CSC3022H:

Machine Learning: Introduction

Geoff Nitschke

Department of Computer Science
University of Cape Town, South Africa

Course Syllabus

- **Introduction:** Basic concepts.

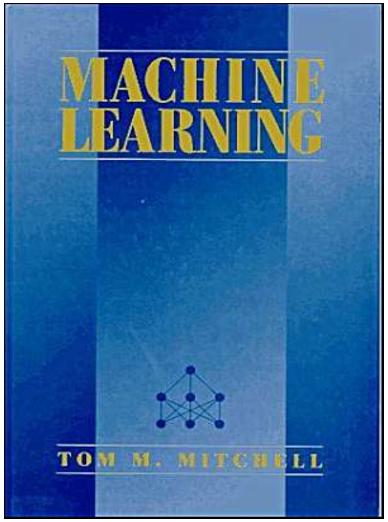
- **Supervised Learning:**
  - ANNs. Back propagation.
  - Generative Learning algorithms. Naïve Bayes.
  - ***Concept Learning.***

- **Unsupervised Learning:**
  - Clustering, Hierarchical clustering, ***K-means***, EC, NE.
  - PCA, ICA, SOM, ART.

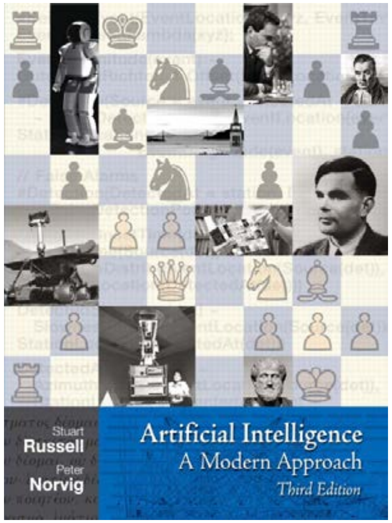
- **Reinforcement Learning:**
  - Q-learning. Policy and Value function approximation.

Module Overview

- Assigned text:
  - Mitchell, T. (1997). Machine Learning, McGraw Hill:
  - http://www.cs.cmu.edu/~tom/mlbook.html

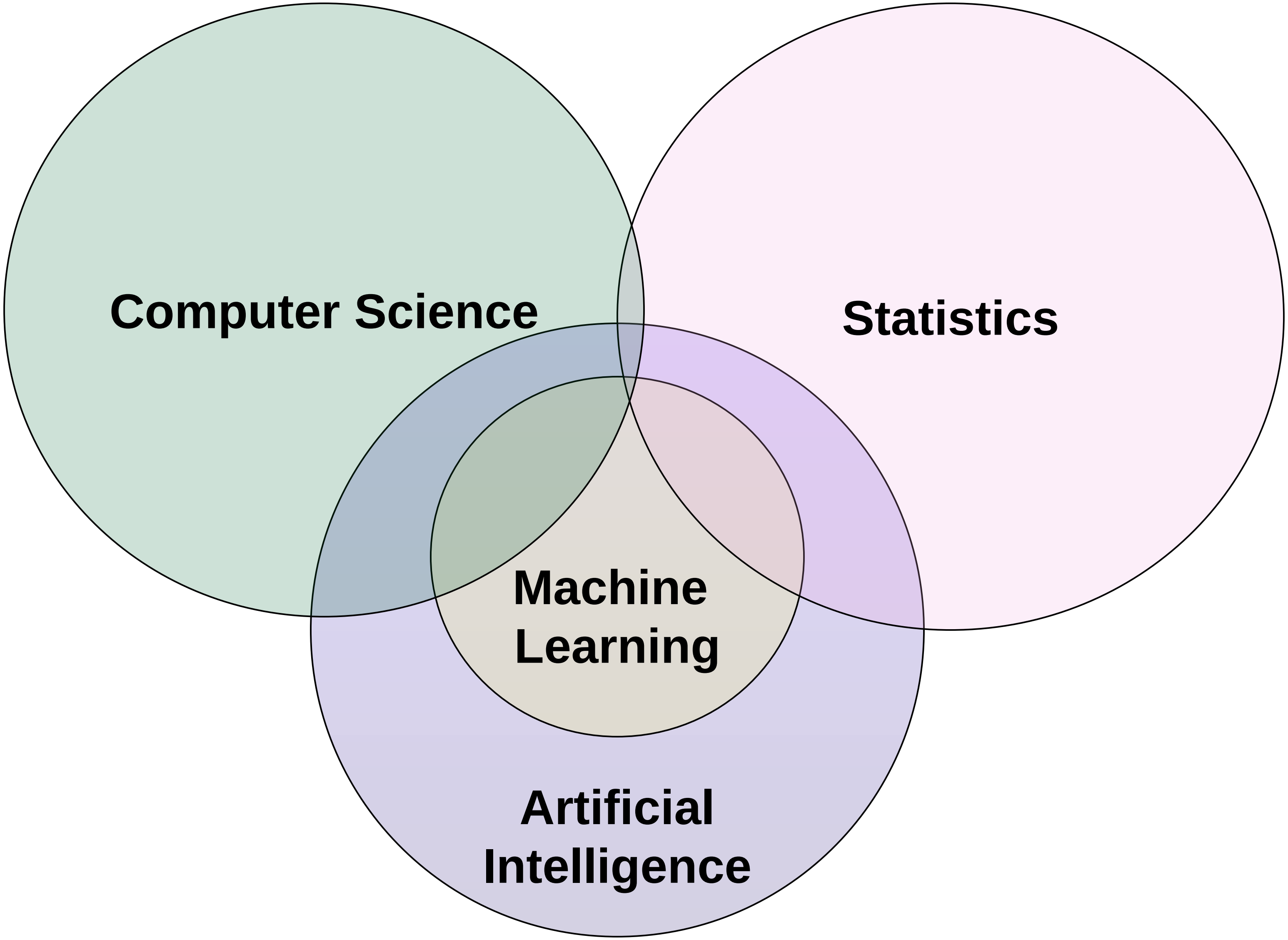


- Recommended text:
  - Russell, S., and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach* – Third Edition. Prentice Hall: http://aima.cs.berkeley.edu/



- ML Labs:
  - TA in Senior Lab every Friday 10.00 – 11.00 am.
- Programming Assignments:
  - 4 weekly tuts + 2 part assignment.
- For the rest: See course outline.

Where Does ML Fit In?



Examples of Machine Learning Types

- Supervised Learning:
  - Classification.
  - Regression.
- **Unsupervised Learning:**
  - **Clustering.**
  - Dimensionality reduction.
- Reinforcement Learning:
  - Value and policy iteration.
  - Q Learning.

Unsupervised Learning

- In supervised learning, data is in the form: ***< x, y >***, where ***y = f (x).*** Goal is to approximate ***f*** well.

- Unsupervised learning: data just contains ***x***.

- Goal is to “summarize" or find “patterns" or “structure" in the data.

  - ***Clustering*** (e.g. partitioning, hierarchical clustering).

  - *Density estimation*.

  - *Dimensionality reduction* (e.g. visualization, compression, pre-processing).

- Often used in data analysis, pre-processing for supervised learning.

What is Clustering?

- Clustering is grouping similar objects together.
  - To classify, or detect outliers.
  - To simplify data for further analysis or learning.
  - To visualize data (dimensionality reduction).

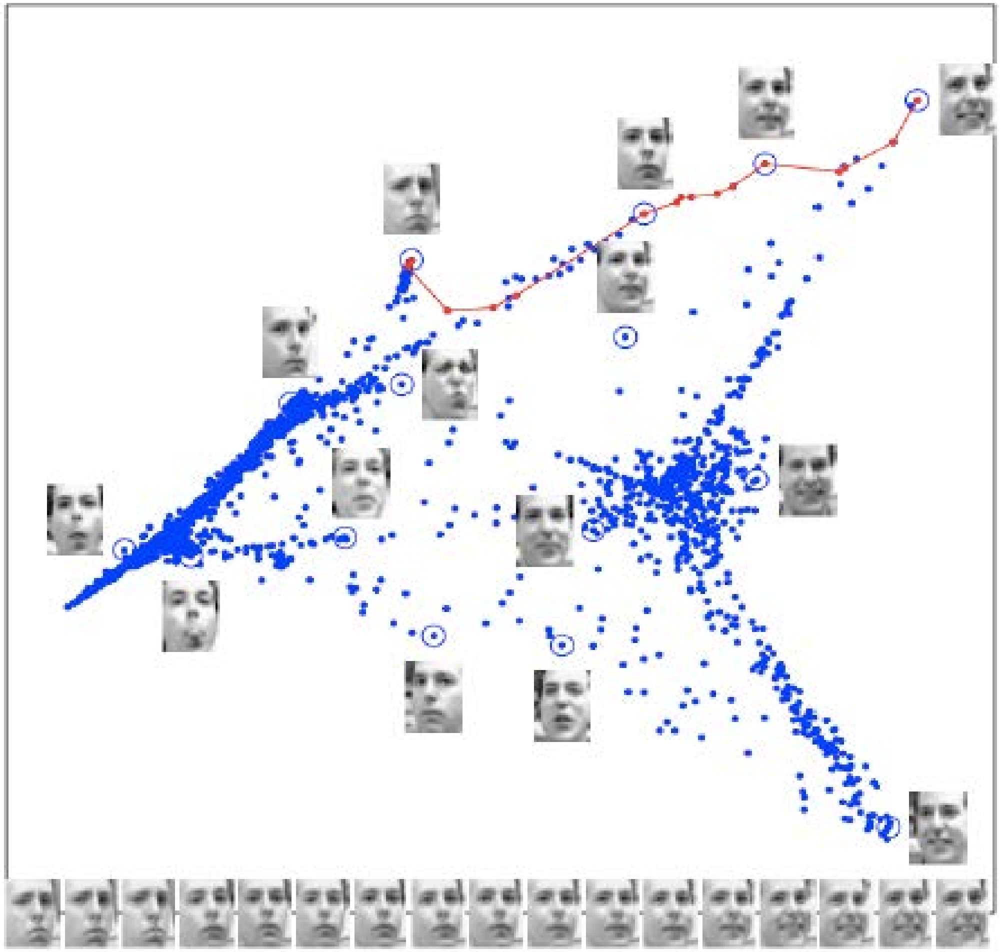
- Clustering is usually not “right" or “wrong”. Different clustering criteria can reveal different things about the data.

- Clustering algorithms:
  - Use some notion of distance between objects.
  - Explicit or implicit criterion defining what a good cluster is.
  - Heuristically optimise criterion to get good clusters.

Example: Clustering Faces

Face data-base:

- Images have thousands or millions of pixels.
- What to cluster?
- Similarity metric?



Clustering

- **K-means clustering**.

- Hierarchical clustering:
  - Agglomerative.
  - Divisive.

- Dimensionality reduction:
  - Principal Component Analysis (PCA).
  - Independent Component Analysis (ICA).

- Self-Organising Maps (SOM).

K-means Clustering

- One of the most commonly-used clustering algorithms.
- Easy to implement and quick to run.
- Assumes objects (instances) to be clustered are **n-**dimensional vectors, $\mathbf{x_i}$ (real valued data).
- Uses a similarity distance metric (e.g. Euclidian distance).
- Goal: Partition the data into K disjoint subsets.

K-means Clustering

- Inputs:
  - A set of n-dimensional real vectors $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$.
  - K, the desired number of clusters.
- Output: A mapping of the vectors into K clusters (disjoint subsets), $C : \{1, \ldots, m\} \mapsto \{1, \ldots, K\}$.

1. Initialize C randomly.
2. Repeat
   (a) Compute the *centroid* of each cluster (the mean of all the instances in the cluster)
   (b) Reassign each instance to the cluster with closest centroid

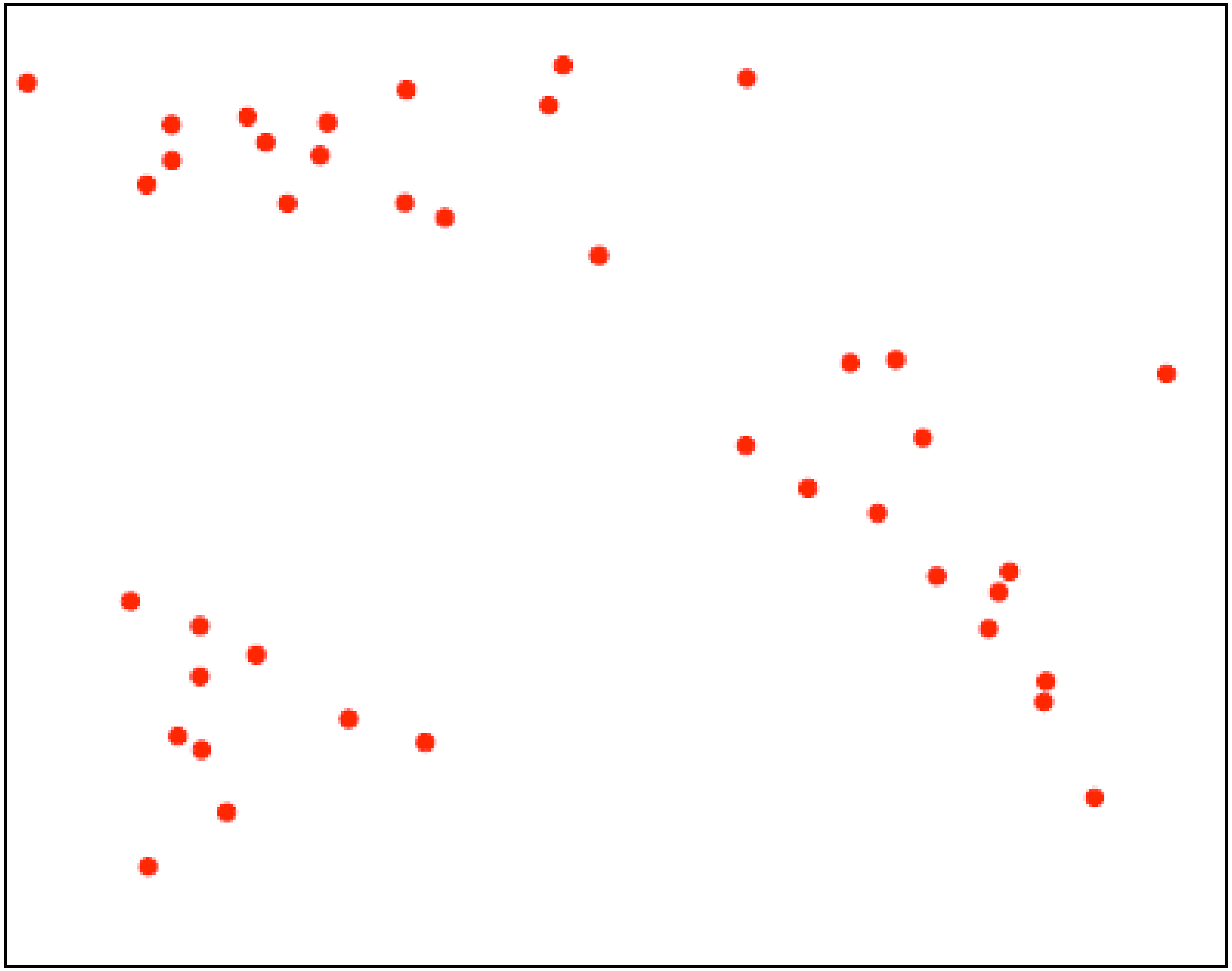
   until C stops changing.

- For given data $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ and a clustering $\mathbf{C}$, consider the sum of the squared Euclidian distance between each vector and the center of its cluster:

$$J = \sum_{i=1}^{m} \|\mathbf{x}_i - \mu_{C(i)}\|^2 ,$$

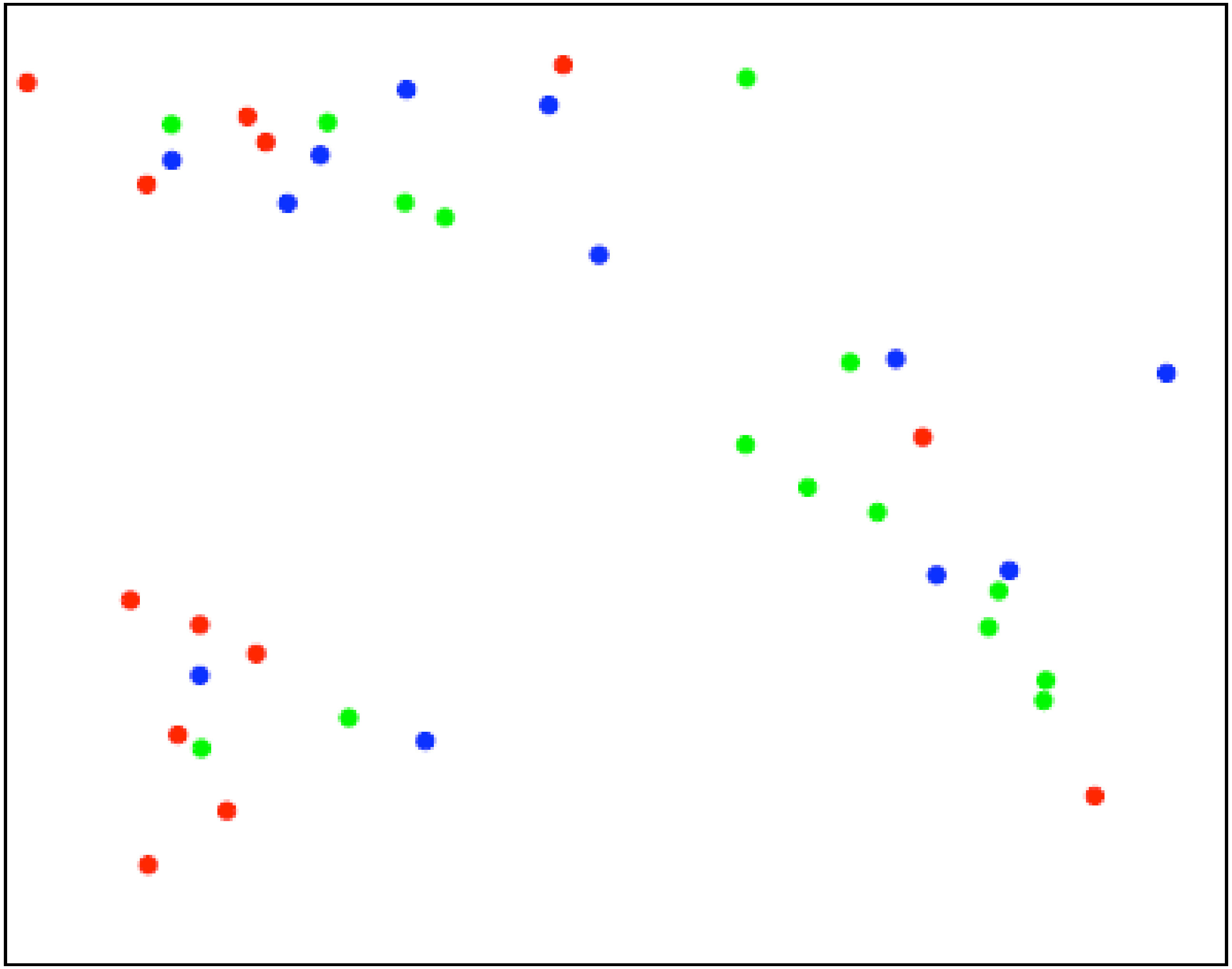
where $\mu_{C(i)}$ denotes the centroid of the cluster containing $\mathbf{x}_i$.

K-means Clustering Example (k=3)



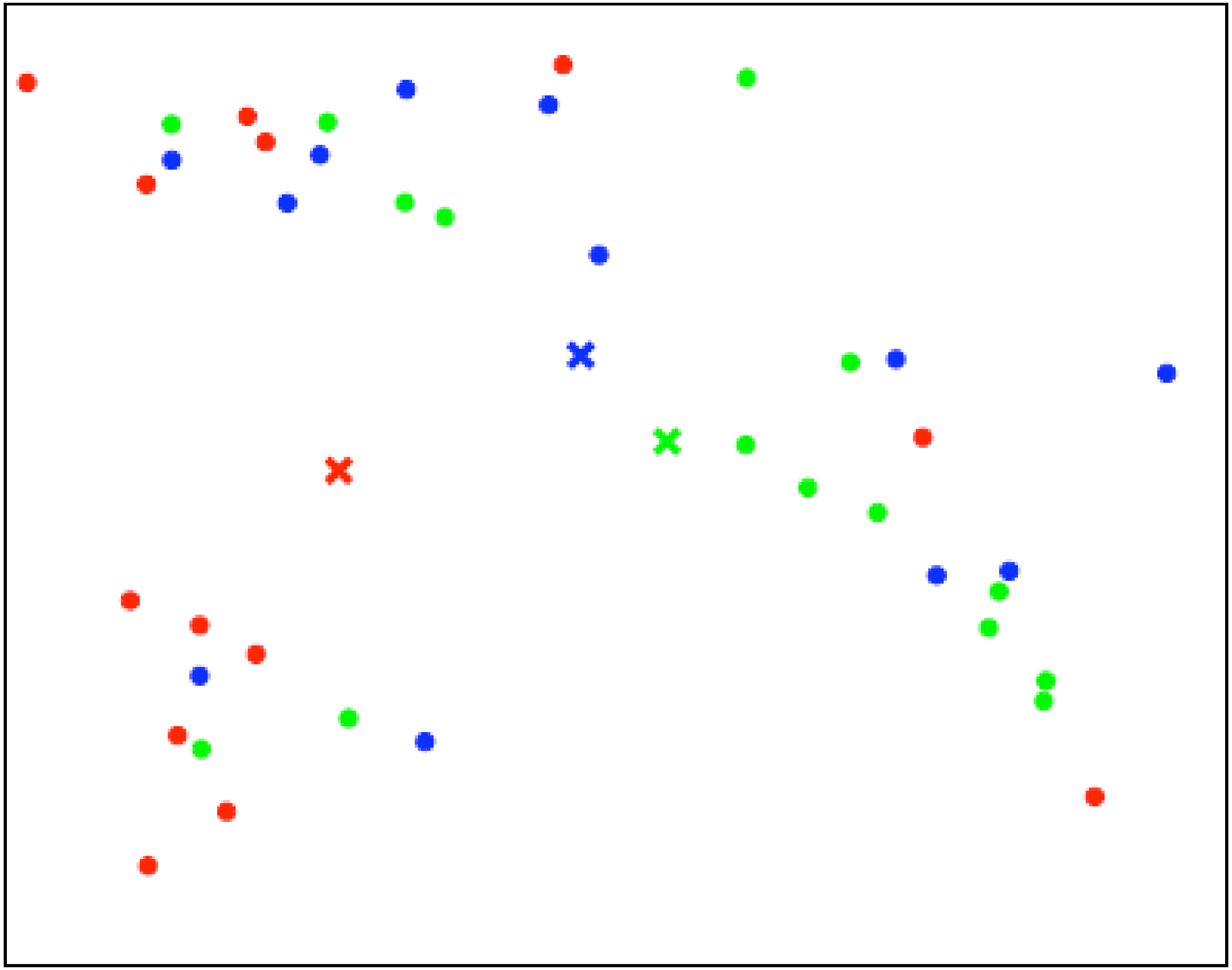
initial data

K-means Clustering Example (k=3)



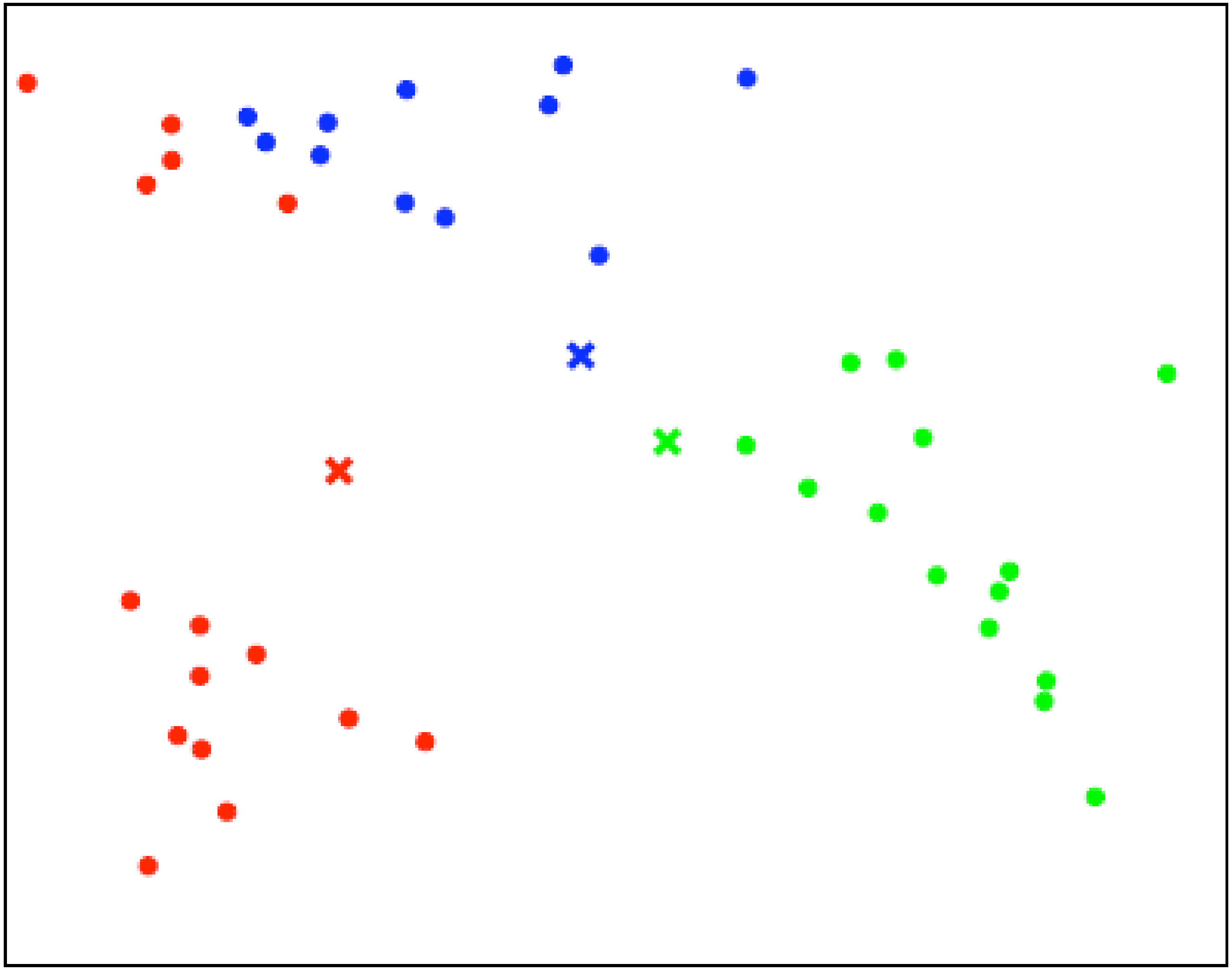
assign into 3 clusters randomly

K-means Clustering Example (k=3)



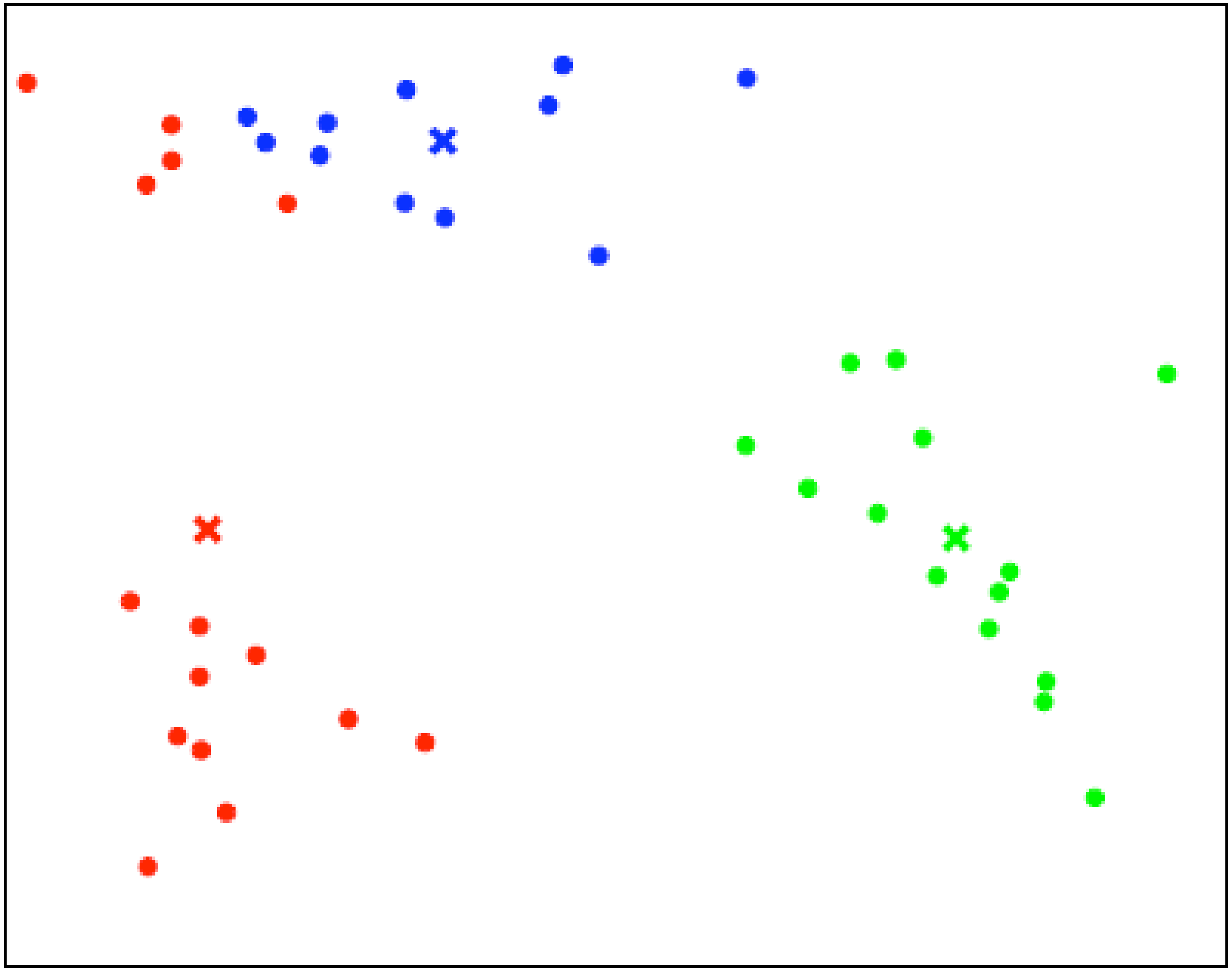
compute centroids

K-means Clustering Example (k=3)



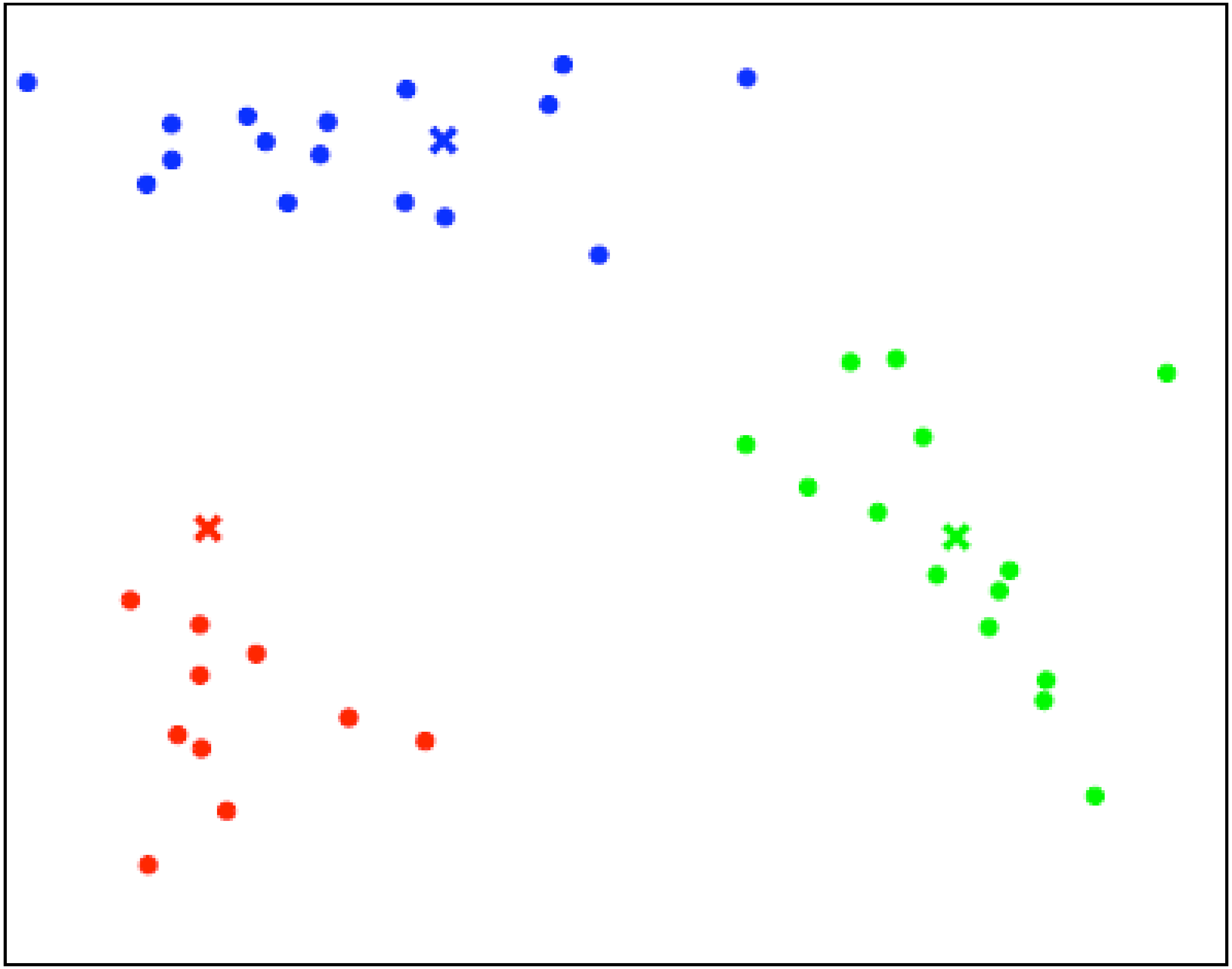
reassign clusters

K-means Clustering Example (k=3)



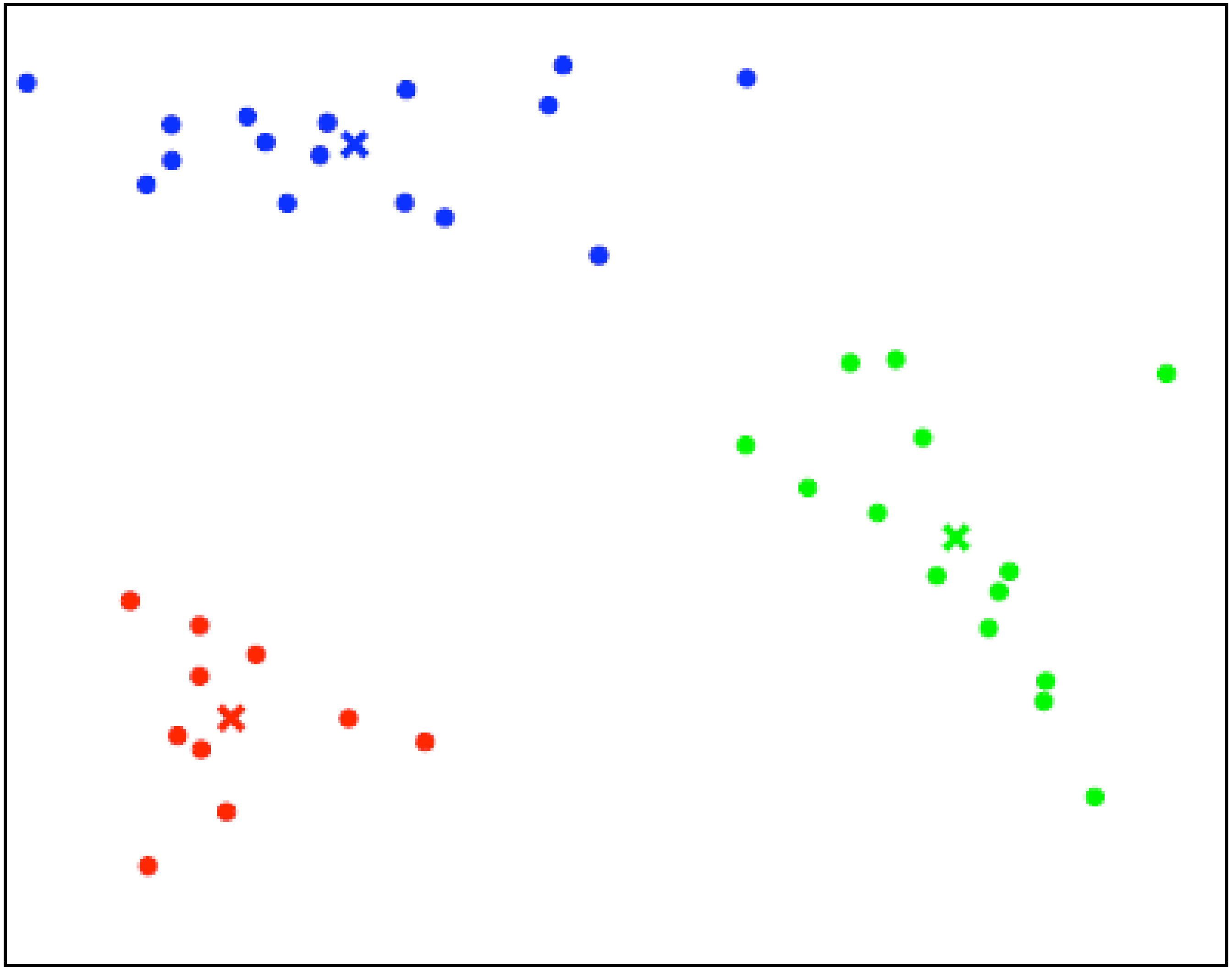
recompute centroids

K-means Clustering Example (k=3)



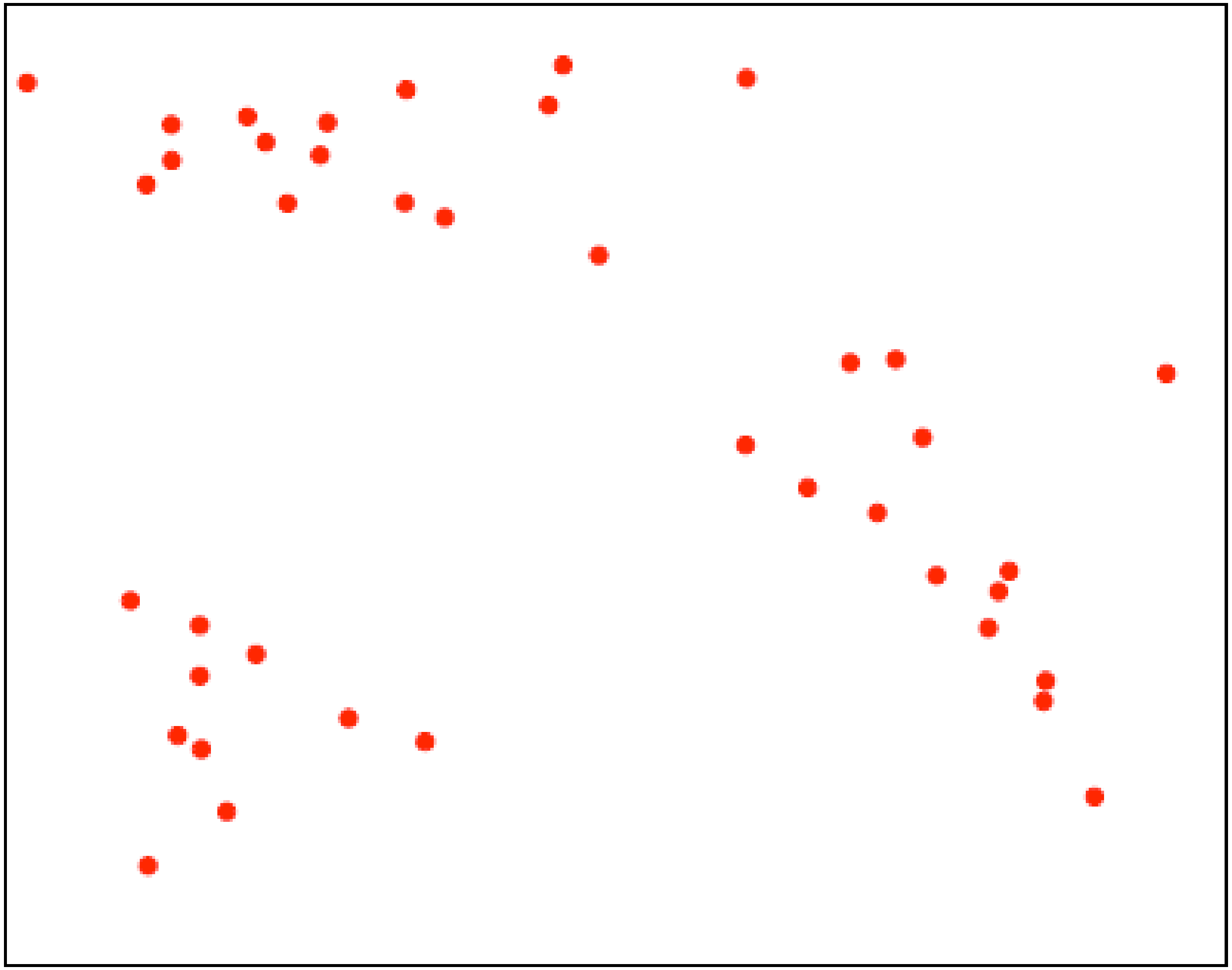
reassign clusters

K-means Clustering Example (k=3)



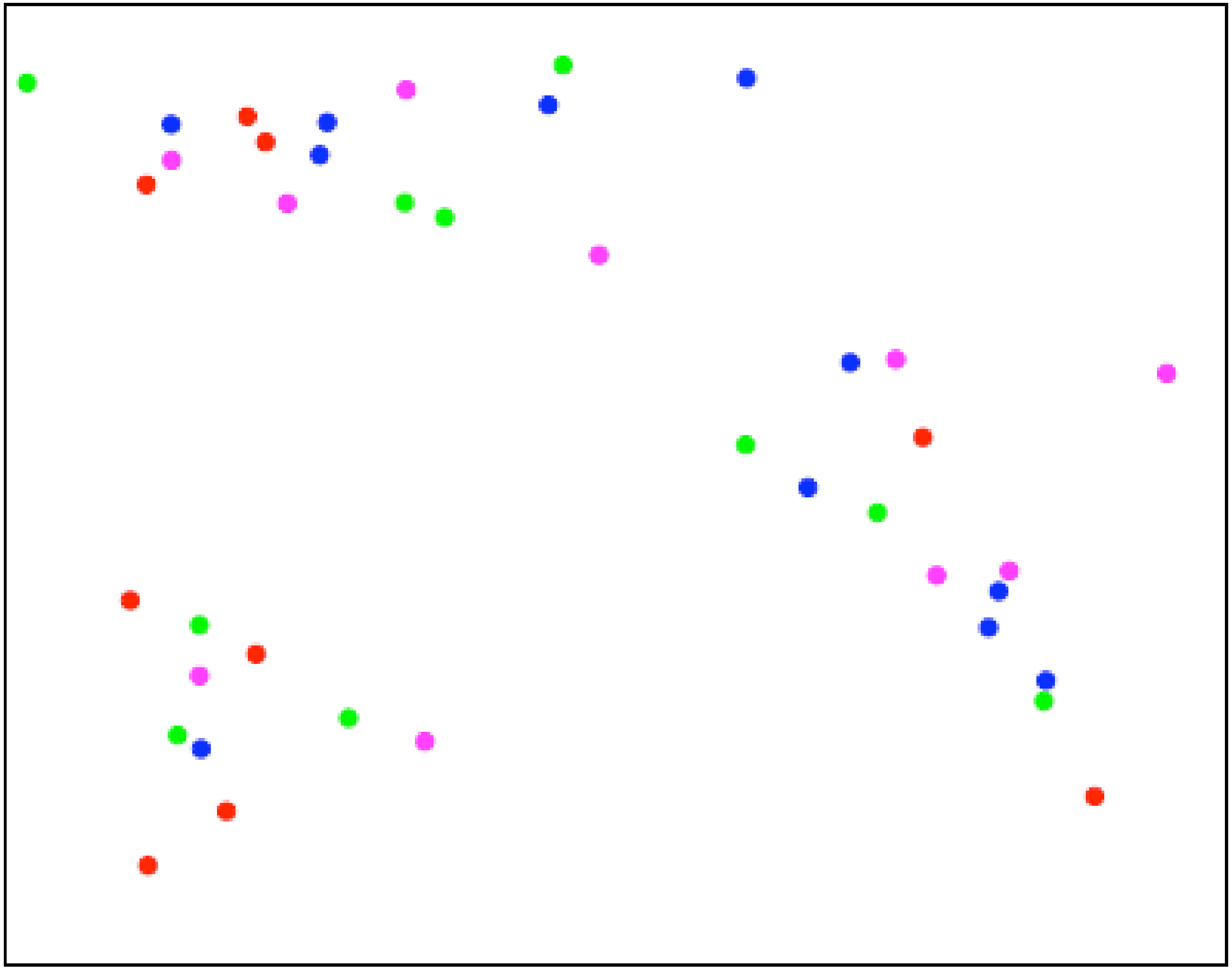
recompute centroids – done!

K-means Clustering



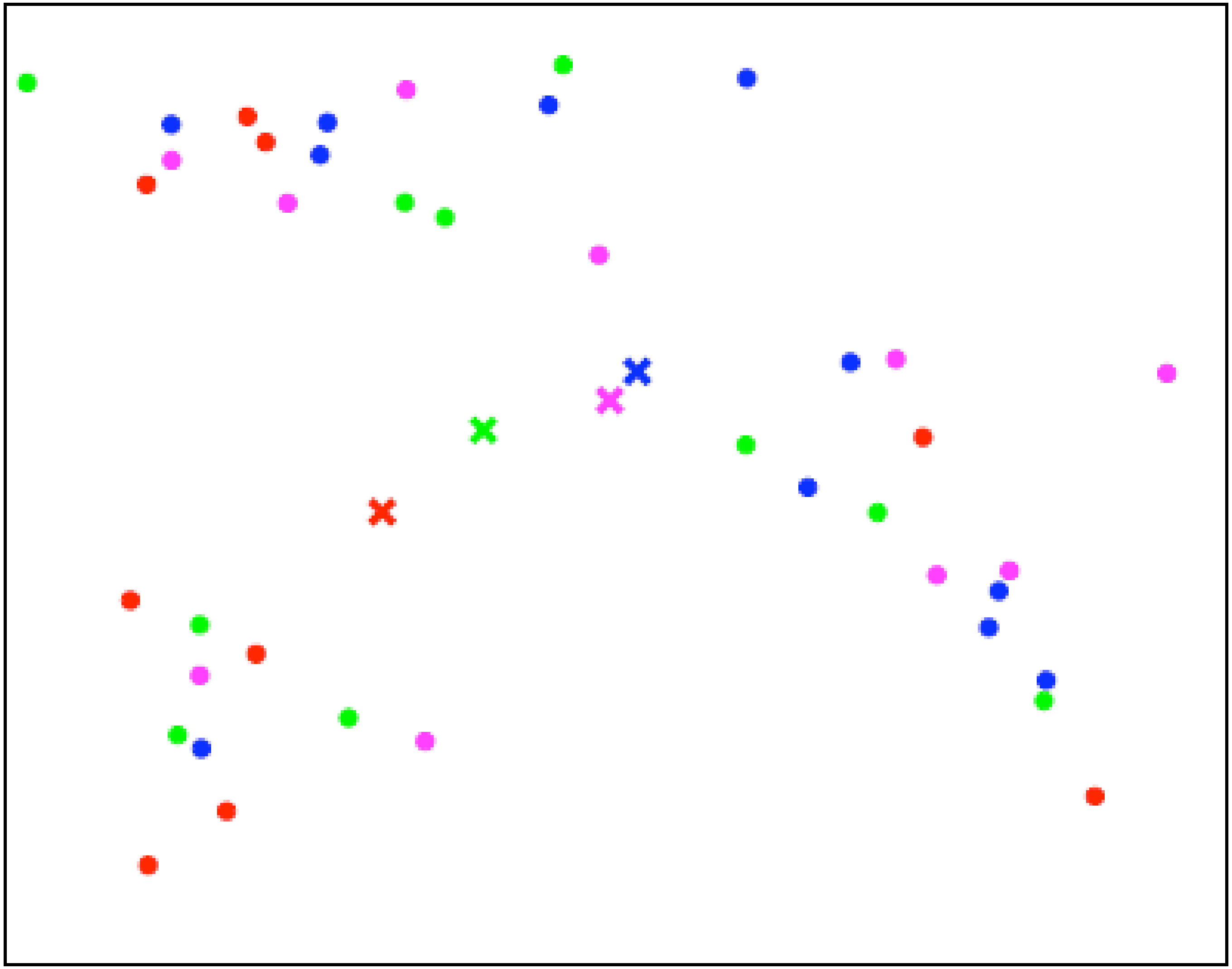
What if we do not know the right number of clusters?

K-means Clustering Example (k=4)



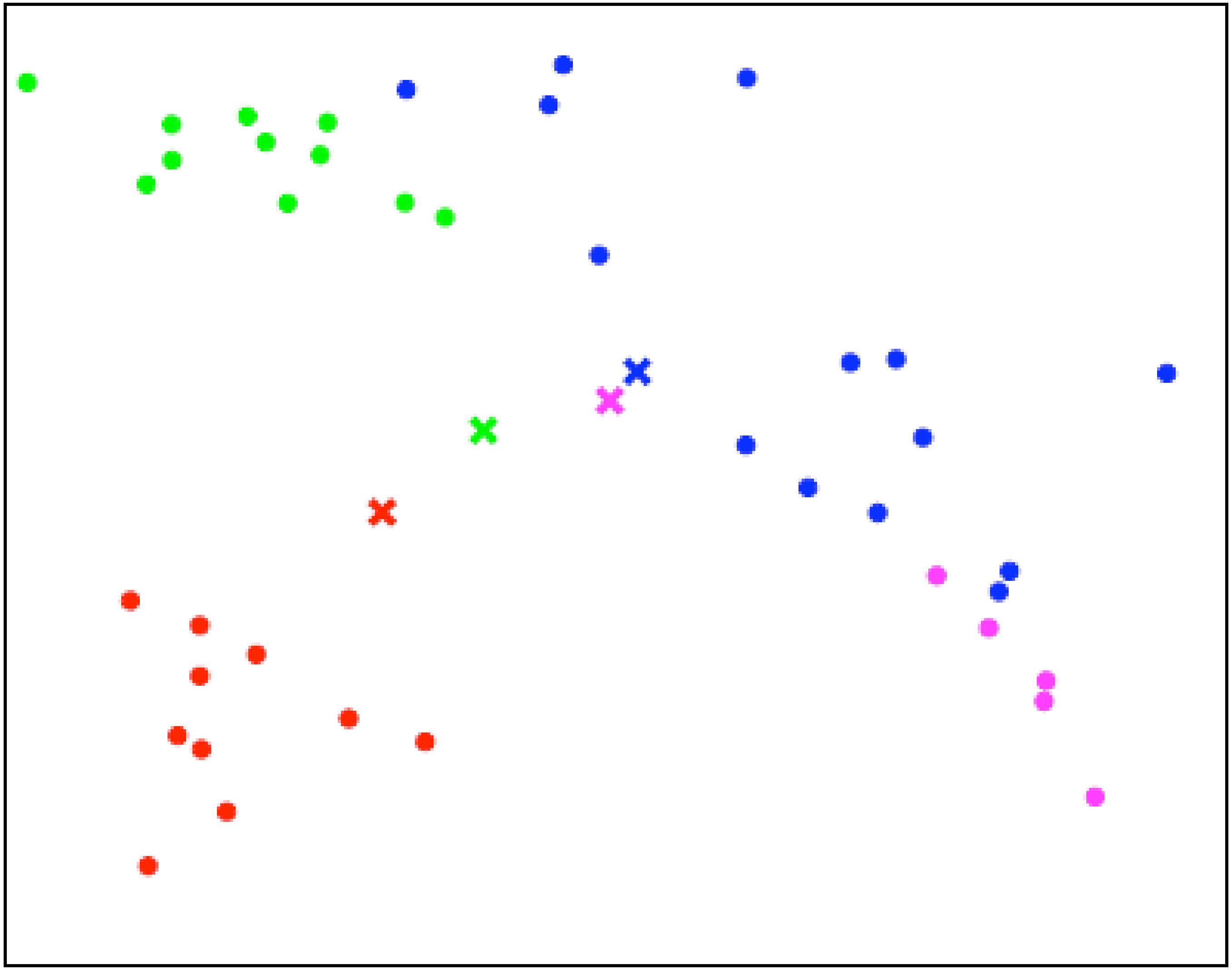
assign into 4 clusters randomly

K-means Clustering Example (k=4)



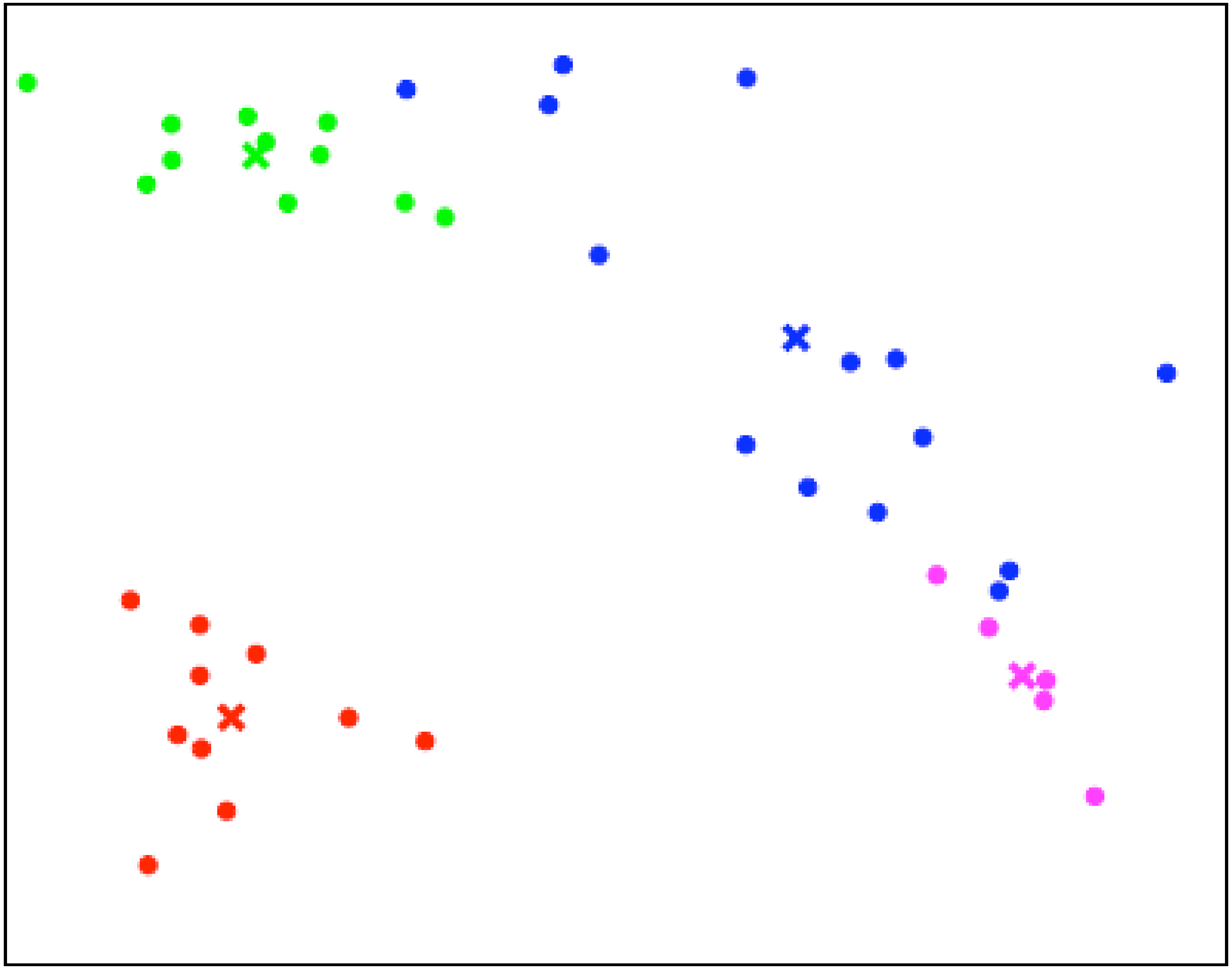
compute centroids

K-means Clustering Example (k=4)



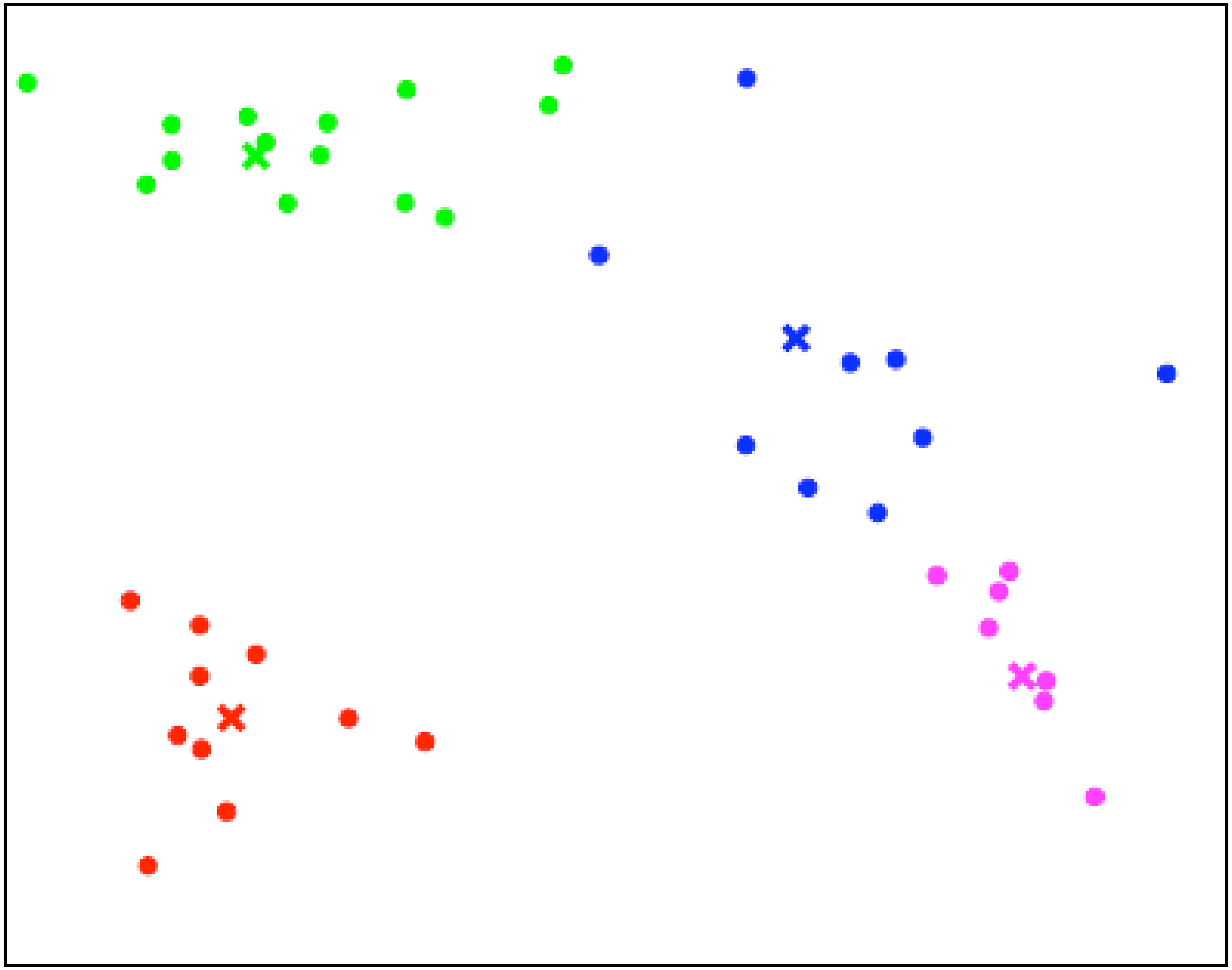
reassign clusters

K-means Clustering Example (k=4)



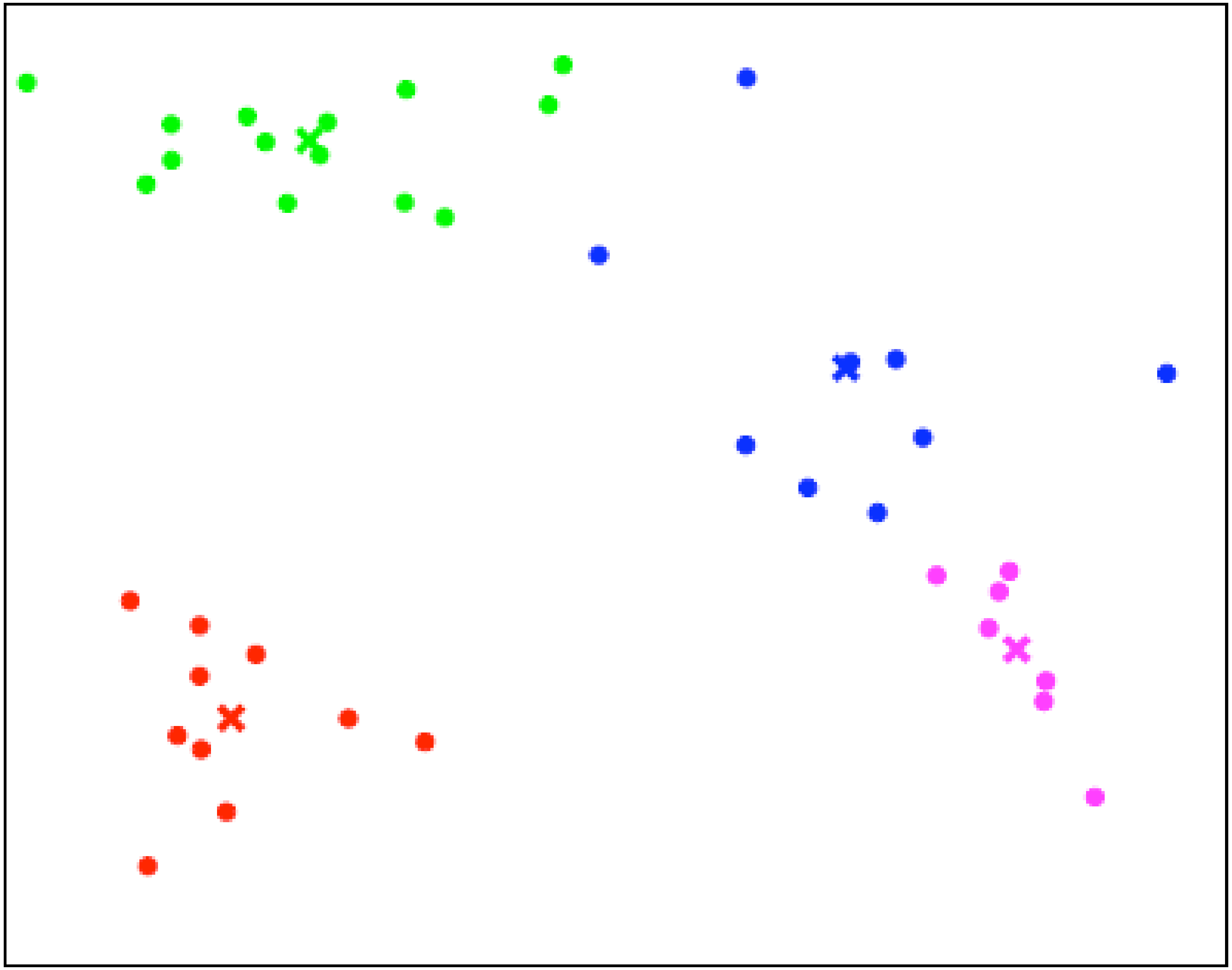
recompute centroids

K-means Clustering Example (k=4)



reassign clusters

K-means Clustering Example (k=4)



recompute centroids – done!

Problems with K-means Clustering?

Problems with K-means Clustering

- Applicable only when mean is defined – what about categorical data? (e.g. enumerated types)?
- Need to specify ***k*** (*number of clusters*) in advance.
- Best method to initially assign instances to clusters?
- Cannot handle noisy data / outliers.
- Will it always find the same answer?

K-means Clustering

- Solution depends on the initial assignment of instances to clusters, random restarts will give different solutions.

- Assigning each item to random cluster in **{1,…, K}** is unbiased, but typically results in cluster centroids near the centroid of all the data.

- Heuristic initialisation – Spread initial centroids around:
  - Place 1st centre on top of a randomly chosen data point.
  - Place 2nd centre on a data point as far as possible from 1st one.
  - Place ***i-th*** centre as far away as possible from the closest of centres ***1 ,…, i – 1.***

- How to find best ***k***?

K-means Clustering

- How to find best ***k***?

- K-means clustering is fast and efficient: ***O(t k n)***.:
  - ***n* =** Number of items (data points)
  - ***k* =** Number of clusters.
  - ***t* =** Number of iterations.

- With a randomised initialisation step, just run K-means multiple times and take the clustering with best result.

ML: Reading & Lab

- **K-Means Clustering Online Tutorials:**
  - http://www.saedsayad.com/clustering_kmeans.htm
  - http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html
- **ML Lab this Friday (27th July).:**
  - **K-Means programming assignment – On Vula now!**
  - **Due: Friday (3rd August).**

Examples of Machine Learning Types

- **Supervised Learning:**
  - **Classification.**
  - Regression.
- Unsupervised Learning:
  - Clustering.
  - Dimensionality reduction.
- Reinforcement Learning:
  - Value and policy iteration.
  - Q Learning.

Concept Learning

Concept Learning

- Learning from examples.
- General to specific ordering of hypotheses.
- Version spaces and candidate elimination algorithm.
- Inductive learning!

Inductive Learning

- **Induction versus Deduction?**
  - Difference?

Inductive Learning

- **Induction (Bottom-up logic):**
  - Generalising about properties of data based on a few data-point observations.
  - Specific to general (Learning from examples).
  - Pull 4 marbles out of a box: 3 are red, 1 is white → Box contains a distribution of 75% red and 25% white marbles.
- **Deduction (Top-down logic):**
  - Conclusion reached from general statements.
  - General to specific.
  - All robots must recharge; Ava is a robot; Ava must recharge.

The Learning Problem

- Example: Credit Approval – Formalisation:

  - Input: $\mathbf{x}$ *(customer application)*
  - Output: y *(good/bad customer?)*
  - Target function: $f : \mathcal{X} \to \mathcal{Y}$ *(ideal credit approval formula)*
  - Data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)$ *(historical records)*

    $\downarrow \quad \downarrow \quad \downarrow$

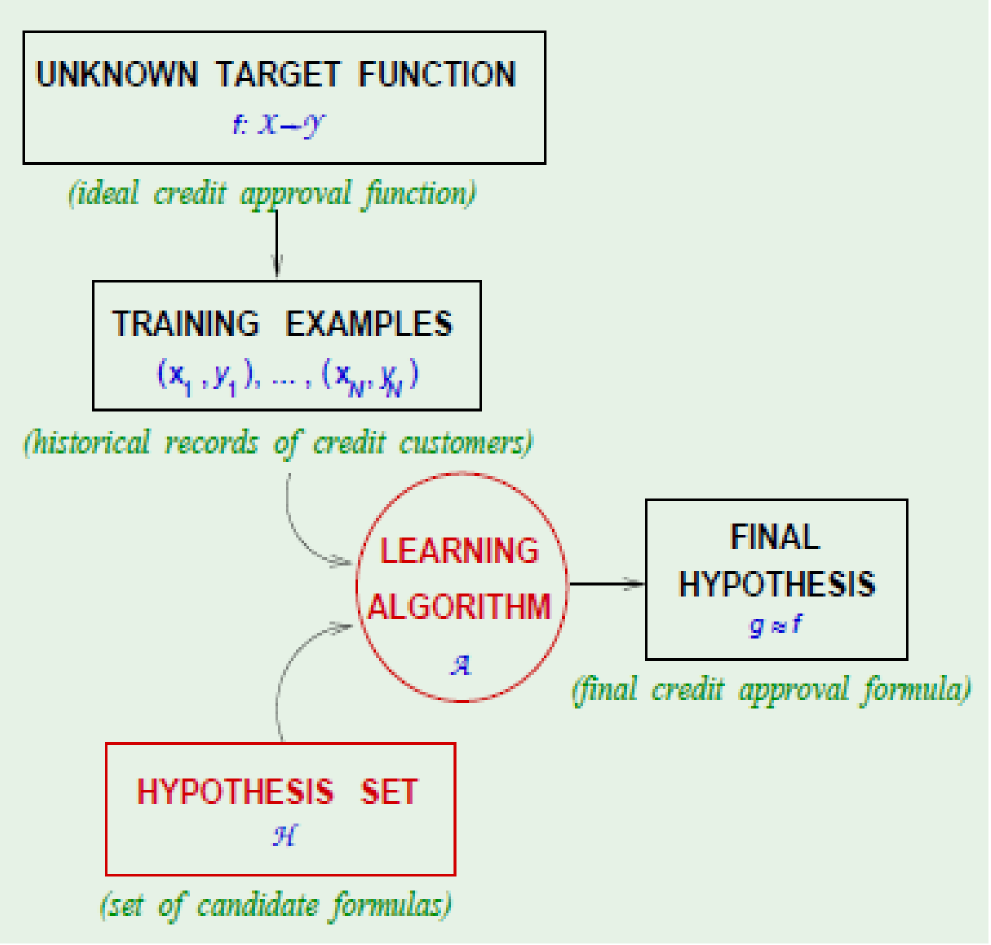
  - Hypothesis: $g : \mathcal{X} \to \mathcal{Y}$ *(formula to be used)*

The Learning Problem

- Two solution components of the learning problem:
  - Hypothesis set:

$$\mathcal{H} = \{h\} \qquad g \in \mathcal{H}$$

  - Learning algorithm.
- Together, they are referred to as the learning model.



Training Examples for Concept: Enjoy Sport

attributes

Sky	Temp	Humid	Wind	Water	Fore-cast	Enjoy Sport
Sunny	Warm	Nor[illegible]	[illegible]	Warm	Same	Yes
Sunny	Warm	Hig[illegible]	[illegible]	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

instance

- **Concept:** Days on which someone enjoys playing sport.
- **Task:** Predict the value of "Enjoy Sport" based on the values of the other attributes.

Inductive Learning Hypothesis

- Any hypothesis found to approximate the target function well over the training examples, will also approximate the target function well over the unobserved examples (Mitchell, 1997).

Concept Learning: Terminology

- **Instance space X:** Set of all possible inputs.
  - **Sky:** < Sunny, Cloudy, Rainy >
  - **AirTemp:** < Warm, Cold >
  - **Humidity:** < Normal, High >
  - **Wind:** < Strong, Weak >
  - **Water:** < Warm, Cold >
  - **Forecast:** < Same, Change >

- **Example:** x = < sunny, warm, normal, strong, warm, same >
  - Number of distinct ***instances*** and ***concepts***?

Concept Learning: Terminology

- **Instance space X:** Set of all possible inputs.
  - **Sky:** < Sunny, Cloudy, Rainy >
  - **AirTemp:** < Warm, Cold >
  - **Humidity:** < Normal, High >
  - **Wind:** < Strong, Weak >
  - **Water:** < Warm, Cold >
  - **Forecast:** < Same, Change >

- Distinct ***instances*** = $(3 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2) = 96$ instances.
- Distinct ***concepts*** = 2^{96}

Concept Learning: Terminology

- **Training examples:** **D = { < x , c(x) > }**
  - Instance **x** from **X** with target concept value **c(x)**.
  - **+ve examples:** c(x) = 1, members of target concept.
  - **-ve examples:** c(x) = 0, non-members of target concept.
  - **Target concept c:** X → { 0, 1 }

- **Hypothesis space H:** Set of possible hypotheses (e.g.: EnjoySport)
  - (5 · 4 · 4 · 4 · 4 · 4) = 5120 syntactically distinct hypotheses.
  - (4 · 3 · 3 · 3 · 3 · 3) = 973 semantically distinct hypotheses.

Concept Learning: Hypotheses

- Hypothesis h is a conjunction of constraints on attributes.

- Each constraint can be:
  - **Specific value:** e.g.: Water = Warm.
  - **Don’t care value:** e.g.: Water = ?
  - **No value allowed** (null hypothesis): e.g.: Water = Ø.

- **Example:** Hypothesis h:

	Sky	Temp	Humid	Wind	Water	Forecast	
<	Sunny	?	?	Strong	?	?	**>**

Concept Learning: Hypotheses

- **Most general hypothesis:**
  - < ?, ?, ?, ?, ?, ? >
- **Most specific hypothesis:**
  - < Ø, Ø, Ø, Ø, Ø, Ø >
- **Notation:**
  - **X:** Set of instances over which the concept is defined.
  - **e.g:** EnjoySport (***concept***) for sets of values (***instances***) for attributes: <Sky, Air Temp, Humidity, Wind, Water, Forecast>.
  - **c:** Target concept.
  - **c(x) = 1** if EnjoySport = Yes; **c(x) = 0** if EnjoySport = No.

Prototypical Concept Learning

- **Given:**
  - **Instance X:** Possible days described by the attributes: <Sky, Temp, Humidity, Wind, Water, Forecast>.
  - **Target concept c:** Enjoy Sport, X → { 0,1 }.
  - **Hypotheses H:** Conjunction of literals – e.g.:

    < Sunny, ?, ?, Strong, ?, Same >
  - **Training set D:** +ve and -ve examples of target function:

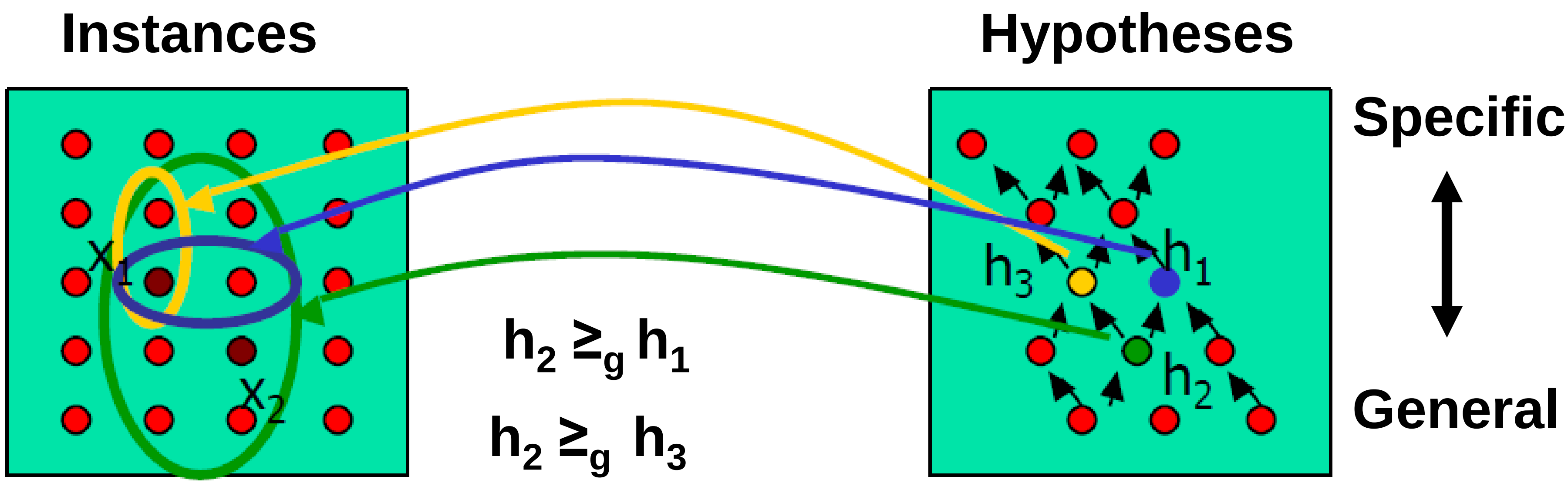
    $< x_1 , c(x_1) > ,\ldots, < x_n , c(x_n) >$
- **Determine:**
  - A hypothesis ***h*** in ***H*** such that ***h(x) = c(x)*** for all ***x*** in ***D***.

General to Specific Order

- **Consider hypotheses:**
  - $\mathbf{h_1 =}$ < Sunny, ?, ?, Strong, ?, ? >
  - $\mathbf{h_2 =}$ < Sunny, ?, ?, ?, ?, ? >
- **Set of instances covered by h_1 and h_2 :**
  - $\mathbf{h_2}$ imposes fewer constraints than $\mathbf{h_1}$ and therefore classifies more instances of x as positive: **h(x) = 1**.
- **Definition:**
  - Let h_j and h_k be Boolean valued functions defined over X.
  - Then h_j is **more general than or equal to** h_k (i.e. $h_j \geq_g h_k$)

iff: $(\forall x \in X)[\, (h_k(x) = 1) \rightarrow (h_j(x) = 1)\,]$

General to Specific Order



x1 = < Sunny, Warm, High, Strong, Cool, Same >

x2 = < Sunny, Warm, High, Light, Warm, Same >

h1 = < Sunny, ?, ?, Strong, ?, ? >

h2 = < Sunny, ?, ?, ?, ?, ? >

h3 = < Sunny, ?, ?, ?, Cool, ? >

Find S Algorithm

Initialise ***h*** to the most specific hypothesis in ***H***.

FOR each **+ve** training instance **x**:

FOR each attribute constraint a_i in ***h***:

IF the constraint a_i in ***h*** is satisfied by ***x*** **THEN** do nothing

ELSE replace a_i in ***h*** by the next more general constraint that is satisfied by ***x***.

Output hypothesis ***h***.

Find-S: Trained with Enjoy Sport

Example	Sky	Temp	Humid	Wind	Water	Forecst	EnjoySpt
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

- Initialise ***h*** to most specific hypothesis in ***H***:
  - $\mathbf{h_0} =$ < Ø, Ø, Ø, Ø, Ø, Ø >
- ?

Find-S: Trained with Enjoy Sport

Example	Sky	Temp	Humid	Wind	Water	Forecst	EnjoySpt
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

- Initialise ***h*** to most specific hypothesis in ***H***:
  - $\mathbf{h_0}$ **=** < Ø, Ø, Ø, Ø, Ø, Ø >
- Hypothesis is too specific – replace with:
  - $\mathbf{h_1}$ **=** < Sunny, Warm, Normal, Strong, Warm, Same >
- ?

Find-S: Trained with Enjoy Sport

Example	Sky	Temp	Humid	Wind	Water	Forecst	EnjoySpt
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

- $\mathbf{h_1}$ is still too specific – 2nd training example forces Find-S to further generalise **h**:
  - “?” in place of any value not satisfied by new example.
  - $\mathbf{h_2}$ = < Sunny, Warm, ?, Strong, Warm, Same >
- ?

Find-S: Trained with Enjoy Sport

Example	Sky	Temp	Humid	Wind	Water	Forecst	EnjoySpt
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

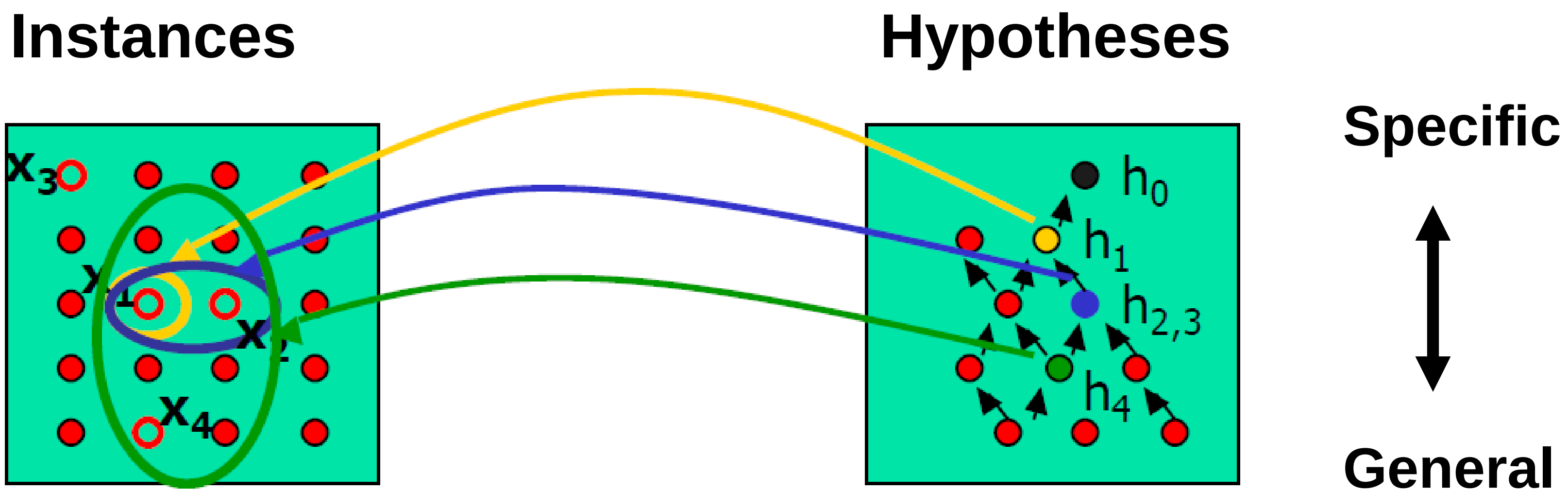
- Find-S ignores 3rd training example (ignores every -ve example).
- ?

Find-S: Trained with Enjoy Sport

Example	Sky	Temp	Humid	Wind	Water	Forecst	EnjoySpt
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

- 4th example - further generalisation of ***h***.
  - h_4 = < Sunny, Warm, ?, Strong, ?, ? >

Hypothesis Space Search by Find-S



h0 = < Ø, Ø, Ø, Ø, Ø, Ø >

x1 = < Sunny, Warm, Normal, Strong, Warm, Same >, +

h1 = < Sunny, Warm, Normal, Strong, Warm, Same >

x2 = <Sunny, Warm, High, Strong, Warm, Same>, +

h2 =< Sunny, Warm, ?, Strong, Warm, Same >

x3 = < Rainy, Cold, High, Strong, Warm, Change >, -

h3 =< Sunny, Warm, ?, Strong, Warm, Same >

x4 = < Sunny, Warm, High, Strong, Cool, Change >, +

h4 = < Sunny, Warm, ?, Strong, ?, ? >

Properties of Find-S

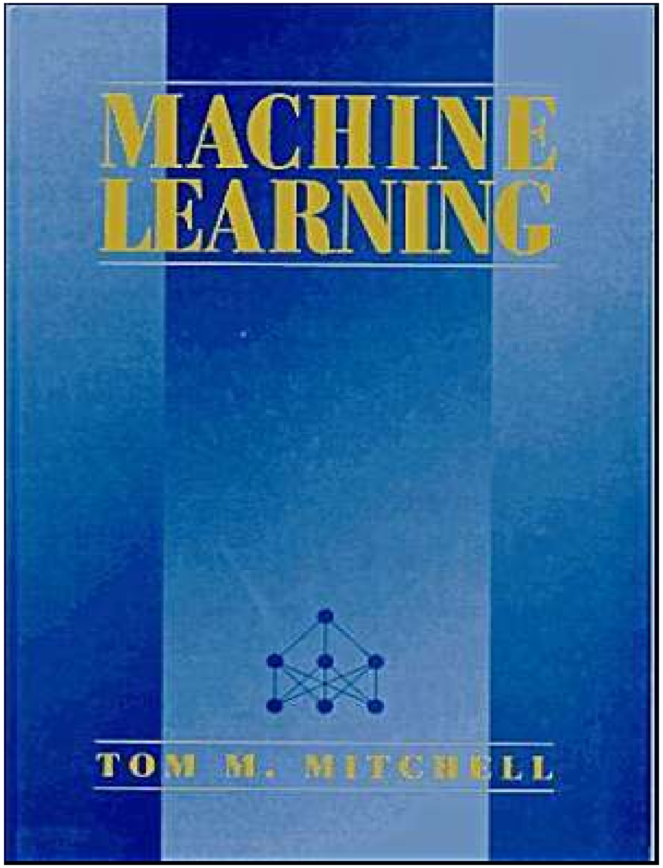
- **Hypothesis Space:** Described by conjunctions of attributes.
- **Find-S:** Outputs the *most specific hypothesis* within ***H*** that is consistent with the +ve training examples.
- Always prefers the most specific hypothesis.
- But – has the learner converged to the only hypothesis in ***H*** consistent with the data (correct target concept)?
- Problems with Find-S?

Problems with Find-S

- Why should we prefer the most specific hypothesis?.
- Impossible to know if only one unique hypothesis remains.
- We will not detect inconsistent data (noise!) since all –ve examples are ignored.
- What if there are multiple maximally specific hypotheses?

Next … Candidate Elimination Algorithm

ML: Reading



Chapter 2: Concept Learning