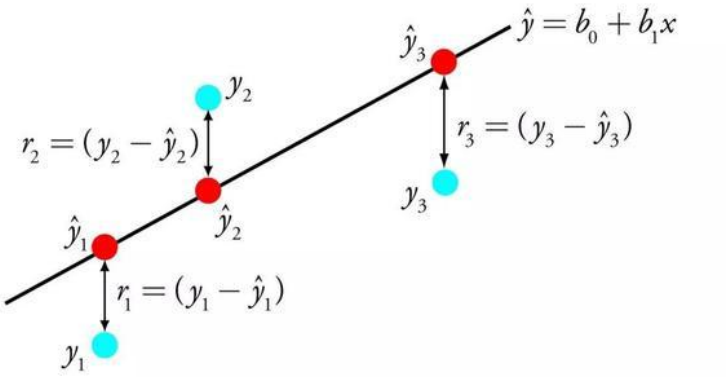
NN

- Neural Network

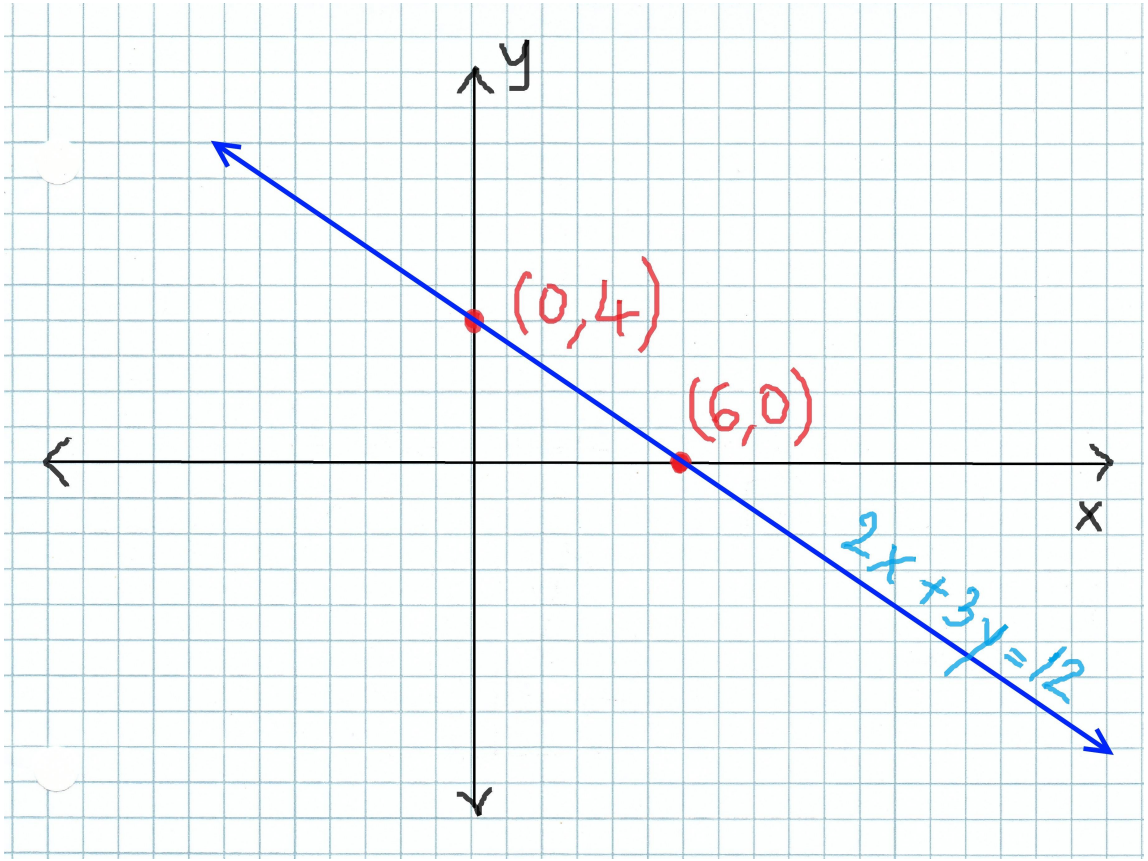
Outline

- Linear Regression
- Logistic/Softmax Regression
- (Feed Forward) NN

Linear Regression



线性方法



房价预测

- 估计竞拍价



代理商的
市场价格

$5,498,000	7	5	4,865 Sq. Ft.
Price	Beds	Baths	$1130 / Sq. Ft.

Redfin Estimate: $5,390,037　On Redfin: 15 days

估计
销售价格

Virtual Tour

- Branded Virtual Tour
- Virtual Tour (External Link)

Parking Information

- Garage (Minimum): 2
- Garage (Maximum): 2
- Parking Description: Attached Garage, On Street
- Garage Spaces: 2

Multi-Unit Information

- # of Stories: 2

School Information

- Elementary School: El Carmelo El
- Elementary School District: Palo A
- Middle School: Jane Lathrop Stan
- High School: Palo Alto High
- High School District: Palo Alto Un

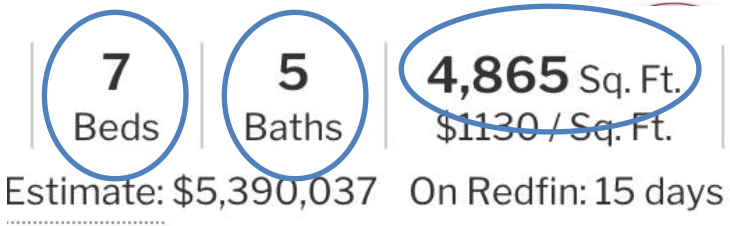
Interior Features

Bedroom Information

- # of Bedrooms (Minimum): 7
- # of Bedrooms (Maximum): 7

- Kitchen Description: Countertop
  Dishwasher, Garbage Disposal, H
  Island with Sink, Microwave, Over

一个简易模型



- 假设 1
  影响房价的关键因素：
  卧室数目，卫浴数目和房子大小，分别用 x_1 x_2 x_3 表示
- 假设 2
  销售价格是关键因素的加权总和：

$$y = w_1x_1 + w_2x_2 + w_3x_3$$

权重和偏差稍后确定。

线性方法

- 给予 n 维输入，$\mathbf{x} = [x_1, x_2, \ldots, x_n]$
- 线性方法有 n 个权重和偏差：

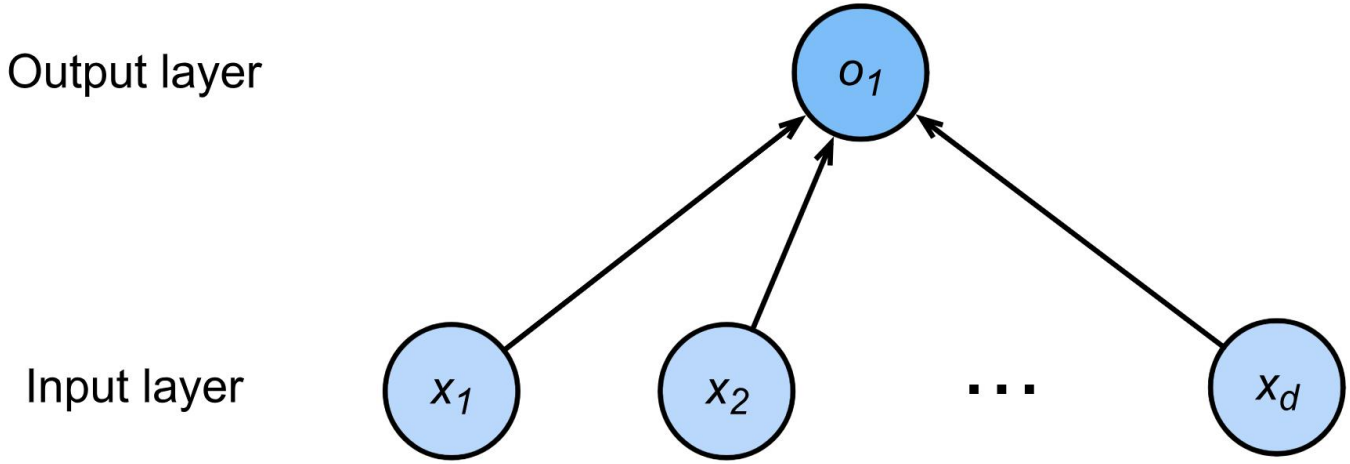
$$\mathbf{w} = [w_1, w_2, \ldots, w_n]^T, b$$

- 输出是输入的加权总和：

$$v = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

矢量化版本：$v = \langle \mathbf{w}, \mathbf{x} \rangle + b$

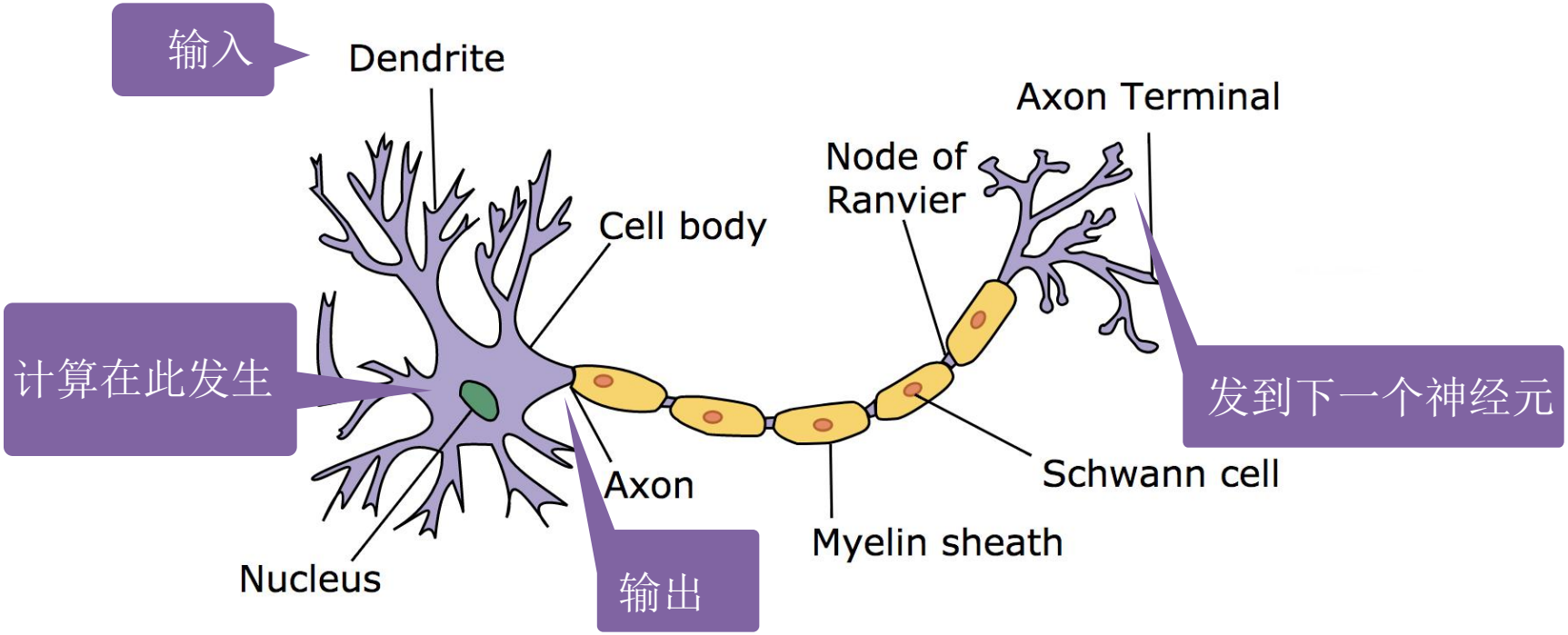
线性方法是一个单层神经网络



我们可以堆叠多个层来获得深层神经网络。

神经科学的灵感

真实的神经元



测量估计质量

- 比较真实值与估计值（实际销售价格与估计的房价）
- 以 y 作为真实值，$\hat{y}$ 作为估计值，我们可以比较损失

平方损失：

$$\ell(y, \hat{y}) = (y - \hat{y})$$

训练数据集

- 收集多个数据点以训练参数（如 在过去6个月内出售的房屋）

$$\mathbf{X} = [\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n]^T \mathbf{y} = [y_0, y_1, \ldots, y_n]$$

- 这个叫做训练数据集
- 训练数据集 越大越好
- 假设有 n 个房屋

学习参数

- 训练损失

$$\ell(\mathbf{X}, \mathbf{y}, \mathbf{w}, b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \langle \mathbf{x}_i, \mathbf{w}\rangle - b)^2 = \frac{1}{n} \| \mathbf{y} - \mathbf{X}\mathbf{w} - b \|^2$$

- 最小化学习参数的损失

$$\mathbf{w}^*, \mathbf{b}^* = \underset{\mathbf{w},b}{\operatorname{argmin}} \ell(\mathbf{X}, \mathbf{y}, \mathbf{w}, b)$$

封闭解

- 将偏差添加到权重中 $\mathbf{X} \leftarrow [\mathbf{X}, \mathbf{1}]\mathbf{w} \leftarrow \begin{bmatrix}\mathbf{w} \\ b\end{bmatrix}$

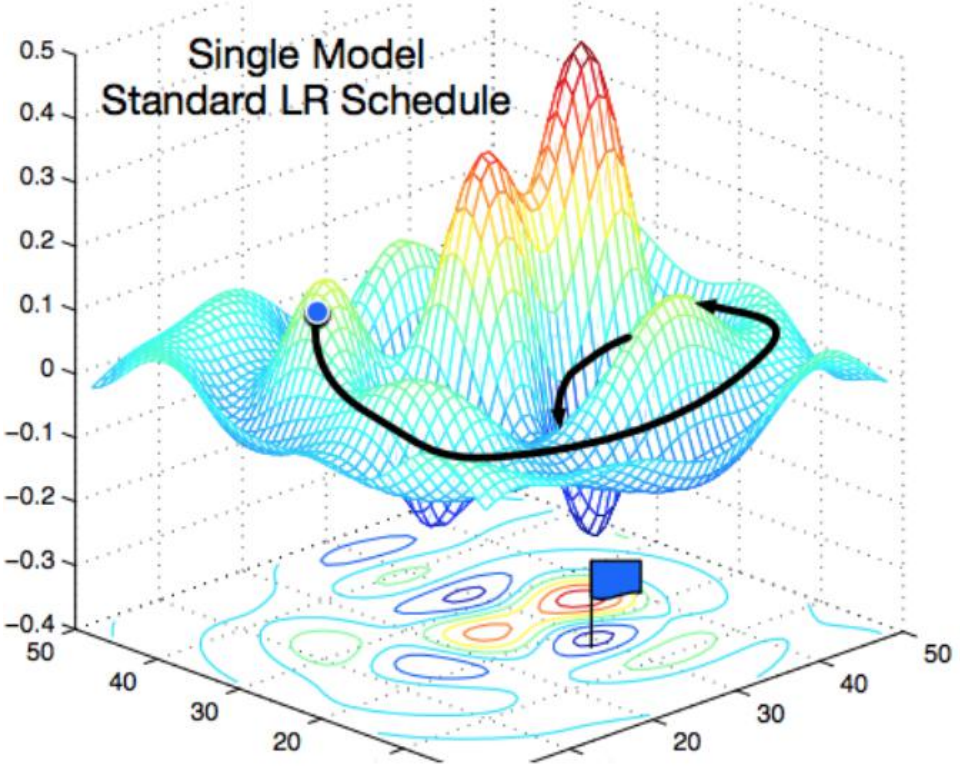
$$\ell(\mathbf{X}, \mathbf{y}, \mathbf{w}) = \frac{1}{n} \| \mathbf{y} - \mathbf{X}\mathbf{w} \|^2 \quad \frac{\partial}{\partial \mathbf{w}} \ell(\mathbf{X}, \mathbf{y}, \mathbf{w}) = \frac{2}{n}(\mathbf{y} - \mathbf{X}\mathbf{w})^T\mathbf{X}$$

- 损失是凸性的，因此最优解满足：

$$\frac{\partial}{\partial \mathbf{w}} \ell(\mathbf{X}, \mathbf{y}, \mathbf{w}) = 0 \Leftrightarrow \frac{2}{n}(\mathbf{y} - \mathbf{X}\mathbf{w})^T\mathbf{X} = 0$$

$$\Leftrightarrow \mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}\mathbf{y}$$

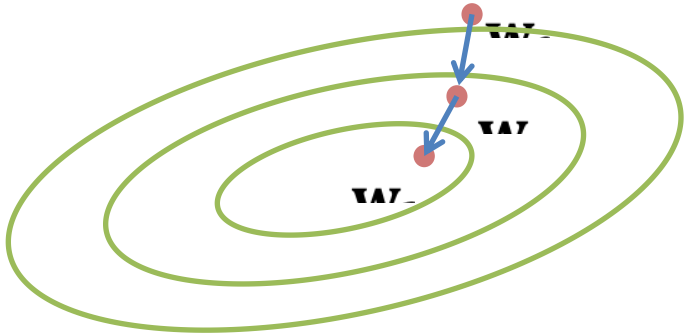
基础优化


Single Model
Standard LR Schedule
0.5
0.4
0.3
0.2
0.1
0
−0.1
−0.2
−0.3
−0.4
50
40
30
20
50
40
30
20

梯度下降

- 选择一个起点 $\mathbf{w}_0$
- 重复更新权重 t = 1,2,3

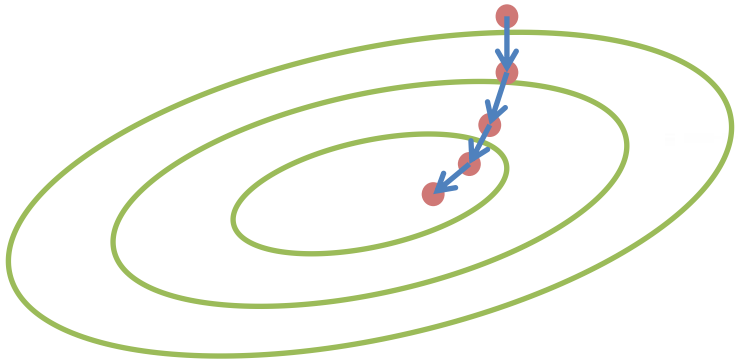
$$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta \frac{\partial t}{\partial \mathbf{w}_t}$$



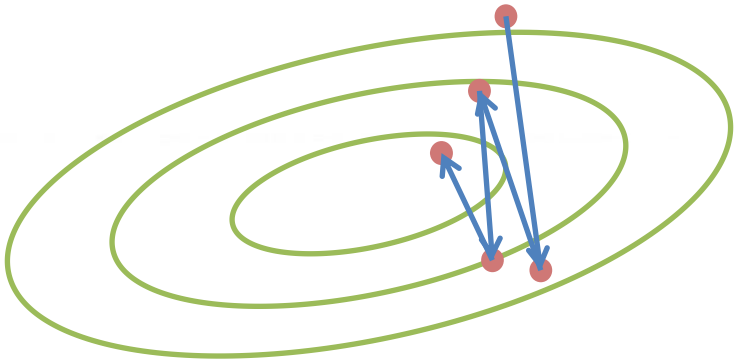
  - 梯度： 更新权重的方向
  - 学习率： 一个超参数指定 每梯度的步长

选择学习率

不要太小



不要太大



小批量随机梯度下降（SGD）

- 计算整个训练数据的梯度太昂贵了
  - DNN模型需要几分钟到几小时

- 解决方案： 随机抽样b个样本$i_1, i_2, \ldots, i$来估算损失

$$\frac{1}{b} \sum_{i \in I_b} \ell(\mathbf{x}_i, y_i, \mathbf{w}$$

  - b 是批量大小，另一个重要的超参数

选择批量值

不要太小

批量值太小，难以充分利用计算资源

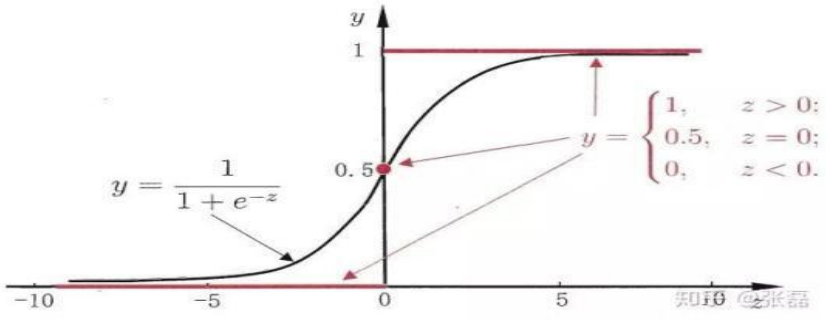
不要太大

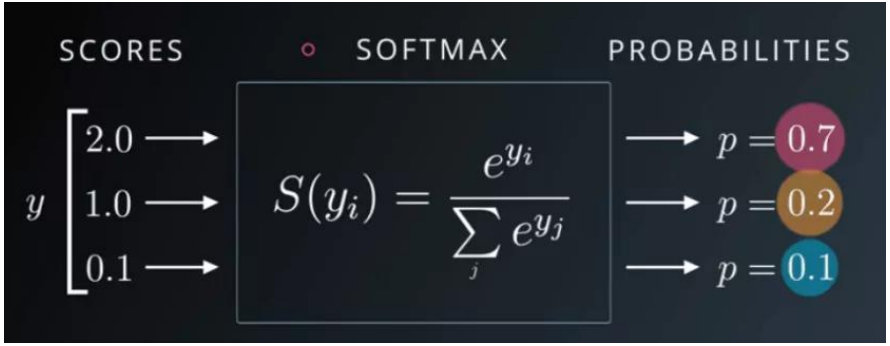
批量值太大，浪费计算资源；
例如 当　都相同时

Linear Regression

- 问题： 估计一个真正的值
- 模型： $y = \langle \mathbf{w}, \mathbf{x} \rangle + b$
- 损失： 平方损失 $\rho(y, \hat{y}) = (y - \hat{y}$
- 小批量随机梯度（mini-batch SGD）学习
  - 选择一个起点
  - 重复
    - 计算梯度
    - 更新参数

Logistic /softmax Regression

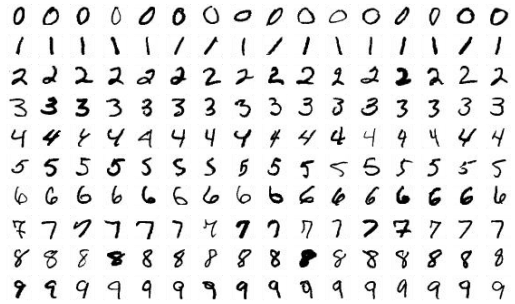




回归与分类

- 回归估计连续值
- 分类预测离散类别

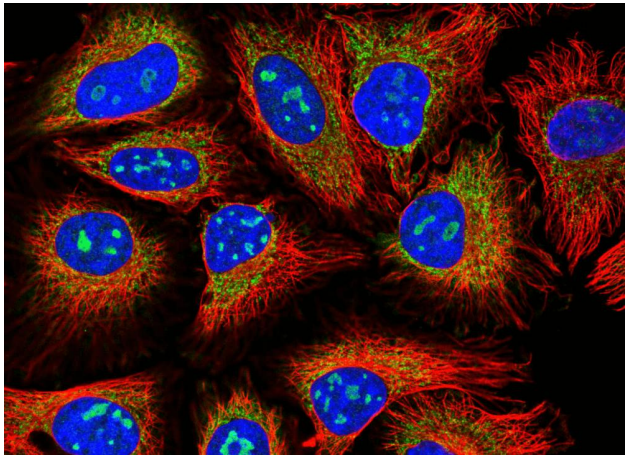
MNIST：对手写数字进行分类（10 类）



ImageNet：对自然对象进行分类（1000 类）

Kaggle上的各种分类任务

- 将人类蛋白质显微镜图像分为28类



```
0.  Nucleoplasm
1.  Nuclear membrane
2.  Nucleoli
3.  Nucleoli fibrillar
4.  Nuclear speckles
5.  Nuclear bodies
6.  Endoplasmic reticu
7.  Golgi apparatus
8.  Peroxisomes
9.  Endosomes
10.  Lysosomes
11.  Intermediate fila
12.  Actin filaments
13.  Focal adhesion si
14.  Microtubules
15.  Microtubule ends
16.  Cytokinetic bridg
```

https://www.kaggle.com/c/human-protein-atlas-image-classification

Kaggle上的各种分类任务

- 将恶意软件分为9类

https://www.kaggle.com/c/malware-classification

Kaggle上的各种分类任务

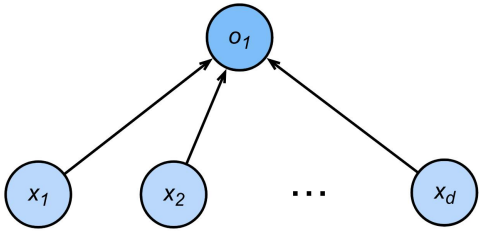
- 将维基百科上的恶意评论分为7类

comment_text	toxic	severe_toxic	obsc
Explanation\nWhy the edits made under my usern...	0	0	0
D'aww! He matches this background colour I'm s...	0	0	0
Hey man, I'm really not trying to edit war. It...	0	0	0
"\nMore\nI can't make any real suggestions on ...	0	0	0
You, sir, are my hero. Any chance you remember...	0	0	0

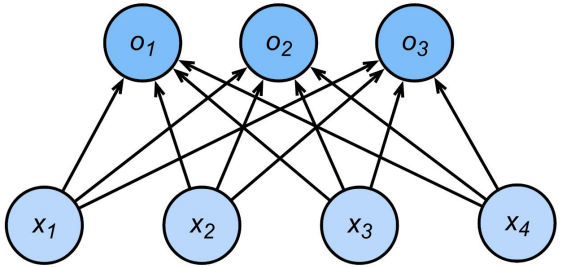
https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

从回归到多类分类

- 单个连续数值输出



- 输出每个类别的置信度分数



- 预测类别

$$\mathrm{arg}\max_i(o_1, o_2, o_3)$$

- *max* 不可求导
- 定义一个损失函数

softmax 函数

$$softmax([x_1, x_2, \ldots, x_n]^T) = \left[\frac{e^{x_1}}{\sum_{i=1}^{n} e^{x_i}}, \frac{e^{x_2}}{\sum_{i=1}^{n} e^{x_i}}, \cdots, \frac{e^{x_n}}{\sum_{i=1}^{n} e^{x_i}}\right]$$

- 用 exp 获得大于0的值
- 除以总和以获得概率分布

例: [1, -1, 2] 的 softmax 为 [0.26, 0.04, 0.7]

是否用平方损失?

- 对 y 进行独热编码（One-hot encoding）

$$\mathbf{y} = [y_1, y_2, \ldots, y_n]^T, y_i = \begin{cases} 1 & if\ i = y \\ 0 & otherwise \end{cases}$$

- 我们需要 softmax 结果接近 y
- 用平方损失：y = [0, 0, 1]

softmax 结果	损失
[0.3, 0, 0.7]	0.18
[0.17, 0.17, 0.66]	0.173

交叉熵

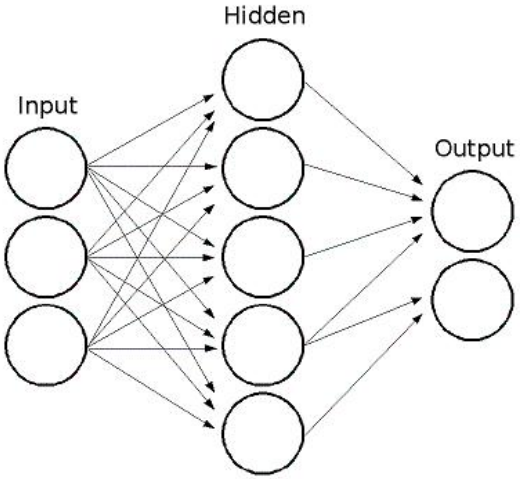
- 独热编码与 softmax 运算结果都为概率分布
- 交叉熵通常用于比较概率分布

$$H(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{n} y_i \log(\hat{y}_i)$$

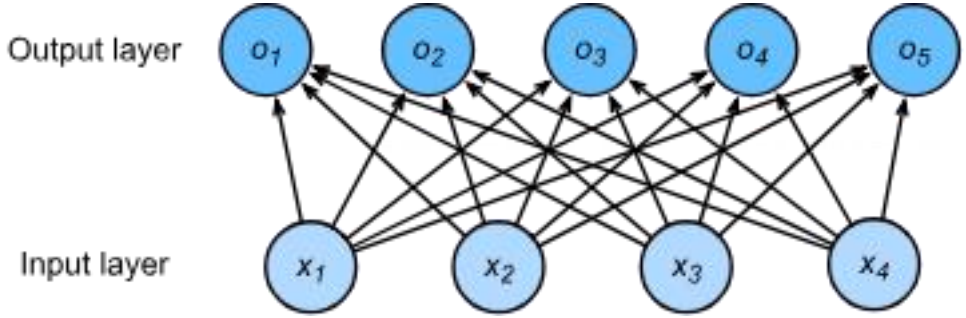
线性回归与softmax回归

	线性回归 （回归问题）	softmax回归 （分类问题）
问题		
模型	$\langle \mathbf{w}, \mathbf{x} \rangle + b$ $\mathbf{w} \in \mathbb{R}^n, b \in \mathbb{R}$	$softmax(\mathbf{W}\mathbf{x} + \mathbf{b})$ $\mathbf{W} \in \mathbb{R}^{k \times n}, \mathbf{b} \in \mathbb{R}^k$
损失	平方损失	交叉熵

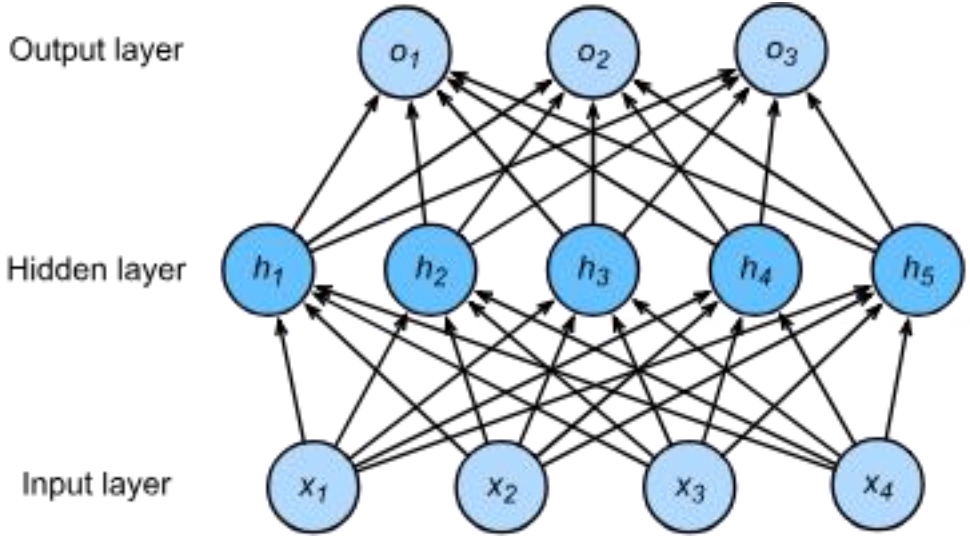
(Feed Forward) NN



无隐藏层



单隐藏层



超参数 - 隐藏层的大小为m

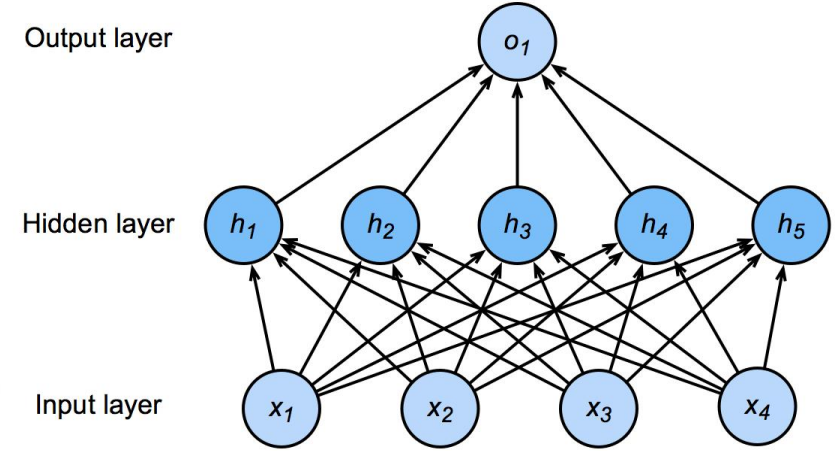
单隐藏层

- 输入：$\mathbf{x} \in \mathbb{R}^n$
- 隐藏层：$\mathbf{W}_1 \in \mathbb{R}^{m \times n}, \mathbf{b}_1 \in \mathbb{R}^m$
- 输出：$\mathbf{w}_2 \in \mathbb{R}^m, b_2 \in \mathbb{R}$

$$\mathbf{h} = \sigma(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1)$$

$$\mathbf{o} = \mathbf{w}_2^T \mathbf{h} + b_2$$

激活函数



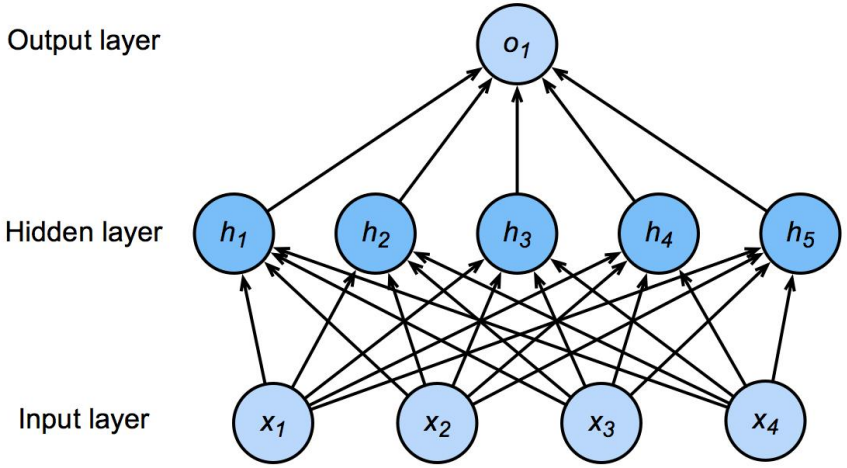
单隐藏层

为什么我们需要非线性激活函数呢？

$$\mathbf{h} = \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$$

$$\mathbf{o} = \mathbf{w}_2^T\mathbf{h} + b_2$$

一个逐元素激活函数



单隐藏层

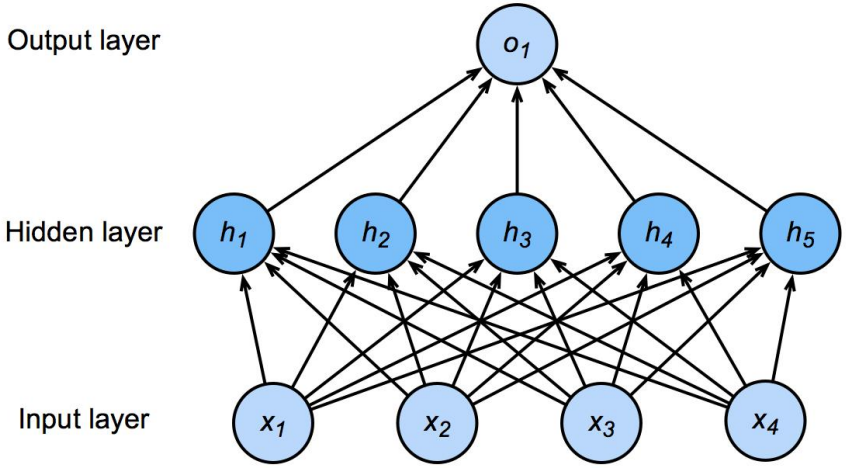
为什么我们需要非线性激活函数呢？

$$\mathbf{h} = \mathbf{W}_1\mathbf{x} + \mathbf{b}_1$$

$$\mathbf{o} = \mathbf{w}_2^T\mathbf{h} + b_2$$

$$\text{hence } o = \mathbf{w}_2^T\mathbf{W}_1\mathbf{x} + b'$$

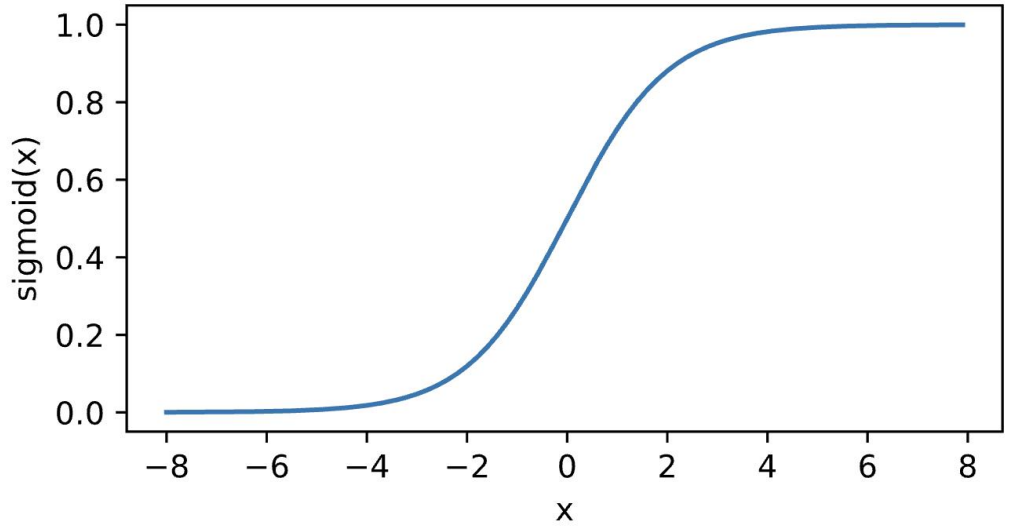
因为它是线性的…



S型（sigmoid）激活函数

将输入映射到 (0, 1)

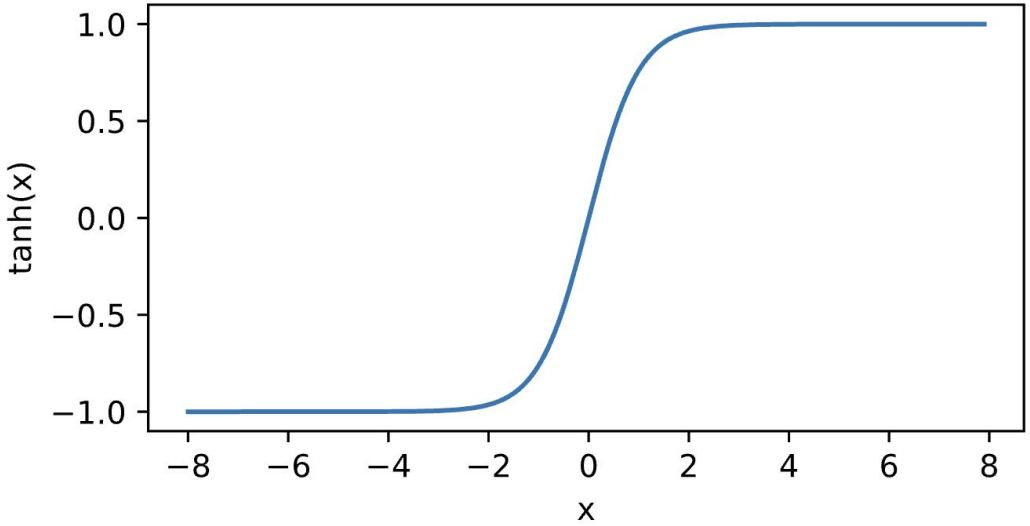
$$\text{sigmoid}(x) = \frac{1}{1 + \exp(-x}$$



双曲正切（tanh）激活函数

将输入映射到 (-1, 1)

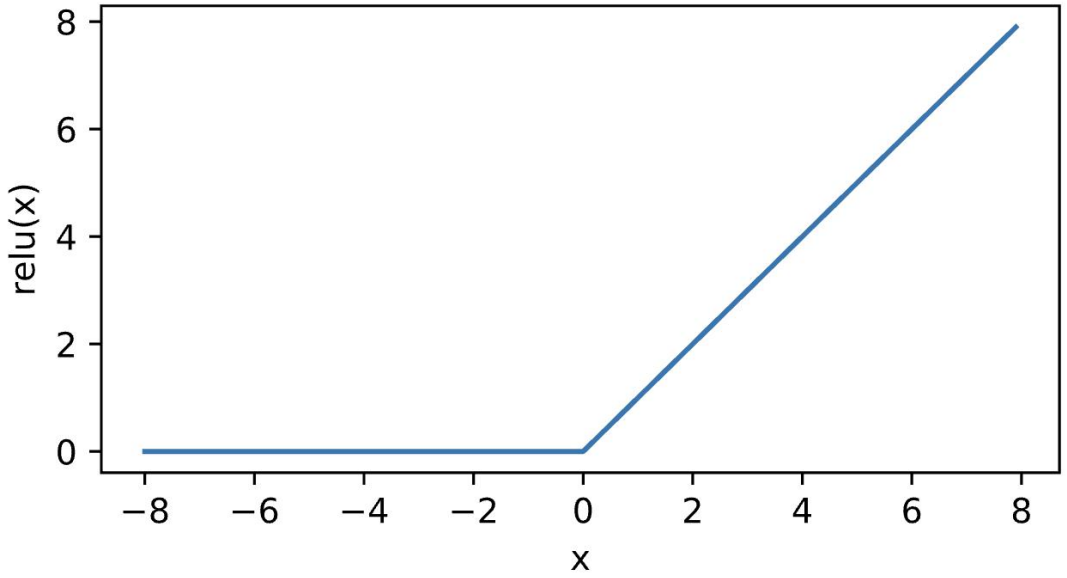
$$\tanh(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)}$$



线性（ReLU）修正函数

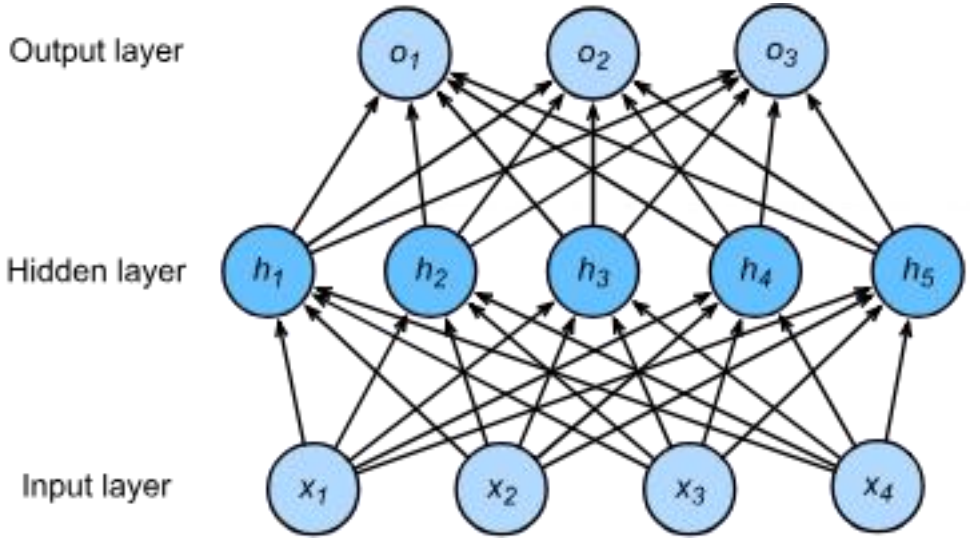
ReLU: 线性修正单元

$$\mathrm{ReLU}(x) = max(x,$$



多类别分类

$$y_1, y_2, \ldots, y_k = \text{softmax}(o_1, o_2, \ldots, o$$



多个隐藏层多类分类

$$\mathbf{h}_1 = \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$$
$$\mathbf{h}_2 = \sigma(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2)$$
$$\mathbf{h}_3 = \sigma(\mathbf{W}_3\mathbf{h}_2 + \mathbf{b}_3)$$
$$\mathbf{o} = \mathbf{W}_4\mathbf{h}_3 + \mathbf{b}_4$$

超参数
- 隐藏层数量
- 每层的隐藏单元数目