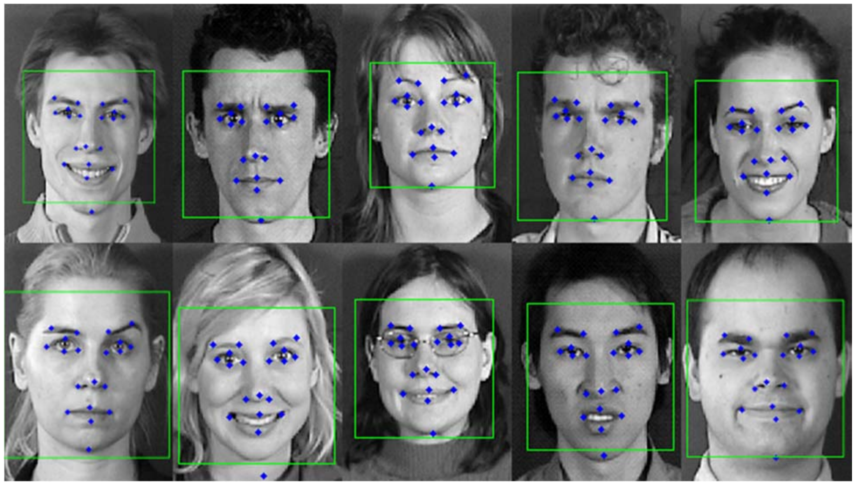
Learning from Data
4. Classification: Nearest Neighbor

Spring 2017

Classification

- Supervised (preditive) learning: given examples with labels, predict labels for all unseen examples
  - Classification:
    - label = category,
    - $\mathbf{y} \in \mathcal{Y} = \{1, \ldots, m\}$, m = number of classes
    - $\ell(\mathbf{x}, y, h) = 1(h(\mathbf{x}) \neq y)$, Risk = $P(Y \neq h(\mathbf{X})) = P(\text{Error})$



$\mathbf{x}$ = facial geometry features

y = gender label

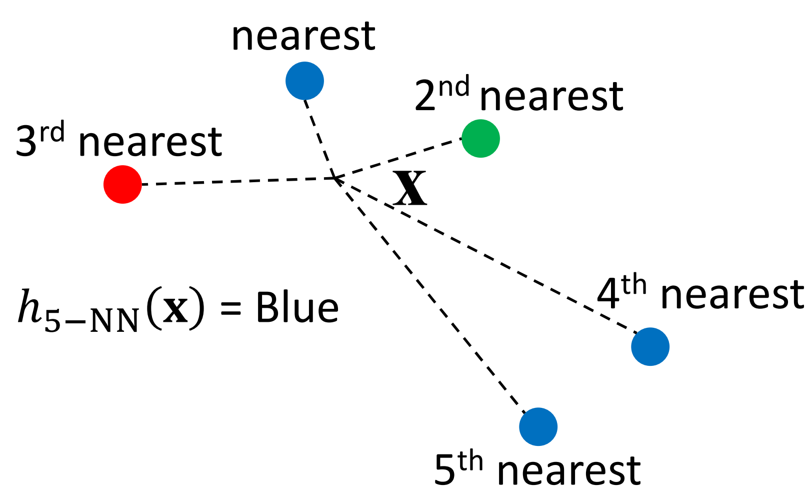
k-Nearest Neighbor (NN) Classifier

- Discriminative classifier: only $p(y|\mathbf{x})$ estimated
- Non-parametric: no parametric model for $p(y|\mathbf{x})$
- Example of memory-based learning
- Need 2 ingredients to specify classifier:
  1. k : number of nearest neighbors, typically chosen to be not a multiple of m (the number of classes)
  2. $d: \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$, a measure of "nearness" $d(\mathbf{x}, \mathbf{x}')$ between $\mathbf{x}$ and $\mathbf{x}$'

  – typically, d is chosen to be a metric, i.e., it is positive, definite, i.e., = 0 if, and only if, $\mathbf{x} = \mathbf{x}'$, symmetric, i.e., $d(\mathbf{x}, \mathbf{x}') = d(\mathbf{x}', \mathbf{x})$, and satisfies the triangle inequality, i.e., $d(\mathbf{x}_1, \mathbf{x}_3) \leq d(\mathbf{x}_1, \mathbf{x}_2)+d(\mathbf{x}_2, \mathbf{x}_3)$ for any three points

k-Nearest Neighbor Classifier

- Classifier description in words: $h_{k-\mathrm{NN}}(\mathbf{x})$ = most abundant label (majority vote) among the labels of the k nearest training examples of $\mathbf{x}$ (breaking ties in some way)



k-Nearest Neighbor Classifier

- Formal description: Given labeled training data $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ and a test point $\mathbf{x}$

- Let $\{(\mathbf{x}_{(1)}, y_{(1)}), \ldots, (\mathbf{x}_{(n)}, y_{(n)})\}$ be a re-ordering of training data such that

$$d(\mathbf{x}, \mathbf{x}_{(1)}) \leq d(\mathbf{x}, \mathbf{x}_{(2)}), \ldots, d(\mathbf{x}, \mathbf{x}_{(n)})$$

- Then,

$$h_{k-\mathrm{NN}}(\mathbf{x}) = \arg \max_{y=1,\ldots,m} \underbrace{\sum_{j=1}^{k} 1\left(y_{(j)} = y\right)}_{\text{number of } k \text{ NNs of } \mathbf{x} \text{ with label } = y}$$

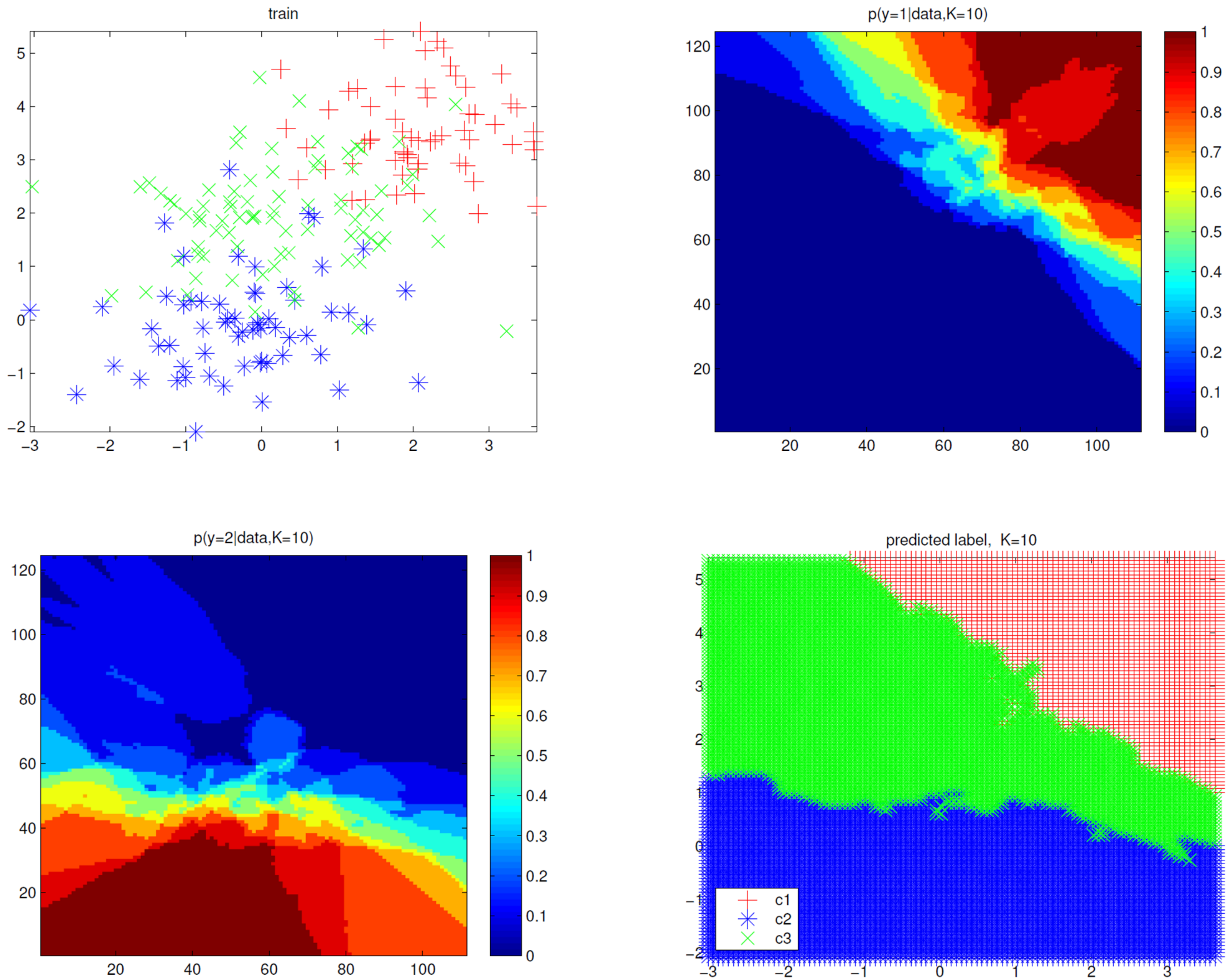
k-Nearest Neighbor Classifier

- Discriminative model based interpretation:

$$p_{k-\mathrm{NN}}(y|\mathbf{x}) = \underbrace{\frac{1}{k}\sum_{j=1}^{k} 1\big(y_{(j)} = y\big)}_{\text{fraction of } k \text{ NNs of } \mathbf{x} \text{ with label } = y}$$

- This is a non-parametric estimate of $p(y|\mathbf{x})$, since the number of parameters = n grows with training data

- k-NN classifier = MPE/MAP rule with the above non-parametric estimate of $p(y|\mathbf{x})$.


train
p(y=1|data,K=10)
p(y=2|data,K=10)
predicted label, K=10
c1
c2
c3

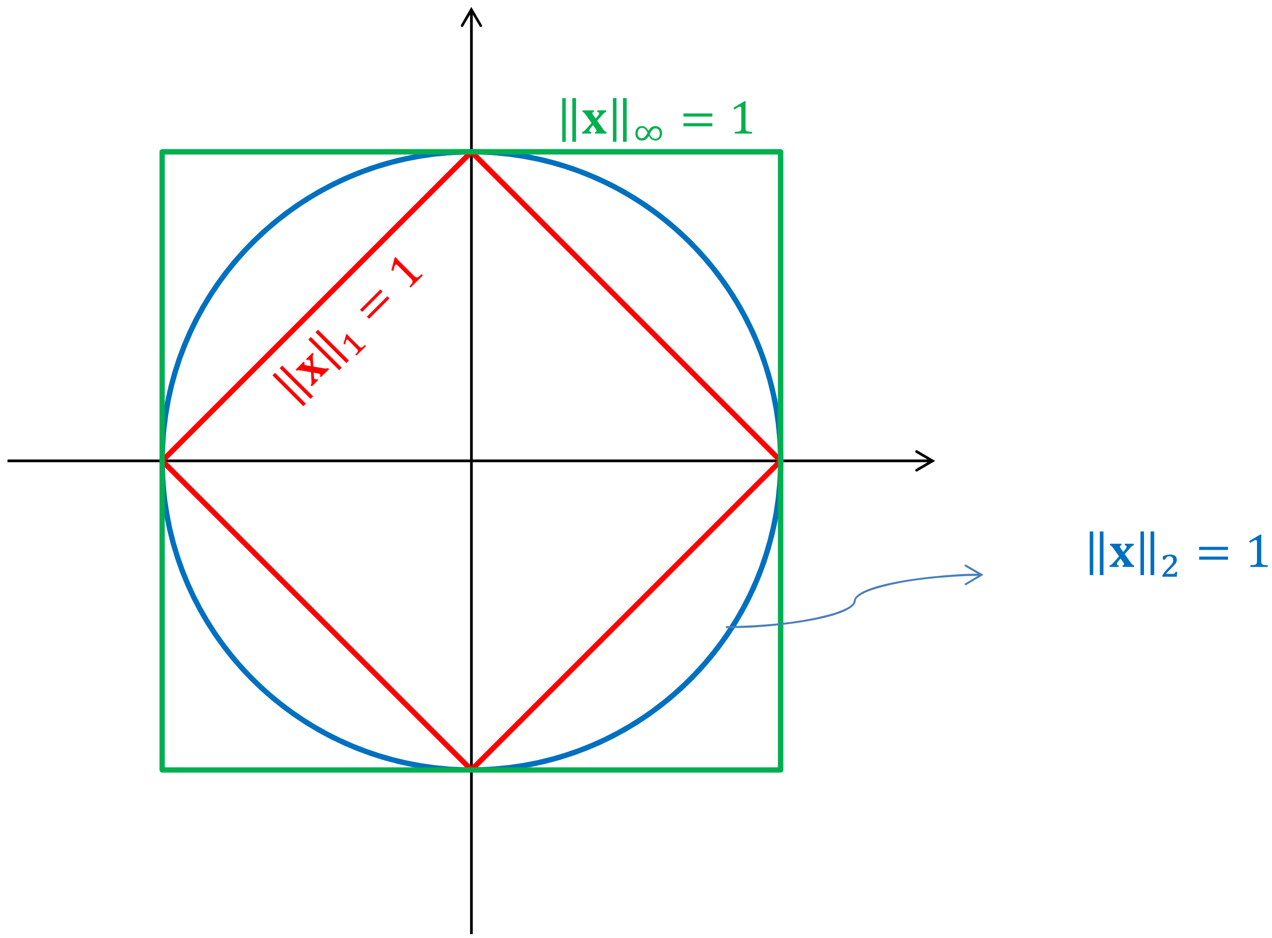
Common distance functions

- ℓ_p distance:

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

  – can prove that this is a norm-distance for all $p \geq 1$
- Euclidean or ℓ_2 distance: ℓ_p distance with $p = 2$
- taxi-cab or city-block or Manhattan or ℓ_1 distance: ℓ_p distance with $p = 1$
- max norm or ℓ_∞ distance: limit of ℓ_p as $p \to \infty$. Can be shown to be equal to: $\max_{1 \leq i \leq d} |x_i - x_i'|$
- Mahalanobis distance: $\sqrt[2]{(\mathbf{x} - \mathbf{x}')^T \Sigma^{-1} (\mathbf{x} - \mathbf{x}')}$ with Σ a positive definite square matrix

Common distance functions



Contours of constant p-norms
(distance from origin)
in 2 dimensions (d=2) for different p

Voronoi regions for 1-NN classifiers

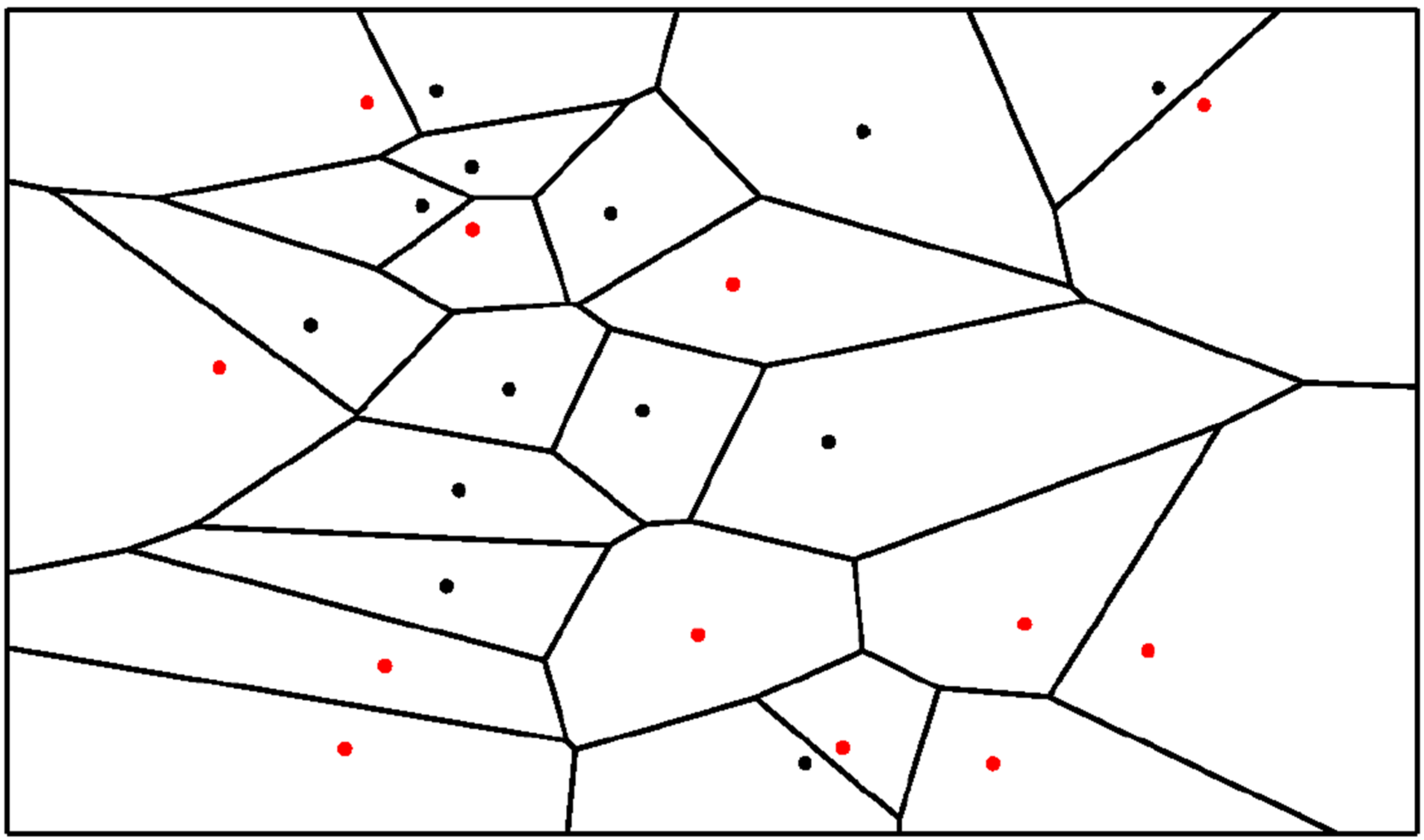
- Voronoi region of a training point $\mathbf{x}_j$ = all points in feature space that are closer to $\mathbf{x}_j$ than to any other training point:

$$\mathcal{V}(\mathbf{x}_j) := \{\mathbf{x} : d(\mathbf{x}, \mathbf{x}_j) \leq d(\mathbf{x}, \mathbf{x}_{j'}), \forall j' \neq j\}$$

- $\Rightarrow$The 1-NN classifier will assign all points in $\mathcal{V}(\mathbf{x}_j)$ the same class label as that of $\mathbf{x}_j$
- The Voronoi tessellation is a partition of the feature space into the Voronoi regions of the training points
- If the distance function is a norm induced by an inner product, e.g.,$\|\boldsymbol{x}\| = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle}$, then the Voronoi regions are convex, i.e., for any two points in the region, the entire line segment joining them is also entirely in the region.

Voronoi regions for 1-NN classifiers

- Voronoi tessellation of the 2-dim plane induced by the 1-NN classifier based on Euclidean distance

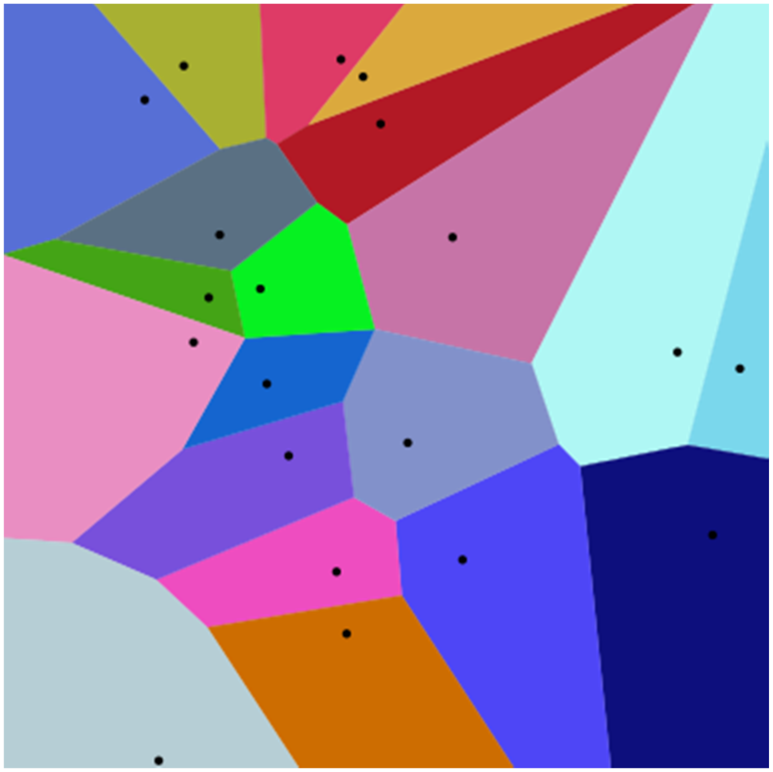


Voronoi regions for 1-NN classifiers

- Voronoi tessellation based on:

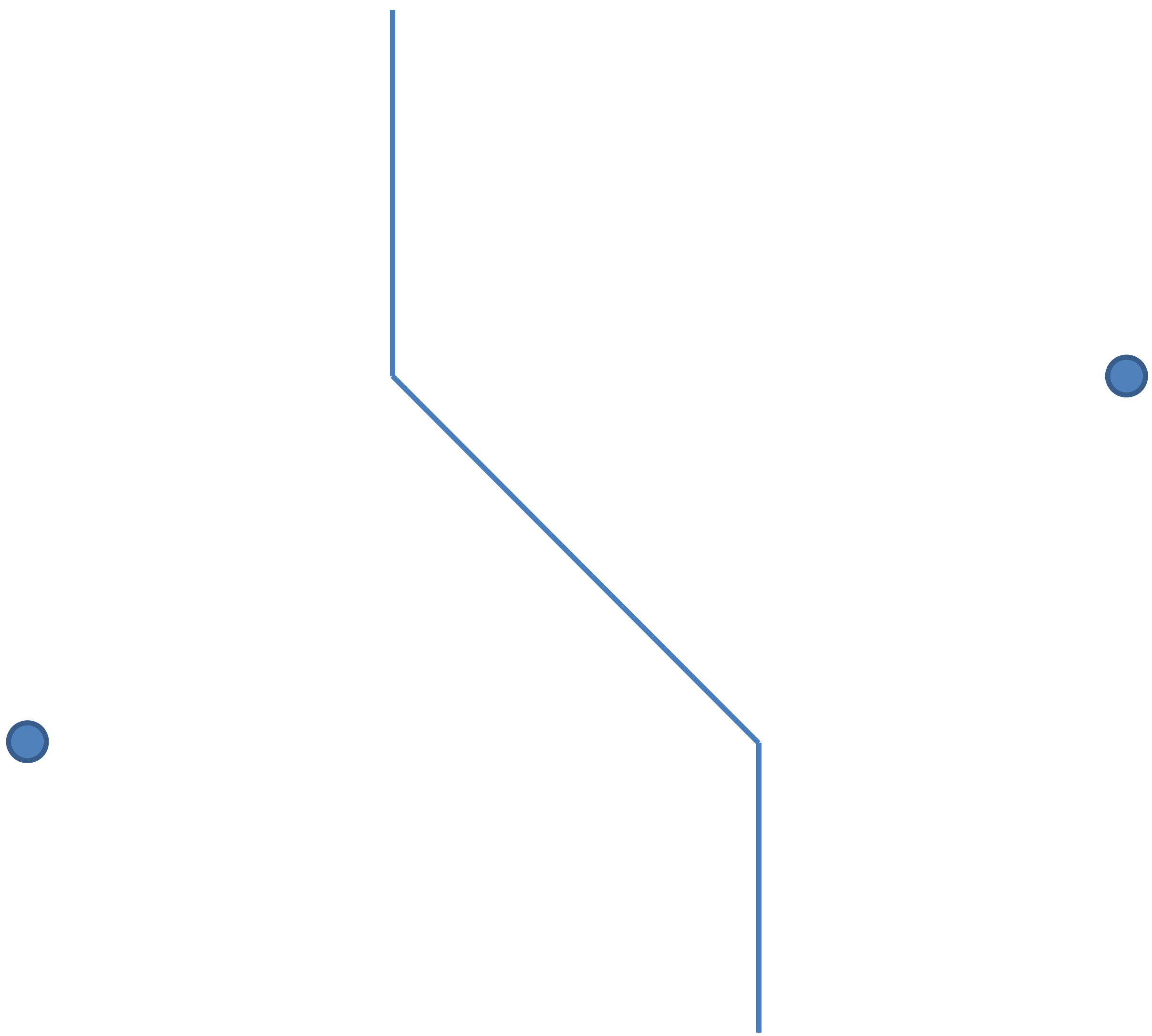
Euclidean distance versus Manhattan distance

(convex) (non-convex)



Voronoi regions for 1-NN classifiers

- 2-point Voronoi diagram for Manhattan distance



Remarks

- Asymptotic performance guarantees:
  - as $n \to \infty$, risk of 1-NN classifier (for 0-1 loss) $\leq 2R_{\text{Bayes}}$ for all "nice" data distributions and distance functions!
  - If as $n \to \infty$,
    - $k_n \to \infty$ …(ensures more training examples in majority vote)
    - $\frac{k_n}{n} \to 0$ ... (ensures the k_n NNs get closer to any test point)

    then, risk of k_n–NN classifier (for 0-1) loss $\to R_{\text{Bayes}}$ for all "nice" data distributions and distance functions!!

- In practice, the best value of k is determined via cross-validation

Remarks

- Useful observation: Euclidean distances can be computed via inner products: $\|\mathbf{x}-\mathbf{x}'\|_2^2 = \langle \mathbf{x},\mathbf{x}\rangle + \langle \mathbf{x}',\mathbf{x}'\rangle - 2\langle \mathbf{x},\mathbf{x}'\rangle$
- Let $\mathrm{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]$, $\mathrm{diag}(A)$ = column vector of the main diagonal elements of square matrix A, and $\mathbf{1}$ = column vector of all ones. Then the square Euclidean distance matrix (EDM) is given by:

$$\mathrm{EDM}(\mathrm{X}) = \mathbf{1} * \mathrm{diag}\left(\mathrm{X}^{\mathrm{T}}\mathrm{X}\right)^{\mathrm{T}} - 2\mathrm{X}^{\mathrm{T}}\mathrm{X} + \mathrm{diag}\left(\mathrm{X}^{\mathrm{T}}\mathrm{X}\right) * \mathbf{1}^{\mathrm{T}}$$

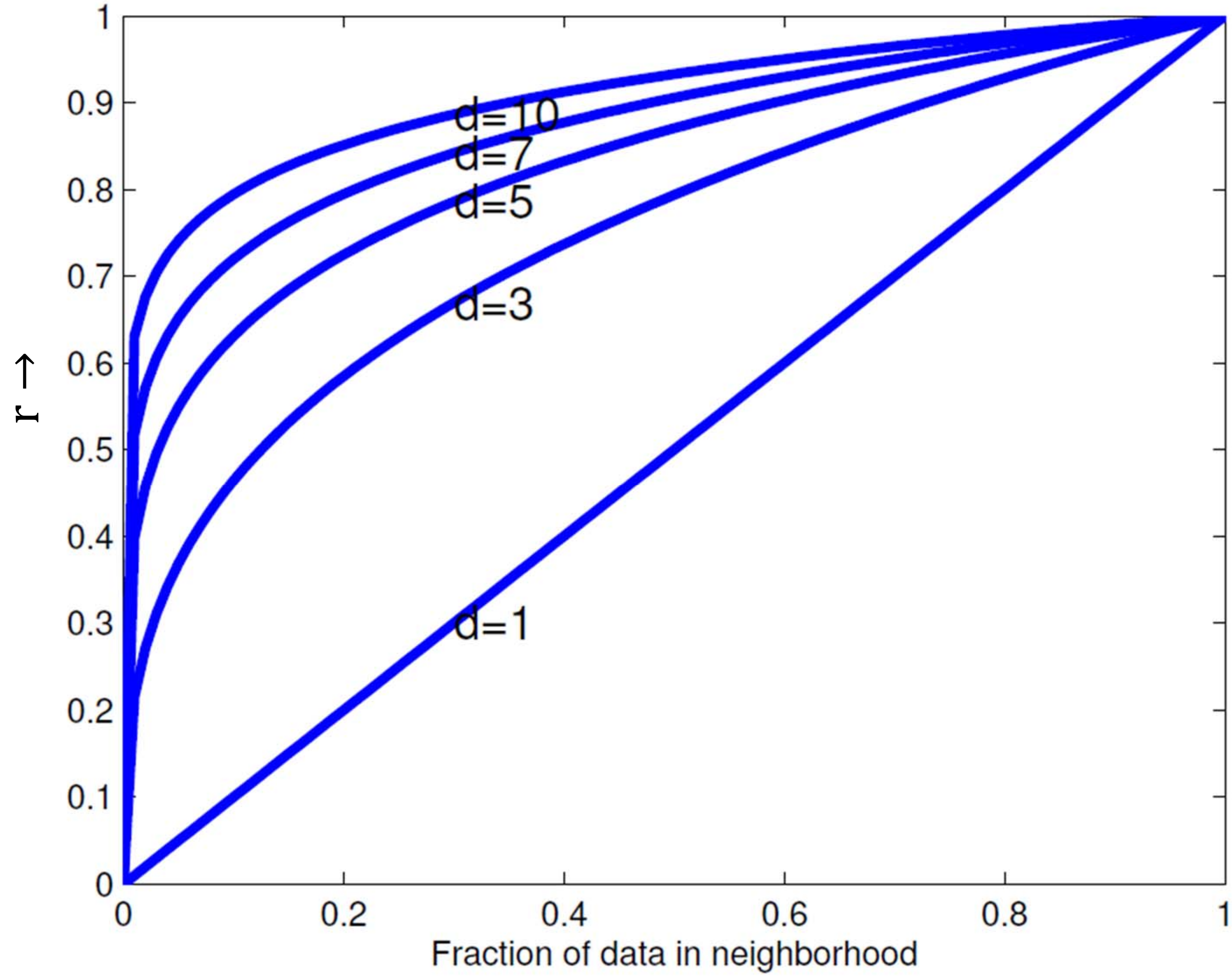
$$\mathrm{EDM}(\mathrm{X})(i,j) = \left\|\mathbf{x}_i - \mathbf{x}_j\right\|_2^2$$

- Note: $\mathrm{rank}\left(\mathrm{EDM}(\mathrm{X})\right) \le d + 2$

Curse of dimensionality

- NN classifiers break down if d is large
- Suppose training points uniformly distributed in d-dimensional unit sphere centered at origin
- To "capture" a fraction f of all training points, the distance r from the origin that one needs to travel is given by: $\frac{r^d}{1^d} = f \Rightarrow r = f^{\frac{1}{d}}$
- Say $d = 10$ (modest).
  - $f = 0.1 \Rightarrow r = 0.8$, i.e., to capture 10% of samples, need to traverse 80% of range! Samples that are so far from a point (non-local) are typically not good predictors of behavior at that point
  - $f = 0.01 \Rightarrow r = 0.63$, i.e., to capture only 1% of samples, still need to traverse 63% of range!
  - Seen another way, to keep decision making local, if we fix $r = 0.1 \Rightarrow$ number of points used for classification = $nr^d = n10^{-10} \Rightarrow$ will need a HUMONGOUS amount of training data to make a reliable decision

Curse of dimensionality



- Turns out that in high dimensions, most points are far from each other!
- Also turns out that in high dimensions, most points are almost orthogonal to each other!! …Weird things happen in high dimensions