
# CSC4007 Advanced Machine Learning

## **Lesson 04**: Classification

by Vien Ngo
EEECS / ECIT / DSSC

# Outline

- The simplest approach: k-nearest neighbour
- Discriminative function
- Logistic regression for binary classification
- Multi-class classification

# Outline

- The simplest approach: k-nearest neighbour
- **Discriminative function**
- Logistic regression for binary classification
- Multi-class classification

# Classification: Example

- Real-world example
  - Temperature prediction: sunny, raining, cloudy, etc
  - Fruit classification (apple vs. orange, ripen, small, large, etc.)
  - Autonomous driving: recognize objects (human, cars, tree, ...)
  - IOT: activity recognition (classify running vs. walking vs. sleeping, etc. )
  - etc.

# Example: movie recommendation system

- A data-driven system to recommend a new movie: e.g. should I like *Gravity* if I know its rating and my own likes on some other movies?

| Movie name | Mary's rating | John's rating | I like? |
|---|---|---|---|
| Lord of the Rings II | 1 | 5 | No |
| ... | ... | ... | ... |
| Star Wars I | 4.5 | 4 | Yes |
| Gravity | 3 | 3 | ? |

Le V. Quoc's tutorial

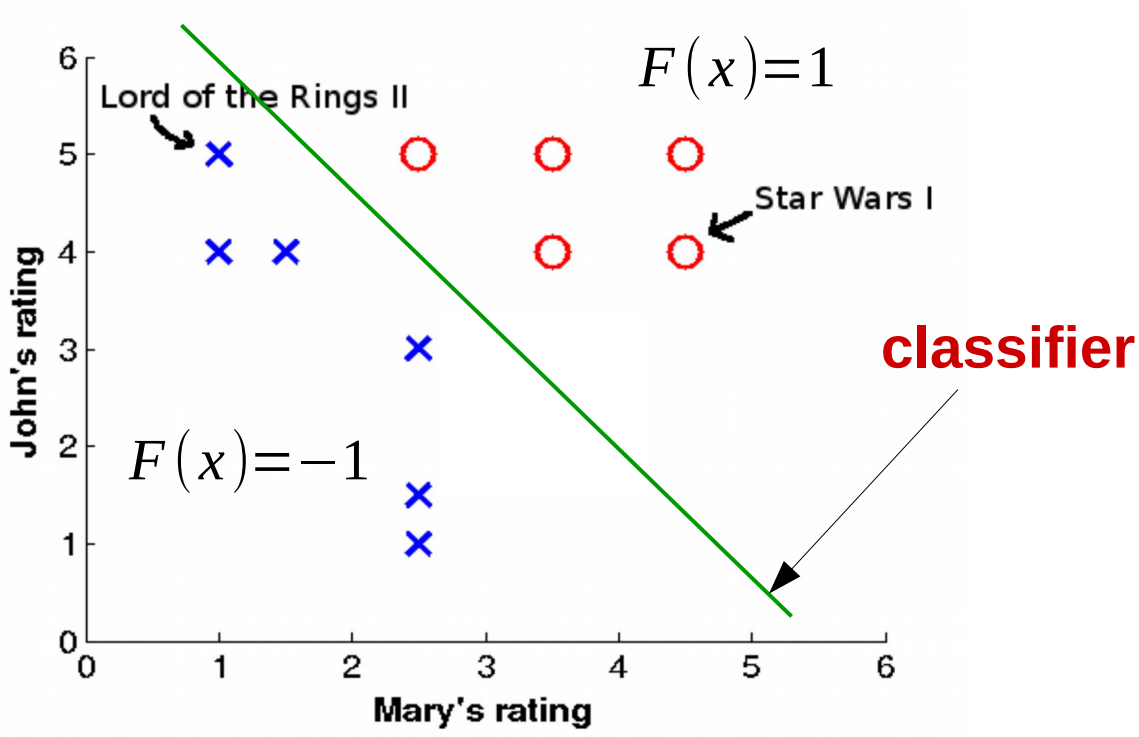
- **inputs** $x_i$: 2-dimensional ($x_{i1}$ = Mary's rating, $x_{i2}$ =John's rating)
  - $x_1 = [1, 5] \quad x_2 = [4.5, 4]$
- **outputs** $y$: Yes or No
  - $y_1$ = No $\quad y_2$ = Yes
- **predictions**: given a new $x$, predict the label of $y$
  - $x = [3, 3]$, $y$ = ? (I would like the movie Gravity or not?)

# Discriminative function

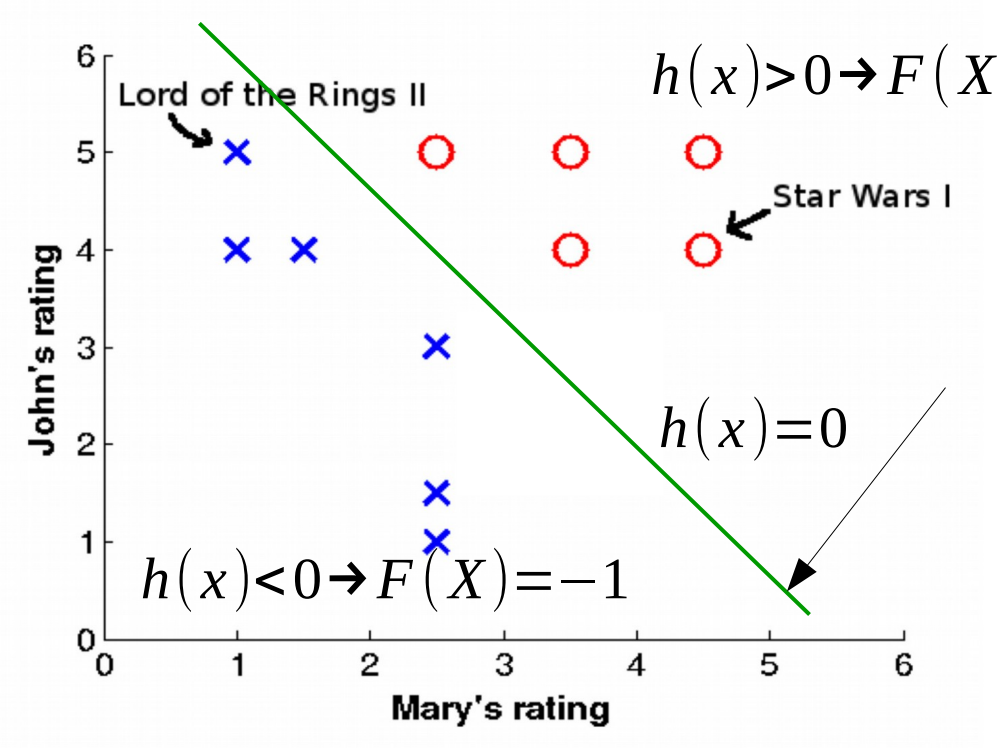
- How to represent hypothesis?
  - e.g. classifer or classification functions

Classifier: {**Yes**=1, **No**=-1}

$F(x)=1$ or $F(x)=-1$



# Discriminative function: binary classification




Regression function: $h(x)=0$ → **classifier**

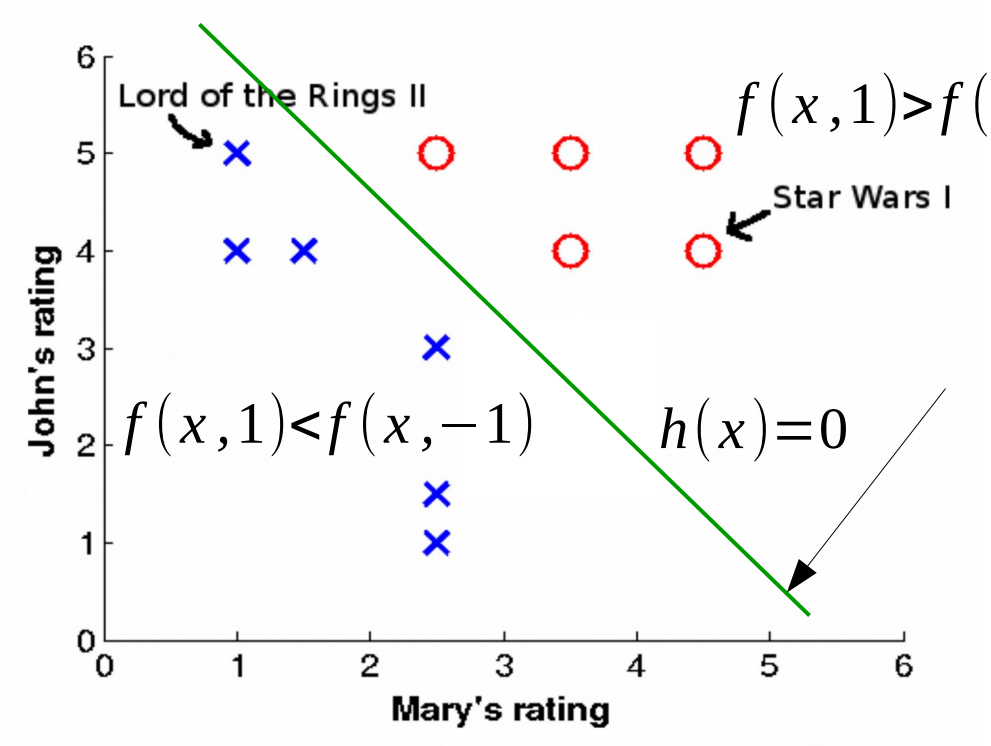
Regression function

$h(x)>0 \rightarrow F(x)=1$

$h(x)<0 \rightarrow F(x)=-1$

→ classification function

# Discriminative function: binary classification




Discriminative function: $h(x)=0=f(x,Yes)-f(x,No)$

$$\underbrace{f(x,1)>f(x,-1)}_{h(x)>0} \rightarrow F(x)=1$$

$$\underbrace{f(x,1)<f(x,-1)}_{h(x)<0} \rightarrow F(x)=-1$$

# Discriminative function

- Discrete-valued classification function

$$F: \mathbb{R}^d \to Y$$

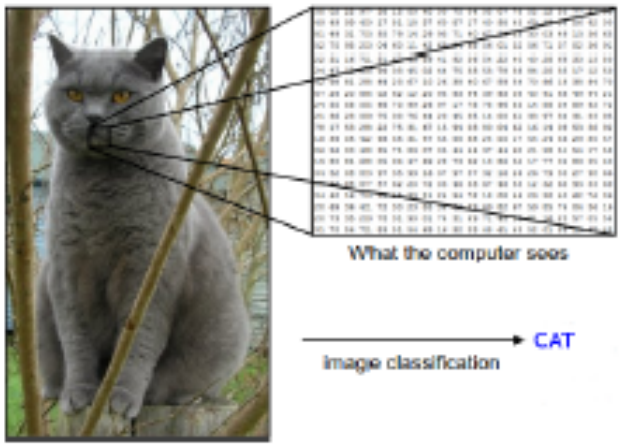
Input x

- Classification via **discriminative function**

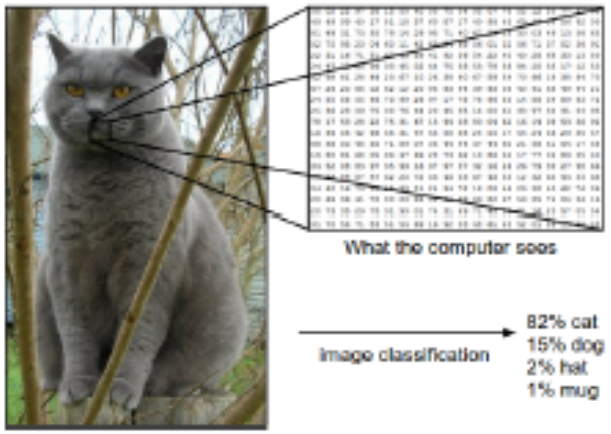
$$f: \mathbb{R}^d \times Y \to \mathbb{R}$$

so: $F: x \mapsto \text{argmax}_y f(x, y)$

Ex.: output $y$ is {cat, dog, hat, mug}




$F(x) = cat$




$f(x, \text{cat}) = 0.82,\ f(x, \text{dog}) = 0.15,$
$f(x, \text{hat}) = 0.02,\ f(x, \text{mug}) = 0.01$

Stanford's cs231n course

**Classification using discriminative function:**

$$F(x) = \arg\max_{\{\text{cat,dog,hat,mug}\}} = \{f(x, \text{cat}), f(x, \text{dog}), f(x, \text{hat}), f(x, \text{mug}) = \textbf{cat}$$

*argmax* operator: return the argument of the max function

# Discriminative function: Example in 1D

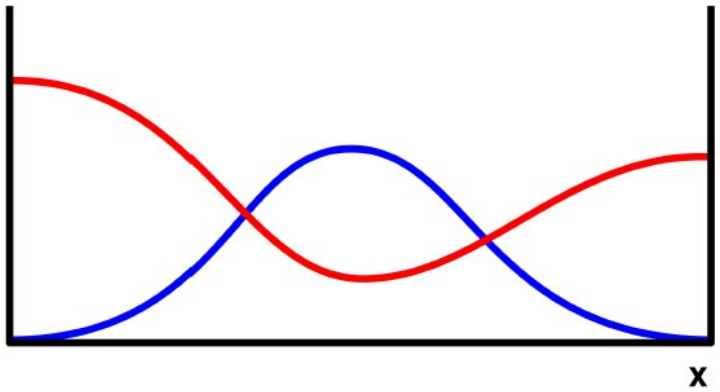
- discriminative function as **continuous function** in 1D: $x \in \mathbb{R}$
  – classification function: $F(x)$ = Red/Blue
  – discriminative function:

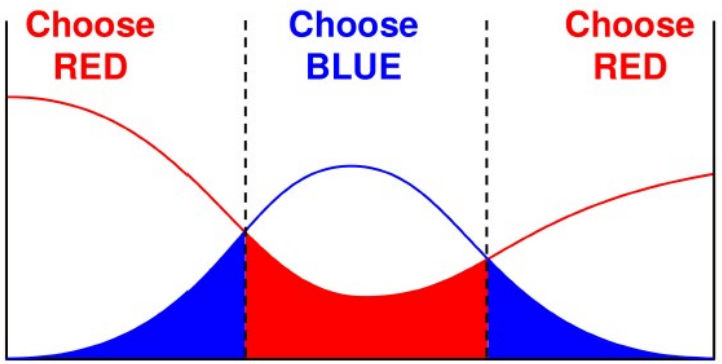
$$f(x = 0.5, y = \text{Red}) = 10 \quad f(x = 0.5, y = \text{Blue}) = 5$$

so

$$F(x) = \arg \max_{\{\text{Blue},\text{Red}\}} \{f(x, \text{Blue}), f(x, \text{Red})\}$$

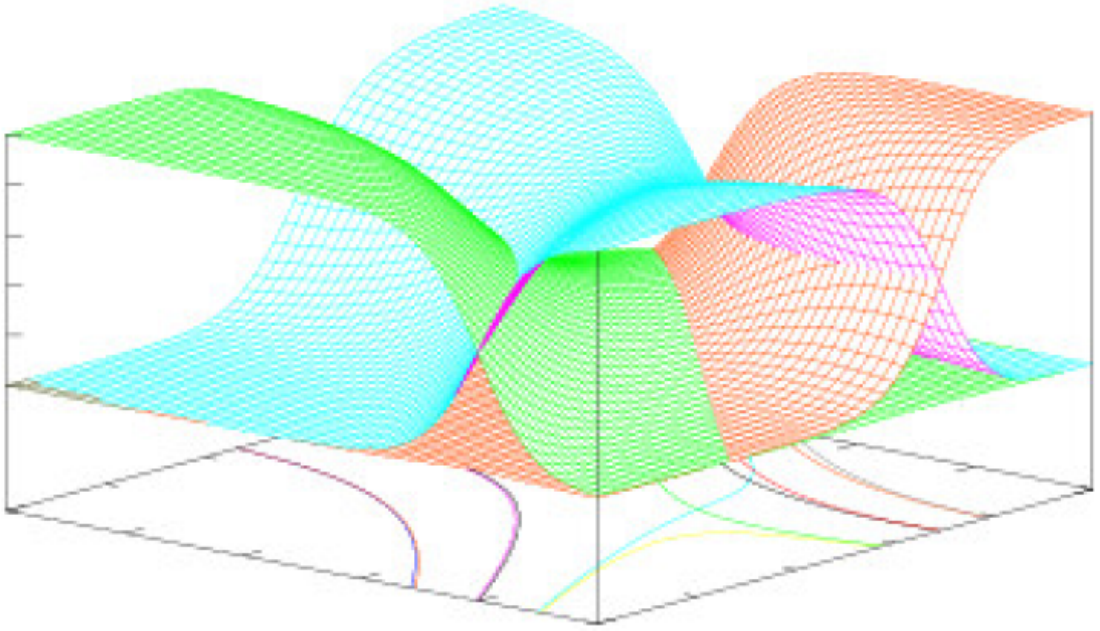
$$= \arg \max_{\{\text{Red},\text{Blue}\}} \{10, 5\} = Red$$



# Discriminative function as probability

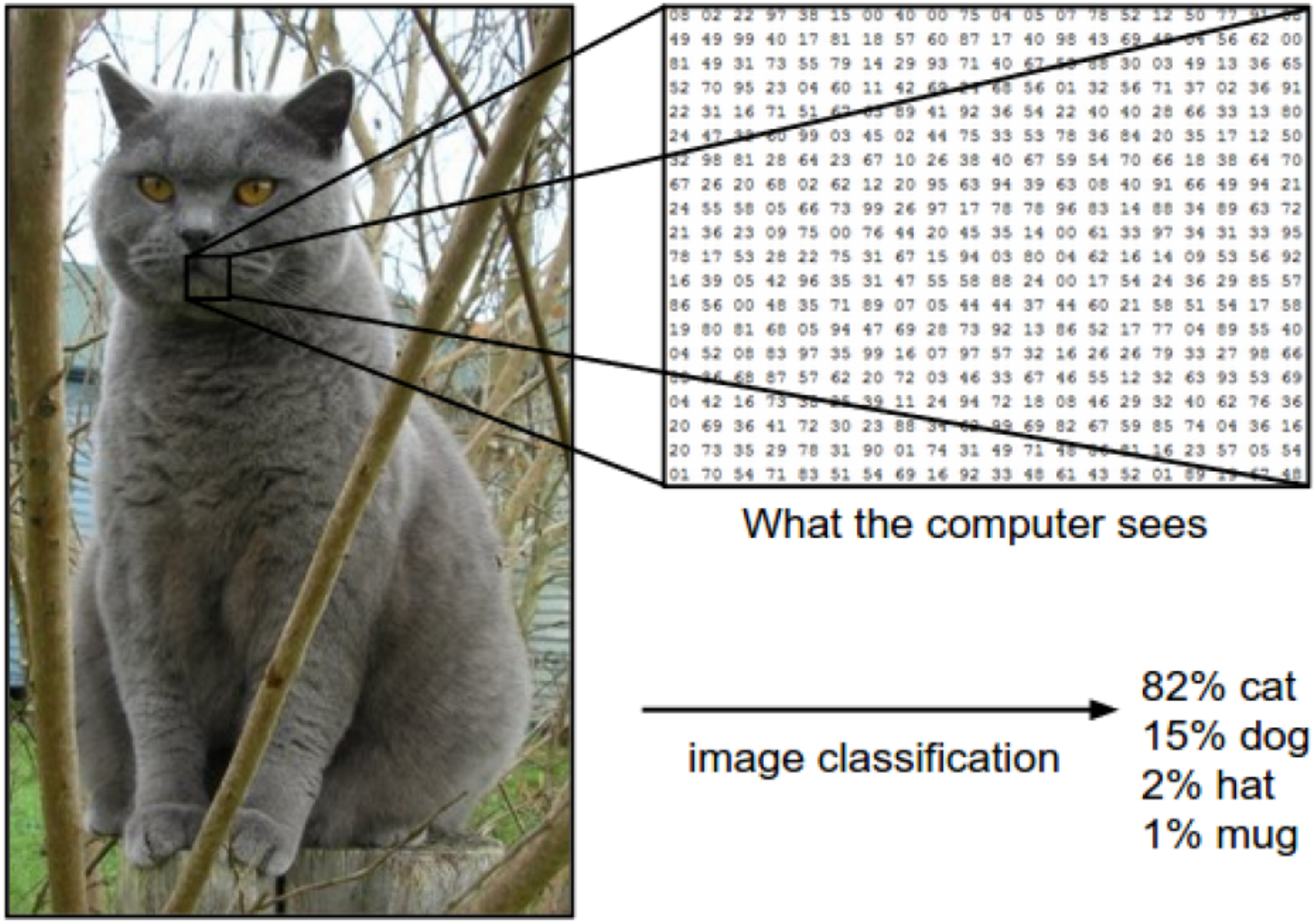
- Input: $x \in \mathbb{R}^2$; output $y \in \{1, 2, 3\}$
  displayed are **likelihood** probabilities $p(y=1|x)$, $p(y=2|x)$, $p(y=3|x)$
- $f(x, y) = p(y|x)$: $p(y=1|x) + p(y=2|x) + p(y=3|x) = 1$



(here already “scaled” to the interval [0,1])

- classification: $F(x) = \arg\max_{\{1,2,3\}} \{p(y = 1|x); p(y = 2|x); p(y = 3|x)\}$

- example of image classification as probability function




– $p(y = \text{cat}|x) = 0.82$, $p(y = \text{dog}|x) = 0.15$, $p(y = \text{cat}|x) = 0.02$, $p(y = \text{mug}|x) = 0.01$

- so, $F(x) =$ **cat** ( because $\max_y p(y|x) = p(y =$ **cat**$|x)$ so **cat** $= \arg\max_y p(y|x)$)

# Discriminative function

- Represent a discrete-valued function $F: \mathbb{R}^n \rightarrow Y$ via a **discriminative function**

$$f: \mathbb{R}^n \times Y \rightarrow \mathbb{R}$$

input x

label y

such that

$$F: x \mapsto \text{argmax}_y f(x, y)$$

That is, a discriminative function $f(x, y)$ maps an input $x$ to an output

$$F(x) = \underset{y}{\text{argmax}}\, f(x, y) \longrightarrow argmax_{cat,dog}\{f(x, cat), f(x, dog)\}$$

- A discriminative function $f(x, y)$ has **high value** if $y$ is a **correct** answer to $x$; and **low** value if $y$ is a **false** answer

# Discriminative function: representation

- linear in features!

$$f(x,y) = \sum_{j=1}^{k} \phi_j(x,y)\beta_j = \phi(x,y)^\top \beta$$

- **example** (linear feature): Let $x \in \mathbb{R}$ and $y \in \{1,2,3\}$. Typical features might be

Linear feature: $\phi(x) = \begin{pmatrix} 1 \\ x \end{pmatrix}$

$$\phi(x,y) = \begin{pmatrix} 1 & [y=1] \\ x & [y=1] \\ 1 & [y=2] \\ x & [y=2] \\ 1 & [y=3] \\ x & [y=3] \end{pmatrix}$$

– where we denote $[y=k]$ means: $[y=k]=1$ if (y==k), $=0$ otherwise

– linear features rewritten: $\phi(x,y) = \begin{pmatrix} \phi(x)[y=0] \\ \phi(x)[y=1] \\ \phi(x)[y=2] \end{pmatrix}$

where $\phi(x) = \begin{pmatrix} 1 \\ x \end{pmatrix}$

# Discriminative function: representation

– so $\beta \in \mathbb{R}^6$ and the discriminative function using linear features is

$$f(x,y) = \begin{pmatrix} 1 & [y=1] \\ x & [y=1] \\ 1 & [y=2] \\ x & [y=2] \\ 1 & [y=3] \\ x & [y=3] \end{pmatrix}^\top \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_6 \end{pmatrix}$$

$$= \beta_1 \cdot 1[y=1] + \beta_2 \cdot x[y=1] + \beta_3 \cdot 1[y=2] + \beta_4 \cdot x[y=2] + \beta_5 \cdot 1[y=3] + \beta_6 \cdot x[y=3]$$

# Example: linear feature, binary cases

- $x \in \mathbb{R}, y \in \{0, 1\}$

binary output

$$\phi(x, y) = \begin{pmatrix} 1 & [y = 0] \\ x & [y = 0] \\ 1 & [y = 1] \\ x & [y = 1] \end{pmatrix}$$

- if $y = 0$ (left) , $y = 1$ (right)

$$\phi(x, 0) = \begin{pmatrix} 1 \\ x \\ 0 \\ 0 \end{pmatrix}, \quad \phi(x, 1) = \begin{pmatrix} 0 \\ 0 \\ 1 \\ x \end{pmatrix}$$

- so classification functions are

$$f(x, y = 0) = \beta_0 + x\beta_1 + 0\beta_2 + 0\beta_3 0 = \beta_0 + x\beta_1$$
$$f(x, y = 1) = 0\beta_0 + 0\beta_1 + \beta_2 1 + \beta_3 x = \beta_2 + x\beta_3$$

decisions

$$F(x) = \operatorname*{argmax}_y f(x, y) = \arg \max_{y=\{0,1\}} \{\beta_0 + x\beta_1, \beta_2 + x\beta_3\}$$

# Example: linear feature, binary cases

## Movie recommendation system example

- an example of two data points $(x_1, y_1), (x_2, y_2)$
  $x_1$ (lord of the ring) $= \{1, 5\}, \quad y_1 = 0$ (No)
  $x_2$ (star wars I) $= \{4.5, 4\}, \quad y_2 = 1$ (Yes)
- linear feature

$$\phi(x_1, y = 0) = \begin{pmatrix} 1 & [y = 0] \\ x_{11} & [y = 0] \\ x_{12} & [y = 0] \\ 1 & [y = 1] \\ x_{11} & [y = 1] \\ x_{12} & [y = 1] \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 5 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

# Example: linear feature, binary cases

## Movie recommendation system example

- an example of two data points $(x_1, y_1), (x_2, y_2)$
  $x_1$ (lord of the ring) $= \{1, 5\}$, $\quad y_1 = 0$ (No)
  $x_2$ (star wars I) $= \{4.5, 4\}$, $\quad y_2 = 1$ (Yes)
- linear feature

$$\phi(x_1, y = 0) = \begin{pmatrix} 1 & [y = 0] \\ x_{11} & [y = 0] \\ x_{12} & [y = 0] \\ 1 & [y = 1] \\ x_{11} & [y = 1] \\ x_{12} & [y = 1] \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 5 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad \phi(x_1, y = 1) = \begin{pmatrix} 1 & [y = 0] \\ x_{11} & [y = 0] \\ x_{12} & [y = 0] \\ 1 & [y = 1] \\ x_{11} & [y = 1] \\ x_{12} & [y = 1] \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 5 \end{pmatrix}$$

# Example: linear feature, binary cases

## Movie recommendation system example

- an example of two data points $(x_1, y_1), (x_2, y_2)$
  $x_1$ (lord of the ring) $= \{1, 5\}$, $\quad y_1 = 0$ (No)
  $x_2$ (star wars I) $= \{4.5, 4\}$, $\quad y_2 = 1$ (Yes)
- linear feature

$$\phi(x_1, y=0) = \begin{pmatrix} 1 & [y=0] \\ x_{11} & [y=0] \\ x_{12} & [y=0] \\ 1 & [y=1] \\ x_{11} & [y=1] \\ x_{12} & [y=1] \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 5 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad \phi(x_1, y=1) = \begin{pmatrix} 1 & [y=0] \\ x_{11} & [y=0] \\ x_{12} & [y=0] \\ 1 & [y=1] \\ x_{11} & [y=1] \\ x_{12} & [y=1] \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 5 \end{pmatrix}$$

– **excercise**:
Compute $\phi(x_2, y=0)$, and $\phi(x_2, y=1)$ ?

# Example: linear feature, binary cases

## Movie recommendation system example

- an example of two data points $(x_1, y_1), (x_2, y_2)$
  $x_1$ (lord of the ring) $= \{1, 5\}$, $\quad y_1 = 0$ (No)
  $x_2$ (star wars I) $= \{4.5, 4\}$, $\quad y_2 = 1$ (Yes)
- linear feature

$$\phi(x_1, y = 0) = \begin{pmatrix} 1 & [y = 0] \\ x_{11} & [y = 0] \\ x_{12} & [y = 0] \\ 1 & [y = 1] \\ x_{11} & [y = 1] \\ x_{12} & [y = 1] \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 5 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \phi(x_1, y = 1) = \begin{pmatrix} 1 & [y = 0] \\ x_{11} & [y = 0] \\ x_{12} & [y = 0] \\ 1 & [y = 1] \\ x_{11} & [y = 1] \\ x_{12} & [y = 1] \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 5 \end{pmatrix}$$

$$\phi(x_2, y = 0) = \begin{pmatrix} 1 & [y_2 = 0] \\ x_{21} & [y_2 = 0] \\ x_{22} & [y_2 = 0] \\ 1 & [y_2 = 1] \\ x_{21} & [y_2 = 1] \\ x_{22} & [y_2 = 1] \end{pmatrix} = \begin{pmatrix} 1 \\ 4.5 \\ 4 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

# Example: linear feature, binary cases

## Movie recommendation system example

- an example of two data points $(x_1, y_1), (x_2, y_2)$
  $x_1$ (lord of the ring) $= \{1, 5\}$, $\quad y_1 = 0$ (No)
  $x_2$ (star wars I) $= \{4.5, 4\}$, $\quad y_2 = 1$ (Yes)
- linear feature

$$\phi(x_1, y=0) = \begin{pmatrix} 1 & [y=0] \\ x_{11} & [y=0] \\ x_{12} & [y=0] \\ 1 & [y=1] \\ x_{11} & [y=1] \\ x_{12} & [y=1] \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 5 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \phi(x_1, y=1) = \begin{pmatrix} 1 & [y=0] \\ x_{11} & [y=0] \\ x_{12} & [y=0] \\ 1 & [y=1] \\ x_{11} & [y=1] \\ x_{12} & [y=1] \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 5 \end{pmatrix}$$

$$\phi(x_2, y=0) = \begin{pmatrix} 1 & [y_2=0] \\ x_{21} & [y_2=0] \\ x_{22} & [y_2=0] \\ 1 & [y_2=1] \\ x_{21} & [y_2=1] \\ x_{22} & [y_2=1] \end{pmatrix} = \begin{pmatrix} 1 \\ 4.5 \\ 4 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \phi(x_2, y=1) = \begin{pmatrix} 1 & [y_2=0] \\ x_{21} & [y_2=0] \\ x_{22} & [y_2=0] \\ 1 & [y_2=1] \\ x_{21} & [y_2=1] \\ x_{22} & [y_2=1] \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 4.5 \\ 4 \end{pmatrix}$$

- so parameters $\beta \in \mathbb{R}^6$, for example

$$f(x_1, y_1) = \phi(x_1, y_1)^\top \beta = \beta_1 + \beta_2 + 5\beta_3 \quad f(x_2, y_2) = \phi(x_2, y_2)^\top \beta = \beta_4 + 4.5\beta_5 + 4\beta_6$$

- classifying?
  – for example: $\beta = [1, 1, 2, 1, 1, 1]$, so

$$f(x_1, y = 0) = \begin{pmatrix} 1 \\ 1 \\ 5 \\ 0 \\ 0 \\ 0 \end{pmatrix}^\top \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_6 \end{pmatrix} = 1 + 1 + 5 \cdot 2 + 0 \cdot 1 + 0 \cdot 1 + 0 \cdot 1 = 12$$

$$f(x_1, y = 1) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 5 \end{pmatrix}^\top \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_6 \end{pmatrix} = 0 \cdot 1 + 0 \cdot 1 + 0 \cdot 2 + 1 + 1 + 5 \cdot 1 = 7$$

decision: $F(x_1) = \arg\max_{y \in \{0,1\}} \left\{ f(x_1, y = 0) = 12, f(x_1, y = 1) = 7 \right\} = 0$

- an example of two data points $(x_1, y_1), (x_2, y_2)$
  $x_1$ (lord of the ring) $= \{1, 5\}$, $\quad y_1 = 0$ (No)
  $x_2$ (star wars I) $= \{4.5, 4\}$, $\quad y_2 = 1$ (Yes)
- linear feature

$$\phi(x_2, y = 0) = \begin{pmatrix} 1 & [y_2 = 0] \\ x_{21} & [y_2 = 0] \\ x_{22} & [y_2 = 0] \\ 1 & [y_2 = 1] \\ x_{21} & [y_2 = 1] \\ x_{22} & [y_2 = 1] \end{pmatrix} = \begin{pmatrix} 1 \\ 4.5 \\ 4 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \phi(x_2, y = 1) = \begin{pmatrix} 1 & [y_2 = 0] \\ x_{21} & [y_2 = 0] \\ x_{22} & [y_2 = 0] \\ 1 & [y_2 = 1] \\ x_{21} & [y_2 = 1] \\ x_{22} & [y_2 = 1] \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 4.5 \\ 4 \end{pmatrix}$$

  – **Exercise**: compute $F(x_2)$ given $\beta = [0, 0, 1, 1, 2, 1]$?

- an example of two data points $(x_1, y_1), (x_2, y_2)$
  $x_1$ (lord of the ring) $= \{1, 5\}$, $\quad y_1 = 0$ (No)
  $x_2$ (star wars I) $= \{4.5, 4\}$, $\quad y_2 = 1$ (Yes)
- linear feature

$$\phi(x_2, y = 0) = \begin{pmatrix} 1 & [y_2 = 0] \\ x_{21} & [y_2 = 0] \\ x_{22} & [y_2 = 0] \\ 1 & [y_2 = 1] \\ x_{21} & [y_2 = 1] \\ x_{22} & [y_2 = 1] \end{pmatrix} = \begin{pmatrix} 1 \\ 4.5 \\ 4 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \phi(x_2, y = 1) = \begin{pmatrix} 1 & [y_2 = 0] \\ x_{21} & [y_2 = 0] \\ x_{22} & [y_2 = 0] \\ 1 & [y_2 = 1] \\ x_{21} & [y_2 = 1] \\ x_{22} & [y_2 = 1] \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 4.5 \\ 4 \end{pmatrix}$$

  – **Exercise**: compute $F(x_2)$ given $\beta = [0, 0, 1, 1, 2, 1]$?
  $F(x_2) = \arg\max_{y \in \{0,1\}} \Big\{ f(x_2, y = 0) = 4, f(x_2, y = 1) = 14 \Big\} = 1$

# Example: quadratic feature, multi-class cases

- Example (**quadratic feature**): Let $x \in \mathbb{R}$ and $y \in \{1, 2, 3\}$. Typical features might be

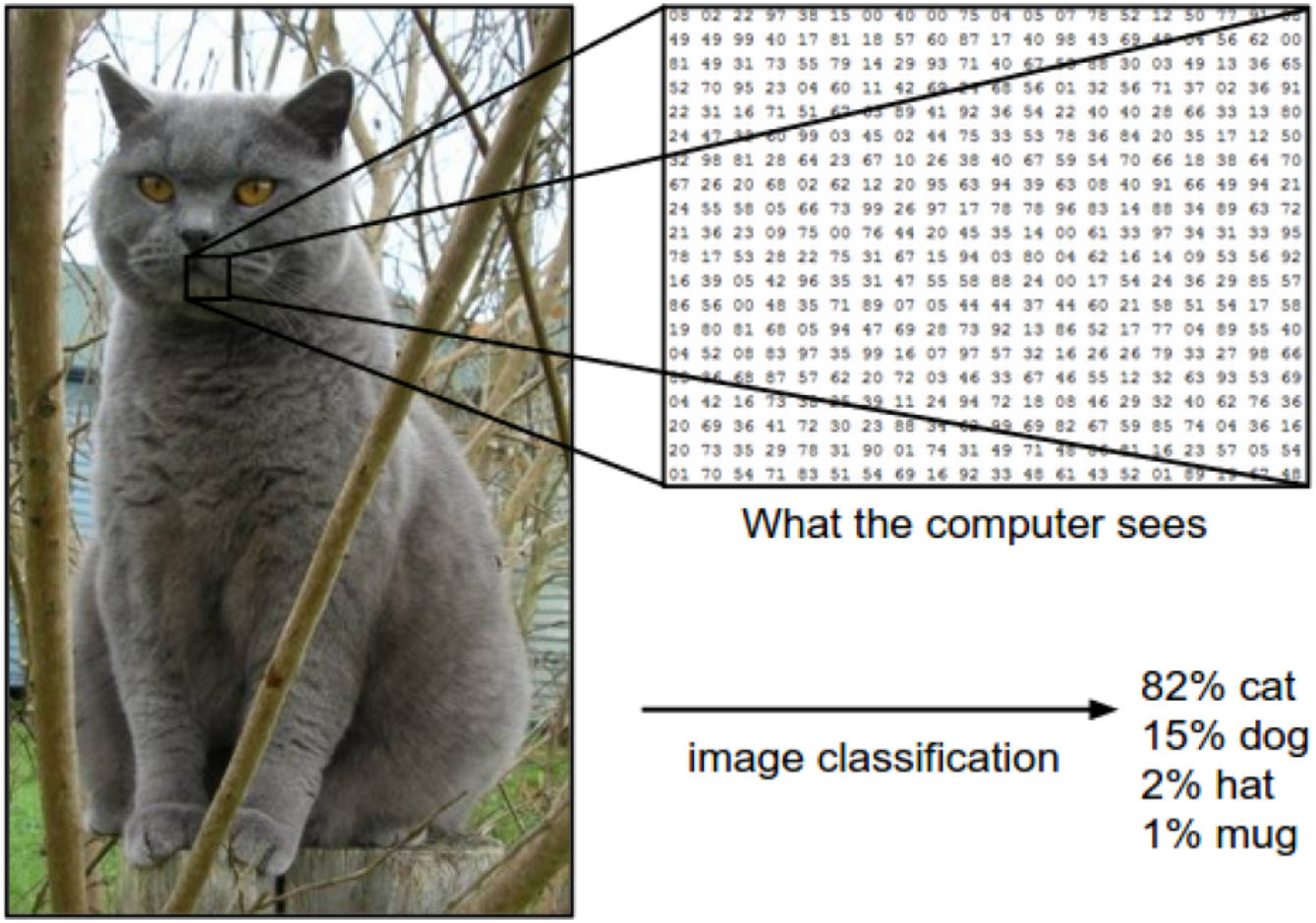
$$\phi(x, y) = \begin{pmatrix} 1 & [y = 1] \\ x & [y = 1] \\ x^2 & [y = 1] \\ 1 & [y = 2] \\ x & [y = 2] \\ x^2 & [y = 2] \\ 1 & [y = 3] \\ x & [y = 3] \\ x^2 & [y = 3] \end{pmatrix}$$

# Outline

- The simplest approach: k-nearest neighbour
- Discriminative function
- **Logistic regression for binary classification**
- Multi-class classification

# Logistic regression

- example of image classification as probability function




– $p(y = \text{cat}|x) = 0.82$, $p(y = \text{dog}|x) = 0.15$, $p(y = \text{hat}|x) = 0.02$, $p(y = \text{mug}|x) = 0.01$

- so, $F(x) =$ **cat** ( because $\max_y p(y|x) = p(y =$ **cat**$|x)$ so **cat** $= \arg\max_y p(y|x)$)

# Logistic regression for binary classification

- Classificatiton using likehood function

$$0 \leqslant p(y|x) \leqslant 1$$

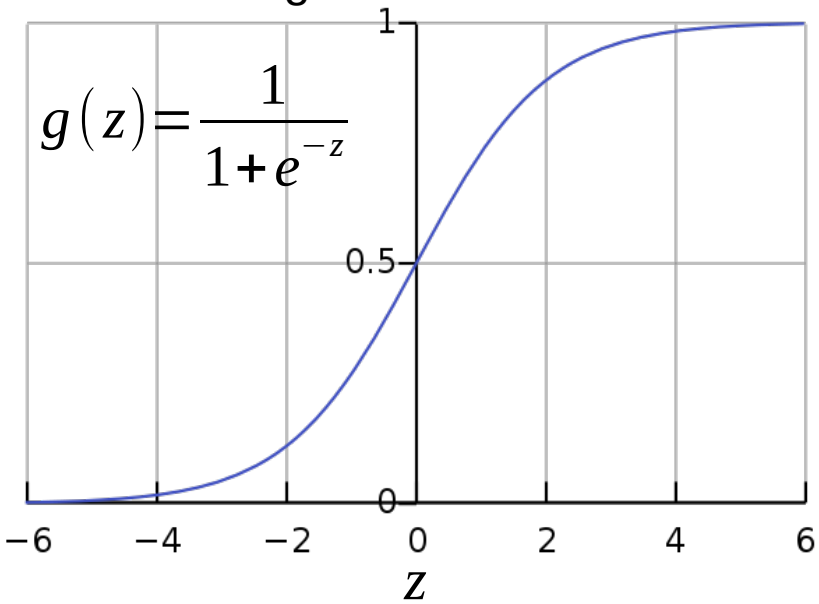
The probability of being y=1 given x

- Using a sigmoid function to transform discriminate function

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

**discriminate function**

sigmoid function




$$0 \leqslant g(z) \leqslant 1$$

# Logistic regression

- Classificatiton using likelihood function

$$0 \leqslant p(y|x) \leqslant 1$$

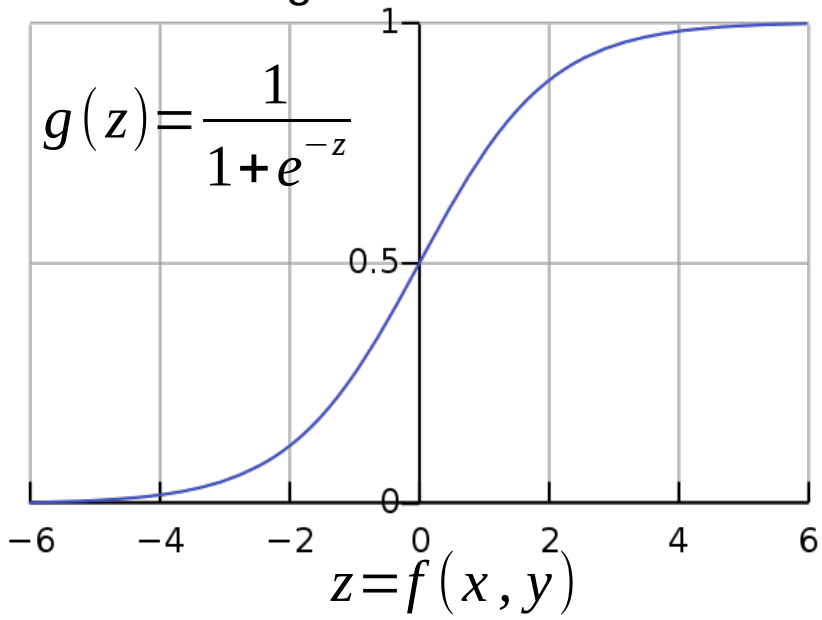
The probability of being y=1 given x

- Using a sigmoid function to transform discriminate function

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

**discriminate function**

sigmoid function




$$0 \leqslant g(z) \leqslant 1$$

# Logistic regression

- Classificatiton using likehood function

$$0 \leqslant p(y|x) \leqslant 1$$

The probability of being y=1 given x

- Using a sigmoid function to transform discriminate function

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

**Linear discriminate function**

$$= \frac{1}{1+e^{-\beta^T \phi(x)}}$$

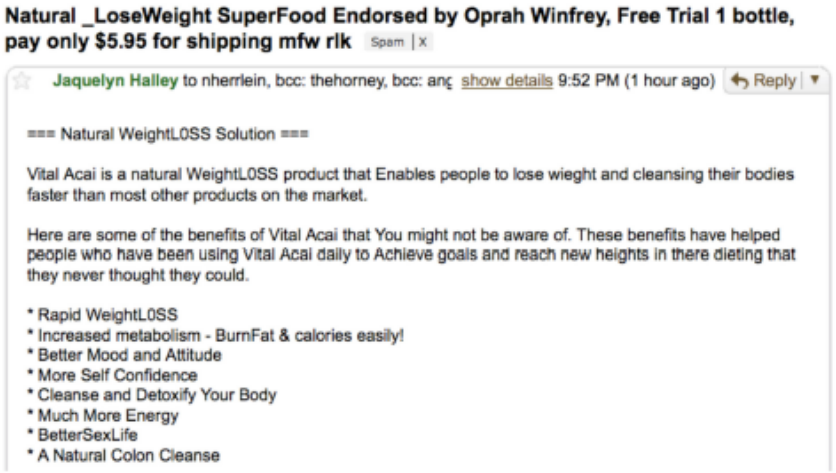
**As y is constant, it is simplified**

# Logistic regression: Interpretation

- Parametric likehood function

$$p(y|x;\theta)=\frac{1}{1+e^{-f(x,y)}}=\frac{1}{1+e^{-\beta^T\phi(x,y)}}$$

$$\longrightarrow p(y=0|x;\beta)=1-p(y|x;\beta)$$




***e.g.*** $x$ = [winner =2, prize =1, $dd=2, ...]

$$p(spam|x;\beta)=0.9$$

With probability 0.9: x is a spam email

$$p(not\ spam|x;\beta)=1-p(not\ spam|x;\beta)=1-0.9=0.1$$

With probability 0.1, x is not a spam email

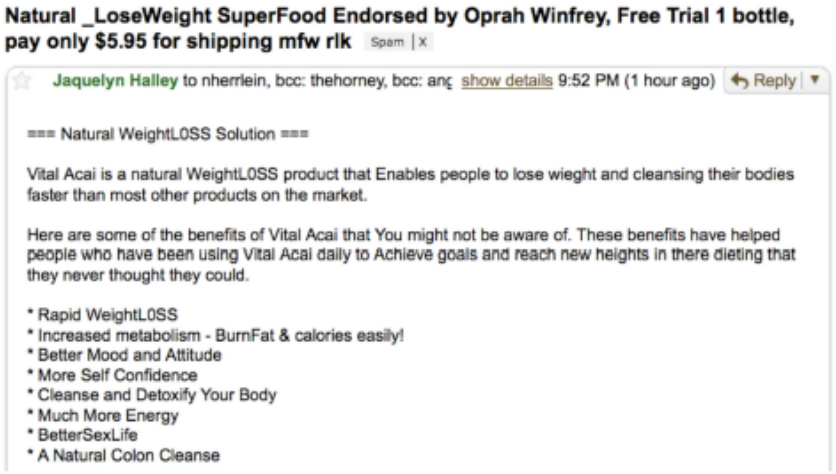
# Logistic regression: Interpretation

- Classifying in binary case y is {**1** or **0**}?

$$p(y|x;\beta) > 0.5 \longrightarrow \text{Predict y=1}$$

$$p(y|x;\beta) \leqslant 0.5 \longrightarrow \text{Predict y=0}$$

The probability of being y=1 given x




***e.g.*** $x = [\text{winner} = 2,\ \text{prize} = 1,\ \$dd = 2,\ \ldots]$

$$p(spam|x;\beta) = 0.9$$

Predict: y = spam

# Logistic regression: movie recommendation example

| Movie name | Mary's rating | John's rating | I like? |
|---|---|---|---|
| Lord of the Rings II | 1 | 5 | No |
| ... | ... | ... | ... |
| Star Wars I | 4.5 | 4 | Yes |
| Gravity | 3 | 3 | ? |

Le V. Quoc's tutorial

- Example: $x_1 = [1, 5]^\top$ (lord of the ring), if $\beta = [1, 0.5, 0]^\top$,
  – using linear feature $\phi(x_1) = [1, x_{11}, x_{12}]^\top$, then

$$p(y = 1|x_1) = \frac{1}{1 + e^{-f(x_1,1)}} = \frac{1}{1 + e^{-\phi(x_1)^T\beta}} = \frac{1}{1 + e^{-1\times1-1\times0.5-5\times0}} = 0.81$$

It is with a probability of 0.81 that I will like (Yes, y=1) the movie **lord of the ring**

$p(y = 1|x_1) + p(y = 0|x_1) = 1.0$, so $p(y = 0|x_1) = 1 - p(y = 1|x_1) = 0.19$

# Logistic regression: movie recommendation example

| Movie name | Mary's rating | John's rating | I like? |
|---|---|---|---|
| Lord of the Rings II | 1 | 5 | No |
| ... | ... | ... | ... |
| Star Wars I | 4.5 | 4 | Yes |
| Gravity | 3 | 3 | ? |

Le V. Quoc's tutorial

- Example: $x_1 = [1, 5]^\top$ (lord of the ring), if $\beta = [1, 0.5, 0]^\top$,
  – using linear feature $\phi(x_1) = [1, x_{11}, x_{12}]^\top$, then

$$p(y = 1|x_1) = \frac{1}{1 + e^{-f(x_1,1)}} = \frac{1}{1 + e^{-\phi(x_1)^T\beta}} = \frac{1}{1 + e^{-1\times1-1\times0.5-5\times0}} = 0.81$$

It is with a probability of 0.81 that I will like (Yes, y=1) the movie **lord of the ring**

$p(y = 1|x_1) + p(y = 0|x_1) = 1.0$, so $p(y = 0|x_1) = 1 - p(y = 1|x_1) = 0.19$

– $p(y = 1|x_1) = 0.81$ means: it's very likely true (I will like this movie)
– $p(y = 0|x_1) = 0.19$ means: it's very unlikely true (I will not like this movie)

# Logistic regression: Optimal parameters

- Given a training dataset of ***n* instances** of input-output

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$ ← $y_i$ is binary {0,1}

- Find optimal parameter $\beta^*$ such that

$$p(y=1|x;\beta) = \frac{1}{1+e^{-\beta^T \phi(x)}}$$ is high for all $(x_i, y_i), i=[1,\ldots,n]$

- **Negative log likelihood cost function**

$$l(x_i, y_i) = -\log p(y_i|x_i;\beta) \quad \text{if} \quad y_i = 1$$

$$l(x_i, y_i) = -\log(1 - p(y_i|x_i;\beta)) \quad \text{if} \quad y_i = 0$$

**NOTE: we only model $p(y=1|x;\beta)$,**
**so $p(y|x;\beta)$ means the probability of y=1 given x**

# Logistic regression: Optimal parameters

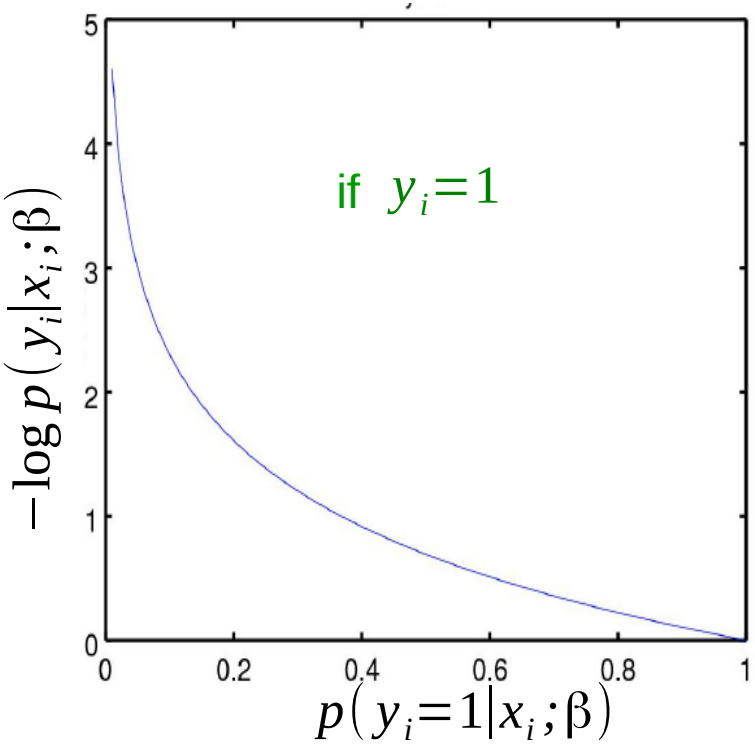
- **Negative log likelihood cost function: interpretation**

$$l(x_i, y_i) = -\log p(y_i|x_i;\beta) \quad \text{if} \quad y_i = 1$$

$$l(x_i, y_i) = -\log(1 - p(y_i|x_i;\beta)) \quad \text{if} \quad y_i = 0$$

Minimum cost = maximum $p(y_i=1|x_i;\beta)$

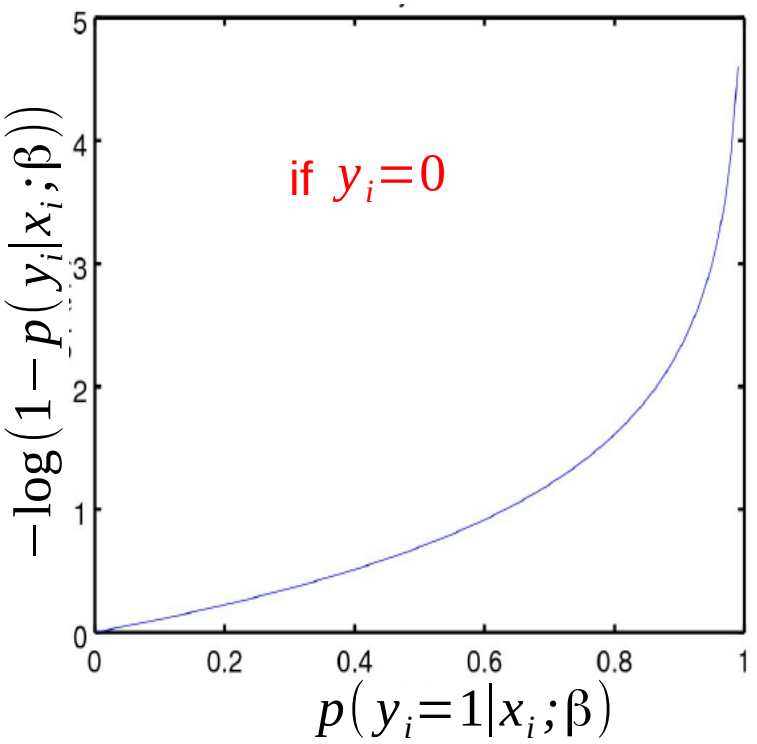
Then prediction $y_i=1$ is most likely




Minimum cost = minimum $p(y_i=1|x_i;\beta)$

Then prediction $y_i=1$ is least likely,

Hence $y_i=0$ is most likely



# Logistic regression: Optimal parameters

- **Negative log likelihood cost function: Find $\beta^*$ that minimizes**

$$L^{\text{logistic}}(\beta) = -\sum_{i=1}^{n} \Big[ y_i \log p(y_i \mid x_i) + (1 - y_i) \log[1 - p(y_i \mid x_i)] \Big] + \lambda \|\beta\|^2$$

Regularization

Cost for all data points

+) if $y_i = 0$, then

$$\Big[ y_i \log p(y_i \mid x_i) + (1 - y_i) \log[1 - p(y_i \mid x_i)] \Big] = \log(1 - p(y_i|x_i))$$

+) if $y_i = 1$, then

$$\Big[ y_i \log p(y_i \mid x_i) + (1 - y_i) \log[1 - p(y_i \mid x_i)] \Big] = \log(p(y_i|x_i))$$

# Logistic regression: Optimal parameters

- **Find $\beta^*$ that minimizes:**

$$L^{\text{logistic}}(\beta) = -\sum_{i=1}^{n} \left[ y_i \log p(y_i \mid x_i) + (1 - y_i) \log[1 - p(y_i \mid x_i)] \right] + \lambda \|\beta\|^2$$

- Using gradient-descent optimization method
  - Taking gradients:

$$\frac{\partial L^{\text{logistic}}(\beta)}{\partial \beta}$$

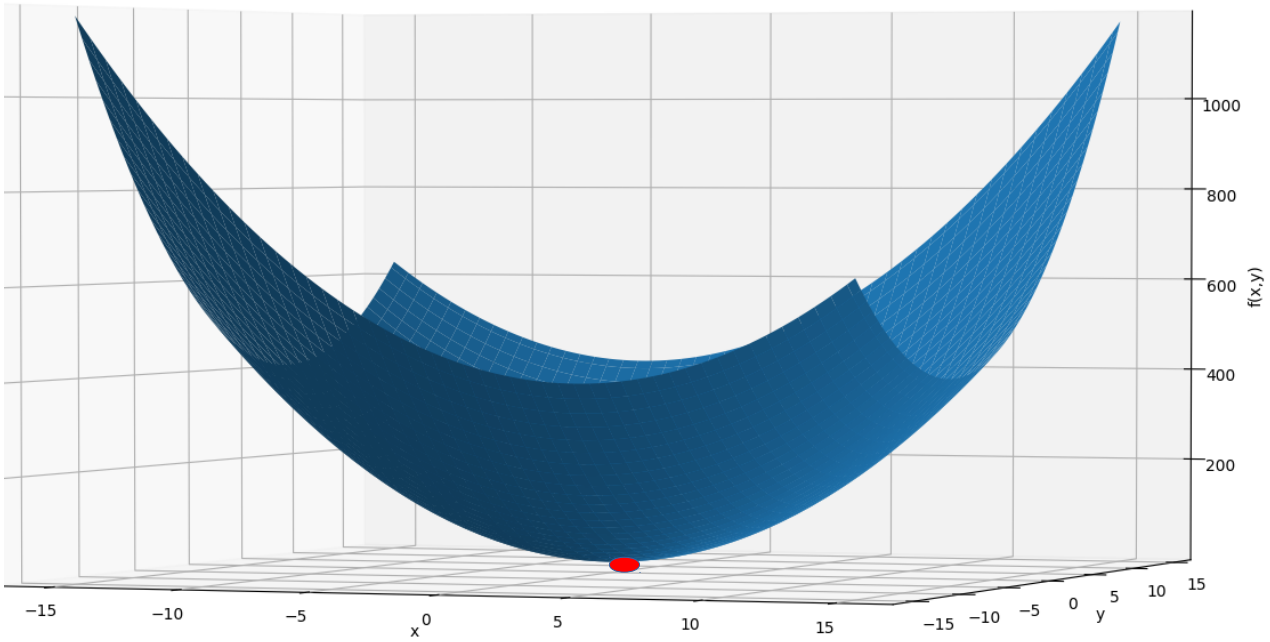
# Review: gradient descent algorithm

- How to optimize an arbitrary function $f(\beta)$, where $\beta \in \Re^d$?

  – e.g: find $x^*$ and $y^*$ that minimizes $f(x, y) = 2 * x^2 + 1.2 * x * y + 2 * y^2$

Parameter $\beta = [x, y] \in \Re^2$

Optimal parameter $\beta^* = [x^*, y^*]$



# Review: gradient descent algorithm

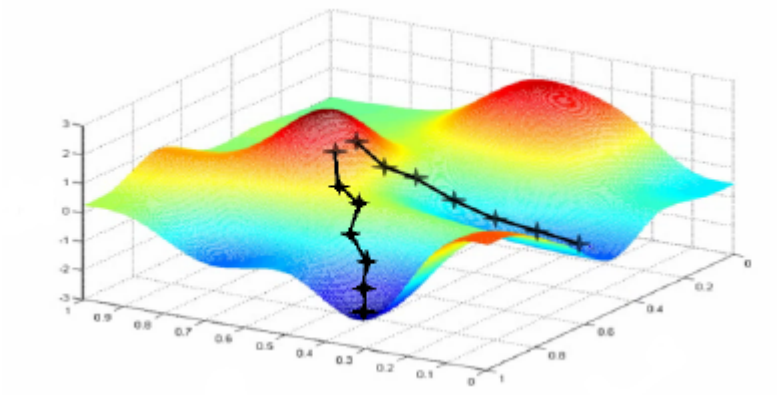
- How to optimize an arbitrary function $f(\beta)$, where $\beta \in \Re^d$?

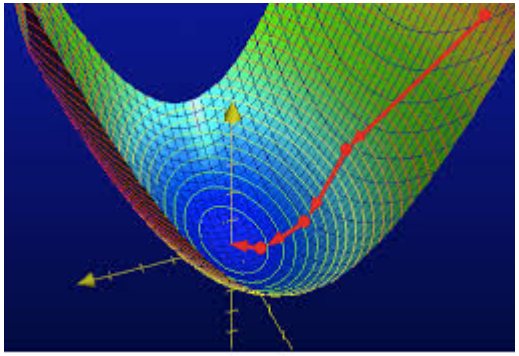
  – e.g: find $x^*$ and $y^*$ that minimizes $f(x, y) = 2 * x^2 + 1.2 * x * y + 2 * y^2$

- **Objective function** (finding its minimum): $f : \mathbb{R}^d \rightarrow \mathbb{R}$

  –assumption: we have the gradient of $f(\beta)$

$$\nabla f(\beta) = \left[\frac{\partial f(\beta)}{\partial \beta}\right]^\top \in \mathbb{R}^d$$





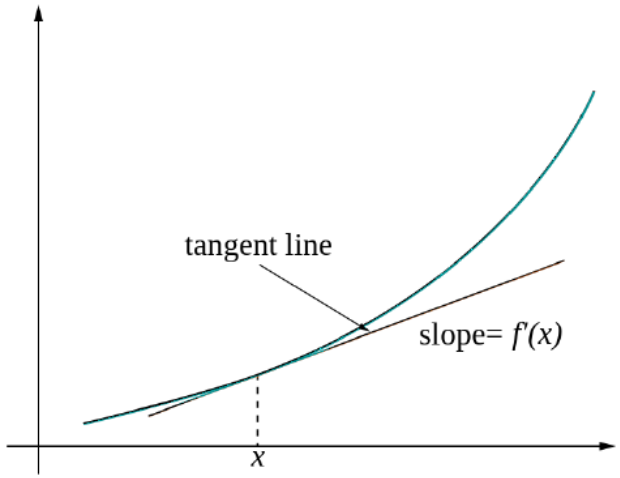
Follow the gradient direction to find an optima solution

# Review: **Computing gradients**

- **Gradient in 1D**

the derivative of a function $y = f(x)$

$$f'(x) = \frac{dy}{dx} = \frac{\text{change in } y}{\text{change in } x}$$




notation: $\frac{\partial f(x)}{\partial x}$, partial derivative w.r.t the variable $x$

# Review: **Computing gradients**

- **Computing gradients in general cases**

  Gradient of a function w.r.t $d$-dim inputs: Assume that $y = f(\beta)$, where $y$ is scalar, and $\beta$ is a $d$-dim vector. It is defined as

$$\frac{\partial f(\beta)}{\partial \beta} = \begin{bmatrix} \frac{\partial f(\beta)}{\partial \beta_1} \\ \frac{\partial f(\beta)}{\partial \beta_2} \\ \cdots \\ \frac{\partial f(\beta)}{\partial \beta_d} \end{bmatrix} \in \mathbb{R}^d$$

  **Ex.**:

$$f(\beta) = \begin{bmatrix} \beta_1^2 + \beta_2 \end{bmatrix}$$

  where $\beta = [\beta_1, \beta_2]$ then

$$\frac{\partial f(\beta)}{\partial \beta} = \begin{bmatrix} 2\beta_1 \\ 1 \end{bmatrix} \quad \begin{matrix} \leftarrow \frac{\partial f(\beta)}{\partial \beta_1} \\ \leftarrow \frac{\partial f(\beta)}{\partial \beta_2} \end{matrix}$$

# Review: gradient descent algorithm

- **Problem** (minimizing or finding a minimum):

$$\min_{\beta} f(\beta)$$

where we can evaluate $f(\beta)$ and $\nabla f(\beta)$ for any $\beta \in \mathbb{R}^d$

- Plain gradient descent: iterative steps in the direction $-\nabla f(\beta)$ (called **descent direction**).

---

**Input:** initialize $\beta \in \mathbb{R}^d$, function $\nabla f(\beta)$, stepsize $\alpha$, tolerance $\theta$
**Output:** $\beta$

1: **repeat**
2: $\quad \beta \leftarrow \beta - \alpha \nabla f(\beta)$
3: **until** $|\Delta\beta| < \theta$ [perhaps for 10 iterations in sequence]

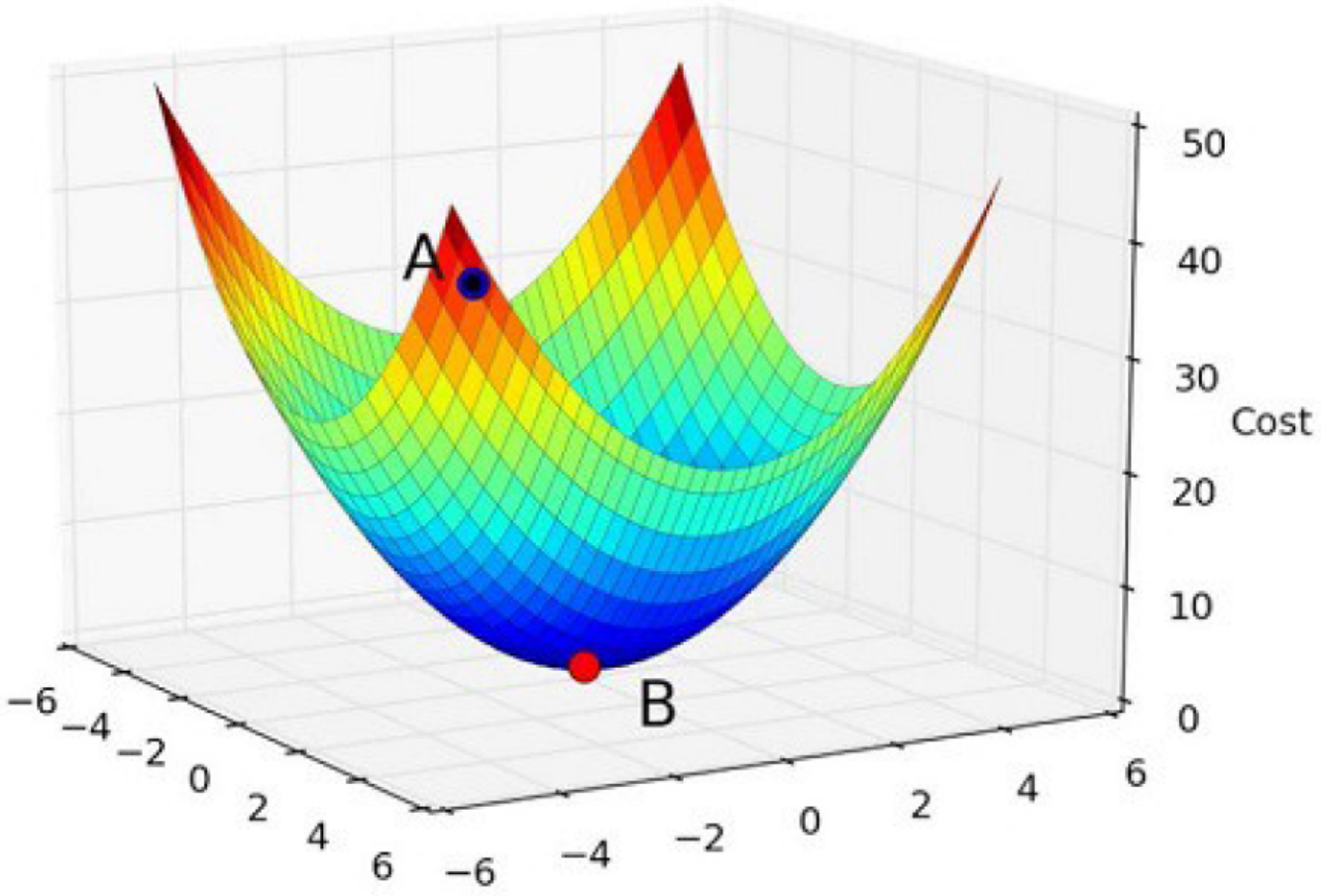
---

$= |\alpha \nabla f(\beta)|$

STOP (**CONVERGENCE**) When the change/update is negligible

# Gradient descent algorithm: working example

minimizing a function $f(\beta) = x_1^2 + {x_2}^2$ where $\beta = [x_1, x_2]$ w.r.t $\beta$



**step 1**: computing the gradient vector

$$\nabla_\beta f(\beta) = \begin{pmatrix} \frac{\partial f(\beta)}{\partial x_1} \\ \frac{\partial f(\beta)}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 2x_1 \\ 2x_2 \end{pmatrix}$$

**step 2**: initialize a starting point (random guess) $\beta = [1, 0]$, init a step-size $\alpha = 0.1$

**step 3**: iterative algorithm

– iteration 1: $\beta = \beta - 0.1 \cdot \begin{pmatrix} 2 \cdot 1 \\ 2 \cdot 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} - 0.1 \begin{pmatrix} 2 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.8 \\ 0.0 \end{pmatrix}$

check $f([1, 0]) = 1^2 + 0^0 = 1$ vs. $f([0.8, 0]) = 0.8^2 + 0^2 = 0.64$

**step 1**: computing the gradient vector

$$\nabla_\beta f(\beta) = \begin{pmatrix} \frac{\partial f(\beta)}{\partial x_1} \\ \frac{\partial f(\beta)}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 2x_1 \\ 2x_2 \end{pmatrix}$$

**step 2**: initialize a starting point (random guess) $\beta = [1, 0]$, init a step-size $\alpha = 0.1$

**step 3**: iterative algorithm

– iteration 1: $\beta = \beta - 0.1 \cdot \begin{pmatrix} 2 \cdot 1 \\ 2 \cdot 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} - 0.1 \begin{pmatrix} 2 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.8 \\ 0.0 \end{pmatrix}$

check $f([1, 0]) = 1^2 + 0^0 = 1$ vs. $f([0.8, 0]) = 0.8^2 + 0^2 = 0.64$

(WE ARE GETTING SMALLER)

**step 1**: computing the gradient vector

$$\nabla_\beta f(\beta) = \begin{pmatrix} \frac{\partial f(\beta)}{\partial x_1} \\ \frac{\partial f(\beta)}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 2x_1 \\ 2x_2 \end{pmatrix}$$

**step 2**: initialize a starting point (random guess) $\beta = [1, 0]$, init a step-size $\alpha = 0.1$

**step 3**: iterative algorithm

– iteration 1: $\beta = \beta - 0.1 \cdot \begin{pmatrix} 2 \cdot 1 \\ 2 \cdot 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} - 0.1 \begin{pmatrix} 2 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.8 \\ 0.0 \end{pmatrix}$

– iteration 2: $\beta = \beta - 0.1 \cdot \begin{pmatrix} 2 \cdot 0.8 \\ 2 \cdot 0 \end{pmatrix} = \begin{pmatrix} 0.8 \\ 0 \end{pmatrix} - 0.1 \begin{pmatrix} 1.6 \\ 0.0 \end{pmatrix} = \begin{pmatrix} 0.64 \\ 0 \end{pmatrix}$

check $f([0.64, 0]) = 0.64^2 + 0^2 = 0.4096$ (SMALLER)

**step 1**: computing the gradient vector

$$\nabla_\beta f(\beta) = \begin{pmatrix} \frac{\partial f(\beta)}{\partial x_1} \\ \frac{\partial f(\beta)}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 2x_1 \\ 2x_2 \end{pmatrix}$$

**step 2**: initialize a starting point (random guess) $\beta = [1, 0]$, init a step-size $\alpha = 0.1$

**step 3**: iterative algorithm

– iteration 1: $\beta = \beta - 0.1 \cdot \begin{pmatrix} 2 \cdot 1 \\ 2 \cdot 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} - 0.1 \begin{pmatrix} 2 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.8 \\ 0.0 \end{pmatrix}$

– iteration 2: $\beta = \beta - 0.1 \cdot \begin{pmatrix} 2 \cdot 0.8 \\ 2 \cdot 0 \end{pmatrix} = \begin{pmatrix} 0.8 \\ 0 \end{pmatrix} - 0.1 \begin{pmatrix} 1.6 \\ 0.0 \end{pmatrix} = \begin{pmatrix} 0.64 \\ 0 \end{pmatrix}$

– iteration $k$

– ...

– stop when $\beta$ (or $f(\beta)$) does not change

# Logistic regression: Optimal parameters

- **Find $\beta^*$ that minimizes:**

$$L^{\text{logistic}}(\beta) = -\sum_{i=1}^{n} \left[ y_i \log p(y_i \mid x_i) + (1 - y_i) \log[1 - p(y_i \mid x_i)] \right] + \lambda \|\beta\|^2$$

- Using gradient-descent optimization method
  - Taking gradients:

$$\frac{\partial L^{\text{logistic}}(\beta)}{\partial \beta}$$

# Logistic regression: Computing the loss's gradient

$$\frac{\partial L^{\text{logistic}}(\beta)}{\partial \beta}$$

– substitute the definition of $p(y = 1|x_i)$

$$p_i = p(y = 1|x_i) = \frac{1}{1 + e^{-f(x_i, y=1)}} = \frac{1}{1 + e^{-\phi(x_i)^\top \beta}}$$

– we obtain a loss function in parameters

$$L^{\text{logistic}}(\beta) = -\sum_{i=1}^{n} \left[ y_i \log \frac{1}{1 + e^{-\phi(x)^\top \beta}} + (1 - y_i) \log[1 - \frac{1}{1 + e^{-\phi(x)^\top \beta}}] \right] + \lambda \|\beta\|^2$$

– the gradient of the loss is

$$\frac{\partial L^{\text{logistic}}(\beta)}{\partial \beta} = -\sum_{i=1}^{n} \left[ y_i \frac{\partial}{\partial \beta} \log \frac{1}{1 + e^{-\phi(x)^\top \beta}} + (1 - y_i) \frac{\partial}{\partial \beta} \log[1 - \frac{1}{1 + e^{-\phi(x)^\top \beta}}] \right] + \frac{\partial}{\partial \beta} \lambda \|\beta\|^2$$

# Logistic regression: Computing the loss's gradient

– the gradient of the loss is

$$\frac{\partial L^{\text{logistic}}(\beta)}{\partial \beta} = -\sum_{i=1}^{n} \left[ y_i \frac{\partial}{\partial \beta} \log \frac{1}{1+e^{-\phi(x)^\top \beta}} + (1-y_i) \frac{\partial}{\partial \beta} \log[1 - \frac{1}{1+e^{-\phi(x)^\top \beta}}] \right] + \frac{\partial}{\partial \beta} \lambda \|\beta\|^2$$

$$= \sum_{i=1}^{n} (p_i - y_i)\phi(x_i) + 2\lambda I \beta = X^\top (p - y) + 2\lambda I \beta$$

where $p = [p_1, p_2, \cdots, p_n]^\top$ in which $p_i := p(y = 1 \mid x_i)$

$$X = \begin{pmatrix} \phi(x_1)^\top \\ \vdots \\ \phi(x_n)^\top \end{pmatrix} \in \mathbb{R}^{n \times k}, \quad y = [y_1, y_2, \ldots, y_n]^\top$$

where $I$ is an identity matrix, $\lambda$ is scalar.

$$I = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

# Logistic regression: The algorithm

- setting: a data set $\mathcal{D} = \{x_i, y_i\}_{i=1}^n$, where $x_i \in \mathbb{R}^d, y_i \in \{0, 1\}$
- setting: features $\phi(x) \in \mathbb{R}^k$
- using gradient descent to find optimum $\beta$ **iteratively**

---

**Input:** initial $\beta \in \mathbb{R}^k$, stepsize $\alpha$, tolerance $\theta$
**Output:** $\beta$

1: **repeat**
2: $$\beta \leftarrow \beta - \underbrace{\alpha}_{\text{step-size}} \underbrace{\left(X^\top(p - y) + 2\lambda I\beta\right)}_{\text{the gradient}}$$
3: **until** $|\Delta\beta| < \theta$ [perhaps for 10 iterations in sequence]

---

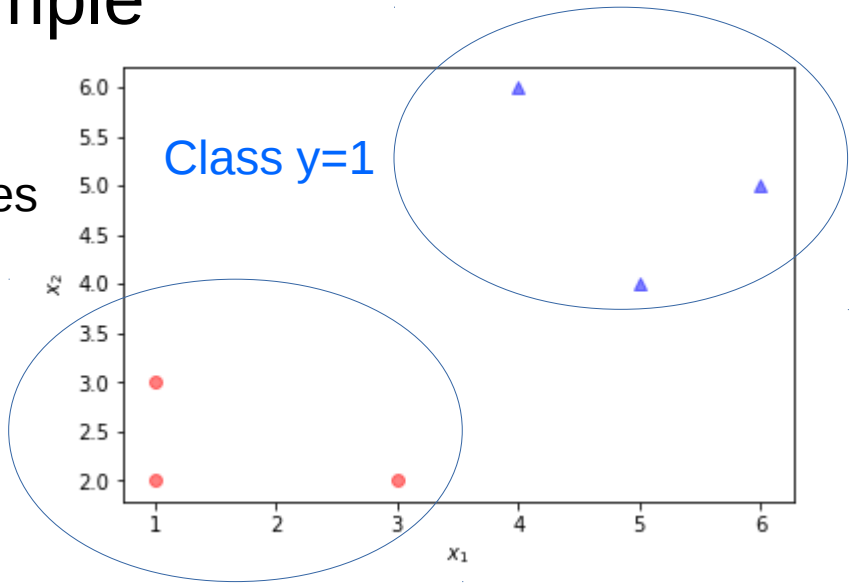
where

$$p = [p_1, p_2, \cdots, p_n]^\top \text{ in which } p_i := p(y = 1 \mid x_i) = \frac{1}{1 + e^{-\phi(x_i)^\top \beta}}$$

$$X = \begin{pmatrix} \phi(x_1)^\top \\ \vdots \\ \phi(x_n)^\top \end{pmatrix} \in \mathbb{R}^{n \times k}$$

# Logistic regression: Example

Given n=6 data points of two classes




randomly =0.1 =0.01

- using gradient descent to find optimum $\beta$ **iteratively**

**Input:** initial $\beta \in \mathbb{R}^k$, stepsize $\alpha$, tolerance $\theta$
**Output:** $\beta$
1: **repeat**
2: $\beta \leftarrow \beta - \underbrace{\alpha}_{\text{step-size}} \underbrace{\left(X^\top(p-y) + 2\lambda I\beta\right)}_{\text{the gradient}}$
3: **until** $|\Delta\beta| < \theta$ [perhaps for 10 iterations in sequence]

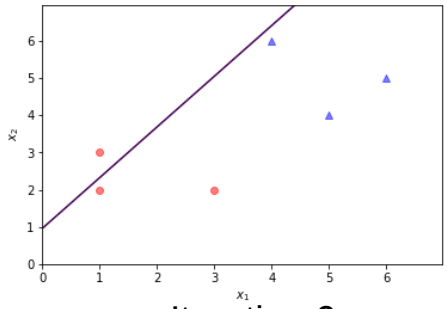
where

$p = [p_1, p_2, \cdots, p_n]^\top$ in which $p_i := p(y = 1 \mid x_i) = \dfrac{1}{1 + e^{-\phi(x_i)^\top \beta}}$

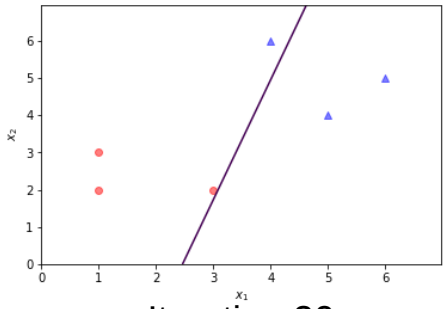
$$X = \begin{pmatrix} \phi(x_1)^\top \\ \vdots \\ \phi(x_n)^\top \end{pmatrix} \in \mathbb{R}^{n \times k} =$$

```
[[1 1 2]
 [1 3 2]
 [1 1 3]
 [1 5 4]
 [1 6 5]
 [1 4 6]]
```

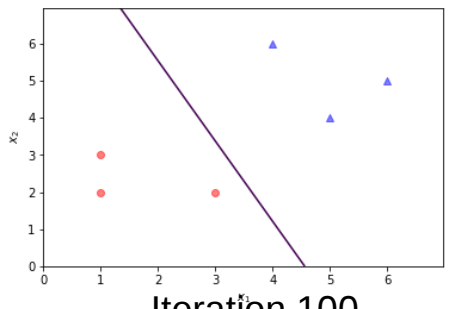
Linear feature: $\phi(x) = [1, x_1, x_2]$



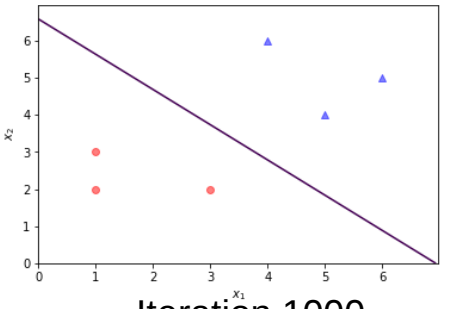
Iteration 0



Iteration 20



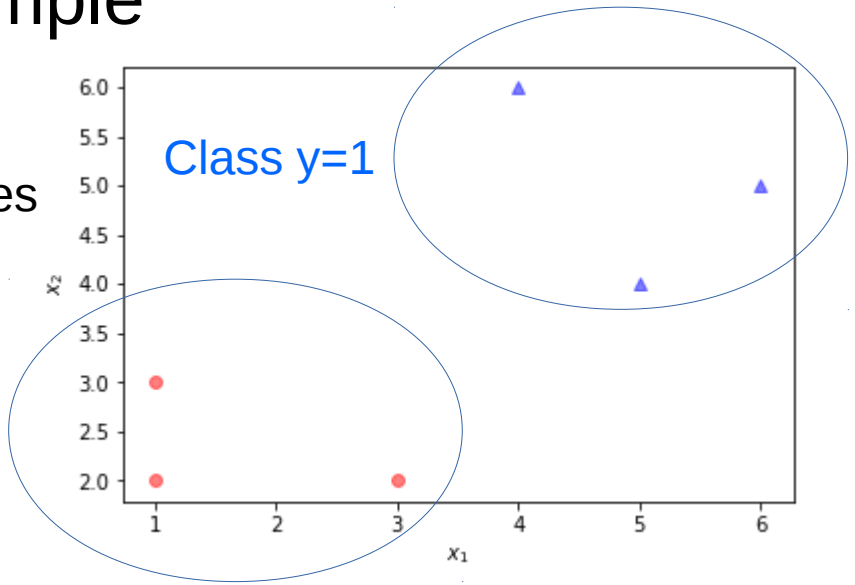
Iteration 100



Iteration 1000

# Logistic regression: Example

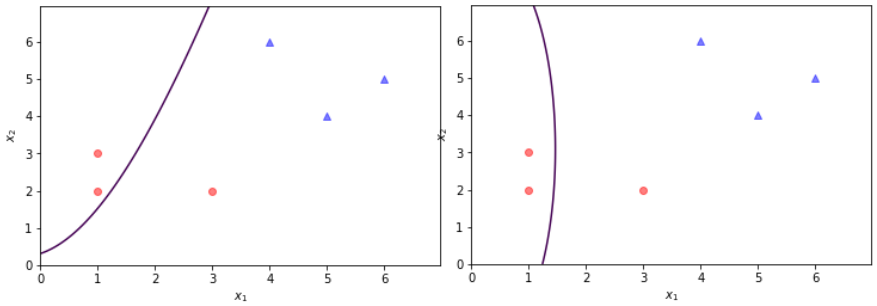
Given n=6 data points of two classes




randomly =0.1 =0.01

- using gradient descent to find optimum $\beta$ **iteratively**

**Input:** initial $\beta \in \mathbb{R}^k$, stepsize $\alpha$, tolerance $\theta$
**Output:** $\beta$
1: **repeat**
2: $\beta \leftarrow \beta - \underbrace{\alpha}_{\text{step-size}} \underbrace{\left(X^\top(p-y) + 2\lambda I\beta\right)}_{\text{the gradient}}$
3: **until** $|\Delta\beta| < \theta$ [perhaps for 10 iterations in sequence]

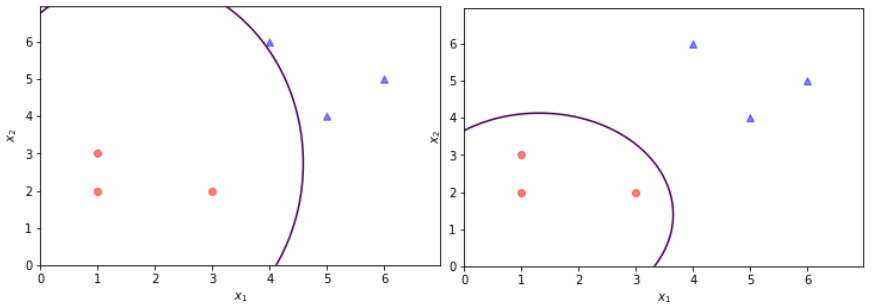



where

$$p = [p_1, p_2, \cdots, p_n]^\top \text{ in which } p_i := p(y=1 \mid x_i) = \frac{1}{1+e^{-\phi(x_i)^\top\beta}}$$

$$X = \begin{pmatrix} \phi(x_1)^\top \\ \vdots \\ \phi(x_n)^\top \end{pmatrix} \in \mathbb{R}^{n\times k} =$$

```
[[ 1  1  2  1  4]
 [ 1  3  2  9  4]
 [ 1  1  3  1  9]
 [ 1  5  4 25 16]
 [ 1  6  5 36 25]
 [ 1  4  6 16 36]]
```



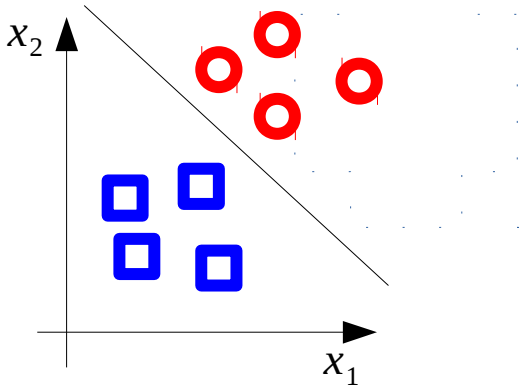

Quadratic feature: $\phi(x) = [1, x_1, x_2, x_1^2, x_2^2]$

# Outline

- The simplest approach: k-nearest neighbour
- Discriminative function
- Logistic regression for binary classification
- **Multi-class classification**

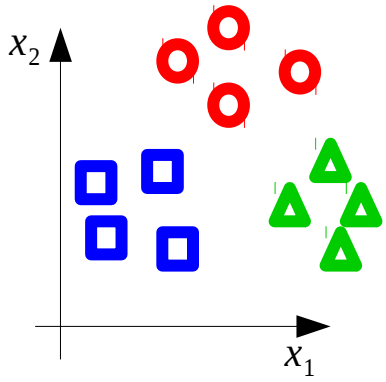
# Logistic regression: multi-class classification

Binary classification



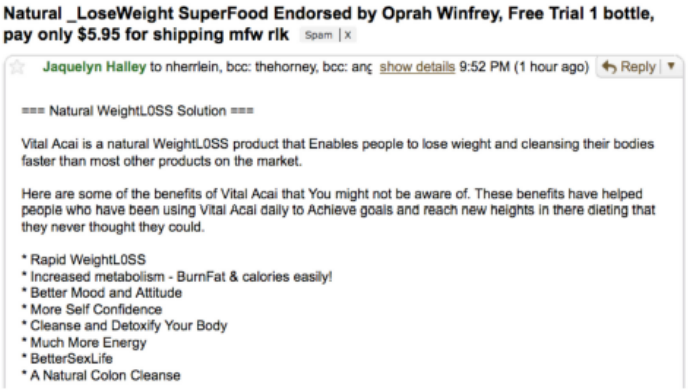

Output y={ , }

Multi-class classification




Output y ={ , , }

Input x=email



**Natural _LoseWeight SuperFood Endorsed by Oprah Winfrey, Free Trial 1 bottle, pay only $5.95 for shipping mfw rlk** Spam | x

Jaquelyn Halley to nherrlein, bcc: thehorney, bcc: anç show details 9:52 PM (1 hour ago) Reply

=== Natural WeightL0SS Solution ===

Vital Acai is a natural WeightL0SS product that Enables people to lose wieght and cleansing their bodies faster than most other products on the market.

Here are some of the benefits of Vital Acai that You might not be aware of. These benefits have helped people who have been using Vital Acai daily to Achieve goals and reach new heights in there dieting that they never thought they could.

* Rapid WeightL0SS
* Increased metabolism - BurnFat & calories easily!
* Better Mood and Attitude
* More Self Confidence
* Cleanse and Detoxify Your Body
* Much More Energy
* BetterSexLife
* A Natural Colon Cleanse

Binary classification:
output y={**spam**, **no spam**}

Multi-class classification:
output y={**spam**, **Normal, Social, Promotion, Family**}

# Logistic regression: multi-class classification

- Given a training dataset of ***n* instances** of input-output

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$ ← $y_i$ Is {0,1,…, M}

- Find optimal parameter $\beta^*$ such that

$$p(y=m|x;\beta)=\frac{e^{\beta^T \phi(x, y=m)}}{\sum_{y'=1}^{M} e^{\beta^T \phi(x, y')}}$$ is high for all $(x_i, y_i), i=[1,\ldots,n]$

**NOTE: we only model $p(y=m|x;\beta)$, the probability of y=m given x**

- **Negative log likelihood cost function**

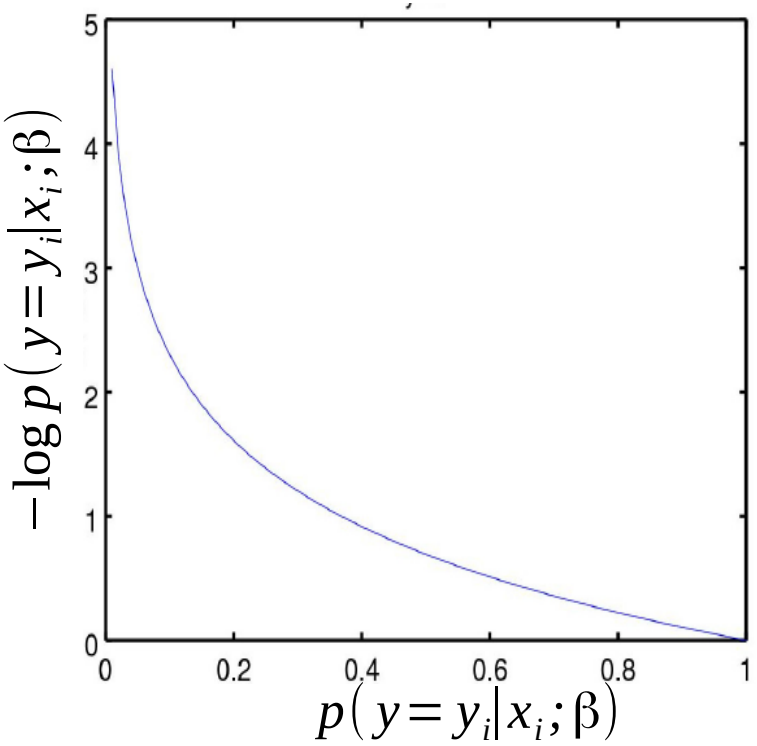
$$l(x_i, y_i) = -\log p(y=y_i|x_i;\beta)$$

# Multi-class classification: Cost function's interpretation

- **Negative log likelihood cost function**

$$l(x_i, y_i) = -\log p(y = y_i | x_i; \beta)$$

Minimum cost = maximum $p(y = y_i | x_i; \beta)$ , then prediction $y = y_i$ is most likely



# Multi-class classification: Cost function

- Find optimal parameter $\beta^*$ such that

$$L^{\text{logistic}}(\beta) = -\sum_{i=1}^{n} \log p(y = y_i \mid x_i; \beta) + \lambda \|\beta\|^2$$

- Using gradient-descent optimization method
  - Taking gradients: $\frac{\partial L^{\text{logistic}}(\beta)}{\partial \beta}$
  - Applying the same iterative update rule for $\beta$

# Summary

- We covered:
  - Discriminative function
  - Logistic regression for binary and multi-class classification: using gradient descent