Basics of machine learning
Nicolas Duchateau, Rémi Emonet, Carole Lartizien

Deep Learning for Medical Imaging
Lyon (FR) - 15/04/2019

Program

1. Introduction
Carole Lartizien
~30min

2. Supervised learning
Rémi Emonet + Carole Lartizien
~75min

3. Unsupervised learning
Nicolas Duchateau + Rémi Emonet
~75min

4. Methods evaluation
Carole Lartizien + Rémi Emonet + Nicolas Duchateau
~30min

5. Conclusion/To go further
~30min

1. Introduction

1. Scientific and medical context
2. Basics of machine learning
3. Some Historical highlights of AI
1. Introduction

Scientific and medical context

- Refine Diagnosis
- Improve therapy planning
- Predict therapy outcome
- Develop preventive medicine.

Some examples

Example 1 = Automatic segmentation?
Example 2 = Differences between healthy and...?
Example 3 = Predict infarct location from myocardial deformation?
Example 4 = Detect outliers in a coherent population?

From imaging data to wisdom

- Diagnostic and Prognostic models
- Region based feature extraction (texture, quantitative or semi-quantitative, gradient...)
- Pixel based feature extraction (texture, quantitative or semi-quantitative, gradient...)
- Multi-modality imaging

Some examples

Example 1 = Automatic segmentation?
Example 2 = Differences between healthy and...?
Example 3 = Predict infarct location from myocardial deformation?
Example 4 = Detect outliers in a coherent population?
1. Introduction

Some examples

Example 1 = Automatic segmentation?
Example 2 = Differences between healthy and...?
Example 3 = Predict infarct location from myocardial deformation?
Example 4 = Detect outliers in a coherent population?

1. Scientific and medical context
2. Basics of machine learning
3. Some Historical highlights of AI
1. Introduction

Basics of machine learning

1. Define a task T
2. Formulate this task as a decision model
3. Learn the decision model based on samples (Data D) and a performance metric P
4. Infer decision from this model on new samples

1. Introduction

Basics of machine learning ...  

2. Problem formulation within the statistical decision framework  

→ Binary supervised classification problem  

Design your model:
- Which features?
- Which classifier model?

1. Introduction

Basics of machine learning ...

3. Learn the decision model based on training samples and performance metric  

→ Feature extraction ➔ Classification ➔ Training input ➔ Reference ➔ Error ➔ Loss

1. Introduction

Basics of machine learning ...

1. Task definition
- Detect lesions on brain T1 MRI

2. Problem formulation as a decision task
- Decide whether each voxel of the brain MR scan is a 'lesion' or 'normal tissue'
  - Binary classification problem
- Depending on the available samples, consider this problem as
  - supervised,
  - unsupervised or
  - weakly supervised learning
1. Introduction

Basics of machine learning ...

4. Infer decision on new samples

...and deep learning

1. Scientific and medical context
2. Basics of machine learning
3. Some Historical highlights of AI

Artificial Intelligence

Machines learn to reproduce specific tasks

Deep Learning
Machines adapt themselves to data

Symbolic AI
Human-like intelligence in a machine

Connectionist AI

1950
Expert systems
Perceptron...

1980
Neural networks
SVM
Decision trees...

2010
GP
Web database (ImageNet..)
1. Introduction

Artificial Intelligence

Symbolic AI
Human-like intelligence in a machine

Machine Learning

Connectionist AI
Machines learn to reproduce specific tasks

Deep Learning
Machines adapt themselves to data

1950
Expert systems
Penrose...

1980
Neural networks
SVM
Decision trees...

2010
GPU
Web database (ImageNet...)

1. Introduction

Deep learning for medical image analysis

Figure 1: Evolution of the papers included in this survey in year of publication, task addressed (Section 3), imaging modality, and application area (Section 6).

Program

~30min 1. Introduction
Carole Lartizien

~75min 2. Supervised learning
Rémi Emonet + Carole Lartizien

~75min 3. Unsupervised learning
Nicolas Duchateau + Rémi Emonet

~30min 4. Methods evaluation
Carole Lartizien + Rémi Emonet + Nicolas Duchateau

~30min 5. Conclusion / To go further

[Litjens, MEDIA 17]
2. Supervised learning

i. Use case
ii. Standard pipeline
iii. Learning a decision function
iv. Decision model based on the minimization of the misclassification error
v. Decision trees
vi. Neural networks

Case study:

- Material: MRI brain images databases with hippocampes manually annotated by experts
- Problem definition
  - To automatically segment hippocampe in MRI T1 images
- Problem formulation as a decision task
  - Decide whether each voxel belongs to the hippocampe structure or not
  - Binary classification problem

Annotated learning database
Extract discriminant feature from the raw image
Learn a decision function
2. Supervised learning

i. Use case

ii. Standard pipeline

iii. Learning a decision function

iv. Decision model based on the minimization of the misclassification error

v. Decision trees

vi. Neural networks

2. Supervised learning

Feature vector $x_i \in X = \mathbb{R}^d$

Label $y_i \in Y = Z$

Preprocessed Data

Feature extraction

Classification

Data ready to be analyzed?

Spatially

Temporally

Different:
- sampling
- geometries
- dynamics

= normalize first!

Duchateau et al. Med Image Anal 2011
De Craene et al. ISBI 2012.
2. Supervised learning

Feature vector \( x_i \in X = \mathbb{R}^d \)

Label \( y_i \in Y = \mathbb{Z} \)

Decision function \( f(x_i) \)

---

Feature extraction

Learning... but on what?

"Known" descriptors / features... or to discover automatically?

**Images**
- Gray level, texture, ...

**Shapes**
- Geometry (meshes, curvature,...), fibers, ...

**Functional features**
- Global: clinical measurements, outcome, ...
- Local: mechanical (motion / deformation), electrical, ...

- Physiology, specific structure (manifold)
- 4D (space + time) ... or 5D (longitudinal data)
- High dimensionality
- Population size: from ~20 to 500+ subjects

---

Learning a decision function in the feature space \( X = \mathbb{R}^d \)

---

This topic will not be covered during this lecture!
2. Supervised learning

Classification versus regression

\[ y \in \mathbb{N} \rightarrow \text{Classification} \]

\[ y \in \mathbb{R} \rightarrow \text{Regression} \]

In this lecture, we focus on classification models.
To simplify, we consider a binary classification problem.

2. Supervised learning

Objective

To learn a function \( f \) that maps an input \( x \) to an output \( y \) based on a series of annotated samples \( S = \{(x_1, y_1), \ldots, (x_n, y_n)\} \):

- \( f \) is an element of some space of possible functions, usually called the hypothesis space.
- Usually, the class \( y \) is not directly outputted
  - \( f \) is either a scoring function e.g. a signed distance to the hyperplane,
    \[ f : \mathbb{R} \rightarrow \mathbb{R} \]
  - Or \( f \) is a probability of \( x \) belonging to class \( y \)
    \[ f : \mathbb{A} \times \mathbb{Y} \rightarrow \mathbb{R} \]
- The output \( y \) is defined by a decision rule applied on the output of the scoring function
  \[ D(x) = \text{sign}(f(x)) \]

2. Supervised learning

Supervised learning in a nutshell

- Split the sample dataset into three parts: a training, a validation and a test dataset.
- Choose a parameterized model function with parameters \( \Theta_1 \) and hyperparameters \( \Theta_2 \) from an hypothesis space \( \mathcal{H} \).
- Fit the model parameters \( \Theta_1 \) to the training dataset for a fixed value of \( \Theta_2 \).
  - Choose an error function that measures the misfit between the decision function \( D(f(x)) \) and the class \( y_i \) of all training data points \( (x_i, y_i) \)
    - Minimize the error function.
- Evaluate the performance of your model on the validation dataset.
- Retrain your model with another hyperparameter set \( \Theta_2 \).
- Select the best parameter set.
- Evaluate the performance of your best model on the test dataset.
2. Supervised learning

Supervised learning in a nutshell

- Split the sample dataset into three parts: a training, a validation and a test dataset.
- Choose a parameterized model function with parameters $\theta_1$ and hyperparameters $\theta_2$ from an hypothesis space $\mathbb{H}$.
- Fit the model parameters $\theta_1$ to the training dataset for a fixed value of $\theta_2$.
  - Choose an error function that measures the misfit between the decision function $D(f(x_i))$ and the class $y_i$ of all training data points $(x_i, y_i)$.
  - Minimize the error function.
- Evaluate the performance of your model on the validation dataset.
- Retrain your model with another hyperparameter set $\theta_2$.
- Select the best parameter set.
- Evaluate the performance of your best model on the test dataset.

2. Supervised learning

How to choose and fit the decision function $f(x)$

There are different approaches to the classification problem.

- Two types of decision models:
  - Linear models: linear SVM, logistic regression, linear discriminant analysis.
  - Non linear models: neural networks, kernel machine, decision trees.

- Different strategies to minimize the error function:
  - Global minimization: In the original feature space $\mathbb{R}^d$.
  - Recursive minimisation: based on a recursive method applied in a one-dimensional space (e.g. decision trees).

- Two types of error functions:
  - Misclassification Error Risk:
    - Bayesian classifier, SVM, logistic regression, neural networks.
  - Other functionals:
    - Fisher criterion for discriminant linear analysis (LDA).
    - Entropy for decision trees or neural networks.
    - ...
2. Supervised learning

Hands-on session = Tuesday

Classification of diseased vs. healthy based on cardiac shapes / function

![Image]

2. Supervised learning

Risk minimisation

Statistical learning theory is based on the notion of risk $R$ also referred to as prediction error.

Parameters of the decision function $f$ for a given classification task are derived from the minimization of the prediction error between the estimated class labels $f(x_i)$ and the true class labels $y_i$.

$$R(f) = \mathbb{E}[L(Y, f(X))] = \int L(y_i, f(x_i)) p(x_i, y_i) dx_idy_i$$

$L(\cdot, \cdot)$ is a cost function quantifying the cost of the prediction error.

$p(x_i, y_i)$ is the joint probability of observing $x_i$ and $y_i$.

2. Supervised learning

Risk minimisation – Discriminative models

The decision function $f(x)$ is estimated directly, i.e.

- Without modeling and estimating the posterior probability densities.
- By modeling directly the decision function and estimating the parameters of this function based on training samples.

2. Supervised learning

i. Use case

ii. Standard pipeline

iii. Learning a decision function

iv. Decision model based on the minimization of the misclassification error

v. Decision trees

vi. Neural networks
2. Supervised learning

Empirical risk minimisation

- As seen above, the minimisation of risk \( R \) requires to estimate the joint probability distribution, which may not be trivial

\[
R(f) = \mathbb{E}[L(Y, f(X))] = \int P(x, y) L(y, f(x)) dx dy
\]

- An alternative is to minimize the empirical risk \( R_{emp}(f) \) based on the learning data samples

\[
R_{emp}(f) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i))
\]

\[
\min_{f \in H} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i))
\]

2. Supervised learning

Risk versus empirical risk minimisation

- Minimisation of the empirical risk minimisation under constraint of good generalization performance

\[
\min_{f \in H} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) + \lambda \Omega(f)
\]

\[\text{Loss function} \quad \text{Regularisation function}\]

2. Supervised learning

Structural risk minimisation

- Minimisation of the empirical risk minimisation under constraint of good generalization performance

\[
\min_{f \in H} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) + \lambda \Omega(f)
\]

\[\text{Loss function} \quad \text{Regularisation function}\]
To solve this minimisation problem under constraints, we make some hypothesis:

- On the model of the decision function \( f \):
  - \( f \) is assumed to be a linear hyperplane in the feature space \( \mathcal{X} = \mathbb{R}^d \)
    \[
    f : \mathcal{X} \to \mathbb{R}, \quad x_i \mapsto w^T x_i + b
    \]
- On the loss function \( L \) and the regularisation function \( \Omega \):

2. Supervised learning

Structural risk minimisation

- To solve this minimisation problem under constraints, we make some hypothesis:
  - On the model of the decision function \( f \):
    - \( f \) is assumed to be a linear hyperplane in the feature space \( \mathcal{X} = \mathbb{R}^d \)
    \[
    f : \mathcal{X} \to \mathbb{R}, \quad x_i \mapsto w^T x_i + b
    \]
  - On the loss function \( L \) and the regularisation function \( \Omega \):

2. Supervised learning

Exemple cost functions

- Loss 0-1
  \[
  L(y, f(x)) = \left\{ \begin{array}{ll}
  0 & \text{if } y f(x) > 0 \\
  1 - \text{sgn}(y f(x)) / 2 & \text{otherwise}
  \end{array} \right.
  \]
- Hinge Loss
  \[
  L(y, f(x)) = \max(0, 1 - y f(x))
  \]
- Quadratic loss
  \[
  L(y, f(x)) = \max(0, 1 - y f(x))^2
  \]
- Logistic loss
  \[
  L(y, f(x)) = \ln(1 + \exp(-y f(x)))
  \]

2. Supervised learning

Exemple regularisation functions

When \( f(x) \) is a linear decision function of the form
\[
 f(x) = w^T x + b
\]
with \( w \) the normal to the hyperplane and \( b \) a term of bias.

The regularisation function belongs to the family of \( \ell_p \) norms
\[
\Omega_p(w) = |w|_p = \left( \sum_{i=1}^d |w_i|^p \right)^{1/p}
\]
For \( p=1 \) (\( \ell_1 \) norm) and \( p=2 \) (\( \ell_2 \) norm)
No closed form solution for $f$ in most of the cases

Quadratic problem that can be solved with a standard optimization algorithm, eg conjugate gradients

Structural risk minimisation

$$\min_{f \in L} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) + \lambda \Omega(f)$$

If $L(.,.)$ and $\Omega(.,.)$ are convex functions

- No closed form solution for $f$ in most of the cases
- Quadratic problem that can be solved with a standard optimization algorithm, eg conjugate gradients

Exemple : Linear SVM

Problem formulation

$$\begin{cases} \max_{w, b} & \frac{1}{n} \sum_{i=1}^{n} \max_{i \in \{0,1\}} (y_i(w^T x_i + b) - m) \\ \text{s.t.} & \|w\|_2 \leq 1 \end{cases}$$

Ill-posed problem: if $(w, b)$ is a solution, then $(k^* w, k^* b)$, $\forall k > 0$ is a solution too

We define:

$$w = \frac{v}{\|v\|} \text{ and } b = \frac{a}{\|v\|}$$

Canonical problem formulation

$$\begin{cases} \min_{w, b} & \frac{1}{n} \sum_{i=1}^{n} \max_{i \in \{0,1\}} (y_i(w^T x_i + b) - 1) \\ \text{s.t.} & \|w\|_2 \leq 1 \end{cases}$$

Find a linear hyperplane $f(x) = v^T x + a$ that maximises the margin $m$ between the training samples of both classes

The margin $m$ is the smallest distance of any training sample to the decision hyperplane
2. Supervised learning

Exemple : Linear SVM

When the data are almost linearly separable:

Errors are modeled as **positive slack variables** $\xi_i$ associated to each sample $(x_i, y_i)$ and measuring the distance to the margin.

- No error: $y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \Rightarrow \xi_i = 0$
- Error: $y_i(\mathbf{w}^T \mathbf{x}_i + b) < 1 \Rightarrow \xi_i = 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b)$

**Hinge loss**

$\xi_i = \max(0, 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b))$

Minimize $\frac{1}{2} \| \mathbf{w} \|^2 + C \sum_i \xi_i$

subject to $y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i$, $i = 1, 2, \ldots, n$

$\xi_i \geq 0$

Equation of the decision function is a weighted sum of scalar products between pairs of vectors of the redescription space $\mathcal{H}$:

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i \langle \Phi(x_i), \Phi(x) \rangle + b$$

Where $\alpha_i$ are the Lagrange coefficients.

This enables to use the kernel trick.

Generalisation to nonlinear problem

Find a mapping function $\Phi$ that maps the data from the original representation space $\mathcal{X}$ into a redescription space $\mathcal{H}$ of higher dimension where the classification problem is linear i.e. the decision function may be written as $h(x) = \mathbf{w}^T \Phi(x) + b$. 
SVM and kernel trick

Instead of defining a non-linear projection $\Phi$, we use a kernel function associated to the projection function:

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle$$

A kernel function $k$ is a similarity function. It has to satisfy some properties referred to as the Mercer conditions to guarantee the existence of the corresponding function $\Phi$

- Symmetry
  $$k(x, x') = k(x', x)$$
- Positive definite
  $$\sum_{i,j} \alpha_i \alpha_j k(x_i, x_j) > 0, \forall x_i \in \mathcal{X}, \forall \alpha \in \mathbb{R}$$

2. Supervised learning

Non linear SVM with kernel formulation

The dual problem is reformulated as

$$\max_{\alpha} -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j) + \sum_{i=1}^{n} \alpha_i$$

with

- $0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n$
- $\sum_{i=1}^{n} \alpha_i y_i = 0$

With solution:

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i k(x_i, x) + b$$

SVM and kernel trick

- Advantages of the kernel function:
  - The computation of the kernel function is performed in the native representation space $\mathcal{X}$, which is less computationally intensive than performing a scalar product in a high-dimensional space.
  - Projection $\Phi$ does not need to be explicitly formulated. It is thus possible to consider complex project in potentially infinite redescription space.
  - The kernel function is constructed based on the Mercer conditions without formulating the corresponding projection function $\Phi$.

Some standard kernels

- Linear kernel: trivial case equivalent to linear classifier.
  $$k(x_i, x_j) = x_i^T x_j$$
- Polynomial kernel
  $$k(x_i, x_j) = (x_i^T x_j + 1)^d$$
- Gaussian kernel
  $$k(x_i, x_j) = \exp\left(-\frac{||x_i - x_j||}{2\sigma^2}\right)$$
2. Supervised learning

Risk minimisation - Generative models

- Make some hypothesis on the distribution of the conditional probabilities $P(X = x_i | Y = k)$ and priors $P(Y = k)$
- Learn the conditional probabilities on the training database
- Estimate the posterior probabilities based on the Bayes Theorem

$$P(Y = 1 | X = x_i) = \frac{P(X = x_i | Y = 1)P(Y = 1)}{P(X = x_i | Y = 1)P(Y = 1) + P(X = x_i | Y = -1)P(Y = -1)}$$

2. Supervised learning

Bayesian classifier

The Bayes classifier minimizes the risk of classifying sample $x_i$ in class $k$ as

$$\arg \min_{k \in [0, \ldots, L]} R(k, x_i)$$

$$R(k, x_i) = \sum_{j=0}^{L} L(k, j)P(Y = j | X = x_i)$$

$L(k, j)$ The cost of assigning a class $j$ to any sample belonging to class $k$

$P(Y = j | X = x_i)$ The posterior probability of assigning class $j$ to sample $x_i$. 

2. Supervised learning

Exemple Nonlinear discriminant analysis using kernel function operator

- Kernel Fisher discriminant analysis
- Kernel logistic regression
- ...
Le classifieur bayésien

Dans le cas d’un problème de classification binaire, l’expression du risque pour chaque classe est
\[ R(1, x_i) = L(1,1) \mathbb{P}(Y = 1 | X = x_i) + L(-1,1) \mathbb{P}(Y = -1 | X = x_i) \]
\[ R(-1, x_i) = L(-1,1) \mathbb{P}(Y = 1 | X = x_i) + L(1,1) \mathbb{P}(Y = -1 | X = x_i) \]
L’étiquette de \( x_i \) sera \( y_{i,*} \) si et seulement si
\[ (L(1,1) - L(-1,1)) \mathbb{P}(Y = 1 | X = x_i) < (L(-1,1) - L(1,1)) \mathbb{P}(Y = -1 | X = x_i) \]
En supposant que \( L(1,1) - L(-1,1) < 0 \), on obtient
\[ \frac{\mathbb{P}(Y = 1 | X = x_i)}{\mathbb{P}(Y = -1 | X = x_i)} > \frac{L(1,1) - L(-1,1)}{L(-1,1) - L(1,1)} \]
Le rapport des probabilités postérieures

2. Supervised learning

Bayesian classifier

For a binary classification problem, the decision function is
\[ f(x_i) = \ln \left( \frac{\mathbb{P}(X = x_i | Y = 1)}{\mathbb{P}(X = x_i | Y = -1)} \right) \]
The corresponding decision rule is
\[ D(x_i) = \begin{cases} 1 & f(x_i) \geq k \\ -1 & \text{sinon} \end{cases} \]

2. Supervised learning

Naive Bayes classifier

Hypothesis of conditional independence between every pair of features given the value of the class variable.
\[ x_i \perp x_j, \forall (j, k) \in \{1, \ldots, d\}, \forall i \in \{1, \ldots, n\} \quad x_i \in \mathbb{R}^d \quad x_i = \left( x_{i,1}, x_{i,2}, \ldots, x_{i,d} \right) \]
The Bayes theorem gives
\[ \mathbb{P}(Y = k | X = x_i) = \frac{\mathbb{P}(X = x_i | Y = k) \mathbb{P}(Y = k)}{\sum_{j} \mathbb{P}(X = x_i | Y = j) \mathbb{P}(Y = j)} \]
Using the naive conditional independence assumption
\[ \mathbb{P}(Y = k | X = x_i) = \frac{\prod_{t=1}^{d} \mathbb{P}(X^t = x_{i,t} | Y = k)}{\sum_{j} \prod_{t=1}^{d} \mathbb{P}(X^t = x_{i,t} | Y = j)} \]
The denominator is constant given the input
\[ \mathbb{P}(Y = k | X = x_i) \propto \mathbb{P}(Y = k) \prod_{t=1}^{d} \mathbb{P}(X^t = x_{i,t} | Y = k) \]
2. Supervised learning

Naive Bayes classifier

we can use the following classification rule:

$$y_i = \arg \max_j \mathbb{P}(Y = j) \prod_{k=1}^{n} \mathbb{P}(X_i^k = x_i^k | Y = j)$$

$$x_i = (x_i^1, x_i^2, \ldots, x_i^n)$$

- The independent conditional probabilities are estimated separately for each feature
- The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of the conditional probabilities

2. Supervised learning

Models based on risk minimisation: Advantages and Limitations

→ Advantages
  - Quiet flexible inputs: based on kernel computation
  - Interpretable (somewhat): people are able to understand decision tree models
  - Produce an exact solution
  - Kernel trick to efficiently compute non-linear models
  - Quite robust to small size and unbalanced training datasets

→ Limitations
  - Low interpretability: difficult to extract the most discriminant features
  - Problem with large scaled datasets

2. Supervised learning

Decision Trees

i. Use case
ii. Standard pipeline
iii. Learning a decision function
iv. Decision model based on the minimization of the misclassification error
v. Decision trees
vi. Neural networks
2. Supervised learning

Decision Trees: Motivation
Existing decision trees based on clinical evidence and expertise, use elementary features and manual thresholds.

2. Supervised learning

Decision Tree in Computer Science

➔ A model describing a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ that
- to any input value/point $x \in \mathcal{X}$
- associates $f(x) \in \mathcal{Y}$

➔ A Tree-shaped representation
- a root node, other split nodes, and leaves
- each split $n$ has a test function $d^n(x)$ that gives a child index
  usually using a single coordinate (e.g., $x_8 \leq 42$)
- each leaf $l$ has a prediction model
  usually very simple (e.g. constant value) $x \mapsto f_l(x)$

➔ A simple way of computing the output (for a point $x$)
- start at the root
- if in a split node $n$, move to child $d^n(x)$
- if in a leaf node $l$, return $f_l(x)$

What is a Decision Tree (DT)?

➔ Big idea of DT
- Use very elementary decision rules
- Combine them into a tree

➔ Example: binary DT for sport practice

Decision Tree in Computer Science
2. Supervised learning

Decision Tree in Computer Science

Good versus Bad Splits for Decision Trees?
which is a better split?

A                                   B                                  C

Toward finding the best test/split (for building classification trees)
➔ Find test for which children are as “pure” as possible
➔ Entropy as purity (borrowed from the information theory)
   ◆ Entropy is a measure of “missing information”
   ◆ More precisely, the number of bits needed to represent the missing information,
   ◆ ... on average, using the optimal encoding
➔ Entropy definition
   ◆ given a set $S$
   ◆ with instances belonging to class $c$ with prob $p_c$
   ◆ we have:
   $$Entropy(S) = - \sum_c p_c \log_2(p_c)$$

Learning Optimal Decision Trees
➔ Very complex (NP-complete), even for simple definitions of “optimal”
➔ Use of heuristics and of a greedy Top-Down approach
➔ Principle of TDIDT (Top-Down Induction/learning of DT)
   ◆ Start with an empty tree and all examples (dataset)
   ◆ Find a good test
     • good test?
     • examples with same class fall on the same side
     • or, similar examples fall on the same side
     • for each possible test outcome, create child node
   ◆ Move each example to a child, according to the test outcome
   ◆ Repeat for each child that is not “pure”
➔ Main question
   ◆ how to decide which test/split is “best”
2. Supervised learning

Entropy

- Considering a node $n$, with a part of the dataset
- Denoting $p = \text{proportion of instances of class } +1 \text{ in set } n$
- Note that if $p=0$, $p \times \log_2(p)$ is undefined but tends to 0
- The maximum entropy of 1 is reached when the 2 classes are perfectly mixed

\[
entropy(S) = -\sum_c p_c \log_2(p_c)
\]

2. Supervised learning

Information gain

- Heuristic for choosing a test in a node
  - (on average over the children)
  - on average, provides most information about the class
  - on average, reduces the class entropy the most
  - expected reduction of entropy = information gain

\[
Gain(S, A) = \text{Entropy}(S) - \sum_v \frac{|S_v|}{|S|} \text{Entropy}(S_v)
\]

2. Supervised learning

Good versus Bad Splits for Decision Trees?

Information gain

\[
Gain(S, A) = \text{Entropy}(S) - \sum_v \frac{|S_v|}{|S|} \text{Entropy}(S_v)
\]

Other purity measure/gain (alternative to entropy)

- Gini impurity index
  - (not to be confused with gini coefficient)
  - “measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset” (lower is better).

\[
\text{Gini}(S) = \sum_c p_c(1 - p_c) = \sum_c p_c - \sum_c p_c^2 = 1 - \sum_c p_c^2
\]

- (for binary classification) $\text{Gini}(S) = 2p(1 - p)$

\[
Gain(S, A) = \text{Gini}(S) - \sum_v \frac{|S_v|}{|S|} \text{Gini}(S_v)
\]
2. Supervised learning

Nature of Decision Tree Inputs \( x \in \mathcal{X} \)
- \( \mathcal{X} \) can have any number of coordinates with arbitrary types
  - numbers, e.g., \( x_i \in \mathbb{N} \) or \( x_i \in \mathbb{R} \)
  - categorical variable, e.g.,
    - \( x_i \in \{TRUE, FALSE\} \)
    - \( x_i \in \{-1, 1\} \)
    - \( x_i \in \{rainy, snowy, sunny\} \)
    - \( x_i \in \{monday, tuesday, …, sunday\} \)
    - \( x_i \in \{yes, no\} \)
  - and others (templates, binary image patches, …)
- Test functions \( d^n(x) \) can take various forms
  - two children: equal or not?
    \[ d^n : x \rightarrow (x_i = cst) \]
  - two children: greater than?
    \[ d^n : x \rightarrow (x_i \geq cst) \]
  - one child per possible outcome:
    - e.g., 3 children with indices
      \[ \text{snowy, sunny, rainy} \]

Nature of Decision Tree Outputs \( y \in \mathcal{Y} \)
- Categorical output, e.g.,
  - Binary classification, \( y \in \{-1, 1\} \)
  - Multiclass classification, \( y \in \{dog, cat, car, truck, …\} \)
  - Rating: 1 to 5 stars
- Leaf \( f^l(x) = cst_l \) (one of the outcomes)
- Numerical output, e.g.,
  - Regression, \( y \in \mathbb{R} \)
  - Multi-dimensional regression, e.g., \( y \in \mathbb{R}^D \)
  - Count-regression
- Leaf \( f^l(x) = cst_l \) or \( f^l(x) = w^T_x + b \) (affine, …)
- Classification and regression trees, but also clustering trees...
- NB: we focused on classification trees

2. Supervised learning

DT: Advantages and Limitations
- Advantages
  - Flexible input: numerical and categorical data, no need for normalization, no assumptions, etc
  - Interpretable (somewhat): people are able to understand decision tree models
  - White-box: easy to know why a decision is taken
  - Performs well on large datasets
  - Universal approximator
  - Automatic feature selection
- Limitations
  - Lack of robustness: small change in the training data \( \Rightarrow \) possible large change in the tree
  - NP-complete problem requires heuristics and greedy algorithms
  - Need to take care of “imbalanced” categorical features
  - Need to take care of the overfitting

Avoiding overfitting with DT
- Option 1
  - stop adding nodes when overfitting starts occurring
  - needs a stopping criterion:
    - predefined (max-depth, min-leaf-size)
    - using a validation set
    - using statistical tests or MDL (minimum description length)
- Option 2
  - don’t bother about overfitting when growing the tree
  - after the tree has been built, start pruning it
  - prune to get better trees (validation)
2. Supervised learning

Random Forests (RF), in a few words
➔ Ensemble learning, bagging: multiple trees (M trees)
  • Random dataset (bootstrap: same size, drawn with replacement)
  • Random features (m features) at each split
  • Fully grown trees (no pruning)

➔ Prediction using a majority vote, or average
➔ NB: bagging allow to estimate the error of each tree

➔ Fast and robust to overfitting
➔ But, non-interpretative
may still overfit in case of noise

➔ Prediction using a majority vote, or average
➔ NB: bagging allow to estimate the error of each tree

➔ Fast and robust to overfitting
➔ But, non-interpretative
may still overfit in case of noise

2. Supervised learning

Example of Decision Forests

Structured Decision Forests for Multi-modal Ultrasound Image Registration
Ozan Dey et al., MICCAI 2015

Fig. 2. Structured decision tree training procedure, label patches are clustered at each node split (a). Mid-ventricle (b), mid-septal (d) and mid-lateral (e) wall landmark localization by using PEsMs (in green)(c) and regression nodes.

2. Supervised learning

Neural Networks

i. Use case
ii. Standard pipeline
iii. Learning a decision function
iv. Decision model based on the minimization of the misclassification error
v. Decision trees
vi. Neural networks
2. Supervised learning

Biological Neurons (40s)

- A single neuron
  - Threshold on a sum of inputs
- Complex organisation
  - Thousands of inputs
  - Billions of connections
  - 3D layout
- Of much interest
  - Biology
  - Neuroscience
  - Neuropsychology
  - Artificial intelligence
  - ...

Artificial Neural Networks: some history

- Started in the 50s
- Became more popular in the 80s
  (*backpropagation* in 1975)
  - Rumelhart, Hinton, and McClelland (1986)
    *A General Framework for Parallel Distributed Processing: explorations in the microstructure of cognition*
- Big slow down in the 90s
- 2010s
  - More data, more processing power (GPU)
  - Advances in optimization, architecture (convolution, ReLU, skip conn.)
  - “Deep Neural Networks”
  - State of the art performance in image, video, audio, … processing

Single Neuron, The Perceptron

\[
o = \varphi \left( w_0 + \sum_{i=1}^{n} w_i x_i \right) \\
o = \varphi \left( \sum_{i=0}^{n} w_i x_i \right) \\
x_0 = 1
\]
2. Supervised learning

Multiple Outputs: Fully Connected Layer

→ One “perceptron” per output
→ Different weights for each output

\[ o_1 = \varphi \left( w_0^1 + \sum_{i=1}^{n} w_i^1 x_i \right) \]
\[ o_2 = \varphi \left( w_0^2 + \sum_{i=1}^{n} w_i^2 x_i \right) \]

2. Supervised learning

Two-layer Perceptron

→ \( h_j = \varphi \left( w_0^j + \sum_{i=1}^{n} w_i^j x_i \right) \)

\[ o = \varphi \left( w_0^o + \sum_{j=1}^{4} w_j^o h_j \right) \]

2. Supervised learning

Multilayer Perceptron (MLP)

Expressive Power of Multilayer Perceptrons
(universal approximation theorem)

We can approximate any continuous function with a multilayer perceptron that has a single hidden layer (not “deep”) but that is sufficiently wide (a lot of neurons on the hidden layer)

→ Question: should we prefer adding
  ◆ more layers (deeper)?
  ◆ more neurons in a single hidden layer (wider)?
→ Deeper networks generalize better

⇒ Most probably because they create successive abstractions (observed empirically, on many real problems)
2. Supervised learning

Training Neural Networks (finding $\theta$, a good set of weights)

- Originally, the perceptron algorithm
- Today, mainly, gradient descent (and variants)
  - We want to optimize $\mathcal{L}(\theta) = \sum_i l(f_\theta(x^i), y^i)$
  - Start with random weights $\theta^0$

$\theta^{t+1} = \theta^t - \gamma \nabla_{\theta} \mathcal{L}(\theta^t)$

Mini-batch Gradient Descent iterates over

$\mathcal{L}_B(\theta^t) = \sum_{i \in B} l(f_\theta(x^i), y^i)$

Each iteration considers a random minibatch of points $B$
- we have to choose a minibatch size, e.g. $|B| = 64$
- Stochastic gradient descent SGD: single sample batch

More About Deep Neural Networks

........ during the whole week