

IFT3395/6390
Fondements de l'apprentissage machine

Méthodes à base de voisinage: k-NN, Parzen pour classification, régression, estimation de densité

Professeur: Pascal Vincent

Méthodes à base de voisinage

- k plus proches voisins (k-PPV ou k-NN)
- Fenêtres de Parzen
- Notion de distance / métrique

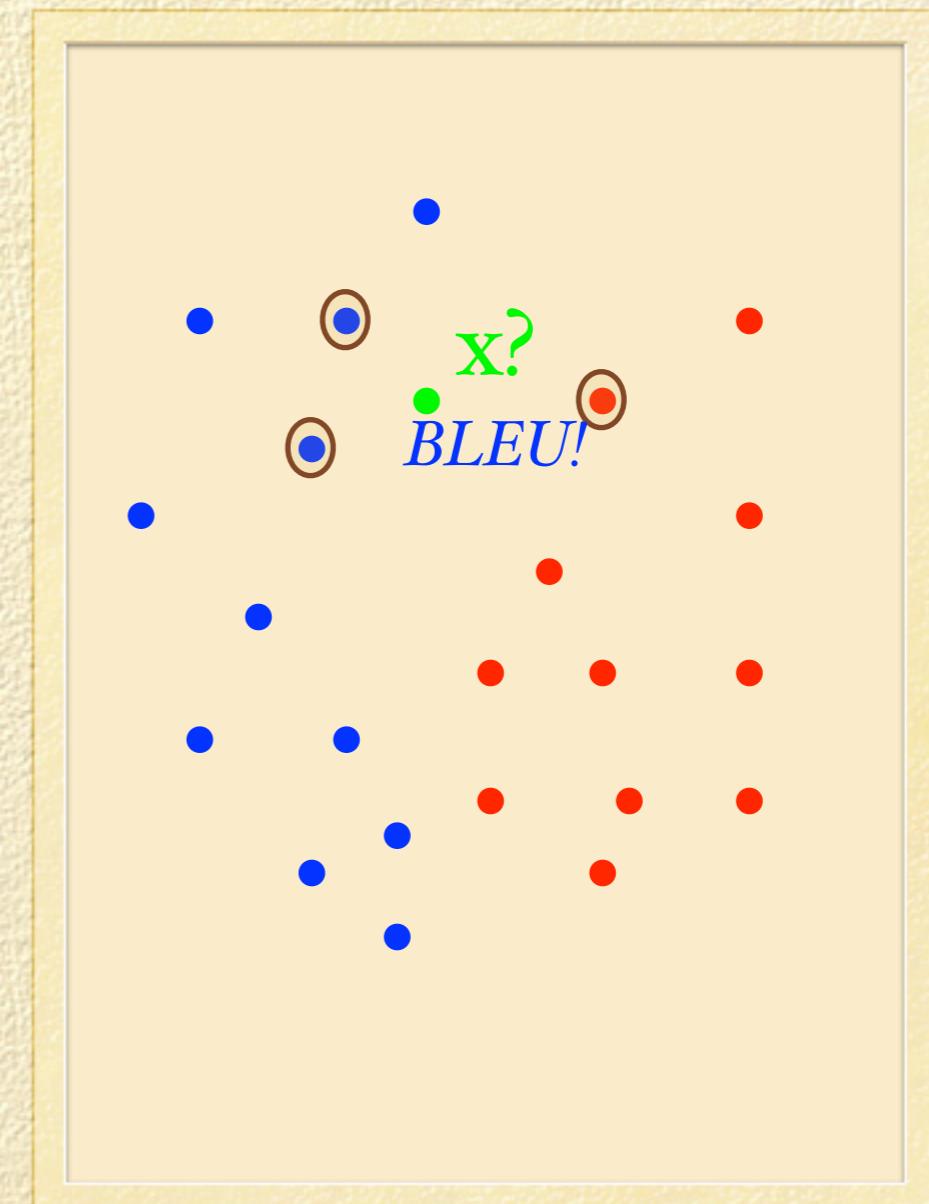
Méthodes à base de voisinage

- ➊ Une idée simple: faire voter les voisins du point de test.
- ➋ Ex. k-NN classification multiclass: “parmi mes k plus proches voisins, quelle classe est majoritaire?”
- ➌ Tout comme les méthodes de type histogramme (quadrillage de l'espace), les méthodes à base de voisinage sont des méthodes dites “non-paramétriques”.

L'algorithme classique des k plus proches voisins (kNN) pour la classification

Pour un point test x :

- On trouve les k plus proches voisins de x parmi l'ensemble d'apprentissage (typiquement selon la distance Euclidienne).
- On associe à x la classe majoritaire parmi ses k voisins



k-NN (k nearest neighbors)

k-PPV (k plus proches voisins)

Pour la classification binaire (avec $Y_i \in \{-1, 1\}$)

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

signe de

la moyenne des valeurs cibles des k voisins les plus proches de x

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

V(x) = ensemble des k plus proches voisins de x dans l'ensemble d'apprentissage

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

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

signe de la moyenne pondérée des valeurs cibles de tous les points d'entraînement, pondérées par un poids indiquant si le point d'entraînement est voisin de x.

k-NN (k nearest neighbors)

k-PPV (k plus proches voisins)

Pour la régression (avec $Y_i \in \mathbb{R}$ ou $Y_i \in \mathbb{R}^m$)

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

la moyenne des valeurs cibles des k voisins les plus proches de x

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

$V(x)$ = ensemble des k plus proches voisins de x dans l'ensemble d'apprentissage

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

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

la moyenne pondérée des valeurs cibles de tous les points d'entraînement, pondérées par un poids indiquant si le point d'entraînement est voisin de x.

k-NN (k nearest neighbors)

k-PPV (k plus proches voisins)

Pour la classification multiclass (avec $Y_i \in 1 \dots m$)

$V(x)$ = ensemble des k plus proches voisins de x dans l'ensemble d'apprentissage

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

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

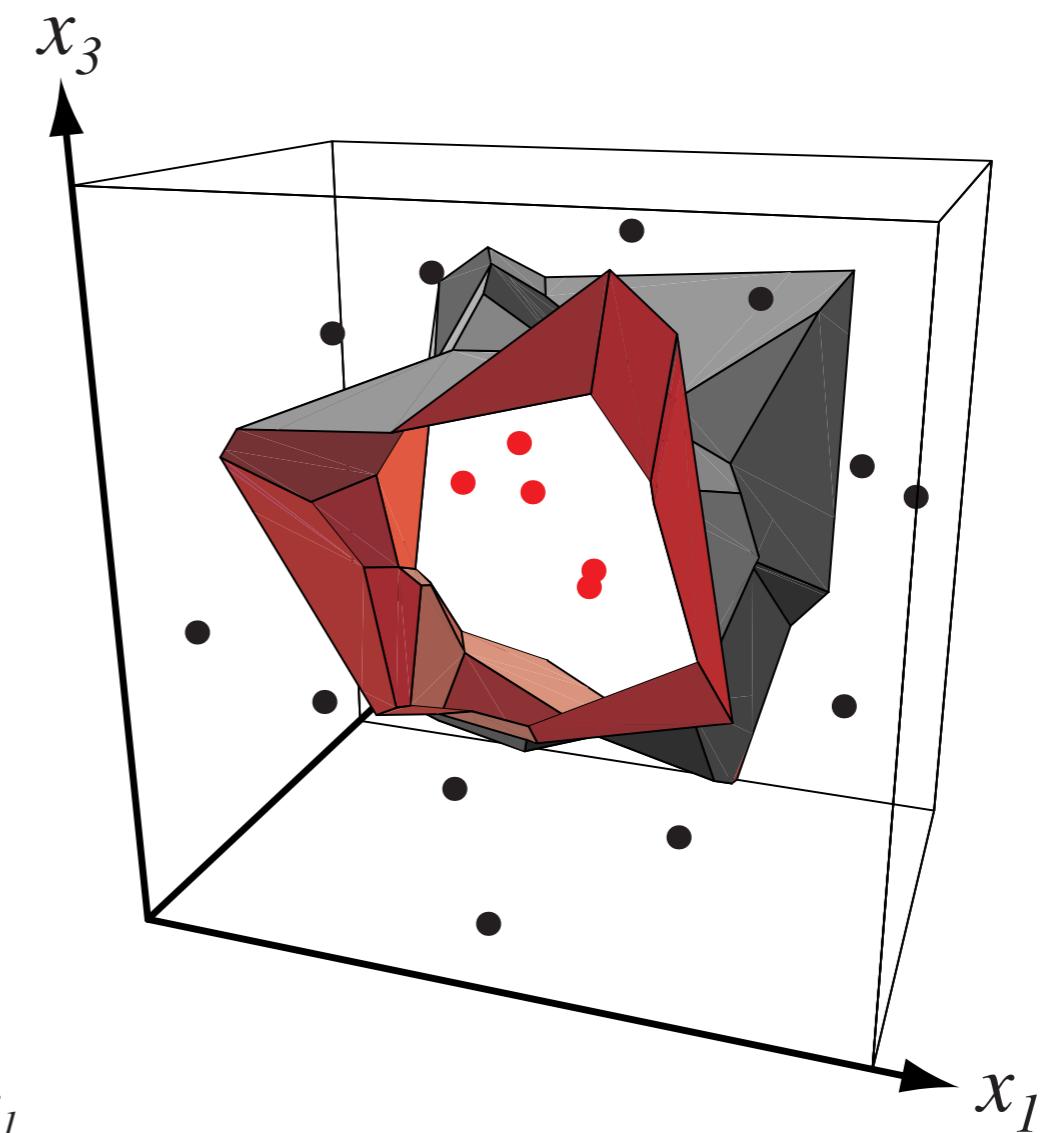
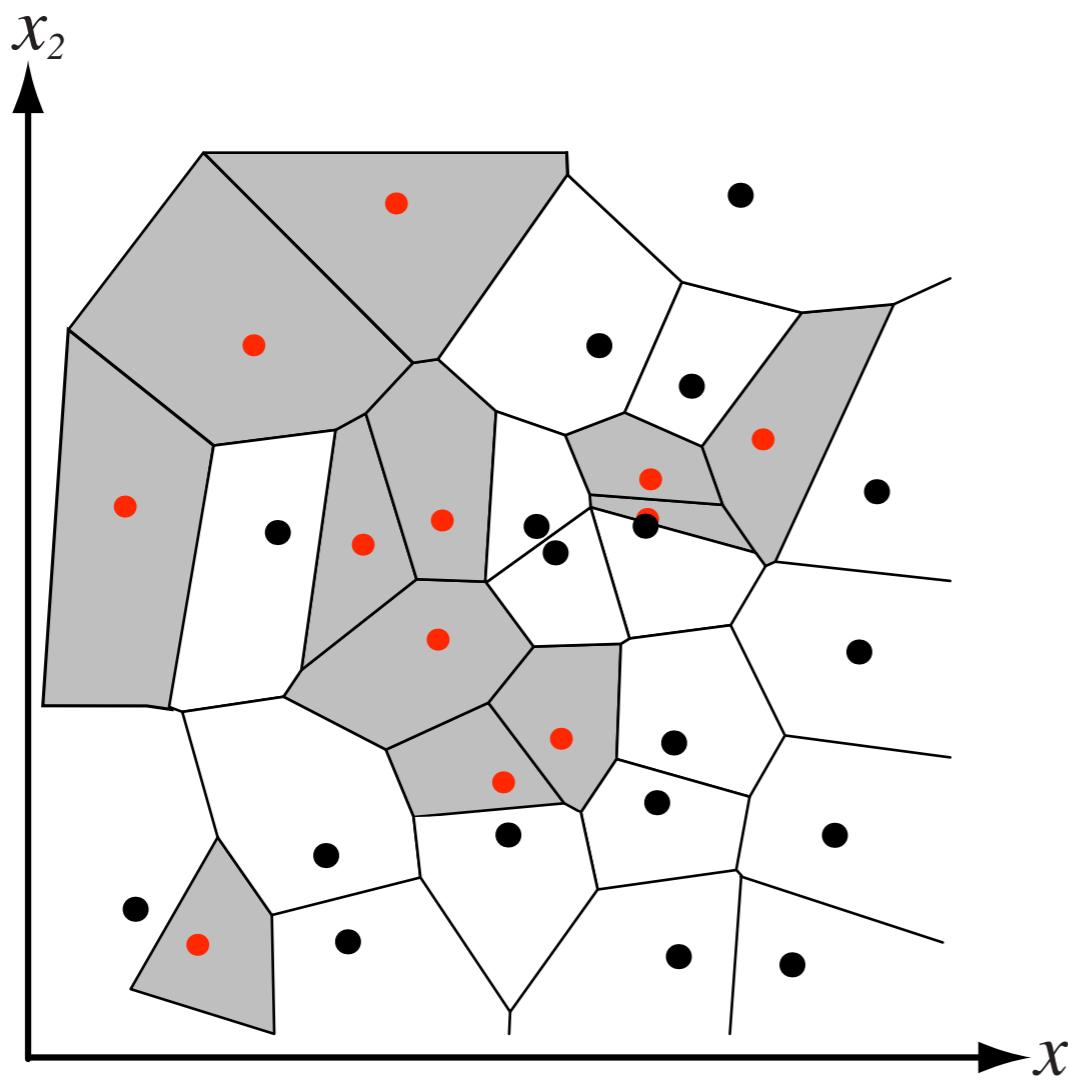
$$f(x) = \arg \max \left(\frac{1}{\sum_{i=1}^n w_i} \sum_{i=1}^n w_i \text{onehot}_m(Y_i) \right)$$

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

Régions et frontière de décision d'un classifieur de plus proche voisin

À quoi ressemblent-elles?

- Partition de Voronoi



k Plus Proches Voisins

Pseudo-code
et
Complexité Algorithmique



Voir le début de la démo

Fenêtres de Parzen

à voisinage dur

- Comme dans K-NN on fait voter les voisins
- Dans K-NN on prend les K voisins les plus proches
- Dans Parzen à voisinage dur, on prend tous les voisins situés en deçà d'une distance (rayon) h fixé.
- K et h sont des hyper-paramètres de ces algorithmes

Fenêtres de Parzen

à voisinage dur

Pour la classification binaire (avec $Y_i \in \{-1, 1\}$)

$V(x)$ = ensemble des points de l'ensemble d'apprentissage situés à moins d'une distance h de x .

signe de la moyenne des valeurs cibles des voisins de x situés à distance $\leq h$

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

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

avec

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

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

signe de la moyenne pondérée des valeurs cibles de tous les points d'entraînement, pondérées par un poids indiquant si le point d'entraînement est voisin de x .

Fenêtres de Parzen

à voisinage dur

Pour la régression (avec $Y_i \in \mathbb{R}$)

$V(x)$ = ensemble des points de l'ensemble d'apprentissage situés à moins d'une distance h de x .

la moyenne des valeurs cibles des voisins de x situés à distance $\leq h$

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

$$f(x) = \frac{1}{\sum_{i=1}^n w_i} \sum_{i=1}^n w_i Y_i \quad \text{avec} \quad w_i = I_{\{d(X_i, x) < h\}}$$

la moyenne pondérée des valeurs cibles de tous les points d'entraînement, pondérées par un poids indiquant si le point d'entraînement est voisin de x .

Fenêtres de Parzen

à voisinage mou

- Généralisation du voisinage «dur»:
- Plutôt que de faire voter un petit nombre de voisins on fait voter **tous** les points de l'ensemble.
- Mais il s'agit d'un **vote pondéré**:
plus le point est proche du point de test plus il compte
- C'est une fonction **noyau K** qui calcule les pondérations.
- K (et de ses paramètres) sont les hyper-paramètre de l'algorithme

Fenêtres de Parzen

à voisinage mou (soft)

Pour la régression (avec $Y_i \in \mathbb{R}$)

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

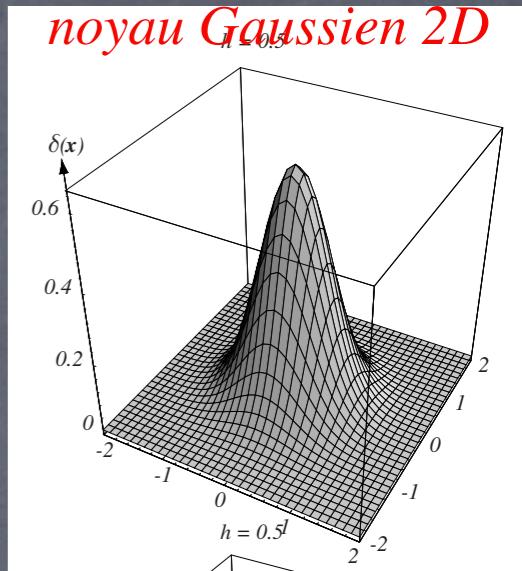
la moyenne **pondérée** des valeurs cibles de **tous** les points d'entraînement, pondérées par un poids indiquant à quel "degré" le point est voisin de x .

K est un noyau (Kernel)

notez que $w_i = I_{\left\{ \frac{d(X_i, x)}{h} < 1 \right\}}$ correspond à un K particulier
(un noyau "dur")

Comme noyau "mou" on choisit souvent un **noyau Gaussien** (correspond à une densité Normale)

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



Fenêtres de Parzen

(régression et classification en résumé)

Pour la régression ($Y_i \in \mathbb{R}$) :

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

Pour la classification binaire ($Y_i \in \{-1, 1\}$) :

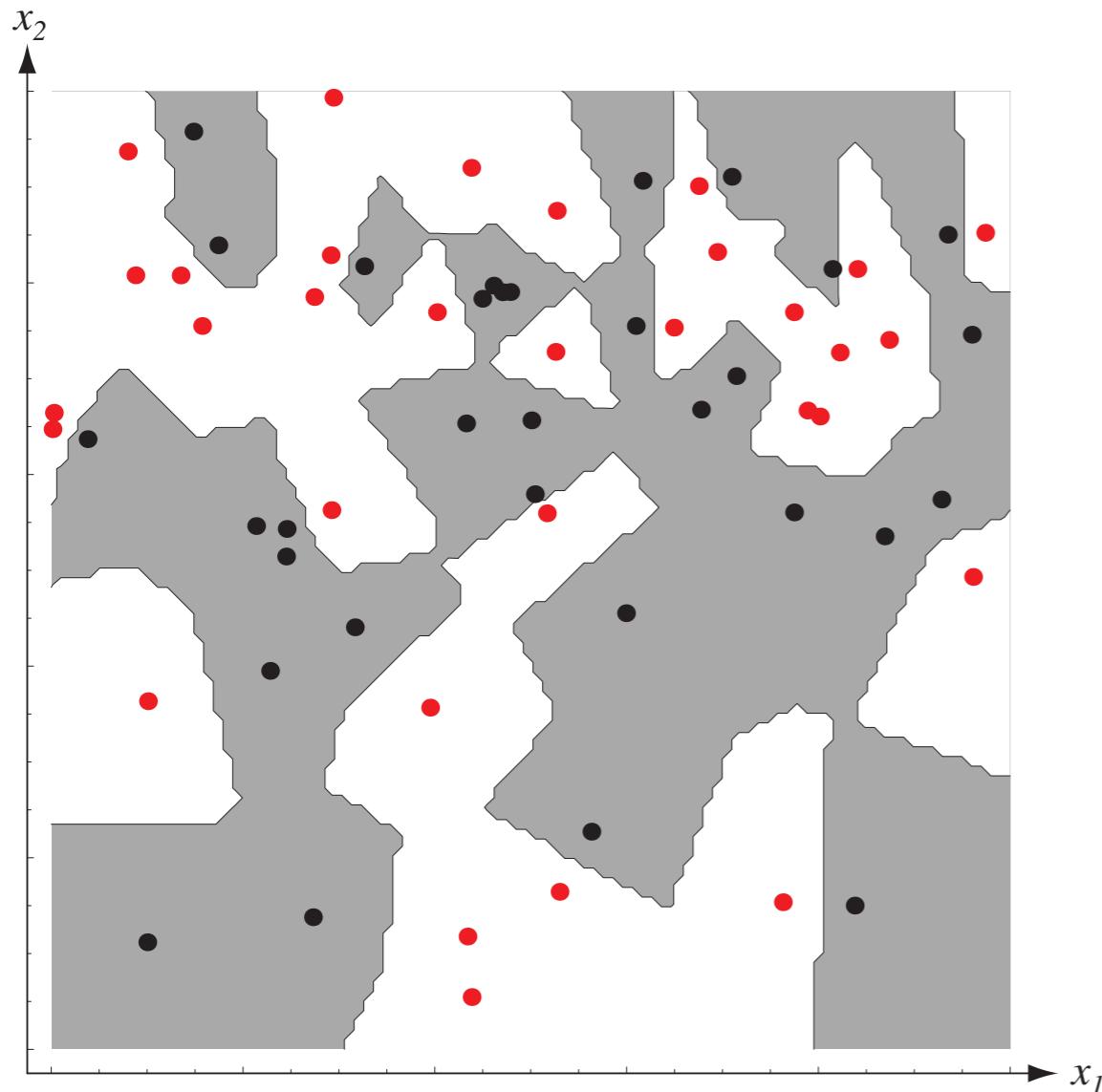
$$f(x) = \text{sign} \left(\frac{1}{\sum_{i=1}^n K(X_i, x)} \sum_{i=1}^n K(X_i, x) Y_i \right)$$

Pour la classification multiclasse ($Y_i \in 1 \dots m$) :

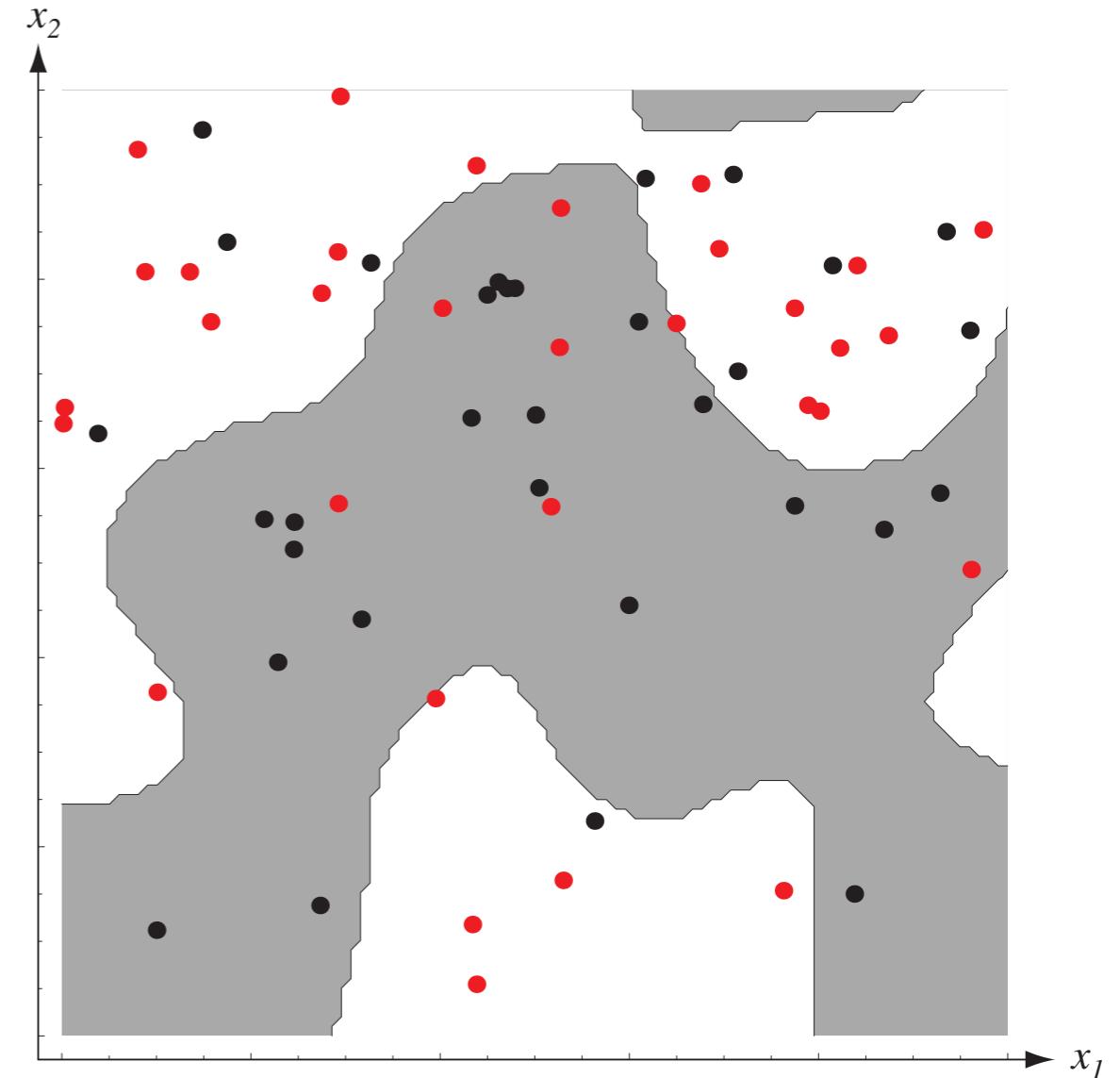
$$f(x) = \arg \max \left(\frac{1}{\sum_{i=1}^n K(X_i, x)} \sum_{i=1}^n K(X_i, x) \text{onehot}_m(Y_i) \right)$$

Exemple: Parzen pour classification 2D

σ plus petit



σ plus grand



En gris: région de décision de la classe noire

En blanc: région de décision de la classe rouge

Régression multiple

- **Régression multiple:** lorsque la cible n'est pas un unique scalaire réel mais un *vecteur* de m réels.
- On peut construire m régresseurs scalaires. (Ex: fenêtre de Parzen régression)
- On peut aussi adapter un algorithme pour manipuler des vecteurs plutôt que des scalaires.
- Ex, pour Parzen régression, ça revient au même: on peut le voir comme m régresseurs de Parzen qui font chacun la *moyenne pondérée de cibles scalaires*, ou comme *un* Parzen qui fait une *moyenne pondérée de vecteurs*.

Comment obtenir un classifieur avec un algo de régression

- Dans le cas binaire, on peut effectuer une régression avec des cibles $Y \in \{0, 1\}$
Un algorithme de régression parfait donnerait une prédiction
$$f(x) = E[Y|X = x] = P(Y = 1|X = x)$$
- Dans le cas de classification multiclasse (Y indique le numéro d'une classe parmi m classes), on peut effectuer une régression multiple avec une cible codée en “un parmi plusieurs” (one-hot) qui génère pour un y donné un **vecteur de dimension m** dont tous les éléments sont à 0 sauf l'élément y (correspondant à la varié classe) qui vaut 1.

Une régression multiclasse parfaite prédirait un vecteur

$$\begin{aligned} f(x) &= E[\text{onehot}_m(Y)|X = x] \\ &= (P(Y = 1|X = x), \dots, P(Y = m|X = x)) \end{aligned}$$

- Ainsi on peut dériver les algos de Parzen classification (binaire et multiclasse) des algos de Parzen régression (scalaire et multiple).

Comment obtenir un classifieur avec un algo de régression

Attention:

- On ne sait pas réaliser de régression “parfaite”.
- Plus particulièrement, la prédiction d'un régresseur pourrait pour certains algos être négative, ou > 1 .
- Aussi ces probabilités de classe ainsi obtenues (les estimés de $P(Y = j|X = x)$) ne sont pas garantis de sommer à 1.
- Cela peut néanmoins donner un bon algo de classification.

Fenêtres de Parzen

Pour l'estimation de densité de probabilité
(resp. de masse de probabilité) :

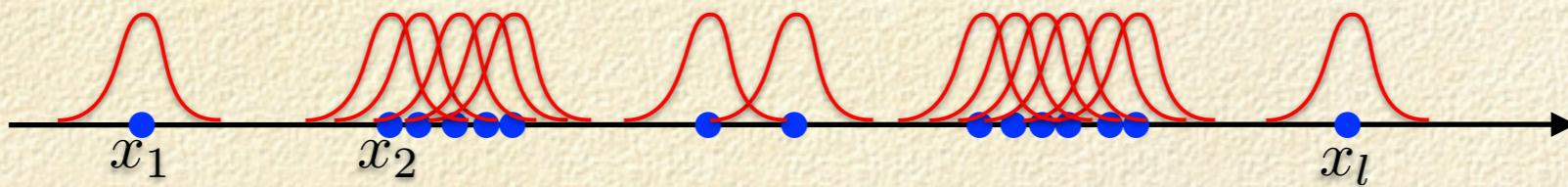
$$f(x) = \hat{p}(x) = \frac{1}{n} \sum_{i=1}^n K(X_i; x)$$

à condition que le noyau K soit bien une fonction de densité (ou de masse) de probabilité.

Ex: une Gaussienne centrée en X_i

L'estimateur de densité de Parzen produit alors une fonction de densité (ou de masse) de probabilité (qui intègre ou somme à 1)

Fenêtres de Parzen pour l'estimation de densité en 1D



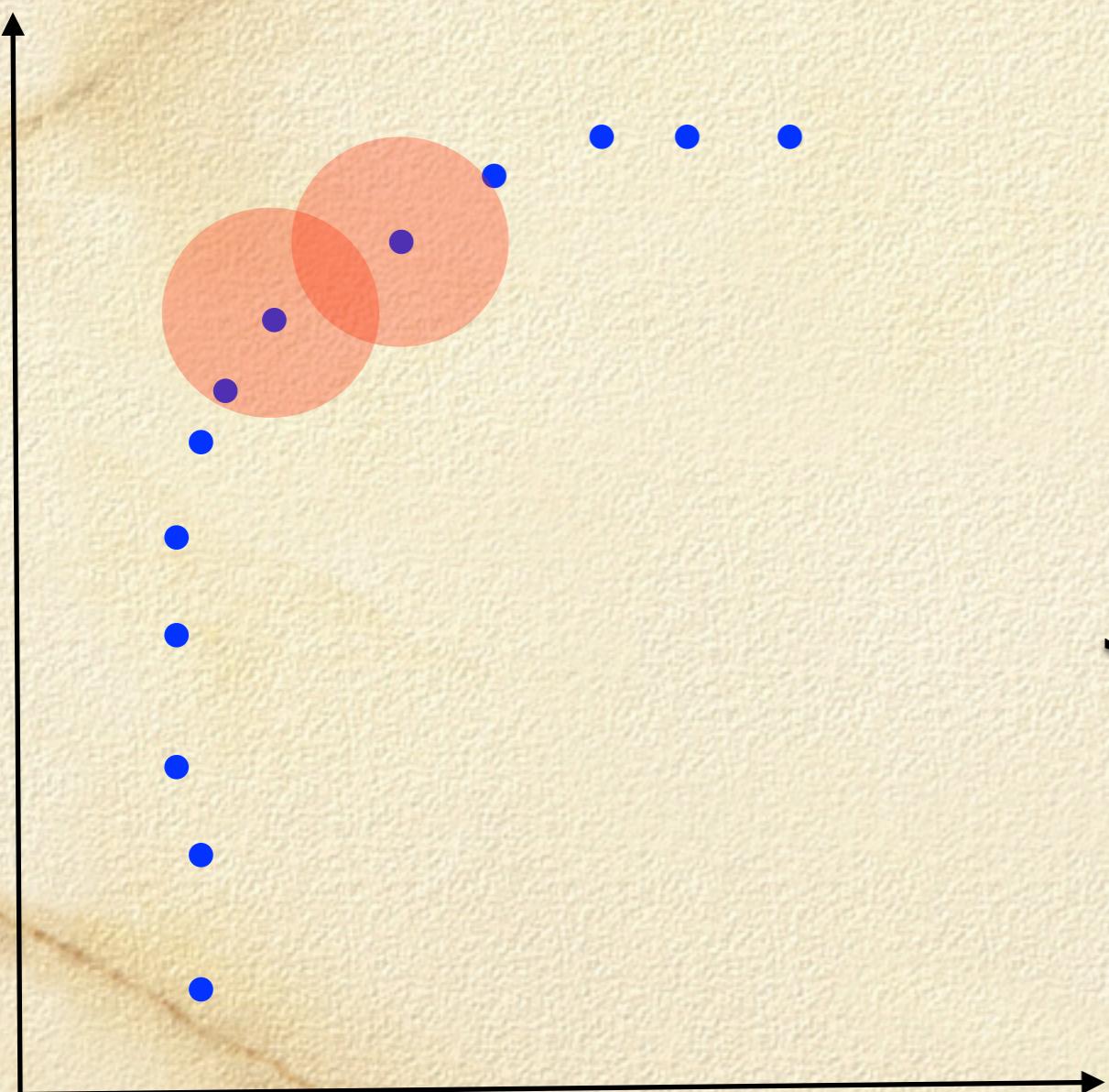
Gaussienne en dimension 1:

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

Estimateur de densité de Parzen:

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

Fenêtres de Parzen pour l'estimation de densité 2D

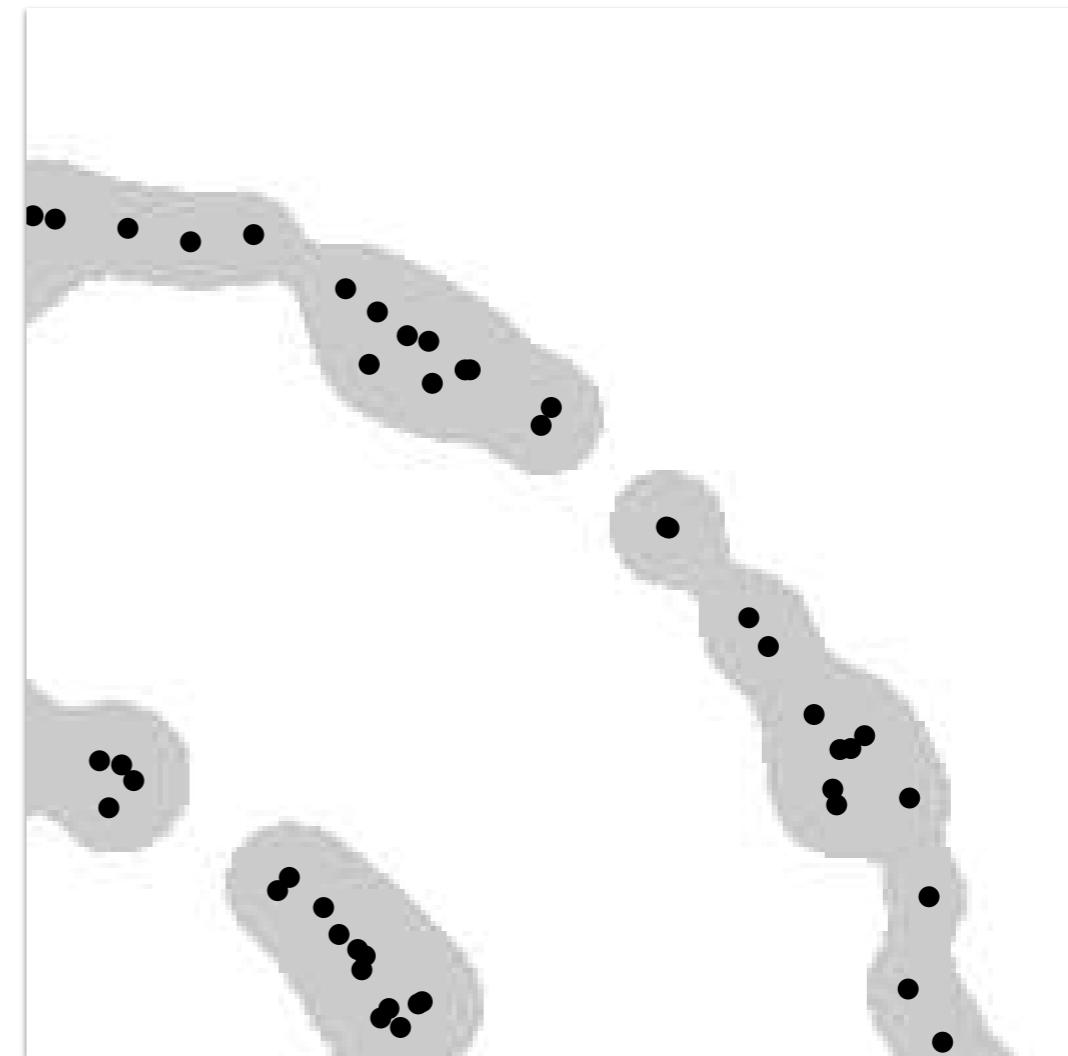


Gaussienne isotropique en dimension d:

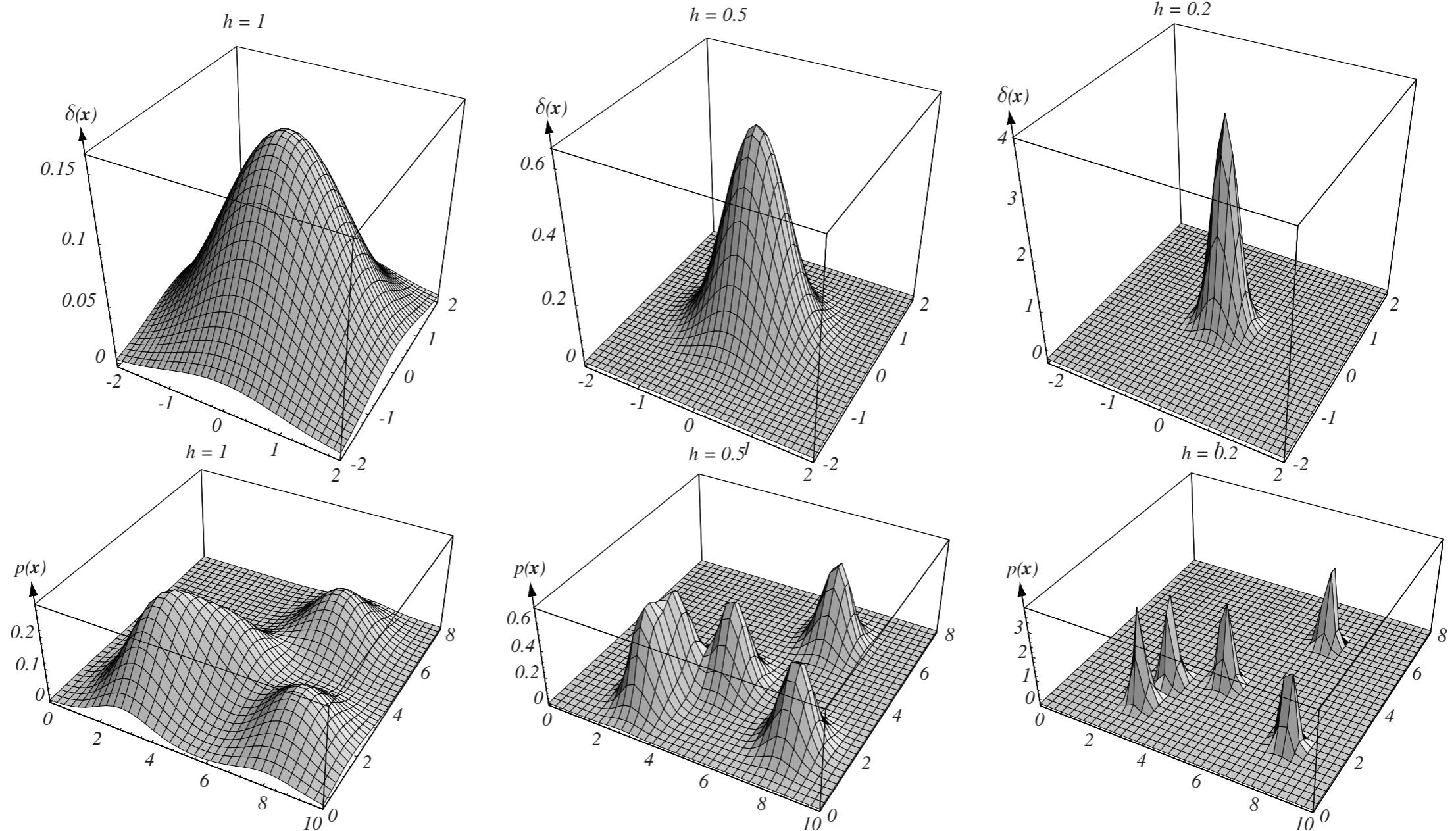
$$\mathcal{N}_{\mu,\sigma}(x) = \frac{1}{(2\pi)^{\frac{d}{2}} \sigma^d} e^{-\frac{1}{2} \frac{\|x-\mu\|^2}{\sigma^2}}$$

Exemple d'estimation de densité 2D

PARZEN WINDOWS

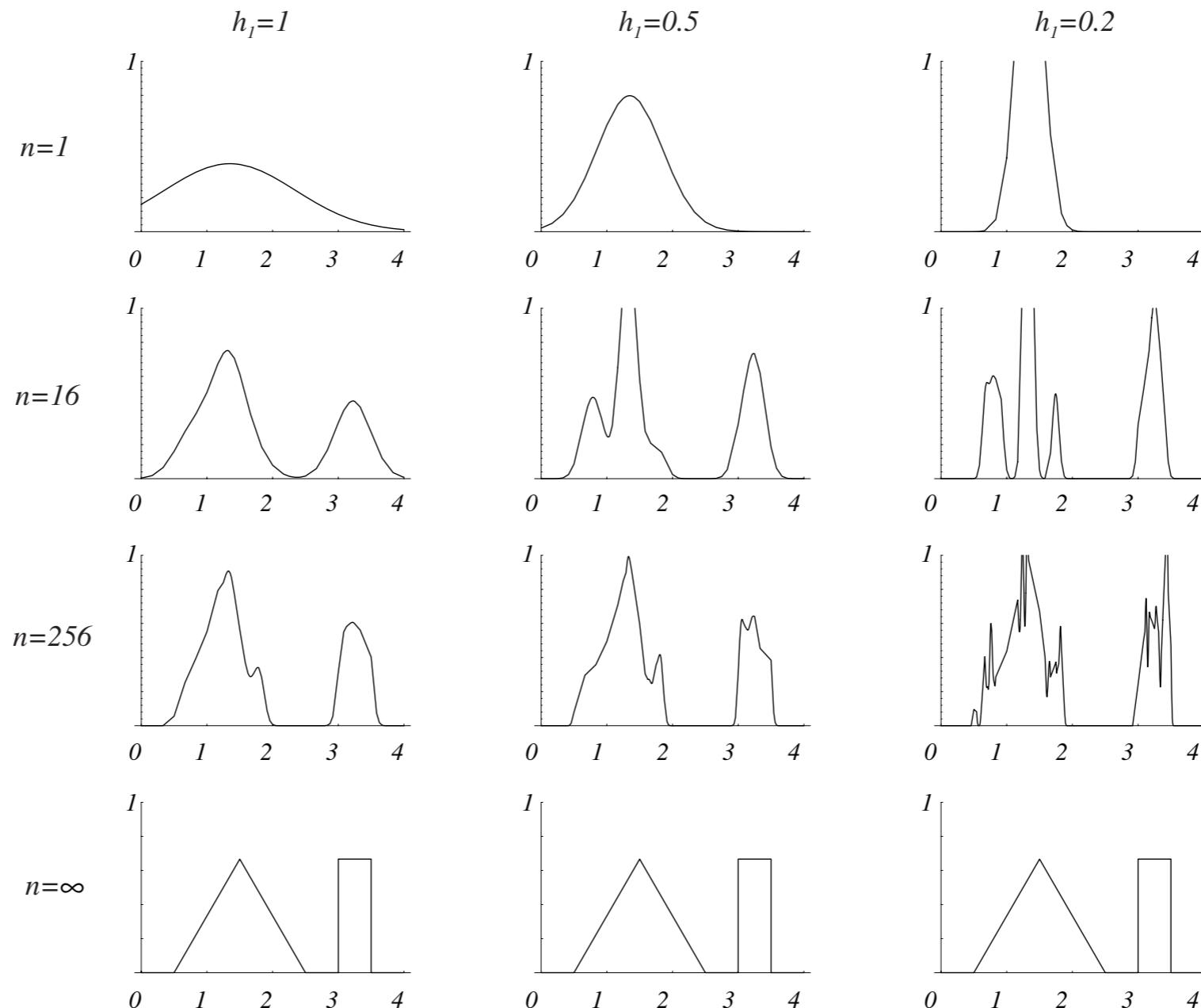


- L'effet de la largeur de fenêtre h_n
 Pour un noyau Gaussien isotropique $h=\sigma$



Exemple: Parzen pour estimation de densité 1D

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



Mesures de distance

- De nombreux algorithmes d'apprentissage sont basés sur une notion de «**distance**» entre points de l'espace d'entrée
- Nécessaire pour définir des «**voisinages**»
- Ex d'algos à base de voisinage: K-NN, Parzen

Distances = Métriques

- Propriétés d'une métrique

- positivité: $d(\mathbf{a}, \mathbf{b}) \geq 0$
- réflexivité: $d(\mathbf{a}, \mathbf{a}) = 0$
- symétrie: $d(\mathbf{a}, \mathbf{b}) = d(\mathbf{b}, \mathbf{a})$
- inégalité de triangle: $d(\mathbf{a}, \mathbf{b}) + d(\mathbf{b}, \mathbf{c}) \geq d(\mathbf{a}, \mathbf{c})$

Exemple:

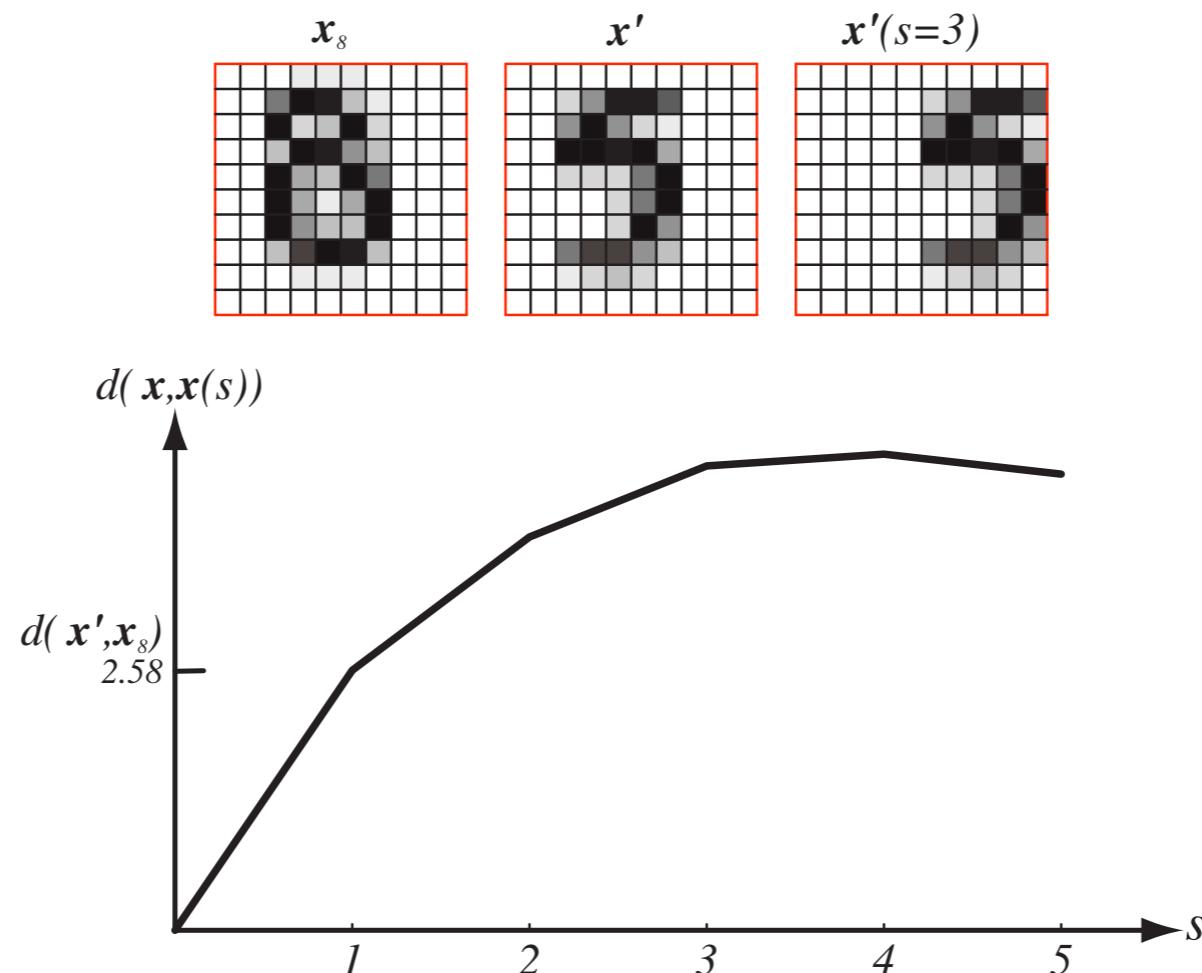
Distance Euclidienne (ou L2)

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

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

Métriques

- Les limitations de la métrique euclidienne

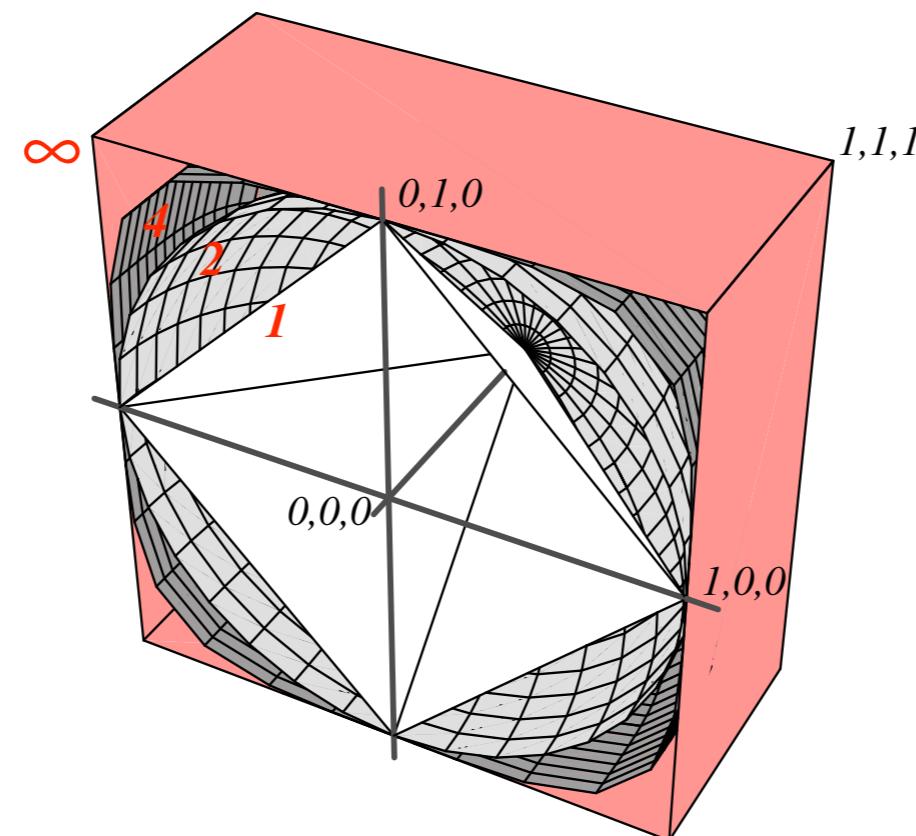


Métriques

- Exemples des métriques

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 a_i - b_i $
	L_∞	$d(\mathbf{a}, \mathbf{b}) = \max_i a_i - b_i $
Minkowski	L_p	$d(\mathbf{a}, \mathbf{b}) = \left(\sum_{i=1}^d a_i - b_i ^p \right)^{1/p}$
Tanimoto	L_{Tanimoto}	$d(S_1, S_2) = \frac{ S_1 + S_2 - 2 S_1 \cap S_2 }{ S_1 + S_2 - S_1 \cap S_2 }$

- La métrique de Minkowski



Il y a quantité d'autres mesures de distance!

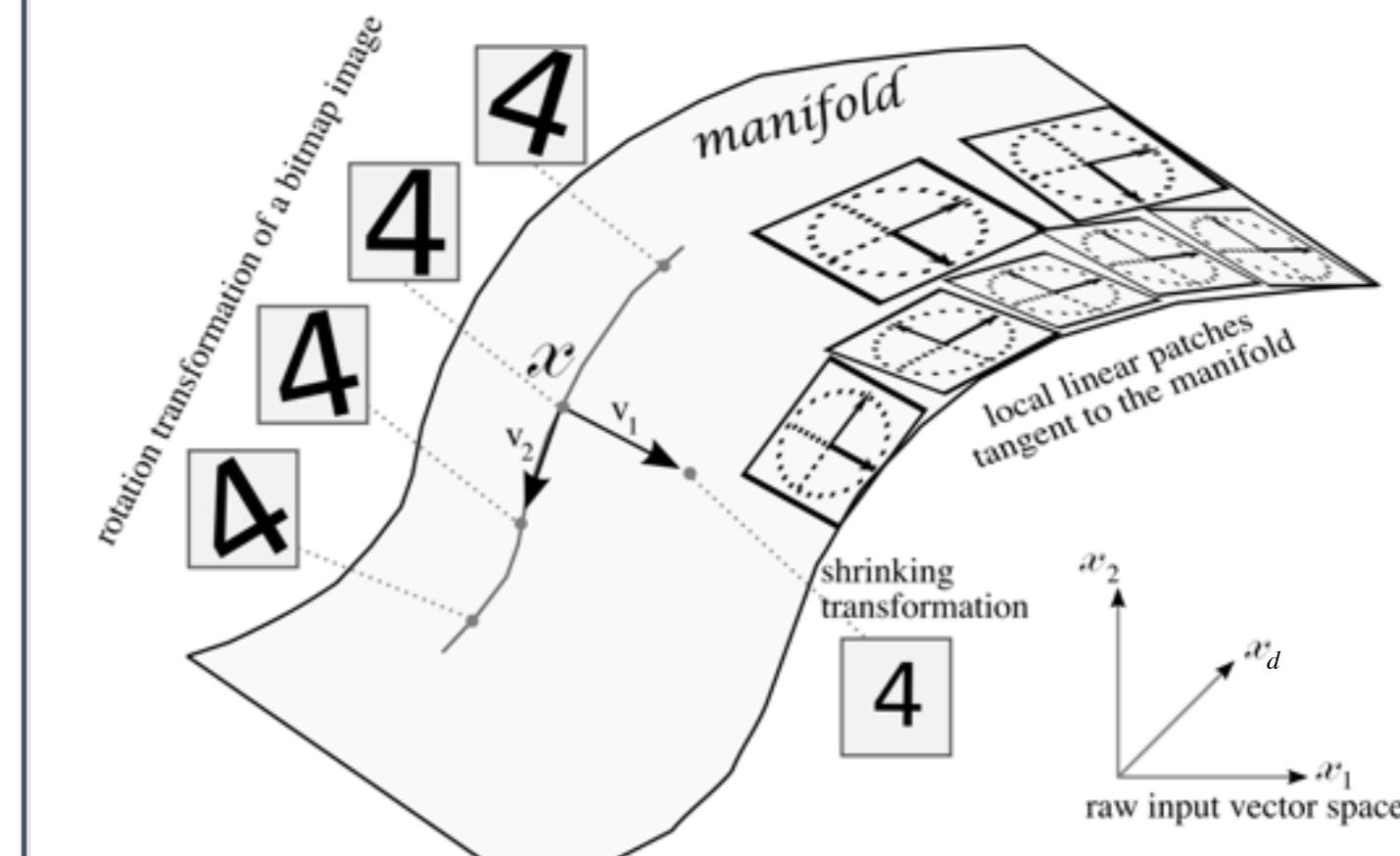
Notion de variété (manifold) de plus faible dimension

Ex: paramètre de pose d'un visage



Image borrowed from University of Dayton Vision Lab website.

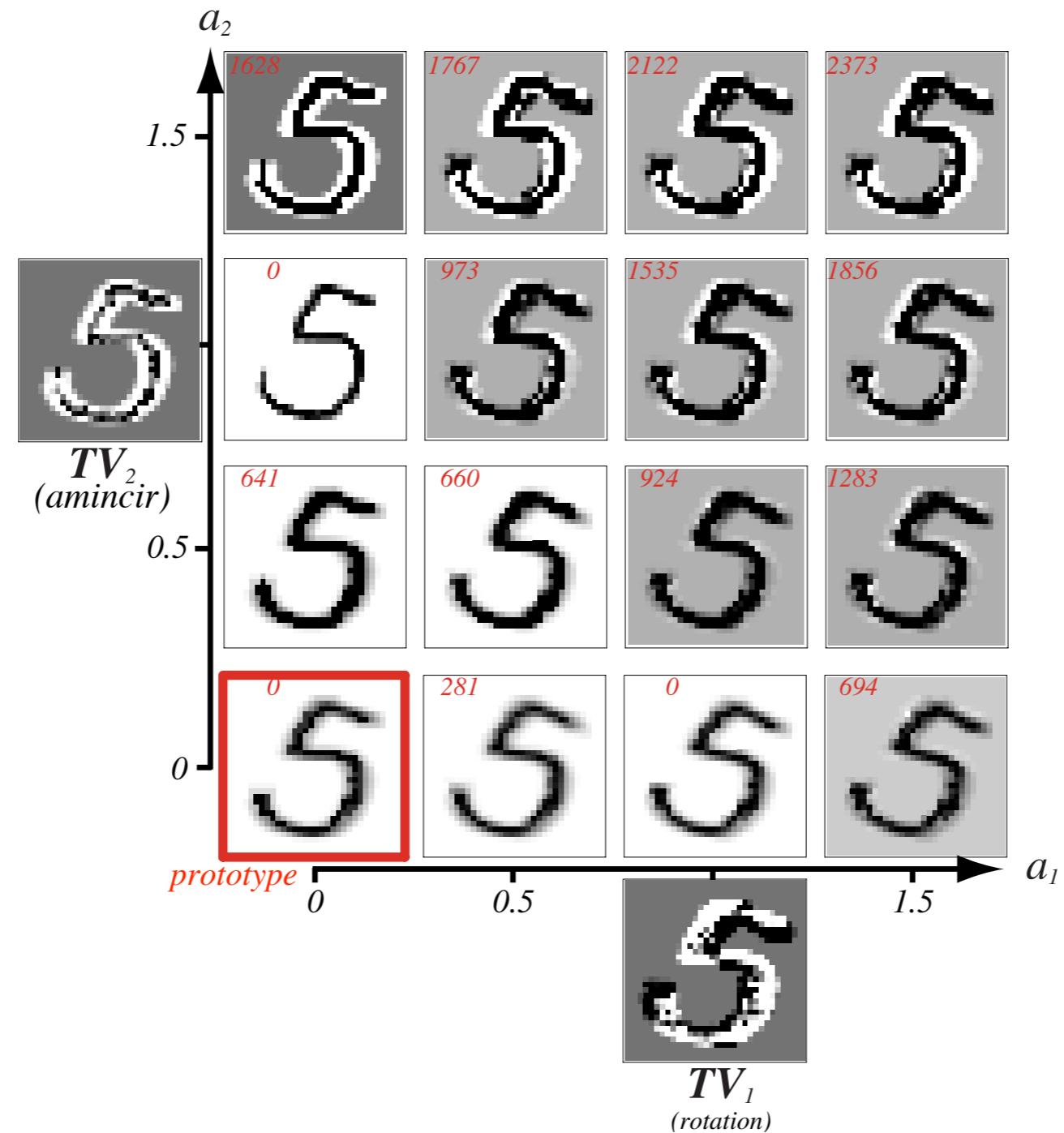
Ex: rotation, taille d'un caractère (+ line thickness, ...)



La distance tangente

- Capturer l'invariance de certaines transformations:

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



La distance tangente

$$d_{tan}(\mathbf{x}', \mathbf{x}) = \min_{\mathbf{a}} [\|(\mathbf{x}' + \mathbf{T}\mathbf{a}) - \mathbf{x}\|]$$

