6 多层感知机

概要

- 单层感知机
  - 决策边界
  - XOR
- 多层感知机
  - 层数
  - 非线性
  - 计算成本

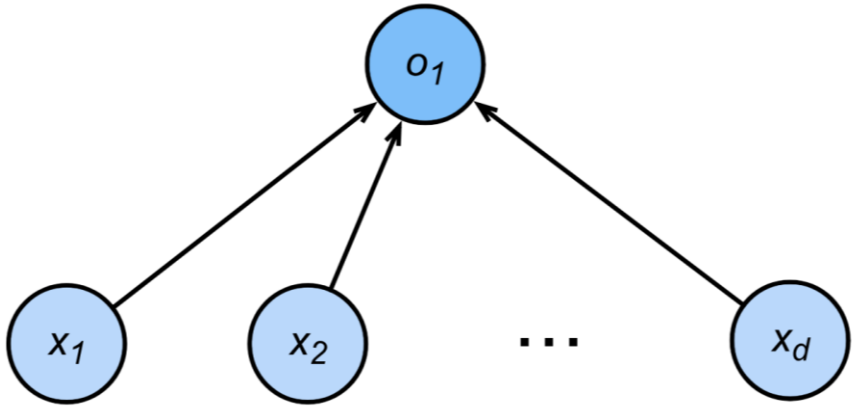
感知机

Mark I Perceptron, 1960

感知机

- 给予输入$\mathbf{x}$， 权重$\mathbf{w}$和偏差b；感知机输出o：

$$o = \sigma(\langle \mathbf{w}, \mathbf{x} \rangle + b), \sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

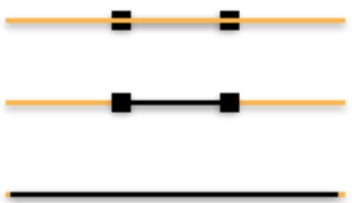


感知机

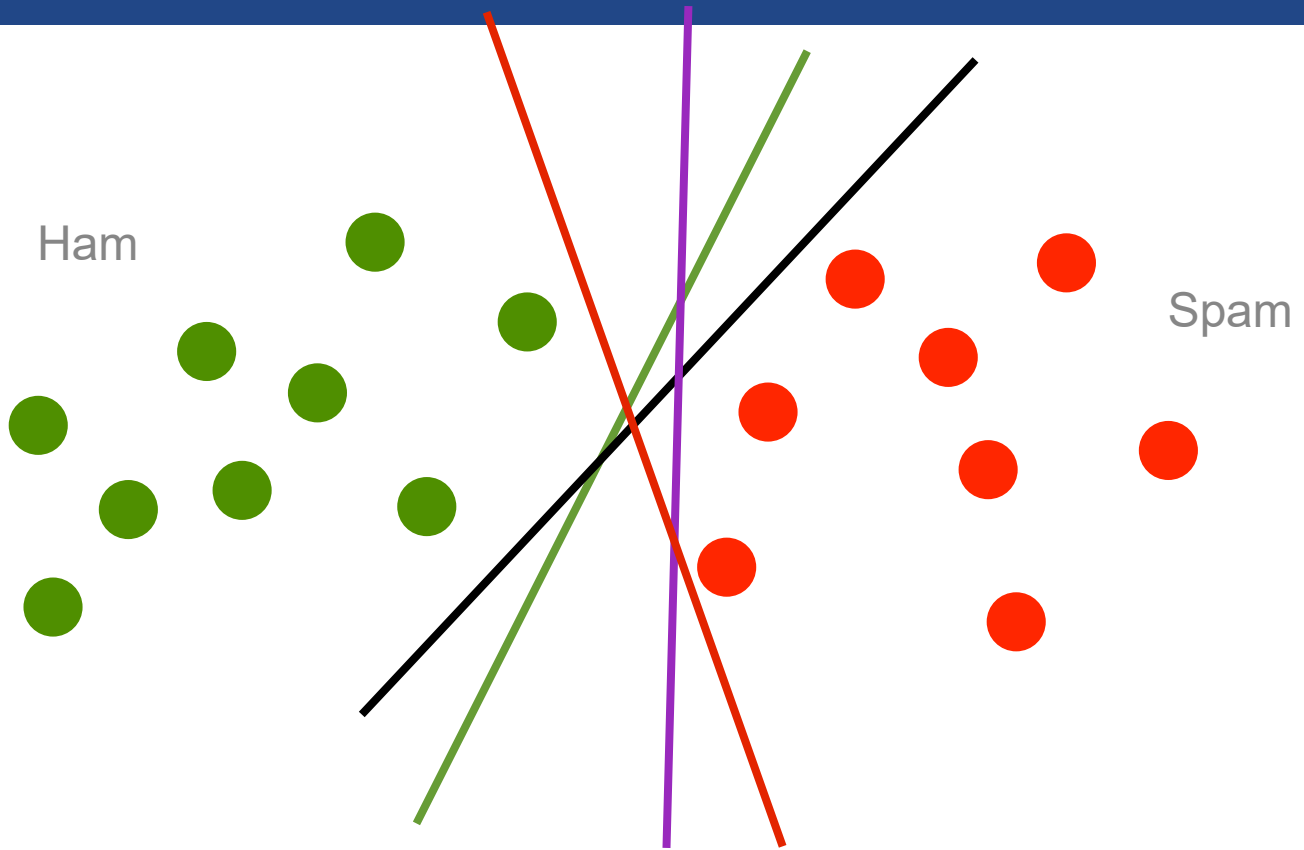
- 给予输入$\mathbf{x}$， 权重$\mathbf{w}$和偏差b；感知机输出o：

$$o = \sigma(\langle \mathbf{w}, \mathbf{x} \rangle + b), \sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

- 二分类 (0 或 1)
  - vs. 回归的标量实际值
  - vs. 逻辑回归的概率



感知机



训练感知机

代码：

initialize $w = 0$ and $b = 0$

repeat

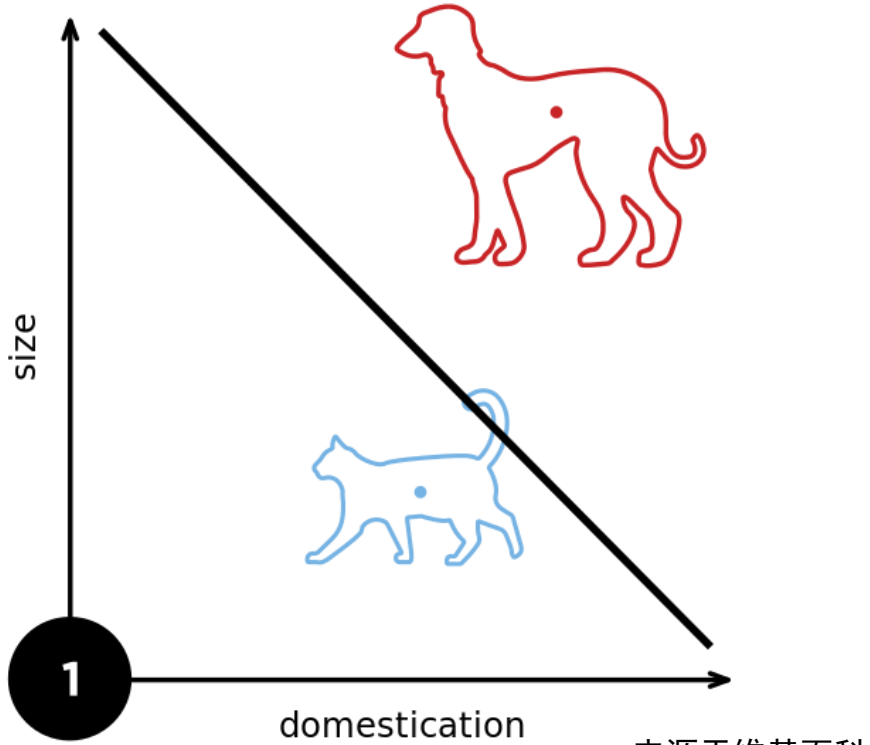
  if $y_i[\langle w, x_i \rangle + b] \leq 0$ then

  $w \leftarrow w + y_i x_i$ and $b \leftarrow b + y_i$ end if

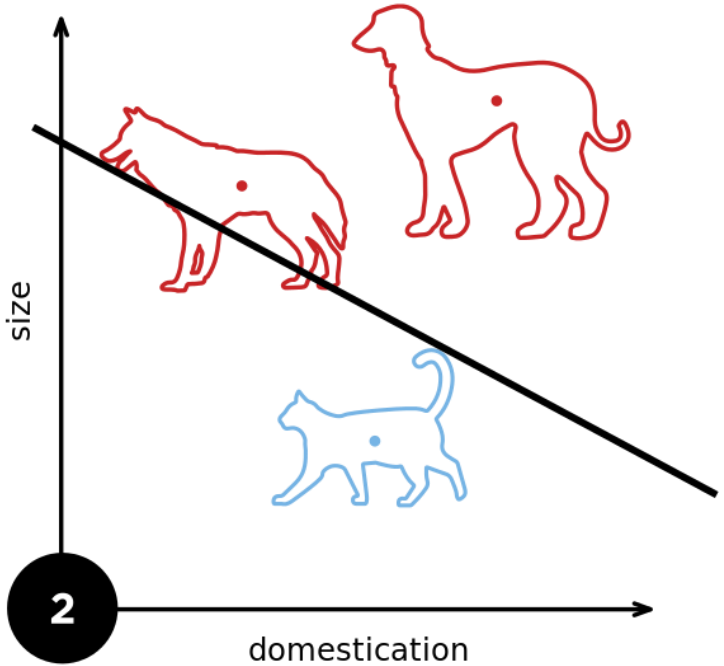
until all classified correctly

相当于批量为 1 的随机梯度下降（SGD）用于一下损失：

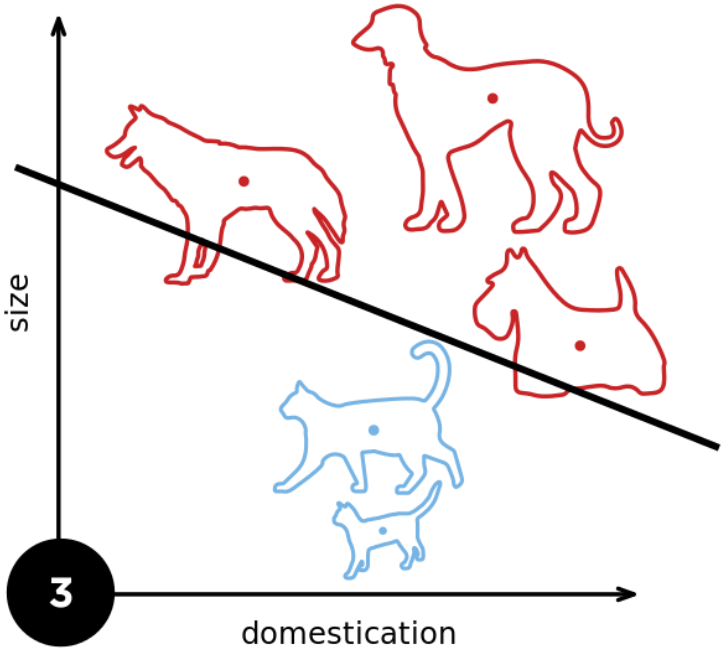
$$\ell(y, \mathbf{x}, \mathbf{w}) = max(0, -y\langle \mathbf{w}, \mathbf{x} \rangle)$$



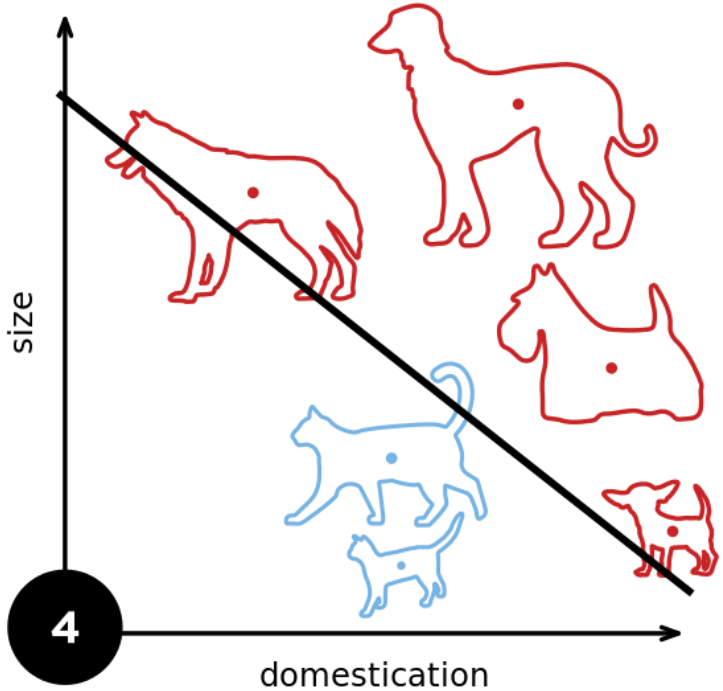
来源于维基百科



来源于维基百科



来源于维基百科



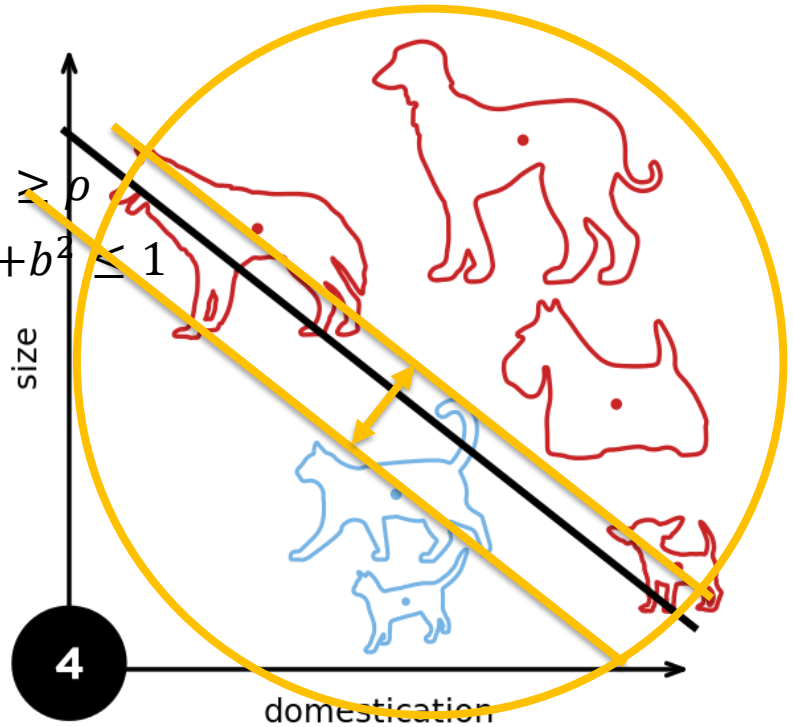
来源于维基百科

收敛定理

- 半径 r 包含数据
- 间隔 ρ 区分类别

$$y(\mathbf{x}^{\top}\mathbf{w}+b) \geq \rho$$
$$for \parallel \mathbf{w} \parallel^2 + b^2 \leq 1$$

- 保证感知机会在$\frac{r^2+1}{\rho^2}$步之后收敛



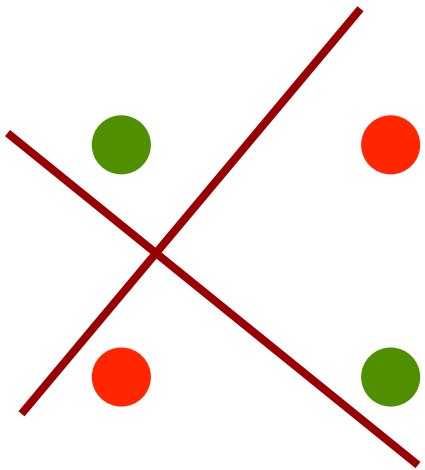
结果

- 只需要存储错误
- 这给出了感知机的压缩界限
- 由于嘈杂的数据而失败

不要用感知机来训练你的头像

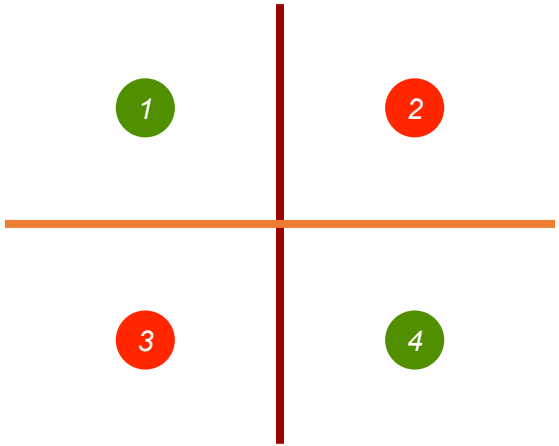
XOR 问题 (Minsky & Papert, 1969)

- 感知机无法学习 XOR 问题（神经元只能生成线性分离器）

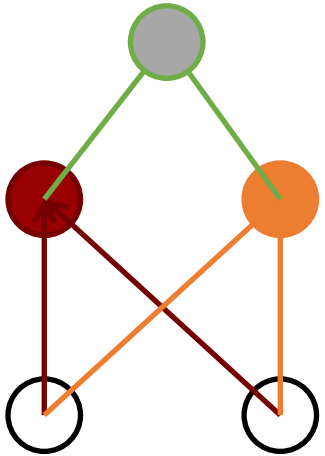


多层感知机

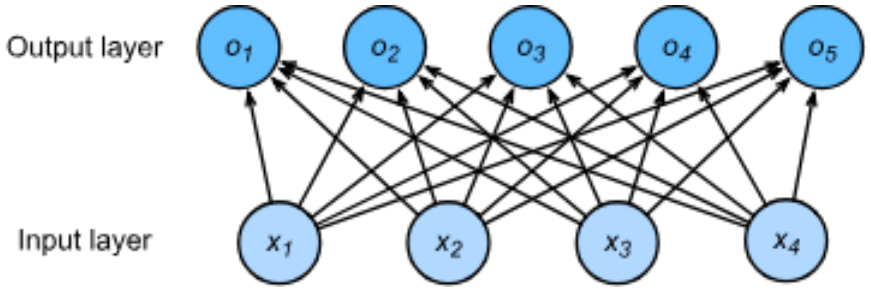
学习 XOR



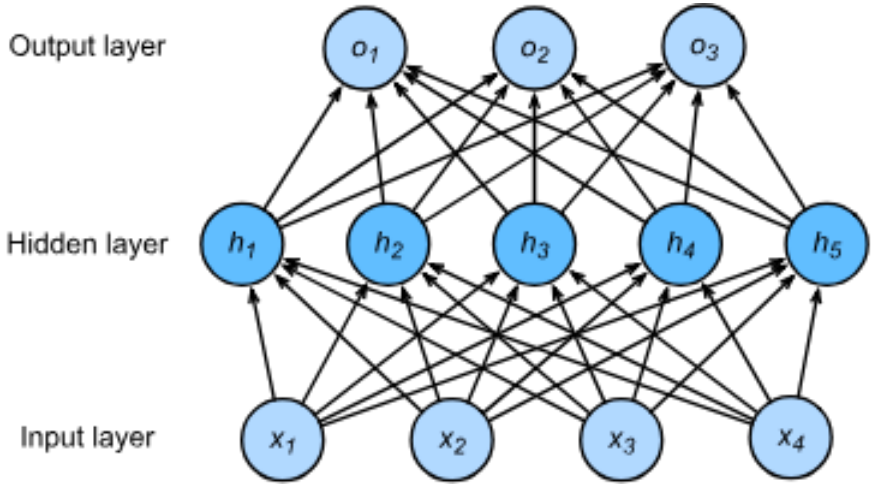
	1	2	3	4
	+	-	+	-
	+	+	-	-
乘积	+	-	-	+



单隐藏层



单隐藏层

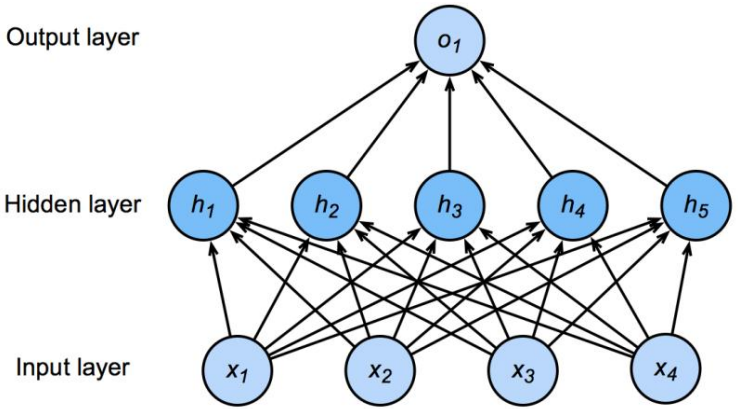


超参数 - 隐藏层的大小为m

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:
  - $\mathbf{w}_2 \in \mathbb{R}^m, b_2 \in \mathbb{R}$
  - $\mathbf{h} = \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$
  - $\mathbf{o} \neq \mathbf{w}_2^T\mathbf{h} + b_2$

激活函数

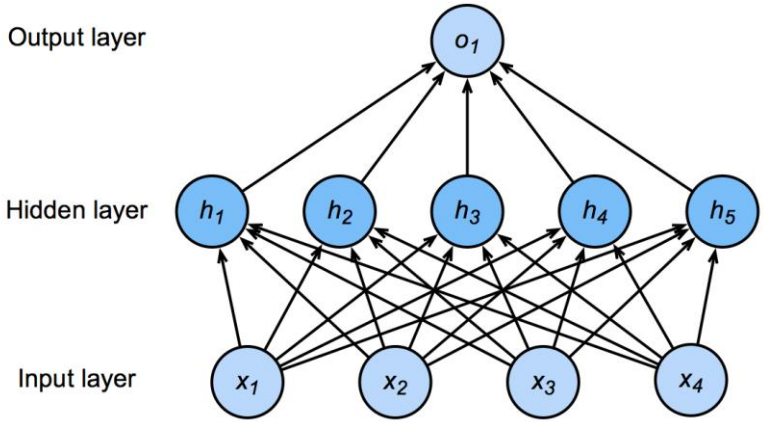


单隐藏层

- $\mathbf{h} = \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$
- $\mathbf{o} \neq \mathbf{w}_2^T\mathbf{h} + b_2$

一个逐元素激活函数

为什么我们需要非线性激活函数呢？



单隐藏层

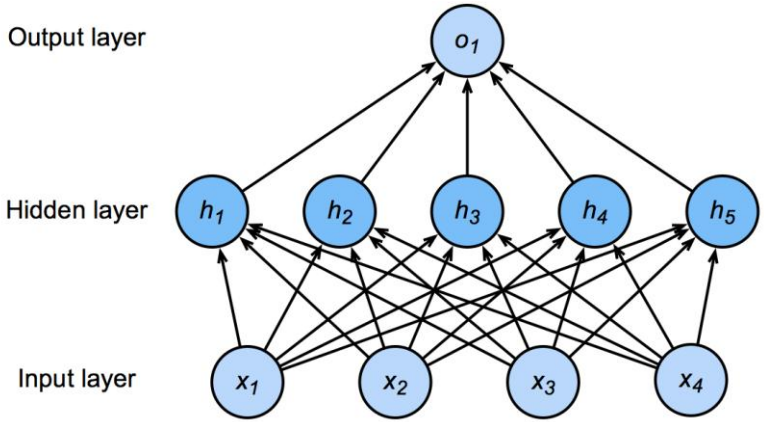
$$\mathbf{h} = \mathbf{W}_1\mathbf{x} + \mathbf{b}_1$$

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

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

因为它是线性的…

为什么我们需要非线性激活函数呢？

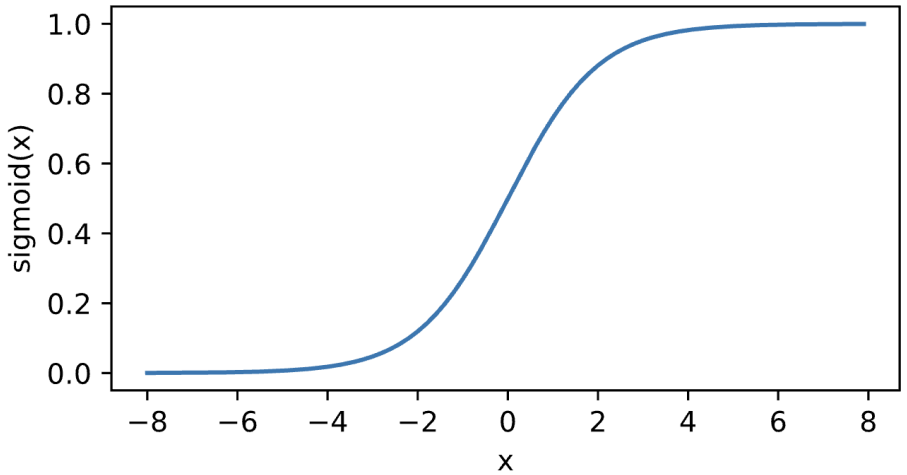


S型（sigmoid）激活函数

将输入映射到 (0, 1)， 一个平滑的版本

$$\sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

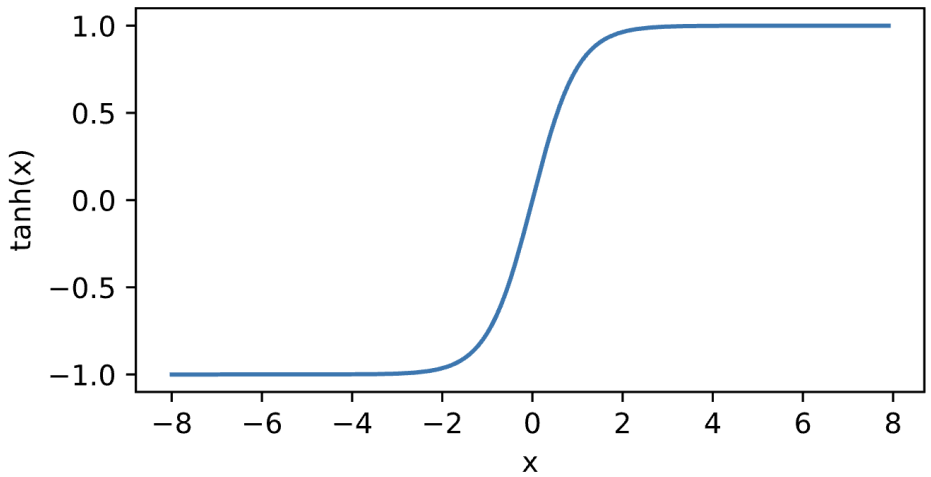
$$\text{sigmoid}(x) = \frac{1}{1 + \exp(-x)}$$



双曲正切（tanh）激活函数

将输入映射到 (-1, 1)

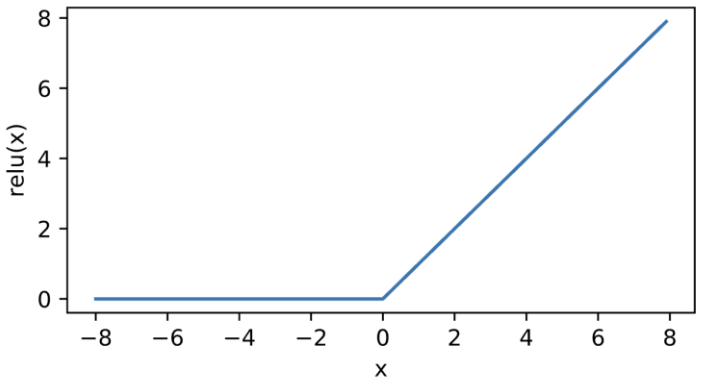
$$\tanh(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)}$$

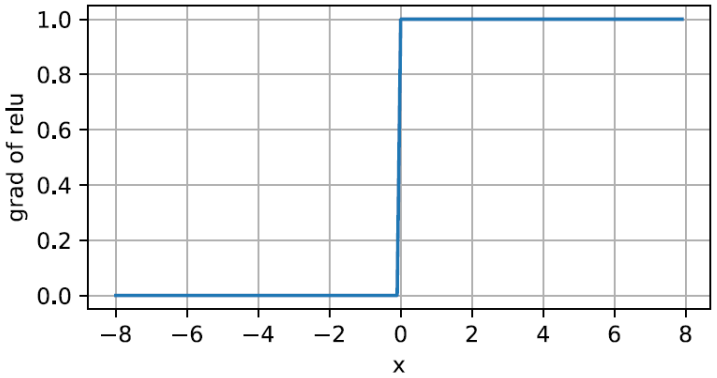


线性（ReLU）修正函数

ReLU: 线性修正单元（Rectified linear unit，ReLU）

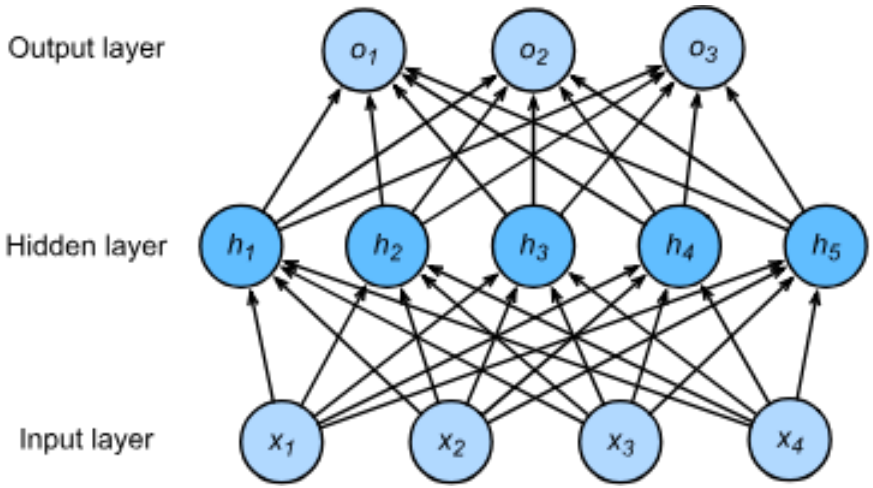
$$\mathrm{ReLU}(x) = max(x, 0)$$





多类别分类

- $y_1, y_2, \ldots, y_k = \text{softmax}(o_1, o_2, \ldots, o_k)$



多类分类

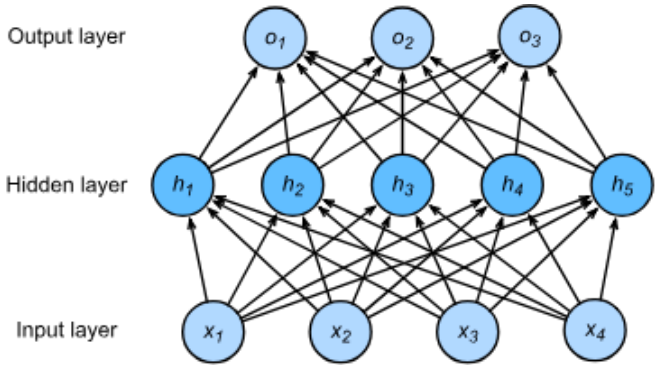
- 输入: $\mathbf{x} \in \mathbb{R}^n$
- 隐藏层: $\mathbf{W}_1 \in \mathbb{R}^{m \times n}, \mathbf{b}_1 \in \mathbb{R}^m$
- 输出:

$$\mathbf{w}_2 \in \mathbb{R}^m, b_2 \in \mathbb{R}$$

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

$$\mathbf{0} \neq \mathbf{w}_2^T \mathbf{h} + b_2$$

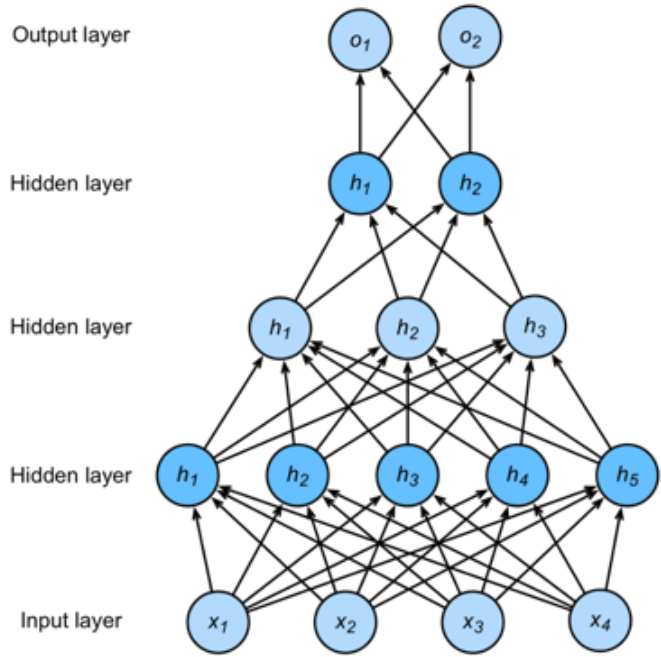
一个按元素激活函数



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



多层感知机的从零开始实现

- 示例

总结

- 单层感知机
  - 决策边界
  - XOR
- 多层感知机
  - 层数
  - 非线性
  - 计算成本

概要

- 模型选择
  - 训练误差和泛化误差
  - K折交叉验证
- 欠拟合与过拟合
  - 模型复杂度
  - VC 维度
- 权重衰减
- 丢弃法

模型选择

训练误差和泛化误差

- 训练误差：出自于训练数据
- 泛化误差：出自于新数据
- 示例：使用历年考试真题准备将来的考试
  - 在历年考试真题取得好成绩（训练误差）并不能保证未来考试成绩好（泛化误差）
  - 学生 A 通过死记硬背学习在历年考试真题取得0错误
  - 学生 B 理解并给出答案的解释

验证数据集和测试数据集

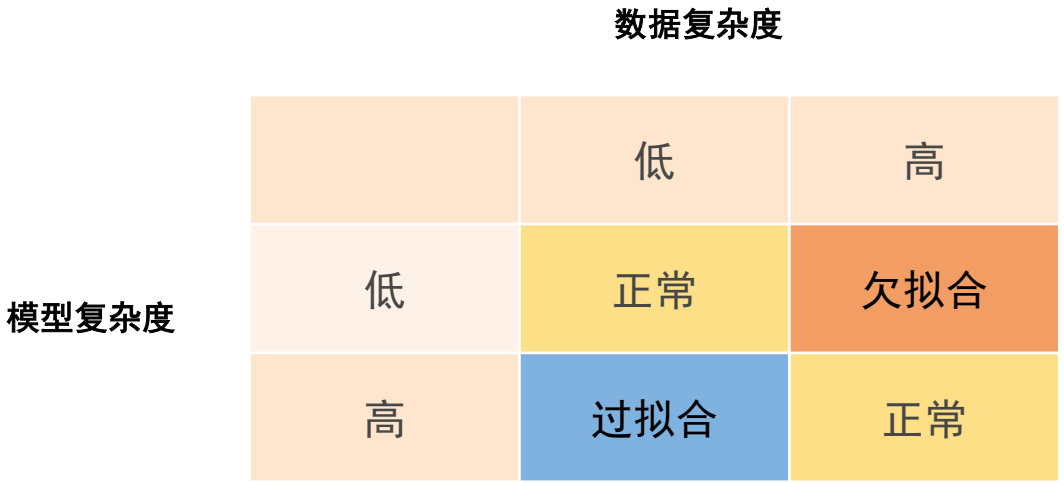
- 验证数据集：用于评估模型的数据集
  - 例如：取出50%的训练数据
  - 不应与训练数据混在一起（#1 常见错误）
- 测试数据集：只可以使用一次数据集，例如
  - 未来的考试
  - 我买的房子售价
  - Kaggle的私人排行榜中使用的数据集

K 折交叉验证

- 在没有足够的数据时非常有用
- 算法：
  - 将训练数据划分为 K 个部分
  - 对于 i = 1，...，K
  - 使用第 i 部分作为验证集，其余部分用于训练
  - 报告 K 个部分在验证时的平均错误
- 常见 K 值选择：5 - 10

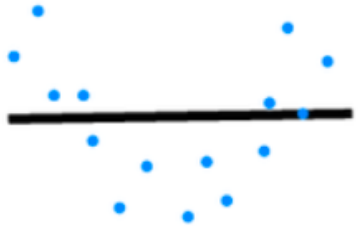
欠拟合，过拟合

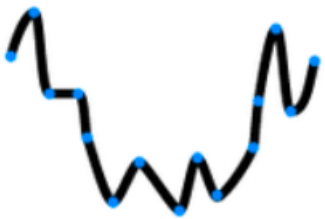
欠拟合 和 过拟合



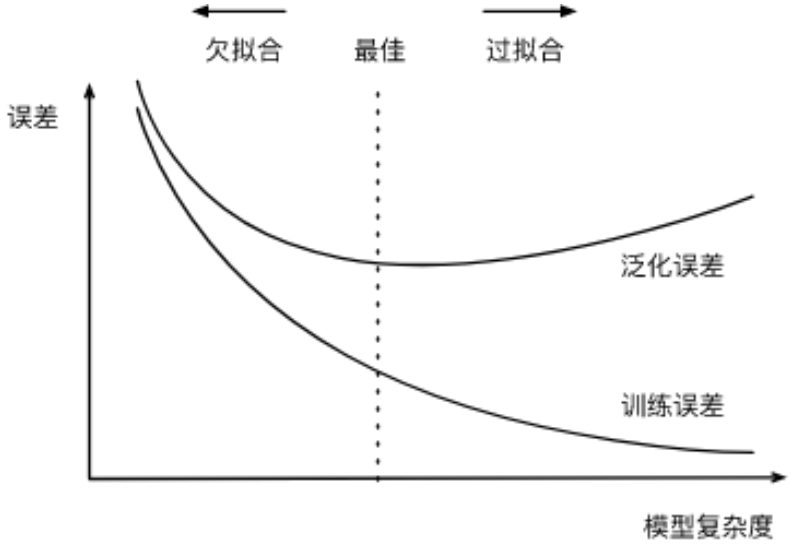
模型复杂度

- 即“可适应不同数据分布”的能力
- 低容量模型难以适应训练集
  - 欠拟合
- 高容量模型可以记忆训练集
  - 过拟合





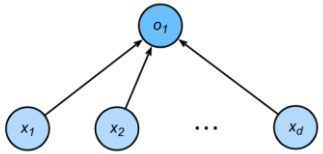
模型复杂度的影响



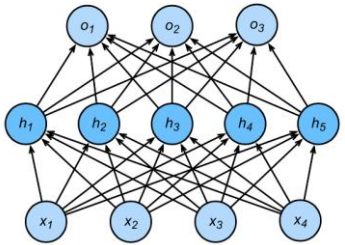
估计模型复杂度

- 很难比较不同算法之间的复杂度
  - 例如 树与神经网络
- 给定算法族，两个主要因素很重要：
  - 参数个数
  - 每个参数采用的值

$d+1$

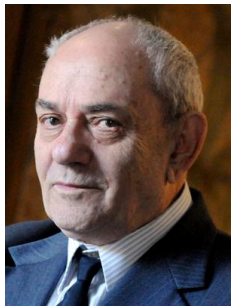


$(d+1)m+(m+1)k$



VC维度

- 统计学习理论的中心主题
- 对于分类模型，VC维度等于这个数据集的大小，无论我们如何分配标签，都存在一个模型来完美地对它进行分类

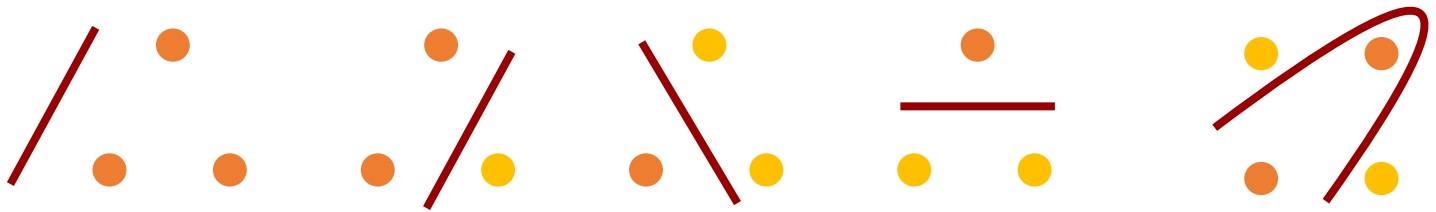


Vladimir **V**apnik

Alexey **C**hervonenkis

线性分类器的VC维度

- 2-D感知机（输入为2-D ）：VCdim = 3
- 可以分3类，但不能分4类（XOR）



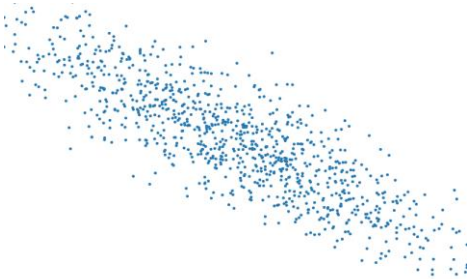
- 具有N个参数的感知机：VCdim = N
- 一些多层感知机：VCdim =$O(N\log_2(N))$

VC维度的效用

- 提供理论解释模型的工作原理
  - 限制了训练误差和泛化误差之间的差距
- 在深度学习的实践中很少使用
  - 边界过于宽松
  - 难以计算深度神经网络的 VC 维数
  - 其他统计学习理论工具也是如此

数据复杂度

- 多种因素很重要
  - 例子数量
  - 每个示例中的元素数量
  - 时间/空间结构
  - 多样性



权重衰减

为什么正则化

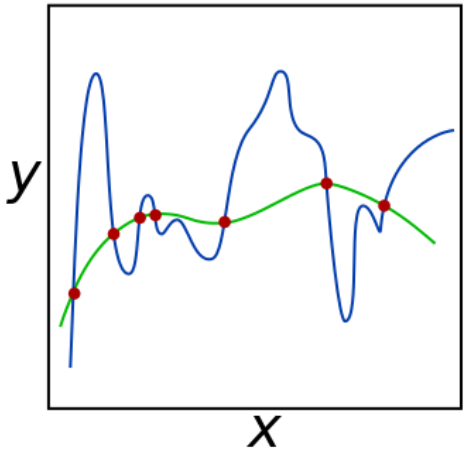
- 动机：缓解过拟合
- 解决方法：
  - 收集更多的训练数据，成本很高
  - 简单地丢弃特征过于生硬。阶数上的微小变化，也会显著增加我们模型的复杂性
  - 通过函数与零的距离来衡量函数的复杂度

平方正则化作为“硬”约束

- 通过限制参数值范围来降低模型复杂性(直接)

$$min\ell(\mathbf{w}, b) \text{subject to } \| \mathbf{w} \|^2 \leq \theta$$

- 通常不限制偏差参数 b
- 做或不做在实践中几乎没有区别
- 小 θ 意味着更多的正则化



平方正则化作为“软”约束

- 对于每一个 θ，我们都可以找到 λ 将硬约束版本重写为

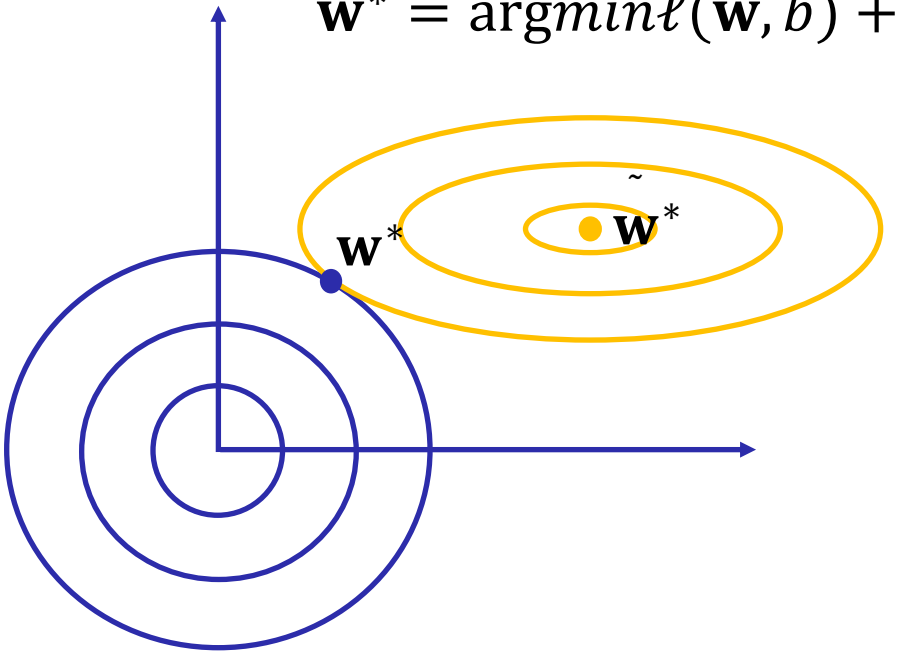
$$min\ell(\mathbf{w}, b) + \frac{\lambda}{2} \| \mathbf{w} \|^2$$

  - 可以用拉格朗日乘数法证明
  - 超参数 λ 控制正则化的重要性
  - $\lambda = 0$： 没有效果

$$\lambda \to \infty, \mathbf{w}^* \to \mathbf{0}$$

图解说明最优解

$$\mathbf{w}^* = \arg min \ell(\mathbf{w}, b) + \frac{\lambda}{2} \| \mathbf{w} \|^2$$



$$\tilde{\mathbf{w}}^* = \arg min \ell(\mathbf{w}, b)$$

更新规则

- 计算梯度
- 在时刻 t 更新权重

$$\frac{\partial}{\partial \mathbf{w}}\left(\ell(\mathbf{w}, b) + \frac{\lambda}{2}\| \mathbf{w} \|^2\right) = \frac{\partial \ell(\mathbf{w}, b)}{\partial \mathbf{w}} + \lambda \mathbf{w}$$

- 通常 $\eta\lambda < 1$，在深度学习中也称为权重衰减

$$\mathbf{w}_{t+1} = (1 - \eta\lambda)\mathbf{w}_t - \eta \frac{\partial \ell(\mathbf{w}_t, b_t)}{\partial \mathbf{w}_t}$$

丢弃法(dropout)

重新审视过拟合

- 当面对更多的特征而样本不足时，线性模型往往会过拟合
- 当给出更多样本而不是特征，通常线性模型不会过拟合
- 线性模型泛化的可靠性是有代价
  - 线性模型没有考虑到特征之间的交互作用
- 泛化性和灵活性之间的这种基本权衡被描述为偏差-方差权衡（bias-variance tradeoff）
  - 线性模型有很高的偏差：它们只能表示一小类函数。然而，这些模型的方差很低：它们在不同的随机数据样本上可以得出相似的结果
  - 深度神经网络位于偏差-方差谱的另一端。神经网络并不局限于单独查看每个特征，而是学习特征之间的交互

深度网络的泛化性质

- 深度神经网络位于偏差-方差谱的另一端。与线性模型不同，神经网络并不局限于单独查看每个特征，而是学习特征之间的交互
- 即使我们有比特征多得多的样本，深度神经网络也有可能过拟合
- 泛化性质的数学基础仍然是悬而未决

动机

- 在输入的适度变化下，一个好的模型应该是稳定的
  - 用输入噪声训练相当于 Tikhonov 正则化
  - 丢弃法：将噪音注入内部隐藏层

添加没有偏差的噪音

- 将噪声添加到 x 中以以获得 x'：

$$\mathbf{E}[\mathbf{x}'] = \mathbf{x}$$

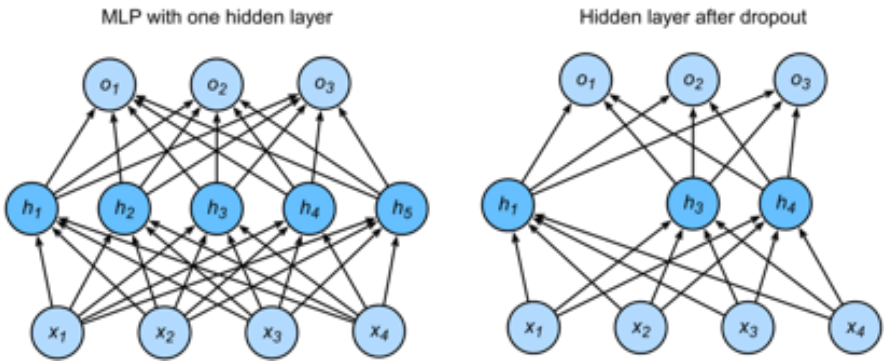
- 丢弃法将噪声添加到每个元素:

$$x_i' = \begin{cases} 0 & \text{with probablity } p \\ \dfrac{x_i}{1-p} & \text{otherise} \end{cases}$$

丢弃法 – 训练

- 通常在全连接层的输出上使用丢弃法

$$
\begin{aligned}
\mathbf{h} &= \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) \\
\mathbf{h}' &= \text{dropout}(\mathbf{h}) \\
\mathbf{o} &= \mathbf{W}_2\mathbf{h}' + \mathbf{b}_2 \\
\mathbf{y} &= \text{softmax}(o)
\end{aligned}
$$



丢弃法 – 推断

- 正规化仅用于模型训练
- 丢弃法在推断中用于：

$$\mathbf{h}' = \text{dropout}(\mathbf{h})$$

  - 保证确定性结果

总结

- 模型选择
  - 训练误差和泛化误差
  - K 折交叉验证
- 欠拟合与过拟合
  - 模型复杂度
  - VC 维度
- 权重衰减
- 丢弃法