Basic RL.5

Judy Tutorial

Before we dive into **DQN**

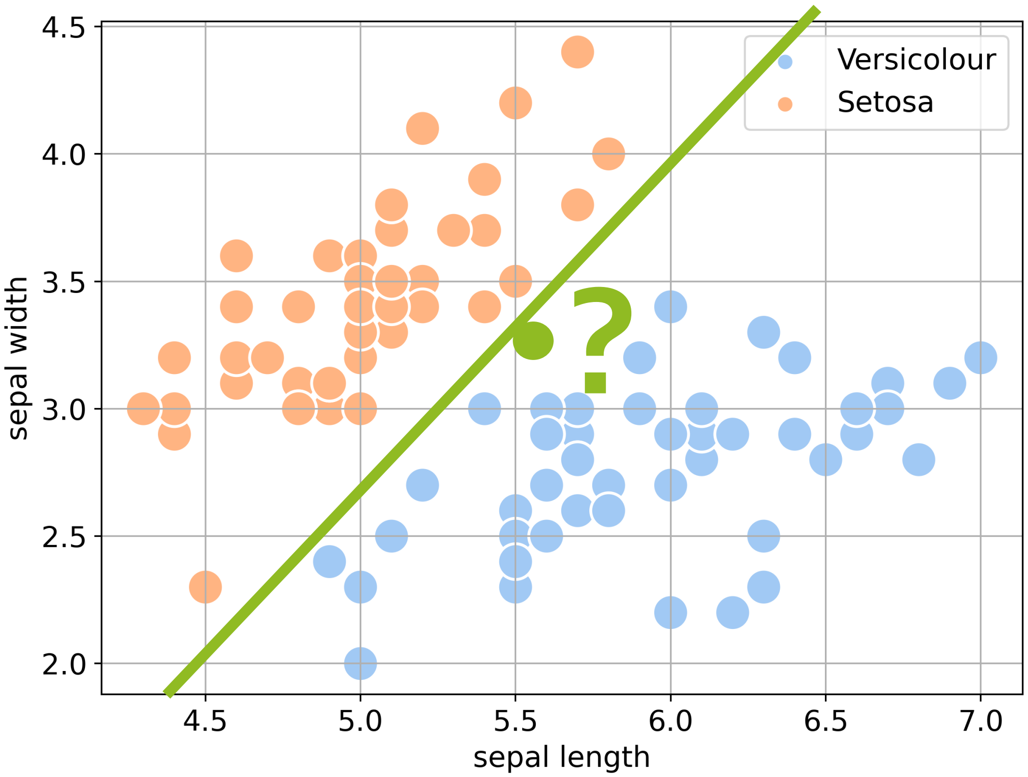
Logistic Regression & **Neural Networks**

MNIST dataset

[Lecun 1998]

Classification Problem

[Iris dataset 1988]



- Find a discriminant line of the labeled dots
- Classify a new dot

Binary Labels

$y \in \{0,1\}$

$$
\overset{\textstyle X^T}{\begin{array}{cc} \text{Sepal length} & \text{Sepal width} \end{array}}
$$

$$
\begin{bmatrix}
6.7 , & 3.1 \\
5.9 , & 3.0 \\
6.1 , & 2.8 \\
6.8 , & 2.8 \\
4.9 , & 3.6 \\
5.3 , & 3.7 \\
6.1 , & 2.8 \\
5.0 , & 2.3 \\
6.2 , & 2.2 \\
6.0 , & 3.4 \\
\vdots &
\end{bmatrix}
$$

100X2

$$
\overset{\textstyle Y^T}{\text{Iris type}}
$$

$$
\begin{bmatrix}
1.0 \\
1.0 \\
1.0 \\
1.0 \\
0.0 \\
0.0 \\
1.0 \\
1.0 \\
1.0 \\
1.0 \\
\vdots
\end{bmatrix}
$$

100X1

Hypothesis function

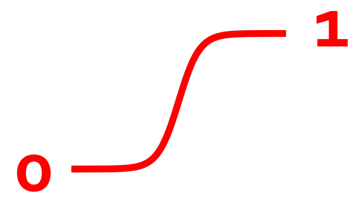
inputs

$$h(\boldsymbol{\theta}) = g(\boldsymbol{\theta}^{\mathrm{T}} X)$$

parameters

Activation function

$$g(z) = \frac{1}{1 + e^{-z}}$$



Objective function

$$J(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N} y^{(i)} \log h(\boldsymbol{\theta})^{(i)} + \left(1 - y^{(i)}\right) \log(1 - h(\boldsymbol{\theta})^{(i)})$$

Cross entropy loss

Objective function

?

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N} y^{(i)} \log h(\boldsymbol{\theta})^{(i)} + \left(1 - y^{(i)}\right) \log(1 - h(\boldsymbol{\theta})^{(i)})$$

Gradient

$$\boxed{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})}$$

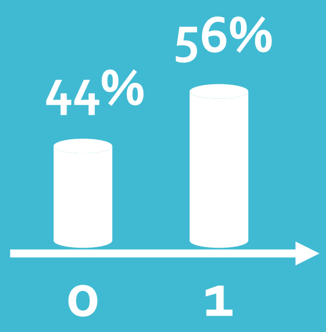
Gradient Descent

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

Negative **Log**-**Likelihood** of

Bernoulli distribution

Bernoulli Distribution



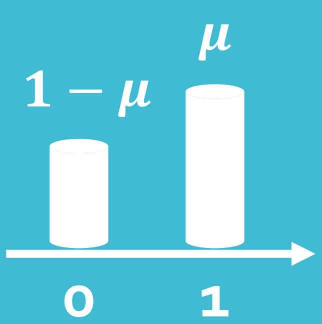
Wiki:

- a model for the set of possible outcomes of any single experiment that asks a yes-no question

Examples:

- What's the probability of getting a 'head' when tossing a coin?
- Which probability is higher? Ppl who like Coca-cola or Pepsi?
- A success or a failure? Cancer or not? Boy or girl? Quit a job or not? Label 0 or label 1? etc

Bernoulli Distribution



- Given a <u>random variable</u>: $x \in \{0,1\}$

$$Ber(x|\mu) = \mu^{x}(1-\mu)^{1-x} \quad 0 \le \mu \le 1$$

distribution parameter

- $Ber(x=1|\mu) = \mu^{1}(1-\mu)^{1-1} = \mu$
- $Ber(x=0|\mu) = \mu^{0}(1-\mu)^{1} = 1-\mu$

$$Ber(x|\mu) = \mu^x (1-\mu)^{1-x}$$

$$p(y|h(\boldsymbol{\theta})) = h(\boldsymbol{\theta})^y \left(1 - h(\boldsymbol{\theta})\right)^{1-y}$$

classification label

$g(\boldsymbol{\theta}^T \boldsymbol{X})$

$0 \leq h(\boldsymbol{\theta}) \leq 1$

Likelihood

Wiki:

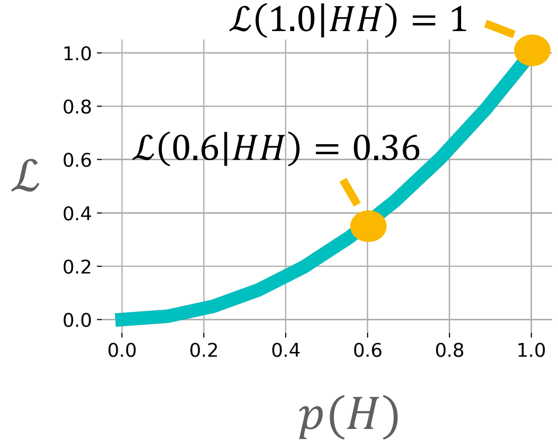
- Describes the joint probability of the observed data as a funcion of the parameters of the chosen statistical model

Observed data

$$\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{X}) = p(\boldsymbol{X}|\boldsymbol{\theta}) = p(x_1, x_2, \ldots|\boldsymbol{\theta})$$

Distribution parameters

- $P(HH|p(H) = 0.3) = 0.3 * 0.3 = 0.09$
- $P(HH|p(H) = 0.6) = 0.6 * 0.6 = 0.36$
- $\mathcal{L}(p(H) = 0.3|HH) = 0.09$
- $\mathcal{L}(p(H) = 0.6|HH) = 0.36$

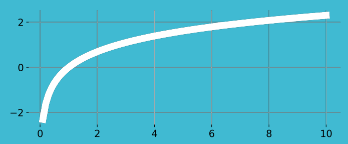


Maximum Likelihood Estimation

$$\hat{\boldsymbol{\theta}} = \operatorname{argmax}_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \mathcal{L}(\boldsymbol{\theta} | \boldsymbol{X})$$

MLE for Logistic Regression

Log



$$\mathcal{L}(\boldsymbol{\theta}) = \prod_{i=1}^{N} p\left(y^{(i)} \middle| h(\boldsymbol{\theta})^{(i)}\right)$$

$$= \prod_{i=1}^{N} {h(\boldsymbol{\theta})^{(i)}}^{y^{(i)}} \left(1 - h(\boldsymbol{\theta})^{(i)}\right)^{1-y^{(i)}}$$

Bernoulli Likelihood

$$\max_{\boldsymbol{\theta}} \log \mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \left[y^{(i)} \log h(\boldsymbol{\theta})^{(i)} + \left(1 - y^{(i)}\right) \log(1 - h(\boldsymbol{\theta})^{(i)})\right]$$

Objective function

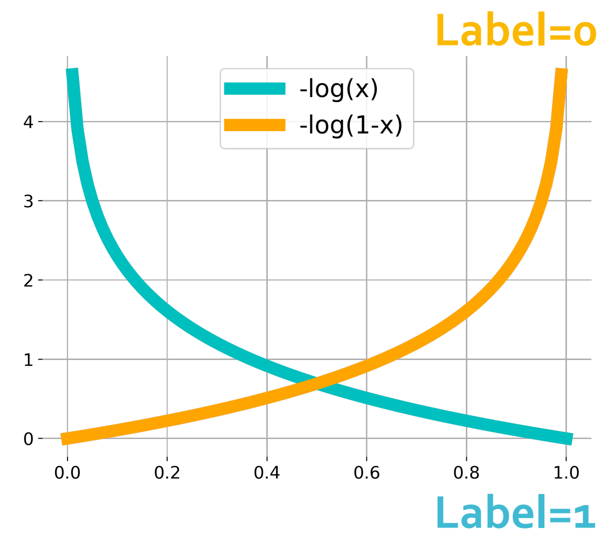
$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = -\frac{1}{N} \sum_{i=1}^{N} \left[y^{(i)} \log h(\boldsymbol{\theta})^{(i)} + \left(1 - y^{(i)}\right) \log(1 - h(\boldsymbol{\theta})^{(i)})\right]$$

Cross entropy loss

Cross Entropy Loss

$$\text{Log Loss} = -\frac{1}{N}\sum_{i=1}^{N} \log p^{(i)}$$

id	actual	predicted	corrected	log	-log
0	1	0.94	0.94	-0.02687	0.02687
1	1	0.9	0.9	-0.04576	0.04576
2	1	0.78	0.78	-0.10791	0.10791
3	0	0.56	0.44	-0.35655	0.35655
4	0	0.51	0.49	-0.3098	0.3098
5	1	0.47	0.47	-0.3279	0.3279
6	1	0.32	0.32	-0.49485	0.49485
7	0	0.1	0.9	-0.04576	0.04576



$$\text{Cross Entropy Loss} = -\frac{1}{N}\sum_{i=1}^{N}\left[y^{(i)}\log p^{(i)} + \left(1-y^{(i)}\right)\log\left(1-p^{(i)}\right)\right]$$

Objective function

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N} y^{(i)} \log h(\boldsymbol{\theta})^{(i)} + \left(1 - y^{(i)}\right) \log(1 - h(\boldsymbol{\theta})^{(i)})$$

Gradient

$$\nabla_{\theta_j} J(\theta_j) = \nabla_{\theta_j} -\frac{1}{N}\sum_{i=1}^{N} y^{(i)} \log h(\theta_j)^{(i)} + \left(1 - y^{(i)}\right) \log(1 - h(\theta_j)^{(i)})$$

$$= \frac{1}{N}\sum_{i=1}^{N} -\left[\frac{y^{(i)}}{h(\theta_j)^{(i)}} - \frac{1 - y^{(i)}}{1 - h(\theta_j)^{(i)}}\right] \nabla_{\theta_j} h(\theta_j)^{(i)}$$

$$= \frac{1}{N}\sum_{i=1}^{N} \left[h(\theta_j)^{(i)} - y^{(i)}\right] x^{(i)}$$

Element-wise

$$= \frac{1}{N}\boldsymbol{X}[\boldsymbol{h}(\boldsymbol{\theta}) - \boldsymbol{Y}]^T$$

Vector-wise

same as linear regression gradient ?!

Gradient Descent

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

Batch

Mini Batch

&

Stochastic Gradient Descent

Require a batch/entire dataset

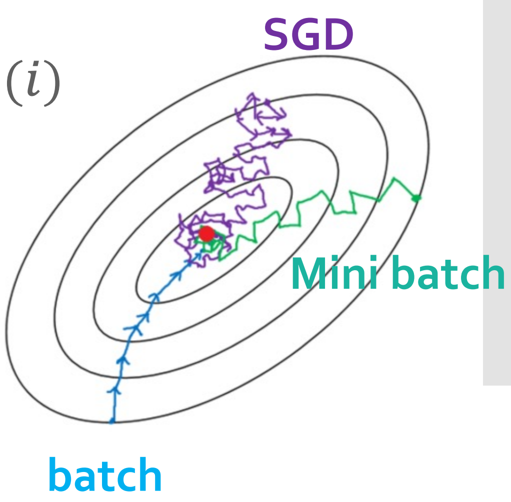
$$\nabla_{\theta_j} J(\theta_j) = \frac{1}{N} \sum_{i=1}^{N} \left[h(\theta_j)^{(i)} - y^{(i)} \right] x^{(i)}$$

Sample random mini batch from the entire dataset

$$\nabla_{\theta_j} J(\theta_j) = \frac{1}{N_{mini}} \sum_{i=1}^{N_{mini}} \left[h(\theta_j)^{(i)} - y^{(i)} \right] x^{(i)}$$

Require only one data point

$$\nabla_{\theta_j} J(\theta_j) = \left[h(\theta_j)^{(i)} - y^{(i)} \right] x^{(i)}$$



in python

```python
def sigmoid(x):

    return 1/(1+np.exp(-x))

def crossentropy_loss(X,y,theta):

    h=sigmoid(X@theta)
    return ((1/len(y))*(((-y).T@np.log(h+1e-5))-((1-y).T@np.log(1-h+1e-5))))[0][0]

def predict(X,theta):

    return np.round(sigmoid(X@theta))
```

avoid underflow

round up probabilities

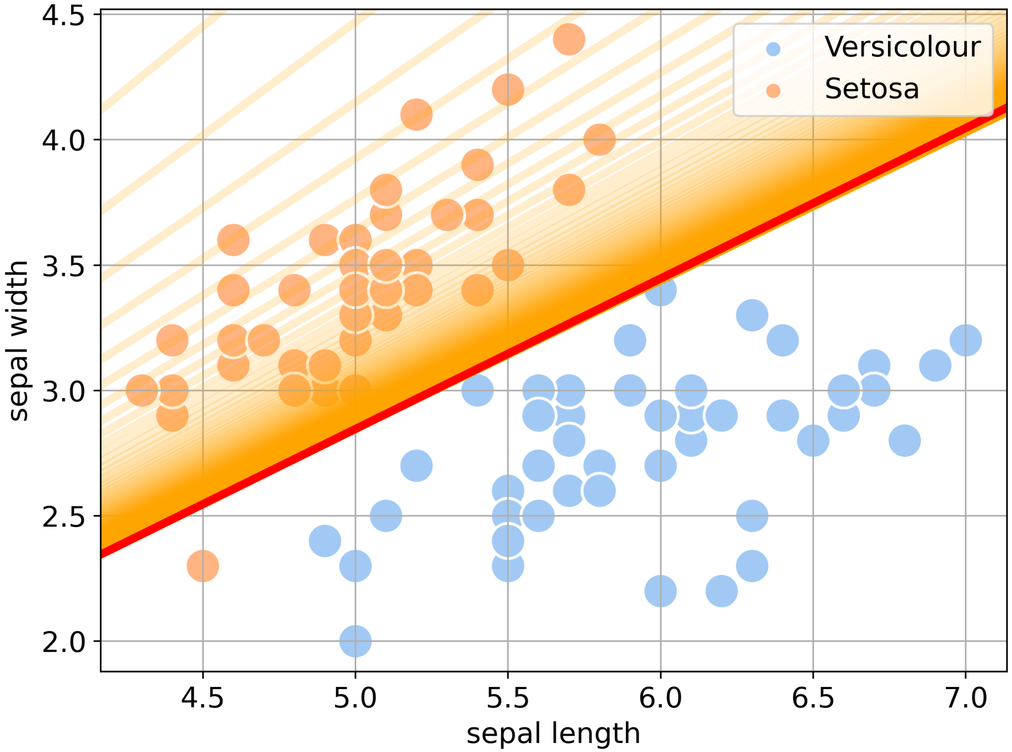
```python
X,y=Xy[:,:2],Xy[:,2]
X=np.hstack((np.ones((len(X),1)),X))
y=y[:,np.newaxis]
theta=np.zeros((X.shape[1],1))

N=3000 #number of iterations
lr=0.05
n=len(y) #number of data points

for i in range(N):

    h=sigmoid(X@theta)
    grad=X.T@(h-y)
    theta=theta-(lr/n)*grad
```

X : (100,3)
y : (100,1)
theta : (3,1)

h=X@theta :
(100,1)<-(100,3)(3,1)

grad=X.T@(h-y) :
(3,1)<-(3,100)(100,1)

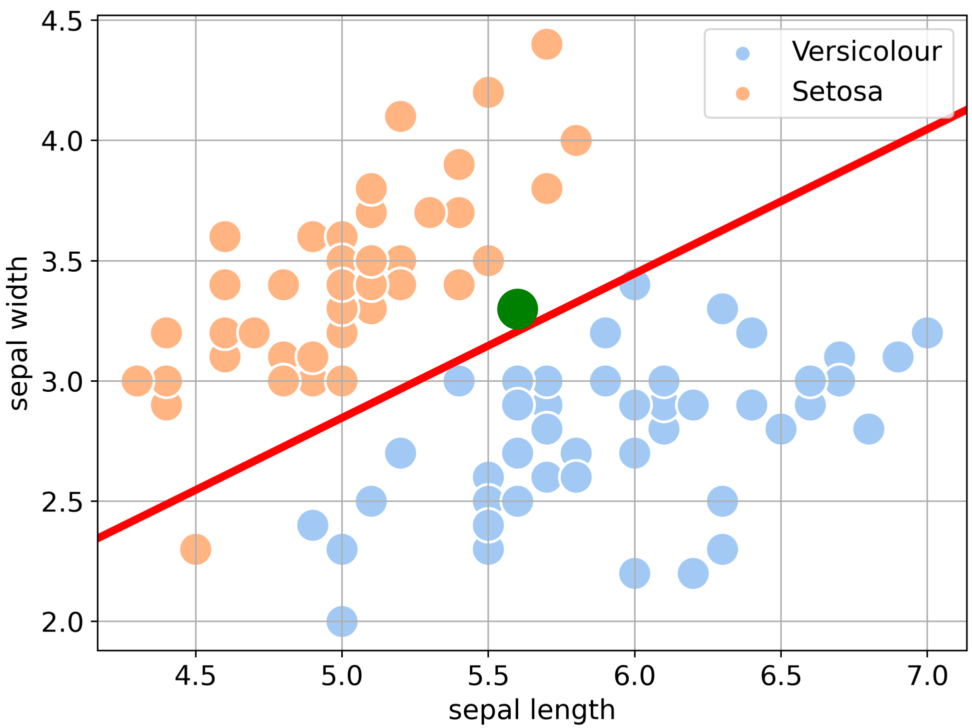
Learning behavior



- $h(\boldsymbol{\theta}) = g\left(\boldsymbol{\theta}^{\mathrm{T}}X\right) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$
- $h(\boldsymbol{\theta}) = \begin{cases} 1 & \text{if } \theta_0 + \theta_1 x_1 + \theta_2 x_2 > 0 \\ 0 & \text{if } \theta_0 + \theta_1 x_1 + \theta_2 x_2 < 0 \end{cases}$
- $\theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0 \Rightarrow x_2 = -\frac{\theta_1}{\theta_2} x_1 - \frac{\theta_0}{\theta_2}$

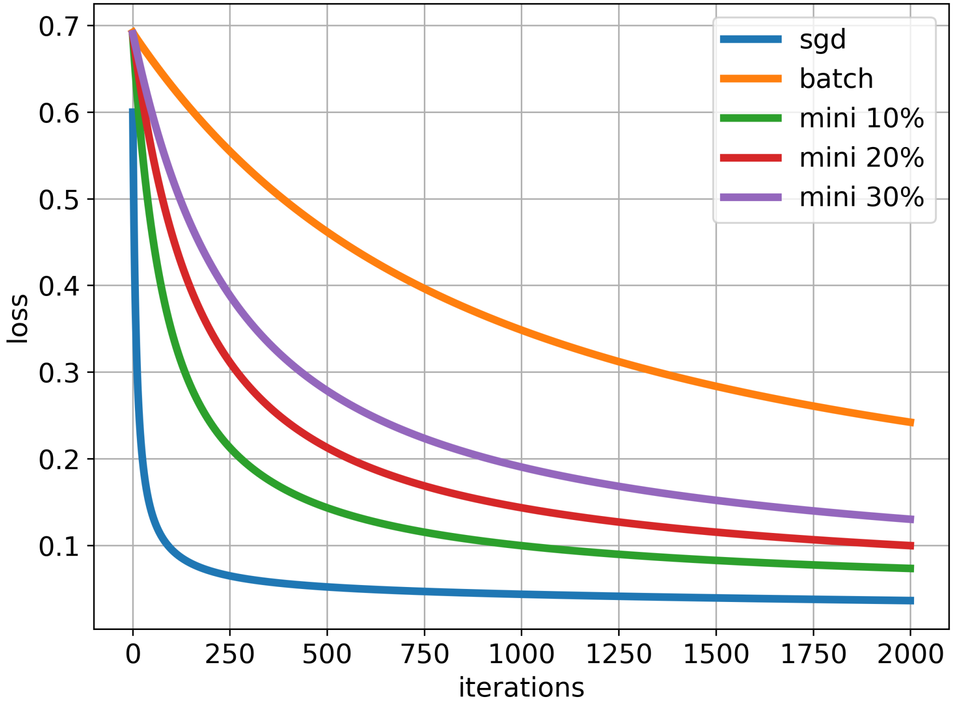
Decision boundary

prediction



- Given a new data: $x_1 = 5.6, x_2 = 3.3$
- $\theta_0 = -0.8966967, \theta_1 = 3.51427467, \theta_2 = -5.85710308$
- $h(\boldsymbol{\theta}) = g\left(\boldsymbol{\theta}^{\mathrm{T}} X\right) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2) = 0$ **setosa**

Behaviors of different gradient descent methods



- Mini batch_size=1 <=> stochastic gradient
- Mini batch_size=N <=> batch gradient
- SGD converges the fastest but may oscillating around the optimal minimum