Apprentissage Automatique

Apprentissage Automatique : en quelques mots

Mots clés :

- Machine learning
- Algorithme
- Modèle statistique (ou mathématique au sens large)
- Données d'entrée(s)
- Méthodes d'optimisation
- Prévisions; prédictions; décisions
- Etapes de la modélisation
- Jeu d'apprentissage, jeu de validation et jeu de tests

Le modèle se nourrit des données

Apprentissage automatique : construction d'algorithmes pouvant apprendre afin de faire des prédictions ou prendre des décisions en construisant un modèle (statistique/mathématique) à partir des (seules) données d'entrée. Les données utilisées sont divisées en plusieurs jeux sollicités aux différentes étapes de la création du modèle : les jeux d'apprentissage, de validation et de test.

Apprentissage automatique - Procédure

L'algorithme est tout d'abord **entraîné sur un jeu de données d'apprentissage** (jeu d'exemples) utilisé pour **ajuster les paramètres du modèle statistique** (par exemple, les coefficients b0, b1,... dans une régression linéaire). Si l'on dispose des valeurs de la variable réponse (**étiquettes**, Variable Y, mesures ou observations) on peut faire appel à un **apprentissage supervisé** en utilisant des **méthodes d'optimisation** (telles que les moindres carrés ordinaires ou la descente de gradient). Le modèle produit **un résultat** (les fameux « y chapeau » : $\hat{y}_i = b_0 + b_1 x_i$ obtenus à l'aide de l'équation issue de la régression), qui est **comparé à la cible** (étiquettes, variable Y des mesures ou des observations), **pour chaque individu dans le jeu de données d'apprentissage**.

Les **paramètres du modèle** sont typiquement **ajustés en minimisant les erreurs** issues de la modélisation. L'ajustement du modèle consiste alors en l'estimation des paramètres (b0, b1, b2,... dans une régression linéaire) et de leur "crédibilité" (p_values des coefficients b et part de la variance de Y expliquée par le modèle, le fameux R^2, dans une régression linéaire).

Apprentissage automatique – Evaluation du modèle

Un jeu de données de validation peut être utilisé pour **confirmer que le modèle converge bien** vers une solution ou **identifier un problème**, le cas échéant. Le jeu de données de validation conduirait alors à un **abandon** (± automatique) **du modèle** proposé.

Enfin, **le jeu de données de test** (également appelé jeu de données d'exclusion) est utilisé pour donner une **évaluation de l'ajustement final du modèle** (qui a été effectué sur le jeu de données d'apprentissage). On peut ainsi comparer le coefficient R2 de la modélisation (régression linéaire) calculé sur le jeu test à celui calculé à partir du jeu d'apprentissage. **Le modèle est crédible si les deux valeurs sont très proches l'une de l'autre** (tendant toutes deux idéalement vers 1).

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

Lorsqu'on ne connaît pas les étiquettes y, on parle d'apprentissage non supervisé.

Dans ce cas, on invoque le plus souvent des modèles de **classification**.

Apprentissage automatique – Evaluation du modèle

Data Set *(données d'entrées supposées de bonne qualité)*

division aléatoire

Python, R, …

La modélisation n'est effectuée que sur le jeu d'entrainement

Train Set

(~ 70 %)

Entrainement du modèle
(recherche du meilleur modèle)

Python, R, …

Résultats
+ Paramètres du modèle

Test Set

(~ 30 %)

Le jeu test sert à évaluer Le modèle d'apprentissage

Application du modèle

Evaluation
du modèle

Python, R, …

New data Set → ⇨ Si l'évaluation est bonne, le modèle peut être utilisé sur tout nouveau jeu de données → **Prévisions**

Scission en jeu d'apprentissage et jeu de test : pratique dans Python

Scinder le jeu est un jeu d'enfants :

- Bibliothèque : sklearn.model_selection
- Classe : train_test_split
- Gestion aléatoire des index : sample(frac=0.8)
- loc[~biotope.index.isin(train.index)] : ***Test*** = *Tout ce qui n'est* ***pas*** *Train*
- Les dataframe sont ensuite utilisables indépendamment

```
# Les bibliothèques utilisées :
from sklearn.model_selection import train_test_split

# Génération (aléatoire) du jeu d'entrainement (training set).
# En fait tout ceci repose tout simplement sur la sélection d'index au hasard

train = biotope.sample(frac=0.8,random_state=1)

# On Sélectionne ensuite tout ce qui n'est pas dans le training set
# et on le met dans le jeu de test. Le tilde ~ représente ici le "NON" logique.

test = biotope.loc[~biotope.index.isin(train.index)]
```

Les étapes de l’apprentissage [Régression Linéaire] : conclusion

Machine Learning = Informatique U Statistique

Le modèle se nourrit des données