Introduction to Machine Learning - Data Classification

Mathieu Fauvel, Florent Chatelain

Introduction to Machine Learning

Data classification

Florent Chatelain [1]   Mathieu Fauvel [2]

December 7, 2018

[1]MCF Grenoble INP, GIPSA-lab

[2]CR1 INRA, CESBIO

Florent Chatelain

- Ph.D. degree in signal processing from the National Polytechnic Institute, Toulouse, France, in 2007
- Post-doc position at INRIA - ARIANA Team, 2007-2008
- Since 2008, Associate Professor at GIPSA-Lab, University of Grenoble, France.
- Research interests are centered around estimation, detection, and the analysis of stochastic processes.

Mathieu Fauvel

- Ph.D. degree in signal and image processing from the National Polytechnic Institute, Grenoble, France, and the University of Iceland, in 2007
- Post-doc position at INRIA - MISTIS Team, 2008-2010
- Assistant Professor (Grenoble, 2007-2008 & Toulouse, 2010-2011)
- Associate Professor at DYNAFOR, National Polytechnic Institute, Toulouse, between 2011-2018.
- Since 2018, Research (CR1) at CESBIO, INRA.
- Research interests are: machine learning for environmental/ecological monitoring

- Pdf and notebooks available here:

  `https://framagit.org/mfauvel/omp_machine_learning`
- Jupyter notebooks binder are available:
- Citation:

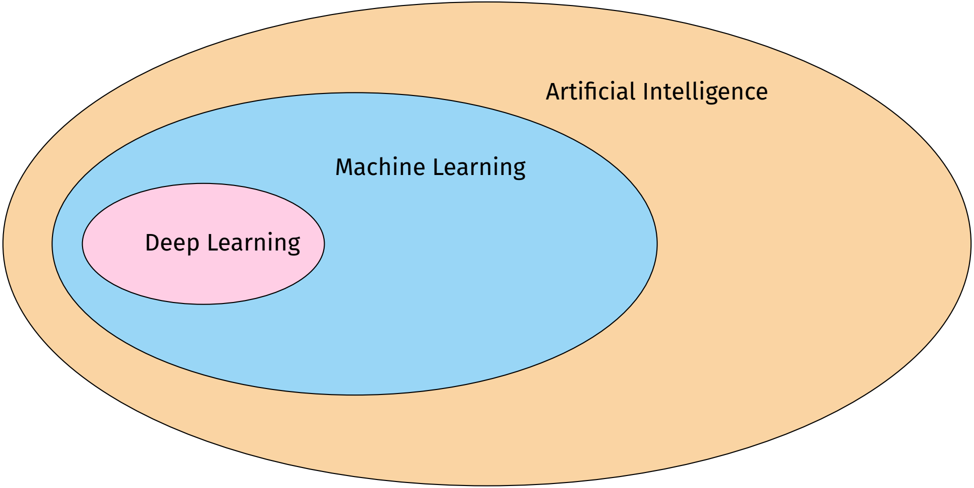
  `doi:10.5281/zenodo.1920227`

Outline

Introduction

Introduction

What is machine learning?



Taken from `https://www.geospatialworld.net/blogs/difference-between-ai%EF%BB%BF-machine-learning-and-deep-learning/`

How to extract knowledge or insights from data ?

Learning problems are at the cross-section of several applied fields and science disciplines

- *Machine learning* arose as a subfield of
  - Artificial Intelligence,
  - Computer Science.

  Emphasis on large scale implementations and applications: algorithm centered
- *Statistical learning* arose as a subfield of
  - Statistics,
  - Applied Maths,
  - Signal Processing, ...

  Emphasizes models and their interpretability: model centered

Machine Learning in Computer Science

Tom Mitchell (The Discipline of Machine Learning, 2006)

A computer program CP is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E

Key points

Machine Learning in Computer Science

Tom Mitchell (The Discipline of Machine Learning, 2006)

A computer program CP is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E

Key points

- Experience E: data and statistics

Machine Learning in Computer Science

Tom Mitchell (The Discipline of Machine Learning, 2006)

A computer program CP is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E

Key points

- Experience E: data and statistics
- Performance measure P: optimization

Machine Learning in Computer Science

Tom Mitchell (The Discipline of Machine Learning, 2006)

A computer program CP is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E

Key points

- Experience E: data and statistics
- Performance measure P: optimization
- tasks T: utility
  - automatic translation
  - playing Go
  - ... doing what human does

Type of data: qualitatives / ordinales / quantitatives variables

- Text: strings
- Speech: time series
- Images/videos: 2/3d dependences
- Networks: graphs
- Games: interaction sequences
- ...

Big data (volume, velocity, variety, veracity)

Data are available without having decided to collect them!

- importance of preprocessings (cleaning up, normalization, coding,...)
- importance of a good representation : from raw data to vectors

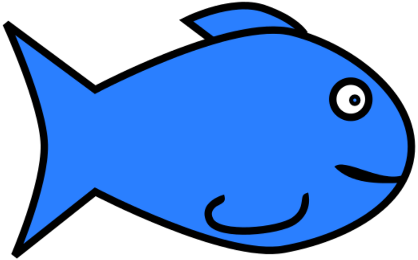
Generalize

- Perform well (minimize P) on new data (fresh data, i.e. unseen during learning)

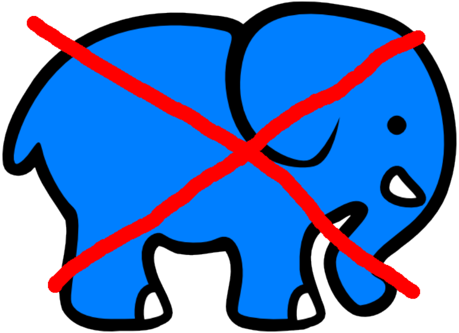
☞ Derive good (P/error rate) prediction functions

Generalize

- Perform well (minimize P) on new data (fresh data, i.e. unseen during learning)
- ☞ Derive good (P/error rate) prediction functions



A fish



A fish

Reference books

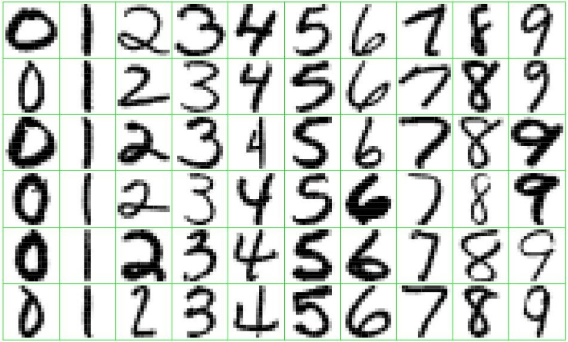
- Trevor Hastie, Robert Tibshirani et Jerome Friedman (2009), The Elements of Statistical Learning (2nd Edition), *Springer Series in Statistics*
- Christopher M. Bishop (2007), Pattern Recognition and Machine Learning, *Springer*
- Kevin P. Murphy (2013), Machine Learning: a Probabilistic Perspective, *Amazon*

Supplementary materials, datasets, online courses, ...

- `http://www-stat.stanford.edu/~tibs/ElemStatLearn/`
- `http://research.microsoft.com/en-us/um/people/cmbishop/prml/`
- `https://www.coursera.org/course/ml` *very popular MOOC (Andrew Ng)*
- `https://work.caltech.edu/telecourse.html` *more involved MOOC (Y. Abu-Mostafa)*
- `https://scikit-learn.org/stable/auto_examples/index.html` *Examples from the sklearn library*

Introduction

Examples



- ☞ Predict the class (0,...,9) of each sample from an image of 16×16 pixels, with a pixel intensity coded from 0 to 255
- Low error rate to avoid wrong allocations of mails!

Supervised classification

Spam

```
WINNING NOTIFICATION
We are pleased to inform you of the result
of the Lottery Winners International
programs held on the 30th january 2005.
[...]  You have been approved for a lump sum
pay out of 175,000.00 euros.
CONGRATULATIONS!!!
```

No Spam

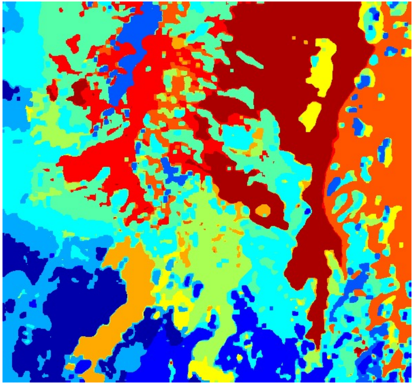
```
Dear George,
Could you please send me the report #1248 on
the project advancement?
Thanks in advance.

Regards,
Cathia
```

☞ Define a model to predict whether an email is spam or not

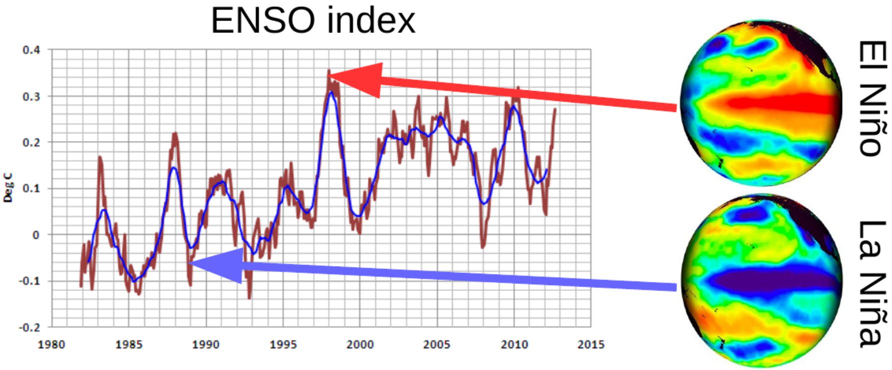
- Low error rate to avoid deleting useful messages, or filling the mailbox with useless emails

supervised classification



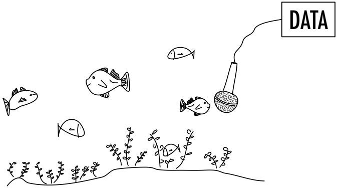
☞ Predict the class of landscape $\in$ { Lava 1970, Lava 1980 I, Lava 1980 II, Lava 1991 I, Lava 1991 II, Lava moss cover, hyaloclastite formation, Tephra lava, Rhyolite, Scoria, Firn-glacier ice, Snow } from digital remote sensing images

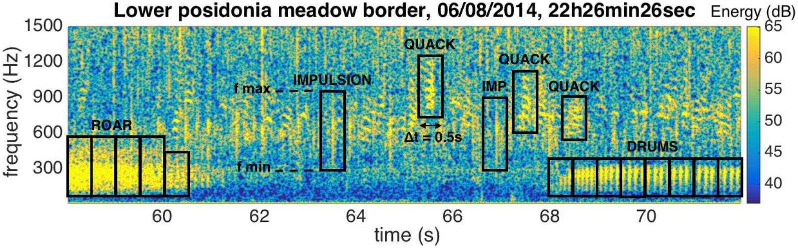
supervised or unsupervised classification



☞ Predict, 6 months in advance, the intensity of an El Niño Southern Oscillation (ENSO) event from ocean-atmosphere datasets (sea level pressure, surface wind components, sea surface temperature, surface air temperature, cloudiness...)

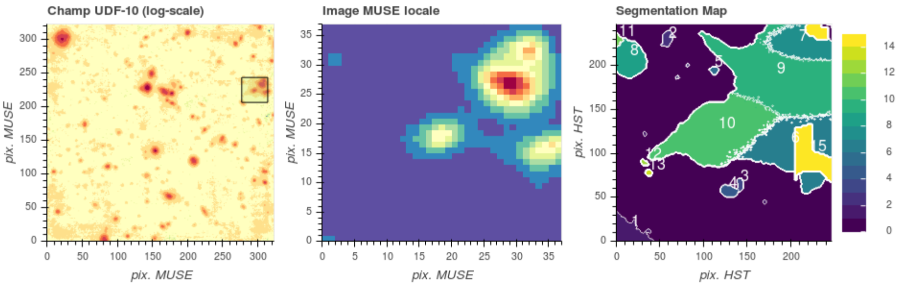
supervised regression





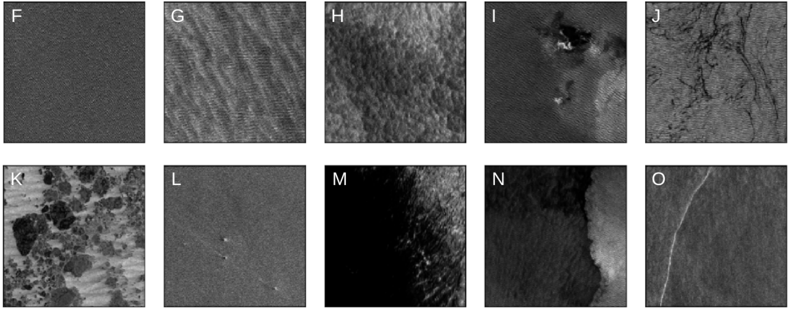
☞ Predict the class of underwater sounds (roar, quack, drums, impulsion) from times series recorded by hydrophones ($f_s = 156kHz$)

supervised or unsupervised classification



☞ Predict galaxy spectra from both hyperspectral MUSE datacubes and Hubble Space Telescope images for better understanding of the early universe

supervised regression



☞ Predict the classes of SAR images of the ocean (convective cells in I, sea ice in K, weather front in N,...) to detect climate-ocean events from water surface roughness

supervised or unsupervised classification

Introduction

Basics

Variable terminology

- Observed data referred to as *input* variables, *predictors* or *features*: X
- Data to predict referred to as *output* variables, or *responses*: Y

Type of prediction problem: regression vs classification

Depending on the type of the *output* variables

- When Y are quantitative data (e.g. ENSO intensity index values): regression
- When Y are categorical data (e.g. handwritten digits $Y \in \{0, \ldots, 9\}$): classification

Two very close problems

Assumptions

- Input variables X_i are vectors in $\mathbb{R}^p$:

$$X_i = (X_{i,1}, \ldots, X_{i,p})^T \in \mathcal{X} \subset \mathbb{R}^p$$

- Output variables Y_i take values:
  - In $\mathcal{Y} \subset \mathbb{R}$ (regression)
  - In a finite set $\mathcal{Y}$ (classification)
- $Y = f(X) + \epsilon$

Prediction rule

Function of prediction / rule of classification $\equiv$ function $\hat{f} : \mathcal{X} \rightarrow \mathcal{Y}$ to get predictions of new elements Y given X

$$\widehat{Y} = \hat{f}(X)$$

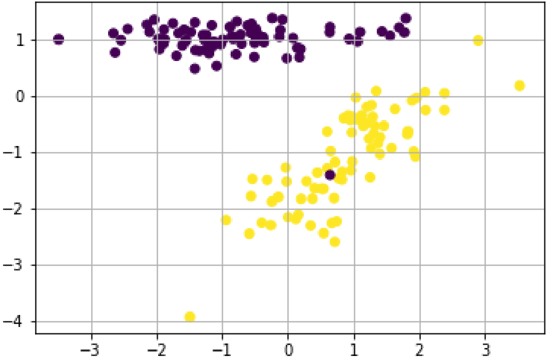
Training set $\equiv$ available sample $\mathcal{T}$ to learn the prediction rule f

For a sized n training set, different cases:

- Supervised learning: $\mathcal{T} \equiv \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ are available
- Unsupervised learning: $\mathcal{T} \equiv (X_1, \ldots, X_n)$ are available only
- Semi-supervised: mixed scenario (often encountered in practice, but less information than in the supervised case)

Introduction

Toy Example


1
0
−1
−2
−3
−4
−3
−2
−1
0
1
2
3

We seek a prediction model based on the linear regression of the outputs $Y \in \{-1, 1\}$:

$$Y = \beta_1 X_1 + \beta_2 X_2 + \epsilon,$$

where $\boldsymbol{\beta} = (\beta_1, \beta_2)^T$ is a 2D unknown parameter vector

Learning problem ⇔ Estimation of $\boldsymbol{\beta}$

Least Squares Estimator $\hat{\boldsymbol{\beta}} = (\hat{\beta}_1, \hat{\beta}_2)^T$: minimize the training error rate (quadratic cost sense)

$$RSS(\boldsymbol{\beta}) = \sum_{i=1}^{N} (Y_i - \beta_1 X_{i,1} - \beta_2 X_{i,2})^2$$

Classification rule based on least squares regression

$$f(X) = \begin{cases} 1 \text{ if } \widehat{Y} = \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 \geq 0, \\ -1 \text{ otherwise} \end{cases}$$

Notebook

Most of methods have a complexity related to their *effective* number of parameters

Linear classification: model order p

E.g. dth degree polynomial regression: $p = d + 1$ parameters a_k s.t.

$$
\begin{aligned}
Y &= \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_d x^d + \epsilon, \\
&= \mathbf{X}_d \boldsymbol{\beta}_d + \epsilon,
\end{aligned}
$$

where

$$
\begin{aligned}
\mathbf{X}_d &= \left[1,\ x,\ x^2, \ldots, x^d\right], \\
\boldsymbol{\beta}_d &= [\beta_0, \beta_1, \beta_2, \ldots, \beta_d]^T.
\end{aligned}
$$

Notebook

Error rate vs polynomial order d

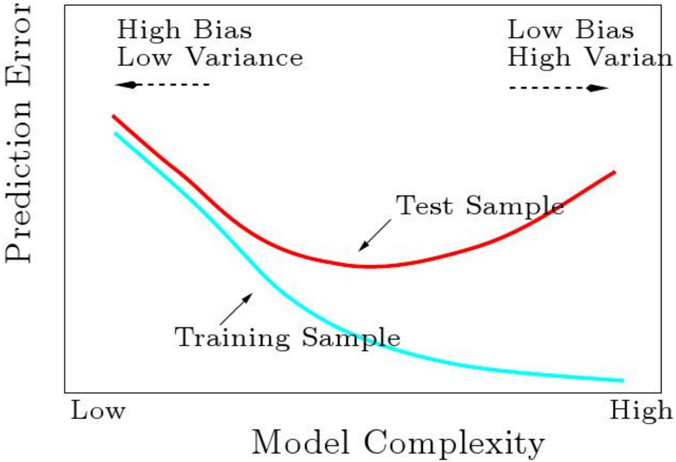
Notebook

- Training error rate (i.e. error rate for train data used for learning) minimized when $d = 19$
- True error rate (i.e. error rate for test data not used for learning) minimized when $d = 5$ ...

☞ Training error always decrease with the model complexity. Can't use alone to select the model!

Fundamental trade-off

- Too simple model (high bias) → under-fitting
- Too complex model (high variance) → over-fitting



If the true model is

$$Y = f(X) + \epsilon,$$

then for any prediction rule $\widehat{f}(X)$, Mean Squared Error (MSE) expresses as

$$E\left[\left(Y - \widehat{f}(x)\right)^2\right] = \mathrm{Var}\left[\widehat{f}(x)\right] + \mathrm{Bias}\left[\widehat{f}(x)\right]^2 + \mathrm{Var}\left[\epsilon\right]$$

- $\mathrm{Var}\left[\epsilon\right]$ is the *irreducible* part
- as the flexibility of $\widehat{f} \nearrow$, its variance $\nearrow$ and the bias $\searrow$

☞ overfitting/underfitting trade-off

Model based approches for classification

Model based approches for classification

Bayes Classifier

Classification problem with K classes: $Y \in \mathcal{Y} = \{1, \ldots, K\}$,

Probability of class $Y = k$ given $X = x$

Bayes rule:

$$p(Y = k|X = x) = \frac{p(x|Y = k)p(Y = k)}{p(x)} = \frac{p(x|Y = k)p(Y = k)}{\sum_{j=1}^{K} p(x|Y = j)p(Y = j)},$$
$$= \frac{\pi_k \, p_k(x)}{\sum_{j=1}^{K} \pi_j \, p_j(x)}$$

- $p_k(x) \equiv p(x|Y = k)$ is the *density* for X in class k
- $\pi_k \equiv p(Y = k)$ is the *weight*, or *prior* probability of class k

Definition

The Bayes classification rule f^* is defined as

$$f^*(x) = \arg\max_{k\in\mathcal{Y}} p(Y = k|X = x).$$

Theorem

The Bayes classification rule f^* is optimal in the misclassification rate sense where $\mathcal{E}[f] = p(f(X) \neq Y)$:

$$\text{for any rule } f,\ \mathcal{E}[f] \geq \mathcal{E}[f^*],$$

Remarks

- $f^*(X) \equiv$ *maximum a posteriori* (MAP) estimate
- In real-word applications, the distribution of (X, Y) is unknown $\Rightarrow$ no analytical expression of $f^*(X)$. But useful reference on academic examples.

Two kinds of approaches based on a model:

1. **Discriminative approaches**: direct learning of $p(Y|X)$,
   e.g. SVM, logistic regression
2. **Generative models**: learning of the joint distribution $p(X, Y)$

$$p(X,Y) = \underbrace{p(X|Y)}_{\text{likelihood}} \underbrace{\Pr(Y)}_{\text{prior}},$$

   e.g. linear/quadratic discriminant analysis, Naïve Bayes

Assumptions

- classification problem with K classes: $Y \in \mathcal{Y} = \{1, \ldots, K\}$,
- input variables: $X \in \mathbb{R}^p$

Bayes rule:

$$p(Y = k|X = x) = \frac{p(x|Y = k)p(Y = k)}{\sum_{j=1}^{K} p(x|Y = j)p(Y = j)}.$$

In practice, the following quantities are unknown:

- densities of each class $p_k(x) \equiv p(x|Y = k)$
- weights, or prior probabilities, of each class $\pi_k \equiv p(Y = k)$

Estimation problem

These quantities must be learned on a training set:

learning problem $\Leftrightarrow$ estimation problem in a parametric or not way

Model based approches for classification

Linear/Quadratic Discriminant Analysis

Supervised classification assumptions

- $X \in \mathbb{R}^p$, $Y \in \mathcal{Y} = \{1, \dots, K\}$,
- sized n training set $(X_1, Y_1), \dots (X_n, Y_n)$

QDA Assumptions

The input variables X, given a class $Y = k$, are distributed according to a parametric and Gaussian distribution:

$$X|Y = k \sim \mathcal{N}(\mu_k, \Sigma_k) \Leftrightarrow p_k(x) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)}$$

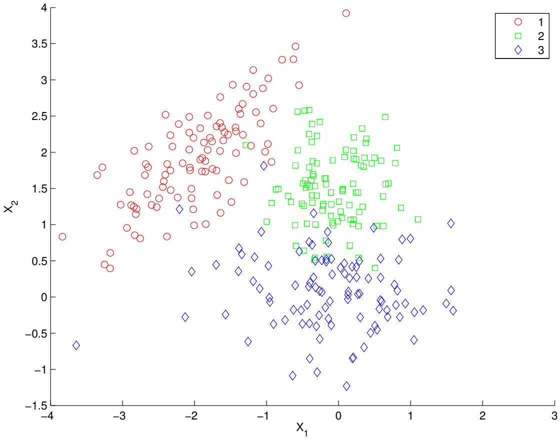
The Gaussian parameters are, for each class $k = 1, \dots, K$

- mean vectors $\mu_k \in \mathbb{R}^p$,
- covariance matrices $\Sigma_k \in \mathbb{R}^{p \times p}$,

☞ set of parameters $\theta_k \equiv \{\mu_k, \Sigma_k\}$, plus the weights π_k, for $k = 1, \dots, K$.

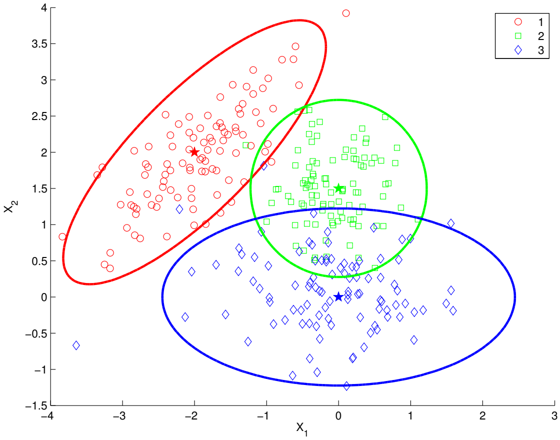
Mixture of $K = 3$ Gaussians

- $Y \in \{1, 2, 3\}$
- $X \in \mathbb{R}^2$



Mixture of $K = 3$ Gaussians

- $Y \in \{1, 2, 3\}$
- $X \in \mathbb{R}^2$



For the training set,

$$\begin{aligned}
\ell\left(\theta_1,\dots,\theta_K,\pi_1,\dots,\pi_{K-1}\right) &= \log p\left((x_1,y_1),\dots,(x_n,y_n)\right),\\
&= \sum_{i=1}^{n} \log p\left((x_i,y_i)\right), \quad \leftarrow \text{ i.i.d. training set},\\
&= \sum_{i=1}^{n} \log\left[p\left(x_i|y_i\right) p(y_i)\right],\\
&= \sum_{i=1}^{n} \log\left[\pi_{y_i}\ p_{y_i}\left(x_i;\theta_{y_i}\right)\right].
\end{aligned}$$

Rk: $\pi_K = 1 - \sum_{j=1}^{K-1} \pi_j$ is not a parameter

Notations

- $n_k = \#\{y_i = k\}$ is the number of training samples in class k,
- $\sum_{y_i=k}$ is the sum over all the indices i of the training samples in class k

(Unbiased) Maximum likelihood estimators (MLE)

- $\widehat{\pi}_k = \dfrac{n_k}{n}, \quad \leftarrow$ sample proportion
- $\widehat{\mu}_k = \dfrac{\sum_{y_i=k} x_i}{n_k}, \quad \leftarrow$ sample mean
- $\widehat{\Sigma}_k = \frac{1}{n_k - 1} \sum_{y_i=k} (x_i - \widehat{\mu}_k)(x_i - \widehat{\mu}_k)^T, \quad \leftarrow$ sample covariance

Rk: $\frac{1}{n_k-1}$ is a bias correction factor for the covariance MLE (otherwise $\frac{1}{n_k}$)

For model based approaches, Bayes classifier is defined as

$$f^*(x) = \arg\max_{k \in \mathcal{Y}} p(Y = k | X = x)$$

- equivalent to consider a set of functions $\delta_k(x)$, for $k \in \mathcal{Y}$, derived from a monotone transformation of posterior probability $p(Y = k | X = x)$
- decision boundary between classes k and l is then defined as the set $\{x \in \mathcal{X} \ : \ \delta_k(x) = \delta_l(x)\}$

Definition

$\delta_k(x)$ are called the discriminant functions of each class k

☞ x is predicted in the k_0 class such that $k_0 = \arg\max_{k \in \mathcal{Y}} \delta_k(x)$

The classification rule becomes

$$f(x) = \arg\max_{k \in \mathcal{Y}} p(Y = k | X = x, \widehat{\theta}, \widehat{\pi}),$$
$$= \arg\max_{k \in \mathcal{Y}} \underbrace{\log p(Y = k | X = x, \widehat{\theta}, \widehat{\pi})}_{\delta_k(x)},$$

where

$$\delta_k(x) = -\frac{1}{2} \log \left|\widehat{\Sigma}_k\right| - \frac{1}{2}(x - \widehat{\mu}_k)^T \widehat{\Sigma}_k^{-1} (x - \widehat{\mu}_k) + \log \widehat{\pi}_k + \cancel{\text{Cst}},$$

is the discriminant function

Remarks

1. different rule than the Bayes classifier as θ replaced by $\widehat{\theta}$ (and π replaced by $\widehat{\pi}$)
2. when $n \gg p$, $\widehat{\theta} \to \theta$ (and $\widehat{\pi} \to \pi$): convergence to the optimal classifier... only if the Gaussian model is correct!

The boundary between two classes k and l is described by the equation

$$\delta_k(x) = \delta_l(x) \Leftrightarrow C_{k,l} + L_{k,l}^T x + x^T Q_{k,l}^T x = 0, \quad \leftarrow \text{quadratic equation}$$

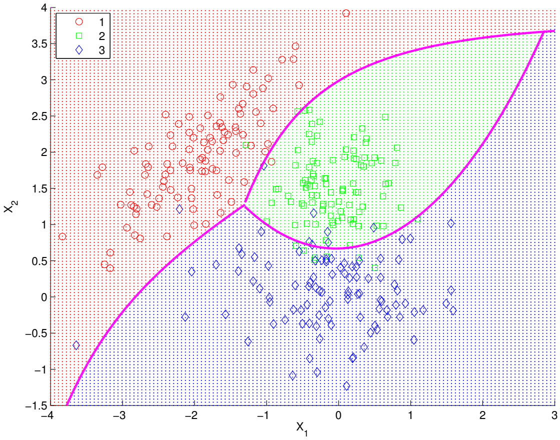
where

- $C_{k,l} = -\frac{1}{2} \log \frac{|\widehat{\Sigma}_k|}{|\widehat{\Sigma}_l|} + \log \frac{\widehat{\pi}_k}{\widehat{\pi}_l} - \frac{1}{2} \widehat{\mu}_k^T \widehat{\Sigma}_k^{-1} \widehat{\mu}_k + \frac{1}{2} \widehat{\mu}_l^T \widehat{\Sigma}_l^{-1} \widehat{\mu}_l, \quad \leftarrow \text{scalar}$
- $L_{k,l} = \widehat{\Sigma}_k^{-1} \widehat{\mu}_k - \widehat{\Sigma}_l^{-1} \widehat{\mu}_l, \quad \leftarrow \text{vector in } \mathbb{R}^p$
- $Q_{k,l} = \frac{1}{2} \left(-\widehat{\Sigma}_k^{-1} + \widehat{\Sigma}_l^{-1} \right), \quad \leftarrow \text{matrix in } \mathbb{R}^{p \times p}$

☞ Quadratic discriminant analysis

Mixture of $K = 3$ Gaussians

- Classification rule: $\arg\max_{k=1,2,3} \delta_k(x)$
- Quadratic boundaries $\{x; \delta_k(x) = \delta_l(x)\}$



LDA Assumptions

Additional simplifying assumption w.r.t. QDA: all the class covariance matrices are identical ("homoscedasticity"), i.e. $\Sigma_k = \Sigma$, for $k = 1, \dots, K$

(Unbiased) Maximum likelihood estimators (MLE)

- $\widehat{\pi}_k$ and $\widehat{\mu}_k$ are unchanged,
- $\widehat{\Sigma} = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{y_i=k} (x_i - \widehat{\mu}_k)(x_i - \widehat{\mu}_k)^T, \quad \leftarrow$ pooled covariance

Rk: $\frac{1}{n-K}$ is a bias correction factor for the covariance MLE (otherwise $\frac{1}{n}$)

LDA discriminant function

$$\delta_k(x) = -\frac{1}{2} \log \left|\widehat{\Sigma}\right| - \frac{1}{2}(x - \widehat{\mu}_k)^T \widehat{\Sigma}^{-1}(x - \widehat{\mu}_k) + \log \widehat{\pi}_k + \cancel{\mathrm{Cst}},$$

The boundary between two classes k and l reduces to the equation

$$\delta_k(x) = \delta_l(x) \Leftrightarrow C_{k,l} + L_{k,l}^T x = 0, \quad \leftarrow \text{linear equation}$$

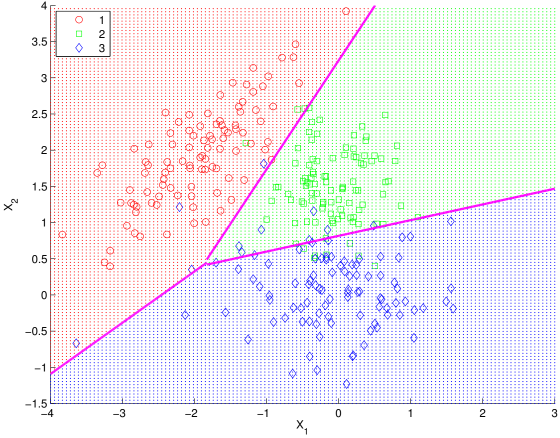
where

- $C_{k,l} = \log \frac{\widehat{\pi}_k}{\widehat{\pi}_l} - \frac{1}{2}\widehat{\mu}_k^T \widehat{\Sigma}^{-1}\widehat{\mu}_k + \frac{1}{2}\widehat{\mu}_l^T \widehat{\Sigma}^{-1}\widehat{\mu}_l, \quad \leftarrow \text{scalar}$
- $L_{k,l} = \widehat{\Sigma}^{-1}\left(\widehat{\mu}_k - \widehat{\mu}_l\right), \quad \leftarrow \text{vector in } \mathbb{R}^p$
- $Q_{k,l} = 0,$

☞ Linear discriminant analysis

Mixture of $K = 3$ Gaussians

- Classification rule: $\arg\max_{k=1,2,3} \delta_k(x)$
- linear boundaries $\{x; \delta_k(x) = \delta_l(x)\}$



Effective number of parameters

- LDA: $(K-1) \times (p+1) = O(Kp)$
- QDA: $(K-1) \times \left(\frac{p(p+3)}{2} + 1\right) = O(Kp^2)$

Remarks

- In high dimension, i.e. $p \approx n$ or $p > n$, LDA is more stable than QDA which is more prone to overfitting,
- Both methods appear however to be robust on a large number of real-word datasets
- LDA can be viewed in some cases as a least squares regression method
- LDA performs a dimension reduction to a subspace of dimension $\leq K-1$ generated by the vectors $z_k = \Sigma^{-1}\widehat{\mu}_k \leftarrow$ dimension reduction from p to $K-1$!

Generative models

- learning/estimation of $p(X, Y) = p(X|Y)p(Y)$,
- derivation of $p(Y|X)$ from Bayes rule,

Different assumptions on the class densities $p_k(x) = p(X = x|Y = k)$

- QDA/LDA: Gaussian parametric model

☞ performs well on many real-word datasets

☞ LDA is especially useful when n is small

Perspectives

Model free approaches: direct learning of the prediction rule f

Notebook

Model free approaches for classification

Model free approaches for classification

K Nearest Neighbors (K-NN)

Binary classification problem

For a binary classification problem $Y \in \{0, 1\}$, the classification rule can be derived, for $X = x$, as

$$f(x) = \begin{cases} 1 \text{ if } \widehat{Y}(x) > \frac{1}{2}, \\ 0 \text{ otherwise} \end{cases}$$

where $\widehat{Y}(x) = \frac{1}{k} \sum_{X_i \in N_k(x)} Y_i$ is the average of the binary labels of the k nearest neighbors of the testing point $X = x$.

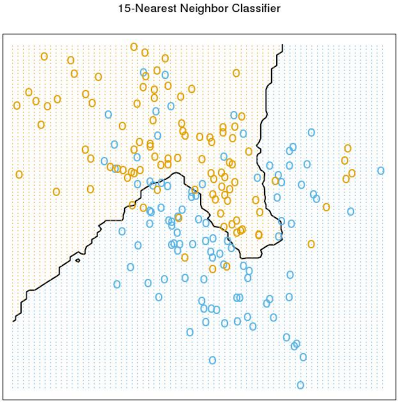
Classification rule associated with k-NN

The binary classification problem can be directly extended for an arbitray number of class K:

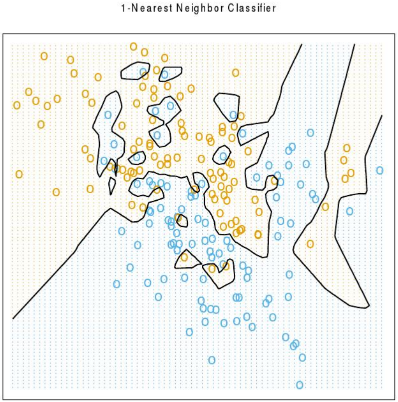
$$\begin{aligned} f(x) &\equiv \text{ majority vote among the } k \text{ closest neighbors of the testing point } x, \\ &\equiv \text{ assignement to the most common class among the } k \text{ nearest neighbors} \end{aligned}$$

k-NN: complexity parameter k

The effective number of parameters expresses as $N_{\text{eff}} = \frac{n}{k}$, where n is the size of the training sample

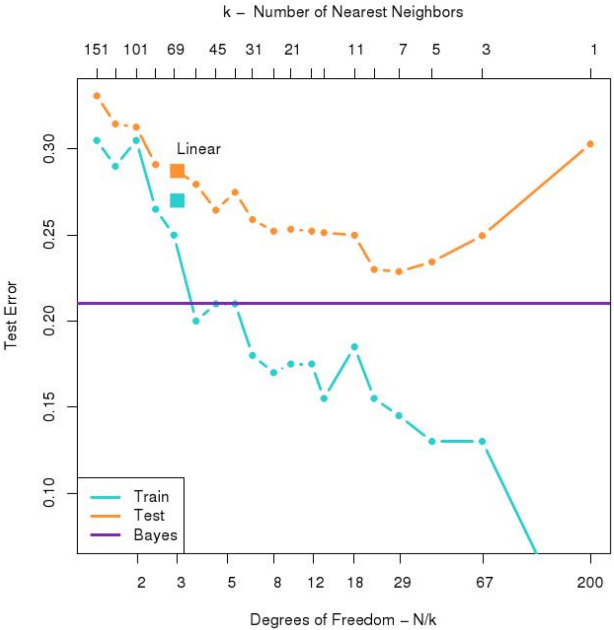


$k = 15$, $N_{\text{eff}} \approx 13$



$k = 1$, $N_{\text{eff}} \approx 200$

- $k = 1 \rightarrow$ training error is always 0 !


k – Number of Nearest Neighbors
151
101
69
45
31
21
11
7
5
3
1
Linear
Test Error
0.30
0.25
0.20
0.15
0.10
Train
Test
Bayes
2
3
5
8
12
18
29
67
200
Degrees of Freedom – N/k

Model free approaches for classification

Support Vector Machine (SVM)

Theory elaborated in the early 1990's (Vapnik *et al*) based on the idea of 'maximum margin'

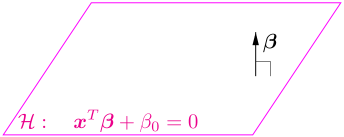
- deterministic criterion learned on the training set $\leftarrow$ supervised classification
- ☞ general, i.e. model free, linear classification rule
- ☞ classification rule is linear in a transformed space of higher (possible infinite) dimension than the original input feature/predictor space

Binary classification problem

- $X \in \mathbb{R}^p$
- $Y \in \{-1, 1\} \leftarrow 2$ classes
- Training set (x_i, y_i), for $i = 1, \ldots, n$

Defining a linear discriminant function $h(x) \Leftrightarrow$ defining a separating hyperplane $\mathcal{H}$ with equation

$$\mathbf{x}^T \boldsymbol{\beta} + \beta_0 = 0,$$



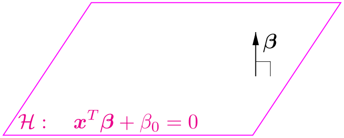
- $\boldsymbol{\beta} \in \mathbb{R}^p$ is the normal vector (vector normal to the hyperplane $\mathcal{H}$),
- $\beta_0 \in \mathbb{R}$ is the intercept/offset (regression or geometrical interpretation)

☞ $\mathcal{H}$ is an *affine subspace* of dimension $p - 1$

☞ $h(x) \equiv \mathbf{x}^T \boldsymbol{\beta} + \beta_0$ is the associated (linear) discriminant function

For a given separating hyperplane $\mathcal{H}$ with equation

$$\mathbf{x}^T\boldsymbol{\beta} + \beta_0 = 0,$$



the prediction rule can be expressed as

- $\widehat{y} = +1$, if $h(\mathbf{x}) = \mathbf{x}^T\boldsymbol{\beta} + \beta_0 \geq 0$,
- $\widehat{y} = -1$, otherwise,

or in an equivalent way:

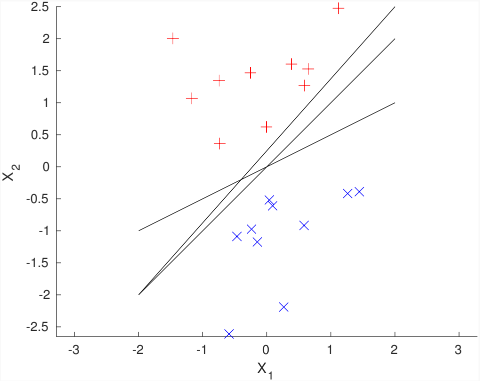
$$\widehat{y} \equiv G(\mathbf{x}) = \text{sign}\left[\mathbf{x}^T\boldsymbol{\beta} + \beta_0\right]$$

Rk: $\mathbf{x}$ is in class $y \in \{-1, 1\}$: prediction $G(\mathbf{x})$ is correct iff $y\left(\mathbf{x}^T\boldsymbol{\beta} + \beta_0\right) \geq 0$

Linear separability assumption: $\exists \boldsymbol{\beta} \in \mathbb{R}^p$ and $\beta_0 \in \mathbb{R}$ s.t. the hyperplane $\boldsymbol{x}^T\boldsymbol{\beta} + \beta_0 = 0$ perfectly separates the two classes on the training set:

$$y_k \left(x_k^T \boldsymbol{\beta} + \beta_0 \right) \geq 0, \quad \text{for } k = 1, \ldots, n,$$

Separable case ($p = 2$ example)

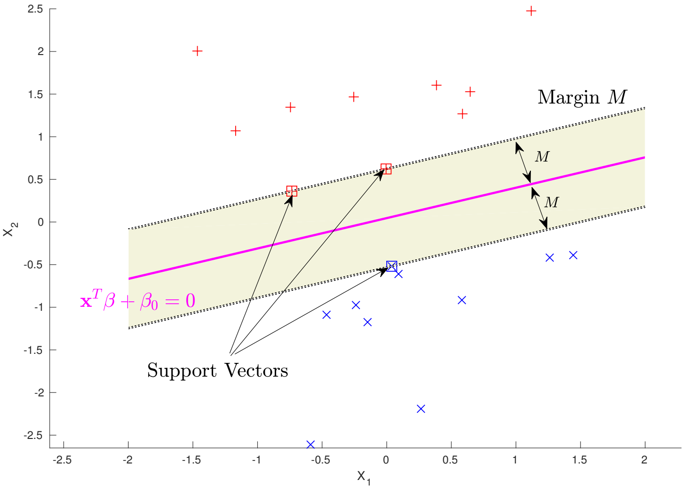


Pb: infinitely many possible perfect separating hyperplanes $\boldsymbol{x}^T\boldsymbol{\beta} + \beta_0 = 0$

☞ Find the 'optimal' separating hyperplane

Maximum margin principle

We are interested in the 'optimal' perfect separating hyperplane maximizing the distance $M > 0$, called the margin, between the separating hyperplane and the training data, i.e. with the biggest gap



Find $\boldsymbol{\beta} \in \mathbb{R}^p$ and $\beta_0 \in \mathbb{R}$ s.t. the margin

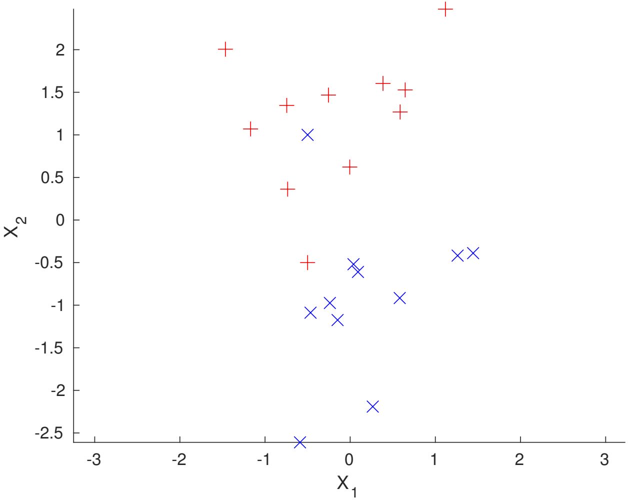
$$M = \min_{1 \le k \le n} \{d(x_k, \mathcal{H})\}$$

is maximized. Subject to

$$y_k \left(x_k^T \boldsymbol{\beta} + \beta_0 \right) \ge 0, \quad \text{for } k = 1, \dots, n,$$

Nonseparable case

- in general, overlap of the 2 classes (unless $n < p$)
- no hyperplane that perfectly separates the training data



Solution for the nonseparable case

Considering a *soft-margin* that allows wrong classifications

- introduction of *slack variables* $\xi_i \geq 0$ s.t.

$$y_i(\mathbf{x_i}^T\boldsymbol{\beta} + \beta_0) \geq (1 - \xi_i)$$

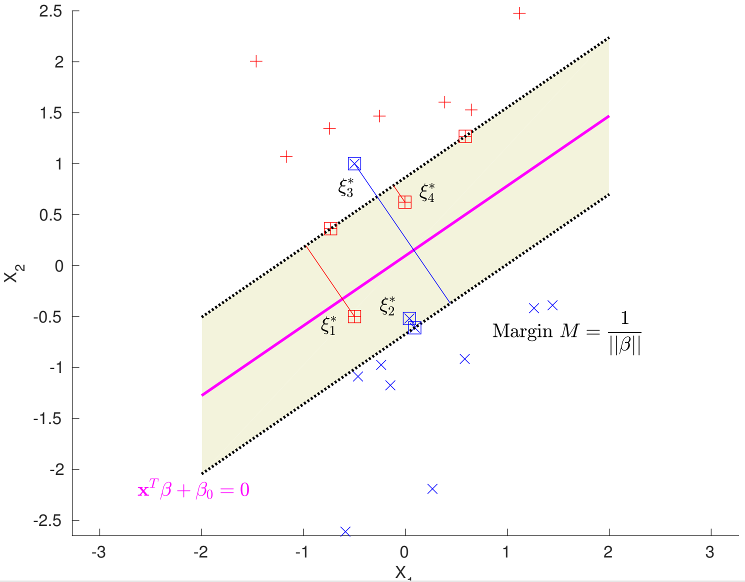
  Support vectors include now the wrong classified points, and the points inside the margins ($\xi_i > 0$)

- Primal problem: adding a constraint on the ξ_i's

$$\begin{cases} \max_{\boldsymbol{\beta},\beta_0,\xi} & M, \\ \text{subject to} & y_i(\mathbf{x_i}^T\boldsymbol{\beta} + \beta_0) \geq 1 - \xi_i, \\ & \sum_{i=1}^n \xi_i \leq C. \end{cases}$$

  where $C > 0$ is the “cost” parameter

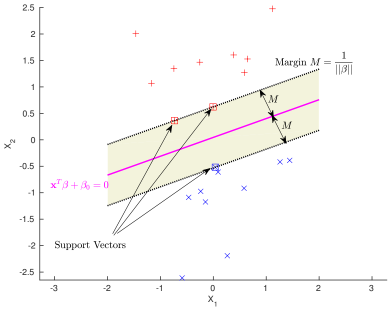
Example (nonseparable case)



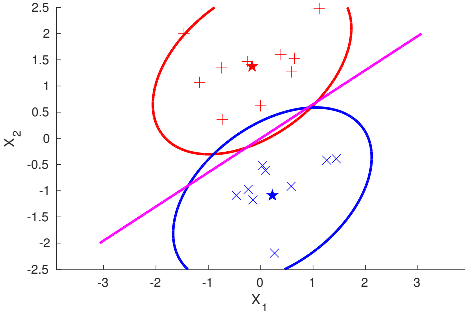
$\xi_i^* \equiv M\xi_i \leftarrow$ distance between a support vector and the margin

Linear discrimination

- Linear Discriminant Analysis (LDA): Gaussian generative model
- SVM: criterion optimization (maximizing the margin)

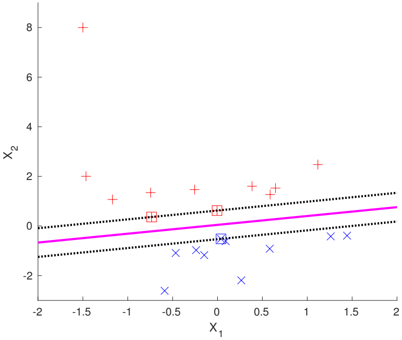


SVM

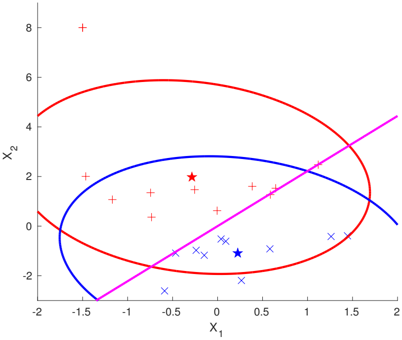


LDA

Adding one atypical data



SVM

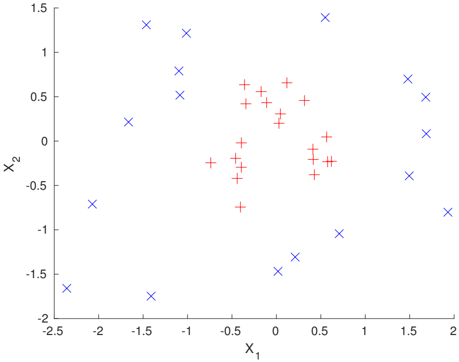


LDA

SVM property

- Nonsensitive to atypical points (outliers) far from the margin

☞ sparse method (information ≡ support vectors)



Transformed space $\mathcal{F}$

- Choice of a transformed space $\mathcal{F}$ (expansion space) where the linear separation assumption is more relevant
- Nonlinear expansion map $\phi : \mathbb{R}^p \to \mathcal{F}, \mathbf{x} \mapsto \phi(\mathbf{x}) \leftarrow$ enlarged features

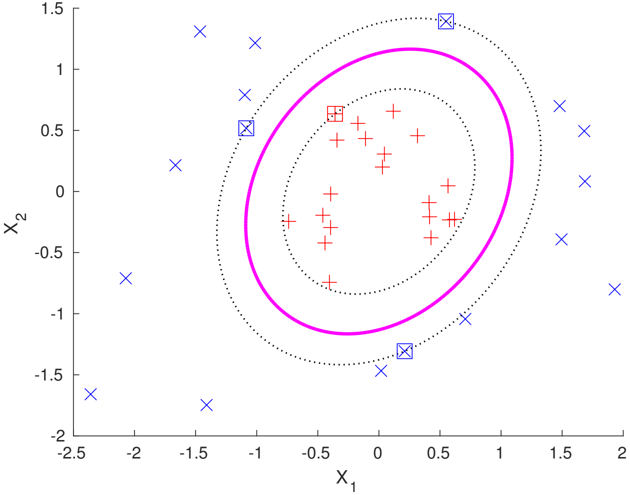
- Projection in the space of monomials of order 2.

$$\begin{aligned}\phi : \mathbb{R}^2 &\to \mathbb{R}^3 \\ \mathrm{x} &\mapsto \phi(\mathrm{x}) \\ (\mathrm{x}_1, \mathrm{x}_2) &\mapsto (\mathrm{x}_1^2, \mathrm{x}_2^2, \sqrt{2}\mathrm{x}_1\mathrm{x}_2)\end{aligned}$$

- In $\mathbb{R}^3$, the inner product can be expressed as

$$\begin{aligned}\langle \phi(\mathrm{x}), \phi(\mathrm{x}') \rangle_{\mathbb{R}^3} &= \sum_{i=1}^{3} \phi(\mathrm{x})_i \phi(\mathrm{x}')_i \\ &= \phi(\mathrm{x})_1\phi(\mathrm{x}')_1 + \phi(\mathrm{x})_2\phi(\mathrm{x}')_2 + \phi(\mathrm{x})_3\phi(\mathrm{x}')_3 \\ &= \mathrm{x}_1^2 {\mathrm{x}'}_1^2 + \mathrm{x}_2^2 {\mathrm{x}'}_2^2 + 2\mathrm{x}_1\mathrm{x}_2{\mathrm{x}'}_1{\mathrm{x}'}_2 \\ &= (\mathrm{x}_1{\mathrm{x}'}_1 + \mathrm{x}_2{\mathrm{x}'}_2)^2 \\ &= \langle \mathrm{x}, \mathrm{x}' \rangle_{\mathbb{R}^2}^2 \\ &= k(\mathrm{x}, \mathrm{x}').\end{aligned}$$

- $X \in \mathbb{R}^2$, $\phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)^T$



Linear separation in the feature space $\mathcal{F}$ $\Rightarrow$ Nonlinear separation in the input space

The SVM solution depends only on the inner product between the input features $\phi(x)$ and the support vectors $\phi(x_{\text{margin}})$

Kernel trick

Use of a kernel function k associated with an expansion/feature map ϕ:

$$
\begin{aligned}
k: \quad \mathbb{R}^p \times \mathbb{R}^p &\to \mathbb{R} \\
(x, x') &\mapsto k(x, x') \equiv \langle \phi(x), \phi(x') \rangle
\end{aligned}
$$

Advantages

- Computations are performed in the original input space: less expansive than in a high dimensional transformed space $\mathcal{F}$
- Explicit representations of the feature map ϕ and enlarged feature space $\mathcal{F}$ are not necessary, the only expression of k is required!
- ☞ Possibility of complex transformations in possible infinite space $\mathcal{F}$
- ☞ Standard trick in machine learning not limited to SVM (kernel-PCA, gaussian process, kernel ridge regression, spectral clustering ...)

Definition (Positive semi-definite kernel)

$k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is positive semi-definite is

- $\forall (\mathbf{x}, \mathbf{x}') \in \mathbb{R}^d \times \mathbb{R}^d, k(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_j, \mathbf{x}_i)$.
- $\forall n \in \mathbb{N}, \forall \xi_1 \ldots \xi_n \in \mathbb{R}, \forall \mathbf{x}_1 \ldots \mathbf{x}_n \in \mathbb{R}^d, \sum_{i,j}^n \xi_i \xi_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0$.

Theorem (Moore-Aronsjan (1950))

To every positive semi-definite kernel k, there exists a Hilbert space $\mathcal{H}$ and a feature map $\phi : \mathbb{R}^d \to \mathcal{H}$ such that for all $\mathbf{x}_i, \mathbf{x}_j$ we have $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle_{\mathcal{H}}$.

Let k_1 and k_2 be positive semi-definite, and $\lambda_{1,2} > 0$ then:

1. $\lambda_1 k_1$ is a valid kernel
2. $\lambda_1 k_1 + \lambda_2 k_2$ is positive semi-definite.
3. $k_1 k_2$ is positive semi-definite.
4. $\exp(k_1)$ is positive semi-definite.
5. $g(\mathbf{x}_i)g(\mathbf{x}_j)$ is positive semi-definite, with $g : \mathbb{R}^d \to \mathbb{R}$.

Usual kernel functions

- Linear kernel ($\mathcal{F} \equiv \mathbb{R}^p$) : $k(x, x') = x^T x'$
- Polynomial kernel (dimension of $\mathcal{F}$ increases with the order d)

$$k(x, x') = (x^T x' + q)^d = \sum_{l=1}^{d} \binom{d}{l} q^{d-l} (x^T x')^l.$$

- Gaussian radial function ($\mathcal{F}$ with infinite dimension)

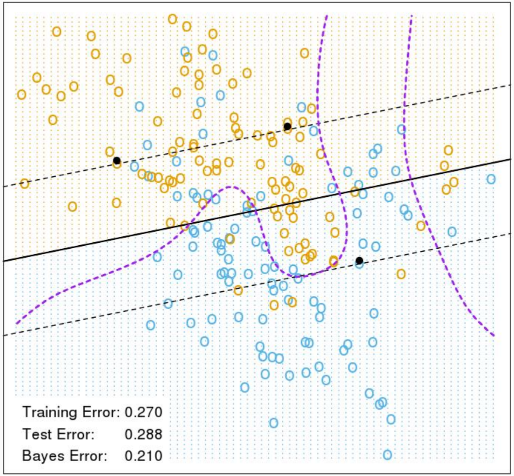
$$k(x, x') = \exp\left(-\gamma ||x - x'||^2\right)$$

- Neural net kernel ($\mathcal{F}$ with infinite dimension)

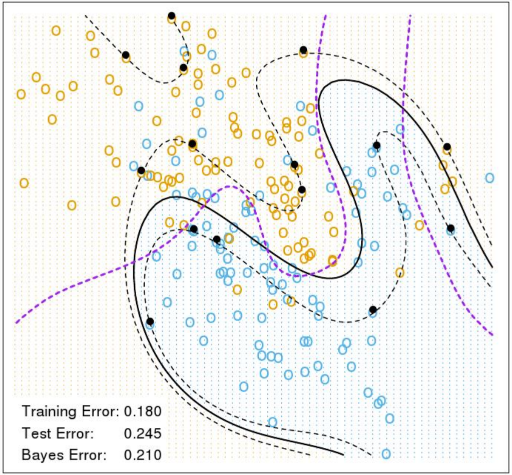
$$k(x, x') = \tanh\left(\kappa_1 x^T x' + \kappa_2\right)$$

☞ standard practice is to estimate optimal values of kernel parameters by cross validation

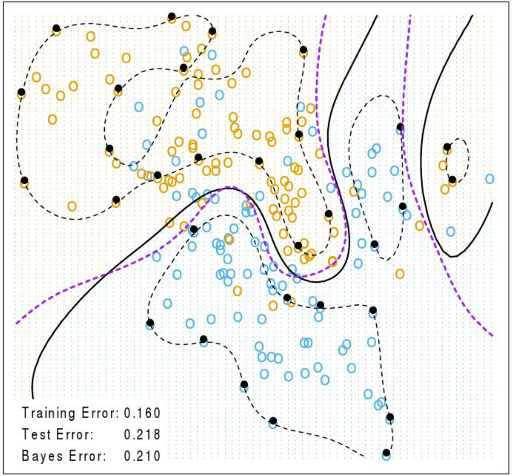
Linear kernel



Polynomial kernel ($d = 4$)



Gaussian radial kernel ($\gamma = 1$)



SCALE YOUR DATA!!

- With Gaussian kernel

$$
\begin{aligned}
k(x, x') &= \exp\left(-\gamma ||x - x'||^2\right) \\
&= \exp\left(-\gamma \sum_{i=1}^{p} (x_i - x_i')^2\right)
\end{aligned}
$$

- Scaling:

$$
\begin{aligned}
\tilde{x}_i &= \frac{x_i - \mu_i}{\sigma_i} \\
\tilde{x}_i &= \frac{x_i - \min_i}{\max_i - \min_i}
\end{aligned}
$$

- Notebook

- $Y \in \{1, \ldots, K\} \leftarrow K$ classes

Standard approach: direct generalization by using multiple binary SVMs

OVA: one-versus-all strategy

- K classifiers between one class ($+1$ label) versus all the other classes (-1 label)

☞ classifier with the highest confidence value (e.g. the maximum distance to the separator hyperplane) assigns the class

OVO: one-versus-one strategy

- $\binom{K}{2} = K(K-1)/2$ classifiers between every pair of classes

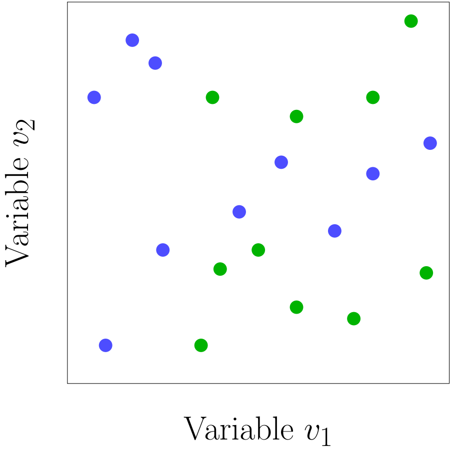
☞ majority vote rule: the class with the most votes determines the instance classification

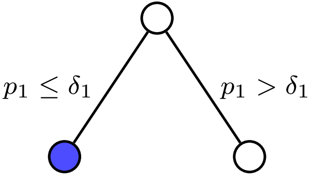
Which to choose? if K is not too large, choose OVO

Model free approaches for classification

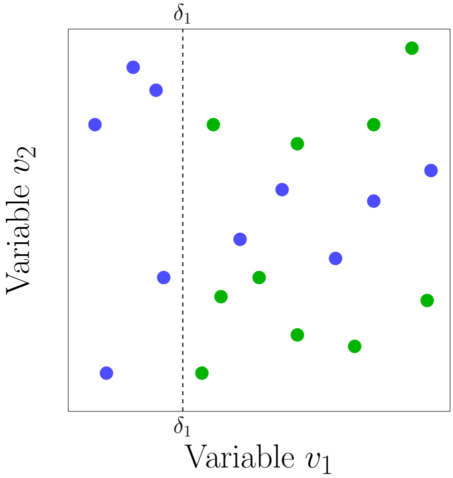
Random Forests

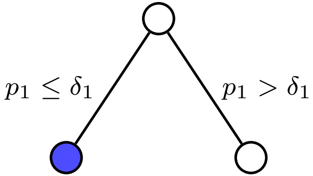
- Introduced in 2001 (Breiman)
- Model free and non linear
- Build a large collection of de-correlated trees and average them
- Combination of weak learner



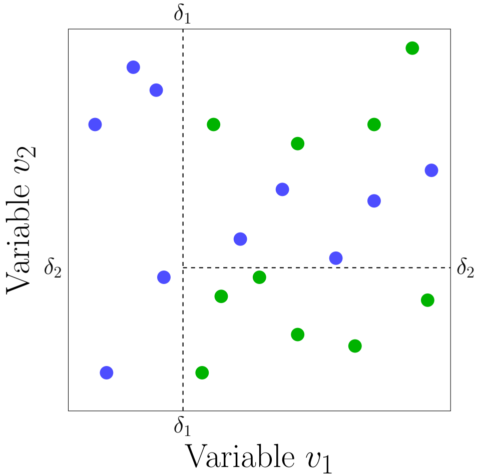


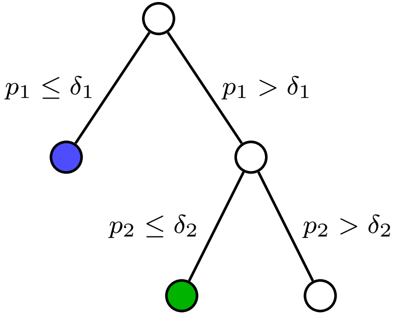
Taken from: Charlotte Pelletier. Cartographie de l'occupation des sols à partir de séries temporelles d'images satellitaires à hautes résolutions Identification et traitement des données mal étiquetées . Interfaces continentales, environnement. Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier), 2017. Français.



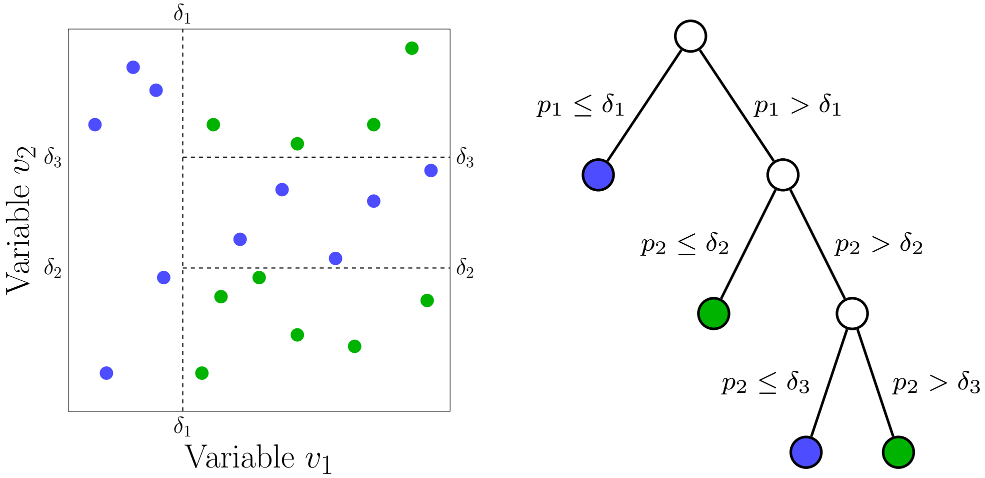


Taken from: Charlotte Pelletier. Cartographie de l'occupation des sols à partir de séries temporelles d'images satellitaires à hautes résolutions Identification et traitement des données mal étiquetées . Interfaces continentales, environnement. Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier), 2017. Français.



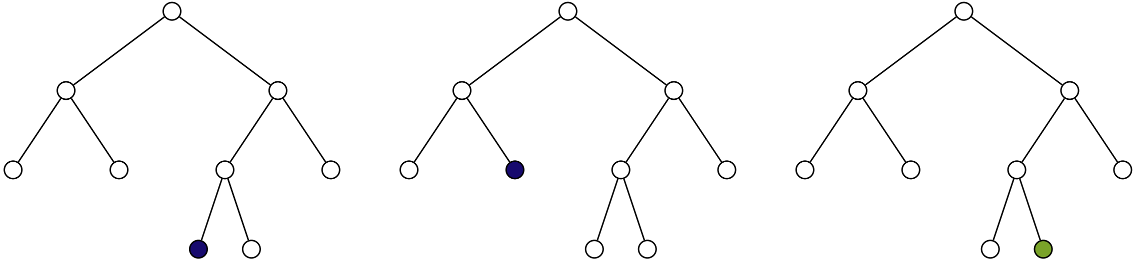


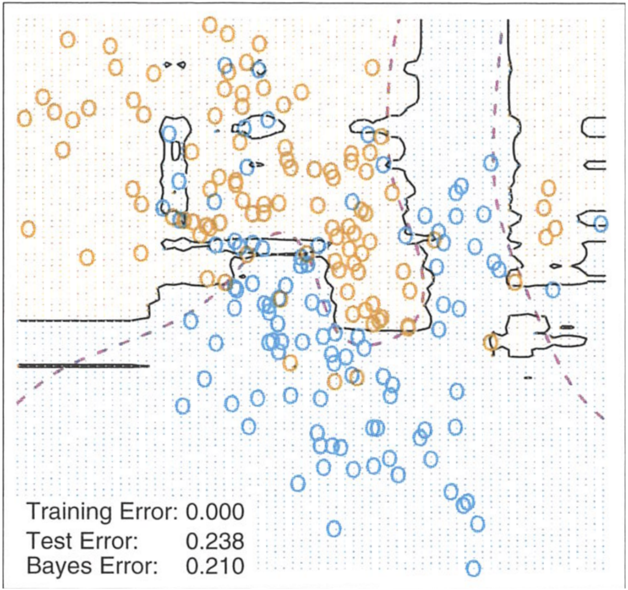
Taken from: Charlotte Pelletier. Cartographie de l'occupation des sols à partir de séries temporelles d'images satellitaires à hautes résolutions Identification et traitement des données mal étiquetées . Interfaces continentales, environnement. Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier), 2017. Français.



Taken from: Charlotte Pelletier. Cartographie de l'occupation des sols à partir de séries temporelles d'images satellitaires à hautes résolutions Identification et traitement des données mal étiquetées . Interfaces continentales, environnement. Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier), 2017. Français.

- For each tree:
  - Draw bootstrap sample X^b for training sample
  - Learn tree, for each node
    - select m features from the initial p features
    - Find the best split (e.g. Gini index, entropy ...)




Training Error: 0.000
Test Error: 0.238
Bayes Error: 0.210

k-NN

- non-parametric method which does not rely on a fixed model
- algorithm which is conceptually among the simplest of all machine learning algorithms
- badly behaved procedure in high dimension: dimension reduction, e.g. PCA, is usually performed prior to k-NN algorithm in order to avoid curse of dimensionality and to reduce computational complexity of the classification rule

SVM

- maximum margin learning criterion $\leftarrow$ model free
- classification algorithm nonlinear in the original input space by performing an implicit linear classification in a higher dimensional space
- sparse solutions characterized by the support vectors
- popular algorithms, with a large literature

Random Forests

- involve decision tree to split the prediction space in simple regions
- combine multiple decision trees to yield a single consensus prediction
- ☞ method able to scale efficiently to high dimensional data

Deep Neural Nets

- Neural Nets with multiple hidden layers between input and output ones
- many variants of deep architectures (Recurrent, Convolutional,...) used in specific domains (speech, vision, ...)
- ☞ supported by empirical evidence
- ☞ dramatic performance jump for several big data applications

Notebook

Feature Extraction/Selection

Feature Extraction/Selection

Motivations

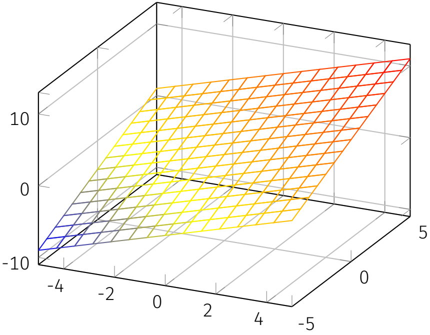
- Curse of dimensionality: it is not possible to get enough data to cover all the observation space.

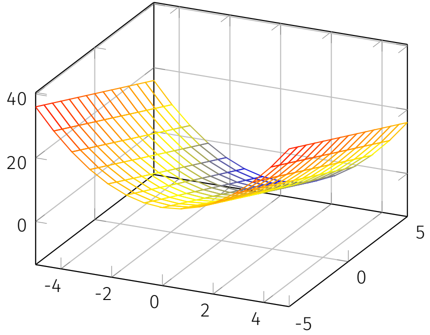
High dimensional spaces are mostly empty !

- Curse of dimensionality: it is not possible to get enough data to cover all the observation space.

High dimensional spaces are mostly empty !

- Multivariate data live in a lower dimensional space, but which one ?





- Feature extraction is important in machine learning because:
  - It reduces the size of the data,
  - It limits the redundancy,
  - It permits visualization of the data,
  - It mitigates the *curse of dimensionality*.
- Extraction techniques:
  - Physically based method,
  - Statistical methods,
  - Linear and non linear filters.

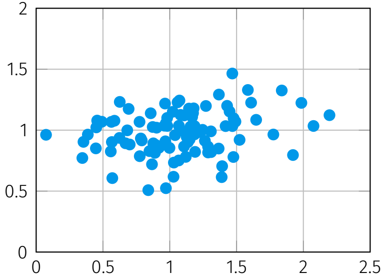
Feature Extraction/Selection

Unsupervised feature extraction

- Linear transformation used to reduce the dimensionality of the data.

$$z_i = \langle \mathbf{v}_i, \mathbf{x} \rangle$$

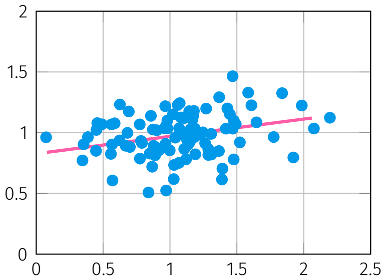
- Find features $\mathbf{z}$ that account for most of the variability of the data:
  - $z_1,\ z_2,\ z_3, \ldots$ are mutually uncorrelated,
  - $\mathrm{var}(z_i)$ is as large as possible,
  - $\mathrm{var}(z_1) > \mathrm{var}(z_2) > \mathrm{var}(z_3) > \ldots$



- Linear transformation used to reduce the dimensionality of the data.

$$z_i = \langle \mathbf{v}_i, \mathbf{x} \rangle$$

- Find features **z** that account for most of the variability of the data:
  - $z_1,\ z_2,\ z_3, \ldots$ are mutually uncorrelated,
  - $\text{var}(z_i)$ is as large as possible,
  - $\text{var}(z_1) > \text{var}(z_2) > \text{var}(z_3) > \ldots$



- Search $\mathbf{v}_1$ such as $\max \operatorname{var}(z_1)$:

$$\begin{aligned} \operatorname{var}(z_1) &= \operatorname{var}(\langle \mathbf{v}_1, \mathbf{x} \rangle) \\ &= \mathbf{v}_1^\top \boldsymbol{\Sigma} \mathbf{v}_1 \end{aligned}$$

- Search $\mathbf{v}_1$ such as $\max \text{var}(z_1)$:

$$\begin{aligned}\text{var}(z_1) &= \text{var}(\langle \mathbf{v}_1, \mathbf{x} \rangle) \\ &= \mathbf{v}_1^\top \boldsymbol{\Sigma} \mathbf{v}_1\end{aligned}$$

- Indetermined: if $\hat{\mathbf{v}}_1$ maximizes the variance, $\alpha\hat{\mathbf{v}}_1$ too! Add a constraint: $\langle \mathbf{v}_1, \mathbf{v}_1 \rangle = 1$

- Search $\mathbf{v}_1$ such as $\max \text{var}(z_1)$:

$$\begin{aligned} \text{var}(z_1) &= \text{var}(\langle \mathbf{v}_1, \mathbf{x} \rangle) \\ &= \mathbf{v}_1^\top \boldsymbol{\Sigma} \mathbf{v}_1 \end{aligned}$$

- Indetermined: if $\hat{\mathbf{v}}_1$ maximizes the variance, $\alpha\hat{\mathbf{v}}_1$ too! Add a constraint: $\langle \mathbf{v}_1, \mathbf{v}_1 \rangle = 1$
- Lagrangian:

$$\mathcal{L}(\mathbf{v}_1, \lambda_1) = \mathbf{v}_1^\top \boldsymbol{\Sigma} \mathbf{v}_1 + \lambda_1(1 - \mathbf{v}_1^\top \mathbf{v}_1)$$

- Search $\mathbf{v}_1$ such as $\max \operatorname{var}(z_1)$:

$$\begin{aligned}\operatorname{var}(z_1) &= \operatorname{var}(\langle \mathbf{v}_1, \mathbf{x}\rangle) \\ &= \mathbf{v}_1^\top \boldsymbol{\Sigma} \mathbf{v}_1\end{aligned}$$

- Indetermined: if $\hat{\mathbf{v}}_1$ maximizes the variance, $\alpha\hat{\mathbf{v}}_1$ too! Add a constraint: $\langle \mathbf{v}_1, \mathbf{v}_1\rangle = 1$
- Lagrangian:

$$\mathcal{L}(\mathbf{v}_1, \lambda_1) = \mathbf{v}_1^\top \boldsymbol{\Sigma} \mathbf{v}_1 + \lambda_1(1 - \mathbf{v}_1^\top \mathbf{v}_1)$$

- Compute the derivative w.r.t $\mathbf{v}_1$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{v}_1} = 2\boldsymbol{\Sigma}\mathbf{v}_1 - 2\lambda_1 \mathbf{v}_1$$

- Search $\mathbf{v}_1$ such as $\max \mathrm{var}(z_1)$:

$$\begin{aligned}\mathrm{var}(z_1) &= \mathrm{var}(\langle \mathbf{v}_1, \mathbf{x} \rangle) \\ &= \mathbf{v}_1^\top \boldsymbol{\Sigma} \mathbf{v}_1\end{aligned}$$

- Indetermined: if $\hat{\mathbf{v}}_1$ maximizes the variance, $\alpha\hat{\mathbf{v}}_1$ too! Add a constraint: $\langle \mathbf{v}_1, \mathbf{v}_1 \rangle = 1$
- Lagrangian:

$$\mathcal{L}(\mathbf{v}_1, \lambda_1) = \mathbf{v}_1^\top \boldsymbol{\Sigma} \mathbf{v}_1 + \lambda_1 (1 - \mathbf{v}_1^\top \mathbf{v}_1)$$

- Compute the derivative w.r.t $\mathbf{v}_1$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{v}_1} = 2\boldsymbol{\Sigma}\mathbf{v}_1 - 2\lambda_1 \mathbf{v}_1$$

- $\mathbf{v}_1$ is an eigenvector of the covariance matrix of $\mathbf{x}$:

$$\boldsymbol{\Sigma}\mathbf{v}_1 = \lambda_1 \mathbf{v}_1$$

- Search $\mathbf{v}_1$ such as $\max \text{var}(z_1)$:

$$\begin{aligned} \text{var}(z_1) &= \text{var}(\langle \mathbf{v}_1, \mathbf{x} \rangle) \\ &= \mathbf{v}_1^\top \boldsymbol{\Sigma} \mathbf{v}_1 \end{aligned}$$

- Indetermined: if $\hat{\mathbf{v}}_1$ maximizes the variance, $\alpha\hat{\mathbf{v}}_1$ too! Add a constraint: $\langle \mathbf{v}_1, \mathbf{v}_1 \rangle = 1$
- Lagrangian:

$$\mathcal{L}(\mathbf{v}_1, \lambda_1) = \mathbf{v}_1^\top \boldsymbol{\Sigma} \mathbf{v}_1 + \lambda_1(1 - \mathbf{v}_1^\top \mathbf{v}_1)$$

- Compute the derivative w.r.t $\mathbf{v}_1$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{v}_1} = 2\boldsymbol{\Sigma}\mathbf{v}_1 - 2\lambda_1 \mathbf{v}_1$$

- $\mathbf{v}_1$ is an eigenvector of the covariance matrix of $\mathbf{x}$:

$$\boldsymbol{\Sigma}\mathbf{v}_1 = \lambda_1 \mathbf{v}_1$$

- $\mathbf{v}_1$ is the eigenvector corresponding to the largest eigenvalues !

$$\text{var}(z_1) = \mathbf{v}_1^\top \boldsymbol{\Sigma} \mathbf{v}_1 = \lambda_1 \mathbf{v}_1^\top \mathbf{v}_1 = \lambda_1$$

- Search $\mathbf{v}_2$ such as $\max \text{var}(z_2)$ and $\langle \mathbf{v}_2, \mathbf{v}_2 \rangle = 1$ and $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle = 0$

- Search $\mathbf{v}_2$ such as $\max \text{var}(z_2)$ and $\langle \mathbf{v}_2, \mathbf{v}_2 \rangle = 1$ and $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle = 0$
- Lagrangian:

$$\mathcal{L}(\mathbf{v}_2, \lambda_2, \beta_1) = \mathbf{v}_2^\top \boldsymbol{\Sigma} \mathbf{v}_2 + \lambda_2(1 - \mathbf{v}_2^\top \mathbf{v}_2) + \beta_1(0 - \mathbf{v}_2^\top \mathbf{v}_1)$$

- Search $\mathbf{v}_2$ such as $\max \text{var}(z_2)$ and $\langle \mathbf{v}_2, \mathbf{v}_2 \rangle = 1$ and $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle = 0$
- Lagrangian:

$$\mathcal{L}(\mathbf{v}_2, \lambda_2, \beta_1) = \mathbf{v}_2^\top \boldsymbol{\Sigma} \mathbf{v}_2 + \lambda_2(1 - \mathbf{v}_2^\top \mathbf{v}_2) + \beta_1(0 - \mathbf{v}_2^\top \mathbf{v}_1)$$

- Compute the derivative w.r.t $\mathbf{v}_2$:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{v}_2} &= 2\boldsymbol{\Sigma}\mathbf{v}_2 - 2\lambda_2\mathbf{v}_2 - \beta_1\mathbf{v}_1 \\ \boldsymbol{\Sigma}\mathbf{v}_2 &= \lambda_2\mathbf{v}_2 + 2\beta_1\mathbf{v}_1 \end{aligned}$$

- Search $\mathbf{v}_2$ such as $\max \operatorname{var}(z_2)$ and $\langle \mathbf{v}_2, \mathbf{v}_2 \rangle = 1$ and $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle = 0$
- Lagrangian:

$$\mathcal{L}(\mathbf{v}_2, \lambda_2, \beta_1) = \mathbf{v}_2^\top \boldsymbol{\Sigma} \mathbf{v}_2 + \lambda_2(1 - \mathbf{v}_2^\top \mathbf{v}_2) + \beta_1(0 - \mathbf{v}_2^\top \mathbf{v}_1)$$

- Compute the derivative w.r.t $\mathbf{v}_2$:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{v}_2} &= 2\boldsymbol{\Sigma}\mathbf{v}_2 - 2\lambda_2\mathbf{v}_2 - \beta_1\mathbf{v}_1 \\ \boldsymbol{\Sigma}\mathbf{v}_2 &= \lambda_2\mathbf{v}_2 + 2\beta_1\mathbf{v}_1 \end{aligned}$$

- At optimality, $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle = 0$. Left-multiplying by $\mathbf{v}_1^\top$ the above equation:

$$\begin{aligned} \mathbf{v}_1^\top \boldsymbol{\Sigma}\mathbf{v}_2 &= 2\beta_1 \\ \lambda_1\mathbf{v}_1^\top \mathbf{v}_2 &= 2\beta_1 \\ 0 &= 2\beta_1 \end{aligned}$$

- Search $\mathbf{v}_2$ such as $\max \text{var}(z_2)$ and $\langle \mathbf{v}_2, \mathbf{v}_2 \rangle = 1$ and $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle = 0$
- Lagrangian:

$$\mathcal{L}(\mathbf{v}_2, \lambda_2, \beta_1) = \mathbf{v}_2^\top \mathbf{\Sigma} \mathbf{v}_2 + \lambda_2(1 - \mathbf{v}_2^\top \mathbf{v}_2) + \beta_1(0 - \mathbf{v}_2^\top \mathbf{v}_1)$$

- Compute the derivative w.r.t $\mathbf{v}_2$:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{v}_2} &= 2\mathbf{\Sigma}\mathbf{v}_2 - 2\lambda_2\mathbf{v}_2 - \beta_1\mathbf{v}_1 \\ \mathbf{\Sigma}\mathbf{v}_2 &= \lambda_2\mathbf{v}_2 + 2\beta_1\mathbf{v}_1 \end{aligned}$$

- At optimality, $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle = 0$. Left-multiplying by $\mathbf{v}_1^\top$ the above equation:

$$\begin{aligned} \mathbf{v}_1^\top \mathbf{\Sigma}\mathbf{v}_2 &= 2\beta_1 \\ \lambda_1\mathbf{v}_1^\top \mathbf{v}_2 &= 2\beta_1 \\ 0 &= 2\beta_1 \end{aligned}$$

- Hence, we have

$$\mathbf{\Sigma}\mathbf{v}_2 = \lambda_2\mathbf{v}_2$$

- Search $\mathbf{v}_2$ such as $\max \text{var}(z_2)$ and $\langle \mathbf{v}_2, \mathbf{v}_2 \rangle = 1$ and $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle = 0$
- Lagrangian:

$$\mathcal{L}(\mathbf{v}_2, \lambda_2, \beta_1) = \mathbf{v}_2^\top \boldsymbol{\Sigma} \mathbf{v}_2 + \lambda_2(1 - \mathbf{v}_2^\top \mathbf{v}_2) + \beta_1(0 - \mathbf{v}_2^\top \mathbf{v}_1)$$

- Compute the derivative w.r.t $\mathbf{v}_2$:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{v}_2} &= 2\boldsymbol{\Sigma}\mathbf{v}_2 - 2\lambda_2 \mathbf{v}_2 - \beta_1 \mathbf{v}_1 \\ \boldsymbol{\Sigma}\mathbf{v}_2 &= \lambda_2 \mathbf{v}_2 + 2\beta_1 \mathbf{v}_1 \end{aligned}$$

- At optimality, $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle = 0$. Left-multiplying by $\mathbf{v}_1^\top$ the above equation:

$$\begin{aligned} \mathbf{v}_1^\top \boldsymbol{\Sigma}\mathbf{v}_2 &= 2\beta_1 \\ \lambda_1 \mathbf{v}_1^\top \mathbf{v}_2 &= 2\beta_1 \\ 0 &= 2\beta_1 \end{aligned}$$

- Hence, we have

$$\boldsymbol{\Sigma}\mathbf{v}_2 = \lambda_2 \mathbf{v}_2$$

- $\mathbf{v}_2$ is the eigenvector corresponding the *second largest* eigenvalues

- Search $\mathbf{v}_2$ such as $\max \text{var}(z_2)$ and $\langle \mathbf{v}_2, \mathbf{v}_2 \rangle = 1$ and $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle = 0$
- Lagrangian:

$$\mathcal{L}(\mathbf{v}_2, \lambda_2, \beta_1) = \mathbf{v}_2^\top \mathbf{\Sigma} \mathbf{v}_2 + \lambda_2(1 - \mathbf{v}_2^\top \mathbf{v}_2) + \beta_1(0 - \mathbf{v}_2^\top \mathbf{v}_1)$$

- Compute the derivative w.r.t $\mathbf{v}_2$:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{v}_2} &= 2\mathbf{\Sigma}\mathbf{v}_2 - 2\lambda_2\mathbf{v}_2 - \beta_1\mathbf{v}_1 \\ \mathbf{\Sigma}\mathbf{v}_2 &= \lambda_2\mathbf{v}_2 + 2\beta_1\mathbf{v}_1 \end{aligned}$$

- At optimality, $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle = 0$. Left-multiplying by $\mathbf{v}_1^\top$ the above equation:

$$\begin{aligned} \mathbf{v}_1^\top \mathbf{\Sigma} \mathbf{v}_2 &= 2\beta_1 \\ \lambda_1 \mathbf{v}_1^\top \mathbf{v}_2 &= 2\beta_1 \\ 0 &= 2\beta_1 \end{aligned}$$

- Hence, we have

$$\mathbf{\Sigma}\mathbf{v}_2 = \lambda_2\mathbf{v}_2$$

- $\mathbf{v}_2$ is the eigenvector corresponding the *second largest* eigenvalues
- $\mathbf{v}_k$ is the eigenvector corresponding the k^{th} *largest* eigenvalues

1. Empirical estimation the mean value:

$$\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{x}_i$$

2. Empirical estimation the covariance matrix:

$$\boldsymbol{\Sigma} = \frac{1}{n-1}\sum_{i=1}^{n}(\mathrm{x}_i - \boldsymbol{\mu})(\mathrm{x}_i - \boldsymbol{\mu})^\top$$

3. Compute p first eigenvalues/eigenvectors... How to choose p ? Explained variance:

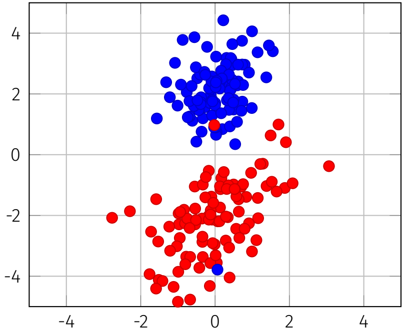
$$\frac{\sum_{i=1}^{d}\lambda_i}{\sum_{i=1}^{p}\lambda_i}$$

Note: Standardization/scaling matters!

Feature Extraction/Selection

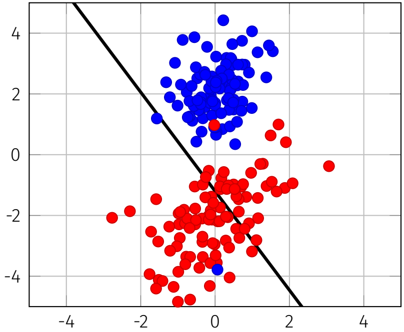
Supervised feature extraction

- We observe some $\{\mathbf{x}_i, y_i\}_{i=1}^n$
- Use the label information to find the linear features that highlight differences among classes



- FDA: find $\mathbf{a}$ such as the ratio between the *between projected variance* and the *sample projected variance* is maximal

- We observe some $\{\mathbf{x}_i, y_i\}_{i=1}^n$
- Use the label information to find the linear features that highlight differences among classes



- FDA: find $\mathbf{a}$ such as the ratio between the *between projected variance* and the *sample projected variance* is maximal

- Between-class covariance matrix:

$$\mathbf{B} = \frac{1}{n}\sum_{c=1}^{C} n_c(\boldsymbol{\mu}_c - \boldsymbol{\mu})(\boldsymbol{\mu}_c - \boldsymbol{\mu})^\top$$

- Class covariance matrix

$$\boldsymbol{\Sigma}_c = \frac{1}{n_c - 1}\sum_{i=1, i\in c}^{n_c} (\mathbf{x}_i - \boldsymbol{\mu}_c)(\mathbf{x}_i - \boldsymbol{\mu}_c)^\top$$

- Within-class covariance matrix

$$\mathbf{W} = \sum_{c=1}^{C} \boldsymbol{\Sigma}_c$$

- The Fisher discriminant subspace is given by the eigenvectors of $\mathbf{W}^{(-1)}\mathbf{B}$
- Remark: there are at most $C-1$ eigenvectors because $\text{Rank}(\mathbf{B}) \leq C-1$. They should be selected similarly to PCA.
- There is an equivalence between FDA and LDA
- Notebook

Model Selection and Model Assessment

Model Selection and Model Assessment

Introduction

- Loss-function $L(y,\hat{y}) = 0$ if $y = \hat{y}$ else 1
- **Train error**: average loss over the training sample

$$\mathrm{Err}_{\mathrm{train}} = \frac{1}{n}\sum_{i=1}^{n} L(y_i, \hat{y}_i)$$

- **Prediction error**: average loss over an independent test sample → Generalization error
- General picture:

$$\mathrm{Err}_{\mathrm{test}} \approx \mathrm{Err}_{\mathrm{train}} + O$$

O would be the average *optimism*.

Model selection

- Estimate the best set of hyperparameters
- Estimate the performance of differents models

Model Assessment

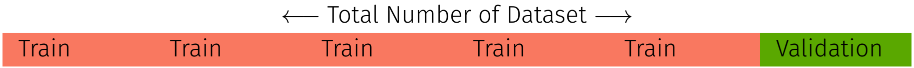
Estimate the generalization error on unseen/test sample

Model selection

- Estimate the best set of hyperparameters
- Estimate the performance of differents models

Model Assessment

Estimate the generalization error on unseen/test sample



Model Selection and Model Assessment

Cross-validation

- Method to estimate prediction error using the training sample
- Based on splitting the data in K-folds :

Model 1	Train	Train	Train	Train	Validation
Model 2	Train	Train	Train	Validation	Train
Model 3	Train	Train	Validation	Train	Train
Model 4	Train	Validation	Train	Train	Train
Model 5	Validation	Train	Train	Train	Train

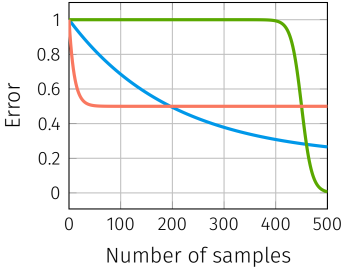
- Expected prediction error:

$$CV(\hat{f}, \theta) = \sum_{k=1}^{K} \mathrm{Err}_k(\hat{f}, \theta)$$

- K ? Usually K=5 or 10 is a good trade-off (K=n is called leave-one-out)

	Bias	Variance
K low	High	Low
K high	Low	High
K = n	Low	Very High

- Be careful to the learning curve



- Model should be trained completely for each fold (i.e., data normalization, optimization, etc ...)
- Notebook

Conclusions

Notebook

- There is no universal best classifier
- Needs to be chosen appropriately
- Pay attention to
  - Scale your data,
  - Try several algorithms, and optimize their hyperparameters
  - Extract/Select/Build relevant features
- In many situations, simple is actually good!
- Sklearn is a good try !

`https://scikit-learn.org/stable/index.html`

Thank you for your attention