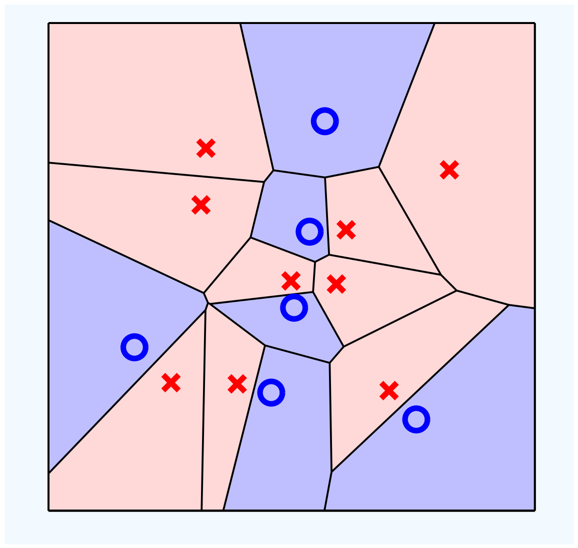
# Nearest neighbor classifiers

## Chapter e-6



## Nearest Neighbor Classification

NN(image):

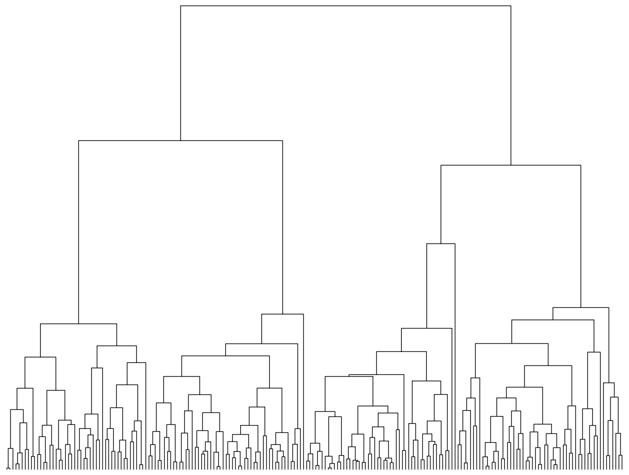
1. Find the image in the training data which is closest to the query image.
2. Return its label.

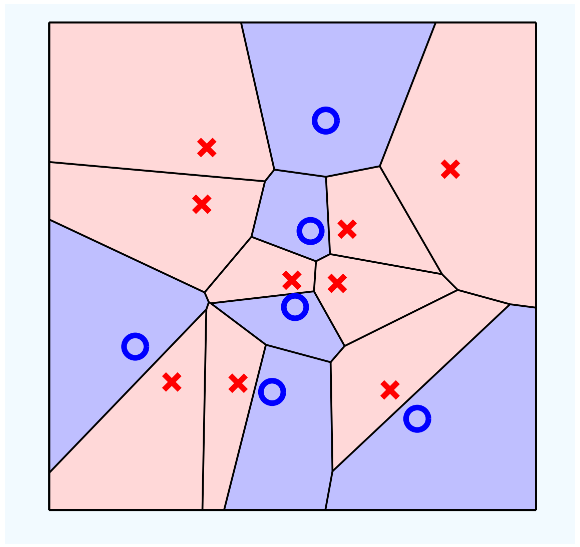
query

closest image

# Distance based methods

- Supervised learning methods (nearest neighbor methods)
- Clustering (k-means and hierarchical clustering)





## Measuring distance

How to measure closeness?

Distance measures for continuous data:

The Euclidean distance:

$$d_2(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - \mathbf{x}'||_2 = \sqrt{(\mathbf{x} - \mathbf{x}')^\intercal (\mathbf{x} - \mathbf{x}')} = \sqrt{\sum_{i=1}^{d} (x_i - x'_i)^2}$$

(based on the 2-norm)

## More distance measures

The Manhattan distance:

$$d_1(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - \mathbf{x}'||_1 = \sum_{i=1}^{d} |x_i - x_i'|$$

This is the distance if we can only travel along coordinate axes.

## The Minkowski distance

The Minkowki distance of order p:

$$d_p(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - \mathbf{x}'||_p = \left( \sum_{i=1}^{d} |x_i - x_i'|^p \right)^{1/p}$$

This is based on the p-norm (sometimes called the $L_p$ norm).

The Euclidean and Manhattan distances are special cases

## The Minkowski distance

The Minkowki distance of order p:

$$d_p(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - \mathbf{x}'||_p = \left( \sum_{i=1}^{d} |x_i - x'_i|^p \right)^{1/p}$$

This is based on the p-norm (sometimes called the $L_p$ norm).

What happens when p goes to infinity?

# The Minkowski distance

The Minkowki distance of order p:

$$d_p(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - \mathbf{x}'||_p = \left( \sum_{i=1}^{d} |x_i - x'_i|^p \right)^{1/p}$$

What happens when p goes to infinity?

$$d_\infty(\mathbf{x}, \mathbf{x}') = \max_i |x_i - x'_i|$$

We get the Chebyshev distance

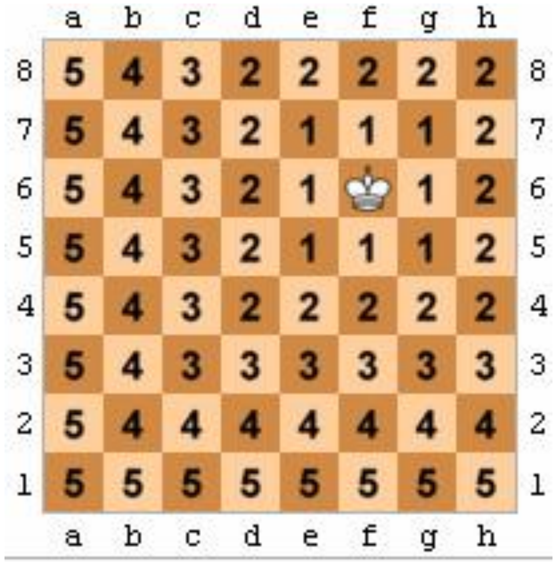
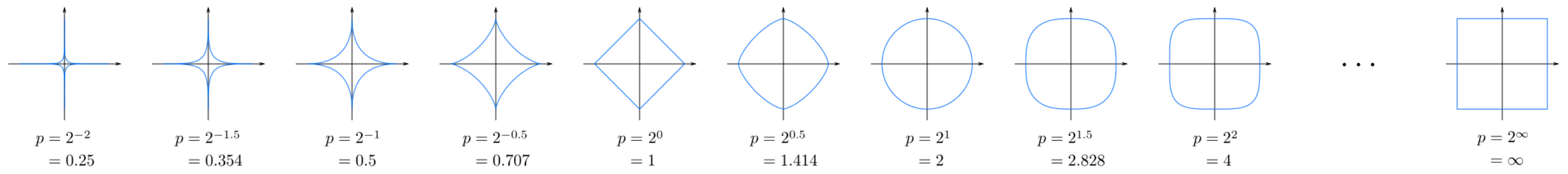


image from https://en.wikipedia.org/wiki/Chebyshev_distance

# The Minkowski distance

## The unit sphere for various values of p



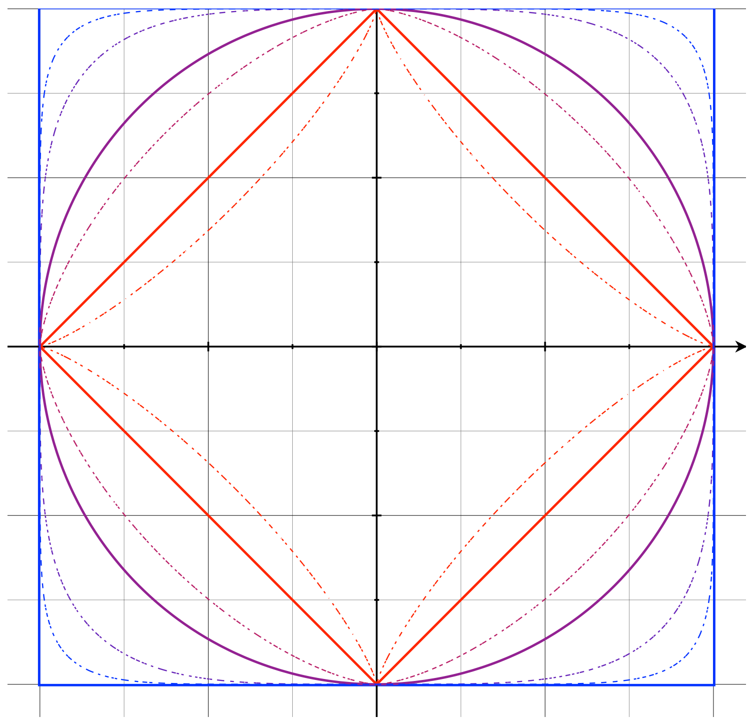


image from https://en.wikipedia.org/wiki/Minkowski_distance

# Properties of a distance

A function d($\mathbf{x}$,$\mathbf{x}'$) is called a distance function if it satisfies the following conditions:

i. d($\mathbf{x}$, $\mathbf{x}$) = 0 (a distance of a point to itself is zero)

ii. d($\mathbf{x}$, $\mathbf{x}'$) ≠ 0 if $\mathbf{x} \neq \mathbf{x}'$ (all other distances are non-zero)

iii. d($\mathbf{x}$, $\mathbf{x}'$) = d($\mathbf{x}'$, $\mathbf{x}$) (distances are symmetric)

iv. d($\mathbf{x}$, $\mathbf{x}'$) <= d($\mathbf{x}$, $\mathbf{x}''$) + d($\mathbf{x}''$, $\mathbf{x}'$) (detours make distances larger)

The last condition is called the triangle inequality

The Minkowski distance with $p<1$ does not satisfy the triangle inequality.

## Data normalization

Is very important in this context!

## The Mahalanobis distance

Sometimes it's useful to use different scales for different coordinates.

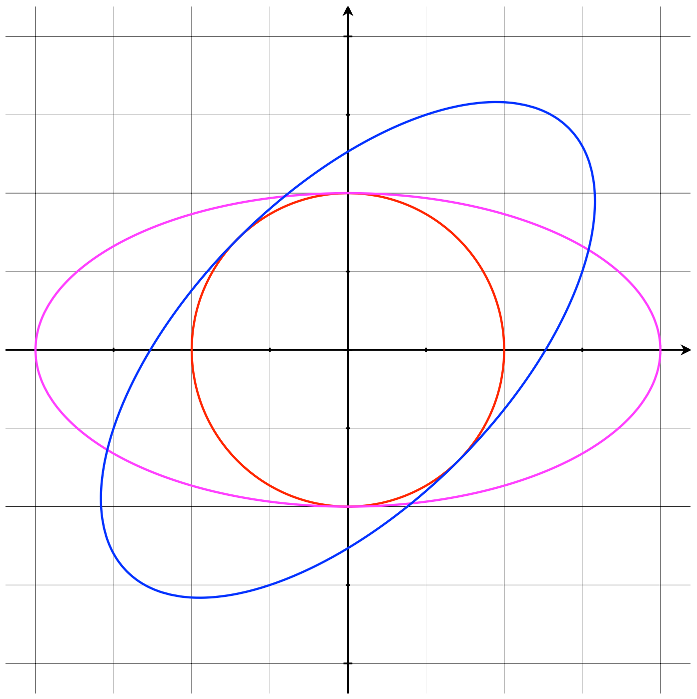
Therefore: use an ellipse rather than a circle to identify points that are a fixed distance away.

Also consider rotating the ellipse.

$$\mathbf{R} = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$$

45 degree clockwise rotation

$$\mathbf{S} = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & 1 \end{pmatrix}$$



$\mathbf{x}^{\mathrm{T}}\mathbf{R}^{\mathrm{T}}\mathbf{S}^2\mathbf{R}\mathbf{x} = 1/4$; the axis-parallel ellipse $\mathbf{x}^{\mathrm{T}}\mathbf{S}^2\mathbf{x} = 1/4$; and the circle $\mathbf{x}^{\mathrm{T}}\mathbf{x} = 1/4$.

## The Mahalanobis distance

The shape of the ellipse can be estimated from data as the inverse of the covariance matrix:

$$Dis_M(\mathbf{x}, \mathbf{y} | \mathbf{\Sigma}) = \sqrt{(\mathbf{x} - \mathbf{y})^\intercal \mathbf{\Sigma}^{-1} (\mathbf{x} - \mathbf{y})}$$

(The inverse of the covariance matrix has the effect of decorrelating and normalizing features)

## Nearest neighbor classification

NN(image):

1. Find the image in the training data which is closest to the query image.
2. Return its label.

query

closest image

# Nearest neighbor classification

The nearest neighbor classifier:

Classify a given test example to the class of the nearest training example.

More formally:

$$\mathcal{D} = (\mathbf{x}_1, y_1) \dots (\mathbf{x}_N, y_N)$$

Reorder the data according to its similarity to an input x:

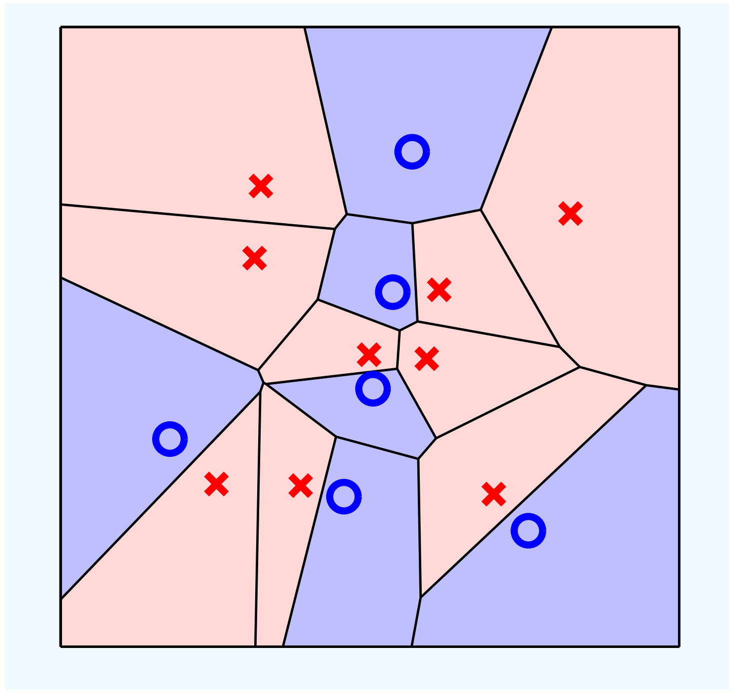
$$(\mathbf{x}_{[n]}(\mathbf{x}), y_{[n]}(\mathbf{x}))$$

i.e. $$d(\mathbf{x}, \mathbf{x}_{[1]}) \leq d(\mathbf{x}, \mathbf{x}_{[2]}) \leq \cdots \leq d(\mathbf{x}, \mathbf{x}_{[N]})$$

The prediction: $$g(\mathbf{x}) = y_{[1]}(\mathbf{x})$$

# Voronoi diagrams

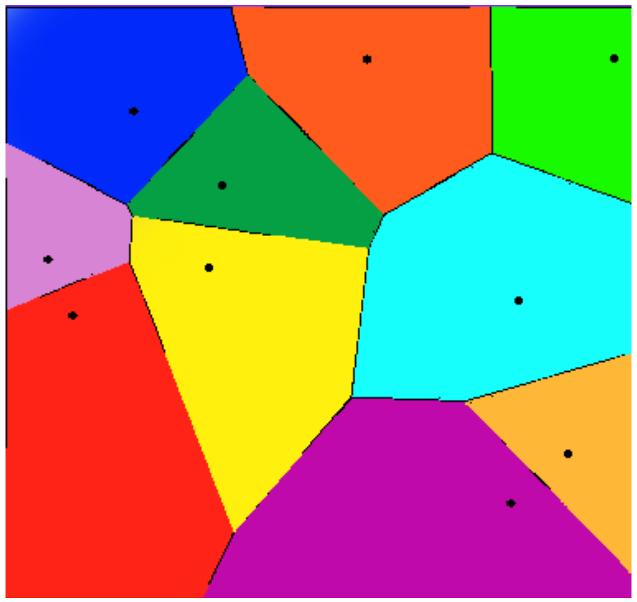
Voronoi diagram with respect to a collection of points $\mathbf{x}_1,\ldots,\mathbf{x}_N$:



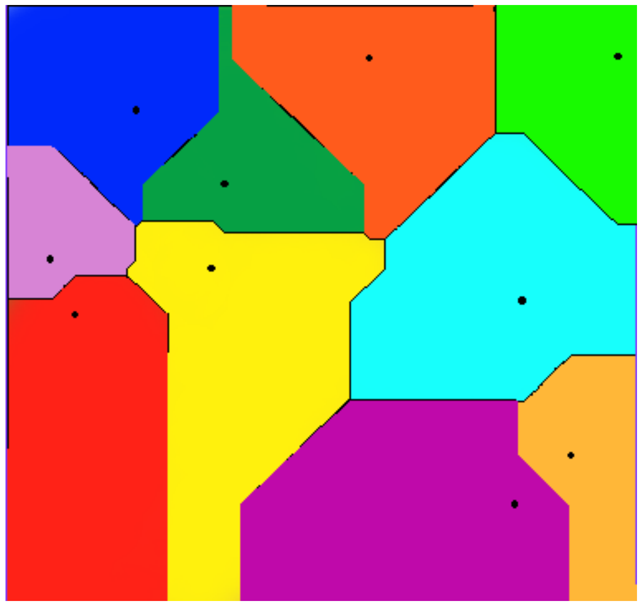
The Voronoi cell associated with point $\mathbf{x}_i$ is the set of points that are closer to $\mathbf{x}_i$ than every other point in the collection

# Voronoi diagrams

The Voronoi diagram depends on the distance measure that is used:



Voronoi diagram computed from Euclidean distance (L2 norm)



Voronoi diagram computed from Manhattan distance (L1 norm)

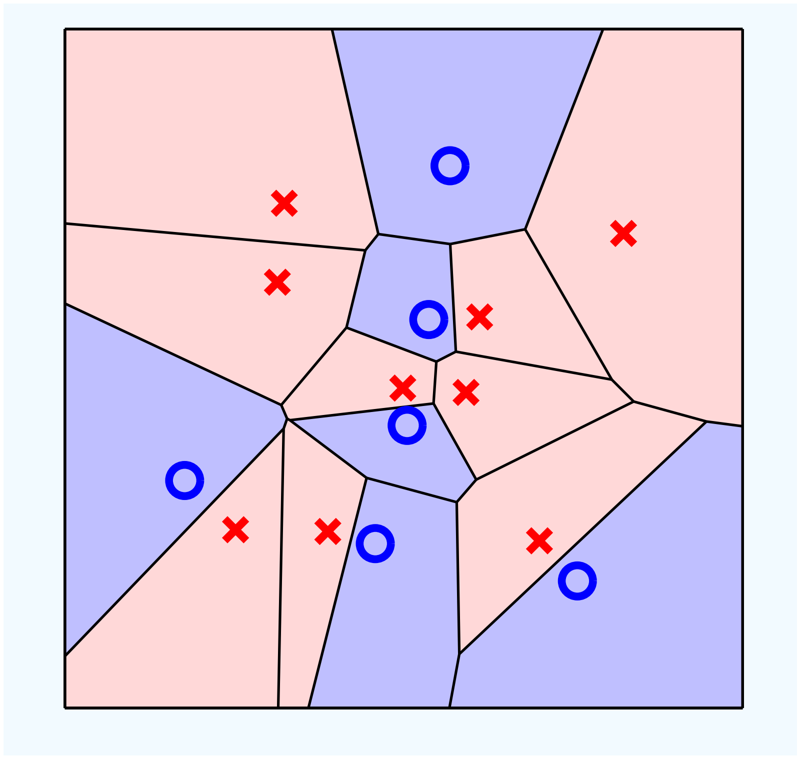
Image from http://en.wikipedia.org/wiki/Voronoi_diagram

# Nearest neighbor classification

The nearest neighbor classifier:

Classify a given test example to the class of the nearest training example.

Decision boundary is the result of fusing adjacent Voronoi cells that are associated with the same class.



# Nearest neighbor classification

The nearest neighbor classifier:

Classify a given test example to the class of the nearest training example.

What is the accuracy of the nearest neighbor classifier when it is tested on the training set? (i.e., what is $E_{in}$)

# Nearest neighbor classification

The nearest neighbor classifier:

Classify a given test example to the class of the nearest training example.

Property of the nearest neighbor classifier:

$E_{out} \leq 2E^*_{out}$ where $E^*_{out}$ is the error of an optimal classifier

More precisely:

Theorem: For any $\delta > 0$ there is a sufficiently large N such that with probability $> 1 - \delta$ the resulting nearest neighbor classifier has $E_{out} \leq 2E^*_{out}$.

## k-NN

Use the closest k neighbors to make a decision instead of a single nearest neighbor

Choose the majority label among the k nearest neighbors

Why do you expect this to work better?

## k-NN

Use the closest k neighbors to make a decision instead of a single nearest neighbor

Choose the majority label among the k nearest neighbors

Can produce confidence scores. How?

## k-NN

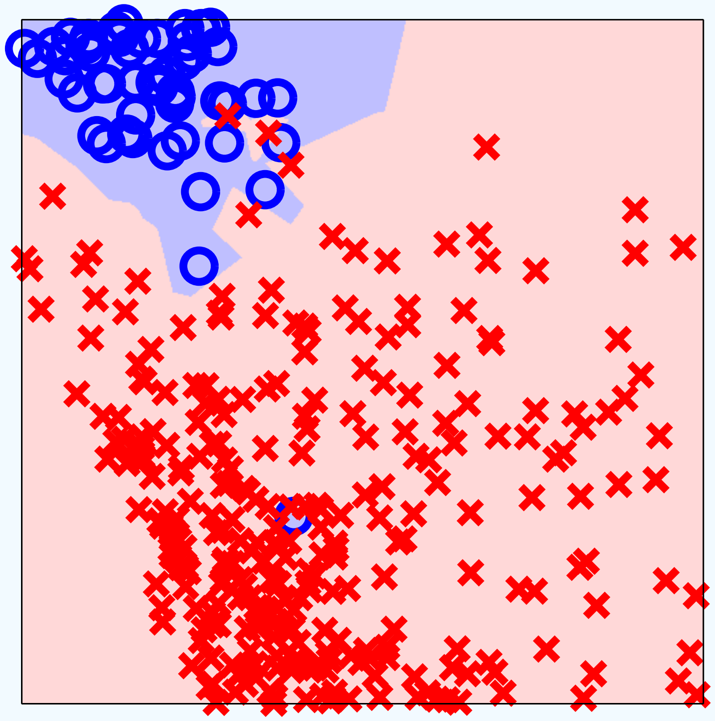
Use the closest k neighbors to make a decision instead of a single nearest neighbor

Choose the majority label among the k nearest neighbors

Other refinements: an example's vote is inversely proportional to its distance

## NN vs k-NN

Decision boundary of NN vs k-NN on the digits data:


(a) 1-NN rule

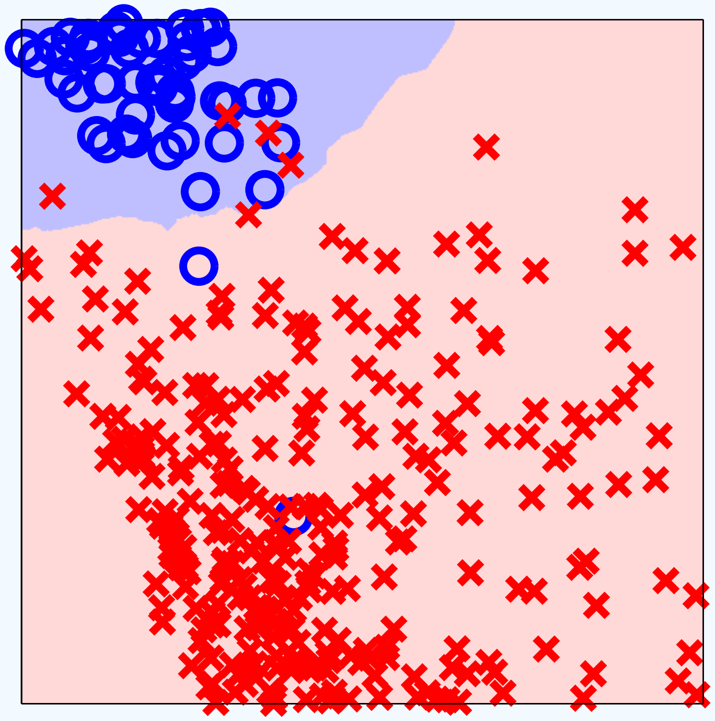

(b) 21-NN rule

Figure 6.2 in chapter e-6

## How to choose k?

The value of k can be chosen using cross-validation, like any classifier parameter:

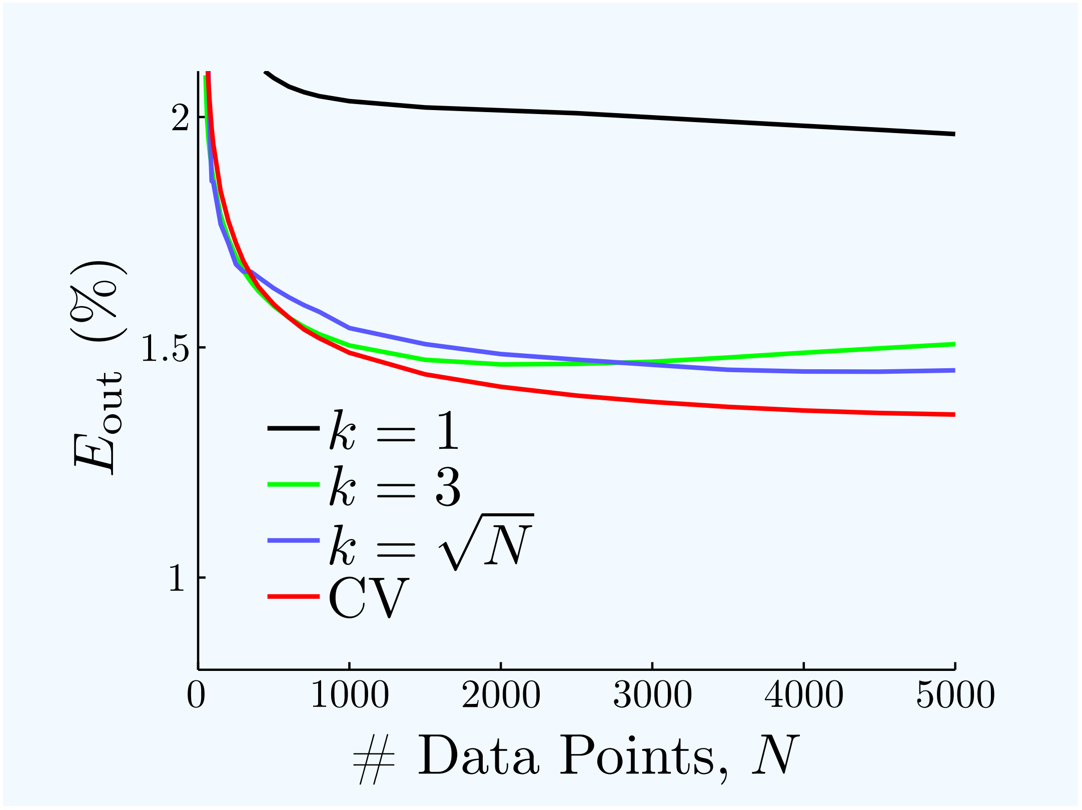


Figure 6.3 in chapter e-6

## Interim conclusions

Properties of nearest neighbor classifiers:

- Simple and easy to implement
- No training required
- Expressive: can achieve zero training error
- Easy to explain the result

But…

- Running time can be an issue
- Not the best in terms of generalization

## Nearest neighbor classification

The nearest neighbor classifier:

Classify a given test example to the class of the nearest training example.

Running time for testing an example when dataset has N examples?

## Nearest neighbor classification

The nearest neighbor classifier:

Classify a given test example to the class of the nearest training example.

Running time for testing an example when dataset has N examples?

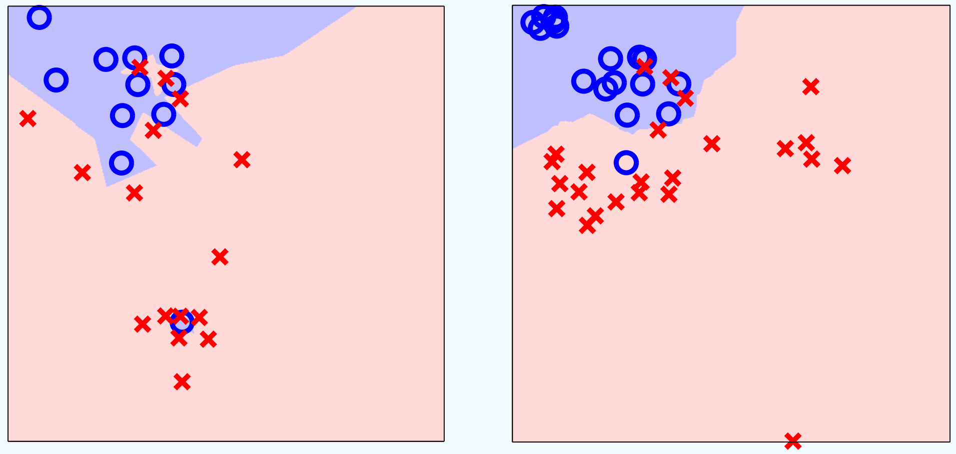
O(N). Expensive when dealing with large datasets.

# Nearest neighbor classification

Running time for testing an example when dataset has N examples is O(N).

Solutions:

"Condense" the dataset


(a) Condensed data for 1-NN (b) Condensed data for 21-NN

Efficient nearest neighbor search
(KD-trees, ball-tress, vantage-point trees)

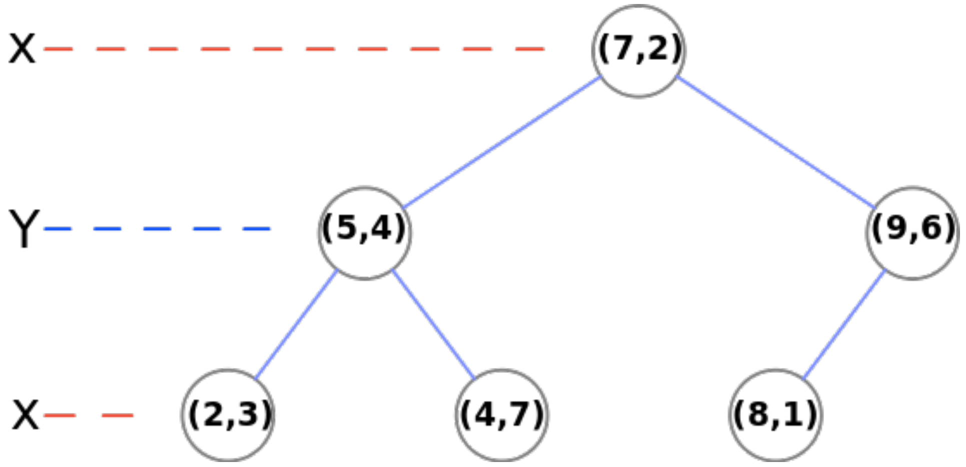
# KD-tree

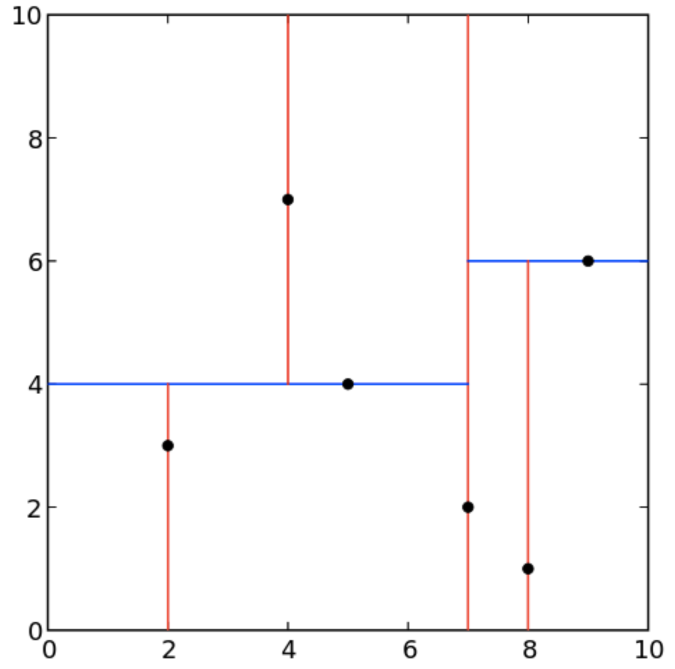
Algorithm:

- Cycle through the coordinates
- Insert a node that corresponds to the median of the given coordinate, and put all other points in the left/right subtree on the basis of that coordinate

A KD-tree for the set of points

(2,3), (5,4), (9,6), (4,7), (8,1), (7,2).





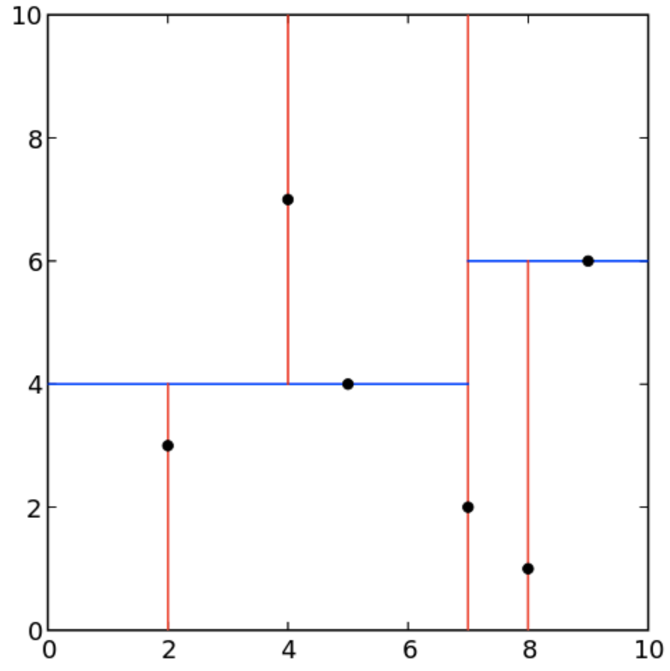
# KD-tree

Algorithm:

- Cycle through the coordinates
- Insert a node that corresponds to the median of the given coordinate, and put all other points in the left/right subtree on the basis of that coordinate

A KD-tree for the set of points

(2,3), (5,4), (9,6), (4,7), (8,1), (7,2).



Can be used for implementing nearest neighbor search in O(log N)

Not effective for high dimensional data (use ball-tree or vantage-point tree)

## Nearest neighbor classification in high dimensions

Distance functions lose their usefulness in high dimensions.

Consider the Euclidean distance for example:

$$d_2(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_{i=1}^{d} (x_i - x'_i)^2}$$

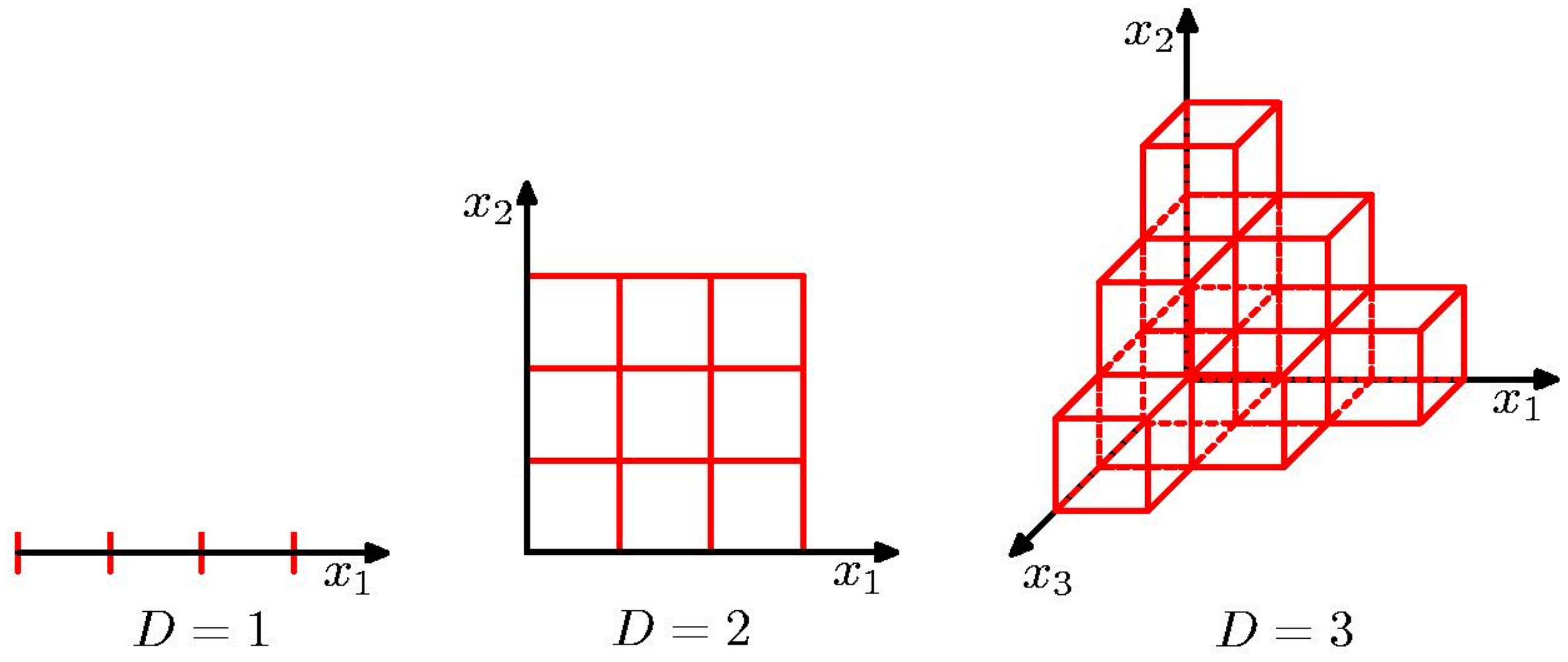
We expect that if d is large, many of the features won't be relevant, and so the signal contained in the informative dimensions can easily be corrupted by the noise.

This can lead to low accuracy of a nearest neighbor classifier.

Solution: feature selection, dimensionality reduction

# The curse of dimensionality

An umbrella term for the issues that can arise in high dimensional data.



## The curse of dimensionality

Some of our intuition from low dimensional spaces breaks in high dimensions.

Example: In high dimensions, most of the volume of the unit sphere is very close to its surface.

Let's compute the fraction of the volume that is between r=1-ε and r=1.

$$V_d(r) = k_d \cdot r^d$$

The required fraction is:

$$\frac{V_d(1) - V_d(1-\epsilon)}{V_d(1)} = 1 - (1-\epsilon)^d$$

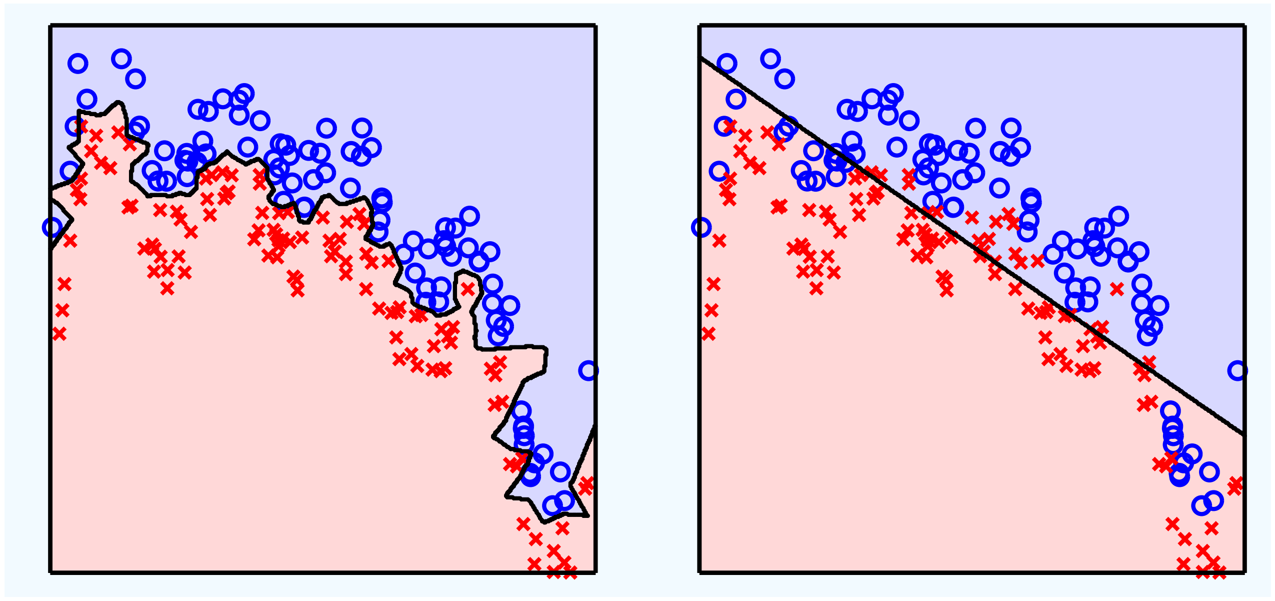
Related fact:

The ratio of the volume of the unit sphere and unit cube tends to 0 as d goes to infinity.

# Non-parametric vs parametric methods

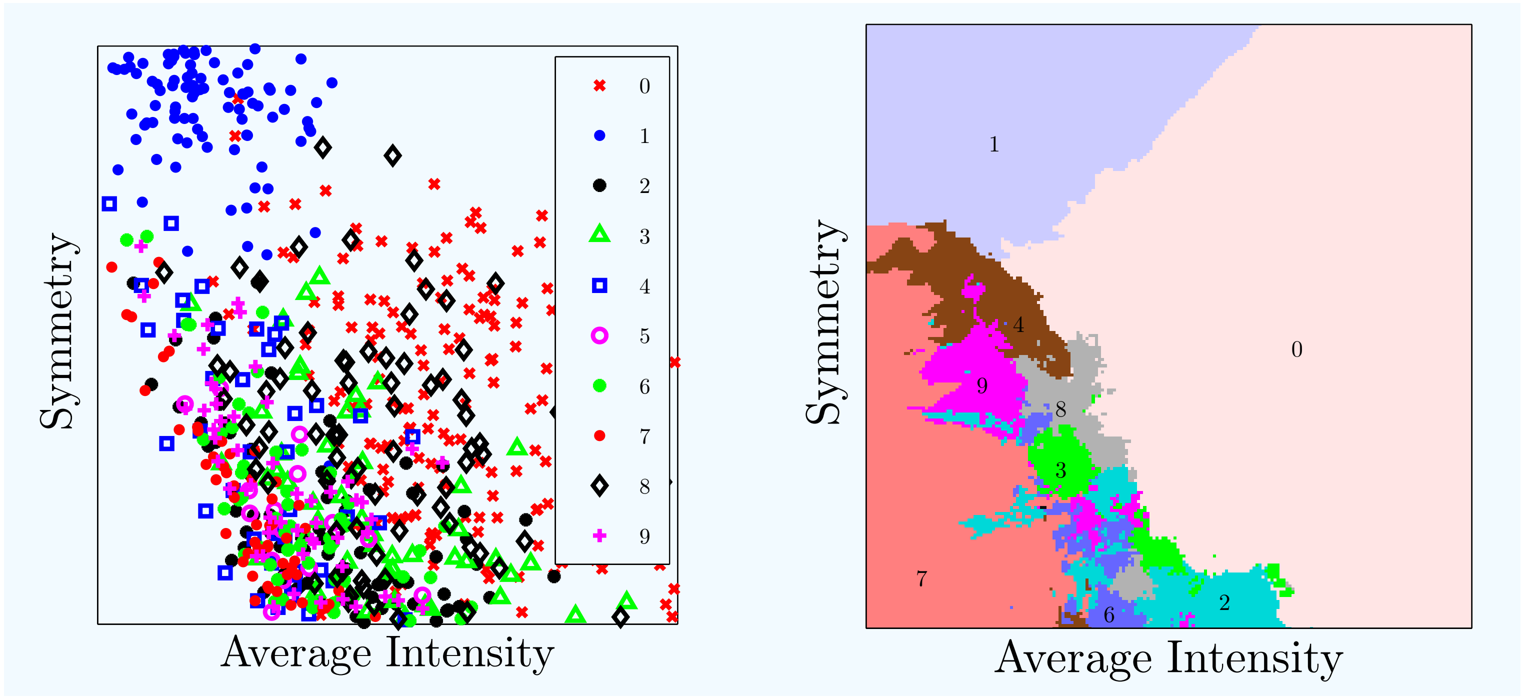
Non-parametric methods don't have any parameters that are learned from the data.

Parametric methods have a specific form that the learned model will have.



# Multi-class problems

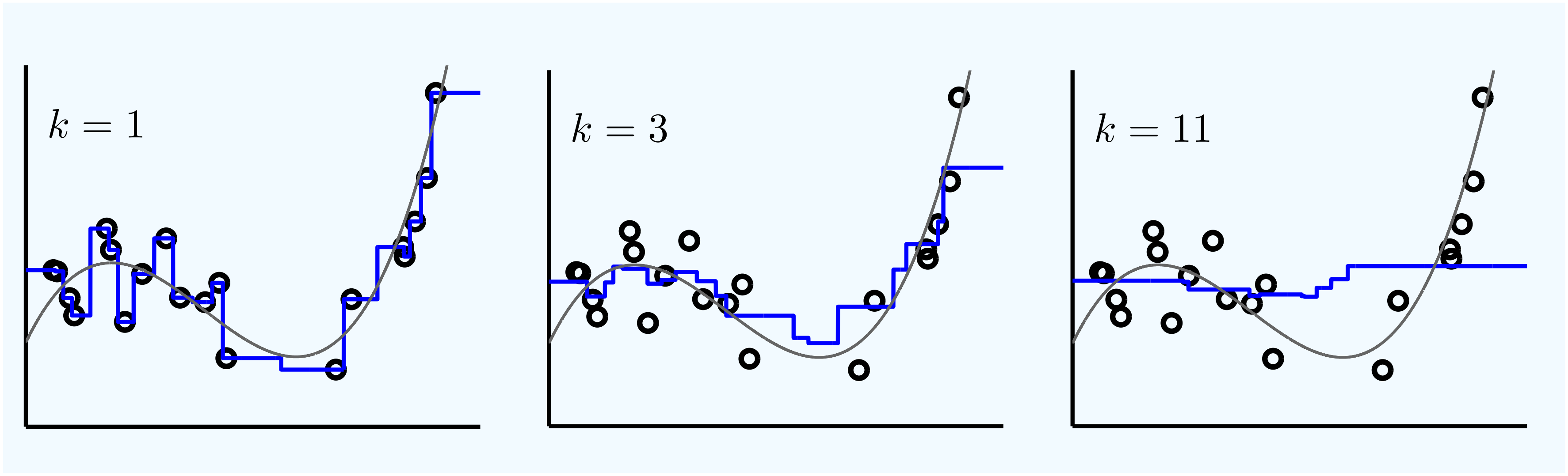
The nearest neighbor algorithm works much the same way for multi-class problems



In fact, nearest neighbor methods are easily adaptable to any ML problem.

# Nearest neighbor regression

How do turn k-NN into a regression method?



## Distances and kernels

The Euclidean distance can be expressed in terms of dot products:

$$Dis_2(\mathbf{x},\mathbf{y}) = ||\mathbf{x}-\mathbf{y}||_2 = \sqrt{(\mathbf{x}-\mathbf{y})^\intercal(\mathbf{x}-\mathbf{y})} = \sqrt{\mathbf{x}^\intercal\mathbf{x} - 2\mathbf{x}^\intercal\mathbf{y} - \mathbf{y}^\intercal\mathbf{y}}$$

Replacing dot products with kernels:

$$Dis_K(\mathbf{x},\mathbf{y}) = \sqrt{K(\mathbf{x},\mathbf{x}) - 2K(\mathbf{x},\mathbf{y}) - K(\mathbf{y},\mathbf{y})}$$

As an alternative, consider kernels as measures of similarity, and rather than looking for the closest points, look for the most similar points, and use kernels directly.

## Distances and kernels

Replacing dot products with kernels:

$$Dis_K(\mathbf{x},\mathbf{y}) = \sqrt{K(\mathbf{x},\mathbf{x}) - 2K(\mathbf{x},\mathbf{y}) - K(\mathbf{y},\mathbf{y})}$$

If we consider a kernel that satisfies K(x,x) = 1, then nearest neighbor classification with kernels or distances is equivalent.

## Summary

Nearest neighbor classification:

Pros:

- Simple and easy to implement
- No training involved
- One method that does it all

Cons:

- Accuracy suffers in high dimensions
- Testing speed is an issue for large datasets