

1 Introduction

C'est une première version : abondance d'erreurs garantie. Merci de me faire des retours! Dans certains cas, on trouve une propriété et son exacte inverse entre les différents sites... Lorsque RAS est indiqué, c'est que je n'ai pas - pour l'instant - de remarques particulières à émettre sur le sujet.

Les critères de comparaison :

- 1. Classification et/ou régression.
- 2. Performances (en termes d'"accuracy").
- 3. Rapidité à traiter les données (en temps de calcul).
- 4. Facilité à entraîner le modèle (en terme de temps de calcul ou de nombre de données d'entraînement).
- 5. Taille de stockage du modèle.
- 6. Versatilité : traite tout type de données, numériques, catégorielles. Hypothèses sur la structure des données. Prend en compte ou non les dépendances stochastiques ou les dépendances fonctionnelles complexes.
- 7. Adapté aux grands jeux de données (n grand).
- 8. Adapté aux grandes dimensions (d grand).
- 9. Peut-elle traiter le cas $d > n$?
- 10. Simplicité à mettre en œuvre (nombre de paramètres à déterminer).
- 11. Calibration des paramètres (choix délicat d'un paramètre).
- 12. Robustesse à l'échelle des données (scaling : nécessite ou non une normalisation).
- 13. Robustesse au bruit et aux valeurs aberrantes.
- 14. Sensibilité au sur-apprentissage.
- 15. Réduit le biais ou la variance.
- 16. Permet l'obtention de probabilités, de vraisemblances ou de scores.
- 17. Interprétabilité.

2 Régression linéaire (RLin)

- 1. C/R : régression uniquement.
- 2. Perf. : correctes si données vraiment linéaires, mauvaises sinon.
- 3. Rapidité : rapide. Mise à jour rapide si nouvelles données.
- 4. Entraîn. : rapide. Mise à jour rapide si nouvelles données.
- 5. Taille : efficace sur de grands jeux de données, peu de données à stocker.
- 6. Versatilité : non. Hypothèse de linéarité restrictive. Multicolinéarité délicate à gérer.

Traite les données catégorielles, mais de façon pénible (provoque une augmentation de la dimension importante).

- 7. n grand : efficace sur de grands jeux de données.
- 8. d grand : peu efficace en grande dimension, sauf si l'on ajoute un terme de régularisation.
- 9. $d > n$: seulement si terme de régularisation (LASSO).
- 10. Mœ : implémentation facile, même pour de grands jeux de données.
- 11. Calibration : difficile (trouver les dépendances stochastiques, sélection des variables difficiles à l'aide de nombreux tests). Peut-être simplifié par l'utilisation du LASSO.
- 12. Robust. "scaling" : oui.
- 13. Robust. bruit : non; sensible aux valeurs aberrantes.
- 14. Sur-app. : robuste si régularisation (LASSO, etc.).
- 15. b/V : fort biais, faible variance. La régularisation Ridge réduit la variance, le LASSO également.
- 16. Score : non.
- 17. Interprétabilité : facile ; point fort de ce modèle.

Paramètres :

- λ : terme de régularisation ou taux d'apprentissage (si LASSO ou autre).

Forme de l'estimateur :

$$\hat{y}(x) = \beta^T \cdot x \quad (1)$$

β vecteur des coefficients de la régression (y compris biais).

3 Régression logistique (Rlog)

- 1. C/R : classification.
- 2. Perf. : bonnes en classification binaire sur des données de petite taille. Nécessite un jeu de données adapté à l'hypothèse de log-linéarité.
- 3. Rapidité : très efficace sur la classification binaire.
- 4. Entraîn. : mise à jour rapide si nouvelles données.
- 5. Taille : peu de données à stocker.
- 6. Versatilité : non. Hypothèse de (log) linéarité restrictive. Multicolinéarité délicate à gérer (risque d'augmentation forte de la dimension).
- 7. n grand : oui.
- 8. d grand : peu efficace en grande dimension (sauf si régularisation).
- 9. $d > n$: non.
- 10. Mœ : facile si données binaires, moins évident si plus de deux classes.

- 11. Calibration : difficile (trouver les dépendances stochastiques, sélection des variables difficiles à l'aide de nombreux tests). On peut simplifier par utilisation de validation croisée ou sélection automatique de variables.
- 12. Robust. "scaling" : oui.
- 13. Robust. bruit : non; sensible aux valeurs aberrantes.
- 14. Sur-app. : robuste si utilisation d'un paramètre de régularisation.
- 15. b/V : biais élevé, variance faible. La régularisation réduit encore la variance.
- 16. Score : oui. Produit des probabilités en sortie.
- 17. Interprétabilité : assez facile.

Paramètres :

- λ : terme de régularisation ou taux d'apprentissage (si LASSO ou autre).

Forme de l'estimateur :

$$\hat{y}(x) = \Lambda(\beta^T \cdot x) \quad (2)$$

β vecteur des coefficients de la régression (y compris biais). Λ fonction logistique.

4 Classifieur naïf de Bayes (CNB)

- 1. C/R : classification.
- 2. Perf. : mauvaises en moyenne, mais bonnes avec des jeux de données de petite taille sur des tâches simples. Utile en temps réel. Problème de la fréquence zéro.
- 3. Rapidité : très rapide.
- 4. Entraîn. : très rapide.
- 5. Taille : RAS.
- 6. Versatilité : Non. Hypothèse d'indépendance conditionnelle irréalistique. Permet de traiter des données catégorielles, sous condition.
- 7. n grand : peu efficace.
- 8. d grand : assez efficace en grande dimension (si tâche et données adaptées).
- 9. $d > n$: oui (mais attention au problème de la fréquence zéro).
- 10. Mœ : très facile.
- 11. Calibration : très facile.
- 12. Robust. Scale : RAS.
- 13. Robust. bruit : RAS.
- 14. Sur-app. : RAS.
- 15. b/V : RAS.
- 16. Score : oui.
- 17. Interprétabilité : mauvaise.

Paramètres :

- Les probabilités *a priori* des classes et lois.

Forme de l'estimateur :

$$\mathbb{P}[C|X_1, \dots, X_n] \quad (3)$$

5 k plus proches voisins (kNN)

- 1. C/R : classification et régression.
- 2. Perf. : bonnes performances en moyenne, mauvaises performances en grande dimension.
- 3. Rapidité : rapide et efficace en petite dimension. Entrainement très rapide, prédiction très lente.
- 4. Entraîn. : rapide.
- 5. Taille : nécessite de la mémoire pour stocker le modèle. Très lourd si n est grand.
- 6. Versatilité : oui. Ne nécessite aucune hypothèse sur les données. Mais ne modélise pas bien les dépendances complexes.
- 7. n grand : prédiction très lente.
- 8. d grand : non adapté à la grande dimension.
- 9. $d > n$: oui.
- 10. Mœ : intuitif, très simple à mettre en œuvre.
- 11. Calibration : choix de k délicat.
- 12. Robust. "scaling" : oui. Nécessite de normaliser les données.
- 13. Robust. bruit : très sensible au bruit et faible robustesse.
- 14. Sur-app. : RAS.
- 15. b/V :
- 16. Score : non.
- 17. Interprétabilité : mauvaise. S'occupe uniquement des distances entre les données.

Paramètres :

- k nombre de voisins à considérer.
- M nombre d'éléments du dictionnaire (taille de la partition $(A_m)_m$).

Forme de l'estimateur (en régression, puis classification) :

$$\hat{g}_n(x) = \hat{\eta}_n(x) = \sum_{m=1}^M \bar{Y}_m \mathbb{1}_{A_m}(x) \quad (4)$$

$$\hat{g}_n(x) = \mathbb{1}_{[\hat{\eta}_n(x) \geq 1/2]} = \sum_{m=1}^M \mathbb{1}_{[\bar{Y}_m \geq 1/2]} \mathbb{1}_{A_m}(x) \quad (5)$$

6 Arbres de décisions (DT)

- 1. C/R : classification et régression.
- 2. Perf. : généralement mauvaises pour un arbre seul.
- 3. Rapidité : coût calculatoire faible si l'arbre est petit, peut devenir lent si l'arbre est profond et d grand.
- 4. Entraîn. : coût calculatoire faible.
- 5. Taille : nécessite un stockage important si l'arbre a beaucoup de nœuds.
- 6. Versatilité : adapté aux données non linéaires, continues ou discrètes, multilabels ou multimodes.
- 7. n grand : RAS.

- 8. d grand : en très grande dimension, l'arbre devient peu informatif avec un fort risque de sur-apprentissage.
- 9. $d > n$: oui.
- 10. Mœ : très simple à mettre en œuvre.
- 11. Calibration : facile selon certains auteurs, délicate selon d'autres. Il faut bien choisir la profondeur de l'arbre, le nombre de feuilles ou autre critère d'arrêt, ainsi que le paramètre $alpha$ d'élagage de Breiman.
- 12. Robust. "scaling" : RAS.
- 13. Robust. bruit : assez robuste aux valeurs aberrantes, mais très sensible au bruit dans les variables explicatives et aux variations faibles du jeu de données.
- 14. Sur-app. : très sensible au sur-apprentissage et au bruit. Souvent instable.
- 15. b/V : faible biais, variance très élevée.
- 16. Score : oui.
- 17. Interprétabilité : oui en petite dimension, mais peu interprétable en grande dimension.

Paramètres :

Comme pour les k -ppv, la taille de la partition est un paramètre, mais cette taille est plutôt caractérisée par la profondeur de l'arbre, l'effectif des feuilles, le nombre maximum de feuilles, etc.

- k nombre de voisins à considérer.
- Profondeur maximale de l'arbre.
- Effectif maximal d'un nœud.
- α coefficient de régularisation pour l'élagage de l'arbre.

Forme de l'estimateur (en régression, puis classification) :

$$\hat{g}_n(x) = \hat{\eta}_n(x) = \sum_{m=1}^M \bar{Y}_m \mathbb{1}_{A_m}(x) \quad (6)$$

$$\hat{g}_n(x) = \mathbb{1}_{[\hat{\eta}(x) \geq 1/2]} = \sum_{m=1}^M \mathbb{1}_{[\bar{Y}_m \geq 1/2]} \mathbb{1}_{A_m}(x) \quad (7)$$

7 Machine à vecteurs de support (SVM)

- 1. C/R : classification et régression.
- 2. Perf. : très bonnes performances en dimension moyennes, ainsi que sur des données fortement non linéaires. Souvent moins performant que les méthodes à base d'agrégation.
- 3. Rapidité : très rapide si linéaire, coût de calcul important pour des noyaux non linéaires quand $n > 10^4$.
- 4. Entraîn. : rapide si noyau linéaire, coût de calcul important pour des noyaux non linéaires.
- 5. Taille : nécessite un espace mémoire important s'il faut stocker beaucoup de vecteurs supports. Mais pour un SVM linéaire, coût mémoire léger.

- 6. Versatilité : oui, via le choix du noyau. Traite des données très diverses, y compris corrélées.
- 7. n grand : oui pour $n \sim 10^3$, bien moins efficace pour des très grands jeux de données ($> 50k$) dans le cas de noyaux non linéaires. Si le noyau est linéaire, très efficace en grande dimension.
- 8. d grand : inefficace pour des grands jeux de données.
- 9. $d > n$: oui.
- 10. Mœ : facile. Modèle peu complexe, mais choix du noyau délicat.
- 11. Calibration : Nécessite un réglage précis du paramètre de régularisation et du choix du noyau.
- 12. Robust. "scaling" : normalisation obligatoire sur les données.
- 13. Robust. bruit : non, les SVM sont sensibles au bruit.
- 14. Sur-app. : assez robuste au sur-apprentissage (paramètre de régularisation) pour n de taille moyenne, mais sensible au bruit en grande dimension.
- 15. b/V : biais faible, variance modérée.
- 16. Score : non.
- 17. Interprétabilité : mauvaise.

Les méthodes à noyaux sont moins universelles que les méthodes à base d'agrégation (Boosting, Bagging, Stacking) mais peuvent être plus efficaces que ces dernières pour des données peu bruitées avec une taille moyenne (quelques milliers).

Paramètres :

- t ou λ : paramètre de lissage et de régularisation.
- K type de noyau RKHS.
- ξ vecteur éventuel des variables d'écart ("slackening").

Forme de l'estimateur :

$$\hat{h}_{\text{SVM}}(x) = \sum_{i=1}^n \alpha_i K(x_i, x) \in \underset{h \in \mathcal{H}: \|h\| \leq t}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^n \phi(-y_i h(x_i))$$

8 Boosting (Boo)

- 1. C/R : classification et régression.
- 2. Perf. : très bonnes performances (optimisée sur les variantes XGBoost, LightGBM, Catboost).
- 3. Rapidité : Gradient Boosting et XGBoost (sur un CPU) sont exigeants en temps de calcul, LightGBM est rapide et XGBoost également sur des GPU.
- 4. Entraîn. : exigeant en temps de calcul (variantes optimisées).
- 5. Taille : nécessite un espace mémoire important si des arbres sont utilisés comme estimateurs constitutifs. LightGBM utilise des leaf-wise trees très compacts et XGBoost peut stocker des arbres compressés.

- 6. Versatilité : oui. Traite les données catégorielles, complexes, structurées.
- 7. n grand : très bonnes performances si n grand, mais temps de calcul important. XGBoost et LightGBM sont très performants même pour $n > 10^6$.
- 8. d grand : très bonnes performances si d grand, mais temps de calcul important.
- 9. $d > n$: oui.
- 10. Mœ : calibration pas facile selon le nombre de paramètres des estimateurs constitutifs.
- 11. Calibration : facile selon certains auteurs, difficile selon d'autres... Boosting a beaucoup d'hyperparamètres qu'il faut calibrer.
- 12. Robust. "scaling" : RAS. La normalisation peut accélérer la convergence.
- 13. Robust. bruit : Gradient Boosting est sensible aux données aberrantes et au bruit, mais CatBoost est très robuste au bruit et XGBoost robuste grâce à la régularisation.
- 14. Sur-app. : sensible au sur-apprentissage.
- 15. b/\mathbb{V} : réduit biais, en théorie augmente la variance, mais cette augmentation peut être contrôlée facilement (utilisation d'arbres peu profonds, choix du Learning Rate, nombre d'itérations contrôlé).
- 16. Score : non.
- 17. Interprétabilité : mauvaise.

Paramètres :

- M : nombre de classificateurs faibles.
- choix des classificateurs faibles.
- λ : taux d'apprentissage (lissage, régularisation).
- Et en plus, tous les paramètres des estimateurs faibles.

$$\hat{h}_M(x) = \left(\sum_{m=1}^M \alpha_m h_m(x) \right) \quad \text{régression.} \quad (8)$$

$$\text{sgn}(\hat{h}_M(x)) \quad \text{classification.} \quad (9)$$

9 Bagging et forêts aléatoires (RF)

- 1. C/R : classification et régression.
- 2. Perf. : généralement très bonnes performances.
- 3. Rapidité : vitesse d'exécution rapide (calcul parallélisable sur les arbres), mais peut être lent si la forêt est très grande.
- 4. Entraîn. : temps d'entraînement important (mais hautement parallélisable).
- 5. Taille : RAS. Peut devenir volumineux si beaucoup d'arbres.
- 6. Versatilité : oui. Traite les données catégorielles. Adapté également aux données non linéaires. Ne gère pas les données textuelles ou séquentielles.

- 7. n grand : efficace sur de très grands jeux de données.
- 8. d grand : RAS. Bon pour grande dimension, mais pas idéal si d est très grand.
- 9. $d > n$: oui.
- 10. Mœ : facile.
- 11. Calibration : facile selon certains auteurs, difficile selon d'autres...
- 12. Robust. Scale : RAS.
- 13. Robust. bruit : très robuste contre le bruit et les valeurs aberrantes.
- 14. Sur-app. : robuste contre le sur-apprentissage.
- 15. b/\mathbb{V} : réduit la variance.
- 16. Score : non.
- 17. Interprétabilité : pas bonne.

Paramètres :

- M : nombre de classificateurs faibles.
- choix des classificateurs faibles.
- Et en plus, tous les paramètres des estimateurs faibles.
- Si forêt aléatoire : tous les paramètres d'un arbre de décision.

$$\hat{\eta}_B(x) = \frac{1}{B} \sum_{b=1}^B \eta_b(x) \quad (10)$$

η_b construit par Bootstrap.

10 Stacking (Stk)

- 1. C/R : classification et régression.
- 2. Perf. : Excellente si bien paramétré.
- 3. Rapidité : exigeant en temps de calcul (même si parallélisable).
- 4. Entraîn. : nécessite un temps d'entraînement important si les modèles constituant sont lourds. Donc dépend fortement des modèles utilisés.
- 5. Taille : stocke tous les modèles. Donc modèle final potentiellement très volumineux.
- 6. Versatilité : oui. Traite les données catégorielles. Adapté également aux données non linéaires.
- 7. n grand : oui.
- 8. d grand : oui.
- 9. $d > n$: oui.
- 10. Mœ : difficile car il faut initialiser différemment chaque estimateur constitutif.
- 11. Calibration : difficile car il faut initialiser différemment chaque estimateur constitutif. Le bon choix et la bonne combinaison des modèles est délicate à calibrer.
- 12. Robust. Scale : dépend des estimateurs constitutifs.
- 13. Robust. bruit : dépend des estimateurs constitutifs.

- 14. Sur-app. : efficace contre le sur-apprentissage, à condition d'avoir mis en place une validation croisée interne et d'avoir un méta modèle bien régularisé.
- 15. b/\mathbb{V} : réduit le biais, pas forcément idem la variance (mais celle-ci reste contrôlée).
- 16. Score : non.
- 17. Interprétabilité : mauvaise.

Paramètres :

- M : nombre de classificateurs constitutifs.
- choix des classificateurs constitutifs (tous différents).
- Et en plus, tous les paramètres des estimateurs constitutifs (tous différents).

$$\hat{h}_M(x) = \sum_{m=1}^M \alpha_m h_m(x) \quad (11)$$

11 Réseaux de neurones (NN)

- 1. C/R : classification et régression.
- 2. Perf. : généralement très bonnes performances. Excellentes sur certains types de données (vision, audio, textes). Mais parfois performances moins bonnes que les techniques à base d'agrégation.
- 3. Rapidité : Les réseaux de neurones nécessitent des GPU, des dizaines/milliers d'epochs, un algorithme de backpropagation très coûteux. Donc : extrêmement lents à entraîner, mais rapides en prédiction.
- 4. Entraînement : nécessite un temps d'entraînement important si les modèles constitutifs sont lourds.
- 5. Taille : nécessite un très grand volume de données pour l'entraînement. Très coûteux en ressources.
- 6. Versatilité : oui. Adapté à tout type de données.
- 7. n grand : adapté aux grands jeux de données.
- 8. d grand : adapté aux données en grande dimension.
- 9. $d > n$: en principe oui, mais risque de sur-apprentissage massif et nécessite une forte régularisation.
- 10. Mœ : difficile.
- 11. Calibration : difficile; de nombreux paramètres à évaluer précisément.
- 12. Robust. Scale : nécessitent de la normalisation, surtout pour des fonctions d'activation de type ReLu ou tanh.
- 13. Robust. bruit : Non. les RN sont sensibles au bruit qui nécessite de bien calibrer le dropout, early stopping, weight decay.
- 14. Sur-app. : RAS.
- 15. b/\mathbb{V} : faible biais, variance très élevée.
- 16. Score : non.
- 17. Interprétabilité : aucune interprétabilité possible.

Paramètres :

- Nombre de couches.
- Dimension des couches.
- Taux d'apprentissage.
- Type d'optimiseur pour le gradient stochastique.
- Fonctions d'activation.
- Probabilité de "dropout".
- Nombre d'itérations ("epochs").

Forme de l'estimateur :

$$\hat{y}(x) = \bigcup_{i=1}^L \sigma(w_i^T \cdot x + b_i) \quad (12)$$

$$= \sigma(w_L \sigma(\dots \sigma(w_1^T \cdot x + b_1) \dots)) \quad (13)$$

où σ fonction d'activation (ReLU, logistique, tanh, etc.), L est le nombre de couches.