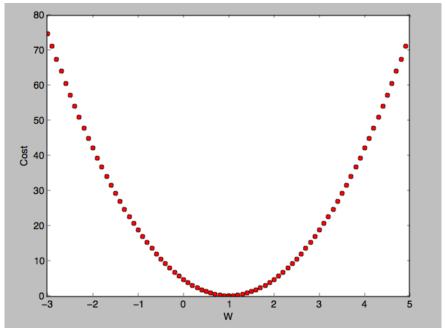
Logistic (regression) classification

Regression

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

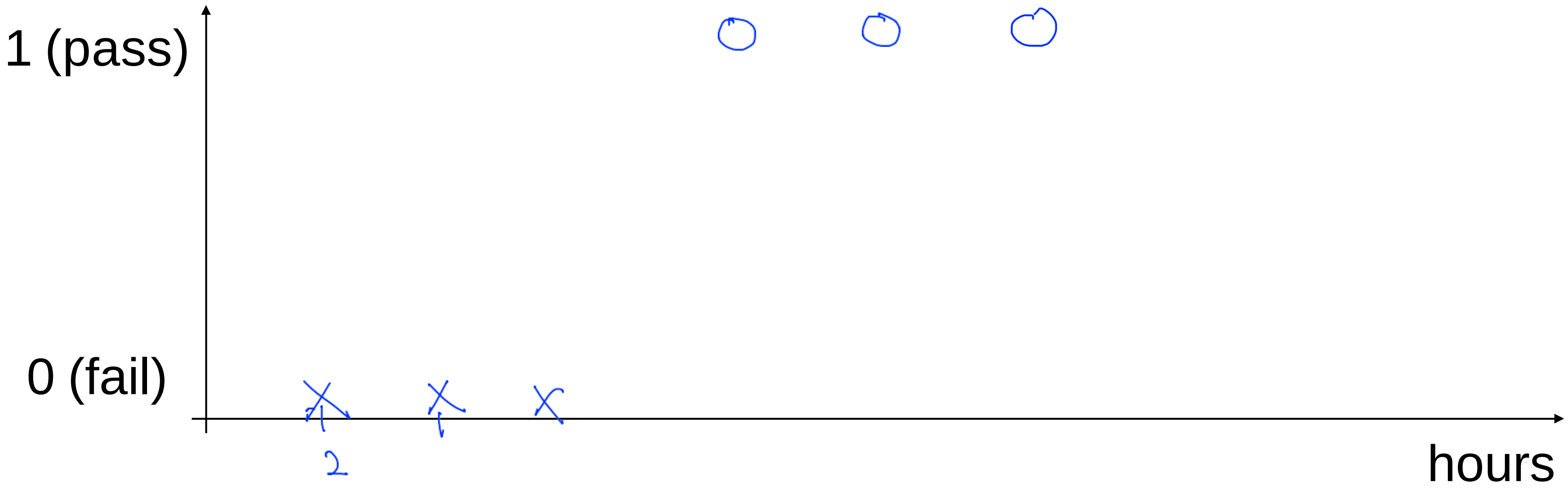
- Hypothesis: $H(X) = WX$
- Cost: $cost(W) = \frac{1}{m}\sum(WX - y)^2$



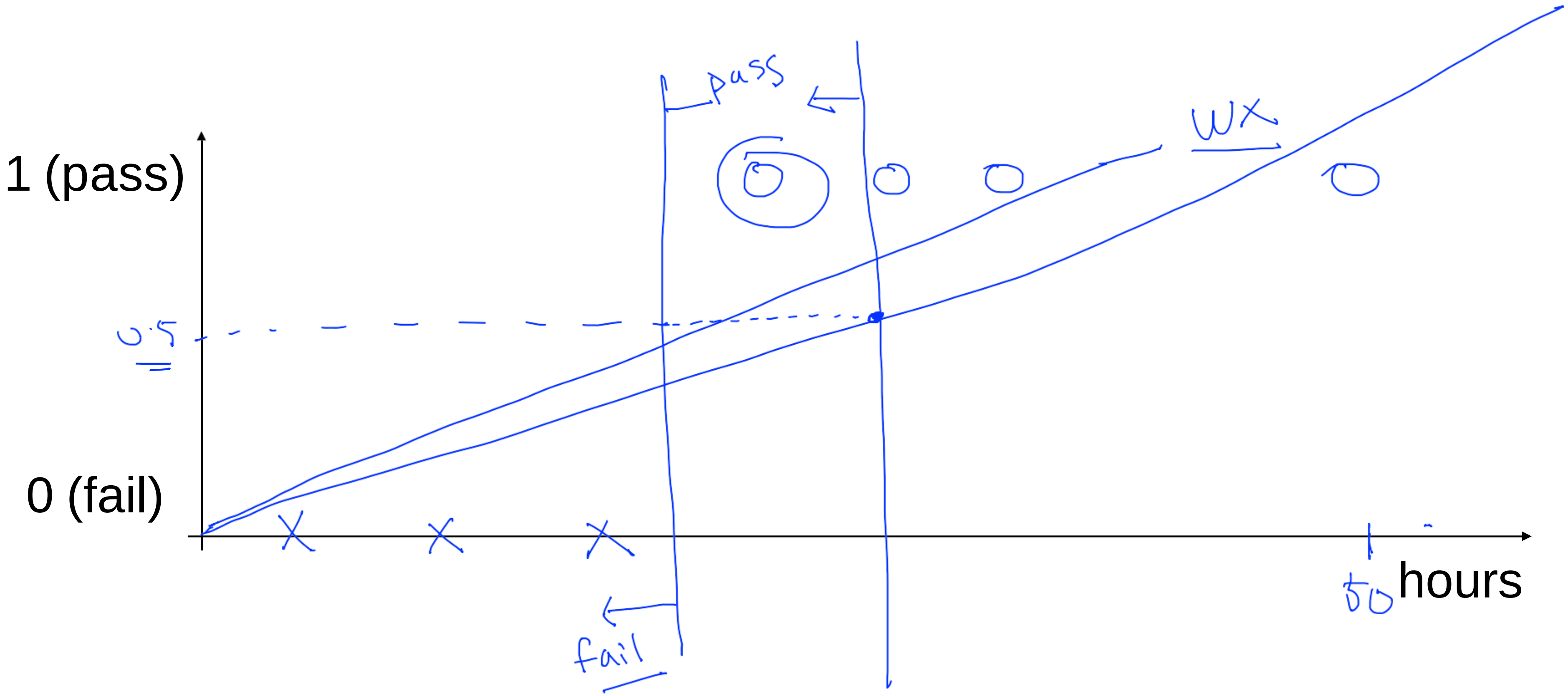
- Gradient decent:
$$W := W - \alpha \frac{\partial}{\partial W} cost(W)$$

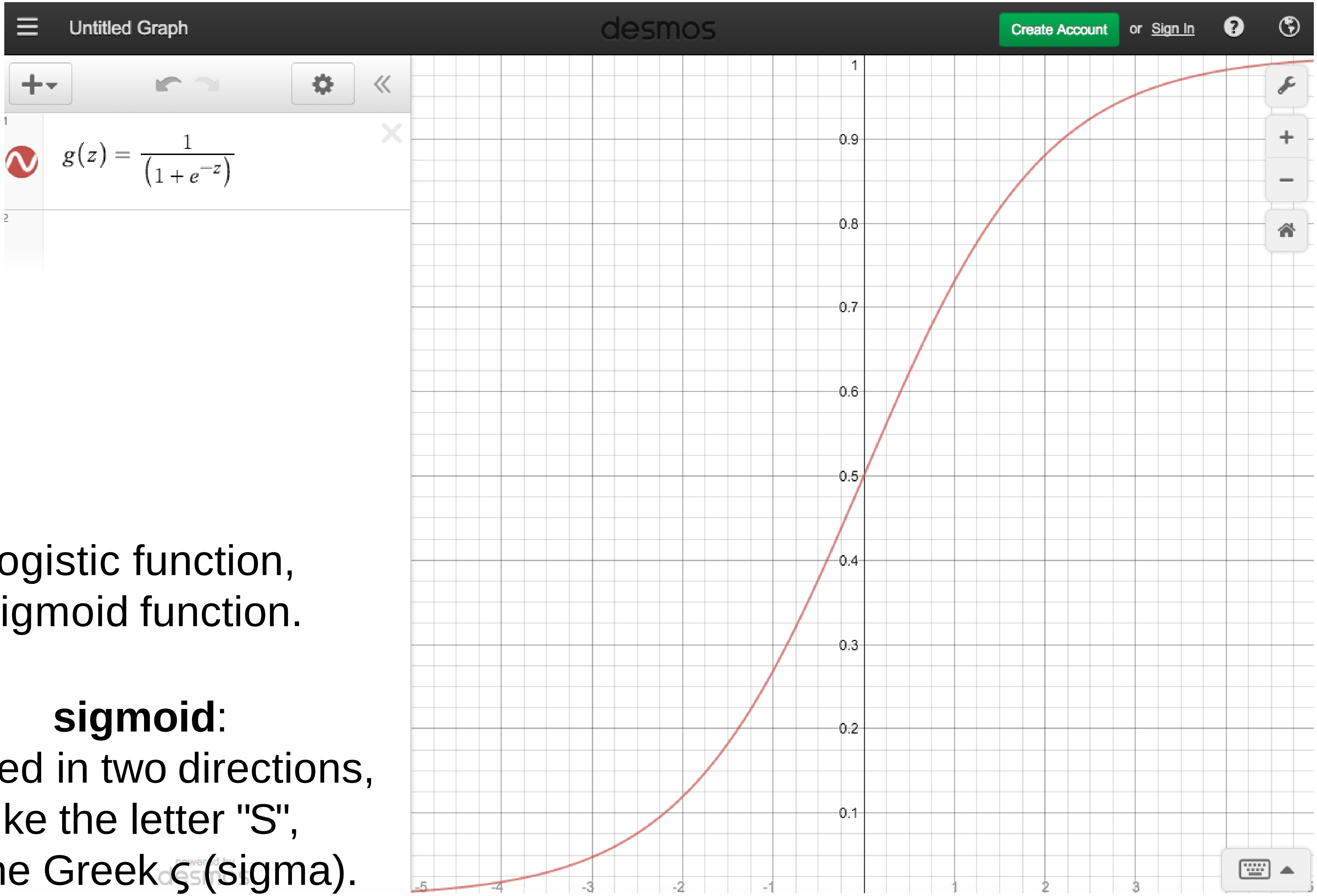
http://www.cse.iitk.ac.in/users/se367/10/presentation_local/Binary%20Classification.html

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



Linear Regression?





logistic function,
sigmoid function.

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

Logistic Hypothesis

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

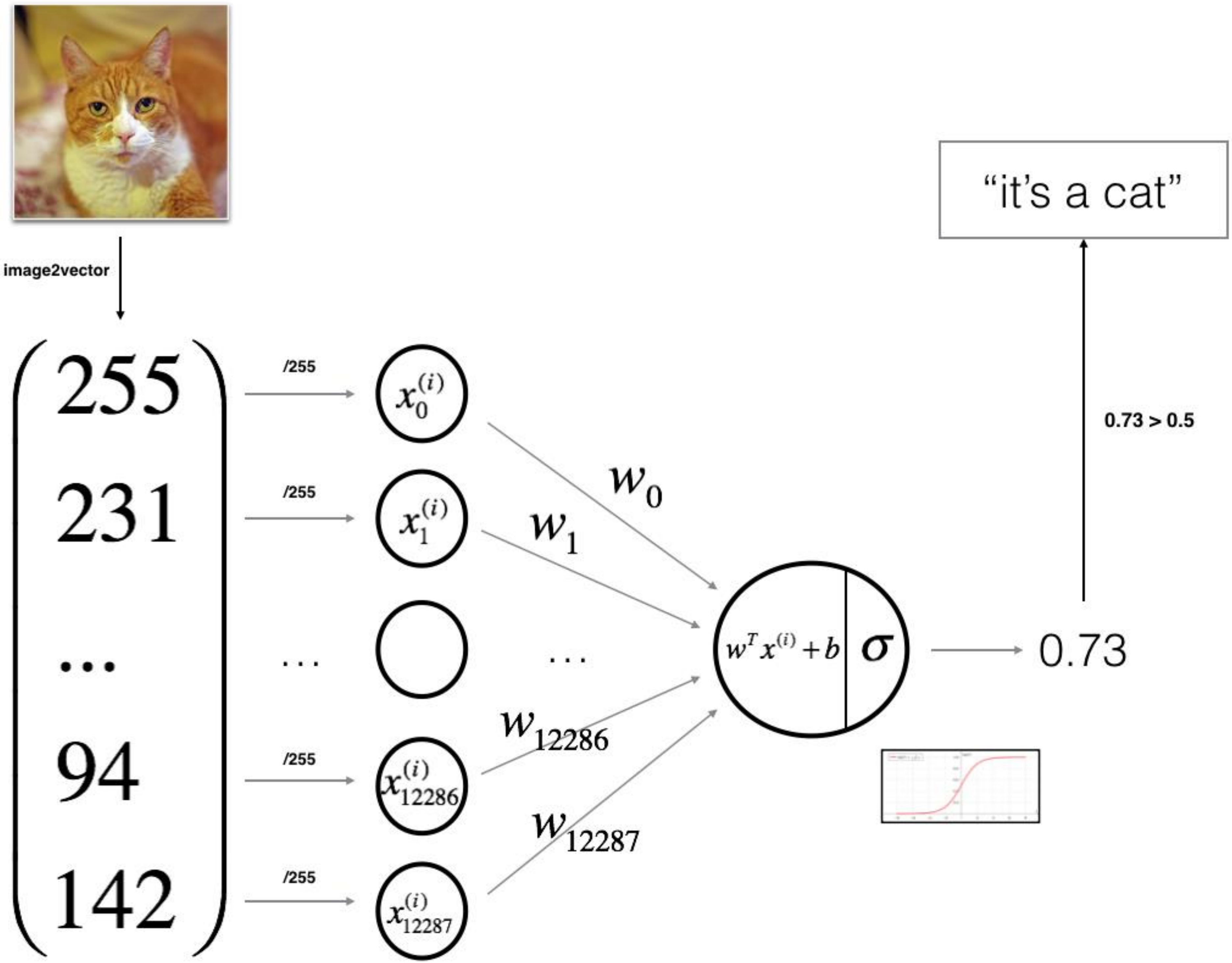
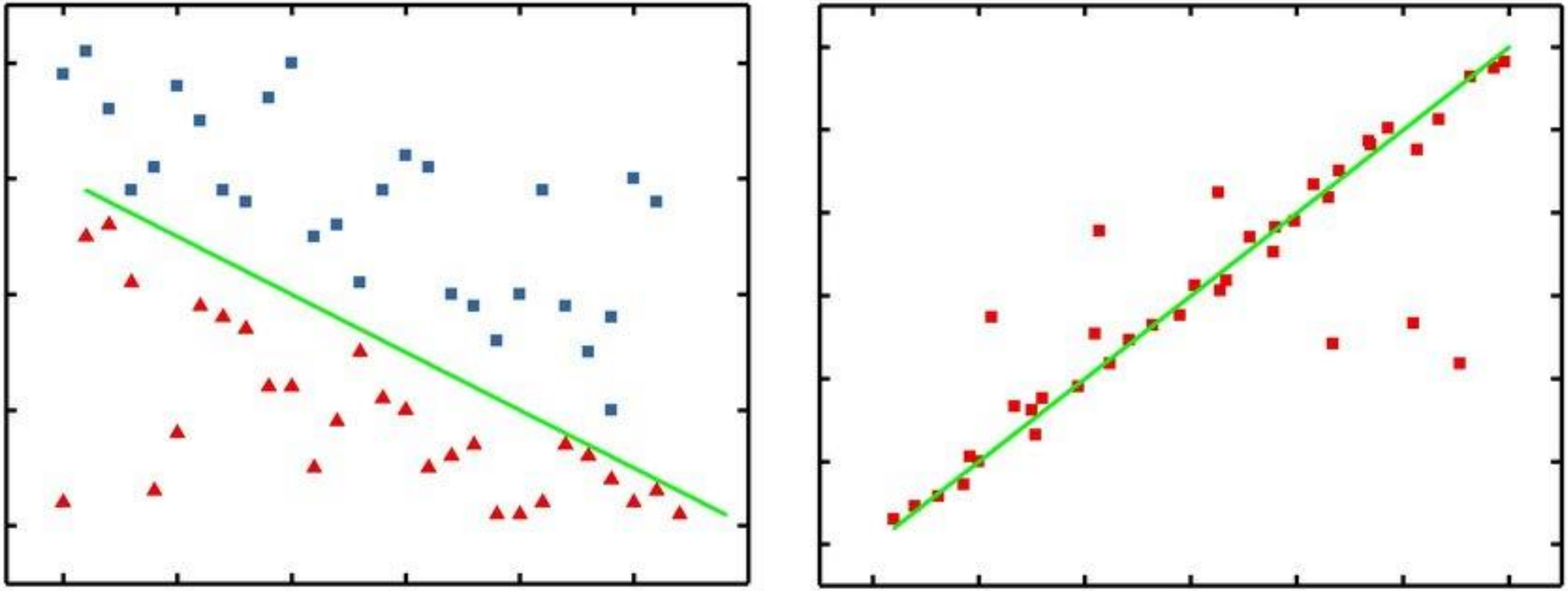


image2vector
255
231
...
94
142
/255
/255
/255
/255
$x_0^{(i)}$
$x_1^{(i)}$
$x_{12286}^{(i)}$
$x_{12287}^{(i)}$
w_0
w_1
w_{12286}
w_{12287}
$w^T x^{(i)} + b$
σ
0.73
0.73 > 0.5
"it's a cat"

Logistic Regression

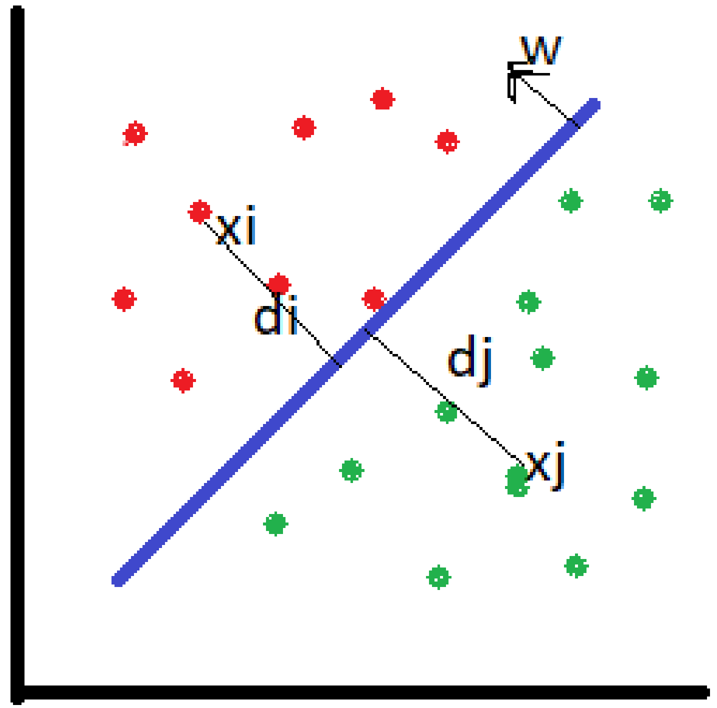
$$H(x) = Wx + b$$



(a) Logistic Regression (b) Linear Regression

Logistic Regression – Geometric Interpretation

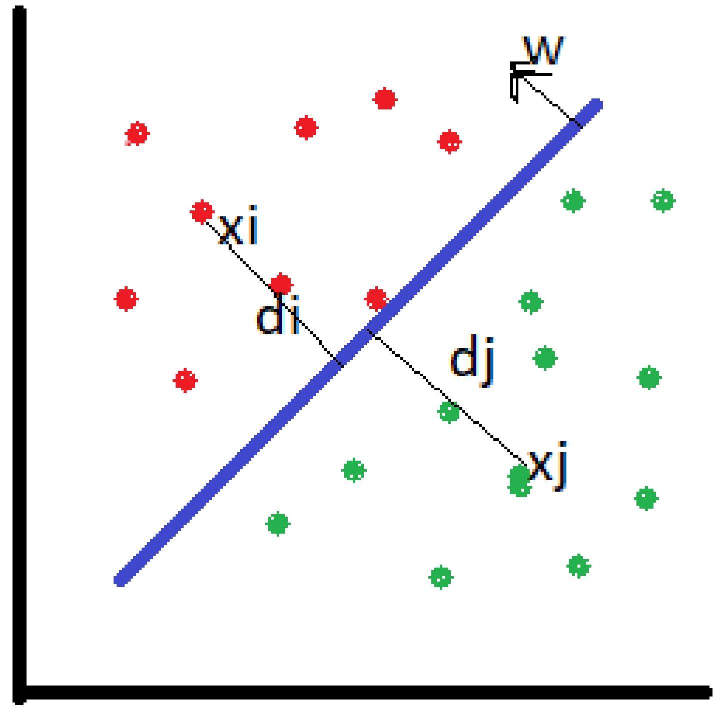
- Distance of any point xi from the plane(π) is
- di=wTxi/||w|| ; w is the normal to the plane
- If w is a unit vector, this implies ||w||=1
- di= wTxi
- Red points represent y=+1
- Green points represent y=-1



https://medium.com/datasciencechannel/logistic-regression-geometric-interpretation-37f08a315a0b

Logistic Regression – Geometric Interpretation

Case1: when yi=+1 and wTxi>0
Case 2: when yi=-1 and wTxi<0
Case 3: yi=+1
Case 4: y=-1
From Case 3 and 4, it implies logistic regression is incorrectly classifying the negative point

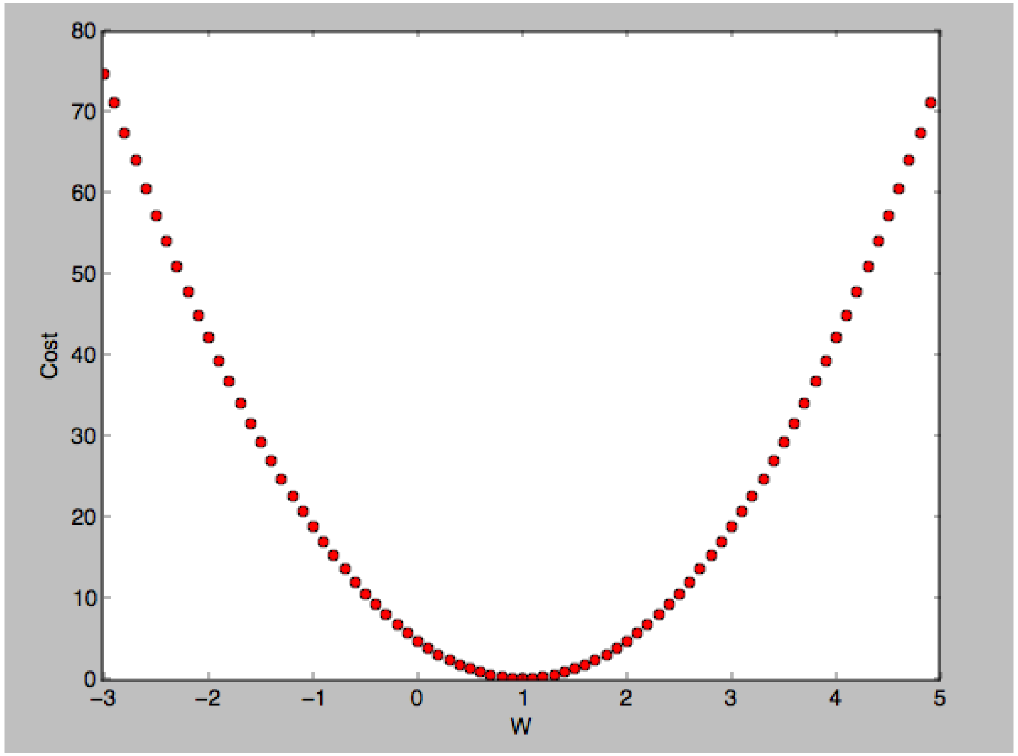


Max ∑ yi wt xi from i=1 to n.
The equation above is the **Optimization problem**

https://medium.com/datasciencechannel/logistic-regression-geometric-interpretation-37f08a315a0b

Cost

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

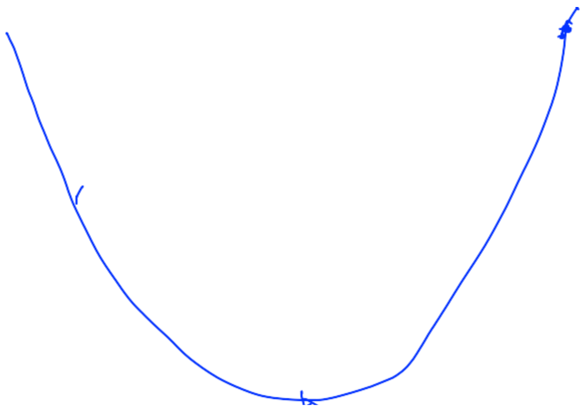


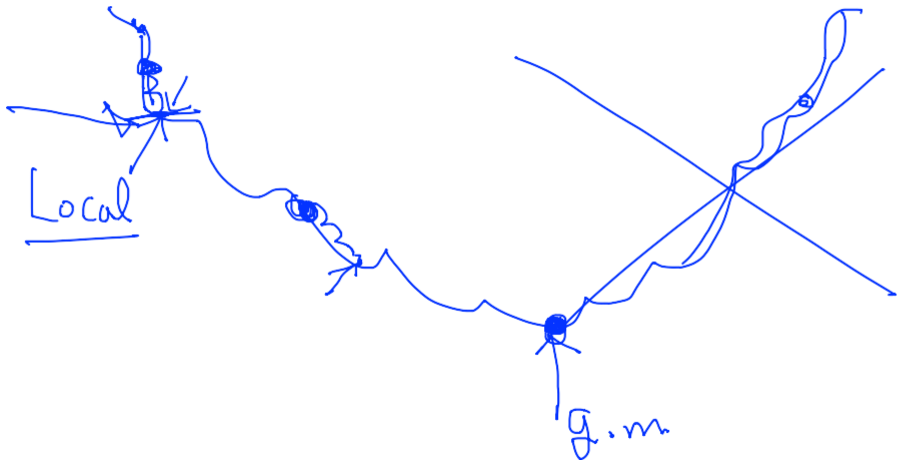
Cost function

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

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

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

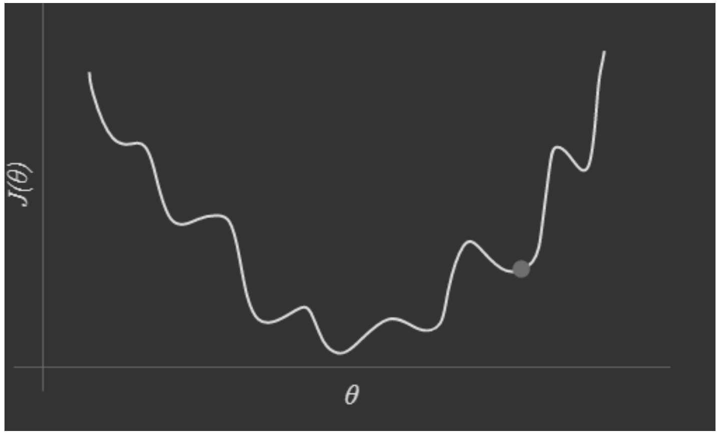




Cost

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

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



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

New cost function for logistic

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

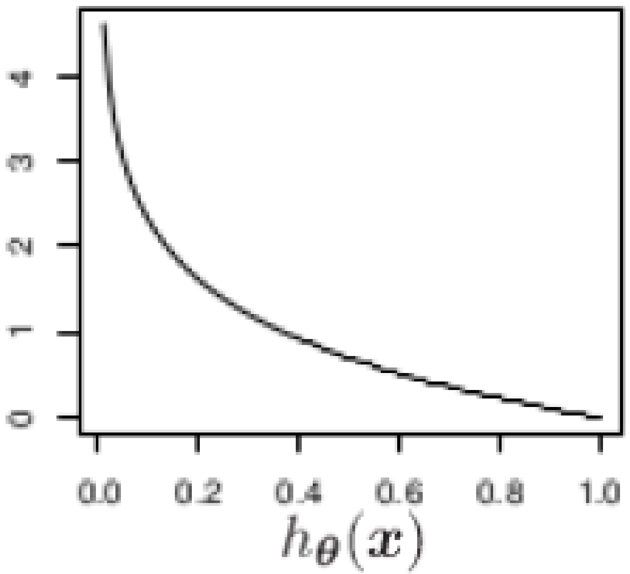
$$c(H(x), y) = \begin{cases} -log(H(x)) & : y = 1 \\ -log(1 - H(x)) & : y = 0 \end{cases}$$

Logistic cross entropy cost

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

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

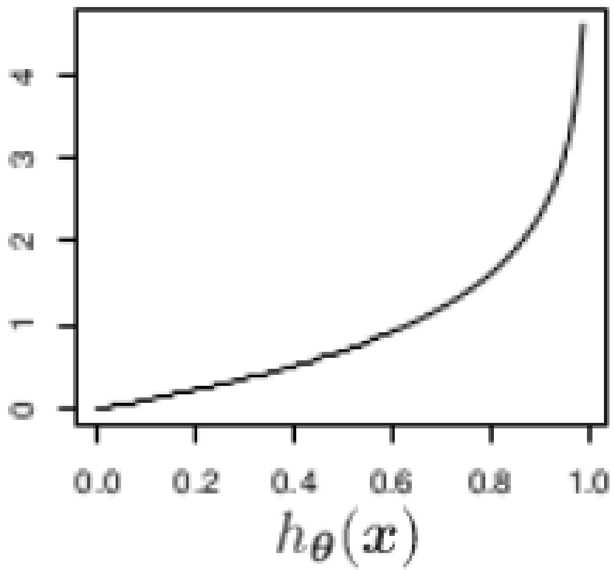
if $\ y = 1$



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

if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) \to 0$
then cost $\to \infty$
predicted
$\text{prob}(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta}) = 0$
but $y = 1$

if $\ y = 0$



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

if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) \to 1$
then cost $\to \infty$
predicted
$\text{prob}(y = 0 \mid \boldsymbol{x}; \boldsymbol{\theta}) = 0$
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)
```

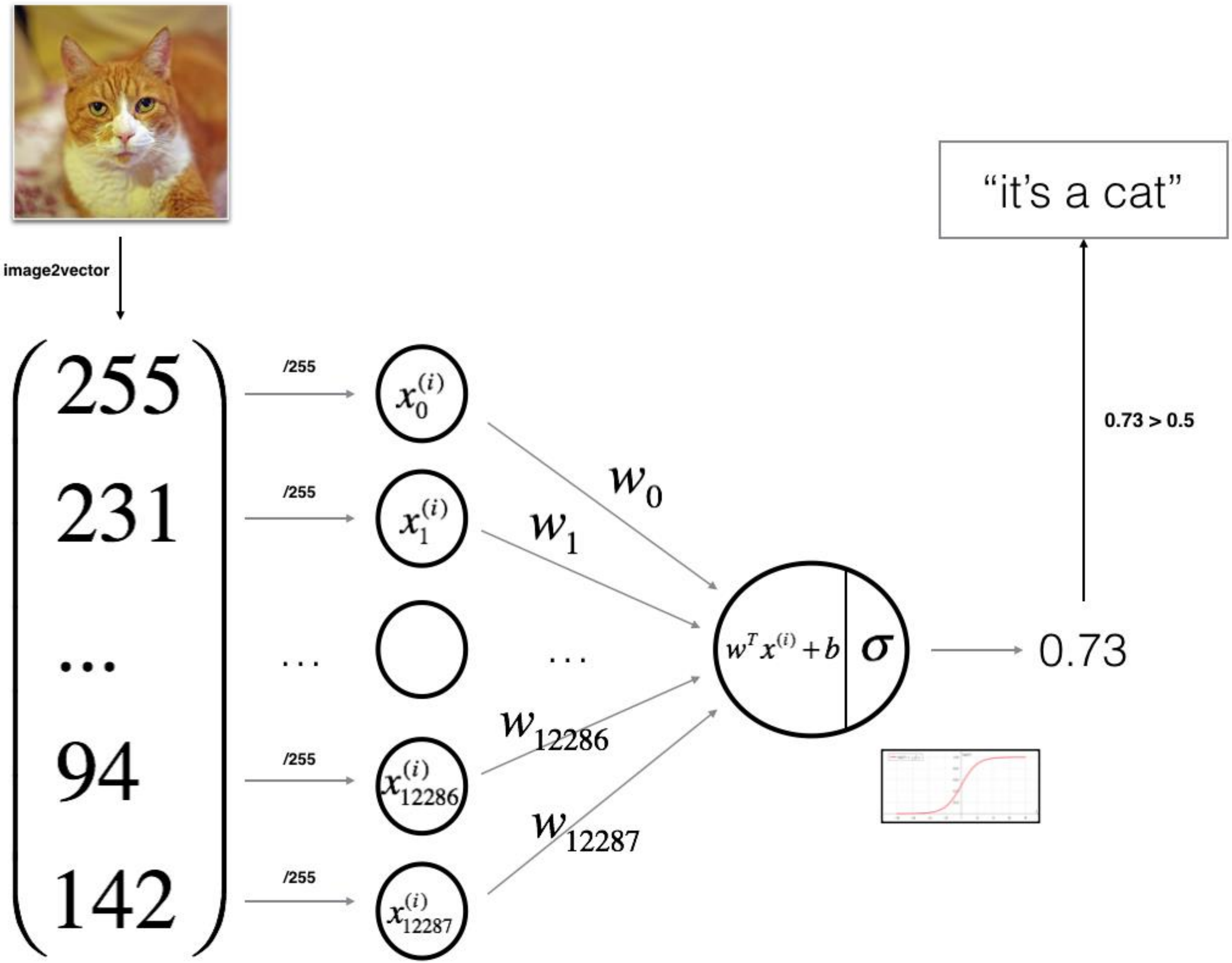


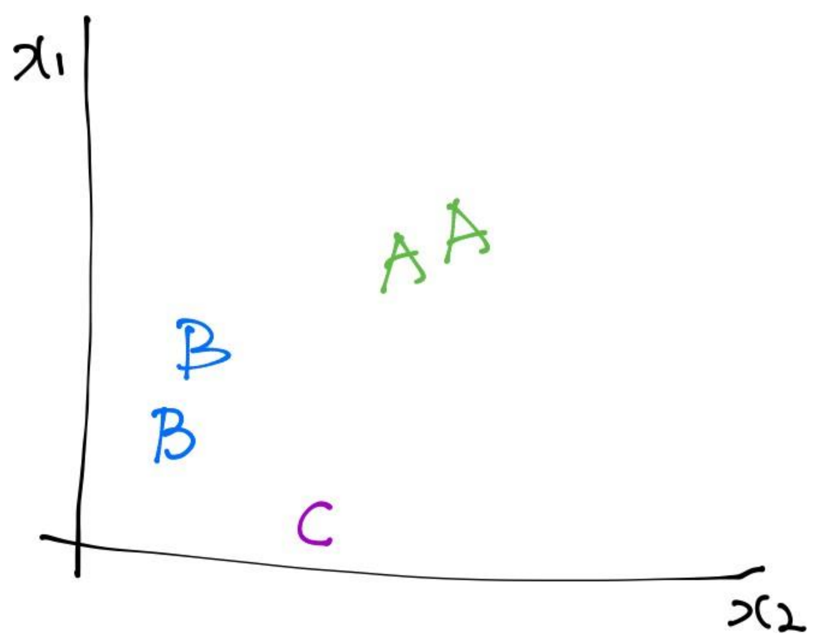
image2vector
255
231
...
94
142
/255
/255
/255
/255
$x_0^{(i)}$
$x_1^{(i)}$
$x_{12286}^{(i)}$
$x_{12287}^{(i)}$
w_0
w_1
w_{12286}
w_{12287}
$w^T x^{(i)} + b$
σ
0.73
0.73 > 0.5
"it's a cat"

Classification

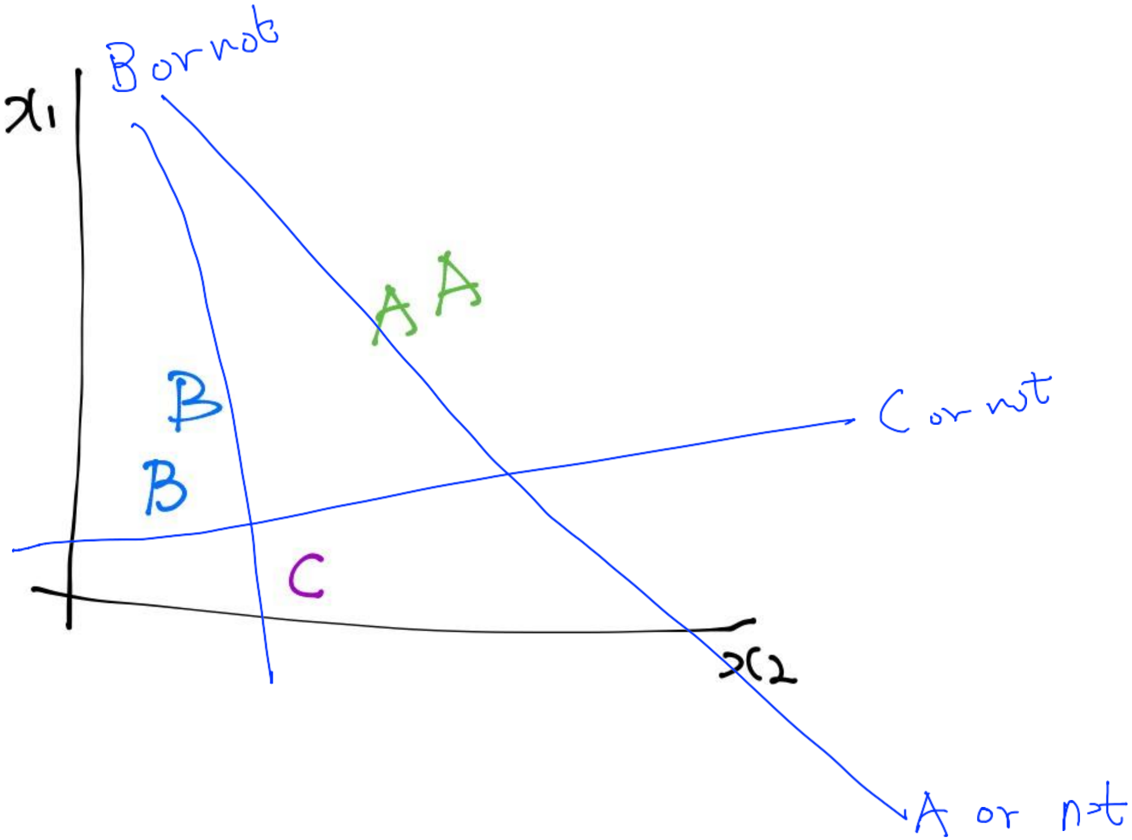
- Spam Detection: Spam or Ham
- Facebook feed: show or hide
- Credit Card Fraudulent Transaction detection: legitimate/fraud

Multinomial classification

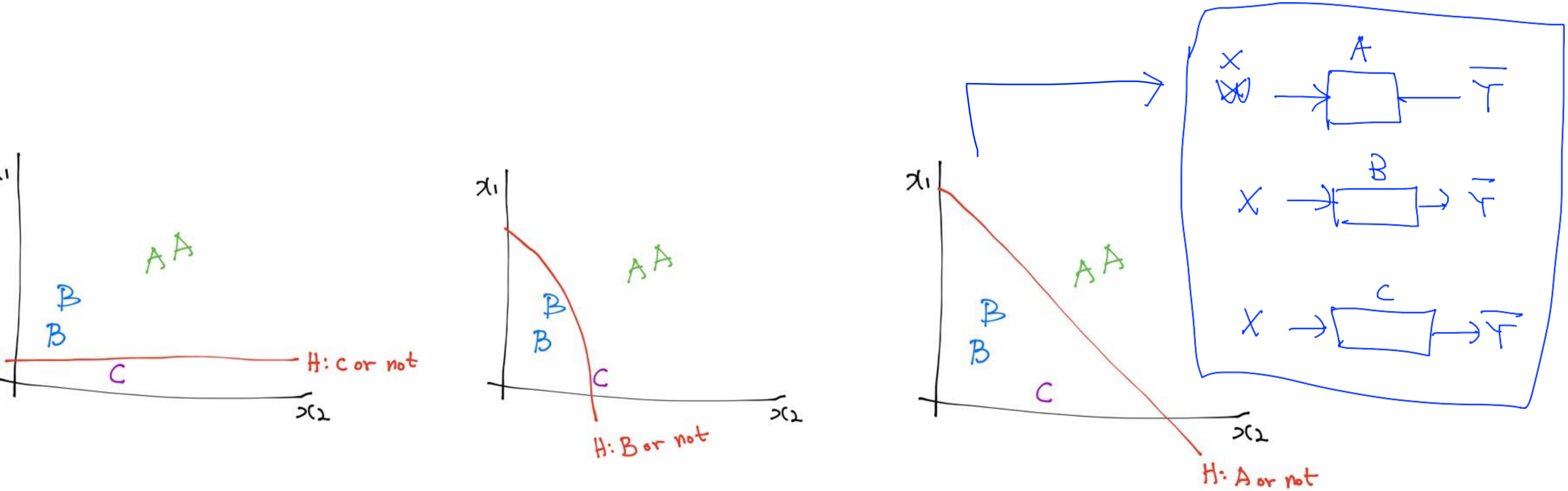
x1 (hours)	x2 (attendance)	y (grade)
10	5	A
9	5	A
3	2	B
2	4	B
11	1	C



Multinomial classification



Multinomial classification

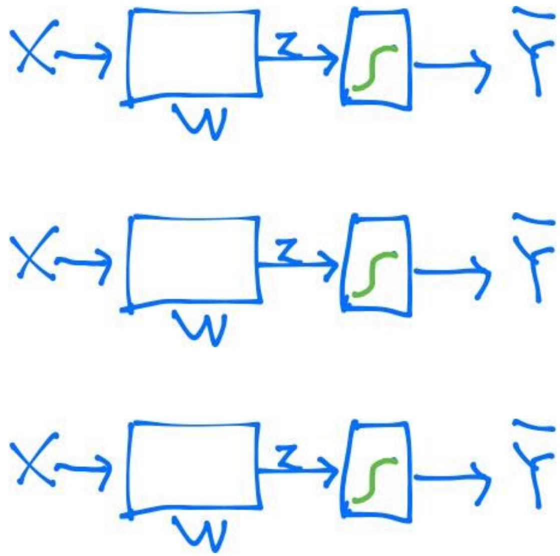


Multinomial classification

$$\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} w_1x_1 + w_2x_2 + w_3x_3 \end{bmatrix}$$

$$\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} w_1x_1 + w_2x_2 + w_3x_3 \end{bmatrix}$$

$$\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} w_1x_1 + w_2x_2 + w_3x_3 \end{bmatrix}$$



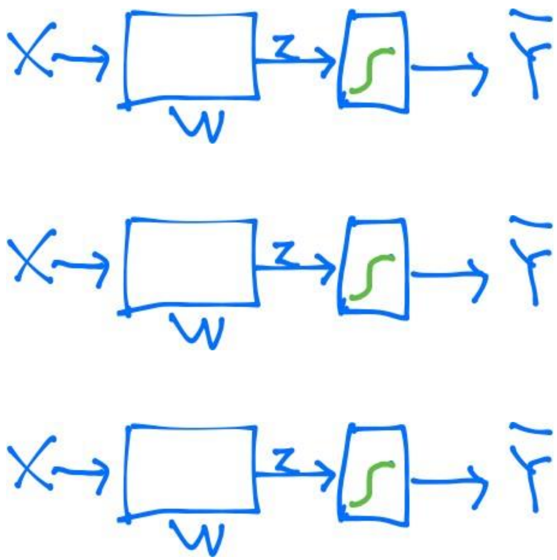
Multinomial classification

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

A

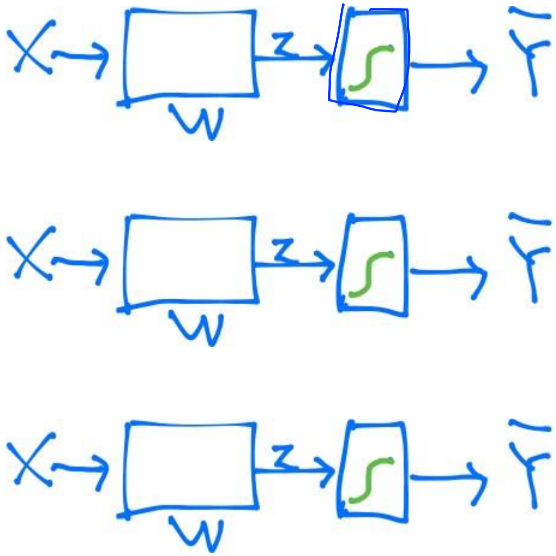
B

C



Where is sigmoid?

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



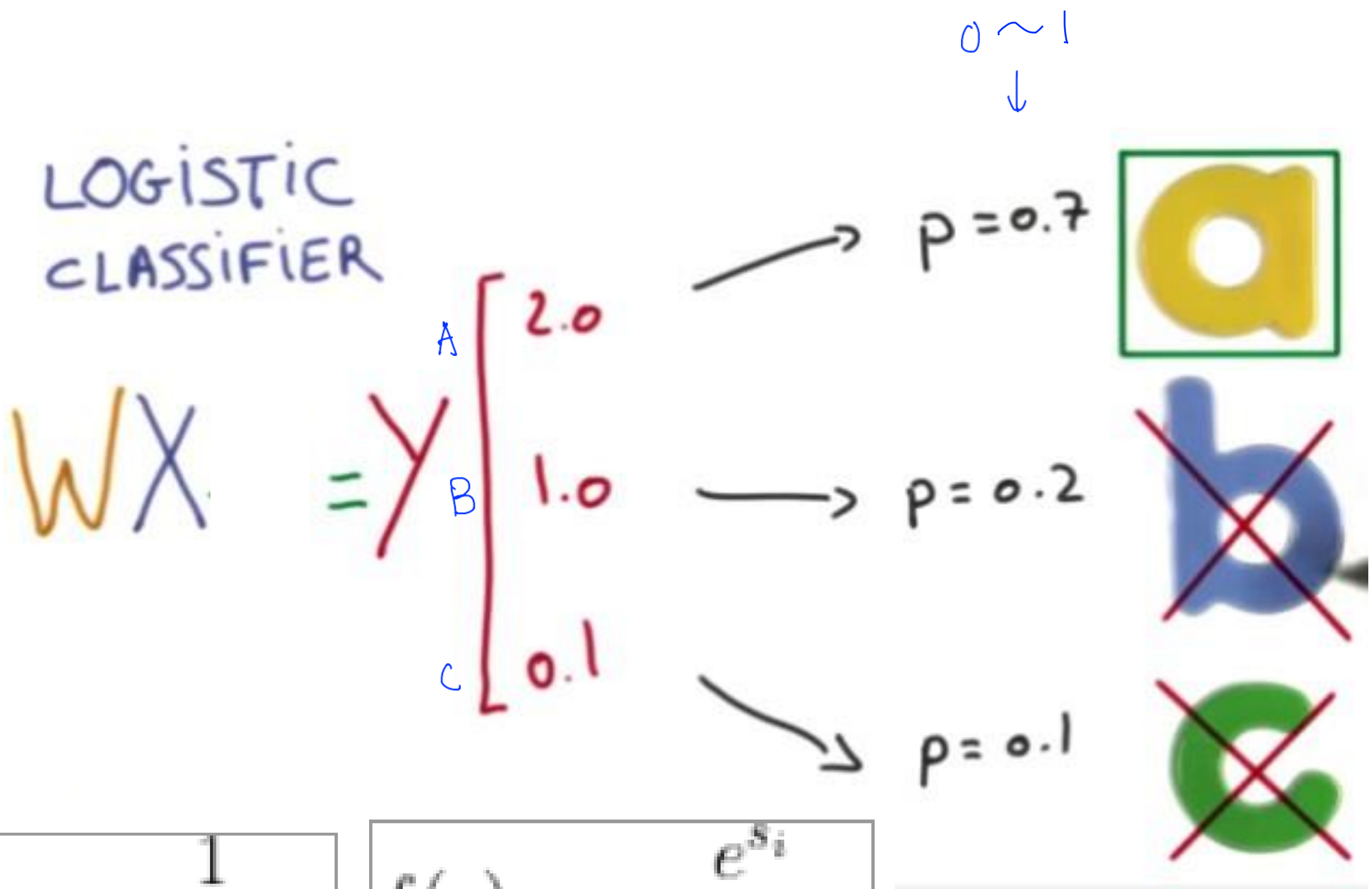
Where is sigmoid?

$$\begin{bmatrix} w_{A1} & w_{A2} & w_{A3} \\ w_{B1} & w_{B2} & w_{B3} \\ w_{C1} & w_{C2} & w_{C3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} w_{A1}x_1 + w_{A2}x_2 + w_{A3}x_3 \\ w_{B1}x_1 + w_{B2}x_2 + w_{B3}x_3 \\ w_{C1}x_1 + w_{C2}x_2 + w_{C3}x_3 \end{bmatrix} = \begin{bmatrix} \bar{y}_A \\ \bar{y}_B \\ \bar{y}_C \end{bmatrix} \begin{bmatrix} 2.0 \\ 1.0 \\ 0.1 \end{bmatrix}$$

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"

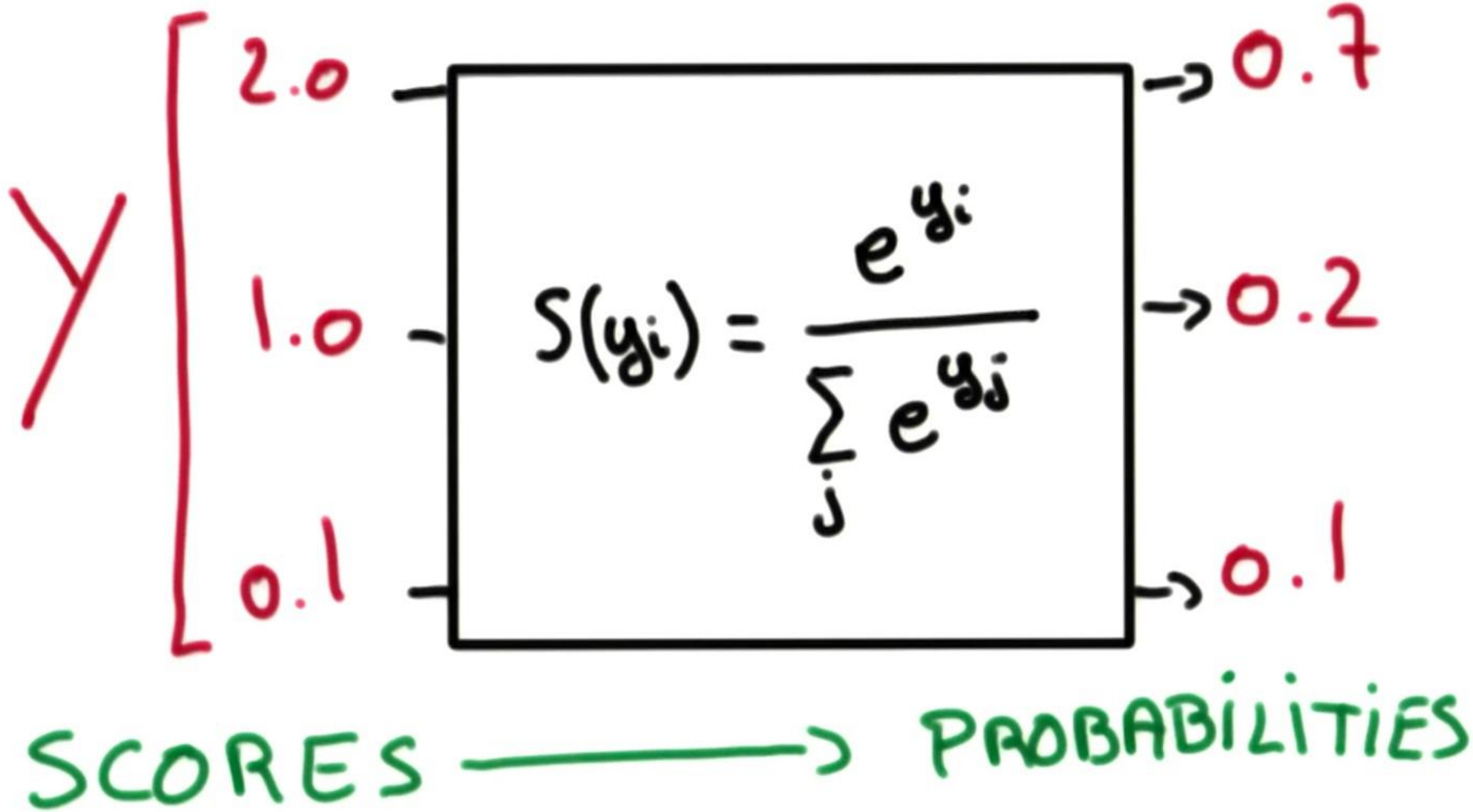
Sigmoid/Softmax?

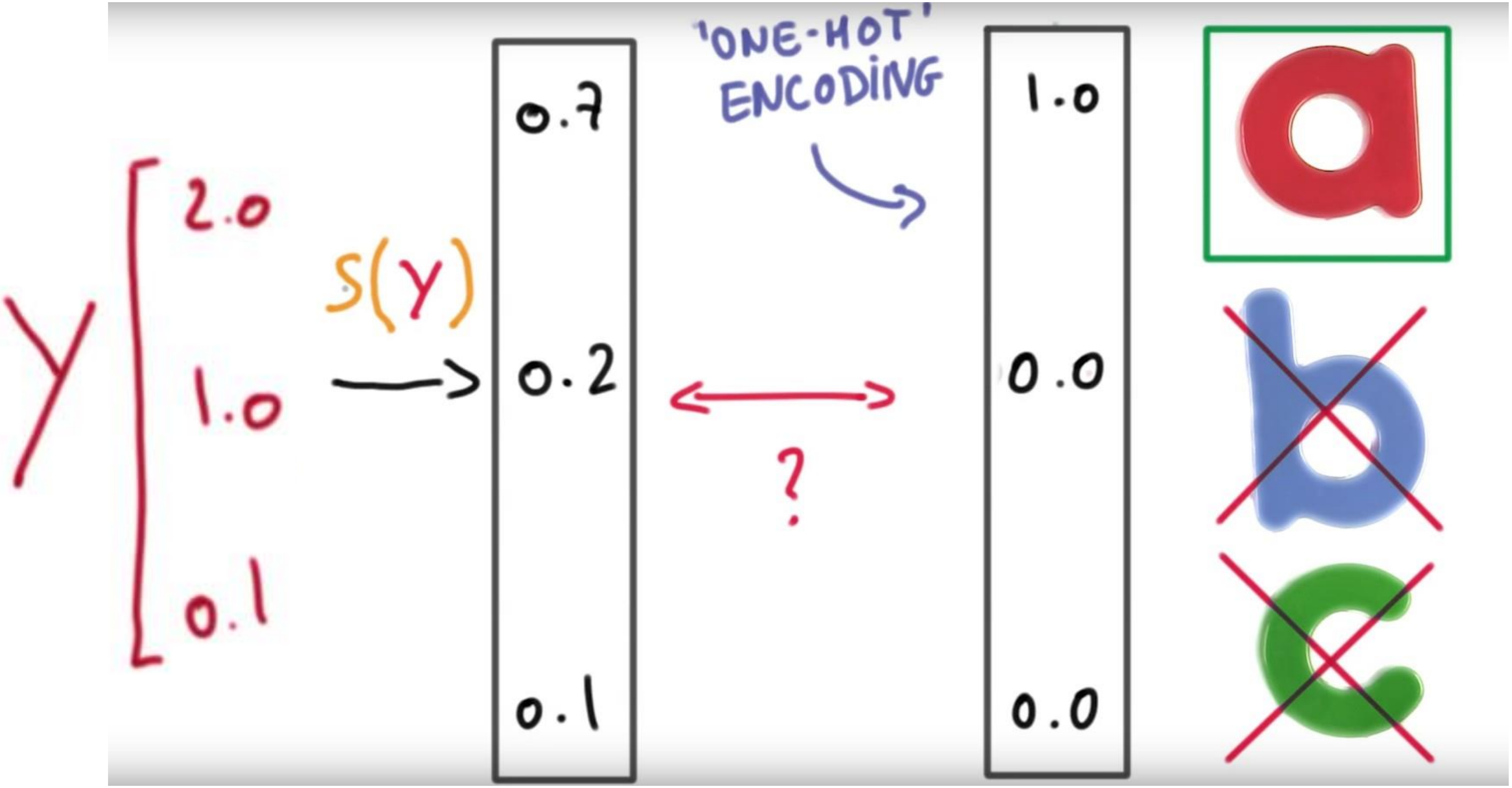


$$f(s_i) = \frac{1}{1 + e^{-s_i}}$$

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

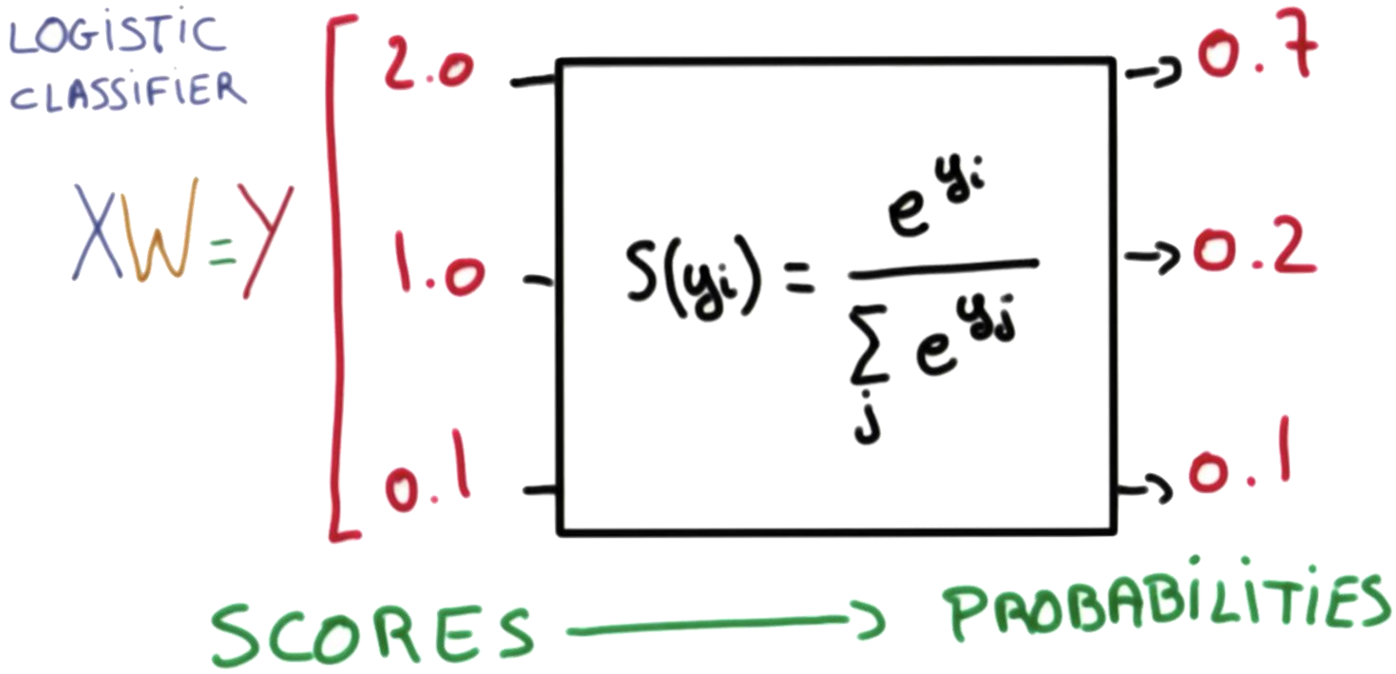
SOFTMAX




'ONE-HOT'
ENCODING
Y
2.0
1.0
0.1
S(Y)
0.7
0.2
0.1
?
1.0
0.0
0.0

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

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



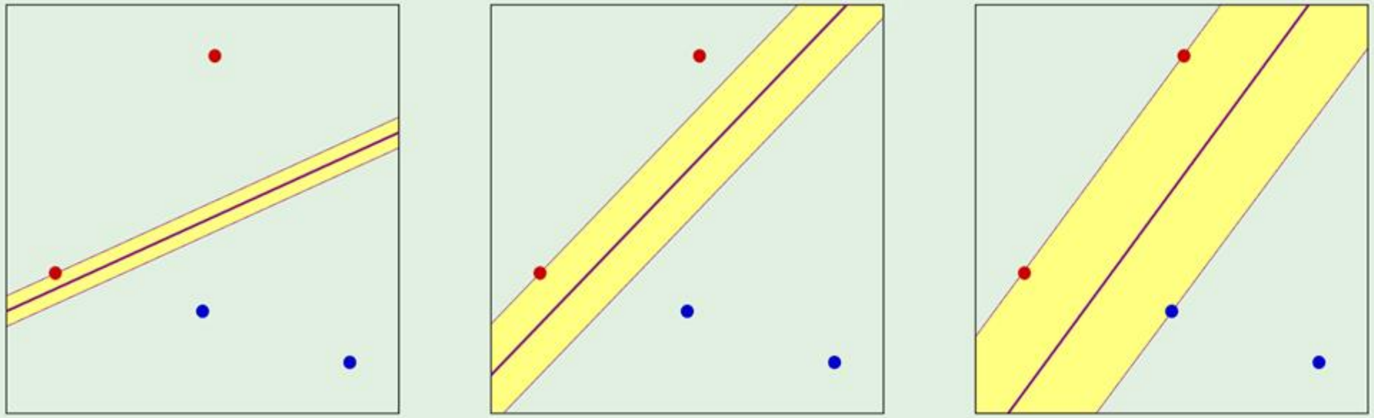
SVM
Support Vector Machines

Better linear separation

Linearly separable data

Different separating lines

Which is best?



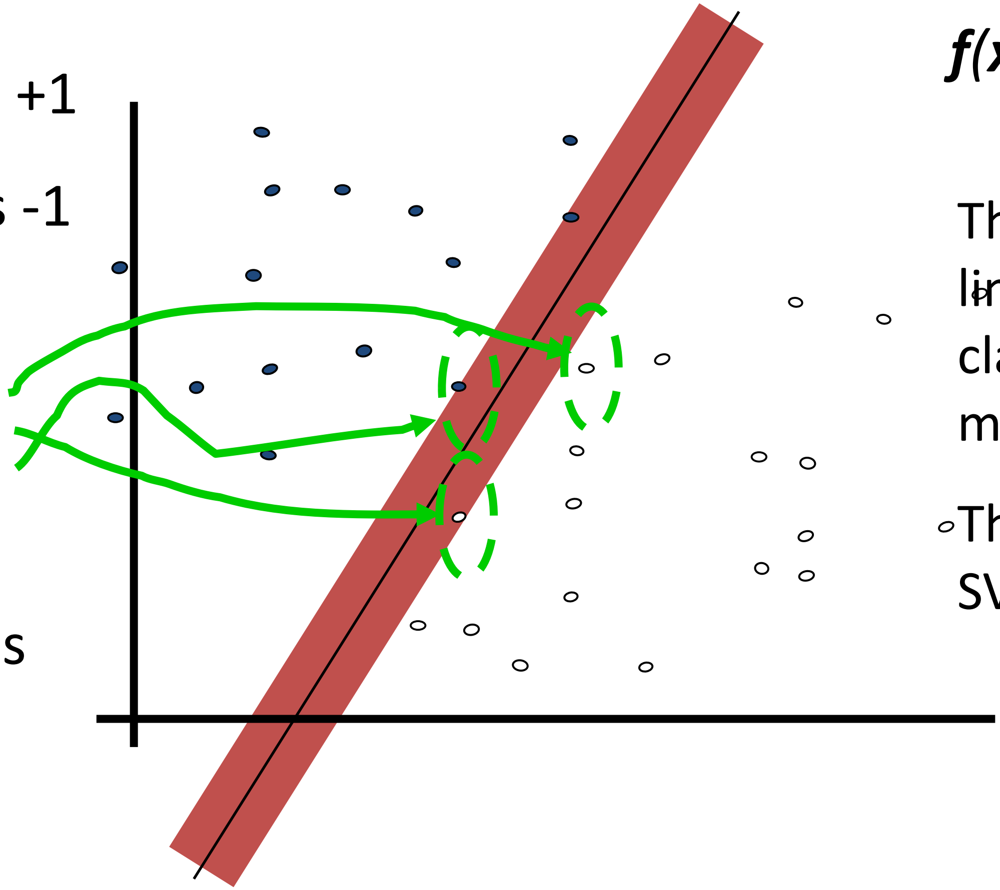
Two questions:

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

$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

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

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

Computing distance

The distance between $\mathbf{x}_n$ and the plane $\mathbf{w}^{\mathsf{T}}\mathbf{x} + b = 0$ where $|\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b| = 1$

The vector $\mathbf{w}$ is $\perp$ to the plane in the $\mathcal{X}$ space:

Take $\mathbf{x}'$ and $\mathbf{x}''$ on the plane

$\mathbf{w}^{\mathsf{T}}\mathbf{x}' + b = 0$ and $\mathbf{w}^{\mathsf{T}}\mathbf{x}'' + b = 0$

$\Longrightarrow \quad \mathbf{w}^{\mathsf{T}}(\mathbf{x}' - \mathbf{x}'') = 0$

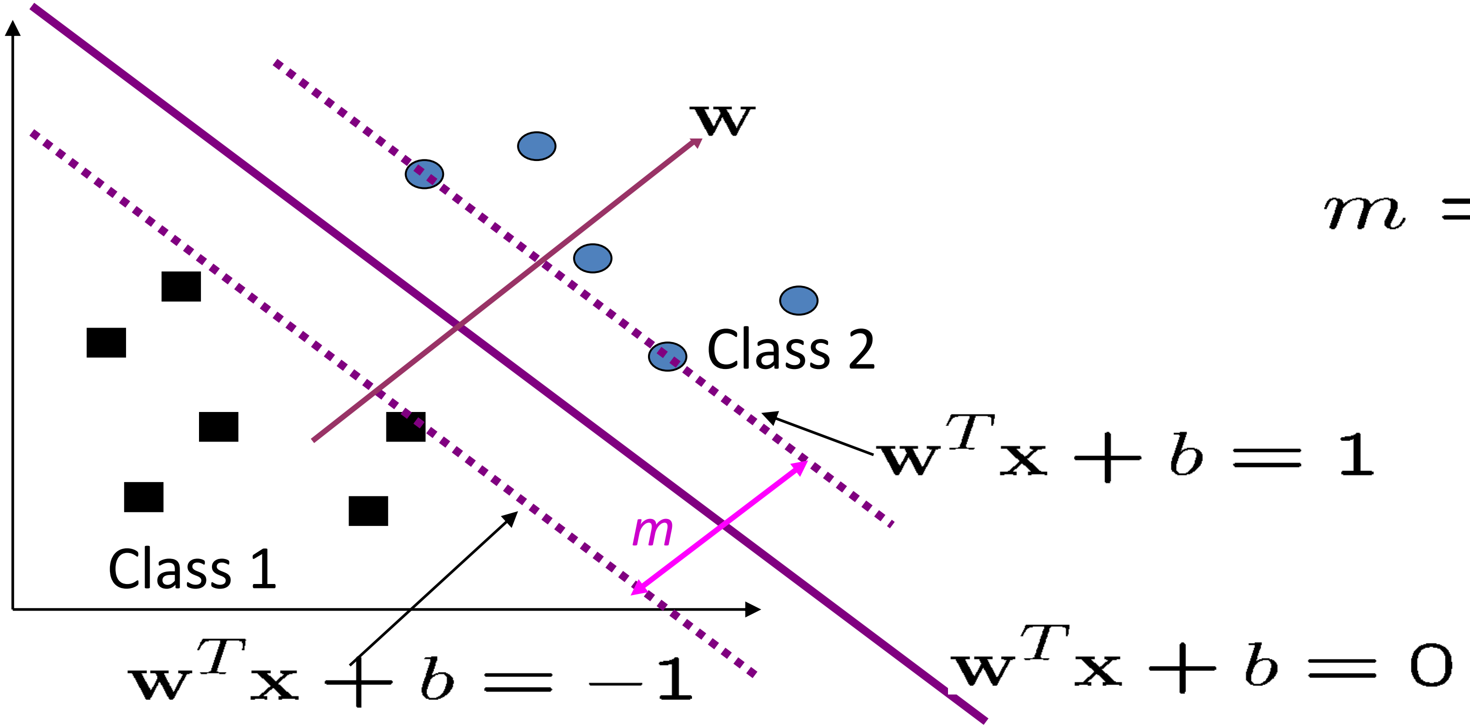
$$\text{distance} = \frac{1}{\|\mathbf{w}\|}$$

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/||**w**||



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

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.

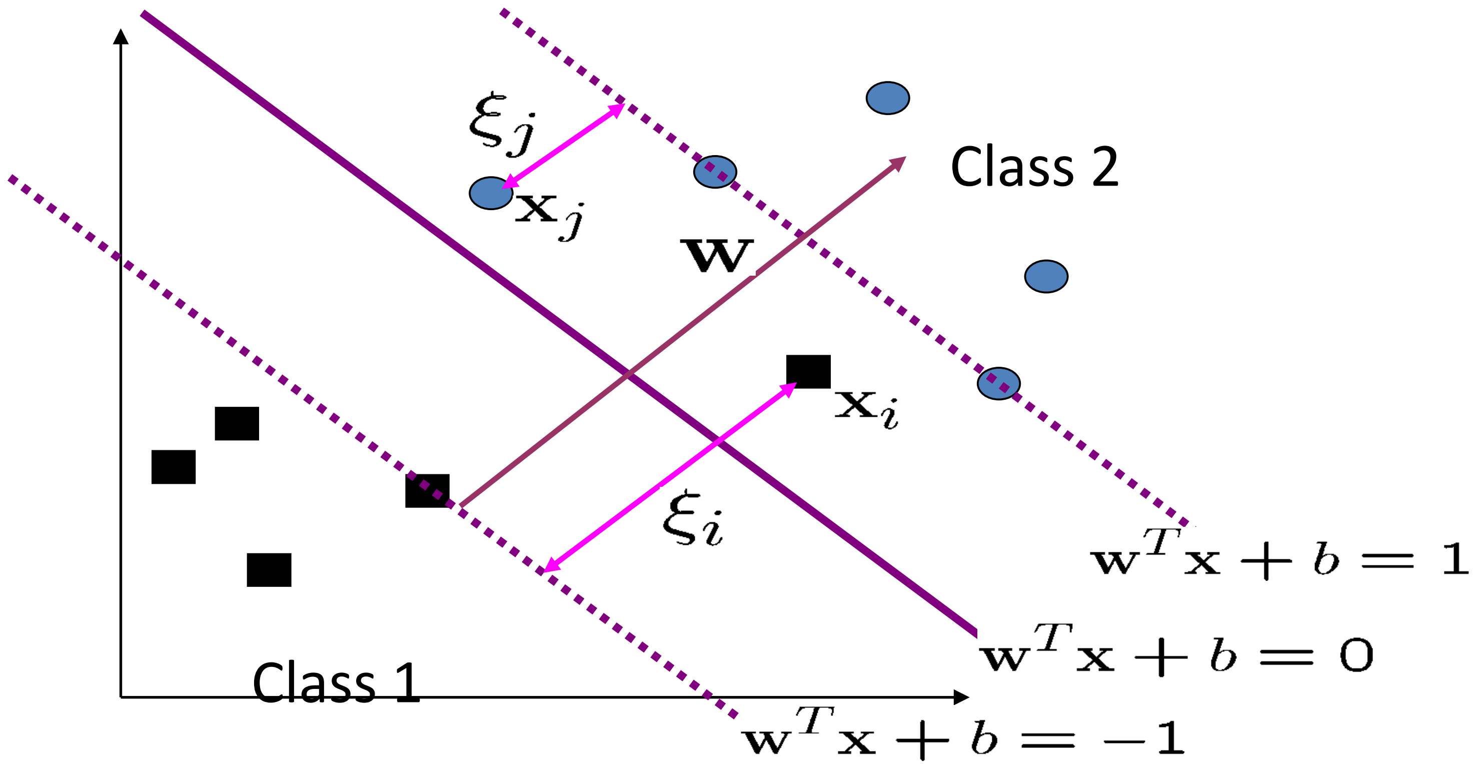
The decision boundary can be found by solving the following constrained optimization problem

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2$$
$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \qquad \forall i$$

Allowing errors in our solutions

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

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 ξ_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

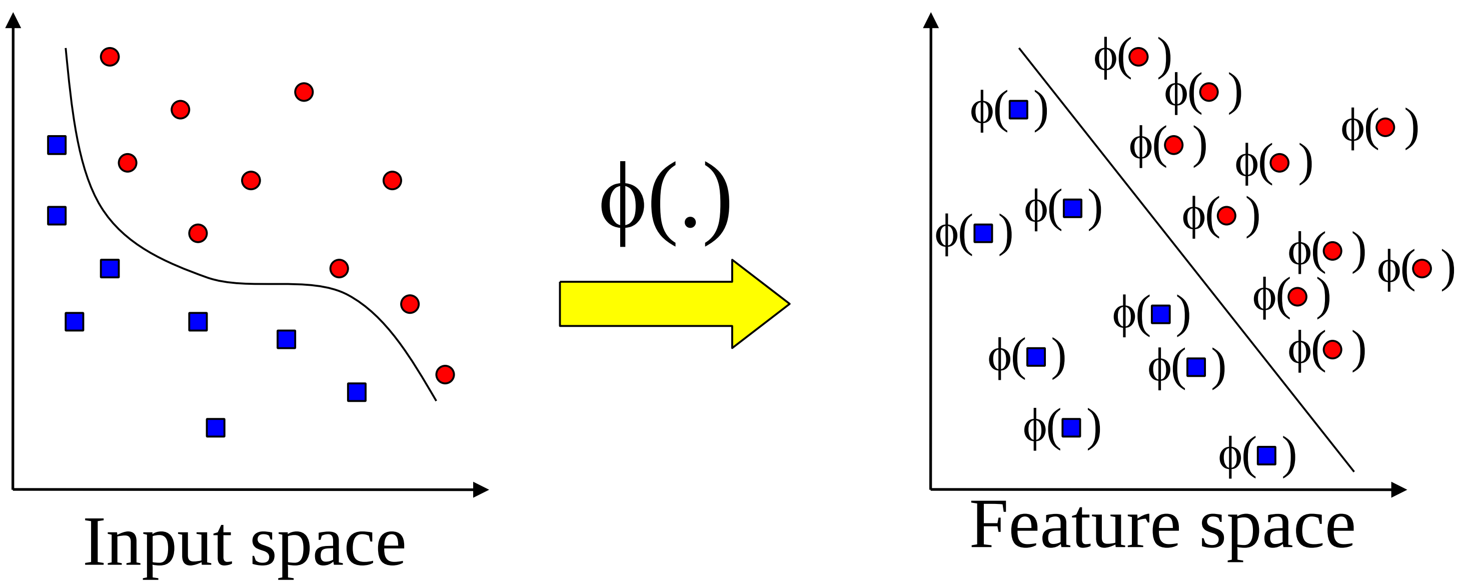
How to generalize it to become nonlinear?

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



Examples of Kernel Functions

Polynomial kernel with degree *d*

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

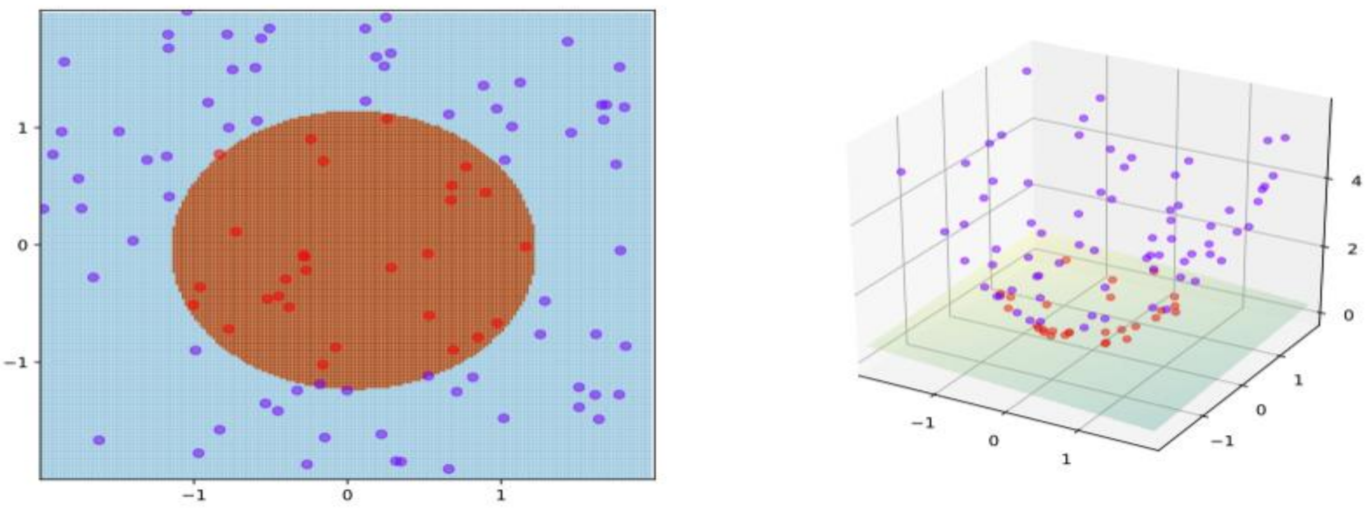
Radial basis function kernel with width σ

$$K(\mathbf{x}, \mathbf{y}) = \exp(-||\mathbf{x} - \mathbf{y}||^2 / (2\sigma^2))$$

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.

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

Resources

http://www.kernel-machines.org/
http://www.support-vector.net/
http://www.support-vector.net/icml-tutorial.pdf
http://www.kernel-machines.org/papers/tutorial-nips.ps.gz
http://www.clopinet.com/isabelle/Projects/SVM/applist.html

Machine Learning Accuracy and Confusion Matrix

Adopted from Andrew Ferlitsch Slide

Accuracy in a Classification Model

- **After a classification model has been trained (e.g., apple vs. pear), a test data set is run against the model to determine accuracy.**
  - **The test data set has labels indicating the expected result (y). E.g., an apple or pear.**
  - **The model outputs predictions (ŷ). E.g., Is it an apple or a pear.**
- **Accuracy is measured as the percentage of predicted results that match the expected results.**

Ex. If there are 1000 results, and 850 predicted results match the expected results, then the accuracy is 85%.

Problem with Accuracy

- **If the training and test data are skewed towards one classification, then the model will predict everything as being that class.**

  Ex. In Titanic training data, 68% of people died. If one trained a model to predict everybody died, it would be 68% accurate.

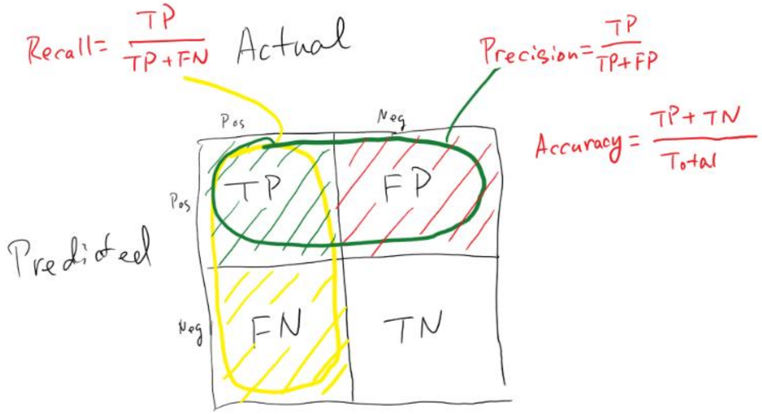
- **The data may be fitted against a feature that is not relevant.**

  Ex. In image classification, if all images of one class have small/similar background, the model may match based on the background, not the object in the image.

Confusion Matrix

https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN



$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

$$ErrorRate = 1 - accuracy$$

$$Precision = Positive\ Predictive\ Value = \frac{TP}{TP + FP}$$

$$Recall = Sensitivity = TP\ Rate = \frac{TP}{TP + FN}$$

$$Specificity = TN\ Rate = \frac{TN}{TN + FP}$$

$$FP\ Rate = \alpha = \frac{FP}{TN + FP} = 1 - specificity$$

$$FN\ Rate = \beta = \frac{FN}{FN + TP} = 1 - sensitivity$$

$$Power = sensitivity = 1 - \beta$$

F1 Score

$$F_1 = 2 \cdot \frac{1}{\frac{1}{\mathrm{recall}} + \frac{1}{\mathrm{precision}}} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}.$$

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN}$$

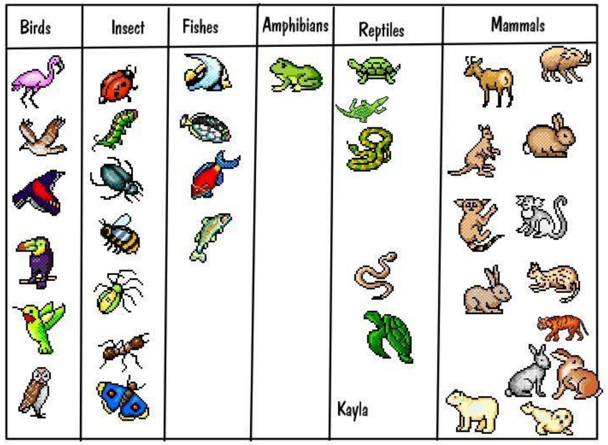
Accuracy vs F-1 Score

- **Accuracy** is used when the True Positives and True negatives are more important while **F1**-**score** is used when the False Negatives and False Positives are crucial

- A medical screening test?

Animal classification

with *softmax_cross_entropy_with_logits*



1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	0
0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	3
1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	0
1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	0
1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	0
1	0	0	1	0	0	0	1	1	1	0	0	4	1	1	1	0
0	0	1	0	0	1	0	1	1	0	0	1	0	1	1	0	3
0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	3
1	0	0	1	0	0	0	1	1	1	0	0	4	0	1	0	0
1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	0
0	1	1	0	1	0	0	0	1	1	0	0	2	1	1	0	1
0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	3
0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	6
0	0	1	0	0	1	1	0	0	0	0	0	4	0	0	0	6
0	0	1	0	0	1	1	0	0	0	0	0	6	0	0	0	6
0	1	1	0	1	0	1	0	1	1	0	0	2	1	0	0	1
1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	0

```
# Predicting animal type based on various features
xy = np.loadtxt('data-04-zoo.csv', delimiter=',', dtype=np.float32)
x_data = xy[:, 0:-1]
y_data = xy[:, [-1]]
```

https://www.kaggle.com/uciml/zoo-animal-classification

tf.one_hot and reshape

1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	0
0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	3
1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	0
1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	0
1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	0
1	0	0	1	0	0	0	1	1	1	0	0	4	1	1	1	0
0	0	1	0	0	1	0	1	1	0	0	1	0	1	1	0	3
0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	3
1	0	0	1	0	0	0	1	1	1	0	0	4	0	1	0	0
1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	0
0	1	1	0	1	0	0	0	1	1	0	0	2	1	1	0	1
0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	3
0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	6
0	0	1	0	0	1	1	0	0	0	0	0	4	0	0	0	6
0	0	1	0	0	1	1	0	0	0	0	0	6	0	0	0	6
0	1	1	0	1	0	1	0	1	1	0	0	2	1	0	0	1
1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	0

```
Y = tf.placeholder(tf.int32, [None, 1])  # 0 ~ 6, shape=(?, 1)
Y_one_hot = tf.one_hot(Y, nb_classes)  # one hot shape=(?, 1, 7)
Y_one_hot = tf.reshape(Y_one_hot, [-1, nb_classes]) # shape=(?, 7)
```