Classification

SUPERVISED LEARNING

Supervised learning

model the relationship between

measured features of data and

some label associated with the data

model helps to assign labels to “new” data

- **classification** tasks (the labels/targets are **discrete** categories)
    - “***binary classification***” target has two possible values
    - “***multiclass classification***” target has more than two possible values,
- **regression** tasks (the labels/targets are **continuous** quantities)

LINEAR REGRESSION (REMINDER)

equation

$$X = [x_1, x_2, x_3, \ldots, x_n]$$

$$\hat{y} = w_1 x_1 + w_2 x_2 + w_3 x_3 + \cdots + w_n x_n + b$$

$$\hat{y} = \sum_{i=1}^{n} w_i x_i + b$$

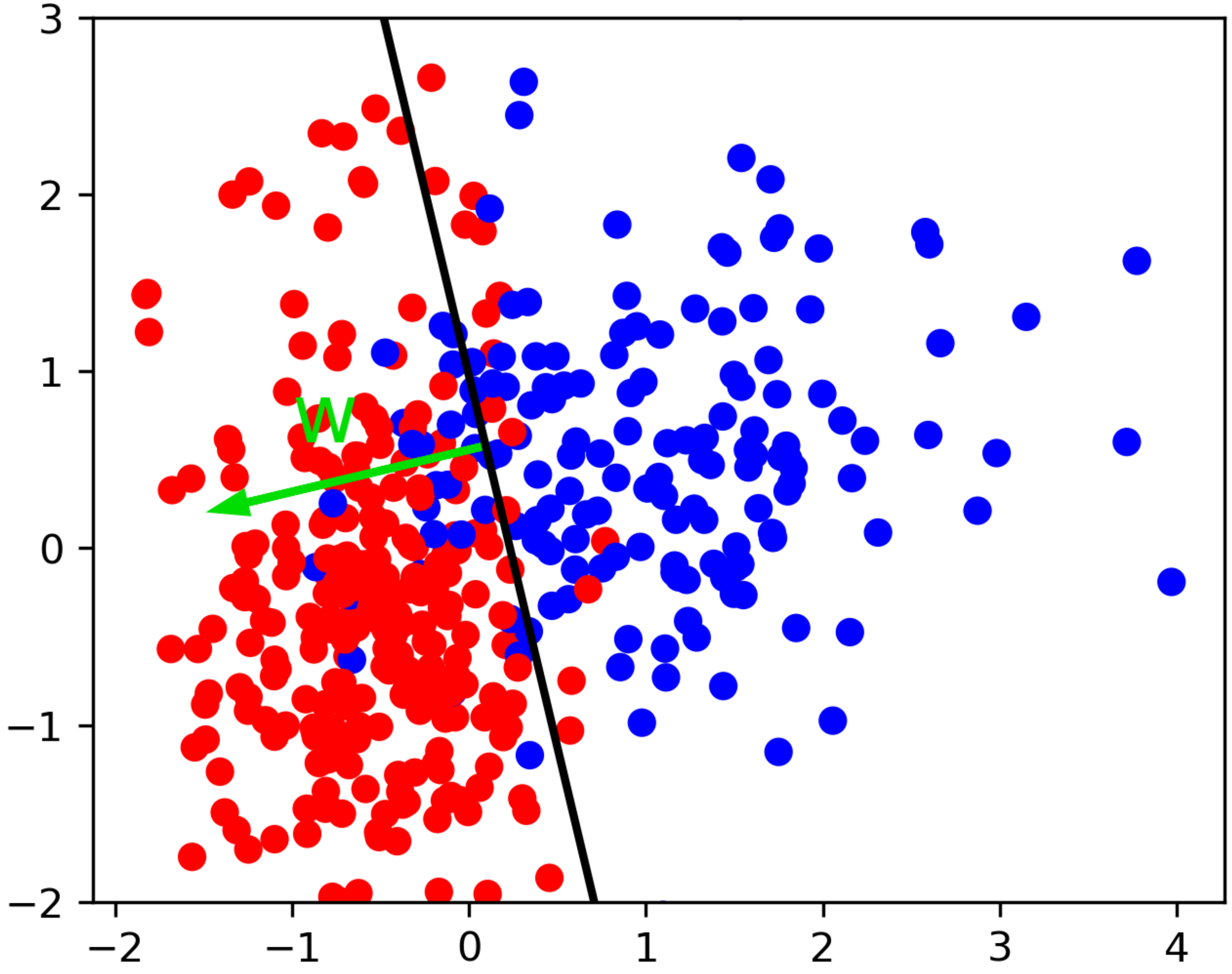
$$\hat{y} = W^T X + b$$

model: matrix notation

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

$$Y = X\beta + \varepsilon$$

BINARY CLASSIFICATION: LINEAR MODELS



$$\hat{y} = \text{sign}(w^T \mathbf{x} + b) = \text{sign}\left(\sum_i w_i x_i + b \right)$$

$$\hat{y} = \text{sign}(w^T \mathbf{x} + b)$$

BINARY CLASSIFICATION: LINEAR MODELS

sign = positive	sign = negative
predicted class	predicted class
"+1"	"-1"
"positive class"	"negative class"
"+" symbol	"-" symbol
"1"	"0"

$$\hat{y} = \text{sign}(w^T \mathbf{x} + b)$$

Loss Function

LOSS FUNCTION

$$\hat{y} = \text{sign}(w^T \mathbf{x} + b)$$

The loss function tells us how badly a model is doing based on training data, i.e. a penalty for an incorrect prediction

Regression: Least Squares:

the squared loss

$$\sum_{i=1}^{n} (true\ target\ i\ - predicted\ target\ i)^2$$

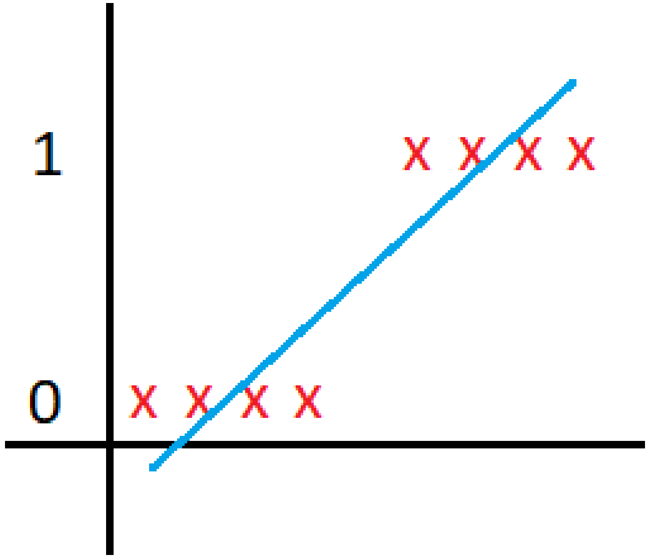
we minimise the sum of squared errors,

with respect to parameters of model

LOSS FUNCTION

$$\hat{y} = \text{sign}(w^T \mathbf{x} + b)$$

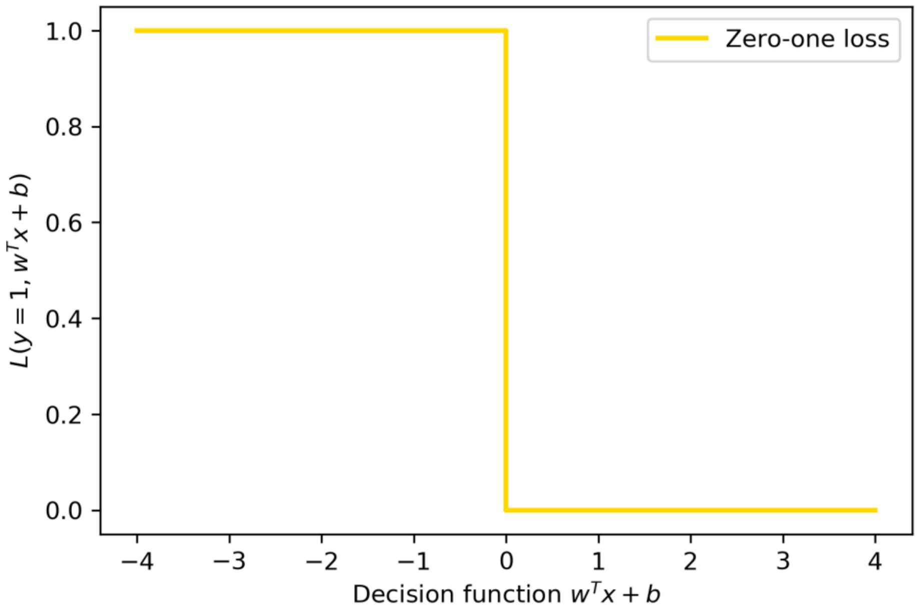
Imagine the scenario of perfect classification,:



the squared loss would be **non** zero

LOSS FUNCTION

$$\hat{y} = \text{sign}(w^T \mathbf{x} + b)$$



$$1_{y_i \neq \text{sign}(w^T \mathbf{x} + b)}$$

indicator which counts 1 every time a mistake occurs

"True class" is ONE:

positive decision function

positive prediction

correct classification: y = "Zero-one loss" = 0

positive decision function

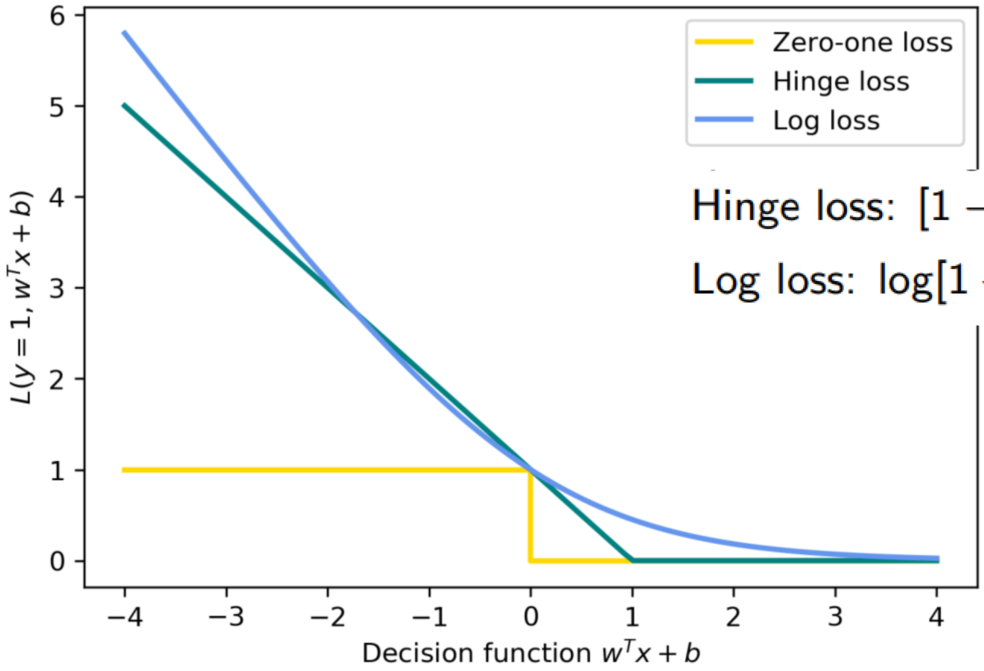
negative prediction =>

wrong classification: y = "Zero-one loss" = 1 [penalized]

minimize number of misclassifications

$$\min_{w \in \mathbb{R}^p, b \in \mathbb{R}} \sum_{i=1}^{n} 1_{y_i \neq \text{sign}(w^T \mathbf{x} + b)}$$

LOSS FUNCTION



Hinge loss: $[1 - y_n \mathbf{w}^T \mathbf{x}_n]_+ = \max\{0, 1 - y_n \mathbf{w}^T \mathbf{x}_n\}$

Log loss: $\log[1 + \exp(-y_n \mathbf{w}^T \mathbf{x}_n)]$

"Zero-one loss" difficult to optimise

- non-convex
- not continuous
- non smooth
- no polynomial time algorithm

Better behaved functions: Hinge Loss, Log Loss

(convex and continuous and upper bounds on the "Zero-one loss")

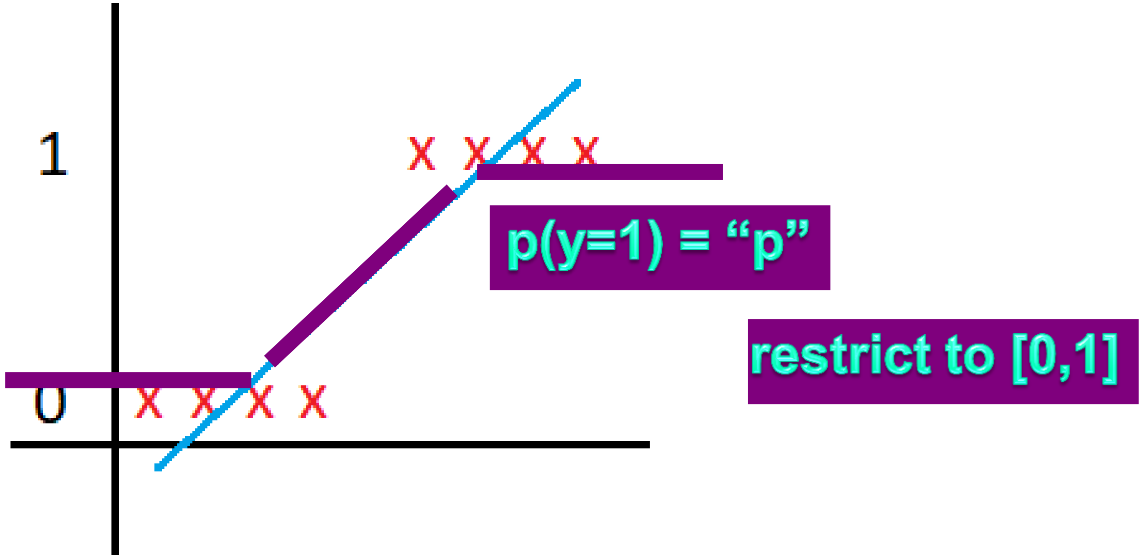
- "how correct" your prediction is

$$\hat{y} = \text{sign}(w^T \mathbf{x} + b)$$

TRY: IMPROVE: THE LOSS

INTUITION

Imagine the scenario of perfect classification,:

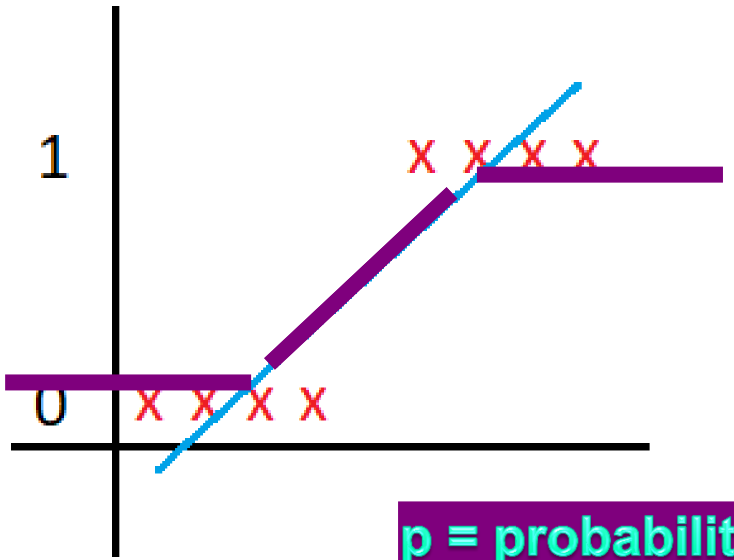


the squared loss would **still** be **non** zero

$$\hat{y} = \mathrm{sign}(w^T\mathbf{x} + b)$$

FURTHER IMPROVE: THE LOSS

INTUITION



p = probability

p = *positive*

$p \leq 1$

p = *exponential*

$$p = \frac{\textit{positive number}}{\textit{positive number} + \textit{small number}}$$

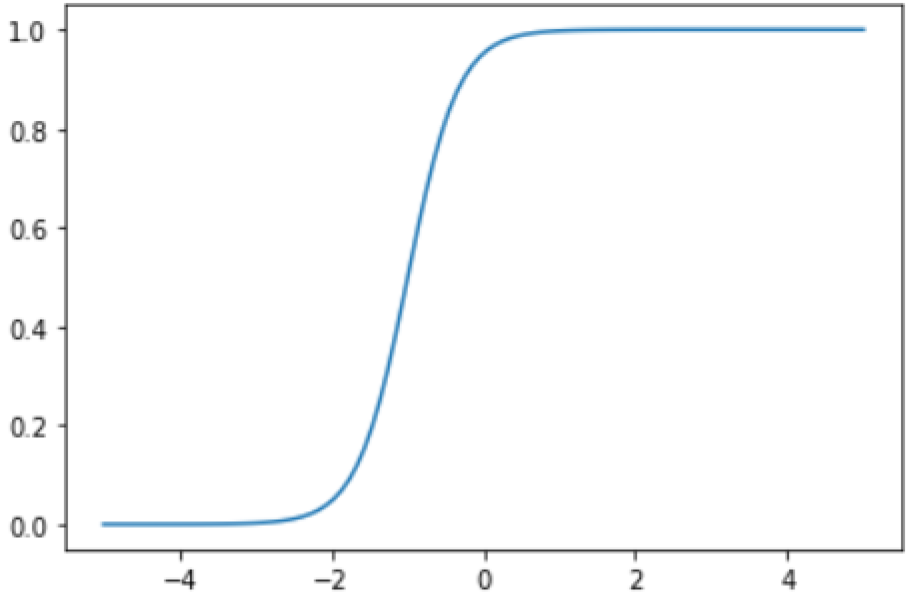
$$p = e^{\beta_0 + \beta_1 x}$$

$$p = \frac{e^{\beta_0 + \beta_1 x}}{e^{\beta_0 + \beta_1 x} + 1}$$

$$\hat{y} = \text{sign}(w^T \mathbf{x} + b)$$

INTUITION

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(-5, 5, 100)
Beta0 = 3
Beta1 = 3
y = (np.exp(Beta0 + Beta1 * x)) / (np.exp(Beta0 + Beta1 * x) + 1)
plt.plot(x, y);
```



$$p = \frac{e^{\beta_0+\beta_1 x}}{e^{\beta_0+\beta_1 x}+1}$$

$$\hat{y} = \text{sign}(w^T\mathbf{x} + b)$$

Sigmoid Function

SIGMOID FUNCTION

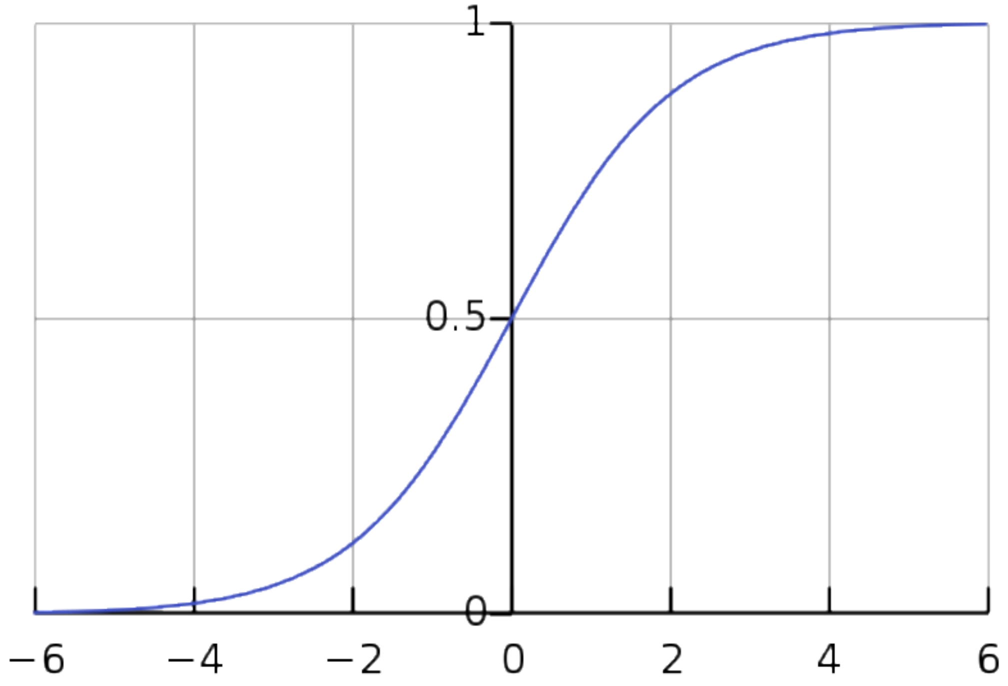
"sigmoid"

or "S-shaped"

or "logistic"

or "inverse logit"

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



useful property of the derivative

$$\begin{aligned}
\frac{d}{dz}\frac{1}{1+e^{-z}} &= \frac{1}{(1+e^{-z})^2}\left(e^{-z}\right) \\
&= \frac{1}{(1+e^{-z})}\cdot\left(1-\frac{1}{(1+e^{-z})}\right) \\
&= g(z)(1-g(z))
\end{aligned}$$

gradient is used in optimisation (ref slide 27)

SIGMOID FUNCTION

"sigmoid"

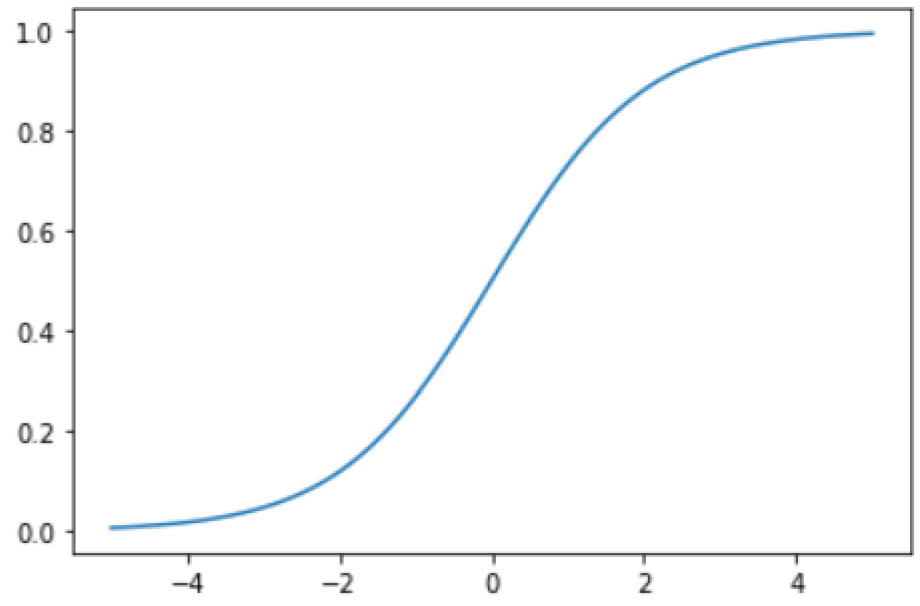
or "S-shaped"

or "logistic"

or "inverse logit"

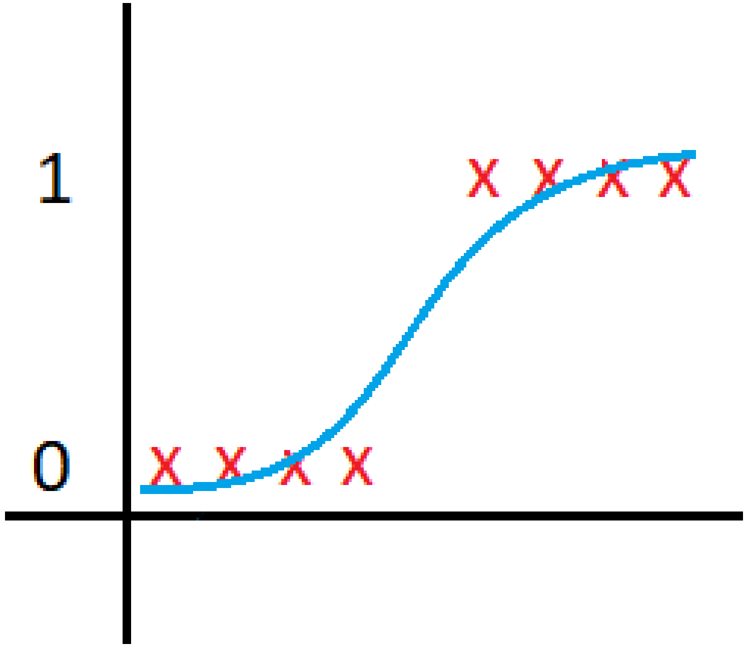
$$g(z) = \frac{1}{1 + e^{-z}}$$

```python
import numpy as np
import matplotlib.pyplot as plt
z = np.linspace(-5, 5, 100)
y = 1 / (1 + np.exp(-x))
plt.plot(z, y);
```



APPLY SIGMOID

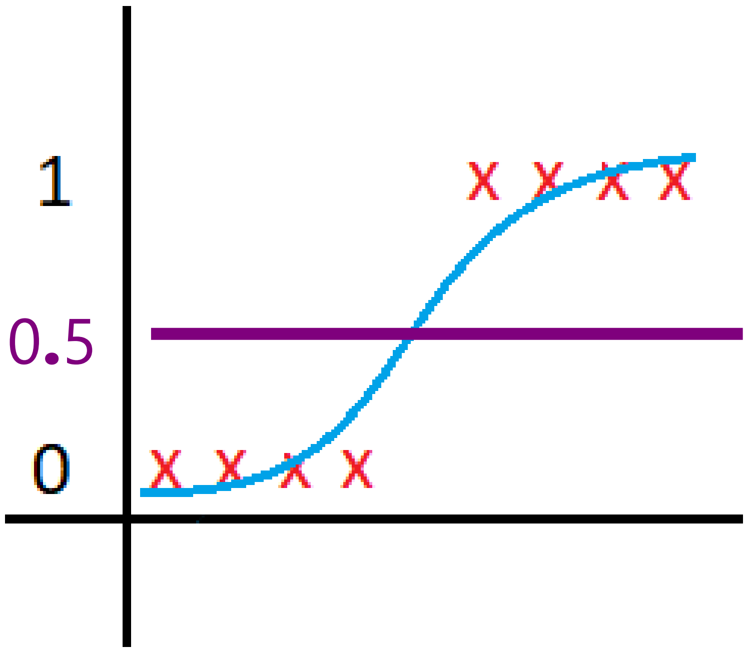
Imagine the scenario of perfect classification:



Consider this classification as a regression with model: $g\ (\mathrm{w}^{\mathrm{T}}x)$,

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

DECISION BOUNDARY



$$\frac{1}{1+e^{-\mathrm{w}^{\mathrm{T}}x}}$$

The prediction function returns a probability (between 0 and 1)

How to map this to a discrete class (true/false, cat/dog) ?

Need a threshold value or tipping point

above which we will classify values into class 1

below which we classify values into class 2.

decision boundary:
if probability ≥ 0.5 then y = 1
[i.e. decision function $w^T x > 0$],
else y = 0

Logistic Regression

LOGISTIC REGRESSION: MODEL

$$\hat{y} = \text{sign}(w^T \mathbf{x} + b)$$

Model: probability of data being from one of two classes:

$$p(\, y = 1 \,|x\,) = \frac{1}{1 + \, e^{-w^T x}}$$

$$p(\, y = 0 \,|x\,) = \frac{e^{-w^T x}}{1 + \, e^{-w^T x}}$$

models probability(y)
i.e. **between** 0 and 1,
(*not:* predicting y: 0 **or** 1)

This can be written more concisely:

$$\frac{p(\, y = 1 \,|x\,; w)}{p(\, y = 0 \,|x\,; w)} = \, e^{w^T x}$$

LOGISTIC REGRESSION: MODEL

$$\frac{p(y = 1 \,|x\,; w)}{p(y = 0 \,|x\,; w)} = e^{w^T x}$$

"odds"

p = 0.8, success
1-p = 0.2, failure

$$\frac{p}{1-p} = \frac{0.8}{0.2} = 4$$

" 4 to 1"

4 times out of 5 the process is expected to be a success

"log odds" = $w^T x$

model the log-odds as a linear regression

$$\hat{y} = \mathrm{sign}(w^T \mathbf{x} + b)$$

LOGISTIC REGRESSION: MODEL

$$ln\left(\frac{p}{1-p}\right) = w^T x$$

$$\frac{p}{1-p} = e^{w^T x}$$

$$p = e^{w^T x}(1-p)$$

$$p = e^{w^T x} - e^{w^T x} p$$

$$p + e^{w^T x} p = e^{w^T x}$$

$$p(1 + e^{w^T x}) = e^{w^T x}$$

$$p = \frac{e^{w^T x}}{1 + e^{w^T x}}$$

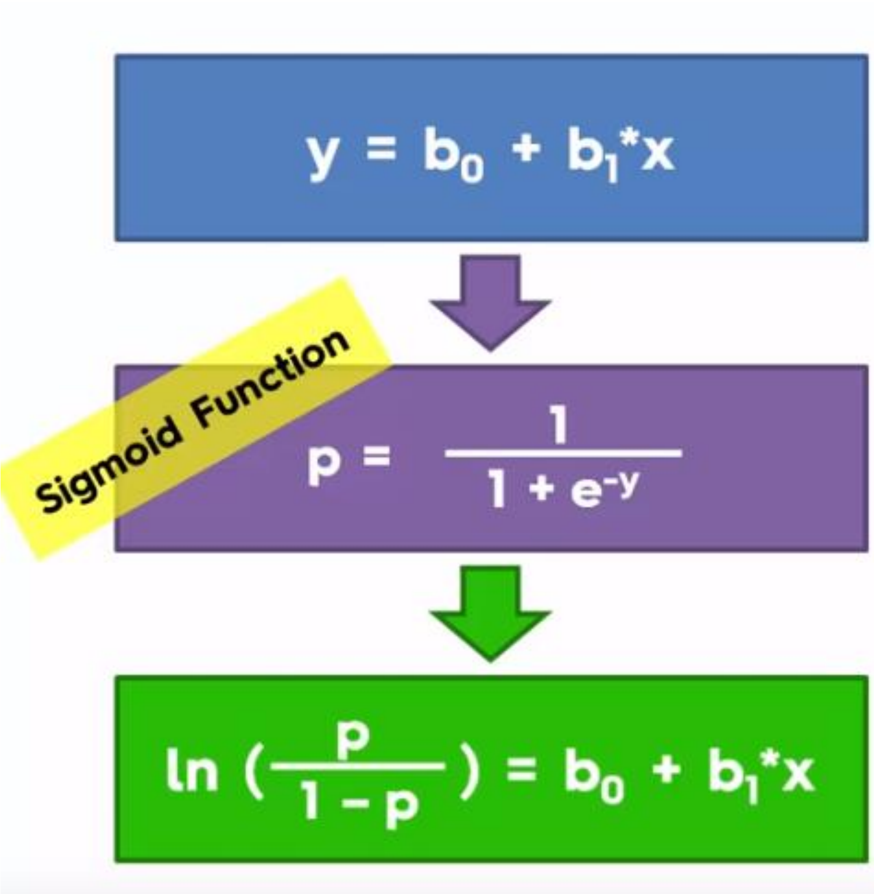
$$p = \frac{\cancel{e^{w^T x}}}{e^{-w^T x} + \cancel{e^{w^T x}}}$$

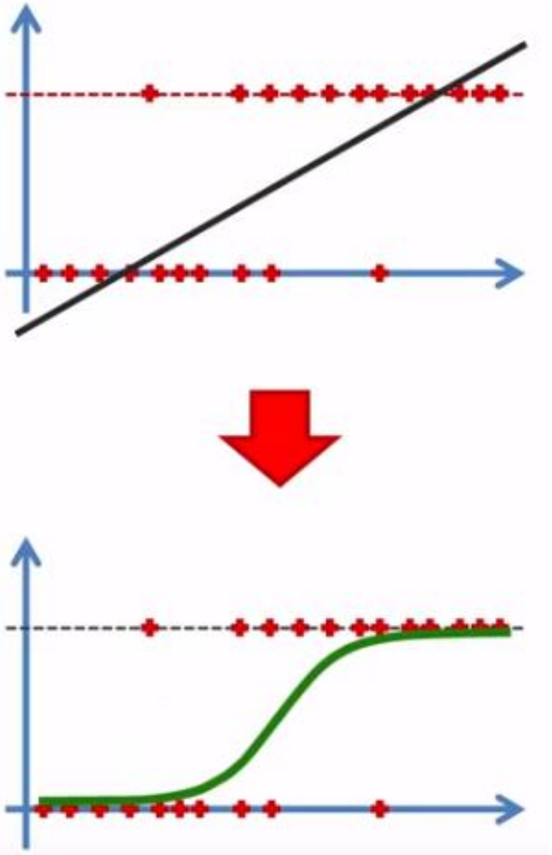
$$p = \frac{1}{e^{-w^T x} + 1}$$

$$p = \frac{1}{1 + e^{-y}}$$

$$\hat{y} = \text{sign}(w^T \mathbf{x} + b)$$

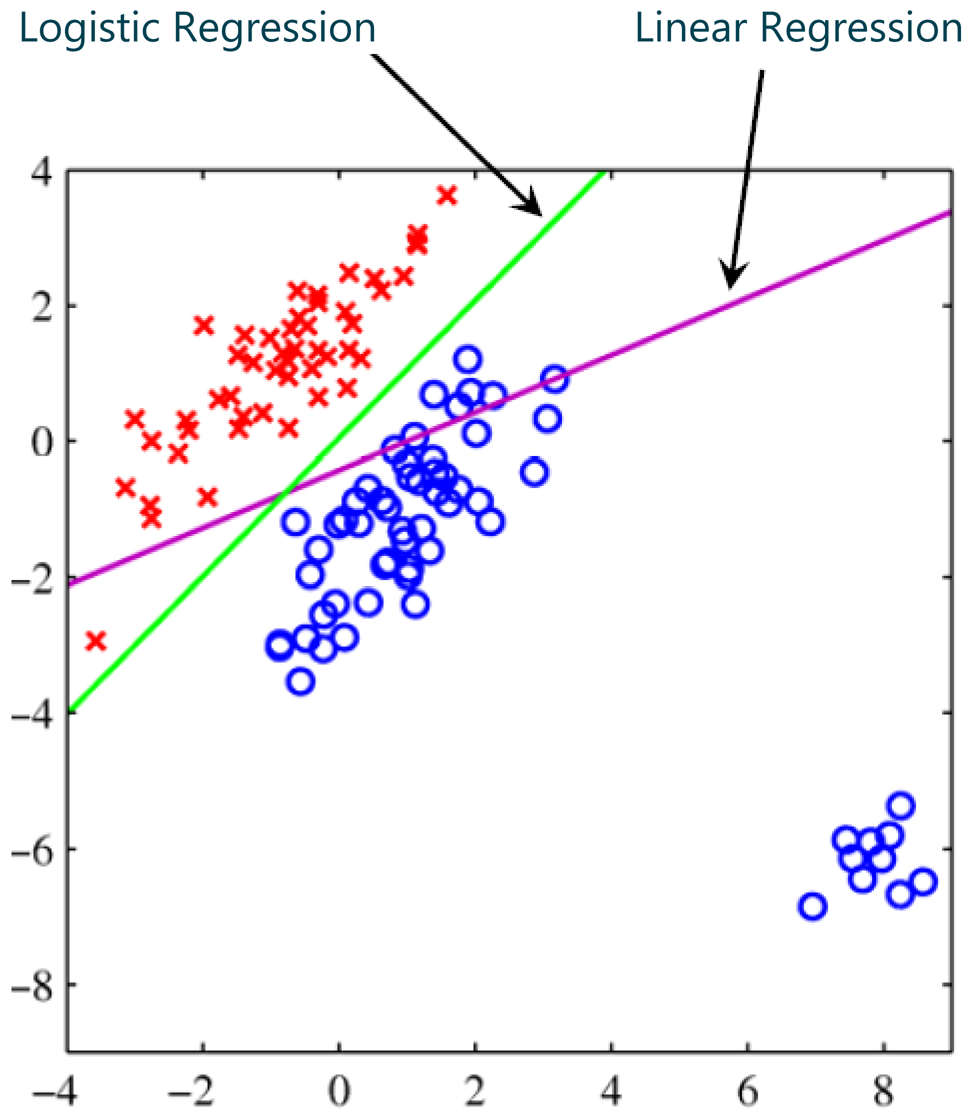
LOGISTIC REGRESSION: MODEL





$$\hat{y} = \mathrm{sign}(w^T \mathbf{x} + b)$$

LOGISTIC REGRESSION: MODEL



$$\hat{y} = \mathrm{sign}(w^T \mathbf{x} + b)$$

Probabilistic Interpretation

LOGISTIC REGRESSION: LIKELIHOOD

Write with θ for parameter weights

and h_θ for the sigmoid hypothesis.

$$\frac{1}{1+e^{-\theta^T x}} = h_\theta(x)$$

"canonical response function"

[wrt: GLM]

The model becomes:

$$\begin{aligned} P(y=1 \mid x;\theta) &= h_\theta(x) \\ P(y=0 \mid x;\theta) &= 1-h_\theta(x) \end{aligned}$$

Compactly: (because Y is a Bernoulli)

$$p(y \mid x;\theta) = (h_\theta(x))^y \left(1-h_\theta(x)\right)^{1-y}$$

Likelihood of θ parameter [m independent training examples]

$$\begin{aligned} L(\theta) &= p(\vec{y} \mid X;\theta) \\ &= \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)};\theta) \\ &= \prod_{i=1}^{m} \left(h_\theta(x^{(i)})\right)^{y^{(i)}} \left(1-h_\theta(x^{(i)})\right)^{1-y^{(i)}} \end{aligned}$$

$$\hat{y} = \mathrm{sign}(w^T \mathbf{x} + b)$$

LOGISTIC REGRESSION: LOG LIKELIHOOD OR COST

The optimal parameters will "maximise the likelihood"

i.e. maximise the probability of training data under our model

i.e. find parameters s.t. maximum probability is assigned to labels observed in the training data

We can optimise either the likelihood function or the log likelihood function, as it is a smooth monotonous function

$$\underbrace{\log L(\theta)}_{\ell(\theta)} = \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)}))$$

Goal: search for a value of θ s.t.

$p(y=1|x) = h_\theta(x)$

is large when x belongs to "1" class

is small when x belongs to "0" class [i.e. p(y=0|x) is large]

$\{(x^{(i)}, y^{(i)}) : i = 1, \dots, m\}$

training examples, binary labels

cost function measures how well a given h_θ fits training data

$$\underbrace{- \sum_i \left(y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))\right)}_{J(\theta)}$$

$$\hat{y} = \mathrm{sign}(w^T \mathbf{x} + b)$$

LOGISTIC REGRESSION: LEARNING PARAMETERS

minimise to find parameters, (take the gradient and set to zero)

$g'(z) = g(z)(1 - g(z))$

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} \ell(\theta) &= \left(y \frac{1}{g(\theta^T x)} - (1 - y) \frac{1}{1 - g(\theta^T x)} \right) \frac{\partial}{\partial \theta_j} g(\theta^T x) \\
&= \left(y \frac{1}{g(\theta^T x)} - (1 - y) \frac{1}{1 - g(\theta^T x)} \right) g(\theta^T x)(1 - g(\theta^T x)) \frac{\partial}{\partial \theta_j} \theta^T x \\
&= \left(y(1 - g(\theta^T x)) - (1 - y) g(\theta^T x) \right) x_j \\
&= \left(y - h_\theta(x) \right) x_j
\end{aligned}$$

$$\hat{y} = \text{sign}(w^T \mathbf{x} + b)$$

LOGISTIC REGRESSION: LEARNING PARAMETERS

$$\hat{y} = \text{sign}(w^T \mathbf{x} + b)$$

minimise to find parameters,

we take the gradient and set to zero

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

in vector form:

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

similar to gradient for linear regression

except that now $h_\theta(x) = \sigma(\theta^\top x)$

rewrite without h_θ & noting y_i takes only two values

$$\min_{w \in \mathbb{R}^p, b \in \mathbb{R}} -\sum_{i=1}^{n} \log(\exp(-y_i(w^T \mathbf{x}_i + b)) + 1)$$

LOGISTIC REGRESSION: WITH OPTIMAL PARAMETERS

$$\hat{y} = \text{sign}(w^T \mathbf{x} + b)$$

Classify a new test point as "1" or "0"

by checking which of these two class labels is most probable:

if $p(y=1|x) > p(y=0|x)$

then we label the example as a "1", and "0" otherwise

i.e. same as checking whether $h_\theta(x) > 0.5$