两层神经网络实现报告

赵丰

2018 年 6 月 6 日

1 概要

1. 复现了两层神经网络，与 Tensorflow 结果进行对比
2. BP 公式推导
3. 用两个简单的例子进行实验

2 两层神经网络

设 x 是网络输入，$\hat{y}$ 是后验概率输出。

$$\begin{aligned} o_1 &= w_1 x + b_1 \\ \hat{o}_1 &= \tanh(o_1) \\ o_2 &= w_2 \hat{o}_1 + b_2 \\ \hat{y} &= \sigma(o_2) \end{aligned}$$

其中 w_1, w_2 是矩阵，b_1, b_2 是列向量，σ 是一个向量值的标量函数。我们采用如下两个约定：

- 一元函数如 $\tanh, \log$ 等作用于向量或矩阵是 `elementwise` 作用的。比如 $\tanh([a_1, a_2] = [\tanh a_1, \tanh a_2])$
- 标量函数对矩阵（或向量）求导数得到的矩阵（或向量）行数列数均不变。比如 $\frac{\partial}{\partial x}(x^T A x) = Ax$，其中 x 是列向量，A 是方阵。

设在一次训练中一个 `batch` 有 $x_1, x_2, \ldots, x_N$ 共 N 个向量（均为列向量），分别对应 $y_1, y_2, \ldots, y_N$ 共 N 个数。我们采用如下三组记号:

$$o_1(i) = w_1 x_i + b_1 \tag{1a}$$

$$\hat{o}_1(i) = \tanh(o_1(i)) \tag{1b}$$

$$o_2(i) = w_2 \hat{o}_1(i) + b_2 \tag{1c}$$

$$X = [x_1, x_2, \ldots, x_N] \qquad m \times N \tag{1d}$$

$$Y = [y_1, y_2, \ldots, y_N] \qquad 1 \times N \tag{1e}$$

$$o_1 = [o_1(1), o_1(2), \ldots, o_1(N)] \qquad K \times N \tag{1f}$$

$$\hat{o}_1 = [\hat{o}_1(1), \hat{o}_1(2), \ldots, \hat{o}_1(N)] \qquad K \times N \tag{1g}$$

$$o_2 = [o_2(1), o_2(2), \ldots, o_2(N)] \tag{1h}$$

$$\hat{y}_i = \sigma(o_2) \tag{1i}$$

$$\hat{Y} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_N] \tag{1j}$$

损失函数采用 Cross Entropy。设 $x \in \mathbb{R}^m$, 神经网络有 K 个 hidden units，则 w_1 是 $K \times m$ 的矩阵.

2.1 1 维输出情形

当网络输出只有 1 维时，w_2 是行向量，b_2 是一个数，σ 取为 `sigmoid` 函数。即

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

此时 $\hat{y}_i = \sigma(o_2(i))$，输出的 $\hat{y}$ 表示 $Y=1$ 的概率，$1-\hat{y}$ 表示 $Y=0$ 的概率（Y 的字母表即为 $\{0,1\}$）。此时损失函数的具体表达式为:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\right) \tag{3}$$

其中 N 为训练的样本个数。

代入 (2) 到 (3) 式中得

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(\log(1+e^{-o_2(i)}) + (1-y_i)o_2(i)) \tag{4}$$

$$\Rightarrow \frac{\partial \mathcal{L}}{\partial o_2(i)} = \frac{1}{N}(-\frac{1}{e^{o_2(i)}+1} + (1-y_i)) \tag{5}$$

$$\Rightarrow \frac{\partial \mathcal{L}}{\partial b_2} = \sum_{i=1}^{N}\frac{\partial \mathcal{L}}{\partial o_2(i)}\frac{\partial o_2(i)}{\partial b_2} \tag{6}$$

$$= \frac{1}{N}\sum_{i=1}^{N}(-\frac{1}{e^{o_2(i)}+1} + (1-y_i)) \tag{7}$$

记行向量 s 与标签行向量 y 具有相同的维数，且 $s_i = -\frac{1}{e^{o_2(i)}+1} + (1-y_i)$ 于是损失函数对 b_2 的导数可看作对 s 的平均。

从 (4) 式出发我们进一步求损失函数对 w_2 的导数，

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial w_2(j)} &= \sum_{i=1}^{N}\frac{\partial \mathcal{L}}{\partial o_2(i)}\frac{\partial o_2(i)}{\partial w_2(j)}\\ &= \sum_{i=1}^{N}\frac{\partial \mathcal{L}}{\partial o_2(i)}\hat{o}_1(j,i)\\ &= \frac{1}{N}\sum_{i=1}^{N}s_i\hat{o}_1(j,i)^1\\ &= \frac{1}{N}(s\hat{o}_1^T)(j)\\ &\Rightarrow \frac{\partial \mathcal{L}}{\partial w_2} = \frac{1}{N}(s\hat{o}_1^T)\end{aligned}$$

我们得到了如下的结果：

$$s = -\frac{1}{e^{o_2}+1} + (1-y) \tag{8}$$

$$\frac{\partial \mathcal{L}}{\partial b_2} = \mathtt{mean}(s) \tag{9}$$

$$\frac{\partial \mathcal{L}}{\partial w_2} = \frac{1}{N}(s\hat{o}_1^T) \tag{10}$$

[1]表示 $\hat{o}_1$ 的第 j 行，第 i 列的元素

接下来处理第一层，

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \hat{o}_1(j,i)} &= \frac{\partial \mathcal{L}}{\partial o_2(i)} \frac{\partial o_2(i)}{\partial \hat{o}_1(j,i)} \\ &= \frac{1}{N} s_i w_2(j) \\ &\Rightarrow \frac{\partial \mathcal{L}}{\partial \hat{o}_1(i)} = \frac{1}{N} s_i w_2^T\end{aligned}$$

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial b_1(j)} &= \sum_{i=1}^{N} \frac{\partial \mathcal{L}}{\partial \hat{o}_1(i)} \cdot \frac{\partial \hat{o}_1(i)}{\partial b_1(j)}{}^{2} \\ &= \frac{1}{N} \sum_{i=1}^{N} s_i \sum_{k=1}^{K} w_2(k) \frac{\partial \hat{o}_1(k,i)}{\partial b_1(j)} \\ &= \frac{1}{N} \sum_{i=1}^{N} \frac{s_i}{\cosh^2 o_1(j,i)} w_2(j) \\ &= \texttt{mean}(\frac{w_2^T s}{\cosh^2 o_1}, \texttt{rowwise})\end{aligned}$$

上式最后一个等式指对 $K \times N$ 的矩阵每一行取平均，得到列向量 b_1。

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial w_1(i,j)} &= \sum_{k=1}^{N} \frac{\partial \mathcal{L}}{\partial \hat{o}_1(k)} \cdot \frac{\partial \hat{o}_1(k)}{\partial w_1(i,j)} \\ &= \frac{1}{N} \sum_{k=1}^{N} s_k \sum_{r=1}^{K} w_2(r) \frac{\partial \hat{o}_1(r,k)}{\partial w_1(i,j)} \\ &= \frac{1}{N} \sum_{k=1}^{N} \frac{s_k w_2(i)}{\cosh^2 o_1(i,k)} X(i,k) \\ &= \frac{1}{N} \frac{w_2^T s}{\cosh^2 o_1} X^T\end{aligned}$$

整理有如下的结果：

$$f = \frac{w_2^T s}{\cosh^2 o_1} \qquad K \times N \tag{11}$$

$$\frac{\partial \mathcal{L}}{\partial b_1} = \texttt{mean}(f, \texttt{rowwise}) \tag{12}$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{1}{N} f X^T \qquad K \times m \tag{13}$$

[2] 两个列向量的内积

3 多维输出的情形

当网络输出为多维 (J 维，$J \geq 2$) 时，采用 `softmax` 输出后验概率:

$$\hat{y}_i(r) = \frac{e^{o_2(r,i)}}{\sum_{k=1}^{J} e^{o_2(k,i)}}, r = 1, 2, \ldots J \tag{14}$$

其中 J 也为输出层的维数，则 w_2, o_2 是 $J \times K$ 维的矩阵，b_2 是 J 维的列向量。$Y \in \{1, 2, \ldots, J\}$（这里假设元素下标从 1 开始)。

则损失函数为

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} \log \hat{y}_i(y_i) \tag{15}$$

将 (14) 代入损失函数 (15) 式中得：记 $t_i = \sum_{k=1}^{J} e^{o_2(k,i)}, t = [t_1, t_2, \ldots, t_N]$

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}(o_2(y_i, i) + \log t_i) \tag{16}$$

$\mathcal{L}$ 对 $o_2(r, i)$ 求导得:

$$\frac{\partial \mathcal{L}}{\partial o_2(r,i)} = -\frac{1}{N}(\delta_{r,y_i} + \hat{y}_i(r)) \tag{17}$$

设 $s(r, i) = \delta_{r,y_i} + \hat{y}_i(r)$, s 为 $J \times N$ 维的矩阵，则类似于 (9) 我们有

$$\frac{\partial \mathcal{L}}{\partial b_2} = \texttt{mean}(s, \texttt{rowwise}) \tag{18}$$

w_2, w_1, b_1 的表达式与 (10)式、(12)、(10)相同。

4 实现

见 `neural_net_cross_entropy.py`。公式推导是在实现后加的。编程实现中存在下标从零开始，一阶张量和二阶张量维数不同做运算要用点积等问题，这和推导中下标从 1 开始，尽量使用矩阵运算的 convention 不太一样，但基本思路是一样的。在实现中仍存在很多可以改进的地方:

- 激活函数需要手动改代码调整
- 目前 C++ 实现只支持单层输出、动态矩阵。使用类模板写 Two_Layer_Net 类可以使得声明的矩阵维数是指定的，便于阅读，效率也更高。
- 增加 momentum 功能。

5 XOR 问题求解

训练数据如表 1所示：采用 2-2-1 的两层神经网络求解，实验中发现并

X		Y
0	0	0
1	1	0
0	1	1
1	0	1

表 1: XOR 问题训练集

不是每次迭代都可以达到 100% 正确率。

6 自动编码问题求解

将原始数据每次 1 个 byte 8 位输入自动编码机，输出是 3bit 的压缩数据，再每 3bit 输入到自动解码机。输出原来的 8 位数据，构成 8-3-8 的两层神经网络（最后一层使用 `sigmoid` 函数将每个分量的输出归一化到 $[0,1]$)，实现 identity mapping。理想化的情况是该组合的网络输入与输出完全相同。但根据信息论的原理，不可能实现无损数据压缩。我们根据现有的神经网络框架在 $1,2,4,...,128$（十进制）8 个数据训练简单了两层神经网络，在 0 255 的 8 位数据上进行预测，用是否大于 0.5 将神经网络的输出结果二值化为 0 或 1。预测结果如下表 2所示。

	无差错	1 位错	2 位错	3 位错	4 位错	5 位错	6 位错	7 位错	8 位错
个数	15	21	43	77	47	36	15	2	0

表 2: 译码器预测结果