中国科学院大学

University of Chinese Academy of Sciences

Deep Learning

Improvement of Generalization

Xinfeng Zhang（张新峰）

School of Computer Science and Technology

University of Chinese Academy of Sciences

Email: xfzhang@ucas.ac.cn

计算机科学与技术学院

SCHOOL OF COMPUTER SCIENCE AND TECHNOLOGY

提纲

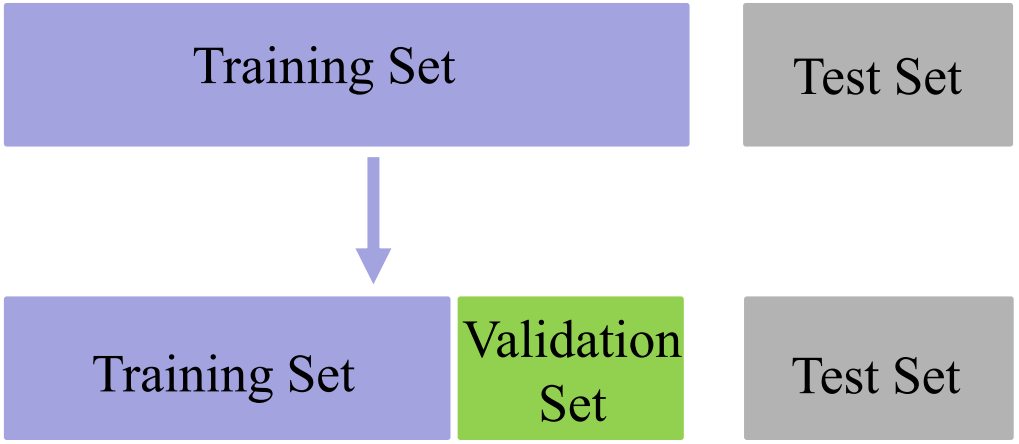
- 正则化方法概述
- 参数范数正则化
- 数据增强
- Batch Normalization
- Bagging
- Dropout和DropConnect
- 提前终止
- 中英文术语对照

1 正则化方法概述

数据集

❑ 数据集分类

- 训练集(Training set)：用于模型拟合的数据样本
- 验证集(Validation set)：是模型训练过程中单独留出的样本集，它可以用于调整模型的超参数和用于对模型的能力进行初步评估
  - 例如SVM中参数c和核函数的选择，或者选择网络结构
- 测试集(Test set)： 用来评估模最终模型的泛化能力。但不能作为调参、选择特征等算法相关的选择的依据



通常可以选择训练集、验证集和测试集数据比例为6:2:2

过拟合和欠拟合

❑ 过拟合

- 是指模型能很好地拟合训练样本，而无法很好地拟合测试样本的现象，从而导致泛化性能下降
- 为防止“过拟合”，可以选择减少参数、降低模型复杂度、正则化等

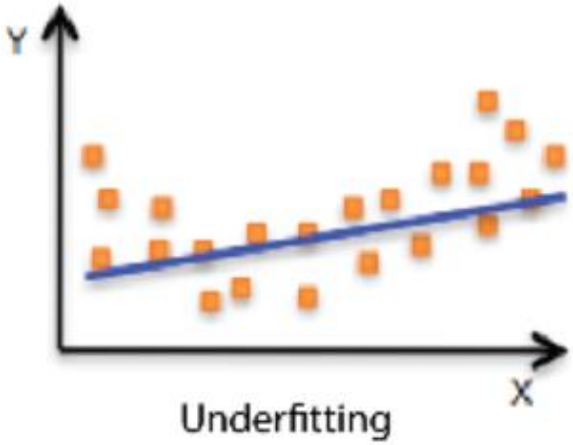
❑ 欠拟合

- 是指模型还没有很好地训练出数据的一般规律，模型拟合程度不高的现象
- 为防止“欠拟合”，可以选择调整参数、增加迭代深度、换用更加复杂的模型等

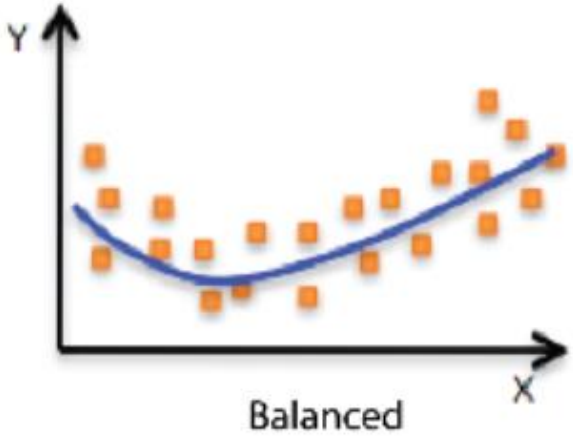
误差分析

❑ 模型训练的状态

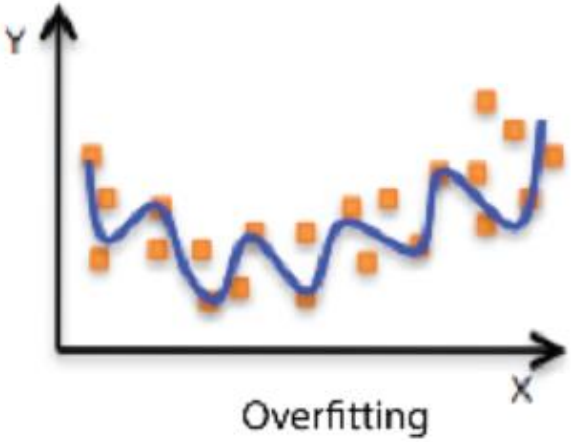
– 欠拟合、合适和过拟合的模型



训练集和测试集误差都比较大



训练集和测试集误差都比较小

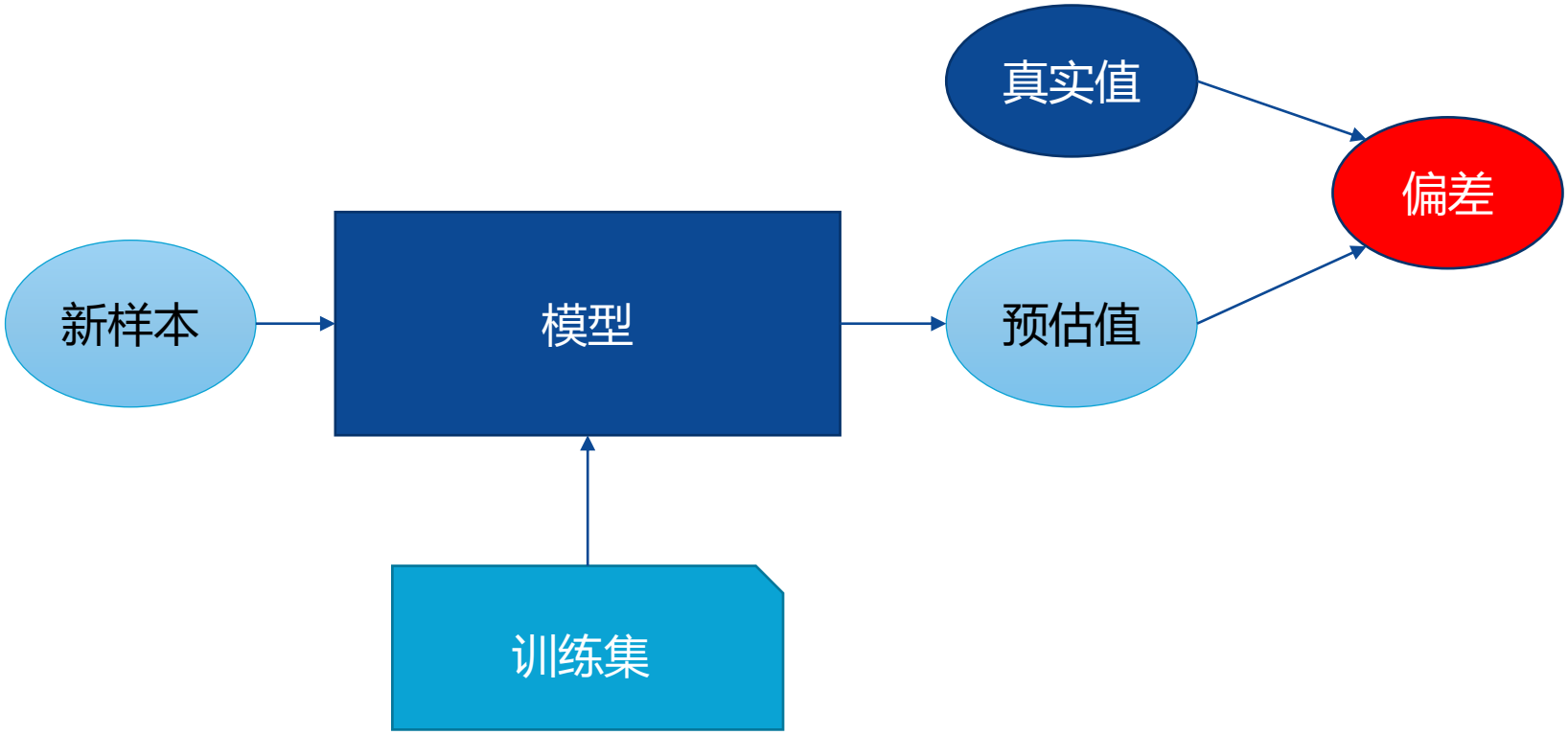


训练集误差小
测试集误差大

误差分析

❑ 偏差（bias）

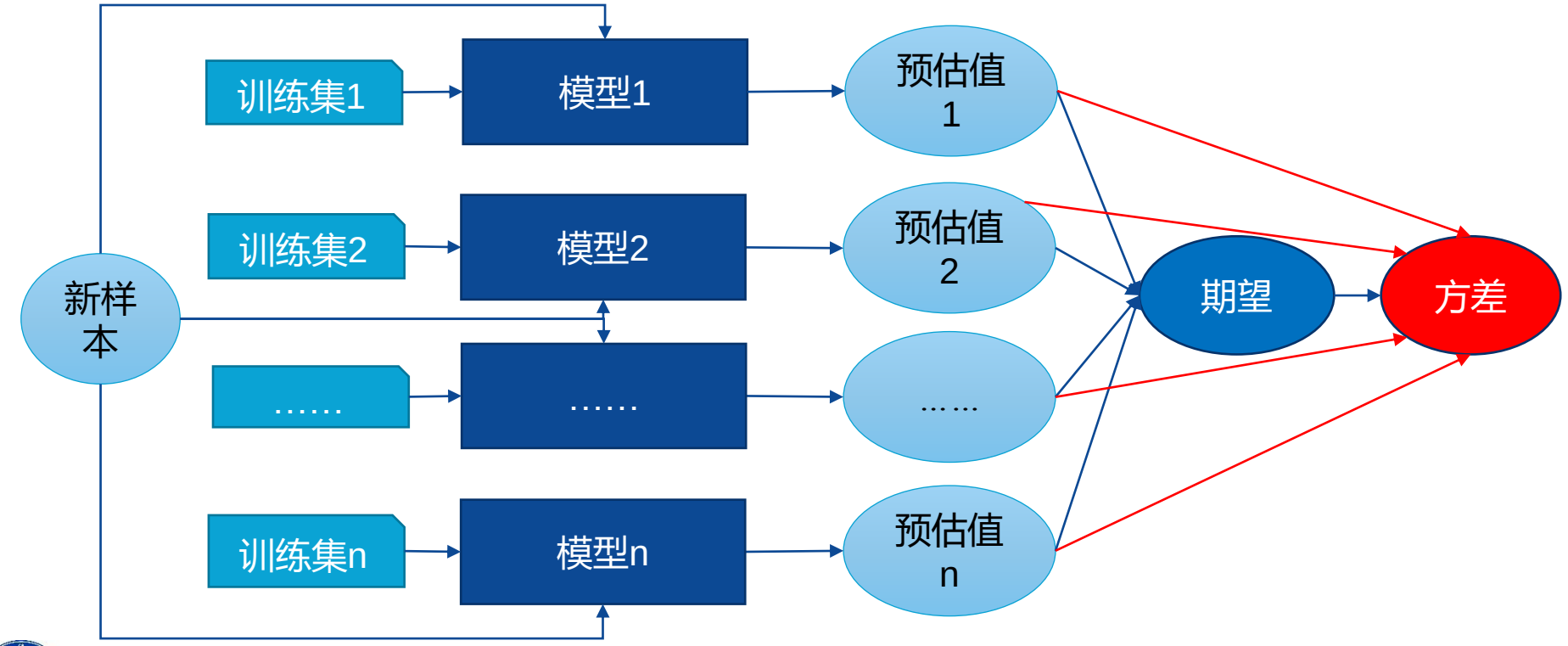
– 反映了模型在样本上的期望输出与真实标记之间的差距，即模型本身的精准度，反映的是模型本身的拟合能力



误差分析

❑ 方差（variance）

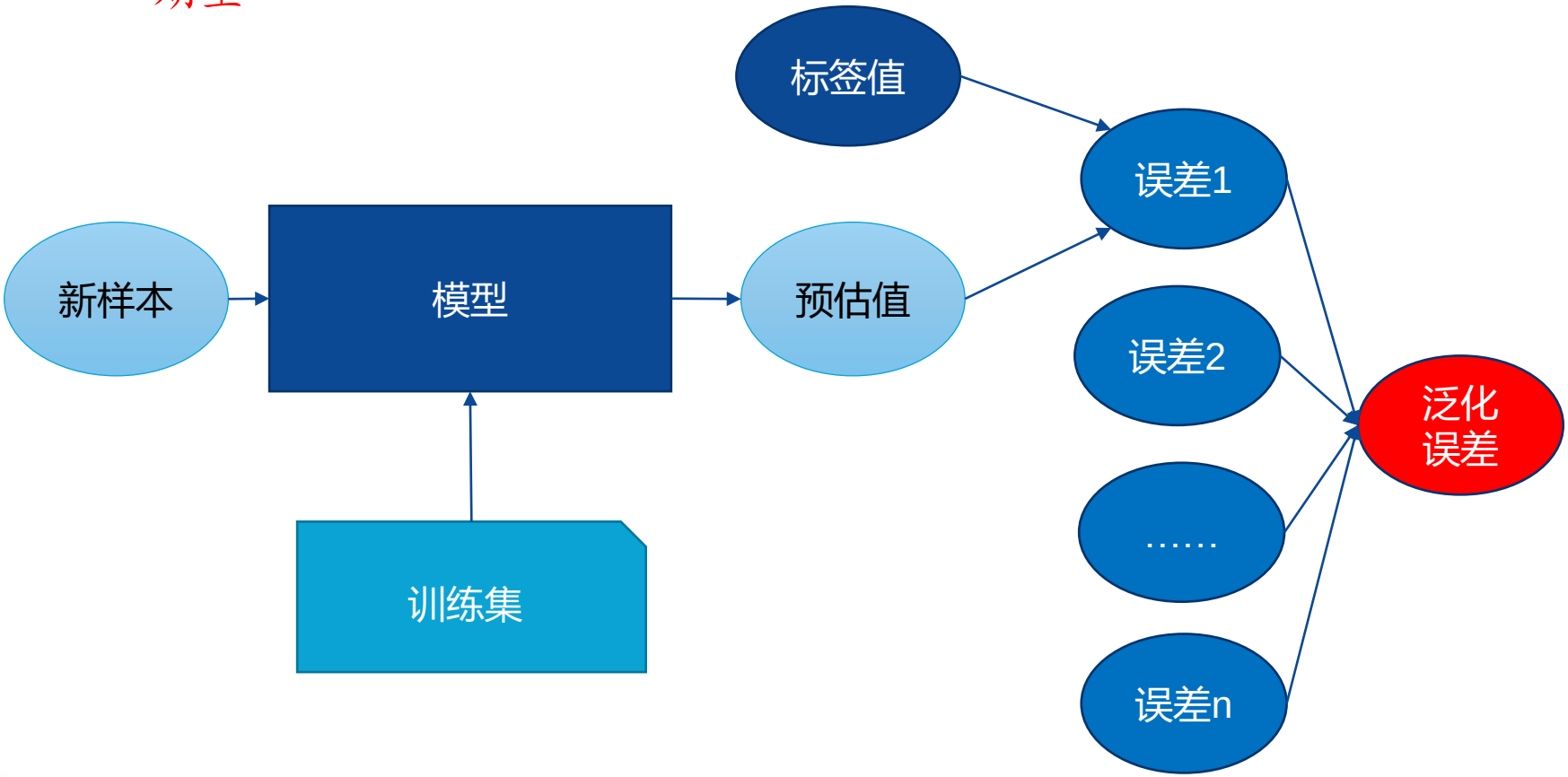
– 反映了模型在不同训练数据集下学得的函数的输出与期望输出之间的误差，即模型的稳定性，反应的是模型的波动情况



误差分析

❑ 泛化误差（Generalization Error）

– 度量训练所得模型在总体数据上得到的预估值和标签值偏离程度的期望



举例说明

- ①、④两种情况的训练集误差都很小，这种就称为low bias，说明训练的很到位了
- ②、③两种情况的训练集误差很大，这就称为high bias

	①	②	③	④
Train set(训练集) error	1%	15%	15%	0.8%
Dev set (验证集) error	11%	16%	30%	1%

（假设human performance为0%）

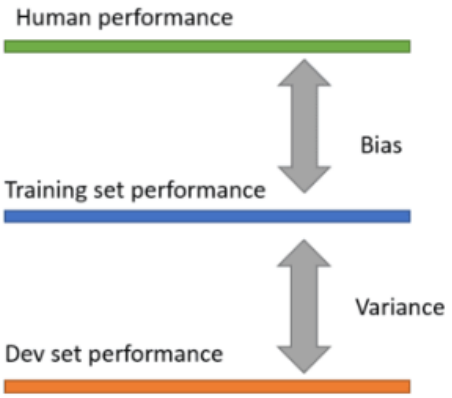
举例说明

- ①情况下，验证集相比训练集误差上升了很多，这就是high variance
- ②情况下，虽然它的验证集误差更大，但是相比它的训练集误差，基本没太大变化
- ④情况下，验证集误差相比训练集误差上升很小，我们说是Low variance

	1	2	3	4
Train set(训练集) error	1%	15%	15%	0.8%
Dev set (验证集) error	11%	16%	30%	1%

（假设human performance为0%）

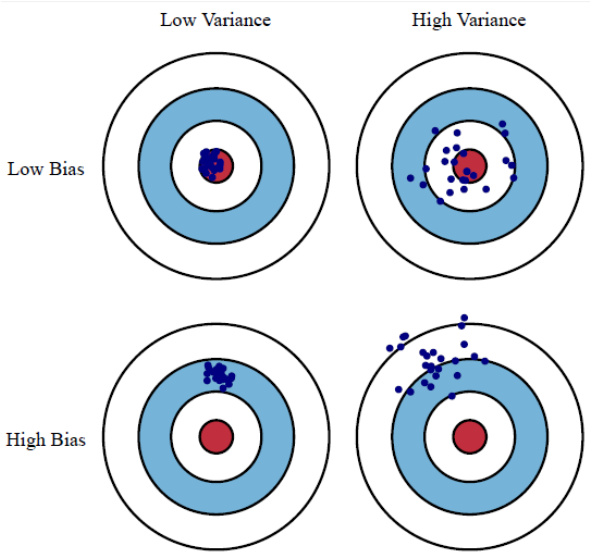
举例说明



	1	2	3	4
Train set(训练集) error	1%	15%	15%	0.8%
Dev set (验证集) error	11%	16%	30%	1%

（假设human performance为0%）

过拟合（Over fitting）　欠拟合（Under fitting）　烂（High bias& High variance）　赞！(Low bias& Low variance)



深度学习的一般步骤

偏差计算(训练集)
High Bias?

- 更复杂的网络结构（增加神经元数量，层数，更改结构）
- 更长的训练时间（迭代次数）

方差计算（验证集）
High Variance?

- 正则化方法
- 更多的数据
- 减少网络参数
- 降低模型复杂度

泛化误差计算（测试集）

正则化概念

❑ 好的模型

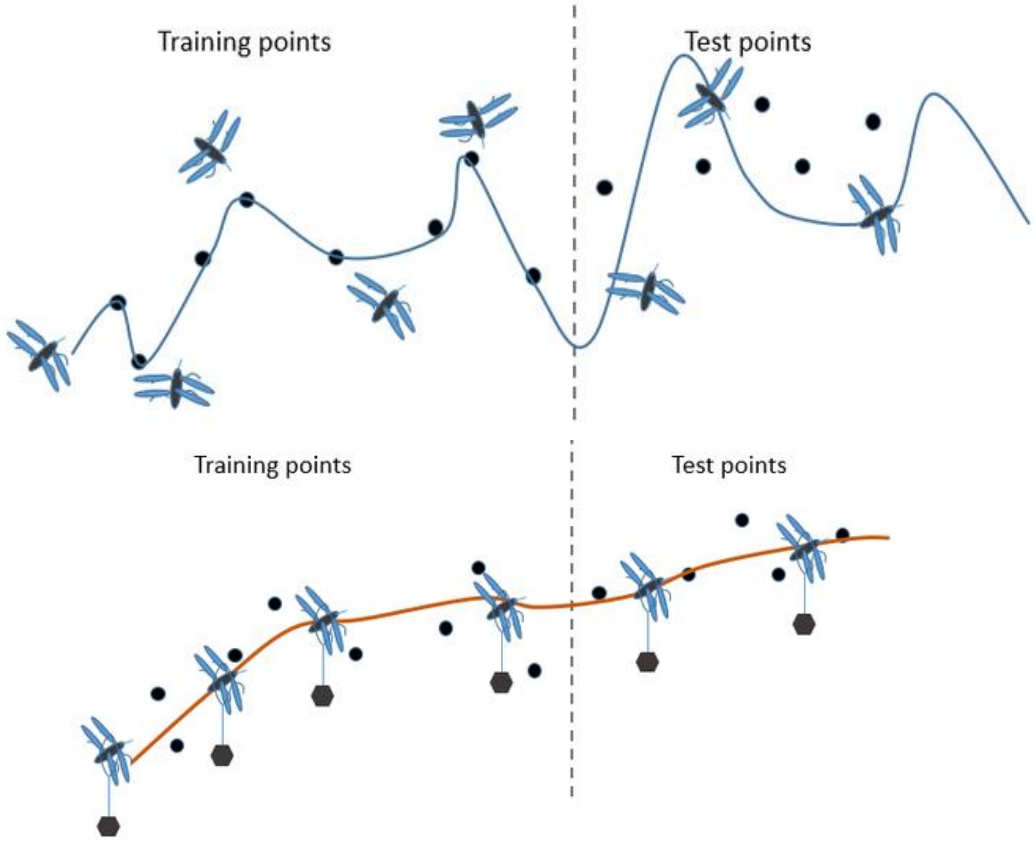
- 在训练数据上表现好？（基本要求）
- 且在新输入（测试数据）上有较好的泛化性。（重点）

❑ 正则化（Regularization）

- 广义正则化：通过某种手段使学习算法在训练误差变化不大的情况下，使得泛化误差显著降低的方法
- 狭义正则化：不减少网络参数，只进行参数范围调整的方法

正则化概念（举例说明）

- 正则化相当于“挂坠”
- 狭义正则化就是“给损失函数加一个正则化项（挂坠）



中国科学院大学
University of Chinese Academy of Sciences

2 参数范数正则化

范数

- 机器学习中，很多时候都需要衡量一个向量的大小，此时便需要用到范数的知识
- 范数是将向量映射到非负值的函数，它满足三条性质：
  - 非负性 $f(x) \geq 0 \left(f(x) = 0 \Leftrightarrow x = 0 \right)$
  - 齐次性 $\forall \alpha \in R, f(\alpha x) = |\alpha| f(x)$
  - 三角不等式 $f(x+y) \leq f(x) + f(y)$

范数

- 常用范数：L_p范数是使用最为广泛的一种范数，$\|x\|_p = (\sum_i |x_i|^p)^{1/p}$，其中p=2 时，该范数等价于向量和原点的欧几里得距离
- 有时候也需要衡量矩阵的大小，在深度学习中，最常见的做法便是使用Frobenius范数(F范数)，即$\|A\|_F = \sqrt{\sum_{i,j} A_{ij}^2}$

参数范数正则化

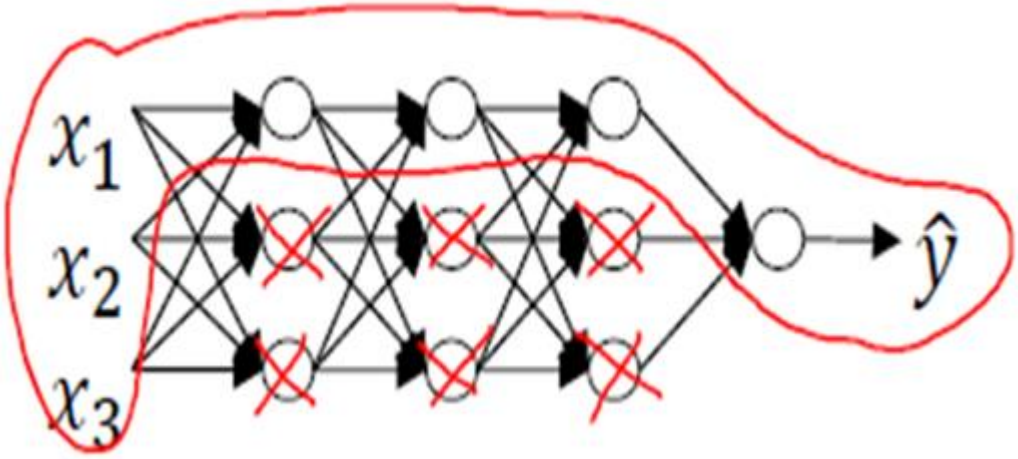
- 通过简单地在损失函数$J(\cdot)$后添加一个参数范数正则化项$\Omega(\theta)$，来限制模型的学习能力，正则化后的损失函数可表示为$\tilde{J}$:

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

  - 其中，α是正则项系数，它是一个超参，用来权衡惩罚项Ω对损失函数的贡献。若α为0，表示无正则化；α越大，正则化惩罚越大。
  - 我们希望正则化后的训练误差（第一项，蓝色）最小，又希望模型尽量简单（第二项，红色）

参数范数正则化

- 当最小化正则化后的损失函数时候，同时降低原有损失函数J(·)和正则化项Ω(θ)的值，一定程度减少了过拟合的程度
- 通俗来说，正则化就是让参数，如权重w多几个等于0，或者接近于0，说明该节点影响很小，如果是神经网络相当于在神经网络中删掉了一些节点，这样模型就变简单了



参数范数正则化

- 为了让参数多几个为0，对于正则化项Ω， 定义如下3种范数

  - L0范数：$\|\mathbf{W}\|_0$，指向量中非0的元素的个数，越小说明0元素越多
  - L1范数： $\|\mathbf{W}\|_1$，指向量中各个元素绝对值之和
  - L2范数： $\|\mathbf{W}\|_2$，即各元素的平方和再开方

L2 正则化

- L2正则化主要用于线性回归，又称为岭回归(Ridge Regression) 或权重衰减(Weight decay)，添加的正则化项形式如下:

$$\Omega(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$$

- 损失函数

$$\tilde{J}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \mathrm{J}(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \frac{\alpha}{2}\|\mathbf{w}\|_2^2$$

L2 正则化

❑ L2参数正则化分析（单步）

– 对于某一模型，假设要对其所有的参数进行参数正则化，则正则化后的整体损失函数$\tilde{J}$如下

$$\tilde{J}(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y}) = J(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y}) + \frac{\alpha}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$$

– 计算梯度

$$\nabla_{\mathbf{w}}\tilde{J}(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y}) = \nabla_{\mathbf{w}}J(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y}) + \alpha\boldsymbol{w}$$

L2 正则化

❑ L2参数正则化分析（单步）

– 使用单步梯度下降更新权重：

$$\mathbf{w} \leftarrow \mathbf{w} - \lambda(\nabla_{\mathbf{w}} J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) + \alpha \boldsymbol{w})$$

$$= (1 - \lambda\alpha)\mathbf{w} - \lambda\nabla_{\mathbf{w}} J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})$$

– 在进行每步梯度更新前，先对参数w进行了缩放。因此，这也是权重衰减名称的来源

L2 正则化

❑ L2参数正则化分析（整个训练过程）

- $\mathbf{w}^*$：未正则化的损失函数J得到的最小损失权重向量，即：

$$\mathbf{w}^* = argmin_{\boldsymbol{w}} J(\boldsymbol{w})$$

- 在$\mathbf{w}^*$附近对损失函数做二次近似，近似的$\hat{J}(\mathbf{w})$如下：

$$\hat{J}(\mathbf{w}) = J(\mathbf{w}^*) + (\mathbf{w} - \mathbf{w}^*)^T \nabla J(\mathbf{w}^*) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^T \boldsymbol{H}(\mathbf{w} - \mathbf{w}^*)$$

- H是J在$\mathbf{w}^*$处计算的*Hessian*矩阵。当$\hat{J}(\boldsymbol{w})$取得最小值时，其梯度应为0

$$\nabla_{\mathbf{w}} \hat{J}(\boldsymbol{w}) = \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*) = 0$$

$$H(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

L2 正则化

❑ L2参数正则化分析（整个训练过程）

– 引入L2正则化项的梯度，设变量$\tilde{\mathbf{w}}$使得$\hat{J}(\boldsymbol{w})$最小

$$\nabla_{\mathbf{w}}\hat{J}(\boldsymbol{w}) = \alpha\tilde{\mathbf{w}} + \boldsymbol{H}(\tilde{\mathbf{w}} - \boldsymbol{w}^*) = 0$$

– 求得：

$$\tilde{\mathbf{w}} = (\boldsymbol{H} + \alpha I)^{-1}\boldsymbol{H}\mathbf{w}^*$$

– 因为$\boldsymbol{H}$是对称矩阵，可分解为$H = Q\Lambda Q^T$，所以

$$\tilde{\mathbf{w}} = Q(\Lambda + \alpha I)^{-1}\Lambda Q^T\mathbf{w}^*$$

L2 正则化

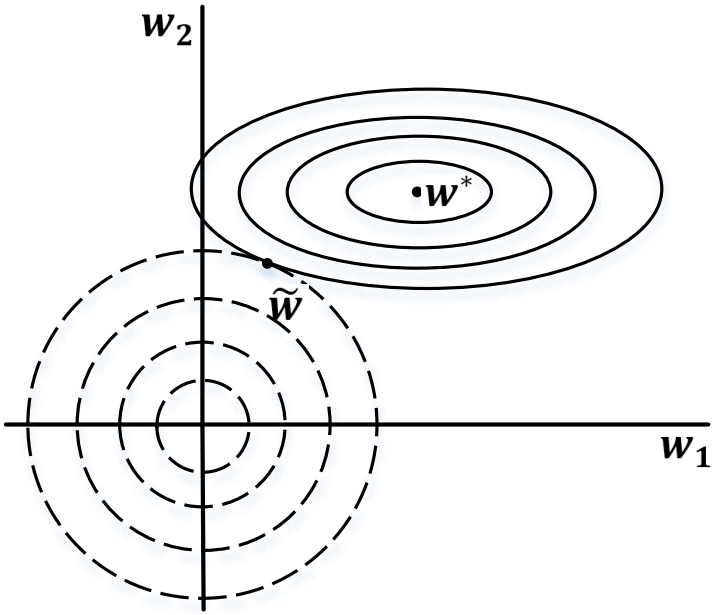
❑ L2参数正则化分析（整个训练过程）

$$\tilde{\mathbf{w}} = Q(\Lambda + \alpha I)^{-1}\Lambda Q^T \mathbf{w}^*$$

$$(\Lambda + \alpha I)^{-1}\Lambda = \begin{bmatrix} \frac{\lambda_1}{\lambda_1+\alpha} & 0 & \cdots & 0 \\ 0 & \frac{\lambda_2}{\lambda_2+\alpha} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\lambda_n}{\lambda_n+\alpha} \end{bmatrix}$$

$$\frac{w_1^2}{a^2} + \frac{w_2^2}{b^2} = 1$$

$$\lambda_1 = \frac{2}{a^2}, \lambda_2 = \frac{2}{b^2}$$



可见，权重衰减是沿着$\boldsymbol{H}$的特征向量的标准正交基所定义的轴缩放$\boldsymbol{w}^*$。当原损失函数在特征值大的方向进行下降时，正则化项对其影响较小；在特征值小的方向进行下降时，正则化项会对其进行限制。也就是说，保证了原损失函数沿着损失下降最大的方向下降

稀疏问题

- **特征选择**：x_i的大部分元素（也就是特征）都是和最终的输出y_i没有关系；但在预测新的样本时，这些没用的信息反而会被考虑，从而干扰了对正确y_i的预测。稀疏规则化算子的引入就是为了完成**特征自动选择**，它会学习地去掉这些没有作用的特征，也就是把这些特征对应的**权重置为0**

- **可解释性**：患病回归模型$y=w_1*x_1+w_2*x_2+\ldots+w_{1000}*x_{1000}+b$，通过学习，如果最后学习到的w*就只有很少的非零元素，例如只有5个非零的w_i，也就是说，患不患这种病只和这5个因素有关，那医生就好分析多了

L1 正则化

- **L1正则化又称为Lasso回归（Lasso Regression），在损失函数中添加待正则化参数w的L1-范数（即w所有参数绝对值之和）作为正则化项**

$$\Omega(\boldsymbol{w}) = \|\boldsymbol{w}\|_1$$

  - 损失函数：

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

L1 正则化

❑ L1参数正则化分析

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

– 计算梯度：

$$\nabla_{\mathbf{w}}\tilde{J}(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y}) = \nabla_{\mathbf{w}}J(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y}) + \alpha \cdot sign(\boldsymbol{w})$$

– 其中，sign(w)只是简单的取w各个元素的正负号。L1正则化与L2不大一样，不一定能得到$J(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y})$二次近似的直接算术解

L1 正则化

❑ L1参数正则化分析

- $\mathbf{w}^*$：未正则化的损失函数J得到的最小损失权重向量，即

$$\mathbf{w}^* = argmin_{\boldsymbol{w}} J(\boldsymbol{w})$$

- 在$\mathbf{w}^*$处对正则化后的损失函数进行二次泰勒展开

$$\hat{J}(\mathbf{w}) = J(\mathbf{w}^*) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^T \boldsymbol{H}(\mathbf{w} - \mathbf{w}^*)$$

- H是J在$\mathbf{w}^*$处计算的$Hessian$矩阵。当$\hat{J}(\boldsymbol{w})$取得最小值时，其梯度应为0

$$\nabla_{\mathbf{w}} \hat{J}(\boldsymbol{w}) = \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*) = 0$$

L1 正则化

❑ L1参数正则化分析

– L1正则项在完全一般化的Hessian矩阵无法得到清晰的代数表达式，进一步假设Hessian为对角矩阵，即$\mathbf{H} = \mathrm{diag}([\mathrm{H}_{1,1}, \cdots, \mathrm{H}_{n,\mathrm{n}}])$，且$\mathrm{H}_{i,i} > 0$。(如果线性回归问题中的数据已被预处理，去除输入特征之间的相关性，那么这一假设成立)

– 可将目标函数二次近似分解为关于参数的总和

$$\tilde{J}(\boldsymbol{w}) = J(\boldsymbol{w}^*) + \sum_i \left[\frac{1}{2} H_{i,i}(w_i - w_i^*)^2 + \alpha|w_i|\right]$$

L1 正则化

❑ L1参数正则化分析

– L1正则项在完全一般化的Hessian矩阵无法得到清晰的代数表达式，进一步假设Hessian为对角矩阵，即$\mathbf{H} = \mathrm{diag}([\mathrm{H}_{1,1}, \cdots, \mathrm{H}_{n,\mathrm{n}}])$，且$\mathrm{H}_{i,i} > 0$。(如果线性回归问题中的数据已被预处理，去除输入特征之间的相关性，那么这一假设成立)

– 可将目标函数二次近似分解为关于参数的总和

$$\tilde{J}(\boldsymbol{w}) = J(\boldsymbol{w}^*) + \sum_i \left[\frac{1}{2} H_{i,i}(w_i - w_i^*)^2 + \alpha|w_i|\right]$$

$$\widetilde{w_i} = w_i^* - sign(w_i)\frac{\alpha}{H_{i,i}}$$

L1 正则化

❑ L1参数正则化分析

– 最小化这个近似损失函数可得

$$\widetilde{w_i} = sign(w_i^*)\max\left\{|w_i^*| - \frac{\alpha}{H_{i,i}}, 0\right\}$$

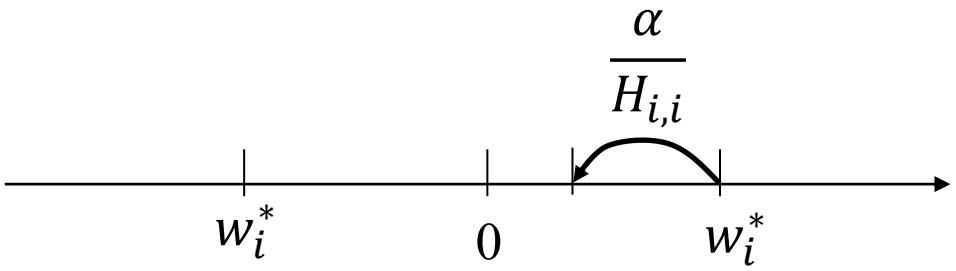
这里w_i和w_i^*符号相同

– w_i>0时分两种情况

- $w_i^* \le \frac{\alpha}{H_{ii}}$; $\widetilde{w_i}$=0
- $w_i^* > \frac{\alpha}{H_{ii}}$; $\widetilde{w_i}$=$w_i^* - \frac{\alpha}{H_{i,i}}$

– w_i <0时分两种情况

- $w_i^* \le -\frac{\alpha}{H_{ii}}$; $\widetilde{w_i}$=$w_i^* + \frac{\alpha}{H_{i,i}}$
- $w_i^* > -\frac{\alpha}{H_{ii}}$; $\widetilde{w_i}$=0



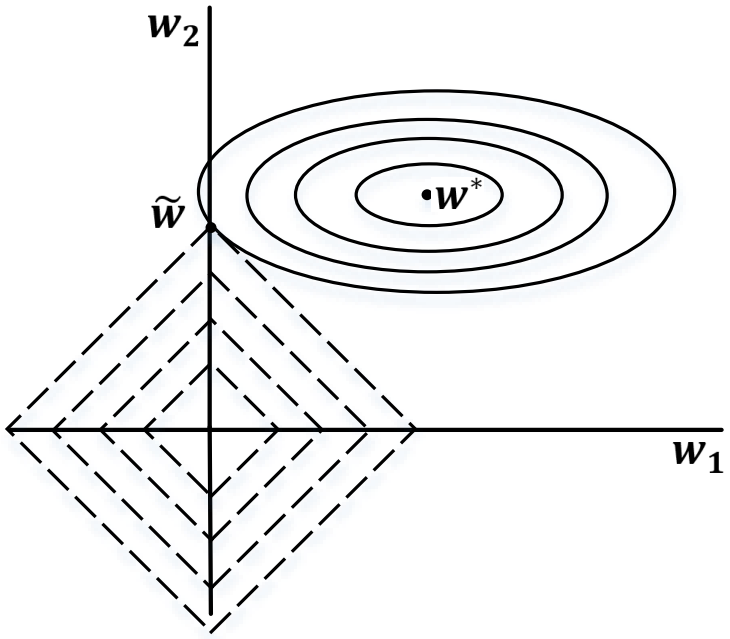
L1 正则化

❑ L1参数正则化分析

– 最小化这个近似损失函数可得

$$\widetilde{w_i} = sign(w_i^*)\max\left\{|w_i^*| - \frac{\alpha}{H_{i,i}}, 0\right\}$$

– 得到的参数$\widetilde{w}$的各个元素可能为0。因此，与L2正则化相比，得到的参数更加稀疏



L1 与L2的区别

- L2范数更有助于计算病态的问题
- L1相对于L2能够产生更加稀疏的模型
- 从概率角度进行分析，很多范数约束相当于对参数添加先验分布，其中L2范数相当于参数服从高斯先验分布；L1范数相当于拉普拉斯分布

$$\underset{\boldsymbol{w}}{\operatorname{argmax}}\, p(\boldsymbol{w}|X) = \underset{\boldsymbol{w}}{\operatorname{argmax}} \frac{p(X|\boldsymbol{w})p(\boldsymbol{w})}{p(X)} = \underset{\boldsymbol{w}}{\operatorname{argmax}}\, p(X|\boldsymbol{w})p(\boldsymbol{w})$$

$$= \underset{\boldsymbol{w}}{\operatorname{argmax}}(\prod_{i=1}^{n} p(x_i|\boldsymbol{w}))p(\boldsymbol{w})$$

$$\operatorname{argmax}_{\boldsymbol{w}} L(\boldsymbol{w}) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} exp\left(\frac{(y_i - \boldsymbol{w}^T x_i)^2}{2\sigma^2}\right) \prod_{j=1}^{d} \frac{1}{\tau\sqrt{2\pi}} exp\left(\frac{(w_j)^2}{2\tau^2}\right)$$

$$\operatorname{argmax}_{\boldsymbol{w}} L'(\boldsymbol{w}) = ln \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{(y_i - \boldsymbol{w}^T x_i)^2}{2\sigma^2}\right) \prod_{j=1}^{d} \frac{1}{\tau\sqrt{2\pi}} exp\left(-\frac{(w_j)^2}{2\tau^2}\right)$$

$$= -\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \boldsymbol{w}^T x_i)^2 - \frac{1}{2\tau^2}\sum_{i=1}^{d}(w_j)^2 + \ldots$$

稀疏表示

- 数据集可视为矩阵S，其中矩阵的行数表示样本的数量，矩阵的列数则表示对应的特征。在很多应用中，矩阵S中存在很多零元素，但这些零元素并不是以整列、整行形式存在
  - 如在文档分类领域，我们将每一篇文档作为一个样本，那么数据集可表示为每一行表示为一个文档，每一列为一个字（词），行列交汇处就是某字（词）在某文档中出现的频率

稀疏表示

- 如果矩阵S是普通的非稀疏矩阵，稀疏表示目的是为这个普通稠密表达的样本S找到一个合适的字典，将样本转换为稀疏表示（Sparse Representation）形式，从而在学习训练的任务上得到简化，降低模型的复杂度。亦被称作字典学习（Dictionary Learning）
  - 稀疏表示的本质是使用尽可能少的数据表示尽可能多的特征，并且同时也能带来计算效率的提升

稀疏表示学习

- 给定数据集为$\{x_1, x_2, \ldots, x_n\}$，字典学习可表示为

$$\min_{B,\alpha_i} \sum_{i=1}^{m} || x_i - B\alpha_i ||_2^2 + \lambda \sum_{i=1}^{m} || \alpha_i ||_1$$

其中$B \in \mathbb{R}^{d \times k}$，$k$为字典的词汇量，$\alpha_i \in \mathbb{R}^k$则是样本$x_i \in \mathbb{R}^d$的稀疏表示

- 第一项保证能够通过字典B由稀疏表示α_i重构样本数据x_i，第二项保证α_i尽可能稀疏。学习任务则是求解字典B，以及x_i的稀疏表示α_i，可采用变量交替优化的策略进行学习

稀疏表示学习

❑ 变量交替优化的策略

– 第一步，固定字典B，按照LASSO解法求解下式，从而可为每一个样本x_i求得对应的α_i

$$\min_{\alpha_i} \| x_i - B\alpha_i \|_2^2 + \lambda \| \alpha_i \|_1$$

– 第二步，利用求得的α_i更新字典B，求解下式进行更新

$$\min_{B} \| X - BA \|_F^2$$

其中$\mathrm{X} = (x_1, x_2, \cdots, x_m) \in \mathbb{R}^{d \times m}$，$\mathrm{A} = (\alpha_1, \alpha_2, \cdots, \alpha_m) \in \mathbb{R}^{k \times m}$，$\| \cdot \|_{\mathrm{F}}$ 为矩阵的Frobenius范数

– 对字典矩阵B进行初始化之后反复迭代上述两步，最终则可求得字典B和样本x_i对应的稀疏表示α_i

中国科学院大学
University of Chinese Academy of Sciences

3 数据增强

数据增强

❑ 数据增强(Data Augmentation)

- 可以简单的理解为对训练数据数量的扩充，利用已有的有限数据产生更多的有效数据。如图像空间变换、像素颜色变换。
- 原因：深度学习离不开大规模标签数据集的出现，如果数据量太少，极有可能使得模型陷入过拟合之中。
- 目的：避免过拟合、提高泛化性
- 实现：OpenCV、PIL及深度学习框架内嵌方法

数据增强

❑ 数据增强(Data Augmentation)

- 基于图像处理的方法
  - 空间变换：最常见也是最直观的图像增强手段之一，如对图像进行随机翻转、旋转、平移、裁剪等操作，在实际进行数据增强操作时，可以同时进行多种操作。
  - 像素颜色变换：对图像像素值进行修改，如在原始图片基础上进行对比度扰动、饱和度扰动、颜色变换、噪声扰动等操作
- 基于深度学习的方法
  - 基于GAN的数据增强（GAN-based Data Augmentation）：使用 GAN 生成模型来生成更多的数据，可用作解决类别不平衡问题的过采样技术
  - 神经网络风格转换（Neural network Style Transfer）：通过神经网络风格迁移来生成不同风格的数据，防止模型过拟

数据增强

❑ 基于图像处理的数据增强方法

– 示例：如在深度学习框架Keras中有一个专门ImageDataGenerator类可用于进行简单的数据增强，其参数如下

```
keras.preprocessing.image.ImageDataGenerator(featurewise_center=False,
                                             samplewise_center=False,
                                             featurewise_std_normalization=False,
                                             samplewise_std_normalization=False,
                                             zca_whitening=False,
                                             zca_epsilon=1e-06,
                                             rotation_range=0,
                                             width_shift_range=0.0,
                                             height_shift_range=0.0,
                                             brightness_range=None,
                                             shear_range=0.0,
                                             zoom_range=0.0,
                                             channel_shift_range=0.0,
                                             fill_mode='nearest',
                                             cval=0.0,
                                             horizontal_flip=False,
                                             vertical_flip=False,
                                             rescale=None,
                                             preprocessing_function=None,
                                             data_format=None,
                                             validation_split=0.0,
                                             dtype=None)
```

- featurewise_center: 将输入数据的均值设置为 0，逐特征进行
- featurewise_std_normalization: 将输入除以数据标准差，逐特征进行
- zca_whitening：对输入数据施加ZCA白化
- rotation_range：数据提升时图片随机转动的角度
- width_shift_range：图片宽度的某个比例，数据提升时图片随机水平偏移的幅度
- height_shift_range：图片高度的某个比例，数据提升时图片随机竖直偏移的幅度
- shear_range：剪切强度(逆时针方向的剪切变换角度)
- horizontal_flip：进行随机水平翻转
- vertical_flip：进行随机竖直翻转
- zoom_range: 随机缩放范围。
- channel_shift_range: 随机通道转换的范围
- fill_mode: 填充新像素的模式
- …

数据增强

❑ 基于图像处理的数据增强方法

– 示例：如在深度学习框架Keras中有一个专门ImageDataGenerator类可用于进行简单的数据增强，其参数如下

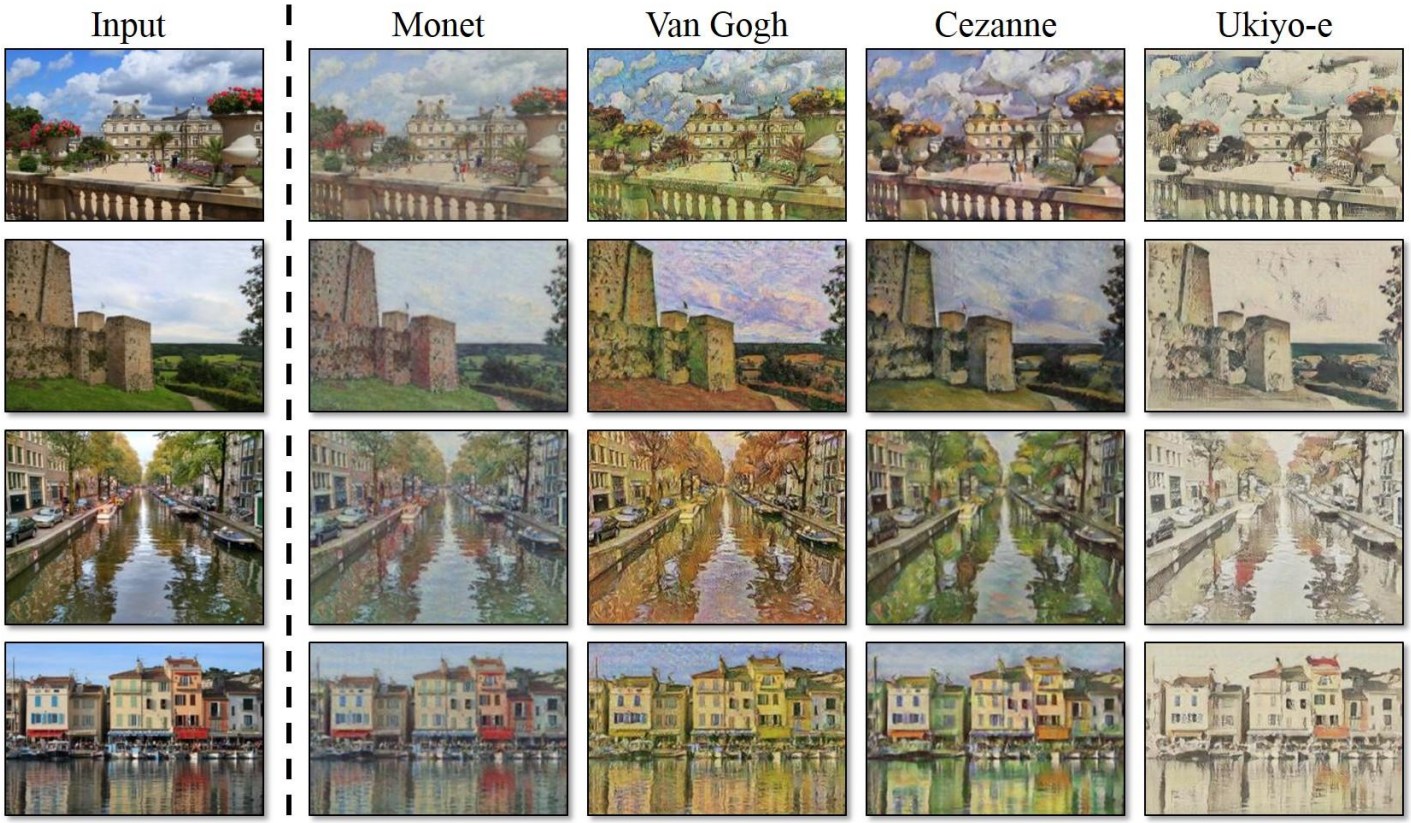
```
keras.preprocessing.image.ImageDataGenerator(featurewise_center=False,
    samplewise_center=False,
    featurewise_std_normalization=False,
    samplewise_std_normalization=False,
    zca_whitening=False,
    zca_epsilon=1e-06,
    rotation_range=0,
    width_shift_range=0.0,
    height_shift_range=0.0,
    brightness_range=None,
    shear_range=0.0,
    zoom_range=0.0,
    channel_shift_range=0.0,
    fill_mode='nearest',
    cval=0.0,
    horizontal_flip=False,
    vertical_flip=False,
    rescale=None,
    preprocessing_function=None,
    data_format=None,
    validation_split=0.0,
    dtype=None)
```

preprocessing_function: 自行编写的图像处理函数，对于ImageDataGenerator没有实现的数据增强操作，也可以自行编写函数传入到preprocessing_function中

数据增强

❑ 基于深度学习的方法

- 对抗生成网络数据增强
- 风格转换网络数据增强



数据增强

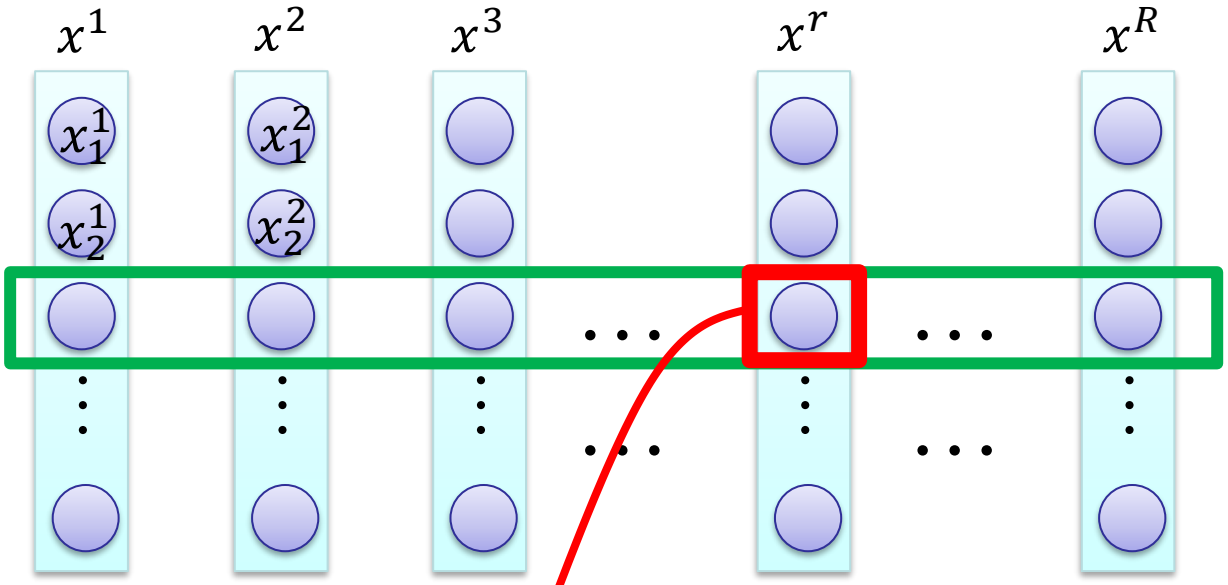
- 需要指出的是，对数据进行数据增强也需要视情况而定。如在手写数字识别任务中，为了区别6和9的区别，也不适合对数据进行垂直翻转以及旋转过大角度

GitHub上数据增强的开源实现https://github.com/aleju/imgaug

4 Batch Normalization

数据预处理

❑ Feature scaling



mean: m_i

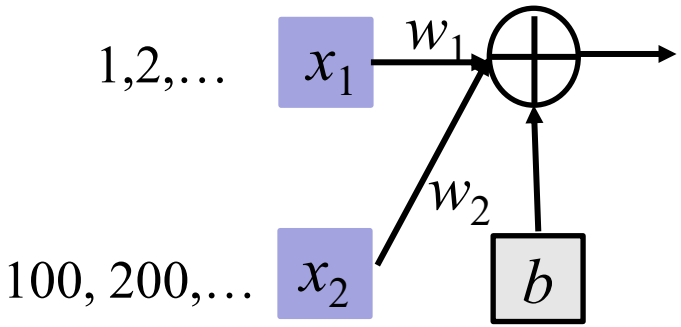
$$x_i^r \leftarrow \frac{x_i^r - m_i}{\sigma_i}$$

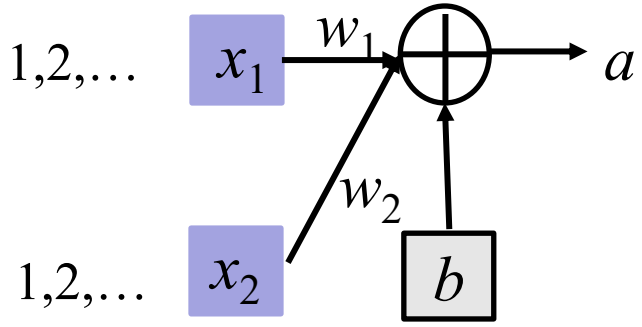
处理后所有维度均值为0方差为1

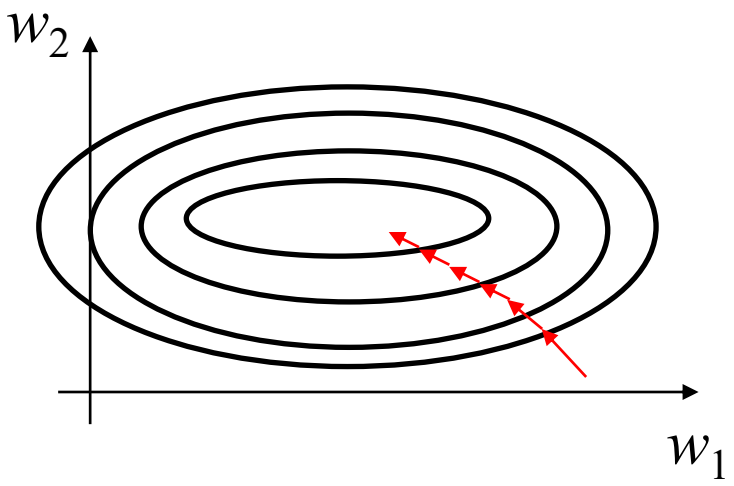
通常处理后会使得训练收敛加快

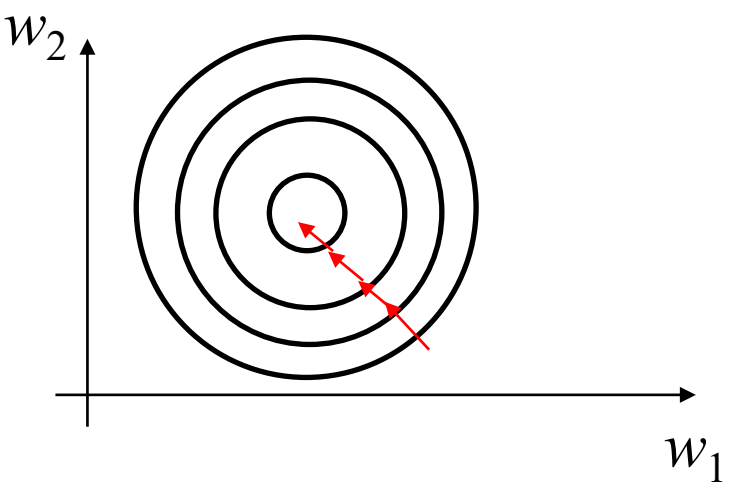
数据预处理

❑ Feature scaling



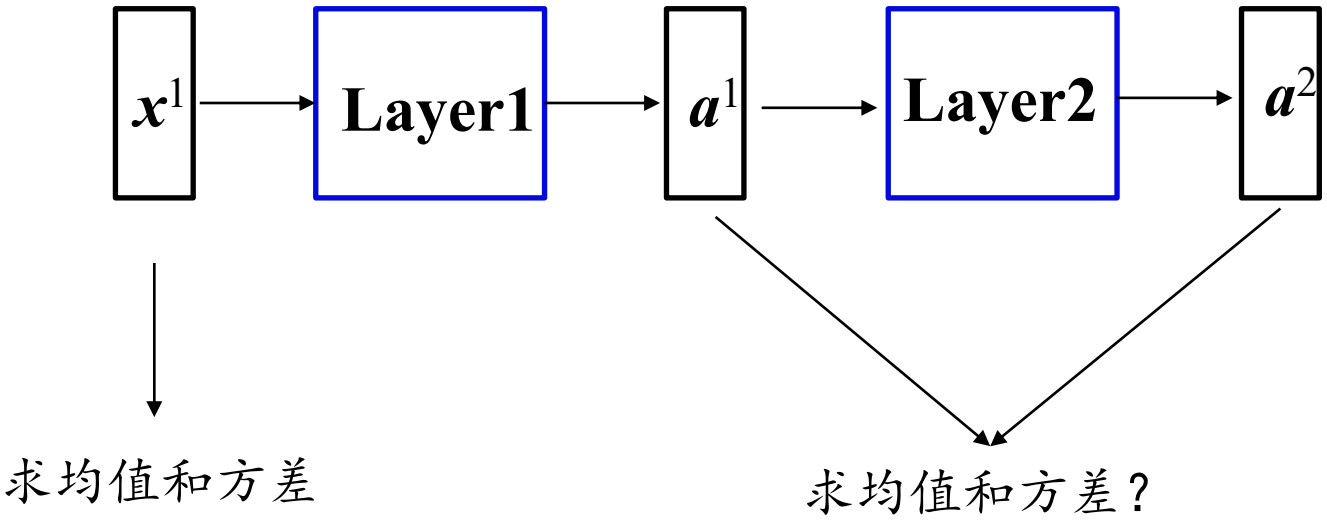






Batch Normalization

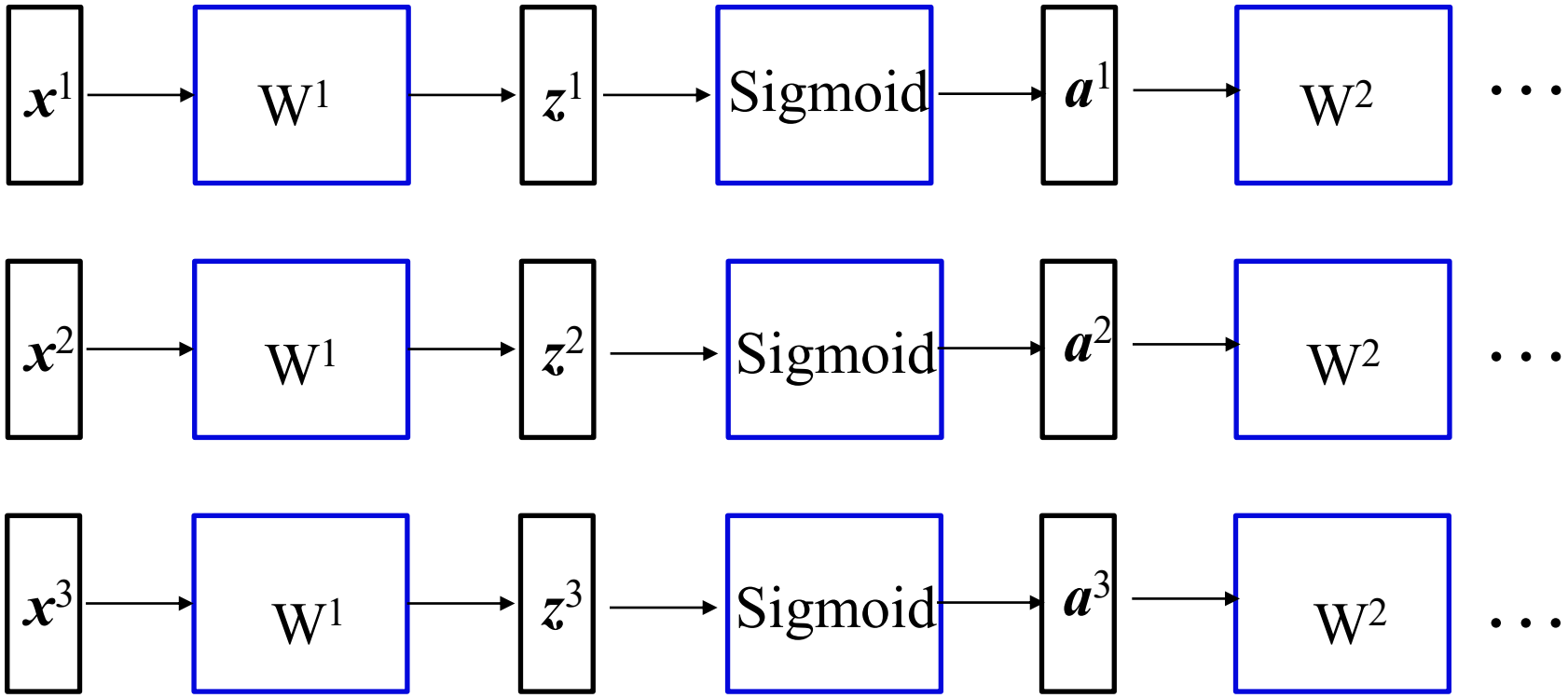
❑ 神经网络中的feature scaling



Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." In International conference on machine learning, pp. 448-456. PMLR, 2015.

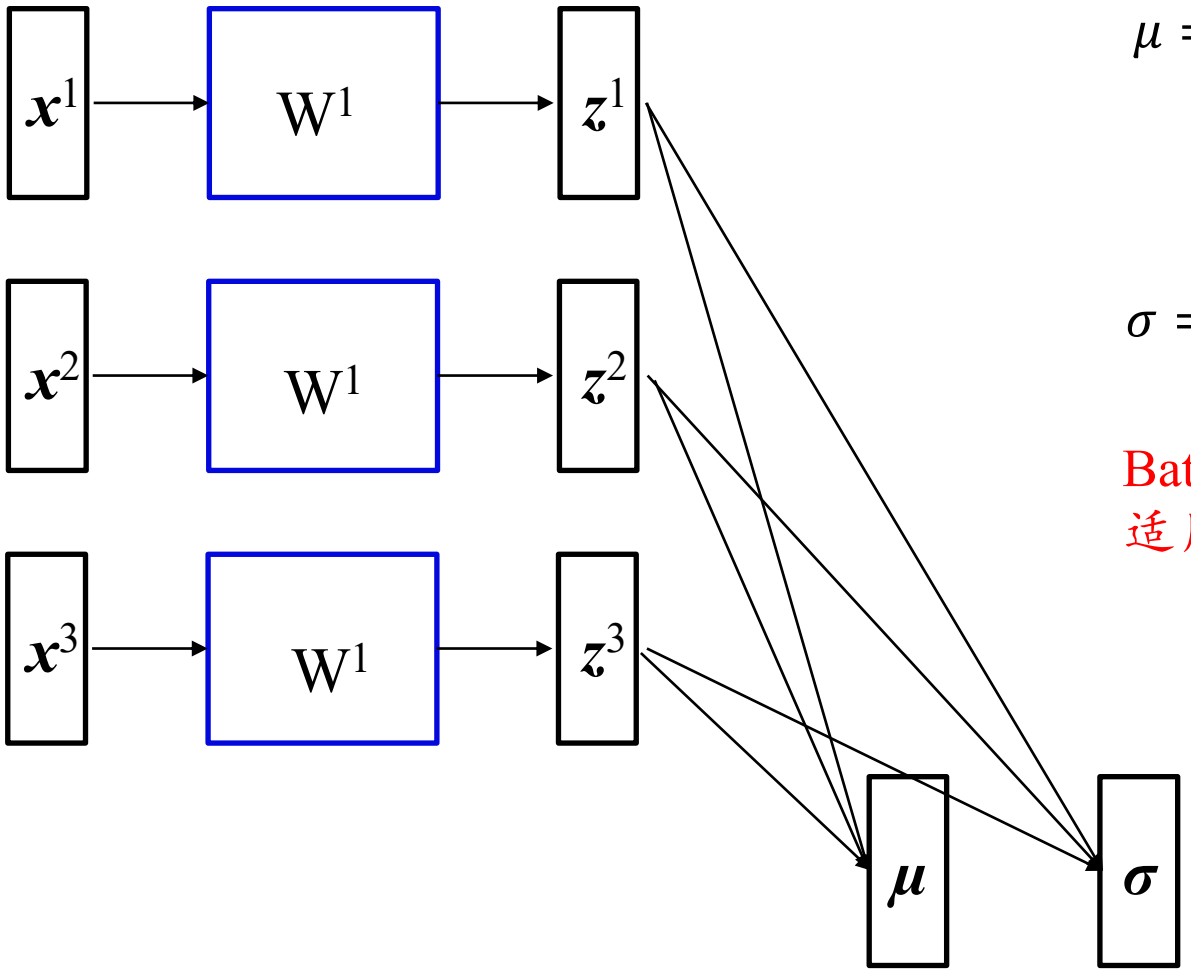
Batch Normalization

❑ 神经网络中的feature scaling



Batch Normalization

❑ 神经网络中的feature scaling



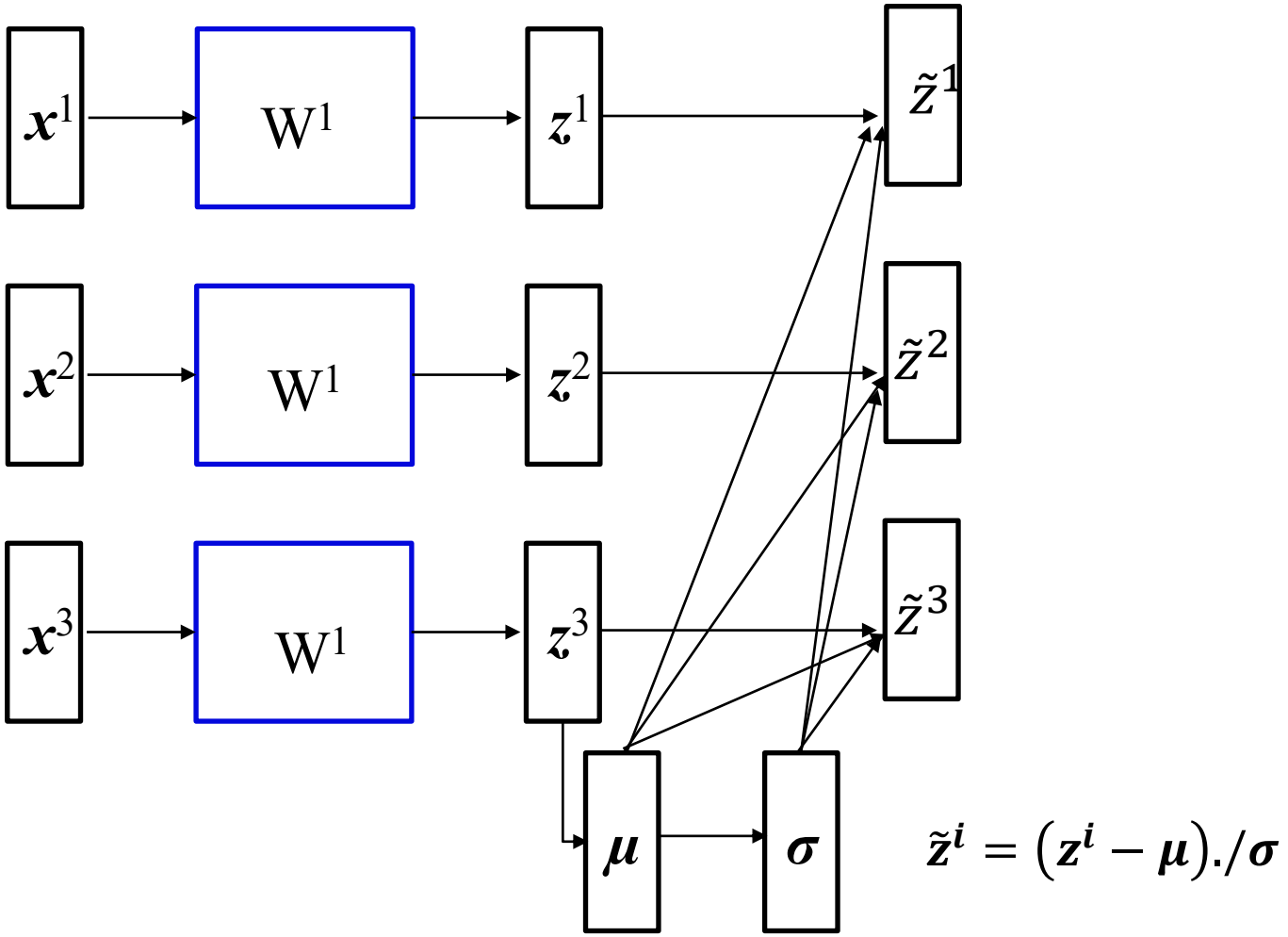
$$\mu = \frac{1}{N}\sum_{i=1}^{N} z^i$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(z^i - \mu)^2}$$

Batch normalization不适用于较小的batch

Batch Normalization

❑ 神经网络中的feature scaling



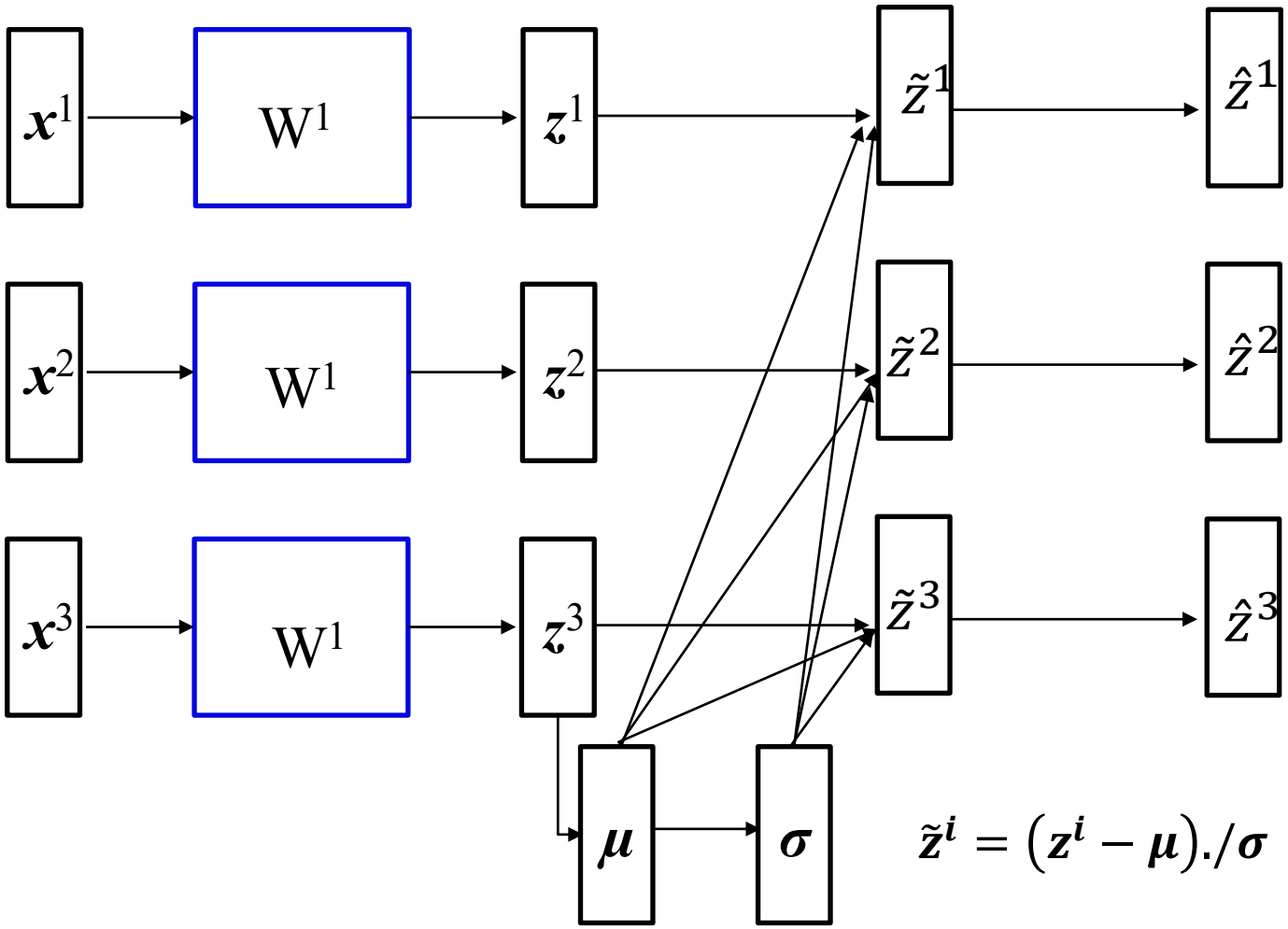
$$\tilde{z}^i = (z^i - \mu)./\sigma$$

Batch Normalization

❑ 神经网络中的feature scaling

$$\hat{z}^i = \gamma \odot \tilde{z}^i + \beta$$

γ和β是网络参数

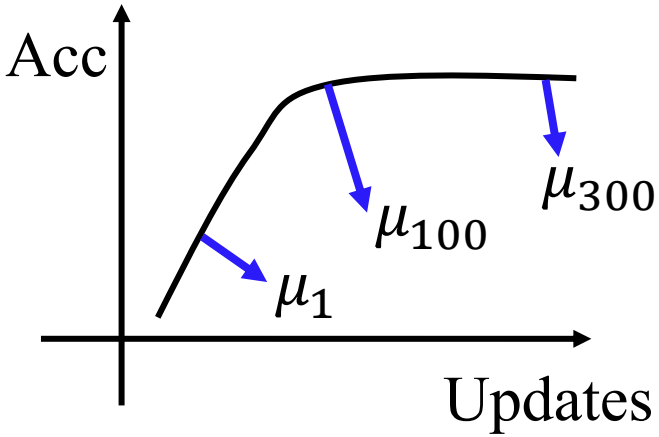


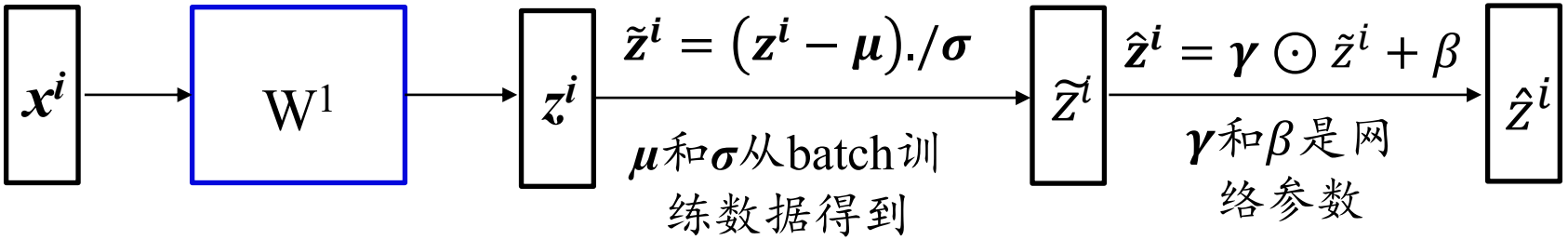
$$\tilde{z}^i = (z^i - \mu)./\sigma$$

Batch Normalization

❑ 神经网络中的feature scaling

– 在测试阶段如何得到μ和σ?





Ideal solution:

Computing μ and σ using the whole training dataset.

Practical solution:

Computing the moving average of μ and σ of the batches during training.

Batch Normalization

❑ Batch Normalization的流程

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

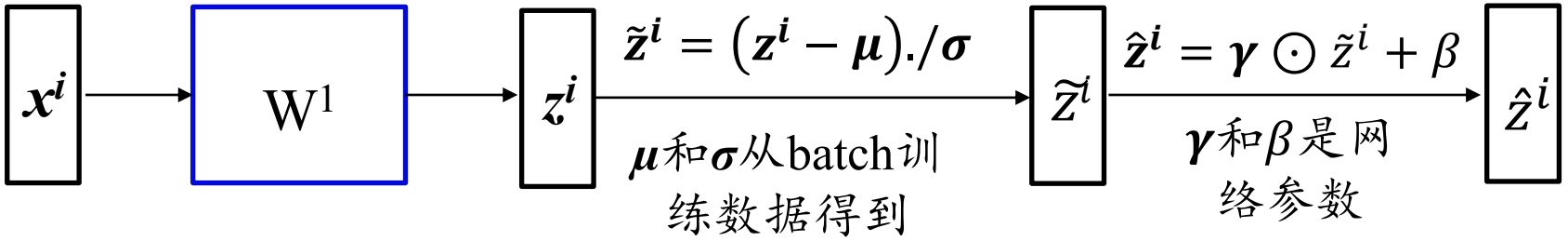
$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

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

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

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

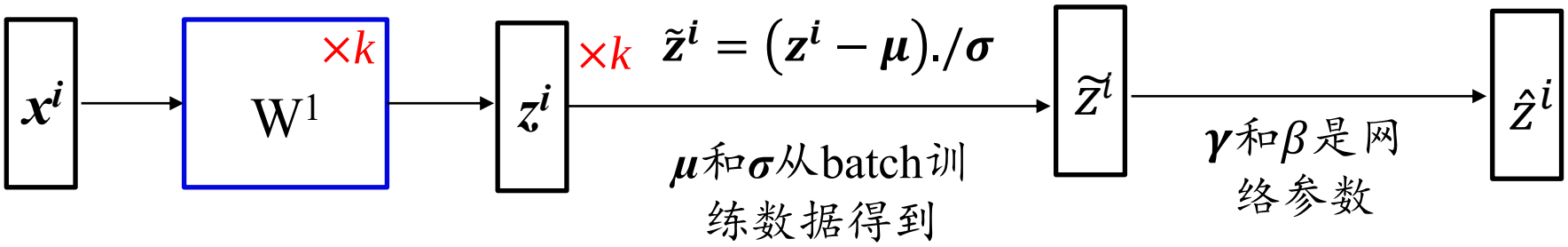
Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.



Batch Normalization

❑ Batch Normalization的优势

- 可以加速训练，使用相对较大的学习率
- 降低梯度消失的风险
  - 特别对于sigmoid这种函数
- 学习过程受初始化影响较小



Batch Normalization

❑ Batch Normalization性能分析

– Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

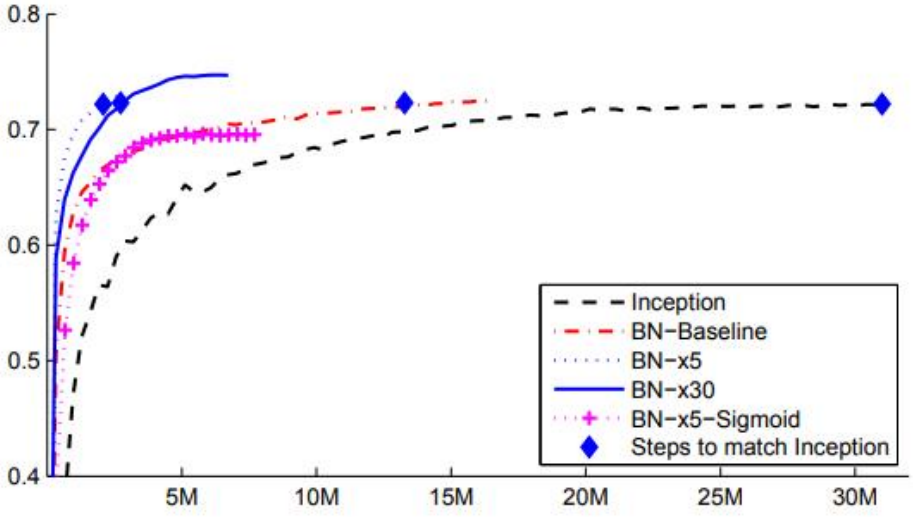


Figure 2: *Single crop validation accuracy of Inception and its batch-normalized variants, vs. the number of training steps.*

Model	Steps to 72.2%	Max accuracy
Inception	$31.0 \cdot 10^6$	72.2%
BN-Baseline	$13.3 \cdot 10^6$	72.7%
BN-x5	$2.1 \cdot 10^6$	73.0%
BN-x30	$2.7 \cdot 10^6$	74.8%
BN-x5-Sigmoid		69.8%

Figure 3: *For Inception and the batch-normalized variants, the number of training steps required to reach the maximum accuracy of Inception (72.2%), and the maximum accuracy achieved by the network.*

Batch Normalization

❑ Batch Normalization性能分析

– Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

Model	Steps to 72.2%	Max accuracy
Inception	$31.0 \cdot 10^6$	72.2%
BN-Baseline	$13.3 \cdot 10^6$	72.7%
BN-x5	$2.1 \cdot 10^6$	73.0%
BN-x30	$2.7 \cdot 10^6$	74.8%
BN-x5-Sigmoid		69.8%

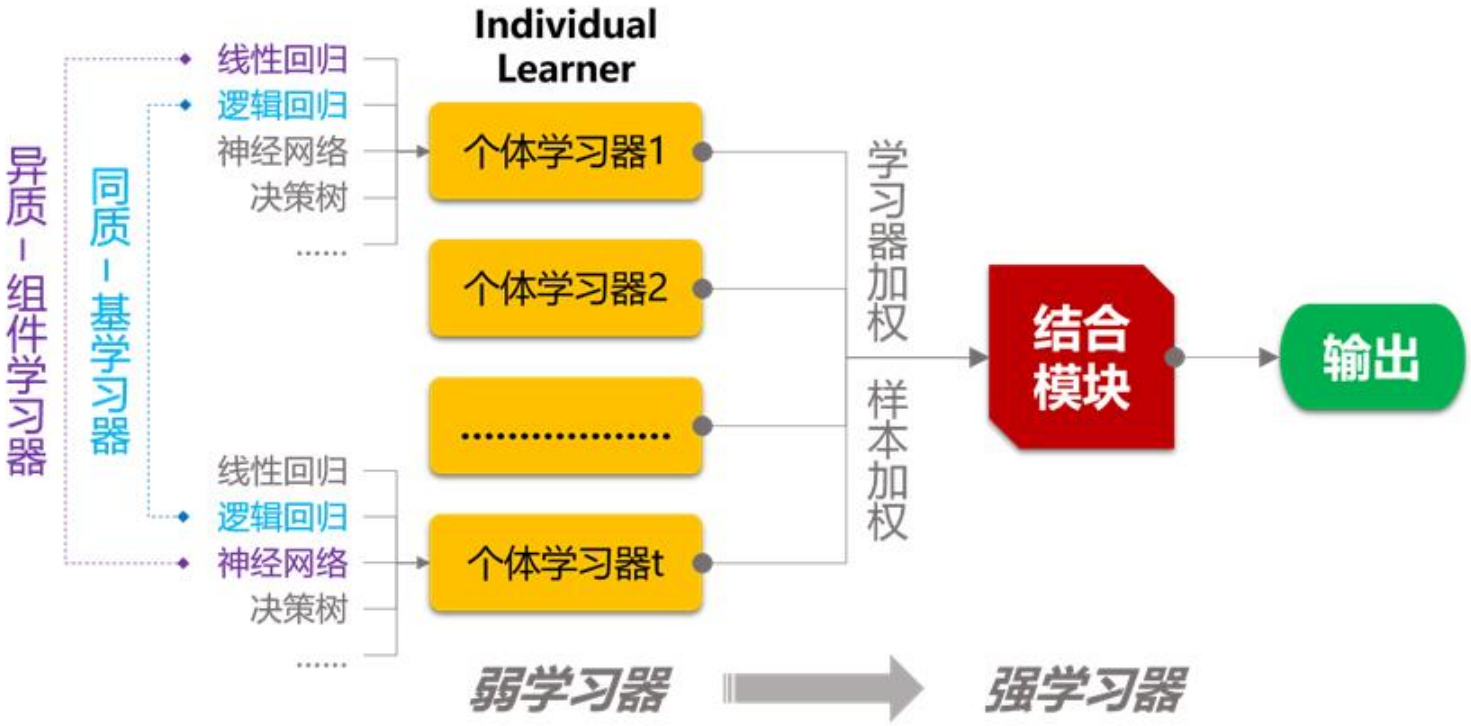
Figure 3: *For Inception and the batch-normalized variants, the number of training steps required to reach the maximum accuracy of Inception (72.2%), and the maximum accuracy achieved by the network.*

Interestingly, increasing the learning rate further (BN-x30) causes the model to train somewhat slower initially, but allows it to reach a higher final accuracy. This phenomenon is counterintuitive and should be investigated further.

5 Bagging

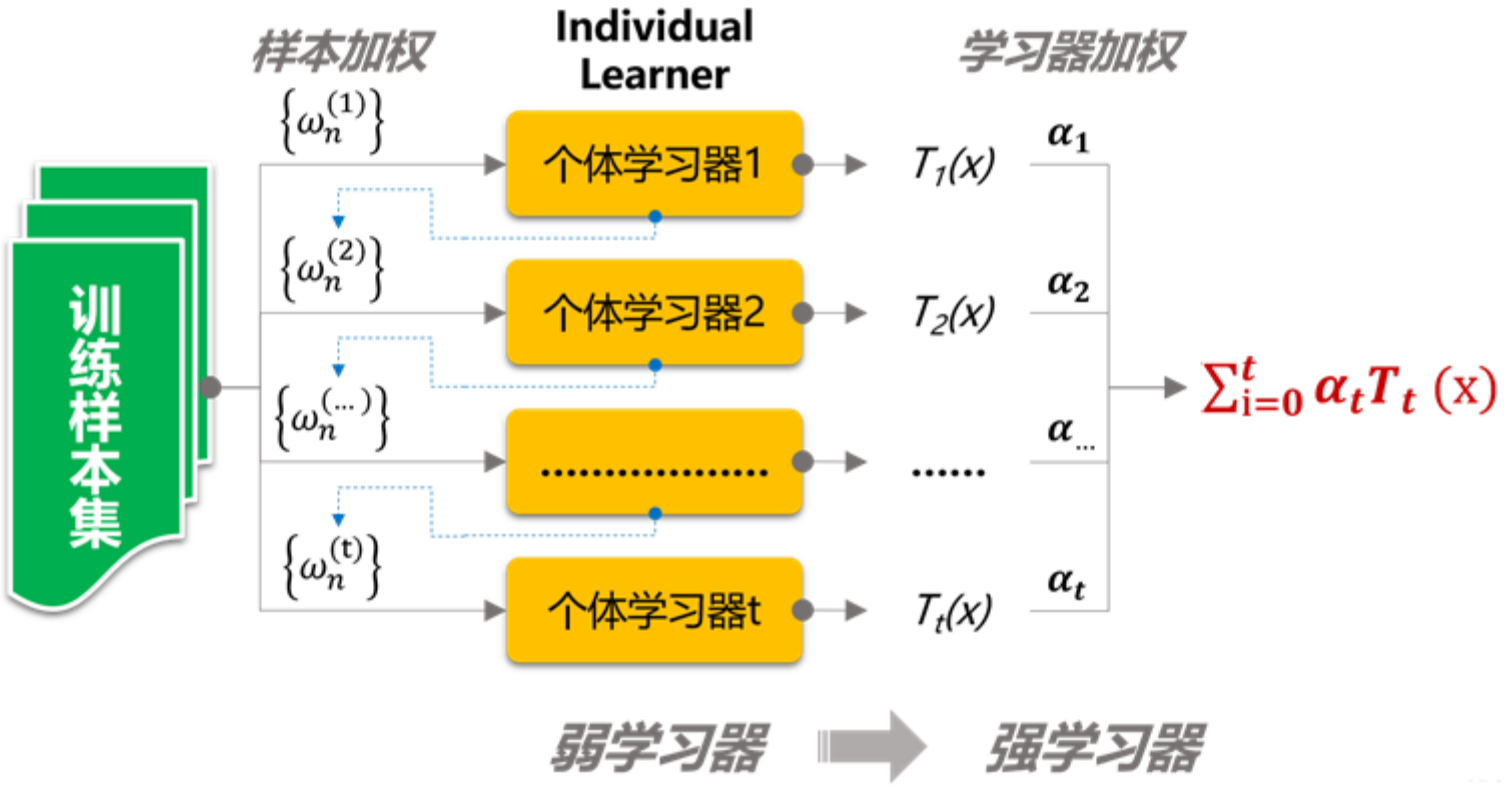
集成学习

- 集成学习将多个弱学习器进行结合，通过对样本加权、学习器加权，获得比单一学习器显著优越的泛化性能的强学习器的学习方法



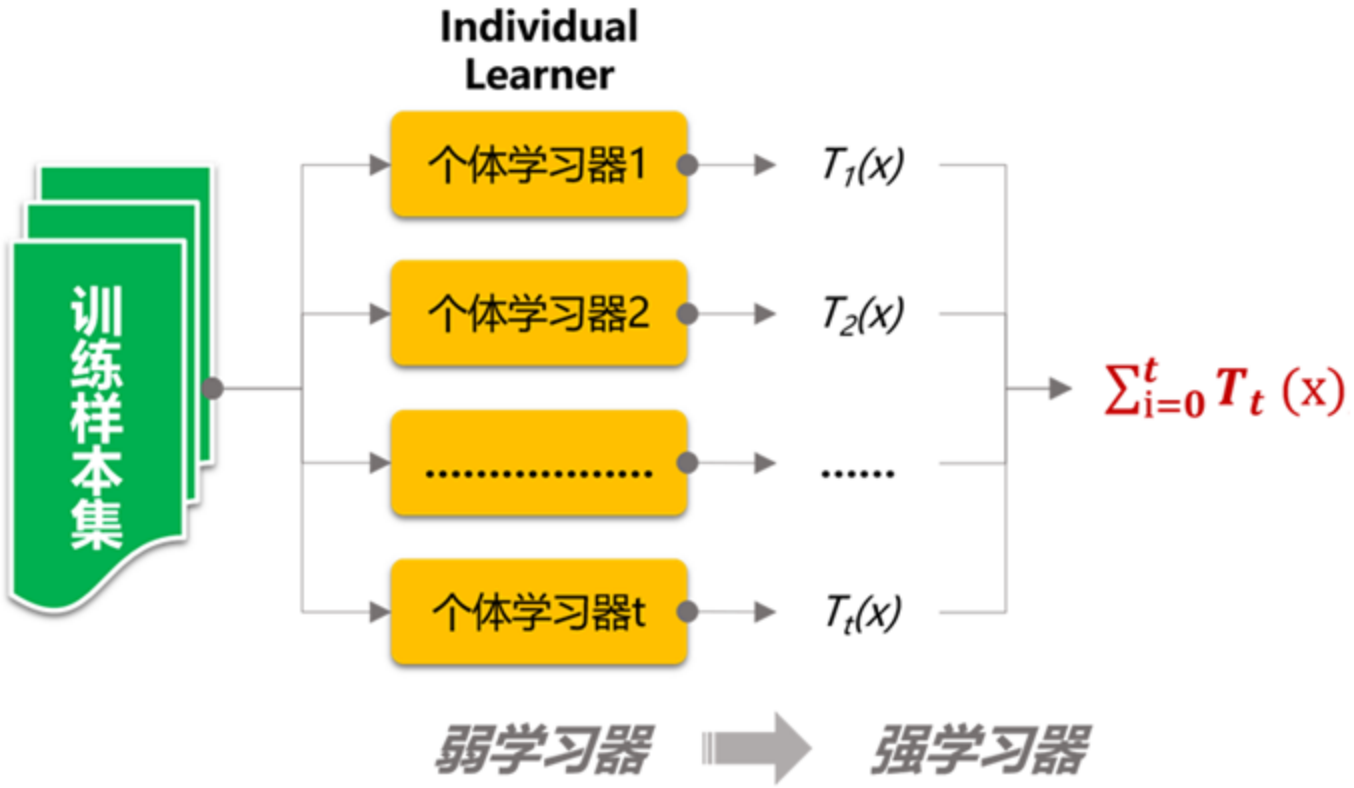
Boosting

- 个体学习器间存在强依赖关系、必须串行生成的序列化方法



Bagging

- 个体学习器间不存在强依赖关系、可同时生成的并行化方法

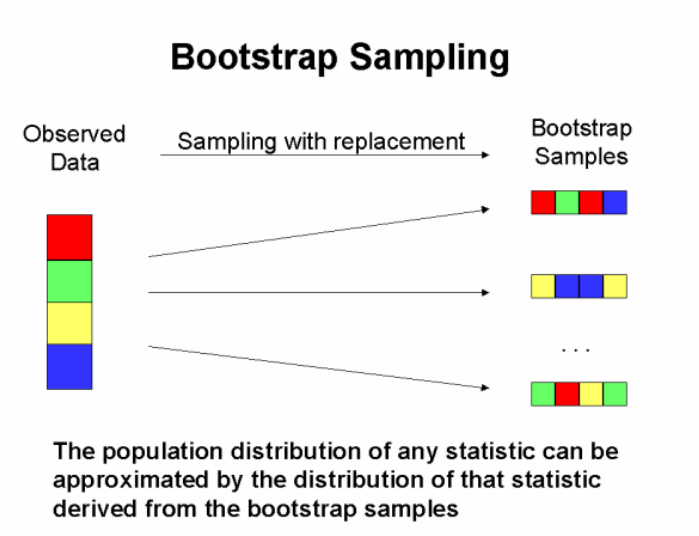


Bagging

- Bagging是一种集成学习算法，设计用来改进统计分类与回归中机器学习算法的稳定性和精确度。它也用来降低方差以帮助避免过拟合（但是不会降低偏差，高偏差模型不适用Bagging）
- Bagging通常应用在决策树算法中，但是它也可以应用到其他算法中。Bagging是模型平均方法的一个特例

Bagging

- **Bagging基于自助采样法(Bootstrap Sampling)，给定包含m个样本的初始训练集S，我们进行m次放回采样，得到包含m个样本的采样集S'**
  - 因此在采样数据集中会缺少一些来自原始训练集中的样本，同时也会包含一些被多次选中重复的样本



Bagging

- 在含m个样本的训练集中，计算一个样本在m次放回采样中始终不被选中的概率：

$$\left(1-\frac{1}{m}\right)^m$$

$$\lim_{m\mapsto\infty}\left(1-\frac{1}{m}\right)^m \mapsto \frac{1}{e}\approx 0.368$$

- 通过自助采样法，数据集中**约有36.8%的数据不会被选中到采样集中**

Bagging

- 基于自助采样法构造k个含有m个训练样本的采样集，并在每个采样集上进行训练得到k个模型
- 每个采样集之间所含样本的差异会导致k个模型之间存在差异，在测试集上产生的误差就不同，而这也是Bagging方法能够起作用的原因
- 如果是分类算法，则k个模型投出最多票数的类别或者类别之一为最终类别。如果是回归算法，对k个模型得到的回归结果进行算术平均得到的值为最终的模型输出

Bagging分析

- 假设构造的第i个模型在每个样本上的误差是ϵ_i，并且ϵ_i服从均值为零、方差为$\mathbb{E}\left[\epsilon_i^2\right] = v$且协方差为$\mathbb{E}\left[\epsilon_i, \epsilon_j\right] = c$的多维正态分布
- 所有模型的平均预测误差的平方是

$$E\left[\left(\frac{1}{k}\sum_i \epsilon_i\right)^2\right] = \frac{1}{k^2}E\left[\sum_i\left(\epsilon_i^2 + \sum_{j\neq i}\epsilon_i\epsilon_j\right)\right]$$

$$= \frac{1}{k}v + \frac{k-1}{k}c$$

Bagging分析

- 如果误差完全相关（$c=v$），均方误差减少到v，因此模型平均没有任何帮助
- 在误差完全不相关（$c=0$）的情况下，该集成平方误差的期望即为v/k
- 因此，集成平方误差的期望会随着集成规模增大而线性减小。集成至少会和它的任何成员表现得一样好，并且如果成员的误差是独立的，集成将显著地比其成员表现得更好

Bagging分析

- 神经网络能找到足够多的不同的解，意味着他们可以从模型平均中受益(即使所有模型都在同一数据集上训练)
- 神经网络中随机初始化的差异、小批量的随机选择、超参数的差异或不同输出的非确定性实现往往足以使得集成中的不同成员具有部分独立的误差

6 Dropout和DropConnect

Dropout的提出

- **Dropout由Hinton在2012年提出。它计算简单、功能强大，大多数神经网络模型都可以使用**
  - Hinton在其论文《Improving neural networks by preventing co-adaptation of feature detectors》中提出Dropout (谷歌学术引用4840次)
  - Alex、Hinton在其论文《ImageNet Classification with Deep Convolutional Neural Networks》中用到了Dropout算法(这篇论文提出的AlexNet网络模型赢得了2012年图像识别大赛冠军, 谷歌学术引用60237次)
- **目前为止，大多数的深度神经网络都使用了Dropout正则化方法**
  - Dropout:A Simple Way to Prevent Neural Networks from Overfitting
  - Improving Neural Networks with Dropout
  - Dropout as data augmentation

[1] Hinton, Geoffrey E., Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. "Improving neural networks by preventing co-adaptation of feature detectors." arXiv preprint arXiv:1207.0580 (2012).

[2] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." In Advances in neural information processing systems, pp. 1097-1105. 2012.

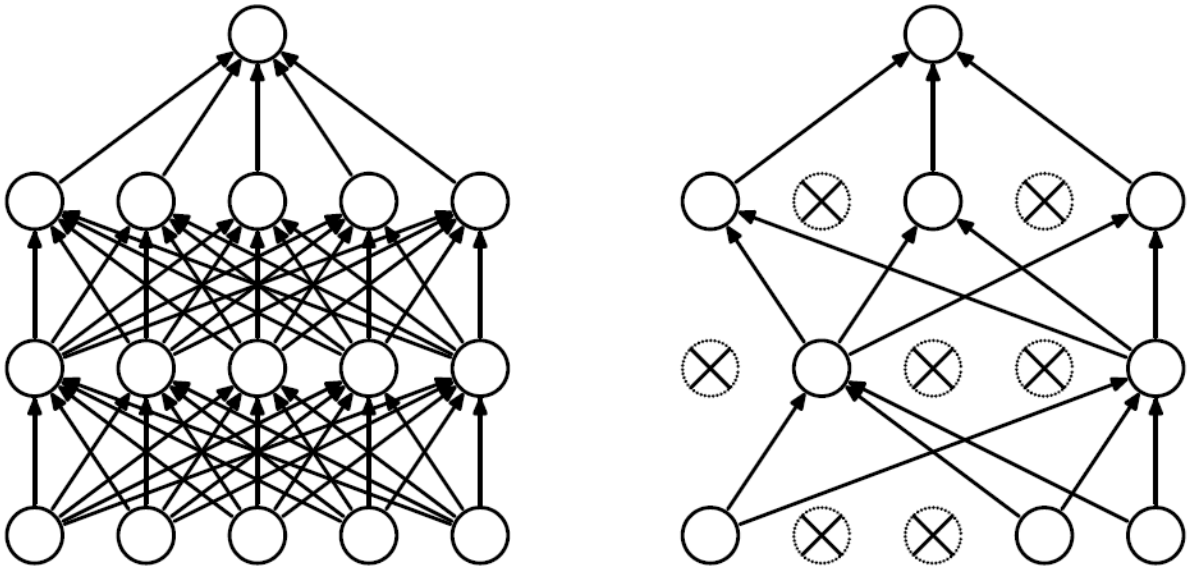
Dropout的工作过程

- 训练深度神经网络的一种技巧（Trick），在每个训练批次中，删掉部分的神经元节点，即在网络前向传播的时候，让某些神经元的激活值以一定的概率p输出为0
  - 输入层和隐藏层都可以dropout一些神经元
  - 隐藏层概率通常取0.5
  - 输入层dropout概率通常取0.2
- 这种方式可以减少层间节点的相互作用，因此可以明显地缓和过拟合现象
- 使模型泛化性更强，因为它不会太依赖某些局部的特征

Dropout的工作过程

❑ Dropout改变网络结构

– 相当于训练一个简化的网络” thinned network”



(a) Standard Neural Net　　(b) After applying dropout.

Dropout的实现

❑ 训练过程

- 将网络中的隐层神经元的激活值（输出）以概率p随机置0（每次迭代激活值置零的神经元不同）
- 对网络进行正向、反向传播，即训练网络，反向传播时激活值被置0神经元的参数保持没有被删除前的值，其他神经元的参数被更新
- 重复这一过程，直至训练结束

Dropout的实现

❑ 训练过程

– 标准的网络结构和Dropout网络训练过程中前向计算

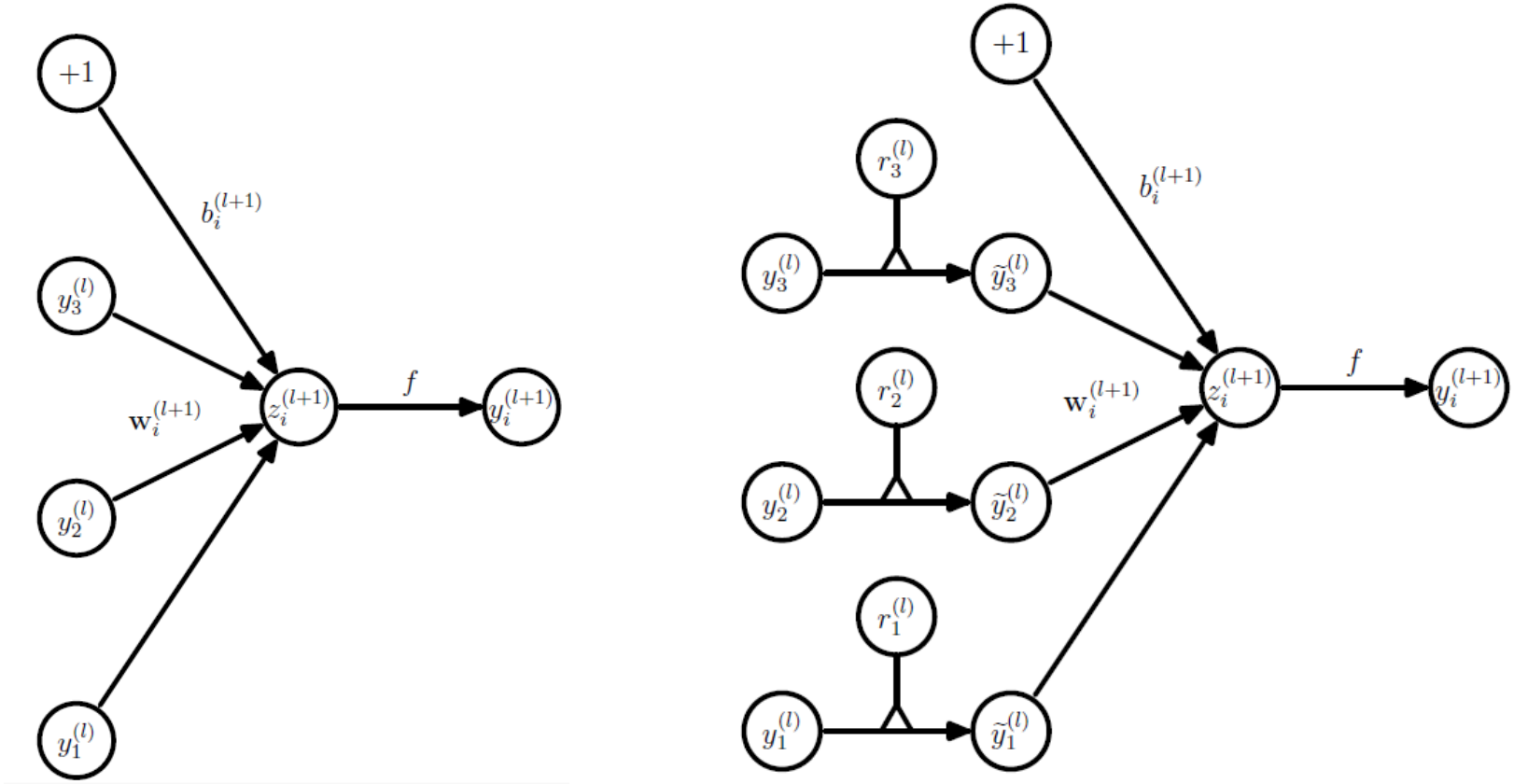
$$
\begin{aligned}
z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)}\mathbf{y}^l + b_i^{(l+1)}, \\
y_i^{(l+1)} &= f(z_i^{(l+1)}),
\end{aligned}
$$

$$
\begin{aligned}
r_j^{(l)} &\sim \text{Bernoulli}(p), \\
\tilde{\mathbf{y}}^{(l)} &= \mathbf{r}^{(l)} * \mathbf{y}^{(l)}, \\
z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)}\tilde{\mathbf{y}}^l + b_i^{(l+1)}, \\
y_i^{(l+1)} &= f(z_i^{(l+1)}).
\end{aligned}
$$

Dropout的实现

❑ 训练过程

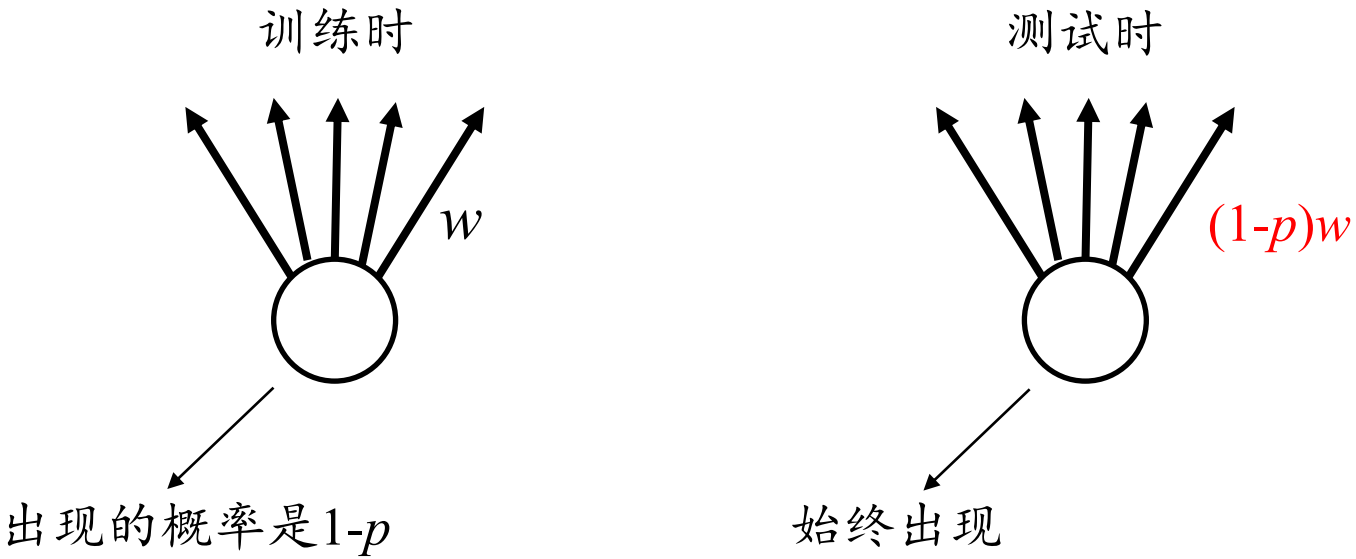
– 标准的网络结构和Dropout网络结构计算过程对比



Dropout的实现

❑ 测试阶段

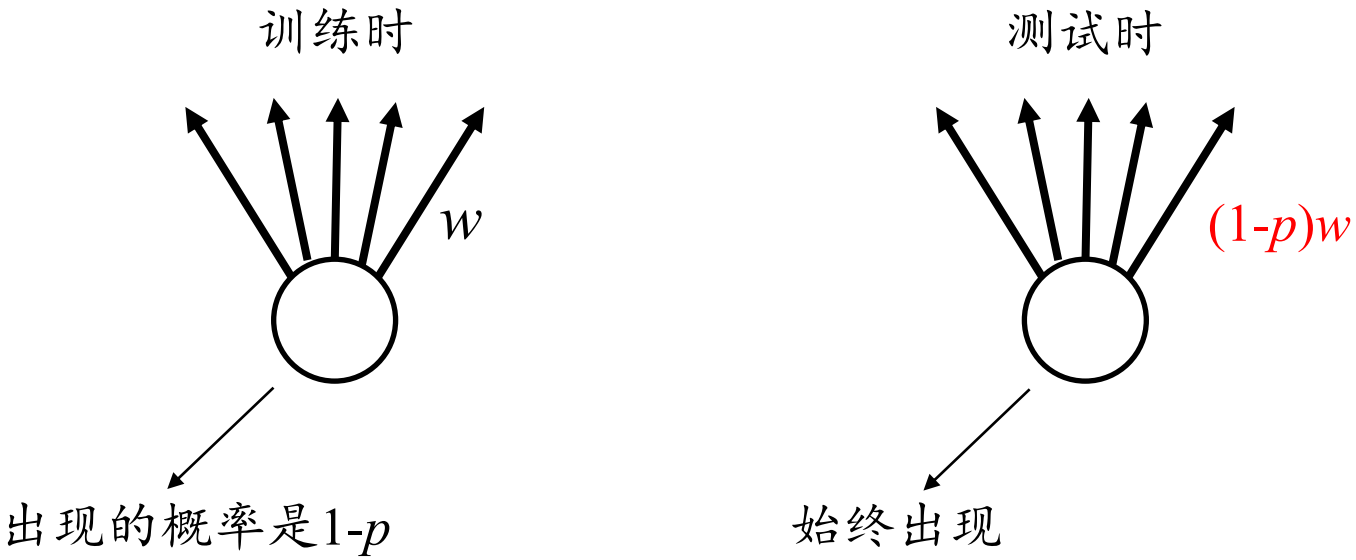
- 测试时使用所有神经元；
- 所有神经元的激活值需要乘以1-p;



Dropout的实现

❑ 测试阶段

- 测试时使用所有神经元；
- 所有神经元的激活值需要乘以1-p;

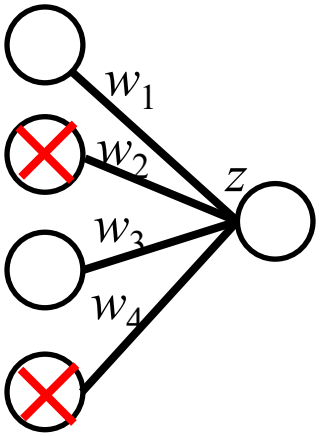


Dropout的实现

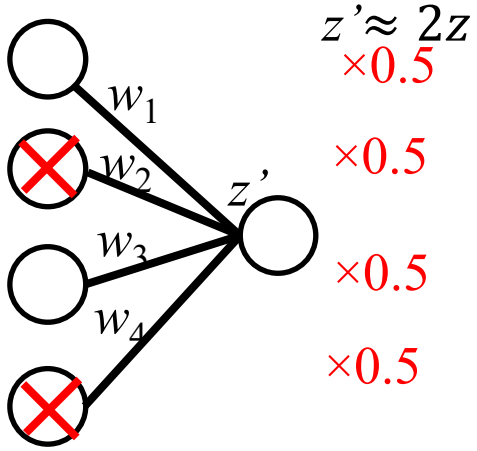
❑ 测试阶段

- 测试时使用所有神经元；
- 所有神经元的激活值需要乘以1-*p*;

训练时，假设dropout rate 50%



测试时



Dropout的解释

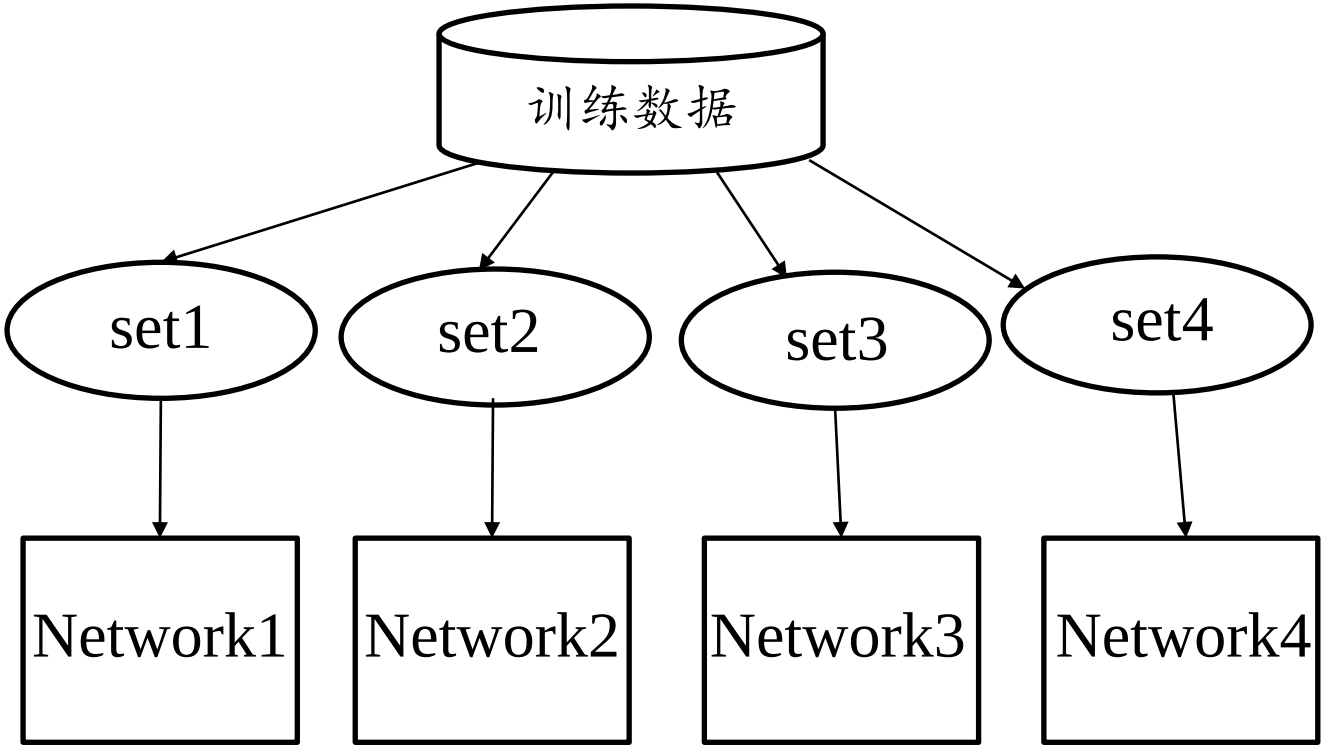
- 有效性的解释
  - 不同的解释角度：数据增广，稀疏约束...
  - 集成学习(Ensemble learning)
- Dropout可以视作一种高效的近似Bagging的集成方法，并且是共享隐层单元的集成方法。可以理解为每次训练随机丢弃节点值，使得模型前后存在差异

Dropout的解释

- Dropout使得隐层节点不一定每次都出现，减少了隐层节点之间的复杂的共适应关系
- 这样权值的更新不再依赖于有固定关系的隐层节点的共同作用，阻止了某些特征仅仅在特定特征下才有效果的情况，迫使网络去学习更加鲁棒的特征

Dropout的解释

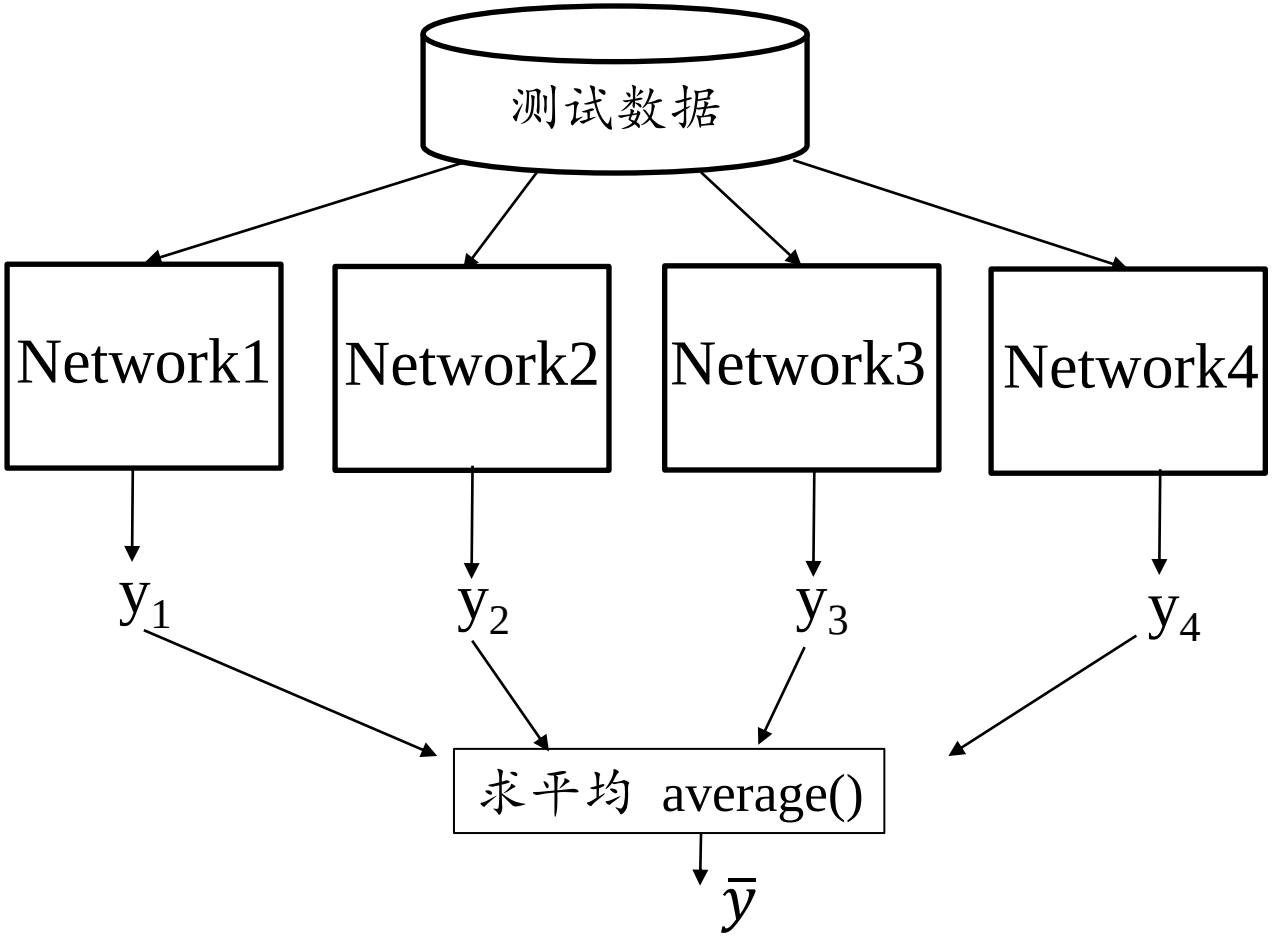
❑ Dropout的近似Bagging集成方法解释



训练多个不同的结构的网络

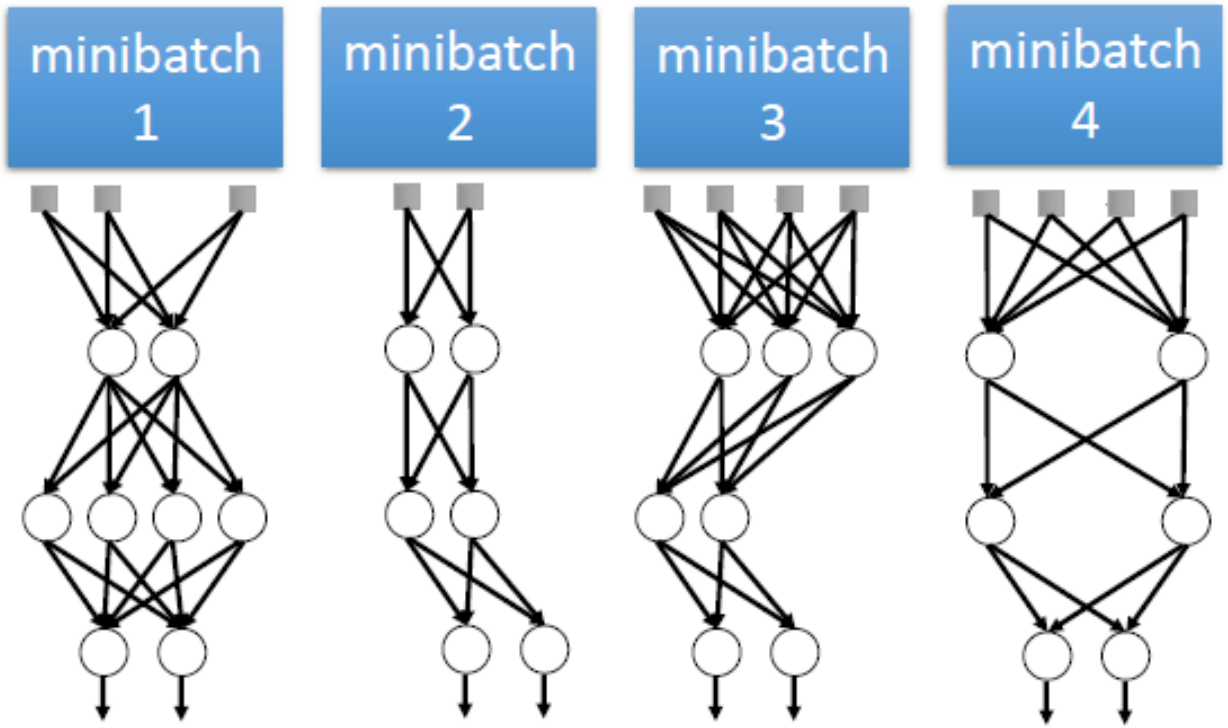
Dropout的解释

- Dropout的近似Bagging集成方法解释



Dropout的解释

❑ Dropout的近似Bagging集成方法解释

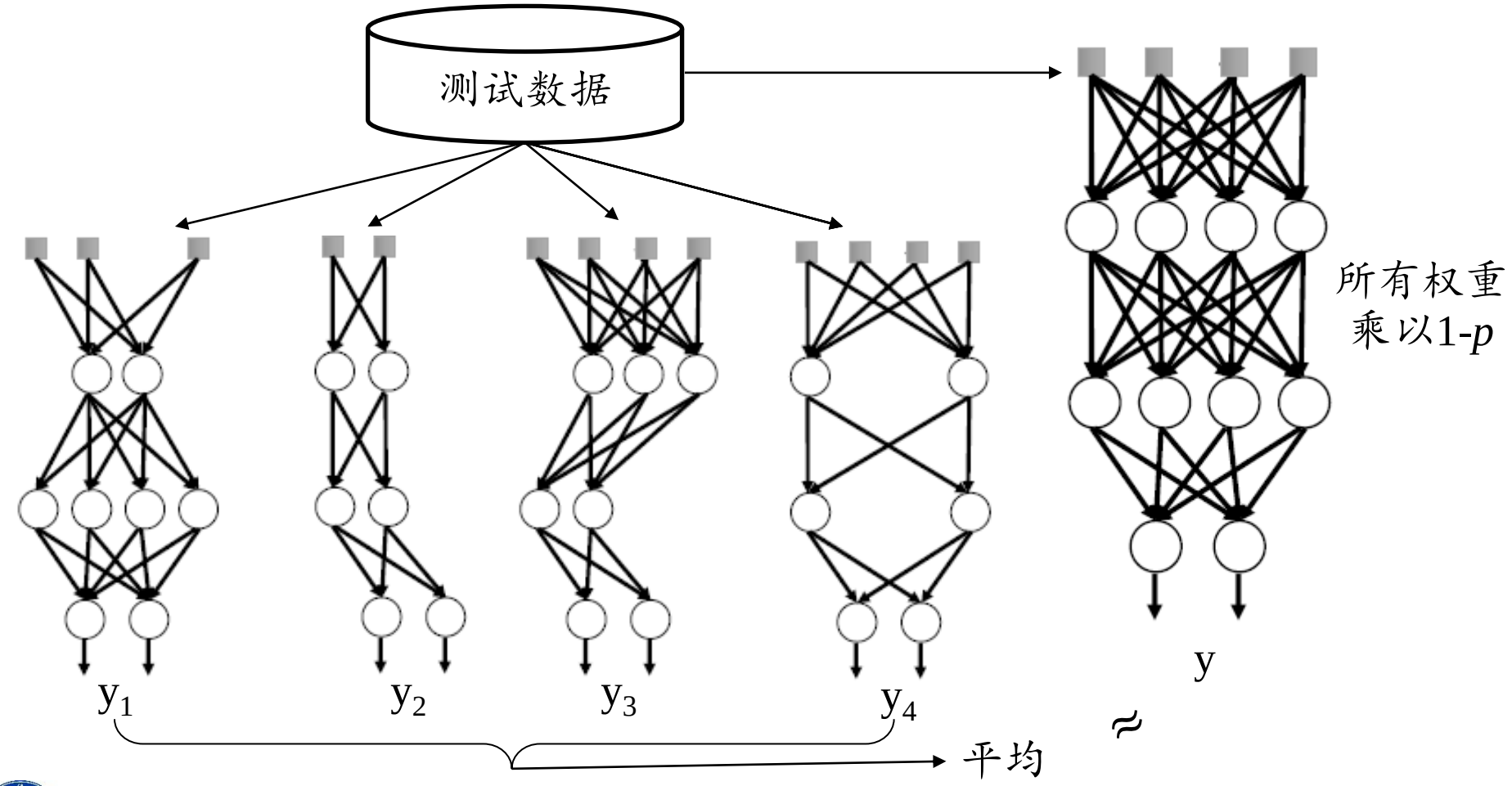


*M*个神经元对应有2^M可能的子网络

- 每个minibatch训练一个子网络
- 不同子网络间神经元参数共享

Dropout的解释

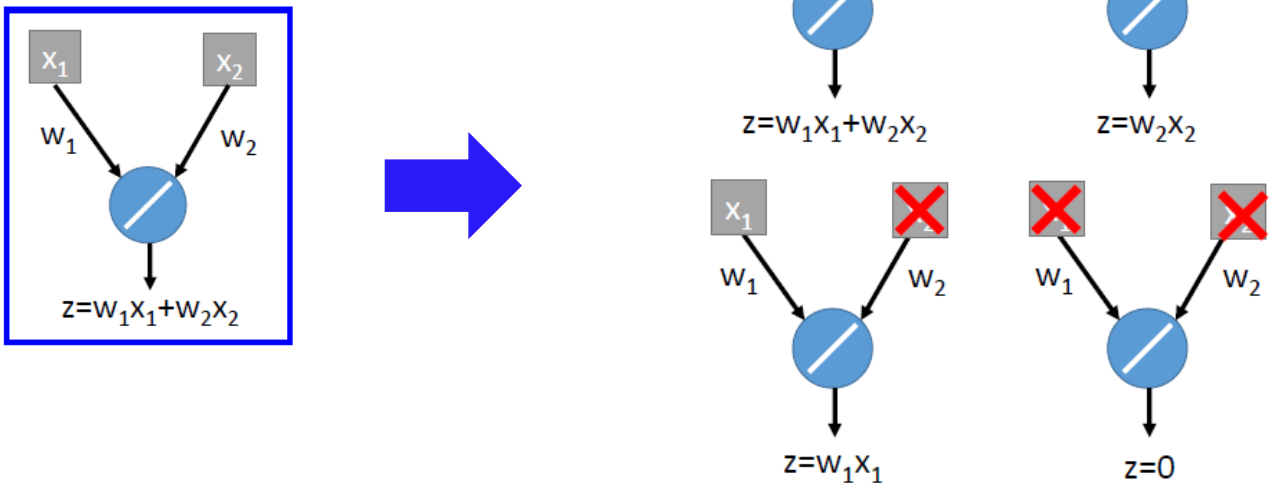
❑ Dropout的近似Bagging集成方法解释



Dropout的解释

❑ Dropout的近似Bagging集成方法解释

– 在神经网络接近linear时dropout性能会好，比如ReLU，Maxout激活函数



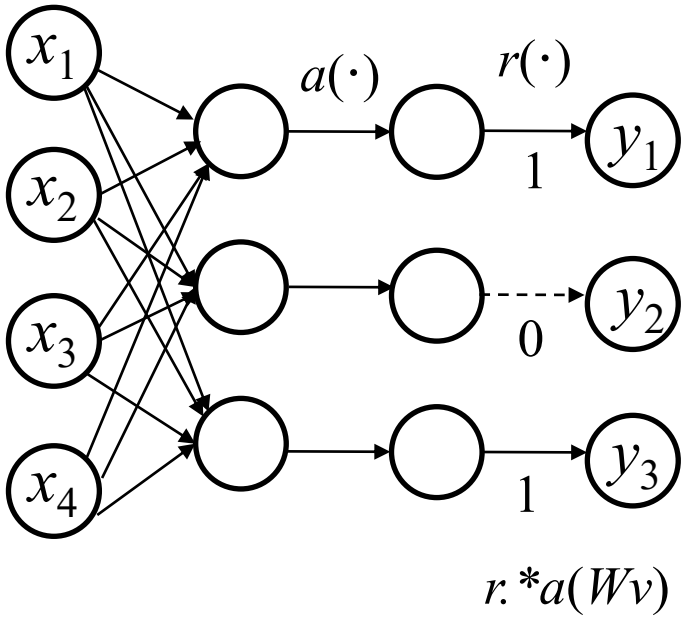
DropConnect

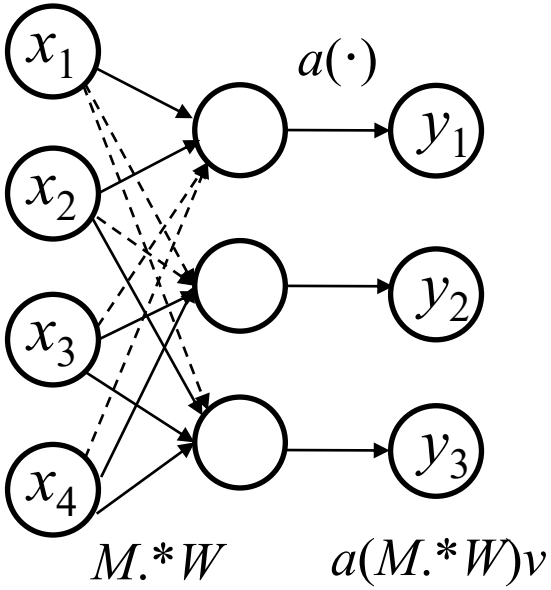
- **DropConnect也是通过舍弃一部分单元来防止过拟合问题的方法**
  - DropConnect可以认为是Dropout的扩展。它是丢掉神经元和神经元之间的连接，包含权重和偏置，即不需要进行激活操作前的线性变换
  - 它比Dropout效率更高且性能得到提升，但两者还是比较相似，差别不大

Wan, Li, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. "Regularization of neural networks using dropconnect." In International conference on machine learning, pp. 1058-1066. 2013.

DropConnect

❑ DropConnect与Dropout的异同





DropConnect

- 使用DropConnect和Dropout在不同数据集上的测试误差
  - 虽然DropConnect的识别性能优于Dropout，但是其训练难度也高于Dropout

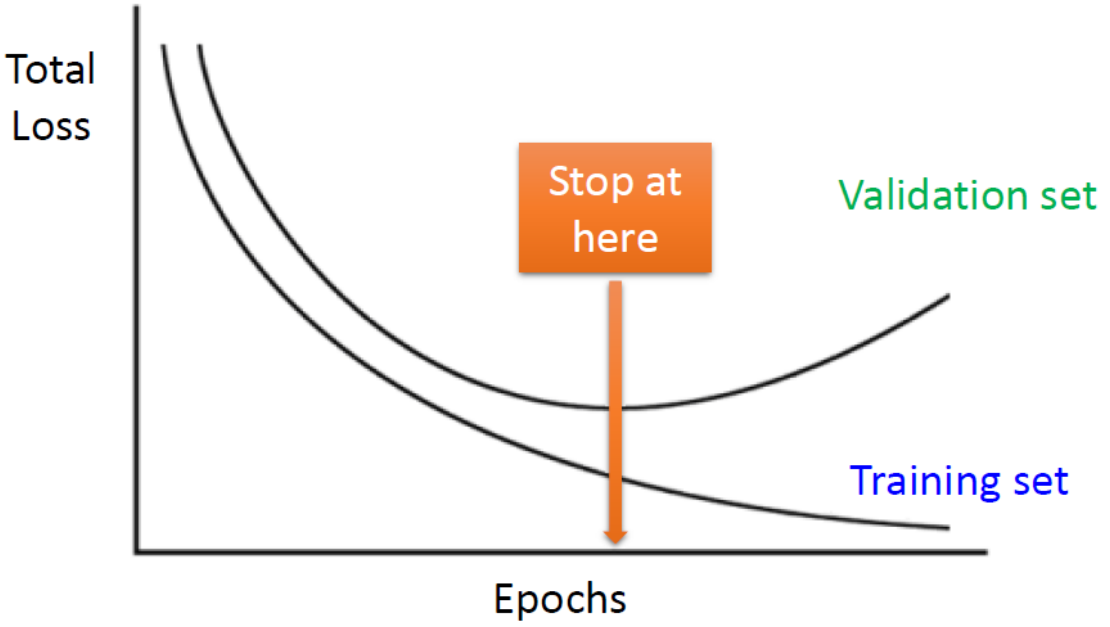
数据集	Dropout	DropConnect
MNIST	0.27	**0.21**
CIFAR-10	9.83	**9.32**
SVHN	1.96	**1.94**
NORB	**3.03**	3.23

7

提前终止

提前终止

- 随着训练迭代次数的增加，训练误差也在逐步降低。但是会发现验证集损失先下降后在某个节点开始上升的现象
  - 提前终止（Early stopping）目的则在于找到这个上升的时间点，并及时终止模型的训练
  - 提前终止是深度学习中最常用的正则化方法，可单独使用或与其他正则化策略结合使用



提前终止实现

- 需要尽量避免训练波动导致的提前终止

提前终止算法

令n为评估间隔的步数，在验证集上连续出现p次较坏结果后终止，设θ_0为初始化的参数

$\theta \leftarrow \theta_0,\ \theta^* \leftarrow \theta$

$i \leftarrow 0,\ j \leftarrow 0$

$v \leftarrow \infty$

$i^* \leftarrow i$

while $j < p$ do

  执行n步训练，同时对θ进行更新

  $i \leftarrow i + n$

  $v' \leftarrow \text{ValidationSetError}(\theta)$

  if $v' < v$ then

    $j \leftarrow 0$

    $\theta^* \leftarrow \theta$

    $i^* \leftarrow i$

    $v \leftarrow v'$

  else

    $j \leftarrow j + 1$

  end if

end while

得到最佳参数θ^*，最佳训练步数i^*

提前终止算法：额外训练

- 使用提前终止策略过程中需要使用到验证集（Validation Set），这也就意味着在训练过程中会有一部分数据无法进行训练。
- 数据宝贵，为了更好地利用验证集中的数据，我们可以在完成提前终止后将验证集中的数据加入到训练集中，进行额外的训练

提前终止算法：额外训练

❑ 策略一

- 在进行额外训练前，已经通过提前终止确定了训练的最佳步数。使用所有的数据进行训练时，在确定的最佳步数时终止训练。
- 但是值得指出的是，我们不能确定按照提前终止确定的训练最佳步数再次训练时仍能得到一个最佳的训练，因为在进行额外训练时训练集变大了，因此在训练过程中对参数的更新次数也会变多

策略 1 利用提前终止确定的最佳训练步数在所有数据上再次训练

已有$(X^{all_train}, y^{all_train})$为全部训练集

将$(X^{all_train}, y^{all_train})$划分为$(X^{train}, y^{train})$和$(X^{val}, y^{val})$

随机初始化参数θ，使用(X^{train}, y^{train})作为训练集，(X^{val}, y^{val})作为验证集进行训练，利用提前终止返回最佳训练步数t

再次随机初始化参数θ，使用$(X^{all_train}, y^{all_train})$作为训练集上训练$t$步

提前终止算法：额外训练

❑ 策略二

- 在提前终止时得到的模型的基础上再次利用所有的训练集进行训练，在这种情况下我们已经没有验证集可以供我们判断何时终止训练，但是我们可以通过监控验证集的平均损失函数，直到它低于之前提前终止时在训练集上的目标损失值
- 这个策略相较于上一个策略避免了重新训练模型，但是这种策略甚至无法保证继续训练是否能达到之前的目标值

策略 2 利用提前终止确定将会过拟合的目标损失值，然后在所有数据上再次训练直到再次到达该值

已有$(X^{all_train}, y^{all_train})$为全部训练集

将$(X^{all_train}, y^{all_train})$划分为$(X^{train}, y^{train})$和$(X^{val}, y^{val})$

随机初始化参数θ，使用(X^{train}, y^{train})作为训练集，(X^{val}, y^{val})作为验证集进行训练，利用提前终止确定目标损失值ϵ

$\epsilon \leftarrow J(\theta, X^{train}, y^{train})$

在$(X^{all_train}, y^{all_train})$继续训练，直到$J(\theta, X^{val}, y^{val}) \leq \epsilon$

8 中英文术语对照

中英文术语对照

- 偏差：Bias
- 方差：Variance
- 泛化误差：Generalization error
- Lp范数：Lp Norm
- Frobenius范数： Frobenius norm
- 正则化：Regularization
- 岭回归：Ridge Regression
- 权重衰减：Weight decay
- 数据增强：Data Augmentation
- 模型平均：Model averaging
- 自助采样法：Bootstrap Sampling

中英文术语对照

- 提前终止：Early Stopping
- 稀疏表示：Sparse Representation
- 字典学习：Dictionary Learning

谢谢！