2019春

模式识别与深度学习 (14)

多层感知器算法-2

左旺孟

综合楼309

视觉感知与认知组

哈尔滨工业大学计算机学院

cswmzuo@gmail.com

13134506692

主要内容

- **随机优化算法**（Stochastic Optimization）
  - 随机优化
  - SGD、AdaGrad、RMSProp、ADAM
- 网络模型：3层 -> 多层
  - 激活函数：ReLU / PReLU, …
  - 网络参数：Dropout
- 泛化能力

常见的损失函数

- 回归问题
  - 均方差损失

$$f^* = \arg\min_f \mathbb{E}_{\mathbf{x},\mathbf{y}\sim p_{\text{data}}} ||\boldsymbol{y} - f(\boldsymbol{x})||^2$$

  - L1损失

$$f^* = \arg\min_f \mathbb{E}_{\mathbf{x},\mathbf{y}\sim p_{\text{data}}} ||\boldsymbol{y} - f(\boldsymbol{x})||_1$$

常见的损失函数

- 分类问题
  - 两类问题

$$z = \boldsymbol{w}^\top \boldsymbol{h} + b \qquad f(x) = \frac{1}{1+e^{-x}}$$

$$\begin{aligned} J(\boldsymbol{\theta}) &= -\log P(y \mid \boldsymbol{x}) \\ &= -\log \sigma((2y-1)z) \\ &= \zeta((1-2y)z). \end{aligned}$$

  - 多类问题

$$\boldsymbol{z} = \boldsymbol{W}^\top \boldsymbol{h} + \boldsymbol{b} \qquad \text{softmax}(\boldsymbol{z})_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

$$z_i - \log \sum_j \exp(z_j)$$

随机优化（ Stochastic Optimization ）

- 随机优化 & SGD

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x},\mathrm{y}\sim\hat{p}_{\mathrm{data}}} L(\boldsymbol{x}, y, \boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$$

  - $L(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2}\| \boldsymbol{x} - \boldsymbol{y} \|^2$

- 梯度下降

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$$

- 随机梯度下降（SGD）（小批量）

$$\boldsymbol{g} = \frac{1}{m'}\nabla_{\boldsymbol{\theta}} \sum_{i=1}^{m'} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}) \qquad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \boldsymbol{g},$$

随机优化

- 最大均值差异（ Maximum Mean Discrepancy， MMD）

$$\mathrm{MMD}^2(s,t) = \sup_{\|\phi\|_{\mathcal{H}}\le 1} \left\| E_{\mathbf{x}^s\sim s}\left[\phi(\mathbf{x}^s)\right] - E_{\mathbf{x}^t\sim t}\left[\phi(\mathbf{x}^t)\right] \right\|_{\mathcal{H}}^2$$

$$\mathrm{MMD}^2(\mathcal{D}_s,\mathcal{D}_t) = \left\| \frac{1}{M}\sum_{i=1}^{M}\phi(\mathbf{x}_i^s) - \frac{1}{N}\sum_{j=1}^{N}\phi(\mathbf{x}_j^t) \right\|_{\mathcal{H}}^2$$

A. Gretton, K. Borgwardt, M. Rasch, B. Scholkopf, and A. Smola. A kernel method for the two-sample problem. In Advances in Neural Information Processing Systems 15, pages 513–520, Cambridge, MA, 2007a. MIT Press.

随机优化

$$\begin{aligned}\mathrm{MMD}^2[\mathcal{F},p,q] &:= \langle \mu_p-\mu_q, \mu_p-\mu_q\rangle_{\mathcal{H}} \\ &= \langle \mu_p, \mu_p\rangle_{\mathcal{H}} + \langle \mu_q, \mu_q\rangle_{\mathcal{H}} - 2\langle \mu_p, \mu_q\rangle_{\mathcal{H}} \\ &= \mathbf{E}_p\langle \phi(x), \phi(x')\rangle_{\mathcal{H}} + \mathbf{E}_q\langle \phi(y), \phi(y')\rangle_{\mathcal{H}} \\ &\quad - 2\mathbf{E}_{p,q}\langle \phi(x), \phi(y)\rangle_{\mathcal{H}},\end{aligned}$$

- Mini-batch计算（4元组Batch）$\{\mathbf{x}_i, \mathbf{x}'_i; \mathbf{y}_i, \mathbf{y}'_i\}$

$$\boldsymbol{g} = \frac{1}{m'}\nabla_{\boldsymbol{\theta}}\sum_{i=1}^{m'} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}) \Rightarrow \frac{1}{m'}\sum_{i=1}^{m'}\frac{\partial\langle\phi(\mathbf{x}_i), \phi(\mathbf{x}'_i)\rangle}{\partial\theta} + \frac{\partial\langle\phi(\mathbf{y}_i), \phi(\mathbf{y}'_i)\rangle}{\partial\theta} - 2\frac{\partial\langle\phi(\mathbf{x}_i), \phi(\mathbf{y}_i)\rangle}{\partial\theta}$$

随机优化

- 随机ADMM（ Alternating Direction Method of Multipliers ）

$$\min_{\mathbf{x}\in\mathcal{X},\mathbf{y}\in\mathcal{Y}} \mathbb{E}_{\boldsymbol{\xi}}\theta_1(\mathbf{x},\boldsymbol{\xi}) + \theta_2(\mathbf{y}) \text{ s.t. } A\mathbf{x} + B\mathbf{y} = \mathbf{b},$$

$$\min_{\mathbf{x}\in\mathcal{X},\mathbf{y}\in\mathcal{Y}} \mathcal{L}_\beta(\mathbf{x},\mathbf{y},\lambda) \equiv \min_{\mathbf{x}\in\mathcal{X},\mathbf{y}\in\mathcal{Y}} \Big[\theta_1(\mathbf{x}) + \theta_2(\mathbf{y}) - \langle\boldsymbol{\lambda}, A\mathbf{x} + B\mathbf{y} - \mathbf{b}\rangle + \frac{\beta}{2}\|A\mathbf{x} + B\mathbf{y} - \mathbf{b}\|^2\Big],$$

$$\boxed{\theta_1(\mathbf{x},\boldsymbol{\xi}_{k+1})}$$

$$\hat{\mathcal{L}}_{\beta,k}(\mathbf{x},\mathbf{y},\boldsymbol{\lambda}) \equiv \boxed{\theta_1(\mathbf{x}_k) + \langle\theta_1'(\mathbf{x}_k,\boldsymbol{\xi}_{k+1}),\mathbf{x}\rangle} + \theta_2(\mathbf{y}) - \langle\boldsymbol{\lambda}, A\mathbf{x} + B\mathbf{y} - \mathbf{b}\rangle + \frac{\beta}{2}\|A\mathbf{x} + B\mathbf{y} - \mathbf{b}\|^2 + \boxed{\frac{\|\mathbf{x}-\mathbf{x}_k\|^2}{2\eta_{k+1}}}$$

Stochastic Alternating Direction Method of Multipliers, ICML 2013

随机优化

- Batch
  - 样本Batch、样本对Batch、（三）四元组Batch
- 学习算法
  - SGD（当前深度学习中的主流方法）
  - 其他优化算法如S-ADMM（可否用于深度网络训练？）

Training Neural Networks Without Gradients: A Scalable ADMM Approach, ICML 2016

随机梯度下降（SGD）

算法 8.1 随机梯度下降（SGD）在第 k 个训练迭代的更新

Require: 学习率 ϵ_k

Require: 初始参数 $\boldsymbol{\theta}$

while 停止准则未满足 **do**

从训练集中采包含 m 个样本 $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ 的小批量，其中 $\boldsymbol{x}^{(i)}$ 对应目标为 $\boldsymbol{y}^{(i)}$。

计算梯度估计：$\hat{\boldsymbol{g}} \leftarrow +\frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$

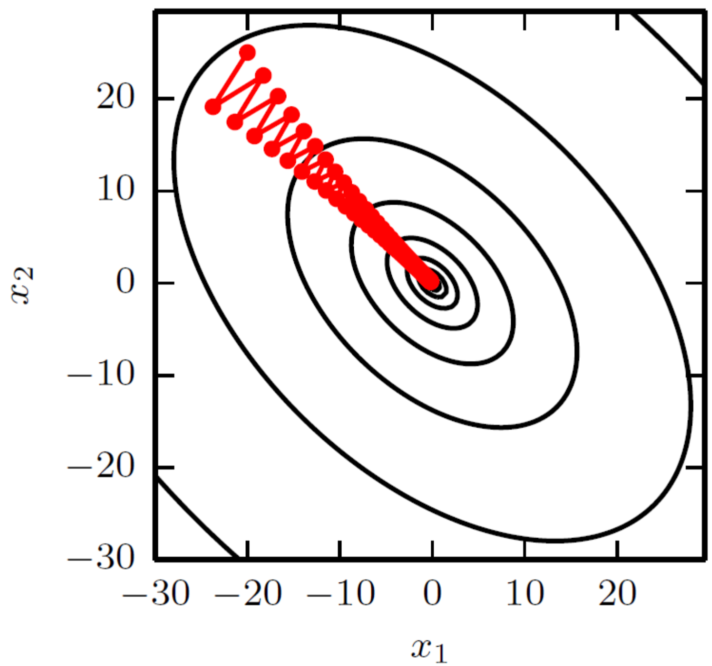
应用更新：$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \hat{\boldsymbol{g}}$

end while

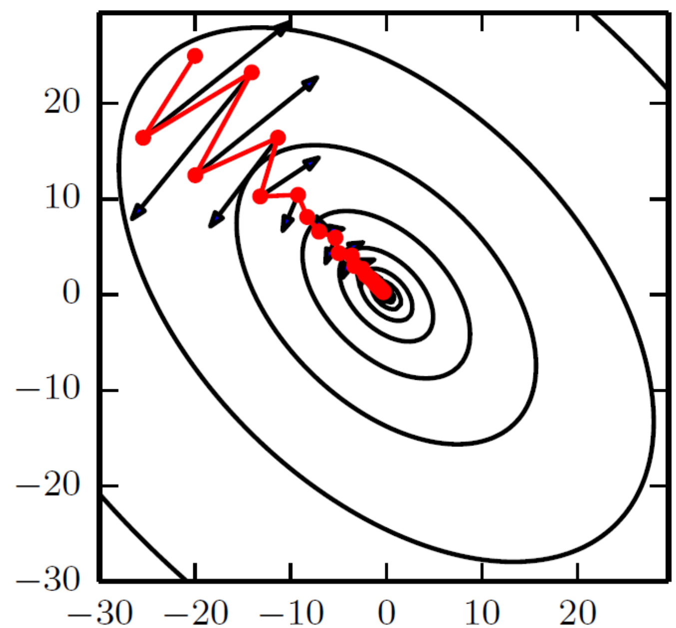
- 学习率：$\sum_{k=1}^{\infty} \epsilon_k = \infty, \quad \sum_{k=1}^{\infty} \epsilon_k^2 < \infty$
- 小批量

动量方法（SGD-Momentum）

- 梯度下降的局限性：病态条件的二次目标函数



SGD



SGD-Momentum

动量方法（SGD-Momentum）

- 动量（物理）：质量x速度
- 神经网络学习
  - 单位质量
  - 速度v

$$\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \epsilon \nabla_{\boldsymbol{\theta}} \left(\frac{1}{m} \sum_{i=1}^{m} L(\boldsymbol{f}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)}) \right)$$

  - α: 一般取值为0.5，0.9 和0.99

Nesterov加速梯度

- 凸优化问题：

$$\mathbf{x}_k = \mathbf{y}_{k-1} - \varepsilon \nabla F(\mathbf{y}_{k-1}) \quad (\varepsilon \le 1/L_F)$$

$$t_{k+1} \leftarrow (1 + \sqrt{1 + 4t_k^2})/2,$$

$$\boxed{\mathbf{y}_{k+1} \leftarrow \mathbf{x}_k + (t_k - 1)/t_{k+1}(\mathbf{x}_k - \mathbf{x}_{k-1})}$$

Theorem 1: Let $\{\mathbf{x}_k\}$ be generated by the APG method and $\mathbf{x}^*$ be any optimal solution, then

$$F(\mathbf{x}_k) - F(\mathbf{x}^*) \le \frac{2L_f \|\mathbf{x}_0 - \mathbf{x}^*\|_F^2}{(k+1)^2}, \qquad \forall k \ge 1. \qquad (20)$$

A Generalized Accelerated Proximal Gradient Approach for Total-Variation-Based Image, IEEE T-IP, 2010

Nesterov 动量

$$\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \epsilon \nabla_{\boldsymbol{\theta}} \left[\frac{1}{m} \sum_{i=1}^{m} L\big(\boldsymbol{f}(\boldsymbol{x}^{(i)}\boxed{; \boldsymbol{\theta} + \alpha \boldsymbol{v}}), \boldsymbol{y}^{(i)}\big) \right]$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v},$$

算法 8.3 使用Nesterov 动量的随机梯度下降（SGD）

Require: 学习率 ϵ，动量参数 α

Require: 初始参数 $\boldsymbol{\theta}$，初始速度 $\boldsymbol{v}$

while 没有达到停止准则 **do**

从训练集中采包含 m 个样本 $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ 的小批量，对应目标为 $\boldsymbol{y}^{(i)}$。

应用临时更新：$\tilde{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta} + \alpha \boldsymbol{v}$

计算梯度（在临时点）：$\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\tilde{\boldsymbol{\theta}}} \sum_i L(f(\boldsymbol{x}^{(i)}; \tilde{\boldsymbol{\theta}}), \boldsymbol{y}^{(i)})$

计算速度更新：$\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \epsilon \boldsymbol{g}$

应用更新：$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$

end while

在随机梯度的情况下，Nesterov动量没有改进收敛率。

AdaGrad

- 具有损失最大偏导的参数相应地有一个快速下降的学习率。在参数空间中更为平缓的倾斜方向会取得更大的下降。

算法 8.4 AdaGrad 算法

Require: 全局学习率 ϵ

Require: 初始参数 $\boldsymbol{\theta}$

Require: 小常数 δ，为了数值稳定大约设为 10^{-7}

初始化梯度累积变量 $\boldsymbol{r} = 0$

while 没有达到停止准则 **do**

从训练集中采包含 m 个样本 $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ 的小批量，对应目标为 $\boldsymbol{y}^{(i)}$。

计算梯度：$\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$

累积平方梯度：$\boldsymbol{r} \leftarrow \boldsymbol{r} + \boldsymbol{g} \odot \boldsymbol{g}$

计算更新：$\Delta\boldsymbol{\theta} \leftarrow -\frac{\epsilon}{\delta + \sqrt{\boldsymbol{r}}} \odot \boldsymbol{g}$ （逐元素地应用除和求平方根）

应用更新：$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$

end while

AdaGrad

- 凸优化情况下有较好的理论性质，但对于深度神经网络一般不成立
- 对于训练深度神经网络模型，从训练开始时积累梯度平方会导致有效学习率在达到这样的凸结构前过早和过量的减小 。
- 在某些深度学习模型上效果不错，但不是全部。

RMSProp

- 使用指数衰减平均以丢弃早期历史，使其能够在找到凸结构后快速收敛；
- 相比于AdaGrad，基于移动平均引入了一个新的超参数ρ，用于控制移动平均的长度范围。
- 可结合SGD和Nesterov 动量方法

RMSProp

算法 8.5 RMSProp 算法

Require: 全局学习率 ϵ，衰减速率 ρ

Require: 初始参数 $\boldsymbol{\theta}$

Require: 小常数 δ，通常设为 10^{-6}（用于被小数除时的数值稳定）

初始化累积变量 $\boldsymbol{r}=0$

while 没有达到停止准则 **do**

从训练集中采包含 m 个样本 $\{\boldsymbol{x}^{(1)},\ldots,\boldsymbol{x}^{(m)}\}$ 的小批量，对应目标为 $\boldsymbol{y}^{(i)}$。

计算梯度：$\boldsymbol{g}\leftarrow\frac{1}{m}\nabla_{\boldsymbol{\theta}}\sum_i L(f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}),\boldsymbol{y}^{(i)})$

累积平方梯度：$\boxed{\boldsymbol{r}\leftarrow\rho\boldsymbol{r}+(1-\rho)\boldsymbol{g}\odot\boldsymbol{g}}$

计算参数更新：$\Delta\boldsymbol{\theta}=\boxed{-\frac{\epsilon}{\sqrt{\delta+\boldsymbol{r}}}}\odot\boldsymbol{g}$ （$\frac{1}{\sqrt{\delta+\boldsymbol{r}}}$ 逐元素应用）

应用更新：$\boldsymbol{\theta}\leftarrow\boldsymbol{\theta}+\Delta\boldsymbol{\theta}$

end while

RMSProp-2

算法 8.6 使用 Nesterov 动量的 RMSProp 算法

Require: 全局学习率 ϵ，衰减速率 ρ，动量系数 α

Require: 初始参数 $\boldsymbol{\theta}$，初始参数 $\boldsymbol{v}$

初始化累积变量 $\boldsymbol{r} = 0$

while 没有达到停止准则 **do**

从训练集中采包含 m 个样本 $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ 的小批量，对应目标为 $\boldsymbol{y}^{(i)}$。

计算临时更新：$\tilde{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta} + \alpha \boldsymbol{v}$

计算梯度：$\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\tilde{\boldsymbol{\theta}}} \sum_i L(f(\boldsymbol{x}^{(i)}; \tilde{\boldsymbol{\theta}}), \boldsymbol{y}^{(i)})$

累积梯度：$\boxed{\boldsymbol{r} \leftarrow \rho \boldsymbol{r} + (1-\rho) \boldsymbol{g} \odot \boldsymbol{g}}$

计算速度更新：$\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \boxed{\frac{\epsilon}{\sqrt{\boldsymbol{r}}}} \odot \boldsymbol{g}$ （$\frac{1}{\sqrt{\boldsymbol{r}}}$ 逐元素应用）

应用更新：$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$

end while

ADAM: ADAptive Moments

算法 8.7 Adam 算法

Require: 步长 ϵ（建议默认为：0.001）

Require: 矩估计的指数衰减速率，ρ_1 和 ρ_2 在区间 $[0, 1)$ 内。（建议默认为：分别为 0.9 和 0.999）

Require: 用于数值稳定的小常数 δ（建议默认为：10^{-8}）

Require: 初始参数 $\boldsymbol{\theta}$

初始化一阶和二阶矩变量 $\boldsymbol{s} = 0$, $\boldsymbol{r} = 0$

初始化时间步 $t = 0$

while 没有达到停止准则 **do**

从训练集中采包含 m 个样本 $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ 的小批量，对应目标为 $\boldsymbol{y}^{(i)}$。

计算梯度：$\boldsymbol{g} \leftarrow \frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$

$t \leftarrow t + 1$

更新有偏一阶矩估计：$\boxed{\boldsymbol{s} \leftarrow \rho_1 \boldsymbol{s} + (1 - \rho_1)\boldsymbol{g}}$

更新有偏二阶矩估计：$\boxed{\boldsymbol{r} \leftarrow \rho_2 \boldsymbol{r} + (1 - \rho_2)\boldsymbol{g} \odot \boldsymbol{g}}$

修正一阶矩的偏差：$\hat{\boldsymbol{s}} \leftarrow \frac{\boldsymbol{s}}{1-\rho_1^t}$

修正二阶矩的偏差：$\hat{\boldsymbol{r}} \leftarrow \frac{\boldsymbol{r}}{1-\rho_2^t}$

计算更新：$\Delta\boldsymbol{\theta} = -\epsilon \frac{\hat{\boldsymbol{s}}}{\sqrt{\hat{\boldsymbol{r}}}+\delta}$（逐元素应用操作）

应用更新：$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$

end while

- ADAM中动量直接并入了梯度一阶矩（指数加权）的估计。
- ADAM 包括了偏置修正，修正从原点初始化的一阶矩（动量项）和（非中心的）二阶矩的估计。
- Adam 通常被认为对超参数的选择相当鲁棒，尽管学习率有时需要修改建议的默认值。

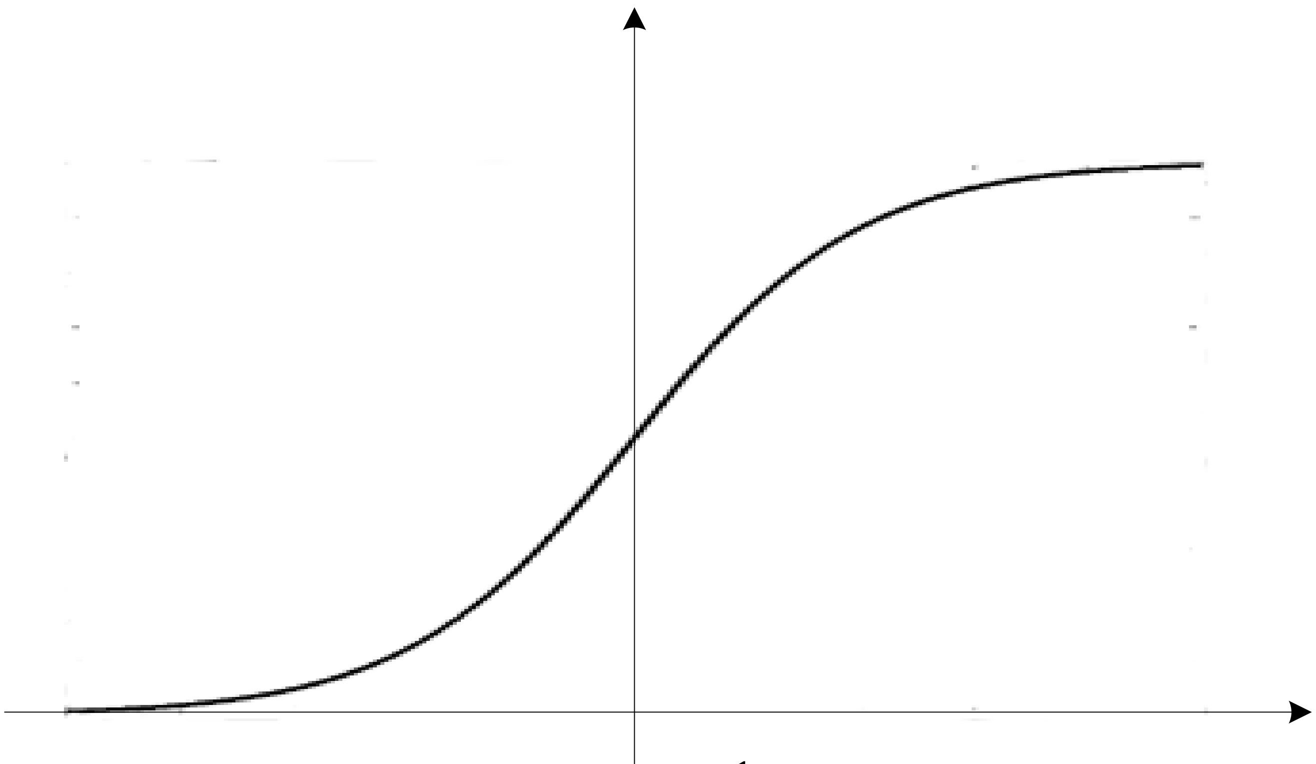
小结

- 基本算法
  - SGD、SGD-Momentum、Nesterov 动量
- 自适应学习率算法
  - AdaGrad、RMSProp、ADAM
- 二阶及其近似：牛顿法、共轭梯度、BFGS
- 建议使用
  - SGD、SGD-Momentum、RMSProp、ADAM

主要内容

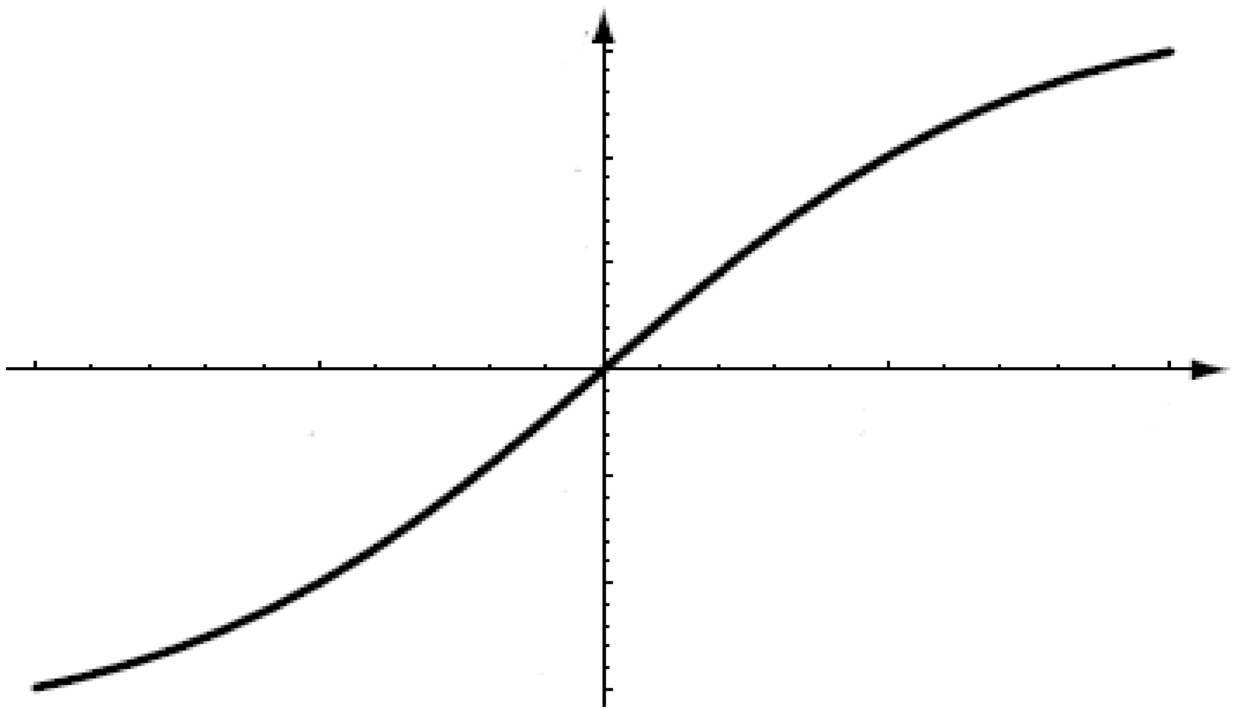
- 随机优化算法（Stochastic Optimization）
  - 随机优化
  - SGD、AdaGrad、RMSProp、ADAM
- **网络模型：**3层 -> 多层
  - **激活函数：**ReLU / PReLU, …
  - 网络学习：Dropout、批标准化（Batch Normalization）
- 泛化能力

激活函数—对数Sigmoid函数



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

激活函数—双曲正切Sigmoid函数

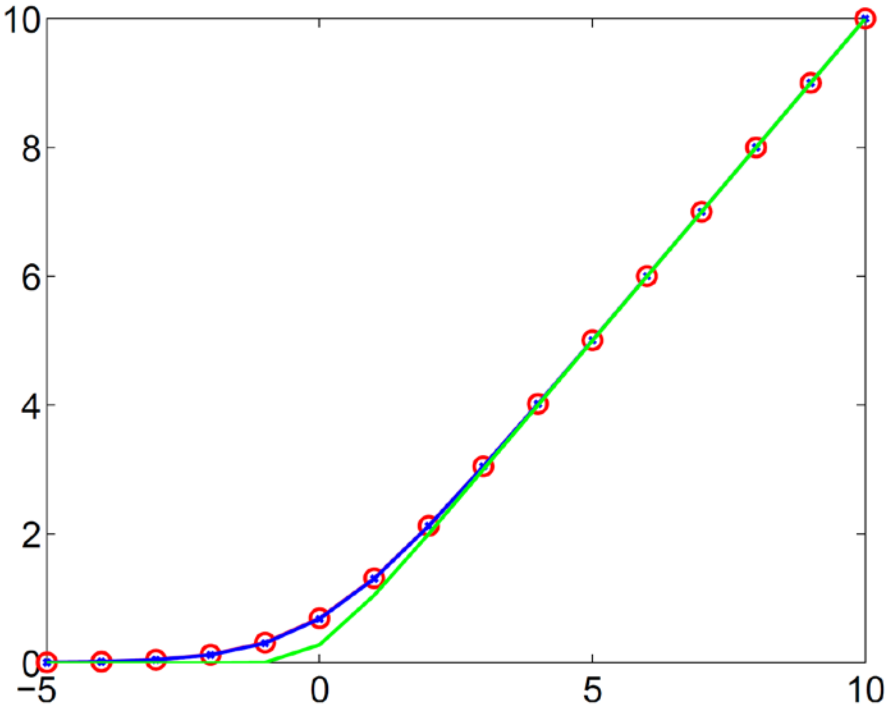


$$f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

激活函数—ReLU (Rectified Linear Units)

- 受限玻尔兹曼机

$$\sum_{i=1}^{N} \sigma(x - i + 0.5) \approx \log(1 + e^x)$$

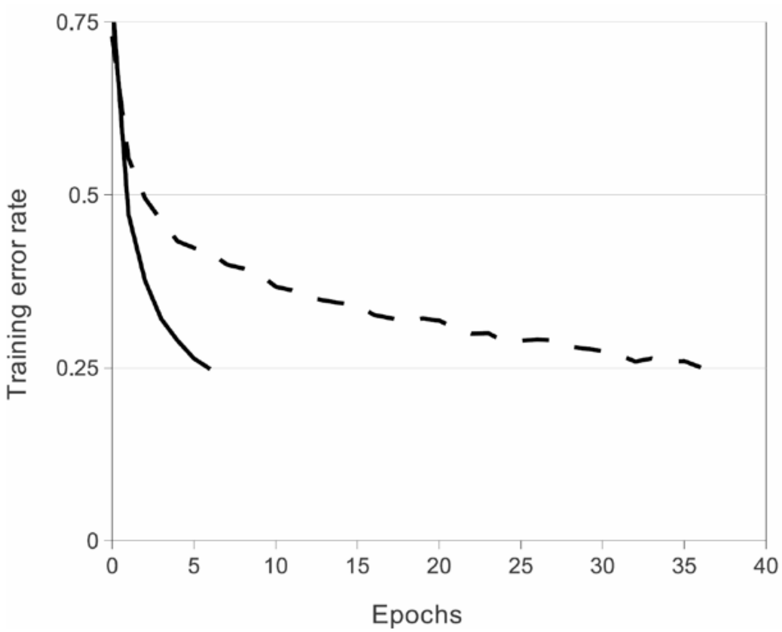


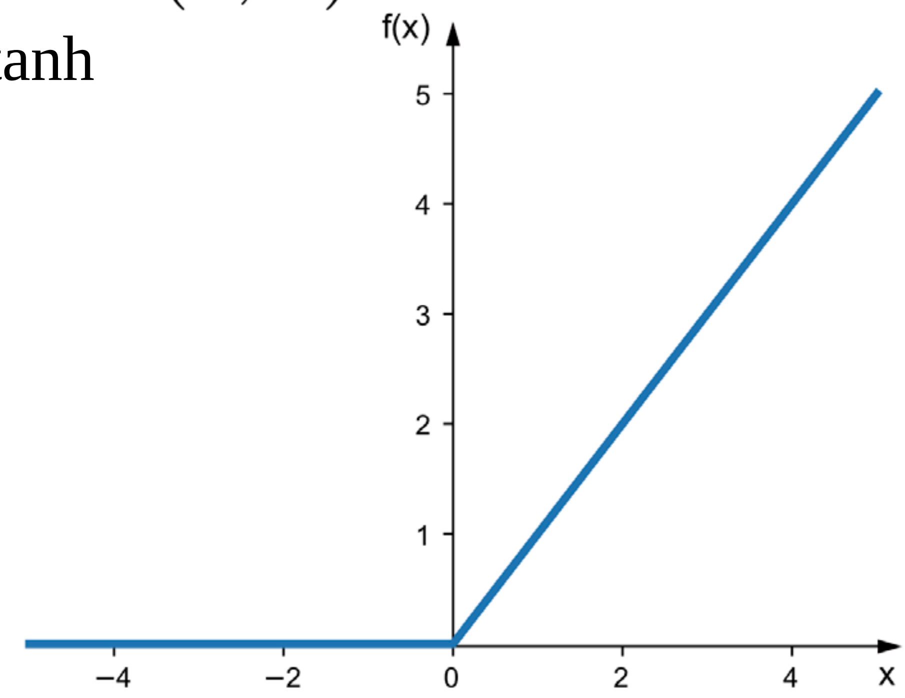
Vinod Nair, Geoffrey E. Hinton, Rectified Linear Units Improve Restricted Boltzmann Machines, ICML 2010.

激活函数—ReLU

- AlexNet $f(x) = \max(0, x)$

solid line: ReLU; dashed line: tanh

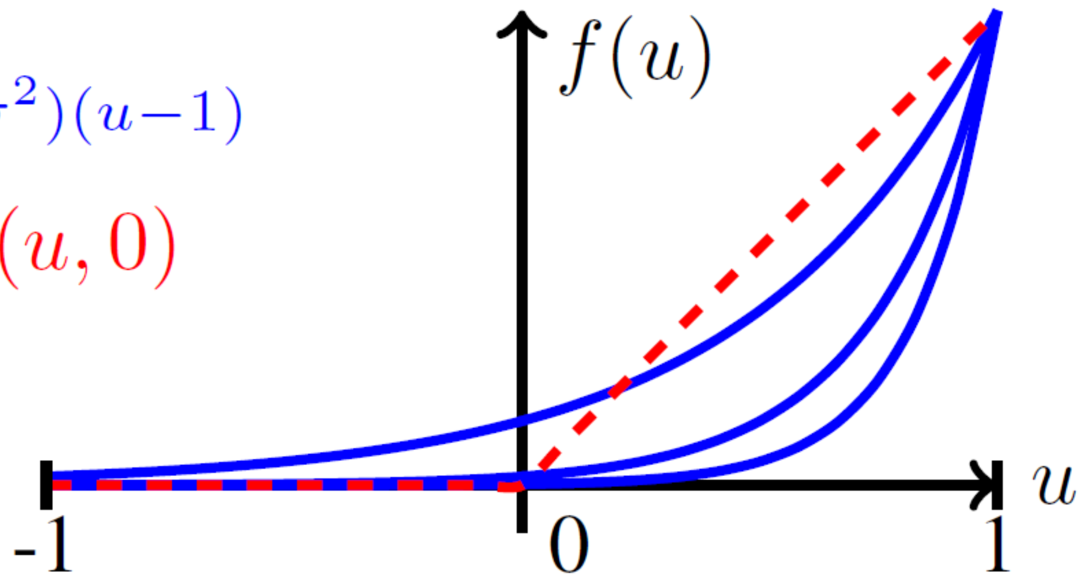




A Krizhevsky, I Sutskever, GE Hinton, Imagenet classification with deep convolutional neural networks, NIPS 2012

激活函数—ReLU

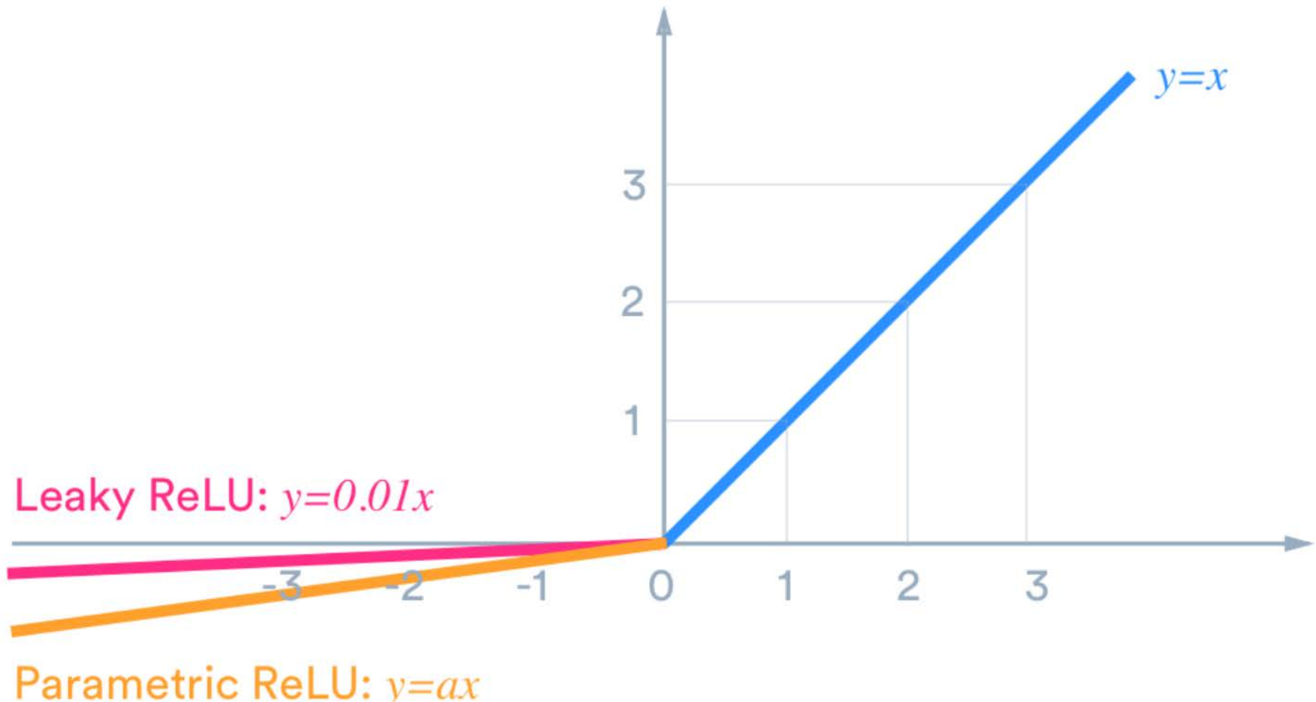
$$f(u) = e^{(2/\sigma^2)(u-1)}$$
$$f(u) = \max(u, 0)$$



J Mairal, P Koniusz, Z Harchaoui, C Schmid, Convolutional Kernel Networks, NIPS 2014

ReLU变体及改进

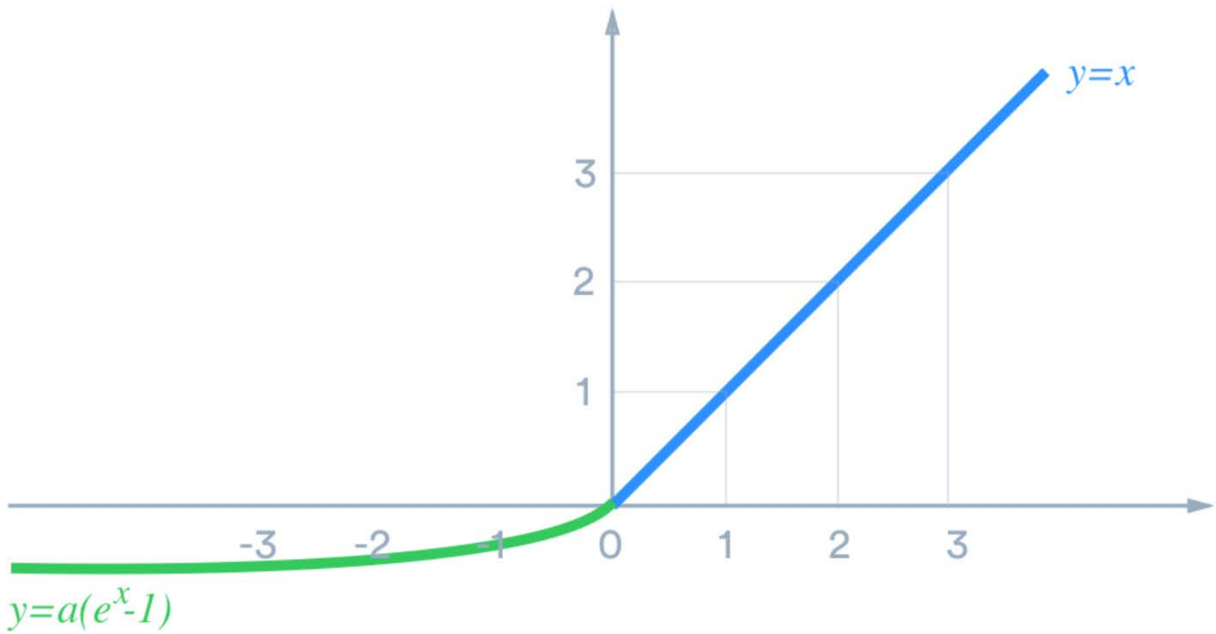
- Leaky ReLU & PReLU



K He, X Zhang, S Ren, J Sun, Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, ICCV 2015

ReLU变体及改进

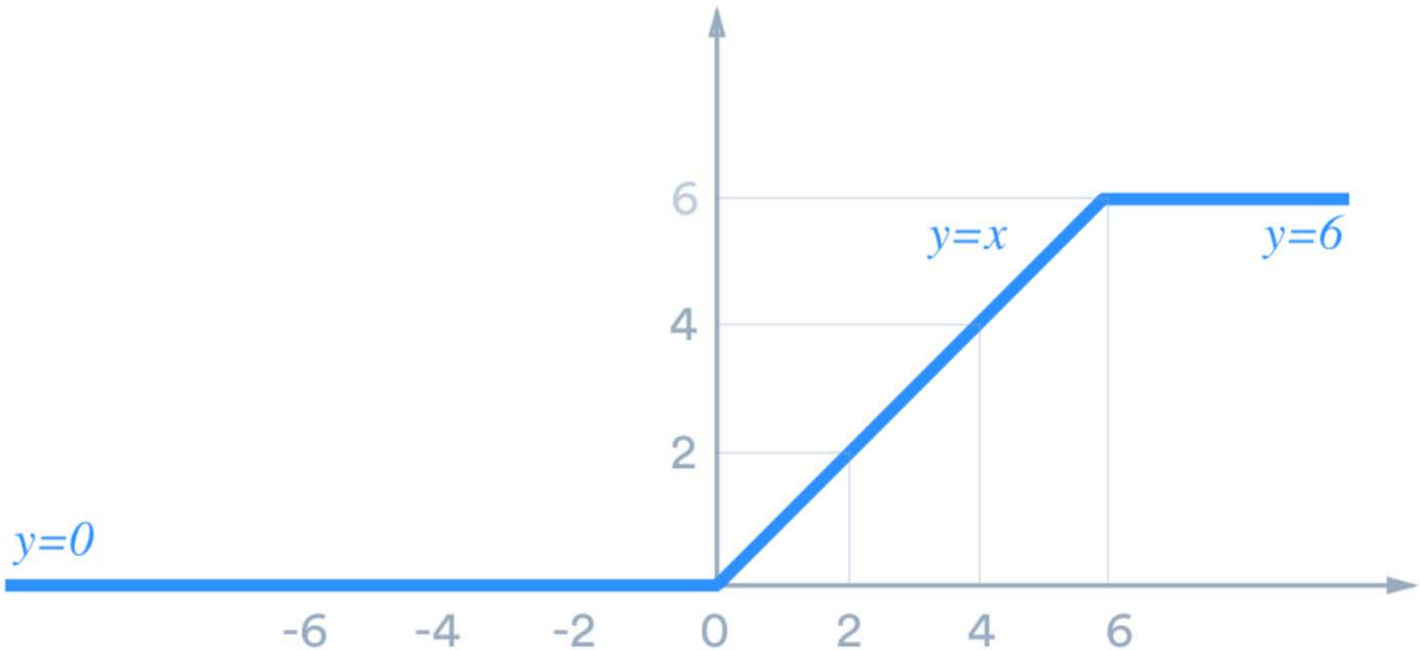
- ELU: Exponential Linear Units



D.-A. Clevert, T. Unterthiner, S. Hochreiter, Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs), ICLR 2016

ReLU变体及改进

- ReLU-6



Alex Krizhevsky, Convolutional Deep Belief Networks on CIFAR-10, Technic Report, 2010

主要内容

- **随机优化算法**（Stochastic Optimization）
  - 随机优化
  - SGD、AdaGrad、RMSProp、ADAM
- **网络模型**：3层 -> 多层
  - 激活函数：ReLU / PReLU, …
  - 网络参数：**Dropout**、批标准化（Batch Normalization）
- 泛化能力

模型平均与Bagging

- 模型平均

假设我们有 k 个回归模型。假设每个模型在每个例子上的误差是 ϵ_i，这个误差服从零均值方差为 $\mathbb{E}[\epsilon_i^2]=v$ 且协方差为 $\mathbb{E}[\epsilon_i\epsilon_j]=c$ 的多维正态分布。通过所有集成模型的平均预测所得误差是 $\frac{1}{k}\sum_i \epsilon_i$。集成预测器平方误差的期望是

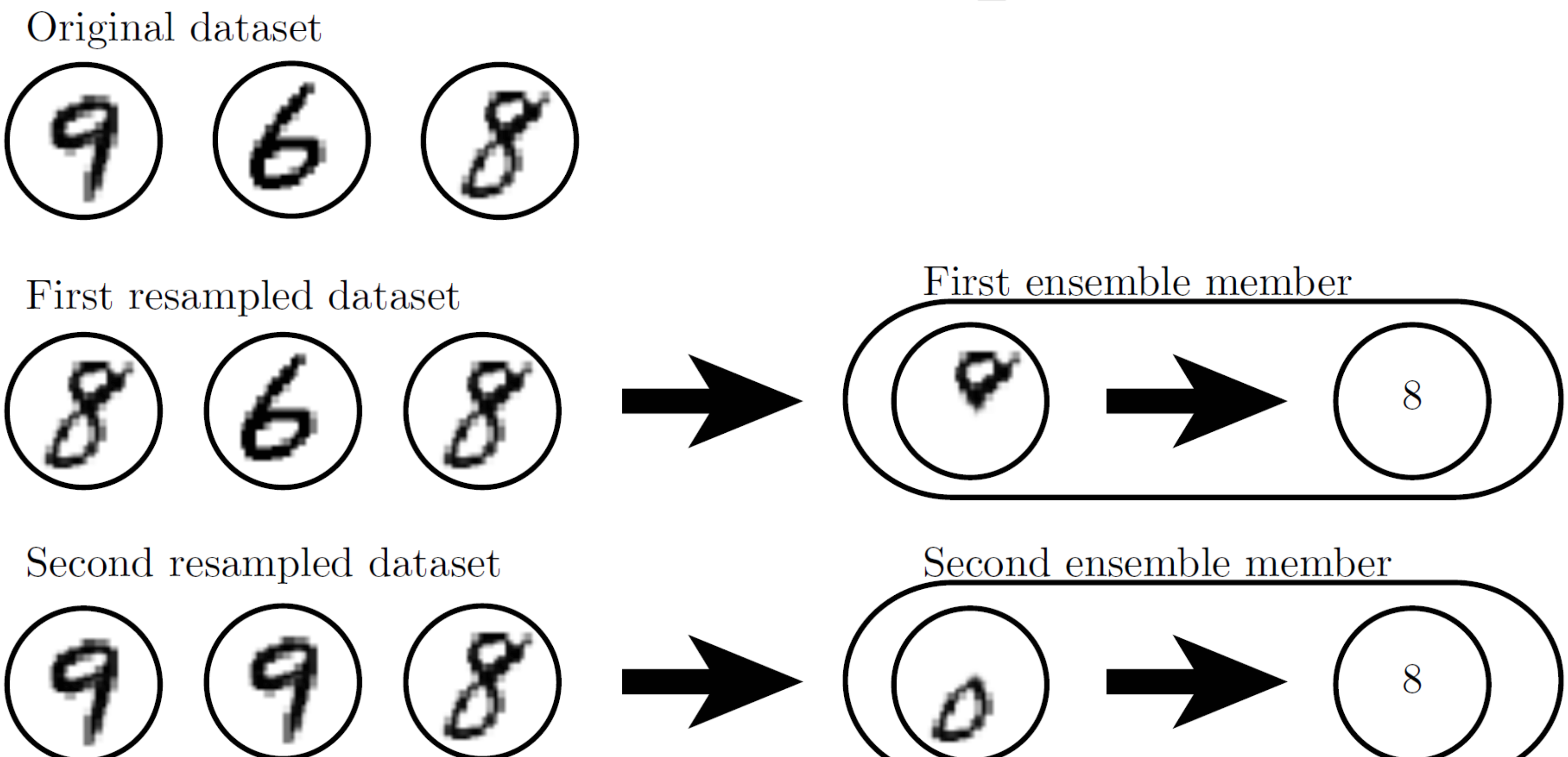
$$\mathbb{E}\left[\left(\frac{1}{k}\sum_i \epsilon_i\right)^2\right] = \frac{1}{k^2}\mathbb{E}\left[\sum_i\left(\epsilon_i^2+\sum_{j\neq i}\epsilon_i\epsilon_j\right)\right], \tag{7.50}$$

$$= \frac{1}{k}v + \frac{k-1}{k}c. \tag{7.51}$$

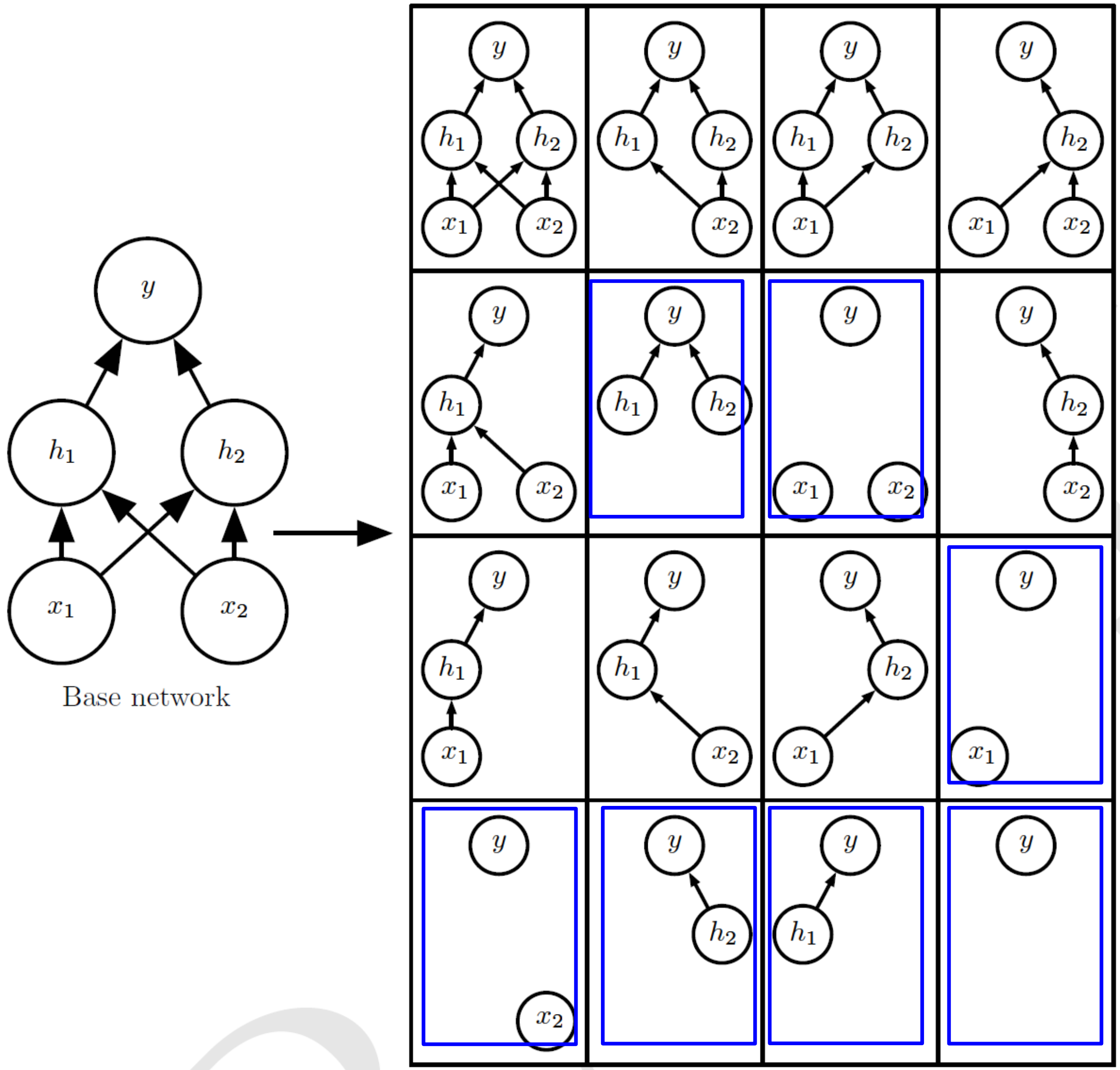
在误差完全相关即 $c=v$ 的情况下，均方误差减少到 v，所以模型平均没有任何帮助。在错误完全不相关即 $c=0$ 的情况下，该集成平方误差的期望仅为 $\frac{1}{k}v$。这意味

模型平均与Bagging

- Bagging



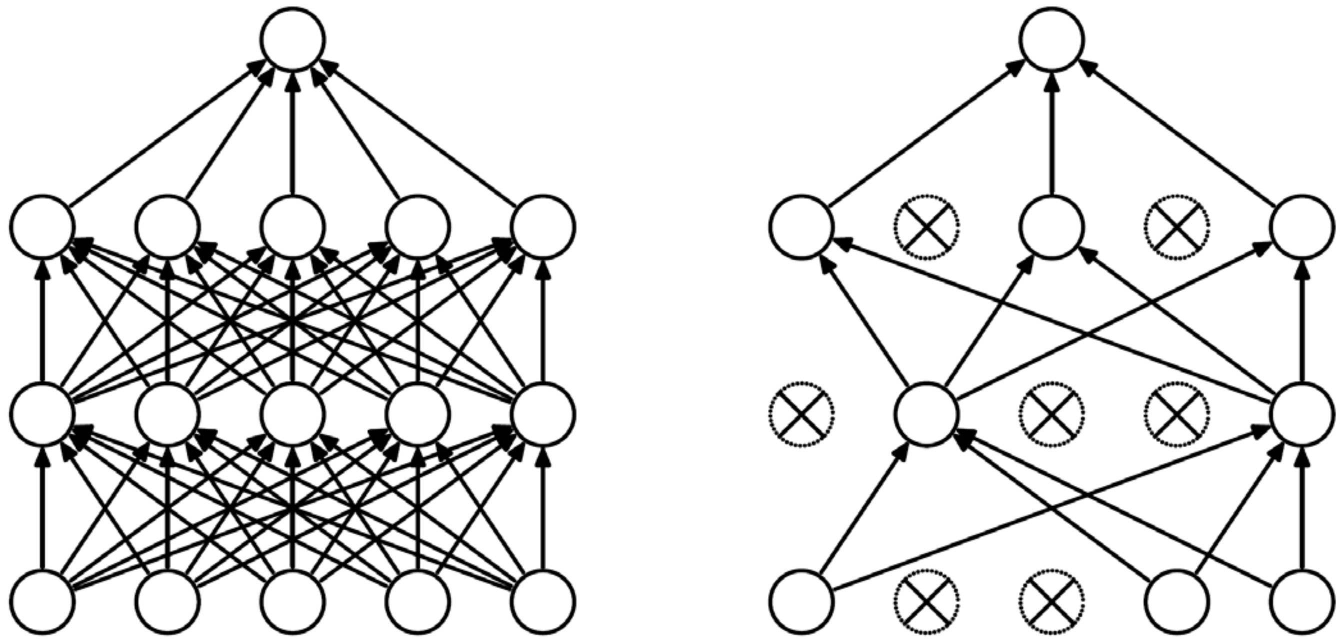
Dropout



Base network

Ensemble of subnetworks

Dropout训练



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

- 在小批量中加载一个样本，然后随机抽样应用于网络中所有输入和隐藏单元的不同二值掩码
- 超参数：掩码值为1 的采样概率

Dropout

- 训练阶段所有模型共享参数，测试阶段直接组装成一个整体的大网络
- 有效避免过拟合
- 可用于前馈神经网络、概率模型，如受限玻尔兹曼机，以及循环神经网络等
- 会需要较多的迭代次数和训练时间
- 理论解释

理论解释

- First-order equivalent to an L2 regularizer applied after scaling the features by an estimate of the inverse diagonal Fisher information matrix.

- Understanding Dropout, NIPS 2013
- Dropout Training as Adaptive Regularization, NIPS 2013

主要内容

- **随机优化算法**（Stochastic Optimization）
  - 随机优化
  - SGD、AdaGrad、RMSProp、ADAM
- **网络模型**：3层 -> 多层
  - 激活函数：ReLU / PReLU, …
  - 网络参数：Dropout、**批标准化**（Batch Normalization）
- 泛化能力

批标准化（Batch Normalization）

- 最激动人心的进展之一，自适应重参数化，减少了多层之间协调更新，有助于训练非常深的模型。
- Mini-batch SGD：Internal covariate shift
- 改善网络学习稳定性
- 正则化

批标准化（Batch Normalization）

- $z = g(\mathrm{BN}(W u))$
- 测试阶段：
- 用训练阶段收集的运行均值和方法代替Batch均值和方差

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

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

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

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

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

反向传播

$$\frac{\partial \ell}{\partial \widehat{x}_i} = \frac{\partial \ell}{\partial y_i} \cdot \gamma$$

$$\frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot (x_i - \mu_{\mathcal{B}}) \cdot \frac{-1}{2}(\sigma_{\mathcal{B}}^2 + \epsilon)^{-3/2}$$

$$\frac{\partial \ell}{\partial \mu_{\mathcal{B}}} = \left(\sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \right) + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{\sum_{i=1}^{m} -2(x_i - \mu_{\mathcal{B}})}{m}$$

$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(x_i - \mu_{\mathcal{B}})}{m} + \frac{\partial \ell}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{m}$$

$$\frac{\partial \ell}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \cdot \widehat{x}_i$$

$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i}$$

批标准化（Batch Normalization）

- 优点:
  - 正则化
  - 允许设置较高的学习率
  - Internal covariate shift
  - 可代替dropout
  - 可代替L2正则化

- 会需要较大的batch size。一般会有效，但不能保证任何情况下都会有效

主要内容

- 随机优化算法（Stochastic Optimization）
  - 随机优化
  - SGD、AdaGrad、RMSProp、ADAM

- 网络模型：3层 -> 多层
  - 激活函数：ReLU / PReLU, …
  - 网络参数：Dropout、批标准化（Batch Normalization）

- **泛化能力**

深度神经网络的泛化能力

- 目前的学习算法原则上只能收敛到一个局部最优解，但看起来学到的局部最优解总能具有较好的分类性能。
  - 过拟合训练数据
  - 真实问题解的分布
  - 能够绕过一些不好的局部最优解（SGD）

深度神经网络的泛化能力

- 不同的深度神经网络在训练集上均能取得零错误率，但在测试集上则会呈现出不同的泛化能力。
  - 过拟合训练数据
  - 经验风险+模型复杂性理论不能解释

- Understanding Deep Learning Requires Rethinking Generalization, ICLR 2017
- On tighter generalization bound for deep neural networks: CNNs, ResNets, and beyond, arXiv:1806.05159
- Do ImageNet Classifiers Generalize to ImageNet, ICLR 2019
- Gradient descent finds global minima of deep neural networks, arXiv:1811.03804

卷积神经网络

- 历史
- 基本操作
  - 卷积、池化
- 新进展
  - 3x3、dilated convolution
- 典型网络架构
  - LeNet、AlexNet、VGGNet、Inception
  - ResNet、SENet、DenseNet、Attention