Logistic (regression) classification

Regression

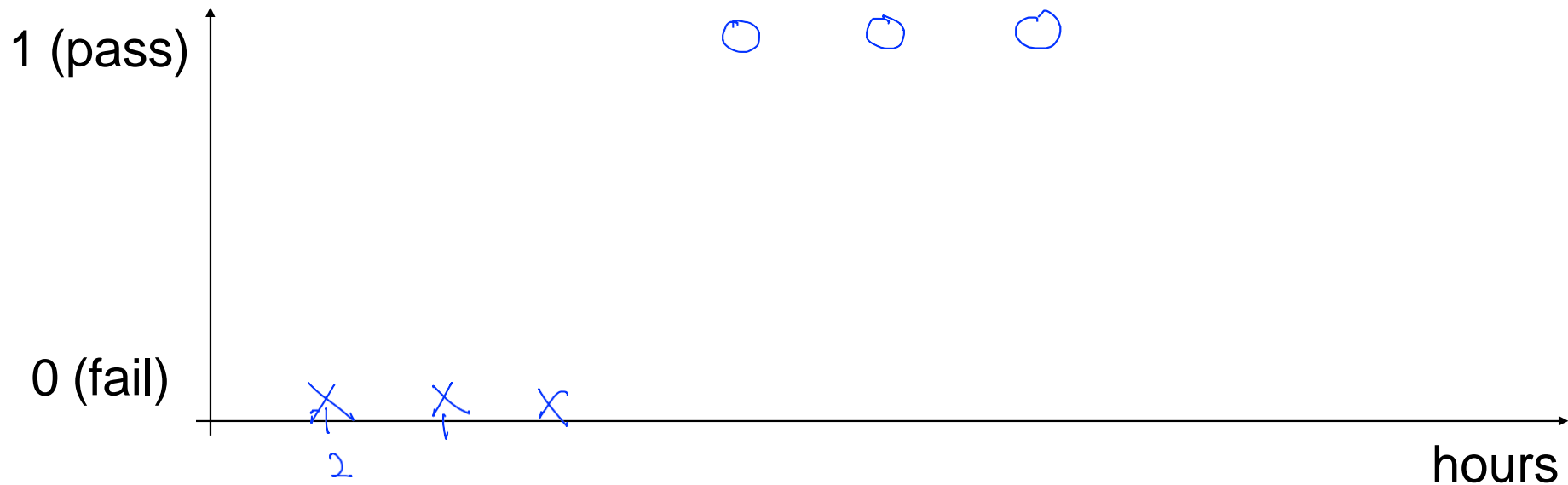| x1 (hours)	x2 (attendance)	y (score)
10	5	90
9	5	80
3	2	50
2	4	60
11	1	40

- Hypothesis: $H(X) = WX$

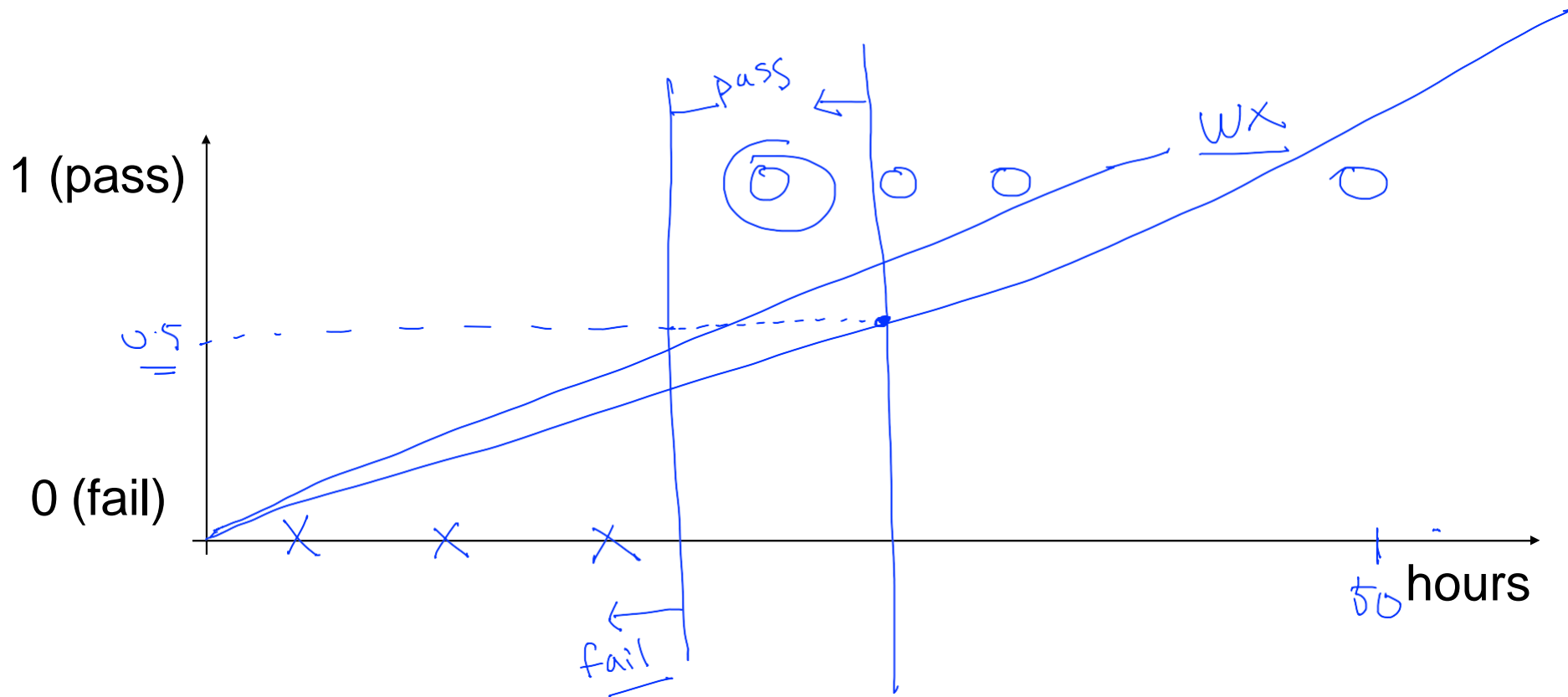- Cost: $cost(W) = \dfrac{1}{m}\sum (WX - y)^2$
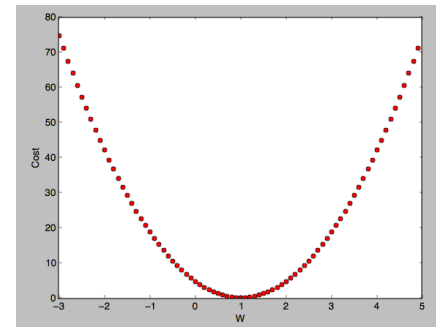
- Gradient decent:
$$W := W - \alpha \frac{\partial}{\partial W} cost(W)$$
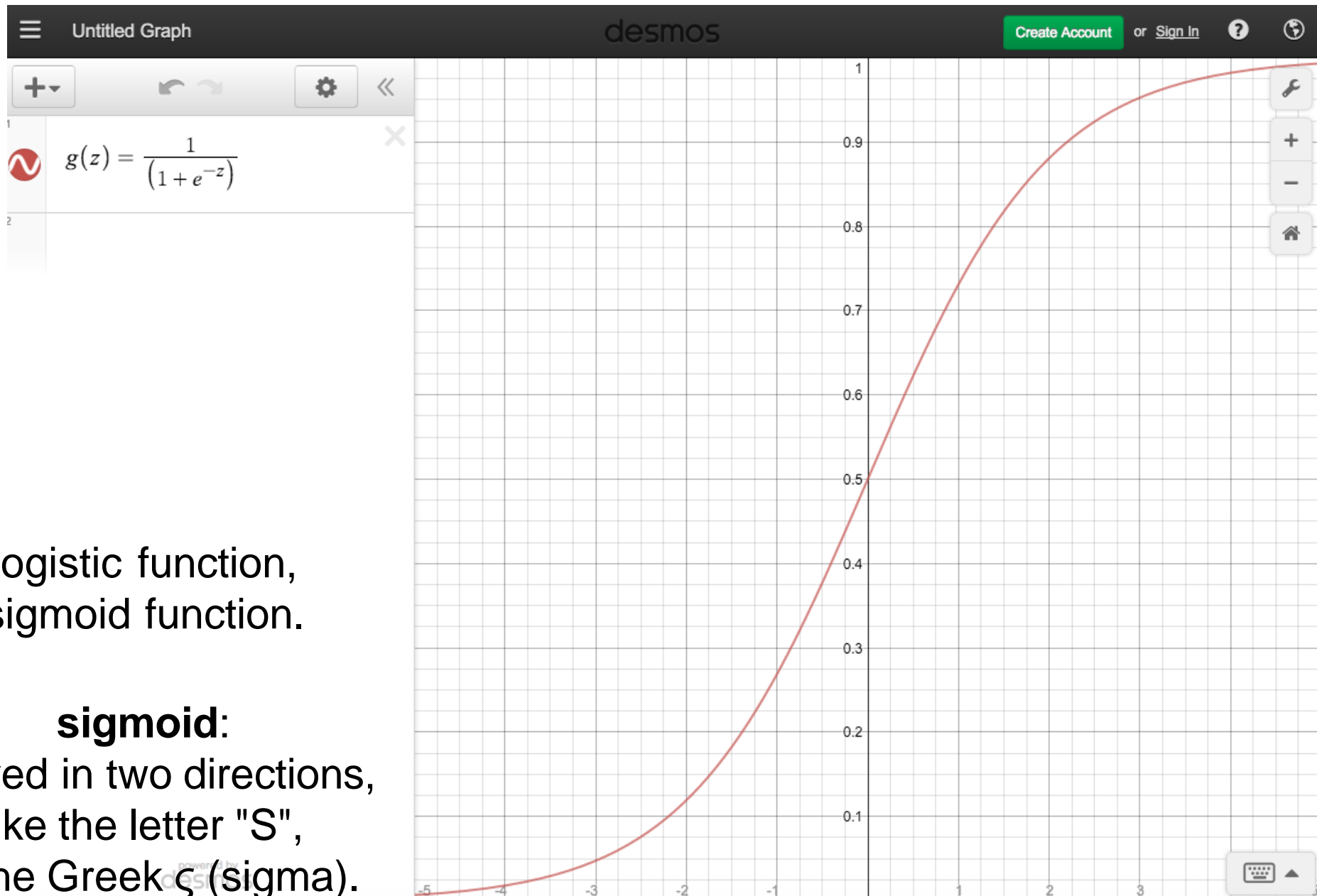
http://www.cse.iitk.ac.in/users/se367/10/presentation_local/Binary%20Classification.html

Pass(1)/Fail(0) based on study hours

Linear Regression?

1 (pass)

0 (fail)

pass

fail

wx

$=0.5$

50 hours
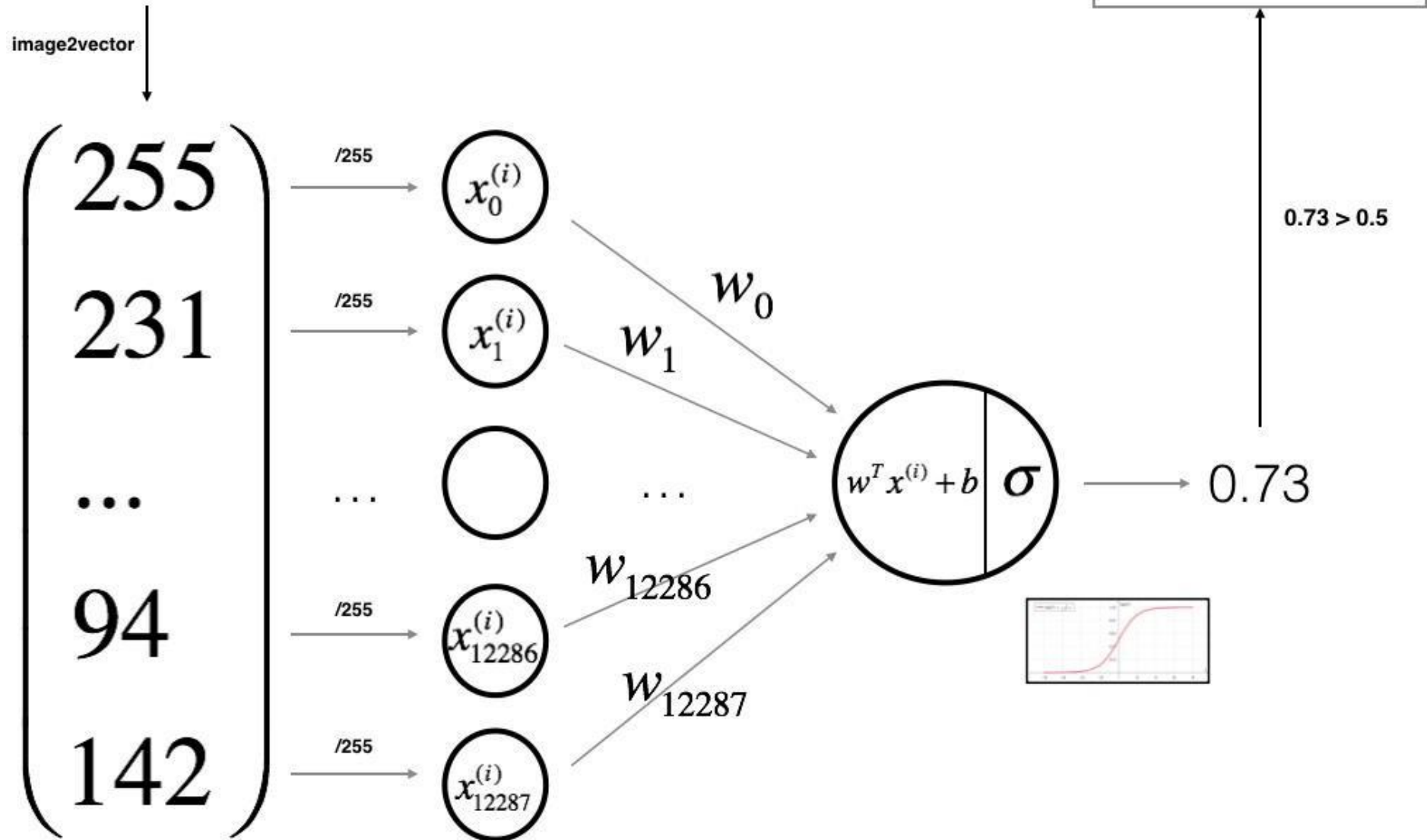
4

logistic function,
sigmoid function.

sigmoid:
Curved in two directions,
like the letter "S",
or the Greek ς (sigma).

Logistic Hypothesis

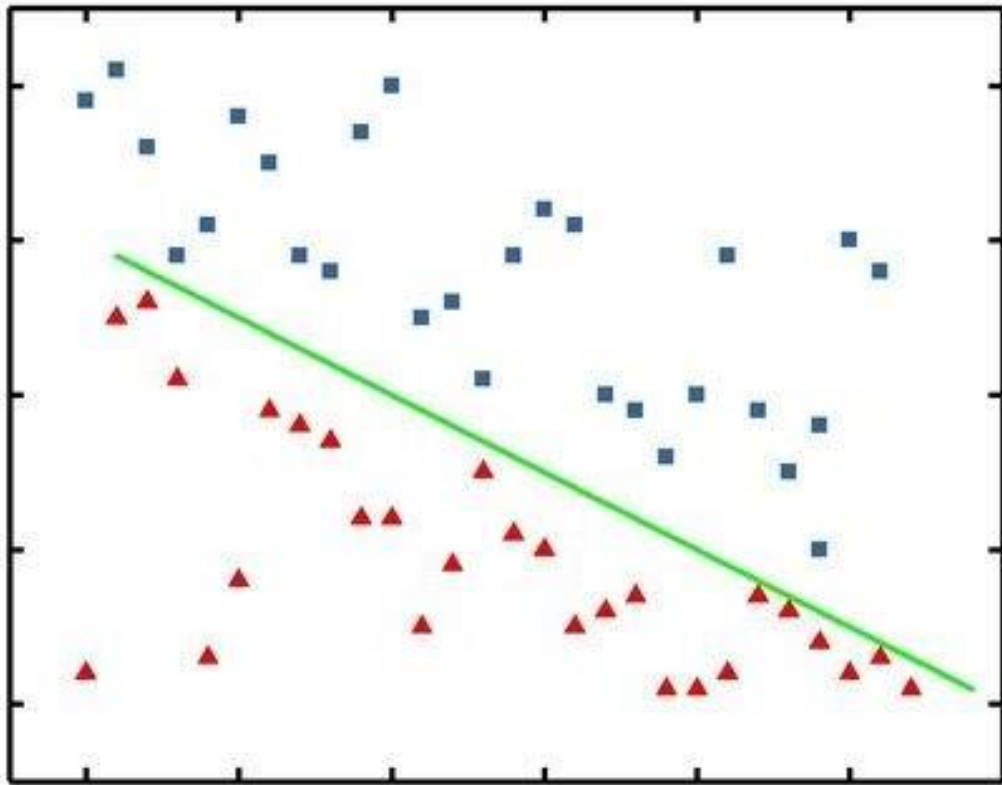$$H(X) = \frac{1}{1 + e^{-(W^T X)}}$$

logistic regression as **a one layer neural network**

"it's a cat"

image2vector

$0.73 > 0.5$

$$\begin{pmatrix} 255 \\ 231 \\ \dots \\ 94 \\ 142 \end{pmatrix}$$

/255 $\longrightarrow$ $x_0^{(i)}$

/255 $\longrightarrow$ $x_1^{(i)}$

$\dots$

/255 $\longrightarrow$ $x_{12286}^{(i)}$

/255 $\longrightarrow$ $x_{12287}^{(i)}$

w_0

w_1

w_{12286}

w_{12287}

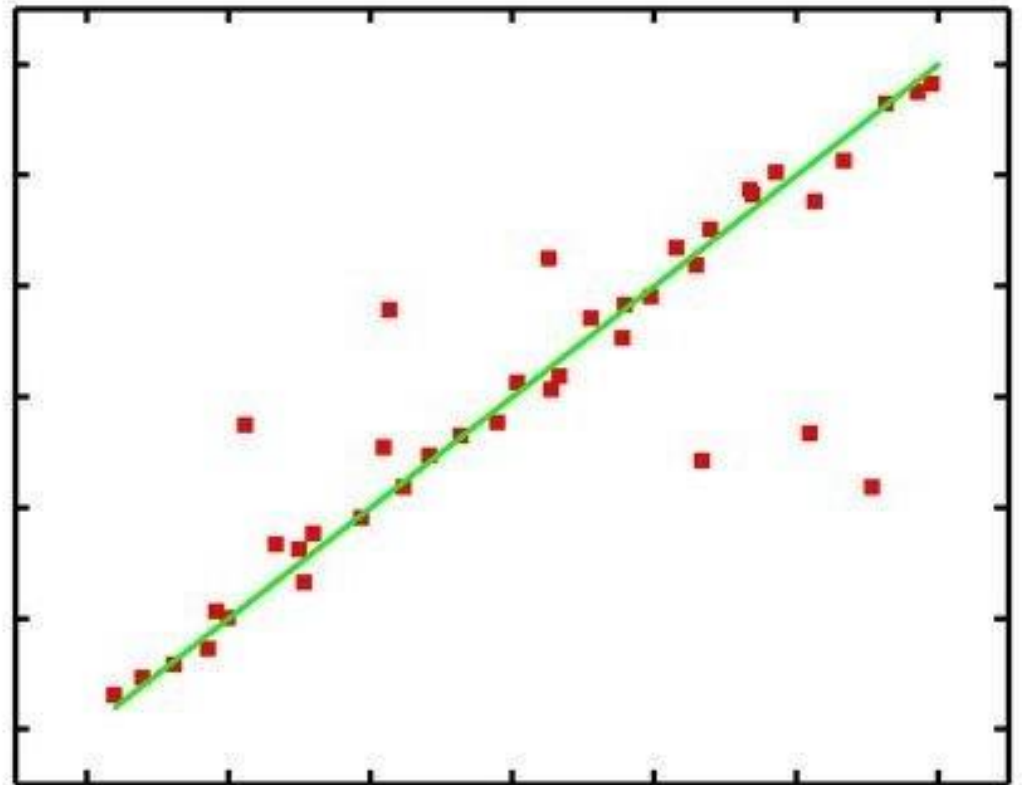$w^T x^{(i)} + b \mid \sigma$ $\longrightarrow$ 0.73
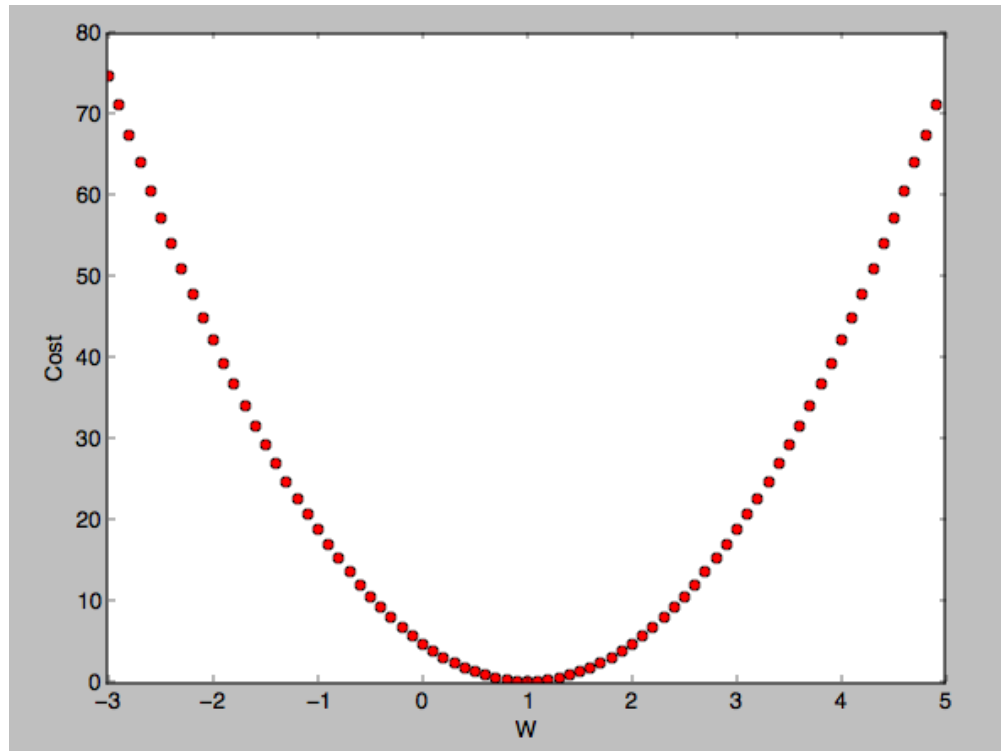
Logistic Regression

$$H(x) = Wx + b$$

(a) Logistic Regression

(b) Linear Regression

Cost

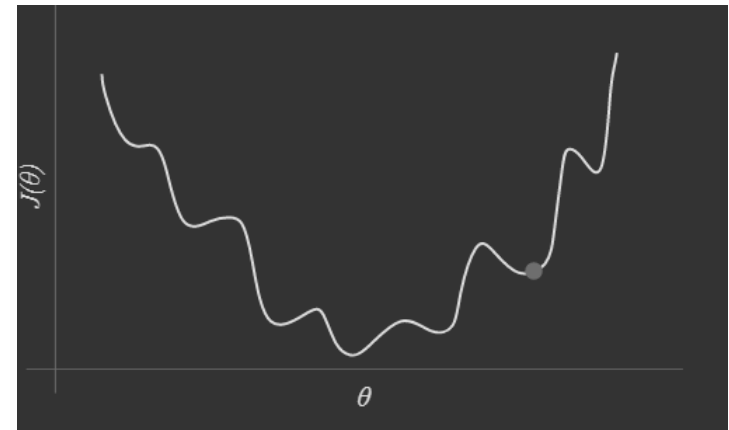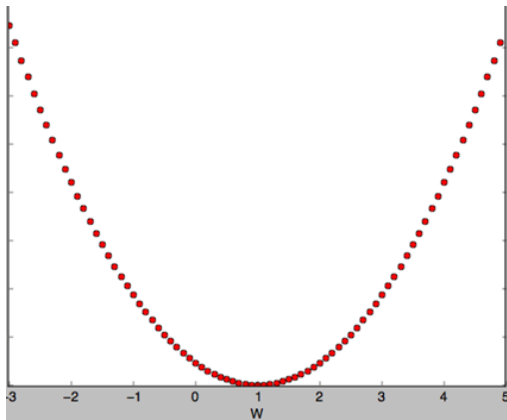$$cost(W, b) = \frac{1}{m} \sum_{i=1}^{m} (H(x^{(i)}) - y^{(i)})^2 \quad \text{when} \quad H(x) = Wx + b$$

Cost function

$$cost(W, b) = \frac{1}{m} \sum_{i=1}^{m} (H(x^{(i)}) - y^{(i)})^2$$

$$H(x) = Wx + b$$
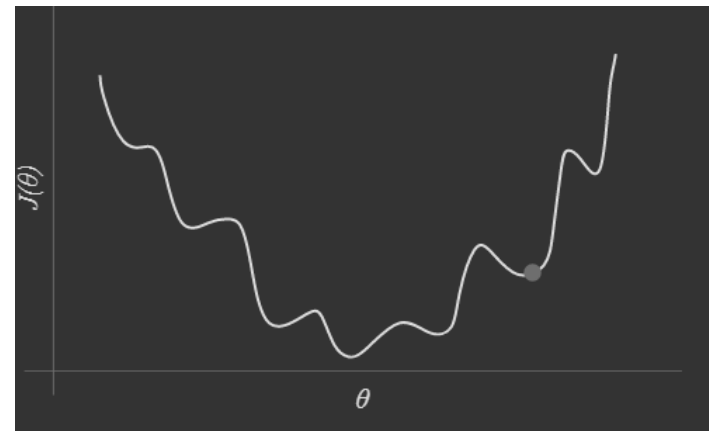
$$H(X) = \frac{1}{1 + e^{W^T X}}$$

Cost

$$cost(W, b) = \frac{1}{m} \sum_{i=1}^{m} (H(x^{(i)}) - y^{(i)})^2 \quad \text{when} \quad H(x) = Wx + b$$

$$H(X) = \frac{1}{1 + e^{-W^T X}}$$

$$\text{cost}\,(h_{\boldsymbol{\theta}}(\boldsymbol{x}),\, y) = \begin{cases} -\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 1 \\ -\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 0 \end{cases}$$

New cost function for logistic

$$cost(W) = \frac{1}{m} \sum c(H(x), y)$$

$$c(H(x), y) = \begin{cases} -log(H(x)) & : y = 1 \\ -log(1 - H(x)) & : y = 0 \end{cases}$$

Cross entropy cost

Logistic cross entropy cost

$$\text{cost}\,(h_\theta(x),\ y) = -y\log(h_\theta(x)) - (1-y)\log(1 - h_\theta(x))$$

$$\text{cost}\,(h_\theta(x),\ y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

if $y = 1$

if $y = 0$

if $h_\theta(x) = 1$
then cost $= 0$

if $h_\theta(x) \to 0$
then cost $\to \infty$
predicted
prob$(y = 1\,|\,x; \theta) = 0$
but $y = 1$

if $h_\theta(x) = 0$
then cost $= 0$

if $h_\theta(x) \to 1$
then cost $\to \infty$
predicted
prob$(y = 0\,|\,x; \theta) = 0$
but $y = 0$

Cost function

$$cost(W) = \frac{1}{m} \sum c(H(x), y)$$

$$c(H(x), y) = \begin{cases} -log(H(x)) & : y = 1 \\ -log(1 - H(x)) & : y = 0 \end{cases}$$

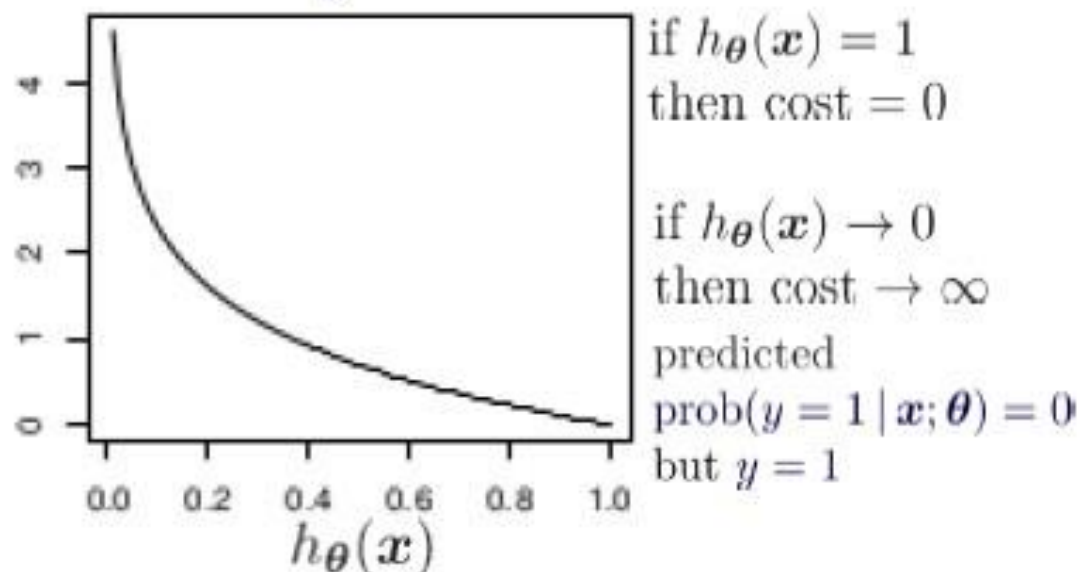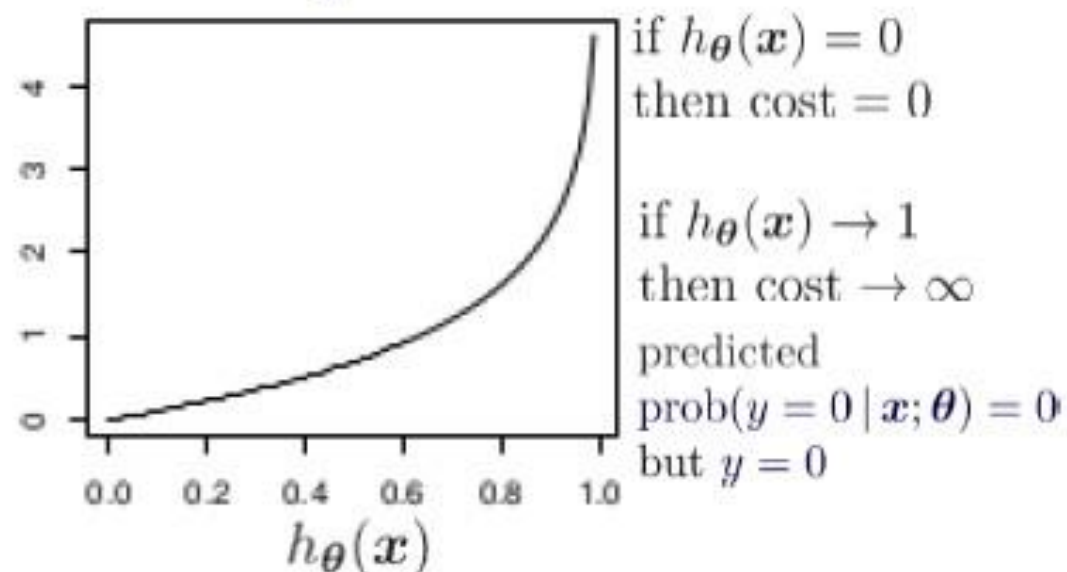$$C(H(x), y) = -ylog(H(x)) - (1 - y)log(1 - H(x))$$

Minimize cost - Gradient decent algorithm

$$cost(W) = -\frac{1}{m} \sum ylog(H(x)) + (1-y)log(1-H(x))$$

$$W := W - \alpha \frac{\partial}{\partial W} cost(W)$$

```
# cost function
cost = tf.reduce_mean(-tf.reduce_sum(Y*tf.log(hypothesis) + (1-Y)*tf.log(1-hypothesis)))

# Minimize
a = tf.Variable(0.1) # Learning rate, alpha
optimizer = tf.train.GradientDescentOptimizer(a)
train = optimizer.minimize(cost)
```
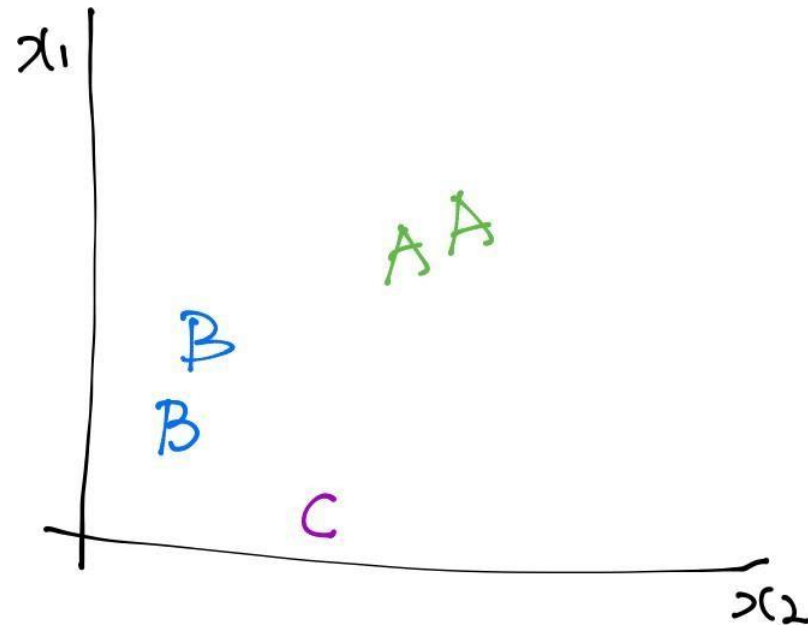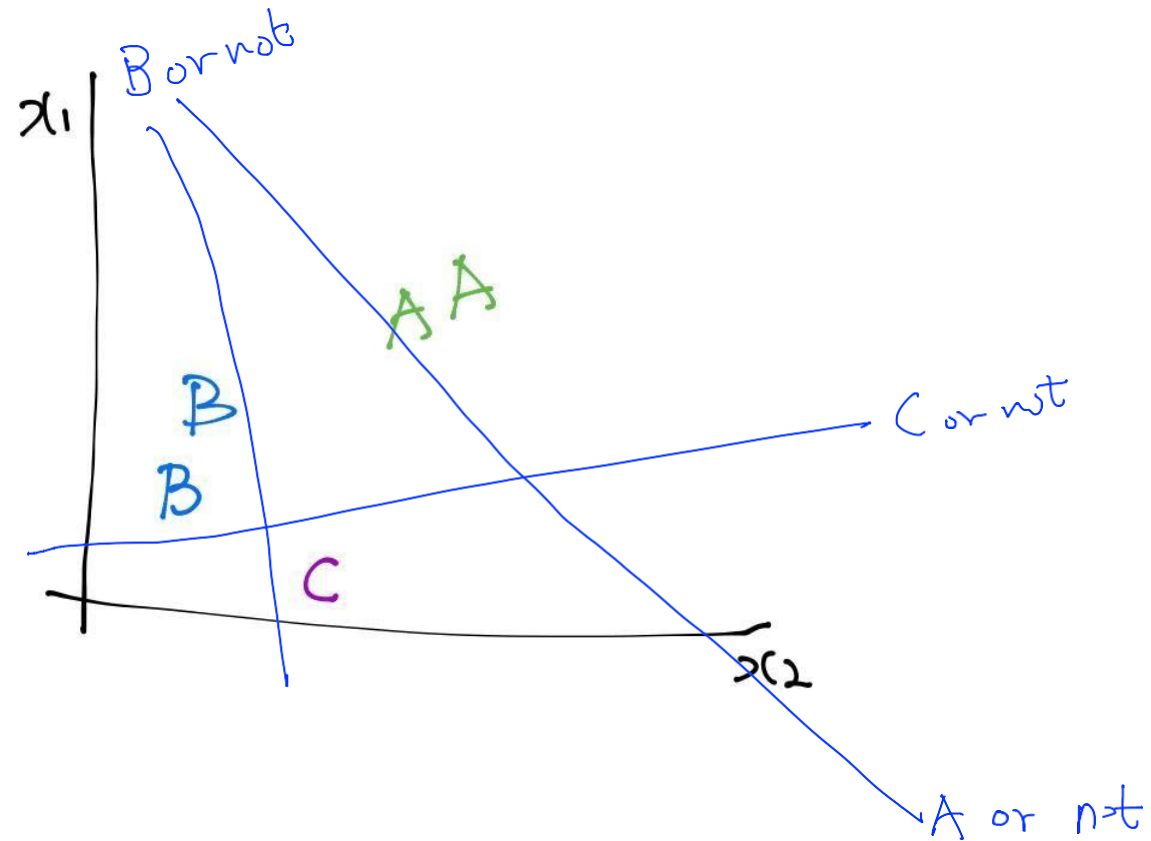
Classification

- Spam Detection: Spam or Ham
- Facebook feed: show or hide

- Credit Card Fraudulent Transaction detection: legitimate/fraud

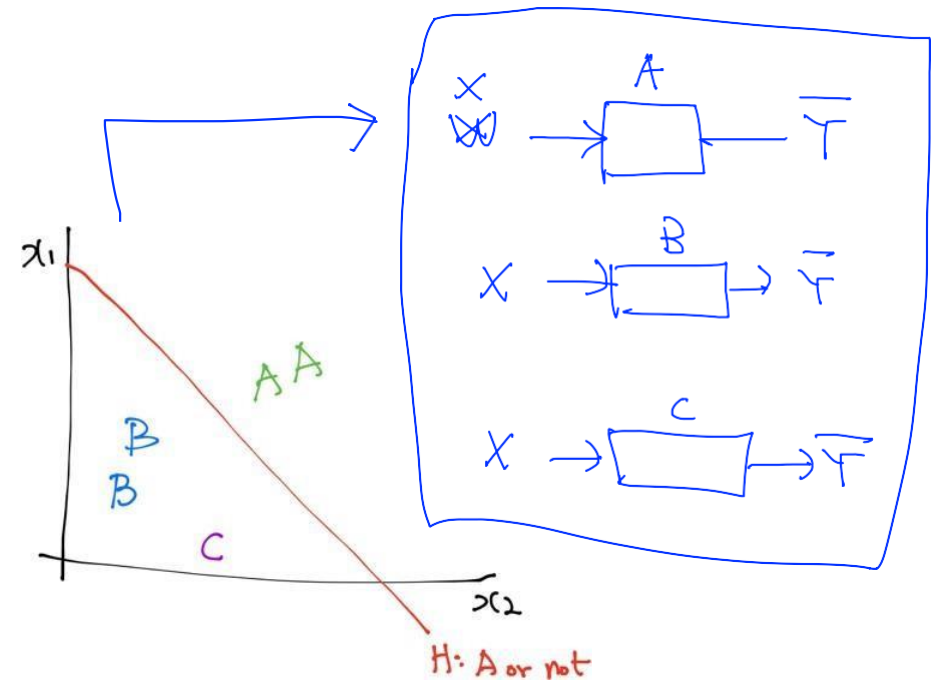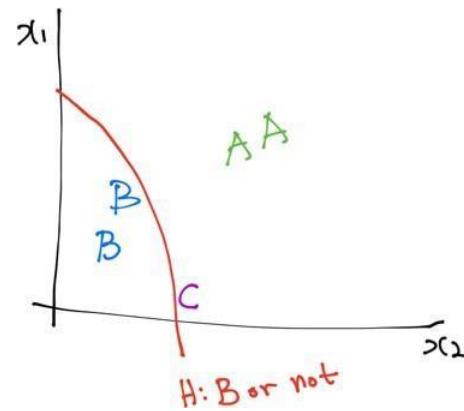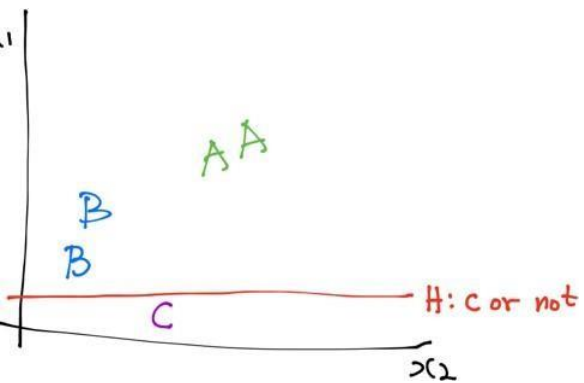# Multinomial classification

x1 (hours)	x2 (attendance)	y (grade)
10	5	A
9	5	A
3	2	B
2	4	B
11	1	C

Multinomial classification

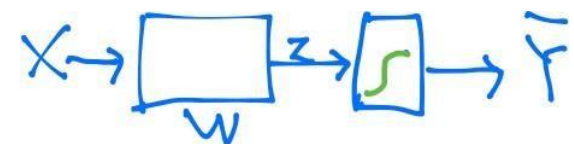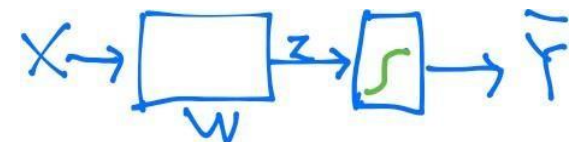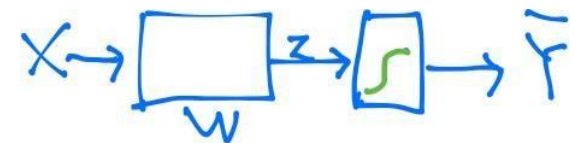# Multinomial classification

Multinomial classification

$$[w_1 \ w_2 \ w_3] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = [w_1 x_1 + w_2 x_2 + w_3 x_3]$$

$$[w_1 \ w_2 \ w_3] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = [w_1 x_1 + w_2 x_2 + w_3 x_3]$$

$$[w_1 \ w_2 \ w_3] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = [w_1 x_1 + w_2 x_2 + w_3 x_3]$$

Multinomial classification

$$\begin{bmatrix} w_{A1} & w_{A2} & w_{A3} \\ w_{B1} & w_{B2} & w_{B3} \\ w_{C1} & w_{C2} & w_{C3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} w_{A1}x_1 + w_{A2}x_2 + w_{A3}x_3 \\ w_{B1}x_1 + w_{B2}x_2 + w_{B3}x_3 \\ w_{C1}x_1 + w_{C2}x_2 + w_{C3}x_3 \end{bmatrix} = \begin{bmatrix} \bar{y}_A \\ \bar{y}_B \\ \bar{y}_C \end{bmatrix}$$

A

B

C

$X \to \boxed{}_{w} \xrightarrow{z} \boxed{\int} \to \tilde{Y}$

$X \to \boxed{}_{w} \xrightarrow{z} \boxed{\int} \to \tilde{Y}$

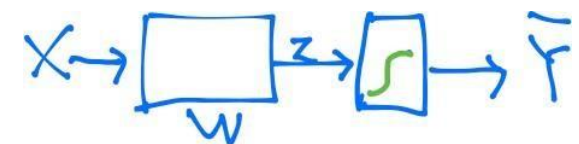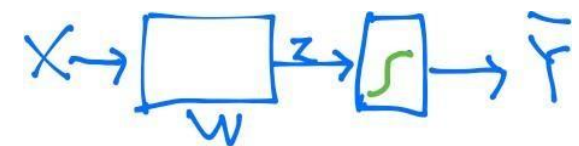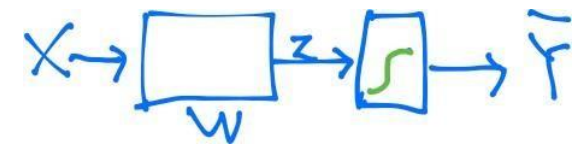$X \to \boxed{}_{w} \xrightarrow{z} \boxed{\int} \to \tilde{Y}$

Where is sigmoid?

$$\begin{bmatrix} w_{A1} & w_{A2} & w_{A3} \\ w_{B1} & w_{B2} & w_{B3} \\ w_{C1} & w_{C2} & w_{C3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} w_{A1}x_1 + w_{A2}x_2 + w_{A3}x_3 \\ w_{B1}x_1 + w_{B2}x_2 + w_{B3}x_3 \\ w_{C1}x_1 + w_{C2}x_2 + w_{C3}x_3 \end{bmatrix} = \begin{bmatrix} \overline{y_A} \\ \overline{y_B} \\ \overline{y_C} \end{bmatrix} \begin{bmatrix} 2.0 \\ 1.0 \\ 0.1 \end{bmatrix}$$

Softmax is used for multi-classification in the Logistic Regression model

Sigmoid is used for binary classification in the Logistic Regression model.

1. there is more than one "right answer"
2. there is only one "right answer"

Sigmoid/Softmax?

$$f(s_i) = \frac{1}{1 + e^{-s_i}}$$

$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}}$$

SOFTMAX

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

$$Y \begin{bmatrix} 2.0 \\ 1.0 \\ 0.1 \end{bmatrix} \rightarrow \begin{matrix} 0.7 \\ 0.2 \\ 0.1 \end{matrix}$$

SCORES $\longrightarrow$ PROBABILITIES

24
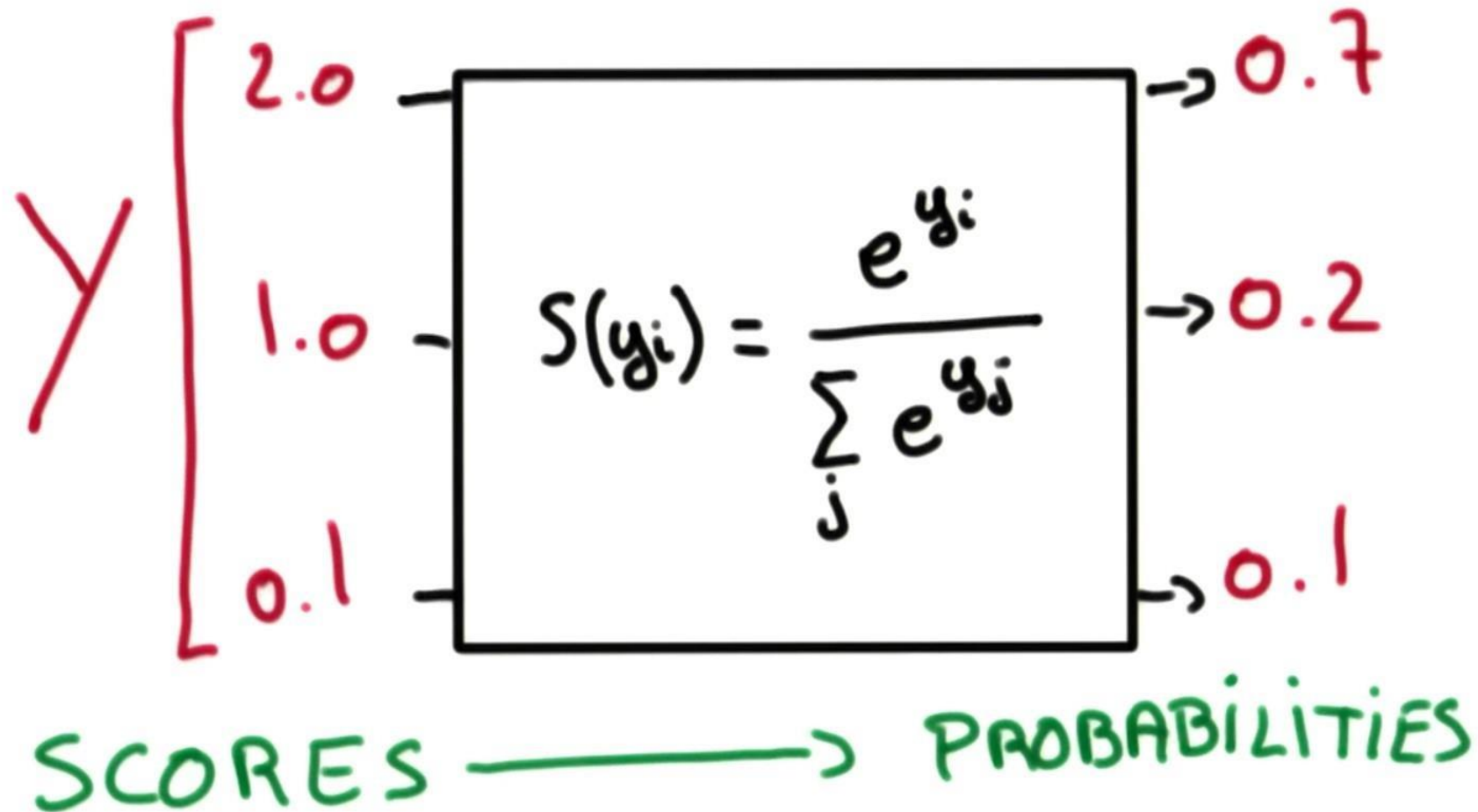
Sigmoid vs Softmax

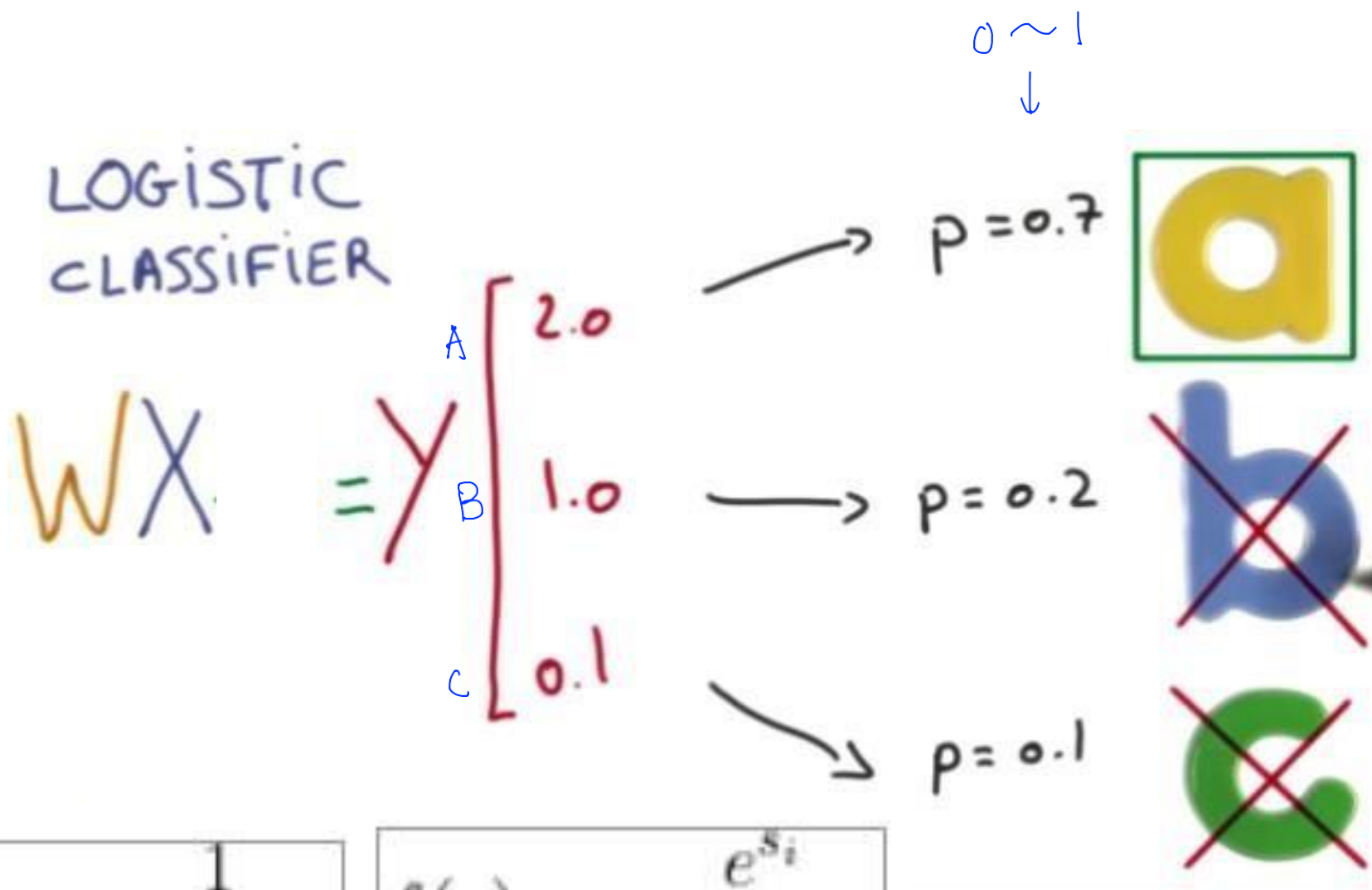| Softmax Function	Sigmoid Fundtion
Used for multi-classification in logistic regression model	Used for binary classification in logistic regression model
The probabilities sum will be 1	The probabilities sum need not to be 1

$Y \begin{bmatrix} 2.0 \\ 1.0 \\ 0.1 \end{bmatrix}$

$S(Y) \longrightarrow \begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix}$

'ONE-HOT' ENCODING

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

?

a

b

c
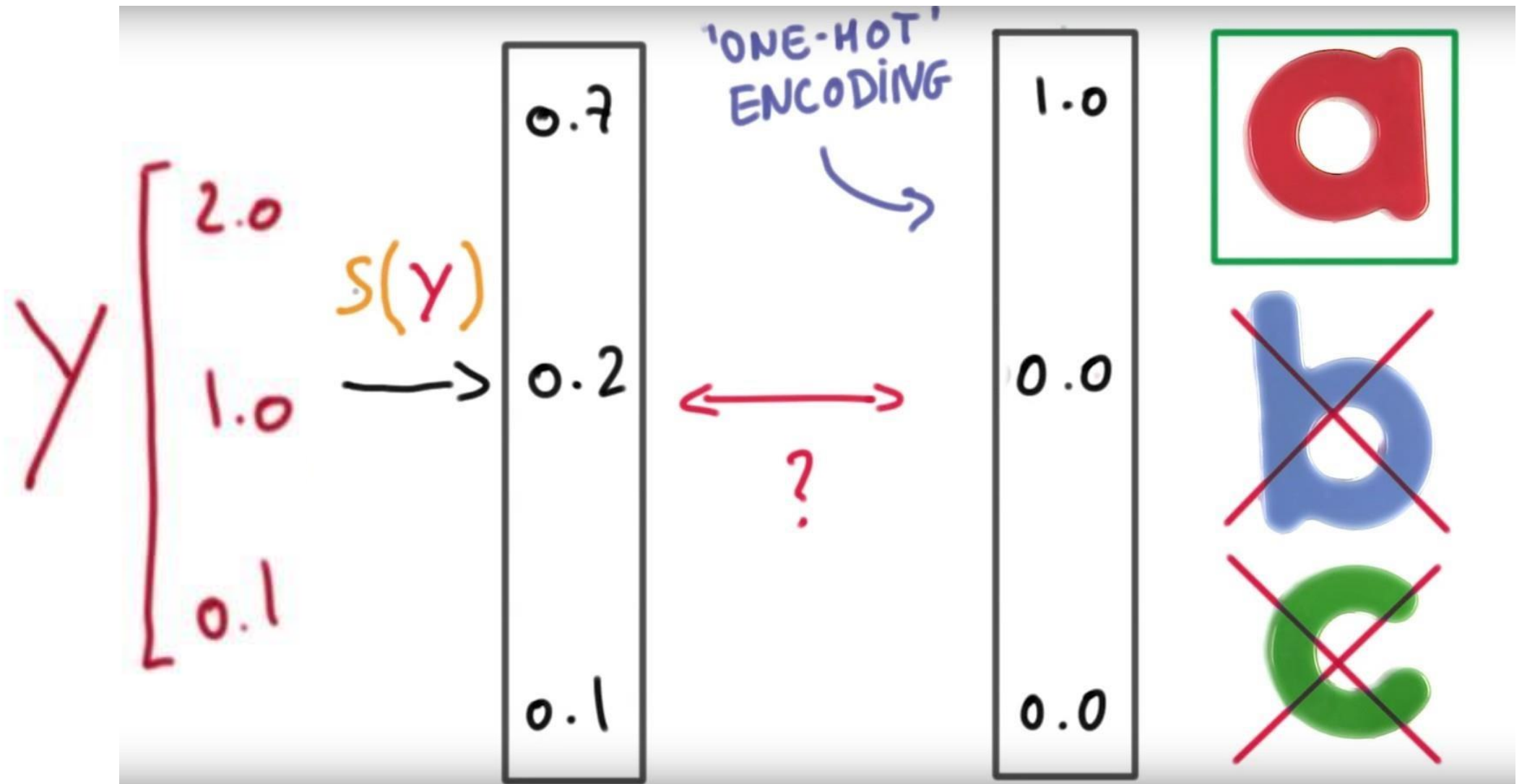
```
tf.matmul(X,W)+b
```

```
hypothesis = tf.nn.softmax(tf.matmul(X,W)+b)
```

LOGISTIC CLASSIFIER

$XW=Y$

$$\begin{bmatrix} 2.0 \\ 1.0 \\ 0.1 \end{bmatrix}$$

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

$\rightarrow 0.7$

$\rightarrow 0.2$

$\rightarrow 0.1$
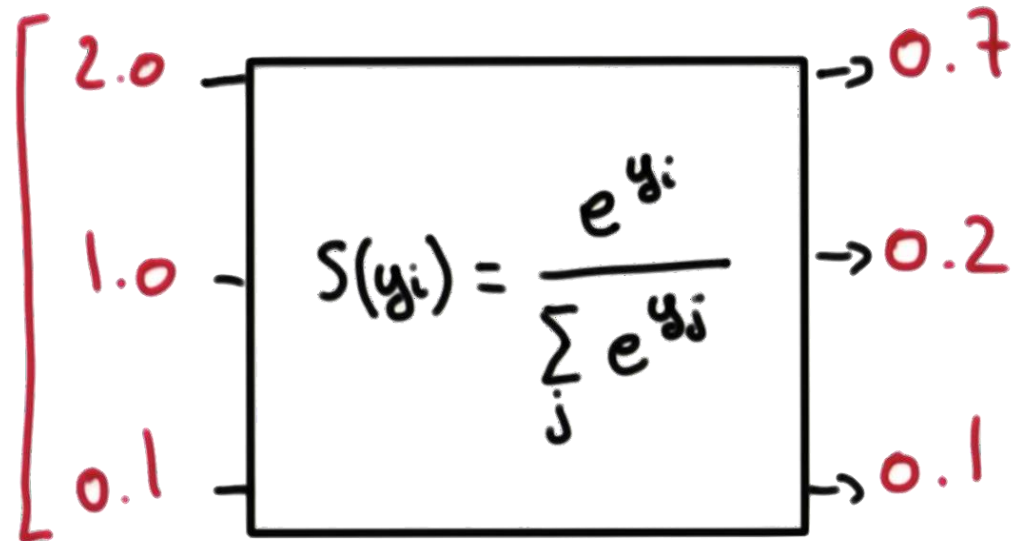
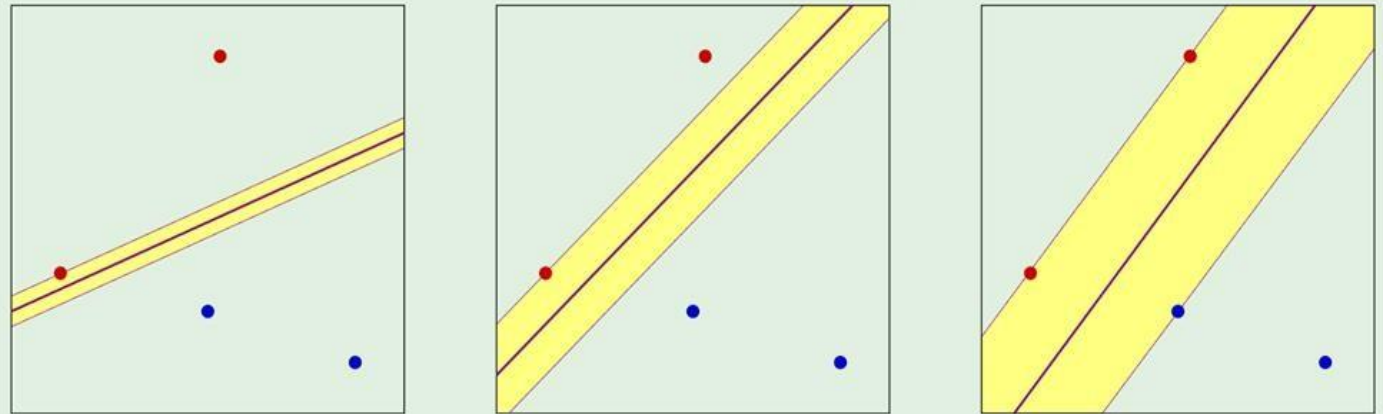SCORES $\longrightarrow$ PROBABILITIES

SVM
Support Vector Machines

Better linear separation

Linearly separable data

Different separating lines

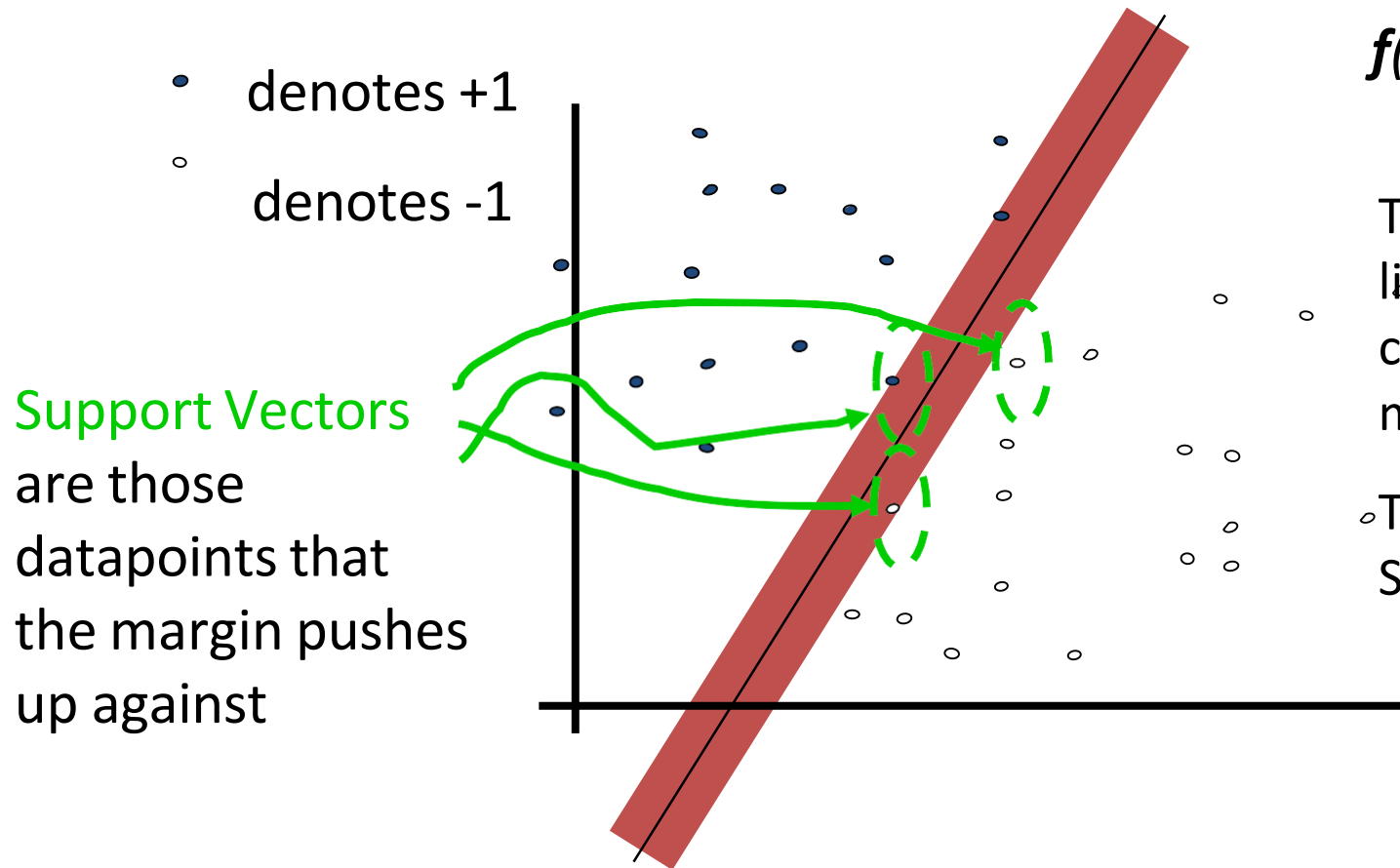Which is best?

Two questions:

1. Why is bigger margin better?

2. Which $\mathbf{w}$ maximizes the margin?

Why Maximum Margin?

denotes +1

denotes -1

Support Vectors
are those
datapoints that
the margin pushes
up against

$f(x, w, b) = sign(w. x - b)$

The maximum margin
linear classifier is the linear
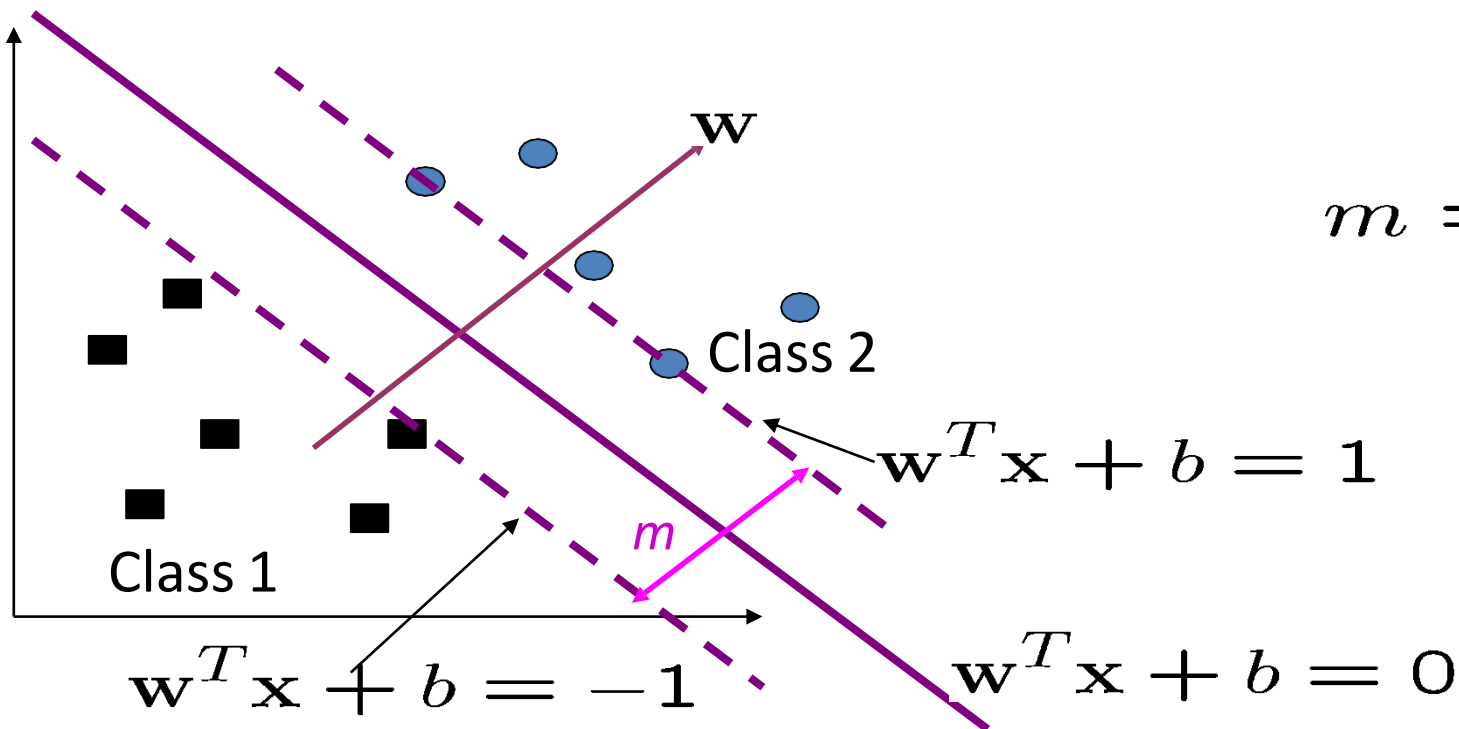classifier with the,
maximum margin.

This is the simplest kind of
SVM (Called an LSVM)

33

Large-margin Decision Boundary

The decision boundary should be as far away from the data of both classes as possible
We should maximize the margin, m
Distance between the origin and the line $\mathbf{w}^t\mathbf{x}$=-b is b/||$\mathbf{w}$||

$$m = \frac{2}{||\mathbf{w}||}$$

$\mathbf{w}$

Class 2

$\mathbf{w}^T\mathbf{x} + b = 1$

m

Class 1

$\mathbf{w}^T\mathbf{x} + b = -1$

$\mathbf{w}^T\mathbf{x} + b = 0$

Finding the Decision Boundary

Let $\{x_1, ..., x_n\}$ be our data set and let $y_i \in \{1, -1\}$ be the class label of x_i

The decision boundary should classify all points correctly

$$\Rightarrow \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \qquad \forall i$$

To see this: when y=-1, we wish (wx+b)<1, when y=1, we wish (wx+b)>1. For support vectors, we wish y(wx+b)=1. The decision boundary can be found by solving the following constrained optimization problem
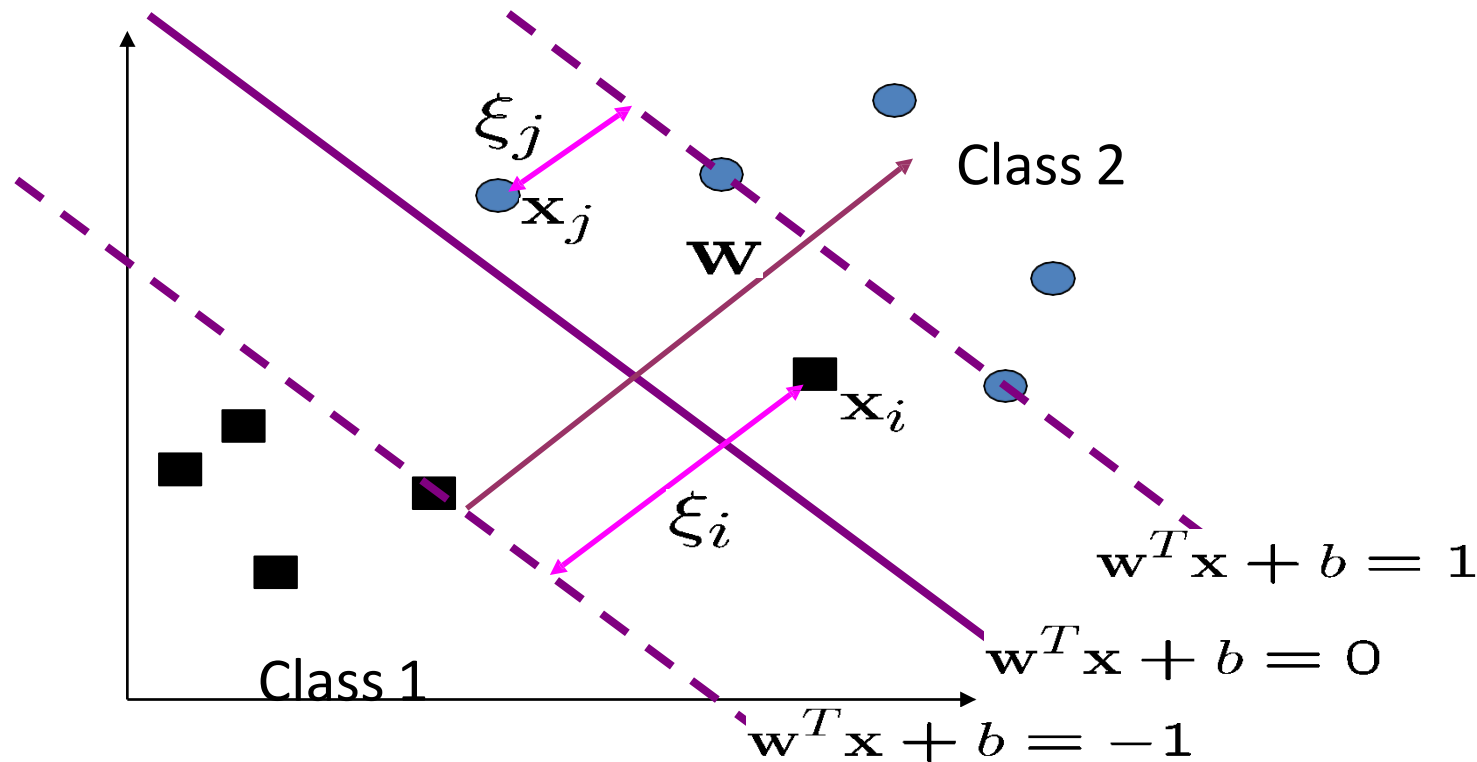
$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2$$
$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \qquad \forall i$$

Allowing errors in our solutions

We allow "error" ξ_i in classification; it is based on the output of the discriminant function $\mathbf{w}^\mathsf{T}\mathbf{x}+b$

ξ_i approximates the number of misclassified samples

Soft Margin Hyperplane

If we minimize $\sum$
$$\begin{cases} \mathbf{w}^T \mathbf{x}_i + b \geq 1 - \xi_i & y_i = 1 \\ \mathbf{w}^T \mathbf{x}_i + b \leq -1 + \xi_i & y_i = -1 \\ \xi_i \geq 0 & \forall i \end{cases}$$

ξ_i are "slack variables" in optimization
Note that $\xi_i=0$ if there is no error for $\mathbf{x}_i$
ξ_i is an upper bound of the number of errors

We want to minimize
$$\frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^n \xi_i$$

C : tradeoff parameter between error and margin

The optimization problem becomes

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^n \xi_i$$
$$\text{subject to } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

Extension to Non-linear Decision Boundary

So far, we have only considered large-margin classifier with a linear decision boundary

How to generalize it to become nonlinear?

Key idea: transform $\mathbf{x}_i$ to a higher dimensional space to "make life easier"

Input space: the space the point $\mathbf{x}_i$ are located
Feature space: the space of $\phi(\mathbf{x}_i)$ after transformation
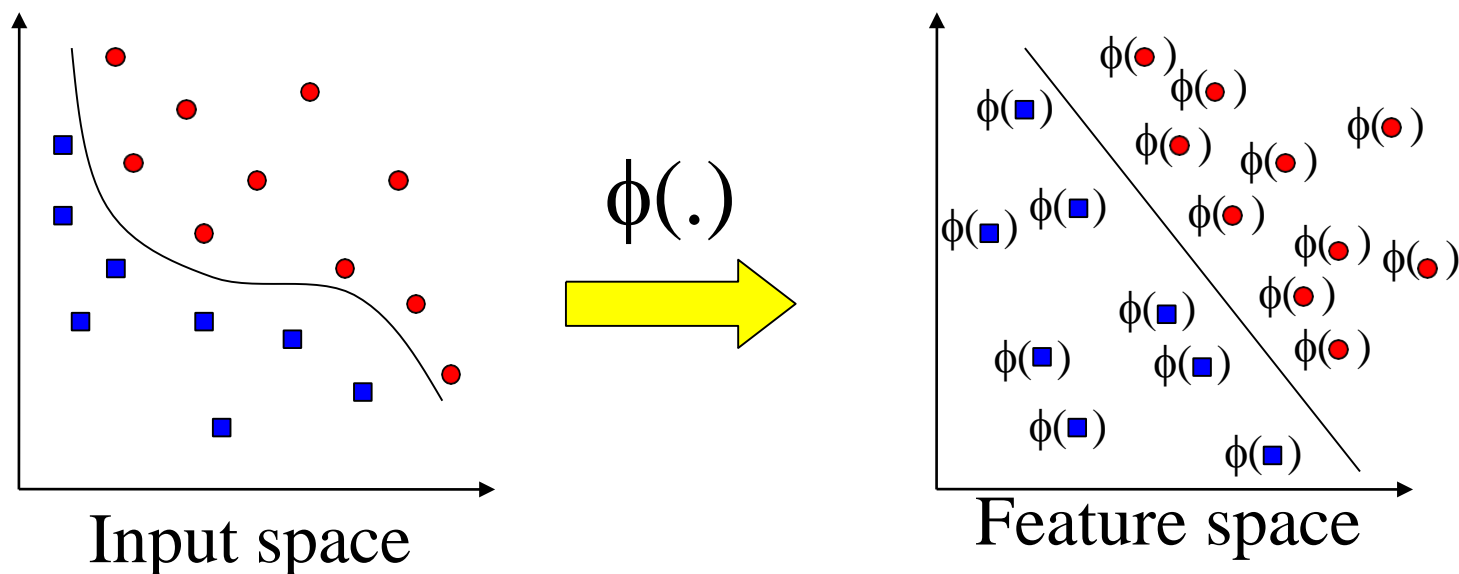
Transformation to Feature Space

Possible problem of the transformation

    High computation burden due to high-dimensionality and hard
    to get a good estimate

SVM solves these two issues simultaneously

    "Kernel tricks" for efficient computation

    Minimize $||\mathbf{w}||^2$ can lead to a "good" classifier

$\phi(.)$

Input space

Feature space

Examples of Kernel Functions

Polynomial kernel with degree d
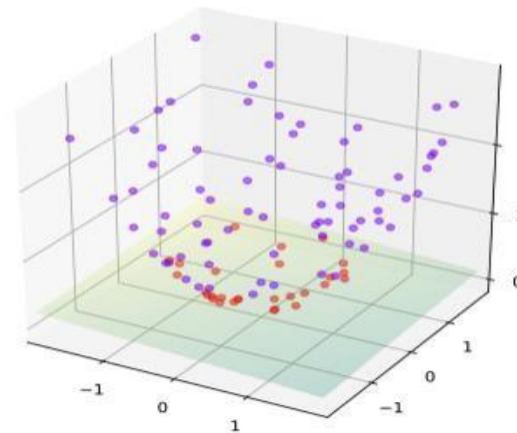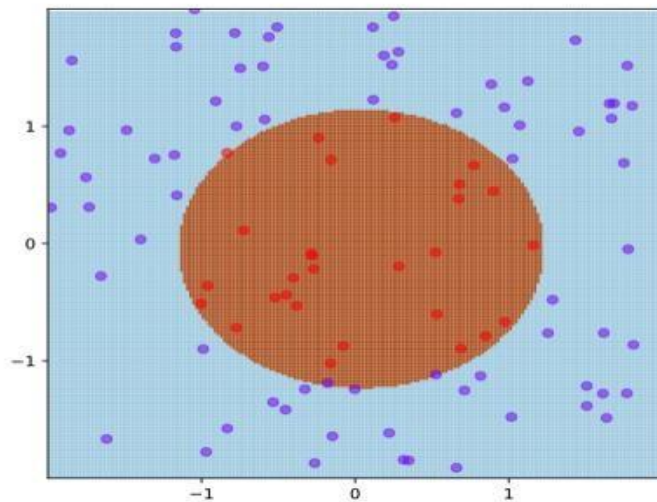$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + 1)^d$$

Radial basis function kernel with width σ
$$K(\mathbf{x}, \mathbf{y}) = \exp(-||\mathbf{x} - \mathbf{y}||^2/(2\sigma^2))$$

Research on different kernel functions in different applications is very active
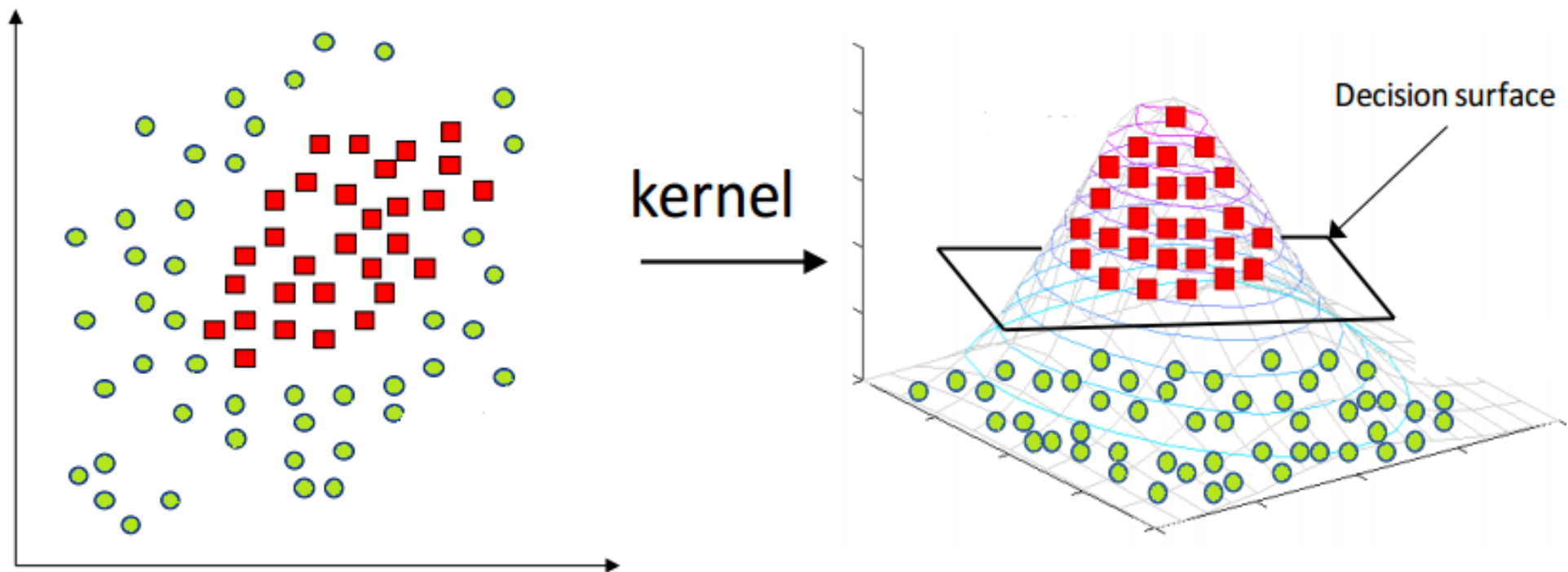
Kernels Tricks

In its simplest form, the kernel trick means transforming data into another dimension that has a clear dividing margin between classes of data.

SVM with kernel given by $\phi((a, b)) = (a, b, a^2 + b^2)$ and thus $K(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y} + \mathbf{x}^2 \mathbf{y}^2$. The training points are mapped to a 3-dimensional space where a separating hyperplane can be easily found.

kernel trick offers a more efficient and less expensive way to transform data into higher dimensions

Choosing the Kernel Function

Probably the most tricky part of using SVM.

Many principles have been proposed (diffusion kernel, Fisher kernel, string kernel, …)
There is even research to estimate the kernel matrix from available information

In practice, a low degree polynomial kernel or RBF kernel with a reasonable width is a good initial try

A list of SVM implementation can be found at http://www.kernel-machines.org/software.html