

Apprentissage Automatique
Arbres de décision & méthodes ensemblistes

S. Herbin

stephane.herbin@onera.fr

Introduction

Rappel du dernier cours

- ▶ Principes généraux d'apprentissage : données apprentissage/validation/test, optimisation, évaluation
- ▶ Deux algorithmes élémentaires de *classification supervisée* : plus proche voisin, classifieur Bayésien

Objectifs de ce cours

- ▶ Un nouveau type de classifieur : l'arbre de décision
- ▶ Un principe de conception : les approches ensemblistes
Intuition : « un groupe prend plus souvent de meilleures décisions qu'un individu »

Arbres de décision et méthodes ensemblistes : plan

Arbres de décision

Méthodes ensemblistes

Bagging

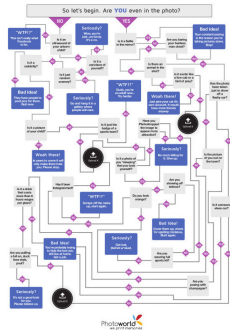
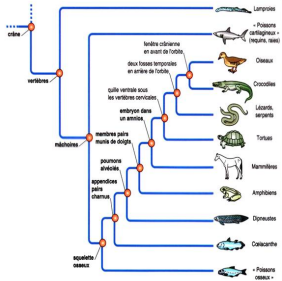
Random Forests

Conclusion

Arbres de décision

Arbres de décision [3, 9]

Un modèle intuitif : décomposer une décision globale en une séquence de décisions locales (questions)



⇒ Ou bien encore comme dans le *jeu des 20 questions*

Arbres de décision

Exemple : attendre une table ou partir ?

- ▶ Un problème de décision binaire.
- ▶ Questions possibles : Autre restaurant à proximité ? Bar dans le restaurant ? Est-on vendredi ? Avons-nous faim ? Le restaurant est-il plein ? Est-il cher ? Quel est le type de restaurant ? Temps d'attente estimé ? Avons-nous une réservation ? Pleut-il ?

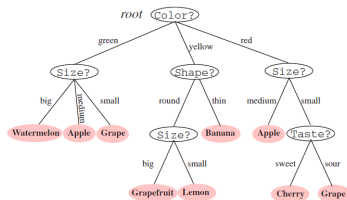
12 exemples d'apprentissage.

Example	Attributes										Target
	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	Wait
X ₁	T	F	F	T	Some	\$\$\$	F	T	French	0-10	T
X ₂	T	F	F	T	Full	\$	F	F	Thai	30-60	F
X ₃	F	T	F	F	Some	\$	F	F	Burger	0-10	T
X ₄	T	F	T	T	Full	\$	F	F	Thai	10-30	T
X ₅	T	F	T	F	Full	\$\$\$	F	T	French	>60	F
X ₆	F	T	F	T	Some	\$\$	T	T	Italian	0-10	T
X ₇	F	T	F	F	None	\$	T	F	Burger	0-10	F
X ₈	F	F	F	T	Some	\$\$	T	T	Thai	0-10	T
X ₉	F	T	T	F	Full	\$	T	F	Burger	>60	F
X ₁₀	T	T	T	T	Full	\$\$\$	F	T	Italian	10-30	F
X ₁₁	F	F	F	F	None	\$	F	F	Thai	0-10	F
X ₁₂	T	T	T	T	Full	\$	F	F	Burger	30-60	T

Arbres de décision

Principe

- ▶ Classification en posant une séquence de questions fermées (= nombre fini de réponses possibles)
- ▶ Questions organisées sous forme d'arbre : la question suivante dépend de la réponse à la question précédente
- ▶ La réponse à la dernière question définit la prédiction finale



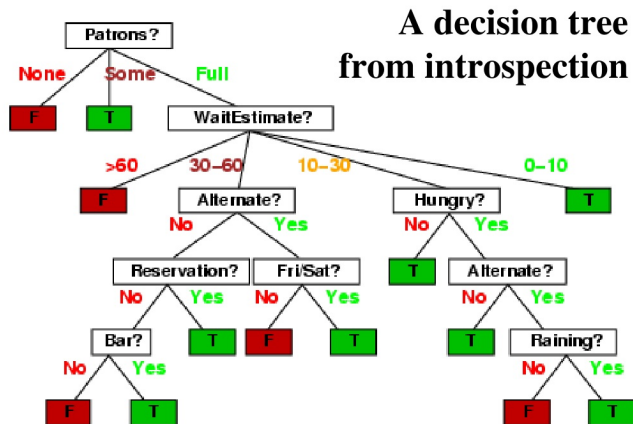


Figure 1 – Un arbre « expert » pour résoudre le problème du restaurant.

Arbres de décision

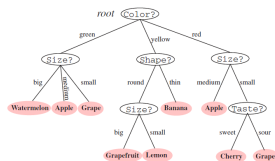
Questions type

- ▶ Sur la valeur d'un attribut caractéristique
- ▶ Sur la véracité d'une clause logique
- ▶ Sur l'appartenance à un intervalle ou un sous-ensemble
- ▶ Sur des attributs discrets ou numériques
- ▶ ...

Grande flexibilité possible mais les questions doivent être simples (à calculer)

Arbres de décision : structure

- ▶ **Données** codée comme ensemble d'attributs (ex : attributs d'un fruit = couleur, taille forme, goût...)
- ▶ **Noeud de décision** associé à un **test** ou **question** sur un des attributs
- ▶ **Branches** qui représentent les valeurs possibles de l'attribut testé ou des réponses aux questions
- ▶ **Noeud terminal** ou feuille, liée à la classe (prédiction)



Arbres de décision : principe de prédiction

- ▶ Les questions découpent (partitionnent) l'espace des attributs à chaque étape
- ▶ Le noeud terminal code un élément de la partition
- ▶ Toutes les données codées par le noeud terminal ont la même prédiction

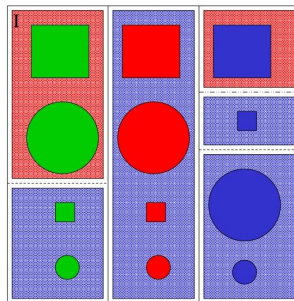
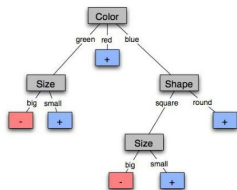


Figure 2 – Partition sur des données symboliques. Les questions portent sur la valeur d'un attribut discret.

Arbres de décision

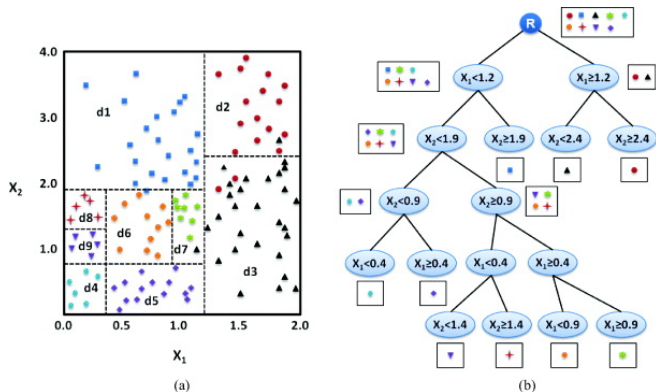
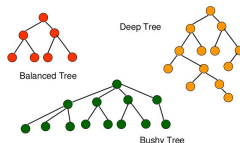


Figure 3 – Partition sur des données numériques. Les questions sont des tests comparant la valeur d'une dimension à un seuil.

Arbres de décision

Quelles questions se poser pour construire un arbre ?



Questions globales :

- ▶ Quelle structure choisir ? (profond, équilibré,...) ?
- ▶ Combien de découpages par noeud ? (binaire, plus)
- ▶ Quand s'arrêter de découper ?

Questions locales :

- ▶ Quel attribut choisir, et quel test lui appliquer ?
- ▶ Si l'arbre est trop grand, comment l'élaguer ?
- ▶ Si une feuille n'est pas pure, quelle classe attribuer ?

Arbres de décision : position du problème d'apprentissage

Soit $X = \{(x_j, y_j)\}_{j \leq N}$ des données décrites chacune par un ensemble d'attributs $x_j = \{A_i^j\}_{1 \leq i \leq M}$, avec A_i^j à valeurs numériques ou symboliques, et y_j la prédiction (vérité terrain).

Recherche du plus petit arbre de décision compatible avec X :

- ▶ Principe du rasoir d'Occam : trouver l'hypothèse la plus simple possible compatible avec les données
- ▶ Principe « Minimum Description Length » : trouver l'hypothèse qui produit le plus petit nombre d'opérations

Mais... recherche optimale impossible (problème NP-complet) [10]

⇒ Heuristique assurant un arbre cohérent sur les données d'apprentissage.

Arbres de décision : algorithme élémentaire

Principe général

Construction incrémentale d'un arbre.

Trois étapes

1. Décider si un noeud est **terminal**
2. Si un noeud n'est pas terminal, choisir un attribut, un test et des **branches** possibles
3. Si un noeud est terminal, lui associer une **prédiction** (une classe, une valeur, etc.)

Remarque : il existe des formulations plus globales [8].

Arbres de décision : algorithme élémentaire

Choisir un attribut et un test

⇒ Algorithmes récursifs (par exemple ID3, C4.5, CART...)

Fonction Construire-arbre(X)

SI tous les points de X sont de même classe,
créer une feuille associée à cette classe

SINON

- ▶ choisir la meilleure paire (A_i, test) pour créer un noeud
- ▶ ce test sépare X en 2 parties X_g et X_d
- ▶ Construire-arbre(X_g)
- ▶ Construire-arbre(X_d)

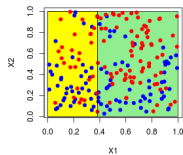
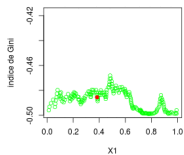
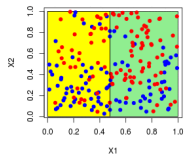
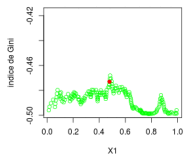
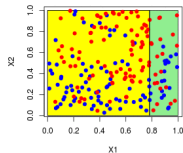
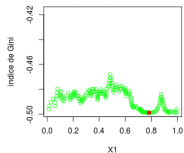
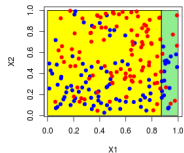
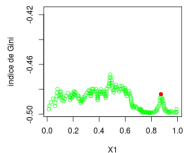
Arbres de décision : algorithmes

Critères de sélection d'un noeud

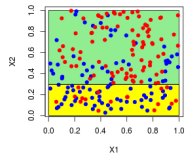
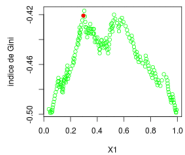
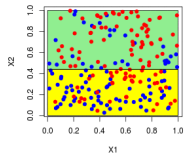
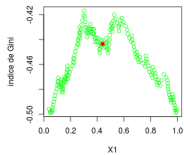
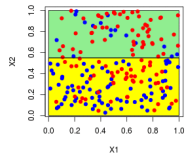
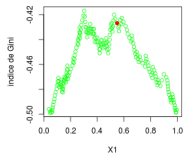
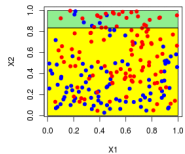
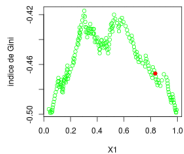
⇒ Mesure de l'hétérogénéité (ou de la pureté).

- ▶ Entropie : $H = - \sum p(c_k) \log_2(p(c_k))$ avec $p(c_k) = N_k/N$
probabilité de la classe c_k dans l'ensemble courant
→ ID3, C4.5, mesure l'information
- ▶ Indice de Gini : $I = \sum p(c_k)(1 - p(c_k)) = 1 - \sum p(c_k)^2$
→ CART, mesure les inégalités
- ▶ Indice d'erreur : $I = 1 - \max(p(c_k))$

Arbres de décision : algorithmes



Arbres de décision : algorithmes



Arbres de décision : algorithmes

Choix attribut et test

⇒ Gain d'homogénéité apporté par un test T pour séparer un noeud V en noeuds V_j .

- ▶ À chaque noeud, choix de T maximisant
$$Gain(V, T) = I(V) - \sum_j p(V_j)I(V_j)$$
- ▶ En pratique, approche empirique pour tout A_i tri des valeurs par ordre croissant et tests tirés selon une approche dichotomique (médiane, etc...)

Arbres de décision : espace continu

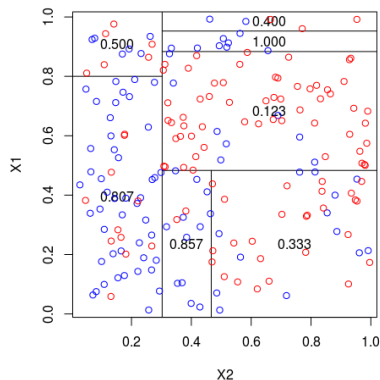
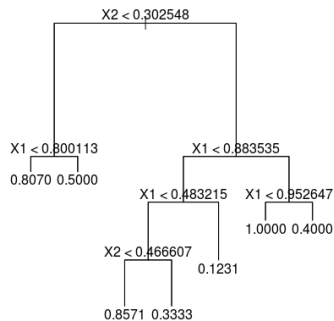


Figure 4 – Arbre et partition obtenu en utilisant un critère de type Gini.

ID3-induced decision tree

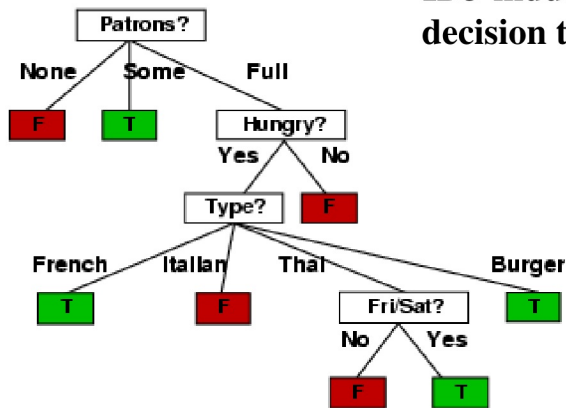
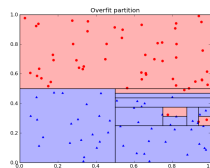
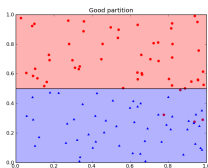
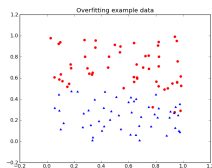


Figure 5 – Un arbre « appris » pour résoudre le problème du restaurant.

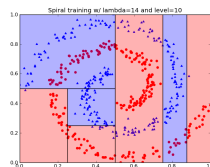
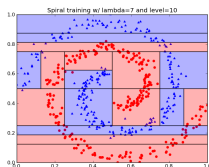
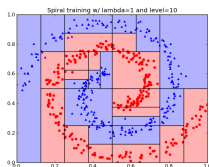
Arbres de décision : comportement



Sur-apprentissage

- ▶ Un arbre trop précis risque de mal généraliser (cf. k-NN)
- ▶ Les arbres peuvent être mal équilibrés
- ▶ On peut utiliser des techniques d'élagage (« pruning ») pour améliorer a posteriori la qualité des arbres

Arbres de décision : comportement



La complexité peut être contrôlée

- ▶ en limitant la profondeur
- ▶ en minorant le gain en homogénéité
- ▶ en ajoutant une pénalisation de complexité dans le coût
- ▶ en garantissant une bonne estimation des coûts (par ex. un nombre minimal d'échantillons par noeud)

Points clés des arbres de décision

- + Interprétabilité
- + Apprentissage et classification rapides et efficaces, y compris en grande dimension.
- Tendance au surapprentissage (mais moyen de contrôle de la complexité)
- Sensibilité au bruit et aux points aberrants, instabilité

Utilisations

- + Classification ou régression...
- + Capable de traiter des données numériques, mais aussi symboliques

Méthodes ensemblistes

Méthodes ensemblistes

Définition

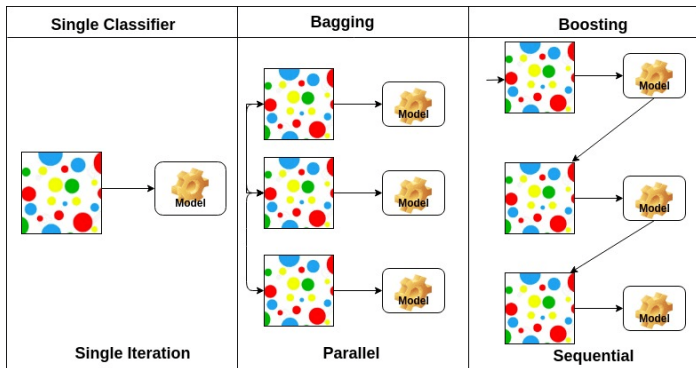
- ▶ Méthodes agréant des *ensembles* de classifieurs ;
- ▶ Produire une variété de classifieurs : en échantillonnant différemment les données, en modifiant les structures de classifieurs ;
- ▶ Classe finale = fusion des prédictions.

Principe

- ▶ *L'union fait la force* : tirer parti de plusieurs classifieurs peu performants (« faibles ») pour construire un classifieur performant (« fort »)
 - ⇒ Réduit la *variance* d'apprentissage et moyenne les erreurs

Méthodes ensemblistes

Deux grandes approches : bagging et boosting



Bagging [1]

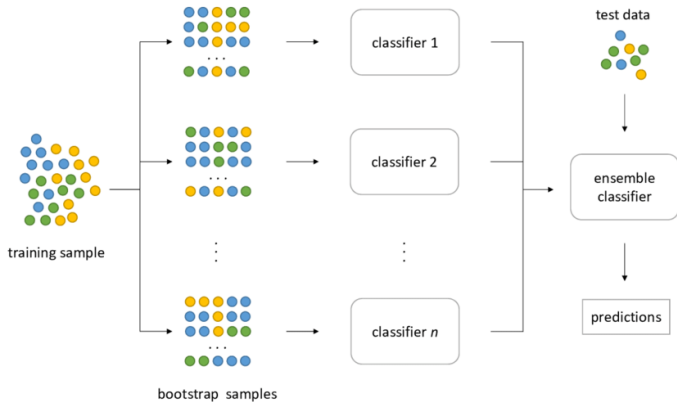
Génération de jeux de données multiples

- ▶ Construction de $\tilde{X}_1, \dots, \tilde{X}_K$ par tirage avec remise sur X .
- ▶ \tilde{X}_k similaires, mais pas trop (proba d'un exemple de ne pas être sélectionné $p = (1 - 1/N)^N$. Quand $N \rightarrow \infty$, $p \rightarrow 0.3679$.)
- ▶ Entraîner K fois le même algorithme f_k (arbre, réseau de neurones, SVM..) sur chaque \tilde{X}_k et agréger par vote majoritaire ou moyenne $f(x) = \frac{1}{K} \sum f_k(x)$

Conséquence

- ▶ Chaque classifieur commet des erreurs différentes, liées à \tilde{X}_k
→ l'agrégat a une plus faible variance d'apprentissage
- ▶ Méthode pour *régulariser* le processus de prédiction.

Bagging



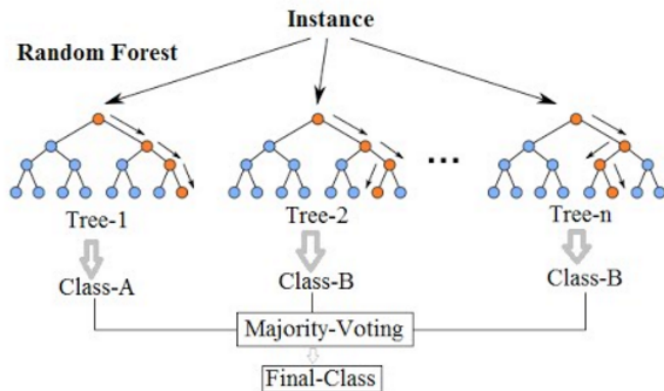
Forêts aléatoires ou *Random forests*

⇒ Combiner hasard et bagging pour construire un ensemble d'arbres de décision encore plus varié (=forêt)

- ▶ La partie calculatoire des arbres de décision est la construction incrémentale de leur structure (meilleure paire attribut & test)
- ▶ Structure = paramètre de contrôle des arbres (profondeur max, critère de pureté des noeuds, nombre d'échantillons par noeud...) + aléatoire sur attributs/données/tests

Random Forests

Random Forest Simplified



Random Forests

Forêts aléatoires ou *Random forests*

Algorithme :

POUR $k = 1 \dots K$:

- ▶ Bagging : tirage de \tilde{X}_k de même taille que X
- ▶ Tirage (avec remise) de q attributs A_i parmi les M possibles
- ▶ Construction de l'arbre G_k avec des seuils aléatoires
- ▶ Construction de f_k la fonction de décision de G_k dont les feuilles sont remplies avec \tilde{X}_k

Agrégation :

- ▶ $f(x) = \frac{1}{K} \sum f_k(x)$ (régression)
- ▶ $f(x) = \text{Vote majoritaire}(f_1(x), \dots, f_K(x))$

Biais et variance

Exemple de la régression

$$y = f(x) + \epsilon$$

Il y a deux sources d'aléatoire :

- ▶ Le bruit : ϵ (un même x peut produire différents y)
- ▶ L'échantillonnage des données d'apprentissage : D

On définit pour un prédicteur appris $\hat{f}_D(x)$:

Erreur écart quadratique moyen entre prédiction et valeur idéale

Biais erreur de la prédiction moyenne par rapport à la valeur idéale

Variance écart quadratique moyen entre prédiction et prédiction moyenne

Compromis biais variance

L'erreur pour un x donné peut se décomposer en :

$$\begin{aligned} \text{Err}(x) &= E_D[(y - \hat{f}_D(x))^2] \\ &= \underbrace{\epsilon^2}_{\text{bruit}^2} + \underbrace{(E_D[\hat{f}_D(x)] - y)^2}_{\text{biais}^2} + \underbrace{E_D[(E_D[\hat{f}_D(x)] - \hat{f}_D(x))^2]}_{\text{variance}} \end{aligned}$$

L'origine de l'erreur de généralisation est double, mais les deux termes sont difficiles à contrôler individuellement.

Rem : pour la classification, une telle décomposition est plus difficile à obtenir, mais les comportements sont comparables.

Biais et variance

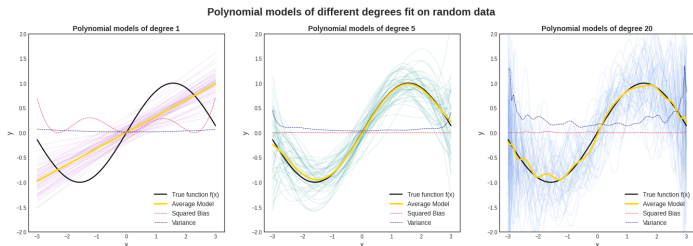


Figure 6 – Simulation d'une régression pour 50 échantillons et polynômes de degrés 1,5,20.

- ▶ Degré 1 : variance faible, mais biais important
- ▶ Degré 5 : variance et biais faibles
- ▶ Degré 20 : variance importante et biais très faible

Intérêt des approches ensemblistes

On introduit une source d'aléatoire supplémentaire : choix des splits, du sous ensemble de variables, etc.

Lorsque les prédicteurs individuels sont sans biais (c'est le cas avec les arbres), la variance du prédicteur ensembliste est :

$$\text{var} \left(\hat{f}_D(x) \right) = \rho \sigma^2 + \frac{1 - \rho}{K} \sigma^2$$

σ variance d'un prédicteur individuel et ρ corrélation entre deux prédicteurs.

On voit que l'on a intérêt à construire des prédicteurs individuels indépendants ($\rho \approx 0$), et en grand nombre (K grand).

Points clés des forêts aléatoires

- + Bonnes performances
- + Arbres plus décorrélés que par simple bagging
- + Grandes dimensions
- + Robustesse
- Temps d'entraînement (mais aisément parallélisable).

Utilisation

- ▶ Choix d'une faible profondeur (2 à 5), autres hyper-paramètres à estimer par validation croisée
- ▶ Classification et régression
- ▶ Données numériques et symboliques

Boosting [5]

Principe

- ▶ $X = \{(x_i, y_i)\}_{i=1}^N$ un ensemble de données où $y_i \in \{-1, 1\}$
- ▶ H un ensemble ou une famille de classifieurs $f \mapsto -1, 1$, pas forcément performants \rightarrow appelés *weak learners*

Objectif du boosting :

- ▶ Construire un classifieur performant $F(x) = \sum_{k=1}^K \alpha_k f_k(x)$
 \rightarrow appelé *strong learner*
 - ▶ Moyenne pondérée des *weak learners*
- ▣ Comment trouver les poids ?

AdaBoost

- ▶ Adaboost = « Adaptive boosting algorithm », algorithme minimisant l'erreur globale de F de manière itérative
- ▶ Principe : à chaque itération k , modifier F^k de manière à *donner plus de poids aux données difficiles* (mal-classées) qui permettent de corriger les erreurs commises par F^{k-1}

AdaBoost : algorithme

Initialiser les poids liés aux données :

$$d^0 \leftarrow \left(\frac{1}{K}, \frac{1}{K}, \dots, \frac{1}{K} \right)$$

POUR $t = 1 \dots K$:

- ▶ Entraîner f_k sur les données X pondérées par d^{k-1}
($f_k = \arg \min_f \sum_i d_i^{k-1} [y_i \neq f(x_i)]$)
- ▶ Prédire $\hat{y} = y^j \leftarrow f_k(x_i), \forall i$
- ▶ Calculer l'erreur pondérée $\epsilon^k \leftarrow \sum_i d_i^{k-1} [y_i \neq \hat{y}_i]$
- ▶ Calculer les paramètres adaptatifs $\alpha^k \leftarrow \frac{1}{2} \log \left(\frac{1-\epsilon^k}{\epsilon^k} \right)$
- ▶ Re-pondérer les données $d^k = d_i^k \leftarrow d_i^{k-1} \exp(-\alpha^k y_i \hat{y}_i)$

Classifieur (pondéré) final : $F(x) = \text{sgn} \left(\sum_{k=1}^K \alpha_k f_k(x) \right)$

Boosting

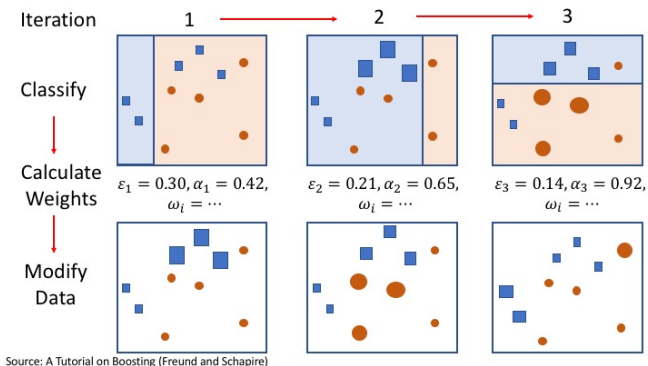
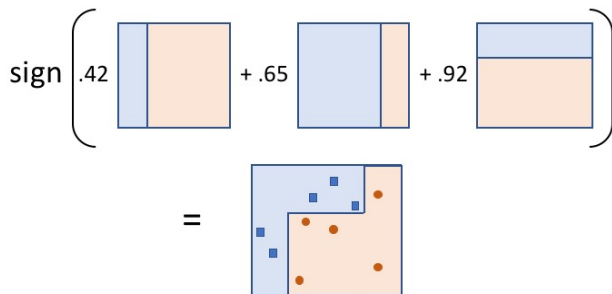


Figure 7 – Apprentissage séquentiel des classifieurs et des pondérations.

Boosting



Source: A Tutorial on Boosting (Freund and Schapire)

Figure 8 – Classifieur final.

Gradient Boosting [6, 7]

Gradient Boosting

Variante : version additive pas-à-pas

- ▶ $X = \{(x_i, y_i)\}_{i=1}^N$ un ensemble de données où $y_i \in \{-1, 1\}$
- ▶ H un ensemble de classifieurs $f \mapsto -1, 1$, pas forcément performants \rightarrow appelés *weak learners*

Objectif du gradient boosting :

- ▶ Construire itérativement un classifieur performant
$$F_T(x) = \sum_{t=1}^T \alpha_t f_t(x) = F_{T-1}(x) + \alpha_T f_T(x)$$
 où f_t est l'un des weak learners h .
- ▶ Il s'agit à chaque étape de minimiser le risque empirique :
$$\mathcal{L}(F_T) = \sum_{n=1}^N l(y_n, F_T(x_n))$$
 où l est un coût (*loss*)

Gradient Boosting

Coûts

- ▶ Adaboost \rightarrow gradient boost avec fonction de coût
 $l(y, f(x)) = \exp(-y \cdot f(x))$
- ▶ Adaboost peut être vu comme la construction itérative d'un classifieur optimal par minimisation du risque empirique à chaque pas.
- ▶ Cadre plus général : d'autres pénalités sont possibles :
 - ▶ LogitBoost : $l(y, f(x)) = \log_2(1 + \exp[-2y \cdot f(x)])$
 - ▶ L_2 Boost : $l(y, f(x)) = (y - f(x))^2/2$
 - ▶ DoomII : $l(y, f(x)) = 1 - \tanh(y \cdot f(x))$
 - ▶ Savage : $l(y, f(x)) = \frac{1}{(1 + \exp(2y \cdot f(x)))^2}$
- ▶ DoomII et Savage sont non-convexes \rightarrow plus robustes aux données bruitées

Gradient Boosting

Pourquoi *Gradient Boosting* ?

- ▶ Chaque étape minimise le risque empirique :
 $\mathcal{L}(F_T) = \sum_{n=1}^N l(y_n, F_T(x_n))$ où l est un coût (*loss*)
- ▶ Lors de la variante additive d'adaboost, $\alpha_T f_T(x)$ peut donc être vu comme le *weak learner* qui approxime le mieux le pas d'une descente de gradient dans l'espaces des fonctions de classification

- ▶ Une version exacte de la descente de gradient donne les Gradient Boosting Models :

$$F_T(x) = F_{T-1}(x) + \alpha_T \sum_{i=1}^N \nabla_{F_{T-1}} l(y_i, f_{T-1}(x_i))$$

Points clés du boosting

- ▶ Agrégation adaptative de classifieurs moyens
- + Résultats théoriques sur la convergence et l'optimalité du classifieur final
- + Très efficace (améliore n'importe quel ensemble de classifieurs)
- + Assez facile à mettre en oeuvre (moins vrai pour Gradient Boosting)
- Sensibilité aux données aberrantes, surapprentissage

Utilisations

- ▶ Choix du *weak learner* : ne doit pas être trop bon, sinon surapprentissage
- ▶ Choix de la pénalité en fonction du bruit des données
- ▶ Variantes pour la classification et la régression

Conclusion

Notions phares du jour

- ▶ Arbres de décision (vote, homogénéité)
- ▶ Aggrégation de classifieurs
- ▶ Bagging, Random Forests
- ▶ Boosting, GradientBoost

Concepts généraux

- ▶ Classification / régression
- ▶ Bagging et randomisation (Forêts aléatoires)
- ▶ Construction adaptative à partir de *weak learners* et optimisation dans l'espace des classifieurs (Boosting)

Références I

- [1] Leo Breiman.
Bagging predictors.
Machine learning, 24(2) :123–140, 1996.
- [2] Leo Breiman.
Random forests.
Machine learning, 45(1) :5–32, 2001.
- [3] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone.
Classification and regression trees.
Routledge, 2017.
- [4] Tianqi Chen and Carlos Guestrin.
Xgboost : A scalable tree boosting system.
In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.
- [5] Yoav Freund and Robert E Schapire.
A decision-theoretic generalization of on-line learning and an application to boosting.
Journal of computer and system sciences, 55(1) :119–139, 1997.
- [6] Jerome H Friedman.
Greedy function approximation : a gradient boosting machine.
Annals of statistics, pages 1189–1232, 2001.
- [7] Jerome H Friedman.
Stochastic gradient boosting.
Computational statistics & data analysis, 38(4) :367–378, 2002.
- [8] Donald Geman and Bruno Jedynak.
Model-based classification trees.
IEEE Transactions on Information Theory, 47(3) :1075–1082, 2001.

Références II

- [9] Trevor Hastie, Robert Tibshirani, and Jerome Friedman.
The elements of statistical learning.
Springer, 2009.
- [10] Hyafil Laurent and Ronald L Rivest.
Constructing optimal binary decision trees is np-complete.
Information processing letters, 5(1) :15–17, 1976.