Data Science for Actuaries (ACT6100)

Arthur Charpentier

Intro # 2 (Glossaire)

automne 2020

https://github.com/freakonometrics/ACT6100/

Supervised vs. Non-Supervised

Supervised Learning

In supervised learning, the algorithms learns a function that maps an input to an output based on example input-output pairs

Example: (Linear) regression (see STT5100)

$$(y_i, \boldsymbol{x}_i),\ \begin{cases} y_i = \text{ouput} \in \mathbb{R} \\ \boldsymbol{x}_i = \text{input} \in \mathbb{R}^p \end{cases}$$

$$\textbf{input} \xrightarrow{\ \boldsymbol{x}\ } \textbf{model} \xrightarrow{\ \widehat{y} = \boldsymbol{x}^\top \widehat{\beta}\ } \textbf{ouput}$$

The prediction $\widehat{y}_i$ can be compared with the actual value y_i through some loss function ℓ, e.g. $\ell(y_i, \widehat{y}_i) = [y_i - \widehat{y}_i]^2$

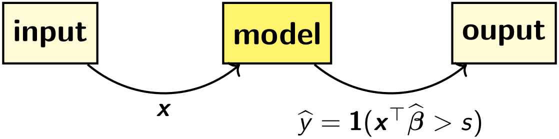
Supervised vs. Non-Supervised

Supervised Learning

In supervised learning, the algorithms learns a function that maps an input to an output based on example input-output pairs

Example: Logistic regression (see STT5100), or classification

$$(y_i, \boldsymbol{x}_i), \begin{cases} y_i = \text{ouput} \in \{0,1\} \\ \boldsymbol{x}_i = \text{input} \in \mathbb{R}^p \end{cases}$$



The prediction $\widehat{y}_i$ can be compared with the actual value y_i through some loss function ℓ, e.g. $\ell(y_i, \widehat{y}_i) = \mathbf{1}(\widehat{y}_i \neq y_i)$

Supervised vs. Non-Supervised

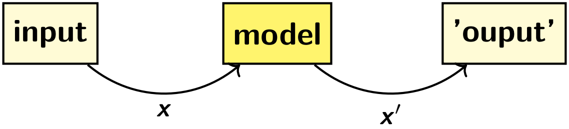
Supervised Learning

In unsupervised learning, the algorithm looks for previously undetected patterns in a data set with no pre-existing labels

(Learn *useful properties of the structure of the data*)

Example: Dimension reduction

$$(\boldsymbol{x}_i),\quad \boldsymbol{x}_i \in \mathbb{R}^p \ \rightarrow\ \boldsymbol{x}'_i \in \mathbb{R}^{p'},\ p' < p$$



possibly with losing to much information
(see principal component analysis)

Supervised vs. Non-Supervised

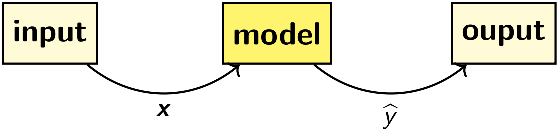
Supervised Learning

In unsupervised learning, the algorithm looks for previously undetected patterns in a data set with no pre-existing labels

(Learn *useful properties of the structure of the data*)

Example: Clustering

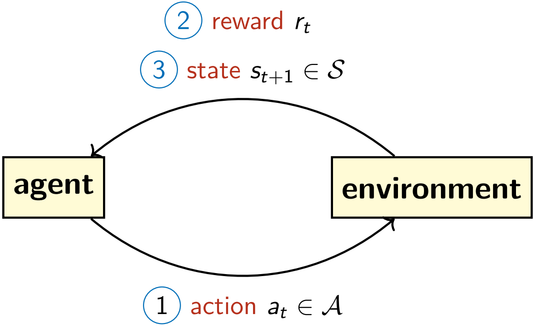
$$(\boldsymbol{x}_i),\quad \boldsymbol{x}_i \in \mathbb{R}^p \ \rightarrow\ y_i' \in \{L_1, L_2, \cdots, L_k\}$$



possibly with strong consistency of the groups (see k-means or hierarchical clustering)

Sequential Learning

Example: Reinforcement Learning

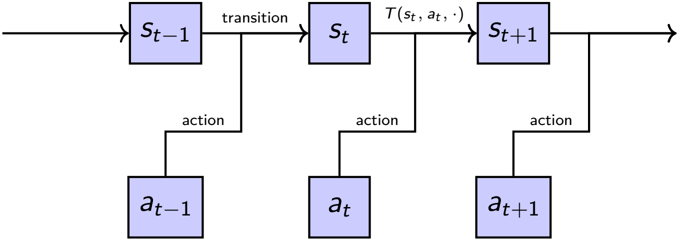


Reinforcement Learning

In reinforcement learning, the algorithm ought to take actions in an environment in order to maximize the notion of cumulative reward

Sequential Learning

- the learner takes an action $a_t \in \mathcal{A}$ (while at state s_t)
- the learner obtains a (short-term) reward $r_t \in \mathcal{R}$
- then the state of the world becomes $s_{t+1} \in \mathcal{S}$



see also multi-armed bandits in decision theory

Loss & Risk for Supervised Learning

Empirical Risk - for sample $\mathcal{D}_n$

The empirical risk associated with $\widehat{m}_n$ is $\widehat{\mathcal{R}}_n(\widehat{m}_n)$

$$\frac{1}{n}\sum_{i=1}^{n}\ell(y_i,\widehat{m}_n(\boldsymbol{x}_i))$$

Average Risk - under $\mathbb{P}$

The average risk associated with $\widehat{m}_n$ is $\mathcal{R}_{\mathbb{P}}(\widehat{m}_n)$

$$\mathbb{E}_{\mathbb{P}}\big[\ell(Y,\widehat{m}_n(\boldsymbol{X}))\big]$$

Cross-Validation for Supervised Learning

Hold-Out Cross Validation

1. Split $\{1, 2, \cdots, n\}$ in T (training) and V (validation)

2 . Estimate $\widehat{m}$ on training sample $(y_i, \boldsymbol{x}_i)$, $i \in T$, $\widehat{m}_T$

$$\underset{m\in\mathcal{M}}{\text{argmin}} \frac{1}{|T|} \sum_{i\in V} \ell(y_i, m(\boldsymbol{x}_i))$$

3. Compute

$$\frac{1}{|V|} \sum_{i'\in V} \ell(y_{i'}, \widehat{m}_T(\boldsymbol{x}_{i'}))$$

Leave-one-Out CV

1. Estimate n models : estimate $\widehat{m}_{-j}$ on sample $(y_i, \boldsymbol{x}_i)$, $i \in \{1, \cdots, n\}\backslash\{j\}$, $\widehat{m}_{-j}$

$$\underset{m\in\mathcal{M}}{\text{argmin}} \frac{1}{n-1} \sum_{i\neq j} \ell(y_i, m(\boldsymbol{x}_i))$$

2. Compute

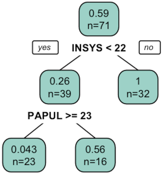
$$\frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \widehat{m}_{-i}(\boldsymbol{x}_i))$$

Trees

For a classification problem $y \in \{0, 1\}$

Trees (CART)

Each interior node corresponds to one of the input variables; there are edges to children for each of the possible values of that input variable. Each leaf represents a value of the target variable given the values of the input variables represented by the path from the root to the leaf.

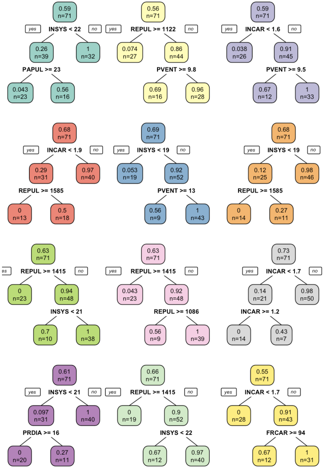


Aggregation & Parallel Learning (bagging)

Bagging (Bootstrap + Aggregation)

1. For $k = 1, \cdots$
   (i) draw a bootstrap sample from $(y_i, \boldsymbol{x}_i)$'s
   (ii) estimate a model $\widehat{m}_k$ on that sample
2. The final model is

$$m^\star(\cdot) = \frac{1}{\kappa}\sum_{k=1}^{\kappa}\widehat{m}_k(\cdot)$$



Aggregation & Sequential Learning (boosting)

Learn slowly - weak learner - and learn from your mistakes

Bosting & Sequential Learning

Starting from $m_0(\cdot) = \overline{y}$, update

$$m_k(\cdot) = m_{k-1}(\cdot) + \underset{h\in\mathcal{H}}{\text{argmin}}\left\{\sum_{i=1}^{n}\ell(\underbrace{y_i - m_{k-1}(\boldsymbol{x}_i)}_{\varepsilon_i}, h(\boldsymbol{x}_i))\right\}$$

Overfit, Generalization, Bias & Variance

Penalization

regularization is the process of adding information in order to solve an ill-posed problem or to prevent overfitting

$$\underset{m\in\mathcal{M}}{\text{argmin}}\left\{\frac{1}{n}\sum_{i=1}^{n}\ell(y_i,m(\boldsymbol{x}_i))+\lambda\text{ penalty}(m)\right\}$$

Usually, it yields biased estimates...