神经网络设计

outline

- 选择损失函数与输出函数
- 选择激活函数
- 深层or浅层
- 正则化方法

通用设计

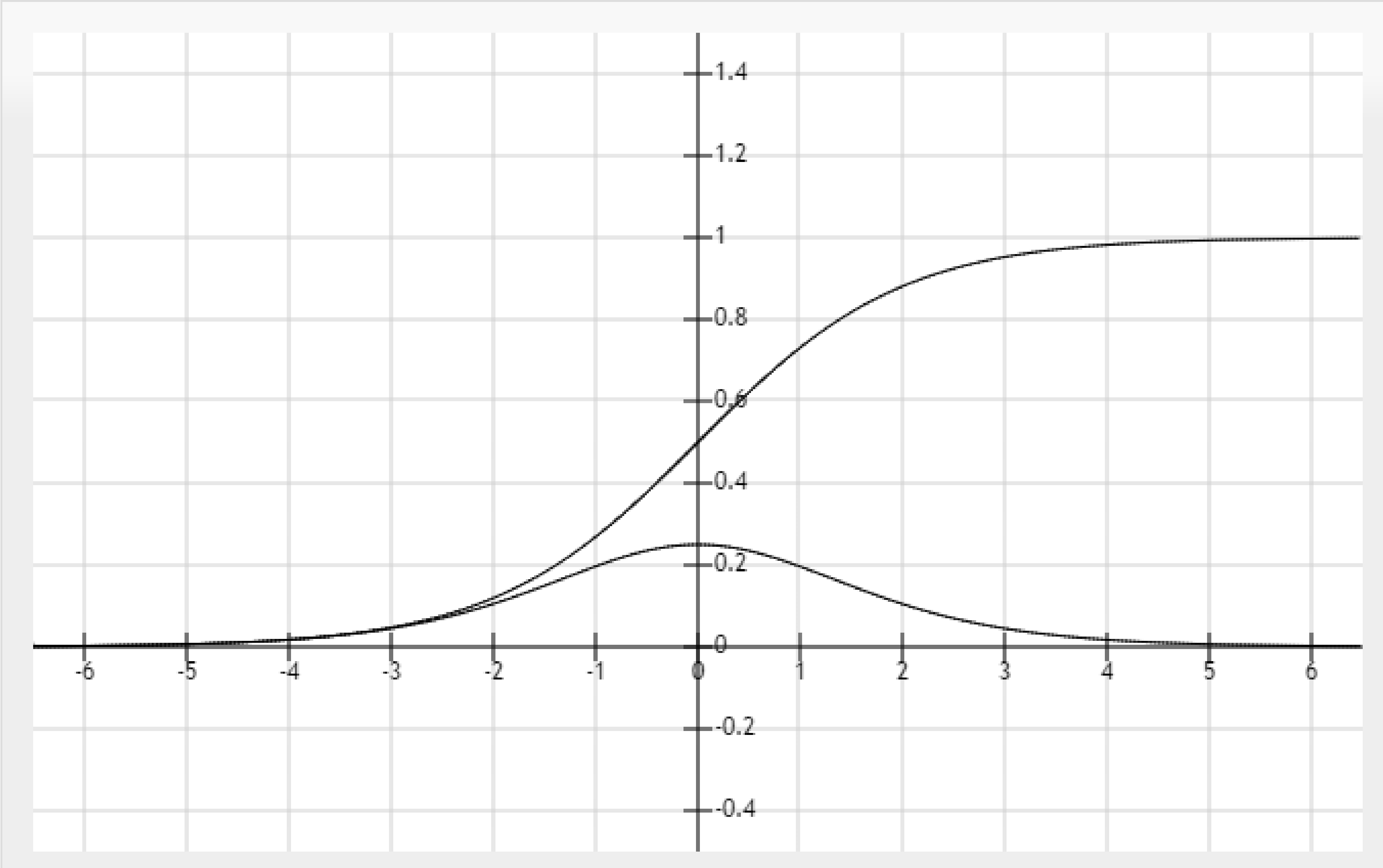
- 优化算法 （基于梯度）
- 输出函数
  - Sigmoid
  - Softmax
- 损失函数（costfunction）

二分类问题损失函数比较

- 均方误差 $J = \frac{1}{2}\left(y - \hat{p}(y=1|x)\right)^{2}$

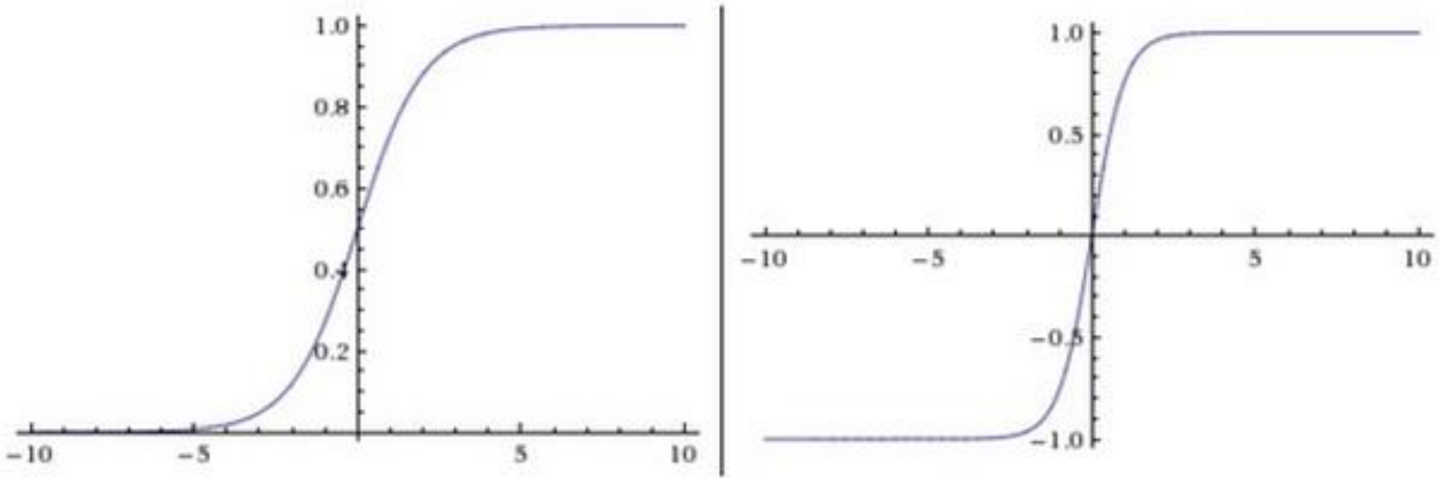
- 极大似然
  - 负对数似然 $J(\boldsymbol{\theta}) = -\mathbb{E}_{\mathbf{x},\mathbf{y}\sim\hat{p}_{\text{data}}} \log p_{\text{model}}(\boldsymbol{y} \mid \boldsymbol{x})$
  - 交叉熵 $H(P,Q) = -\mathbb{E}_{\mathrm{x}\sim P} \log Q(x)$

$$J = -\log\ p\left(y|x\right)$$


1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-6
-5
-4
-3
-2
-1
0
1
2
3
4
5
6

选择激活函数（active area）

- linear
- sigmoid
- tanh



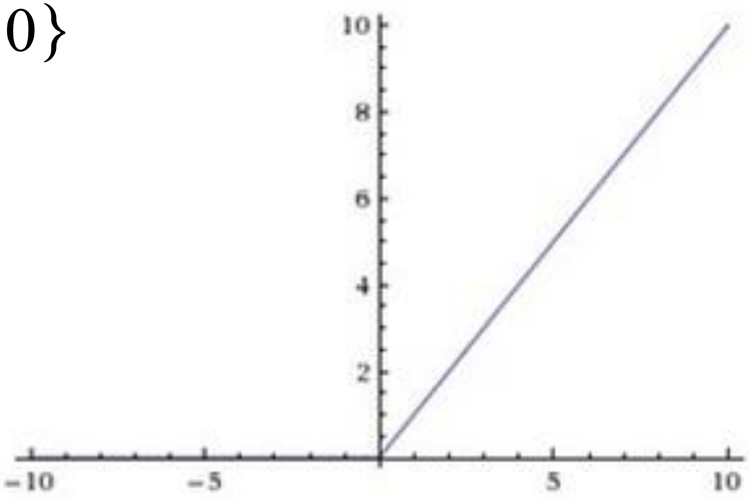
选择激活函数（active area）

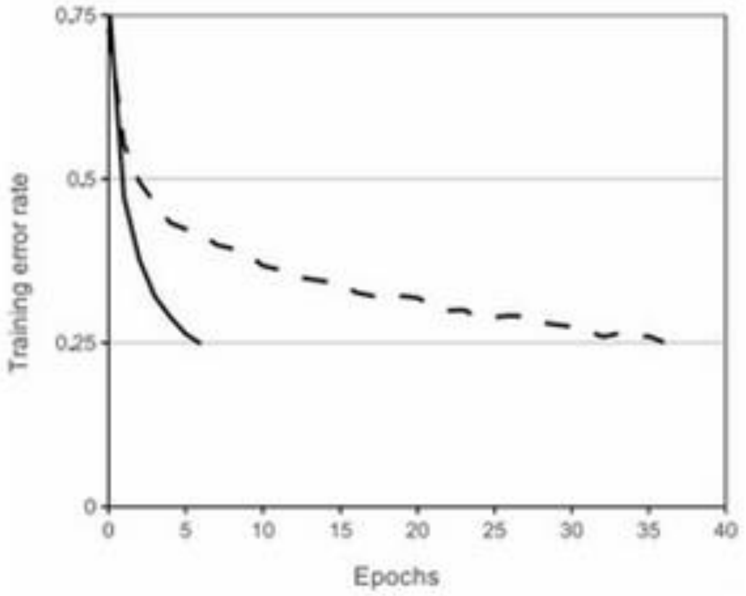
- ReLU $f(x) = \max\{x, 0\}$
  - leakyReLU

$$f(x) = \begin{cases} \alpha x \ for \ x \leqslant 0 \\ x \ for \ x \geqslant 0 \end{cases}$$

  - maxout

$$f(x) = \max(w_1 x + b_1, w_2 x + b_2)$$





深层与浅层

- 表达能力
  - 矩阵的线性变换加激活函数的非线性变换
- 单隐层网络-通用近似器 *（universal approximators）*
  - 神经元数量庞大
  - 实践效果差
- 深层网络
  - 泛化误差小
  - 抽象能力强
  - 训练难度大

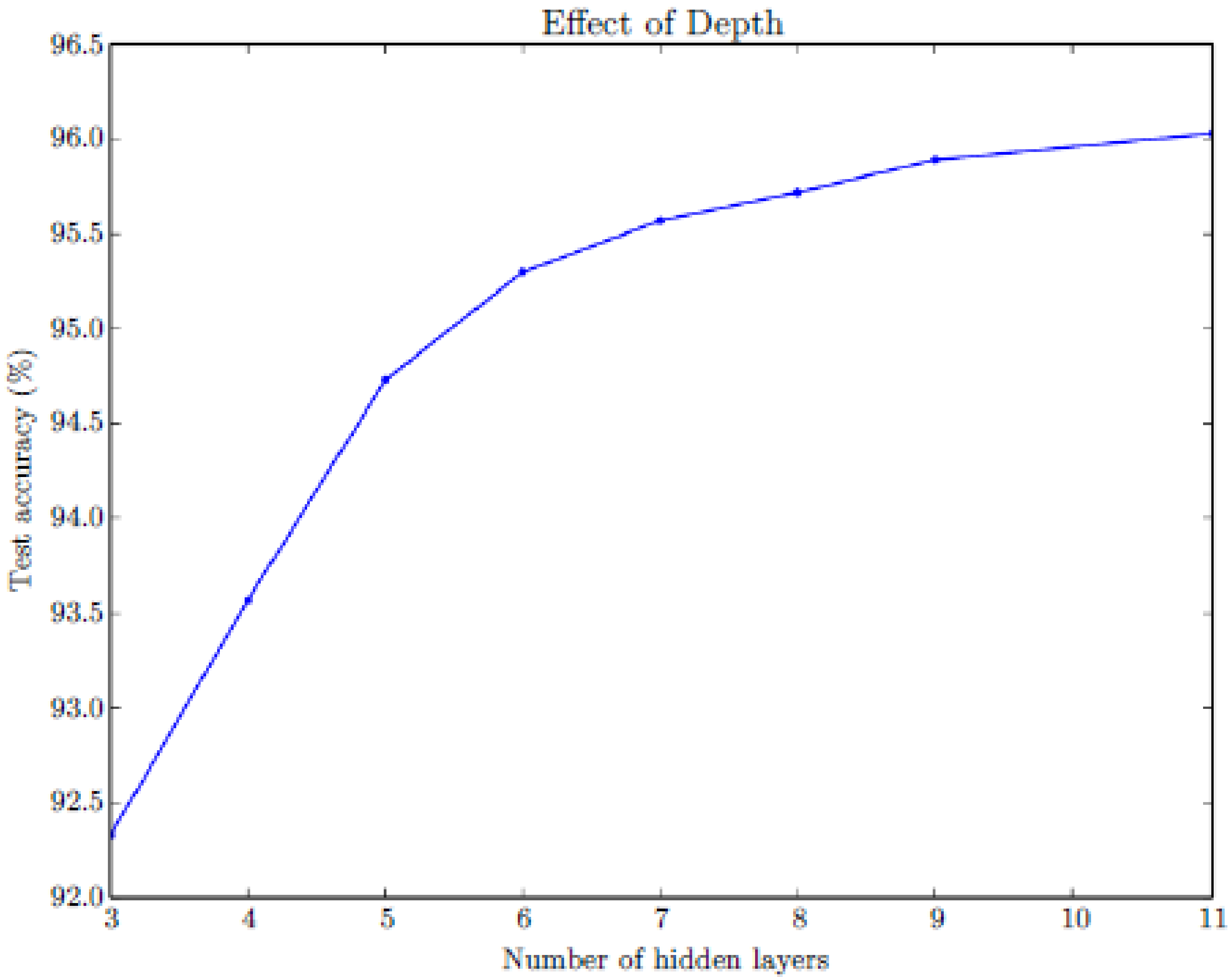


Figure 6.6: Empirical results showing that deeper networks generalize better when used to transcribe multi-digit numbers from photographs of addresses. Data from Goodfellow *et al.* (2014d). The test set accuracy consistently increases with increasing depth. See Fig. 6.7 for a control experiment demonstrating that other increases to the model size do not yield the same effect.

正则化方法

- We defined ***regularization*** as "any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error."
- 参数范数罚
  - L2：

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

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

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

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

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \epsilon\left(\alpha\boldsymbol{w} + \nabla_{\boldsymbol{w}}J(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y})\right)$$

$$\boldsymbol{w} \leftarrow (1-\epsilon\alpha)\boldsymbol{w} - \epsilon\nabla_{\boldsymbol{w}}J(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y})$$

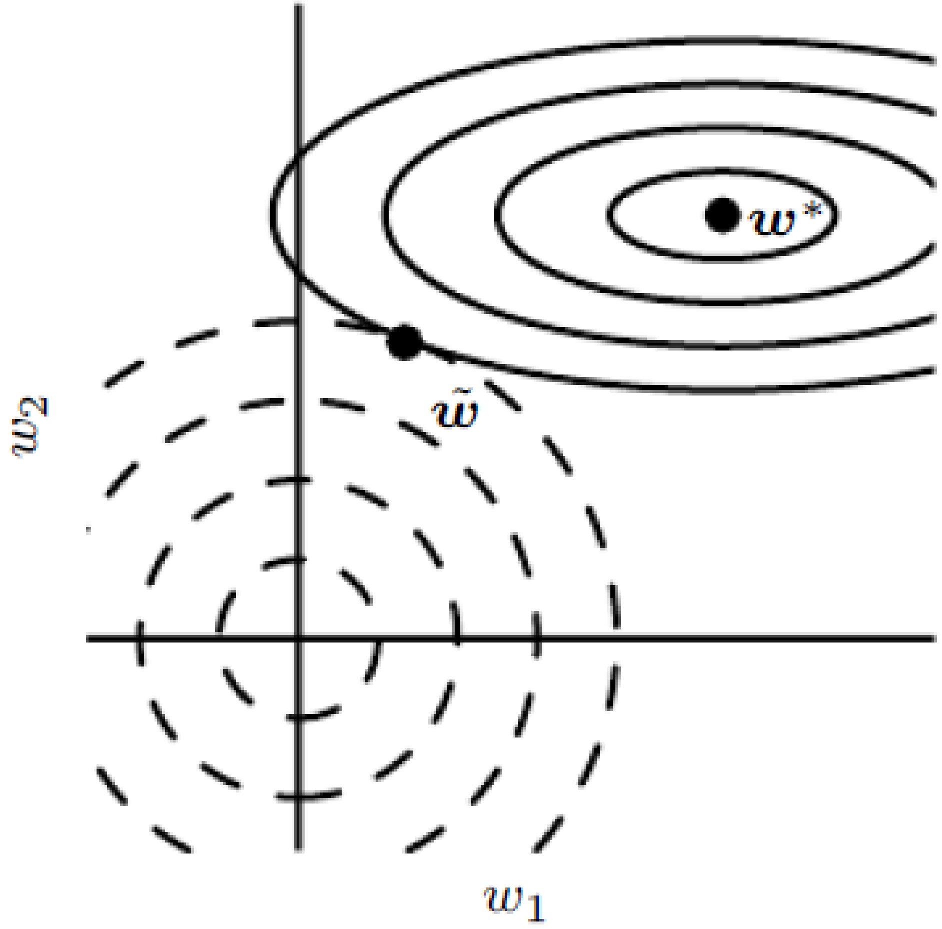


Figure 7.1: An illustration of the effect of L^2 (or weight decay) regularization

正则化方法

- 参数范数罚
  - L2，L1
- 数据扩充
  - 物体识别-旋转图片、放缩图片

正则化方法

- 参数范数罚
  - L2，L1
- 数据扩充
  - 物体识别-旋转图片、放缩图片
- 早停
  - 划分测试集、验证集
  - 简单有效

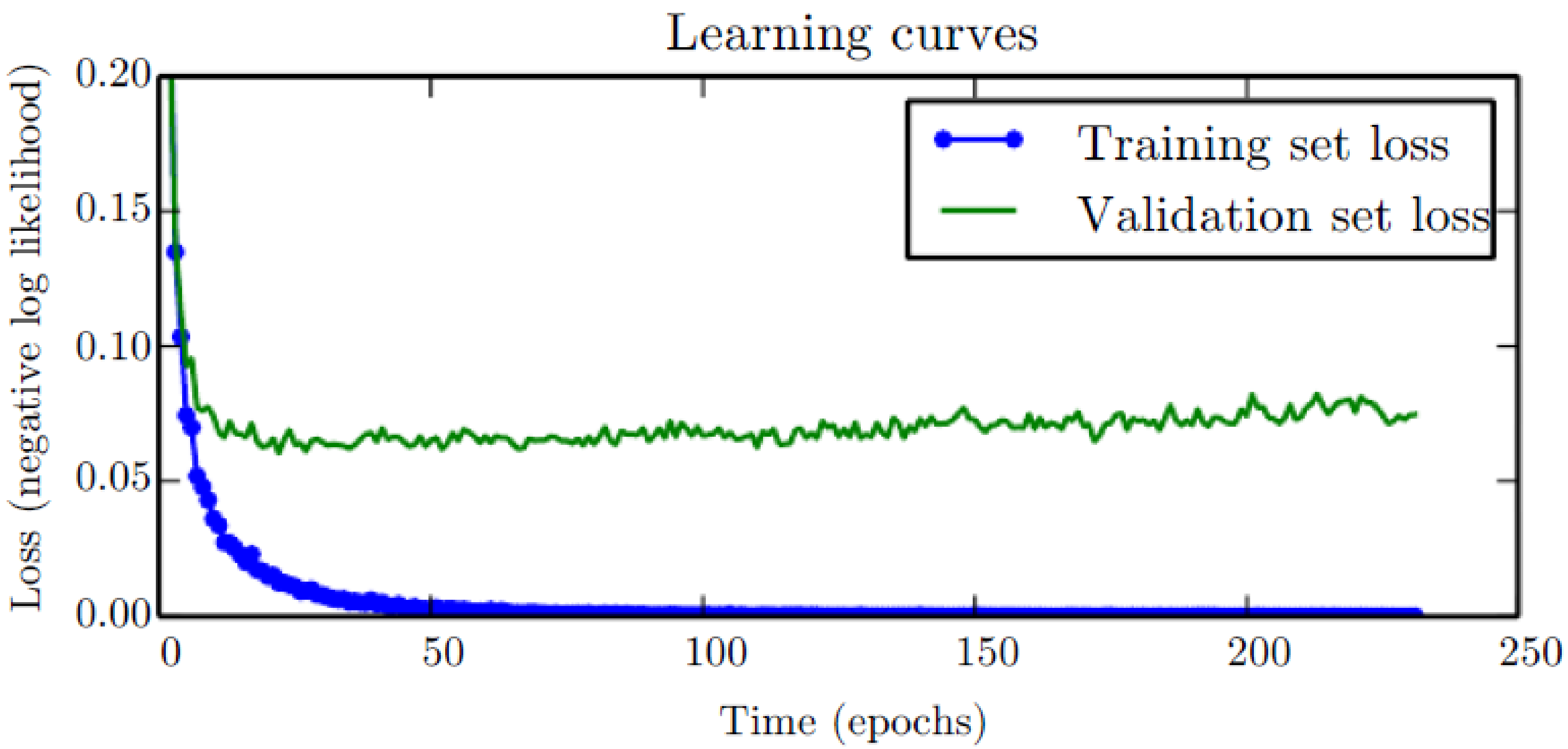


Figure 7.3: Learning curves showing how the negative log-likelihood loss changes over time (indicated as number of training iterations over the dataset, or *epochs*). In this example, we train a maxout network on MNIST. Observe that the training objective decreases consistently over time, but the validation set average loss eventually begins to increase again, forming an asymmetric U-shaped curve.

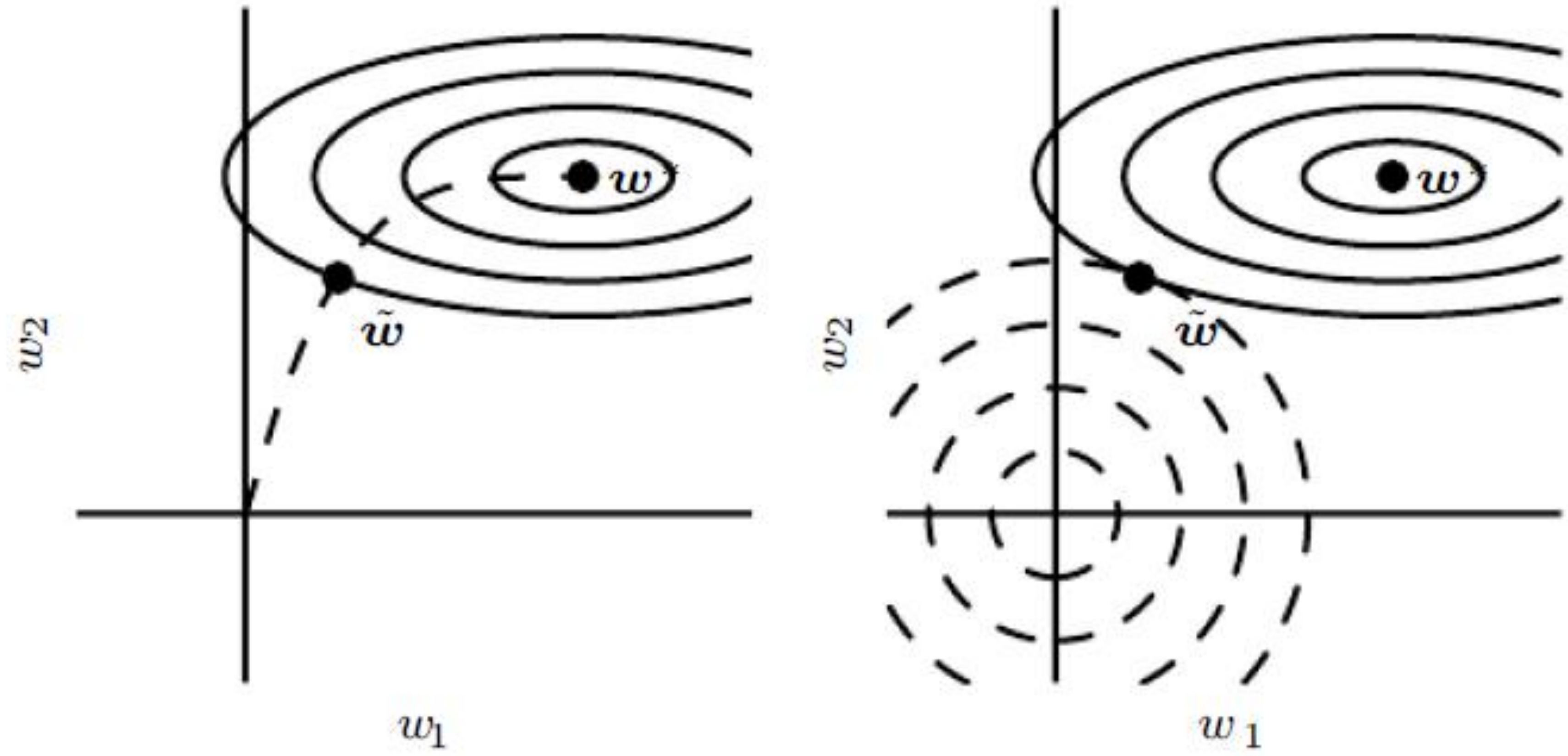


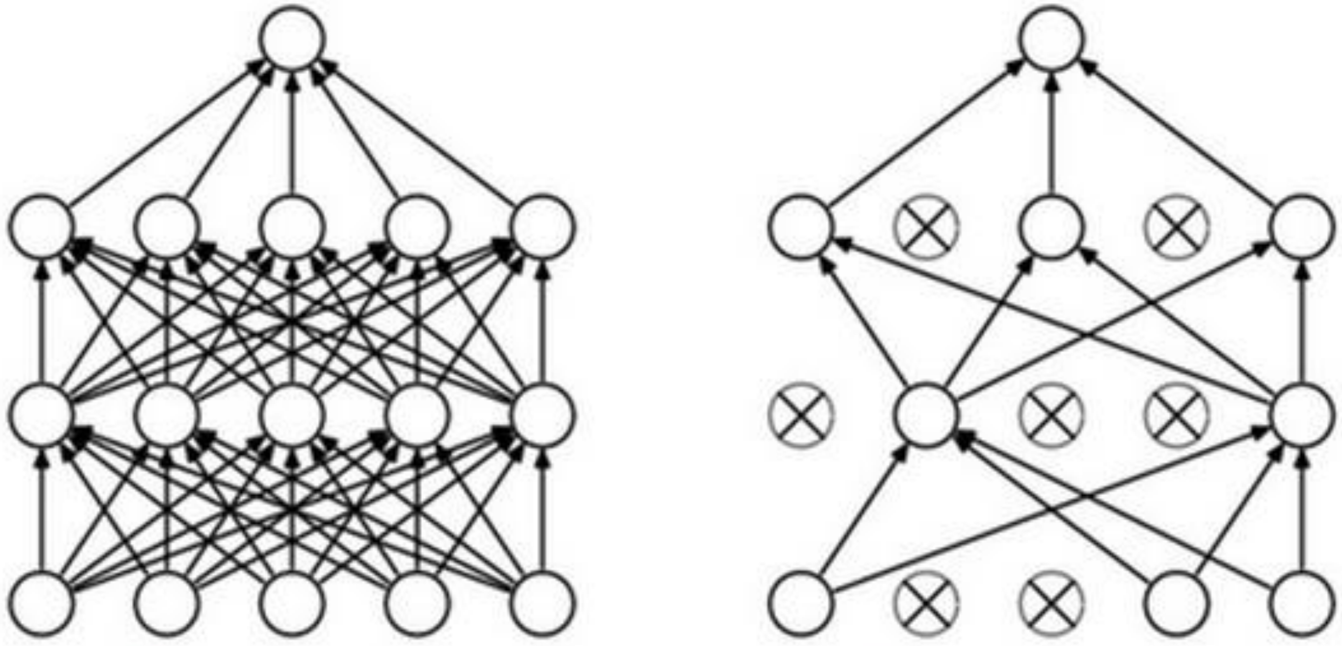
Figure 7.4: An illustration of the effect of early stopping.

正则化方法

- 参数范数罚
- 数据扩充
- 早停
- 组合算法
  - Bagging-构建k个数据集，训练k个神经网络

正则化方法

- 参数范数罚
- 数据扩充
- 早停
- 组合算法
- 随机失活（dropout）



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

  - 神经元以超参数p的概率被激活或者被设置为0

最好的赞赏，就是相约下一期DMC~

赞赏