# ML2025

Dima Bykhovsky

April 2, 2025

# Contents

# Chapter 1

# Introduction

## 1.1 Data types

Goal: Define notation of data.

### 1.1.1 Basic

Typically, the data types of interest are:

#### Numerical

**Binary** Used for binary information represented by $\{0, 1\}$ or $\{\text{True}, \text{False}\}$. Typically used for binary classification problems.

**Integer** Typically used to describe the data with limited number of possible numerical descriptors. The most popular representations used signed or unsigned numbers with 8, 16, 32 or 64 bit representation.

**Real** The basic numerical representation. Standard representations are 32 or 64 bit for CPU and 8(new!), 16, 32 bit for GPU.

#### Categorical

**Nominal** Variables with values selected from a group of categories, while not having any kind of natural order.
Example: car type

**Ordinal** A categorical variable whose categories can be meaningfully ordered.
Example: age, grade of exam.

### 1.1.2 Signals and time-series

Two main categories:

- Discrete-time signals that are representation of physical continuous-time prototype signal. Signals typically have constant sampling frequency, and typically handled by signal processing techniques.
  Example: Voltage measurement is a signal and once-an-hour power-meter measurements.
- Time-series are typically derived from a time-stamped discrete-time origin in social sciences. Sometimes have an arbitrary sample times.
  Example: economical parameters.

### 1.1.3 Dataset

The basic dataset includes matrix $\mathbf{X} \in \mathscr{R}^{M \times N}$ ($M$ rows and $N$ columns) and vector $\mathbf{y} \in \mathscr{R}^{M}$.
A single dataset entry is a vector

$$\mathbf{x}_i^T = \begin{bmatrix} x_{i1} & x_{i2} & \cdots & x_{iN} \end{bmatrix}, \tag{1.1}$$

where $i$ is the number of *features* (raw) and $N$ if the dimension (number of columns), $\mathbf{x}_i \in \mathscr{R}^N$ and $x_{ij} \in \mathscr{R}$. All the values of $M$ entries are organized in a matrix form,

$$\mathbf{X} = \begin{bmatrix} — & \mathbf{x}_1^T & — \\ — & \mathbf{x}_2^T & — \\ — & \vdots & — \\ — & \mathbf{x}_M^T & — \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{M1} & x_{M2} & \cdots & x_{MN} \end{bmatrix} \tag{1.2}$$

### 1.1.4 Adversarial attacks

Adversarial example is an input to a machine learning model that is purposely designed to cause a model to make a mistake.
These attacks can happen both during collecting dataset for training and during the inference of an already trained model.

## 1.2 Tasks

Typical related task:

- Prediction or regression, $\mathbf{y}$ is quantitative (Fig. 1.1).
- Classification, $\mathbf{y}$ is categorical.
- Clustering, no $\mathbf{y}$ is provided - it is learned from dataset.
- Anomaly detection, somewhere between classification and clustering.
- Segmentation
- Simulation
- Signal processing tasks: noise removal, smoothing (filling missing values), event/condition detection.

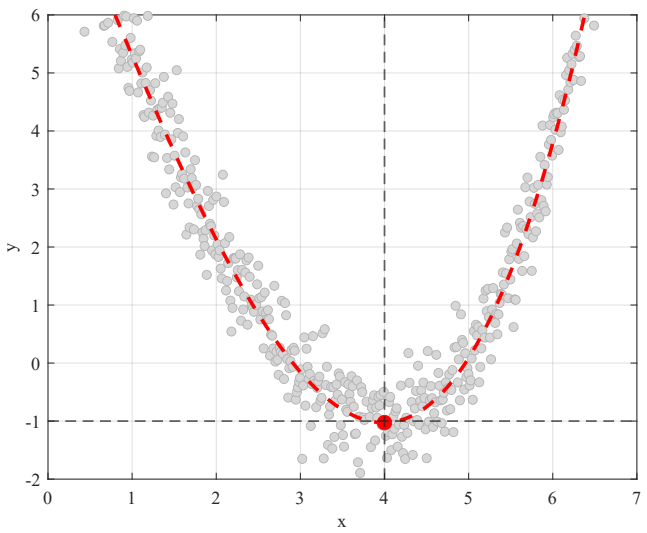



Figure 1.1: Regression example: what is the value of $y$ for given $x$?

## 1.3 Basic workflow

The basic ML/DL workflow is presented in Fig. 1.2. The workflow parts are:

- Data: available data
- Pre-processing: preliminary dataset exploration and validation of dataset integrity (e.g., same physical units for all values of the same feature).
- Model: basic assumptions about the hidden pattern within the data
- Model training: minimization of the loss functions to derive the most appropriate parameters.
- Performance assessment according predefined metrics.

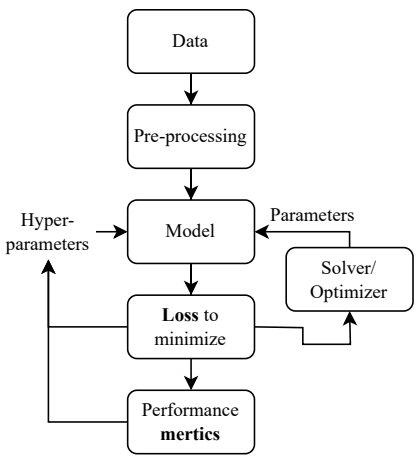



Figure 1.2: Basic workflow of ML/DL solution.

**Baseline** The basic end-to-end workflow implementation is called baseline.

## 1.4 Model

We assume that there is an underling problem (e.g., regression and classification) formulation is of the form

$$y = f(\mathbf{x}) + \epsilon \tag{1.3}$$

where the values of $\mathbf{x}$ (scalar or vector) and $y$ are known (it is the dataset) and $\epsilon$ is some irreducible noise. Sometimes, zero-mean noise is assumed.
The goal is to find the function $f(\cdot)$. The way to define the $f(\cdot;\mathbf{w})$ is termed *model* that depends on some model parameters vector $\mathbf{w}$. The process of finding regression solution by a set of parameters $\mathbf{w}$ is called learning, such as the resulting model can provide output

$$\hat{y}_0 = f(\mathbf{x}_0;\mathbf{w}) \tag{1.4}$$

for some new data $\mathbf{x}_0$.

#### Parameters vs hyper-parameters

**Model parameters**: Model parameters are learned directly from a dataset.
**Hyper-parameters**: Model parameters that are not learned directly from a dataset are called **hyper-parameters**. They are learned in in-direct way during cross-validation process in the follow.

#### Parametric vs non-parametric models

There are two main classes of models: parametric and non-parametric, summarized in Table 1.1.

## 1.5 Loss Function

Loss (or cost) function is a function that relates between dataset outputs $\mathbf{y}$ and model outputs $\hat{\mathbf{y}}$. The parameters $\mathbf{w}$ are minimum of that function,

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \mathscr{L}(\mathbf{y}, \hat{\mathbf{y}}) \tag{1.5}$$

The minimization of the loss function is also termed *training*.

### 1.5.1 Loss Function Minimization

**Goal:** Minimum of the loss function for a given model.

**Closed-form solution** A closed-form solution for $\mathbf{w}$ is a solution that is based on basic mathematical functions. For example, a "normal equation" is a solution for linear regression/classification.

Table 1.1: Comparison of parametric and non-parametric models.

| Aspect | Parametric | Non-parametric |
|---|---|---|
| Dependence on number of parameters on dataset size | Fixed | Flexible |
| Interpretability | Yes | No |
| Underlying data assumptions | Yes | No |
| Risk | Underfitting due to rigid structure | Overfitting due to high flexibility |
| Dataset size | Smaller | Best for larger |
| Complexity | Often fast | Often complex |
| Examples | Linear regression | k-NN, trees |

**Local-minimum gradient-based iterative algorithms** This family of algorithms is applicable only for convex (preferably strictly convex) loss functions. For example, gradient descent (GD) and its modifications (e.g., stochastic GD) are used to evaluate NN parameters. Another example is the Newton-Raphson algorithm.

- Some advanced algorithms under this category also employ (require) second-order derivative $\frac{\partial^2}{\partial \mathbf{w}} \mathscr{L}$ for faster convergence.
- If either derivative is not available as a closed-form expression, it is evaluated numerically.

**Global optimizers** The goal of global optimizers is to find a global minimum of non-convex function. These algorithms may be gradient-free, first-derivative or second-derivative. The complexity of these algorithms is significantly higher than the local optimizer and can be prohibitive for more than a few hundred variables in $\mathbf{X}$.

## 1.6 Metrics

Metrics are quantitative performance indicators of the model that relate between $\mathbf{y}$ and $\hat{\mathbf{y}}$. Sometimes, the minimum of the loss function is also a metric, e.g. mean squared error (MSE).

# Chapter 2

# Least-squares and Linear Regression

**Goal:**
- The goal of the least squares (LS) method is to minimize MSE (or RMSE) between the given data and the parametric model.
- Define and analyze a model that is based on a linear relation between data and the outcome.
- Find the linear model parameters by LS.

## 2.1 Uni-variate Linear LS

### 2.1.1 Definition

The simplest sub-case is the (random) experiment that produces a set of $M$ points (or measurements), $\{x_k, y_k\}_{k=1}^{M}$ [7]. The **linear model** is

$$y = f(x; w_0, w_1) = w_0 + w_1 x + \epsilon, \tag{2.1}$$

where $w_0$ and $w_1$ are the model weights (or parameters) and $\epsilon$ is zero-mean noise. The model outcomes (predictions) are

$$\hat{y}_k = f(x_k; w_0, w_1) = w_0 + w_1 x_k, \tag{2.2}$$

where $\hat{y}_k$ is the prediction outcome of $x_k$.
The performance **metric** is mean-square error (MSE) that is given by

$$\begin{aligned} J_{mse}(w_0, w_1) &= \frac{1}{M}\sum_{k=1}^{M}(y_k - \hat{y}_k)^2 \\ &= \frac{1}{M}\sum_{k=1}^{M} e_k^2 \end{aligned} \tag{2.3}$$

or root-MSE (RMSE)

$$J_{rmse}(w_0, w_1) = \sqrt{J_{mse}(w_0, w_1)}. \tag{2.4}$$

Note, sometimes MSE is termed as sum of squared errors (SSE).
For both of these metrics, the corresponding **loss** (or cost) function to minimize is

$$\begin{aligned} \mathscr{L}(w_0, w_1) &= \sum_{k=1}^{M}(y_k - \hat{y}_k)^2 \\ &= \sum_{k=1}^{M}(y_k - w_0 - w_1 x_k)^2 \end{aligned} \tag{2.5}$$

since either root and/or constant multiplication does not change the desired minimum,

$$\begin{aligned} w_0, w_1 &= \arg\min_{w_0, w_1} J_{mse}(w_0, w_1) \\ &= \arg\min_{w_0, w_1} J_{rmse}(w_0, w_1) \\ &= \arg\min_{w_0, w_1} \mathscr{L}(w_0, w_1) \end{aligned} \tag{2.6}$$

Note that loss function and performance metrics does not have to be the same.

### 2.1.2 Minimization

This minimum is given by a solution of the set of equations,

$$\begin{cases} \dfrac{\partial}{\partial w_0}\mathscr{L}(w_0, w_1) = 0 \\ \dfrac{\partial}{\partial w_1}\mathscr{L}(w_0, w_1) = 0 \end{cases} \tag{2.7}$$

The resulting equations are

$$\begin{cases} 2\sum_{k=1}^{M}(y_k - w_0 - w_1 x_k)\cdot(-1) = 0 \\ 2\sum_{k=1}^{M}(y_k - w_0 - w_1 x_k)\cdot(-x_k) = 0 \end{cases} \tag{2.8}$$

After some basic algebraic manipulations, the resulting set of equations is

$$\begin{cases} w_0 M & + w_1 \sum_{k=1}^{M} x_k = \sum_{k=1}^{M} y_k \\ w_0 \sum_{k=1}^{M} x_k & + w_1 \sum_{k=1}^{M} x_k^2 = \sum_{k=1}^{M} x_k y_k \end{cases} \tag{2.9}$$

This set of equations is termed *normal equation*.
The interesting and numerically stable form of the numerical solution is by usage of average estimation by mean,

$$E[\mathbf{z}] = \bar{\mathbf{z}} = \frac{1}{N}\sum_{k=1}^{N} z_k \tag{2.10}$$

$$\mathrm{Var}[\mathbf{z}] = \overline{\mathbf{z}^2} - \bar{\mathbf{z}}^2 \tag{2.11}$$

$$\mathrm{Cov}[\mathbf{x}, \mathbf{y}] = \overline{\mathbf{x}\mathbf{y}} - \bar{\mathbf{x}}\bar{\mathbf{y}} \tag{2.12}$$

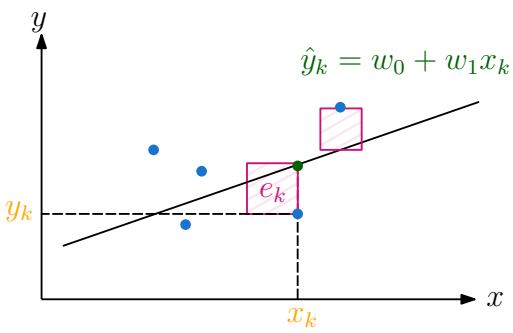



Figure 2.1: Linear regression visualization. The goal is to minimize the total area $\sum_k e_k^2$ of the rectangles.

The resulting prediction is

$$\hat{\mathbf{y}} = E[\mathbf{y}] + \frac{\mathrm{Cov}[\mathbf{x}, \mathbf{y}]}{\mathrm{Var}[\mathbf{x}]}(\mathbf{x} - E[\mathbf{x}]) \tag{2.13}$$

This is *probabilistic* result.
Notes:

- $\mathrm{Var}[\mathbf{x}] \neq 0$ requirement.
- $E[\mathbf{y}] = E[\mathbf{x}] = 0 \Rightarrow w_0 = 0$.

Concluding notes:

- The resulting model is also termed as linear regression, linear trend-line and linear prediction.
- The straightforward solution may result in ill-conditioned matrix. Reformulation of the solution can result in a better numerical stability, e.g. [7, Ch. 5, Question 5, pp. 260]. There are more accurate algorithms than just multiply inverse matrix.
- For numerical stability, the variance of $x_k$ samples is required to be non-zero (distinct $x_k$ values).

## 2.2 Vector/Matrix Notation

### 2.2.1 Uni-variate model

To improve the mathematical representation, vector notation can be used. This time, the points $\{x_k, y_k\}_{k=1}^M$ are organized into vectors, with a few additional ones, as follows,

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{bmatrix}, \quad \mathbf{1}_M = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathbb{R}^M, \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} \tag{2.14}$$

The resulting model notation is

$$\hat{\mathbf{y}} = f(\mathbf{X}; \mathbf{w}) = \mathbf{1}_M w_0 + \mathbf{x} w_1 = \mathbf{X}\mathbf{w}, \tag{2.15}$$

where $\mathbf{X} = \begin{bmatrix} \mathbf{1}_M & \mathbf{x} \end{bmatrix} \in \mathbb{R}^{M \times 2}$ and $\mathbf{w} = \begin{bmatrix} w_0 & w_1 \end{bmatrix}^T$.
The corresponding loss functions is

$$\begin{aligned} \mathscr{L}(\mathbf{w}) &= (\mathbf{y} - \hat{\mathbf{y}})^T (\mathbf{y} - \hat{\mathbf{y}}) = \|\mathbf{y} - \hat{\mathbf{y}}\|^2 \\ &= (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 \end{aligned} \tag{2.16}$$

and the corresponding optimal minimum (Eq. (2.6)) results from the solution of normal equation (matrix form)

$$\nabla_{\mathbf{w}} \mathscr{L}(\cdot) = -\mathbf{X}^T (\mathbf{y} - \mathbf{X}\mathbf{w}) = 0 \tag{2.17}$$

and is given by

$$\begin{aligned} \mathbf{X}^T (\mathbf{y} - \mathbf{X}\mathbf{w}) &= \mathbf{X}^T \mathbf{y} - \mathbf{X}^T \mathbf{X}\mathbf{w} = 0 \\ \mathbf{X}^T \mathbf{y} &= \left(\mathbf{X}^T \mathbf{X}\right) \mathbf{w} \end{aligned}$$

Finally,

$$\mathbf{w}_{opt} = \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{y} \tag{2.18}$$

### 2.2.2 Multivariate LS

For the multivariate $N$-dimensional formulation,

$$\mathbf{X} = \begin{bmatrix} \mathbf{1} & \mathbf{x}_1 & \cdots & \mathbf{x}_N \end{bmatrix} \in \mathbb{R}^{M \times (N+1)} \tag{2.19}$$

$$\mathbf{w} = \begin{bmatrix} w_0 & w_1 & \cdots & w_N \end{bmatrix}^T \in \mathbb{R}^{N+1} \tag{2.20}$$

All the LS discussion on $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$ is the same independent from the number of variables.

#### Dataset

All the data rows in $(\mathbf{X}, \mathbf{y})$ are called dataset. The matrix $\mathbf{X}$ is assumed *full-rank*, i.e. columns are linearly independent.

#### Moore–Penrose inverse (pseudo-inverse)

Moore–Penrose inverse is the extension of an ordinary inverse matrix for none-rectangular matrices,

$$\mathbf{X}^+ = \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{X}^T \in \mathscr{R}^{(N+1) \times M}, \tag{2.21}$$

such that

$$\mathbf{X}^+ \mathbf{X} = \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{X} = \mathbf{I}_{N+1}$$

Note, the by-definition implementation of $\mathbf{X}^+$ may have numerical stability problems with $\left(\mathbf{X}^T \mathbf{X}\right)^{-1}$. All the modern programming languages have numerically-stable and efficient implementation of pseudo-inverse calculations.
The common numerical notation is

$$\mathbf{w}_{opt} = \mathbf{X}^+ \mathbf{y} \tag{2.22}$$

Implementation note: there are numerically optimized algorithms for $\mathbf{w}_{opt}$, such as:

1. `lsqminnorm` (Matlab)
2. Python, `numpy.linalg.lstsq` and `scipy.linalg.lstsq`

### Projection matrix

The model output is given by

$$\begin{aligned}\hat{\mathbf{y}} &= \mathbf{X}\mathbf{w} = \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y} \\ &= \mathbf{X}\mathbf{X}^{+}\mathbf{y} = \mathbf{P}\mathbf{y}\end{aligned} \tag{2.23}$$

where

$$\mathbf{P} = \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T \in \mathscr{R}^{M\times M} \tag{2.24}$$

is a *projection* matrix, i.e. projection of $\mathbf{y}$ into a base derived from $\mathbf{X}$.
Important properties of the matrix $\mathbf{P}$:

- Symmetric $\mathbf{P} = \mathbf{P}^T$,
- Idempotent $\mathbf{P} = \mathbf{P}^2$,
- Orthogonality, $\mathbf{P} \perp (\mathbf{I} - \mathbf{P})$
  Proof. $\mathbf{P}(\mathbf{I} - \mathbf{P}) = \mathbf{P} - \mathbf{P}^2 = \mathbf{0}$.
- $\mathbf{I} - \mathbf{P}$ is also projection matrix.

### Model error

The model error is

$$\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}} = \mathbf{y} - \mathbf{P}\mathbf{y} = (\mathbf{I} - \mathbf{P})\,\mathbf{y}, \tag{2.25}$$

such that $\mathscr{L}(\mathbf{w}) = \mathbf{e}^2$, $J_{mse}(\mathbf{w}) = \overline{\mathbf{e}^2}$, $J_{rmse}(\mathbf{w}) = \sqrt{\overline{\mathbf{e}^2}}$.

### Error and data orthogonality

$$\mathbf{e} \perp \mathbf{X} \Rightarrow \mathbf{X}^T\mathbf{e} = \mathbf{0}_{N+1} \tag{2.26}$$

Proof:

$$\begin{aligned}\mathbf{X}^T\mathbf{e} = \mathbf{X}^T\mathbf{y} - \mathbf{X}^T\mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y} \\ \mathbf{X}^T\mathbf{y} - \mathbf{X}^T\mathbf{y} = \mathbf{0}\end{aligned} \tag{2.27}$$

### Error and prediction orthogonality

$$\mathbf{e} \perp \hat{\mathbf{y}} \Rightarrow \hat{\mathbf{y}}^T\mathbf{e} = \mathbf{e}^T\hat{\mathbf{y}} = 0 \tag{2.28}$$

Proof:

$$\begin{aligned}\hat{\mathbf{y}}^T\mathbf{e} &= \mathbf{y}^T\mathbf{P}\,(\mathbf{I} - \mathbf{P})\,\mathbf{y} \\ &= \mathbf{y}^T\mathbf{P}\mathbf{y} - \mathbf{y}^T\mathbf{P}\mathbf{P}\mathbf{y} \\ &= \mathbf{y}^T\mathbf{P}\mathbf{y} - \mathbf{y}^T\mathbf{P}\mathbf{y} = 0\end{aligned} \tag{2.29}$$

The interesting outcome of this property is a relation between error and prediction,

$$\|\mathbf{y}\|^2 = \|\hat{\mathbf{y}}\|^2 + \|\mathbf{e}\|^2 \tag{2.30}$$

*Proof.*

$$\begin{aligned}\|\mathbf{y}\|^2 &= \|\hat{\mathbf{y}} + \mathbf{e}\|^2 \\ &= (\hat{\mathbf{y}} + \mathbf{e})^T(\hat{\mathbf{y}} + \mathbf{e}) \\ &= \hat{\mathbf{y}}^T\hat{\mathbf{y}} + \mathbf{e}^T\mathbf{e}\end{aligned} \tag{2.31}$$

### Average error

The average error is zero-mean,

$$\begin{aligned}\bar{\mathbf{e}} &= \frac{1}{M}\sum_{k=1}^{M} e_k \\ &= \sum_{k=1}^{M} e_k = \mathbf{1}^T\mathbf{e} = 0\end{aligned} \tag{2.32}$$

*Proof.*

$$\begin{aligned}&\mathbf{X}^T\mathbf{e} = 0 \leftarrow (2.26) \\ &\Rightarrow \begin{bmatrix}\mathbf{1}^T \\ \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_N^T\end{bmatrix}\mathbf{e} = \begin{bmatrix}\mathbf{1}^T\mathbf{e} \\ \vdots\end{bmatrix} = 0\end{aligned} \tag{2.33}$$

The interesting consequence is

$$\bar{\mathbf{y}} = \bar{\hat{\mathbf{y}}} \tag{2.34a}$$

$$= w_0 + w_1\bar{\mathbf{x}}_1 + \cdots + w_N\bar{\mathbf{x}}_N \tag{2.34b}$$

*Proof.*

$$\begin{aligned}\bar{\mathbf{y}} &= \overline{\hat{\mathbf{y}} + \mathbf{e}} \\ &= \bar{\hat{\mathbf{y}}} + \bar{\mathbf{e}}\end{aligned} \tag{2.35}$$

### Error distribution

The values of the error vector $\mathbf{e}$ are assumed to be normally distributed, due to Central Limit Theorem (CLT). Typically, this assumption is not need in ML, but it is important for statistical analysis for small values of $M$.

### MSE

The reduced expression for the resulting minimal loss is

$$\begin{aligned}\mathscr{L}_{min} &= \sum_{k=1}^{M} y_k^2 - \sum_{j=0}^{N} w_j\mathbf{y}^T\mathbf{x}_j \\ &= \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\mathbf{w}\end{aligned} \tag{2.36}$$

*Proof.*

$$\begin{aligned}mse_{min} &= \mathbf{e}^T\mathbf{e} \\ &= (\mathbf{y} - \hat{\mathbf{y}})^T\mathbf{e} \\ &= \mathbf{y}^T\mathbf{e} - \cancel{\hat{\mathbf{y}}^T\mathbf{e}} \\ &= \mathbf{y}^T(\mathbf{y} - \mathbf{X}\mathbf{w}) \\ &= \mathbf{y}^T\mathbf{y} - \underbrace{\mathbf{y}^T\begin{bmatrix}\mathbf{1} & \mathbf{x}_1 & \cdots & \mathbf{x}_N\end{bmatrix}}_{\mathscr{R}^{1\times(N+1)}}\mathbf{w}\end{aligned} \tag{2.37}$$

The MSE or RMSE evaluation from the loss is straightforward.

## 2.3 Coefficient of Determination

To emphasis the difference between loss and metrics, the following example of LR metric is provided, A coefficient of determination, denoted $R^2$ or $r^2$ (R-square) is based on the relation

$$\underbrace{\sum_{k=1}^{M}(y_k-\bar{\mathbf{y}})^2}_{\text{SST}}=\underbrace{\sum_{k=1}^{M}(\hat{y}_k-\bar{\mathbf{y}})^2}_{\text{SSR}}+\underbrace{\sum_{k=1}^{M}e_k^2}_{\text{SSE}} \tag{2.38}$$

- SST - total sum of squares
- SSR - sum of squares due to regression
- SSE - sum of square errors (or residual sum of squares)

$$R^2=\frac{\text{SSR}}{\text{SST}}=1-\frac{\text{SSE}}{\text{SST}} \tag{2.39}$$

Typically, $0 \lesssim R^2 \leq 1$ (may be negative under certain circumstances). $R^2$ is unitless.

## 2.4 Iterative Solution - Gradient descent (GD)

**Goal:** Find the minimum of the function:
- First-order derivative based.
- Local minimum.

Let's assume some function $y=f(x)$, with $x,y\in\mathscr{R}$, differentiable with $\frac{dy}{dx}=f'(x)$.
- $f'(x)$ is a slope of $f(x)$ at a point $x$.
- By the definition of the derivative, for some small $\epsilon$,
$$f(x+\epsilon)\approx f(x)+\epsilon f'(x)$$
- Given the sign of the derivative,
$$\begin{aligned} f(x-\epsilon)<f(x), &\quad f'(x)>0\\ f(x+\epsilon)<f(x), &\quad f'(x)<0 \end{aligned}$$
- For sufficiently small $\epsilon$,
$$f(x-\epsilon\,\text{sign}(f'(x)))\leq f(x)$$

The idea of the algorithm is to reduce $f(x)$ by going in direction opposite sign of derivative, $f'(x)$.
**Gradient descent (GD) - scalar function**: For differentiable function $f(x)$, the iterative algorithm

$$x_{n+1}=x_n-\alpha f'(x_n) \tag{2.40}$$

converges to some local minimum of $f(x)$.
Required parameters are:
- Step-size $\alpha>0$ is some positive constant or some function of $n$, $\alpha_n$.
- $x_0$ is an initial guess.

Some of the most common stopping conditions are:
- Reaching the point of slow convergence, $\left|x_{n+1}-x_n\right|<\epsilon$.
- Limiting the number of iterations, $n\leq n_0$.

**Gradient descent (GD) - vector function**: For differentiable multivariate and multidimensional function $f(\mathbf{x}):\mathscr{R}^N\to\mathscr{R}^N$, the iterative algorithm is

$$\mathbf{x}_{n+1}=\mathbf{x}_n-\alpha\nabla_{\mathbf{x}}f'(\mathbf{x}_n). \tag{2.41}$$

Each dimension is iteratively reduced according to its derivative. Notes:
- Easy to implement.
- Requires analytical or numerical derivative.
- Non-trivial selection of the optimal value of $\alpha$. In more general case, vector of $n$-dependent values may be desirable.
- Useful only for the function with single (global) minimum, such as MSE minimization.

**GD for MMSE**: Optimal values of $\mathbf{w}$ may be found by

$$\begin{aligned} \mathbf{w}_{n+1}&=\mathbf{w}_n-\alpha\nabla_{\mathbf{w}}\mathscr{L}\\ &=\mathbf{w}_n-\frac{\alpha}{M}\mathbf{X}^T\left(\mathbf{X}\mathbf{w}_n-\mathbf{y}\right) \end{aligned} \tag{2.42}$$

## 2.5 Takeaways

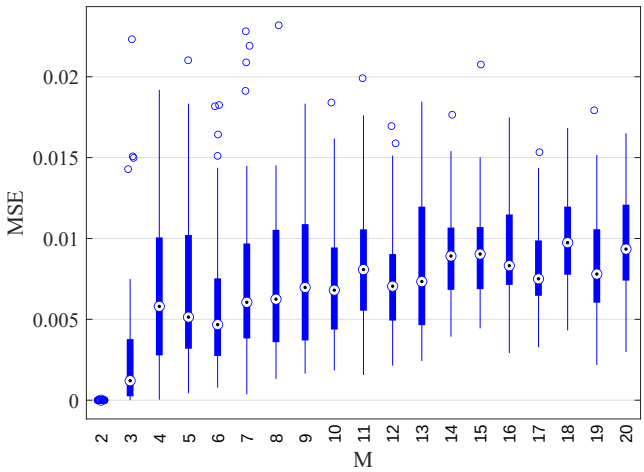



Figure 2.2: MSE as a function of number of data points, $M$.

These assignments will help you understand the practical implications of linear regression and the influence of data size on model performance.

# Preface

In recent years, the convergence of machine learning (ML) and signal processing (SP) has gathered growing attention in engineering education. Students are often introduced to ML principles at an early stage, yet many advanced SP topics, ranging from linear systems and time-frequency analysis to probabilistic modeling, traditionally require multiple specialized courses [4]. Although these SP methods yield comprehensive performance insights and rigorous conclusions, teaching them can be both timely and demanding.

A key bridge between basic ML concepts and advanced SP techniques is the *least squares* (LS) method. LS is grounded in a simple and intuitive idea: minimizing the sum of squared errors. While direct LS computations may be $\mathcal{O}(N^3)$, and thus less efficient than typical SP methods ($\mathcal{O}(N \log N)$ to $\mathcal{O}(N^2)$), the LS perspective fosters a simpler, data-driven understanding of fundamental SP tasks. For example, the estimation of sinusoidal signal parameters in noise can be introduced by viewing it purely as a regression problem, bypassing the need for more involved probabilistic analyses. Likewise, the discrete Fourier transform (DFT) can be reframed as an extension of sinusoidal parameter estimation, illustrating SP principles with real arithmetic alone.

An LS-centric viewpoint aligns well with the foundational prerequisites of many ML courses and can be integrated at an early stage of engineering or data science programs. It offers an accessible path for teaching core SP ideas to engineering students who might lack extensive mathematical or probabilistic training. Although the underlying techniques are not new, this data-driven, regression-based interpretation may be more intuitive for those already familiar with basic ML concepts, enabling them to explore SP topics with minimal additional theoretical overhead.

# Chapter 3

# Basic Signal Analysis

**Goal:** This chapter introduces the fundamental concepts and methods for analyzing and estimating (learning) parameters of a discrete-time sinusoidal signal observed in additive noise.

## 3.1 Signal Preliminaries

A general continuous-time cosine signal can be written as

$$\begin{aligned} y(t) &= A\cos(2\pi F_0 t + \theta) + \epsilon(t), \\ &= A\cos(\Omega_0 t + \theta) + \epsilon(t), \end{aligned} \tag{3.1}$$

where

- $A > 0$ is the amplitude,
- $-\pi < \theta \leq \pi$ is the phase,
- $F_0$ is the frequency in Hz,
- $\Omega_0 = 2\pi F_0$ is the radial frequency in rad/sec,
- $\epsilon(t)$ is zero-mean additive noise.

The only assumption for the additive noise is that it is zero-mean,

$$\sum_n \epsilon[n] = 0. \tag{3.2}$$

No additional assumptions, such as Gaussianity, are applied; however, the special case of additive white Gaussian noise (AWGN) is is further refined as tips for selected topics.
For the further analysis, we use the sampled version $x[n]$ of the continuous-time signal $x(t)$, sampled with frequency $F_s = 1/T$,

$$\begin{aligned} y[n] &= y(nT) \\ &= A\cos(\omega_0 n + \theta) + \epsilon[n] \quad n = 0, \ldots, L-1, \end{aligned} \tag{3.3}$$

where

$$\omega_0 = 2\pi F_0 T = 2\pi \frac{F_0}{F_s} \tag{3.4}$$

is the angular frequency (measured in rad) derived from the analog frequency $F_0$ and $L$ is the resulting number of samples.
In order to accurately reproduce a cosine signal, the Nyquist criterion demands $F_0 < F_s/2$, which implies $\omega_0 < \pi$. This requirement can be easily illustrated by the following example. Consider two signals:

$$x_1(t) = \cos(0.6\pi t), \quad x_2(t) = \cos(2.6\pi t)$$

Sampling with $F_s = 1$ Hz results in two identical signals,

$$\begin{aligned} x_1[n] &= \cos(0.6\pi n), \\ x_2[n] &= \cos(2.6\pi n) = \cos(0.6\pi n + 2\pi n) = x_1[n]. \end{aligned}$$

This phenomenon is called aliasing. Note, when $\omega_0 = 0$ the signal is the DC level, $y(t) = y[n] = A\cos(\theta)$. Therefore, the sampling frequency requirement is $0 \leqslant \omega < \pi$. This relation holds for all the following discussions and derivations.
The energy of the signal $x[n]$ is defined as

$$E_{\mathbf{x}} = \|\mathbf{x}\|^2 = \sum_n x^2[n], \tag{3.5}$$

where $\mathbf{x}$ is the vector of samples of the signal $x[n]$. The corresponding power is

$$P_{\mathbf{x}} = \frac{1}{N} E_{\mathbf{x}} = \frac{1}{N}\|\mathbf{x}\|^2. \tag{3.6}$$

## 3.2 Amplitude estimation

**Goal:** Find the amplitude of a sinusoidal signal in noise that best fits the model in a least squares (LS) sense.

Given a signal model with a known frequency $\omega_0$,

$$y[n] = A\cos(\omega_0 n) + \epsilon[n] \quad n = 0, \ldots, L-1 \tag{3.7}$$

the goal is to estimate the amplitude $A$ that best fits a provided model,

$$\hat{y}[n] = A\cos(\omega_0 n) \tag{3.8}$$

Technically, we are looking for the value of $A$ that minimizes the squared error,

$$\mathscr{L}(A) = \sum_n (y[n] - \hat{y}[n])^2. \tag{3.9}$$

In the linear LS regression formulation, we define the corresponding parameters $\mathbf{y}, \mathbf{X}$ and $\mathbf{w}$. First, the required weight is $\mathbf{w} = A$. The matrix $\mathbf{X}$ is formed by samples of the signal $\{\cos(\omega_0 n)\}_{n=0}^{L-1}$,

$$\mathbf{X} = \begin{bmatrix} 1 & \cos(\omega_0) & \cos(2\omega_0) & \cdots & \cos\big((L-1)\omega_0\big) \end{bmatrix}^T \tag{3.10}$$

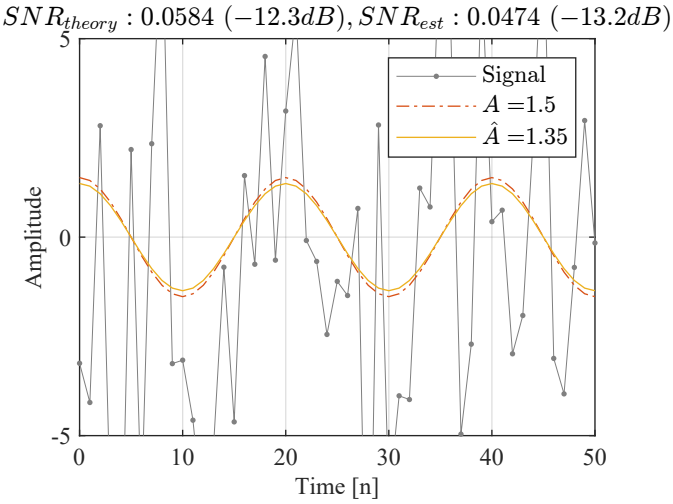
Finally, $\mathbf{y}$ is the vector of samples of $\{y[n]\}_{n=0}^{L-1}$. The resulting solution is straightforward,

$$\begin{aligned}\hat{A} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \\ &= \frac{\sum_n x[n]y[n]}{\sum_n x^2[n]} \\ &= \frac{\sum_n y[n]\cos(\omega_0 n)}{\sum_n \cos^2(\omega_0 n)}\end{aligned} \tag{3.11}$$

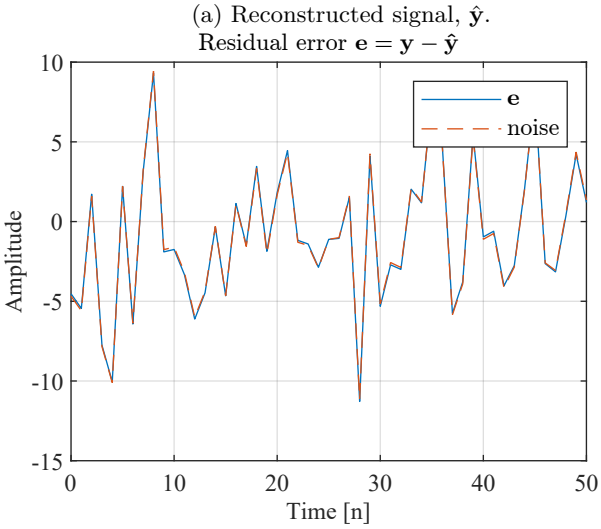
If we substitute the model into the resulting solution,

$$\begin{aligned}\hat{A} &= \frac{\sum_n x[n]\left(Ax[n]+\epsilon[n]\right)}{\sum_n x^2[n]} \\ &= A + \frac{\sum_n x[n]\epsilon[n]}{\sum_n x^2[n]},\end{aligned} \tag{3.12}$$

it produces a true value of $A$ with some additive noise.




(a) Reconstructed signal, $\hat{\mathbf{y}}$.




(b) Residual error. Ideally, if the model was perfect, the residual would be equal to the added noise.

Figure 3.1: Example of the cosine signal amplitude estimation. Note the negative sign of SNR in dB units.

The resulting residual error is given by

$$\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}}. \tag{3.13}$$

Since $\mathbf{e} \perp \hat{\mathbf{y}}$ the power/energy terms can be decomposed as follows,

$$\|\mathbf{y}\|^2 = \|\hat{\mathbf{y}}\|^2 + \|\mathbf{e}\|^2, \quad P_{\mathbf{y}} = P_{\hat{\mathbf{y}}} + P_{\mathbf{e}}. \tag{3.14}$$

An interesting interpretation of this result is estimated signal to noise ratio (SNR), defined as

$$\widehat{SNR} = \frac{\left\|\hat{\mathbf{y}}^2\right\|}{\left\|\mathbf{e}^2\right\|} \tag{3.15}$$

Moreover, due to zero-mean property of the noise, the estimated variance of the noise is

$$\hat{\sigma}_\epsilon^2 = \frac{1}{L}\|\hat{\mathbf{e}}\|^2. \tag{3.16}$$

The following example (Fig. 3.1) uses a synthetic cosine signal of length $L = 51$ samples, angular frequency $\omega_0 = 0.1\pi$ and amplitude $A = 1.5$. Gaussian noise with standard deviation $\sigma = 5$ is then added to create a noisy observation. A least-squares regression is applied to estimate the amplitude, yielding $\hat{\sigma}_\epsilon = 4.43$.

## 3.3 Amplitude and phase estimation

**Goal:** Find amplitude and phase of a sinusoidal signal in noise.

The following analysis is provided for the more general model,

$$y[n] = A\cos(\omega_0 n + \theta) + \epsilon[n] \quad n = 0, \ldots, L-1, \tag{3.17}$$

with two unknown parameters, the amplitude $A$ and the phase $\theta$.
The linear LS reformulation of the signal model

$$\hat{y}[n] = A\cos(\omega_0 n + \theta) \tag{3.18}$$

involves the use of trigonometric identities to express the cosine with a phase shift as a linear combination of sine and cosine signals,

$$A\cos(\omega_0 n + \theta) = w_c\cos(\omega_0 n) + w_s\sin(\omega_0 n), \tag{3.19}$$

where

$$\begin{aligned}w_c &= A\cos(\theta) \\ w_s &= -A\sin(\theta).\end{aligned} \tag{3.20}$$

This transforms the problem into a two-parameter linear LS problem in terms of $w_c$ and $w_s$ [1]. The resulting LS formulation involves a two-valued vector of linear coefficients, $\mathbf{w} = \begin{bmatrix} w_c & w_s \end{bmatrix}^T$, the vector $\mathbf{y}$ of samples of $y[n]$, and the matrix $\mathbf{X}$ of dimensions $L \times 2$ that is given by

$$\mathbf{X} = \begin{bmatrix} 1 & 0 \\ \cos(\omega_0) & \sin(\omega_0) \\ \cos(2\omega_0) & \sin(2\omega_0) \\ \vdots & \vdots \\ \cos\big((L-1)\omega_0\big) & \sin\big((L-1)\omega_0\big) \end{bmatrix}. \tag{3.21}$$

Once $\hat{\mathbf{w}}$ has been found, the amplitude and phase can be recovered from

$$A = \sqrt{w_c^2 + w_c^2}$$
$$\theta = -\arctan\left(\frac{w_c}{w_s}\right) \tag{3.22}$$

SNR and noise variance interpretations are similar to in the previous model in Eqs. (3.15) and (3.16).
The numerical example is presented in Fig. 3.2. The configuration is similar to the previous figure, expect the lower noise variance, $\sigma = 1.5$. Nevertheless, there is a decrease in performance, since two parameters are estimated simultaneously.

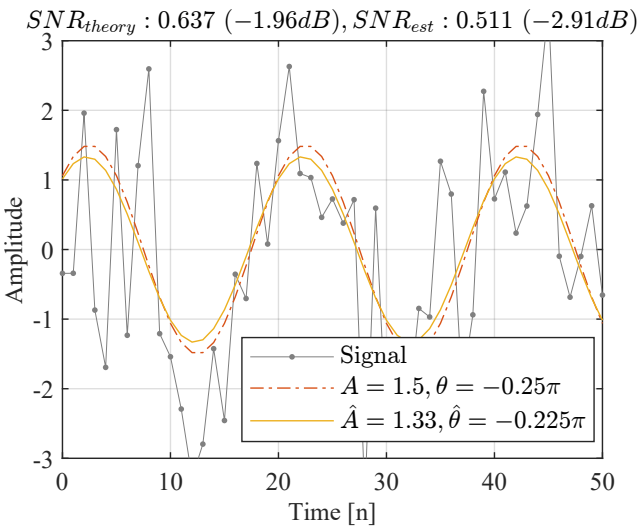



Figure 3.2: Example of the cosine signal amplitude and phase estimation. Note the lower estimation accuracy compared to Fig. 3.1, since two parameters are estimated simultaneously.

**Implementation Tip**

- This estimation procedure is optimal in the maximum likelihood (ML) sense under additive white Gaussian noise (AWGN) and achieves the Cramér-Rao lower bound (CRLB) [1, 6].
- The theoretical lower bound (also termed Cramer-Rao lower bound(CRLB)) on the average estimation accuracy of $w_c, w_s$ is given by [1, Eqs. (5.47-48)]
$$\mathrm{Var}\left[\hat{w}_{c,s}\right] \gtrsim \frac{2\sigma^2}{L} \tag{3.23}$$
This bound is the tightest for the AWGN case and is less accurate for other noise distributions.
- The approximated estimation variance can be easily evaluated by Monte-Carlo simulations for any set of parameters and any distribution of interest.

## 3.4 Frequency estimation

**Goal:** If the frequency $\omega_0$ is also unknown, it can be estimated by searching for the $\hat{\omega}_0$ that best fits a sinusoidal model for the observed data, i.e., that minimizes the residual error norm or maximizes the reconstructed signal energy.

Since $\omega_0$ is unknown, the matrix $\mathbf{X}$ may be parameterized as a frequency-dependent one, $\mathbf{X}(\omega)$. Here, the estimated signal is frequency-dependent

$$\hat{\mathbf{y}}(\omega) = \mathbf{X}(\omega)\mathbf{w}(\omega), \tag{3.24}$$

where $\mathbf{w}(\omega)$ are the estimated parameters $w_c$ and $w_s$ at that frequency. The corresponding frequency-dependent residual error is given by

$$\mathbf{e}(\omega) = \mathbf{y} - \hat{\mathbf{y}}(\omega). \tag{3.25}$$

Since the error $\mathbf{e}(\omega)$ is orthogonal to $\hat{\mathbf{y}}$,

$$\|\mathbf{y}\|^2 = \|\hat{\mathbf{y}}(\omega)\|^2 + \|\mathbf{e}(\omega)\|^2. \tag{3.26}$$

To find the frequency that best represents the data, we seek the one that maximizes the energy of the reconstructed signal (or equivalently minimizes the residual error), as mentioned above

$$\hat{\omega_0} = \arg\min_{\omega} \|\mathbf{e}(\omega)\|^2 = \arg\max_{\omega} \|\hat{\mathbf{y}}(\omega)\|^2. \tag{3.27}$$

Note, this optimization problem can be challenging because the objective function may exhibit multiple local maxima/minima. Therefore, an appropriate numerical global optimization method is required.
Once $\hat{\omega}_0$ has been found, the amplitude and phase are estimated using the corresponding linear LS solution $\mathbf{w}(\omega_0)$. This solution also results in SNR and noise variance estimations, as in Eqs. (3.15) and (3.16).

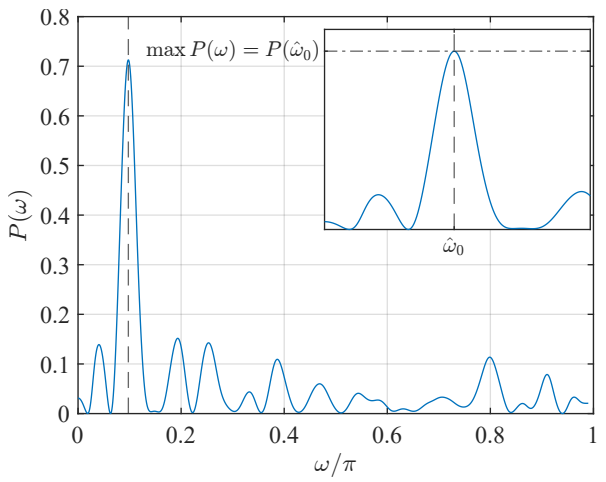
**Tip: Interpretation in Terms of the Periodogram**
The function

$$P(\omega) = \frac{1}{L}\|\hat{\mathbf{y}}(\omega)\|^2 \tag{3.28}$$

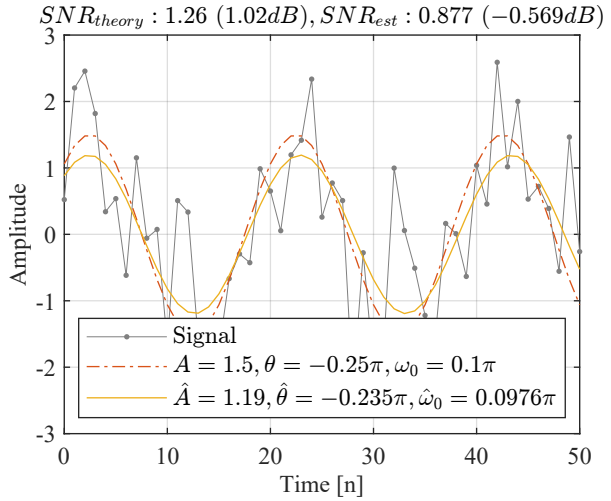
as a function of $\omega$ is termed a periodogram that is a frequency-dependent measure of signal power that approximates the power spectral density (PSD) of the signal. By scanning over frequencies, the $\omega$ that yields the maximum periodogram value is taken as the frequency estimate, $\omega_0$.
A numerical example of the signal with additive white Gaussian noise (AWGN), and with the parameters $A = 1.5, \omega_0 = 0.1\pi, \theta = -\pi/4$ and $\sigma_\epsilon^2 = 1$, is presented in Fig. 3.3. First, periodogram peak is found (Fig. 3.3a). Than, the subsequent amplitude/phase estimation result is presented (Fig. 3.3b).

**Tip: Theoretical performance bounds** Under AWGN assumption, theoretical SNR is given by

$$SNR = \frac{A^2}{2\sigma^2} \tag{3.29}$$




(a) The periodogram $P(\omega)$ with a prominent peak at $\omega_0 \approx 0.1\pi$.




(b) Reconstructed signal.

Figure 3.3: The reconstruction in (b) uses the estimated amplitude, phase, and angular frequency $(\hat{A}, \hat{\theta}, \hat{\omega}_0)$ found by maximizing the periodogram in (a). Note the lower estimation accuracy than in Fig. 3.2, since three parameters are estimated.

and the corresponding CRLB on the estimation variances are [6]

$$\mathrm{Var}\left[\hat{A}\right] \geqslant \frac{2\sigma^2}{L} \quad [V^2] \tag{3.30}$$

$$\mathrm{Var}[\hat{\omega}_0] \geqslant \frac{12}{SNR \times L(L^2-1)} \approx \frac{12}{SNR \times L^3} \quad \left[\left(\frac{rad}{sample}\right)^2\right] \tag{3.31}$$

$$\mathrm{Var}\left[\hat{\theta}\right] \geqslant \frac{2(2L-1)}{SNR \times L(L+1)} \approx \frac{4}{SNR \times L} \quad [rad^2] \tag{3.32}$$

For analog frequency $F_0 = \frac{\omega_0}{2\pi} F_s$,

$$\mathrm{Var}[F_0] = \mathrm{Var}[\omega_0] \left(\frac{F_s}{2\pi}\right)^2 \quad [Hz^2] \tag{3.33}$$

In practice, for short data lengths or non-Gaussian noise, these bounds provide only approximate guides to achievable performance.

## 3.5 Harmonic Signal Analysis

A particularly important class of signals encountered in many practical applications is the *harmonic* or *periodic* signal. Such a signal can be expressed as a sum of cosine terms whose frequencies are integer multiples (harmonics) of a fundamental frequency $\omega_0$.

$$y[n] = A_0 + \sum_{m=1}^{M} A_m \cos(m\omega_0 n + \theta_m), \tag{3.34}$$

where:

- $A_0$ is the constant (DC) component,
- $A_m$ and $\theta_m$ represent the amplitude and phase of the $m$-th harmonic,
- $\omega_0$ is the fundamental angular frequency,
- $m\omega_0$ corresponds to the frequency of the $m$-th harmonic,
- and $M$ is the number of harmonics in the model.

Given $\omega_0$, the model is linear in terms of the unknown parameters $\{A_m, \theta_m\}$ for each harmonic $m = 1, \ldots, M$. Similar to the single-frequency case, the LS matrix $\mathbf{X}$ is constructed with columns corresponding to $\cos(m\omega_0 n)$ and $\sin(m\omega_0 n)$ for $m = 1, \ldots, M$, plus a column of ones for the DC component. Each pair $(A_m, \theta_m)$ can be recovered from the LS estimated cosine and sine coefficients in the manner described for single-frequency amplitude-phase estimation. The resulting SNR and noise variance estimates are similar to those described in the previous sections.

The model order $M$ (number of harmonics) is a hyper-parameter that should be chosen carefully. Too few harmonics can fail to capture essential signal structure, while too many may overfit noise. The maximum value of $M$ is bounded by the Nyquist criterion, $M < \pi/\omega_0$. If $\omega_0$ is not known, the approach that is described in the frequency estimation section can also be applied here. Once $\hat{\omega}_0$ has been determined from a maximum of the harmonic periodogram,

$$P_h(\omega) = \frac{1}{L} \sum_{m=1}^{M} \left\|\mathbf{y}(m\omega)\right\|^2, \tag{3.35}$$

the harmonic amplitudes and phases can be estimated via LS at this frequency [3].

Total harmonic distortion (THD) is a measure commonly used in electrical engineering, audio processing, and other fields to quantify how much the harmonic components of a signal differ from a pure sinusoid at the fundamental frequency. It is defined as the ratio of the root-sum-square of the harmonic amplitudes and the amplitude of the fundamental frequency,

$$THD = \frac{\sqrt{\sum_{m=2}^{M} A_m^2}}{A_1}. \tag{3.36}$$

A lower THD value indicates that the signal is closer to a pure sinusoidal shape, whereas a higher THD signifies a stronger presence of higher-order harmonics.
The example is the sampled current of a switch-mode power supply in a 50Hz network sampled at a 50kHz frequency [2]. Figure 3.4a shows a reconstruction of the signal with $M = 250$ harmonics. The estimated amplitudes $\hat{A}_m$ are shown (Fig. 3.4b) as a function of the harmonic index $m$, including the DC term at $m = 0$. A larger magnitude indicates a more prominent harmonic component. The first non-DC harmonic amplitude $m = 1$ corresponds to the fundamental frequency, $\omega_0$, while higher indices capture additional harmonics in the signal. The estimated fundamental frequency is 50.104Hz with the corresponding THD of about 1.6. Figure 3.4c shows estimated SNR (top) and the noise standard deviation (bottom) vary as the number of harmonics $M$ in the model increases.
**Tip:** The frequency estimator is an effective ML estimator with known analytical CRLB [3].

## 3.6 Discrete Fourier Transform (DFT)

The discrete Fourier transform (DFT) can be viewed as a systematic way of decomposing a finite-length signal into a sum of harmonically related sinusoids. In fact, it is a special case of the harmonic signal representation discussed earlier. Specifically, setting the fundamental angular frequency to $\omega_0 = \frac{2\pi}{N}$ and using $N \geq L - 1$ harmonics, the harmonic model reduces exactly to a DFT decomposition that provides a natural harmonic decomposition of the signal into $N$ harmonics that are evenly spaced in frequency.
DFT representation assumes that any arbitrary, finite-time signal $y[n]$ may be represented as a sum of sinusoidal signals,

$$y[n] = \sum_{k=0}^{N-1} A_k \cos\left(k\frac{2\pi}{N}n + \theta_k\right), \quad n = 0, \ldots, L-1 \tag{3.37}$$

When $N \geq L$, the DFT allows for perfect reconstruction of the signal using its harmonic representation:

$$\mathbf{y} = \mathbf{X}\hat{\mathbf{w}}.$$

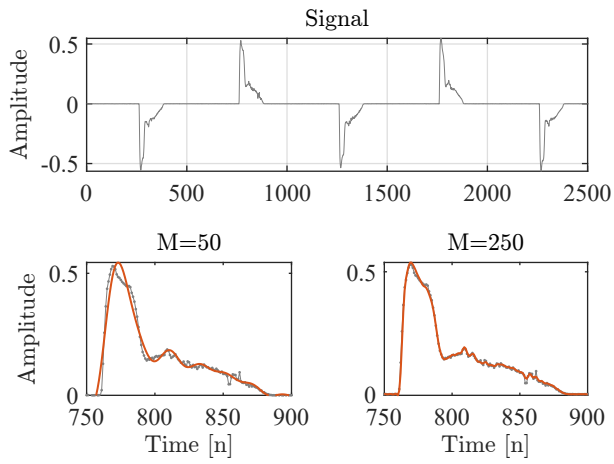
It is also worth noting the symmetry in the DFT,

$$\begin{aligned} \cos\big((N-k)\Delta\omega\big) &= \cos\big(k\Delta\omega\big) \\ \sin\big((N-k)\Delta\omega\big) &= -\sin\big(k\Delta\omega\big), \end{aligned} \tag{3.38}$$

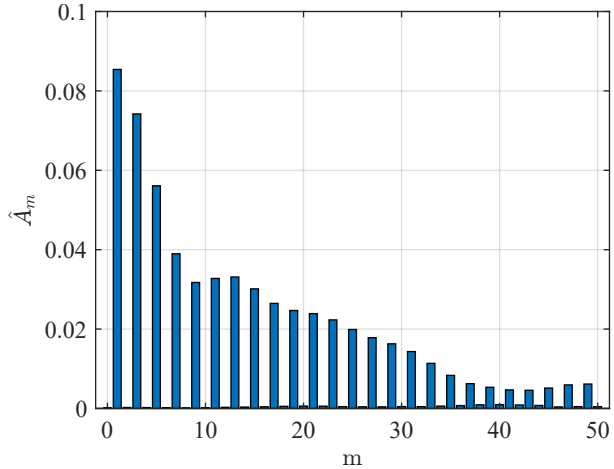
resulting in redundant information for frequencies $k\omega_0$ above and below $\pi$,

$$\begin{aligned} A_k &= A_{N-k} \\ \theta_k &= -\theta_{N-k} \end{aligned}. \tag{3.39}$$

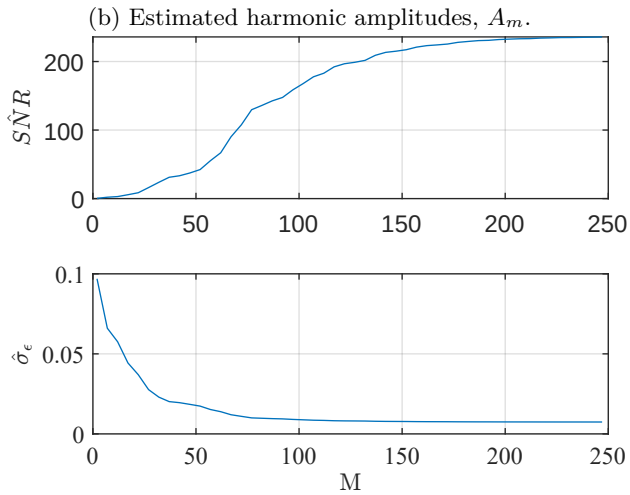
As a result, only frequencies $k\omega_0 \leq \pi$ need to be considered uniquely.




(a) The upper plot shows the signal overlaid with the least-squares harmonic reconstruction using the estimated frequency, amplitude, and phase parameters. The lower plots zoom in on a smaller portion of the time axis for different values of $M$, and demonstrate the challenging shape of the signal.




(b) Estimated harmonic amplitudes, $A_m$.




(c) Estimated SNR and noise std, $\hat{\sigma}_\epsilon$.

Figure 3.4: Example of a harmonic signal analysis.

### 3.6.1 Single frequency analysis

Consider a signal $y[n]$ assume a discrete frequency $\omega_0 = \frac{2\pi}{N}k$ is given. To estimate amplitude and phase at this predefined frequency, we can form a matrix $\mathbf{X}$,

$$\mathbf{X}_1 = \begin{bmatrix} \mathbf{x}_c & \mathbf{x}_s \end{bmatrix},$$

where

$$\mathbf{x}_c = \begin{bmatrix} \cos\left(2\pi\frac{k}{N}\cdot 0\right) \\ \cos\left(2\pi\frac{k}{N}\cdot 1\right) \\ \vdots \\ \cos\left(2\pi\frac{k}{N}(L-1)\right) \end{bmatrix} \text{ and } \mathbf{x}_s = \begin{bmatrix} \sin\left(2\pi\frac{k}{N}\cdot 0\right) \\ \sin\left(2\pi\frac{k}{N}\cdot 1\right) \\ \vdots \\ \sin\left(2\pi\frac{k}{N}(L-1)\right) \end{bmatrix}.$$

By evaluating $\mathbf{X}_1^T\mathbf{X}_1$, we find that the sine and cosine columns form an orthogonal basis for this single frequency, with

$$\mathbf{x}_c^T\mathbf{x}_c = \frac{N}{2}, \tag{3.40a}$$

$$\mathbf{x}_s^T\mathbf{x}_s = \frac{N}{2}, \tag{3.40b}$$

$$\mathbf{x}_c^T\mathbf{x}_s = 0. \tag{3.40c}$$

Stacking these results for all $k = 0, \ldots, N-1$ yields the complete DFT matrix forms a complete orthogonal basis for the $L$-sample signal space. The further discussion of $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$ matrix properties may be found in Examples 4.2 and 8.5 in [6].

Moreover, since $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$ takes a particularly simple diagonal form, and the least squares solution $\hat{\mathbf{w}}$ for the parameters $w_{c,k}$ and $w_{s,k}$ (corresponding to amplitude and phase components at $\omega_k$) is

$$w_{c,k} = \frac{2}{N}\sum_{n=0}^{L-1} y[n]\cos\left(2\pi\frac{k}{N}n\right), \tag{3.41a}$$

$$w_{s,k} = \frac{2}{N}\sum_{n=0}^{L-1} y[n]\sin\left(2\pi\frac{k}{N}n\right). \tag{3.41b}$$

**Tip** The fast Fourier transform (FFT) algorithm efficiently computes $Y[k]$, providing $A_k = |Y[k]|/N$ and $\theta_k = \angle(Y[k])$ with significantly lower memory requirements and complexity than direct calculation in Eq. (3.41). When only a single frequency value is of interest, Goertzel algorithm is more efficient method for the task. Moreover, it can be used for computationally effective peaking of the maximum in Eq. (3.27).

### 3.6.2 Power Spectral Density

The power of a signal of the form

$$x_k[n] = A_k\cos\left(k\frac{2\pi}{N}n + \theta_k\right) \tag{3.42}$$

is

$$P_{\mathbf{y}_k} = \frac{1}{L}\|\mathbf{y}_k\|^2 = \frac{A_k^2}{2}. \tag{3.43}$$

This value is known as the power spectral density (PSD) at the frequency $\omega = k\frac{2\pi}{N}$. The corresponding squared magnitude values $A_k^2/2$ are known as the discrete-frequency periodogram (Eq. (3.28)), and this is the basic method for the PSD estimation of a signal. Plotting such a periodogram gives a frequency-domain representation of the signal's power distribution, highlighting which frequencies carry the most power.

DFT is energy conservation transform (Parseval's Theorem) that states the relation

$$\sum_{k=0}^{N-1} A_k^2 = \frac{1}{L}\|\mathbf{y}\|^2. \tag{3.44}$$

### 3.6.3 Spectral Spreading and Leakage

In an idealized setting, a pure cosine signal has a perfectly defined frequency representation. For instance, consider the discrete-time signal,

$$x[n] = A\cos\left(k_0\frac{2\pi}{L}n\right),\ k_0 \in \{1, \cdots, L-1\} \tag{3.45}$$

where $k_0$ is the frequency index. The Fourier transform of this signal yields a single spectral component at frequency $w_0 = k_0\frac{2\pi}{L}$, such that the spectral amplitude $A_k$ at each value of $k$ is given by

$$A_k = \begin{cases} \frac{A}{2} & k = k_0, N-k_0 \\ 0 & \text{otherwise} \end{cases}. \tag{3.46}$$

Under these conditions, the signal's spectral representation seems to be strictly localized at the specific frequency $\omega_k$, with no energy distributed elsewhere in the spectrum. However, practical scenarios deviate from this ideal case. In particular, if a denser frequency grid is employed (i.e. $N > L$) or the frequency varies continuously (as in Eq. (3.24)), the resulting spectral distribution can differ substantially from the discrete, single-peak ideal (Fig. 3.5). This difference arises because, in general, $\mathbf{X}(\omega)^T\mathbf{X}(\omega)$ is not orthogonal as in Eq. (3.40). As a result, two effects are introduced:

- The main frequency peak broadens, resulting in "spectral spreading".
- Additional frequency components emerge beyond the broadened main peak, termed "spectral leakage."

## 3.7 Summary

The summary of the presented approach is shown in Table 3.1. The presented approach involves a design of matrix $\mathbf{X}$ and using LS to estimate unknown parameters. The key addressed task are as follows.

Table 3.1: Comparison and summary of different signal estimation methods.

| Task | Parameters | Matrix $\mathbf{X}$ | SNR |
|---|---|---|---|
| Amplitude only | $A$ given $\omega_0$ | A single column of $\cos(\omega_0 n)$ | $\frac{\lVert\hat{\mathbf{y}}\rVert^2}{\lVert\mathbf{e}\rVert^2}$ |
| Amplitude & phase | $A, \theta$ given $\omega_0$ | Two columns of $\cos(\omega_0 n)$ and $\sin(\omega_0 n)$ | $\frac{\lVert\hat{\mathbf{y}}\rVert^2}{\lVert\mathbf{e}\rVert^2}$ |
| Frequency estimation | $\omega_0, A, \theta$ | Frequency-dependent $\cos(\omega n)$ and $\sin(\omega n)$ columns | Maximum of $\frac{\lVert\hat{\mathbf{y}}(\omega)\rVert^2}{\lVert\mathbf{e}(\omega)\rVert^2}$ |
| Fourier series (harmonic decomposition) | $A_0, \{A_m, \theta_m\}_{m=1}^{M}$, possibly $\omega_0$ | Harmonic cos/sin columns at multiples of $\omega_0$, $\cos(m\omega_0 n)$, $\sin(m\omega_0 n)$ | $\frac{\lVert\hat{\mathbf{y}}\rVert^2}{\lVert\mathbf{e}\rVert^2}$, can include frequency dependence if $\omega_0$ unknown |
| DFT | $\{A_k, \theta_k\}_{k=0}^{N-1}$ | Multiple pairs of columns $\cos\left(\frac{2\pi k}{N}n\right)$, $\sin\left(\frac{2\pi k}{N}n\right)$ for $k = 0, \ldots, N-1$ | Not used directly. Perfect reconstruction for $N \geq L$ |

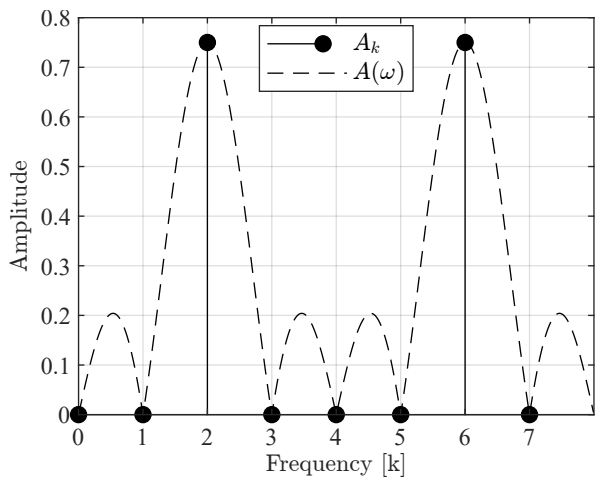



Figure 3.5: Illustration of a single-frequency cosine signal's spectrum under ideal assumptions (discrete, integer-multiple frequencies) versus practical conditions (denser frequency grid or non-integer frequencies). Note how the ideal single peak broadens and additional low-level components appear, highlighting the effects of spectral spreading and leakage.

**Amplitude Estimation** With a known frequency $\omega_0$, the amplitude $A$ is found via LS. The resulting residuals provide noise variance and SNR estimates.
**Amplitude and Phase Estimation:** For known $\omega_0$, rewriting

$$A\cos(\omega_0 n + \theta) = w_c \cos(\omega_0 n) + w_s \sin(\omega_0 n)$$

transforms the problem into a two-parameter LS regression.

**Frequency Estimation:** If $\omega_0$ is unknown, it is found by searching for the frequency that maximizes the fitted signal energy.

**Harmonic Signal Analysis:** Signals can be expressed as sums of multiple harmonics. Extending the LS approach to multiple harmonics allows estimation of each amplitude and phase. THD quantifies deviations from a pure tone.

**Discrete Fourier Transform (DFT):** The DFT is a special case of harmonic modeling, decomposing a signal into equally spaced frequency components. Efficiently computed by the FFT, the DFT is central to signal spectral analysis.
Although the estimators presented above have been extensively analyzed for the specific case of additive white Gaussian noise (AWGN) in the statistical signal processing literature [6, 5], conducting such an analysis requires a significantly more extensive mathematical framework. Furthermore, it is worth noting that any bias and variance in these estimators can be readily approximated via Monte Carlo simulations under various parameter settings and noise distributions.

# Appendices

## 3.A Single frequency analysis

### 3.A.1 Theory

- $\mathbf{X}^T\mathbf{X}$ analysis:

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} \mathbf{x}_c^T\mathbf{x}_c & \mathbf{x}_s^T\mathbf{x}_c \\ \mathbf{x}_s^T\mathbf{x}_c & \mathbf{x}_s^T\mathbf{x}_s \end{bmatrix} \tag{3.47}$$

with the following values

$$\begin{aligned}
\mathbf{x}_c^T\mathbf{x}_c &= \sum_{n=0}^{N-1}\cos^2\left(2\pi\frac{k}{N}n\right) = \frac{N}{2}\\
\mathbf{x}_c^T\mathbf{x}_s &= \sum_{n=0}^{N-1}\cos\left(2\pi\frac{k}{N}n\right)\sin\left(2\pi\frac{k}{N}n\right) = 0\\
&= \mathbf{x}_s^T\mathbf{x}_c\\
\mathbf{x}_s^T\mathbf{x}_s &= \sum_{n=0}^{N-1}\sin^2\left(2\pi\frac{k}{N}n\right) = \frac{N}{2}
\end{aligned} \tag{3.48}$$

- $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$ analysis: The resulting matrix

$$\left(\mathbf{X}^T\mathbf{X}\right)^{-1} = \begin{bmatrix} \frac{2}{N} & 0 \\ 0 & \frac{2}{N} \end{bmatrix} \tag{3.49}$$

- Finally, $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$ is

$$w_{c,k} = \frac{2}{N}\sum_{n=0}^{L-1} y[n]\cos\left(2\pi\frac{k}{N}n\right) \tag{3.50a}$$

$$w_{s,k} = \frac{2}{N}\sum_{n=0}^{L-1} y[n]\sin\left(2\pi\frac{k}{N}n\right) \tag{3.50b}$$

The orthogonality in more general form is given by

$$\sum_{n=0}^{N-1}\cos\left(2\pi\frac{j}{N}n\right)\cos\left(2\pi\frac{k}{N}n\right) = \frac{N}{2}\delta\left[j-k\right] \tag{3.51}$$

$$\sum_{n=0}^{N-1}\cos\left(2\pi\frac{k}{N}n\right)\sin\left(2\pi\frac{k}{N}n\right) = 0 \quad \forall j,k \tag{3.52}$$

$$\sum_{n=0}^{N-1}\sin\left(2\pi\frac{j}{N}n\right)\sin\left(2\pi\frac{k}{N}n\right) = \frac{N}{2}\delta\left[j-k\right], \tag{3.53}$$

### 3.1.2 Power

For a more general case of an arbitrary $\omega$ values, the signal of the form

$$y[n] = A\cos(\omega_0 n) \tag{3.54}$$

has the $\omega_0$-dependent power,

$$P_{\mathbf{y}} = \frac{A^2}{4L}\left(1 + 2L - \frac{\sin(\omega_0 - 2L\omega_0)}{\sin(\omega_0)}\right), \tag{3.55}$$

that results from the time-limited origin of the signal $y[n]$. For the infinite number of samples, the resulting power converges to a continuous-time power expression,

$$\lim_{L\to\infty} P_{\mathbf{y}} \to \frac{A^2}{2} \tag{3.56}$$

## 3.2 Takeaways

# Chapter 4

# ARMA Model

**Goal:** Linear prediction coefficients for systems-inspired models.

This chapter extends least-squares (LS)-based modeling to systems and time-series contexts, focusing on linear prediction coefficients derived from signals. Instead of restricting to a predefined parametric form like a sinusoid, here we consider arbitrary zero-mean finite-time signals $x[n]$ and $y[n], n = 0, \ldots, L-1$, in the presence of zero-mean noise, $\epsilon[n]$.

## 4.1 Auto-Correlation Function

**Goal:** Evaluate correlation between a signal and its time-shifted replicas to derive meaningful insights.

### 4.1.1 Linear Prediction & AR(1)

In system modeling and time-series analysis, a common approach is to express the current sample of a signal in terms of its past values. A simple first-order autoregressive (AR(1)) model predicts the current sample $x[n]$ from a single past sample $x[n-1]$, using the system model

$$x[n] = a_1 x[n-1] + \epsilon[n]. \tag{4.1}$$

and

$$\hat{x}[n] = a_1 x[n-1] \tag{4.2}$$

Here, $a_1$ is the linear prediction coefficient that indicates how strongly the previous sample $x[n-1]$ influences the current sample $x[n]$.
To determine $a_1$, minimization of SE loss function

$$\mathscr{L}(a_1) = \sum_n \left(x[n] - a_1 x[n-1]\right)\right)^2 \tag{4.3}$$

is used. Setting the derivative of $\mathscr{L}(a_1)$ with respect to $a_1$ to zero leads to

$$\frac{d\mathscr{L}(a)}{da} = 2\sum_n (x[n] - a_1 x[n-1])(-x[n-1]) = 0 \tag{4.4}$$

Finally, the LS solution for $a_1$ is

$$a_1 = \frac{\sum_n x[n]x[n-1]}{\sum_n x^2[n-1]}. \tag{4.5}$$

The value $a_1$ is termed as the (single) prediction coefficient of AR(1) model.

**Matrix formulation**

The coefficient may be formulated as a relation between two vectors

$$\underbrace{\begin{bmatrix} x[1] \\ x[2] \\ \vdots \\ x[L-2] \\ x[L-1] \end{bmatrix}}_{\hat{\mathbf{y}}} = a_1 \underbrace{\begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[L-3] \\ x[L-2] \end{bmatrix}}_{\mathbf{x}}, \tag{4.6}$$

or in a compresed form

$$\hat{\mathbf{y}} = a_1 \mathbf{x}. \tag{4.7}$$

The correspoding MMSE loss is

$$\mathscr{L}(a_1) = \|\mathbf{y} - a_1\mathbf{x}\|^2 \tag{4.8}$$

and its minimum is given by normal equation, (Eq. (2.18)),

$$a_1 = \left(\mathbf{x}^T\mathbf{x}\right)^{-1}\mathbf{x}^T\mathbf{y}. \tag{4.9}$$

The substitution results in the solution identical to Eq. (4.5).

### 4.1.2 Auto-correlation function (ACF)

To generalize beyond a single lag, we use one-coefficient prediction from the current sample $x[n]$ and its time-shifted version $x[n-k]$, using the $k$-step predictor,

$$\hat{x}[n] = a_k x[n-k]. \tag{4.10}$$

The corresponding solution is very similar to the solution for $k = 1$,

$$a_k = \frac{\sum_n x[n]x[n-k]}{\sum_n x^2[n-k]}. \tag{4.11}$$

The nominator of Eq. (4.11) is termed (raw, or unscaled) auto-correlation function (ACF) at lag $k$,

$$R_{\mathbf{xx}}[k] = \sum_n x[n]x[n-k], \quad k = 0, \ldots, L-1 \tag{4.12}$$

The ACF provides a measure of how linearly dependent the signal is on its shifted versions. Variants like biased, unbiased, and normalized ACF offer different normalization schemes to handle finite data length and scaling issues.

**Biased auto-correlation** Another useful definition is averaged sum of $y[n]y[n-k]$,

$$R_{\mathbf{xx},biased}[k] = \frac{1}{L}\sum_n x[n]x[n-k], \quad k = 0,\dots,L-1 \tag{4.13}$$

$$= \frac{1}{L}R_{\mathbf{xx}}[k] \tag{4.14}$$

is termed biased ACF.

**Normalized auto-correlation** Another version is normalized ACF of the form

$$R_{\mathbf{xx},norm}[k] = \frac{R_{\mathbf{xx}}[k]}{R_{\mathbf{xx}}[0]} \lesssim a_k. \tag{4.15}$$

Note, the difference between denominator above and the one in Eq. (4.11) is

$$\underbrace{\sum_n x^2[n]}_{R_{\mathbf{xx}}[0]} - \sum_n x^2[n-k] = \sum_{n<k} x^2[n] = x^2[0]+\cdots x^2[k-1] \tag{4.16}$$

Nevertheless, $a_k$ expression in Eq. (4.11) and the latter expression are approximately equal for a sufficiently high $L$. Moreover, the denominator is $k$-independent.

**Unbiased auto-correlation** Note, that the ACF includes summation only of $n-k$ terms. Another useful normalization is

$$\begin{aligned} R_{\mathbf{xx},unbiased}[k] &= \frac{1}{L-k}\sum_n x[n]x[n-k] \\ &= \frac{1}{L-k}R_{\mathbf{xx}}[k] \end{aligned} \tag{4.17}$$

This time it is assumed that

$$\begin{aligned} \frac{1}{L}\sum_n x^2[n] &\approx \frac{1}{L-k}\sum_n x^2[n-k] \\ \frac{x^2[0]+x^2[1]+\cdots x^2[L-1]}{L} &\approx \frac{x^2[k]+\cdots x^2[L-1]}{L-k} \end{aligned} \tag{4.18}$$

and the resulting expression

$$\begin{aligned} \frac{L}{L-k}\frac{R_{\mathbf{xx}}[k]}{R_{\mathbf{xx}}[0]} &= \frac{L}{L-k}R_{\mathbf{xx},norm}[k] \\ &= \frac{R_{\mathbf{xx},unbiased}[k]}{R_{\mathbf{xx},unbiased}[0]} \approx a_k \end{aligned} \tag{4.19}$$

is assumed to be closer approximation to $a_k$ than the normalized auto-correlation, for a relatively small values of $k$.

### 4.1.3 ACF Properties

The signal energy is given by

$$E_{\mathbf{x}} = \sum_n x^2[n] = R_{\mathbf{xx}}[0] \tag{4.20}$$

and it is also the higher value of ACF,

$$R_{\mathbf{xx}}[0] > R_{\mathbf{xx}}[k]. \tag{4.21}$$

Theoretically, $R_{\mathbf{xx}}[0] = R_{\mathbf{xx}}[k]$ may happen under certain conditions but unachievable for the practical time-limited signals.
The corresponding average power is given by

$$P_{\mathbf{x}} = \frac{1}{L}\sum_n x^2[n] = R_{\mathbf{xx},biased}[0] \tag{4.22}$$

The ACF has inherent time symmetry,

$$R_{\mathbf{xx}}[k] = R_{\mathbf{xx}}[-k] \tag{4.23}$$

**Correlation Coefficient Interpretation**

The resulting loss is given by (following Eq. (2.36) and the discussion above for $L \to \infty$)

$$\begin{aligned} \mathscr{L}_{min}(a_k) &= \sum_{n=0}^{L-1} x^2[n] - a_k \sum_{n=0}^{L-1} x[n]x[n-k] \\ &= R_{\mathbf{xx}}[0] - a_k R_{\mathbf{xx}}[k] \\ &\lesssim R_{\mathbf{xx}}[0]\left(1 - \left(\frac{R_{\mathbf{xx}}[k]}{R_{\mathbf{xx}}[0]}\right)^2\right) \\ &= R_{\mathbf{xx}}[0]\left(1 - R^2_{\mathbf{xx},norm}[k]\right) \\ &\approx R_{\mathbf{xx}}[0]\left(1 - \rho^2_{\mathbf{xx}}[k]\right) \end{aligned} \tag{4.24}$$

The value of $\rho_{\mathbf{xx}}[k]$ is termed *correlation coefficient* between $x[n]$ and $x[n-k]$,

$$\rho_{\mathbf{xx}}[k] \approx \frac{L}{L-k}R_{\mathbf{xx},norm}[k] \approx R_{\mathbf{xx},norm}[k]. \tag{4.25}$$

It is a measure of a linear dependence. It is bounded by

$$\left|\rho_{\mathbf{xx}}[k]\right| \le 1 \tag{4.26}$$

For noiseless data and a linear relation between the samples, the prediction is perfect, $\left|\rho_{\mathbf{xx}}[k]\right| = 1$ and $\mathscr{L}_{min} = 0$. On the other side, without any linear dependence, $\rho_{\mathbf{xx}}[k] = a_k = 0$. This can be summarized as

$$0 \le \mathscr{L}_{min}(a_k) \le \sum_{n=0}^{L-1} x^2[n] = R_{\mathbf{xx}}[0] \tag{4.27}$$

In matrix form (by Eq. (2.36)), the loss is given by

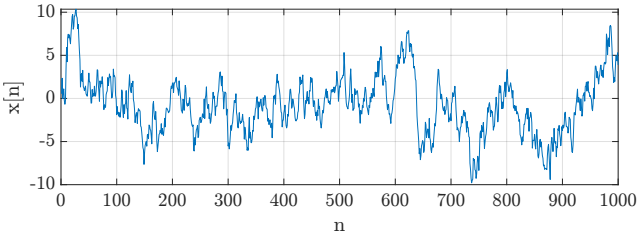
$$\mathscr{L}_{min}(a_k) = \mathbf{e}^T\mathbf{e} = \mathbf{y}^T\mathbf{y} - a_k\mathbf{y}^T\mathbf{x} \tag{4.28}$$

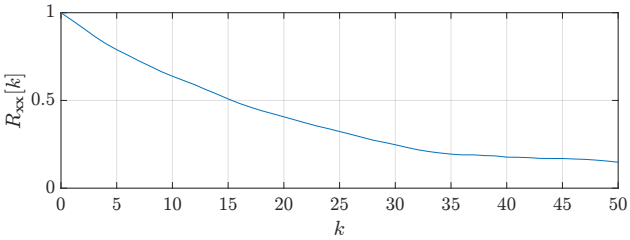
The illustration of the correlation coefficient principles is presented in Fig. 1.

**MSE and RMSE** The corresponding MSE and RMSE metrics are given by

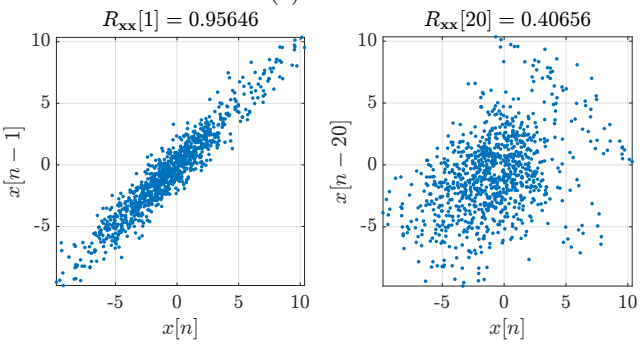
$$MSE(a_k) = \frac{1}{L}\mathscr{L}_{min}(a_k) \tag{4.29a}$$

$$RMSE(a_k) = \sqrt{\frac{1}{L}\mathscr{L}_{min}(a_k)} \tag{4.29b}$$

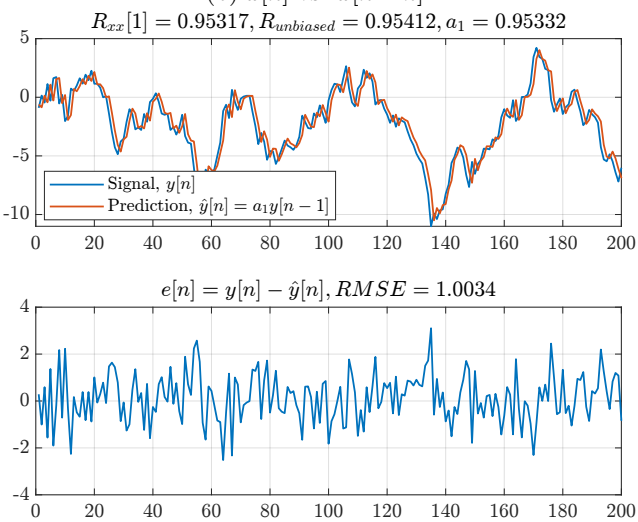



(a) ACF




(b) $x[n]$ vs. $x[n-k]$




(c) Prediction

Figure 1: Illustration of the linear dependence between $x[n]$ and $x[n-k]$.

**Correlation time** The correlation time, $k_c$ is the lag where $\rho_{\mathbf{xx}}[k_c]$ falls below a threshold,

$$\rho_{\mathbf{xx}}[k_c] = 0.5 \text{ or } 0.1 \text{ or } \exp(-1) \tag{4.30}$$

The predictability is assumed negligible for $k > k_c$,

$$\rho_{\mathbf{xx}}[k > k_c] \approx 0 \tag{4.31}$$

The decision threshold depends on the field of application. Note, correlation time is mostly only in physics and engineering models.

### 4.1.4 Confidence Interval

Another way to quantify the "importance" of the coefficients is to use confidence bound.

**General idea** The ACF estimate at a given lag is essentially a sample statistic derived from sums of products of random variables (the data values at different time points). Under typical assumptions of stationarity and weak dependence, these sample averages of random variables invoke the Central Limit Theorem (CLT). The CLT states that when you sum (or average) a sufficiently large number of independent or weakly dependent random variables with finite variance, the resulting distribution approaches a normal distribution, regardless of the variables' original distribution.
In other words, the estimation of the ACF involves averaging products like $x[n]x[n-k]$ across many time indices $n$. Provided the underlying process is stationary and not too strongly dependent, these averaged terms behave like sums of numerous random contributions. As the sample size $L$ grows large, the distribution of the ACF estimate at each lag approaches normality by virtue of the CLT. This is why the ACF at a given lag can be approximated as a normally distributed random variable in large-sample scenarios.

**Derivation** Since the estimated ACF at a given lag $k$ can be approximated as a normally distributed random variable with zero mean and variance approximately $\frac{1}{L}$ for large $L$, the 95% confidence bound is given by

$$R_{\mathbf{xx},norm}[k] \pm \frac{\sqrt{2}\,\mathrm{erf}(0.95)}{\sqrt{L}} \text{ or } R_{\mathbf{xx},norm}[k] \pm \frac{1.96}{\sqrt{L}}. \tag{4.32}$$

The value 1.96 comes from the properties of the standard normal distribution. In a standard normal distribution (mean 0, variance 1), approximately 95% of the area under the curve lies within $\pm 1.96$ standard deviations from the mean.

**Interpretation** If the estimated ACF value at a particular lag falls outside this range, it is statistically significant at the 95% confidence level (i.e., unlikely to be a result simply due to random chance). If it remains within the interval, it suggests that the observed correlation could be attributed to randomness in the data rather than a meaningful linear relationship.

### 4.1.5 Auto-covariance

For simplicity, a zero-average, $\bar{x}[n] = 0$, was assumed. When the signals is non-zero mean, the subtraction of signal average from the signal, $x[n] = x[n] - \bar{x}[n]$

before auto-correlation calculation is termed as auto-covariance.
Note, both auto-correlation and auto-covariance have similar abbreviation, ACF. Typically, ACF is used for auto-correlation, since majority of the signals are zero-mean.
**Tip:**

- ACF calculation may be significantly speed-up with appropriate algorithms and the bounded maximum value of $k \leq k_{max}$. The value of $k_{max}$ may be decided by Eq. (4.31).

### 4.1.6 Stationarity and Relation between ACF and PSD

**Stationarity definition** For stationary signals, statistical properties like mean, variance, and ACF remain constant over time segments. The direct outcomes are the the signal is without trend and with constant variance.

**Relation between ACF and PSD** From the signal processing point of view, the interpretation of a DFT of some general random signal is non-trivial. For example, phases $\theta_k$ of some random origin will be quite different for each time-segment (or realization) of the signal. The Wiener–Khinchin theorem states that the PSD (page 13, $|A_k|^2$ values) of stationary random signal is the Fourier transform of the ACF. Moreover, due to the symmetry (Eq. (4.23)) of $R_{\mathbf{xx}}[k]$ it results $\theta_k = 0 \forall k$. This relation shows, that random signal also includes spectral interpretation.

**Interpretation** The PSD shows where (in frequency) the signal has most of its energy. Peaks in the PSD correspond to frequencies at which the signal exhibits strong periodic or quasi-periodic components.
Slowly decaying (long-memory) correlations in the time domain often translate into a PSD that has more energy at low frequencies (indicating slow variations in time). Conversely, if the ACF shows periodicity, the PSD will have distinct peaks at the corresponding harmonic frequencies.
The ACF reveals how similar a signal is to itself at different time shifts. A slowly decaying ACF indicates strong long-term correlations, while a rapidly decaying ACF suggests only short-range predictability.

## 4.2 AR(p) Model

**Goal:** Extend $AR(1)$ to $AR(p)$ model.

The auto-regressive (AR) signal model describes a signal $y[n]$ as a linear combination of its $p$ previous samples plus noise. Formally, an $AR(p)$ model is

$$\begin{aligned} y[n] &= a_1 y[n-1] + a_2 y[n-2] + \cdots + a_p y[n-p] + \epsilon[n] \\ &= \sum_{m=1}^{p} a_m y[n-m] + \epsilon[n] \end{aligned} \tag{4.33}$$

and

$$\hat{y}[n] = \sum_{m=1}^{p} a_m y[n-m] \tag{4.34}$$

where $a_1, \ldots, a_p$ are the AR model coefficient and $p$ is the model order chosen as hyper-parameter.
This model can be easily formulated in the matrix form by using $L$ sample of $y[n]$,

$$\underbrace{\begin{bmatrix} \hat{y}[1] \\ \hat{y}[2] \\ \vdots \\ \hat{y}[L-1] \\ \hat{y}[L] \end{bmatrix}}_{\hat{\mathbf{y}}} = \underbrace{\begin{bmatrix} y[0] & 0 & 0 & \\ y[1] & y[0] & 0 & \vdots \\ y[2] & y[1] & y[0] & \\ y[3] & y[2] & y[1] & \\ \vdots & \vdots & \vdots & \vdots \\ y[L-2] & y[L-3] & y[L-2] & \\ y[L-1] & y[L-2] & y[L-3] & \end{bmatrix}}_{\mathbf{X}} \underbrace{\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix}}_{\mathbf{a}}, \tag{4.35}$$

where $\hat{\mathbf{y}} \in \mathscr{R}^{L-1}, X \in \mathscr{R}^{L-1 \times p}, \mathbf{a} \in \mathscr{R}^p$.
The AR coefficients $\mathbf{a}$ can be found by a LS regression that minimizes the MSE loss

$$\mathscr{L}(a_i) = \sum_n \left(y[n] - \hat{y}[n]\right)^2 = \|\mathbf{y} - \mathbf{X}\mathbf{a}\|^2. \tag{4.36}$$

The LS solution is straightforward,

$$\mathbf{a} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}. \tag{4.37}$$

The corresponding dimensions are $\mathbf{X} \in \mathscr{R}^{L \times p}, \mathbf{a} \in \mathscr{R}^p$.
Note, the practical approach for evaluation of $\mathbf{a}$ is presented in the following section.

### 4.2.1 Yule-Walker Form

The resulting $\mathbf{X}^T\mathbf{X}$ matrix is termed (Toeplitz) auto-correlation matrix and can be interpreted in terms of auto-correlation values,

$$\begin{aligned} \mathbf{R} &= \mathbf{X}^T\mathbf{X} \\ &= \begin{bmatrix} R_{\mathbf{yy}}[0] & R_{\mathbf{yy}}[1] & \cdots & R_{\mathbf{yy}}[p-1] \\ R_{\mathbf{yy}}[1] & R_{\mathbf{yy}}[0] & \cdots & R_{\mathbf{yy}}[p-2] \\ \vdots & \vdots & \ddots & \vdots \\ R_{\mathbf{yy}}[p-1] & R_{\mathbf{yy}}[p-2] & \cdots & R_{\mathbf{yy}}[0] \end{bmatrix}, \end{aligned} \tag{4.38}$$

where unscaled $R_{\mathbf{yy}}[k]$ is defined above. It is common practice to ignore the changes in the vector lengths in calculating ACF for the same time-shift for $p \ll L$, e.g. diagonal matrix values are

$$R_{\mathbf{yy}}[0] = \sum_{n=0}^{L-1} y^2[n] \approx \sum_{n=0}^{L-\mathbf{2}} y^2[n] \tag{4.39}$$

The vector $\mathbf{X}^T\mathbf{y}$ is also comprised of the corresponding $R_{\mathbf{xx}}[k]$ values,

$$\mathbf{r} = \mathbf{X}^T\mathbf{y} = \begin{bmatrix} R_{\mathbf{yy}}[1] \\ R_{\mathbf{yy}}[2] \\ \vdots \\ R_{\mathbf{yy}}[p] \end{bmatrix} \tag{4.40}$$

Using these formulations, the resulting coefficients are given by a solution of a set of linear equations,

$$\mathbf{R}\mathbf{a} = \mathbf{r} \tag{4.41}$$

and the result is similar to Eq. (4.37),

$$\mathbf{a} = \mathbf{R}^{-1}\mathbf{r} \tag{4.42}$$

**Squared error** The resulting loss (squared error) is given by (following Eqs. (2.36) and (4.24))

$$\begin{aligned} \mathscr{L}_{min} &= \sum_{n=0}^{L-1} x^2[n] - \sum_{k=1}^{p} a_k \sum_{n=0}^{L-1} x[n]x[n-k] \\ &= R_{\mathbf{xx}}[0] - \sum_{k=1}^{p} a_k R_{\mathbf{xx}}[k] \end{aligned} \tag{4.43}$$

Theoretically, higher value of $p$ results in lower loss in the presence of the sufficiently long correlation time (Eq. (4.30)). Note, the accuracy drops for high values of $p$ to due to reduced accuracy of $R_{\mathbf{xx}}[p]$.

**Example 4.1**: Learn linear prediction of $y[8]$ for $p = 2$ and signal $y[0], y[1], \ldots, y[7]$.

*Solution*:

$$\underbrace{\begin{bmatrix} y[0] & 0 & 0 \\ y[1] & y[0] & 0 \\ y[2] & y[1] & y[0] \\ y[3] & y[2] & y[1] \\ y[4] & y[3] & y[2] \\ y[5] & y[4] & y[3] \\ y[6] & y[5] & y[4] \end{bmatrix}}_{\mathbf{X}} \underbrace{\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}}_{\mathbf{a}} = \underbrace{\begin{bmatrix} y[1] \\ y[2] \\ y[3] \\ y[4] \\ y[5] \\ y[6] \\ y[7] \end{bmatrix}}_{\mathbf{y}} \tag{4.44}$$

Finding vector $\mathbf{a}$ values is (almost) trivial by a minimum of the loss function of the form

$$\mathscr{L} = \|\mathbf{y} - \mathbf{X}\mathbf{a}\|^2. \tag{4.45}$$

Once found, $\hat{y}[8] = a_1 y[7] + a_2 y[6] + a_3 y[5]$.

> The commonly adopted notation of the signal model in most of the books and software packages is
>
> $$\sum_{m=0}^{p} \tilde{a}_m y[n-m] = \epsilon[n] \tag{4.46}$$
>
> with $\tilde{a}_0 = 1$ and $\tilde{a}_m = -a_m$.

> This definition follows discrete-time definition and the corresponding Z-transform of all-pole system. It also can be directly applied as a denominator coefficients for filter command.

**Biased signal** The presence of a non-zero average (bias) in the signal modifies the AR model formulation. Instead of assuming a zero-mean process, a constant bias term $\mu$ is introduced,

$$\hat{y}[n] = \mu + a_1 y[n-1] + a_2 y[n-2] + \cdots + a_p y[n-p] + \epsilon[n] \tag{4.47}$$

In matrix form, this requires adding a column of ones $\mathbf{1}_M$ to the data matrix $\mathbf{X}$, similarly to how it is done in multivariate LS (Eq.(2.19)).

**Tips:**

- For computational and memory efficiency, the Levinson-Durbin algorithm $O\left(L^2\right)$ is often used to solve for AR coefficients [1], instead of direct $O\left(L^3\right)$.
- The AR parameters $\mathbf{a}$ are also referred to as linear prediction coefficients (LPC), emphasizing their role in predicting the current value from past samples.
- The auto-correlation matrix $\mathbf{R}$ is guaranteed to be positive-definite (and, therefore, invertible) for ordinary or biased definitions (normalization constants reduce each other in (4.37)).
- Theoretically, the AR model coefficients can be regularized to prevent overfitting.
- Although this section focuses on linear AR models, nonlinear variants also exist, allowing modeling of more complex dynamics.
- Note, there are other methods for LPC calculation, such as Burg's method, as it is done in `librosa.lpc`.

### 4.2.2 Moving Average

A simple special case of an AR-like model is the moving average filter, where all coefficients are equal and sum to one, $a_i = \frac{1}{p}$. In this case,

$$\begin{aligned} \hat{y}[n] &= \frac{1}{p}\left(y[n-1] + \cdots + y[n-p]\right) \\ &= \frac{1}{p}\sum_{m=1}^{p} y[n-m] \end{aligned} \tag{4.48}$$

This is not strictly an AR model but shares the idea of using past samples. It smooths the signal by averaging the most recent $p$ values.

[1] For example, `solve_toeplitz`

### 4.2.3 Nearest Neighbor (Naïve)

A simple baseline is to use the immediate past sample as the prediction with $a_1 = 1$,

$$\hat{y}[n] = y[n-1] \tag{4.49}$$

This "1-nearest neighbor" approach ignores signal dynamics and history. While often not very accurate, it serves as a useful baseline to compare against more sophisticated models.

### 4.2.4 Time-Domain Filtering

If we have two versions of a signal: one clean and another noisy,

$$\begin{cases} y[n] \\ \tilde{y}[n] = y[n] + \epsilon[n] \end{cases} \tag{4.50}$$

we can use AR modeling to filter or "denoise" the noisy version. The key idea is to construct the AR matrix $\mathbf{X}$ in Eq. (4.35) from the shifted versions of $\tilde{y}[n]$ rather than $y[n]$. By fitting an AR model the resulting coefficients effectively learn how to reconstruct the clean signal from the noisy input.
This approach sets the foundation for adaptive filtering methods, where the model continuously adjusts its parameters to best estimate the clean signal under changing noise conditions.

**Example 4.2**: Evaluate the filter that removes a time-shifted and attenuation replica of the signal (echo) of the form

$$\tilde{y}[n] = y[n] + \alpha y[n-n_0] \tag{4.51}$$

which means the signal is contaminated by a delayed and attenuated version of itself (an echo). The model tries to learn the relationship

$$\hat{y}[n] = a_1\tilde{y}[n-1] + a_2\tilde{y}[n-2] + \cdots + a_p\tilde{y}[n-p] \tag{4.52}$$

With a suitably chosen model order $p$ that includes the lag $n_0$, the model's estimated coefficients can help isolate and remove the echo term from the observed signal, effectively filtering it out.

## 4.3 Linear Prediction of Sinusoidal Signal

**Goal:** This section demonstrates that a second-order auto-regressive (AR(2)) model can represent a pure sinusoidal signal perfectly, predicting it without error from just the two previous samples.

Consider the noise-free sinusoidal signal

$$y[n] = \cos(\omega_0 n) \tag{4.53}$$

For simplicity, phase is initially taken as zero. We aim to model it with

$$\hat{y}[n+1] = a_0 y[n] + a_1 y[n-1] \tag{4.54}$$

The corresponding loss is

$$\begin{aligned} \mathscr{L}(a_0, a_1) &= \sum_n \big(y[n+1] - \hat{y}[n+1]\big)^2 \\ &= \sum_n \big(y[n+1] - a_0 y[n] - a_1 y[n-1]\big)^2 \end{aligned} \tag{4.55}$$

The required minimum is given by a solution of a system of normal equations,

$$\begin{cases} \dfrac{\partial}{\partial a_0}\mathscr{L}(a_0, a_1) = 0 \\ \dfrac{\partial}{\partial a_1}\mathscr{L}(a_0, a_1) = 0 \end{cases} \tag{4.56}$$

The resulting equations are

$$\begin{cases} 2\sum_n \big(y[n+1] - a_0 y[n] - a_1 y[n-1]\big)\cdot(-y[n]) = 0 \\ 2\sum_n \big(y[n+1] - a_0 y[n] - a_1 y[n-1]\big)\cdot(-y[n-1]) = 0 \end{cases} \tag{4.57}$$

These equations involve sums of trigonometrical functions. Using standard trigonometric identities and assuming a large number of samples $L$, these sums simplify due to the oscillatory nature of sine and cosine functions. Some important quantities are

$$\begin{aligned} y[n+1]y[n] &= \cos\big(\omega_0[n]\big)\cos\big(\omega_0[n+1]\big) \\ &= \cos(\omega_0 n)\left[\cos(\omega_0)\cos(\omega_0 n) - \sin(\omega_0)\sin(\omega_0 n)\right] \\ &= \cos(\omega_0)\cos^2(\omega_0 n) - \sin(\omega_0)\cos(\omega_0 n)\sin(\omega_0 n) \end{aligned} \tag{4.58}$$

For $1/\omega_0 \ll L$, the common assumption is

$$\sum_n \cos^2(\omega_0 n) \approx \frac{L}{2}$$

$$\sum_n \cos(\omega_0 n)\sin(\omega_0 n) \approx 0$$

$$\begin{aligned} \sum_n y[n+1]y[n] &\approx \frac{L}{2}\cos(\omega_0) \\ &\approx \sum_n y[n]y[n-1] \end{aligned}$$

$$\begin{aligned} \sum_n y^2[n] &= \sum_n \cos^2(\omega_0 n) \\ &\approx \sum_n y^2[n-1] \approx \frac{L}{2} \end{aligned}$$

$$\begin{aligned} \sum_n y[n+1]y[n-1] &= \frac{1}{2}\sum_n \cos(2\omega_0 n) + \frac{1}{2}\sum_n \cos(2\omega_0) \\ &\approx \frac{L}{2}\cos(2\omega_0) \end{aligned}$$

The approximations show that sum terms reduce to manageable forms. Substituting these approximations into the normal equations leads to a system

$$\begin{aligned} a_0\frac{L}{2} + a_1\frac{L}{2}\cos(\omega_0) &= \frac{L}{2}\cos(\omega_0) \\ a_0\frac{L}{2}\cos(\omega_0) + a_1\frac{L}{2} &= \frac{L}{2}\cos(2\omega_0) \end{aligned} \tag{4.59}$$

Solving these two linear equations for prediction coefficients yields

$$\begin{aligned} a_0 &= 2\cos(\omega_0) \\ a_1 &= -1 \end{aligned} \tag{4.60}$$

This result is exact in the idealized scenario of an infinite, noise-free sinusoid.
f the signal had a phase $\theta$,

$$y[n] = \cos(\omega_0 n + \theta) \tag{4.61}$$

the same coefficients still apply, only the initial conditions (the first two samples) differ,

$$\begin{aligned} y[0] &= \cos(\theta) \\ y[1] &= \cos(\omega_0 + \theta). \end{aligned} \tag{4.62}$$

The AR(2) model still perfectly represents the sinusoidal sequence. The example of the resulting signal is presented in Fig. 2.

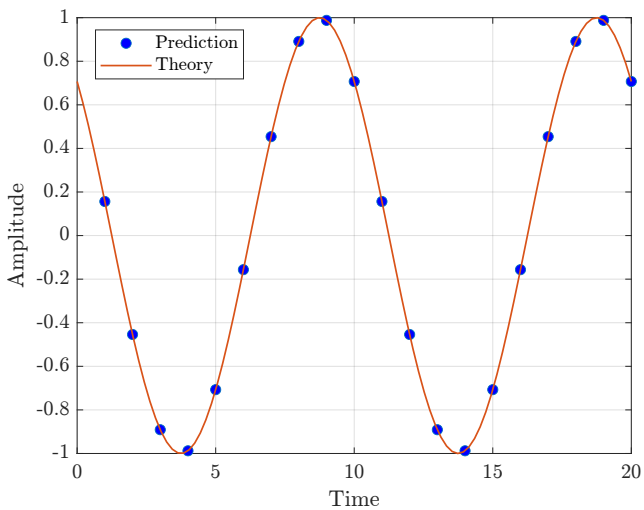



Figure 2: Oscillator example.

**Results interpretation:**

- Any complex signal can be thought of as a sum of sinusoids. In theory, each sinusoidal component can be perfectly modeled by an AR(2) process corresponding to its frequency.
- In practice, signals are finite in length and often corrupted by noise. Under these realistic conditions, perfect prediction is not achievable, and the AR model's accuracy diminishes as noise and non-idealities increase. The prediction horizon is limited even for a periodic low-noise signals.
- Nevertheless, this analysis provides a foundational insight: sinusoids have a natural AR(p) representation, explaining why AR(p) models are often effective in capturing periodic components of signals for a small-step prediction.

## 4.4 Partial auto-correlation function

**Goal:** The partial autocorrelation function (PACF) measures the correlation between a time series and its lagged values after removing the influence of all shorter lags. In other words, the PACF at lag $k$ shows the direct effect of $y[n-k]$ on $y[n]$, excluding any intermediary correlations through lags less than $k$.

The partial autocorrelation at $k$ is the correlation that results after removing the effect of any correlations due to the terms at shorter lags,

$$\underline{x[0]}, \underbrace{x[1], x[2], \ldots, x[j-1]}_{\text{partial out}}, \underline{x[j]}, x[j+1], \ldots$$

The PACF at lag $k$ can be extracted by fitting an AR(k) model and observing the coefficient associated with $y[n-k]$ in this model[2]. For each $k$,

$$\hat{y}_k[n] = \phi_{k,1}y[n-1] + \phi_{k,2}y[n-2] + \ldots + \phi_{k,k}y[n-k] + \epsilon[n] \tag{4.63}$$

The $k$-th partial autocorrelation, $\beta[k]$ is $\phi_{k,k}$, the coefficient of $y[n-k]$ in the AR(k) model. Each of these AR(k) models can be solved using standard LS methods. The algorithm is as follows:

- For the first value of PACF, $\beta[1]$, fit AR(1) model

$$\hat{y}_1[n] = \phi_{1,1}y[n-1] + \epsilon[n] \tag{4.64}$$

  and the coefficient $\beta[1] = \hat{\phi}_{1,1}$ is given by the model solution, $a_1$ (Eq. (4.5)).
- For the second order PACF, it would be the $\beta[2] = \phi_{2,2}$ coefficient of AR(2) model,

$$\hat{y}_2[n] = \phi_{2,1}y[n-1] + \phi_{2,2}y[n-2] + \epsilon[n] \tag{4.65}$$

- Continue this process up to the desired lag, each time extracting the coefficient of the highest-order lag as the PACF value.

Tip:

- While the ACF reflects both direct and indirect correlations (where an earlier lag may influence a later lag through intermediate values), the PACF isolates the direct contribution of each individual lag once all shorter-term effects are factored out.
- Though conceptually the PACF is obtained by fitting multiple AR models, practically more efficient algorithms like the Levinson-Durbin recursion can compute these values quickly and without having to solve a new LS problem at every step.

[2] Stackexchange

### 4.4.1 Relation between PACF and AR(p)

The order $p$ of AR(p) model is related to the statistically significant (over a confidence bound) coefficients of PACF. Observing where the PACF cuts off helps identify the appropriate order $p$ for AR(p) model fitting. An example of a synthetic signal analysis with $p = 2$ is presented in Fig. 3.

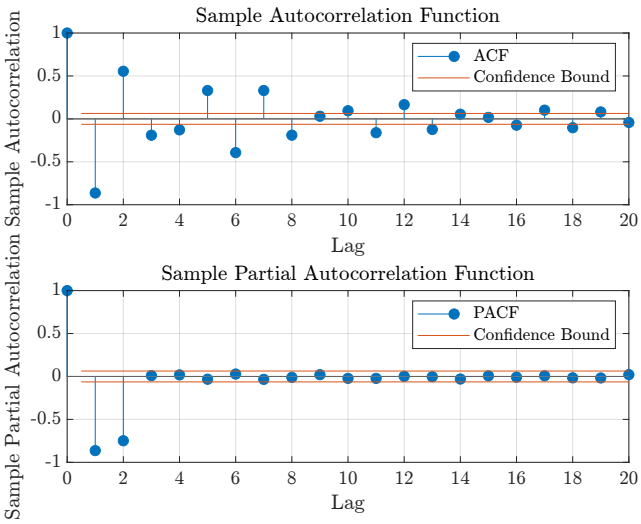



Figure 3: ACF and PACF of AR(2) signal. Note short PACF and long ACF plots.

## 4.5 MA model

**Goal:** The moving average (MA) model expresses the current output $y[n]$ as a linear combination of current and past noise terms. Unlike the AR model, which relies on past values of the output, the MA model directly models the output as filtered white noise.

Another model is the moving average model (MA), where the output is a linear combination of the noise values at the different times [6, Example 4.3, pp. 90]

$$y[n] = \epsilon[n] + b_1\epsilon[n-1] + \cdots + b_q\epsilon[n-q] \tag{4.66}$$

Because the noise terms $\epsilon[n]$ are not directly observable (unlike the past outputs in AR modeling), deriving a closed-form solution for MA parameters through simple linear regression is not as direct and is not presented here. [3] While the underlying theory is well-established, practical estimation of MA parameters often involves advanced numerical methods rather than a simple closed-form solution.

### 4.5.1 MA and AR relations

There exists a duality between AR and MA processes in terms of infinite expansions:

[3] e.g. A simple Estimator of an MA(1) Models.

**Presenting AR(p) as MA($\infty$)** An AR model can be represented as an infinite MA model when the autoregressive parameters are stable. Consider the recursive simple AR(1),

$$\begin{aligned} y[n] &= a_1 y[n-1] + \epsilon[n] \\ &= a_1(a_1 y[n-2] + \epsilon[n-1]) + \epsilon[n] \\ &= a_1^2 y[n-2] + a_1\epsilon[n-1] + \epsilon[n] \\ &= a_1^3 y[n-3] + a_1^2\epsilon[n-2] + a_1\epsilon[n-1] + \epsilon[n] \end{aligned} \tag{4.67}$$

Continuing this process indefinitely and assuming stability, $a_1 < 1$, $\lim_{k\to\infty} a_1^k \to 0$, we have

$$\begin{aligned} y[n] &= \epsilon[n] + a_1\epsilon[n-1] + a_1^2\epsilon[n-2] + a_1^3\epsilon[n-3] + \cdots \\ &= \sum_{i=0}^{\infty} a_1^i \epsilon[n-i] \end{aligned} \tag{4.68}$$

that it is MA($\infty$) model, where the coefficients of the MA representation are the infinite geometric sequence powers of $a_1$.

**Presenting MA(1) as AR($\infty$)** An MA(1) process is given by

$$x[n] = \epsilon[n] + b_1\,\epsilon[n-1]. \tag{4.69}$$

We can express the process as an infinite-order AR($\infty$) model.
First, note that we can write

$$\epsilon[n] = x[n] - b_1\,\epsilon[n-1]. \tag{4.70}$$

Substituting for $\epsilon[n-1]$ recursively, we have

$$\begin{aligned} \epsilon[n] &= x[n] - b_1\left(x[n-1] - b_1\,\epsilon[n-2]\right) \\ &= x[n] - b_1\,x[n-1] + b_1^2\,\epsilon[n-2] \\ &= x[n] - b_1\,x[n-1] + b_1^2\left(x[n-2] - b_1\,\epsilon[n-3]\right) \\ &= x[n] - b_1\,x[n-1] + b_1^2\,x[n-2] - b_1^3\,\epsilon[n-3] \\ &\vdots \\ &= \sum_{i=0}^{\infty}(-b_1)^i\,x[n-i]. \end{aligned} \tag{4.71}$$

Rearranging, the ARAR($\infty$) representation becomes

$$x[n] = \epsilon[n] + \sum_{i=1}^{\infty}(-b_1)^i\,x[n-i]. \tag{4.72}$$

Note the invertibility condition, $|b_1| < 1$.

**Interpretation** AR and MA are not mutually exclusive categories. A stable AR process can be seen as a special case of an infinite MA, and a stable MA can be thought of as an infinite AR.

### 4.5.2 The relation between MA(q) and ACF

The MA(q) model has a distinctive fingerprint in terms of its ACF; it is nonzero for up to lag $q$ and essentially zero afterward (except for sampling and noise effects). Just as the partial autocorrelation function (PACF) helps determine the order $p$ of an AR(p) process by pinpointing where its PACF cuts off (Sec. 4.4.1), the ACF helps identify the order $q$ of an MA(q) process.
An example of a synthetic MA(4) signal analysis is presented in Fig. 4. After lag 4, the ACF values remain within the confidence bounds, essentially zero, suggesting the data arise from an MA(4) process.

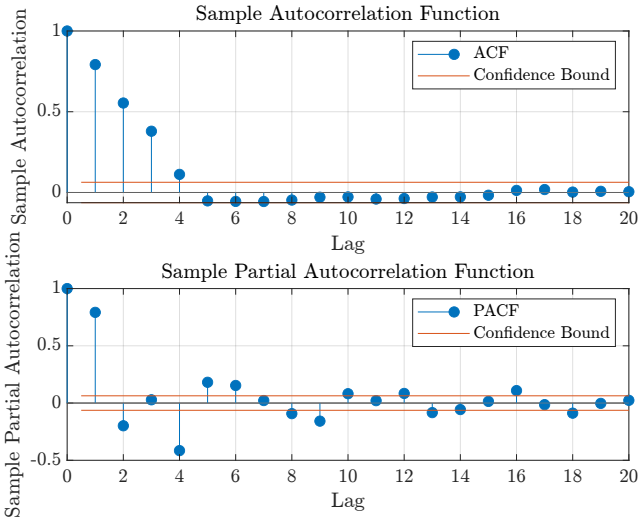



Figure 4: ACF and PACF of a synthetic MA(4) signal. Note short ACF and long PACF plots.

## 4.6 ARMA

**Goal:** The ARMA model extends the concepts of AR and MA models by combining their elements to capture more complex dynamics.

The ARMA(p,q) model is given by

$$\begin{aligned}\hat{y}[n] &= a_1 y[n-1] + \cdots + a_p y[n-p] \\ &\quad + b_1 \epsilon[n-q] + \cdots + b_q \epsilon[n-q] + \epsilon[n] \\ &= \sum_{i=1}^{p} a_i y[n-i] + \sum_{k=0}^{q} b_k \epsilon[n-k]\end{aligned}, \tag{4.73}$$

where $b_0 = 1$ by definition.
One of the ways to describe MA part of ARMA, is that MA uses to model the difference unexplained by AR model,

$$\hat{y}[n] - \sum_{i=1}^{p} a_i y[n-i] = \sum_{k=0}^{q} b_k \epsilon[n-k] \tag{4.74}$$

As with AR models (Eq. (4.47)), a constant bias $\mu$ can be included,

$$\hat{y}[n] = \sum_{i=1}^{p} a_i y[n-i] + \sum_{k=0}^{q} b_k \epsilon[n-k] + \mu \tag{4.75}$$

# Chapter 5

# Models Characterization and Tuning

The goal is:

- Estimate some metrics of the model and understand the limitations on this estimate.
- Trial and error approach.
- Understand the limitations on the effectiveness of the model's ability to make inferences on unfamiliar data.

## 5.1 Generalization

Without loss of generality, and for simplicity, uni-variate definition is used. As we already stated, our data is a set of $(x_k, y_k)$ pairs. Let's assume that the dataset $(\mathbf{x}, \mathbf{y})$ are $M$ samples drawn from some (unknown) joint probability distribution, $(x, y) \sim \mathscr{D}$. In practice, the value of $M$ is (very) limited.
When we use some data to "train" some model, $f(\mathbf{X}; \mathbf{w})$, the question is, how can we assess the performance on other points from $\mathscr{D}$.
**Generalization**: The difference between performance metrics over data that is used to train model (train performance) and performance metric over all (theoretically) possible points from $\mathscr{D}$ (generalization error),

$$p_{gen} = E_{\mathscr{D}}[J(y, \hat{y})]. \tag{5.1}$$

The problem is that the distribution $\mathscr{D}$ is unknown in most of the practical applications. In the following, method to approximate the generalization probability will be discussed.
The LS example performance is presented in Fig. 1.
Notes:

- Better generalization $\Rightarrow$ smaller difference between model performance and generalization performance.
- Can be evaluated theoretically only for very simple models and datasets.
- Require some practical assessment tools, as follows.

## 5.2 Polynomial model

Goal:

- Extension of a linear model "engine" to polynomial models. The polynomial model is very flexible, i.e. due to the Taylor expansion theorem.
- Illustration of generalization principle.

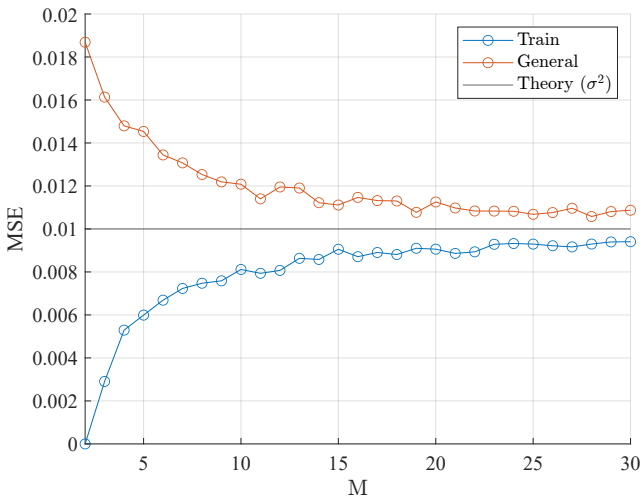



Figure 1: The difference between the results in Fig. 2.2 and the "realistic" performance.

The $N$-degree uni-variate polynomial model is

$$\begin{aligned}\hat{y} = f(x; \mathbf{w}) &= w_0 + w_1 x + w_2 x^2 + \cdots + w_N x^N \\ &= \sum_{j=0}^{N} w_j x^j\end{aligned} \tag{5.2}$$

The corresponding prediction is

$$\hat{y_k} = \sum_{j=0}^{N} w_j x_k^j \tag{5.3}$$

This problem is linear by change of variables, $z_{kj} = x_k^j$,

$$\hat{y_k} = \sum_{j=0}^{N} w_j z_{kj} \tag{5.4}$$

Using matrix notation (also termed Vandermonde matrix),

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^N \\ 1 & x_2 & x_2^2 & \cdots & x_2^N \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_M & x_M^2 & \cdots & x_M^N \end{bmatrix} \tag{5.5}$$

the weights values are straightforward.

## 5.3 Overfitting and underfitting

Goal: Two common and fundamental problems in machine learning.

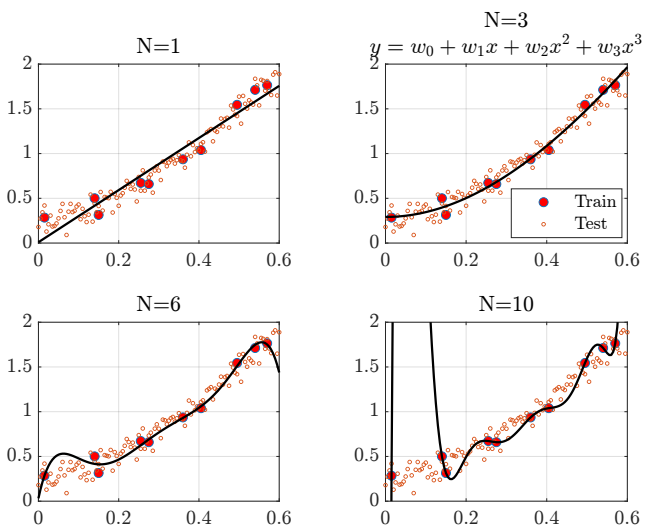
**Overfitting** when model is too complex, i.e. have too many parameters.

- Too many parameters relative to the number of observations. Ideally, we want an order of magnitude more observations than parameters.
- Follow the training data very closely.
- Fail to generalize well to unseen data.

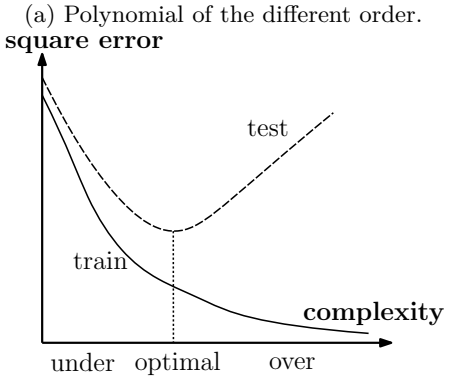
**Underfitting** happens when a model is too simple.

- Unable to capture the underlying pattern of the data and hence misses the trends in the data.
- Performs poorly on the training data and fail to generalize.

Overfitting and underfitting are complimentary and balancing between them is key to building robust machine learning models that perform well on new, unseen data, i.e. generalize well.




(a) Polynomial of the different order.




(b)

Figure 2: (a) Overfitting and underfitting polynomial example. (b) Increasing model complexity improves the prediction error for training data, but from a certain point of transitioning from underfitting to overfitting, it increases the error for test data.

**Hyper-parameter optimization** The order $N$ is the hyper-parameter of the polynomial model. Selecting the most appropriate hyper-parameters value is called hyper-parameter optimization.

## 5.4 Cross-validation

Goal: Trial and error approach to quantify generalization performance and overfitting-underfitting balance.

The cross-validation is also termed performance assessment.

- First step of any following technique is resample the dataset into the **random order**.

### Big dataset (tens of thousands): train/validation/test

Split into three *distinctive* datasets:

- *Training* (50-80%): used for learning of model parameters, e.g. weights $\mathbf{w}$.
- *Validation* (10-25%): used for assessment of model hyper-parameters influence.
- *Test* (10-25%): performance assessment that is supposed to be an estimation for the generalization error.

### Medium datasets (hundreds to thousandths): k-fold

Steps:

- *Data Splitting*: First, the available dataset is divided into $k$ subsets of approximately equal size. These subsets are often referred to as "folds."
- *Model Training and Evaluation*: The model is trained k times. In each iteration, one of the subsets is used as the test set, and the remaining $k-1$ subsets are used as the training/validation sets. This means that in each iteration, the model is trained and validated on a different combination of training and test data.
- *Performance Evaluation*: After training the model $k$ times, the performance of the model is evaluated by averaging the performance metrics obtained in each iteration.

Usually, $k$ is defaulted to 5 or 10.

**Very small datasets (tens to hundreds): one-hold-out**

Uses $k$-fold with $k = M$, which means that each fold will contain only one data point.

### 5.4.1 Summary

Model performance insights from the differences between train and test datasets and undefitting-overfitting trade-off are presented in Fig. 2(a).

## 5.5 Noisy deterministic function interpretation and bias-variance trade-off

The general model beneath the dataset model is

$$y = h(\mathbf{x}) + \epsilon, \tag{5.6}$$

where $f(\mathbf{x})$ is some unknown function and $\epsilon$ is some random noise with unknown distribution and with $E[\epsilon] = 0, \mathrm{Var}[\epsilon] = \sigma^2$.
Prediction is by $\hat{y}_i = f(\mathbf{x}_i; \boldsymbol{\theta})$ and the resulting mse loss of the model is

$$\begin{aligned}
\mathscr{L} &= E\left[(\hat{y} - y)^2\right] = \frac{1}{M}\sum_{i=1}^{M}(\hat{y}_i - y_i)^2 \\
&= E\left[\hat{y}^2\right] - 2E[y]\,E[\hat{y}] + E\left[y^2\right] \\
E\left[\hat{y}^2\right] &= \mathrm{Var}[\hat{y}] + E^2[\hat{y}] \\
E[y] &= E\left[h(\mathbf{x})\right] = h(\mathbf{x}) \\
E\left[y^2\right] &= E\left[(h(\mathbf{x}) + \epsilon)^2\right] = \frac{1}{M}\sum_{i=1}^{M}\left(h(\mathbf{x}_i) + \epsilon_i\right)^2 \\
&= E\left[h^2(\mathbf{x})\right] + 2E\left[h(\mathbf{x})\right]\cancelto{0}{E[\epsilon]} + E\left[\epsilon^2\right] \\
&= E\left[h^2(\mathbf{x})\right] + \sigma^2 \\
&= h^2(\mathbf{x}) + \sigma^2 \\
\mathscr{L} &= \mathrm{Var}[\hat{y}] + E^2[\hat{y}] - 2h(\mathbf{x})E[\hat{y}] + h^2(\mathbf{x}) + \sigma^2 \\
&= \underbrace{\left(E[\hat{y}] - h(\mathbf{x})\right)^2}_{bias^2} + \underbrace{\mathrm{Var}[\hat{y}]}_{variance} + \underbrace{\sigma^2}_{noise}
\end{aligned} \tag{5.7}$$

The visualization of this principle for polynomial regression is presented in Fig. 3. Underfitting is low variance and high bias, and overfitting is high variance and low bias.

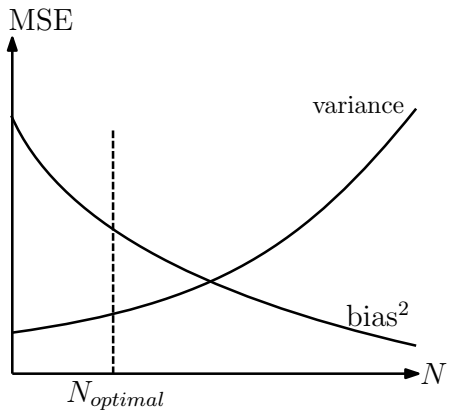



Figure 3: Bias-variance trade-off for polynomial regression.

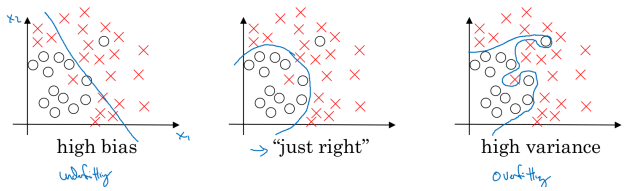



Figure 4: Overfitting and underfitting at classification bounds.

## 5.6 Takeaways

# Chapter 6

# Overfitting Management

**Goal:** Typically, the most applied models (beyond the basic linear one) in their general form are overfit data. The goal is to provide the list of the methods that are used to manage the overfit.

## 6.1 Dataset size

**Goal:** Dataset level in Fig. 1.2.

Adding data usually improves the performance for an overly complex system as presented in Fig. 1.

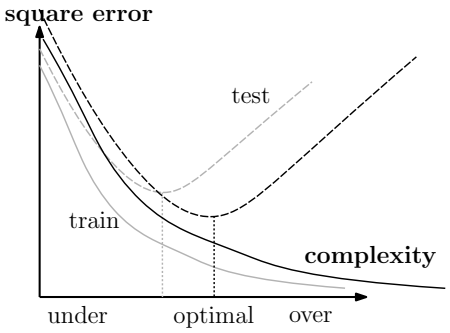



Figure 1: Bias-variance trade-off for polynomial regression.

## 6.2 Regularization

**Goal:**
- Tweak the bias-variance trade-off by penalizing weights size.
- Loss function level in Fig. 1.2.
- Introduces new hyper-parameter that have to be tuned.

**Regularization**: Penalty to the loss function

$$\mathscr{L}_{\text{new}} = \mathscr{L} + \lambda g(\mathbf{w}) \tag{6.1}$$

$\lambda$ is termed *regularization parameter*.
$L_1$**-regularization**: Special case of $g(\cdot)$, where

$$g(\mathbf{w}) = \frac{\lambda}{2M}\|\mathbf{w}\|_1 = \frac{\lambda}{2M}\sum_{i=1}^{N}|w_i| \tag{6.2}$$

$L_2$**-regularization**: Special case of $g(\mathbf{w}) = \frac{1}{2M}\|\mathbf{w}\|^2$,

$$\mathscr{L}_{\text{new}} = \mathscr{L} + \lambda \mathbf{w}^T\mathbf{w}\frac{\lambda}{2M}\sum_{i=1}^{N} w_i^2 \tag{6.3}$$

where vector of weights $\mathbf{w}$ does not include $w_0$ weight. Moreover, when normalization is used, no $w_0$ weight is needed.

### 6.2.1 Ridge Regression

**Ridge regression**: $L_2$-regularized linear regression.

$$\begin{aligned}\mathscr{L}(\mathbf{w}) &= \frac{1}{2M}\|\mathbf{y}-\mathbf{X}\mathbf{w}\|^2 + \frac{\lambda}{2M}\underbrace{\|\mathbf{w}\|^2}_{\sum_{i=1}^{N} w_i^2} \\ &= \frac{1}{2M}(\mathbf{y}-\mathbf{X}\mathbf{w})^T(\mathbf{y}-\mathbf{X}\mathbf{w}) + \frac{\lambda}{2M}\mathbf{w}^T\mathbf{w}\end{aligned} \tag{6.4}$$

Derivative

$$\nabla\mathscr{L}(\mathbf{w}) = \frac{1}{M}\left(-\mathbf{X}^T(\mathbf{y}-\mathbf{X}\mathbf{w}) + \lambda\mathbf{w}\right) = 0 \tag{6.5}$$

Solution

$$\begin{aligned}&-\mathbf{X}^T(\mathbf{y}-\mathbf{X}\mathbf{w}) + \lambda\mathbf{w} = -\mathbf{X}^T\mathbf{y} + \mathbf{X}^T\mathbf{X}\mathbf{w} + \lambda\mathbf{w} = 0 \\ &\mathbf{X}^T\mathbf{y} = \mathbf{X}^T\mathbf{X}\mathbf{w} + \lambda\mathbf{w}\end{aligned}$$

$$\mathbf{w} = \left(\lambda\mathbf{I} + \mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}, \tag{6.6}$$

Can be viewed as special case of linear regression, of the form

$$\begin{aligned}\tilde{y} &= \tilde{\mathbf{X}}\mathbf{w} \\ \begin{bmatrix}\mathbf{y}\\0\\\vdots\\0\end{bmatrix} &= \begin{bmatrix}- & \mathbf{X} & - \\ \sqrt{\lambda} & & 0 \\ & \ddots & \\ 0 & & \sqrt{\lambda}\end{bmatrix}\begin{bmatrix}w_1\\\vdots\\w_N\end{bmatrix} + w_0\end{aligned} \tag{6.7}$$

**GD Solution**

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \frac{\alpha}{M}\left[\mathbf{X}^T(\mathbf{X}\mathbf{w}_n - \mathbf{y}) + \lambda\mathbf{w}_n\right]$$

$$= \mathbf{w}_n \left(1 - \frac{\alpha}{M}\lambda\right) - \frac{\alpha}{M}\mathbf{X}^T\left(\mathbf{X}\mathbf{w}_n - \mathbf{y}\right) \quad (6.8)$$

The $\alpha$ value is chosen such that

$$0.99 \gtrsim \left(1 - \frac{\alpha}{M}\lambda\right) \gtrsim 0.95 \quad (6.9)$$

### 6.2.2 General aspects

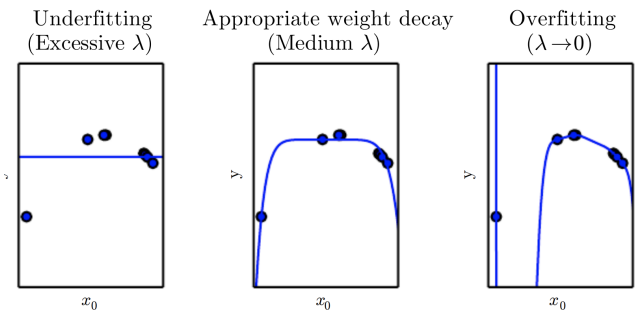



Figure 2: Illustration of $\lambda$ influence on model fitting.

**Slope interpretation** Higher value of $\lambda$ reduces (mostly) the highest slopes (weights $w_i$) $\Rightarrow$ the dependency on the parameters with these weights is reduced.

**Polynomial interpretation** One-dimensional dataset $(x_i, y_i)$ with nonlinear mapping $\mathbf{x}_i = [x_i, x_i^2, \ldots, x_i^N]$. The resulting coefficients $w_i$ will be significantly higher for higher $i$ ( Fig. 3). Reducing them results in smooth $\hat{y}$ prediction.

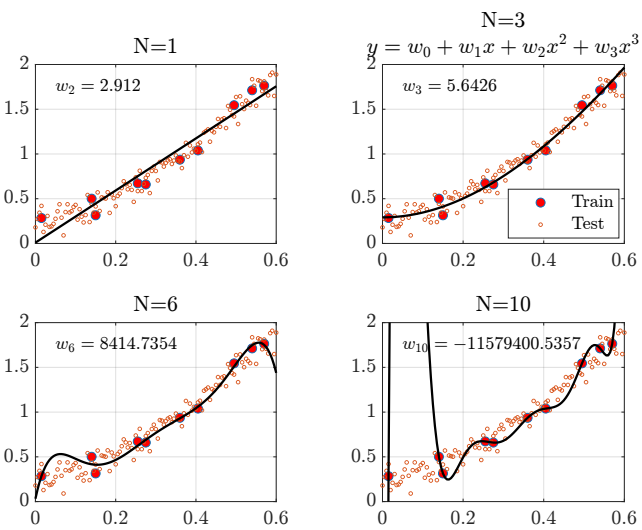



Figure 3: Illustration of weight grows with the polynomial model complexity.

**Eigenvalues interpretation** The this calculation limits the smallest eigenvalues to $> \frac{1}{\lambda}$ and thus improve the numerical stability.

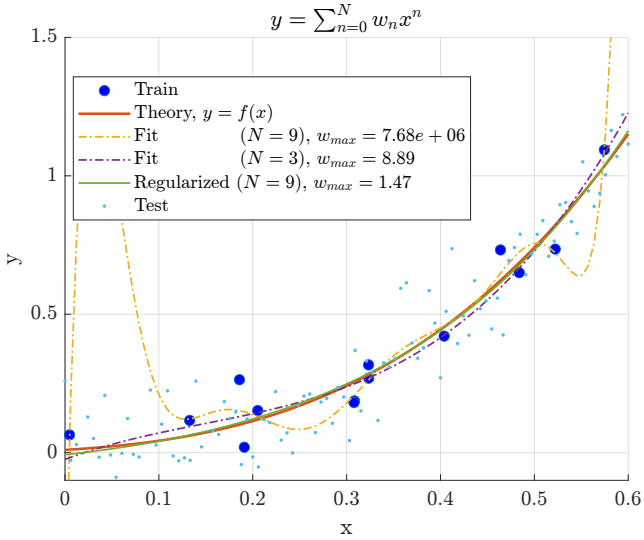



Figure 4: Illustration of regularization influence on the polynomial model.

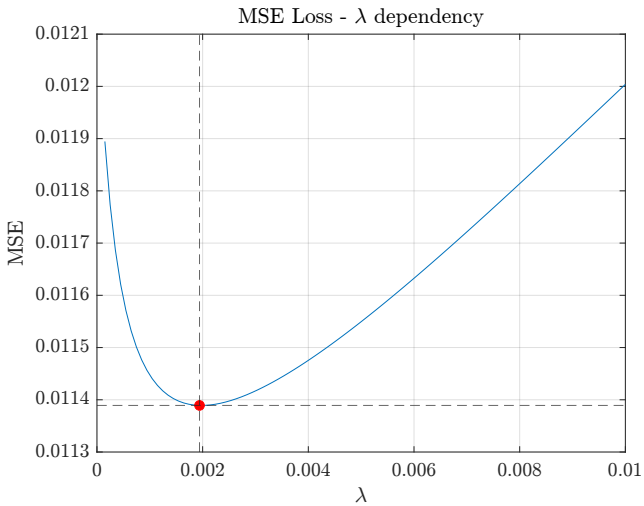



Figure 5: Dependence of MSE on $\lambda$ for regularization in Fig. 4. Overfitting on the left ($\lambda \to 0$) and undefitting on the right.

## 6.3 Normalization and Standardization

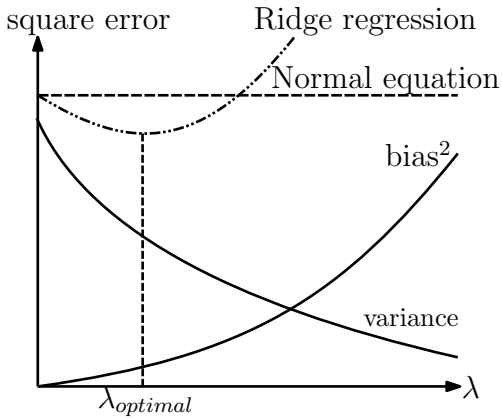
Goal: Data pre-processing level in Fig. 1.2. Used for multi-variate data.

Values in different columns in $\mathbf{X}$ (vectors $\mathbf{x}_j$) may be different by orders magnitudes, i.e. $\|\mathbf{x}_i\| \gg \|\mathbf{x}_j\|$. This results in:

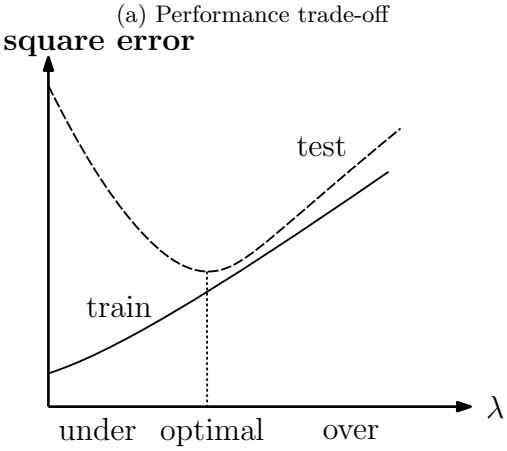
- Some columns have significantly higher influence on $\hat{\mathbf{y}}$.
- Numerical instabilities.

**Standardization**: Mapping all the input values such that they follow a distribution with zero mean and unit variance.

$$z_{std} = \frac{\mathbf{x} - \bar{\mathbf{x}}}{\sigma_{\mathbf{x}}}, \quad (6.10)$$




(a) Performance trade-off




(b) Train/test

Figure 6: Ridge regression: visualization of bias-variance trade-off.

where

$$\bar{\mathbf{x}} = \frac{1}{M}\sum_{j=1}^{M} x_j$$

$$\sigma_{\mathbf{x}}^2 = \frac{1}{M}\sum_{j=1}^{M} \left(x_j - \bar{\mathbf{x}}\right)^2$$

Implementation steps:

1. On **train** dataset, evaluate $\bar{\mathbf{x}}$ and $\sigma_{\mathbf{x}}$.
2. Apply normalization on **train** dataset, using $\bar{\mathbf{x}}$ and $\sigma_{\mathbf{x}}$.
3. Apply normalization on **test** dataset, using **same** $\bar{\mathbf{x}}$ and $\sigma_{\mathbf{x}}$ (no recalculation).

When normalization is applied to $\mathbf{y}$, the output of the model is transformed back, $\hat{\mathbf{y}} = \hat{\mathbf{y}}_{std}\sigma_{\mathbf{y}} + \bar{\mathbf{y}}$.
Example: `zscore` command in Matlab.
Note, standardization of both $\mathbf{X}$ and $\mathbf{y}$ results in $w_0 = 0$ and removes the requirement for $\mathbf{1}$ column in $\mathbf{X}$.
**Normalization**: Mapping all values of a feature to be in the range $[0, 1]$ by the transformation

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{6.11}$$

Implementation steps for normalization are similar to standardization.

Beware, normalization and standardization are used interchangeably.

# Chapter 7

# Logistic Regression

**Goal:** Binary classification with linear decision boundary.

## 7.1 Generalized Binary Linear Classification Models

**Generalized linear model**: Generalized linear model is the model that applies some (non-linear) function $g(\cdot) : \mathscr{R} \to \mathscr{R}$ on $\mathbf{w}^T\mathbf{x}_i$,

$$\hat{y}_i = g(\mathbf{w}^T\mathbf{x}_i) \tag{7.1}$$

For classification, $y \in \{0, 1\}$.

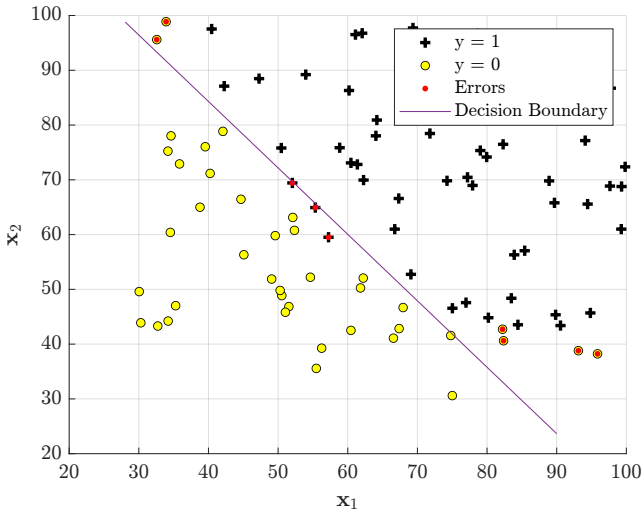



Figure 1: An example of linear classification boundary.

**Example 7.1**: Examples of a function $g(x)$ are,

$$g(x) = x \text{ basic linear model} \tag{7.2}$$

$$\sigma(x) = \frac{1}{1 - \exp(x)} \text{ sigmoid} \tag{7.3}$$

$$0 \leq \sigma(x) \leq 1$$

**Goal:** How to find weights $\mathbf{w}$?

Remainder for the line equation,

$$\mathbf{x}^T\mathbf{y} = \|\mathbf{x}\|\|\mathbf{y}\| \cos(\theta) \tag{7.4}$$

$\mathbf{x} \perp \mathbf{w} \Rightarrow \theta = 90^\circ \Rightarrow \mathbf{x}^T\mathbf{w} = 0$

## 7.2 Basic Linear Model

$$\tilde{\mathbf{y}} = \mathbf{X}\mathbf{w} \tag{7.5}$$

$$\hat{y}_j = \begin{cases} 1 & \tilde{y}_j > \frac{1}{2} \\ 0 & \tilde{y}_j < \frac{1}{2} \end{cases} \tag{7.6}$$

**Example**

$$\mathbf{X}^T = \left[\begin{array}{cccc|ccc} 1 & 1 & \cdots & 1 & 1 & \cdots & 1 \\ 10 & 11 & \cdots & 19 & 20 & \cdots & 29 \end{array}\right]$$

$$\mathbf{y}^T = \left[\begin{array}{cccc|ccc} 1 & 1 & \cdots & 1 & 0 & \cdots & 0 \end{array}\right]$$

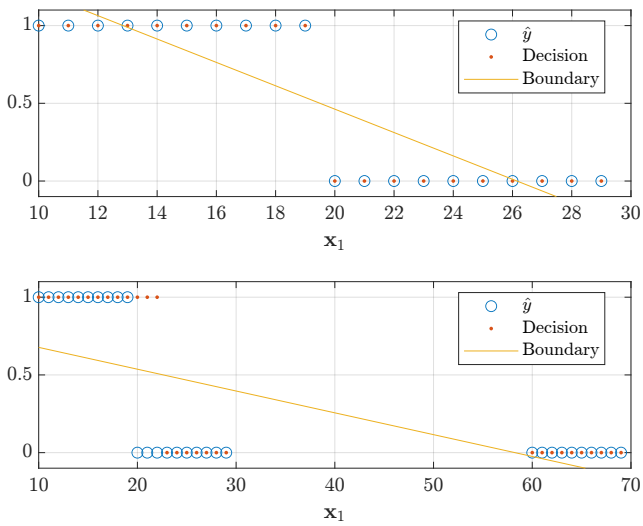



Figure 2: 1D synthetic example of classification by linear regression.

**Summary**

- $\tilde{y}$ may be higher than 1 and lower than 0.
- **Outlier** has a dramatic influence.

## 7.3 Logistic Model

**Goal:** Binary classification model with:

- Linear model
- Outliers handling
- Probabilistic interpretation

Logistic regression is one of the generalized linear classification models that is based on sigmoid function.
**Sigmoid function**:

$$\sigma(x) = \frac{\exp(x)}{1+\exp(x)} = \frac{1}{1+\exp(-x)} \tag{7.7}$$

The function is visualized in Fig. 3.

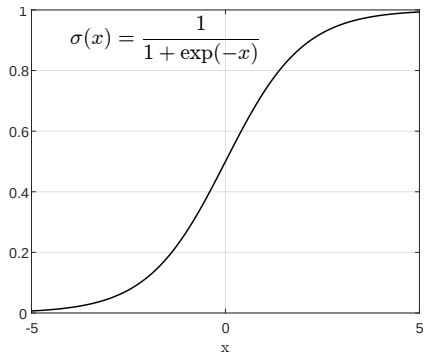



Figure 3: Plot of a sigmoid function.

**Logistic regression**:

$$\hat{y}_i = \sigma(\mathbf{w}^T\mathbf{x}_i) = \frac{1}{1+\exp\left(-\mathbf{w}^T\mathbf{x}_i\right)} \tag{7.8}$$

$$\hat{\mathbf{y}} = \sigma(\mathbf{X}\mathbf{w}) \tag{7.9}$$

**Loss function**: MSE loss has no closed-form solution and has local minimums⇒ not used.

$$\mathscr{L}(\cdot) = \frac{1}{2M}\|\hat{\mathbf{y}} - \mathbf{y}\|^2 \tag{7.10}$$

Loss function is not necessary metric.

## 7.4 Cross-entropy loss

**Goal:** Probabilistic loss.

The resulting value of $\hat{\mathbf{y}} = f_{\boldsymbol{\theta}}(\mathbf{X})$ has **probabilistic** interpretation and $L(\hat{\mathbf{y}}, \mathbf{y})$ quantifies a distance between target and output distributions. Typically, the distance metrics between probability density functions (PDFs) are used.

### 7.4.1 Entropy

**Entropy**: For the discrete distribution $P = \{p_i = \Pr[X = x_i]\}$, the entropy is given by

$$H(P) = -\sum_i p_i \log(p_i) \tag{7.11}$$

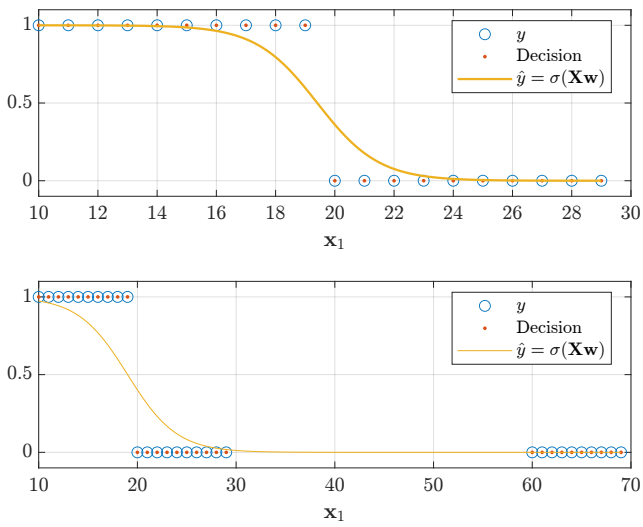



Figure 4: 1D synthetic example of classification by linear regression.

The sign depends on the context of the definition (sometimes + is used).
Entropy is a measure of the uncertainty associated with a given distribution, $P$. It has the maximum value for $p_i = p_j \; \forall i, j$ and drops down for any other combinations.
**Coding interpretation**: The entropy is an information theory measure. One of the interpretations of entropy (with base-2 logarithm and '-' sign) is the theoretical limit on the average number of bits needed to compress the outcomes of the distribution $p$. Numerical example:

$$p_1 = p_2 = \frac{1}{2} \Rightarrow H(p) = -2 \cdot \frac{1}{2}\log_2\left(\frac{1}{2}\right) = 1$$

$$p_1 = \frac{1}{10}, p_2 = \frac{9}{10} \Rightarrow H(p) = -\frac{1}{10}\log_2\left(\frac{1}{10}\right) - \frac{9}{10}\log_2\left(\frac{9}{10}\right) \approx 0.4690$$

Numerical example:

- Equal probabilities:
  - Consider the transmission of $\{A, B, C, D\}$ sequences over a binary channel. If all 4 letters are equally likely (25%) probable, $p_i = 0.25$.
  - The possible code is $\{00, 01, 11, 10\}$. One can not do better than using two bits to encode each letter.
  - $H(P) = -4 * \frac{1}{4}\log_2\left(\frac{1}{4}\right) = 2$
  - The lowest possible coding rate is achieved.
- Unequal probabilities example in Table 1:
  - Average coding rate is

$$\sum_i \text{length}_i \cdot p_i = 1 * 0.7 + 2 * 0.26 + 3 * 0.02 + 3 * 0.02 = 1.34$$

  - The theoretical lowest coding rate.

$$H(P) = -0.7\log_2(0.7) - 0.26\log_2(0.26) - 0.02\log_2(0.02) - 0.02\log_2(0.02) \approx 1.0912$$

Table 1: Unequal probabilities example.

| Word | Probability, $p_i$ | Codeword $C_i$ | Codeword length, $\text{length}_i$ |
|---|---|---|---|
| A | 0.7 | 0 | 1 |
| B | 0.26 | 10 | 2 |
| C | 0.02 | 110 | 3 |
| D | 0.02 | 111 | 3 |

$$\boxed{E[Y] = \sum_i y_i \Pr[Y = y_i]}$$

### 7.4.2 Cross-entropy

**Cross-entropy**: For two discrete distributions, $p$ and $q$, the cross-entropy is given by

$$H(p, q) = \pm \sum_i p_i \log(q_i) \tag{7.12}$$

The minimum value of $P(p, q)$ is when $p = q$, and as a consequence $H(p, q) = H(p)$.
**Coding interpretation**: One of the interpretations of entropy (with base-2 logarithm and $'-'$ sign) is the theoretical limit on the average number of bits that are required to encode distribution $p$ with the theoretically optimal code for $q$. Numerical example: Let's encode $\{A, B, C, D\}$ with probabilities $q_i = \left\{\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right\}$. In this case, one of the optimal coding options is to use two-bit binary code. Now, let's encode $p_i = \left\{\frac{1}{2}, \frac{1}{2}, 0, 0\right\}$ with this code. The resulting quantities are $H(p) = 1$ and $H(p, q) = 2$ which means that instead of optimal 1-bit code for $p$, the 2-bit code is required if the code optimal for $q$ is used to encode $p$.
Notes:

- $\lim_{x \to 0} x \log(x) \to 0$
- For a loss function, typically $e$-base logarithm is used.
- Maximum likelihood estimation (MLE) has the same minimum for $\boldsymbol{\theta}$.

### 7.4.3 Binary Cross-Entropy (BCE)

The visualization of BCE of the form

$$H(p, q) = -p_0 \log(q_0) - p_1 \log(q_1) \tag{7.13}$$

is presented in Fig. 5 For example, when $p_0 = 0$ and $p_1 = 1 - p_0 = 1$, the expression reduces to $H(p, q) = \log(q_1), q_1 \in [0, 1]$.

### 7.4.4 Binary Cross-Entropy (BCE) Loss

**Goal:** Minimum cross entropy.

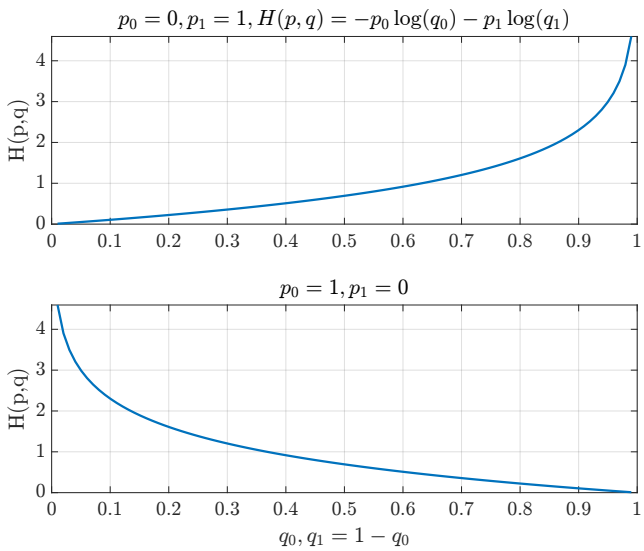



Figure 5: Illustration of BCE (Eq. (7.13).

For example, for a single decision for $y = 1$ in this example, we would like that $f_{\boldsymbol{\theta}}(\mathbf{x}) \to 1$,

$$\begin{aligned}
p_0 &= \Pr(y = 0) = 1 - y \\
p_1 &= \Pr(y = 1) = y \\
q_0 &= \Pr(\hat{y} = 0) = 1 - f_{\boldsymbol{\theta}}(\mathbf{x}) \\
q_1 &= \Pr(\hat{y} = 1) = f_{\boldsymbol{\theta}}(\mathbf{x}) \\
H(p, q) &= -p_0 \log(q_0) - p_1 \log(q_1) \\
&= -(1 - y) \log\big(1 - f_{\boldsymbol{\theta}}(\mathbf{x})\big) - y \log\big(f_{\boldsymbol{\theta}}(\mathbf{x})\big)
\end{aligned}$$

The discussion is symmetric for $y = 0, f_{\boldsymbol{\theta}}(\mathbf{x}) \to 0$.
**BCE loss**: Binary cross-entropy (BCE) loss function

$$\mathscr{L}(y, \hat{y}) = -(1 - y) \log(1 - \hat{y}) - y \log(\hat{y}) \tag{7.14}$$

For multi-valued vector $\mathbf{y}$ the loss is the average (or sum) over all $y_i$ elements,

$$\begin{aligned}
\mathscr{L} &= -\frac{1}{M} \sum_{j=1}^{M} (1 - y_i) \log(1 - \hat{y}_i) + y_i \log(\hat{y}_i) \\
&= -\frac{1}{M} \left[(1 - \mathbf{y}) \log(1 - \hat{\mathbf{y}}) + \mathbf{y} \log(\hat{\mathbf{y}})\right]
\end{aligned} \tag{7.15}$$

**Properties**: The BCE loss is continuous, differentiable and convex.

## 7.5 BCE Loss for Logistic Regression

**Probabilistic prediction**:

$$\begin{aligned}
p(y = 1 | \mathbf{x}, \mathbf{w}) &= \sigma(\tilde{\mathbf{x}}\mathbf{w}) \\
p(y = 0 | \mathbf{x}, \mathbf{w}) &= 1 - \sigma(\tilde{\mathbf{x}}\mathbf{w})
\end{aligned} \tag{7.16}$$

**Classification decision**: $\hat{y} \lessgtr \frac{1}{2}$

Another way:

$$\hat{\mathbf{y}} = \begin{cases} 1 & \mathbf{x}^T\mathbf{w} \geq 0 \\ 0 & \mathbf{x}^T\mathbf{w} < 0 \end{cases} \tag{7.17}$$

Vector notation

$$\mathscr{L} = \frac{1}{M}\left[-\mathbf{y}^T \log\big(\sigma(\mathbf{X}\mathbf{w})\big) - (1-\mathbf{y})^T \log\big(1-\sigma(\mathbf{X}\mathbf{w})\big)\right] \tag{7.18}$$

The first order gradient does not have a closed-form solution.

$$\nabla_{\mathbf{w}}\mathscr{L}(\mathbf{w}) = \frac{1}{M}\mathbf{X}^T\left(\sigma\left(\mathbf{X}\mathbf{w}\right) - \mathbf{y}\right) \tag{7.19}$$

However, it can easily found by GD minimization that involves only vector and matrix operations:

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \alpha\nabla_{\mathbf{w}}\mathscr{L}(\mathbf{w}) \tag{7.20}$$

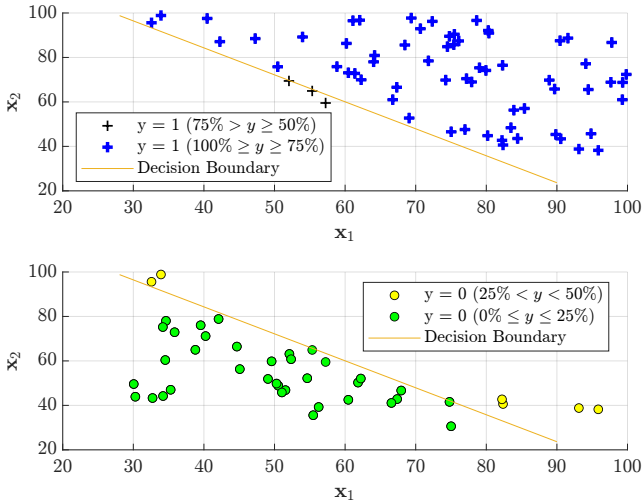



Figure 6: Example of $\sigma(\mathbf{X}\mathbf{w}) \geq 0.75$ and $\sigma(\mathbf{X}\mathbf{w}) \leq 0.25$ (see also Fig. 1).

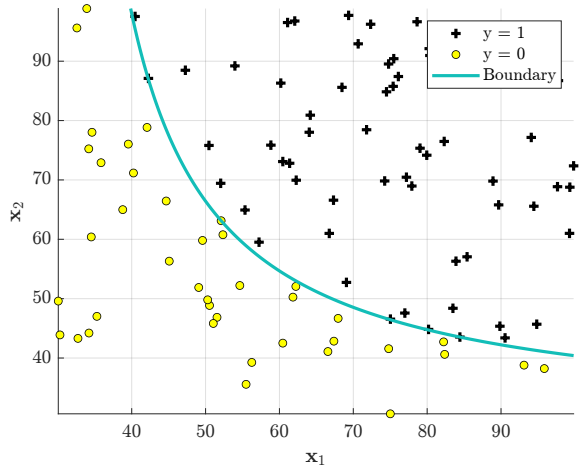
Important properties of the logistic regression with BCE loss function:

- Global minimum.
- Continuous, differentiable and convex.
- Regularization can be applied,

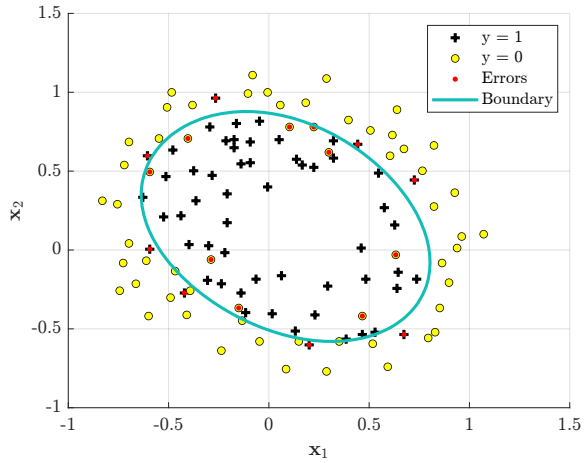
$$\mathscr{L}_{new} = \mathscr{L}(\mathbf{y}, \hat{\mathbf{y}}) + \frac{\lambda}{2M}\sum_{i=1}^{N} w_i^2 \tag{7.21}$$

- Mapping functions or kernels can be applied. For example, for polynomial mapping

$$\varphi(x_1, x_2) = \langle 1, x_1, x_1^2, x_2, x_2^2, x_1x_2, x_1^2x_2, x_1x_2^2, x_1^2x_2^2\rangle$$




(a) Example from Figs. 1 and 6.




(b) More challenging example.

Figure 7: Example polynomial $\varphi(x_1, x_2)$ mapping.

## 7.6 k-NN

Uses $k$ nearest neighbors for a decision.
Different distance metrics, some of them with additional hyper-parameter(s). For example:

- Euclidean distance metric,

$$\begin{aligned} d(\mathbf{a}, \mathbf{b}) &= \|\mathbf{a} - \mathbf{b}\| \\ &= \sqrt{(a_1-b_1)^2 + (a_2-b_2)^2 + \cdots + (a_M-b_M)^2} \end{aligned} \tag{7.22}$$

- City block (Manhattan) distance

$$\begin{aligned} d(\mathbf{a}, \mathbf{b}) &= \sum_{j=1}^{M} \left|a_j - b_j\right| \\ &= |a_1 - b_1| + \cdots + |a_M - b_M| \end{aligned} \tag{7.23}$$

- Minkowski distance with (hyper) parameter $p$,

$$d(\mathbf{a}, \mathbf{b}) = \sqrt[p]{\sum_{j=1}^{M} \left|a_j - b_j\right|^p} \tag{7.24}$$

For the special case of $p = 1$ the Minkowski distance gives the city block distance. For $p = 2$, the

Minkowski distance gives the Euclidean distance.
Tie-breaking (same number of neighbors) algorithm for $k > 1$ neighbors:

- Random selection.
- Use the class with the nearest neighbor.

**Summary**

- Fair baseline performance.
- High inference complexity. It requires $M$ distance calculations for each new point (e.g., logistic regression uses $\mathbf{w}$ vector.)
- Can not handle outliers.
- 100% training performance.
- Normalization is required!
- Two hyper-parameters: $k$ and distance metric.
- Can also be applied for regression.
- Decrease in performance with growth of $N$.

### 7.6.1 Curse of Dimensionality

Hypervolume of a Thin Shell between 2 Hyperspheres

# Chapter 8

# Classification Performance Metrics

**Goal:** Quantify the performance of a binary classifier on a test dataset.

Definitions:

- $\mathbf{y}$ - target values vector of the test database, $\mathbf{y} \in \mathbb{R}^M$
- $\hat{\mathbf{y}}$ - predicted value, $\hat{\mathbf{y}} \in \mathbb{R}^M$, output of some classifier $\hat{\mathbf{y}} = f_{\boldsymbol{\theta}}(\mathbf{X})$.

Typically, in binary classification, $y_i \in \{0, 1\}$.

## 8.1 Definitions

**Goal:** Classification between two groups (only).

Basic terminology:

- '1' – positive group or result
- '0' – negative group or result
- $Y$ – actual class
- $\hat{Y}$ – predicted class

Positive/negative terminology is rather arbitrary. Typically, the result of interest is termed positive.

## 8.2 Confusion matrix

**Goal:** Summarize classification results of a test set of a particular database.

The summarization is in the form of a 2D non-normalized histogram of $(Y, \hat{Y})$.
The (test) database has $M$ values, among them:

- TP + FN positive values
- FP + TN negative values

This is the most common way to summarize the performance of a particular classifier on a particular dataset. It can be easily extended for multi-class classifiers.

## 8.3 Performance Metrics

**Goal:** Characterization is useful to compare classifiers and/or performance on different datasets.

### 8.3.1 Accuracy

**Goal:** The most intuitive metric, fraction of $Y = \hat{Y}$, among all the classification results, $\Pr\left(Y = \hat{Y}\right)$.

| | | **Predicted** values | |
|---|---|---|---|
| | | **Positive**, $\hat{Y} = 1$ | **Negative**, $\hat{Y} = 0$ |
| **Actual** values | **Positive**, $Y = 1$ | **TP**<br>True Positive<br>$Y = 1, \hat{Y} = 1$ | **FN**<br>False Negative<br>$Y = 1, \hat{Y} = 0$ |
| | **Negative**, $Y = 0$ | **FP**<br>False Positive<br>$Y = 0, \hat{Y} = 1$ | **TN**<br>True Negative<br>$Y = 0, \hat{Y} = 0$ |

Figure 1: Confusion matrix. Note, sometimes, transposed representation is used.

$$\begin{aligned} \text{Accuracy} &= \frac{\text{correct predictions}}{\text{total predictions}} \\ &= \frac{TP + TN}{TP + NT + FP + FN} \end{aligned} \tag{8.1}$$

**Example 8.1**: Covid antibody (fast non-PCR) test performance. The example includes test statistics of 239 participants [**?**], as presented below.

| | | Predicted | |
|---|---|---|---|
| | | Yes | No |
| Actual | Yes | 141 | 67 |
| | No | 0 | 31 |

The resulting accuracy is

$$\text{Accuracy} = \frac{141 + 31}{239} = 0.7196652 \approx 72.0\% \tag{8.2}$$

| Term | Radar Interpretation |
|---|---|
| Accuracy | Percentage of all correctly identified as planes or not planes |
| Precision | Among all classified as planes, the portion that is correctly classified as planes |
| Recall sensitivity | Among all existing planes, portion of correctly classified as planes |
| Specificity | Among all classified as non-planes, portion of correctly classified as non-planes |

Table 1: Radar interpretation of the classification metrics.

In the example, FN=67 is a bad performance, and FP=0 is probably something good. However, accuracy does not reflect the discrepancy between these two. Additional metrics are used to quantify these aspects.

### 8.3.2 Precision

**Goal:** Proportion of positive classification that is actually correct, $\Pr\left(Y=1|\hat{Y}=1\right)$.

From the probability theory,

$$\Pr\left(Y=1|\hat{Y}=1\right)=\frac{\Pr\left(Y=1,\hat{Y}=1\right)}{\Pr(Y=1)} \tag{8.3}$$

$$\Pr(Y=1)=\Pr\left(Y=1,\hat{Y}=0\right)+\Pr\left(Y=1,\hat{Y}=1\right) \tag{8.4}$$

$$\text{Precision}=\frac{TP}{FP+TP}=\frac{\text{Correctly predicted 1's}}{\text{All predicted 1's}} \tag{8.5}$$

TP = Correctly predicted 1's
TP + FP = All predicted 1's

**Example 8.1**: Back to the previous example,

$$\text{Precision}=\frac{141}{0+141}=1=100\% \tag{8.6}$$

The high value of the precision is due to the low FP. From the medical point of view, all positive results are actually positive. Whoever was identified by this test as Covid-positive is really positive.

### 8.3.3 Recall (sensitivity)

**Goal:** Proportion of positives identified correctly, $\Pr\left(\hat{Y}=1|Y=1\right)$.

From the probability theory,

$$\Pr\left(\hat{Y}=1|Y=1\right)=\frac{\Pr\left(Y=1,\hat{Y}=1\right)}{\Pr\left(\hat{Y}=1\right)} \tag{8.7}$$

$$\Pr\left(\hat{Y}=1\right)=\Pr\left(\hat{Y}=1,Y=0\right)+\Pr\left(\hat{Y}=1,Y=1\right) \tag{8.8}$$

$$\text{Recall}=\frac{TP}{TP+FN}=\frac{\text{Correctly predicted 1's}}{\text{Actual 1's}} \tag{8.9}$$

Medical meaning: portion of correctly classified ill among all the ill.

**Example 8.1**: Back to the previous example,

$$\text{Recall}=\frac{141}{141+67}=0.678=67.8\% \tag{8.10}$$

The low value of the recall is due to the high FN. From the medical point of view, among all the positive results, only 67.8% are actually positive.

### 8.3.4 Specificity

**Goal:** Proportion of negatives identified correctly, $\Pr\left(\hat{Y}=0|Y=0\right)$.

$$\text{Specificity}=\frac{TN}{FP+TN}=\frac{\text{Correctly predicted 0's}}{\text{Actual 0's}} \tag{8.11}$$

Medical meaning: portion of classified healthy among all the healthy.

**Example 8.1**: Back to the previous example,

$$\text{Specificity}=\frac{31}{0+31}=1=100\% \tag{8.12}$$

From the medical point of view, all negative results are really negative.

### 8.3.5 $F_1$-score

**Goal:** Combination of precision and recall.

The harmonic mean between precision and recall,

$$F_1=\frac{2}{\frac{1}{recall}+\frac{1}{precision}}=\frac{TP}{TP+\frac{1}{2}\left(FP+FN\right)} \tag{8.13}$$

**Example 8.1**: Back to the previous example,

$$F_1=\frac{141}{141+\frac{1}{2}\left(0+67\right)}=0.808=80.8\% \tag{8.14}$$

## 8.4 Imbalanced Dataset

**Imbalanced dataset**: Dataset with significant differences between the numbers of labels of each class. The following examples present a few problems related to imbalanced datasets.

**Example 8.2**: Let's take a dataset with 1000 samples:

- 990 samples labeled '0'

- 10 samples labeled '1'

What are the performance metrics of the classifier that always predicts $\hat{Y} = 0$?

*Solution*: The resulting confusion matrix is

| | | Predicted | |
|---|---|---|---|
| | | Yes | No |
| Actual | Yes | 0 | 10 |
| | No | 0 | 990 |

and the resulting quantities are

$$\begin{aligned}
\text{Accuracy} &= \frac{990}{1000} = 0.99 = 99\% \\
\text{Precision} &= \frac{TP}{FP+TP} = \frac{0}{0+0} = \text{Undefined} \\
\text{Recall} &= \frac{TP}{TP+FN} = \frac{0}{0+10} = 0 \\
\text{Specificity} &= \frac{TN}{FN+TN} = \frac{990}{1000} = 0.99 = 99\% \\
F_1 &= \frac{TP}{TP+\frac{1}{2}(FP+FN)} = \frac{0}{\cdots} = 0
\end{aligned} \tag{8.15}$$

- Note, accuracy is insufficient metrics!
- Note, while the convention is to label '1' for the most important class outcome, sometimes it is interchangeable.

### Majority classifier

Majority class classifier is where the most frequent class in the data is predicted all the time, e.g. as in example above. This theoretical "classifier" is often used as a baseline metric for improvement by other machine learning techniques.

### Small dataset problem

**Example 8.3**: We have the dataset from the previous example (Example 9.2). This time, let's assume that the theoretical performance of the classifier on class '1' is $p = 0.8$. What is the probability that the classifier will classify only 6 samples or less correctly, from 10 measurements?

*Solution*: The probabilities follow the distribution. $X \sim Bin(n = 10, p = 0.8)$ with a question $\Pr(X \leq 6) =?$. The numerical solution is

$$\Pr(X \leq 6) = \Pr(X = 0) + \cdots + \Pr(X = 6) \approx 12.09\%$$

Moreover, $\Pr(X = 10) = 10.74\%$.

The analysis of the issue illustrated in the example is called *confidence* analysis. While the discussion of confidence intervals is out of the scope of this document, this example emphasizes the problem of a small dataset, particularly in imbalanced data.

**Anomaly detection** Sub-field of imbalanced problem: anomaly detection.

## 8.5 Decision threshold

### 8.5.1 Receiver Operating Characteristics (RoC)

With the probabilistic loss function, the classifier output is the probability of

$$\Pr(\hat{y} = 1) = f_{\boldsymbol{\theta}}(x). \tag{8.16}$$

The binary decision for $\hat{\mathbf{y}}$ is to compare $f_{\boldsymbol{\theta}}(x)$ with some predefined threshold,

$$\hat{\mathbf{y}} = \begin{cases} 1 & f_{\boldsymbol{\theta}}(x) \geq \mathsf{thr} \\ 0 & f_{\boldsymbol{\theta}}(x) < \mathsf{thr} \end{cases} \tag{8.17}$$

with the default value of $\mathsf{thr} = 0.5$. The change of $\mathsf{thr}$ may significantly influence the resulting confusion matrix.

Goal: To quantify the trade-off between confusion matrix elements as a function of $\mathsf{thr}$. The used quantities are:

- True Positive Rate (**TPR**) is a synonym for recall.
- False Positive Rate (**FPR**) is defined by

$$FPR = \frac{FP}{FP+TN} = 1 - \text{specificity} \tag{8.18}$$

RoC is a legacy term from the field of detector theory and communication system theory.

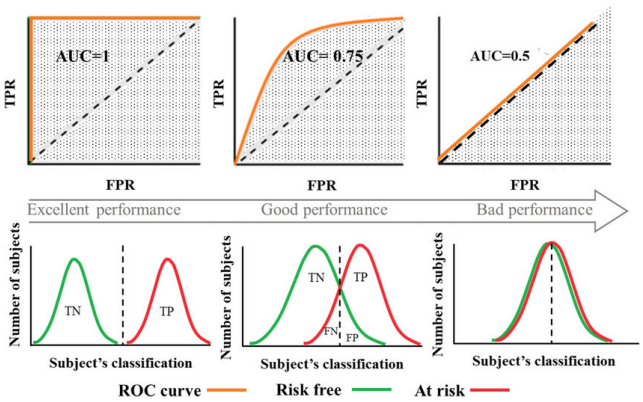



Figure 2: Influence of threshold on RoC.

### 8.5.2 Area under curve (AUC)

Goal: Quantify threshold-independent performance.

**AUC**: AUC is the area under the RoC curve.
**Range**: Borderline cases are a coin toss with $\mathsf{AUC} = 0.5$ and an ideal classifier with $\mathsf{AUC} = 1$. All other classifiers fall in the range $0.5 \leq \mathsf{AUC} \leq 1$.
Properties:

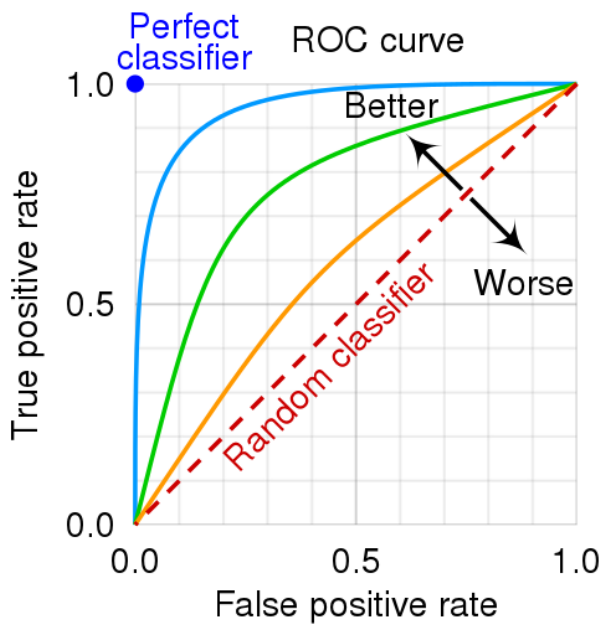



Figure 3: RoC comparison.

- AUC is scale-invariant. It measures how well predictions are ranked, rather than their absolute values.
- AUC is classification-threshold-invariant. It measures the quality of the model's predictions irrespective of what classification threshold is chosen.
- Scale invariance is not always desirable for a performance assessment.
- Classification-threshold invariance is not always desirable. Sometimes some trade-off between false negatives vs. false positives is required. For example, when doing email spam detection, you likely want to prioritize minimizing false positives (even if that results in a significant increase of false negatives). AUC isn't a useful metric for this type of optimization.

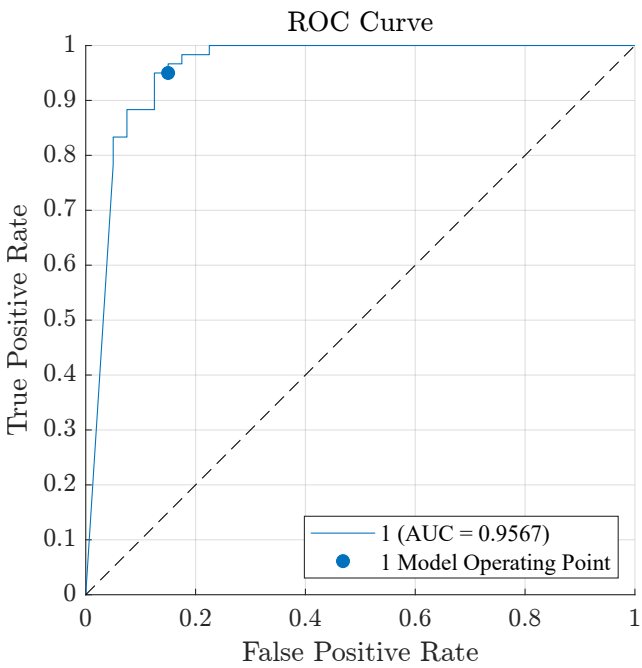



Figure 4: RoC of logistic regression example in Fig. 1. The model operation point is $\mathsf{thr} = 0.5$.

# Chapter 9

# Notation

### Numbers and indexing

| | |
|---|---|
| $a$ | Scalar |
| $\mathbf{a}$ | Vector |
| $a_i$ | Element $i$ of a vector $a$, indexing starting at 1 |
| $\mathbf{A}$ | Matrix |
| $a_{ij}$ | Element $i, j$ of a matrix $\mathbf{A}$, indexing starting at 1 |
| $\mathscr{R}$ | Real numbers domain |
| $\mathscr{R}^D$ | $D$-dimensional vector |
| $\mathscr{R}^{D_1 \times D_2}$ | matrix of a dimension $D_1 \times D_2$ |

### Datasets

| | |
|---|---|
| $N$ | Number of features |
| $M$ | Number of entries in the dataset |
| $K$ | Number of classes |
| $\mathbf{w}$ | Model parameters |
| $f(\cdot; \mathbf{w})$ | Model |
| $x_{ij}$ | Singe data value |
| $\mathbf{x}_i$ | Singe data vector, $i$ column number in $\mathbf{X}$ |
| $\mathbf{X}$ | Data matrix |
| $\mathbf{y}$ | Target vector for the data in $\mathbf{X}$ |
| $\hat{\mathbf{y}}$ | Prediction vector of $\mathbf{y}$ |
| $y_i$ | Target value |
| $\hat{y}_i$ | Predicted target value |
| $\mathscr{L}(\mathbf{y}, \hat{\mathbf{y}})$ | Loss function (vector domain) |
| $\mathscr{L}(y_i, \hat{y}_i)$ | Loss function (scalar domain) |
| $\mathbf{a}^{[k]}$ | Activation of layer $k$ |
| $\mathbf{z}^{[k]}$ | Output of layer $k$ |
| $g_k(\cdot)$ | Activation function of layer $k$ |

# Bibliography


[1] Tomas Andersson. *Selected topics in frequency estimation.* PhD thesis, KTH Royal Institute of Technology, 2003.

[2] Dima Bykhovsky. Experimental lognormal modeling of harmonics power of switched-mode power supplies. *Energies*, 15(2), 2022.

[3] Dima Bykhovsky and Asaf Cohen. Electrical network frequency (ENF) maximum-likelihood estimation via a multitone harmonic model. *IEEE Transactions on Information Forensics and Security*, 8(5):744–753, 2013.

[4] Sharon Gannot, Zheng-Hua Tan, Martin Haardt, Nancy F Chen, Hoi-To Wai, Ivan Tashev, Walter Kellermann, and Justin Dauwels. Data science education: The signal processing perspective [sp education]. *IEEE Signal Processing Magazine*, 40(7):89–93, 2023.

[5] Monson H Hayes. *Statistical Digital Signal Processing and Modeling.* John Wiley & Sons, 1996.

[6] Steven M. Kay. *Fundamentals of Statistical Signal Processing, Volume I: Estimation Theory.* Prentice Hall, 1993.

[7] John H Mathews and Kurtis D Fink. *Numerical methods using MATLAB.* Pearson, $4^{\text{th}}$ edition, 2004.