Learning and Estimation of Dynamical Systems

Dante Piotto

Spring semester 2023

Contents

1 Introduction **5**
  1.1 What is a system? . . . 5
    1.1.1 Learning modesl from data . . . 6
  1.2 Types of learning . . . 6
  1.3 fields related to learning from data . . . 6
    1.3.1 System identification . . . 6
  1.4 Learning steps . . . 7

2 Stochastic Processes **9**
  2.1 stationary stochastic processes . . . 9
    2.1.1 cross-correlation and cross-covariance . . . 9
  2.2 vector stochastic processes . . . 9
  2.3 gaussian processes . . . 9
  2.4 white processes . . . 9

3 Stochastic models **11**
  3.1 Matlab stuff . . . 11

4 Estmation problem **13**

5 Linear regression **15**
  5.1 The Least Squares Method . . . 15
    5.1.1 Geometrical interpretation of the LS estimate . . . 16
    5.1.2 Statistical properties of the LS estimator . . . 16
    5.1.3 LS estimation fo ARX models . . . 17
    5.1.4 ARX optimal (one step ahead) predictor . . . 18
    5.1.5 LS estimation of AR models . . . 18
  5.2 Recursive least squares . . . 18
    5.2.1 Asymptotic behaviour of the RLS algorithm . . . 21
    5.2.2 Recursive wighted least squares . . . 21

6 Prediction error methods **23**
    6.0.1 The Newton-Raphson Algorithm . . . 25
    6.0.2 The Gauss-Newton Method . . . 25
    6.0.3 Identification of ARMAX models . . . 26
    6.0.4 Identification of ARARX models . . . 26
    6.0.5 Statistical properties of PEM estimators . . . 26
    6.0.6 MISO ARX models . . . 27

7 Statistical hypothesis testing **29**

8 Model complexity selection 31
8.1 The F-test . . . 31
8.2 The final prediction error (FPE) criterion . . . 32
8.3 Criteria with complexity terms . . . 33
    8.3.1 Akaike information criterien (AIC) . . . 33

9 Model assesment (validation) 35
9.1 Whiteness test . . . 35
9.2 Test of cross-correlation . . . 36

10 Maximum likelihood estimation 37
10.1 the gaussian case . . . 37
    10.1.1 the Cramér-Rao lower bound . . . 38

11 Classification: probabilistic models 39
11.1 The Bayes classifier . . . 40
11.2 Logistic regression . . . 40
    11.2.1 The gradient descent algorithm . . . 42
    11.2.2 Multiclass problems . . . 43
    11.2.3 Dealing with nonlinear boundaries . . . 43
11.3 Linear discriminant analysis . . . 44
    11.3.1 Gaussian class densities with commmon covariance matrix . . . 44
    11.3.2 Gaussian class denisities with different covariance matrices . . . 45

12 Classification: deterministic models 47
    12.0.1 Separating hyperplanes . . . 48
    12.0.2 the maximum margin classifier . . . 48
12.1 Constrained optimization problems: the Lagrange multipliers . . . 48
12.2 Support vector machine . . . 51
    12.2.1 Dealing with nonlinear boundaries: the kernel trick . . . 53
    12.2.2 Multiclass problems . . . 54

13 Regularization 55
13.1 Regularized least squares: the Ridge regression . . . 55

14 Optimal estimation of random signals 59
14.1 the Kalman filter . . . 60
    14.1.1 Deterministic state space models: the Luenberger observer . . . 60
    14.1.2 Stochastic state space models . . . 61
    14.1.3 Derivation of the Kalman filter . . . 61
    14.1.4 Kalman filter equations: predictor-corrector form . . . 62
    14.1.5 Kalman predictor form . . . 63
    14.1.6 Convergence of the difference Riccati equation . . . 63
    14.1.7 Time invariant filter (predictor) . . . 63

Chapter 1

Introduction

Goal of the course: build matemathical models of dynamical systems from data
Password of slides: LEDS$2023
Office appointments to be decided with professor as needed

1.1 What is a system?

A slice of reality whose evoultion in time can be described by a certain number of measurable attributes (variables)
Inputs: independent variables(causes) which describe the action the surrounding environment on the system
Outputs: dependent variables (effects) which describe the reactoin of the syste
Mathematica model: a set of quantitative relationships between the system variables
Solving probelms in scientific disciplicnes by means of mathematical models:

1. determination of a mathematical model of the system
2. solution of the problem by using the model (i.e. in the mathematical world)
3. implementation fo the obtained solution on the real process

Competent model: a good model for solving a given problem in a certain problem context

- different mathematical models can be associated with the same system
- classification of models based on the modeling objectives

modeling objectives:

- inference
- control
- prediction
- filtering
- diagnosis
- predictive maintenance
- simulation
- speech and image recognition

1.1.1 Learning modesl from data

Data (set of samples)

$$u(1), u(2), \ldots, u(N) \qquad y(1), y(2), \ldots, y(N) \qquad\qquad u(t) \in \mathcal{U}, y(t) \in \mathcal{Y}$$

Target model(function)

$$\mathcal{M}_p(\theta)$$

$\mathcal{M}_p(\cdot)$ represents a function linking input and output samples, θ is a set of parameters and p is a set of hyperparameters
Static models: $f : \mathcal{U} \to \mathcal{Y}$
Dynamic models: $f : (\mathcal{U}, \mathcal{Y}) \to \mathcal{Y}$

1.2 Types of learning

Supervised learning (u, y known):

- $\mathcal{Y}$ is discrete: classification
- $\mathcal{Y}$ is continuous: regression

Unsupervised learning (u, y uknown):

- $\mathcal{Y}$ is discrete: clustering
- $\mathcal{Y}$ is continuous: dimensionality reduction

Classification: assign the input to one of a finite number of classes
Regression: find an input-output relation
Reinforcement learning: finding suitable actions to take in a given situation in order to maximize a reward. Both u, y are known but we haev to find the "optimal" output for a given input

This course deals with Supervised learning

1.3 fields related to learning from data

- Machine learning
- Pattern recognition
- Statistical learning
- Data mining
- System identification

1.3.1 System identification

System identification is the art and science of building mathematical models of dynamic systems from observed input-output data
Learning from data is a *data-driven(black box) approach*: a model is selected within a specified model class by using a selection criterion on the only basis of experimental (observed) data. No reference tot he physical nature of the system is made

- the obtained models have limited validity
- the model parameters may lack any physical meaning
- models relatively easy to construct and use
- ability to extract only some relevant aspects from complex frameworks

In contrast, *physical modeling* is a white box approach. The systems is partitiond into subsystems that aredescribed by using known laws of physics. Then, the model of the system is obtained by joining such relations.

Grey box approach

It often happens that a model based on physical modeling contains a number of uknown parameters: identification(learning) methods can be applied to estimate the uknown parameters.

1.4 Learning steps

The planned use of the model is important in designing the experiment to collect data (when possible).

Chapter 2

Stochastic Processes

Let us consider a random experiment, with sample space Ω, and let us associate to each event ω_i in the sample space a signal $x(t, \omega_i)$. With a fixed ω we have a function of time $x(t)$, and at each fixed t_i, $x(\omega)$ is a random variable

Definition (discrete time stochastic process):
A function $x(t, \omega)$ where $t \in \{\dots, -2, -1 - 0, 1, 2, 1 \dots\}$ is time and $\omega \in \Omega$ is an outcome of the sample space. $x(t, \omega_i)$ is called a *realization* of the stochastic process Given $t = t_1$ the first order cdf and pdf are:

$$F(x; t_1) = P(x(t_1) \leq x) \qquad f(x; t_1) = \frac{\delta F(x; t_1)}{\delta x}$$

Autocovariance: relation proven using linearity of the expectation operaror

2.1 stationary stochastic processes

A stochastic process is stationary if

$$F(x_1, x_2, \dots, x_k; t_1, t_2, \dots, t_k) = F(x_1, x_2, \dots, x_k; t_1 + \tau, t_2 + \tau, \dots, t_k + \tau) \tag{2.1}$$
$$\forall \tau, \forall k, \forall \{t_1, t_2, \dots, t_k\} \tag{2.2}$$

this property also holds for the pdf Consequences:

$$\mu_x(t) = \mu_x \tag{2.3}$$
$$\sigma_x^2(t) = \sigma_x^2 \tag{2.4}$$
$$r_x(t_1, t_2) = r_x(t_1 - t_2) = r_x(\tau) \tag{2.5}$$
$$c_x(t_1, t_2) = c_x(t_1 - t_2) = c_x(\tau) \tag{2.6}$$

A process is weakly stationary (or wide-sense stationary) if the 4 properties above hold
Toeplitz matrix: symmetric matrix with all elements belonging to a diagonal being equal.
Cross-correlation and cross-covariance can only be defined for stationary stochastic proceesses

2.1.1 cross-correlation and cross-covariance

2.2 vector stochastic processes

2.3 gaussian processes

2.4 white processes

it is not possible to define a continuous time white process

Chapter 3

Stochastic models

PSD= Power Spectral Density

Moving Average noise is also called pink noise Matlab considers the normalized autocorrelation.

3.1 Matlab stuff

rand(): generates white noise vector
cov(X): gives the variance of vector X
randn(): generates gaussian distributed white noise vector
autocorr(X): gives the normalized autocorrelation
to get the non normalized autocorrelation: autocorr(X)*cov(X)
filter(): useful to generate AR, MA, and ARMA processes

Chapter 4

Estmation problem

w.p.1: with probability 1 A biased estimator with small variance and a sufficiently small bias may be preferrable to an unbiased estimator with high variance. *bias-variance tradeoff*

Chapter 5

Linear regression

Let us consider the static model

$$y(t) = f(u(t)) + e(t) \quad \text{model class } \mathcal{M}_p(\theta)$$

If the function f is linear in the parameters (elements of θ), the modell can be written in the *linear regression* form:

$$y(t) = \varphi^T(t)\theta + e(t)$$

5.1 The Least Squares Method

available data set:

$$y(1), y(2), \dots, y(N), u(1), u(2), \dots, u(N)$$

If an estimate $\hat{\theta}$ were available, we would compute the misfit between $y(t)$ and its 'prediction' $\hat{y}(t)$:

$$\varepsilon(t) = y(t) - \hat{y}(t) = y(t) - \varphi^T(t)\hat{\theta}$$

where the error term $\epsilon(t)$ is the residual. The LS method finds the estimate $\hat{\theta}$ that minimizes the loss function

$$J(\theta) = \sum_{t=1}^{N} \varepsilon^2(t) = \sum_{t=1}^{N} (y(t) - \varphi^T(t)\hat{\theta})^2 \tag{5.1}$$

In matrix form:

$$\varepsilon = Y - \Phi\theta$$

where

$$\varepsilon = \begin{bmatrix} \varepsilon(1) \\ \varepsilon(2) \\ \vdots \\ \varepsilon(N) \end{bmatrix} \quad y = \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(N) \end{bmatrix} \quad \varphi = \begin{bmatrix} \varphi(1) \\ \varphi(2) \\ \vdots \\ \varphi(epsilonN) \end{bmatrix}$$

so that

$$J(\theta) = \sum_{t=1}^{N} \varepsilon^2(t) = \|\varepsilon\|^2 = \|Y - \Phi\theta\|^2 \tag{5.2}$$

The optimization problem to be solved is

$$\min_{\theta \in \mathcal{M}_p(\theta)} J(\theta)$$

The solution can be found by using the following relations:

$$\frac{\partial A^T x}{\partial x} = A, \quad \frac{\partial x^T A}{\partial x} = A, \quad \frac{\partial x^T A x}{\partial x} = (A + A^T)x$$

where A is an $n \times n$ matrix and x is a $n \times 1$ vector
From the above relations we obtain the *normal equations*

$$\Phi^T \Phi \theta = \Phi^T Y$$

proof was covered in class

$$\begin{aligned}
J(\theta) &= Y^T Y - 2Y^T \Phi \theta + \theta^T \Phi^T \Phi \theta \\
\frac{\partial J(\theta)}{\partial \theta} &= 0 - 2\Phi^T Y + (\Phi^T \Phi + \Phi^T \Phi)\theta \\
\frac{\partial J(\theta)}{\partial \theta} &= -2\Phi^T Y + 2\Phi^T \Phi \theta \\
\frac{\partial J(\theta)}{\partial \theta} &= 0 \implies \Phi^T \Phi \theta = \Phi^T Y
\end{aligned}$$

To complete the proof we must prove that the second derivative of the matrix is positive definite

$$\frac{\partial^2 J(\theta)}{\partial \theta^2} = 2\Phi^T \Phi$$

If the $N \times p$ matrix Φ is tall ($N > p$) and full rank, the LS estimate is given by

$$\hat{\theta} = (\Phi^T \Phi)^{-1} \Phi^T Y \tag{5.3}$$

5.1.1 Geometrical interpretation of the LS estimate

The estimated function is such that the sum of the squares of the distances, evaluated along the y-axis, between $y(t)$ and its 'prediction' $\hat{y}(t)$ is minimized.
Consider the linear map described by Φ:

$$\Phi : \mathbb{R}^p \to \mathbb{R}^N$$

Let $\Phi_1, \Phi_2, \ldots, \Phi_p$ be the columns of matrix Φ. The LS problem consists in finding the linear combination of $\Phi_1, \Phi_2, \ldots, \Phi_p$ that approximates Y as closely as possible. Therefore, the solution is given by the orthogonal projection of Y onto the subspace spanned by $\Phi_1, \Phi_2, \ldots, \Phi_p$, i.e. the orthogonal projection of Y onto the image of A
$\hat{\theta}$ is such that $\varepsilon = Y - \Phi\hat{\theta}$ is ortogonal to $\Phi_1, \Phi_2, \ldots, \Phi_p$

$$\implies \quad \varepsilon^T \Phi = 0$$

$$\begin{aligned}
\varepsilon &= Y - \Phi\hat{\theta} = Y - \hat{Y} \\
\varepsilon^T \Phi &= (Y - \Phi\hat{\theta} = Y - \hat{Y})^T \Phi = Y^T \Phi - \hat{\theta} \Phi^T \Phi \\
&= Y^T \Phi - Y^T \Phi (\Phi^T \Phi)^{-1} \Phi^T \Phi = 0
\end{aligned}$$

The LS solution $\hat{\theta}$ can be obtained by considering the pseudoinverse of Φ:

$$\hat{\theta} = \Phi^\dagger Y$$

If Φ is full rank, its pseudoinverse is just given by $\Phi^\dagger = (\Phi^T \Phi)^{-1} \Phi^T$

5.1.2 Statistical properties of the LS estimator

Assume the true model

$$y(t) = \varphi^T(t)\theta^* + w(t)$$

where $w(t)$ is a zero mean white process with variance σ_w^2. Let $\hat{\theta}_N$ be an estimate obtained by using N input-output samples
It follows that

$$\begin{aligned}
E[\hat{\theta}] &= E[(\Phi^T\Phi)^{-1}\Phi^T Y] \\
Y &= \Phi\theta^* + w \\
E[\hat{\theta}]E &= [(\Phi^T\Phi)^{-1}\Phi^T Y = \Phi\theta^* + w] = E[\theta^* + (\Phi^T\Phi)^{-1}\Phi^T w] \\
&= \theta^*(\Phi^T\Phi)^{-1}\Phi^T E[w] = \theta^*
\end{aligned}$$

therefore the LS estimator is unbiased.

5.1.3 LS estimation fo ARX models

The closed form solution of the LS problem is obtained in a similar fashion to the FIR model case.

Identifiablity

- $u(t)$ has to be persistently exciting of order $\geq n$
- In order to have a well-condition matrix H (i.e.) with a low condition number), it is also necessary that n is not greater than the minimal order of an ARX model compatible with the data

Statistical properties

True model:

$$y(t) = \varphi^T(t)\theta^* + w(t)$$

It follows that

$$\begin{aligned}
\hat{\theta} &= \left(\frac{1}{N}\sum_{t=1}^{N}\varphi(t)\varphi^T(t)\right)^{-1}\frac{1}{N}\sum_{t=1}^{N}\varphi(t)\left(\varphi^T(t)\theta^* + w(t)\right) \\
\hat{\theta} &= \theta^* + \left(\frac{1}{N}\sum_{t=1}^{N}\varphi(t)\varphi^T(t)\right)^{-1}\frac{1}{N}\sum_{t=1}^{N}\varphi(t)w(t)
\end{aligned}$$

The estimate is biased: $E[\hat{\theta}] \neq \theta^*$ Consisstency:

$$\begin{aligned}
\lim_{N\to\infty}\hat{\theta}_N &= \theta^* + \lim_{N\to\infty}\left(\frac{1}{N}\sum\varphi(t)\varphi^T(t)\right)^{-1}\frac{1}{N}\sum\varphi(t)w(t) \\
\lim_{N\to\infty}\hat{\theta}_N &= \theta^* + E[\varphi(t)\varphi^T(t)]^{-1}E[\varphi(t)w(t)] = \theta^* - \Sigma_\varphi^{-1}r_{\varphi w}
\end{aligned}$$

where $\Sigma_\varphi^{-1} = E(\varphi(t)\varphi^T(t)$ and $r_{\varphi w}$ is the cross correlation. Because $u(t)$ is pe of order n, and $w(t)$ is white, Σ_φ is invertible, and $r_{\varphi w} = 0$ because $y(t-n)$ does not depend on $w(t)$ for $n > 0$ and $u(t)$ in general does not depend on w, therefore

$$\lim_{N\to\infty}\hat{\theta} = \theta^*$$

It is also possible to prove that

$$\sqrt{N}(\hat{\theta}\theta^* \to \mathcal{N}(0, P), \text{ for } N \to \infty$$

with

$$P = \sigma_w^2\Sigma_\varphi^{-1}$$

A consistent estimate of σ_w^2 is given by

$$\sigma_w^2 = J(\hat{\theta})$$

Then, $cov(\hat{\theta})$ can be estimated as

$$\frac{\hat{\sigma}_w^2\hat{\Sigma}_\varphi^{-1}}{N} = \hat{\sigma}_w^2(H^T H)^{-1}$$

The equivalence can be derived considering

$$\hat{\Sigma}_\varphi = \frac{1}{N}\sum\varphi(t)\varphi^T(t) = \frac{H^T H}{N}$$

5.1.4 ARX optimal (one step ahead) predictor

True model:

$$y(t) = \varphi^T(t)\theta^* + w(t)$$

Problem: find the optimal (minimal variance) prediction of $y(t)$ given the past data $y(t-1), u(t-1), y(t-2), u(t-2), \dots$ It is easy to show that the optimal predictor $\hat{y}(t|t-1)$ is given by

$$y(t|t-1) = \varphi^T(t)\theta^*$$

we know that

$$y(t) = -a_1^* y(t-1) - \dots - a_n^* y(t-n) + b_1^* u(t-1) + \dots + b_n^* u(t-n) + w(t)$$

because $w(t)$ is a white process, the best estimate we can make for it is its mean, 0.

For any other predictor $\hat{\hat{y}}(t)$

$$E[(y(t) - \hat{\hat{y}}(t))^2] \geq E[(y(t) - \hat{y}(t|t-1))^2] = \sigma_w^2$$

As a consequence, the LS estimation $\hat{\theta}$ leads to a predictive model and the residual $\varepsilon(t)$ can be seen as a prediction error. In fact, from

$$\lim_{N\to\infty} \hat{\theta} = \theta^*$$

it follows that

$$\varepsilon(t) = y(t) - \varphi^T(t)\hat{\theta} \xrightarrow[N\to\infty]{} \theta^*$$

5.1.5 LS estimation of AR models

$$y(t) + a_1 y(t-1) + \dots + a_n y(t-n) = e(t)$$

Linear regression form:

$$y(t) = \varphi^T(t)\theta + e(t)$$

where:

$$\varphi(t) = \begin{bmatrix} -y(t-1) & -y(t-2) & \cdots & -y(t-n) \end{bmatrix}^T$$
$$\theta = \begin{bmatrix} a_1 & \cdots & a_n \end{bmatrix}^T$$
$$Y = -H_y(n)\theta + \varepsilon$$

LS estimate:

$$\hat{\theta} = -(H_y^T(n)H_y(n))^{-1}H_y^T(n)Y$$

Statistical properties and optimale predictor: similar considerations to those of the ARX case.

5.2 Recursive least squares

Let $\hat{\theta}(t-1)$ be a LS estimate obtained form data collected up to time $t-1$:

$$\hat{\theta}(t-1) = \left(\sum_{k=1}^{t-1} \varphi(k)\varphi^T(k)\right)^{-1} \sum_{k=1}^{t-1} \varphi(k)y(k) \qquad (1)$$

Recursive identification methods consist of updating $\hat{\theta}(t-1)$ by some "simple modification" once data at time t becomes available to compute $\hat{\theta}(t)$

Starting from (1) and defining:

$$\sum_{k=1}^{t} \varphi(k)\varphi^T(k) = S(t) = S(t-1) + \varphi(t)\varphi^T(t)$$

we can observe that:

$$\begin{aligned}
&\sum_{k=1}^{t}\varphi(k)y(k) = \sum_{k=1}^{t-1}\varphi(k)y(k) + \varphi(t)y(t)\\
&\hat{\theta}(t) = S(t)^{-1}\sum_{k=1}^{t}\varphi(k)y(k)\\
&\hat{\theta}(t-1) = S(t-1)^{-1}\sum_{k=1}^{t-1}\varphi(k)y(k)
\end{aligned}$$

from which

$$\sum_{k=1}^{t-1}\varphi(k)y(k) = S(t-1)\hat{\theta}(t-1) \tag{5.4}$$

$$\tag{5.5}$$

by simply substituting we obtain

$$\sum_{k=1}^{t}\varphi(k)y(k) = S(t-1)\hat{\theta}(t-1) + \varphi(t)y(t)$$

and therefore

$$\begin{aligned}
\hat{\theta}(t) &= S(t)^{-1}(S(t-1)\hat{\theta}(t-1) + \varphi(t)y(t))\\
&= S(t)^{-1}\left[\left(S(t) - \varphi(t)\varphi(t)^T\right)\hat{\theta}(t-1) + \varphi(t)y(t)\right]\\
&= \hat{\theta}(t-1) + S(t)^{-1}\varphi(t)\left[y(t) - \varphi(t)^T\hat{\theta}(t-1)\right]\\
&= \hat{\theta}(t-1) + K(t)\varepsilon(t)
\end{aligned}$$

where

$$K(t) = S(t)^{-1}\varphi(t)$$

and we can recall that

$$\varepsilon(t) = y(t) - \varphi^T(t)\hat{\theta}(t-1)$$

is the prediction error. This leads to the following recursive least squares algorithm:

RLS I

1. $S(t) = S(t-1) + \varphi(t)\varphi^T(t)$
2. $K(t) = S(t)^{-1}\varphi(t)$
3. $\varepsilon(t) = y(t) - \varphi^T(t)\hat{\theta}(t-1)$
4. $\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)\varepsilon(t)$

By defining $R(t) = \frac{S(t)}{t}$ it is possible to derive a RLS algorithm for

$$\hat{\theta}(t) = \left(\frac{1}{t}\sum_{k=1}^{t}\varphi(k)\varphi^T(k)\right)^{-1}\frac{1}{t}\sum_{k=1}^{t}\varphi(k)y(k)$$

in fact, $R(t)$ can be easily updated

$$R(t) = \frac{S(t)}{t} = \frac{S(t-1) + \varphi(t)\varphi^T(t)}{t} = \frac{t-1}{t}R(t-1) + \frac{\varphi(t)\varphi^T(t)}{t}$$

Following the same steps used to derive RLS I we get (computation was developed in class):

RLS II

1. $R(T) = \frac{t-1}{t}R(t-1) + \frac{1}{t}\varphi(t)\varphi^T(t)$
2. $K(t) = \frac{1}{t}R(t)^{-1}\varphi(t)$
3. $\varepsilon(t) = y(t) - \varphi^T(t)\hat{\theta}(t-1)$
4. $\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)\varepsilon(t)$

In order to not compute a matrix inversion at each step and instead updating the inverse itself, we can rely on the following result

Matrix inversion lemma (Woodbury identity)

Let A, C be square and invertible. Then

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}$$

$$\begin{gathered}
S(t)^{-1} = (S(t-1) + \varphi(t)\varphi^T(t) =^{-1} \\
A = S(t-1) \\
B = \varphi(t) \\
C = 1 \\
D = \varphi^T(t) \\
S(t)^{-1} = S(t-1)^{-1} - S(t-1)^{-1}\varphi(T)(1 + \varphi^T(t)S(t-1)^{-1}\varphi(t))^{-1}\varphi^T(t)S(t-1)^{-1}
\end{gathered}$$

note that $(1 + \varphi^T(t)S(t-1)^{-1}\varphi(t))$ is a scalar, so we can write:

$$S(t)^{-1} = S(t-1)^{-1} - \frac{S(t-1)^{-1}\varphi(T)\varphi^T(t)S(t-1)^{-1}}{1 + \varphi^T(t)S(t-1)^{-1}\varphi(t)}$$

We can now modify RLS I and RLS II in order to avoid matrix inversion

RLS III

1. $S(T) = S(t-1)^{-1} - \frac{S(t-1)^{-1}\varphi(T)\varphi^T(t)S(t-1)^{-1}}{1+\varphi^T(t)S(t-1)^{-1}\varphi(t)}$
2. $K(t) = S(t)^{-1}\varphi(t)$
3. $\varepsilon(t) = y(t) - \varphi^T(t)\hat{\theta}(t-1)$
4. $\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)\varepsilon(t)$

RLS IV

1. $P(T) = \frac{t}{t-1}P(t-1) - \frac{t}{t-1}\frac{P(t-1)^{-1}\varphi(T)\varphi^T(t)P(t-1)^{-1}}{t-1+\varphi^T(t)P(t-1)\varphi(t)}$
2. $K(t) = P(t)\frac{1}{t}phi(t)$
3. $\varepsilon(t) = y(t) - \varphi^T(t)\hat{\theta}(t-1)$
4. $\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)\varepsilon(t)$

with $P(t) = R(t)^{-1}$

Initialization

The RLS algorithm needs to be initialized. One possibility is to start with an initial batch estimate, which will be better the more data is available. It is also possible to start with some other type of guess on the model parameters, potentially also very bad, e.g. a vector of zeros. However, a guess for the gain matrix (either $S(0), R(0), P(0)$ depending on the specific algorithm) is necessary. With the RLS IV, a good initialization would be

$$P(0) = \alpha I_p$$

If we are confident about the initial guess of the parameters, a small value of α is appropriate. On the flipside, if we expect the inital estimator to be bad, the value of α shall be quite large.

5.2.1 Asymptotic behaviour of the RLS algorithm

$$y(t) = \varphi^T(t)\theta^* + w(t)$$
$$\lim_{t\to\infty} \hat{\theta}(t) = \theta^* \quad \text{w.p. } 1$$

$R(t)$ in the RLS II algo is an estimate of the covariance matrix. Looking at the RLS II we can observe that from step 2, $K(t)$ approaches 0. Therefore, asymptotically, there is no correction.

$$lim_{t1\to\infty} K(t) = \frac{\Sigma_\varphi^{-1}}{\infty}\varphi(t) = 0$$

5.2.2 Recursive wighted least squares

A modification of the RLS algorithm aimed at tracking parameter variations by giving less importance to past data and more importance to recenet data.

$$J(\theta) = \sum_{t=1}^{N} \lambda^{N-t}\varepsilon^2(t) = \varepsilon^T W \varepsilon$$

where

$$W = \text{diag}\begin{bmatrix}\lambda^{N-1} & \lambda^{N-2} & \cdots & \lambda & 1\end{bmatrix}$$

and $\lambda \in (0,1) \subset \mathbb{R}$ is the *forgetting factor*. We have

$$\hat{\theta} = \left(\sum_{t=1}^{N} \lambda^{N-t}\varphi(t)\varphi^T(t)\right)^{-1} \sum_{t=1}^{N} \lambda^{N-t}\varphi(t)y(t)$$

The scalar λ should be chosen by a trade-off between the ability to track parameter changes on one hand, and good estimation accuracy on the other hand.

Recursive weighted LS: determine a recursive form of

$$\hat{\theta}(t) = \left(\sum_{k=1}^{t} \lambda^{t-k}\varphi(k)\varphi^T(k)\right)^{-1} \sum_{k=1}^{t} \lambda^{t-k}\varphi(k)y(k)$$

Define

$$S(t) = \sum_{k=1}^{t} \lambda^{t-k}\varphi(k)\varphi^T(k) = \lambda S(t-1) + \varphi(t)\varphi^T(t)$$

Following the same reasoning useed to derive the previous recursive algorighms we can derive the RWLS algorithms

RWLS I

1. $S(t) = \lambda S(t-1) + \varphi(t)\varphi^T(t)$
2. $K(t) = S(t)^{-1}\varphi(t)$
3. $\varepsilon(t) = y(t) - \varphi^T(t)\hat{\theta}(t-1)$
4. $\hat{\theta}(t-1) + K(t)\varepsilon(t)$

Chapter 6

Prediction error methods

Let us consider the ARMAX model

$$A(z^{-1})y(t) = B(z^{-1})u(t) + C(z^{-1})w(t)$$

or

$$y(t) + a_1 y(t-1) + \cdots + a_n y(t-n) = b_1 u(t-1) + \cdots + b_n u(t-n) + w(t) + c_1 w(t-1) + \cdots + c_n w(t-n)$$

Then

$$y(t) = \varphi^T(t)\theta + w(t)$$

where

$$\varphi(t) = \begin{bmatrix} -y(t-1) & \cdots & -y(t-n) & u(t-1) & \dots & u(t-n) & w(t-1) & \cdots & w(t-n) \end{bmatrix}^T$$

In a real data setting we may assume the model

$$A(z^{-1})y(t) = B(z^{-1})u(t) + C(z^{-1})\varepsilon(t)$$

where $\varepsilon(t)$ is the residual. Problem: the residual can be computed once the estimate $\hat{\theta}$ is available, so that it is a function of $\theta \implies \varepsilon(t,\theta)$

$$y(t) = \varphi^T(t,\theta)\theta + \varepsilon(t)$$

This is no longer a linear regression and the least squares method cannot be applied. If we consider

$$\frac{C(z^{-1})}{A(z^{-1})}$$

as a disturbance $d(t)$ and only try to estimate the plant, we obtain

$$y(t) = \bar{\varphi}^T(t)\bar{\theta} + e(t)$$

and we can use the least squares method. However, $e(t)$ is now a coloured process and this estimator is not unbiased. Let us assume a true model exists θ^*.

$$y(t) = \bar{\varphi}^T(t)\theta^+ + e(t)$$

$$\hat{\hat{\theta}}_{LS} = \left(\frac{1}{N}\sum_{t=1}^{N}\bar{\varphi}(t)\bar{\varphi}^T(t)\right)\frac{1}{N}\sum_{t=1}^{N}\bar{\varphi}(t)y(t)$$

$$= \theta^+ + \left(\frac{1}{N}\sum_{t=1}^{N}\bar{\varphi}(t)\bar{\varphi}^T(t)\right)^{-1}\frac{1}{N}\sum_{t=1}^{N}\bar{\varphi}(t)e(t)$$

$$\lim_{N\to\infty}\hat{\hat{\theta}}_{LS} = \theta^* + \Sigma_{\hat{\varphi}}^{-1}r_{\bar{\varphi}e}$$

$$r_{\bar{\varphi}e} = E[\bar{\varphi}(t)e(t)] = E[\begin{bmatrix}-y(t-1)\\ \vdots\\ -y(t-n)\\ u(t-1)\\ \vdots\\ u(t-n)\end{bmatrix}e(t)]$$

$$e(t) = w(t)c_1w(t-1) + \cdots + c_nw(t-n)$$

$$r_{\bar{\varphi}e} = \begin{bmatrix}\\ \\ 0\\ 0\\ \vdots\\ 0\end{bmatrix}\text{ with n zeros}$$

$y(t-1) = f(w(t-1), w(t-2, \dots)$
$e(t) = f(w(t), w(t-1), \dots, w(t-n)$ so there is a correlation between the terms of y and those of e, so the first elements of the above vector are not zero, therefore

$$\lim_{N\to\infty}\hat{\hat{\theta}}_{LS} = \theta^* + \Sigma_{\hat{\varphi}}^{-1}r_{\bar{\varphi}e} \neq 0$$

so the estimator is biased.

General model structure:

$$y(t) = G(z^{-1})u(t) + H(z^{-1})w(t)$$

where $w(t)$ is a zero mean white process with variance σ_w^2 and uncorrelated with $u(t)$
Available measurements:

$$y(1-n), y(2-n), \dots, y(N), u(1-n), u(2-n), \dots, u(N)$$

Prediction error method: find the estimate $\hat{\theta}$ that minimizes the loss function

$$J(\theta) = \frac{1}{N}\sum_{t=1}^{N}\varepsilon^2(t,\theta) = \frac{1}{N}\sum_{t=1}^{N}(y(t) - \hat{y}(t|t-1,\theta))^2$$

where $\hat{y}(t|t-1,\theta)$ is the *optimal one step ahead prediction* of $y(t)$. Optimal (minimal variance) prediction: a prediction of $y(t)$ given θ and the past I/O data up to time $t-1$ s.t. the variance of the prediction error $\varepsilon(t)$ is minimal. The model can be rewritten as

$$y(t) = G(z^{-1})u(t) + (H(z^{-1}) - 1)w(t) + w(t)$$

by replacing

$$w(t) = \frac{1}{H(z^{-1})}y(t) - \frac{G(z^{-1})}{H(z^{-1})}u(t)$$

we get

$$y(t) = Gu + (H-1)\left(\frac{1}{H}y - \frac{G}{H}u\right) + w$$

from which we get

$$y(t) = \left(1 - \frac{1}{H(z^{-1})}\right) y(t) + \frac{G(z^{-1})}{H(z^{-1})} u(t) + w(t)$$

from which it is easy to prove

$$\hat{y}(t|t-1,\theta) = \left(1 - \frac{1}{H(z^{-1})}\right) y(t) + \frac{G(z^{-1})}{H(z^{-1})} u(t)$$

For any other predictor $y^p(t)$ we have

$$E[(y(t) - y^p(t))^2] \geq E[(y(t) - \hat{y}(t|t-1,\theta))^2]$$

$$E[(y(t) - y^p(t))^2] = E[(Gu(t) + Hw(t) - y^p(t))^2]$$
$$= E[\left(\left(1 - \frac{1}{H}\right) y + \frac{G}{H} u + w(t) - y^p(t)\right)^2]$$

We can notice that $w(t)$ is uncorrelated with all other members of the sum as $y^p(t)$ depends only on the first $t-1$ samples, therefore

$$= E\left[\left(\left(1 - \frac{1}{H}\right) y + \frac{G}{H} u - y^p(t)\right)^2\right] + E[w(t)^2] = E\left[\left(\left(1 - \frac{1}{H}\right) y + \frac{G}{H} u - y^p(t)\right)^2\right] + \sigma_{w^2}$$

6.0.1 The Newton-Raphson Algorithm

It is an algorithm for zero-finding. $J(\theta)$ is approximated around $\hat{\theta}^k$ with a quadratic function $V(\theta)$

$$V(\theta) = J(\hat{\theta}^k) + J'(\hat{\theta}^k)(\theta - \hat{\theta}^k) + \frac{1}{2}(\theta - \hat{\theta}^k)^T J''(\hat{\theta}^k)(\theta - \hat{\theta}^k)$$

The algorithm updates $\hat{\theta}^k$ with the position of the minimum of $V(\theta)$. For this purpose, we compute the derivative of $V(\theta)$:

$$V'(\theta) = J'(\hat{\theta}^k)^T + J''(\hat{\theta}^k)\theta - J''(\hat{\theta}^k)\hat{\theta}^k \tag{6.1}$$

By setting this to zero we get our update:

$$\hat{\theta}^{k+1} = \hat{\theta}^k - J''(\hat{\theta}^k)^{-1} J'(\hat{\theta}^k)^T$$

6.0.2 The Gauss-Newton Method

the application to PEM of the Newton-Raphson algorithm. Let us define

$$\psi(t,\theta) = -\left(\frac{\partial \varepsilon(t,\theta)}{\partial \theta}\right)^T$$

we get

$$J'(\theta) = -\frac{2}{N} \sum_{t=1}^{N} \varepsilon(t,\theta) \psi^T(t,\theta)$$

and

$$J''(\theta) = \frac{2}{N} \sum_{t=1}^{N} \psi(t,\theta)\psi^T(t,\theta) - \frac{2}{N} \varepsilon(t,\theta) \frac{\partial^2 \varepsilon(t,\varepsilon)}{\partial \theta^2} \approx \frac{2}{N} \sum_{t=1}^{N} \psi(t,\theta)\psi^T(t,\theta) \geq 0$$

therefore the update rule is given by:

$$\hat{\theta}^{k+1} = \hat{\theta}^k + \eta^k (\frac{2}{N} \sum_{t=1}^{N} \psi(t,\theta)\psi^T(t,\theta) \geq 0)^{-1} \frac{2}{N} \sum_{t=1}^{N} \varepsilon(t,\theta)\psi^T(t,\theta)$$

- the approximated Hessian is positive definite, guaranteeing that the update happens in a descending direction
- the algorithm can stop at a local minimum
- convergence is dependant on the initialization

6.0.3 Identification of ARMAX models

Model:

$$A(z^{-1})y(t) = B(z^{-1})u(t) + C(z^{-1})w(t)$$

the optimal predictor is given by:

$$\hat{y}(t|t-1,\theta) = \frac{C(z^{-1}) - A(z^{-1})}{C(z^{-1})}y(t) + \frac{B(z^{-1})}{C(z^{-1})}u(t)$$

from which we derive

$$C(z^{-1})\varepsilon(t) = A(z^{-1})y(t) - B(z^{-1})u(t)$$

The residual is therefore given by:

$$\varepsilon(t,\theta) = -c_1\varepsilon(t-1,\theta) - c_2\varepsilon(t-2,\theta), \cdots - c_n\varepsilon(t-n,\theta) + y(t) + a_1y(t-1) + \cdots + b_nu(t-n)$$

We must now derive $\psi(t,\theta)$. From the above relation we can derive

$$\frac{\partial\varepsilon(t,\theta)}{\partial a_i} = \frac{1}{C(z^{-1})}y(t-i) \qquad \frac{\partial\varepsilon(t,\theta)}{\partial b_i} = -\frac{1}{C(z^{-1})}u(t-i)$$

$$\frac{\partial\varepsilon(t,\theta)}{\partial c_i} = -\frac{1}{C(z^{-1})}\varepsilon(t-i,\theta)$$

by introducing the filtered signals

$$y^F(t) = \frac{1}{C(z^{-1})}y(t), \qquad u^F(t) = \frac{1}{C(z^{-1})}u(t), \qquad \varepsilon^F(t)\frac{1}{C(z^{-1})}\varepsilon(t)$$

we have

$$\psi(t,\theta) = \begin{bmatrix}-y^F(t-1) & \cdots & -y^F(t-n) & u^F(t-1) & \cdots & u^F(t-n) & \varepsilon^F(t-1) & \cdots & \varepsilon^F(t-n)\end{bmatrix}^T$$

6.0.4 Identification of ARARX models

The optimal predictor is

$$\hat{y}(t|t-1\theta) = (1 - a(z^{-1})D(z^{-1}))y(t) + B(z^{-1})D(z^{-1})u(t)$$

which can be seen as

$$\hat{y}(t|t-1\theta) = (1 - \bar{A}(z^{-1})) - \bar{B}(z^-1)u(t)$$

which is equivalent to an ARX model. One could apply LQ estimation to the model and try to identify the model by finding common roots between $\bar{A}$ and $\bar{B}$ or if just a predictive model is required LQ estimation itself is sufficient.

6.0.5 Statistical properties of PEM estimators

Assume that a true model exists and

- the input is persistently exciting of sufficiently high order
- The Hessian $J''(\theta)$ is nonsingular at least locally around the minimum points of $J(\theta)$

In this case, the PEM estimate is consistent (proof skipped)

$$\lim_{N\to\infty} \hat{\theta} = \theta^*$$

Moreover

$$\sqrt{N}(\hat{\theta} - \theta^*) \to \mathcal{N}(0,P), \quad \text{for } N \to \infty$$

with

$$P = \sigma_w^2(E[\psi(t,\theta^*)\psi(t,\theta^*)^T])^{-1} = \sigma_w^2\Sigma_\psi^{-1}$$

If $w(t)$ is gaussian distributed

$$w(t) \sim \mathcal{N}(0,\sigma_w^2)$$

The PEM estimate is also asymptotically efficient, therefore P is the lowest covariance matrix of the estimate that can be obtained.

6.0.6 MISO ARX models

$$u(t) \in \mathbb{R}^r \quad u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_r(t) \end{bmatrix}$$

$$A(z^{-1})y(t) = B_1(z^{-1})u_1(t) + \cdots + B_r u_r(t) + e(t)$$

$$B_1(z^{-1}) = b_{11}z^{-1} + b_{12}z^{-2} + \cdots + b_{1n}z^{-n}$$

$$B_2(z^{-1}) \text{ similar } P = n + rn = (r+1)n$$

$$H = \begin{bmatrix} -H_y(n) & H_{u_1}(n) & \dots & H_{u_n}(n) \end{bmatrix}$$

$$\hat{\theta}_{LS} \left(\frac{H^T H}{N} \right)^{-1} \frac{H^T Y}{N}$$

$$H^T H \theta = H^T Y$$

Chapter 7

Statistical hypothesis testing

Chapter 8

Model complexity selection

Model complexity selection: given a model class $\mathcal{M}(\theta)$, estimate the model complexity p, i.e. choose the best model class $\mathcal{M}_p(\theta)$

Training set: the set of data used for learning the model
Underfitting: the model is not rich enough to fit the data well
Overfitting: the model is too rich and adapts too closely to the training data
Validation set: the set of data used for evaluating the predictive capabilities of the models obtained with the training set in order to estimate the model complexity.

- General rule: 60-70% of the data used for training set, rest for validation
- when using the training set, the higher the model complexity, the better the data fitting. the prediction error is thus underestimated
- when dealing with real data, the loss function exhibits a monotone decrease in the training set and a *U-shape* in the validation set.

ARX MODEL:

$$\hat{\theta}_{LS} \to \theta^* \text{ for } N \to \infty$$

$$\varepsilon(t) \to w(t) \text{ for } N \to \infty$$

$$J(\hat{\theta}_{LS}) = \frac{1}{N}\sum_{t=1}^{N} \varepsilon(t, \hat{\theta}_{LS})^2 = \hat{\sigma}_\varepsilon^2$$

$$J(\hat{\theta}_{LS}) \to \sigma_w^2 \text{ for } N \to \infty$$

$$n = 4 \quad \theta^* = \begin{bmatrix} a_1^* & a_3^* & a_3^* & a_4^* & b_1^* & b_2^* & b_3^* & b_4^* \end{bmatrix}$$

If the validation set is used repeatedly to estimate the model complexity, the prediction error may be underestimated as well. This happens for very complex models like neural networks. For this reason, when dealing with neural networks the dataset is split into three parts. The third set is called *test set* and is used for model assessment, i.e. for testing the predictive capabilities of the final chosen model. This requires a very large number of available samples. It is possible to design criteria that allow to estimate the model complexity by using the training test

8.1 The F-test

Let $\mathcal{M}_{p_1}(\theta), \mathcal{M}_{p_2}(\theta)$ such that $p_1 < p_2$ ($p = 2n$ for ARX models, $p = n$ for AR models etc.) Consider the test quantity

$$x = N\frac{J(\theta_N^{\hat{1}})J(\theta_N^{\hat{2}})}{J(\theta_N^{\hat{2}})}$$

where $\hat{\theta}_N^1, \hat{\theta}_N^2$ are PEM (or LS) estimates: intuitively:

1. x large: the decrease in the loss function is significant, hence $\mathcal{M}_{p_2}(\theta)$ is better
2. x small: $\mathcal{M}_{p_1}(\theta)$ and $\mathcal{M}_{p_2}(\theta)$ are almost equivalent so that $\mathcal{M}_{p_1}(\theta)$ should be chosen according to the parsimony principle

How to quantify "large" and "small"?

- If $\mathcal{M}_{p_1}(\theta)$ is not large enough to include the true system:

$$J(\hat{\theta}_N^1) - J(\hat{\theta}_N^2) \text{ is } O(1) \quad x \text{ is of magnitude } N$$

- If $\mathcal{M}_{p_1}(\theta)$ is large enough:

$$x \to \chi^2(p_2 - p_1), \quad \text{for } N \to \infty$$

The following statistical test can be performed:

$$\begin{cases} H_0 : \mathcal{M}_{p_1}(\theta) \text{ is suitable to describe the system} \\ H_1 : \mathcal{M}_{p_1}(\theta) \text{ is not suitable} \end{cases}$$

that is:

$$\begin{cases} H_0 : x \sim \chi^2(p_2 - p_1) \\ H_1 : \text{not } H_0 \end{cases}$$

after the choice of the significance level:

$$\begin{cases} x \leq \chi^2_\alpha(p_2 - p_1) \implies \text{accept } H_0 \\ x > \chi^2_\alpha(p_2 - p_1) \implies \text{accept } H_1 \end{cases}$$

8.2 The final prediction error (FPE) criterion

Let $\hat{\theta}_N$ be a PEM (or LS) estimate of a model of complexity p and assume that it is then used to predict future data. Assume also that a true model exists and consider the prediction error variance (the expectation is with respect to future data):

$$V(\hat{\theta}_N) = E\left[\left(y(t) - \hat{y}(t|t-1, \hat{\theta}_N)\right)^2\right]$$

by replacing $y(t) = \varphi^T(t)\theta^* + w(t)$ in $V(\hat{\theta}N)$ and computing the expectation we get

$$V(\hat{\theta}_N) = \sigma_w^2 + (\hat{\theta}_N - \theta^*)^T \Sigma_\varphi (\hat{\theta}_N - \theta^*)$$

consider now the criterion function

$$FPE = E\left[V(\hat{\theta}_N)\right]$$

where the expectation is with respect to past data. By taking into account that:

- $E[(\hat{\theta}_N - \theta^*)^T \Sigma_\varphi (\hat{\theta}_N - \theta^*)] = E[\text{trace}(\Sigma_\varphi (\hat{\theta}_N - \theta^*)(\hat{\theta}_N - \theta^*)^T]$[1]
- Asymptotically: $\sqrt{N}(\hat{\theta}_N - \theta^*) \sim \mathcal{N}(0, \sigma_w^2 \Sigma_\varphi^{-1})$

it is easy to obtain

$$FPE \approx \sigma_w^2 \left(1 + \frac{p}{N}\right)$$

in practice we need an asymptotically unbiased estimate of σ_ω^2

$$\hat{\sigma}_\omega^2 = \frac{1}{N-p} \sum_{t=1}^{N} \varepsilon(t, \hat{\theta}_N)^2$$

---

[1] E[trace($v^T A v$)] =E[trace($A v v^T$)]

so that

$$FPE = \hat{\sigma}_e^2 \frac{N+p}{N} = \frac{N+p}{N-p} J(\hat{\theta}_N) = \frac{N\left(1+\frac{p}{N}\right)}{N\left(1-\frac{p}{N}\right)} J(\hat{\theta}_N) = \frac{1+\frac{p}{N}-\frac{p}{N}+\frac{p}{N}}{1-\frac{p}{N}} J(\hat{\theta}_N) = J(\hat{\theta}_N) + \frac{2\frac{p}{N}}{1-\frac{p}{N}} J(\hat{\theta}_N)$$

Note that for large N:

$$FPE \approx J(\hat{\theta}_N) + \frac{2p}{N} J(\hat{\theta}_N)$$

for large N, this criterion belongs to the family of *criteria with complexity terms* (terms that penalise complex models).

The criterion is applied by minimizing the value of $FPE(n)$

8.3 Criteria with complexity terms

These criteria are obtained by penalizing in some way the decrease of $J(\hat{\theta}_N)$with increasing orders. The order giving the smallest value fo the criterion is selected. General form:

$$V(\hat{\theta}_N) = Nlog(\hat{\theta}_N) + f(N,p)$$

where $f(N,p)$ penalizes high order models.

8.3.1 Akaike information criterien (AIC)

$$AIC = NlogJ(\hat{\theta}_N) + 2p$$

AIC and FPE are asymptotically equivalent. They do not give consistent estimates of n (the probability of overestimating the order is non-null). To get consistent estimates, the penalizing function must be such that

$$\begin{cases} f(N,p) = kpg(N) \\ \lim_{N\to\infty} g(N) = \infty \\ \lim_{N\to\infty} \frac{g(N)}{N} = 0 \end{cases}$$

Minimum description lenght (MDL) criterion

$$MDL = NlogJ(\hat{\theta}_N) + 2plog(N)$$

MDL leads. in general, to models of lower complexity wrt AIC and FPE. Even though the derivation is different, the MDL approach is formally equivalent to the *bayesian information criterion (BIC)*

Chapter 9

Model assesment (validation)

Consists in evaluating the capability of the identified model to describe the process that has generated the data in a way compatible with its planned use.

Linear regression models:

$$y(t) = \varphi^T(t)\theta^* + w(t), \quad w(t) \text{ zero mean and white} \quad E[u(t)w(t-\tau)] = 0, \forall\tau$$

The PEM estimate $\hat{\theta}_N$ is consistent, then

$$\hat{\theta}_N \to \theta^* \text{ for } N \to \infty, \quad \varepsilon(t, \hat{\theta}_N) = y(t) - \varphi^T(t)\hat{\theta}_N \to w(t) \text{ for } N \to \infty$$

If we assume that our real data are well described by a linear regression model, we can make the following assumptions about the residual $\varepsilon(t, \hat{\theta}_N)$:

1. $\varepsilon(t, \hat{\theta}_N)$ is a zero mean white process
2. $\varepsilon(t, \hat{\theta}_N)$ is is uncorrelated with the input signal $u(t)$

It is thus possible to perform (on the training set) the following *test on residuals*:

- test of whiteness of $\varepsilon(t, \hat{\theta}_N)$
- test of cross-correlation between $\varepsilon(t, \hat{\theta}_N)$ and $u(t)$

9.1 Whiteness test

Sequence of residuals: $\varepsilon(1, \hat{\theta}_N), \varepsilon(2, \hat{\theta}_N), \dots, \varepsilon(N, \hat{\theta}_N)$

$$\begin{cases} H_0 : \varepsilon(t, \hat{\theta}_N) \text{ is a zero mean white process} \\ H_1 : \text{ not } H_0 \end{cases}$$

Consider the sample variance $\hat{r}_\varepsilon(0)$ and the first m sample autocorrelations of $\varepsilon(t)$ and define the vector

$$\hat{r}_\varepsilon = \begin{bmatrix} \hat{r}_\varepsilon(1) \\ \hat{r}_\varepsilon(2) \\ \vdots \\ \hat{r}_\varepsilon(m) \end{bmatrix}$$

Under H_0:

$$\hat{r}_\varepsilon(0) \to \sigma_w^2 \text{ for } N \to \infty, \quad \hat{r}_\varepsilon(\tau) \to 0 \text{ for } N \to \infty, \forall\tau \neq 0$$

and it is possible to prove that

$$\sqrt{N}\hat{r}_\varepsilon \xrightarrow[N\to\infty]{\sim} \mathcal{N}(0, P), \quad P = \lim_{N\to\infty} E[N\hat{r}_\varepsilon\hat{r}_\varepsilon^T] = \sigma_w^4 I \quad (*)$$

as a consequence

$$x = N\frac{\hat{r}_\varepsilon^T \hat{r}_\varepsilon}{\hat{r}_\varepsilon^2(0)} \xrightarrow[N\to\infty]{\sim} \chi^2(m)$$

This leads to the statistical test

$$\begin{cases} x \le \chi^2_\alpha(m) & \Longrightarrow \quad \text{accept } H_0 \\ x > \chi^2_\alpha(m) & \Longrightarrow \quad \text{accept } H_1 \end{cases}$$

where α is the chosen significance level. Define the normalized text quantities

$$\hat{\gamma}(\tau) = \frac{\hat{r}_\varepsilon(\tau)}{\hat{r}_\varepsilon(0)}, \quad \tau = 1, 2, \dots, m$$

From (*) it follows that:

$$\sqrt{N}\hat{\gamma}(\tau) \xrightarrow[N\to\infty]{\sim} \mathcal{N}(0,1), \quad \tau = 1, 2, \dots, m$$

so that a set of m gaussian tests can also be performed. The whiteness model test can be of help in model order estimation.

9.2 Test of cross-correlation

$$\begin{cases} H_0 : \varepsilon(t\hat{\theta}_N) \text{ and } u(t) \text{ are uncorrelated} \\ H_1 : \text{ not } H_0 \end{cases}$$

Consider the following vector of sample cross correlations:

$$\hat{r}_{\varepsilon u} = \begin{bmatrix} \hat{r}_{\bar{\tau}+1} \\ \hat{r}_{\bar{\tau}+2} \\ \vdots \\ \hat{r}_{\bar{\tau}+m} \end{bmatrix}$$

Consider also the vector

$$\varphi_u(t,m) = \begin{bmatrix} u(t-1) & u(t-2) & \cdots & u(t-m) \end{bmatrix}^T$$

and the sample autocorrelation matrix $\hat{\Sigma}_u(m)$, which is an estimate of $E[\varphi_u(t,m)\varphi_u^T(t,m)]$

It is possible to prove that

$$\sqrt{N}\hat{r}_{\varepsilon u} \xrightarrow[N\to\infty]{\sim} \mathcal{N}(0,P), \quad P = \lim_{N\to\infty} E[N\hat{r}_{\varepsilon u}\hat{r}_{\varepsilon u}^T] = \sigma_w^2 \Sigma_u(m)$$

As a consequence

$$x = N\frac{\hat{r}_{\varepsilon u}^T \hat{\Sigma}_u^{-1} \hat{r}_{\varepsilon u}}{\hat{r}_{\varepsilon u}(0)} \xrightarrow[N\to\infty]{\sim} \chi^2(m)$$

This leads to the statistical test

$$\begin{cases} H_0 : x \le \chi^2_\alpha(m) & \Longrightarrow \quad \text{accept } H_0 \\ H_0 : x > \chi^2_\alpha(m) & \Longrightarrow \quad \text{accept } H_1 \end{cases}$$

where α is the chosen significance level. $\bar{\tau}$ must be chosen with care. In some methods $\hat{r}_{\varepsilon u}(\tau)$ is constrained to be zero for some values of τ by construction. We define the normalized test quantities

$$\hat{\gamma}(\tau) = \frac{\hat{r}_{\varepsilon u}(\tau)}{\sqrt{\hat{r}_\varepsilon(0)\hat{r}_u(0)}}, \quad \tau = 1, 2, \dots, m$$

It follows that

$$\sqrt{N}\hat{\gamma} \xrightarrow[N\to\infty]{\sim} \mathcal{N}(0,1), \quad \tau = 1, 2, \dots, \dots, m$$

so that a set of m gaussian tests can also be performed. Once the significance level α has been chosen, m gaussian tests are performed and the final decision is taken by considering the Anderson test.

Chapter 10

Maximum likelihood estimation

Let $y(t)$ be an observed stochastic process whose distribution depends on an uknown parameter vector θ. The *likelihood function* $L(\theta)$ is the pdf of $y(t)$ given θ:

$$L(\theta) = f(y(t)|\theta)$$

The maximum likelihood estimate of θ is

$$\hat{\theta}_{ML} = \arg\max_{\theta} L(\theta)$$

- $\hat{\theta}_{ML}$ is the vector θ that makes the a posteriori probability of the observations as large as possible
- ML estimates require the knowledge of the conditional pdf and are often difficult to calculate analytically
- In the gaussian case the problem often becomes more tractable. For dynamic systems, the problem can be simplified if the effect of initial conditions is neglected.

10.1 the gaussian case

$$y(t) = G(z^{-1})u(t) + H(z^{-1})w(t) \qquad\qquad w(t) \sim \mathcal{N}(0, \sigma_w^2)$$

Neglecting the effect of the initial conditions, the pdf $f(y(t)|\theta)$ can be replaced with $f(w(t)|\theta)$. By introducing also the approximation $\varepsilon(t,\theta) \approx w(t)$ we can write

$$L(\theta) = \frac{1}{\sqrt{(2\pi\sigma_\varepsilon^2)^N}} \exp\left(-\frac{1}{2\sigma\varepsilon^2}\sum_{t=1}^{N}\varepsilon^2(t,\theta)\right)$$

By considering the logarithmic likelihood function (log-likelihood function)

$$\log L(\theta) = -\frac{N}{2}\log 2\pi - \frac{N}{2}\log\sigma_\varepsilon^2 - \frac{1}{2\sigma_\varepsilon^2}\sum_{t=1}^{N}\varepsilon^2(t,\theta) \quad (*) \tag{10.1}$$

Assume now that σ_ε^2 is an additional parameter to be estimated $\implies \log L(\theta, \sigma_\varepsilon^2)$. First maximize wrt σ_ε^2 by solving

$$\frac{\partial \log L(\theta, \sigma_\varepsilon^2)}{\partial \sigma_\varepsilon^2} = 0$$

We get

$$\sigma_\varepsilon^2 = \frac{1}{N}\sum_{t=1}^{N}\varepsilon^2(t,\theta) = J(\theta)$$

Then inserting the above expression in 10.1 we find

$$\log L(\theta) = -\frac{N}{2}\log 2\pi - \frac{N}{2} - \frac{N}{2}\log J(\theta)$$

$\hat{\theta}_{ML}$ is thus obtained by minimizing $J(\theta)$ $\implies$ the PEM estimate can be interpreted as the ML estimate providede that $w(t)$ is gaussian distributed

- As a consequence, the least squares estimate of static models, FIR models, ARX models and AR models, can be interpreted as the ML estimate provided that $w(t)$ is gaussian distributed
- Strictly speaking, for dynamic models $L(\theta)$ is not the exact likelihood function, because of the effect of the initial conditions.
- When the number of samples is large, the difference between the exact and the approximated likelihood functions becomes small.

10.1.1 the Cramér-Rao lower bound

Let Y be a stochastic vector and $L(\theta^*) = g(Y|\theta)$. For an arbitrary unbiased estimate $\hat{\theta}(Y)$ of θ^* determined from Y we have

$$cov(\hat{\theta}(Y)) \geq \left[E\left(\frac{\partial \log L(\theta^*)}{\partial \theta^*}^T\right)\frac{\partial \log L(\theta^*)}{\partial \theta^*}\right]^{-1} = \left[E\frac{\partial^2 \log L(\theta^*)}{\partial \theta^{*2}}\right]^{-1} = F^{-1}$$

Where F is called the *Fisher information matrix.* The matrix F^{-1} is thus a lower bound on the covariance matrix of the estimate and is called *Cramér-Rao lower bound (CRLB).*

- if an estimate attains the CRLB, it is efficient
- the maximum likelihood estimator is (in geneal) consistent and efficient
- The least squares estimate of static models, FIR models, ARX models and AR models is (asymptotically) efficient provided that $w(t)$ is gaussian distributed
- the PEM estimate of ARMAX, ARMA and ARARX models is asymptotically efficient provided that $w(t)$ is gaussian distributed.

We write the likelihood function for a LS estimate:

$$L(\theta^*) = \frac{1}{\sqrt{(2\pi\sigma_w^2)^N}} \exp\left(-\frac{1}{2\sigma_w^2}\sum_{t=1}^{N} w^2(t)\right) = \frac{1}{\sqrt{(2\pi\sigma_w^2)^N}} \exp\left(-\frac{1}{2\sigma_w^2}(Y - \Phi\theta^*)^T(Y - \Phi\theta^*)\right)$$

and therefore the log-likelihood function is

$$\log L(\theta^*) = -\frac{N}{2}\log 2\pi - \frac{N}{2}\log \sigma_w^2 - \frac{1}{w\sigma_\varepsilon^2}(Y - \Phi\theta^*)^T(Y - \Phi\theta^*)$$

from which

$$\frac{\partial^2 \log L(\theta^*)}{\partial \theta^{*2}} = \frac{\Phi^T\Phi}{\sigma_w^2} = F$$

so the CRLB turns out to be

$$F^{-1} = (\Phi^T\Phi)^{-1}\sigma_w^2$$

which corresponds to the covariance of an LS estimate when the noise is zero-mean gaussian distributed. This implies that the LS estimate in this case is efficient.

Chapter 11

Classification: probabilistic models

Classification: assigning the input $u(t) \in \mathcal{U}$ to one of M classes $C_1, C_2, \ldots, C_M$. The input is numerical and the output is categorical. Available data:

$$(u(1), y(1)), (u(2), y(2)), \ldots, (u(N), y(N))$$

A possible way to represent class labels numerically is

$$y(t) = y^i \quad \text{if} \quad u(t) \in C_i$$

where y^i is a $M \times 1$ vector whose elements are zero except for the i-th element, which is equal to 1. For a two-class problem we will use

$$y(t) = \begin{cases} 1 & \text{if } u(t) \in C_1 \\ 0 & \text{if } u(t) \in C_2 \end{cases}$$

The input space $\mathcal{U}$ is divided into M regions labeled according to the classification rule. The boundaries of these regions are called *decision boundaries.*

Deterministic models: the decision boundaries are explicitly modeled. This involves the construction of a *discriminant function* that directly assigns each input $u(t)$ to a specific class.

Probabilistic models: first, the posterior probabilities $P(C_i|u(t)) = P(y(t) = y^i|u(t)),\ i = 1, 2, \ldots, M$ are estimated. Then, these probabilities are used to decide to which class a given input $u(t)$ belongs.

The performance of a classifier can be evaluated by computing the error rate:

$$Er = \frac{1}{N} \sum_{t=1}^{N} I(y(t), \hat{y}(t))$$

where $\hat{y}(t)$ is the prediction of $y(t)$ provided by the classifier and $I(y(t)\hat{y}(t))$ is an *indicator variable*

$$I(y(t)\hat{y}(t)) = \begin{cases} 1 & \text{if } y(t) \neq \hat{y}(t) \\ 0 & \text{if } y(t) = \hat{y}(t) \end{cases}$$

The error rate is thus the fraction of misclassified observations.

- The error rate can be evaluated on the training set (training error rate), on the validation set (validation error rate) and on the test set (test error rate)
- A good classifier is one for which the validation (test) error rate is small
- As for regression problems, choosing the model complexity by using the training error rate leads to overfitting.

11.1 The Bayes classifier

The Bayes classifier assigns each input to the most likely class. Given an input $u(t) \in \mathcal{U}$, it will be assigned to the class C_i with largest posterior probability.

$$C_i : \underset{i=1,2,\ldots,M}{\arg\max} \{P(C_i|u(t)) = P(y(t) = y^i|u(t))\}$$

It maximizes the conditional probability that $y(t) = y^i$ given the observed input value $u(t)$.

- In a two-class problem, the Bayes classifier consists in assigning $u(t)$ to class C_1 if $P(C_1|u(t)) > 0.5$ and to class C_2 otherwise
- It is possible to show that this classifier leads to the minimum classification (prediction) error on a future unseen dataset (validation or test set)
- When dealing with real data, the conditional distribution of $\mathcal{Y}$ given $\mathcal{U}$ is usually uknown so that the Bayes classifier cannot be used in practice
- Probabilistic models use estimated probabilities $\hat{P}(C_i|u(t)),\ i = 1, 2, \ldots, M$ rather than the true ones

11.2 Logistic regression

Consider a two-class classification problem with r inputs ($u(t) = [u_1(t) \ldots u_r(t)]^T$) and assume a linear decision boundary. This boundary is a hyperplane in $\mathbb{R}^r$ described by the equation

$$\beta_0 + \beta_1 u_1(t) + \beta_2 u_2(t) + \cdots + \beta_r u_r(t) = 0$$

that is

$$\varphi^T(t)\theta = 0$$

with $\varphi(t) = \begin{bmatrix}1 & u_1(t) & u_2(t) & \cdots & u_r(t)\end{bmatrix}^T$ and $\theta = \begin{bmatrix}\beta_0 & \beta_1 & \cdots & \beta_r\end{bmatrix}^T$

How to model $P(C_1|u(t))$? We need a function $f(z(t))$, where $z(t) = \varphi^T(t)\theta$, such that

- $f(z)$ takes values in the range $(0, 1)$
- $f(0) = 0.5$
- $f(z) > 0.5$ ($f(z) < 0.5$) when $z > 0$ ($z < 0$)
- $f(z) \to 1$ when $z \to \infty$
- $f(z) \to 0$ when $z \to -\infty$

Logistic sigmoid function:

$$f(z) = \frac{e^z}{1 + e^z} = \frac{1}{1 + e^{-z}}$$

We have

$$P(C_1|u(t)) = f(z(t)), \quad P(C_2|u(t)) = 1 - f(z(t)) = \frac{1}{1 + e^z}$$

from which we can simply derive

$$z(t) = \log \frac{P(C_1|u(t))}{P(C_2|u(t))} = \log \frac{f(z)}{1 - f(z)}$$

How to estimate θ?

Find the estimate $\hat{\theta}$ that maximizes the posterior probability of the observation $Y = \begin{bmatrix}y(1) & y(2) & \cdots & y(N)\end{bmatrix}^T \implies$ maximum likelihood estimation

$$L(\theta) = P(Y|\theta) = \prod_{t=1}^{N} P(C_1|u(t))^{y(t)} P(C_2|u(t))^{1-y(t)} = \prod_{t=1}^{N} f(z(t))^{t(t)} (1 - f(z(t)))^{1-y(t)}$$

Log-likelihood function:

$$\log P(Y|\theta) = \sum_{t=1}^{N} [y(t) \log f(z(t)) + (1 - y(t)) \log(1 - f(z(t)))]$$

Maximizing the above function is equivalent to minimizing its negative, so we have to find the estimate $\hat{\theta}$ minimizing

$$J(\theta) = -\log P(Y|\theta)$$

To solve this optimization problem we can use the Newton-Raphson algorithm:

$$\hat{\theta}^{k+1} = \hat{\theta}^{K} - \eta^k J''(\hat{\theta}^k)^{-1} J'(\hat{\theta}^k)^T$$

By using

$$\frac{\partial f(z)}{\partial z} = f(z)(1 - f(z))$$

It is possible to show that

$$J'(\theta) = \frac{J(\theta)}{\partial \theta} = \sum_{t=1}^{N} [f(z(t)) - y(t)]\varphi(t) = \Phi^T (F(\theta) - Y)$$

where

$$\Phi = \begin{bmatrix} 1 & u_1(1) & \cdots & u_r(1) \\ 1 & u_1(2) & \cdots & u_r(2) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & u_1(N) & \cdots & u_r(N) \end{bmatrix} \quad Y = \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(N) \end{bmatrix} \quad F(\theta) = \begin{bmatrix} f(z(1)) \\ f(z(2)) \\ \vdots \\ f(z(N)) \end{bmatrix}$$

and

$$J''(\theta) = \frac{\partial^2 J(\theta)}{\partial \theta^2} = \sum_{t=1}^{N} f(z(t))(1 - f(z(t)))\varphi(t)\varphi^T(t) = \Phi^T W(\theta) \Phi$$

where $W(\theta)$ is the diagonal matrix

$$W(\theta) = diag \begin{bmatrix} f(z(1))(1 - f(z(1))) & f(z(2))(1 - f(z(2))) & \cdots & f(z(N))(1 - f(z(N))) \end{bmatrix}$$

The iterative algorithm is

$$\hat{\theta}^{k+1} = \hat{\theta}^k - \eta^k (\Phi^T W(\theta) \Phi)^{-1} \Phi^T (F(\theta) - Y)$$

- the elements of $W(\theta)$ are in the range $(0, 1)$, therefore the hessian $\Phi^T W(\theta)\Phi$ is positive definite and the loss function $J(\theta)$ has a unique minumum.
- $\hat{\theta}^0$ is ofthen chosen as a vector of zeros. It can also be chosen randomly.

How to classify a new input $u(t)$?

Once an estimate $\hat{\theta}$ has been computed and a new input $u(t)$ becomes available:

1. Compute $\hat{z}(t) = \varphi^T(t)\hat{\theta}$
2. Compute $f(\hat{z}(t)) = \hat{P}(C_1|u(t)) = \frac{e^{\hat{z}}}{1 + e^{\hat{z}}}$, which is an estimate of $P(C_1|u(t))$
3. If $f(\hat{z}(t)) > 0.5$ assign $u(t)$ to class C_1 ($\hat{y}(t) = 1$), otherwise assign it to class C_2 ($\hat{y}(t) = 0$)

Depending on the specific classification problem, the decision criterion could be different from that described in the above step 3

11.2.1 The gradient descent algorithm

$$\hat{\theta}^{k+1} = \hat{\theta}^k - \eta^k J'(\hat{\theta}^k)$$

- It only requires computation of the gradient of the loss function
- At each step $\hat{\theta}^k$ is moved in the direction of the greatest rate of decrease of the loss function $J(\theta)$
- The step size η^k (also called *learning rate*) plays a very important role. Both very large and very small values can lead to inefficiency. A variable step size is often adopted
- The algorithm can converge to a local minimum
- The procedure can be repeated by starting from different initial guesses
- *Stopping criterion*: $\|\hat{\theta}^{k+1} - \hat{\theta}^k\| < \delta$ or $|J(\hat{\theta}^{k+1}) - J(\hat{\theta}^k)| < \delta$ or $\|J'(\hat{\theta}^k)\| < \delta$ where δ is a small positive number

For the logistic regression problem, the iterative algorithm is

$$\hat{\theta}^{k+1} = \hat{\theta}^k - \eta^k \Phi^T (F(\theta) - Y)$$

Stochastic gradient descent

Instead of using all samples, a smaller number of samples is randomly picked for gradient computation at each iteration.

Confusion matrix

Suppose a training set is used to obtain an estimate on a two-class classification problem, and a validation set is used to verify the predictive capabilities of the model, with 5000 samples corresponding to class C_1 (H) and 1000 samples belonging to class C_2 (F) the confusion matrix is:

	H	F	Total
H	4950	250	5200
F	50	750	800
Total	5000	1000	6000

The error rate is:

$$Er = \frac{300}{6000} = 5\%$$

The error rate concerning the first class is

$$Er_H = \frac{50}{5000} = 1\%$$

The error rate concerning the second class is

$$Er_H = \frac{250}{1000} = 25\%$$

If these errors are tolerable, we may keep the chosen decision rule, otherways we may change it. New decision rule: $\hat{P}(C|u(t)) > 0.75 \implies u(t) \in C_1$ leading to a new confusion matrix:

	H	F	Total
H	4750	110	4860
F	250	890	1140
Total	5000	1000	6000

The error rate is:

$$Er = \frac{360}{6000} = 6\%$$

The error rate concerning the first class is

$$Er_H = 5\%$$

The error rate concerning the second class is

$$Er_H = 11\%$$

11.2.2 Multiclass problems

One-vs-all

the multiclass problem is split into M two-class problems. The generic i-th binary problem involves the classes C_i and $\bar{C}_i = \{C_1, \dots, C_{i-1}, C_{i+1}, \dots, C_M\}$. Therefore, M conditional probabilities $\hat{P}(C_i|u(t)), i = 1, 2, \dots, M$ are estimated. A new input $u(t)$ is assigned to the class C_k such that

$$k = \underset{i \in \{1,2,\dots,M\}}{\arg\max} \ \hat{P}(C_i|u(t))$$

Note that the sum of the posterior probabilities $\hat{P}(C_i|u(t))$ is not 1 as multiple two-class classification problems are being considered.

Multiclass logistic regression

the posterior probabilities are modelled using the *normalized exponential* (or *softmax function*)

$$P(C_i|u(t)) = \frac{e^{z_i(t)}}{e^{z_1(t)} + e^{z_2(t)} + \cdots + e^{z_M(t)}}, \quad i = 1, 2, \dots, M$$

where

$$z_i(t) = \varphi^T(t)\theta_i, \quad i = 1, 2, \dots, M$$

Note that $\sum_{i=1}^{M} P(C_i|u(t)) = 1$

The input space $\mathcal{U} = \mathbb{R}^r$ can thus be devided into M regions defined by a set of M hyperplanes. Each hyperplane represents the linear decision boundary between two classes. More precisely, the hyperplane separating classes k and j is described by the equation

$$(z_k(t) - z_j(t)) = 0 \quad \text{that is} \quad \varphi^T(t)(\theta_k - \theta_j) = 0$$

An alternative consists in selecting a single class as the baseline. If we select the M-th class the posterior probabilities are given by:

$$P(C_i|u(t)) = \frac{e^{z_i(t)}}{1 + e^{z_1(t)} + e^{z_2(t)} + \cdots + e^{z_{M-1}(t)}}, \quad i = 1, 2, \dots, M-1$$

and

$$P(C_M|u(t)) = \frac{1}{1 + e^{z_1(t)} + e^{z_2(t)} + \cdots + e^{z_{M-1}(t)}}$$

Both alternatives lead to the same results. For both metods, the parameter vectors $\theta_i, \theta_2, \dots$ are estimated by using maximum likelihood and Newton's method (or gradient descent).

11.2.3 Dealing with nonlinear boundaries

By means of a nonlinear transformation in the input space

$$u(t) \in \mathcal{U} \to \bar{u}(t) = g(u(t)) \in \bar{\mathcal{U}}$$

a nonlinear boundary in $\mathcal{U}$ can be mapped into a linear one in $\bar{\mathcal{U}}$. The logistic regression is then applied to the transformed inputs.

- In general, the dimension of $\bar{\mathcal{U}}$ is greater than that of the original input space $\mathcal{U}$ and the complexity of the model increases
- Nonlinear transformations cannot remove class overlap. They can increase the level of overlap or create overlap where none existed in $\mathcal{U}$

11.3 Linear discriminant analysis

The posteior probabilities are estimated through the Bayes theorem:

$$P(C_i|u(t)) = \frac{P(u(t)|C_i)P(C_i)}{P(u(t))} = \frac{P(u(t)|C_i)P(C_i)}{\sum_{k=1}^{M} P(u(t)|C_k)p(C_k)}$$

if $u(t)$ is a continuous random variable, $P(u(t)|C_i)$ is replaced with the conditional pdf $f_u(u|C_i)$. In fact, $f_u(u|C_i)du$ is the probability that $u(t) \in [u, u+du]$.
How does it work?

1. Assume a specific probability density function for $P(u(t)|C_i), i = 1, 2, \ldots, M$ and estimate the parameters characterizing such distribution
2. Estimate the prior probabilities $P(C_1), P(C_2), \ldots, P(C_M)$.
3. Compute estimates of the posterior probabilities $P(C_i|u(t)), i = 1, 2, \ldots, M$
4. Assign a new input $u(t)$ to the class C_k for which $\hat{P}(C_i|u(t)), i = 1, 2, \ldots, M$ is largest

11.3.1 Gaussian class densities with commmon covariance matrix

Two class classification problem

$$f_u(u|C_i) = \frac{1}{\sqrt{(2\pi)^r det(\Sigma)}} \exp\left(-\frac{(u(t)-\mu_i)^T\Sigma^{-1}(u(t)-\mu_i)}{2}\right), \quad i = 1, 2$$

It follows that

$$P(C_1|u(t)) = \frac{f_u(u|C_1)P(C_1)}{f_u(u|C_1)P(C_1) + f_u(u|C_2)P(C_2)} = \frac{1}{1+e^{-z(t)}}$$

where

$$z(t) = u^T(t)\Sigma^{-1}(\mu_1 - \mu_2) + \frac{1}{2}\mu_2^T\Sigma^{-1}\mu_2 - \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 + \log\frac{P(C_1)}{P(C_2)} = \varphi^T(t)\theta$$

with

$$\varphi(t) = \begin{bmatrix} 1 & u_1(t) & u_2(t) & \cdots & u_r(t) \end{bmatrix}^T, \quad \theta = \begin{bmatrix} \beta_0 & \beta_1 & \cdots & \beta_r \end{bmatrix}^T = \begin{bmatrix} \beta_0 \beta^T \end{bmatrix}^T$$

and

$$\beta_0 = \frac{1}{2}\mu_2^T\Sigma^{-1}\mu_2 - \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 + \log\frac{P(C_1)}{P(C_2)}, \quad \beta = \Sigma^{-1}(\mu_1 - \mu_2)$$

As for the logistic regression, the decision boundary is given by the hyperplane

$$z(t) = 0 \quad \text{that is} \quad \varphi^T(t)\theta = 0$$

The estimates required in steps 1 and 2 can be computed as

$$\hat{\mu}_1 = \frac{1}{N_1}\sum_{t=1}^{N} y(t)u(t) \qquad \hat{\mu}_1 = \frac{1}{N_2}\sum_{t=1}^{N}(1-y(t))u(t)$$

$$\hat{\Sigma} = \frac{1}{N}\sum_{t=1}^{N} y(t)(u(t)-\hat{\mu}_1)(u(t)-\hat{\mu}_1)^T + \frac{1}{N}\sum_{t=1}^{N}(1-y(t))(u(t)-\hat{\mu}_2)(u(t)-\hat{\mu}_2)^T$$

$$\hat{P}(C_1) = \frac{N_1}{N_1+N_2}, \quad \hat{P}(C_2) = 1 - \hat{P}(C_1)$$

where N_1 is the number of inputs belonging to class C_1 and $N_2 = N - N_1$ is the number of inputs belonging to class C_2. A new input $u(t)$ is assigned to class C_1 if $\hat{P}(C_1|u(t)) > 0.5$ and to class C_2 otherwise. As for logistic regression, a different decision critetrion can be used.

Multiclass problem

$$f_u(u|C_i) = \frac{1}{\sqrt{(2\pi)^r det(\Sigma)}} \exp\left(-\frac{(u(t)-\mu_i)^T\Sigma^{-1}(u(t)-\mu_i)}{2}\right), \quad i = 1,2,\dots,M$$

It follows that:

$$P(C_i|u(t)) = \frac{f_u(u|C_i)P(C_i)}{\sum_{k=1}^M f_u(u|C_k)P(C_k)} = \frac{e^{z_i(t)}}{\sum_{k=1}^M e^{z_k(t)}}$$

where

$$z_i(t) = u^T(t)\Sigma^{-1}\mu_i - \frac{1}{2}\mu_i^T\Sigma^{-1}\mu_i + \log P(C_i) = u^T(t)\beta + \beta_0 = \varphi^T(t)\theta_i$$

The input space $\mathcal{U} = \mathbb{R}^r$ can thus be divided into M regions defined by a set of M hyperplanes. Each hyperplane represents the linear decisiion boundary between two classes. More precisely, the hyperplane separating classes k and j is described by the equation

$$(z_k(t) - z_j(t)) = 0 \quad \text{that is} \quad \varphi^T(t)(\theta_k - \theta_j) = 0$$

The required estimates are computed as follows

$$\hat{\mu}_i = \frac{1}{N_i}\sum_{t:y(t)=y^i} u(t) \qquad i = 1,2,\dots,M$$

$$\hat{\Sigma} = \sum_{i=1}^M \frac{1}{N}\sum_{t:y(t)=y^i}(u(t)-\hat{\mu}_i)(u(t)-\hat{\mu}_i)^T$$

$$\hat{P}(C_i) = \frac{N_i}{N}, \qquad i = 1,2,\dots,M$$

Where N_i is the number of inputs belonging to class C_i
A new input $u(t)$ is assigned to the class C_k such that

$$k = \underset{i\in\{1,2,\dots,M\}}{\arg\max}\ \hat{P}(C_i|u(t))$$

11.3.2 Gaussian class denisities with different covariance matrices

$$f_u(u|C_i) = \frac{1}{\sqrt{(2\pi)^r det(\Sigma_i)}} \exp\left(-\frac{(u(t)-\mu_i)^T\Sigma_i^{-1}(u(t)-\mu_i)}{2}\right), \quad i = 1,2,\dots,M$$

$$P(C_i|u(t)) = \frac{f_u(u|C_i)P(C_i)}{\sum_{k=1}^M f_u(u|C_k)P(C_k)} = \frac{e^{z_i(t)}}{\sum_{k=1}^M e^{z_k(t)}}$$

where

$$z_i(t) = -\frac{1}{2}\log det(\Sigma_i) - \frac{1}{2}(u(t)-\mu_i)^T\Sigma_i^{-1}(u(t)-\mu_i) + \log P(C_i) = \varphi^T(t)\theta_i$$

In this case, the input space $\mathcal{U}$ can be divided into M regions defined by M quadratic decision boundaries. The quadratic boundary separatingk classes k and j is described by the equation

$$(z_k(t) - z_j(t)) = 0 \quad \text{that is} \quad \varphi^T(t)(\theta_k - \theta_j) = 0$$

For this reason in this case the approach is called *quadratic discriminant analysis*

Chapter 12

Classification: deterministic models

The decision boundaries are explicitly modeled. This involves the construction of a discriminant function that directly assigns each input $u(t)$ to a specific class

Assumptions:

1. two-class classification problems
2. Linear decision boundaries

Because of 2, the discriminant function $z(t)$ is such that

$$z(t) = \varphi^T(t)\theta = u^T(t)\beta + \beta_0$$

where $u(t) = \begin{bmatrix} u_1(t) & u_2(t) & \cdots & u_r(t) \end{bmatrix}^T$ and $\beta = \begin{bmatrix} \beta_1 & \beta_2 & \cdots & \beta_r \end{bmatrix}^T$. For the output $y(t)$ we will use

$$y(t) = \begin{cases} 1 & \text{if } u(t) \in C_1 \quad (z(t) > 0) \\ -1 & \text{if } u(t) \in C_2 \quad (z(t) < 0) \end{cases}$$

The equation $z(t) = 0$, i.e.

$$u^T(t)\beta + \beta_0 = 0$$

defines a hyperplane $\mathcal{H} \in \mathcal{U} = \mathbb{R}^r$ $u_A, u_B \in \mathcal{H} \implies \beta^T(u_A - u_B) = 0$ imples that β is orthogonal to $\mathcal{H}$

$$ui \in \mathcal{H} \implies u^T\beta + \beta_0 = 0$$

so that

$$\frac{\beta^T u}{\|\beta\|} = -\frac{\beta_0}{\|\beta\|}$$

is the perpendicular distance of $\mathcal{H}$ from the origin

$$u \in \mathcal{U} \implies z(t) = \beta^T u + \beta_0$$

so that

$$\frac{z(t)}{\|\beta\|}$$

is the signed distance of $u(t)$ to $\mathcal{H}$ finally, note that, due to the classification rule

$$y(t)\frac{z(t)}{\|\beta\|} \geq 0$$

is the distance (unsigned) of $u(t)$ to the hyperplane $\mathcal{H}$ $(y(t)z(t) \geq 0 \forall t)$

12.0.1 Separating hyperplanes

A dateset whose classes can be separated exactly by linear decision boundaries are said to be linearly separable. These linear decision boundaries are called separating hyperplanes. In general, if the dataset is linearly separable, there exists an infinite number of separating hyperplanes. Once a separating hyperplane $z(t) = 0$ has been found, a new input $u(t)$ is classified as follows:

$$u(t) \in \begin{cases} C_1 & \text{if } z(t) > 0 \\ C_2 & \text{if } z(t) < 0 \end{cases} \implies y(t) = \begin{cases} 1 & \text{if } z(t) > 0 \\ -1 & \text{if } z(t) < 0 \end{cases}$$

12.0.2 the maximum margin classifier

Margin: the smallest distance from the observations to the separating hyperplane
Optimal (maximum margin) hyperplane: it is the separating hyperplane for which the margin is largest. It is found by solving the optimization problem

$$\arg\max_{\beta_0,\beta} \left\{ \min_t \left[y(t) \frac{z(t)}{\|\beta\|} \right] \right\} = \arg\max_{\beta_0,\beta} \left\{ \min_t \left[y(t) \frac{\beta_0 + u^T(t)\beta}{\|\beta\|} \right] \right\}$$

that, in turn, is equivalent to

$$\max_{\beta_0,\beta} D \text{ subject to } y(t)\frac{z(t)}{\|\beta\|} \geq D, \quad t = 1, 2, \ldots, N$$

The optimal hyperplane can be seen as the mi-line of the widest "slab" that can be inserted between two classes. Although the identification of the maximum margin requires all the available data, the optimal hyperplane depends directly on only a small subset of the data points, known as *support vectors*.

If we make the rescaling $\beta \to k\beta$ and $\beta_0 \to k\beta_0$, the distance from any point $u(t)$ to the decision boundary is unchanged, therefore, we can arbitrarily set $\|\beta\| = 1/D$. As a consequence, the optimization problem is equivalent to

$$\max_{\beta_0,\beta} D \text{ subject to } y(t)z(t) \geq 1, \quad t = 1, 2, \ldots, N$$

that is

$$\max_{\beta_0,\beta} \|\beta\|^2 \text{ subject to } y(t)z(t) \geq 1, \quad t = 1, 2, \ldots, N$$

This is a convex optimization problem with N linear inequality constraints. The constraints define an empty slab around the linear decision boundary of thickness $2D = 2/\|\beta\|$

12.1 Constrained optimization problems: the Lagrange multipliers

Consider the following optimization problem with equality constraint:

$$\min_x f(x) \quad \text{subject to} \quad g(x) = 0$$

where $x \in \mathbb{R}^p$. The constraint $g(x) = 0$ represents a $(p-1)$-dimensional hypersurface $\mathcal{S} \in \mathbb{R}^p$. We denote the gradients of $f(x)$ and $g(x)$ as $\nabla f(x), \nabla g(x)$
Since

1. at any pont $x \in \mathcal{S}$ the gradient $\nabla g(x)$ is orthogonal to $\mathcal{S}$
2. for a point $x^* \in \mathcal{S}$ that minimizes $f(x), \nabla f(x^*)$ is orthogonal to $\mathcal{S}$[1]

the gradient $\nabla f(x^*)$ and $\nabla g(x^*)$ are parallel or anti-parallel. Then, ther exists a scalar λ called *Lagrange multiplier* such that

$$\nabla f(x^*) = \lambda \nabla g(x^*)$$

---

[1] this is due to the fact that if the gradient were not orthogonal to S by moving in the direction given by the opposite of the gradient projected onto S we would reach a lower point for $f(x)$ belonging to S

We introduce the Lagrangian function

$$L(x, \lambda) = f(x) - \lambda g(x)$$

The original problem can be reformulated as

$$\min_x L(x, \lambda)$$

in fact:

$$\frac{\partial L(x,\lambda)}{\partial x} = 0 \implies \nabla f(x) - \lambda \nabla g(x) = 0$$

$$\frac{\partial L(x,\lambda)}{\partial \lambda} = 0 \implies g(x) = 0$$

The minimization problem with N_e equality constraints

$$\min_x f(x) \quad \text{subject to} \quad g_j(x) = 0, \quad j = 1, 2, \ldots, N_e$$

can be reformulated as

$$\min_x L(x, \lambda)$$

where $L(x, \lambda)$ is the Lagrangian function

$$L(x, \lambda) = f(x) - \sum_{j=1}^{N_e} \lambda_j g_j(x)$$

and $\lambda_1, \lambda_2, \ldots, \lambda_{N_e}$ are Lagrange multipliers.

Consider the follwing optimization problem with inequaility constraints:

$$\min_x f(x) \quad \text{subject to} \quad g(x) \geq 0$$

where $x \in \mathbb{R}^p$. Again the solution will be based on a Lagrange multiplier λ and the associated Lagrangian funciton. Let x^* be a solution of the problem. Two possibilties exist:

1. $g(x^*) > 0$: In this case the constraint is *inactive* and the solution is found by setting $\nabla f(x) = 0$ so that $g(x)$ plays no role $\implies \lambda = 0$
2. $g(x^*) = 0$: In this case the constraint is *active* so that $\lambda \neq 0$. Moreover, the descending direction $-\nabla f(x^*)$ must be oriented away from the region $g(x) > 0$ otherwise $f(x^*)$ does not correspond to a minimum. Consequently, we have $\nabla f(x^*) = \lambda \nabla g(x^*)$ for some $\lambda > 0$

Note that both the above cases lead to $\lambda g(x) = 0$ We also have that $\lambda \geq 0$. The original problem can be reformulated as follows:

$$\min_x L(x, \lambda) = f(x) - \lambda g(x) \quad \text{subject to} \quad \begin{cases} g(x) \geq 0 \\ \lambda \geq 0 \\ \lambda g(x) = 0 \end{cases}$$

The above constraints are known as the Karush-Kuhn-Tucker (KKT) conditions.

- Note that $L(x, \lambda)$ has to be minimized wrt x and maximized wrt λ to prevent $L(x, \lambda) \to -\infty$

Consider now the more general optimization problem with N_i inequality constraints

$$\min_x f(x) \quad \text{subjet to} \quad g_j(x) \geq 0 \quad j = 1, 2, \ldots, N_i$$

Define the Lagrangian function

$$L(x, \lambda) = f(x) - \sum_{j=1}^{N_i} \lambda_j g_j(x)$$

where $\lambda = \begin{bmatrix} \lambda_1 & \lambda_2 & \ldots & \lambda_{N_i} \end{bmatrix}^T$ The problem can be reformulated as

$$\min_x L(x, \lambda) = f(x) - \lambda g(x) \quad \text{subject to} \quad \begin{cases} g_j(x) \geq 0 \quad j = 1, \ldots, N_i \\ \lambda_j \geq 0 \quad j = 1, \ldots, N_i \\ \lambda_j g_j(x) = 0 \quad j = 1, \ldots, N_i \end{cases}$$

The number of KKT conditions is thus $3N_i$. The Lagrangian function $L(x, \lambda)$ has to be minimized wrt x and maximized wrt λ

Finding the maximum margin classifier by using the Lagrange multipliers

The problem

$$\min_{\beta_0,\beta} \frac{1}{2}\|\beta\|^2 \quad \text{subject to} \quad y(t)z(t) \geq 1, \quad t = 1, 2, \ldots, N$$

becomes

$$\min_{\beta_0,\beta} J(\beta_0, \beta, \lambda) = \frac{1}{2}\|\beta\|^2 - \sum_{t=1}^{N} \lambda_t (y(t)z(t) - 1) = \frac{1}{2}\|\beta\|^2 - \sum_{t=1}^{N} \lambda_t [y(t)(\beta_0 + u^T(t)\beta) - 1]$$

subject to the constraints

$$\begin{aligned} y(t)z(t) - 1 &\geq 0 \\ \lambda_t &\geq 0 \qquad t = 1, 2, \ldots, N \\ \lambda_t(y(t)z(t) - 1) &= 0 \end{aligned}$$

- the points for which $\lambda_t = 0$ do not contribute to the estimated model $\hat{\theta} = \begin{bmatrix} \hat{\beta}_o & \hat{\beta}^T \end{bmatrix}^T$. The remaining points, for which $y(t)z(t) = 1$, are the *support vectors* and lie on the edges of the margin. This means that the estimated function $\hat{z}(t)$ will depend only on a small part of the training data.

Setting to zero the derivatives of J wrt β, β_0 we get

$$\frac{\partial J}{\partial \beta} = 0 \implies \beta = \sum_{t=1}^{N} \lambda_t y(t) u(t) \tag{12.1}$$

$$\frac{\partial J}{\partial \beta_0} = 0 \implies 0 = \sum_{t=1}^{N} \lambda_t y(t) \tag{12.2}$$

Substituting the above relations in $J(\beta_0, \beta, \lambda)$ we obtain the *dual objective function*

$$J(\lambda) = \sum_{t=1}^{N} \lambda_t - \sum_{t=1}^{N} \sum_{k=1}^{N} \lambda_t \lambda_k y(t) y(k) u^T(t) u(k)$$

to be maximized wrt λ only subject to

$$\lambda_t \geq 0, \quad t = 1, 2, \ldots, N$$

$$\sum_{t=1}^{N} \lambda_t y(t) = 0$$

This is a simpler convex optimization problem, involving a quadratic function of λ.

By replacing 12.1 in $z(t) = \beta_0 + u^T(t)\beta$ we get

$$z(t) = \beta_0 + u^T(t) \sum_{k=1}^{N} \lambda_k y(k) u(k) = \beta_0 + \sum_{k=1}^{N} \lambda_k y(k) u^T(t) u(k)$$

Since only the support vectors have $\lambda_k \neq 0$ we have

$$z(t) = \beta_0 + \sum_{k \in \mathcal{S}_v} \lambda_k y(k) u^T(t) u(k)$$

where $\mathcal{S}_v$ is the set of indices of the support vectors.

Once the estimate $\hat{\theta}, \hat{\lambda}$ has been determined, a new input $u(t)$ is classified as in slide 3 by using

$$\hat{z}(t) = \varphi^T(t)\hat{\theta} = \hat{\beta}_0 + \sum_{k \in \mathcal{S}} \hat{\lambda}_k y(k) u^T(t) u(k)$$

The above relation shows how the obtained solution only depends on a limited number of scalar products $u^T(t)u(k)$

- Although none of the training inputs are misclassified by construction, this will not necessarily be the case for future unseen data points. Hopefully, a large margin on the training data will lead to a good separation on the validation (test) data
- The distance of a point from the hyperplane can be seen as a measure of the confidence that the observation was correctly classified.
- The maximum margin hyperplane can be extremely sensitive to changes in the available data. Even a change in a single observation could lead to a dramatic change in the optimal hyperplane.
- If the training set is not linearly separable, the optimization problem has no feasible solution

12.2 Support vector machine

Support vector (soft margin) classifier: the previous approach is modified so that some of the training points are allowed to be on the "wrong" side of the margin or even on the wrong side of the hyperplane but with a penalty that increases with the distance from the hyperplane. In this way, the non separable case can also be tackled.

To solve this problem we introduce a set of *slack variables* $\xi_1, \xi_2, \dots, \xi_N$ with $\xi_t \geq 0, \forall t$ More precisely, each input $u(t)$ is associated with a slack variable as follows:

$$\xi_t = \begin{cases} 0 & \text{for data points that are on the correct side of the margin or on the correct edge of the margin} \\ |y(t) - z(t)| & \text{for other points} \end{cases}$$

It follows that

- $\xi_t = 0 \implies u(t)$ is correctly classified with distance $\geq D$
- $0 < \xi_t < 1 \implies u(t)$ is correctly classified with distanec $< D$
- $\xi_t = 1 \implies u(t)$ is on the decision boundary
- $\xi_t > 1 \implies u(t)$ is misclassified

These considerations can be derived considering that the distance of a point from the hyperplane is given by

$$d = \frac{y(t)z(t)}{\|\beta\|}$$

which by setting $\|\beta\| = 1/D$ becomes

$$d = y(t)z(t)D$$

and by remembering that

$$y(t)^2 = 1$$

The optimization problem becomes

$$\max_{\beta_0, \beta, \xi} D \quad \text{subject to} \begin{cases} y(t)\dfrac{z(t)}{\|\beta\|} \geq D(1 - \xi) \\ \xi_t \geq 0 \\ \displaystyle\sum_{t=1}^{N} \xi_t \leq K \end{cases} \qquad t = 1, 2, \dots, N$$

where $\xi = \begin{bmatrix} \xi_1 & \xi_2 & \cdots & \xi_N \end{bmatrix}^T$ and $K > 0$ is a tuning parameter that controls the trade-off between the slack variable penalty and the margin. It is an upper bound on the number of misclassified points. As K increases, the margin increases.

The problem can be rewritten as

$$\max_{\beta_0,\beta,\xi} D \quad \text{subject to} \quad \begin{cases} y(t)z(t) \geq (1-\xi_t) \\ \xi_t \geq 0 \\ \sum_{t=1}^{N} \xi_t \leq K \end{cases} \qquad t = 1, 2, \ldots, N$$

by rescaling β_0 by a factor η such that $\|\beta\| = 1/D$ Computationally, it is convenient to exploit the equivalent form

$$\min_{\{\beta_0,\beta,\xi\}} \left\{ \frac{1}{2}\|\beta\|^2 + C\sum_{t=1}^{N} \xi_t \right\} \quad \text{subject to} \quad \begin{cases} y(t)z(t) \geq (1-\xi_t) \\ \xi_t \geq 0 \end{cases} \qquad t = 1, 2, \ldots, N$$

Where the tuning parameter $C > 0$ plays the role of $1/K$ in the previous form (as C increases the margin narrows). The scalar $1/2$ has been included for later convenience. Once again, we have a convex optimization problem that can be solved by using the method of Lagrange multipliers. By introducing the vectors $\lambda = \begin{bmatrix} \lambda_1 & \lambda_2 & \cdots & \lambda_N \end{bmatrix}^T$ and $\mu = \begin{bmatrix} \mu_1 & \mu_2 & \cdots & \mu_N \end{bmatrix}^T$, the above problem can be seen as the optimization of the loss function

$$J(\beta_0, \beta, \xi, \lambda, \mu) = \frac{1}{2}\|\beta\|^2 + C\sum_{t=1}^{N} \xi_t - \sum_{t=1}^{N} \lambda_t(y(t)z(t) - 1 + \xi_t) - \sum_{t=1}^{N} \mu_t \xi_t$$

subject to the constraints (KKT conditinos):

$$\begin{aligned} y(t)z(t) - 1 + \xi_t &\geq 0 \\ \lambda_t &\geq 0 \\ \lambda_t \left(y(t)z(t) - 1 + \xi_t\right) &= 0 \\ \xi_t &\geq 0 \\ \mu_t &\geq 0 \\ \mu_t \xi_t &= 0 \end{aligned}$$

As before, the points for which $\lambda_t = 0$ do not contribute to the estimated model $\hat{\theta}$. The remaining points, for which $y(t)z(t) = 1 - \xi_t$ are the support vectors. They may have $\xi_t = 0$ (lie on the edges of the margin), $0 < \xi_t \leq 1$ (lie inside the margin) or $\xi_t > 1$ (misclassified)

Setting to zero the derivatives of J wrt β, β_0 and $\xi_t (t = 1, \ldots, N)$ we get

$$\frac{\partial J}{\partial \beta} = 0 \implies \beta = \sum_{t=1}^{N} \lambda_t y(t) u(t) \tag{12.3}$$

$$\frac{\partial J}{\partial \beta_0} = 0 \implies 0 = \sum_{t=1}^{N} \lambda_t y(t) \tag{12.4}$$

$$\frac{\partial J}{\partial \xi_t} = 0 \implies \lambda_t = C - \mu_t, \quad t = 1, \ldots, N \tag{12.5}$$

Subsituting the above relations in $J(\beta_0, \beta, \xi, \lambda, \mu)$ we obtain the *dual objective function*

$$J(\lambda) = \sum_{t=1}^{N} \lambda_t - \frac{1}{2}\sum_{t=1}^{N}\sum_{k=1}^{N} \lambda_t \lambda_k y(t) y(k) u^T(t) u(k)$$

to be maximized wrt λ subject to

$$0 \leq \lambda_t \leq C, \quad t = 1, 2, \ldots, N$$

$$\sum_{t=1}^{N} \lambda_t y(t) = 0$$

By looking at 12.3, it is clear that the solution of the optimization problem depends on input samples only through the scalar products

$$u^T(t)u(k), \quad t, k = 1, 2, \ldots, N$$

By replacing 12.3 in $z(t) = \beta_0 + u^T(t)\beta$ we get

$$z(t) = \beta_0 + u^T(t) \sum_{k=1}^{N} \lambda_k y(k) u(z) = \beta_0 + \sum_{k=1}^{N} \lambda_k y(k) u^T(t) u(k)$$

Since only the support vectors have $\lambda_k \neq 0$ we have

$$z(t) = \beta_0 + \sum_{k \in \mathcal{S}_v} \lambda_k y(k) u^T(t) u(k)$$

Where $\mathcal{S}_v$ is the set of indices of the support vectors. Once that $\hat{\lambda}$ and $\hat{\beta}_0$ have been computed, a new input $u(t)$ is classified by using

$$\hat{z}(t) = \varphi^T(t)\hat{\theta} = \hat{\beta}_0 + \sum_{k \in \mathcal{S}_v} \hat{\lambda}_k y(k) u^T(t) u(k)$$

Again, all we need is scalar products.

12.2.1 Dealing with nonlinear boundaries: the kernel trick

Nonlinear decision boundaries can be tackled by introducing a generalization of the scalar product

$$\Psi(u(t), u(k))$$

where $\Psi(x, y) = \Psi(y, x)$ is a symmetric function called kernel, that involves nonlinear functions of the input samples. The scalar product $\Psi(x, y) = x^T y$ is a linear kernel.

By using kernels, we can fit a support vector classifier in a higher-dimensional space involving nonlinear functions rather than in the original input space. The risulting classifier is know as *support vector machine.*

How does it work?

1. Choose a kernel function $\Psi(u(t), u(k))$.
2. Solve the optimization problem

$$\max_{\lambda} J(\lambda) = \sum_{t=1}^{N} \lambda_t - \frac{1}{2} \sum_{t=1}^{N} \sum_{k=1}^{N} \lambda_t \lambda_k y(t) y(k) \Psi(u(t), u(k))$$

   subject to

$$0 \leq \lambda_t \leq C, \quad t = 1, 2, \ldots, N$$
$$\sum_{t=1}^{N} \lambda_t y(t) = 0$$

   and find the estimates $\hat{\lambda}$. Then, find the estimate $\hat{\beta}_0$

3. Classify the new inputs by using

$$\hat{z}(t) = \varphi^T(t)\hat{\theta} = \hat{\beta}_0 + \sum_{k \in \mathcal{S}} \hat{\lambda}_k y(k) \Psi(u(t), u(k))$$

Two popular choices for the kernel function Ψ are:

Polynomial kernels

$$\Psi(u(t), u(k)) = (1 + u^T(t) u(k))^q$$

Radial basis (gaussian) kernels

$$\Psi(u(t), u(k)) = \exp(-\gamma \|u(t) - u(k)\|^2)$$

where $\gamma > 0$ is a hyperparameter.

By using kernels, we can determine complex nonlinear decision boundaries without explicitly working in the enlarged feature space. The enlarged feature space can even be implicit and infinite-dimensional. This is the case for the radial basis kernels.

12.2.2 Multiclass problems

One-vs-all approach

Consists in solving M two-class SVM problems. The generic i-th problem involves the classes C_i and $\bar{C}_i = \{C_1, \ldots, C_{i-1}, C_{i+1}, \ldots, C_M\}$. Therefore, we get M different estimates for λ and β_0 and, consequently, M discriminant functions $\hat{z}^1(t), \hat{z}^2(t), \ldots, \hat{z}^M(t)$. For any new input $u(t)$, the values $\hat{z}^i(u(t)), i = 1, 2, \ldots, M$ are computed in order to take the final decision.

This method does not lead to an optimal solution and inconsistent results could be obtained, that is, an input could be assigned to multiple classes simultaneously. For this reason, the most common approach is to assign a new input u(t) to the class k such that

$$k = \underset{i \in \{1,2,\ldots,M\}}{\arg\max} \ \hat{z}^i(t)$$

However, this is a heuristic approach that suffers form the problem that the M obtained classifiers were trained on different tasks.

One-vs-one approach

Consists in solving $M(M-1)/2$ two-class SVM problems. The generic binary problem involves the classes C_i and C_k, so that only part of the training data are used. For any new input $u(t)$, the values $\hat{z}^i(u(t)), i = 1, 2, ..., M(M-1)/2$ are computed. The new input is then classified according to the class that got the highest number of "votes".

- Again, the obtained solution is not optimal and the approach can lead to ambiguities in the resulting classification
- For large M, the required computational effort significantly increases.

Chapter 13

Regularization

Consists in adding a penalty term to the loss function

$$V(\theta) = J(\theta) + \lambda g(\theta)$$

where $\lambda > 0$ is a tuning parameter. The solution of the learning from data problem becomes

$$\hat{\theta} = \arg\min_{\theta \in \mathcal{M}_p(\theta)} V(\theta)$$

This technique is used both in regression and classification problems. One of the most common form is *quadratic regularization*

$$V(\theta) = J(\theta) + \lambda \|\theta\|_2^2 = J(\theta) + \lambda \theta^T \theta$$

Why regularization?

- To counteract the effects of overfitting
- To make the optimization problem better conditined

Note: the tuning parameter λ must be determined separately as for the model complexity.

13.1 Regularized least squares: the Ridge regression

Consider the static model

$$y(t) = \varphi^T(t)\theta + \varepsilon(t), \quad t = 1, 2, \ldots, N$$

and the associated quadratic regularized LS problem

$$\min V(\theta): \quad v(\theta) = \sum_{t=1}^{N} \varepsilon^2(t) + \lambda \theta^T \theta = \|Y - \Phi\theta\|^2 + \lambda \|\theta\|_2^2$$

This estimation method is called Ridge regression. By setting the gradient of $v(\theta)$ to zero it is easy to find the normal equations

$$(\Phi^T \Phi + \lambda I_p)\theta = \Phi^T Y$$

where p is the complexity of the model. The regularized LS estimator is thus given by

$$\hat{\theta} = (\Phi^T \Phi + \lambda I_p)^{-1} \Phi^T Y$$

the penalty term $\lambda \|\theta\|_2^2$ is also called a *shrinkage penalty* in statistics because it has the effect of shrinking the estimated parameters towards zero.

Assume now the existence of the true model

$$y(t) = \varphi^T(t)\theta^* + w(t) \quad t = 1, 2, \ldots, N$$

By analyzing the statistical properties of the Ridge estimator we get

$$\begin{aligned} E[\hat{\theta}] &= E[(\Phi^T\Phi + \lambda I_p)^{-1}\Phi^T Y] \qquad\qquad Y = \Phi\theta^* + w \\ E[\hat{\theta}] &= E[(\Phi^T\Phi + \lambda I_p)^{-1}\Phi^T(\Phi\theta^* + w)] = \\ &= E[(\Phi^T\Phi + \lambda I_p)^{-1}\Phi^T\Phi\theta^*] + E[(\Phi^T\Phi + \lambda I_p)^{-1}\Phi^T w] = \\ &= (\Phi^T\Phi + \lambda I_p)^{-1}\Phi^T\Phi\theta^* \neq \theta^* \end{aligned}$$

and

$$\mathrm{cov}(\hat{\theta}) = E\left[(\hat{\theta} - \theta^*)(\hat{\theta} - \theta^*)^T\right]$$

by considering that

$$\begin{aligned} \hat{\theta} &= (\Phi^T\Phi + \lambda I_p)^{-1}\Phi^T Y \\ &= (\Phi^T\Phi + \lambda I_p)^{-1}\Phi^T\Phi(\Phi^T\Phi)^{-1}\Phi^T Y \\ &= (\Phi^T\Phi + \lambda I_p)^{-1}\Phi^T\Phi\hat{\theta}_{LS} \end{aligned}$$

we can compute the covariance in the following way:

$$\begin{aligned} \mathrm{cov}(\hat{\theta}) &= (\Phi^T\Phi + \lambda I_p)^{-1}\Phi^T\Phi\,\mathrm{cov}(\hat{\theta}_{LS})\Phi^T\Phi(\Phi^T\Phi + \lambda I_p)^{-1} \\ &= (\Phi^T\Phi + \lambda I_p)^{-1}\Phi^T\Phi\sigma_w^2(\Phi^T\Phi)^{-1}\Phi^T\Phi(\Phi^T\Phi + \lambda I_p)^{-1} \\ &= \sigma_w^2(\Phi^T\Phi + \lambda I_p)^{-1}\Phi^T\Phi(\Phi^T\Phi + \lambda I_p)^{-1} \end{aligned}$$

It can also be proven that

$$\mathrm{cov}(\hat{\theta}) = \sigma_w^2(\Phi^T\Phi + \lambda I_p)^{-1}\Phi^T\Phi(\Phi^T\Phi + \lambda I_p)^{-1} < \sigma_w^2(\Phi^T\Phi)^{-1} = \mathrm{cov}(\hat{\theta}_{LS})$$

- The Ridge estimator $\hat{\theta}$ is biased and $\mathrm{cov}(\hat{\theta}) < \mathrm{cov}(\hat{\theta}_{LS})$
- as λ increases, the shrinkage of the estimated cocefficients leads to a reduction in the variance of the estimates at the expense of an increase in bias $\implies$ λ is a hyperparameter that can manage the bias-variance tradeoff
- If $\Phi^T\Phi$ is ill-conditioned (or even rank deficient), the use of λ leads to the better conditioned matrix $\Phi^T\Phi + \lambda I_p$

Regularization can be exploited also for the identification of dynamic models and for classification problems. In general:

- If the model complexity p is high (many parameters) it may not be possible to estimate several of the accurately $\implies$ it is advantageous to pull them towards zero as the ones having the smallest influence on $J(\theta)$ will be affected most by the shrinkage property $\implies$ regularization allows complex models to be trained on small data sets without severe overfitting.
- The problem of minimizing $J(\theta)$ may be ill conditioned, especially when the complexity p is high, in the sense that the Hessian $J''(\theta)$ may be ill-conditioned $\implies$ adding the norm penalty will add λI_p to this matrix so that it becomes better conditioned.
- The choice of λ is a crucial issue as we may think of λ as a knob to control the bias-variance tradeoff (the larger λ the larger the number of parameters that will be close to zero). The best way consists in choosing the values of λ leading to the smallest value of the loss function evaluated on the validation set.

Alternative formulation for Ridge regression

It is possible to show that the problem

$$\arg\min_{\theta\in\mathcal{M}_p(\theta)} V(\theta), \quad v(\theta) = J(\theta) + \lambda\theta^T\theta$$

is equivalent to the problem

$$\arg\min_{\theta\in\mathcal{M}_p(\theta)} J(\theta), \quad \text{subject to} \quad \theta^T\theta \le K$$

- For every λ there is some K such that the solutions $\hat{\theta}$ of the two optimization problems are the same. The two approaches can be related using Lagrange multipliers.
- The parameter K can be seen as a budget for how large the norm of θ can be
- Note that K plays the role of $1/\lambda$. In fact, decreasing K has the effect of shrinking the estimated parameters towards zero.

Look at prof. notes for geometrical interpretation (The solution will lie on the level curve of $J(\theta)$ that intercepts the constraint curve)

Chapter 14

Optimal estimation of random signals

The fundamental theorem of estimation theory

Let $x(t), y(t)$ be mutually correlated stochastic process, where $y(t)$ is observed and $x(t)$ is uknown. The optimal (*minimal variance*) estimate of $x(t)$ based on $y(t)$ is given by the conditional expectation

$$\hat{x}(t) = E[x(t)|y(t)]$$

For any other estimator:

$$\hat{x}'(t) : E[\|x(t) - \hat{x}(t)\|^2] < E[\|x(t) - \hat{x}'(t)\|^2]$$

if $x(t)$ and $y(t)$ are jointly gaussian, then

$$\hat{x}(t) = \mu_x + C_{xy}\Sigma_y^{-1}(y(t) - \mu_y)$$

where $C_{xy} = E[(x(t) - \mu_x)(y(t) - \mu_y)^T]$ and $\Sigma_y = E[(y(t) - \mu_y)(y(t) - \mu_y)^T]$.

Best linear (least mean squares estimator)

$$\hat{x}(t)_{LMS} = \mu_x + C_{xy}\Sigma_y^{-1}(y(t) - \mu_y)$$

The LMS estimator is the best one within the class of linear estimators $\hat{x}(t) = Ay(t) + b$. When $x(t)$ and $y(t)$ are jointly gaussian $\hat{x}_{LMS} \equiv \hat{x}(t)$

Properties of the optimal (linear) estimator

- Ubiasedness: $E[\hat{x}(t)] = E[x(t)]$
- For any other estimator $\hat{x}'(t)$:

  $$E[(x(t) - \hat{x}(t))(x(t) - \hat{x}(t))^T] \leq E[(x(t) - \hat{x}'(t))(x(t) - \hat{x}'(t))^T]$$

- Orthogonality: $E[(x(t) - \hat{x}(t))y^T(t)] = 0 \implies$ [1]$E[(x(t) - \hat{x}(t))\hat{x}^T(t)] = 0$
- Linearity: $z(t) = \alpha x(t) + \beta y(t) \implies \hat{z}(t) = \alpha\hat{x}(t) + \beta\hat{y}(t)$
- Let $y_1(t), y_2(t), \ldots, y_k(t)$ be mutually uncorrelated. It follows that

  $$E[x(t)|y_1(t), y_2(t), \ldots, y_k(t)] = E[x(t)|y_1(t)] + E[x(t)|y_2(t)] + \cdots + E[x(t)|y_k(t)] - (k-1)\mu_x$$

Optimal prediction and filtering:
Define $Y_t = \{y(t), y(t-1), y(t-2), \ldots, y(1)\}$
Optimal prediction: $\hat{x}(t|t-1) = E[x(t)|Y_{t-1}]$
Optimal filtering: $\hat{x}(t|t) = E[x(t)|Y_t]$

---

[1] due to y being a LC of $\hat{x}$

The innovation sequence

Optimal one step ahead prediction of $y(t)$ given $y(t-1), y(t-2), \ldots$:

$$\hat{y}(t|t-1) = E[y(t)|y(t-1)], y(t-2), \ldots, y(1)] = E[y(t)|Y_{t-1}]$$

The corresponding prediction error is called the *innovation*

$$\varepsilon(t) = y(t9)) - \hat{y}(t|t-1)$$

Innovation properties

1. $\varepsilon(t)$ is a linear function of $y(t), y(t-1), y(t-2) \ldots$
2. $E[\varepsilon(t) Y_{t-1}^T] = 0$
3. $\varepsilon(t)$ is a zero mean white process:
$$E[\varepsilon(t)] = 0, \quad E[\varepsilon(t+\tau)\varepsilon(t)] = \sigma_\varepsilon^2 \delta(\tau)$$
4. $\varepsilon(t)$ is the new piece of information in $y(t)$ that was not known at time $t-1$

As a consequence, $\varepsilon(t)$ and $y(t)$ have the same information content

$$\hat{y}(t|t-1) = E[y(t)|\varepsilon(t-1), \varepsilon(t-2), \ldots, \varepsilon(1)] = E[y(t)|\tilde{Y}_{t-1}]$$

where

$$\tilde{Y}_{t-1} = \{\varepsilon(t-1), \varepsilon(t-2), \ldots, \varepsilon(1)\}$$

With reference to the optimal prediction and filtering problems, we have

$$\begin{aligned} \hat{x}(t|t-1) &= E[x(t)|Y_{t-1}] = E[x(t)|\tilde{Y}_{t-1}] \\ \hat{x}(t|t) &= E[x(t)|Y_t] = E[x(t)|\tilde{Y}_t] \end{aligned}$$

14.1 the Kalman filter

14.1.1 Deterministic state space models: the Luenberger observer

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned}$$

Problem: given A, B, C determine an estimate of $x(t)$ from past input and output samples $y(t-1), u(t-1), y(t-2), u(t-2), \ldots$

Luenberger observer

$$\begin{aligned} \hat{x}(t+1) &= A\hat{x}(t) + Bu(t) + K(y(t) - C\hat{x}(t)) \quad \hat{x}(t) = \hat{x}_0 \\ &= (A - KC)\hat{x}(t) + Bu(t) + Ky(t) \end{aligned}$$

$\hat{x}_0$: initial guess for x_0

- (A, C) observable $\implies \lim_{t\to\infty} \hat{x}(t) = x(t)$ with arbitrary dynamics
- (A, C) detectable $\implies \lim_{t\to\infty} \hat{x}(t) = x(t)$ but not with arbitrary dynamics

14.1.2 Stochastic state space models

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) + Gw(t) \qquad && x(0) = x_0 \quad x(t) \in \mathbb{R}^n, u(t) \in \mathbb{R}^r, w(t) \in \mathbb{R}^p \\ y(t) &= Cx(t) + v(t) && y(t) \in \mathbb{R}^m \end{aligned}$$

- $u(t)$ is a deterministic process
- $w(t)$ and $v(t)$ are mutually uncorrelated zero-mean white processes
- x_0 is a random vector, uncorrelated with $w(t)$ and $v(t)$

$$E[w(t+\tau)w(t)^T] = Q\delta(\tau) \quad E[v(t+\tau)v(t)^T] = R\delta(\tau) \quad E[(x_o - \mu_0)(x_o - \mu_0)^T] = P_0$$

Problem 1 (optimal prediction): given $A, B, C, Q, R, \mu_0, P_0$ determine the optimal estimate of $x(t)$ from past input and output samples
Problem 2 (optimal filtering): given $A, B, C, Q, R, \mu_0, P_0$ determine the optimal estimate of $x(t)$ from present and past input adn output samples

14.1.3 Derivation of the Kalman filter

By applying conditional expectation to the state and output equations, we can write the innovation as:

$$\varepsilon(t) = C(x(t) - \hat{x}(t|t-1)) + v(t) = Ce_x(t) + v(t)$$

where $e_x(t)$ is the state prediction error

$$e_x(t) = x(t) - \hat{x}(t|t-1)$$

We define

$$P(t|t-1) = E[e_x(t)e_x^T(t)]$$

And derive the covariance matrix of the innovation as follows:

$$\begin{aligned} E[\varepsilon(t)\varepsilon^T(t)] &= E[(Ce_x(t) + v(t))(Ce_x(t) + v(t))^T] \\ &= E[Ce_x(t)e_x^T(t)] + E[Ce_x(t)v^T(t)] + E[v(t)e_x^T(t)C^T] + E[v(t)v^T(t)] \\ &= CP(t|t-1)C^T + R \end{aligned}$$

To get $\hat{x}(t|t)$ we do the following computations:

$$\begin{aligned} \hat{x}(t|t) &= E[x(t)|\tilde{Y}_t] \\ &= E[x(t)|\varepsilon(t)] + E[x(t)|\tilde{Y}_{T-1}] - E[x(T)] \\ &= E[x(t)] + E[(x(t) - E[x(t)])\varepsilon^T(t)]E[\varepsilon(t)\varepsilon^T(t)]^{-1}\varepsilon(t) + E[x(t)|\tilde{Y}_{T-1}] - E[x(T)] \end{aligned}$$

Let us break down computation

$$\begin{aligned} E[(x(t) - E[x(t)])\varepsilon^T(t)] &= E[x(t)\varepsilon^T(t)] - E[x(t)]E[\varepsilon^T(t)] \\ &= E[x(t)(Ce_x(t) + v(t))^T] \\ &= E[x(t)e_x^T(t)]C^T + E[x(t)v^T(t)] \\ &= E[(e_x(t) + \hat{x}(t|t-1))e_x^T(t)] \\ &= E[e_x(t)e_x^T(t)] + E[\hat{x}(t|t-1)e_x^T(t)] \\ &= P(t|t-1) \end{aligned}$$

Therefore we get

$$\hat{x}(t|t) = P(t|t-1)C^T(CP(t|t-1)C^T + R)^{-1}\varepsilon(t) + \hat{x}(t|t-1)$$

which we rewrite as

$$\hat{x}(t|t) = K(t)\varepsilon(t) + \hat{x}(t|t-1)$$

Where $K(t)$ is the Kalman filter gain. The prediction error is

$$e_x(t+1) = x(t+1) - \hat{x}(t+1|t) = A(x(t) - \hat{x}(t|t)) + Gw(t)$$

and its covariance computes as:

$$\begin{aligned} P(t+1|t) &= E[e_x(t+1)e_x^T(t+1)] \\ &= E[(A(x(t) - \hat{x}(t|t) + Gw(t)))(A(x(t) - \hat{x}(t|t) + Gw(t)))^T] \\ &= AP(t|t)A^T + GQG^T \end{aligned}$$

Where $P(t|t)$ is the covariance matrix of the filtering error

$$x(t) - \hat{x}(t|t) = x(t) - K(t)\varepsilon(t) - \hat{x}(t|t-1)$$

which we can simply rewrite as

$$(I - K(t)C)e_x(t) - K(t)v(t)$$

from which it follows that

$$P(t|t) = (I - K(t)C)P(t|t-1)(I - K(t)C)^T + K(t)RK^T(t)$$

by writing, for the sake of lightening notation,

$$P(t|t-1) = P, \quad K(t) = K$$

we get

$$\begin{aligned} P(t|t) &= P - PC^TK^T - KCP + KCPC^TK^T + KRK^T \\ &= P - PC^TK^T - KCP + K(CPC^T + R)K^T \\ &= P_PC^TK^T - KCP + PCT(CPC^T + R)^{-1}(CPC^T + R)K^T \\ &= P - PC^TK^T - KCP + PC^TK^T \\ &= P(t|t-1) - K(t)CP(t|t-1) \end{aligned}$$

We have:

$$P(t|t) \le P(t|t-1)$$

14.1.4 Kalman filter equations: predictor-corrector form

Start:

$$\hat{x}(0|-1) = \mu_0, P(0|-1) = P_0$$

Correction:

$$K(t) = P(t|t-1)C^T(CP(t|t-1)C^T + R)^{-1} \tag{1}$$

$$\varepsilon(t) = y(t) - \hat{y}(t|t-1) = y(t) - C\hat{x}(t|t-1) \tag{2}$$

$$\hat{x}(t|t) = \hat{x}(t|t-1) + K(t)\varepsilon(t) \tag{3}$$

$$P(t|t) = (I - K(t)C)P(t|t-1) \tag{4}$$

Prediction:

$$\hat{x}(t+1|t) = A\hat{x}(t|t) + Bu(t) \tag{5}$$

$$P(t+1|t) = AP(t|t)A^T + GQG^T \tag{6}$$

Remarks:

- The filter is time-varying even if the system is time-invariant
- $P(t+1|t)$ and $K(t)$ do not depend on the data and can be computed in advance
- The state estimation is not exact even if x_0 is known, because of the presence of $w(t)$
- The measurement (innovation) is not fully reliable because of the presence of $v(t)$
- $K(t)$ is the best tradeoff between the use of the model and the use of the measurements
- To start the algorithm in practice we can set $\hat{x}(0|-1) = 0,\ P(0|-1) = \alpha I_n$, where α is a positive number that reflects the confidence in the initial estimate $\hat{x}(0|-1)$

14.1.5 Kalman predictor form

By replacing $\hat{x}(t|t)$ in $\hat{x}(t+1|t)$ and $P(t|t)$ in $P(t+1|t)$ we get

$\hat{x}(t+1|) = A\hat{x}(t|t-1) + AK(t)\varepsilon(t) + Bu(t)$
$= (A - K_p(t)C)\hat{x}(t|t-1) + Bu(t) + K_p(t)y(t)$

where $K_p(t) = AK(t)$, and

$$P(t+1|t) = AP(t|t-1)A^T - AP(t|t-1)C^T(CP(t|t-1)C^T + R)^{-1}CP(t|t-1)A^T + GQG^T$$

The above equation is called the *difference Riccati equation*

14.1.6 Convergence of the difference Riccati equation

Theorem 1

Let the system be asymptotically stable. Then:

- $P(t+1|t)$ converges to a constant matrix P for every initial positive semidefinite condition

  $$\lim_{t\to\infty} P(t+1|t) = P \quad \forall P_0 >\geq 0$$

  where P is the solution of the *algebraic Riccati equation*

  $$P = APA^T - APC^T(CPC^T + R)^{-1}CPA^T + GQG^T$$

- As a consequence, the Kalmann filter gain $K(t)$ converges:

  $$K = \lim_{t\to\infty} K(t) = PC^T(CPC^T + R)^{-1}$$

- The corresponding Kalmann predictor is asymptotically stable, i.e. all the eigenvalues of $A - K_pC$ are inside the unit disc, where $K_p = AK$

Theorem 2

Let

$$Q = \bar{Q}\bar{Q}^T, \quad \bar{Q} > 0 \quad \text{and} \quad \bar{G} = G\bar{Q}$$

If $(A, \bar{G})$ is stabilizable and (A, C) is detectable, the same results of theorem 1 hold

14.1.7 Time invariant filter (predictor)

If the assumptions of Th. 1 or Th. 2 are satisfied, it is possible to use the steady-state Kalman filter gain K from the beginning so that the difference Riccati equation is avoided $\Longrightarrow$ suboptimal filter. The equations become:

1. Start: solve the algebraic Riccati equation to get P and then K, set $\hat{x}(0|-1)$
2. Correction:

   $$\varepsilon(t) = y(t) - \hat{y}(t|t-1)^s = y(t) - C\hat{x}(t|t-1)^s \tag{1}$$
   $$\hat{x}(t|t)^s = \hat{x}(t|-1)^s + K\varepsilon(t) \tag{2}$$

3. Prediction:

   $$\hat{x}(t+1|t)^s = A\hat{x}(t|t)^s + Bu(t) \tag{3}$$

The suboptimal estimates convege to the optimal ones:

$$\lim_{t\to\infty} \hat{x}(t|t)^s = \hat{x}(t|t), \quad \lim_{t\to\infty} \hat{x}(t+1|t)^s = \hat{x}(t+1|t)$$

One nice property of the Kalman filter is that it can also be used for time-varying systems

$$\begin{cases} x(t+1) = A(t)x(t) + B(t)u(t) + g(t)w(t) \\ y(t) = C(t)x(t) + v(t) \end{cases}$$

This application requires no modification of the Kalman filter equations. However, in this case the invariant version of the filter is not applicable as convergence of P cannot be proven as the terms of the algebraic Riccati equation would be time-varying.