Machine Learning and Data Analytics
ME 5013- Fall 2019

<u>Lectures 09 and 10</u>

- Logistic Regression
- K Nearest Neighbors

The University of Texas at San Antonio™

Adel Alaeddini, PhD
Associate Professor of Mechanical Engineering
Advanced Data Engineering Lab
adel.alaeddini@utsa.edu

Disclosure: the slides are adopted from various online resources

Qualitative target/response variables take values in an unordered set C

Binary classification

Email ∈ {Spam , Not Spam}
Online Transactions ∈ { Fraudulent, Non-fradulent}
Tumor ∈ { Malignant , Benign}

$y \in \{0,1\}$ 0: "Negative Class" (e.g., benign tumor)
1: "Positive Class" (e.g., malignant tumor)

Multiclass classification

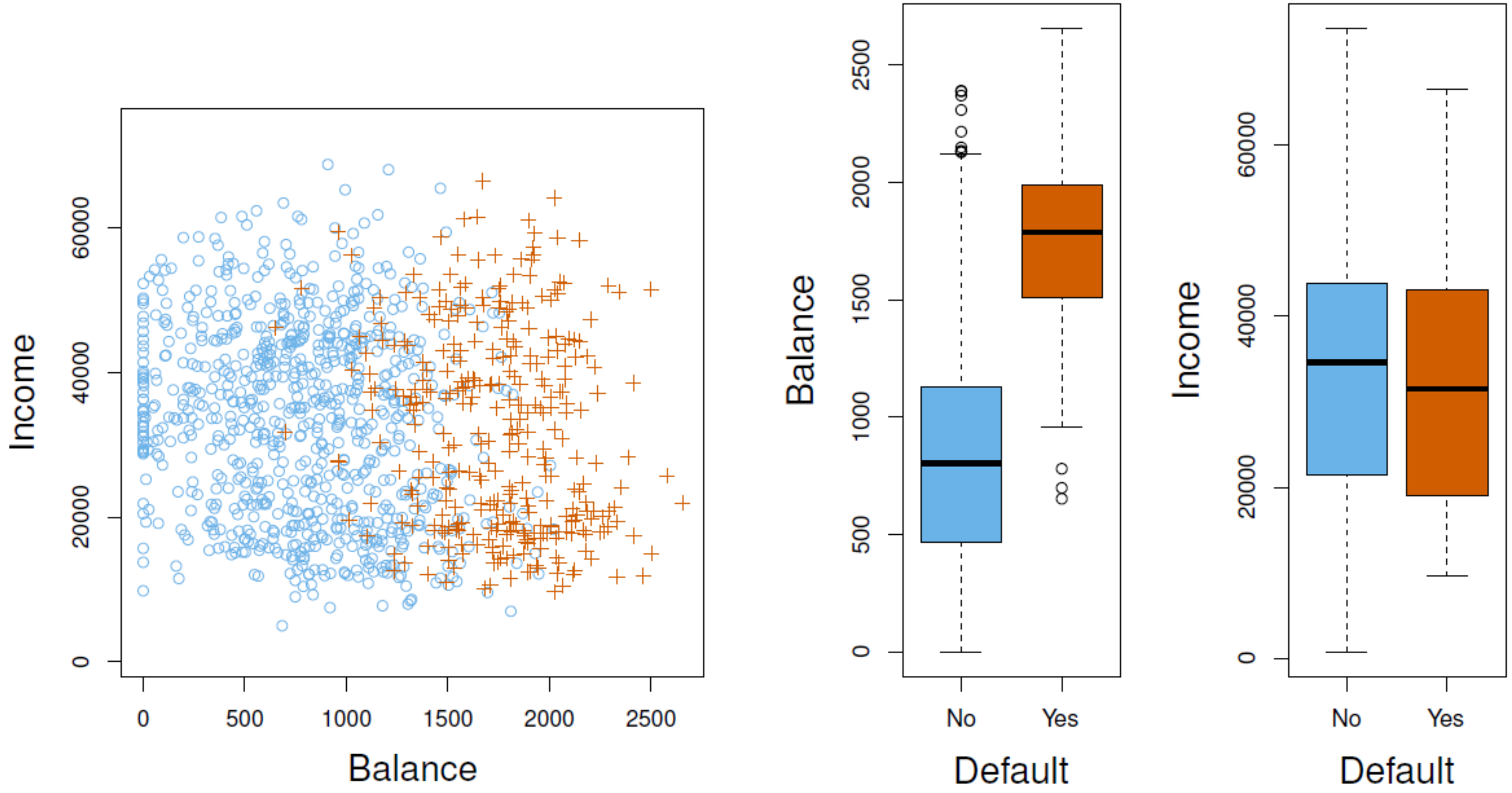
eye color ∈ {brown, blue, green}

$y \in \{1, \dots, k\}$

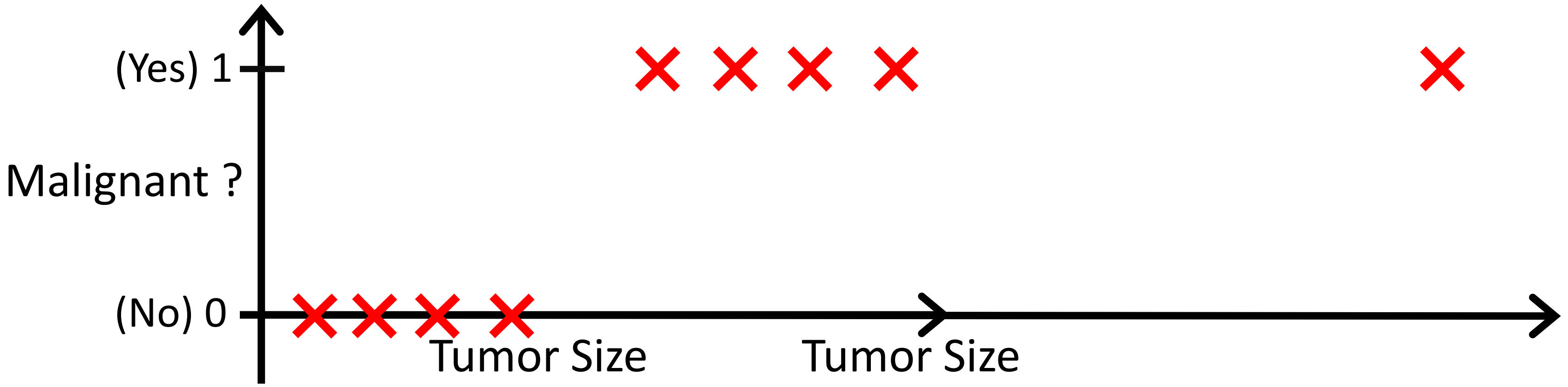
- Often we are more interested in estimating the *probabilities* that x belongs to each category in C, i.e. $p(y = k|x)$.

- Example: it is more valuable to have an estimate of the probability that an insurance claim is fraudulent, than a classification fraudulent or not

Example



Income
Balance
0
20000
40000
60000
0
500
1000
1500
2000
2500
Balance
Default
No
Yes
Income
Default
No
Yes



Threshold classifier output $h_\beta(\mathbf{x})$ at 0.5:

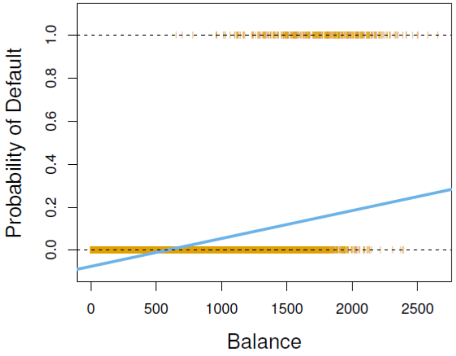
If $h_\beta(x) \geq 0.5$, predict "y = 1"

If $h_\beta(x) < 0.5$, predict "y = 0"

Binary Classification: y = 0 or 1

linear regression

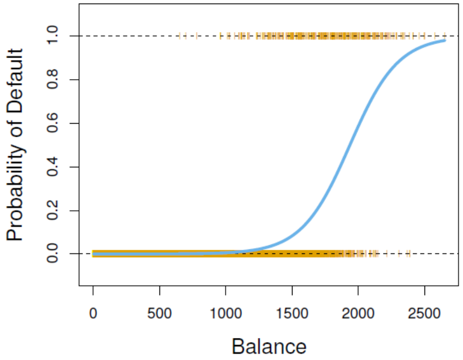
- Can be affected more by outliers
- Not appropriate for multi-class classification
- Might produce probabilities less than zero or bigger than one



$$h_\beta(x) \text{ can be} > 1 \text{ or} < 0$$

Logistic Regression:

- produce probabilities between zero and one.



$$0 \leq h_\beta(x) \leq 1$$

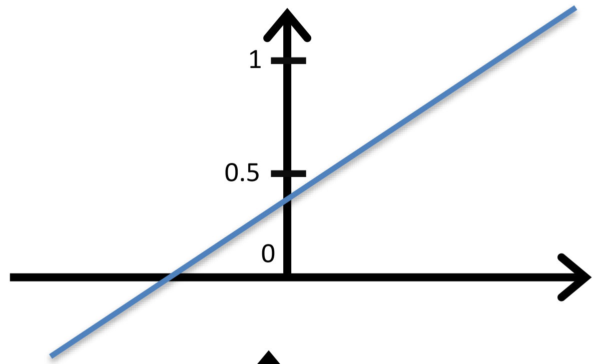
Hypothesis Representation

Linear regression

$$h_\beta(x) = x\beta$$

$$\beta = \begin{bmatrix}\beta_0 \\ \beta_1\end{bmatrix}$$

$$x = [x_0 = 1 \quad x_1]$$



Logistic regression

$$h_\beta(x) = g(x\beta)$$

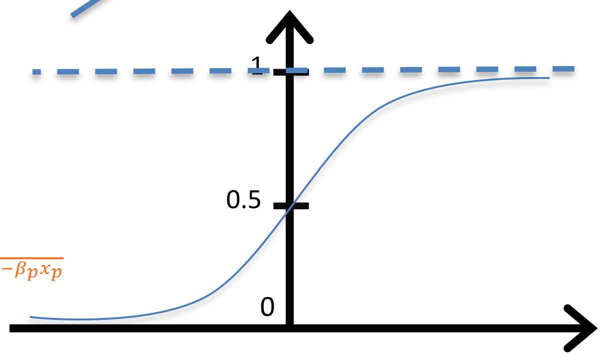
$$g(z) = \frac{1}{1+e^{-z}}$$

Link function (g):
- Sigmoid function
- Logistic function

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

For simple binary problem: $\frac{1}{1+e^{-\beta_0-\beta_1 x_1}}$

For multiple binary problem: $\frac{1}{1+e^{-\beta_0-\beta_1 x_1-\cdots-\beta_p x_p}}$



- An alternative format of $h_\beta(x)$ is $h_\beta(x) = \frac{e^{x\beta}}{1+e^{x\beta}}$
- $h_\beta(x)$ is indeed the probability of y belong to positive, i.e. class $P(x) = p(y = 1|x, \beta)$
- $e \approx 2.71$ is the Euler's number
- no matter what values$x$and β take, $p(x)$ will have values between 0 and 1
- By a bit rearrangement we get $\log\left(\frac{h_\beta(x)}{1-h_\beta(x)}\right) = \log\left(\frac{p(x)}{1-p(x)}\right) = x\beta$, which is called the log odds or logit transformation of $h_\beta(x)$ or $p(x)$

Interpretation of Hypothesis Output

$h_{\boldsymbol{\beta}}(\mathbf{x})$ = estimated probability that y = 1 on input x

Example: If $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$

$h_\beta(x)$=0.7

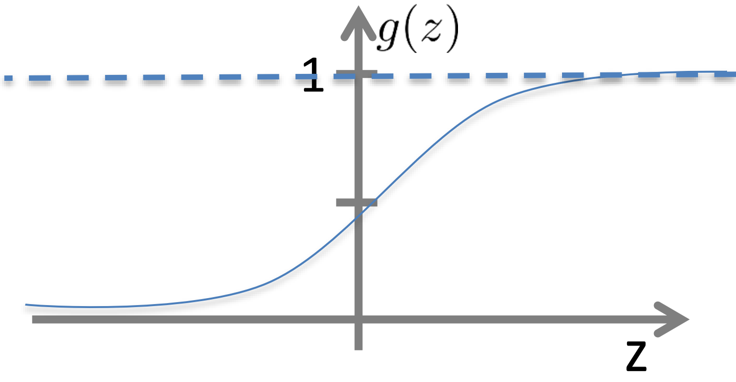
Tell patient that 70% chance of tumor being malignant

"probability that y = 1, given x, parameterized by θ"

$$P(y=0|x;\beta) + P(y=1|x;\beta) = 1$$
$$P(y=0|x;\beta) = 1 - P(y=1|x;\beta)$$

Decision Boundary

The **decision boundary** is the line that separates the area where y = 0 and where y = 1. It is created by our hypothesis function.



Logistic regression

$$h_\beta(\mathrm{x}) = g(\mathrm{x}\beta)$$

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

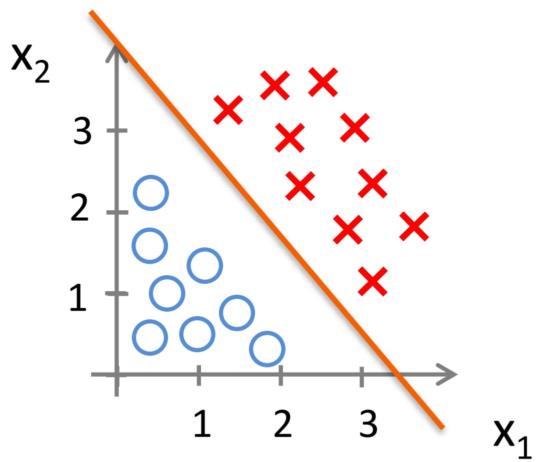
Suppose predict "$y = 1$" if $h_\beta(x) \geq 0.5$

predict "$y = 0$" if $h_\beta(x) < 0.5$

Decision Boundary

$$h_\beta(\mathrm{x}) = g(\beta_0 + \beta_1 x_1 + \beta_2 x_2)$$

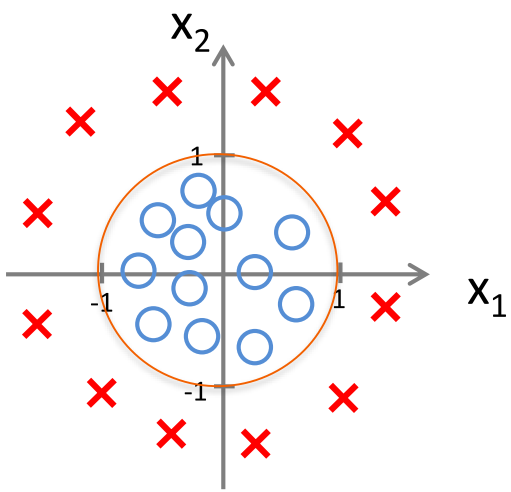
Predict "$y = 1$" if $-3 + x_1 + x_2 \geq 0$



Non-linear decision boundaries

$$h_\beta(\mathrm{x}) = g(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4 x_2^2)$$

Predict "$y = 1$" if $-1 + x_1^2 + x_2^2 \geq 0$



Cost function

Training set:

$$X = \begin{array}{c} \begin{array}{ccccc} x_0 & x_1 & & & x_p \end{array} \\ \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} \end{array}$$

$$y \in \{0, 1\}$$

n examples

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}$$

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

How to choose parameters β ?

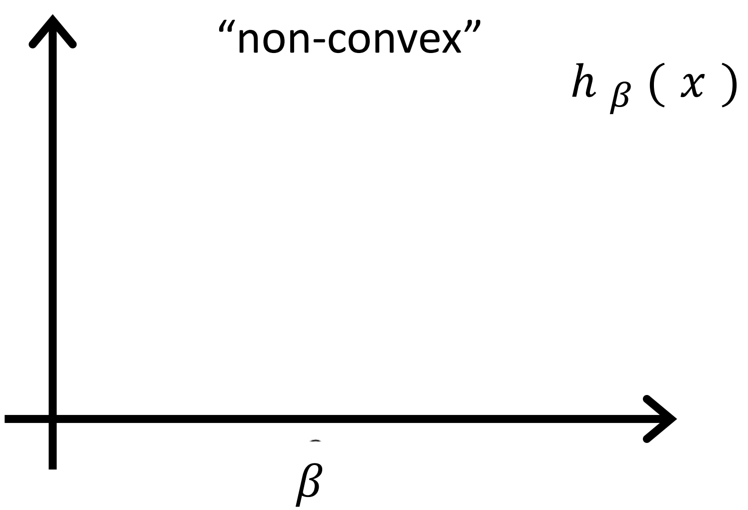
Cost function

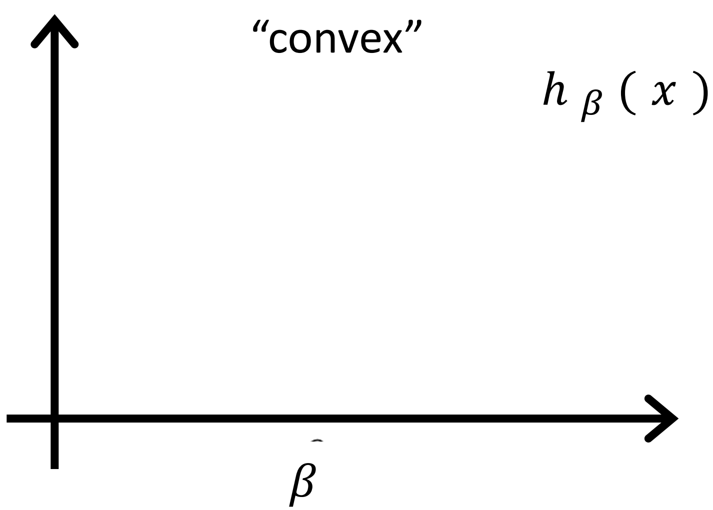
Linear regression:

Hypothesis: $h_\beta(x) = \beta_0 + \beta_1 x$

$$\min_{\beta_0,\beta_1} J(\beta_0, \beta_1) = \frac{1}{2n}\sum_{i=1}^{n}(y_n - \beta_0 - \beta_1 x_i)^2$$

$$\mathrm{Cost}\left(h_\beta(x^{(i)}), y^{(i)}\right) = \frac{1}{2}\left(h_\beta(x^{(i)}) - y^{(i)}\right)^2$$

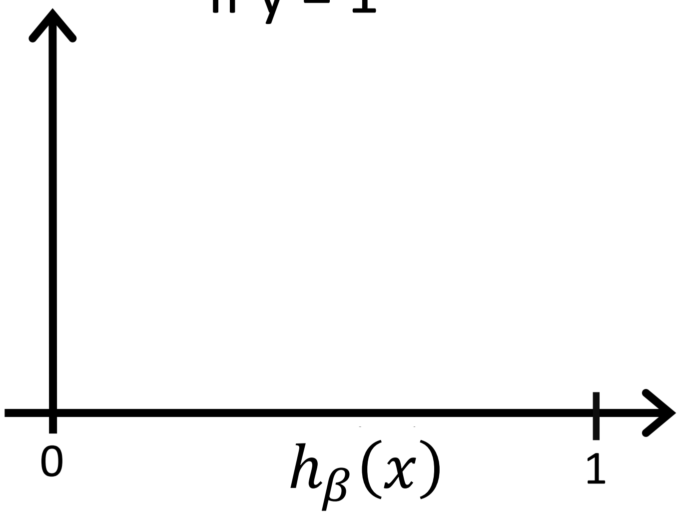




Logistic regression cost function

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

If y = 1



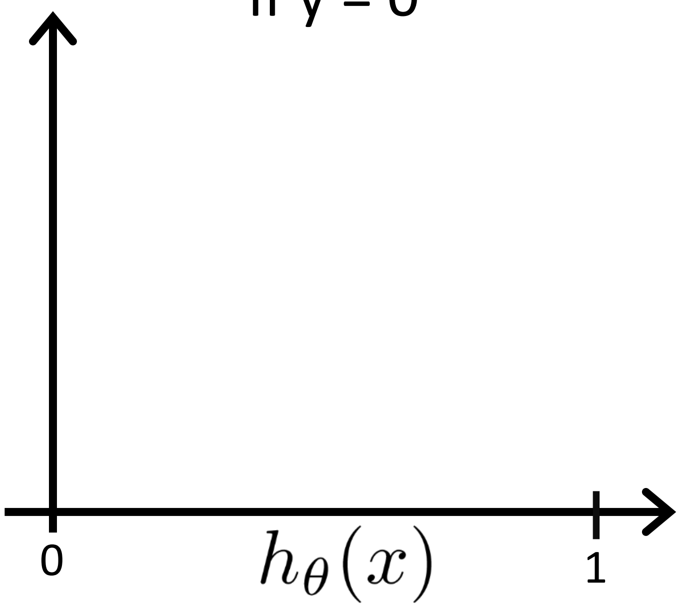
Cost $= 0$ if $y = 1, h_\beta(x) = 1$
But as $h_\beta(x) \to 0$
$Cost \to \infty$

Captures intuition that if $h_\beta(x) = 0$, (predict $P(y = 1|x;\ \beta) = 0$), but $y = 1$, we'll penalize learning algorithm by a very large cost.

Logistic regression cost function

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

If y = 0



Simplified cost function and gradient descent

Logistic regression cost function

$$J(\beta) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{Cost}(\, h_\beta(x^{(i)})\,, y^{(i)})$$

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

Note: $y = 0$ or 1 always

Simplified cost function and gradient descent

$$J(\beta) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{Cost}(\, h_\beta(x^{(i)})\, , y^{(i)})$$
$$= -\frac{1}{n}[\sum_{i=1}^{n} y^{(i)} \log\ h_\beta(x^{(i)}) + (1-y^{(i)}) \log(1-\ h_\beta(x^{(i)})\ )]$$

To fit parameters β :

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

To make a prediction given new x :

Output $h_\beta(x) = \dfrac{1}{1+e^{-x\beta}}$

- Equivalent to maximum likelihood to estimate the parameters
- The likelihood (below function) gives the probability of the observed zeros and ones in the data.

$$\ell(\beta_0, \beta) = \prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} (1-p(x_i)).$$

Gradient Descent

$$J(\beta) = -\frac{1}{n}\left[\sum_{i=1}^{n} y^{(i)} \log\ h_\beta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\beta(x^{(i)}))\right]$$

Want $\min_{\beta} J(\beta)$:

Repeat $\{$

$\beta_j := \beta_j - \alpha \frac{\partial}{\partial \beta_j} J(\beta_0, \beta_1)$ (for $j = 0$ and $j = 1$)

$\}$ (simultaneously update all parameters)

Closed form formula of the gradients

$$\beta := \beta + \alpha \frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} - h_\beta\left(x^{(i)}\right)\right) x^{(i)}$$

$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \end{bmatrix}$

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

$x = [x_0 = 1 \quad x_1 \quad \ldots]$

The formula can be used for arbitrary number of explanatory factors

Example

	Coefficient	Std. Error	Z-statistic	P-value
Intercept	-10.6513	0.3612	-29.5	< 0.0001
balance	0.0055	0.0002	24.9	< 0.0001

What is our estimated probability of default for someone with a balance of $1000?

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 X}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 X}} = \frac{e^{-10.6513+0.0055\times 1000}}{1+e^{-10.6513+0.0055\times 1000}} = 0.006$$

With a balance of $2000?

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 X}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 X}} = \frac{e^{-10.6513+0.0055\times 2000}}{1+e^{-10.6513+0.0055\times 2000}} = 0.586$$

Example

Lets do it again, using `student` as the predictor.

	Coefficient	Std. Error	Z-statistic	P-value
`Intercept`	-3.5041	0.0707	-49.55	< 0.0001
`student[Yes]`	0.4049	0.1150	3.52	0.0004

$$\widehat{\Pr}(\texttt{default=Yes}|\texttt{student=Yes}) = \frac{e^{-3.5041+0.4049\times 1}}{1+e^{-3.5041+0.4049\times 1}} = 0.0431,$$

$$\widehat{\Pr}(\texttt{default=Yes}|\texttt{student=No}) = \frac{e^{-3.5041+0.4049\times 0}}{1+e^{-3.5041+0.4049\times 0}} = 0.0292.$$

Example

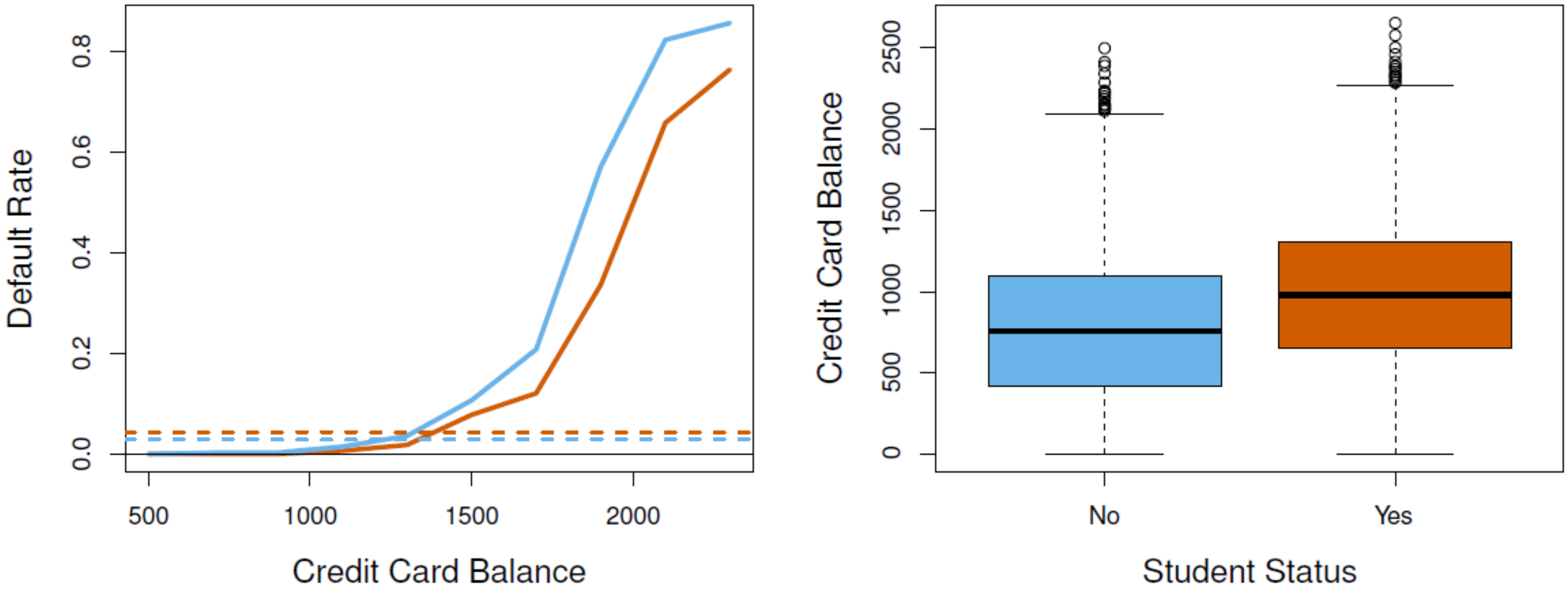
$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}{1+e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}$$

	Coefficient	Std. Error	Z-statistic	P-value
`Intercept`	-10.8690	0.4923	-22.08	< 0.0001
`balance`	0.0057	0.0002	24.74	< 0.0001
`income`	0.0030	0.0082	0.37	0.7115
`student[Yes]`	-0.6468	0.2362	-2.74	0.0062

Why is coefficient for `student` negative, while it was positive before?

Example



- Students tend to have higher balances than non-students, so their marginal default rate is higher than for non-students.
- But for each level of balance, students default less than non-students.
- Multiple logistic regression can tease this out.

Other optimization algorithms

Given θ, we have code that can compute

- $J(\theta)$
- $\frac{\partial}{\partial \theta_j} J(\theta)$ (for $j = 0, 1, \ldots, n$)

Optimization algorithms:

- Gradient descent
- Conjugate gradient
- BFGS
- L-BFGS

Advantages:

- No need to manually pick α
- Often faster than gradient descent.

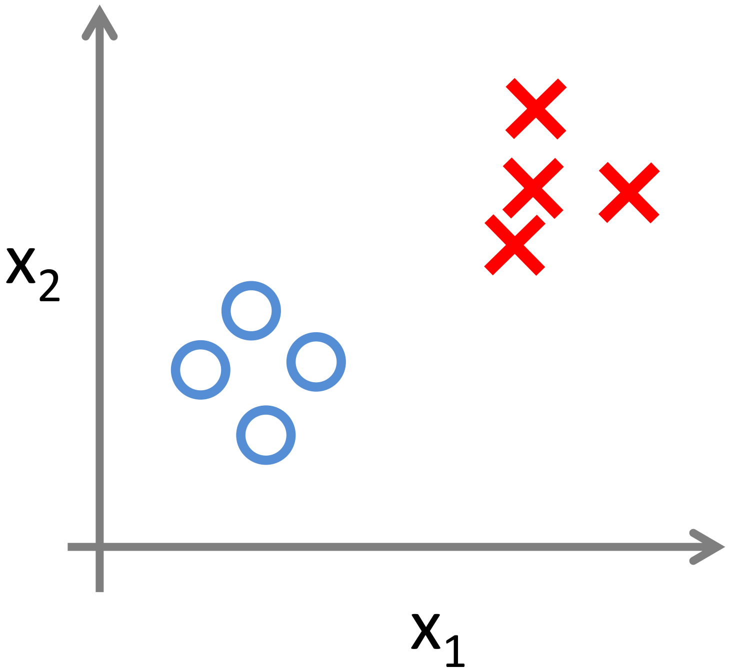
Disadvantages:

- More complex

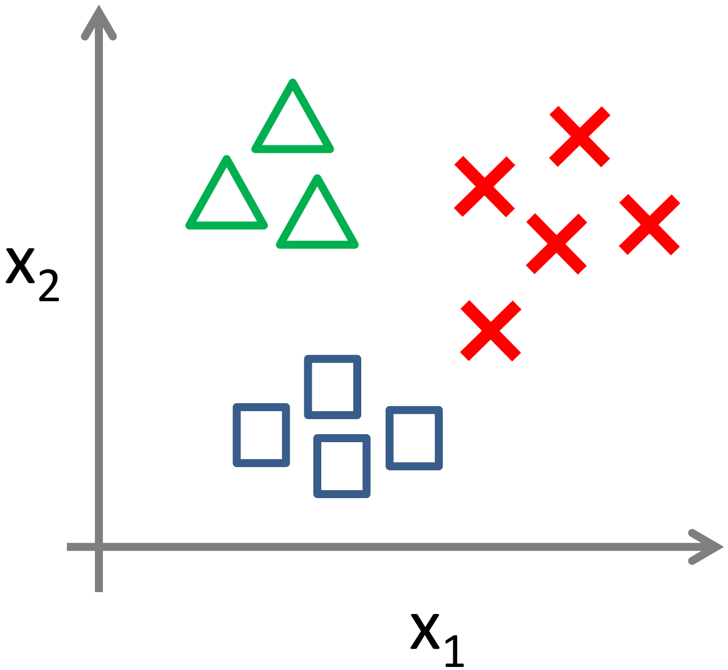
Multiclass classification

- Logistic regression with more than two classes
  - Email foldering/tagging: Work, Friends, Family, Hobby
  - Medical diagrams: Not ill, Cold, Flu
  - Weather: Sunny, Cloudy, Rain, Snow

Binary classification:

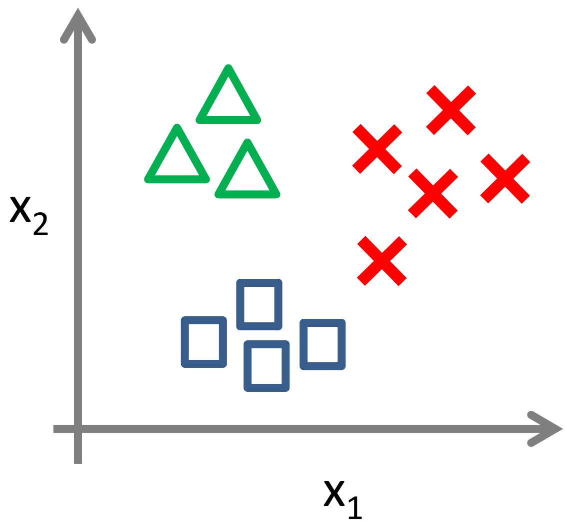


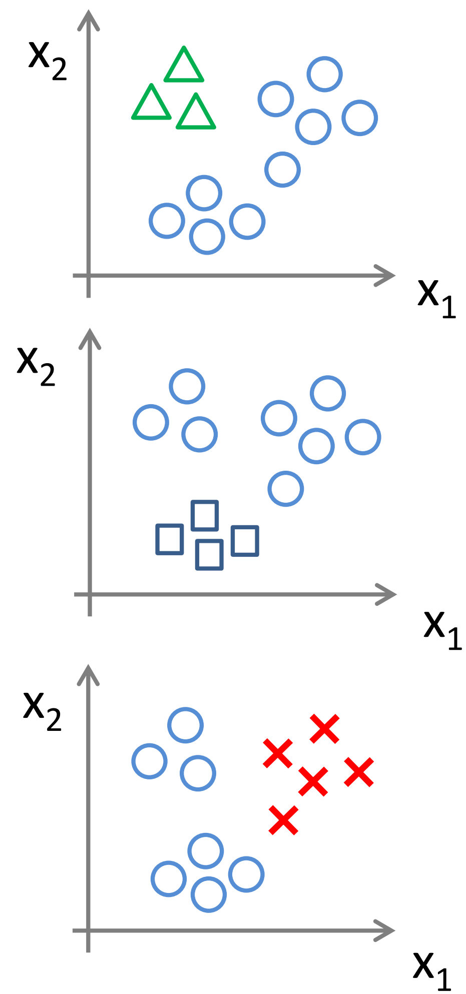
Multi-class classification:



Multiclass classification

One-vs-all (one-vs-rest):





Class 1: △

Class 2: □

Class 3: ✕

$$h_\theta^{(i)}(x) = P(y = i|x;\theta) \qquad (i = 1, 2, 3)$$

One-vs-all

Train a logistic regression classifier $h_\beta^{(k)}(x)$ for each class k to predict the probability that $y = k$.

On a new input x, to make a prediction, pick the class k that maximizes $\max_k h_\beta^{(k)}(x)$

Multinomial regression

$$p(y = k|x) = h_\beta^{(k)}(x) = \frac{e^{x\beta^{(l)}}}{\sum_{l=1}^{K} e^{x\beta^{(l)}}} \quad , \quad l = 1, \ldots k$$

$$= \frac{e^{\beta_0^{(l)} + \beta_1^{(l)} x_1 + \cdots + \beta_p^{(l)} x_p}}{\sum_{l=1}^{K} e^{\beta_0^{(l)} + \beta_1^{(l)} x_1 + \cdots + \beta_p^{(l)} x_p}} \quad , l = 1, \ldots k$$

Alternative formula

$$p(y = k|x) = \frac{e^{x\beta^{(l)}}}{1 + \sum_{l=1}^{K-1} e^{x\beta^{(l)}}} \, , l = 1, \ldots k - 1$$

- There is a linear function for each class.
- Multiclass logistic regression is also referred to as multinomial regression.

Summary

- **Hypothesis**

$$h_\beta(x) = \frac{1}{1+e^{-x\beta}} \qquad \beta = \begin{bmatrix}\beta_0\\ \beta_1\\ \vdots\end{bmatrix} \qquad x = [x_0 = 1 \quad x_1 \quad \ldots]$$

- **Cost function**

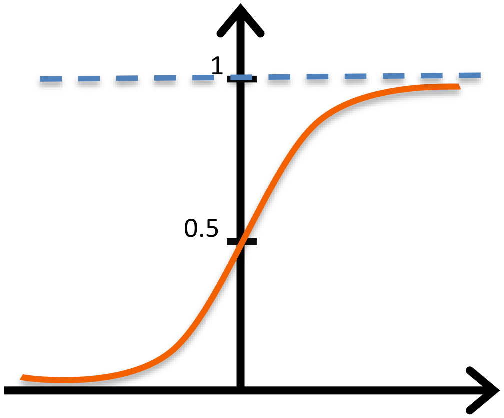
$$J(\beta) = -\frac{1}{n}\left[\sum_{i=1}^{n} y^{(i)} \log h_\beta(x^{(i)}) + (1-y^{(i)}) \log (1 - h_\beta(x^{(i)}))\right]$$

- **Gradient descent**

Repeat {

$$\beta := \beta + \alpha \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - h_\beta\left(x^{(i)}\right)\right)x^{(i)}$$

} (simultaneously update all parameters)



- **Multinomial logistic regression**

Train a logistic regression classifier $h_\beta^{(k)}(x)$ for each class k to predict the probability that $y = k$.

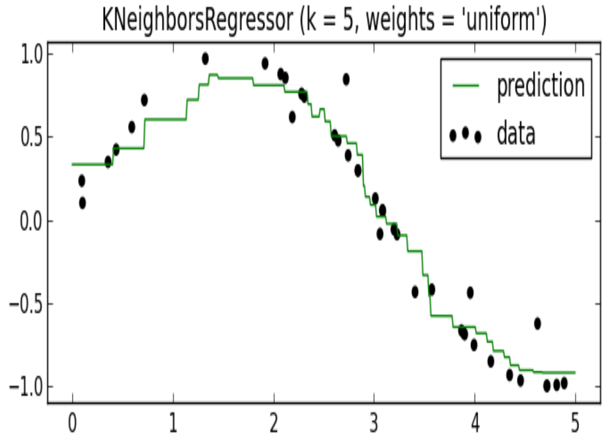
KNN Approach

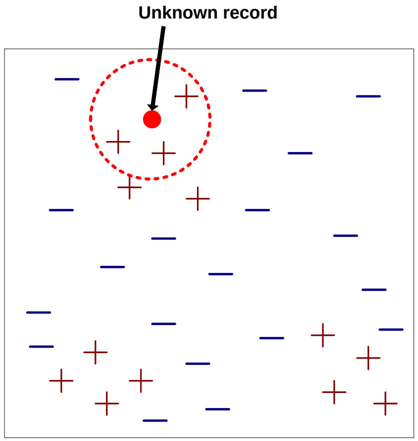
Regression

- An object (a new instance) value is estimated by the (weighted) average of its neighbor value.
- The weight and neighbors are identified based on *a distance function*

Classification

- An object (a new instance) is classified by a majority votes for its neighbor classes. (common class amongst its K nearest neighbors)
- The neighbors are identified based on *a distance function*





KNN with weights

- Refinement to KNN is to weight the contribution of each *k* neighbor according to the distance to the query point x_q
- Greater weight to closer neighbors

Weight function

$$w_i = \begin{cases} \dfrac{1}{d(x_q, x_i)^2} & if \quad x_q \neq x_i \\ 1 & else \end{cases}$$

For continuous target functions

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$$

For discrete target functions

$$\hat{f}(x_q) \leftarrow \underset{v \in V}{\arg\max} \sum_{i=1}^{k} w_i \delta(v, f(x_i))$$