Lecture 4 – Discriminant Analysis, k-Nearest Neighbors

UPPSALA
UNIVERSITET

Sebastian Mair
https://smair.github.io/
Department of Information Technology
Uppsala University

Course webpage

Summary of Lecture 3 (I/VI)

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

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

Common classifier predicts each input as belonging to the **most likely class** according to the conditional probabilities

$$p(y = m \,|\, \mathbf{x}) \text{ for } m \in \{1, \ldots, M\}.$$

Summary of Lecture 3 (II/VI)

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

$$p(y = 1 \,|\, \mathbf{x}) \approx g(\mathbf{x}) = \frac{e^{\theta^{\mathsf{T}}\mathbf{x}}}{1 + e^{\theta^{\mathsf{T}}\mathbf{x}}}.$$

The model parameters θ are found by maximum likelihood by (numerically) maximizing the log-likelihood function,

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

Summary of Lecture 3 (III/VI)

Using the common classifier, we get the prediction model

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

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

More generally, we can use

$$\widehat{y}(\mathbf{x}) = \begin{cases} 1 & \text{if } g(\mathbf{x}) > r \\ -1 & \text{otherwise,} \end{cases}$$

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

$$\widehat{y}(\mathbf{x}) = \begin{cases} 1 & \text{if } g(\mathbf{x}) > r \\ -1 & \text{otherwise,} \end{cases}$$

the numbers in the confusion matrix will **depend on the threshold** r.

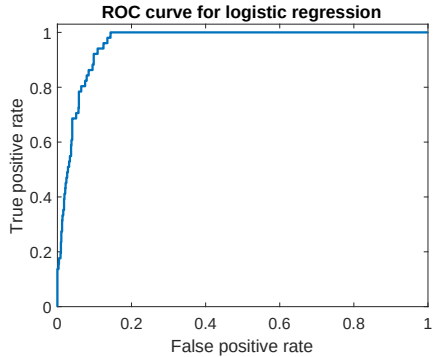
- Decreasing r $\Rightarrow$ **TN**, **FN** decrease and **FP**, **TP** increase.
- Increasing r $\Rightarrow$ **TN**, **FN** increase and **FP**, **TP** decrease.

Summary of Lecture 3 (V/VI)

Some commonly used 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

Generative models

A **discriminative model** describes how the output y is generated, i.e., $p(y \mid \mathbf{x})$.

A **generative model** describes how both the output y and the input $\mathbf{x}$ is generated via $p(y, \mathbf{x})$.

Predictions can be derived using laws of probability

$$p(y \mid \mathbf{x}) = \frac{p(y, \mathbf{x})}{p(\mathbf{x})} = \frac{p(y)p(\mathbf{x} \mid y)}{\int_y p(y, \mathbf{x})}.$$

Generative models for classification

We have built classifiers based on models $g_m(\mathbf{x}_\star)$ of the conditional probabilities $p(y = m \,|\, \mathbf{x})$.

A generative model for classification can be expressed as

$$p(y = m \,|\, x) = \frac{p(y = m)p(\mathbf{x} \,|\, y = m)}{\sum_{m=1}^{M} p(y = m)p(\mathbf{x} \,|\, y = m)}$$

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

Multivariate Gaussian density

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

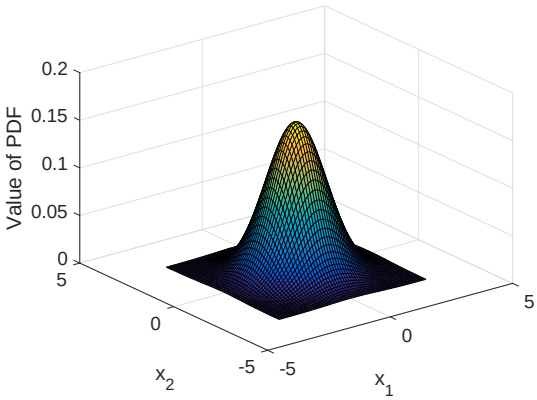
$$\mathcal{N}(\mathbf{x} \,|\, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right),$$

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

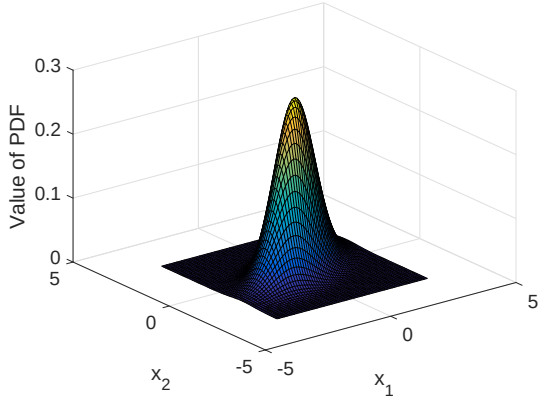
Let $\mathbf{x} = (x_1, \ldots, x_p)^{\mathsf{T}} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$,

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

Multivariate Gaussian density



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



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

Linear discriminant analysis

We need

1. The prior class probabilities $\pi_m \triangleq p(y = m)$, $m \in \{1, \ldots, M\}$.
2. The conditional probability densities of the input $\mathbf{x}$, $f_m(\mathbf{x}) \triangleq p(\mathbf{x} \mid y = m)$, for each class m.

This gives us the model

$$g_m(\mathbf{x}) = \frac{\pi_m f_m(\mathbf{x})}{\sum_{m=1}^{M} \pi_m f_m(\mathbf{x})}.$$

Linear discriminant analysis

For the **first task**, a natural *estimator* is the proportion of training samples in the mth class

$$\widehat{\pi}_m = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}\{y_i = m\} = \frac{n_m}{n},$$

where n is the size of the training set and n_m the number of training samples of class m.

Linear discriminant analysis

For the **second task**, a simple model is to *assume* that $f_m(x)$ is a multivariate normal density with mean vector $\boldsymbol{\mu}_m$ and covariance matrix $\boldsymbol{\Sigma}_m$,

$$f_m(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}_m|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_m)^\top \boldsymbol{\Sigma}_m^{-1}(\mathbf{x}-\boldsymbol{\mu}_m)\right).$$

If we further *assume* that all classes share the same covariance matrix,

$$\boldsymbol{\Sigma} \stackrel{\text{def}}{=} \boldsymbol{\Sigma}_1 = \cdots = \boldsymbol{\Sigma}_M,$$

the remaining parameters of the model are

$$\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_M, \boldsymbol{\Sigma}.$$

Linear discriminant analysis

These parameters are naturally estimated as the (within class) sample means and sample covariance, respectively:

$$\widehat{\boldsymbol{\mu}}_m = \frac{1}{n_m} \sum_{i:y_i=m} \mathbf{x}_i, \qquad m = 1, \ldots, M,$$

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

Modeling the class probabilities using the normal assumptions and these parameter estimates is referred to as ***Linear Discriminant Analysis (LDA)***.

Linear discriminant analysis, summary

The LDA classifier assigns a test input $\mathbf{x}$ to class m for which,

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

is largest, where

$$\widehat{\pi}_m = \frac{n_m}{n}, \qquad m = 1, \ldots, M,$$

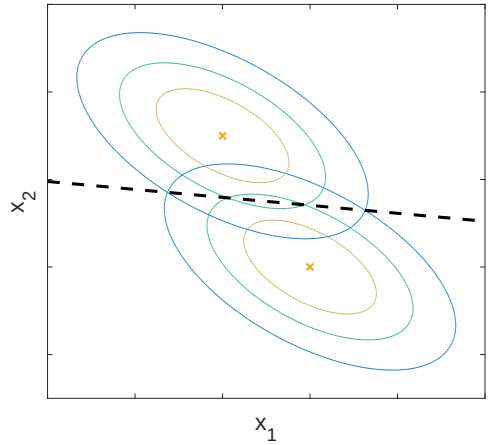
$$\widehat{\boldsymbol{\mu}}_m = \frac{1}{n_m}\sum_{i:y_i=m}\mathbf{x}_i, \qquad m = 1, \ldots, M,$$

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

LDA is 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, $\widehat{\delta}_1(\mathbf{x}) = \widehat{\delta}_2(\mathbf{x})$.



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

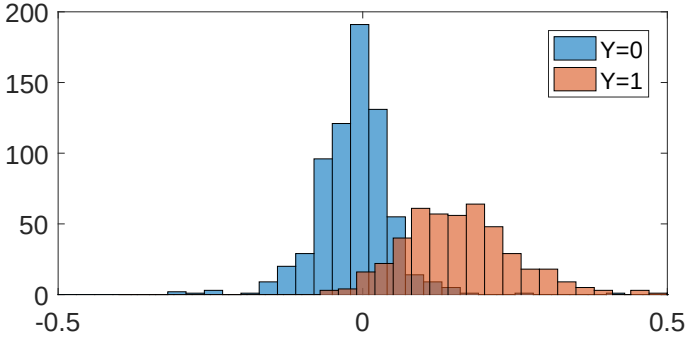
Table: Threshold $r=0.9$

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.



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

Official Gmail blog February 6, 2019:

"With TensorFlow, we are now blocking around 100 million additional spam messages every day" — Neil Kumaran, Product Manager, Gmail Security

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).

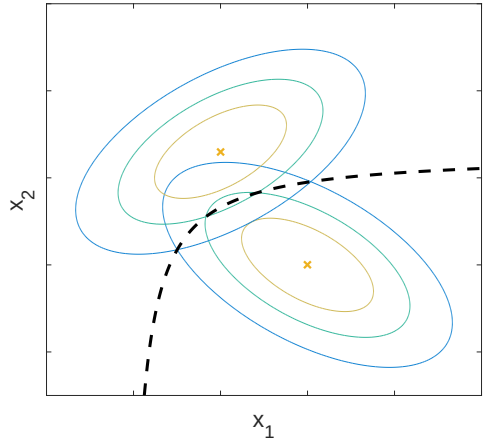
Whether we should choose LDA or QDA has to do with the **bias-variance trade-off**, i.e., the risk of **over- or underfitting**.

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.



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 set

$$R_\star = \{i : \mathbf{x}_i \text{ is one of the } k \text{ training data points closest to } \mathbf{x}_\star\}.$$

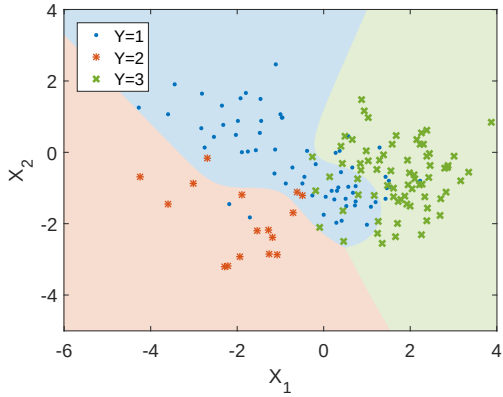
2. Classify $\mathbf{x}_\star$ according to a majority vote within the neighborhood $R_\star$, i.e., assign $\mathbf{x}_\star$ to class m for which

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

is largest.

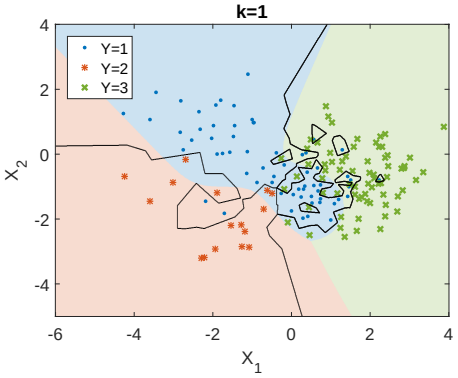
ex) k-NN on a toy model

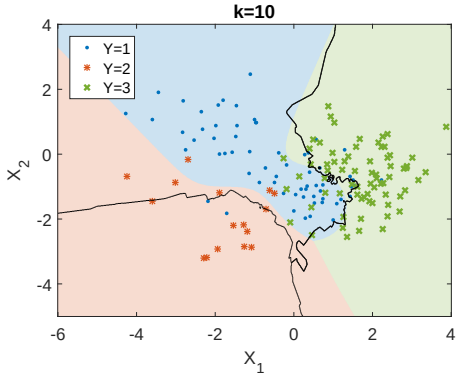
We illustrate the k-NN classifier on a synthetic example where the optimal classifier is known (colored regions).



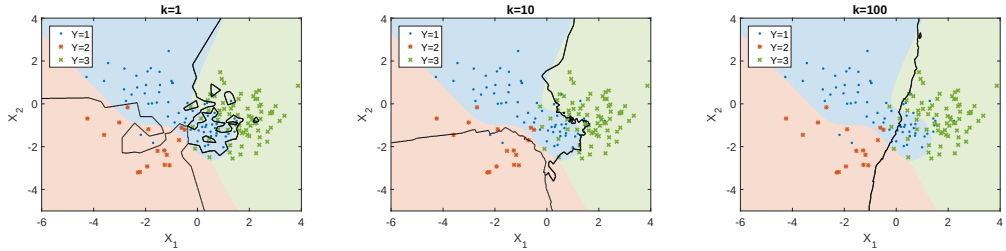
ex) k-NN on a toy model

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





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.

A few concepts to summarize lecture 4

Generative model: A model that describes both the output y and the input $\mathbf{x}$.

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.