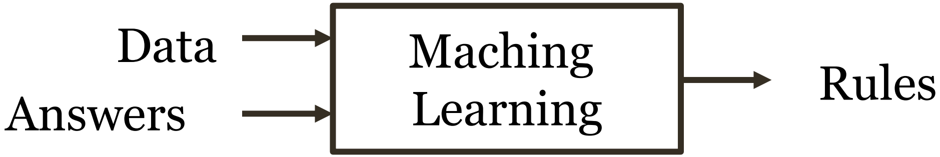
Introduction to Machine Learning

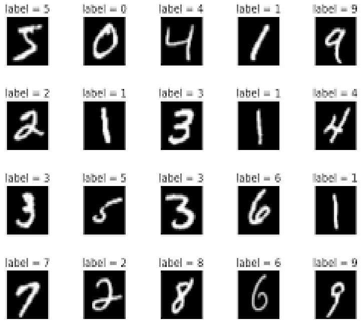
Abdessalam Bouchekif
abdessalam.bouchekif@epita.fr

2017/2018

Why Machine Learning?



What rule could we use to tell one digit from another?



- Machine learning aims at gaining insights from data and making predictions based on it.

- The goal of ML is to make machines able to learn and solve problems on their own

Data for machine learning

Structured Data

Area	estate type	Distance to center	Energy class	Age	Number bedrooms	Price
100	Apartement	1,1	A	20	3	130000
150	House	5,6	A	21	5	180000
247	House	2,2	C	20	7	250000
987	House	0,5	D	1	10	1250000

features

targets/lables

Data for machine learning

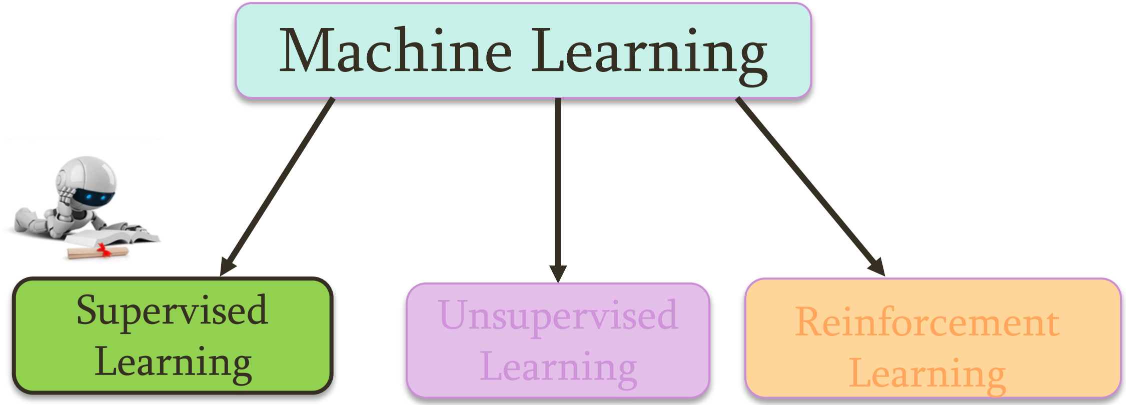
Unstructured data

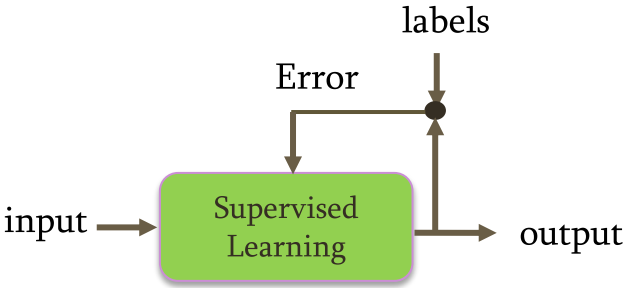
Category	Content
Sports	Bayern Munich was defeated by Real Madrid in the Champions League quarter finals...
Politics	Theresa May called for a snap general election on Monday...

targets/lables

features

Types of Machine Learning



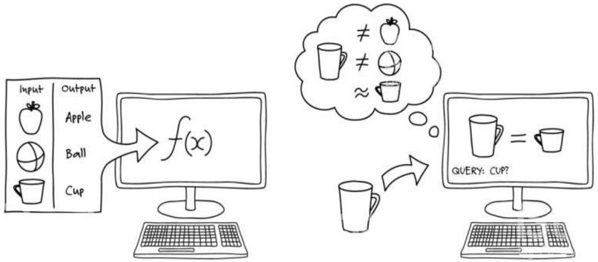


Training set $\{(x_1, y_1), \ldots, (x_n, y_n)\}$

Where each y_j was generated by a function unknown $y = f(x)$

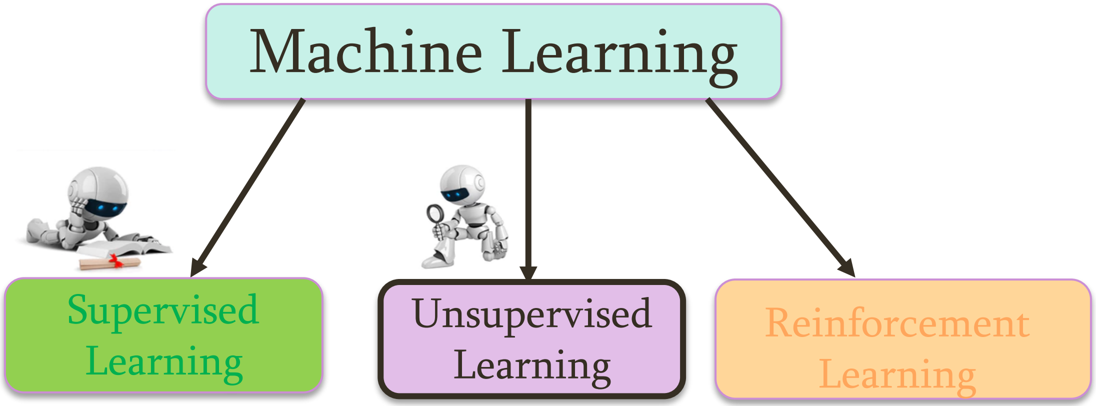
f is continuous $\Rightarrow$ regression

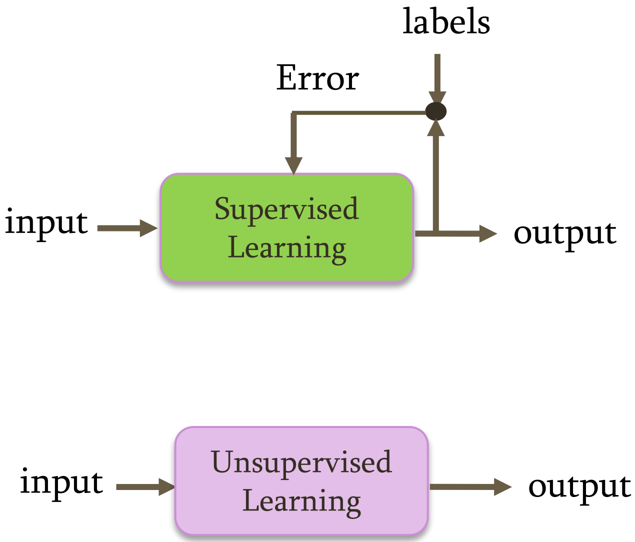
f is discrete $\Rightarrow$ classification



The goal is to find a function h that approximates the function f $(i.e\ f(x) \approx h(x))$

Types of Machine Learning



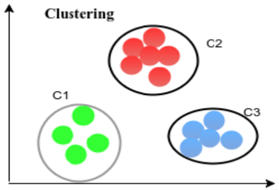


Problem: too much data!

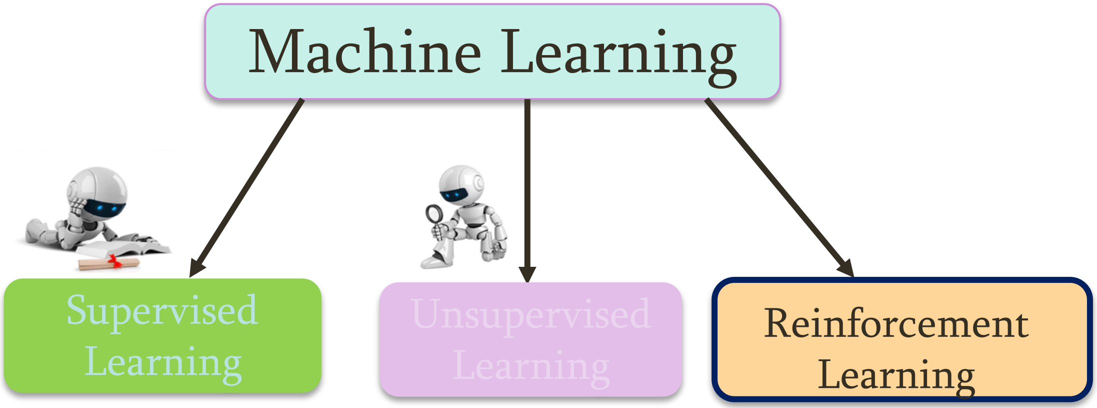
Solution: reduce it

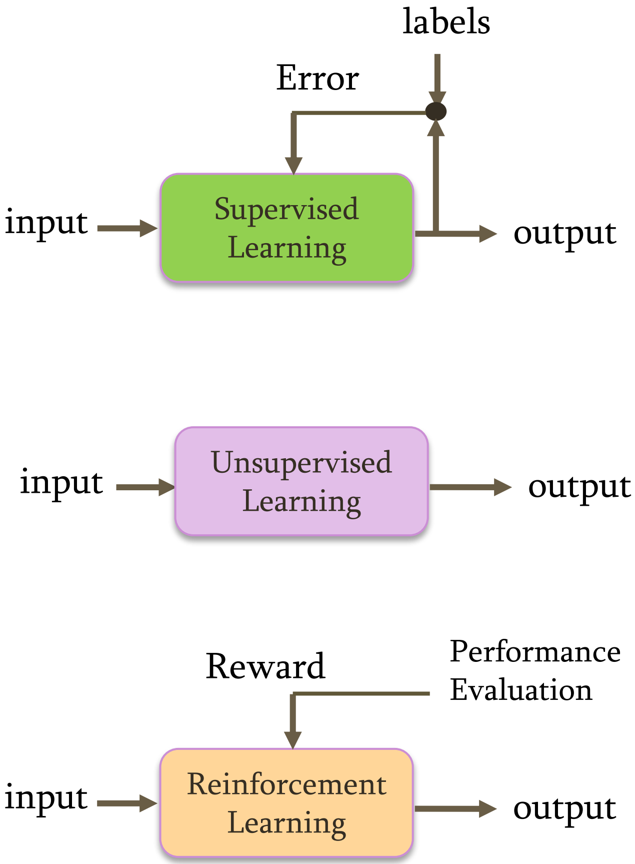
Clustering: reduce number of examples (discrete)

Dimensionality reduction: reduce number of dimensions (continuous)



Types of Machine Learning

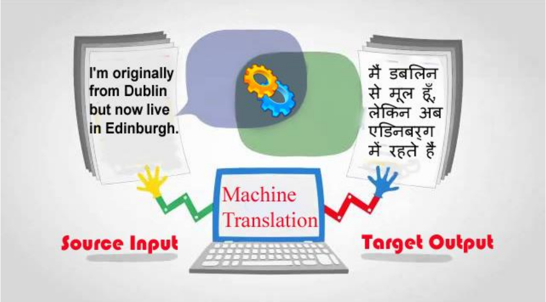


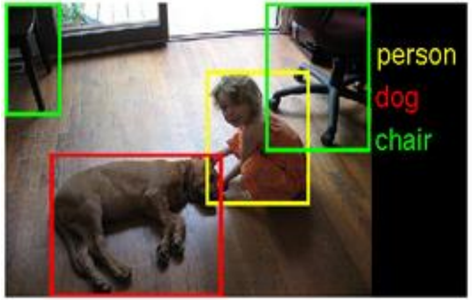


Examples applications of machine Learning

- determine sentiment (e.g., negative, neutral, positive) classification
- group newspaper articles according to topic clustering
- identify the broad topic (e.g., Sports, Politics, Culture) of a newspaper article classification
- predict monthly rent of an apartment you want to rent out regression
- predict fuel consumption of a car based on weight and horsepower regression
- classify an incoming **e-mail** as **spam** or **not-spam** classification
- find communities of users in a social network based on their interests and comments that they write clustering

Applications of Machine Learning





Regression

Linear Regression

$Price = $ 250,000 €

?

$Price = $ 990,000 €

$Area\ (m^2)$	$Price$ (€)
100	13000
150	18000
247	250000
987	990000

one training example

Notation

m: number of training examples

x: input / features

y: output / target

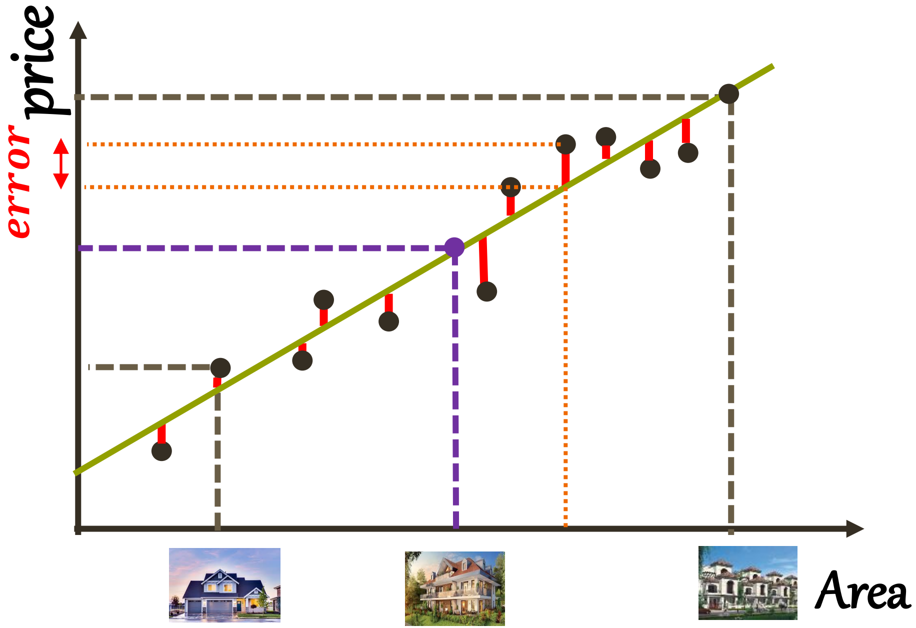
$(x^{(i)}, y^{(i)})$: i^{th} training example

Learning Algorithm

Training Set

Learning model

Size of house → h → Estimated price



$$error_i = \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

Best model = minimizes the prediction error

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x_1$

Parameters: $\theta_0, \ \theta_1$

Cost function:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{N} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

Goal:

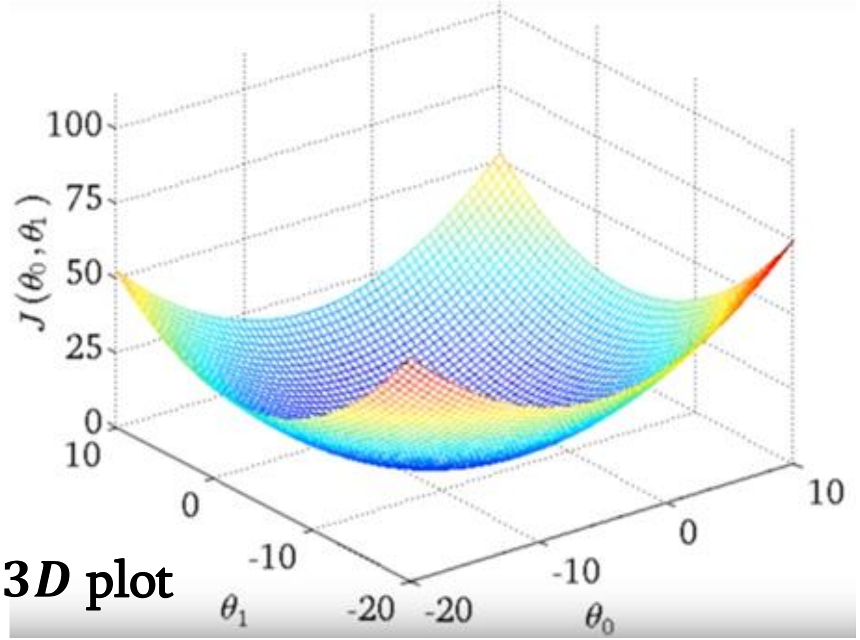
$$\underset{\theta_0, \theta_1}{minimize}\, J(\theta_0, \theta_1)$$

Mean Squared Error

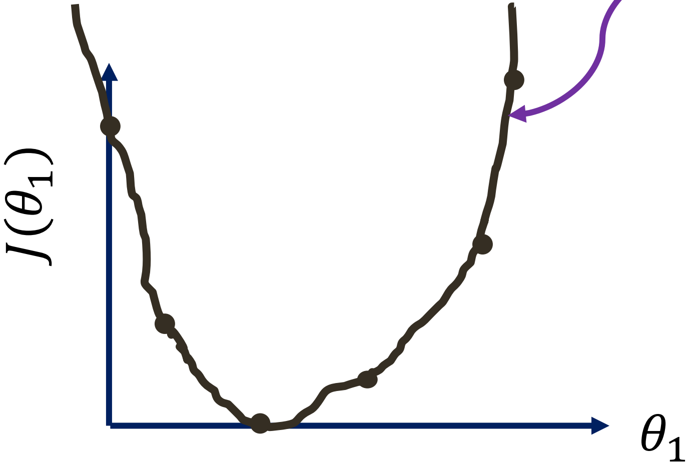
Erreur quadratique moyenne

Plot cost function

$$J(\theta_0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{N}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$



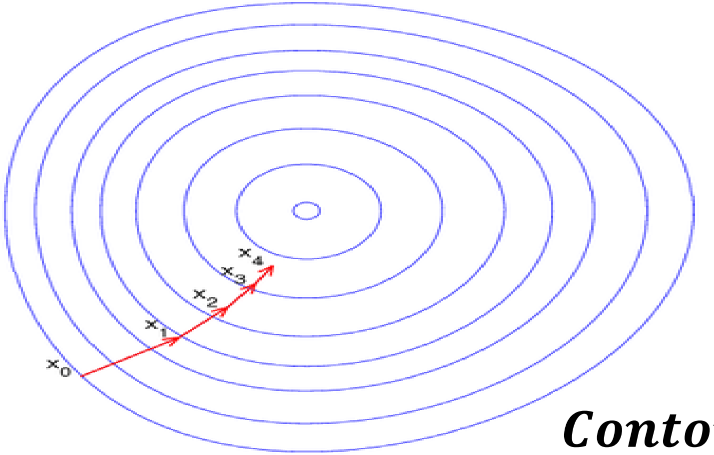
$3D$ plot



convex function

no local minima

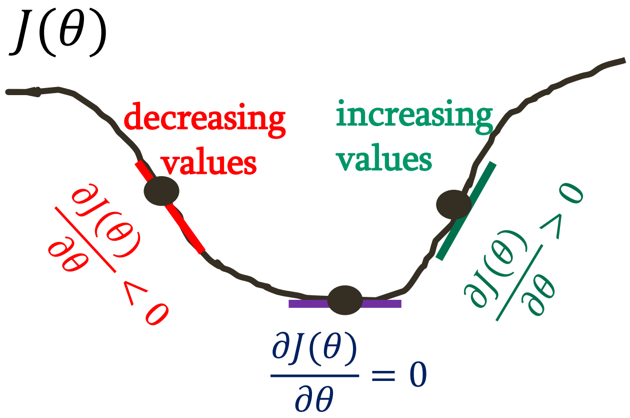
one global minimum



Contour plot

Gradient descent [3]

The gradient is the derivation of a multi-variable function



- How to change θ_0, θ_1 to improve $J(\theta_0, \theta_1)$?
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$

$$\frac{\partial J(\theta)}{\partial \theta} > 0 \Rightarrow decrease\ \theta$$

$$\frac{\partial J(\theta)}{\partial \theta} < 0 \Rightarrow increase\ \theta$$

Gradient descent

$initialization \;\; \theta$

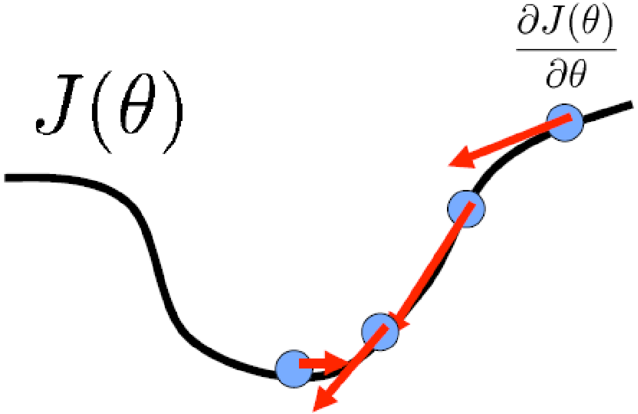
$\color{navy}{while} \;\; not \; converged$

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

$Return \;\; \theta_0, \theta_1$

Simultaneously update θ_1 and θ_2

for
$j = 0..1$



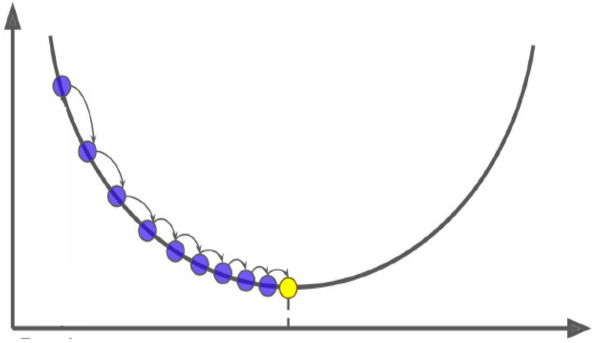
α learning rate controls how much of a change we make to our model parameters

$$\begin{cases} \theta_0 = \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \\ \theta_1 = \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x^{(i)} \end{cases}$$

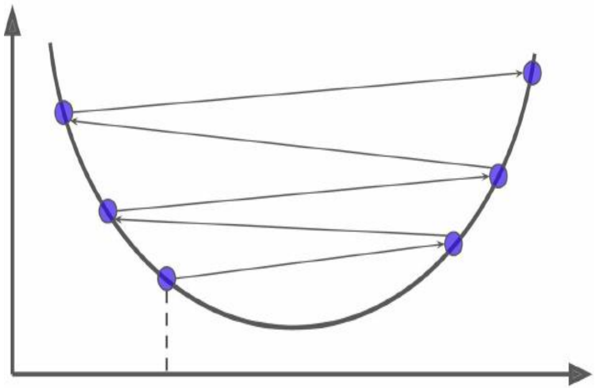
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Learning rate

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$



α is **small** ➡ Many iterations until convergence and trapping in local minima.



α is **too large** ➡ Overshooting.

Often $\alpha = 0{,}001$

Using multiple input features

Area	estate type	energy class	age	number bedrooms	price
100	Apartement	A	20	3	130000
150	House	A	21	5	180000
247	House	C	20	7	250000
987	House	D	1	10	1250000

$$\hat{y} = h_\theta(x) = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_N$$

Notation

N: number of training examples

x: input / features

y: output / target

$(x^{(i)}, y^{(i)})$: i^{th} training example

$x_j^{(i)}$ feature j in i^{th} training example

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n$

Parameters: $\theta = \theta_0,\ \theta_1, \ldots, \theta_n$

Cost function:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

Vectorized form of lineair regression

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n$$

$$X = \begin{bmatrix} x_0 \\ x_1 \\ . \\ . \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ . \\ . \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

with definition $x_0 = 1$

$$h_\theta(x) = [\theta_0 \quad \theta_1 \quad \ldots \theta_n] \begin{bmatrix} x_0 \\ x_1 \\ . \\ . \\ x_n \end{bmatrix}$$

$$= \theta^T X$$

Gradient descent ($n > 1$)

$initialization\ \ \theta$

$while\ \ not\ converged$

Simultaneously update θ_j (j=1,..,n)

$$tmp_j \leftarrow \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) x_j^{(i)}$$

$$Return \begin{pmatrix} tmp_0 \\ \vdots \\ tmp_1 \end{pmatrix}$$

$$\left\{\begin{array}{l} \theta_0 = \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) \\ \theta_1 = \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) x^{(i)} \\ \theta_2 = \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) x^{(i)} \\ \quad \cdots \\ \quad \cdots \\ \theta_n = \theta_n - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) x^{(i)} \end{array}\right.$$

$$\begin{aligned} \frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 \\ &= \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} \left(\left(\theta_0 x_0^{(i)} + \cdots + \theta_n x_n^{(i)}\right) - y^{(i)}\right)^2 \\ &= \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)} \end{aligned}$$

Gradient descent in practice

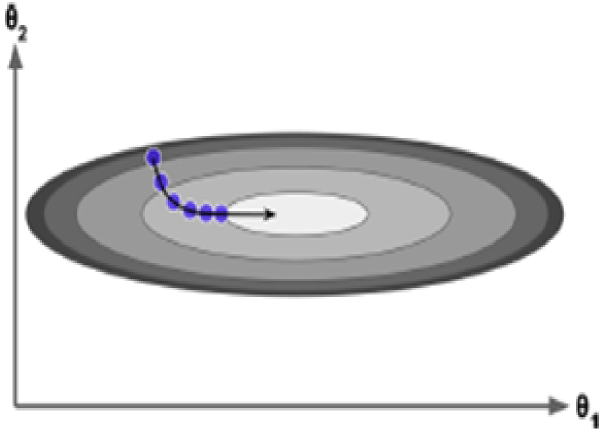
Gradient descent converges faster for features on similar scale

E.g. $x_1 = area(100 - 987)$

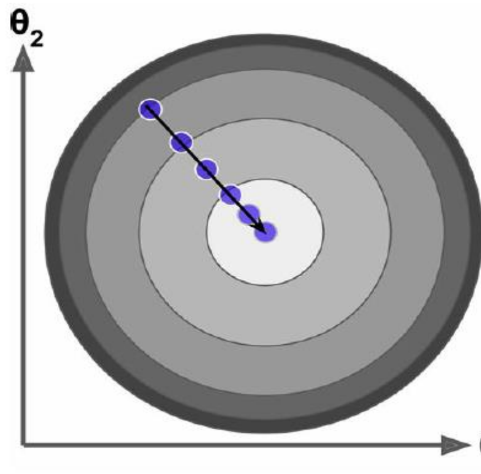
$x_2 = number\ of\ bedrooms\ (3 - 10)$

$$x_1 = \frac{area}{987}$$

$$x_2 = \frac{number\ of\ bedrooms}{10}$$



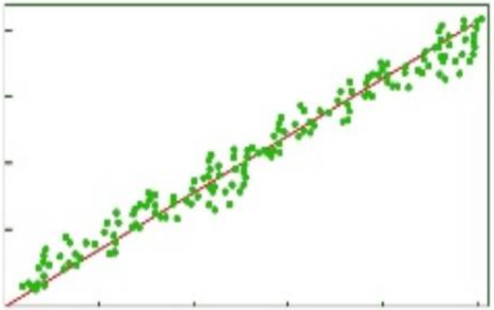
$$0 \le x_i \le 1$$



Types of Regression

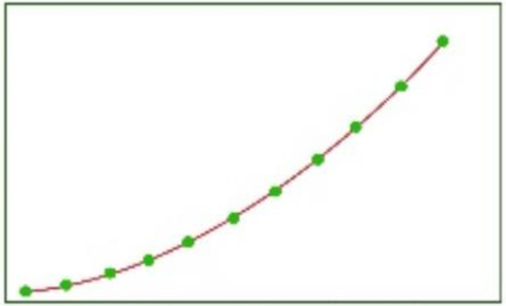
Linear Regression

When there is linear relationship between independant (predictor) and dependent (target) variables.



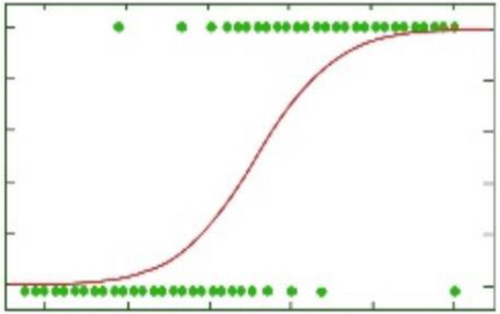
Polynomial Regression

When there is no linear relationship between independant and dependent variables.

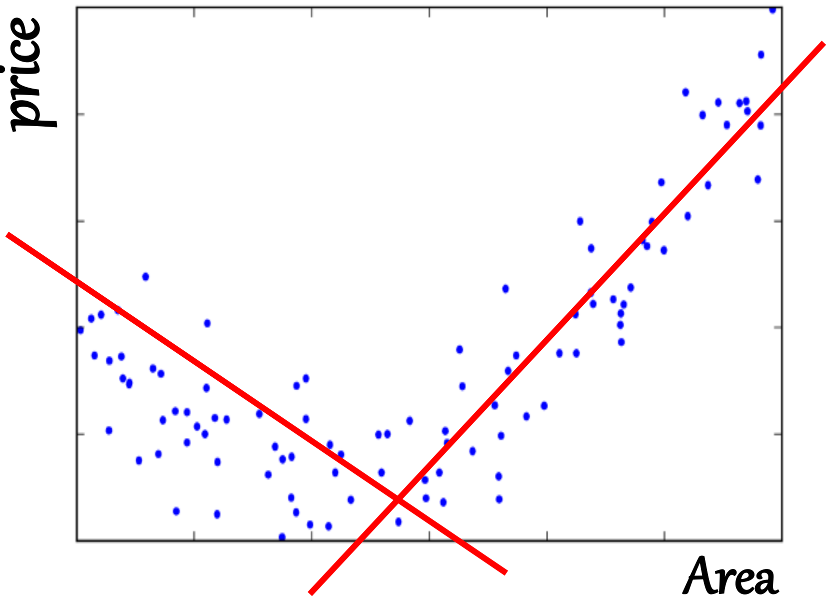


Logistic Regression

When the dependent variables is categorical (True /False, negative / positive/neutral, …) in nature



Polynomial Regression



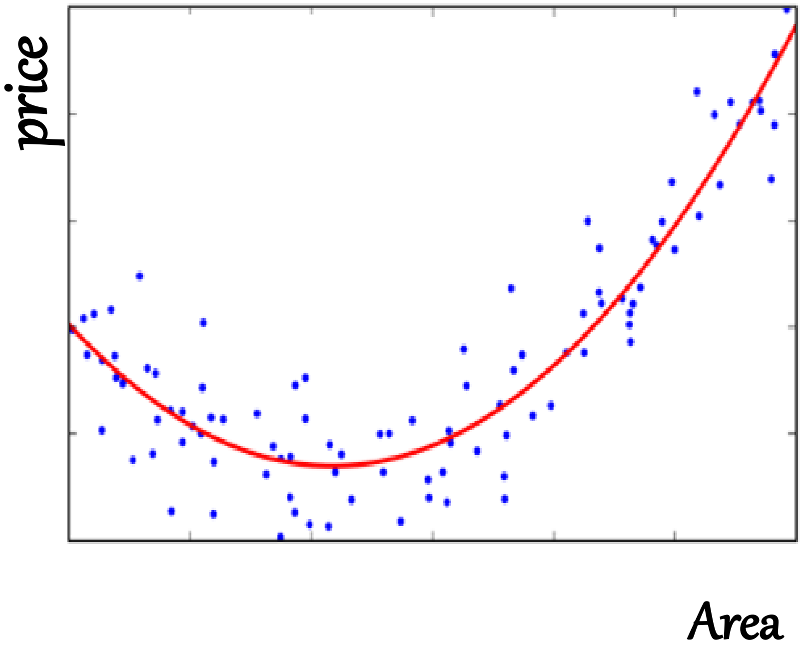
Idea

- Add powers of each feature as new features

$$h_\theta(x_1, \ldots, x_n) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \cdots$$

$$h_\theta(x_1, \ldots, x_n) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_2 x_1^2 x_2 + \cdots$$

Polynomial Regression



Idea

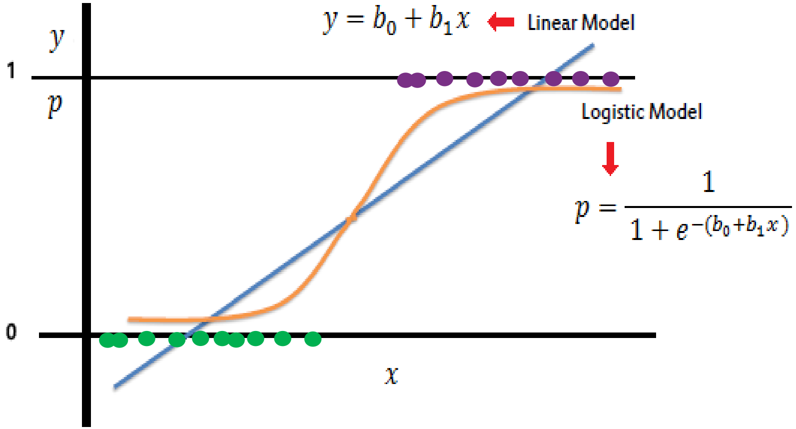
- Add powers of each feature as new features

$$h_\theta(x_1, \ldots, x_n) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \cdots$$

$$h_\theta(x_1, \ldots, x_n) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_2 x_1^2 x_2 + \cdots$$

Logistic Regression

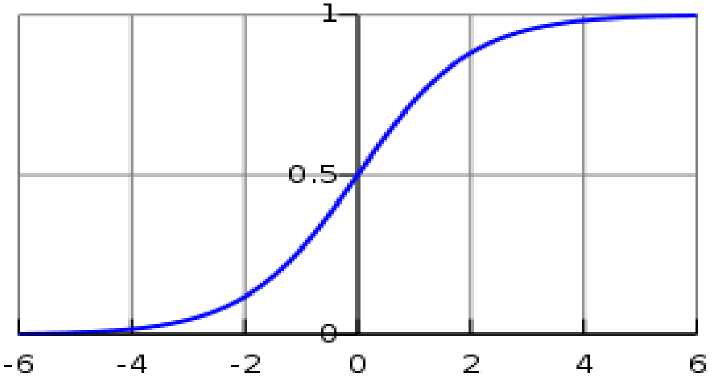
Logistic Regression



$$h_\theta(x) = \sigma(\theta^T . X)$$

$\sigma(.)$ is a sigmoid (or logistic) function

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



Linear regression: predicted y can exceed 0 and 1 range

Logistic regression: predicted y lies within 0 and 1 range

Logistic Regression

- Used to estimate the probability that an instance belongs to a particular class

- There are three types of logistic regression

  - Binary logistic model used to estimate the probability of a binary response ($i.e$ binary classification).

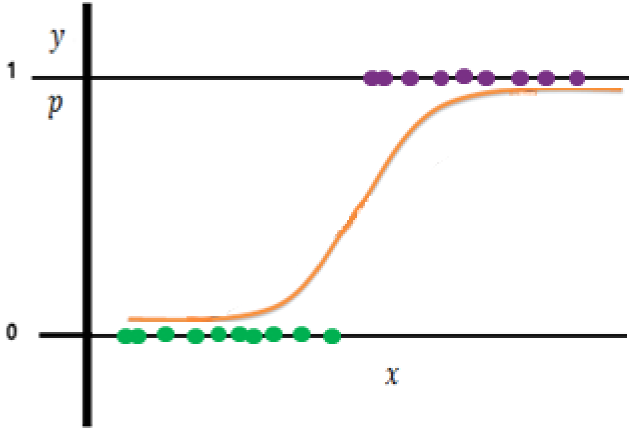
  - Ordinal logistic model generalizes binary logistic to multiclass problems.

    Example: classify the tweet into one of 3 categories: positive, neutral and negative

  - Nominal logistic model = ordinal logistic but takes into account the order of dependent variables

    Example: classify the tweet into one of 5 categories: very positive/ slightly positive/neutral/slightly negative/very negative

Binary Logistic Regression



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

or equivalenty $\theta^T x \geq 0$

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

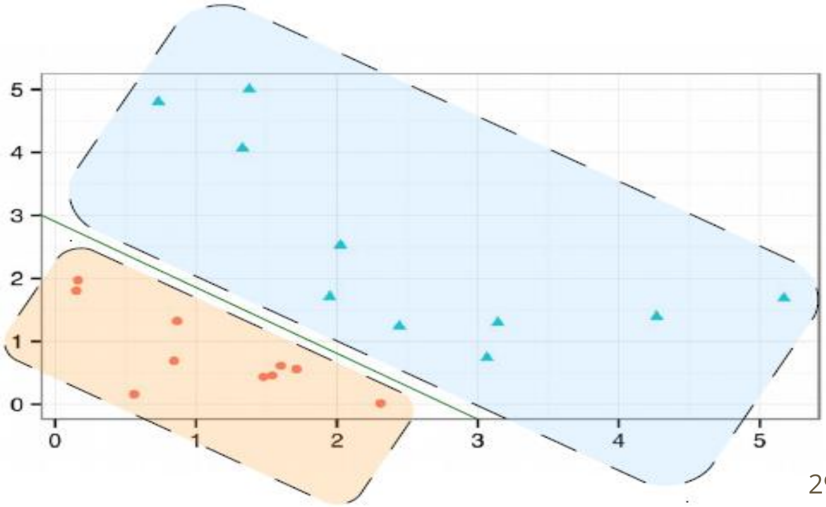
or equivalenty $\theta^T x < 0$

Example $h_\theta(x) = \sigma(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

$$and\ \theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$$

Prediction $y = 1$ whenever

$$-3 + x_1 + x_2 \geq 0$$



Non-linear decision boundaries

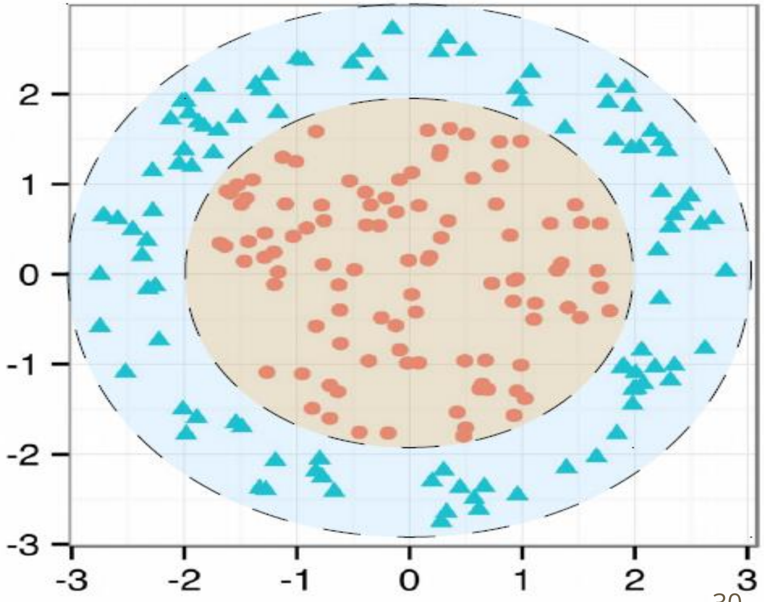
$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_1 + \theta_4 x_2^2$$

$$and$$

$$\theta = [-4 \quad 0 \quad 0 \quad 1 \quad 1]^T$$

Prediction $y = 1$ whenever

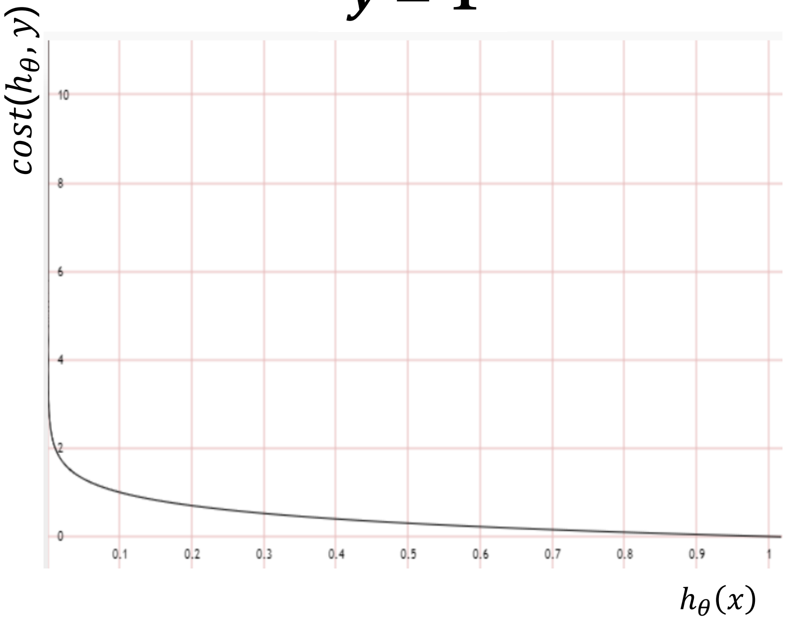
$$x_1^2 + x_2^2 \geq 4$$



Logistic regression cost function

$$cost(h_\theta, y) = \begin{cases} -\log(h_\theta(x)) & if \quad y = 1 \\ -\log\left(1 - h_\theta(x)\right) & if \quad y = 0 \end{cases}$$

$\boldsymbol{y = 1}$



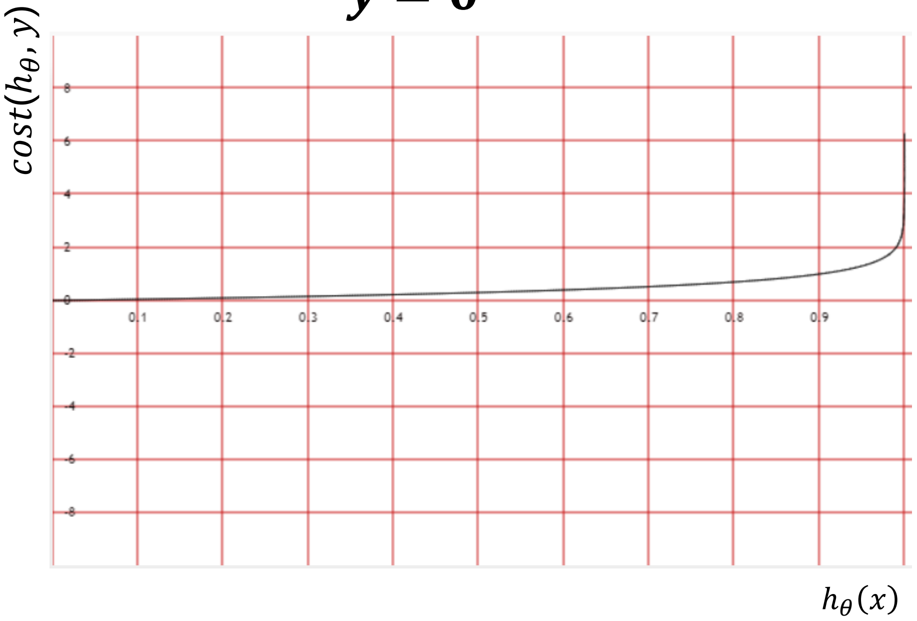
$$h_\theta(x) = 1 \implies cost(h_\theta, y) = 0$$

$$h_\theta(x) = 0 \implies cost(h_\theta, y) \text{ very lage}$$

Logistic regression cost function

$$cost(h_\theta, y) = \begin{cases} -\log(h_\theta(x)) & if \quad y = 1 \\ -\log\left(1 - h_\theta(x)\right) & if \quad y = 0 \end{cases}$$

$\boldsymbol{y = 0}$

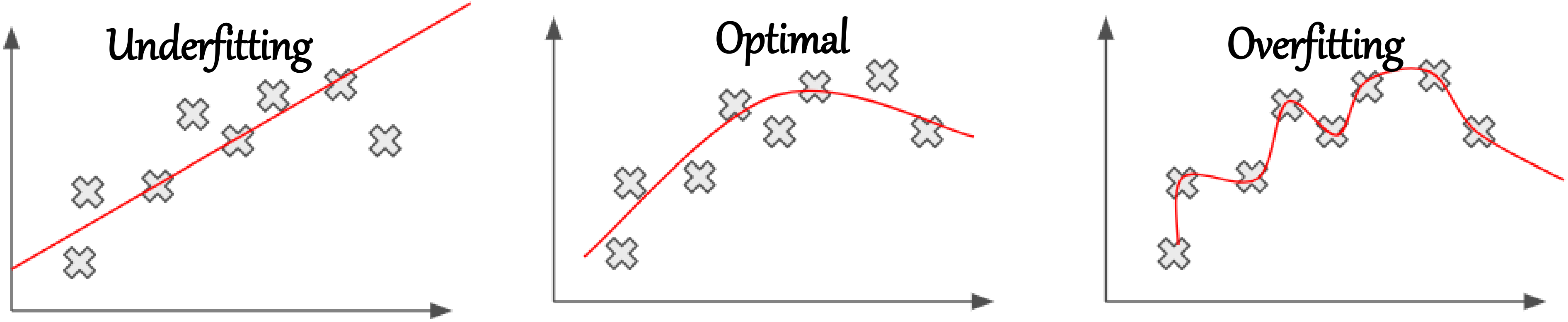


$h_\theta(x) = 0 \implies cost(h_\theta, y) = 0$

$h_\theta(x) = 1 \implies cost(h_\theta, y)$ very lage

Overfitting

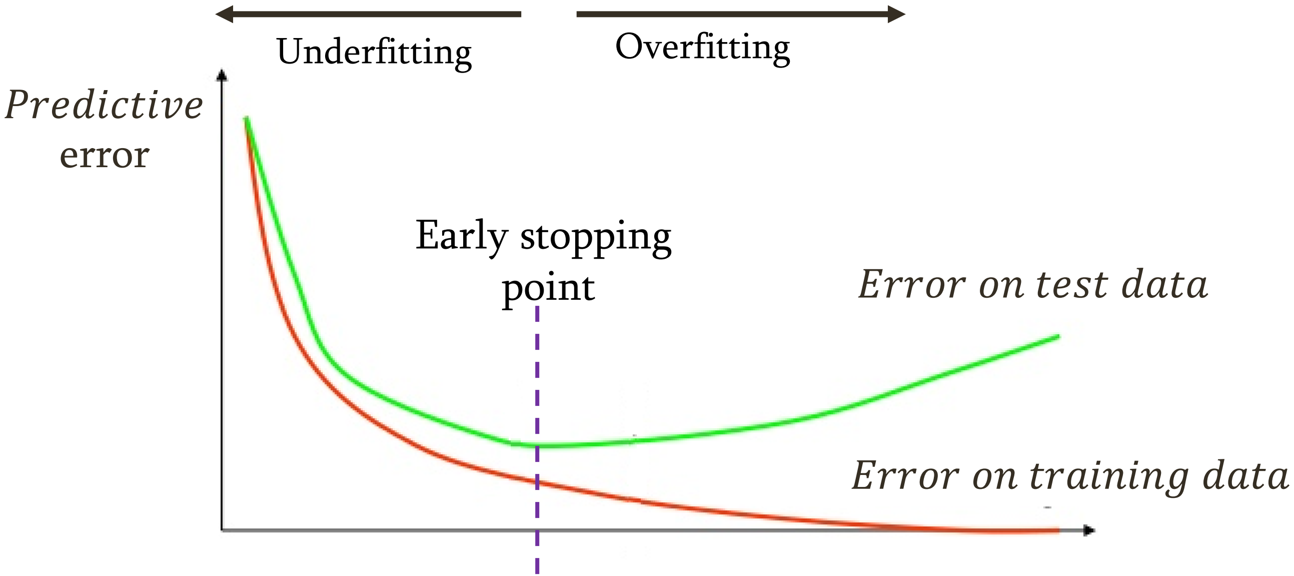
Which is the best?



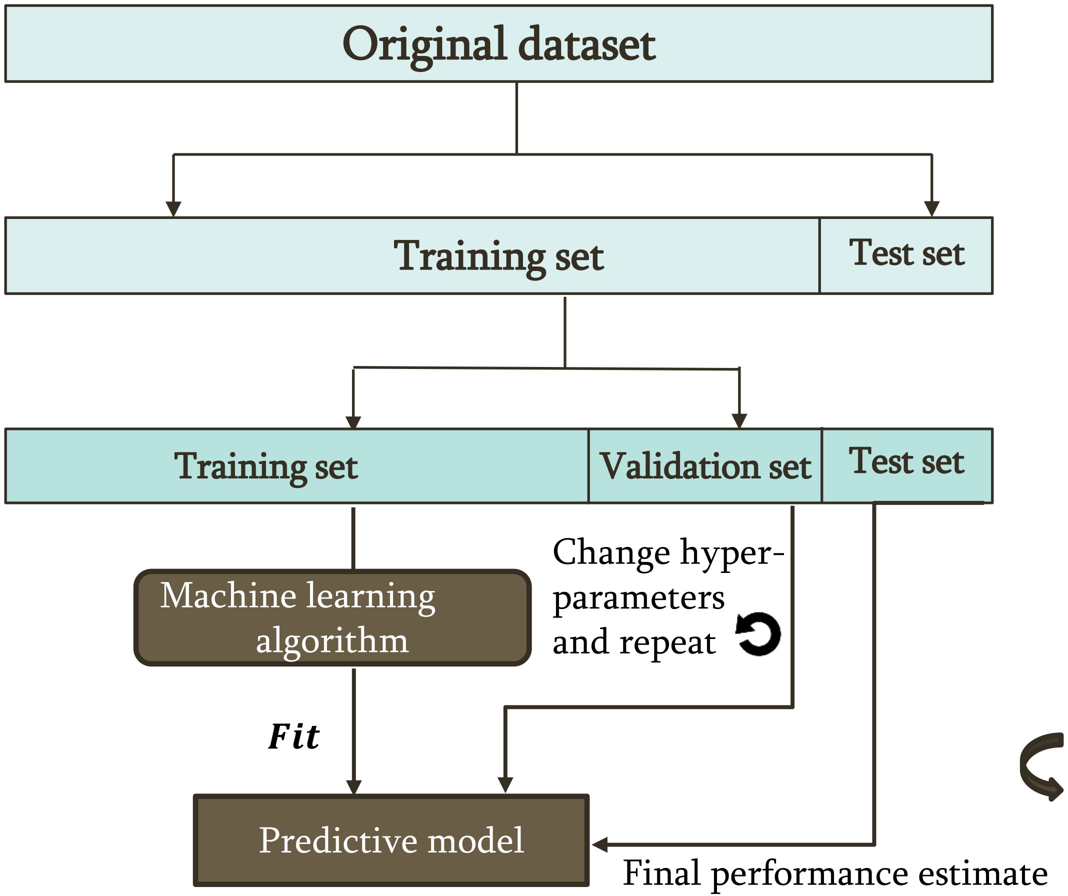
- Model performs well on the training data, but not generalizing well to new data.

Detecting Overfitting

- Separate the initial dataset into two sets
  - Training (70 %)
  - Test (30 %)



Detecting Overfitting

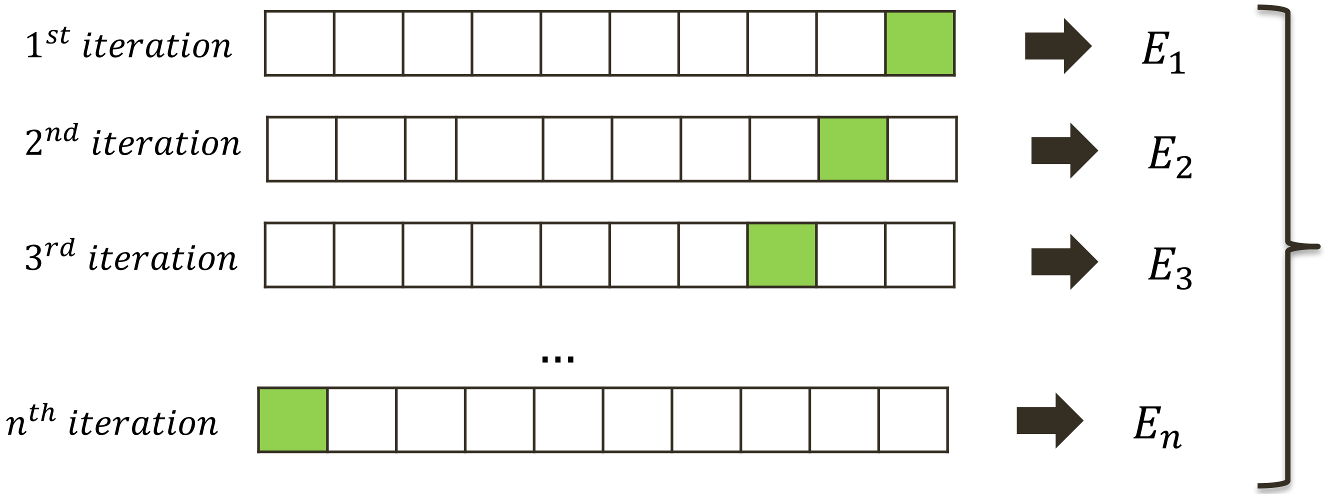


- **Training set** is used to learn the models
- **Validation set** is used to estimate prediction error for model selection
- **Test set** is used for assessment of the generalization error of the model

Reduce the number of samples which can be used for learning the model

K-fold cross-validation

- Divide the data the training set into k parts
- Use $k-1$ of the parts for training and 1 for testing.
- Repeat the procedure k times, rotating the test set.



$$E = \frac{1}{n}\sum_{i=1}^{n} E_i$$

- This approach can be computationally expensive

Evaluating Models

Confusoin Matrix

- Show how many predictions have been done right and how many have been wrong.

- Let P the label of class 1 and N the label of second class or the label of all classes that are not class 1

		Predicted	
		P	N
Actual	P	True positives (TP)	False Nagatives (FN)
	N	False Positives (FP)	True Negatives (TN)

Metrics - Classification

True positive rate

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

False positive rate

$$FPR = \frac{FP}{FP + TN}$$

Accuracy

$$Acc = \frac{TP + TN}{FP + FN + TP + TN}$$

Precision

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

Recall

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

F-score

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

Example

		Predicted class	
		cancer	no_cancer
Actual class	cancer	90	210
	no_cancer	140	9560

$$Acc = \frac{90 + 9560}{140 + 210 + 90 + 9560} = 96{,}5\%$$

$$Precision = \frac{90}{230} = 39{,}13\%$$

$$Recall = \frac{90}{300} = 30{,}00\%$$

Correctly classified

- 90 of samples that belong to class *cancer* (TP)
- 9560 of samples that belong to class *no_cancer* (TN)

Misclassified

- 210 samples from class *cancer* as class *no_cancer* (FN)
- 140 samples from class no_*cancer* as class *no_cancer* (FN)

Conclusion

Is this A or B? → Classification algorithms (Supervised Learning)

How much or how many? → Regression algorithms

How is organized? → Clustering

What should I do next? → Reinforcement