深度学习_0.3

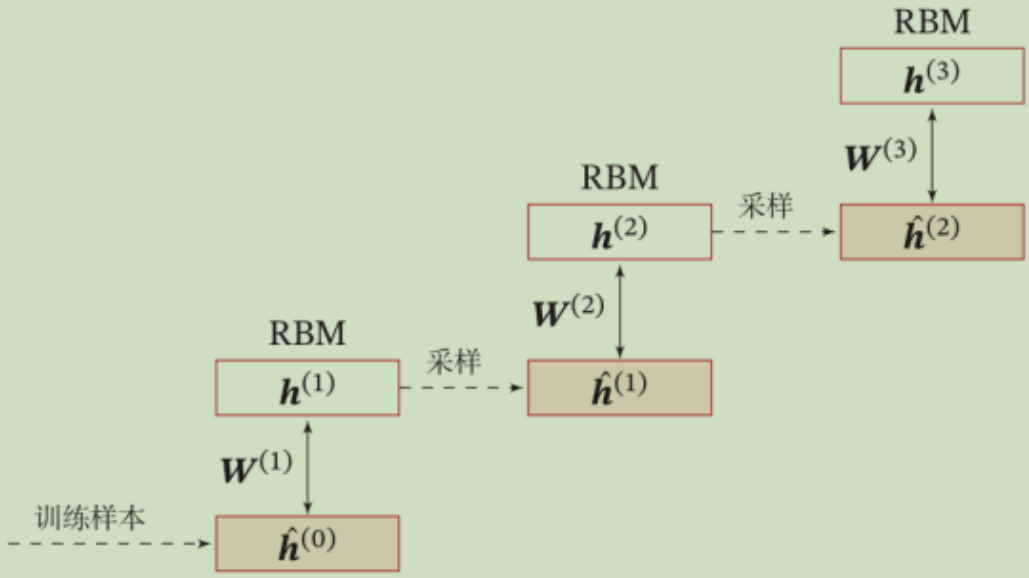
✓ 伪代码

反向传播

输入: 训练集 $\mathcal{D}=\{(\boldsymbol{x}^{(n)},y^{(n)})\}_{n=1}^{N}$, 验证集 $\mathcal{V}$, 学习率 α, 正则化系数 λ, 网络层数 L, 神经元数量 $M_l, 1\leq l\leq L$.

随机初始化 $\boldsymbol{W},\boldsymbol{b}$;
repeat
  对训练集 $\mathcal{D}$ 中的样本随机重排序;
  for $n=1\cdots N$ **do**
    从训练集 $\mathcal{D}$ 中选取样本 $(\boldsymbol{x}^{(n)},y^{(n)})$;
    前馈计算每一层的净输入 $\boldsymbol{z}^{(l)}$ 和激活值 $\boldsymbol{a}^{(l)}$, 直到最后一层;
    反向传播计算每一层的误差 $\delta^{(l)}$; // 公式(4.63)
    // 计算每一层参数的导数
    $\forall l,\quad \frac{\partial \mathcal{L}(y^{(n)},\hat{y}^{(n)})}{\partial \boldsymbol{W}^{(l)}}=\delta^{(l)}(\boldsymbol{a}^{(l-1)})^{\mathrm{T}}$; // 公式(4.68)
    $\forall l,\quad \frac{\partial \mathcal{L}(y^{(n)},\hat{y}^{(n)})}{\partial \boldsymbol{b}^{(l)}}=\delta^{(l)}$; // 公式(4.69)
    // 更新参数
    $\boldsymbol{W}^{(l)}\leftarrow \boldsymbol{W}^{(l)}-\alpha(\delta^{(l)}(\boldsymbol{a}^{(l-1)})^{\mathrm{T}}+\lambda \boldsymbol{W}^{(l)})$;
    $\boldsymbol{b}^{(l)}\leftarrow \boldsymbol{b}^{(l)}-\alpha\delta^{(l)}$;
  end
until 神经网络模型在验证集 $\mathcal{V}$ 上的错误率不再下降;
输出: $\boldsymbol{W},\boldsymbol{b}$

深度信念网络的逐层与训练方法



算法 12.2　深度信念网络的逐层预训练方法

输入: 训练集 $\{\hat{\boldsymbol{v}}^{(n)}\}_{n=1}^{N}$，学习率 α，深度信念网络层数 L，第 l 层权重 $\boldsymbol{W}^{(l)}$，第 l 层偏置 $\boldsymbol{a}^{(l)}$，第 l 层偏置 $\boldsymbol{b}^{(l)}$;

for $l = 1 \cdots L$ **do**
  初始化: $\boldsymbol{W}^{(l)} \leftarrow 0$, $\boldsymbol{a}^{(l)} \leftarrow 0$, $\boldsymbol{b}^{(l)} \leftarrow 0$;
  从训练集中采样 $\hat{\boldsymbol{h}}^{(0)}$;
  for $i = 1 \cdots l-1$ **do**
    根据分布 $p(\boldsymbol{h}^{(i)}|\hat{\boldsymbol{h}}^{(i-1)})$ 采样 $\hat{\boldsymbol{h}}^{(i)}$;
  end
  将 $\hat{\boldsymbol{h}}^{(l-1)}$ 作为训练样本，充分训练第 l 层受限玻尔兹曼机，得到参数 $\boldsymbol{W}^{(l)}, \boldsymbol{a}^{(l)}, \boldsymbol{b}^{(l)}$;
end

输出: $\{\boldsymbol{W}^{(l)}, \boldsymbol{a}^{(l)}, \boldsymbol{b}^{(l)}\}, 1 \le l \le L$

对比散度

算法 12.1　单步对比散度算法

输入: 训练集 $\{\hat{\boldsymbol{v}}^{(n)}\}_{n=1}^{N}$，学习率 α

初始化: $\boldsymbol{W} \leftarrow 0, \boldsymbol{a} \leftarrow 0, \boldsymbol{b} \leftarrow 0$;
for $t = 1 \cdots T$ **do**
  for $n = 1 \cdots N$ **do**
    选取一个样本 $\hat{\boldsymbol{v}}^{(n)}$，用公式(12.46)计算 $p(\boldsymbol{h} = \boldsymbol{1}|\hat{\boldsymbol{v}}^{(n)})$，并根据这个分布采集一个隐向量 $\boldsymbol{h}$;
    计算正向梯度 $\hat{\boldsymbol{v}}^{(n)}\boldsymbol{h}^{\top}$;
    根据 $\boldsymbol{h}$，用公式(12.47)计算 $p(\boldsymbol{v} = \boldsymbol{1}|\boldsymbol{h})$，并根据这个分布采集重构的可见变量 $\boldsymbol{v}'$;
    根据 $\boldsymbol{v}'$，重新计算 $p(\boldsymbol{h} = \boldsymbol{1}|\boldsymbol{v}')$ 并采样一个 $\boldsymbol{h}'$;
    计算反向梯度 $\boldsymbol{v}'\boldsymbol{h}'^{\top}$;
    // 更新参数
    $\boldsymbol{W} \leftarrow \boldsymbol{W} + \alpha(\hat{\boldsymbol{v}}^{(n)}\boldsymbol{h}^{\top} - \boldsymbol{v}'\boldsymbol{h}'^{\top})$;
    $\boldsymbol{a} \leftarrow \boldsymbol{a} + \alpha(\hat{\boldsymbol{v}}^{(n)} - \boldsymbol{v}')$;
    $\boldsymbol{b} \leftarrow \boldsymbol{b} + \alpha(\boldsymbol{h} - \boldsymbol{h}')$;
  end
end

输出: $\boldsymbol{W}, \boldsymbol{a}, \boldsymbol{b}$

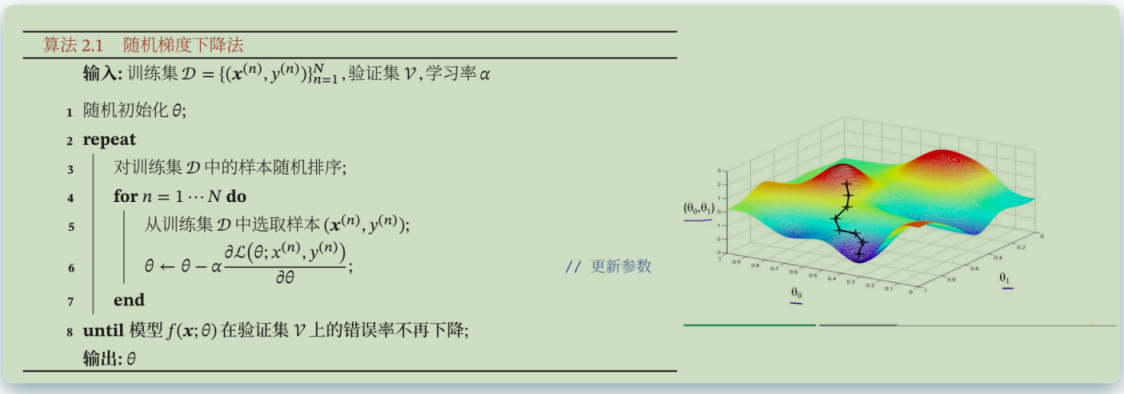
吉布斯采样，不需要等到收敛，只需要K步。

下面的式子中，第一行为公式12.46，第二行为公式12.47（均为向量化后公式）.

$$p(\boldsymbol{h}=1|\boldsymbol{v})=\sigma(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{v}+\boldsymbol{b})$$

$$p(\boldsymbol{v}=1|\boldsymbol{h})=\sigma(\boldsymbol{W}\boldsymbol{h}+\boldsymbol{a}).$$

随机梯度下降



算法 2.1　随机梯度下降法

输入: 训练集 $\mathcal{D}=\{(\boldsymbol{x}^{(n)},y^{(n)})\}_{n=1}^{N}$，验证集 $\mathcal{V}$，学习率 α

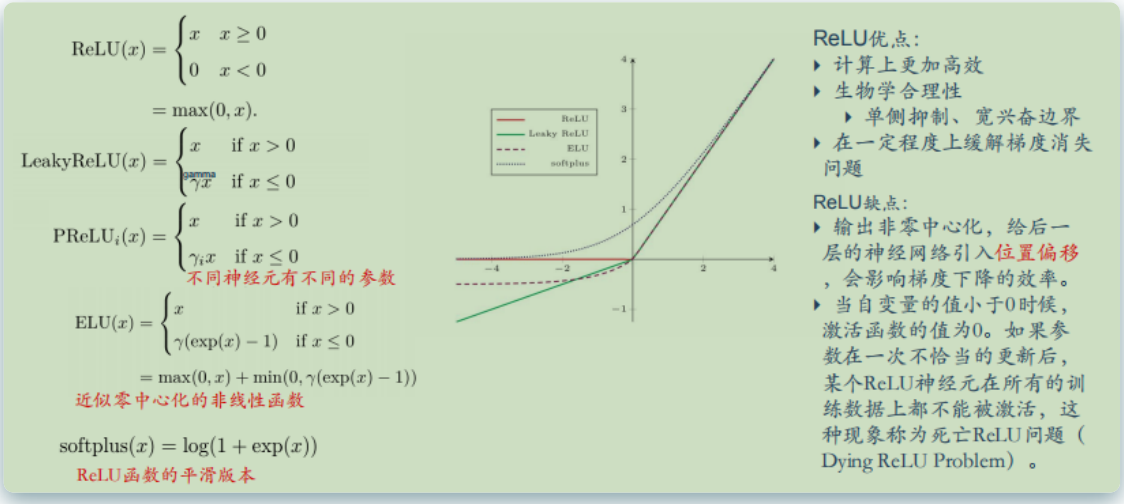
随机初始化 θ;
repeat
　对训练集 $\mathcal{D}$ 中的样本随机排序;
　for $n=1\cdots N$ **do**
　　从训练集 $\mathcal{D}$ 中选取样本 $(\boldsymbol{x}^{(n)},y^{(n)})$;
　　$\theta\leftarrow\theta-\alpha\frac{\partial\mathcal{L}(\theta;\boldsymbol{x}^{(n)},y^{(n)})}{\partial\theta}$;　// 更新参数
　end
until 模型 $f(\boldsymbol{x};\theta)$ 在验证集 $\mathcal{V}$ 上的错误率不再下降;
输出: θ

✓ 课后练习部分

4.-3 说明死亡ReLu问题并提出解决方案。

说明：当自变量的值小于0时候，激活函数的值为0。如果参数在一次不恰当的更新后，某个ReLU神经元在所有的训练数据上都不能被激活，这种现象称为死亡ReLU问题。

解决：选用其他的激活函数（Leaky ReLU，ELU，PReLu激活函数），避免完全死亡问题。



$$\mathrm{ReLU}(x)=\begin{cases}x & x\geq 0\\ 0 & x<0\end{cases}$$
$$=\max(0,x).$$
$$\mathrm{LeakyReLU}(x)=\begin{cases}x & \text{if } x>0\\ \gamma x & \text{if } x\leq 0\end{cases}$$
$$\mathrm{PReLU}_i(x)=\begin{cases}x & \text{if } x>0\\ \gamma_i x & \text{if } x\leq 0\end{cases}$$
不同神经元有不同的参数
$$\mathrm{ELU}(x)=\begin{cases}x & \text{if } x>0\\ \gamma(\exp(x)-1) & \text{if } x\leq 0\end{cases}$$
$$=\max(0,x)+\min(0,\gamma(\exp(x)-1))$$
近似零中心化的非线性函数
$$\mathrm{softplus}(x)=\log(1+\exp(x))$$
ReLU函数的平滑版本

ReLU优点:
- 计算上更加高效
- 生物学合理性
  - 单侧抑制、宽兴奋边界
- 在一定程度上缓解梯度消失问题

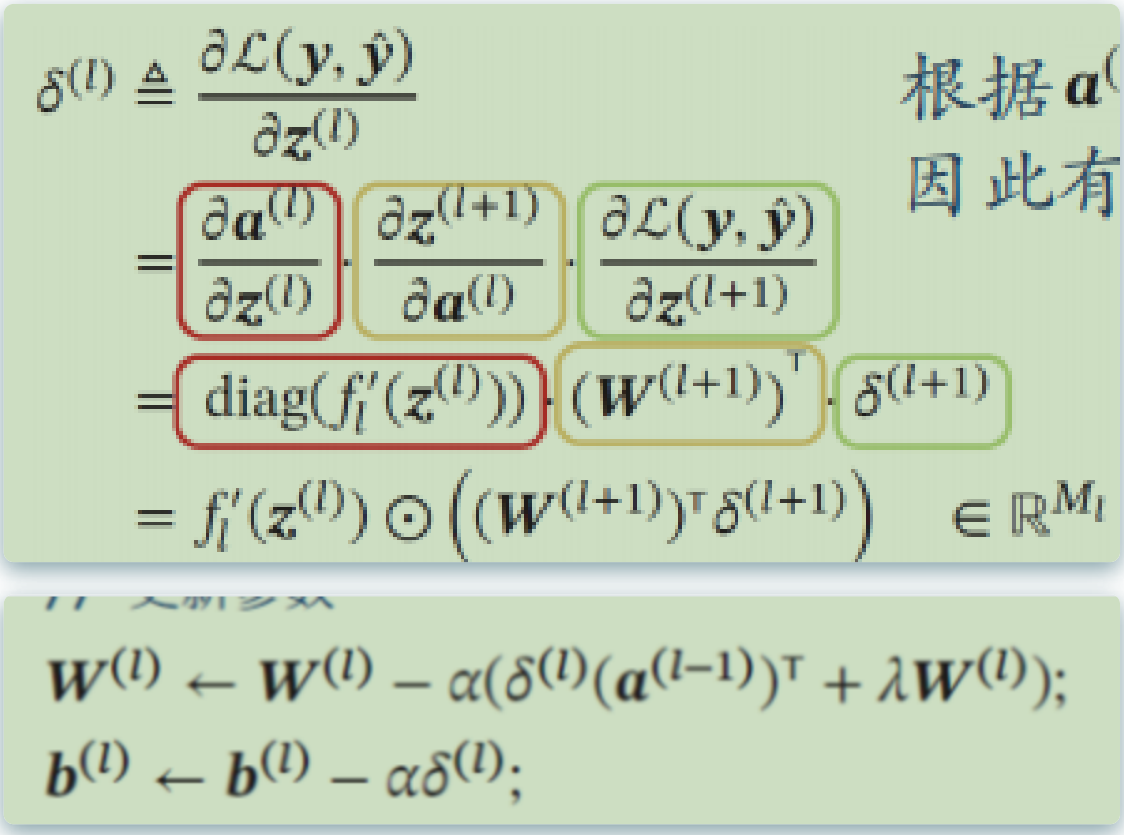
ReLU缺点:
- 输出非零中心化，给后一层的神经网络引入位置偏移，会影响梯度下降的效率。
- 当自变量的值小于0时候，激活函数的值为0。如果参数在一次不恰当的更新后，某个ReLU神经元在所有的训练数据上都不能被激活，这种现象称为死亡ReLU问题（Dying ReLU Problem）。

4-8 为什么在用反向传播算法进行参数学习时要采用随机初始化参数的方式，而不是直接令W=0，b=0?

如果所有权重初始化为零，那么每个神经元在前向传播时将计算出相同的值，并在反向传播时得到相同的梯度。这种情况下，所有神经元都会执行相同的更新，失去个体差异，无法学习到不同的特征。

4-9 梯度消失问题是否可以通过增加学习率来缓解?

反向传播公式



根据公式可知学习率的变化会改变权参数的变化强度，同时学习率增加，导致参数更新过程中出现剧烈波动，还可能导致梯度爆炸问题，所以不是最佳的解决方案。

推荐使用合适的激活函数、权重初始化方法、批量归一化等方法。

12-6 在深度信念网络中，试分析逐层训练背后的理论依据。

高效的参数初始化：逐层训练为每层提供了有效的权重初始化，使每层能够捕捉有用特征，并将这些特征通过隐藏层表示传递给下一层。

梯度消失问题的缓解：独立训练每一层减少了复杂依赖和长距离的权重调整需求，确保梯度可以在较浅的网络中有效传播。逐步细化的训练方法确保每一层在接收到输入之前都已经部分优化，减少了深层次反向传播的负担。

6-1 分析延时神经网络，卷积神经网络和循环神经网络的异同点。

共同点：

神经网络结构：都包含输入层、隐藏层和输出层，通过反向传播训练。

特征提取：能够从输入数据中提取特征，但在不同维度和数据类型上应用不同方法。

多层结构：可以构建为多层网络，通过逐层提取和处理特征，形成深度模型。

不同点：

延时神经网络：在时间维度上滑动窗口，处理时间序列数据的短期依赖。

卷积神经网络：在空间维度上滑动窗口，处理图像数据，具有平移不变性。

循环神经网络：通过递归连接捕捉时间序列中的长期依赖，适用于处理序列数据。

参考图片：

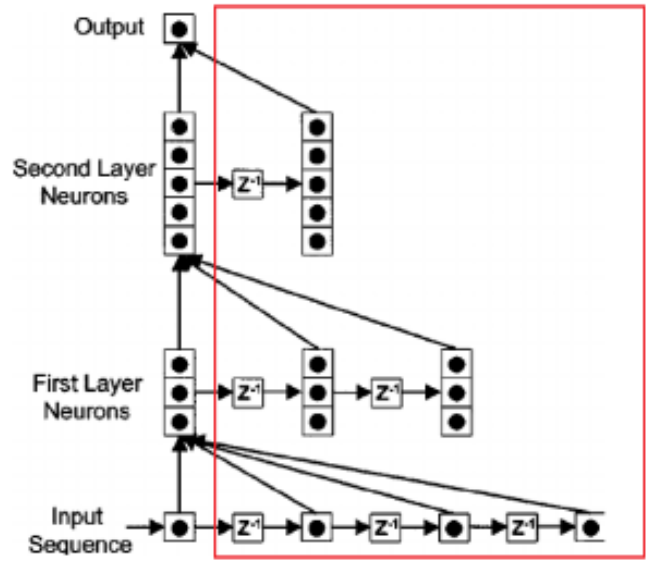
simo-an commented on Jan 3, 2022

题目

习题6-1 分析延时神经网络、卷积神经网络和循环神经网络的异同点.

解答

1. 延时神经网络

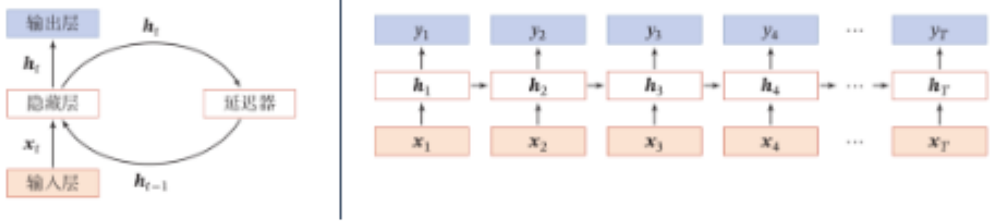


建立一个额外的延时单元，用来存储网络的历史信息

2. 卷积神经网络

延时神经网络去掉红框中的结构（上图），即可得到卷积神经网络

3. 循环神经网络



循环神经网络通过使用带自反馈的神经元，能够处理任意长度的时序数据。

6-3 当使用公式$h_t = h_{t-1} + g(x_t, h_{t-1}; \theta)$ 作为循环神经网络的状态更新公式时，分析其可能存在梯度爆炸的原因并给出解决方法。

原因：

（1）梯度爆炸问题：令$\boldsymbol{z}_k = \boldsymbol{U}\boldsymbol{h}_{k-1} + \boldsymbol{W}\boldsymbol{x}_k + \boldsymbol{b}$为在第$k$时刻函数$g(\cdot)$的输入，在计算公式(6.34)中的误差项$\delta_{t,k} = \frac{\partial \mathcal{L}_t}{\partial \boldsymbol{z}_k}$时，梯度可能会过大，从而导致梯度爆炸问题.

方便理解梯度爆炸部分（不用记）6.34公式为$\delta_{t,k}$部分：

U 是权重矩阵，它是参数 θ的一部分。

梯度消失/爆炸

- 梯度

$$\frac{\partial \mathcal{L}}{\partial U} = \sum_{t=1}^{T}\sum_{k=1}^{t} \delta_{t,k}\mathbf{h}_{k-1}^{\mathrm{T}}$$

- 其中

$$\delta_{t,k} = \prod_{\tau=k}^{t-1}\Big(\mathrm{diag}(f'(\mathbf{z}_\tau))U^{\mathrm{T}}\Big)\delta_{t,t}$$

定义$\gamma \cong \|\mathrm{diag}(f'(\mathbf{z}_t))U^{\mathrm{T}}\|$，则

$$\delta_{t,k} \cong \gamma^{t-k}\delta_{t,t}$$

若$\gamma > 1$，当$t-k \to \infty$时，$\gamma^{t-k} \to \infty$。当间隔$t-k$比较大时，梯度也变得很大，会造成系统不稳定，称为梯度爆炸问题。

若$\gamma < 1$，当$t-k \to \infty$时，$\gamma^{t-k} \to 0$。当间隔$t-k$比较大时，梯度也变得非常小，会出现和深层前馈神经网络类似的梯度消失问题。

由于梯度爆炸或消失问题，实际上只能学习到短周期的依赖关系。这就是所谓的长程依赖问题。

解决方式：

L2正则化：在损失函数中添加权重的L2正则化项，限制权重的大小。

调整学习率： 减小学习率可以减少每次参数更新的步长，从而降低梯度爆炸的风险。

新的网络结构：引入长短期记忆神经网络LSTM，利用门控（遗忘，输入，输出）解决。

7-1 在小批量梯度下降中，试分析为什么学习率和批量大小成正比?

小批量梯度的方差为:$Var(g(\theta)) = \frac{|\sigma 2|}{B}$

B:小批量中的样本集合

σ^2 是单个样本梯度的方差。

小批量梯度的期望等于全数据集的梯度的期望.

梯度下降算法中，参数更新的公式为：

$\theta_{t+1} = \theta_t - \eta g(\theta_t)$

$g(\theta_t)$是在第 t轮的梯度估计。

所以，接下图文字

- 批量大小不影响随机梯度的期望，但会影响随机梯度的方差。
- 批量越大，随机梯度的方差越小，引入的噪声也越小，训练也越稳定，因此可以设置较大的学习率。
- 而批量较小时，需要设置较小的学习率，否则模型会不收敛。

7-7 从再参数化的角度来分析批量归一化中缩放和平移的意义。

对净输入 $z(l)$ 的标准归一化会使得其取值集中到 0 附近， （如果使用 Sigmoid型激活函数时， 这个取值区间刚好是接近线性变换的区间， 减弱了神经网络的非线性性质，**括号可省**）**因此， 为了使得归一化不对网络的表示能力造成负面影响， 可以通过一个附加的缩放和平移变换改变取值区间。**

参考图片：

线性性质. 因此，为了使得归一化不对网络的表示能力造成负面影响，可以通过一个附加的缩放和平移变换改变取值区间.

$$\hat{\boldsymbol{z}}^{(l)} = \frac{\boldsymbol{z}^{(l)} - \boldsymbol{\mu}_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \odot \boldsymbol{\gamma} + \boldsymbol{\beta} \tag{7.55}$$

$$\triangleq \mathrm{BN}_{\boldsymbol{\gamma},\boldsymbol{\beta}}(\boldsymbol{z}^{(l)}), \tag{7.56}$$

其中 $\boldsymbol{\gamma}$ 和 $\boldsymbol{\beta}$ 分别代表缩放和平移的参数向量. 从最保守的角度考虑，可以通过标准归一化的逆变换来使得归一化后的变量可以被还原为原来的值. 当 $\boldsymbol{\gamma} = \sqrt{\sigma_{\mathcal{B}}^2}$，$\boldsymbol{\beta} = \boldsymbol{\mu}_{\mathcal{B}}$ 时，$\hat{\boldsymbol{z}}^{(l)} = \boldsymbol{z}^{(l)}$.

批量归一化操作可以看作一个特殊的神经层，加在每一层非线性激活函数之前，即

$$\boldsymbol{a}^{(l)} = f\big(\mathrm{BN}_{\boldsymbol{\gamma},\boldsymbol{\beta}}(\boldsymbol{z}^{(l)})\big) = f\left(\mathrm{BN}_{\boldsymbol{\gamma},\boldsymbol{\beta}}(\boldsymbol{W}\boldsymbol{a}^{(l-1)})\right), \tag{7.57}$$

其中因为批量归一化本身具有平移变换，所以仿射变换 $\boldsymbol{W}\boldsymbol{a}^{(l-1)}$ 不再需要偏置参数.

8-1 试分析LSTM模型中隐藏层神经元数量与参数数量之间的关系。

LSTM模型中隐藏层神经元数量 h与参数数量之间的关系呈二次方增长，即参数数量随着隐藏层神经元数量的平方增加。

参考资料：

- x_t 是时间步 t 的输入向量，其维度为 d。
- h_t 是时间步 t 的隐藏状态向量（即LSTM单元输出），其维度为 h。

维度分析

输入到门的权重矩阵 W

- **输入向量** x_t：维度为 d。
- **输出** $i_t, f_t, \tilde{C}_t, o_t$：维度为 h，与隐藏状态 h_t 维度相同。

为了使得 $W_i x_t$ 的结果是维度为 h，权重矩阵 W_i 的维度应该是 $h \times d$：
$W_i : h \times d$

隐藏状态到门的权重矩阵 U

- **隐藏状态向量** h_{t-1}：维度为 h。
- **输出** $i_t, f_t, \tilde{C}_t, o_t$：维度为 h。

为了使得 $U_i h_{t-1}$ 的结果是维度为 h，权重矩阵 U_i 的维度应该是 $h \times h$：
$U_i : h \times h$

LSTM单元的更新公式如下：

1. **输入门**：

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

2. **遗忘门**：

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

3. **候选记忆单元状态**：

$$\tilde{C}_t = \tanh(W_C x_t + U_C h_{t-1} + b_C)$$

4. **记忆单元状态**：

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

5. **输出门**：

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

6. **隐藏状态**：

$$h_t = o_t \odot \tanh(C_t)$$

参数数量

每个LSTM单元包含四个门（输入门、遗忘门、输出门和候选记忆单元状态），每个门有其对应的权重矩阵和偏置向量。

- **权重矩阵**：
  - 输入到门的权重矩阵 W 的维度是 $h \times d$。
  - 隐藏状态到门的权重矩阵 U 的维度是 $h \times h$。
- **偏置向量**：
  - 每个门都有一个偏置向量 b，其维度为 h。

总参数数量

LSTM单元有四个门，因此总参数数量为：
$4 \times (h \times d + h \times h + h)$

简化为：
$4h(d + h + 1)$

关系分析

由上述公式可以看出，LSTM模型中隐藏层神经元数量 h 与参数数量 P 之间的关系是非线性的，并且随着 h 的增加，参数数量会显著增加。这种关系可以总结为：
$P = 4h(d + h + 1)$

其中：

- h 是隐藏层神经元的数量。
- d 是输入向量的维度。
- P 是总参数数量。

8-2 分析缩放点积模型可以缓解Softmax函数梯度消失的原因。

缩放点积模型 $s(\boldsymbol{x}, \boldsymbol{q}) = \frac{\boldsymbol{x}^{\top}\boldsymbol{q}}{\sqrt{D}},$

在Softmax函数中，梯度的计算如下：

$$\frac{\partial \alpha_i}{\partial z_j} = \alpha_i(\delta_{ij} - \alpha_j)$$

Softmax函数，其梯度与输入值的大小相关。输入值较小时，梯度更稳定且不容易消失。通过缩放，可以使梯度保持在一个适中的范围内，从而缓解梯度消失问题。

参考资料：

Softmax函数

Softmax函数定义为：

$$\alpha_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

其中，z_i 是输入分数，α_i 是Softmax输出的概率。

在Softmax函数中，梯度的计算如下：

$$\frac{\partial \alpha_i}{\partial z_j} = \alpha_i(\delta_{ij} - \alpha_j)$$

当 z_i 很大时，α_i 接近于0或1，导致梯度 $\frac{\partial \alpha_i}{\partial z_j}$ 接近于0，从而引起梯度消失。通过缩放点积结果，可以将 z_i 控制在一个适中的范围内，使得 α_i 更均匀地分布在 (0, 1) 范围内，梯度也更大、更稳定。

附录（概率会考内容，最后记忆）：

卷积类型

- 卷积的结果按输出长度不同可以分为三类：
  - 窄卷积：步长 $T=1$，两端不补零 $P=0$，卷积后输出长度为 $M-K+1$
  - 宽卷积：步长 $T=1$，两端补零 $P=K-1$，卷积后输出长度 $M+K-1$
  - 等宽卷积：步长 $T=1$，两端补零 $P=(K-1)/2$，卷积后输出长度 M

- 在早期的文献中，卷积一般默认为窄卷积。
- 而目前的文献中，卷积一般默认为等宽卷积。