Université de Montréal
Département d'informatique et de recherche opérationnelle

IFT6390
Fondements de l'apprentissage machine

Neighbourhood Methods: k-NN, Parzen
for classification, regression and density estimation

Professor: Ioannis Mitliagkas
Slides: Pascal Vincent

Neighbourhood Methods

- k-nearest neighbours (k-PPV or k-NN)
- Parzen windows
- Distances and metrics

Neighbourhood Methods

- Simple idea: neighbours vote on unseen test images

- Ex. k-NN multiclass classification: "which class is the most represented within the k nearest neighbours?"

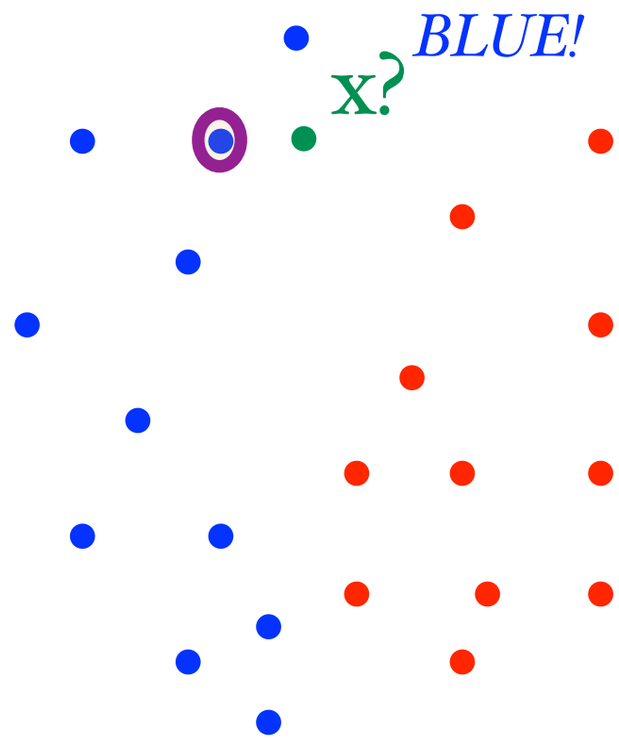
- Neighbourhood methods are "non-parametric": no strong assumption on the form of the target function.

The nearest neighbor algorithm

For a test point x:

- We find the **nearest neighbour** of x within the training set by some measure of distance (eg Euclidean distance).

- We associate x with the class of this nearest neighbor.



Training set

k-NN (k-nearest neighbors)

For binary classification (with $Y_i \in \{-1, 1\}$)

$$f(x) = \text{sign}\left(\frac{1}{k} \sum_{\{i \in 1 \ldots n | X_i \in V(x)\}} Y_i\right)$$

sign of the mean of target values of x's k nearest neighbors

$$f(x) = \text{sign}\left(\frac{1}{k} \sum_{i=1}^{n} I_{\{X_i \in V(x)\}} Y_i\right)$$

V(x) : set containing the k nearest neighbors of x (i.e. closest training points)

$$f(x) = \text{sign}\left(\frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} w_i Y_i\right) \quad \text{with} \quad w_i = I_{\{X_i \in V(x)\}}$$

sign of the **weighted** mean of the target values of **all** the training points, where each weight indicates whether the training point is one of x's closest neighbours.

k-NN (k-nearest neighbors)

For regression (with $Y_i \in \mathbb{R}$ or $Y_i \in \mathbb{R}^m$)

$$f(x) = \cancel{\text{sign}} \left(\frac{1}{k} \sum_{\{i \in 1 \ldots n | X_i \in V(x)\}} Y_i \right)$$

mean of target values of x's k nearest neighbours

$$f(x) = \cancel{\text{sign}} \left(\frac{1}{k} \sum_{i=1}^{n} I_{\{X_i \in V(x)\}} Y_i \right)$$

$V(x)$ = set containing the k nearest neighbors of x (i.e. closest training points)

$$f(x) = \cancel{\text{sign}} \left(\frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} w_i Y_i \right) \quad \text{with} \quad w_i = I_{\{X_i \in V(x)\}}$$

weighted mean of the target values of **all** the training points, where each weight indicates whether the training point is one of x's closest neighbours.

k-NN (k-nearest neighbors)

For multi-class classification (with $Y_i \in 1 \dots m$)

$V(x)$ = set containing the k nearest neighbors of x (i.e. closest training points)

$$f(x) = \arg\max \left(\frac{1}{k} \sum_{\{i \in 1 \dots n | X_i \in V(x)\}} \text{onehot}_m(Y_i) \right)$$

$$f(x) = \arg\max \left(\frac{1}{k} \sum_{i=1}^{n} I_{\{X_i \in V(x)\}} \text{onehot}_m(Y_i) \right)$$

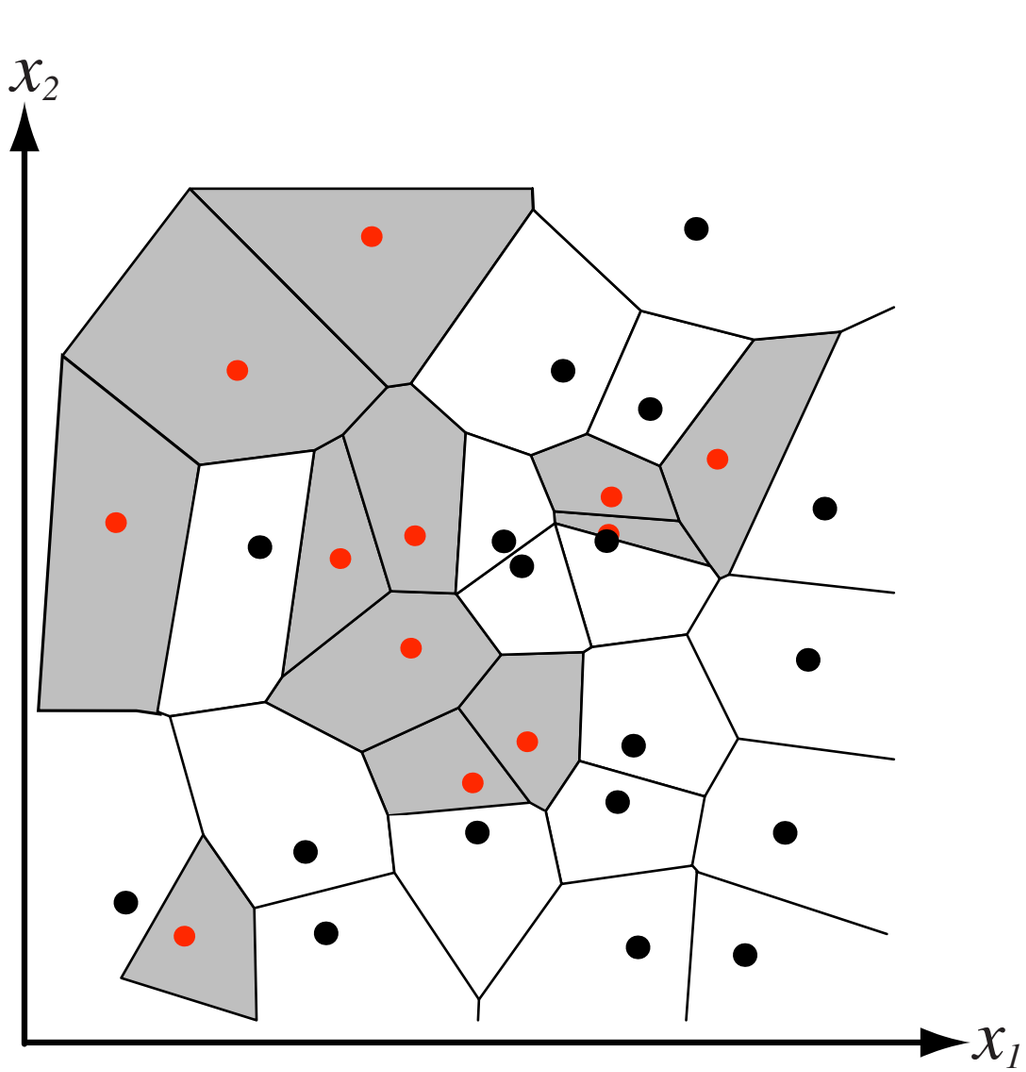
$$f(x) = \arg\max \left(\frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} w_i \text{onehot}_m(Y_i) \right)$$

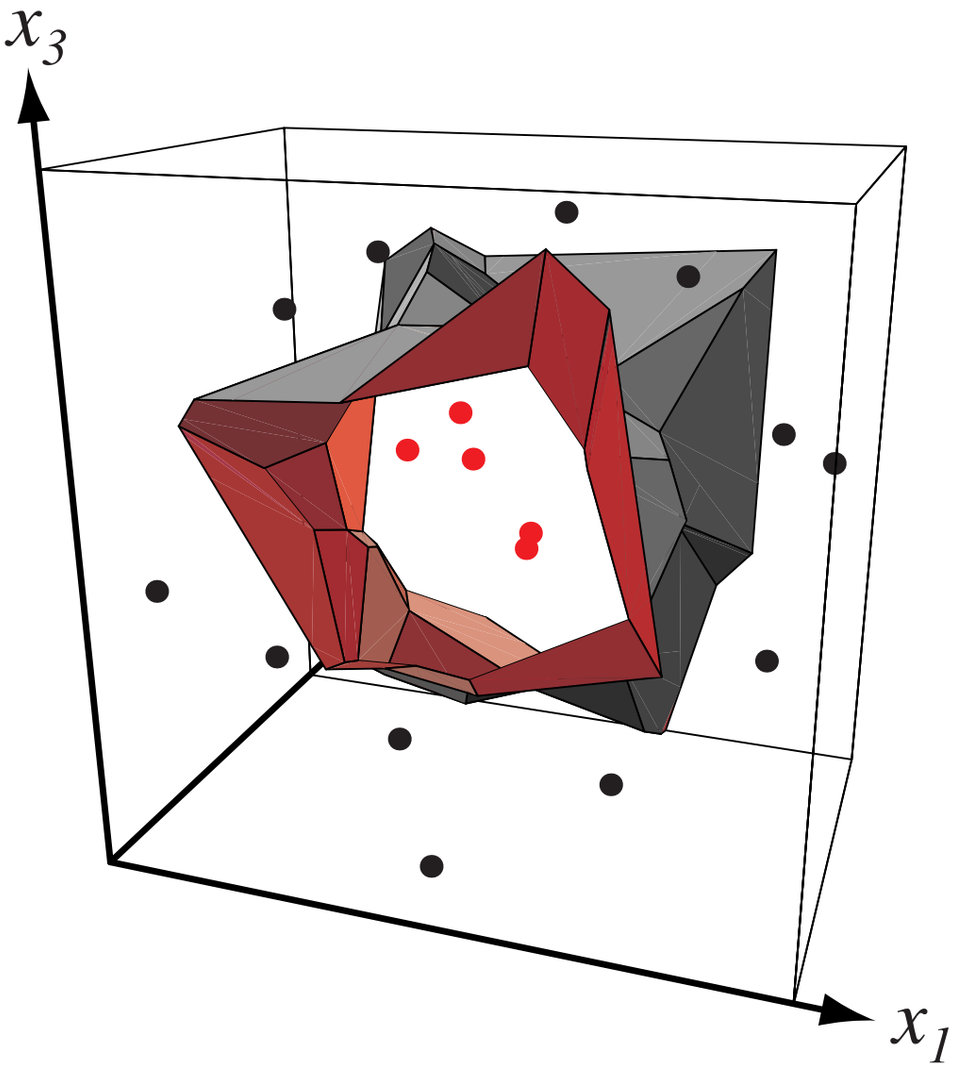
avec $w_i = I_{\{X_i \in V(x)\}}$

Decision boundary of the k-nearest neighbours classifer.

What do they look like?

Voronoi diagrams:





k-Nearest Neighbours

Pseudo-code
and
Algorithmic Complexity

This week's lab session!

Different approach to the neighbourhood method

Parzen windows
(with hard neighbourhood)

- As in k-NN, decision is based on neighbours vote
- In k-NN we take the **k** closest neighbours
- In Parzen with hard neighbourhood, we consider all neighbours within some given length **h.**
- **k** and **h** are hyper-parameters of the algorithms.

Parzen windows
(with hard neighbourhood)

For binary classification (with $Y_i \in \{-1, 1\}$)

$V(x)$ = set of training points at distance less than h from x.

sign of the mean of target values of neighbours of x that are at a distance ≤ h.

$$f(x) = \text{sign}\left(\frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} w_i Y_i\right) \quad \text{with} \quad w_i = I_{\{X_i \in V(x)\}}$$

$$w_i = I_{\{d(X_i,x)<h\}}$$

$$w_i = I_{\{\frac{d(X_i,x)}{h}<1\}}$$

sign of **weighted** mean of the target values of **all** the training points, where each weight indicates whether the training point is at distance ≤ h

Parzen windows
(with hard neighbourhood)

For regression (with $Y_i \in \mathbb{R}$)

$V(x)$ = set of training points at distance less than h from x.

mean of target values of neighbours of x that are at a distance $\leq h$.

$$f(x) = \frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} w_i Y_i \quad \text{avec} \quad w_i = I_{\{X_i \in V(x)\}}$$

$$w_i = I_{\{d(X_i,x)<h\}}$$

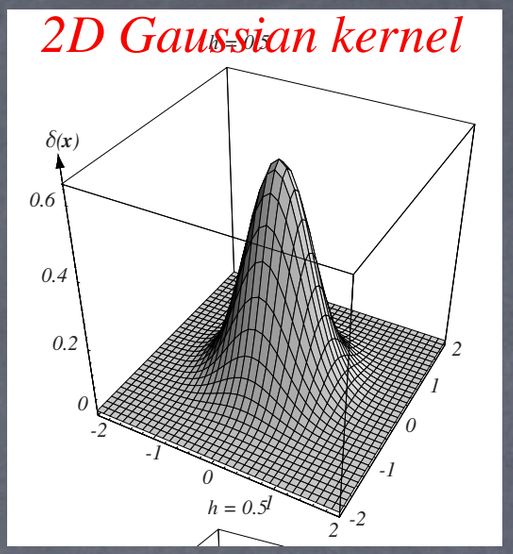
$$w_i = I_{\{\frac{d(X_i,x)}{h}<1\}}$$

weighted mean of the target values of **all** the training points, where each weight indicates whether the training point is at distance $\leq h$

Parzen windows
(with soft neighbourhood)
a.k.a. kernel density estimation

- Generalization of the hard neighbourhood:
- Instead a few points voting, all training points vote.
- But we take a **weighted vote** (i.e. weighted mean):
  the closest the point is to the test point, the more weight its vote has.
- We use **a kernel K** (a similarity measure) to compute the weights.
- The kernel K (and its parameters) are the hyper-parameters of the algorithm.

Parzen window / KDE



For regression (with $Y_i \in \mathbb{R}$)

$$f(x) = \frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} w_i Y_i \qquad \text{with} \qquad w_i = K(X_i, x)$$

weighted mean of the target values of **all** the training points,
where each weight indicates how similar the training point is to x.

K is a *kernel*

Observe that $w_i = I_{\{\frac{d(X_i,x)}{h}<1\}}$ is one possible choice for K
("hard neighbourhood" kernel)

A common choice of kernel function is the
Gaussian kernel (~ gaussian density),
also known as the RBF kernel
(radial basis function)

$$\begin{aligned} K(X_i, x) &= \mathcal{N}_{x,\sigma^2}(X_i) = \mathcal{N}_{X_i,\sigma^2}(x) \\ &= \frac{1}{(2\pi)^{\frac{d}{2}}\sigma^d} e^{-\frac{1}{2}\frac{d(X_i,x)^2}{\sigma^2}} \end{aligned}$$

Parzen window / KDE

(regession and classification summary)

Regression ($Y_i \in \mathbb{R}$) :

$$f(x) = \frac{1}{\sum_{i=1}^{n} K(X_i, x)} \sum_{i=1}^{n} K(X_i, x) Y_i$$

Binary classification ($Y_i \in \{-1, 1\}$) :

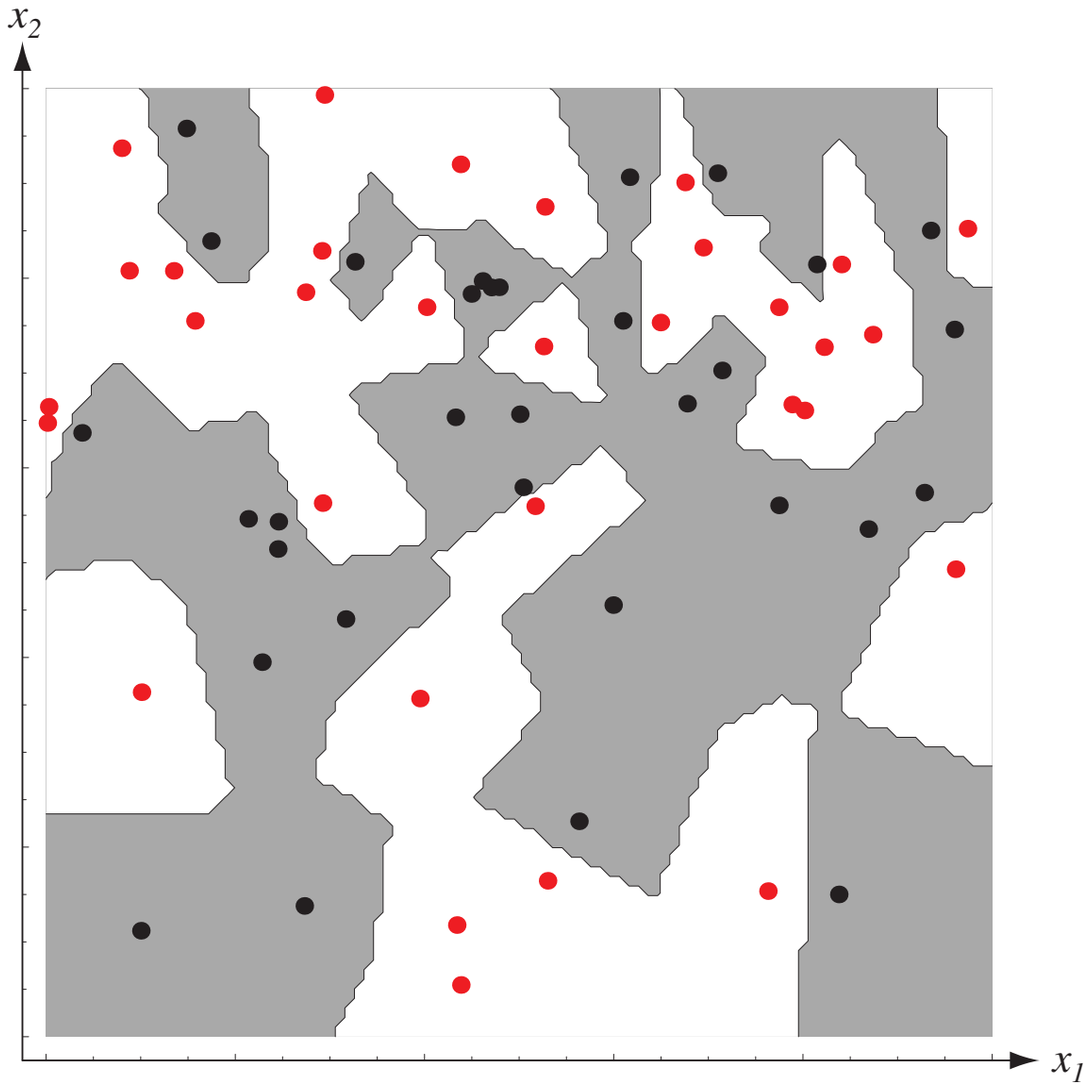
$$f(x) = \text{sign}\left(\frac{1}{\sum_{i=1}^{n} K(X_i, x)} \sum_{i=1}^{n} K(X_i, x) Y_i \right)$$

Multi-class classification ($Y_i \in 1 \ldots m$) :

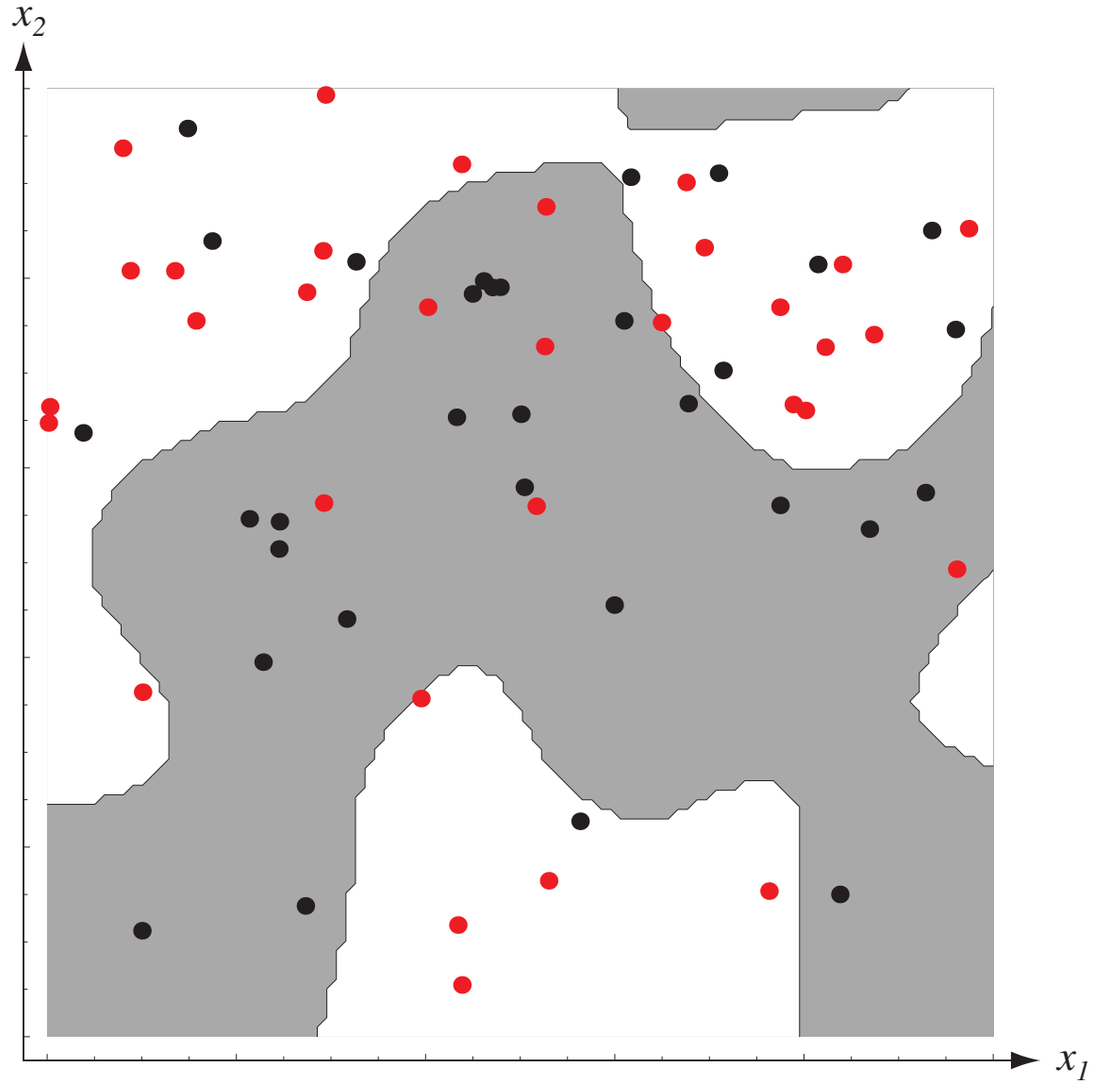
$$f(x) = \arg\max\left(\frac{1}{\sum_{i=1}^{n} K(X_i, x)} \sum_{i=1}^{n} K(X_i, x) \text{onehot}_m(Y_i) \right)$$

Example: 2D classification with Parzen

small σ



large σ



Gray region: black label predicted
White region: red label predicted

Parzen windows / KDE

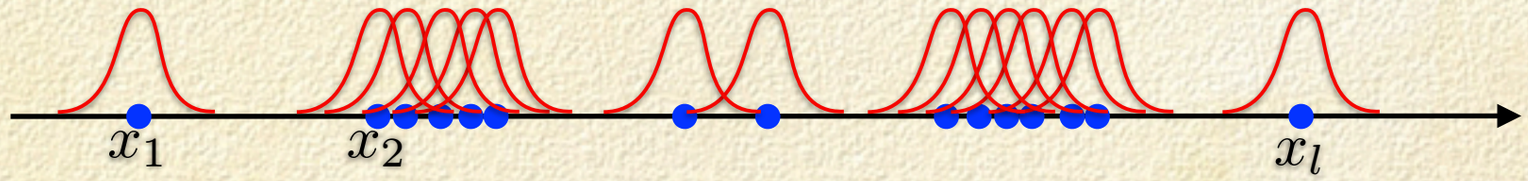
For density estimation:

$$f(x) = \hat{p}(x) = \frac{1}{n} \sum_{i=1}^{n} K(X_i; x)$$

where K needs to be a probability density/mass function.
Ex: Gaussian centered in X_i

The Parzen estimator is indeed a probability density/mass function
(it is positive and integrates/sums to one).

1D density estimation with Parzen windows



If we choose

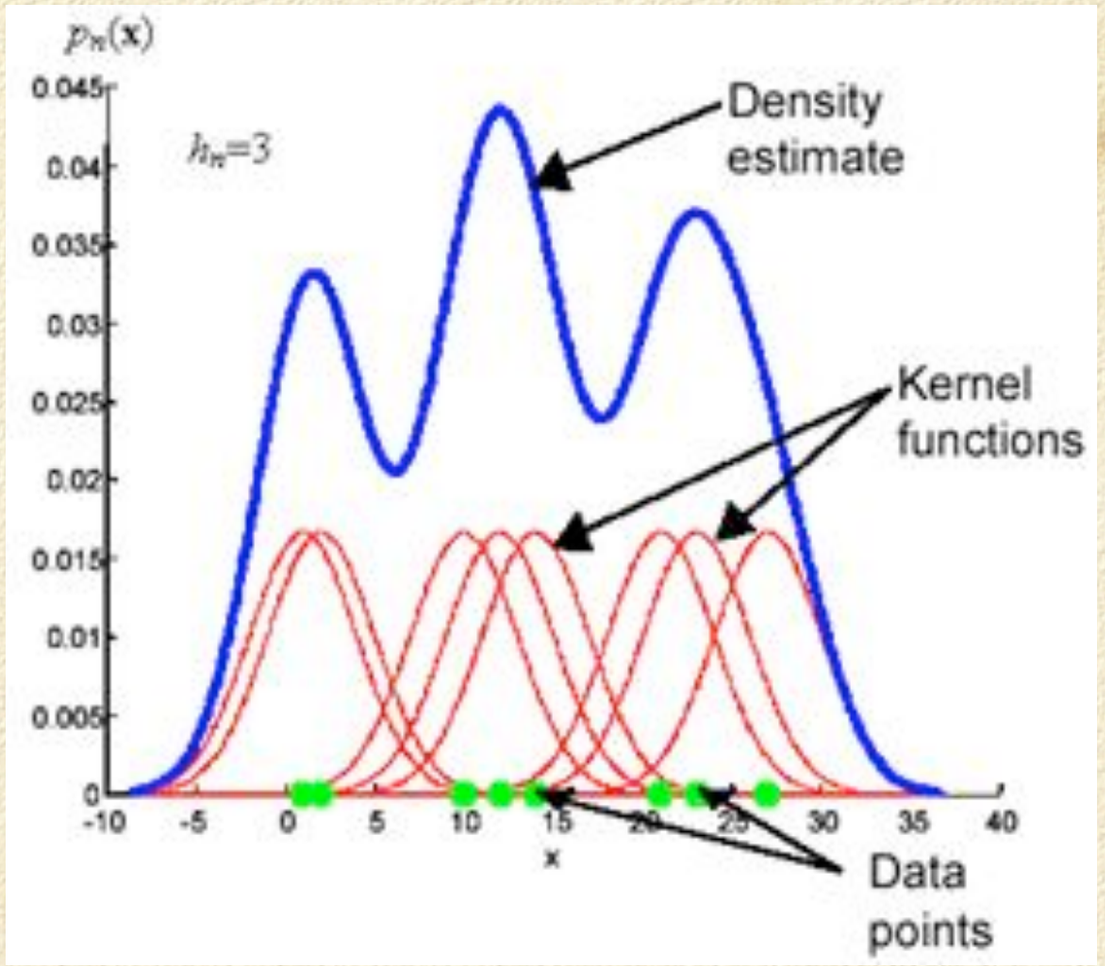
$$K(X_i; x) = \mathcal{N}_{X_i,\sigma}(x)$$

a 1-dimensional normal distribution:

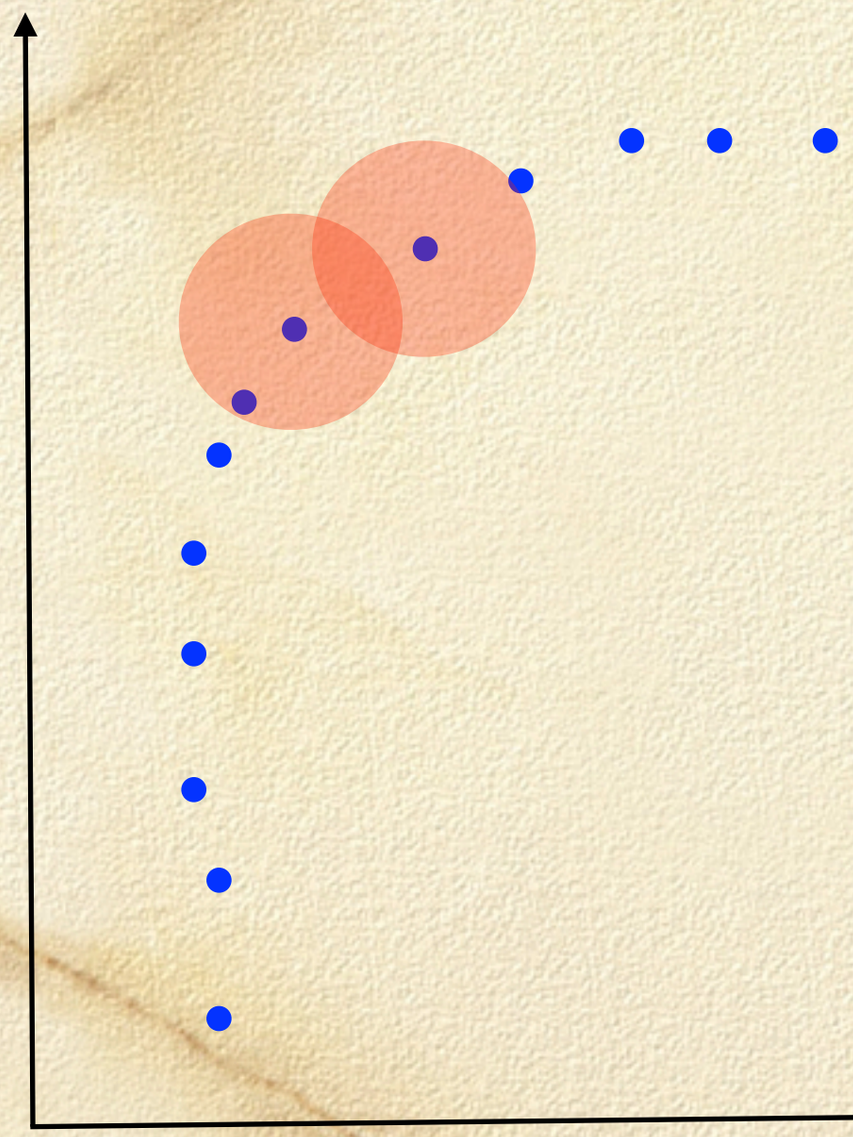
$$\mathcal{N}_{\mu,\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Parzen density estimator:

$$\hat{p}(x) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{N}_{X_i,\sigma}(x)$$



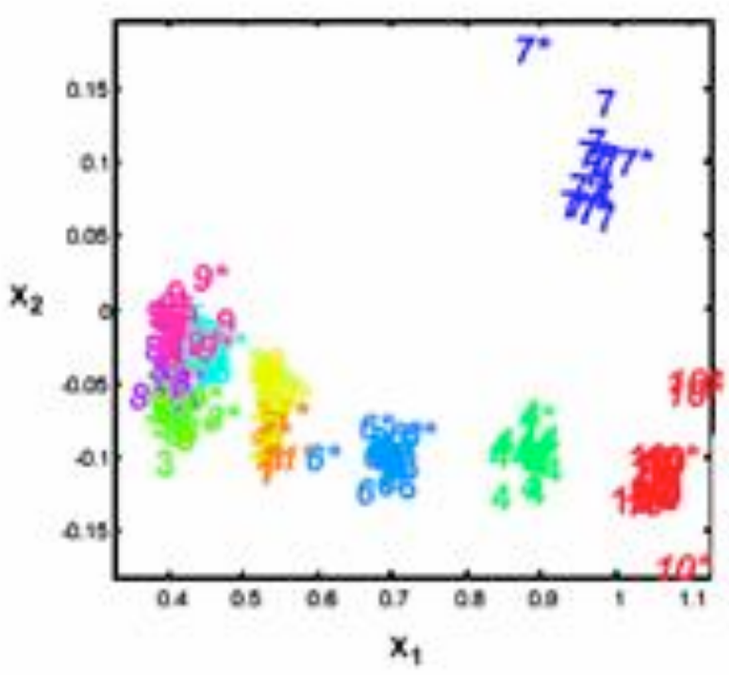
Parzen window
for density estimation in 2D



Isotropic d-dimensional Gaussian:

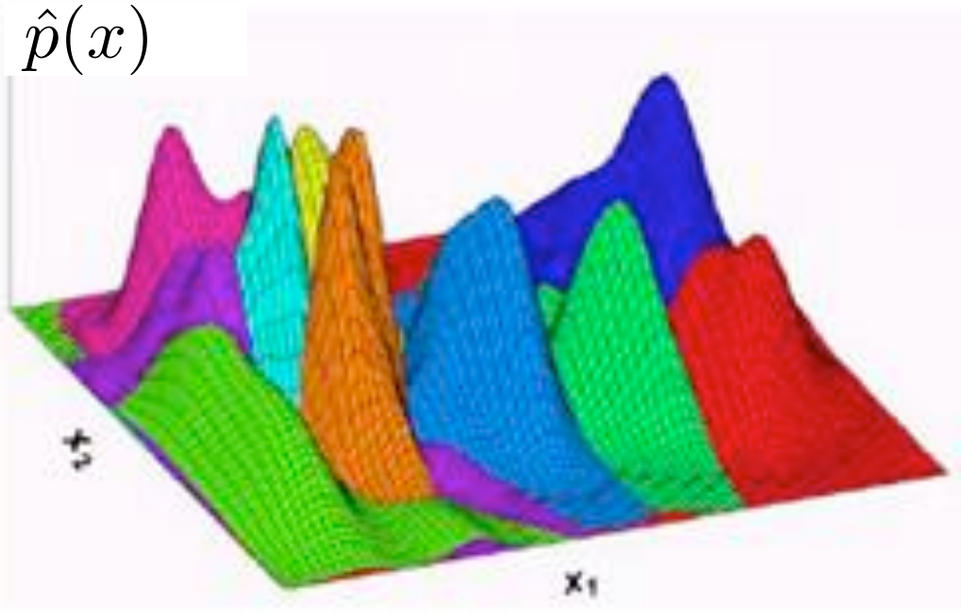
$$\mathcal{N}_{\mu,\sigma}(x) = \frac{1}{(2\pi)^{\frac{d}{2}}\sigma^d} e^{-\frac{1}{2}\frac{\|x-\mu\|^2}{\sigma^2}}$$

Parzen windows



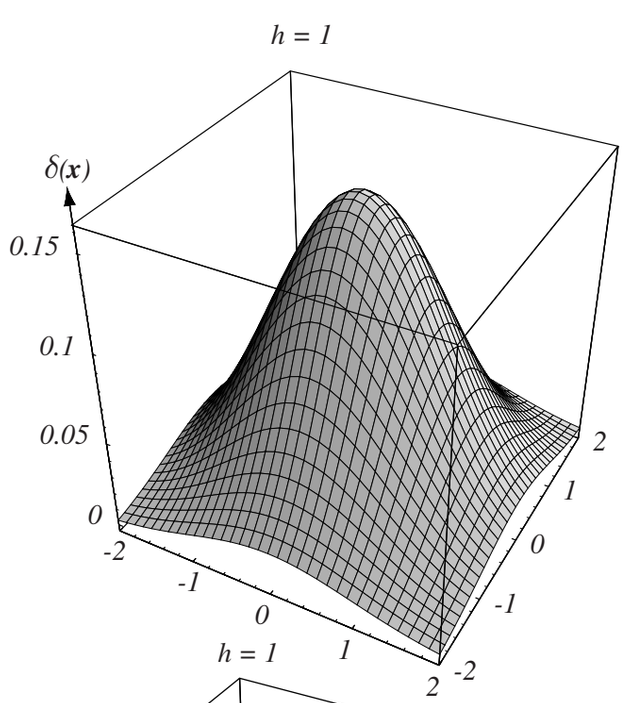
NON-PARAMETRIC
DENSITY ESTIMATION

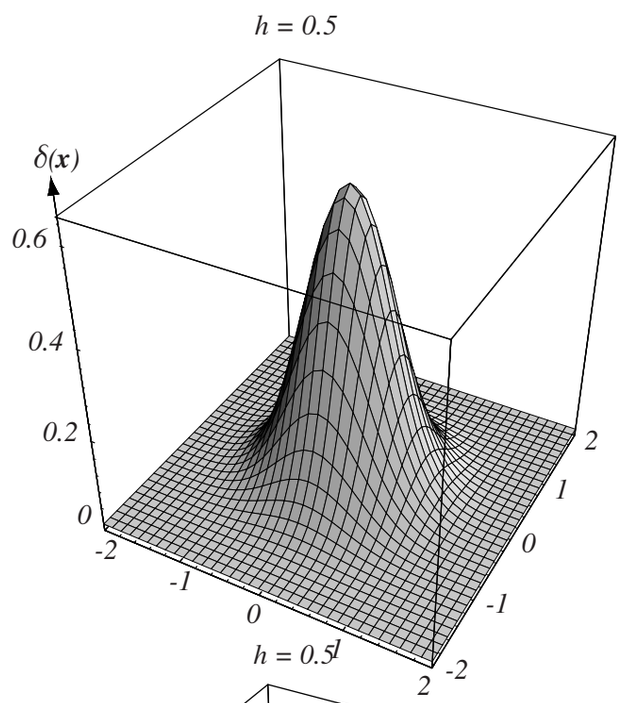
$\hat{p}(x)$

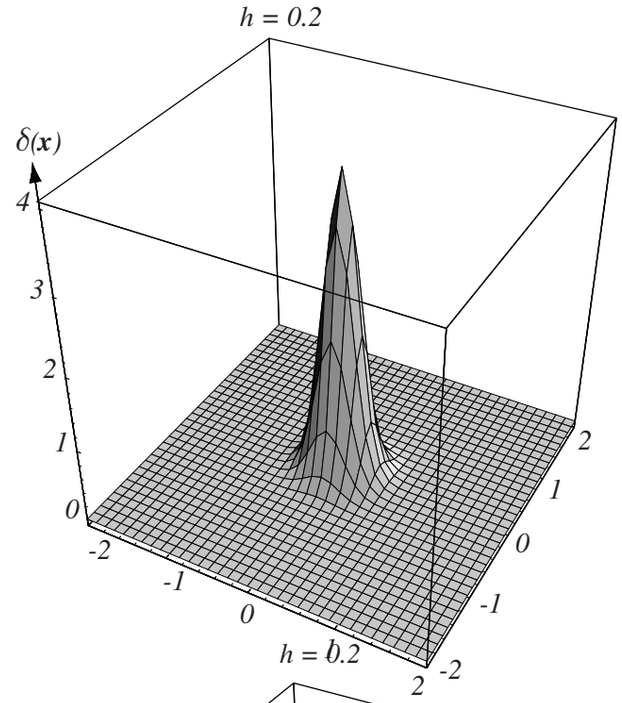


Effect of the width σ of the (isotropic) Gaussian kernel

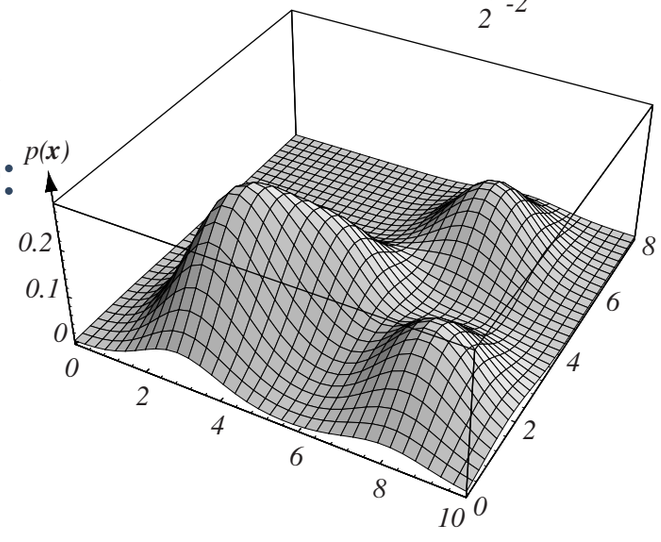
Kernel:

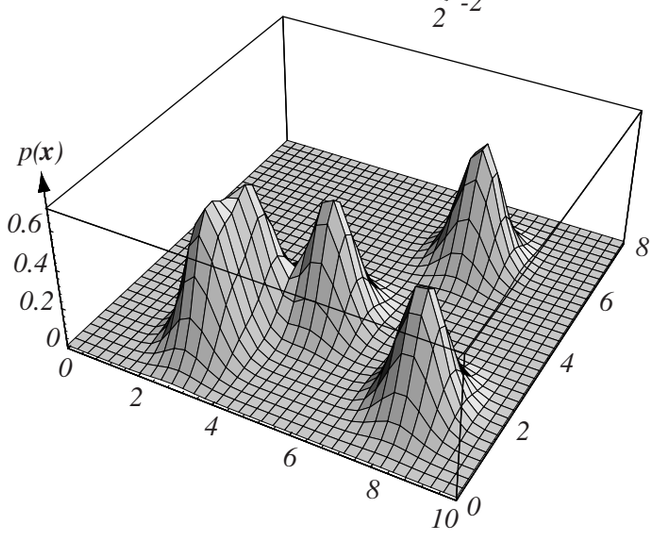


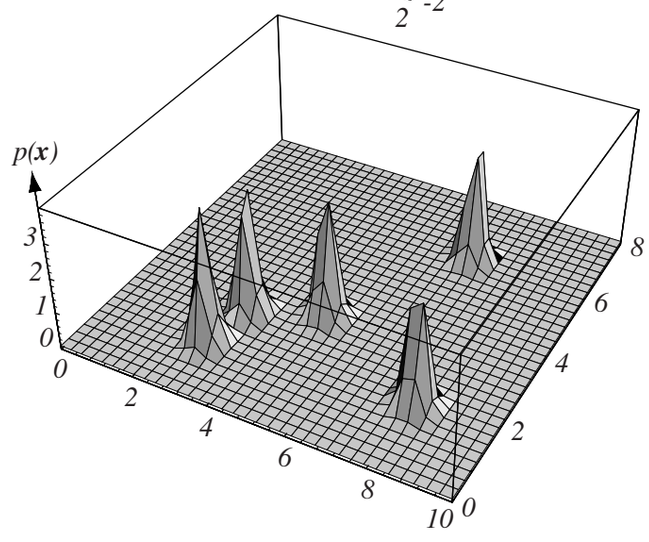




Density estimate:

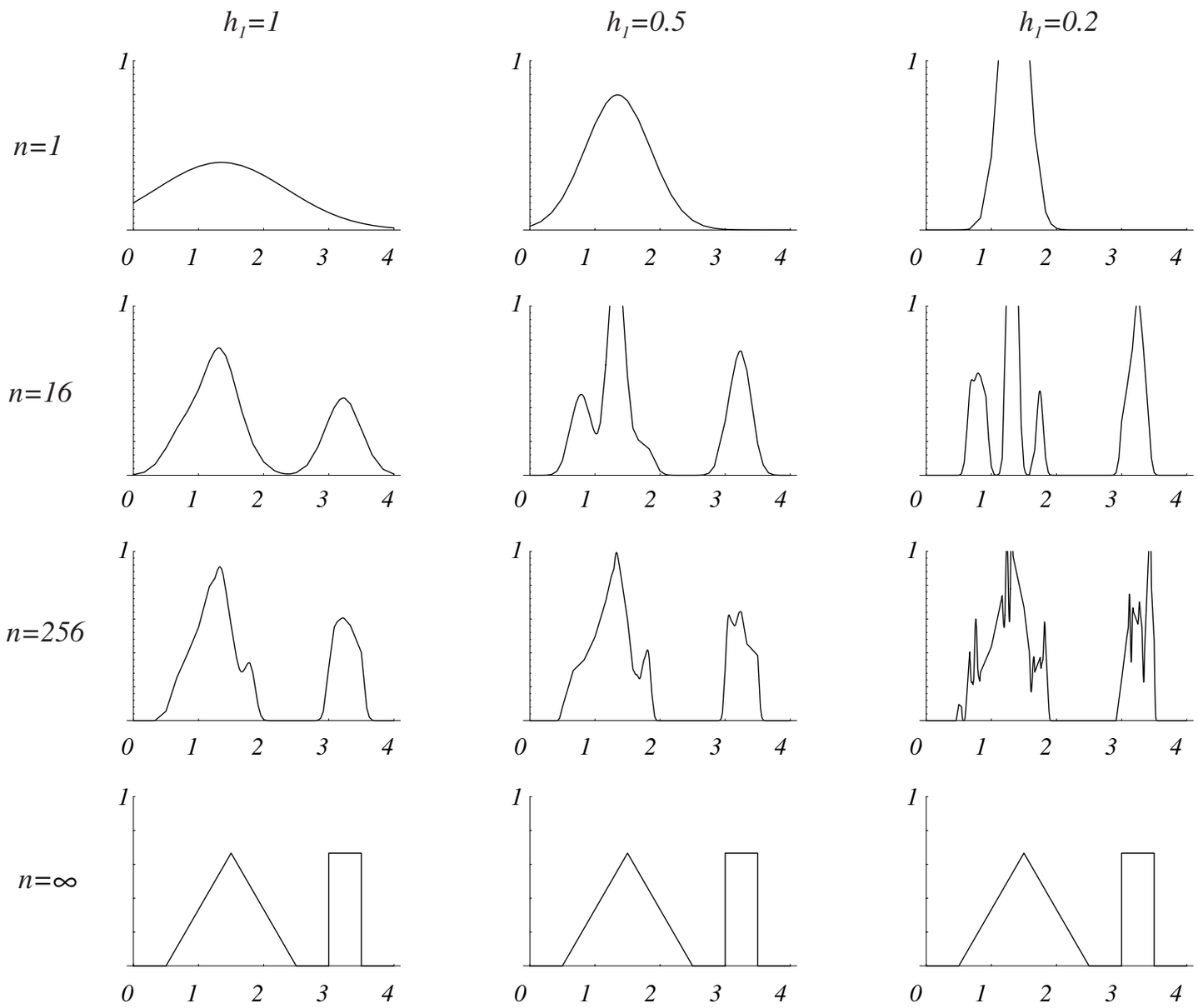






Example: 1D density estimation with Parzen

- Exemple: $p(x) \sim$ triangle + uniform, $\phi(u) = \frac{1}{\sqrt{2\pi}} e^{-u^2/2}$



Distances and similarity measures

- Many learning algorithms rely on some notion of «distance» or «similarity» between input points
- Necessary to define «neighbourhoods»
- Examples of neighbourhood methods: K-NN, Parzen

Distances = Metrics

Metric properties:

- positive: $d(\mathbf{a}, \mathbf{b}) \geq 0$
- identity of indiscernibles: $d(a, b) = 0$ if and only if $a = b$
- symetric: $d(\mathbf{a}, \mathbf{b}) = d(\mathbf{b}, \mathbf{a})$
- triangle inequality: $d(\mathbf{a}, \mathbf{b}) + d(\mathbf{b}, \mathbf{c}) \geq d(\mathbf{a}, \mathbf{c})$

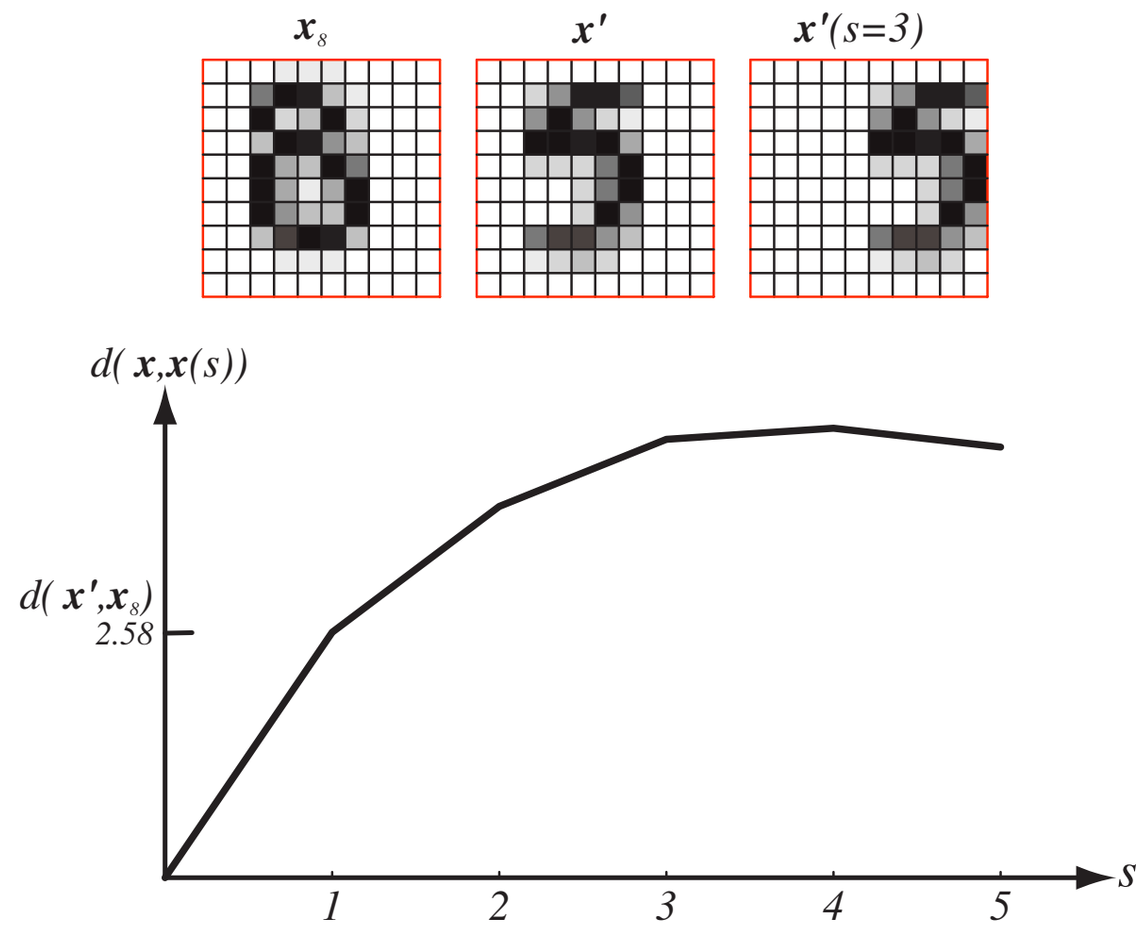
Exemple:

Euclidian distance (L2)

$$\mathbf{a} \in \mathbb{R}^d, \quad \mathbf{b} \in \mathbb{R}^d$$

$$d(\mathbf{a}, \mathbf{b}) = \|\mathbf{a} - \mathbf{b}\| = \sqrt{\sum_{i=1}^{d} (a_i - b_i)^2}$$

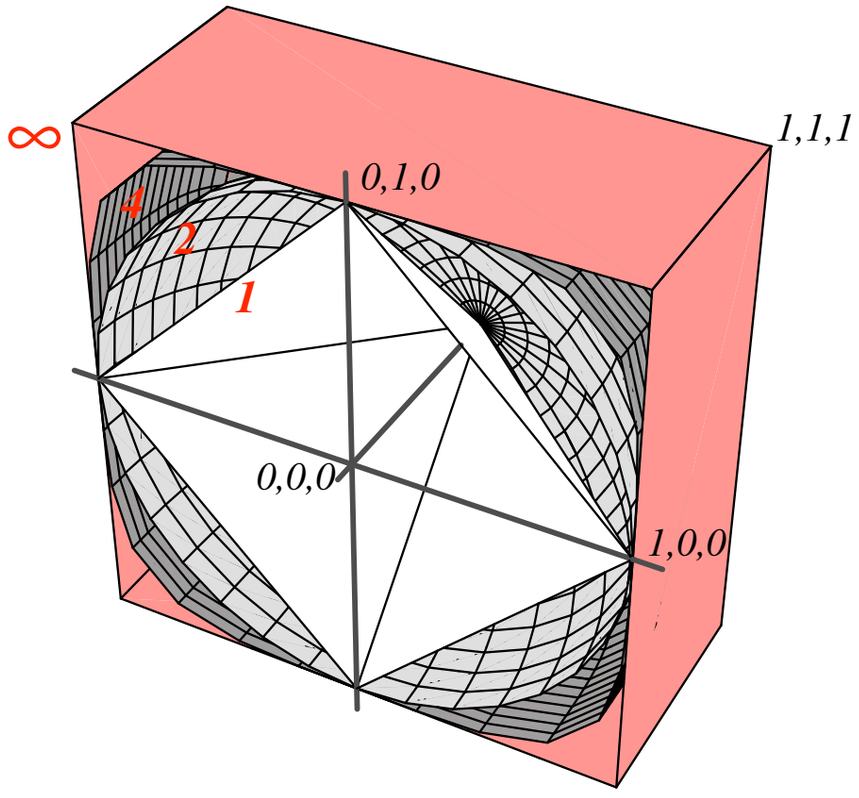
Limitations of Euclidean metric



Examples of metrics

euclidienne	L_2	$d(\mathbf{a},\mathbf{b}) = \left(\sum_{i=1}^{d} (a_i - b_i)^2 \right)^{1/2}$
Manhattan	L_1	$d(\mathbf{a},\mathbf{b}) = \sum_{i=1}^{d} \lvert a_i - b_i \rvert$
	L_∞	$d(\mathbf{a},\mathbf{b}) = \max_i \lvert a_i - b_i \rvert$
Minkowski	L_p	$d(\mathbf{a},\mathbf{b}) = \left(\sum_{i=1}^{d} \lvert a_i - b_i \rvert^p \right)^{1/p}$
Cosine	$L_{\cos}$	$d(\mathbf{a},\mathbf{b}) = 1 - \frac{\langle \mathbf{a},\mathbf{b} \rangle}{\lVert \mathbf{a} \rVert \cdot \lVert \mathbf{b} \rVert}$

Minkowski metric (Lp)



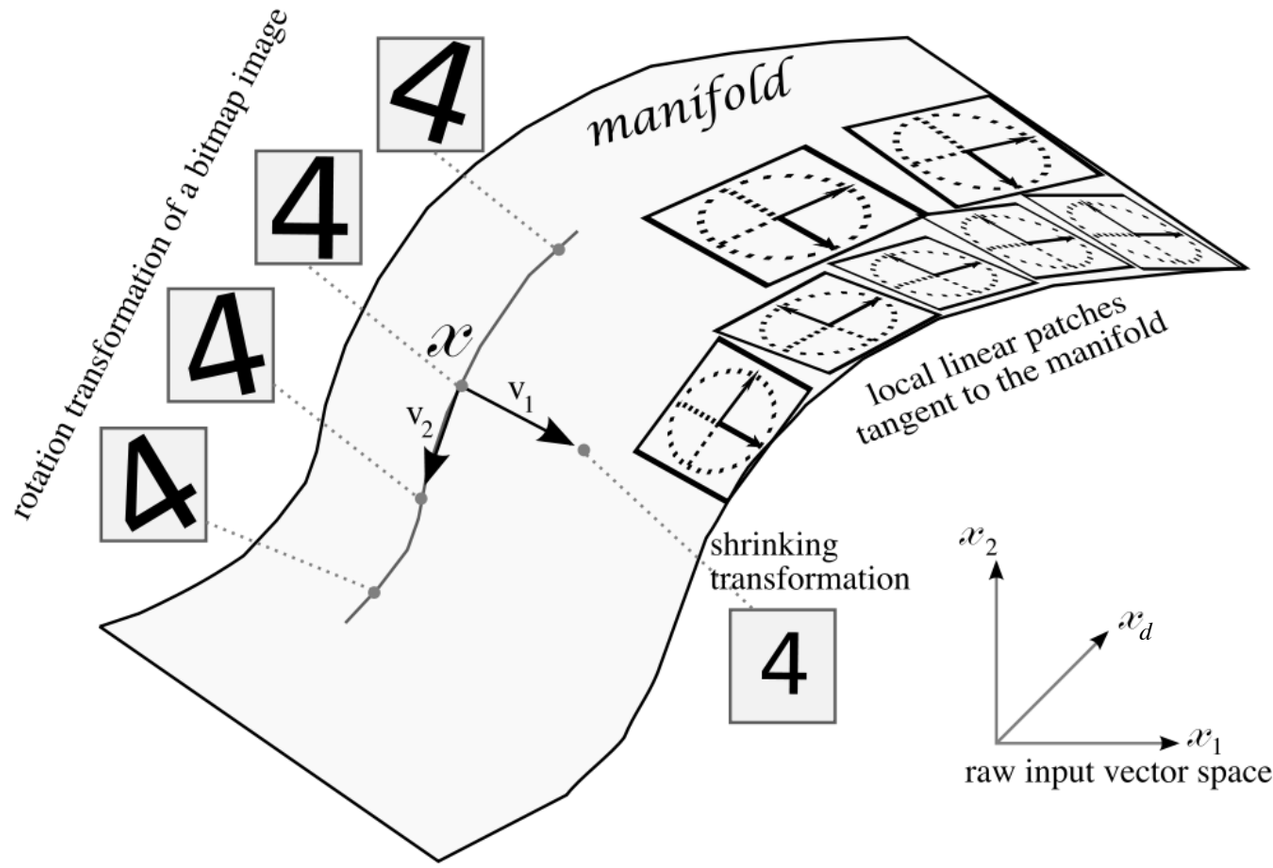
A lot of other metrics out there!

Low-dimensional manifolds

Ex: face orientation parameter

Image borrowed from University of Dayton Vision Lab website.

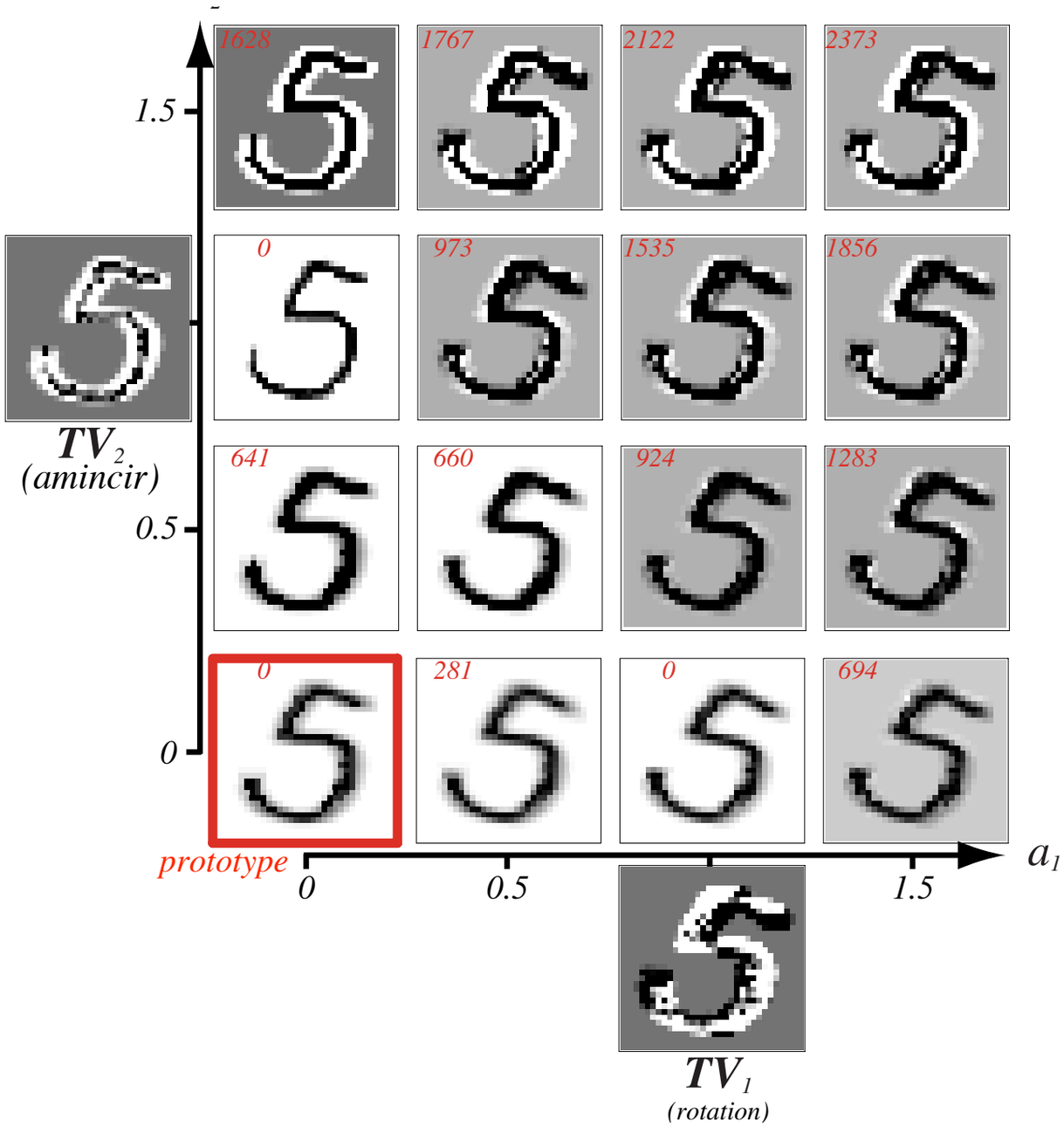
Ex: rotation, character size (+ line thickness, ...)



Tangent distance

- Capture invariance under some transforms:

$$\mathbf{TV}_i = F_i(\mathbf{x}'; a_i) - \mathbf{x}'$$



Tangent distance

$$d_{tan}(\mathbf{x}', \mathbf{x}) = \min_{\mathbf{a}}[\|(\mathbf{x}' + \mathbf{Ta}) - \mathbf{x}\|]$$