学习笔记-1：Task 1

1 - 线性回归

模型公式

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{W} + \boldsymbol{b}$$

其中，X 为输入矩阵，W 为权重矩阵，b 为偏移量

损失函数

平方函数，在评估索引为 i 的样本误差的表达式为

$$l^{(i)}(\mathbf{w}, b) = \frac{1}{2}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

$$L(\mathbf{w}, b) = \frac{1}{n}\sum_{i=1}^{n} l^{(i)}(\mathbf{w}, b) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{2}\left(\mathbf{w}^\top \mathbf{x}^{(i)} + b - y^{(i)}\right)^2$$

优化函数：随机梯度下降 (Stochastic gradient descent, SGD)

$$(\mathbf{w}, b) \leftarrow (\mathbf{w}, b) - \frac{\eta}{|\mathcal{B}|}\sum_{i \in \mathcal{B}} \partial_{(\mathbf{w}, b)} l^{(i)}(\mathbf{w}, b)$$

其中，学习率 η 代表在每次优化中，能够学习的步长的大小；批量大小 $\mathcal{B}$ 是小批量计算中的批量大小batch size

错题总结

Q：课程中的损失函数定义为

```
def squared_loss(y_hat, y):
        return (y_hat - y.view(y_hat.size())) ** 2 / 2
```

将返回结果替换为下面的哪一个会导致会导致模型无法训练：

```
A. (y_hat.view(-1) - y) ** 2 / 2
B. (y_hat - y.view(-1)) ** 2 / 2
C. (y_hat - y.view(y_hat.shape)) ** 2 / 2
D. (y_hat - y.view(-1, 1)) ** 2 / 2
```

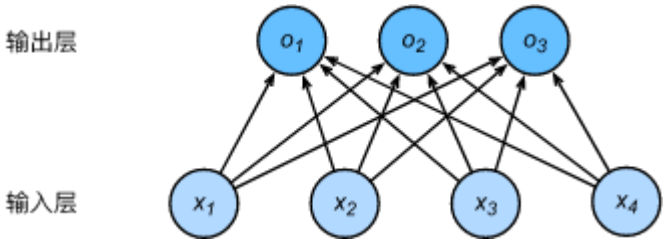
A：应该选B. `y_hat`的形状是`[n, 1]`，而`y`的形状是`[1, n]`，两者相减得到的结果的形状是`[n, n]`，即用`y_hat`的每一个元素分别减去`y`的所有元素，无法得到正确的损失值. 对于A，`y_hat.view(-1)`的形状是`[1, n]`，与`y`一致，可以相减；对于B，`y.view(-1)`的形状仍是`[1, n]`，没有解决问题；对于C和D，`y.view(y_hat.shape)`和`y.view(-1, 1)`的形状都是`[n, 1]`，与`y_hat`一致，可以相减.

2 - Softmax与分类模型

softmax的基本概念

单层神经网络

softmax回归同线性回归一样，也是一个单层神经网络。由于每个输出o_1, o_2, o_3的计算都要依赖于所有的输入x_1, x_2, x_3, x_4，softmax回归的输出层也是一个全连接层



输出问题

直接使用输出层的输出有两个问题：

1. 一方面，由于输出层的输出值的范围不确定，我们难以直观上判断这些值的意义
2. 另一方面，由于真实标签是离散值，这些离散值与不确定范围的输出值之间的误差难以衡量
   softmax 运算符 (softmax operator) 解决了以上两个问题. 它通过下式将输出值变换成值为正且和为1的概率分布：

$$\hat{y}_1, \hat{y}_2, \hat{y}_3 = \text{softmax}(o_1, o_2, o_3)$$

其中

$$\hat{y}_1 = \frac{\exp(o_1)}{\sum_{i=1}^{3} \exp(o_i)}, \quad \hat{y}_2 = \frac{\exp(o_2)}{\sum_{i=1}^{3} \exp(o_i)}, \quad \hat{y}_3 = \frac{\exp(o_3)}{\sum_{i=1}^{3} \exp(o_i)}$$

容易看出$\hat{y}_1 + \hat{y}_2 + \hat{y}_3 = 1$且$0 \leq \hat{y}_1, \hat{y}_2, \hat{y}_3 \leq 1$，因此$\hat{y}_1, \hat{y}_2, \hat{y}_3$是一个合法的概率分布. 且

$$\arg\max_i o_i = \arg\max_i \hat{y}_i$$

因此 softmax 运算不改变预测类别输出.

小批量矢量计算表达式

给定一个小批量样本，其批量大小为n，输入个数(特征数)为d，输出个数(类别数)为q. 设批量特征为$\boldsymbol{X} \in \mathbb{R}^{n \times d}$. 假设softmax回归的权重和偏差参数分别为$\boldsymbol{W} \in \mathbb{R}^{d \times q}$和$\boldsymbol{b} \in \mathbb{R}^{1 \times q}$. softmax回归的矢量计算表达式为

$$\boldsymbol{O} = \boldsymbol{X}\boldsymbol{W} + \boldsymbol{b}$$

$$\hat{\boldsymbol{Y}} = \text{softmax}(\boldsymbol{O})$$

其中的加法运算使用了广播机制，$\boldsymbol{O}, \hat{\boldsymbol{Y}} \in \mathbb{R}^{n \times q}$且这两个矩阵的第$i$行分别为样本$i$的输出$\boldsymbol{o}^{(i)}$和概率分布$\hat{\boldsymbol{y}}^{(i)}$.

交叉熵损失函数

对于样本i，我们构造向量$\boldsymbol{y}^{(i)} \in \mathbb{R}^q$ ，使其第$y^{(i)}$（样本i类别的离散数值）个元素为1，其余为0。这样我们的训练目标可以设为使预测概率分布$\hat{\boldsymbol{y}}^{(i)}$尽可能接近真实的标签概率分布$\boldsymbol{y}^{(i)}$
交叉熵(cross entropy)定义为：

$$H\left(\boldsymbol{y}^{(i)}, \hat{\boldsymbol{y}}^{(i)}\right) = -\sum_{j=1}^{q} y_j^{(i)} \log \hat{y}_j^{(i)}$$

其中带下标的$y_j^{(i)}$是向量$\boldsymbol{y}^{(i)}$中非0即1的元素. 在上式中，我们知道向量$\boldsymbol{y}^{(i)}$ 中只有第$y^{(i)}$个元素$y_{y^{(i)}}^{(i)}$为1，其余全为0，于是

$$H(\boldsymbol{y}^{(i)}, \hat{\boldsymbol{y}}^{(i)}) = -\log \hat{y}_{y^{(i)}}^{(i)}$$

也就是说，交叉熵只关心对正确类别的预测概率，因为只要其值足够大，就可以确保分类结果正确.

假设训练数据集的样本数为n，交叉熵损失函数定义为

$$\ell(\boldsymbol{\Theta}) = \frac{1}{n} \sum_{i=1}^{n} H\left(\boldsymbol{y}^{(i)}, \hat{\boldsymbol{y}}^{(i)}\right)$$

其中$\boldsymbol{\Theta}$代表模型参数. 同样地，如果每个样本只有一个标签，那么交叉熵损失可以简写成
$\ell(\boldsymbol{\Theta}) = -(1/n) \sum_{i=1}^{n} \log \hat{y}_{y^{(i)}}^{(i)}$.
从另一个角度来看，最小化$\ell(\boldsymbol{\Theta})$等价于最大化$\exp(-n\ell(\boldsymbol{\Theta})) = \prod_{i=1}^{n} \hat{y}_{y^{(i)}}^{(i)}$ ，即最小化交叉熵损失函数等价于最大化训练数据集所有标签类别的联合预测概率

错题总结

Q：softmax([100, 101, 102])的结果等于以下的哪一项

```
A. softmax([10.0, 10.1, 10.2])
B. softmax([-100, -101, -102])
C. softmax([-2, -1, 0])
D. softmax([1000, 1010, 1020])
```

A：应该选C. softmax函数具有常数不变性，即softmax(x)=softmax(x+c)，推导如下：

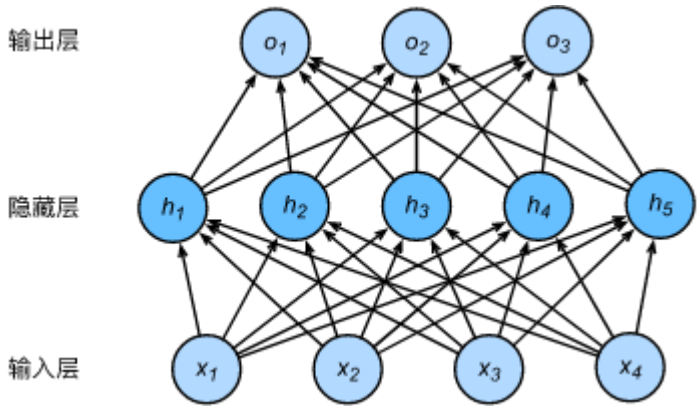
$$softmax(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j)}$$

$$softmax(x_i + c) = \frac{exp(x_i + c)}{\sum_j exp(x_j + c)} = \frac{exp(c)exp(x_i)}{exp(c)\sum_j exp(x_j)} = \frac{exp(x_i)}{\sum_j exp(x_j)} = softmax(x_i)$$

3 - 多层感知器

隐藏层

下图展示了一个多层感知机的神经网络图，它含有一个隐藏层，该层中有5个隐藏单元。



给定一个小批量样本$\boldsymbol{X} \in \mathbb{R}^{n \times d}$，其批量大小为$n$，输入个数为$d$. 假设多层感知机只有一个隐藏层，其中隐藏单元个数为h. 记隐藏层的输出为$\boldsymbol{H}$，有$\boldsymbol{H} \in \mathbb{R}^{n \times h}$. 设隐藏层的权重参数和偏差参数分别为$\boldsymbol{W}_h \in \mathbb{R}^{d \times h}$和$\boldsymbol{b}_h \in \mathbb{R}^{1 \times h}$，输出层的权重和偏差参数分别为$\boldsymbol{W}_o \in \mathbb{R}^{h \times q}$和$\boldsymbol{b}_o \in \mathbb{R}^{1 \times q}$.
则输出 $\boldsymbol{O} \in \mathbb{R}^{n \times q}$ 的计算为：

$$\boldsymbol{O} = (\boldsymbol{X}\boldsymbol{W}_h + \boldsymbol{b}_h)\boldsymbol{W}_o + \boldsymbol{b}_o = \boldsymbol{X}\boldsymbol{W}_h\boldsymbol{W}_o + \boldsymbol{b}_h\boldsymbol{W}_o + \boldsymbol{b}_o.$$

可以看出，虽然神经网络引入了隐藏层，却依然等价于一个单层神经网络

激活函数

上述问题的根源在于全连接层只是对数据做仿射变换，而多个仿射变换的叠加仍然是一个仿射变换. 解决问题的一个方法是引入非线性变换，例如对隐藏变量使用按元素运算的非线性函数进行变换，然后再作为下一个全连接层的输入。这个非线性函数被称为激活函数.

几个常用的激活函数：

- ReLU函数
  ReLU(rectified linear unit)函数提供了一个很简单的非线性变换。给定元素x，该函数定义为

  $$\mathrm{ReLU}(x) = \max(x, 0)$$

- Sigmoid函数
  sigmoid函数可以将元素的值变换到0和1之间：

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

  依据链式法则，sigmoid函数的导数

  $$\mathrm{sigmoid}'(x) = \mathrm{sigmoid}(x)\left(1 - \mathrm{sigmoid}(x)\right).$$

- tanh函数
  tanh(双曲正切)函数可以将元素的值变换到-1和1之间：

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

  依据链式法则，tanh函数的导数

  $$\tanh'(x) = 1 - \tanh^2(x).$$

关于激活函数的选择

- ReLu函数是一个通用的激活函数，目前在大多数情况下使用。但是，ReLU函数只能在隐藏层中使用
- 用于分类器时，sigmoid函数及其组合通常效果更好。由于梯度消失问题，有时要避免使用sigmoid和tanh函数
- 在神经网络层数较多的时候，最好使用ReLu函数，ReLu函数比较简单计算量少，而sigmoid和tanh函数计算量大很多
- 在选择激活函数的时候可以先选用ReLu函数如果效果不理想可以尝试其他激活函数

多层感知机

多层感知机就是含有至少一个隐藏层的由全连接层组成的神经网络，且每个隐藏层的输出通过激活函数进行变换.
多层感知机按以下方式计算输出：

$$\boldsymbol{H} = \phi(\boldsymbol{X}\boldsymbol{W}_h + \boldsymbol{b}_h)$$

$$\boldsymbol{O} = \boldsymbol{H}\boldsymbol{W}_o + \boldsymbol{b}_o$$

其中ϕ表示激活函数