Some basics on ML

Supervised and Unsupervised Learning

Simachew Mengiste

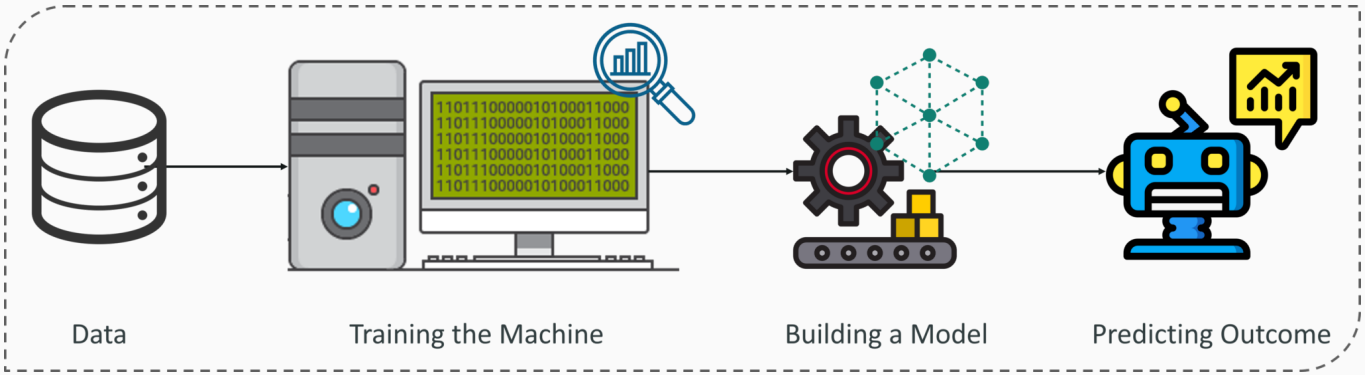
April 03, 2020

Machine Learning

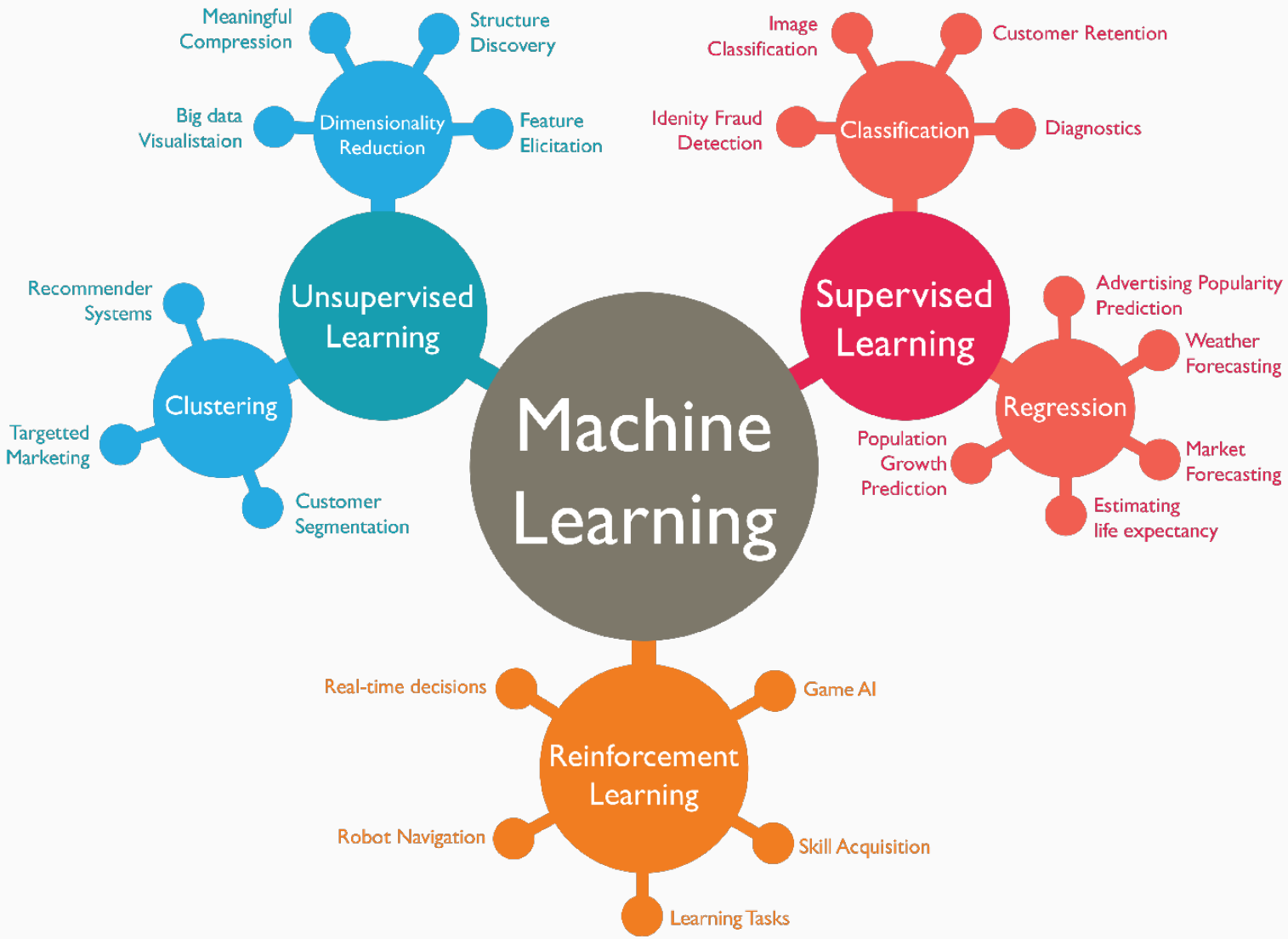
Machine learning

- **Arthur Samuel (1959)**
  Machine learning is a field of study that gives computers the ability to learn without being explicitly programmed.
- **Tom Mitchell (1998)**
  A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

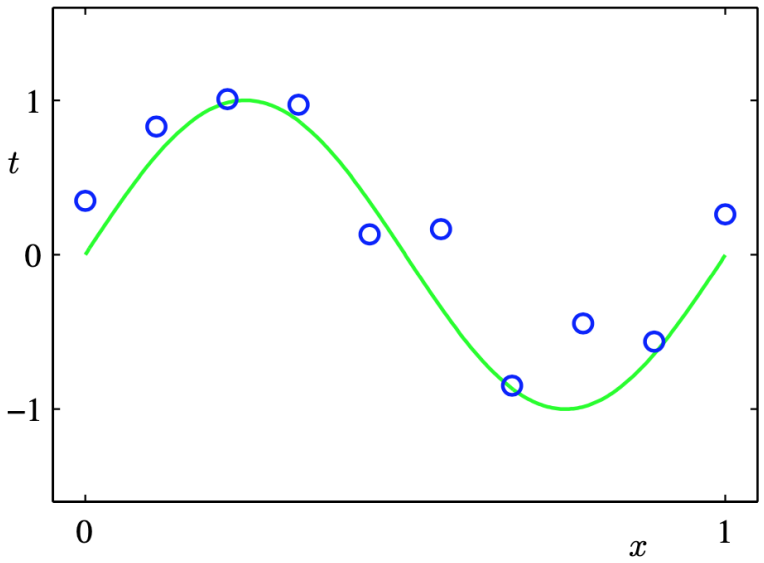
- Example: spam email filtering problem
  - T Classifying emails as spam or not spam.
  - E Watching you label emails as spam or not spam.
  - P The number of emails correctly classified as spam/not spam.



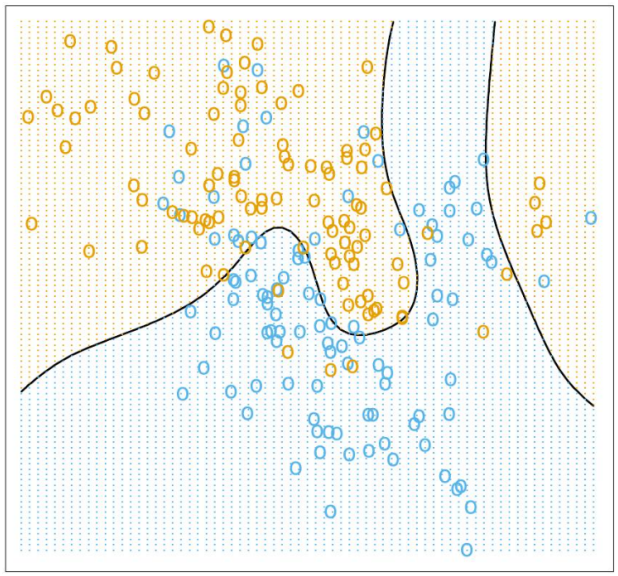
Categories of learning algorithms



Supervised learning



(a) Regression



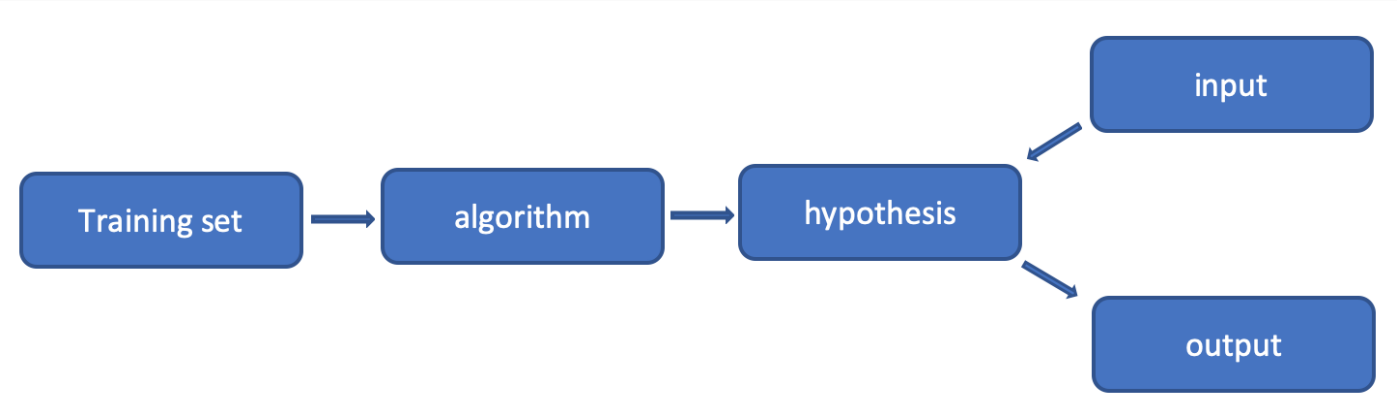
(b) Classification

- fitting function
  - outputs real values
  - univariate or multivariate

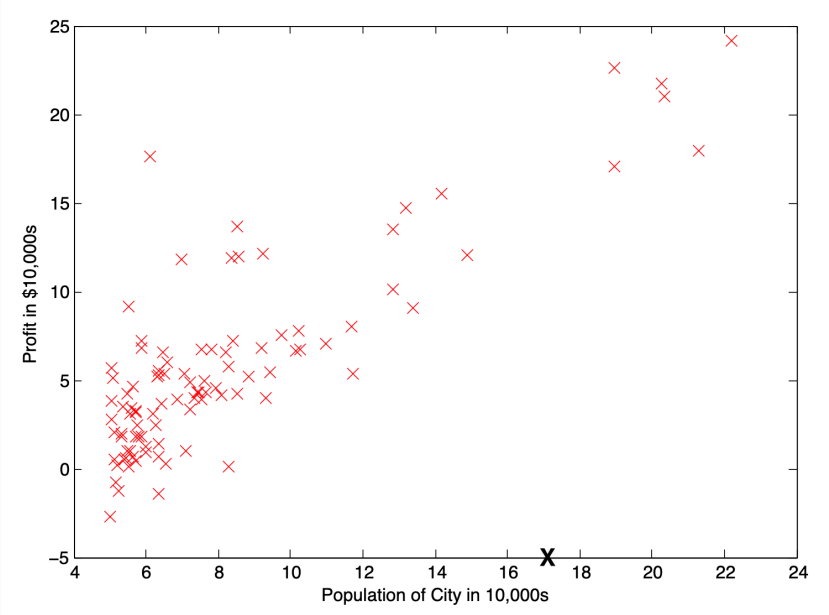
- boundary function
  - outputs labels
  - binary or multi-class

Linear regression

Linear regression

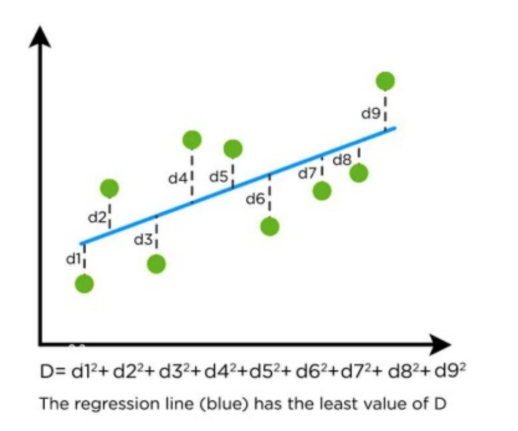


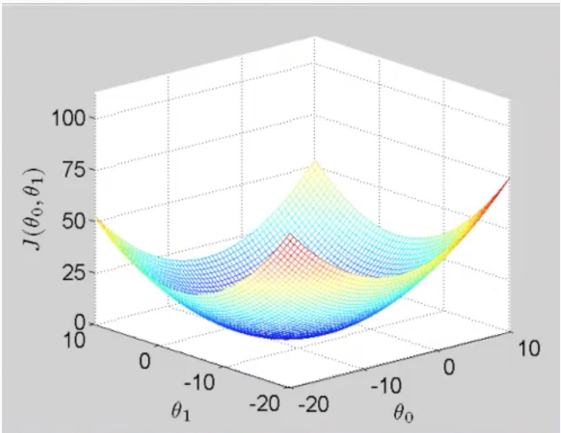
Univariate case:

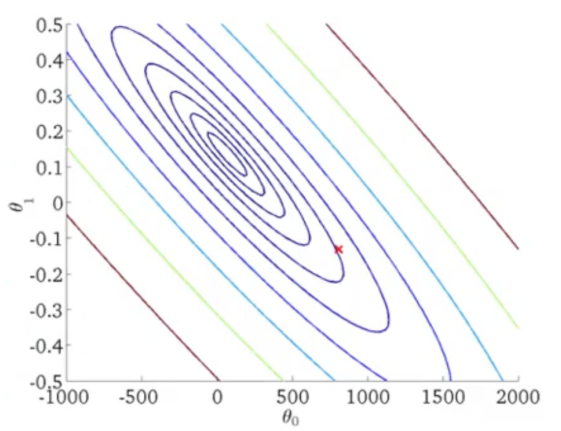


- profit from 170K ppl?
- linear model: $h_\theta(x) = \theta_0 + \theta_1 x$
- estimating $\theta = (\theta_0, \theta_1)$ for good prediction
- better is less error in the estimate

Cost function







For a training set m samples: $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$,

$d_i = (y^{(i)} - \tilde{y}^{(i)})^2$, where $\tilde{y}^{(i)} = h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$.

- $y^{(i)}$ is the observed output and $\tilde{y}^{(i)}$ is the estimated output for $x^{(i)}$

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

$$= \frac{1}{2m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)} \right)^2$$

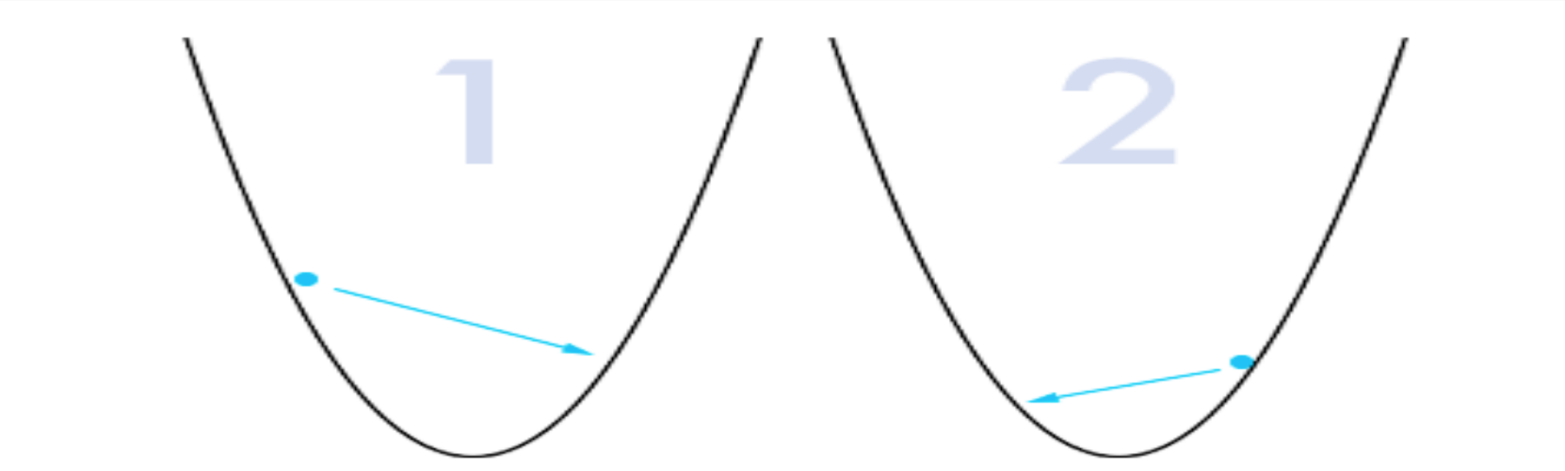
How can we find the 'best' θ?

Gradient descent

$$
\begin{aligned}
\theta_j &:= \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \\
&= \theta_j - \alpha \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta x^{(i)} - y^{(i)} \right)^2 \\
&= \theta_j - \alpha \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)} \right)^2 \\
&= \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)} \right) x_j^{(i)} \\
&= \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta x^{(i)} - y^{(i)} \right) x_j^{(i)}
\end{aligned}
$$

Eg. $\theta_0 = 0$:

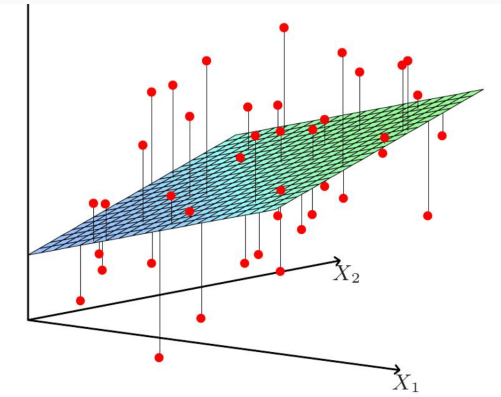
- data: $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$
- parameter: $\theta = \theta_1$
- model: $h_\theta(x^{(i)}) = \theta_1 x^{(i)}$
- **cost**: $J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(\theta_1 x^{(i)} - y^{(i)} \right)^2$
- gradient descent: $\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$

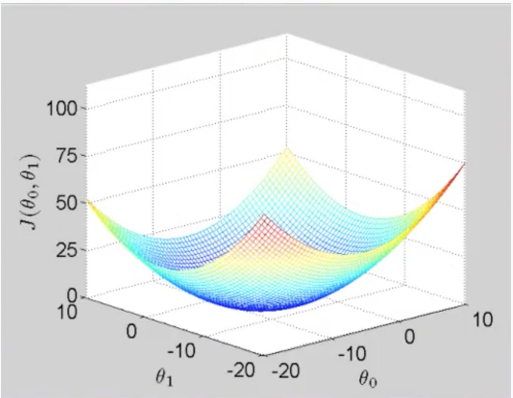


Multivariate linear regression

Size (feet²)	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
...	...	...	...	...

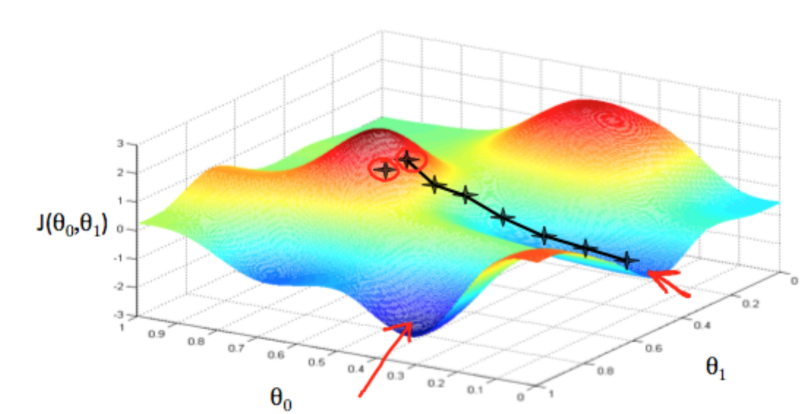
- $x^{(i)} := (x_1^{(i)}, x_2^{(i)}, \ldots, x_d^{(i)})$
- $\theta = (\theta_0, \theta_1, \ldots, \theta_d)$
- $h_\theta x^{(i)} = \theta_0 + \theta_1 x_1^{(i)} + \cdots + \theta_d x_d^{(i)}$
- $J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta x^{(i)} - y^{(i)} \right)^2$
- $\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta x^{(i)} - y^{(i)} \right) x_j^{(i)}$

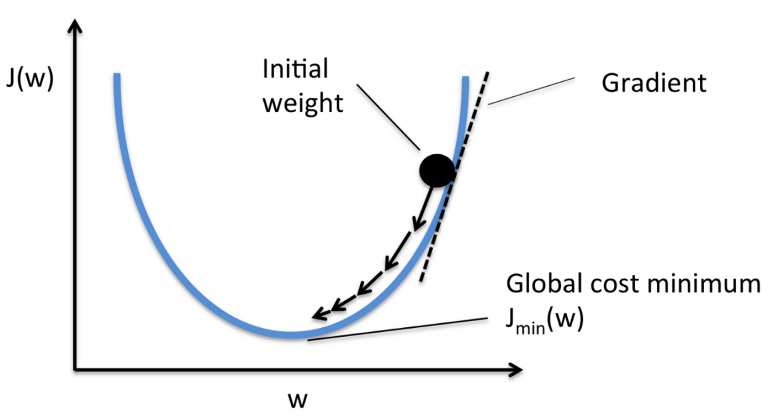




Data preprocessing

- sample size and feature size
- feature scaling
- parameter initialization
- the learning rate





Linear algebra towards vectorizing

Matrix $\times$ vector $\left(X^T y = \sum_{i=1}^{n} y_i x_i\right)$

$$\begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = y_1 \begin{bmatrix} | \\ x_1 \\ | \end{bmatrix} + y_2 \begin{bmatrix} | \\ x_2 \\ | \end{bmatrix} + \cdots + y_n \begin{bmatrix} | \\ x_n \\ | \end{bmatrix}$$

Matrix $\times$ matrix $\left(X^T X = \sum_{i=1}^{n} x_i x_i^T\right)$

$$\begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} -x_1^T - \\ -x_2^T - \\ \vdots \\ -x_n^T - \end{bmatrix} = x_1 x_1^T + \cdots + x_n x_n^T$$

Vector forms

vectorization:

- Training data:

  $(x^{(i)},y^{(i)})$, where $x^{(i)}:=(x_1^{(i)},x_2^{(i)},\ldots,x_d^{(i)})$

- Parameters:

  $\theta = (\theta_0,\theta_1,\ldots,\theta_d)$

- Hypothesis:

  $h_\theta x^{(i)} = \theta_0+\theta_1 x_1^{(i)}+\cdots+\theta_d x_d^{(i)} = \theta x^{(i)}$

- Cost function:

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

- Gradient descent:

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

- Training data (X,y)

$$X = \begin{bmatrix} -\left(x^{(1)}\right)^T- \\ -\left(x^{(2)}\right)^T- \\ \vdots \\ -\left(x^{(m)}\right)^T- \end{bmatrix} \quad y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

- Cost function:

$$J(\theta) = \frac{1}{2m}(X\theta - y)^T(X\theta - y)$$

- Gradient descent

$$\nabla J(\theta) = \frac{1}{m}X^T(X\theta - y)$$

$$\theta := \theta - \alpha\nabla J(\theta)$$

Least squares and maximum likelihood estimate

LS:

- Minimum error:
  $J(\theta) = \frac{1}{2m}(X\theta - y)^T(X\theta - y) = \frac{1}{2m}||y - X\theta||^2$

  $\theta_{LS} = \arg\min_{\theta} J(\theta) = \arg\min_{\theta} ||y - X\theta||^2$

  $\nabla_\theta J = 0 \quad \Rightarrow \quad \nabla_\theta J(\theta) = 2X^T X\theta - 2X^T y = 0 \quad \Rightarrow \quad \theta_{\mathrm{LS}} = \left(X^T X\right)^{-1} X^T y$

MLE

- Maximum estimation:
  If $\mu = X\theta$ and $\Sigma = \sigma^2 I$ is a cov mat, the Gaussian density function is:
  $p\left(y|\mu, \sigma^2\right) = \frac{1}{\left(2\pi\sigma^2\right)^{\frac{n}{2}}} \exp\left(-\frac{1}{2\sigma^2}(y-\mu)^T(y-\mu)\right)$

  $$\theta_{\mathrm{ML}} = \arg\max_{\theta} p\left(y|\mu = X\theta, \sigma^2\right) = \arg\max_{\theta} \ln p\left(y|\mu = X\theta, \sigma^2\right)$$
  $$= \arg\max_{\theta} -\frac{1}{2\sigma^2}\|y - X\theta\|^2 - \frac{n}{2}\ln\left(2\pi\sigma^2\right) = \arg\max_{\theta} -\|y - X\theta\|^2$$

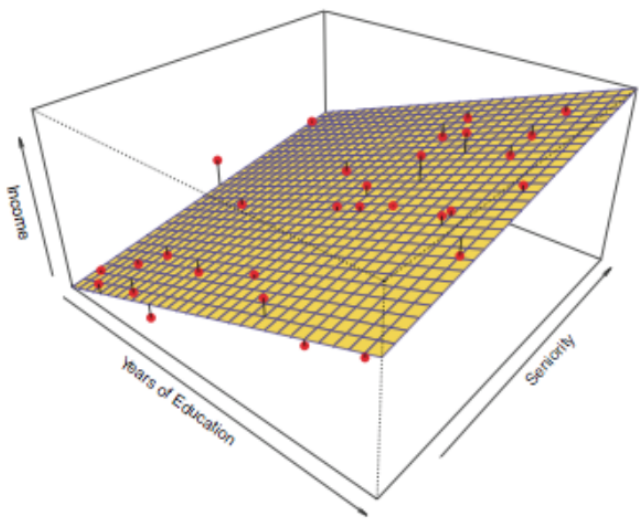
Extending to polynomial regression

Example: 2nd and 3rd order polynomial regression in $\mathbb{R}^2$

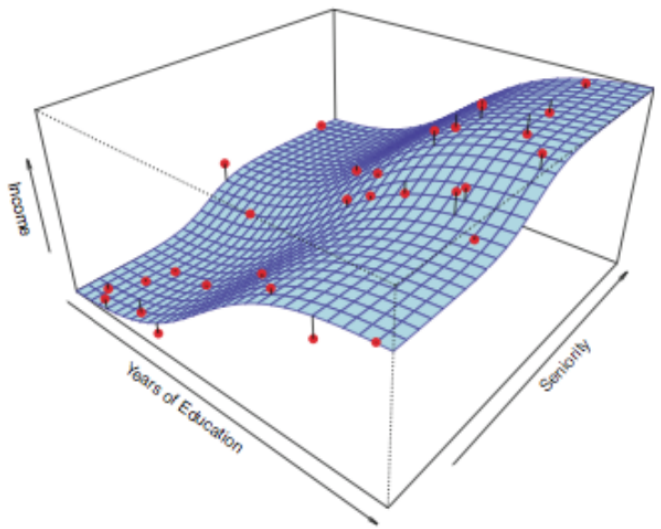
The width of X grows as (order) $\times$ (dimensions) + 1.

$$\text{2nd order:} \quad y_i = w_0 + w_1 x_{i1} + w_2 x_{i2} + w_3 x_{i1}^2 + w_4 x_{i2}^2$$

$$\text{3rd order:} \quad y_i = w_0 + w_1 x_{i1} + w_2 x_{i2} + w_3 x_{i1}^2 + w_4 x_{i2}^2 + w_5 x_{i1}^3 + w_6 x_{i2}^3$$



(a) 1st order

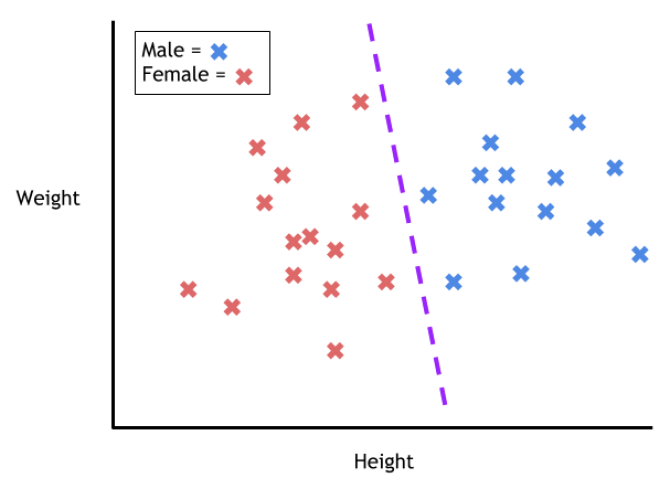


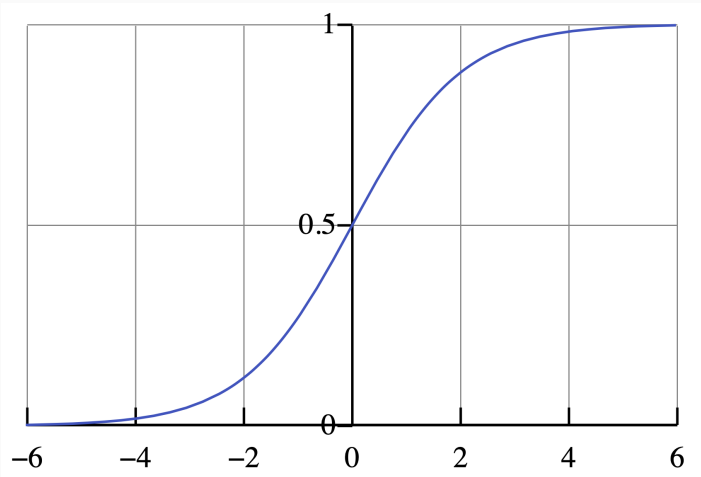
(b) 3rd order

Classification

Classification

- Data: $(x^{(i)}, y^{(i)})$ where $y^{(i)}$ is either 1 or 0 (for binary).
  Eg. labeling as yes/no, normal/abnormal, 0/1, present/absent, etc.
- Decision boundary in contrast to a fitting curve
- $y^{(i)}$ is either 0 or 1. Thus, $0 \leq h_\theta(x^{(i)}) \leq 1$, for each i
- logistic function $h_\theta(x) = (1 + exp(-\theta x))^{-1}$
- h used as estimate probability, $p(y^{(i)} = 1 | x^{(i)}; \theta)$
  eg. $h = .8$ *. . . prob. of raining, etc*

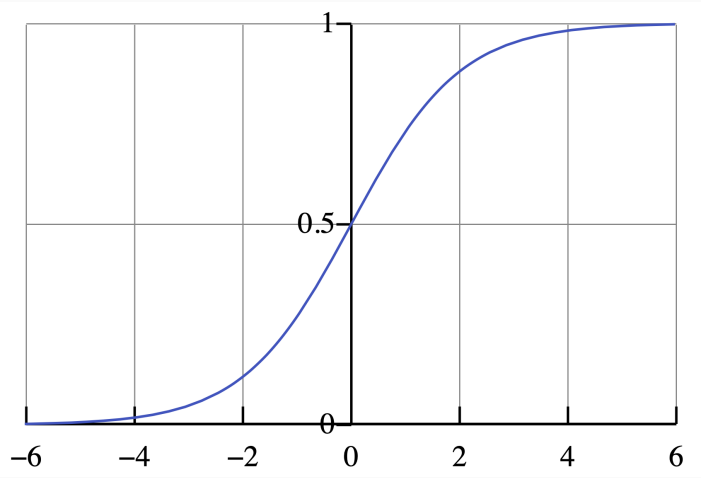


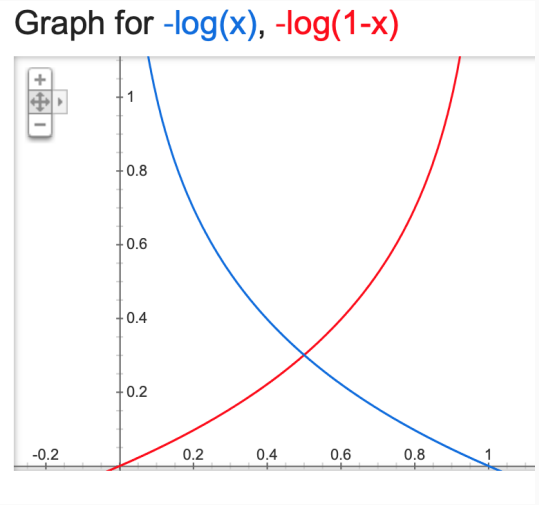


Logistic regression

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

  $\text{cost} = -\log\left(h_\theta(x)\right)$ if $y = 1$ and $-\log\left(1 - h_\theta(x)\right)$ if $y=0$.





$\text{cost}\left(h_\theta(x), y\right) = 0$ if $h_\theta(x) = y$ (correct prediction)
$\text{cost}\left(h_\theta(x), y\right) \to \infty$ if $y = 0$ and $h_\theta(x) \to 1$
$\text{cost}\left(h_\theta(x), y\right) \to \infty$ if $y = 1$ and $h_\theta(x) \to 0$

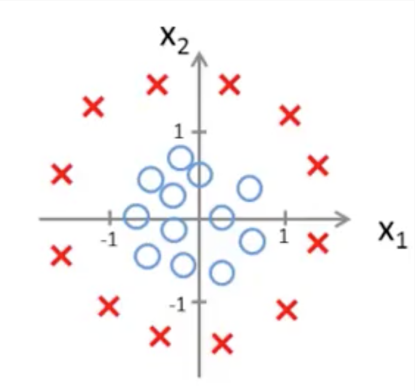
Gradient descent to get a good separating boundary.

Extention

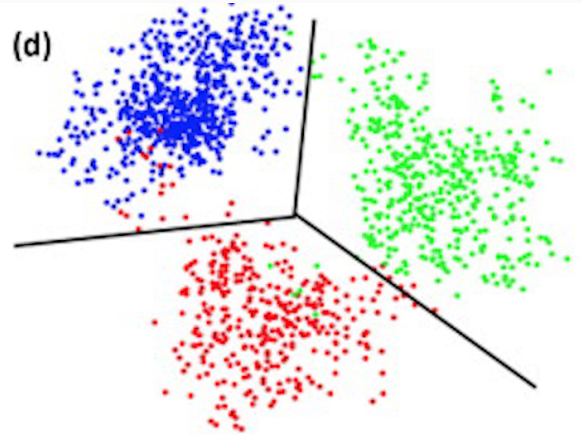
⁕ **Beyond a line boundary:**

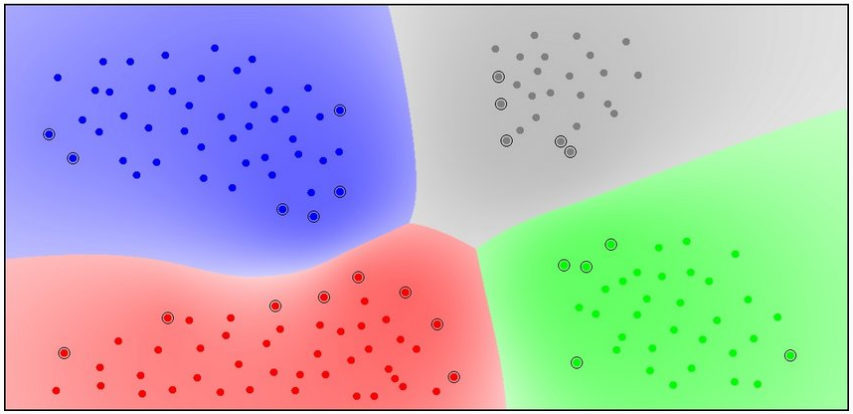
$$h_\theta(x) = g\left(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2\right)$$

Setting $\theta = (-1, 0, 0, 1, 1)$, for example would do.

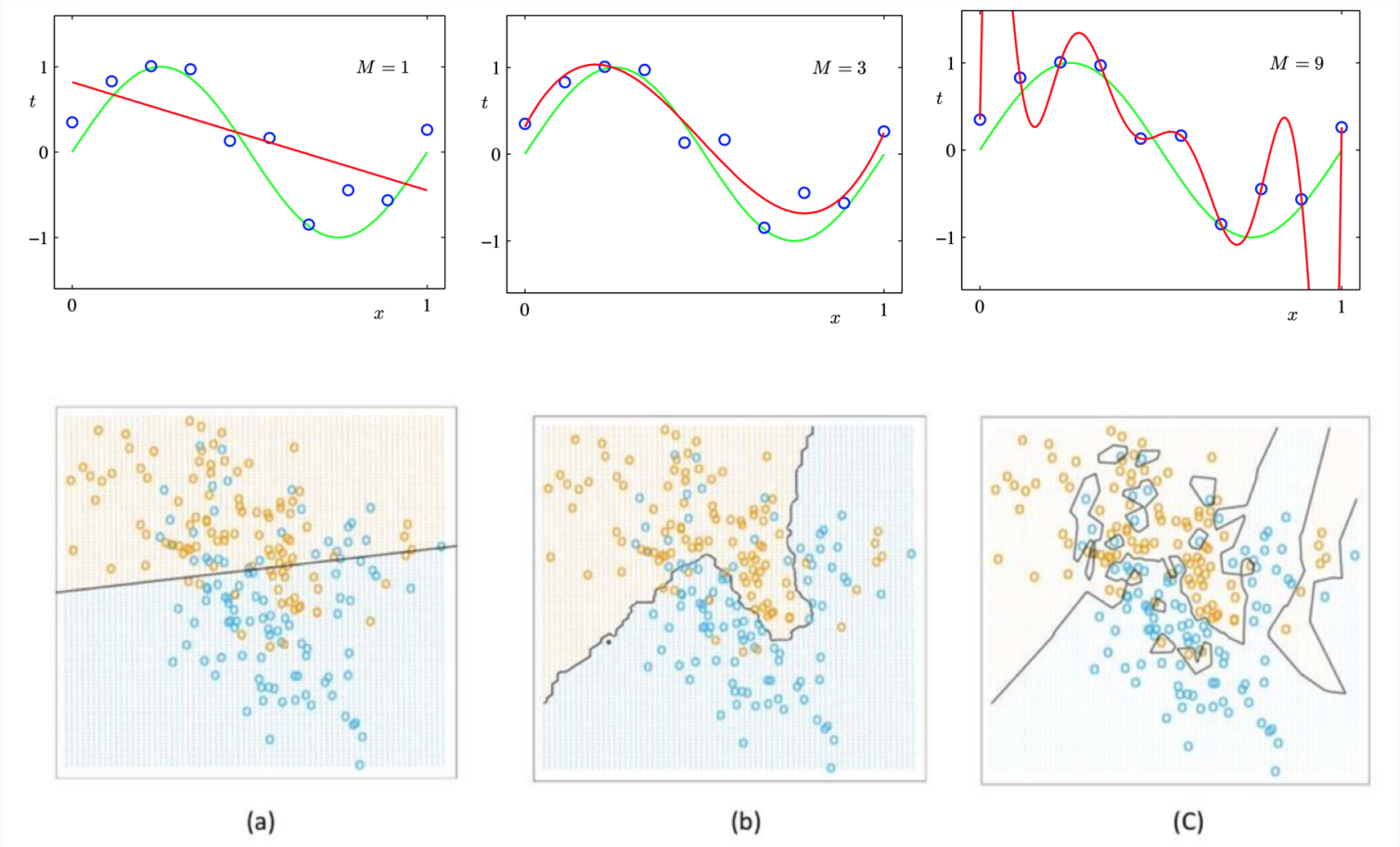


⁕ **Beyond binary class classification:**





Bias and variance trade-off



Regularization:

$\theta_{\mathrm{R}} = \arg\min_\theta \|y - X\theta\|^2 + \lambda g(\theta)$

eg. 1. LASSO: $g(\theta) = \|\theta\|_1$

eg. 2. Tikhonov: $g(\theta) = \|\theta\|_2^2$

$\nabla_\theta L = (y - X\theta)^T (y - X\theta) + \lambda \theta^T \theta = 0$

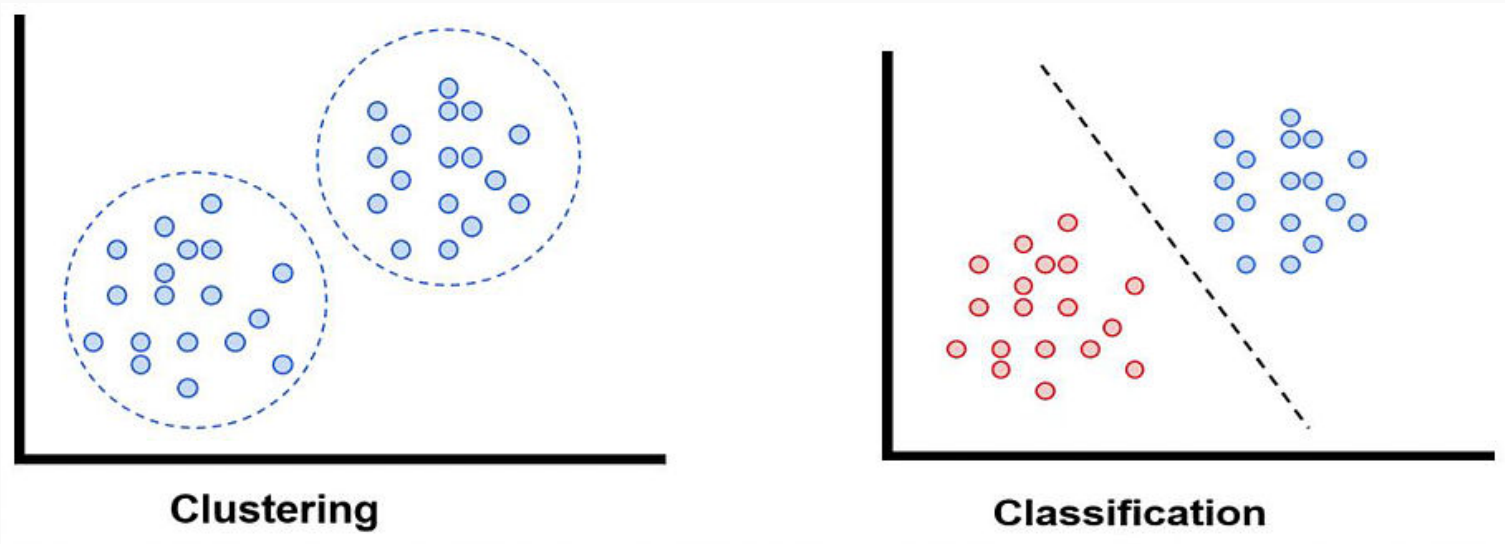
$\Leftrightarrow \quad -2X^T y + 2X^T X\theta + 2\lambda\theta = 0$

$\Leftrightarrow \quad \theta_{RR} = \left(\lambda I + X^T X\right)^{-1} X^T y$

Clustering

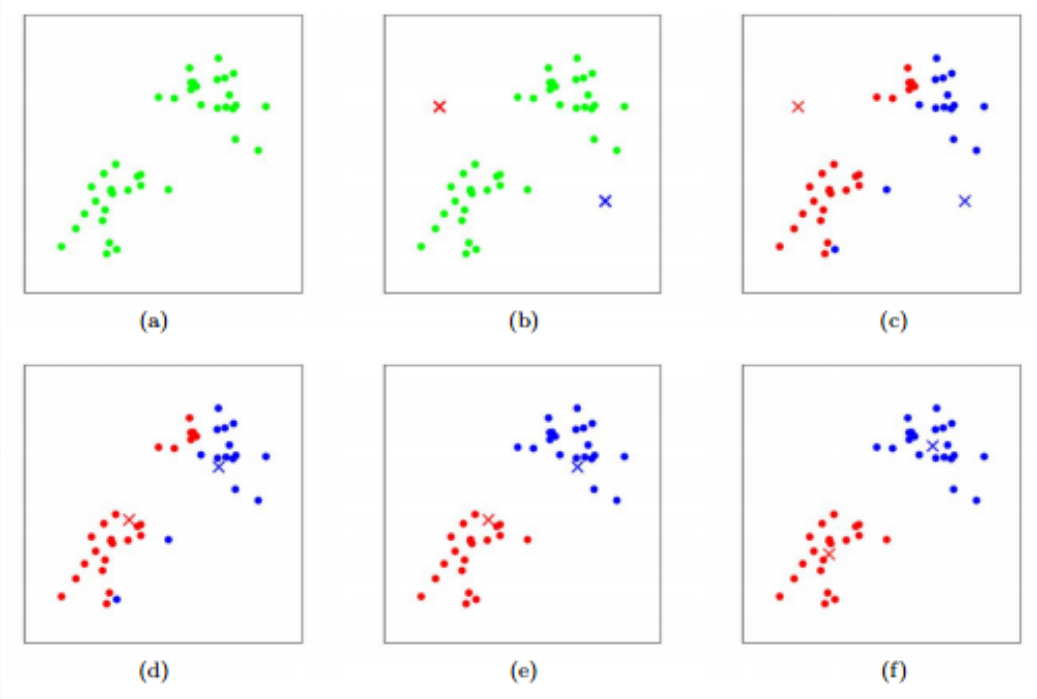
Unsupervised learning – Clustering algorithm

- pattern extraction and grouping
- data points are not labeled. i.e. $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$
- market segmentation, recommender system, gene expression analysis
- common methods: k-means and hierarchical clustering

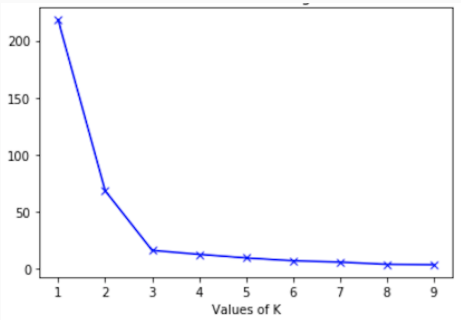


k-means clustering algorithm

- choose k and label each cluster $c^{(i)}$.
- choose k initial centroids, $\mu_{c^{(i)}}$.
- assign membership of each $x^{(i)}$ to a centroid.
- update centroids as the mean of group datasets.
- repeat until satisfied with the grouping.

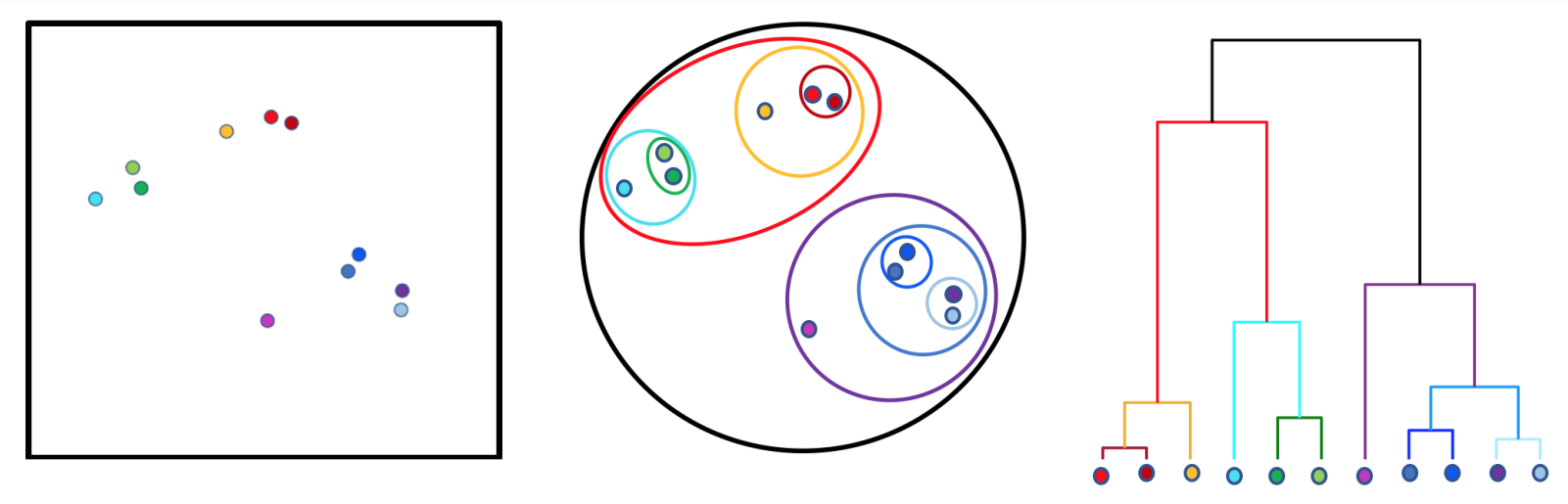


- Cost function: $J\left(\mu_{c^{(1)}}, \ldots, \mu_{c^{(k)}}\right) = \frac{1}{m} \sum_{i=1}^{m} \left\| x^{(i)} - \mu_{c^{(i)}} \right\|^2$
- k-means can get stuck in local minima.
- choice of k



Hierarchical clustering algorithm

- start by considering each data point as a cluster.
- create hierarchical clusters by nearest distance rule.
- repeat until the giant cluster is formed.



◦ Distance between clusters:

- $L(r,s) = \min\left(D\left(x_{r,i}, x_{s,j}\right)\right)$
- $L(r,s) = \max\left(D\left(x_{r,i}, x_{s,j}\right)\right)$
- $L(r,s) = mean\left(D\left(x_{r,i}, x_{s,j}\right)\right)$