Handson 8

CIE5141 Computer Vision in Construction

b07501113 林庭瑄

天天開心～

Feedforward Neural Network

Single-layer Perceptron(20pt)

```
# read the Excel file
df = pd.read_excel('data.xlsx')
# select the list where we are interested.
# <hint>: you need to change the format into float for the later
calculation
lst=df.values.tolist()
# change the list to NumPy array
arr=np.array(lst)
# print the NumPy array
print(arr)
```

```
# Data setting
b = 1
x = np.hstack(([row[1:4] for row in arr],np.ones((len(arr),1))))
y = [row[0] for row in arr]
w = [0]*x.shape[1]
# Hyperparameter setting
alpha = 0.5
step = 200
```

Single-layer Perceptron with batch(20pt)

```
# Data setting
b = 1
x = np.hstack(([row[1:4] for row in arr],np.ones((len(arr),1))))
y = [row[0] for row in arr]
w = np.array([0.0]*x.shape[1])
# Hyperparameter setting
alpha = 0.5
step = 200
```

```
# Training loop
# <hint>steps are very similar to above one. The only difference is
that you don't need to calculate one by one.
for t in tqdm(range(step)):
   y_pred= np.heaviside(np.dot(x, w), 0)
   w += alpha * np.dot(y - y_pred,x)
```

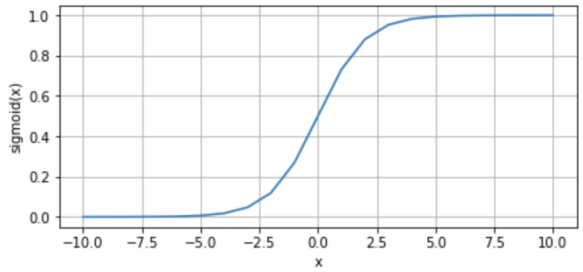
Single-layer Perceptron with different activation function

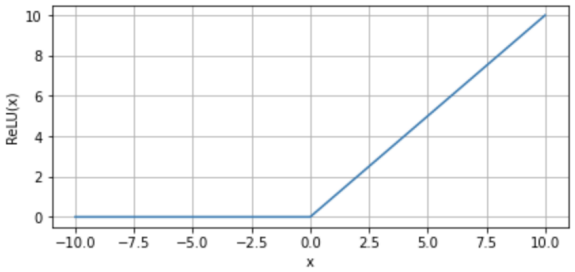
Define and plot the function (10pt)

```
# define the activation function
def ReLU(x):
   for i in range(len(x)):
       if x[i] <= 0:
           x[i]=0;
   return x

def sigmoid(x):
   y=[0.0]*len(x)
   for i in range(len(x)):
       try:
           y[i]=1/(1+math.exp(-x[i]))
       except OverflowError:
           y[i]=0
   return y
```

```
# Plot the two activation functions with x from -10 to 10
# <hint>some useful funtions in plt: plot, subplot, xlabel, ylebel,
grid, figsize
x_relu=np.arange(-10.0,11.0)
y_relu=ReLU(x_relu)

x_sigmoid=np.arange(-10.0,11.0)
y_sigmoid=sigmoid(x_sigmoid)

plt.figure(figsize=(15,3))
plt.subplot(121,xlabel="x",ylabel="sigmoid(x)")
plt.xlim([-10,10])
plt.plot(x_sigmoid, y_sigmoid)
plt.grid()
plt.subplot(122,xlabel="x",ylabel="ReLU(x)")
plt.xlim([-10,10])
```

```
plt.plot(x_relu, y_relu)
plt.grid()
plt.show()
```





Single-layer Perceptron with sigmoid(20pt)

```
# Training data
# Data setting
b = 1
x = np.hstack(([row[1:4] for row in arr],np.ones((len(arr),1))))
y = [row[0] for row in arr]
w = [0.0]*x.shape[1]
# Hyperparameter setting
alpha = 0.5
step = 200

# Training data
for t in tqdm(range(step)):
    y_pred= sigmoid(np.dot(x, w))
    w += alpha * np.dot(np.array(y) - y_pred,x)
```

Single-layer Perceptron with ReLU(20pt)

```
# Training data
# Data setting
b = 1
x = np.hstack(([row[1:4] for row in arr],np.ones((len(arr),1))))
y = [row[0] for row in arr]
w = [0.0]*x.shape[1]
# Hyperparameter setting
alpha = 0.5
```

step = 200

```
# Training data
for t in tqdm(range(step)):
    y_pred= ReLU(np.dot(x, w))
    w += alpha * np.dot(np.array(y) - y_pred,x)
```

Bouns part (10pt)

```
# Training data
# Data setting
b = 1
x = np.hstack(([row[1:4] for row in arr],np.ones((len(arr),1))))
y = [row[0] for row in arr]
w = [0.0]*x.shape[1]
# Hyperparameter setting
alpha = 0.001
step = 200

for t in tqdm(range(step)):
   y_pred= ReLU(np.dot(x, w))
   w += alpha * np.dot(np.array(y) - y_pred,x)

# To see the prediction
result=ReLU(np.dot(x, w))
print(alpha,step,result)
```