Lecture 4 – k-Nearest Neighbors, Discriminant Analysis,

UPPSALA
UNIVERSITET

David Sumpter
Division of Systems and Control
Department of Information Technology
Uppsala University.

Email:

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

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

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

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

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.

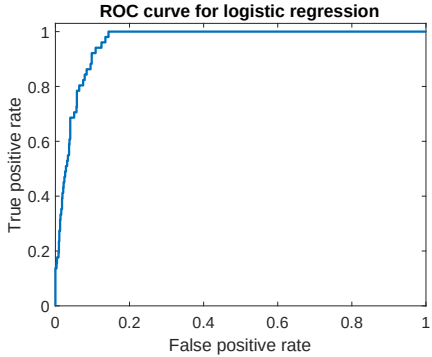
- Decreasing r $\Rightarrow$ **TN**, **FN** decrease and **FP**, **TP** increase.
- Increasing r $\Rightarrow$ **TN**, **FN** increase and **FP**, **TP** decrease.

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

Parametric and non-parametric models

Up to now we have looked at **parametric models**,

- linear regression,
- logistic regression,
- LDA and QDA (later in this lecture)

all of which involve assuming (in all these examples) a Normal distribution and estimating parameters like β, μ and σ.

Another approach is to use a non-parametric model. A model which directly relates the input data to the output data. We make no assumption about how the data is distributed (all models are wrong anyway!).

UPPSALA UNIVERSITET

A simple example of k-nearest neighbors

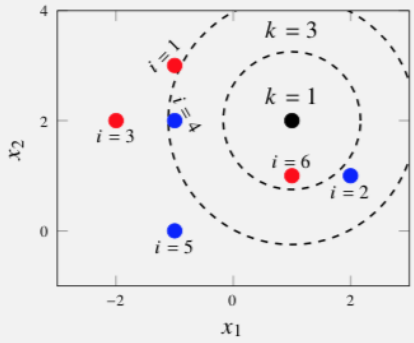
.

i	x_1	x_2	y
1	−1	3	Red
2	2	1	Blue
3	−2	2	Red
4	−1	2	Blue
5	−1	0	Blue
6	1	1	Red

A simple example of k-nearest neighbors

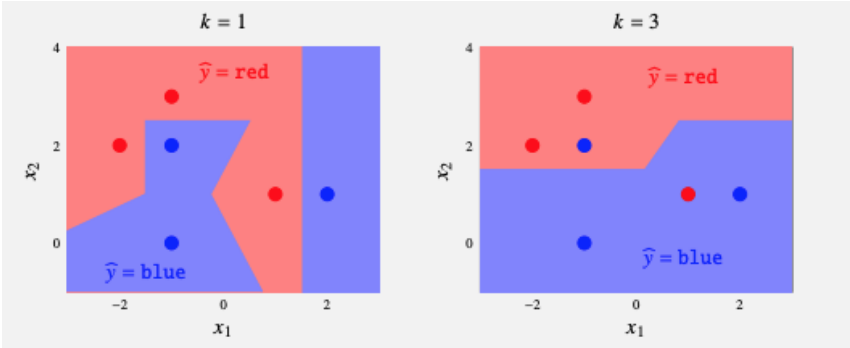
.

i	$\|\mathbf{x}_i - \mathbf{x}_\star\|$	y_i
6	$\sqrt{1}$	Red
2	$\sqrt{2}$	Blue
4	$\sqrt{4}$	Blue
1	$\sqrt{5}$	Red
5	$\sqrt{8}$	Blue
3	$\sqrt{9}$	Red



A simple example of k-nearest neighbors

.



k-nearest neighbors is a non-linear classifier.

Logistic regression is a linear classifier.

k-NN algorithm

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

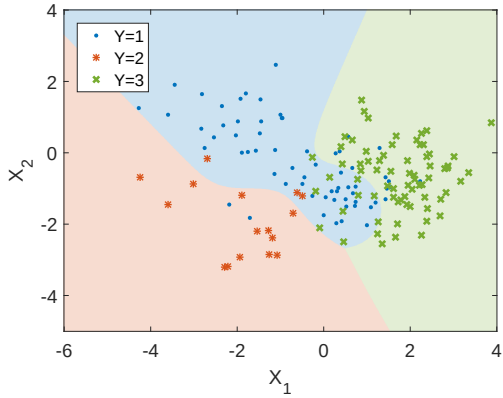
$$\sum_{i \in R_\star} \mathbb{I}\{y_i = m\}$$

is largest.

UPPSALA UNIVERSITET

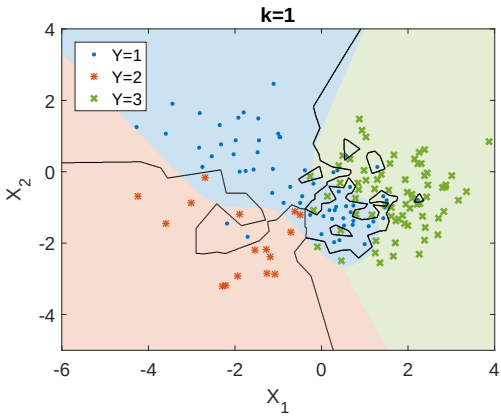
ex) k-NN with three classes

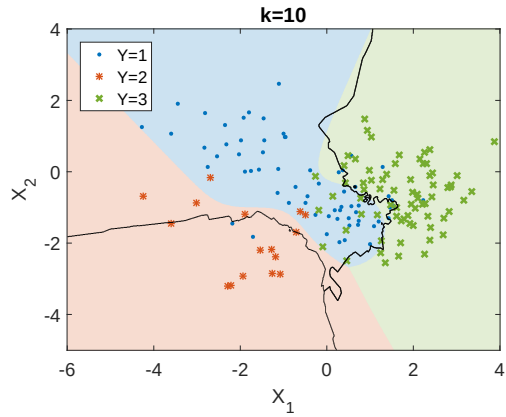
We illustrate the k-NN classifier on a synthetic example where the optimal classifier is known (colored regions).



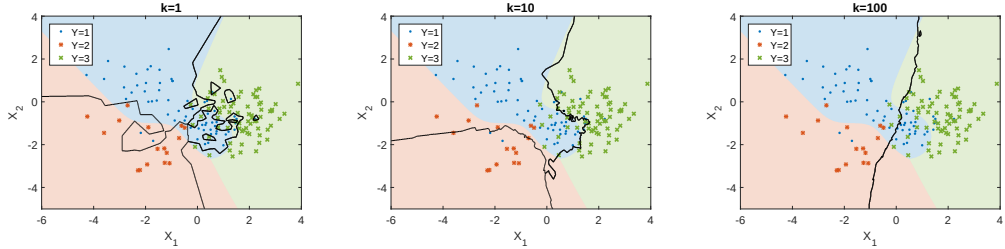
ex) k-NN on a toy model

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





ex) k-NN with three classes



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

UPPSALA UNIVERSITET

Parametric vs. non-Parametric

Advantages of non-parametric models

- Allow the flexibility of the model to grow with the amount of available data.
- Can be ***very*** flexible (= low bias, see next lecture)
- Intuitive. Less theory involved.

Disadvantages of non-parametric models

- Have very little theoretical support
- Can suffer from overfitting (= high variance, see next lecture)
- Can be computing and memory intensive
- Difficult to interpret what the results mean

The ***bias-variance trade-off*** (next lecture) is going to be a useful tool.

UPPSALA UNIVERSITET

Contents – Lecture 4

Bayes' theorem

If A and B are two events with $\Pr(B) > 0$, then

$$\Pr(A \mid B) = \frac{\Pr(B \mid A)\Pr(A)}{\Pr(B)}.$$

Can also be written,

$$\Pr(A \mid B) = \frac{\Pr(B \mid A)\Pr(A)}{\Pr(B \mid A)\Pr(A) + \Pr(B \mid A^C)\Pr(A^C)}.$$

where A^C is the complement of A.

Example of Bayes' theorem

Let A be the event that you have a disease. Let B be the event that a test you have taken is positive.

Assume that the test gives an error 1 in 100 tests., i.e. $\Pr(B|A) = 99/100$ and $\Pr(B|A^C) = 1/100$.

Further assume a randomly selected person has probability 1 in 1000 people of having the disease, i.e. $\Pr(A) = 1/1000$

You have no symptoms but take a test and test positive. What is the probability you have the disease?

UPPSALA UNIVERSITET

Example of Bayes' theorem

You have no symptoms but take a test and test positive. What is the probability you have the disease?

$$\begin{aligned}
\Pr(A \mid B) &= \frac{\Pr(B \mid A)\Pr(A)}{\Pr(B \mid A)\Pr(A) + \Pr(B \mid A^C)\Pr(A^C)} \\
&= \frac{99/100 \cdot 1/1000}{99/100 \cdot 1/1000 + 1/100 \cdot 999/1000} \\
&\approx 1/11
\end{aligned}$$

!!!!!!

Discriminative vs. Generative models

A **discriminative model** describes how the output y is generated drieectly, i.e. $p(y \mid \mathbf{x})$. This is what we do in logistic regression.

A **generative model** describes how both the output y and the input $\mathbf{x}$ is generated via $p(y, \mathbf{x}) = p(y)p(\mathbf{x} \mid y)$. This is what we do now.

A discriminative model takes the role of $\Pr(A \mid B)$. A generative model is based on $\Pr(B \mid A)$.

Discriminat analysis

We need to complete two steps:

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

We can then use Bayes' theorem to create a discriminative classifier from a generative model:

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

Discriminant analysis

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

$$\Pr(y = m) \text{ is estimated by } \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}\{y_i = m\} = \frac{n_m}{n}$$

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

This the idea of an uninformative prior probability: before we use the model the probability it is in any class is proportional to number in that class.

Discriminant analysis

For the **second step**, one model is to *assume* that the pdf of $p(\mathbf{x}|y = m)$ is a multivariate normal density with mean vector $\boldsymbol{\mu}_m$ and covariance matrix $\boldsymbol{\Sigma}_m$,

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

The parameters of the model are the means

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

And the variance

$$\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_M,$$

the remaining parameters of the model are

Multivariate Gaussian density

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

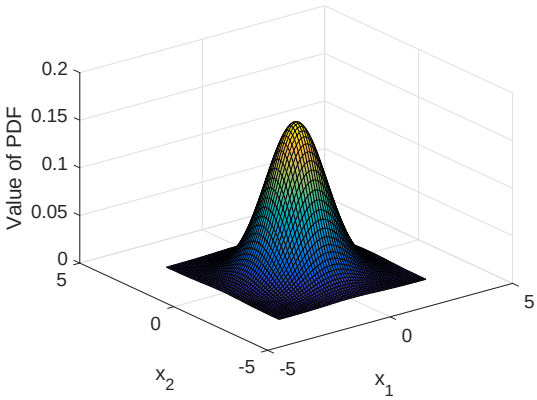
$$\mathcal{N}(\mathbf{x} \,|\, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right),$$

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

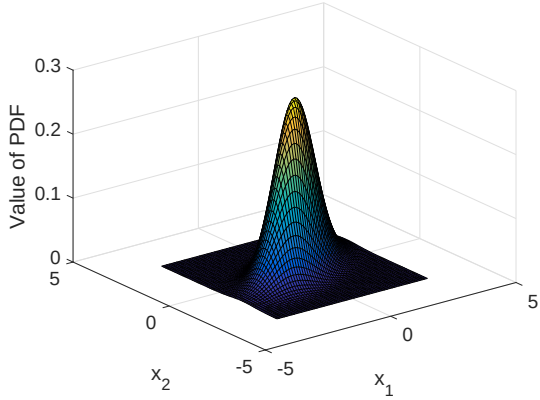
Let $\mathbf{x} = (x_1, \ldots, x_p)^{\mathsf{T}} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$,

- 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



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



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

Estimating the co-variance matrix

The mean vector $\boldsymbol{\mu}$ is estimated by

$$\widehat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i, \tag{1}$$

The covariance matrix $\boldsymbol{\Sigma}$ is estimated by

$$\widehat{\boldsymbol{\Sigma}}_m = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}-\widehat{\boldsymbol{\mu}})(\mathbf{x}-\widehat{\boldsymbol{\mu}}_m)^{\mathsf{T}}. \tag{2}$$

The element (j,k) of the covariance matrix is equal to the covariance between $\mathbf{x}_j$ and $\mathbf{x}_k$, i.e. the entry has value

$$\frac{1}{n-1}\sum_{i=1}^{n}(x_{ij}-\widehat{\mu}_j)(x_{ik}-\widehat{\mu}_k)$$

Linear discriminant analysis: estimation

Linear discriminant analysis assumes that the means are the same for all classes m

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

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

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

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

Linear discriminant analysis: classification

The probability (density function) of a point y is in class m is estimated by

$$\frac{\mathcal{N}\left(\hat{\boldsymbol{\mu}_m}, \hat{\boldsymbol{\Sigma}}\right) \frac{n_m}{n}}{\sum_{j=1}^{K} \mathcal{N}\left(\hat{\boldsymbol{\mu}}_j, \hat{\boldsymbol{\Sigma}}\right) \frac{n_j}{n}}.$$

Since the denominator is independent of m we just need to find the value of m for which

$$\mathcal{N}\left(\hat{\boldsymbol{\mu}_m}, \hat{\boldsymbol{\Sigma}}\right) \frac{n_m}{n}$$

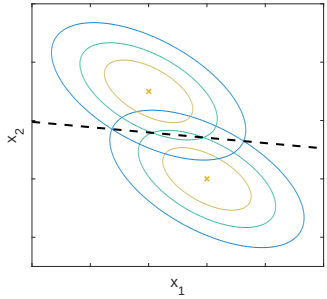
is maximised. Or equivalently, for which the log of this is maximised.
Thus, the LDA classifier assigns a test input $\mathbf{x}$ to class m for which,

$$\widehat{\delta}_m(\mathbf{x}) = \underbrace{\mathbf{x}^\mathsf{T} \widehat{\boldsymbol{\Sigma}}^{-1} \widehat{\boldsymbol{\mu}}_m - \frac{1}{2} \widehat{\boldsymbol{\mu}}_m^\mathsf{T} \widehat{\boldsymbol{\Sigma}}^{-1} \widehat{\boldsymbol{\mu}}_m}_{\text{step 2}} + \underbrace{\log \frac{n_m}{n}}_{\text{step 1}}$$

is largest.

Linear discriminant analysis: decision boundary

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



LDA is a ***linear classifier*** (its decision boundaries are linear).

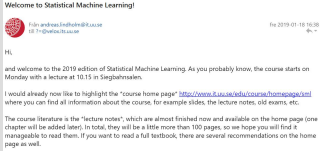
ex) Simple spam filter

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



Welcome to Statistical Machine Learning!

Från andreas.lindholm@it.uu.se fre 2019-01-18 16:38
till ?<@wiloo.its.uu.se

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	$x_3 = 0.60$
$\vdots$	$\vdots$	$\vdots$
char ;	0	$x_{49} = 0$
char (	6	$x_{50} = 0.32$
#Capitals	34	$x_{57} = 34$

$\Pr(\texttt{ham} \mid \mathbf{x}) = 96.4\%$ (according to LDA model)

ex) Simple spam filter

Please note that I have switched rows and columns here to be consistent with book and other sources

LDA confusion matrix (test data):

$r = 0.5$	Spam $y = 1$	Ham $y = 0$
$\widehat{y} = 1$	358	23
$\widehat{y} = 0$	102	667

True positive rate: 358/(358+102).
False positive rate: 23/(667+23).

$r = 0.9$	Spam $y = 1$	Ham $y = 0$
$\widehat{y} = 1$	223	3
$\widehat{y} = 0$	237	687

True positive rate: 223/(223+237).
False positive rate: 3/(687+3).

Where an email is categorised as spam if $p(y = \text{'spam'} \,|\, \mathbf{x}) > r$.

UPPSALA UNIVERSITET

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).

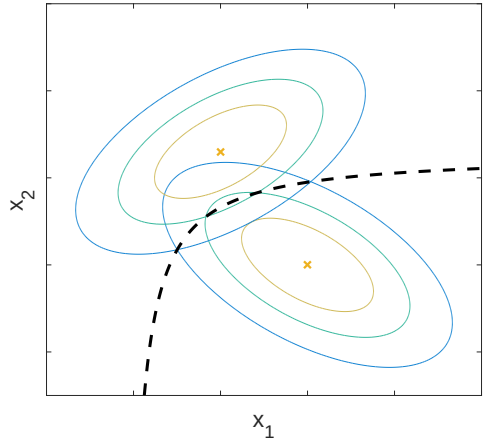
Whether we should choose LDA or QDA has to do with the **bias-variance trade-off**, i.e. the risk of **over- or underfitting**.

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.



UPPSALA UNIVERSITET

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.