Méthodes par ensemble

Introduction à l'apprentissage automatique – GIF-4101 / GIF-7005

Professeur : Christian Gagné

Semaine 11

UNIVERSITÉ LAVAL

11.1 Notions d'ensemble

Méthodes par ensemble

- Théorème du *no free lunch*
  - Aucun algorithme d'apprentissage n'est supérieur aux autres pour tous problèmes
- Arguments statistiques pour l'utilisation d'ensembles
  - Moyenne d'un ensemble d'échantillons plus fiable que valeur d'un seul échantillon
  - Éliminer la variance en moyennant les décisions d'ensemble
  - Retire le bruit des décisions individuelles des classifieurs
- Plusieurs têtes valent mieux qu'une
  - Méthodes par votes
  - Codes à correction d'erreurs
  - Échantillonnage dynamique de données ou de caractéristiques
  - Mixtures d'experts

Théorème du jury de Condorcet

- Quelle est la probabilité qu'un jury obtienne une décision à majorité qui soit correcte ?
  - Deux décisions possibles : décision correcte ou décision erronée
  - Chaque jury a une probabilité p de prendre une décision correcte
  - Lorsque la probabilité $p > 1/2$, la probabilité de décision correcte du jury tend vers 1 avec un très grand nombre de participants au jury
  - Inversemement, avec une probabilité $p < 1/2$, la probabilité de décision du jury correcte est réduite par l'augmentation de la taille du jury
  - Suppose que les votes sont indépendants et identiquement distribués (iid)
- Proposé par le Marquis de Condorcet en 1785
  - Justification mathématique de la démocratie, étudiée en science politique

Approches pour création d'ensembles

- Différents algorithmes d'apprentissage
  - Différentes hypothèses sur les données (biais et variance)
- Différents hyperparamètres
  - Nombre de neurones/couches cachées
  - Nombre de voisins
  - Type de matrice de covariance
- Différentes représentations
  - Différentes mesures/capteurs
  - Différentes caractéristiques (forêt aléatoire, *random subspaces*)
- Différents jeux de données d'entraînement
  - Échantillonnages aléatoires des données (*bagging*)
  - Échantillonnages selon données mal classées (*boosting*)

Complexité, combinaison, formalisation

- Complexité des classifieurs de base
  - Classifieurs de base n'ont pas à être très précis individuellement
  - La simplicité est souvent préférable à la performance
  - Diversité dans le classement, spécialisation dans certains domaines
  - Si les erreurs des classifieurs sont iid

$$\lim_{L\to\infty} E_{ensemble} \to E_{Bayes}$$

- Approches pour combinaisons
  - Combinaisons d'experts multiples (parallèle)
    - Votes, mixtures d'experts, *stacked generalization*
  - Combinaisons multi étages (série)
    - Classifieurs d'étages suivants appelés seulement lorsque doutes aux étages précédents (classifieurs en cascade)
- Formalisation des méthodes par ensemble

$$\bar{\mathrm{h}}(\mathbf{x}|\Phi) = \mathrm{f}(\mathrm{h}_1(\mathbf{x}), \mathrm{h}_2(\mathbf{x}), \ldots, \mathrm{h}_L(\mathbf{x})|\Phi)$$

11.2 Votes et combinaison bayésienne

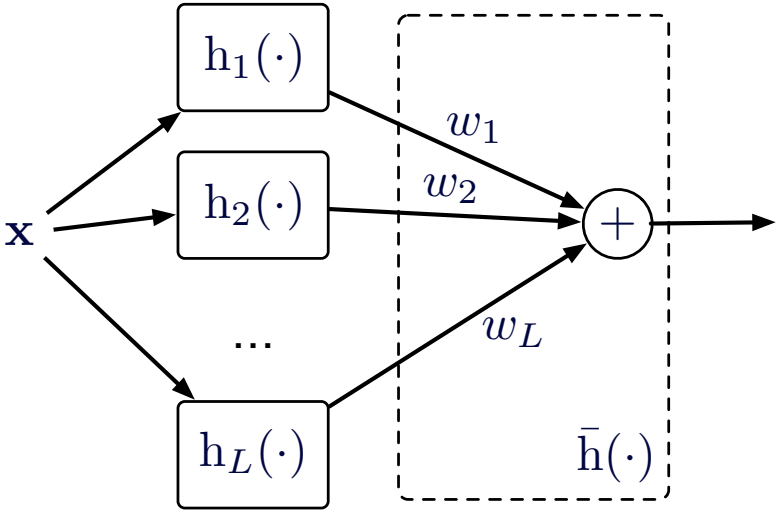
Votes

- Méthode des votes
  - Assigner à la classe la plus fréquente parmi les réponses des classifieurs de base
- Formulation générale : pondérer chaque vote par un facteur w_j

$$\bar{\mathrm{h}}(\mathbf{x}) = \sum_{j=1}^{L} w_j \mathrm{h}_j(\mathbf{x}), \quad \text{où } w_j \geq 0, \forall j \text{ et } \sum_j w_j = 1$$

  - Modèle linéaire de combinaison parallèle
  - Dans le cas de vote simple, $w_j = 1/L$
  - Poids peuvent représenter la confiance dans chaque classifieur

Votes


$\mathrm{h}_1(\cdot)$
$\mathrm{h}_2(\cdot)$
...
$\mathrm{h}_L(\cdot)$
$\mathbf{x}$
w_1
w_2
w_L
+
$\bar{\mathrm{h}}(\cdot)$

Combinaison de modèles bayésiens

- Modèle bayésien de combinaison

$$P(C_i|\mathbf{x}) = \sum_{\forall \mathcal{M}_j} P(C_i|\mathbf{x},\mathcal{M}_j)P(\mathcal{M}_j)$$

  - $w_j = P(\mathcal{M}_j)$ et $\mathrm{h}_j(\mathbf{x}) = P(C_i|\mathbf{x},\mathcal{M}_j)$
  - Vote simple est le cas de probabilités *a priori* égales, $P(\mathcal{M}_j) = 1/L$

Biais et variance

- Biais et variance dans les ensembles de classifieurs à deux classes
  - h_j sont iid, avec espérance $\mathbb{E}[\mathrm{h}_j]$ et variance $\mathrm{Var}(\mathrm{h}_j)$

$$\mathbb{E}[\bar{\mathrm{h}}] = \mathbb{E}\left[\sum_{j=1}^{L} \frac{1}{L}\mathrm{h}_j\right] = \frac{1}{L} L\, \mathbb{E}[\mathrm{h}_j] = \mathbb{E}[\mathrm{h}_j]$$

$$\mathrm{Var}(\bar{\mathrm{h}}) = \mathrm{Var}\left(\sum_{j=1}^{L} \frac{1}{L}\mathrm{h}_j\right) = \frac{1}{L^2} L\, \mathrm{Var}(\mathrm{h}_j) = \frac{1}{L}\mathrm{Var}(\mathrm{h}_j)$$

- Variance diminue lorsque le nombre de voteurs indépendants L augmente
  - Avec ensembles, on peut donc réduire la variance sans affecter le biais
  - Erreur quadratique est d'autant réduite

Diversité et corrélation négative

- Variance d'ensembles, cas général

$$\mathrm{Var}(\bar{\mathrm{h}}) = \frac{1}{L^2}\mathrm{Var}\left(\sum_j \mathrm{h}_j\right) = \frac{1}{L^2}\left[\sum_j \mathrm{Var}\left(\mathrm{h}_j\right) + 2\sum_j \sum_{i>j} \mathrm{Cov}(\mathrm{h}_j, \mathrm{h}_i)\right]$$

  - Réduction supplémentaire de la variance avec voteurs corrélés négativement
  - Erreur quadratique peut être réduite, pourvu que la corrélation négative n'affecte pas le biais de l'ensemble
- Diversité dans les réponses des classifieurs d'ensemble
  - Objectif dans la formation d'ensemble : obtenir des classifieurs ne faisant pas les mêmes erreurs
  - Cas limite d'ensemble sans diversité : L copies du même classifieur

11.3 Matrices de décision et codes à correction d'erreur

Matrice de décision

- Classement multi classes avec ensembles, avec vote pondéré

$$\bar{\mathrm{h}}_i(\mathbf{x}) = \sum_j w_{i,j} \mathrm{h}_{j,i}(\mathbf{x})$$

- Matrice de décision $\mathbf{W}$: valeurs des poids $w_{i,j}$
- Matrice de décision pour classement un contre tous (exemple avec $L = K = 4$)

$$\mathbf{W} = \begin{bmatrix} +1 & -1 & -1 & -1 \\ -1 & +1 & -1 & -1 \\ -1 & -1 & +1 & -1 \\ -1 & -1 & -1 & +1 \end{bmatrix}$$

- Ambiguïté lorsque mauvaise décision d'un classifieur de base
  - Deux valeurs $\bar{\mathrm{h}}_i(\mathbf{x}) = 0$
  - Similarité trop élevée entre les codes (faible distance de Hamming)

Ensembles avec redondance

- Matrice de décision pour décisions par paires (exemple avec $K = 4$, $L = K(K-1)/2 = 6$)

$$\mathbf{W} = \begin{bmatrix} +1 & +1 & +1 & 0 & 0 & 0 \\ -1 & 0 & 0 & +1 & +1 & 0 \\ 0 & -1 & 0 & -1 & 0 & +1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{bmatrix}$$

  - Valeur de $w_{i,j} = 0$ signifie que la décision est ignorée
  - Erreur d'un classifieur de base n'implique pas nécessairement une ambiguïté
  - Valeur L croît quadratiquement selon K
- Généralisation de l'approche : codes à correction d'erreur
  - Utiliser une matrice de décision $\mathbf{W}$ de taille L préétablie
  - Distance de Hamming entre lignes est maximisée

Codes à correction d'erreur

- Codes à correction d'erreur (CCE)
  - Avec K classes, il y a $2^{(K-1)} - 1$ problèmes à deux classes différents
  - Diversité de discriminants : colonnes différentes
  - Correction d'erreur : composantes différentes pour une ligne
- Exemple de matrice avec CCE ($K = 4$ et $L = 9$)

$$\mathbf{W} = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 & +1 & +1 \\ -1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 & -1 \\ -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & +1 \\ +1 & -1 & +1 & -1 & +1 & -1 & +1 & +1 & -1 \end{bmatrix}$$

  - Différence (distance de Hamming) minimale de $d = 5$ entre chaque paire de lignes
    - Tolère donc jusqu'à $\lfloor \frac{d-1}{2} \rfloor = \lfloor \frac{5-1}{2} \rfloor = 2$ erreurs de classifieurs de base
  - Choix de la classe selon $\bar{\mathrm{h}}_i(\mathbf{x})$ maximum
  - Valeur $\bar{\mathrm{h}}_i(\mathbf{x})$ normalisée dans $[0, 1]$ peut être interprétée comme une probabilité
  - Choix des valeurs $\mathbf{W}$ en partie arbitraires, certaines dichotomies peuvent être plus difficiles que d'autres

11.4 Bagging, random subspaces et forêts aléatoires

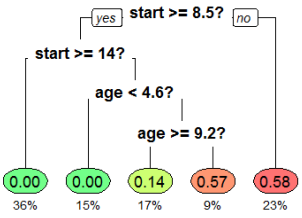
Bagging et random subspaces

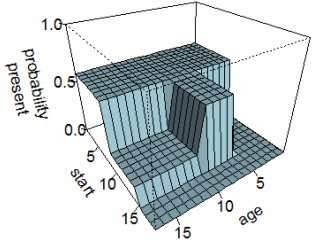
- *Bagging* : ensemble de classifieurs entraînés sur ensembles de données légèrement différents
  - Chaque classifieur de base entraîné sur ensemble $\mathcal{X}_j$
    - $\mathcal{X}_j$: tirage **avec remise** de N données dans $\mathcal{X}$
    - Remise : plusieurs exemplaires de certaines données, absence de certaines autres
  - Idéalement, classifieurs de base devraient être instables
    - Algorithme d'entraînement instable : pour jeux de données légèrement différents, donne des classifieurs avec comportements différents
    - Stable : k-plus proches voisins, classement paramétrique
    - Instable : perceptron multicouche, condensation de Hart
    - En général, algorithmes instables ont une grande variance
- *Random subspaces*
  - Générer chaque classifieur de base par un échantillonnage aléatoire d'un sous-ensemble de caractéristiques

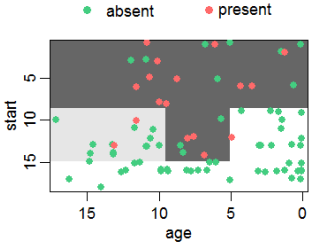
Arbres de décision

- Arbres de décision
  - Séparation hiérarchique (récursive) de l'espace d'entrée
  - Chaque nœud de l'arbre est un test sur valeur avec issues discrète
  - Effectue une division de plus en plus fine de l'espace d'entrée
- Propriétés des arbres de décision
  - Construction descendante (*top-down*) des arbres selon critère de performance (ex. entropie)
  - Élagage (*pruning*) permet réduire sur-spécialisation
  - Utile pour extraire des règles de décision interprétables

Arbres de décision







Par Stephen Milborrow, CC-SA 4.0, https://commons.wikimedia.org/wiki/File:Cart_tree_kyphosis.png.

Forêt aléatoire

- Problème avec arbres de décision pour classement
  - Classifieurs à biais faible et variance élevée
  - Ce qui implique risque élevé de sur-apprentissage (même si élagage est utilisé)
- Solution : faire ensemble d'arbres
  - Moyennage permet de garder biais faible tout en réduisant variance d'ensemble
  - Mais ensemble doit comporter diversité d'arbres
- Générer arbres « aléatoires » avec bagging et random subspaces
  - Pour apprendre chaque nœud, utiliser sous-ensembles de données et de variables différents
- Ensemble d'arbres aléatoires correspond à une forêt aléatoire
  - Moyenner décisions des arbres
  - Variance sur les décisions est un bon indicateur de la confiance de l'ensemble

11.5 Boosting

Boosting

- *Bagging* : nécessite des algorithmes instables
  - Diversité générée passivement
- *Boosting* : générer activement de nouveaux classifieurs à partir des données difficiles pour les classifieurs actuels
  1. Diviser aléatoirement le jeu de données en trois ($\mathcal{X}_1$, $\mathcal{X}_2$ et $\mathcal{X}_3$)
  2. Entraîner classifieur h_1 sur $\mathcal{X}_1$
  3. Évaluer données $\mathcal{X}_2$ avec h_1, utiliser données mal classées et un nombre égal de données bien classées pour former $\mathcal{X}_2'$
  4. Entraîner classifieur h_2 sur $\mathcal{X}_2'$
  5. Évaluer données $\mathcal{X}_3$ avec h_1 et h_2, utiliser données où h_1 et h_2 en désaccord pour former $\mathcal{X}_3'$
  6. Entraîner classifieur h_3 sur $\mathcal{X}_3'$
- Évaluer classement de données : tester données avec h_1 et h_2, si désaccord utiliser décision de h_3
- Améliore les performances, mais requiert de très grands jeux de données

AdaBoost

- AdaBoost (*adaptive boosting*) : réutiliser le même jeu de données pour les classifieurs de base
  - Contrairement au *boosting* classique, ne requiert pas de très grands jeux de données
  - Peut générer un nombre arbitrairement élevé de classifieurs
- AdaBoost.M1 : la probabilité d'échantillonner une donnée change en fonction des erreurs des classifieurs de base
  - Initialement, $p_1^t = 1/N$, $t = 1, \ldots, N$
  - Échantillonner jeu $\mathcal{X}_j$ à partir de $\mathcal{X}$ selon probabilités p_j^t
  - Entraîner classifieur h_j avec $\mathcal{X}_j$
  - Si taux d'erreur de h_j supérieur à $\epsilon_j > 0{,}5$, interrompre l'algorithme, $\epsilon_j = \sum_t p_j^t \ell_{0-1}(r^t, \mathrm{h}_j(\mathbf{x}^t))$
  - Calculer les probabilités p_{j+1}^t selon le classement de $\mathcal{X}$ avec h_j
  - Répéter pour générer les L classifieurs de base

Weak learner

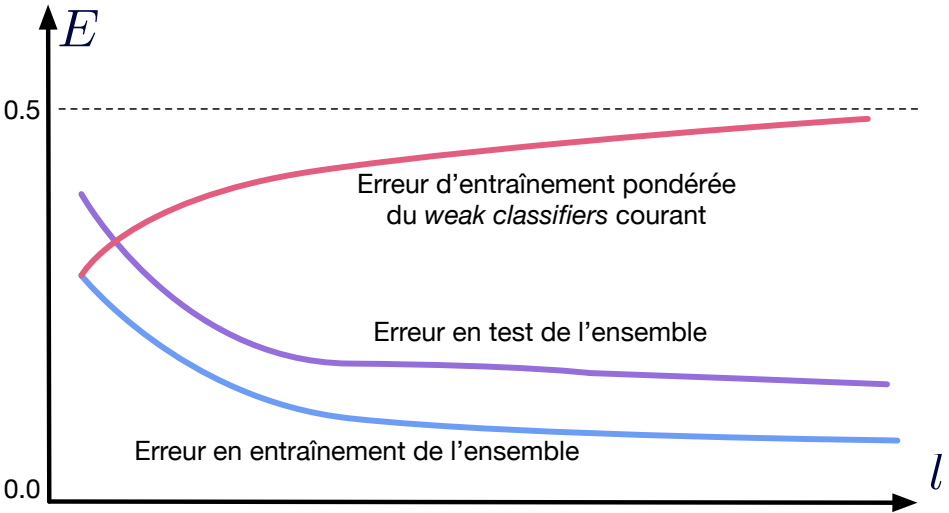
- *Boosting* et AdaBoost ne nécessitent pas de classifieurs très précis
  - *Weak learner* : algorithme ayant une probabilité d'erreur de moins de $1/2$ en deux classes (mieux qu'un classement aléatoire) et relativement instable (variations soutenues dans le classement)
  - Utiliser des *weak learners* permet une bonne diversité dans le classement
- Souches de décision : *weak learner* couramment utilisé avec AdaBoost
  - Décisions basées sur un seuil appliqué à une seule dimension

$$\mathrm{h}(\mathbf{x}|\theta,\upsilon,\gamma) = \mathrm{sgn}(\theta(x_\gamma - \upsilon)), \quad \theta \in \{-1,1\},\ \gamma \in \{1,\dots,D\},\ \upsilon \in \mathbb{R}$$

  - Entraînement déterministe de souches de décisions

$$\begin{aligned}
\tilde{x}_j^k &= x_j^t \mid \tilde{x}_j^1 \leq \tilde{x}_j^2 \leq \cdots \leq \tilde{x}_j^{k-1} \leq x_j^t \leq \tilde{x}_k^{k+1} \leq \cdots \leq \tilde{x}_j^N \\
v_j^k &= 0{,}5(\tilde{x}_j^k + \tilde{x}_j^{k+1}),\ k = 1,\dots,N-1 \\
\mathcal{A}_j &= \left\{(s_j, v_j^k, j) \mid \forall s_j \in \{-1,1\},\ \forall k \in \{1,\dots,N-1\}\right\} \\
\mathcal{A} &= \mathcal{A}_1 + \mathcal{A}_2 + \cdots + \mathcal{A}_D \\
(\theta,\upsilon,\gamma) &= \underset{(s_j,u_j^k,j)\in\mathcal{A}}{\mathrm{argmin}} \; E(\mathrm{h}(\cdot|s_j,u_j^k,j)|\mathcal{X})
\end{aligned}$$

Erreurs avec AdaBoost



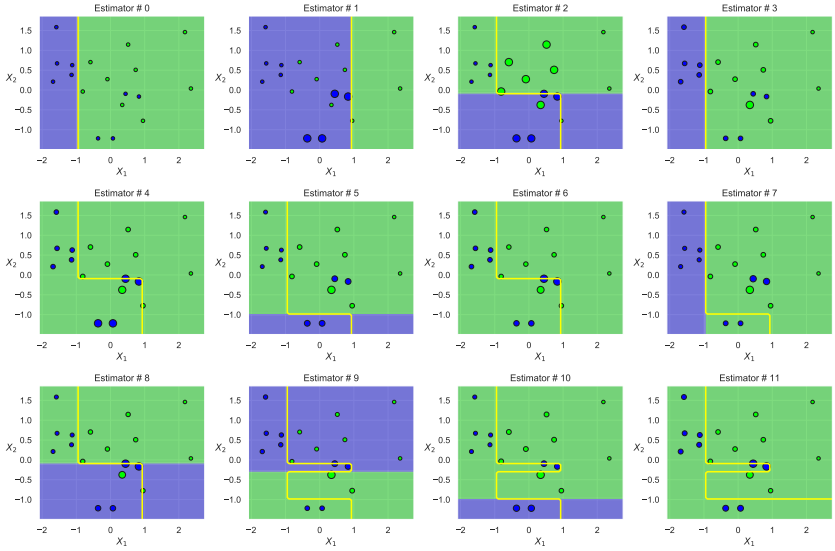
Algorithme AdaBoost

1. Initialiser les probabilités de chaque donnée, $p_1^t = 1/N$, $t = 1,\ldots,N$
2. Pour chaque classifieur de base $j = 1,\ldots,L$:
   - 2.1 Échantillonner jeu $\mathcal{X}_j$ à partir de $\mathcal{X}$ selon probabilités p_j^t
   - 2.2 Entraîner classifieur h_j avec jeu $\mathcal{X}_j$
   - 2.3 Calculer l'erreur du classifieur, $\epsilon_j = \sum_t p_j^t \ell_{0-1}(r^t, \mathrm{h}_j(\mathbf{x}^t))$
   - 2.4 Si erreur $\epsilon_j > 0{,}5$, alors $L = j - 1$ et arrêter l'algorithme
   - 2.5 Calculer $\beta_j = \frac{\epsilon_j}{1-\epsilon_j}$
   - 2.6 Calculer les nouvelles probabilités p_{j+1}^t

$$p_{j+1}^t = \frac{q_j^t}{\sum_s q_j^s}, \quad q_j^t = \begin{cases} \beta_j p_j^t & \text{si } \mathrm{h}_j(\mathbf{x}^t) = r^t \\ p_j^t & \text{autrement} \end{cases}, \quad t = 1,\ldots,N$$

Évaluation du classement d'une donnée : $\bar{\mathrm{h}}(\mathbf{x}) = \sum_{j=1}^{L} \left(\log \frac{1}{\beta_j} \right) \mathrm{h}_j(\mathbf{x})$

Exemple avec AdaBoost



Maximisation des marges avec AdaBoost

- AdaBoost maximise les marges pour le classement
  - Apprentissage avec probabilités plus fortes pour les données difficiles à classer
    - Données difficiles : données dans la marge
  - $\bar{\mathrm{h}}_i$ est le résultat d'un vote pondéré

$$\bar{\mathrm{h}}_i = \frac{\text{votes pour classe } i - \text{votes contre classe } i}{\text{nombre total de votes}}$$

  - Avec grand nombre de classifieurs, $\bar{\mathrm{h}}_i(\mathbf{x}) \to 1$ si $\mathbf{x} \in C_i$ et $\bar{\mathrm{h}}_i(\mathbf{x}) \to -1$ autrement
  - Larges marges $\Rightarrow$ meilleure généralisation
- Nombreuses variantes de *boosting*
  - LPBoost : apprendre les $\alpha_j = \log \frac{1}{\beta_j}$ par programmation linéaire
    - À chaque génération de classifieur de base, réapprend les α_j de tous les classifieurs actuels
    - Nombreux parallèles à faire avec les SVM

11.6 Autres modèles de combinaison

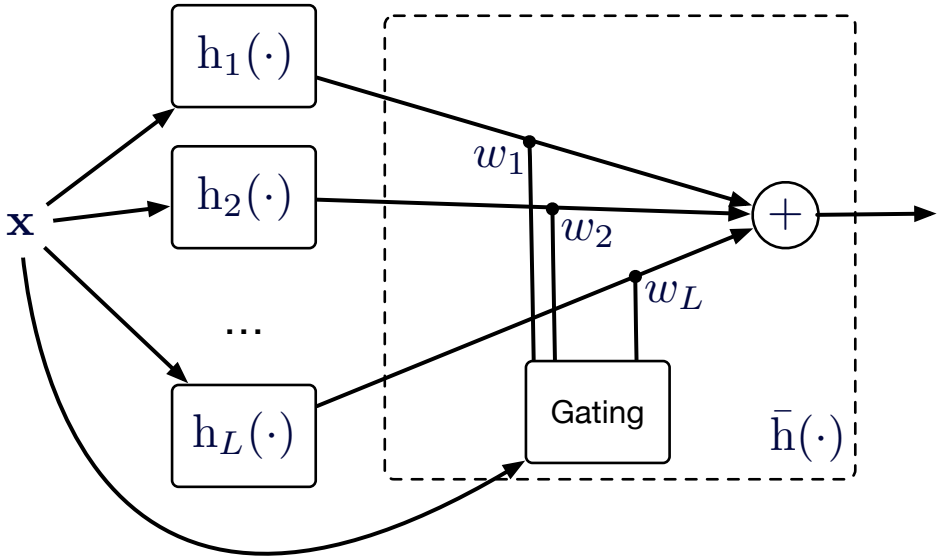
Mixture d'experts

- Mixture d'experts
  - Classifieurs-experts spécialisés sur certains aspects du problème
  - Travaillent en parallèle, avec fonction de routage pondérant les décisions selon l'expertise
  - Similaire au vote pondéré, mais avec pondération non constante

$$\bar{\mathrm{h}}(\mathbf{x}) = \sum_{j=1}^{L} w_j(\mathbf{x}) \mathrm{h}_j(\mathbf{x})$$

- Spécialisation dans différentes régions de l'espace réduit corrélation
- Génère donc des experts biaisés, mais négativement corrélés
  - Implique une réduction globale de la variance, donc de l'erreur
- Fonction de routage peut être non linéaire (ex. perceptron multicouche)
  - Peut réduire le biais, au risque d'augmenter la variance (sur apprentissage)

Mixtures d'experts



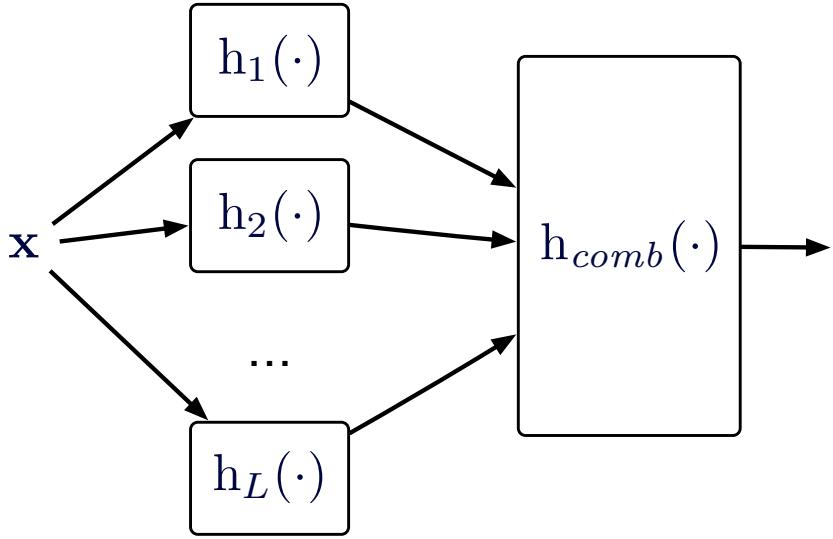
Stacked generalization

- *Stacked generalization* : système à deux étages
  - Premier étage : classifieurs de base fonctionnant en parallèle
  - Deuxième étage : système de combinaison associant sortie des classifieurs de base avec étiquette désirée

$$\bar{\mathrm{h}}(\mathbf{x}) = \mathrm{h}_{comb}(\mathrm{h}_1(\mathbf{x}),\mathrm{h}_2(\mathbf{x}),\ldots,\mathrm{h}_L(\mathbf{x}))$$

- Système de combinaison : classifieur standard
  - Apprend comment les classifieurs de base font des erreurs
  - Entraînement du système de combinaison doit se faire sur données non vues par les classifieurs de base
  - Permet d'estimer et de corriger les biais des classifieurs de base

Stacked generalization



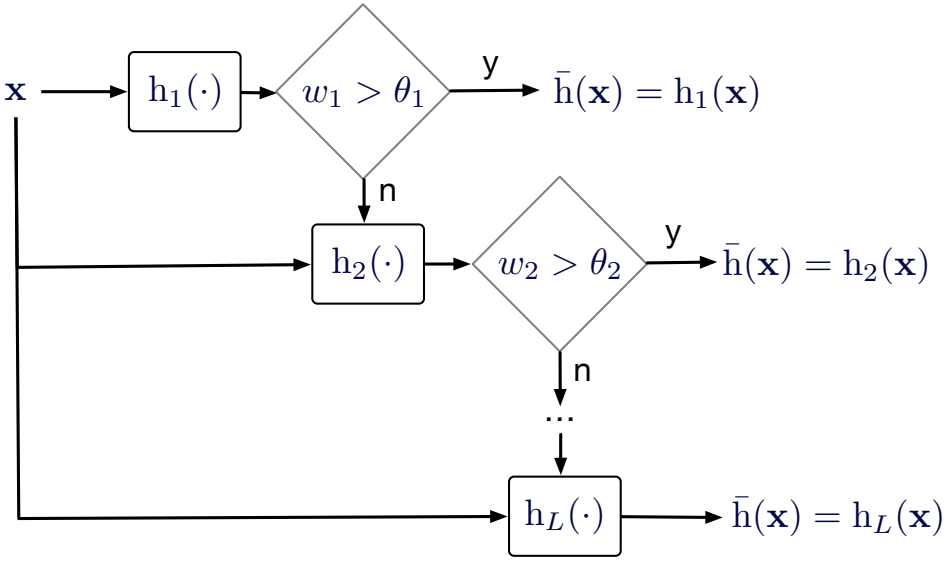
Classifieurs en cascade

- Classifieurs en cascade : séquence de classifieurs de base
  - Passage d'un étage à l'autre si le classifieur k a une confiance faible en son classement, $w_j(\mathbf{x}) < \theta_j$

$$\bar{\mathrm{h}}(\mathbf{x}) = \mathrm{h}_j(\mathbf{x}) \quad \text{si } w_j(\mathbf{x}) \geq \theta_j \text{ et } w_k(\mathbf{x}) < \theta_k, \forall k < j$$

  - Confiance $w_j(\mathbf{x})$ peut correspondre à la probabilité *a posteriori* $P(C_i|\mathbf{x})$ du classifieur
  - Seuil sur la confiance θ_j devrait être élevé (taux de rejet élevé) pour les premiers étages
- Entraînement d'une cascade
  - Classifieur h_1 entraîné avec $\mathcal{X}_1 = \mathcal{X}$
  - Jeu $\mathcal{X}_{j+1}$ est formé des rejets de $\mathcal{X}_j$ avec classifieur h_j
  - Classifieur h_{j+1} entraîné avec jeu $\mathcal{X}_{j+1}$
- Classifieurs de base de complexités croissantes
  - Classifieurs simples (peu coûteux) gèrent la plupart des cas
  - Classifieurs complexes (coûteux) sur les derniers étages gèrent les cas difficiles

Classifieurs en cascade



Surproduction et sélection

- $\bar{\mathrm{h}}(\mathbf{x}|\Phi) = f(\mathrm{h}_1(\mathbf{x}), \mathrm{h}_2(\mathbf{x}), \ldots, \mathrm{h}_L(\mathbf{x})|\Phi)$: métaclassifieur
  - Chaque classifieur $\mathrm{h}_i(\mathbf{x})$ peut être vu comme une caractéristique (ou une fonction de base) du méta classifieur
- Surproduction et sélection
  - Générer une vaste variété de classifieurs candidats
    - Ex. méthode des *random subspaces*
  - Sélectionner un sous-ensemble de ces classifieurs pour former l'ensemble final
- Sélection possible par les méthodes de sélection de caractéristiques
  - Sélection séquentielle vorace avant
  - Sélection séquentielle vorace arrière
  - Algorithmes évolutionnaires multiobjectifs

11.7 Ensembles dans scikit-learn

Scikit-learn

- `ensemble.BaggingClassifier` : plusieurs variantes de *Bagging* de classifieurs, incluant *random subspaces*
- `ensemble.RandomForestClassifier` : forêt aléatoire pour le classement
- `ensemble.AdaBoostClassifier` : variantes AdaBoost.SAMME de l'algorithme AdaBoost
- `ensemble.VotingClassifier` : vote de classifieurs, incluant vote à majorité et somme pondérées des probabilités
- `multiclass.OutputCodeClassifier` : combinaison de classifieurs avec un code pour la décision, pouvant être un code à correction d'erreur