Expectation Maximisation (EM)

CS 486/686: Introduction to Artificial Intelligence
University of Waterloo

Overview

- Learning from incomplete data
  - EM algorithm
- Unsupervised Learning

Incomplete Data

- So far we have seen problems where
  - Values of all attributes are known
  - Learning is relatively easy

- Many real-world problems have **hidden variables**
  - Incomplete data
  - Missing attribute values

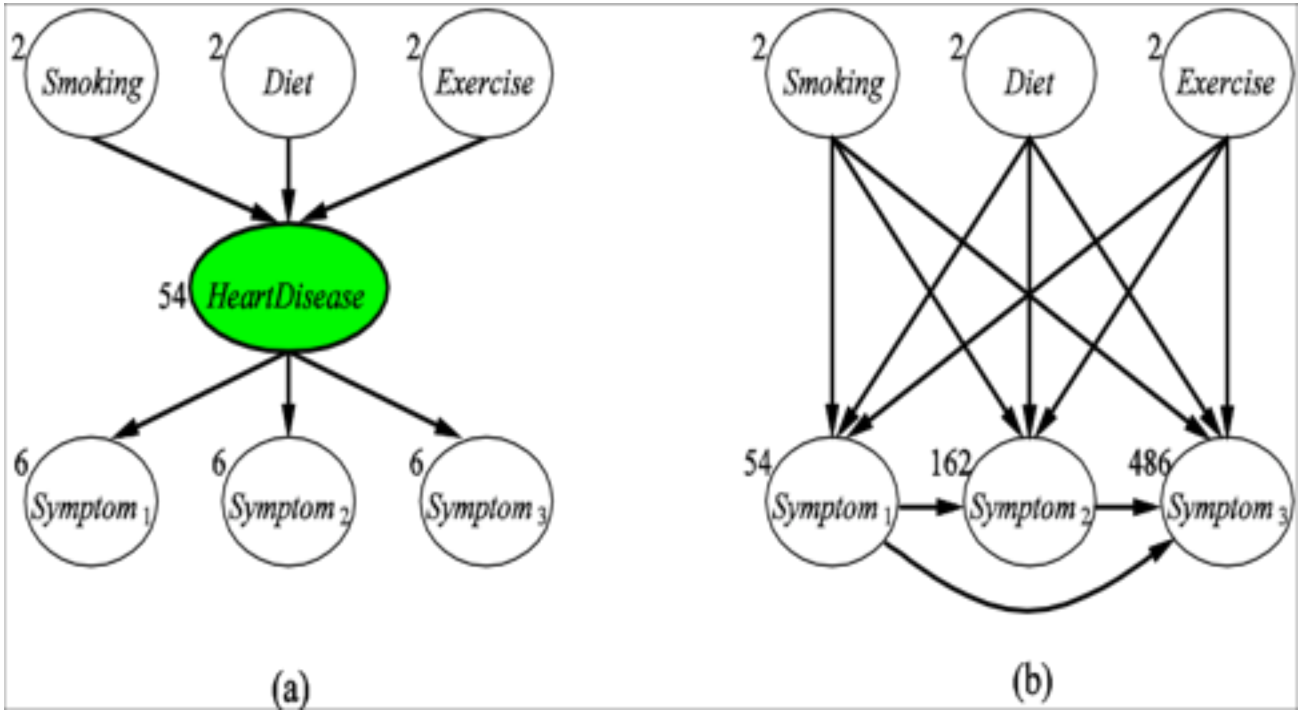
Bayes Nets: Maximum Likelihood Learning

- Review: ML Learning of Bayes nets parameters
  - $\Theta_{V=true,\ Par(V)=x} = P(V=true|Par(V)=x)$
  - $\Theta_{V=true,\ Par(V)=x}$ (#Insts V=true)/(Total #V)
  - Assumes all attributes have values
    - What if some values are missing?

Naïve Solutions

- Ignore examples with missing attribute values
  - What if all examples have missing attribute values?
- Ignore hidden variables
  - Model might become much more complex

Hidden Variables
Heart disease example



a) Uses a Hidden Variable, simpler (fewer CPT parameters)
b) No Hidden Variable, complex (many CPT parameters)

“Direct” ML

- Maximize likelihood directly
  - Let Z be hidden vars, and E observable
  - h_{ML}=argmax$_h$P(e|h)

    =argmax$_h$Σ$_z$P(e,z|h)

    =argmax$_h$Σ$_z$∏$_i$CPT(Vi)

    =argmax$_h$ log Σ$_z$ ∏$_i$ CPT(Vi)

– Can’t push log past sum to linearize product

Expectation-Maximization (EM)

- If we knew the missing values computing h_{ML} is trivial
- Guess h_{ML}
- Iterate
  - **Expectation**: based on h_{ML} compute expectation of missing values
  - **Maximization**: based on expected missing values compute new h_{ML}

Expectation-Maximization (EM)

- Formally
  - Approximate maximum likelihood
  - Iteratively compute:
  - $\underbrace{h_{i+1}=\mathrm{argmax}_h \underbrace{\Sigma_Z P(Z|h_i,e)\log P(e,Z|h)}_{\textbf{Expectation}}}_{\textbf{Maximization}}$

EM Derivation

- Derivation
  - $\log P(e|h) = \log[P(e,Z|h)/P(Z|e,h)]$

$$= \log P(e,Z|h) - \log P(Z|e,h)$$

$$= \Sigma_Z P(Z|e,h) \log P(e,Z|h) - \Sigma_Z P(Z|e,h) \log P(Z|e,h)$$

$$\geq \Sigma_Z P(Z|e,h) \log P(e,Z|h)$$

- EM finds a **local maximum** of $\Sigma_Z P(Z|e,h) \log P(e,Z|h)$ which is a **lower bound** of $\log P(e|h)$

EM

- Log inside sum can linearize the product
  - $h_{i+1} = \text{argmax}_h \Sigma_Z P(Z|h,e) \log P(e,Z|h)$

    $= \text{argmax}_h \Sigma_Z P(Z|h,e) \log \Pi_j CPT_j$

    $= \text{argmax}_h \Sigma_Z P(Z|h,e) \Sigma_j \log CPT_j$

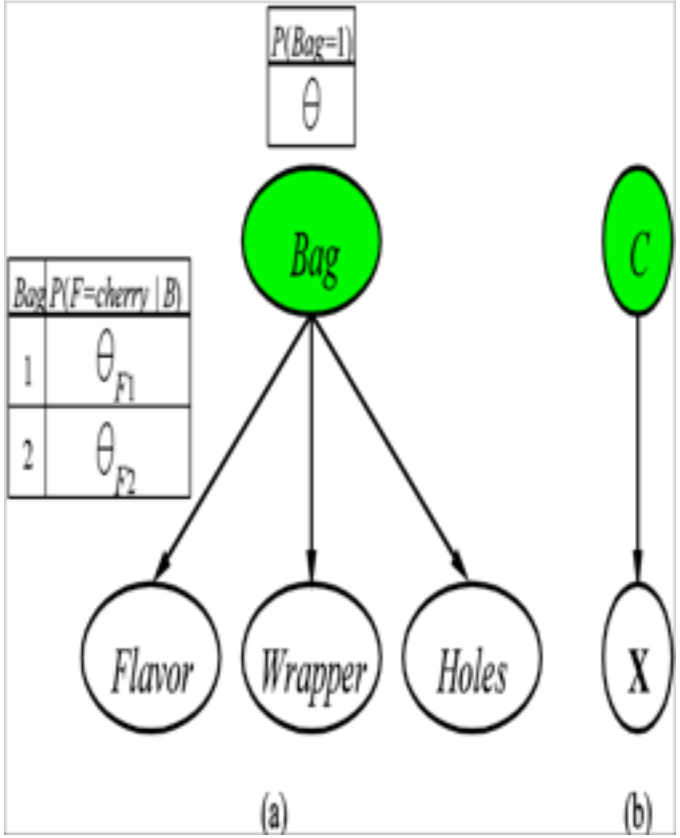
- Monotonic improvement of likelihood
  - $P(e|h_{i+1}) \geq P(e|h_i)$

Candy Example

- You buy two bags of candies of unknown type (flavour ratios)
- You plan to eat candies from each bag to learn the flavour ratios
- Your rotten roommate mixes both bags
- How do you learn the type of each bag despite being mixed?

Unsupervised Clustering

- “Class” variable (Bag) is hidden
- Naïve Bayes model



Candy Example

- Unknown Parameters
  - Θ_i=P(Bag=i)
  - Θ_{Fi}=P(Flavour=cherry|Bag=i)
  - Θ_{Wi}=P(Wrapper=red|Bag=i)
  - Θ_{Hi}=P(Hole=yes|Bag=i)

- When eating a candy:
  - F, W, and H are observable
  - B is hidden

Candy Example

- Let true parameters be:
  - Θ=0.5, Θ_{H1}= Θ_{W1}= Θ_{H1}=0.8,

    and Θ_{F2}= Θ_{W2}= Θ_{H2}=0.3
- After eating 1000 candies

	W=red		W=green	
	H=1	H=0	H=1	H=0
F=cherry	273	93	104	90
F=line	79	100	94	167

EM Algorithm

- Guess h_0
  - Θ=0.6
  - $\Theta_{F1}= \Theta_{W1}= \Theta_{H1}$=0.6
  - $\Theta_{F2}= \Theta_{W2}= \Theta_{H2}$=0.4
- Alternate
  - Expectation: expected # of candies in each bag
  - Maximization: new parameter estimates

Candy Example

- Expectation: expected # of candies in each bag
  - #[Bag=i]=Σ_j P(B=i|f_j,w_j,h_j)
  - Compute P(B=i|f_j,w_j,h_j) by variable elimination

- Example
  - #[Bag=1]=612
  - #[Bag=2]=388

Candy Example

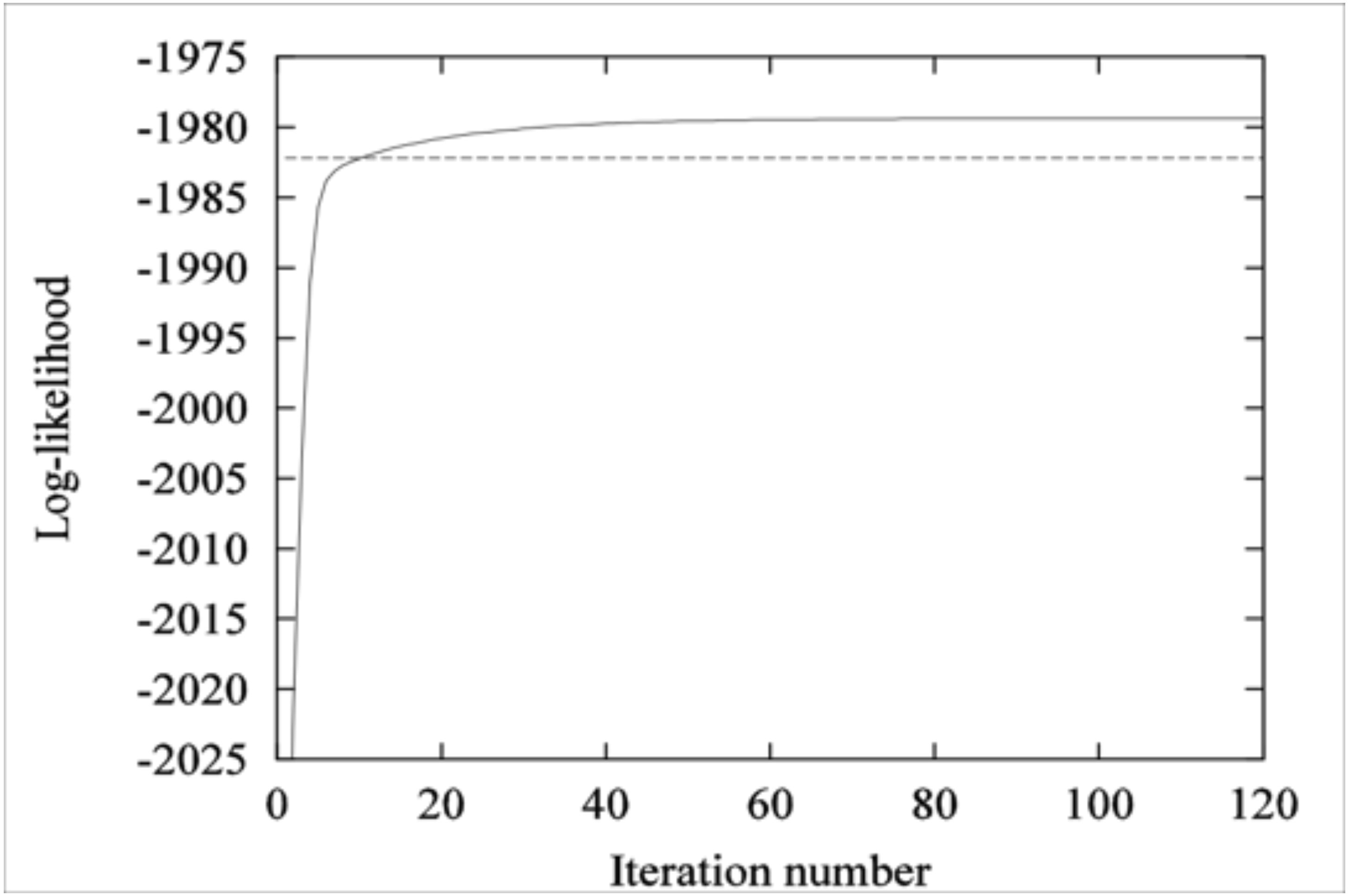
- Maximization: relative frequency of each bag
  - Θ_1=612/1000 = 0.612
  - Θ_2=388/1000=0.388

Candy Example

- Expectation: expected # cherry candies in each bag
  - $\#[B=i,F=cherry]=\Sigma_j P(B=i|f_j=cherry,w_j,h_j)$
  - Compute $P(B=i|f_j=cherry,w_j,h_j)$ by var. elimination

- Maximization:
  - $\Theta_{F1}=\#[B=1,F=cherry]/\#[B=1]=0.668$
  - $\Theta_{F2}=\#[B=2,F=cherry]/\#[B=2]=0.389$

Candy Example



Bayesian Networks

- EM algorithm for general Bayes nets
- Expectation:
  - $\#[V_i=v_{ij},Par(V_i)=pa_{ik}]$=expected frequency
- Maximization
  - $\Theta_{vij,paik}=\#[V_i=v_{ij},Par(V_i)=pa_{ik}]/\#[Par(V_i)=pa_{ik}]$

Unsupervised Learning

- Incomplete data ->Unsupervised learning

- Examples
  - Market segmentation for marketing
  - Categorizing stars by astronomers
  - Identification of species
  - Etc…

Clustering/Unsupervised Learning

- Target features are not given in the training examples
- **Goal**: construct a natural classification that can be used to predict features in the data
- Examples are partitioned into clusters or classes
  - Best clustering minimizes error
- Types of clustering
  - Hard clustering
  - Soft clustering

k-means Algorithm

- *k-means algorithm* is used for hard clustering
- Inputs
  - training examples
  - number of classes, k
- Outputs
  - a prediction of a value for each feature for each class
  - an assignment of examples to classes

k-means Algorithm

- Input:
  - E is set of all examples
  - Input features $X_1,...,X_n$
  - $val(e,X_j)$ is value of feature j for example e
  - k classes {1,2,...,k}
- k-means algorithm outputs
  - function class: E->{1,...,k} where class(e)=i means example e is in class i
  - pval function where $pval(i,X_j)$ is the prediction for each example in class i for feature X_j

k-means Algorithm

- Sum-of-squares error for *class i* and *pval* is

$$\sum_{e \in E} \sum_{j=1}^{n} (\mathrm{pval}(\mathrm{class}(e), X_j) - \mathrm{val}(e, X_j))^2$$

- Goal: Final class and pval that minimizes sum-of-squares error.

Minimizing the error

$$\sum_{e \in E} \sum_{j=1}^{n} (\text{pval}(\text{class}(e), X_j) - \text{val}(e, X_j))^2$$

- Given *class*, the *pval* that minimizes sum-of-square error is the mean value for that class
- Given *pval*, each example can be assigned to the class that minimizes the error for that example

k-means Algorithm

- Randomly assign the examples to classes
- Repeat the following two steps until E step does not change the assignment of any example
  - **M**: For each class i and feature Xj

$$\mathrm{pval}(i, X_j) = \frac{\sum_{e:\mathrm{class}(e)=i} \mathrm{val}(e, X_j)}{|\{e : \mathrm{class}(e) = i\}|}$$

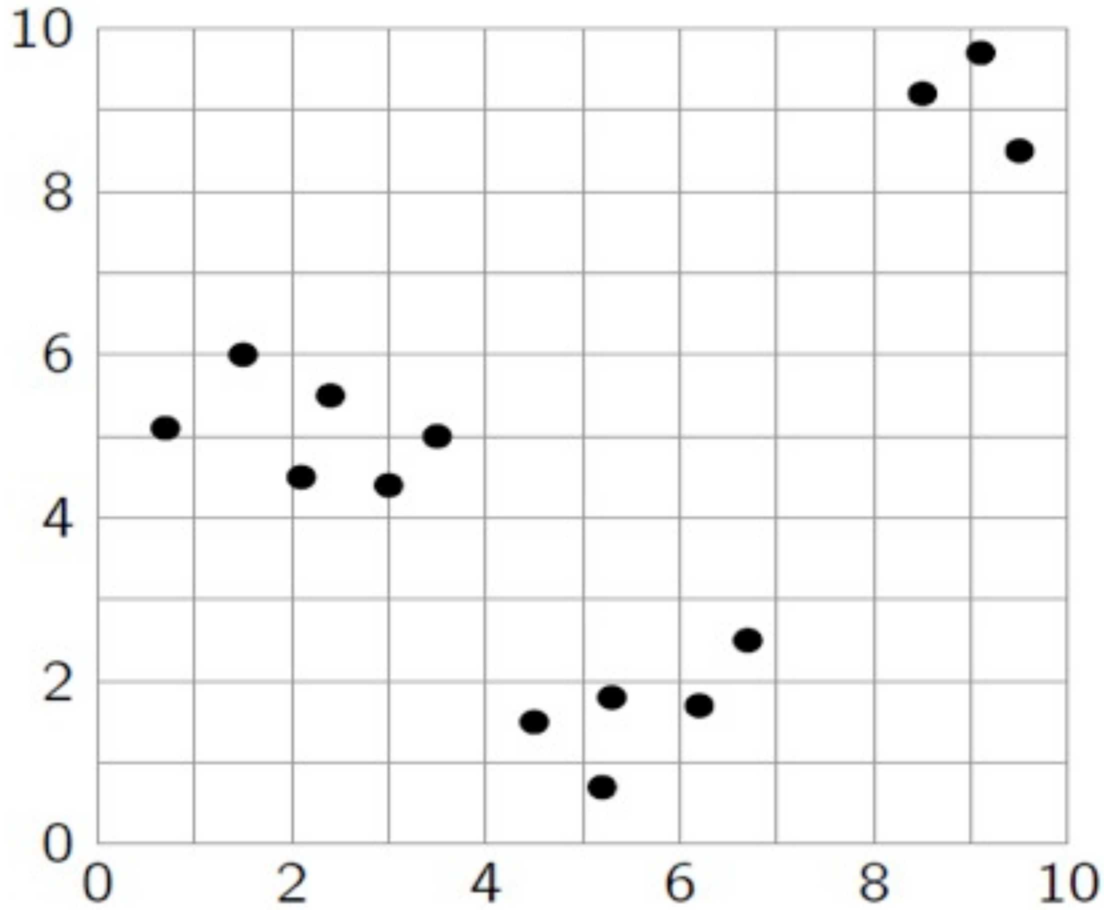
  - **E**: For each example e, assign e to the class that minimizes

$$\sum_{j=1}^{n} (\mathrm{pval}(\mathrm{class}(e), X_j) - \mathrm{val}(e, X_j))^2$$

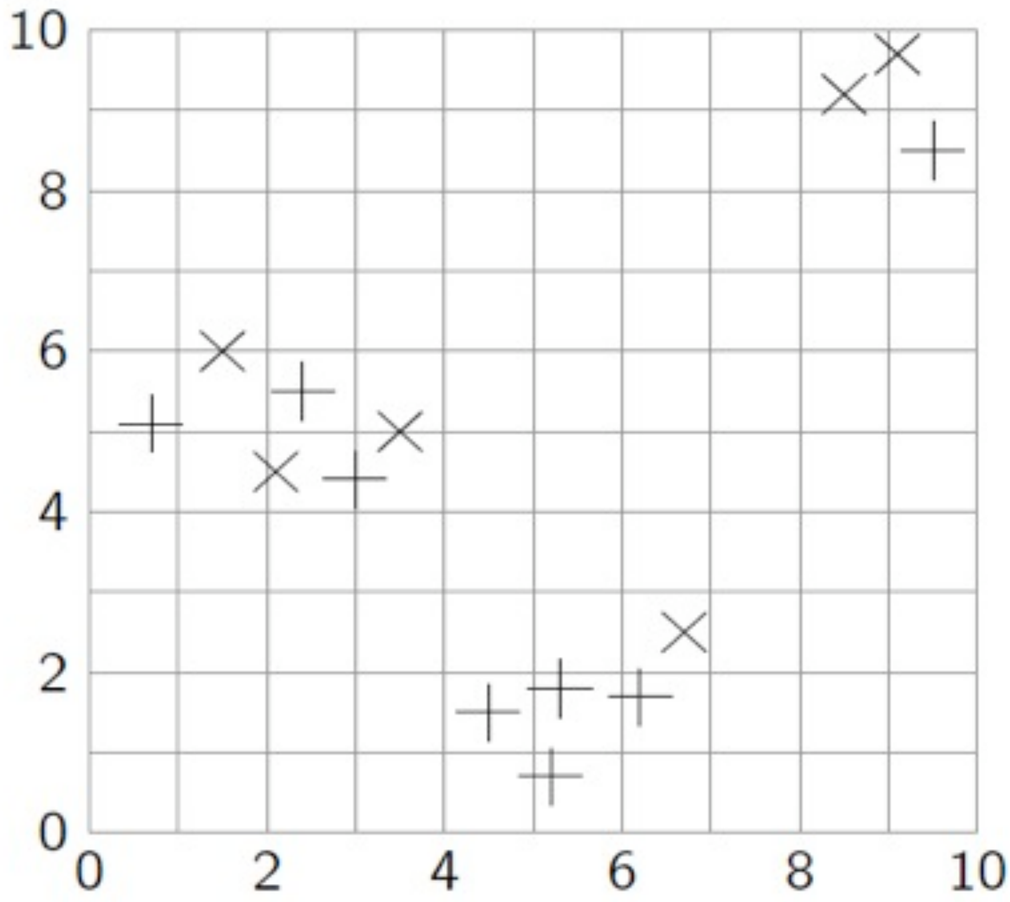
k-means Example

- Data set: (X,Y) pairs
  - (0.7,5.1) (1.5,6), (2.1, 4.5), (2.4, 5.5), (3, 4.4), (3.5, 5), (4.5, 1.5), (5.2, 0.7), (5.3, 1.8), (6.2, 1.7), (6.7, 2.5), (8.5, 9.2), (9.1, 9.7), (9.5, 8.5)

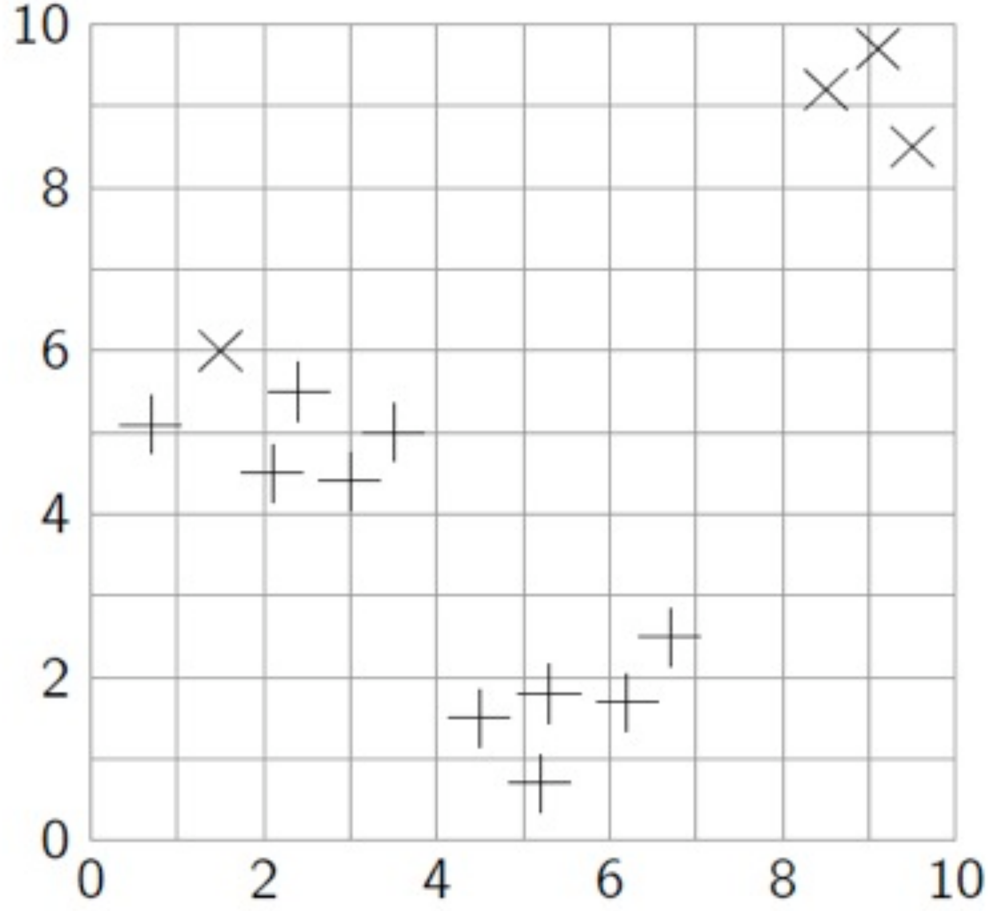
Example Data



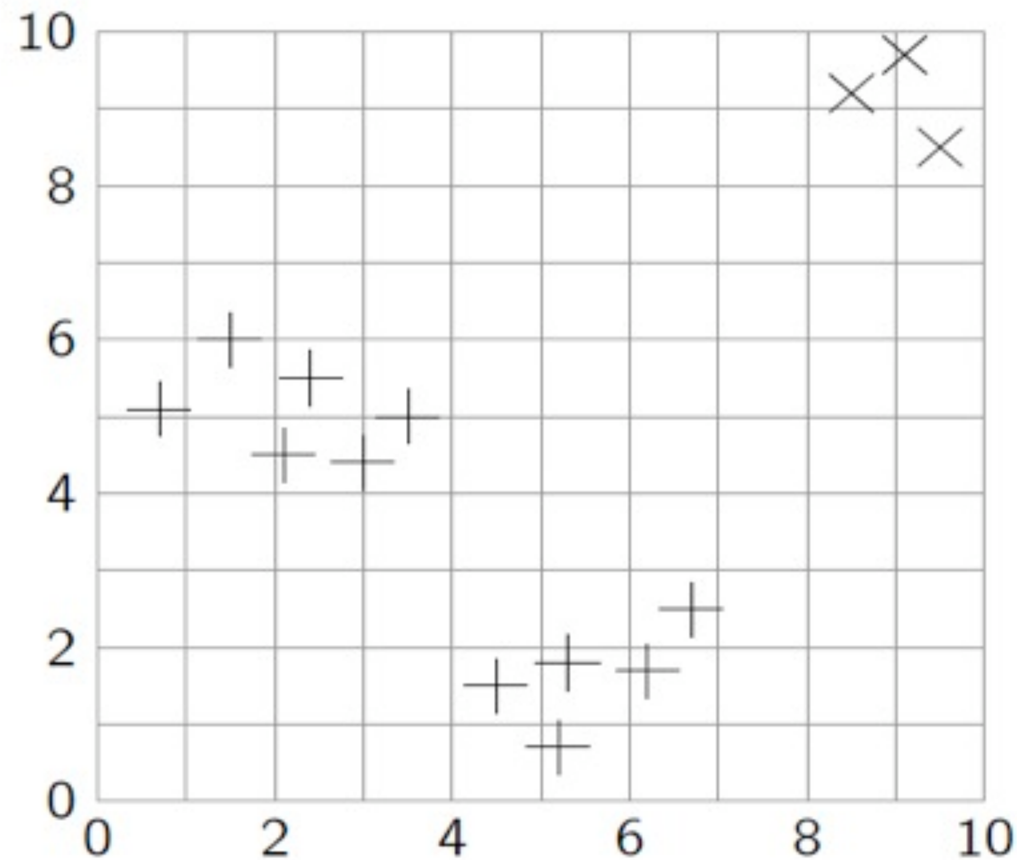
Random Assignment to Classes



Assign Each Example to Closest Mean



Reassign each example



Properties of k-means

- An assignment is *stable* if both M step and E step do not change the assignment
  - Algorithm will eventually converge to a stable local minimum
  - No guarantee that it will converge to a global minimum
- Increasing k can always decrease error until k is the number of different examples