深度学习：homework1

张驰　　前沿交叉学科研究院

2019 年 5 月 9 日

1　探索 softmax 的其它表达方式

1.1　实验准备

本次实验要求探索 softmax 的其它表达方式，期望找到 softmax 的其它替代方式，如图 1 所示，在这里讨论以下 4 种情况：

$$\text{softmax}(x) = \frac{e^x}{\sum e^x}$$

Which of below work as alternative to Softmax?

$$\text{abs-max}(x) = \frac{|x|}{\sum |x|}$$

$$\text{square-max}(x) = \frac{x^2}{\sum x^2}$$

$$\text{plus-one-abs-max}(x) = \frac{1+|x|}{\sum 1+|x|}$$

$$\text{non-negative-max}(x) = \frac{\max(0,x)}{\sum \max(0,x)}$$

图 1: softmax 其它表达方式

本次实验中，我没有使用 extra_32*32.mat 数据，因此将'use_extra_data' 设置为 False。其次在实际运行中，并不需要迭代 60 次，因此我减少了迭代次数来减少程序运行的时间，并对应地改变了学习率的衰减，保证结果能最后收敛，如下所示：

```
nr_epoch = 40
boundaries = [xx * 10,xx * 30]
```

其次，由于 tensorflow 自带的 softmax_cross_entropy 函数内部已经定义为 softmax 的形式，所以我们不能使用 baseline 的损失函数，需要将损失函数进行修改，修改代码如下

```
loss=-tf.reduce_mean(tf.reduce_sum(tf.cast(label_onehot,dtype=tf.float32)*tf.log(tf.
                                   clip_by_value(preds,1e-10,1.0)),axis=-1))+loss_reg
```

这里利用 tf.clip_by_value() 的方法将预测得到的概率进行平滑，保证结果中不会出现 0 概率对 log 运算的反向传播。

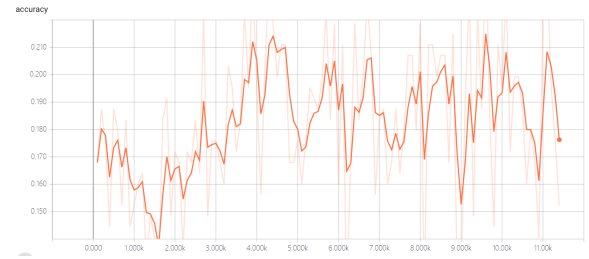
1.2 实验分析

首先来看一下实验运行结果，表 1 是各 softmax 方法的最好验证精度结果:

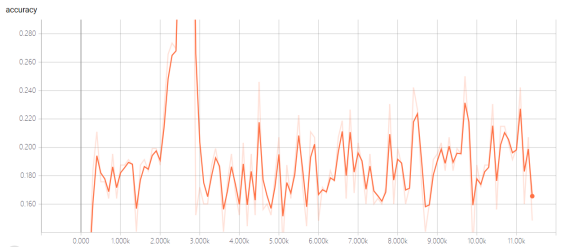
表 1: 各 softmax 最优验证结果

Name	Method	test−accuracy
baseline	softmax	94.3%
q1.1	abs-max	19.7%
q1.2	square-max	19.9%
q1.3	plus-one-abs-max	95.1%
q1.4	non-negative-max	70.1%

从结果中我们可以发现，plus-one-abs-max 方法基本上和 baseline 的结果相差不多，但是 abs-max，square-max，non-negative-max 方法在给定的迭代次数中不能有效的收敛，测试精度和训练精度都比较低，如图 2 所示

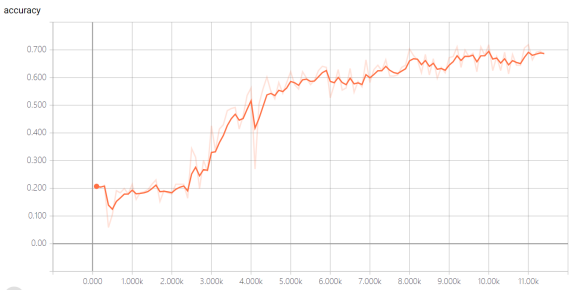


(a) train-loss for abs-max



(b) train-loss for square-max

(c) train-loss for plus-one



(d) train-loss for non-negative

图 2: abs-max，square-max，plus-one-abs-max,non-negative-max 方法的训练精度变化

分析结果：1. 首先 baseline 的 softmax 方法进入到交叉熵中，损失函数是一个凸函数，且 loss 越大，梯度越大，便于反向传播实现快速收敛。

2. 从 abs-max 方法来看，由于其是一个偶函数，对于 logits 中绝对值高的输出值经过该方法后会有较大的概率，在代入到交叉熵函数中进行计算时不是凸函数，因此从图中可以发现其训练精度始终在 0.2 左右徘徊，需要更长的时间才能收敛。

3.square-max 方法和 abs-max 方法比较相似，同样是由于其为偶函数，代入到交叉熵函数中进行计算时不是凸函数，但其求导结果后相对 abs-max 方法有 2 倍的提高，因此，从图中可以观察训练精度一度达到 0.3，并且随着迭代次数的增加，square-max 方法有望比 abs-max 方法更早达到收敛。

4.plus-one-abs-max 方法是这 4 个方法当中唯一接近 baseline 的方法，训练精度不像前两个方法处于一个较低的精度，而是持续增高接近 1，对应的测试精度也达到 0.95，原因可能是加 1 后实现了平滑，避免的分子分母过分产生的异常状况。

5.non-negative-max 方法结果比较可观，从曲线图可以发现，曲线持续增高，随着迭代次数增加，仍有可能继续提高训练精度以及对应的测试精度。分析原因可能是 non-negative-max 方法实际是是 relu 函数，将小于 0 的预测值都设置为 0，那么大于 0 的预测值将获得较高的概率，由于我们设置了 1e-10 代替 0，使得反向传播过程仍可以对小于 0 的输出值进行反向传播修改，不会因为出现 0 值而几乎” 停止”。它的收敛速度快于前两个方法，但低于 plus-one-abs-max 方法。

2 回归损失与分类损失对比

- 将 baseline 的交叉熵损失改为预测概率和真实标签的独热向量的欧式距离

2.1 实验准备

该实验主要是对损失函数进行修改，欧式距离即为平方差损失，关键代码如下：

```
    if args.loss == 'regression':
      loss = tf.reduce_mean(tf.reduce_sum(tf.square(preds - tf.cast(label_onehot, dtype=tf.float32))
                                          , axis=1))
    else:
      loss = tf.losses.softmax_cross_entropy(label_onehot, logits) + loss_reg
```

2.2 实验分析

实验结果如表 2 所示：

表 2: 回归与分类损失最优结果对比

Name	Method	test−accuracy
baseline	softmax	94.3%
q2	euclidean distance	95.2%

从实验结果可以看出，使用预测概率和真实标签的独热向量的平方差作为损失函数结果与 baseline 的交叉熵损失相比有略微的提高。但从原理上来看，交叉熵损失是比平方差回归收敛速度更快的，收敛结果更好的。所以这里出现相反的结果让我捉摸不透，希望老师在课上能解答。

3 lp-pooling

3.1 实验准备

该实验要求将所有的池化层改为 lp-pooling,lp-pooling 的表达式如下：

$$O = \left(\sum \sum I(i,j)^P * G(x,y) \right)^{\frac{1}{P}} \tag{1}$$

根据论文中 (图 3) 的描述，这个 pooling 与我们以往使用的 avg-pooling 和 max-pooling 不一样，lp-pooling 更像是一个利用卷积来实现池化的功能，这里将卷积核替换为高斯核。

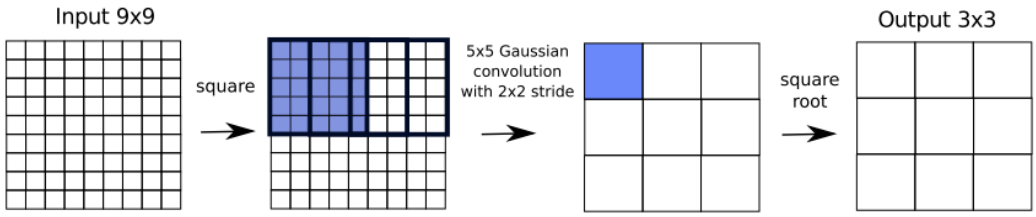


图 3: lp-pooling 池化原理图

根据卷积运算的原理，直接利用 [whc] 的特征，加上 [kkc] 的卷积核进行卷积的话，输出的是一张 [wh1] 的特征图 (c 维会累加成 1 个维度)，因此这里需要做一个拆解：将输入的特征维度 [whc] 拆解为 c 个 [wh1] 的特征，对应的 [kkc] 卷积拆解为 c 个 [kk1] 的高斯核，每个对应地放入卷积函数中生成 c 个 [wh1] 的输出特征并利用 concat 拼接成 [whc] 尺寸，这样就相当于对每一个 channel 的特征进行了卷积，最后利用卷积中 stride 步长设置为 2，最终就完成利用卷积实现池化的效果，lp-pooling 关键代码如下。

```
def LP_pooling_kernelsize3(self, inp, stride, pnum):
  pip = tf.pow(inp, pnum)
  conv_filter = tf.constant([0.05711826, 0.12475775, 0.05711826,
    0.12475775, 0.27249597, 0.12475775,
  0.05711826, 0.12475775, 0.05711826],shape=[3,3],dtype=tf.float32)
  conv_filter = tf.expand_dims(conv_filter, axis=-1)  #channel
  conv_filter = tf.expand_dims(conv_filter, axis=-1)  #卷积核个数(k,k,1,1)
  pips = tf.unstack(pip, axis=-1) #nwh + c
  z = []
  for p in pips:
    p = tf.expand_dims(p, axis=-1)#con2d要求input是4维的向量(?,w,h,1)
    z.append(tf.nn.conv2d(p, conv_filter, strides=[1, stride, stride, 1], padding='SAME'))
  res = tf.concat(z, axis=-1)
  res = tf.pow(res, 1. / pnum)
  return res
```

其中高斯卷积核是通过 kernel_size=3，sigma=0.8 时计算得到，关键代码如下：

```
def gaussian_2d_kernel(kernel_size=3, sigma=0):
  kernel = np.zeros([kernel_size, kernel_size])
  center = kernel_size // 2
  if sigma == 0:
    sigma = ((kernel_size - 1) * 0.5 - 1) * 0.3 + 0.8

  s = 2 * (sigma ** 2)
  sum_val = 0
  for i in range(0, kernel_size):
    for j in range(0, kernel_size):
      x = i - center
      y = j - center
      kernel[i, j] = np.exp(-(x ** 2 + y ** 2) / s)
      sum_val += kernel[i, j]
  # /(np.pi * s)
  sum_val = 1 / sum_val
  return kernel * sum_val
```

3.2 实验分析

实验结果如表 3 所示：

表 3: lp-pooling 实验最佳结果

Name	Method	test−accuracy
baseline	max-pooling	94.3%
q3	lp-pooling p=-1	93.4%
q3	lp-pooling p=1	93.4%
q3	lp-pooling p=2	95.2%
q3	lp-pooling p=4	95.1%

从实验结果可以看出,lp-pooling 相对 baseline 的 max-pooling 而言，测试精度上有一个明显的提高，说明高斯核对输入特征进行卷积平滑操作后，选取了比最大值更好的特征。其次，对比不同 p 值，其结果差异不大，当 p=2 时效果是最好的，在实验中发现当 p=4 时，迭代到一定次数后会出现 loss 为 nan 的情况, 原因可能是 4 次方导致结果溢出。

4 正则化

- 1. 尝试使用不同 p 值的 lp 正则化，并选择最好的 p 值.
- 2. 将模型的损失加上正则化项改为减去正则化项.

4.1 lp 正则化实验准备

本次实验我们希望尝试不同 P 值的 lp 正则化，并检验对验证精度的影响，选出最佳的正则化项。以下是 lp 正则化的关键代码：

```
weight_decay = 1e-3

self.reg = lambda w: config.weight_decay * tf.reduce_sum(tf.pow(tf.abs(w), config.lp_reg))
```

此外，由于 baseline 设置的 weight_decay 太小，经过实验发现改变 P 值对结果影响不大。这里我将其值调整为 1e-3，因此结果运行的结果不能再和 baseline 进行比较，只在其内部进行比较。

4.2 lp 正则化实验分析

通过尝试不同的 p 值组合，最后得到的结果如表 4 所示：

表 4: lp 正则化不同 P 值对应的实验结果

P	test−accuracy
-2	77.4%
-1	82.8%
0.5	90.6%
1	90.9%
2	93.6%
4	95.4%

从试验的 5 个值的结果中我们可以看到，负值的运行结果都比较差，当 P=4 时，测试精度是最高的，因此 lp4 正则化对结果的约束在本次实验中是最有效的。

4.3 负正则化实验准备

选择合适的 P 值 (根据上一个实验的结果，这里选择 P=2)，将正则化项前乘以 (-1)，使整个卷积神经网络都减去正则化项。

```
self.reg=lambda w: (-1)*config.weight_decay * tf.reduce_sum(tf.pow(tf.abs(w), config.lp_reg))
```

4.4 负正则化实验分析

实验结果如表 5 所示：

表 5: 负正则化实验结果

Method	test−accuracy
baseline	94.3%
q4.2	94.3%

从结果可以看出，正则化项的正负结果对整个神经网络的精度几乎没有影响。

参考文献

[1] Sermanet P , Chintala S , Lecun Y . Convolutional neural networks applied to house numbers digit classification[C]// 2012 21st International Conference on Pattern Recognition (ICPR 2012). IEEE Computer Society, 2012.