Neural Networks and Deep Learning

Zhang Minxue

April 11, 2018

Southeast University

About this book

Neural Networks and Deep Learning

- Neural networks, a beautiful biologically-inspired programming paradigm which enables a computer to learn from observational data
- Deep learning, a powerful set of techniques for learning in neural networks

Neural networks and deep learning currently provide the best solutions to many problems in image recognition, speech recognition, and natural language processing. This book will teach you many of the core concepts behind neural networks and deep learning.

Contents

Part I

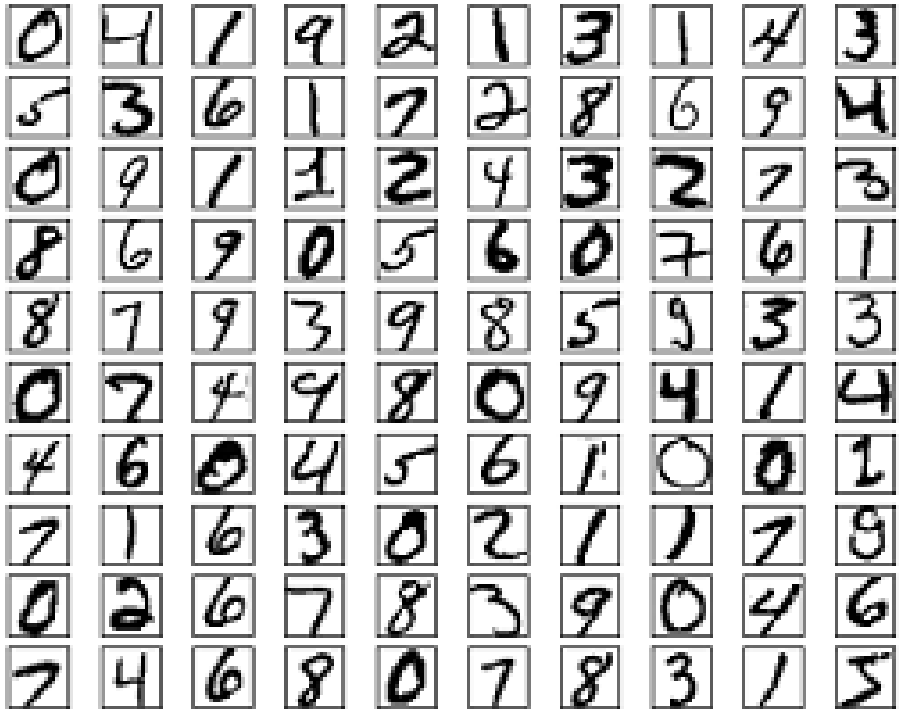
使用神经网络来识别手写数字

使用神经网络来识别手写数字

1. 感知机 (Perceptrons)
2. Sigmoid 神经元
3. 神经网络的结构
4. 分类手写数字的简单网络
5. 用梯度下降法来学习
6. 实现我们的神经网络
7. Toward 深度学习

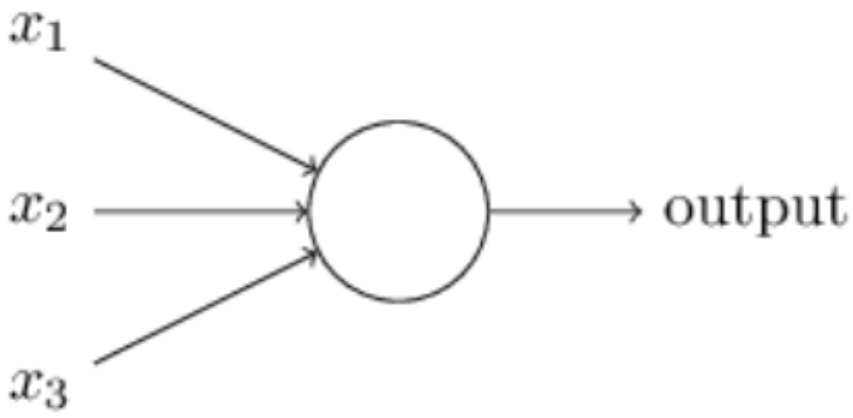
问题描述

输入：待识别的数字图像
输出：图像识别结果



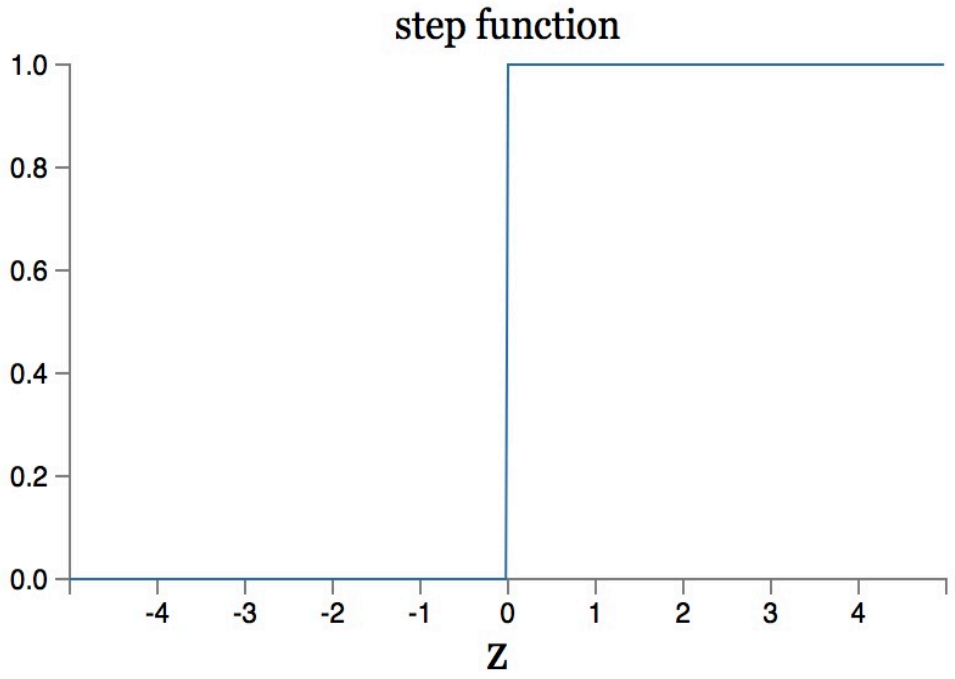
使用神经网络来识别手写数字

感知机 (Perceptrons)



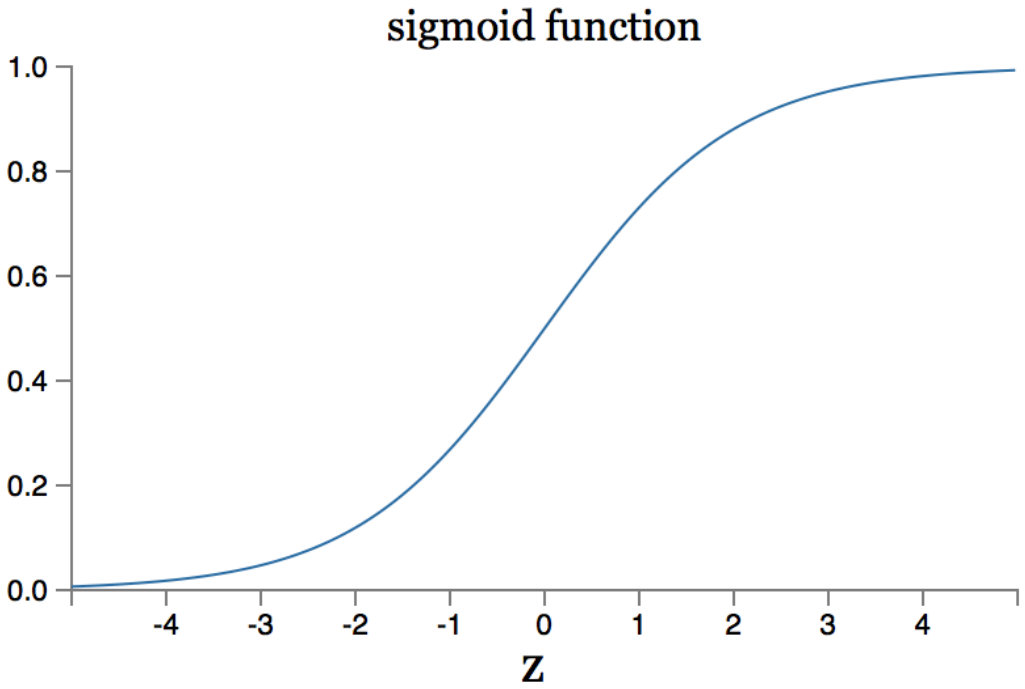
$$output = f(x_1, x_2, x_3) = f(\sum_{i=1}^{n} w_i x_i - \theta)$$

理想中的激活函数



$$y = \begin{cases} 0 & z < 0 \\ 1 & z > 0 \end{cases}$$

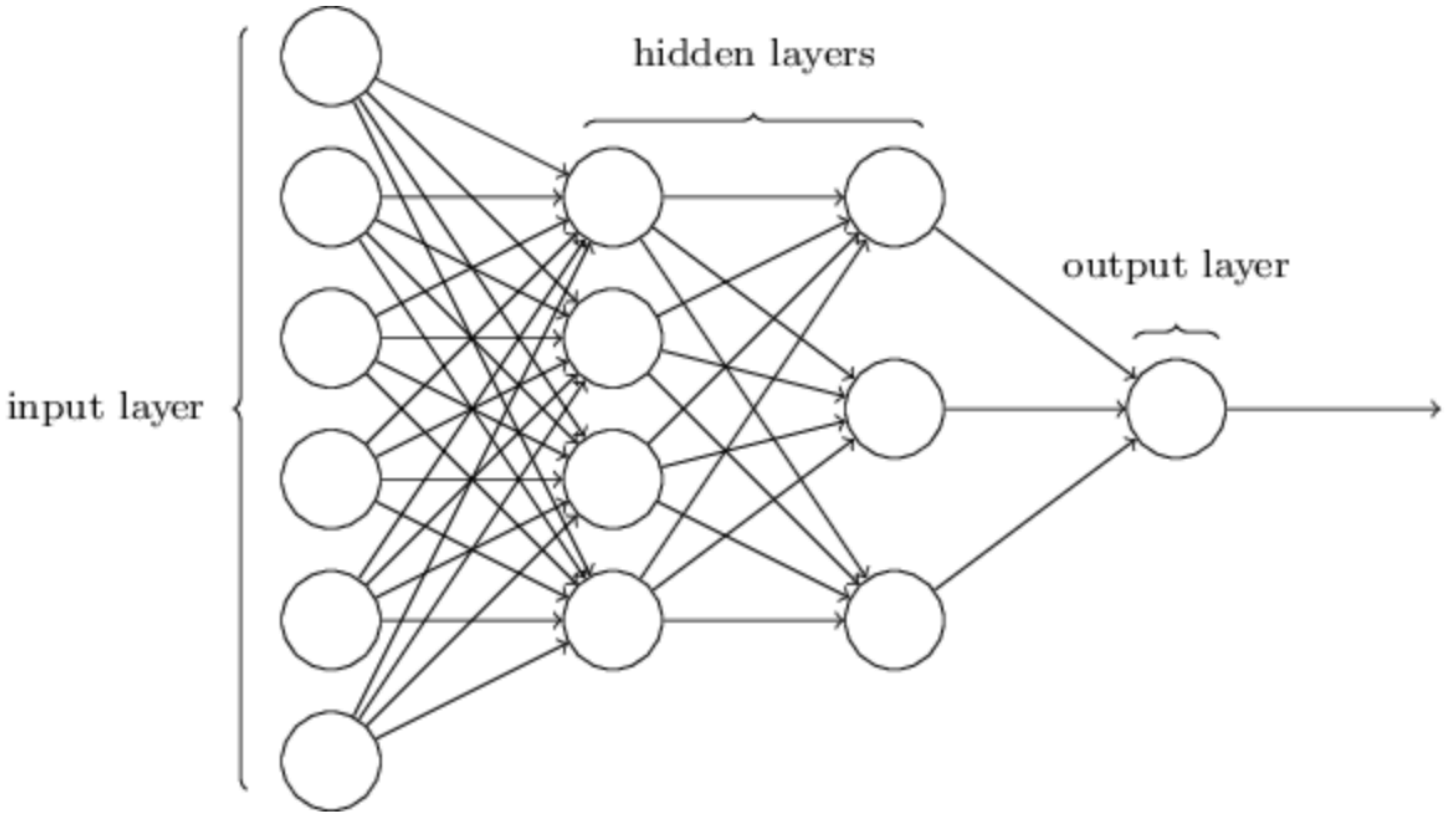
Sigmoid 神经元



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

使用神经网络来识别手写数字

神经网络的结构

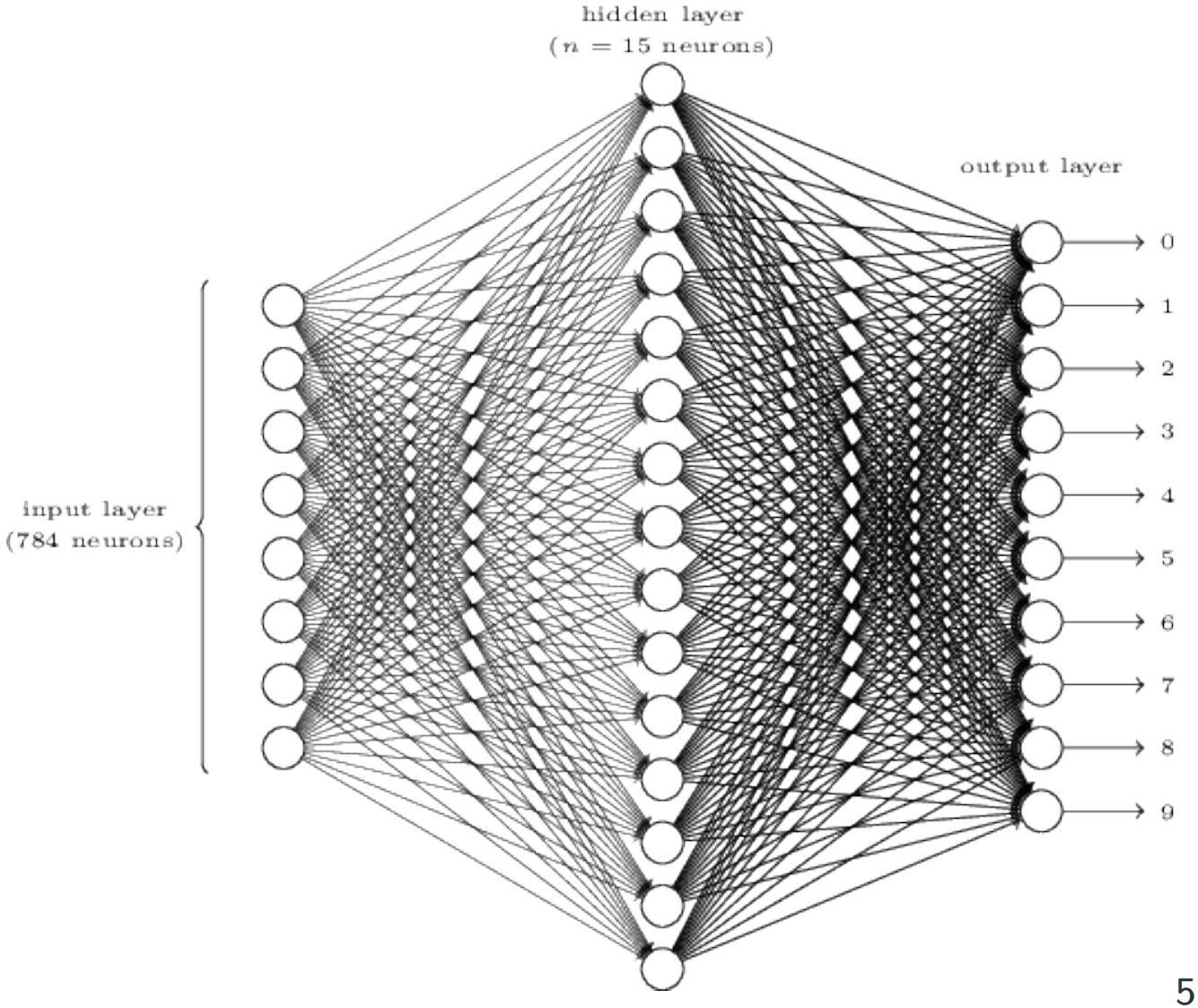


使用神经网络来识别手写数字

分类手写数字的简单网络

输入层：将图像数据编码为神经网络可接受的数据

输出层：将神经网络的输出编码为人可以接受的数据



损失函数 Cost Function

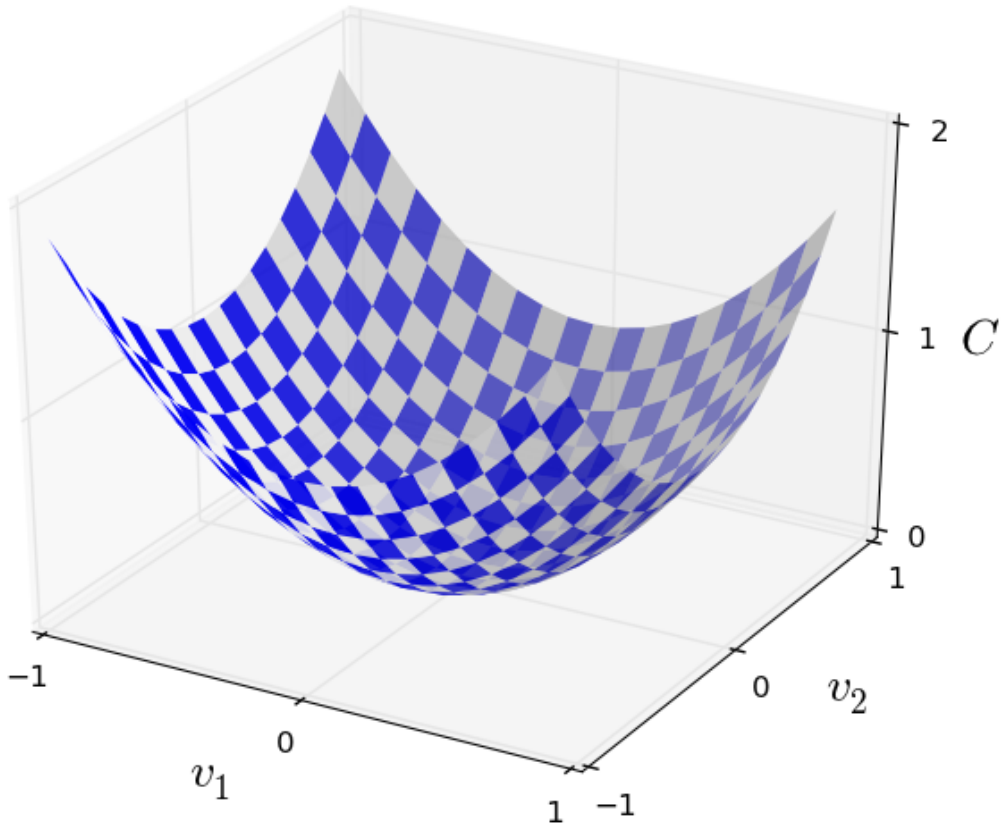
$$C(w,b) = \frac{1}{2n}\sum_{x} ||y(x) - a||^2$$

$y(x)$ 是神经网络的预测值

a 是标签值 (实际值)

n 是训练样本的数量

二维情况下均方误差损失函数的函数曲线图



使用神经网络来识别手写数字

梯度下降法 Gradient Descent

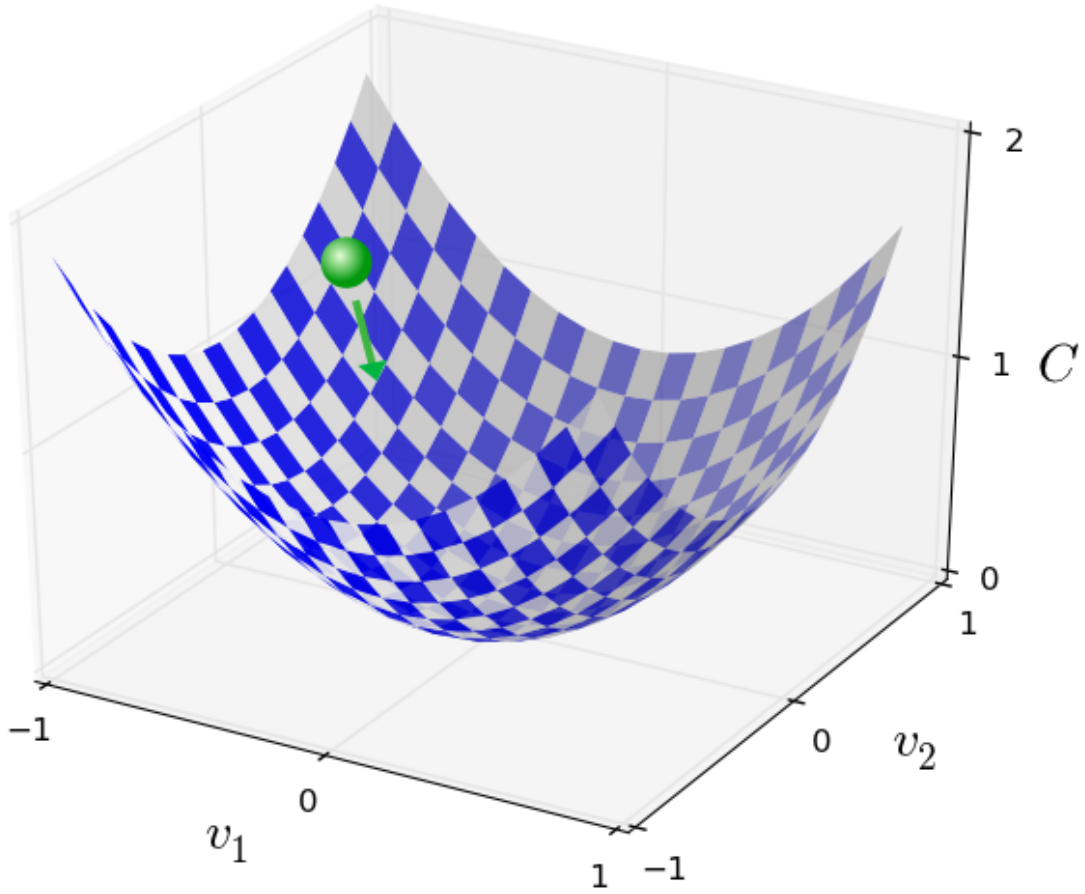
$$w' = w - \eta \frac{\partial C}{\partial w}$$

$$b' = b - \eta \frac{\partial C}{\partial b}$$

η 是神经网络的学习率

使用神经网络来识别手写数字

反映在二维情况下的梯度



使用神经网络来识别手写数字

随机梯度下降法 Stochastic Gradient Descent

每次从训练样本里面选取一小块数据集来进行训练，计算并更新梯度值，这样下来，一轮就相当于进行了很多次训练，效率大大提高

$$\frac{\sum_{j=1}^{m} \nabla C_{X_j}}{m} \approx \frac{\sum_x \nabla C_x}{n} = \nabla C$$

$$w' = w - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial w}$$

$$b' = b - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial b}$$

X_j 为 *mini_batch*，也就是用来训练的一小块数据集

使用神经网络来识别手写数字

实现我们的神经网络

使用神经网络来识别手写数字

Toward 深度学习

Part II

逆传播算法的工作机制

逆传播算法的工作机制

误差逆传播算法

Summary: the equations of backpropagation

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \tag{BP1}$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \tag{BP2}$$

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j \tag{BP3}$$

$$\frac{\partial C}{\partial w^l_{j\,k}} = a^{l-1}_k \delta^l_j \tag{BP4}$$

逆传播算法的工作机制

误差逆传播算法

后向传播算法

后向传播等式给我们提供了一种计算代价函数的方法

1. **输入 x**：为输入层设置对应的激活 a^1
2. **向前反馈**：对于每一层 $l=2,3,\cdots,L$ $\Rightarrow$

   计算 $z^l = w^l a^{l-1} + b^l$ 和 $a^l = \sigma(z^l)$
3. **输出层误差 δ^L**：计算向量 $\delta^L = \nabla_a C \odot \sigma'(z^L)$
4. **后向传播误差**：对每一层 $l=L-1, L-2, \cdots, 2$ $\Leftarrow$

   计算 $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$
5. **输出**：代价函数的梯度为 $\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$ 和 $\frac{\partial C}{\partial b^l_j} = \delta^l_j$

Part III

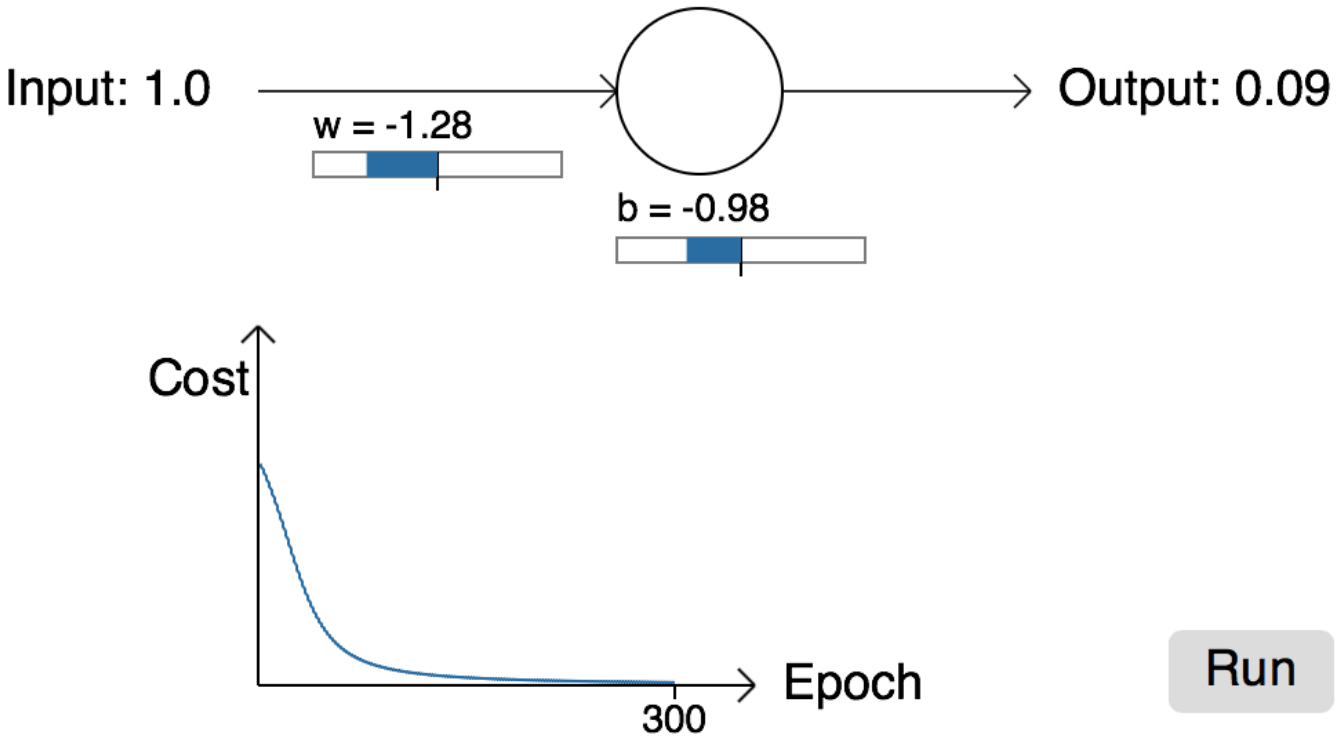
优化神经网络的学习

优化神经网络的学习

1. 交叉熵损失函数
2. 过拟合和正则化
3. 一种更好的权重初始化方法
4. 如何为神经网络选择超参数
5. 选择超参数的其他方法

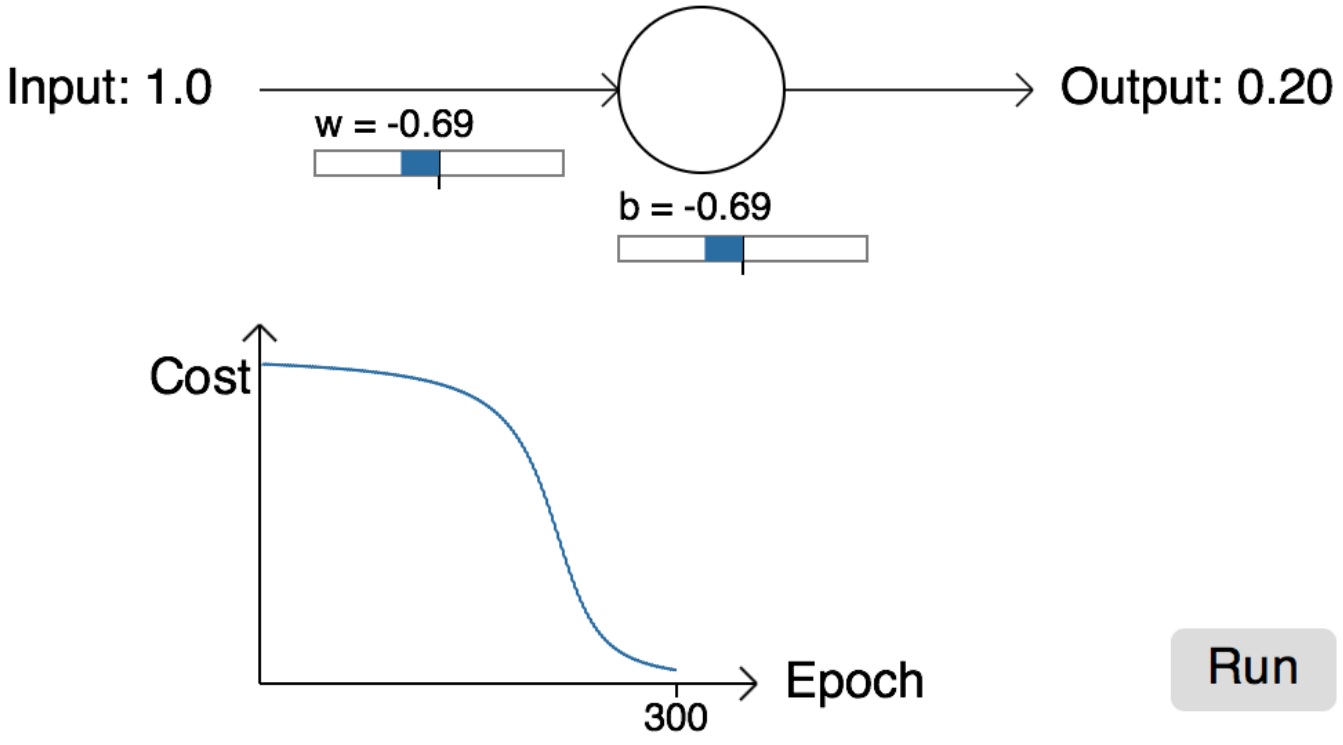
优化神经网络的学习

预测误差较小的情况下学习速率曲线图



优化神经网络的学习

预测误差较大的情况下学习速率曲线图



预测误差较大的情况下，学习速率明显降低

优化神经网络的学习

为什么在预测误差较大的情况下学习速率会降低

$$C = \frac{(y-a)^2}{2}$$

$$a = \sigma(z)$$

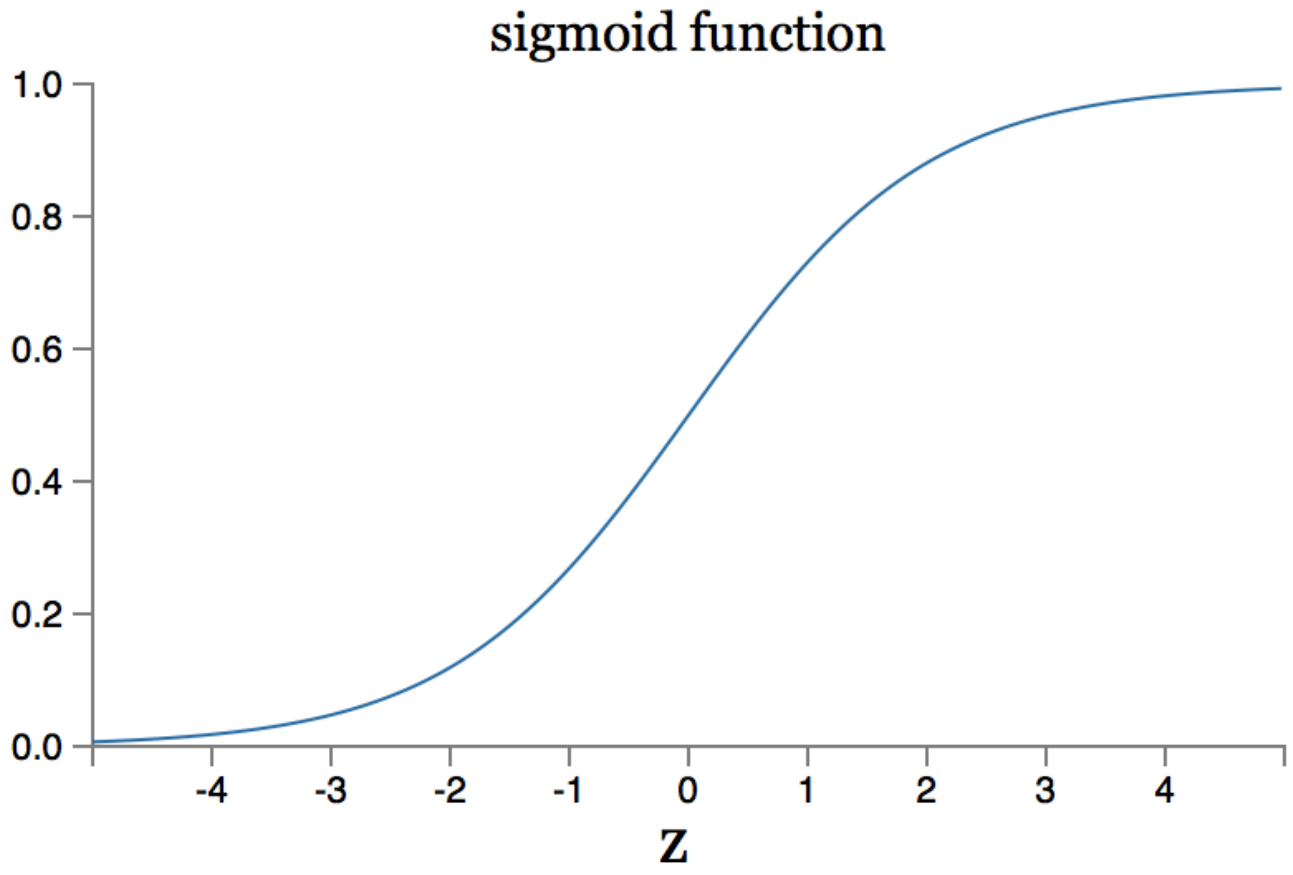
$$z = wx + b$$

$$\frac{\partial C}{\partial w} = (a-y)\sigma'(z)x = a\sigma'(z)$$

$$\frac{\partial C}{\partial b} = (a-y)\sigma'(z) = a\sigma'(z)$$

优化神经网络的学习

预测偏差很大时，**sigmoid** 函数的值接近于 **0** 或者 **1**，使得 **sigmoid** 函数的导数值趋近于 **0**



交叉熵损失函数

为了避免这一问题，提出了交叉熵损失函数。

$$C = -\frac{1}{n}\sum_{x}[y \ln a + (1-y)\ln(1-a)]$$

其中，y 为神经网络的预测值，a 为训练数据的标签值，也就是真实值

为什么交叉熵损失函数可以作为损失函数

交叉熵损失函数的形式很奇怪，怎么看都不像是一个损失函数
我们可以这样想：
首先，$C > 0$，因为 a 和 y 都在 $[0, 1]$ 之间取值，这使得累加项全部小于 0，最后加了个负号使得 $C > 0$ 成立
其次，当神经元的输出和期望值接近的时候，C 的值接近于 0，假定 $y = a$，a 趋于 0 的时候，右边是 0，左边极限算出来也是 0，所以 C 是趋于 0 的

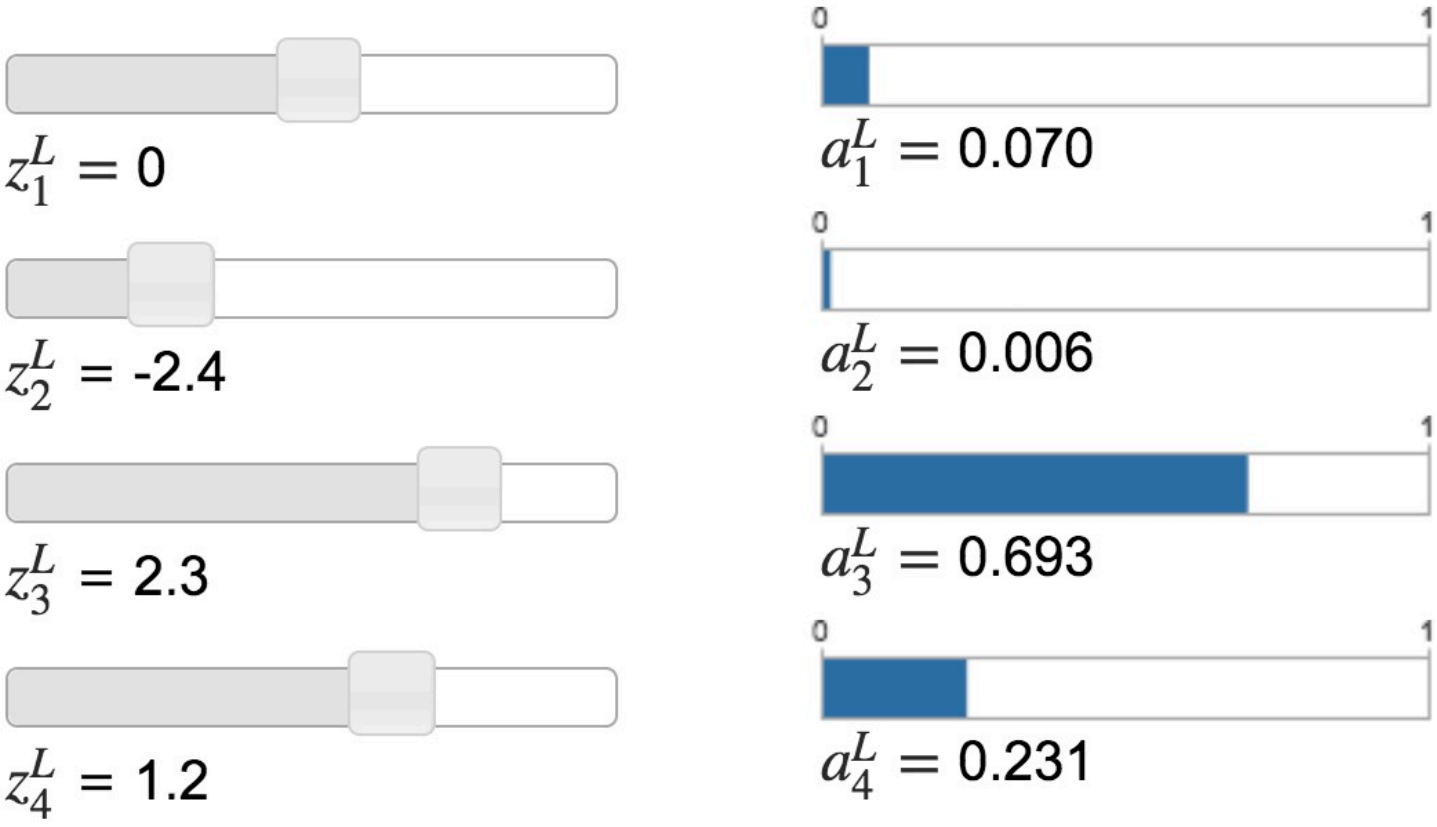
为什么交叉熵损失函数解决了学习速率减慢的问题

$$\frac{\partial C}{\partial w_j} = \frac{1}{n}\sum_x x_j(\sigma(z) - y)$$

$$\frac{\partial C}{\partial b} = \frac{1}{n}\sum_x (\sigma(z) - y)$$

我们可以看到，损失函数的偏导数不再与 $\sigma'(z)$ 有关，而是与 $\sigma(z) - y$ 有关。当我们的神经网络预测偏差很大的时候，$\sigma(z) - y$ 的值就会偏高，学习速率减慢的问题就不复存在了。

SoftMax



SoftMax

SoftMax 是用于多分类的，将整个网络的输出结果转换为 [0-1] 之间的值，这些值加起来和为 1。我们可以将这个值当作概率值。

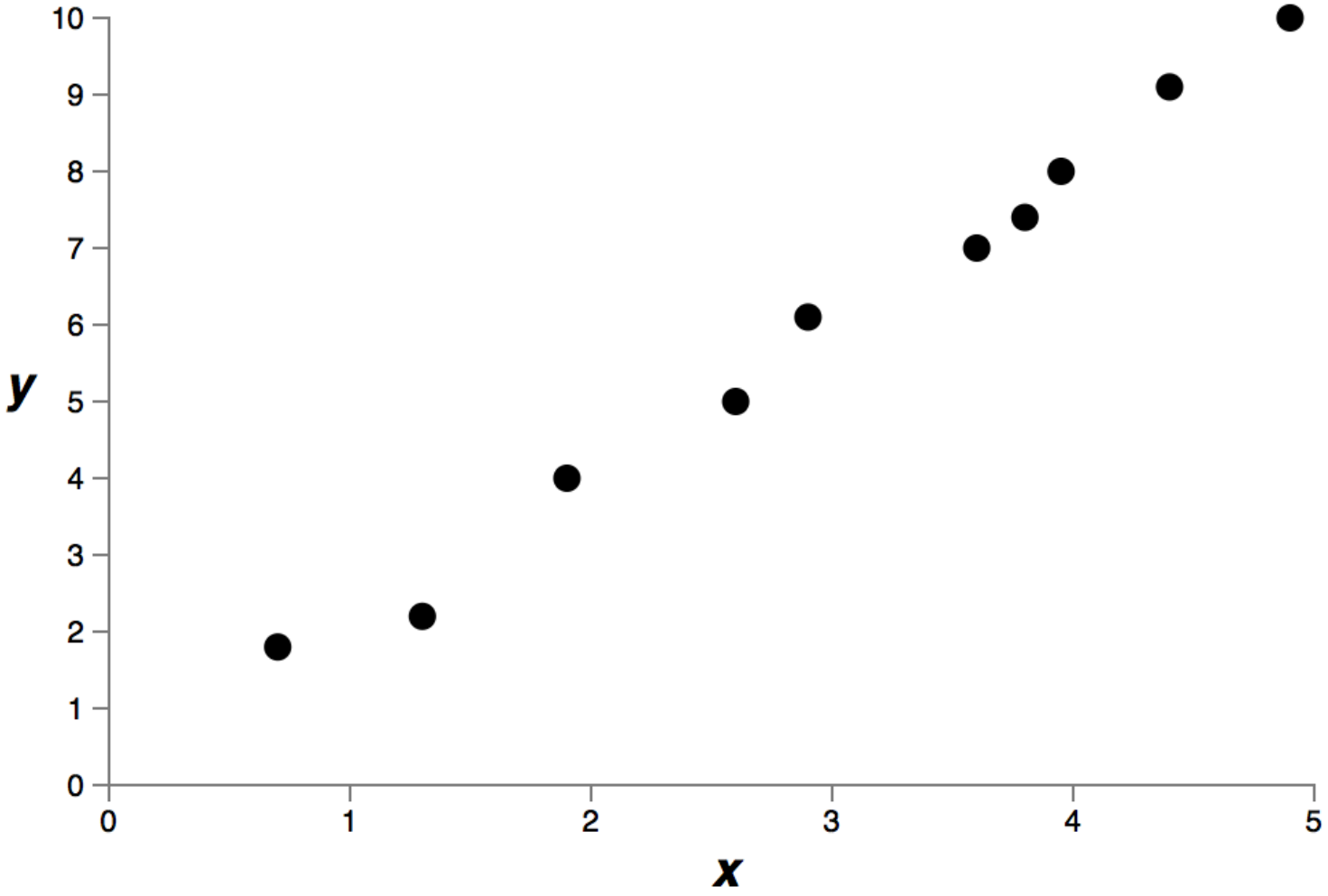
过拟合和正则化

过拟合问题是神经网络在训练的时候要极力避免的问题

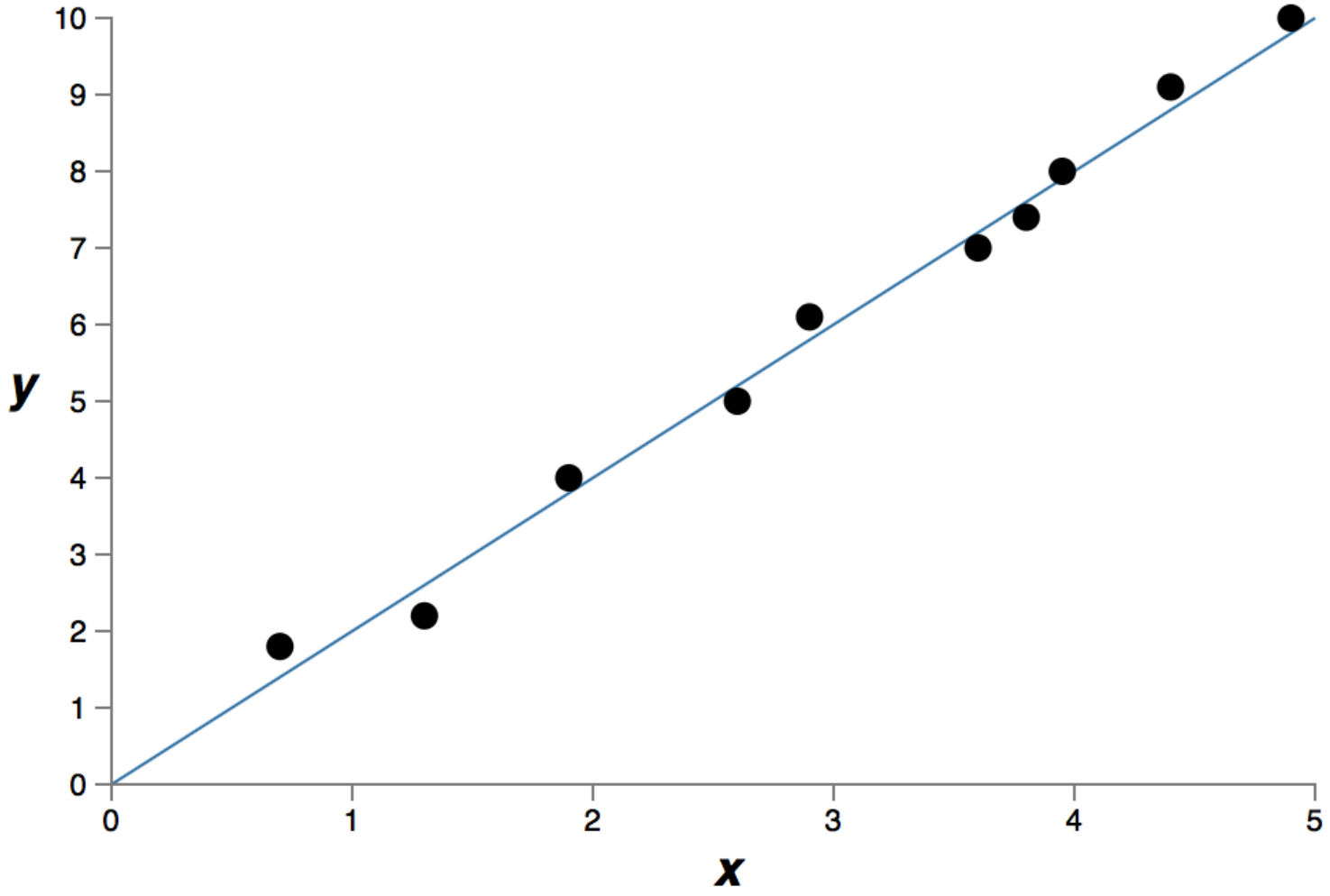
什么是过拟合问题?

优化神经网络的学习

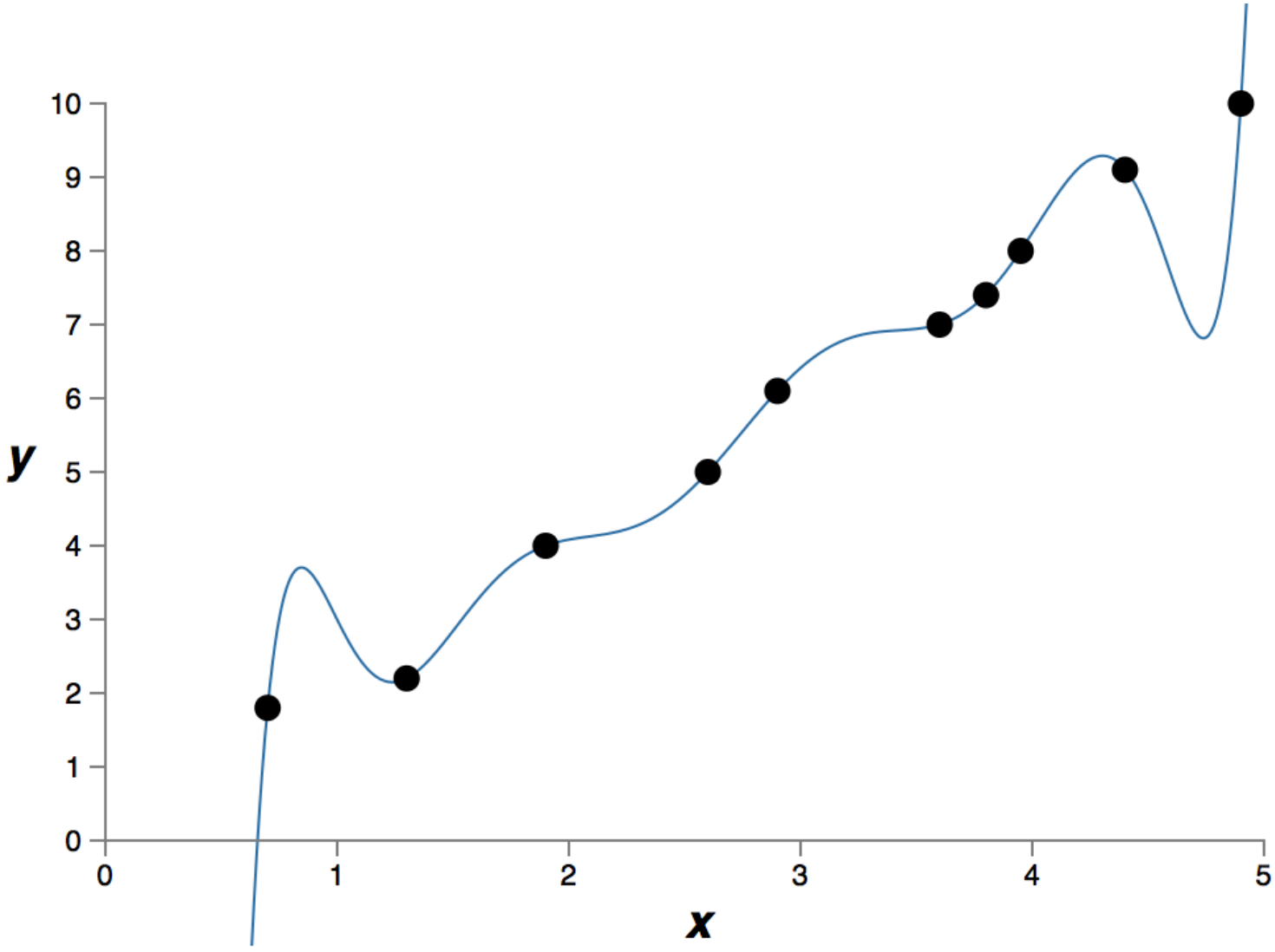
假设这是我们要拟合的一堆数据点



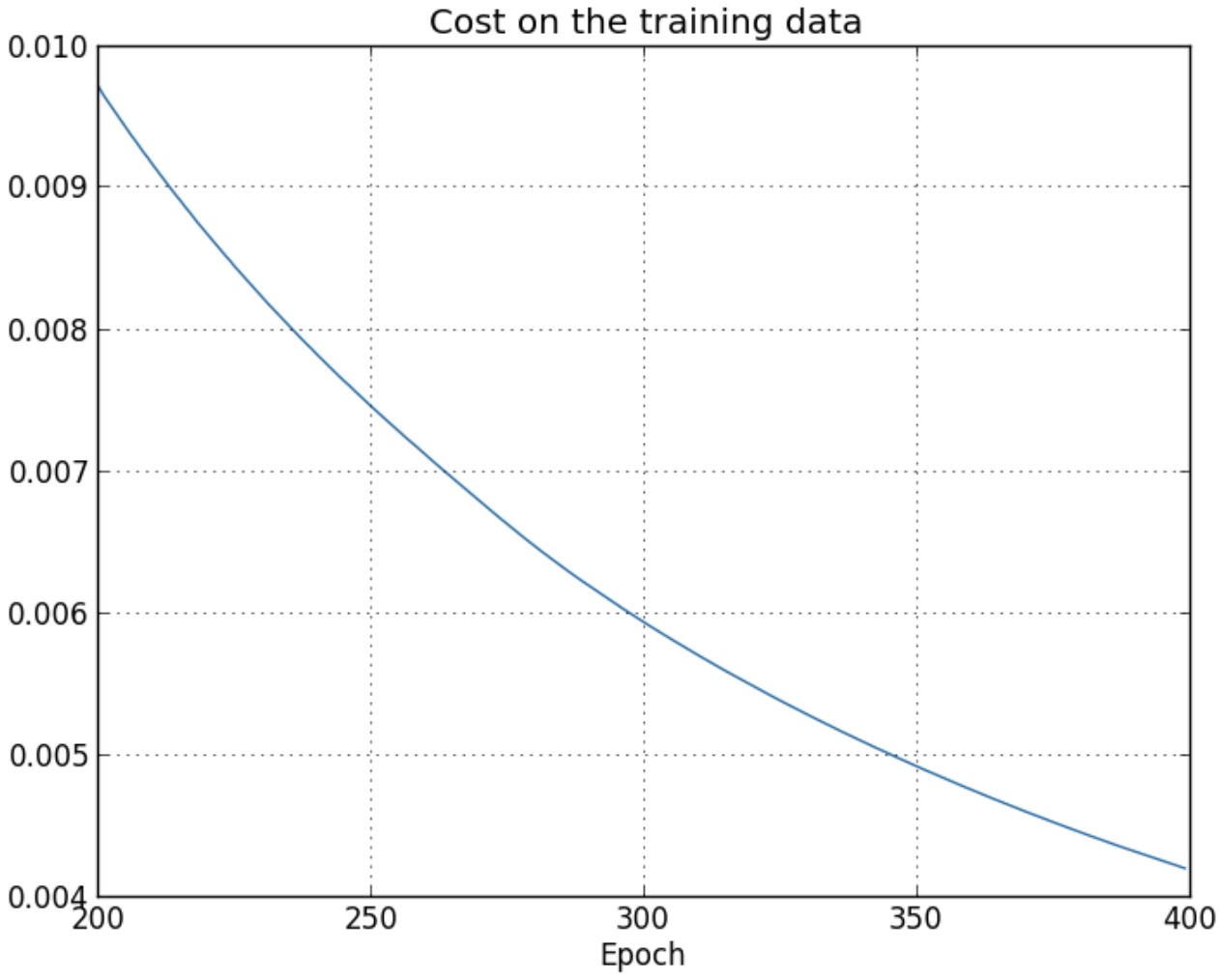
用一次函数拟合的结果



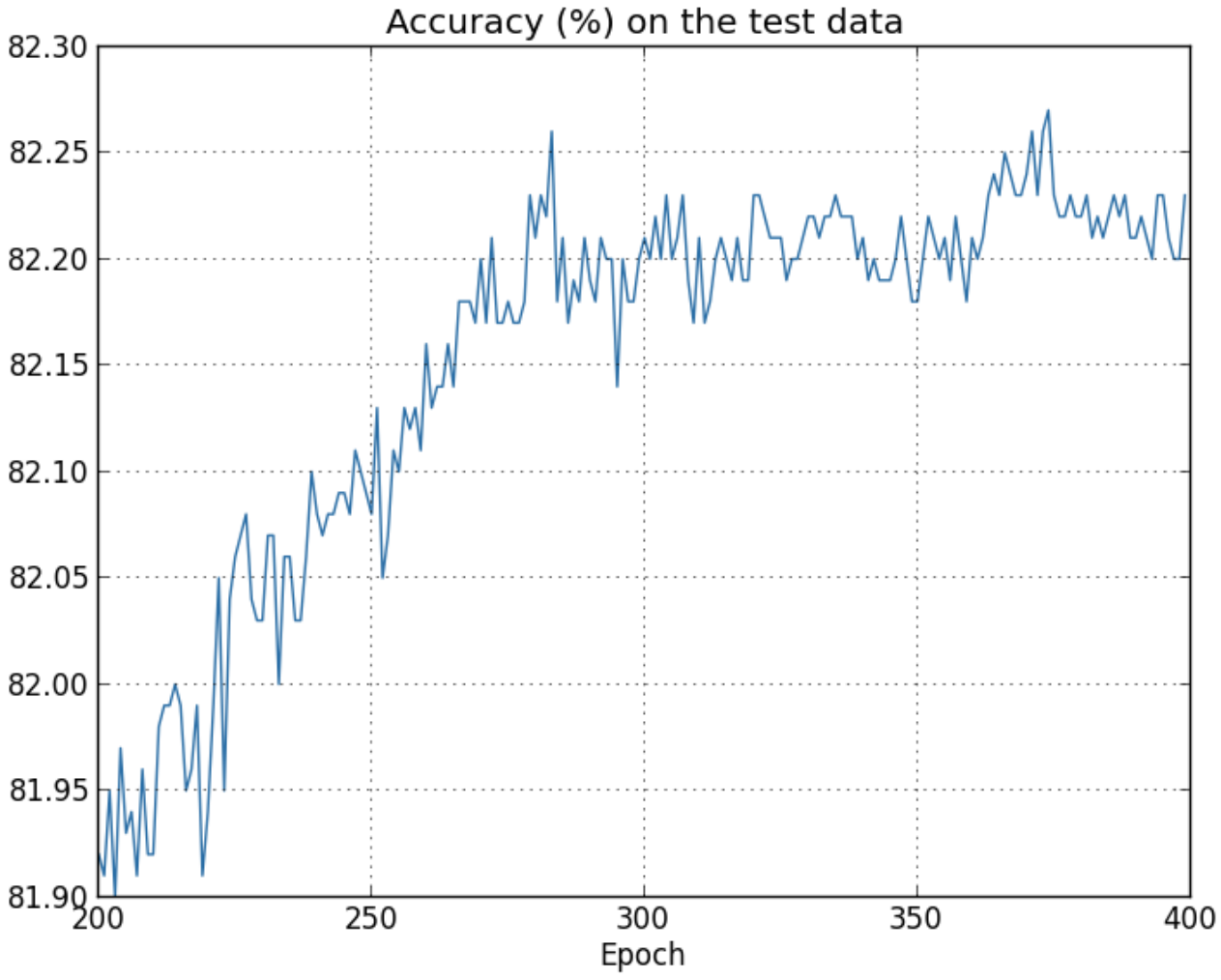
用九次函数拟合的结果



在训练集上的损失值

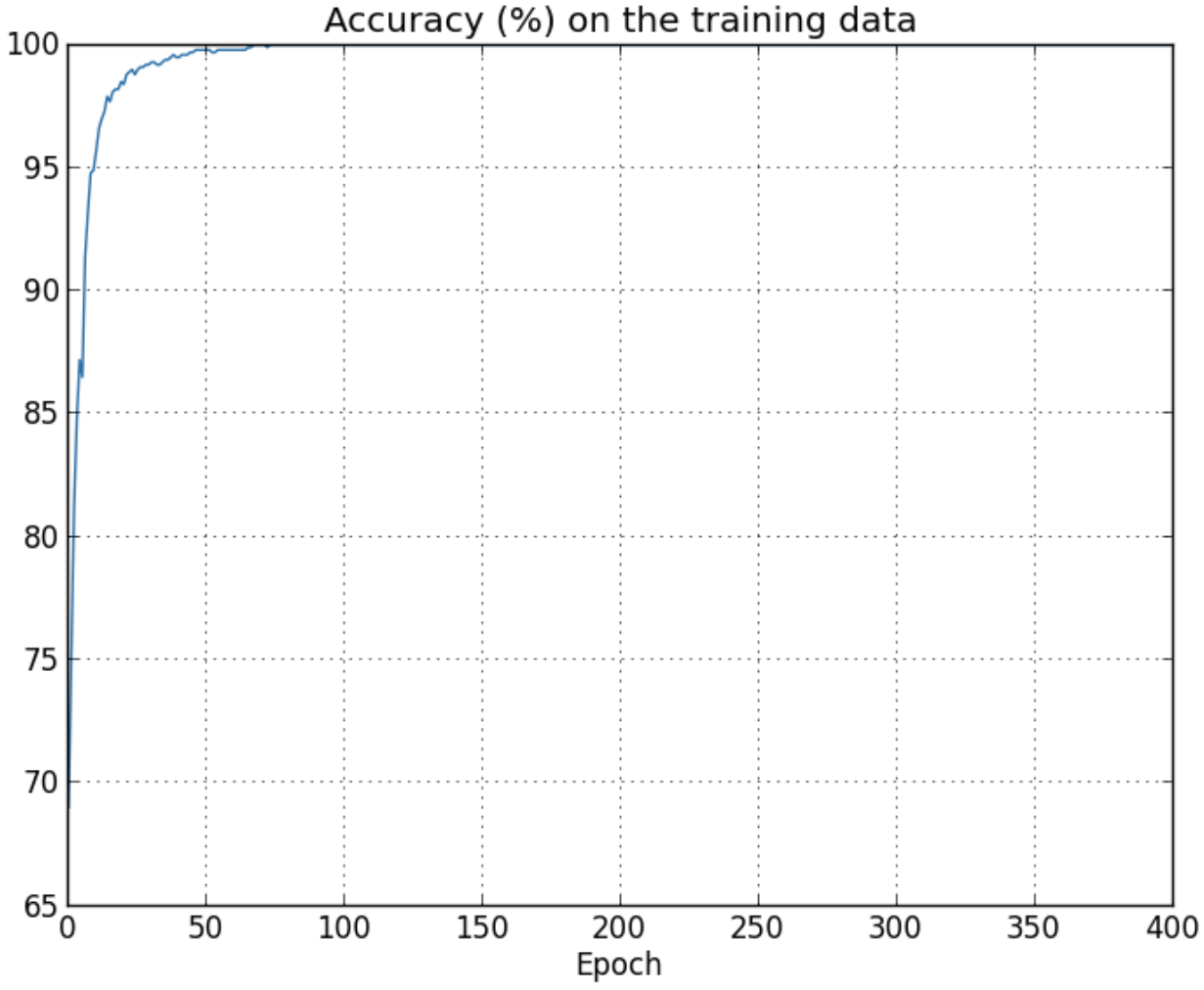

Cost on the training data
0.010
0.009
0.008
0.007
0.006
0.005
0.004
200
250
300
350
400
Epoch

在测试集上的准确率

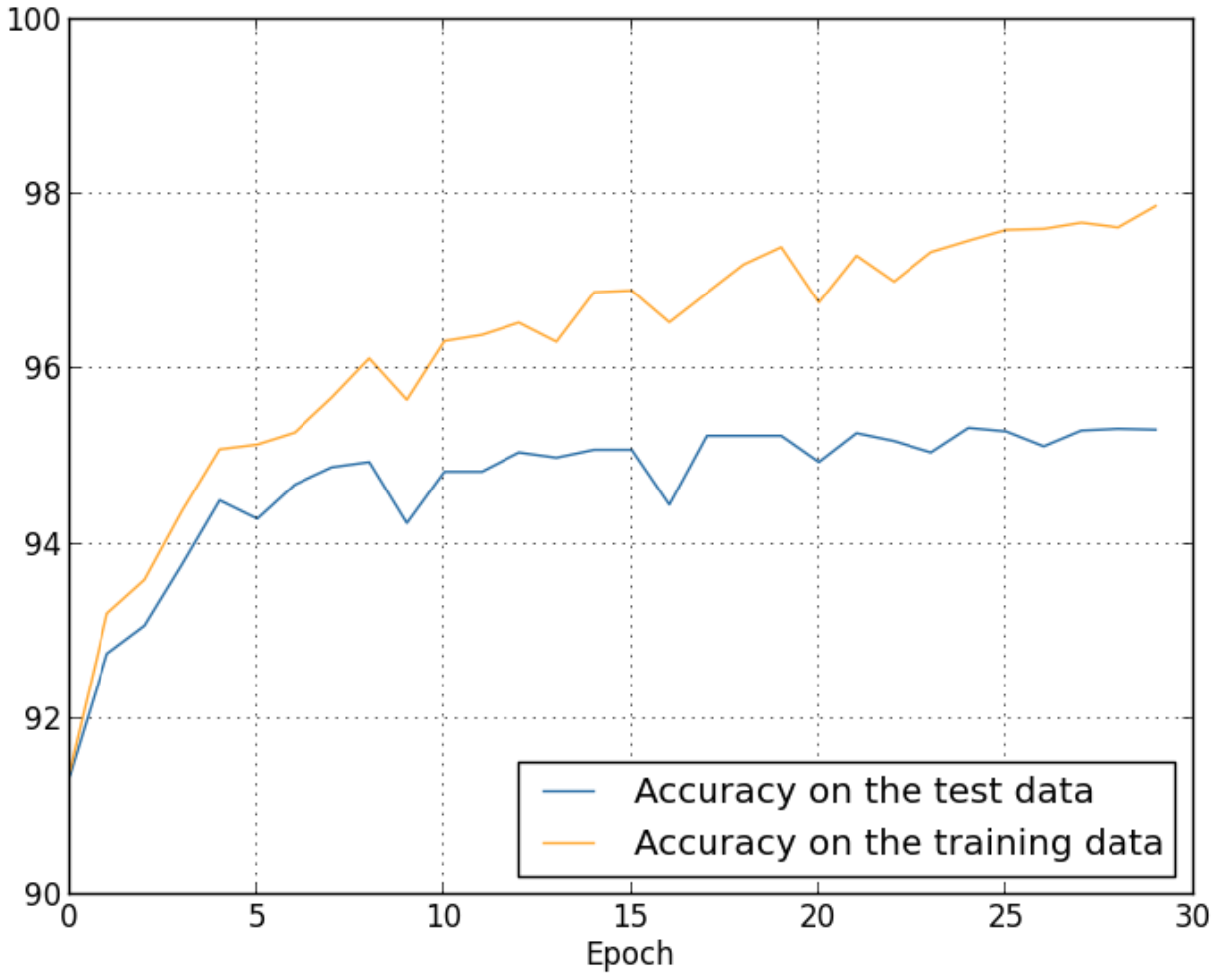

Accuracy (%) on the test data
82.30
82.25
82.20
82.15
82.10
82.05
82.00
81.95
81.90
200
250
300
350
400
Epoch

优化神经网络的学习

在训练集上的准确率

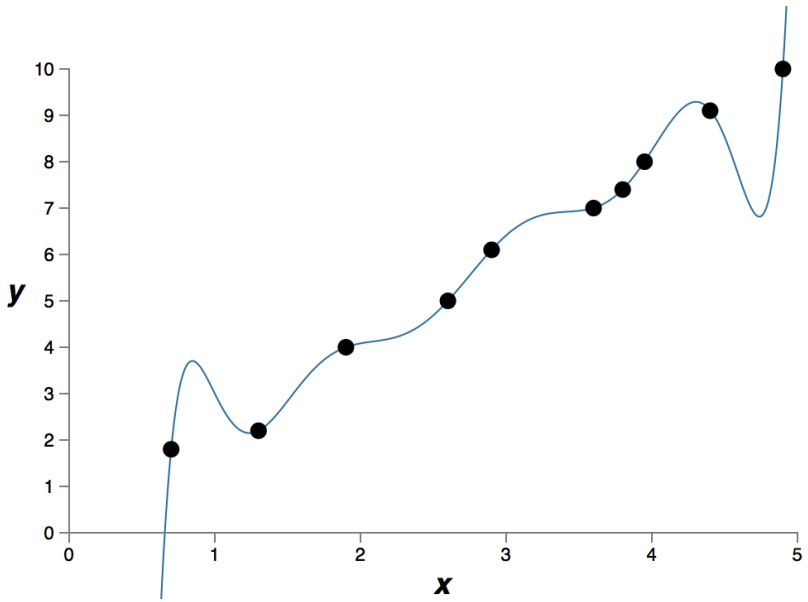


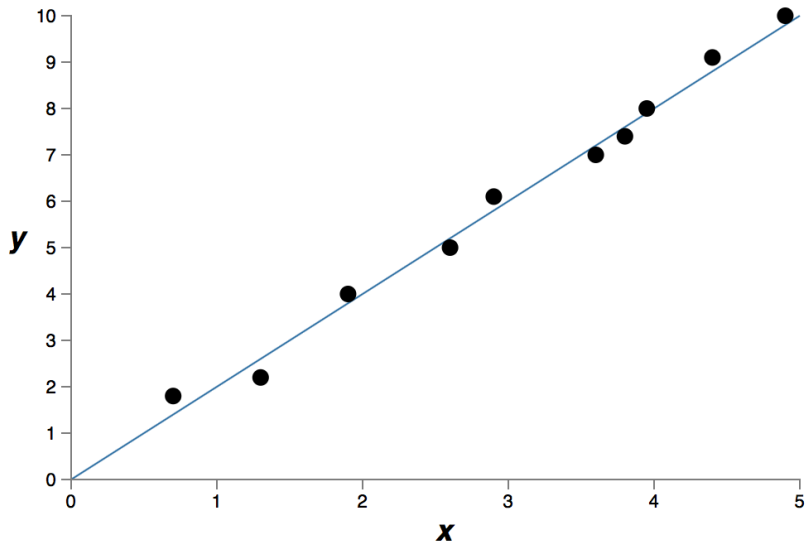
准确率的比较


100
98
96
94
92
90
0
5
10
15
20
25
30
Epoch
Accuracy on the test data
Accuracy on the training data

如何防止过拟合

思想：降低模型的复杂度，偏向于学习小的权值





正则化 (防止过拟合) 的几个方法

1. L2 正则化
2. L1 正则化
3. dropout
4. 增大训练集

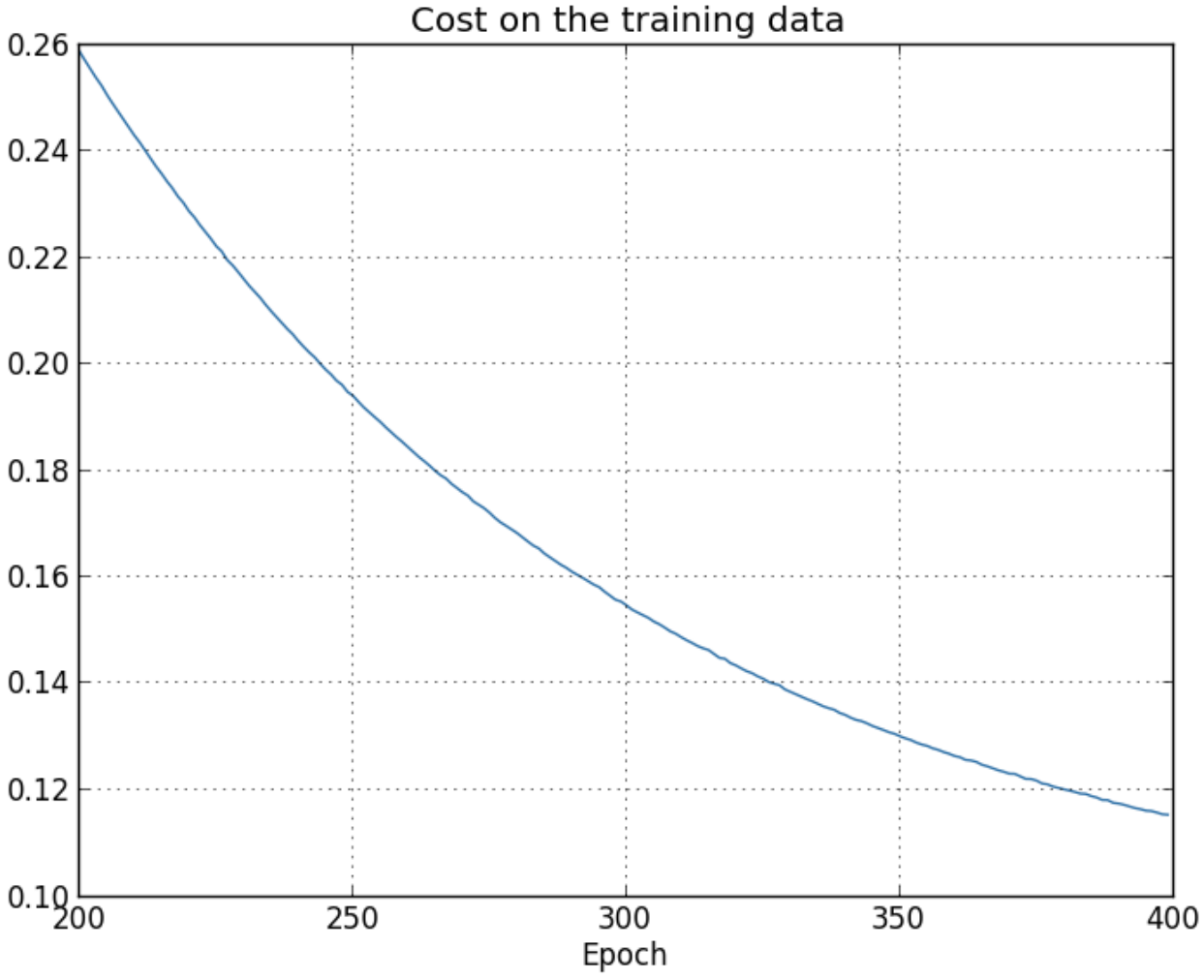
L2 正则化

L2 正则化将过高的权值也当作一种损失，使神经网络偏向于学习到低的权值

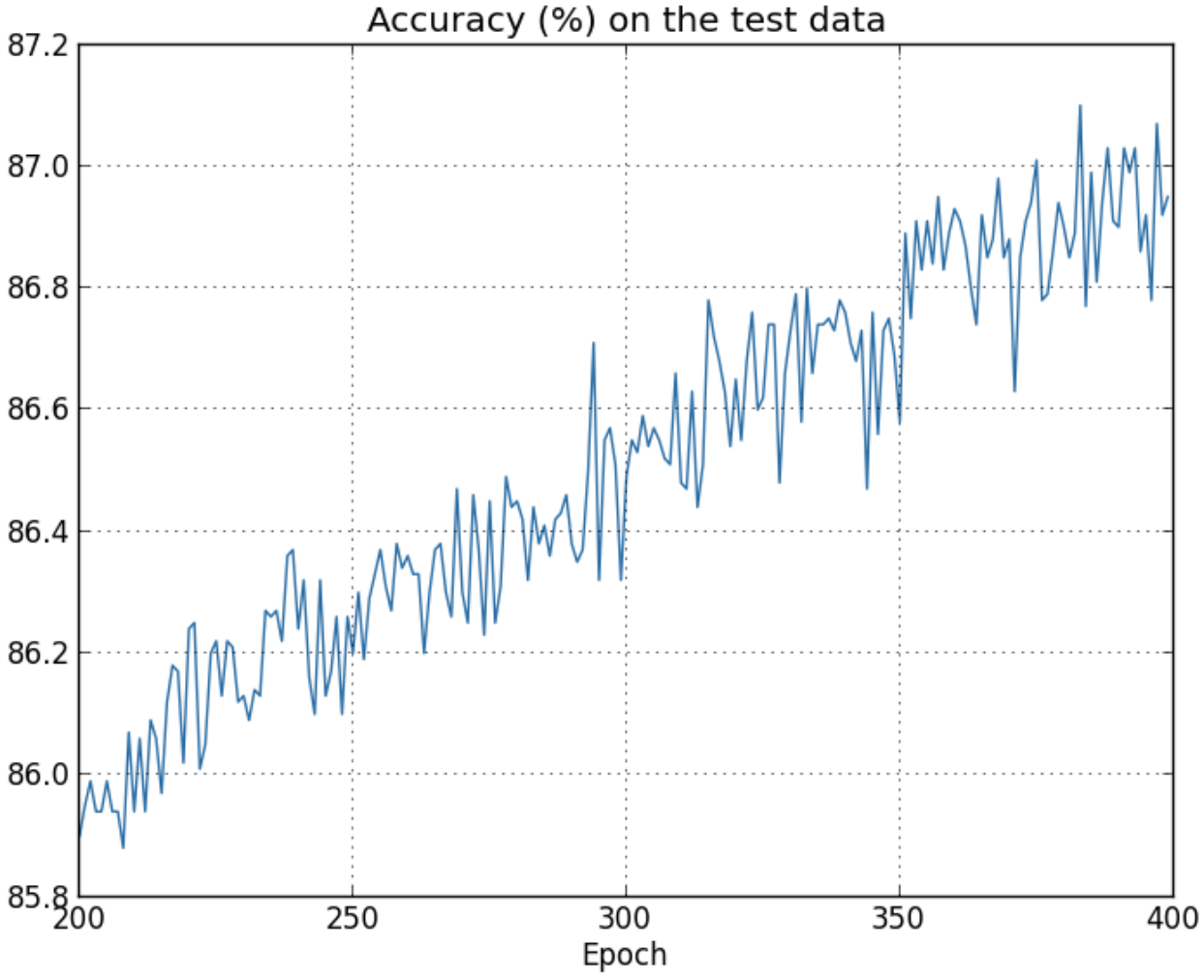
$$C = C_0 + \frac{\lambda}{2n} \sum_w w^2$$

λ 是正则项的系数

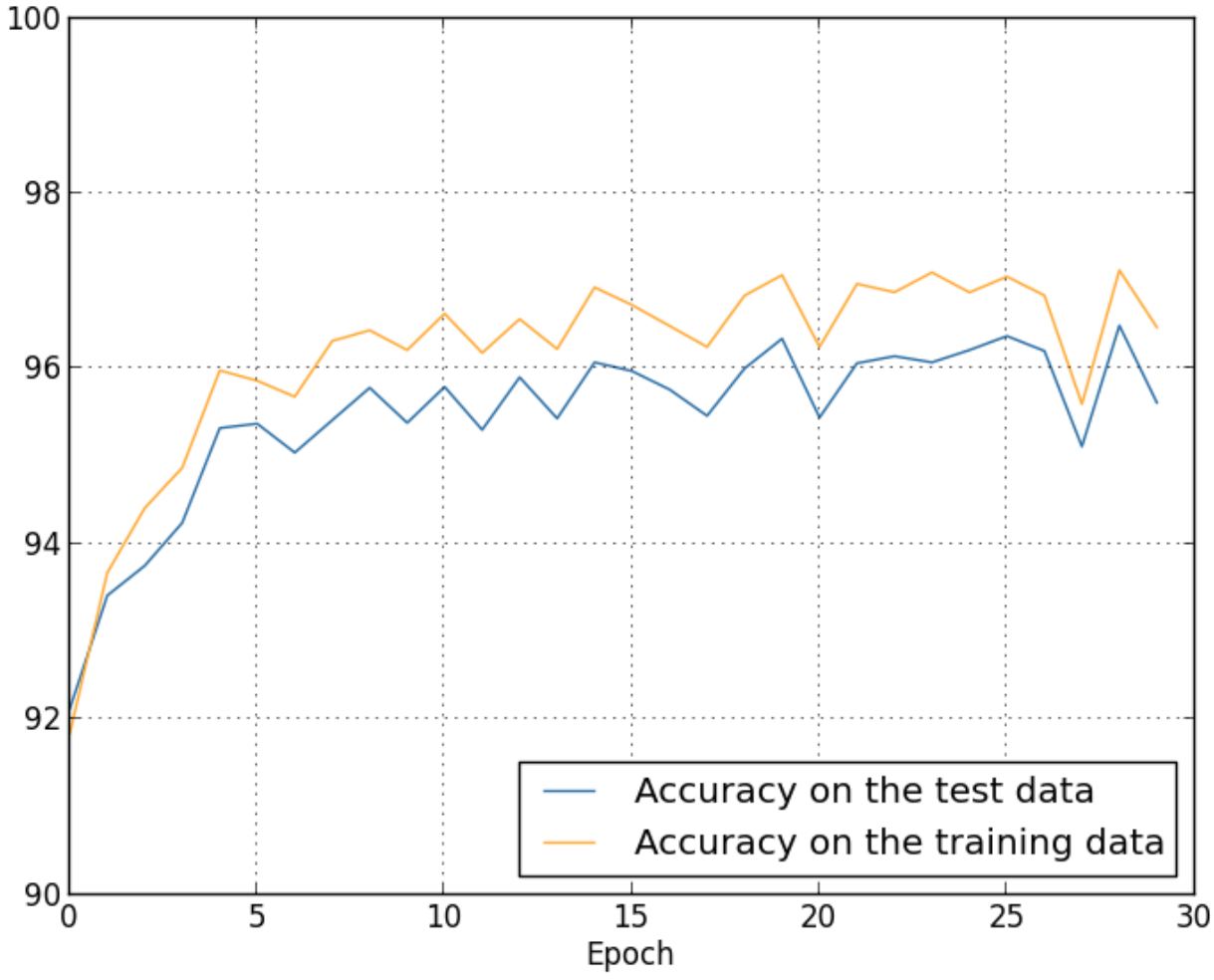
使用 L2 正则化 - 在训练集上的损失值



使用 L2 正则化 - 在测试集上的准确率



使用 L2 正则化 - 准确率的比较



L1 正则化

L1 正则化同样将过高的权值也当作一种损失，使神经网络偏向于学习到低的权值

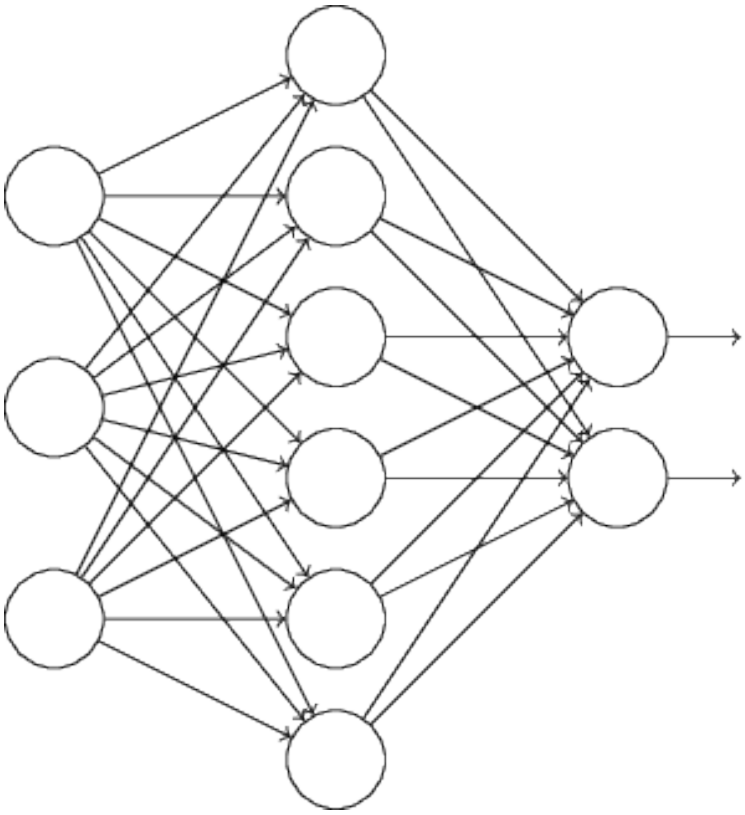
$$C = C_0 + \frac{\lambda}{n} \sum_w |w|$$

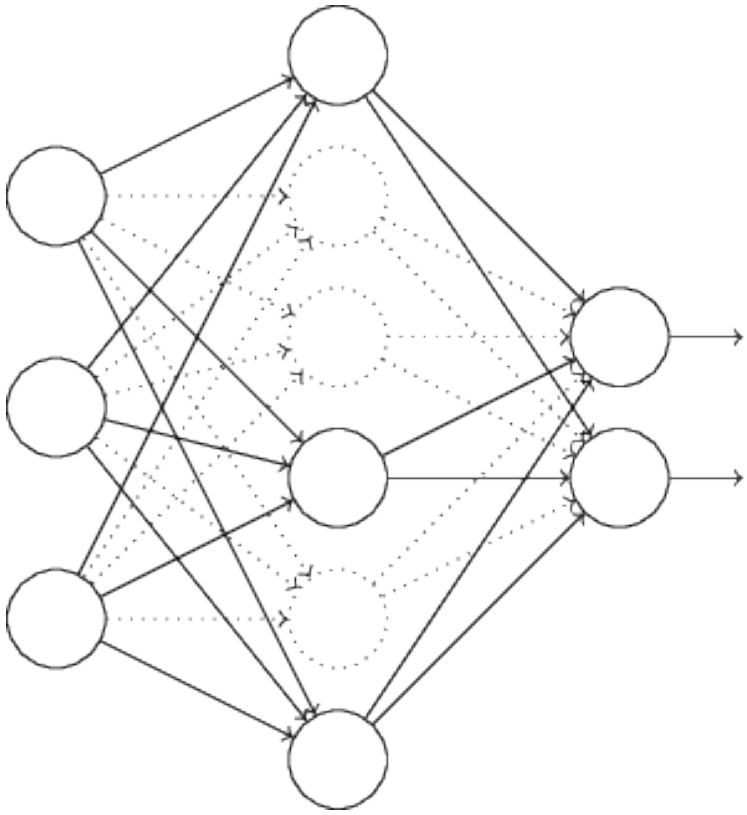
λ 是正则项的系数
L1 正则化引入了绝对值，有时候会使偏导数的值不存在，此时可以不用正则化。

优化神经网络的学习

dropout

在训练的时候随机丢弃一些神经元





其实是在训练多个网络

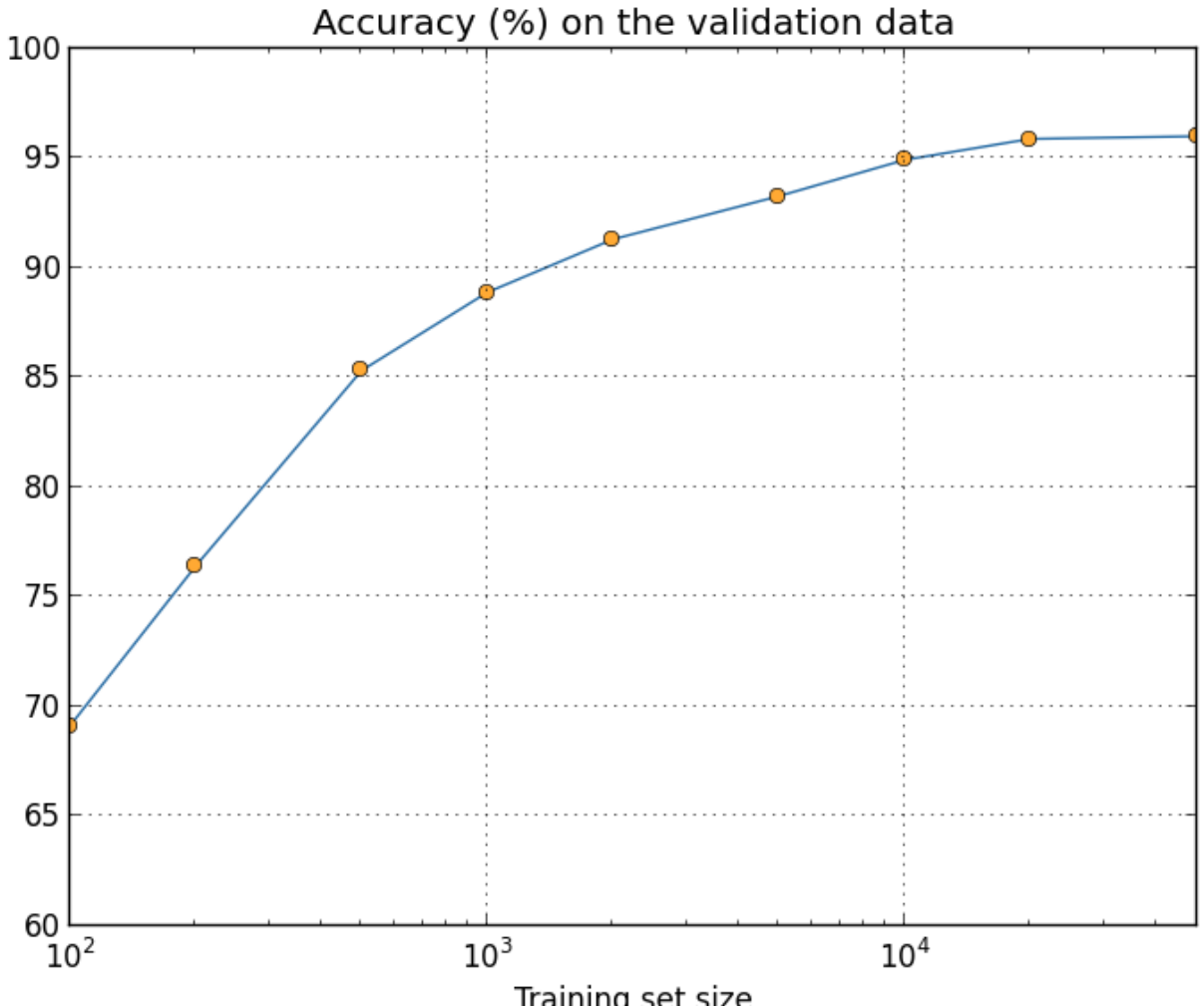
理解

dropout 其实是在同时训练多个网络，最后进行预测的时候将多个网络的结果进行平均，单个网络过拟合了，但是另一个并不一定过拟合

dropout 也是倾向于学习到小的权值，它的思想是，**丢掉一些神经元对整个网络的影响并不大**，这一点只有在权值较小的情况下才会成立

增大训练集

手动增大训练集，使过拟合的难度大大增加



一种更好的权重初始化方法

优化神经网络的学习

如何为神经网络选择超参数

选择超参数的其他方法

Part IV

三层网络可以模拟任何函数的直观证明

三层网络可以模拟任何函数的直观证明

Part V

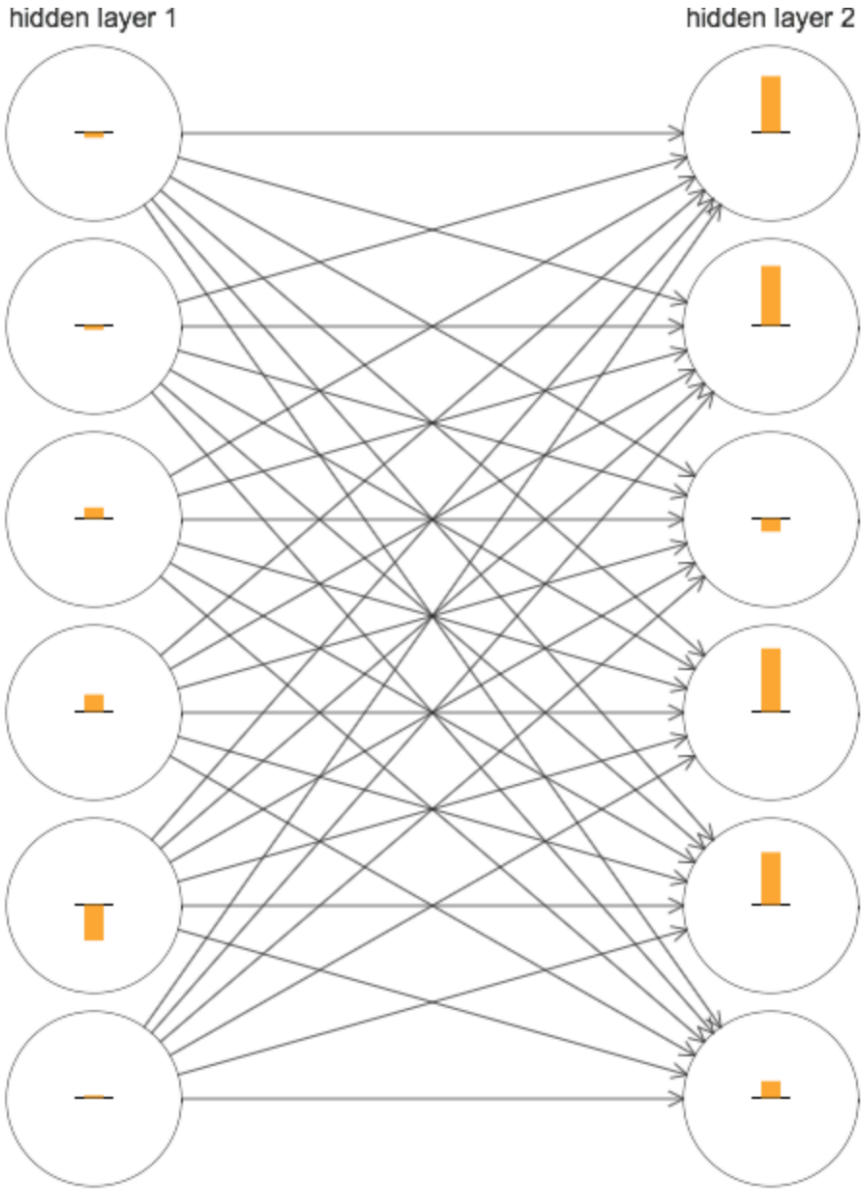
为什么深度网络难以训练

为什么深度网络难以训练

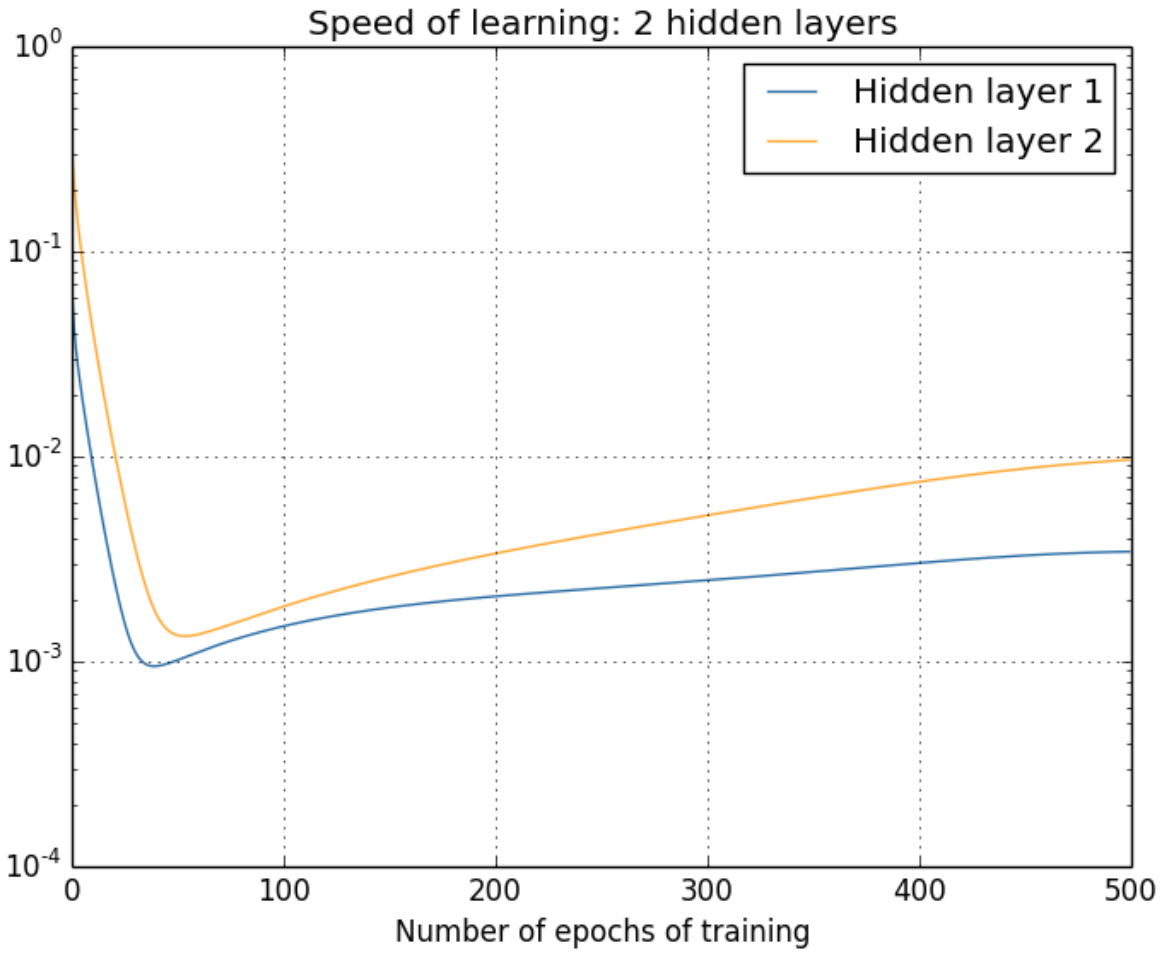
我们尝试增加隐含层的数量，评估训练的结果：

1. [784, 30, 10]：96.48%
2. [784, 30, 30, 10]：96.90%
3. [784, 30, 30, 30, 10]：96.57%
4. [784, 30, 30, 30, 30, 10]：96.53%

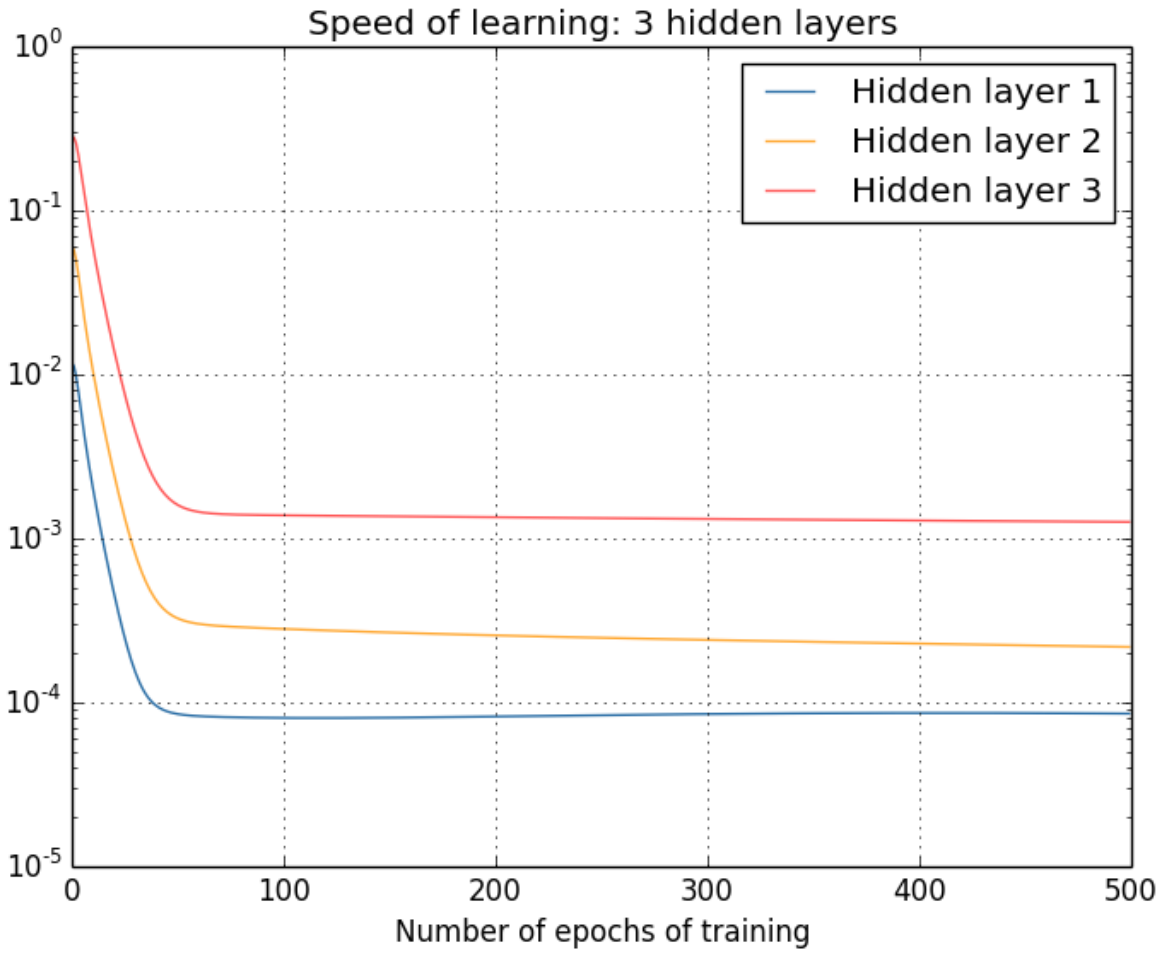
为什么深度网络难以训练



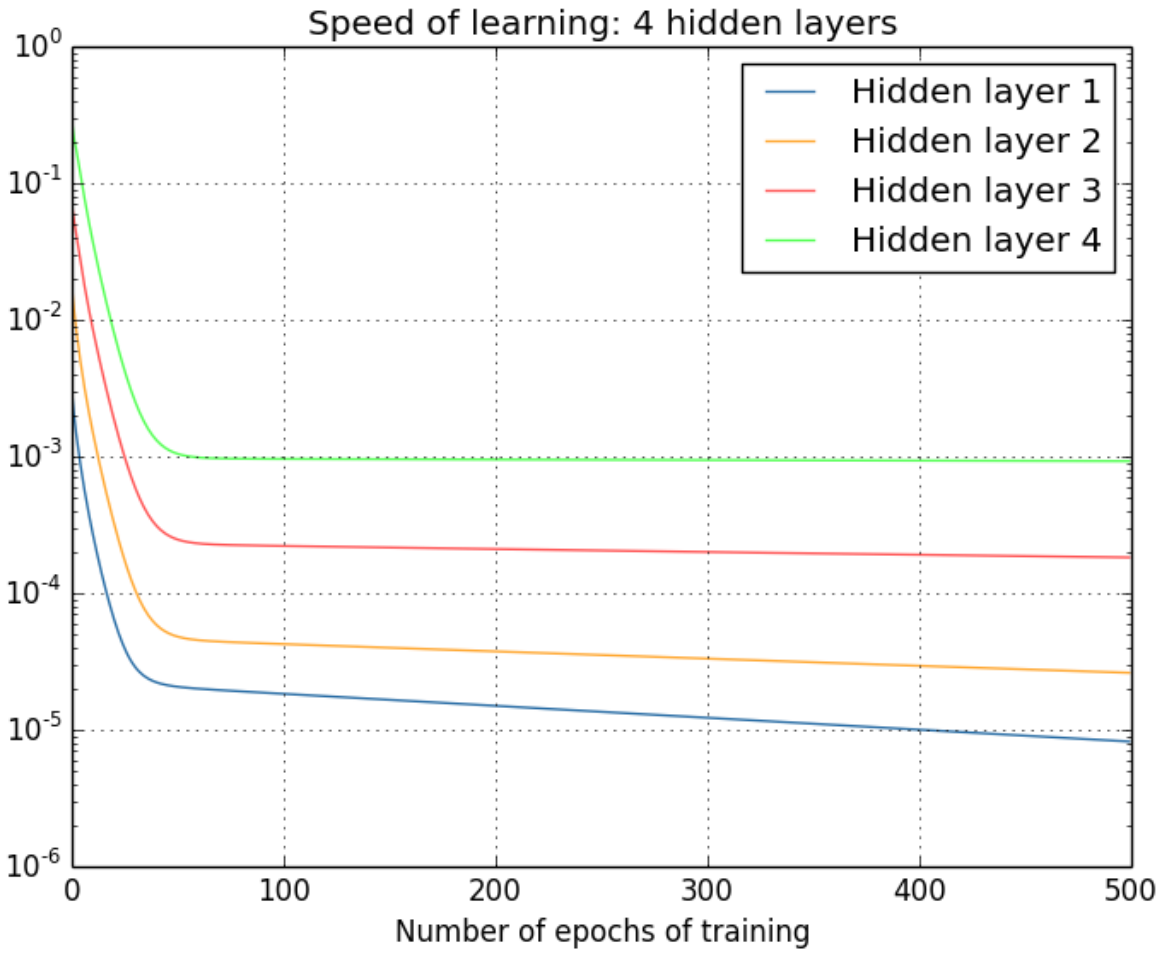
为什么深度网络难以训练



为什么深度网络难以训练



为什么深度网络难以训练



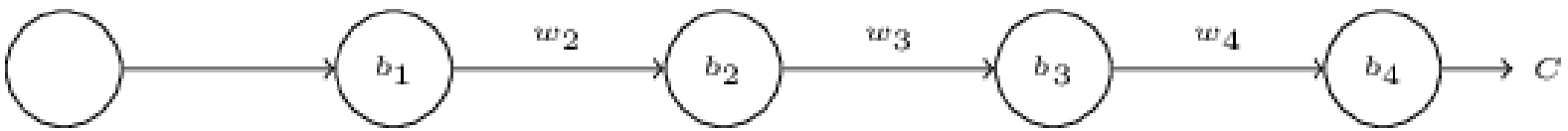
为什么深度网络难以训练

我们考虑一种最简单的情况，一共有三个隐含层，每层都只有一个神经元

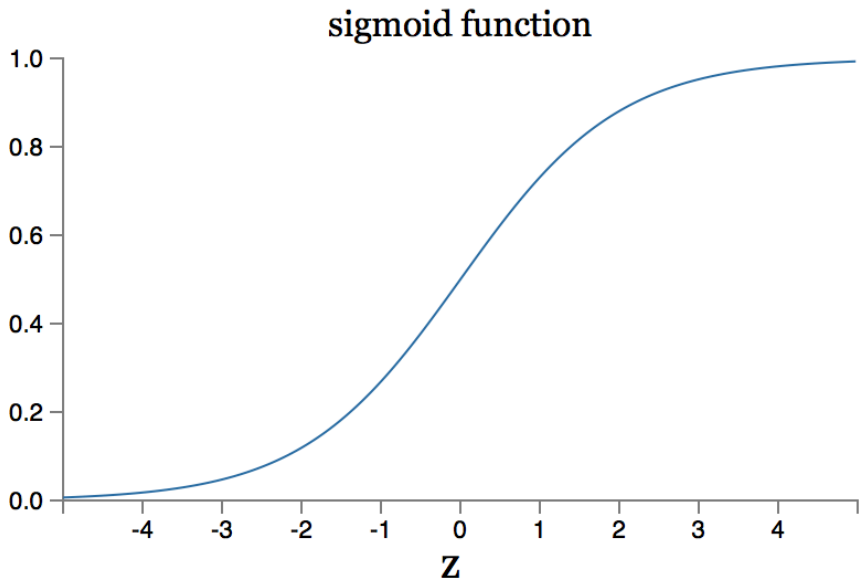
为什么深度网络难以训练

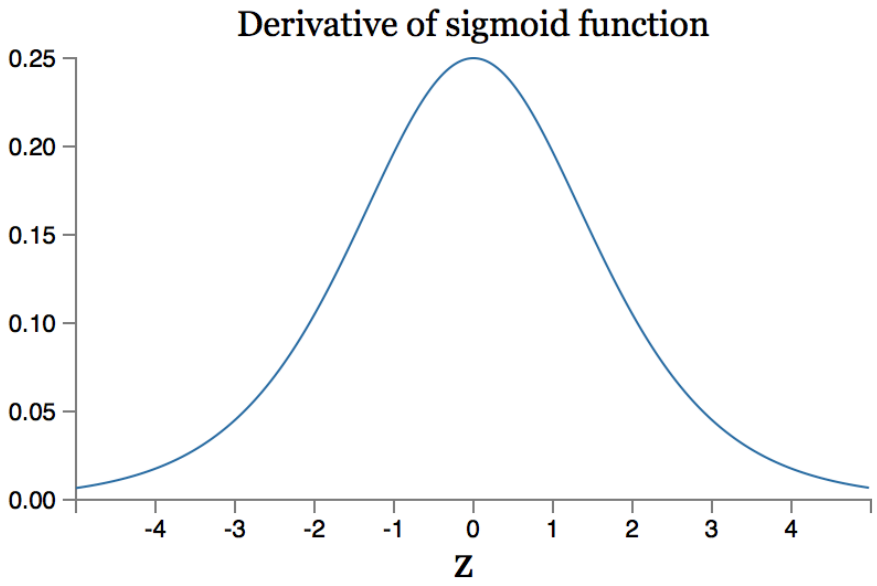
早期层的梯度是最后一层的梯度乘以各层的 sigmoid 函数的导数以及各层的权重值

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$



为什么深度网络难以训练





为什么深度网络难以训练

梯度消失问题

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1)\overbrace{w_2\sigma'(z_2)}^{<\frac{1}{4}}\overbrace{w_3\sigma'(z_3)}^{<\frac{1}{4}}\underbrace{w_4\sigma'(z_4)\frac{\partial C}{\partial a_4}}$$

common terms

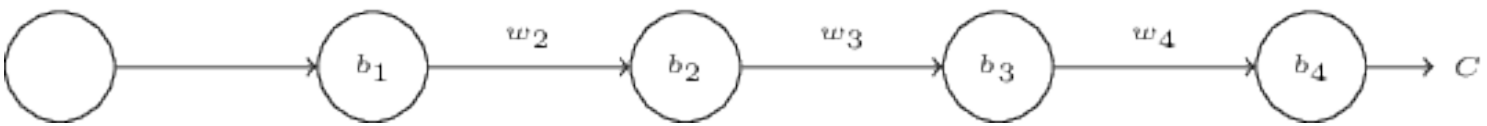
$$\overbrace{\frac{\partial C}{\partial b_3} = \sigma'(z_3)\, w_4\sigma'(z_4)\frac{\partial C}{\partial a_4}}$$

早期层的梯度值随网络的层数呈指数级的递减

为什么深度网络难以训练

梯度爆炸问题

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$



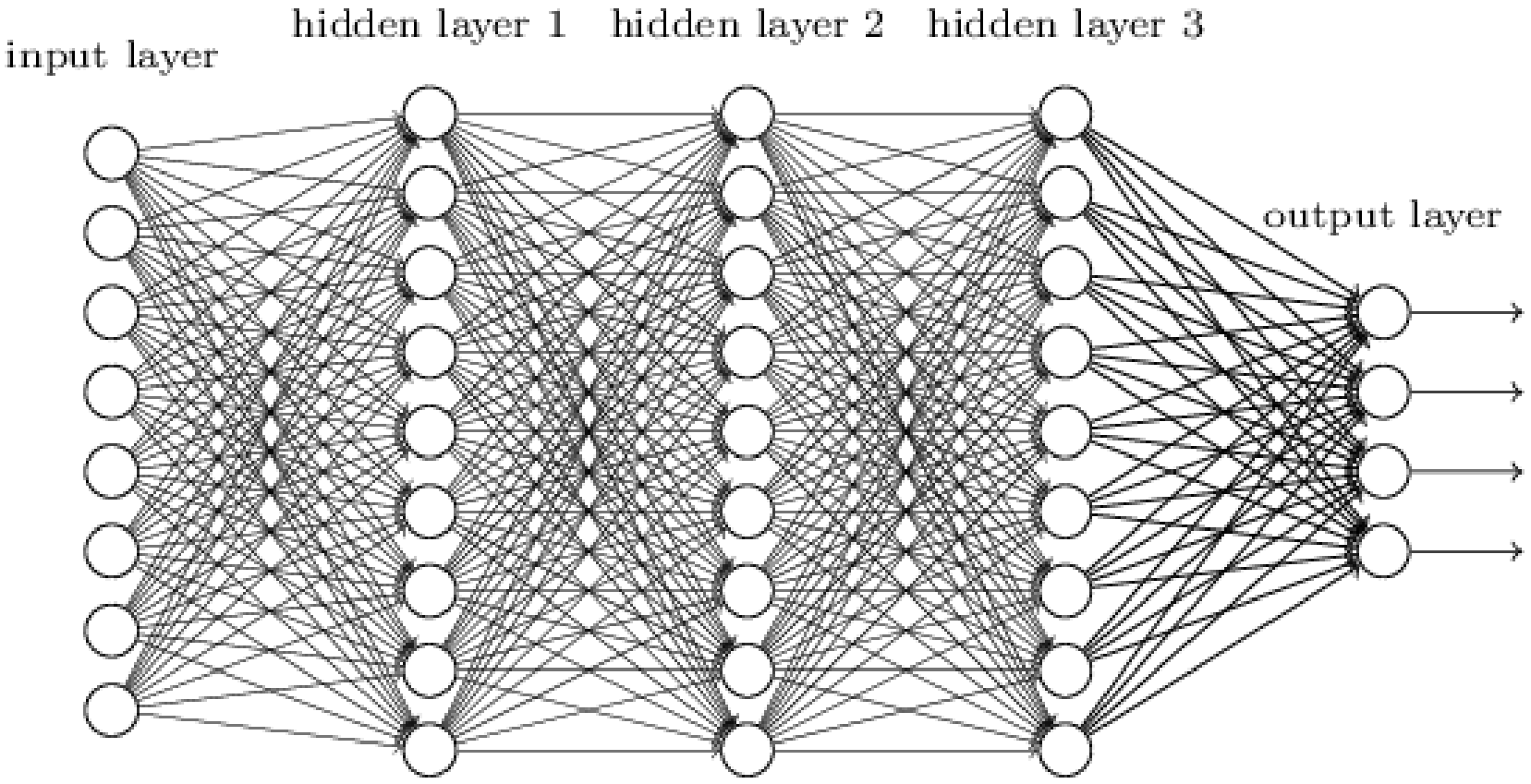
假定权值为 100，$\sigma'(z) = \frac{1}{4}$，这样式子中的每一项就都是 25。只要最后一层的梯度不是太小，那早期层的梯度就会随网络的层数呈指数级的增长。

为什么深度网络难以训练

不稳定的梯度问题

问题的根本不是梯度消失或者梯度爆炸，而是早期层的梯度是所有后面层梯度项的乘积。当有很多层时，这就是一个本质上不稳定的情形。

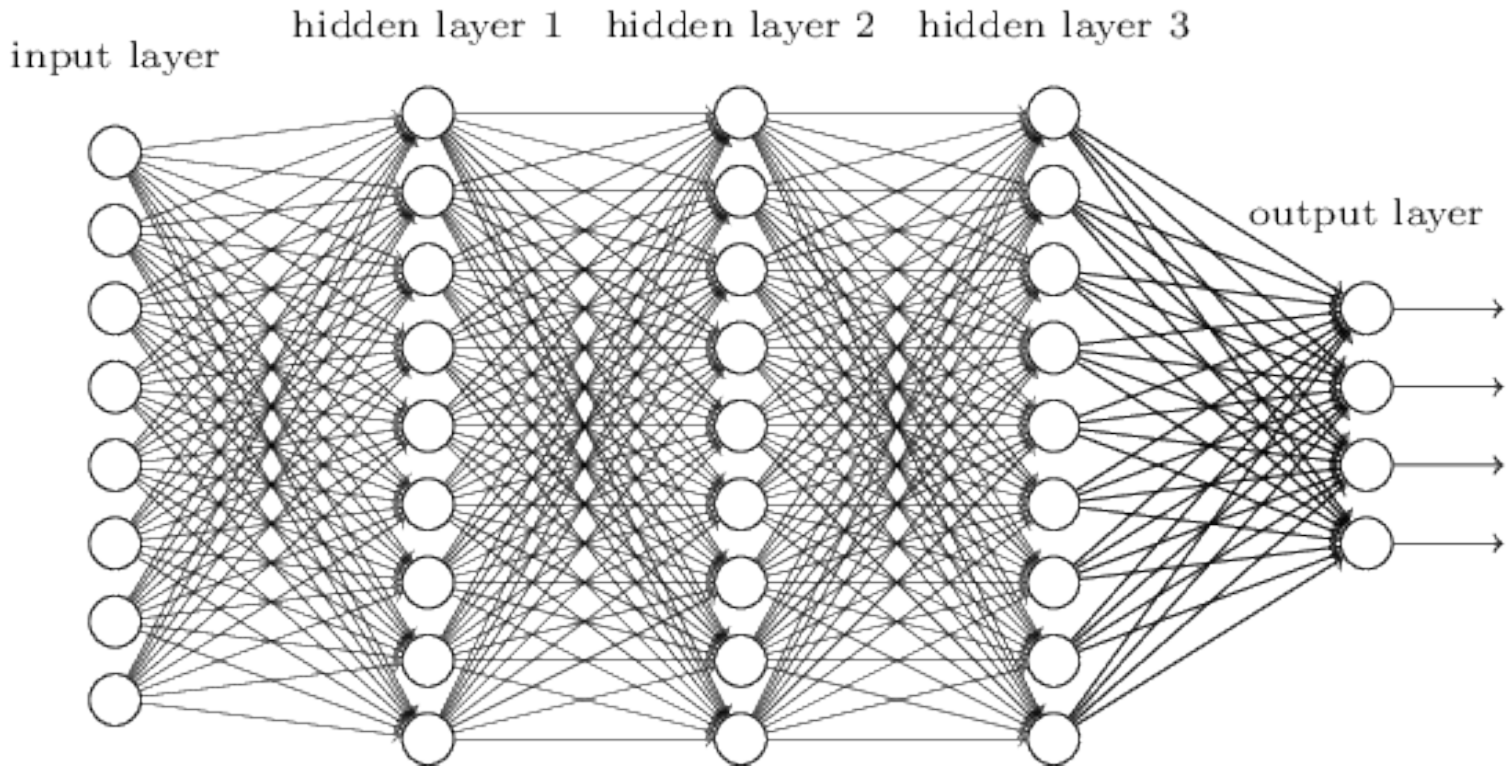
为什么深度网络难以训练



$$\delta^l = \sum'(z^l)(w^{l+1})^T \sum'(z^{l+1})(w^{l+2})^T \ldots \sum'(z^L)\nabla_a C$$

Part VI
深度学习

深度神经网络 - 全连接



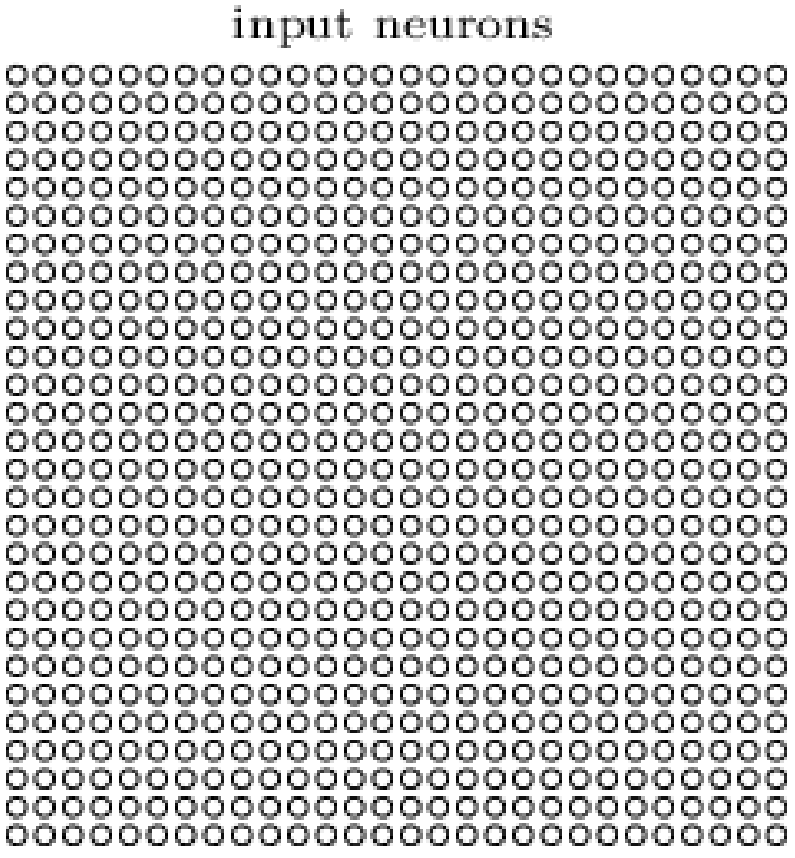
对于识别手写数字问题来说，全连接的深度神经网络忽略了图像的空间结构

卷积神经网络的三个基本概念

1. 局部感知野
2. 共享权重
3. 池化

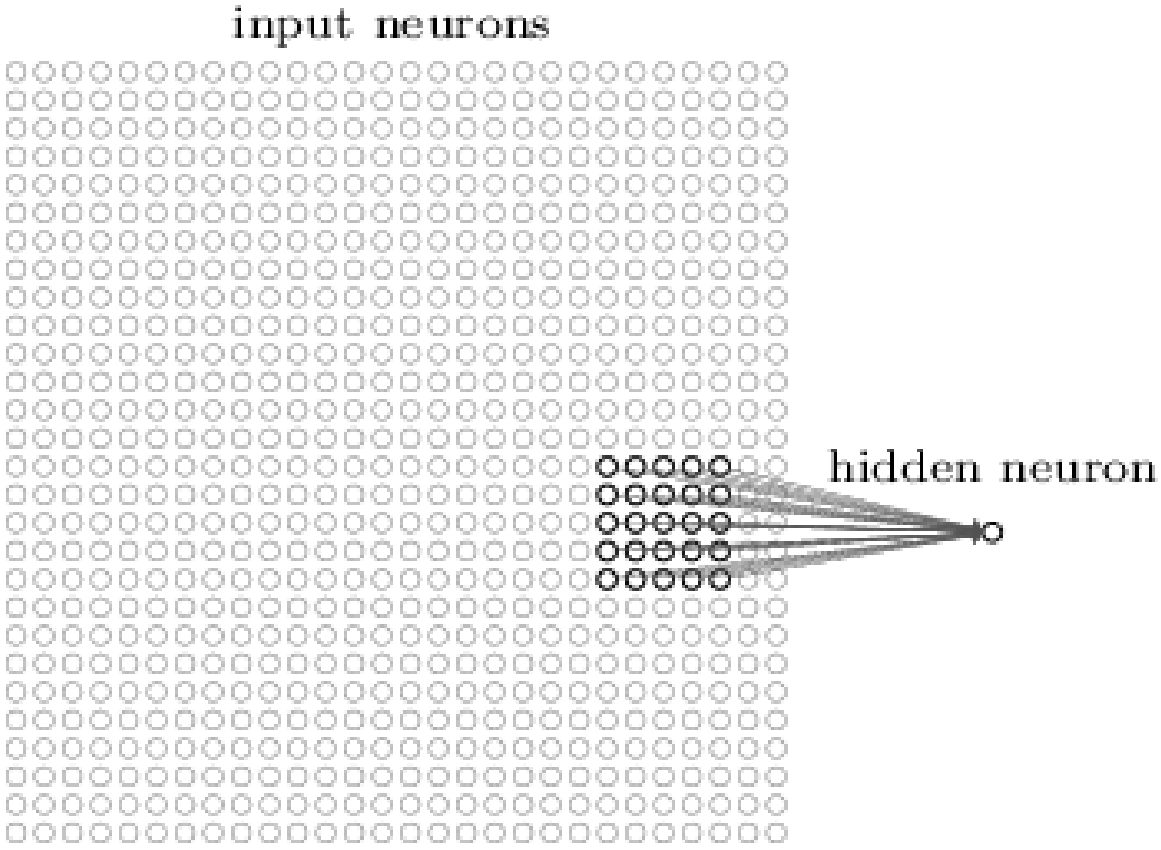
卷积神经网络

我们将输入排列成矩阵的形式

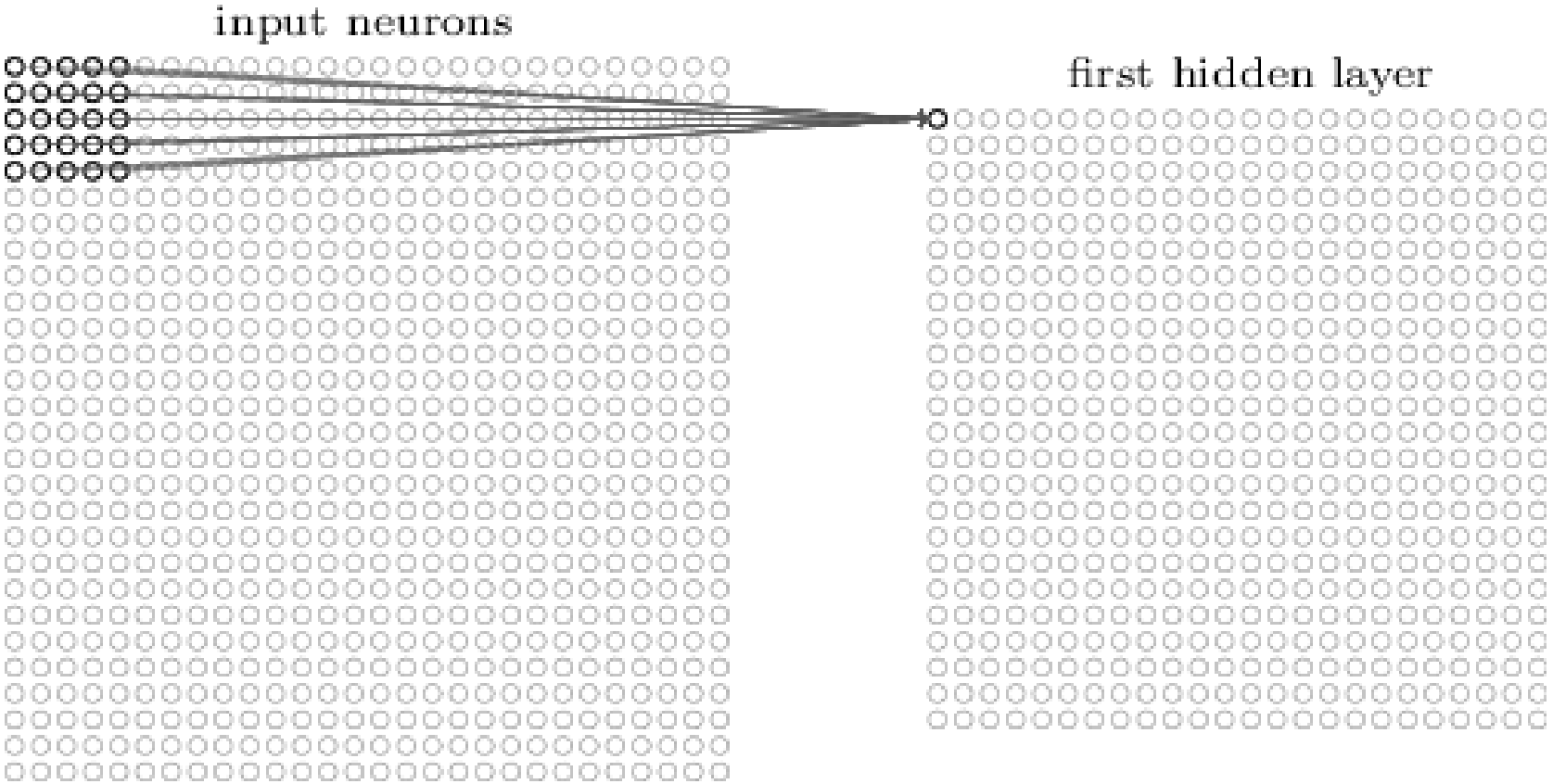


局部感知野

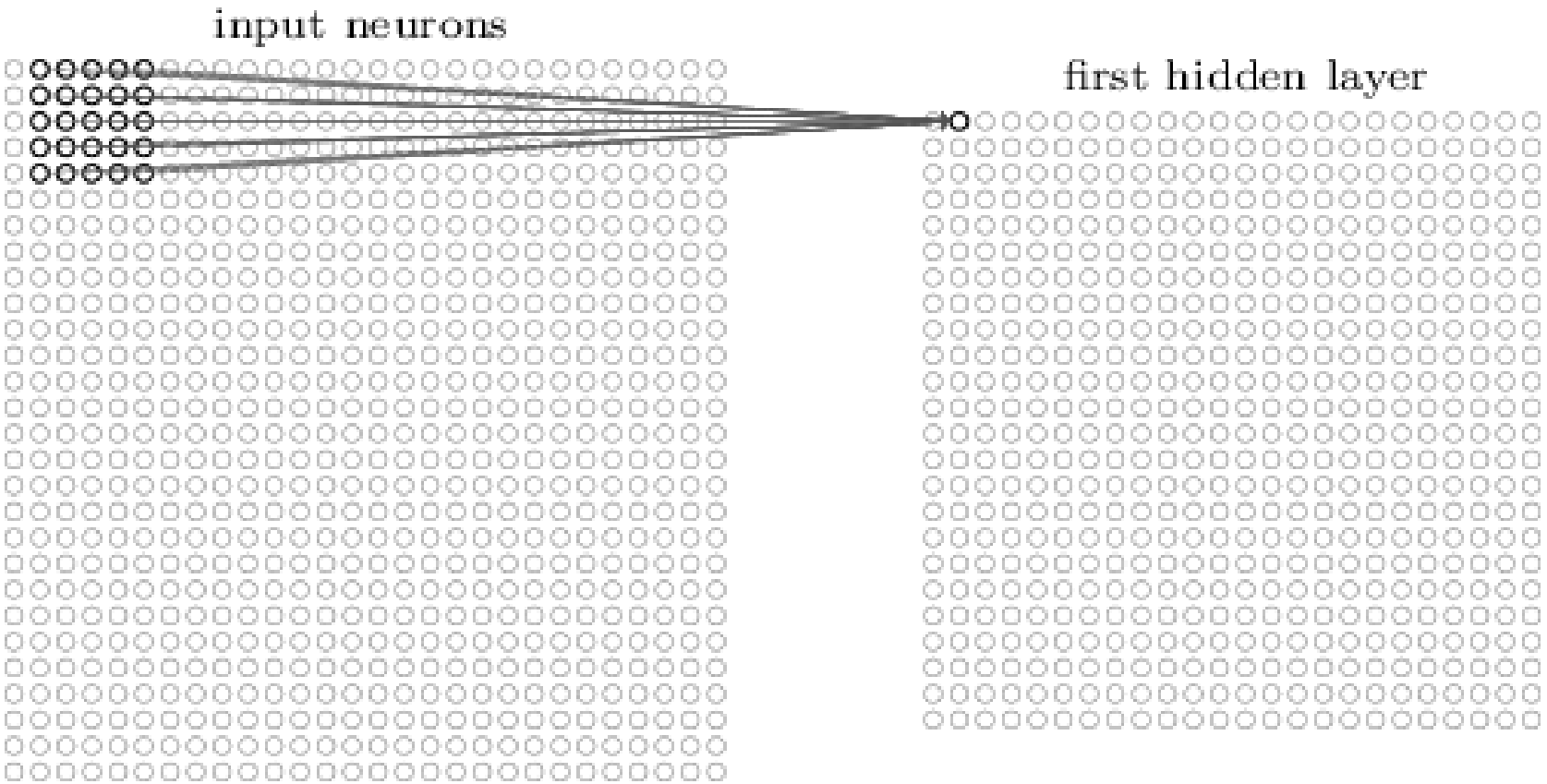
输入层和隐含层不再是全连接的，与隐含层相连接的输入神经元叫做它的局部感知野



在整个输入图像上滑动感知野



每个感知野都有一个不同的隐含层神经元与其对应



共享权重和偏差

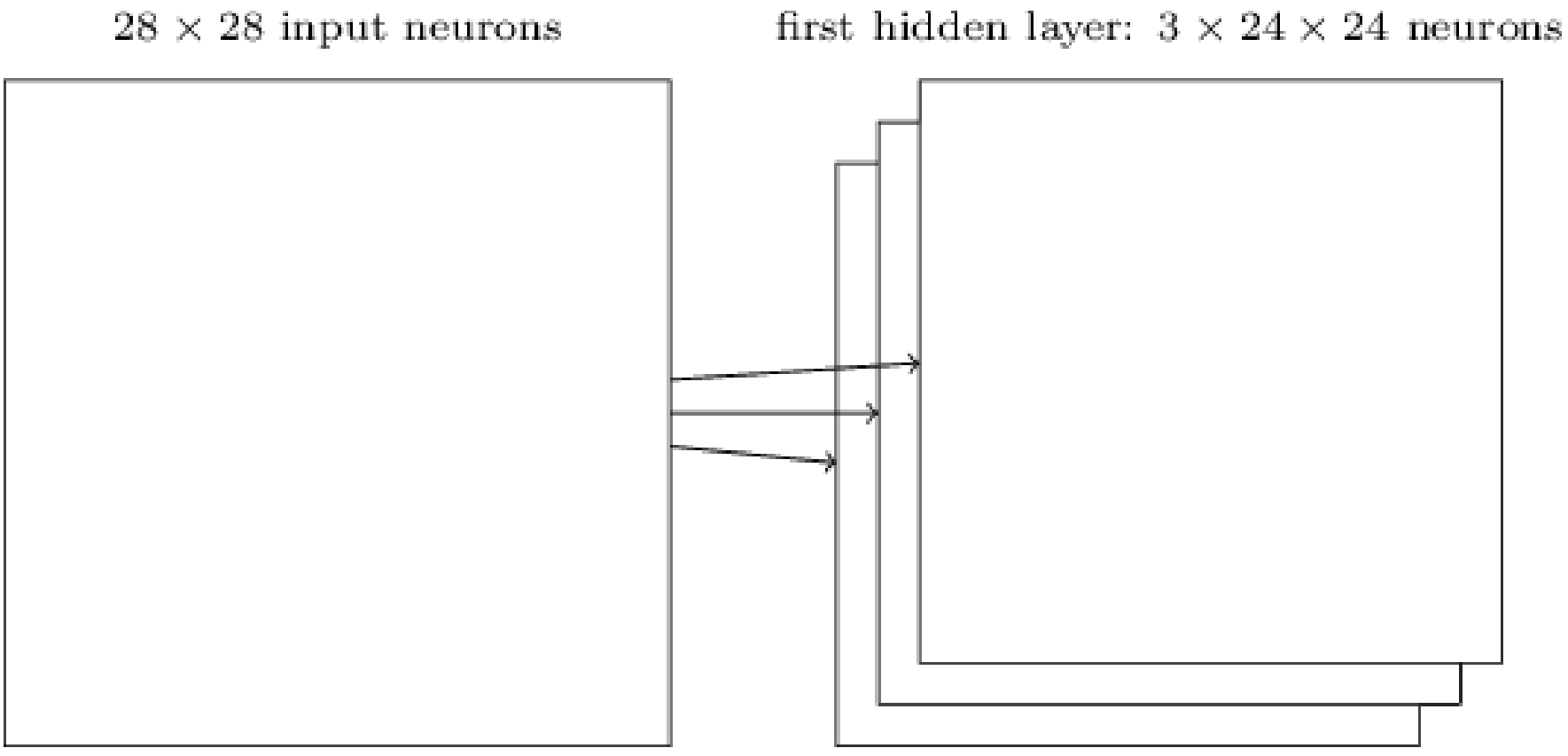
对于同一隐含层的神经元，权重和偏差由整个层来共享

这使得卷积层的参数数量大大减少

$$a^1 = \sigma(b + \sum_{l=0}^{4}\sum_{m=0}^{4} w_{l,m} a^0_{j+l,k+m}) = \sigma(b + w * a^0)$$

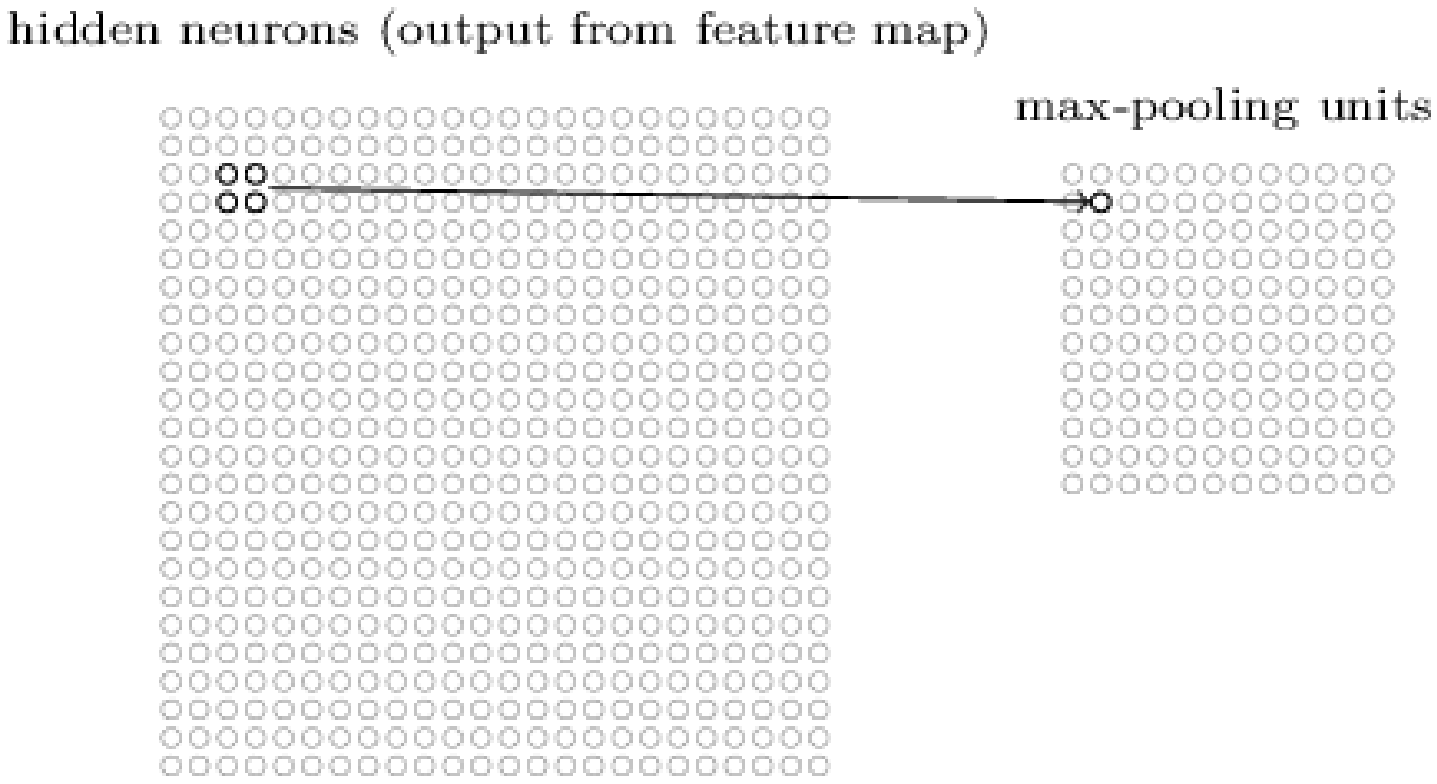
$*$ 为卷积运算符

卷积层

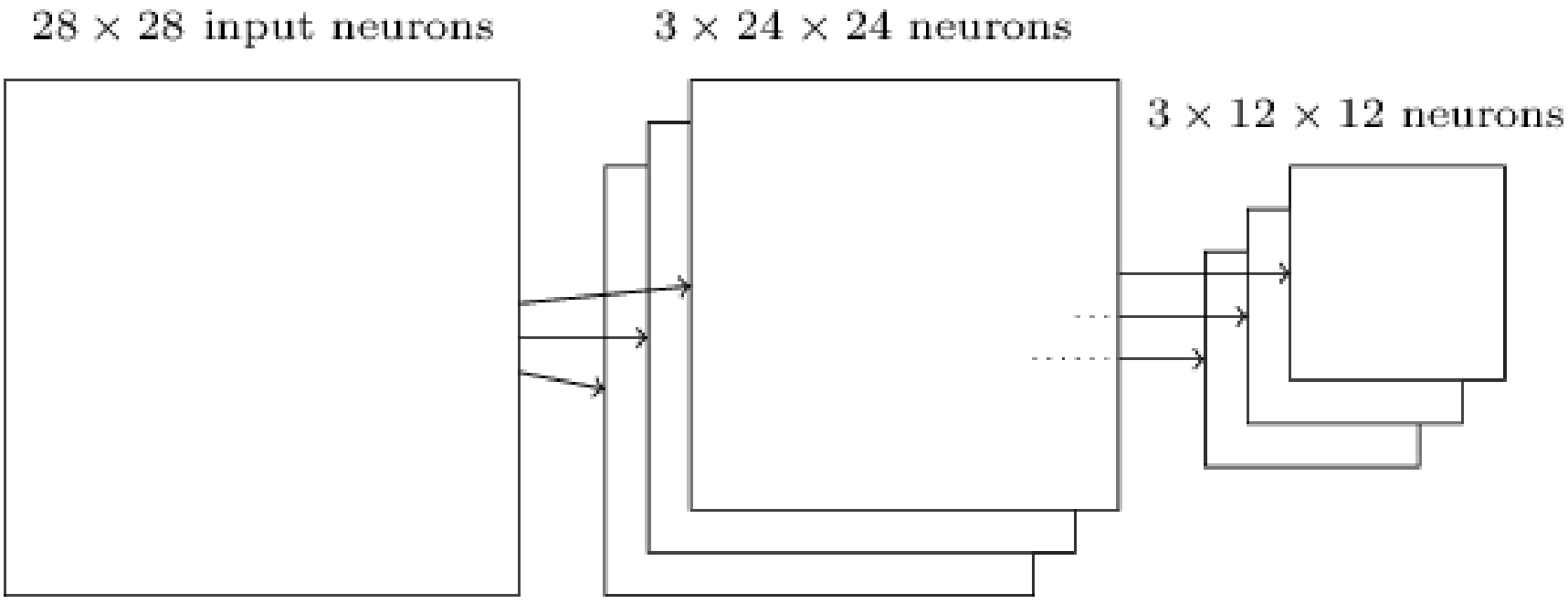


池化

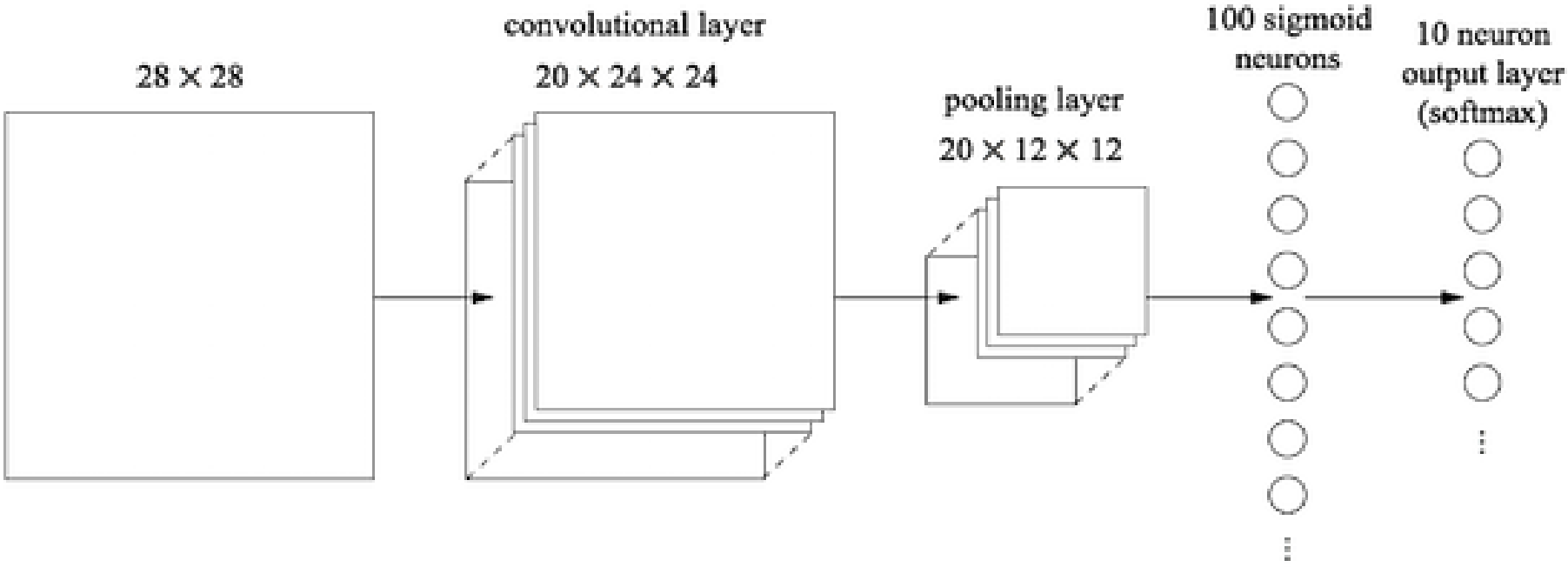
池化层通常直接用在卷积层之后，池化层的工作就是简化卷积层的输出信息。



卷积层 + 池化层



卷积神经网络


28 × 28
convolutional layer
20 × 24 × 24
pooling layer
20 × 12 × 12
100 sigmoid neurons
10 neuron output layer (softmax)

Reference

- Michael A. Nielsen, Determination Press, 2015 *Neural Networks and Deep Learning*
- `https://www.bilibili.com/video/av15532370` 【官方双语】深度学习之神经网络的结构 *Part 1 ver 2.0*