记录一次失败的字节跳动面试「算法」

简介

字节跳动面崩了，记录一下。

神经网络参数如何初始化

Deeplearning.ai的教程Initializing neural networks :

首先神经网络参数不能初始化为0或者任意相同的常量。如果网络参数都是相同常量，那个每个隐层节点对最终节点的贡献度一致，导致各个节点按照相同的方式更新。不能使不同神经元学习不同的东西。

考虑线性模型，初始权重过小导致梯度消失，过大梯度爆炸。需要寻找合适的参数，有两个假设为

1. 激活函数输出后的均值为0
2. 每激活函数输出的方差应该在层与层间保持相同

假设某层的前传播公式为

$$
\begin{aligned}
a^{[l-1]} &= g^{[l-a]}(z^{[l-1]}) \\
z^{[l]} &= W^{[l]}a^{[l-1]} + b^{[l]} \\
a^{[l]} &= g^{[l]}(z^{[l]})
\end{aligned}
$$

对应假设为

$$
\begin{aligned}
E\left[a^{[l-1]}\right] &= E\left[a^{[l]}\right] \\
Var\left(a^{[l-1]}\right) &= Var\left(a^{[l]}\right)
\end{aligned}
$$

Xavier 初始化

一般建议使用 Xavier 初始化:

$$
\begin{aligned}
W^{[l]} &\sim \mathcal{N}(\mu = 0, \sigma^2 = \frac{1}{n^{[l-1]}}) \\
b^{[l]} &= 0
\end{aligned}
$$

权重通过正态分布采样，$n^{[l-1]}$ 是上层神经元的个数。偏置初始化为0. 推导过程如下，假设网络激活函数为 $tanh$

$$
\begin{aligned}
z^{[l]} &= W^{[l]}a^{[l-1]} + b^{[l]} \\
a^{[l]} &= tanh(z^{[l]})
\end{aligned}
$$

其中:

$$
tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}
$$

目的需要找到两层间方差的对应关系，即$Var(a^{[l-1]})$ 和 $Var(a^{[l]})$ 。首先注意tanh的性质: $tanh(-x) = -tanh(x)$ 另外就是在接近0的时候 $tanh(x) \approx x$ 。在初始化后，参数都是很小的值，有

$$
Var(a^{[l]}) = Var(tanh(z^{[l]})) \approx Var(z^{[l]})
$$

考虑 $z^{[l]} = W^{[l]}a^{[l-1]} + b^{[l]} = vector(z_1^{[l]}, z_2^{[l]}, \cdots, z_3^{[l]})$，对应的关系可以推导为

$$
z_k^{[l]} = \sum_{j=1}^{n^{[l-1]}} w_{kj}^{[l]} a_j^{[l-1]} + b_k^{[l]}
$$

考虑每个元素的方差, 注意偏置项都是0，可以省略掉

$$
Var(a_k^{[l]}) = Var(z_k^{[l]}) = Var\left(\sum_{j=1}^{n^{[l-1]}} w_{kj}^{[l]} a_j^{[l-1]}\right)
$$

先使用3个假设:

1. 权重独立同分布
2. 输入独立同分布
3. 权重和输入相互独立

继续展开:

$$
Var(a_k^{[l]}) = Var(z_k^{[l]}) = Var\left(\sum_{j=1}^{n^{[l-1]}} w_{kj}^{[l]} a_j^{[l-1]}\right) = \sum_{j=1}^{n^{[l-1]}} Var(w_{kj}^{[l]} a_j^{[l-1]})
$$

考虑方差展开式

$$
Var(w_i x_i) = E[w_i]^2 Var(x_i) + E[x_i]^2 Var(w_i) + Var(w_i)Var(x_i)
$$

注意输入和权重的期望是0，代入可以得到下个公式，这个公式是方差缩放公式。

$$Var(z_k^{[l]}) = \sum_{j=1}^{n^{[l-1]}} Var(w_{kj}^{[l]}) Var(a_j^{[l-1]}) = n^{[l-1]} Var(W^{[l]}) Var(a^{[l-1]})$$

上述等式成立的时候，需要有下列假设成立

$$Var(w_{kj}^{[l]}) = Var(w_{11}^{[l]}) = Var(w_{12}^{[l]}) = \cdots = Var(W^{[l]})$$

同理要满足

$$Var(z^{[l]}) = Var(z_k^{[l]})$$

那么要满足开始的假设

$$Var(a^{[l]}) = n^{[l-1]} Var(W^{[l]}) Var(a^{[l-1]}) = Var(a^{[l-1]})$$

需要有

$$Var(W^{[l]}) = \frac{1}{n^{[l-1]}}$$

在实际过过程中，Xavier初始化方式有两种方差，如下

$$W^{[l]} \sim \mathcal{N}(0, \frac{1}{n^{[l-1]}})$$
$$W^{[l]} \sim \mathcal{N}(0, \frac{2}{n^{[l-1]} + n^{[l]}})$$

He 初始化

使用ReLu通常一般神经元输出为0，此时分布的方差为恒等函数时候的一半，此时考虑前向传播，理想方差为

$$Var(w^{[l]}) = \frac{2}{n^{[l-1]}}$$

采用高斯分布方差如上，如果采用区间为$[-r, r]$ 均匀分布初始化参数，那么$r = \sqrt{\frac{6}{n^{[l-1]}}}$

Dropout在forward里面怎么做

Dropout来自论文 「Dropout: A Simple Way to Prevent Neural Networks from Overfitting」

在前向传播过程中，公式为

$$
\begin{aligned}
r_j^{(l)} &\sim \text{Bernoulli}(p) \\
\widetilde{\mathbf{y}}^{(l)} &= \mathbf{r}^{(l)} * \mathbf{y}^{(l)} \\
z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)} \widetilde{\mathbf{y}}^{(l)} + b_i^{(l+1)} \\
y_i^{(l+1)} &= f(z_i^{(l+1)})
\end{aligned}
$$

$\mathbf{r}$ 是概率1为p 的伯努利分布生成的向量。在前向传播过程中，训练得到参数被更新 $W_{test}^{(l)} = pW^{(l)}$

L1和L2正则化的区别

参考「百面机器学习」

L1 正则化使模型参数具有稀疏性。即很多权重参数都是0. 优点是可以自动做特征选择。 L2 正则化对防止模型过拟合效果更好，因为网络倾向去使用所有的输入特征而不是严重依赖小部分特征。

考虑L1正则

$$
J(\mathbf{w}) = \frac{1}{n}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \lambda\|\mathbf{w}\|
$$

最小化代价函数等价于

$$
\begin{aligned}
&\min_{\mathbf{w}} \frac{1}{n}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 \\
&s.t. \quad \|w\| \leq C
\end{aligned}
$$

函数空间和解空间已经有了，最有参数为解空间和函数空间的交集中使得函数最小的点。「百面机器学习」绘制出了下图

最优解是解空间边缘和函数等高线的交点。L1棱形解空间更容易在尖角处与等高线碰撞得到稀疏解。

AUC是什么，写一下代码

> *Probabilistic interpretation of AUC*
>
> *The Probabilistic Interpretation of AUC*

AUC时ROC曲线下的面积。 ROC通过FP(假阳性)和TP(真阳性)计算。 对于二分类需要考虑混淆矩阵

			预测
		$\hat{y}=c$	$\hat{y}\neq c$
真实类别	$y=c$	TP	FN
	$y\neq c$	FP	TN

ROC(receiver operating characteristic curve) 通过 TPR 和 FPR得到

$$TPR=\frac{TP}{TP+FN}$$

$$FPR=\frac{FP}{FP+TN}$$

通过FPR为横轴， TPR为纵轴，在不同分类置信度阈值下，可以绘制ROC曲线。如下图，ROC一定会经过(0, 0), (1, 1)：

AUC是ROC曲线下的面积。ROC曲线有一个很好的性质，在测试集正负样本分布变化的时候，ROC曲线保持不变。AUC计算方式主要有两种：

- 从预测的置信度排序，按照排序后由高到低选择选择阈值，计算对应TPR和 FPR绘制曲线
- AUC具有概率学上的意义：随机选取一个正样本和一个负样本，分类器给正样本打分大于分类器给负样本打分的概率。使用组合数学求解

$$AUC=\frac{\sum_{i\in positiveClass} rank_i-\frac{M(1+M)}{2}}{M\times N}$$

公式中 M 和 N 分别为正负样本个数。$rank$是将样本按照置信度排序后，置信度最高的样本$rank=M+N$

python 代码如下, 来自AUC曲线计算方法及代码实现

```python
import numpy as np
from sklearn.metrics import roc_auc_score


def calc_auc(y_labels, y_scores):
    f = list(zip(y_scores, y_labels))
    rank = [values2 for values1, values2 in sorted(f, key=lambda x: x[0])]
    rankList = [i + 1 for i in range(len(rank)) if rank[i] == 1]
    pos_cnt = np.sum(y_labels == 1)
    neg_cnt = np.sum(y_labels == 0)
```

```
    auc = (np.sum(rankList) - pos_cnt * (pos_cnt + 1) / 2) / (pos_cnt * neg_cnt)
    return auc


def get_score():
    # 随机生成100组label和score
    y_labels = np.zeros(100)
    y_scores = np.zeros(100)
    for i in range(100):
        y_labels[i] = np.random.choice([0, 1])
        y_scores[i] = np.random.random()
    return y_labels, y_scores


if __name__ == '__main__':
    y_labels, y_scores = get_score()
    print('sklearn AUC:', roc_auc_score(y_labels, y_scores))
    print(calc_auc(y_labels, y_scores))

```

编程题，类似实现 ndarray.shape

以字符串形式给出一个 ndarray，如 "[[[7,7,7],[8,8,8]]]"，请写一个程序，输出它的 shape，以上面的例子为例，输出：(1,2,3)。

输入："[[[0.7,7,7],[8,8,8]]]"，输出：(1,2,3). 输入："[[[7],[7],[7]],[[8],[8],[8]]]”, 输出：(2,3,1)
输入："[[[7],[7],[]]]”, 输出：error

代码如下，面试没写出来，递归来统计，从内层到外层一次统计。

```
def _count_shape(strs):
    cnt = 0
    for idx, c in enumerate(strs):
        if c == '[':
            cnt += 1
        else:
            break
    if cnt == 0:
        return False, [-1]
    if cnt == 1:
        nums = strs[1:-1].split(',')
        try:
            nums = list(map(float, nums))
        except Exception as E:
            return False, [-1]
```

```
        if len(nums) == 0:
            return False, [-1]
        return True, [len(nums)]
    else:
        splits = []
        pre = ''
        cur_cnt = 0
        idx = 1
        while idx < len(strs) - 1:
            c = strs[idx]
            if c == '[':
                cur_cnt += 1
            elif c == ']':
                cur_cnt -= 1
            pre += c
            idx += 1
            if cur_cnt == 0:
                splits.append(pre)
                pre = ''
                while idx < len(strs) - 1 and strs[idx] !=
'[':
                    idx += 1
        ret = [_count_shape(_strs) for _strs in splits]
        if not ret:
            return False, [-1]
        isvalid, r = ret[0]
        if not isvalid:
            return False, [-1]
        for ci, cr in ret[1:]:
            if not ci or r != cr:
                return False, [-1]
        return True, r + [len(ret)]


def shape(strs):
    ret, shape = _count_shape(strs)
    if ret:
        return shape[::-1]
    else:
        print('error')
        return -1


if __name__ == '__main__':
    print(shape("[[[0.7,7,7],[8,8,8]]]"))
    print(shape("[[[7],[7],[7]],[[8],[8],[8]]]"))
```

```
61      print(shape("[[[7],[7],[]]]"))
```

运行结果

```
[1, 2, 3]
[2, 3, 1]
error
-1
```