

CS7304H: Statistical Learning

Project Report

pangbo

2024 年 2 月 19 日

1 数据

本次的实验数据来自于一个文本分类数据集，这些样本被分类到 20 个不同的类别中。原始文本已经进行了特征提取预处理，每个样本为一个 10000 维向量。数据集分为训练数据与测试数据，训练数据包含 11314 个样本，测试数据包含 7532 个样本。

特征提取处理后的数据具有极高的稀疏度，训练数据与测试数据中分别约有 99.15% 与 99.17% 的元素为 0。

从 10000 个特征维度来看：对于训练数据集，每个维度平均有 95.6 个非零元素，最多 5272 个非零元素，最少的维度没有非零元素，维度之间非零元素个数的方差为 46729；对于测试数据集，每个维度平均有 62.2 个非零元素，最多 3546 个非零元素，最少的维度没有非零元素。特征维度非零元素个数的分布如图1所示。

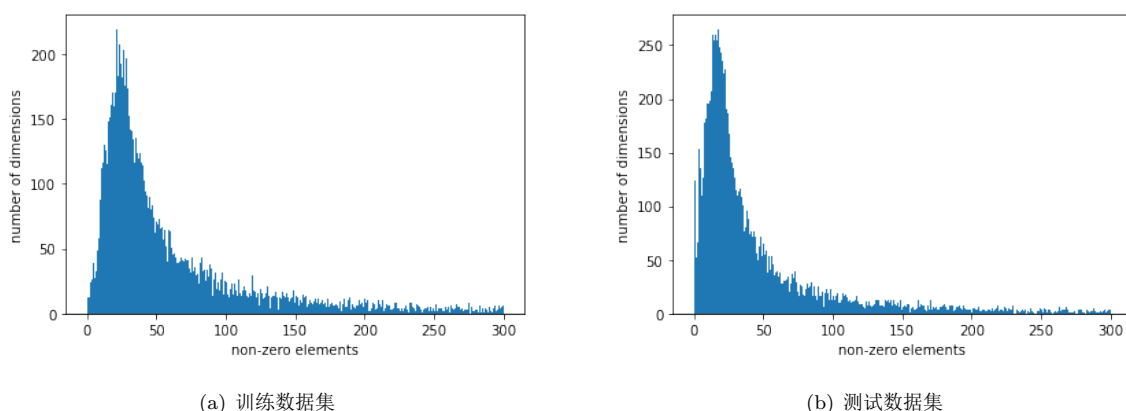
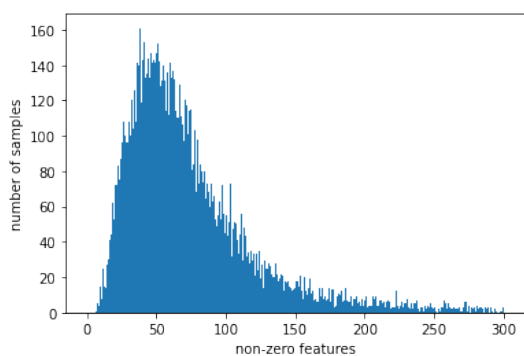


图 1: 特征维度非零元素数分布

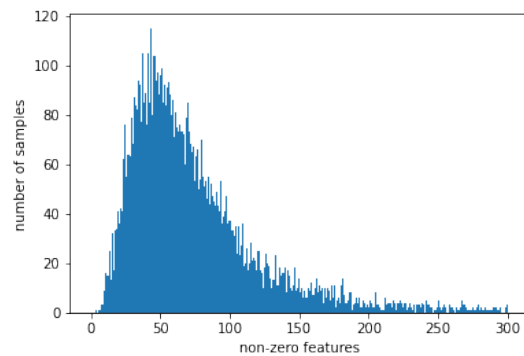
从样本的角度看：对于训练数据集，平均每个样本有 84.5 个非零维度，单个样本最多有 1600 个非零维度，最少 6 个非零维度，非零维度数方差为 8635；对于测试数据集，平均每个样本有 82.6 个非零维度，单个样本最多有 1600 个非零特征维度，最少 3 个非零维度。样本非零特征维度数的分布如图2所示。

训练数据的标签分布如图3所示。可以看出，训练数据的标签分布较为均匀，类别不平衡问题不明显，我们可以近似认为所有类别的先验概率是一致的。

为了更好展现数据的特征，我们分别使用 t-SNE 算法与 PCA 算法将数据降维到二维空间，降



(a) 训练数据集



(b) 测试数据集

图 2: 样本非零特征数分布

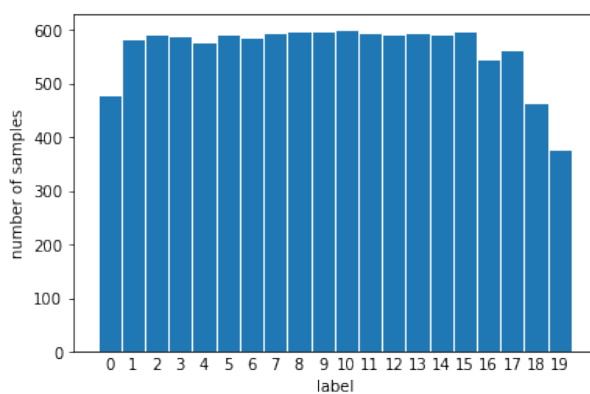
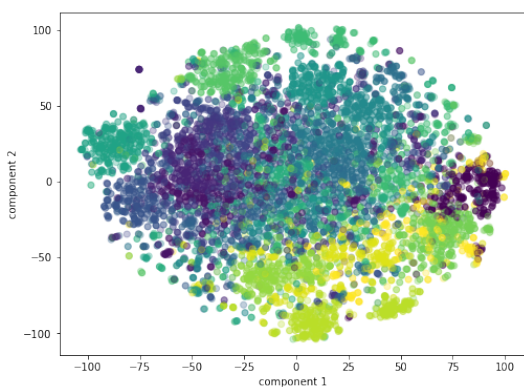
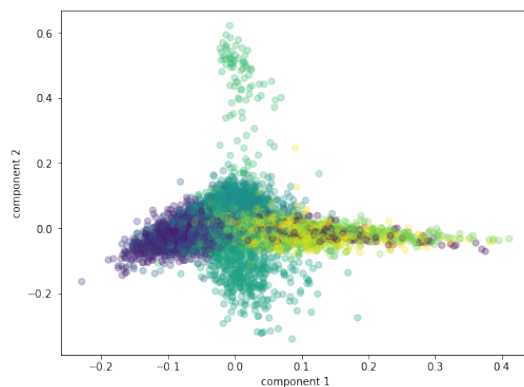


图 3: 训练数据标签分布

维结果如图4所示。图中每一个点代表训练集中的一个样本，点的颜色代表其所属的分类。从低维来看，这个分类问题是较为复杂甚至完全不可分的。



(a) t-SNE



(b) PCA

图 4: 样本降维可视化

2 模型

2.1 SVM

支持向量机 (Support Vector Machine, SVM) 是一种二分类模型, 它的基本模型是定义在特征空间上的间隔最大的线性分类器, 间隔最大使它有别于感知机; 将 SVM 与核方法结合, SVM 也可以用来求解非线性模型。

SVM 优化问题可以表示为一个约束优化问题:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2} \|w\|^2 \\ \text{s.t.} \quad & y_i(w^T x_i + b) \geq 1, \quad i = 1, 2, \dots, n \end{aligned} \quad (1)$$

对于一个实际问题, 直接求解问题 1 是非常困难的。首先, 输入 x 维度可能是极高的, 其次由于约束条件 $y_i(w^T x_i + b) \geq 1$ 的存在, 这个问题可能不存在解, 也就是我们的数据集在数学上可能不是线性可分的。

为了解决这个问题, 我们可以使用利用罚函数法, 将约束条件转化为目标函数的惩罚项, 从而将约束优化问题转化为无约束优化问题。

$$\min_{w,b} \quad \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \max \{0, 1 - y_i(w^T x_i + b)\} \quad (2)$$

优化问题2中 C 是一个超参数, 用来控制惩罚项的权重。当 C 趋于无穷大时, 惩罚项的权重趋于无穷大, 约束条件变为严格限制, 此时目标函数的最优解就是问题1的最优解。需要注意的是问题2的解不一定满足问题1的约束条件, 这表明训练得到的 SVM 可能会误分类部分训练数据点, 在本场景下, 我们可以容忍这种情况的出现。

为了优化问题2, 我们可以使用梯度下降法, 算法伪代码如算法1所示。

Algorithm 1: SVM 优化算法

Input: Train dataset $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$, Learning rate η , Penalty factor

C

Output: SVM parameters w, b

```
1 Initialize  $w = 0, b = 0$ 
2 while do
3   for  $i = 1, 2, \dots, n$  do
4     if  $y_i(w^T x_i + b) < 1$  then
5        $w \leftarrow w - \eta w + \eta C y_i x_i$ 
6        $b \leftarrow b + \eta C y_i$ 
7     else
8        $w \leftarrow w - \eta w$ 
9     end
10  end
11 end
```

单个 SVM 只能进行二分类, 为了实现多分类, 需要将多个 SVM 进行组合, 常用的方法有 One-vs-One 和 One-vs-Rest 两种:

1. **One-vs-One:** 对于 K 个类别, 训练 $K(K-1)/2$ 个 SVM, 每个 SVM 只对两个类别进行分类, 预测时, 将输入样本输入到所有的 SVM 中, 选择获得最多正样本的类别作为预测结果。
2. **One-vs-Rest:** 对于 K 个类别, 训练 K 个 SVM, 每个 SVM 只对一个类别进行分类。训练时, 将这个类别的样本作为正样本, 其他类别的样本作为负样本。预测时, 将输入样本输入到所有的 SVM 中, 选择输出最大的类别作为预测结果。

由于本文的数据集有 20 个分类, One-vs-One 方法需要训练 190 个 SVM, 而 One-vs-Rest 方法只需要训练 20 个 SVM, 因此我们考虑使用 One-vs-Rest 方法。

2.2 Logistic Regression

逻辑回归 (Logistic Regression) 是一种广义的线性回归模型。对于二分类问题, 其形式为:

$$P(Y = 1|x) = \frac{e^{w^T x + b}}{1 + e^{w^T x + b}} \quad (3)$$

逻辑回归可以较为容易地推广到多分类问题, 对于 K 分类问题, 其形式为:

$$P(Y = k|x) = \frac{e^{w_k^T x + b_k}}{\sum_{i=1}^K e^{w_i^T x + b_i}} \quad (4)$$

对分类问题, 我们可以使用交叉熵损失函数计算预测损失:

$$L = - \sum_{i=1}^n \sum_{k=1}^K y_{ik} \log P(Y = k|x_i) \quad (5)$$

基于交叉熵损失函数, 我们可以使用梯度下降法优化模型参数 w, b , 其梯度为

$$\begin{aligned} \frac{\partial L}{\partial w_k} &= - \sum_{i=1}^n x_i (y_{ik} - P(Y = k|x_i)) \\ \frac{\partial L}{\partial b_k} &= - \sum_{i=1}^n (y_{ik} - P(Y = k|x_i)) \end{aligned} \quad (6)$$

优化算法伪代码如算法2所示。

Algorithm 2: Logistic Regression 优化算法

Input: Train dataset $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$, Learning rate η

Output: Logistic Regression parameters w, b

```

1 Initialize  $w = 0, b = 0$ 
2 while do
3   for  $i = 1, 2, \dots, n$  do
4      $p_i \leftarrow \text{SOFTMAX}(w^T x_i + b)$ 
5      $w \leftarrow w + \eta x_i (y_i - p_i)$ 
6      $b \leftarrow b + \eta (y_i - p_i)$ 
7   end
8 end
```

2.3 神经网络

全连接网络是最简单的神经网络结构，它的每个神经元都与前一层的所有神经元相连，形成一个密集的连接结构。全连接网络的每个神经元都有一个权重向量和一个偏置项，输入向量限于权重向量进行内积，加上偏置项，再通过激活函数进行非线性变换，便可以得到神经元的输出。

单层全连接网络可以表示成如下的数学形式：

$$z^{(l)} = W^{(l)}a^{(l-1)} + b^{(l)} \quad (7)$$

$$a^{(l)} = f(z^{(l)}) \quad (8)$$

其中 $W^{(l)}$ 是第 l 层的权重矩阵， $b^{(l)}$ 是第 l 层的偏置向量， f 是激活函数， $a^{(l)}$ 是第 l 层的输出向量。

神经网络可以通过反向传播算法进行训练：我们首先定义一个损失函数，然后让损失函数值对网络中的每个参数求偏导，得到每个参数的梯度，最后使用梯度下降法更新参数。对于分类问题，我们可以使用交叉熵损失函数计算预测损失，其函数形式已在公式5中给出。

由于本场景下的问题并不复杂，我们使用单个隐藏层的全连接网络进行拟合。

神经网络拥有众多参数，这导致模型有较高的过拟合风险。我们在隐藏层与输出层之间加入了 Dropout 层，Dropout 层会随机地将一些神经元的输出置为 0，从而减少神经元之间的依赖关系，防止过拟合。此外我们也将使用 L2 正则化，对模型的权重进行惩罚，抑制模型过拟合。

3 模型选择

模型选择旨在选出一个超参数集合，使得该超参数集合所代指的模型簇在指定问题上获得最好的泛化能力。

3.1 Cross Validation

交叉验证（Cross Validation）是一种统计学上将数据样本切分为训练集和测试集的方法。在交叉验证中，我们将数据集切分为 K 个大小相同的子集，每个子集都尽可能保持数据集的原始分布。然后我们依次将每个子集作为测试集，其他子集作为训练集，训练模型并计算模型在测试集上的性能。最后我们将 K 次测试结果的平均值作为模型的性能。

3.2 Bootstrap

自助法（Bootstrap）是一种统计学上用来估计统计量偏差的方法。在自助法中，我们从数据集中有放回地抽取 n 个样本，作为训练集，剩余的样本作为测试集，训练模型并计算模型在测试集上的性能。重复这个过程 K 次，最后我们将 K 次测试结果的平均值作为模型的性能。

相较于交叉验证，自助法允许数据放回，对于训练数据较少的场景，自助法可以更好地利用数据集。

3.3 Grid Search

网格搜索（Grid Search）是一种超参数优化方法。对于实际的机器学习问题，模型往往有多个超参数，超参数之间会互相影响，如果我们每次只对一个超参进行搜索，最终的组合结果并不一定是最优的。

在网格搜索中，我们首先选定每个超参数的取值范围，然后取笛卡尔积，也即穷举所有超参数的取值组合，对于每个超参数组合，使用交叉验证或者自助法计算模型的性能，最后选出性能最好的超参数组合。

4 实验

在实验部分，我们会比较不同模型的性能。对于每一种模型，我们首先会使用网格搜索找到性能最好的一组超参数，然后使用该组超参数在完整训练集上训练模型，用得到的模型预测测试集，提交预测结果获得测试分数。

在网格搜索中，对于每一组超参，我们使用 5 折交叉验证计算模型的性能。

4.1 Logistic Regression

逻辑回归拥有两个超参数：学习率与迭代轮数。对这两个超参数的搜索结果如表4.1所示。在所有的超参数组合中，学习率为 100，训练轮数为 1000 时，模型获得了最好的性能，5 折交叉验证的平均准确率为 0.9020。

表 1: Logistic Regression 网格搜索结果

lr \ n_iters			
	500.0	1000.0	2000.0
0.1	0.4578	0.6419	0.7412
1.0	0.7970	0.8211	0.8422
10.0	0.8662	0.8854	0.8950
100.0	0.9004	0.9020	0.9000
1000.0	0.8809	0.8802	0.8817

使用上述超参数在训练集上训练模型，预测测试集，提交预测结果，获得测试分数为 0.83488，如图5所示。



图 5: Logistic Regression 测试分数

4.2 SVM

支持向量机拥有两个超参数：学习率与惩罚因子。对这两个超参数的搜索结果如表4.2所示。在所有的超参数组合中，学习率为 10^{-6} ，惩罚因子为 10^5 时，模型获得了最好的性能，5 折交叉验证的平均准确率为 0.9045。

使用上述超参数在训练集上训练模型，预测测试集，提交预测结果，获得测试分数为 0.82267，如图6所示。

表 2: SVM 网格搜索结果

lr \ C	10^4	10^5	10^6	10^7
10^{-7}	0.8643	0.9045	0.8972	0.8820
10^{-6}	0.8951	0.8975	0.8857	0.8751
10^{-5}	0.8867	0.8675	0.8122	0.7728
10^{-4}	0.7781	0.6021	0.5692	0.5667



submit_SVM.csv

Complete · 13d ago

0.82267



图 6: SVM 测试分数

4.3 神经网络

神经网络具有较多超参数，我们首先固定模型具有一个隐藏层，且训练中的学习率为 10^{-3} ，其余需要搜索的超参数包括：参数正则化系数、Dropout 层的丢弃率、隐藏层的神经元个数、训练轮数。

对这些超参数的搜索结果如表4.3所示。表中参数的含义如下：

- decay: Adam 优化器的 weight_decay 参数，等效于 L2 正则化的系数。
- drop: Dropout 层的丢弃率，即在训练时屏蔽单个神经元输出的概率。
- hidden: 隐藏层的神经元个数。
- epoch: 训练轮数。

在所有的超参数组合中，参数正则化系数为 10^{-6} ，Dropout 层的丢弃率为 0.9，隐藏层的神经元个数为 1024，训练轮数为 20 时，模型获得了最好的性能，5 折交叉验证的平均准确率为 0.9167。

使用上述超参数在训练集上训练模型，预测测试集，提交预测结果，获得测试分数为 0.85771，如图7所示。



submit_nn.csv

Complete · 8d ago

0.85771



图 7: 神经网络测试分数

4.4 降维

最后，我们还尝试使用 PCA 对特征进行降维，然后使用降维后的特征训练模型。

降维后的测试分数如图8所示。根据测验结果，使用 PCA 降维后训练结果并不理想，最终准确率不如使用原始数据训练的模型。我认为可能的原因在于 PCA 降维后，数据的可分性变差了，尤其是本实验中大多采用线性模型，只有对于足够稀疏的特征空间，线性决策边界才能将不同类别的样本分开，而 PCA 降维后，样本点分布变得紧密与复杂，导致线性模型的性能变差。

表 3: 神经网络网格搜索结果

	hidden	1024			1536			2048		
	epoch	15	20	25	15	20	25	15	20	25
decay	drop									
0	0.7	0.9130	0.9135	0.9121	0.9123	0.9115	0.9118	0.9122	0.9121	0.9121
	0.8	0.9144	0.9135	0.9121	0.9141	0.9129	0.9122	0.9125	0.9117	0.9129
	0.9	0.9146	0.9147	0.9142	0.9151	0.9152	0.9136	0.9136	0.9140	0.9135
10^{-7}	0.7	0.9129	0.9120	0.9132	0.9118	0.9121	0.9124	0.9128	0.9119	0.9123
	0.8	0.9141	0.9132	0.9128	0.9127	0.9118	0.9133	0.9126	0.9115	0.9131
	0.9	0.9129	0.9167	0.9149	0.9154	0.9135	0.9136	0.9144	0.9140	0.9136
10^{-6}	0.7	0.9129	0.9117	0.9113	0.9117	0.9113	0.9122	0.9117	0.9110	0.9107
	0.8	0.9130	0.9137	0.9122	0.9133	0.9121	0.9117	0.9113	0.9116	0.9119
	0.9	0.9142	0.9154	0.9152	0.9153	0.9142	0.9138	0.9146	0.9131	0.9121
10^{-5}	0.7	0.9109	0.9104	0.9087	0.9095	0.9086	0.9080	0.9080	0.9075	0.9068
	0.8	0.9123	0.9106	0.9101	0.9115	0.9099	0.9092	0.9099	0.9104	0.9080
	0.9	0.9141	0.9142	0.9126	0.9127	0.9129	0.9122	0.9142	0.9133	0.9104



	submit_lr_pca.csv Complete · 19h ago	0.82797	<input type="checkbox"/>
	submit_NN_pca.csv Complete · 19h ago	0.8293	<input type="checkbox"/>
	submit_SVM_pca.csv Complete · 9d ago	0.7632	<input type="checkbox"/>

图 8: 降维后测试分数