Predict the use of shared bicycles

陈梓烨

Deep Learning Foundation

Outline

Contents

Back Propagation Neural Network

- Computing systems inspired by the biological neural networks that constitute animal brains

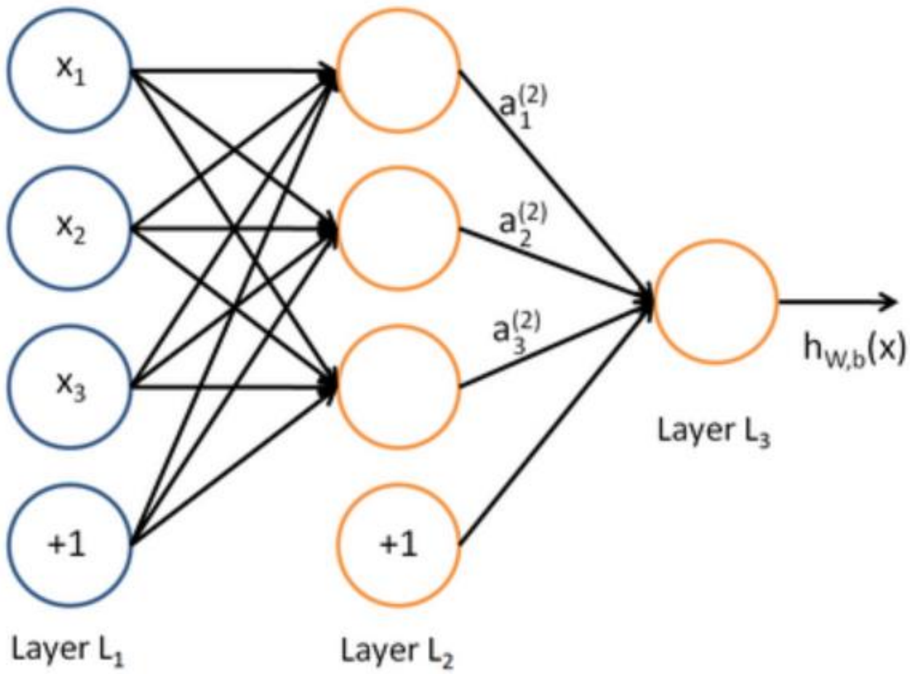


Fig.1 Figure of Three-layer Neural Network

Convolutional Neural Network (CNN)

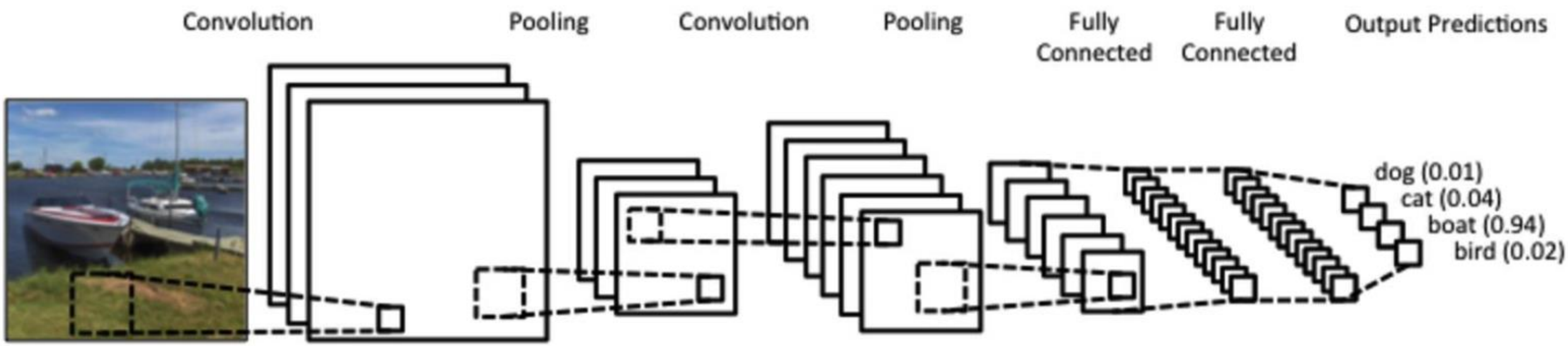


Fig.3 Convolutional neural network

Sensitive with image

Recurrent Neural Network(RNN)

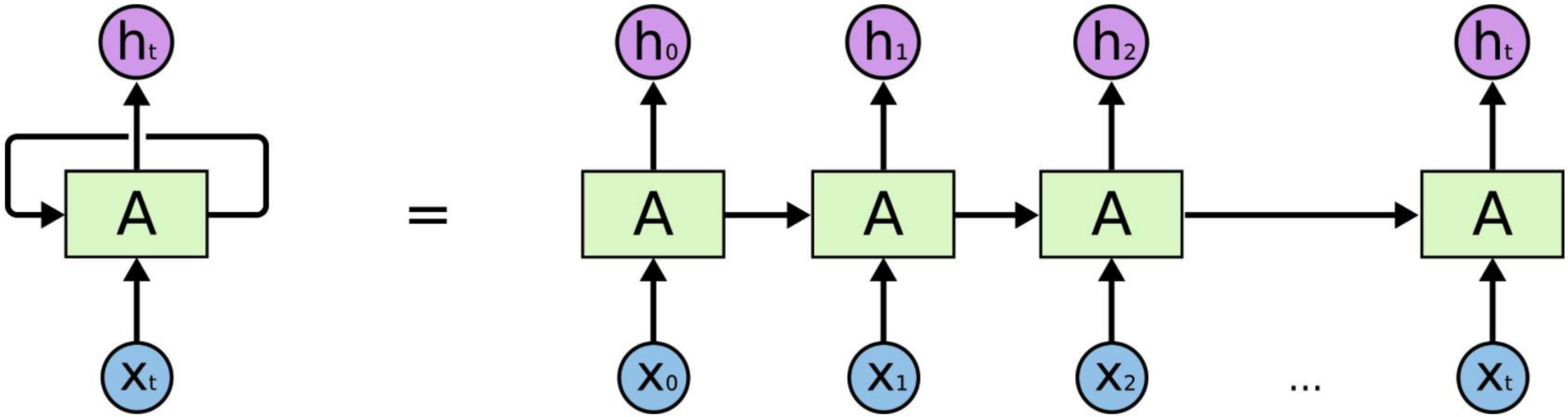


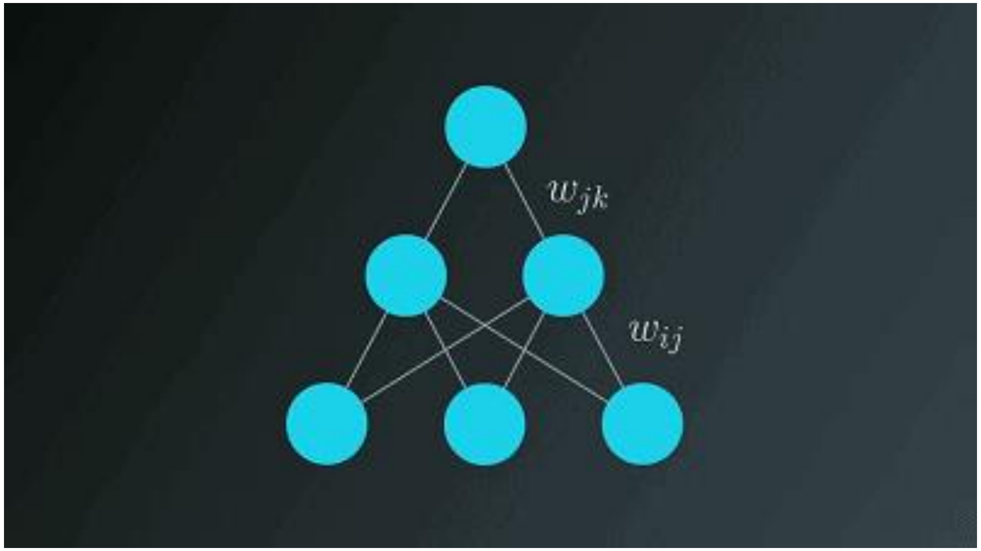
Fig.2 Recurrent Neural Network

Sensitive with sequential data

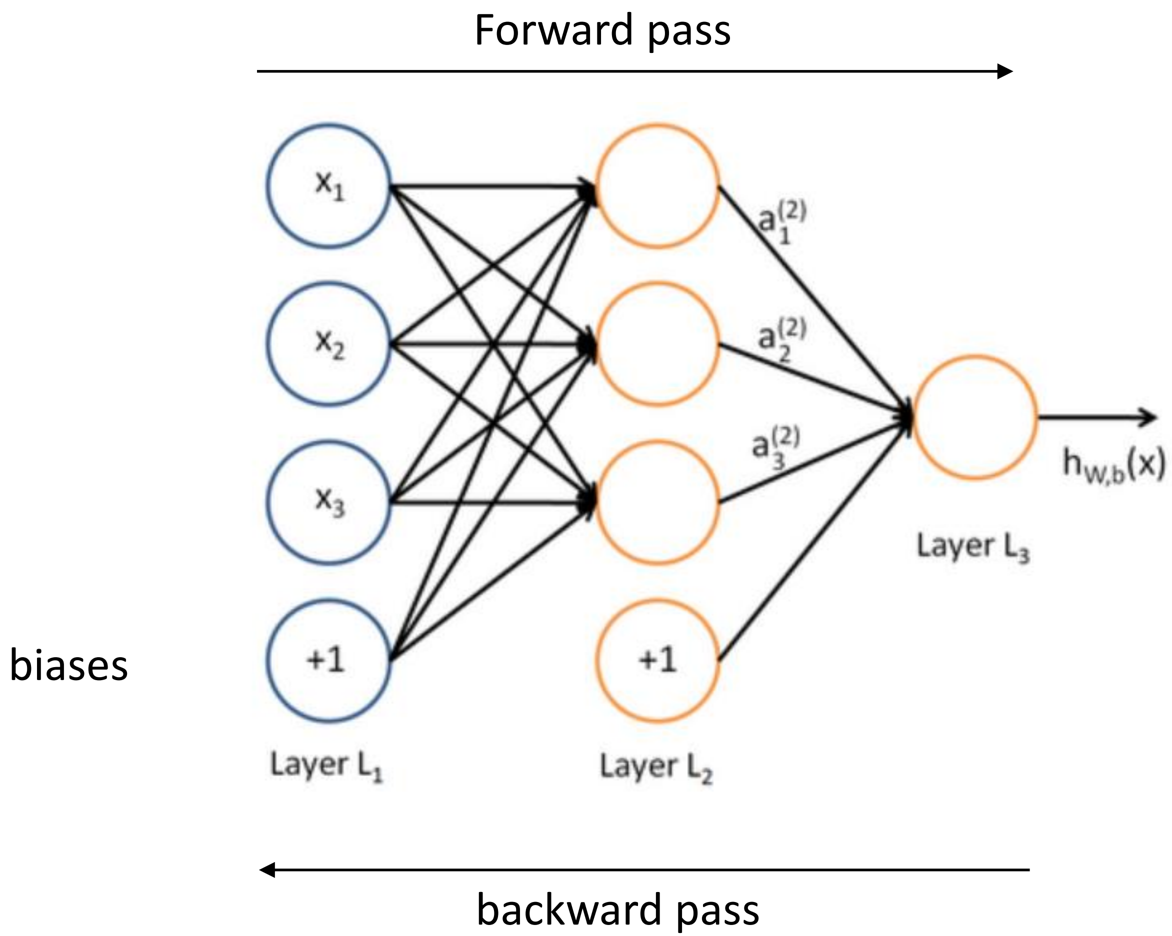
Contents

Back propagation neural network

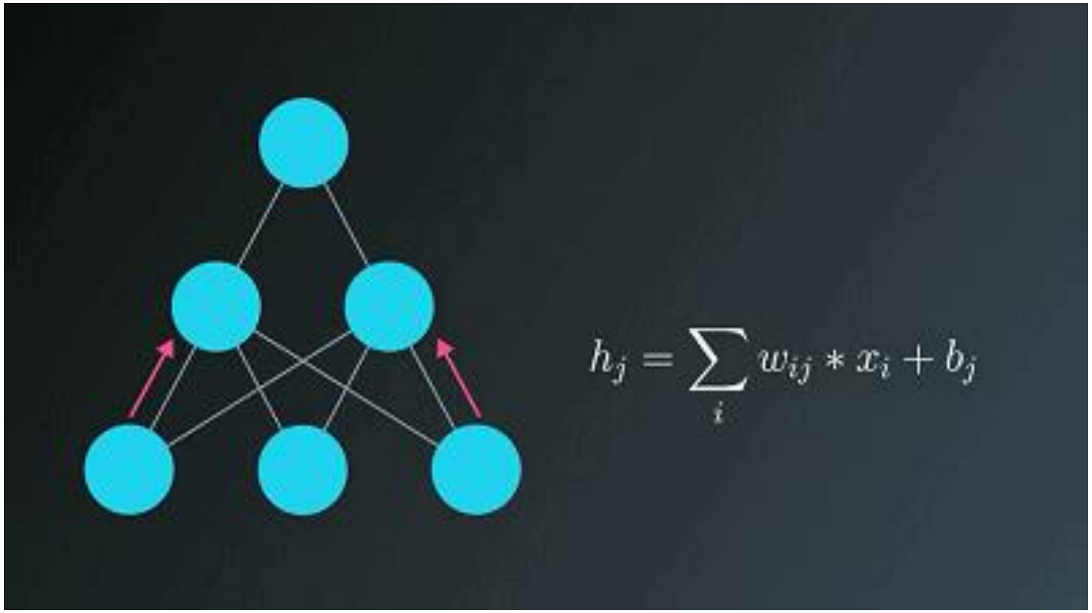
实验要求：三层神经网络（输入层，隐藏层，输出层）



Back propagation neural network



Forward pass



Forward pass

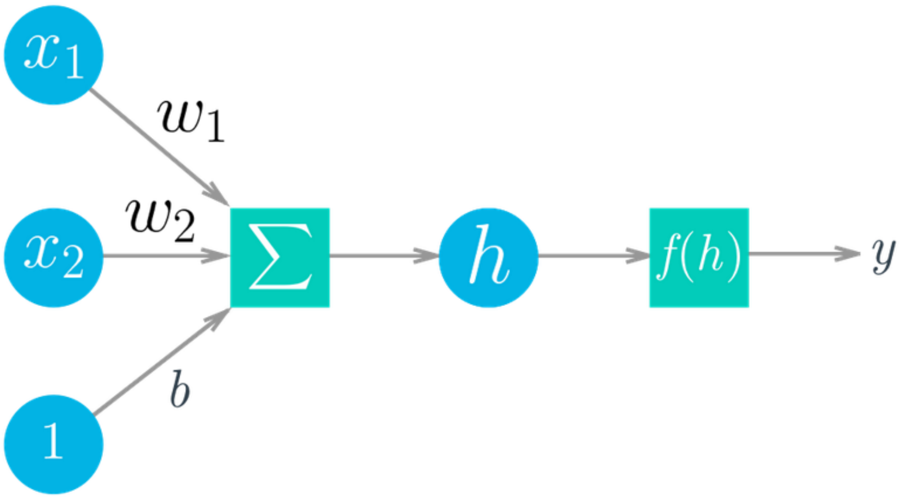


Fig.3 神经网络示意图

在这个架构中f(h)称为激活函数，这个函数可以为很多不同的函数，例如如果让f(h) = h。则网络的输出为：

$$y = \sum_i w_i x_i + b$$

Forward pass

将数据从 $(-\infty,\infty)$ 映射到 (0,1)

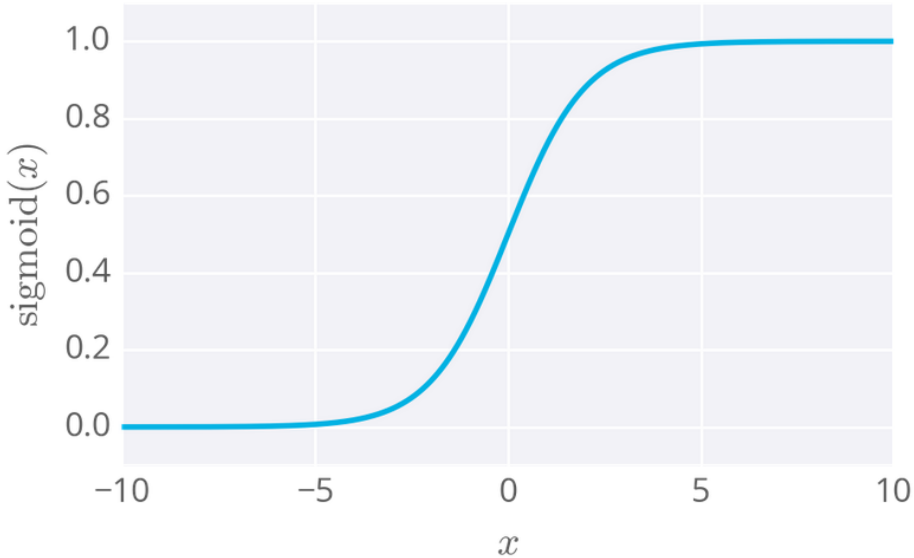
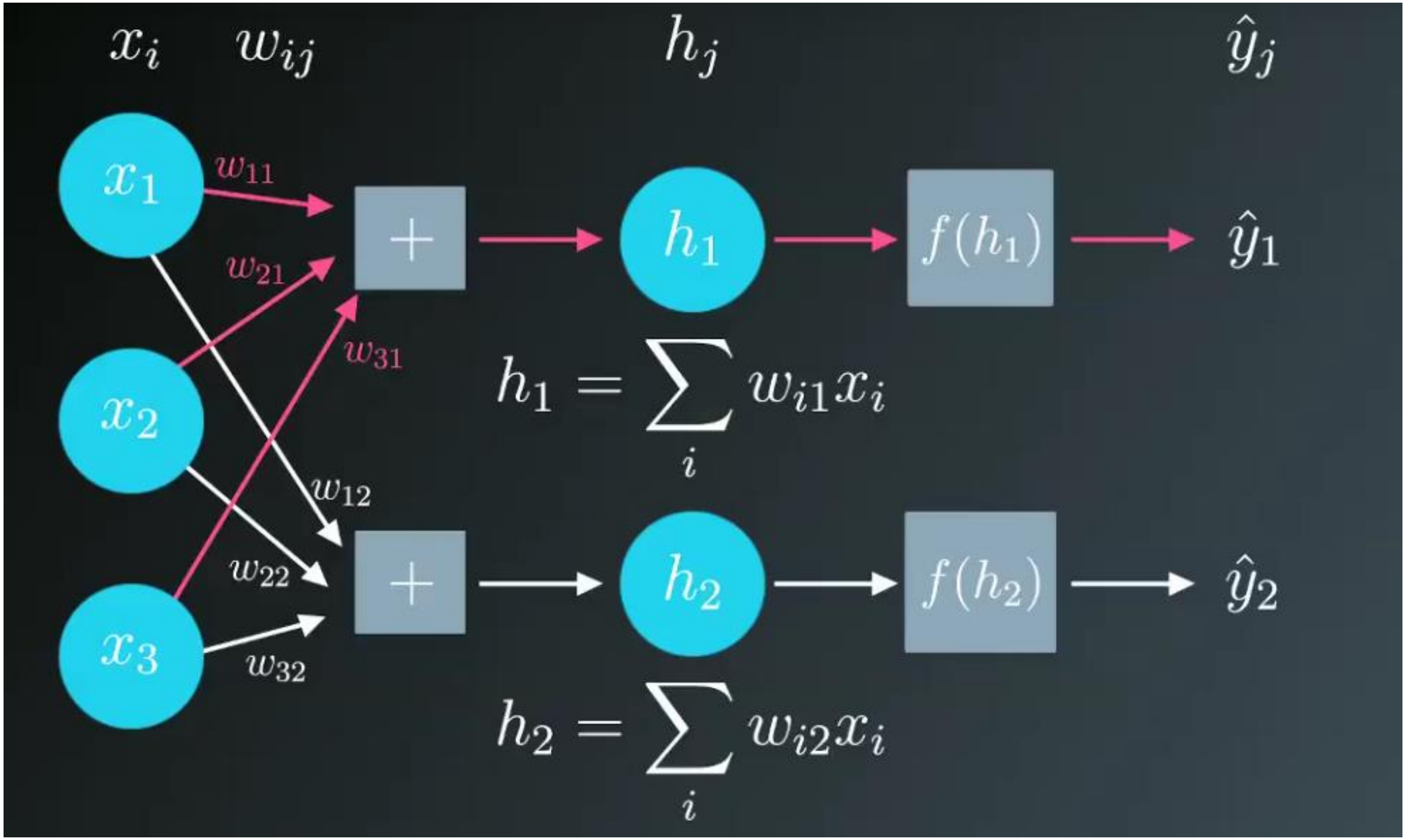


Fig.3 sigmoid函数

公式：

$$\mathrm{sigmoid}(x) = 1/(1 + e^{-x})$$

Forward pass

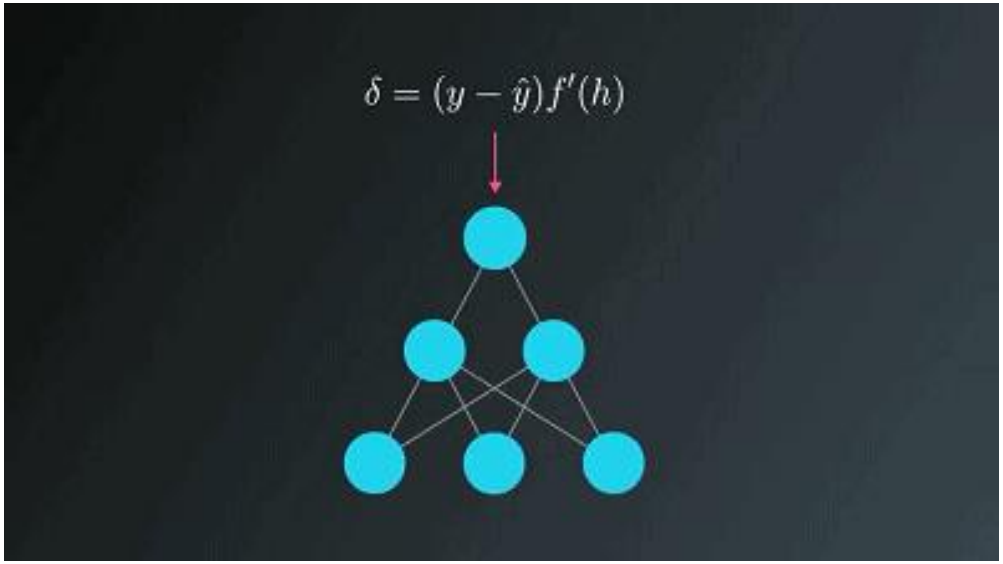


Forward pass

Loss function

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

Backward pass



Backward pass

Loss function：

$$J(W,b)=\frac{1}{m}\sum_{i=1}^{m}J(W,b;x^{i},y^{i})+\frac{\lambda}{2}\sum_{l=1}^{nl-1}\sum_{i=1}^{S_l}\sum_{j=1}^{S_l+1}\left(W_{ji}^{(l)}\right)^{2}$$

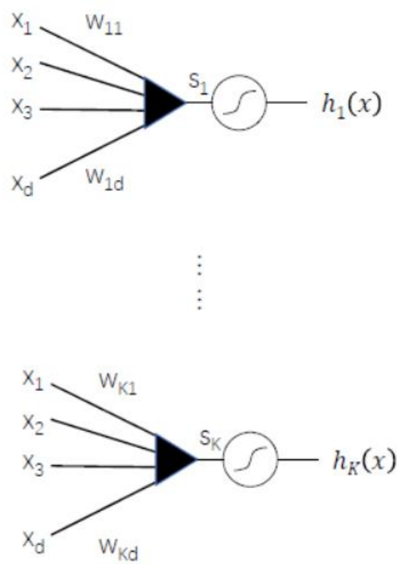
其中：

$$\mathrm{J}(W,b;x,y)=\frac{1}{2}\left\|h_{wb}(x)-y\right\|^{2}$$

Backward pass

对于二分类问题，使用sigmoid函数。

对于K分类问题（K>2），使用softmax回归。



$$\left\{\begin{aligned} & s_j = \sum_{i=1}^{d} w_{ji} x_i + w_{j0} x_0 = \sum_{i=0}^{d} w_{ji} x_{ji} = \boldsymbol{W}_j \boldsymbol{x} \\ & t_j(\boldsymbol{x}) = \exp(s_j) \\ & \mathrm{p}_j(y = j) = \frac{t_j(x)}{\sum_j t_j(x)} \end{aligned}\right. \quad , \quad j \in [1, K]$$

简而言之：

$$p(y = j \mid x, \boldsymbol{W}_j) = h_j(\boldsymbol{x}) = softmax(\boldsymbol{W}_j \boldsymbol{x}) = \frac{\exp(\boldsymbol{W}_j \boldsymbol{x})}{\sum_{i=1}^{K} \exp(\boldsymbol{W}_j \boldsymbol{x})}$$

Backward pass

似然函数对比

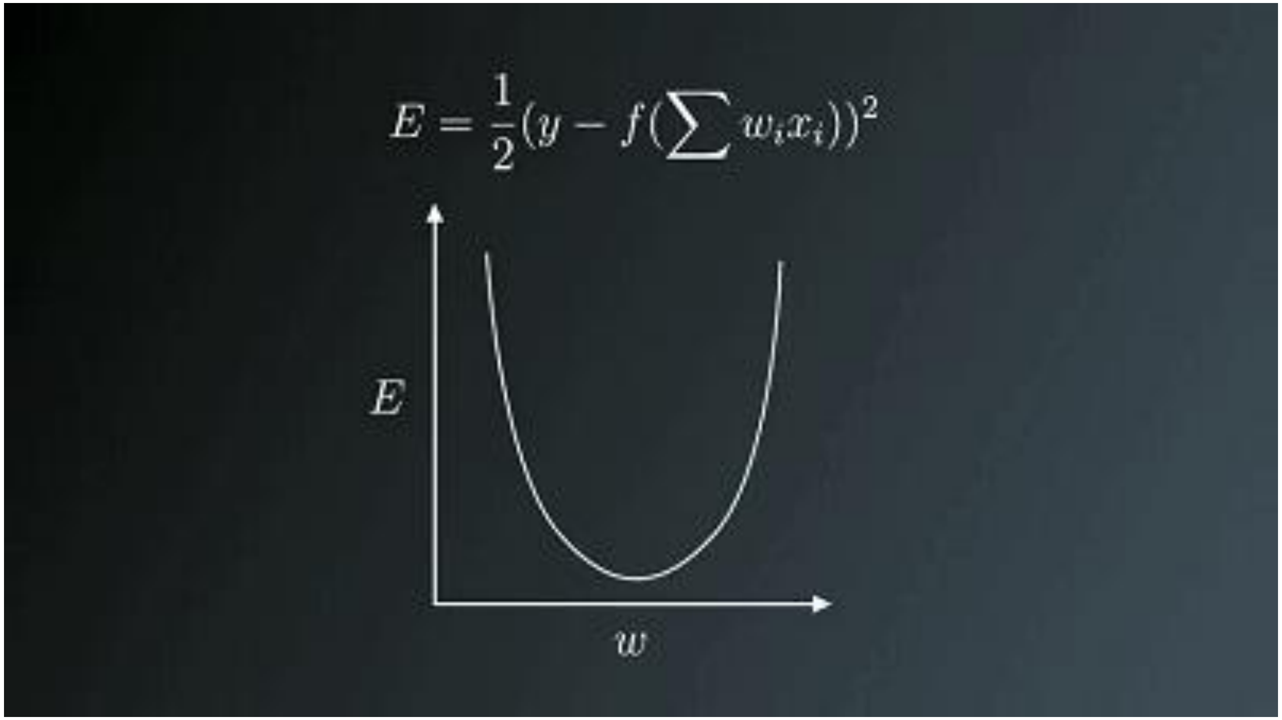
假设数据集中有N个样本 $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$，对应的标签集合为 $\{y_1, \ldots, y_n\}$

逻辑回归： $$likelihood = \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n) = \prod_{n=1}^{N} h(\boldsymbol{x}_n)^{y_n}\left(1 - h(\boldsymbol{x}_n)\right)^{1-y_n}$$

softmax回归： $$likelihood = \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n) = \prod_{n=1}^{N} h(\boldsymbol{x}_n)^{y_n}$$

更新过程相似，计算负对数似然，并对权重求导数后使用梯度下降法更新。

Backward pass



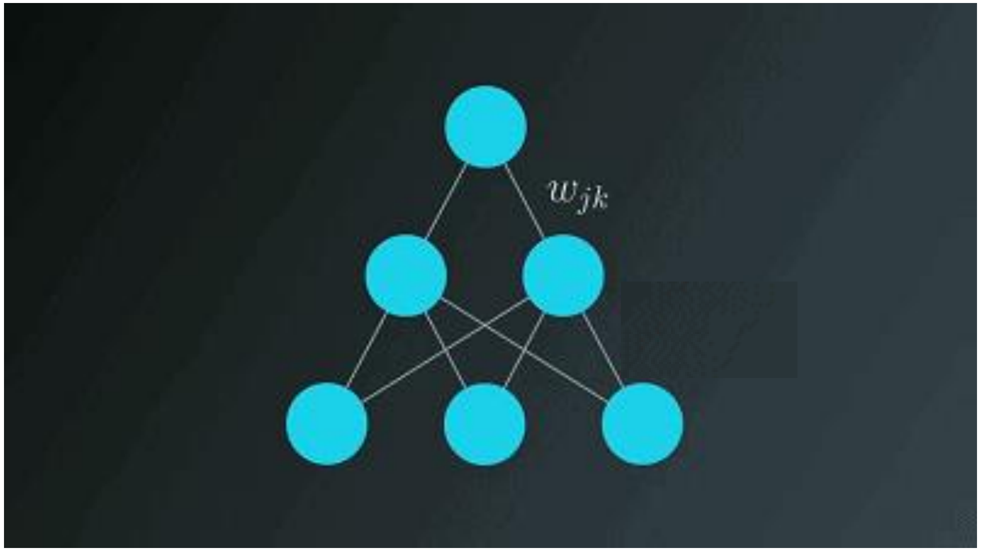
Backward pass

$$w_i = w_i + \Delta w_i$$

$$\Delta w_i \propto -\frac{\partial E}{\partial w_i} \longrightarrow \text{THE GRADIENT}$$

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

Backward pass



Backward pass

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i}\frac{1}{2}(y-\hat{y})^2$$

w_i为最后一层权重

$$= \frac{\partial}{\partial w_i}\frac{1}{2}(y-f(h))^2$$

通过链式求导

$$\frac{\partial}{\partial z}p(q(z)) = \frac{\partial p}{\partial q}\frac{\partial q}{\partial z}$$

Backward pass

$$\hat{y} = f(h) \quad \text{where} \quad h = \sum_i w_i x_i$$

$$\frac{\partial E}{\partial w_i} = -(y - \hat{y})\frac{\partial \hat{y}}{\partial w_i}$$

$$= -(y - \hat{y}) f'(h) \frac{\partial}{\partial w_i} \sum w_i x_i$$

Backward pass

$$\frac{\partial}{\partial w_i}\sum_i w_i x_i$$

$$= \frac{\partial}{\partial w_1}[w_1 x_1 + w_2 x_2 + \ldots + w_n x_n]$$

$$= x_1 + 0 + 0 + 0 + \ldots$$

$$\frac{\partial}{\partial w_i}\sum_i w_i x_i = x_i$$

Backward pass

$$\frac{\partial E}{\partial w_i} = -(y - \hat{y})f'(h)x_i$$

$$\downarrow$$

$$\delta = (y - \hat{y})f'(h)$$

$$w_i = w_i + \eta\delta x_i$$

Backward pass

h+1层的误差为δ_k^{h+1}， h层节点 j 的误差即为h+1层误差乘以两层间的权重矩阵和激活函数的导数

$$\delta_j^h = \sum W_j \delta_k^{h+1} f'(h_j)$$

梯度下降与之前相同，只是用当前层的误差

$$\Delta w_{ij} = \eta \delta_j^h x_i$$

Backward pass

当f（x） = sigmoid（x）时：

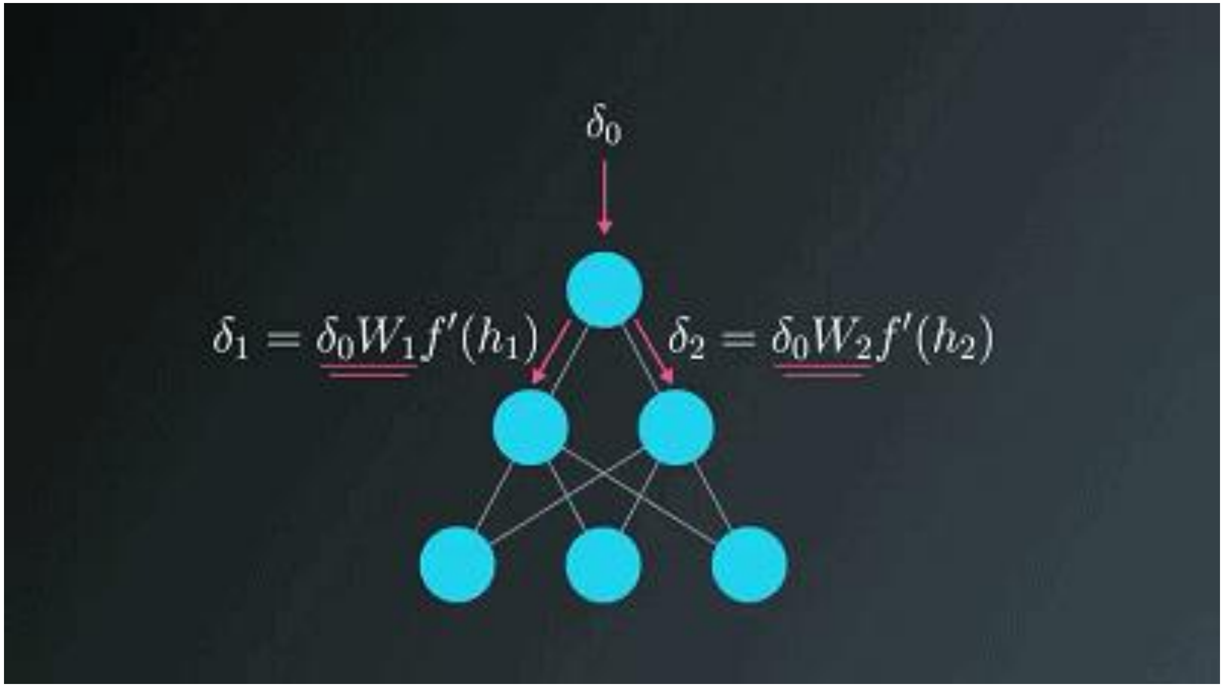
$$
\begin{aligned}
\frac{\partial E_d}{\partial net_j} &= \sum_{k \in Downstream(j)} \frac{\partial E_d}{\partial net_k} \frac{\partial net_k}{\partial net_j} \\
&= \sum_{k \in Downstream(j)} -\delta_k \frac{\partial net_k}{\partial net_j} \\
&= \sum_{k \in Downstream(j)} -\delta_k \frac{\partial net_k}{\partial a_j} \frac{\partial a_j}{\partial net_j} \\
&= \sum_{k \in Downstream(j)} -\delta_k w_{kj} \frac{\partial a_j}{\partial net_j} \\
&= \sum_{k \in Downstream(j)} -\delta_k w_{kj} a_j (1 - a_j) \\
&= -a_j (1 - a_j) \sum_{k \in Downstream(j)} \delta_k w_{kj}
\end{aligned}
$$

Backward pass

$$
\begin{aligned}
\frac{\partial \mathrm{J}(W,b)}{\partial W_{ji}^{(l)}} &= \frac{1}{m}\sum_{t=1}^{m}\frac{\partial J(W,b;x^t,y^t)}{\partial W_{ji}^{(l)}} + \lambda W_{ji}^{(l)} \\
&= \frac{1}{m}\sum_{t=1}^{m}\left(\frac{\partial J(W,b;x^t,y^t)}{\partial Z_j^{l+1}} \times \frac{\partial Z_j^{l+1}}{\partial W_{ji}^{(l)}}\right) + \lambda W_{ji}^{(l)} \\
&= \frac{1}{m}\sum_{t=1}^{m}\left(\delta_j^{l+1} \times \frac{\partial Z_j^{l+1}}{\partial W_{ji}^{(l)}}\right) + \lambda W_{ji}^{(l)} \\
&= \frac{1}{m}\sum_{t=1}^{m}\left(\delta_j^{l+1} \times \frac{\partial\left(\sum_{k=1}^{S_l}\left(W_{jk}^{l}a_k^{l}\right) + b_i^{l}\right)}{\partial W_{ji}^{(l)}}\right) + \lambda W_{ji}^{(l)} \\
&= \frac{1}{m}\sum_{t=1}^{m}\left(\delta_j^{l+1} \times a_i^{l}\right) + \lambda W_{ji}^{(l)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial \mathrm{J}(W,b)}{\partial b_i^{l}} &= \frac{1}{m}\sum_{t=1}^{m}\frac{\partial J(W,b;x^t,y^t)}{\partial b_i^{l}} \\
&= \frac{1}{m}\sum_{t=1}^{m}\left(\frac{\partial J(W,b;x^t,y^t)}{\partial Z_j^{l+1}} \times \frac{\partial Z_j^{l+1}}{\partial b_i^{l}}\right) \\
&= \frac{1}{m}\sum_{t=1}^{m}\left(\delta_j^{l+1} \times \frac{\partial Z_j^{l+1}}{\partial b_i^{l}}\right) \\
&= \frac{1}{m}\sum_{t=1}^{m}\left(\delta_j^{l+1} \times \frac{\partial\left(\sum_{k=1}^{S_l}\left(W_{jk}^{l}a_k^{l}\right) + b_i^{l}\right)}{\partial b_i^{l}}\right) \\
&= \frac{1}{m}\sum_{t=1}^{m}\left(\delta_j^{l+1}\right)
\end{aligned}
$$

Backward pass



Contents

Background of the project

In this project, you'll build your first neural network and use it to predict daily bike rental ridership.

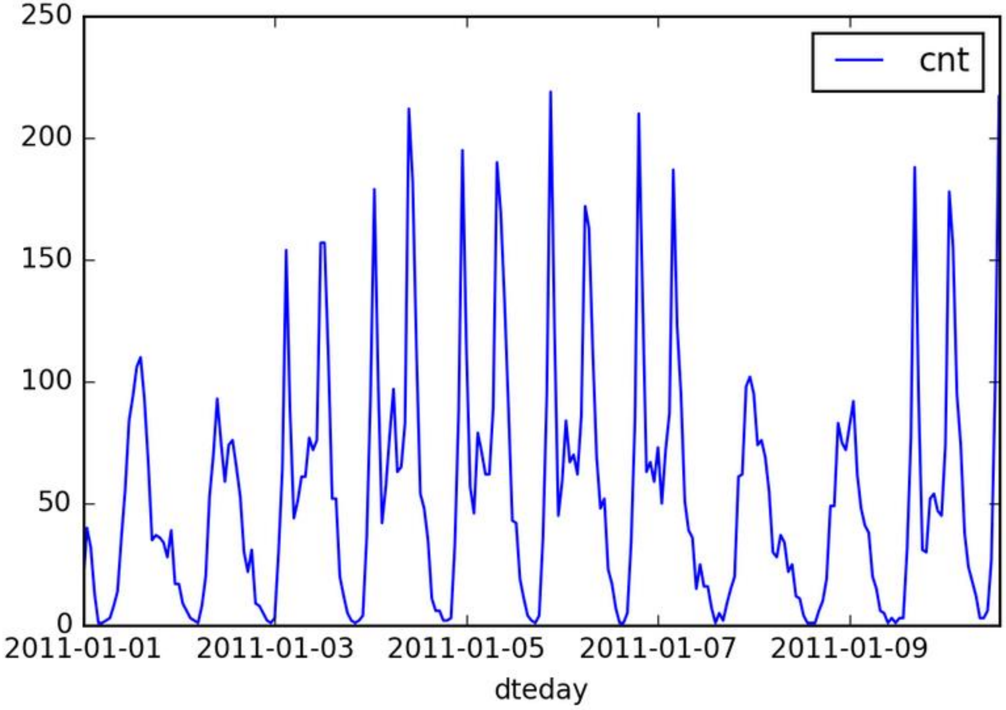


Fig. 4 Subset of the data

Load and prepare the data

In [2]:
```
data_path = 'Bike-Sharing-Dataset/hour.csv'

rides = pd.read_csv(data_path)
```

In [3]:
```
rides.head()
```

Out[3]:

	instant	dteday	season	yr	mnth	hr	holiday	weekday
0	1	2011-01-01	1	0	1	0	0	6
1	2	2011-01-01	1	0	1	1	0	6
2	3	2011-01-01	1	0	1	2	0	6
3	4	2011-01-01	1	0	1	3	0	6
4	5	2011-01-01	1	0	1	4	0	6

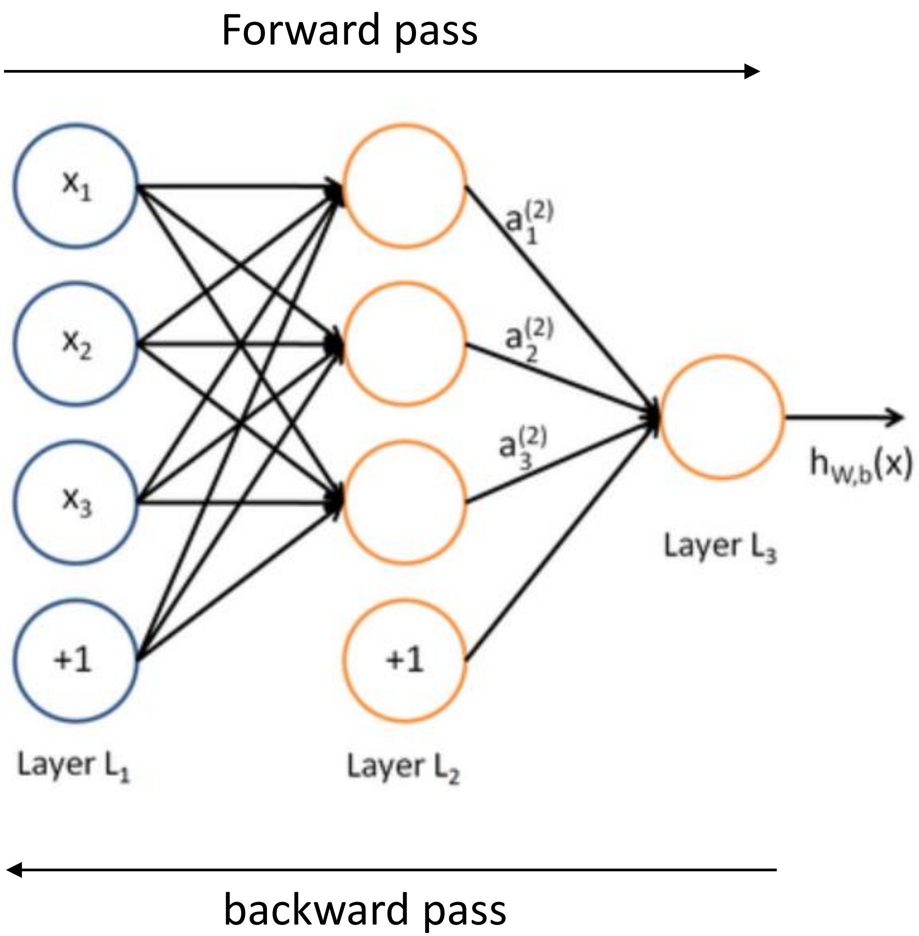
Fig. 8 show the dataset

Build the Network

TODO list:

- 将所有权重进行随机初始化
- 把每一层权重更新的初始梯度设置为 0
  - 输入到隐藏层的权重更新是 $\Delta w_{ij} = 0$
  - 隐藏层到输出层的权重更新是 $\Delta W_j = 0$
- 对训练数据当中的每一个点
  - 让它正向通过网络，计算输出 $\hat{y}$
  - 计算输出节点的误差梯度 $\delta^o = (y - \hat{y}) f'(z)$ 这里 $z = \sum_j W_j a_j$ 是输出节点的输入。
  - 误差传播到隐藏层 $\delta_j^h = \delta^o W_j f'(h_j)$
  - 更新权重步长:
    - $\Delta W_j = \Delta W_j + \delta^o a_j$
    - $\Delta w_{ij} = \Delta w_{ij} + \delta_j^h a_i$
- 更新权重, 其中 η 是学习率，m 是数据点的数量:
  - $W_j = W_j + \eta \Delta W_j / m$
  - $w_{ij} = w_{ij} + \eta \Delta w_{ij} / m$
- 重复这个过程 e 代。

Training and Validating



中山大學
SUN YAT-SEN UNIVERSITY

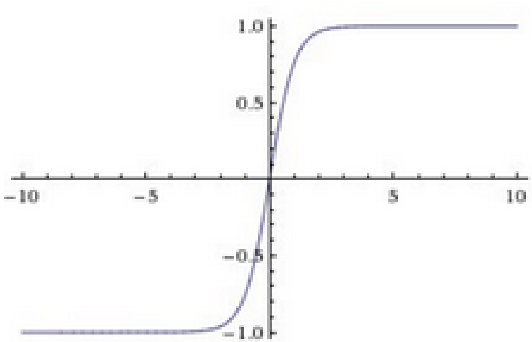
Contents

- L2正则化

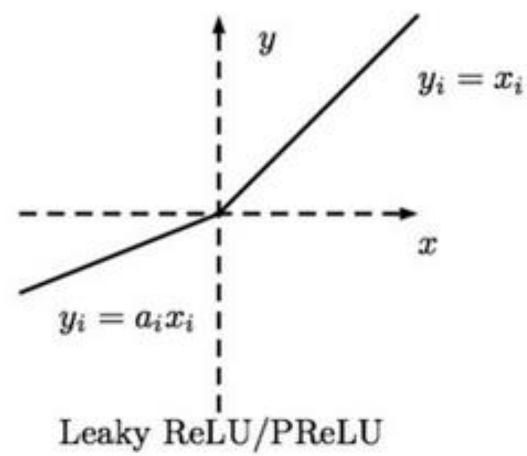
$$J(W,b) = \frac{1}{m}\sum_{i=1}^{m} J(W,b;x^i,y^i) + \frac{\lambda}{2}\sum_{l=1}^{nl-1}\sum_{i=1}^{S_l}\sum_{j=1}^{S_l+1}\left(W_{ji}^{(l)}\right)^2$$

- 不同的激活函数

Tanh



Relu



- 数据预处理
- 多层深度神经网络
- 不同神经网络架构
- Mini-batch

任务布置

- 共享单车使用量预测任务
- 标签内容参考readme，包含时间、天气等信息
- 必须实现三层神经网络（输入层，隐藏层，输出层）
- 必须在给出的优化建议中任意选择一项实现
- 自己划分验证集（报告里说明是怎么分的）调整参数

思考题

- 尝试说明下其他激活函数的优缺点。
- 有什么方法可以实现传递过程中不激活所有节点？
- 梯度消失和梯度爆炸是什么？可以怎么解决？

实验要求

- 提交文件
  - 实验报告: 18******_wangxiaoming.pdf。
  - 代码文件夹: 18******_wangxiaoming。如果代码分成多个文件，最好写份readme

- DDL:
  - 报告：10月22号 23:59:59
  - 验收：10月23号 18:00:00

THANKS