MSc on Intelligent Critical Infrastructure Systems

Machine Learning

Lecture 1

Marios Polycarpou
Director, KIOS Research and Innovation Center of Excellence
Professor, Electrical and Computer Engineering
University of Cyprus
Fourth Industrial Revolution

- What is a technological revolution?
- 1st Industrial Revolution (1760)
- 2nd Industrial Revolution (1900)
- 3rd Industrial Revolution (1960)

→ Combines hardware, software, and biology (cyber-physical systems), and emphasizes advances in communication and connectivity. It is expected to be marked by breakthroughs in fields such as machine learning, robotics, nanotechnology, biotechnology, the internet of things (IoT), wireless technologies (5G), 3D printing and fully autonomous vehicles.
Motivation for Machine Learning

- Sensor technology
  - wealth of sensors
  - new generation sensors

- Information & Communication Technology (ICT)
  - store, process and transmit data collected by sensors
**Motivation for Machine Learning**

- **Internet of Things (IoT)**
  - *sensor enabled devices* connected to the internet and able to communicate with each other

- **Big Data**
  - *sensor technology* and ICT enabled the collection of extremely large data sets
  - contribute to the better perception of complex systems
Motivation for Machine Learning

- Image processing
- Speech recognition
- Prediction in financial services
- Finding patterns in marketing and sales
- Improving healthcare
- Automation – no human intervention needed
- Continuous improvement
- Wide applications
How is Machine Learning related to Intelligent Critical Infrastructure Systems?

**Critical Infrastructure Systems (CIS)**
- Heterogeneous - Interdependent - Interconnected
- Risks, Faults, Attacks

**Information Communication Technologies (ICT)**
- Sensors, Actuators
- Big Data, Internet of Things

**Intelligent Systems and Networks**
- Monitoring, Control
- Management, Security
- Big Data → Smart Decisions
How is Machine Learning related to Intelligent Critical Infrastructure Systems?

Smart Grids

Smart Buildings
How is Machine Learning related to Intelligent Critical Infrastructure Systems?

Water Systems

Revenue & Water Losses  Water Quality  Energy Consumption  Safety & Security

Water loss: seven things you need to know about an invisible global problem

China water contamination affects 2.4m after oil leak

Cyber-attack claims at US water facility
How is Machine Learning related to Intelligent Critical Infrastructure Systems?

Intelligent Transportation

Multi-Agent Formation
How is Machine Learning related to Intelligent Critical Infrastructure Systems?

Smart Cities

- Physical Interconnections
- Cyber Interconnections
- Interdependencies
Defining Machine Learning

A machine learning algorithm is an algorithm that is able to learn from data. Other related terms: data-driven methods, computational intelligence, artificial intelligence (AI).

In general, Artificial Intelligence and Computational Intelligence are bigger concepts aimed at creating intelligent machines that can simulate human thinking capability and behaviour. Machine Learning is a subset of AI that allows machines to learn from data without being programmed explicitly.
Defining Machine Learning

A computer program is said to learn from experience $E$ with respect to some class of tasks $T$ and performance measure $P$, if its performance at tasks in $T$, as measured by $P$, improves with experience $E$ (Mitchell, 1997).

The Task $T$

- Classification
- Regression
- Monitoring and anomaly detection
- Transcription
- Speech recognition
- Machine translation
- Navigation and Control
The Task $T$ -- Classification

$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$

Features

$y = \{1, 2, \ldots K\}$

Classes

$x \rightarrow \text{Classifier} \rightarrow y = \{1, 2, \ldots K\}$

$f : \mathbb{R}^n \mapsto \{1, 2, \ldots K\}$
The Task $T$ -- Regression

\[ f : \mathbb{R}^n \mapsto \mathbb{R} \]
The Performance Measure \( P \)

- Error rate; error properties
- **Training** set; **Test** set
- Different performance measures may be used
- What are norms?

### Examples of norms \((\varepsilon \in \mathbb{R}^n)\)

<table>
<thead>
<tr>
<th>Norm</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-norm</td>
<td>( |\varepsilon|<em>1 := \sum</em>{i=1}^{n}</td>
</tr>
<tr>
<td>2-norm</td>
<td>( |\varepsilon|<em>2 := \left( \sum</em>{i=1}^{n}</td>
</tr>
<tr>
<td>(\infty)-norm</td>
<td>( |\varepsilon|<em>{\infty} := \max</em>{1 \leq i \leq n}</td>
</tr>
</tbody>
</table>

(1-norm) Euclidean norm (2-norm) \((\infty\)-norm)
The Experience $E$

What type of dataset is available?

**Types of Learning**
- Supervised Learning (there exist labelled examples)
- Unsupervised Learning (clustering, dimensionality reduction)
- Semi-Supervised Learning
- Reinforcement Learning
Machine Learning - Simple Example

Dataset:

\[ f(x) = \sin(2\pi x) \]

\[ y = f(x) + \varepsilon \]

\[ x = (x_1, x_2, \ldots x_N) \quad N = 10 \]
Machine Learning - Simple Example

Adaptive Approximation Model: polynomial

\[ \hat{f}(x; \theta) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_M x^M = \sum_{j=0}^{M} \theta_j x^j \]

Learning: minimizing a cost function (or error function)

\[ J(\theta) = \frac{1}{2} \sum_{n=1}^{N} \left( \hat{f}(x_n; \theta) - y_n \right)^2 \]
Machine Learning - Simple Example

- Underfitting
- Underfitting
- Appropriate capacity
- Overfitting
underfitting and overfitting depends on both $M$ and $N$
Capacity, Overfitting and Underfitting

- The ability to perform well on previously unobserved inputs is called **generalization**. This is a key challenge in machine learning.
- Training set vs. test set. Typically, the generalization error (test error) is measuring the performance on a test set.
- The expected test error is greater than or equal to the expected training error

The performance of a machine learning algorithm is based on both:
1. Making the training error small
2. Making the gap between the training and test error small
Capacity, Overfitting and Underfitting

- **Occam’s Razor** – is a principle that states that among competing hypothesis that explain known observations equally well, we should choose the *simplest* one. Named after William of Ockham (1287-1347).

- **Statistical Learning Theory** provides various means of quantifying model capacity. The Vapnik-Chervonenkis dimension (VC dimension) is the most well known. However, it is not very practical with advanced machine learning algorithms.
Regularization

\[ J_R(\theta) = \frac{1}{2} \sum_{n=1}^{N} \left( \hat{f}(x_n; \theta) - y_n \right)^2 + \frac{\lambda}{2} \| \theta \|^2 \]
Hyperparameters and Validation Sets

- **Hyperparameters** – parameters that affect the performance of the algorithm but they are not adapted by the learning algorithm itself; e.g., the degree of the polynomial is a capacity hyperparameter; the value of $\lambda$ in regularization is another hyperparameter.

- **Validation Set** – is used to select the hyperparameters. Split the training set into two disjoint subsets; one subset of data is used to learn the parameters, the other subset (validation set) is used to estimate the generalization error in order to update the hyperparameters; e.g.; 80% of data is used for training and 20% for validation.

- Three types of data: **Training set, Validation set, Test set.**
Batch Learning vs. Online Learning

- **Batch Learning** – machine learning methods that are based on learning on the entire training data set.
- **Online Learning** – machine learning methods that are based on data becoming available in sequential order and is sued to update the parameters at each step. Data does not need to be stored after it is used to update the adjustable parameters.
- Something in between batch learning and online learning is to use a window of data (not the entire data) to update the parameters.
Mathematical Foundations for Machine Learning

- Linear Algebra
- Optimization
- Approximation Theory
- Probability Theory
- Random Signals and Systems
Optimization

- Optimization is fundamental in multiple science and engineering fields.
- Main idea is to take the derivative and set it to zero (but it soon becomes more complicating than that......)
Optimization

Optimization problem:

\[ \text{minimize } F(x) \quad \text{subject to } c_i(x) = 0 \quad i = 1, 2, \ldots, m' \]
\[ c_i(x) \geq 0 \quad i = m', m' + 1, \ldots, n \]

Some notation: \( F : \mathbb{R}^n \mapsto \mathbb{R} \)

Gradient of \( F \) : \( \nabla F = \left[ \frac{\partial F}{\partial x_1}, \frac{\partial F}{\partial x_2}, \ldots, \frac{\partial F}{\partial x_n} \right]^\top \)

Hessian matrix:

( \( n \times n \) symmetric matrix)

\[ \nabla^2 F = \frac{\partial^2 F}{\partial x_i \partial x_j} = \begin{bmatrix}
\frac{\partial^2 F}{\partial x_1^2} & \frac{\partial^2 F}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 F}{\partial x_1 \partial x_n} \\
\frac{\partial^2 F}{\partial x_2 \partial x_1} & \frac{\partial^2 F}{\partial x_2^2} & \cdots & \vdots \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial^2 F}{\partial x_n \partial x_1} & \cdots & \cdots & \frac{\partial^2 F}{\partial x_n^2}
\end{bmatrix}^\top \]
Optimization

Necessary and sufficient conditions for unconstrained optimization

\[
\text{minimize } f(x) \quad \text{(univariate case)}
\]

\[
\begin{cases}
  f'(x^*) = 0 \\
  f''(x^*) \geq 0
\end{cases}
\]

\text{Necessary conditions for } x^* \text{ to be a minimum}

\[
\begin{cases}
  f'(x^*) = 0 \\
  f''(x^*) > 0
\end{cases}
\]

\text{Sufficient conditions for } x^* \text{ to be a minimum}
Optimization

\[
\begin{align*}
\text{minimize } & \quad F(x) \\
\text{subject to } & \quad x \in \mathbb{R}^n \\
\end{align*}
\]

\[F(x) = F(x_1, x_2, \ldots, x_n) \quad \text{(multivariate case)}\]

\text{Necessary Conditions} \quad \begin{cases} 
\nabla F(x^*) = 0 \\
\nabla^2 F(x^*) \text{ is positive semi-definite}
\end{cases}

\text{Sufficient Conditions} \quad \begin{cases} 
\nabla F(x^*) = 0 \\
\nabla^2 F(x^*) \text{ is positive definite}
\end{cases}
Optimization

When is a matrix $A$ positive (semi)-definite?

$A$ is **positive-definite** ($A > 0$) if $x^T A x > 0$ for any non-zero $x$.

$A$ is **positive-semidefinite** ($A \geq 0$) if $x^T A x \geq 0 \ \forall x$.

**Lemma:** $A > 0 \iff \lambda_i(A) > 0 \ \forall i = 1, \ldots, n$

all the eigenvalues are positive
Optimization

**Convexity:**
The set $S$ is convex if \( \lambda x + (1 - \lambda)y \in S \) $\forall x, y \in S$, $\forall \lambda \in [0,1]$. $\lambda$ $\lambda$

A function $F : S \mapsto \mathbb{R}$ is convex over $S$ if $F(\lambda x + (1 - \lambda)y) \leq \lambda F(x) + (1 - \lambda)F(y)$ $\forall x, y \in S$, $\forall \lambda \in [0,1]$

**Lemma:** If $F \in C^1(-\infty, \infty)$ and $F$ is convex over $\mathbb{R}^n$ then $x^*$ is a global minimum of $F$ iff $\nabla F(x^*) = 0$. $\lambda$ $\lambda$
Optimization

Example: Quadratic function

\[
\text{minimize } \quad c^\top x + \frac{1}{2} x^\top G x
\]

\[c \in \mathbb{R}^n \quad G \in \mathbb{R}^{n \times n} \quad \text{symmetric} \]

\[\nabla F(x) = c + Gx \quad \Rightarrow \quad \nabla F(x^*) = 0 \quad \Rightarrow \quad Gx^* = -c\]

\[\nabla^2 F(x) = G \quad x^* \text{ is a minimum of } F(x) \quad \Rightarrow \begin{cases} Gx^* = -c \\ G \geq 0 \end{cases}\]

Note: \( F(x) \) is convex \( \Rightarrow G \geq 0 \)

Warning: not all functions that we want to minimize are smooth
Optimization

Contour plots:
\[ F(x) = \kappa \] for different values of \( \kappa \).

- \[ F(x_1, x_2) = x_1^2 + x_2^2 = \kappa \]

- \[ F(x_1, x_2) = e^{x_1} (4x_1^2 + 2x_2^2 + 4x_1x_2 + 2x_2 + 1) \]
Optimization

- **Linear Programming**: (Simplex method)

  Standard Form: \[
  \begin{align*}
  \text{minimize} & \quad c^\top x \\
  \text{subject to} & \quad Ax = b, \quad A, b, c \text{ are given.} \\
  & \quad x \geq 0
  \end{align*}
  \]

- **Unconstrained Minimization**: (Steepest descent; Newton method)

  \[\min_{x \in \mathbb{R}^n} F(x)\]

- **Constrained Minimization**: (Gradient Projection; Penalty methods; Lagrange methods)

  \[
  \begin{align*}
  \text{minimize} & \quad F(x) \\
  \text{subject to} & \quad c(x) = 0 \\
  & \quad d(x) \geq 0
  \end{align*}
  \]
Optimization

Optimization Algorithms

What are the performance criteria?
- Convergence
- Rate of convergence
- Complexity
Optimization

Algorithms for Unconstrained Minimization

$$\min_{x \in \mathbb{R}^n} F(x)$$

(A) Steepest Descent Method (Gradient Method)

Start $x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow \ldots x_k \rightarrow \ldots$

$$x_{k+1} = x_k - \lambda^*_k \nabla F(x_k)$$

$\lambda^*_k > 0$

$\lambda^*_k$ is determined be a linear search so that $x_{k+1}$ minimizes $F(x)$ in the direction $d_k = -\nabla F(x_k)$ from $x_k$. 
**Optimization**

**Trade-offs in choosing \( \lambda \):**
- speed
- convergence
- complexity

Simplest gradient descent:

\[
 x_{k+1} = x_k - \lambda \nabla F(x_k) \quad \lambda = \text{const.}
\]
Optimization

Intuition for Steepest Descent

1-dimensional
\[ x \in \mathbb{R} \]

2-dimensional
\[ x \in \mathbb{R}^2 \]

Contour lines
\[ F(x_1, x_2) = \kappa \]
Optimization

Example: \( F(x) = \frac{1}{2}x^2 \)

\[
\nabla F(x_k) = x_k
\]

\[
x_{k+1} = x_k - \lambda x_k
\]

\[
x_k = (1 - \lambda)^k x_0
\]

Example: \( F(x_1, x_2) = e^{x_1} \left( 4x_1^2 + 2x_2^2 + 4x_1x_2 + 2x_2 + 1 \right) \)

\[
\nabla F(x_1, x_2) = \begin{bmatrix} e^{x_1} \left( 4x_1^2 + 2x_2^2 + 4x_1x_2 + 8x_1 + 6x_2 + 1 \right) \\ e^{x_1} \left( 4x_1 + 4x_2 + 2 \right) \end{bmatrix}
\]

\[
\begin{bmatrix} x_{k+1} \\ \end{bmatrix} = \begin{bmatrix} x_k \\ \end{bmatrix} - \lambda_k^* \begin{bmatrix} \nabla F(x_k) \\ \end{bmatrix}
\]
**Optimization**

In continuous-time: \( x_{k+1} = x_k - \lambda \nabla F(x_k) \Rightarrow \frac{x_{k+1} - x_k}{\lambda} = -\nabla F(x_k) \)

as \( \lambda \to 0 \)

\[ \dot{x} = -\nabla F(x) \quad x(0) = x^0 \]

After rescaling: \( \dot{x} = -\Gamma \nabla F(x) \quad \Gamma = \Gamma^\top > 0 \) \( \Gamma \) is symmetric & positive definite

Example: \( \min_{x \in \mathbb{R}^n} x^\top G x = F(x) \)

\[ \nabla F(x) = Gx \Rightarrow \dot{x} = -Gx \quad x(0) = x^0 \]

\[ x(t) = e^{-Gt}x^0 \]
(B) Newton's Method: \[ \min_{x \in \mathbb{R}^n} F(x) \quad (F \text{ is convex}) \]
\[ \nabla F(x^*) = 0 \quad \text{(necessary condition)} \]

\[ \nabla F(x) \approx \nabla F(x_k) + \frac{d}{dx} \nabla F(x) \bigg|_{x=x_k} (x - x_k) \]

Hessian

Let \[ H(x) := \nabla^2 F(x) \]

\[ \nabla F(x) \approx \nabla F(x_k) + H(x_k)(x - x_k) = 0 \quad \text{(we want)} \]

\[ 0 = \nabla F(x_{k+1}) = \nabla F(x_k) + H(x_k)(x_{k+1} - x_k) \]

\[ x_{k+1} = x_k - H(x_k)^{-1} \nabla F(x_k) \quad \text{(assuming } H(x_k)^{-1} \text{ exists)} \]
Optimization

In continuous-time: \[ \dot{x} = -\beta H(x)^{-1} \nabla F(x) \quad ; \quad \beta > 0 \]

**Example:** \[ F(x) = \frac{1}{2} x^2 \]

\[ \nabla F(x) = x \quad H(x) = 1 \quad H(x)^{-1} = 1 \]

\[ x_{k+1} = x_k - H(x_k)^{-1} \nabla F(x_k) \]

\[ x_{k+1} = x_k - x_k = 0. \quad x_0 \rightarrow x_1 = 0 \]

Convergence in one step!
Optimization

Example: \( F(x_1, x_2) = e^{x_1} \left( 4x_1^2 + 2x_2^2 + 4x_1x_2 + 2x_2 + 1 \right) \)

\[
\nabla F(x_1, x_2) = \begin{bmatrix}
e^{x_1} \left( 4x_1^2 + 2x_2^2 + 4x_1x_2 + 8x_1 + 6x_2 + 1 \right) \\
e^{x_1} (4x_1 + 4x_2 + 2)
\end{bmatrix}
\]

\[
H(x_1, x_2) = e^{x_1} \begin{bmatrix}
4x_1^2 + 2x_2^2 + 4x_1x_2 + 16x_1 + 10x_2 + 9 & 4x_1 + 4x_2 + 6 \\
4x_1 + 4x_2 + 6 & 4
\end{bmatrix}
\]

\[
H(x_1, x_2)^{-1} = \frac{-e^{x_1}}{8(x_2^2 + 2x_1x_2 + 2x_1 - x_2)} \begin{bmatrix}
4 & -(4x_1 + 4x_2 + 6) \\
-(4x_1 + 4x_2 + 6) & 4x_1^2 + 2x_2^2 + 4x_1x_2 + 16x_1 + 10x_2 + 9
\end{bmatrix}
\]

\[
x_{k+1} = x_k - H(x_k)^{-1} \nabla F(x_k)
\]