Assignment 1

第一次作业一共有四道题目，教授提示每道题目都需要思考，但是不需要很长的答案。我在完成的过程中发现，每道大题中的小题目都是循序渐进的，需要实现的代码基本都是依据前面几道小题实现的。作业的工程量真的很大，但是做完之后，有些课程视频中不明白的地方就会明白了。

1. softmax

(a). 证明softmax函数的平移不变性，对输入添加敞亮便宜，softmax值保持不变，即 $softmax(x) = softmax(x + c)$， 其中，$c$是常量，$x$是任意维度的张量。记住：

$$softmax(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

> 注意：实际上，在计算softmax时，为了保证数值稳定，通常选择$c = -max_i x_i$， 即$\boldsymbol{X}$ 减去$\boldsymbol{X}$ 中最大的那个元素。

解：

$$softmax(x + c)_i = \frac{e^{x_i + c}}{\sum_j e^{x_j + c}} = \frac{e^{x_i}\, e^{c}}{\sum_j e^{x_j}\, e^{c}} = \frac{e^{x_i}\, e^{c}}{e^{c} \sum_j e^{x_j}} = \frac{e^{x_i}}{\sum_j e^{x_j}} = softmax(x)_i$$

(b). 假设一个N行d列的输入矩阵，对每行数据计算softmax的预测值。代码在文件q1_softmax.py中实现，测试命令"python q1_softmax.py"。需要特别注意单行向量的情况。

> 本次作业里的测试代码都是并不详尽，需要我们自己提供更多的测试用例。后续作业需要参考这部分的代码，所以我们要尽量正确的完成，并且我们的实现要向量化。

解：

答案见代码文件。

2. Neural Network Basics

(a). 推导sigmoid函数的梯度，结果可以被重写为值的函数（即结果可以转化为 $\sigma(x)$ 形式的函数）。假设输入 x 是一个标量，sigmoid函数式是：

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

解：

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

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

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

$$= \sigma(x)(1 - \sigma(x))$$

(b). 使用交叉熵作为评测的损失函数，推导softmax函数函数的梯度。假如输入向量 θ ，用softmax得到预测结果 $\hat{y} = softmax(\theta)$ ，交叉熵函数：

$$CE(y, \hat{y}) = -\sum_i y_i log(\hat{y}_i) \text{ 。}$$

其中，y表示one-hot类别向量， $\hat{y}$ 是预测的所有类别的概率向量。

> 你(我们)可能需要考虑 y 的大部分值为0的情况，所以只需要考虑值为1的第 k 维。

解：

我们分情况讨论：

（1）当 $i = k$ 时，$CE(y, \hat{y}) = -log\frac{e^{\theta_i}}{\sum_j e^{\theta_j}} = -loge^{\theta_i} + log\sum_j e^{\theta_j}$

$$\frac{\partial}{\partial \theta_i} CE(y, \hat{y}) = -\frac{1}{e^{\theta_i}} e^{\theta_i} + \frac{1}{\sum_j e^{\theta_j}} \frac{\partial}{\partial \theta_i} \sum_j e^{\theta_j} = -1 + \frac{e^{\theta_i}}{\sum_j e^{\theta_j}}$$

$$= -1 + \hat{y}_i$$

（2）当$i \neq k$ 时，y_i只会影响归一化因子，也就是分母，这时：

$$\frac{\partial}{\partial \theta_i} CE(y, \hat{y}) = \frac{\partial}{\partial \theta_i} (-loge^{\theta_k} + log\sum_j e^{\theta_j})$$

$$= \frac{\partial}{\partial \theta_i} log\sum_j e^{\theta_j} = \frac{e^{\theta_i}}{\sum_j e^{\theta_j}}$$

$$= \hat{y}_i$$

所以，综合两种情况得到：

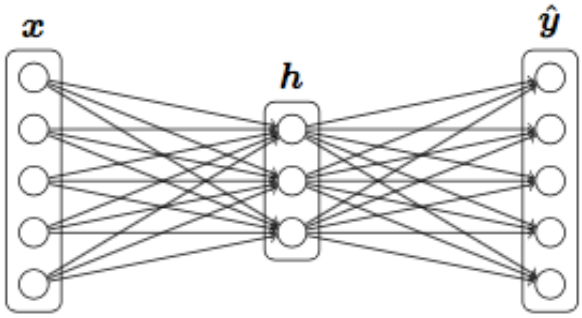
$$\frac{\partial}{\partial \theta} CE(y, \hat{y}) = \hat{y} - y$$

这道题也可以不分情况讨论，根据题目得到损失函数：

$$CE(y, \hat{y}) = -\sum_i y_i lne^{\theta_i} + ln\sum_j e^{\theta_j}$$

这里第一项lne^{θ_i}前面乘以y_i，而$ln\sum_j e^{\theta_j}$前不乘以y_i是因为y_i只影响与抽样相关的项，而不影响归一化因子。换句话说，当y_i为真的时候，需要对这项θ_i求导，当y_i为假的时候，这项就相当于一个常数，导数为0。

(c). 推导只含有一个隐藏层的神经网络的梯度，也就是输入是x，损失函数为J，求$\frac{\partial J}{\partial x}$。这个神经网络隐含层的激活函数使用sigmoid函数，输出层的激活函数使用softmax函数。假设y为one-hot的类别向量，使用交叉熵作为损失函数。可以使用$\sigma'(x)$作为sigmoid梯度的简写，也可以随意定义你需要的其他变量。



我们来回顾前向传播的公式：

$$h = sigmoic(xW_1 + b_1) \qquad \hat{y} = softmax(hW_2 + b_2)$$

> 注意这里我们假设输入向量x是行向量，隐藏层变量和输出概率向量也是行向量，这也是为了保持与后续编程任务的一致性。我们对一个向量使用sigmoid，就是对向量的每一个值都计算sigmoid。W_i和b_i $(i = 1, 2)$分别是两层的权重和偏差。

解：

首先，我们分析前向传播的过程和每个过程的梯度：

$$I = xW_1 + b_1 \qquad \frac{\partial I}{\partial x} = W_1^T$$

$$h(I) = \sigma(I) \qquad \frac{\partial h(I)}{\partial I} = \sigma(I)(1 - \sigma(I))$$

$$O(h) = hW_2 + b_2 \qquad \frac{\partial O(h)}{\partial h} = W_2^T$$

$$\hat{y} = softmax(O) \qquad \frac{\partial I}{\partial O} = \hat{y} - y$$

由**链式法则**可以得到：

$$\frac{\partial J}{\partial x} = \frac{\partial J}{\partial O}\frac{\partial O}{\partial h}\frac{\partial h}{\partial I}\frac{\partial I}{\partial x} = (\hat{y} - y)W_2^T\sigma(xW_1 + b_1)(1 - \sigma(xW_1 + b_1))W_1^T$$

这里需要注意矩阵的求导!

(d). 在上题的神经网络里，假设输入 x 有 D_x 维，输出 y 有 D_y 维，隐含层有 H 个单元，共有多少个参数?

解：共有 $D_xH + H + D_yH + D_y$ 个参数。

(e). 在代码文件q2_sigmoid.py中实现sigmoid激活函数和它的梯度。测试命令"python q2_sigmoid.py"，同样，测试用例并不详尽。

解：

答案见代码文件。

(f). 为了更容易调试，我们来实现一个梯度检测器。在文件q2_gradcheck.py中实现gradcheck_naive方法的代码，测试命令“python q2_gradcheck.py”。

解：

梯度检测原理：为了检验我们推导的梯度公式是否正确，我们从导数的数学定义入手，假设以$\boldsymbol{\theta}$为自变量的目标函数为J, 导数为：

$$\frac{\partial J}{\partial \theta} = lim_{h->0} \frac{J(\theta + h) - J(\theta - h)}{2h} \approx g(\theta)$$

通常h设置很小，为10^{-4}数量级。

答案见代码文件。

(g). 现在，我们来实现一个神经网络的前向传播和后向传播，使用一层sigmoid函数为隐含层。在文件q2_neural.py中完成代码。

解：

首先，这个神经网络的思路在(c)小题中出现过，在上面的几道题中，我们也求出了前向传播的损失。那么，后向传播就是分别对W_1，b_1，W_2，b_2求导的过程。

前向传播过程：

$$I = xW_1 + b_1 \qquad H = \sigma(I) = \sigma(xW_1 + b_1)$$

$$O = HW_2 + b_2 \qquad \hat{y} = softmax(O) = softmax(HW_2 + b_2)$$

后向传播求梯度：

$$\frac{\partial \hat{y}}{\partial W_2} = \frac{\partial \hat{y}}{\partial O}\frac{\partial O}{\partial H} = (\hat{y} - y)H^T$$

$$\frac{\partial \hat{y}}{\partial b_2} = \frac{\partial \hat{y}}{\partial O}\frac{\partial O}{\partial b_2} = (\hat{y} - y)$$

$$\frac{\partial \hat{y}}{\partial W_1} = \frac{\partial J}{\partial O}\frac{\partial O}{\partial H}\frac{\partial H}{\partial I}\frac{\partial I}{\partial W_1} = (\hat{y} - y)W_2^T sigmoid\_grad(H)x^T$$

$$\frac{\partial \hat{y}}{\partial b_1} = \frac{\partial J}{\partial O}\frac{\partial O}{\partial H}\frac{\partial H}{\partial I}\frac{\partial I}{\partial b_1} = (\hat{y} - y)W_2^T sigmoid\_grad(H)$$

答案见代码文件。

3. word2vec

(a). 假设skipgram模型给定一个中心词c的预测词向量v_c， 预测时使用word2vec模型中的softmax函数：

$$\hat{y}_o = p(o|c) = \frac{exp(u_o^T v_c)}{\sum_{w=1}^{W} exp(u_w^T v_c)}$$

其中，w 表示第w个词，u表示词的输出词向量。假设以交叉熵为损失函数，词o为目标词，求 v_c 的梯度。

> 使用第2题中的标记很有帮助。例如，用$\hat{y}$标记softmax预测的词向量，y标记期望的词向量，损失函数$J_{softmax_{CE}}(o, v_c, U) = CE(y, \hat{y})$，我们用$U = [u_1, u_2, \dots, u_w]$表示所有输出向量的矩阵。尤其要注意向量 和矩阵的方法。

解：

这道题目教授在课上白板推导过。这里简单写一下。

$$\frac{\partial J}{\partial v_c} = \frac{\partial}{\partial v_c}(-log\frac{exp(u_o^T v_c)}{\sum_{w=1}^{W} exp(u_w^T v_c)}) = \frac{\partial}{\partial v_c}(-logexp(u_0^T v_c)) + \frac{\partial}{\partial v_c} log \sum_{w=1}^{W} exp(u_w^T v_c)$$

$$= -u_o + \frac{1}{\sum_{w=1}^{W} exp(u_w^T v_c)} \sum_{t=1}^{W} \frac{\partial}{\partial v_c} exp(u_t^T v_c)$$

$$= -u_o + \frac{1}{\sum_{w=1}^{W} exp(u_w^T v_c)} \sum_{t=1}^{W} exp(u_t^T v_c) u_t$$

$$= -u_o + \sum_{t=1}^{W} \frac{exp(u_t^T v_v)}{\sum_{w=1}^{W} exp(u_w^T v_c)} u_t$$

$$= -u_o + \sum_{t=1}^{W} \hat{y}_t u_t$$

$$= U^T(\hat{y} - y)$$

(b).继续上题，求所有输出词向量u_w的梯度，包括u_o。

解：

我们还是分为 $w = o$ 和 $w \neq o$ 两种情况讨论：

（1）$w = o$ 时：

$$\frac{\partial J}{\partial u_w} = \frac{\partial}{\partial u_w}(-logexp(u_o^T v_c)) + \frac{\partial}{\partial u_w} log \sum_{w=1}^{W} exp(u_w^T v_c)$$

$$= -v_c + \frac{1}{\sum_{w=1}^{W} exp(u_w^T v_c)} exp(u_w^T v_c) v_c$$

$$= -v_c + \hat{y}_w v_c$$

$$= v_c(\hat{y}_w - 1)$$

（2）$w \neq o$ 时，损失函数分子为常量，导数为0，所以只需要对分母求导：

$$\frac{\partial}{\partial u_w} = \frac{\partial}{\partial u_w} log \sum_{w=1}^{W} exp(u_w^T v_c)$$

$$= \frac{1}{\sum_{w=1}^{W} exp(u_w^T v_c)} exp(u_w^T v_c) v_c$$

$$= \hat{y}_w v_c$$

综合两种情况：

$$\frac{\partial J}{\partial u_w} = v_c(\hat{y} - y)$$

(c). 假设我们使用负采样方法来预测词向量v_c，期望输出词为o，重复(a)(b)两小题中的梯度推导。假设采样K个负样本，用$1, \ldots, K$来标记，$o \notin \{1, \ldots, K\}$，同样，指定词$o$的输出向量为$u_o$。本例中负采样的损失函数为：

$$J_{neg-sample}(o, v_c, U) = -log(\sigma(u_o^T v_c)) - \sum_{k=1}^{K} log(\sigma(-u_k^T v_c))$$

其中$\sigma()$为sigmoid函数。当做完这些后，用一句话描述为什么这个损失函数的计算效率比softmax损失的计算效率高很多？（你可以提供速度提升比例，例如，负采样损失计算运行时间除以softmax损失运行时间。）

> 与Mikolov在原始论文中的损失函数相比，这里取了负值，因为我们的代码要求最小化，而不是最大化。

解：

$$\frac{\partial J}{\partial v_c} = -\frac{\partial}{\partial v_c} log(\sigma(u_o^T v_c)) - \sum_{k=1}^{K} \frac{\partial}{\partial v_c} log(\sigma(-u_k^T v_c))$$

$$= -\frac{1}{\sigma(u_o^T v_c)}\sigma(u_o^T v_v)(1-\sigma(u_o^T v_c))u_o - \sum_{k=1}^{K}\frac{1}{\sigma(-u_k^T v_c)}\sigma(-u_k^T v_c)(1-\sigma(-u_k^T v_c))(-u_k)$$

$$= (\sigma(u_o^T v_c)-1)u_o - \sum_{k=1}^{K}(\sigma(-u_k^T v_c)-1)u_k$$

$$\frac{\partial J}{\partial u_o} = -\frac{\partial}{\partial u_o} log(\sigma(u_o^T v_c))$$

$$= (\sigma(u_o^T v_c)-1)v_c$$

$$\frac{\partial J}{\partial u_k} = -\frac{\partial}{\partial u_k}\sum_{k=1}^{K} log(\sigma(-u_k^T v_c))$$

$$= -(\sigma(-u_k^T v_c)-1)v_c$$

(d). 在上题的基础上，给定上下文窗口$[word_{c-m},\dots,word_{c-1},word_c,word_{c+1},\dots,word_{c+m}]$，对skip-gram和CBOW模型中的所有词向量推导梯度。用v_k和u_k标记$word_k$的输入向量和输出向量。

> 可以随意使用$F(o,v_c)$替代这部分的损失函数$J_{softmax-CE}(o,v_c,\dots)$和$J_{neg_sample}(o,v_c,\dots)$，在编码的部分你将看到这种简写非常有用。所以本题答案可以包含$\frac{\partial F(o,v_c)}{\partial \dots}$这种形式。

回顾一下skip-gram模型，以c为中心词的窗口损失函数为：

$$J_{skip-gram}(word_{c-m\dots c+m}) = \sum_{-m\le j\le m, j\neq 0} F(w_{c+j}, v_c)$$

CBOW模型有点不同，这里我们用 $\hat{v}$ 代替 v_c 作为预测向量。对于相对简单的CBOW的变形，我们将窗口词的输入向量相加作为输入向量，所以CBOW的损失函数为：

$$\hat{v} = \sum_{-m\le j\le m, j\neq 0} v_{c+j}$$

$$J_{CBOW}(word_{c-m\dots c+m}) = F(w_c, \hat{v})$$

> 在编码部分也要保持使用$\hat{v}$的一致性，skip-gram模型中$\hat{v} = v_c$

解：

这道题看着很麻烦，有很多参数要求，但是教授在题目中提到了可以用F代替，所以结果就比较简洁了。

首先我们统计所有需要求的参数有，输入向量 $v_{c-m},\dots v_{c-1},v_c,v_{c+1},\dots,v_{c+m}$，输出向量 $u_{c-m},\dots,u_{c-1},u_c,u_{c+1},\dots,u_{c+m}$。定义 $U=[u_{c-m},\dots,u_{c-1},u_c,u_{c+1,..,u_{c+m}}]$为所有词的输出向量。

对于skip-gram模型：

$$\frac{\partial J_{skip-gram}(w_{c-m,\dots,c+m})}{\partial U}=\sum_{-m\le j\le m,j\neq 0}\frac{\partial F(w_{c+j},v_c)}{\partial U}$$

$$\frac{\partial J_{skip-gram}(w_{c-m,\dots,c+m})}{\partial v_c}=\sum_{-m\le j\le m,j\neq 0}\frac{\partial F(w_{c+j},v_c)}{\partial v_c}$$

$\frac{\partial J_{skip-gram}(w_{c-m,..,c+m})}{\partial v_j}=0,-m\le j\le m,j\neq 0$（因为损失函数与上下文词的输入向量无关）

对于CBOW模型：

$$\frac{\partial J_{CBOW}(w_{c-m,\dots,c+m})}{\partial U}=\frac{\partial F(w_c,\hat{v})}{\partial U}$$

$$\frac{\partial J_{CBOW}(w_{c-m,\dots,c+m})}{\partial v_c}=0$$

$$\frac{\partial J_{CBOW}(w_{c-m,\dots,c+m})}{\partial v_j}=\frac{\partial F(w_c,\hat{v})}{\partial v_j},-m\le j\le m,j\neq 0$$

(e). 在这小题中我们将编码实现word2vec模型，并且用SGD来训练词向量。首先，在q3_word2vec.py文件中实现归一化每行矩阵的帮助函数。同样在该文件中，补充softmax和negative sample损失值和梯度的代码。然后，实现skip-gram模型的损失和梯度代码。全部完成后，使用命令"python q3_word2vec.py"命令测试代码。

> 如果你选择不实现CBOW模型（也就是(h)题），把代码中的NotImplementedError移除即可。

解：

这道题就是要实现(a)(b)(c)(d)四小题中我们推导出来的公式，只要对前四道题理解到位了，代码其实很简单。矩阵相乘时要注意方向！

答案见代码文件。

(f). 在文件q3_sgd.py中实现SGD优化方法的代码，并且用命令"python q3_sgd.py"测试。

解：

slides里有伪代码，答案见代码文件。

(g). 现在，我们将会下载一些真实数据用我们刚刚实现的代码来训练词向量。我们使用的数据是斯坦福情感树库(SST)，稍后也会将它们应用到简单的情感分析任务中。这部分不需要再写代码，只要运行"python q3_run.py"即可。

> 训练花费的时间取决于你的代码质量，可能会花费很长时间。一个有效的实现大概需要一个小时的训练时间。

脚本运行结束后，会生成q3_word_vectors.png文件来保存词向量，用不超过3个句子简洁的描述你在图片上看到了什么?

解：

答案见代码文件。

(h). 附加题。在文件q3_word2vec.py中实现CBOW模型。

解：

答案见代码文件。

4.Sentiment Analysis

现在，我们使用上题训练好的词向量来完成一个简单的情感分析任务。对SST数据集中的每一个句子，我们使用句子中所有词向量的平均值作为句子的特征，然后尝试预测句子的情感等级，原始数据集中的情感等级是使用真实数值表示的，这里我们使用5个类别表示：

“非常负面” “负面” “中性” “正面” “非常正面”

情感等级在代码中使用0-4之间的数字表示。在这部分，我们将会学习用SGD训练softmax回归器，然后通过训练、调试验证来提高回归器的表现。

(a). 在文件q4_softmaxreg.py中实现句子特征提取器和softmax 回归的代码，然后使用命令"python q4_softmaxreg.py"命令测试。

解：

本题代码中需要注意的时候有正则化选项，所以在计算损失和梯度时需要考虑正则化。

答案见代码文件。

(b). 用不超过3句话解释为什么在分类任务中要引入正则化。（实际上，大多数机器学习任务都需要）

解：

避免训练时过拟合，以及在未知数据上的不适应性。

(c). 在文件q4_sentiment.py中实现选择超参数的代码，也就是实现寻找“最优”的正则参数（惩罚因子），你会如何选择呢？提交你的训练，调试和测试精度，在最多一个句子上验证你的选择方法。

> 调试时你至少能获得30%的精度。

解：

首先我将超参数的范围设定在[0.0, 0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1]之间进行测试，对比结果发现在1e-5和1e-4数量级附近效果较好，于是我又将范围设定为[0.0, 0.00001, 0.00003, 0.00007, 0.0001, 0.0003, 0.0007, 0.001]，最终发现较好的效果在1e-5数量级。

训练，调试，测试精度分别为29.06%，29.79%，28.28%。

(d).画出训练和调试过程中的精度曲线，x轴使用对数规范。这些都应该自动完成（使用代码实现）。完成后你的目录会生成q4_reg_acc.png图片，用不超过三句话简单解释你在图片上看到了什么?

解：

答案见图片文件。