CMI LAB

UNIVERSITY OF WOLVERHAMPTON
KNOWLEDGE ▪ INNOVATION ▪ ENTERPRISE

6CS012 – Artificial Intelligence and Machine Learning

Ahsan Adeel

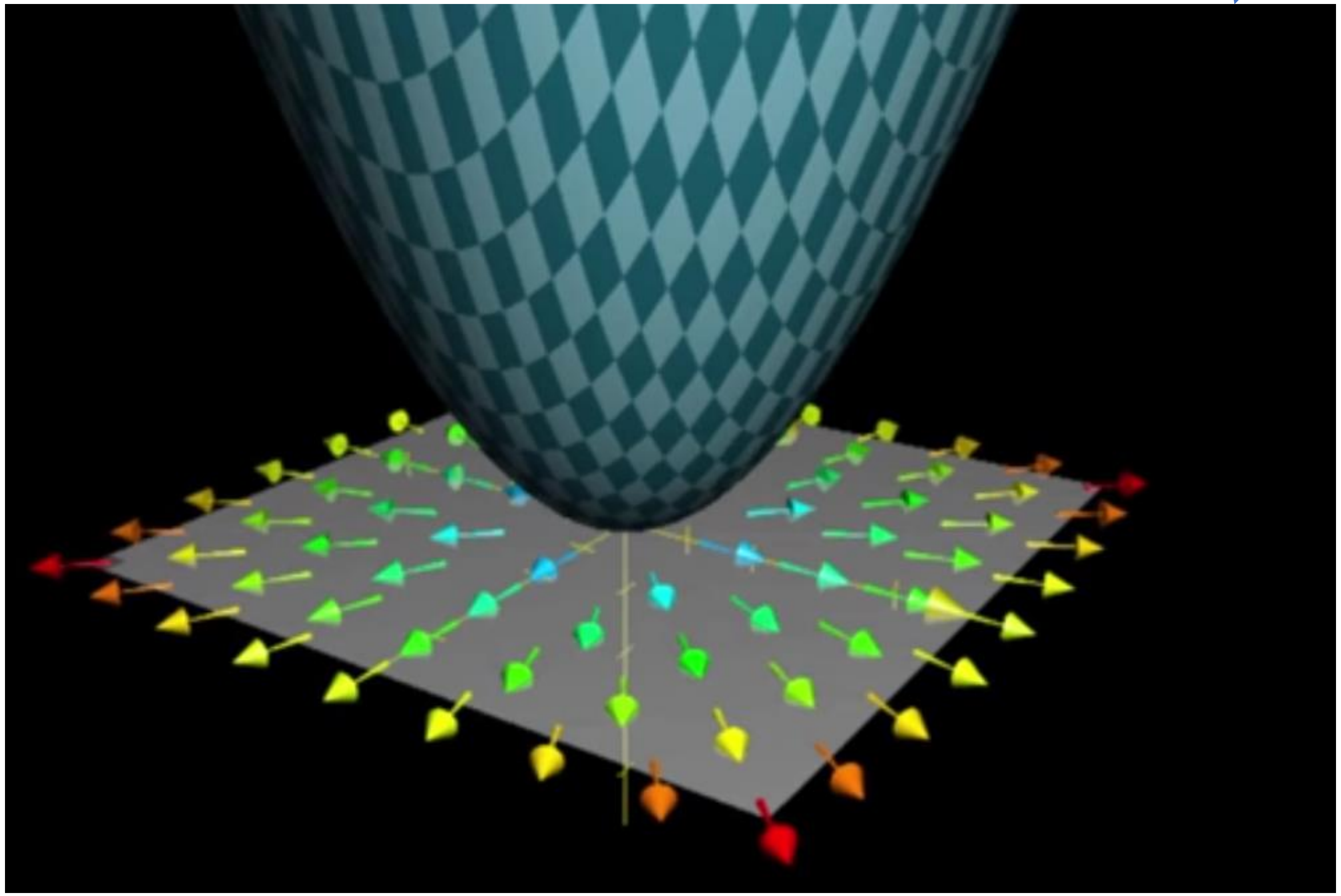
Theme lead, Conscious Multisensory Integration (CMI) Lab
Fellow, MIT Synthetic Intelligence Lab, MIT
and Oxford Computational Neuroscience Lab, University of Oxford
Visiting EPSRC/MRC fellow, University of Stirling
http://www.cmilab.org/
http://www.cmilab.org/dr_ahsan_adeel.html#Home

NUFFIELD DEPARTMENT OF SURGICAL SCIENCES
Medical Sciences Division

Lecture 4 – Logistic Regression, Softmax Regression, and Cross-Entropy

Lecture 3 Review

$$\boldsymbol{\theta}^{(\text{ next step })} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \ \text{MSE}(\boldsymbol{\theta})$$



Example 1

$f(x) = x^2$

- Starting point: x= -2
- $\eta = 0.1$
- ∇ = [d/dx (f(x))] = 2x

$$\theta^{(\text{next step})} = \theta - \eta \nabla_{\theta} \ \text{MSE}(\theta)$$

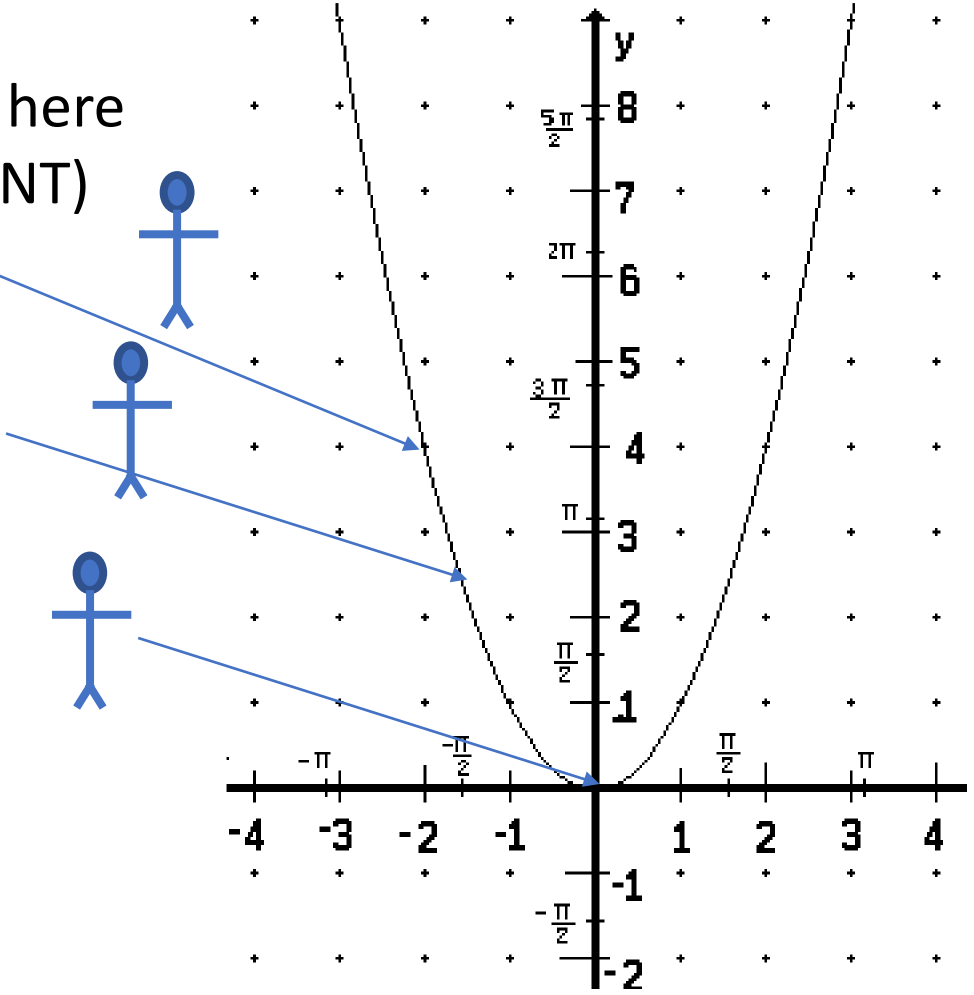
X (next step) = X - η ∇f(X)

Iteration ‘1’

- X (next step) = (-2) – (0.1)(2(-2))
- X (next step) = -2 + 0.4 = -1.6

Iteration ‘2’
Iteration ‘3’
Iteration ‘4’
Iteration ‘n’

Suppose we are here
(STRATING POINT)

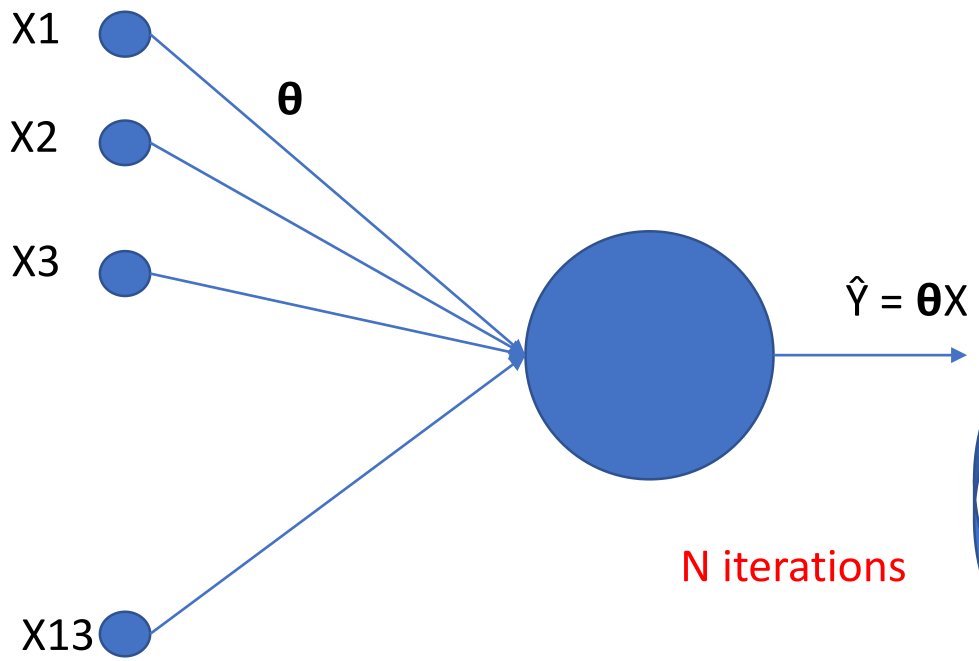


WS-3 Task 3

- Solve task 3 of WS-2 using GD

$\hat{Y} = h_{\boldsymbol{\theta}}(X) = X\boldsymbol{\theta}^T$

Optimal $\boldsymbol{\theta}$?



$$\boldsymbol{\theta}^{(\text{next step})} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \ \text{MSE}(\boldsymbol{\theta})$$

$$\text{MSE}(\mathbf{X}, h_{\boldsymbol{\theta}}) = \frac{1}{m}\sum_{i=1}^{m}\left(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)}\right)^2$$

$$\nabla_{\boldsymbol{\theta}} \ \text{MSE}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial}{\partial\theta_0}\text{MSE}(\boldsymbol{\theta}) \\ \frac{\partial}{\partial\theta_1}\text{MSE}(\boldsymbol{\theta}) \\ \vdots \\ \frac{\partial}{\partial\theta_n}\text{MSE}(\boldsymbol{\theta}) \end{pmatrix} = \frac{2}{m}\mathbf{X}^T(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

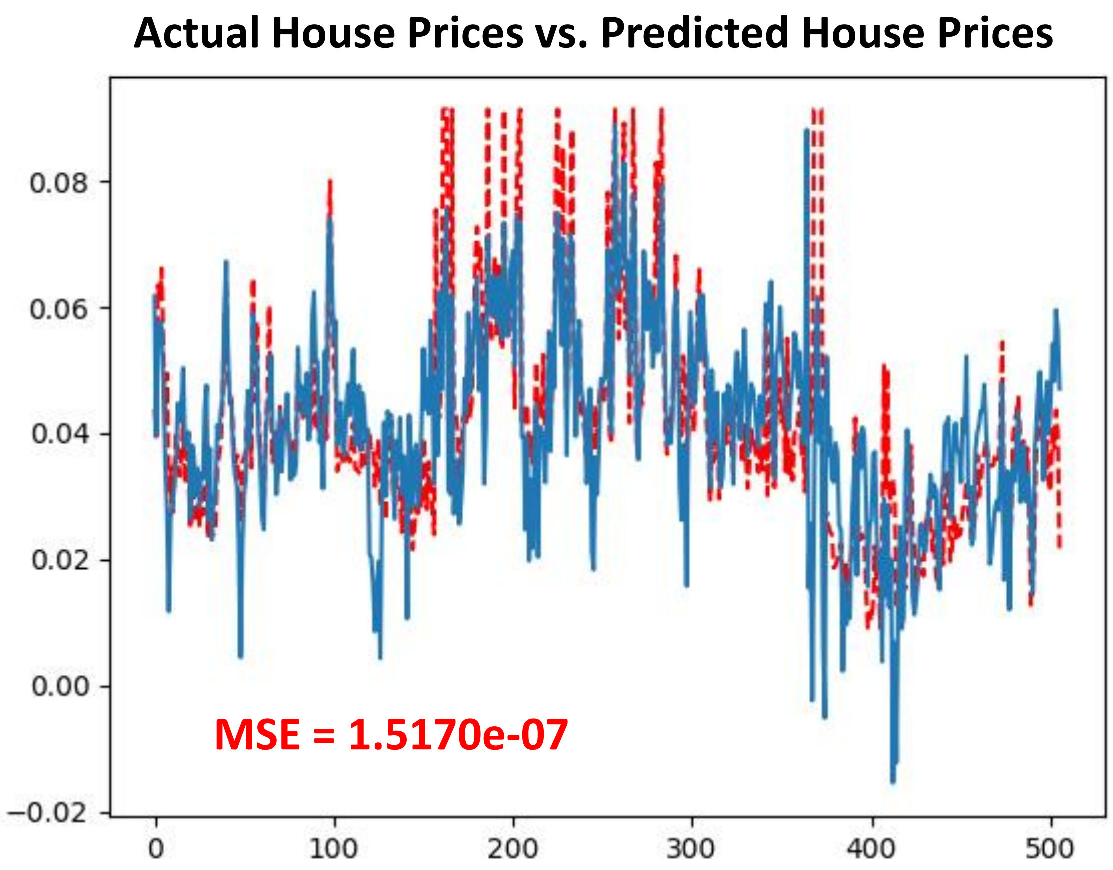
N iterations

Sub-optimal $\boldsymbol{\theta}$

WS-3 Task 3

n_iterations = 100000

<u>**$\hat{Y} = \theta X$ ---> Blue line**</u>



Lecture 4 – Logistic Regression, Softmax Regression, and Cross-Entropy

Logistic Regression

- Previously we learned how to predict continuous-valued quantities (e.g., housing prices) as a linear function of input values (e.g., the size of the house).

- Sometimes we will instead wish to predict a discrete variable such as predicting whether a grid of pixel intensities represents a “0” digit or a “1” digit.

- This is a classification problem. Logistic regression is a simple classification algorithm for learning to make such decisions.

Logistic Regression

- In linear regression we tried to predict the value of y(i) for the i'th example x(i) using a linear function: $\hat{Y} = h_{\boldsymbol{\theta}}(X) = X\boldsymbol{\theta}^T$
- This is clearly not a great solution for predicting binary-valued labels: $(y(i) \in \{0,1\})$.
- In logistic regression we use a different hypothesis class to try to predict the probability that a given example belongs to the "1" class versus the probability that it belongs to the "0" class.

Logistic Regression

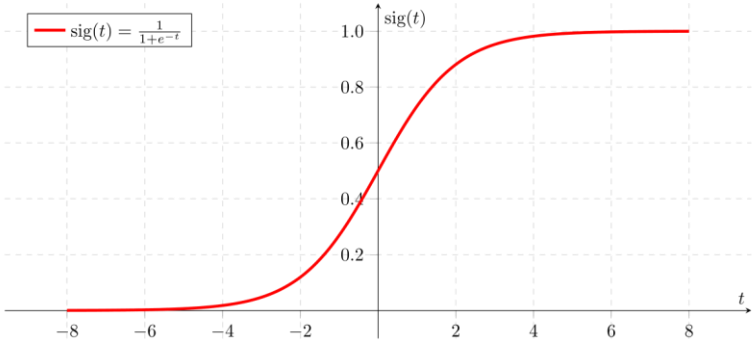
- Specifically, we will try to learn a function of the form:

$$P(y=1|x) = h_\theta(x) = \frac{1}{1+\exp(-\theta^\top x)} \equiv \sigma(\theta^\top x),$$
$$P(y=0|x) = 1 - P(y=1|x) = 1 - h_\theta(x).$$

- In simle terms: $\hat{Y} = h_{\boldsymbol{\theta}}(X) = \sigma(X\boldsymbol{\theta}^\top)$

- The function $\boldsymbol{\sigma(z) = 1/1+\exp(-z)}$ the “sigmoid” or “logistic” function

- it is an S-shaped function that “squashes” the value of ϑTx into the range [0,1] so that we may interpret $h\vartheta(x)$ as a probability.



Logistic Regression

- Our goal is to search for a value of ϑ so that the probability $P(y=1|x)=h\vartheta(x)$ is large when x belongs to the "1" class and small when x belongs to the "0" class (so that $P(y=0|x)$ is large).

- Cost function: $J(\theta) = -\sum_i \left(y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right)$

- Note that only one of the two terms in the summation is non-zero for each training example (depending on whether the label $y(i)$ is 0 or 1).

- We now have a cost function that measures how well a given hypothesis fits our training data.

- We can learn to classify our training data by minimizing $J(\vartheta)$ to find the best choice of ϑ.

Logistic Regression

- To minimize $J(\vartheta)$ we can use the same tools as for linear regression.

- We need to provide a function that computes $J(\vartheta)$ and $\nabla J(\vartheta)$ for any requested choice of ϑ. The derivative of $J(\vartheta)$ as given above with respect to ϑ is:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \sum_i x_j^{(i)} (h_\theta(x^{(i)}) - y^{(i)}).$$

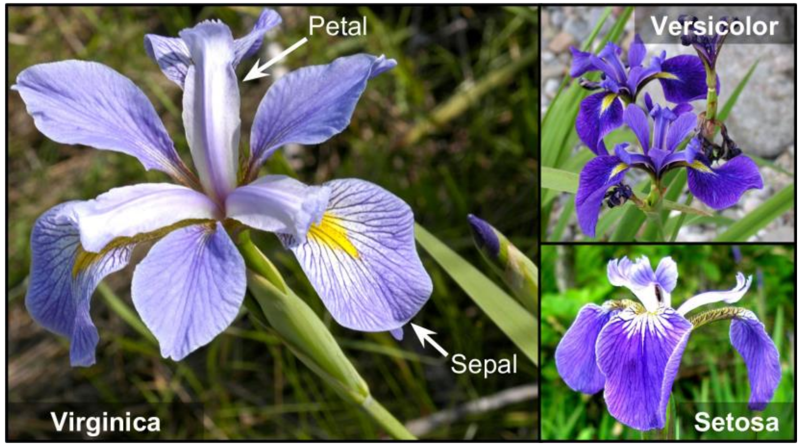
$$\nabla_\theta J(\theta) = \sum_i x^{(i)} (h_\theta(x^{(i)}) - y^{(i)})$$

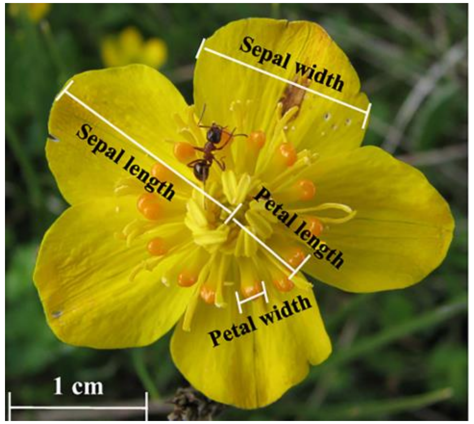
Logistic Regression

- Classification problem
- **Iris multivariate data set:** The dataset contains a set of 150 records under five attributes - petal length, petal width, sepal length, sepal width and species.

petal length	petal width	sepal length	sepal width	**flower type**

- iris flowers of three different species: Iris-Setosa, Iris-Versicolor, and Iris-Virginica





Logistic Regression

iris.data

	0	1	2	3
0	5.10000	3.50000	1.40000	0.20000
1	4.90000	3.00000	1.40000	0.20000
2	4.70000	3.20000	1.30000	0.20000
3	4.60000	3.10000	1.50000	0.20000
4	5.00000	3.60000	1.40000	0.20000
5	5.40000	3.90000	1.70000	0.40000
6	4.60000	3.40000	1.40000	0.30000
7	5.00000	3.40000	1.50000	0.20000
8	4.40000	2.90000	1.40000	0.20000
9	4.90000	3.10000	1.50000	0.10000
10	5.40000	3.70000	1.50000	0.20000
11	4.80000	3.40000	1.60000	0.20000
12	4.80000	3.00000	1.40000	0.10000
13	4.30000	3.00000	1.10000	0.10000
14	5.80000	4.00000	1.20000	0.20000

iris.target

	0	1	2	3	4	5
0	0	0	0	0	0	0

iris.data

	0	1	2	3
48	5.30000	3.70000	1.50000	0.20000
49	5.00000	3.30000	1.40000	0.20000
50	7.00000	3.20000	4.70000	1.40000
51	6.40000	3.20000	4.50000	1.50000
52	6.90000	3.10000	4.90000	1.50000
53	5.50000	2.30000	4.00000	1.30000
54	6.50000	2.80000	4.60000	1.50000
55	5.70000	2.80000	4.50000	1.30000
56	6.30000	3.30000	4.70000	1.60000
57	4.90000	2.40000	3.30000	1.00000
58	6.60000	2.90000	4.60000	1.30000
59	5.20000	2.70000	3.90000	1.40000
60	5.00000	2.00000	3.50000	1.00000
61	5.90000	3.00000	4.20000	1.50000
62	6.00000	2.20000	4.00000	1.00000

iris.target

	48	49	50	51	52	53
0	0	0	1	1	1	1

iris.data

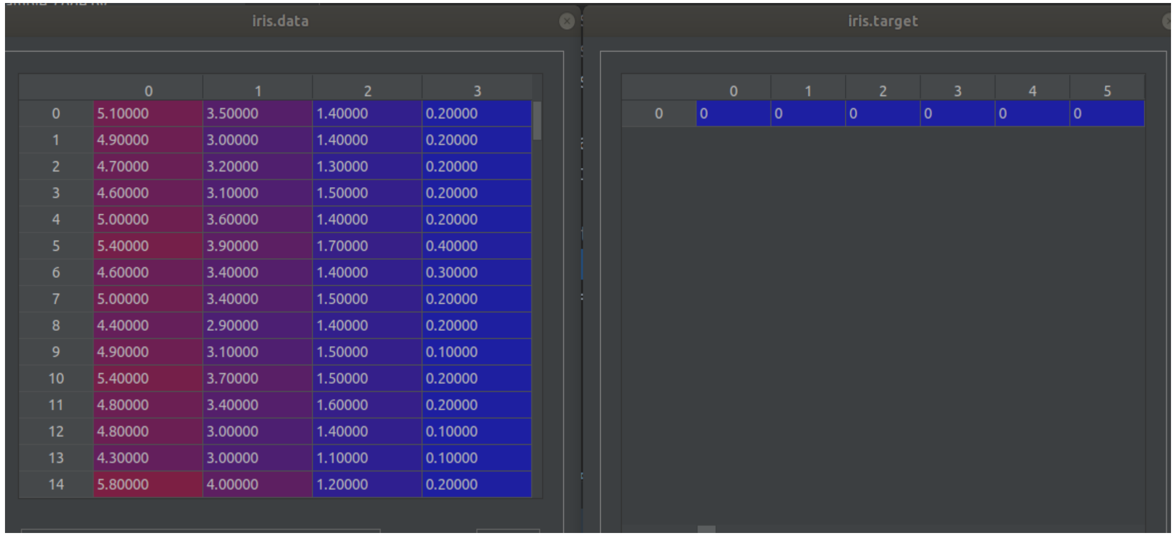
	0	1	2	3
92	5.80000	2.60000	4.00000	1.20000
93	5.00000	2.30000	3.30000	1.00000
94	5.60000	2.70000	4.20000	1.30000
95	5.70000	3.00000	4.20000	1.20000
96	5.70000	2.90000	4.20000	1.30000
97	6.20000	2.90000	4.30000	1.30000
98	5.10000	2.50000	3.00000	1.10000
99	5.70000	2.80000	4.10000	1.30000
100	6.30000	3.30000	6.00000	2.50000
101	5.80000	2.70000	5.10000	1.90000
102	7.10000	3.00000	5.90000	2.10000
103	6.30000	2.90000	5.60000	1.80000
104	6.50000	3.00000	5.80000	2.20000
105	7.60000	3.00000	6.60000	2.10000
106	4.90000	2.50000	4.50000	1.70000

iris.target

	98	99	100	101	102	1
0	1	1	2	2	2	2

Logistic Regression

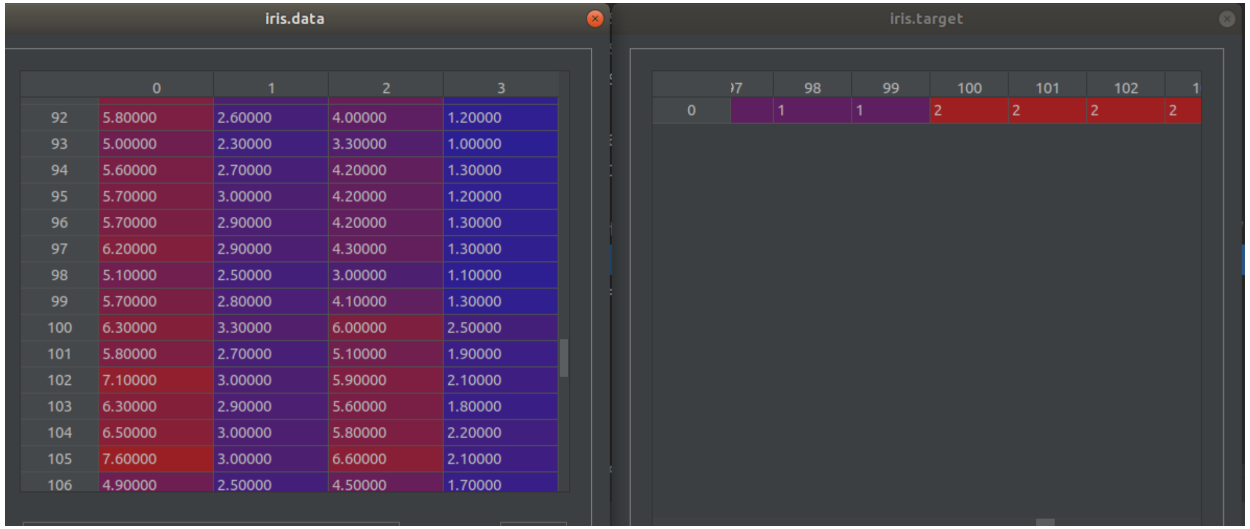
- In this problem, there are three categories, can Logistic Regression handle this? **NO**
- **Remember,** we can only predict binary-valued labels (0/1): (y(i) ∈ {0,1}).
- Let's transform this problem into a binary problem (**use first 100 samples**)


iris.data

	0	1	2	3
0	5.10000	3.50000	1.40000	0.20000
1	4.90000	3.00000	1.40000	0.20000
2	4.70000	3.20000	1.30000	0.20000
3	4.60000	3.10000	1.50000	0.20000
4	5.00000	3.60000	1.40000	0.20000
5	5.40000	3.90000	1.70000	0.40000
6	4.60000	3.40000	1.40000	0.30000
7	5.00000	3.40000	1.50000	0.20000
8	4.40000	2.90000	1.40000	0.20000
9	4.90000	3.10000	1.50000	0.10000
10	5.40000	3.70000	1.50000	0.20000
11	4.80000	3.40000	1.60000	0.20000
12	4.80000	3.00000	1.40000	0.10000
13	4.30000	3.00000	1.10000	0.10000
14	5.80000	4.00000	1.20000	0.20000

iris.target

	0	1	2	3	4	5
0	0	0	0	0	0	0


iris.data

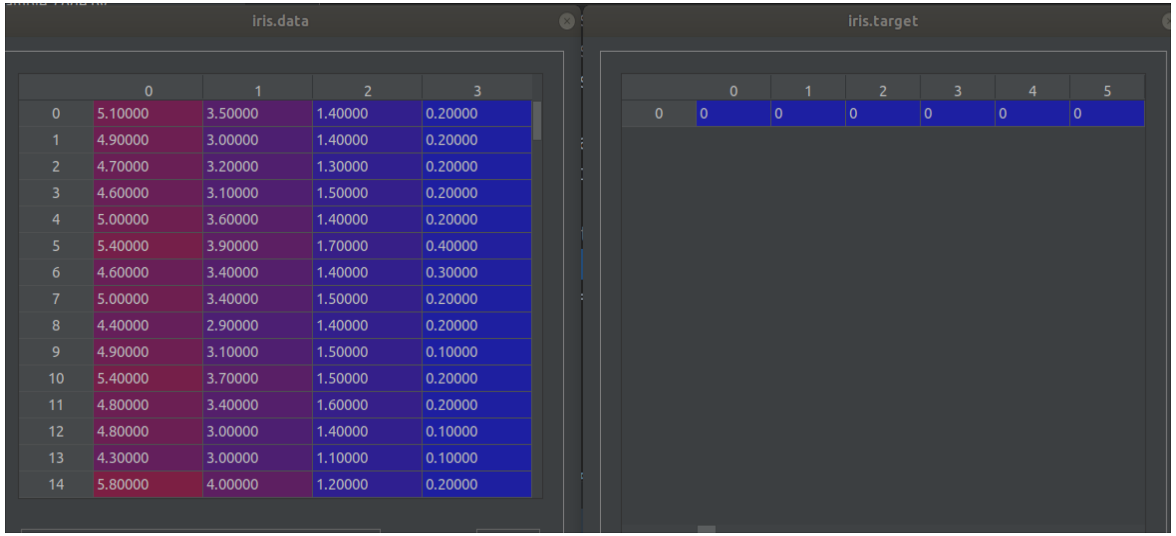
	0	1	2	3
92	5.80000	2.60000	4.00000	1.20000
93	5.00000	2.30000	3.30000	1.00000
94	5.60000	2.70000	4.20000	1.30000
95	5.70000	3.00000	4.20000	1.20000
96	5.70000	2.90000	4.20000	1.30000
97	6.20000	2.90000	4.30000	1.30000
98	5.10000	2.50000	3.00000	1.10000
99	5.70000	2.80000	4.10000	1.30000
100	6.30000	3.30000	6.00000	2.50000
101	5.80000	2.70000	5.10000	1.90000
102	7.10000	3.00000	5.90000	2.10000
103	6.30000	2.90000	5.60000	1.80000
104	6.50000	3.00000	5.80000	2.20000
105	7.60000	3.00000	6.60000	2.10000
106	4.90000	2.50000	4.50000	1.70000

iris.target

	97	98	99	100	101	102	1
0		1	1	2	2	2	2

Logistic Regression

- Let's transform this problem into a binary problem (**<u>use first 100 samples</u>**)
- Now your labels are either 0 or 1

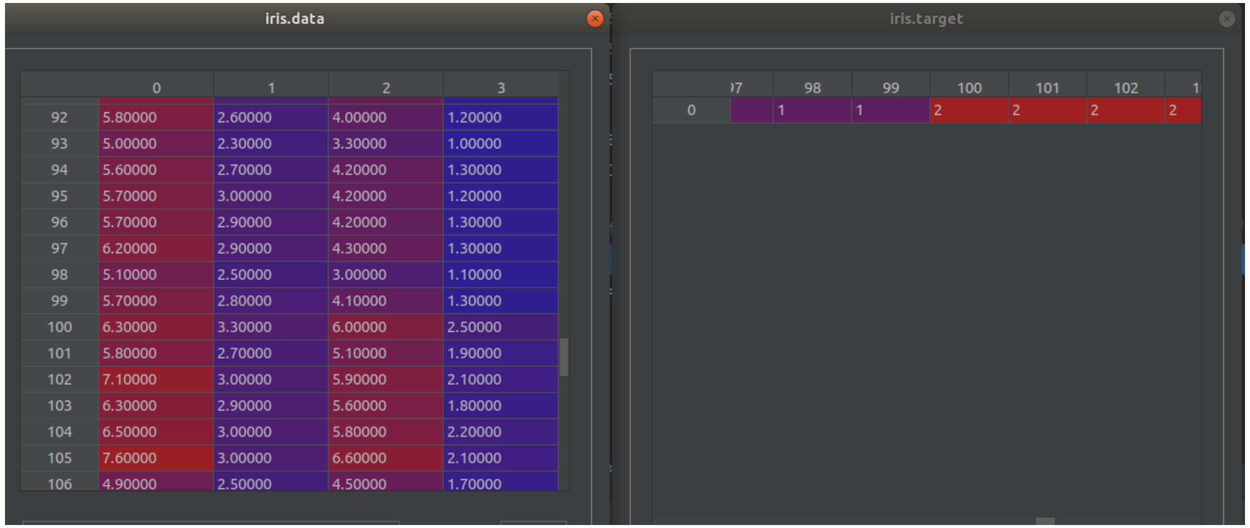


iris.data

	0	1	2	3
0	5.10000	3.50000	1.40000	0.20000
1	4.90000	3.00000	1.40000	0.20000
2	4.70000	3.20000	1.30000	0.20000
3	4.60000	3.10000	1.50000	0.20000
4	5.00000	3.60000	1.40000	0.20000
5	5.40000	3.90000	1.70000	0.40000
6	4.60000	3.40000	1.40000	0.30000
7	5.00000	3.40000	1.50000	0.20000
8	4.40000	2.90000	1.40000	0.20000
9	4.90000	3.10000	1.50000	0.10000
10	5.40000	3.70000	1.50000	0.20000
11	4.80000	3.40000	1.60000	0.20000
12	4.80000	3.00000	1.40000	0.10000
13	4.30000	3.00000	1.10000	0.10000
14	5.80000	4.00000	1.20000	0.20000

iris.target

	0	1	2	3	4	5
0	0	0	0	0	0	0

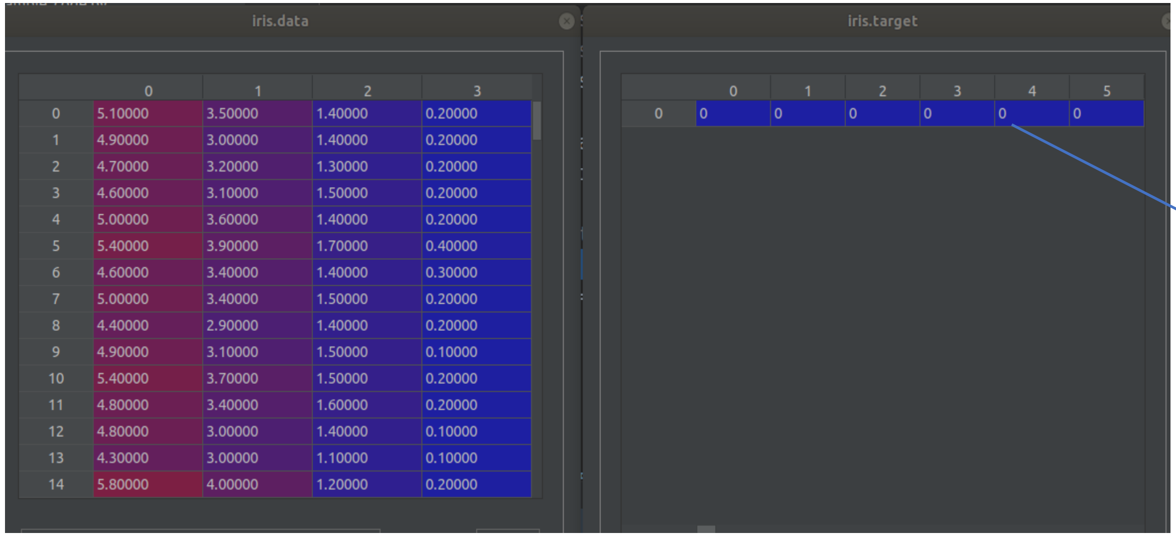


iris.data

	0	1	2	3
92	5.80000	2.60000	4.00000	1.20000
93	5.00000	2.30000	3.30000	1.00000
94	5.60000	2.70000	4.20000	1.30000
95	5.70000	3.00000	4.20000	1.20000
96	5.70000	2.90000	4.20000	1.30000
97	6.20000	2.90000	4.30000	1.30000
98	5.10000	2.50000	3.00000	1.10000
99	5.70000	2.80000	4.10000	1.30000
100	6.30000	3.30000	6.00000	2.50000
101	5.80000	2.70000	5.10000	1.90000
102	7.10000	3.00000	5.90000	2.10000
103	6.30000	2.90000	5.60000	1.80000
104	6.50000	3.00000	5.80000	2.20000
105	7.60000	3.00000	6.60000	2.10000
106	4.90000	2.50000	4.50000	1.70000

iris.target

	97	98	99	100	101	102	1
0		1	1	2	2	2	2



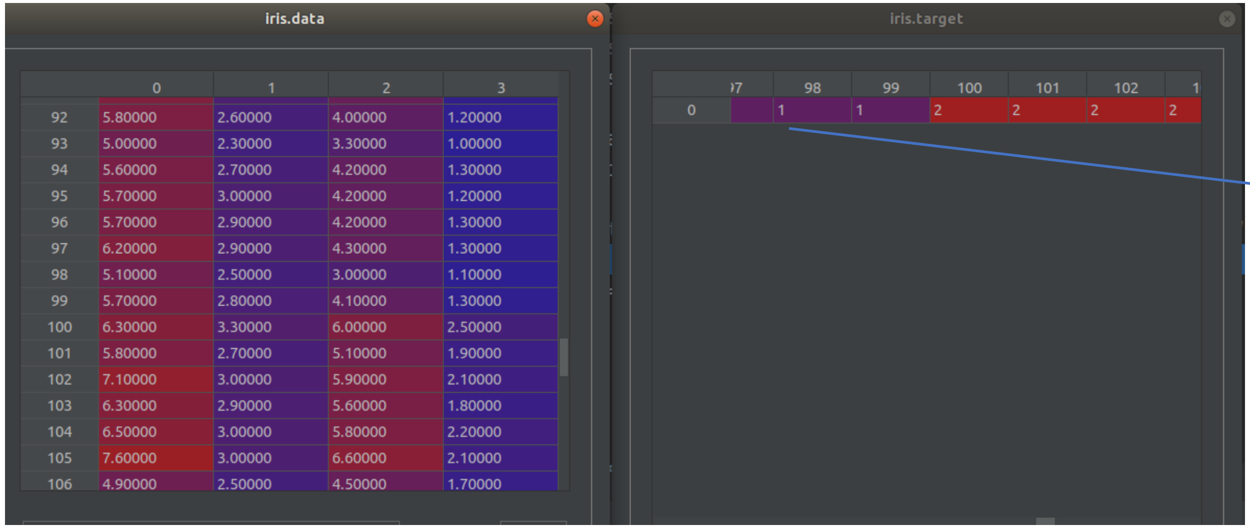
iris.data

	0	1	2	3
0	5.10000	3.50000	1.40000	0.20000
1	4.90000	3.00000	1.40000	0.20000
2	4.70000	3.20000	1.30000	0.20000
3	4.60000	3.10000	1.50000	0.20000
4	5.00000	3.60000	1.40000	0.20000
5	5.40000	3.90000	1.70000	0.40000
6	4.60000	3.40000	1.40000	0.30000
7	5.00000	3.40000	1.50000	0.20000
8	4.40000	2.90000	1.40000	0.20000
9	4.90000	3.10000	1.50000	0.10000
10	5.40000	3.70000	1.50000	0.20000
11	4.80000	3.40000	1.60000	0.20000
12	4.80000	3.00000	1.40000	0.10000
13	4.30000	3.00000	1.10000	0.10000
14	5.80000	4.00000	1.20000	0.20000

iris.target

	0	1	2	3	4	5
0	0	0	0	0	0	0

Iris-Setosa
Y=0



iris.data

	0	1	2	3
92	5.80000	2.60000	4.00000	1.20000
93	5.00000	2.30000	3.30000	1.00000
94	5.60000	2.70000	4.20000	1.30000
95	5.70000	3.00000	4.20000	1.20000
96	5.70000	2.90000	4.20000	1.30000
97	6.20000	2.90000	4.30000	1.30000
98	5.10000	2.50000	3.00000	1.10000
99	5.70000	2.80000	4.10000	1.30000
100	6.30000	3.30000	6.00000	2.50000
101	5.80000	2.70000	5.10000	1.90000
102	7.10000	3.00000	5.90000	2.10000
103	6.30000	2.90000	5.60000	1.80000
104	6.50000	3.00000	5.80000	2.20000
105	7.60000	3.00000	6.60000	2.10000
106	4.90000	2.50000	4.50000	1.70000

iris.target

	)7	98	99	100	101	102	1
0		1	1	2	2	2	2

Iris-Versicolor
Y=1

WS4: Task 1

- **Use GD and build a classifier to detect the Iris-Virginica type: <u>Iris-Setosa (Y=0)</u> or <u>Iris-Versicolor (Y=1).</u>**

(i) Explain the whole procedure in your words

(ii) find the accuracy

- **Here is the pre-processing code:**

```
from sklearn import datasets
iris = datasets.load_iris()
list(iris.keys())
X = iris["data"] # petal width
X=X[0:99]
X = normalize(X, norm='l2')
y = iris["target"]
y=np.reshape(y, (1, 150))
y=y.T
y=y[0:99]
y = normalize(y, norm='l2')
```

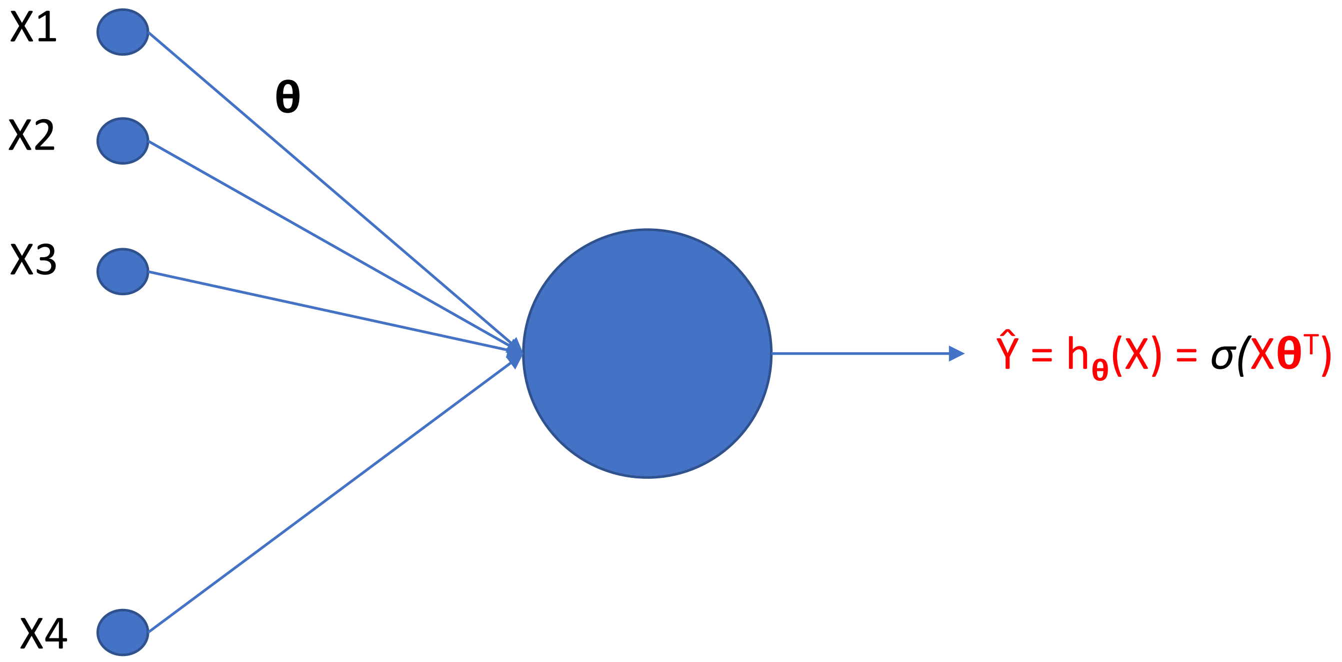
- **Hints:** $\hat{Y} = h_{\boldsymbol{\theta}}(X) = \sigma(X\boldsymbol{\theta}^T)$

$$\nabla_\theta J(\theta) = \sum_i x^{(i)}(h_\theta(x^{(i)}) - y^{(i)})$$

$$\boldsymbol{\theta}^{(\text{ next step })} = \boldsymbol{\theta} - \eta \nabla_\theta J(\theta)$$

- **Hints:**

WS4: Task 1



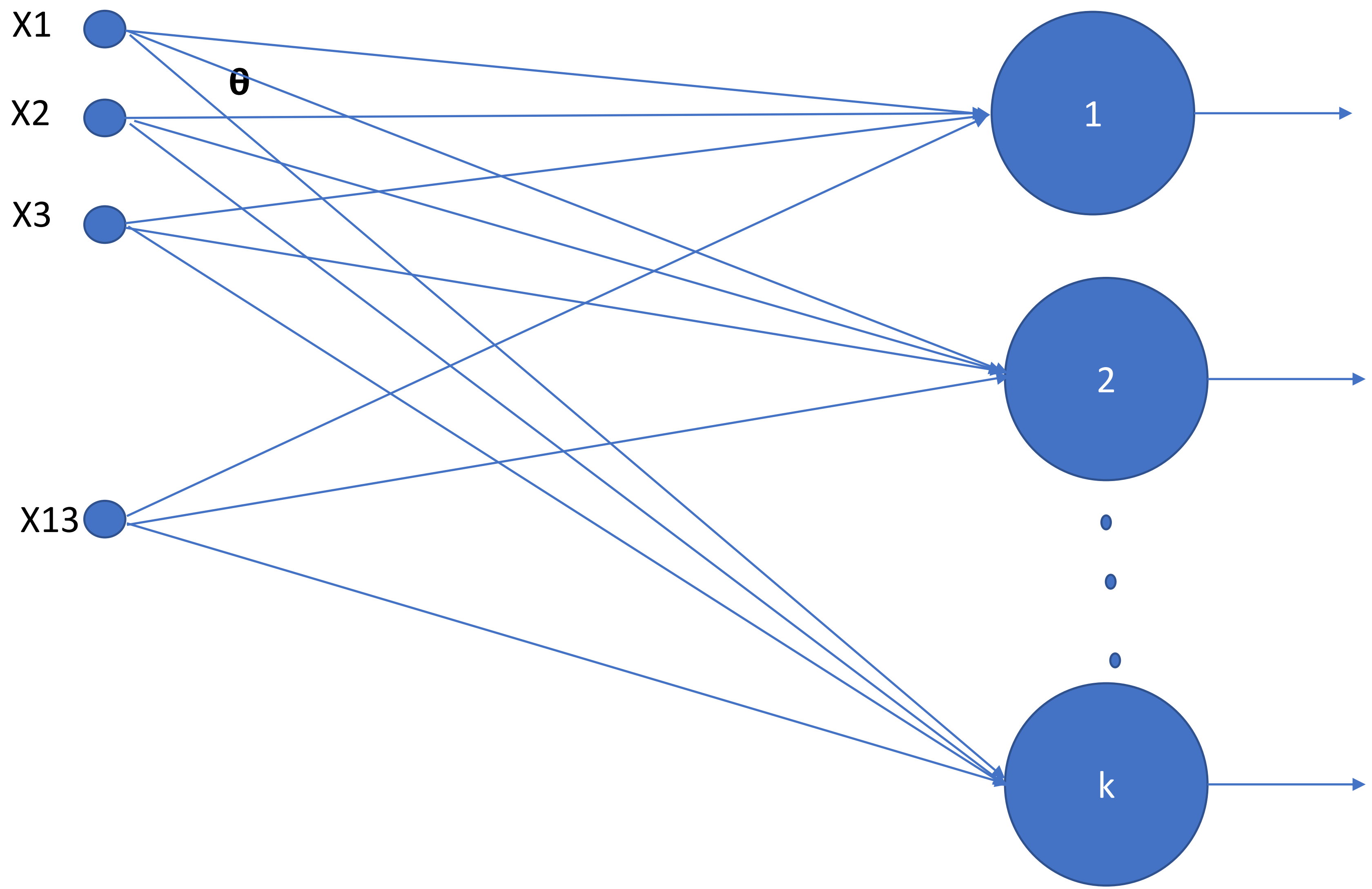
Softmax regression

- Softmax regression (or multinomial logistic regression) is a generalization of logistic regression to the case where we want to handle multiple classes.

- In logistic regression we assumed that the labels were binary: $y(i) \in \{0,1\}$. We used such a classifier to distinguish between two kinds of hand-written digits.

- Softmax regression allows us to handle $y(i) \in \{1,\ldots,K\}$ where K is the number of classes.

Softmax regression



WS4: Task 2

- **Use GD and build a classifier to detect the Iris-Virginica type: <u>Iris-Setosa</u> or <u>Iris-Versicolor</u> or <u>Iris-Virginica.</u>**
- **(i) Explain the whole procedure in your words**
- **(ii) find the accuracy**

Here is the pre-processing code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import normalize
import scipy.sparse
from sklearn import datasets
iris = datasets.load_iris()
list(iris.keys())
X = iris["data"]

X = normalize(X, norm='l2')
y = iris["target"]



def oneHotIt(Y):
    m = Y.shape[0]
    #Y = Y[:,0]
    OHX = scipy.sparse.csr_matrix((np.ones(m),

(Y, np.array(range(m)))))
    OHX = np.array(OHX.todense()).T
    return OHX

y_mat = oneHotIt(y)

  y=y_mat
```

References:

- http://deeplearning.stanford.edu/tutorial