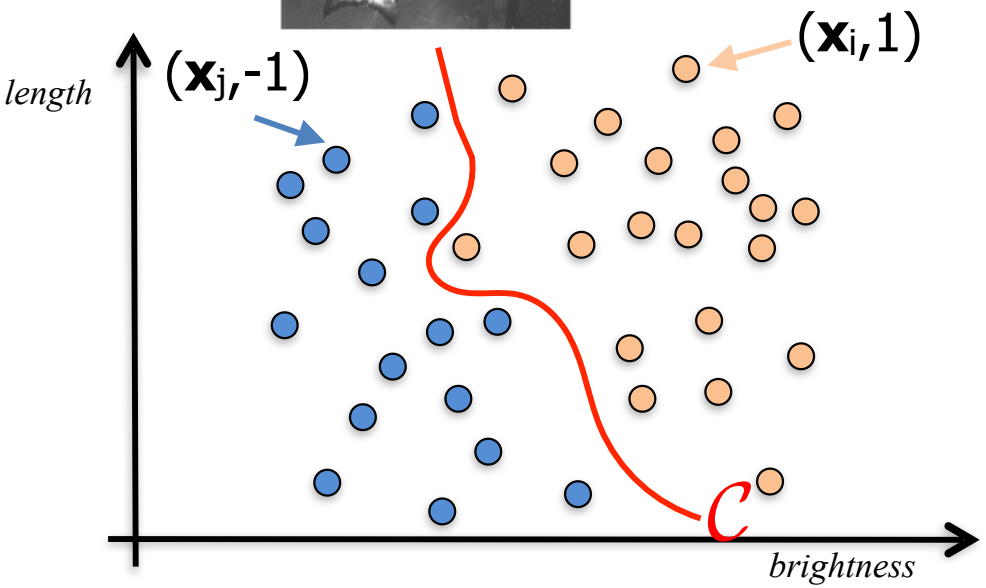
Nearest Neighbors

Pascal Fua
IC-CVLab

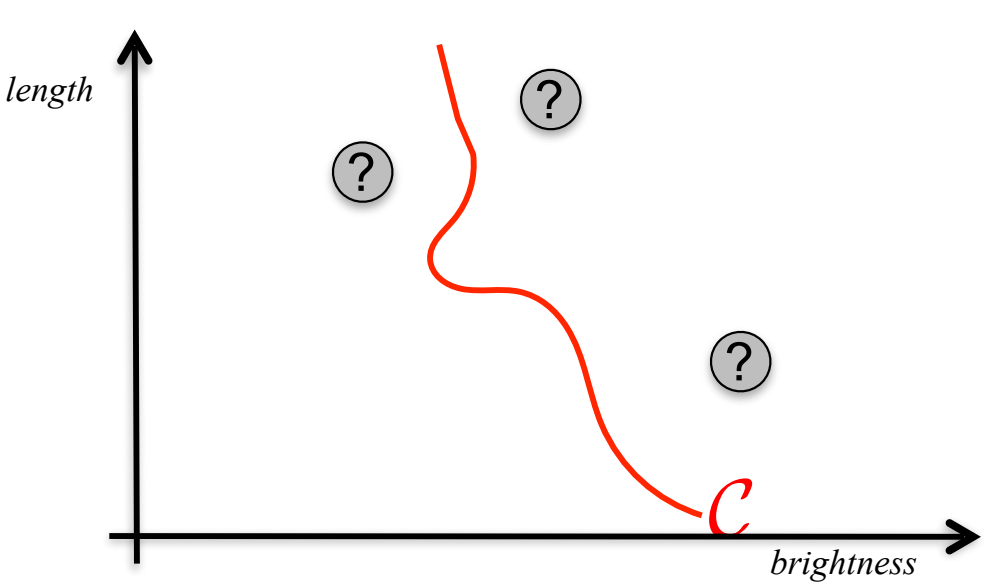
Simple Heuristic

Some algorithm

$$\begin{pmatrix} brightness \\ length \end{pmatrix}$$



Training set = {●,●}



Test set = {?,?,?, ….}

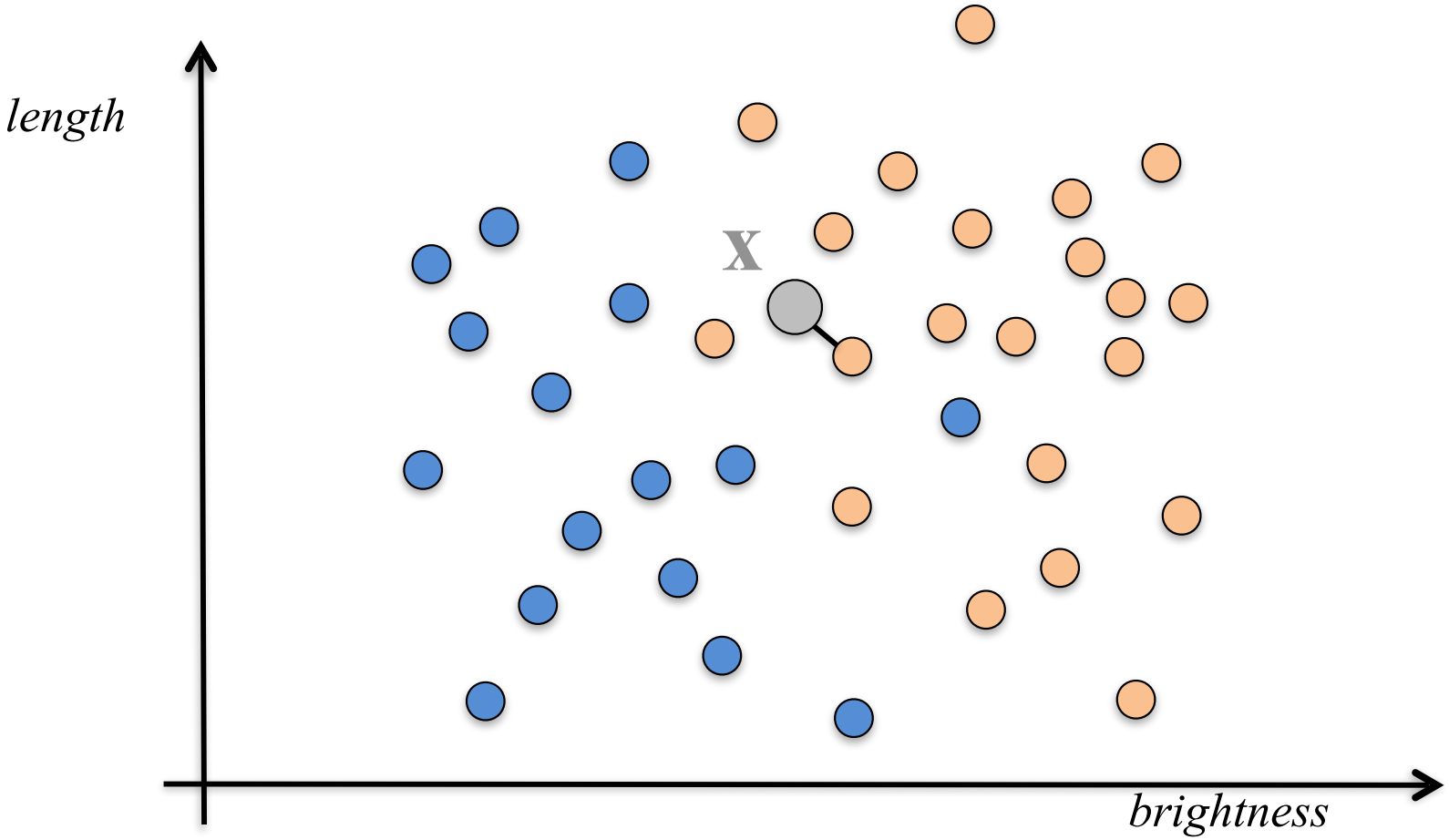
Basic intuition:

- If a ? is close to many ●, it's probably one.
- Same thing with the ○ .

Nearest-Neighbor Classifier

Simplest algorithm:

- Given a new **x** to be classified, find the nearest neighbor in the training set.
- Classify the point according to the label of this nearest neighbor.



2D Voronoi Diagram

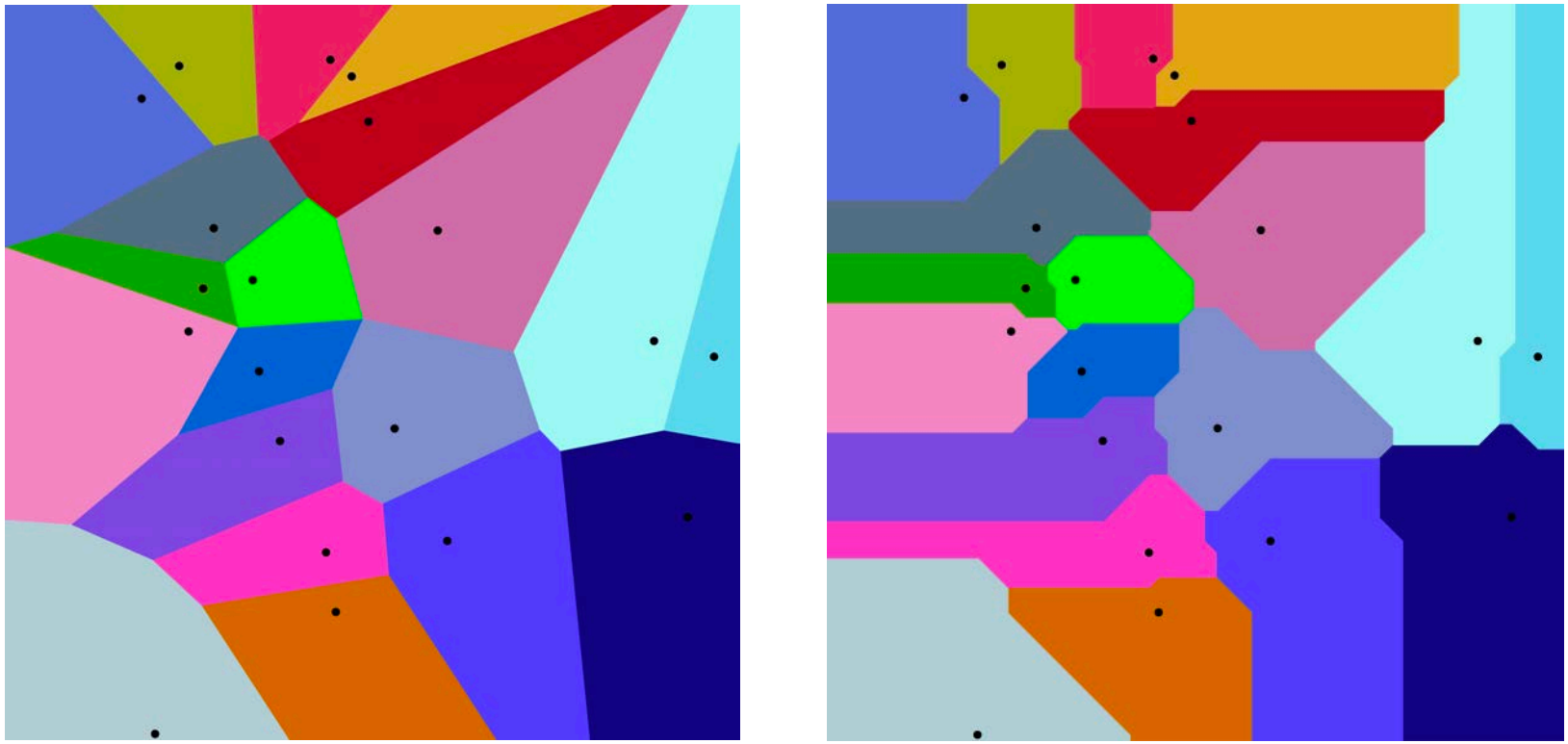
Given the set of N training samples $\{\mathbf{x}_n\}_{1 \leq n \leq N}$, we can define:

- N Voronoi cells

$$C_n = \{\mathbf{x} \in \mathbf{X} | \forall j \neq n \ , \ d(\mathbf{x}, \mathbf{x}_n) \leq d(\mathbf{x}, \mathbf{x}_j)\} \ ,$$

- and the Voronoi diagram

$$V = \{C_n\}_{1 \leq n \leq N} \ .$$



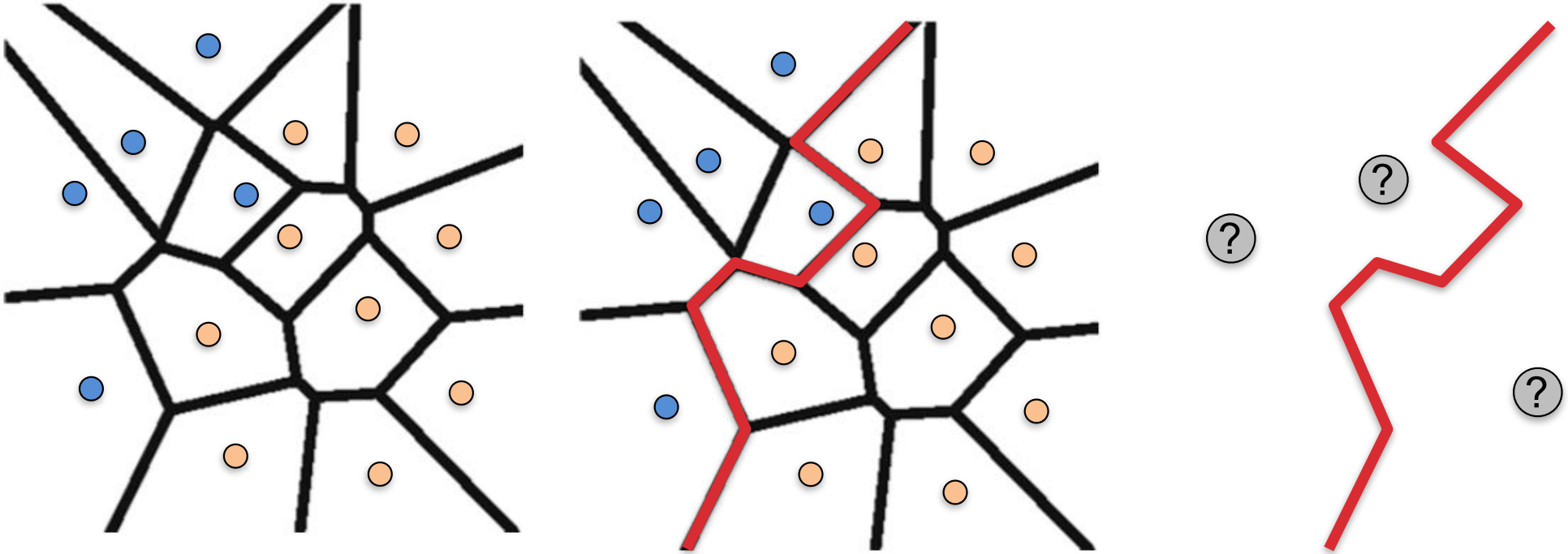
Euclidean distance

$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

Manhattan distance

$|x_2 - x_1| + |y_2 - y_1|$

Nearest-Neighbor Classifier



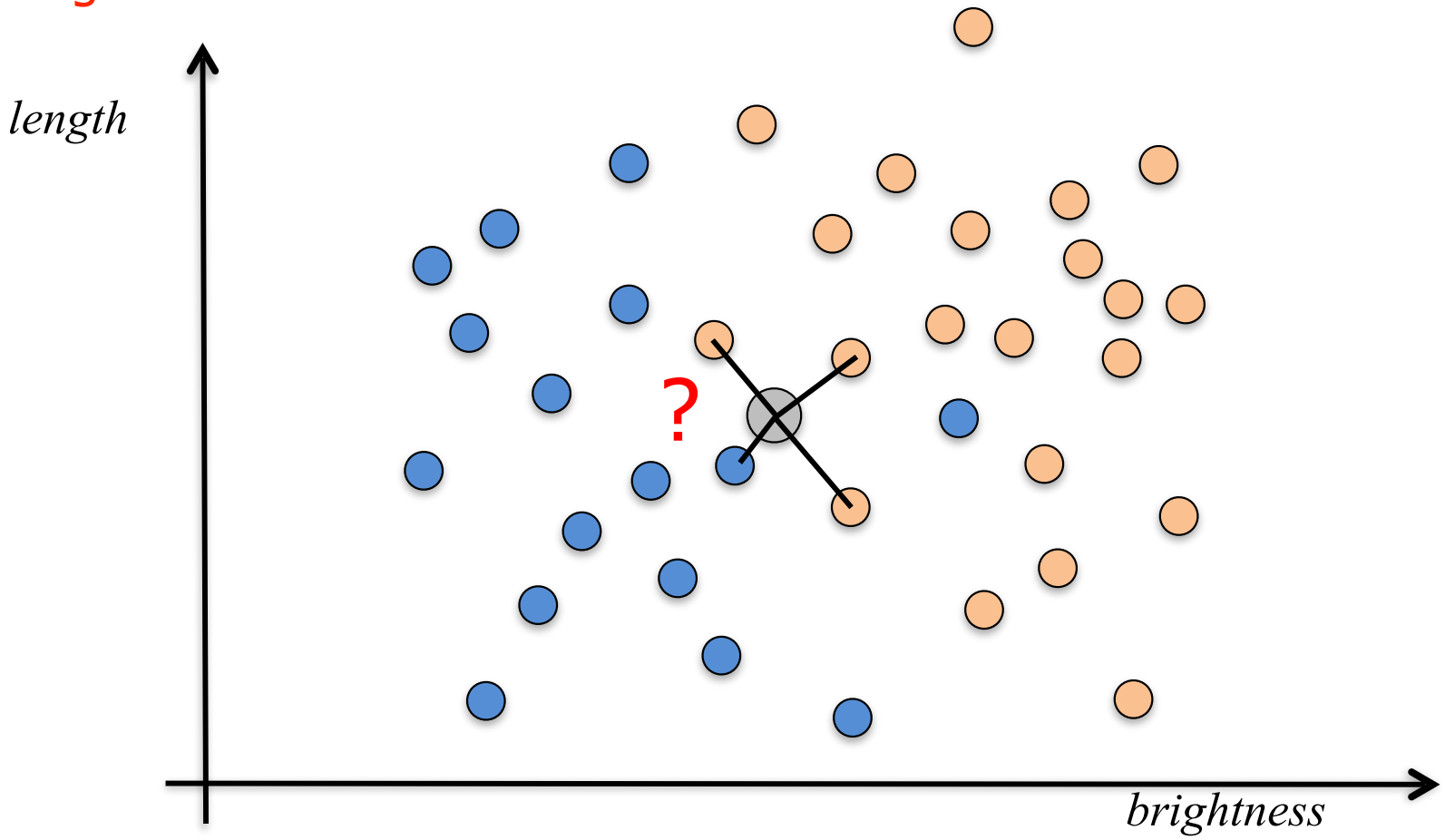
Voronoi diagram Decision boundary Classification

—> The decision boundary is formed by selected edges of the Voronoi Diagram.

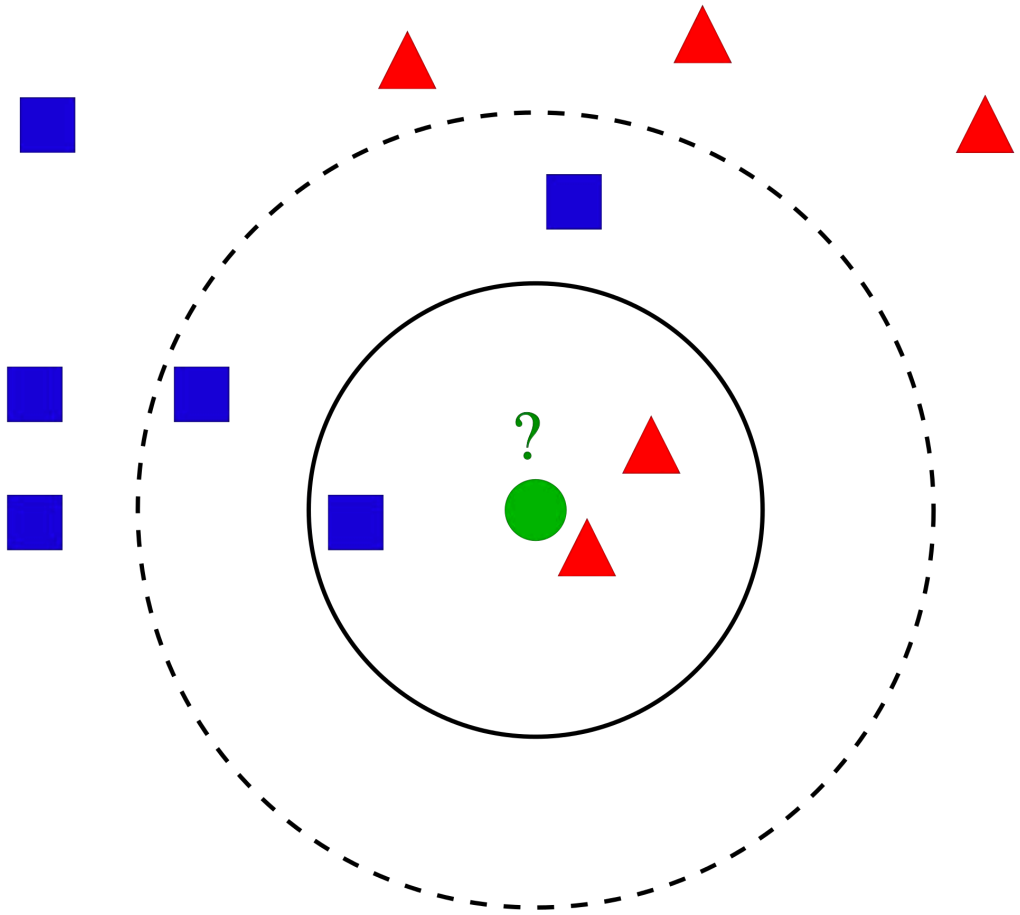
K-Nearest-Neighbor Classifier

Improved algorithm:

- Given a new **x** to be classified, find its k nearest neighbors in the training set.
- Classify the point according to the majority of labels of its nearest neighbors.



Red or Blue?



k=1: red
k=3: red
k=5: blue

Reminder: Supervised Learning

Train using an annotated training set:

{( , face), ( , face), ( , face), … ,

( , non-face), ( , non-face), ( , non-face),

… }

Run on images that do **not** belong to the training set:

 —> face or non-face?

Key Assumption

- The training set and the test set are drawn from the same statistical distribution.
- Otherwise, there is no reason for a decision boundary learned on the training set to be useful on the test set.
- For example, in the face detection example, the training set must be representative of all faces the system is likely to encounter.

Working in N-Dimensional Spaces

- In our fish example, the samples $\mathbf{x}_i$ are 2-D vectors.
- In the case of faces, the samples are whole images. For an WxH images, $\mathbf{x}_i$ is of size WH and is therefore high-dimensional.

When the samples are of dimension D, we write:

$$\mathbf{x} = [x_1, \ldots, x_D]^T ,$$

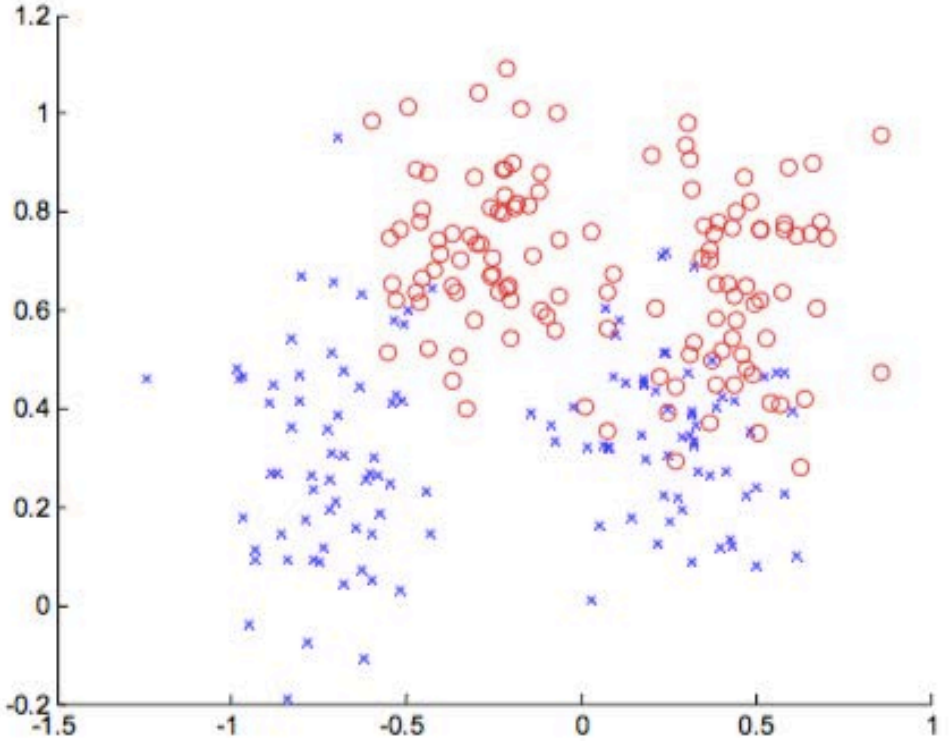
$$\mathbf{x}' = [x'_1, \ldots, x'_D]^T ,$$

$$d_2(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_{d=1}^{D} (x_d - x'_d)^2} , \qquad (L_2 \text{ or Euclidean Distance})$$

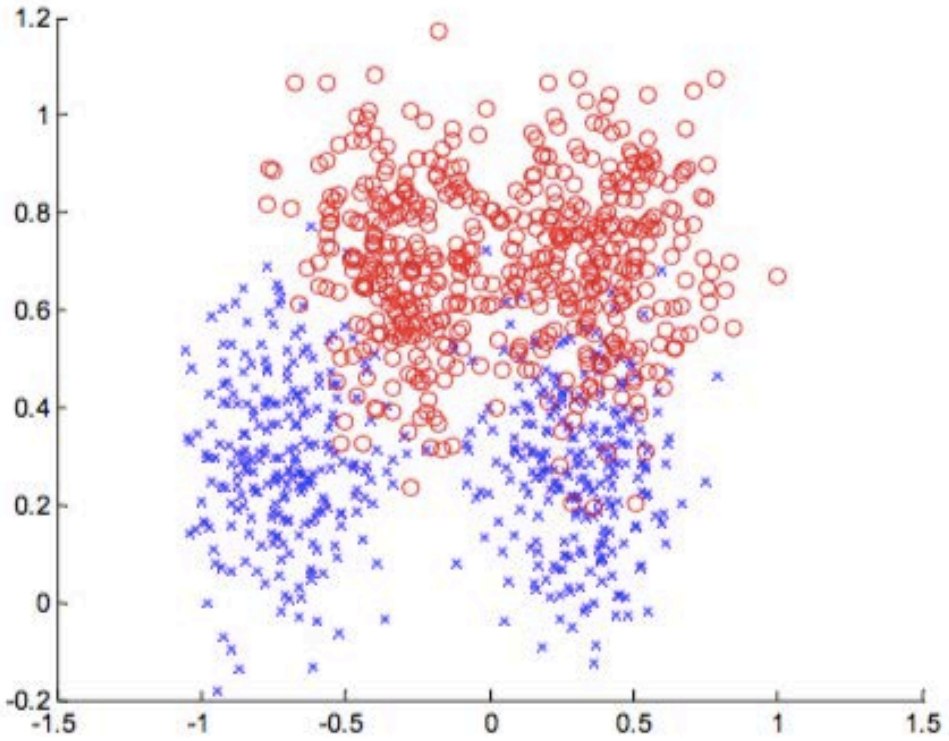
$$d_1(\mathbf{x}, \mathbf{x}') = \sum_{d=1}^{D} |x_d - x'_d| . \qquad (L_1 \text{ or Manhattan Distance})$$

—> The formulation remains essentially unchanged

Training and Test Sets



Training set



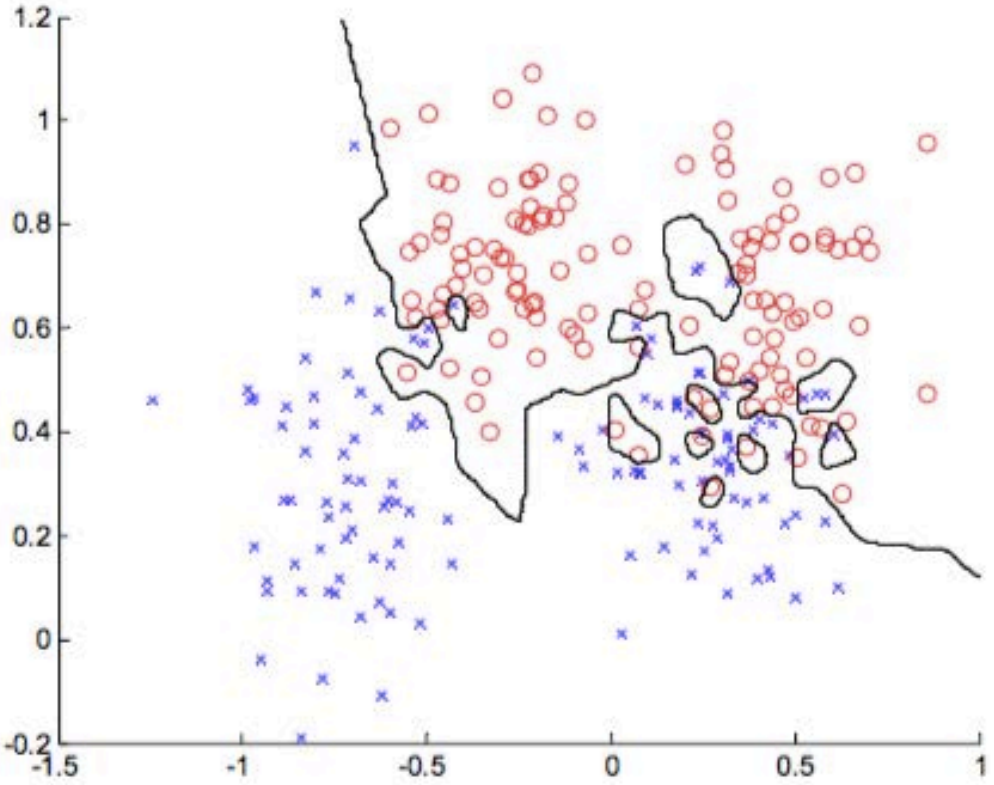
Test set

Two classes shown as different colors.

- Use the training set to learn a classifier.
- Use the test set to gauge its performance.

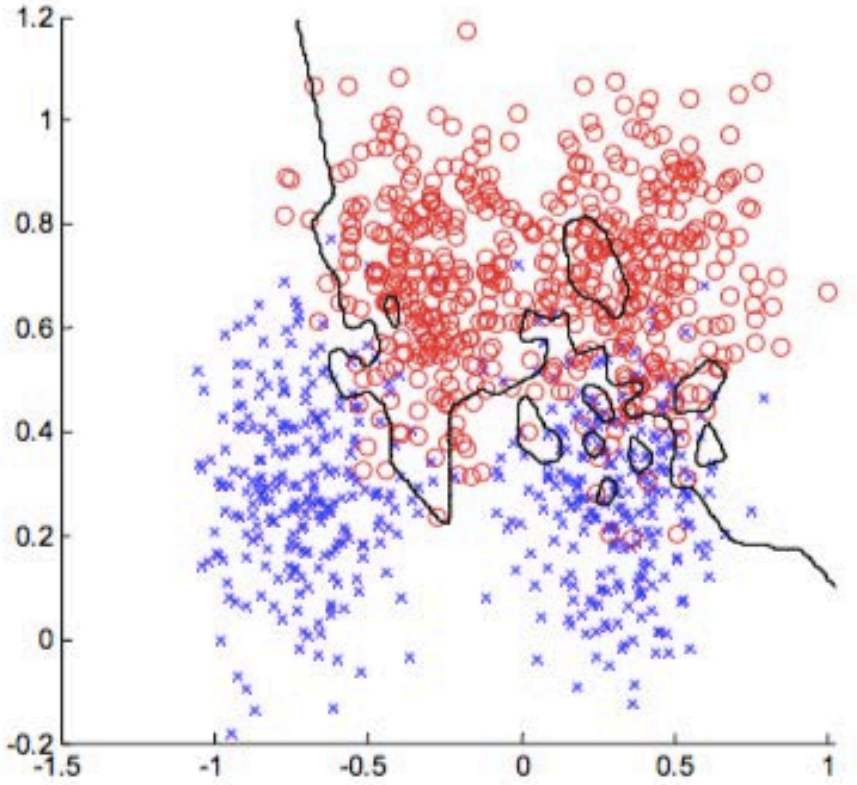
Classification Error, k = 1

Decision boundary



Training set

error = 0.0



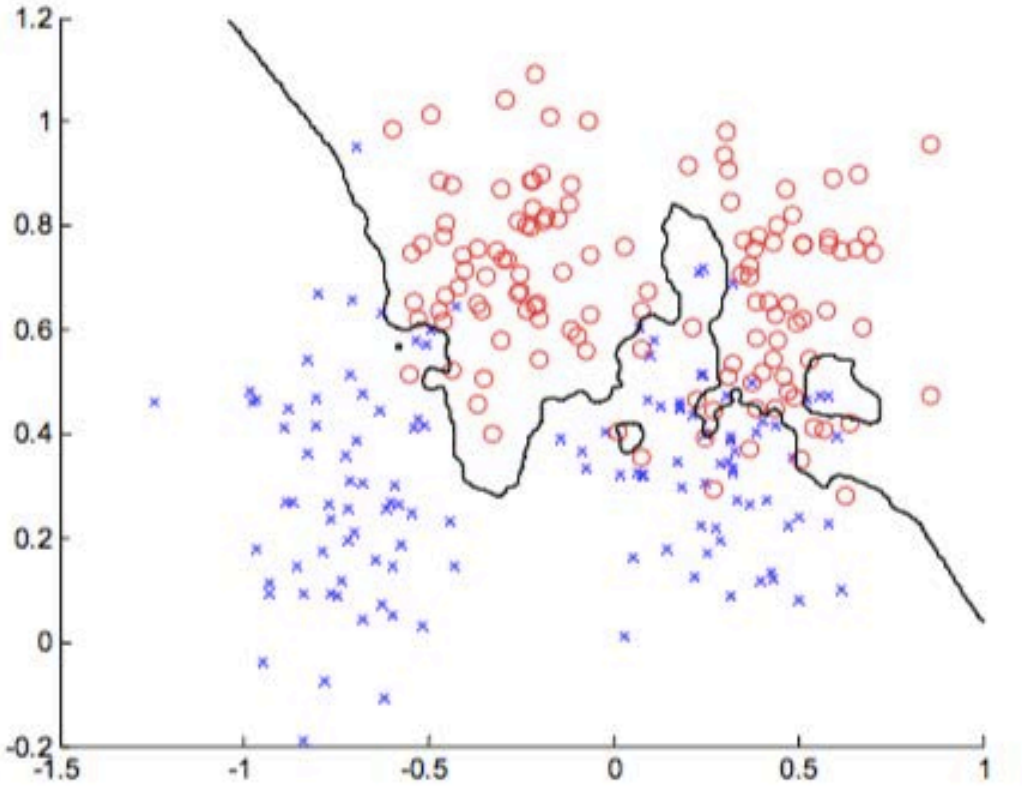
Test set

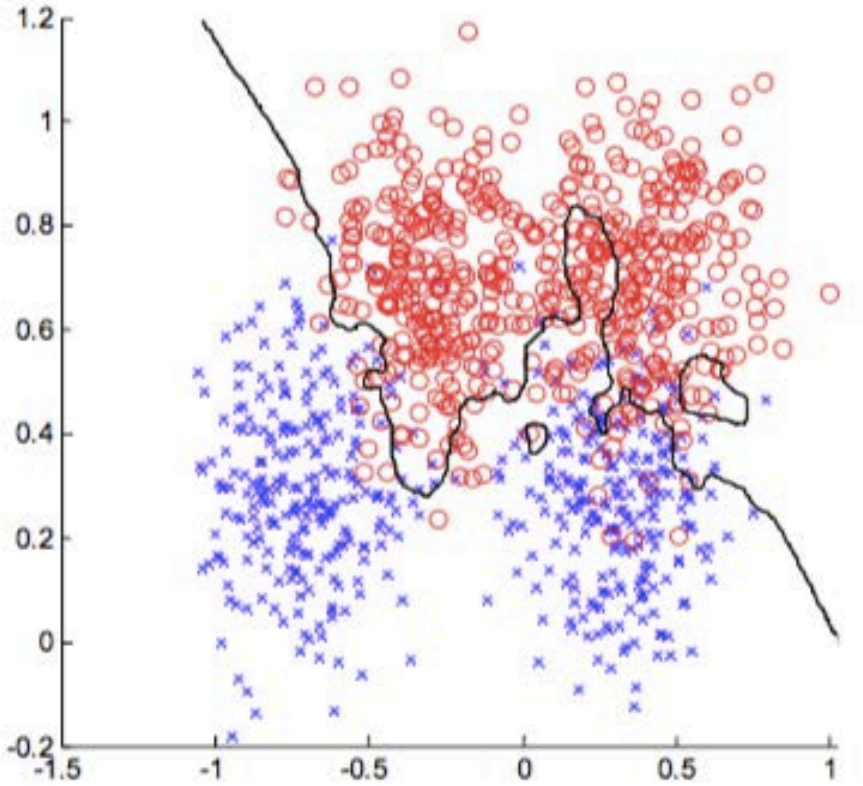
error = 0.15

This is known as overfitting.

Classification Error, k = 3

Smoother decision boundary





Training set

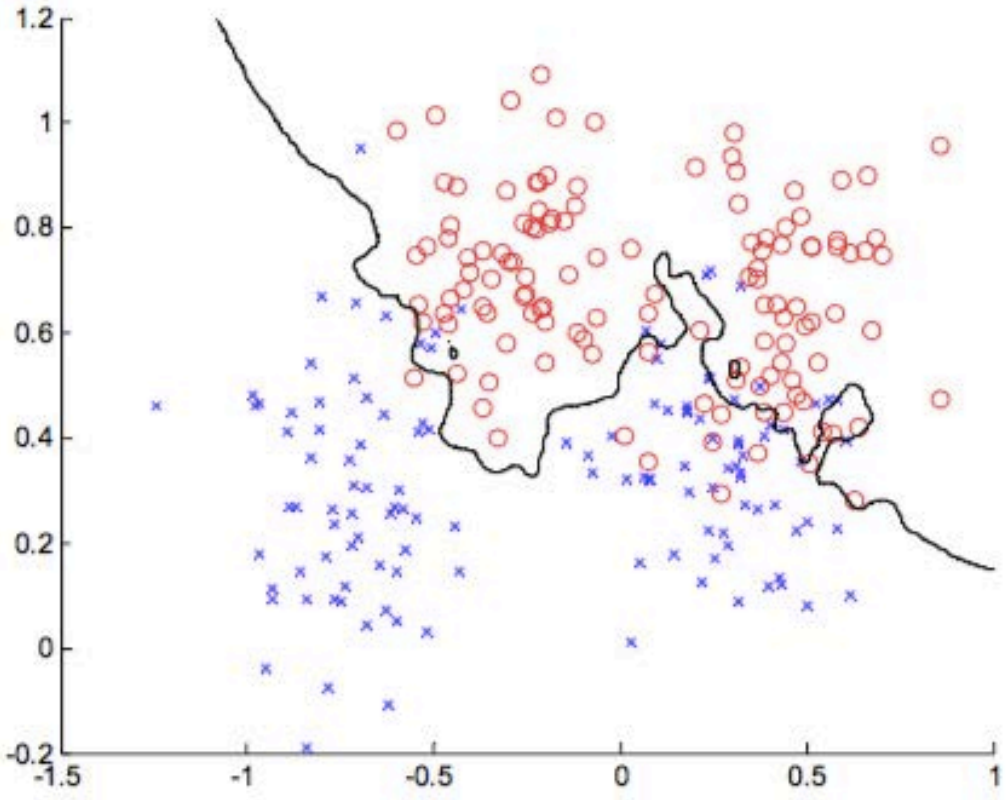
error = 0.0760

Test set

error = 0.1340

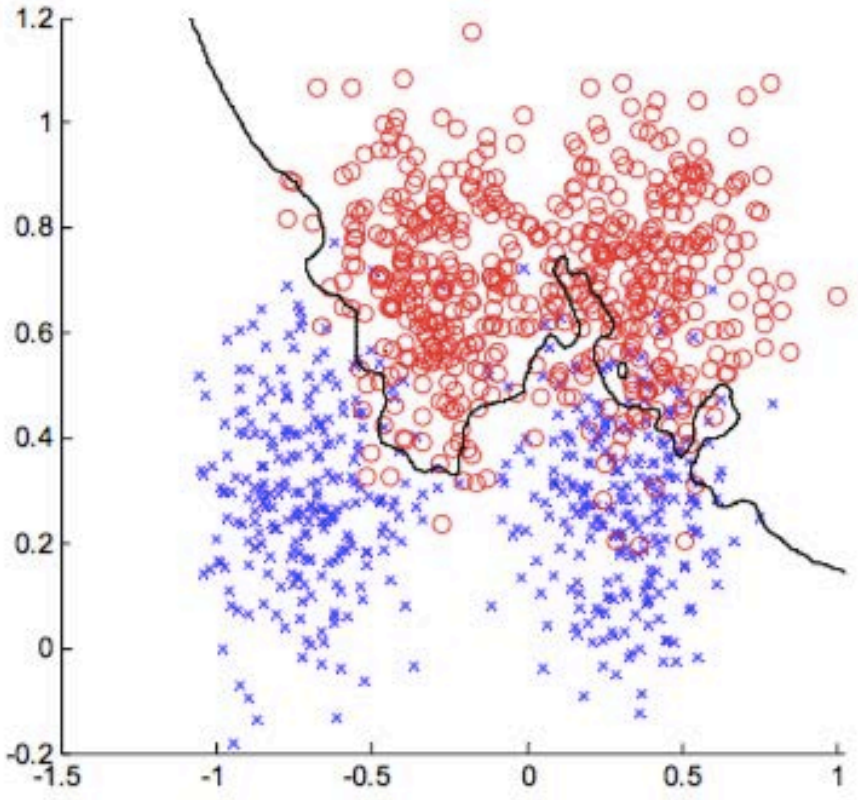
The boundary becomes smoother. The training error increases, but the testing error decreases.

Classification Error, k = 7



Training set

error = 0.1320

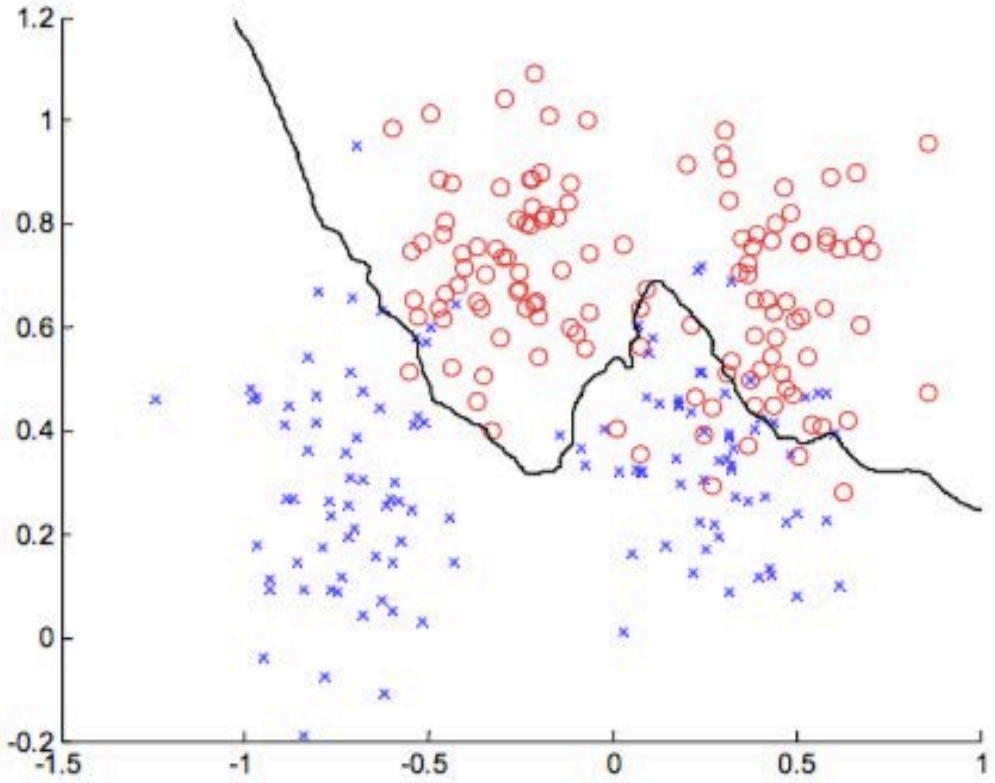


Test set

error = 0.1110

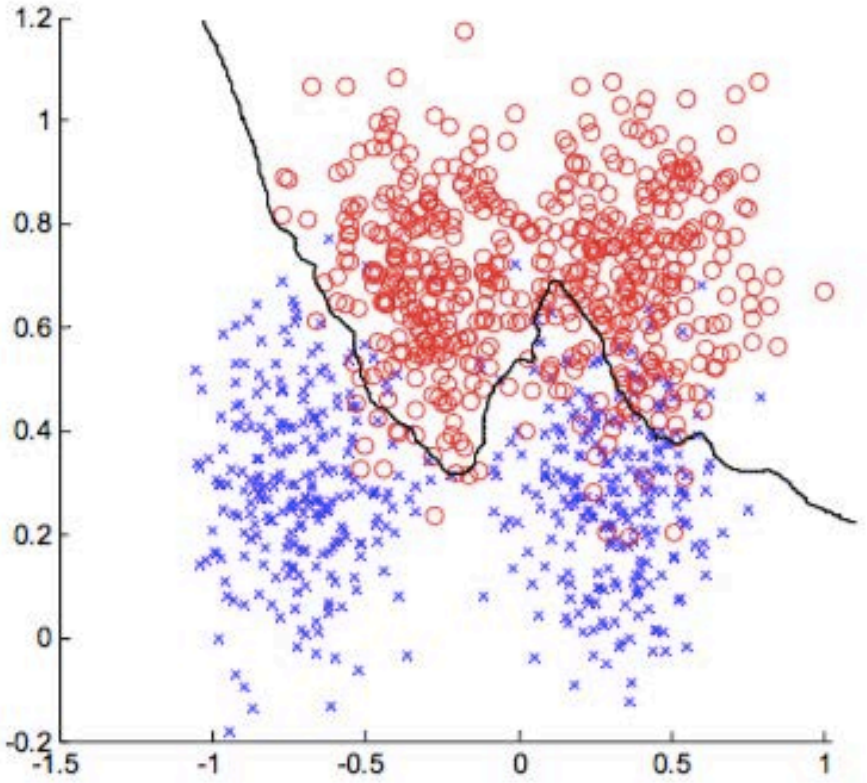
The boundary becomes smoother. The training error increases, but the testing error decreases.

Classification Error, k = 21



Training set

error = 0.1120

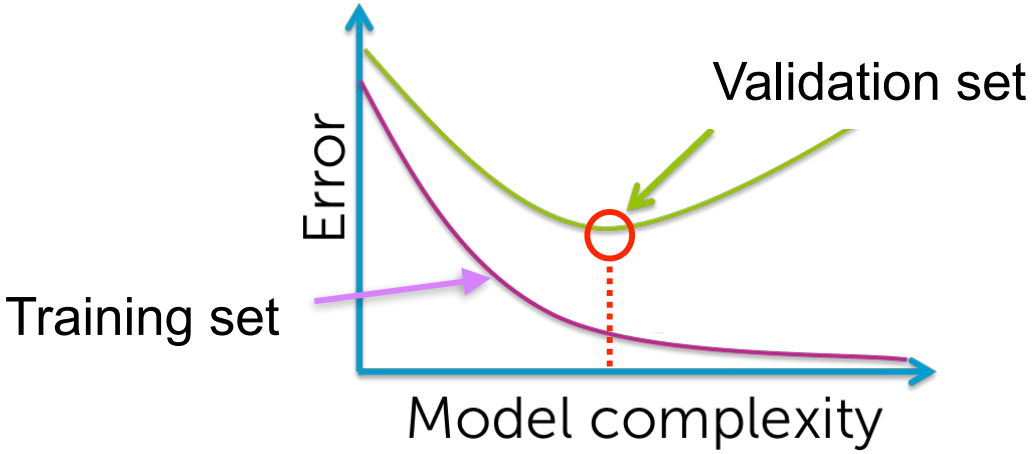


Test set

error = 0.0920

The testing error decreases again but would start increasing if we chose k even larger.

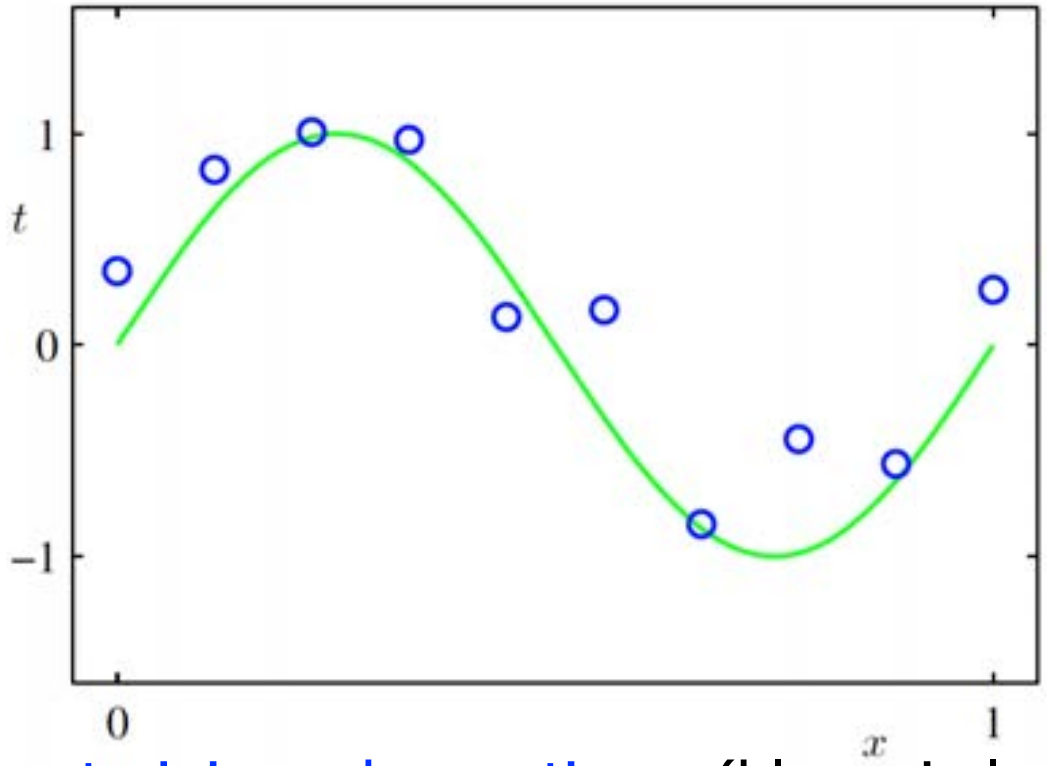
Choosing k to Avoid Overfitting



- Split the training set into a real training set and a **validation set**.
- Choose k that minimizes the classification error on the validation set.

This is known as **cross-validation** and is used in most supervised algorithms.

Underfitting and Overfitting in a Different Context

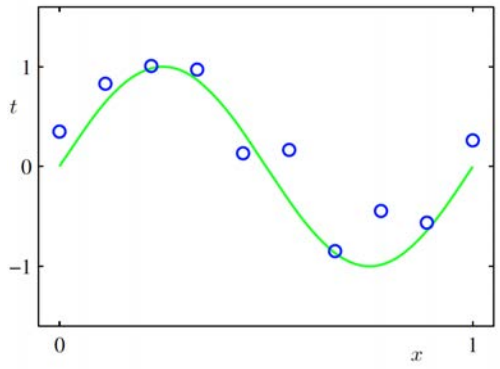


- We have noisy training observations (blue circles, 1D input, 1D output) coming from the true green curve.
- We seek to find a polynomial function that approximates the true curve using these observations.

This is known as polynomial curve fitting.

Optional

Polynomial Curve Fitting (1)



Such a polynomial function of degree M can be written as

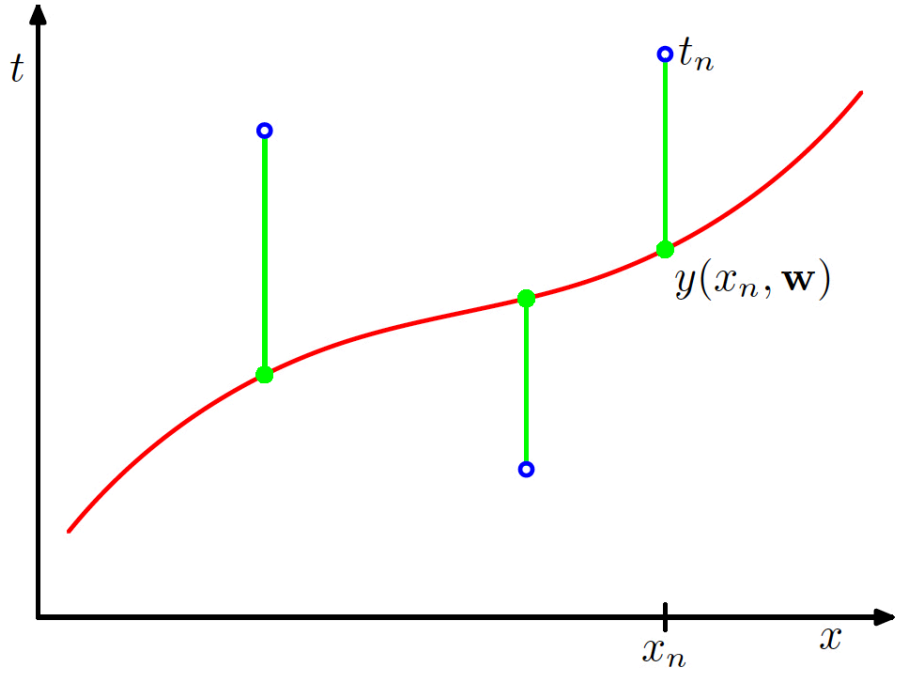
$$y = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

where the w_j are the coefficients of the different terms.

- For $M = 0$, we have the constant function $y = w_0$
- For $M = 1$, we have the line $y = w_0 + w_1 x$
- For $M = 2$, we have the quadratic function $y = w_0 + w_1 x + w_2 x^2$

Optional

Polynomial Curve Fitting (2)

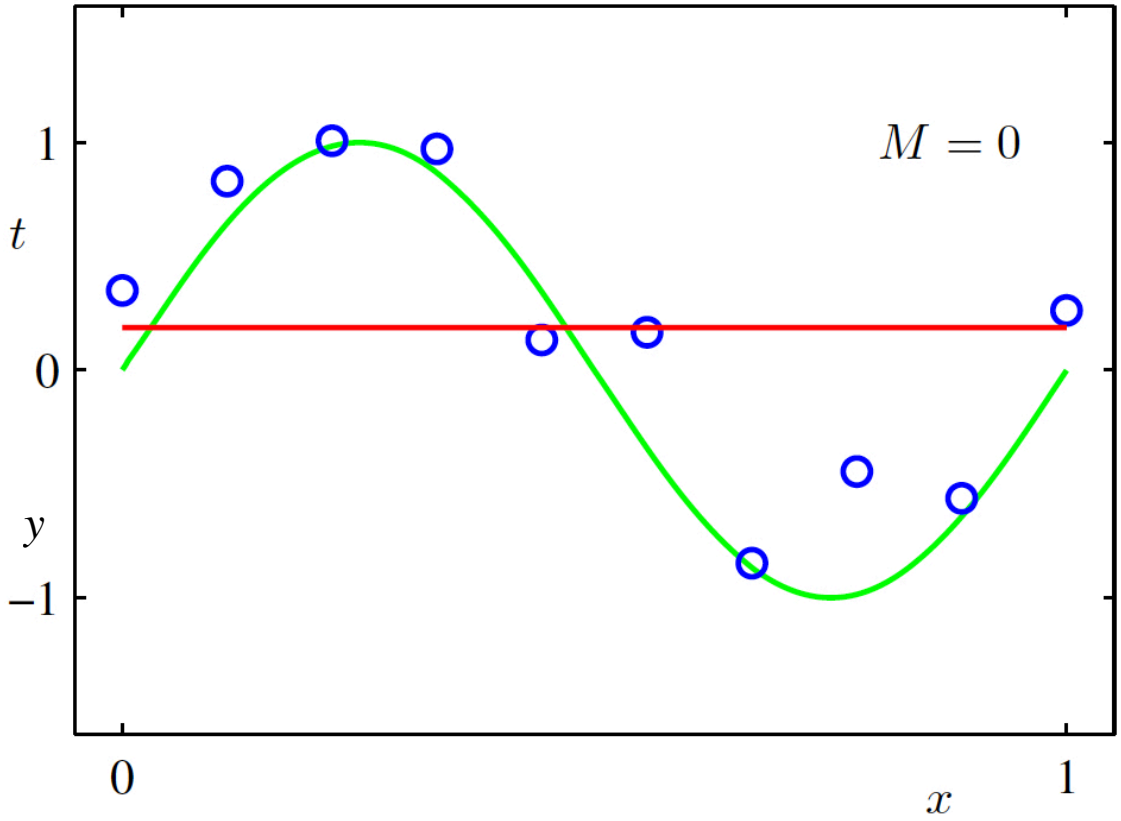


For each value of M, find the coefficients $\mathbf{w} = [w_0, \ldots, w_M]$ that make the curve closest to the observations in terms of the sum of square differences

$$\sum_n (y(x_n, \mathbf{w}) - t_n)^2$$

Optional

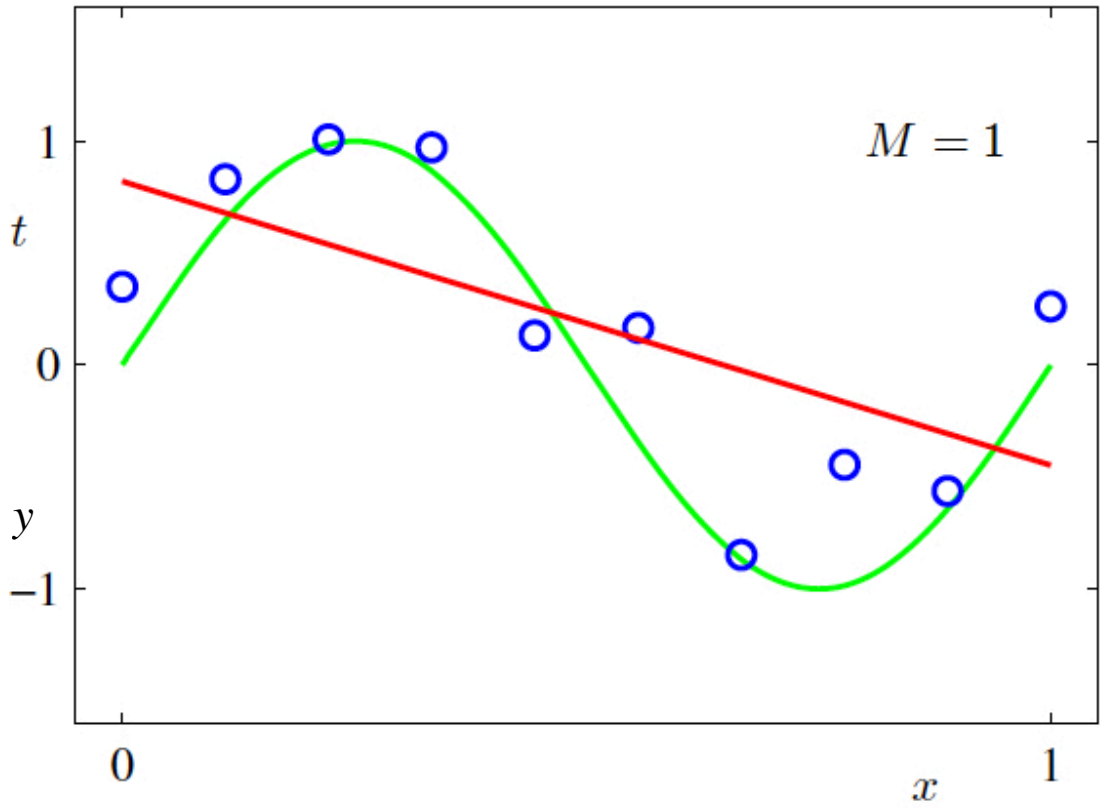
M=0 -> Underfitting



Approximates the observations poorly and is far from the unknown true curve

Optional

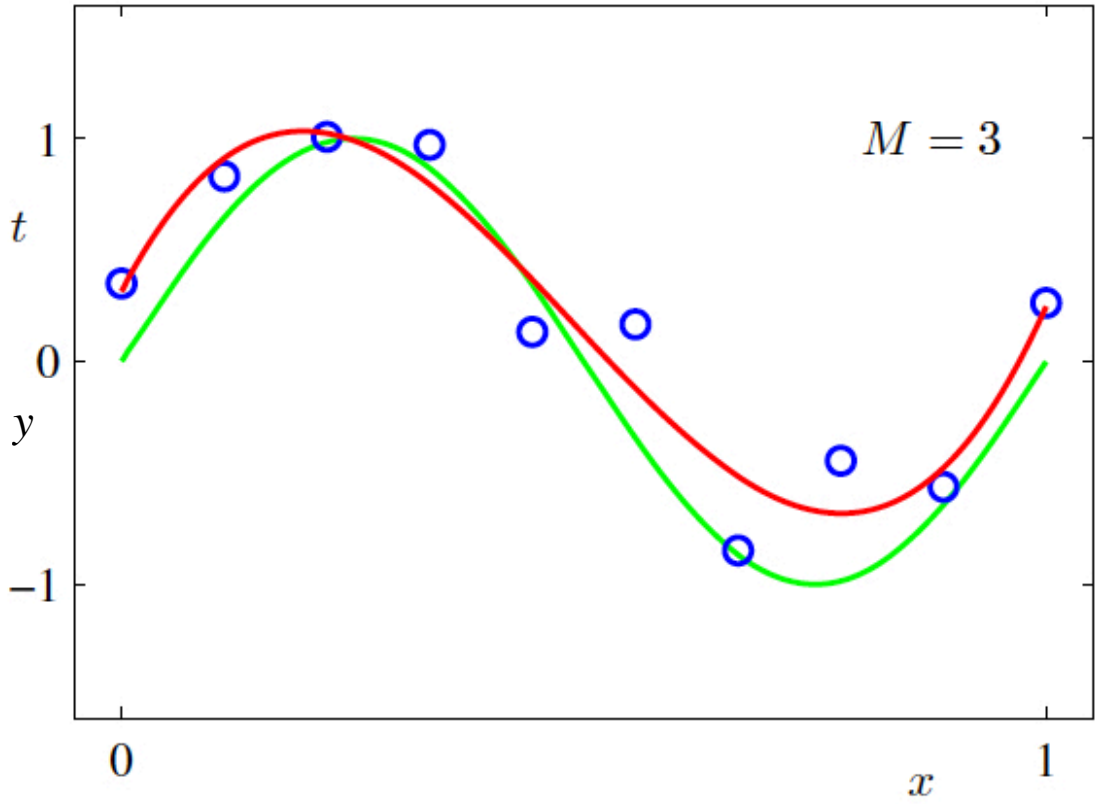
M=1 -> Underfitting



Still approximates the observations poorly and is far from the unknown true curve.

Optional

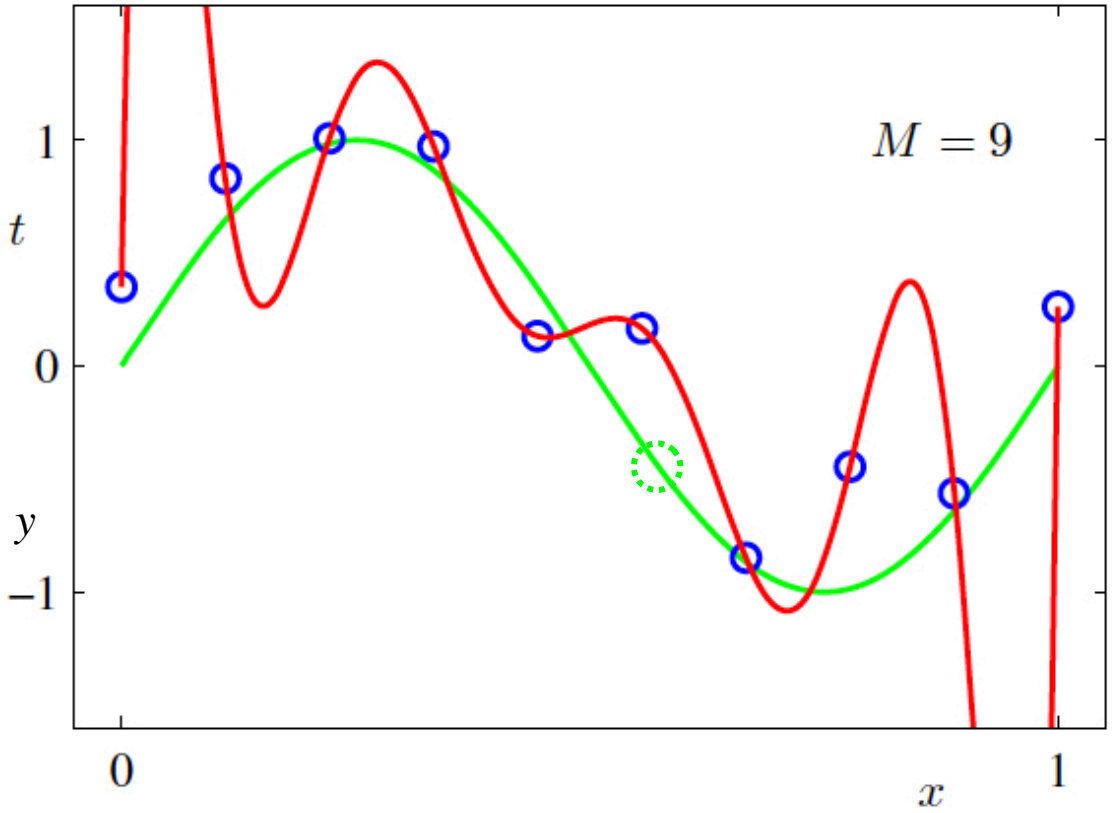
M=3 -> Good Fit



Approximates the observations well and is close to the unknown true curve.

Optional

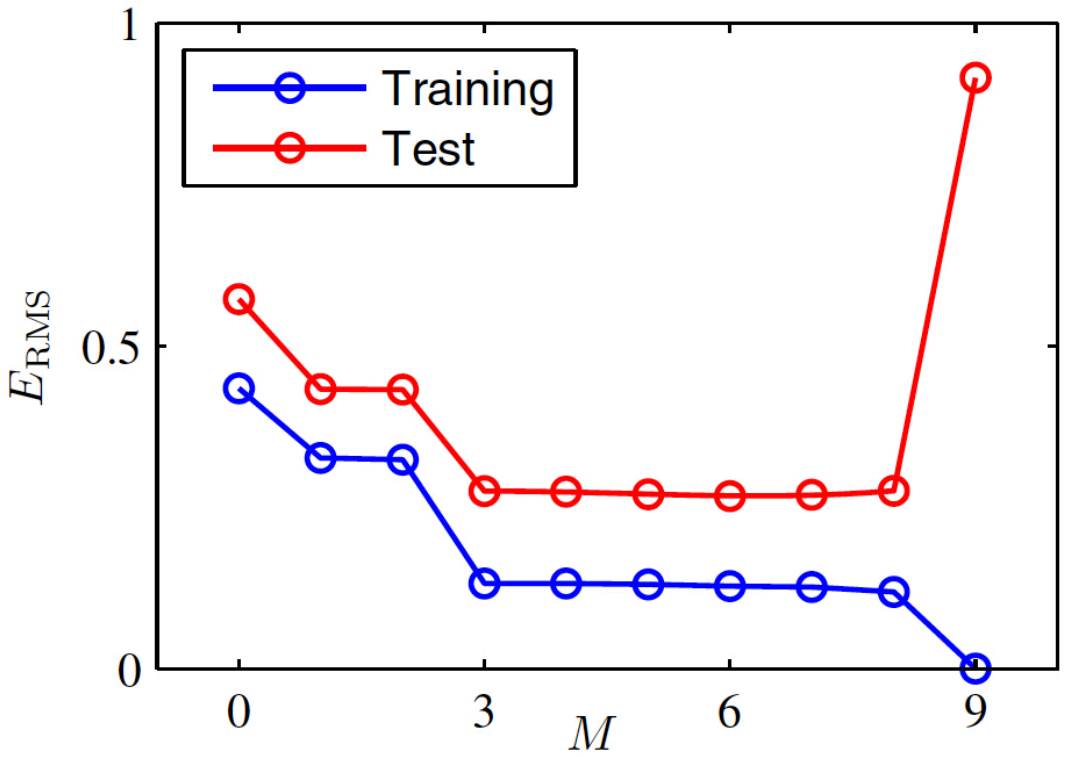
M=9 -> Overfitting



- Approximates the observations perfectly but is far from the unknown true curve.
- Validation data is required to detect that!

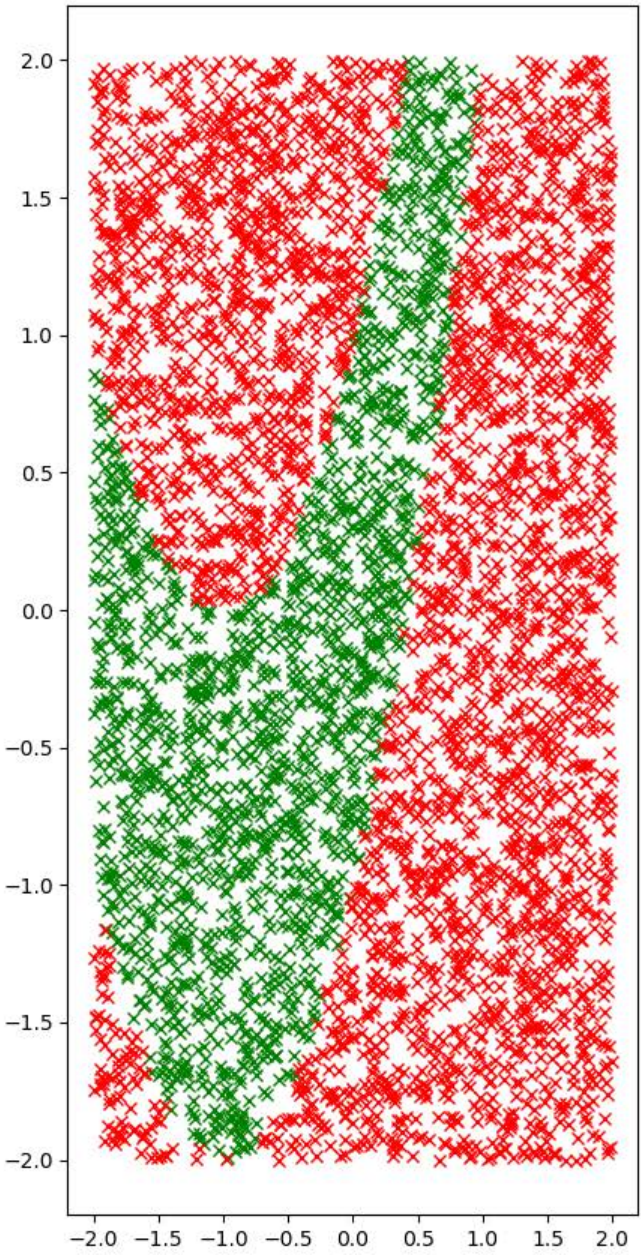
Optional

Moral of the Story



- While the training curve always decreases as the degree increases, the test curve eventually increases again.
- Virtually all ML algorithms behave in this way.
- We will revisit all these issues when we talk about Non Linear ML later in the class.

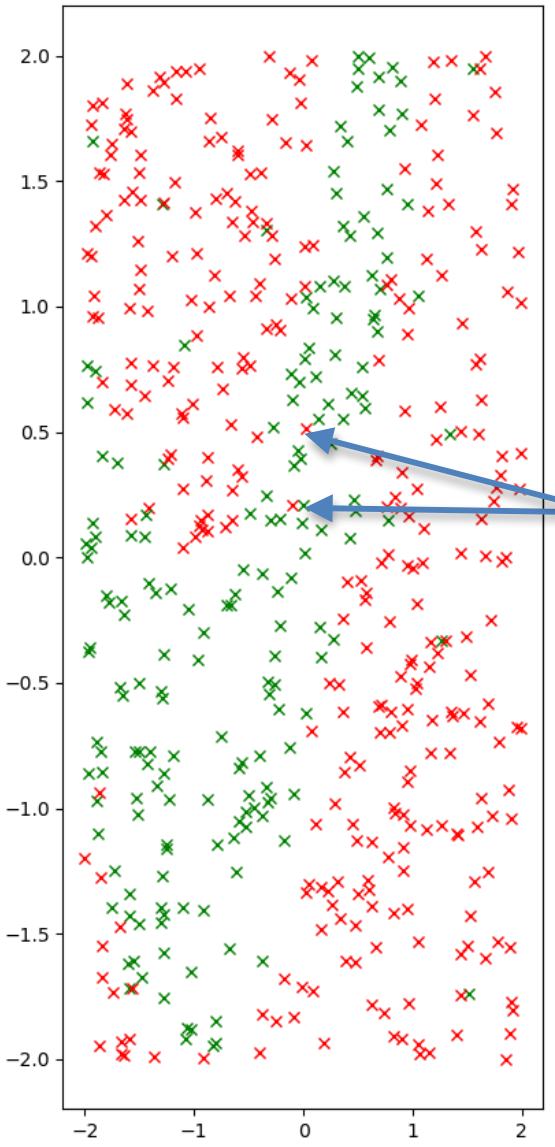
Rosenbrock Function



$$r(x,y) = 100 * (y - x^2)^2 + (1 - x)^2$$

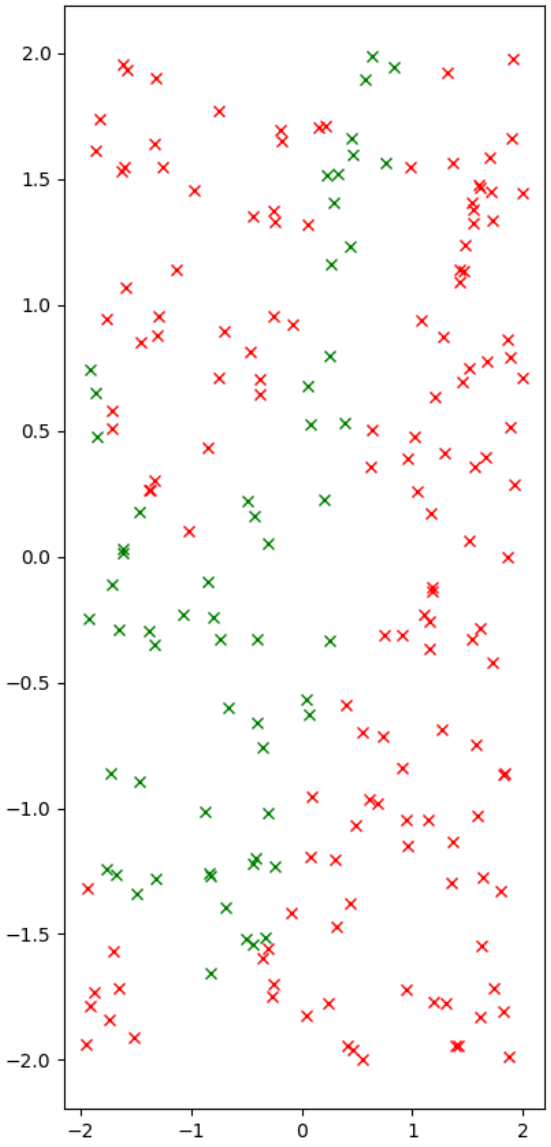
$$f(x,y) = \begin{cases} \text{-1} & \text{if } r(x,y) < \text{T} \\ 1 & \text{otherwise} \end{cases}$$

Noisy Training Data



Mislabeled samples

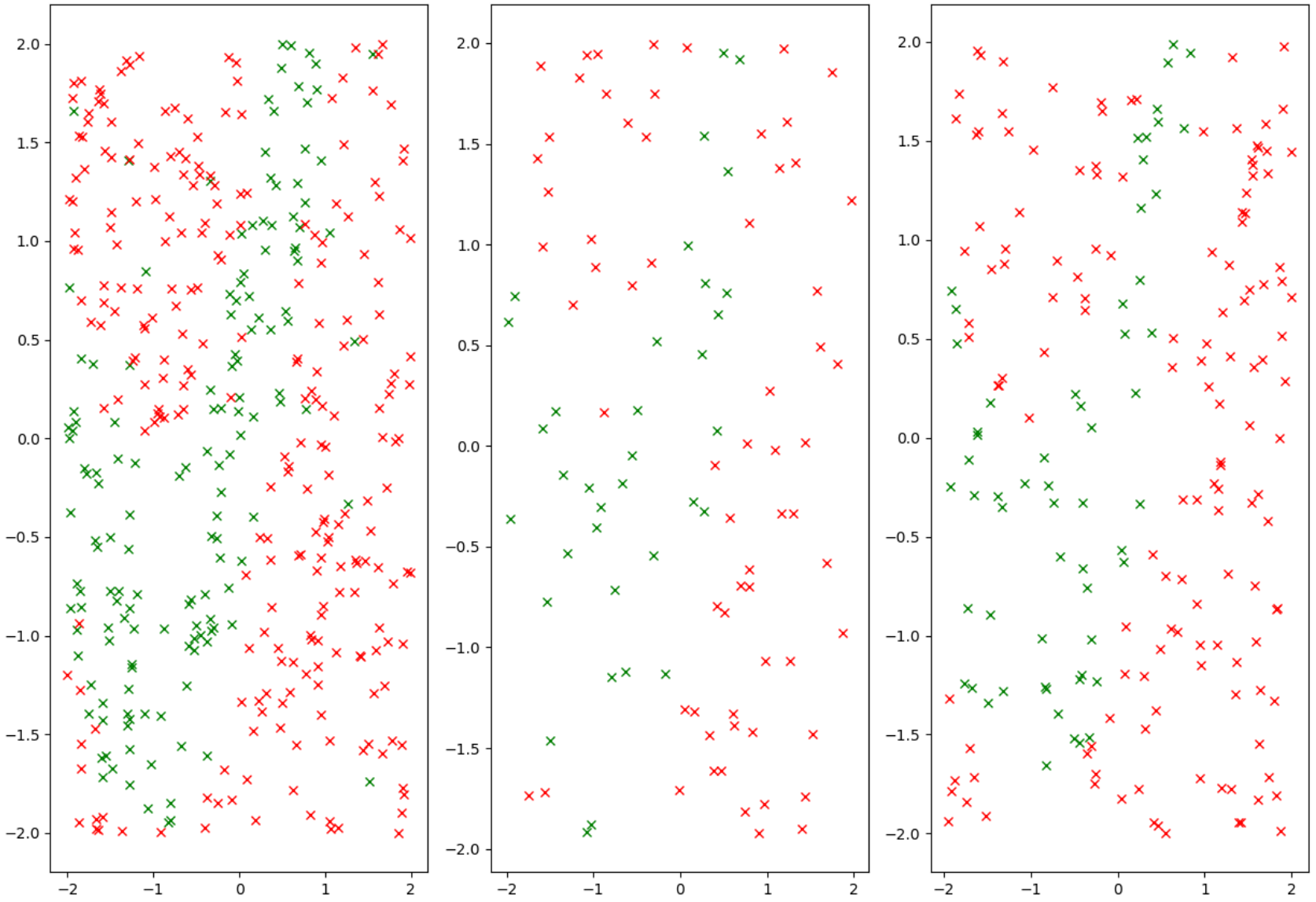
Training set



Test set

The test data should not be used during training!

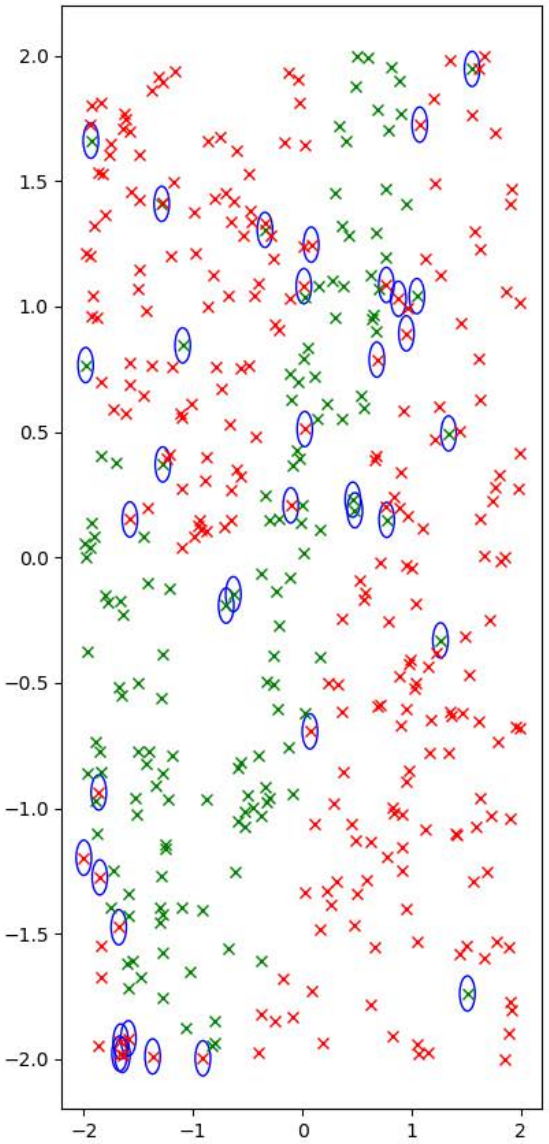
Validation Set



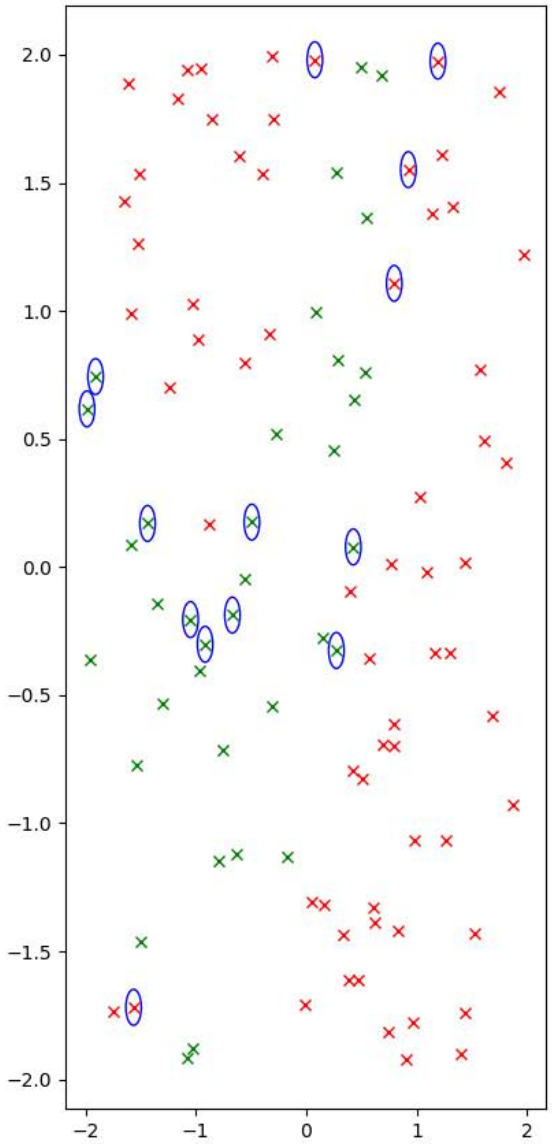
Training set | Validation set | Test set

—> Split the training set and use some of it for validation purposes.

Train for different values of K



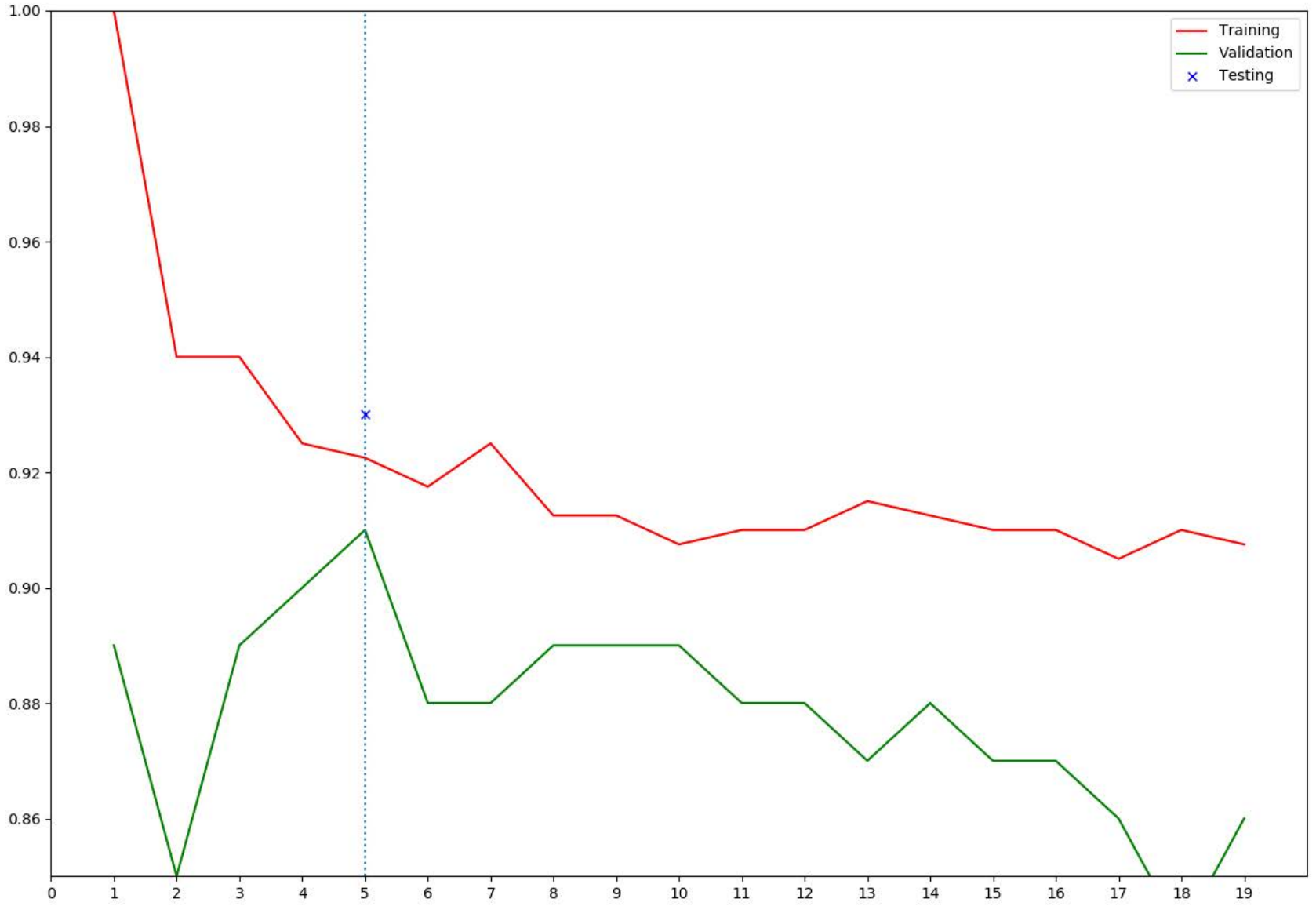
Training set



Validation set

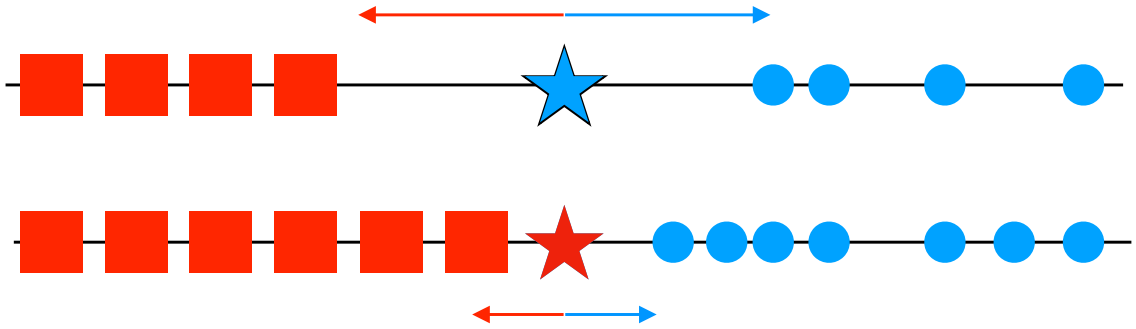
- Run K-Means for k from 1 to 20.
- Use only the training and validation sets.

Pick Best K



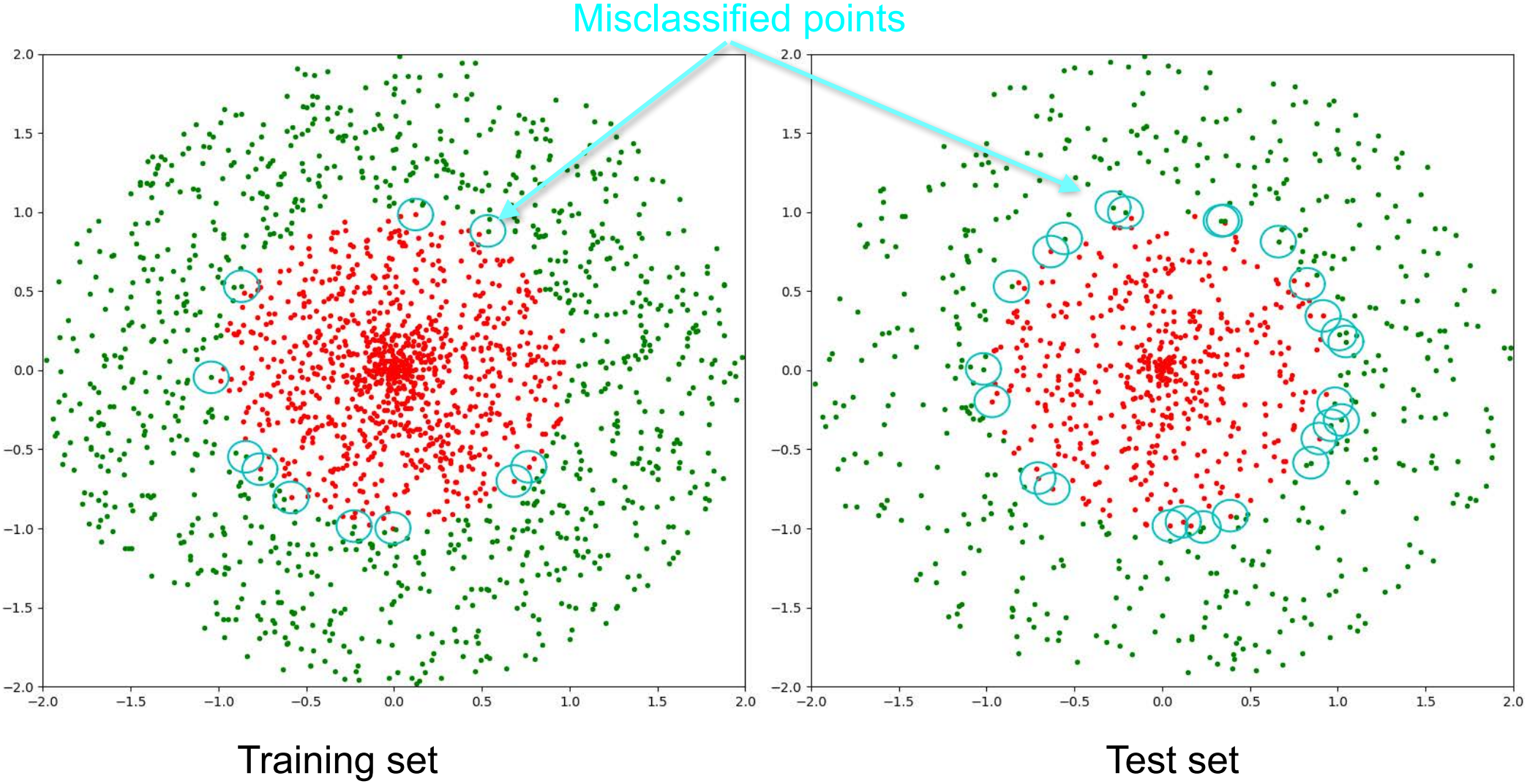
Pick the k value that yields the lowest error on the validation set.

Trouble at the Border



Depending on the how the training set is sampled, classification results may change.

Trouble at The Border

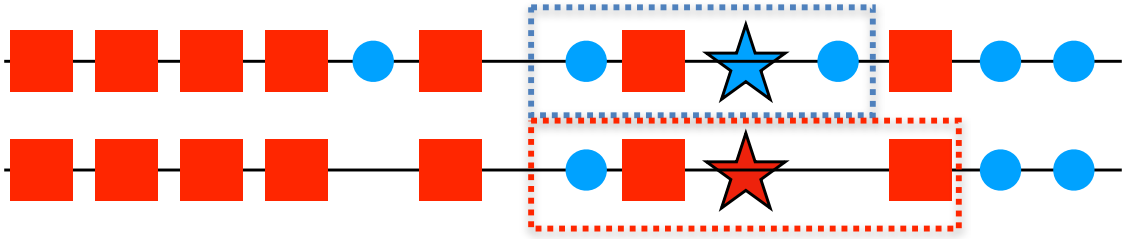


The Knn algorithm is prone to misclassifying points near the decision boundary.

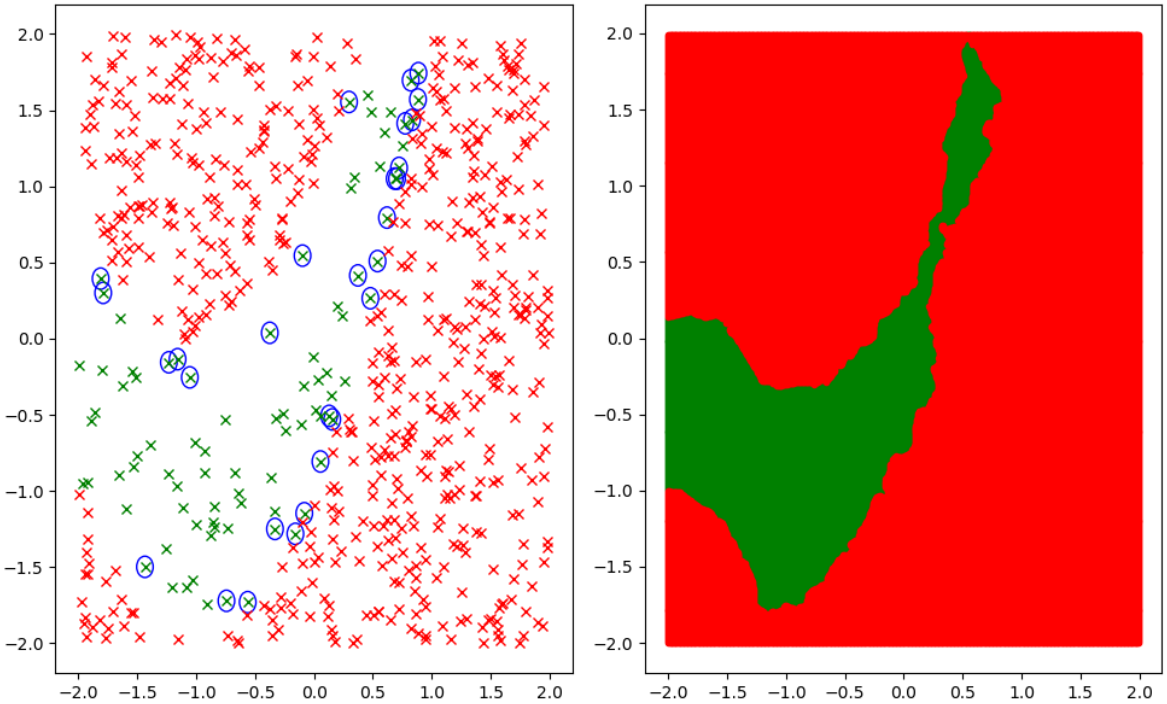
Unbalanced Training Set

What happens when there are more training examples of one kind than the other?

1D

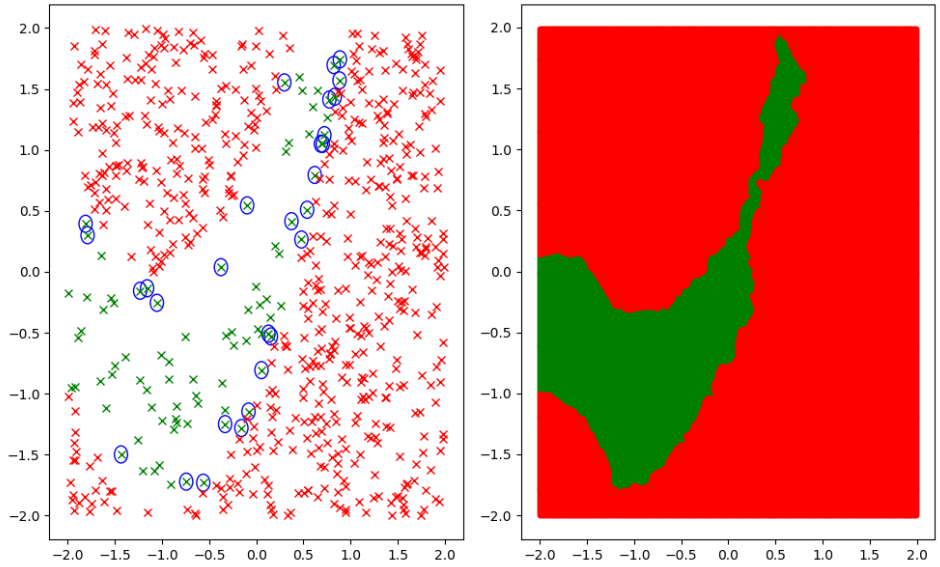


2D



—> The better represented class is unduly favored.

Imbalanced Training Set



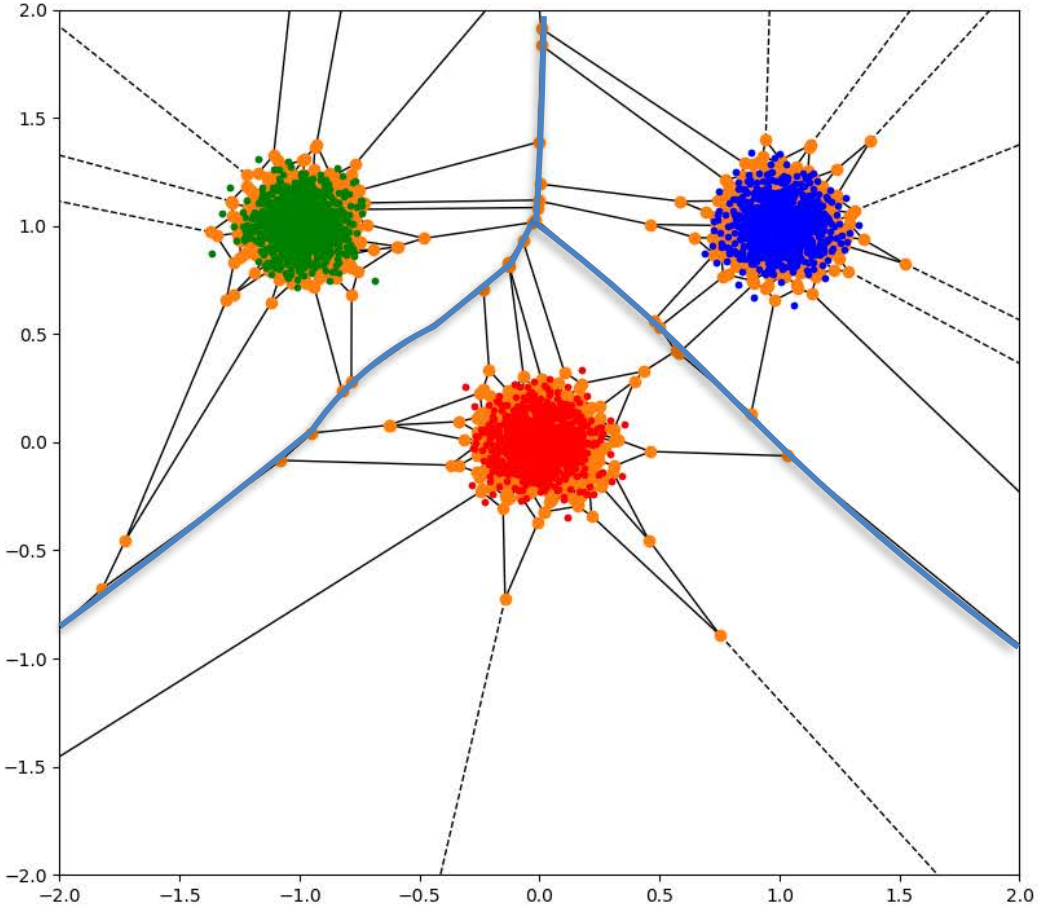
Imbalanced classes appear in many domains, including:

- Fraud detection.
- Spam filtering.
- Disease screening.

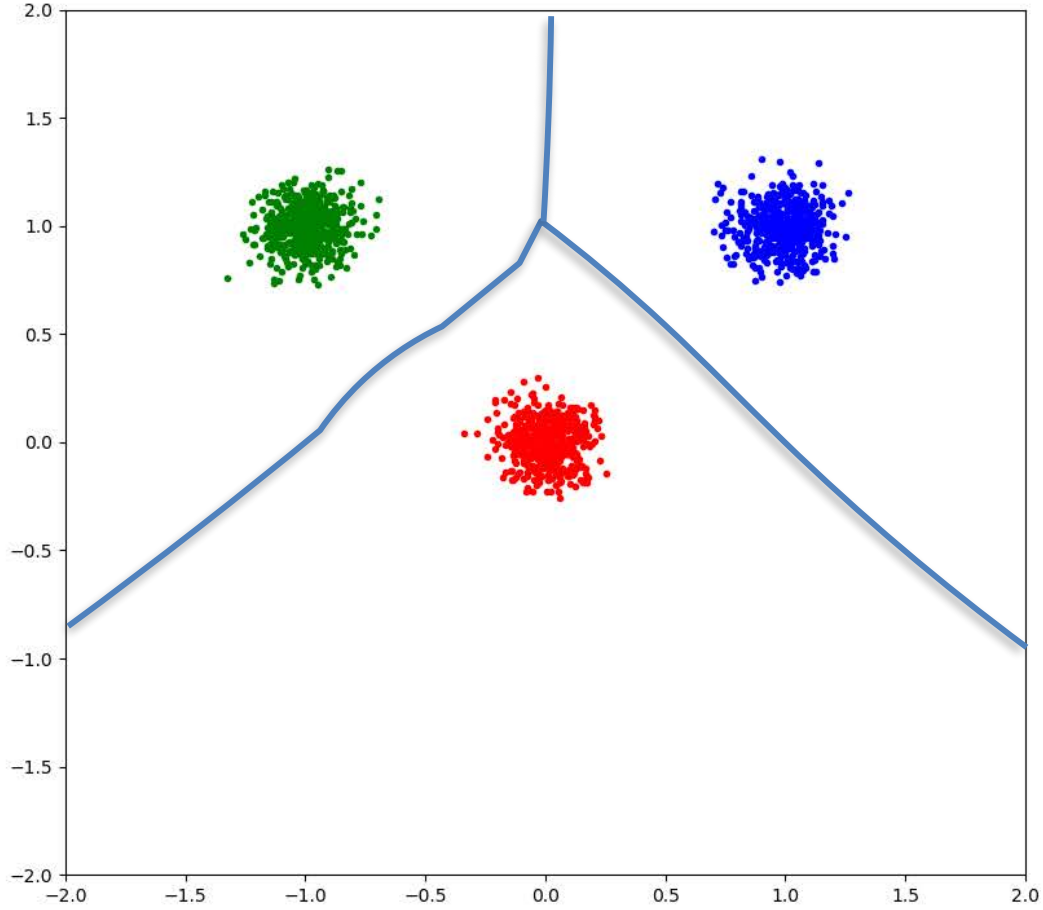
Potential remedies:

- Weighing neighbors by the inverse of their class size converts neighbor counts into the fraction of each class that falls in your K nearest neighbors.
- Under-sampling the dominant class.
- Augmenting the other class by generating synthetic examples.

Multi-Class k-NN



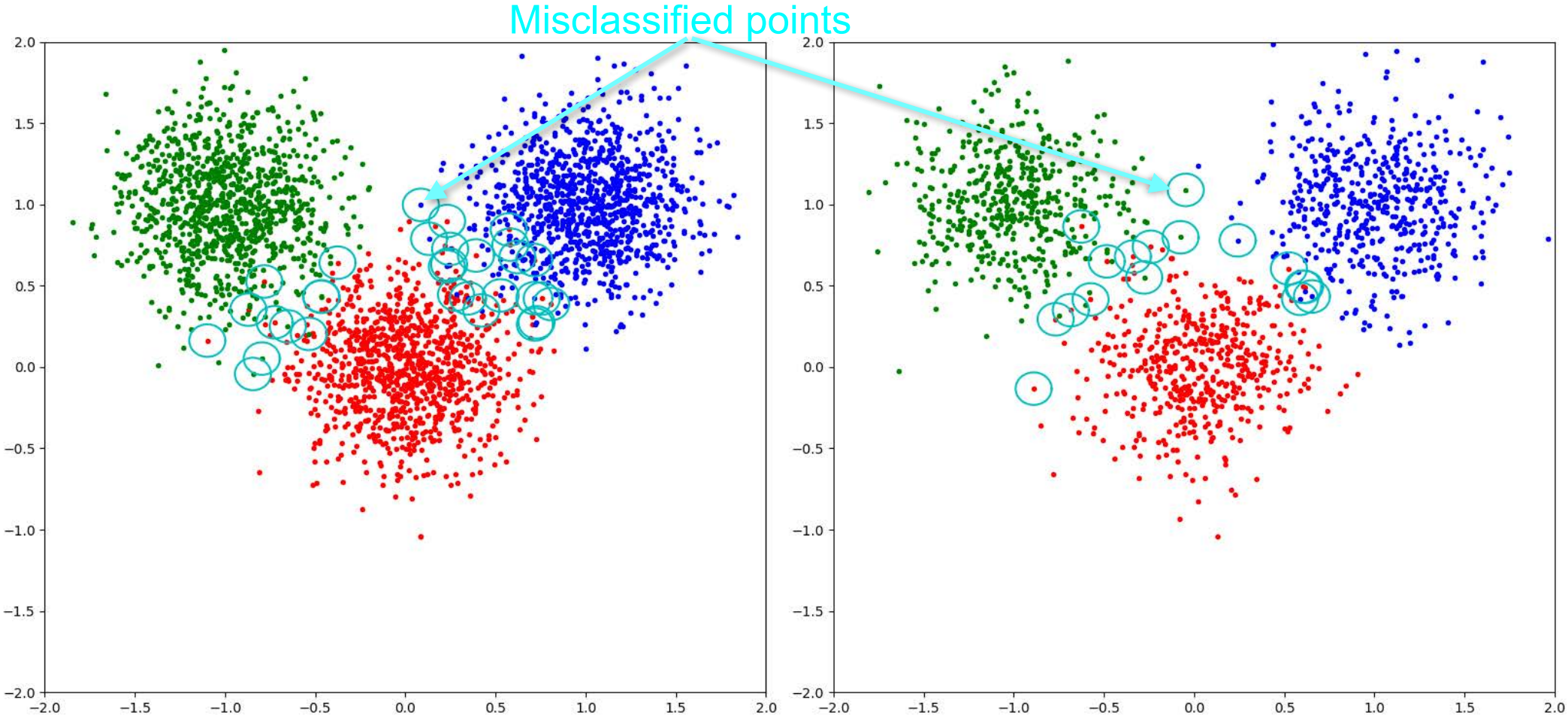
Training set



Test set

—> The decision boundary is still formed by selected edges of the Voronoi Diagram.

Multi-Class k-NN



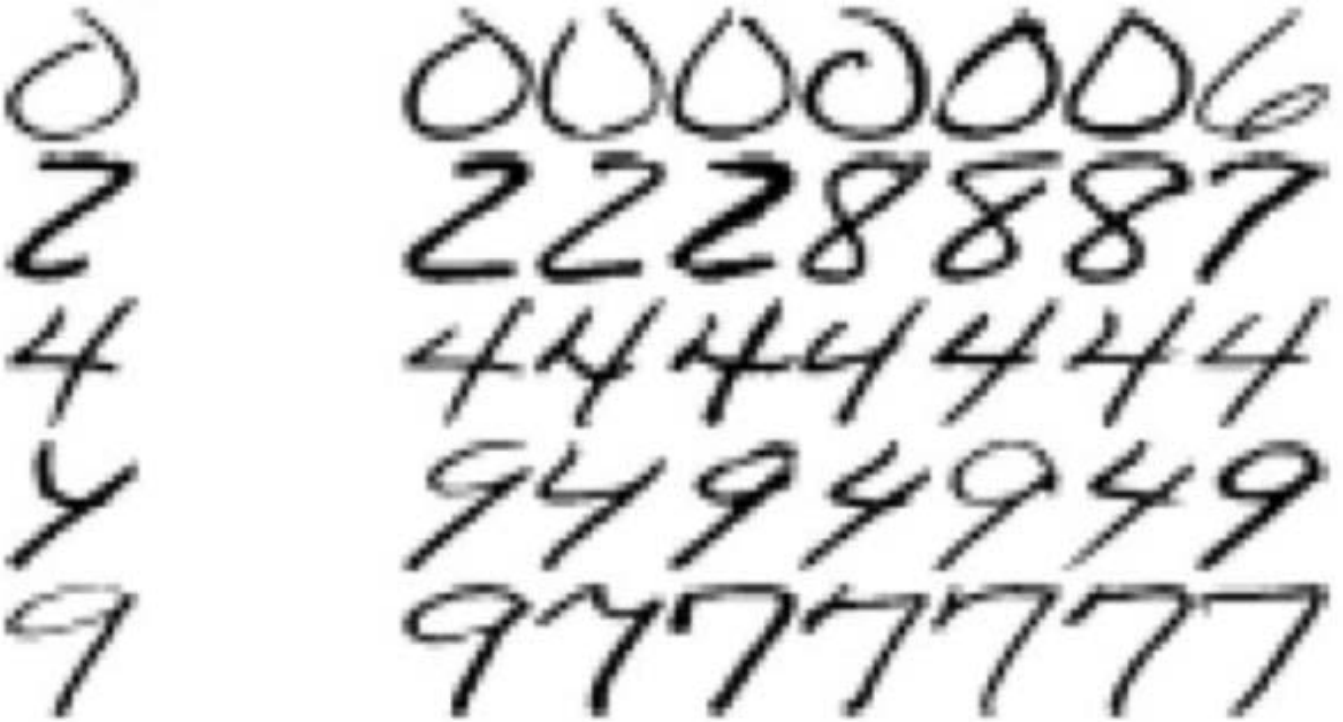
Training set

Test set

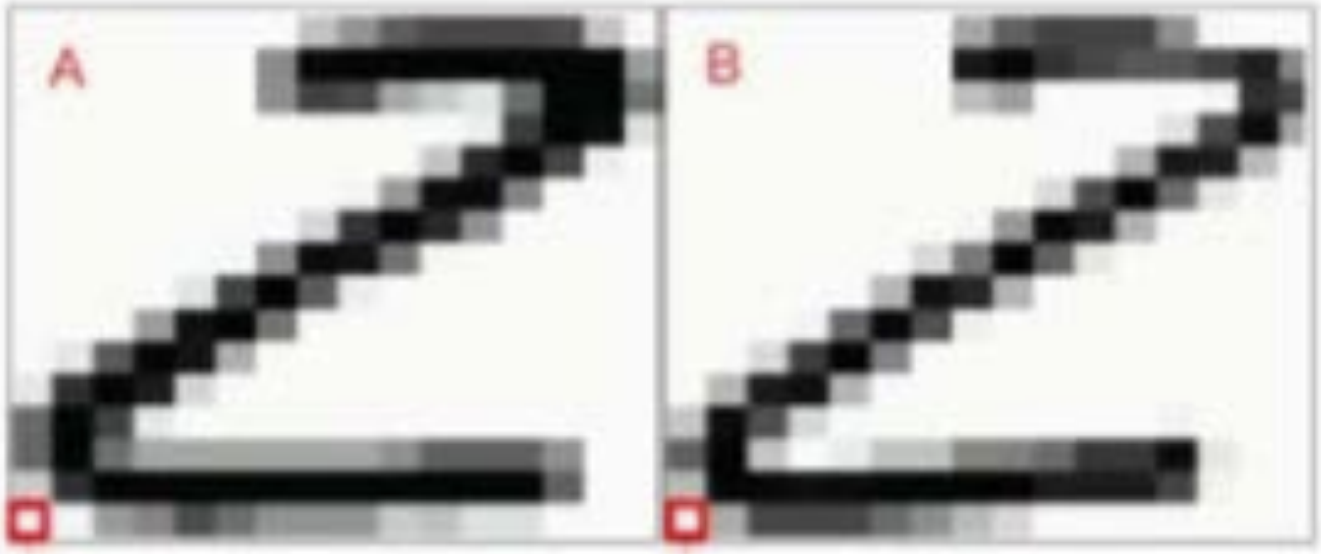
Recognizing Hand-Written Digits

Test sample

Nearest neighbors

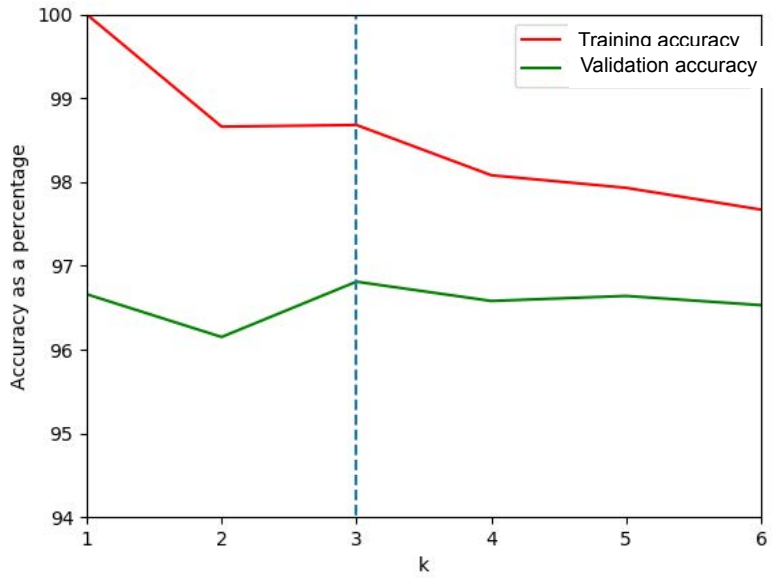


Distance Between Two Samples



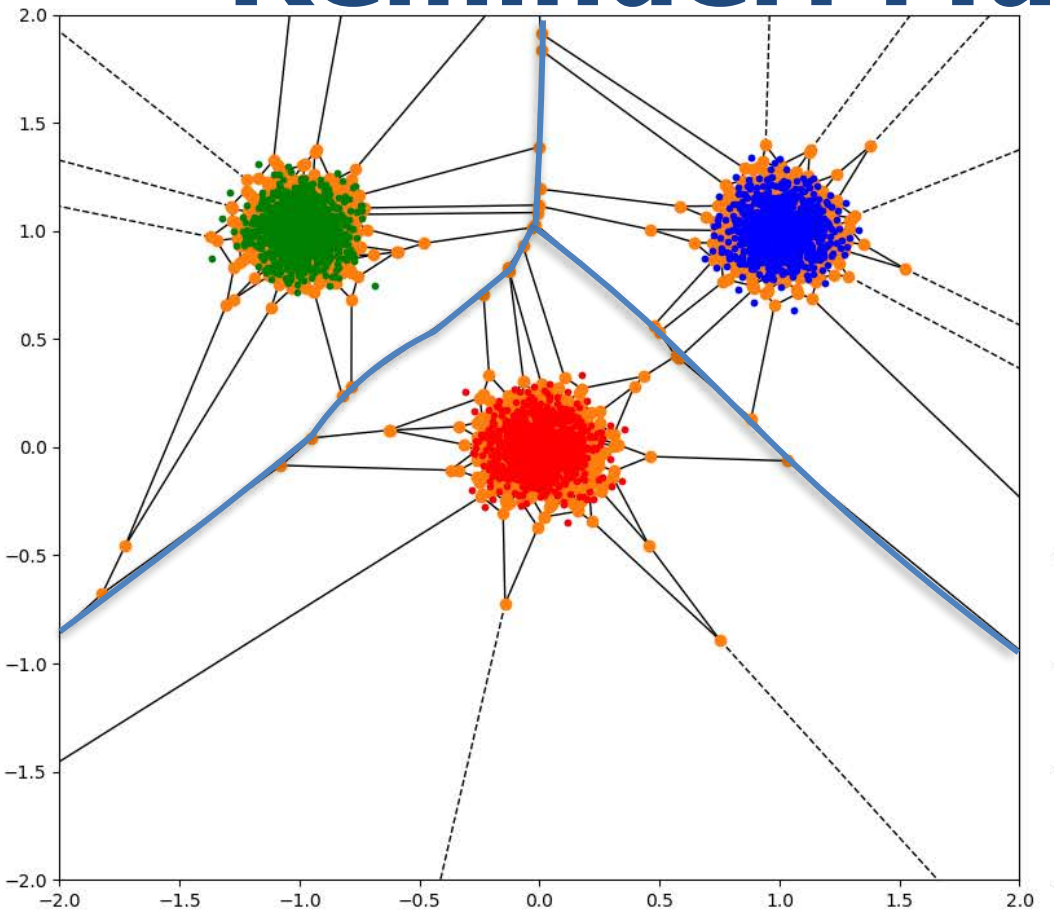
$$\text{dist}(A, B) = \sqrt{\sum_{i,j} (A(i,j) - B(i,j))^2}$$

Validation

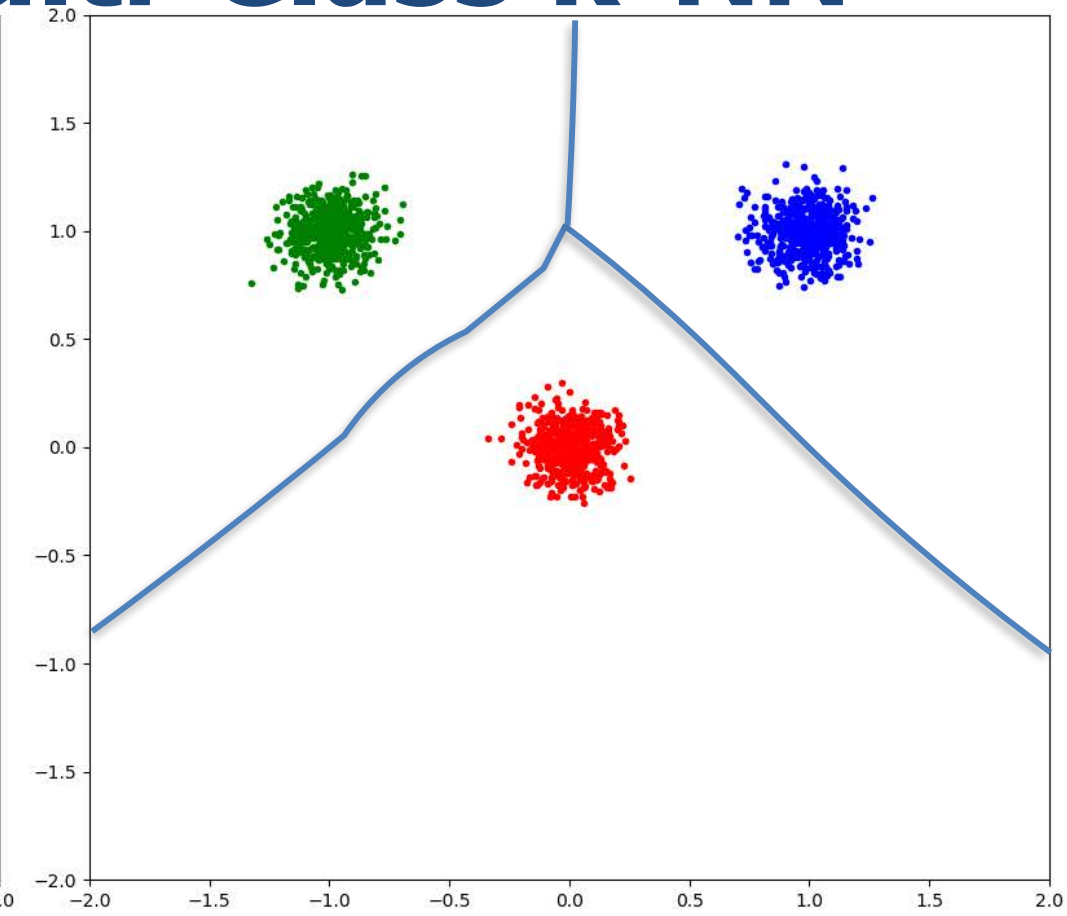


- Simple but effective classification method: On MNIST, 60k training samples, 10k validation samples, 96.8% accuracy.
- Only one single meta-parameter k, which can easily be tuned using cross-validation.

Reminder: Multi-Class k-NN



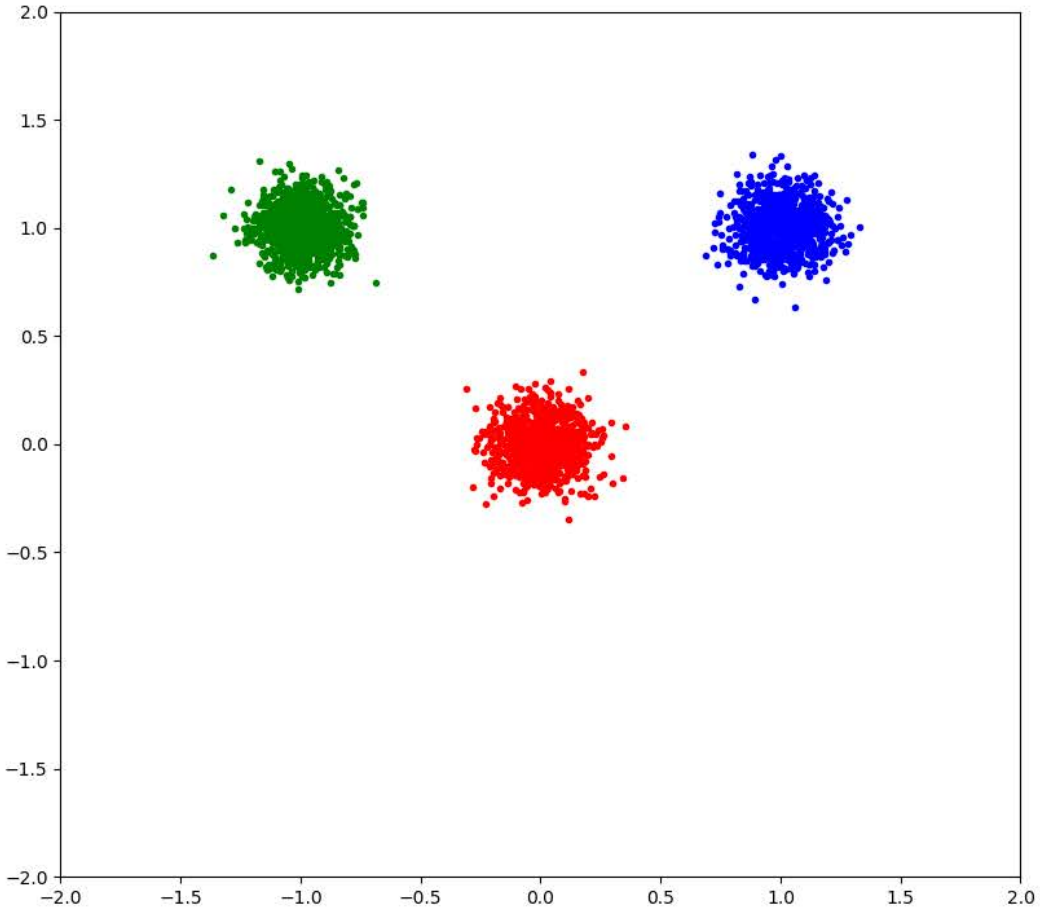
Training set



Test set

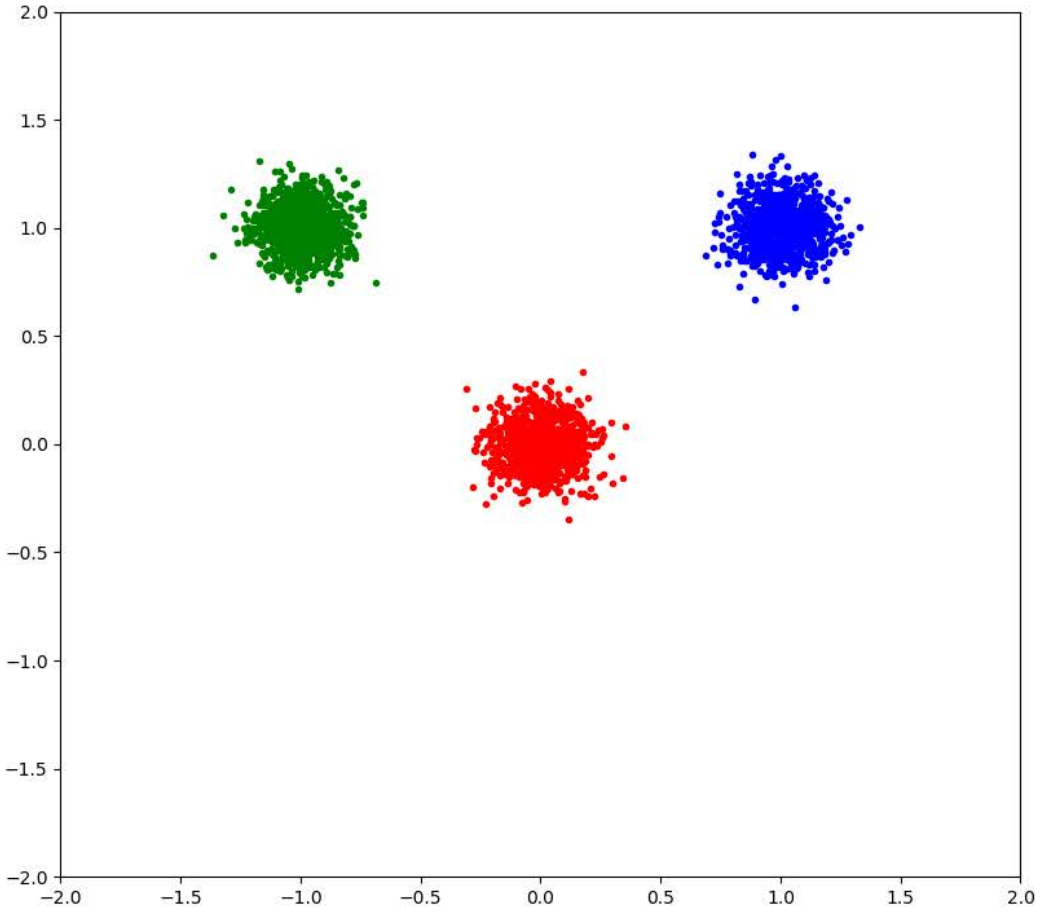
- The decision boundary is formed by selected edges of the Voronoi Diagram.
- Classifying a point requires comparing it to all points in the learning set.

Data Reduction

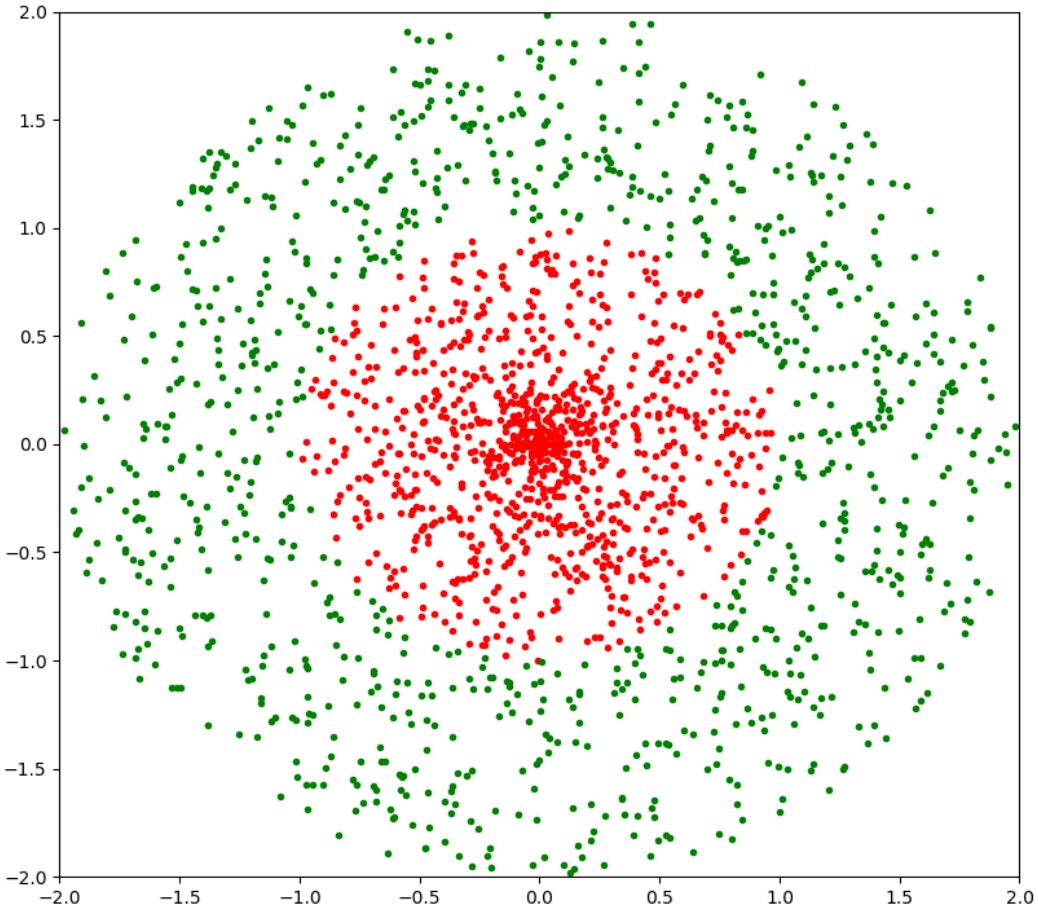


- Do we really need all the points in the training set?
- Can we replace them by a smaller set of prototypes?
- How do we find them?

Data Reduction

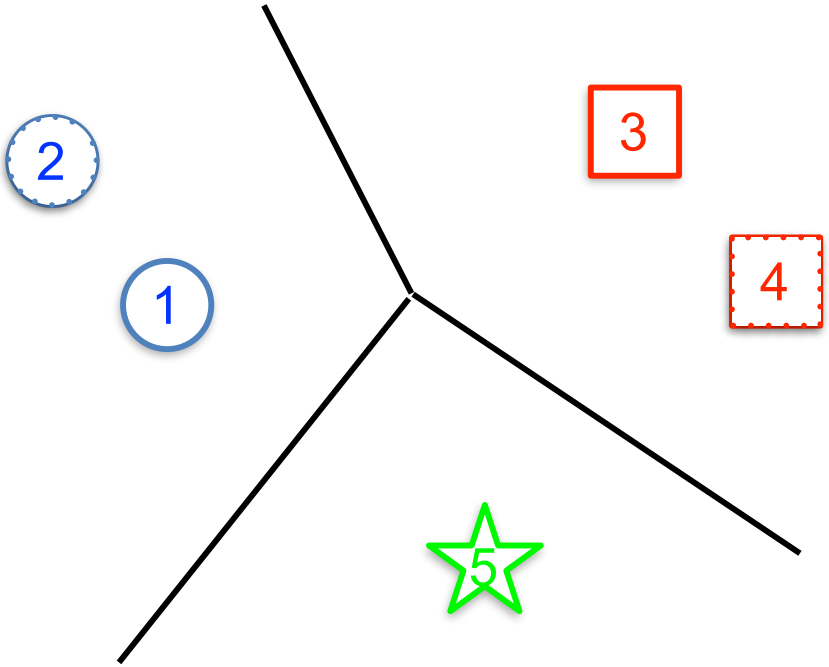


In this case we could use the centers of gravity …



… but not in this one.

Condensed Nearest Neighbors



Initialization: X = {2,3,4,5}
P = {1}

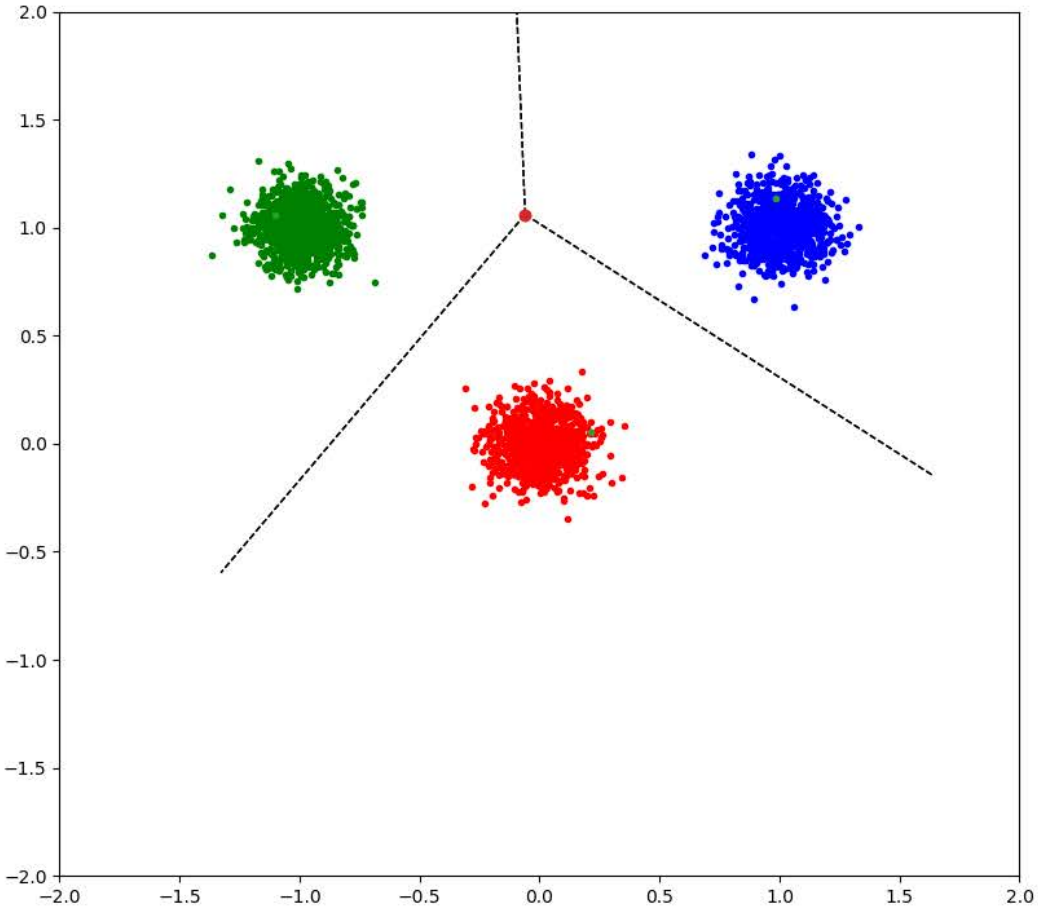
Iteration 1: X = {2,4,5}
P = {1,3}

Iteration 2: X = {2,4}
P = {1,3,5}

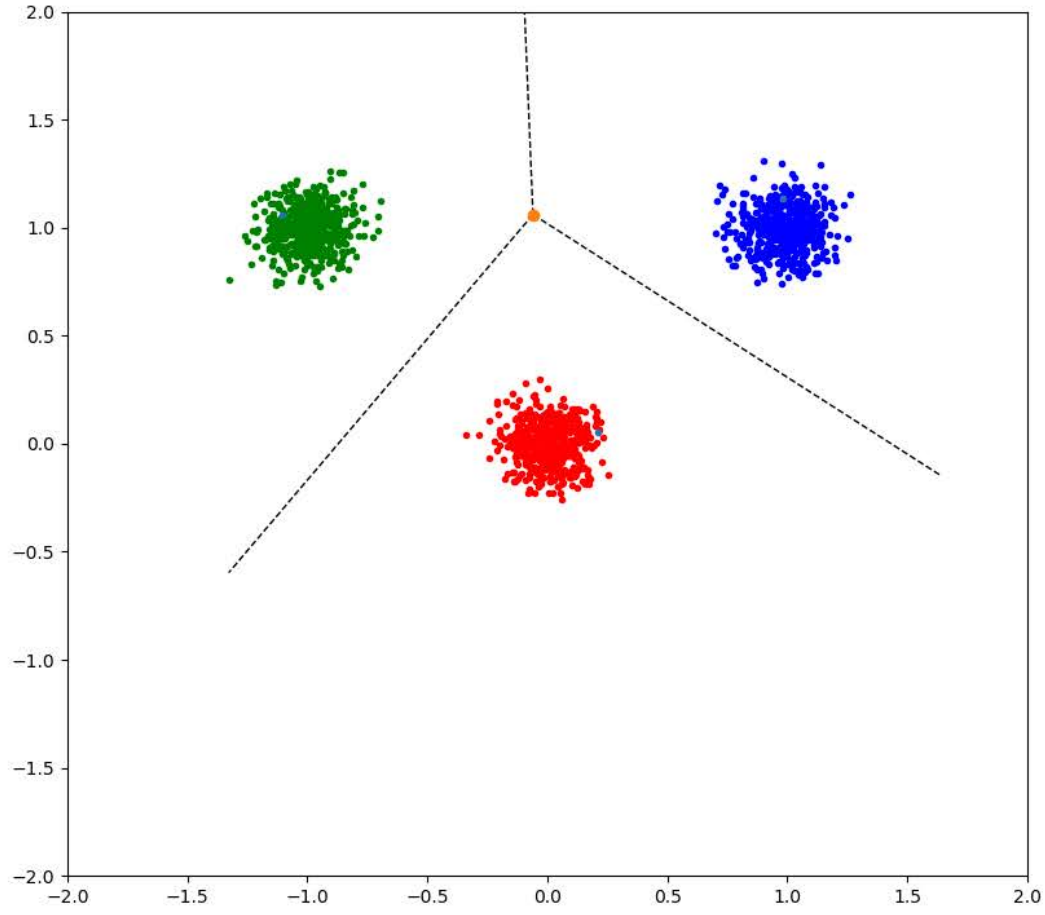
Let X be the set of training samples and P the set of prototypes:

- Initialize
- Repeat
  1. Look for **x** in X such that its nearest prototype in P has a different label than itself.
  2. Remove **x** from X and add it to P.

Condensed Multi-Class 1-NN



Training set

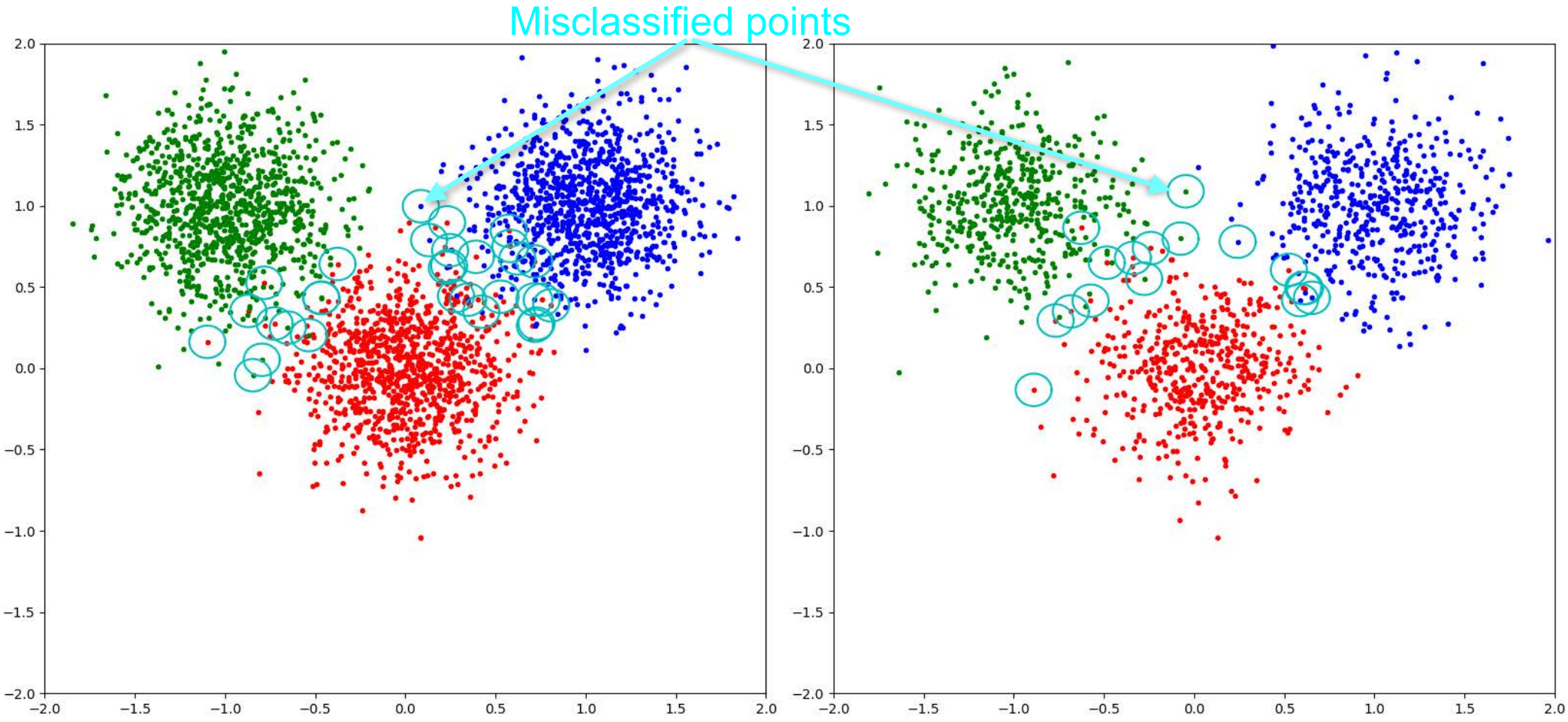


Test set

- There are only three cells in the Voronoi diagram.
- Classifying a point only requires comparing it to the three prototypes.

—> Fast computation at inference time.

Reminder: Standard Multi-Class k-NN

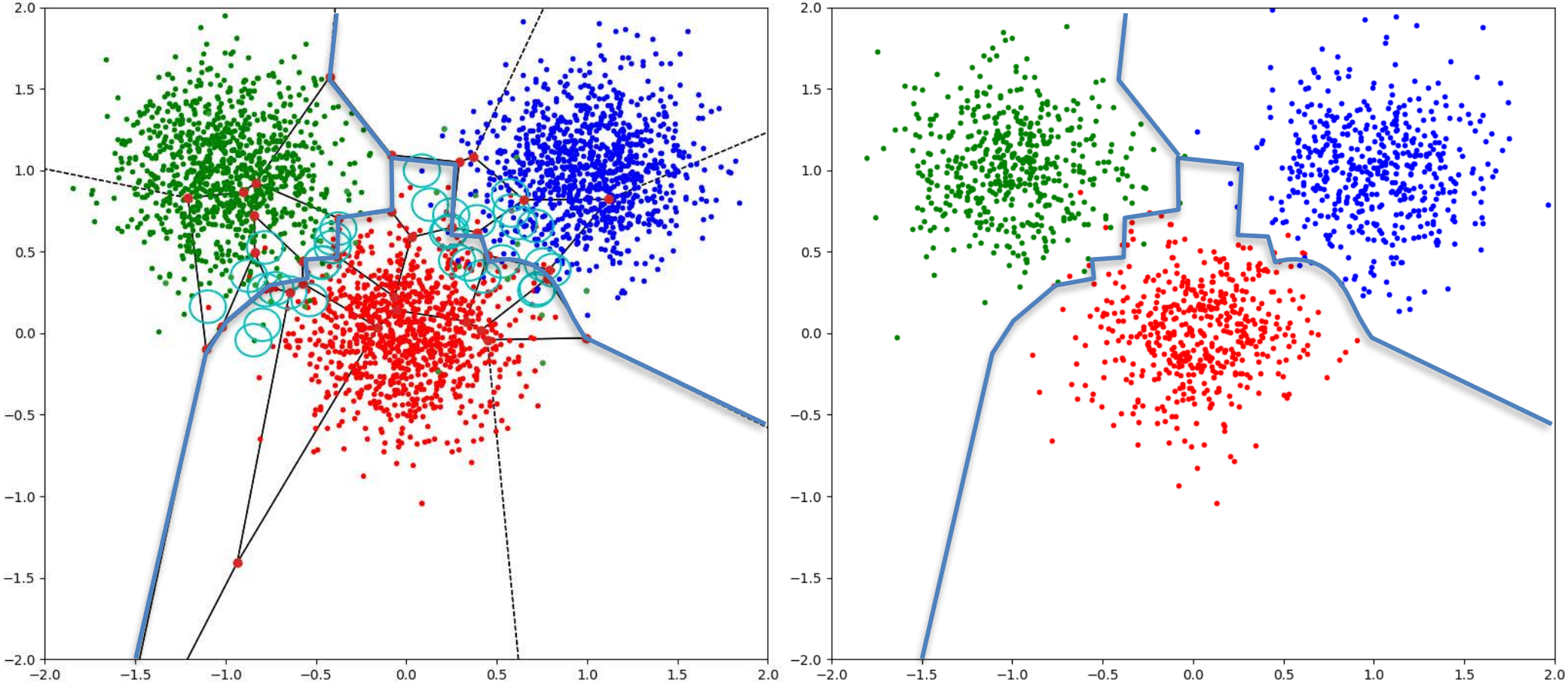


Training set

Test set

Classification errors occur mostly near the decision surface.

Condensed Multi-Class 1-NN

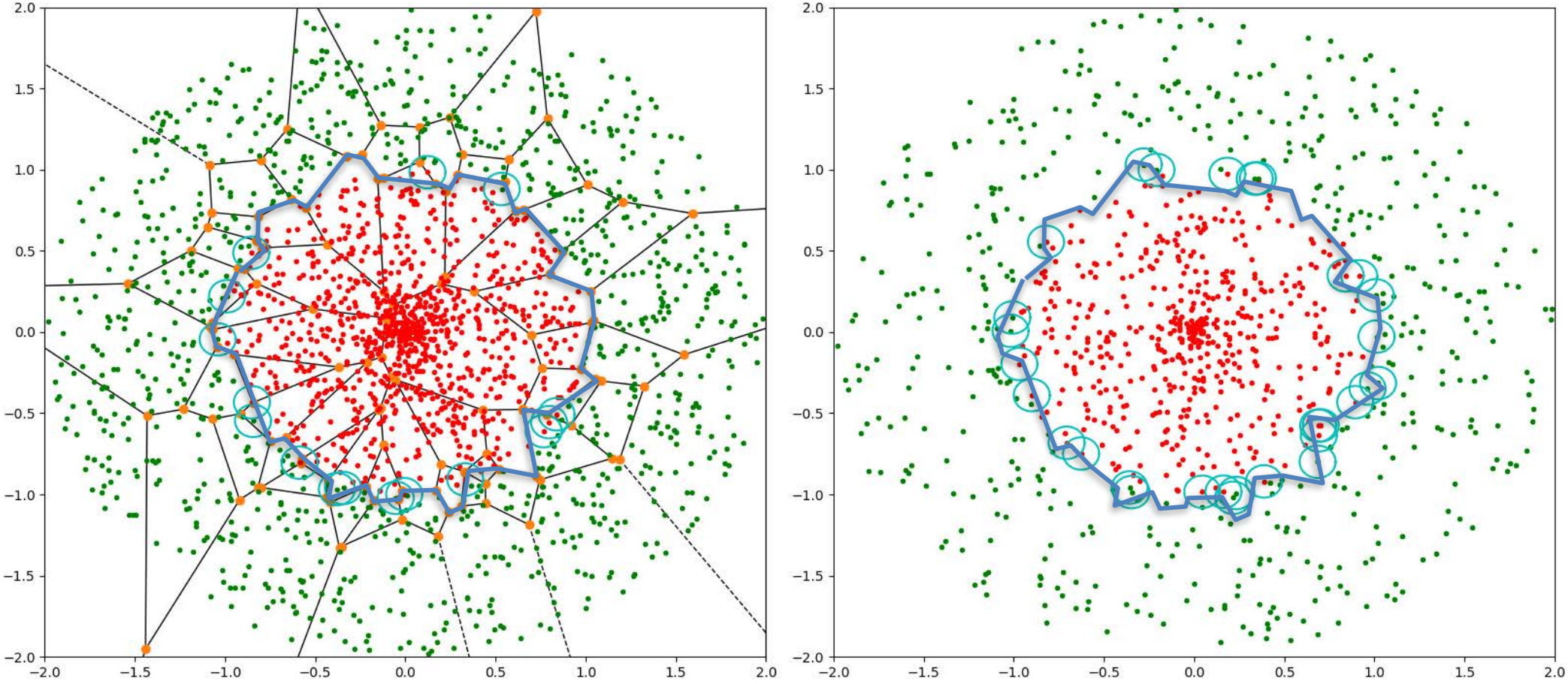


Training set

Test set

The 1-NN algorithm on the prototypes has about the same accuracy than the k-NN algorithm on the original training set and is much faster.

Condensed Binary 1-NN

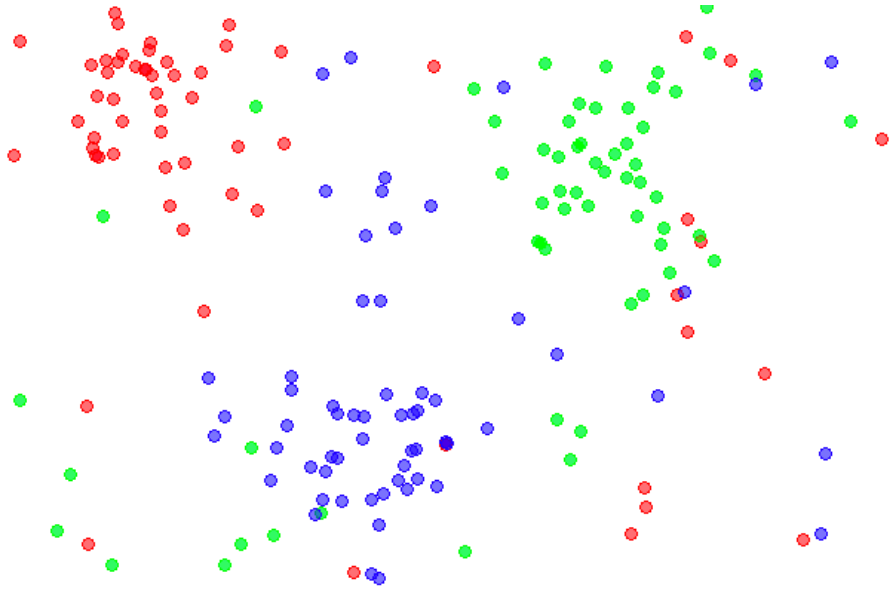


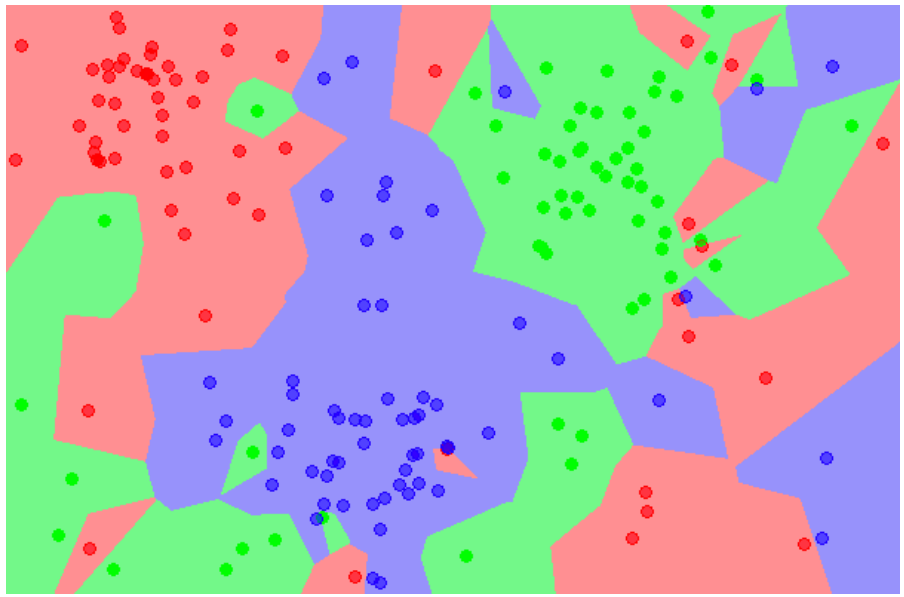
Training set

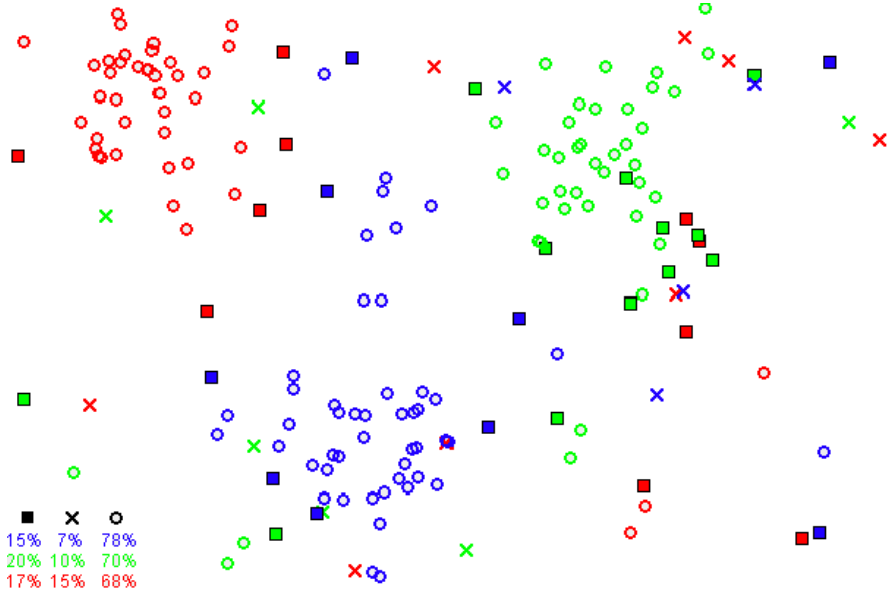
Test set

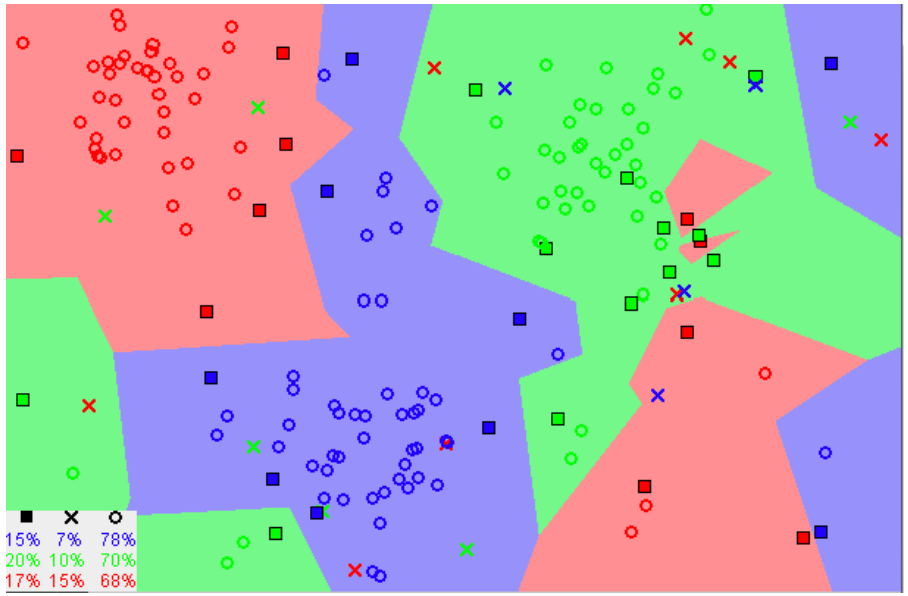
The 1-NN algorithm on the prototypes has about the same accuracy than the k-NN algorithm on the original training set and is much faster.

Multi-Class 1-NN









Application Example: Recommender Systems

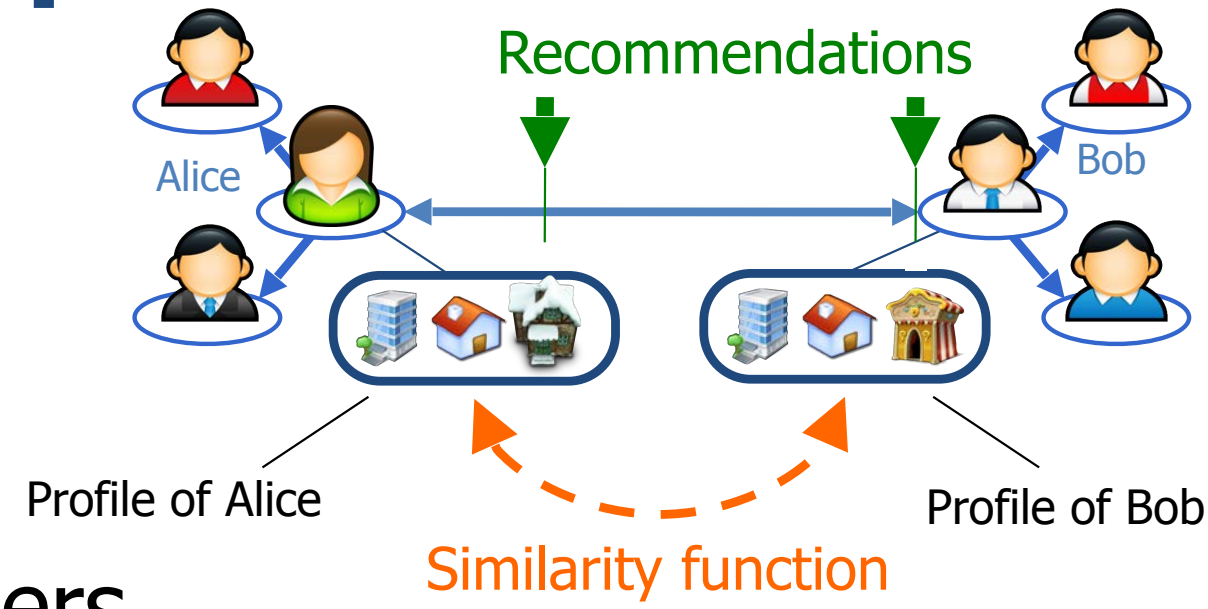
Feature vector:

- What films have you watched?
- Did you like them?

Predictor:

- List of films to propose.

k-NN Graph Construction

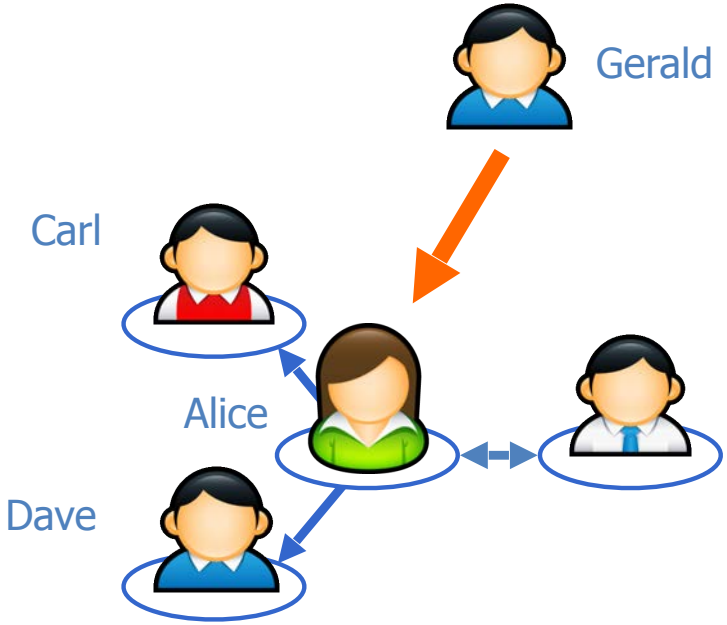


- **Entities:** Users.
- **Distance:** Similarity in movie choices
- **Goal**: For each user find **k closest ones.**

- **Complexity:** $O(N^2)$

—> This does not scale up!

Greedy k-NN Graph Construction

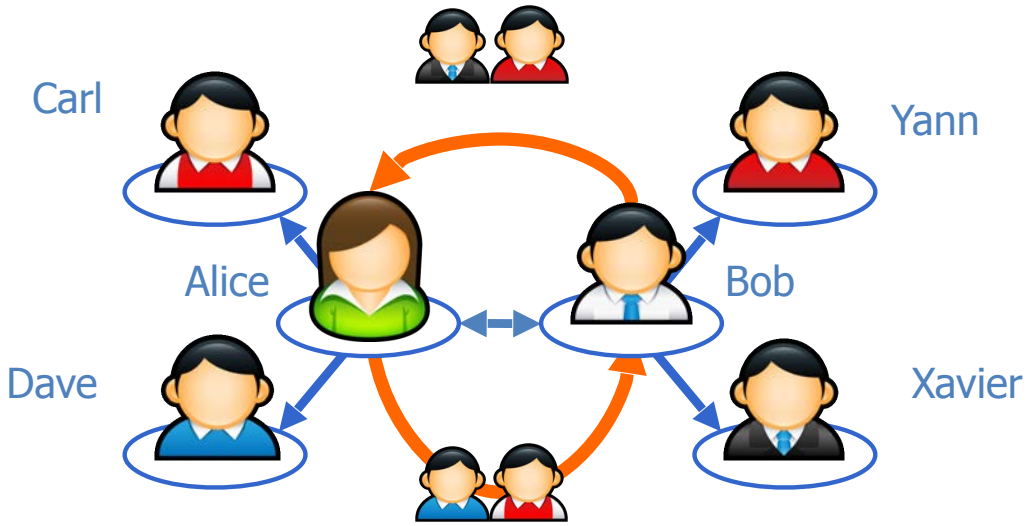


Parallel-iterative algorithm:

Given a **random** graph, each node looks for potential new neighbors:

1. Among random nodes (optional).

Greedy k-NN Graph Construction

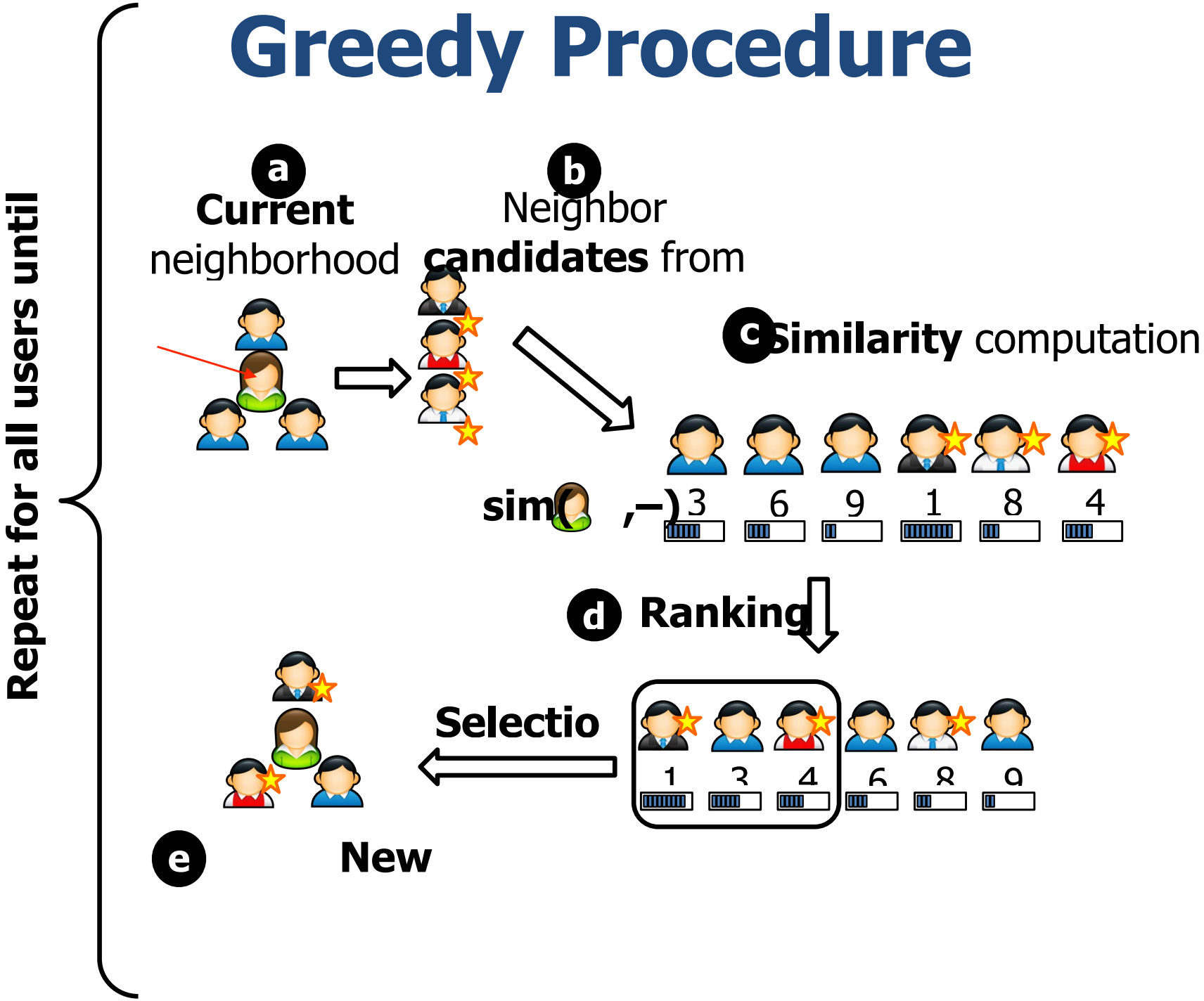


Parallel-iterative algorithm:

Given a **random** graph, each node looks for potential new neighbors:

1. Among random nodes (optional).
2. Among "friends of friends".

Greedy Procedure



Gossip Based Computing

- Highly parallel.
- Creates a random graph.
- Robust to churn, partition, breakdowns.

—> Well adapted to peer-to-peer networks.

Number of Neighbors

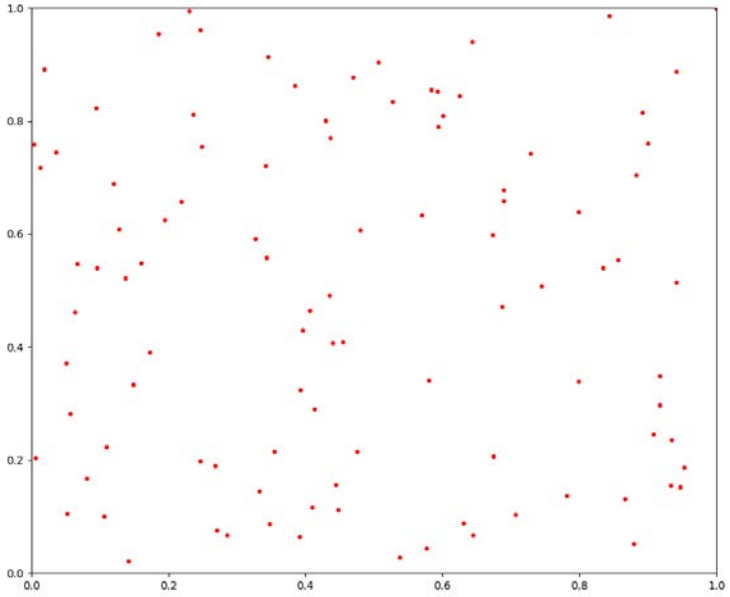
- More neighbors: More coverage.
- Fewer Neighbors: Better accuracy.

—> In practice, between 25 and 100 for practical recommender systems.

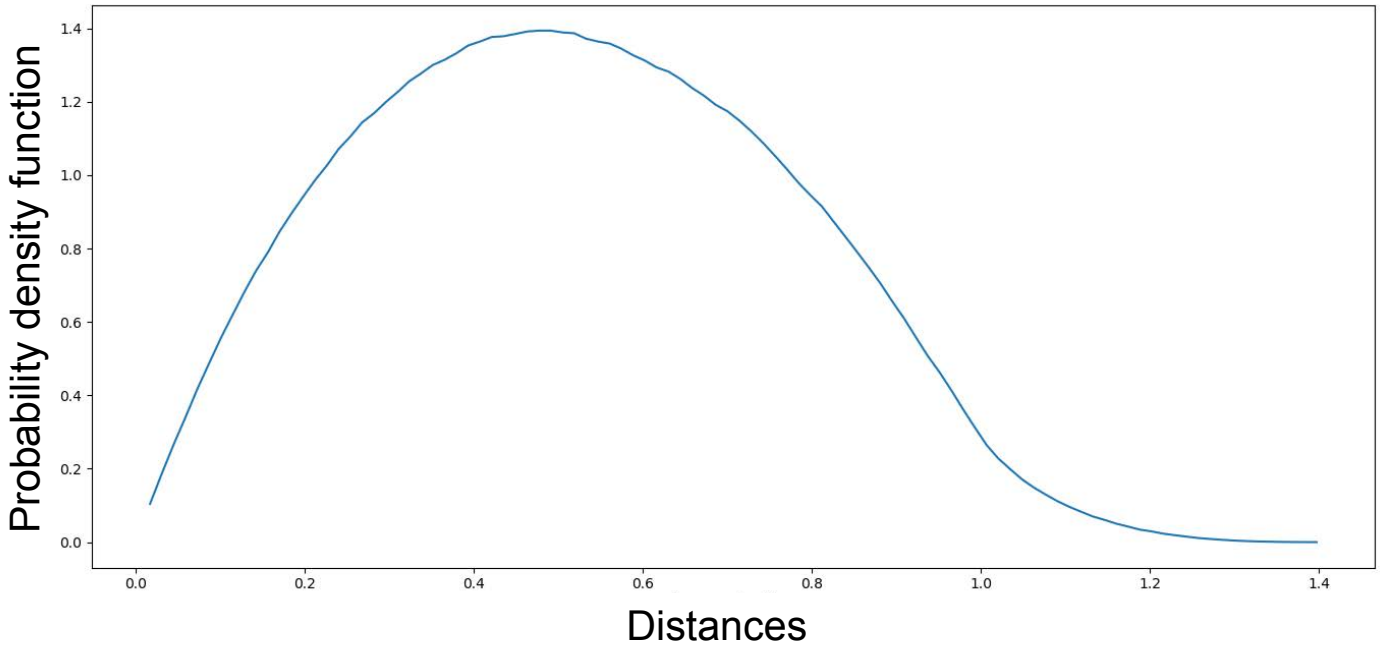
Knn Limitations

- **Performance issues:** Must load all of the training data and calculate distances to all training samples. It can be done in a naive way or using fancier data structures such as K-D trees. However, this is still slow for large datasets.
- **Distance metric:** The vanilla version is used with the simple Euclidean distance, which is a problematic distance metric in high dimensions as well as with noisy features or features of different type.

Distribution of Distances in 2-D

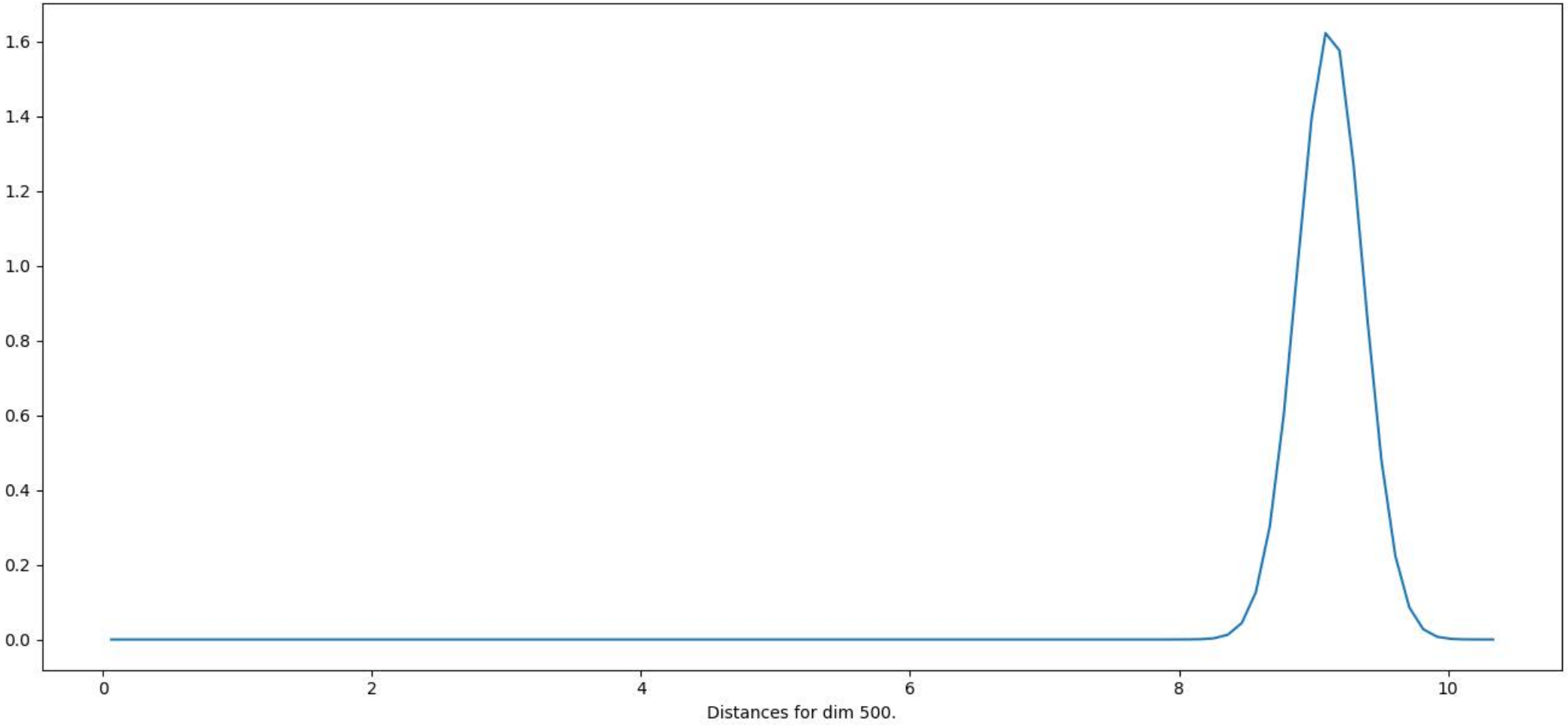


Uniformly distributed points



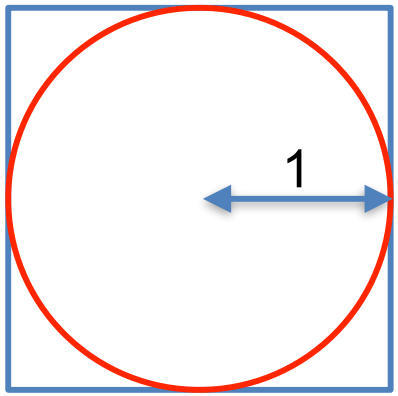
Normalized histogram of pairwise distances

Distribution of Distances in N-D

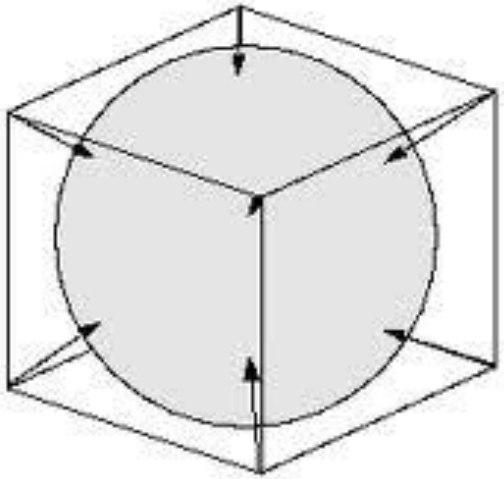


- In a high-dimensional space, everything is far from everything else.
- The Euclidean distance becomes less and less meaningful as the distance increases.

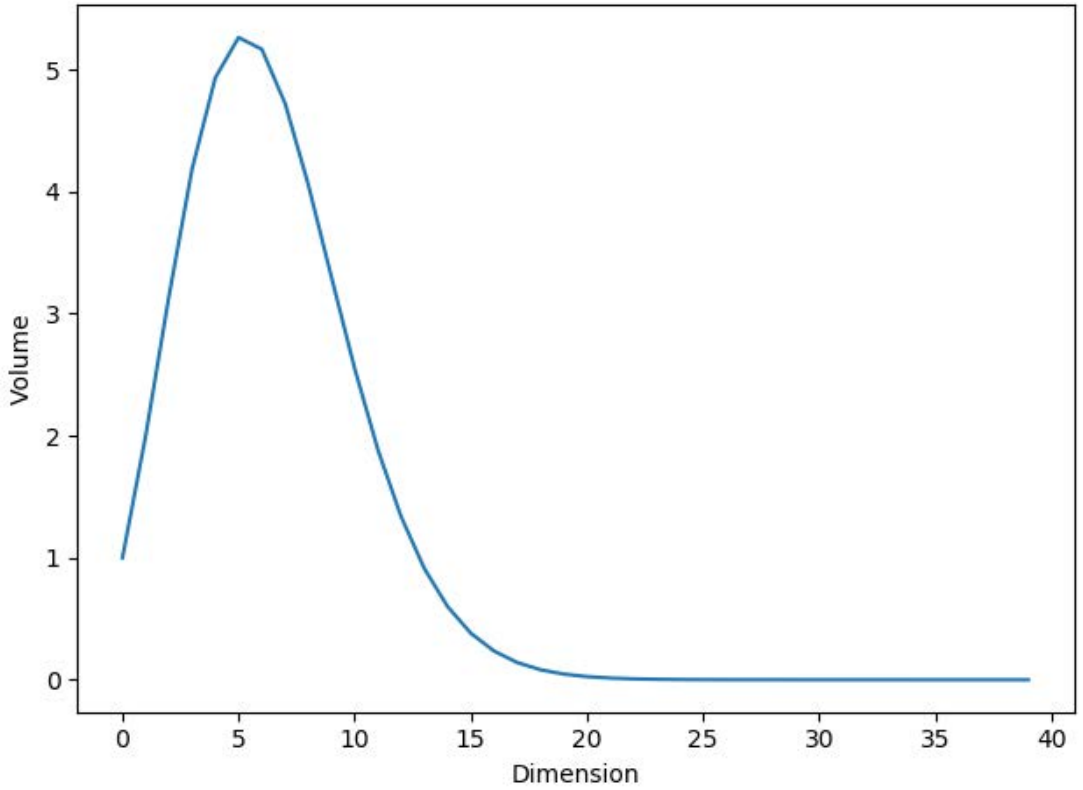
Volume of a Sphere in N-D



3.142



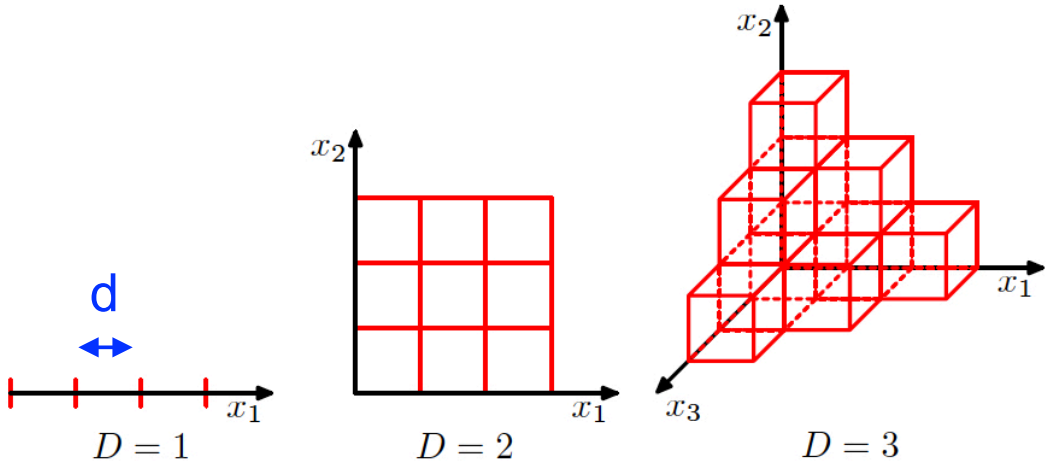
4.189



- In a high-dimensional space, the volume of a sphere (n-ball) of radius 1 becomes a small fraction of the enclosing hypercube.
- All the "mass" is in the corners.

The Curse of Dimensionality

To guarantee the effectiveness of an estimator, the distance between neighboring training samples must be less than some value d that depends on the problem.



d = 1/n
n = 1/d

$n = (1/d)^2$

$n = (1/d)^3$

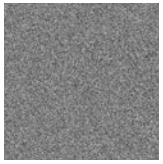
In D dimensions, $n=(1/d)^D$. Assuming that each dimension is coded by 32-bit floating point number, storing the training set would require $4*D*(1/d)^D$ bytes.

D = 10 d=0.05 —> 4.10 10^{14} bytes = 0.41 petabytes.
D = 20, d=0.05 —> 8.38 10^{27} bytes = billions of hexabytes.

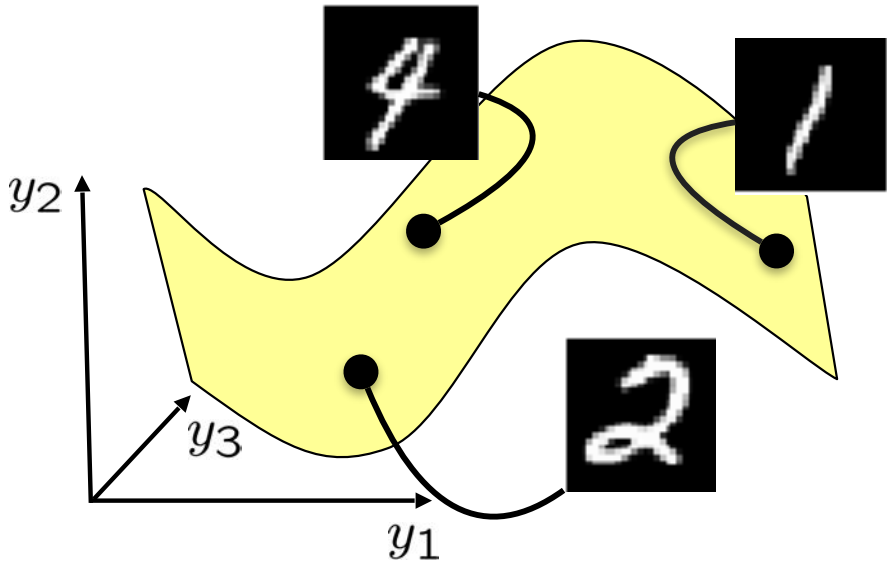
Yet, in practice it often works!!

Images

- The Mnist Images are 28x28 and are represented by 784-D vectors.

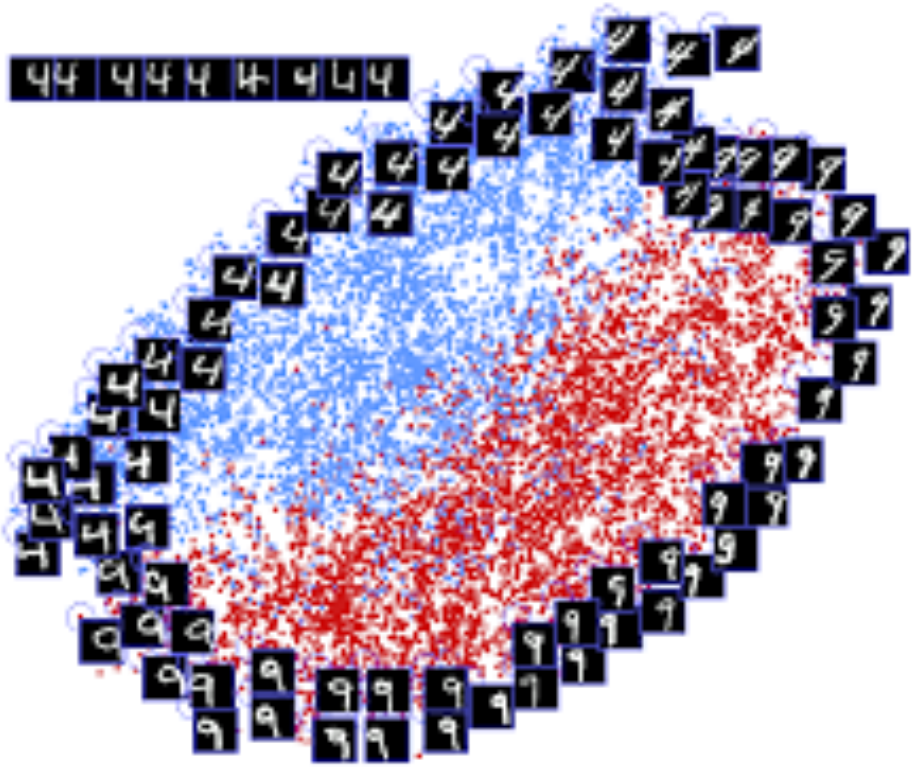


- But we never see images like this one.



—> Real data often lies in much smaller subspaces.

Dimensionality Reduction

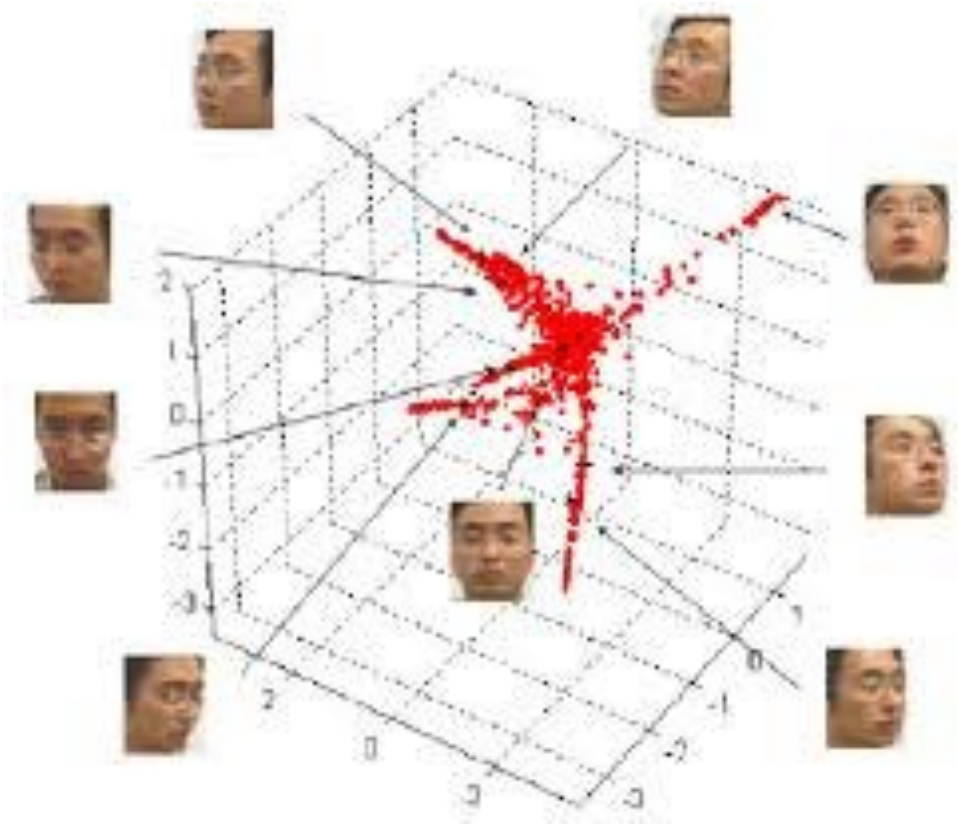


Four and Nines

2D Visualization

- The MNIST images are 28x28 arrays.
- They are **not** uniformly distributed in R^{784}.
- In fact they exist on a low dimensional manifold.

Face Images



- The same can be said about face images.
- And of many other things.

—> k-NN classification can be used in practice.

He et al. , PAMI'05 Fan et al. AVBPA'05

A Potential Take on ML

- Machine Learning algorithms are designed to do what k-NN does but without the computational explosion.
- Machine Learning can be thought of as "glorified nearest neighbors".
- This is much less trivial than it sounds because the larger the training database is, the better the algorithms work.
- Companies such as Google or Facebook offer extreme examples of this philosophy put into practice because they have the means to collect and process the required **IMMENSE** databases.