Aprenentatge Automàtic 1

GCED

Lluís A. Belanche

belanche@cs.upc.edu

Soft Computing Research Group
Dept. de Ciències de la Computació (Computer Science)

Universitat Politècnica de Catalunya

2019-2020

LECTURE 6: Classification theory and linear classification models (II)

Generative Bayesian classifiers

Discriminant functions for the Gaussian density

- We showed that the Bayes rule minimizing the probability of error could be formulated in terms of a family of **discriminant functions**:

"assign the feature vector $\boldsymbol{x}$ to class ω_k whenever $g_k(\boldsymbol{x})$ is the largest, $1 \leq k \leq K$"

- When $\boldsymbol{X}_{|\Omega=\omega_k} \sim \mathcal{N}(\boldsymbol{\mu}_k, \Sigma_k)$, this family can be reduced to very simple expressions.

  Using Bayes rule and the natural log, the discriminant function for class ω_k becomes:

$$g_k(\boldsymbol{x}) = \ln\{P(\omega_k)p(\boldsymbol{x}|\omega_k)\} = \ln\, P(\omega_k) - \ln\,\left\{(2\pi)^{\frac{d}{2}}|\Sigma_k|^{\frac{1}{2}}\right\} - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^\top \Sigma_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)$$

  Eliminating constant terms:

$$g_k(\boldsymbol{x}) = \ln\, P(\omega_k) - \frac{1}{2}\Big(\ln\, |\Sigma_k| + (\boldsymbol{x}-\boldsymbol{\mu}_k)^\top \Sigma_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)\Big)$$

This expression is called a **quadratic discriminant function**, and the **decision boundaries** $g_i(\boldsymbol{x}) = g_j(\boldsymbol{x})$ are general hyper-quadrics in d-dimensional space

Generative Bayesian classifiers

Discriminant functions for the Gaussian density

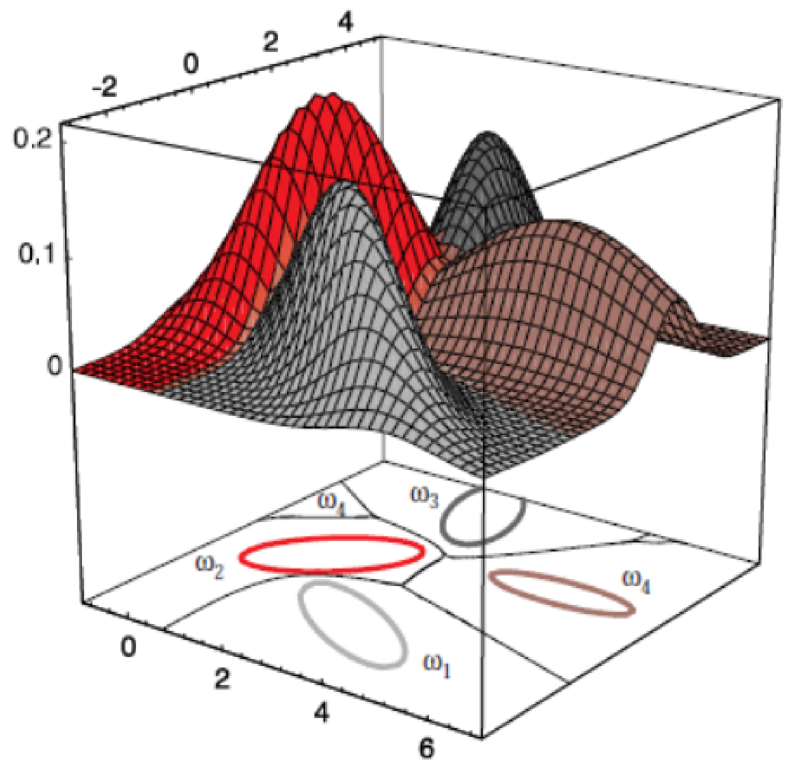


Figure 2.16 from Duda et al. book

Generative Bayesian classifiers

Discriminant functions for the Gaussian density

If we assume that all class-conditional distributions $p(\boldsymbol{x}|\omega_k)$ have the **same covariance** matrix Σ, after some polishing we get:

$$g_k(\boldsymbol{x}) = \ln P(\omega_k) + \boldsymbol{\mu}_k^\top \Sigma^{-1}\boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu}_k^\top \Sigma^{-1}\boldsymbol{\mu}_k$$

Reorganizing terms we obtain $g_k(\boldsymbol{x}) = \boldsymbol{w}_k^\top \boldsymbol{x} + w_{k0}$, where

$$\begin{aligned} \boldsymbol{w}_k &= \Sigma^{-1}\boldsymbol{\mu}_k \\ w_{k0} &= -\frac{1}{2}\boldsymbol{\mu}_k^\top \Sigma^{-1}\boldsymbol{\mu}_k + \ln P(\omega_k) \end{aligned}$$

These are **linear discriminant functions** (linear in $\boldsymbol{x}$) and the **decision boundaries** $g_i(\boldsymbol{x}) = g_j(\boldsymbol{x})$ are hyper-planes in d-dimensional space

Generative Bayesian classifiers

Discriminant functions for the Gaussian density

If we further assume that all X_i, X_j pairs are **statistically independent**, that is, $\Sigma = \text{diag}(\sigma_1^2, \ldots, \sigma_d^2)$, we get:

$$g_k(\boldsymbol{x}) = \ln P(\omega_k) - \frac{1}{2} \sum_{i=1}^{d} \frac{(\mu_{ki} - x_i)^2}{\sigma_i^2}$$

If we further assume that all X_i have the **same variance** σ^2, that is $\Sigma = \sigma^2 I_d$, we get:

$$g_k(\boldsymbol{x}) = \ln P(\omega_k) - \frac{1}{2\sigma^2} \|\boldsymbol{\mu}_k - \boldsymbol{x}\|^2$$

If we further assume that all classes have the **same prior** $P(\omega_k) = 1/K$, we get:

$$g_k(\boldsymbol{x}) = -\|\boldsymbol{\mu}_k - \boldsymbol{x}\|^2$$

Generative Bayesian classifiers

Discriminant functions for the Gaussian density

In all cases, we have a **minimum-distance** classifier in $\mathbb{R}^d$:

- In the general case (some covariance matrices are different), the classifier uses a different Mahalanobis distance (a fully-weighted Euclidean distance) from $\boldsymbol{x}$ to each class

  The technique is called **quadratic discriminant analysis** (QDA)

- In case all covariance matrices are equal, the classifier uses the same Mahalanobis distance from $\boldsymbol{x}$ to all classes

  The technique is called **linear discriminant analysis** (LDA)

- In case all covariance matrices are diagonal, the classifier uses a simply-weighted Euclidean distance from $\boldsymbol{x}$ to all classes

- In case all covariance matrices are a multiple of the identity matrix, the classifier uses an unweighted Euclidean distance from $\boldsymbol{x}$ to all classes

Generative Bayesian classifiers

A numerical example (I)

Derive a linear discriminant function for the two-class classification problem defined by the following Gaussian class-conditional densities:

$$\boldsymbol{\mu}_1 = (0,0,0)^\top, \boldsymbol{\mu}_2 = (1,1,1)^\top, \Sigma_1 = \Sigma_2 = \mathrm{diag}\left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right), P(\omega_2) = 2P(\omega_1)$$

Solution: since all the X_i, X_j are statistically independent $(i \neq j)$ and have the same variance $\sigma^2 = \frac{1}{4}$, that is $\Sigma = \frac{1}{4}I$, we get:

$$g_1(\boldsymbol{x}) = \ln\, P(\omega_1) - \frac{1}{2}\frac{\|\boldsymbol{\mu}_1 - \boldsymbol{x}\|^2}{\sigma^2} = \ln\, \frac{1}{3} - \frac{1}{2}\frac{\|(0,0,0) - \boldsymbol{x}\|^2}{\frac{1}{4}}$$

$$g_2(\boldsymbol{x}) = \ln\, P(\omega_2) - \frac{1}{2}\frac{\|\boldsymbol{\mu}_2 - \boldsymbol{x}\|^2}{\sigma^2} = \ln\, \frac{2}{3} - \frac{1}{2}\frac{\|(1,1,1) - \boldsymbol{x}\|^2}{\frac{1}{4}}$$

Generative Bayesian classifiers

A numerical example (II)

Then we can build an optimal **dichotomizer**:

$$g(\boldsymbol{x}) = g_1(\boldsymbol{x}) - g_2(\boldsymbol{x}) \underset{\underset{<}{\omega_2}}{\overset{\omega_1}{>}} 0, \quad \boldsymbol{x} = (x_1, x_2, x_3)^\top$$

$$g(\boldsymbol{x}) = -\ln 2 - 2\|\boldsymbol{x}\|^2 + 2\|(1,1,1) - \boldsymbol{x}\|^2 \underset{\underset{<}{\omega_2}}{\overset{\omega_1}{>}} 0$$

Which results in: $(x_1 + x_2 + x_3) \underset{\underset{<}{\omega_1}}{\overset{\omega_2}{>}} \frac{3}{2} - \frac{1}{4} \ln 2$

The **prediction** for the test example $\boldsymbol{x}^* =$(0.1, 0.7, 0.8)$^\top$ is $\boldsymbol{x}^* \in \omega_2$, given that $0.1 + 0.7 + 0.8 = 1.6 > \frac{3}{2} - \frac{1}{4} \ln 2 \approx 1.32$

Generative Bayesian classifiers

Computations in practice

In practical situations, only an i.i.d data sample S is available. Let $S_k \subset S$ be the subset of observations known to belong to class ω_k. Then $S_1, \ldots, S_K$ is a partition of S.

We can use **unbiased estimates** for the vector means and for the class priors:

$$\widehat{\boldsymbol{\mu}}_k = \frac{1}{|S_k|} \sum_{\boldsymbol{x} \in S_k} \boldsymbol{x}; \qquad \widehat{P}(\omega_k) = \frac{|S_k|}{|S|}$$

1. If we know (or assume) that covariance matrices are **different** (wish to use **QDA**):

$$\widehat{\Sigma}_k = \frac{1}{|S_k| - 1} \sum_{\boldsymbol{x} \in S_k} (\boldsymbol{x} - \widehat{\boldsymbol{\mu}}_k)(\boldsymbol{x} - \widehat{\boldsymbol{\mu}}_k)^\top$$

2. If we know (or assume) that covariance matrices are **equal** (wish to use **LDA**):

$$\widehat{\Sigma}_{\text{pooled}} = \frac{1}{|S| - K} \sum_{k=1}^{K} (|S_k| - 1)\widehat{\Sigma}_k$$

Generative Bayesian classifiers

Discussion

- Bayesian classifiers are optimal when the class-conditional densities and priors are known; the methods are well-principled, fast and reliable

- For Gaussian classes, we get a quadratic classifier - QDA (if all covariance matrices are equal, a linear classifier - LDA); using a specific distance function corresponds to certain statistical assumptions:
  - If the class-conditional densities are far from the assumptions, the model will be poor
  - Even if the class-conditional densities are Gaussian, the parameters should be reliably estimated (particularly for QDA)
  - Once we use sample statistics instead of population parameters, we loose optimality!

- The question whether these assumptions hold can rarely be answered in practice; in most cases we are limited to posing and answering the question "*does this classifier give satisfactory predictions or not?*"

Generative Bayesian classifiers

Regularized Discriminant Analysis

- If $d > |S_k|$ for some k, QDA cannot be applied, because the class covariance matrix $\hat{\Sigma}_k$ is singular

- If $d > N$, neither QDA nor LDA can be used, because both $\hat{\Sigma}_k$ and $\hat{\Sigma}_{\text{pooled}}$ are singular

- These problems can be overcome by applying **regularization**:

$$\hat{\Sigma}_k(\lambda, \gamma) := (1 - \gamma)\left[(1 - \lambda)\hat{\Sigma}_k + \lambda\hat{\Sigma}_{\text{pooled}}\right] + \frac{\gamma}{d} Tr\left[\hat{\Sigma}_k(\lambda)\right] I_d$$

LDA is $(\lambda, \gamma) = (1, 0)$ and QDA is $(\lambda, \gamma) = (0, 0)$

Generative Bayesian classifiers

Pros & Cons

- Assumption of Gaussianity may be far from true
- Needs sufficient examples per class if we wish to use QDA
- Requires matrix inversions (costly or numerically delicate)
- Adaptable to all class-conditional distributions (not only Gaussian), even with mixed variables
- Very resistant to overfitting the data sample
- Accepts class priors and losses for misclassifications

The Naive-Bayes classifier

- We showed that the 0/1 loss Bayes rule minimizing the probability of error could be formulated in terms of discriminant functions:

  $$g_k(\boldsymbol{x}) = P(\omega_k)P(\boldsymbol{x}|\omega_k), k = 1, \ldots, K.$$

- We can expand the conditional probability:

  $$P(\omega_k)P(\boldsymbol{x}|\omega_k) = P(\omega_k)P(X_1 = x_1 \wedge X_2 = x_2 \wedge \ldots \wedge X_d = x_d \mid \omega_k)$$

  $$= P(\omega_k)P(X_1 = x_1|\omega_k) \prod_{j=2}^{d} P(X_j = x_j \mid \omega_k, X_1 = x_1 \wedge \ldots \wedge X_{j-1} = x_{j-1})$$

  assuming $X_1, \ldots, X_d$ are pairwise independent *given the class*:

  $$= P(\omega_k)P(X_1 = x_1|\omega_k) \prod_{j=2}^{d} P(X_j = x_j \mid \omega_k)$$

  $$= P(\omega_k) \prod_{j=1}^{d} P(X_j = x_j \mid \omega_k) \equiv \mathsf{NB}_k(\boldsymbol{x})$$

The Naive-Bayes classifier

Example

Outlook	Temperature	Humidity	Wind	PlayTennis?
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

The Naive-Bayes classifier

Example

The prediction for $\boldsymbol{x}^* =$ (Sunny,Hot,Normal,Weak)$^\top$ is $\boldsymbol{x}^* \in$ Yes:

- $\widehat{P}(\text{No}) \cdot \widehat{P}(\text{Sunny}|\text{No}) \cdot \widehat{P}(\text{Hot}|\text{No}) \cdot \widehat{P}(\text{Normal}|\text{No}) \cdot \widehat{P}(\text{Weak}|\text{No})$
  $= \frac{5}{14} \cdot \frac{3}{5} \cdot \frac{2}{5} \cdot \frac{1}{5} \cdot \frac{2}{5} = \frac{6}{875} \approx 6{,}86 \cdot 10^{-3}$

- $\widehat{P}(\text{Yes}) \cdot \widehat{P}(\text{Sunny}|\text{Yes}) \cdot \widehat{P}(\text{Hot}|\text{Yes}) \cdot \widehat{P}(\text{Normal}|\text{Yes}) \cdot \widehat{P}(\text{Weak}|\text{Yes})$
  $= \frac{9}{14} \cdot \frac{2}{9} \cdot \frac{2}{9} \cdot \frac{6}{9} \cdot \frac{6}{9} = \frac{8}{567} \approx 0{,}0141$

Note that the true posteriors would be:

$$\widehat{P}(\text{No}|\boldsymbol{x}^*) = \frac{6/875}{6/875+8/567} \approx 0{,}329, \qquad \widehat{P}(\text{Yes}|\boldsymbol{x}^*) = \frac{8/567}{6/875+8/567} \approx 0{,}671$$

The Naive-Bayes classifier

Extensions

1. A numerical suggestion: take logs!

$$\mathrm{NB}_k(\boldsymbol{x}) = \ln P(\omega_k) + \sum_{j=1}^{d} \ln P(X_j = x_j \mid \omega_k)$$

2. How do we deal with continuous variables?

   a) Assume a particular pdf for the variable and estimate its parameters from the data

   b) Discretize the variable and treat it as discrete

The Naive-Bayes classifier

Null empirical probabilities

In test examples $\boldsymbol{x}^*$, it may happen that some variable X_j has a value x_j^* not present in the sample used to create the classifier. In this case, $\widehat{P}(X_j = x_j^* \mid \omega_k) = 0$ and therefore we are in trouble ...

A possible workaround is the **Laplace correction**:

$$\widehat{P}_L(X_j = x_j^* \mid \omega_k) = \frac{|\{\boldsymbol{x} \in S_k \ \wedge \ X_j = x_j\}| + p}{|\{\boldsymbol{x} \in S_k\}| + p \cdot V_k}, \ p \in \mathbb{N}$$

where p is the "weight" assigned to the prior probability and V_k is the number of modalities of variable k

---

Example. Take $p = 1$ and $V_k = 3$ and assume $|\{\boldsymbol{x} \in S_k \ \wedge \ X_j = x_j\}| = 0$. Then $\widehat{P}_L(X_j = x_j^* \mid \omega_k) = \frac{1}{|\{\boldsymbol{x} \in S_k\}| + 3}$. Can you give an interpretation?

The kNN classifier

Let X be a set. A **metric** in X is a two-place function $d : X \to \mathbb{R}^{+} \cup \{0\}$ satisfying, forall $x, y, z \in X$:

1. $d(x, y) = 0 \iff x = y$
2. $d(x, y) = d(y, x)$
3. $d(x, y) \leq d(x, z) + d(z, y)$

We say that the pair (X, d) is a **metric space**.

The kNN classifier

1. The **1NN technique** classifies any $x \in X$ in the same class of the "nearest neighbour" of x in S, that is:

$$\boxed{\text{the class of } x \text{ is the class of } \arg\min_{x' \in S \setminus \{x\}} d(x, x')}$$

2. The **kNN technique** considers the $k \geq 1$ "nearest neighbours" of x in S and votes for the most represented class:

$$\boxed{\text{the class of } x \text{ is the majority class among its } k \text{ closest elements in } S}$$

- ties may happen and are broken randomly
- for two-class problems and odd k there can be no ties

The kNN classifier

Asymptotic analysis

Theorem (Cover & Hart '65). Call ϵ_{1NN} the probability of error of 1NN and ϵ_B be the Bayes error, then:

$$\epsilon_B \leq \epsilon_{1NN} \leq \epsilon_B \left(2 - \epsilon_B \frac{K}{K-1}\right) \leq 2\epsilon_B$$

in the limit $N \to \infty$.

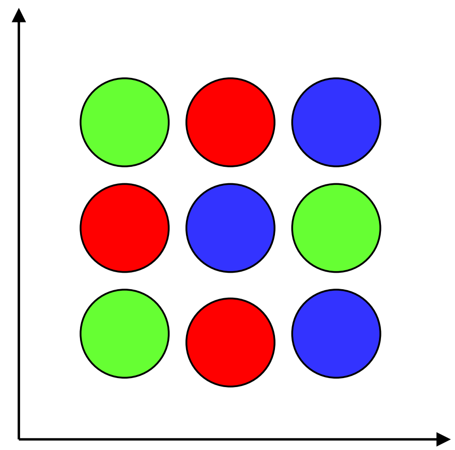
In particular, for two-class problems ($K = 2$),

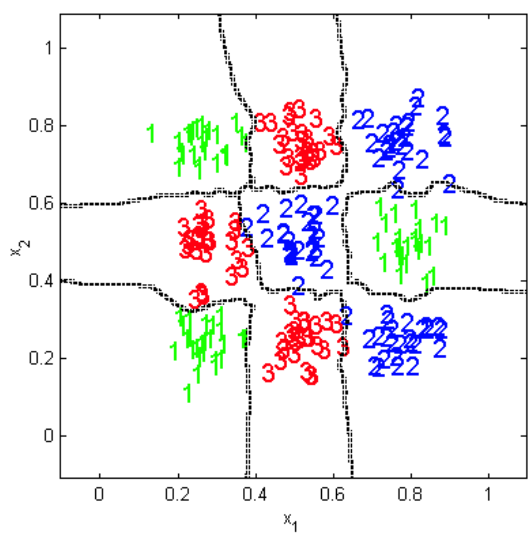
$$\epsilon_B \leq \epsilon_{1NN} \leq 2\epsilon_B(1 - \epsilon_B)$$

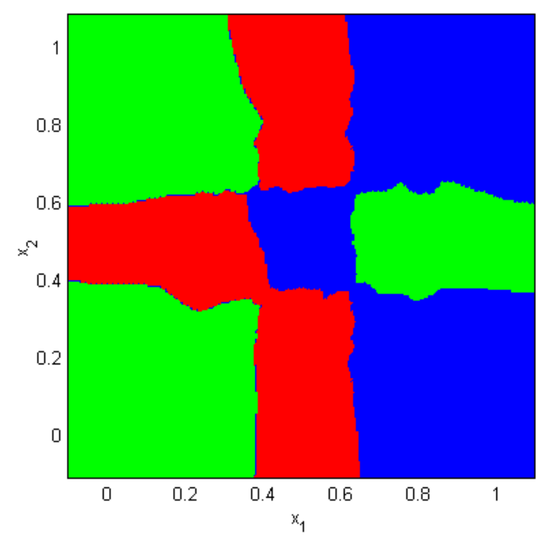
kNN in action

Example I

– Three-class 2D problem with non-linearly separable, multimodal likelihoods
– We use the kNN rule ($k = 5$) and the Euclidean distance
– The resulting decision boundaries and decision regions are shown below

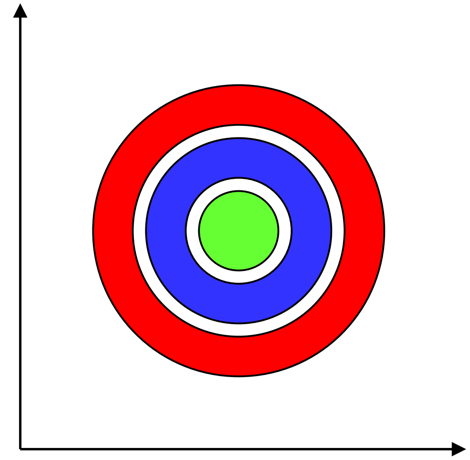


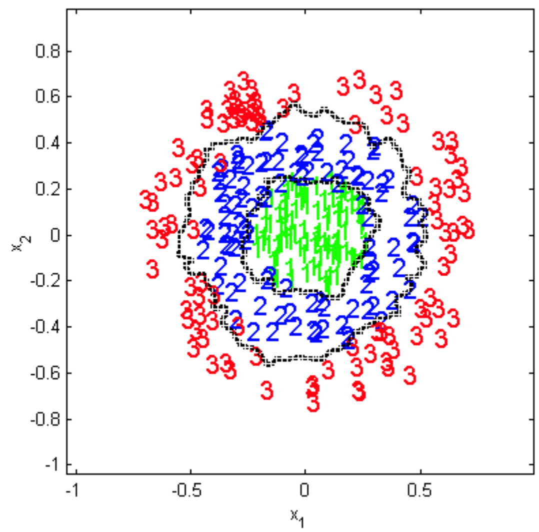


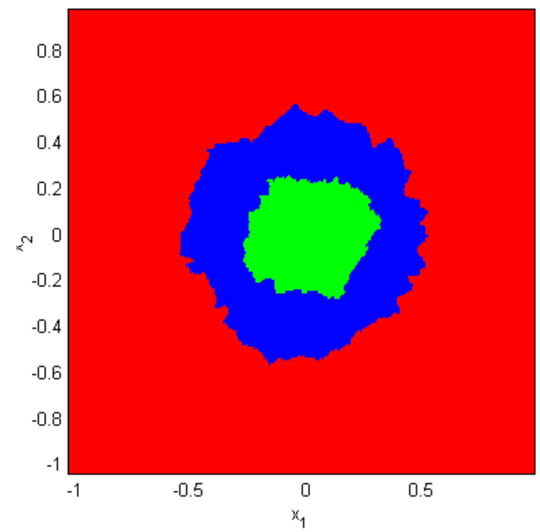


Example II

- Two-dim 3-class problem with unimodal likelihoods with a common mean; these classes are also not linearly separable
- We used the kNN rule ($k = 5$), and the Euclidean distance as a metric







The kNN classifier

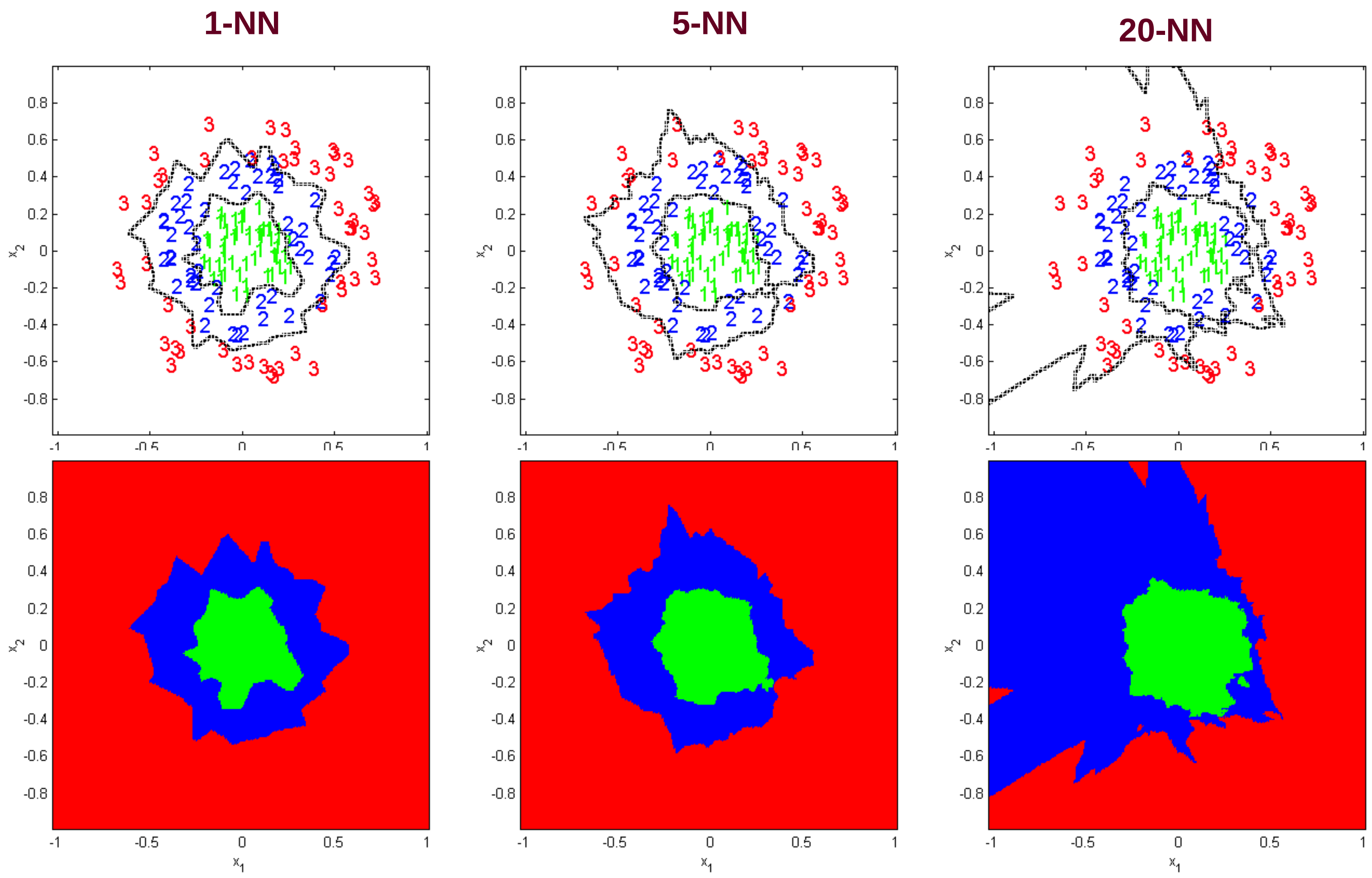
Characteristics of the kNN classifier

- Simple to understand and implement; no training time
- Nearly optimal in the large sample limit
- Uses local information, therefore highly adaptive behavior
- Lends itself very easily to parallel implementations

- Large storage requirements
- No explicit model (data $\equiv$ model)
- Large testing time
- Likely to break in large dimensions
- Choice of best k is very difficult

Influence of k

kNN versus 1NN

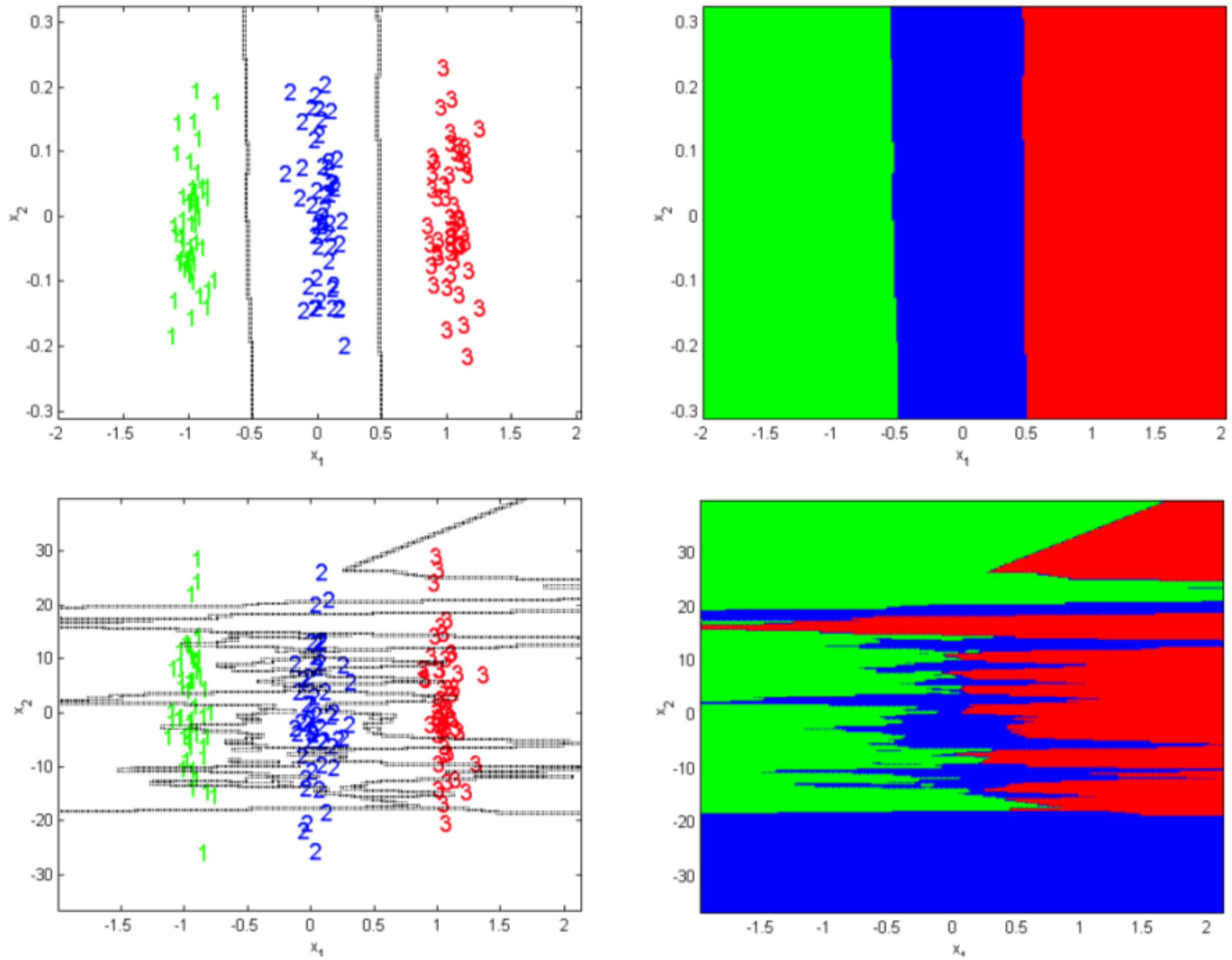


Influence of k

As k is increased ...

- We get smoother decision regions
- Less influence of “noise” (local fluctuations of class)
- We can get some probabilistic information
- Variance decreases
- Locality is lost
- The computational burden is increased
- Bias increases

Importance of standardization



The horizontal axis contains all the discriminatory information, the second axis is white noise. Top: both axes are scaled properly (kNN ($k = 5$) finds correct decision boundaries. Bottom: the scale of the vertical axis has been increased 100 times.

(p. 25)