1.softmax loss理解

对于SVM来讲，margin的概念十分清晰易懂，那么对于Softmax而言，如果我们要引入**Max Margin**的概念，那么所谓的margin究竟是什么？

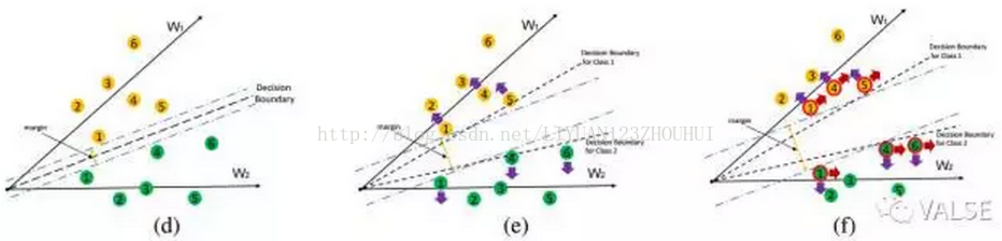
为了能对这一系列工作有更深刻的理解，我们需要从几何上来理解softmax loss究竟在做什么。为了方便表述，下文所指softmax loss均指softmax之前的全连接层+softmax loss function，即

$$\mathcal{L}_{\text{softmax}} = -\frac{1}{N}\sum_{i=1}^{N}\log\left(\frac{e^{\mathbf{w}_{y_i}^T\mathbf{f}_i+\mathbf{b}_{y_i}}}{\sum_{j=1}^{K}e^{\mathbf{w}_j^T\mathbf{f}_i+\mathbf{b}_j}}\right).$$

其中f_i为第i个sample的feature，y_i为其对应的label，w_{y_i}为y_i类对应的weight。为了方便后文推导，我们可以认为bias term b = 0。这里有一个关键的变换是，我们把inner product表示为：

$$\mathbf{w}_j^T\mathbf{f}_i = \|\mathbf{w}_j\|\|\mathbf{f}_i\|\cos(\theta_{\mathbf{w}_j,\mathbf{f}_i}),$$

这个简单的变换是后面这一系列工作的核心：我们可以看到这个inner product的大小，和三个元素相关：**w的l2 norm，f的l2 norm，w和f这两个向量的夹角**。如果我们对于类别没有特殊的假设，那么w的l2 norm可以认为是一致的。那么**可以影响结果的就只有后两项了**，后面介绍的一系列工作都是在这两项上进行改动。在开始之前，我们需要一个在二维的空间中直观的解释来帮助理解：



在这些图中，黄色和绿色的点分别代表两类样本，黑色的虚线代表decision boundary，**黑色的实线代表两个class的weight。这些weight vector，把整个空间划分成了若干个锥形，每个锥形对应着一类样本的空间。**如果假设每一类的weight的l2 norm是一致的，那么每个样本属于哪一类就只和**这个样本在这些weight vector的方向上投影长度相关。**

最左侧的图中，显示的是原始的softmax，可以看到，由于softmax并没有显式地加强margin，会导致训练样本可以分对，但是泛化性能并不好的情况。尤其是对于Deep Learning 模型而言，由

于模型复杂度较高，大量的样本在训练后期正确类别的概率都会在0.99以上，这会导致大量的样本在训练后期回传的gradient非常小，从而不能更好地指导模型的训练。

所以为了拓展这个margin，根据前面的分析，主要有两个方向：**第一个方向是限定f的l2 norm的情况下让不同类之间的角度维持一个margin，这个方向上代表的两个工作是[1, 2]；第二个方向在第一个方向上更进一步，尽量让f的l2 norm变大[3]**。这分别对应的是中间和右侧的示意图。在中间的图中，可以看到如果我们在投影的夹角上加入一个margin，可以让每一类的样本push向正确类别的weight方向（蓝色箭头），最终都集中在过原点的一个锥中，这样可以显著改善类间和类内距离；最右侧的图中，我们更进一步，我们不仅仅希望在这样的一个锥中，还希望样本的特征表示f尽量远离原点（红色箭头），在这样一个锥形的“底部”。下面分别介绍下这两类工作的formulation。

在第一类工作中，以[1]举例，我们希望得到一个更严格的formulation，对于合适的m，我们希望满足：

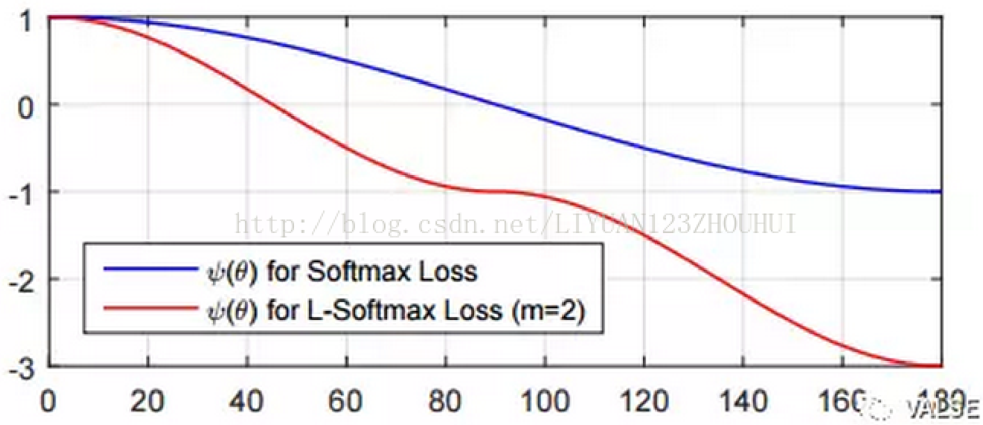
$$\|\boldsymbol{W}_1\|\|\boldsymbol{x}\|\cos(\theta_1) \geq \|\boldsymbol{W}_1\|\|\boldsymbol{x}\|\cos(m\theta_1) > \|\boldsymbol{W}_2\|\|\boldsymbol{x}\|\cos(\theta_2).$$

其对应的loss function为：

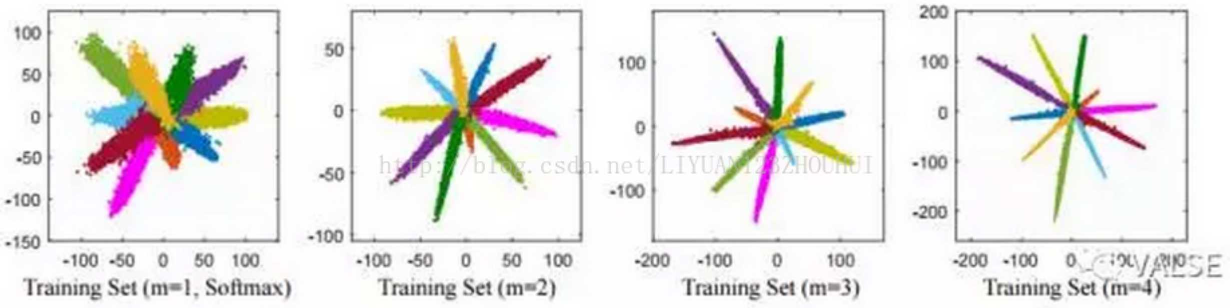
$$L_i = -\log\left(\frac{e^{\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|\psi(\theta_{y_i})}}{e^{\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|\psi(\theta_{y_i})} + \sum_{j\neq y_i} e^{\|\boldsymbol{W}_j\|\|\boldsymbol{x}_i\|\cos(\theta_j)}}\right) \quad (4)$$

$$\psi(\theta) = (-1)^k\cos(m\theta) - 2k, \quad \theta \in \left[\frac{k\pi}{m}, \frac{(k+1)\pi}{m}\right] \quad (6)$$

虽然看上去phi的定义比较复杂，原因在于cos函数并不是单调的，通过合适的phi定义是为了保障loss function是单调递减的。一个示例如下图



整个BP的过程就不再赘述，是很标准的求导操作。实验中，作者使用mnist做了一个很直观的illustration，相信大家看了也会一目了然



[2]在这个基础上，引入了对weight的normalization，提供了更好的理论分析，以及对open set样本性能的提升。

对于第二个方向，[3]给出了很好的探索。首先作者给出了理论证明，feature norm并不需要无限增长来保障泛化性能，所以在设计对应的regularization的时候，作者希望这个term的强度和feature norm的大小成反比。所以作者使用了下述的方式：

$$\underbrace{\lambda \frac{1}{N} \sum_{i=1}^{N} \frac{h(i)}{\|\mathbf{f}_i\|^2 + \epsilon}}_{\text{feature incay}}$$

[1]Liu, Weiyang, Yandong Wen, Zhiding Yu, and Meng Yang. “Large-Margin Softmax Loss for Convolutional Neural Networks.” InProceedings of The 33rd International Conference on Machine Learning, pp. 507-516. 2016.

[2] Liu, Weiyang, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. “SphereFace: Deep Hypersphere Embedding for Face Recognition.” arXiv preprint arXiv:1704.08063 (2017).

[3] Yuan, Yuhui, Yang Kuiyuan, Zhang Chao. “Feature Incay for Representation Regularization” arXiv preprint arXiv:1704.08063 (2017).

[4] Chunjie, Luo, and Yang Qiang. “Cosine Normalization: Using Cosine Similarity Instead of Dot Product in Neural Networks.” arXiv preprint arXiv:1702.05870 (2017).

[5] Ranjan, Rajeev, Carlos D. Castillo, and Rama Chellappa. “L2-constrained Softmax Loss for Discriminative Face Verification.” arXiv preprint arXiv:1703.09507 (2017).

[6] Wang, Feng, Xiang Xiang, Jian Cheng, and Alan L. Yuille. “NormFace: L_2 Hypersphere Embedding for Face Verification.” arXiv preprint arXiv:1704.06369 (2017).

[7] Liu, Yu, Hongyang Li, and Xiaogang Wang. “Learning deep features via congenerous cosine loss for person recognition.” arXiv preprint arXiv:1702.06890 (2017).

2.center loss

下面公式1中log函数的输入就是softmax的结果（是概率），而Ls表示的是softmax loss的结果（是损失）。wx+b是全连接层的输出，因此log的输入就表示xi属于类别yi的概率。

$$\mathcal{L}_S = -\sum_{i=1}^{m} \log \frac{e^{W_{y_i}^T \boldsymbol{x}_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T \boldsymbol{x}_i + b_j}} \tag{1}$$

In Equation 1, $\boldsymbol{x}_i \in \mathbb{R}^d$ denotes the ith deep feature, belonging to the y_ith class. d is the feature dimension. $W_j \in \mathbb{R}^d$ denotes the jth column of the weights $W \in \mathbb{R}^{d \times n}$ in the last fully connected layer and $\boldsymbol{b} \in \mathbb{R}^n$ is the bias term. The size of mini-batch and the number of class is m and n, respectively. We omit

那么center loss到底是什么呢？先看看center loss的公式LC。cyi表示第yi个类别的特征中心，xi表示全连接层之前的特征。后面会讲到实际使用的时候，m表示mini-batch的大小。因此这个公式就是希望一个batch中的每个样本的feature离feature 的中心的距离的平方和要越小越好，也就是类内距离要越小越好。这就是center loss。

$$\mathcal{L}_C = \frac{1}{2} \sum_{i=1}^{m} \|\boldsymbol{x}_i - \boldsymbol{c}_{y_i}\|_2^2$$

关于LC的梯度和cyi的更新公式如下：

The gradients of $\mathcal{L}_C$ with respect to $\boldsymbol{x}_i$ and update equation of $\boldsymbol{c}_{y_i}$ are computed as:

$$\frac{\partial \mathcal{L}_C}{\partial \boldsymbol{x}_i} = \boldsymbol{x}_i - \boldsymbol{c}_{y_i} \tag{3}$$

$$\Delta \boldsymbol{c}_j = \frac{\sum_{i=1}^{m} \delta(y_i = j) \cdot (\boldsymbol{c}_j - \boldsymbol{x}_i)}{1 + \sum_{i=1}^{m} \delta(y_i = j)} \tag{4}$$

这个公式里面有个条件表达式如下式，这里当condition满足的时候，下面这个式子等于1，当不满足的时候，下面这个式子等于0.

$$\delta(condition)$$

因此上面关于cyi的更新的公式中，当yi（表示yi类别）和cj的类别j不一样的时候，cj是不需要更新的，只有当yi和j一样才需要更新。

作者文中用的损失L的包含softmax loss和center loss，用参数lambda控制二者的比重，如下式所示。这里的m表示mini-batch的包含的样本数量，n表示类别数。

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_S + \lambda \mathcal{L}_C \\ &= -\sum_{i=1}^{m} \log \frac{e^{W_{y_i}^T \boldsymbol{x}_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T \boldsymbol{x}_i + b_j}} + \frac{\lambda}{2} \sum_{i=1}^{m} \|\boldsymbol{x}_i - \boldsymbol{c}_{y_i}\|_2^2\end{aligned}$$

Algorithm 1 The discriminative feature learning algorithm

Input: Training data $\{x_i\}$. Initialized parameters θ_C in convolution layers. Parameters W and $\{c_j | j = 1, 2, ..., n\}$ in loss layers, respectively. Hyperparameter λ, α and learning rate μ^t. The number of iteration $t \leftarrow 0$.

Output: The parameters θ_C.

while not converge **do**

$t \leftarrow t + 1$.

Compute the joint loss by $\mathcal{L}^t = \mathcal{L}^t_S + \mathcal{L}^t_C$.

Compute the backpropagation error $\frac{\partial \mathcal{L}^t}{\partial x^t_i}$ for each i by $\frac{\partial \mathcal{L}^t}{\partial x^t_i} = \frac{\partial \mathcal{L}^t_S}{\partial x^t_i} + \lambda \cdot \frac{\partial \mathcal{L}^t_C}{\partial x^t_i}$.

Update the parameters W by $W^{t+1} = W^t - \mu^t \cdot \frac{\partial \mathcal{L}^t}{\partial W^t} = W^t - \mu^t \cdot \frac{\partial \mathcal{L}^t_S}{\partial W^t}$.

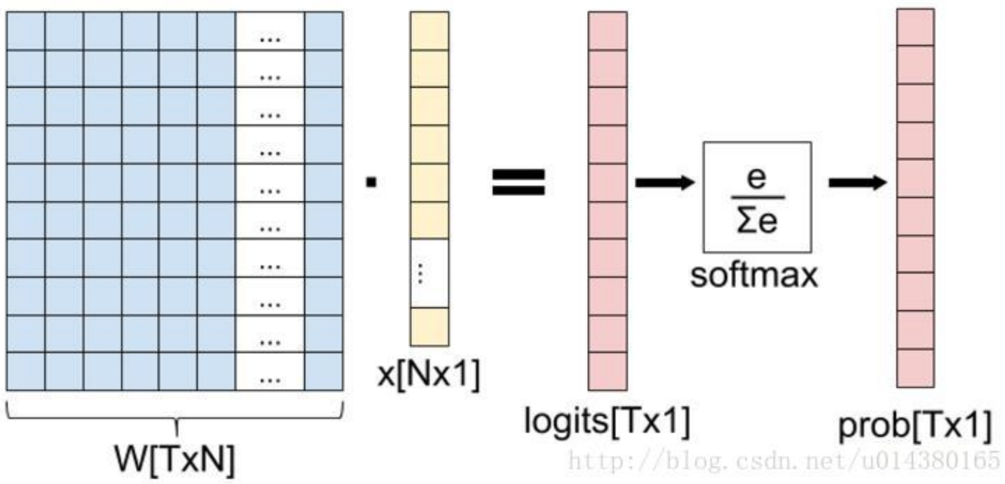
Update the parameters c_j for each j by $c^{t+1}_j = c^t_j - \alpha \cdot \Delta c^t_j$.

Update the parameters θ_C by $\theta^{t+1}_C = \theta^t_C - \mu^t \sum^m_i \frac{\partial \mathcal{L}^t}{\partial x^t_i} \cdot \frac{\partial x^t_i}{\partial \theta^t_C}$.

end while

3.由softmax到softmaxloss到cross entropy

这一篇主要介绍**全连接层和损失层**的内容，算是网络里面比较基础的一块内容。先理清下**从全连接层到损失层**之间的计算。来看下面这张图，来自参考资料1（自己实在懒得画图了）。



这张图的等号左边部分就是全连接层做的事，W是全连接层的参数，我们也称为权值，**X是全连接层的输入，也就是特征。**从图上可以看出特征X是N*1的向量，这是怎么得到的呢？**这个特征就是由全连接层前面多个卷积层和池化层处理后得到的**，假设全连接层前面连接的是一个卷积层，这个卷积层的输出是100个特征（也就是我们常说的feature map的channel为100），每个特征的大小是4*4，**那么在将这些特征输入给全连接层之前会将这些特征flat成N*1的向量（这个时候N就是100*4*4=1600）**。解释完X，再来看W，W是全连接层的参数，是个T*N的矩阵，这个N和X的N对应，**T表示类别数，比如你是7分类，那么T就是7。我们所说的训练一个网络，对于全连接层而言就是寻找最合适的W矩阵。**因此全连接层就是执行WX得到一个T*1的向量（也就是图中的logits[T*1]），这个向量里面的每个数都没有大小限制的，**也就是从负无穷大到正无穷大。**然后如果你是多分类问题，一般会在全连接层后面接一个softmax层，**这个softmax的输入是T*1的向量，输出也是T*1的向量（也就是图中的prob[T*1]，这个向量的每个值表示这个样本属于每个类的概率），只不过输出的向量的每个值的大小范围为0到1。**

现在你知道softmax的输出向量是什么意思了，就是概率，该样本属于各个类的概率！

那么softmax执行了什么操作可以得到0到1的概率呢？先来看看softmax的公式（以前自己看这些内容时候对公式也很反感，不过静下心来看就好了）：

$$S_j = \frac{e^{a_j}}{\sum_{k=1}^{T} e^{a_k}}$$

公式非常简单，前面说过softmax的输入是WX，假设模型的输入样本是I，讨论一个3分类问题（类别用1，2，3表示），样本I的真实类别是2，那么这个样本I经过网络所有层到达softmax层之前就得到了WX，也就是说WX是一个3*1的向量，那么上面公式中的aj就表示这个3*1的向量中的第j个值（最后会得到S1，S2，S3）；而分母中的ak则表示3*1的向量中的3个值，所以会有个求和符号（这里求和是k从1到T，T和上面图中的T是对应相等的，也就是类别数的意思，j的范围也是1到T）。因为e^x恒大于0，所以分子永远是正数，分母又是多个正数的和，所以分母也肯定是正数，因此Sj是正数，而且范围是(0,1)。**如果现在不是在训练模型，而是在测试模型，那么当一个样本经过softmax层并输出一个T*1的向量时，就会取这个向量中值最大的那个数的index作为这个样本的预测标签。**

因此我们训练全连接层的W的目标就是使得其输出的WX在经过softmax层计算后其对应于真实标签的预测概率要最高。

举个例子：假设你的WX=[1,2,3]，那么经过softmax层后就会得到[0.09,0.24,0.67]，这三个数字表示这个样本属于第1,2,3类的概率分别是0.09,0.24,0.67。

弄懂了softmax，就要来说说softmax loss了。
那softmax loss是什么意思呢?如下：

$$L = -\sum_{j=1}^{T} y_j log_{S_j}$$

首先L是损失。Sj是softmax的输出向量S的第j个值，前面已经介绍过了，**表示的是这个样本属于第j个类别的概率**。yj前面有个求和符号，j的范围也是1到类别数T，因此**y是一个1*T的向量，里面的T个值，而且只有1个值是1，其他T-1个值都是0。那么哪个位置的值是1呢？答案是真实标签对应的位置的那个值是1，其他都是0**。所以这个公式其实有一个更简单的形式：

$$L = -log_{S_j}$$

当然此时要限定j是指向当前样本的真实标签。

来举个例子吧。假设一个5分类问题，然后一个样本I的标签y=[0,0,0,1,0]，也就是说样本I的真实标签是4，假设模型预测的结果概率（**softmax的输出**）p=[0.1,0.15,0.05,**0.6**,0.1]，可以看出这个预测是对的，那么对应的损失L=-log(0.6)，**也就是当这个样本经过这样的网络参数产生这样的预测p时，它的损失是-log(0.6)**。那么假设p=[0.15,0.2,0.4,**0.1**,0.15]，这个预测结果就很离谱了，因为真实标签是4，而你觉得这个样本是4的概率只有0.1（远不如其他概率高，如果是在测试阶段，那么模型就会预测该样本属于类别3），对应损失L=-log(0.1)。那么假设p=[0.05,0.15,0.4,**0.3**,0.1]，这个预测结果虽然也错了，但是没有前面那个那么离谱，对应的损失L=-log(0.3)。我们知道log函数在输入小于1的时候是个负数，而且**log函数是递增函数，所以-log(0.6) < -log(0.3) < -log(0.1)。简单讲就是你预测错比预测对的损失要大，预测错得离谱比预测错得轻微的损失要大。**

理清了softmax loss，就可以来看看cross entropy了。
corss entropy是交叉熵的意思，它的公式如下：

$$E = -\sum_{j=1}^{T} y_i log_{P_j}$$

是不是觉得和softmax loss的公式很像。**当cross entropy的输入P是softmax的输出时，cross entropy等于softmax loss**。Pj是输入的概率向量P的第j个值，**所以如果你的概率是通过softmax公式得到的，那么cross entropy就是softmax loss**。这是我自己的理解，如果有误请纠正。