FACULTY OF INFORMATION TECHNOLOGY

Machine Learning

(Máy Học)

Semester 2, 2022/2023

Chapter 2. Regression

Some slides taken from course materials of Andrew Ng

Content

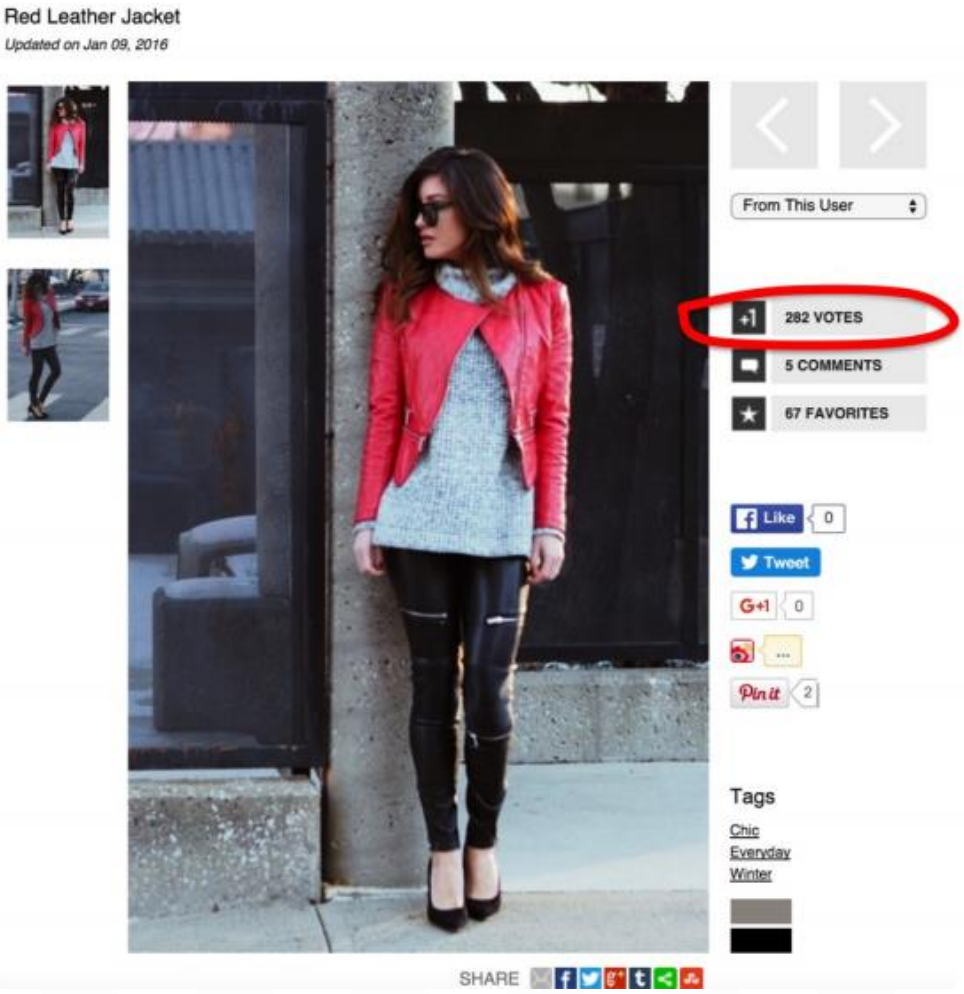
- What is regression?
- Univariate linear regression
- Linear Regression for multiple variables
- Logistic regression

Problems for Today

- Goal: Predict movie rating automatically

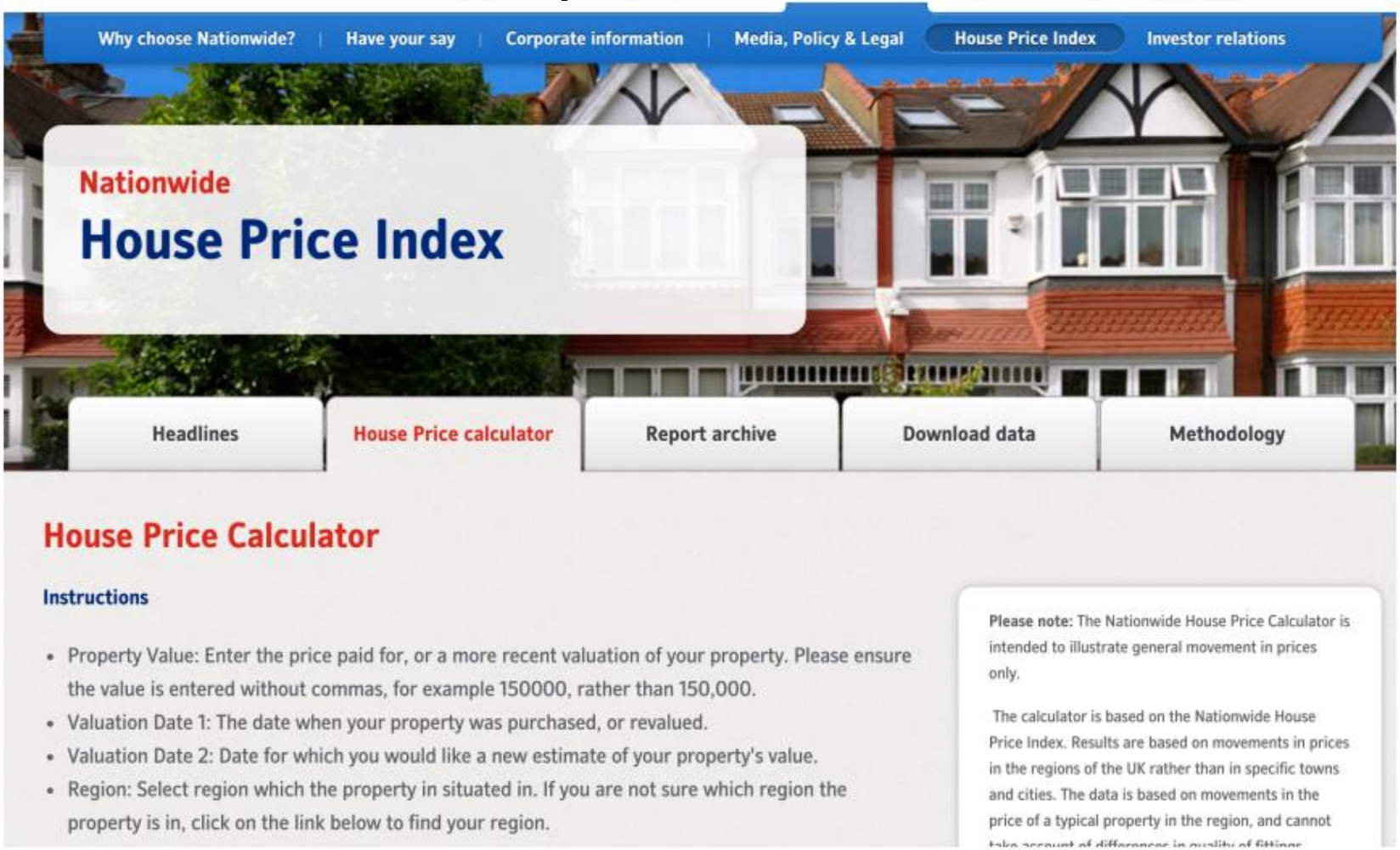
Problems for Today

- Goal: How many votes will I get?



Problems for Today

- Goal: Predict the price of a house?



Predict the price of a house

- For a given dataset as follows:

Living Area (feet^2)	Price ($)
2104	400.000
1600	330.000
2400	369.000
1416	232.000
3000	540.000
…	…

Training set

- How can we learn to predict the prices of houses of other sizes in the city, as a function of their living area?

Predict the price of a house (cont.)

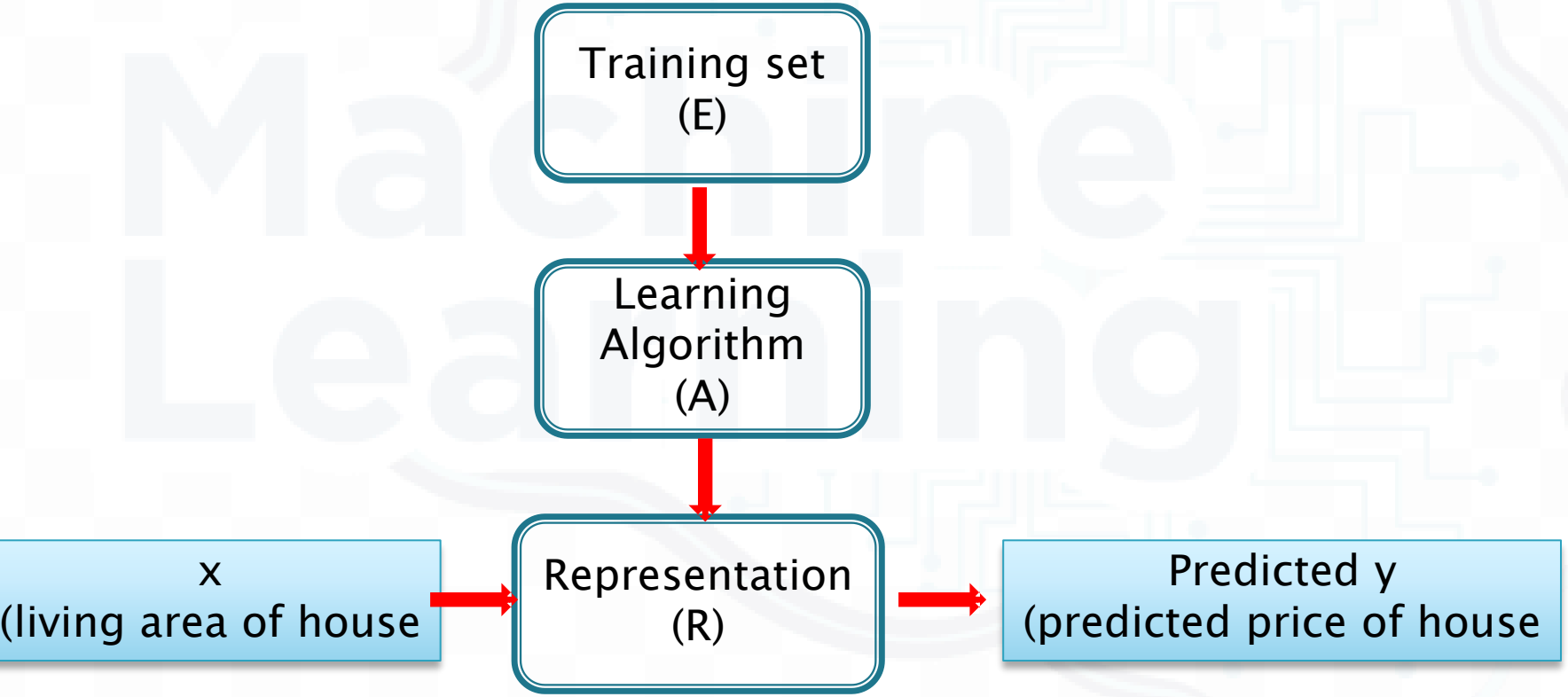
- For a given dataset as follows:

Living Area (feet^2)	Price ($)
2104	400.000
1600	330.000
2400	369.000
1416	232.000
3000	540.000
…	…

- What is learning type?
  - Supervised learning
- Regression or Classification?
  - When the target variable we are trying to predict is continuous, regression problem.

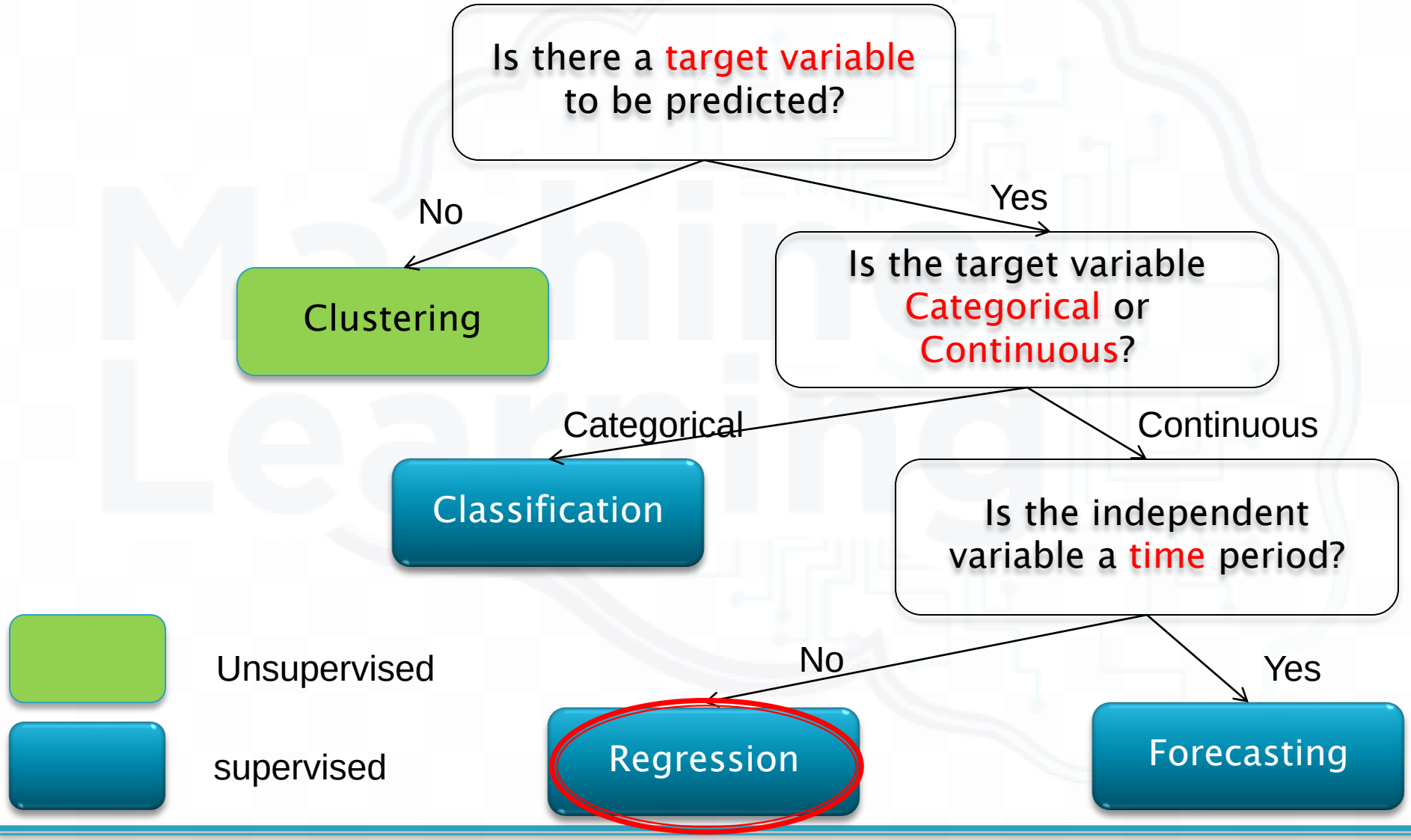
How to use the training set?

- Learn a representation R, so that R is a good predictor for the corresponding value of y



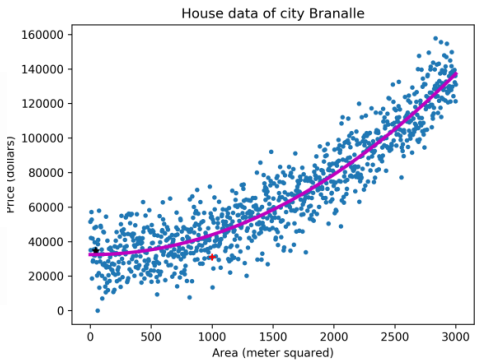
***R*: often called a hypothesis h**

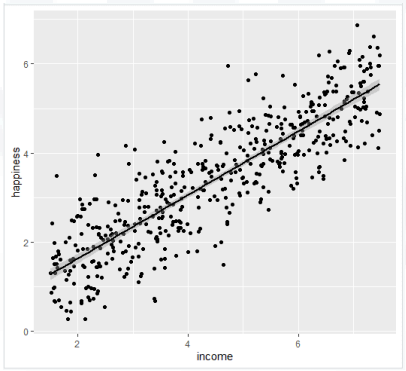
Supervised vs Unsupervised Learning

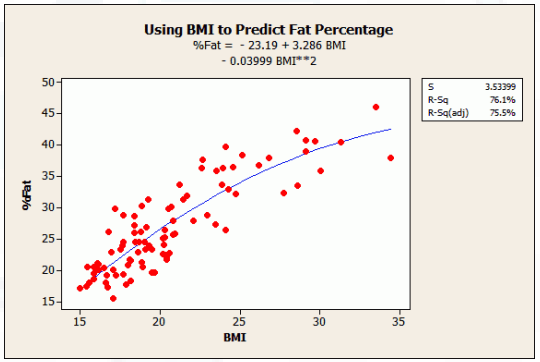


What is regression?

- What do all these problems (predicting prices of houses, votes, movie ratings) have in common?
  - Continuous outputs, we'll call these y
- Predicting continuous outputs is called regression.







What is regression? (cont.)

- What do we need in order to predict these outputs?
  - Features (inputs), we'll call these x (or **x** if vectors)
  - Training examples, many $x^{(i)}$ for which $y^{(i)}$ is known (e.g., many movies for which we know the rating)
  - A model, a function that represents the relationship between x and y
  - A loss/a cost/an objective function, which tells us how well our model approximates the training examples
  - Optimization, a way of finding the parameters of our model that minimizes the loss function

Notations

- x, y, N, k: lower/uppercase, scalar (italic or not)
- **x**; **y**: bold, lowercase, vector
- **X**;**Y**: bold, uppercase, matrix
- $\mathbb{R}$: set of real numbers
- $\mathbb{N}$: set of natural numbers
- $\mathbb{R}^m$: set of vectors with m real numbers
- $\mathbb{R}^{mn}$: set of matrices with m rows and n columns
- $\mathbf{A}^T$: Transpose of matrix A
- …

Represent the Data

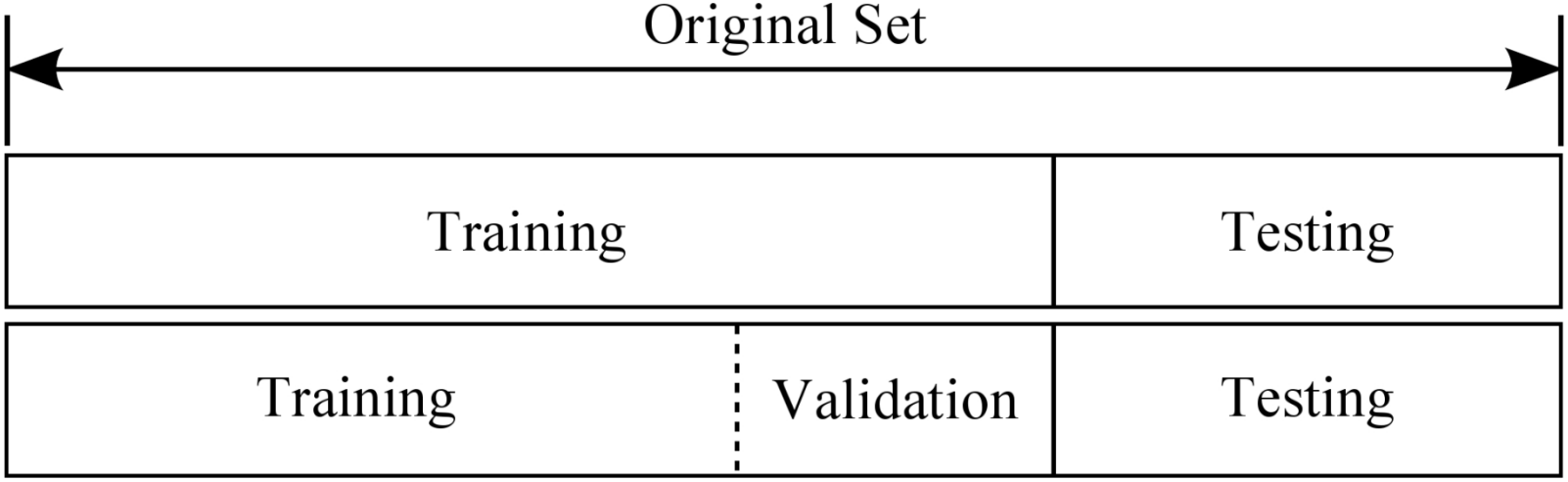
- Data $D = \{D_1, D_2, \ldots, D_n\}$; $D_i = \langle \mathbf{x}^{(i)}, y^{(i)} \rangle$; $i = 1, 2, \ldots, n$; set of n examples
  - $\mathrm{x}^{(i)} = \left(x_1^{(i)}, x_2^{(i)}, \ldots, x_m^{(i)} \right)$: an input vectors (features)
  - $y_i \in \mathbb{R}$: the target output (countinous) ➔ **Regression**
  - i: indicates the training examples (we have n in this case)

- **Objective**: learn the mapping $f: X \rightarrow Y$

  s.t. $y^{(i)} \approx f(\mathbf{x}^{(i)})$ for all $i = 1, 2, \ldots, n$

Represent the Data (cont.)

- Divide the dataset into training and testing examples
  - Use the training examples to construct hypothesis, or function approximator, that maps x to predicted y
  - Evaluate hypothesis on testing set

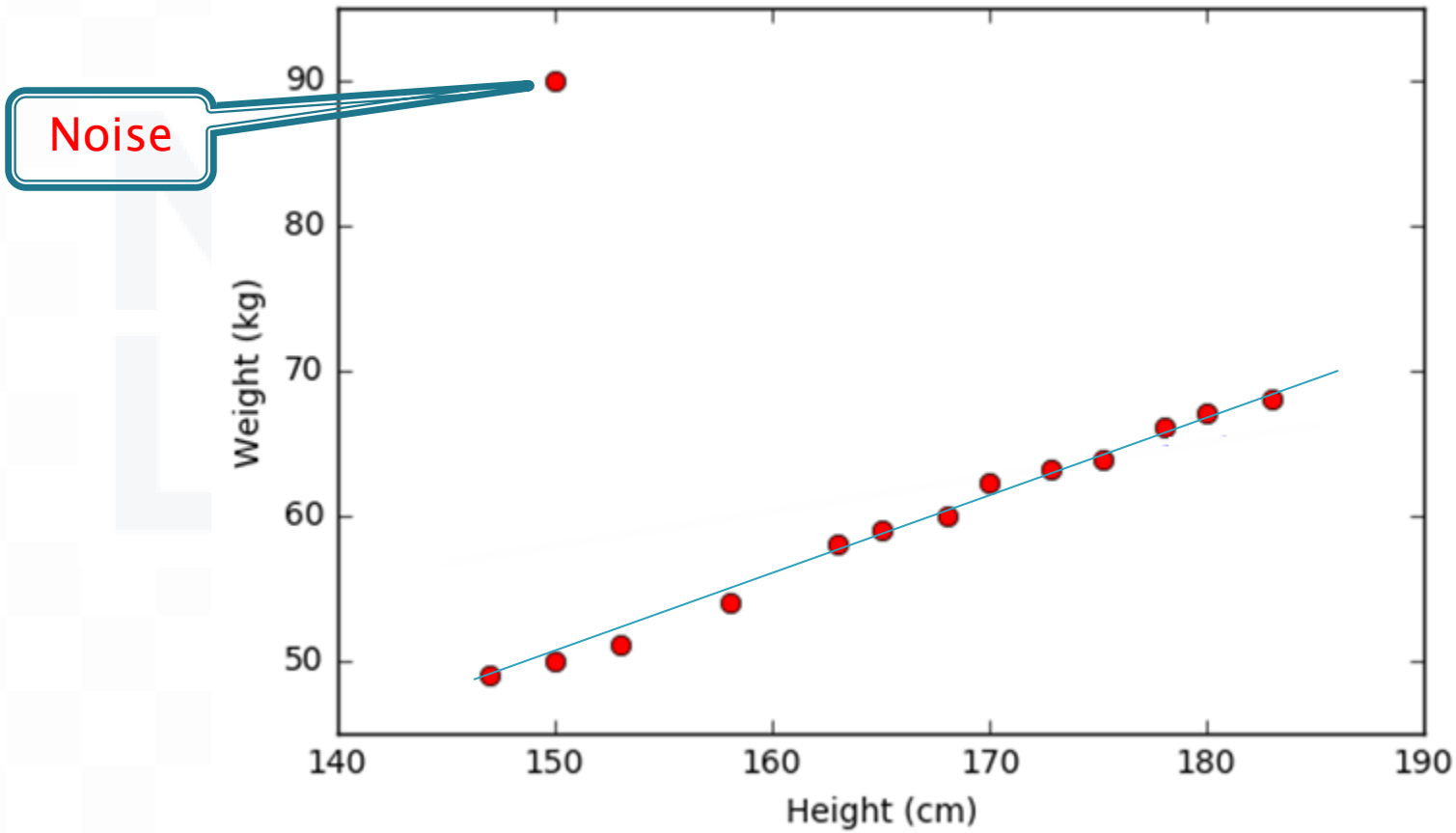


Noise

- A simple model typically does not exactly fit the data
  - lack of fit can be considered noise
- Sources of noise:
  - Imprecision in data attributes (input noise, e.g., noise in per-capita crime)
  - Errors in data targets (mis-labeling, e.g., noise in house prices)
  - Additional attributes not taken into account by data attributes, affect target values (latent variables).
    - E.g., what else could affect house prices?
  - Model may be too simple to account for data targets

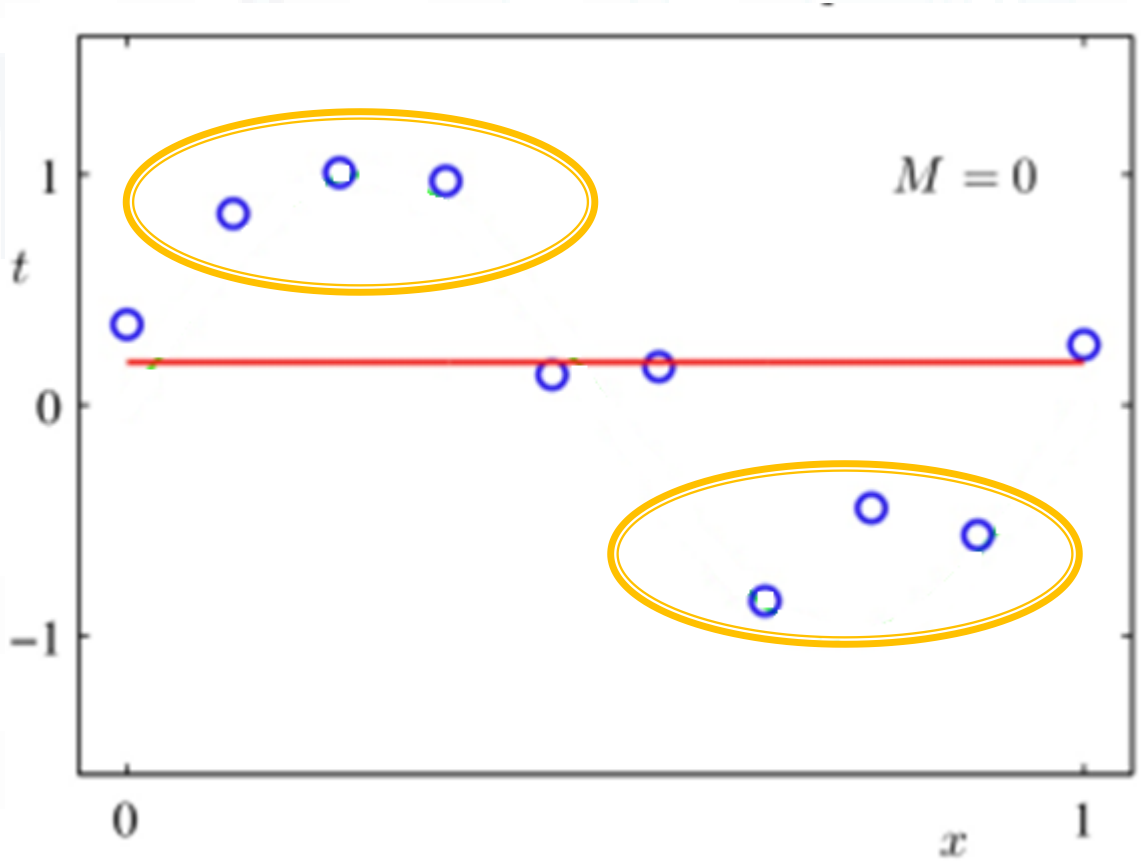
Noise (cont.)

- Linear regression: weight and height



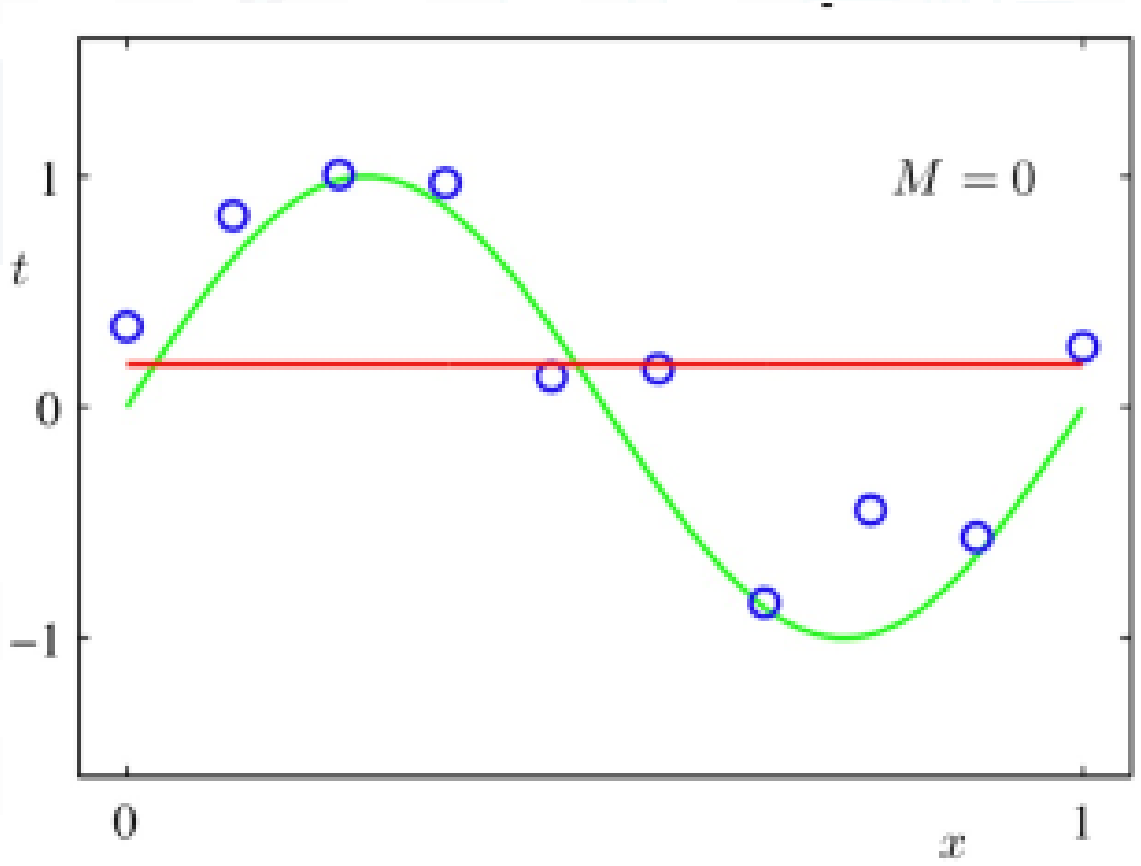
Noise (cont.)

- How about the data points (not in the red line)?

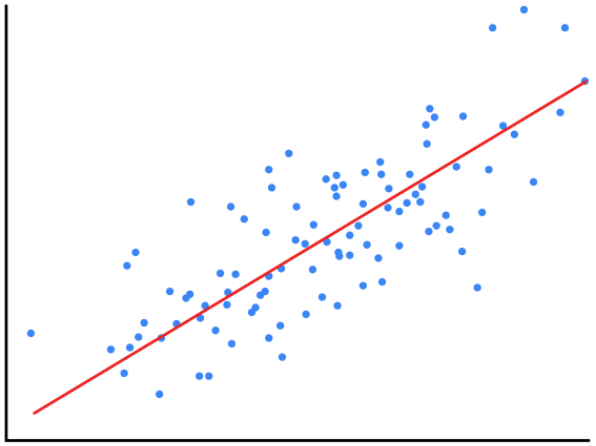


Noise (cont.)

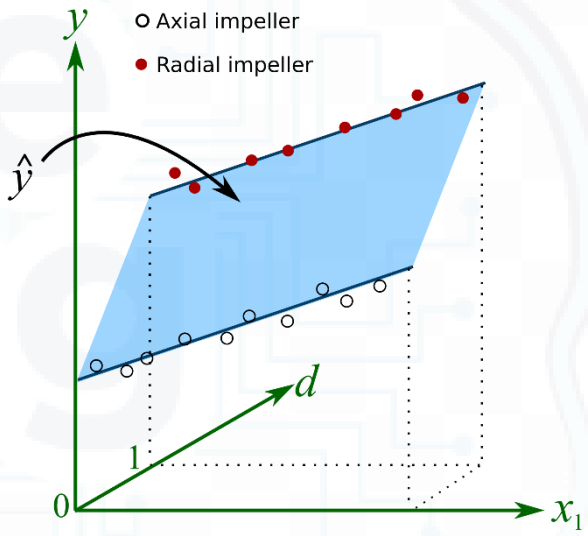
- How about the data points (not in the red line)?

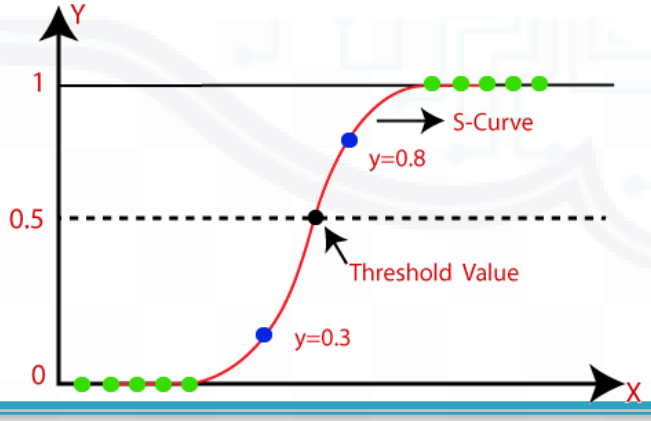


Types of regression

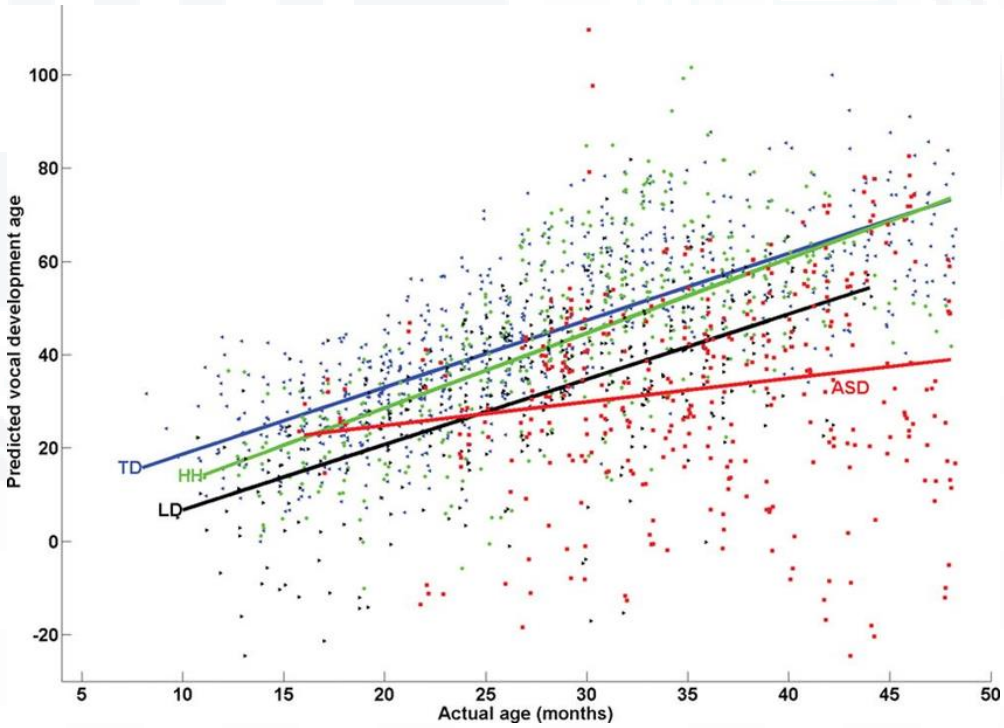


- Linear regression (Univariate)
- Linear Regression for multiple variables
- Logistic regression





Linear Regression

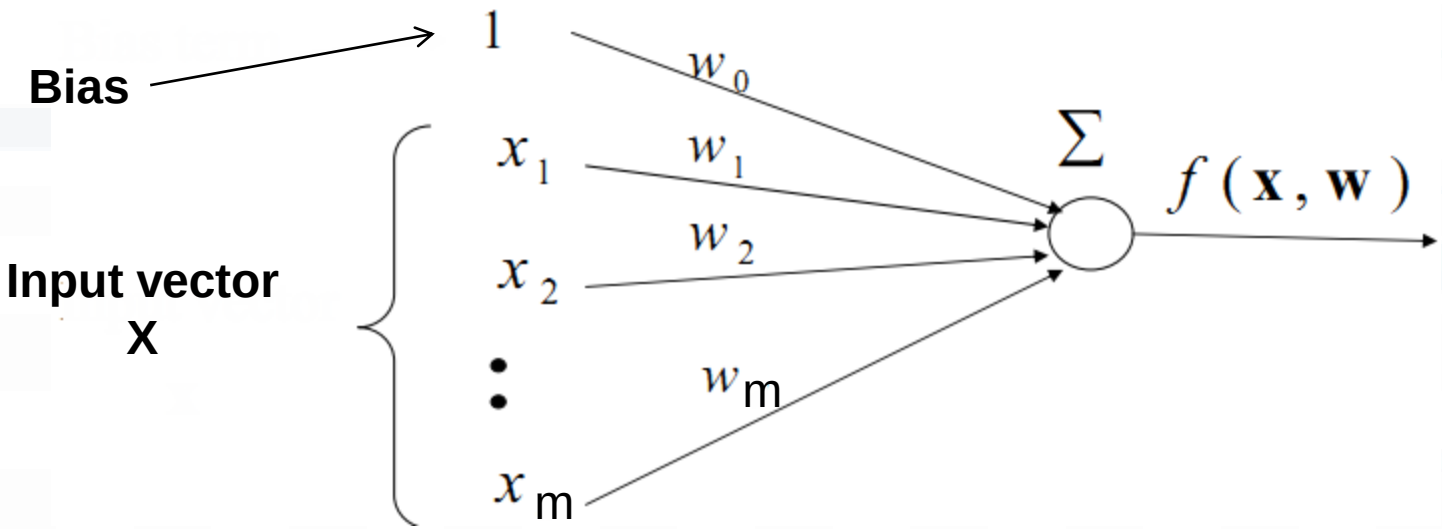


Linear regression

- Function $f: X \rightarrow Y$ is a linear combination of input components

$$f(\mathrm{x}) = w_0 + w_1 x_1 + \ldots + w_m x_m = w_0 + \sum_{j=1}^{m} w_j x_j$$

Where $w_0, w_1, \ldots, w_m$ – parameters (weights)

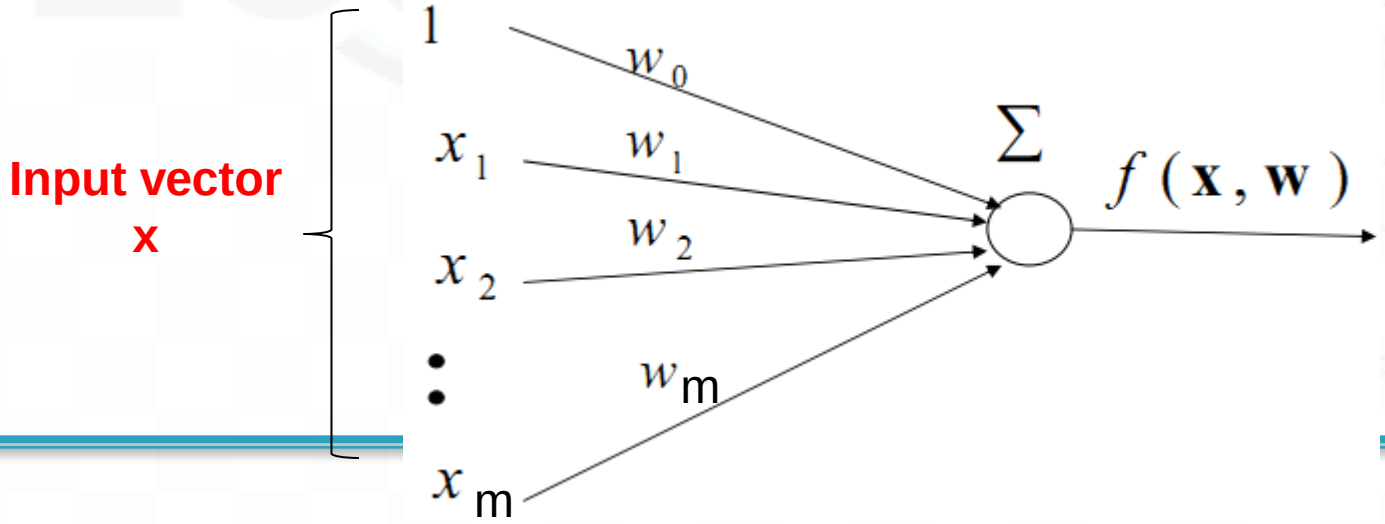


Linear regression (cont.)

- Shorter (vector) definition of the model
  - Include bias constant in the input vector
  - $\mathbf{x}=(1, x_1, x_2, \ldots, x_m)$

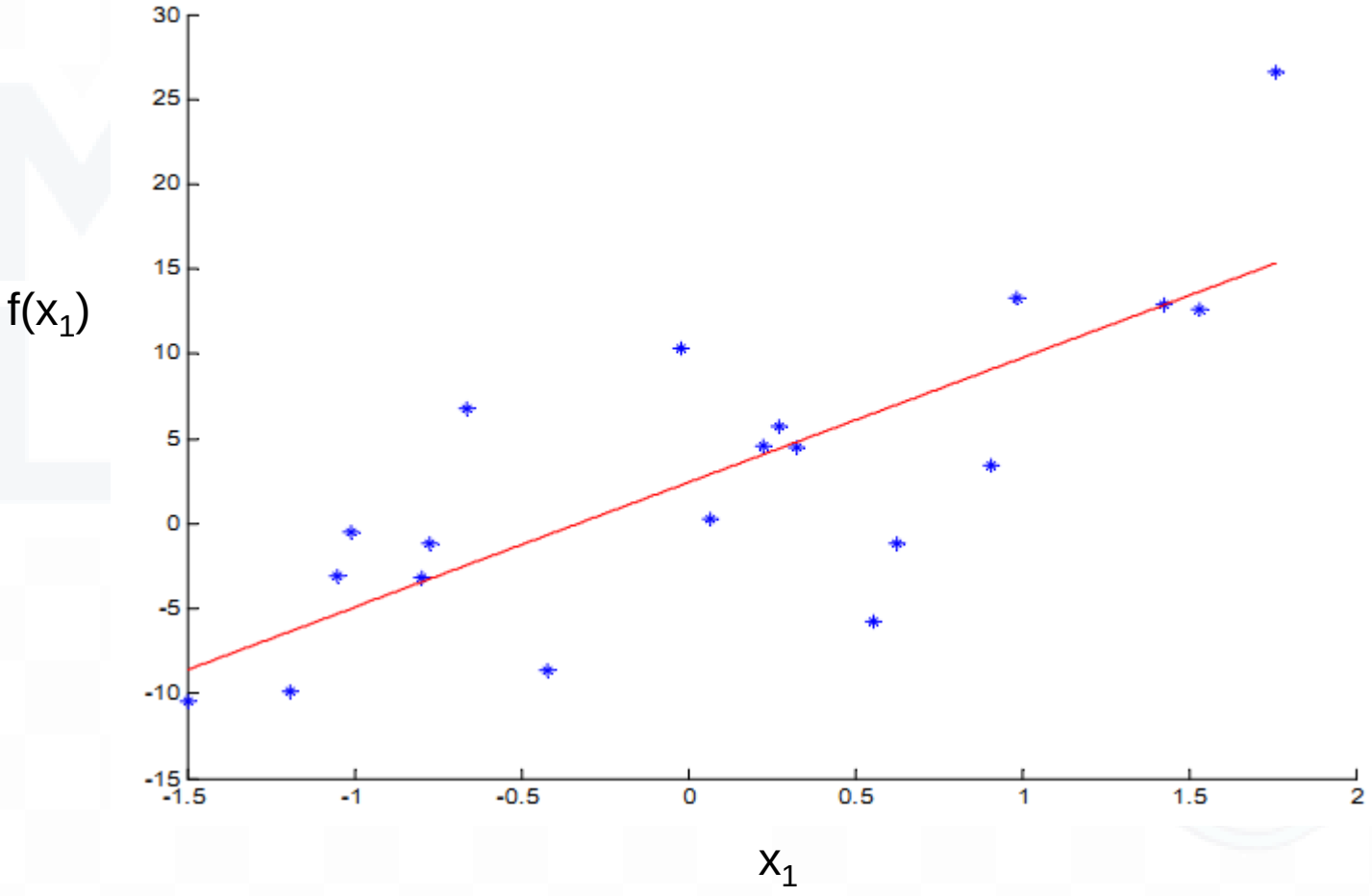
$$f(\mathrm{x}) = w_0 x_0 + w_1 x_1 + \ldots + w_m x_m = w_0 + \sum_{j=1}^{m} w_j x_j$$

Where $w_0, w_1, \ldots, w_m$ – parameters (weights)



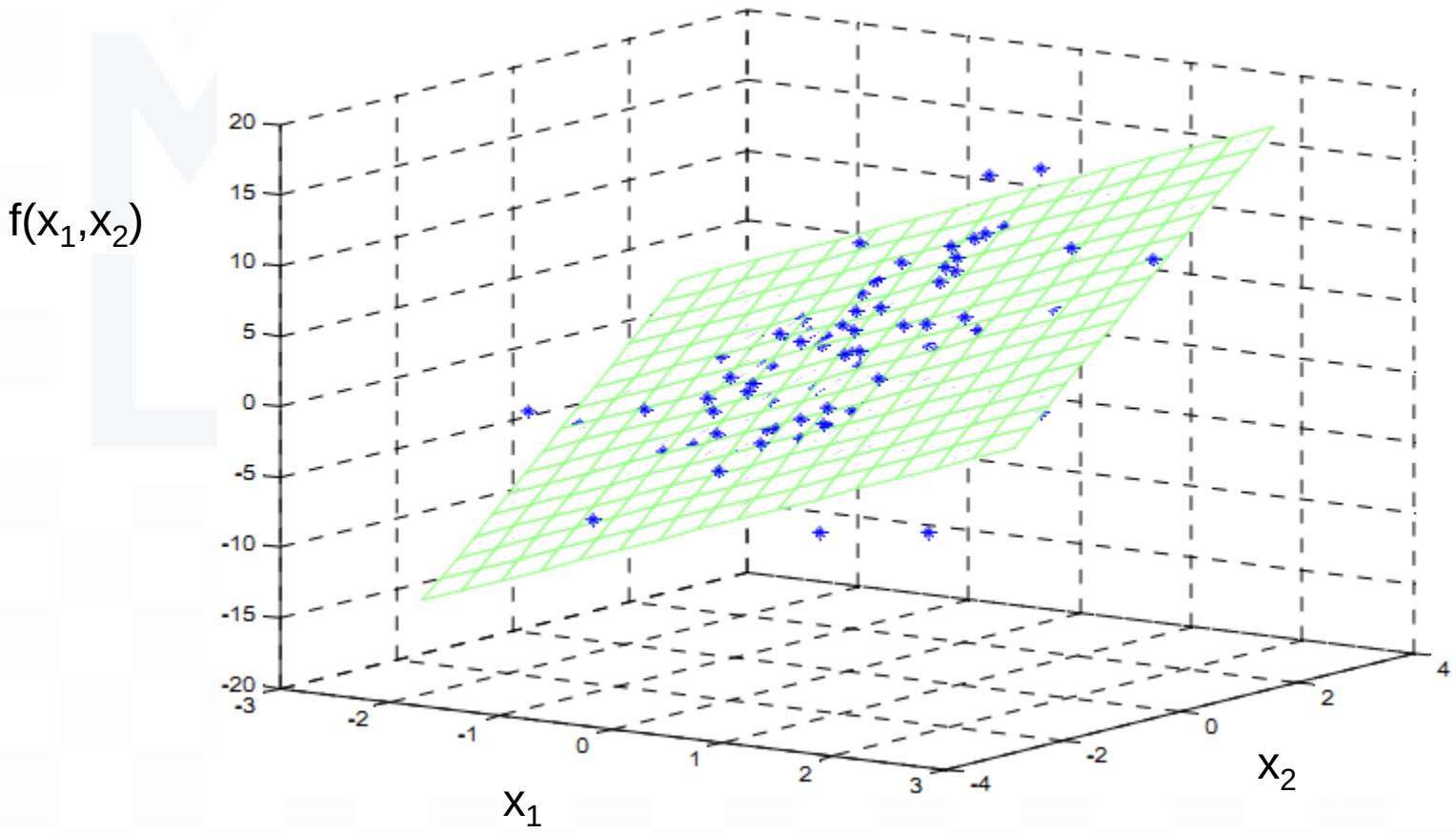
Linear regression: Example 1

- 1 dimensional input: $x=(x_1)$



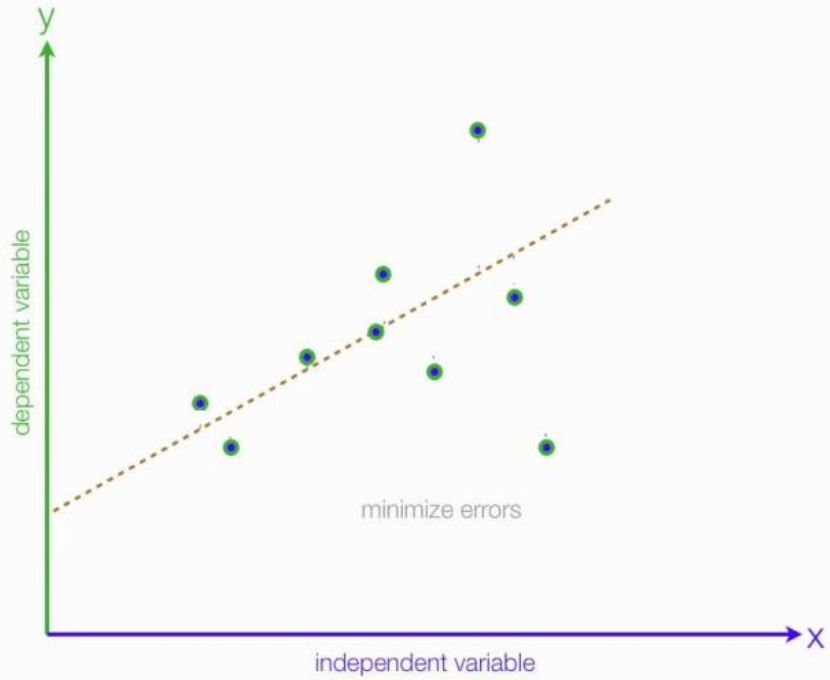
Linear regression: Example 2

- 2 dimensional input: $x=(x_1, x_2)$



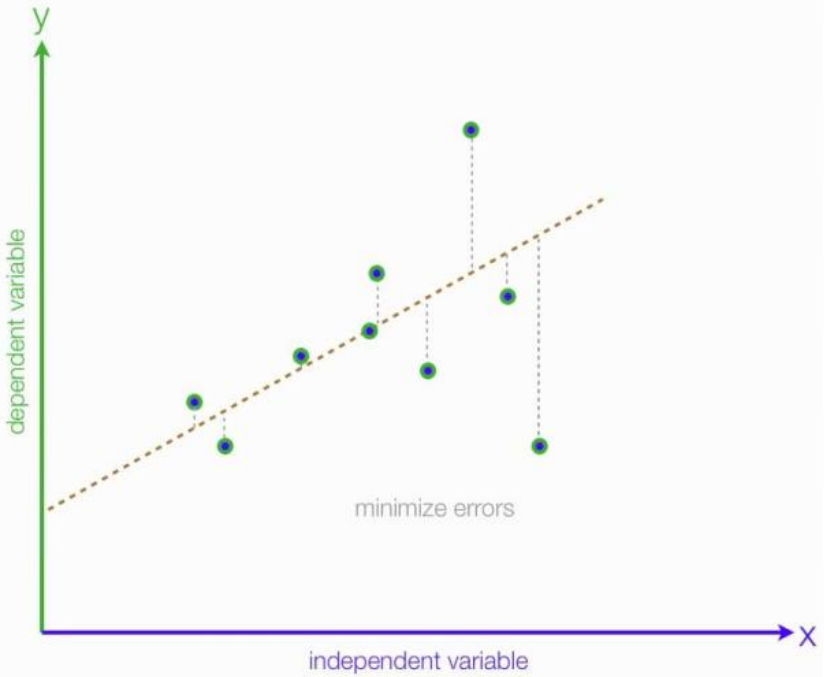
How to evaluate the goodness of a trained model?

- (x, y): a training example
- (x, h(x)): prediction of the model
- (h(x) – y): prediction error for this particular training example
- Minimize the prediction error across all training examples



How to evaluate the goodness of a trained model?

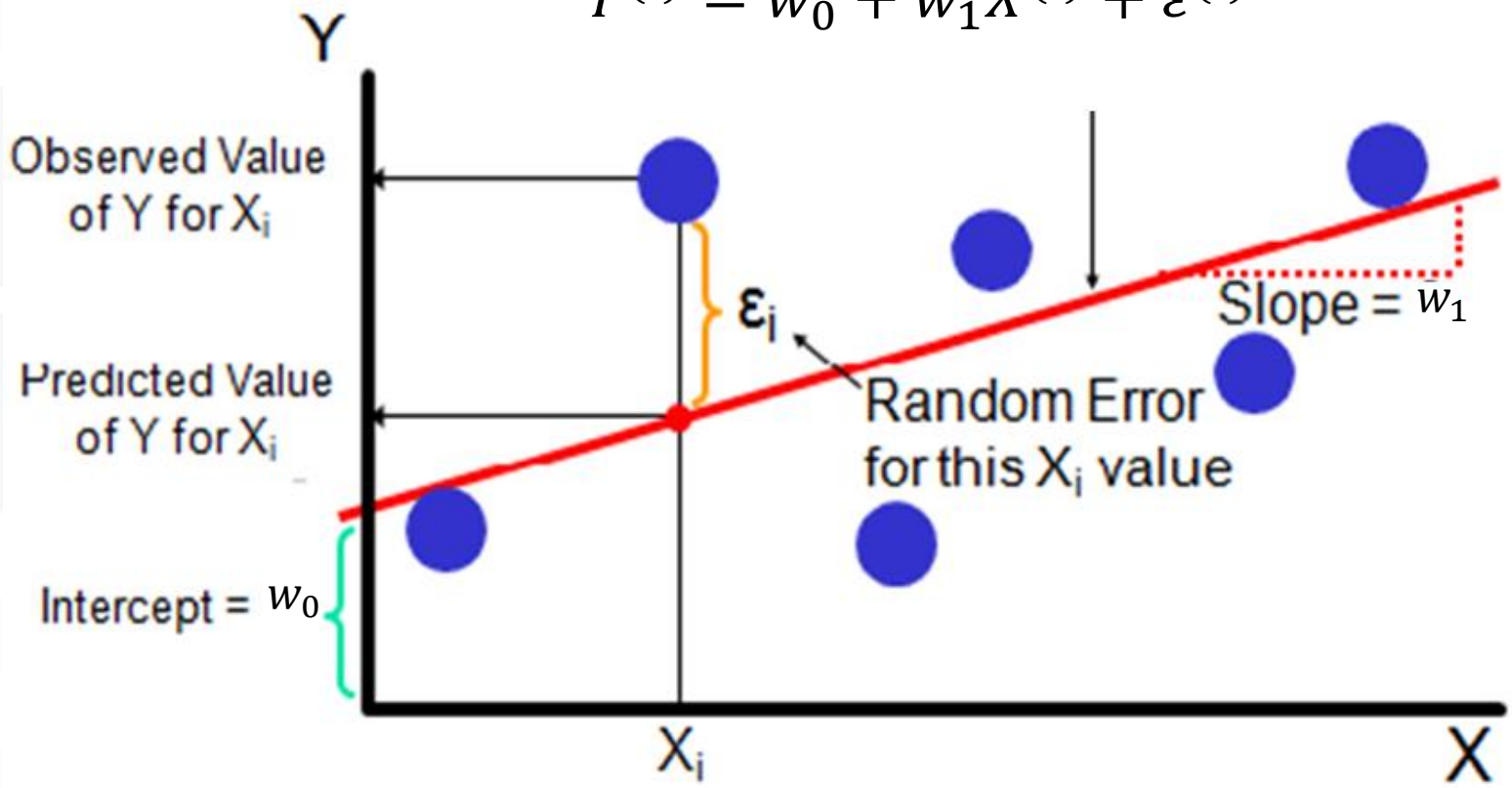
- (x, y): a training example
- (x, h(x)): prediction of the model



- (h(x) – y): prediction error for this particular training example
- Minimize the prediction error across all training examples

Minimize the prediction error

$$Y^{(i)} = w_0 + w_1 X^{(i)} + \varepsilon^{(i)}$$



Performance measures

- Mean absolute error:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y^{(i)} - f(x^{(i)})\right|$$

- Root Mean Squared Error:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - f(x^{(i)})\right)^2}$$

- Relative absolute error:

$$RAE = \frac{\sum_{i=1}^{n}\left|f(x^{(i)} - y^{(i)})\right|}{\sum_{i=1}^{n}|\bar{y} - y^{(i)}|}$$

- Root relative squared error:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}\left(y^{(i)} - f(x^{(i)})\right)^2}{\sum_{i=1}^{n}(\bar{y} - y^{(i)})^2}}$$

Loss function

- Measure of how close the predictions are to the actual y-values
- Average over all the n training instances
- Squared error cost function L(w):

$$L(\mathrm{w}) = \frac{1}{2n}\sum_{i=1}^{n}\left(y^{(i)} - f\left(\mathrm{x}^{(i)}\right)\right)^2 = \frac{1}{2n}\sum_{i=1}^{n}\left(y^{(i)} - \mathrm{w}^{\mathrm{T}}\mathrm{x}^{(i)}\right)^2$$

- Choose parameters w so that L(w) is minimized

Minimizing a function

- For now, let us consider some arbitrary function (not necessarily a cost function)
- Analytical minimization not scalable to complex functions of hundreds of parameters
- Algorithm called gradient descent
  - Efficient and scalable to thousands of parameters
  - Used in many applications of minimizing functions

Minimizing a function (cont.)

- How does gradient descent works?

- Have some function $L(w_0, w_1)$

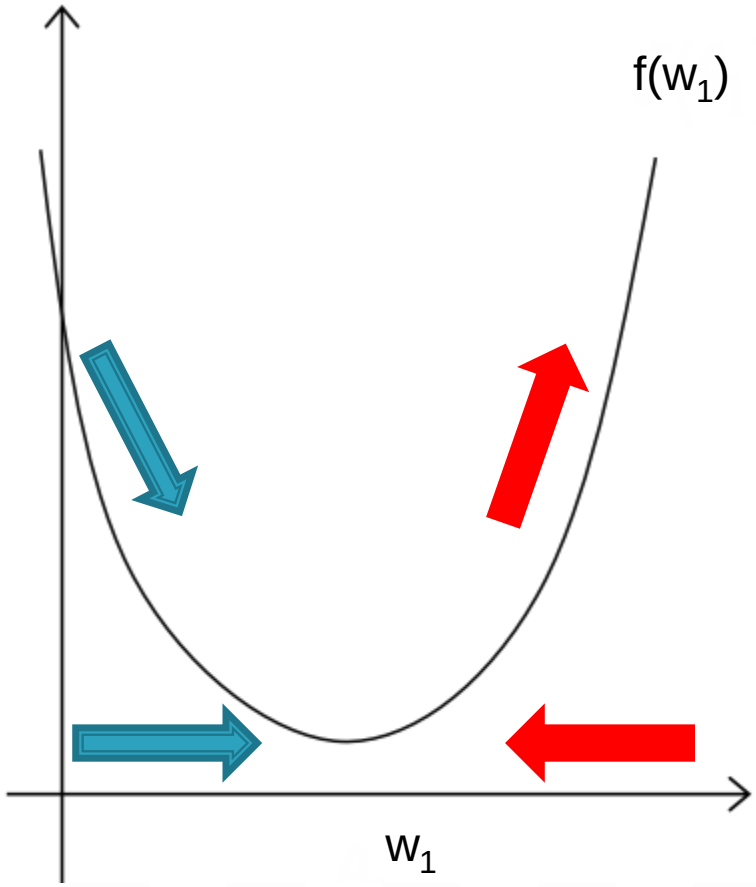
  s.t. min $L(w_0,w_1)$

- Outline:
  - Start with some w_0, w_1
  - Keep changing w_0, w_1 to reduce $L(w_0,w_1)$ until we hopefully end up at a minimum.

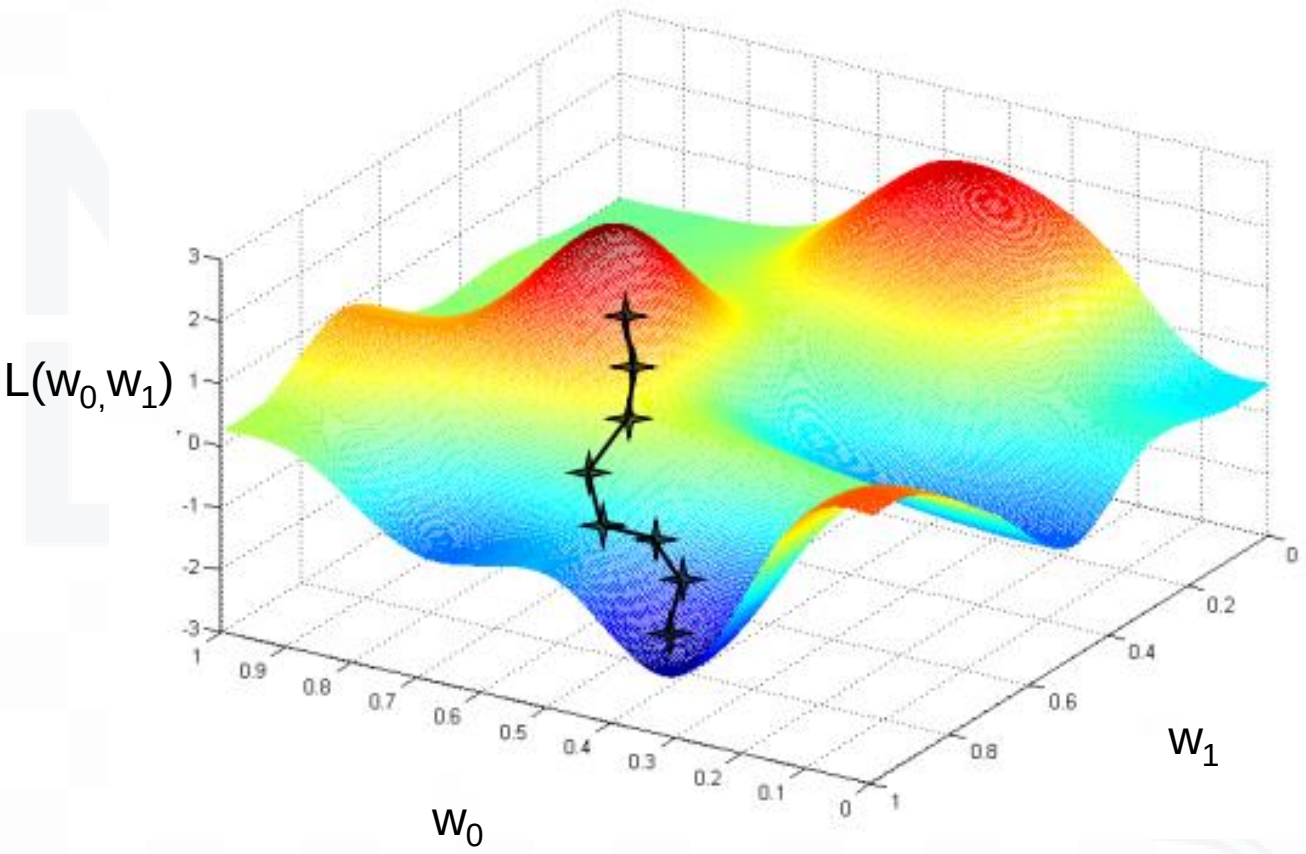
Minimizing a function (cont.)

- For simplicity, let us first consider a function of a single variable
  - If the derivative is positive, reduce value of w_1
  - If the derivative is negative, increase value of w_1

$$w_1 = w_1 - \lambda \frac{\partial f(w_1)}{\partial w_1}$$

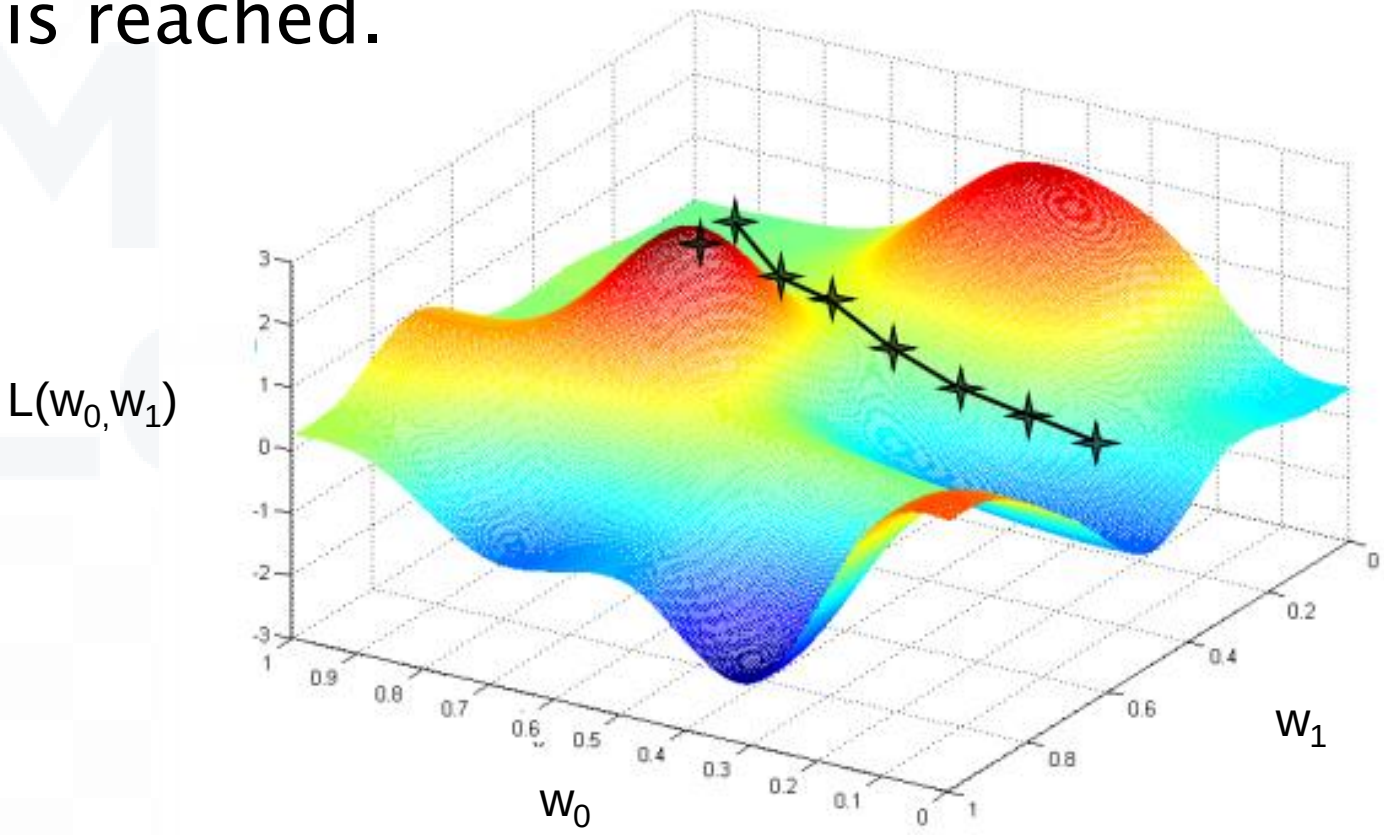


Gradient descent



Gradient descent (cont.)

- If the function has multiple local minima, where one starts can decide which minimum is reached.



Gradient descent (cont.)

- Repeat until convergence {

$$w_j = w_j - \lambda \frac{\partial L(\mathrm{w})}{\partial w_j}$$

}

Simultaneously update j=0, j=1, …, j=m

- λ: the learning rate

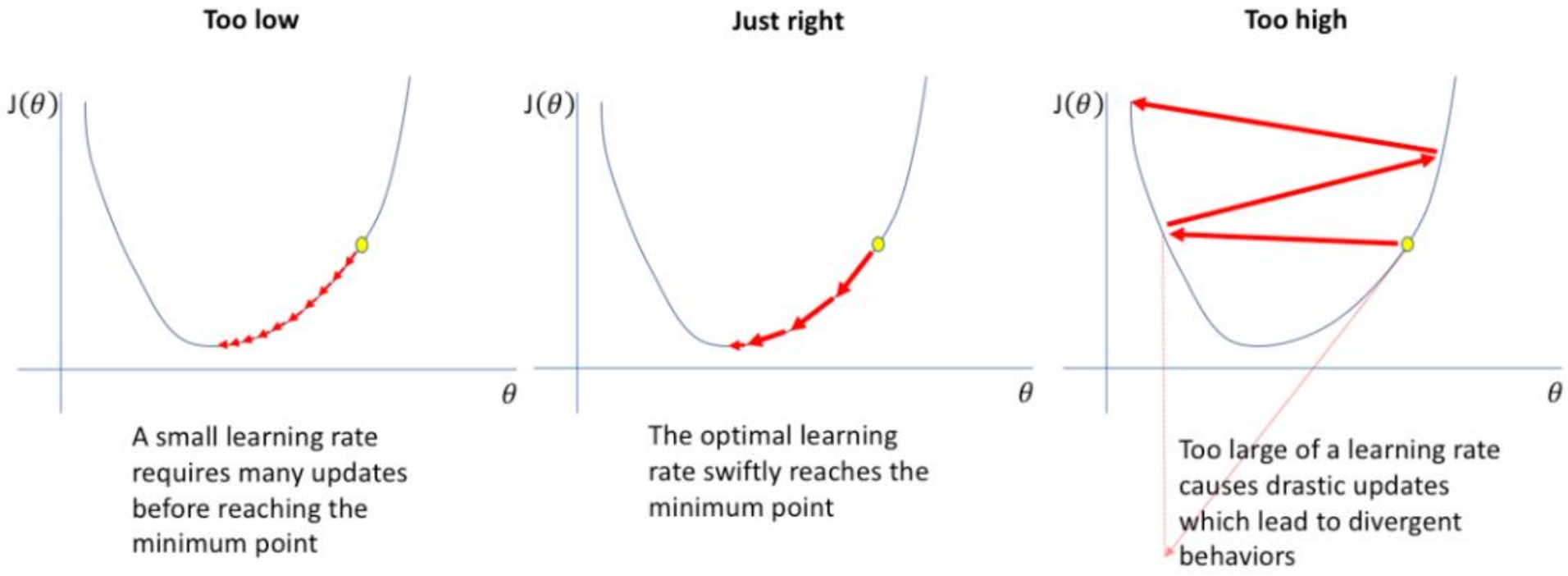
Learning rate

- Do we need to change learning rate over time?
  - No, Gradient descent can converge to a local minimum, even with the learning rate λ fixed
  - Step size adjusted automatically

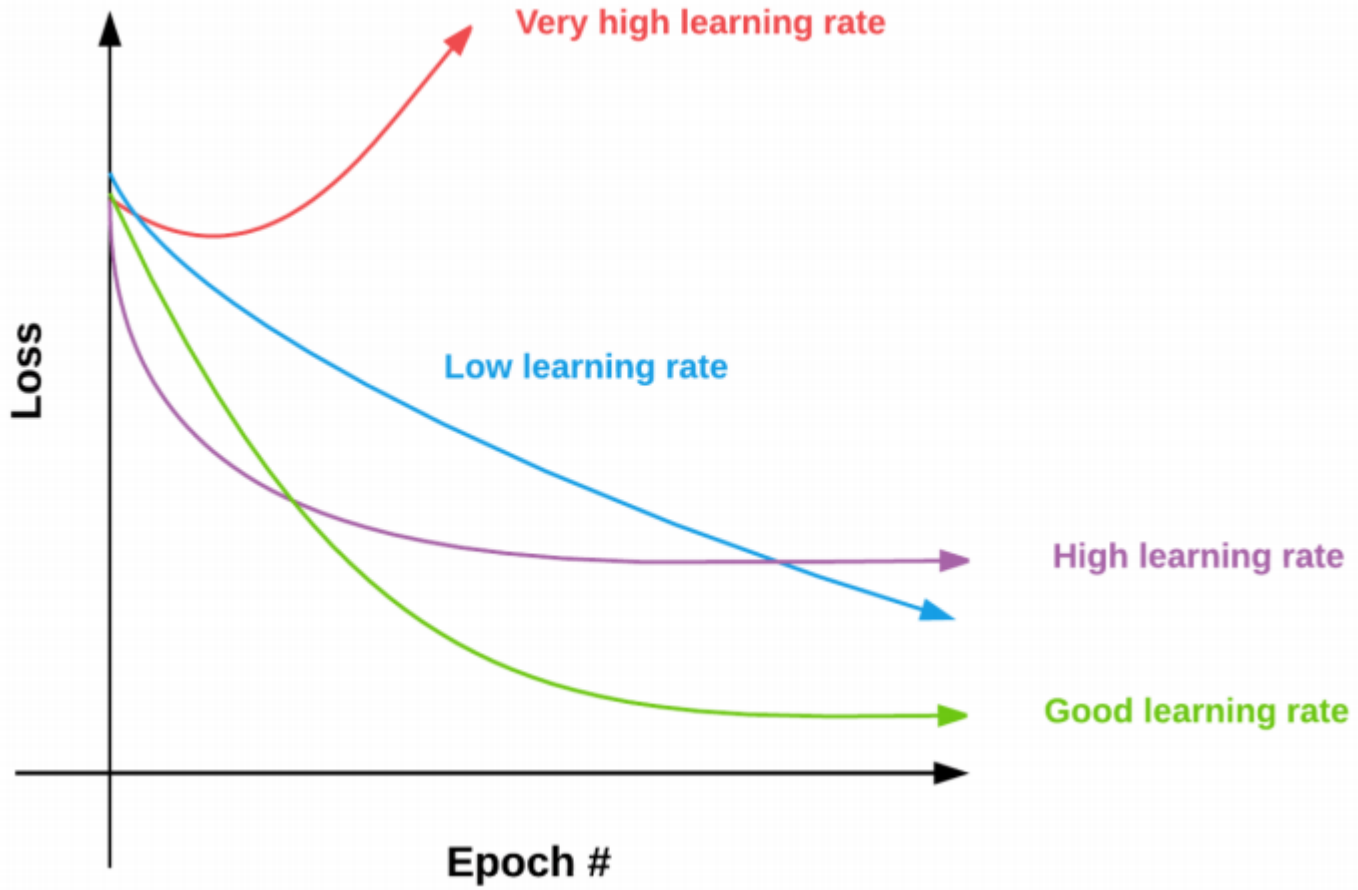
- But, value needs to be chosen judiciously
  - If λ is too small, gradient descent can be slow to converge
  - If λ is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.

Impact of Leaning rate

- Different learning rate values



Impact of Leaning rate (cont.)



Gradient descent: univariate linear regression

- Linear Regression Model: $f(x) = w_0 + w_1 x$

$$L(w) = \frac{1}{2n} \sum_{i=1}^{n} \left(f(x^{(i)}) - y^{(i)} \right)^2$$

- Gradient descent for univariate linear regression

Repeat until convergence {

$$w_j = w_j - \lambda \frac{\partial L(w_0, w_1)}{\partial w_j}$$

}

(for j=0 and j=1)

Gradient descent (cont.)

- Gradient descent for univariate linear regression

Repeat until convergence {

$$w_0 = w_0 - \lambda \frac{1}{n} \sum_{i=1}^{n} \left(f(x^{(i)}) - y^{(i)} \right)$$

$$w_1 = w_1 - \lambda \frac{1}{n} \sum_{i=1}^{n} \left(f(x^{(i)}) - y^{(i)} \right) x^{(i)}$$

}

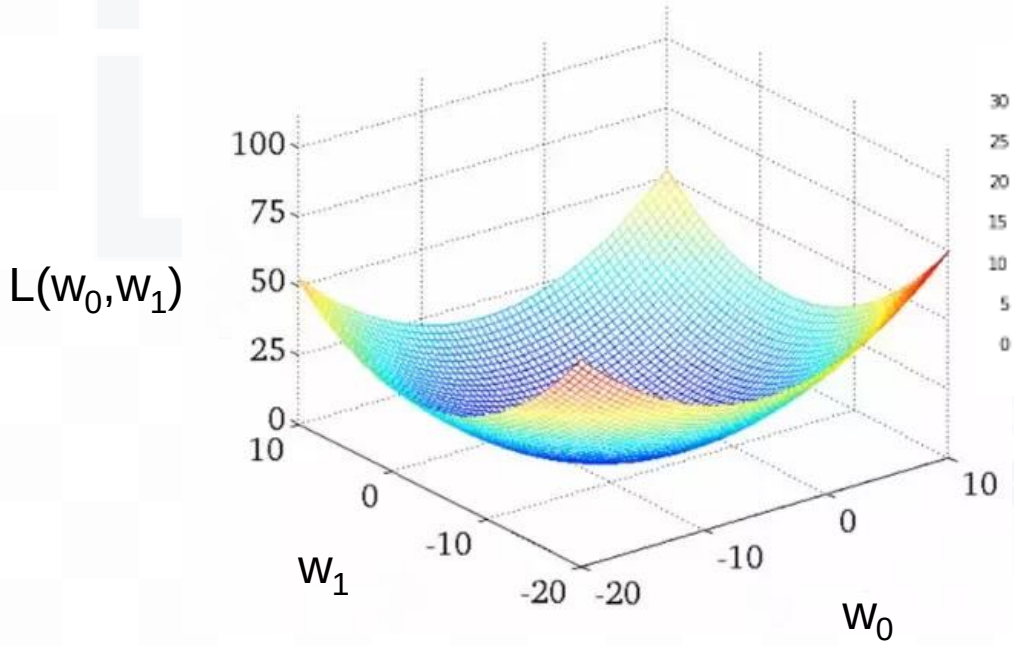
“Batch” Gradient Descent

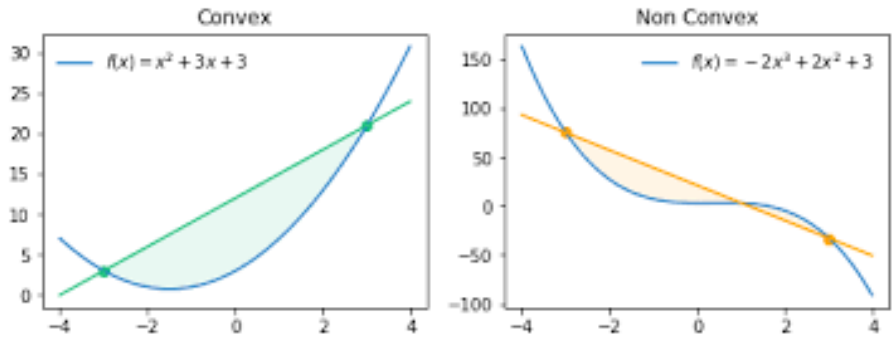
- “Batch”: Each step of gradient descent uses all the training examples.

- There are other variations like “stochastic gradient descent” (used in learning over huge datasets)

What about multiple local minima?

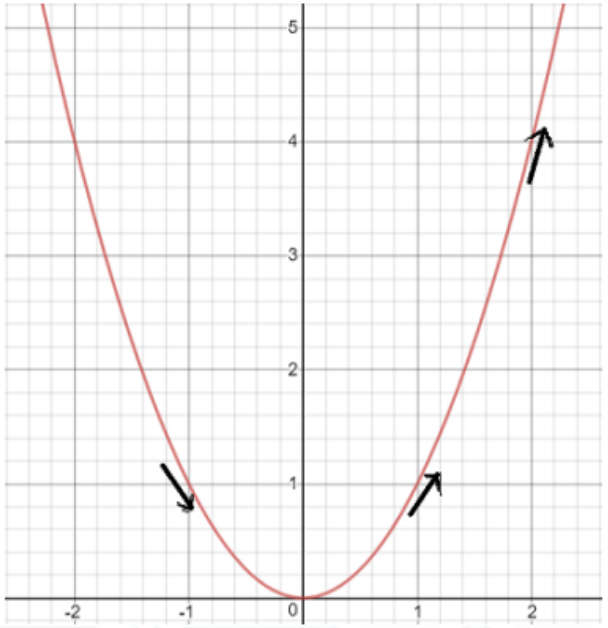
- The loss function in linear regression is always a convex function – always has a single global minimum
- So, gradient descent will always converge





Gradient descent: Example

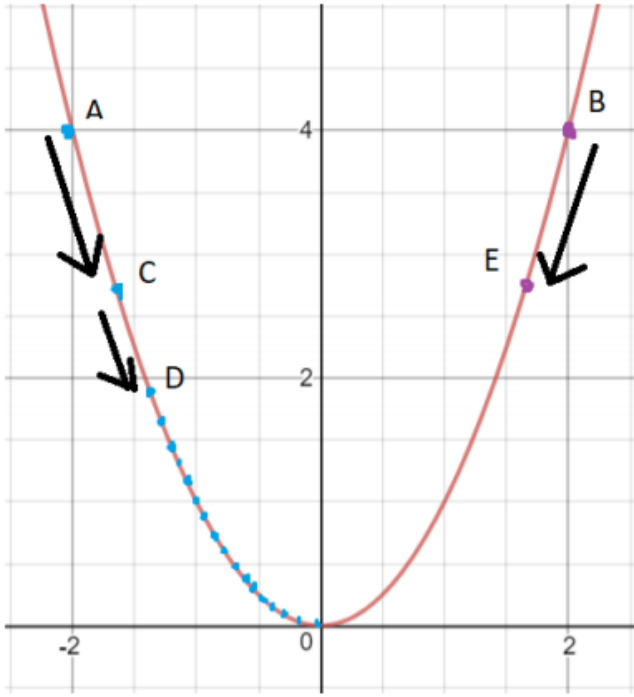
- For a given function $f(x) = x^2$
- Derivative of f(x): $f'(x)=2x$
- $\lambda=0.001$
- Gradient descent:
  - **Step 1**. Init random $x=x_0$
  - **Step 2**. $x=x-\lambda f'(x)$
  - **Step 3**.
    - If f(x) is minimal enough, STOP.
    - Else, return to Step 2.



Gradient descent: Example

- **Step 1**: suppose $x = -2$ (A)
- **Step 2**: $x = x - \lambda f'(x)$

  $-2 - 0.01 * 2 * (-2) = 1.96$
- **Step 3**: return to Step 2 until
  - f(x) small enough or
  - MAX_ITERATION is reached



Gradient descent: Example

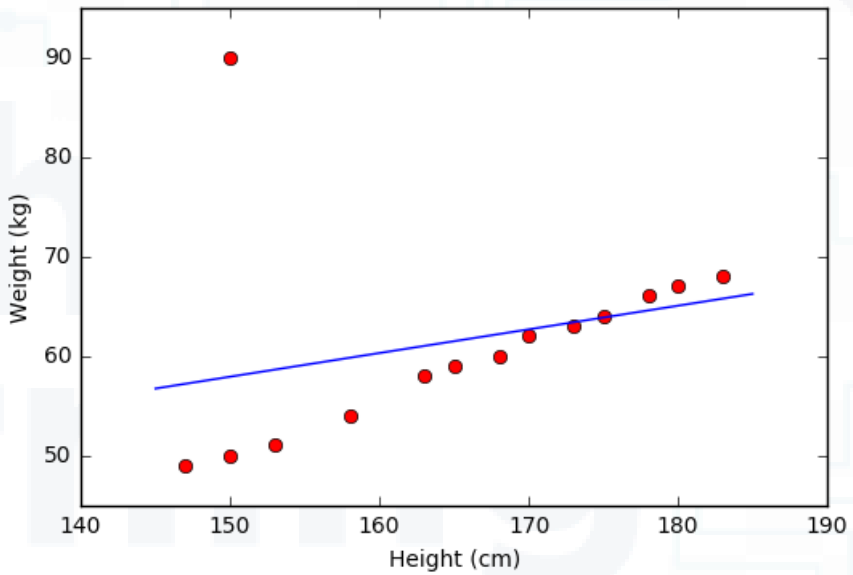
- Suppose $x_0=10$, $f(x)=x^2$

	A	B	C	D
1			**Learning Rate:**	**0.1**
2		x	f(x)	
3	No	10.00	100.00	
4	1	8.00	64.00	
5	2	6.40	40.96	
6	3	5.12	26.21	
7	4	4.10	16.78	
8	5	3.28	10.74	
9	6	2.62	6.87	
10	7	2.10	4.40	
11	8	1.68	2.81	
12	9	1.34	1.80	
13	10	1.07	1.15	

	A	B	C	D
1			**Learning Rate:**	**0.01**
2		x	f(x)	
3	No	10.00	100.00	
4	1	9.80	96.04	
5	2	9.60	92.24	
6	3	9.41	88.58	
7	4	9.22	85.08	
8	5	9.04	81.71	
9	...	...	...	
10	99	1.35	1.83	
11	100	1.33	1.76	

Limitation of Linear Regression

- Sensitive to noise. Consider the following example:



- Data must be independent
  - any **multicollinearity** must be removed before applying linear regression.

Multicollinearity

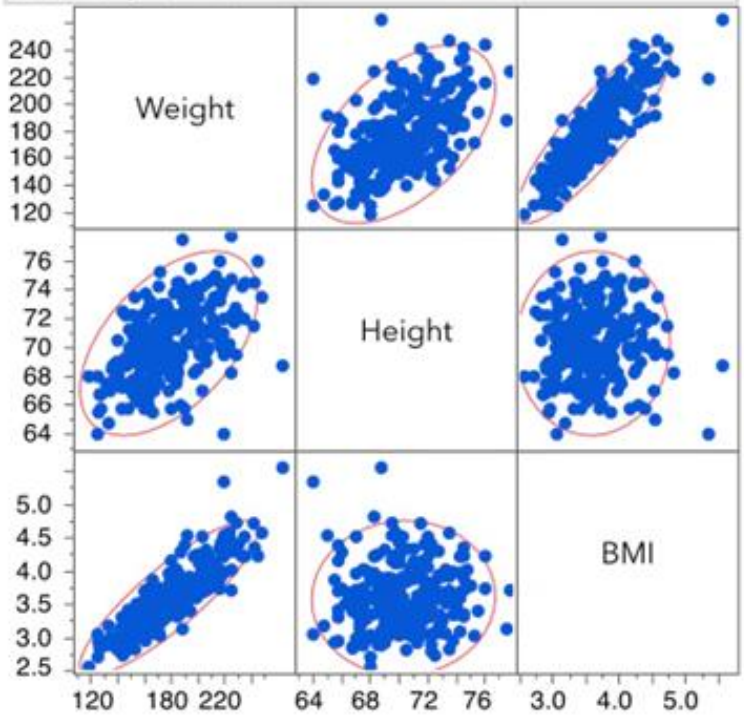
- Multicollinearity: two or more independent variables in a regression model are correlated

Multivariate

Correlations

	Weight	Height	BMI
Weight	1.0000	0.5129	0.8668
Height	0.5129	1.0000	0.0220
BMI	0.8668	0.0220	1.0000

Scatterplot Matrix



Exercise 1

- Using gradient descent to find the minimal value of the following function ($\lambda=0.01$ và $\lambda=0.1$):

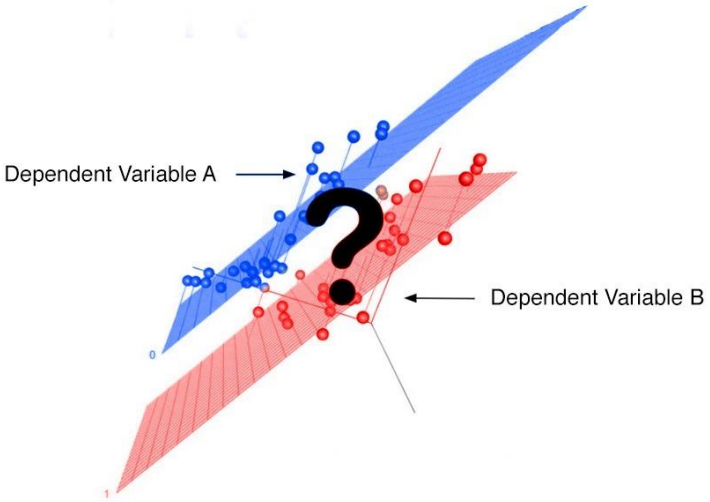
$$f(x) = x^2+2x+5$$

Exercise 2

- Determine the linear regression model for the following dataset.

x	y
1	1
2	3
4	3
3	2
5	5

Multivariate Linear Regression



Multiple features (variables)

- How can we learn to predict the prices of houses using the following given dataset?

Size (feet^2)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

Multiple features (variables)

Size (feet^2)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

- Notation:
  - m: number of features. n: number of training examples
  - $x^{(i)}$: input (features) of i^{th} training example.
  - $x^{(i)}_j$: value of feature j in i^{th} training example.

Hypothesis

- For univariate linear regression:

$$f(\mathrm{x}) = \mathrm{w}_0 + w_1 x$$

- For multi-variate linear regression:

$$f(\mathrm{x}) = \mathrm{w}_0 + w_1 x_1 + w_2 x_2 + \ldots + w_m x_m$$

$$\mathrm{x} = \left(x_1, x_2, \ldots, x_m \right)$$

Multivariate Linear Regression

- Hypothesis $f(\mathrm{x}) = \mathrm{w}_0 + w_1 x_1 + w_2 x_2 + \ldots + w_m x_m$

- Parameters: $\mathrm{w}_0, \mathrm{w}_1, \mathrm{w}_2, \ldots, \mathrm{w}_m$
- Loss function:

$$L(\mathrm{w}) = L(w_0, w_1, w_2, \ldots w_m) = \frac{1}{2n} \sum_{i=1}^{n} \left(f\left(\mathrm{x}^{(i)}\right) - y^{(i)} \right)^2$$

- Gradient descent:

repeat {

$$w_j = w_j - \lambda \frac{\partial L(w_0, w_1, \ldots, w_m)}{\partial w_j}$$

} Simultaneously update w_j *for* $j = 0,1,2,\ldots,m$

Gradient descent

repeat {

$$w_j = w_j - \lambda \frac{1}{n} \sum_{i=1}^{n} \left(f(\mathrm{x}^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

}

Gradient descent (cont.)

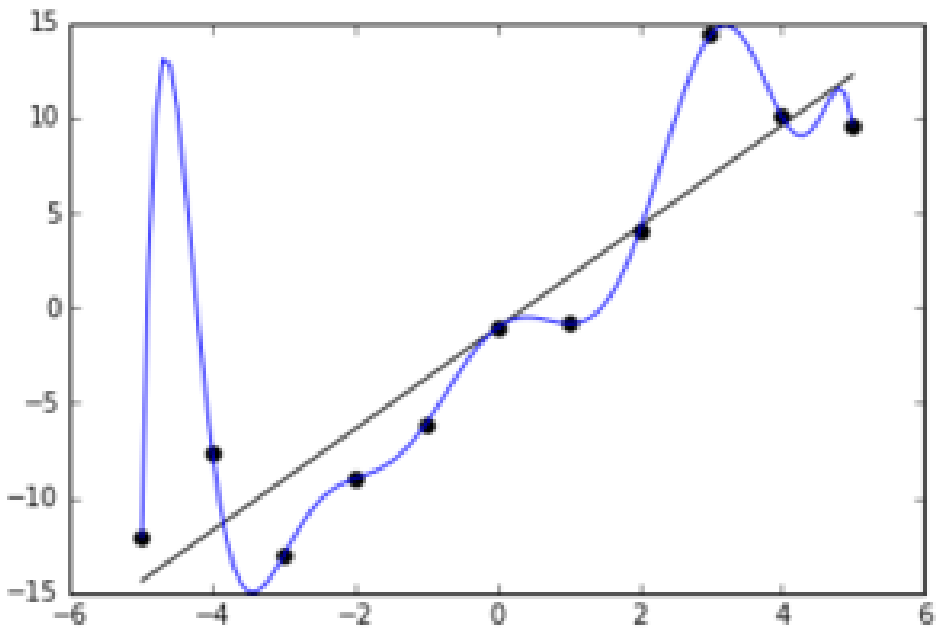
$$w_0 = w_0 - \lambda \frac{1}{n} \sum_{i=1}^{n} \left(f(\mathrm{x}^{(i)}) - y^{(i)} \right) x_0^{(i)}$$

$$w_1 = w_1 - \lambda \frac{1}{n} \sum_{i=1}^{n} \left(f(\mathrm{x}^{(i)}) - y^{(i)} \right) x_1^{(i)}$$

$$w_2 = w_2 - \lambda \frac{1}{n} \sum_{i=1}^{n} \left(f(\mathrm{x}^{(i)}) - y^{(i)} \right) x_2^{(i)}$$

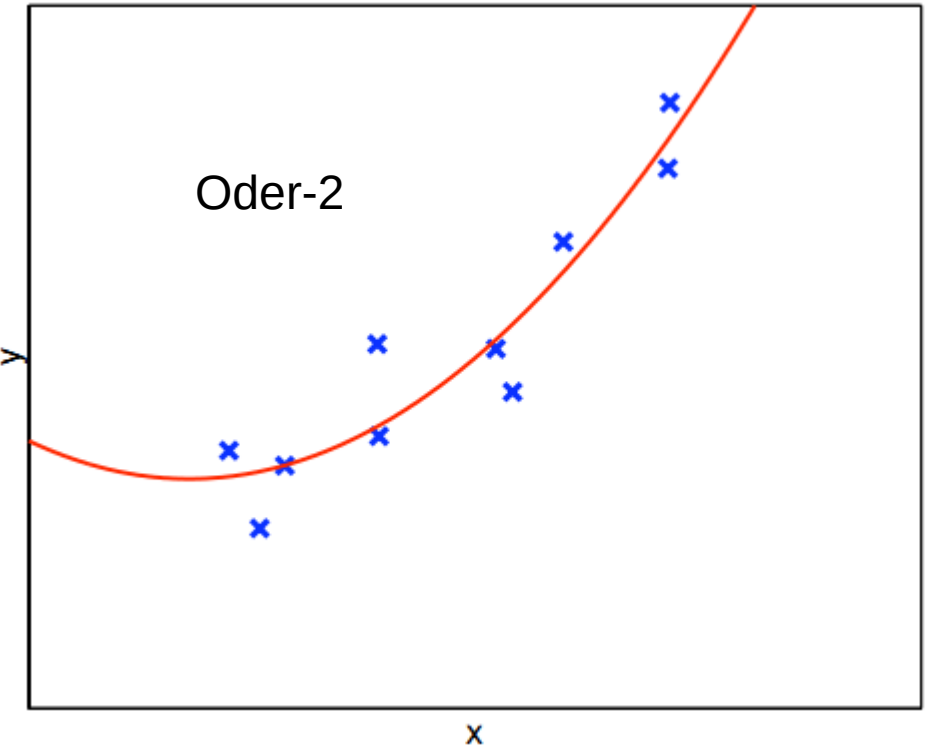
...

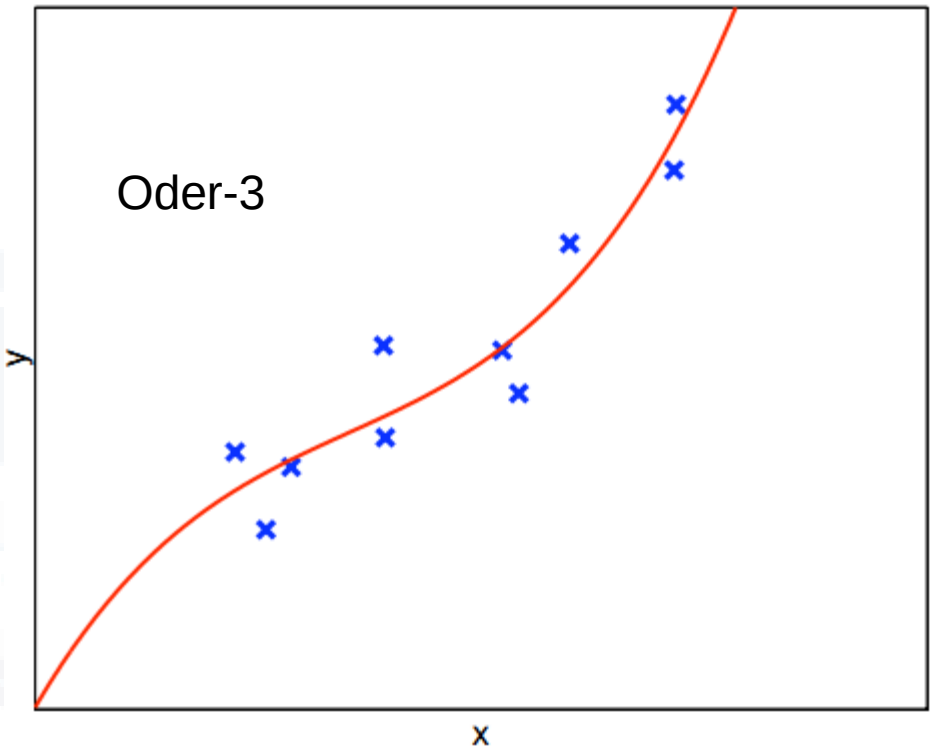
Overfitting



Regression using polynomial curve

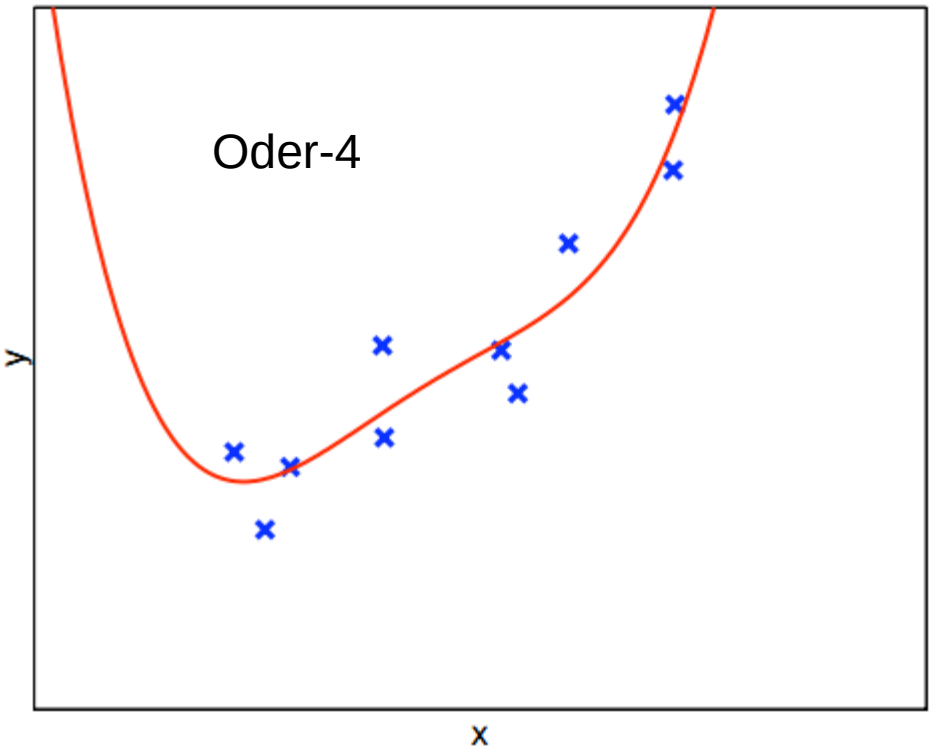
- Which one is the best fit to the data?

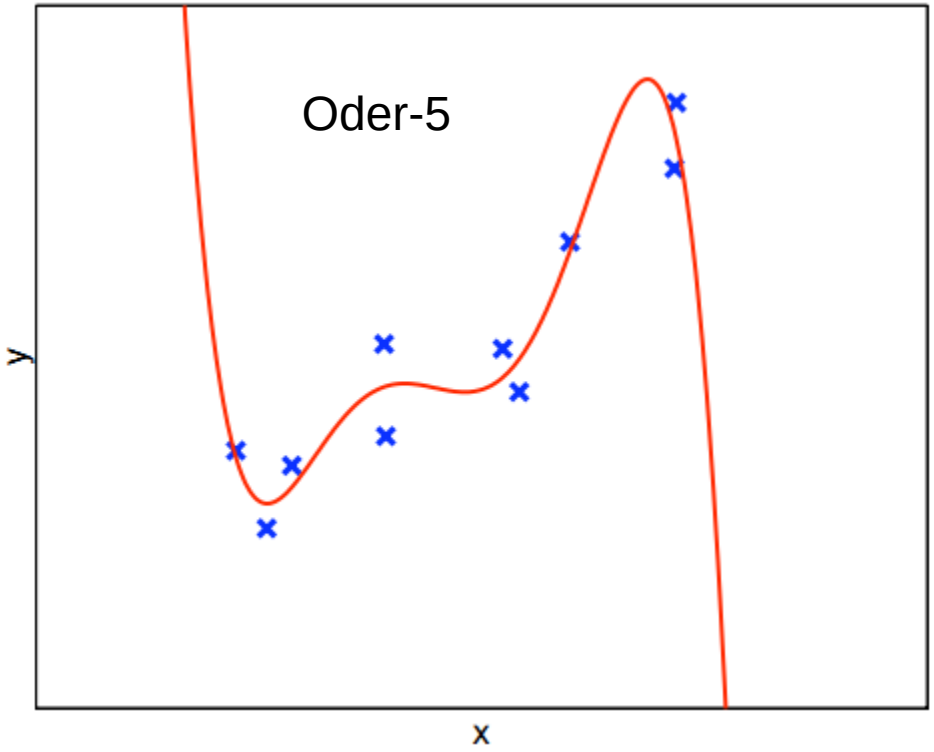




Regression using polynomial curve

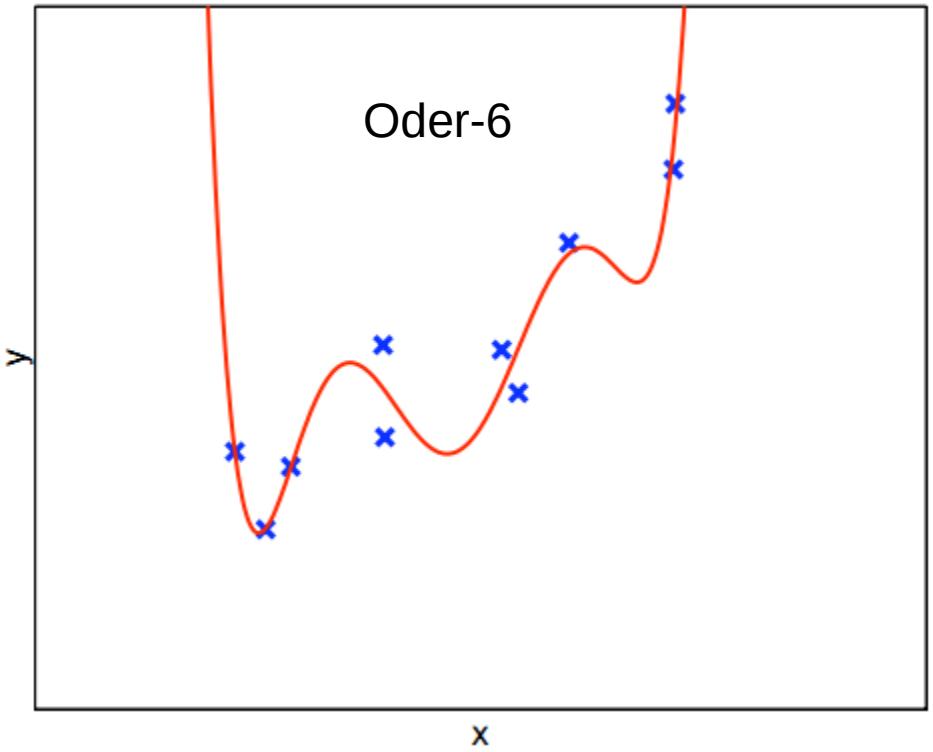
- Which one is the best fit to the data?

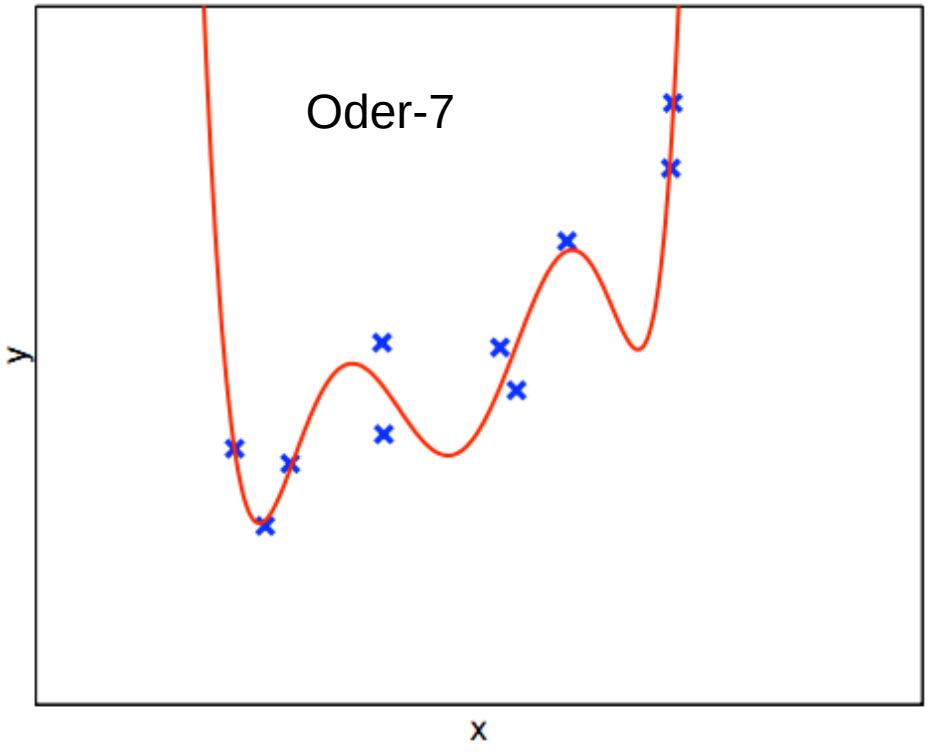




Regression using polynomial curve

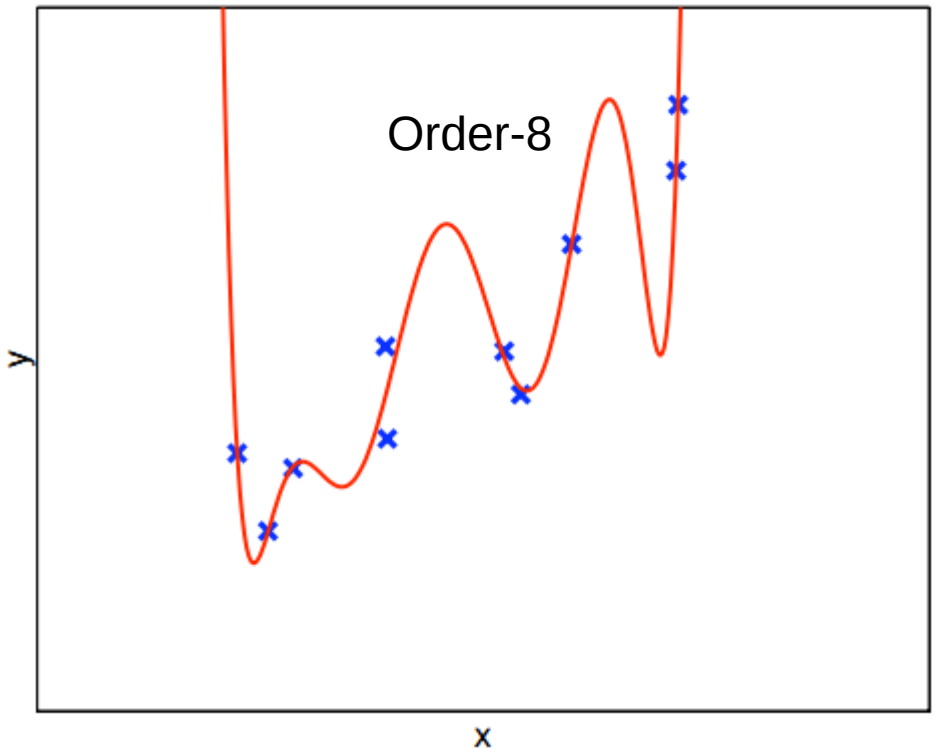
- Which one is the best fit to the data?

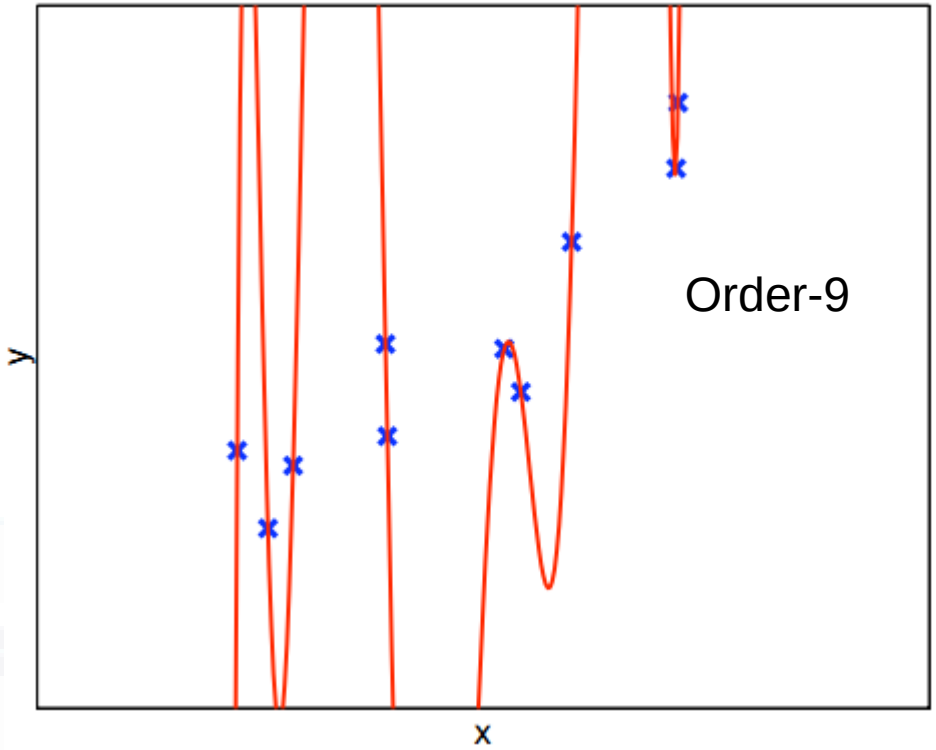




Regression using polynomial curve

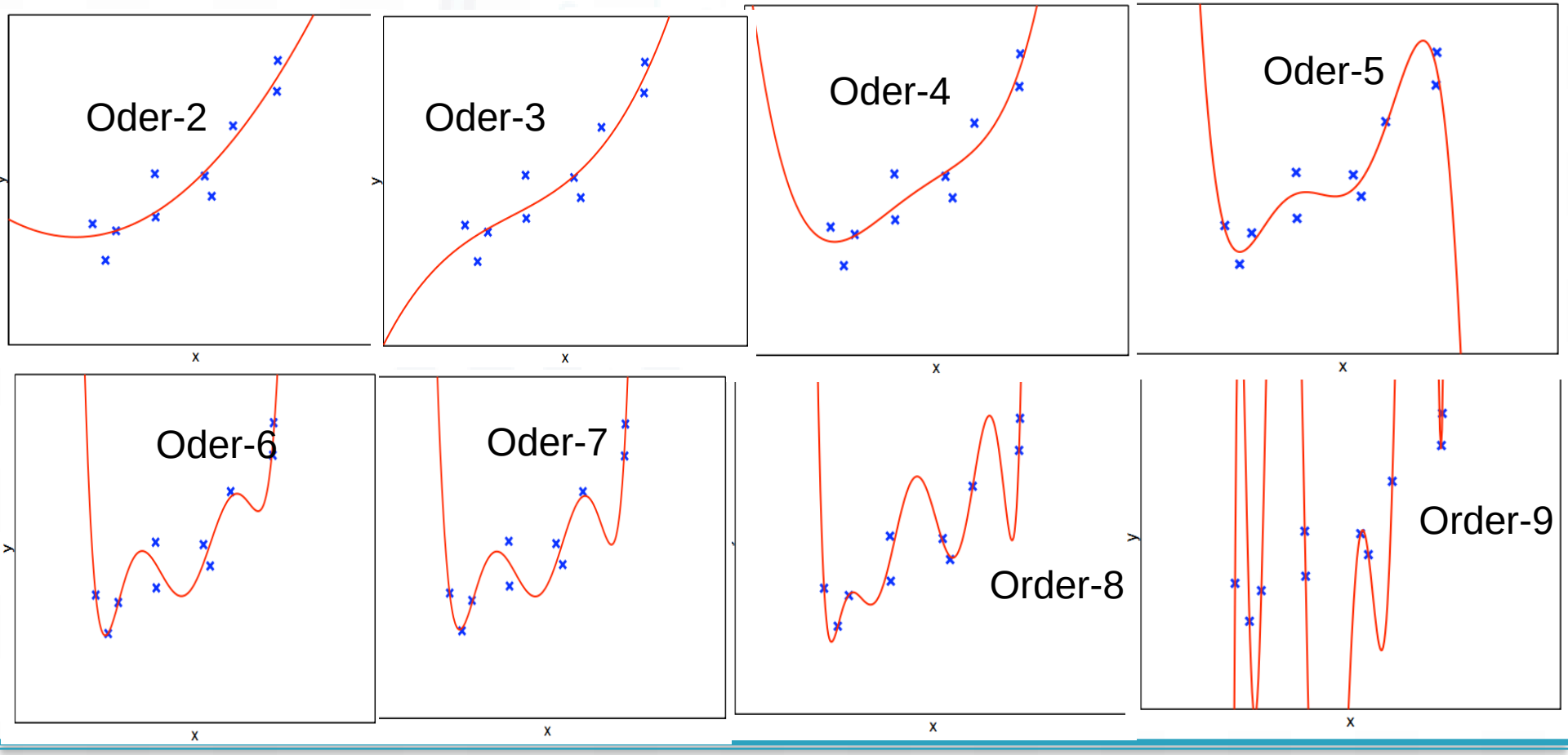
- Which one is the best fit to the data?





Regression using polynomial curve

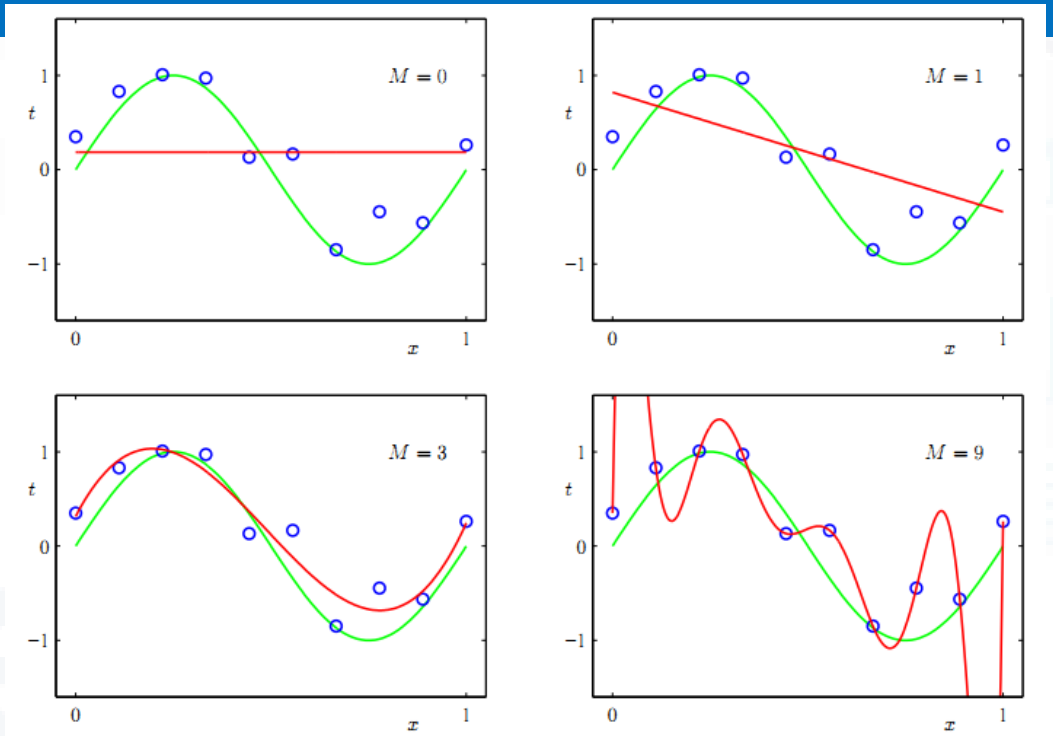
- Which one is the best fit to the data?



Overfitting

- A general, HUGELY IMPORTANT problem for all machine learning algorithms
- We can find a hypothesis (or representation R) that predicts perfectly the training data but does not generalize well to new data
- E.g., a lookup table!

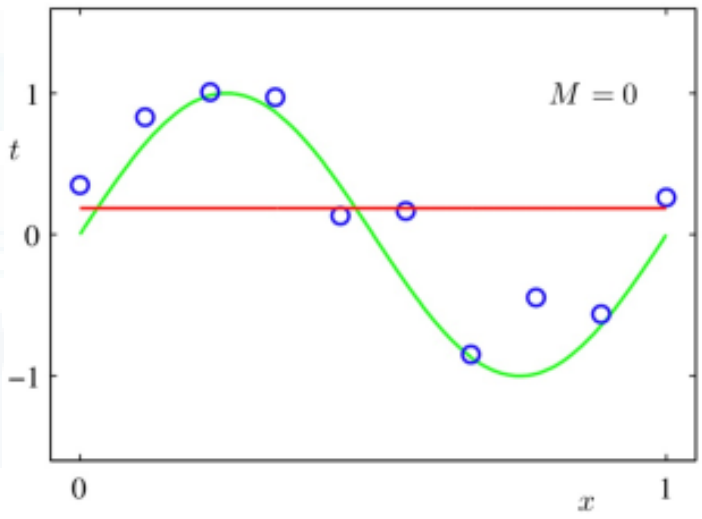
Another overfitting example



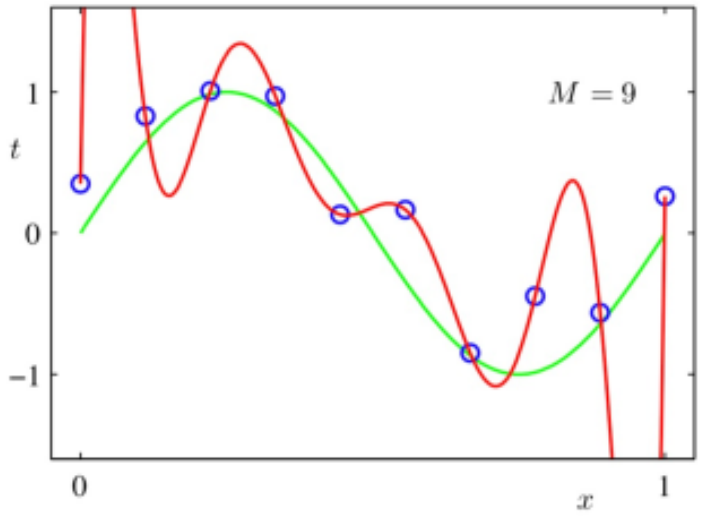
- The higher the degree of the polynomial M, the more degrees of freedom, and the more capacity to overfitting the training data
- Typical overfitting means that error on the training data is very low, but error on new instances is high

Overfitting vs Underfitting

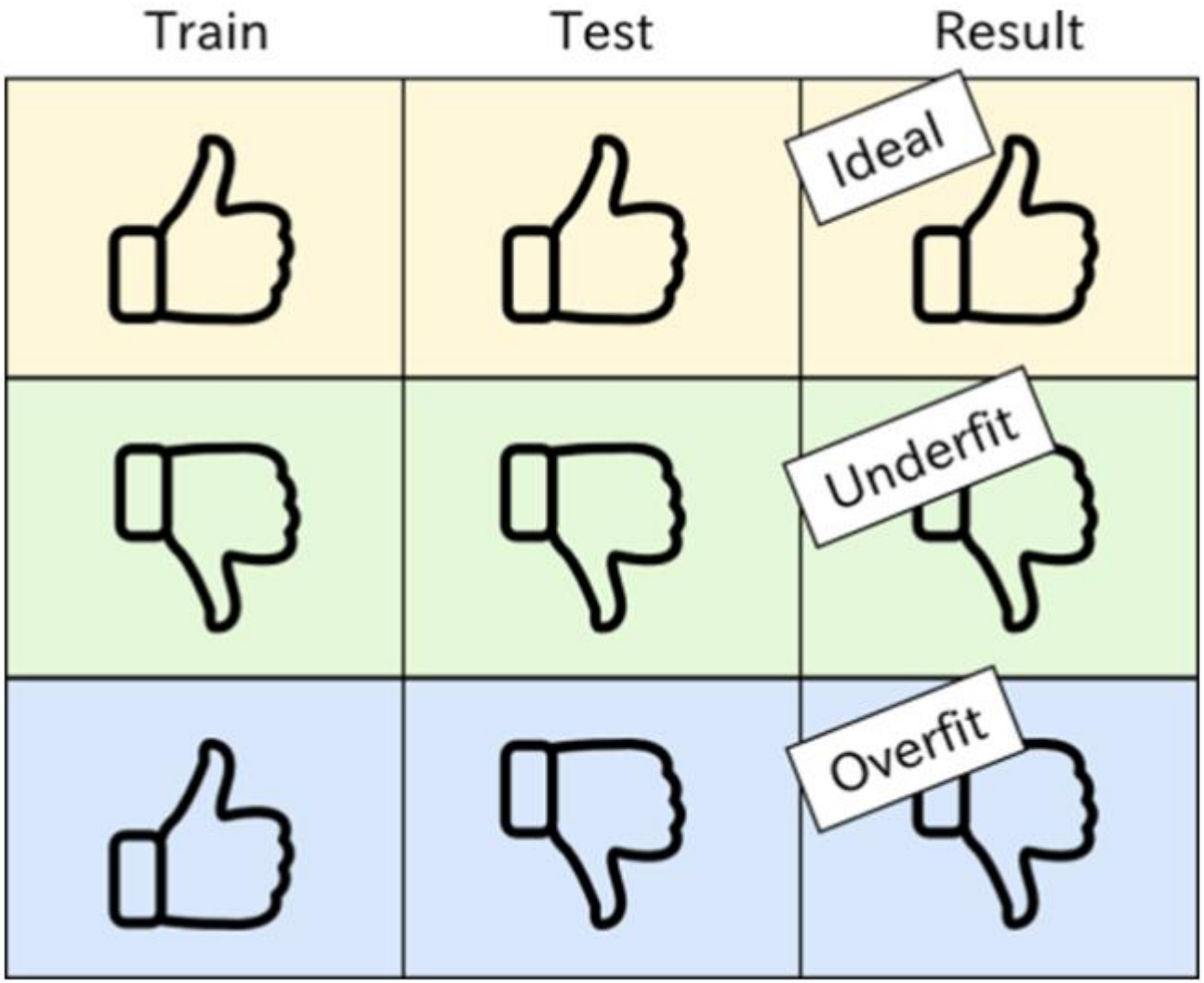
- Underfitting:
  - The model is too simple – does not fit the data.



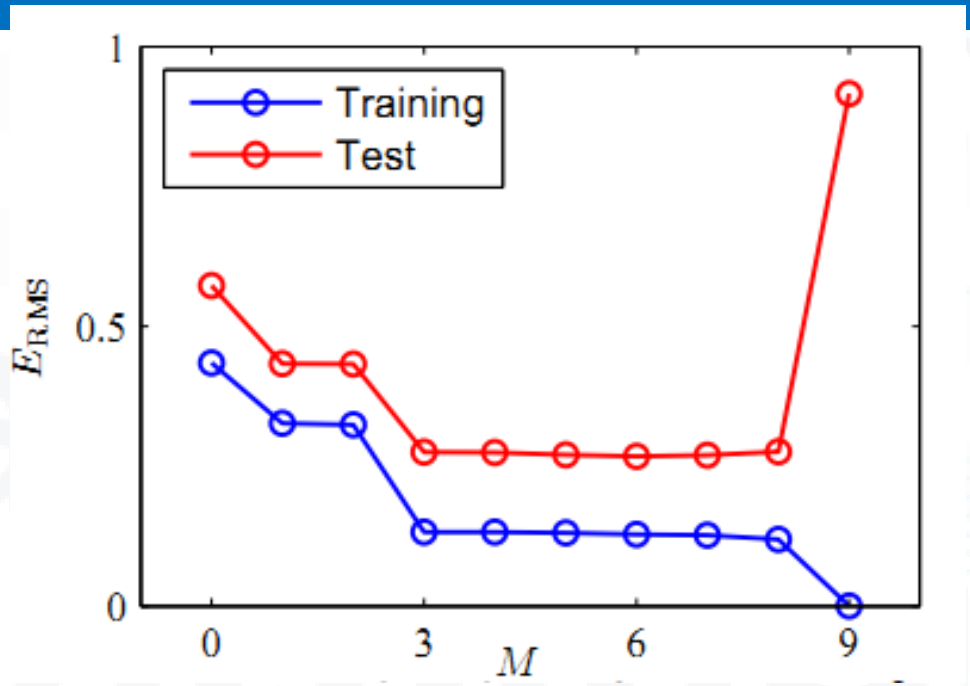
- Overfitting:
  - The model is too complex – fits perfectly, does not generalize.



Underfitting vs Overfitting (cont.)



Typical overfitting plot



- The training error decreases with the degree of the polynomial M, i.e. the complexity of the hypothesis
- The testing error, measured on independent data, decreases at first, then starts increasing

Typical overfitting plot (cont.)

- Cross-validation helps us:
  - Find a good hypothesis class (M in our case), using a validation set of data
  - Report unbiased results, using a test set, untouched during either parameter training or validation

Cross-validation

- A general procedure for estimating the true error of a predictor
- The available data is split into two subsets:
  - A training and validation set used only to find the right predictor
  - A test set used to report the prediction error of the algorithm
- These sets must be disjoint!
- The process is repeated several times, and the results are averaged to provide error estimates.

Model selection with leave-one-out cross-validation

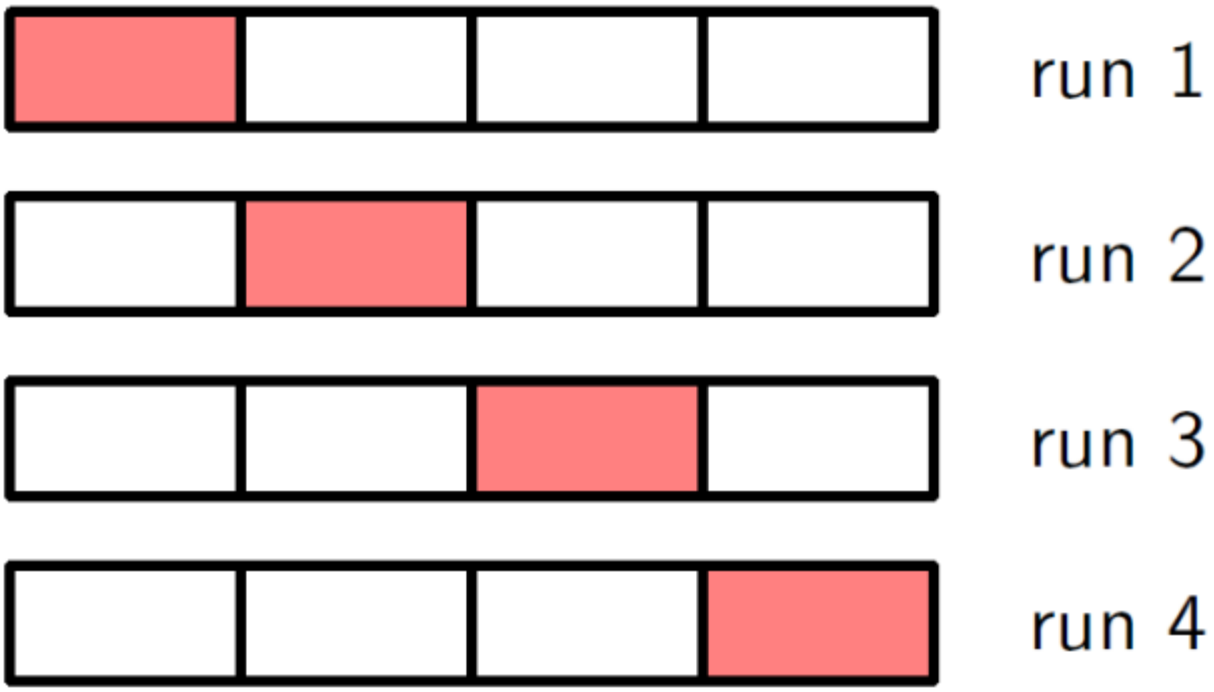
- 1. For each order of polynomial, d:
  - (a) Repeat the following procedure n times:
    - i) Leave out i^{th} instance from the training set, to estimate the true prediction error; we will put it in a validation set
    - ii) Use all the other instances to find best parameter vector, $w_{d;i}$
    - iii) Measure the error in predicting the label on the instance left out, for the $w_{d;i}$ parameter vector; call this $L_{d;i}$
    - iv) This is a (mostly) unbiased estimate of the true prediction error
  - (b) Compute the average of the estimated errors: $L_d = \frac{1}{n}\sum_{i=1}^{n} L_{d,i}$
- 2. Choose the d with lowest average estimated error: $d^* = \text{argmin}_d\ L(d)$

Summary of leave-one-out cross-validation

- A very easy to implement algorithm
- Provides a great estimate of the true error of a predictor
- Computational cost scales with the number of instances (examples), so it can be prohibitive, especially if finding the best predictor is expensive
- Alternatives:
  - Leave-k-out generalizes LOO, computationally very expensive.
  - k-fold cross-validation: split the data set into k parts, then proceed as above.

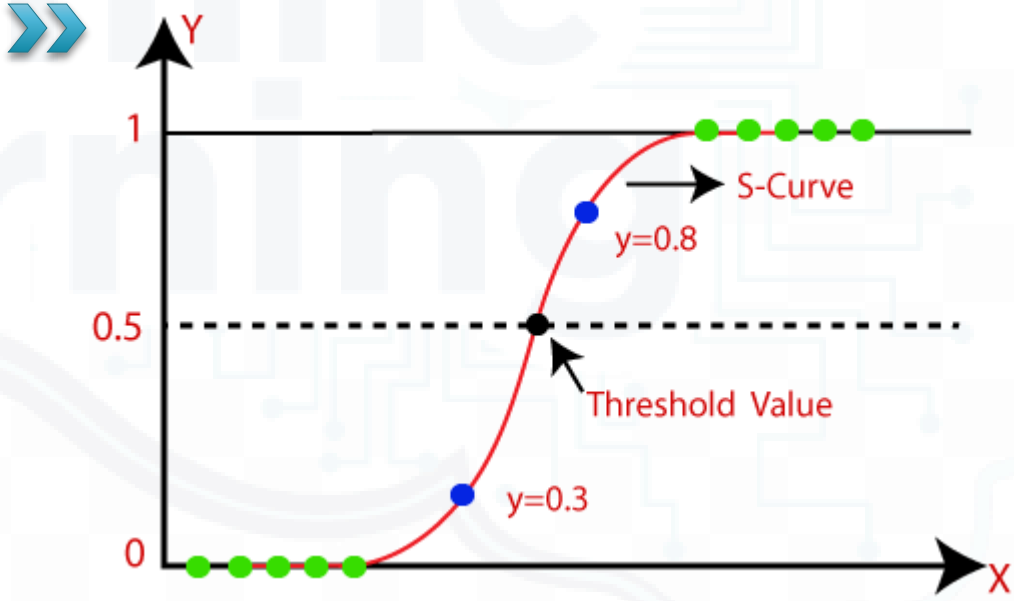
Model selection: k-fold cross validation

- Partition the training data into several groups
- Each time use one group as validation set



4 groups

Logistic regression



Notes on probability

- Let **x** be an example (having set of features); **y** be the binary (categorical) target ouput, $y \epsilon \{0, 1\}$
- $p(y_0)$: the prior probability of y_0
  - E.g., the probability that any given customer will buy a computer regardless of age, income, …
- $p(x|y_i)$: the *posterior probability* of x conditioned on y_i
  - E.g., Given that *x* will buy computer (y_i=*yes*), the prob. that x is 31..40, medium income
- p(x): the prior probability of x
- $p(y_i)$, $p(x|y_i)$, p(x): estimated from the given data

id	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rainy	mild	high	weak	yes
5	rainy	cool	normal	weak	yes
6	rainy	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rainy	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rainy	mild	high	strong	no

- y_0: play=yes; x: outlook=sunny and wind=strong
- $p(y_0)$: the probability that a person plays tennis regardless of outlook and wind, …
- $p(x)$: the probability that outlook=sunny and wind=strong.
- $p(x|y_0)$: the probability that outlook=sunny and wind=strong, if we know a person plays tennis
- $p(y_0|x)$: the probability that a person plays tennis if outlook=sunny and wind=strong.

y_0: play=yes;

x: outlook=sunny

and wind=strong

id	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rainy	mild	high	weak	yes
5	rainy	cool	normal	weak	yes
6	rainy	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rainy	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rainy	mild	high	strong	no

- $p(y_0)$=?
- $p(x)$=?
- $p(x|y_0)$=?
- $p(y_0|x)$=?

y_0: play=yes;

x: outlook=sunny and wind=strong

id	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rainy	mild	high	weak	yes
5	rainy	cool	normal	weak	yes
6	rainy	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rainy	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rainy	mild	high	strong	no

- $p(y_0)$=9/14 (9 instances having play=yes)
- p(x)=2/14 = 1/7 (2 instances having outlook=sunny and wind=strong)
- $p(x|y_0)$=1/9 (1 instance having outlook=sunny and wind=strong, if play=yes.

y_0: play=yes;

x: outlook=sunny

and wind=strong

id	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rainy	mild	high	weak	yes
5	rainy	cool	normal	weak	yes
6	rainy	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rainy	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rainy	mild	high	strong	no

- $p(y_0|x) = \frac{p(x|y_0)\,p(y_0)}{p(x)} = \frac{1/9 \times 9/15}{1/7}$

Notes on probability (cont.)

- The probability that the hypothesis holds given the observed data sample x

$$p(y_0 \mid \mathrm{x}) = \frac{p(\mathrm{x} \mid y_0)\, p(y_0)}{p(\mathrm{x})}$$ **(Bayes' theorem)**

- In addition, $$p(\mathrm{x}) = \sum_{i=\{0,1\}} p(\mathrm{x} \mid y_i)\, p(y_i)$$

- Then, $$p(y_0 \mid \mathrm{x}) = \frac{p(\mathrm{x} \mid y_0)\, p(y_0)}{p(\mathrm{x} \mid y_0)\, p(y_0) + p(\mathrm{x} \mid y_1)\, p(y_1)}$$

Notes on probability (cont.)

- (from previous slide)

$$p(y_0 \mid \mathrm{x}) = \frac{p(\mathrm{x} \mid y_0)\, p(y_0)}{p(\mathrm{x} \mid y_0)\, p(y_0) + p(\mathrm{x} \mid y_1)\, p(y_1)}$$

- It is equivalent to:

$$p(y_0 \mid \mathrm{x}) = \frac{1}{1 + \dfrac{p(\mathrm{x} \mid y_1)\, p(y_1)}{p(\mathrm{x} \mid y_0)\, p(y_0)}}$$

- Let

$$a = \ln \frac{p(\mathrm{x} \mid y_0)\, p(y_0)}{p(\mathrm{x} \mid y_1)\, p(y_1)}$$

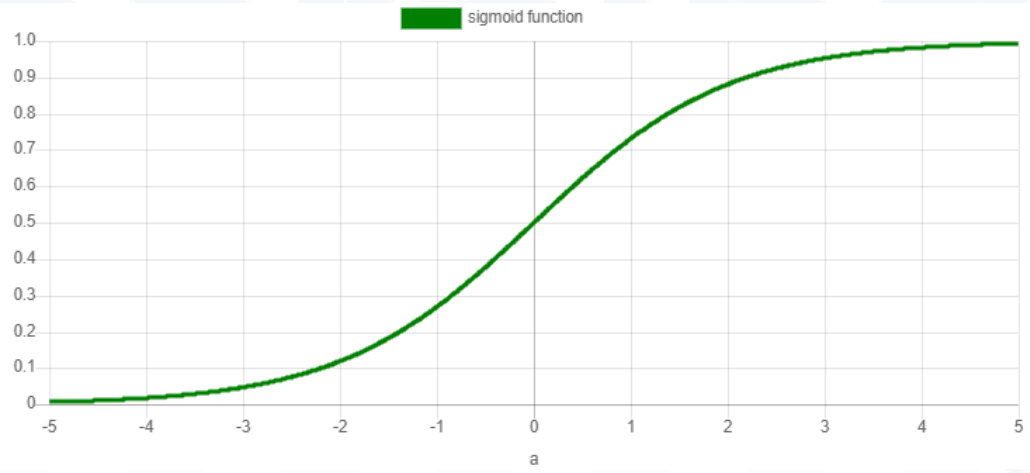
- Then, we have:

$$p(y_0 \mid \mathrm{x}) = \frac{1}{1 + e^{-a}} = \sigma(a)$$

Notes on probability (cont.)

- σ (a): sigmoid function, σ (a) ϵ [0,1]

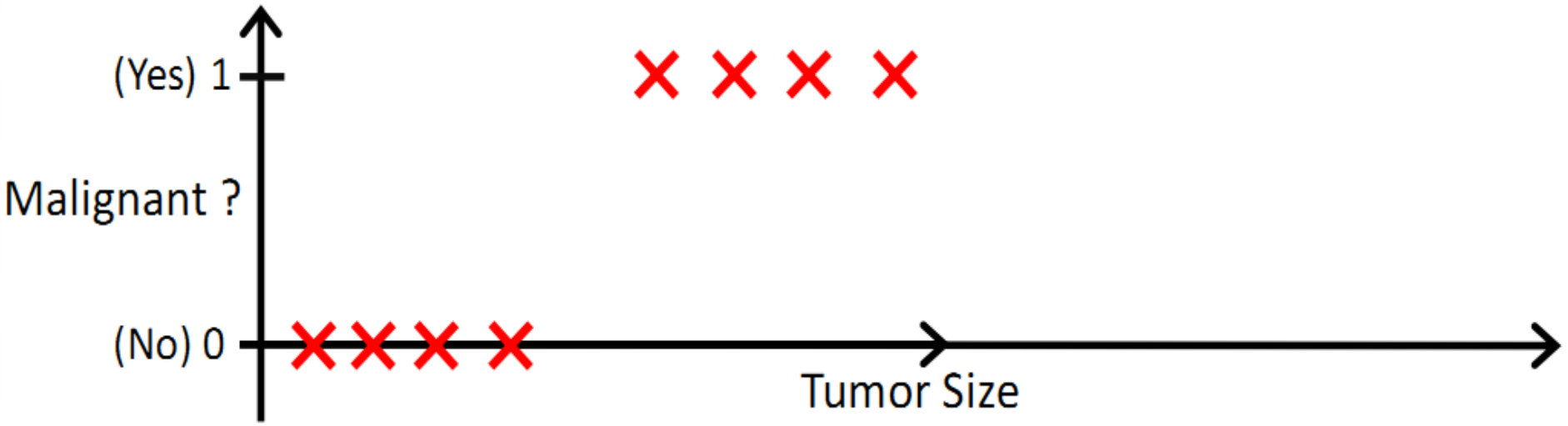
$$p(y_0 \mid \mathrm{x}) = \frac{1}{1+e^{-a}} = \sigma(a)$$



Problems for Today

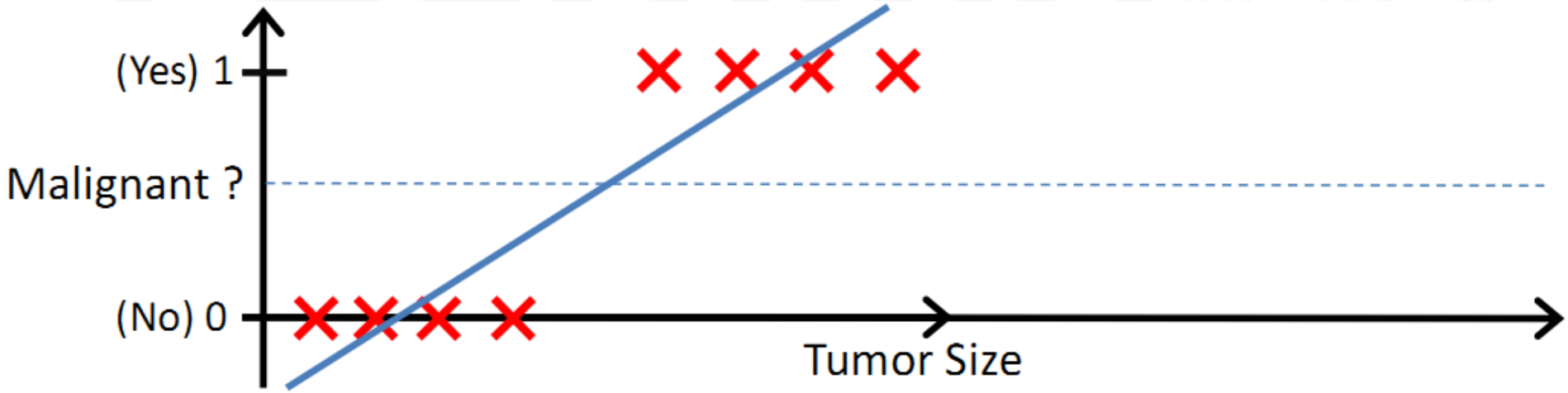
- Can we solve the problem using linear regression?

Tumor prediction



Why not use Linear Regression?

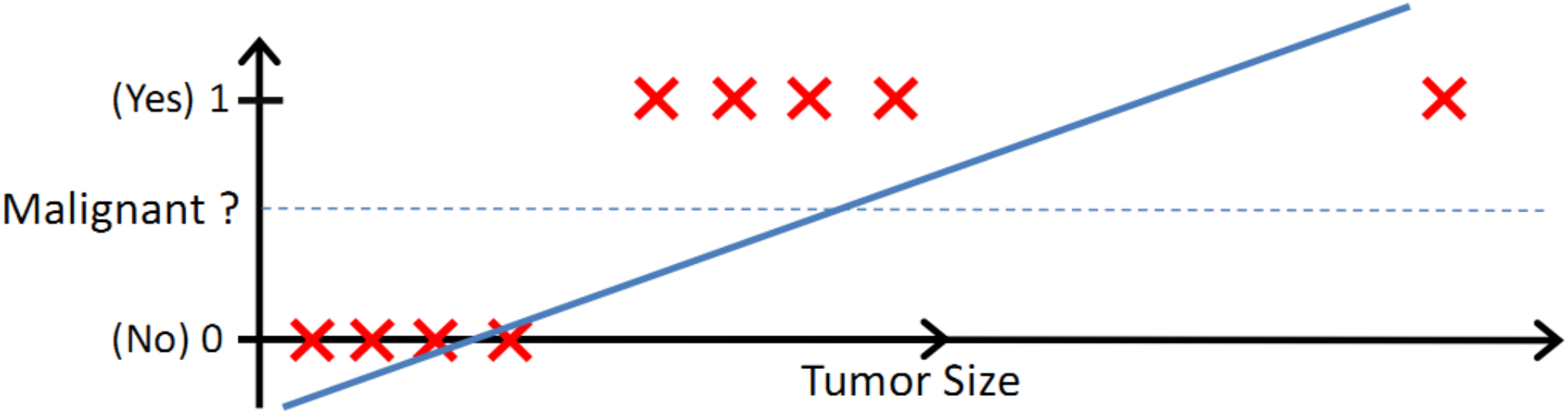
- Can we solve the problem using linear regression?
  - E.g., fit a straight line and define a threshold at 0.5
  - Threshold classifier output f(x) at 0.5:
    - If f(x) ≥ 0.5, predict "y=1"
    - If f(x)< 0.5, predict "y=0"



Why not use Linear Regression?

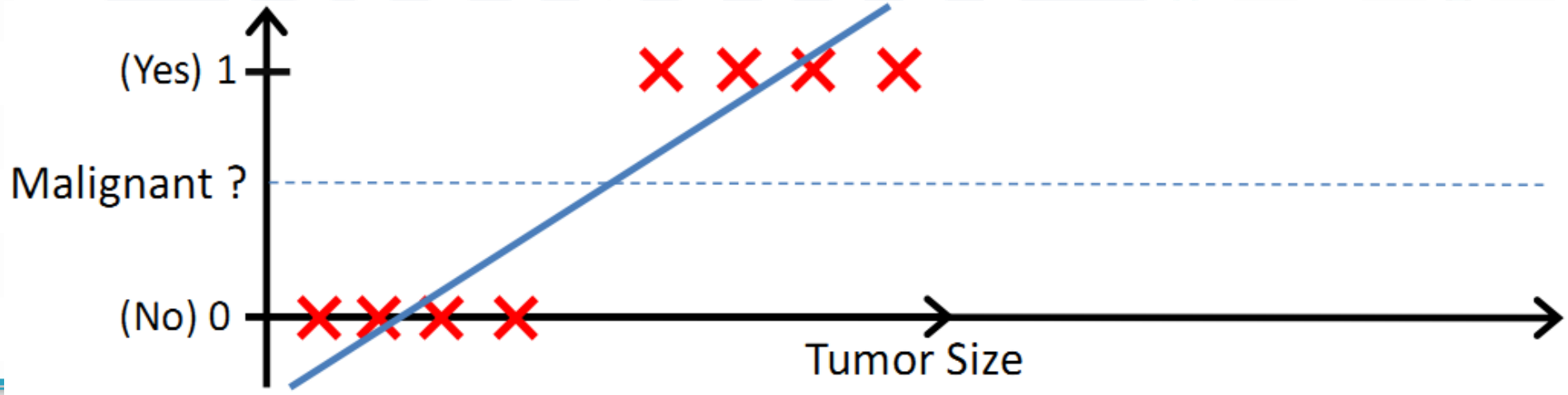
- Can we solve the problem using linear regression?
  - E.g., fit a straight line and define a threshold at 0.5
  - Threshold classifier output f(x) at 0.5:
    - If f(x)>= 0.5, predict "y=1"
    - If f(x)< 0.5, predict "y=0"

Failure due to adding a new point



Why not use Linear Regression?

- Another drawback of using linear regression for this problem
  - Classification: y=0 or y=1
  - Here, f(x) can be > 1or < 0
- What we need:
  - Logistic Regression: $0 \leq f(x) \leq 1$



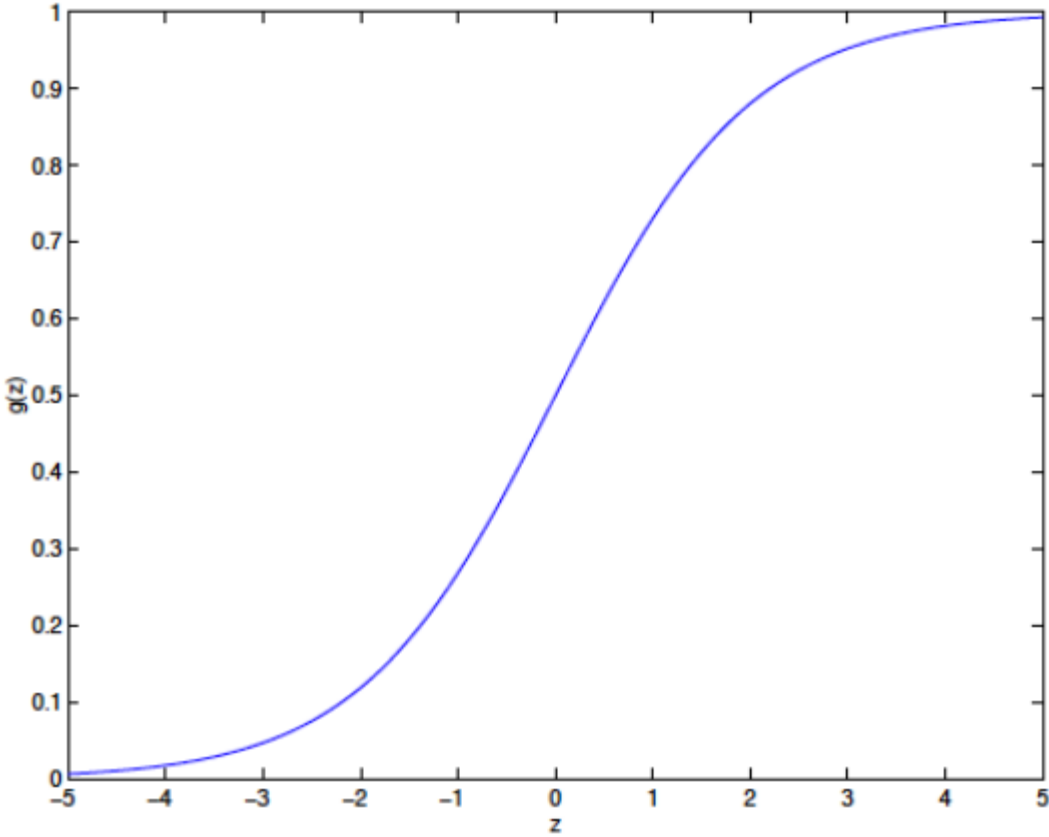
What is logistic regression?

- LR: used to solve the classification problems, called as Classification Algorithm that models the probability of output class.
  - The output required is represented in discrete values like binary (0, 1), categorical, nominal.
  - It estimates relationship between a dependent variable (output) and one or more independent variable (predictors).

Logistic Regression Model

- Logistic regression model:
  - $0 \leq f(x) \leq 1$
  - $f(x) = g(w^T x)$
- Sigmoid function:

$$g(w^T x) = \frac{1}{1 + e^{-w^T x}}$$



A useful property: easy to compute differential at any point

Interpretation of Hypothesis Output

- Estimated probability that $y = 1$ on input x

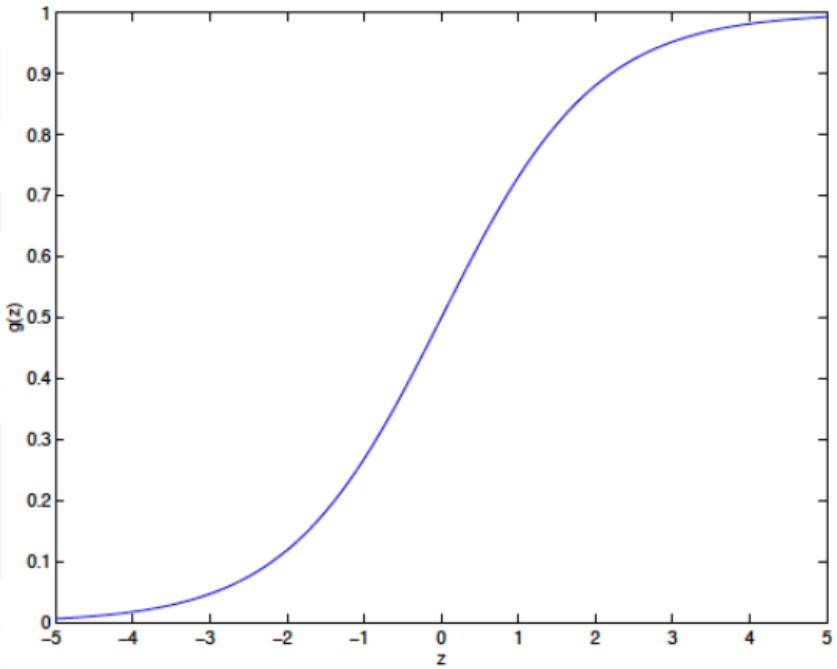
"probability that $y = 1$, given x, parameterized by w"

- Example: If $f(x) = 0.7$, tell patient that 70% chance of tumor being malignant

Logistic regression

- Logistic regression model:
  - $0 \leq f(x) \leq 1$
  - $f(x) = g(w^T x)$
- Sigmoid function:

$$g(w^T x) = \frac{1}{1 + e^{-w^T x}}$$



- Suppose predict:
  - y=1 if $f(x) \geq 0.5$ when $w^T x \geq 0$
  - y=0 if $f(x) < 0.5$ when $w^T x < 0$

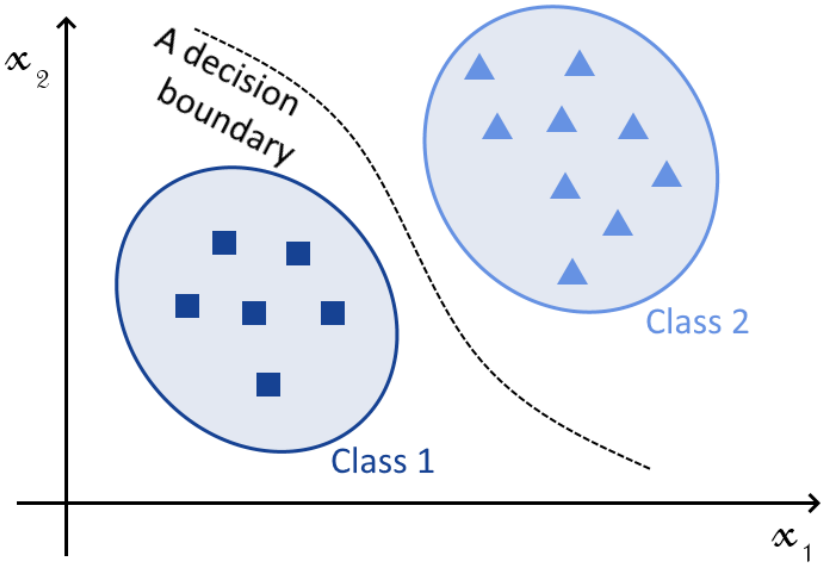
Types of Logistic Regression

- Binary logistic regression
  - The dependent variable is dichotomous.

- Multinomial logistic regression
  - The dependent/outcomes variable has more than two categories.

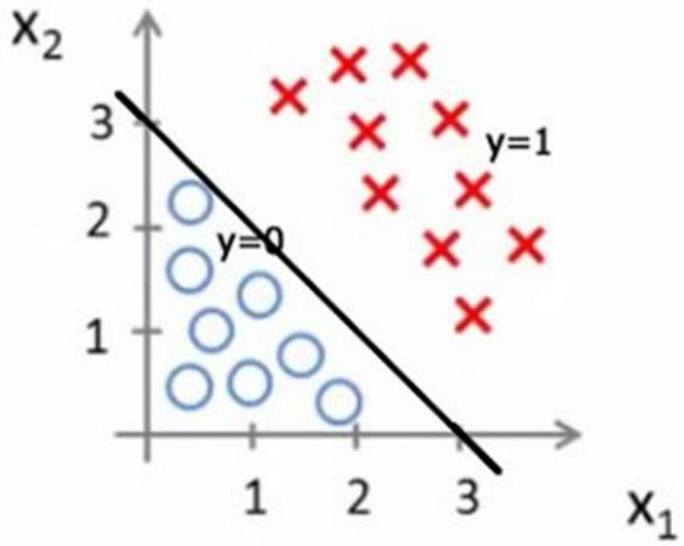
Separating two classes of points

- We are attempting to separate two given sets/ classes of points
- Separate two regions of the feature space
- Concept of Decision Boundary
- Finding a good decision boundary => learn appropriate values for the parameters w



What is Decision Boundary?

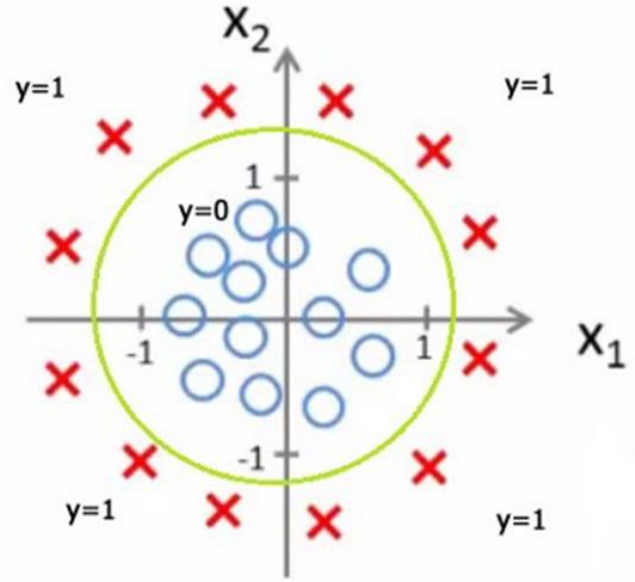
- Decision boundary helps to differentiate probabilities into positive class and negative class.



For y = 1, Equation of line would be $x_1 + x_2 >= 3$

For y = 0, Equation of line would be $x_1 + x_2 < 3$

Linear Decision Boundary



For y=1, equation would be $x_1^2+x_2^2 >= 1$

For y=0, equation would be $x_1^2+x_2^2 < 1$

Non Linear Decision Boundary

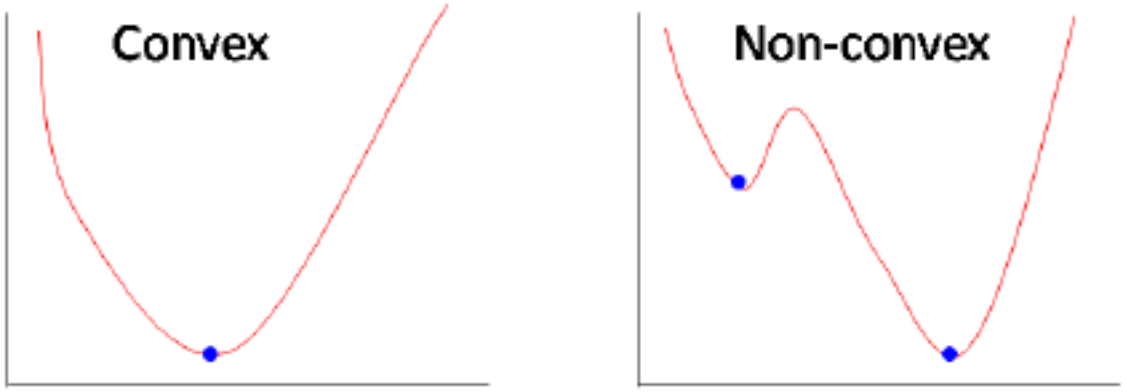
Loss function for Logistic Regression

Convex vs Non-Convex function

- Loss function of Linear Regression is a convex function

$$L(\mathrm{w}) = L(w_0, w_1, w_2, \ldots w_m) = \frac{1}{2n}\sum_{i=1}^{n}\left(f\left(\mathrm{x}^{(i)}\right) - y^{(i)}\right)^2$$

- Apply L(w) to Logistic regression ➔ Non-convex function



How to build loss function?

- Let $p(y^{(i)}=1|x^{(i)};w)$ be the probability that the binary output y is 1 given the input feature vector $x^{(i)}$

$$p\left(y^{(i)}=1|\mathrm{x}^{(i)};\mathrm{w}\right)=\frac{1}{1+e^{-\mathrm{w}^T\mathrm{x}^{(i)}}}=f\left(\mathrm{w}^{\mathrm{T}}\mathrm{x}^{(i)}\right)$$

- Similarly, the probability $p(y^{(i)}=0|x^{(i)};w)$:

$$p\left(y^{(i)}=0|\mathrm{x}^{(i)};\mathrm{w}\right)=1-\frac{1}{1+e^{-\mathrm{w}^T\mathrm{x}^{(i)}}}=1-f\left(\mathrm{w}^{\mathrm{T}}\mathrm{x}^{(i)}\right)$$

How to build loss function?

- Let $z^{(i)}=f(w^T x^{(i)})$
- We have the following equation for both cases ($y^{(i)}=0$, $y^{(i)}=1$)

$$p\left(y^{(i)} \mid \mathrm{x}^{(i)}; \mathrm{w}\right)=\left(z^{(i)}\right)^{y^{(i)}}\left(1-z^{(i)}\right)^{1-y^{(i)}}$$

- The objective: $y^{(i)} \approx f(w^T x^{(i)})$

➔ Need to optimize

$$\mathrm{w}=\arg\max_{\mathrm{w}} p\left(\mathrm{y} \mid \mathrm{X}; \mathrm{w}\right)$$

$X=(x^{(1)}, x^{(2)}, \ldots, x^{(n)})$, $y=(y^{(1)}, y^{(2)}, \ldots, y^{(n)})$

How to build loss function?

- It is an maximum likelihood estimation with w

$$p(\mathrm{y} \mid \mathrm{X}; \mathrm{w}) = \prod_{i=1}^{n} p\left(y^{(i)} \mid \mathrm{x}^{(i)}; \mathrm{w}\right) = \prod_{i=1}^{n} \left(z^{(i)}\right)^{y^{(i)}} \left(1 - z^{(i)}\right)^{1-y^{(i)}}$$

- Apply logarit for both sides (multiply –1):

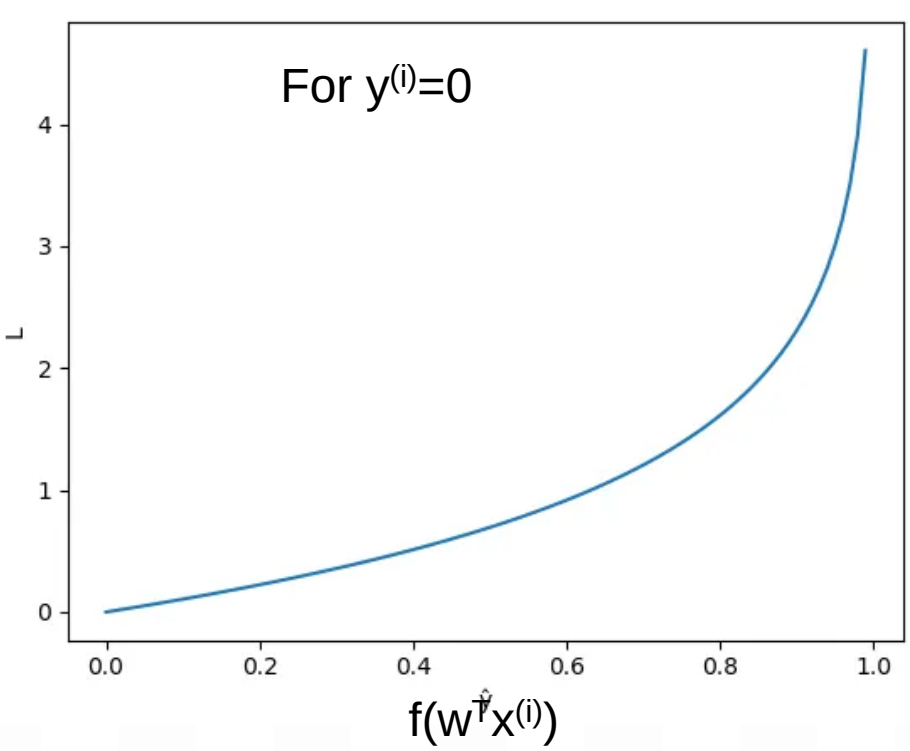
$$L(\mathrm{w}) = -\frac{1}{n} \log p(\mathrm{y} \mid \mathrm{X}; \mathrm{w})$$

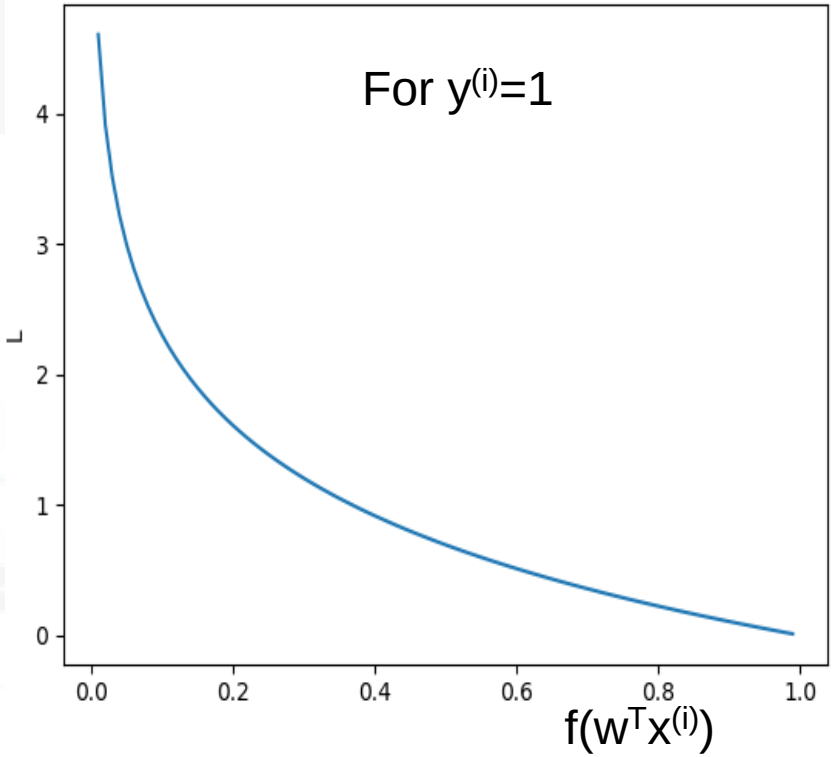
Need to miminize L(w)

$$= -\frac{1}{n} \sum_{i=1}^{n} y^{(i)} \log z^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - z^{(i)}\right)$$

Loss function

$$L(\mathrm{w}) = -\frac{1}{n}\sum_{i=1}^{n} y^{(i)} \log z^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - z^{(i)}\right)$$





Notes on Derivative

$(k.x)' = k$	$(k.u)' = k.u'$
$(x^n)' = n.x^{n-1}$	$(u^n)' = n.u^{n-1}.(u)'$
$(\frac{1}{x})' = -\frac{1}{x^2}$	$(\frac{1}{u})' = -\frac{(u)'}{u^2}$
$(\sqrt{x})' = \frac{1}{2\sqrt{x}}$	$(\sqrt{u})' = \frac{u'}{2\sqrt{u}}$
$(\sin x)' = \cos x$	$(\sin u)' = \cos u.(u)'$
$(\cos x)' = -\sin x$	$(\cos u)' = -\sin u.(u)'$
$(\tan x)' = 1 + \tan^2 x = \frac{1}{\cos^2 x}$	$(\tan u)' = (1 + \tan^2 u).u' = \frac{u'}{\cos^2 u}$
$(\cot x)' = -(1 + \cot^2 x) = -\frac{1}{\sin^2 x}$	$(\cot u)' = -(1 + \cot^2 u).u' = -\frac{u'}{\sin^2 u}$
$(e^x)' = e^x$	$(e^u)' = e^u.u'$
$(a^x)' = a^x.\ln a$	$(a^u)' = a^u.\ln a.u'$
$(\ln x)' = \frac{1}{x}$	$(\ln u)' = \frac{u'}{u}$
$(\log_a x)' = \frac{1}{x.\ln a}$	$(\log_a u)' = \frac{u'}{u.\ln a}$

Differentiation Rules	
Constant Rule	$\frac{d}{dx}[c] = 0$
Power Rule	$\frac{d}{dx} x^n = nx^{n-1}$
Product Rule	$\frac{d}{dx}[f(x)g(x)] = f'(x)g(x) + f(x)g'(x)$
Quotient Rule	$\frac{d}{dx}\left[\frac{f(x)}{g(x)}\right] = \frac{g(x)f'(x) - f(x)g'(x)}{[g(x)]^2}$
Chain Rule	$\frac{d}{dx}[f(g(x))] = f'(g(x))g'(x)$

Chain Rule

Suppose that we have two functions $f(x)$ and $g(x)$ and they are both differentiable.

1. If we define $F(x) = (f \circ g)(x)$ then the derivative of $F(x)$ is,

$$F'(x) = f'(g(x)) \quad g'(x)$$

2. If we have $y = f(u)$ and $u = g(x)$ then the derivative of y is,

$$\frac{dy}{dx} = \frac{dy}{du} \quad \frac{du}{dx}$$

Chain Rule (cont.)

- Example:

Suppose we want to differentiate $y = \cos x^2$.

Let $u = x^2$ so that $y = \cos u$.

It follows immediately that

$$\frac{\mathrm{d}u}{\mathrm{d}x} = 2x \qquad \frac{\mathrm{d}y}{\mathrm{d}u} = -\sin u$$

The chain rule says

$$\frac{\mathrm{d}y}{\mathrm{d}x} = \frac{\mathrm{d}y}{\mathrm{d}u} \times \frac{\mathrm{d}u}{\mathrm{d}x}$$

and so

$$\begin{aligned} \frac{\mathrm{d}y}{\mathrm{d}x} &= -\sin u \times 2x \\ &= -2x \sin x^2 \end{aligned}$$

Gradient descent

- Loss function:

$$L(\mathrm{w}) = -\frac{1}{n}\sum_{i=1}^{n} y^{(i)} \log z^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - z^{(i)}\right)$$

- Derivative of loss function:

$$\frac{\partial L(\mathrm{w})}{\partial w_j} = \frac{\partial\left(-\frac{1}{n}\sum_{i=1}^{n} y^{(i)} \log z^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - z^{(i)}\right)\right)}{\partial w_j}$$

Gradient descent (cont.)

- Derivative of sum of terms:

$$\frac{\partial\left(y^{(i)}\log z^{(i)}\right)+\partial\left(\left(1-y^{(i)}\right)\log\left(1-z^{(i)}\right)\right)}{\partial w_j}$$

- Derivative of log f (x)

$$\left(\frac{y^{(i)}}{z^{(i)}}-\frac{1-y^{(i)}}{1-z^{(i)}}\right)\frac{\partial z^{(i)}}{\partial w_j}$$

$$\frac{\partial\left(y^{(i)}\log z^{(i)}\right)+\partial\left(\left(1-y^{(i)}\right)\log\left(1-z^{(i)}\right)\right)}{\partial w_j}$$

$$\frac{\partial\left(y^{(i)}\log z^{(i)}\right)}{\partial w_j}+\frac{\partial\left(\left(1-y^{(i)}\right)\log\left(1-z^{(i)}\right)\right)}{\partial w_j}$$

$$\frac{y^{(i)}\partial\left(\log z^{(i)}\right)}{\partial w_j}+\frac{\left(1-y^{(i)}\right)\partial\left(\log\left(1-z^{(i)}\right)\right)}{\partial w_j}$$

$$\frac{y^{(i)}}{z^{(i)}}\times\frac{\partial(z^{(i)})}{\partial w_j}+\frac{\left(1-y^{(i)}\right)}{1-z^{(i)}}\times\frac{\partial\left(\left(1-z^{(i)}\right)\right)}{\partial w_j}$$

$$\frac{y^{(i)}}{z^{(i)}}\times\frac{\partial(z^{(i)})}{\partial w_j}-\frac{\left(1-y^{(i)}\right)}{1-z^{(i)}}\times\frac{\partial\left(z^{(i)}\right)}{\partial w_j}=\left(\frac{y^{(i)}}{z^{(i)}}-\frac{\left(1-y^{(i)}\right)}{1-z^{(i)}}\right)\times\frac{\partial(z^{(i)})}{\partial w_j}$$

Gradient descent (cont.)

- (Previous slides): $\left(\frac{y^{(i)}}{z^{(i)}} - \frac{\left(1 - y^{(i)}\right)}{1 - z^{(i)}} \right) \times \frac{\partial (z^{(i)})}{\partial w_j}$

where $z^{(i)} = f\left(\mathrm{w}^{\mathrm{T}} \mathrm{x}^{(i)}\right) = \frac{1}{1 + e^{-\mathrm{w}^T \mathrm{x}^{(i)}}} = \sigma\left(\mathrm{w}^T \mathrm{x}^{(i)}\right)$

- How to compute derivative of $\sigma(w^T x^{(i)})$?

Gradient descent (cont.)

- How to compute the derivative of sigmoid function?

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

- Then, we have:

$$\sigma'(x) = \frac{e^{-x}}{\left(1+e^{-x}\right)^2} = \frac{1}{1+e^{-x}} \times \frac{e^{-x}}{1+e^{-x}}$$

$$= \sigma(x) \times \left(\frac{1-1+e^{-x}}{1+e^{-x}} \right) = \sigma(x) \times \left(\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}} \right)$$

$$= \sigma(x)\left(1-\sigma(x)\right)$$

Gradient descent (cont.)

- Similarly, the derivative of $z^{(i)}=\sigma(w^T x^{(i)})$: $\sigma(u)$

$$\frac{\partial(z^{(i)})}{\partial w_j} = z^{(i)}\left(1-z^{(i)}\right)\frac{\partial\left(w^T x^{(i)}\right)}{\partial w_j}$$

- Chain rule + derivative of sigma:

$$\left(\frac{y^{(i)}}{z^{(i)}} - \frac{1-y^{(i)}}{1-z^{(i)}}\right) z^{(i)}\left(1-z^{(i)}\right) x_j^{(i)}$$

Gradient descent (cont.)

- Algebraic manipulation

$$\left(\frac{y^{(i)} - z^{(i)}}{z^{(i)}\left(1 - z^{(i)}\right)} \right) z^{(i)}\left(1 - z^{(i)}\right) x_j^{(i)}$$

- Cancelling terms

$$\left(y^{(i)} - z^{(i)}\right) x_j^{(i)}$$

- In conclusion:

$$\frac{\partial L(\mathrm{w})}{\partial w_j} = \sum_{i=1}^{n} \left(y^{(i)} - f\left(\mathrm{w}^T \mathrm{x}^{(\mathrm{i})}\right) \right) x_j^{(i)}$$

Gradient descent (cont.)

- Loss function:

$$L(\mathrm{w}) = -\frac{1}{n}\sum_{i=1}^{n} y^{(i)} \log z^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - z^{(i)}\right)$$

$z^{(i)}=f(w^{T}x^{(i)})$

repeat {

$$w_j = w_j - \lambda \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - f\left(\mathrm{w}^{T}\mathrm{x}^{(\mathrm{i})}\right)\right) x_j^{(i)}$$

}

Where j=0,1, …, m
(simultaneously update all w_j)

Notes on Gradient descent

- Algorithm looks identical to linear regression, but the hypothesis function is different for logistic regression.
- Thus, we can use gradient descent to learn parameter values, and hence compute for a new input:
- To make a prediction given new x
- Output:

$$f\left(\mathrm{w}^T x\right) = \frac{1}{1+e^{-\mathrm{w}^T x}}$$

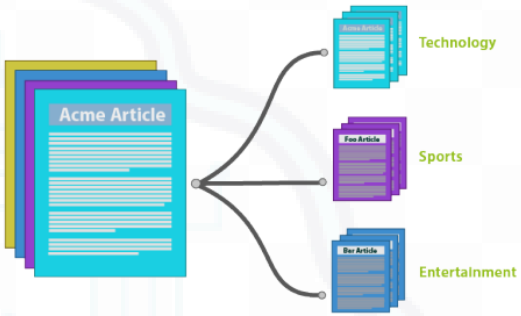
estimated probability that y = 1 on input x

Multi-class classification

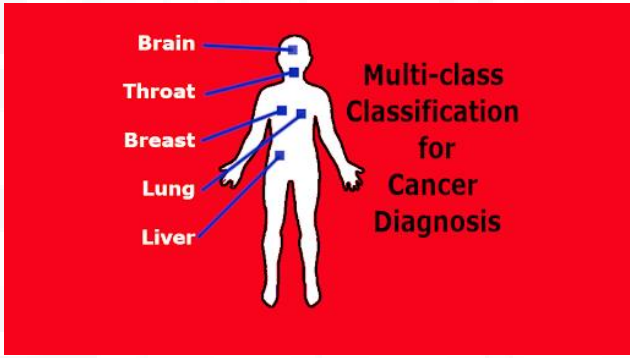
- One vs. All

Multiclass classification

- News article tagging: Politics, Sports, Movies, Religion, …



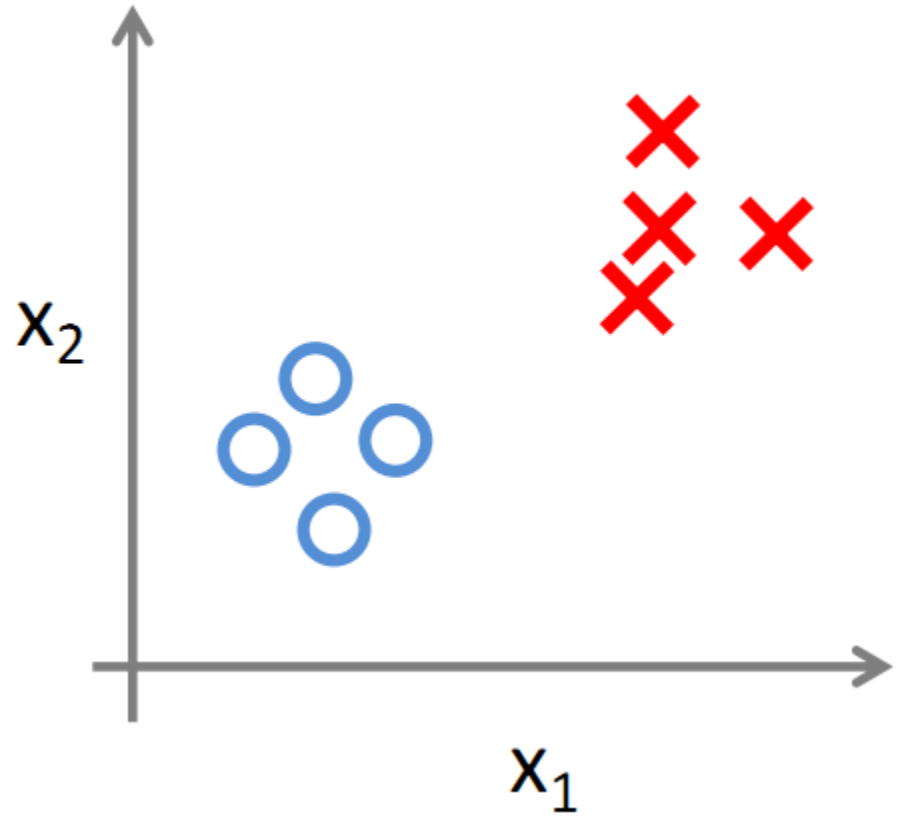
- Medical diagnosis: Not ill, Cold, Flu, Fever



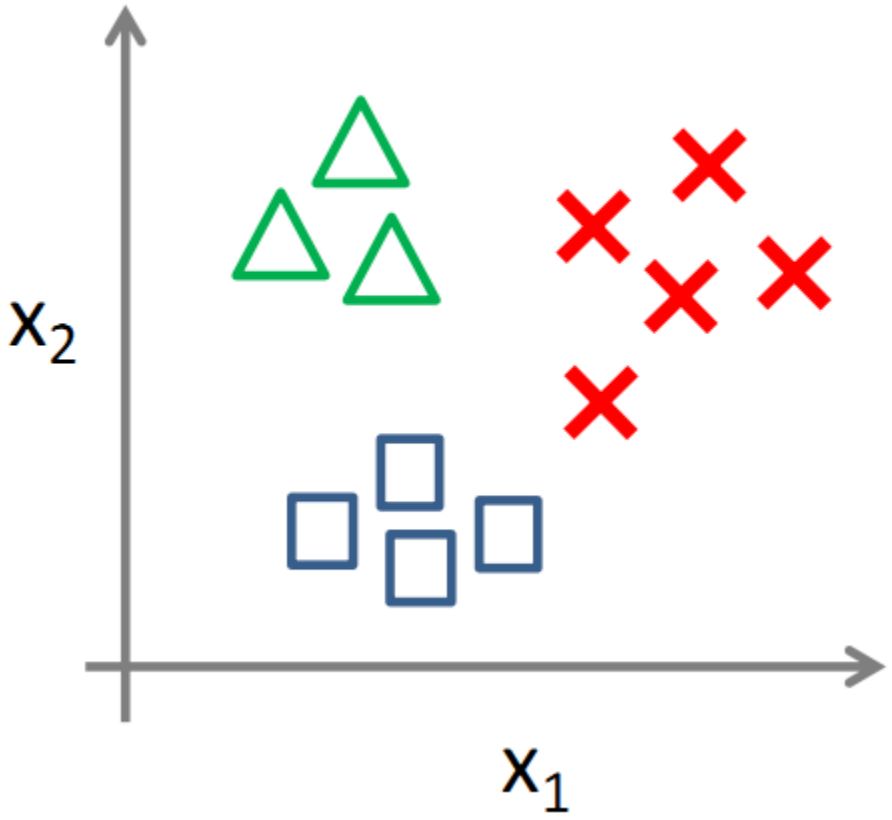
- Weather: Sunny, Cloudy, Rain, Snow

Binary vs multi-class classification

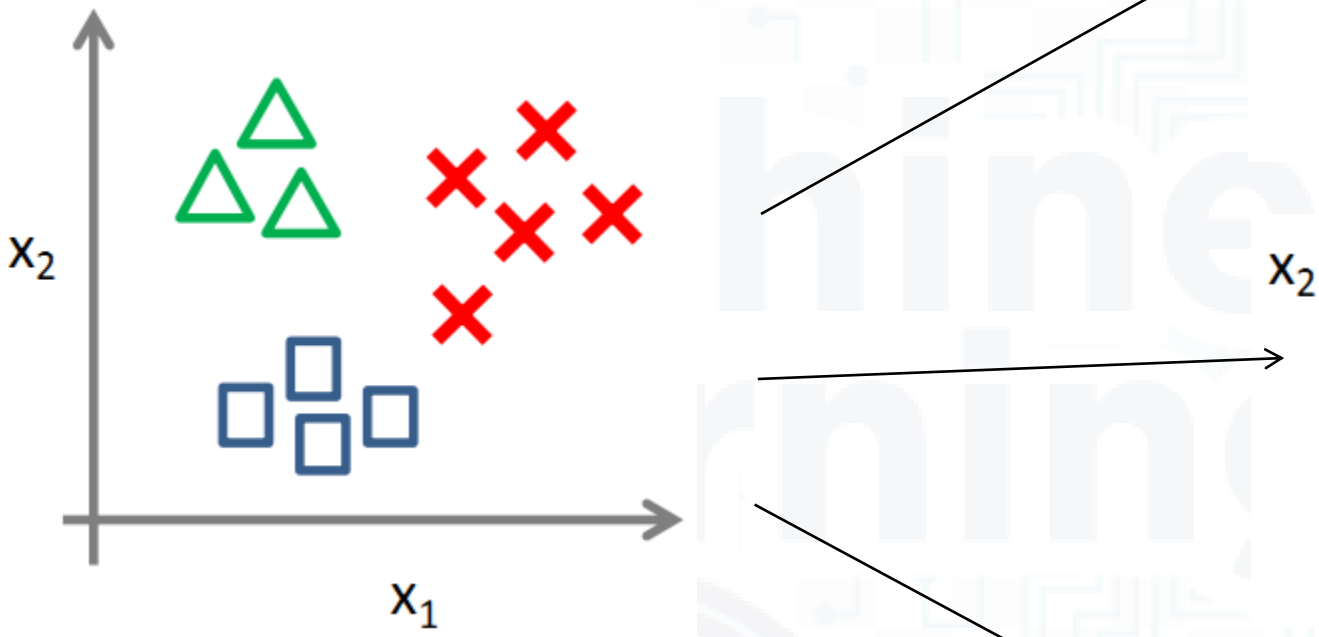
Binary classification:

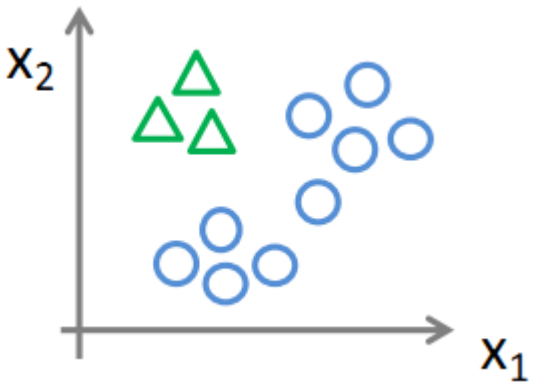


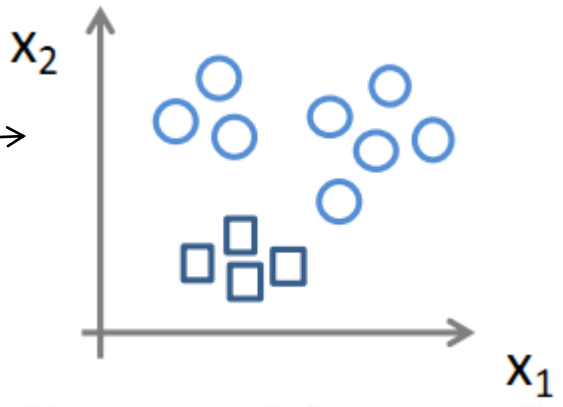
Multi-class classification:

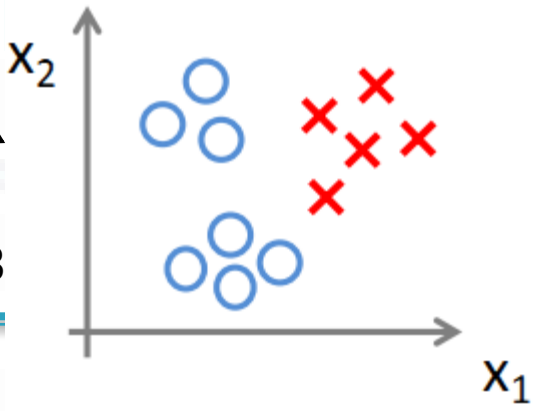


One-vs-all (one-vs-rest)









Class 1: △

Class 2: □

Class 3: ×

$$f^{(i)}\left(w^T x\right) = P\left(y = i \mid x; w\right)$$

i=1,2,3

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

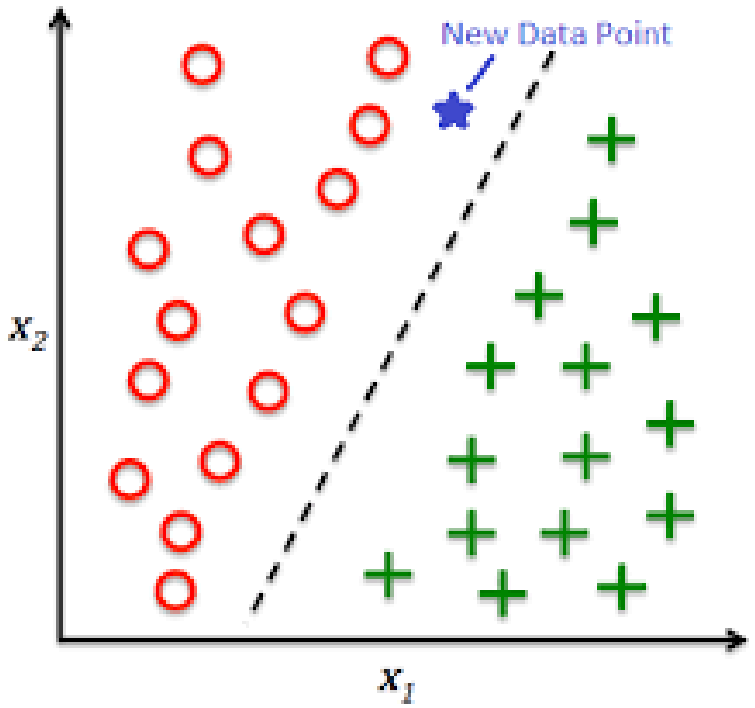
- Train a logistic regression classifier $f^{(i)}(w^T x)$ for each class i to predict the probability that y=i.

- On a new input x, to make a prediction, pick the class i that maximizes

$$\max_i f^{(i)}\left(w^T x\right)$$

FACULTY OF INFORMATION TECHNOLOGY

Classification



Classification: Definition

- Given a collection of records (training set):
  - Each record is by characterized by a tuple (x,y), where x is the attribute set and y is the class label

    - x: attribute, predictor, independent variable, input

    - y: class, response, dependent variable, output

Classification (cont.)

- Task:
  - Learn a model that maps each attribute set x into one of the predefined class labels y

Classification Task: examples

Task	Attribute set, x	Class label, y
Categorizing email messages		
Identifying tumor cells		

Classification Task: examples

Task	Attribute set, x	Class label, y
Categorizing email messages	Features extracted from email message header and content	spam or non-spam
Identifying tumor cells		

Classification Task: examples

Task	Attribute set, x	Class label, y
Categorizing email messages	Features extracted from email message header and content	spam or non-spam
Identifying tumor cells	Features extracted from MRI scans	malignant or benign cells

Classification vs Prediction

- Classification
  - predicts categorical class labels (discrete or nominal)
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data

- Numeric Prediction: models continuous-valued functions, i.e., predicts unknown or missing values

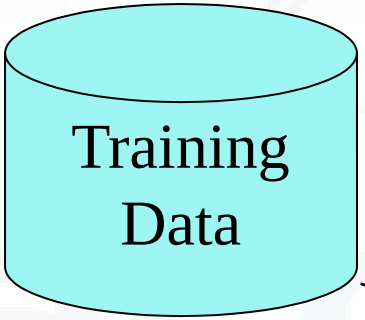
Classification—1: Model construction

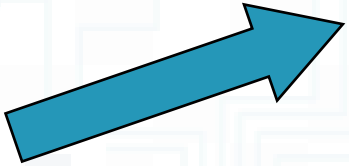
- Model construction: describing a set of pre-determined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision trees, or mathematical formulae

Classification— 2: Model usage

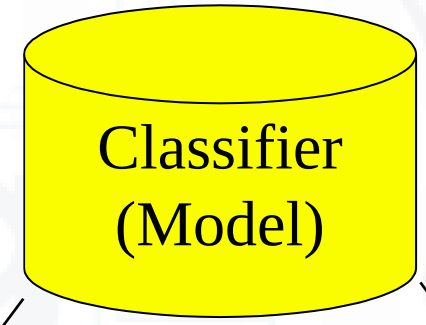
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set, otherwise overfitting will occur
  - If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known.

Example: Step 1





Classification
Algorithms

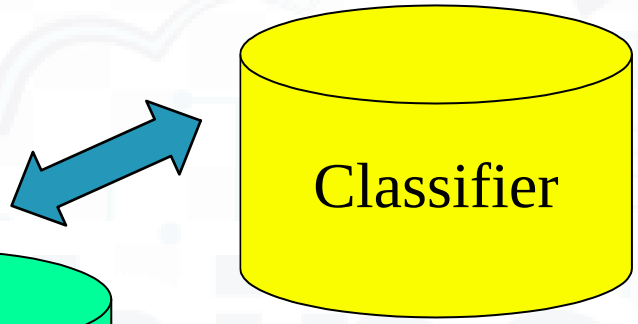


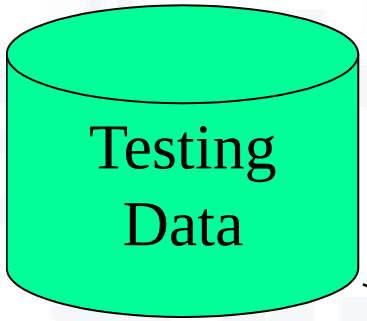
NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

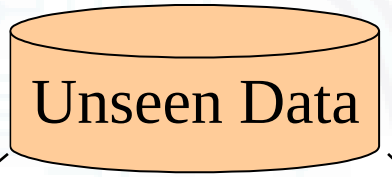
IF rank = ‘professor’
OR years > 6
THEN tenured = ‘yes’

Example: Step 2

IF rank = ‘professor’
OR years > 6
THEN tenured = ‘yes’







(Jeff, Professor, 4)

NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Tenured?

Yes

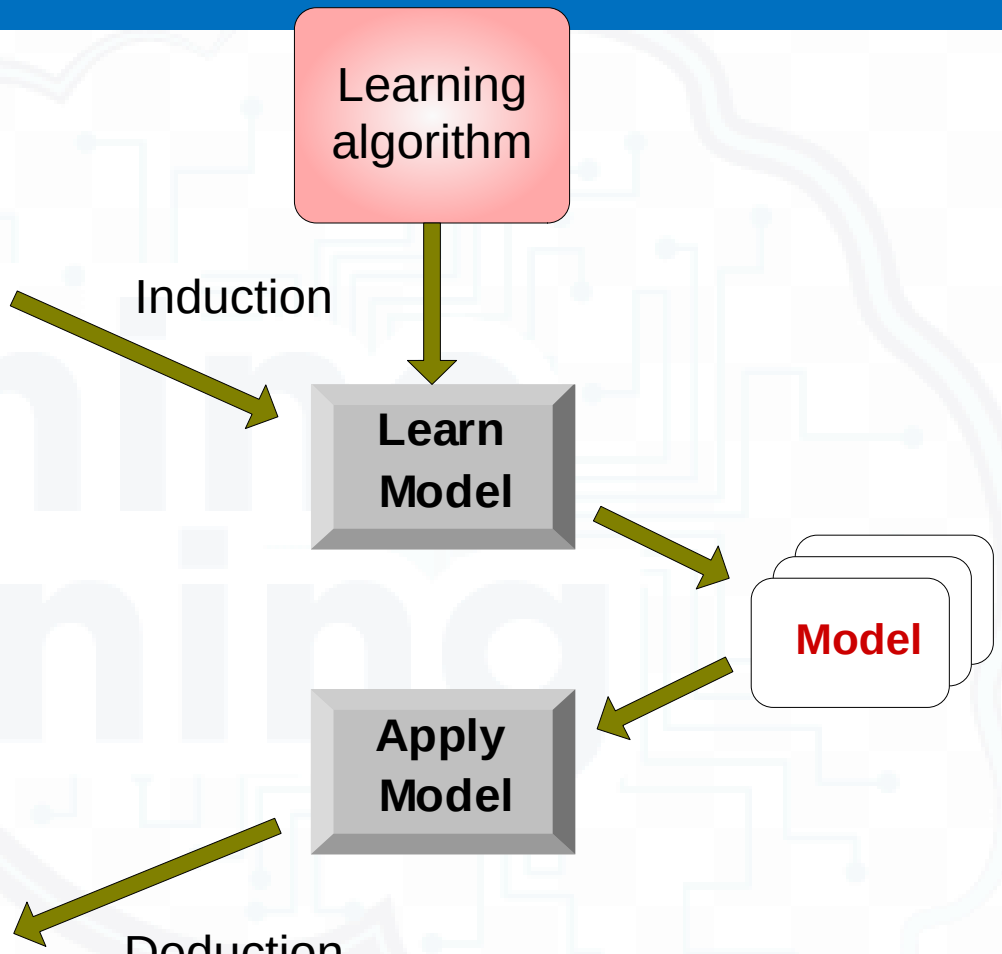
General Approach for Building Classification Model

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

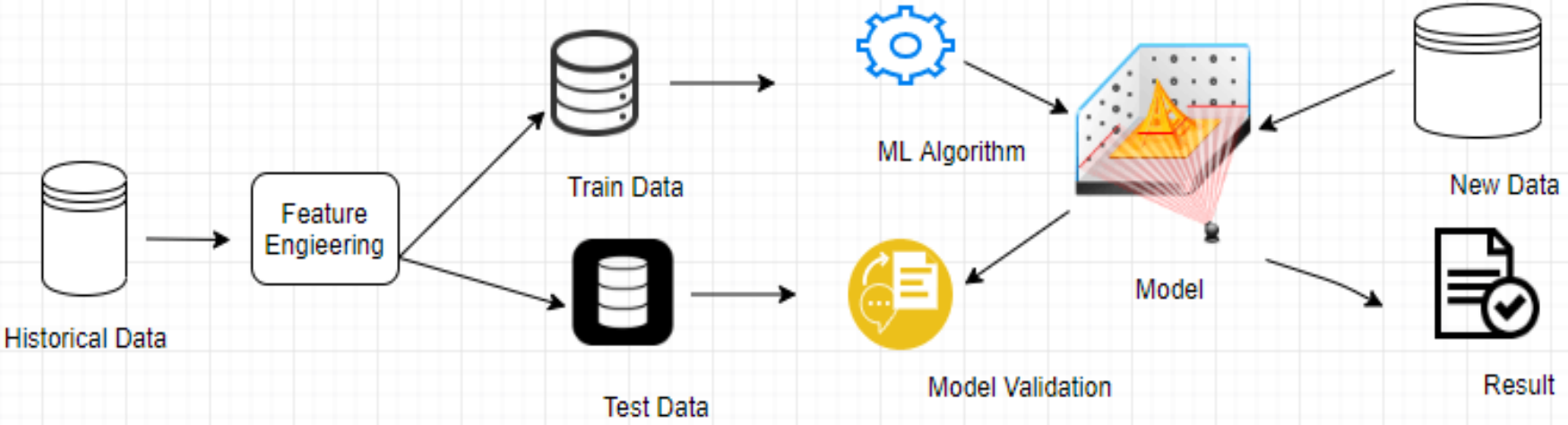


Classification Techniques

- Base Classifiers
  - Decision Tree-based Methods
  - Rule-based Methods
  - Nearest-neighbor
  - Neural Networks
  - Deep Learning
  - Naïve Bayes and Bayesian Belief Networks
  - Support Vector Machines
  - …
- Ensemble Classifiers
  - Boosting, Bagging, Random Forests

Model Evaluation

Model Evaluation



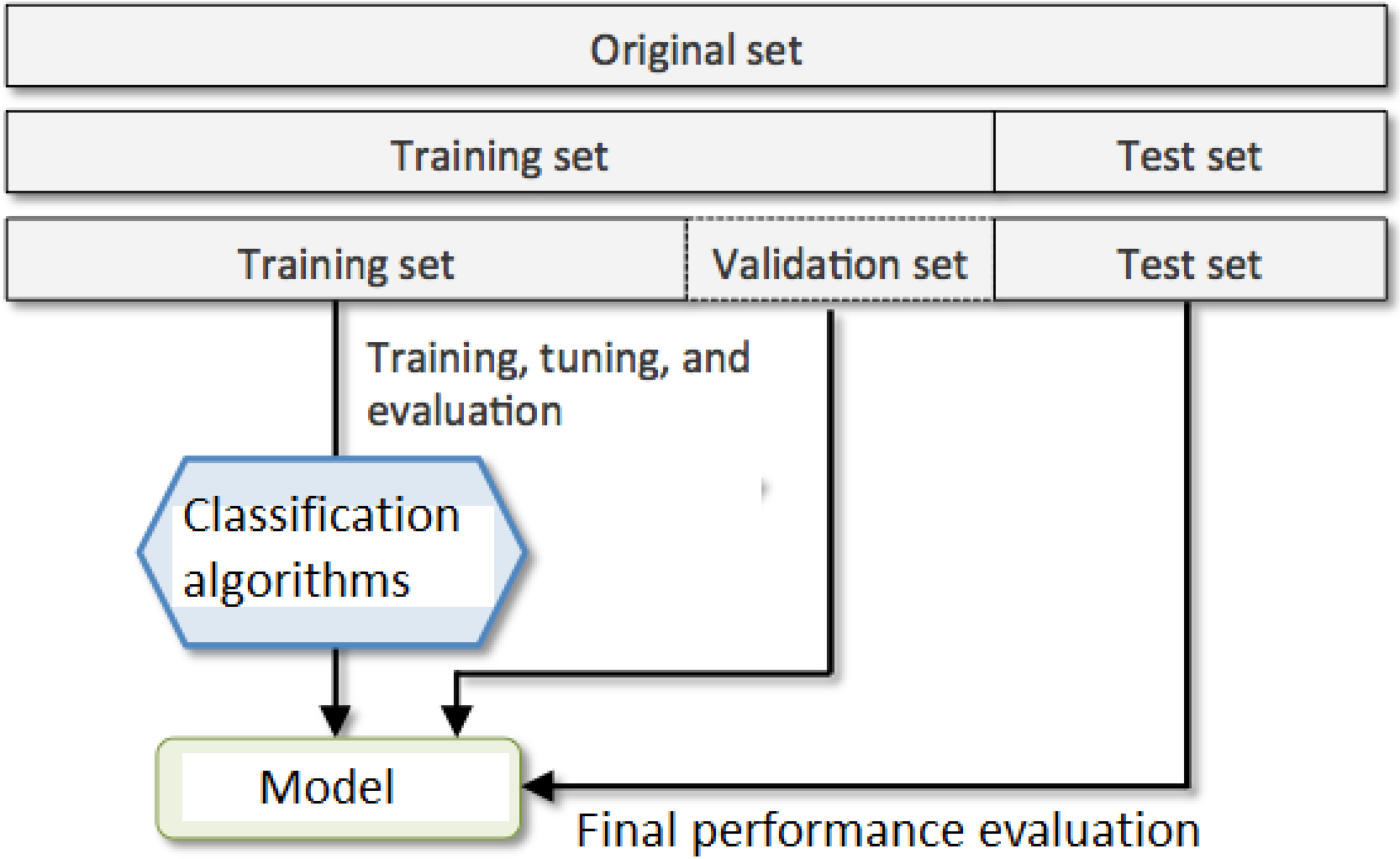
Model Evaluation and Selection

- Evaluation metrics: How can we measure accuracy? Other metrics to consider?
- Use **validation test set** of class-labeled tuples instead of training set when assessing accuracy
- Methods for estimating a classifier's accuracy:
  - Holdout method, random subsampling
  - Cross-validation
  - Bootstrap
- Comparing classifiers:
  - Confidence intervals
  - Cost-benefit analysis and ROC Curves

Model Evaluation

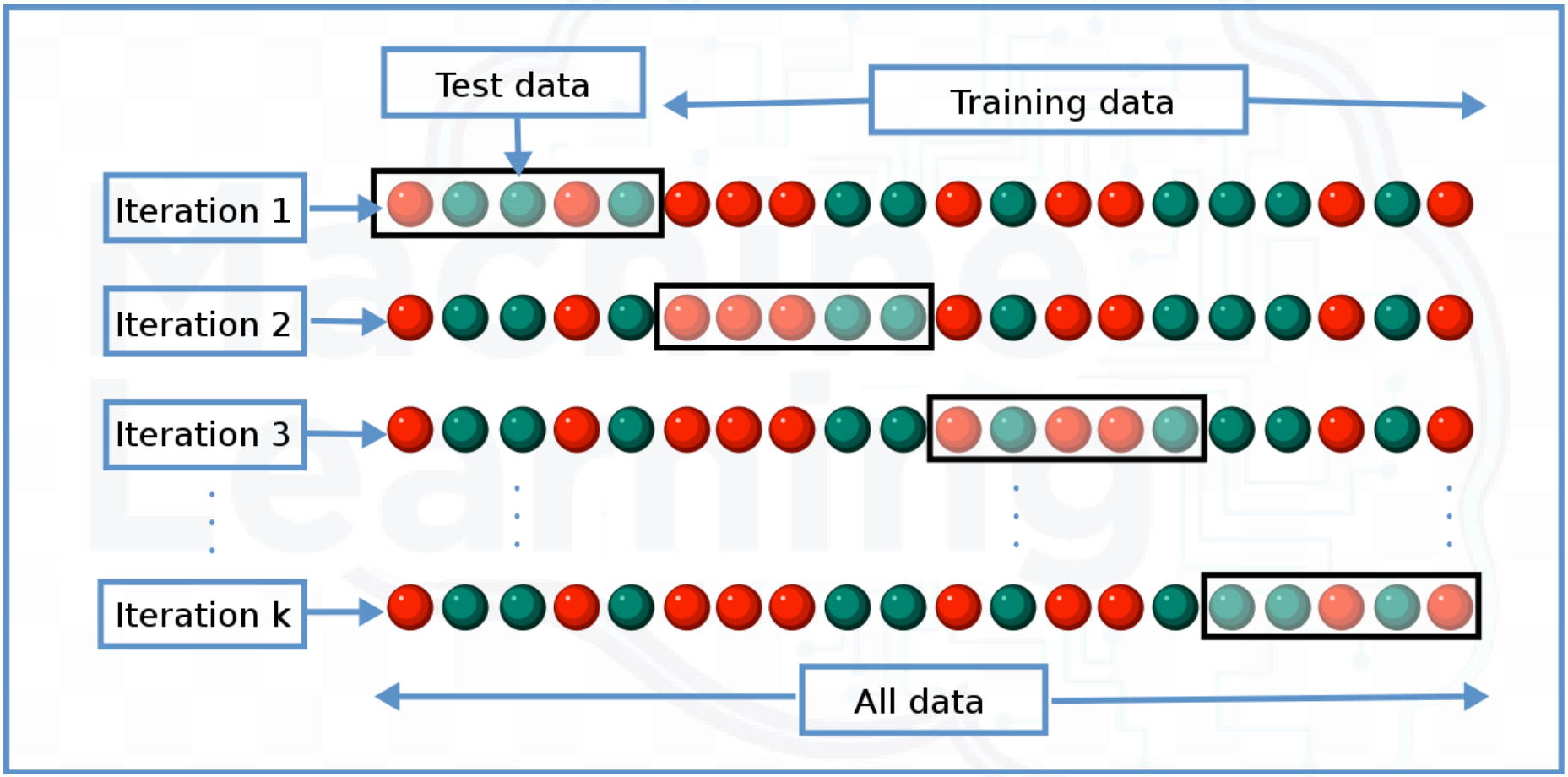
- Purpose:
  - To estimate performance of classifier on previously unseen data (test set)
- Holdout
  - Reserve k% for training and (100-k)% for testing
  - Random subsampling: repeated holdout
- Cross validation
  - Partition data into k disjoint subsets
  - k-fold: train on k-1 partitions, test on the remaining one
  - Leave-one-out: k=n

Model Evaluation: Holdout



bigdatauni.com

Model Evaluation: Cross validation



https://en.wikipedia.org/wiki/Cross-validation_(statistics)

Classifier Evaluation Metrics: Confusion Matrix

Actual class\Predicted class	C_1	$\neg C_1$
C_1	**True Positives (TP)**	**False Negatives (FN)**
$\neg C_1$	**False Positives (FP)**	**True Negatives (TN)**

Example of Confusion Matrix:

Actual class\Predicted class	buy_computer = yes	buy_computer = no	Total
buy_computer = yes	**6954**	**46**	7000
buy_computer = no	**412**	**2588**	3000
Total	7366	2634	10000

- Given *m* classes, an entry, $CM_{i,j}$ in a **confusion matrix** indicates # of tuples in class *i* that were labeled by the classifier as class *j*
- May have extra rows/columns to provide totals

Classifier Evaluation Metrics: Accuracy, Error Rate, Sensitivity and Specificity

- **Classifier Accuracy,** or recognition rate: percentage of test set tuples that are correctly classified

  Accuracy = (TP + TN)/All

- **Error rate:** *1 – accuracy*, or

  Error rate = (FP + FN)/All

A\P	C	¬C	
C	**TP**	**FN**	**P**
¬C	**FP**	**TN**	**N**
	P'	**N'**	**All**

Classifier Evaluation Metrics: Accuracy, Error Rate, Sensitivity and Specificity

- **Class Imbalance Problem**:
  - One class may be *rare*, e.g. fraud, or HIV-positive
  - Significant *majority of the negative class* and minority of the positive class
  - **Sensitivity**: True Positive recognition rate
    - **Sensitivity = TP/ (TP + FN)**
  - **Specificity**: True Negative recognition rate
    - **Specificity = TN/(FP + TN)**

A\P	C	¬C	
C	**TP**	**FN**	**P**
¬C	**FP**	**TN**	**N**
	P'	**N'**	**All**

Classifier Evaluation Metrics: Precision and Recall, and F-measures

- **Precision**: exactness – what % of tuples that the classifier labeled as positive are actually positive

$$precision = \frac{TP}{TP + FP}$$

- **Recall**: completeness – what % of positive tuples did the classifier labeled as positive? Perfect score is 1.0

$$recall = \frac{TP}{TP + FN}$$

A\P	C	¬C	
C	**TP**	**FN**	**P**
¬C	**FP**	**TN**	**N**
	P'	**N'**	**All**

Classifier Evaluation Metrics: Precision and Recall, and F-measures (cont.)

- **Precision**: exactness – what % of tuples that the classifier labeled as positive are actually positive

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{50}{50 + 0} = 1$$

➔ No negative tuples (¬C) are classified as positive (C).

Whether all positive tuples are labeled?

A\P	C	¬C	
C	TP=50	FN=50	100
¬C	FP=0	TN=50	50
	50	100	150

Classifier Evaluation Metrics: Precision and Recall, and F-measures (cont.)

- **Recall**: completeness – what % of positive tuples did the classifier labeled as positive? Perfect score is 1.0

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{50}{50 + 0} = 1$$

➔ All positives tuples are labeled correctedly.

How about negative tuples (¬C) are labeled as positive (C)?

A\P	C	¬C	
C	TP=50	FN=0	50
¬C	FP=50	TN=50	100
	100	50	150

Classifier Evaluation Metrics: Precision, Recall, and F-measures (cont.)

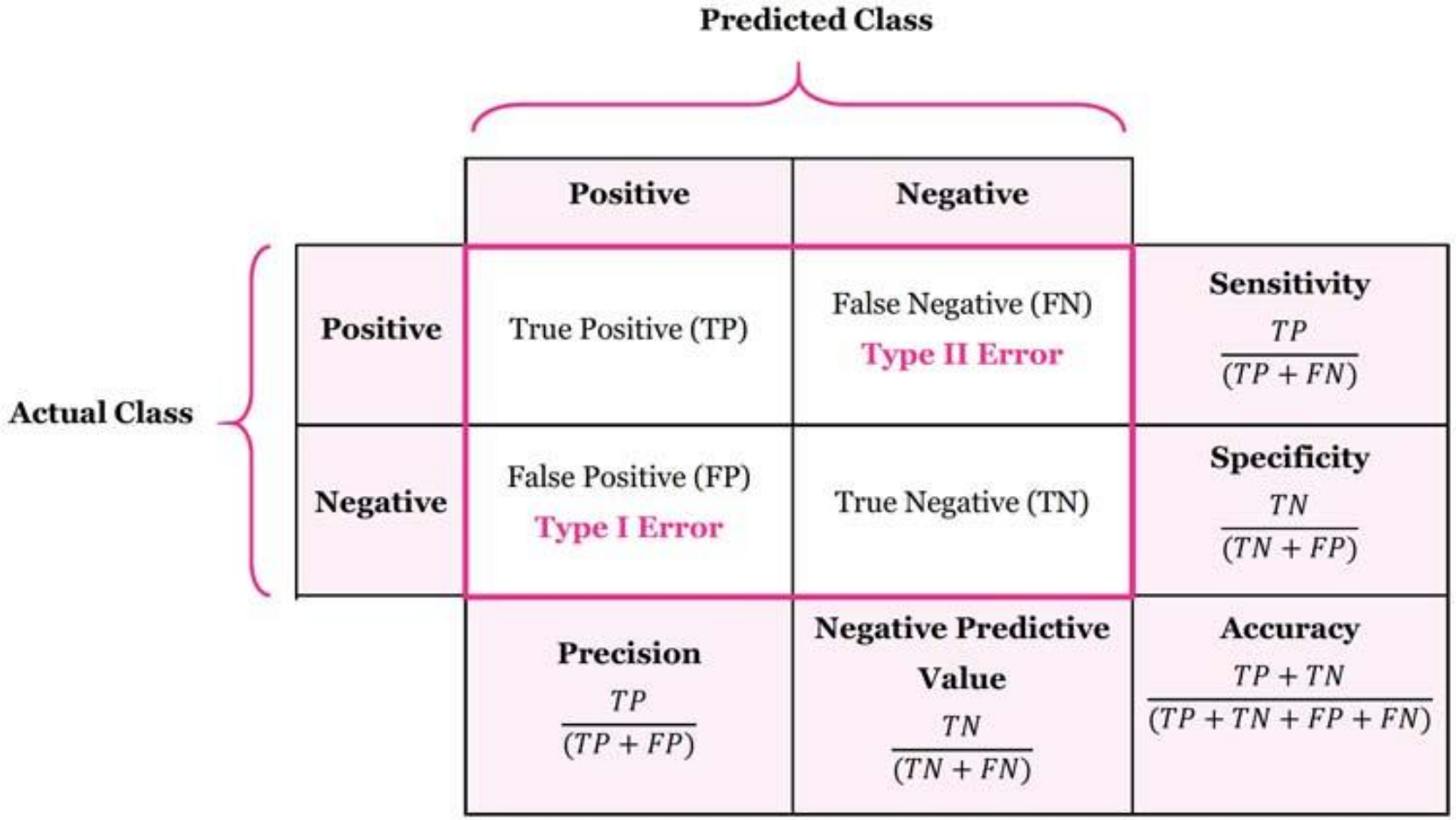
- Inverse relationship between precision & recall
- ***F* measure (*F*$_1$ or *F*-score)**: harmonic mean of precision and recall,

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

- F_β: weighted measure of precision and recall
  - assigns ß times as much weight to recall as to precision

$$F_\beta = \frac{(1 + \beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$$

Confusion matrix



Confusion matrix (cont.)

- *False Positive Rate (FPR):* called False Alarm Rate;
- *False Negative Rate (FNR)*: called Miss Detection Rate.

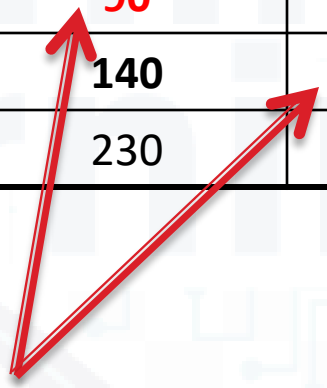
Actual\Predicted	Positive	Negative
Positive	$TPR = TP/(TP + FN)$	$FNR = FN/(TP + FN)$
Negative	$FPR = FP/(FP + TN)$	$TNR = TN/(FP + TN)$

- In some cases, a model having *high False Alarm Rate* can be accepted to get *low Miss Detection Rate*.

Classifier Evaluation Metrics: Example

- *Precision* = 90/230 = 39.13%
- *Recall* = 90/300 = 30.00%

Actual Class\Predicted class	cancer = yes	cancer = no	Total	Recognition(%)
cancer = yes	**90**	**210**	300	30.00 (*sensitivity*
cancer = no	**140**	**9560**	9700	98.56 (*specificity)*
Total	230	9770	10000	96.40 (*accuracy*)



Objective: these values are higher others