



Machine learning application

目录

Machine Learning application
Bayes hypothesis and Bayes noise
Bayes Hypothesis
定义
Bayes Hypothesis的作用
特定情境下的Bayes Hypothesis
定理！
贝叶斯误差 (Bayes Error)
Bias-Variance Trade-off
泛化误差的分解
偏差-方差权衡的重要性
偏差-方差权衡的图形解释
Example!
Deep Learning(深度学习)
Artificial neural networks
激活函数
单层感知器 (Single-layer Perceptron)
多层感知器 (Multilayer Perceptron, MLP)



Bayes hypothesis and Bayes noise

在机器学习中，监督学习的目标是从输入（也称为特征空间，记为 \mathcal{X} ）到输出（或标签空间，记为 \mathcal{Y} ）中找到一个映射关系，使得我们的预测与实际数据尽可能接近。我们假设数据是从某个分布 μ_Z 中生成的，并且我们使用一个损失函数 L 来衡量预测与真实值之间的误差。



Bayes Hypothesis



定义

为了找到最优的预测，我们需要找到一个能够最小化泛化误差 R 的函数（或模型）。这种“最优的”模型在这里称为 **Bayes Hypothesis**，记作 h^{Bayes} 。它的定义如下：

$$h^{\text{Bayes}} \in \arg \min_{h: \mathcal{X} \rightarrow \mathcal{Y}} \int_Z L(y, h(x)) d\mu_Z(x, y).$$

在这个公式中：

- $\arg \min_{h: \mathcal{X} \rightarrow \mathcal{Y}}$ 表示我们在所有从输入空间 \mathcal{X} 到输出空间 \mathcal{Y} 的函数中寻找最小化泛化误差的那个函数。
- $\int_Z L(y, h(x)) d\mu_Z(x, y)$ 是一个积分表达式，表示在所有可能的数据 (x, y) 上，对损失函数 $L(y, h(x))$ 的加权平均（权重由数据分布 μ_Z 确定）。
- 损失函数 $L(y, h(x))$ 表示真实值 y 与预测值 $h(x)$ 之间的误差。



Bayes Hypothesis的作用

Bayes Hypothesis可以被看作是一个“理论上的最优预测器”，即在给定的损失函数 L 下，它是无法被进一步改进的。换句话说，它为我们提供了一个基准，我们的任何模型都试图去接近这个最优值。

在监督学习中，我们通过模型训练，隐式地在逼近这个Bayes Hypothesis，尽管我们可能永远无法达到它。

特定情境下的Bayes Hypothesis

在一些特殊情况下，Bayes Hypothesis是可以具体表达出来的。例如，在 平方损失 (quadratic loss) 下，Bayes Hypothesis可以写作：

$$h^{\text{Bayes}}(x) = \mathbb{E}[Y|X = x],$$

即在给定输入 $X = x$ 的条件下，对应输出 Y 的条件期望。这意味着在平方损失的情况下，最优的预测就是每个输入点上输出的平均值。这也是统计中最优预测器的一个经典结果。

对于 零一损失 (zero-one loss)，也可以推导出对应的Bayes Hypothesis。

好的，Penny，让我们详细地解释一下这个定理和它的证明过程。

定理 1

问题背景：我们考虑一个二分类问题，其中输入空间为 $\mathcal{X} \subset \mathbb{R}^M$ ，标签空间为 $\mathcal{Y} = \{-1, 1\}$ 。在这个设定下，损失函数是 零一损失 (zero-one loss)，表示为 $L(y, y') = 1[y \neq y']$ ，意思是当预测值和真实值不同的时候损失为1，否则为0。假设 $(X, Y) \sim \mu_Z$ ，其中 X 和 Y 的联合分布为 μ_Z 。

结论：Bayes最优预测函数 $h^{\text{Bayes}}(x)$ 在这种情况下可以表示为：

$$h^{\text{Bayes}}(x) = 2 \cdot 1[\mathbb{P}(Y = 1|X = x) > 1/2] - 1$$

或等价地写为：

$$h^{\text{Bayes}}(x) = \begin{cases} 1, & \text{如果 } \mathbb{P}(Y = 1|X = x) > 1/2, \\ -1, & \text{如果 } \mathbb{P}(Y = -1|X = x) \geq 1/2. \end{cases}$$

这表明：在二分类问题中，最优预测总是选择条件概率较大的那个标签。比如，如果在给定 x 的条件下， $Y = 1$ 的概率超过 $1/2$ ，那么我们预测 $Y = 1$ ，否则预测 $Y = -1$ 。

解释

该定理说明了在二分类问题中，最优模型应选择具有更大条件概率的标签。这类似于投票中的“多数胜出”原则。即在给定输入 x 时，哪个类别的条件概率更高，我们就预测为哪个类别。

证明思路

1. 泛化误差的定义

对于任何预测函数 $h : \mathcal{X} \rightarrow \mathcal{Y}$ ，它的泛化误差 $R(h)$ 可以写成：

$$\int_Z L(y, h(x)) d\mu_Z(x, y) = \int_{\mathcal{X}} \sum_{y \in \mathcal{Y}} 1[y \neq h(x)] \pi_{Y|X=x}(y) \pi_{\mathcal{X}}(x) dx,$$

其中 $\pi_{Y|X=x}(y)$ 表示在给定 $X = x$ 的条件下 $Y = y$ 的条件概率， $\pi_{\mathcal{X}}(x)$ 表示 x 的边缘概率密度。

2. 条件概率最大化

在给定 $x \in \mathcal{X}$ 的条件下，Bayes最优预测 $h^{\text{Bayes}}(x)$ 应选择能够最小化零一损失的标签。换句话说，我们选择 y 使得 $\pi_{Y|X=x}(y)$ 最大。

由于 \mathcal{Y} 只有两个标签（1和-1），如果 $\mathbb{P}(Y = 1|X = x) > 1/2$ ，我们选择 $Y = 1$ ；如果 $\mathbb{P}(Y = -1|X = x) \geq 1/2$ ，我们选择 $Y = -1$ 。

3. 泛化误差最小化

我们证明在这种选择下， $h^{\text{Bayes}}(x)$ 使得泛化误差 $R(h)$ 最小。直观上，选取概率较大的标签可以最小化错误预测的风险，从而最小化损失。

贝叶斯误差 (Bayes Error)

Bayes误差，记作 $R^{\text{Bayes}} := R(h^{\text{Bayes}})$ ，是基于 Bayes Hypothesis 计算的泛化误差。即使有最优的机器学习模型，这个误差依然存在，这是由于数据的固有不确定性导致的。这种误差被称为不可约误差 (irreducible error)。

在二分类问题中，当使用零一损失时，最优的Bayes分类器会选择条件概率较大的标签，这可以保证错误率最小。这就是为什么机器学习算法通常追求逼近 Bayes 最优分类器的原因——它代表了分类问题中可以达到的最佳性能。

在机器学习中，偏差-方差权衡（Bias–Variance Trade-off）是选择模型时的一个关键因素。不同的模型可能会有不同的偏差和方差，这直接影响它们在训练集和测试集上的表现。偏差和方差的平衡关系对于减少模型的泛化误差（即测试误差）非常重要。它解释了为什么我们需要在模型复杂度和模型泛化能力之间做权衡，以避免模型出现欠拟合和过拟合。

假设我们有一个假设 $h \in \mathcal{H}$ ，它属于一个假设空间 \mathcal{H} 。

泛化误差的分解

我们可以将某个假设 h 的泛化误差 $R(h)$ 分解为三个部分：

$$R(h) = R^{\text{Bayes}} + \underbrace{\left(R(h) - \inf_{h' \in \mathcal{H}} R(h') \right)}_{= \text{'variance'}} + \underbrace{\left(\inf_{h' \in \mathcal{H}} R(h') - R^{\text{Bayes}} \right)}_{= \text{'bias'}}.$$

在这个公式中：

1. R^{Bayes} : 这是 Bayes 误差，表示了数据本身的不可约误差。即使我们有一个完美的模型，由于数据中的随机性或噪声，某些数据点可能还是无法正确分类。因此， R^{Bayes} 是“问题固有的误差”。
2. **偏差 (Bias)** : $\inf_{h' \in \mathcal{H}} R(h') - R^{\text{Bayes}}$ 表示最优模型的误差与 Bayes 误差之间的差距。偏差反映了模型的“固有误差”——即使我们在假设空间 \mathcal{H} 中找到最优模型，它的误差仍然与最理想的模型有一定距离。

这里的偏差告诉我们，假设空间 \mathcal{H} 中的最优模型与 Bayes 模型之间的差距。如果 \mathcal{H} 太小，无法包含一个接近 Bayes 假设的模型，则偏差会较大。

- **选择合适的假设空间**: 在选择假设空间 \mathcal{H} 时，需要确保它足够小，使得模型可以有效泛化（即方差小），但也要足够大，以便包含一个接近 Bayes 假设的函数（即偏差小）。
- **嵌套假设空间**: 在实际应用中，我们经常有多个嵌套的假设空间 $\mathcal{H}_1 \subset \mathcal{H}_2 \subset \dots$ 。当我们选择更大的假设空间时，偏差通常会减少，但方差可能会增加。因此，我们需要找到一个合适的假设空间，使得偏差和方差都保持在合理范围内。

1. **方差 (Variance)** : $R(h) - \inf_{h' \in \mathcal{H}} R(h')$ 表示当前模型 h 与假设空间 \mathcal{H} 中最优模型的泛化误差之间的差异。方差反映了模型对于数据波动的敏感性。高方差通常意味着模型过于复杂，容易对训练数据过拟合，从而在测试数据上表现不佳。

1. **经验误差 $\tilde{R}(h)$** : 通常情况下，我们通过最小化训练数据上的经验误差 $\tilde{R}(h)$ 来选择模型。这里假设 h 是使 $\tilde{R}(h)$ 最小的模型。

2. **泛化误差界限**: 我们可以利用经验误差和泛化误差之间的界限来描述这个误差的大小。假设 h 最小化 $\tilde{R}(h)$ ，那么我们有：

$$-\tilde{R}(h) \leq \inf_{h' \in \mathcal{H}} R(h') - \tilde{R}(h) \leq R(h) - \tilde{R}(h) \leq \text{Rad}(\mathcal{H}) + \sqrt{\frac{\log(1/\delta)}{2N}},$$

其中， $\text{Rad}(\mathcal{H})$ 是假设空间的 Rademacher 复杂度， δ 是置信水平， N 是样本数量。

3. **方差估计**: 通过这个不等式，我们可以进一步得到方差的上界：

$$R(h) - \inf_{h' \in \mathcal{H}} R(h') \leq 2 \max \left\{ \tilde{R}(h), \text{Rad}(\mathcal{H}) + \sqrt{\frac{\log(1/\delta)}{2N}} \right\},$$

其中右边的项表示了模型方差的一个上界。

综合起来，泛化误差 $R(h)$ 由以下三部分组成：

- Bayes 误差：这是问题固有的误差，不可避免。
- 偏差：这是模型选择导致的误差。如果模型过于简单（例如使用线性模型来拟合非线性数据），则偏

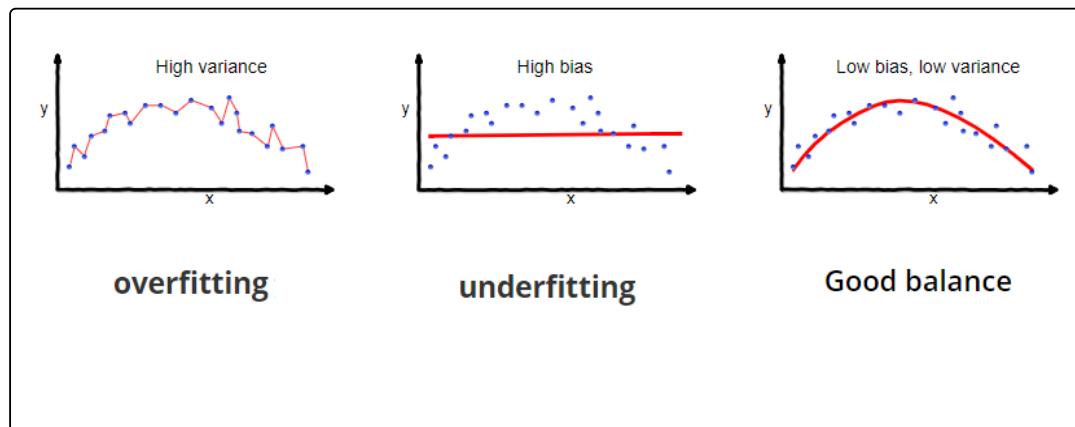
差会较大。

- 方差：这是模型对数据变化的敏感性。若模型过于复杂，它对训练数据的细节也会非常敏感，从而导致在测试集上的表现不稳定。

💡 偏差-方差权衡的重要性

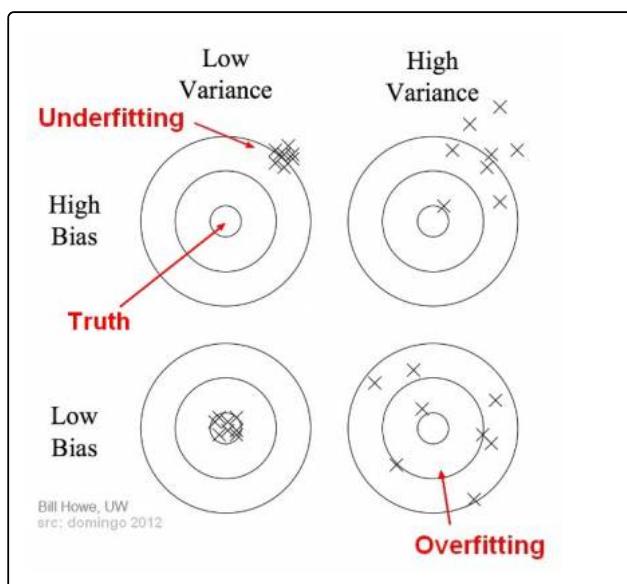
在模型选择时，我们需要在偏差和方差之间找到平衡：

- 如果模型过于简单，则偏差大，但方差小，称为欠拟合。
- 如果模型过于复杂，则偏差小，但方差大，称为过拟合。

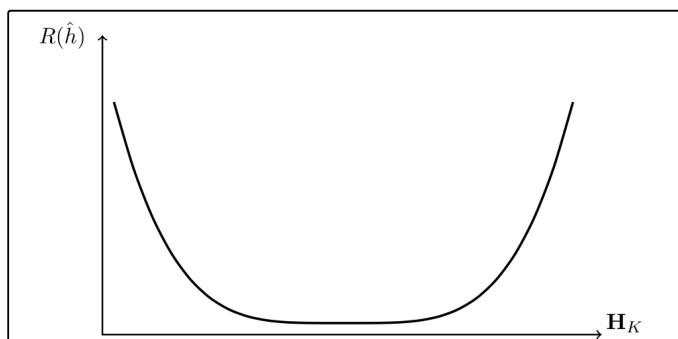


一个好的模型应该适度复杂，使得偏差和方差都尽可能小，从而使得泛化误差最低。

💡 偏差-方差权衡的图形解释



在上图中，目标的中心是一个完美预测正确值的模型。随着我们远离靶心，我们的预测变得越来越糟。我们可以重复我们的模型构建过程，以获得对目标的单独命中。



上面这张图很好地展示了泛化误差 $R(\hat{h})$ 与假设空间 \mathcal{H}_K 的关系，帮助我们理解偏差-方差权衡如何影响模型选择。

图中横轴表示假设空间的复杂度 \mathcal{H}_K ，随着假设空间的增大，模型能够拟合更复杂的数据模式。纵轴表示模型的泛化误差 $R(\hat{h})$ 。当我们选择一个假设空间 \mathcal{H}_K 来最小化经验误差时，泛化误差的变化通常表现为一个U型曲线：

1. 左侧（高偏差区域）：

- 在假设空间较小的情况下，模型可能过于简单，不能有效地捕捉数据的特征。这导致了高偏差（bias），泛化误差较大。
- 此时，模型的泛化误差主要受偏差影响，因为它无法很好地拟合数据的结构，表现为欠拟合。

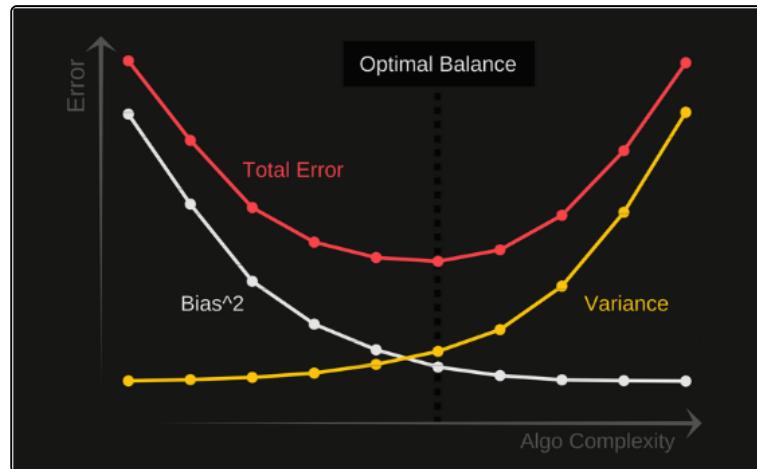
2. 右侧（高方差区域）：

- 随着假设空间的增大，模型可以更灵活地拟合数据。然而，假设空间过大时，模型可能会过于复杂，以至于对训练数据的噪声也进行了拟合。这导致了高方差（variance），泛化误差再次增大。
- 此时，模型的泛化误差主要受方差影响，因为它在训练数据上表现良好，但在测试数据上表现不佳，表现为过拟合。

3. 中间区域（偏差-方差权衡的最佳点）：

- 在假设空间复杂度适中的情况下，模型的泛化误差最低，这是偏差和方差之间的最佳平衡点。
- 这里的模型既不过于简单也不过于复杂，能很好地概括数据的规律，同时对数据的波动不敏感，通常是模型选择的理想位置。

当然这张图也可以更好表现出这个关系



💡 Example1

假设：

1. 输入空间 \mathcal{X} 和输出空间 \mathcal{Y} 都是实数域 \mathbb{R} 。
2. 损失函数 L 是平方损失函数（quadratic loss function），即 $L(y, y') = (y - y')^2$ ，用于衡量预测值与真实值之间的差异。
3. 字典 $\{\psi_1, \psi_2, \dots\}$ 是一组基函数，属于 $L^2(\mathcal{X}, \mu_{\mathcal{X}}; \mathcal{Y})$ 空间（即平方可积函数空间）。这组基函数可以用来表示目标函数的近似。

我们通过基函数的数量 K 来构建一个包含不同模型复杂度的假设空间族。具体定义如下：

$$\mathcal{H}_K := \left\{ \sum_{k=1}^K w_k \psi_k(\cdot) : w_1, \dots, w_K \in \mathbb{R} \right\},$$

其中， $K \in \mathbb{N}$ 表示我们使用的基函数的数量。

解释

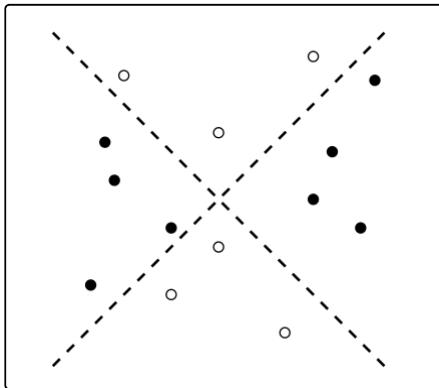
○ 假设空间族 \mathcal{H}_K : 这里的 \mathcal{H}_K 表示由 K 个基函数组成的假设空间。每一个假设空间 \mathcal{H}_K 包含了所有可以表示为基函数加权和的函数，权重为 w_1, \dots, w_K 。

○ 模型复杂度: K 的大小决定了假设空间的复杂度。 K 越大，假设空间 \mathcal{H}_K 越复杂，可以拟合的函数种类越多； K 越小，假设空间越简单。

在实际应用中，基函数可以是不同的类型，例如多项式、傅里叶基、或小波基。对于不同的 K 值，我们可以得到一系列嵌套的假设空间，从而实现不同的模型复杂度。这与偏差-方差权衡有关：当 K 较小时，模型可能会欠拟合（高偏差）；当 K 较大时，模型可能会过拟合（高方差）。

Deep Learning (深度学习)

之前的学习讲的是线性分类器，具有一定的局限性，比如，某些数据分布的类别并不能通过一个超平面（即一个直线或一个平面）来分开。下图就是一个例子，展示了一组带标签的数据点（黑色点和白色圈），这些数据点的分布使得无法使用单一的超平面将两类分开。



上图展示了一组位于 $\mathcal{X} := \mathbb{R}^2$ 上的数据点，标签为 $\mathcal{Y} := -1, 1$ 。其中，黑点（●）和白圈（○）分别代表不同的类别。由于这些数据点的分布特性，单一的超平面无法将它们分开。

像这样的复杂分类问题，我们需要更复杂的假设空间来实现更好的分离。深度学习是一种能够组合简单假设以构建更复杂假设空间的方法，它使用多层神经网络来实现这一点。

Artificial neural networks

这一部分介绍了人工神经网络中常用的激活函数（activation function）。激活函数是神经网络的关键组件，它们为神经元引入非线性，使得网络可以学习复杂的模式和特征，可以任意逼近任何非线性函数，这样神经网络就可以应用到众多非线性模型中。

激活函数

假设 $\varphi: \mathbb{R} \rightarrow \mathbb{R}$ 是一个标量函数，我们将其称为激活函数。在神经网络中，激活函数可以应用于向量的每一个分量，定义为：

$$\varphi(x) := (\varphi(x_1), \dots, \varphi(x_n))^T,$$

其中 $x \in \mathbb{R}^n$, $n \in \mathbb{N}$ 。

以下是一些常见的激活函数：

1. Heaviside函数（阶跃函数）：

$$\varphi(x) := 1[x \geq 0] \quad (x \in \mathbb{R}),$$

它是一个二值函数，在 $x \geq 0$ 时输出 1，否则输出 0。

2. Logistic函数（逻辑函数或S形函数）：

$$\varphi(x) := \frac{1}{1 + \exp(-x)} \quad (x \in \mathbb{R}),$$

它将输入压缩到 $[0, 1]$ 区间，常用于二分类模型的输出层。

3. ReLU (Rectified Linear Unit) :

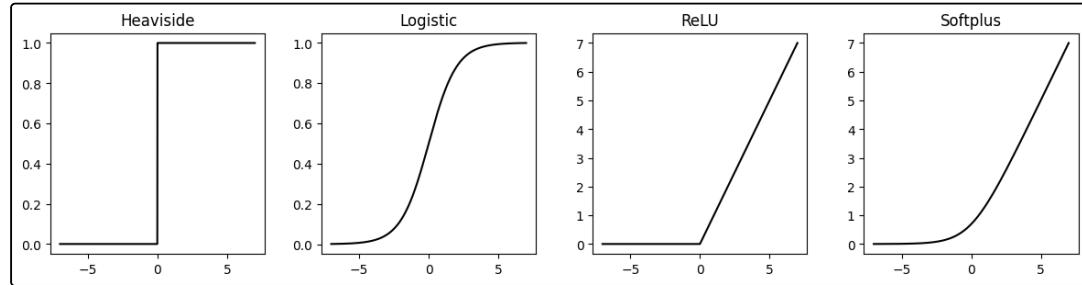
$$\varphi(x) := \max\{x, 0\} = x1[x \geq 0] \quad (x \in \mathbb{R}),$$

ReLU在 $x \geq 0$ 时输出 x , 否则输出0。ReLU是现代神经网络中最常用的激活函数之一, 因其计算简单且缓解了梯度消失问题。

4. Softplus函数:

$$\varphi(x) := \log(1 + \exp(x)) \quad (x \in \mathbb{R}),$$

Softplus是ReLU的光滑版本, 适用于需要平滑非线性的场景。



单层感知器 (Single-layer Perceptron)

假设我们有一个输入空间 $\mathcal{X} := \mathbb{R}^M$ 和标签空间 $\mathcal{Y} \subset \mathbb{R}^J$ 。单层感知器的定义如下:

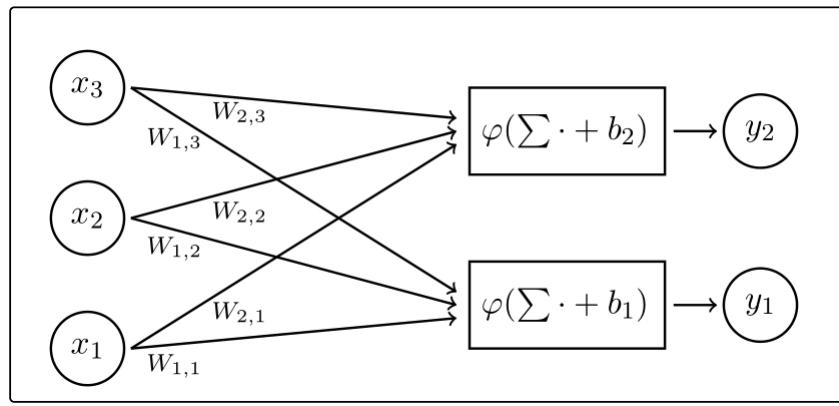
$$H(x; (W, b)) := \varphi(Wx + b),$$

其中:

- $W \in \mathbb{R}^{J \times M}$: 权重矩阵, 每个元素表示连接不同输入和输出之间的权重。
- $b \in \mathbb{R}^J$: 偏置向量, 用于调整激活函数的输出。

该模型通过激活函数 φ (例如ReLU或sigmoid) 对线性组合 $Wx + b$ 进行非线性变换。这个结构类似于大脑神经元的行为, 每个输入信号都会被加权并传递给激活函数。

下图是单层感知器的示意图, 它包含三个输入节点 (x_1, x_2, x_3) , 两个输出节点 (y_1, y_2) , 以及相应的权重和偏置。



多层感知器 (Multilayer Perceptron, MLP)

为了处理更复杂的问题, 单层感知器通常不足以拟合复杂的非线性关系。为了解决这个问题, 我们可以将多个单层感知器堆叠起来形成多层感知器, 即神经网络中的前馈神经网络。其模型表示为:

$$H(x, (W^{(l)}, b^{(l)})_{l=1}^L) := a^{(L)},$$

其中:

- L : 网络的层数。

○ $a^{(\ell)}$: 第 ℓ 层的输出, 递归定义如下:

○ $a^{(0)} := x$ 为输入层的输出。

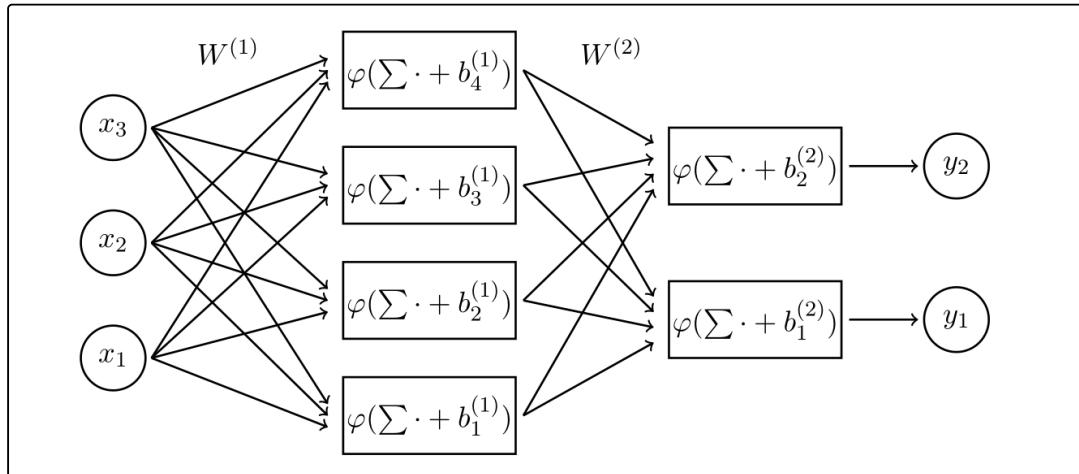
○ $a^{(\ell)} := \varphi(W^{(\ell)}a^{(\ell-1)} + b^{(\ell)})$, 其中 $\ell = 1, \dots, L$ 。

这里, $W^{(\ell)}$ 和 $b^{(\ell)}$ 分别表示第 ℓ 层的权重矩阵和偏置向量。

在多层感知器中, 参数空间 \mathcal{W} 由所有层的权重矩阵和偏置向量组成, 定义为:

$$\mathcal{W} := \mathbb{R}^{K(1) \times K(0)} \times \mathbb{R}^{K(1)} \times \dots \times \mathbb{R}^{K(L) \times K(L-1)} \times \mathbb{R}^{K(L)},$$

其中 $K(0) = M$ 是输入的维数, $K(L) = J$ 是输出的维数, 其他 $K(\ell)$ 表示每层的神经元数量。



上图展示了一个两层多层感知器 (multilayer perceptron, MLP) 的结构。

在图中我们可以得到以下几点:

1. 输入层: 输入向量为 $\mathcal{X} = \mathbb{R}^3$, 即输入有三个特征 (x_1, x_2, x_3) 。

2. 隐藏层:

○ 第一层隐藏层有 $K(1) = 4$ 个神经元。

○ 权重矩阵 $W^{(1)}$ 的维度是 4×3 (4个输出和3个输入), 属于 $\mathbb{R}^{4 \times 3}$ 。

○ 偏置向量 $b^{(1)}$ 是一个 4 维向量, 属于 \mathbb{R}^4 。

3. 输出层:

○ 输出层有 $K(2) = 2$ 个神经元, 即输出为 $\mathcal{Y} = \mathbb{R}^2$ 。

○ 权重矩阵 $W^{(2)}$ 的维度是 2×4 (2个输出和4个输入), 属于 $\mathbb{R}^{2 \times 4}$ 。

○ 偏置向量 $b^{(2)}$ 是一个 2 维向量, 属于 \mathbb{R}^2 。

整个network的数学表示如下:

1. 第一层计算:

○ 首先计算 $a^{(1)} = \varphi(W^{(1)}x + b^{(1)})$, 其中 x 是输入向量, $W^{(1)}$ 是第一层的权重矩阵, $b^{(1)}$ 是偏置向量, φ 是激活函数。

○ 计算结果 $a^{(1)}$ 是第一层的输出向量, 它将作为下一层的输入。

2. 第二层计算:

○ 第二层的输出为 $a^{(2)} = \varphi(W^{(2)}a^{(1)} + b^{(2)})$ 。

○ 这里的 $a^{(2)}$ 就是网络的最终输出, 用于预测或分类任务。

- **隐藏层 (Hidden Layers)**：在多层感知器中，除了最后一层（输出层 $a^{(L)}$ ）以外的所有中间层都被称为隐藏层。这些层不直接与网络的输入或输出相连，而是负责处理和提取输入数据中的特征信息。
- **输出层 (Output Layer)**：最后一层 $a^{(L)}$ 是输出层，负责输出最终的预测结果。在网络结构中，我们通常将这层直接映射为输出数据的格式。
- **输入层 (Input Layer)**：输入层并不直接参与神经元的计算，表示为 $a^{(0)} = x$ 。它只是用来接收输入数据，并将其传递给第一个隐藏层。

1. 神经网络的命名：

- 单层感知器 (Single-layer Perceptron) 和多层感知器 (Multilayer Perceptron, MLP) 统称为人工神经网络 (Artificial Neural Network, NN)。
 - 如果一个神经网络包含多个隐藏层 (即深层结构)，那么这个网络也被称为深度神经网络 (Deep Neural Network, DNN)。
2. 神经元 (Neurons)：上面两个感知器示意图中的每个矩形框代表一个神经元。神经元是网络的基本计算单元，每个神经元会接受输入、进行加权求和，然后通过激活函数生成输出。
3. 全连接 (Fully Connected)：全连接神经网络是一种多层感知器结构。每一层的每一个节点与下一层的每一个节点相连接，并伴有运算关系。这种连接方式增加了模型的复杂度，但同时会带来计算开销。在大型神经网络中，使用全连接并不总是最优的，因为这样会导致过多的参数。

- 单层感知器只能构建简单的线性模型，通常用于简单的线性可分问题。

- 多层感知器通过多层结构引入复杂的非线性映射，适用于复杂的分类和回归任务。