Uncertainty, Information and Learning Mechanisms
(Part 2)

Intelligence for Embedded Systems
Ph. D. and Master Course
Manuel Roveri
Politecnico di Milano, DEIB, Italy
Uncertainty and Learning

- The real world is prone to **uncertainty**
- At different levels of the data analysis process
  - Acquiring data
  - Representing information
  - Processing the information
  - Learning mechanisms

- **Two questions:**
  - Why do we need learning mechanisms?
  - Which are basics of supervised statistical learning?
A “toy” example

- Physical model? I did not complete my PhD in Physics yet
- Data-driven might be a good solution (brute-force as well)

What is the learning goal here?
Data-processing and applications

P
Data generating process

Model of the system

Application
Not rarely we wish to generate a functional dependency among sensor datastreams (model) to solve a problem. Some examples

- **Design a Classifier**
  - e.g., optical character recognition, face recognition, explosive detection, quality analysis in the manufacturing industry…

- **Construct a functional dependency** (function regression)
  - e.g., function reconstruction, data regularization, predictive modeling…
In some cases the physical equations describing the process are available but some parameters need to be determined.

In others, we do not know the equations ruling the system.

If the considered model is linear we speak about system identification, when not linear about learning.
Theoretical modelling
- Physical laws known; parameters known
  - White-box models
    - Linear/non-linear differential equations

Experimental modelling
- Signals measurable
  - Light-grey-box models
    - Differential equations w. parameter estimation
- Signals measurable
  - Dark-grey-box models
    - Fuzzy-models with parameter estimation
- Input/output signals measurable; assumption of a model structure
  - Black-box models
    - Impulse response; neural networks
Learning the system model

\[ p(x|t) = \sum_{y \in \Lambda} p(y|t)p(x|y, t), \]
\[ y(k) = h(u(k)) + d(k) \]
\[ A(z)y(k) = \sum_{i=1}^{m} \frac{B_i(z)}{F_i(z)} u_i(k) + \frac{C(z)}{D(z)} d(k) \]

We will come back to the learning mechanism later

---

**P**

Data generating process

\((x, y)\)

Estimate a model

Application

---

\[ p(x|t) = \sum_{y \in \Lambda} p(y|t)p(x|y, t), \]
\[ y(k) = h(u(k)) + d(k) \]
\[ A(z)y(k) = \sum_{i=1}^{m} \frac{B_i(z)}{F_i(z)} u_i(k) + \frac{C(z)}{D(z)} d(k) \]
• A real example: classifying microacoustic bursts
In addition:
Many temperature sensors

Strain gauges
High precision inclinometers

Pluviometers
Mid precision inclinometers

Flow meters

MEMS accelerometer

The Torrioni di Rialba Monitoring system
- We wish to automatically identify microacoustic bursts to reduce the false alarms (data trigger).
- The application requires to design a classifier assigning a relevant/not relevant label to bursts.
Designing a Classifier

1. Pre-Processing

- Burst alignment: bursts need to be aligned to provide a common comparison

2. Feature Extraction

- Bursts are then processed to extract scalar features

3. Design the Classifier

- Design the classifier from available bursts

4. Use the classifier

- Bursts are automatically classified
Microacoustic Bursts

- Each burst is projected onto an adaptive orthonormal three-axis system estimated with a Principal Component Analysis (PCA)
Pre-processing

- Only the principal and the least significant components are further processed.
Feature Extraction

Time Domain Features

- Mean (before PCA)
- PCA eigenvalue
- Max amplitude

- SNR
- PSNR
- Peak decay
Feature Extraction

Fourier Domain Features (Power Spectrum)

- Max amplitude
- Main frequency
- Variance of peaks
Designing a classifier requires identification of the function separating the labeled points.
Designing a classifier: Burst separation

Determination of the linear separating border with Linear Discriminant Analysis (LDA). We assume that the two classes have a gaussian distribution with different means and identical covariance matrix:

- Squares correspond to LDA values of real fractures
- Circles correspond to LDA values of false alarms
Some issues we need to focus on…

- Linear vs. non linear
- Many points versus the available points
- Several techniques are available to design the classifier (KNN, feedforward, SVM…)

- Some diagrams illustrating the concepts of linear and non-linear classification, with points and lines representing different classes.
Non linear regression

Given a set of $n$ noise affected couples $(x_i, y_i)$ we wish to reconstruct the unknown function
• Supervised Learning: Statistical framework
The time invariant process generating the data

\[ y = g(x) + \eta, \]

provides, given input \( x_i \) output instance

\[ y_i = g(x_i) + \eta_i \]

We collect a set of couples \((\text{training set})\)

\[ Z_N = \{(x_1, y_1), \ldots, (x_N, y_N)\} \]

And wish to model unknown \( g(x) \) with parameterized family of models \( f(\theta, x) \)

The goal of learning is to build the simplest approximating model able to explain past data \( Z_N \) and future instances provided by the data generating process.
The parameterized family of models $f(\theta, x)$ takes advantage of a priori information about $g(x)$.

- belongs to a nested hierarchy of model families
- is mostly continuous and differentiable
- is chosen according to the a priori information we have about $g(x)$ (e.g., gray box modeling), model complexity (linear families) or universal function approximation ability.
- Not rarely it is linear
- It can also have dynamics (e.g. AR, ARX, recurrent networks)
The statistical learning framework

Structural risk

$$\bar{V}(\theta) = \int L(y, f(\theta, x)) p_{x,y} dxy$$

$$\theta^o = \arg\min_{\theta \in \Theta} \bar{V}(\theta)$$

Empirical risk

$$V_N(\theta) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(\theta, x_i))$$

$$\hat{\theta} = \arg\min_{\theta \in \Theta} V_N(\theta)$$

Learning procedure

$$\theta_{i+1} = \theta_i - \eta \frac{\partial V_N(\theta, Z_N)}{\partial \theta} |_{\theta_i}$$
Inherent, approximation and estimation risks

\[ \bar{V}(\hat{\theta}) = (\bar{V}(\hat{\theta}) - \bar{V}(\theta^0)) + (\bar{V}(\theta^0) - V_I) + V_I. \]

- The **inherent risk** depends only on the structure of the learning problem and, for this reason, can be reduced only by improving the problem itself.
- The **approximation risk** depends on how close the model family (also named hypothesis space) is to the process generating the data.
- The **estimation risk** depends on the ability of the learning algorithm to select a parameter vector \( \hat{\theta} \) close to \( \theta^0 \).
Approximation and estimation risks

Target Space

Model Space

Selected Model

Best Reachable Model

Optimal Model
Approximation and estimation risks

- Approximation Error
- Estimation Error

Target Space: Model Space

- Selected Model
- Best Reachable Model
- Optimal Model
Choosing a universal function approximator

Not rarely \( f(\theta, x) \) is chosen as a feedforward neural network.
And the function implemented by a neuron is

The activation function can be

- A hard limiter function
- A sigmoidal-like function
- A linear one (output layer)
Quality assessment of the solution

«How good is your ‘good’?»
Assessing the performance (1)

- **Apparent Error Rate (AER), or resubstitution:** The whole set $Z_N$ is used both to infer the model and to estimate its error.

- **Sample Partitioning (SP):** $S_D$ and $S_E$ are obtained by randomly splitting $Z_N$ in two disjoint subsets. $S_D$ is used to estimate the model and $S_E$ to estimate its accuracy.

- **Leaving-One-Out (LOO):** $S_E$ contains one pattern in $Z_N$, and $S_D$ contains the remaining $n - 1$ patterns. The procedure is iterated $n$ times by holding out each pattern in $Z_N$, and the resulting $n$ estimates are averaged.
Assessing the performance (2)

- **w-fold Crossvalidation (wCV):** $Z_N$ is randomly split into $w$ disjoint subsets of equal size. For each subset the remaining $w - 1$ subsets are merged to form $S_D$ and the reserved subset is used as $S_E$. The resulting $w$ estimates are averaged. This procedure can be iterated and the results averaged when $w \ll n$ in order to reduce the random resampling variance. This estimate is a generalization of LOO.
An example: Binomial law confidence interval for the classifier accuracy
… what about the confidence of an estimator?

• Bootstrap
• Bag of little bootstrap
Consider a data set $Z_n$ obtained by extracting $n$ i.i.d. samples $x_1, \ldots, x_n$ from random variable $x$ defined over $X$, i.e., $Z_n = \{x_1, \ldots, x_n\}$ and construct the estimator $\Phi_n = \Phi(Z_n)$. We are interested in providing an indication of the quality $\zeta$ of $\Phi_n$, e.g., we wish to provide a confidence interval for $\Phi_n$.

Clearly, the ideal framework would recommend to carry out the following procedure

1. Extract $m$ independent data sets of cardinality $n$ from $X$ so as to generate datasets $Z_n^1, \ldots, Z_n^m$;
2. Evaluate, in correspondence of the generic $i$-th data set $Z_n^i$ the estimator $\Phi_n^i = \Phi(Z_n^i)$. Repeat this procedure for all $i = 1, \ldots, m$;
3. Estimate the quality $\zeta(\Phi_n^1, \ldots, \Phi_n^m)$ of the estimator $\Phi_n$ based on the $m$ realizations $\Phi_n^i = \Phi(Z_n^i), i = 1, \ldots, m$. 
The bootstrap

Unfortunately, the above framework is mostly theoretical: if we have \( m \) independent datasets \( Z_n \), we should use all \( nm \) data to provide a better estimate. This means that in practical applications we have only a dataset but, at the same time, we are interested in evaluating the quality \( \zeta \) of the estimator \( \Phi_n \).

- In the bootstrap method data sets \( Z_n^i, i = 1, \ldots, m \) are extracted with replacement from \( Z_n \)

---

Algorithm 15: The bootstrap algorithm

\[
i = 0;\\
\text{while } i < m \text{ do}\\
\quad \text{Extract } n \text{ samples with replacement from } Z_n \text{ and insert them in } Z_n^i;\\
\quad \text{Compute } \Phi_n^i = \Phi(Z_n^i);\\
\quad i = i + 1;\\
\text{end}\\
\]

Evaluate the assessment \( \zeta(\Phi_n^1, \ldots, \Phi_n^m) \) of the quality of the estimator \( \Phi_n \).
The bag of little bootstraps method

- BLB shows to be accurate and appears to over-perform all other bootstrap methods in terms of computational complexity, hence becoming a very appealing method for Big Data.

**Algorithm 16:** The Bag of Little Bootstraps algorithm

\[
\begin{align*}
n' &= n'^r; \\
i &= 0; \\
\text{while } i < m \text{ do} & \\
& \quad \text{Extract } n' \text{ samples without replacement from } Z_n \text{ and insert them in } Z_{n'}^i; \\
& \quad j = 1; \\
& \quad \text{while } j < r \text{ do} & \\
& \quad \quad \text{Extract } n \text{ samples with replacement from } Z_{n'}^i \text{ and insert them in } Z_{n,r}^i; \\
& \quad \quad \text{Compute } \Phi_{n,r}^i = \Phi(Z_{n,r}^i); \\
& \quad \quad j = j + 1; \\
& \quad \text{end} \\
& \quad \text{Evaluate } \zeta_i = \zeta \left( \Phi_{n,1}^i, \ldots, \Phi_{n,r}^i \right); \\
& \quad i = i + 1; \\
& \text{end} \\
& \text{Evaluate the assessment } \zeta \text{ for the quality of the estimator } \Phi_n \text{ as } \zeta = \frac{\sum_{i=1}^{m} \zeta_i}{m}. 
\end{align*}
\]  

\( \gamma = 0.6 \)
Let’s play with MATLAB

- Download the examples related to Lecture 3 – Part 2

- In the ZIP file:
  - Example 3_2_A.m
    - Linear dataset, linear model
  - Example 3_2_B.m
    - Linear dataset, nonlinear model
  - Example 3_2_C.m
    - Nonlinear dataset, linear model
  - Example 3_2_D.m
    - Nonlinear dataset, nonlinear model
  - Example 3_2_E.m
    - Different number of hidden units
  - Example 3_2_F.m
    - Classification and Apparent Error Rate
  - Example 3_2_G.m
    - Classification and Sample Partitioning