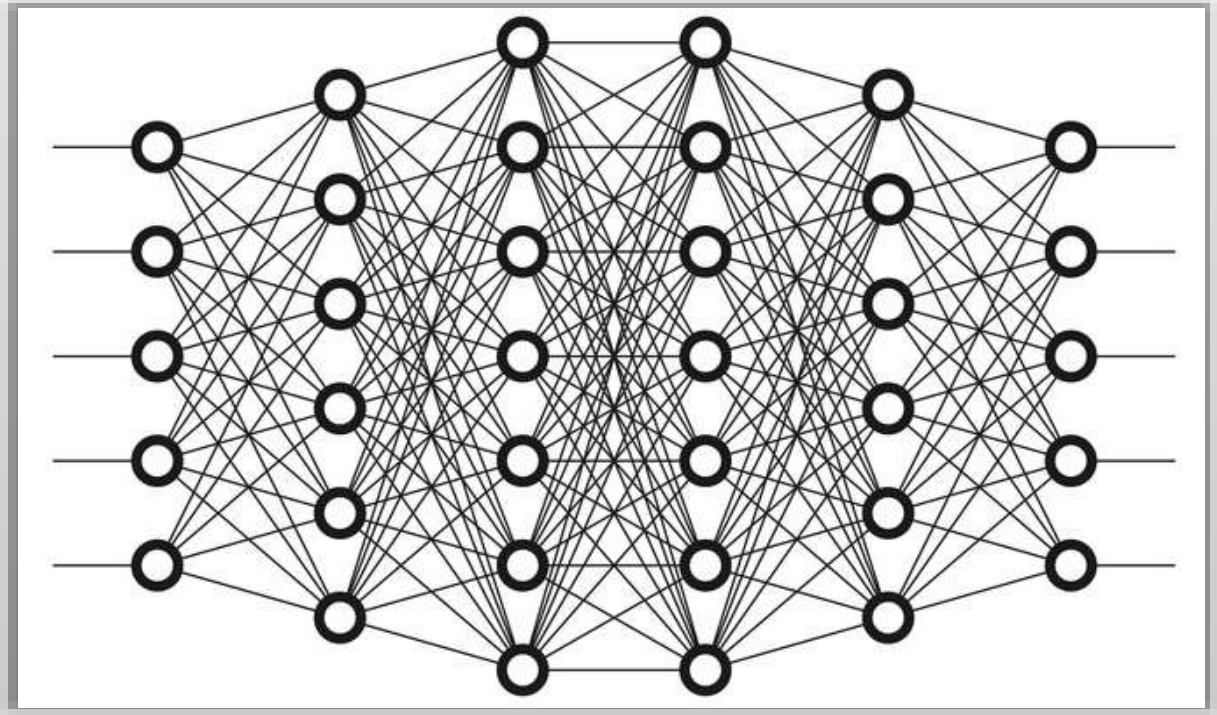
2023 年度云南大学软件学院本科生课程

神经网络与反向传播算法

教 师： 李 劲（**lijin@ynu.edu.cn**）

2023 年 11 月

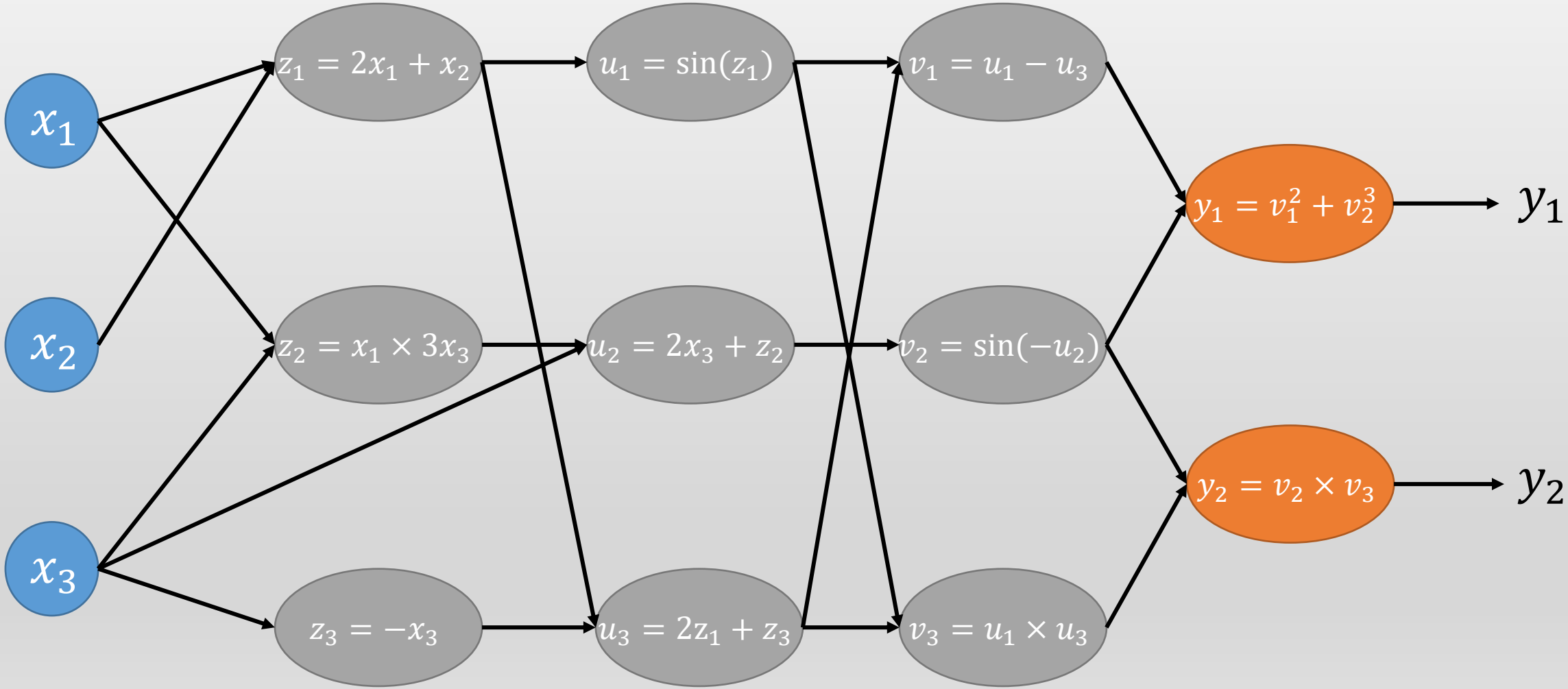


实验3. 神经网络与反向传播算法

3.1 计算图

复合函数的计算图

下面给出了复合函数$(y_1, y_2) = f(x_1, x_2, x_3)$的计算图



复合函数的计算图

实验要求1：基于numpy实现$(y_1, y_2) = f(x_1, x_2, x_3)$的反向传播算法（不允许使用自动微分），程序应能够正确计算函数f的雅克比矩阵.

实验要求2：基于pytorch实现$(y_1, y_2) = f(x_1, x_2, x_3)$的反向传播算法（可以使用torch中自动微分），程序应能够正确计算函数f的雅克比矩阵，即：

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1}, \frac{\partial y_1}{\partial x_2}, \frac{\partial y_1}{\partial x_3} \\ \frac{\partial y_2}{\partial x_1}, \frac{\partial y_2}{\partial x_2}, \frac{\partial y_2}{\partial x_3} \end{bmatrix}$$

(x1,x2,x3)=[2,1,3]

y1 to x:tensor([45.2487, 27.3204, -24.2664])
y2 to x:tensor([25.7484, 0.0614, 23.6467])

输出示例：

Input X	Output Y	Numpy Jacobian Y to X	Pytorch Jacobian Y to X
[1. 1. 1.]	[24.49047994 0.67661701]	[[55.7644109 -9.55245783] [29.05600639 -4.47599576] [-13.63041979 -1.13610056]]	[[55.764523 -9.552442] [29.05603 -4.4759927] [-13.630323 -1.1360837]]
[2. 1. 3.]	[64.08711522 -6.07866752]	[[45.24827007 25.74870621] [27.3203119 0.06141661] [-24.26659743 23.64693485]]	[[55.764523 -9.552442] [29.05603 -4.4759927] [-13.630323 -1.1360837]]
[-1. -1. -3.]	[8.17038442 -0.05974457]	[[-34.72445337 -4.53069838] [-17.09608532 -0.37929324] [5.65862746 -0.4390384]]	[[-34.72444 -4.530696] [-17.09606 -0.3792935] [5.658613 -0.43903807]]

基于Numpy实现的代码框架（供参考）

```
class Computational_Graph:
    def __init__(self) -> None:
        #初始化x1，x2，x3
        #初始化z1，z2，z3，用以存储z1，z2，z3的计算结果
        #初始化u1，u2，u3，用以存储u1，u2，u3的计算结果
        #初始化v1，v2，v3，用以存储v1，v2，v3的计算结果
        #初始化y1，y2，用以存储y1，y2的计算结果

    def function_z1(self,x1,x2):
    def function_z2(self,x1,x3):
    def function_z3(self,x3):
    def function_u1(self,z1):
    def function_u2(self,x3,z2):
    def function_u3(self,z1,z3):
    def function_v1(self,u1,u3):
    def function_v2(self,u2):
    def function_v3(self,u1,u3):
    def function_y1(self,v1,v2):
    def function_y2(self,v2,v3):

    #前向传播
    def forward(self,x1,x2,x3):
    #反向传播
    def backward(self):
```

```
x1=2
x2=1
x3=3
model=Computational_Graph()
y1,y2=model.forward(x1,x2,x3)
x_grad=model.backward()
print(x_grad)
```

3.2 基于前向神经网络的分类预测

FFNN classification with multiple hidden layers

多隐层多分类的前向神经网络

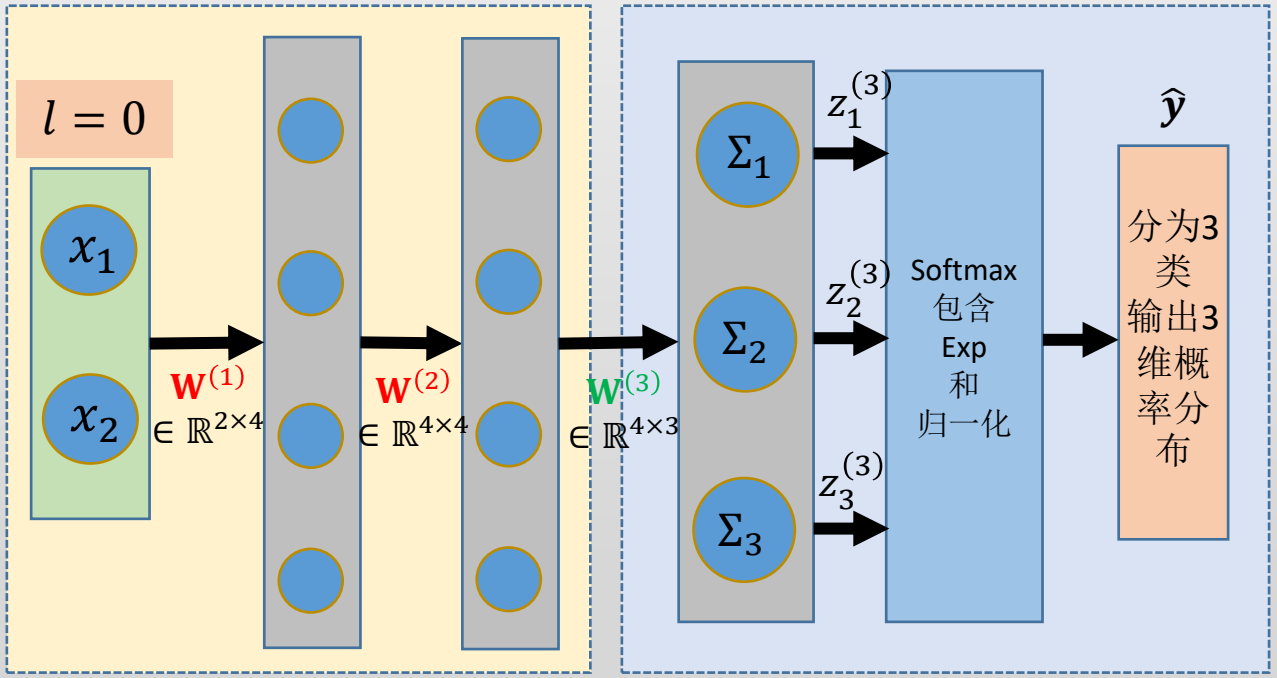
$l=1$ $l=2$

$\mathbf{z}^{(1)} = \mathbf{W}^{(1)\top}\mathbf{x}$ $\mathbf{z}^{(2)} = \mathbf{W}^{(2)\top}\boldsymbol{a}^{(1)}$

$\boldsymbol{a}^{(1)} = \sigma(\mathbf{z}^{(1)})$ $\boldsymbol{a}^{(2)} = \sigma(\mathbf{z}^{(2)})$

$\mathbf{z}^{(3)} = \mathbf{W}^{(3)\top}\boldsymbol{a}^{(2)}$

$$\frac{\exp\left(z_k^{(3)}\right)}{\sum_{j=1}^{3}\exp\left(z_j^{(3)}\right)}$$



神经网络层参数
$\mathbf{W}^{(1)}, \mathbf{W}^{(2)}$
每层最终由非线性激活输出

分类任务层
$\mathbf{W}^{(3)}$
无非线性激活，只有Σ, exp

① $\boldsymbol{\delta}^L = \nabla_a L \odot \sigma'(\mathbf{z}^L)$

（$\mathbf{z}^L$ 是神经网络最后一层隐层输出结果（即 $\boldsymbol{a}^{(L-1)}$）经分类任务模型变换后的结果。

$\boldsymbol{a}^L$ 是$\mathbf{z}^L$ 经任务模型处理后输入到最终损失函数的内容）

$$= \nabla_{\hat{\boldsymbol{y}}}\mathrm{CE}(\boldsymbol{y},\hat{\boldsymbol{y}}) \odot \left(\frac{\exp(\ )}{\sum\exp(\ )}\right)'(\mathbf{z}^{(3)})$$

② $\boldsymbol{\delta}^l = \left(\mathbf{W}^{l+1}\boldsymbol{\delta}^{l+1}\right) \odot \sigma'(\mathbf{z}^l)$

③ $\frac{\partial L}{\partial \mathbf{W}^{(2)}} = \delta_j^l a_i^{l-1}$

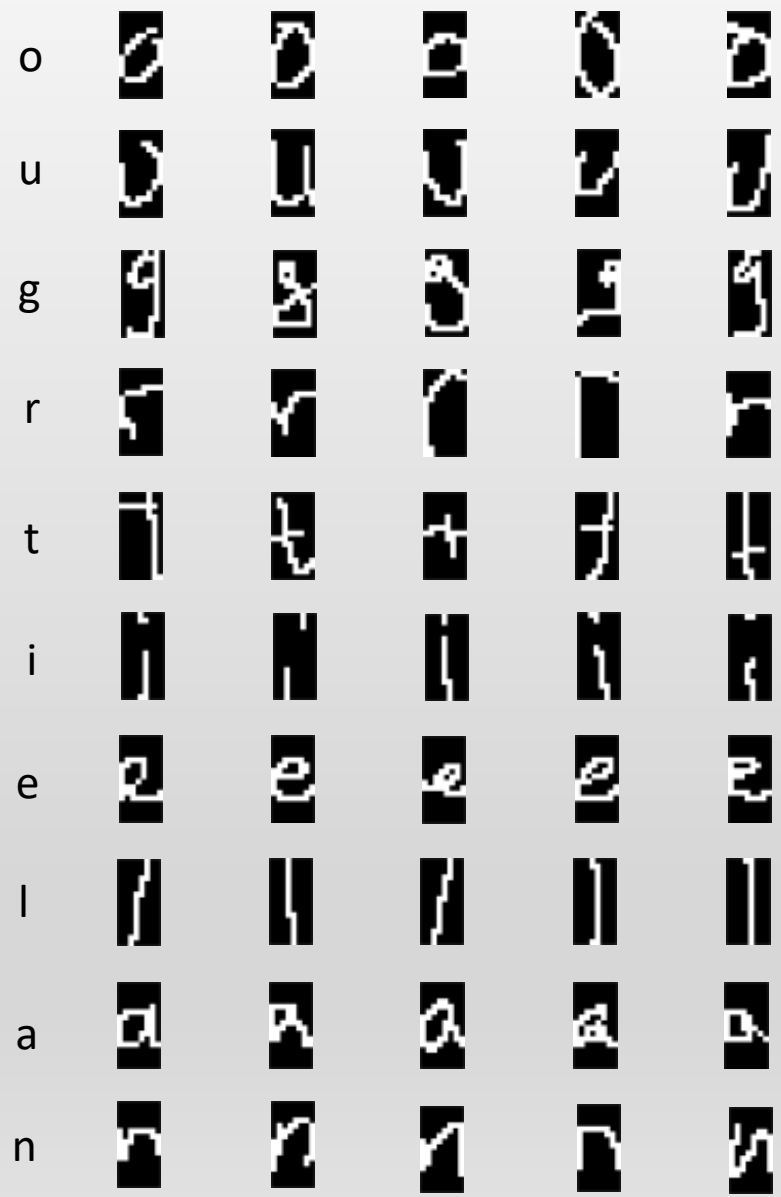
④ $\frac{\partial L}{\partial \boldsymbol{b}^{(2)}} = \delta_j^l$

实验数据集

Optical Character Recognition (OCR) data set. This data includes images of all 26 handwritten letters; our subset will include only the letters “a,” “e,” “g,” “i,” “l,” “n,” “o,” “r,” “t,” and “u.” The handout contains three data sets drawn from this data:

(1) a small data set with 60 samples per class (50 for training and 10 for testing),
(2) a medium data set with 600 samples per class (500 for training and 100 for testing), and
(3) a large data set with 1000 samples per class (900 for training and 100 for testing).

Each data set (small, medium, and large) consists of two csv files—train and validation. Each row contains 129 columns separated by commas. The first column contains the label and columns 2 to 129 represent the pixel values of a 16 × 8 image in a row major format. Label 0 corresponds to “a,” 1 to “e,” 2 to “g,” 3 to “i,” 4 to “l,” 5 to “n,” 6 to “o,” 7 to “r,” 8 to “t,” and 9 to “u.”



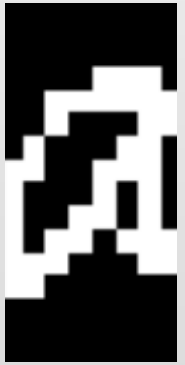
实验数据集

标签列

特征列

	A	B	C	D	E	F	G	H	I	J	K	L
1	6	0	0	0	0	0	0	0	0	0	0	0
2	9	0	0	0	0	0	0	0	0	0	0	0
3	3	0	0	0	0	0	1	0	0	0	0	0
4	7	0	0	0	0	0	0	0	0	0	0	0
5	8	0	0	0	0	0	1	0	0	0	0	0
6	1	0	0	0	0	0	0	0	0	0	0	0
7	4	0	0	0	0	0	1	0	0	0	0	0
8	4	0	0	0	1	0	0	0	0	0	0	0
9	5	0	0	0	0	0	0	0	0	0	0	0
10	3	0	0	0	0	0	0	0	0	1	1	1
11	6	0	0	0	0	0	0	0	0	0	0	0
12	4	0	0	0	0	0	1	0	0	0	0	0
13	3	0	0	0	1	0	0	0	0	0	0	0
14	7	0	0	0	0	0	0	0	0	0	0	0
15	9	0	0	0	0	0	0	0	0	0	0	0
16	7	0	0	0	0	1	1	1	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0
18	7	0	0	0	0	0	1	1	1	0	0	0
19	9	0	0	0	0	0	0	0	0	0	0	0
20	9	0	0	0	0	0	0	0	0	0	0	0

16 × 8 image in a row major format



实验要求

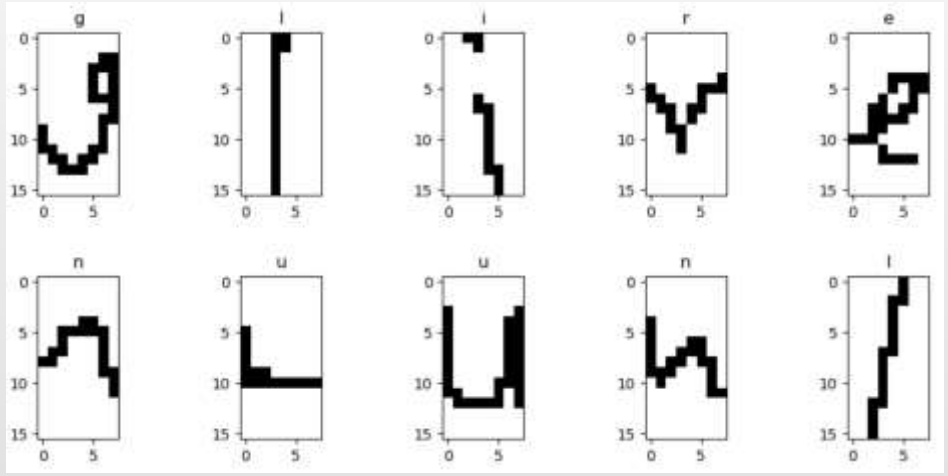
In this assignment, you will implement a two-hidden-layer neural network with a sigmoid activation function for the hidden layer, and a softmax on the output layer.

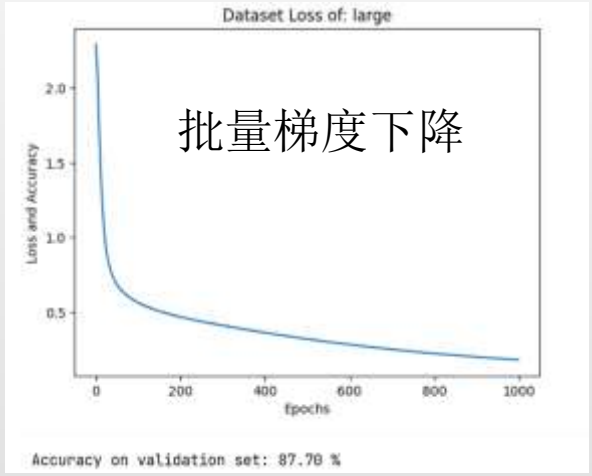
Let the input vectors x be of length M, the hidden layer z consist of D hidden units, and the output layer yˆ be a probability distribution over K classes. That is, each element yk of the output vector represents the probability of x belonging to the class k

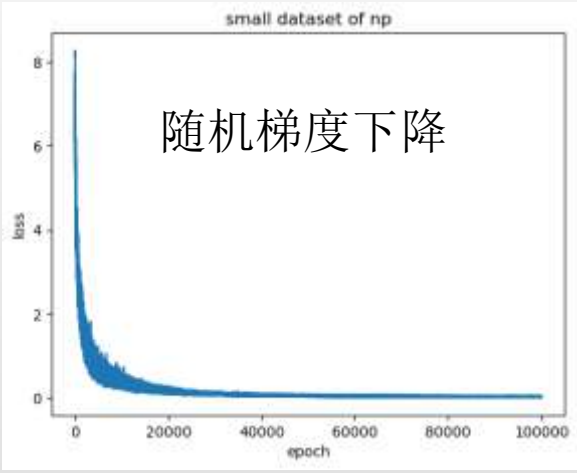
- Do not use any machine learning libraries. You may and please do use NumPy.
- Try to "vectorize" your code as much as possible. In Python, you want to avoid for-loops and instead rely on NumPy calls to perform operations such as matrix multiplication, transpose, subtraction, etc. over an entire NumPy array at once. This is much faster; using NumPy over list can speed up your computation by 200x!
- You'll want to pay close attention to the dimensions that you pass into and return from your functions.

实验要求

1. 分别读入small、 medium、 large三个数据集，每个数据集从训练集中随机抽出10个样本进行展示（黑白图像）

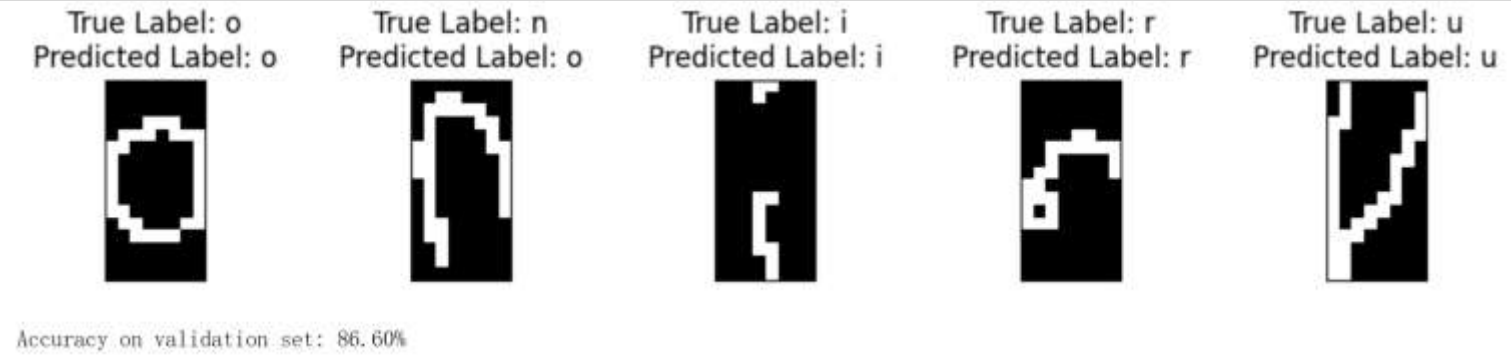






2. 基于Numpy实现2-隐层前向神经网络的前向计算及反向传播梯度计算，在此基础之上，分别利用small、 medium、 large三个数据集中的训练集对模型进行训练，绘制出loss函数变化情况。模型训练完成后，分别在三个数据集的测试集上评价所训练出来的模型在测试集上预测准确率。

3. 最后，在small、 medium、 large三个数据集的每个测试集上给出5个示例，即: (测试图像+模型预测标签)



实验要求

1. 分别读入small、 medium、 large三个数据集，每个数据集从训练集中随机抽出10个样本进行展示（黑白图像）

2. 基于Pytorch实现一个2-隐层的前向神经网络模型。在此基础之上，分别利用small、 medium、 large三个数据集中的训练集对模型进行训练，绘制出loss函数变化情况。模型训练完成后，分别在三个数据集的测试集上评价所训练出来的模型在测试集上预测准确率。

3. 最后，在small、 medium、 large三个数据集的每个测试集上给出5个示例，即: (测试图像+模型预测标签)

3.3 基于前向神经网络的回归预测

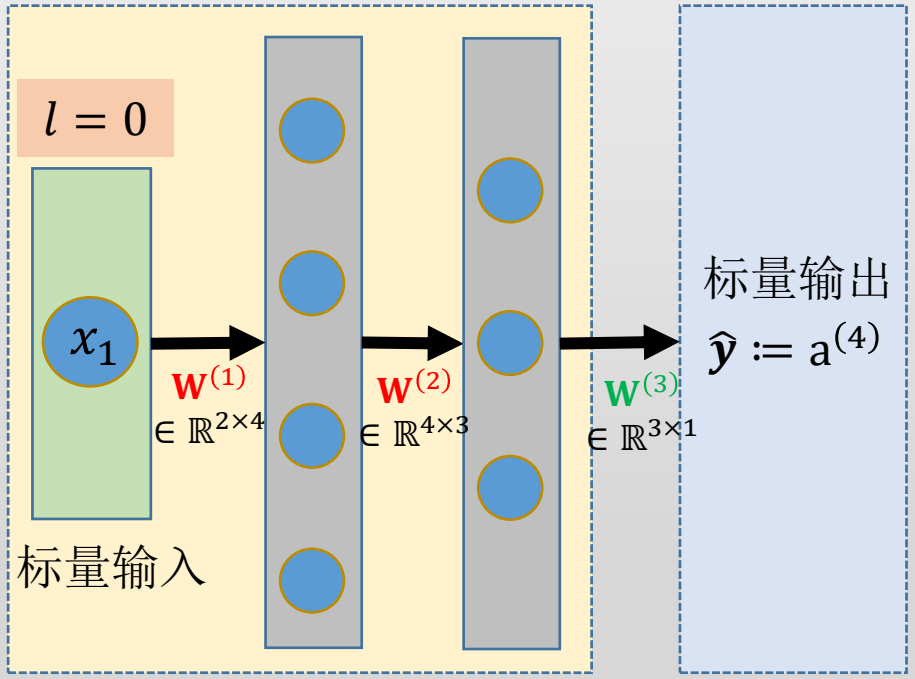
FFNN Regression with multiple hidden layers

多隐层回归预测的前向神经网络

$l=1$ $l=2$

$\mathbf{z}^{(1)}=\mathbf{W}^{(1)^\top}\mathbf{x}$ $\mathbf{z}^{(2)}=\mathbf{W}^{(2)^\top}\boldsymbol{a}^{(1)}$ $\mathrm{z}^{(3)}=\mathbf{W}^{(3)^\top}\boldsymbol{a}^{(2)}$

$\boldsymbol{a}^{(1)}=\sigma(\mathbf{z}^{(1)})$ $\boldsymbol{a}^{(2)}=\sigma(\mathbf{z}^{(2)})$ $\mathrm{a}^{(3)}=1\times\mathrm{z}^{(3)}$



神经网络层参数
$\mathbf{W}^{(1)},\mathbf{W}^{(2)}$

回归任务层
$\mathbf{W}^{(3)}$

① $\boldsymbol{\delta}^{L}=\nabla_a L\odot\sigma'(\mathbf{z}^{L})$

$\boldsymbol{\delta}^{(3)}=\nabla_a L\odot\sigma'(\mathbf{z}^{(3)})$

$=\nabla_{\hat{y}}L(y,\hat{y})\odot(a^{(3)}=\mathrm{z}^{(3)})'(\mathrm{z}^{(3)})$

$=(y-\hat{y})\odot1$

$=(y-\hat{y})$

② $\boldsymbol{\delta}^{l}=(\mathbf{W}^{l+1}\boldsymbol{\delta}^{l+1})\odot\sigma'(\mathbf{z}^{l})$

$\boldsymbol{\delta}^{(2)}=(\mathbf{W}^{(3)}\boldsymbol{\delta}^{(3)})\odot\mathrm{sigmoid}'(\mathbf{z}^{(2)})$

$\boldsymbol{\delta}^{(1)}=(\mathbf{W}^{(2)}\boldsymbol{\delta}^{(2)})\odot\mathrm{sigmoid}'(\mathbf{z}^{(1)})$

③ $\frac{\partial L}{\partial\mathbf{W}^{(l)}}=\boldsymbol{\delta}^{(l)}\boldsymbol{a}^{(l-1)}$

$\frac{\partial L}{\partial\mathbf{W}^{(3)}}=\boldsymbol{\delta}^{(3)}\boldsymbol{a}^{(2)}=(y-\hat{y})\boldsymbol{a}^{(2)};\quad\frac{\partial L}{\partial\mathbf{W}^{(2)}}=\boldsymbol{\delta}^{(2)}\boldsymbol{a}^{(1)}$

$\frac{\partial L}{\partial\mathbf{W}^{(1)}}=\boldsymbol{\delta}^{(1)}\boldsymbol{a}^{(0)}$

④ $\frac{\partial L}{\partial\boldsymbol{b}^{(l)}}=\boldsymbol{\delta}^{(l)}$

$\frac{\partial L}{\partial\boldsymbol{b}^{(3)}}=\boldsymbol{\delta}^{(3)};\quad\frac{\partial L}{\partial\boldsymbol{b}^{(2)}}=\boldsymbol{\delta}^{(2)};\quad\frac{\partial L}{\partial\boldsymbol{b}^{(1)}}=\boldsymbol{\delta}^{(1)}$

datasets

任务1. 读取数据集并可视化。

将 data_train.csv ($N = 2400$) 随机划分为训练集（分布内）:1800；测试集（分布内）: 600。

将 data_valid.csv +data_test.csv合并后作为测试集（分布外）。

使用不用颜色将上述3个数据集绘制出来。

任务2. 构建模型及预测

分别（1）基于numpy（需自己实现前向和反向计算）、（2）基于pytorch，实现一个2-隐层的前向神经网络。

利用训练集（分布内）训练神经网络模型，并在测试集（分布内）和测试集（分布外）上进行预测，并输出RMSE。

绘制训练好的神经网络回归曲线