TASK0 机器学习入门

监督学习和无监督学习的区别

1. 在定义上的区别：

- **监督学习**：
  - 是一种利用有标记的数据信息进行学习的方法。
  - 在监督学习中，训练的数据包含输入特征和对应的输出标签。模型的目标是学习从输入特征到输出标签的映射关系，利用所学习到的关系对新的输入数据预测它的输出标签。例如，在房价预测问题中，输入特征可以是房屋的面积、卧室数量、地理位置等特征数据，输出标签是房屋的价格。监督学习算法通过分析大量已知房价的房屋特征数据，学习如何根据房屋特征预测房价。
- **无监督学习**：
  - 是一种利用无标记的数据信息进行学习的方法。
  - 在无监督学习中，训练的数据只包含输入特征，没有对应的输出标签。模型的目标是从数据中发现潜在的模式、结构或规律。常见的无监督学习任务包括聚类、降维和关联规则挖掘等。例如，在客户分群问题中，可以使用无监督学习算法对客户的购买行为数据进行聚类分析，将客户分为不同的群体，每个群体具有相似的购买行为模式。

2. 在数据需求上的区别：

- **监督学习**：
  - 需要有标记的数据，每个输入样本都有对应的正确输出标签，并将数据分为训练集和测试集。
  - 获取有标记的数据通常需要耗费大量的人力和时间，因为需要人工对数据进行标记。
  - 数据量相对较少时，监督学习可能会面临过拟合的问题，即模型在训练数据上表现很好，但在新的数据上表现不佳。因此，通常需要足够多的有标记数据来训练一个有效的监督学习模型。并在训练后用测试集进行测试。
- **无监督学习**：
  - 只需要无标记的数据。
  - 数据的获取相对容易。
  - 无监督学习可以利用大量的无标记数据来发现数据中的潜在模式，结构和规律。无监督学习算法通常对数据量的要求较高，因为更多的数据可以提供更丰富的信息，有助于发现更准确的模式，结构和数据。

3. 在应用上的区别：

- **监督学习**：
  - 常用于有明确输出的问题，如图像分类，股票价格预测，欺诈检测；
- **无监督学习**：
  - 常用于数据探索，数据预处理和特征提取等问题，如客户分群，数据降维，异常检测。

4. 在模型评估上的区别：

- **监督学习**：
  - 可以使用明确的评估指标来衡量模型的性能，如准确率、精确率、召回率、F1 值等。例如，在分类问题中，可以使用准确率来衡量模型正确分类的样本比例；在回归问题中，可以使用均方误差来衡量模型预测值与真实值之间的差异。
- **无监督学习**：

- 评估无监督学习模型的性能相对较困难，因为没有明确的输出标签可以进行比较。通常使用一些间接的评估方法，如可视化结果、内部评估指标或与后续任务的结合效果等。例如，在聚类问题中，可以通过观察聚类结果的可视化图来判断聚类的质量；或者使用内部评估指标如轮廓系数来衡量聚类的紧凑性和分离性。

机器学习和深度学习的区别

1. 在定义和基本原理上的区别：

- **机器学习**：
  - 是一种数据分析技术，是计算机能够在没有明确编程的情况下进行数据分析和结果预测；
  - 依赖于算法和统计模型，例如决策树、支持向量机、随机森林等。这些模型通过对已知数据的学习，建立输入数据和输出结果之间的映射关系。例如，在垃圾邮件分类问题中，机器学习算法可以通过分析大量已人为标记的垃圾邮件和正常邮件的不同特征，比如邮件标题、发件人、关键词等，来学习如何区分垃圾邮件和正常邮件。之后，能够通过已经学习到的特征，判断不带标签的邮件是否为垃圾邮件。
- **深度学习**：
  - 是机器学习里的一个分支，通过构建具有多个层次的神经网络模型来处理，解释和分类数据；
  - 深度神经网络通常由多个神经元层组成，包括输入层、隐藏层和输出层。每一层的神经元通过权重和激活函数与前一层和后一层的神经元相连。通过调整这些权重，网络可以学习到数据中的特征和模式。例如，在图像识别任务中，深度学习模型可以自动学习图像中的低级特征（如边缘、纹理等）和高级特征（如物体形状、面部特征等），从而实现对图像的准确分类。

2. 在数据需求上的区别：

- **机器学习**：
  - 通常只需要相对较少的数据量就可以进行有效的学习和训练。对于一些简单的问题，几百或几千个样本可能就足够了。在数据量有限的情况下，机器学习算法可以通过特征工程提取数据中有用的信息，从而提高模型的性能。
- **深度学习**：
  - 一般需要大量的数据进行学习和训练。深度神经网络具有大量的参数，需要大量的数据来进行训练，以避免过拟合。

3. 在应用上的区别：

- **机器学习**：
  - 广泛适用于各种中小型规模的问题，比如垃圾邮件分类、信用评估、医疗诊断等。在这些问题中，数据量相对较小，模型的可解释性和计算效率比较高。也可以用于特征工程，为深度学习等更复杂的模型提供有效的输入特征。
- **深度学习**：
  - 适用于具有大量数据的大规模问题，如图像识别、语音识别、自然语言处理等领域。深度学习模型能够自动学习数据中的复杂特征，从而实现高精度的分类和预测。例如，在图像识别领域，深度学习模型已经达到了甚至超过人类的水平，可以准确地识别各种物体、场景和面部特征。在语音识别领域，深度学习模型可以将语音信号转换为文本，实现高效的语音交互。在自然语言处理领域，深度学习模型可以进行文本分类、机器翻译、情感分析等任务。

偏导数、链式法则、梯度、矩阵等数学概念在机器学习中的作用

1. **偏导数**：

- 特征重要性分析：
  - 对于输入的特征数据，偏导数可以帮助我们了解每个特征对模型输出的影响程度。如果某个特征的偏导数较大，说明该特征对模型输出的影响较大，可能是比较重要的特征；反之，如果偏导数较小，说明该特征对模型输出的影响较小，可能不是那么重要，可以考虑将其去除或降低其权重。
- 参数优化：
  - 一些优化算法在计算时会以来偏导数计算梯度，以便确定参数的更新方向和幅度。例如，梯度下降算法，是沿着损失函数的负梯度方向更新模型参数的算法，以最小化损失函数。偏导数在此过程中用于计算损失函数对每个模型参数的梯度，通过不断迭代计算偏导数并更新参数，模型可以逐渐拟合训练数据，提高预测准确性。

2. **链式法则**：

- 反向传播算法：
  - 反向传播是训练神经网络的核心算法之一，它依赖于链式法则。在神经网络中，前向传播过程计算模型的输出，然后根据输出和真实标签计算损失函数。在反向传播过程中，利用链式法则从输出层开始，逐层计算损失函数对每个神经元的权重和偏导数，然后根据这些导数更新参数。这样可以有效地训练神经网络，使其能够学习到输入数据和输出标签之间的复杂关系。

3. **梯度**：

- 参数更新：
  - 梯度是一个向量，它包含了函数对各个变量的偏导数。在机器学习中，梯度指示了损失函数在参数空间中的变化方向。我们可以沿着梯度的反方向更新模型的参数，以减小损失函数的值。这就是梯度下降等优化算法的基本思想。通过不断地迭代更新参数，模型可以逐渐收敛到一个较好的解，使得损失函数最小化。
- 反向传播：
  - 如前所述，反向传播算法依赖于梯度的计算。在神经网络中，通过反向传播可以高效地计算损失函数对于每个参数的梯度，从而实现对参数的更新。
- 模型评估：
  - 梯度可以用于评估模型的性能。在验证集或测试集上计算损失函数的梯度，可以了解模型是否已经收敛。如果梯度趋近于零，说明模型在当前参数下已经接近最优解；如果梯度仍然较大，说明模型还有进一步优化的空间。
- 特征选择和正则化：
  - 梯度可以帮助我们评估特征对于模型的贡献。对于线性模型，可以通过特征的系数（对应于梯度）的大小来衡量特征的重要性，从而进行特征选择。此外，梯度还可以用于正则化方法的实现。通过在损失函数中引入正则化项，并根据梯度信息调整参数，可以控制模型的复杂度，防止过拟合。

4. **矩阵**：

- 数据表示：
  - 矩阵可以用来表示和处理高维数据，如图像、文本和音频等。例如，一张灰度图像可以表示为一个二维矩阵，其中每个元素代表图像中对应像素的灰度值；一段文本可以通过词向量表示为一个

矩阵，其中每行代表一个单词的向量表示。通过矩阵的形式，我们可以方便地对数据进行存储、操作和分析。

- 模型构建：
  - 许多机器学习模型都涉及到矩阵的运算。例如，线性回归模型中的参数估计可以通过矩阵乘法来实现；支持向量机中的核函数可以表示为矩阵运算；神经网络中的前向传播和反向传播过程也大量使用了矩阵乘法和加法等运算。矩阵运算的高效性和线性性质使得机器学习模型能够处理大规模的数据和复杂的计算。
- 优化和迭代：
  - 机器学习模型的训练过程通常需要通过迭代来优化参数。在这个过程中，涉及到矩阵的求导、求逆、求解线性方程组等操作。例如，在使用梯度下降等优化算法时，需要计算损失函数对参数的梯度，这可能涉及到矩阵的求导；在求解线性回归等模型的参数时，可能需要求解线性方程组，这也涉及到矩阵的相关运算。

常见的激活函数

在神经网络中，输入经过权值加权计算并求和之后，需要经过一个函数的作用才继续向下一层传输，这个函数就是激活函数（可分为饱和激活函数和非饱和激活函数）。而如果不在神经网络中输入激活函数，则在这个网络中，每一层往下一层输入的都是线性数据，只能用来解决一些线性问题。输入激活函数可以增加一些非线性因素，使神经网络可以解决非线性的问题。

1. **饱和激活函数**：
   指在输入值趋向于正无穷或负无穷时，函数的导数趋近于0的函数。这类函数的特点是输出值的范围有界。

- **Sigmoid**：

$$Sigmoid(x) = \frac{1}{1+e^{-x}}$$

  - 输出范围为0~1之间
  - 优点：
    1. 可以用来对数据进行归一化；
    2. 也可以用于输出是预测概率的模型；
  - 缺点：
    1. 当输入非常大或非常小时，输出的变化非常小，导致梯度为0；
    2. 输出不是0均值，进而导致后一层神经元将得到上一层输出的非0均值的信号作为输入。随着网络的加深，会改变原始数据的分布趋势；
    3. 幂运算耗时较长。
- **Tanh**：

$$Tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

  - 输出范围为-1~1之间
  - 优点：
    1. Tanh函数在原点附近与y=x函数形式相近，当输入的激活值较低时，可以直接进行矩阵运算，训练相对容易；
    2. 解决了Sigmoid函数输出不是0均值的问题；
    3. Tanh函数是一个奇函数，图像关于原点对称，使它能够较好地处理一些对称分布的数据。
  - 缺点：

    1. 梯度消失问题仍然存在；
    2. 幂运算耗时较长

1. **非饱和激活函数**：
   在输入值很大或很小的时候，仍然能够保持一定的梯度，有助于解决梯度消失的问题，加快网络的收敛速度。

- **ReLU**：

$$ReLU(x) = max(0, x)$$

  - 优点：
    1. 在正区间上解决了梯度消失的问题；
    2. 计算速度非常快，只需要判断输入值是否大于0；
    3. ReLU输出使部分值为0，带来了网络稀疏性。
  - 缺点：
    1. 输出不是0均值；
    2. Dead ReLU Problem，指的是某些神经元可能永远都不会被激活，导致相应的参数永远不能被更新。
- **ReLU6**：

$$ReLU(x) = min(max(0, x), 6)$$

  - ReLU6就是普通的ReLU但是限制最大输出为6。
  - 优点：
    1. ReLU6具有ReLU的优点；
    2. 使用float16/int8低精度的时候也能良好工作。
  - 缺点：
    1. 与ReLU缺点类似。
- **PReLU**：

$$PReLU(x) = \{ \alpha x \quad \text{if } x < 0 \; x \quad \text{if } x \geq 0$$

  - 参数α通常为0到1之间的数字，并且通常相对较小。这个参数使PReLU能够在输入值为负数时调整输出，而不是像ReLU那样直接输出0，使PReLU在处理负输入时比ReLU更加灵活和有效。
  - 优点：
    1. 能有效缓解梯度消失问题；
    2. 参数α是可学习的，提高了模型的灵活性。
  - 缺点：
    1. 由于PReLU函数引入了额外的可学习参数，模型的自由度增加，这使得模型更容易过拟合训练数据。
- **Leaky ReLU**：

$$f(x) = max(0.01x, x)$$

  - 优点：
    1. 调整了ReLU中负值的零梯度问题；
    2. 扩大了ReLU的函数范围。
  - 注意：从理论上讲，Leaky ReLU具有ReLU的所有优点，但在实际操作中，并没有完全证明Leaky ReLU总是比ReLU更好。

- **ELU**：

$$ELU(x) = \{ x \quad \text{if x}>0 \ \alpha(e^x - 1) \quad \text{if x}\leq 0$$

  - 优点：
    1. 具有ReLU所有的优点；
    2. 且没有Dead ReLU Problem；
    3. 正常梯度更接近单位自然梯度。
  - 缺点
    1. 计算强度较高，含有幂运算；
    2. 在实践中同样没有比Relu更突出的效果。
- **Swish**：

$$Swish(x) = x * Sigmoid(x)$$

  - 优点：
    1. Swish具有一定ReLU函数的优点；
    2. Swish具有一定Sigmoid函数的优点；
    3. 可以看做是介于线性函数与ReLU函数之间的平滑函数，能在优化和泛化中起重要作用。
  - 缺点：
    1. 运算复杂，速度较慢。
- **Maxout**：

$$Maxout(x) = max(\omega_i x + b_i)$$

  - 优点：
    1. Maxout的拟合能力非常强，可以拟合任意的凸函数;
    2. Maxout具有ReLU的所有优点，线性、不饱和性;
    3. 能缓解梯度消失，同时又规避了ReLU函数神经元死亡的问题。
  - 缺点：
    1. 增加了参数和计算量。

神经网络的基本结构

1. **神经元**：

- 一个包含输入，输出和计算功能的模型。输入可以类比为神经元的树突，而输出可以类比为神经元的轴突，计算则可以类比为细胞核。

2. **输入层**：

- 功能：
  - 这是神经网络的起始层，负责接收原始数据。这些数据是神经网络的输入信号，其格式和类型取决于具体的任务。例如，在图像识别任务中，输入层接收的是图像的像素值；在文本处理任务中，输入的可能是文本的词向量或字符编码等。
- 特点：
  - 输入层的神经元数量通常由输入数据的特征数量决定。每个神经元对应一个输入特征，将其值传递给下一层（通常是隐藏层），并且输入层的神经元只负责将数据传递出去，不进行复杂的计算操作。

3. **隐藏层**：

- 功能：
  - 隐藏层位于输入层和输出层之间，是神经网络进行特征提取和模式识别的关键部分。它对输入数据进行非线性变换和复杂的计算，以提取出更高级、更抽象的特征表示。这些特征对于解决具体的任务非常重要，例如识别图像中的物体、预测时间序列数据的趋势等。
- 结构：
  - 隐藏层可以有一层或多层，层数越多，神经网络的表达能力越强，但同时也会增加训练的难度和计算成本。每一层隐藏层都包含多个神经元，神经元之间通过连接权重相互连接。这些连接权重是神经网络学习的参数，通过不断调整这些权重，神经网络可以逐渐优化对输入数据的处理和特征提取能力。

4. **输出层**：

- 功能：
  - 输出层是神经网络的最后一层，其作用是将隐藏层提取的特征转换为最终的输出结果。输出结果的形式取决于具体的任务，例如在分类任务中，输出层可能输出每个类别的概率分布，以确定输入数据属于哪个类别；在回归任务中，输出层可能输出一个连续的值，用于预测某个数值。
- 特点：
  - 输出层的神经元数量通常由任务的输出维度决定。例如，对于二分类问题，输出层可以只有一个神经元，其输出值表示输入数据属于正类或负类的概率；对于多分类问题，输出层的神经元数量则等于类别数，每个神经元的输出值表示输入数据属于该类别的概率。
- 连接权重和偏置：
  - 连接权重：连接神经网络各层神经元之间的参数，它决定了输入信号在神经元之间传递的强度和方向。权重的值可以是正数、负数或零，正数表示加强信号传递，负数表示减弱信号传递，零表示不传递信号。在训练过程中，神经网络通过优化算法不断调整连接权重，以提高模型的性能和准确性。
  - 偏置：偏置是每个神经元的一个额外参数，它类似于线性回归中的截距项。偏置的值与输入数据无关，通常是一个常数。偏置的作用是在神经元的输入信号加权求和后，再加上偏置值，然后通过激活函数得到神经元的输出。偏置可以帮助神经网络更好地拟合数据，特别是在处理非线性问题时，偏置可以使神经网络的决策边界更加灵活。

5. **激活函数**：

- 功能：激活函数应用于神经元的输出，对神经元的输出进行非线性变换，使神经网络能够学习和表示复杂的非线性关系。如果没有激活函数，神经网络将只能处理线性问题，其表达能力将受到极大的限制。

机器学习中的数据处理

1. **定义**：

- 数据处理是一个较为广泛的概念，涵盖了机器学习中与数据相关的所有操作。包括数据的收集，传输，存储，管理，分析和对数据进行变换以适应不同的模型训练和评估的操作。（数据预处理是数据处理中的一个重要环节）

2. **内容**：

- **数据采集**：通过传感器、监测设备、物联网等手段，采集来自实际物体或系统的数据。这些数据可以是温度、压力、振动、电流等物理量的测量值，也可以是图像、视频等感知数据。
  - 常见方法：

    1. 随机采样：从数据集中随机抽取一部分样本；
    2. 分层采样：将数据集按照不同的特征进行分层，从每个层次中独立地抽取部分样本，可以确保不同特征都有足够的样本代表；
    3. 过采样和欠采样：当数据集中的样本分类不平衡时，过采样可以增加少数类的样本数量，欠采样可以减少多数类的样本数量。
- **数据传输**：将采集到的数据按照一定格式传输到中心服务器或云平台进行存储和处理。传输可以通过有线网络、无线网络或蜂窝通信等方式实现。
- **数据预处理**：对采集到的原始数据进行清洗和处理，去除噪声、异常值和重复数据，确保数据的准确性和一致性。
    - 原因：
        1. 通常来说，我们得到的原始数据往往混乱且不全面，机器学习模型无法从中识别并提取有效信息；或数据量、数据维度过大，存在冗余，模型无法有效学习，也会影响速度。这时需要进行数据清洗，数据转换，数据压缩等行为，将其处理为机器学习算法能直接处理的有意义数据；
    - 内容：
        1. 数据清洗：处理缺失值，可采用删除、插值等方法；去除异常值，通过统计方法或领域知识判断异常值并进行处理。
        2. 数据集成：将多个来源的数据合并在一起，处理数据格式不一致、重复数据等问题。
        3. 数据变换：进行标准化、归一化，使不同特征具有相似的尺度；进行特征编码，将类别型特征转换为数值型特征。
        4. 特征选择：从原始特征集中选择对模型性能贡献较大的特征，去除冗余和不相关特征。
- **数据存储**：将清洗后的数据存储到数据库、数据湖或其他存储系统中。选择合适的数据存储技术和架构可以确保数据的可靠性、可扩展性和安全性。
- **数据分析**：对存储的数据进行分析和处理，提取有价值的信息和模式。
- **数据可视化**：将分析结果以可视化的形式展示，通常使用图表、图像、仪表盘等方式展示数据和分析的结果。数据可视化有助于用户理解和解释数据。
- **建立信息库**：方便之后使用数据。

3. **目的**：

- 数据处理可以提高数据的质量；
- 数据处理可以增强数据的可用性。数据集成将来自不同数据源的数据集成到一起，为模型提供更全面的信息，数据变换可以以增强数据的表达能力，使模型能够更好地学习数据中的模式和关系；
- 数据处理可以提升模型性能。合理的数据采样可以提高模型对不同类别的泛化能力，还可以减少模型的训练时间和计算资源需求，提高模型的训练效率。提高模型的泛化能力，防止模型过拟合。

过拟合和欠拟合

1. **定义**

- **过拟合**：
    - 过拟合是指模型在训练数据上表现得非常好，但在测试数据或新数据上表现不佳的情况。模型过度学习了训练数据中的细节和噪声，以至于它不能很好地泛化到未见过的数据。就像一个学生死记硬背了课本上的例题答案，但是当遇到稍微变化一点的题目时，就无法正确解答。在机器学习中，这是因为模型过于复杂，参数过多，导致它记住了训练数据中的特殊情况，而没有真正学习到数据背后的一般规律。
- **欠拟合**：

- 欠拟合则是指模型没有很好地捕捉到数据的特征和规律，无论是在训练数据还是测试数据上都表现不好。模型过于简单，无法拟合数据的复杂性。这就好比一个学生没有掌握课程的基本概念，对所有的题目（无论是课本上的还是考试中的新题目）都无法很好地回答。在机器学习中，可能是模型的复杂度不够，比如特征数量不足或者模型的参数太少，导致它不能很好地描述数据的内在结构。

2. **产生原因**

- **过拟合产生的原因**：
  - **模型复杂度高**：如果模型有过多的参数或者过于复杂的结构，如深度神经网络中神经元数量过多、层数过深，它就有足够的能力去拟合训练数据中的每一个细节，包括噪声，而这些模式可能并不是图像类别的本质特征。
  - **训练数据量小**：当训练数据有限时，模型更容易过拟合。因为模型可以轻松地学习到这些少量数据中的所有细节。
  - **训练时间过长或迭代次数过多**：在训练过程中，如果让模型持续训练，不断地调整参数以减小训练误差，最终可能会导致模型过拟合。
- **欠拟合产生的原因**：
  - **模型复杂度低**：模型太简单，无法学习到数据中的复杂关系。例如，用一个简单的线性回归模型来拟合一个包含二次函数关系的数据，它只能捕捉到线性部分，而无法拟合数据中的曲线部分。
  - **特征选择不当**：如果没有选择足够相关的特征，模型可能无法学习到数据的真正规律。例如，在预测房屋价格时，如果只考虑房屋的面积，而忽略了房屋的房龄、周边配套设施等重要因素，模型就很难准确地拟合数据。
  - **数据质量差**：数据存在大量错误、缺失值或者噪声过大，也可能导致模型欠拟合。

3. **解决方法**

- **过拟合的解决方法**：
  - **增加训练数据**：收集更多的数据可以让模型学习到更广泛的规律，减少对特定数据细节的过度拟合。
  - **正则化**：这是一种常用的方法，通过在损失函数中添加惩罚项来限制模型参数的大小。
  - **早停止**：在训练过程中，监测模型在验证集上的性能。当验证集性能不再提高或者开始下降时，停止训练。例如，在训练一个深度学习模型时，每经过一定的训练轮次，就计算模型在验证集上的准确率。如果准确率连续几个轮次都没有提高，就停止训练，这样可以防止模型过度拟合训练数据。
  - **模型简化**：减少模型的复杂度，如减少神经网络的层数或神经元数量。在一些简单的任务中，如果一个神经网络有过多的层，可以适当减少层数，使模型能够更好地泛化。
- **欠拟合的解决方法**：
  - **增加模型复杂度**：可以使用更复杂的模型，如将线性模型改为多项式模型，或者在神经网络中增加层数或神经元数量。
  - **特征工程**：选择更合适的特征或者对现有特征进行变换。例如，在预测用户购买行为时，可以对用户的年龄、收入、购买历史等特征进行组合、归一化等操作，或者添加新的衍生特征，如用户购买频率、最近一次购买时间间隔等，以帮助模型更好地拟合数据。
  - **清洗数据**：处理数据中的错误、缺失值和噪声。对于缺失值，可以采用均值填充、中位数填充或者使用机器学习算法来预测缺失值。对于噪声数据，可以采用滤波、平滑等数据预处理技术，提高数据质量，使模型能够更好地学习数据中的规律。