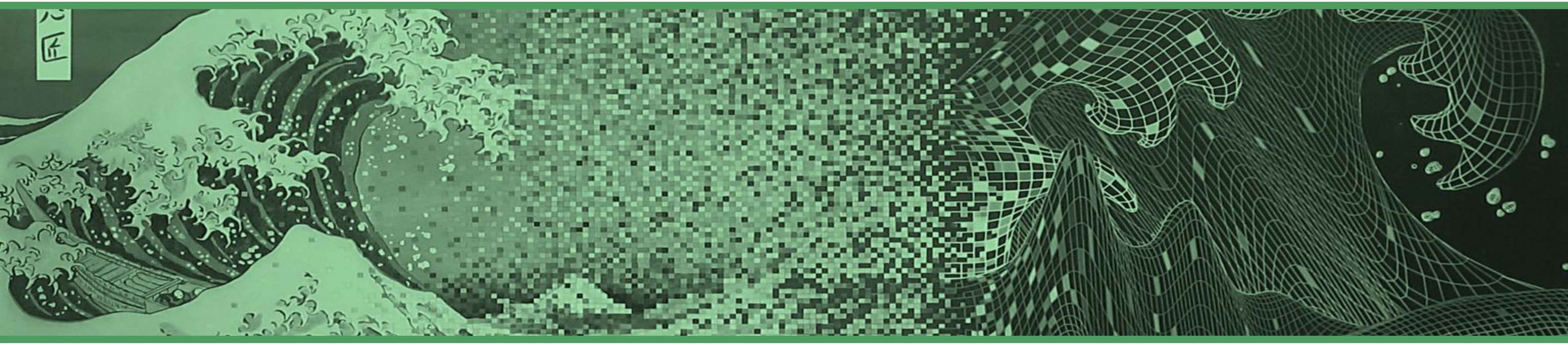
TÉCNICO LISBOA

Lazy learning

Local learning in the vector space



Aprendizagem 2023

Rui Henriques rmch@tecnico.ulisboa.pt
Andreas Wischert andreas.wichert@tecnico.ulisboa.pt

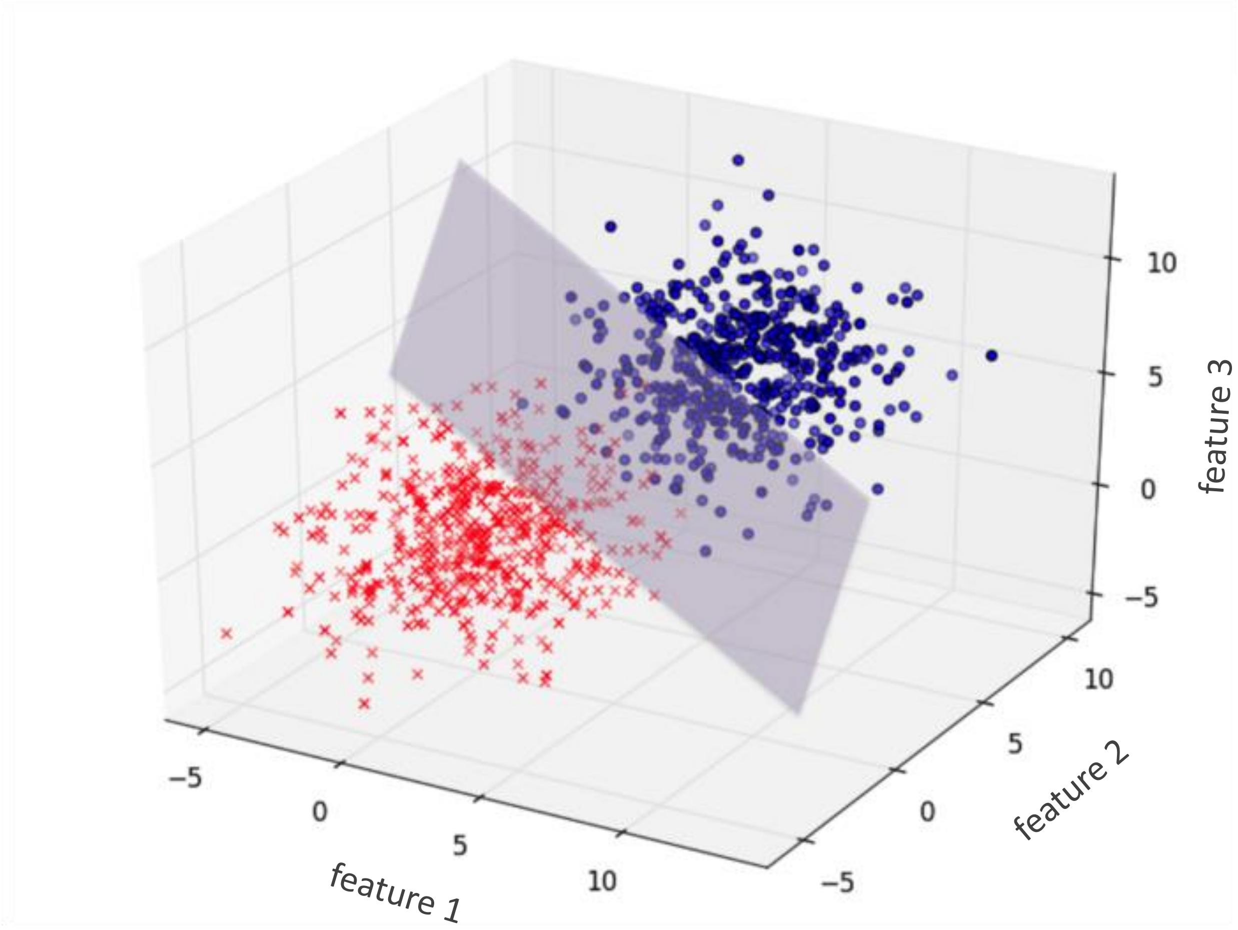
Outline

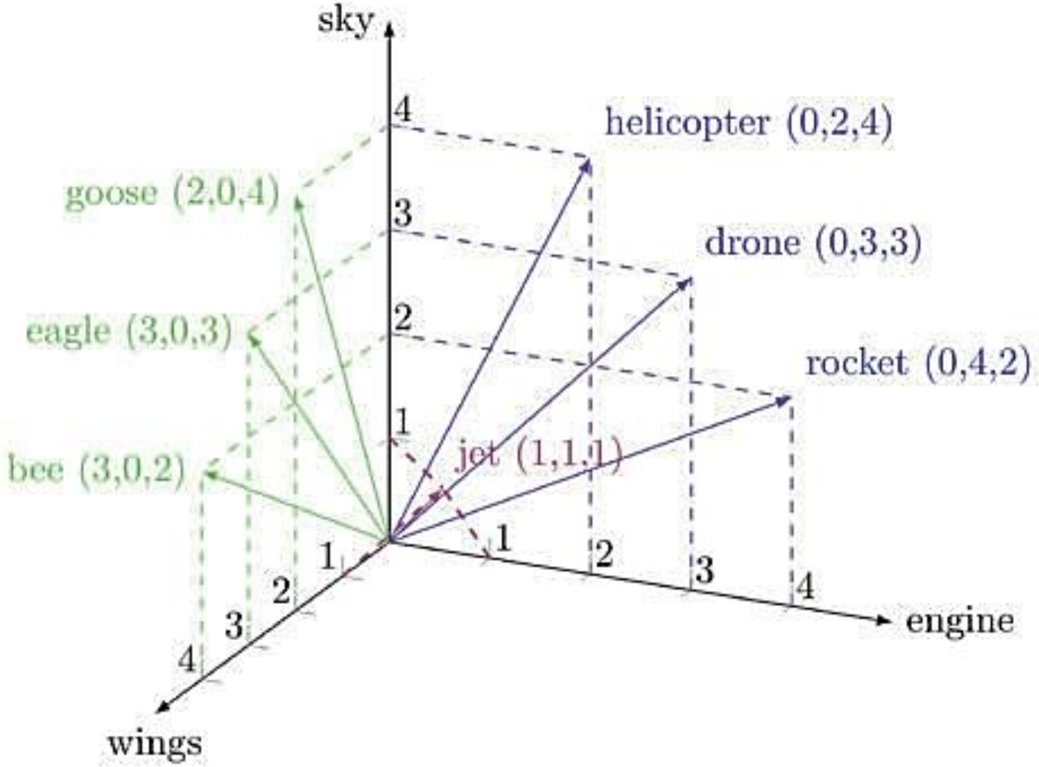
- **Learning in the vector space**
  - from univariate to multivariate data spaces
  - Minkowski distances
  - norm and cosine similarity
  - encodings for categoric variables and mixed data
- **Instance-based learning**
  - parametric vs non-parametric models
  - k-nearest neighbor classifier
  - k-nearest neighbor regressor
  - distance-based kNN
  - challenges: non-iid variables and dimensionality
  - learning vector quantization

Outline

- **Learning in the vector space**
  - **from univariate to multivariate data spaces**
  - **Minkowski distances**
  - **norm and cosine similarity**
  - **encodings for categoric variables and mixed data**
- Instance-based learning
  - parametric vs non-parametric models
  - k-nearest neighbor classifier
  - k-nearest neighbor regressor
  - distance-based kNN
  - challenges: non-iid variables and dimensionality
  - learning vector quantization

Multivariate feature space



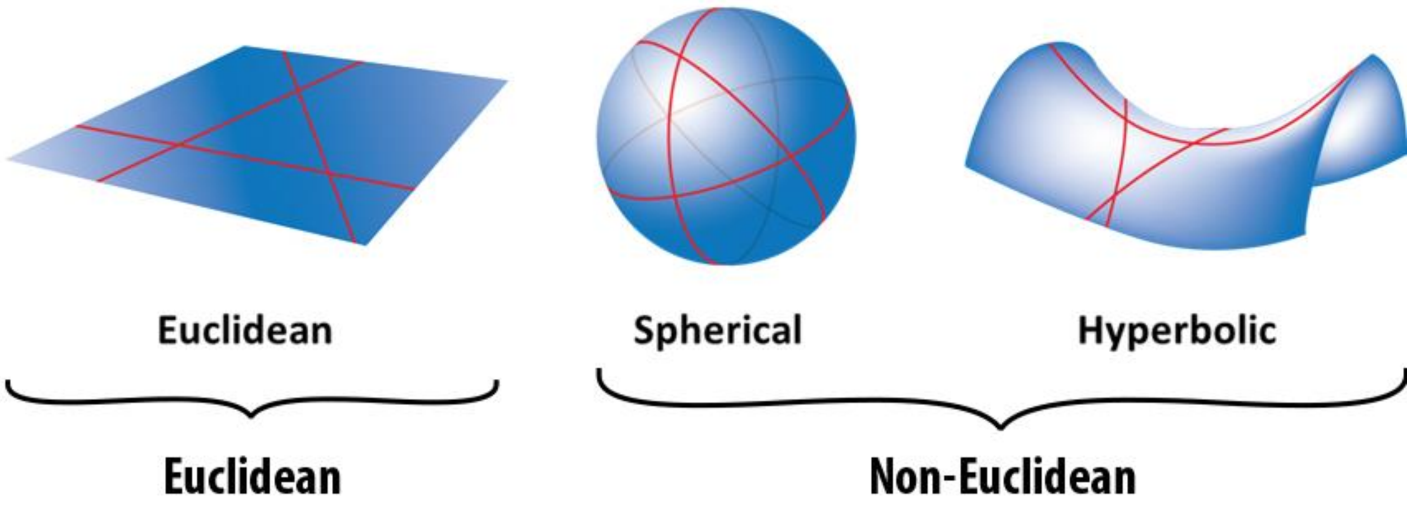


Feature space

- Recall the principles behind the feature space
  - a **feature** is an individual measurable property or characteristic of a phenomenon
    - phenomena is described by a random variable
  - features are usually **numeric** or **categorical**
    - yet **structural features** such as images, videos, strings, time series or graphs can be as well observed
  - recall that a **multivariate observation** of order m is a data instance with m features
  - the set of possible features forms a m-**dimensional feature space** (or m-order multivariate space)
  - when features are numeric:
    - the m-dimensional feature space is a $\mathbb{R}^m$ Euclidean space
    - each observation can be conveniently described by a **point** or **vector** in this $\mathbb{R}^m$ space
  - we will see how can we bring categorical features later on

Feature space

- Bivariate data spaces
  - vector properties and distances are generally assessed on a Euclidean plane
  - alternative continuous spaces yield properties of interest



- Question: consider a single variable that measures the orientation of the wind from 0º to 360º
  - is the univariate Euclidean space that best option for this univariate data space?

Distances and metrics

- A distance function is a **metric** if the following conditions are met:
  - non-negative:

$$d(\mathbf{x}, \mathbf{y}) \geq 0$$

  - distance to point itself is zero:

$$d(\mathbf{x}, \mathbf{x}) = 0$$

  - symmetry:

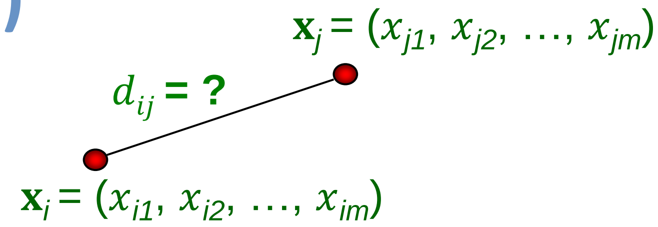
$$d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$$

  - triangular inequality:

$$d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{y})$$

Common distance metrics

(numeric data)



Minkowski distance

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt[q]{\underbrace{|x_{i1} - x_{j1}|}_{\text{1st dimension}}^q + \underbrace{|x_{i2} - x_{j2}|}_{\text{2nd dimension}}^q + \ldots + \underbrace{|x_{im} - x_{jm}|}_{\text{pth dimension}}^q}$$

Euclidean distance

q = 2

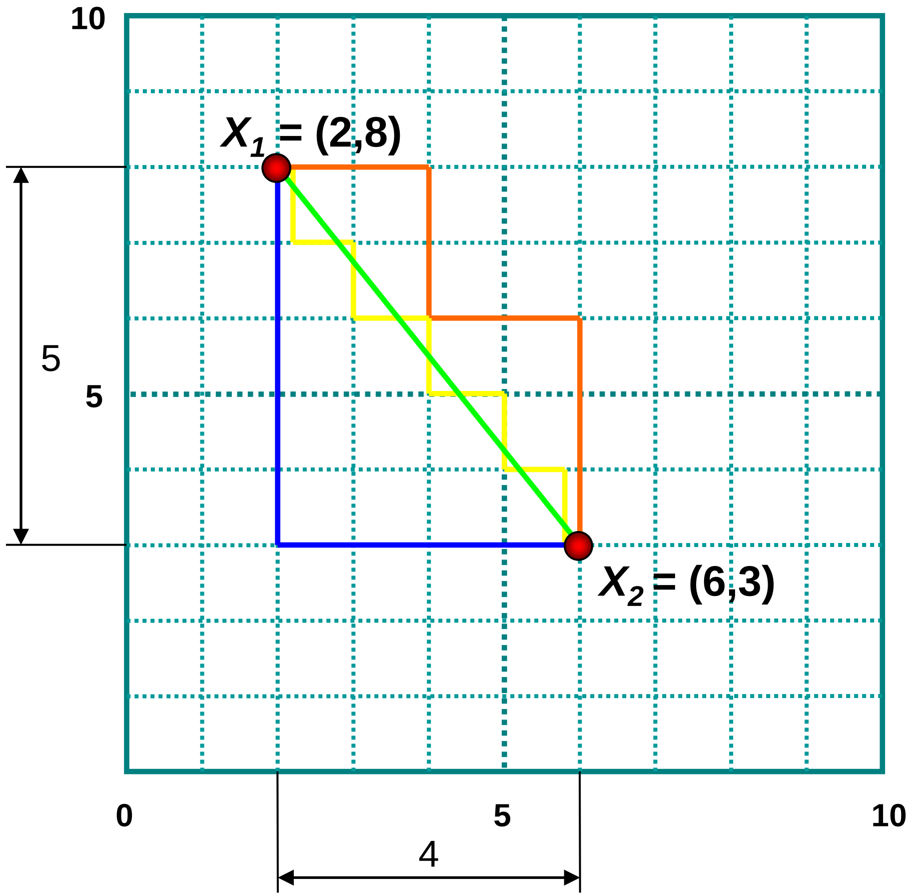
$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \ldots + |x_{im} - x_{jm}|^2}$$

Manhattan distance

q = 1

$$d(\mathbf{x}_i, \mathbf{x}_j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \ldots + |x_{im} - x_{jm}|$$

Common distance metrics
(numeric data)



2D example

$\mathbf{x}_1 = (2,8)$

$\mathbf{x}_2 = (6,3)$

Euclidean distance

$$d(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{|2-6|^2 + |8-3|^2} = \sqrt{41}$$

Manhattan distance

$$d(\mathbf{x}_1, \mathbf{x}_2) = |2-6| + |8-3| = 9$$

Chebyshev distance
(numeric data)

- In case of $q \rightarrow \infty$, the metric selects the error associated with the highest dissimilar feature
- Useful if the worst case must be avoided:

$$d_\infty(\mathbf{x}, \mathbf{y}) = \lim_{q \to \infty} \left(\sum_{i=1}^{n} |x_i - y_i|^q \right)^{1/q}$$

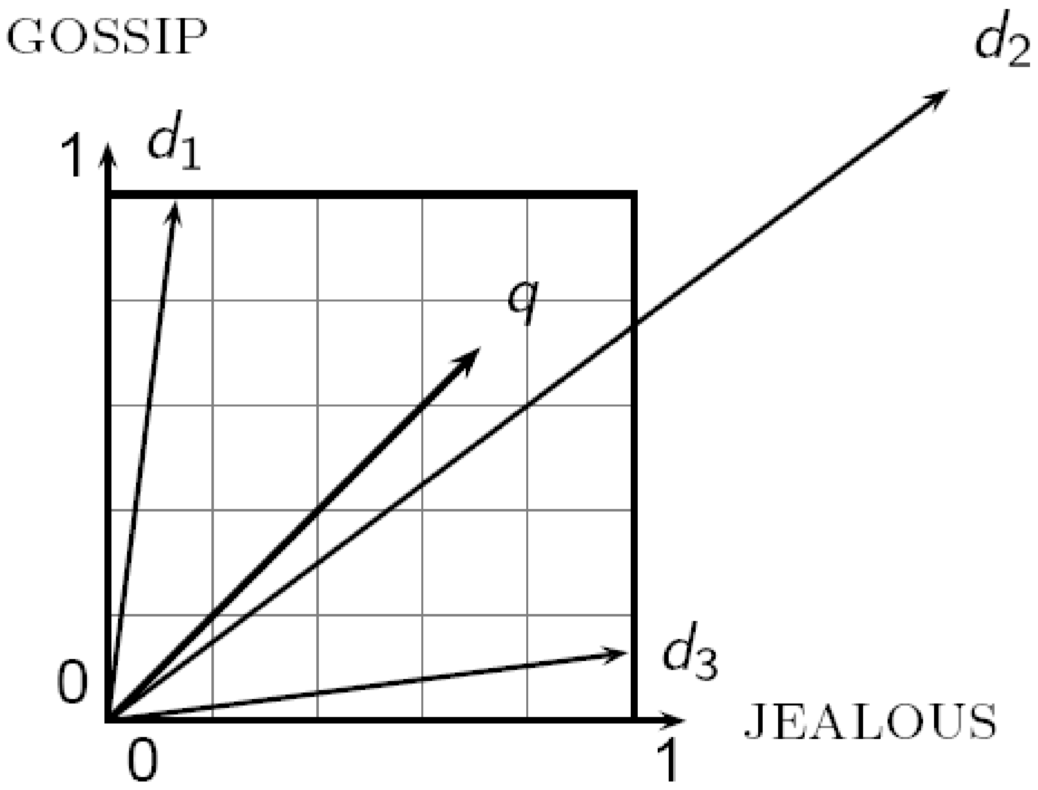
$$= \max(|x_1 - y_1|, |x_2 - y_2|, \dots, |x_n - y_n|)$$

Example:

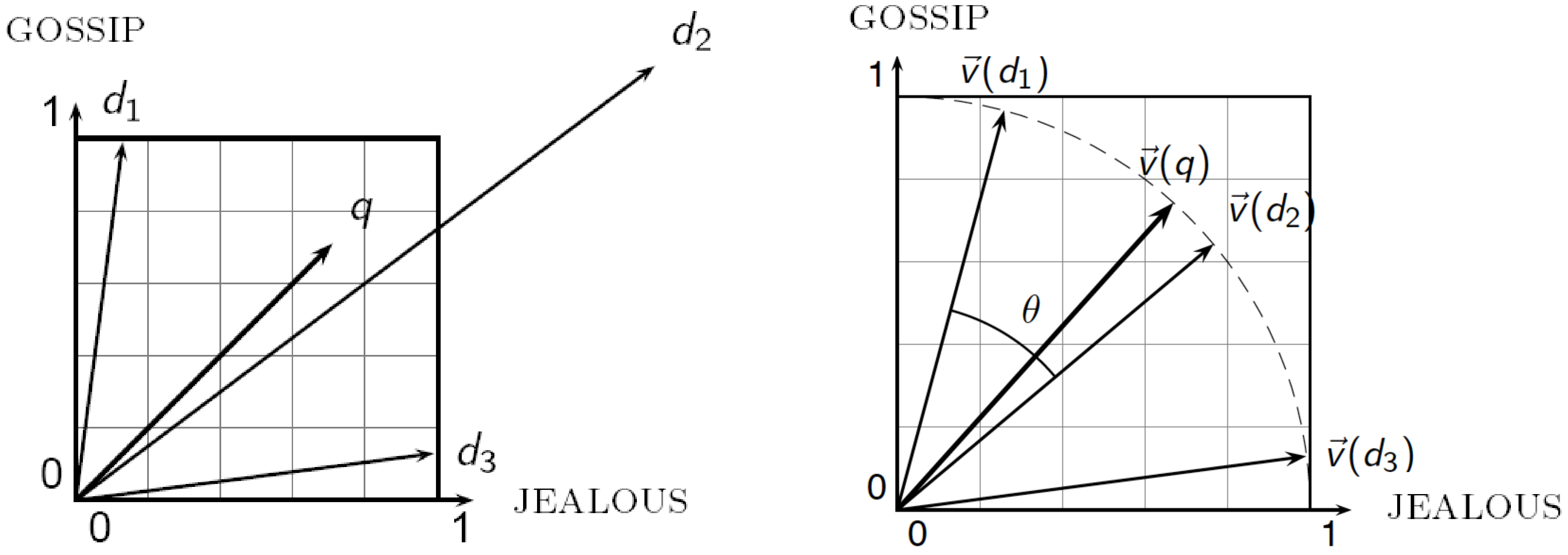
$$d_\infty((2,8), (6,3)) = \max(|2 - 6|, |8 - 3|) = \max(4,5) = 5$$

Normalizing vectors?

- Consider the scenario where you have three documents (observations): $\mathbf{d}_1$, $\mathbf{d}_2$ and $\mathbf{d}_3$
  - each document is characterized by the frequency of terms in the text (features)
- Now consider we enter a query $\mathbf{q}$ to retrieve documents similar to $\mathbf{q}$
- Euclidean distance is… a bad idea as it is large for vectors of different lengths
  - the Euclidean distance between $\mathbf{q}$ and $\mathbf{d}_2$ is large even though the distribution of terms in the query $\mathbf{q}$ and $\mathbf{d}_2$ are very similar



Vector norm



- the normalization of a multivariate observation, should not be misled with *variable normalization*
  - variable normalization guarantees that the measurements of a given variable (e.g. virus concentration) yield statistical properties of interest, e.g. [0,1] or $N(0,1)$

Vector norm

- Given a vector space $\mathbb{R}^m$, the norm is a function that maps a vector $\mathbf{x} \in \mathbb{R}^m$ into a real number, such that:

$$||\mathbf{x}|| \geq 0$$

$$||\alpha \cdot \mathbf{x}|| = |\alpha| \cdot ||\mathbf{x}|| \text{ where } \alpha \text{ is a scalar}$$

$$||\mathbf{x} + \mathbf{y}|| \leq ||\mathbf{x}|| + ||\mathbf{y}||$$

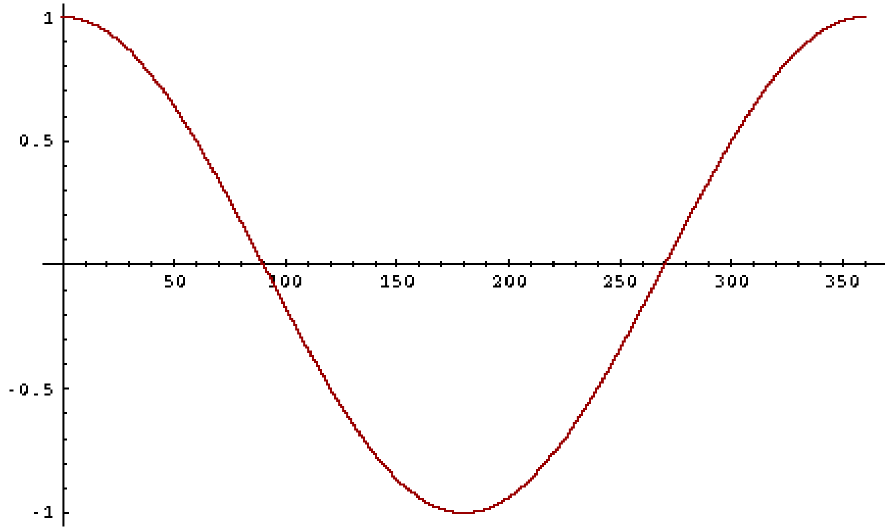
- The l_p norm (or p-norm),

$$||\mathbf{x}||_p = \left(\sum_{i=1}^{m} {x_i}^p \right)^{1/p}$$

  - $p = 2$ is the Euclidean norm and $||\mathbf{x}||_2 = 1$
  - $p = 1$ is the absolute (Manhattan) norm
- To normalize a vector, we divide its values by the vector norm, i.e. $normalize(\mathbf{x}) = \frac{\mathbf{x}}{||\mathbf{x}||}$

From angles to cosines

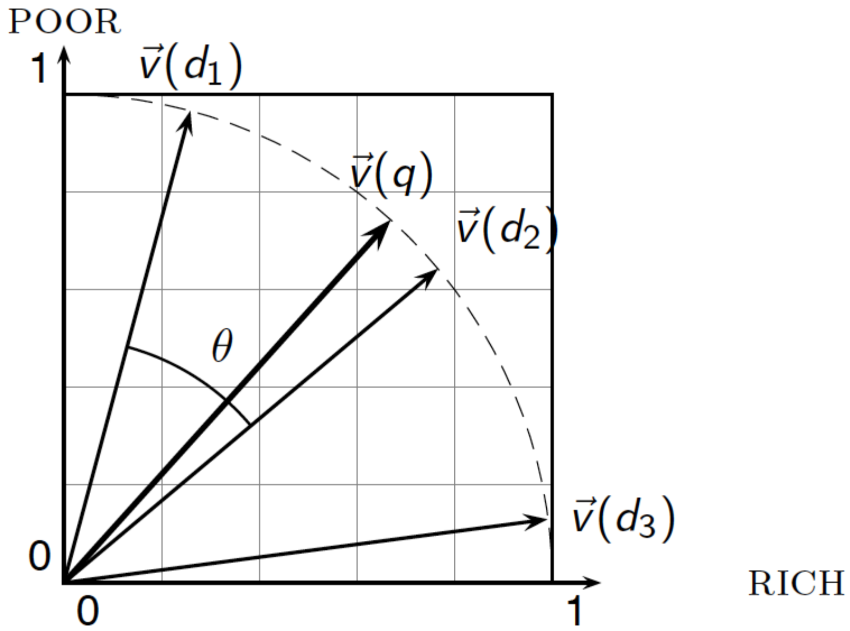
- the following two notions are equivalent:
  - *distance* of the angle between two vectors
  - *similarity* of the cosine between two vectors
- cosine is a monotonically decreasing function for the interval [0°, 180°]



Cosine similarity

$$\cos(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_1 \cdot \mathbf{x}_2}{\|\mathbf{x}_1\| \|\mathbf{x}_2\|} = \frac{\sum_i x_{1i}\, x_{2i}}{\sqrt{\sum_i {x_{1i}}^2}\sqrt{\sum_i {x_{2i}}^2}}$$

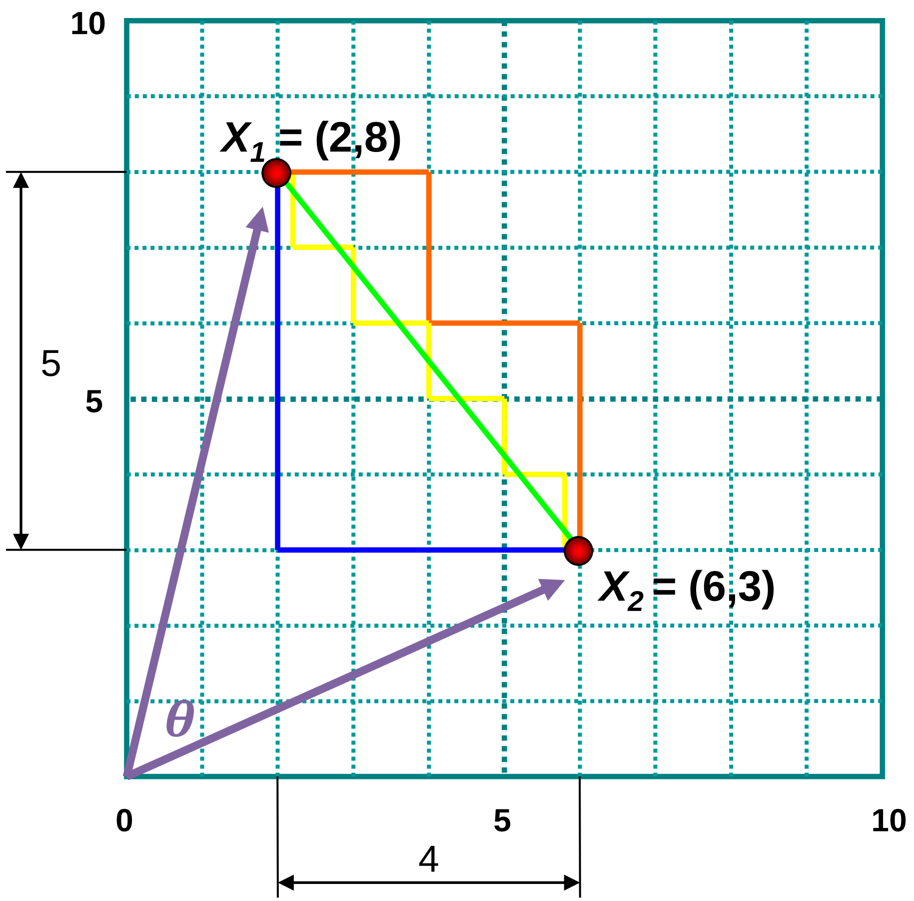
- x_{1i} is the feature from variable y_i in observation $\mathbf{x}_1$
- $\|\mathbf{x}_1\|$ and $\|\mathbf{x}_2\|$ are the (2-norm) lengths of vectors $\mathbf{x}_1$ and $\mathbf{x}_2$
- $\cos(\mathbf{x}_1, \mathbf{x}_2)$ is the (cosine) similarity of $\mathbf{x}_1$ and $\mathbf{x}_2$



For length-normalized vectors, cosine similarity is simply the dot product (or scalar product):

$$\cos(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1 \cdot \mathbf{x}_2 = \sum_{i=1}^{m} x_{1i}\, x_{2i}$$

Cosine similarity



Euclidean distance

$$l_2(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{|2-6|^2 + |8-3|^2} = \sqrt{41}$$

Manhattan distance

$$l_1(\mathbf{x}_1, \mathbf{x}_2) = |2-6| + |8-3| = 9$$

Cosine similarity

$$\cos(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_1 \cdot \mathbf{x}_2}{\|\mathbf{x}_1\| \|\mathbf{x}_2\|} = \frac{2\times6+8\times3}{\sqrt{2^2+8^2}\sqrt{6^2+3^2}} = 0.65$$

Correlation

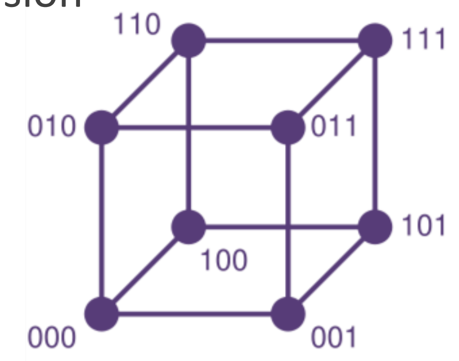
- Assessing similarity between observations can recur to different functions:
  - distance functions
  - cosine angle
  - correlation coefficients
- Recall correlation basics
  - positive (negative): two variables vary in the same (opposite) way
  - maximum value of 1 (-1) if $\mathbf{x}_1$ and $\mathbf{x}_2$ are perfectly direct (inverse) correlated
- Example: gene expression arrays

$$\mathbf{x}_1 = (1,2,3,4,5)$$
$$\mathbf{x}_2 = (100,200,300,400,500)$$
$$\mathbf{x}_3 = (5,4,3,2,1)$$

Which genes are similar according to Euclidean and Pearson correlation?

Categorical features

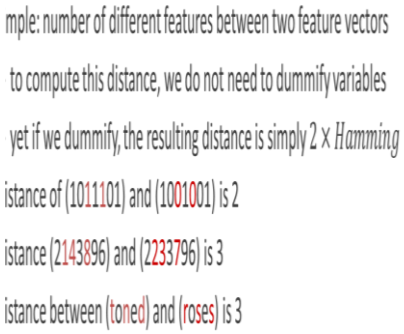
- How do we represent categorical features in a high-dimensional space?
  - Let us start with **binary variables** (e.g. gender)
    - a binary feature can be seen as an axis in a m-dimensional where values are constrained to 0 or 1
  - Let us now go to **ordinal variables** (e.g. high-moderate-low)
    - we can find a numeric encoding f for an ordinal variable
      - e.g. $f(high) = 5, f(moderate) = 1, f(low) = 0$
    - in the absence of an encoding, one can assume equally spaced integers (e.g. 0, 1, 2…)
      - problems?
  - Let us finally consider **nominal variables** with cardinality higher than 2 (e.g. Europe, Africa, America)
    - challenge? As there is no order, we cannot position values along a single axis/dimension
    - solution? Create p axes for a nominal variable with cardinality p
      - only one of the p axes is active (1), while remaining are inactive (0)
      - this operation is called **dummification**
      - some machine learning methods depend on dummification



Hamming distance
(binary and categorical data)

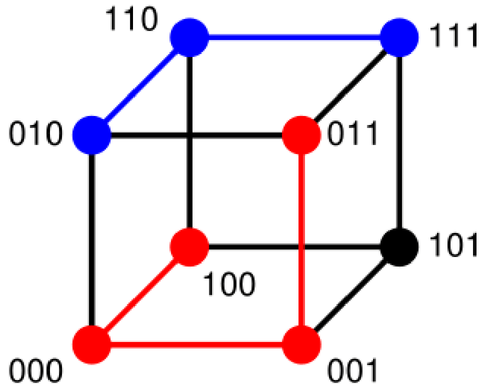
- Simple: number of different features between two feature vectors
  - to compute this distance, we do not need to dummify variables
  - yet if we dummify, the resulting distance is simply $2 \times Hamming$
- Distance of (1011101) and (1001001) is 2
- Distance (2143896) and (2233796) is 3
- Distance between (toned) and (roses) is 3

3-bit binary cube



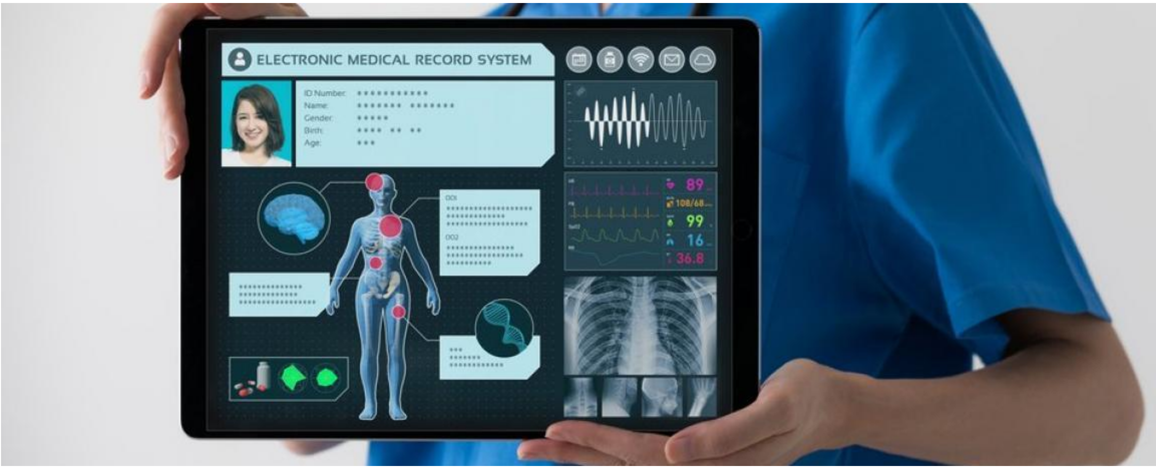
100->011 has distance 3 (red path)

010->111 has distance 2 (blue path)



Mixed data

- A data space can be composed by **non-iid numeric variables**
  - iid stands for **i**ndependent and **i**dentically **d**istributed
  - two numeric variables may be correlated, i.e. not independent
  - two numeric variables may have distinct distributions (e.g. $U(0,100)$ and $N(3,5)$), i.e. non-identically distributed
- A dataset can be composed of numeric variables (continuous and discrete), nominal and ordinal variables
  - we informally term such data as **mixed data**
  - examples?



Distances for mixed data

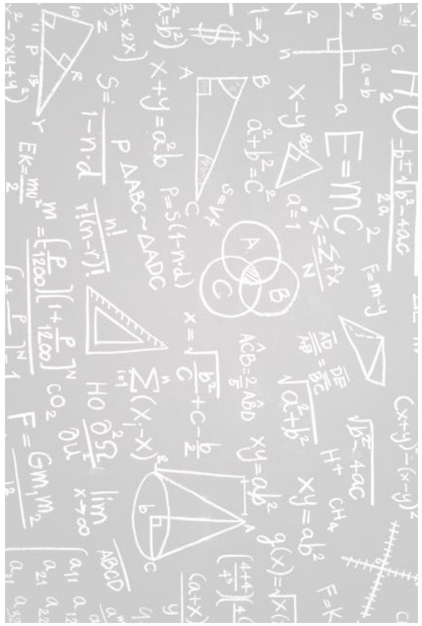
- First concern: two non-identically distributed variables – e.g. $Y_1 \sim U(0,1)$ and $Y_2 \sim U(0,100)$
  - *Problem*? In the given example, distances are most affected by variable Y_2
    - yet why should this variable have higher weight than Y_1!
  - *Solution*? Normalize variable features
    - do not confuse variable normalization with the normalization of a data instance/observation!
    - in the given example, $Y_2 \sim U'(0,1)$ by applying for instance min-max scaling

- Second concern: how to deal with simultaneous categoric and numeric variables?
  - distance can be a composition: $d(\mathbf{x}_1, \mathbf{x}_2) = \alpha\, d_{numeric}(\mathbf{x}_1, \mathbf{x}_2) + \beta\, d_{categoric}(\mathbf{x}_1, \mathbf{x}_2)$
    - where α and β are parameters that reveal the importance of each component, generally $\alpha + \beta = 1$
      - parameters can be fixed based on the number of variables or based on available domain knowledge

Algebra theory

- We had a first glimpse on few algebraic foundations on high-dimensional spaces
  – vector representations and distances in distinct spaces

- More to come in the upcoming lectures…
  – *multivariate Bayesian* and *neural network learning*
    – high-order **matrix operations**: product, determinant, inverse
  – *kernel methods*
    – high-order **data space transformations**
  – *dimensionality reduction*
    – orthogonal projections (eigenvectors and eigenvalues)



Outline

- Learning in the vector space
  - from univariate to multivariate data spaces
  - Minkowski distances
  - norm and cosine similarity
  - encodings for categoric variables and mixed data
- **Instance-based learning**
  - **parametric vs non-parametric models**
  - **k-nearest neighbor classifier**
  - **k-nearest neighbor regressor**
  - **distance-based kNN**
  - **challenges: non-iid variables and dimensionality**
  - **learning vector quantization**

Parametric models

- We select a hypothesis and learn a fixed set of parameters
  - example: linear regression to describe a real-valued feature changing along time
    - parameters? two: slope and intercept
    - *predictive modeling:* what are the expected values for upcoming points in time?
    - *descriptive modeling*: how the values change along time?
- We assume that the parameters $\Phi = \{\phi_1, \dots, \phi_K\}$ summarise the (training) data
  - compression

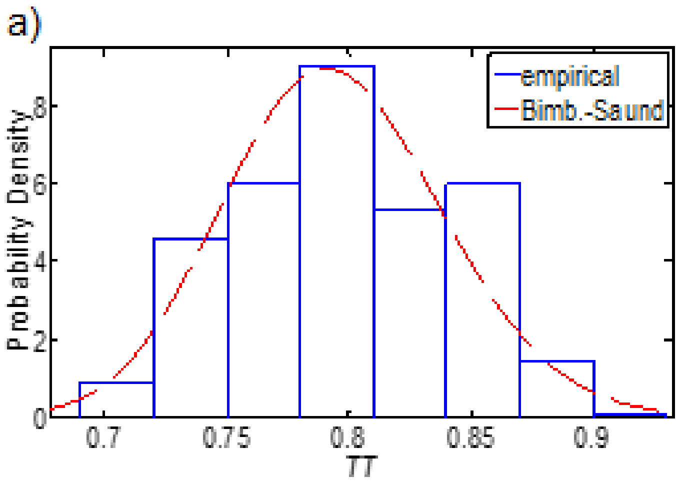
- These methods are called **parametric models**
  - examples: Bayesian models, neural networks (to come!)
  - when we have a small amount of data: models are simpler, need less parameters
  - when we have a large quantity of data: models can be more complex

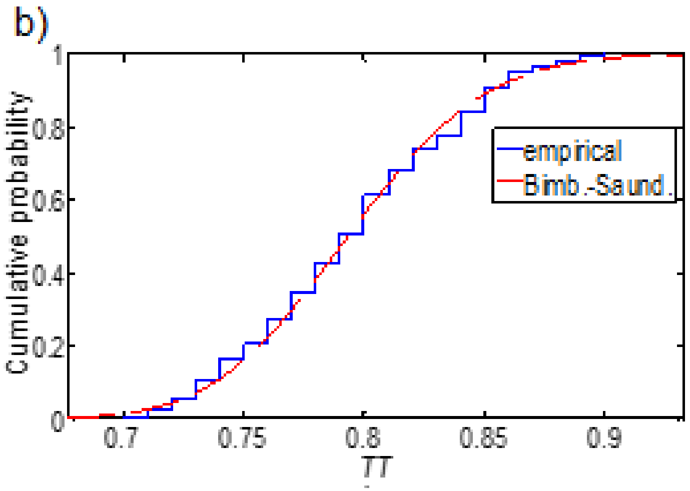
Non parametric Learning

- A **non-parametric model** is one that is not characterised by a well-defined set of parameters
- A family of non-parametric models is **instance-based learning**
  - instance-based learning is based on data memorisation
  - there is not a model associated with the learned concepts
    - predictive and descriptive tasks are obtained by looking into the memorised examples
    - … and inspecting the set of similar-related instances in memory
- Yet.. to assess similarity we need to have a (dis)similarity measure! Isn't this a paremeter?
  - ***parameters*** in classic stances are learned to describe the predictive/descriptive function
  - similarity criterion is not something that is learned but an *a priori* decision on the target function
    - ***hyperparameters*** are more appropriate term

(Non-)parametric univariate modeling

- Theoretical probability density/mass functions are parametric
  - e.g. univariate normal and uniform distributions have 2 parameters, Poisson has 1 parameter
- Empirical probability functions are non-parametric
  - observations can be sorted and probabilities indexed based on values or percentiles
  - there is no need to learn parameters





Instance-based learning

- When?
  - **Predictive modeling**, $M: X \rightarrow Z$
    - approximating real-valued ($Z = \mathbb{R}$) or discrete-valued ($Z = \Sigma$) target functions
  - **Descriptive modeling**, e.g. clustering, $M: X \rightarrow \{X_1 .. X_K\}$ with $X_k \subseteq X$

- Learning principles
  - store available (training) data
  - identify the most similar instances
    - in predictive modeling: given new instance, retrieve most similar training observations
    - in clustering: group instances below a similarity threshold

- Let us now focus on *predictive modeling*

Instance-based learning

- **Neighborhood** of an instance: most similar instances
  - which strategies of (dis)similarity have we covered?
- Local approximation of the target function using the *neighborhood* of a given instance
  - instance-based learning also known as **local learning**
- In supervised learning, the training cost is *0*, all the cost is in the computation of the prediction
  - this kind learning is also known as *lazy learning*
- *Claim*: never construct an approximation designed to perform well over the entire instance space! Why?
- Disadvantage:
  - *testing efficiency*
    - pairwise distances to test new instance against all training instances
    - nearly all computation takes place at classification time rather than learning time
  - *memory efficiency*: need to store available training data

k-Nearest Neighbor

- Nearest-neighbor learning is a specific instance-based learning
- In **predictive modeling**, the target of a new observation is estimated in two steps:
    - the *k* nearest observations (neighborhood) are retrieved
    - a estimator is applied over the targets of the training observations in the neighborhood
        - classification (discrete output variable):
            - mode estimator
        - regression (numeric output variable)
            - mean or median estimator
- In **clustering**
  - observations are grouped when at least *k* neighbors are below a given distance threshold
  - let us leave this for later in the course

kNN classifier

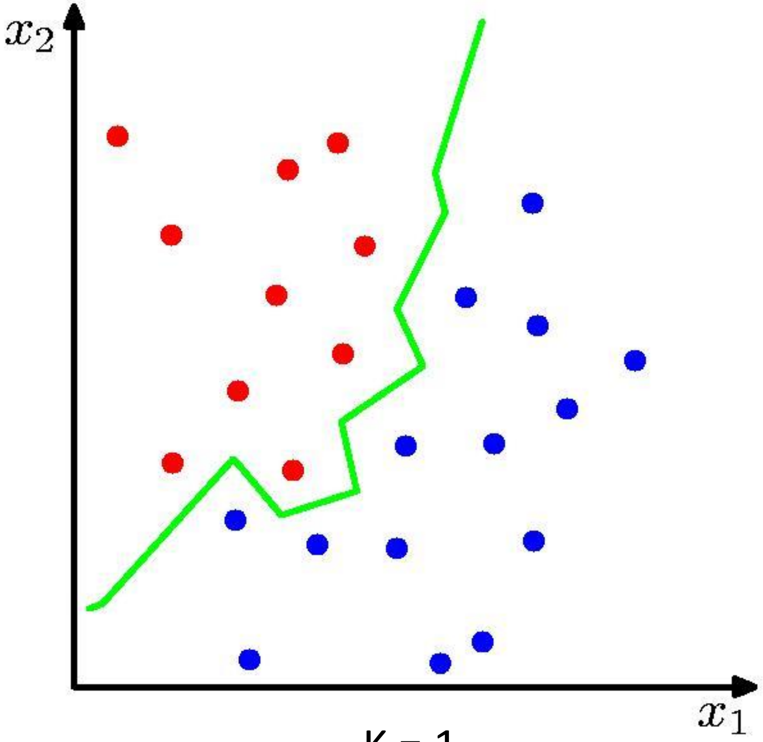
- Data = $\{(\mathbf{x}_1, z_1), (\mathbf{x}_2, z_2) \ldots, (\mathbf{x}_n, z_n)\}$
- Goal: given data, learn classifier f to label new observations, $\hat{z}_{new} = f(\mathbf{x}_{new})$
- kNN **training** algorithm
  - for each training pair observation-target $(\mathbf{x}_i, z_i)$, add the example to the list
- kNN **testing**/classification algorithm
  - given a query instance $\mathbf{x}_{new}$
    - find the k nearest neighbors to $\mathbf{x}_{new}$
    - let $\{\mathbf{x}_1, .., \mathbf{x}_k\}$ be the $\mathbf{x}_{new}$ neighborhood

$$f(\mathbf{x}_{new}) \leftarrow \mathrm{argmax}_{c \in Z} \sum_{i=1}^{k} \delta(c, f(\mathbf{x}_i))$$

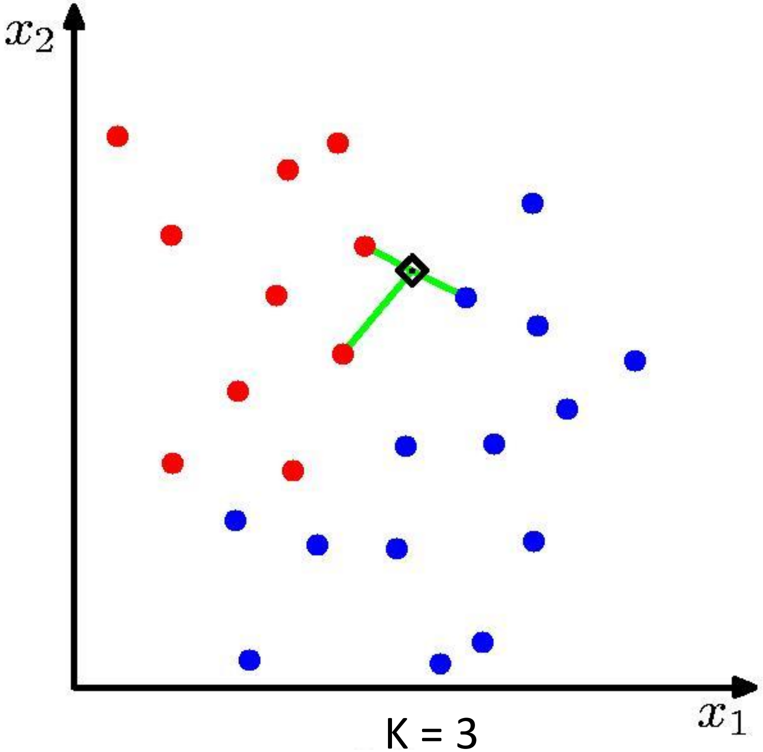
where $\delta(c, \hat{z}) = 1$ if $\hat{z} = c$, else $\delta(c, \hat{z}) = 0$ (**Kronecker** function)

*k*NN classifier

- Exercise: draw the kNN boundary for the following data assuming Euclidean distance and k=3



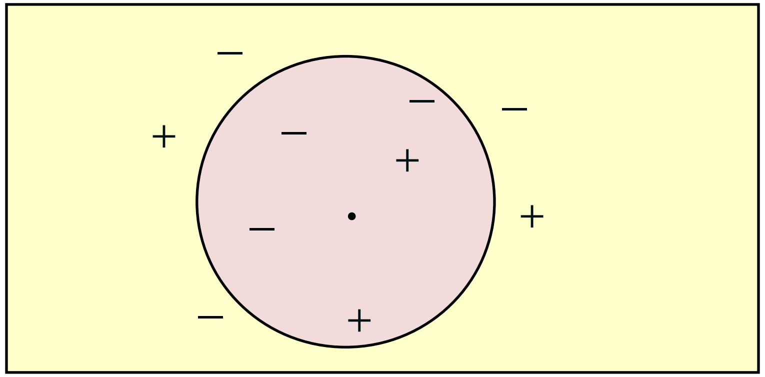
K = 1

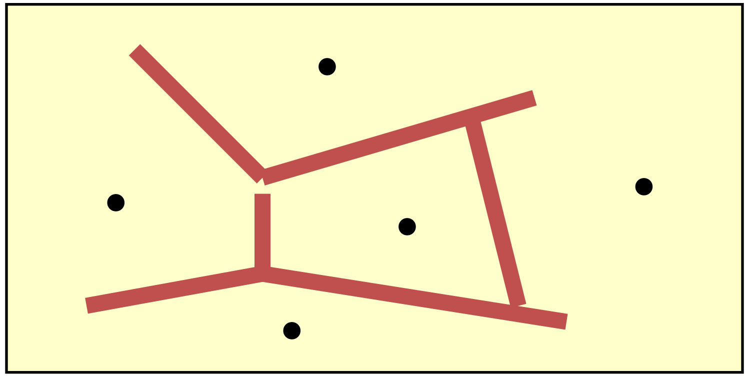


K = 3

Inspecting *k*NN boundaries

- Let us look to classification
  - kNN rule leeds to partition of the space into cells (**Voronoi cells**) enclosing the training points labelled as belonging to the same class
  - the **decision boundary** in a Voronoi tessellation of the feature space resembles the surface of a crystall



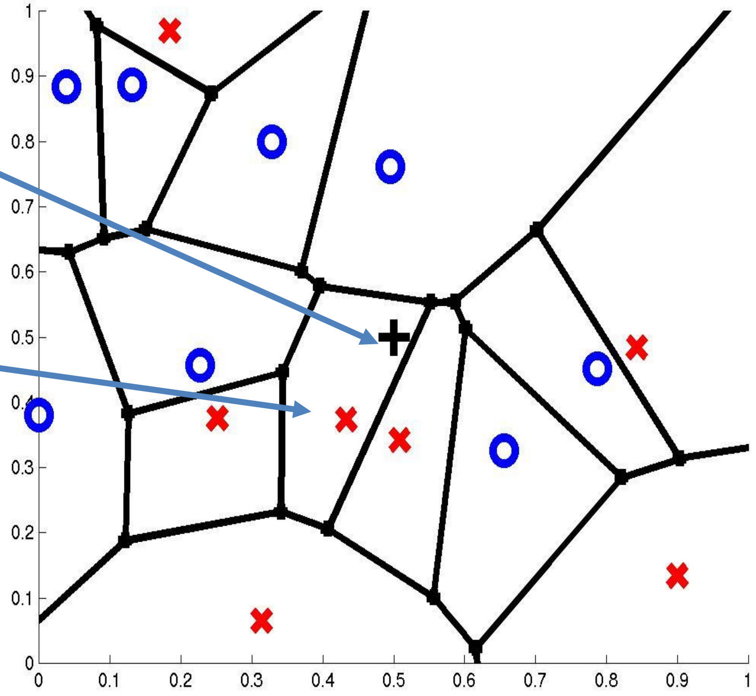


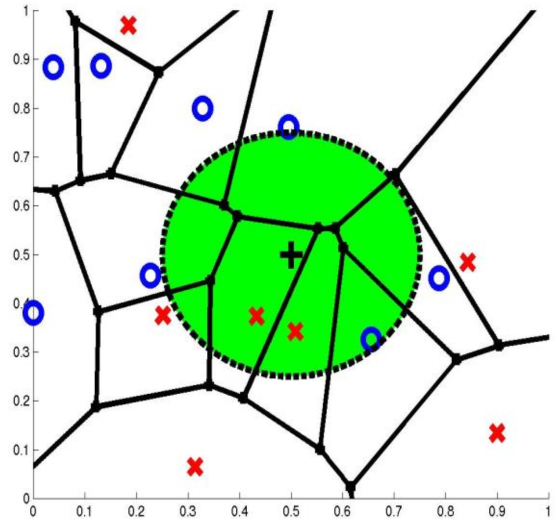
Inspecting *k*NN boundaries

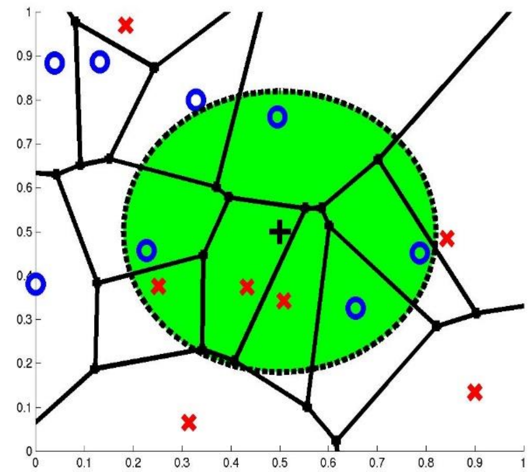
1NN boundaries

query observation $\mathbf{x}_{new}$

nearest neighbor $\mathbf{x}_i$



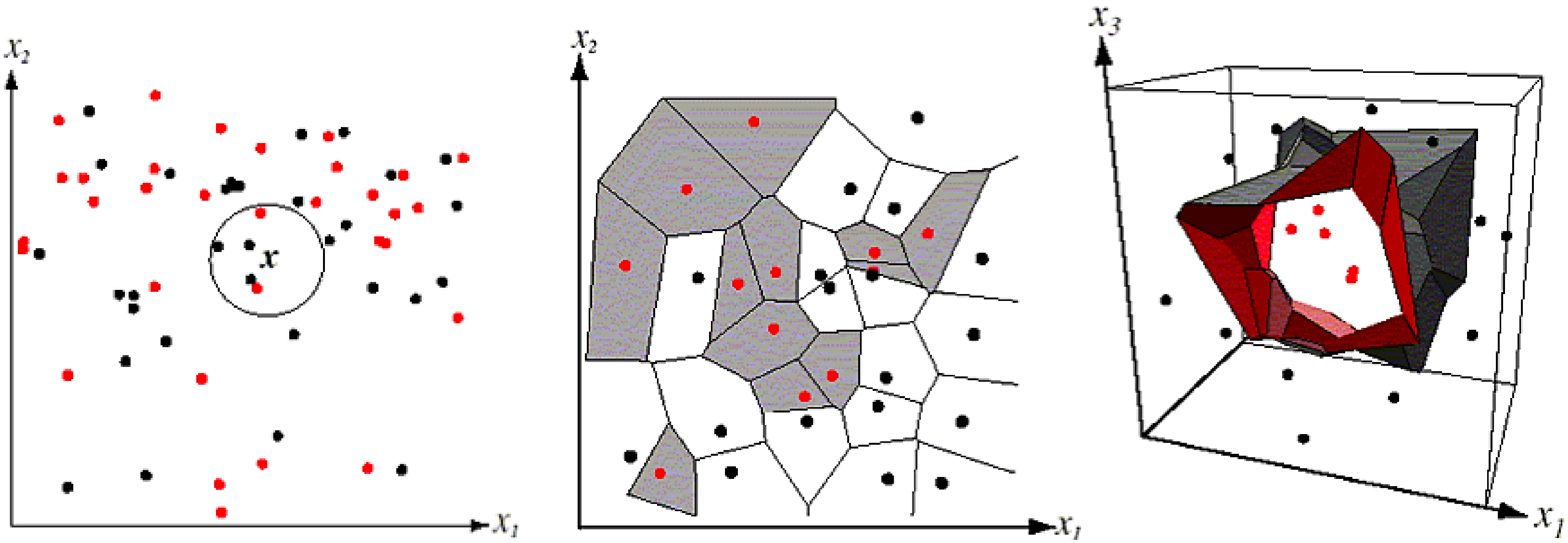




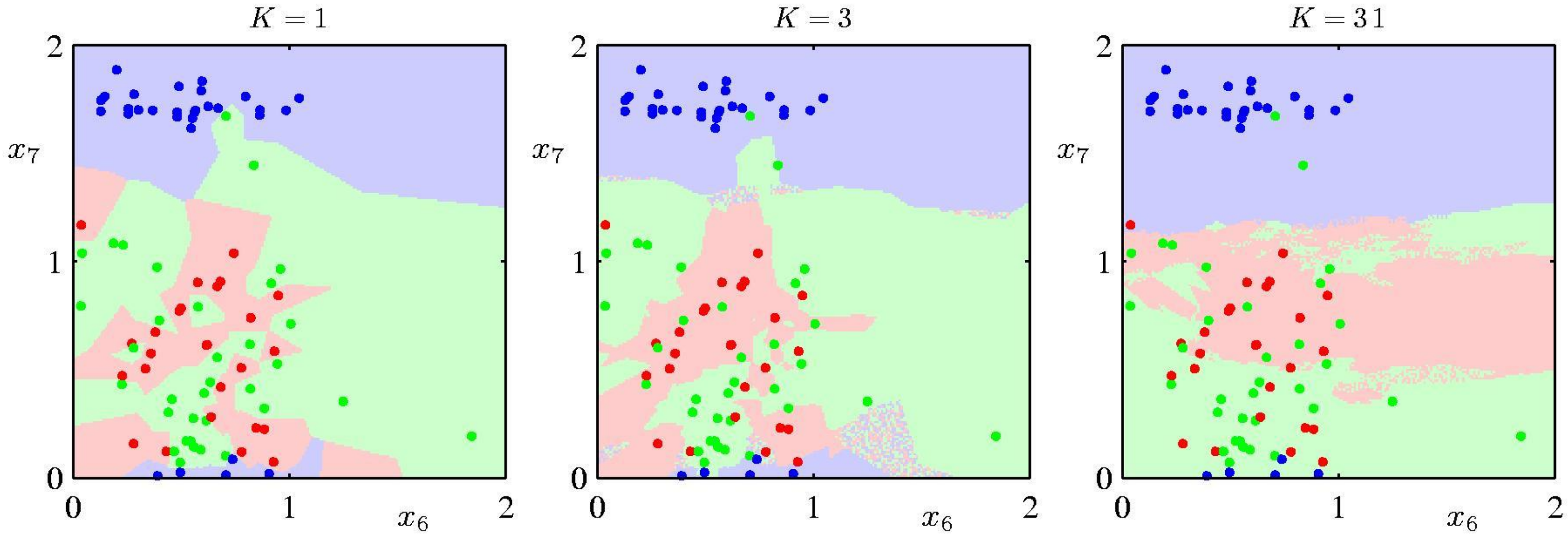
Boundaries preserved in **3NN**? And **7NN**?

Inspecting *k*NN boundaries

- Going into higher-dimensional spaces



kNN classifier: k and smoothing



- k acts as a smother
- For $n \longrightarrow \infty$, the error rate of the 1-nearest-neighbour classifier is never more than twice the optimal error (obtained from the true conditional class distributions)

How to determine *k*?

- Determined experimentally
  - use a test set to validate the error rate of the classifier
  - start with $k = 1$ and assess the error rate
  - repeat with $k = k + 2$
  - choose the value of k for which the error rate is minimum

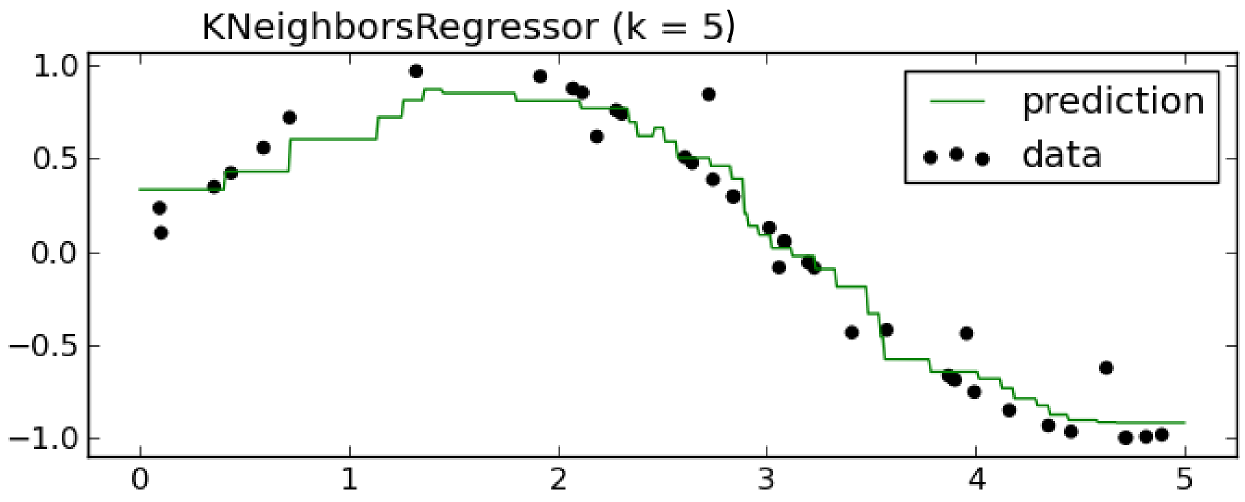
- Note: k should be odd number to avoid ties
- More to come on
  - simple error rate = number of observations incorrectly classified. Better alternatives?
  - training *versus* testing error rates?

*k*NN regressor

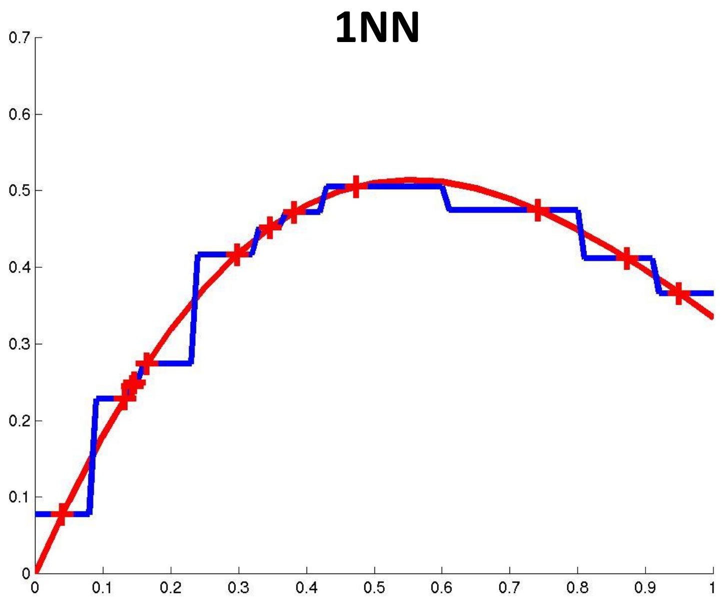
- *k*NN can alternatively approximate continous-valued target functions
  - … by calculating the mean or median value of the *k* nearest training examples

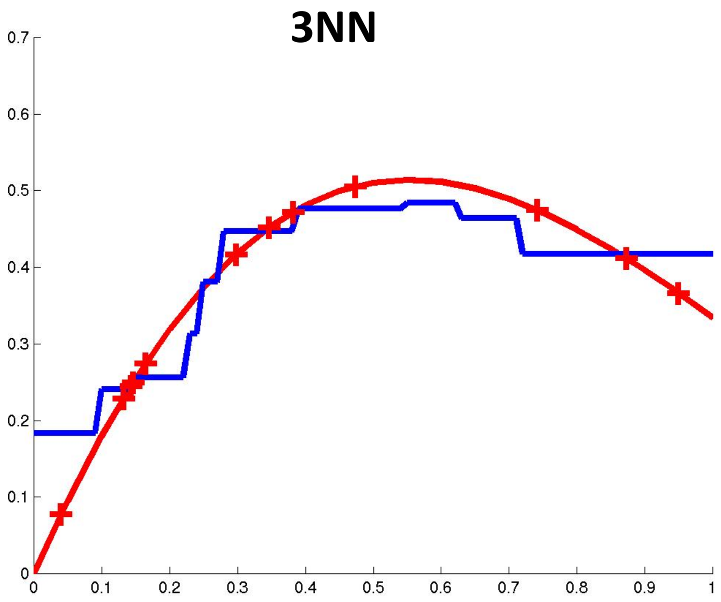
$$f : X \rightarrow \mathbb{R}$$

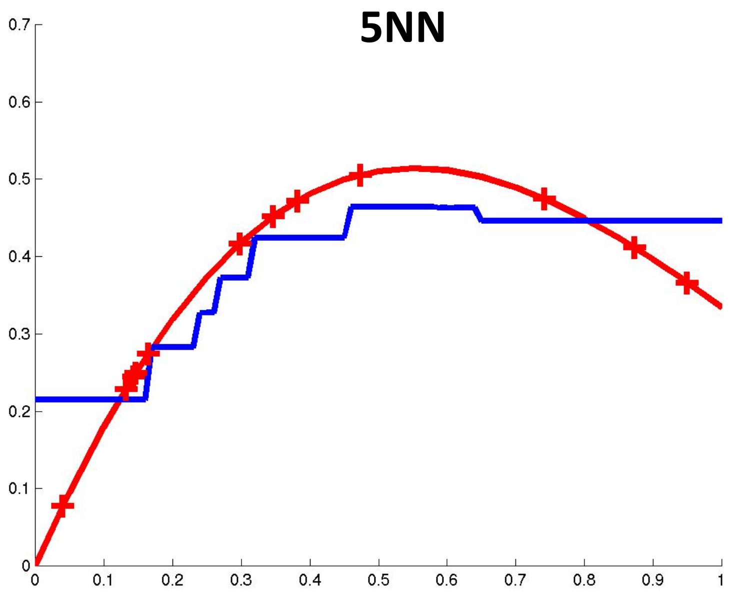
$$f(\mathbf{x}_{new}) \leftarrow \frac{\sum_{i=1}^{k} f(\mathbf{x}_i)}{k}$$



*k*NN regressor







– in **red**: samples of a well-defined signal
– in **blue**: kNN regression function

*k*NN in numeric, categorical and mixed data spaces

- kNN can be parameterize with different similarity criteria
  - *numeric data spaces*
    - **Manhattan** (taxicab) and **Euclidean** distances are common choices
    - recall: when are **Chebyshev** distance, **cosine** similarity and **correlation** coefficient desirable?
  - *categorical data*
    - **encodings** for ordinal variables
    - **Hamming** for nominal variables
    - what is the problem with discrete variables with different cardinalities?
      - solutions?
  - *mixed data*
    - weighted similarity contributions from numeric, ordinal and nominal variables
- kNN can therefore be applied on vector and symbolic spaces

Distance weighted kNN

- Until now we assumed that the weights of the contributions of each k neighbor is **Uniform**
  - i.e. every neighbor in the neighborhood has the same weight to the final decision
- However, some neighbors may be closer to the target observation than others!
- kNN can be refined by considering non-uniform weights
  - How to set **weights**? According to the distance to the query point $\mathbf{x}_{new}$
    - greater weight to closer neighbors
  - For discrete target functions (**classification**)

$$f(\mathbf{x}_{new}) \leftarrow \operatorname{argmax}_{c \in Z} \sum_{i=1}^{k} w_i \delta\big(c, f(\mathbf{x}_i)\big)$$

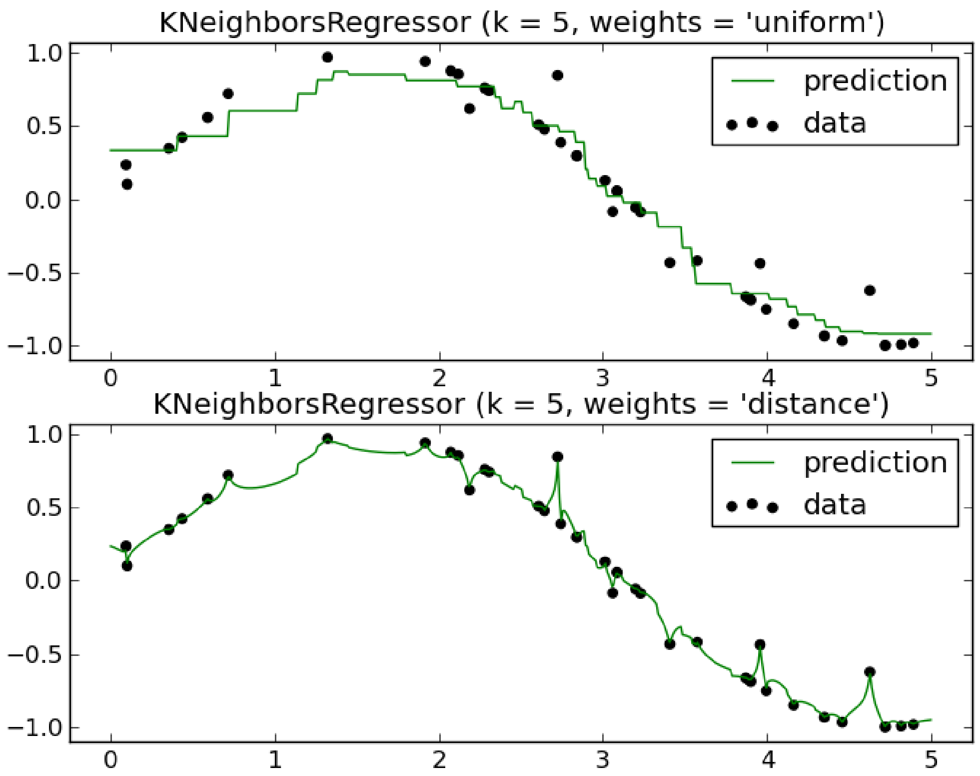
$$w_i(\mathbf{x}_{new}) = \begin{cases} \dfrac{1}{d(\mathbf{x}_{new}, \mathbf{x}_i)} & \text{if } \mathbf{x}_{new} \neq \mathbf{x}_i \\ 1 & \text{else} \end{cases}$$

Distance weighted kNN

- In **regression** (numeric targets):

$$f(\mathbf{x}_{new}) \leftarrow \frac{\sum_{i=1}^{k} w_i f(\mathbf{x}_i)}{\sum_{i=1}^{k} w_i}$$

$$w_i(\mathbf{x}_{new}) = \begin{cases} \dfrac{1}{d(\mathbf{x}_{new}, \mathbf{x}_i)} & \text{if } \mathbf{x}_{new} \neq \mathbf{x}_i \\ 1 & \text{else} \end{cases}$$



*k*NN challenges and solutions

- Let us assume we have three numeric variables, $y_1 \in [0,1], y_2 \in [0,1]$ and $y_3 \in [0,100]$
  - ***Problem*** in the Euclidean space?
    - variables with wider range of values have higher importance when measuring distances
  - **Solution**: variable **normalization**

- Let us assume we have two variables, $y_1 \in \{0,1\}$ and $y_2 \in [0,1]$
  - Comparable problem? Solution?

- Let us assume we have hundreds of variables
  - ***Problem*** in the Euclidean space?
    - contributions from an important subset of variables will get lost amidst all variables!
  - **Solution**: **dimensionality reduction**, e.g. feature selection

Learning Vector Quantization (LQV)

- Are there other instance-based learning algorithms?
  - **LQV** is one example
    - similarly to kNN: distance of unknown observation to reference observations is sought
    - dissimilarly to kNN: not all training examples are stored
      - *classification*: a fixed number of reference observations for each class
      - *regression*: a fixed number of reference observations with considerable target variability
    - How to find reference observations?
      - optimized during learning process
        - guided by class discrimination, numeric output correlation
    - How to place predictions? Given a query observation $\mathbf{x}_{new}$
      - generally a single (k=1) nearest reference observation already provide good predictive quality

Outline

- **Learning in the vector space**
  - from univariate to multivariate data spaces
  - Minkowski distances
  - norm and cosine similarity
  - encodings for categoric variables and mixed data
- **Instance-based learning**
  - parametric vs non-parametric models
  - k-nearest neighbor classifier
  - k-nearest neighbor regressor
  - distance-based kNN
  - challenges: non-iid variables and dimensionality
  - learning vector quantization

Thank You

rmch@tecnico.ulisboa.pt
andreas.wichert@tecnico.ulisboa.pt