CSC4007 Advanced Machine Learning

Lesson 04: Classification – Logistic Regression

by Vien Ngo
EEECS / ECIT / DSSC

Outline

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

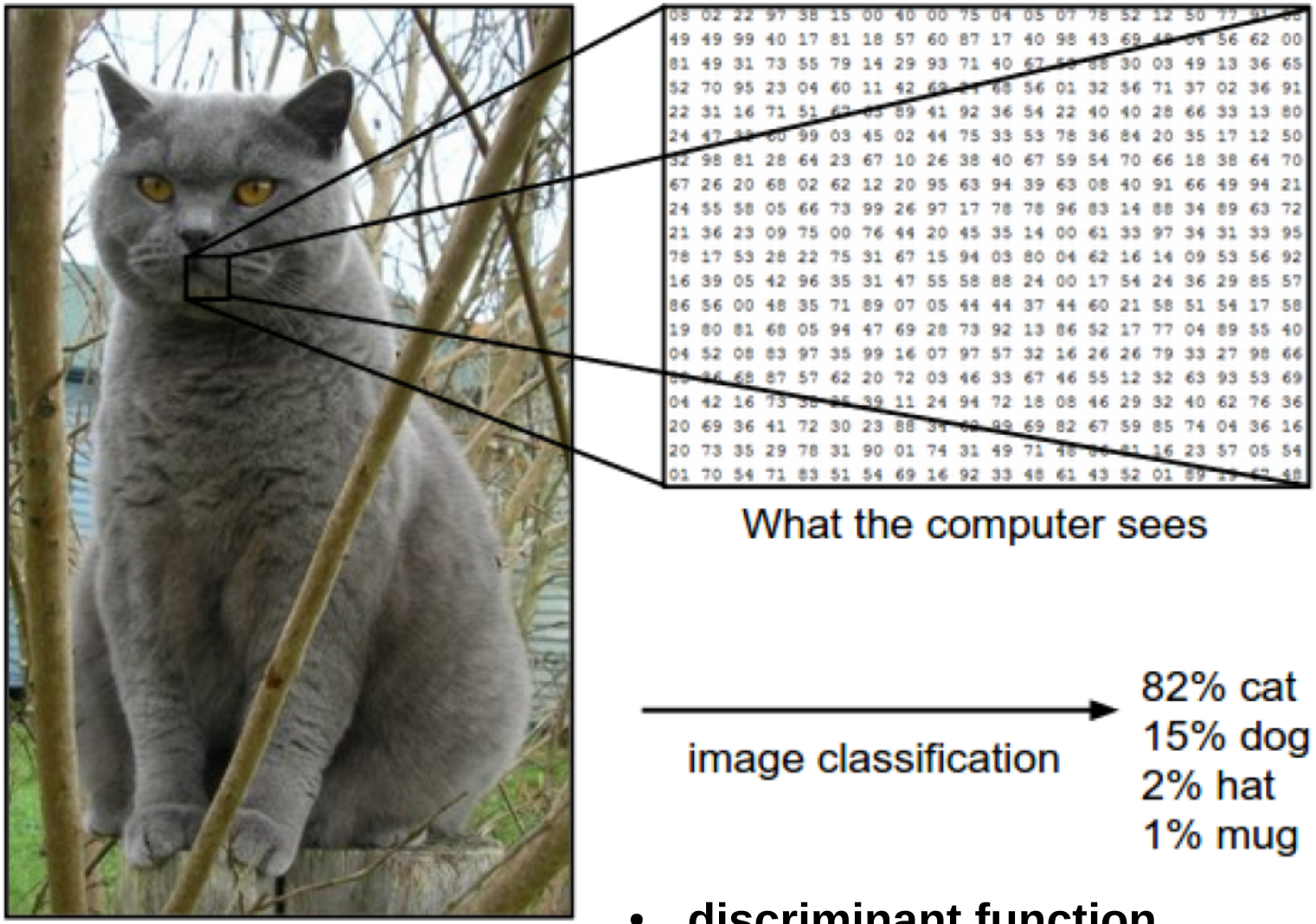
Logistic Regression

- Discriminant function predicts the class
- Instead of predicting class, logistic regression gives the probability of the instance being that class
  - Classificatiton using likelihood function

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

  - **Logistic regression**: transform an open-ranged discriminant function into a probabilistic function.
- **Don't be confused**: Logistic regression is a classification algorithm

- example of image classification as probability function



- **discriminant function**

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

$f(x, cat)$ $f(x, dog)$ $f(x,hat)$ $f(x, mug)$

Probability: The basics

$$p(event) + p(\neg event) = 1$$

Notation: not

e.g.: coin tossing (binary outcomes)

$$p(head) + p(\neg head) = p(head) + p(tail) = 1$$

e.g.: image classification (4 classes: cat, dog, monkey, mug)

$$p(cat|image) + p(\neg cat|image) = 1$$

$$p(cat|image) + p(dog|image) + p(mug|image) + p(monkey|image) = 1$$

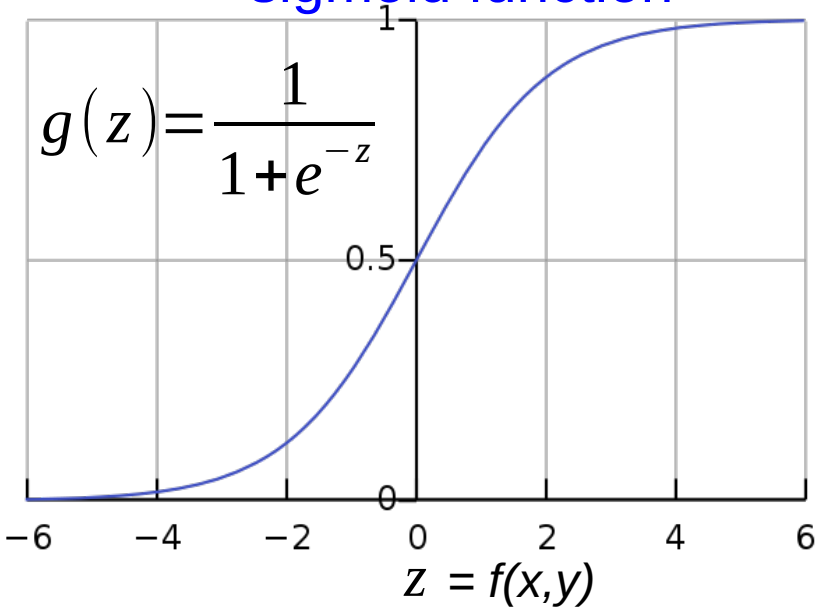
Logistic regression for binary classification

- Classificatiton algorithm for binary cases $y \in \{0,1\}$
- We want our classifier to output values between 0 and 1
- Using a sigmoid function to transform discriminant function

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

discriminant function

sigmoid function



$0 \leqslant g(z) \leqslant 1$ (z is real value) is the **sigmoid function**, or the **logistic function**

- Asymptotes at 0 and 1

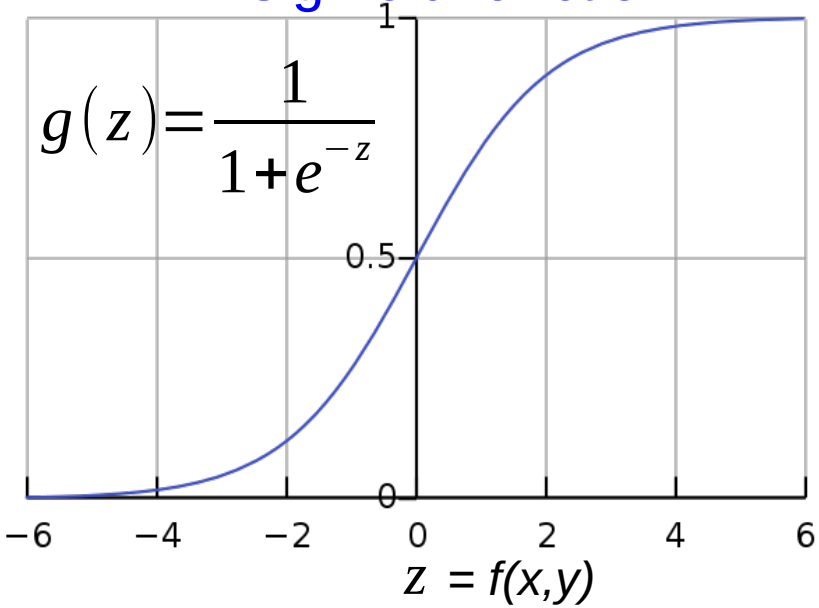
e=2.718281828459045 is an Euler's number

Logistic regression for binary classification

- Classificatiton algorithm for binary cases $y \in \{0,1\}$
- We want our classifier to output values between 0 and 1
- Using a sigmoid function to transform discriminant function

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

sigmoid function



$0 \leqslant g(z) \leqslant 1$ (z is real value) is the **sigmoid function**, or the **logistic function**

- Asymptotes at 0 and 1
- f(x,y)=0 (classifier) results in p(y|x) = 0.5 (boundary)

e=2.718281828459045 is an Euler's number

Logistic regression for binary classification

- Classificatiton using likehood probability (**of *y* being equal to 1**)

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

The probability of being y=1 given x

- Our model assumed to be the likelihood of y equal to 1 given x:

$$p(y|x) = p(y=1|x)$$

- Using a sigmoid function to transform discriminant function

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

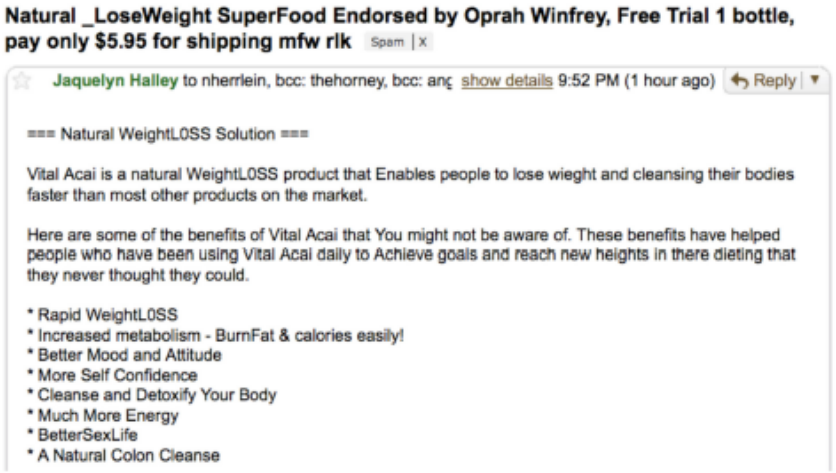
Linear discriminant function

Logistic regression: Interpretation

- Parametric likehood function

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

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


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

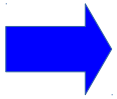
Jaquelyn Halley to nherrlein, bcc: thehorney, bcc: anç show details 9:52 PM (1 hour ago) Reply

=== 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



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

At current value of β

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

With probability 0.9: x is a spam email

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

With probability 0.1, x is not a spam email

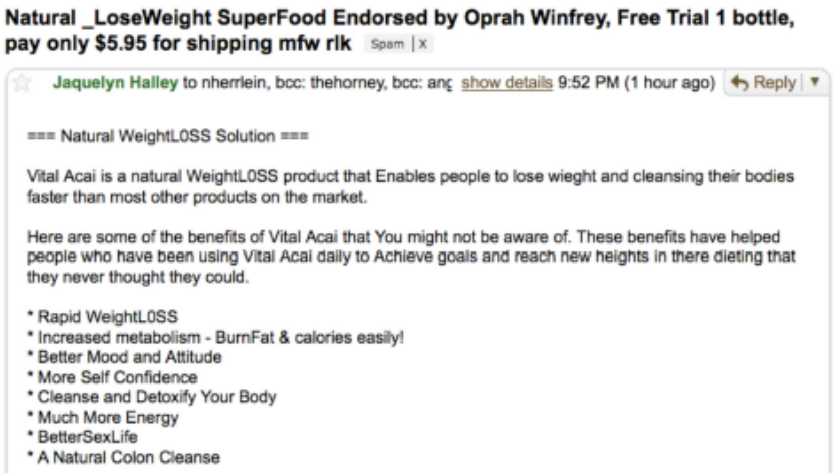
Logistic regression: Interpretation

- 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



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

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

At current value of β

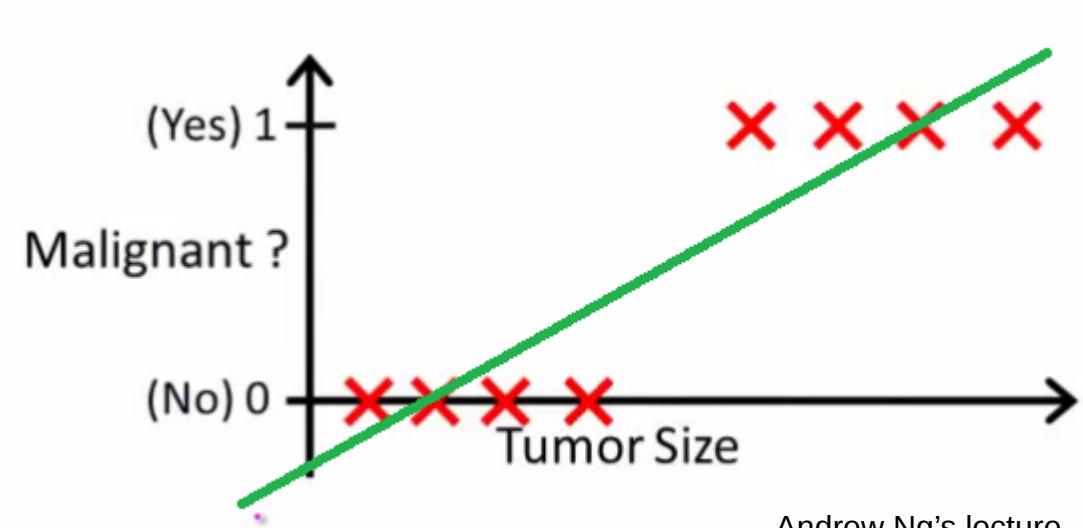
$$p(spam|x;\beta) = 0.9$$

Predict: y = spam

Logistic regression: Interpretation

- Worked example: Cancer diagnosis from tumor size
  - Tumour size (input x) vs. malignancy (output y = 0 or 1)
  - e.g. feature vector (linear): $\phi(x) = [1, tumorsize]$
  - *Predictive model: Given a tumor size, the probability of being cancer is*

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



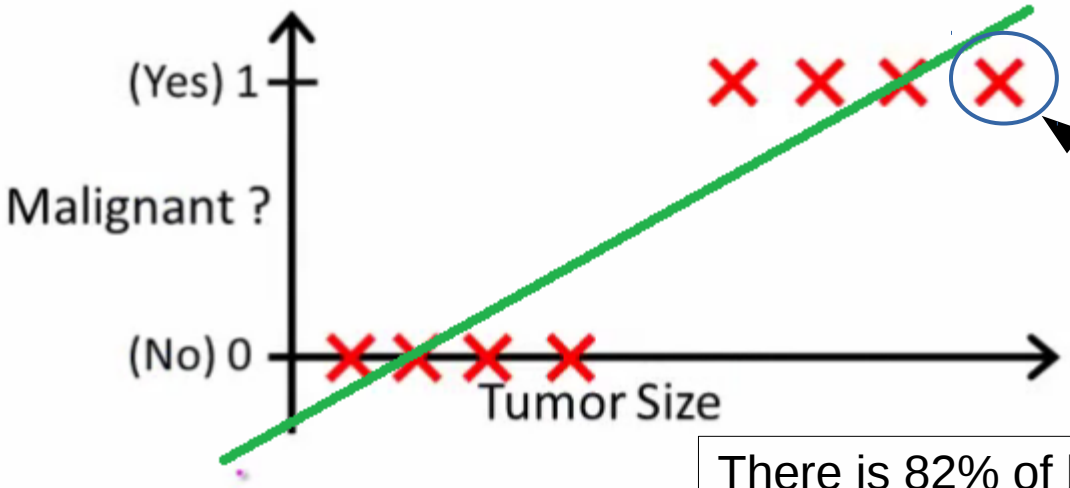
Model parameters: $\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$

Andrew Ng's lecture

Logistic regression: Interpretation

- Worked example: Cancer diagnosis from tumor size
  - Tumour size vs. malignancy (0 or 1)
  - Feature vector (linear): $\phi(x) = [1, tumorsize]$
  - *Predictive model: Given a tumor size, the probability of being cancer is*

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



e.g. $\beta = \begin{bmatrix} 0.5 \\ 0.1 \end{bmatrix}$

Size = 10(cm) $\phi(x) = [1, 10]$

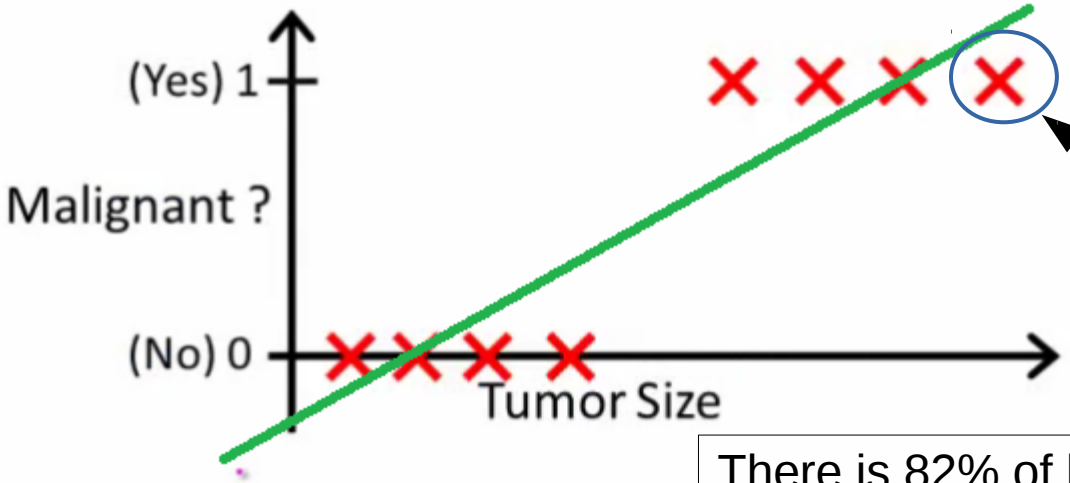
$$p(y|x;\beta) \approx 0.82$$

There is 82% of being cancer if the tumor size is 10 cm

Logistic regression: Interpretation

- Worked example: Cancer diagnosis from tumor size
  - Tumour size vs malignancy (0 or 1)
  - Feature vector (linear): $\phi(x) = [1, tumorsize]$
  - *Predictive model: Given a tumor size, the probability of being cancer is*

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



e.g. $\beta = \begin{bmatrix} 0.5 \\ 0.1 \end{bmatrix}$

Size = 10(cm) $\phi(x) = [1, 10]$

$p(y|x;\beta) \approx 0.82$

There is 82% of being cancer if the tumor size is 10 cm

Note that: $p(y=0|x;\beta) = 1 - p(y|x;\beta) = 1 - 0.82 = 0.18$

There is 18% of being NO cancer if the tumor size is 10 cm

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	?

- 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

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

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 $p(y = 0|x_1) = 1 - p(y = 1|x_1) = 0.19$

It is with a probability of 0.19 I will NOT like the movie **lord of the ring**

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	?

- 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

$$p(y = 1|x_1) = \frac{1}{1 + e^{-f(x_1)}} = \frac{1}{1 + e^{-\phi(x_1)^T \beta}} = \frac{1}{1 + e^{-1\times 1 - 1\times 0.5 - 5\times 0}} = 0.81$$

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: Optimum 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 optimum parameter β^* such that for each $i=[1,\ldots,n]$

$$p(y_i|x_i;\beta)=\frac{1}{1+e^{-\beta^T \phi(x_i)}}$$

is high (optimum at =1) if $y_i=1$

is low (optimum at = 0) if $y_i=0$

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

- When $p(y_i|x_i;\beta)$ is **high** that means the event of $y_i=1$ is likely to happen (this matches the given data $y_i=1$)

- When $p(y_i|x_i;\beta)$ is **low** that means the event of $y_i=1$ is NOT likely to happen (this matches the given data $y_i=0$)

Logistic regression: Optimum 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 β^* such that minimizes the **Negative log likelihood cost**
- **Negative log likelihood cost function: for each** $(x_i, y_i), i = [1, \ldots, n]$

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

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

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

Logistic regression: Optimum parameters

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

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

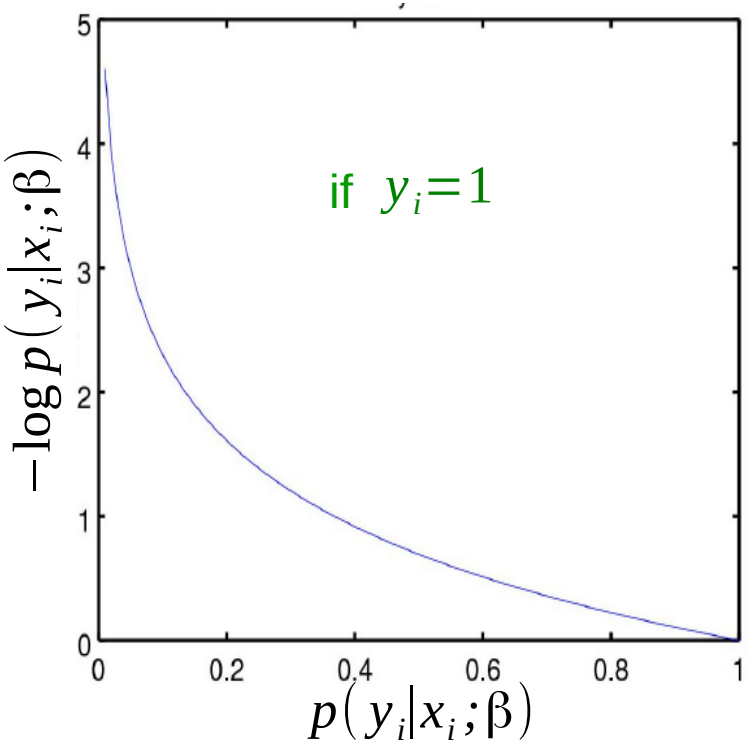
$$l(x_i, y_i) = -\log(1 - p(y_i | x_i; \beta)) \quad \text{if} \quad y_i = 0$$

Case 1: if $y_i = 1$

Minimum cost when $p(y_i = 1 | x_i; \beta) = 1$
That means prediction $y_i = 1$ is most likely (GOOD)

Maximum cost when $p(y_i = 1 | x_i; \beta) = 0$
That means prediction $y_i = 1$ is less likely
So, it contradicts the data (incur infinite cost).

Cost decreases when $p(y_i = 1 | x_i; \beta)$ increases
Then prediction $y_i = 1$ becomes more likely



Logistic regression: Optimum parameters

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

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

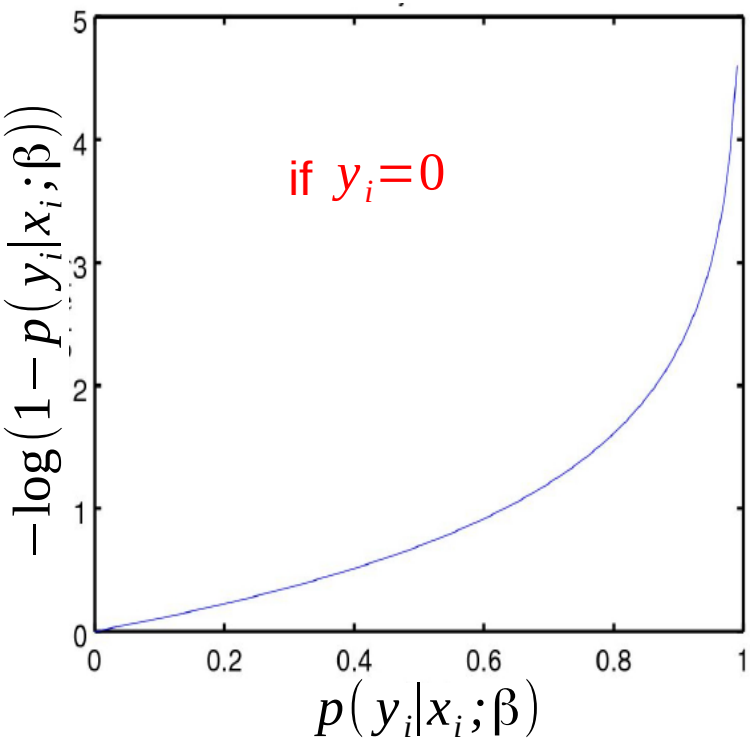
$$l(x_i, y_i) = -\log(1 - p(y_i|x_i;\beta)) \quad \text{if} \quad y_i = 0$$

Case 2: if $y_i = 0$

Minimum cost when $p(y_i = 1|x_i;\beta) = 0$
That means prediction $y_i = 1$ is least likely
So, $y_i = 0$ most likely (GOOD)

Maximum cost when $p(y_i = 1|x_i;\beta) = 1$
That means prediction $y_i = 1$ is most likely
So, $y_i = 0$ is least likely (incur infinite cost)

Cost decrease when $p(y_i = 1|x_i;\beta)$ decreases
Then prediction $y_i = 1$ becomes less likely
So, $y_i = 0$ becomes more likely



Logistic regression: Optimal parameters

- **Negative log likelihood cost function: Find β^* 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 β^* that minimizes:**

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

  - This cost function can be derived from statistics using the principle of maximum likelihood estimation
  - The cost function is convex → using gradient descent can converge to a global minimum

- Using gradient-descent optimization method
  - Taking gradients:

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

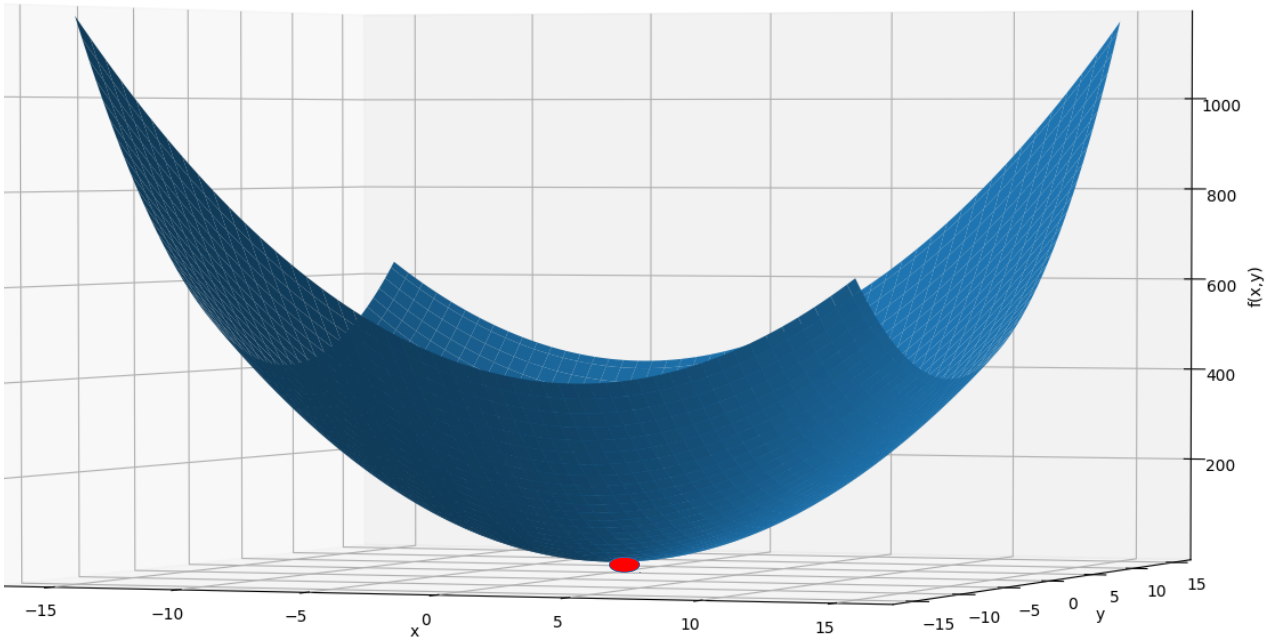
Review: gradient descent algorithm

- 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 $\beta^* = [x^*, y^*]$

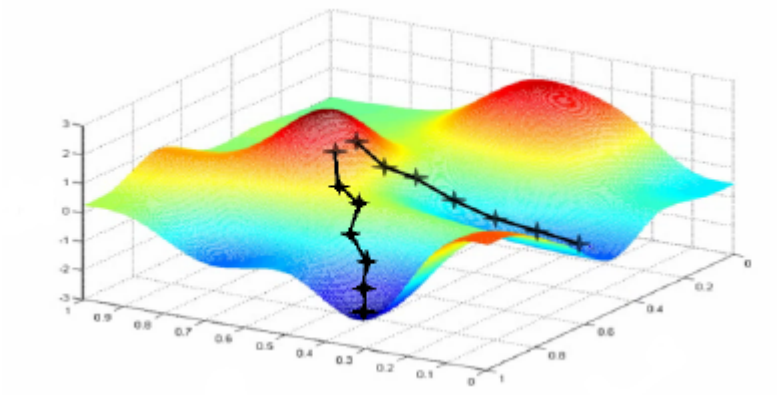


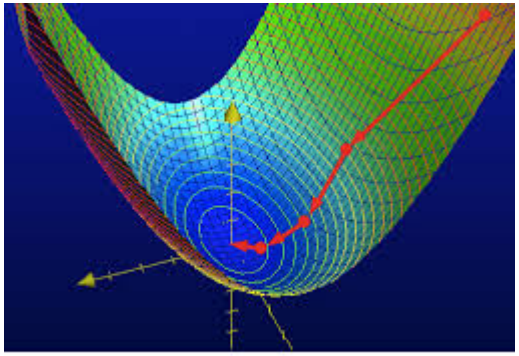
Review: gradient descent algorithm

- **Here look at advanced concepts for minimizing the cost function for logistic regression**
  - Good for large machine learning problems (e.g. huge feature set)
- **what is gradient descent actually doing?**
  - We have some cost function $f(\beta)$ and we want to minimize it
  - We need to write code which can take β as input and compute the following:

Gradient of a function w.r.t its parameters $\nabla f(\beta)$

Where partial derivative w.r.t to a parameter variable β_i defined as: $\frac{\partial f(\beta)}{\partial \beta_i}$





Follow the gradient direction to find an optima solution

Andrew Ng's lecture

Review: gradient descent algorithm

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

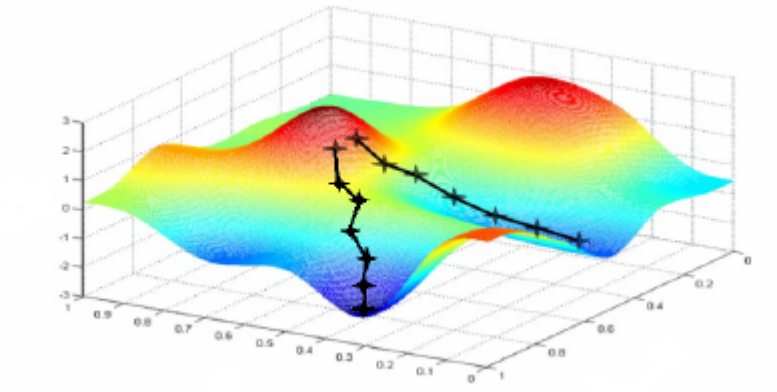
  – e.g: find x^* and y^* that minimizes $f(x, y) = 2 * x^2 + 1.2 * x * y + 2 * y^2$

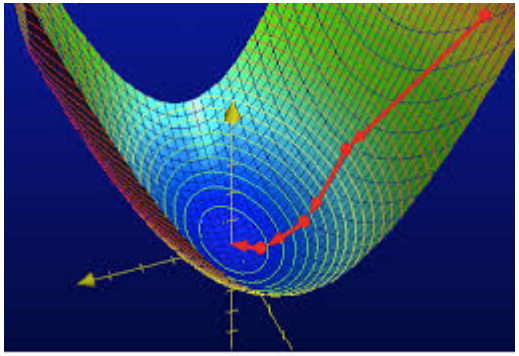
  –assumption: we have the gradient of $f(\beta)$

$$\nabla f(\beta) = \left[\frac{\partial f(\beta)}{\partial \beta_i}\right]^\top \in \mathbb{R}^d, i \in [1, \ldots, d]$$

Notation: Gradient of a function w.r.t its parameters $\nabla f(\beta)$

Partial derivative w.r.t to a parameter variable β_i $\frac{\partial f(\beta)}{\partial \beta_i}$





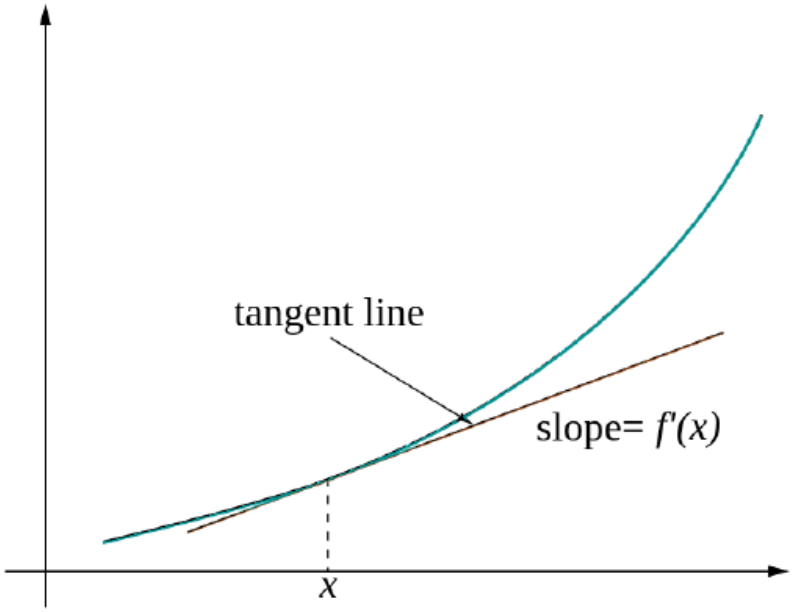
Follow the gradient direction to find an optima solution

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

$$f(x) = x^3 + 3x^2 + 4x + 9 \quad \text{then} \quad f'(x) = 3x^2 + 6x + 4$$

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 β is a d-dim vector. It is defined as

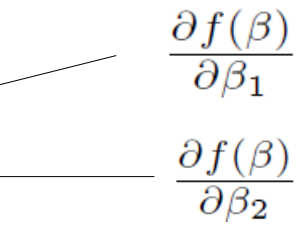
$$\frac{\partial f(\beta)}{\partial \beta} = \begin{bmatrix} \frac{\partial f(\beta)}{\partial \beta_1} \\ \frac{\partial f(\beta)}{\partial \beta_2} \\ \cdots \\ \frac{\partial f(\beta)}{\partial \beta_d} \end{bmatrix} \in \mathbb{R}^d$$

  - **Ex.**:

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

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

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



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 α, tolerance θ
Output: β

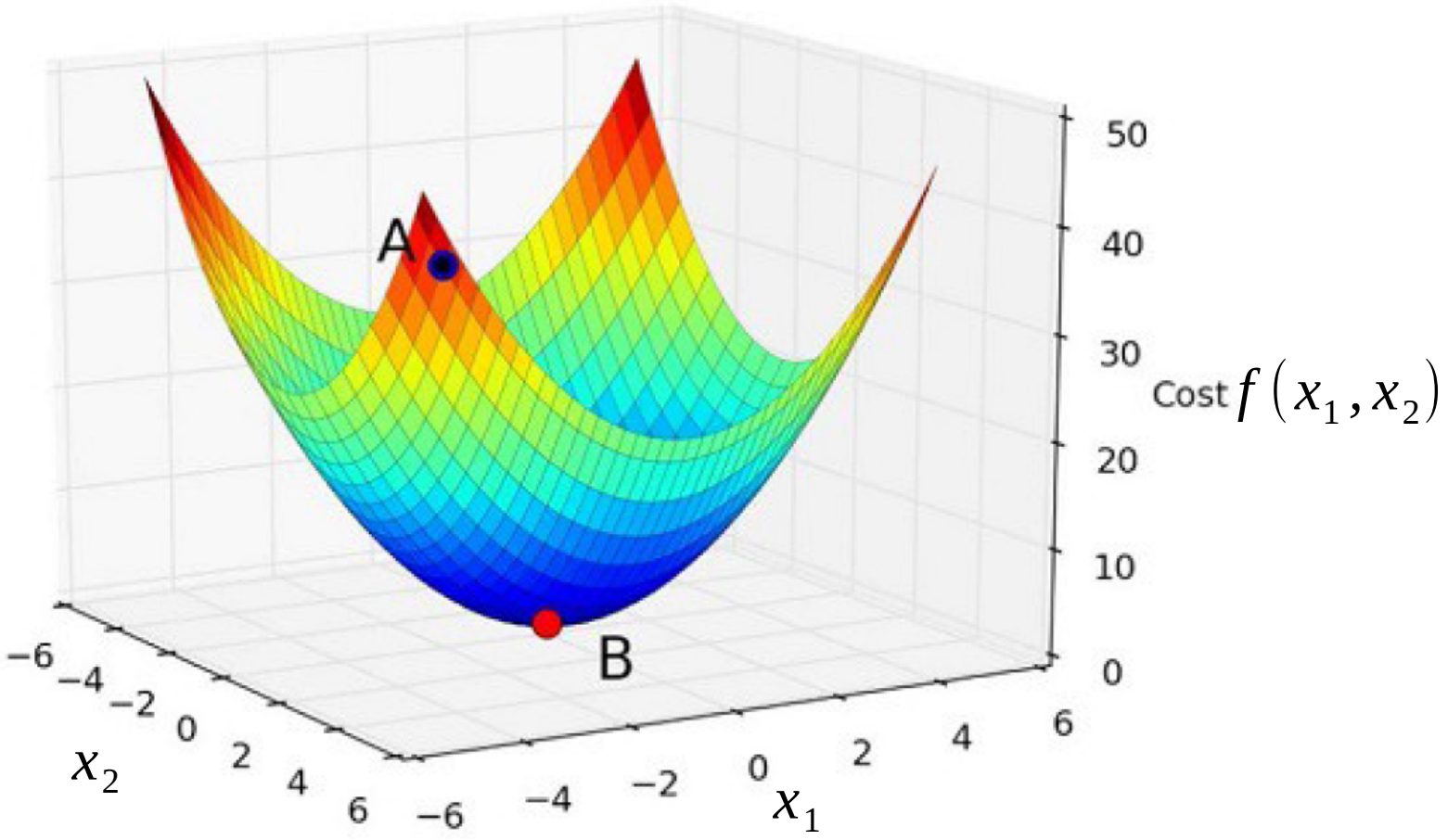
1: **repeat**
2: $\quad \beta \leftarrow \beta - \alpha \nabla f(\beta)$
3: **until** $|\Delta\beta| < \theta$ [perhaps for 10 iterations in sequence]

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

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 β



step 1: computing the gradient vector

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

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

$\beta \leftarrow \beta - \alpha \nabla f(\beta)$

step 3: iterative algorithm

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

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

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

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)

Iteration 2? Exercise

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

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

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

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

– iteration k

– ...

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

Review: gradient descent algorithm

- **Alternatively, instead of gradient descent to minimize the cost function we could use**
  - Conjugate gradient
  - BFGS (Broyden-Fletcher-Goldfarb-Shanno)
  - L-BFGS (Limited memory - BFGS)
- **These are more optimized algorithms which take that same input and minimize the cost function**
  - These are very complicated algorithms

- **These advanced alternative's advantages**:
  - No need to manually pick alpha (learning rate)
  - Have a clever inner loop (line search algorithm) which tries a bunch of alpha values and picks a good one
  - Often faster than gradient descent
  - Do more than just pick a good learning rate
  - Can be used successfully without understanding their complexity
- **Disadvantages**:
  - Could make debugging more difficult
  - Should not be implemented themselves
  - Different libraries may use different implementations - may hit performance

Logistic regression: Optimum parameters

- **Find β^* 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

- Taking gradient:

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

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

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

we obtain (re-writing the cost function)

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

Logistic regression: Computing the loss's gradient

– 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

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

Computing the loss's gradient: Sketch

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

– compute partial derivatives

$$\frac{\partial}{\partial \beta} \log \frac{e^{\phi(x_i)^\top \beta}}{1+e^{\phi(x_i)^\top \beta}} \qquad \frac{\partial}{\partial \beta} \log \left(1 - \frac{e^{\phi(x_i)^\top \beta}}{1+e^{\phi(x_i)^\top \beta}} \right)$$

use the rule:

+) derivative of $\log$ function: $\frac{\partial \log(x)}{\partial x} = \frac{1}{x}$

+) derivative of exponential function: $\frac{\partial e^x}{\partial x} = e^x$

+) chain rule: $\frac{\partial f(y(x))}{\partial x} = \frac{\partial f(y(x))}{\partial y} \frac{\partial y(x)}{\partial x}$

ex.: $f(\beta) = e^{\phi(x)^\top \beta}$, use chain rule as $(f(y) = e^y; y = \phi(x)^\top \beta)$

$$\frac{\partial f(\beta)}{\partial \beta} = \frac{\partial f(y)}{\partial y} \frac{\partial y}{\partial \beta} = e^y \phi(x) = e^{\phi(x)^\top \beta} \phi(x)$$

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, λ 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 β **iteratively**

Input: initial $\beta \in \mathbb{R}^k$, stepsize α, tolerance θ
Output: β

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]

---

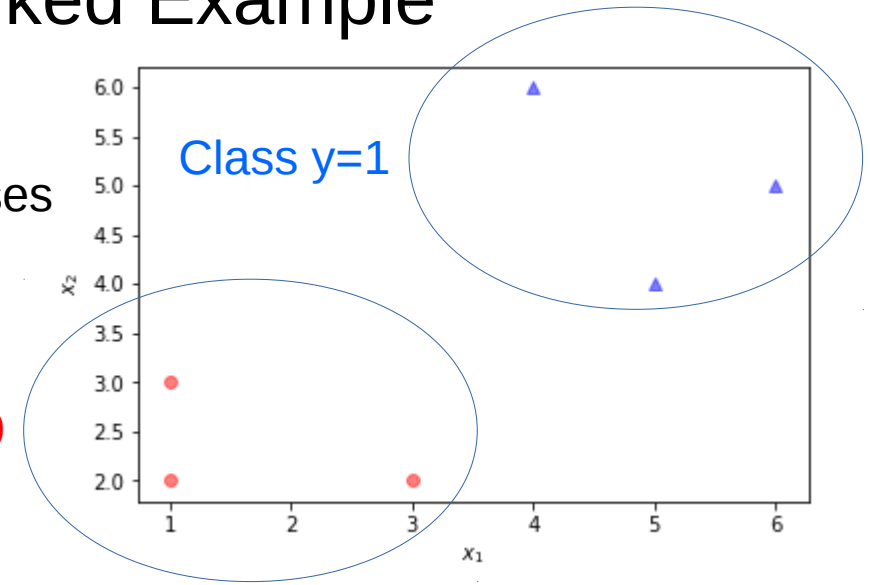
where

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

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

Logistic regression: Worked Example

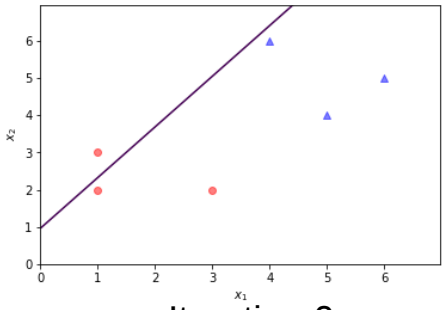
Given n=6 data points of two classes

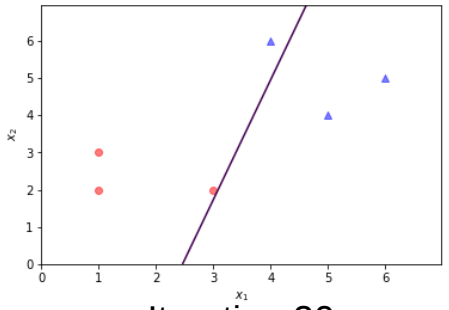


randomly =0.1 =0.01

- using gradient descent to find optimum β **iteratively**

Input: initial $\beta \in \mathbb{R}^k$, stepsize α, tolerance θ
Output: β
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]





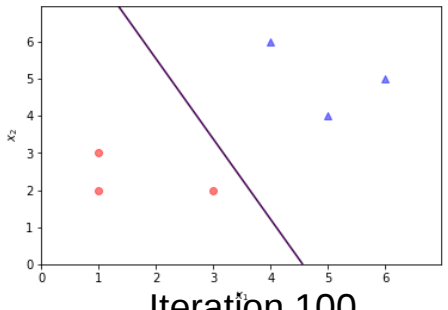
where

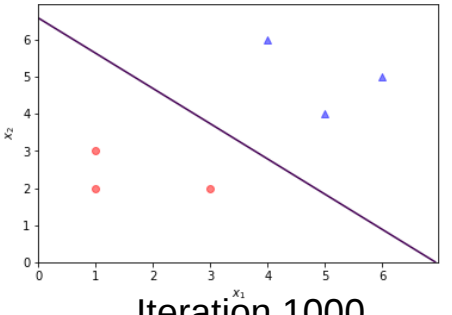
$$p = [p_1, p_2, \cdots, p_n]^\top \text{ in which } p_i := p(y = 1 \mid x_i) = \frac{1}{1 + e^{-\phi(x_i)^\top \beta}}$$

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

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

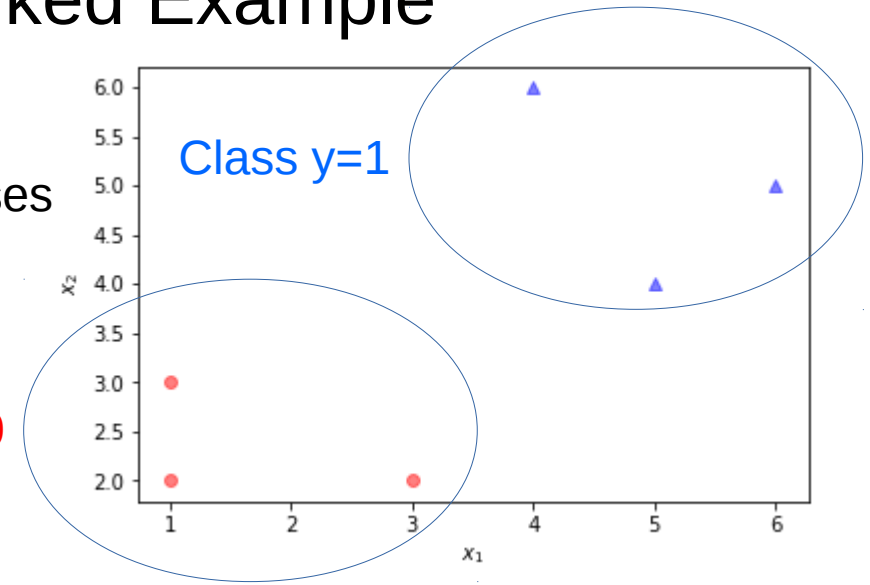
Linear feature: $\phi(x) = [1, x_1, x_2]$





Logistic regression: Worked Example

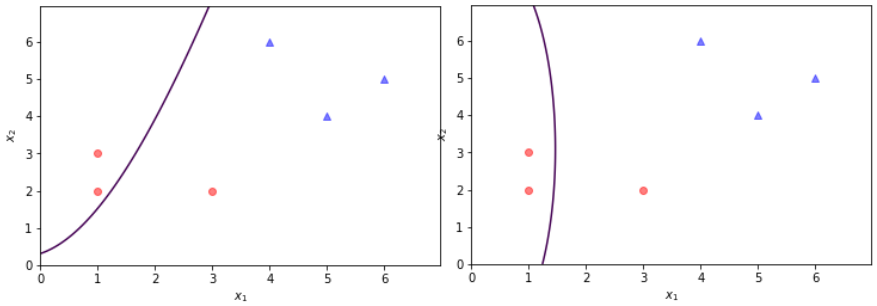
Given n=6 data points of two classes
(see jupyter notebook file)



randomly =0.1 =0.01

- using gradient descent to find optimum β **iteratively**

Input: initial $\beta \in \mathbb{R}^k$, stepsize α, tolerance θ
Output: β
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]

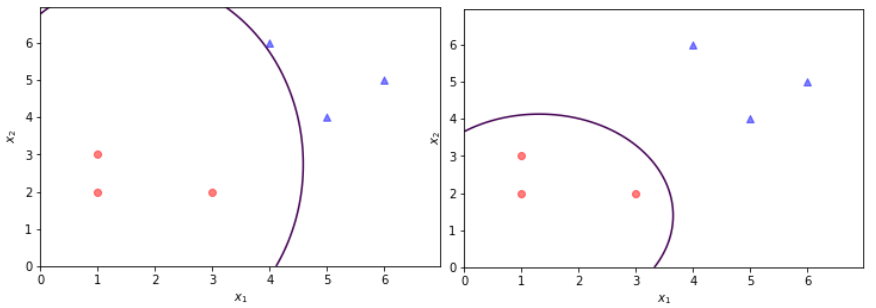


where

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

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

```
[[ 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]]
```



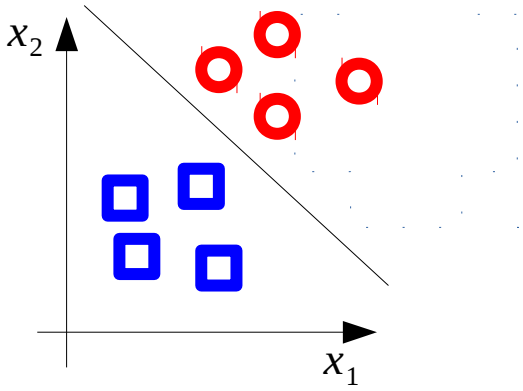
Quadratic feature: $\phi(x) = [1, x_1, x_2, x_1^2, x_2^2]$

Outline

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

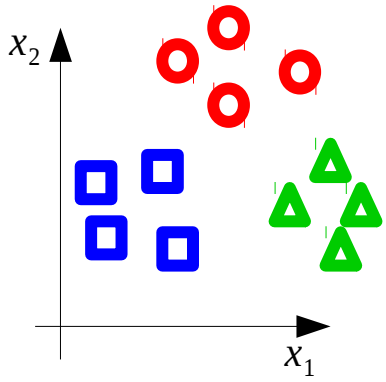
Logistic regression: multi-class classification

Binary classification



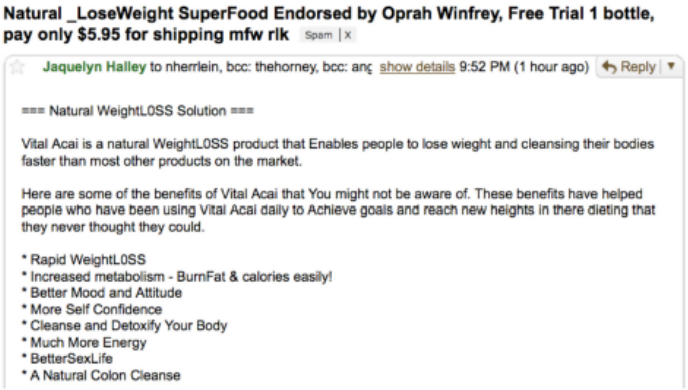
Output y={ , }

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 β^* 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**

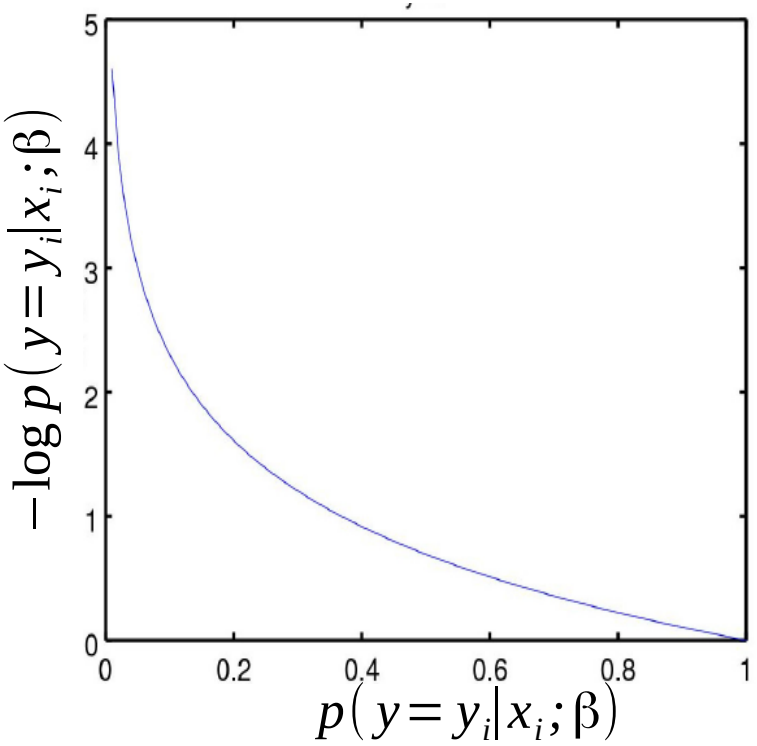
$$l(x_i, y_i) = -\log p(y=y_i|x_i;\beta)$$

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 β^* 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 β

The gradient is computed as (w.r.t parameters of function k):

$$\frac{\partial L^{\text{logistic}}(\beta)}{\partial \beta_k}^{\top} = \sum_{i=1}^{n} (p_{ik} - [y_i = k])\phi(x_i) + 2\lambda I \beta_k = X^{\top}(p_k - [y = k]) + 2\lambda I \beta_k$$

, $p_{ik} = p(y = k \mid x_i)$ and $[y = k] = \begin{bmatrix} [y_1 = k] \\ [y_2 = k] \\ \vdots \\ [y_n = k] \end{bmatrix}$

Advanced Topics in Classification

Structured Output & Structured Input

- regression:

$$\mathbb{R}^n \rightarrow \mathbb{R}$$

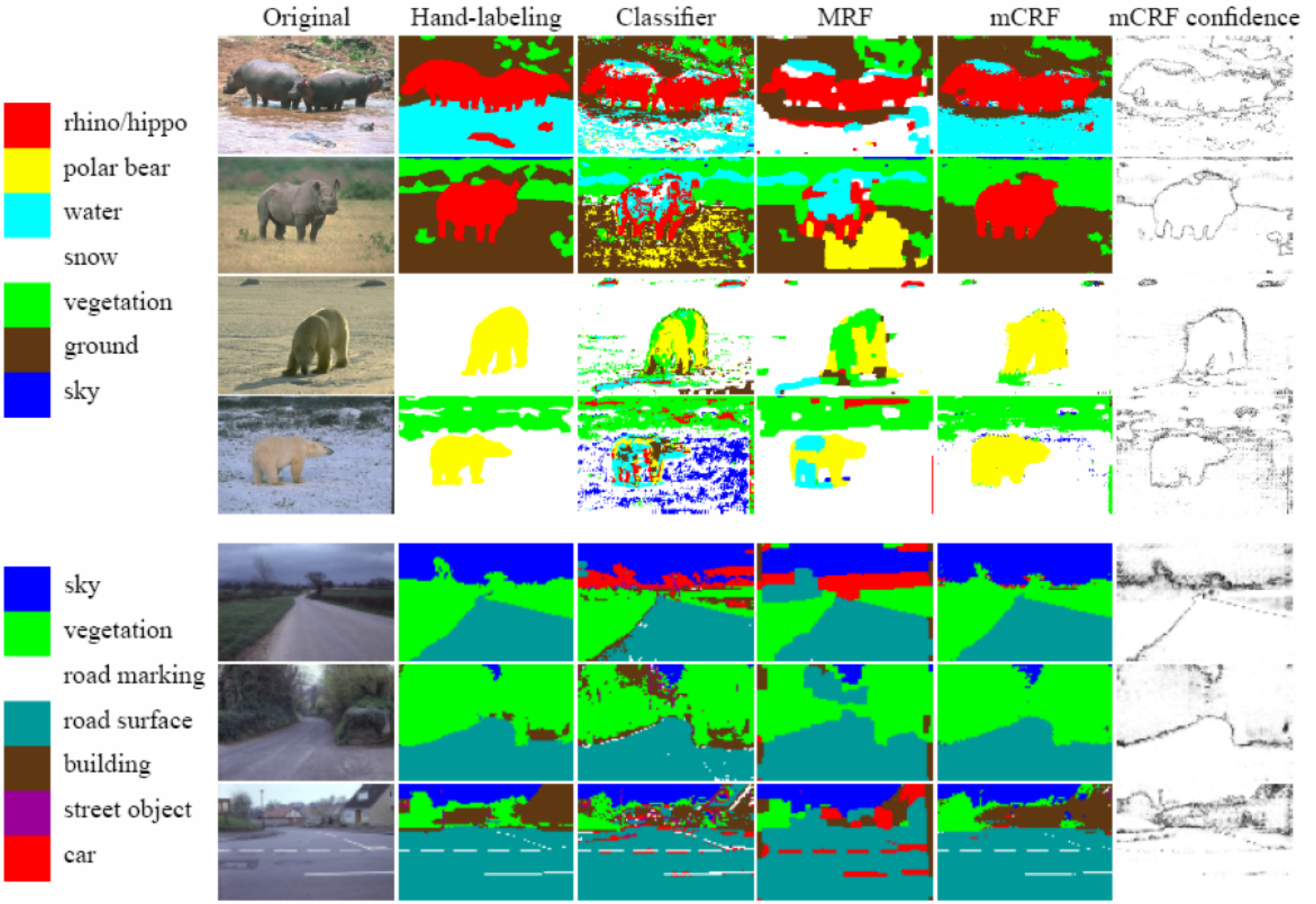
- structured output:

$$\mathbb{R}^n \rightarrow \text{binary class label } \{0, 1\}$$
$$\mathbb{R}^n \rightarrow \text{integer class label } \{1, 2, .., M\}$$
$$\mathbb{R}^n \rightarrow \text{sequence labelling } y_{1:T}$$
$$\mathbb{R}^n \rightarrow \text{image labelling } y_{1:W,1:H}$$
$$\mathbb{R}^n \rightarrow \text{graph labelling } y_{1:N}$$

- structured input:

$$\text{relational database} \rightarrow \mathbb{R}$$
$$\text{labelled graph/sequence} \rightarrow \mathbb{R}$$

Structured Output & Structured Input: Example



Summary

- We covered:
  - KNN (k-nearest neighbor) algorithm
  - Discriminant function
  - Logistic regression for binary and multi-class classification: using gradient descent