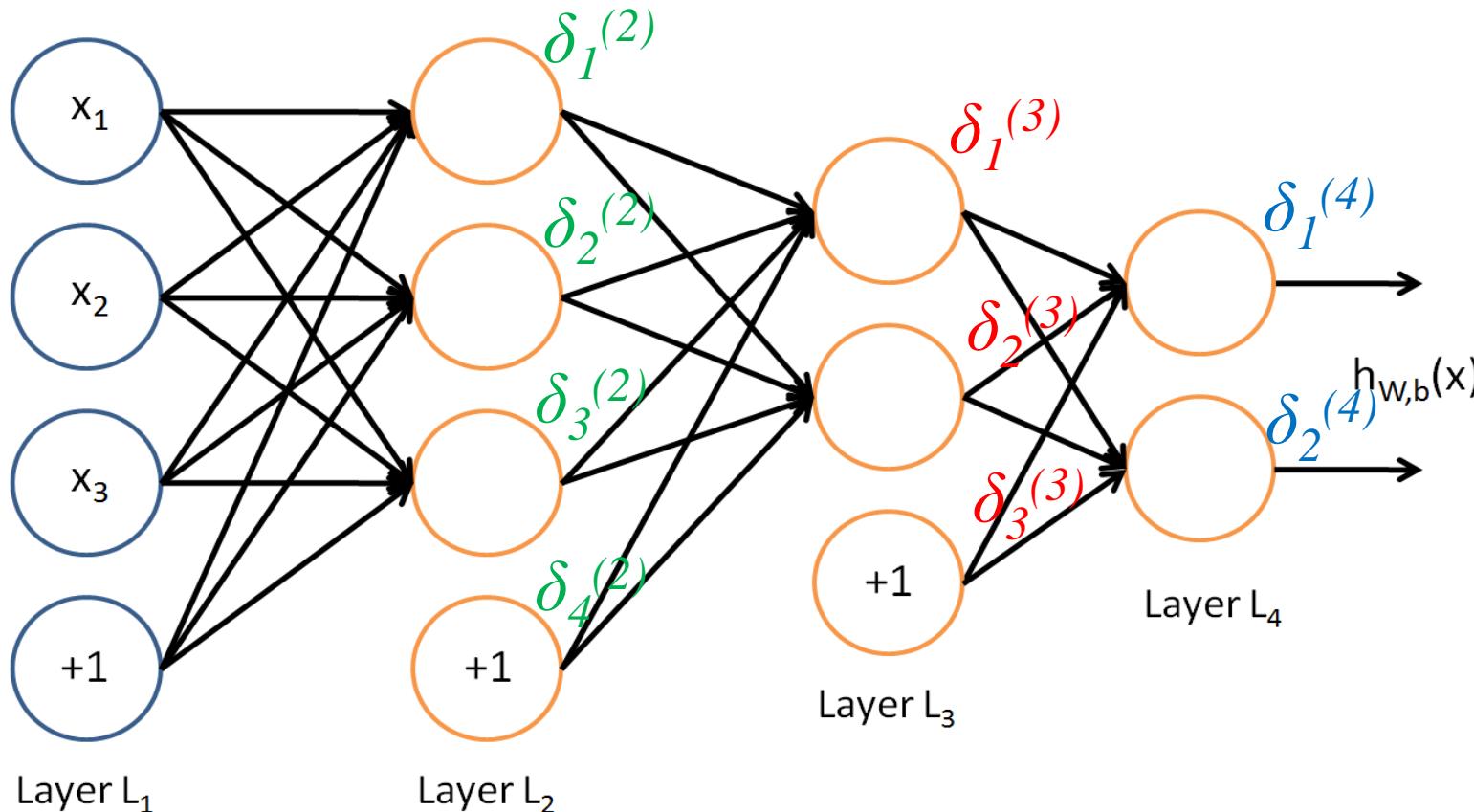
←参数 $W^{(l)}$ 和偏置 $b^{(l)}$ 更新规则

梯度（偏导）的具体计算？

误差反向传播算法 backpropagation algorithm

什么是误差的反向传播? $\delta_i^{(l)}$ 定义为第 l 层第 i 个单元的误差

$$\text{正向} \rightarrow \delta_1^{(4)} = f(W_{11}^{(3)} \delta_1^{(3)} + W_{21}^{(3)} \delta_2^{(3)} + W_{31}^{(3)} \delta_3^{(3)})$$



$$\delta_1^{(3)} = f^{-1}(W_{11}^{(3)} \delta_1^{(4)}) + f^{-1}(W_{23}^{(3)} \delta_2^{(4)}) \quad \leftarrow \text{反向传播}$$

误差反向传播算法 backpropagation algorithm

迭代过程中的参数更新

定义 $\delta_i^{(l)}$ 为第 l 层第 i 个单元的误差

$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}} \frac{1}{2} \|y - h_{W,b}(x)\|^2 = -(y_i - a_i^{(n_l)}) \cdot f'(z_i^{(n_l)}) \quad \leftarrow \text{输出层}$$

$$\delta_i^{(l)} = \left(\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)} \delta_j^{(l+1)} \right) f'(z_i^{(l)}) \quad \leftarrow \text{隐藏层}$$

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W, b; x, y) = a_j^{(l)} \delta_i^{(l+1)}$$

$$\frac{\partial}{\partial b_i^{(l)}} J(W, b; x, y) = \delta_i^{(l+1)}$$

$$\begin{aligned} y &= f(x) \\ dy/dx &= f'(x) \\ \Delta y / \Delta x &\approx f'(x) \\ \Delta y &\approx \Delta x f'(x) \end{aligned}$$

S函数求导较方便

$$\begin{aligned} f(z) &= \frac{1}{1 + \exp(-z)} \\ f'(z) &= f(z)(1 - f(z)) \\ f'(z_i^{(l)}) &= a_i^{(l)}(1 - a_i^{(l)}) \end{aligned}$$

完整的误差反向传播算法

1. Set $\Delta W^{(l)} := 0$, $\Delta b^{(l)} := \mathbf{0}$ (matrix/vector of zeros) for all l .
2. For $i = 1$ to m ,
 1. Use backpropagation to compute $\nabla_{W^{(l)}} J(W, b; x, y)$ and $\nabla_{b^{(l)}} J(W, b; x, y)$.
 2. Set $\Delta W^{(l)} := \Delta W^{(l)} + \nabla_{W^{(l)}} J(W, b; x, y)$.
 3. Set $\Delta b^{(l)} := \Delta b^{(l)} + \nabla_{b^{(l)}} J(W, b; x, y)$.
3. Update the parameters:

$$W^{(l)} = W^{(l)} - \alpha \left[\left(\frac{1}{m} \Delta W^{(l)} \right) + \lambda W^{(l)} \right]$$
$$b^{(l)} = b^{(l)} - \alpha \left[\frac{1}{m} \Delta b^{(l)} \right]$$

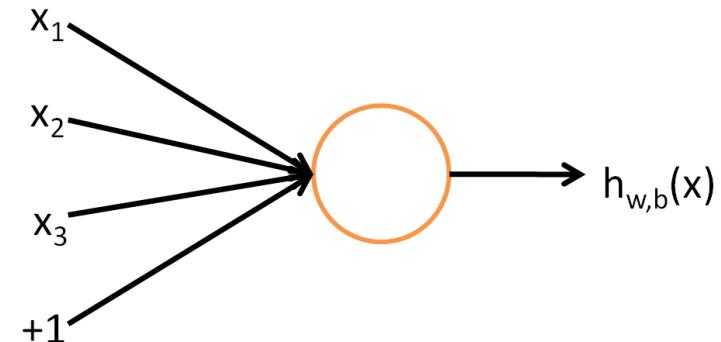
反向传播算法（**backpropagation algorithm**）是一种神经网络学习算法。后向传播算法在多层前馈神经网络上学习。**向前传播输入**: 首先，训练元组提供给网络的输入层。计算隐藏层和输出层每个单元的净输入和输出。**向后传播误差**: 通过更新权重和反映网络预测误差的偏倚，向后传播误差。

不同类型的人工神经网络模型

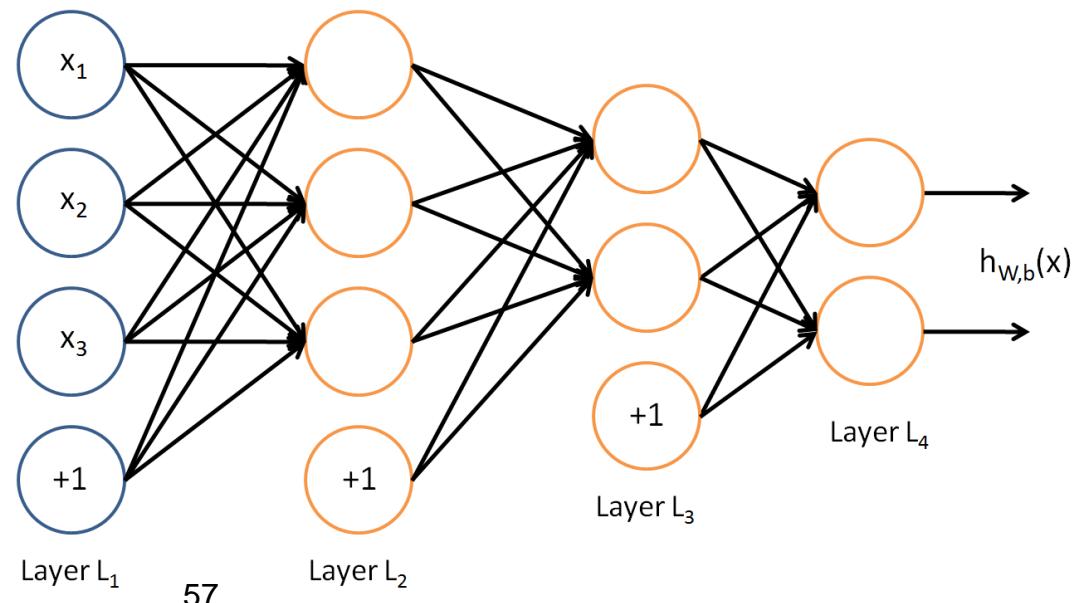
- 感知器神经网络
- 线性神经网络
- BP神经网络
- 径向基函数网络
 - 隐层节点通过基函数执行一种非线性变化，将输入空间映射到一个新的空间，输出层节点则在该新的空间实现线性加权组合。
- 自组织网络
- 反馈网络
 - 典型代表如Elman网络和Hopfield网络

小结：人工神经网络

- 神经元
 - 输入、输出、激励函数
- 神经网络
 - 训练：获取各层的权重参数 $W^{(l)}$ 和偏置 $b^{(l)}$
 - 误差反向传播算法 backpropagation algorithm



$$\begin{aligned} z^{(l+1)} &= W^{(l)} a^{(l)} + b^{(l)} \\ a^{(l+1)} &= f(z^{(l+1)}) \end{aligned}$$



本讲内容： 支持向量机及机器学习方法

- 支持向量机
 - 二元线性SVM
 - SVM用于非线性分类
- 机器学习方法
 - 人工神经网络 (Artificial Neural Network, ANN)
 - 深度学习 (Deep Learning) 现状
 - 经典的深度学习模型/算法
 - 卷积神经网络 Convolutional Neural Networks (CNN)
 - 自动编码器 AutoEncoder
 - 受限玻尔兹曼机 Restricted Boltzmann Machine, RBM
 - 深度置信网络 (Deep Belief Nets, DBN)

Deep Learning Since 2006

2006年，加拿大多伦多大学教授Geoffrey Hinton和他的学生Ruslan Salakhutdinov在《科学》上发表了一篇文章，开启了深度学习在学术界和工业界的浪潮。

materials are identical for all configurations. The blue bars in Fig. 1 summarize the measured SHG signals. For excitation of the *LC* resonance in Fig. 1A (horizontal incident polarization), we find an SHG signal that is 500 times above the noise level. As expected for SHG, this signal closely scales with the square of the incident power (Fig. 2A). The polarization of the SHG emission is nearly vertical (Fig. 2B). The small angle with respect to the vertical is due to deviations from perfect mirror symmetry of the SRRs (see electron micrographs in Fig. 1). Small detuning of the *LC* resonance toward smaller wavelength (i.e., to 1.3- μm wavelength) reduces the SHG signal strength from 100% to 20%. For excitation of the Mie resonance with vertical incident polarization in Fig. 1D, we find a small signal just above the noise level. For excitation of the Mie resonance with horizontal incident polarization in Fig. 1C, a small but significant SHG emission is found, which is again po-

Reducing the Dimensionality of Data with Neural Networks

G. E. Hinton* and R. R. Salakhutdinov

High-dimensional data can be converted to low-dimensional codes by training a multilayer neural network with a small central layer to reconstruct high-dimensional input vectors. Gradient descent can be used for fine-tuning the weights in such “autoencoder” networks, but this works well only if the initial weights are close to a good solution. We describe an effective way of initializing the weights that allows deep autoencoder networks to learn low-dimensional codes that work much better than principal components analysis as a tool to reduce the dimensionality of data.

Dimensionality reduction facilitates the classification, visualization, communication, and storage of high-dimensional data. A simple and widely used method is principal components analysis (PCA), which

finds the directions of greatest variance in the data set and represents each data point by its coordinates along each of these directions. We describe a nonlinear generalization of PCA that uses an adaptive, multilayer “encoder” network

浅层学习 (Shallow Learning)

深度学习 (Deep Learning)

- 浅层学习是机器学习的第一次浪潮

- 20世纪80年代末期，用于人工神经网络的反向传播算法（也叫Back Propagation算法或者BP算法）的发明，给机器学习带来了希望，掀起了基于统计模型的机器学习热潮。这个热潮一直持续到今天。
- 20世纪90年代，相继提出多种浅层机器学习模型，如SVM、Boosting、最大熵方法等。这些模型的结构一般只有**一层隐层节点**（如SVM、Boosting），**或没有隐层节点**（如Logistic Regression）。

- 深度学习是机器学习的第二次浪潮

- 当前多数分类、回归等学习方法为浅层结构算法，其局限性在于有限样本和计算单元情况下对复杂函数的表示能力有限，针对复杂分类问题其泛化能力受到一定制约。深度学习可通过学习一种深层非线性网络结构，实现复杂函数逼近，表征输入数据分布式表示，并展现了强大的从少数样本集中学习数据集本质特征的能力。
- 区别于传统浅层学习，深度学习：1) 强调了**模型结构的深度**，通常有5层、6层，甚至10多层的隐层节点；2) 明确突出了**特征学习的重要性**，也就是说，通过逐层特征变换，将样本在原空间的特征表示变换到一个新特征空间，从而使分类或预测更加容易。

2013年十大突破性技术之首

The screenshot shows the homepage of MIT Technology Review's website. At the top, there is a navigation bar with links for HOME, MENU, CONNECT, THE LATEST, POPULAR, MOST SHARED, and a user icon. The main title "10 BREAKTHROUGH TECHNOLOGIES 2013" is prominently displayed. Below the title, there are ten cards, each representing a technology:

- Deep Learning**: With massive amounts of computational power, machines can now recognize objects and translate speech in real time. Artificial intelligence is finally getting smart.
- Temporary Social Media**: Messages that quickly self-destruct could enhance the privacy of online communications and make people freer to be spontaneous.
- Prenatal DNA Sequencing**: Reading the DNA of fetuses will be the next frontier of the genomic revolution. But do you really want to know about the genetic problems or musical aptitude of your unborn child?
- Additive Manufacturing**: Skeptical about 3-D printing? GE, the world's largest manufacturer, is on the verge of using the technology to make jet parts.
- Baxter: The Blue-Collar Robot**: Rodney Brooks's newest creation is easy to interact with, but the complex innovations behind the robot show just how hard it is to get along with people.
- Memory Implants**
- Smart Watches**
- Ultra-Efficient Solar Power**
- Big Data from Cheap Phones**
- Supergrids**

MIT Technology Review 杂志, 2013年4月23日

席卷学术届到工业届的浪潮

2012年6月，《纽约时报》披露了Google Brain。项目由斯坦福大学的机器学习教授Andrew Ng和在大规模计算机系统方面的世界顶尖专家Jeff Dean共同主导，用16000个CPU Core的计算平台训练一种称为“深度神经网络”(DNN, Deep Neural Networks)的机器学习模型（有10亿个节点），在语音识别和图像识别等领域获得了巨大的成功。



2012. 6 “谷歌大脑”项目

Scientists See Promise in Deep-Learning Programs



A voice recognition program translated a speech
in Chinese.

By JOHN MARKOFF
Published: November 23, 2012

The New York Times

2012. 11 微软智能同声传译

2012年11月，微软在中国天津的一次活动上公开演示了一个全自动的同声传译系统，演讲者用英文演讲，后台的计算机一气呵成自动完成语音识别、英中机器翻译和中文语音合成，效果非常流畅。

图像识别领域的突破



ImageNet Challenge

<http://www.image-net.org/> Imagenet Large Scale Visual Recognition Challenge (ILSVRC)

This challenge evaluates algorithms for object detection and image classification at large scale.

- 2014:
1. A PASCAL-style detection challenge on fully labeled data for 200 categories of objects.
 2. An image classification plus object localization challenge with 1000 categories.

72%, 2010

74%, 2011

85%, 2012

语音识别领域的突破

task	hours of training data	DNN-HMM	GMM-HMM with same data
Switchboard (test set 1)	309	18.5	27.4
Switchboard (test set 2)	309	16.1	23.6
English Broadcast News	50	17.5	18.8
Bing Voice Search (Sentence error rates)	24	30.4	36.2
Google Voice Input	5,870	12.3	
Youtube	1,400	47.6	52.3

错误率降低20%-30%

Slide Courtesy: Geoff Hinton

百度投入基础技术研发

- 2013年初年会上，Robin宣布要成立研究院：

我们会吸引这个领域里全球最顶尖的高手陆续加盟，为我们新一年的产品和业务发展提供最坚实的基础！

我希望百度IDL会成为像AT&T-Bell labs, Xerox PARC这样的顶尖的研究机构，**为中国，为全世界的创新历史再添一笔浓墨重彩！**

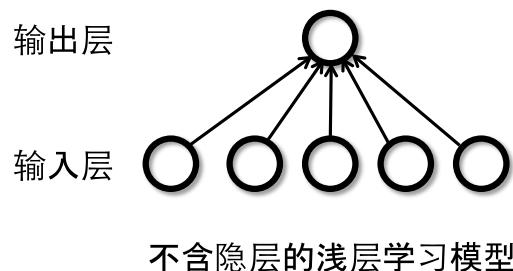
2014年5月17日，吴恩达(Andrew Ng)正式加盟百度，担任百度首席科学家，负责百度研究院，全面领导“百度大脑”、“深度学习”等研究方向。5月18日，百度硅谷人工智能实验室启用。

2015年3月3日，全国政协委员、百度公司CEO李彦宏日前公布了自己的两会提案；李彦宏的提案主要有两条：一、建议全面开放医院挂号号源，让病人找到最合适的医生；二是建议设立“中国大脑”计划，推动人工智能跨越发展，抢占新一轮科技革命制高点。



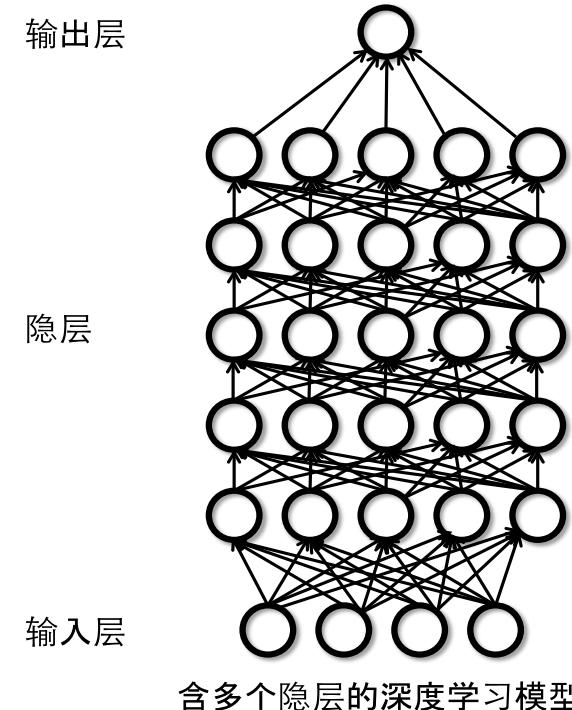
机器学习技术的两次浪潮

deep learning采用了神经网络相似的分层结构，系统由包括输入层、隐层（多层）、输出层组成的多层网络。传统神经网络中，采用的是back propagation的方式进行，然而对于一个deep network（7层以上），残差传播到最前面的层已经变得太小，出现所谓的gradient diffusion（梯度扩散）。



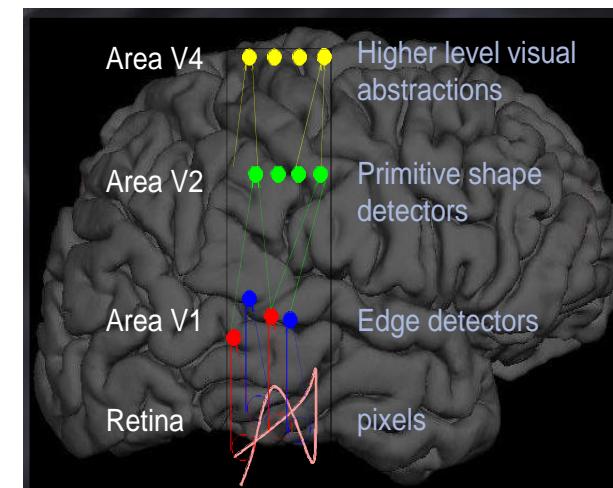
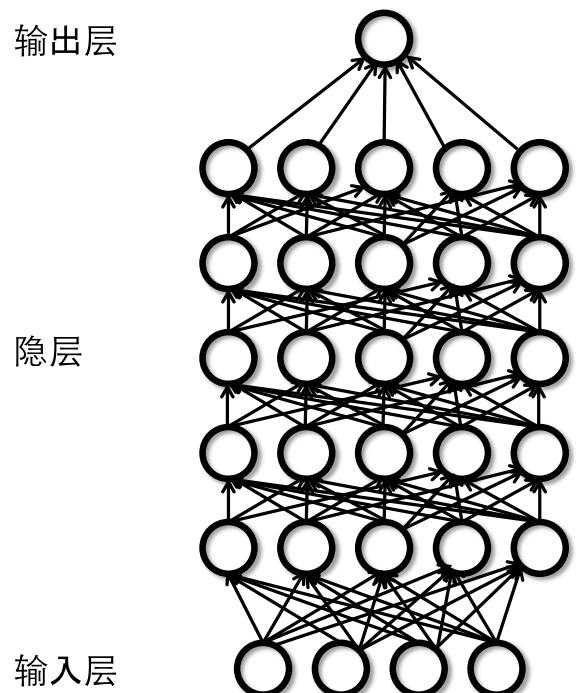
1990 第一次浪潮
浅层学习

←问题的出现，BP不再合适
需要新的训练方法

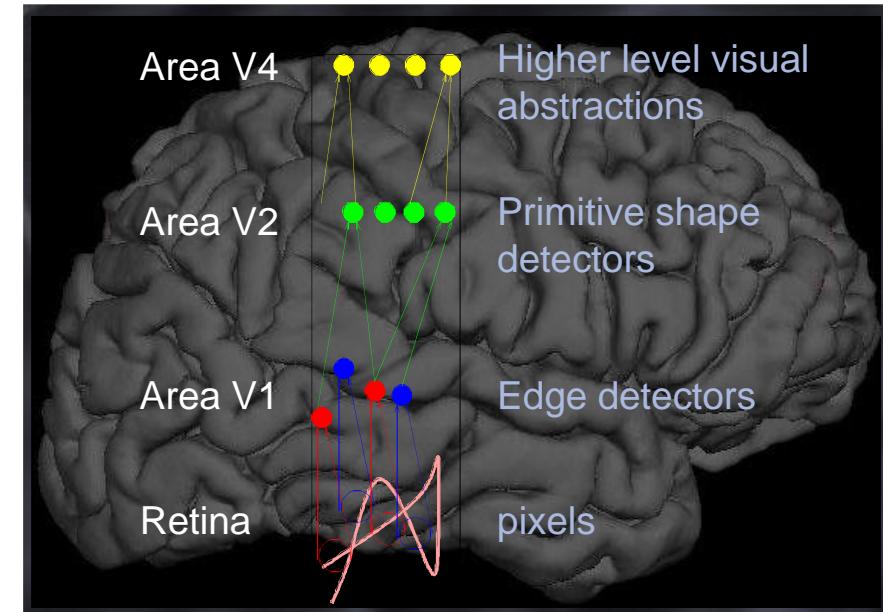
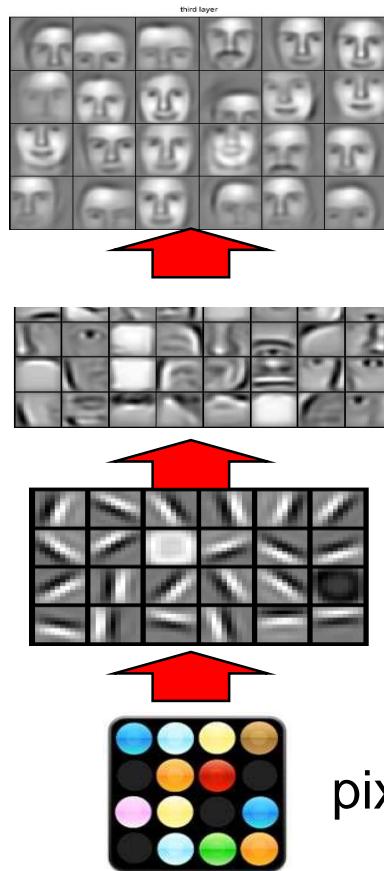


2010 第二次浪潮
深度学习

深度学习机制更接近人脑



深度学习和人类学习的共同点



总的来说，人的视觉系统的信息处理是分级的。从低级的V1区提取边缘特征，再到V2区的形状或者目标的部分等，再到更高层，整个目标、目标的行为等。也就是说高层的特征是低层特征的组合，从低层到高层的特征表示越来越抽象，越来越能表现语义或者意图。而抽象层面越高，存在的可能猜测就越少，就越利于分类。

小结：深度学习

- 浅层学习是机器学习的第一次浪潮
 - Back Propagation算法
- 深度学习是机器学习的第二次浪潮
 - 强调了模型结构的深度
 - 突出了特征学习的重要性
- 常见的深度学习模型/算法
 - 受限波尔兹曼机（Restricted Boltzmann Machine, RBN）
 - 深度置信网络（Deep Belief Networks, DBN）
 - 卷积神经网络（Convolutional Neural Network）
 - 堆栈式自动编码器（Stacked Auto-encoders）

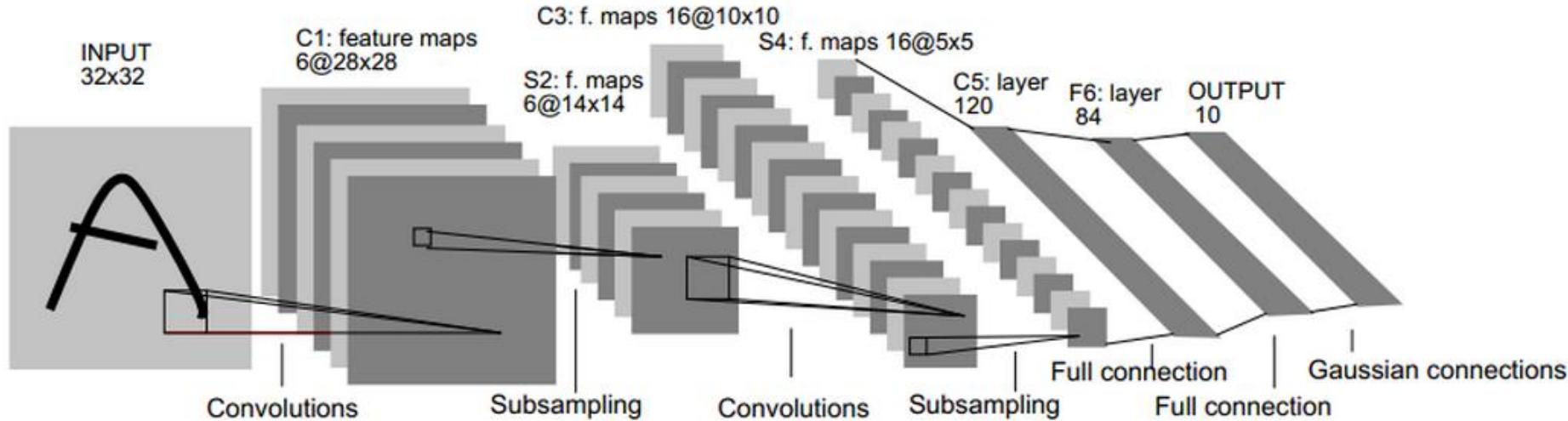
本讲内容： 支持向量机及机器学习方法

- 支持向量机
 - 二元线性SVM
 - SVM用于非线性分类
- 机器学习方法
 - 人工神经网络 (Artificial Neural Network, ANN)
 - 深度学习 (Deep Learning) 现状
 - 经典的深度学习模型/算法
 - 卷积神经网络 Convolutional Neural Networks (CNN)
 - 自动编码器 AutoEncoder
 - 受限玻尔兹曼机 Restricted Boltzmann Machine, RBM
 - 深度置信网络 (Deep Belief Nets, DBN)

卷积神经网络

Convolutional Neural Networks (CNN)

A Convolutional Neural Network (CNN) is comprised of one or more **convolutional layers** (often with a **subsampling** step) and then followed by one or more fully connected layers as in a standard multilayer neural network.



LeNet-5系统结构：用于文字识别的7层卷积网络

20世纪60年代，Hubel和Wiesel在研究猫脑皮层中用于**局部敏感**和**方向选择**的神经元时发现其独特的网络结构可以有效地降低反馈神经网络的复杂性，继而提出了**CNN**。

Convolutional Neural Networks (CNN)

卷积 Convolutions

此处的卷积：来自上层局部感受野数据的加权和

1	1	1	1	0	0
0	1	1	1	1	0
0	0	1	1	1	1
0	0	1	1	1	0
0	1	1	0	0	0

Image

4		

Convolved Feature

1	1	1	0	0	0
0	1	1	1	1	0
0	0	1	1	1	1
0	0	1	1	1	0
0	1	1	0	0	0

Image

4	3	4
2		

Convolved Feature

1	1	1	0	0	0
0	1	1	1	1	0
0	0	1	1	1	1
0	0	1	1	1	0
0	1	1	0	0	0

Image

4	3	4
2	4	3
2		

Convolved Feature

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image

4	3	

Convolved Feature

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image

4	3	4
2	4	3

Convolved Feature

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image

4	3	4
2	4	3
2		

Convolved Feature

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image

4	3	4
2	4	3
2		

Convolved Feature

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image

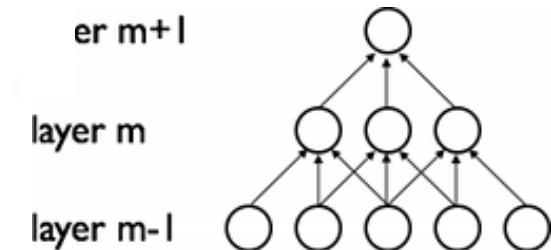
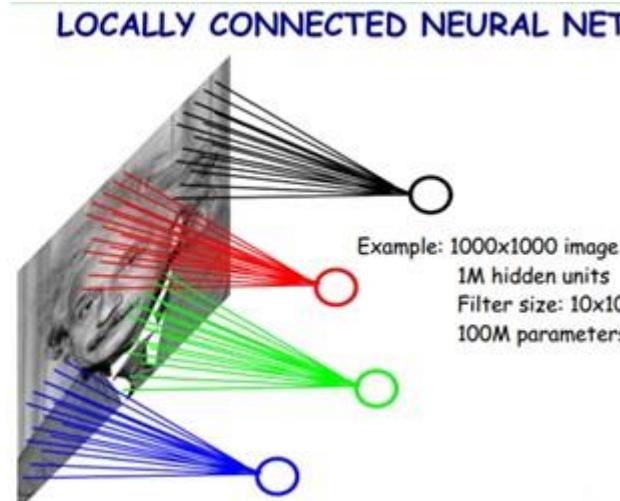
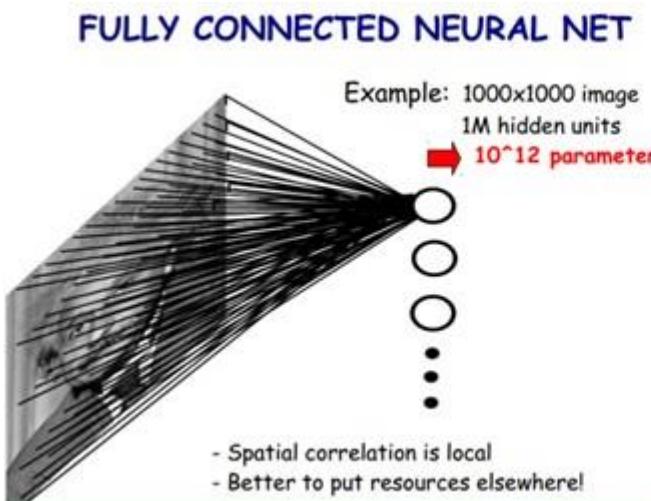
4	3	4
2	4	3
2	3	4

Convolved Feature

Convolutional Neural Networks (CNN)

局部感受野 local receptive fields

Each unit in a layer receives inputs from a set of units located in a small neighborhood in the previous layer. The idea of connecting units to local receptive fields on the input goes back to the Perceptron in the early 60s, and was almost simultaneous with Hubel and Wiesel discovery of locally-sensitive, orientation selective neurons in the cat's visual system.

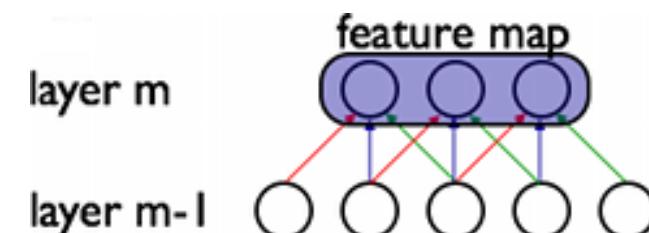
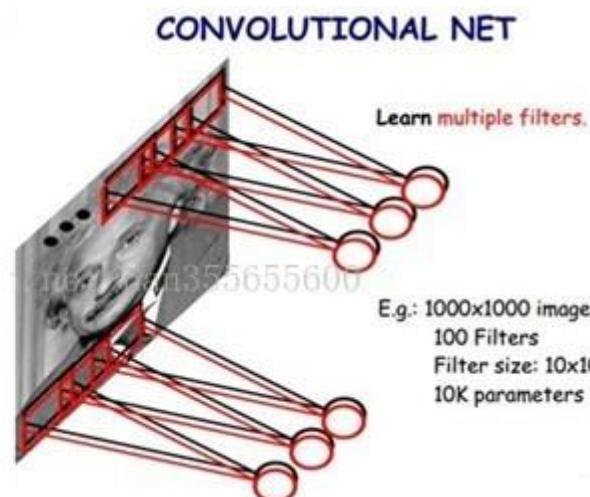
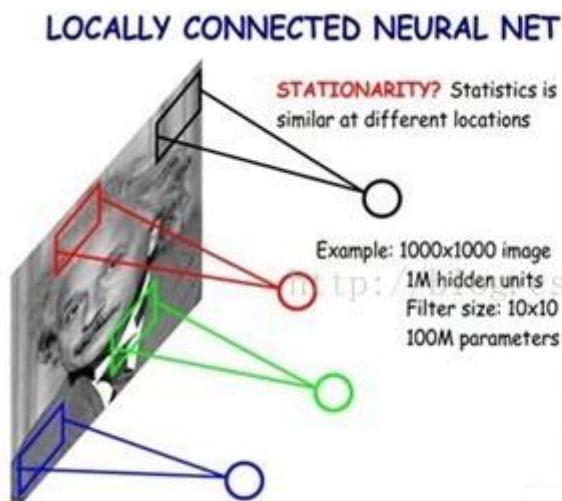


Sparse Connectivity

Convolutional Neural Networks (CNN)

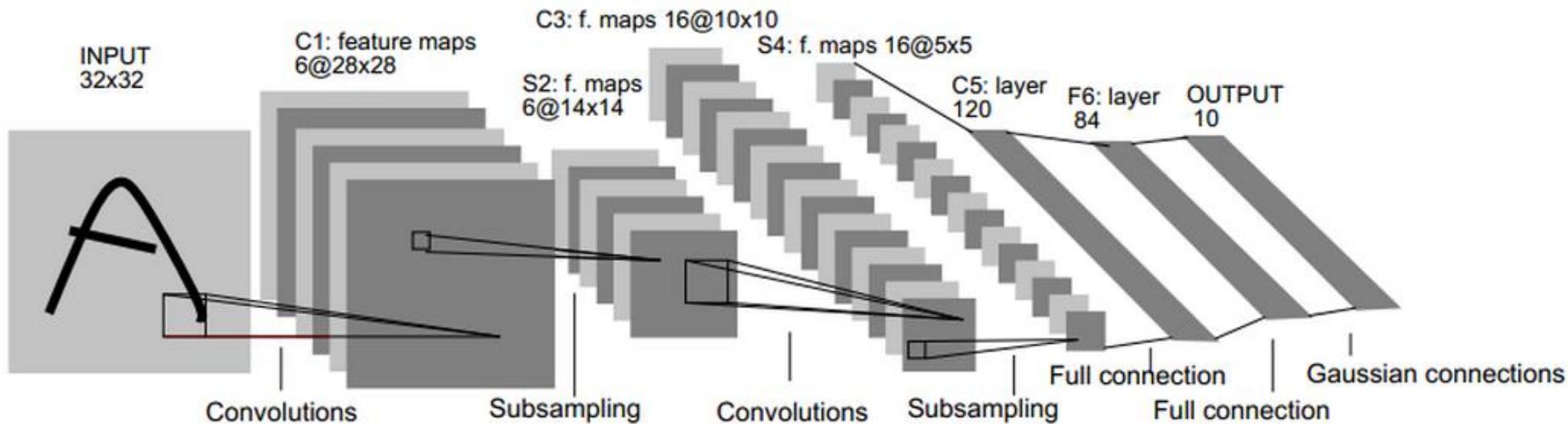
权值共享 Shared Weights

Natural images have the property of being “stationary”, meaning that the statistics of one part of the image are the same as any other part. This suggests that the features that we learn at one part of the image can also be applied to other parts of the image, and we can use the same features at all locations.



Convolutional Neural Networks (CNN)

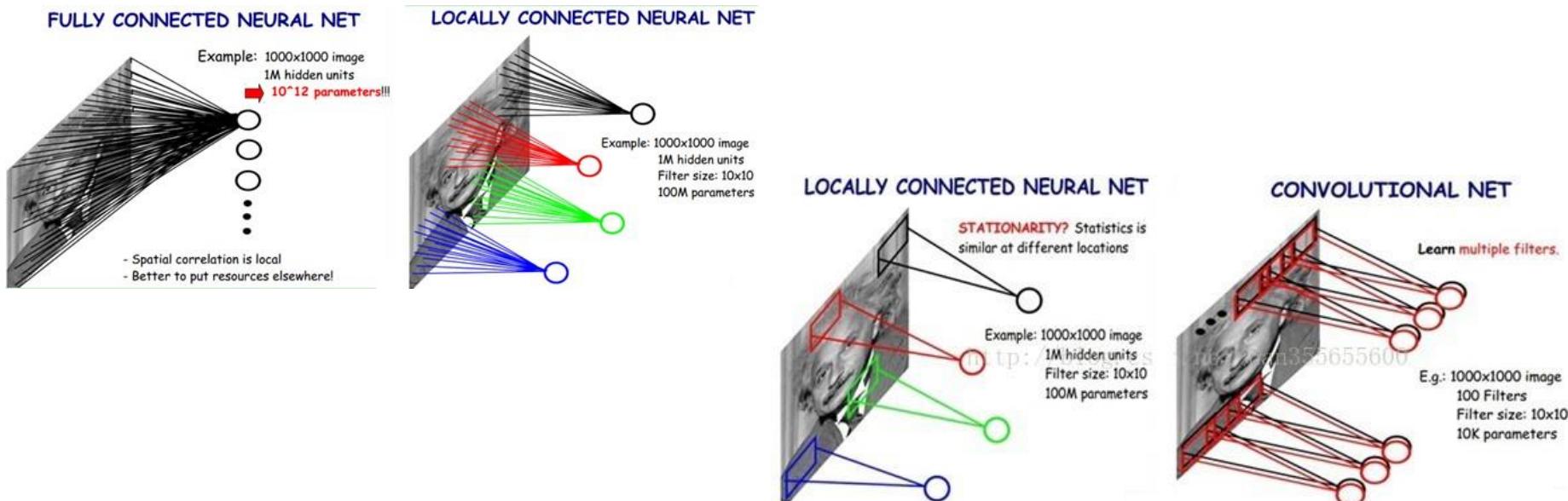
LeNet-5文字识别系统的卷积神经网络



- C1层是一个卷积层，由6个特征图Feature Map构成（每个特征图对应一种卷积核）。特征图中每个神经元与输入中 $5*5$ 的邻域相连。
- S2层是一个下采样层，有6个 $14*14$ 的特征图。特征图中的每个单元与C1中的 $2*2$ 邻域相连接。
- C3层也是一个卷积层，它同样通过 $5*5$ 的卷积核去卷积层S2，然后得到的特征map就只有 $10x10$ 个神经元，但是它有16种不同的卷积核，所以就存在16个特征map了。
- S4层是一个下采样层，由16个 $5*5$ 大小的特征图构成。每个单元与C3中 $2*2$ 邻域相连接。
- C5层是一个卷积层，有120个特征图。
- F6层计算输入向量和权重向量之间的点积，再加上一个偏置。
- 输出层由欧式径向基函数（Euclidean Radial Basis Function）单元组成，每个单元代表一个类别（如果要识别0-9数字的话，需要10个节点）。

小结： Convolutional Neural Networks (CNN)

- **Convolution Layers** 卷积层
- **Sub-sampling Layers** 子采样层
- **Local Receptive Rields** 局部感受野
- **Shared Weights** 权值共享

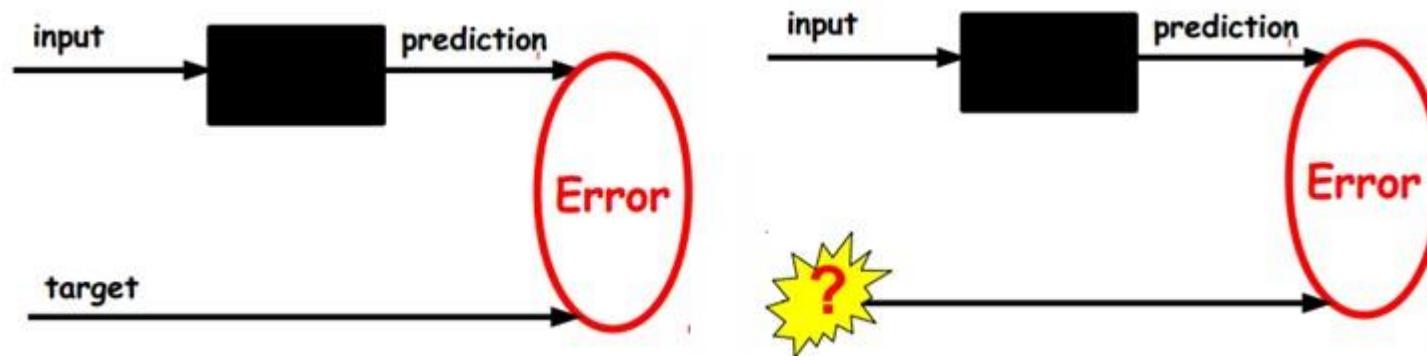


本讲内容： 支持向量机及机器学习方法

- 支持向量机
 - 二元线性SVM
 - SVM用于非线性分类
- 机器学习方法
 - 人工神经网络 (Artificial Neural Network, ANN)
 - 深度学习 (Deep Learning) 现状
 - 经典的深度学习模型/算法
 - 卷积神经网络 Convolutional Neural Networks (CNN)
 - 自动编码器 AutoEncoder
 - 受限玻尔兹曼机 Restricted Boltzmann Machine, RBM
 - 深度置信网络 (Deep Belief Nets, DBN)

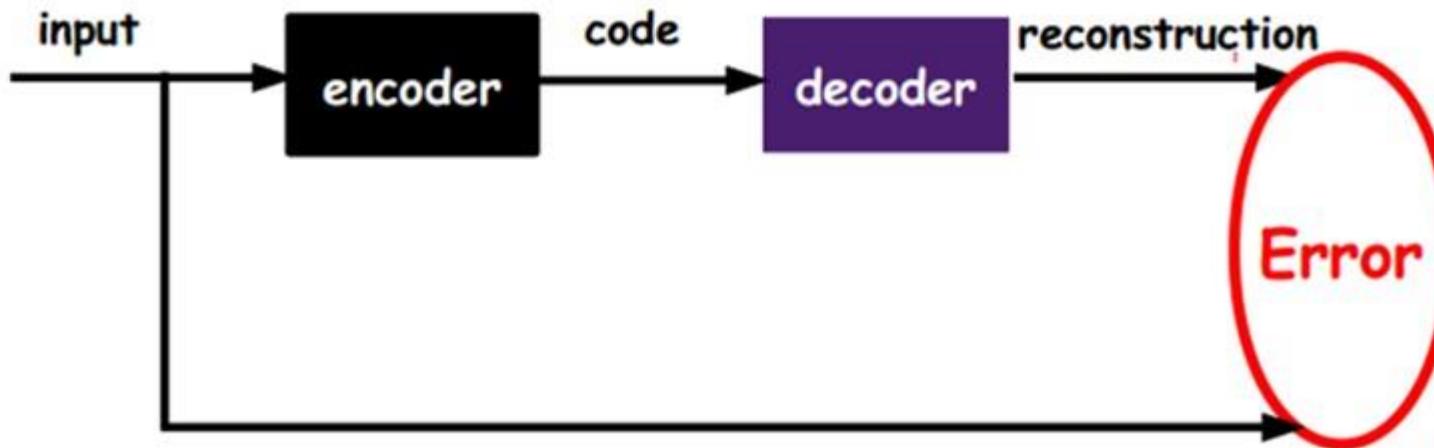
自动编码器 AutoEncoder

如果给定一个神经网络，假设其输出与输入是相同的，然后训练调整其参数，得到每一层中的权重。自然地，我们就得到了输入的几种不同表示（每一层代表一种表示），这些表示就是特征。自动编码器就是一种尽可能复现输入信号的神经网络。



如左图，输入的样本是有标签的，即 (**input, target**)，这样我们根据当前输出和**target (label)**之间的差去改变前面各层的参数，直到收敛。但现在我们只有无标签数据，也就是右边的图。那么这个误差怎么得到呢？

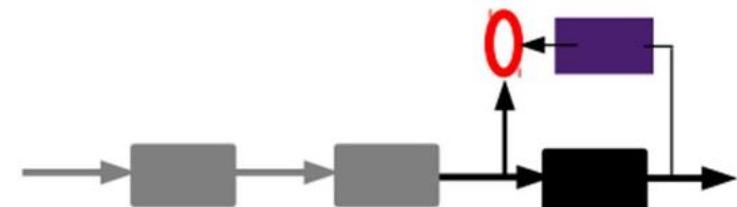
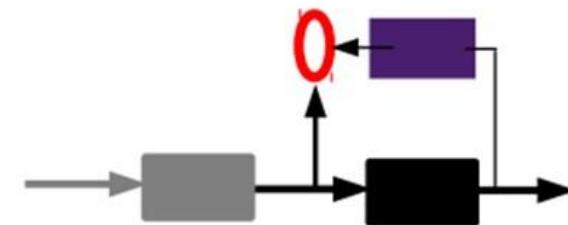
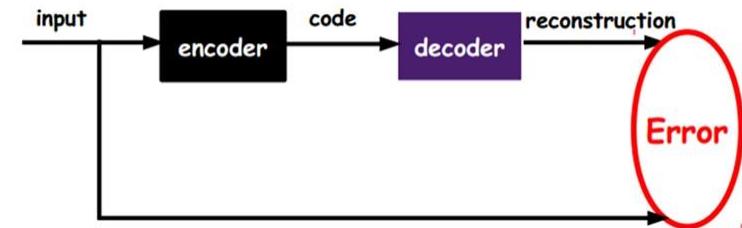
自动编码器 AutoEncoder 如何训练一层内的参数？



如上图，我们将input输入一个encoder编码器，就会得到一个code，这个code也就是输入的一个表示，那么我们怎么知道这个code表示的就是input呢？我们加一个decoder解码器，这时候decoder就会输出一个信息，那么如果输出的这个信息和一开始的输入信号input是很像的（理想情况下就是一样的），那很明显，我们就有理由相信这个code是靠谱的。所以，我们就通过调整encoder和decoder的参数，使得重构误差最小，这时候我们就得到了输入input信号的第一个表示了，也就是编码code了。因为是无标签数据，所以误差的来源就是直接重构后与原输入相比得到。

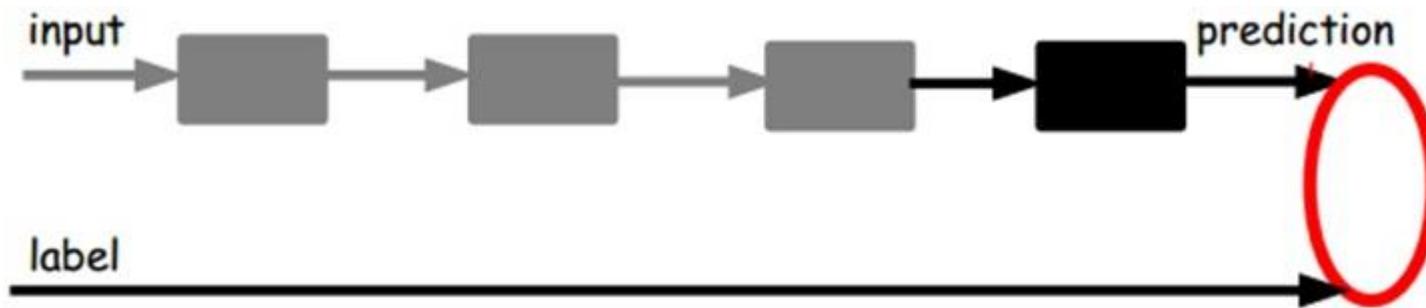
自动编码器 AutoEncoder 如何训练多个层的参数？

那第二层和第一层的训练方式就没有差别了，我们将第一层输出的code当成第二层的输入信号，同样最小化重构误差，就会得到第二层的参数，并且得到第二层输入的code，也就是原输入信息的第二个表达了。其他层就同样的方法炮制（训练这一层，前面层的参数都是固定的，并且他们的decoder已经没用了，都不需要了）。



自动编码器 AutoEncoder 如何应用到分类系统中？

完成各层参数的训练后，这个AutoEncoder还不能用来分类数据，因为它还没有学习如何去连结一个输入和一个类。它只是学会了如何去重构或者复现它的输入而已。或者说，它只是学习获得了一个可以良好代表输入的特征，这个特征可以最大程度上代表原输入信号。



为了实现分类，可以在AutoEncoder的最顶的编码层添加一个分类器（如SVM），然后通过标准的多层神经网络的监督训练方法（梯度下降法）去训练。一旦监督训练完成，这个网络就可以用来分类了。神经网络的最顶层可以作为一个线性分类器。

小结： AutoEncoder

- **AutoEncoder**
 - AutoEncoder是一种尽可能复现输入信号的ANN
 - 在AutoEncoder的最顶的编码层添加一个分类器可以使之用于分类问题
- **AutoEncoder的变体**
 - Sparse AutoEncoder（稀疏自动编码器）：约束每一层中的节点中大部分都要为0，只有少数不为0。稀疏的表达往往比其他的表达要有效（人脑好像也是这样的，某个输入只是刺激某些神经元，其他的大部分的神经元是受到抑制的）。
 - Denoising AutoEncoders（降噪自动编码器）：训练数据加入噪声，所以自动编码器必须学习去去除这种噪声而获得真正的没有被噪声污染过的输入。

本讲内容： 支持向量机及机器学习方法

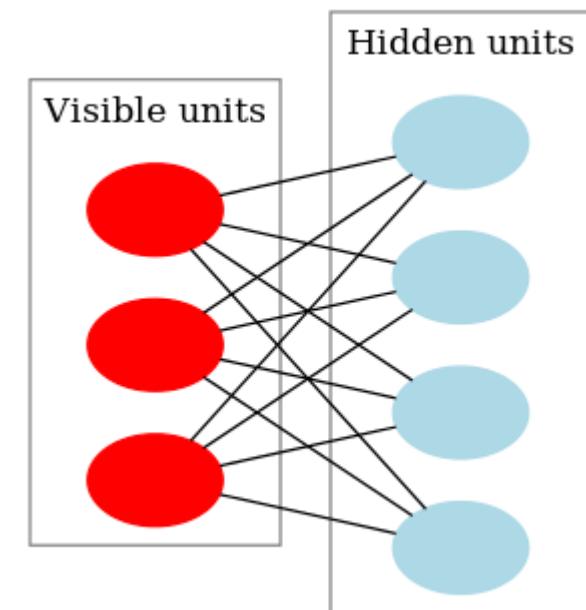
- 支持向量机
 - 二元线性SVM
 - SVM用于非线性分类
- 机器学习方法
 - 人工神经网络 (Artificial Neural Network, ANN)
 - 深度学习 (Deep Learning) 现状
 - 经典的深度学习模型/算法
 - 卷积神经网络 Convolutional Neural Networks (CNN)
 - 自动编码器 AutoEncoder
 - 受限玻尔兹曼机 Restricted Boltzmann Machine, RBM
 - 深度置信网络 (Deep Belief Nets, DBN)

受限玻尔兹曼机

Restricted Boltzmann Machine, RBM

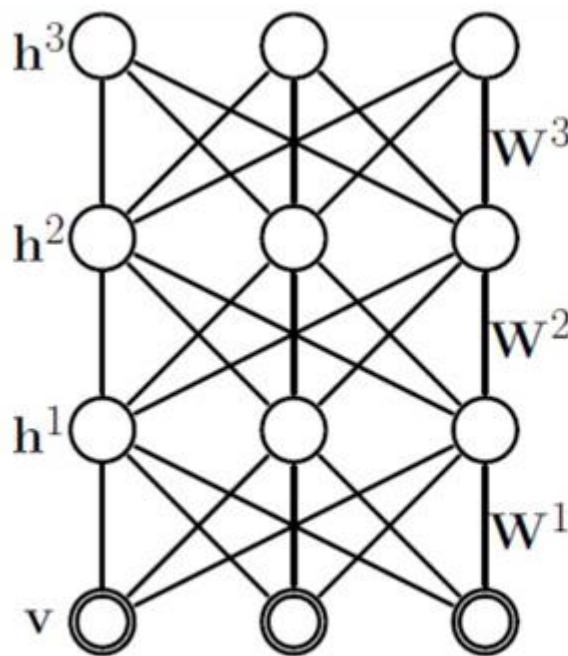
受限玻尔兹曼机（restricted Boltzmann machine, RBM）是一种可通过输入数据集学习概率分布的随机生成神经网络。受限玻尔兹曼机在降维、分类、协同过滤、特征学习和主题建模中得到了应用。根据任务的不同，受限玻尔兹曼机可以使用监督学习或无监督学习的方法进行训练。

经典的神经网络模型是一种确定的结构，而玻尔兹曼网络是一种**随机网络**。**受限玻尔兹曼机**是一类具有**两层结构**、对称链接无自反馈的随机神经网络模型，**层与层之间是全连接，层内无链接**，也就是说是一个**二部图**。由多个RBM结构堆叠而成的深度信念网络能提取出更好更抽象的特征，从而用来分类。

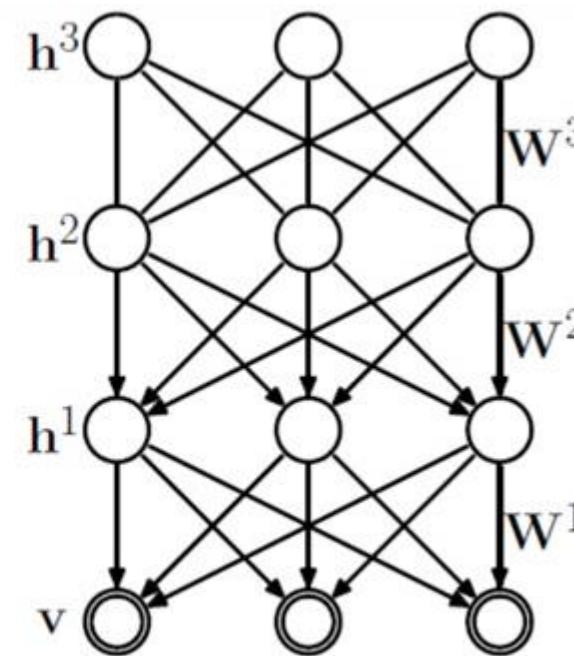


Restricted Boltzmann Machine, RBM 深度置信网络 (Deep Belief Nets, DBN)

把隐藏层的层数增加，可以得到Deep Boltzmann Machine (DBM)；如果在靠近可视层的部分使用贝叶斯置信网络（即有向图模型，当然这里依然限制层中节点之间没有链接），而在最远离可视层的部分使用 Restricted Boltzmann Machine，我们可以得到Deep Belief Net (DBN)。



Deep Boltzmann Machine



Deep Belief Network

小结：经典的深度学习模型/算法

- **Convolutional Neural Networks (CNN)**
 - Convolution、Sub-sampling
 - Local Receptive Fields、Shared Weights
- **AutoEncoder**
 - 是一种尽可能复现输入信号的ANN
 - 在最顶的编码层添加一个分类器可以使之用于分类问题
- **Restricted Boltzmann Machine (RBM)**
 - 两层结构、无自反馈的随机神经网络模型
 - 层与层之间是全连接，层内无链接
- **Deep Belief Nets (DBN)**
 - RBM + 贝叶斯置信网络

本讲内容小结：支持向量机及机器学习方法

- 支持向量机
 - 二元线性SVM
 - SVM用于非线性分类
- 机器学习方法
 - 人工神经网络 (Artificial Neural Network, ANN)
 - 深度学习 (Deep Learning) 现状
 - 经典的深度学习模型/算法
 - 卷积神经网络 Convolutional Neural Networks (CNN)
 - 自动编码器 AutoEncoder
 - 受限玻尔兹曼机 Restricted Boltzmann Machine, RBM
 - 深度置信网络 (Deep Belief Nets, DBN)

谢谢大家!