# Lecture 4 – Discriminant Analysis, $k$-Nearest Neighbors

UPPSALA
UNIVERSITET

**Fredrik Lindsten**
Division of Systems and Control
Department of Information Technology
Uppsala University.

Email: fredrik.lindsten@it.uu.se

# The mini project has now started!

**Task:** Classify songs according to whether Andreas likes them or not (based on features from the Spotify Web-API).

**Training data:** 750 songs labeled as `like` or `dislike`.

**Test data:** 200 **unlabeled** songs. Upload your predictions to the leaderboard!

Examination: A written report (max 6 pages).

Instructions and data is available at the course homepage
`http://www.it.uu.se/edu/course/homepage/sml/project/`

You will receive your group password per e-mail tomorrow. Deadline for submitting your final solution: February 21

# Summary of Lecture 3 (I/VI)

The classification problem amounts to modeling the relationship between the input $\mathbf{x}$ and a **qualitative output** $y$, i.e., the output belongs to one out of $K$ distinct **classes**.

A **classifier** is a prediction model $\widehat{y} = \widehat{g}(\mathbf{x})$ that maps any input $\mathbf{x}$ into a predicted class $\widehat{y} \in \{1, \ldots, K\}$.

The Bayes classifier predicts each input as belonging to the most likely class, according to the conditional probabilities

$$\Pr(y = k \,|\, \mathbf{x}) \text{ for } k \in \{1, \ldots, K\}.$$

The Bayes classifier is **optimal w.r.t. misclassification error**.

# Summary of Lecture 3 (II/VI)

For binary (two-class) classification, $y \in \{0, 1\}$, the **logistic regression model** is

$$\Pr(y = 1 \,|\, \mathbf{x}) = \frac{e^{\boldsymbol{\beta}^\mathsf{T}\mathbf{x}}}{1 + e^{\boldsymbol{\beta}^\mathsf{T}\mathbf{x}}}.$$

The model parameters $\boldsymbol{\beta}$ are found by maximum likelihood by (numerically) maximizing the log-likelihood function,

$$\log \ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left( y_i \boldsymbol{\beta}^\mathsf{T}\mathbf{x}_i - \log\left(1 + e^{\boldsymbol{\beta}^\mathsf{T}\mathbf{x}_i}\right)\right).$$

# Summary of Lecture 3 (III/VI)

Approximating the Bayes classifier, we get the prediction model

$$\widehat{g}(\mathbf{x}) = \begin{cases} 1 & \text{if } \Pr(y = 1 \,|\, \mathbf{x}) > 0.5 \\ 0 & \text{otherwise} \end{cases} \quad \Leftrightarrow \widehat{\beta}^{\mathsf{T}} \mathbf{x} > 0$$

This attempts to **minimize the total misclassification error**.

More generally, we can use

$$\widehat{g}(\mathbf{x}) = \begin{cases} 1 & \text{if } \Pr(y = 1 \,|\, \mathbf{x}) > r \\ 0 & \text{otherwise} \end{cases} \qquad \Longrightarrow \qquad \widehat{g}(\mathbf{x}) = \mathbb{I}\{\Pr(y = 1 \,|\, \mathbf{x}) > r\}$$

where $0 \leq r \leq 1$ is a **user chosen threshold**.

# Summary of Lecture 3 (IV/VI)

**Confusion matrix:**

| | | Predicted condition $\widehat{y}=0$ | Predicted condition $\widehat{y}=1$ | Total |
|---|---|---|---|---|
| True condition | $y=0$ | **TN** | **FP** | **N** |
| | $y=1$ | **FN** | **TP** | **P** |
| | Total | **N*** | **P*** | |

For the classifier $\widehat{g}(\mathbf{x}) = \mathbb{I}\{\Pr(y=1\,|\,\mathbf{x}) > r\}$ the numbers in the confusion matrix will **depend on the threshold** $r$.

- Decreasing $r$ $\Rightarrow$ **TN**, **FN** decrease and **FP**, **TP** increase.
- Increasing $r$ $\Rightarrow$ **TN**, **FN** increase and **FP**, **TP** decrease.

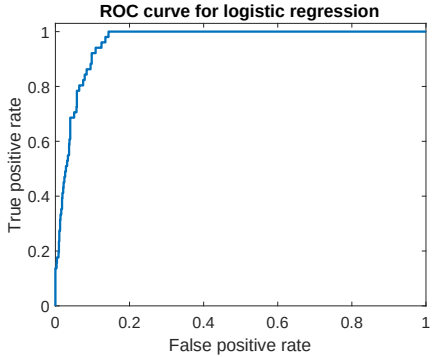

fredrik.lindsten@it.uu.se

# Summary of Lecture 3 (V/VI)

Some commonly used derived performance measures are:

- True positive rate: $\mathbf{TPR} = \dfrac{\mathbf{TP}}{\mathbf{P}} = \dfrac{\mathbf{TP}}{\mathbf{FN} + \mathbf{TP}} \in [0, 1]$
- False positive rate: $\mathbf{FPR} = \dfrac{\mathbf{FP}}{\mathbf{N}} = \dfrac{\mathbf{FP}}{\mathbf{FP} + \mathbf{TN}} \in [0, 1]$
- Precision: $\mathbf{Prec} = \dfrac{\mathbf{TP}}{\mathbf{P^*}} = \dfrac{\mathbf{TP}}{\mathbf{FP} + \mathbf{TP}} \in [0, 1]$

# Summary of Lecture 3 (VI/VI)




- **ROC: plot of TPR vs. FPR** as $r$ ranges from $0$ to $1$.[1]
- **Area Under Curve (AUC):** condensed performance measure for the classifier, taking all possible thresholds into account.

[1] Possible to draw ROC curves based on other tuning parameters as well.

UPPSALA UNIVERSITET

# Contents – Lecture 4

1. Linear Discriminant Analysis
2. Quadratic Discriminant Analysis
3. A nonparametric classifier – $k$-Nearest Neighbors

# Bayes' theorem

Figure: (Possibly) Thomas Bayes, c. 1701–1761.

Bayes' theorem describes a fundamental relationship between **conditional** and **marginal** probabilities.

$$p(z_1 \mid z_2) = \frac{p(z_2 \mid z_1)p(z_1)}{p(z_2)} = \frac{p(z_2 \mid z_1)p(z_1)}{\int p(z_2 \mid z_1)p(z_1)dz_1}$$

# Bayes' theorem for classification

Recall the Bayes classifier, based on the conditional probabilities $\Pr(y = k \,|\, \mathbf{x})$.

- $\Pr(y = k)$ denotes the **marginal** (prior) probability of class $k \in \{1, \ldots, K\}$.
- $p(\mathbf{x} \,|\, y = k)$ denotes the **conditional** probability density of $\mathbf{x}$ for an observation that comes from the $k$th class.

The Bayes classifier can then be expressed using Bayes' theorem:

$$\Pr(y = k \,|\, \mathbf{x}) = \frac{p(\mathbf{x} \,|\, y = k) \Pr(y = k)}{\sum_{j=1}^{K} p(\mathbf{x} \,|\, y = j) \Pr(y = j)}.$$

# Multivariate Gaussian density

The $p$-dimensional Gaussian (normal) probability density function with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ is,

$$\mathcal{N}(\mathbf{x} \,|\, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right),$$

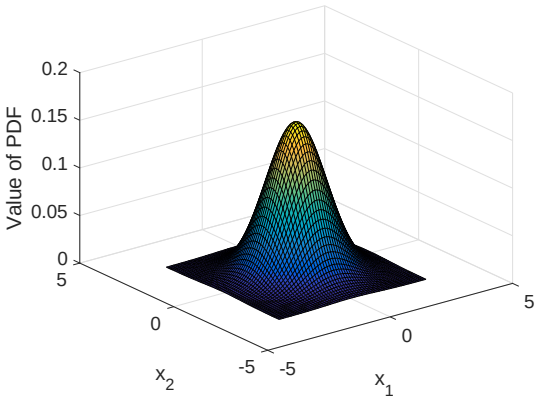
where $\boldsymbol{\mu}$ : $p \times 1$ vector and $\boldsymbol{\Sigma}$ : $p \times p$ positive definite matrix.

Let $\mathbf{x} = (x_1, \ldots, x_p)^{\mathsf{T}} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$,

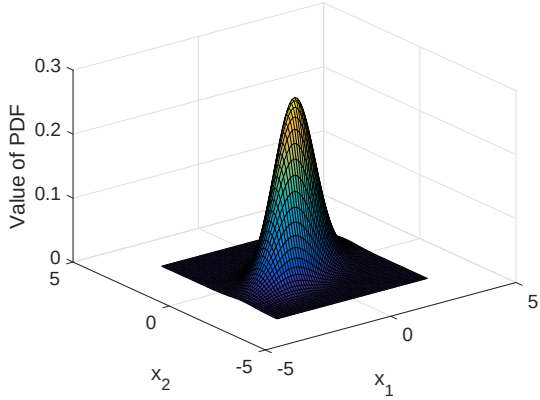
- $\mu_j$ is the mean of $x_j$,
- $\Sigma_{jj}$ is the variance of $x_j$,
- $\Sigma_{ij}$ $(i \neq j)$ is the covariance between $x_i$ and $x_j$.

 fredrik.lindsten@it.uu.se

# Multivariate Gaussian density




$$\mathbf{\Sigma} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$




$$\mathbf{\Sigma} = \begin{pmatrix} 1 & 0.4 \\ 0.4 & 0.5 \end{pmatrix}$$


fredrik.lindsten@it.uu.se

# LDA, summary

The LDA classifier assigns a test input $\mathbf{x} = \mathbf{x}_\star$ to class $k$ for which,

$$\widehat{\delta}_k(\mathbf{x}_\star) = \mathbf{x}_\star^\mathsf{T} \widehat{\boldsymbol{\Sigma}}^{-1} \widehat{\boldsymbol{\mu}}_k - \frac{1}{2} \widehat{\boldsymbol{\mu}}_k^\mathsf{T} \widehat{\boldsymbol{\Sigma}}^{-1} \widehat{\boldsymbol{\mu}}_k + \log \widehat{\pi}_k$$

is largest, where

$$\widehat{\pi}_k = n_k / n \qquad k = 1, \ldots, K,$$

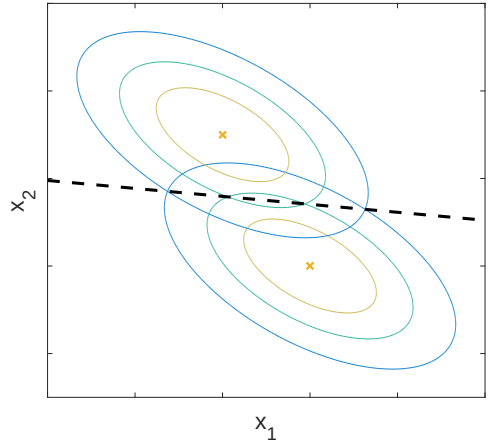
$$\widehat{\boldsymbol{\mu}}_k = \frac{1}{n_k} \sum_{i:y_i=k} \mathbf{x}_i, \qquad k = 1, \ldots, K,$$

$$\widehat{\boldsymbol{\Sigma}} = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_k)^\mathsf{T}.$$

$\widehat{\delta}_k(\mathbf{x})$ is a ***linear function*** in $\mathbf{x}$ and LDA is therefore a ***linear classifier*** (its decision boundaries are linear).

# *ex)* LDA decision boundary

*Illustration of LDA decision boundary* – the level curves of two Gaussian PDFs with **the same covariance matrix** intersect along a straight line, $\widehat{\delta}_1(\mathbf{x}) = \widehat{\delta}_2(\mathbf{x})$.





fredrik.lindsten@it.uu.se

UPPSALA UNIVERSITET

# *ex)* Simple spam filter

We will use LDA to construct a ***simple*** spam filter:

- **Output:** $y \in \{\texttt{spam}, \texttt{ham}\}$
- **Input:** $\mathbf{x} =$ 57-dimensional vector of "features" extracted from the email
  - Frequencies of 48 predefined words (`make`, `address`, `all`, ...)
  - Frequencies of 6 predefined characters (`;`, `(`, `[`, `!`, `$`, `#`)
  - Average length of uninterrupted sequences of capital letters
  - Length of longest uninterrupted sequence of capital letters
  - Total number of capital letters in the e-mail
- Dataset consists of $4,601$ emails classified as either `spam` or `ham` (split into 75 % training and 25 % testing)

UCI Machine Learning Repository — Spambase Dataset
(`https://archive.ics.uci.edu/ml/datasets/Spambase`)

# *ex)* Simple spam filter

LDA confusion matrix (test data):

| | $\widehat{y}=0$ | $\widehat{y}=1$ |
|---|---|---|
| $y=0$ | 667 | 23 |
| $y=1$ | 102 | 358 |

Table: Threshold $r=0.5$

| | $\widehat{y}=0$ | $\widehat{y}=1$ |
|---|---|---|
| $y=0$ | 687 | 3 |
| $y=1$ | 237 | 223 |

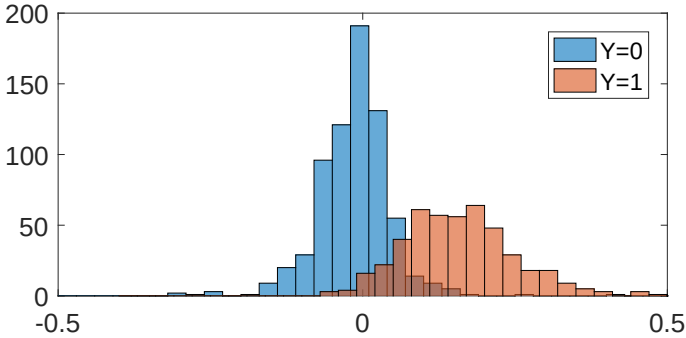
Table: Threshold $r=0.9$

# *ex)* Simple spam filter

A **linear** classifier uses a **linear decision boundary** for separating the two classes as much as possible.

In $\mathbb{R}^p$ this is a $(p-1)$-dimensional hyperplane. How can we visualize the classifier?

**One way:** project the (labeled) test inputs onto the normal of the decision boundary.





fredrik.lindsten@it.uu.se

# *ex)* Simple spam filter

**Welcome to Statistical Machine Learning!**

Från andreas.lindholm@it.uu.se
till ?=@velox.its.uu.se

fre 2019-01-18 16:38

Hi,

and welcome to the 2019 edition of Statistical Machine Learning. As you probably know, the course starts on Monday with a lecture at 10.15 in Siegbahnsalen.

I would already now like to highlight the "course home page" http://www.it.uu.se/edu/course/homepage/sml where you can find all information about the course, for example slides, the lecture notes, old exams, etc.

The course literature is the "lecture notes", which are almost finished now and available on the home page (one chapter will be added later). In total, they will be a little more than 100 pages, so we hope you will find it manageable to read them. If you want to read a full textbook, there are several recommendations on the home page as well.

If you want to come well prepared to the first lecture, have a "look at the lecture notes", in particular Chapter 1 and the list of contents (which also is the content of the course). On the home page, we also post links to a few "warm-up videos" for each lecture, which we recommend you to watch before coming to the lecture.

An important part of the course will be "your own work in problem solving sessions, the lab and the mini project". We recommend you to use either Python with scikit-learn (we suggest using Jupyter Notebooks) or R (we suggest using RStudio). A good start is therefore to also have a look and install one of those on your computer (if you don't know which one to choose, we'd say use Python).

The "mini project" will start after lecture 4, but before that "you have to sign up to a group". Each group should be 3-4 students, and you register using Studentportalen. (If you want to be randomly assigned to a group, there is an option for that, but you still have to register.)

If you have any questions, don't hesitate to contact us.

Have a nice weekend, I am looking forward to see you all on Monday!
Andreas Lindholm

| Feature | Count | Input ($\mathbf{x}$) |
|---|---|---|
| `make` | 0 | $x_1 = 0$ |
| `address` | 0 | $x_2 = 0$ |
| `all` | 2 | $x_3 = 0.60$ |
| ⋮ | ⋮ | ⋮ |
| `char ;` | 0 | $x_{49} = 0$ |
| `char (` | 6 | $x_{50} = 0.32$ |
| `char [` | 0 | $x_{51} = 0$ |
| `char !` | 1 | $x_{52} = 0.05$ |
| ⋮ | ⋮ | ⋮ |
| #Capitals | 34 | $x_{57} = 34$ |

# Gmail's spam filter

Official Gmail blog, July 9, 2015.

*"... the spam filter now uses an artificial neural network to detect and block the especially sneaky spam—the kind that could actually pass for wanted mail."* — Sri Harsha Somanchi, Product Manager

# Quadratic discriminant analysis

Do we have to assume a common covariance matrix?

**No!** Estimating a separate covariance matrix for each class leads to an alternative method, Quadratic Discriminant Analysis (QDA).

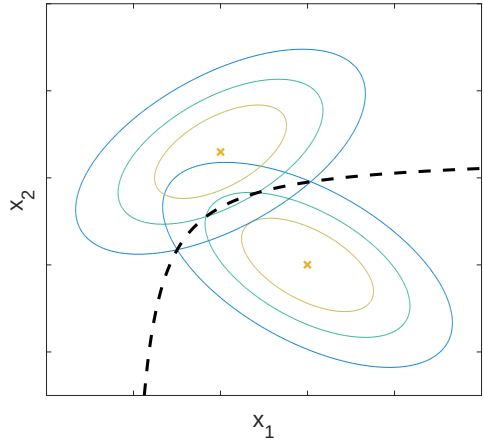
Whether we should choose LDA or QDA has to do with the **bias-variance trade-off**, i.e. the risk of **over- or underfitting**.

Compared to LDA, QDA...

- has more parameters
- is more flexible (lower bias)
- has higher risk of overfitting (larger variance)

# *ex)* QDA decision boundary

*Illustration of QDA decision boundary* – the level curves of two Gaussian PDFs with **different covariance matrices** intersect along a curved (quadratic) line.





fredrik.lindsten@it.uu.se

# Parametric and nonparametric models

So far we have looked at a few **parametric models**,

- linear regression,
- logistic regression,
- LDA and QDA,

all of which are parametrized by a ***fixed-dimensional parameter***.

**Non-parametric models** instead allow the flexibility of the model to grow with the amount of available data.

- ▲ can be ***very*** flexible (= low bias)
- ▼ can suffer from overfitting (high variance)
- ▼ can be compute and memory intensive

As always, the ***bias-variance trade-off*** is key also when working with non-parametric models.

# $k$-NN

The **$k$-nearest neighbors ($k$-NN)** classifier is a simple non-parametric method.

Given training data $\mathcal{T} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, for a test input $\mathbf{x}_\star$,

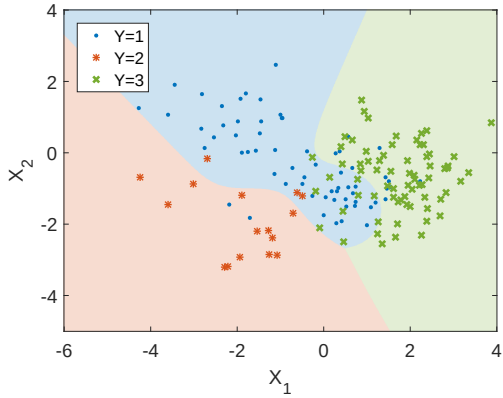
1. Identify the $k$ training inputs $\mathbf{x}_i$ nearest to $\mathbf{x}_\star$, represented by the set $R_\star = \{i : \mathbf{x}_i \text{ is in the } k\text{-neighborhood of } \mathbf{x}_\star\}$.
2. Classify $\mathbf{x}_\star$ according to a majority vote within the neighborhood $R_\star$, i.e. assign $\mathbf{x}_\star$ to class $j$ for which

$$\sum_{i \in R_\star} \mathbb{I}\{y_i = j\}$$

is largest.

# *ex)* $k$-NN on a toy model

We illustrate the $k$-NN classifier on a synthetic example where the Bayes classifier is known (colored regions).

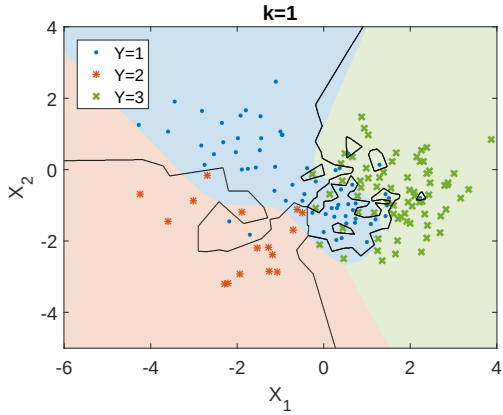




fredrik.lindsten@it.uu.se

# *ex)* $k$-NN on a toy model

The decision boundaries for the $k$-NN classifier are shown as black lines.

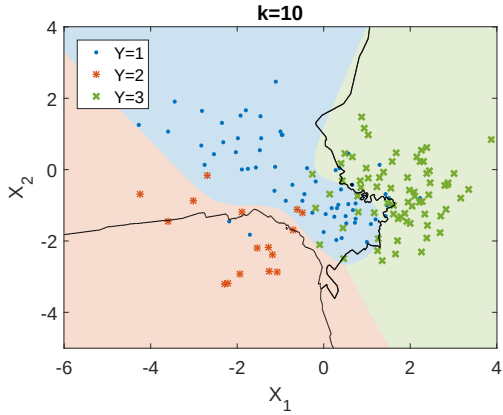




fredrik.lindsten@it.uu.se

# *ex)* $k$-NN on a toy model

The decision boundaries for the $k$-NN classifier are shown as black lines.

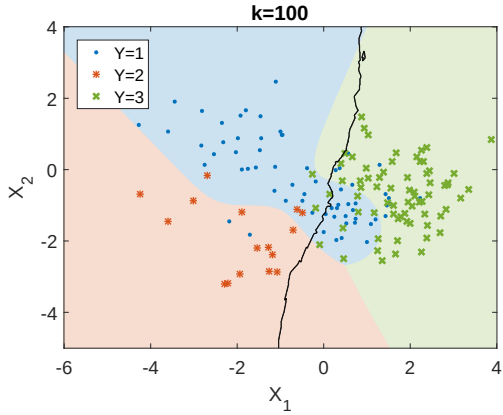




fredrik.lindsten@it.uu.se

# *ex)* $k$-NN on a toy model

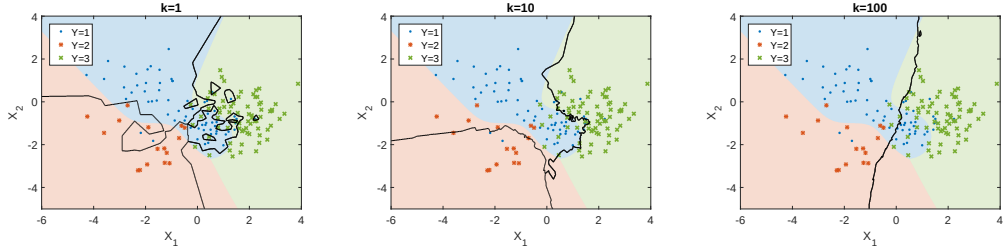
The decision boundaries for the $k$-NN classifier are shown as black lines.





fredrik.lindsten@it.uu.se

# *ex)* $k$-NN on a toy model

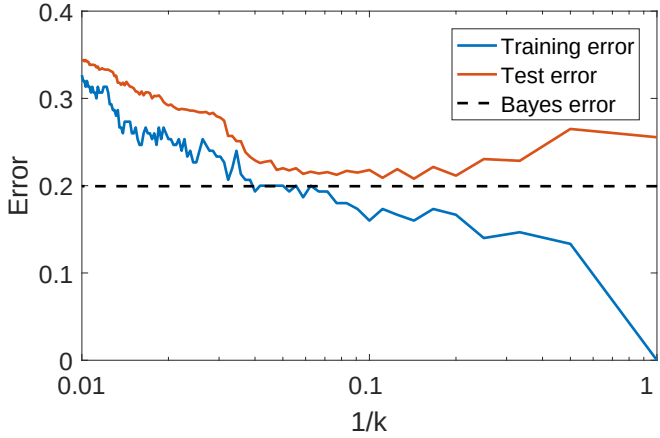



The choice of the ***tuning parameter*** $k$ controls the model flexibility:

- Small $k \Rightarrow$ small bias, large variance
- Large $k \Rightarrow$ large bias, small variance


fredrik.lindsten@it.uu.se

# *ex)* $k$-NN on a toy model





fredrik.lindsten@it.uu.se

# A few concepts to summarize lecture 4

**Bayes' theorem:** A formula relating conditional and marginal probabilities of two events.

**Linear discriminant analysis (LDA):** A classifier based on the assumption that the distribution of the input $\mathbf{x}$ is multivariate Gaussian for each class, with different means but the same covariance. LDA is a linear classifier.

**Quadratic discriminant analysis (QDA):** Same as LDA, but where a different covariance matrix is used for each class. QDA is *not* a linear classifier.

**Non-parametric model:** A model where the flexibility is allowed to grow with the amount of available training data.

**$k$-NN classifier:** A simple non-parametric classifier based on classifying a test input $\mathbf{x}$ according to the class labels of the $k$ training samples nearest to $x$.