引言

监督学习：

定义：监督学习是一种通过带标签的数据进行训练的机器学习方法。所谓带标签的数据，就是每个输入数据都有一个对应的正确输出，模型通过学习这些输入输出对来进行预测。

特点：

1. **训练数据包含输入和输出**：训练数据由输入特征和对应的标签（输出）组成。
2. **目标是预测输出**：模型的目标是学会从输入特征中预测正确的输出。

常见算法：

- 回归（Regression）：预测连续值，例如房价预测。
- 分类（Classification）：预测离散值，例如垃圾邮件分类。

无监督学习：

定义：无监督学习是一种通过未标注的数据进行训练的机器学习方法。没有预先定义的标签，模型需要自己找出数据中的模式或结构。

特点：

1. **训练数据只有输入没有输出**：训练数据只有输入特征，没有对应的标签。
2. **目标是发现数据中的模式**：模型的目标是从数据中找到隐藏的模式或结构。

常见算法：

- 聚类（Clustering）：将数据分组，例如K-means聚类。
- 降维（Dimensionality Reduction）：减少数据的维度，例如PCA（主成分分析）。

1.单变量线性回归

1.1 代价函数

单变量线性回归的代价函数（也称为损失函数）用于评估模型预测结果与实际结果之间的差距。对于单变量线性回归，假设我们有训练数据集 $(x^{(i)}, y^{(i)})$（其中 (i) 表示第 (i) 个训练样本），我们的模型是：
$h_\theta(x) = \theta_0 + \theta_1 x$
单变量线性回归的代价函数（均方误差）定义为：

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

其中：
-m 是训练样本的数量。
-$h_\theta(x^{(i)})$是第 i个样本的预测值，即 $\theta_0 + \theta_1 x^{(i)}$。
-$y^{(i)}$ 是第 i个样本的实际值。

1.2 梯度下降

梯度下降的核心思想是沿着代价函数的梯度方向更新参数，使得每次迭代都朝着代价函数值减小的方向移动，从而逐步逼近最小值。

具体步骤

1. **初始化参数**：选择初始参数值（通常为零或随机小值）。
2. **计算代价函数的梯度**：对于单变量线性回归，代价函数 J(θ) 是参数 θ 的函数。我们需要计算 J(θ)对 θ_0和 θ_1 的偏导数。
3. **更新参数**：根据梯度的负方向更新参数。更新规则如下：

   $\theta_j := \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j}$

   其中：

   - θ_j表示第 j 个参数。
   - α 是学习率（learning rate），控制更新步伐的大小。
   - $\frac{\partial J(\theta)}{\partial \theta_j}$是代价函数 J(θ) 对参数 θ_j 的偏导数。

计算梯度

以单变量线性回归为例，代价函数 J(θ)J(\theta)J(θ) 的偏导数计算如下：

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \frac{\partial J(\theta)}{\partial \theta_1} = \frac{1}{m} \sum_{i=1}^{m} \left((h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)} \right)$$

其中：

- $\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$ 是模型的预测值。

更新参数

参数更新规则变为：

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left((h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)} \right)$$

迭代直到收敛

重复上述步骤，直到代价函数的值收敛到一个较小的范围内或达到预定的迭代次数。

2.多变量线性回归

2.1 代价函数

在多变量线性回归中，我们有多个输入特征 $x1, x2, \ldots, xn$。我们的目标是找到一组参数 $\theta_0, \theta_1, \ldots, \theta_n$ 来最小化代价函数。

假设我们的训练数据集有m个样本，每个样本有 n 个特征，我们的代价可以表示为:

$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$

其中假设函数（模型）可以表示为：

$\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n = \theta^T x$

2.2 梯度下降

代价函数$J(\theta)$ 对每个参数 θ_j 的偏导数计算如下：

$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} \left((h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)} \right)$

参数更新规则为：

$$\theta_0 := \theta_0 - a \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_0^{(i)}$$

$$\theta_1 := \theta_1 - a\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})\,x_1^{(i)}$$

$$\theta_2 := \theta_2 - a\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})\,x_2^{(i)}$$

$\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}\left((h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}\right)$

特征缩放

特征的尺度都尽量缩放到-1到1之间

最简单的方法是令：$x_n = \frac{x_n - \mu_n}{s_n}$，其中 μ_n是平均值，s_n是标准差。

学习率

如果学习率过小，则达到收敛所需的迭代次数会非常高；如果学习率过大，每次迭代可能不会减小代价函数，可能会越过局部最小值导致无法收敛。

通常可以考虑尝试些学习率：

$$\alpha = 0.01，\ 0.03，\ 0.1，\ 0.3，\ 1，\ 3，\ 10$$

2.3 多项式回归

对于一个单变量的情况，如果我们希望拟合一个 ddd 次多项式，我们的模型表示为：

$\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \ldots + \theta_d x^d$

对于多变量情况，如果我们有多个输入特征 $x_1, x_2, \ldots, x_n$，我们可以对每个特征进行多项式扩展。

比如一个二次方模型：$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2$

或者三次方模型：$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_3^3$

通常我们需要先观察数据然后再决定准备尝试怎样的模型。 另外，我们可以令：

$x_2 = x_2^2, x_3 = x_3^3$，从而将模型转化为线性回归模型。

根据函数图形特性，我们还可以使：

$$h_\theta(x) = \theta_0 + \theta_1(size) + \theta_2(size)^2$$

或者:

$$h_\theta(x) = \theta_0 + \theta_1(size) + \theta_2\sqrt{size}$$

如果我们采用多项式回归模型，在运行梯度下降算法前，特征缩放非常有必要。

2.4 特征方程

用于求解线性回归问题的闭式解法，它避免了迭代计算。基于最小二乘法，直接通过矩阵运算来求解最优参数。

特征方程的表示为：

$\theta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$

其中：

- $\mathbf{X}$ 是 m$\times(n+1)$ 的特征矩阵，包含了所有样本的特征（包括偏置项$x_0 = 1$）。
- $\mathbf{y}$ 是$m \times 1$的标签向量，包含了所有样本的实际值。
- θ 是 $(n+1) \times 1$ 的参数向量，包含了所有参数。

比较梯度下降和特征方程

- **梯度下降**：是一种迭代优化算法，适用于大规模数据集和复杂模型。需要选择合适的学习率，并且可能需要多次迭代才能收敛。
- **特征方程**：是一种闭式解法，适用于线性模型和相对较小的数据集。计算直接，结果精确，但在特征矩阵不可逆或数据集规模很大时计算可能不稳定或不可行。

3.逻辑回归

它通过将线性回归的输出映射到概率值来预测某个事件发生的可能性，特别适用于二分类问题。

逻辑回归模型的假设是：$h_\theta(x) = g(\theta^T\mathbf{X})$其中：$\mathbf{X}$代表特征向量 $\mathbf{g}$ 代表逻辑函数（**logistic function**)是一个常用的逻辑函数为**S**形函数（**Sigmoid function**），公式为：

$$g(z) = \frac{1}{1+e^{-z}}$$

可以表示为：

$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

3.1 代价函数

$J(\theta)$定义为：

$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]$

其中：

- m 是训练样本的数量。
- $y^{(i)}$ 是第 iii 个样本的实际标签（0 或 1）。
- $\theta(x^{(i)})$ 是第 iii 个样本的预测概率。

3.2 成本函数和梯度下降

导数计算如下：

$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}\right)$

其中：

- $h_\theta(x^{(i)}) = \frac{1}{1+e^{-\theta^T x^{(i)}}}$1 是第 iii 个样本的预测概率。
- $\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j{}^{(i)}$

4.正则化

发现了过拟合问题，应该如何处理？

1.丢弃一些不能帮助我们正确预测的特征。可以是手工选择保留哪些特征，或者使用一些模型选择的算法来帮忙（例如**PCA**）

2.正则化。 保留所有的特征，但是减少参数的大小（**magnitude**）

4.1 代价函数

防止过拟合问题的假设：$J(\theta) = \frac{1}{2m}[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2]$

如果选择的正则化参数λ过大，则会把所有的参数都最小化了，造成欠拟合。

4.2 正则化线性回归

正则化线性回归的代价函数为：

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}[((h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2)]$$

对上面的算法中 $j = 1,2,\ldots,n$ 时的更新式子进行调整可得：

$$\theta_j := \theta_j(1 - a\frac{\lambda}{m}) - a\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

$$\theta_0 := \theta_0 - a\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)}$$

可以看出，正则化线性回归的梯度下降算法的变化在于，每次都在原有算法更新规则的基础上令 θ 值减少了一个额外的值。

我们同样也可以利用正规方程来求解正则化线性回归模型，方法如下所示：

$$\theta = \left(X^T X + \lambda \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix} \right)^{-1} X^T y$$

5.神经网络

在多分类问题中，我们通常使用 Softmax 函数作为输出层的激活函数，并使用分类交叉熵（Categorical Cross-Entropy）作为代价函数。反向传播算法用于计算梯度并更新参数以最小化代价函数。对于包含多个隐藏层的神经网络，反向传播算法更加复杂，但基本原理仍然相同。

5.1 多分类神经网络的结构

假设我们有一个包含L层的神经网络（不包括输入层），第l层有n_l个神经元。我们使用以下符号：

- $a^{(l)}$：第 l 层的激活值。
- $z^{(l)}$：第 l 层的线性组合（即 $z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)}$）。
- $W^{(l)}$：第 l 层的权重矩阵。
- $b^{(l)}$：第 l 层的偏置向量。
- y：实际标签向量，使用 one-hot 编码表示。

5.2 代价函数

对于多分类问题，常用的代价函数是分类交叉熵，并且包含正则化项以防止过拟合。代价函数 $J(\theta)$ 定义为：

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} y_k^{(i)} \log(h_\theta(x^{(i)})_k) + \frac{\lambda}{2m} \sum_{l=1}^{L} \sum_{j=1}^{n_l} \sum_{k=1}^{n_{l-1}} (W_{jk}^{(l)})^2$$

其中：

- m 是训练样本的数量。
- K是输出类别的数量。
- $y_k^{(i)}$) 是第 iii 个样本的实际标签（one-hot 编码）。
- $h_\theta(x^{(i)})_k$是第 l 个样本预测为类别 K的概率。
- λ是正则化参数。

5.3 反向传播算法

反向传播算法用于计算每个参数的梯度。对于包含多个隐藏层的神经网络，反向传播过程分为以下几步：

1. **前向传播**：
   - 计算每一层的激活值 $a^{(l)}$ 和线性组合 $z^{(l)}$。
2. **计算输出层的误差项**：
   - 对于输出层（第 l 层），误差项 $\delta^{(L)}$ 计算为： $\delta^{(L)} = a^{(L)} - y$ 其中 $a^{(L)}$是输出层的激活值（即预测值）。
3. **计算隐藏层的误差项**：

- 对于第 l 层，误差项$\delta^{(l)}$ 计算为：$)\delta^{(l)} = (W^{(l+1)})^T\delta^{(l+1)} \cdot g'(z^{(l)})$其中 $g'(z^{(l)})$是第 l 层激活函数的导数。

4. **计算梯度**：
   - 对于第 l 层，梯度 $\frac{\partial J}{\partial W^{(l)}}$ 和 $\partial\frac{\partial J}{\partial b^{(l)}}$ 分别为： $\frac{\partial J}{\partial W^{(l)}} = \frac{1}{m}\delta^{(l)}(a^{(l-1)})^T + \frac{\lambda}{m}W^{(l)}$
   - $\frac{\partial J}{\partial b^{(l)}} = \frac{1}{m}\sum_{i=1}^{m}\delta^{(l)}$
5. **更新参数**：
   - 使用梯度下降或其他优化算法更新参数：$W^{(l)} := W^{(l)} - \alpha\frac{\partial J}{\partial W^{(l)}}$ $b^{(l)} := b^{(l)} - \alpha\frac{\partial J}{\partial b^{(l)}}$ 其中 α 是学习率。

5.4 向量化表示

为了提高计算效率，我们可以将反向传播的计算向量化。以下是向量化的反向传播算法步骤：

1. **前向传播**：
   - 计算每层的激活值 $a^{(l)}$ 和线性组合 $z^{(l)}$：$z^{(l)} = W^{(l)}a^{(l-1)} + b^{(l)}$ $a^{(l)} = g(z^{(l)})$
2. **输出层误差**：
   - 计算输出层的误差项： $\delta^{(L)} = a^{(L)} - y$
3. **隐藏层误差**：
   - 逐层计算隐藏层的误差项： $\delta^{(l)} = (W^{(l+1)})^T\delta^{(l+1)} \cdot g'(z^{(l)})$**梯度计算**：
   - 计算每层的梯度：$\frac{\partial J}{\partial W^{(l)}} = \frac{1}{m}\delta^{(l)}(a^{(l-1)})^T + \frac{\lambda}{m}W^{(l)}$
   - $\frac{\partial J}{\partial b^{(l)}} = \frac{1}{m}\sum_{i=1}^{m}\delta^{(l)}$
4. **参数更新**：
   - 更新每层的参数： $W^{(l)} := W^{(l)} - \alpha\frac{\partial J}{\partial W^{(l)}}$
   - $b^{(l)} := b^{(l)} - \alpha\frac{\partial J}{\partial b^{(l)}}$

###