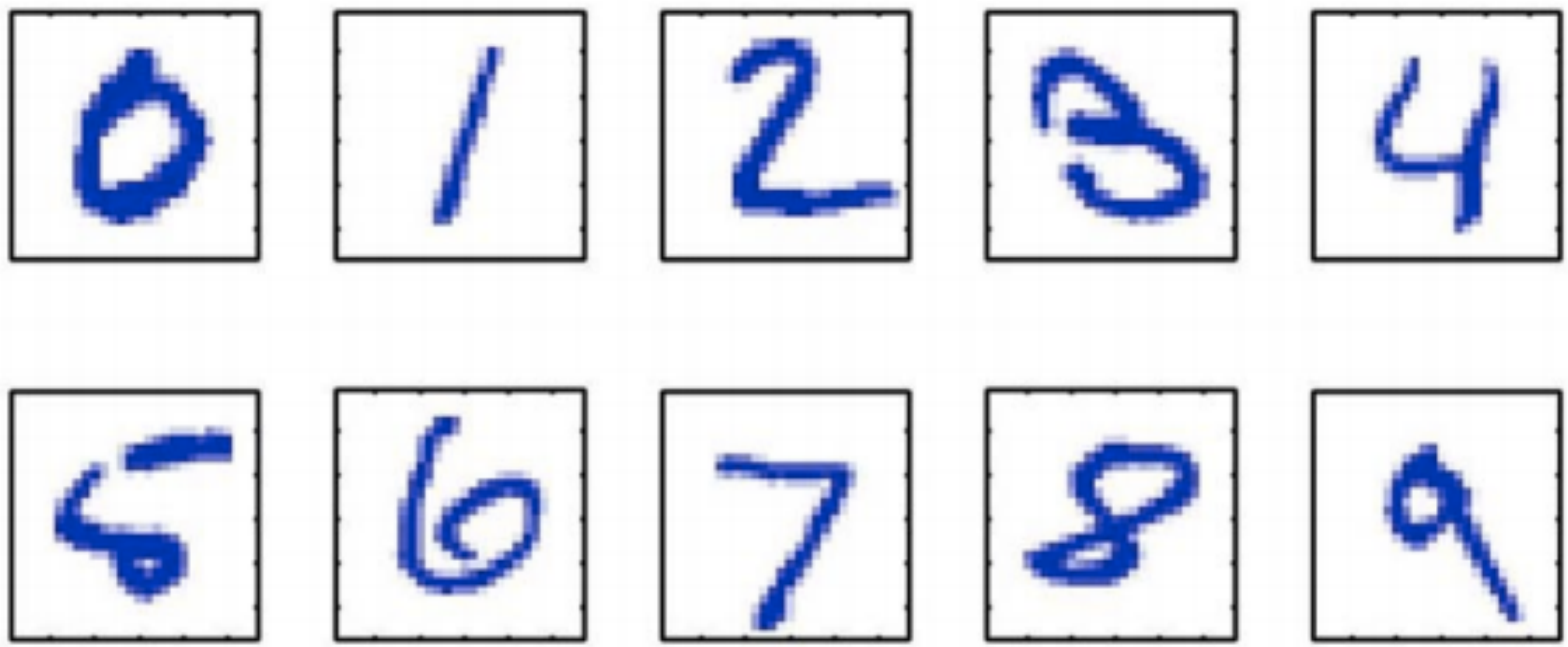
Classification

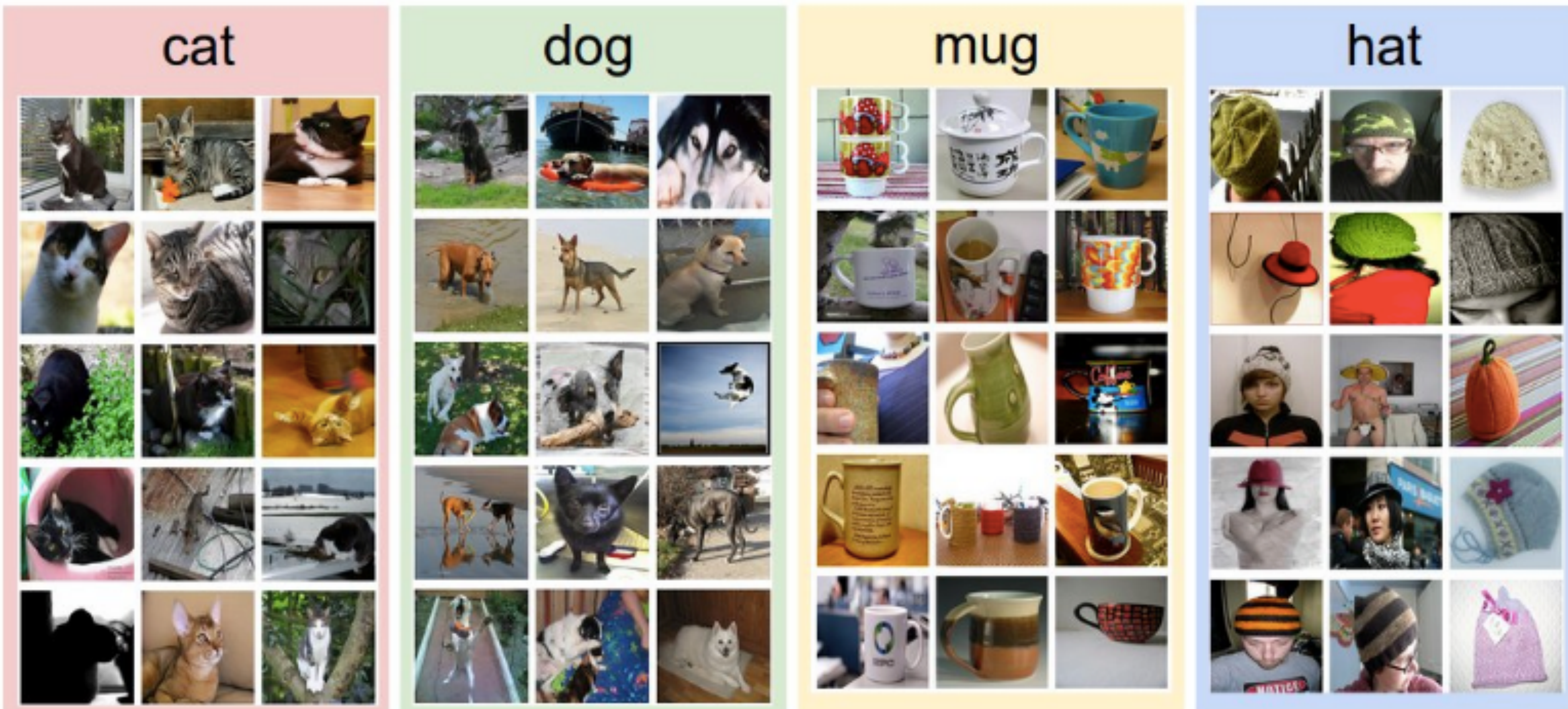


Images are 28 x 28 pixels

Represent input image as a vector $\mathbf{x} \in \mathbb{R}^{784}$

Learn a classifier $f(\mathbf{x})$ such that,

$$f : \mathbf{x} \rightarrow \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

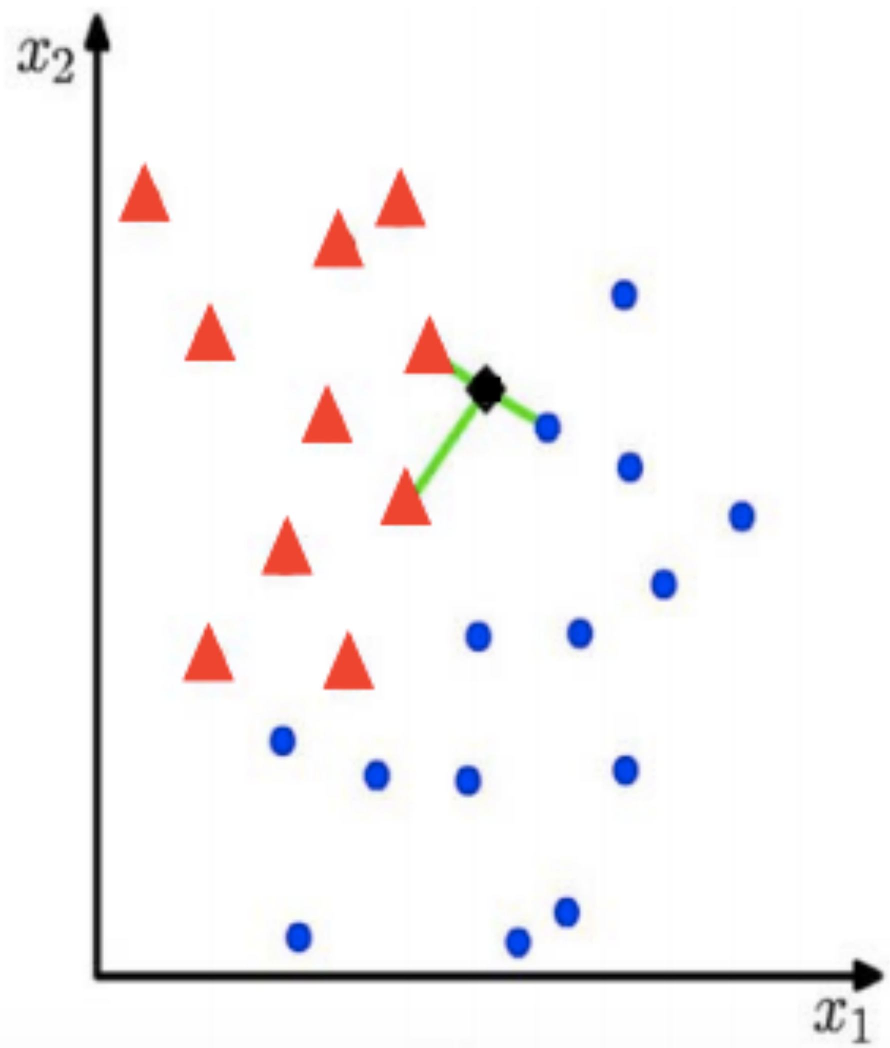

cat
dog
mug
hat

K-nearest neighbor classification

Algorithm

- For each test point, x, to be classified, find the K nearest samples in the training data
- Classify the point, x, according to the majority vote of their class labels

e.g. K = 3



- applicable to multi-class case

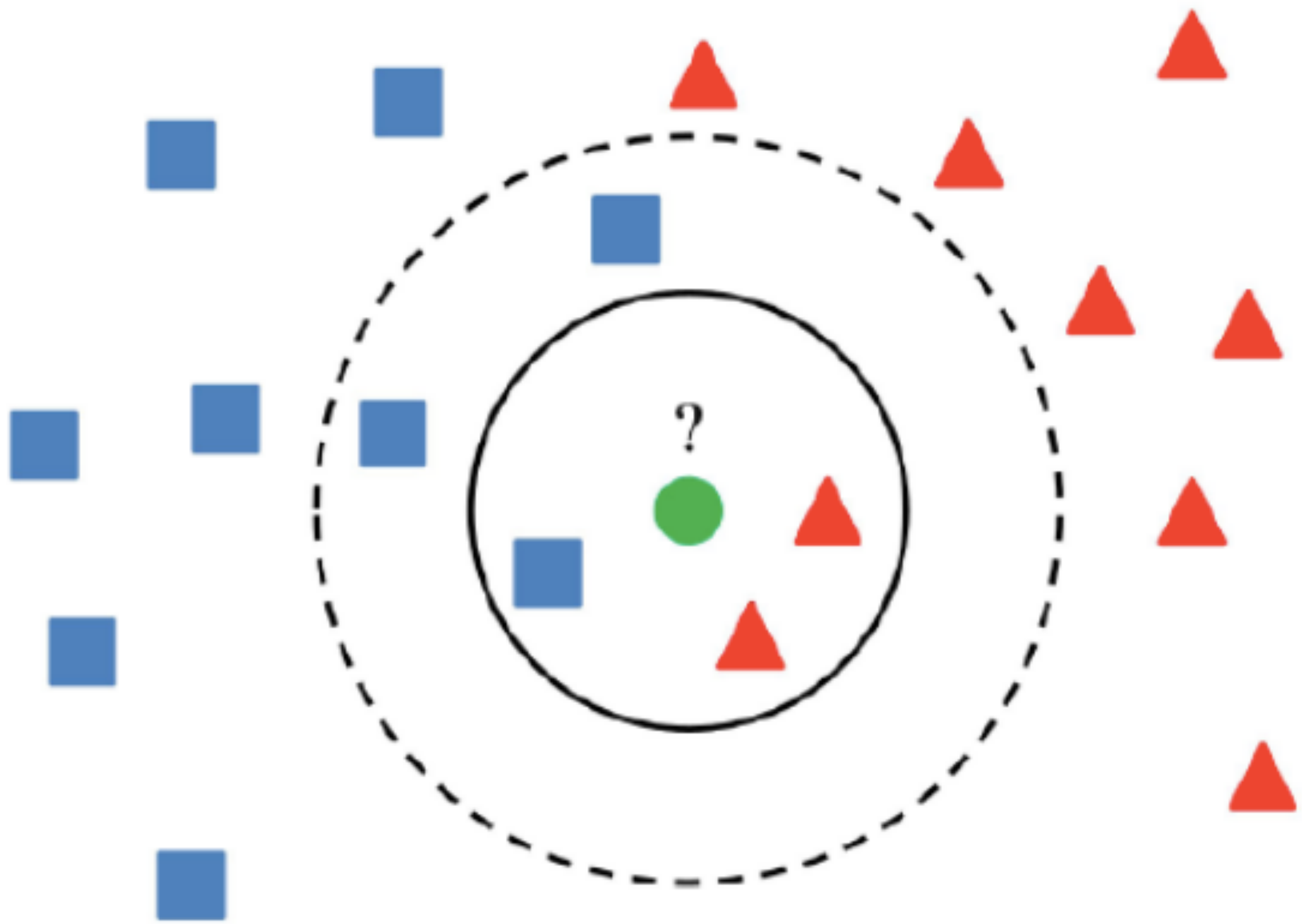
Distance functions

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

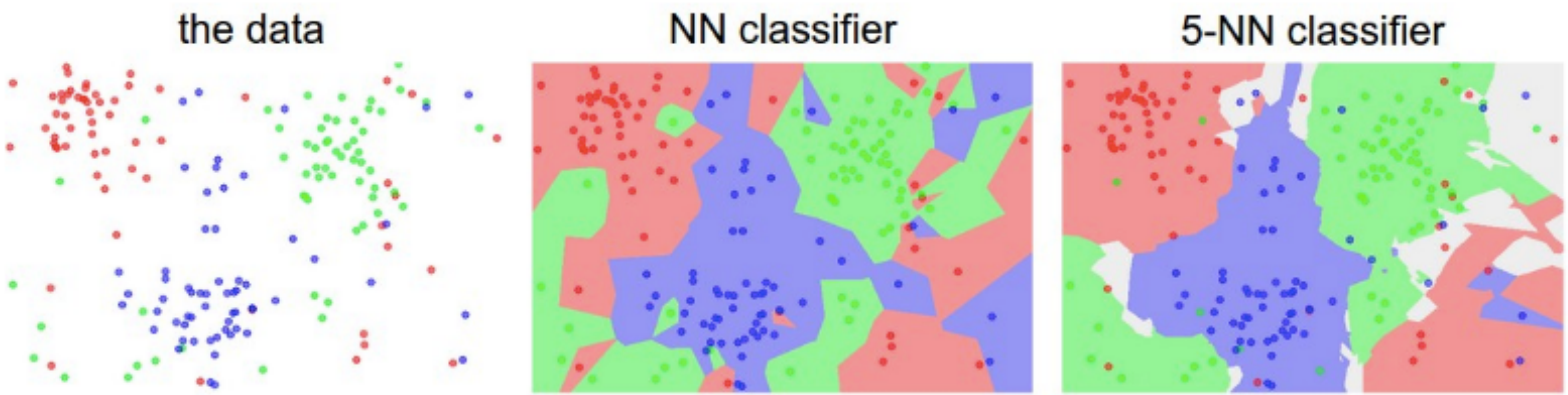
Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

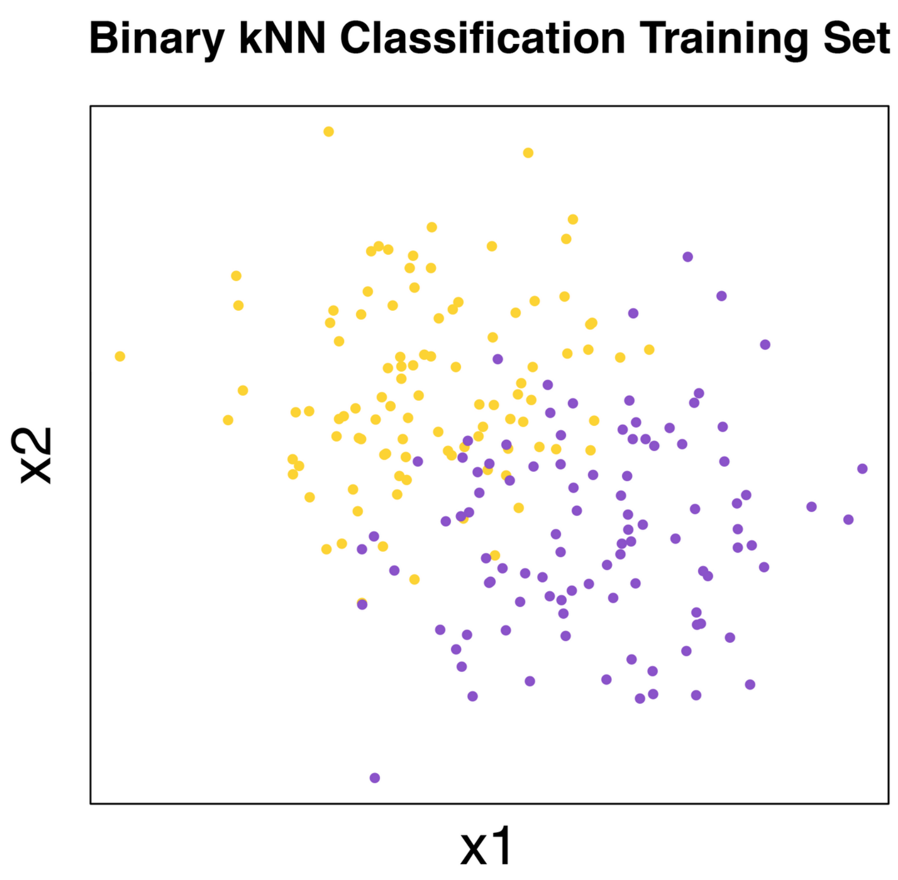
Minkowski $\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

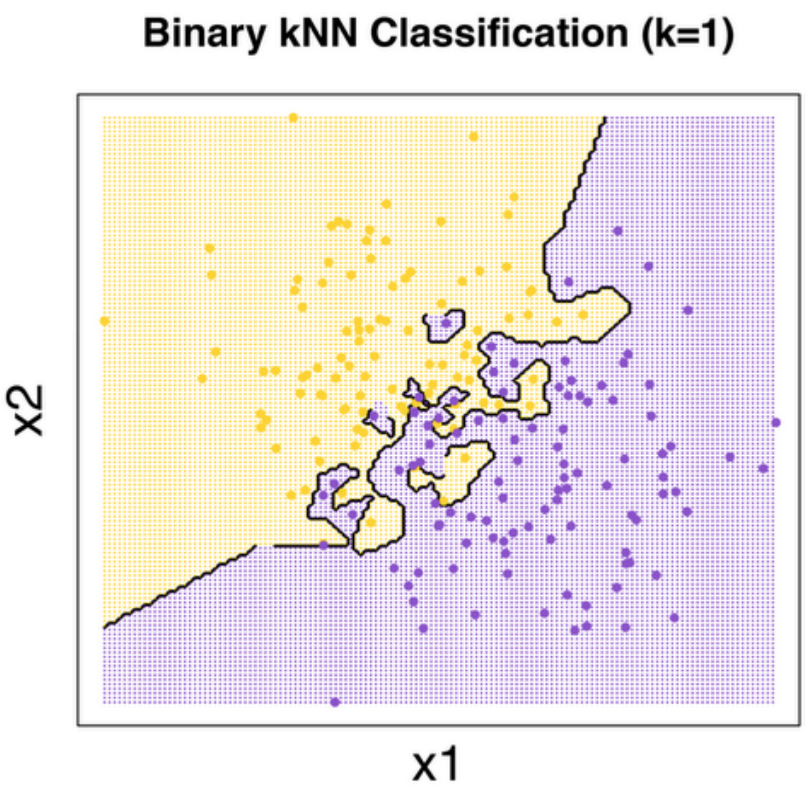
Choice of k

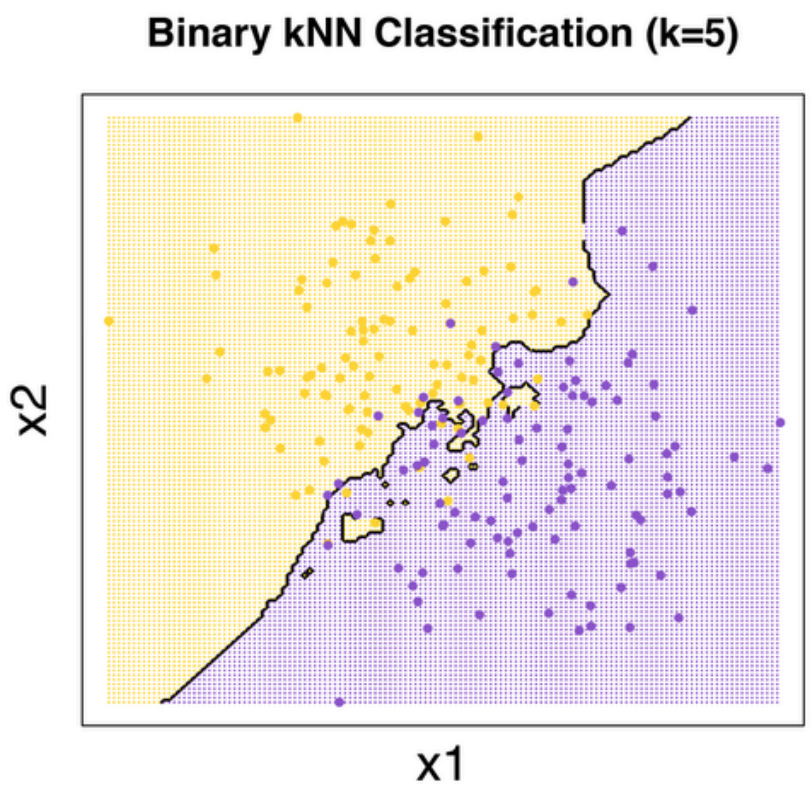


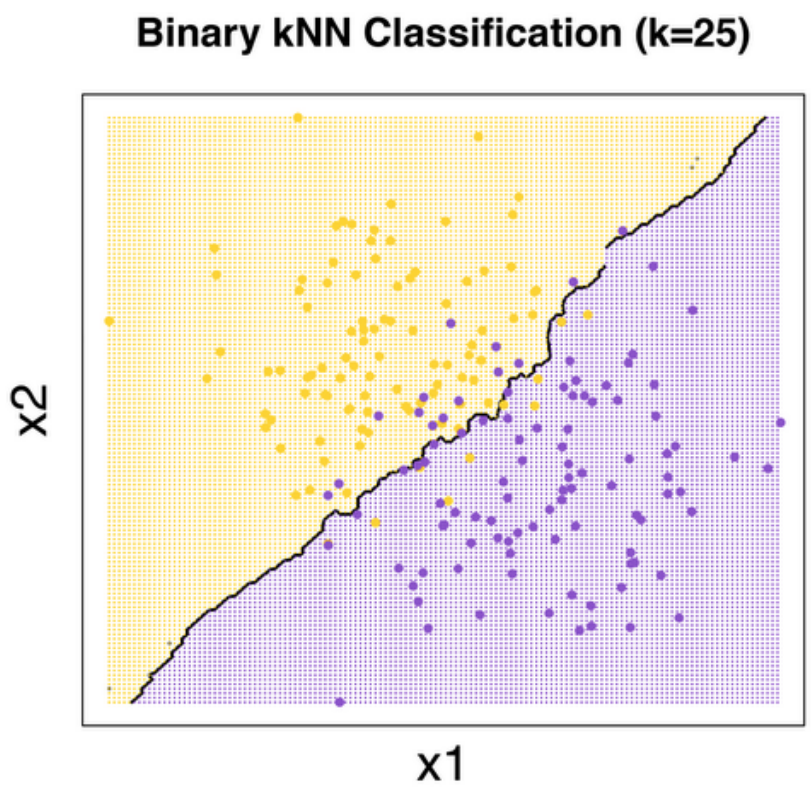
Choice of k




Binary kNN Classification Training Set
x2
x1

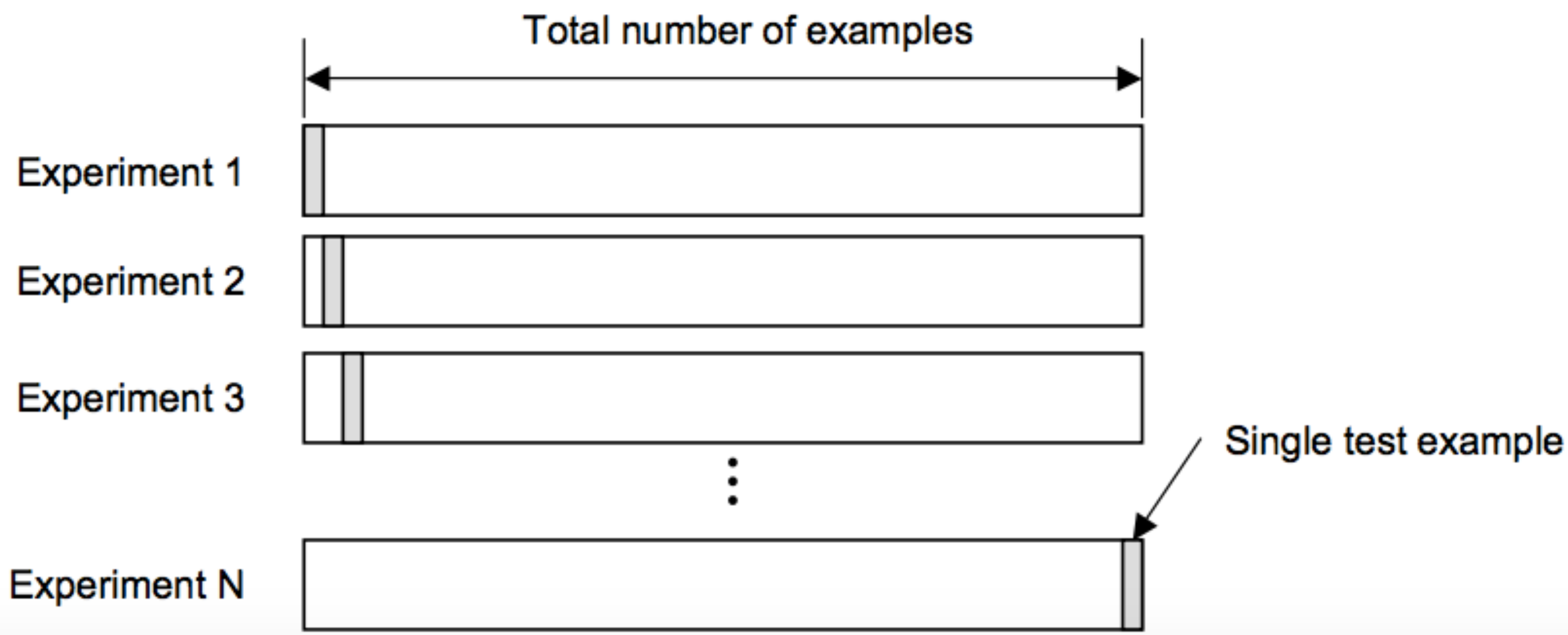

Binary kNN Classification (k=1)
x2
x1


Binary kNN Classification (k=5)
x2
x1


Binary kNN Classification (k=25)
x2
x1

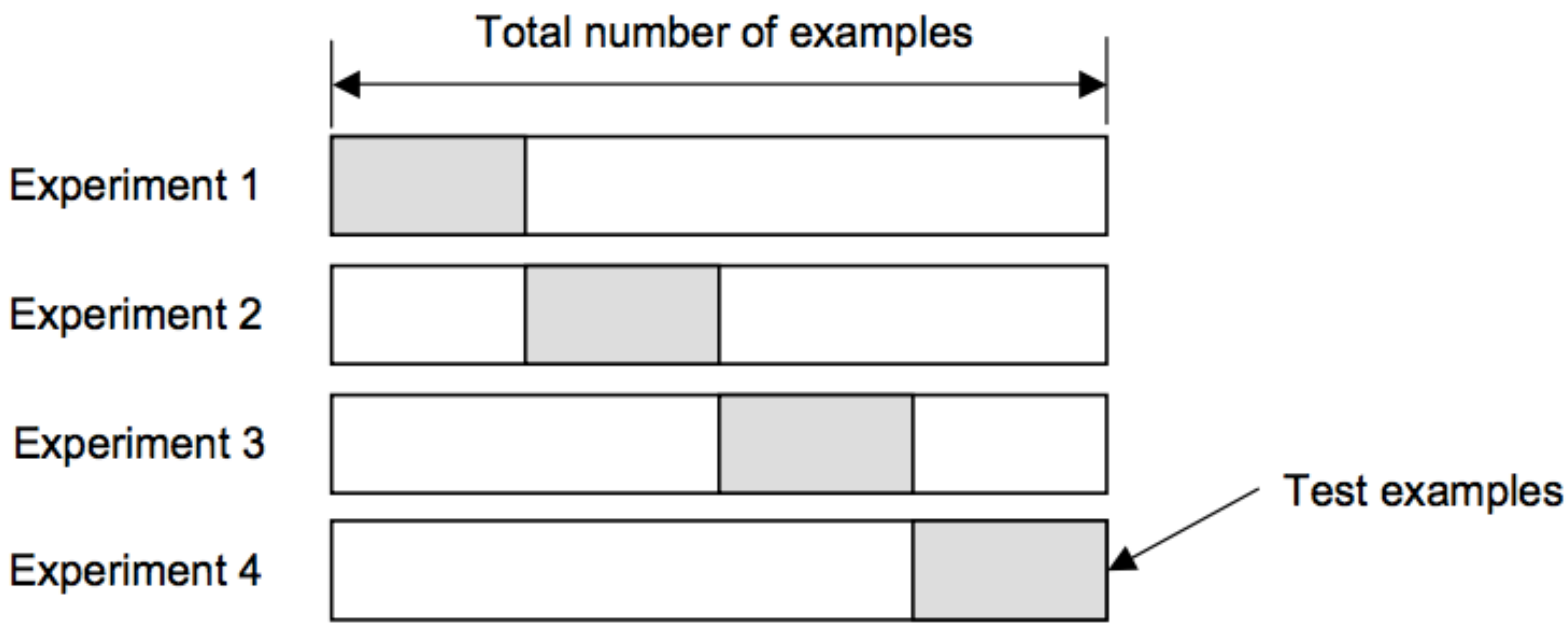
Leave-one-out cross validation

- For a dataset with N examples, perform N experiments
- For each experiment use N-1 examples for training and the remaining example for testing



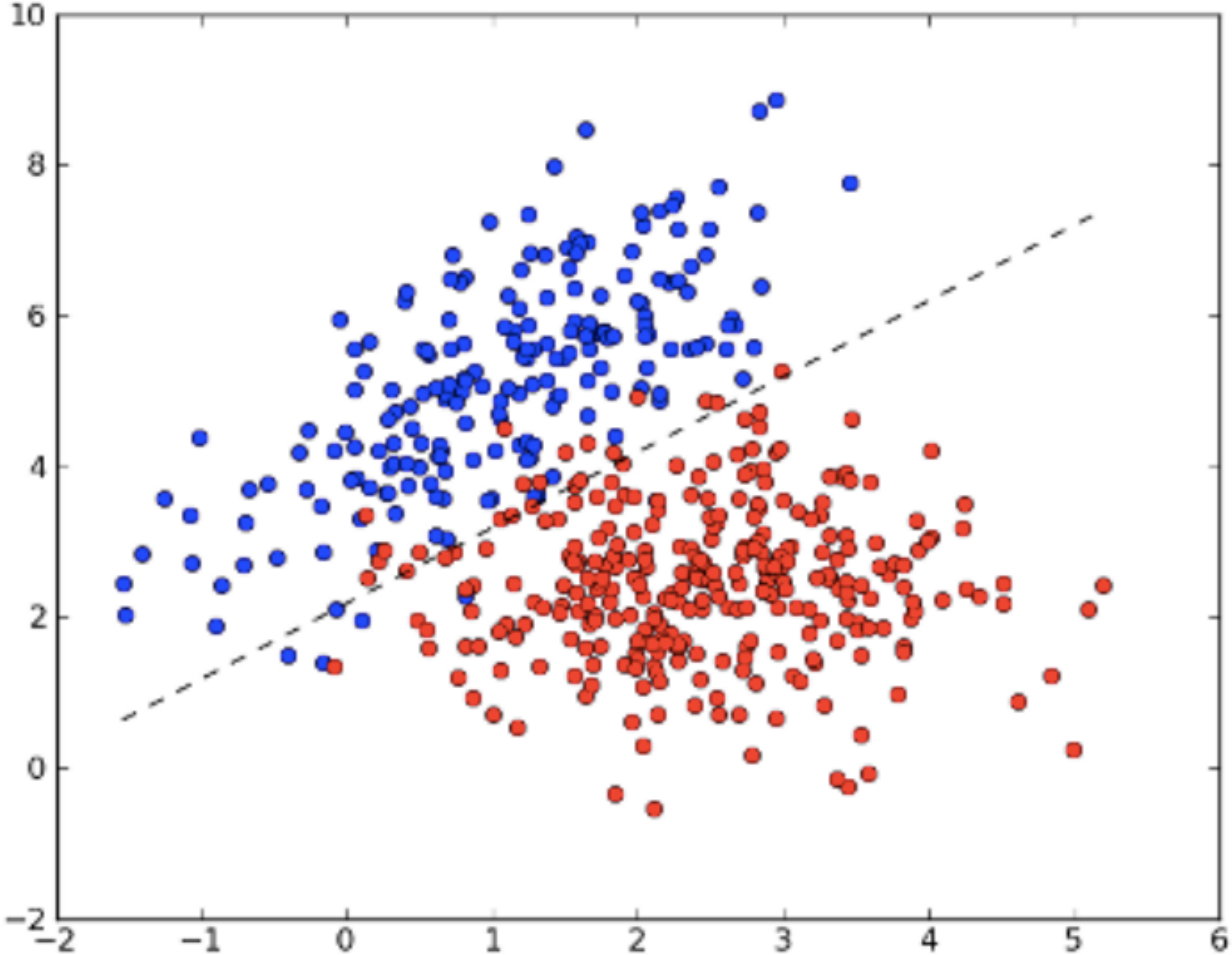
K-fold cross validation

- For each of K experiments, use K-1 folds for training and the remaining one for testing



Classification Error = Average classification error on K folds

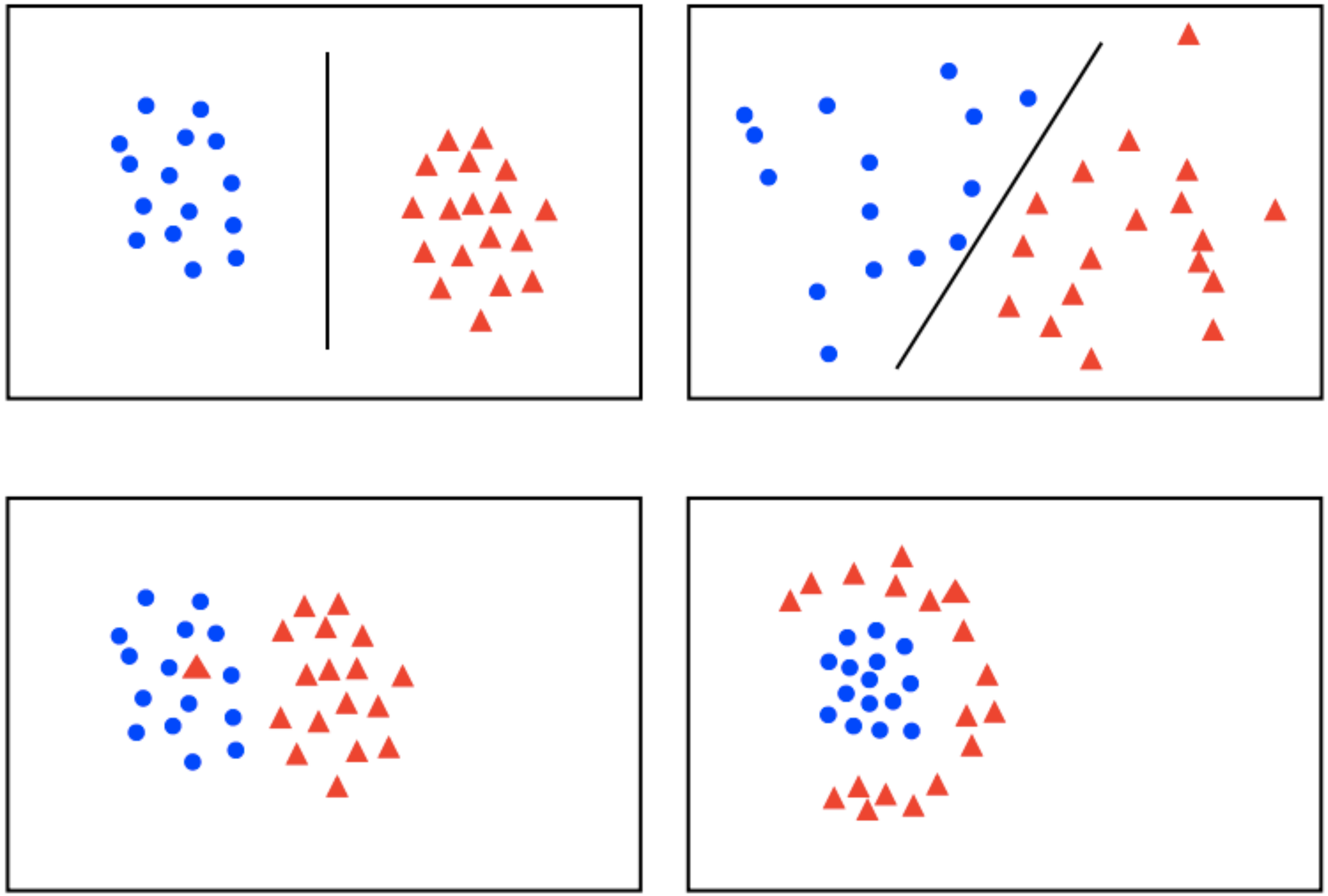
Linear Classification



Linear separability

linearly
separable

not
linearly
separable

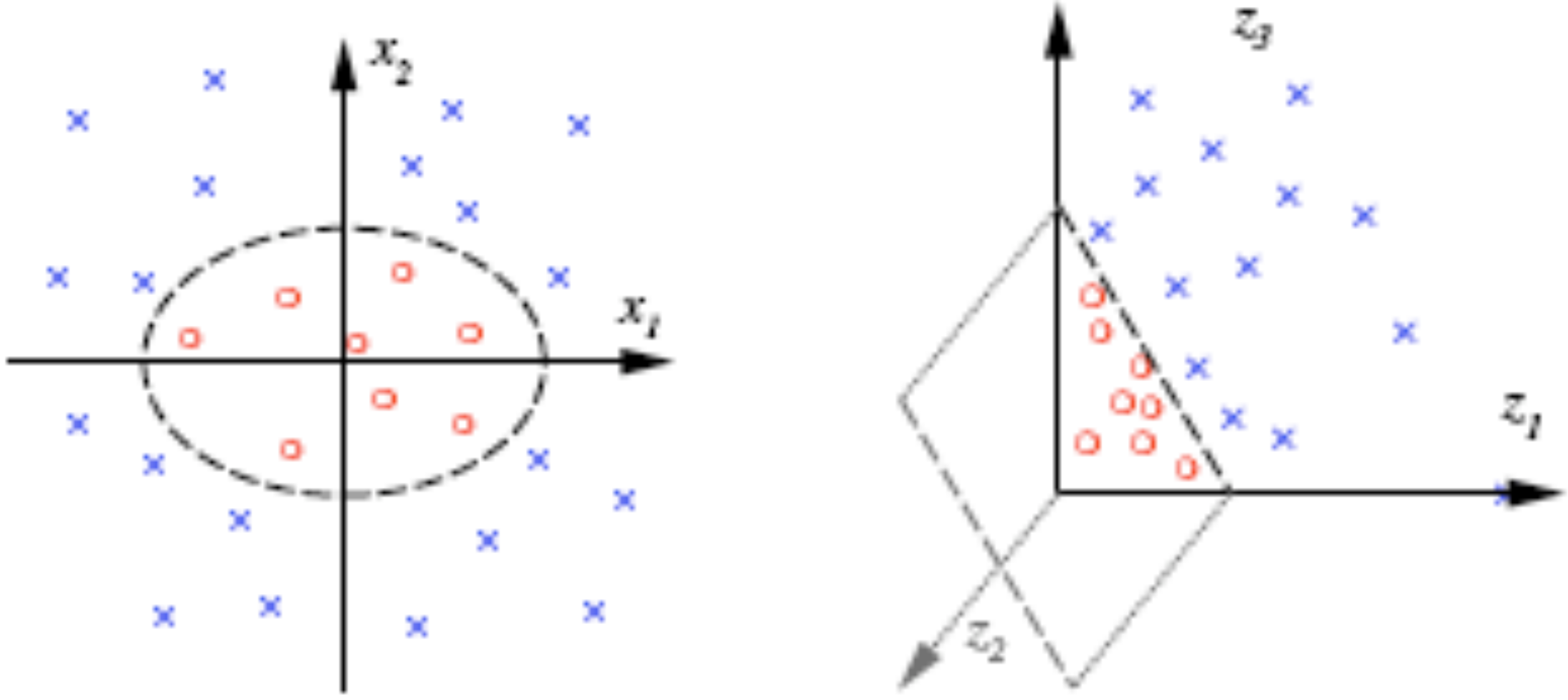


Inseparability

- Real world problems: there may not exist a hyperplane that separates cleanly
- Solution to this “inseparability” problem: map data to higher dimensional space
  - Called the “feature space”, as opposed to the original “input space”
  - Inseparable training set can be made separable with proper choice of feature space

Feature map

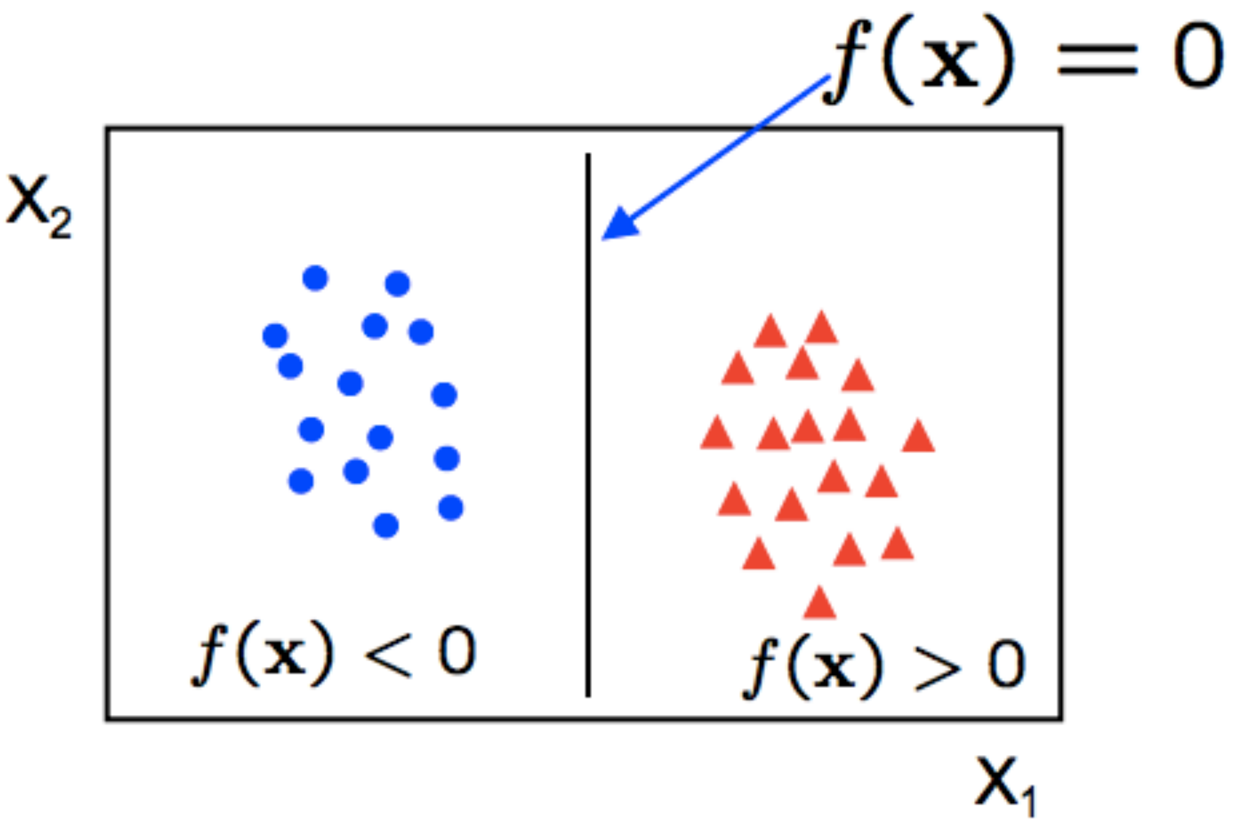
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)$$



Linear classifier

A linear classifier has the form

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$

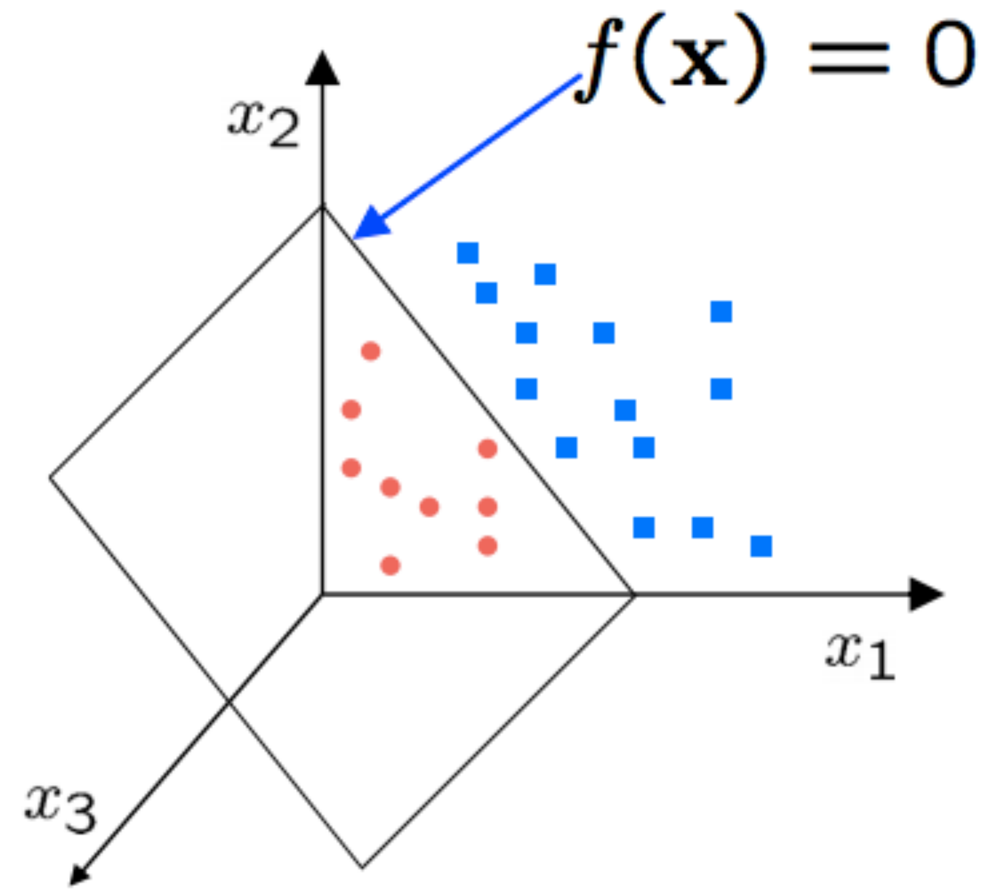


- in 2D the discriminant is a line
- $\mathbf{w}$ is the normal to the line, and b the bias
- $\mathbf{w}$ is known as the weight vector

Linear classifier

A linear classifier has the form

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$



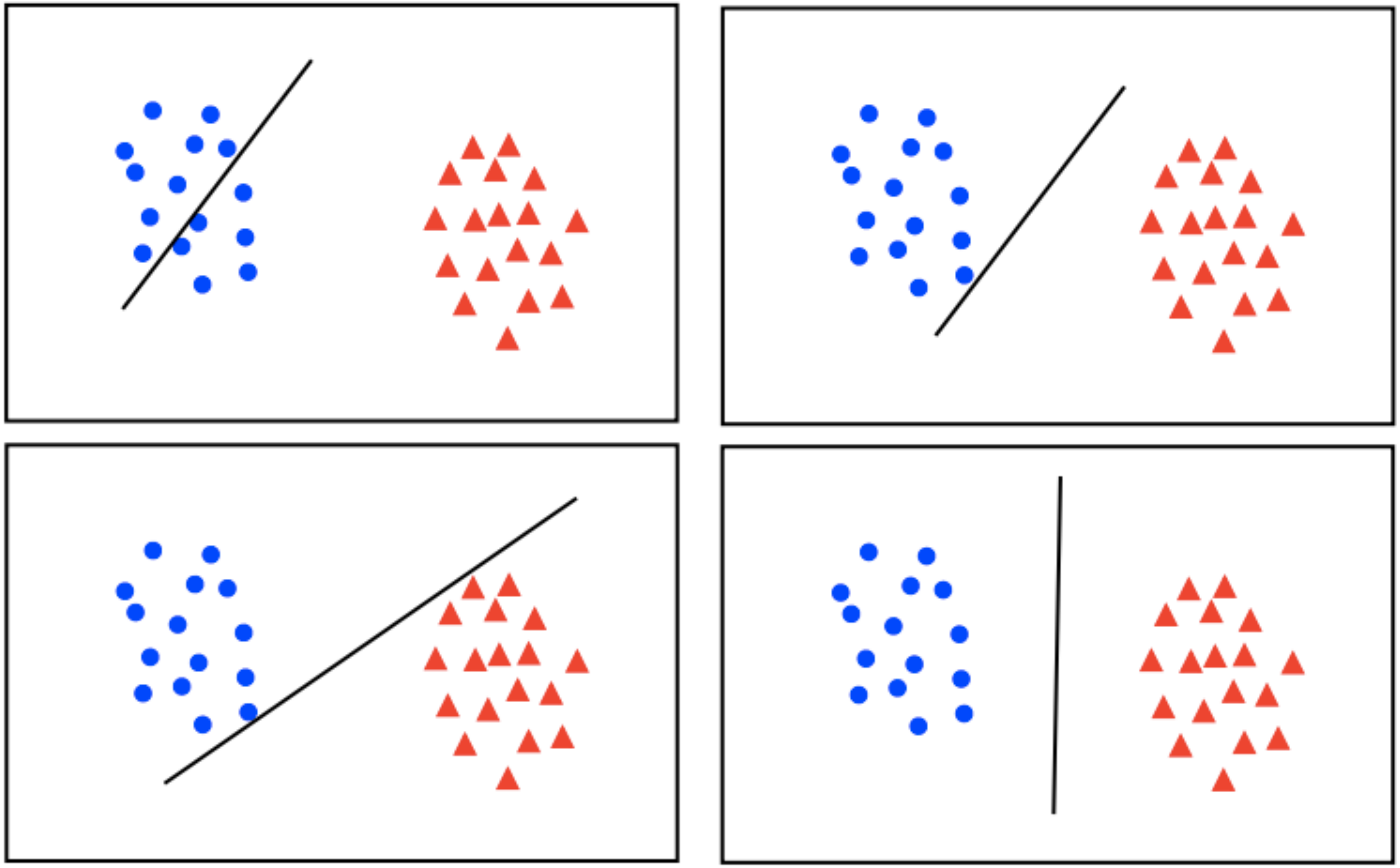
- in 3D the discriminant is a plane, and in nD it is a hyperplane

For a K-NN classifier it was necessary to `carry' the training data

For a linear classifier, the training data is used to learn **w** and then discarded

Only **w** is needed for classifying new data

Good and bad linear classifiers

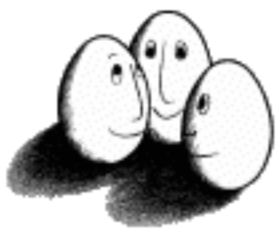


• maximum margin solution: most stable under perturbations of the inputs

Support Vector Machine

Two popular implementations

svmlight.joachims.org



CORNELL

SVM*light*

Support Vector Machine

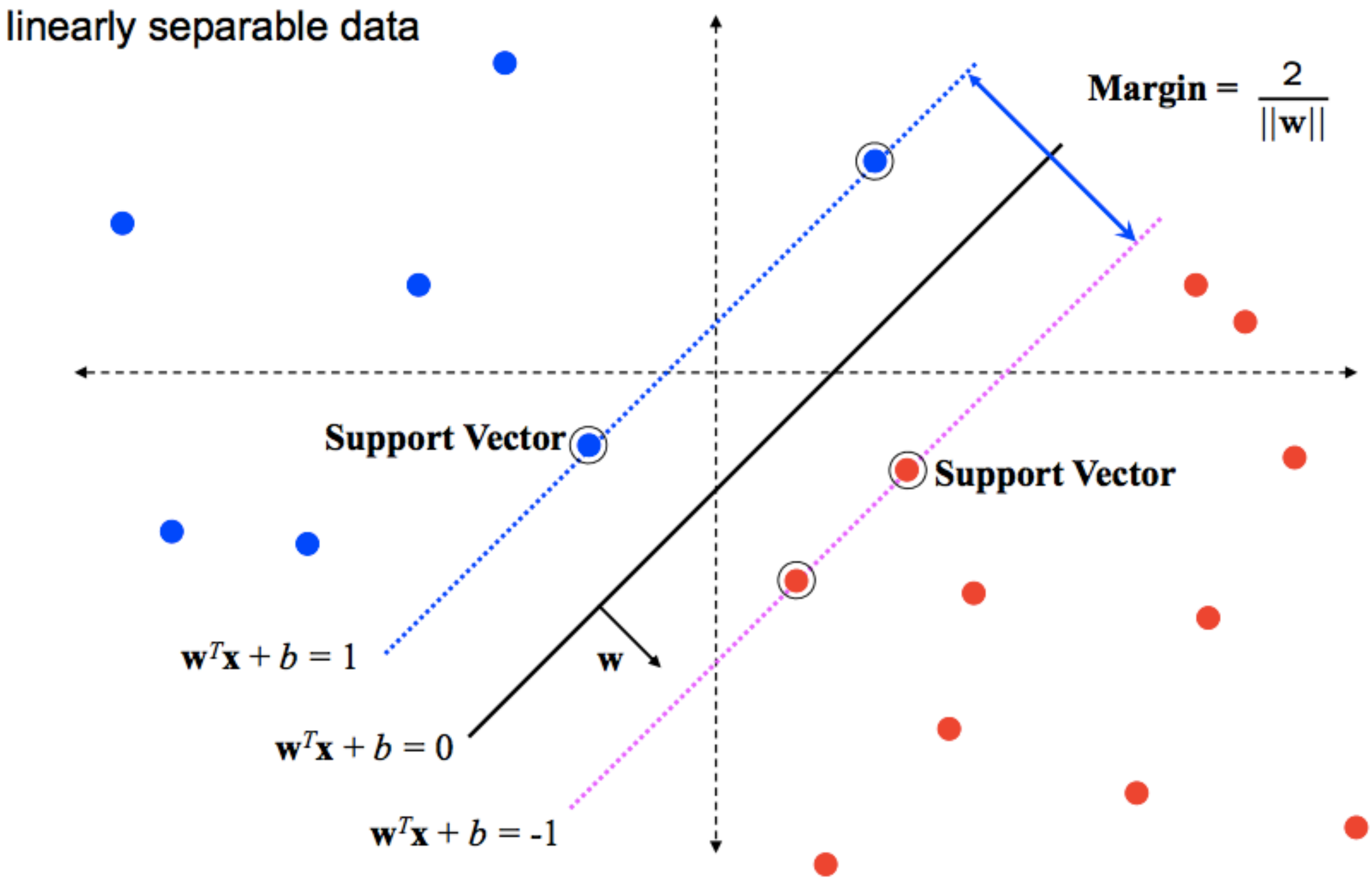
Author: Thorsten Joachims <thorsten@joachims.org>
Cornell University
Department of Computer Science

Secure | https://www.csie.ntu.edu.tw/~cjlin/libsvm/

LIBSVM -- A Library for Support Vector Machines

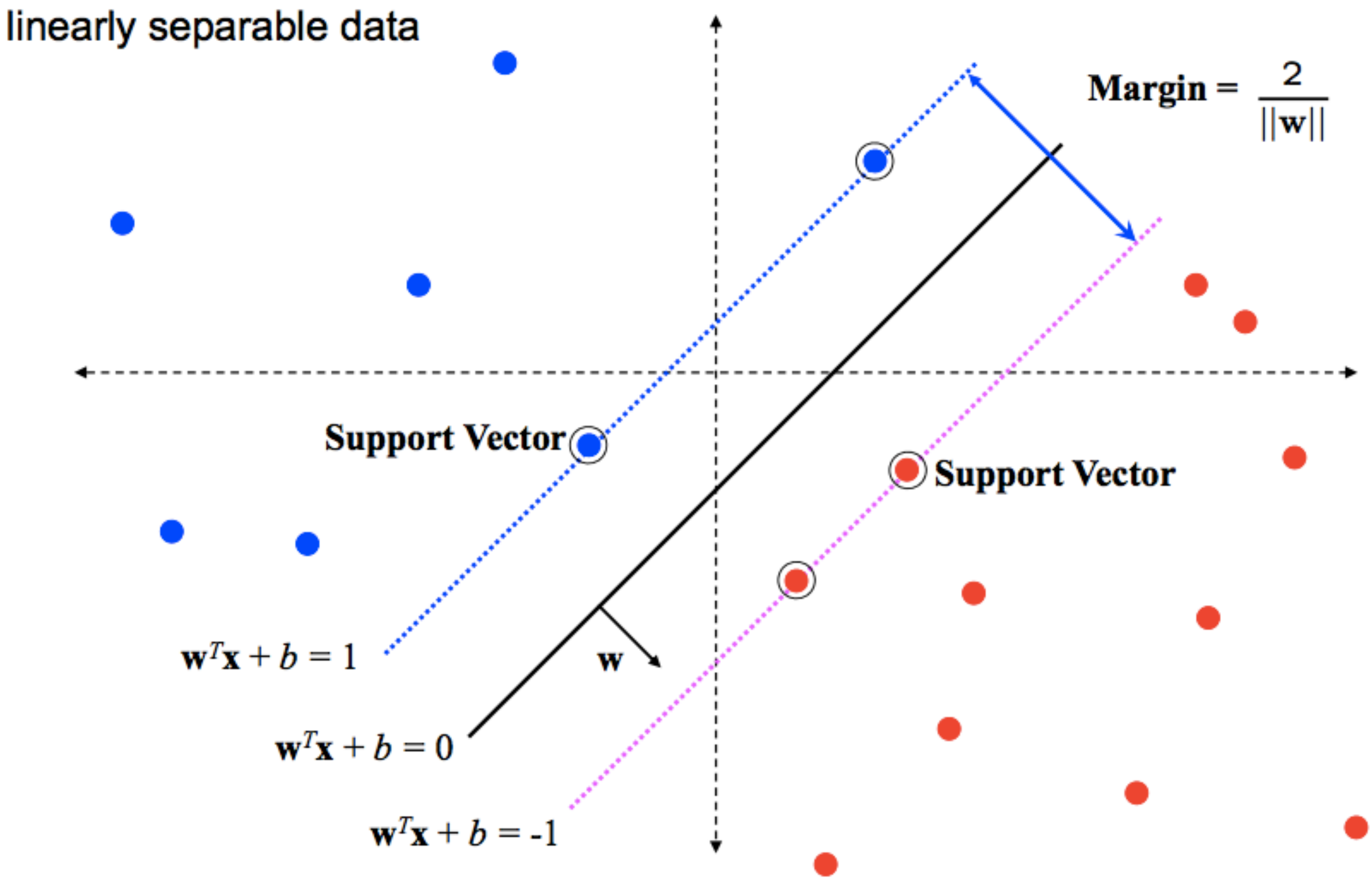
Chih-Chung Chang and Chih-Jen Lin

Margin



linearly separable data
Margin = $\frac{2}{||\mathbf{w}||}$
Support Vector
Support Vector
$\mathbf{w}^T\mathbf{x} + b = 1$
$\mathbf{w}$
$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b = -1$

Margin



linearly separable data
Margin = $\frac{2}{||\mathbf{w}||}$
Support Vector
Support Vector
$\mathbf{w}^T\mathbf{x} + b = 1$
$\mathbf{w}$
$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b = -1$

Linear Support Vector Machine

- Learning the SVM can be formulated as an optimization:

$$\max_{\mathbf{w}} \frac{2}{||\mathbf{w}||} \text{ subject to } \mathbf{w}^\top \mathbf{x}_i + b \begin{array}{ll} \geq 1 & \text{if } y_i = +1 \\ \leq -1 & \text{if } y_i = -1 \end{array} \text{ for } i = 1 \dots N$$

- Or equivalently

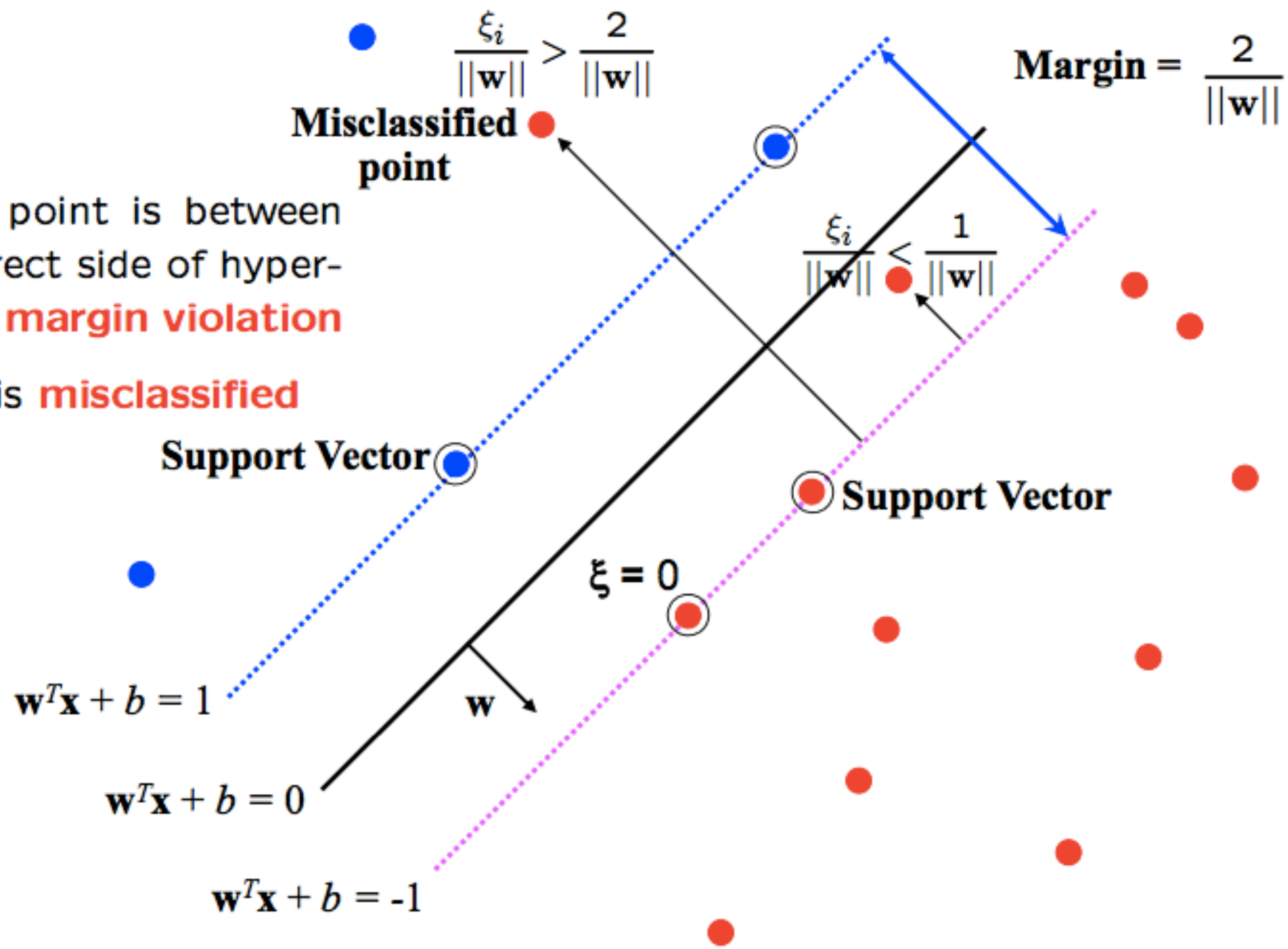
$$\min_{\mathbf{w}} ||\mathbf{w}||^2 \text{ subject to } y_i \left(\mathbf{w}^\top \mathbf{x}_i + b \right) \geq 1 \text{ for } i = 1 \dots N$$

- This is a quadratic optimization problem subject to linear constraints and there is a unique minimum

Inseparable case

$$\xi_i \geq 0$$

- for $0 < \xi \leq 1$ point is between margin and correct side of hyper-plane. This is a **margin violation**
- for $\xi > 1$ point is **misclassified**



Linear SVM

The optimization problem becomes

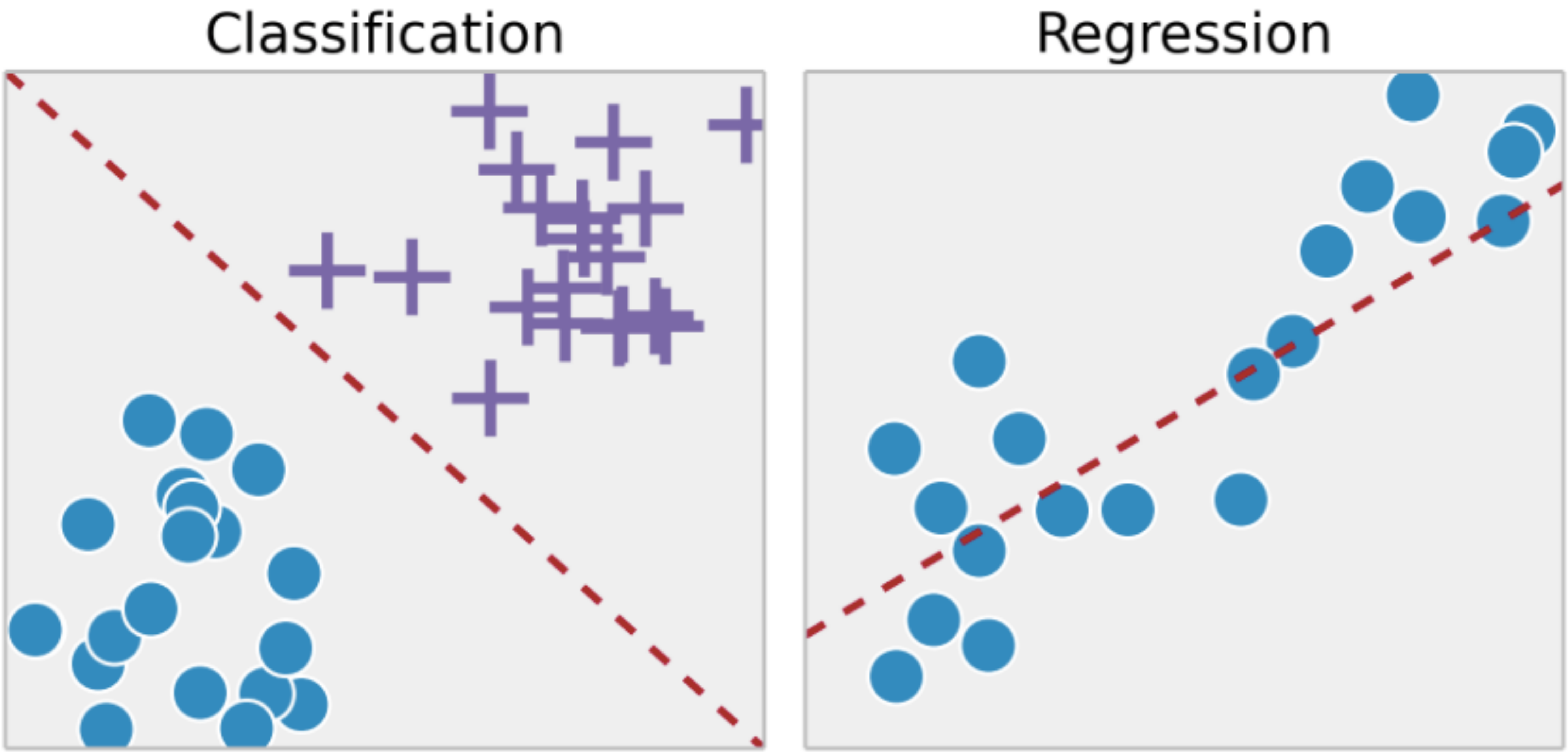
$$\min_{\mathbf{w}\in\mathbb{R}^d,\xi_i\in\mathbb{R}^+} ||\mathbf{w}||^2 + C\sum_i^N \xi_i$$

subject to

$$y_i\left(\mathbf{w}^\top \mathbf{x}_i + b\right) \geq 1 - \xi_i \text{ for } i = 1 \dots N$$

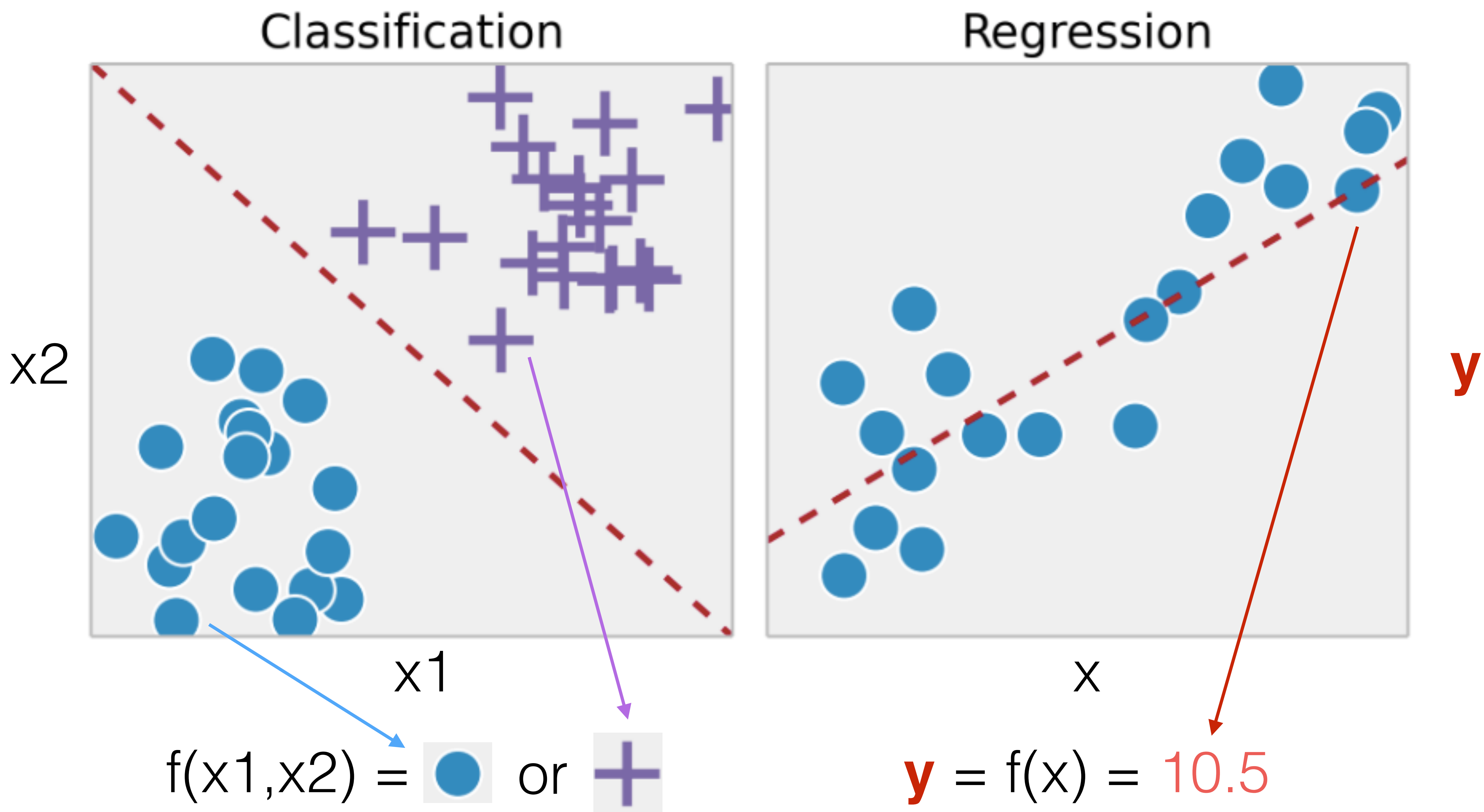
- Every constraint can be satisfied if ξ_i is sufficiently large
- C is a regularization parameter:
  - small C allows constraints to be easily ignored $\rightarrow$ large margin
  - large C makes constraints hard to ignore $\rightarrow$ narrow margin
  - $C = \infty$ enforces all constraints: hard margin
- This is still a quadratic optimization problem and there is a unique minimum. Note, there is only one parameter, C.

Classification vs Regression



Discrete

Continuous



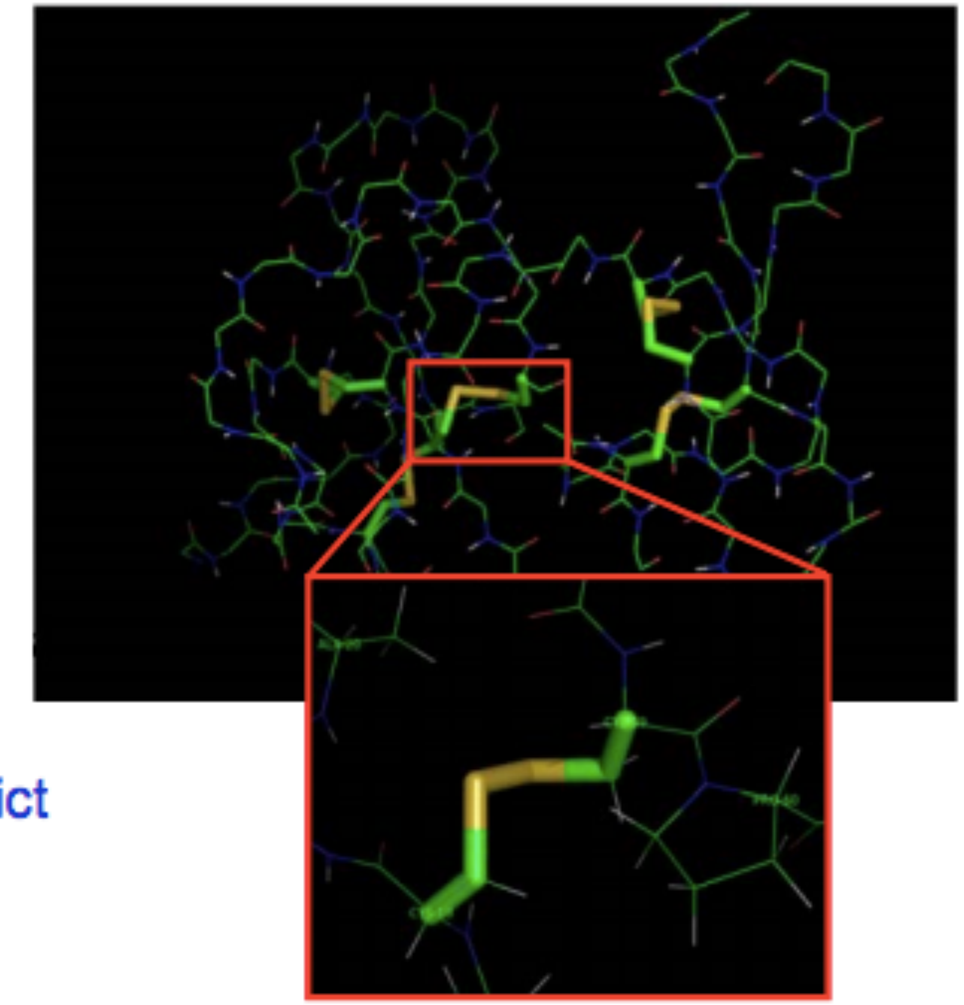
Classification
Regression
x2
x1
x
y
f(x1,x2) = or
y = f(x) = 10.5

Protein structure prediction as regression

x

AVITGACERDLQCG
KGTCCAVSLWIKSV
RVCTPVGTSGEDCH
PASHKIPFSGQRMH
HTCPCAPNLACVQT
SPKKFKCLSK

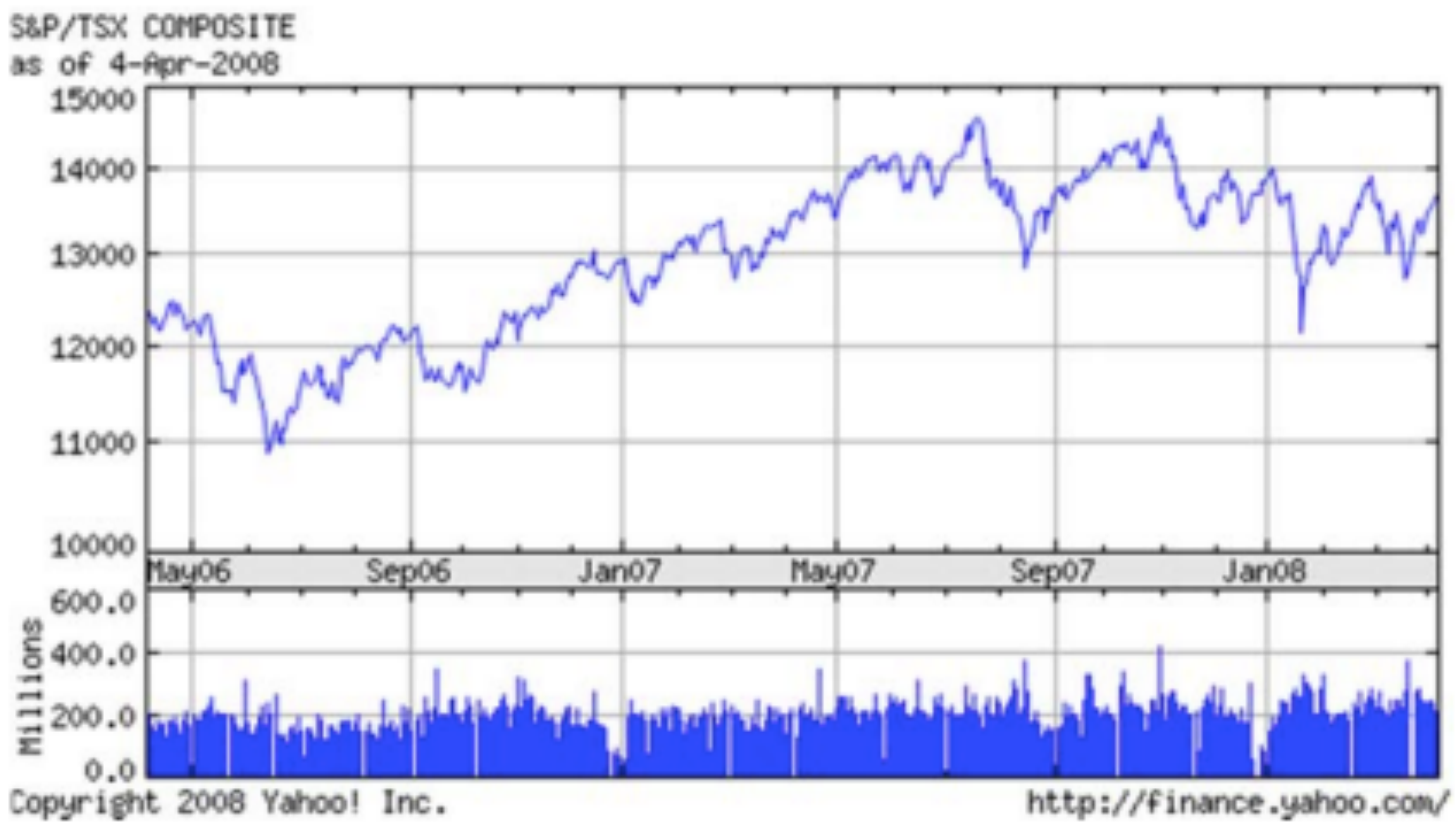
y



Regression task: given sequence predict 3D structure

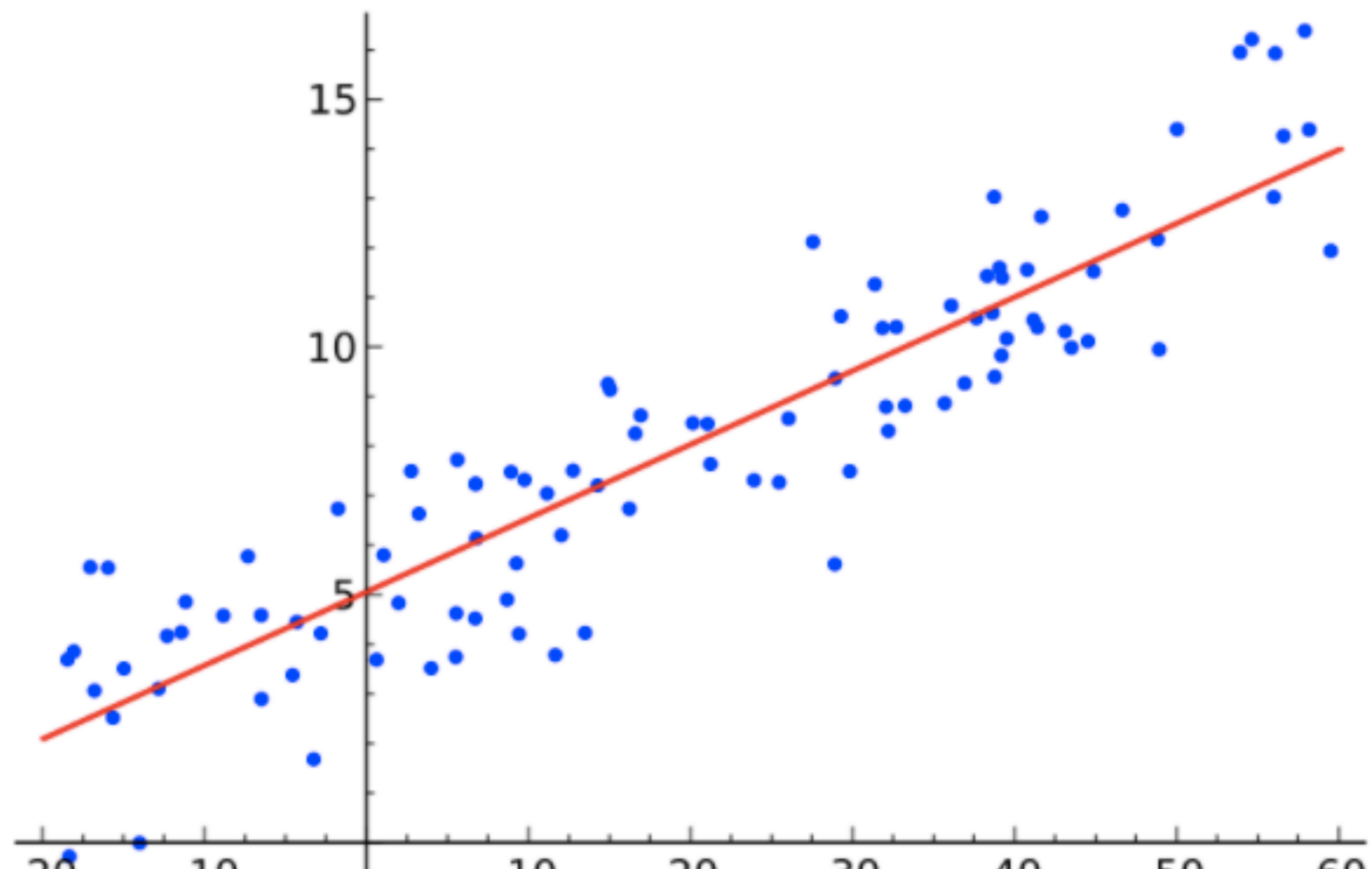
3D coordinates and angles

Stock price prediction

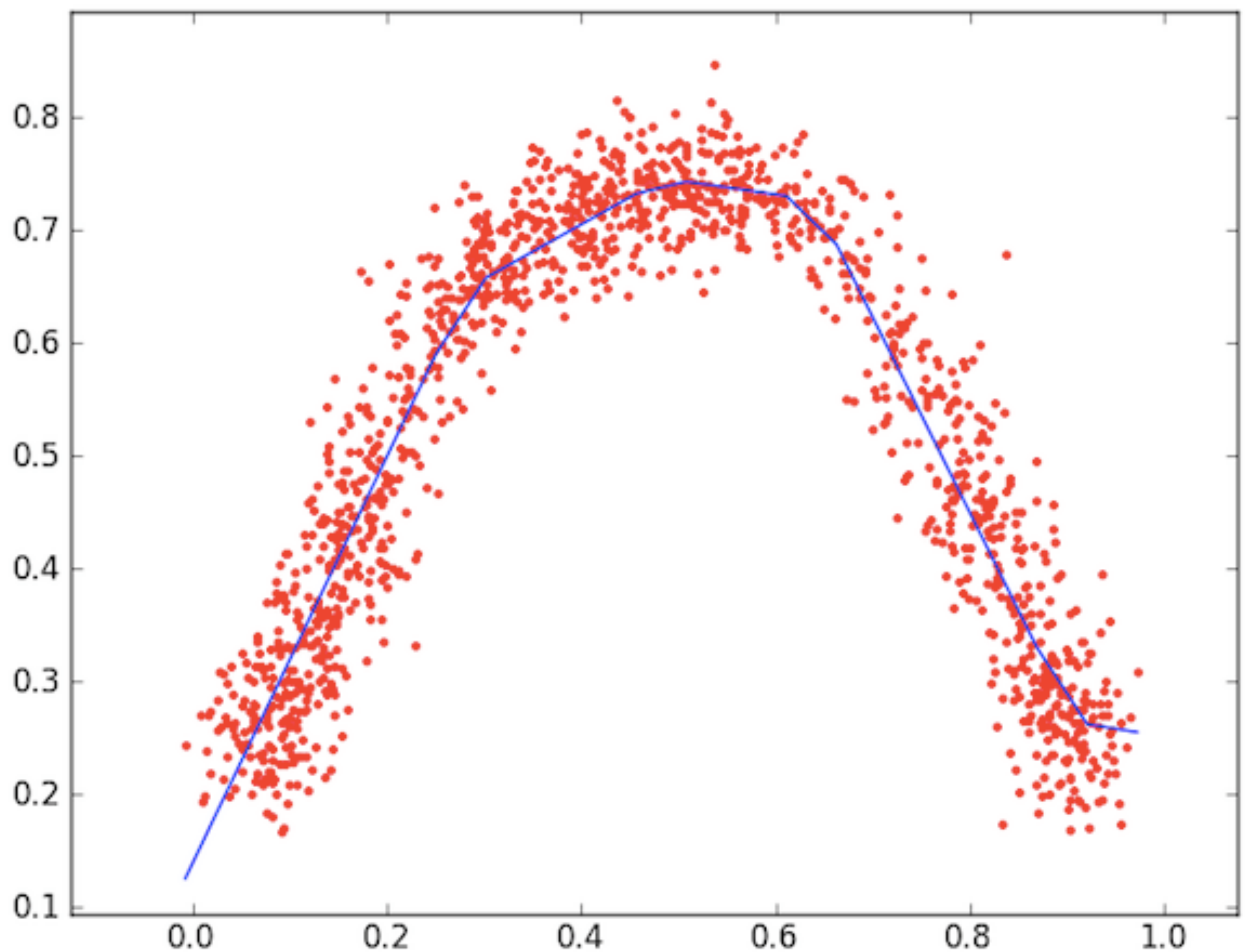


- Task is to predict stock price at future date
- This is a regression task, as the output is continuous

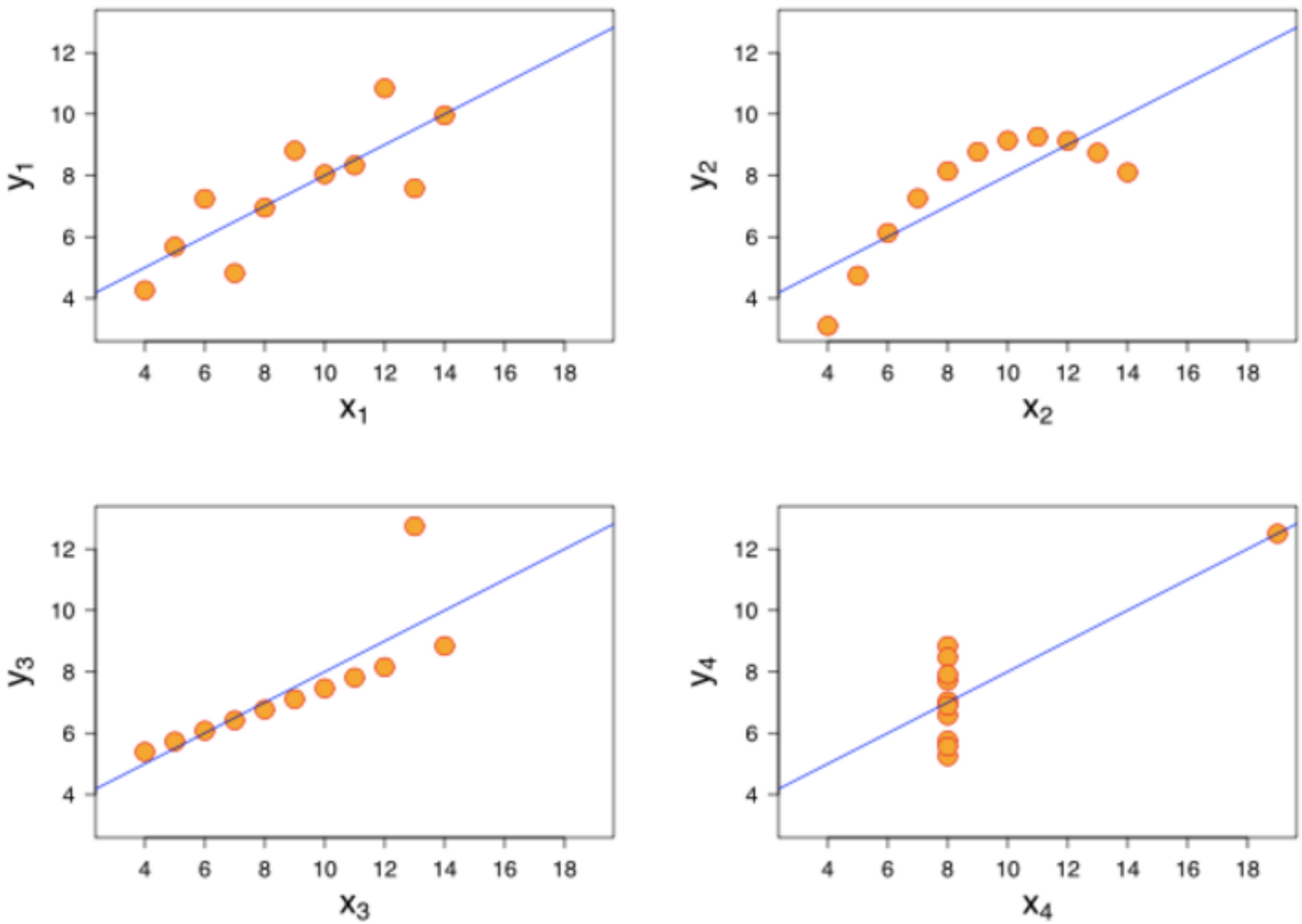
Linear regression



Nonlinear regression



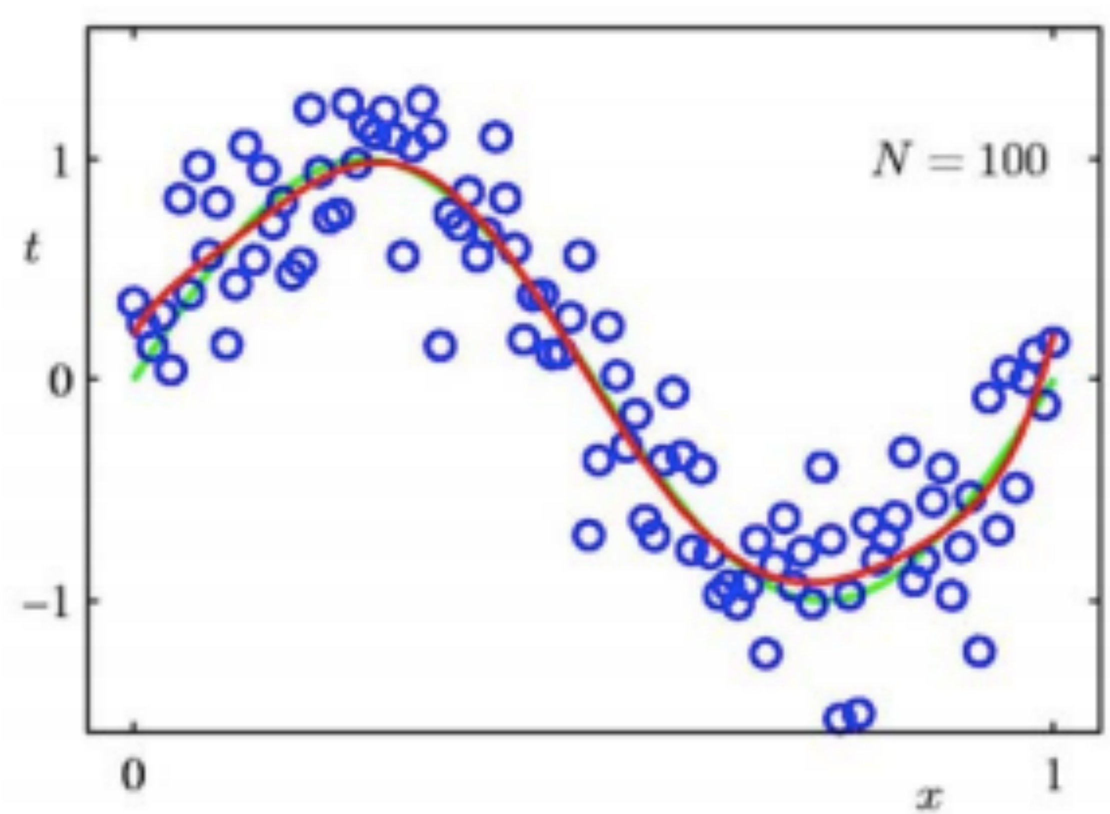
When does linear regression work?



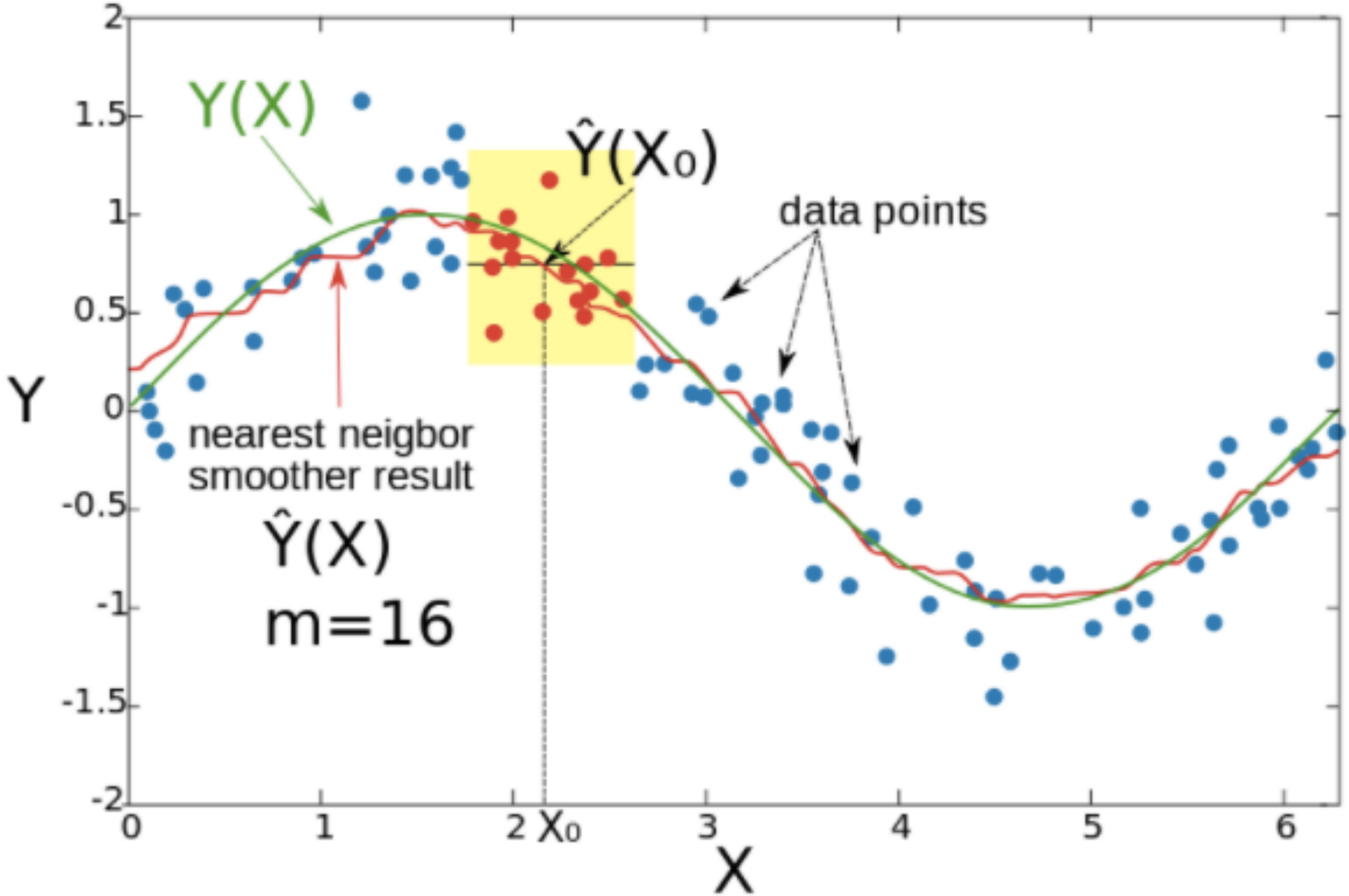
K nearest neighbor regression

Algorithm

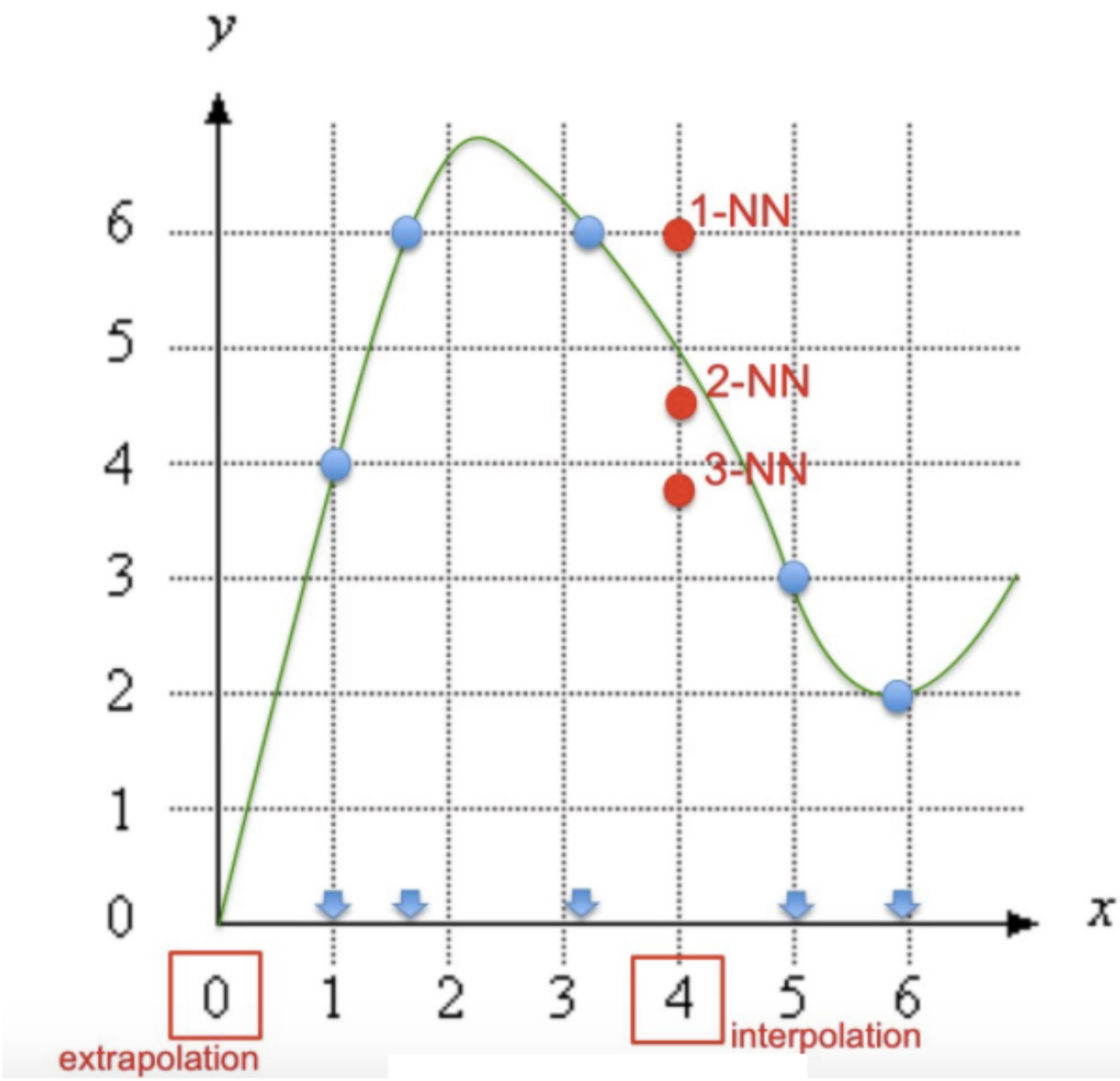
- For each test point, x, find the K nearest samples x_i in the training data and their values y_i
- Output is mean of their values $f(\mathbf{x}) = \frac{1}{K}\sum_{i=1}^{K} y_i$
- Again, need to choose (learn) K



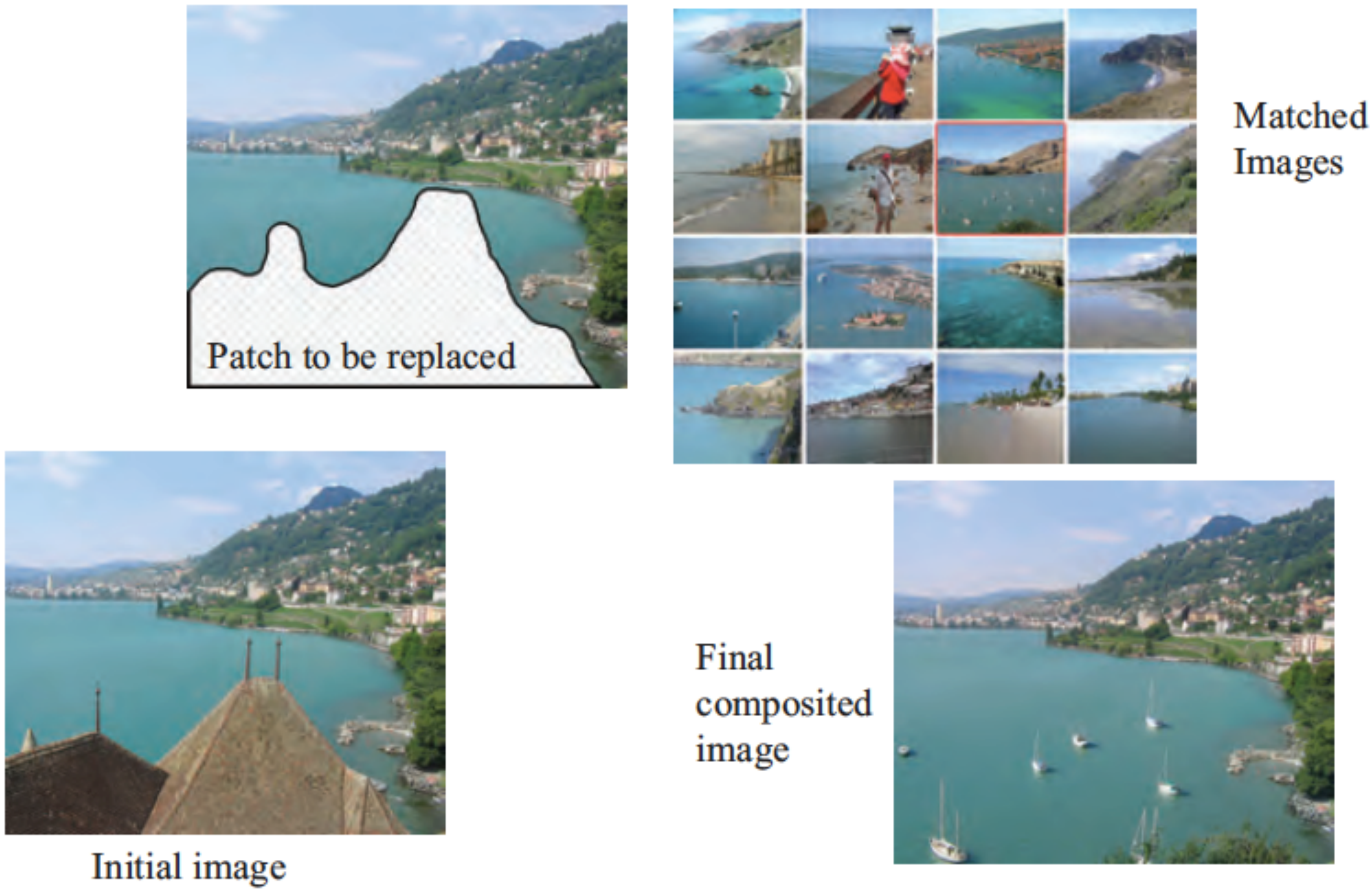
Nearest neighbor regression



Nearest neighbor regression

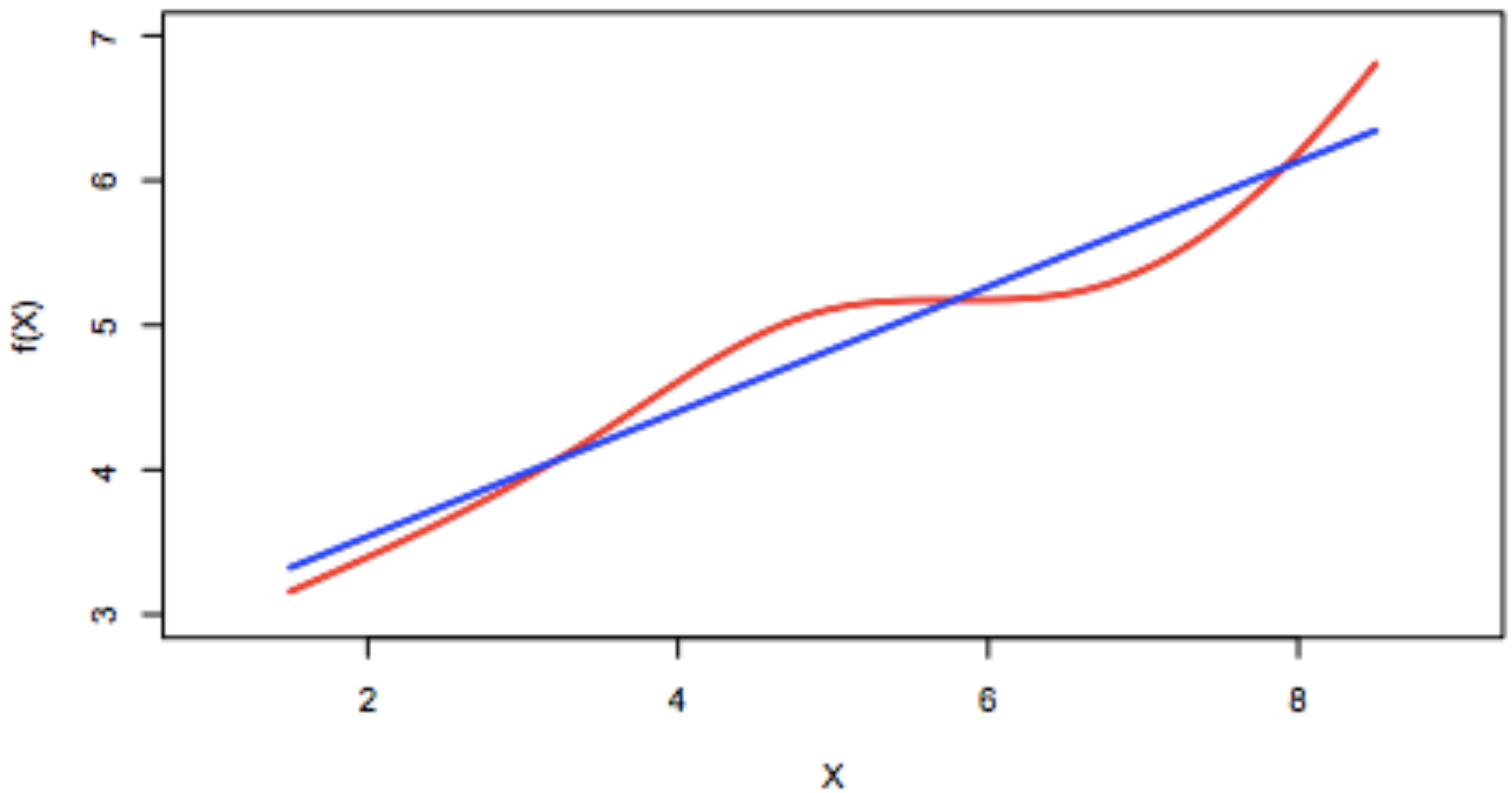


Filling patches in images



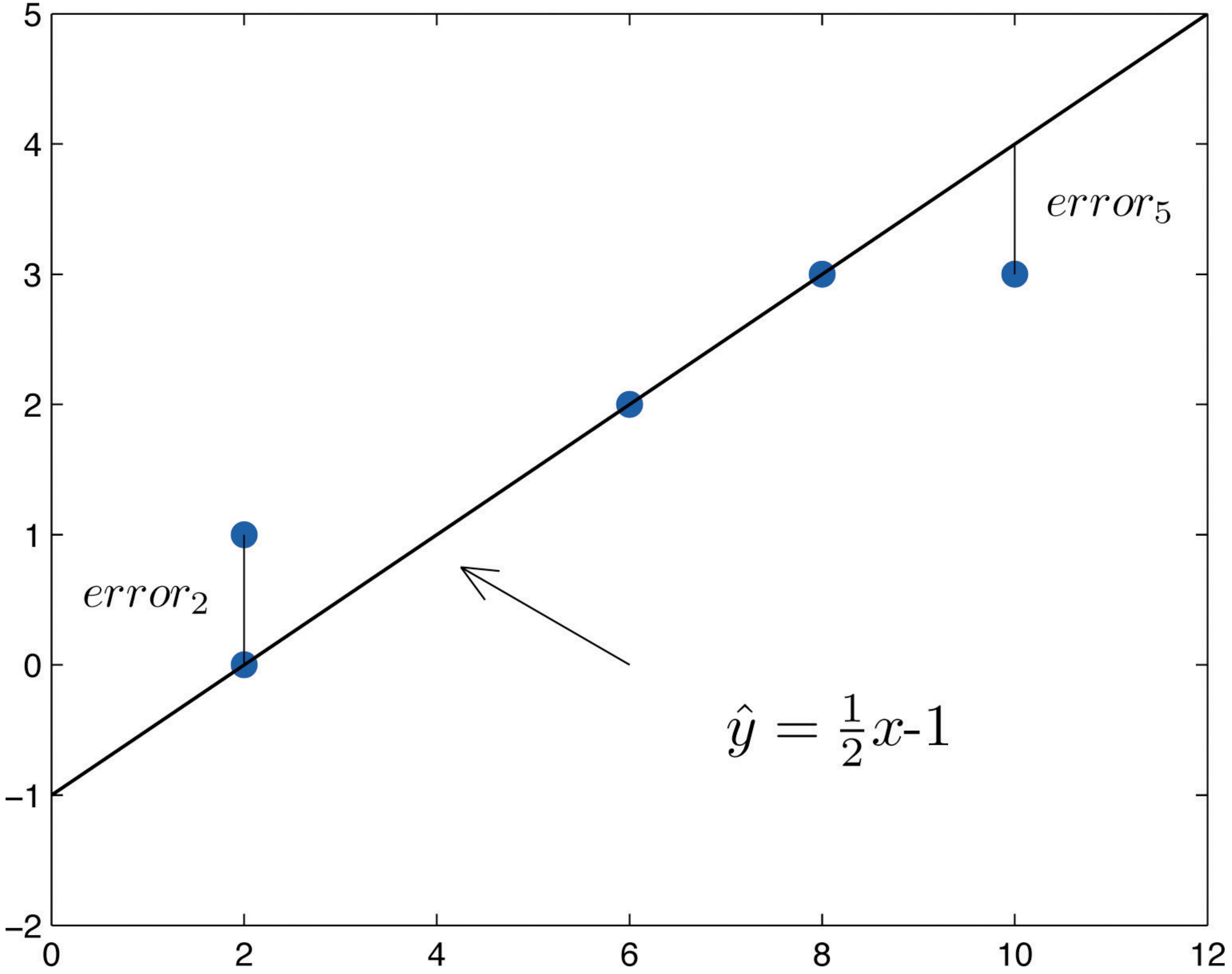
Linear regression

- Linear regression is a simple approach to supervised learning. It assumes that the dependence of Y on $X_1, X_2, \ldots X_p$ is linear.
- True regression functions are never linear!

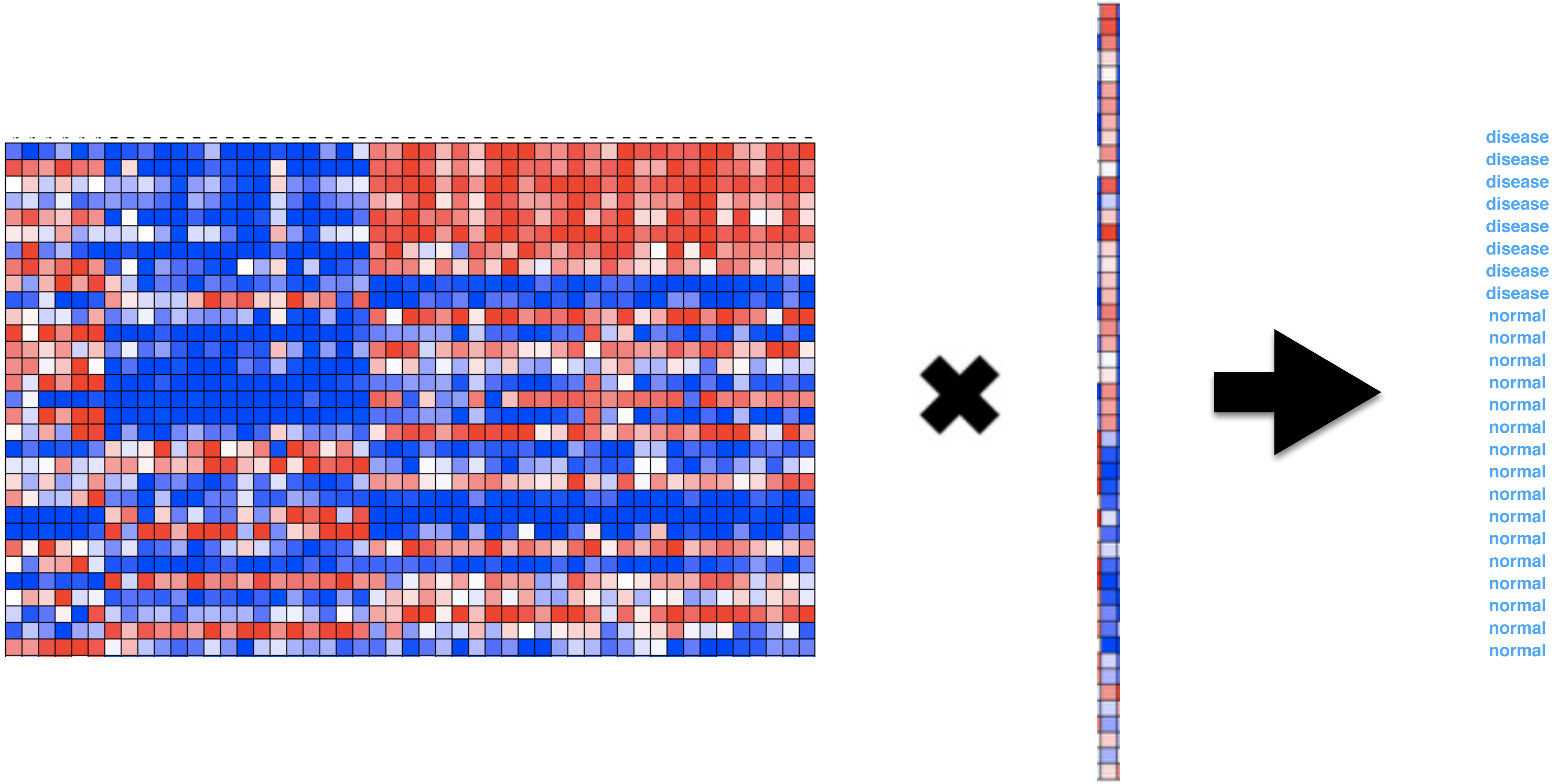


- although it may seem overly simplistic, linear regression is extremely useful both conceptually and practically.

How to measure the accuracy of linear regression models



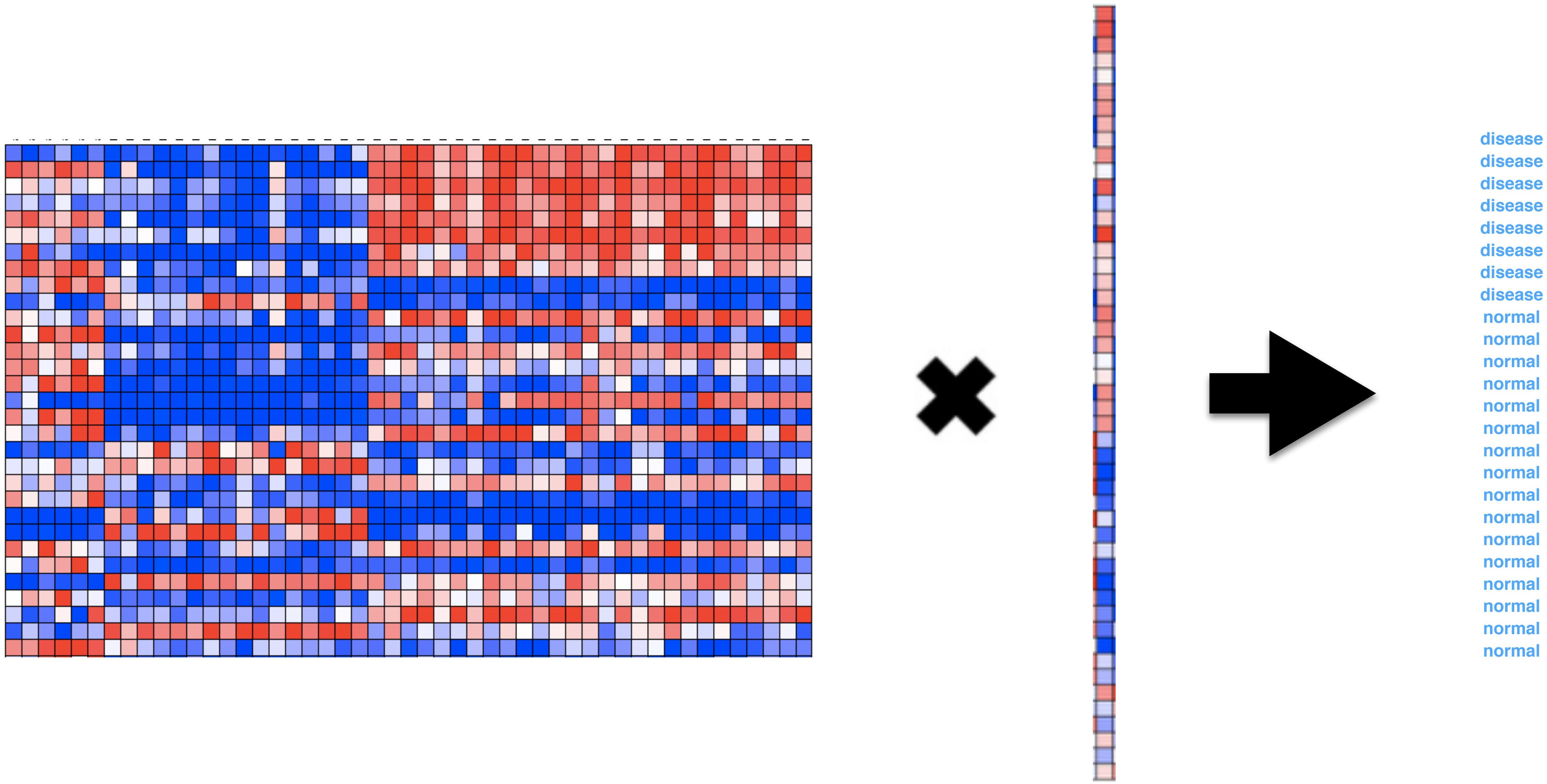
Linear Regression



$$y_i = x_i^T \beta + \epsilon_i = \sum_j \beta_j X_{i,j} + \epsilon_i$$

Fitting error: ϵ_i

Linear Regression



Assumption: errors are Gaussian noises

$$y = X\beta + \epsilon$$

$$\beta^* = \arg\min_{\beta} \sum_i (y_i - \sum_j \beta_j X_{i,j})^2$$

Linear Regression

$$\beta^* = \arg\min_\beta \sum_i (y_i - \sum_j \beta_j X_{i,j})^2$$

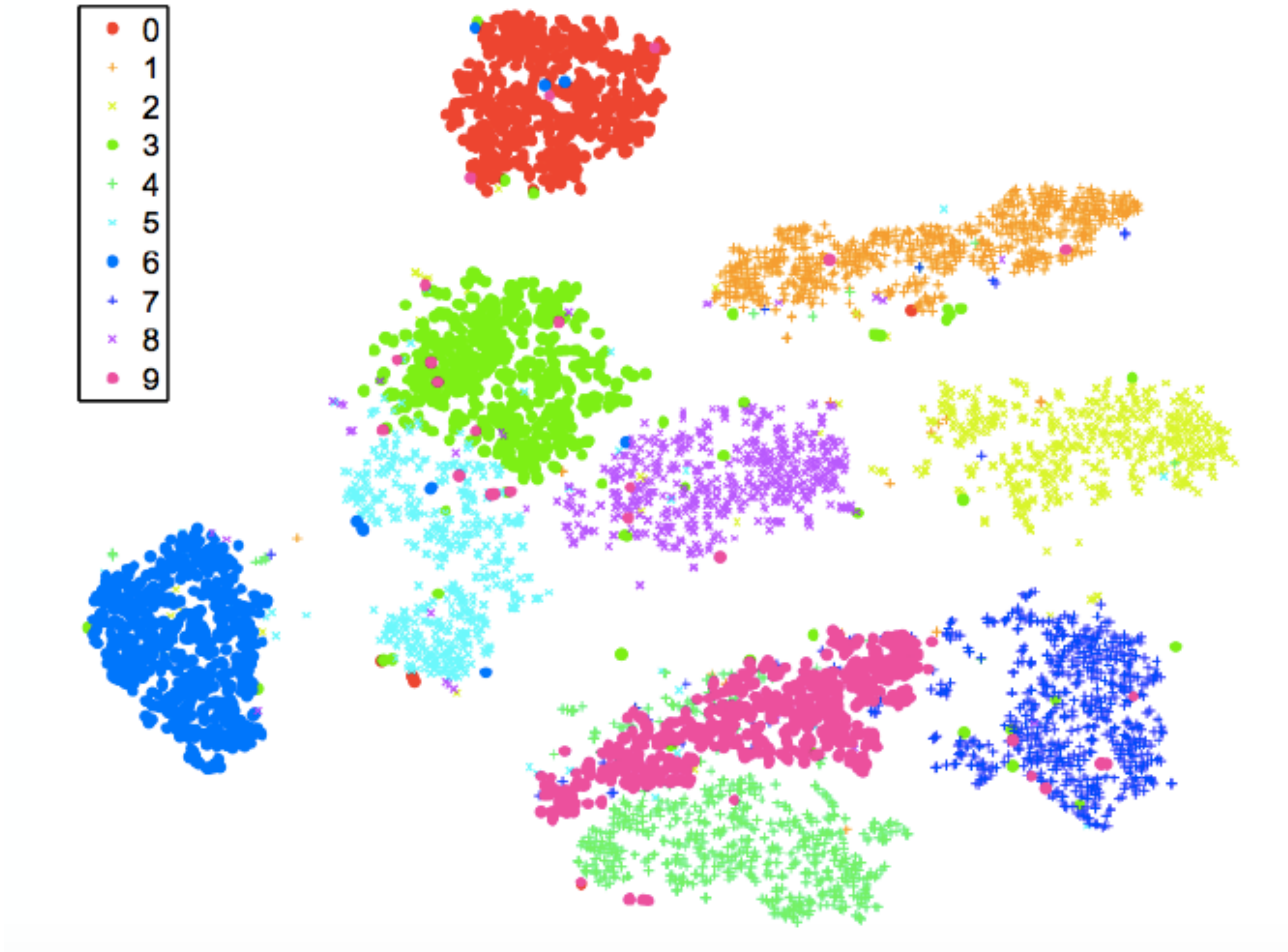
$$= \arg\min_\beta (y - X\beta)^T (y - X\beta)$$

$$= (X^T X)^{-1} X^T y$$

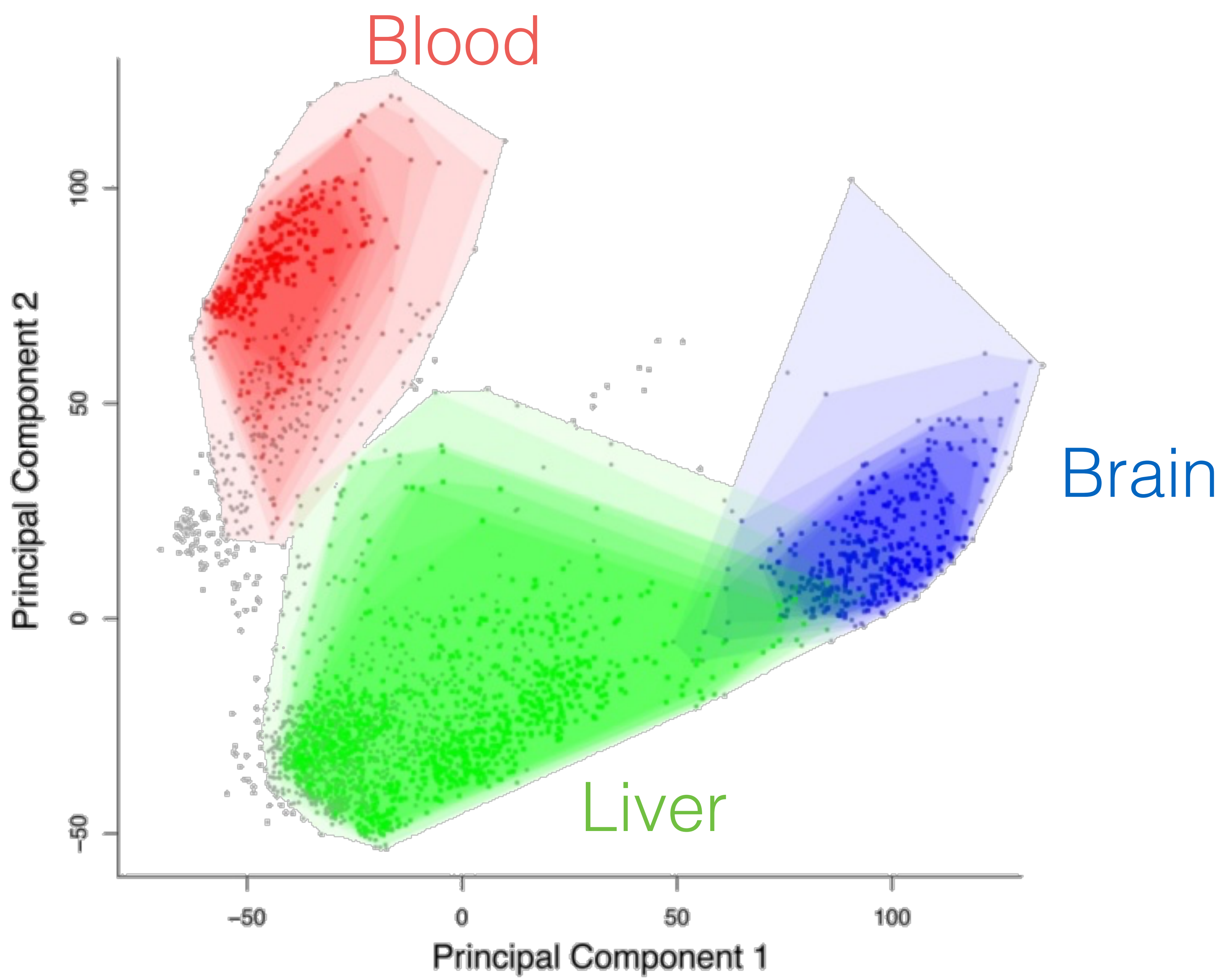
Question: How to derive the closed-form solution?

Clustering

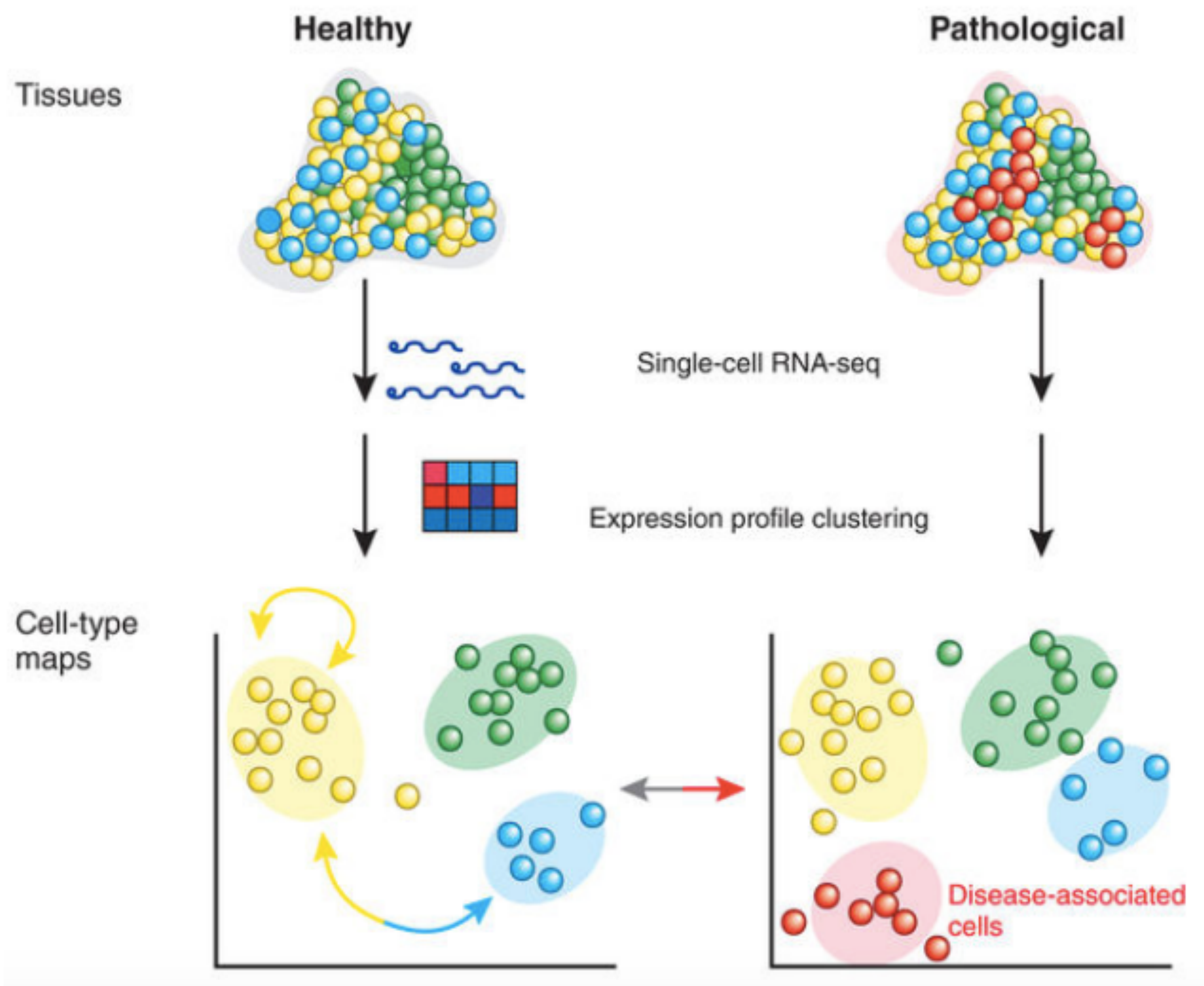
Finding hidden structure in data



Expression analysis



Single-cell expression analysis



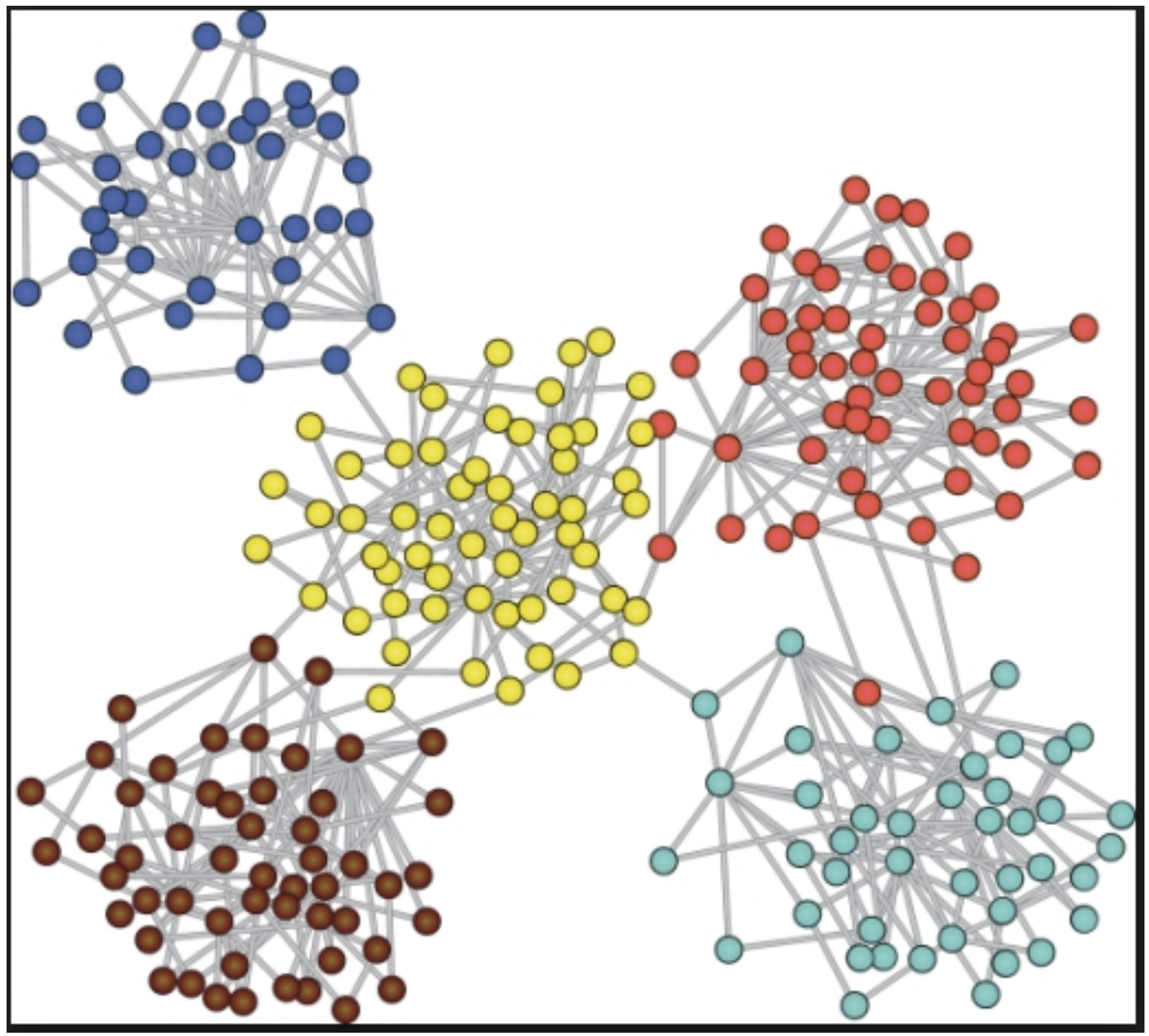
Clustering: examples

Image segmentation

Goal: Break up the image into meaningful or perceptually similar regions

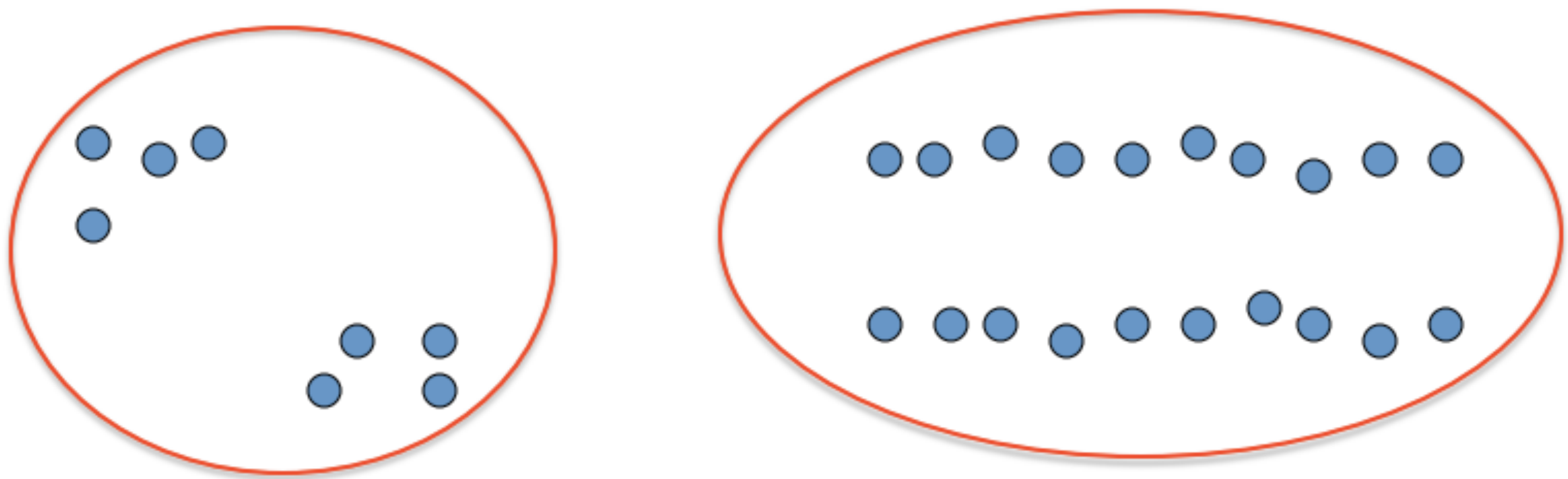
[Slide from James Hayes]

Network clustering



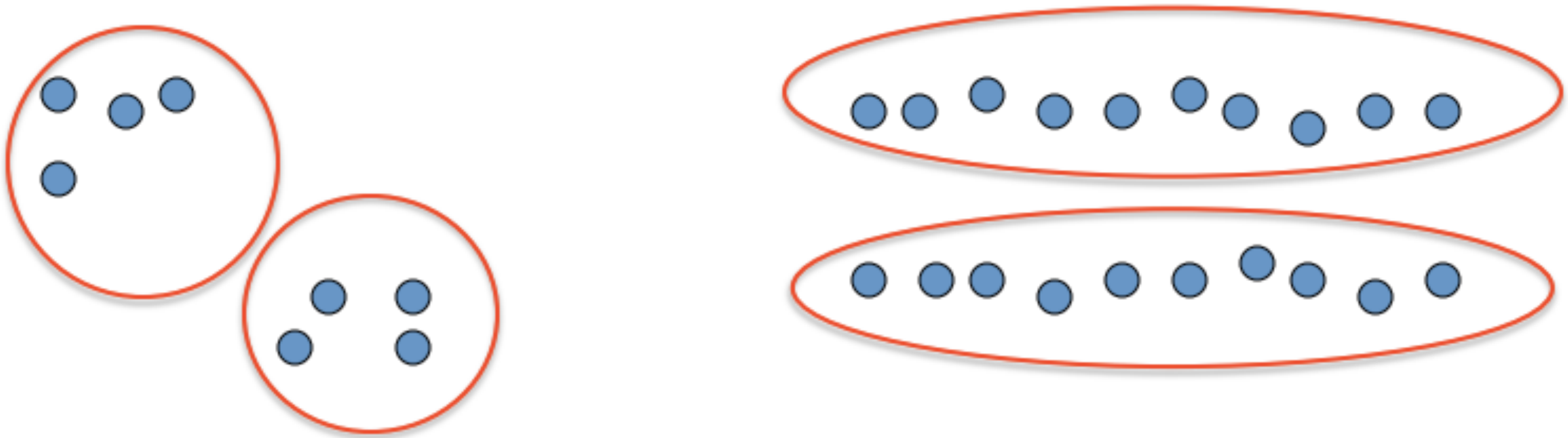
Clustering

- Basic idea: group together similar instances
- Example: 2D point patterns



Clustering

- Basic idea: group together similar instances
- Example: 2D point patterns

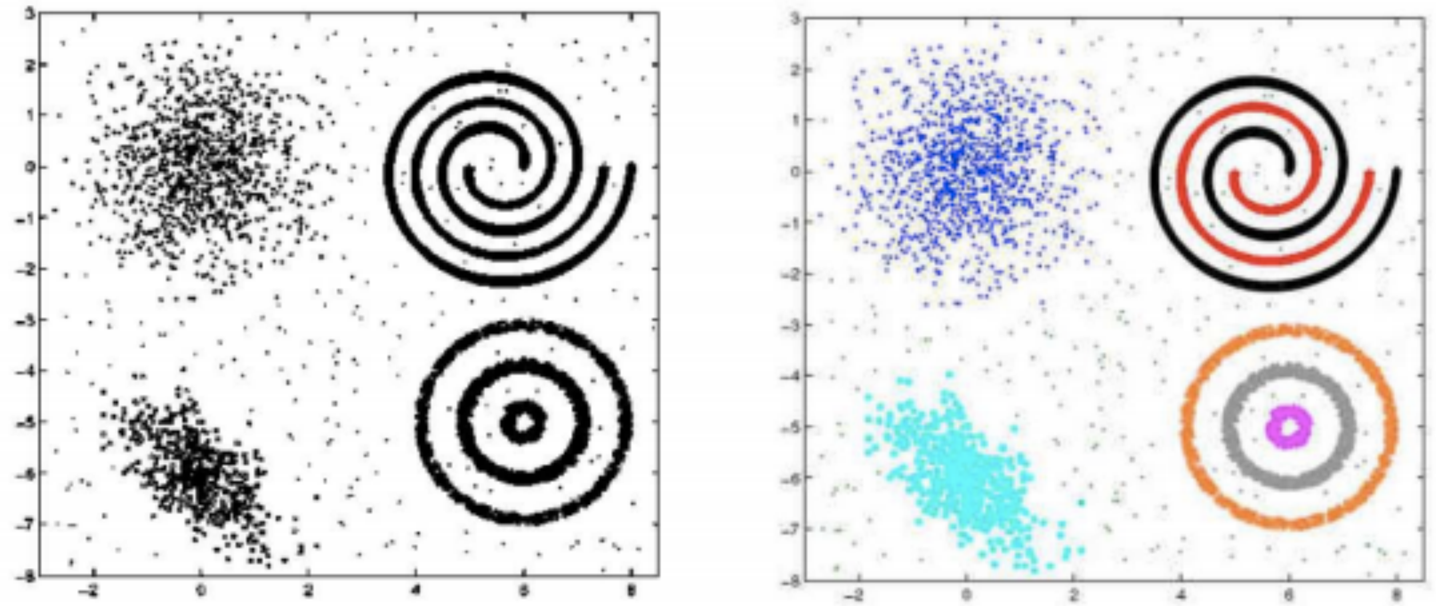


- What could "similar" mean?
  - One option: small Euclidean distance (squared)

$$\text{dist}(\vec{x}, \vec{y}) = ||\vec{x} - \vec{y}||_2^2$$

  - Clustering results are crucially dependent on the measure of similarity (or distance) between "points" to be clustered

- Given: N unlabeled examples $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$; the number of partitions K
- Goal: Group the examples into K partitions

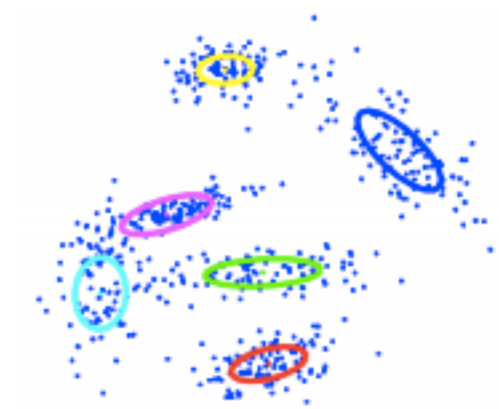


(a) Input data (b) Desired clustering

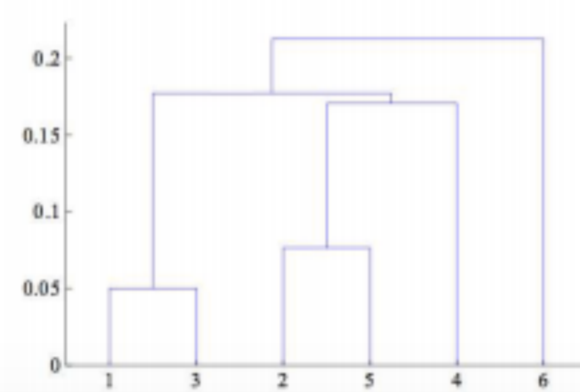
- The only information clustering uses is the similarity between examples
- Clustering groups examples based of their mutual similarities

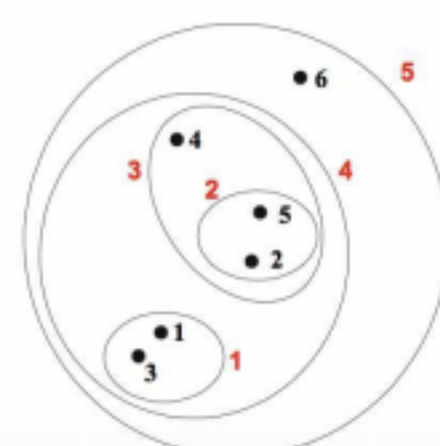
Clustering algorithms

1. **Flat or Partitional clustering** (*K*-means, Gaussian mixture models, etc.)
   - Partitions are independent of each other



2. **Hierarchical clustering** (e.g., agglomerative clustering, divisive clustering)
   - Partitions can be visualized using a tree structure (a dendrogram)
   - Does not need the number of clusters as input
   - Possible to view partitions at different levels of granularities (i.e., can refine/coarsen clusters) using different *K*





K-means

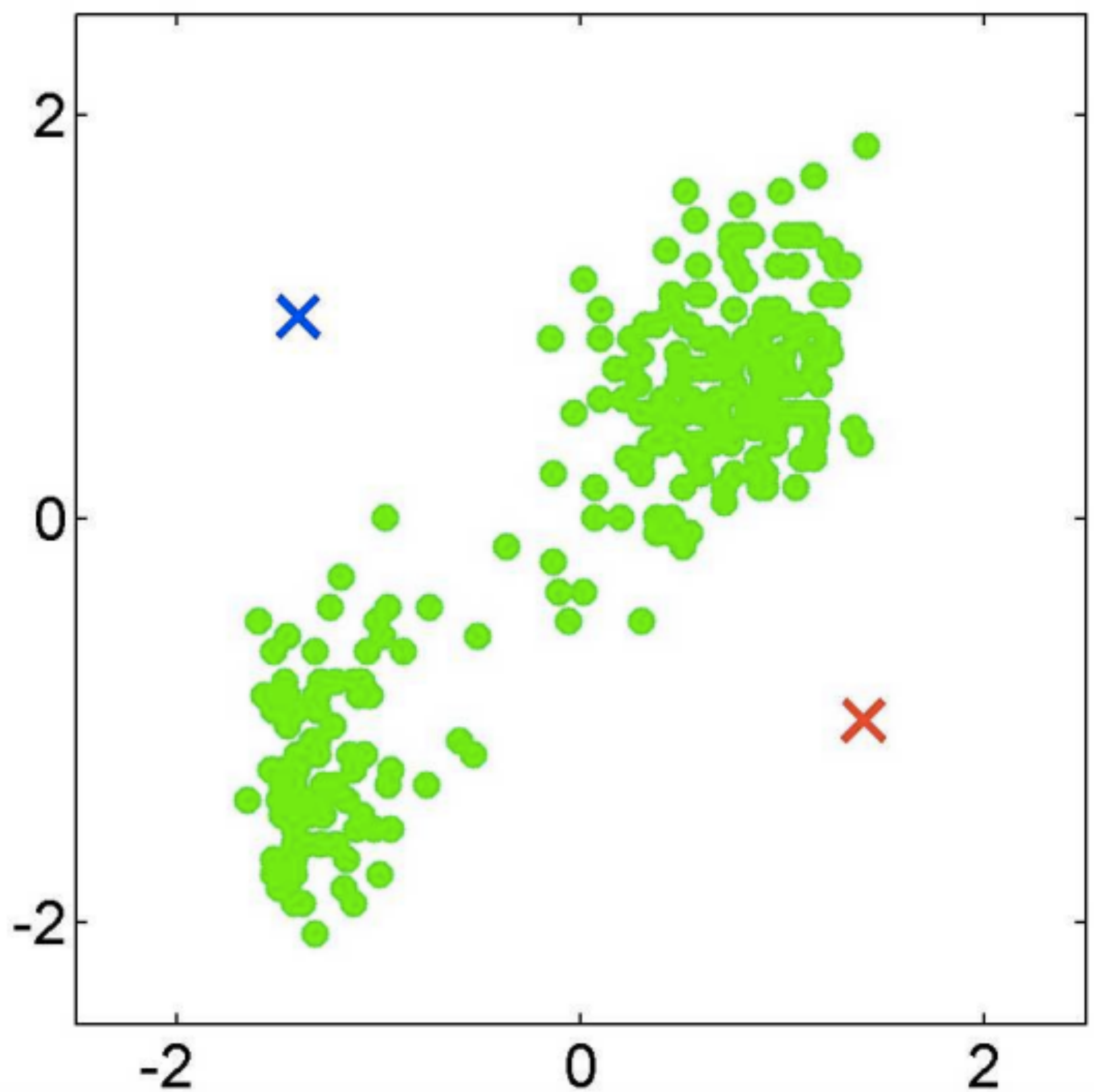
- **Input:** N examples $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ ($\mathbf{x}_n \in \mathbb{R}^D$); the number of partitions K
- **Initialize:** K cluster centers $\mu_1, \ldots, \mu_K$. Several initialization options:
  - Randomly initialized anywhere in $\mathbb{R}^D$
  - Choose any K examples as the cluster centers
- **Iterate:**
  - Assign each of example $\mathbf{x}_n$ to its closest cluster center

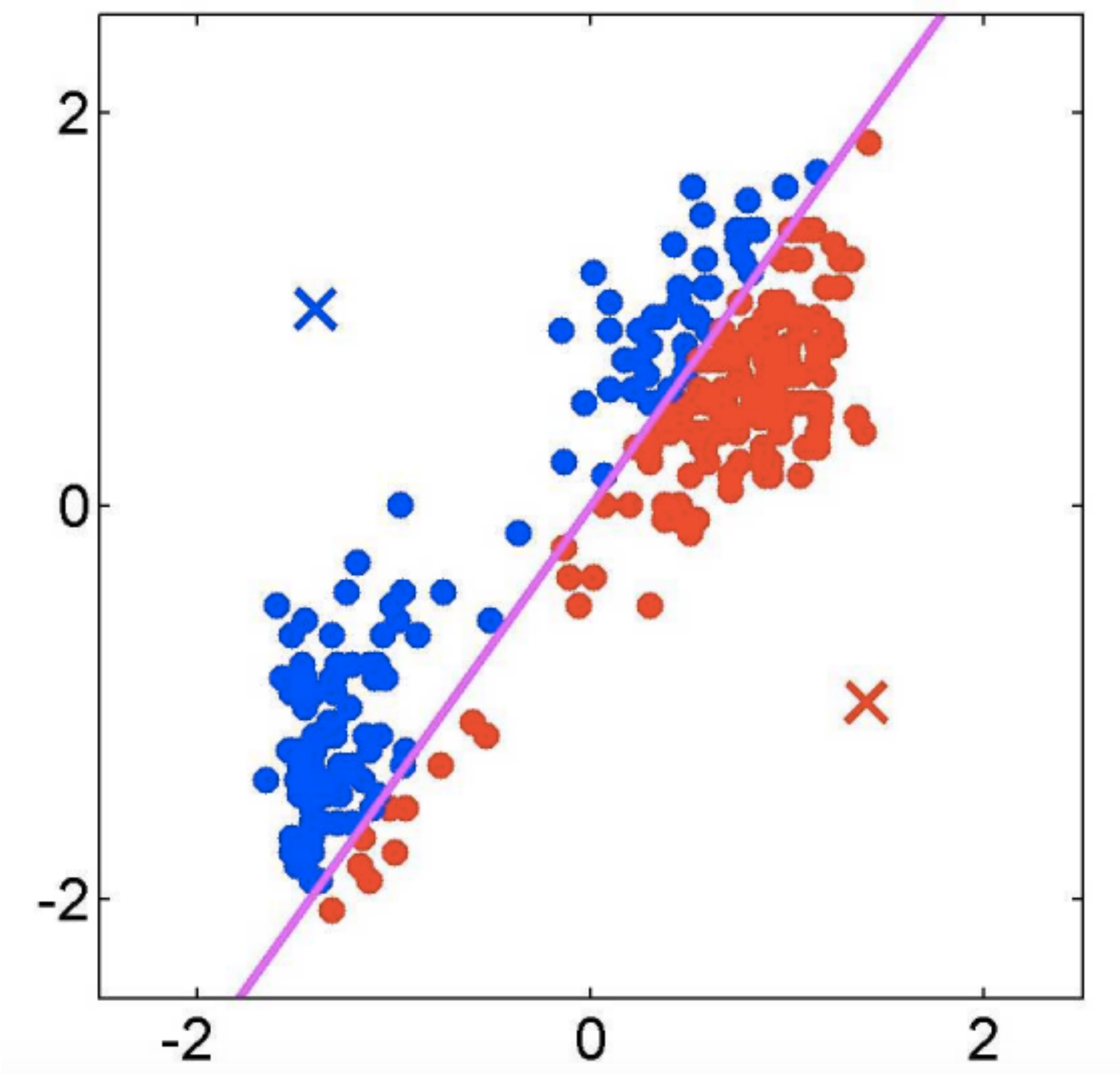
$$\mathcal{C}_k = \{n : \quad k = \arg\min_k ||\mathbf{x}_n - \mu_k||^2\}$$

    ($\mathcal{C}_k$ is the set of examples closest to μ_k)
  - Recompute the new cluster centers μ_k (mean/centroid of the set $\mathcal{C}_k$)

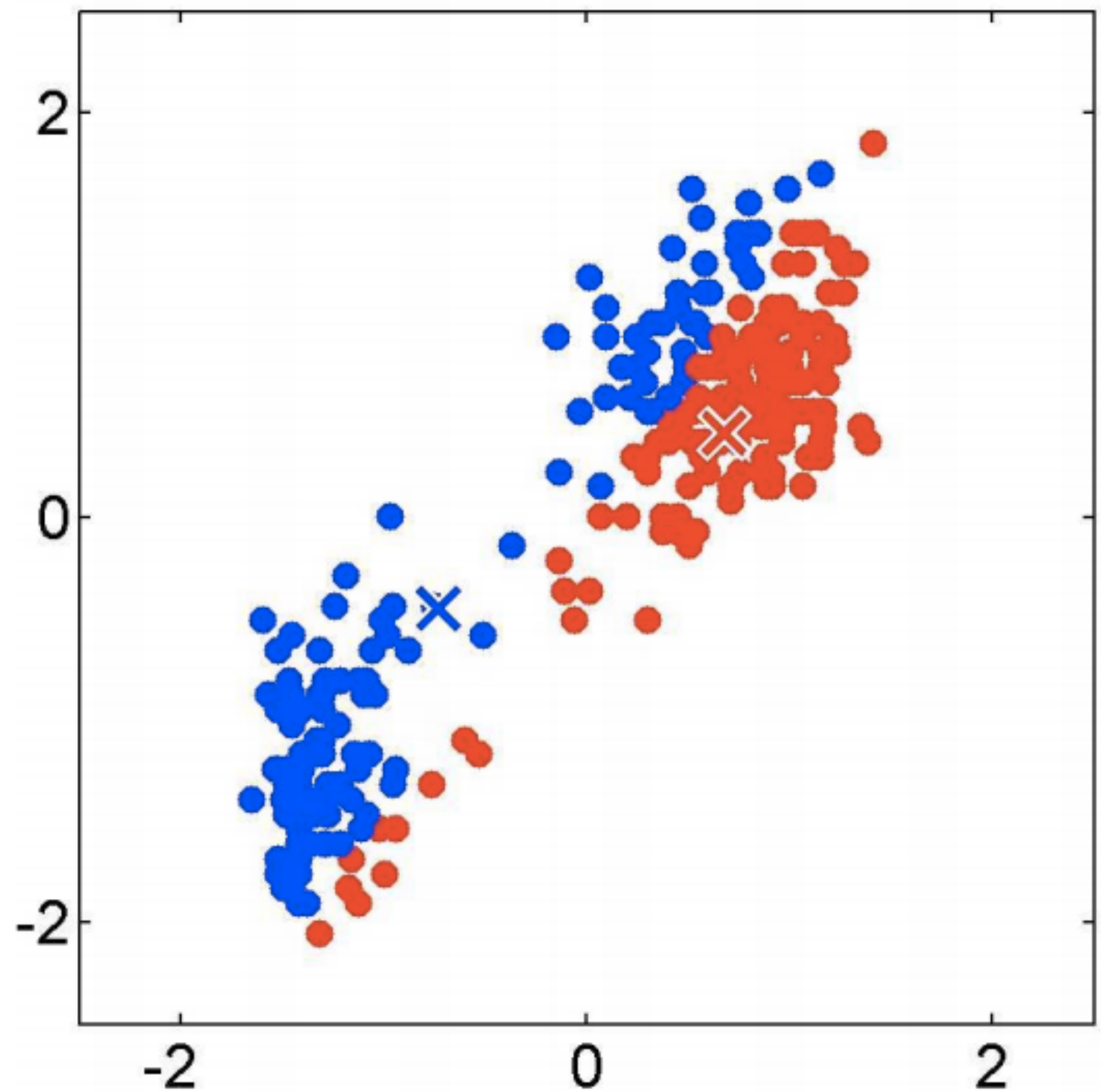
$$\mu_k = \frac{1}{|\mathcal{C}_k|} \sum_{n \in \mathcal{C}_k} \mathbf{x}_n$$

  - Repeat while not converged

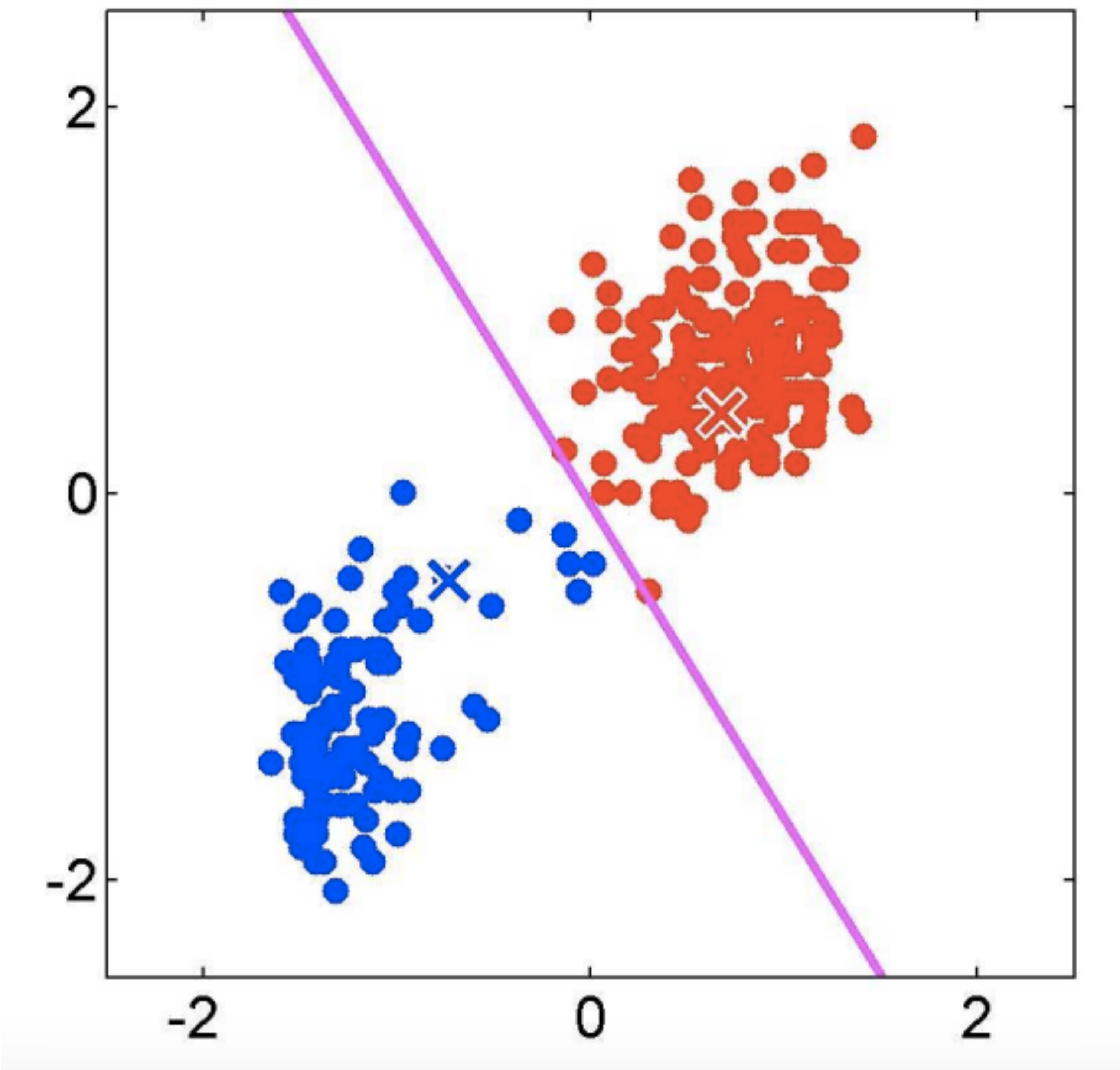

2
0
-2
-2
0
2



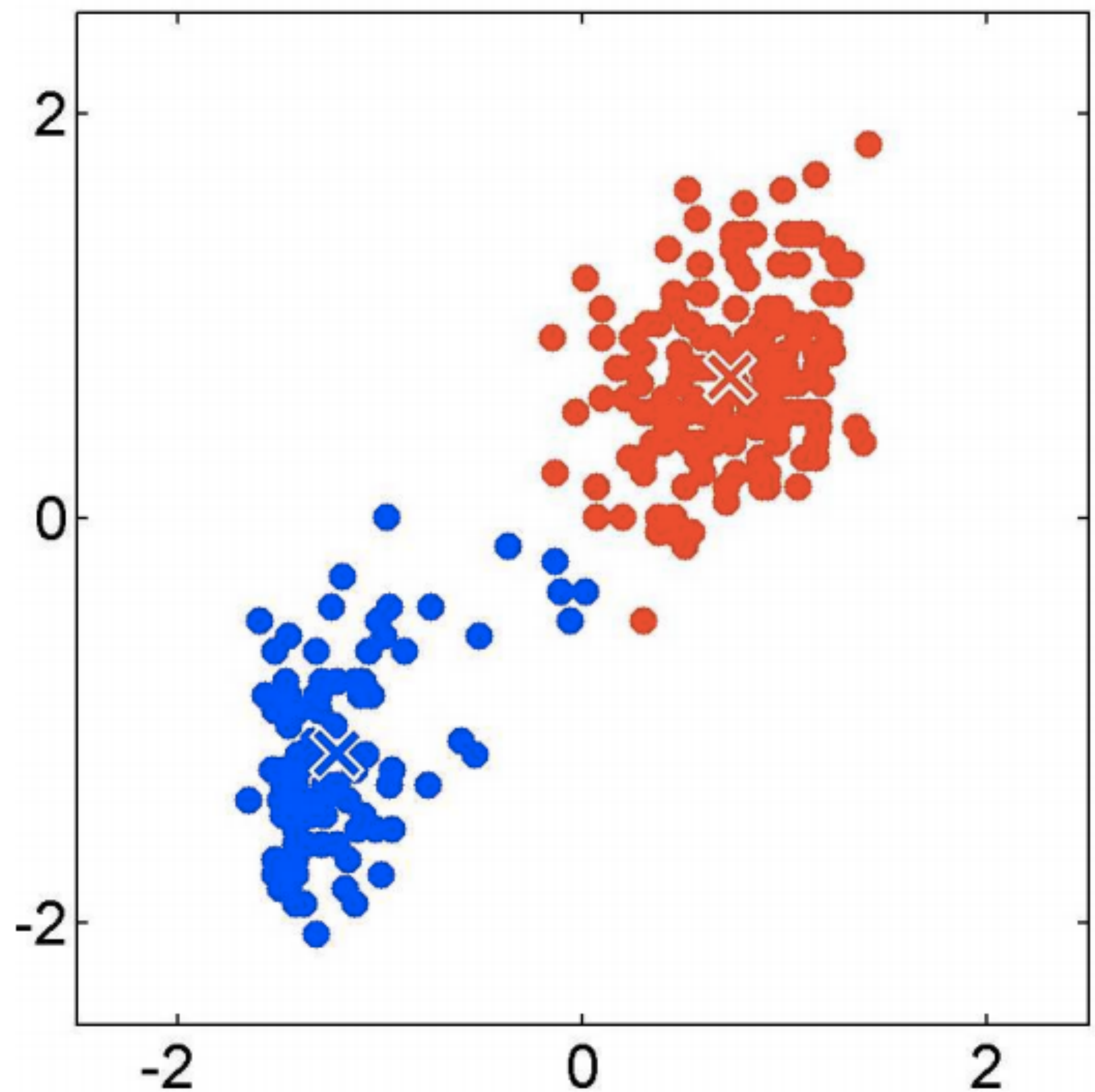
2
0
-2
-2
0
2



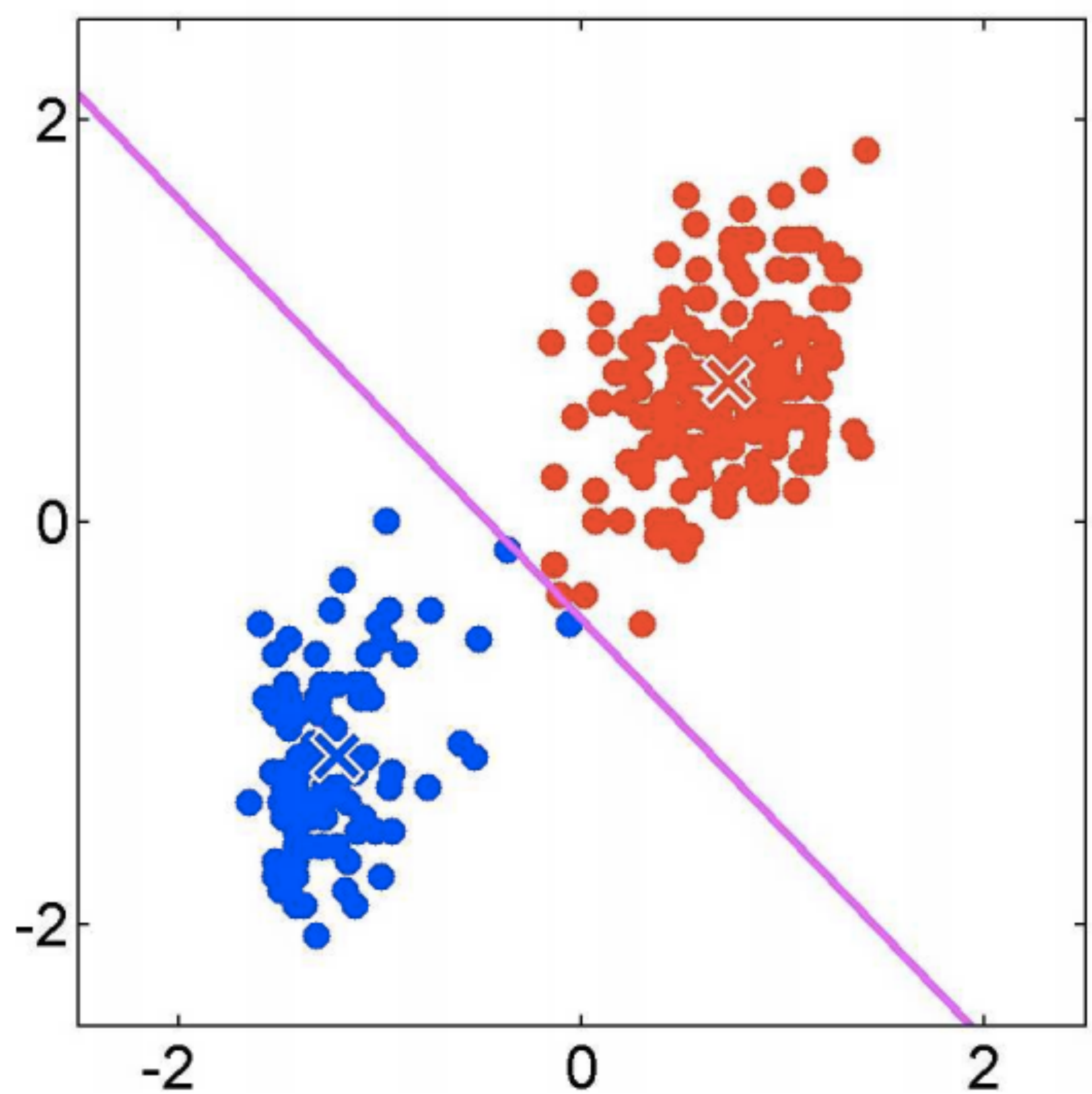
2
0
-2
-2
0
2



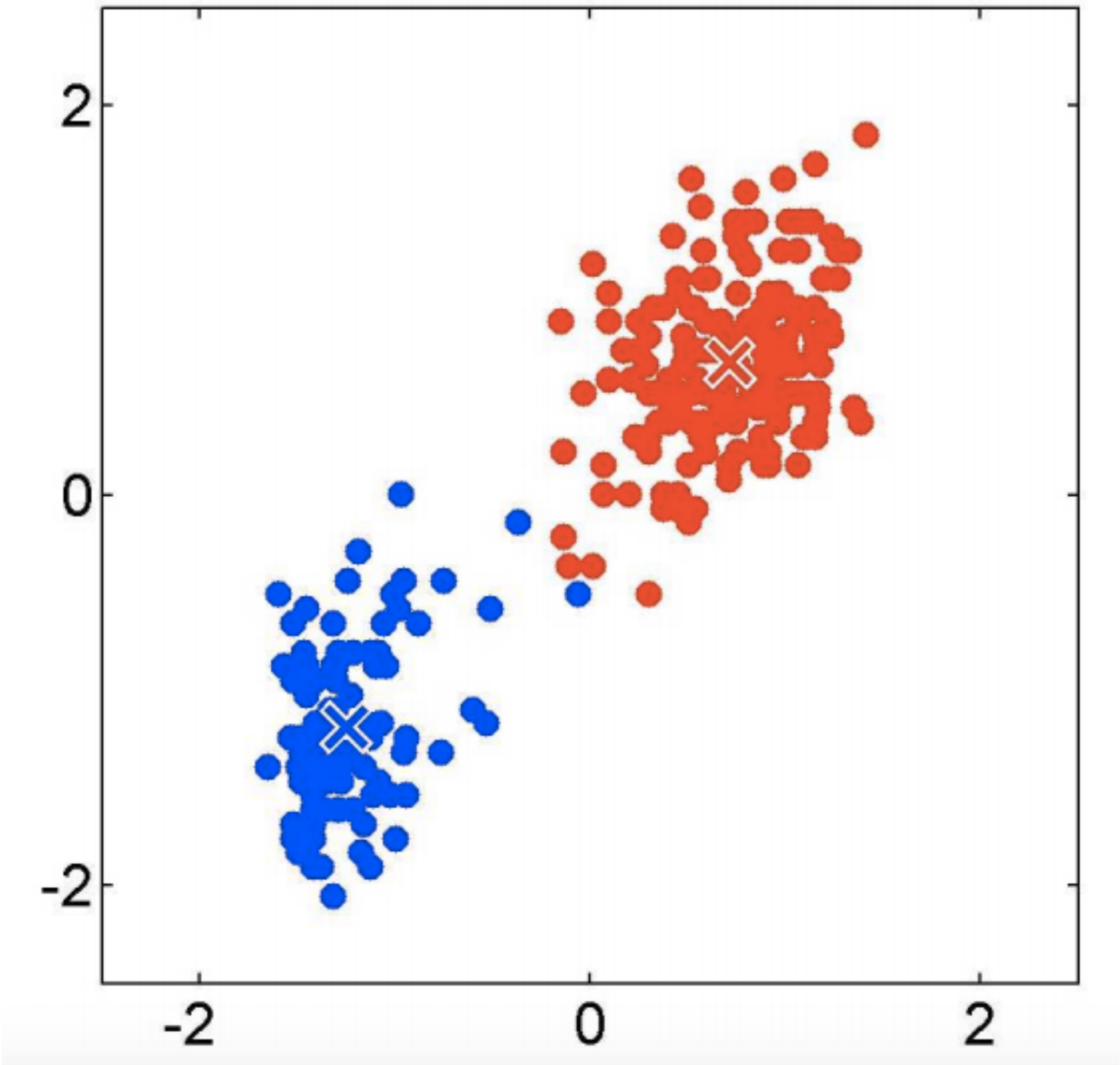
2
0
-2
-2
0
2

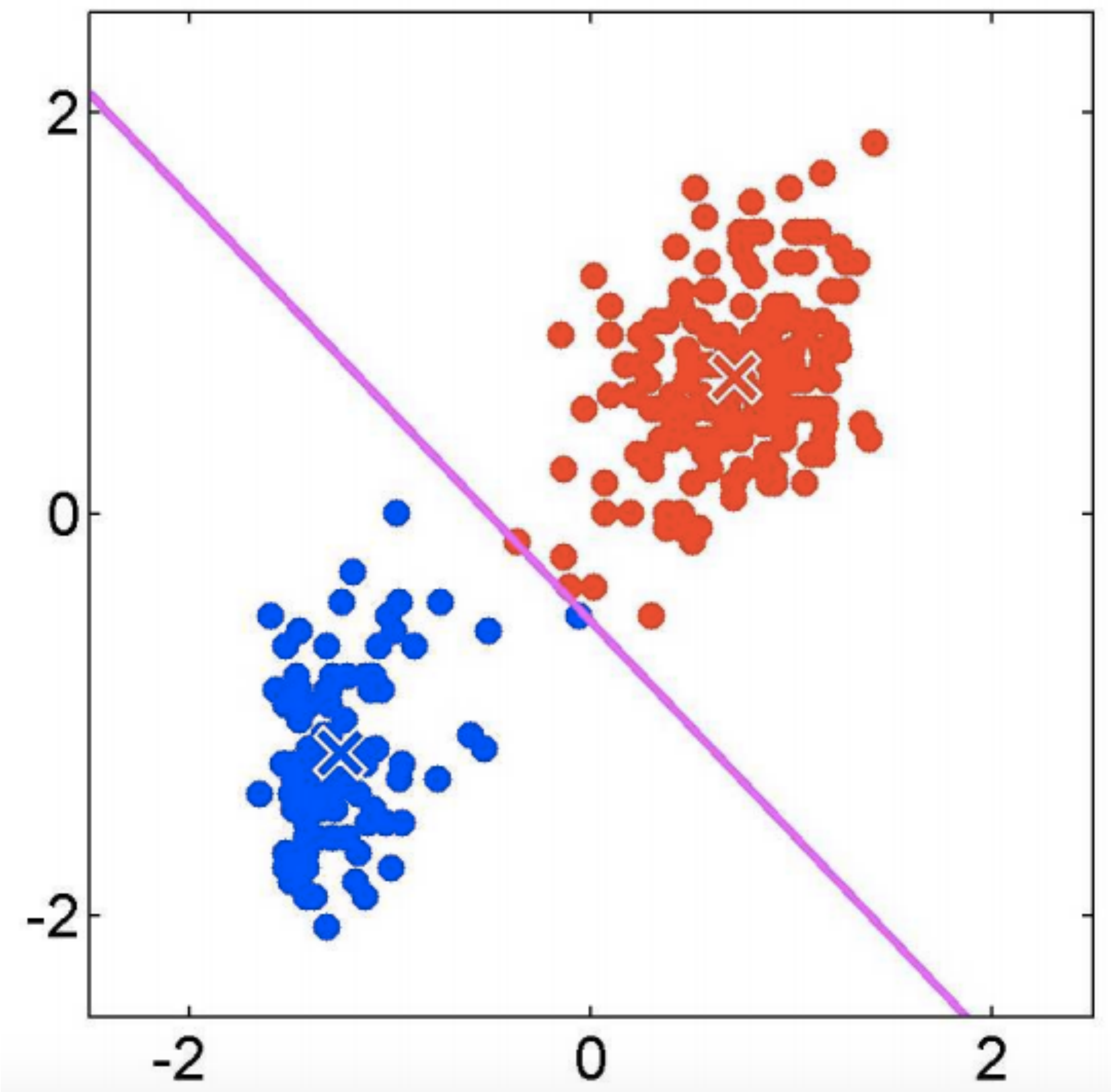


2
0
-2
-2
0
2

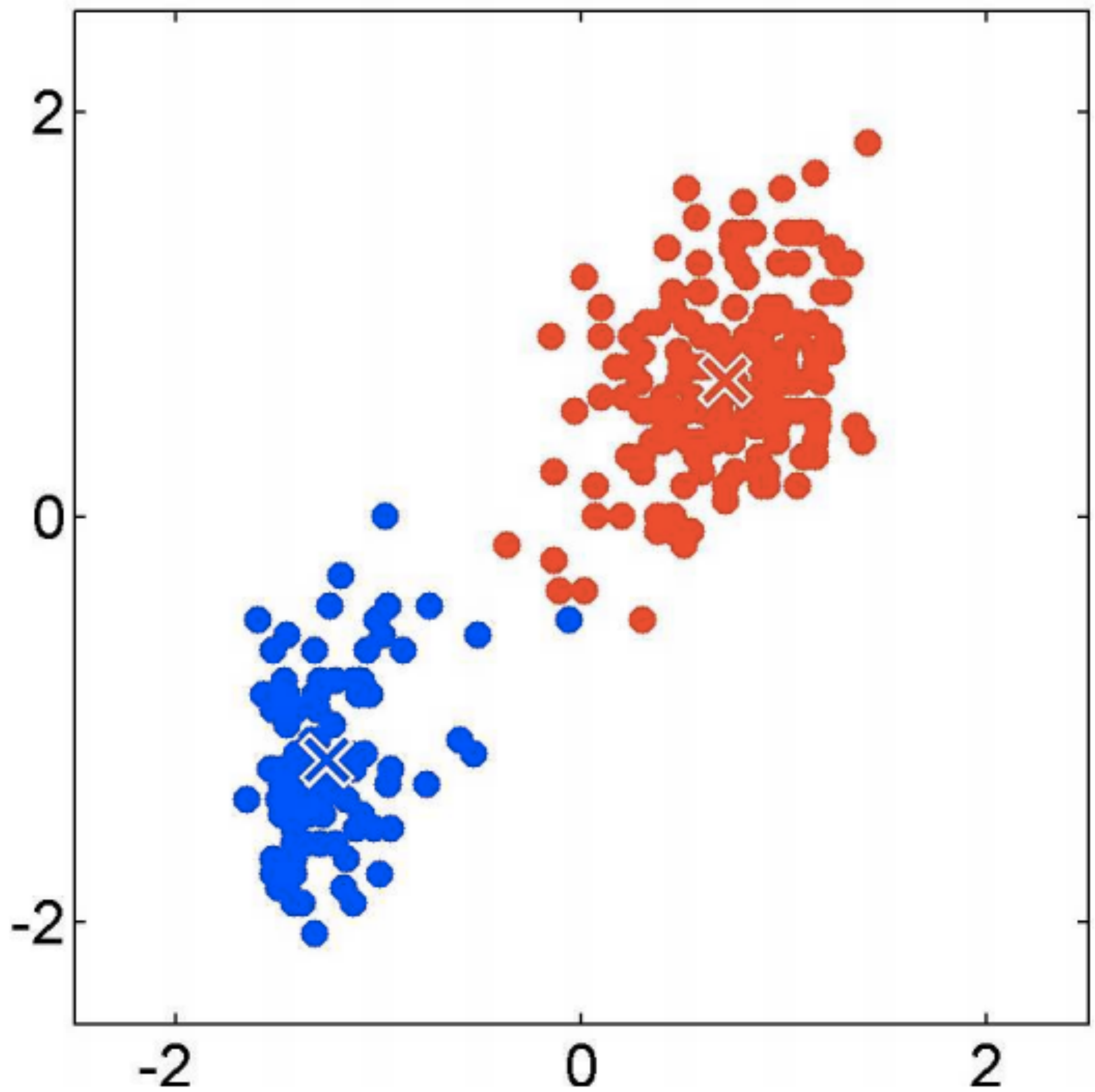


2
0
-2
-2
0
2


2
0
-2
-2
0
2



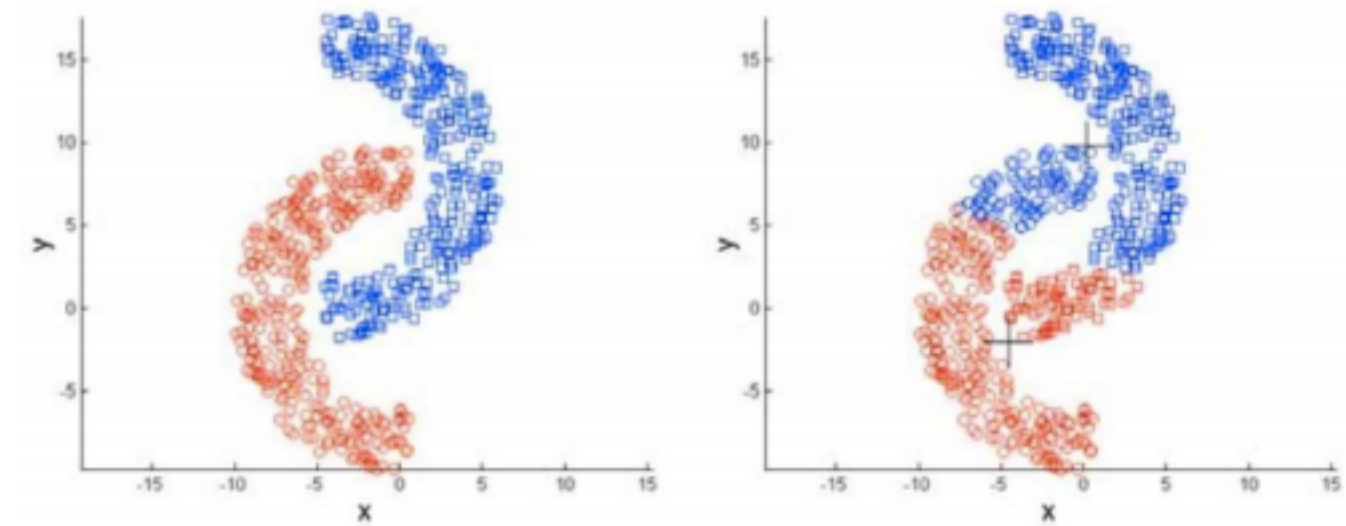
2
0
-2
-2
0
2


2
0
-2
-2
0
2

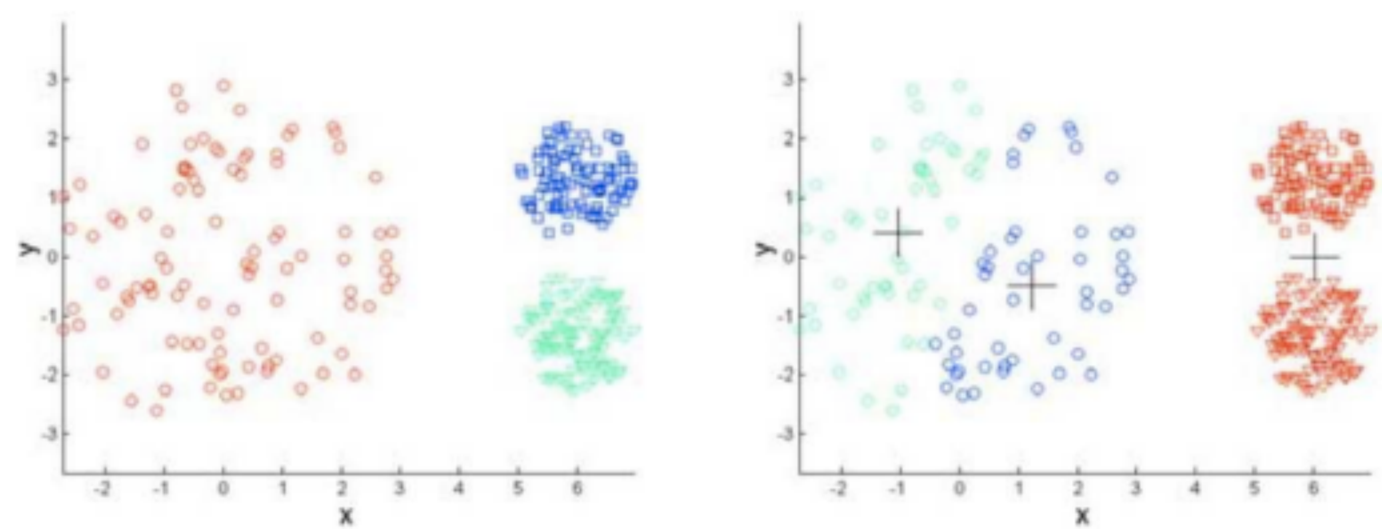
K-means for segmentation

When will K-means fail?

Non-convex/non-round-shaped clusters: Standard *K*-means fails!

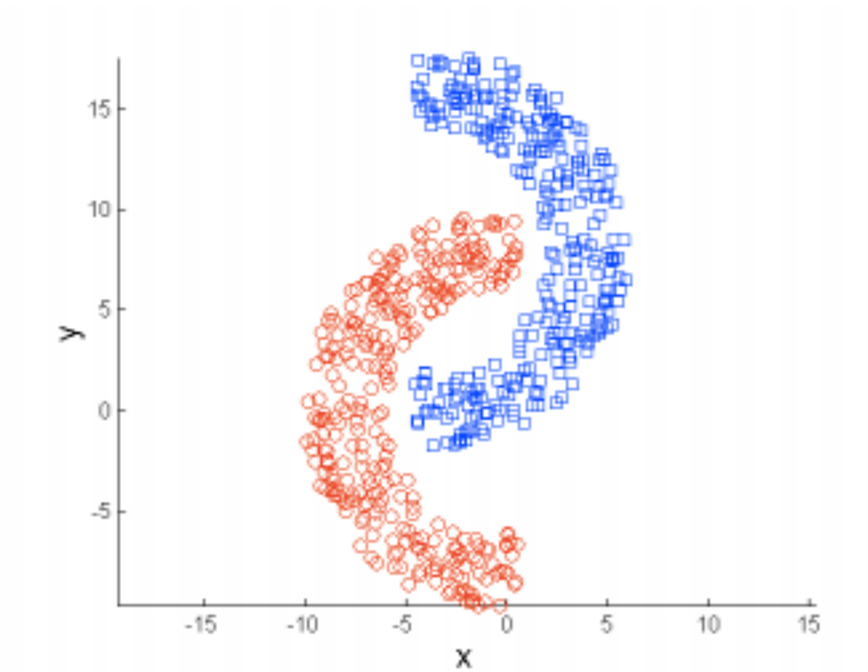


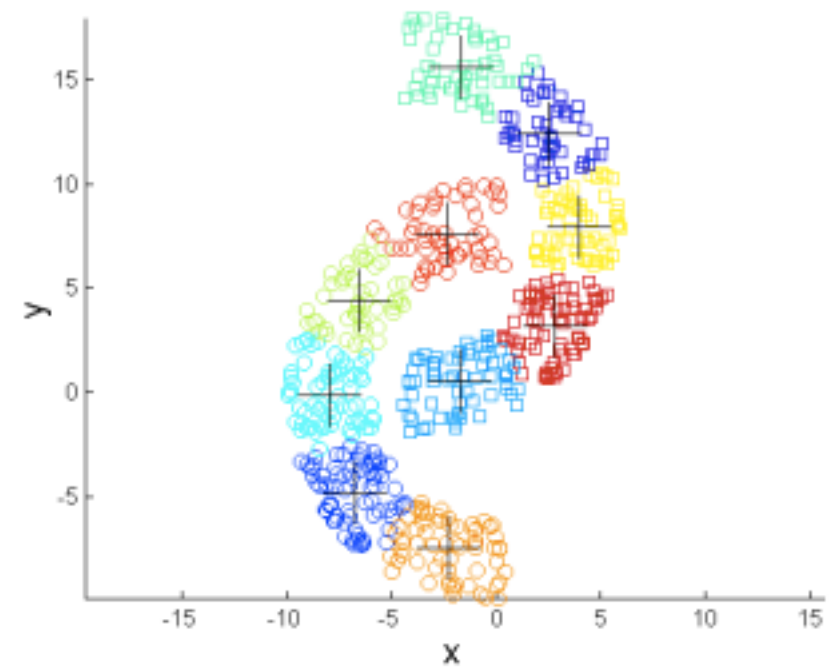
Clusters with different densities



Hierarchical clustering

A hierarchical approach can be useful when considering versatile cluster shapes:



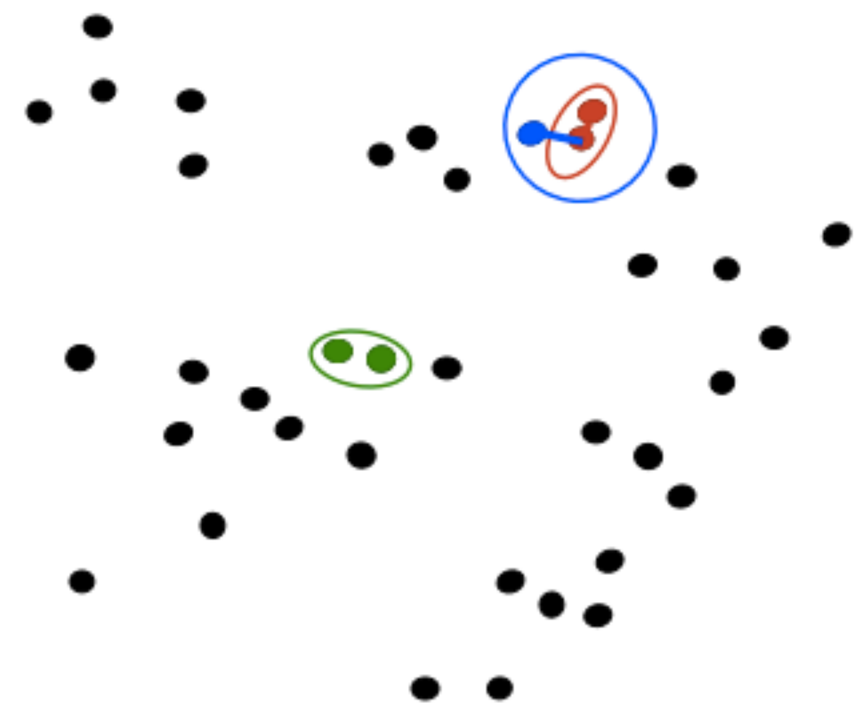


10-means

By first detecting many small clusters, and then merging them, we can uncover patterns that are challenging for partitional methods.

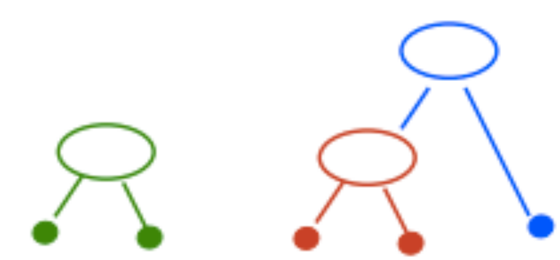
Agglomerative clustering

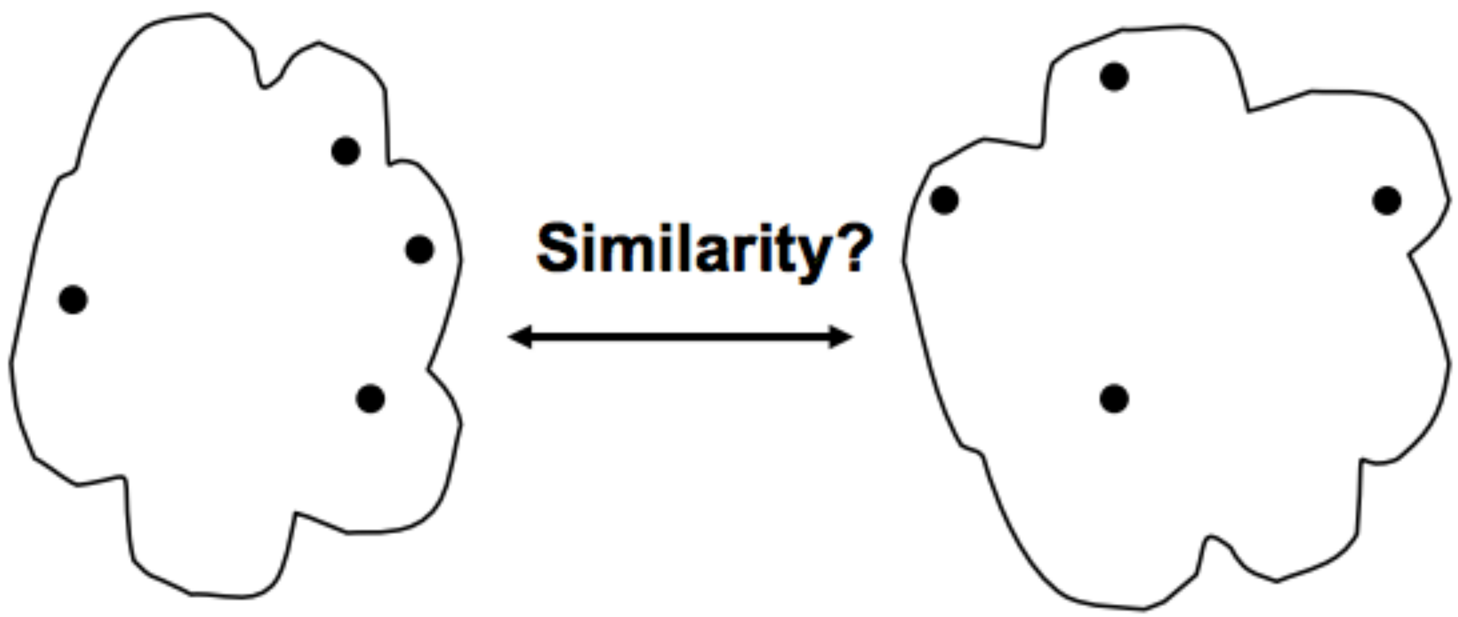
- Agglomerative clustering:
  - First merge very similar instances
  - Incrementally build larger clusters out of smaller clusters



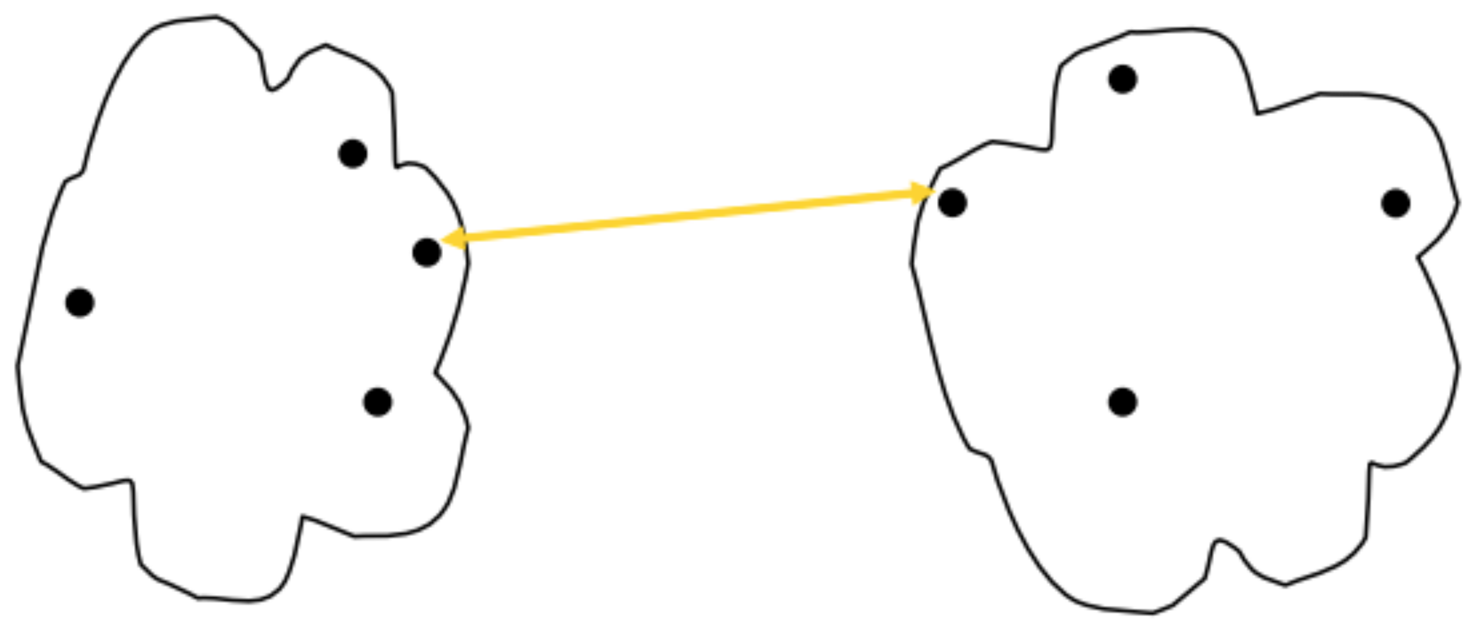
- Algorithm:
  - Maintain a set of clusters
  - Initially, each instance in its own cluster
  - Repeat:
    - Pick the two closest clusters
    - Merge them into a new cluster
    - Stop when there's only one cluster left

- Produces not one clustering, but a family of clusterings represented by a dendrogram



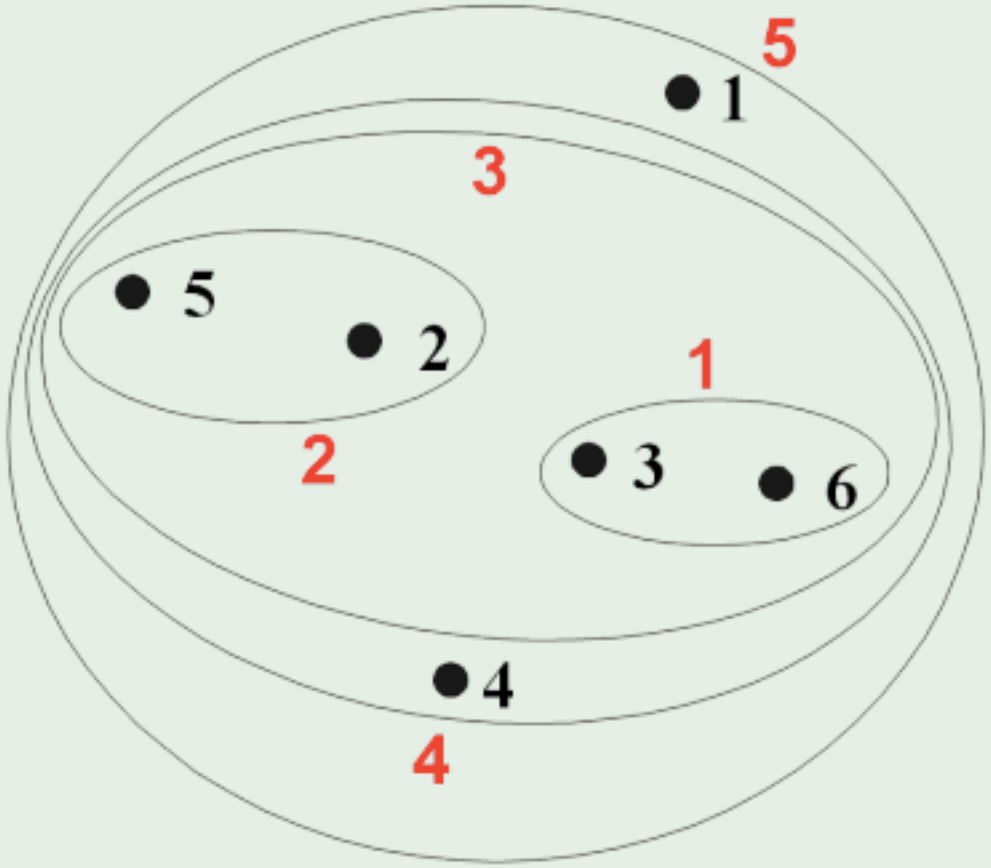


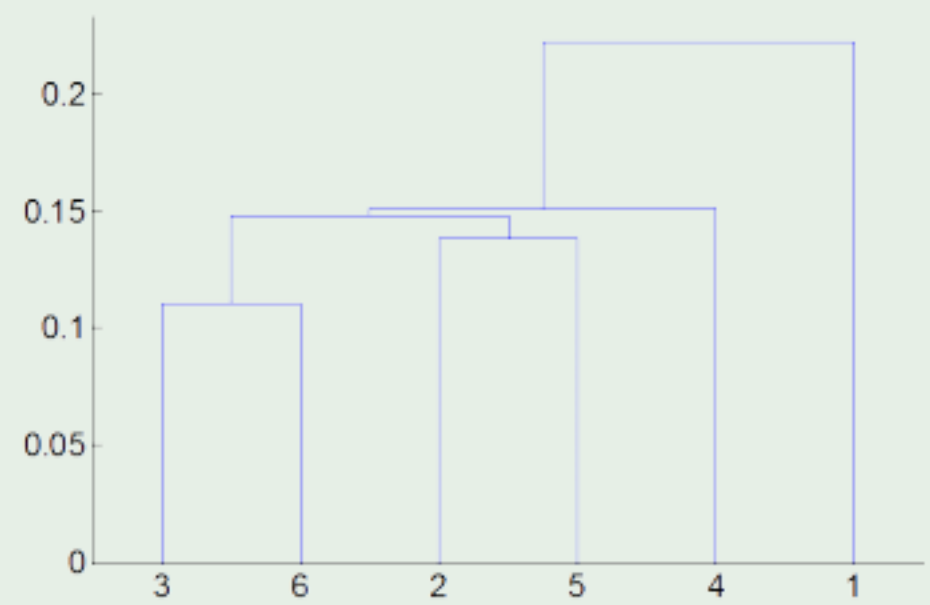
We need a notion of similarity between clusters.

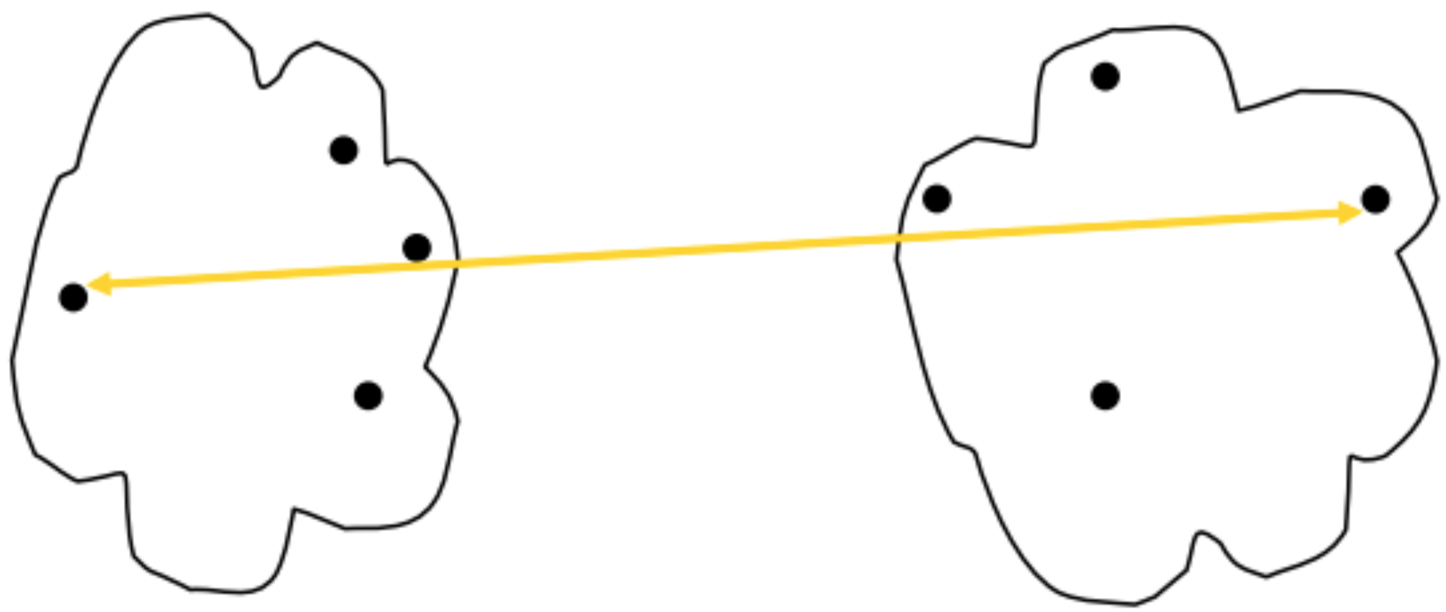


Single linkage uses the minimum distance.

Example

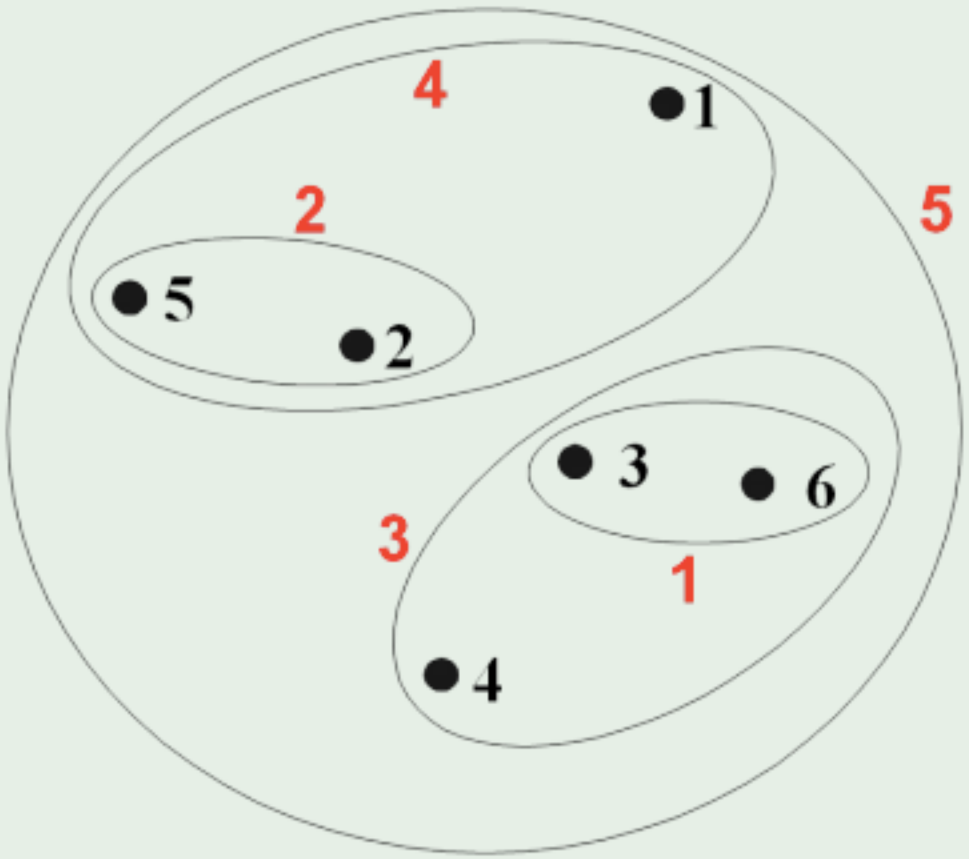


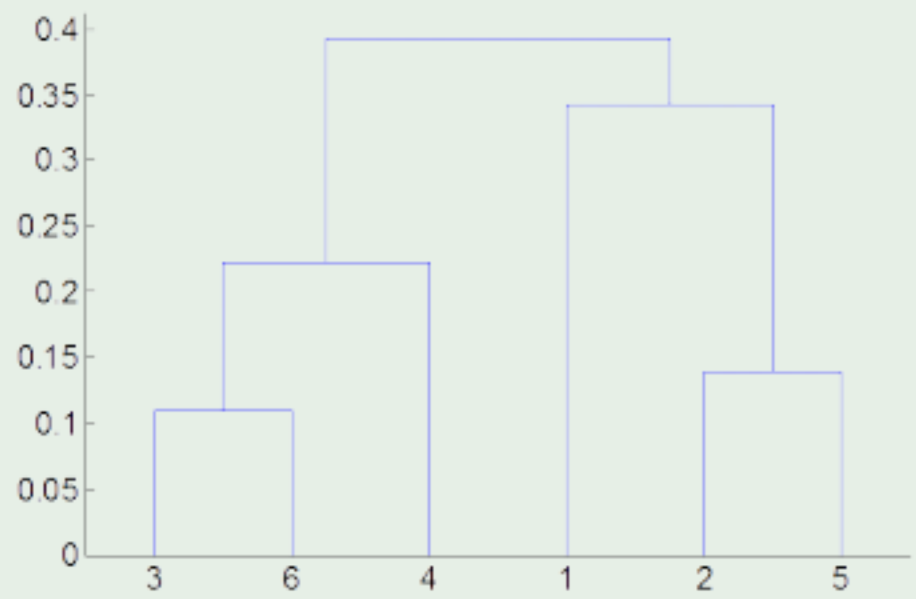


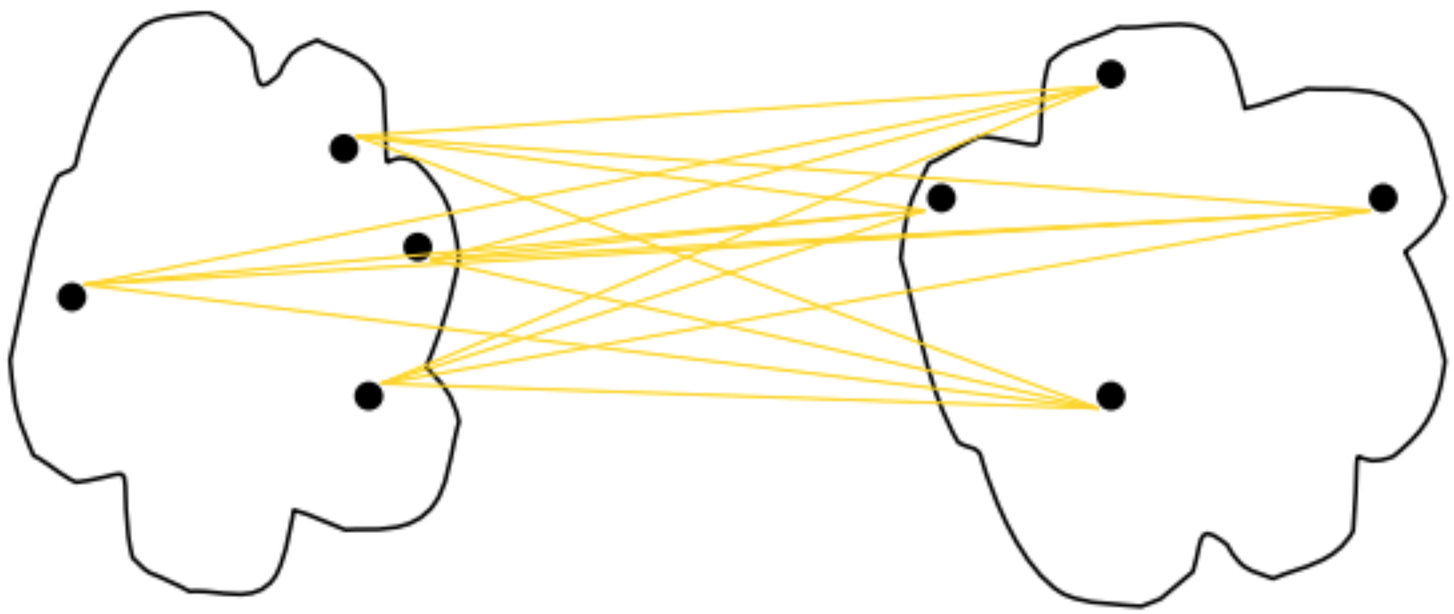


Complete linkage uses the maximum distance.

Example

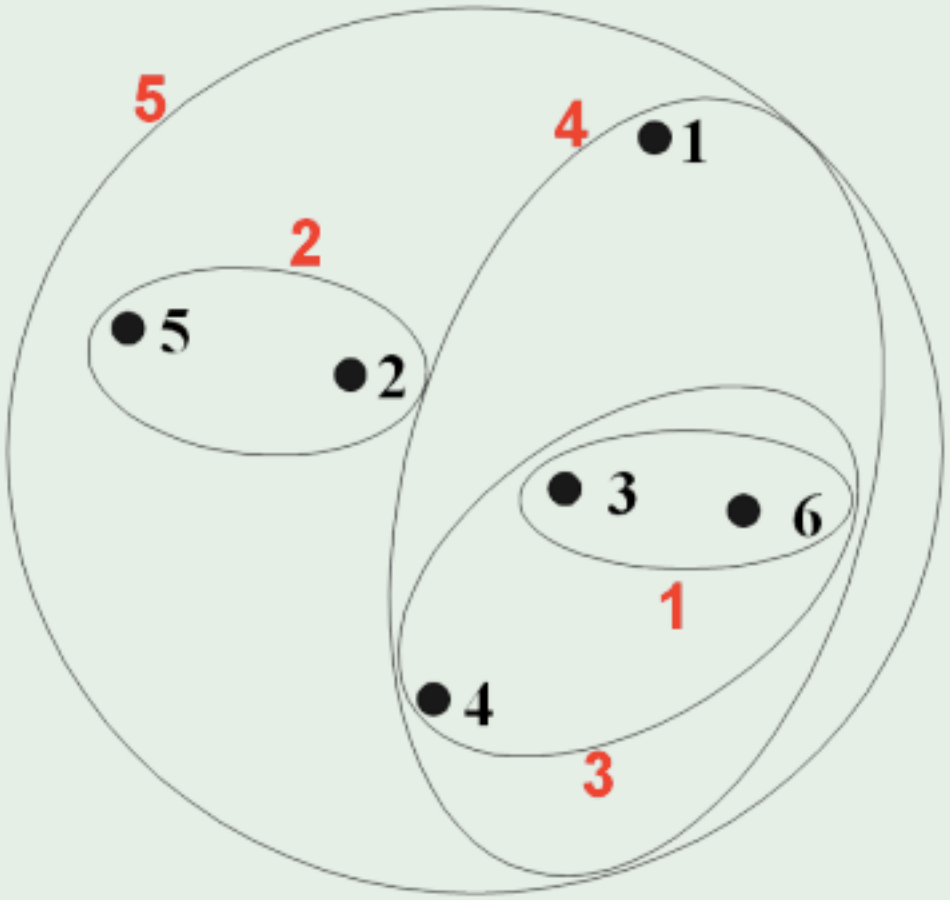


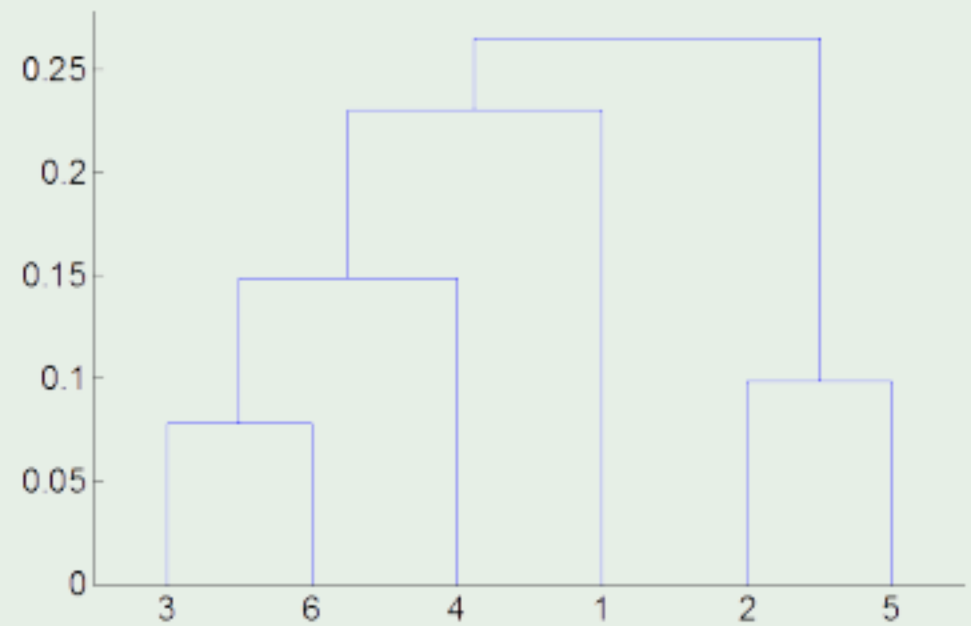


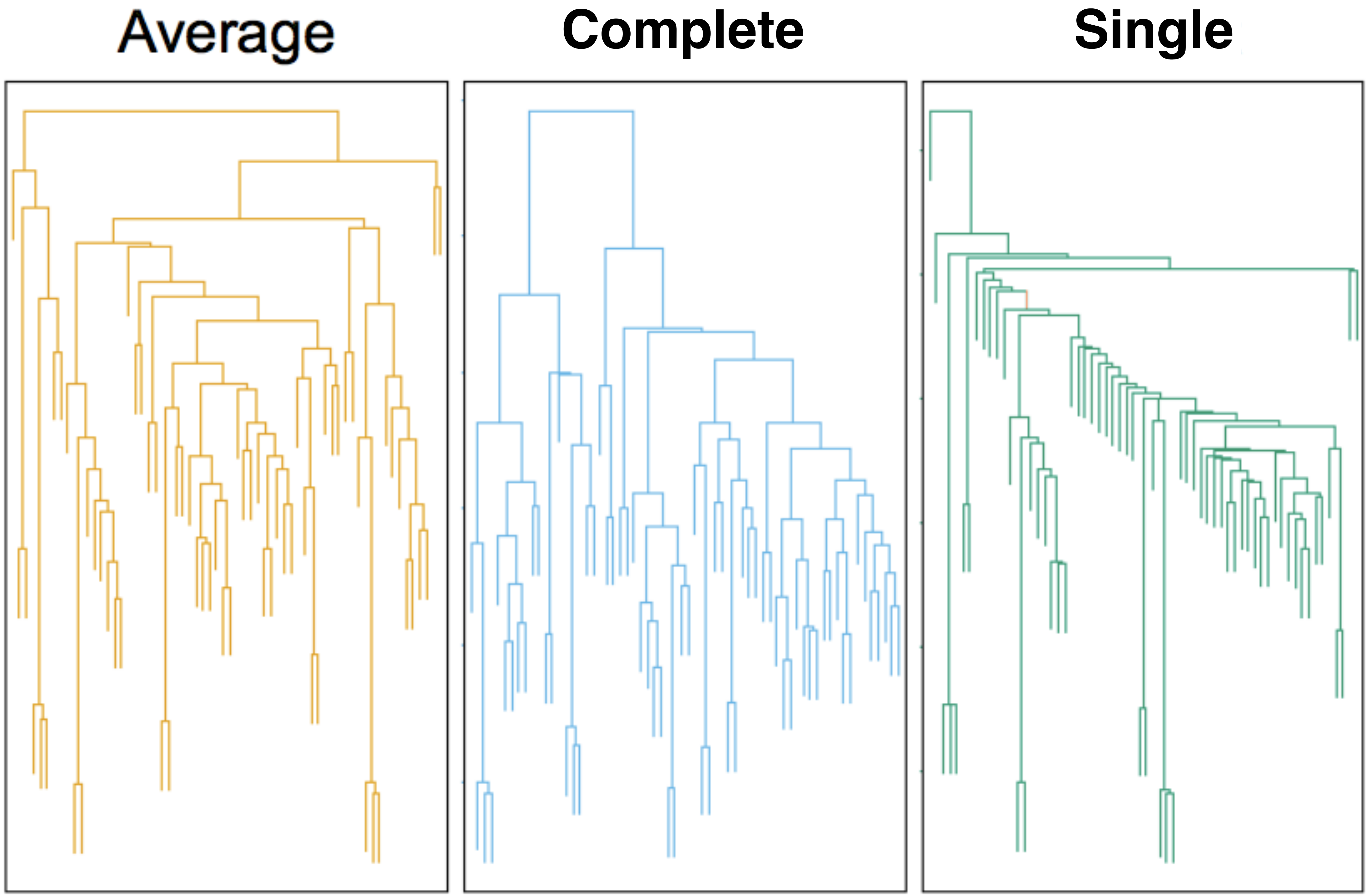


Group average linkage uses the average distance between groups.

Example







Mouse tumor data from [Hastie *et al.*]

Application to breast cancer expression data