Apprentissage Automatique Avancé

Lesson 3 : Ensemble learning

Sébastien Verel

Laboratoire d'Informatique, Signal et Image de la Côte d'opale (LISIC)
Université du Littoral Côte d'Opale, Calais, France
http://www-lisic.univ-littoral.fr/~verel/

2024 - 205

Plan

1. Introduction
2. Random forest
3. Gradient boosting

Principle

Unity is strength

Instead of using a single (poor) learner

Use a set of learners, and combine their prediction

Principle

Unity is strength

Instead of using a single (poor) learner

Use a set of learners, and combine their prediction

Classification :

$A : 1$, $B : 0$, $C : 1$, $D : 0$, $E : 0$

Answer of the set/team of classifiers ?

Principle

Unity is strength

Instead of using a single (poor) learner

Use a set of learners, and combine their prediction

Classification :

$A : 1$, $B : 0$, $C : 1$, $D : 0$, $E : 0$

Answer of the set/team of classifiers ?

Classification with confidence probability :

$A : 1(0.8)$, $B : 0(0.7)$, $C : 1(0.9)$, $D : 0(0.5)$, $E : 0(0.6)$

Answer of the set/team of classifiers ?

Principle

Unity is strength

Instead of using a single (poor) learner

Use a set of learners, and combine their prediction

Classification :

$A : 1$, $B : 0$, $C : 1$, $D : 0$, $E : 0$

Answer of the set/team of classifiers ?

Classification with confidence probability :

$A : 1(0.8)$, $B : 0(0.7)$, $C : 1(0.9)$, $D : 0(0.5)$, $E : 0(0.6)$

Answer of the set/team of classifiers ?

Regression :

$A : 2.3$, $B : 9.1$, $C : 4.3$, $D : 4.1$, $E : 3.8$

Answer of the set/team of regressors ?

Main idea

What is a good team ?

Main idea

What is a good team ?

Diverse (independent ?) answers/learners/models

Imagine a team of clones...

Main idea

What is a good team ?

Diverse (independent ?) answers/learners/models

Imagine a team of clones...

Drawback

More models/learners to train : can be time consuming

But, sometime, parallel computing can be possible...

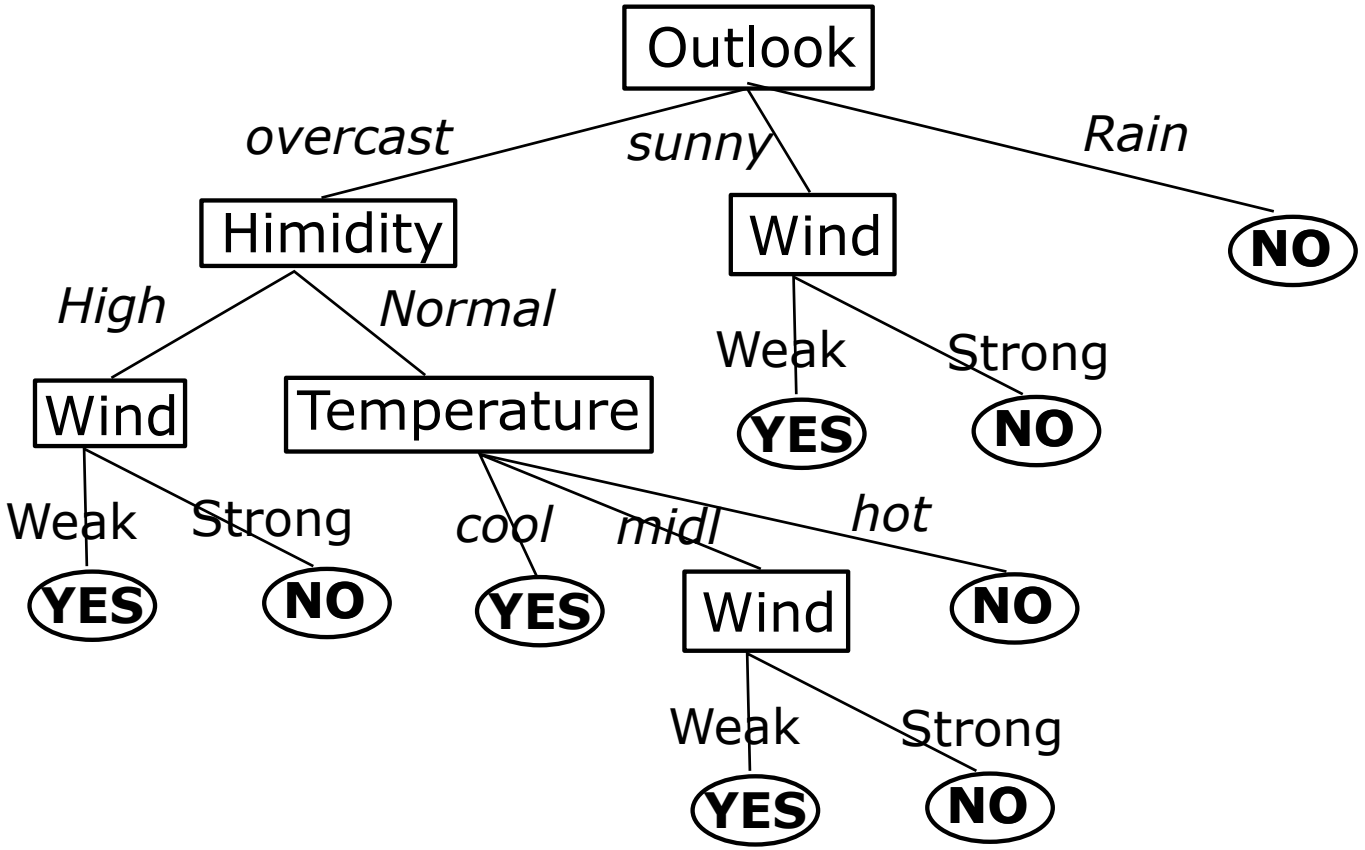
Design approaches

How to design a good team ?

- Parallel approach :
  Design independent learners in parallel
- Sequencial approach :
  use the knowledge (error) of previous learner to train the next learner

Reminder : Decision tree

Classification method



Remarque : un arbre code en fait un ensemble de règles (conjonctions, disjonctions)

Si Outlook = "overcast" et Humidity =... alors playball = Yes

Algorithme d'apprentissage

Apprentissage par arbre de décision

Construction un arbre :

- Noeuds internes : sélectionner d'un attribut comme étiquette, les arcs sont étiquetés par les valeurs de l'attribut
- Feuilles : couper l'arbre avec une valeur de l'attribut cible

On veut en général :

- Un taux d'erreur faible
- Une bonne généralisation
- Un arbre de petite taille compréhensible pour un non expert
- etc.

Nombreux algos : ID3, C4.5, CART, CHAID, algo. évo., etc.

Une classe d'algorithmes d'apprentissage

Algorithme top-down

Procédure : `construireArbre`(X)
if tous les exemple de X appartiennent à la même classe **then**
  Créer une feuille portant le nom de cette classe
else
  Choisir un attribut pour créer un noeud
  Le test associé à ce noeud sépare X en X_d et X_g
  `construireArbre`(X_g)
  `construireArbre`(X_d)
end if

Une classe d'algorithmes d'apprentissage

Algorithmes top-down greedy

Pour chaque noeud interne,

- le "meilleur" attribut est sélectionné selon l'ensemble d'apprentissage
- l'ensemble d'apprentissage est partitionné selon les valeurs possibles de l'attribut du noeud

Le processus est répété en chaque noeud et s'arrête lorsque :

- tous les exemples ont la même valeur d'attribut cible
- un nouveau partitionnement n'augmente pas la qualité de la prédiction

- Top-down : construction à partir de la racine
- Greedy : meilleur choix local, pas de remise en cause
  Les optima locaux guettent ! Optimalité locale *vs.* globale

Critique

Avantages

- Simple à comprendre et à interpréter
- Le modèle est "white-box" (rés. neurones est black-box)
- Peu de préparation des données : pas de normalisation, etc.
- Données numériques et catégorielles possibles
- Robuste aux données aberrantes (outliers)

Inconvénients

- Apprendre un arbre de décision optimal : NP-complet
- Heuristique d'apprentissage greedy : arbre sous optimal
- Création d'arbres trop complexes, sur-spécialisé
- Biais vers certaines formes : attribut avec plus de valeurs, petit arbre, etc.
- Détection difficile des interactions entre attributs
- Certains problèmes sont difficiles à apprendre sous forme d'arbre (xor, parité, multiplexer)

ID3 (Iterative Dichotomiser 3)

Ross Quinlan, 1986

Algorithme top-down greedy
basé sur le gain d'information (information gain)

Principe

1. Calculer l'entropie de tous les attributs en utilisant l'ensemble d'apprentissage S
2. Partitionner l'ensemble S en utilisant l'attribut pour lequel l'entropie est minimum (gain d'information maximum)
3. Construire le noeud de l'arbre avec cet attribut
4. Recommencer récursivement sur chaque sous arbre avec chaque sous-ensemble

Mesure d'entropie

Entropie H

Mesure de la quantité d'incertitude dans un ensemble (dispersion)

$$H(S) = -\sum_{x \in X} p(x) \log_2 p(x)$$

- S : ensemble des données
- D_S : ensemble des classes de S
- $p(s)$: proportion de la classe $s \in D_S$ dans S

Lorsque $H(S) = 0$, S est parfaitement classé.

Gain d'information

Information mutuelle (entropie croisée)

Mesure de l'information conjointe de 2 variables aléatoires (information d'une variable par rapport à l'autre)

$$I(S,T) = -\sum_{s \in D_S, t \in D_T} p(s,t) \log_2 \frac{p(s,t)}{p(s)p(t)}$$

Information gain (- information mutuelle)

Mesure de la différence d'entropie entre avant et après le partitionnement selon un attribut

$$IG(S,T) = -I(S,T) = H(S) - \sum_t p(S_t) H(S_t)$$

- $T = \{S_1, \ldots\}$ partitionnement de $S = \cup_t S_t$
- $p(S_t) = \sharp S_t / \sharp S$
- $H(S)$, $H(S_t)$: entropies de S et de S_t

Pseudo code

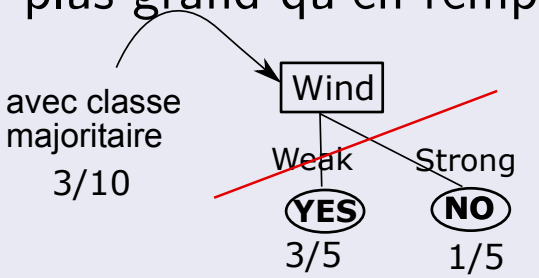
ID3(exemples, cible, attributs) :
- **si** tous les exemples sont positifs (resp. négatifs) **alors**
  - **retourner** une feuille avec l'étiquette positif (resp. négatif)
- **si** attributs est vide **alors**
  - **retourner** une feuille avec l'étiquette la plus fréquente
- **sinon**
  - $A \leftarrow$ attribut de plus grand gain d'information
  - **construire** un noeud avec l'étiquette A
  - **pour** chaque valeurs v_i de A
    - **ajouter** la branche v_i au noeud
    - **si** exemples($A = v_i$) est vide **alors**
      - **ajouter** à la branche la feuille avec l'étiquette la plus fréquente
    - **sinon**
      - **ajouter** à la branche le sous-arbre ID3(exemples($A = v_i$), cible, attributs $-A$)

C4.5

Ross Quinlan, 1993

Amélioration de ID3

- Utilisation du ratio de gain d'information au lieu de IG :
  $IG(S, T)$ biaisé vers attributs ayant un grand nombre de valeurs
  $ratioIG(S, T) = IG(S, T)/H(T)$
- Possibilité de valeur "null" :
  Exemple ignoré lors dans le calcul du noeud
- Prise en compte des attributs à valeur continue :
  Discrétisation par $P(A < a_i)$
  pour toutes les valeurs possibles de A, calcul de IG
- Elagage (pruning) pour réduire la taille de l'arbre :
  Technique bottom-up : branches finales élaguées
  lorsque taux d'erreur plus grand qu'en remplaçant par une feuille



Random forest overview

- Ensemble method, and Parallel method
- One of the efficient ML algorithm (classification, regression)
- "Quick" to train, available in many language (R, sklean)
- Less easier to understand than decision tree, mais better precision (often)
- Variable importance is available

Breiman, Leo (2001). "Random Forests". Machine Learning 45 (1) : 5-32.

Baseline idea

- Ensemble method :
  use a set of simple (weak) predictors
- Weak learner :
  decision tree (trained with CART)
- Prediction :
  - Majority voting for classification
  - Mean of prediction for regression
- Key of the training phase :
  design a diversity of trees

Principle

Create a set of decision trees, with the techniques :

$$\text{random forest} = \text{bagging} + \text{features sampling}$$

Each tree has partial view of the problem to create diverse trees.

Prediction for classification problem (majority voting) :

$$h(x) = \text{argmax}_c \sum_{b=1}^{B} \delta^c_{t_b(x)}$$

Prediction for regression problem (average) :

$$h(x) = \frac{1}{B} \sum_{b=1}^{B} t_b(x)$$

Bagging (Bootstrap aggregating)

Goal

Create diverse Training sub-sets

Algorithm

Given a training sample (X, Y) of size n

for $b = 1, \ldots, B$ **do**

Sample with replacement n training examples from (X, Y) : (X_b, Y_b)

Train a weak learner (decision tree) t_b from (X_b, Y_b)

end for

Agregate for the result the b weak learners t_b

Feature sampling

Goal

Create diverse set of sub models

Algorithm

Random projection of the set of features :

Given a set of features $\{X_1, \ldots, X_p\}$ of size p
for $b = 1, \ldots, B$ **do**
  Sample without replacement k features from X :
  $\{X_{j_1}, \ldots, X_{j_k}\}$
  Train a weak learner (decision tree) t_b on the k features
end for
Aggregate for the result the b weak learners t_b

Usually, $k = \sqrt{p}$,
but the optimal value can be computed using a model selection technique (cross-validation, etc.)

Random forest algorithm

Algorithm

Given a set of features $\{X_1, \ldots, X_p\}$ of size p
Given a training sample (X, Y) of size n
for $b = 1, \ldots, B$ **do**
  Sample without replacement k features from X : $\{X_{j_1}, \ldots, X_{j_k}\}$
  Sample with replacement n training examples from $(X_{j_1}, \ldots, X_{j_k}, Y) : (X_b, Y_b)$
  Train a weak learner (decision tree) t_b from (X_b, Y_b)
end for
Aggregate for the result the b weak learners t_b

Question

What are the parameters of Random Forest ?

Variance reduction of the forest

$$\sigma^2_{forest} = \rho\sigma^2_{tree} + \frac{1-\rho}{B}\sigma^2_{tree}$$

where :

- σ^2_{tree} : variance of decision trees training
- B : number of trees
- ρ : correlation between pair of trees

Feature selection, and tree bagging
can reduce the correlation coefficient ρ

Variantes

- Extremely randomized trees :
  When design one decision tree, the split can be randomly selected (not using information criterion)

Geurts, Pierre and Ernst, Damien and Wehenkel, Louis, Extremely randomized trees, Machine Learning, vol. 6, 1, pp. 3-42, 2006.

Out-of-bag error

For each bag, some sample is not used for training :

- For bag b : $X = X_b \cup X_{\text{out-of-bag } b}$
- Compute the error of t_b on $X_{\text{out-of-bag } b}$
- Average errors of data on all bags

"For free", an almost cross-validation error can be computed

Variable importance

Gini importance (or mean decrease impurity)

When constructing a decision tree, the information gain is computed for internal node to split/select the feature :

- Measure how the feature X_j decrease the impurity of the split
- Average overall trees the decreasing for variable X_J
- Relative values between features can be compared

Avantages : quick to compute
Drawback : features with high cardinality are preferred

Variable importance

Permutation Based Feature Importance

Importance of feature X_j :

- Values of variables X_j are permuted among the training data. The out-of-bag error is computed.
- Importance = average of the difference in out-of-bag error before and after the permutation over all trees.
- The score is normalized by the standard deviation

Avantages : not bias
Drawback : can be expensive to compute

In practice with Scikit-learn

`https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html`

Boosting

Bibliographie

- adaBoost pour adaptive boosting (prix Gödel 2003) :

  Yoav Freund et Robert Schapire, A desicion-theoretic generalization of on-line learning and an application to boosting, Journal of Computer and System Sciences, vol. 55, Num. 1,1997, p. 119-139

- Gradient boosting :

  J.H. Friedman, Greedy function approximation a gradient boosting machine, Ann. Statist., Vol., Num. 5 (2001), 1189-1232.

Principle of boosting

Principe

- Iterative sequential construction of weak learners (decision tree for example)
- The error of the previous iteration is taking into account to design the next one :
  Weak leaners are "boosted"

Boosting *vs.* random forest

- Random forest : majority voting

$$h(x) = \text{argmax}_c \sum_{b=1}^{B} \delta^c_{t_b(x)}$$

- Boosting : Weighted sum of weak learners

$$h(x) = \text{argmax}_c \sum_{b=1}^{B} \alpha_b \delta^c_{t_b(x)}$$

Origin : Adaboost

$\mathcal{H}$: set of possible weak learners
Initialize the distribution of examples $\forall i = 1, \ldots, n,\ D_1(i) = \frac{1}{n}$

for $t = 1, \ldots, T$ **do**
Compute weighted error : $\epsilon_t(h) = \sum_{i=1}^{n} D_t(i).[y_i \neq h(x_i)]$
Train new learner on weigthed error : $h_t = \underset{h \in \mathcal{H}}{\operatorname{argmin}}\ \epsilon_t(h)$
Weight of the learner : $\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
Update weigths : $\forall i = 1, \ldots, m,\ D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$
with $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$
end for

Classifier $H(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x))$

Gradient boosting : Principle

gradient boosting = **gradient descent** + **boosting**

- Generalization of error function
  and using a **gradient descent**
- **boosting** : regression on the residus of previous iteration

Follow the slides :
`http://orbi.ulg.ac.be/bitstream/2268/163521/1/slides.pdf`
And, also : `https://towardsdatascience.com/all-you-need-to-know-about-gradient-boosting-algorithm-par`