BPNet

> *Written by Zheng Jiacan, Shenzhen University, 2020/11/1.*

Basic Framework and Notation

假设网络有 $l+1$ 个节点层（包括输入和输出层），从左往右，依次为，第 0 层称为输入层 (input layer)，第 l 层称为输出层 (output layer)，中间的层称为隐藏层(hidden layer)。

当我们讨论第 k 层时，我们指的是从第 k-1 层节点到 第 k 层节点之间处理过程。各个层的节点数记为 n_k，特别的，有 $n_0 = d$，而 $n_l = r$.

对于第 k 层，其起点是一个输入特征矩阵，记之为 $\mathbf{Y}^{(k-1)}$，特别地，对于第一层，有 $\mathbf{Y}^{(0)} = \mathbf{X}$，其中 $\mathbf{X}$ 是原始的数据矩阵，其中每一行表示一个数据样本。然后是一个投影矩阵，记之为 $\mathbf{B}^{(k)} \in \mathbb{R}^{n_{k-1} \times n_k}$，接着是一个第 k 层节点的激活函数，一般是非线性函数，常用的有sigmod函数，ReLU函数，softmax函数，记之为 $g^{(k)}(\cdot)$，严格来讲，对于有sigmod函数，ReLU函数，$g^{(k)}(\cdot)$ 是一个 $\mathbb{R} \to \mathbb{R}$ 的映射，但是为了方便，当我们使用 $g^{(k)}(\mathbf{Z})$，其表示逐矩阵元素运算：

$$[g^{(k)}(\mathbf{Z})]_{ij} = g^{(k)}\left([\mathbf{Z}]_{ij}\right)$$

而对于 softmax函数，$g^{(k)}(\cdot)$ 是一个 $\mathbb{R}^d \to \mathbb{R}^d$ 的映射，但是为了方便，当我们使用 $g^{(k)}(\mathbf{Y})$，其表示逐矩阵行运算：

$$(g^{(k)}(\mathbf{Y}))_i = g^{(k)}\left(\mathbf{Y}_i\right)$$

第 k 层最后输出第 k 层节点的特征矩阵 $\mathbf{Y}^{(k)}$.

> *对于中间层的节点，可以选用 sigmoid 函数 或者 ReLU 函数。*
>
> *对于输出层的节点，根据输出值的特点，选用相应的函数作为输出层节点的激活函数，如果是二分类的，选用 sigmoid函数，多分类的选用 softmax 函数，只取非负值的选用 ReLU 函数。*

综上，对于每一层的传递，其公式为

$$\mathbf{Y}^{(k)}=g^{(k)}\left(\mathbf{Y}^{(k-1)}\mathbf{B}^{(k)}\right)=g^{(k)}\left(\mathbf{Z}^{(k)}\right)$$

整个神经网络的映射则为

$$\mathbf{Y}^{(l)}=g^{(l)}\left(g^{(l-1)}\left(\dots g^{(2)}\left(g^{(1)}\left(\mathbf{Y}^{(0)}\mathbf{B}^{(1)}\right)\mathbf{B}^{(2)}\right)\dots\right)\mathbf{B}^{(l)}\right)$$
$$\mathbf{Y}^{(l)}=f(\mathbf{X};\mathbf{B}^{(1)},\mathbf{B}^{(2)},\dots,\mathbf{B}^{(l)})=f(\mathbf{X};\mathbf{\Theta})$$

神经网路的目标在于学习上述映射，即在网络结构架构好，目标损失函数选择好的情况下，学习各个投影矩阵：

$$\mathbf{\Theta}=(\mathbf{B}^{(1)},\mathbf{B}^{(2)},\dots,\mathbf{B}^{(l)})$$

因此，这里有三个关键的问题：

- 网络架构的设计问题；
- 目标损失函数的设计问题；
- 投影矩阵的优化问题；

而 BP 神经网络对应的设计为

- 全连接网络结构；
- one-hot 编码，softmax 输出层，二者的最小二乘损失目标；
- 批量梯度下降算法；

Back Propagating Gradient Optimization

BP神经网络利用梯度下降法来最值化损失函数，因此其重点在于梯度表达式的推导。好的记号可以让大脑思考得更为高效，因此引入记号如下：

Notation

- $g^{(k)}$：第i层的激活函数。对于中间隐藏层来讲，其一般是ReLU及其变体，sigmoid函数、tanh函数（梯度饱和问题）等，而对于最后一层的输出层，往往需要根据建模数据的概率分布先验来决定。对于一个分类问题，如果先验分布是多项分布，那么一般采用softmax（二项分布情况下为sigmoid函数）；如果是回归问题，一般先验分布为高斯分布。
- $\mathbf{y}^{(k)}=(\mathbf{Y}_j^{(k)})^T$：第 k 层处理完成后的第 j 个样本的特征列向量
- $\mathbf{z}^{(k)}=(\mathbf{Z}_j^{(k)})^T$：第 k 层投影后的第 j 个样本的特征列向量
- $\mathbf{B}^{(k)}$ ：第 $(k-1)\to k$ 层的连接权重矩阵，第 j 列表示第 j 个节点的连接权值，可以形象地把 $\mathbf{B}^{(k)^T}$ 当是一个线性变换矩阵，这样求导时也方便理解。
- $J(\mathbf{Y}^{(l)};\mathbf{Y})$: 输出层的损失函数。

> *注意到，有时在第 k 层，我们想要考虑偏置的参数，为了数学表示的简洁性，我们把其集成到 $\mathbf{B}^{(k)}$ 的第一行中，即*
>
> $$\mathbf{B}_1^{(k)}=\mathbf{b}^{(k)}$$
>
> *相应地，我们需要在第 k 层的输入特征矩阵 $\mathbf{Y}^{(k-1)}$ 中，增加第一列全一向量:*

> $$\mathbf{Y}^{(k-1)} := [\mathbf{1}; \mathbf{Y}^{(k-1)}]$$
>
> *注意到，因为这里我们对$\mathbf{Y}^{(k-1)}$凭空添加了一列，因此在梯度的后向传播中，$\mathbf{Y}^{(k-1)}$*

前向传播的递推式

1. 线性层

$$\mathbf{Z}^{(k)} = \mathbf{Y}^{(k-1)}\mathbf{B}^{(k)}$$

2. 激活层

$$\mathbf{Y}^{(k)} = g^{(k)}(\mathbf{Z}^{(k)})$$

3. 损失函数层，其中$\mathbf{Y}^{(l)}$为最后的输出层，一般表示一个条件分布

$$J(\mathbf{Y}^{(l)}; \mathbf{H}) = \frac{1}{2}\|\mathbf{Y}^{(l)} - \mathbf{H}\|_F^2$$
$$J(\mathbf{Y}^{(l)}; \mathbf{H}) = -tr\left(\mathbf{H}^T \log \mathbf{Y}^{(l)}\right)$$

> *注意，为了方便，样本量规范化系数$\frac{1}{m}$可以暂时不考虑，留到最后计算梯度的时候再考虑进去即可。彼时只需要步长上，令$\alpha := \frac{1}{m}\alpha$即可。*

梯度推导

我们的目标是利用损失函数J和网络架构，导出损失函数关于各个投影矩阵$\mathbf{\Theta} = (\mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(l)})$的梯度，从而可以用来更新参数。这里参数是线性系数$\mathbf{B}^{(k)}$，因此我们需要导出

$$\frac{\partial J}{\partial \mathbf{B}^{(k)}}$$

从前向传播式中，我们发现前两个环节均是可以根据样本来分解的，即

$$\mathbf{z}^{(k)} = {\mathbf{B}^{(k)}}^T \mathbf{y}^{(k-1)}$$
$$\mathbf{y}^{(k)} = g^{(k)}(\mathbf{z}^{(k)})$$

这意味着在传播到$\mathbf{Y}_i^{(l)}$时，$\mathbf{Y}_i^{(l)}$只与参数$\mathbf{\Theta} = (\mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(l)})$和$\mathbf{Y}_i^{(0)}$有关。从而求

$$\frac{\partial \mathbf{y}^{(l)}}{\partial [\mathbf{B}^{(k)}]_j}$$

并不需要考虑其他样本。

对于两个损失函数，可以发现其是样本可加的，即

$$J(\mathbf{Y}^{(l)}; \mathbf{Y}) = \|\mathbf{Y}^{(l)} - \mathbf{Y}\|_F^2 = \sum_{i=1}^m \|\mathbf{Y}_i^{(l)} - \mathbf{Y}_i\|_F^2 = \sum_{i=1}^m J_i(\mathbf{y}^{(l)}; \mathbf{y})$$
$$J(\mathbf{Y}^{(l)}; \mathbf{Y}) = -tr\left(\mathbf{H}^T \log \mathbf{Y}^{(l)}\right) = \sum_{i=1}^m -\mathbf{H}_i^T \log \mathbf{Y}_i^{(l)} = \sum_{i=1}^m J_i(\mathbf{y}^{(l)}; \mathbf{y})$$

因此求导时也可以分解，即

$$\frac{\partial J}{\partial [\mathbf{B}^{(k)}]_j} = \sum_{\mathbf{y}^{(l)}} \frac{\partial J_i}{\partial \mathbf{y}^{(l)}} \frac{\partial \mathbf{y}^{(l)}}{\partial [\mathbf{B}^{(k)}]_j}$$

因此有

$$\frac{\partial J}{\partial \mathbf{B}^{(k)}} = \sum_{i=1}^{m} \frac{\partial J_i}{\partial \mathbf{B}^{(k)}}$$

对于第 k 层，我们记

- 损失函数 J 关于第 k 层中的 $\mathbf{y}^{(k)}$ 的导数（行向量，线性变换）

$$\mathbf{y}_G^{(k)} = \frac{\partial J}{\partial \mathbf{y}^{(k)}}$$

- 损失函数 J 关于第 l 层中的 $\mathbf{z}^{(k)}$ 的导数（行向量，线性变换）

$$\mathbf{z}_G^{(k)} = \frac{\partial J}{\partial \mathbf{z}^{(k)}}$$

- 损失函数 J 关于第 k 层中的 $\mathbf{B}^{(k)}$ 的导数（矩阵，线性变换）

$$\mathbf{B}_G^{(k)} = \frac{\partial J}{\partial \mathbf{B}^{(k)}}$$

- 激活函数$g^{(k)}$ 往往是逐位操作的，根据激活层公式 $\mathbf{y}^{(k)} = g^{(k)}(\mathbf{z}^{(k)})$，此时的雅各比矩阵 $\frac{\partial g^{[l]}}{\partial z^{[l]}}$ 是对角矩阵，我们将其提取为**对角行向量*，记之为 $g_G^{(k)}$（为了形式方便）

$$g_G^{(k)} = \mathbf{diag}\left(\frac{\partial g^{(k)}}{\partial \mathbf{z}^{(k)}}\right)$$

利用向量函数矩阵导数的法则，我们可以利用后一层的求导结果，可以推得前一层的导数。递推式如下：(将神经网络图形化)

- 损失函数 J 关于第 k 层中的 $\mathbf{y}^{(k-1)}$ 的导数（行向量），因为

$$\mathbf{z}^{(k)} = {\mathbf{B}^{(k)}}^T \mathbf{y}^{(k-1)}$$

所以

$$\frac{\partial \mathbf{z}^{(k)}}{\partial \mathbf{y}^{(k-1)}} = {\mathbf{B}^{(k)}}^T$$

利用链式法则，我们有

$$\frac{\partial J}{\partial \mathbf{y}^{(k-1)}} = \frac{\partial J}{\partial \mathbf{z}^{(k)}} \frac{\partial \mathbf{z}^{(k)}}{\partial \mathbf{y}^{(k-1)}}$$

从而得到

$$\mathbf{y}_G^{(k-1)} = \mathbf{z}_G^{(k)} {\mathbf{B}^{(k)}}^T$$

> *因为添加了偏置信息，所以 $\mathbf{y}^{(k-1)}$ 中的第一个元素多了一个 1，这个是凭空添加进去的，因此这个梯度是传递不到后续的参数更新的，因此需要将 $\mathbf{y}_G^{(k-1)}$ 的第一列去除。*

- 损失函数 J 关于第 l 层中的 $\mathbf{B}^{(k)}$ 的导数（ Jocobia 矩阵记法）

前向传播式为

$$\mathbf{z}^{(k)} = {\mathbf{B}^{(k)}}^T \mathbf{y}^{(k-1)}$$

因为这是一个矩阵到向量的函数，因此它的导数是不好表示的，但是我们可以写成

$$\mathbf{z}^{(k)^T} = \mathbf{y}^{(k-1)^T}\mathbf{B}^{(k)}$$

从而发现

$$\frac{\partial \mathbf{z}^{(k)}}{\partial [\mathbf{B}^{(k)}]_j} = \begin{bmatrix} 0 \\ \vdots \\ \mathbf{y}^{(k-1)^T} \\ \vdots \\ 0 \end{bmatrix} = \mathbf{e}_j \mathbf{y}^{(k-1)^T}$$

利用链式法则，我们有

$$\frac{\partial J}{\partial [\mathbf{B}^{(k)}]_j} = \frac{\partial J}{\partial \mathbf{z}^{(k)}} \frac{\partial \mathbf{z}^{(k)}}{\partial [\mathbf{B}^{(k)}]_j} = \mathbf{z}_G^{(k)} \mathbf{e}_j \mathbf{y}^{(k)^T} = [\mathbf{z}_G^{(k)}]_j \mathbf{y}^{(k-1)^T}$$

将上式推广到整个矩阵 $\mathbf{B}^{(k)}$ 上，可以得到

$$\begin{aligned} &\frac{\partial J}{\partial \mathbf{B}^{(k)}} \\ =&\begin{bmatrix} \frac{\partial J}{\partial [\mathbf{B}^{(k)}]_1} \\ \vdots \\ \frac{\partial J}{\partial [\mathbf{B}^{(k)}]_{n_k}} \end{bmatrix} \\ =&\begin{bmatrix} [\mathbf{z}_G^{(k)}]_1 \mathbf{y}^{(k-1)^T} \\ \vdots \\ [\mathbf{z}_G^{(k)}]_{n_k} \mathbf{y}^{(k-1)^T} \end{bmatrix} \\ =&\mathbf{z}_G^{(k)^T} \mathbf{y}^{(k-1)^T} \end{aligned}$$

从而得到

$$\mathbf{B}_G^{(k)} = \mathbf{z}_G^{(k)^T} \mathbf{y}^{(k-1)^T}$$

- 损失函数 J 关于第 k 层中的 $\mathbf{z}^{(k)}$ 的导数（行向量），因为

$$\mathbf{y}^{(k)} = g^{(k)}(\mathbf{z}^{(k)})$$

所以

$$\frac{\partial \mathbf{y}^{(k)}}{\partial \mathbf{z}^{(k)}} = \frac{\partial g^{(k)}}{\partial \mathbf{z}^{(k)}}$$

利用链式法则，我们有

$$\frac{\partial J}{\partial \mathbf{z}^{(k)}} = \frac{\partial J}{\partial \mathbf{y}^{(k)}} \frac{\partial \mathbf{y}^{(k)}}{\partial \mathbf{z}^{(k)}} = \mathbf{y}_G^{(k)} \frac{\partial g^{(k)}}{\partial \mathbf{z}^{(k)}}$$

由于激活函数 $g^{(k)}$ 往往是逐位操作的，此时的雅各比矩阵 $\frac{\partial g^{[l]}}{\partial z^{[l]}}$ 是对角矩阵，因此相乘可以变成点乘对角行向量，从而得到

$$\mathbf{z}_G^{(k)} = \mathbf{y}_G^{(k)} \odot g_G^{(k)}$$

> *注意到，如果在 第k+1 层我们用了偏置，那么 $\mathbf{y}^{(k)}$ 会在经过 $\mathbf{y}^{(k)} = g^{(k)}(\mathbf{z}^{(k)})$ 处理之后多了一个值为 1 的第一个分量元素，在后向求导时，这个分量元素是与 $\mathbf{z}^{(k)}$ 无关的，因此求导时要考虑到这一点，也就是要去除掉 $\mathbf{y}_G^{(k)}$ 行向量的第一列，即把*
>
> $$\mathbf{y}_G^{(k)} = \mathbf{z}_G^{(k+1)} \mathbf{B}^{(k+1)^T}$$
>
> *对其中 $\mathbf{B}^{(k+1)}$ 的第一行去除，得到*
>
> $$\mathbf{y}_G^{(k)} = \mathbf{z}_G^{(k+1)} \mathbf{B}^{(k+1)^T}_{[-1]}$$

> *因此等价于将* $\mathbf{y}_G^{(k-1)}$ *的第一列去除。*
>
> *然后再得到公式*
>
> $$\mathbf{z}_G^{(k)} = \mathbf{y}_G^{(k)} \odot g_G^{(k)}$$

上述三个式子称为后向传播递推式，其总结如下

1. $$\mathbf{y}_G^{(k-1)} = \mathbf{z}_G^{(k)} \mathbf{B}^{(k)^T}$$
2. $$\mathbf{B}_G^{(k)} = \mathbf{z}_G^{(k)^T} \mathbf{y}^{(k-1)^T}$$
3. $$\mathbf{z}_G^{(k-1)} = \mathbf{y}_G^{(k-1)} \odot g_G^{(k-1)}$$

这三个式子告诉我们，只要求出第 l 层的梯度 $\mathbf{z}_G^{(l)}$，则后续的梯度均可以通过三个后向传播式计算。

三个常见的激活函数及其导数

- ReLU

  $$g(x) = \max\{x, 0\}$$
  $$g_G(x) = \mathbf{1}(x > 0)$$

- sigmoid

  $$g(x) = \frac{\exp(x)}{1 + \exp(x)}$$
  $$g_G(x) = \frac{\exp(x)}{1 + \exp(x)} \frac{1}{1 + \exp(x)} = g(x)(1 - g(x))$$

- tanh

  $$g(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$
  $$g_G(x) = 1 - g(x)^2$$

  tanh 可以看成是 sigmoid 的拉伸版本，将值域从 $(0, 1)$ 拉伸至 $(-1, 1)$.

  $$\frac{2\exp(x)}{1 + \exp(x)} - 1 = \frac{\exp(x) - 1}{\exp(x) + 1}$$

  令 $x := 2x$，即将 x 轴拉伸 2 倍 （不影响值域），得到

  $$\frac{\exp(2x) - 1}{\exp(2x) + 1} = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$

  因此，对于tanh 和 sigmoid，有如下关系

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

  从而其导数为

$$\begin{aligned}\tanh^{'}(x)=&4\sigma^{'}(2x)\\=&4\sigma(2x)(1-\sigma(2x))\\=&4\left[\frac{1}{2}(\tanh(x)+1)\right]\left[\frac{1}{2}(1-\tanh(x))\right]\\=&(1+\tanh(x))(1-\tanh(x))\\=&1-\tanh^2(x)\end{aligned}$$

注意，对于 $\tanh(x)$，我们有

$$\lim_{\alpha\to+\infty}\tanh(\alpha x)=\operatorname{sgn}(x),\quad \forall x\neq 0$$

因此这个函数在 **HashNet** 被用来近似符号函数。

回归网络中的loss function 及 输出层导数

记回归矩阵为 $\mathbf{H}$，特别地，在自回归（自编码）任务中，$\mathbf{H}=\mathbf{X}$. 对于其损失函数，最直接地，可以通过输出值与样本标签值的**距离**定义损失函数，当我们采用欧式距离时，损失函数为：（均方误差）（最小二乘）

$$J(\mathbf{Y}^{(l)};\mathbf{Y})=\frac{1}{2m}\sum_{i=1}^{m}(y^{(i)}-\hat{y}^{(i)})^2$$

$$J(\mathbf{Y}^{(l)};\mathbf{H})=\frac{1}{2}\|\mathbf{Y}^{(l)}-\mathbf{H}\|_F^2$$

从而

$$\frac{\partial J(\mathbf{Y}^{(l)};\mathbf{H})}{\partial\mathbf{Y}^{(l)}}=\mathbf{Y}^{(l)}-\mathbf{H}$$

$$\mathbf{y}_G^{(l)}=\frac{\partial J(\mathbf{y}^{(l)};\mathbf{h})}{\partial\mathbf{y}^{(l)}}=(\mathbf{y}^{(l)}-\mathbf{h})^T$$

由于回归任务中，一般回归的范围是整个实数域，因此 **最后一层激活函数** 可以设置为 identical function，即

$$\mathbf{Y}^{(l)}=g^{(l)}\left(\mathbf{Z}^{(l)}\right)=\mathbf{Z}^{(l)}$$

在这样的设置下，则有

$$\mathbf{z}_G^{(l)}=\mathbf{y}_G^{(l)}=(\mathbf{y}^{(l)}-\mathbf{y})^T$$

分类网络中的loss function 及 输出层导数

记有 r 个类别，概率标签矩阵为 $\mathbf{H}$，则输出矩阵 $\mathbf{Y}^{(l)}$ 一般是一个离散概率分布列矩阵，那么可以根据似然原理或 KL 散度定义损失函数：

$$J(\mathbf{Y}^{(l)};\mathbf{Y})=\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{k}y_j^{(i)}\log(\hat{y}_j^{(i)})$$

$$J(\mathbf{Y}^{(l)};\mathbf{H})=-tr\left(\mathbf{H}^T\log\mathbf{Y}^{(l)}\right)=-\left\langle\mathbf{H},\log\mathbf{Y}^{(l)}\right\rangle$$

从而有

$$J(\mathbf{y}^{(l)};\mathbf{h}) = -\mathbf{h}^T \log \mathbf{y}^{(l)}$$
$$\frac{\partial J(\mathbf{y}^{(l)};\mathbf{h})}{\partial \mathbf{y}^{(l)}} = -\left(\mathbf{h} \odot \frac{1}{\mathbf{y}^{(l)}}\right)^T$$

其中

$$\left[\frac{1}{\mathbf{Y}^{(l)}}\right]_{ij} = \frac{1}{\left[\mathbf{Y}^{(l)}\right]_{ij}}$$

从而得到

$$\mathbf{y}_G^{(l)} = -\mathbf{h}^T \frac{1}{\mathbf{y}^{(l)}}$$

而对于 $\mathbf{z}_G^{(l)}$，我们知道有

$$\mathbf{z}_G^{(l)} = \mathbf{y}_G^{(l)} \frac{\partial g^{(l)}}{\partial \mathbf{z}^{(l)}}$$

由于分类任务中，有效取值为 [0 , 1] 区间，因此 **最后一层激活函数** 可以设置为 softmax 激活函数，其定义为

$$\mathbf{y}_j^{(l)} = [g^{(l)}(\mathbf{z}^{(l)})]_j = \frac{\exp\left(\mathbf{z}_j^{(l)}\right)}{\sum_{t=1}^{r} \exp\left(\mathbf{z}_t^{(l)}\right)}$$

所以，对于 $i \neq j$，有

$$\begin{aligned}
\frac{\partial \mathbf{y}_j^{(l)}}{\partial \mathbf{z}_i^{(l)}} =& \frac{0 - \exp\left(\mathbf{z}_j^{(l)}\right) \exp\left(\mathbf{z}_i^{(l)}\right)}{\left[\sum_{t=1}^{r} \exp\left(\mathbf{z}_t^{(l)}\right)\right]^2} \\
=& -\frac{\exp\left(\mathbf{z}_j^{(l)}\right)}{\sum_{t=1}^{r} \exp\left(\mathbf{z}_t^{(l)}\right)} \frac{\exp\left(\mathbf{z}_i^{(l)}\right)}{\sum_{t=1}^{r} \exp\left(\mathbf{z}_t^{(l)}\right)} \\
=& \mathbf{y}_j^{(l)}(-\mathbf{y}_i^{(l)})
\end{aligned}$$

而对于 $i = j$，有

$$\begin{aligned}
\frac{\partial \mathbf{y}_j^{(l)}}{\partial \mathbf{z}_j^{(l)}} =& \frac{\exp\left(\mathbf{z}_j^{(l)}\right)\left[\sum_{t=1}^{r} \exp\left(\mathbf{z}_t^{(l)}\right)\right] - \exp\left(\mathbf{z}_j^{(l)}\right) \exp\left(\mathbf{z}_j^{(l)}\right)}{\left[\sum_{t=1}^{r} \exp\left(\mathbf{z}_t^{(l)}\right)\right]^2} \\
=& \frac{\exp\left(\mathbf{z}_j^{(l)}\right)\left[\sum_{t\neq j}^{r} \exp\left(\mathbf{z}_t^{(l)}\right)\right]}{\left[\sum_{t=1}^{r} \exp\left(\mathbf{z}_t^{(l)}\right)\right]^2} \\
=& \frac{\exp\left(\mathbf{z}_j^{(l)}\right)}{\sum_{t=1}^{r} \exp\left(\mathbf{z}_t^{(l)}\right)} \frac{\sum_{t\neq j}^{r} \exp\left(\mathbf{z}_t^{(l)}\right)}{\sum_{t=1}^{r} \exp\left(\mathbf{z}_t^{(l)}\right)} \\
=& \frac{\exp\left(\mathbf{z}_j^{(l)}\right)}{\sum_{t=1}^{r} \exp\left(\mathbf{z}_t^{(l)}\right)} \left(1 - \frac{\exp\left(\mathbf{z}_j^{(l)}\right)}{\sum_{t=1}^{r} \exp\left(\mathbf{z}_t^{(l)}\right)}\right) \\
=& \mathbf{y}_j^{(l)}(1 - \mathbf{y}_j^{(l)})
\end{aligned}$$

综上，得到

$$\frac{\partial \mathbf{y}_j^{(l)}}{\partial \mathbf{z}^{(l)}} = \mathbf{y}_j^{(l)}(\mathbf{e_j} - \mathbf{y}^{(l)})^T$$

从而

$$\frac{\partial \mathbf{y}^{(l)}}{\partial \mathbf{z}^{(l)}} = \frac{\partial g^{(l)}}{\partial \mathbf{z}^{(l)}} = (\mathbf{y}^{(l)}\mathbf{1}^T) \odot \left(\mathbf{I} - \mathbf{1}{\mathbf{y}^{(l)}}^T\right)$$

因此

$$\begin{aligned}
\mathbf{z}_G^{(l)} =& \mathbf{y}_G^{(l)} \frac{\partial g^{(l)}}{\partial \mathbf{z}^{(l)}} \\
=& -\left(\mathbf{h} \odot \frac{1}{\mathbf{y}^{(l)}}\right)^T \left[(\mathbf{y}^{(l)}\mathbf{1}^T) \odot \left(\mathbf{I} - \mathbf{1}{\mathbf{y}^{(l)}}^T\right)\right] \\
=& -\mathbf{diag}\left(\left(\left[\mathbf{h} \odot \frac{1}{\mathbf{y}^{(l)}}\mathbf{1}^T\right] \odot (\mathbf{y}^{(l)}\mathbf{1}^T)\right)^T \left(\mathbf{I} - \mathbf{1}{\mathbf{y}^{(l)}}^T\right)\right)^T \\
=& -\mathbf{diag}\left(\mathbf{1}\mathbf{h}^T \left(\mathbf{I} - \mathbf{1}{\mathbf{y}^{(l)}}^T\right)\right)^T \\
=& -\mathbf{h}^T \left(\mathbf{I} - \mathbf{1}{\mathbf{y}^{(l)}}^T\right) \\
=& {\mathbf{y}^{(l)}}^T - \mathbf{h}^T \\
=& \left(\mathbf{y}^{(l)} - \mathbf{h}\right)^T
\end{aligned}$$

> *上式中，我们利用到了*
>
> $$\begin{aligned}
> tr[A^T(B \odot C)] =& tr[(B \odot C)^T A] \\
> =& \langle A, B \odot C \rangle \\
> =& \langle B, A \odot C \rangle \\
> =& tr[(A \odot C)^T B] \\
> =& \langle C, A \odot B \rangle \\
> =& tr[(A \odot B)^T C]
> \end{aligned}$$
>
> *以及 (a 为列向量)*
>
> $$a^T(B \odot C) = \mathbf{diag}[(a\mathbf{1}^T \odot B)^T C]^T$$
>
> *注意到有*
>
> $$\mathbf{diag}(\mathbf{h}\mathbf{1}^T) = \mathbf{h}$$

以上可以发现，回归网络 和 分类网络的 $\mathbf{z}_G^{(l)}$ 的公式是一致的。

经典的一些 loss function

在有标签数据的情况下，softmax 是一个非常常用的损失函数，但是它也有一些缺陷：

> *Softmax loss based network can converges quickly when training, however, as a classification network, softmax loss based network can not explicitly optimize the **interclass and intra-class distance** like metric learning, so the performance is not **particularly good and robust**. In addition, since it doesn't optimize the hidden feature layer, the directly trained features generally do not have good generalization ability.*

> *softmax 损失函数只关注网络最后输出矩阵与标签矩阵的拟合度，对于网络学习到的特征，softmax 是没有进行约束控制的，它没有显地优化类内散度和类间散度，从而使得分类表现并不是特别好和鲁棒性。由于没有对学习到特征做约束学习，因此泛化能力也会下降。*
>
> *下面介绍的loss 基本上承接的是传统模型的思路，也就是LDA 的类内散度和类间散度，以及 近邻保持 目标函数的东西。*

Contrastive Loss

> *LDA 的类间和类内散度思想 + 近邻保持的形式*

The contrastive loss is mainly used for dimensionality reduction, that is, two samples that are originally similar are still similar in the feature space after dimensionality reduction (feature extraction), and that originally dissimilar samples are still dissimilar in the feature space. Similarly, contrastive loss function can also express the degree of matching of pairs of samples.

Contrastive Loss Function (对比损失函数)，可以有效处理成对数据标签的特征学习问题。利用数据标签矩阵 $\mathbf{H}$，我们可以先生成一个 affinity matrix $\mathcal{W} \in \mathbb{R}^{m \times m}$，其中 $[\mathcal{W}]_{ij} = 0$ 如果 i, j 样本是同一类的，反之则为1，该矩阵有解析形式，为

$$\mathcal{W} = \mathbf{H}\mathbf{H}^T$$

有了这个矩阵，就可以设计目标函数。对于分类任务而言，我们希望同类数据相吸引，异类数据相排斥，从而对于 $\mathbf{x}_i$ 和 $\mathbf{x}_j$，假设学习到的特征为 $\mathbf{y}_i$ 和 $\mathbf{y}_j$，则我们有两个目标：

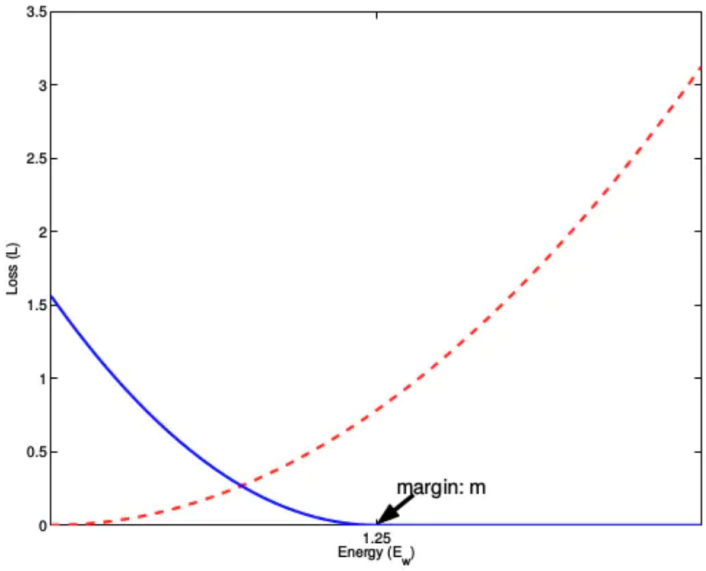
- if $[\mathcal{W}]_{ij} = 1$，then $\min \quad \mathcal{D}^2(\mathbf{y}_i, \mathbf{y}_j)$;
- if $[\mathcal{W}]_{ij} = 0$，then $\max \quad \mathcal{D}^2(\mathbf{y}_i, \mathbf{y}_j)$; 由于这是一个非凸问题，结合 SVM 的合页损失函数，我们将其修改为

  $$\min \quad \{\max[0, M - \mathcal{D}(\mathbf{y}_i, \mathbf{y}_j)]\}^2$$

  也就是如果异类点的距离大于阈值 M，则不做惩罚，如果小于 M，则做惩罚;

综上，把两个目标结合到一块，则可以得到

$$\min_{\mathbf{y}_i, \mathbf{y}_j} \quad \mathcal{D}^2(\mathbf{y}_i, \mathbf{y}_j)[\mathcal{W}]_{ij} + \{\max[0, M - \mathcal{D}(\mathbf{y}_i, \mathbf{y}_j)]\}^2(1 - [\mathcal{W}]_{ij})$$



> 这张图表示的就是损失函数值与样本特征的欧式距离之间的关系，其中红色虚线表示的是相似样本的损失值，蓝色实线表示的不相似样本的损失值。

将所有样本考虑进来，则目标函数为

$$\min_{\mathbf{y}_i,\mathbf{y}_j} \quad \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\left[\mathcal{D}^2(\mathbf{y}_i,\mathbf{y}_j)[\mathcal{W}]_{ij}+\{\max[0,M-\mathcal{D}(\mathbf{y}_i,\mathbf{y}_j)]\}^2(1-[\mathcal{W}]_{ij})\right]$$

假设网络层得到的特征矩阵为 $\mathbf{Y}^{(l)}$，其两两距离矩阵为 $\mathcal{D}$，则目标函数的矩阵形式为

$$\min_{\mathbf{Y}^{(l)}} \quad \frac{1}{2}\left[\langle\mathcal{D}^2,W\rangle+\langle\{\max(0,M-\mathcal{D})\}^2,1-\mathcal{W}\rangle\right]$$

如果选择欧式距离，则可以先利用坐标矩阵导出内积矩阵，再利用内积矩阵导欧式距离矩阵。

$$\mathcal{B}=\mathbf{Y}^{(l)}\mathbf{Y}^{(l)^T}$$
$$[\mathcal{D}]_{ij}=[\mathcal{B}]_{ii}+[\mathcal{B}]_{jj}-2[\mathcal{B}]_{ij}$$

该函数的求导是比较关键的，即我们需要导出

$$\mathbf{Y}_G^{(l)}=\frac{\partial J}{\partial\mathbf{Y}^{(l)}}$$

Triplet Loss

> *Triplet loss considers the relative difference in distance between samples, which is the opposite of the contrastive loss considering the absolute distance of the non-matching pairs and the matching pairs.*
>
> *Triplet loss requires the face image triplets that consist of an anchor sample, a negative sample and a positive sample. In the face image triplets, the anchor sample and the positive sample belong to the same entity, while the anchor sample and the negative sample belong to different entities.*

> *LDA 的类间和类内散度思想.*

选择一个 anchor sample，记之为 $\mathbf{x}_a$，其正样本记之为 $\mathbf{x}_p$，负样本记之为 $\mathbf{x}_n$，则对于这个三样本，学习的目标为

$$\min_{\mathbf{y}_a,\mathbf{y}_p,\mathbf{y}_n} \quad \max\left(\mathcal{D}^2(\mathbf{y}_a,\mathbf{y}_p)-\mathcal{D}^2(\mathbf{y}_a,\mathbf{y}_n)+M,0\right)$$

可以看到，这个问题的最优解为

$$\mathcal{D}^2(\mathbf{y}_a,\mathbf{y}_p)=0,\quad \mathcal{D}^2(\mathbf{y}_a,\mathbf{y}_n)=M$$

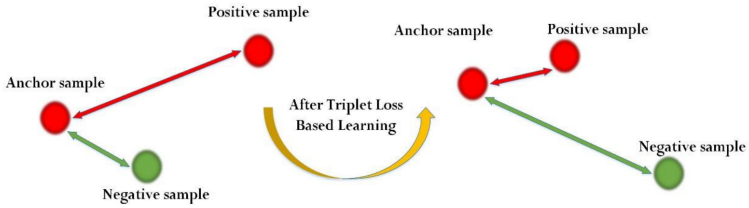


Fig. 9. The learning process of the triplet loss. The distance between the anchor sample and the positive sample is closer and the distance between the anchor sample and the negative sample is farther by learning.

> *However, the triplet loss converges slowly and sometimes does not converge.*

Center Loss

> *However, the performance obtained only by using the contrastive loss or triplet loss is not good. To achieve better recognition performance,* ***Wen et al. added a center loss to maintain the category center on the basis of softmax loss****. This method can gather features to the center of the category, thus achieving the effect similar to triple loss. Center loss learns a center for each category and pulls all feature vectors for each category toward the center of the corresponding category, which makes the intra-class distance smaller.*

Center Loss 是一个项，其为

$$\mathcal{L}_C=\frac{1}{2}\sum_{i=1}^{m}\|\mathbf{y}_i-\mathcal{C}_k\|_2^2$$

其中 $\mathcal{C}_k=\left([\mathbf{H}]_k{}^T[\mathbf{H}]_k\right)^{-1}[\mathbf{H}]_k{}^T\mathbf{Y}$.

标准矩阵记法下后向传播公式

对于目标梯度值，有

$$\frac{\partial J}{\partial \mathbf{B}^{(k)}}=\sum_{i=1}^{m}\frac{\partial J_i}{\partial \mathbf{B}^{(k)}}$$

对于分类和回归网络，均有

$$\mathbf{z}_G^{(l)}=\left(\mathbf{y}^{(l)}-\mathbf{h}\right)^T$$

写成标准矩阵的形式，则为

$$\mathbf{Z}_G^{(l)} = \mathbf{Y}^{(l)} - \mathbf{H}$$

对于 $\mathbf{y}_G^{(k-1)} = \mathbf{z}_G^{(k)} \mathbf{B}^{(k)^T}$，其标准矩阵记法为

$$\mathbf{Y}_G^{(k-1)} = \mathbf{Z}_G^{(k)} \mathbf{B}^{(k)^T}$$

而对于 $\mathbf{B}_G^{(k)} = \mathbf{z}_G^{(k)^T} \mathbf{y}^{(k-1)^T}$，其对应的标准矩阵记法为

$$\mathbf{B}_G^{(k)} = \mathbf{Z}_G^{(k)^T} \mathbf{Y}^{(k-1)}$$

而对于 $\mathbf{z}_G^{(k-1)} = \mathbf{y}_G^{(k-1)} \odot g_G^{(k-1)}$，其标准矩阵记法为

$$\mathbf{Z}_G^{(k-1)} = \mathbf{Y}_G^{(k-1)} \odot \mathbf{G}_G^{(k-1)}$$

其中

$$\mathbf{G}_G^{(k-1)} = \begin{bmatrix} g_G^{(k-1)}(\mathbf{Y}_1^{(k-1)}) \\ \vdots \\ g_G^{(k-1)}(\mathbf{Y}_i^{(k-1)}) \\ \vdots \\ g_G^{(k-1)}(\mathbf{Y}_m^{(k-1)}) \end{bmatrix}$$

综上，标准矩阵记法下后向传播公式为

- $$\mathbf{B}_G^{(k)} = \mathbf{Z}_G^{(k)^T} \mathbf{Y}^{(k-1)}$$
- $$\mathbf{Y}_G^{(k-1)} = \mathbf{Z}_G^{(k)} \mathbf{B}^{(k)^T}$$
- $$\mathbf{Z}_G^{(k-1)} = \mathbf{Y}_G^{(k-1)} \odot \mathbf{G}_G^{(k-1)}$$

> $\mathbf{Z}_G^{(k)}$ *和* $\mathbf{Y}_G^{(k)}$ *和*$\mathbf{B}_G^{(k)}$ *均采用Jocobia 的记法。*
>
> *后向传播时，* $\mathbf{Y}_G^{(k-1)}$ *的第一列要去除。等价于将*$\mathbf{B}^{(k)}$ *的第一行（偏置行）去除，得到*
>
> $$\mathbf{Y}_G^{(k-1)} = \mathbf{Z}_G^{(k)} \mathbf{B}^{(k)^T}_{[-1]}$$
>
> *然后再计算* $\mathbf{G}_G^{(k-1)}$ *时，输入* $\mathbf{Y}^{(k-1)}$ *时也要把第一列的全一向量去除。*

初始启动公式为

$$\mathbf{Z}_G^{(l)} = \mathbf{Y}^{(l)} - \mathbf{H}$$

> *梯度更新公式：*
>
> *令* $\alpha := \frac{1}{m}\alpha$，$\mathbf{B}^{(k)} := \mathbf{B}^{(k)} - \alpha\, \mathbf{B}_G^{(k)^T}$.
>
> *注意，如果使用sigmoid 顶层激活函数作为二分类网络的输出函数，那么此时的标签矩阵只有一列，为*$\mathbf{h}$*，输出也只有一列，为*$\mathbf{y}^{(l)}$*，可以证明，其导数仍然为（与softmax 激活函数形式保持一致.）*

> $$\mathbf{z}_G^{(l)} = \mathbf{y}^{(l)} - \mathbf{h}$$

在标准记法下，前向传播的公式为

1. 线性层
$$\mathbf{Z}^{(k)} = \mathbf{Y}^{(k-1)}\mathbf{B}^{(k)}$$
2. 激活层
$$\mathbf{Y}^{(k)} = g^{(k)}(\mathbf{Z}^{(k)})$$
3. 损失函数层，其中 $\mathbf{Y}^{(l)}$ 为最后的输出层，一般表示一个条件分布
$$J(\mathbf{Y}^{(l)};\mathbf{H}) = \frac{1}{2}\|\mathbf{Y}^{(l)} - \mathbf{H}\|_F^2$$
$$J(\mathbf{Y}^{(l)};\mathbf{H}) = -tr\left(\mathbf{H}^T \log \mathbf{Y}^{(l)}\right)$$

> *初始化时，$\mathbf{B}^{(k)}$ 第一行插入一行初始化为全零的偏置向量，前向传播时，$\mathbf{Y}_G^{(k-1)}$ 第一列插入一个全一向量。*

初始启动公式为

$$\mathbf{Z}^{(1)} = \mathbf{Y}^{(0)}\mathbf{B}^{(1)}$$

Traning Remark

1. 充分利用矢量化计算的速度

   BPNet 的训练中涉及到比较多的计算，直接利用程序的循环结构来实现迭代是比较慢的，因此需要尽可能地利用矩阵乘法和一些矢量化计算，因此这些函数在python 的 numpy 库中是经过优化过的，可以使得训练速度提高。

2. 解决指数溢出问题
   sigmoid 与 softmax 均有指数计算 e^x，这很容易导致过大溢出，但是可以通过简单的方法完美避开。
   对于 softmax 函数，选 $M = \max\{x_i\}$，则有
   $$\frac{e^{x_j}}{1+\sum_{i=1}^{k-1}e^{x_i}} = \frac{e^{x_j-M}}{e^{-M}+\sum_{i=1}^{k-1}e^{x_i-M}}$$
   此时分子小于等于1，而分母大于1。

   对于 sigmoid 函数，他其实是softmax的二维情况，但他有一个更合适的方法：

   - 当$x>0$，利用$\frac{1}{1+e^{-x}}$计算；
   - 当$x<0$，利用$\frac{e^x}{1+e^x}$ 计算。

   对于 tanh 函数，可以直接利用其与 sigmoid 函数的关系计算
   $$\tanh(x) = 2\sigma(2x) - 1$$

3. 参数的初始化问题

   参数的初始化，截距项可以初始化为0，而连接权值的初始化有多种策略，在理想状态下，网络的参数在收敛状态下应该保持正负各半的情况，但是初始化时让每一层的矩阵元素均值为0. 注意到全部初始化为0是不适合的。

   一种策略是，让每一层的矩阵元素初始化为一个均值为0，方差较小的分布中的元素，比如高斯分布或者均匀分布。

为了使得训练更加有效和稳定，我们需要对网络每一层的输出的神经元的方差进行规范化。考虑第 k 层网络，对于输入的 $\mathbf{y}$，其输出 $\mathbf{z}$ 的方差应该不变：

$$\mathbf{z} = \mathbf{y}\mathbf{B}, \quad \mathbf{B} \in \mathbb{R}^{n_{k-1} \times n_k}$$

假设 $\mathbf{y}$ 的各个维度独立，其均值为0，且方差相同，记之为 $Var(\mathbf{y}) = Var(\mathbf{y}_i)$，然后假设 $\mathbf{B}$ 与 $\mathbf{y}$ 是独立的，且 $\mathbf{B}$ 的均值也为0，方差相同，记之为 $Var(\mathbf{B}) = Var(\mathbf{B}_{ij})$.

利用二者的独立性，有

$$E(\mathbf{y}_i([\mathbf{B}]_j)_i) = E(\mathbf{y}_i)E(([\mathbf{B}]_j)_i)$$

从而可以得到两个变量乘积的方差为：

$$\begin{aligned} Var(\mathbf{y}_i([\mathbf{B}]_j)_i) =& E(\mathbf{y}_i^2([\mathbf{B}]_j)_i^2) - E(\mathbf{y}_i([\mathbf{B}]_j)_i)^2 \\ =& E(\mathbf{y}_i^2)E(([\mathbf{B}]_j)_i^2) - E(\mathbf{y}_i)^2E(([\mathbf{B}]_j)_i)^2 \\ =& E(\mathbf{y}_i^2)E(([\mathbf{B}]_j)_i^2) \\ =& Var(\mathbf{y}_i)Var(([\mathbf{B}]_j)_i) \\ =& Var(\mathbf{y})Var(\mathbf{B}) \end{aligned}$$

因此，对于输出的 $\mathbf{z}_j$，有

$$\begin{aligned} Var(\mathbf{z}_j) =& Var(\mathbf{y}[\mathbf{B}]_j) \\ =& \sum_{i=1}^{n_{k-1}} Var(\mathbf{y}_i([\mathbf{B}]_j)_i) \\ =& n_{k-1}Var(\mathbf{y})Var(\mathbf{B}) \end{aligned}$$

因此为了使得输出的神经元 $\mathbf{z}_j$ 的方差 $Var(\mathbf{z}_j)$ 一致且与输入的神经元 $\mathbf{y}$ 相等，则需要满足下式：

$$Var(\mathbf{z}) = Var(\mathbf{z}_j) = Var(\mathbf{y})$$

此即

$$\begin{aligned} n_{k-1}Var(\mathbf{B}) &= 1 \\ Var(\mathbf{B}) &= \frac{1}{n_{k-1}} \end{aligned}$$

有时我们考虑输入输出神经元的综合，取其平均，得到

$$
\begin{aligned}
n &= \frac{n_{k-1}+n_k}{2} \\
Var(\mathbf{B}) &= \frac{1}{n} = \frac{2}{n_{k-1}+n_k}
\end{aligned}
$$

这次便得到著名的 Xavier/He 的方法，即

$$
\mathbf{B}^{(k)} \sim N\left(0, \frac{2}{n_{k-1}+n_k}\right)
$$

> *Xavier/He initialization encourages (in) to be similar to (out).*
> *—— cs229, Andrew Ng*
>
> *另外一种初始化策略是，直接利用 pre-trained model 的参数初始化。*

4. 批量梯度优化算法
   批量梯度下降是介于全局梯度下降和随机梯度下降的一个梯度优化算法，其本质是利用以小批量的样本来估计出全局梯度。