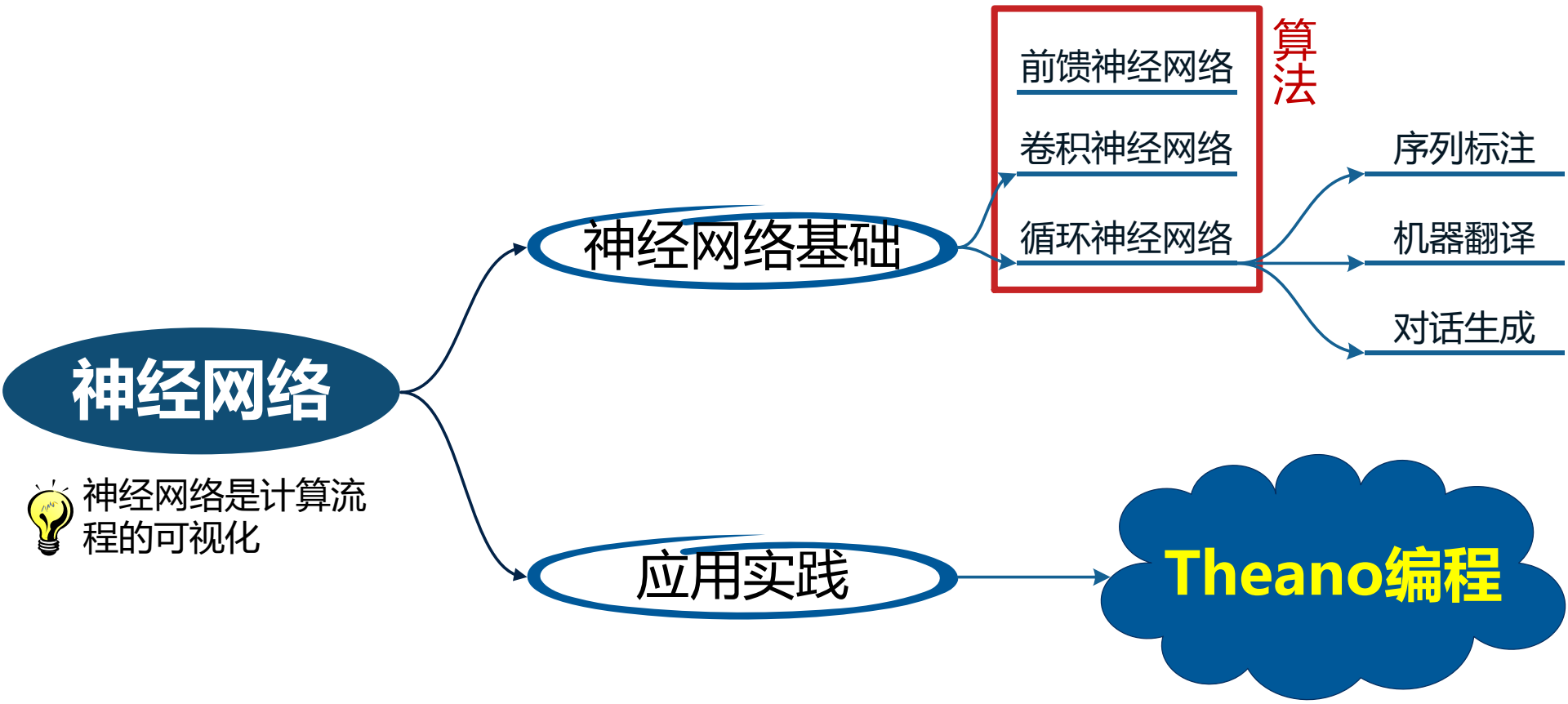
课程专题三：神经网络方法



提纲

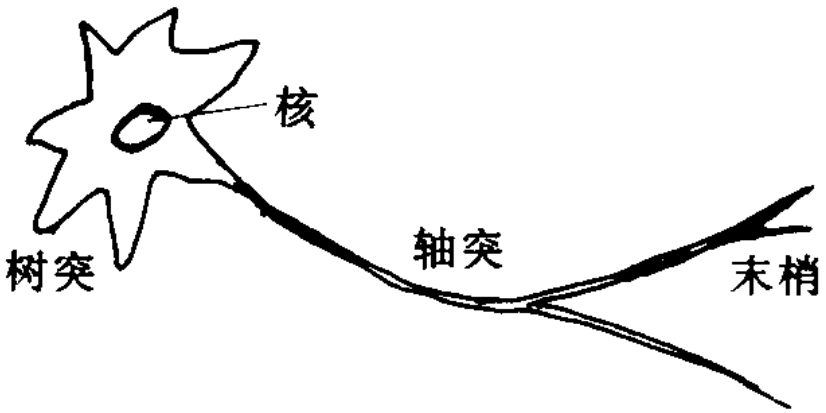
- **神经网络模型结构和物理意义**
- 全连接前馈神经网络
  - 目标函数
  - 优化算法
  - 预测
- 神经网络学习实用技巧

人工神经网络概述

- 什么是人工神经网络？
- T.Koholen的定义：“人工神经网络是由具有适应性的简单单元组成的广泛并行互连的网络，它的组织能够模拟生物神经系统对真实世界物体所作出的交互反应。”

神经元与神经网络

- 大脑可视作为1000多亿神经元组成的神经网络



脑神经信息活动的特征

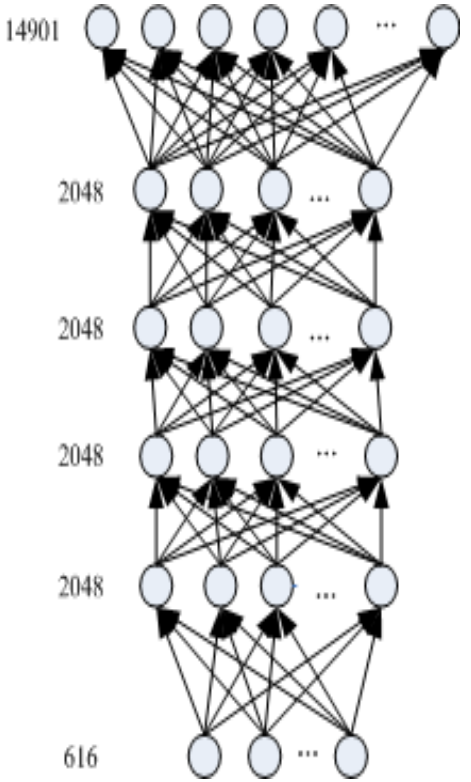
(1)巨量并行性

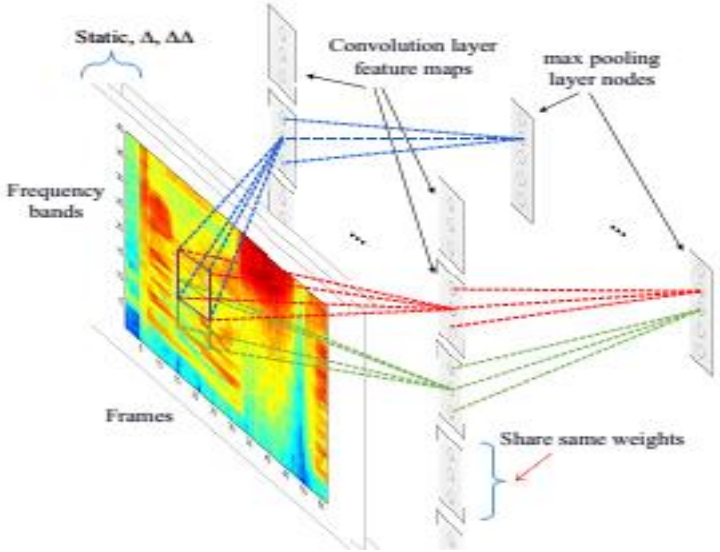
(2)信息处理和存储单元结合在一起

(3)自组织自学习功能

神经网络模型基础

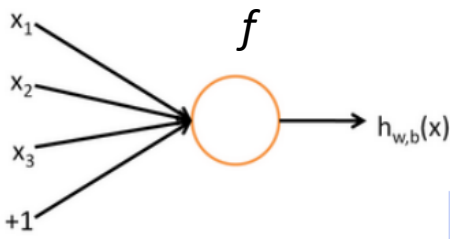
- 神经网络常见种类
  - 全连接网络
  - 卷积神经网络 (Convolution)
  - 循环神经网络 (Recurrent)





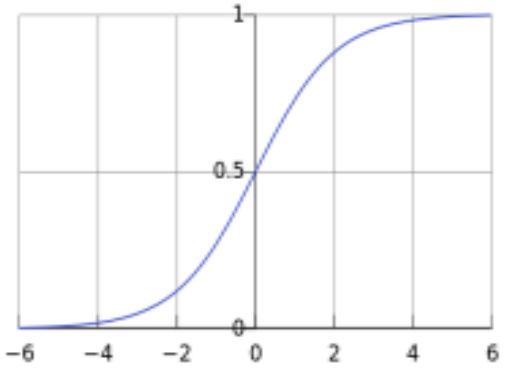
神经网络模型基础

- 神经元（Logistic Regression）

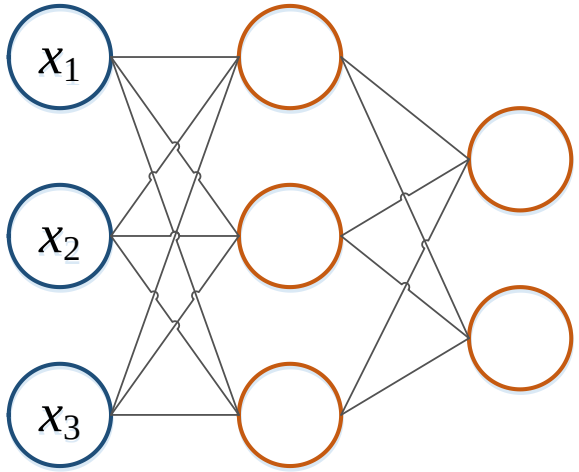


$$h_{w,b}(x) = f(w^{\mathsf{T}}x + b)$$

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



- 一个**全连接**神经网络，**从左到右计算**



输入x　　第1层　　第2层

实际的观点：
神经网络以图形化的方式刻画出从输入到输出的计算过程

提纲

- 神经网络模型结构和物理意义
- 全连接前馈神经网络
  - **目标函数**
  - 优化算法
  - 预测
- 神经网络学习实用技巧

分类：Softmax目标函数

神经网络在最后可以连接具有不同的功能的网络层，产生的具有不同物理意义的输出。如图，输出层为K分类的softmax函数时，

$$p_i = \frac{\exp w_i x}{\sum_{k=1}^{K} \exp w_k x}$$

其中w_k是模型参数，p_i是模型估计的样本x在第i类上的概率。

思考：
与Logistic Regression的2分类问题的联系

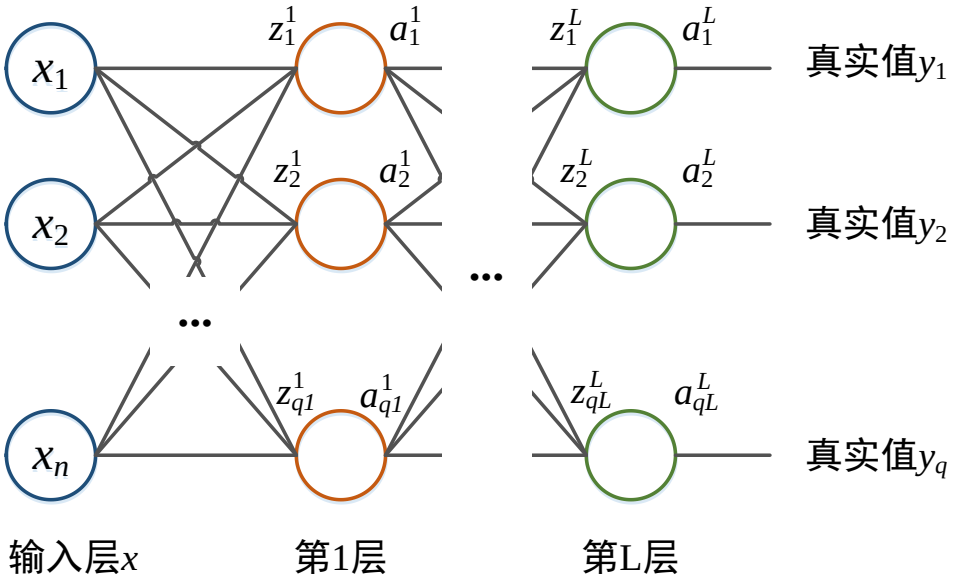
Softmax怎么表示为计算流程的形式

分类： Softmax目标函数

$$J(\boldsymbol{\theta}) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} judge\{y^{(i)} = j\} log \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}\right]$$

- 其中，
  - m = 样本个数
  - k = 类别个数
  - $judge\{x\} = \begin{cases} 1, & x\ is\ True \\ 0, & x\ is\ False \end{cases}$ ，也可写作 $1\{x\}$
  - 没有隐含层

回归：目标函数



我们还可以训练神经网络拟合某个函数。给定一个数据集$D = \{(\boldsymbol{x}, \boldsymbol{y})\}$， $\boldsymbol{x}$为样本的输入值，$\boldsymbol{y}$为样本的真实输出值。若网络对数据$\boldsymbol{x}$的输出值为$\boldsymbol{a}^L(\boldsymbol{x})$，则我们最小化损失函数

$$C = \frac{1}{2|D|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in D} (\boldsymbol{y} - \boldsymbol{a}^L(\boldsymbol{x}))^2$$

也就是最小化总体的平方误差。在不引起误会的前提下，$\boldsymbol{a}^{\boldsymbol{l}}(\boldsymbol{x})$等符号简写作$\boldsymbol{a}^{\boldsymbol{l}}$。

提纲

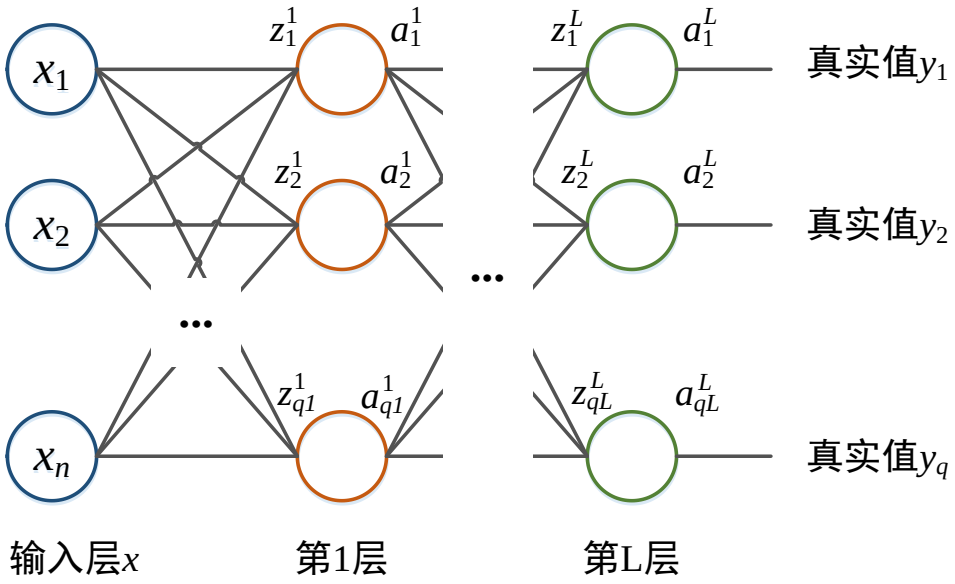
- 神经网络模型结构和物理意义
- 全连接前馈神经网络
  - 目标函数
  - **优化算法**
  - 预测
- 神经网络学习实用技巧

全连接多层网络

- 网络结构
  - 输入层有n个神经元(n维)，第l层有q^l个神经元
  - 除了输入层，网络共有L层
- 变量定义
  - 输入向量 $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$
  - 第l层输入向量 $\boldsymbol{z}^l = (z_1^l, z_2^l, \dots, z_{q^l}^l)$
  - 第l层输出向量 $\boldsymbol{a}^l = (a_1^l, a_2^l, \dots, a_{q^l}^l)$
  - 真实值向量 $\boldsymbol{y} = (y_1, y_2, \dots, y_{q^L})$

核心：
建立矩阵的思维

模型参数



$$a^l = f(z^l)$$

$$W^1 = [\ ]_{q^1 \times n}$$

输入层与第一层之间的权重矩阵，第*j*行对应下一层的第*j*个神经元的输入

$$W^1 \cdot x = z^1$$

$$W^{l+1} = [\ ]_{q^{l+1} \times q^l}$$

第*l*层与第(*l*+1)层之间的权重矩阵，第j行对应下一层的第j个神经元的输入

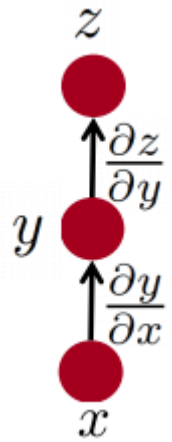
$$W^{l+1} \cdot a^l = z^{l+1}$$

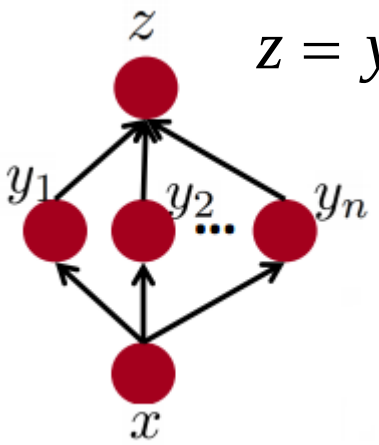
导数法则

- 链式法则

$$z = f(y) \quad y = g(x) \quad \frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$

- 复合函数求导





$$z = y_1 + \cdots + y_n$$

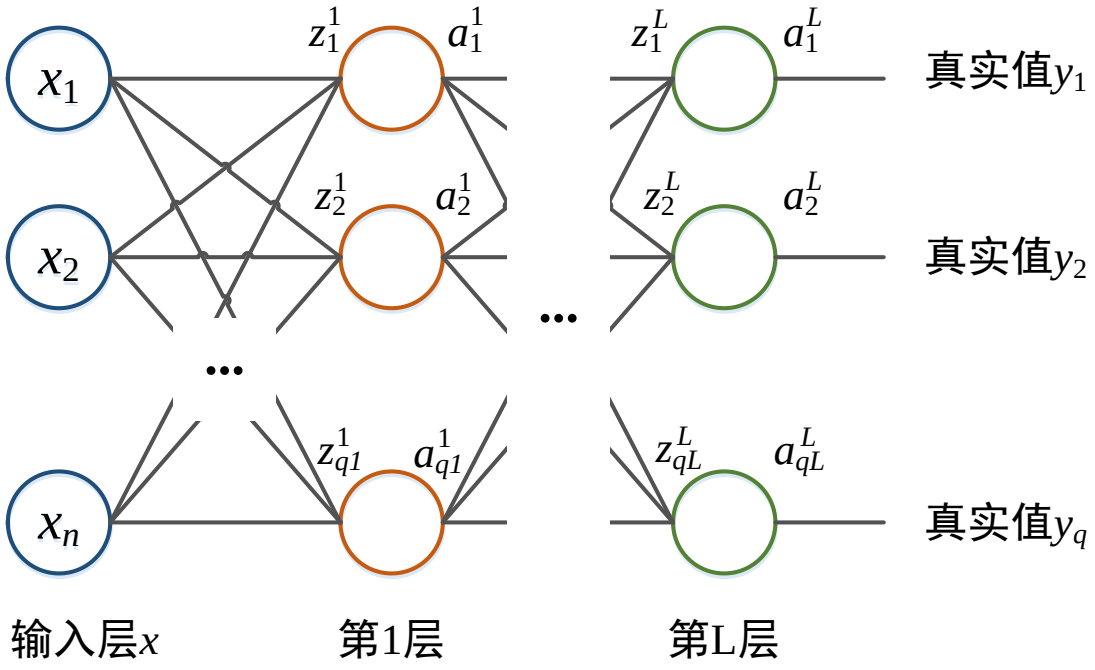
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x} \qquad \frac{\partial z}{\partial x} = \sum_{i=1}^{n} \frac{\partial z}{\partial y_i}\frac{\partial y_i}{\partial x}$$

BP(Back Propagation)训练算法

- 它是有监督训练的前馈多层网络训练算法，是靠调节各层的加权，使网络学会由输入输出组成的训练组
- 其基本方法仍是梯度下降法

反向传播算法

算法的目的：根据真实的输入和输出数据，计算模型的参数（权系数）。从右向左反传误差。



预测：从左到右的计算
训练：从右到左反传误差

BP神经网络模型

- 第l层和第l-1层的连接权值: 矩阵W^l
- 第l层神经元的阈值: 向量$\boldsymbol{b}^l$
- 数据集: D
- 激活函数: $f(\cdot)$
- 误差函数:

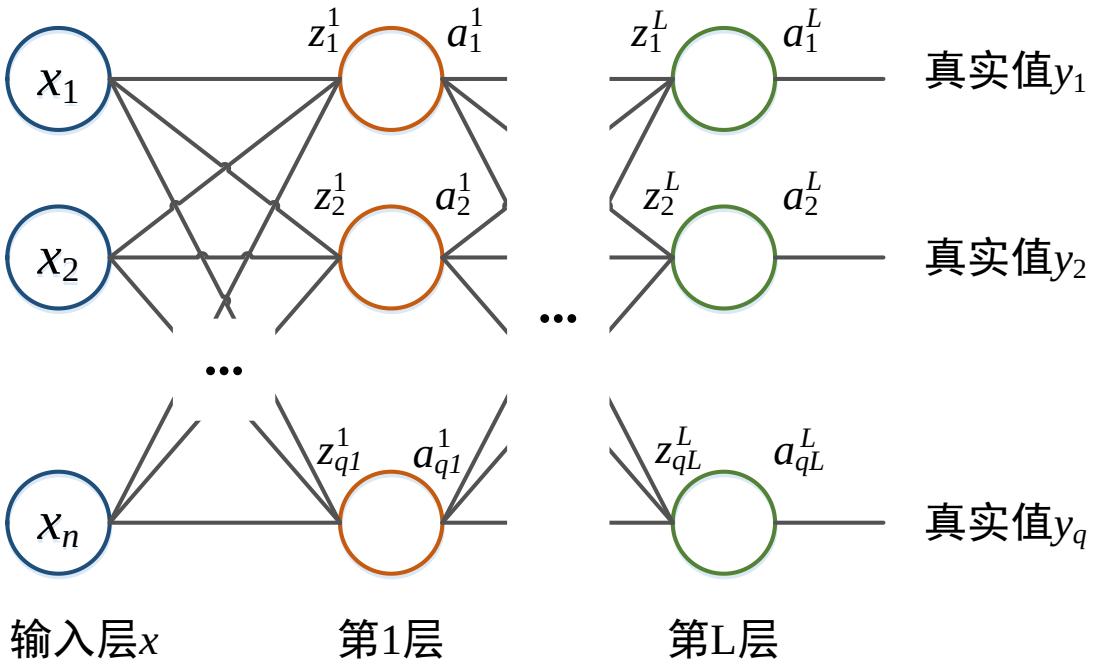
$$C = \frac{1}{2|D|} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in D} (\boldsymbol{y} - \boldsymbol{a}^L(\boldsymbol{x}))^2$$

BP网络训练

- 第一步，网络初始化
  - 给各连接权值分别赋一个随机数，随机初始化方法有很多种，这里我们用较简单的(-0.01, 0.01)内均匀随机数
  - 确定激活函数f的类型
  - 定义损失函数
  - 给定计算精度值ε和最大学习次数
- 第二步，从训练集中抽取一个样本(x, y)

BP网络训练

- 第三步，计算隐含层、输出层各神经元的输入和输出
- 第l层：
  - $\boldsymbol{z}^l = W^l \boldsymbol{a}^{l-1} + \boldsymbol{b}^l$
  - $\boldsymbol{a}^l = f(\boldsymbol{z}^l)$



BP网络训练

- 第四步，利用网络输出值和真实值，计算网络各层误差

- 我们定义：第l层的第j个神经元的误差为$\delta_j^l \equiv \frac{\partial C}{\partial z_j^l}$

目标函数对神经元的输入求导

  - 由$C = \frac{1}{2}\sum(\boldsymbol{y} - \boldsymbol{a}^L)^2$

最末层

  - $\delta_j^L = \frac{\partial C}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L} \cdot \frac{\partial a_j^L}{\partial z_j^L} = (a_j^L - y_j) f'(z_j^l)$

  - 记$\boldsymbol{\delta}^l = \left(\delta_1^l, \delta_2^l, \ldots, \delta_{q^l}^l\right)$，则$\boldsymbol{\delta}^L = (\boldsymbol{a}^L - \boldsymbol{y}) \odot f'(z^L)$，$\odot$代表向量按位乘法

BP网络训练

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} = \sum_k \frac{\partial C}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} = \sum_k \delta_k^{l+1} \frac{\partial z_k^{l+1}}{\partial z_j^l} \qquad (1)式$$

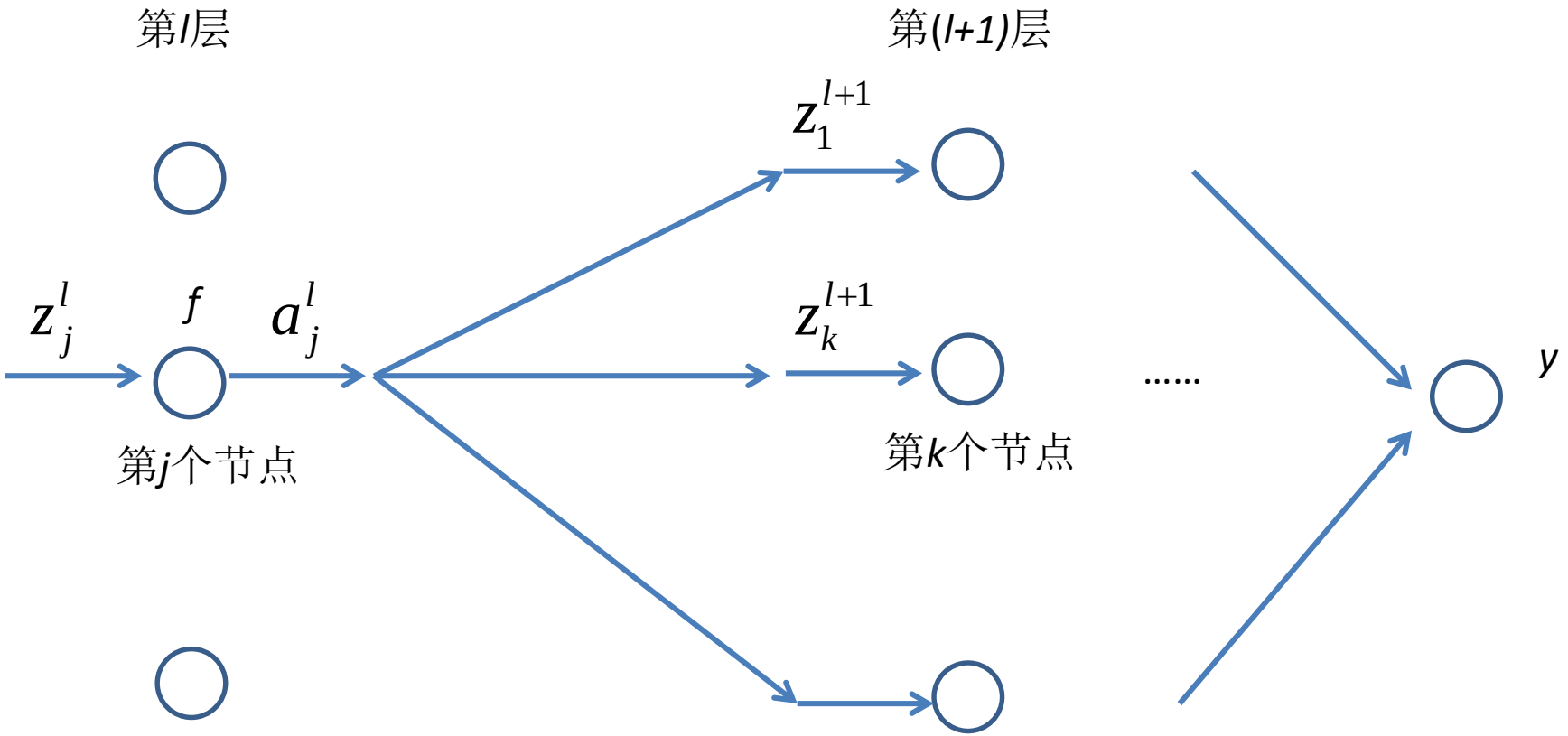
$$z_k^{l+1} = \sum_j w_{kj}^{l+1} a_j^l + b_k^{l+1} = \sum_j w_{kj}^{l+1} f(z_j^l) + b_k^{l+1} \qquad (2)式$$

$$\frac{\partial z_k^{l+1}}{\partial z_j^l} = w_{kj}^{l+1} f'(z_j^l)$$

$$\delta_j^l = \left(\sum_k \delta_k^{l+1} w_{kj}^{l+1} \right) f'(z_j^l)$$

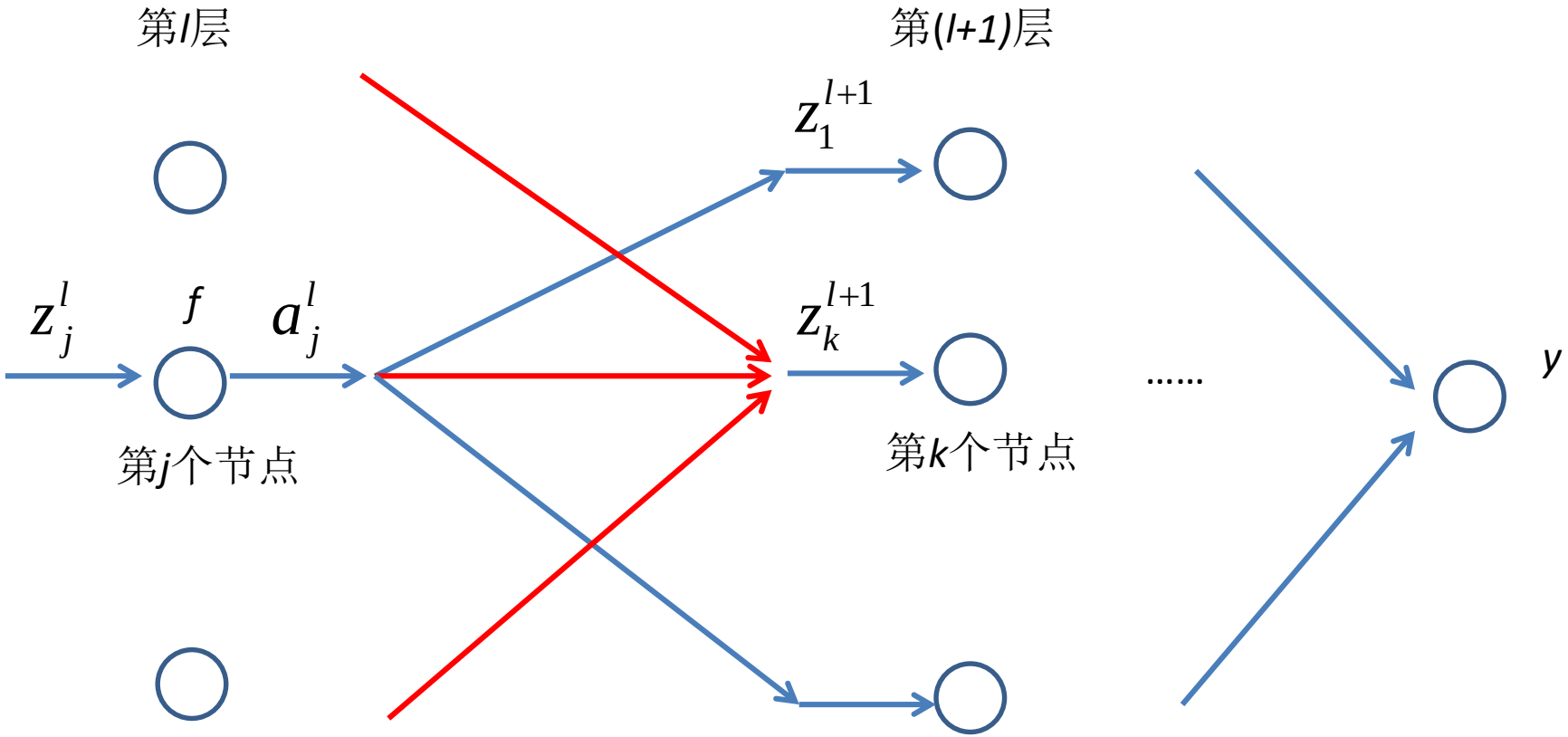
误差反传公式：$\delta^l = ((W^{l+1})^T \cdot \delta^{l+1}) \square f'(z^l)$

BP网络训练



22页中(1)式的图示

BP网络训练



22页中(2)式的图示

BP网络训练

- 第五步，利用计算出的误差δ，导出各层w和b的偏导：
- 提示：$z_j^l = w_{j1}a_1^{l-1} + w_{j2}a_2^{l-1} + \cdots + w_{jn}a_n^{l-1} + b_j^l$

$$\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial b_j^l} = \delta_j^l$$

$$\frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{jk}^l} = \delta_j^l a_k^{l-1}$$

BP网络训练

- 第六步，更新各层w和b：

$$b^{new} = b - \eta \frac{\partial C}{\partial b}$$

$$w^{new} = w - \eta \frac{\partial C}{\partial w}$$

- 其中η是学习率，一般取较小，如0.1

BP网络训练

第八步，判断训练误差是否满足要求。

- 当误差达到预设精度或学习次数大于设定的最大次数，则结束算法。
- 否则，抽取下一个样本，返回到第三步继续训练。（随机梯度下降的方法）

注意：深度学习平台已经把梯度的计算自动化了；只要提供目标函数，梯度会自动计算

提纲

- 神经网络模型结构和物理意义
- 全连接前馈神经网络
  - 目标函数
  - 优化算法
  - **预测**
- 神经网络学习实用技巧

提纲

- 神经网络模型结构和物理意义
- 全连接前馈神经网络
  - 目标函数
  - 优化算法
  - 预测
- **神经网络学习实用技巧**

神经网络训练实用技巧

- 数据块的选择
- 数据随机性
- 数据规整
- 激励函数
- 正则化
- 惯性系数
- 学习率调节

实用训练技巧

- 数据块的选择
  - Batch training
    - 可并行
    - 但是每次要遍历整个数据集，效率低
  - SGD
    - 从单样本点进行的梯度估计是整体训练集梯度的无偏估计
    - 更随机，可以跳出局部最优点
  - miniBatch
    - 折中方案
    - 兼顾了随机跳跃和对梯度估计稳定的双方优势

实用训练技巧

- 数据规整
  - 特征规整

$$Mean = \frac{1}{T}\sum_{t=0}^{T} O(t), \quad Var = \sqrt{\frac{1}{T-1}\sum_{t=0}^{T} O^2(t) - Mean^2}$$

$$O'(t) = \frac{O(t) - Mean}{Var}$$

  - Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

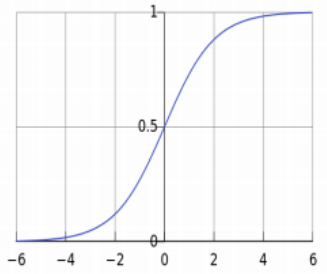
Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

实用训练技巧

- 激励函数　　局部可导函数

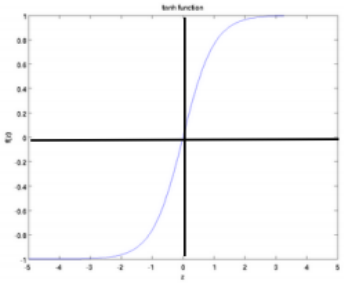
logistic ("sigmoid")

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

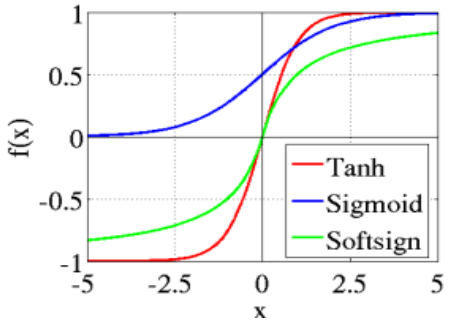


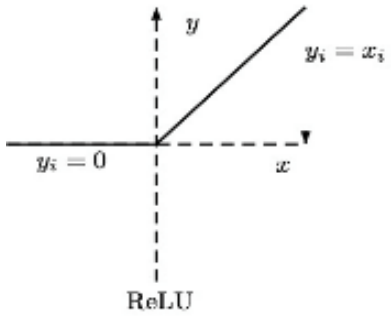
tanh

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

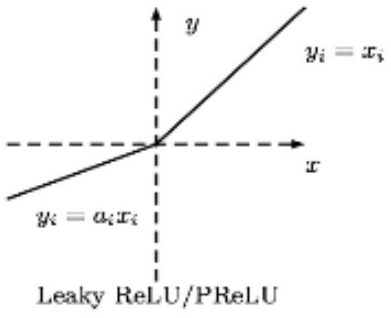


$$\text{softsign}(z) = \frac{a}{1+|a|}$$

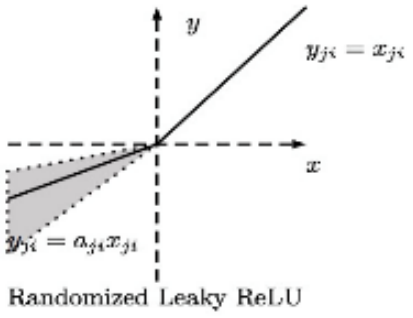




ReLU



Leaky ReLU/PReLU



Randomized Leaky ReLU

实用训练技巧

- 在目标函数中加入参数正则化项（防止过拟合）
  - 过拟合和欠拟合
  - 原始的损失函数，后边加上 $\sum_{\boldsymbol{w}}\|\boldsymbol{w}\|_F + \sum_{\boldsymbol{b}}\|\boldsymbol{b}\|_F$
  - Dropout：
    - 训练

每轮随机抽样被丢弃的神经元

      - 选定一定丢弃比例β
      - 前向对比例为β的神经元，激活值为0
      - 后向对应神经元误差也清0，误差不会经过这些神经元

实用训练技巧

- 学习率调节
  - 预设learning rate
  - 参考验证集调节
  - 学习率指数衰减
  - AdaDELTA, RMSProp

总结

- 神经网络：可视化了从输入到输出的计算流程
- 全连接神经网络
  - 要素一：
    - 引入隐含层的作用；
    - Softmax处理多分类的问题；
    - 目标函数可以处理回归问题（Square Loss），可以处理分类问题
  - 要素二：
    - 总体还是梯度下降算法；
    - BP算法：相邻层之间的梯度有迭代计算关系
  - 要素三：
    - 预测简单

作业

- http://playground.tensorflow.org/
  - 在4个数据上找到完美分类的参数
    - 使用的神经元越少越好
  - 提交要求：
    - 上传分类好的4张图片：每个分类问题一张