深度学习基础

一、 训练误差和泛化误差

训练误差主要是指的型在训练数据集上表现出的误差；泛化误差主要是指的模型在任意一个测试数据样本上表现出的误差的期望。

一般情况下，由训练数据集学到的模型参数会使模型在训练数据集上的表现优于或等于在测试数据集上的表现。

1.1 模型选择

选择模型（线性回归，逻辑回归等），也可以是选择有着不同超参数的同类模型（参数不同，调参）。

1.1.1 验证数据集

可以预留一部分在训练数据集和测试数据集以外的数据来进行模型选择。这部分数据被称为验证数据集，简称验证集（validation set）。例如，我们可以从给定的训练集中随机选取一小部分作为验证集，而将剩余部分作为真正的训练集。

1.1.2 k 折交叉验证

原始训练数据被分成k个不重叠的子集。然后执行次模型训练和验证，每次在$k-1$个子集上进行训练， 并在剩余的一个子集（在该轮中没有用于训练的子集）上进行验证。 最后，通过对k次实验的结果取平均来估计训练和验证误差。

1.2 模型复杂度

时间复杂度和空间复杂度是衡量一个算法的两个重要指标,用于表示算法的最差状态所需的时间增长量和所需辅助空间.

在深度学习神经网络模型中我们也通过：

计算量/FLOPS（时间复杂度）即模型的运算次数

访存量/Bytes（空间复杂度）即模型的参数数量

1.3 欠拟合与过拟合

给定训练数据集，如果模型的复杂度过低，很容易出现欠拟合；如果模型复杂度过高，很容易出现过拟合。应对欠拟合和过拟合的一个办法是针对数据集选择合适复杂度的模型。

1.3.1 欠拟合

模型无法得到较低的训练误差，我们将这一现象称作欠拟合（underfitting）

1.3.2 过拟合

模型的训练误差远小于它在测试数据集上的误差，我们称该现象为过拟合

二、 权重衰减

权重衰减是一个正则化技术，作用是抑制模型的过拟合，以此来提高模型的泛化性。正则化是减少数据扰动对预测结果的影响。训练数据点距离真实模型的偏离程度就是数据扰动。

模型权重数值越小，模型的复杂度越低。通过增加惩罚项可以限制参数大小，抑制过拟合。可以用公式表示为：

$$L = L_0 + \frac{\lambda}{2} \|W\|^2 \tag{2 - 1}$$

式中 λ ———— 超参数

$\|\mathbb{W}\|^2$是模型参数的 2 范数的平方

L_0 是原本的损失函数

假设模型有n个参数，$W = [w_1 \ \ w_2 \ \ w_3 \ \ ..... \ \ w_n]$，L 可以表示为：

$$\begin{aligned} L &= L_0 + \frac{\lambda}{2}\left(\sqrt{w_1^2 + w_2^2 + w_3^2 + ... + w_n^2}\right)^2 \\ &= L_0 + \frac{\lambda}{2}(w_1^2 + w_2^2 + w_3^2 + ... + w_n^2) \end{aligned} \tag{2 - 2}$$

这样在 SGD 中的参数更新由$w_i \leftarrow w_i - \gamma\frac{\partial(L)}{\partial(w_i)}$ 变为

$$\begin{aligned} w_i &\leftarrow w_i - \gamma\left(\frac{\partial(L_0)}{\partial(w_i)} + \lambda w_i\right) \\ &= w_i - \gamma\lambda w_i - \gamma\frac{\partial(L_0)}{\partial(w_i)} \\ &= w_i(1 - \gamma\lambda) - \gamma\frac{\partial(L_o)}{\partial(w_i)} \end{aligned} \tag{2 - 3}$$

L_2范数正则化令权重w_1和w_2, 先自乘小于 1 的数，再减去不含惩罚项的梯度。

三、 丢弃法(倒置丢弃法)

使用丢弃法也可以应对过拟合的问题。随机丢弃一部分神经元（同时丢弃其对应的连接边）来避免过拟合。

在多层感知机中单个隐藏层单元的计算为：

$$h_i = \varphi(x_1 w_1 i + x_2 w_2 i + x_3 w_3 i + x_4 w_4 i + b_i) \tag{3 - 1}$$

当对该隐藏层使用丢弃法时，该层的隐藏单元将有一定概率被丢弃掉。设丢弃概率为p，那么有p的概率h_i会被清零，有$1-p$的概率h_i会除以$1-p$做拉伸。丢弃概率是丢弃法的超参数。可以表示为：

$$h_i' = \begin{cases} 0 & \text{if } p \\ \frac{\zeta_i}{1-p} h_i & \text{else } 1-p \end{cases} \tag{3 - 2}$$

式中 p ———— 超参数

四、 正向传播，反向传播，计算图

4.1 正向传播

正向传播（forward propagation）是指对神经网络沿着从输入层到输出层的顺序，依次计算并存储模型的中间变量（包括输出）。

4.2 反向传播

反向传播(back-propagation)指的是计算神经网络参数梯度的方法。总的来说，反向传播依据微积分中的链式法则，沿着从输出层到输入层的顺序，依次计算并存储目标函数有关神经网络各层的中间变量以及参数的梯度。

4.3 计算图

通过绘制计算图（computational graph）来可视化运算符和变量在计算中的依赖关系。

五、 数值稳定性和模型初始化

层数较多时，梯度的计算也更容易出现衰减或爆炸。每层的参数值会变的特别大或特别小。

5.1 随机初始化模型参数

如果将每个隐藏单元的参数都初始化为相等的值，那么在正向传播时每个隐藏单元将根据相同的输入计算出相同的值，并传递至输出层。在反向传播中，每个隐藏单元的参数梯度值相等。因此，这些参数在使用基于梯度的优化算法迭代后值依然相等。之后的迭代也是如此。在这种情况下，无论隐藏单元有多少，隐藏层本质上只有 1 个隐藏单元在发挥作用。因此，正如在前面的实验中所做的那样，我们通常对神经网络的模型参数，特别是权重参数，进行随机初始化。

六、 独热编码

在一些数据集中遇到的一些特征不都是数字而是分类特征，比如性别中的男女，还有国籍中的中国美国等，这就需要一种方法来将这些离散量数字化 ———— 独热编码。

One-Hot 编码，又称为一位有效编码，主要是采用 N 位状态寄存器来对 N 个状态进行编码，每个状态都由他独立的寄存器位，并且在任意时候只有一位有效。

One-Hot 编码是分类变量作为二进制向量的表示。这首先要求将分类值映射到整数值。然后，每个整数值被表示为二进制向量，除了整数的索引之外，它都是零值，它被标记为 1。

6.1 举例

Table 6 - 1: 举例 1

ID	性别
1	男
2	女

这样男就可以看为 10，女可以看为 01

Table 6 - 2: 举例 2

ID	国籍

1	中国
2	美国
3	法国

这样中国可以写为 100，美国 010，法国 001。

在进行独热编码后前面的两个例子将会变为：

Table 6 - 3: 独热编码-举例 1

ID	男	女
1	1	0
2	0	1

Table 6 - 4: 独热编码-举例 2

ID	中国	美国	法国
1	1	0	0
2	0	1	0
3	0	0	1

6.2 Python 实现独热编码

```
import pandas as pd

# 创建一个示例 DataFrame
data = {
    'color': ['red', 'blue', 'green', 'blue', 'red'],
    'size': ['S', 'M', 'L', 'XL', 'S']
}
df = pd.DataFrame(data)
print("原始 DataFrame:")
print(df)

# 对 'color' 列进行独热编码
df_color_encoded = pd.get_dummies(df['color'], prefix='color')
print("\n'color' 列的独热编码:")
print(df_color_encoded)

# 对 'size' 列进行独热编码
df_size_encoded = pd.get_dummies(df['size'], prefix='size')
print("\n'size' 列的独热编码:")
```

```
print(df_size_encoded)

# 将独热编码的列与原始 DataFrame 合并
df_encoded = pd.concat([df, df_color_encoded, df_size_encoded], axis=1)
print("\n合并后的 DataFrame:")
print(df_encoded)

# 如果需要，可以删除原始的分类列
df_encoded.drop(['color', 'size'], axis=1, inplace=True)
print("\n删除原始分类列后的 DataFrame:")
print(df_encoded)
```

七、 标签编码

标签编码也是将将分类变量转换为数值标签的一种方法。标签编码将每个类别映射到整数值，从 0 开始递增。这种方法对于具有有序关系的类别特征很有用，但它不适用于没有明显顺序的类别。

八、 数据预处理

8.1 缺失值处理

8.1.1 检查缺失值

```
missing_values_count = df.isna().sum()
print(missing_values_count)
```

8.1.2 使用平均值填充缺失值

```
mean_value = df["LotFrontage"].mean()
df_filled = df["LotFrontage"].fillna(mean_value)
df["LotFrontage"] = df_filled
```

8.2 删除无关的列

```
df.drop(['color', 'size'], axis=1, inplace=True)
```

8.3 添加列

```
import pandas as pd
```

```python
# 创建一个示例 DataFrame
data = {
    'A': [1, 2, 3],
    'B': [4, 5, 6]
}
df = pd.DataFrame(data)
print("原始 DataFrame:")
print(df)

# 添加一列，所有值都是同一个值
df['C'] = 7
print("\n添加列 C 后的 DataFrame:")
print(df)

# 使用现有列进行计算来添加新列
df['D'] = df['A'] + df['B']
print("\n添加列 D (A + B) 后的 DataFrame:")
print(df)

# 基于条件添加新列
df['E'] = df['A'].apply(lambda x: 'Even' if x % 2 == 0 else 'Odd')
print("\n添加列 E (基于 A 的条件) 后的 DataFrame:")
print(df)

# 创建另一个 DataFrame
data2 = {
    'F': [10, 11, 12]
}
df2 = pd.DataFrame(data2)

# 将新列合并到原始 DataFrame
df['F'] = df2['F']
print("\n添加列 F (从另一个 DataFrame) 后的 DataFrame:")
print(df)

# 使用 assign 方法添加新列
df = df.assign(G=df['A'] * 2)
print("\n使用 assign 方法添加列 G (A * 2) 后的 DataFrame:")
print(df)

# 添加多列
df[['H', 'I']] = pd.DataFrame({
    'H': [100, 101, 102],
    'I': [200, 201, 202]
})
print("\n添加多列 H 和 I 后的 DataFrame:")
print(df)
```

九、 k 折交叉验证

当有多个不同的模型（结构不同、超参数不同等）可以选择时，我们通过 K 折交叉验证来选取对于特定数据集最好的模型。

1. 将含有 N 个样本的数据集，分成 K 份，每份含有 $\frac{N}{K}$ 个样本。选择其中一份作为验证集，另外 K – 1 份作为训练集，验证集集就有 K 种情况。
2. 在每种情况中，用训练集训练模型，用验证集测试模型，计算模型的泛化误差。
3. 交叉验证重复 K 次，平均 K 次的结果作为模型最终的泛化误差。
4. K 的取值一般在 [2,10] 之间。K 折交叉验证的优势在于,同时重复运用随机产生的子样本进行训练和验证,10 折交叉验证是最常用的。
5. 训练集中样本数量要足够多，一般至少大于总样本数的 50%。
6. 训练集和验证集必须从完整的数据集中均匀采样。均匀采样的目的是希望减少训练集、验证集与原数据集之间的偏差。当样本数量足够多时，通过随机采样，便可以实现均匀采样的效果。

十、 引用

[1] 深度学习模型数值稳定性——梯度衰减和梯度爆炸的说明-CSDN 博客[EB].https://blog.csdn.net/m0_49963403/article/details/132394707.

[2] 3.11. 模型选择、欠拟合和过拟合 — 《动手学深度学习》 文档[EB] https://zh-v1.d2l.ai/chapter_deep-learning-basics/underfit-overfit.html.

[3] 4.4. 模型选择、欠拟合和过拟合 — 动手学深度学习 2.0.0 documentation[EB]. https://zh.d2l.ai/chapter_multilayer-perceptrons/underfit-overfit.html.

[4] 机器学习 K 折交叉验证知识详解（深刻理解版）（全网最详细）五折交叉验证得到五个模型-CSDN 博客[EB].https://blog.csdn.net/Rocky6688/article/details/107296546.

[5] 理解深度学习模型复杂度评估 全连接层的计算复杂度-CSDN 博客[EB] https://blog.csdn.net/coco_12345/article/details/105742205.

[6] 权重衰减 weight_decay 参数从入门到精通 weight decay-CSDN 博客[EB] https://blog.csdn.net/zhaohongfei_358/article/details/129625803.

[7] 深度学习入门笔记-13 正则化-丢弃法 Dropout[EB]-知乎专栏. https://zhuanlan.zhihu.com/p/608914928.

[8] 机器学习：数据预处理之独热编码（One-Hot）详解-CSDN 博客[EB/OL]. [2024-06-19]. https://blog.csdn.net/zyc88888/article/details/103819604. [9] 机器学习 K 折交叉验证知识详解（深刻理解版）（全网最详细）五折交叉验证得到五个模

型-CSDN 博客[EB/OL]. [2024-06-19]. https://blog.csdn.net/Rocky6688/article/details/107296546.