

# 机器学习：神经网络与深度学习

## 目录

- 1. 模式与结构 (Patterns and structure)
- 2. 维度诅咒 (The Curse of Dimensionality)
- 3. 神经网络简介 (Neural networks)
  - 3.1 什么是神经网络? (What is a neural network?)
  - 3.2 感知机 (The perceptron)
  - 3.3 层 (Layers)
  - 3.4 架构 (The architecture)
- 4. 感知机的功能 (What can perceptrons do?)
  - 4.1 作为线性分类器的感知机 (The perceptron as a linear classifier)
  - 4.2 作为基本逻辑函数的感知机 (The perceptron as a basic logical function)
  - 4.3 增加深度以推导新的逻辑函数 (Adding depth to derive new logical functions)
  - 4.4 作为网格模式检测器的感知机 (The perceptron as a grid pattern detector)
  - 4.5 结合线性分类器和逻辑函数 (Combining linear classifiers and logical functions)
- 5. 作为机器学习模型的神经网络 (Neural networks as machine learning models)
  - 5.1 机器学习流程：回顾 (Machine learning pipelines: Reminder)
  - 5.2 作为可调优流程的神经网络 (Neural networks as tunable pipelines)
  - 5.3 训练神经网络：代价函数 (Training neural networks: Cost function)
  - 5.4 梯度下降和反向传播 (Gradient descent and back-propagation)
  - 5.5 训练注意事项 (Training considerations)
  - 5.6 迁移学习 (Transfer learning)
- 6. 层的类型 (Types of layers)
  - 6.1 全连接层 (Fully-connected layer)
  - 6.2 网格数据中的等方差 (Equivariance in grid data)
  - 6.3 卷积层 (Convolutional layers)
  - 6.4 池化层 (Pooling layers)
- 7. 深度学习架构 (Deep learning architectures)
  - 7.1 VGG16 网络示例 (Example: The VGG16 network)
- 8. 总结 (Summary)
  - 8.1 神经网络：三种视角 (Neural networks: the three views)
  - 8.2 神经网络：一个机器学习模型家族 (Neural networks: A family of machine learning models)
  - 8.3 神经网络：通用机器 (Neural networks: Universal machines)

## 1. 模式与结构 (Patterns and structure)

- 定义:
  - 模式 (Patterns) 是数据中的规律性 (regularities in our data)。
  - 结构 (structure) 是目标群体中的规律性 (a regularity in our target population)。
- 机器学习项目依赖于通过识别数据中的模式来发现潜在的结构 (Machine learning project relies on discovering the underlying structure by identifying patterns in data)。
- 主要挑战是区分相关的模式和虚假的模式 (The main challenge is to distinguish relevant from spurious patterns)。

## 2. 维度诅咒 (The Curse of Dimensionality)

- 随着维度增加，数据变得更稀疏 (As the dimensionality increases, data becomes sparser)。

- **维度诅咒的警告:** 不相关的属性不会相互抵消，它们会表现为虚假模式。添加更多属性 (以防万一) 实际上会导致模型性能变差 (Irrelevant attributes do not cancel each other out, they show up as spurious patterns. Adding more attributes (just in case) can actually result in worse-performing models)。
- **解决方案:** 我们可以使用特征选择 (feature selection) 技术来降低问题的维度，但首先，让我们使用我们的领域知识 (domain knowledge)。

## 3. 神经网络简介 (Neural networks)

### 3.1 什么是神经网络？(What is a neural network?)

- **定义:** 神经网络是一种受人类神经系统松散启发的计算系统 (computing system)，通常用作机器学习模型的一个家族 (family)。
- **从功能的 (functional) 角度来看，神经网络:**
  - 根据输入 (预测变量) 产生输出 (标签) (Produces an output (label) given an input (predictors))。
  - 具有可以调整 (训练) 的权重 (参数) (Has weights (parameters) that can be tuned (trained))。
  - $W$  表示神经网络的所有权重,  $h_W(x)$  表示预测值  $\hat{y}$  依赖于  $W$ 。
- **从计算的 (computational) 角度来看，神经网络由相互连接的单元组成，这些单元 (松散地) 模仿神经元 (mimic neurons)。**这种架构具有吸引力，因为:
  - **神经科学 (Neuroscience)** 表明生物神经网络可以解决任何问题。
  - **数学 (Mathematics)** 表明人工神经网络可以再现任何输入/输出关系，只要它们足够复杂。
- **因此，神经网络模型家族可以被视为通用机器 (universal machine)。**

### 3.2 感知机 (The perceptron)

- **定义:** 感知机是神经网络的基本单元 (basic unit)。它由一个权重向量 (weight vector)  $w$  和一个激活函数 (activation function)  $h(\cdot)$  定义，该激活函数将扩展向量  $x$  映射到输出  $a$ 。系数  $w_0$  被称为偏置 (bias)。
- **激活函数是非线性的 (The activation function is non-linear)。**
- **公式:**  $a = h(wTx)$

### 3.3 层 (Layers)

- **定义:** 层是使用相同输入的感知机的集合 (a collection of perceptrons that use the same input)。层中的每个感知机产生一个单独的输出 (Each perceptron within a layer produces a separate output)。
- **权重数量:** 由  $L$  个感知机和  $K$  个输入组成的层具有  $L \times (K + 1)$  个权重。

### 3.4 架构 (The architecture)

- **定义:** 神经网络的架构描述了层是如何连接的 (describes how layers are connected)。
- **组成:**
  - **输入层 (input layer):** 预测变量向量。
  - **隐藏层 (hidden layers):** 产生内部特征。
  - **输出层 (output layer):** 产生预测。
- 向量  $\hat{y} = [\hat{y}_1, \dots, \hat{y}_M]^T = h_W(x)$  是神经网络的输出。(注意：这里的  $\hat{y}_j$  不是第  $j$  个样本的预测值!)
- **神经网络的深度 (depth):** 层的数量决定了神经网络的深度 (hence the distinction between shallow and deep neural networks)。

## 4. 感知机的功能 (What can perceptrons do?)

### 4.1 作为线性分类器的感知机 (The perceptron as a linear classifier)

- 使用阶跃激活函数 (step activation function) 的感知机实际上是一个线性分类器 (linear classifier)。
- 使用逻辑函数 (logistic function) 的感知机是一个逻辑回归分类器。
- 在 3D 预测变量空间中，具有阶跃激活函数的感知机通过一个平面分隔两个决策区域!
- **阶跃函数公式:**  $h_{step}(d) = \{1, \text{ if } d > 0; 0, \text{ otherwise}\}$

### 4.2 作为基本逻辑函数的感知机 (The perceptron as a basic logical function)

- 我们可以使用感知机来实现逻辑函数 (logical functions)。
- 例如，可以实现逻辑与 (AND) 和逻辑或 (OR) 功能。

4.3 增加深度以推导新的逻辑函数 (Adding depth to derive new logical functions)

- 通过组合多个感知机 (增加神经网络的深度), 可以实现更复杂的逻辑函数, 例如异或 (XOR)。

4.4 作为网格模式检测器的感知机 (The perceptron as a grid pattern detector)

- 感知机可以被用来检测网格中的特定模式, 例如垂直边缘或水平边缘。

4.5 结合线性分类器和逻辑函数 (Combining linear classifiers and logical functions)

- 可以通过组合线性分类器和逻辑函数来构建更复杂的神经网络分类器。

5. 作为机器学习模型的神经网络 (Neural networks as machine learning models)

5.1 机器学习流程：回顾 (Machine learning pipelines: Reminder)

- 机器学习流程是一系列数据操作 (a sequence of data operations)。
- 一个简单的流程包括一个转换阶段 (transformation stage), 然后是一个机器学习模型:
  - 转换阶段产生派生特征 (derived features)。
  - 模型使用派生特征作为输入。
  - 转换阶段也可以被训练。

5.2 作为可调优流程的神经网络 (Neural networks as tunable pipelines)

- 神经网络可以被视为一个完整的可调优机器学习流程 (an entire tunable machine learning pipeline), 其中:
  - 每个感知机使用其他特征 (原始的或派生的) 定义一个派生特征或概念 (derived feature or concept)。
  - 增加一层中感知机的数量 (Increasing the number of perceptrons) 允许每层创建更多的新概念。
  - 增加层数 (更深的网络) (Increasing the number of layers) 允许创建越来越复杂的概念。

5.3 训练神经网络：代价函数 (Training neural networks: Cost function)

- 每个机器学习算法都需要一个模型 (model)、一个代价函数 (cost function) 和一个优化方法 (optimisation method)。
- 对于二分类问题, 一个常用的代价函数是负对数似然函数 (negative log-likelihood function):
  - $l(W) = - (1/N) * \sum (y_i * \log[\hat{y}_i] + (1 - y_i) * \log[1 - \hat{y}_i])$  (其中  $\hat{y}_i = hw(x_i)$ )
  - 此代价函数可以扩展到多类分类器。

5.4 梯度下降和反向传播 (Gradient descent and back-propagation)

- 梯度下降 (Gradient descent)** 是找到代价函数  $l(W)$  的最优系数集  $W$  的首选方法。
- 反向传播 (Back-propagation)** 是一种计算梯度的有效算法 (an efficient algorithm to compute the gradient)。然后, 优化算法使用该梯度来更新  $W$ 。
- 反向传播利用了微积分的链式法则 (chain rule of calculus)。为了计算一层中的梯度, 我们只需要来自下一层的信息。反向传播从输出开始: 它获取代价并向后计算隐藏单元的梯度。

5.5 训练注意事项 (Training considerations)

- 初始化 (Initialisation):** 如果初始权重为零, 则反向传播失败。初始权重值应该是随机的 (random)。
- 过拟合 (Overfitting):** 神经网络可以有数百万个参数。使用正则化 (regularisation) 和基于验证的提前停止 (validation-based early stop) 来避免过拟合。
- 非凸性 (Non-convexity):** 代价函数有多个局部最小值。从不同的随机起始值重新训练。
- 输入的缩放 (Scaling of the inputs):** 输入值的范围会影响权重的。进行标准化 (Standardise) 以确保输入得到平等对待。
- 架构 (Architecture):** 不同的架构适合不同的问题。

5.6 迁移学习 (Transfer learning)

- 为一个问题 A 成功训练的神经网络可以重新用于相关的问题 B, 例如:

- 我们可以保持早期阶段不变 (unchanged)，成为一个固定的转换阶段  $T(x)$ 。
  - 我们可以使用新数据  $f(z)$  重新训练 (retrain) 后期阶段。
- 本质上，我们是在迁移一个已经学习的转换 (transferring an already learnt transformation) 并将其重新用于不同的问题:
  - 无需训练  $T(x)$  (问题 A 和 B 的参数相同)。
  - 问题 B 的  $f(z)$  的最佳参数将接近于为问题 A 找到的参数 (更短的训练时间!)

## 6. 层的类型 (Types of layers)

### 6.1 全连接层 (Fully-connected layer)

- **定义:** 到目前为止，我们已经考虑了全连接 (fully-connected, FC) 层，其中所有感知机都接收来自前一层的所有输出。
- **缺点:** FC 层具有大量的参数 (large number of parameters)，训练它们可能具有挑战性。

### 6.2 网格数据中的等方差 (Equivariance in grid data)

- 图像和时间序列是由与定义空间关系的规则网格 (regular grid) 相关联的各个属性组成的复杂数据类型。
- 一些网格数据表现出等方差 (equivariance) 特性，根据该特性，可以在网格的不同位置预期相同的模式。

### 6.3 卷积层 (Convolutional layers)

- 卷积层施加了额外的限制:
  - 感知机被排列成一个称为特征图 (feature map) 的网格。
  - 关注输入网格中不同的有限区域 (limited regions)。
  - 共享它们的参数 (share their parameters)，表示为称为内核 (kernel) 的网格。
- 特征图被有效地计算为内核和输入的卷积 (convolution)，或者换句话说，用内核过滤输入 (filtering the input with the kernel)。
- 卷积层可以有几个特征图 (several feature maps)，每个特征图都与一个不同的概念 (different concept) 相关联。它们形成一个特征图堆栈 (stack)。
- **内核的维度 (dimensions of a kernel):**  $H \times W \times D$ ，其中  $H$  是高度， $W$  是宽度， $D$  是深度 (输入特征图的数量)。
- **每个内核的权重总数 (weights per kernel):**  $H \times W \times D + 1$  (偏置)。
- **训练 (Training):** 训练卷积层意味着使用数据来调整每个内核的权重。

### 6.4 池化层 (Pooling layers)

- 池化 (Pooling) 减小特征图的大小 (reduces the size of feature maps)。池化层由它们正在减少为单个数字的区域的大小定义，并插入到连续的卷积层之间。
- **类型:**
  - **最大池化 (Max pooling):** 输出是过滤器区域内的最大值。
  - **平均池化 (Average pooling):** 输出是过滤器区域内值的平均值。
- **注意:** 池化层不需要训练!

## 7. 深度学习架构 (Deep learning architectures)

- 深度神经网络不是任意层 (arbitrary layers) 的任意序列 (arbitrary sequences)。相反，它们具有适合特定目标的预定义架构 (predefined architecture)。
- 在分类中，常见的架构具有以下特点:
  - 第一层定义了一些简单的 (few, simple) 概念，最后一层定义了许多复杂的 (many, complex) 概念。
  - 特征图在我们深入网络时会收缩 (shrink)。
- 相同的中间概念可以用于不同的目标。我们可以使用迁移学习 (transfer learning) 来重用现有的解决方案。

### 7.1 VGG16 网络示例 (Example: The VGG16 network)

- VGG16 是为 ImageNet 大规模视觉识别挑战赛 (ImageNet Large-Scale Visual Recognition Challenge) 设计的 (属于 1000 个类别的图像数据集)。
- VGG16 有 16 层，3x3 内核和 1.38 亿个权重。

# 8. 总结 (Summary)

## 8.1 神经网络： 三种视角 (Neural networks: the three views)

- **功能的 (Functional):** 将输入  $x$  映射到输出  $\hat{y}$  的系统。通过调整其参数集  $w$ ，我们改变了映射。
- **认知的 (Cognitive):** 从原始数据和其他概念中创建新概念 (new concepts) 的系统。通过调整  $w$ ，我们创建了不同的概念。
- **计算的 (Computational):** 由称为感知机的互连计算单元组成的流程 (pipeline)。通过调整  $w$ ，我们改变了计算。

## 8.2 神经网络： 一个机器学习模型家族 (Neural networks: A family of machine learning models)

- 不应将神经网络视为与逻辑回归分类器等相同的机器学习模型。
- 神经网络是一个机器学习模型家族 (a family of machine learning models)。
- 每个神经网络都有一个架构 (architecture)，其中最简单的是单个感知机。
- 一些架构与其他机器学习模型没有本质上的不同 (not substantially different) (例如，感知机是一个线性分类器)。
- 我们可以将神经网络视为创建新模型的框架 (framework)。
- 我们训练一个特定的架构，并且可以使用验证 (validation) 方法来选择正确的架构。
- 我们应该避免说神经网络在给定问题上表现良好。相反，我们应该说这种神经网络架构表现良好。

## 8.3 神经网络： 通用机器 (Neural networks: Universal machines)

- 对于任何问题，我们都可以找到一个解决它的神经网络架构。从这个意义上说，神经网络被称为通用机器 (universal machines)。
- **注意:** 这并不意味着特定的神经网络架构可以解决任何问题。
- 合适的神经网络架构的存在并不意味着我们将能够找到它。
- 灵活性是通过增加复杂性来实现的 (adding complexity)，这增加了过拟合的风险。
- **神经网络专家 (Neural network experts)** 为问题设计正确的架构，并使用迁移学习的原理重用现有的解决方案。
- **神经网络蛮力使用者 (Neural network brutes)** 没有意识到猴子定理 (Monkey Theorem)，并使用所有可用的计算能力和数据来训练尽可能多的复杂架构。他们的碳足迹 (carbon footprint) 巨大。