Semi-supervise Learning

Ramesh Ragala
VIT Chennai

Semi-supervised Learning

- Semi-supervised learning (SSL) is a class of machine learning techniques that make use of both labeled and unlabeled data for training.
- Unsupervised Learning
  - Let $X = (x_1, x_2, \ldots x_n)$ be a set of n examples or points, where $x_i \in \mathcal{X}$ for all $i \in [n] = \{1, 2, \ldots, n\}$.
  - it is assumed that the points are drawn i.i.d. (independently and identically distributed) from a common distribution on $\mathcal{X}$.
  - The goal of unsupervised learning is to find interesting structure in the data X.
  - It has been argued that the problem of unsupervised learning is fundamentally that of estimating a density which is likely to have generated X.

Semi-supervised Learning

- supervised Learning
  - The goal is to learn a mapping from x to y, given a training set made of pairs (x_i, y_i).
  - Here, the $y_i \in Y$ are called the labels or targets of the examples x_i → If the labels are numbers Y denotes the column vector of labels.
  - The pairs (x_i, y_i) are sampled i.i.d. from some distribution which here ranges over $X \times Y$. → This task is well defined, since a mapping can be evaluated through its predictive performance on test examples
  - it is assumed that the points are drawn i.i.d. (independently and identically distributed) from a common distribution on $\mathcal{X}$.
  - When $y \in R$ or R^d (i.e. when the labels are continuous), the task is called regression.
  - There are two families of algorithms for supervised learning.
    - Generative algorithms → try to model the class-conditional density by some unsupervised learning procedure. A predictive density can
    - then be inferred by applying Bayes theorem

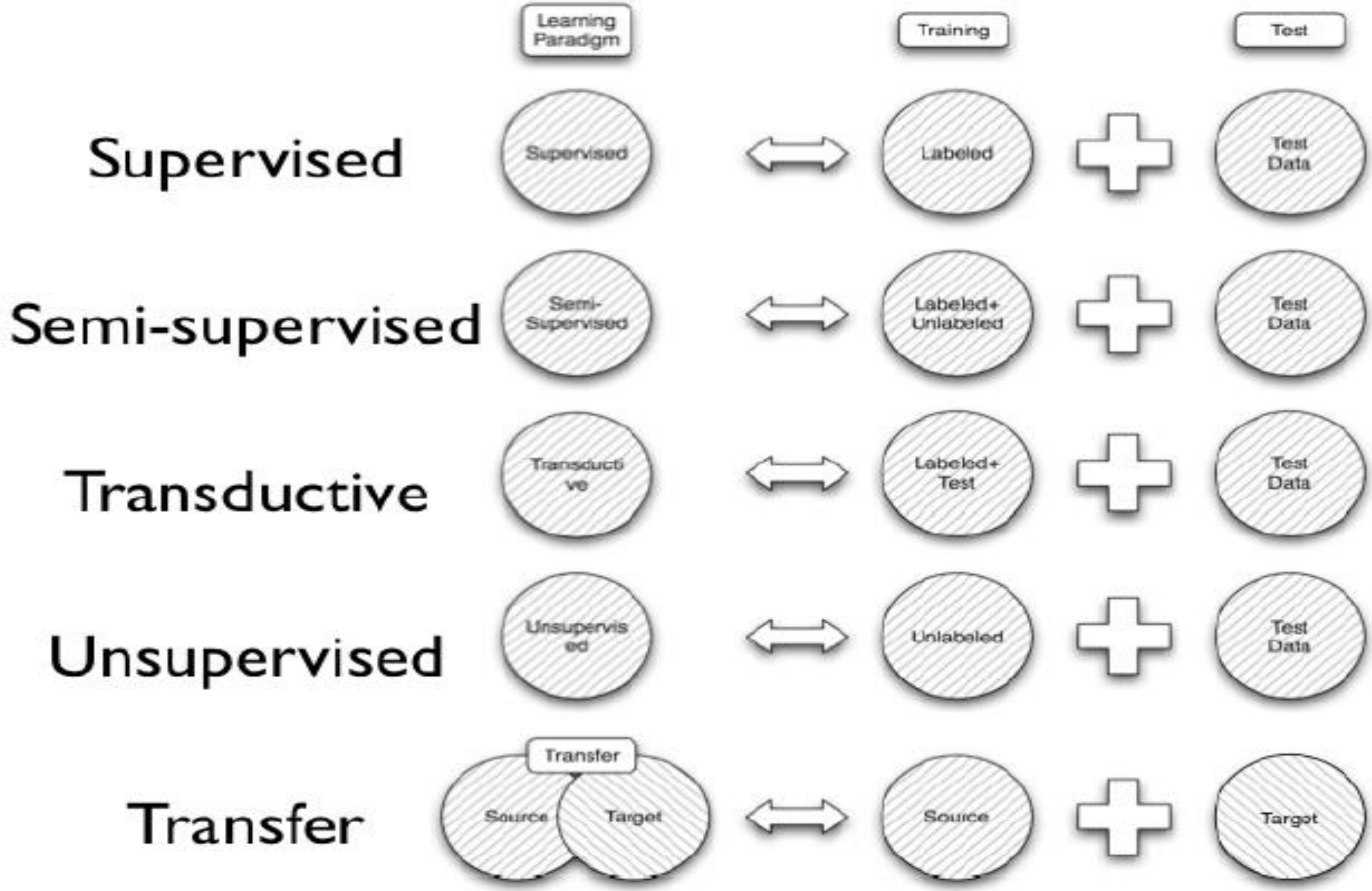
Semi-supervised Learning

- supervised Learning
- There are two families of algorithms for supervised learning.
  - Generative algorithms → try to model the class-conditional density p(x/y) by some unsupervised learning procedure. A predictive density can then be inferred by applying Bayes theorem:

$$p(y|x) = \frac{p(x|y)p(y)}{\int_y p(x|y)p(y)dy}.$$

  - Discriminative algorithms → do not try to estimate how the x_i have been generated, but instead concentrate on estimating p(y/x). Some discriminative methods even limit themselves to modeling whether p(y/x) is greater than or less than 0.5. Example SVM

Semi-supervised Learning



Semi-supervised Learning

- Semi-supervised learning falls between
  - Unsupervised learning (without any labeled training data) and
  - Supervised learning (with completely training data)
- Learn predictive tasks
  - Uses both labeled data and unlabeled data
  - Small amount of labeled data
  - Large amount of unlabeled data
- The dataset $X = (x_i)_{i \in [n]}$ can be divided into two parts:
  - The points $X_l = (x_1, x_2, \ldots, x_l)$ for which labels $Y_l = (y_1, y_2, \ldots y_l)$ are provided.
  - The points $X_u = (x_{l+1}, x_{l+2}, \ldots x_{l+u})$ does not know the labels
  - This is a standard semi-supervised learning
- Semi-supervised learning with constraints
  - Partial supervision is possible
  - Example: these points have the same target.

Semi-supervised Learning

- Two types of learning will be used in Semi-supervised learning and sometime semi-supervised learning may refer either of
  - Transductive learning
  - Inductive learning
- Transductive learning
  - The idea of transduction is to perform predictions only for the test points. i.e it is used to infer the correct labels for the given unlabeled data (x_{l+1}, x_{l+2}, … x_{l+u}) only.

Given $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{l}$ and $\{\mathbf{x}_i\}_{i=l+1}^{l+u}$, learn a function $f : \mathcal{X}^{l+u} \longrightarrow \mathcal{Y}^{l+u}$ so that f is expected to be a good predictor on the unlabeled data $\{\mathbf{x}_i\}_{i=l+1}^{l+u}$.

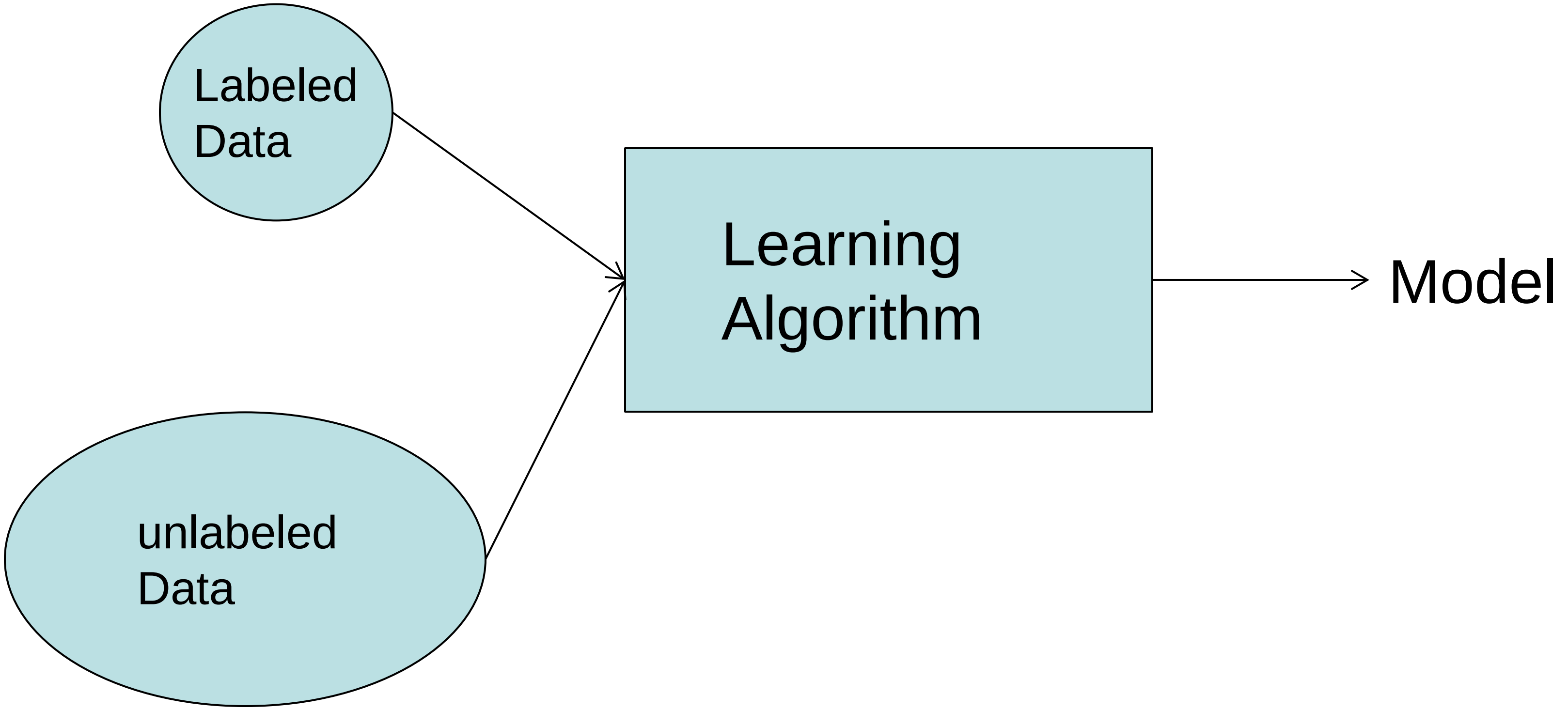
- Inductive learning
  - the goal is to output a prediction function which is defined on the entire space X. i.e. it is used to infer the correct mapping from X to Y.

Given $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{l}$ and $\{\mathbf{x}_i\}_{i=l+1}^{l+u}$, learn a function $f : \mathcal{X} \longrightarrow \mathcal{Y}$ so that f is expected to be a good predictor on future data.

Semi-supervised Learning

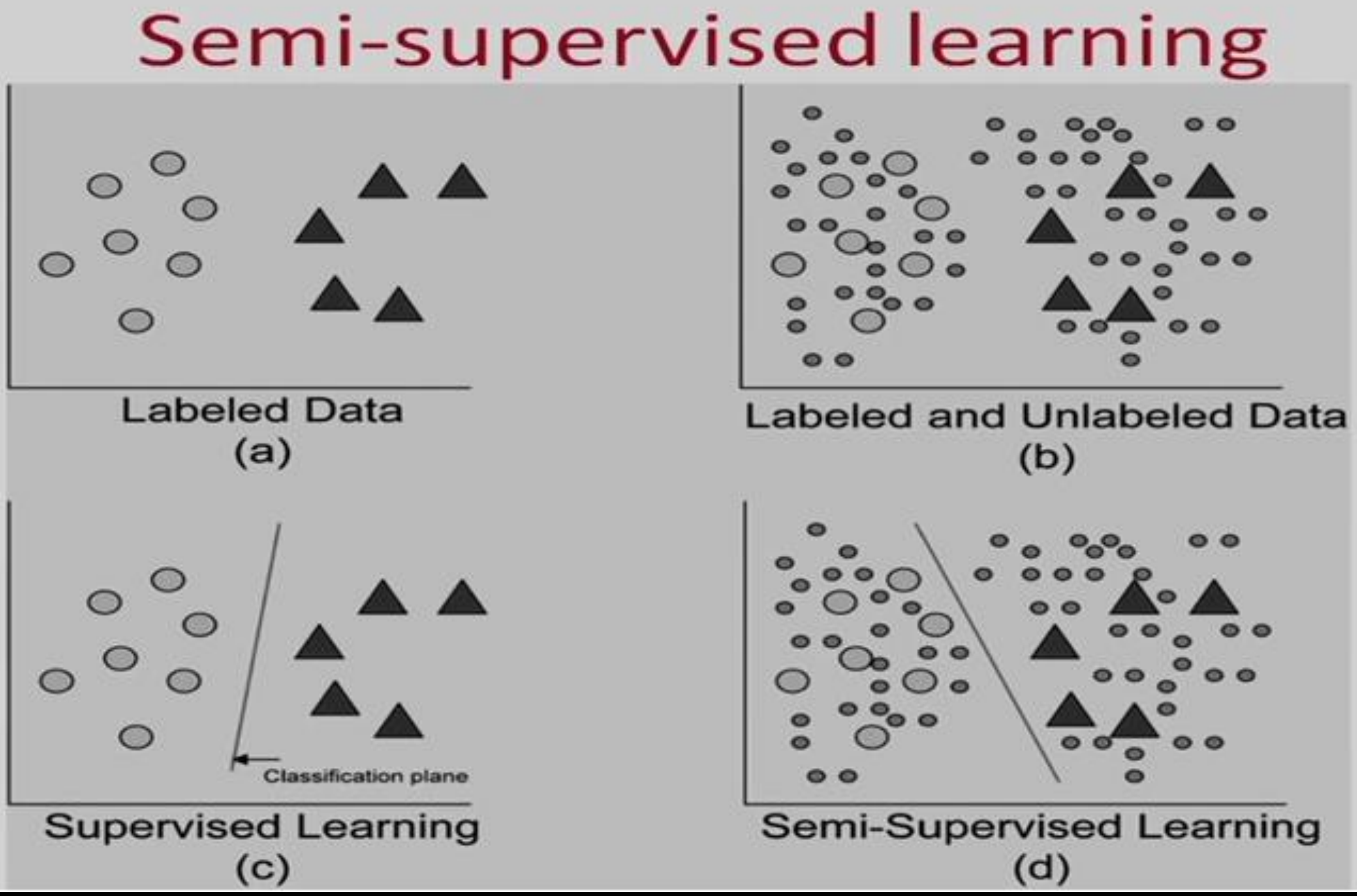
- Examples of Semi-supervised learning
  - Self-training
  - Co-training
- Applications of Semi-supervised Learning
  - Speech analysis
  - Telephone conversation transcription
    - 400 hours annotation time for each hour of speech
  - Protein sequence classification
  - Web page classification

Semi-supervised Learning



Simple architecture of Semi-supervised learning

Semi-supervised Learning

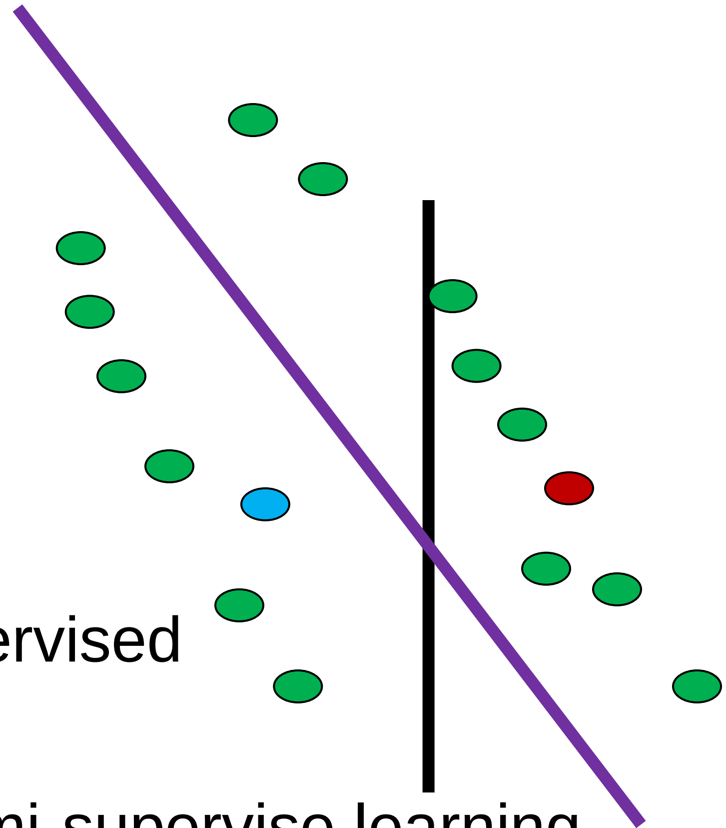


Need of Semi-supervised Learning

- Labeled data is costly for many applications
- The acquisition of labeled data for learning problem often requires a skilled human agent or a physical experiments
- Examples:
  - Speech Analysis
  - Classification of web based text
- Unlabeled data is not expensive and able to get large quantity also
- By using these combination, it can produce considerable improvement in learning accuracy

Semi-supervised Learning

Supervised Learning



- Class – 1 sample
- Class – 2 sample
- Decision boundary using supervised
- Unlabeled sample
- Decision boundary using semi-supervise learning

Brief history of Semi-supervised Learning

- The earliest idea about using unlabeled data in classification is “selflearning”or “self-training” or “self-labeling” or “decision-directed learning”.
- This is a wrapper-algorithm that repeatedly uses a supervised learning method.
- It starts by training on the labeled data only.
- In each step a part of the unlabeled points is labeled according to the current decision function;
- Then the supervised method is retrained using its own predictions as additional labeled points.

Brief history of Semi-supervised Learning

- An unsatisfactory aspect of self-learning is that the effect of the wrapper depends on the supervised method used inside it.
- If self-learning is used with empirical risk minimization and 1-0-loss, the unlabeled data will have no effect on the solution at all.
- If instead a margin maximizing method is used, as a result the decision boundary is pushed away from the unlabeled points.

Brief history of Semi-supervised Learning

- An unsatisfactory aspect of self-learning is that the effect of the wrapper depends on the supervised method used inside it.
- Closely related to semi-supervised learning is Transduction or Transductive Inference. It is coined by Vapnik.
- In contrast to inductive inference, no general decision rule is inferred, but only the labels of the unlabeled (or test) points are predicted.

Semi-supervised Learning

- The classes of semi-supervised learning methods
  - Generative Models
  - Low – Density Separation
  - Graph – Based Methods
  - Change of Representation
  - Self-training
  - Co-training

Semi-supervised Clustering

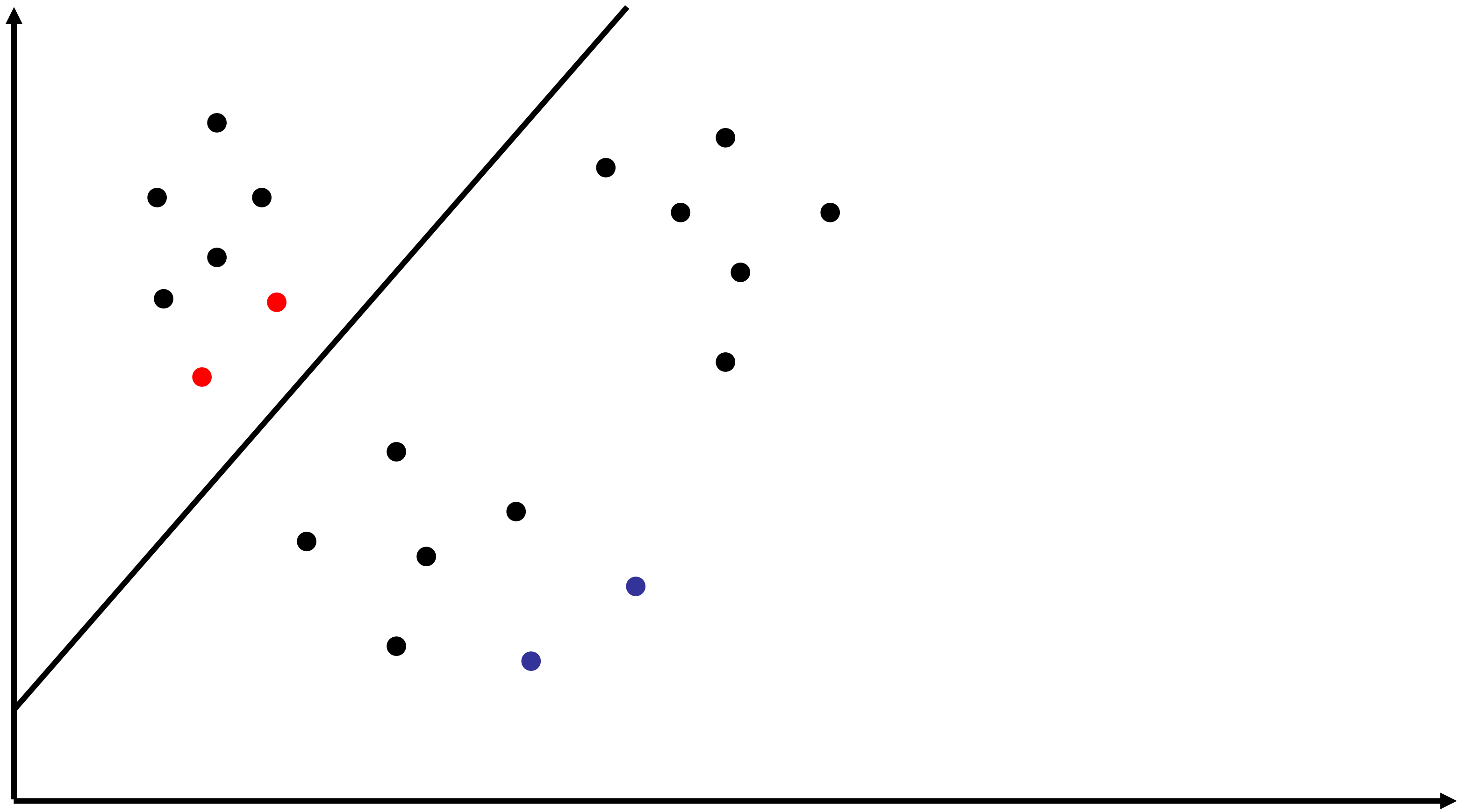
- In certain clustering tasks it is possible to obtain limited supervision in the form of pairwise constraints, i.e., pairs of instances labeled as belonging to same or different clusters.
- The resulting problem is known as semi-supervised clustering, an instance of semi-supervised learning stemming from a traditional unsupervised learning setting.
- Several algorithms exist for enhancing clustering quality by using supervision in the form of constraints

Semi-supervised Clustering

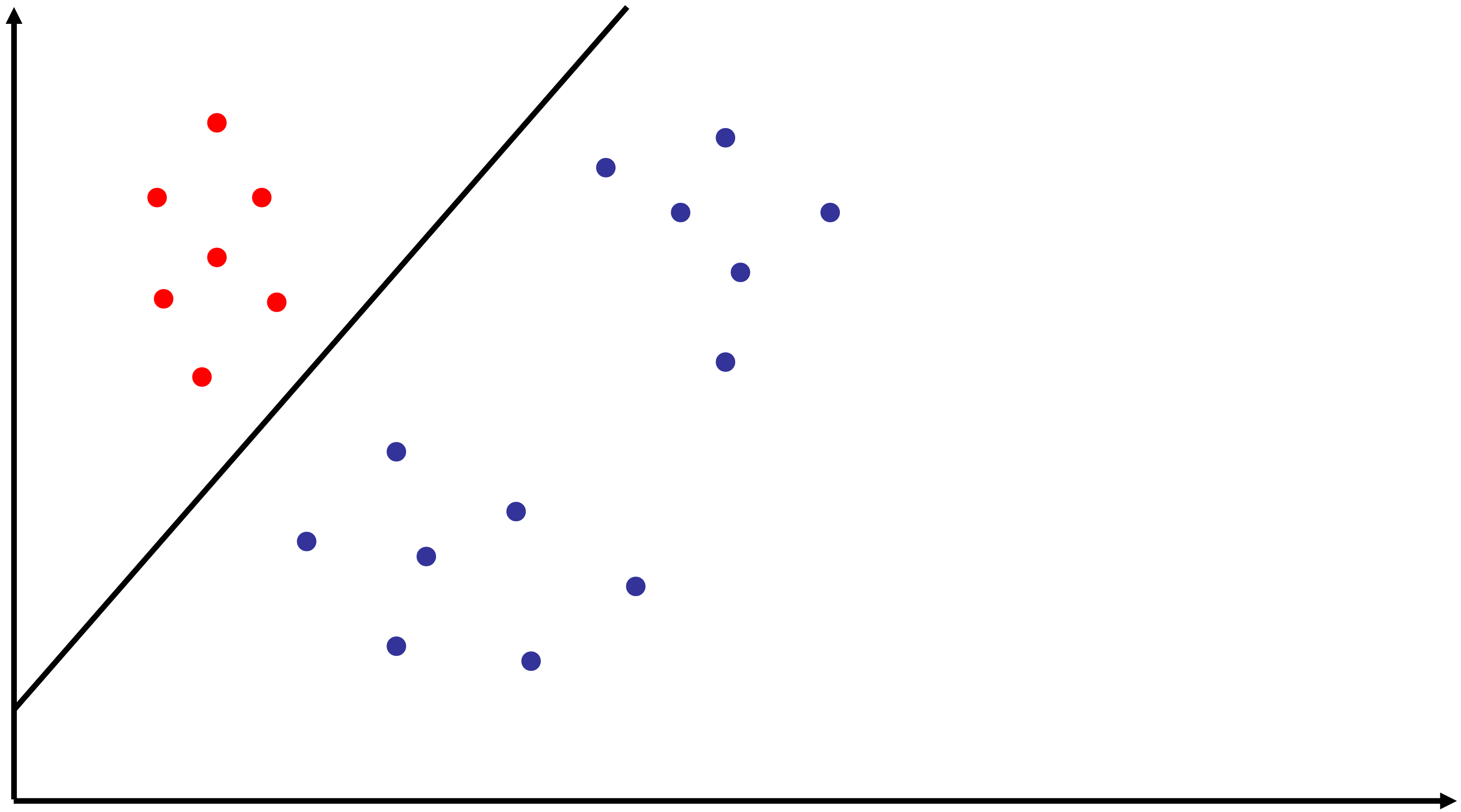
- Combines labeled and unlabeled data during training to improve performance:
  - Semi-supervised classification: Training on labeled data exploits additional unlabeled data, frequently resulting in a more accurate classifier.
  - Semi-supervised clustering: Uses small amount of labeled data to aid and bias the clustering of unlabeled data.

Unsupervised clustering	⟷	Semi-supervised learning	⟷	Supervised classification

Semi-Supervised Classification Exar



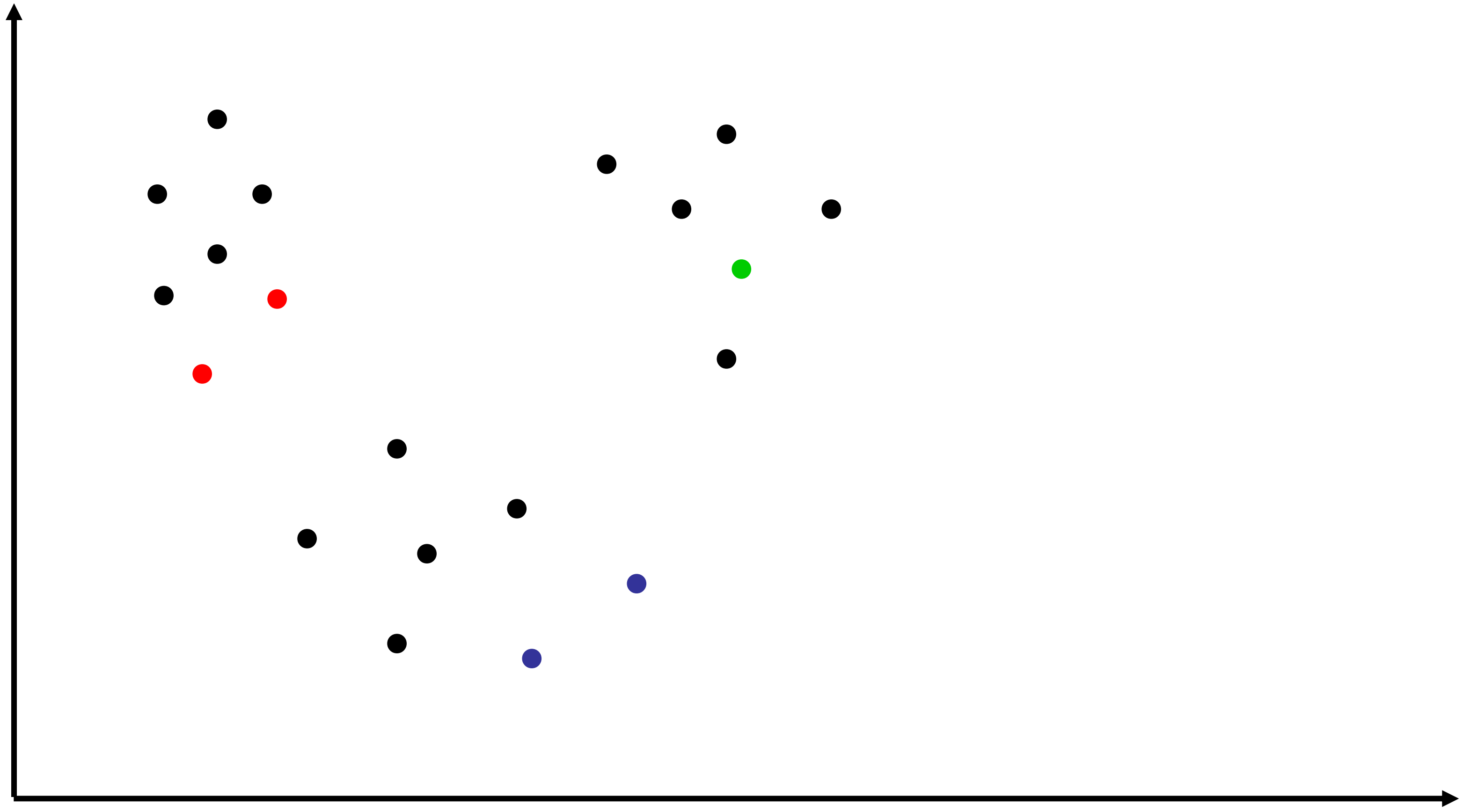
Semi-Supervised Classification Example



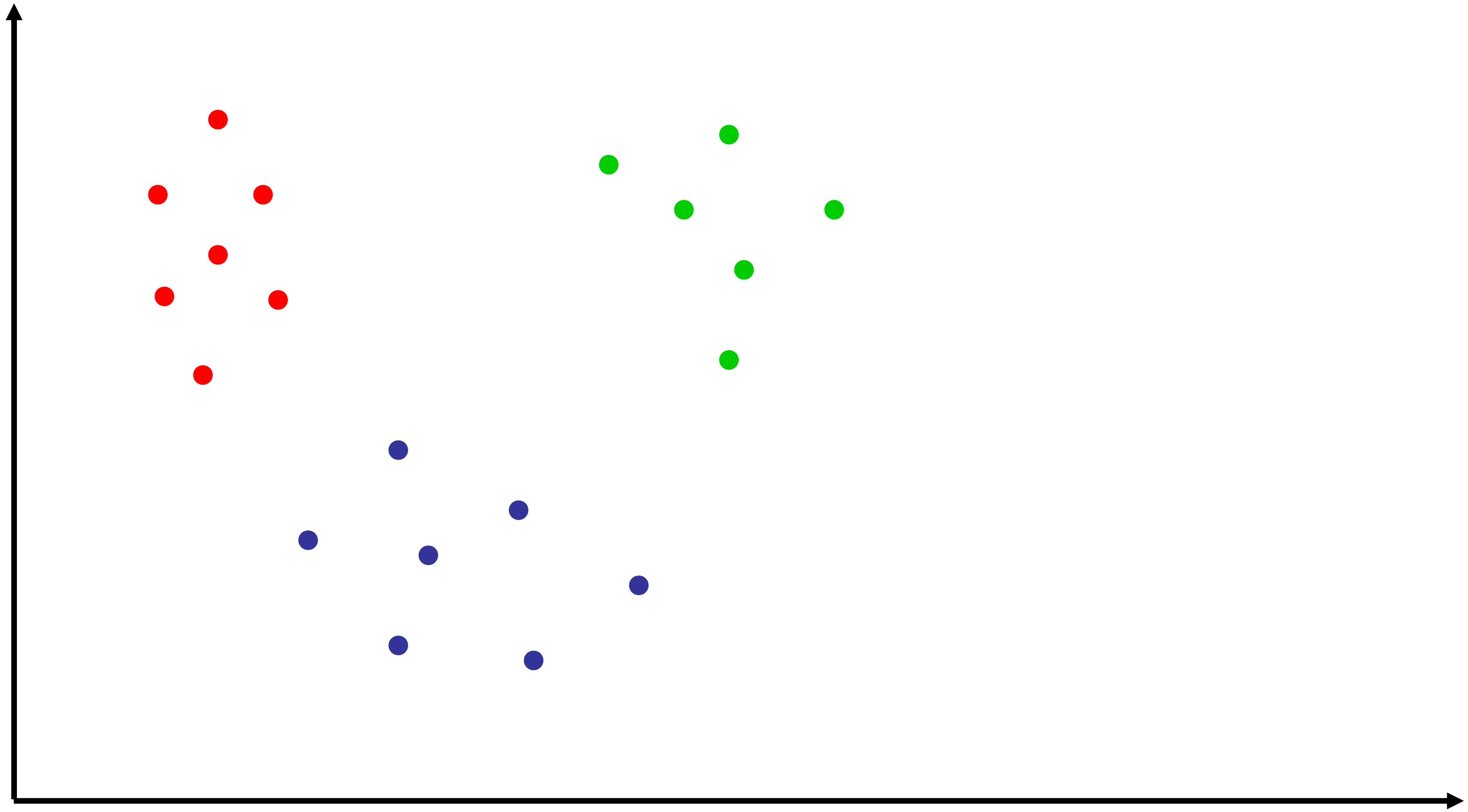
Semi-supervised Classification

- Algorithms:
  - Semisupervised EM [Ghahramani:NIPS94,Nigam:ML00].
  - Co-training [Blum:COLT98].
  - Transductive SVM's [Vapnik:98,Joachims:ICML99].
  - Graph based algorithms
  - Assumptions:
  - Known, fixed set of categories given in the labeled data.
  - Goal is to improve classification of examples into these known categories.

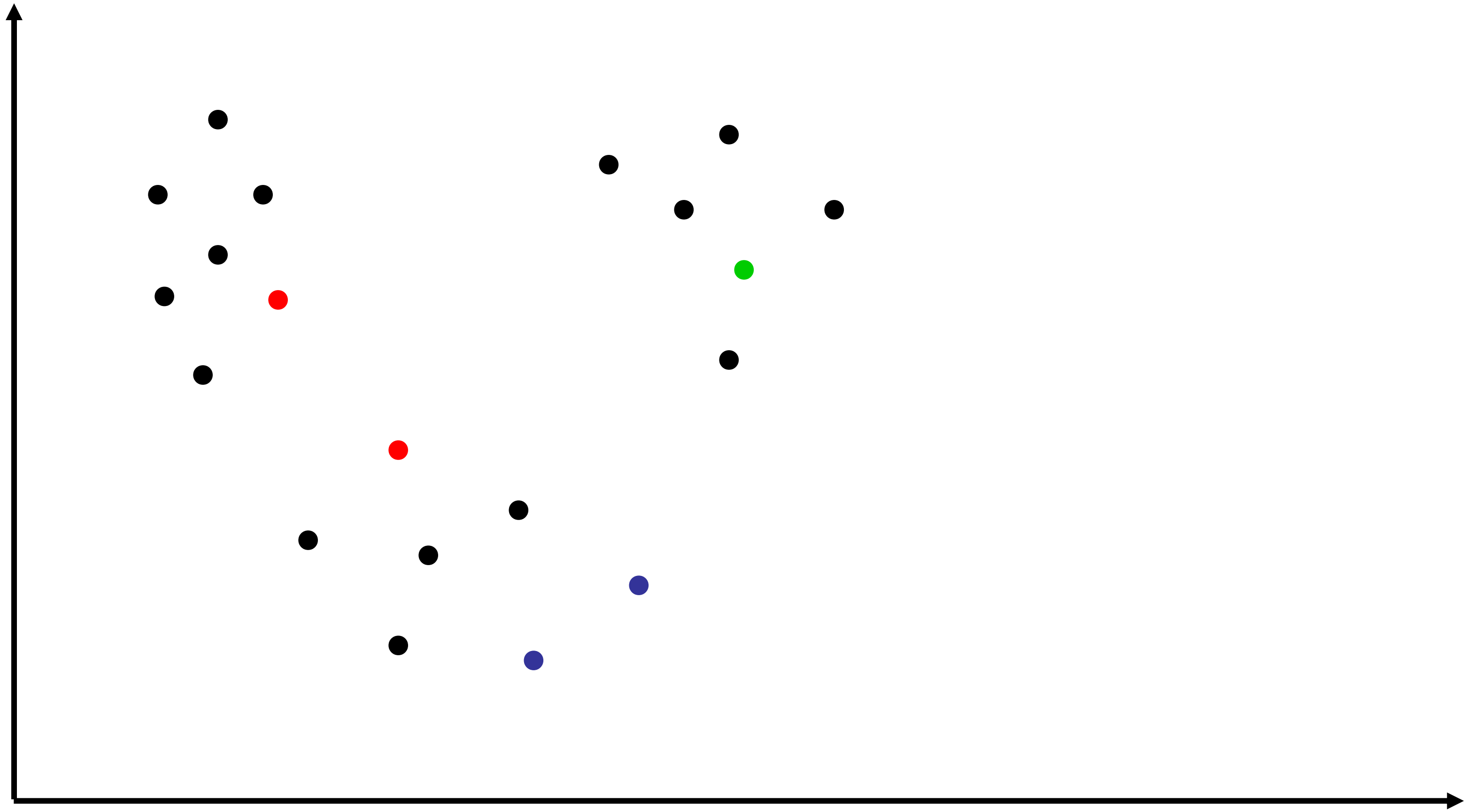
Semi-Supervised Clustering Exampl



Semi-Supervised Clustering Exampl



Second Semi-Supervised Clustering Example



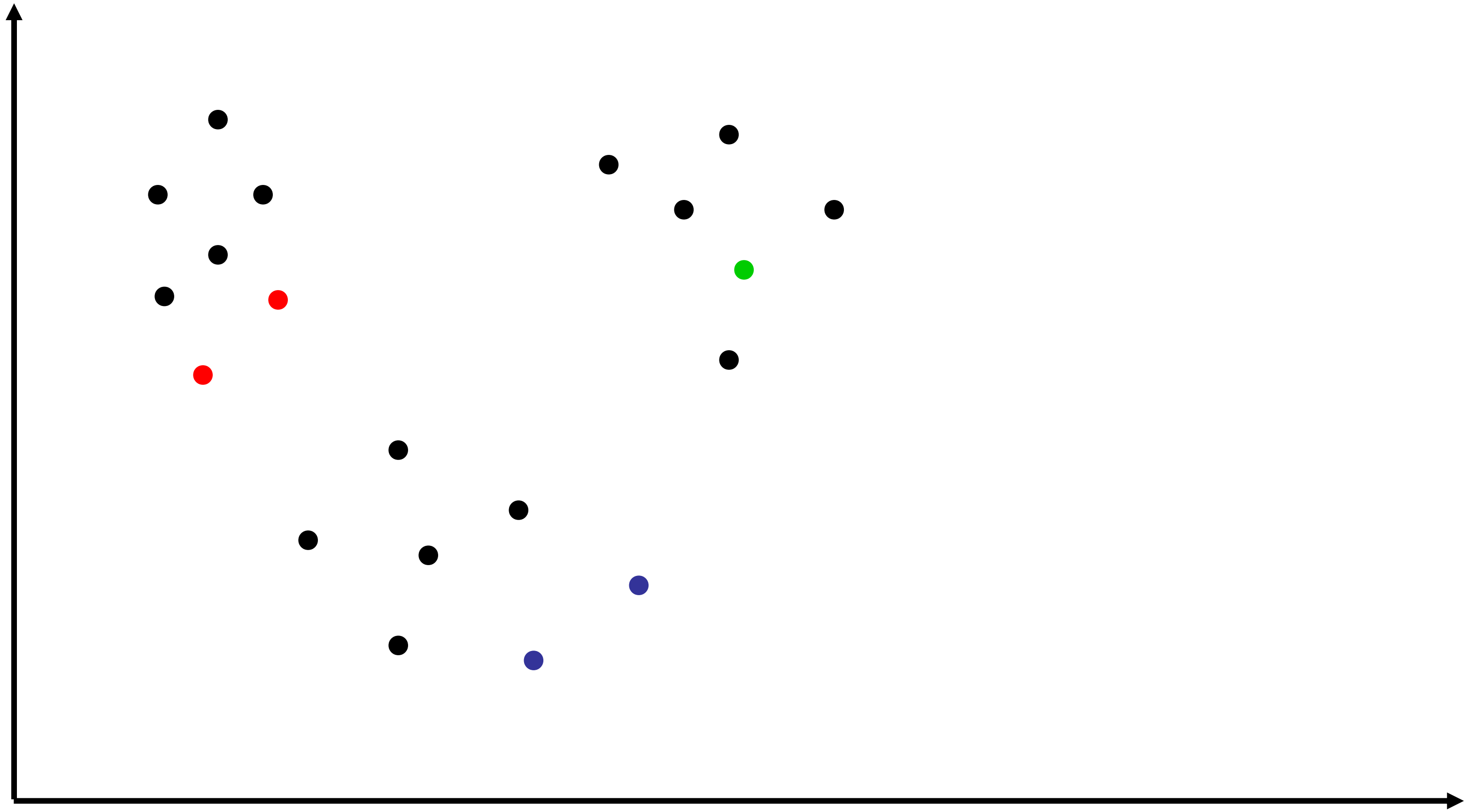
Semi-supervised Clustering - problem definition

- Input:
  - A set of unlabeled objects, each described by a set of attributes (numeric and/or categorical)
  - A small amount of domain knowledge
- Output:
  - A partitioning of the objects into k clusters (possibly with some discarded as outliers)
- Objective:
  - Maximum intra-cluster similarity
  - Minimum inter-cluster similarity
  - High consistency between the partitioning and the domain knowledge

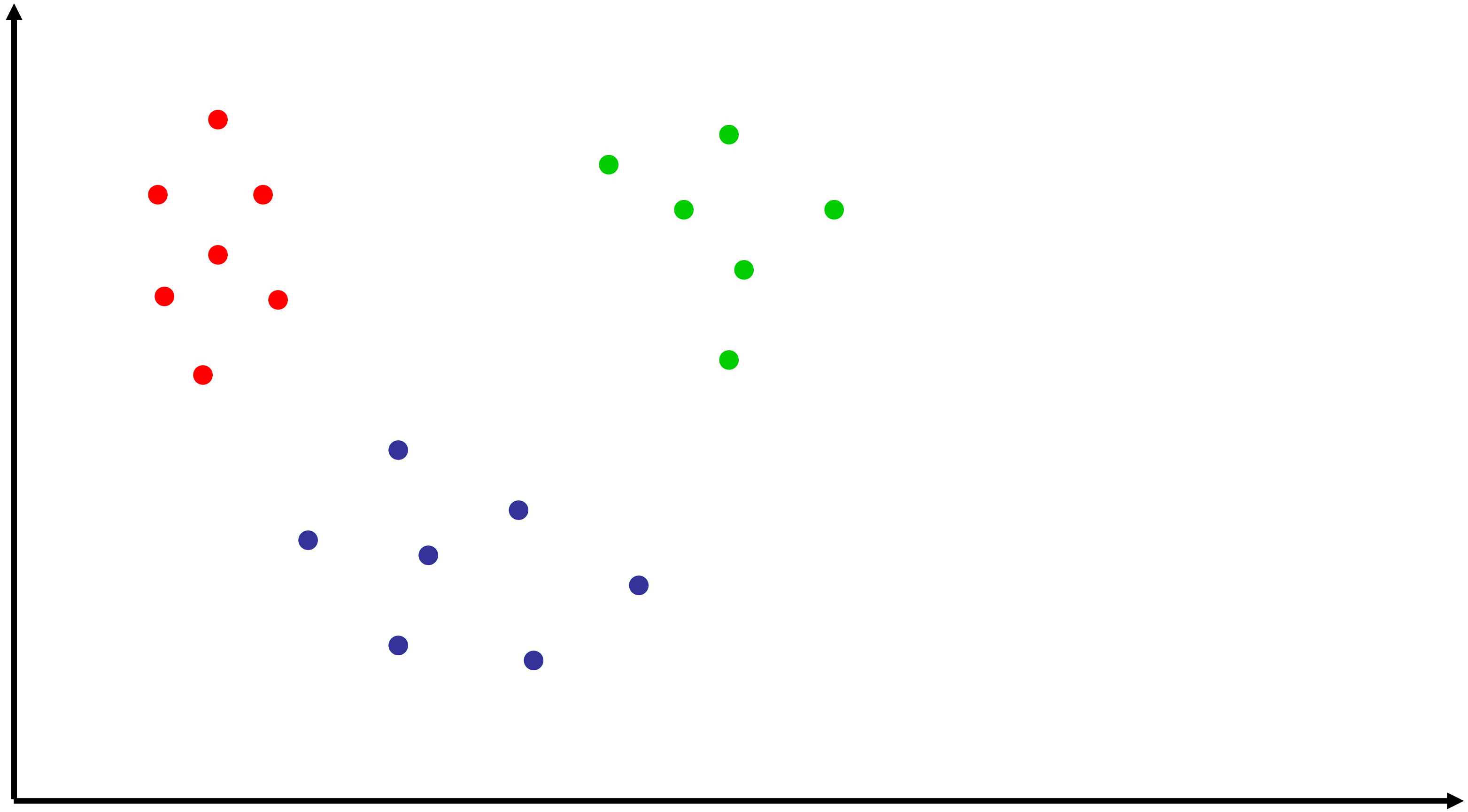
Why Semi-supervised Clustering

- Why not clustering?
  - The clusters produced may not be the ones required.
  - Sometimes there are multiple possible groupings.
- Why not classification?
  - Sometimes there are insufficient labeled data.
- Potential applications
  - Bioinformatics (gene and protein clustering)
  - Document hierarchy construction
  - News/email categorization
  - Image categorization

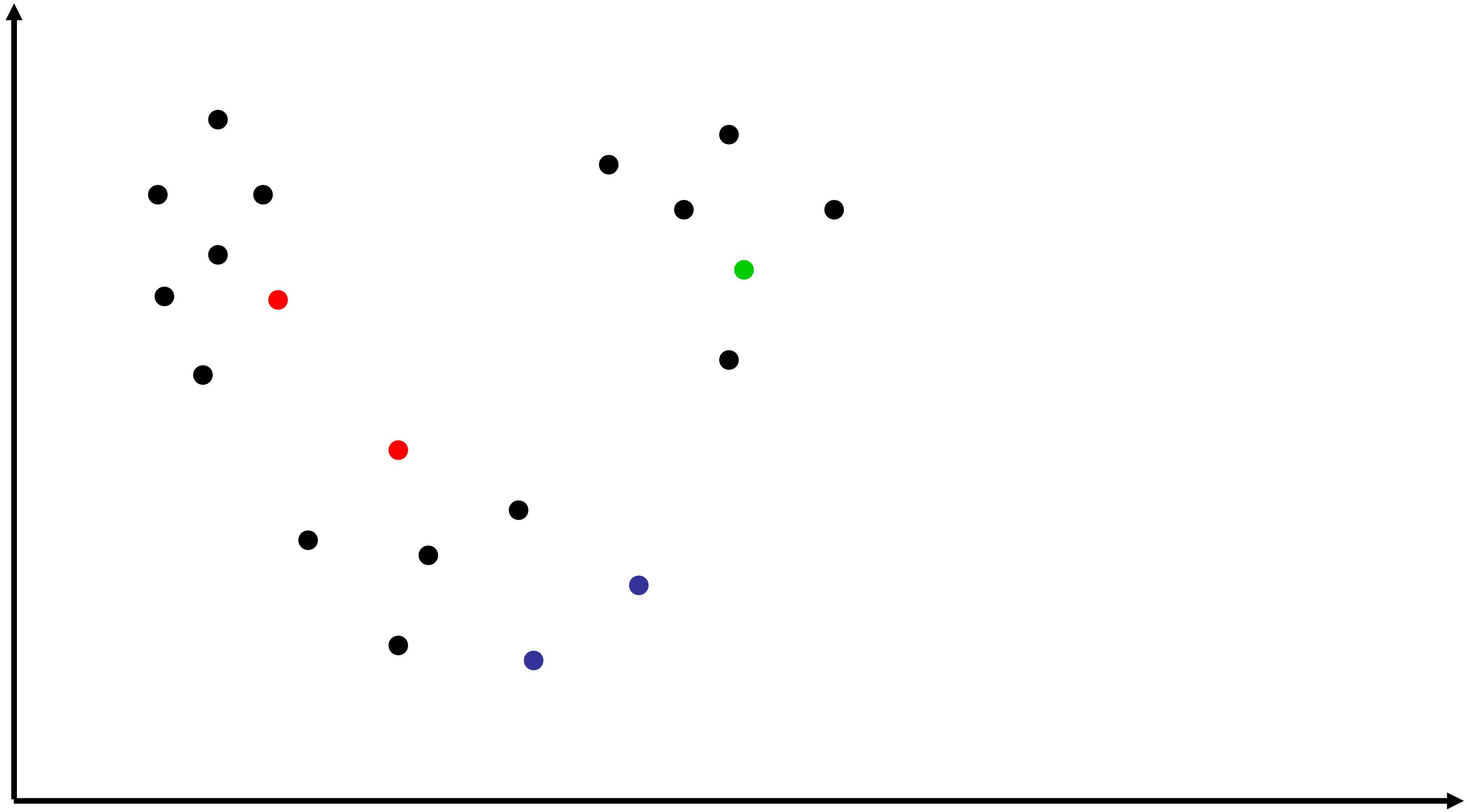
Semi-Supervised Clustering Example



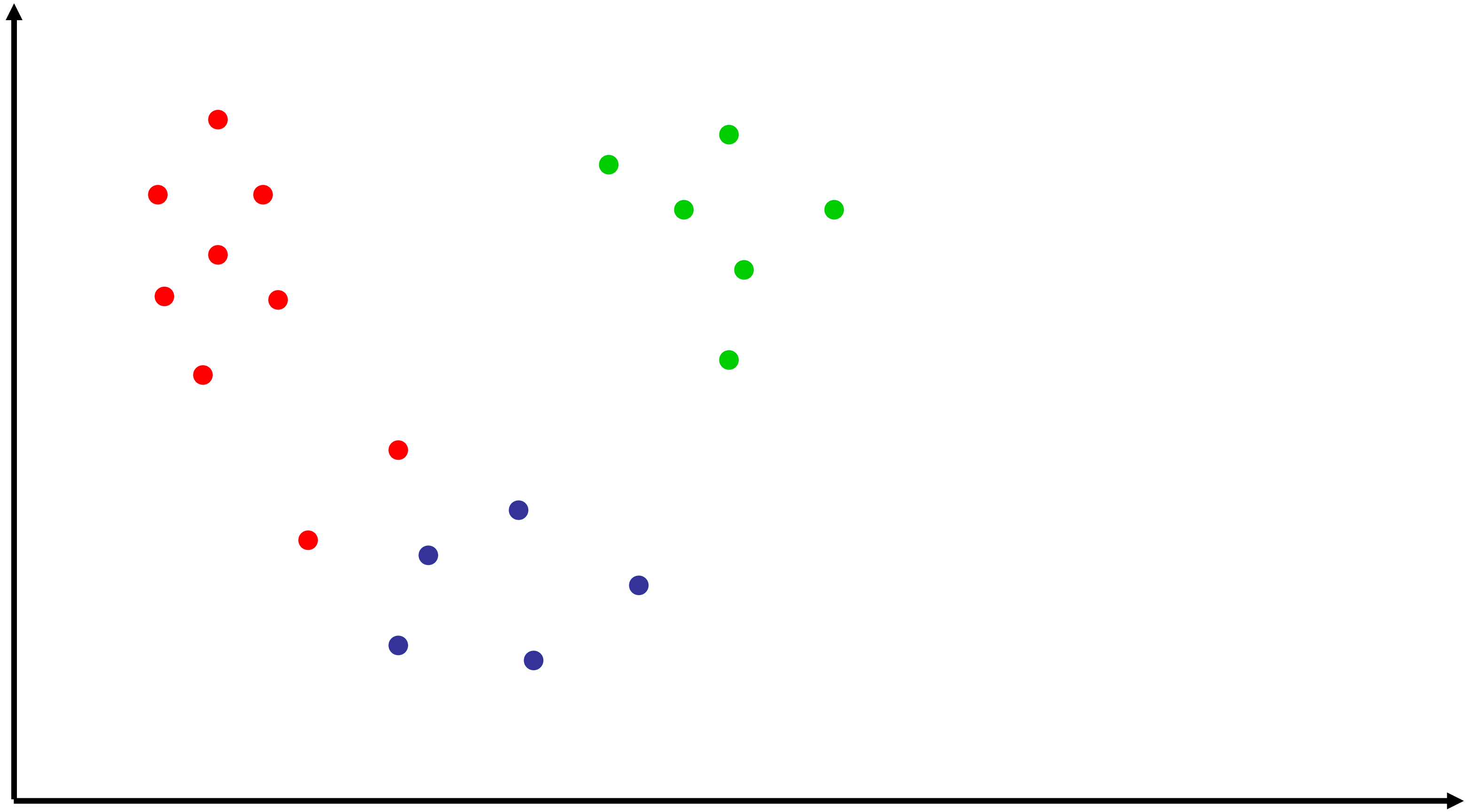
Semi-Supervised Clustering Example



Second Semi-Supervised Clustering Ex



Second Semi-Supervised Clustering Exa



Semi-supervised clustering: problem definition

- Input:
  - A set of unlabeled objects, each described by a set of attributes (numeric and/or categorical)
  - A small amount of domain knowledge
- Output:
  - A partitioning of the objects into k clusters (possibly with some discarded as outliers)
- Objective:
  - Maximum intra-cluster similarity
  - Minimum inter-cluster similarity
  - High consistency between the partitioning and the domain knowledge

Why semi-supervised clustering?

- Why not clustering?
  - The clusters produced may not be the ones required.
  - Sometimes there are multiple possible groupings.
- Why not classification?
  - Sometimes there are insufficient labeled data.
- Potential applications
  - Bioinformatics (gene and protein clustering)
  - Document hierarchy construction
  - News/email categorization

Semi-Supervised Clustering

- Domain knowledge
  - Partial label information is given
  - Apply some constraints (must-links and cannot-links)

- Approaches
  - Search-based Semi-Supervised Clustering
    - Alter the clustering algorithm using the constraints
  - Similarity-based Semi-Supervised Clustering
    - Alter the similarity measure based on the constraints
  - Combination of both

Search-Based Semi-Supervised Clustering

- Alter the clustering algorithm that searches for a good partitioning by:
  - Modifying the objective function to give a reward for obeying labels on the supervised data [Demeriz:ANNIE99].
  - Enforcing constraints (*must-link, cannot-link*) on the labeled data during clustering [Wagstaff:ICML00, Wagstaff:ICML01].
  - Use the labeled data to initialize clusters in an iterative refinement algorithm (kMeans,) [Basu:ICML02].

Overview of K-Means Clusterin

- K-Means is a partitional clustering algorithm based on iterative relocation that partitions a dataset into *K* clusters.

Algorithm:

Initialize *K* cluster centers $\{\mu_l\}_{l=1}^{K}$ randomly. Repeat until *convergence:*

- **Cluster Assignment Step**: Assign each data point *x* to the cluster X_l, such that L_2 distance of *x* from μ_l (center of X_l) is minimum
- **Center Re-estimation Step**: Re-estimate each cluster center μ_l

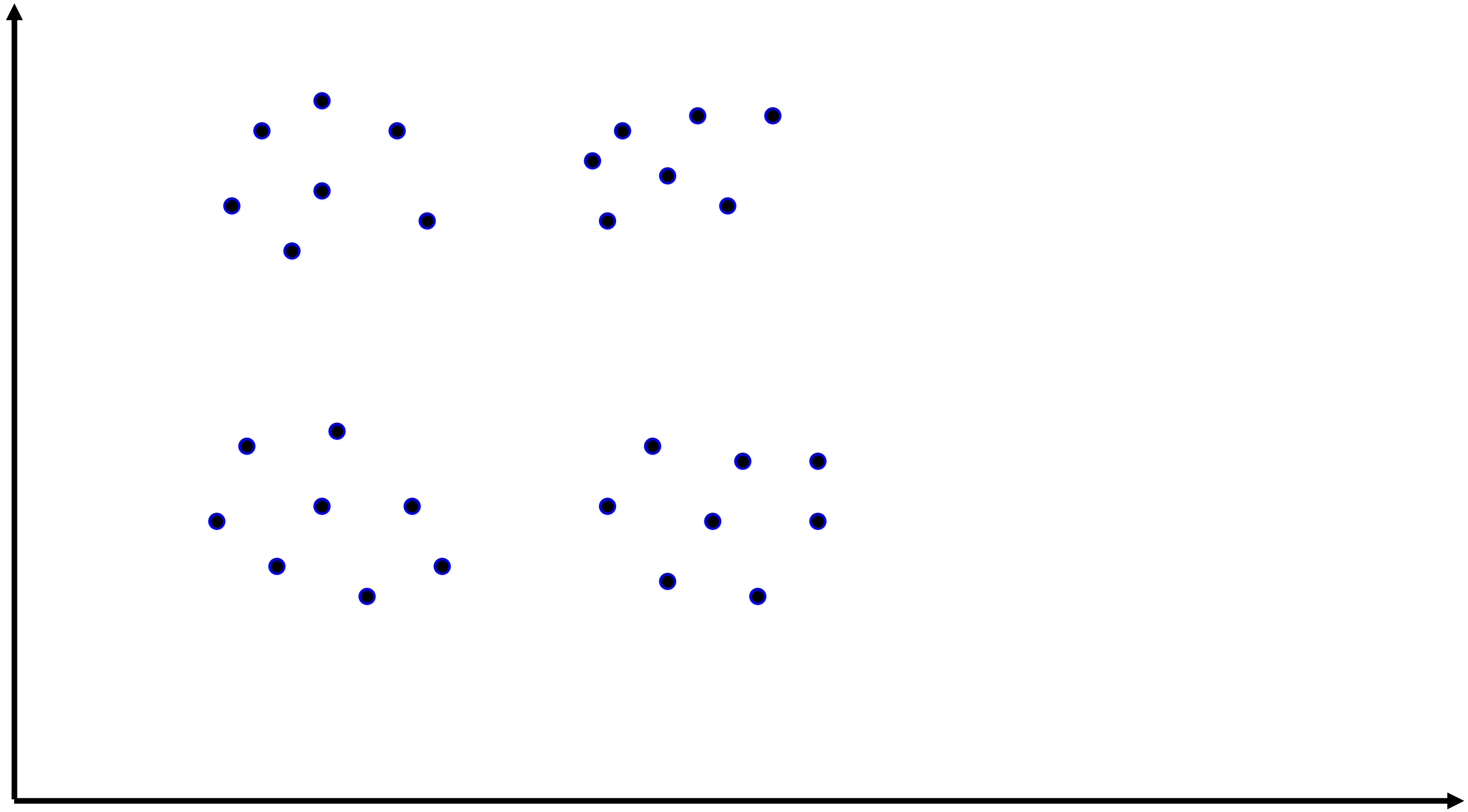
K-Means Objective Function

- Locally minimizes sum of squared distance between the data points and their corresponding cluster centers:

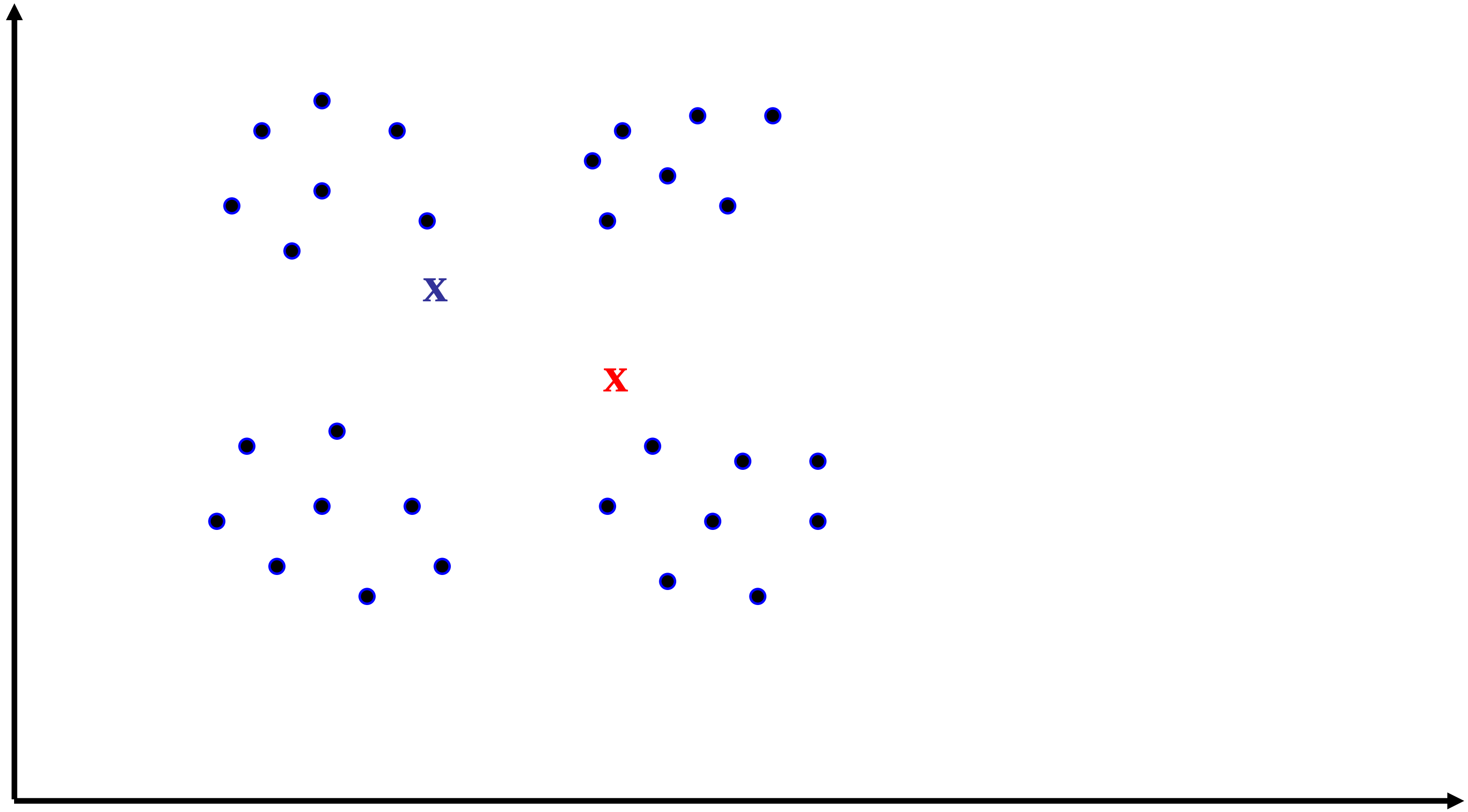
$$\sum_{l=1}^{K} \sum_{x_i \in X_l} \| x_i - \mu_l \|^2$$

- Initialization of K cluster centers:
  - Totally random
  - Random perturbation from global mean
  - Heuristic to ensure well-separated centers

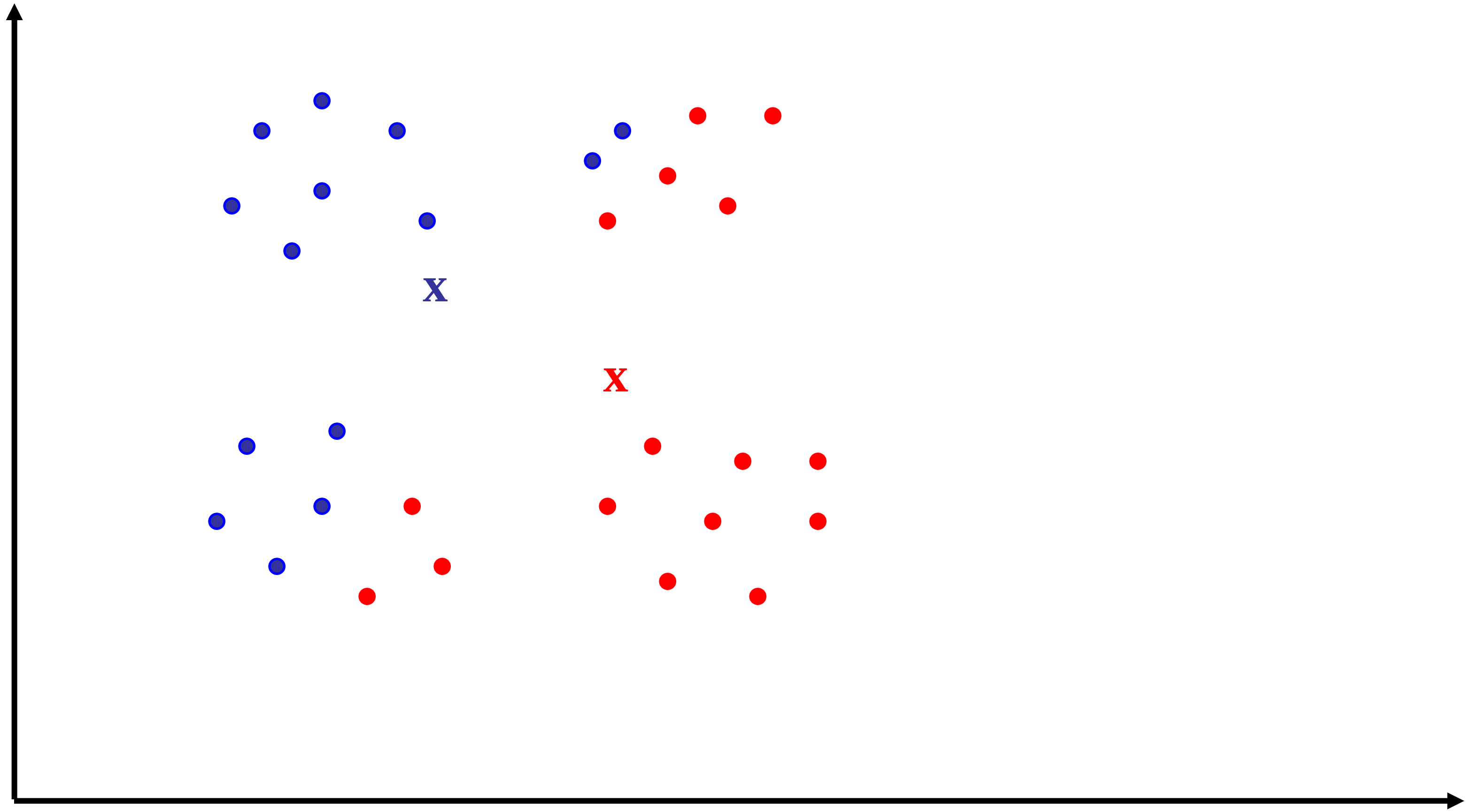
K Means Example



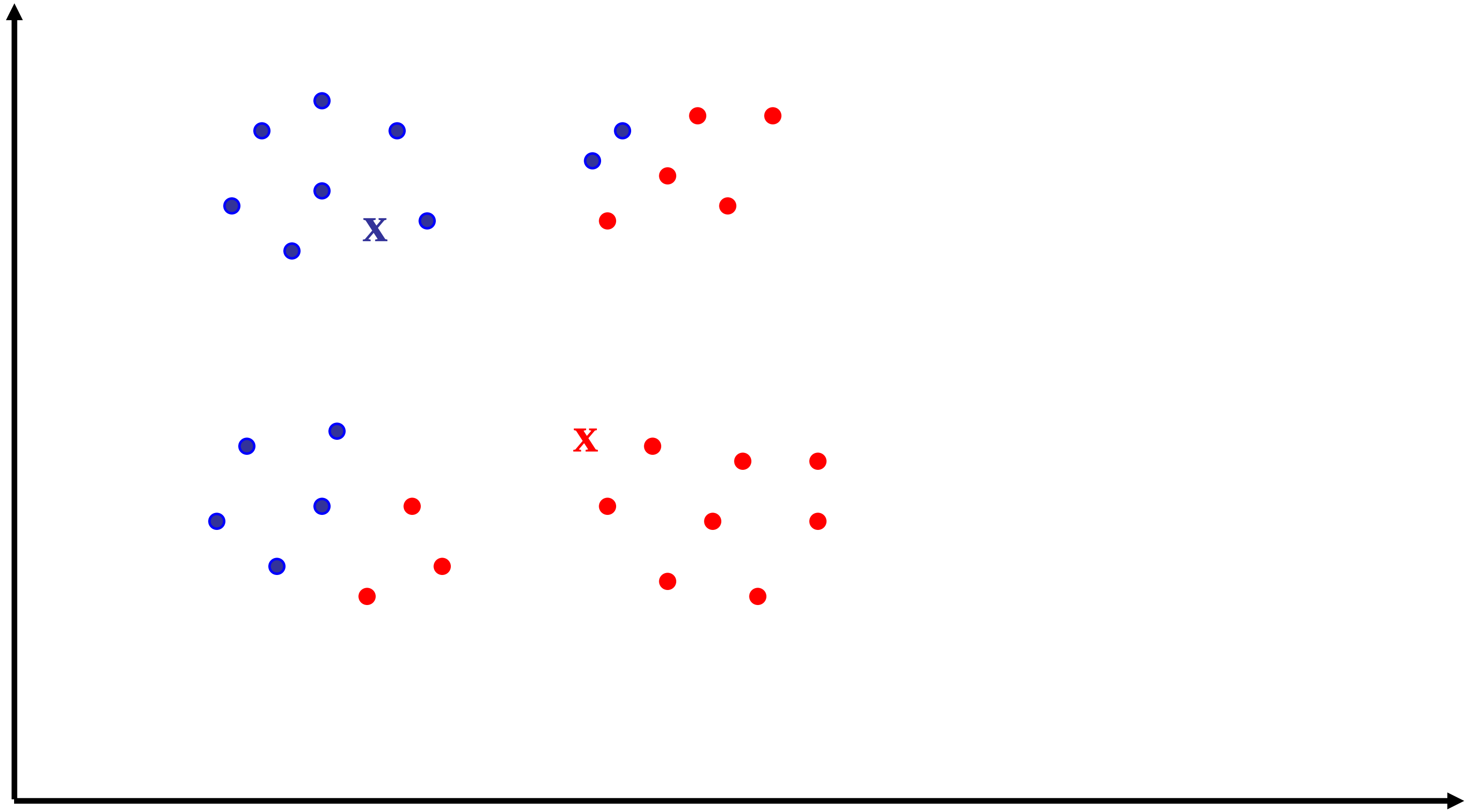
K Means Example
Randomly Initialize Means



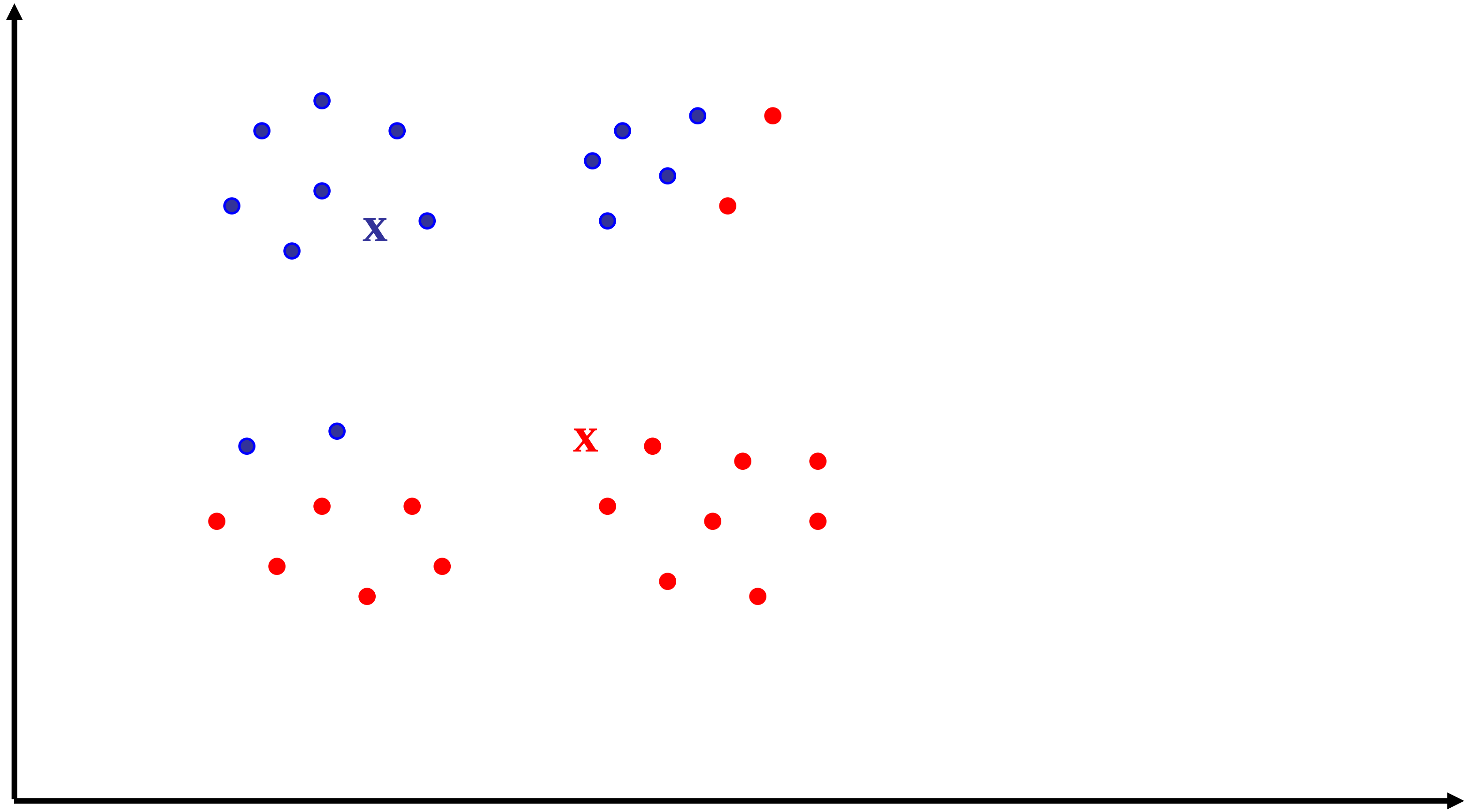
K Means Example
Assign Points to Clusters



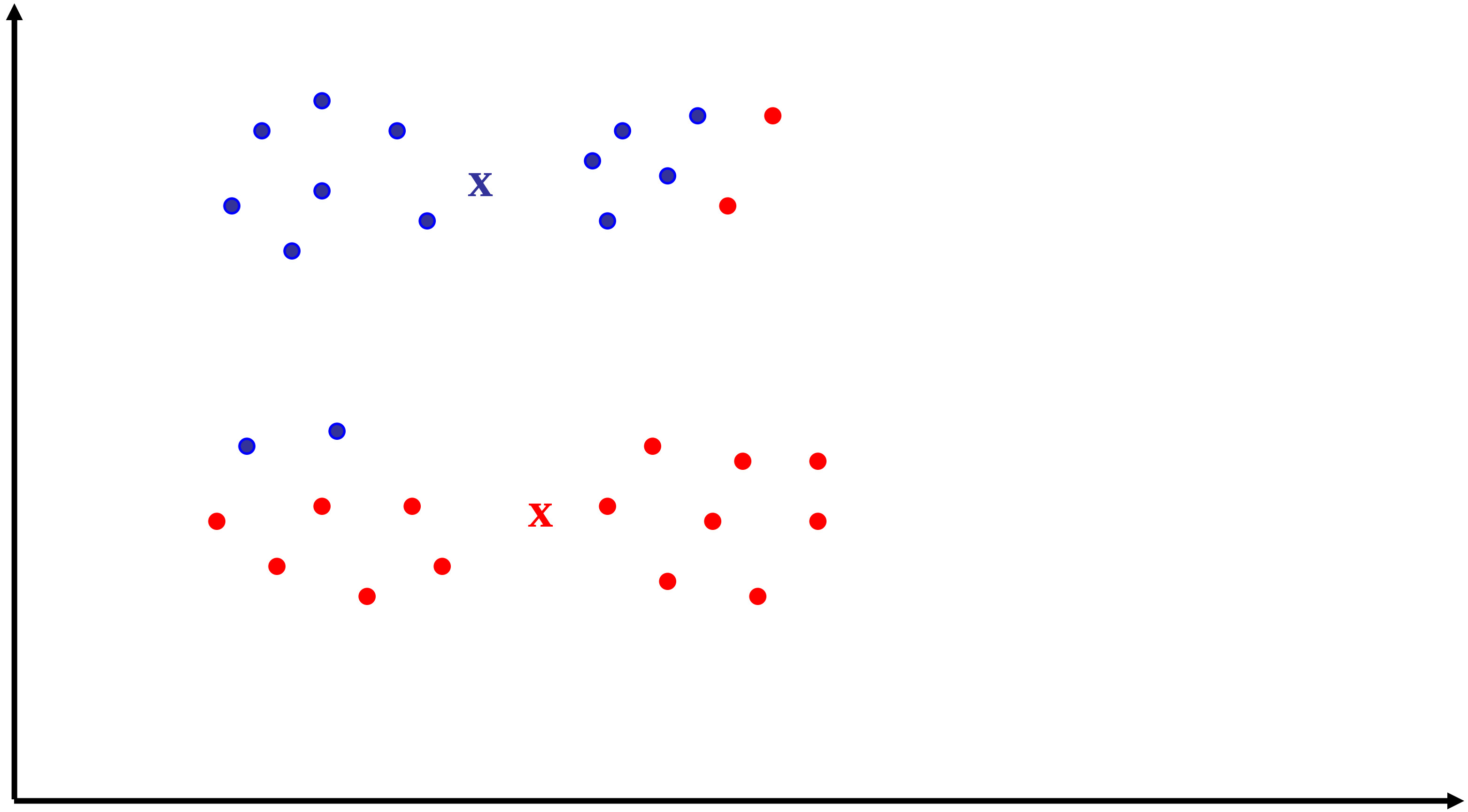
K Means Example
Re-estimate Means



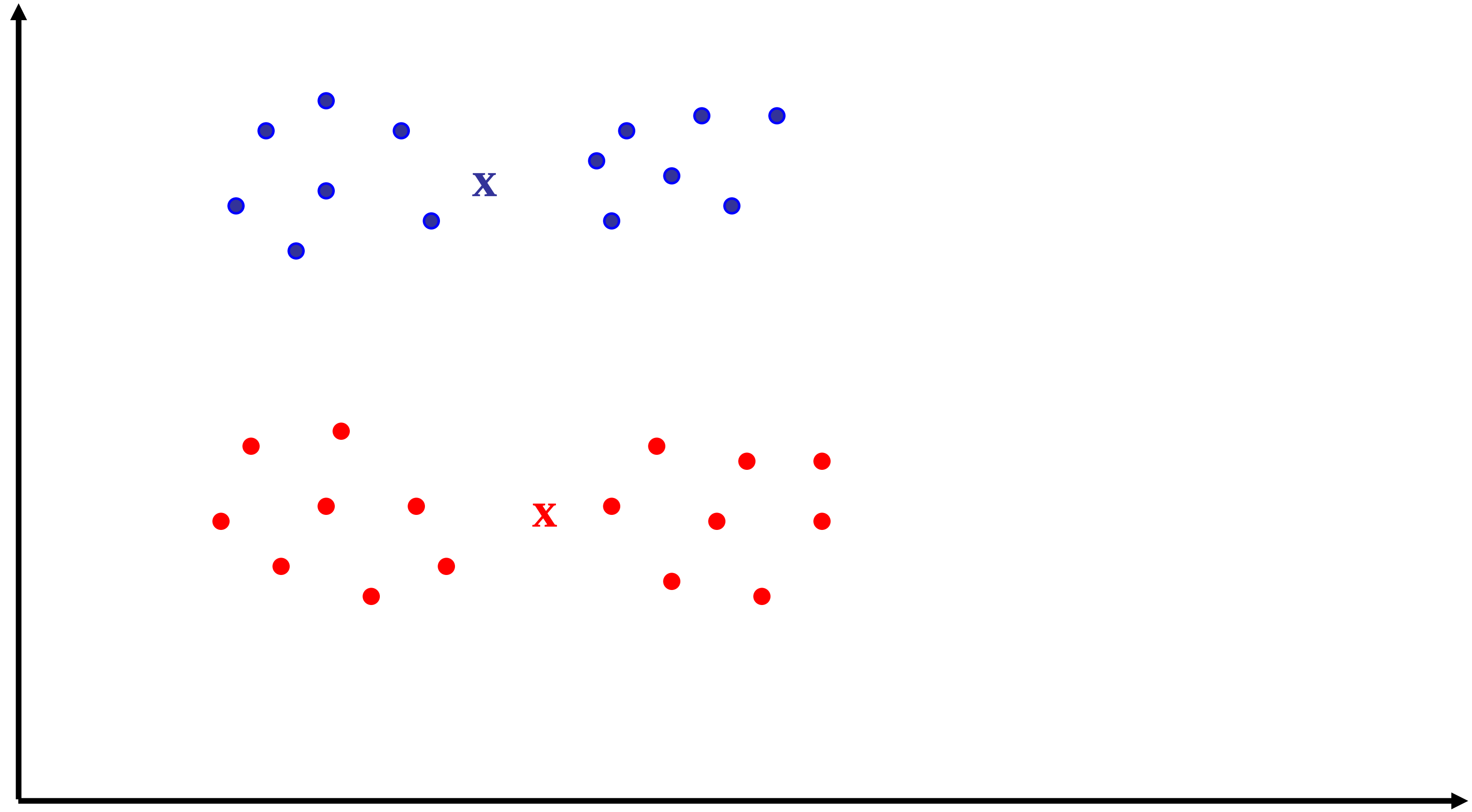
K Means Example
Re-assign Points to Clusters



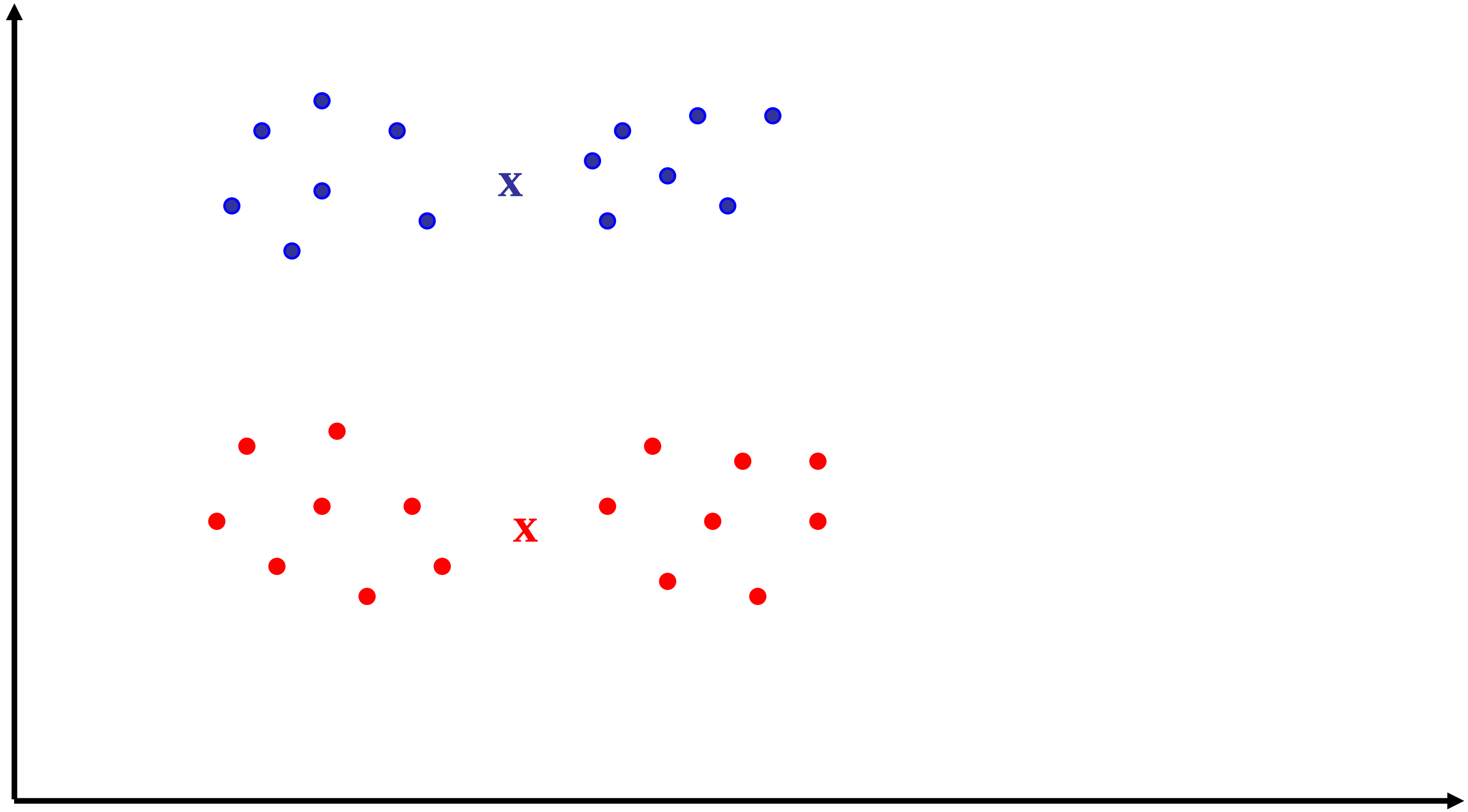
K Means Example
Re-estimate Means



K Means Example
Re-assign Points to Clusters



K Means Example
Re-estimate Means and Converge



Semi-Supervised K-Means

- Partial label information is given
  - Seeded K-Means
  - Constrained K-Means

- Constraints (Must-link, Cannot-link)
  - COP K-Means

Semi-Supervised K-Means for p labeled data

- **Seeded K-Means:**
  - Labeled data provided by user are used for initialization: initial center for cluster *i* is the mean of the seed points having label *i.*
  - Seed points are only used for initialization, and not in subsequent steps.
- **Constrained K-Means:**
  - Labeled data provided by user are used to initialize K-Means algorithm.
  - Cluster labels of seed data are kept unchanged in the cluster assignment steps, and only the labels of the non-seed data are re-estimated.

Seeded K-Means

Algorithm: Seeded-KMeans

Input: Set of data points $\mathcal{X} = \{x_1, \cdots, x_N\}, x_i \in \mathbb{R}^d$, number of clusters K, set $\mathcal{S} = \cup_{l=1}^{K} \mathcal{S}_l$ of initial seeds

Output: Disjoint K partitioning $\{\mathcal{X}_l\}_{l=1}^{K}$ of $\mathcal{X}$ such that KMeans objective function is optimized

Method:

1. intialize: $\mu_h^{(0)} \leftarrow \frac{1}{|\mathcal{S}_h|} \sum_{x \in \mathcal{S}_h} x$, for $h = 1, \ldots, K; t \leftarrow 0$
2. Repeat until *convergence*

2a. assign_cluster: Assign each data point x to the cluster h^* (i.e. set $\mathcal{X}_{h^*}^{(t+1)}$), for $h^* = \arg\min_h \|x - \mu_h^{(t)}\|^2$

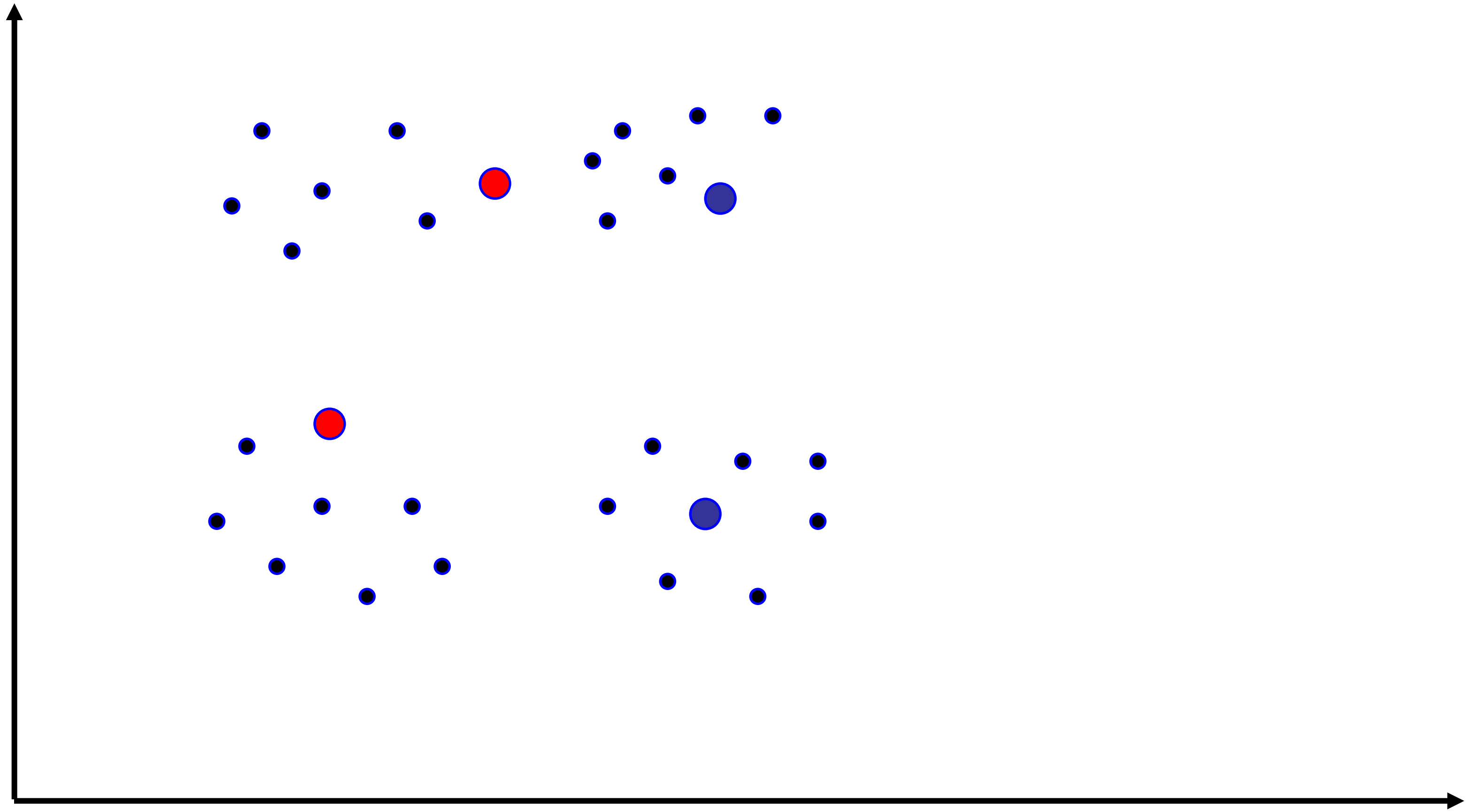
2b. estimate_means: $\mu_h^{(t+1)} \leftarrow \frac{1}{|\mathcal{X}_h^{(t+1)}|} \sum_{x \in \mathcal{X}_h^{(t+1)}} x$

2c. $t \leftarrow (t+1)$

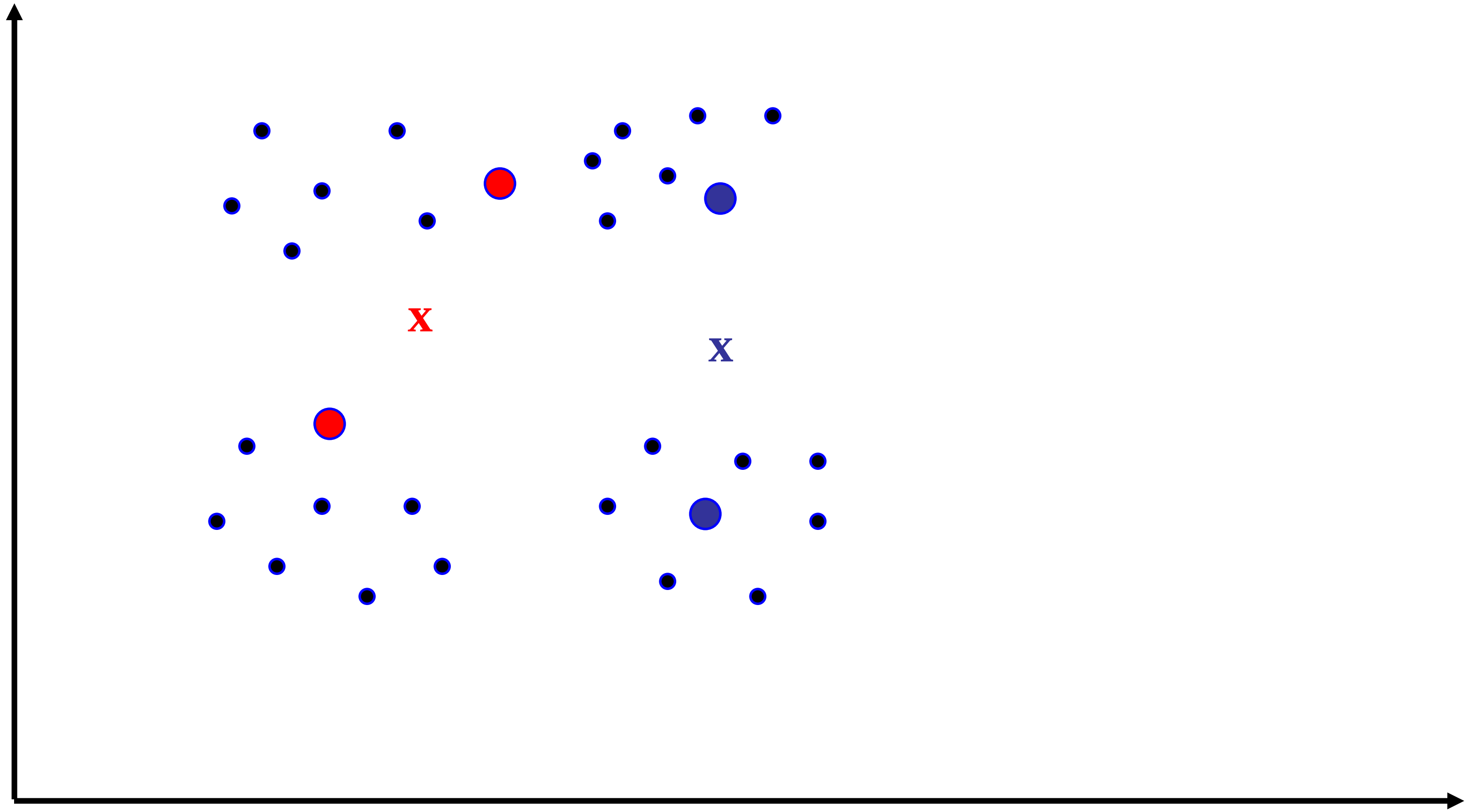
Use labeled data to find the initial centroids and then run K-Means.

The labels for seeded points may change.

Seeded K-Means Example

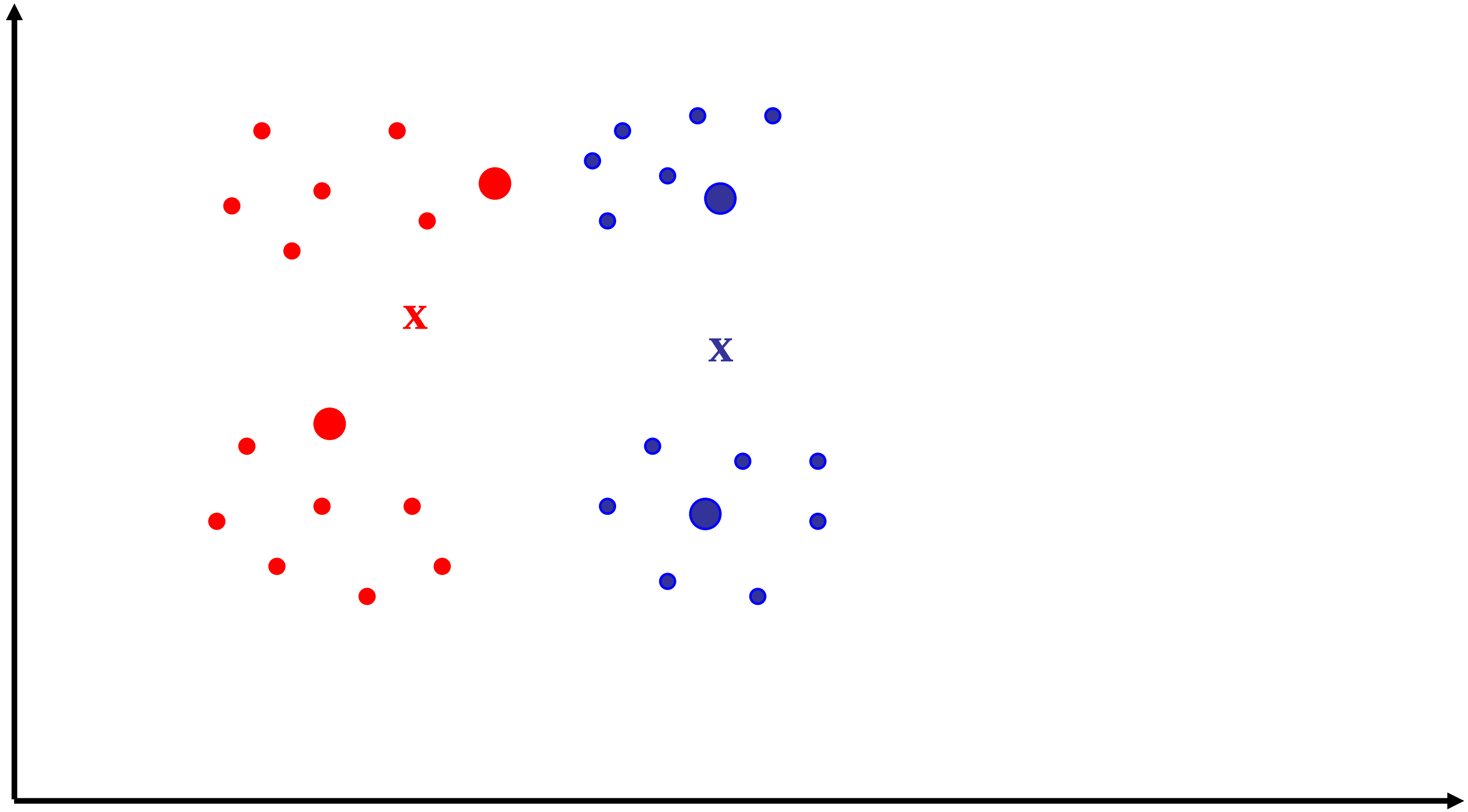


Seeded K-Means Example
Initialize Means Using Labeled Data

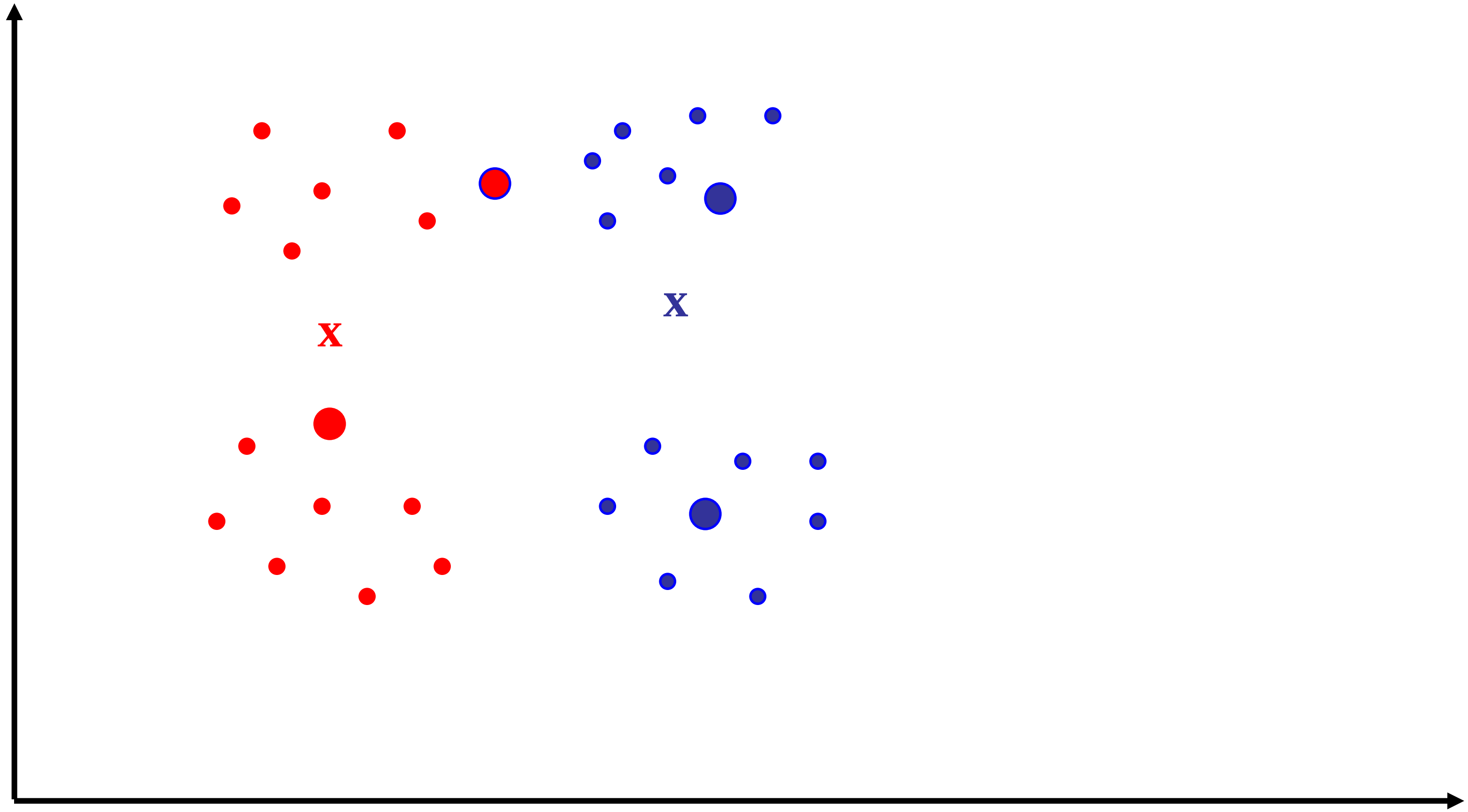


Seeded K-Means Example

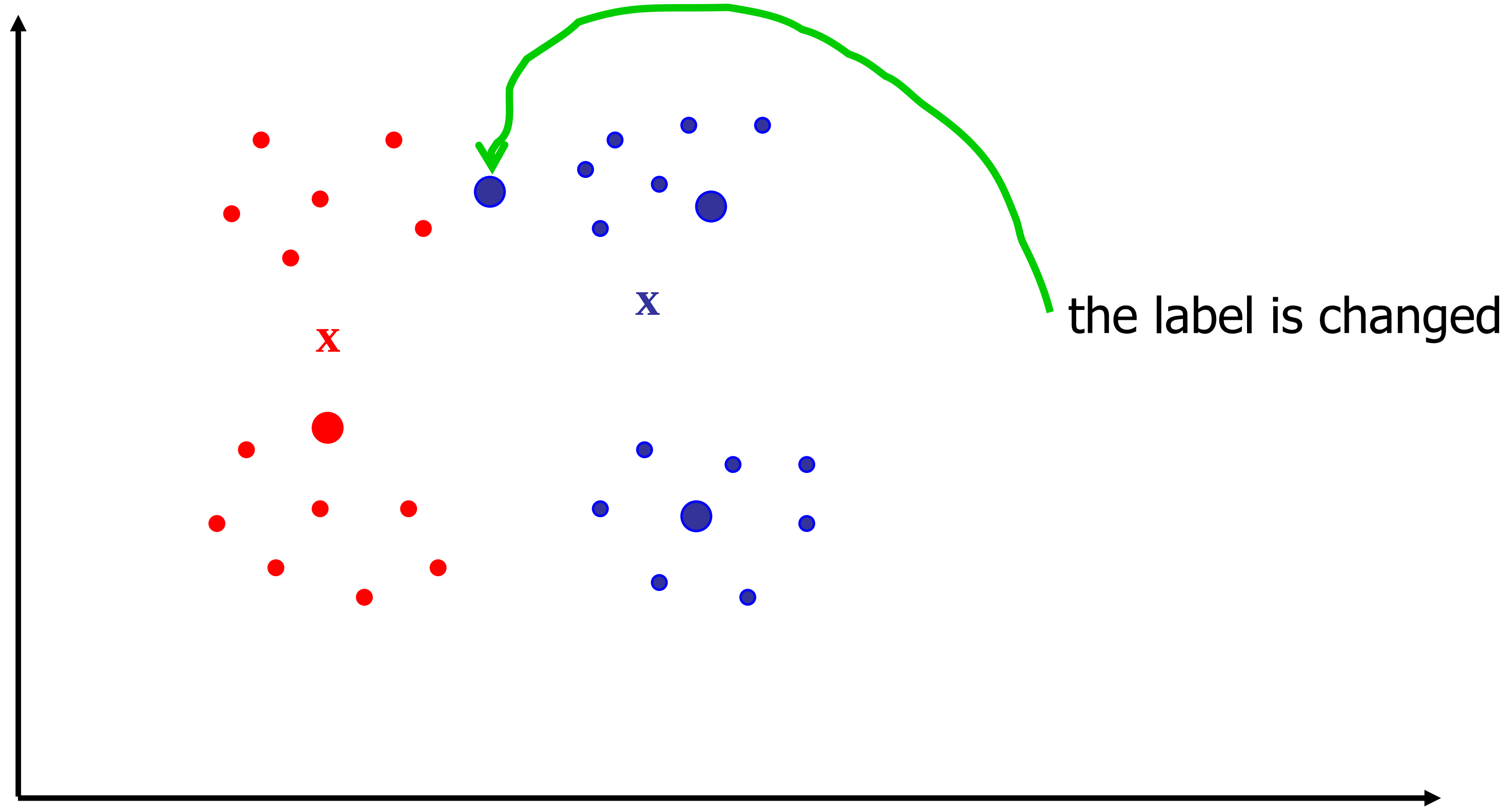
Assign Points to Clusters



Seeded K-Means Example
Re-estimate Means



Seeded K-Means Example
Assign points to clusters and Converge



Constrained K-Means

Algorithm: `Constrained-KMeans`

Input: Set of data points $\mathcal{X} = \{x_1, \cdots, x_N\}, x_i \in \mathbb{R}^d$, number of clusters K, set $\mathcal{S} = \cup_{l=1}^{K} \mathcal{S}_l$ of initial seeds

Output: Disjoint K partitioning $\{\mathcal{X}_l\}_{l=1}^{K}$ of $\mathcal{X}$ such that the KMeans objective function is optimized

Method:

1. `intialize`: $\mu_h^{(0)} \leftarrow \frac{1}{|\mathcal{S}_h|} \sum_{x \in \mathcal{S}_h} x$, for $h = 1, \ldots, K; t \leftarrow 0$
2. Repeat until *convergence*

2a. `assign_cluster`: For $x \in \mathcal{S}$, if $x \in \mathcal{S}_h$ assign x to the cluster h (i.e., set $\mathcal{X}_h^{(t+1)}$). For $x \notin \mathcal{S}$, assign x to the cluster h^* (i.e. set $\mathcal{X}_{h^*}^{(t+1)}$), for $h^* = \arg\min_h \|x - \mu_h^{(t)}\|^2$

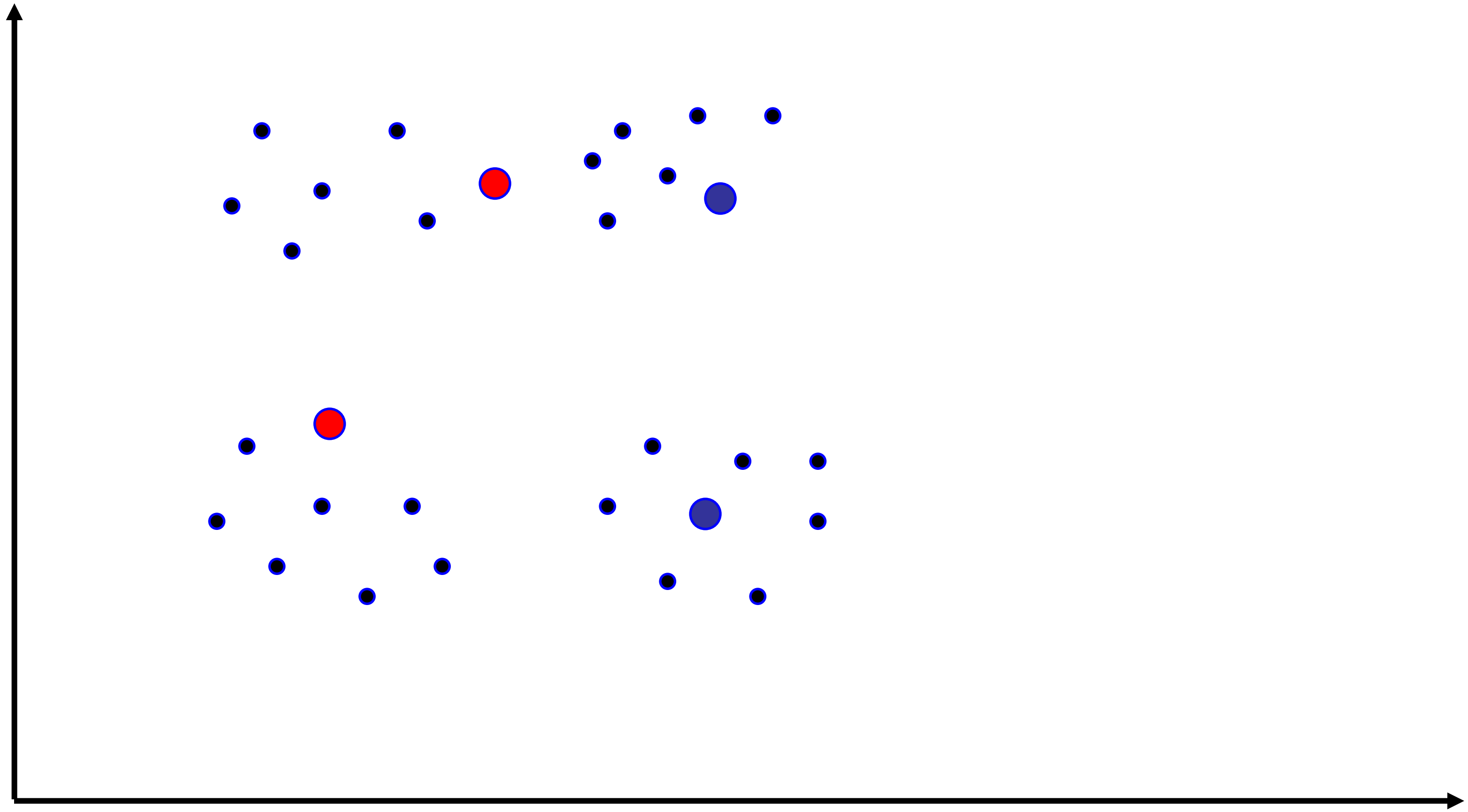
2b. `estimate_means`: $\mu_h^{(t+1)} \leftarrow \frac{1}{|\mathcal{X}_h^{(t+1)}|} \sum_{x \in \mathcal{X}_h^{(t+1)}} x$

2c. $t \leftarrow (t + 1)$

Use labeled data to find the initial centroids and then run K-Means.

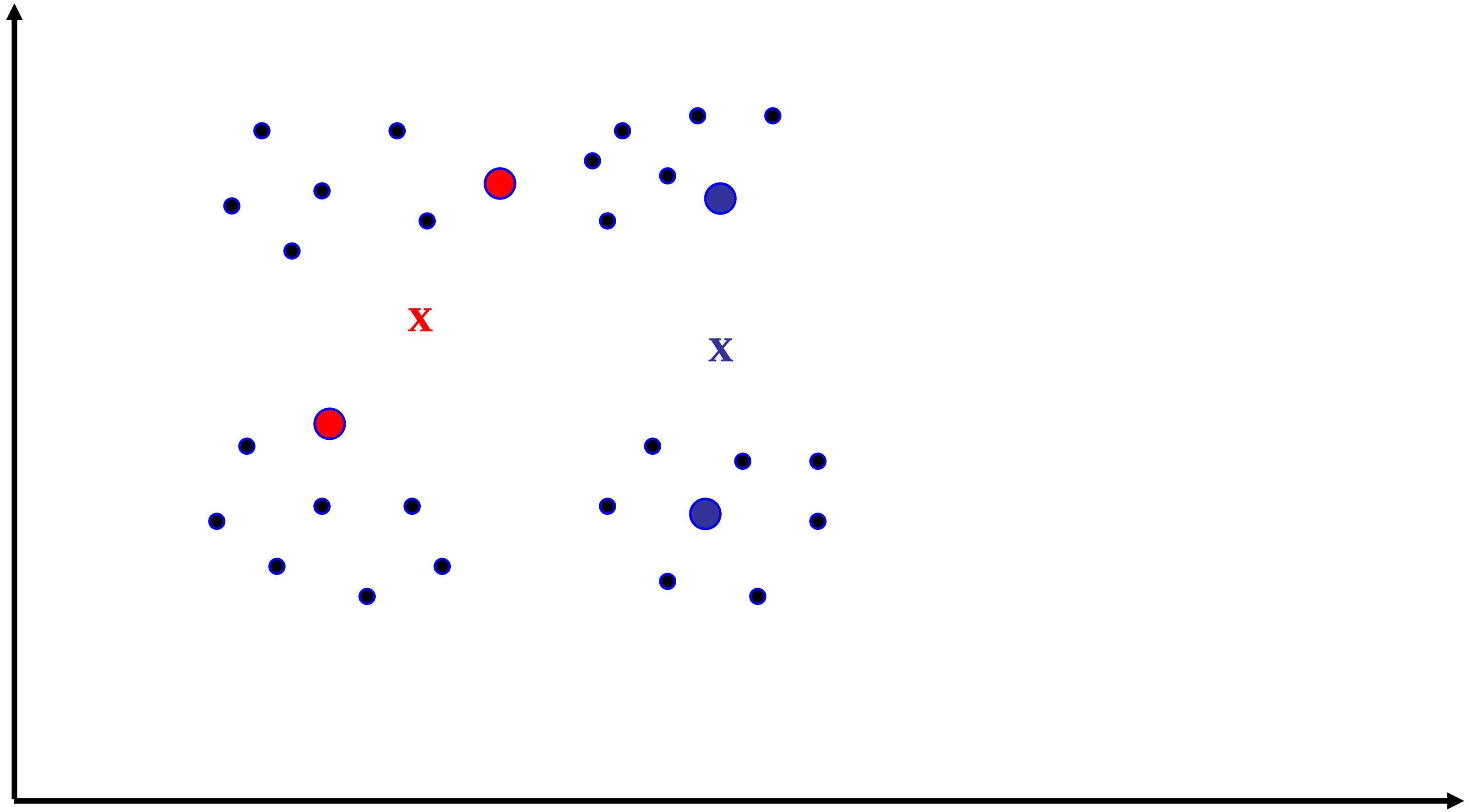
The labels for seeded points will not change.

Constrained K-Means Example



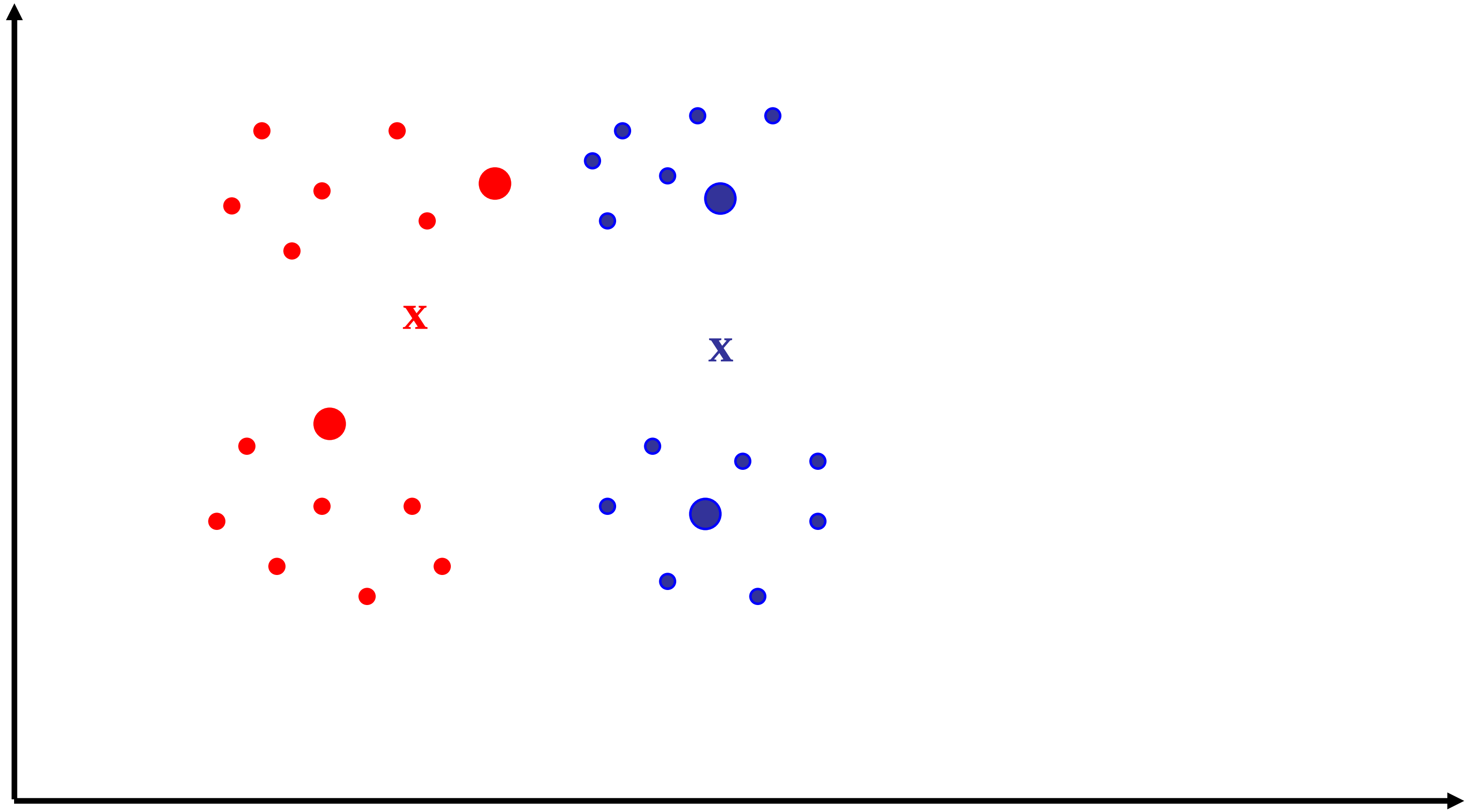
Constrained K-Means Example

Initialize Means Using Labeled Data



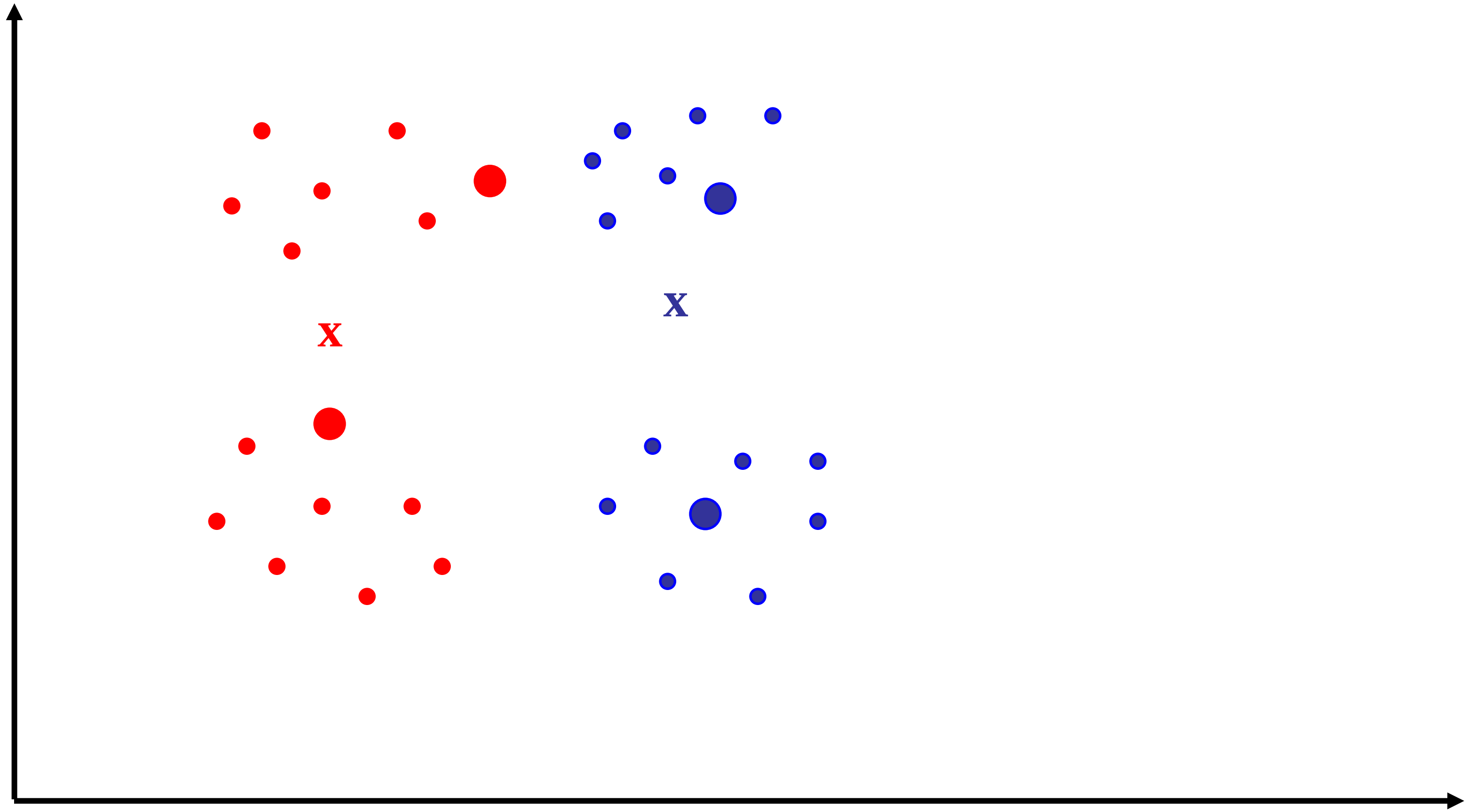
Constrained K-Means Example

Assign Points to Clusters



Constrained K-Means Example

Re-estimate Means and Converge



COP K-Means

- COP K-Means [Wagstaff *et al*.: ICML01] is K-Means with must-link (must be in same cluster) and cannot-link (cannot be in same cluster) constraints on data points.
- **Initialization***:* Cluster centers are chosen randomly, but as each one is chosen any must-link constraints that it participates in are enforced (so that they cannot later be chosen as the center of another cluster).
- **Algorithm***:* During cluster assignment step in COP-K-Means, a point is assigned to its nearest cluster without violating any of its constraints. If no such assignment exists, abort.

COP K-Means Algorithm

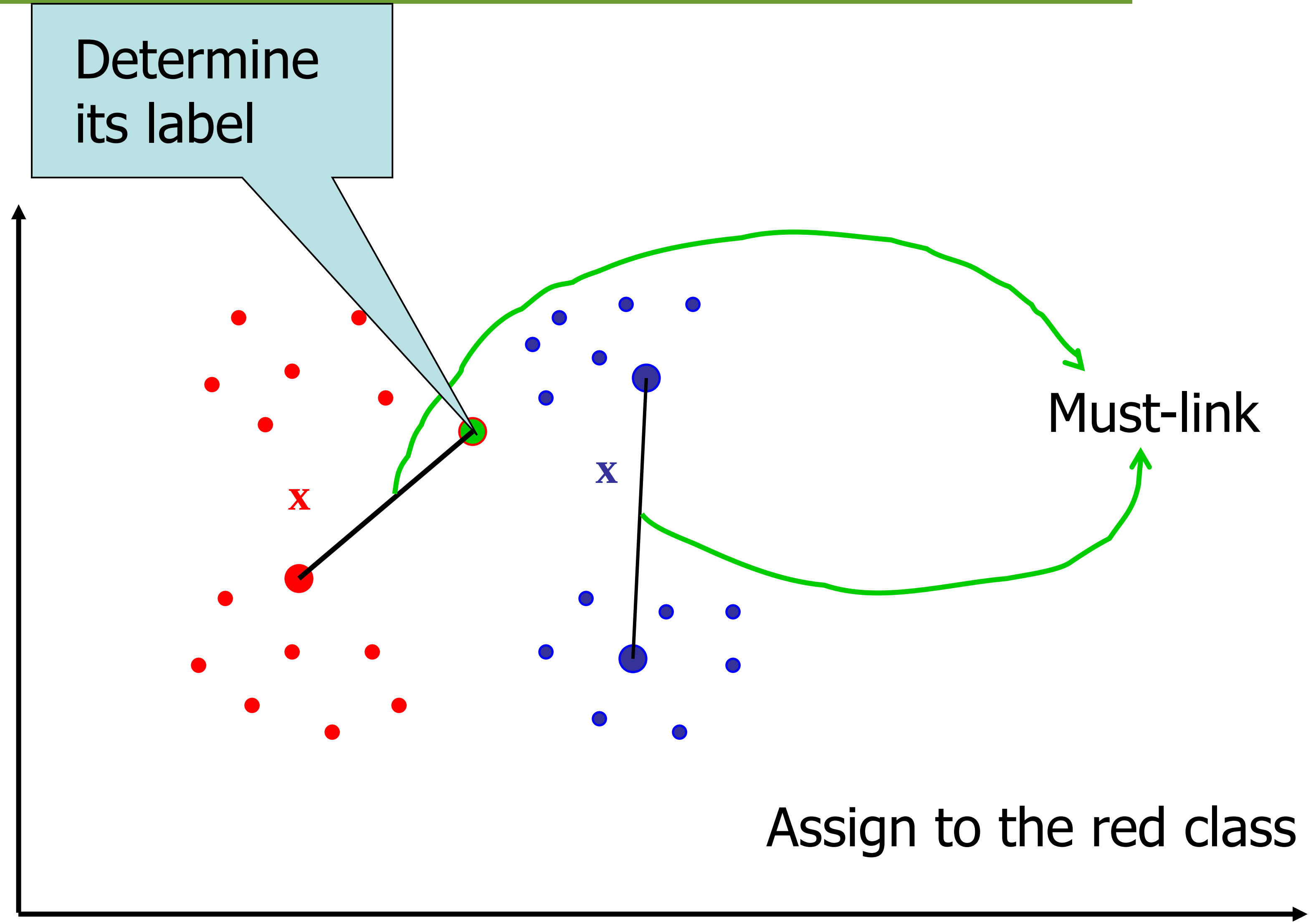
COP-KMEANS(data set D, must-link constraints $Con_= \subseteq D \times D$, cannot-link constraints $Con_{\neq} \subseteq D \times D$)

1. Let $C_1 \ldots C_k$ be the initial cluster centers.
2. For each point d_i in D, assign it to the closest cluster C_j **such that** VIOLATE-CONSTRAINTS(d_i, C_j, $Con_=$, $Con_{\neq}$) **is false. If no such cluster exists, fail (return {})**.
3. For each cluster C_i, update its center by averaging all of the points d_j that have been assigned to it.
4. Iterate between (2) and (3) until convergence.
5. Return $\{C_1 \ldots C_k\}$.

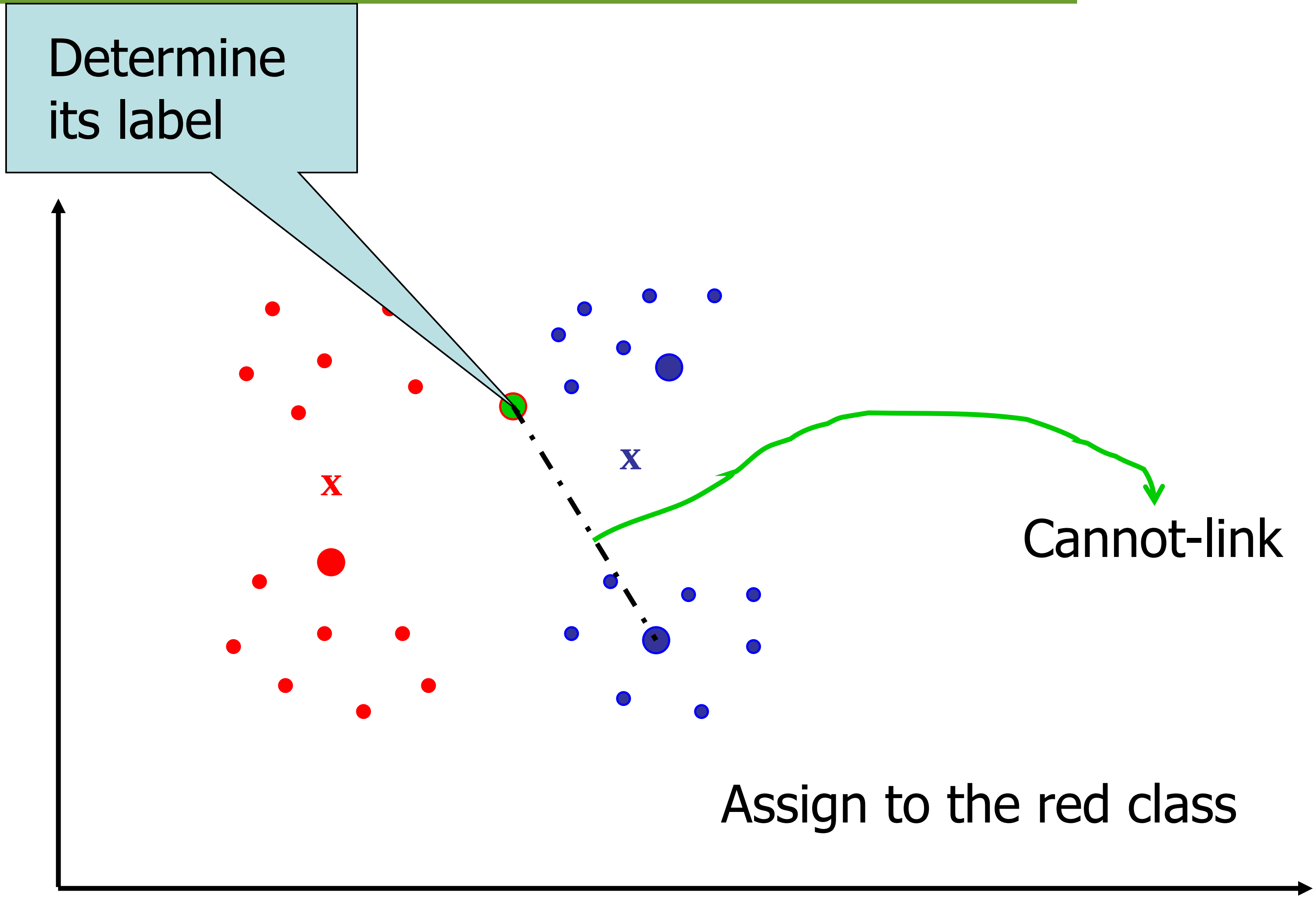
VIOLATE-CONSTRAINTS(data point d, cluster C, must-link constraints $Con_= \subseteq D \times D$, cannot-link constraints $Con_{\neq} \subseteq D \times D$)

1. For each $(d, d_=) \in Con_=$: If $d_= \notin C$, return true.
2. For each $(d, d_{\neq}) \in Con_{\neq}$: If $d_{\neq} \in C$, return true.
3. Otherwise, return false.

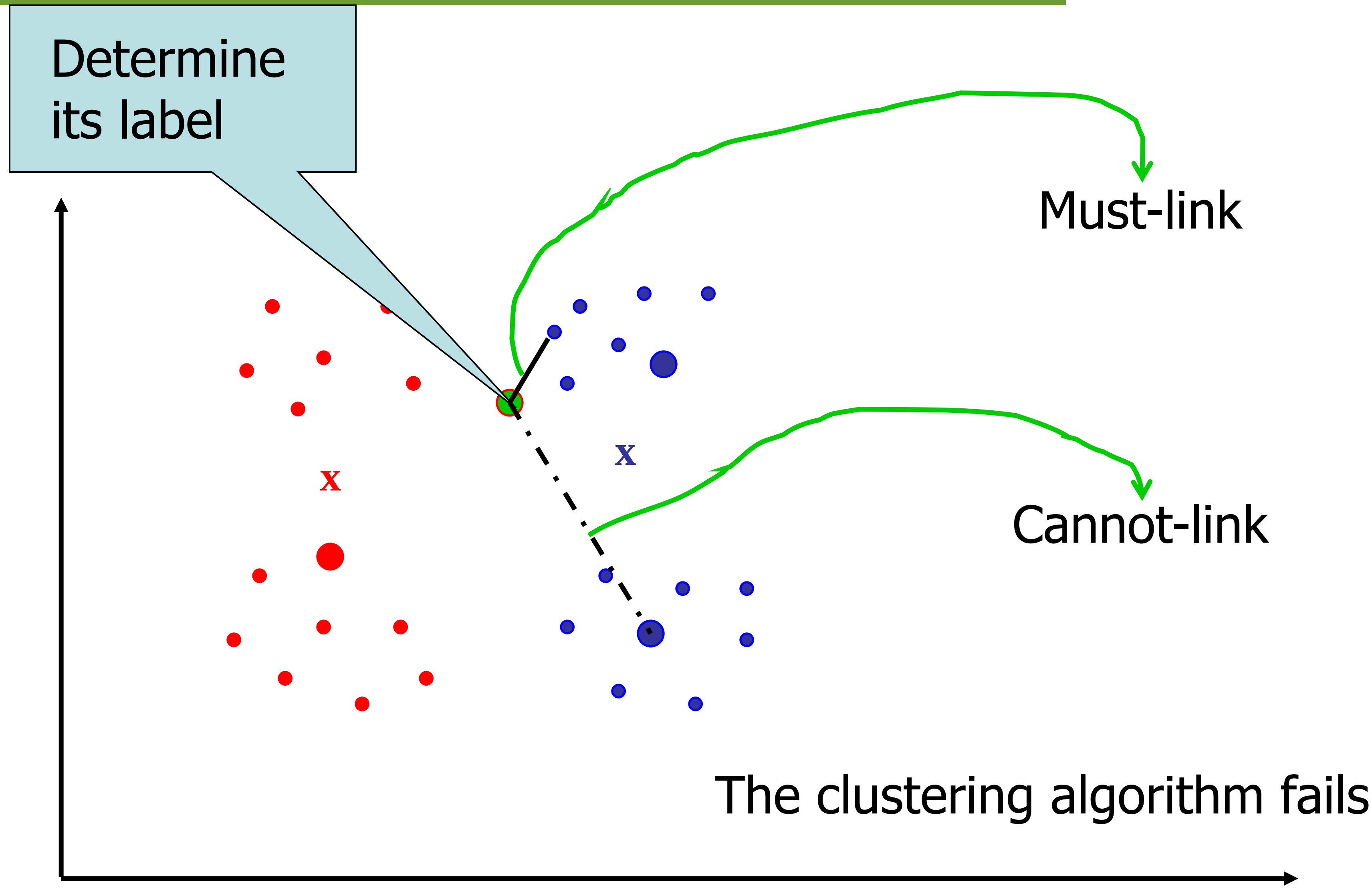
Illustration



Illustration



Illustration



Similarity-based semi-supervised clustering

- Alter the similarity measure based on the constraints
- Paper: From Instance-level Constraints to Space-Level Constraints: Making the Most of Prior Knowledge in Data Clustering. D. Klein *et al.*

Two types of constraints: Must-links and Cannot-links

Clustering algorithm: Hierarchical clustering

Constraints

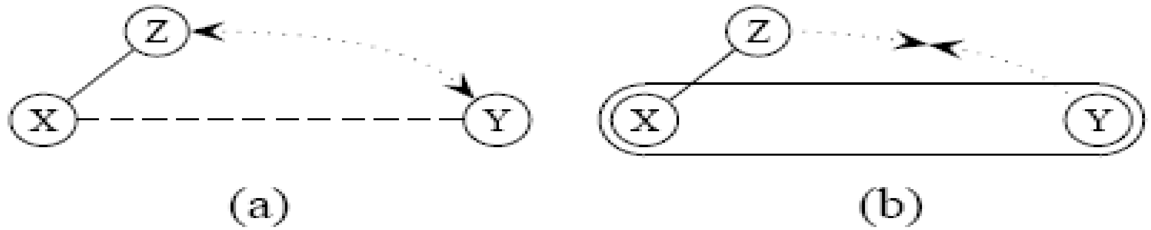


Figure 3. Constrained pairs have implications for nearby points. If X and Z are very close, then (a) constraining X away from Y should push Z from Y and (b) constraining X towards Y should pull Z towards Y.

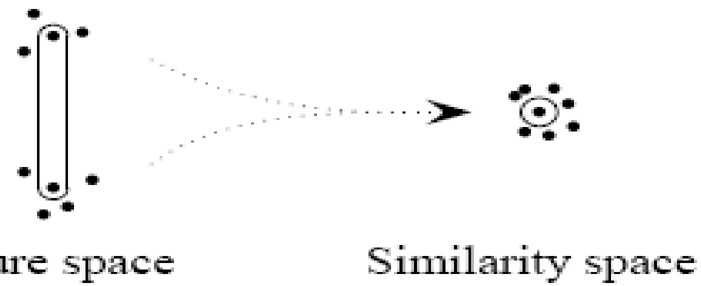


Figure 4. Clusters which are distant in feature space can be brought together in similarity space with a propagated must-link constraint.

Transductive Learning

- Transductive learning is a particular case of semi-supervised learning
  - Because it allows the learning algorithm to exploit the unlabeled examples in the test set.
- The learning algorithm does not necessarily have to learn a general rule, but it only needs to predict accurately for a finite number of **test examples**.
- **The test examples are known a priori and can be observed by the learning algorithm during training**.
- This allows the learning algorithm to exploit any information that might be contained in the location of the test examples.

Transductive Learning

- Given is a set S = {1,2,…,n}, that enumerates all n possible examples.
- Assume, each object in the dataset is used to represent collection of documents
- It would be one index-i for each document in the collection.
- Assume that each example-i is represented by a feature vector $x_i \in R^d$.
- For text documents, this could be a TFIDF vector representation, where each document is represented by a scaled and normalized histogram of the words it contains.

Transductive Learning

- The collection of feature vectors for all examples in S is denoted as $X = (x_1,x_2,..,x_n)$.
- For the examples in S, labels, $Y = (y_1,y_2,\ldots,y_n)$, are generated independently according to a distribution $P(y1,y2,..,yn)=\prod_{i=1}^{n} P(y_i)$.
- For simplicity, we assume binary labels $y_i \in \{-1,+1\}$.
- As the training set, the learning algorithm can observe the labels of l randomly selected examples $S_{train} \subset S$.
- The remaining $u = n - l$ examples form the test set

  $S_{test} = S \setminus Strain$.

  i.e : $S_{train} = \{l_1, \ldots, l_l\}$ $S_{test} = \{u_1, \ldots, u_u\}$

Transductive Learning

- When training a Transductive learning algorithm $\mathcal{L}$, it not only has access to the training vectors X_{train} and the training labels Y_{train}, i.e $X_{train} = (x_{l1}, x_{l2}, ..., x_{ll})$, $Y_{train} = (y_{l1}, y_{l2}, ..., y_{ll})$, **but also to the unlabeled test vectors** $X_{test} = (x_{u1}, x_{u2}, ..., x_{ul})$.
- The Transductive learner uses X_{train}, Y_{train}, and X_{test} to produce predictions $Y^*_{test} = (y^*_{u1}, y^*_{u2}, ..., y^*_{uu})$, for the labels of the test examples.
- The learner's goal is to minimize the fraction of erroneous predictions,

$$Err_{test}(Y^*_{test}) = \frac{1}{u} \sum_{i \in S_{test}} \delta_{0/1}(\mathbf{y}^*_i, \mathbf{y}_i),$$

on the test set. δ0/1(a, b) is zero if a = b, otherwise it is one

- However, a crucial difference is that the inductive strategy would ignore any information potentially conveyed in X_{test}.

Transductive Learning

- What information do we get from studying the test sample X_{test} and how could we use it?
- The fact that we deal with only a finite set of points means that the hypothesis space H of a Transductive learner is necessarily finite — namely, all vectors $\{-1,+1\}^n$.
- Following the principle of structural risk structural we can structure H into a nested structure

$$\mathcal{H}_1 \subset \mathcal{H}_2 \subset \cdots \subset \mathcal{H} = \{-1,+1\}^n$$

- The structure should reflect prior knowledge about the learning task.
- In particular, the structure should be constructed so that, with high probability, the correct labeling of S is contained in an element H_i of small cardinality.
- This structuring of the hypothesis space H can be motivated using generalization error bounds from statistical learning theory.

Transductive Learning

- In particular, for a learner $\mathcal{L}$ that searches for a hypothesis $(Y^*_{train}, Y^*_{test}) \in H_i$ with small training error,

$$Err_{test}(Y^*_{train}) = \frac{1}{l} \sum_{i \in S_{train}} \delta_{0/1}(\mathbf{y}_i^*, \mathbf{y}_i),$$

- it is possible to upper-bound the fraction of test errors $Err_{test}(Y^*_{test})$
- With probability 1 − η, $Err_{test}(Y^*_{test}) \leq Err_{train}(Y^*_{train}) + \Omega(l, u, |\mathcal{H}_i|, \eta)$
- where the confidence interval $\Omega(l, u, |H_i|, \eta)$ depends on the number of training examples l, the number of test examples u, and the cardinality $|H_i|$ of H_i.
- The smaller the cardinality $|H_i|$, the smaller is the confidence interval Ω(l, u, |Hi|, η) on the deviation between training and test error.
- The bound indicates that a good structure ensures accurate prediction of the test labels.

Transductive Learning

- In particular, in the Transductive setting it is possible to encode prior knowledge we might have about the relationship between the geometry of $X = (x_1, ..., x_n)$ and $P(y_1, ..., y_n)$.
- If such a relationship exists, we can build a more appropriate structure and reduce the number of training examples necessary for achieving a desired level of prediction accuracy.

Support Vector Machine

- Please refer SVM: https://www.youtube.com/watch?v=_PwhiWxHK8o

- Please refer SVM algorithm working https://www.youtube.com/watch?v=1NxnPkZM9bc

- Please refer SVM with simple animation https://www.youtube.com/watch?v=5zRmhOUjjGY

Transductive SVM (TSVM)

- It is assume a particular geometric relationship between $X = (x_1,\dots, x_n)$ and $P(y_1,\dots, y_n)$.
- Here it builds a structure on H based on the margin of hyperplanes $\{x : w \cdot x + b = 0\}$ on the complete sample $X = (x_1, x_2,\dots, x_n)$, including both the training and the test vectors.
- The margin of a hyperplane on X is the minimum distance to the closest example vectors in X.

$$\min_{i \in [1..n]} \left[\frac{\mathbf{y}_i}{\|\mathbf{w}\|} (\mathbf{w} \cdot \mathbf{x}_i + b) \right]$$

- The structure element H_ρ contains all labelings of X which can be achieved with hyperplane classifiers $h(x) = \text{sign}\{x \cdot w + b\}$ that have a margin of at least ρ on X.
- Intuitively, building the structure based on the margin gives preference to labelings that follow cluster boundaries over labelings that cut through clusters.

Transductive SVM (TSVM)

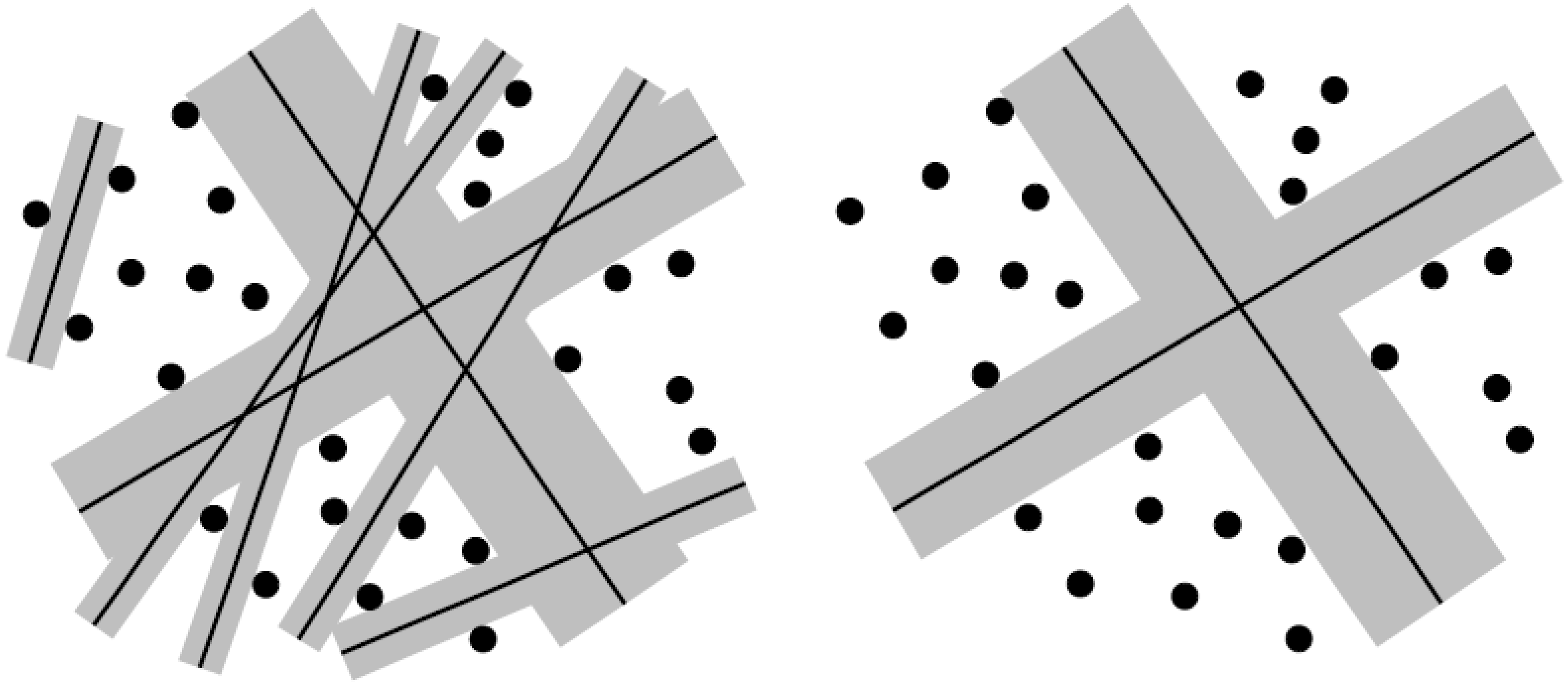


Figure 6.1 The two graphs illustrate the labelings that margin hyperplanes can realize dependent on the margin size. Example points are indicated as dots: the margin of each hyperplane is illustrated by the gray area. The left graph shows the separators $\mathcal{H}_\rho$ for a small margin threshold ρ. The number of possible labelings N_ρ decreases as the margin threshold is increased, as in the graph on the right.

Transductive SVM (TSVM)

- It is assume a particular geometric relationship between X = (x_1,..., x_n) and P(y_1,..., y_n).
- Vapnik shows that the size of the margin ρ can be used to control the cardinality of the corresponding set of labelings H_ρ.

Theorem 6.1 ((Vapnik, 1998))
For any n vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ that are contained in a ball of diameter R, the number $|\mathcal{H}_\rho|$ of possible binary labelings $\mathbf{y}_1, \ldots, \mathbf{y}_n \in \{-1, +1\}$ that can be realized with hyperplane classifiers $h(\mathbf{x}) = sign\{\mathbf{x} \cdot \mathbf{w} + b\}$ of margin at least ρ,

$$\forall_{i=1}^{n} : \frac{\mathbf{y}_i}{\|\mathbf{w}\|} [\mathbf{w} \cdot \mathbf{x}_i + b] \geq \rho \tag{6.13}$$

is bounded by

$$|\mathcal{H}_\rho| \leq e^{d\left(\ln \frac{n+k}{d} + 1\right)}, \qquad d = \frac{R^2}{\rho^2} + 1. \tag{6.14}$$

Transductive SVM (TSVM)

- More formally, the above theorem provides an upper bound on the number of labelings $|H_\rho|$ that can be achieved with hyperplanes that have a margin of at least ρ.
- Note that the number of labelings $|H_\rho|$ does not necessarily depend on the number of features d.
- As suggested by the theorem, TSVMs sort all labelings by their margin ρ on X to build the structure on H.
- Structural risk minimization argues that a learning algorithm should select the labeling $Y * \in H_\rho$ for which training error $Err_{train}(Y^*_{\ train})$ and cardinality of H_ρ minimize the generalization error bound (specified in theorem)
- For the special case of requiring zero training error, optimizing the bound means finding the labeling with the largest margin on the complete set of vectors. → This leads to the optimization problems → Hard-Margin and Soft-Margin

Summary

- Seeded and Constrained K-Means: partially labeled data
- COP K-Means: constraints (Must-link and Cannot-link)

- Constrained K-Means and COP K-Means require all the constraints to be satisfied.
  - May not be effective if the seeds contain noise.
- Seeded K-Means use the seeds only in the first step to determine the initial centroids.
  - Less sensitive to the noise in the seeds.
- Transductive SVM