Quelques éléments pour mieux comprendre les méthodes ensemblistes

Nathan Noiry et Yannick Guyonvarch

1 Les méthodes basées sur le rééchantillonage

Les méthodes comme le bagging ou les forêts aléatoires reposent sur la capacité à construire à partir d'un unique jeu de données un ensemble de classifieurs/prédicteurs indépendants et à les aggréger pour obtenir un classifieur/prédicteur final plus performant. Comment est-il possible d'apprendre des modèles indépendants avec un seul jeu de données? Il faut avoir recours aux méthodes dites de rééchantillonage (bootstrap et subsampling essentiellement) qui consistent à créer artificiellement un surcroît d'aléa dans les données à disposition. Dans ce qui suit, nous notons pour $1 \leq i \leq n$, $z_i = (y_i, x_i)$ la i-ème observation.

1.1 Le bootstrap

Nous allons raisonner conditionnellement aux données observées $(z_i)_{i=1}^n$, ce qui revient à traiter $(z_i)_{i=1}^n$ comme une suite de vecteurs fixés, déterministes. L'idée du bootstrap est de générer des nouvelles observations artificiellement en tirant dans une loi uniforme sur les points $\{z_1, \ldots, z_n\}$. Cet ensemble contient n points, les poids du tirage uniforme sont donc $1/n$. En effectuant un tirage uniforme de la sorte, j'obtiens une variable aléatoire Z_1^* qui vaut z_1 avec probabilité $1/n$, z_2 avec probabilité $1/n$, etc... Si j'effectue un deuxième tirage uniforme de la même manière, j'obtiens une nouvelle variable Z_2^* qui est indépendante de Z_1^* et a la même loi. Construire un échantillon bootstrap de taille m revient à construire un échantillon $(Z_1^*, \ldots, Z_m^*)$. Remarquez qu'il peut y avoir des répétitions dans cet échantillon: la probabilité que tous les Z_i^* soient égaux est même non-nulle. Cela explique l'autre nom de ce tirage: tirage uniforme de taille m avec remise. En pratique, m est choisi comme une fraction de n ($m = n$ ou $m = n/2$ sont courants) et donc il n'y a pas un nombre très important de répétitions dans $(Z_i^*)_{i=1}^m$ en pratique. Pour le bagging ou les forêts aléatoires, un échantillon $(Z_i^*)_{i=1}^m$ va être utilisé pour construire un classifieur/régresseur (un arbre). Lorsque l'on construit une forêt à B arbres, cela signifie que l'on réalise l'opération suivante:

1. pour b allant de 1 à B, j'effectue un tirage uniforme de taille m_b:
   (a) j'obtiens un échantillon $(Z_{i,b}^*)_{i=1}^{m_b}$;
   (b) à partir de cet échantillon, j'apprends l'arbre b qui est un classifieur/régresseur de la forme $x \mapsto T_b(x)$.
2. j'obtiens ma forêt (mon classifieur/régresseur final) en combinant les arbres: $F(x) = g(T_1(x), \ldots, T_B(x))$.

Il est important d'avoir en tête que les B échantillons $(Z_{i,b}^*)_{i=1}^{m_b}$ sont bien indépendants entre eux, et donc les arbres $(T_b(\cdot))_{b=1}^B$ sont appris indépendamment les uns des autres. A titre d'exemple, pour une tâche de régression, chaque arbre sera un arbre de régression et la forêt prendra la forme $F(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$. Il est en général recommandé de prendre B grand ($B \geq 100$). Par ailleurs, m_b est en général choisi identique pour tous les tirages bootstrap ($m_b = n$ ou

$m_b = n/2$ sont de bons choix. Une règle importante est de prendre m_b de l'ordre de n). Pour évaluer la performance de votre forêt, il est conseillé d'effectuer une séparation train/test, de construire la forêt comme indiqué précédemment uniquement sur le train (*i.e* $(z_i)_{i=1}^n$ correspond à l'échantillon train) et de vérifier sa qualité sur le test.

Il existe une autre technique de rééchantillonage connue qui permet de ne pas tirer plusieurs fois la même observation dans un échantillon bootstrap. Cette méthode s'appelle le subsampling (ou sous-échantillonage en français).

1.2 Le subsampling

Pour construire T_b, nous construisons un nouvel échantillon à partir des données observées de la manière suivante: nous tirons m_b éléments dans l'ensemble $\{z_1, \dots, z_n\}$ uniformément mais sans remise. Autrement dit, nous tirons $Z_{1,b}^*$ uniformément dans $\{z_1, \dots, z_n\}$ (loi uniforme sur n points), puis nous tirons $Z_{2,b}^*$ uniformément dans $\{z_1, \dots, z_n\}$ privé de $Z_{1,b}^*$ (loi uniforme sur $n-1$ points), etc... A la différence du bootstrap, les éléments sélectionnés $(Z_{i,b}^*)_{i=1}^{m_b}$ ne sont pas indépendants. Par contre les échantillons pour $b \neq b'$ demeurent indépendants, et donc les arbres T_b et $T_{b'}$ aussi. Remarquons qu'il n'est pas intéressant de choisir $m_b = n$ pour le subsampling, car nous récupérons simplement l'échantillon initial. En pratique, m_b est souvent choisi comme une fraction de n. Quand n est grand, il est préférable de prendre $m_b = n^\gamma$ ($\gamma \in (0,1)$) car pour certains problèmes d'apprentissage, la validité théorique du subsampling n'est assurée que en prenant $m_b = o(n)$.

2 Gradient boosting

Cet algorithme est une généralisation d'AdaBoost à un cadre plus large que la classification (régression en particulier). Nous avons des labels $(Y_i)_{i=1}^n$, des features $(X_i)_{i=1}^n$ et une classe $\mathcal{F}$ d'experts possibles. Comme pour AdaBoost, les experts vont être des prédicteurs intermédiaires que l'on va chercher à combiner. Nous nous donnons une fonction de perte $\ell(y, x)$ différentiable en x (ex: moindres carrés en régression, perte exponentielle ou logistique en classification binaire...) et un nombre fixé d'itérations T.

La première étape de l'algorithme consiste à minimiser la fonction de perte agrégée pour trouver notre premier expert F_1

$$F_1 \in \operatorname*{argmin}_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, F(X_i))$$

Il est important de remarquer que même en classification, les experts que l'on cherche à construire sont des fonctions de régression (*i.e* des approximations de $\mathbb{P}(Y = 1 \mid X = x)$). Ce n'est qu'à la fin que l'on renvoie un prédicteur qui à pour valeurs -1 ou 1.

Ensuite, pour chaque itération $2 \leq t \leq T$, nous allons réaliser plusieurs étapes. Tout d'abord, nous cherchons

$$f_t \in \operatorname*{argmin}_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\big(Y_i, F_{t-1}(X_i) + f(X_i)\big)$$

Au lieu de résoudre directement ce problème, remarquons qu'à l'aide d'un développement de Taylor très heuristique, nous pouvons écrire

$$\min_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} \ell(Y_i, F_{t-1}(X_i) + f(X_i)) \approx \min_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} \Big\{ \ell\big(Y_i, F_{t-1}(X_i)\big) + f(X_i)\partial_2\ell\big(Y_i, F_{t-1}(X_i)\big) \Big\}$$

avec ∂_2 la dérivée par rapport au deuxième argument. Comme $\frac{1}{n}\sum_{i=1}^{n} \ell(Y_i, F_{t-1}(X_i))$ ne dépend pas de f, nous obtenons approximativement

$$f_t \in \operatorname*{argmin}_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} f(X_i)\partial_2\ell(Y_i, F_{t-1}(X_i)) = \frac{1}{n}\langle \overrightarrow{f}, \overrightarrow{\partial_2\ell}\rangle,$$

où $\overrightarrow{f} = (f(X_1), \cdots, f(X_n))^T$ et $\overrightarrow{\partial_2\ell} = (\partial_2\ell(Y_1, F_{t-1}(X_1)), \cdots, \partial_2\ell(Y_1, F_{t-1}(X_1)))^T$. Il se trouve que l'on peut montrer que à un coefficient d'échelle près, $f_t(x)$ doit être choisi égal à $-\partial_2\ell(y_i, F_{t-1}(x_i))$ (par un argument du même type que celui derrière l'algorithme de descente de gradient). L'idée du gradient boosting est de trouver f_t en minimisant en f une perte $\widetilde{\ell}$ entre $-\partial_2\ell(y, F_{t-1}(x))$ (qui remplace y) et $f(x)$. Cette nouvelle perte peut être prise égale à la première perte ℓ ou non. Une fois f_t calculé, nous pouvons calculer le véritable expert à l'itération t, F_t de la manière suivante:

$$F_t(x) = F_{t-1}(x) - \gamma_t f_t(x),$$

où γ_t est choisi à la manière du paramètre de pas de gradient dans un algorithme de descente de gradient.

Après T itérations, nous obtenons notre expert final F_T. Celui-ci est directement notre prédicteur dans le cas de la régression. Dans le cas de la classification, ce n'est pas possible car F_M est une fonction continue, qui n'est même pas à valeur dans $(0, 1)$ a priori. Ainsi le prédicteur "hard" est $\text{sign}(F_T(x))$ et le prédicteur "soft" va dépendre de la perte/du modèle choisi. Si l'on souhaite récupérer les probabilités prédites associées à un modèle Logistique, il faut calculer $\sigma(F_T(x))$ où σ est la fonction sigmoïde.