

# Basic RL.5

Judy Tutorial

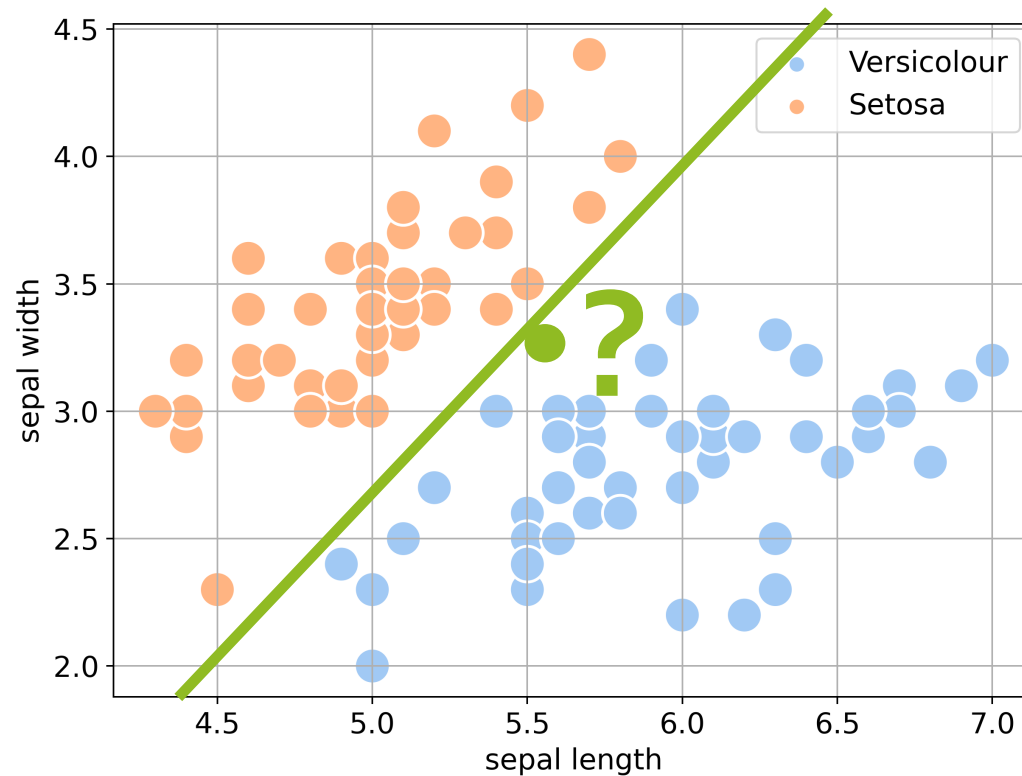
Before we dive into **DQN**

**Logistic Regression & Neural Networks**



# Classification Problem

[Iris dataset 1988]



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

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

$X^T$

Sepal length   Sepal width

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
⋮	

100X2

$Y^T$

Iris type

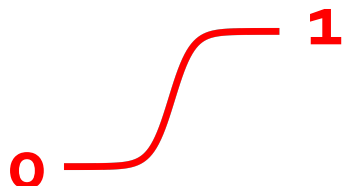
1.0
1.0
1.0
1.0
0.0
0.0
1.0
1.0
1.0
1.0
⋮

100X1

## Hypothesis function

$$h(\theta) = g(\underbrace{\theta^T}_{\text{parameters}} \underbrace{X}_{\text{inputs}})$$

## Activation function


$$g(z) = \frac{1}{1 + e^{-z}}$$
A red sigmoid curve starting at 0 on the left and approaching 1 on the right. The number 0 is at the start and 1 is at the end of the curve.

## Objective function

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

Cross entropy loss

## Objective function

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




## Gradient

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



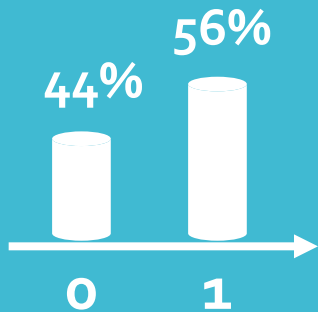
## Gradient Descent

$$\theta_{t+1} \leftarrow \theta_t - \alpha \nabla_{\theta} J(\theta)$$

# Negative **Log-Likelihood** of **Bernoulli distribution**



# Bernoulli Distribution



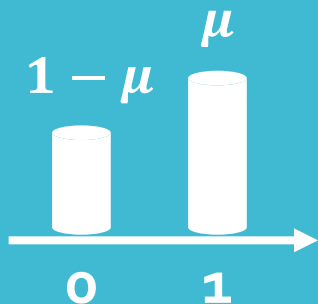
Wiki:

- 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 random variable:  $x \in \{0,1\}$

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

└ distribution parameter

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

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

|

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

/

classification  
label

|

$$g(\theta^T X)$$

$$0 \leq h(\theta) \leq 1$$

# Likelihood

Wiki:

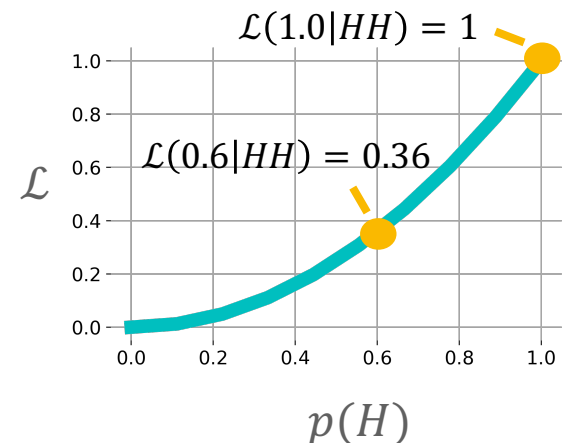
- Describes the joint probability of the observed data as a function of the parameters of the chosen statistical model

┐ Observed data

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

└ Distribution parameters

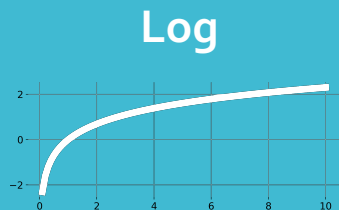
- $P(HH | p(H) = 0.3) = 0.3 * 0.3 = 0.09$
- $P(HH | p(H) = 0.6) = 0.6 * 0.6 = 0.36$
- $\mathcal{L}(p(H) = 0.3 | HH) = 0.09$
- $\mathcal{L}(p(H) = 0.6 | HH) = 0.36$



# Maximum Likelihood Estimation

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

# MLE for Logistic Regression



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

Bernoulli Likelihood

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

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

Objective function



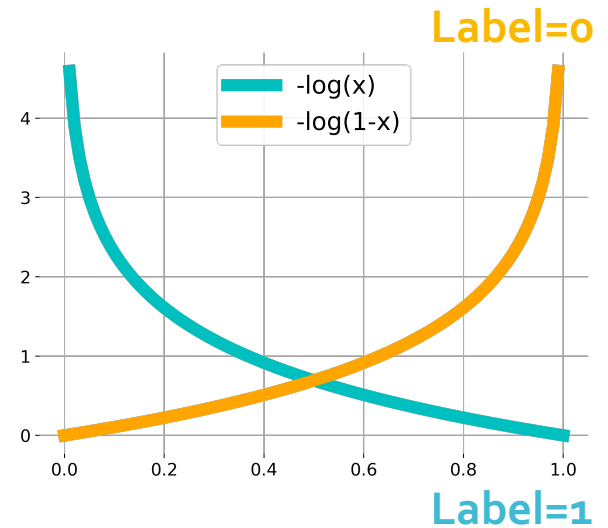
$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = -\frac{1}{N} \sum_{i=1}^N [y^{(i)} \log h(\boldsymbol{\theta})^{(i)} + (1 - y^{(i)}) \log(1 - h(\boldsymbol{\theta})^{(i)})]$$

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



Cross Entropy Loss =

$$-\frac{1}{N} \sum_{i=1}^N [y^{(i)} \log p^{(i)} + (1 - y^{(i)}) \log(1 - p^{(i)})]$$

## Objective function

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

## Gradient

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

Element-wise

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

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

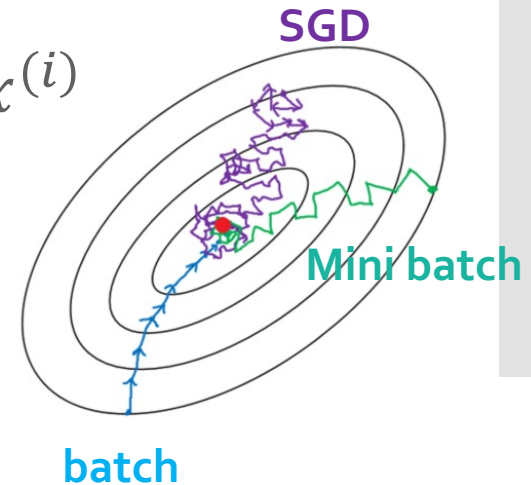
$$\nabla_{\theta_j} J(\theta_j) = \frac{1}{N} \sum_{i=1}^N \left[ h(\theta_j)^{(i)} - y^{(i)} \right] x^{(i)}$$

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

```
def sigmoid(x):  
    return 1/(1+np.exp(-x))  
  
def crossentropy_loss(X,y,theta):  
    h=sigmoid(X@theta)  
    return ((1/len(y))*(((1-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

```
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
```

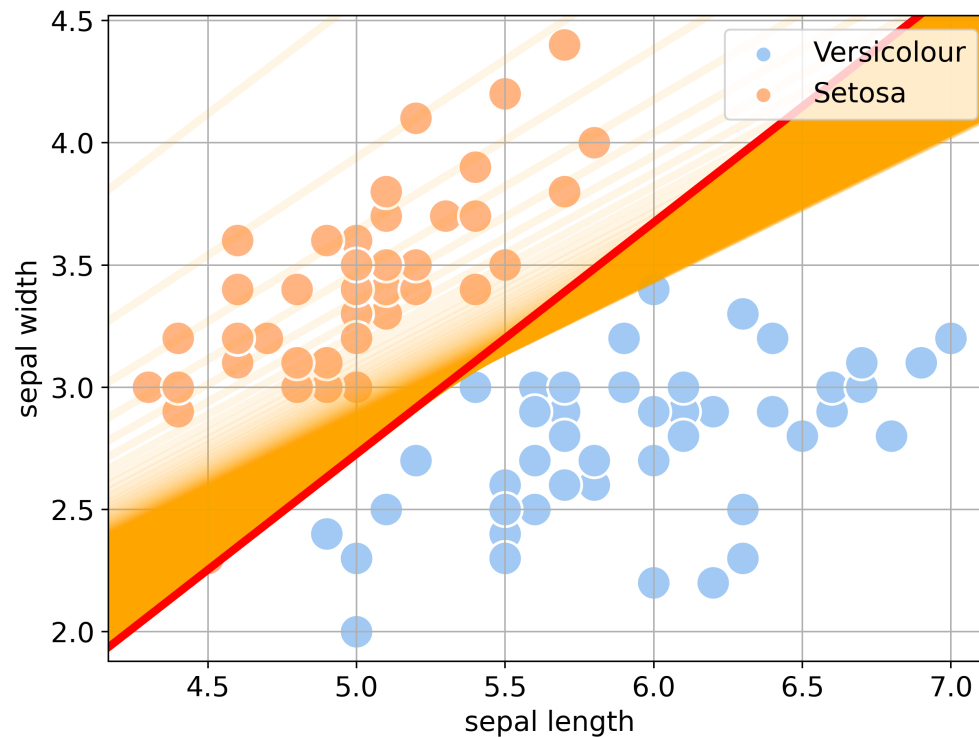
X : (100,3)  
y : (100,1)  
theta : (3,1)

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

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

# Learning behavior

[200,000 iterations with batch gradient]

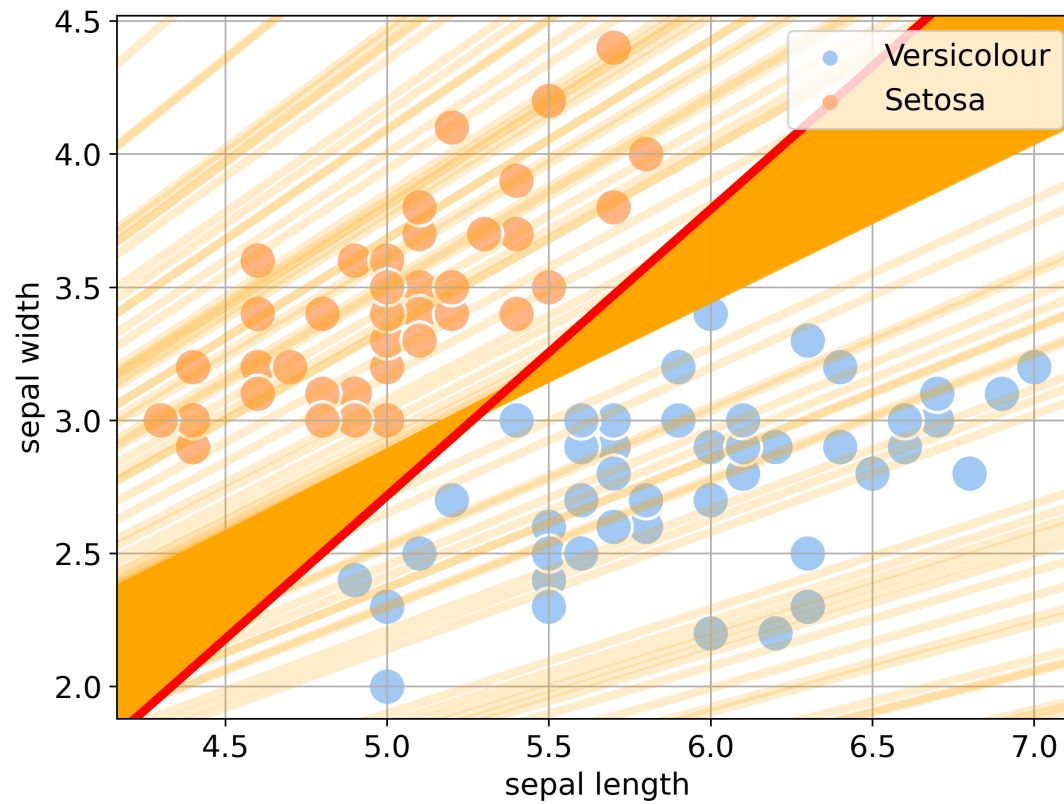


- $h(\theta) = g(\theta^T X) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$
- $h(\theta) = \begin{cases} 1 & \text{if } \theta_0 + \theta_1 x_1 + \theta_2 x_2 > 0 \\ 0 & \text{if } \theta_0 + \theta_1 x_1 + \theta_2 x_2 < 0 \end{cases}$
- $\theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0 \Rightarrow x_2 = -\frac{\theta_1}{\theta_2}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

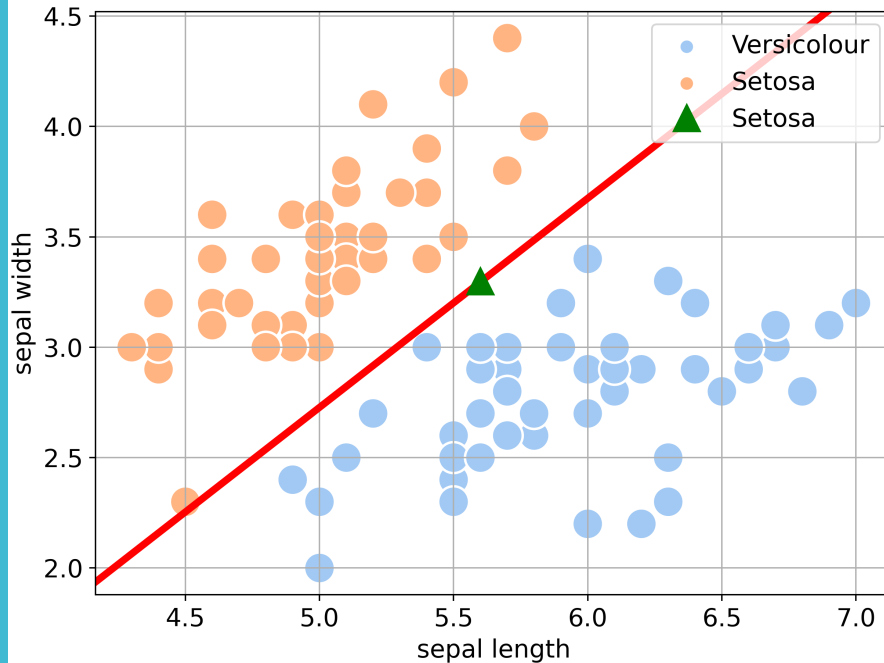
Decision boundary

# Learning behavior

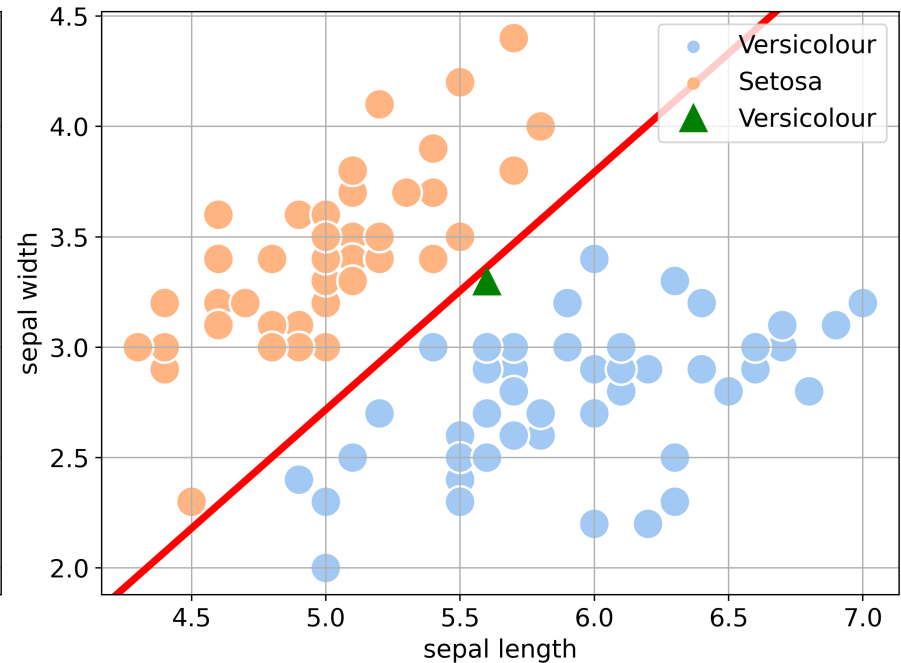
[10,000,iterations with SGD]



Batch 200,000 iterations



SGD 10,000 iterations



Given a new data:  $x_1 = 5.6, x_2 = 3.3$

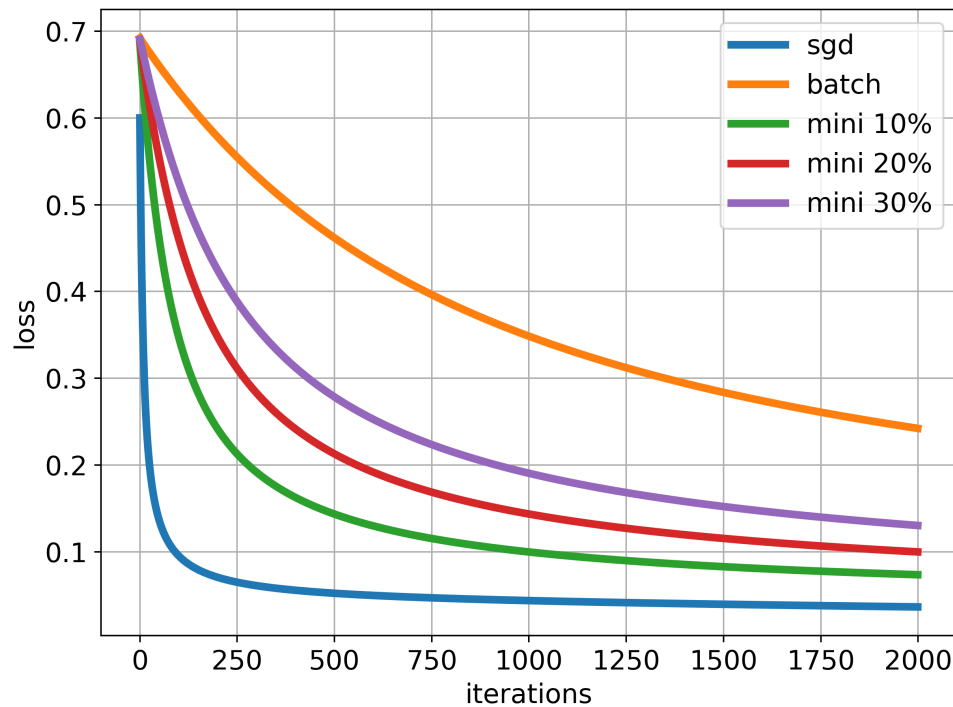
- $\theta_0 = -25.647, \theta_1 = 12.096, \theta_2 = -12.772$
- $h(\theta) = g(\theta^T X) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2) = 0$
- $\theta_0 = -48.67, \theta_1 = 19.684, \theta_2 = -18.305$
- $h(\theta) = 1$

setosa

versicolour

# Behaviors of different gradient descents

[2000 iterations]



- Mini batch\_size=1  $\Leftrightarrow$  stochastic gradient
- Mini batch\_size=N  $\Leftrightarrow$  batch gradient
- SGD converges the fastest but may oscillating around the optimal minimum