INF7370

Apprentissage d'ensemble et forêts aléatoires

bouguessa.mohamed@uqam.ca

Plan

1 – Apprentissage par combinaison de classificateurs

2 – Bagging

3 – Boosting

4 – Forêt d'arbres décisionnels (Random Forest)

Apprentissage par combinaison de classificateurs

❑ Idée générale

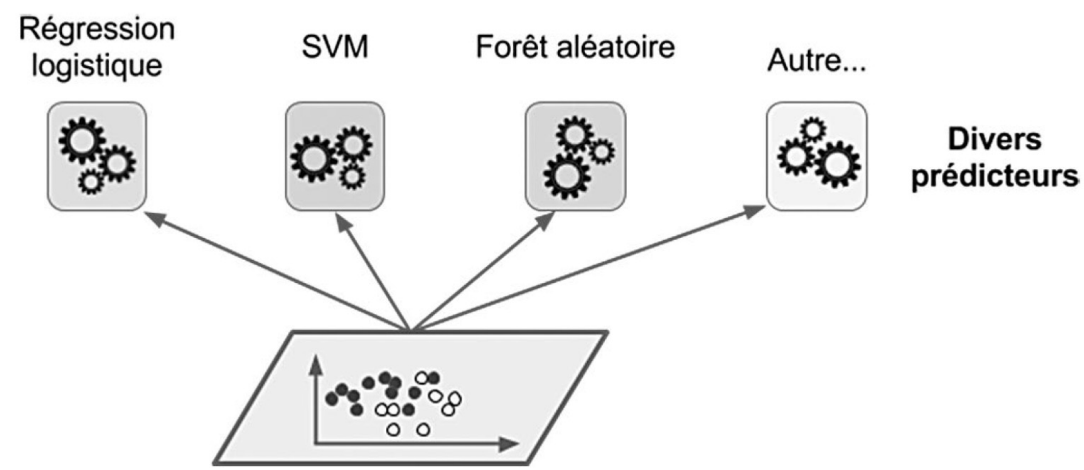
Réunir un « comité d'experts »:
chacun peut se tromper, mais en combinant les avis,
on a plus de chance d'avoir la bonne prédiction !…
→ Plusieurs experts valent mieux qu'un

- Un expert peut être bien un algorithme d'apprentissage.
- En apprentissage artificiel, il est possible d'atteindre une décision aussi précise que souhaité par une combinaison judicieuse d'algorithmes correctement entraînés.

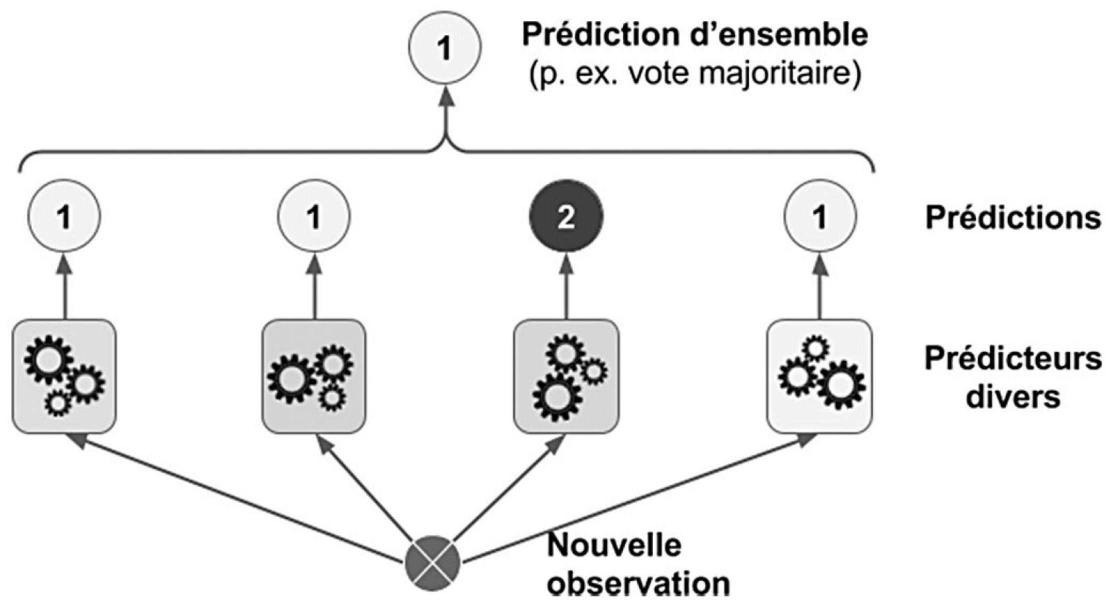
→ La décision finale est prise à partir du vote majoritaire des résultats des différents algorithmes utilisés.

Apprentissage par combinaison de classificateurs

Entraînement de différents classificateurs



Classification par vote



Figures tirées de : Aurélien Géron. Machine Learning avec Scikit-Learn : Mise en œuvre et cas concrets. Dunod, 2019.

Apprentissage par combinaison de classificateurs

❑ Formulation

• Supposons que nous avons N classificateurs, faisant chacun une erreur $R_{Emp} = \varepsilon$.

• Si on décide « à la majorité », alors on se trompe si et seulement si plus de la moitié du « comité » se trompe.

• Le taux d'erreur de l'ensemble des N classificateurs est donc:

$$E_{ensemble} = \sum_{i=\frac{N}{2}}^{N} C_N^i \, \varepsilon^i \, (1-\varepsilon)^{N-i}$$

$$C_N^i = \binom{N}{i} = \frac{N!}{i!\,(N-i)!}$$

Apprentissage par combinaison de classificateurs

❑ Exemple

• Supposons que nous avons 25 classificateurs, faisant chacun une erreur $\varepsilon = 0.35$. Un apprentissage par combinaison de classificateurs permet de prédire la classe d'un objet de l'échantillon de test par vote majoritaire (sélection de la classe majoritaire à partir des résultats des 25 classificateurs).

• Généralement, deux situations se présentent

1. Si les 25 classificateurs fonctionnent de façon identique, il est clair que l'erreur de l'ensemble reste la même, c.-à-d. 0.35.

Apprentissage par combinaison de classificateurs

❑ Exemple (suite)

2. Si les 25 classificateurs sont indépendants, l'erreur est donc

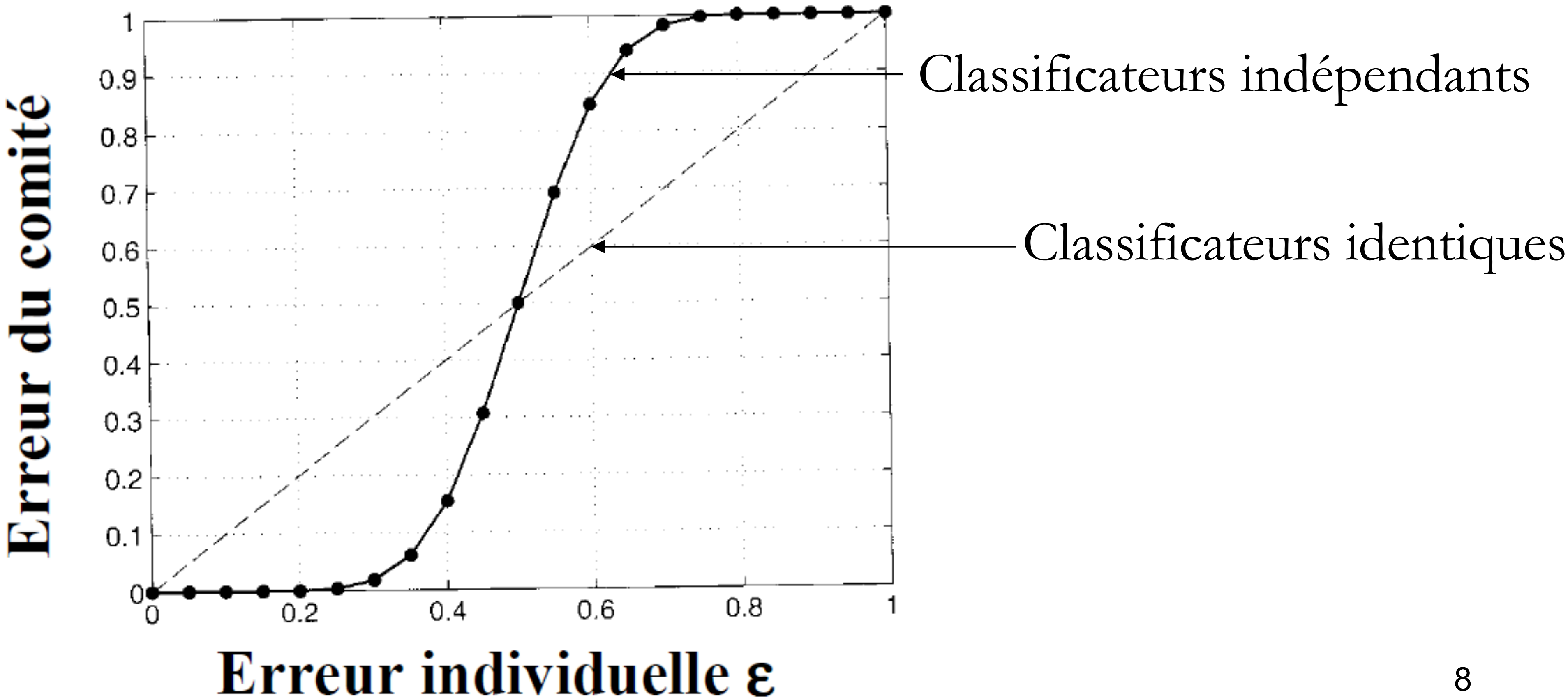
$$E_{ensemble} = \sum_{i=13}^{25} C_i^{25} \times 0.35^i (1 - 0.35)^{25-i} = 0.06$$

qui est considérablement plus petite que le taux d'erreur de chaque classificateur.

→Décision fortement améliorée (sous réserve que ε <0.5)…

Apprentissage par combinaison de classificateurs

La figure suivante montre le taux d'erreur pour 25 classificateurs combinés ($E_{ensemble}$) pour différents taux d'erreur ε de chaque classificateur individuelle.



Apprentissage par combinaison de classificateurs

À partir du graphique précédent, on observe bien que le taux d'erreur de l'ensemble de classificateurs ($E_{ensemble}$) s'améliore seulement lorsque $\varepsilon < 0,5$.

- Il y a donc deux conditions pour utiliser une combinaison de classificateurs :

1. Les classificateurs doivent être indépendants;
2. $\varepsilon < 0.5$.

Méthodes de construction d'ensembles

• Manipulation des données d'apprentissage : entrainer N classificateurs sur différents ensembles d'apprentissage. Deux approches sont généralement utilisées : **Bagging** et **Boosting**.

• Manipulation des caractéristiques (attributs) : choisir un sous-ensemble d'attributs pour former un ensemble d'apprentissage. Les sous-ensembles d'attributs peuvent être choisis aléatoirement ou bien sous la recommandation d'un expert. Dans ce contexte, l'approche qui est généralement utilisée : **Random forest**.

Plan

1 – Apprentissage par combinaison de classificateurs

2 – Bagging

3 – Boosting

4 – Forêt d'arbres décisionnels (Random Forest)

Bagging

- Bagging est une contraction de bootstrap aggregating.
- L'idée de base est d'entrainer un algorithme d'apprentissage sur plusieurs échantillons d'apprentissage obtenus par tirage aléatoire avec remise à partir de l'ensemble d'apprentissage original.
- Chaque échantillon obtenu est appelé « bootstrap » (un tirage avec remise s'appelle bootstrapping en anglais).
- Généralement, un bootstrap a la même taille que l'ensemble d'apprentissage original.
- Comme la construction des échantillons bootstrap est faite de manière aléatoire avec remise, il est possible de trouver, dans un même bootstrap, que certains objets peuvent apparaître plusieurs fois alors que d'autres objets, qui existent toujours dans l'ensemble d'apprentissage original, sont omis.

Bagging

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- Chaque objet à une probabilité de $1 - (1 - 1/n)^n$ d'être sélectionné.

Bagging

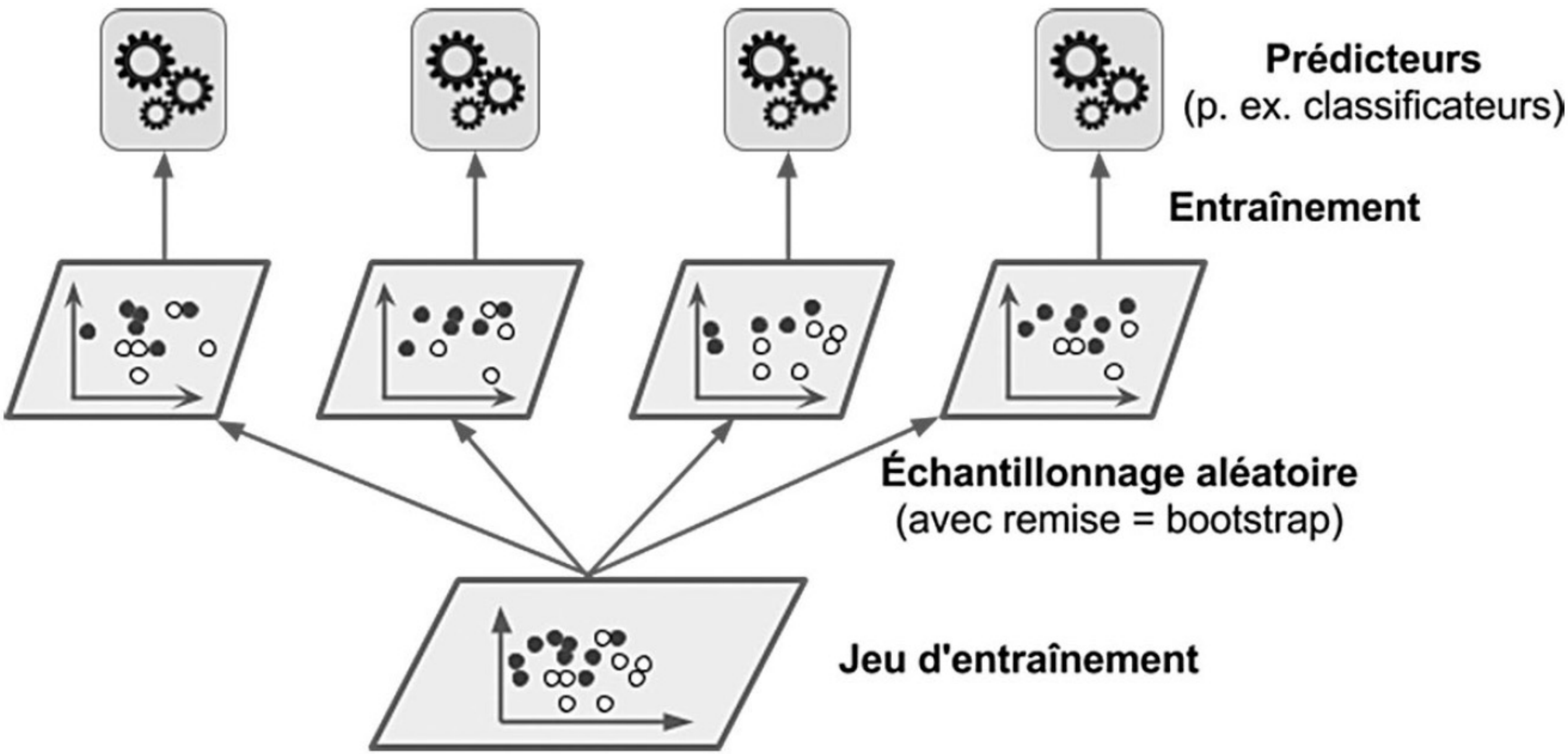


Figure tirée de : Aurélien Géron. Machine Learning avec Scikit-Learn : Mise en œuvre et cas concrets. Dunod, 2019.

L'algorithme du Bagging

1. Soit k le nombre d'échantillons bootstrap et N la taille de l'ensemble d'apprentissage T.

2. pour $i = 1, \dots, k$ **faire**

   2.1 Créer un échantillon bootstrap Di de taille N.

   2.2 Entrainer un classificateur Ci sur l'ensemble Di.

3. fin pour

4. pour chaque objet x de l'ensemble T **faire**

   4.1 $C^*(x)$ = Vote-majoritaire$(C1(x), C2(x), \dots, Ck(x))$

5. fin pour

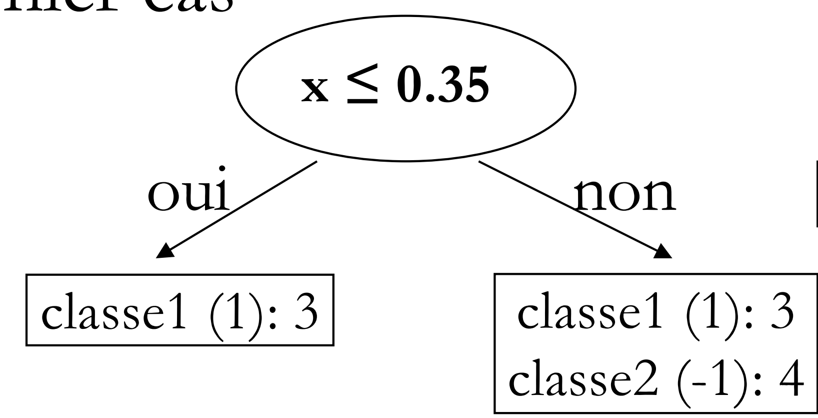
Illustration du Bagging

- Soit l'ensemble de données suivant :

données	x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
classe	y	1	1	1	-1	-1	-1	-1	1	1	1

- On suppose que notre algorithme d'apprentissage est un arbre de décision binaire à seul niveau. Dans ce contexte, deux cas sont possibles:

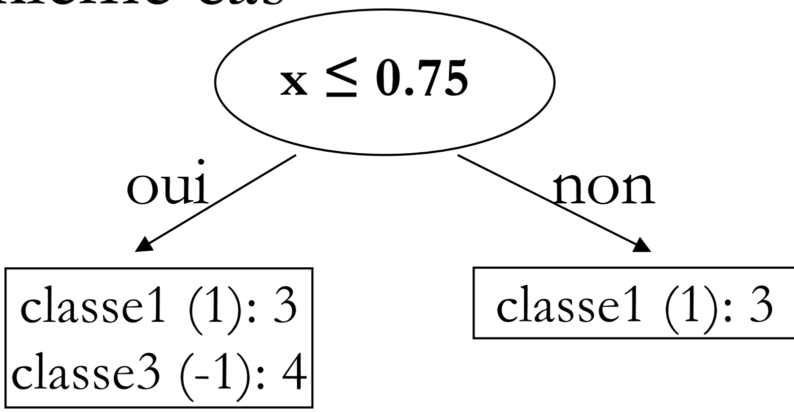
1. Premier cas



⇒ Précision de la classification = 70%

Illustration du Bagging

2. Deuxième cas



Précision de la classification = 70%

• Les tableaux présentés dans l'acétate suivant illustrent le processus du Bagging.

• Prendre note que la partie de droite de chaque table montre la décision prise par un classificateur pour séparer les deux classes.

Exemple

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

x <= 0.35 ➔ y = 1
x > 0.35 ➔ y = -1

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.5	0.9	1	1	1
y	1	1	1	-1	-1	-1	1	1	1	1

x <= 0.7 ➔ y = 1
x > 0.7 ➔ y = 1

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.35 ➔ y = 1
x > 0.35 ➔ y = -1

Bagging Round 4:

x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.3 ➔ y = 1
x > 0.3 ➔ y = -1

Bagging Round 5:

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1
y	1	1	1	-1	-1	-1	-1	1	1	1

x <= 0.35 ➔ y = 1
x > 0.35 ➔ y = -1

*Source : Tan et. Al “Introduction to Data Mining”, 2nd edition, Pearson, 2018.

Exemple (suite)

Bagging Round 6:

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1
y	1	-1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 7:

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1
y	1	-1	-1	-1	-1	1	1	1	1	1

x <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 8:

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 9:

x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 10:

x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9
y	1	1	1	1	1	1	1	1	1	1

x <= 0.05 ➔ y = 1
x > 0.05 ➔ y = 1

Exemple (suite)

Résumé des décisions :

Round	Split Point	Left Class	Right Class
1	0.35	1	-1
2	0.7	1	1
3	0.35	1	-1
4	0.3	1	-1
5	0.35	1	-1
6	0.75	-1	1
7	0.75	-1	1
8	0.75	-1	1
9	0.75	-1	1
10	0.05	1	1

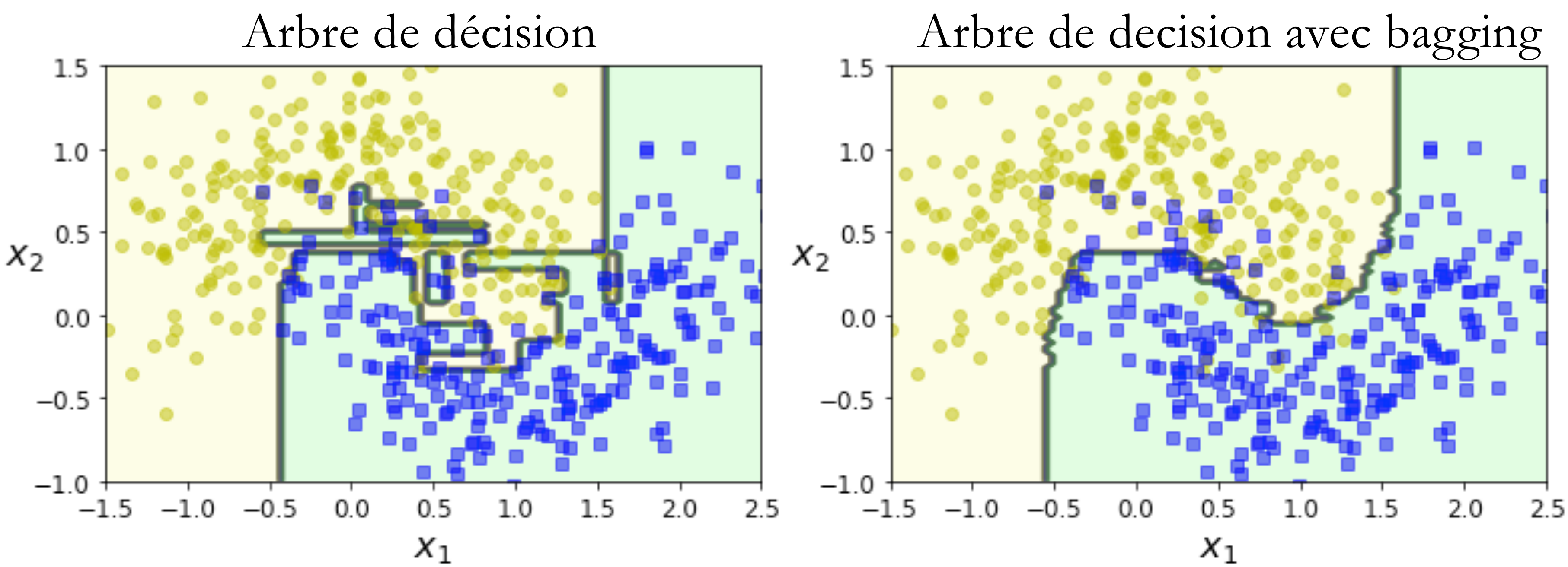
Exemple (suite et fin)

Le tableau suivant montre comment combiner la prédiction des 10 classificateurs à l'aide de la méthode de Bagging. Dans ce tableau, tous les exemples sont parfaitement classés.

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
4	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1	1	1	1
7	-1	-1	-1	-1	-1	-1	-1	1	1	1
8	-1	-1	-1	-1	-1	-1	-1	1	1	1
9	-1	-1	-1	-1	-1	-1	-1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
Sum	2	2	2	-6	-6	-6	-6	2	2	2
Sign	1	1	1	-1	-1	-1	-1	1	1	1

Illustration

Comparaison entre un arbre de décision unique (à gauche)
et un ensemble de type bagging de 500 arbres (à droite)



Figures tirées de : Aurélien Géron. Machine Learning avec Scikit-Learn : Mise en œuvre et cas concrets. Dunod, 2019.

Bagging : conclusion

- La méthode du Bagging tente de réduire l'erreur de généralisation.

- La méthode du Bagging ne focalise pas sur des objets particuliers dans l'ensemble d'apprentissage, car chaque objet à la même probabilité d'être sélectionné dans l'échantillon bootstrap. Cette méthode souffre donc du problème de l'overfitting si les données d'apprentissage contiennent des objets mal étiquetés (bruit).

Plan

1 – Apprentissage par combinaison de classificateurs

2 – Bagging

3 – Boosting

4 – Forêt d'arbres décisionnels (Random Forest)

Boosting : Principe

- Une question de Kearns : « Est-il possible de rendre aussi bon que l'on veut un algorithme d'apprentissage faible » (c'est-à-dire un peu meilleur que le hasard) ?
- La réponse de Schapire : « Oui ! », et le premier algorithme élémentaire de boosting, qui montre qu'un algorithme de classification binaire faible peut toujours améliorer sa performance en étant entraîné sur trois échantillons d'apprentissage bien choisis.

→ L'idée est d'utiliser un algorithme d'apprentissage sur trois sous-ensembles d'apprentissage.

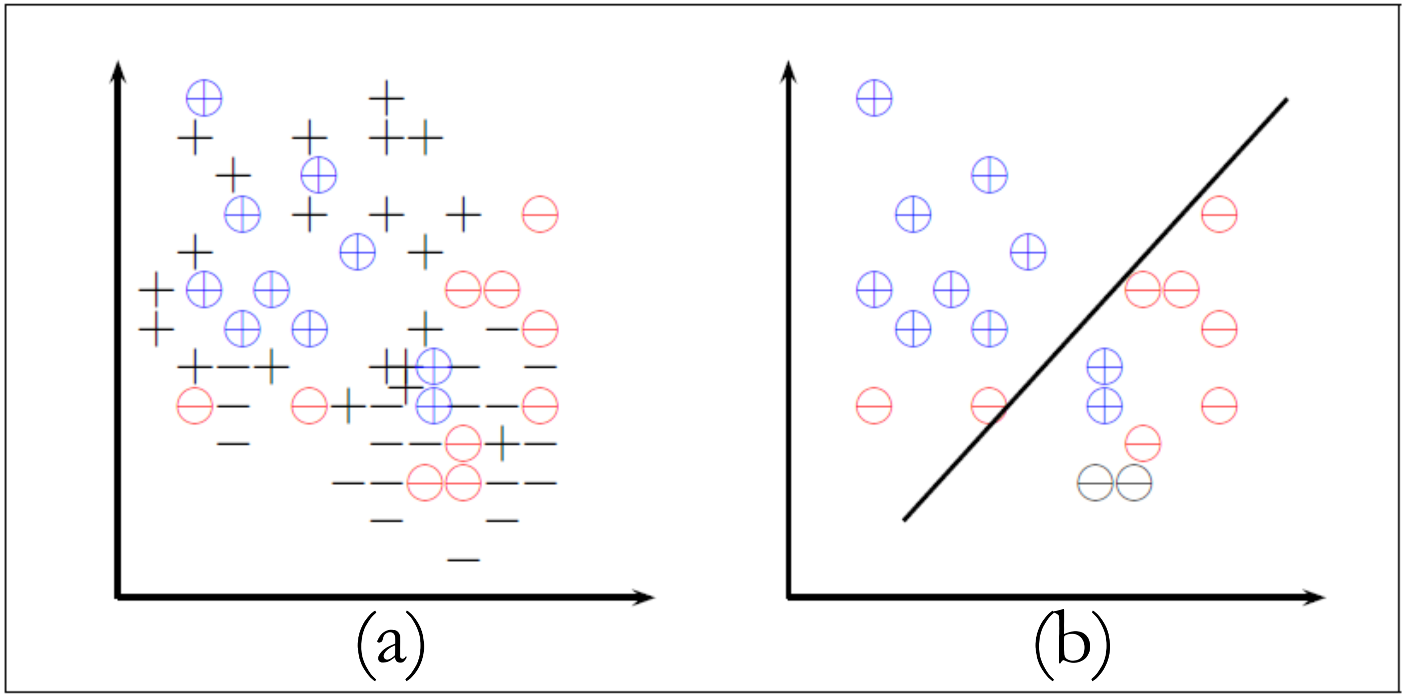
Le Boosting élémentaire

1. On obtient d'abord une première hypothèse h_1 sur un sous-échantillon $S1$ d'apprentissage de taille $n_1 < n$ (n étant la taille de S l'échantillon d'apprentissage disponible).
2. On apprend alors une deuxième hypothèse h_2 sur un échantillon $S2$ de taille n_2 choisi dans $S - S1$ dont la moitié des exemples sont mal classés par h_1.
3. On apprend finalement une troisième hypothèse h_3 sur n_3 exemples tirés dans $S - S1 - S2$
4. L'hypothèse finale est obtenue par un vote majoritaire des trois hypothèses apprises : H = vote majoritaire(h_1, h_2, h_3)

Le Boosting élémentaire

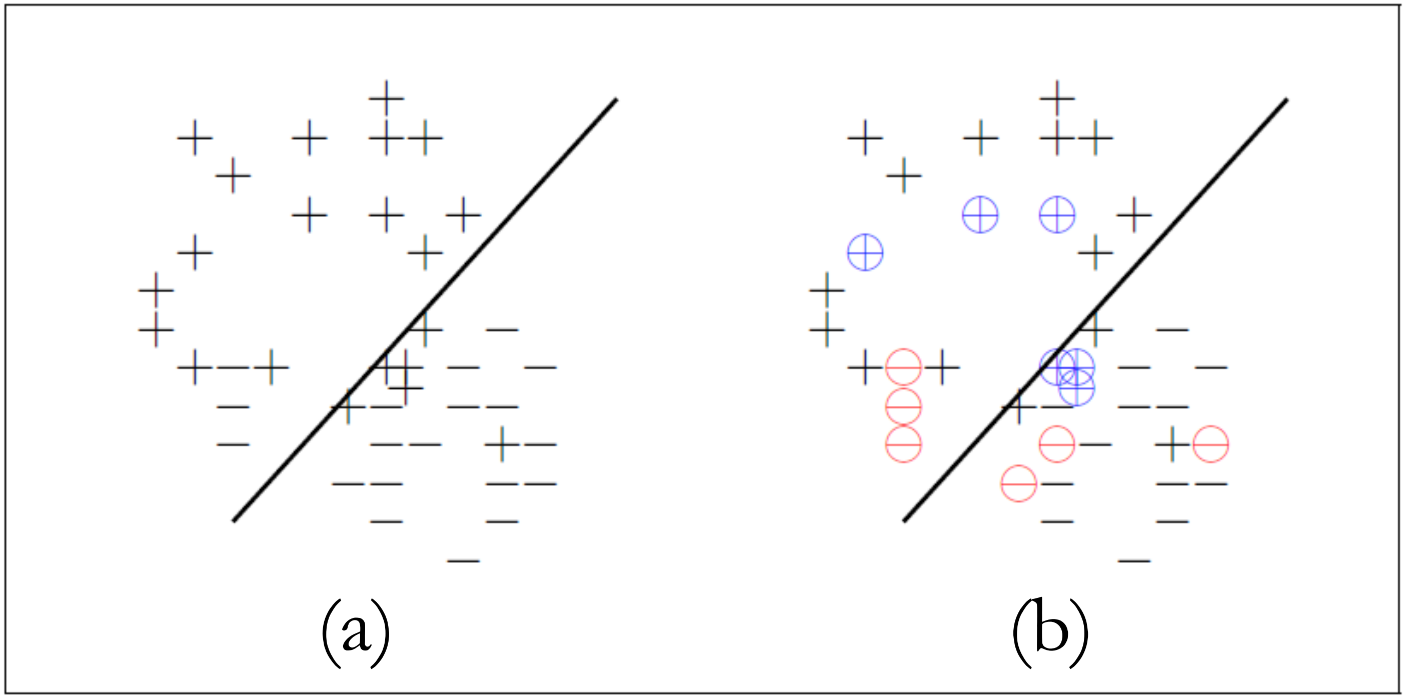
- Schapire à démontré que H a une performance supérieure à celle de l'hypothèse qui aurait été apprise directement sur l'échantillon *S*.
- Une illustration graphique du boosting selon cette technique de base est présentée dans les figures suivantes.
- Le classificateur de base est un hyperplan.

Début



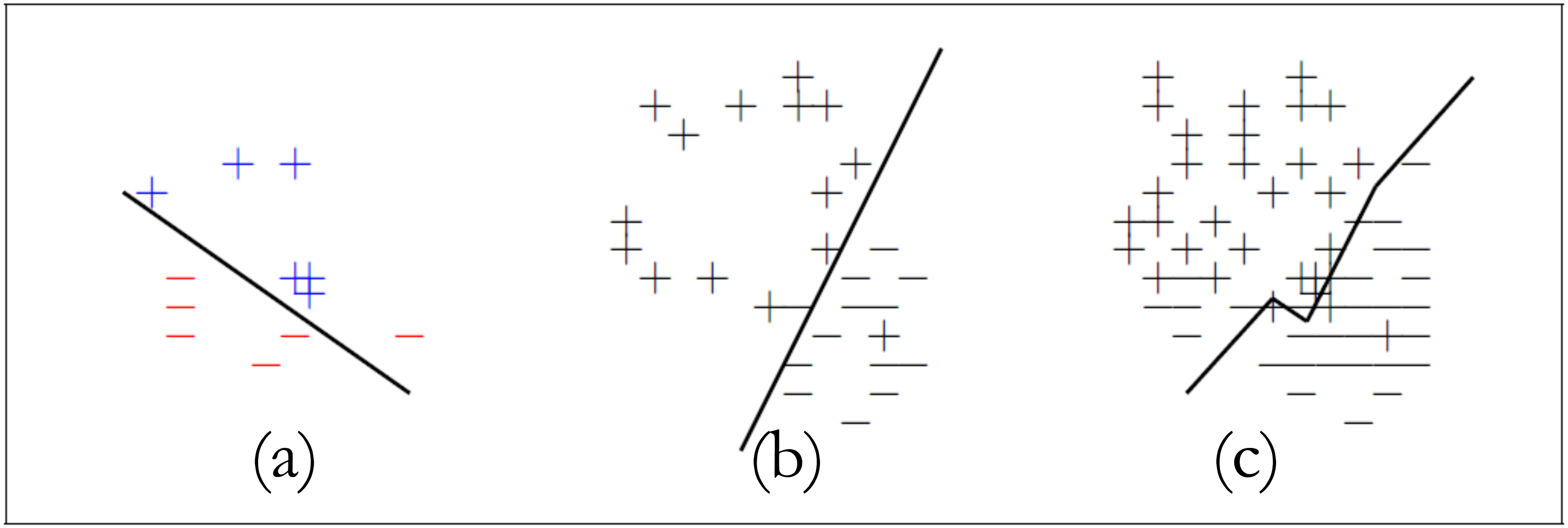
- (a) : l'ensemble d'apprentissage *S* et le sous-ensemble *S1* (points entourés).
- (b) : l'ensemble *S1* et l'hypothèse h_1 apprise sur cet ensemble.

Suite

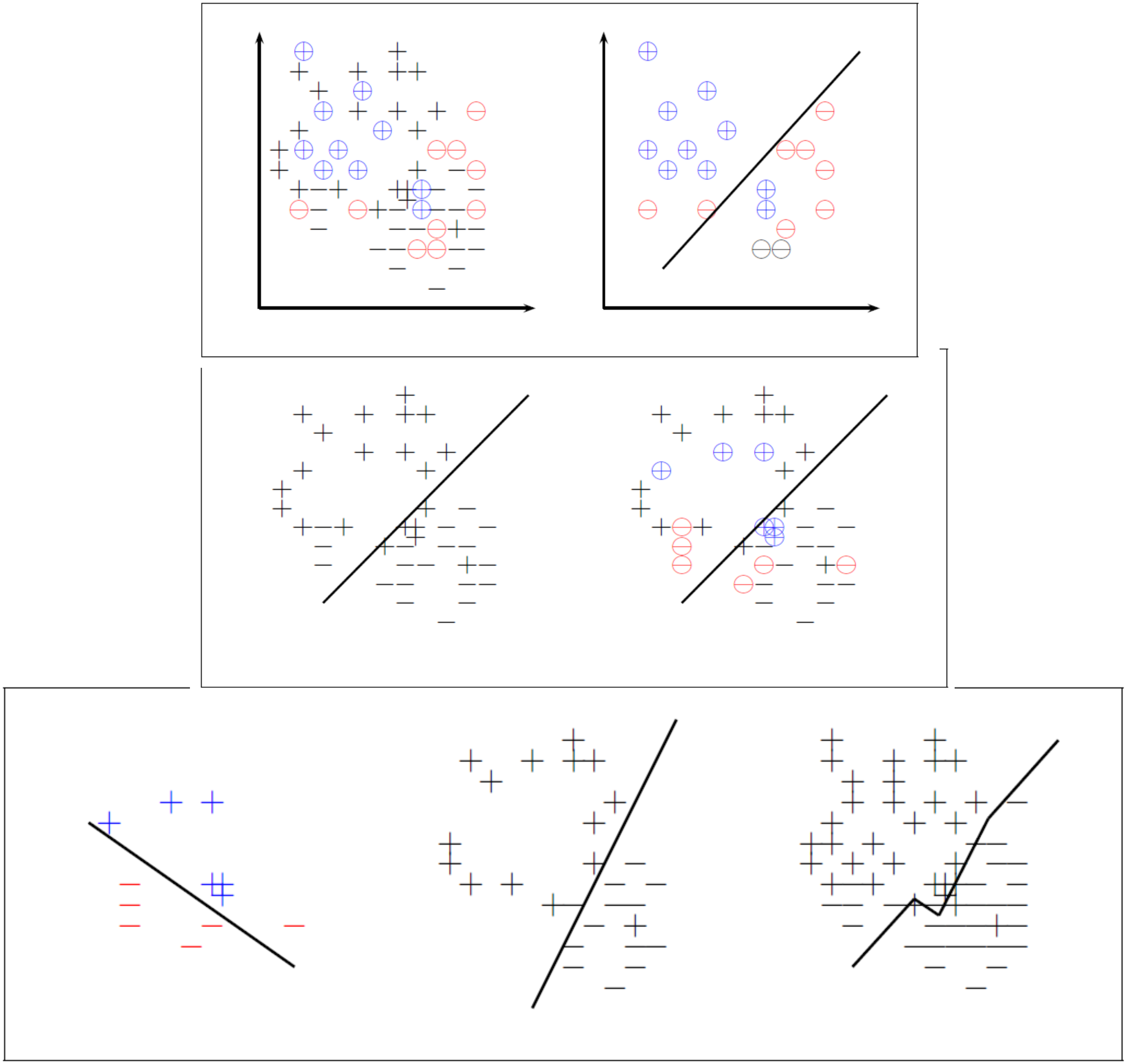


- (a) : l'ensemble $S - S1$ et la droite h_1 apprise sur $S1$.
- (b) : les points entourés représentent l' échantillon $S2$ ($S2$ est choisi à partir de $S - S1$). On observe que la moitié des points de $S2$ sont mal classés par h_1.

… et fin.



- (a) : l'ensemble *S2* et la droite séparatrice h_2 apprise sur cet ensemble.
- (b) : l'ensemble $S3 = S - S1 - S2$ et la droite séparatrice h_3 apprise sur cet ensemble.
- (c) : l'ensemble *S* et la combinaison de h_1, h_2 et h_3.



Boosting probabiliste

Trois idées pour généraliser vers le boosting probabiliste :

1. L'utilisation d'un comité d'experts que l'on fait voter pour atteindre une décision.
2. La pondération adaptative des votes par une technique de mise à jour multiplicative.
3. La modification de la distribution des exemples disponibles pour entraîner chaque expert, en surpondérant au fur et à mesure les exemples mal classés aux étapes précédentes.

La technique de base

Paramètre : Un apprenant faible h

Entrée : Un échantillon S

Initialisation : Tous les exemples ont le même poids (ex. poids = $1/n$ avec n la taille de S)

pour $i = 1, \ldots, k$ **faire** (k le nombre de round)

- Tirer un échantillon d'apprentissage S_i de S selon les poids des exemples.
- Appliquer h sur S_i.
- Renforcer le poids des exemples mal classés et diminuer le poids des exemples classés correctement.

fin pour

Sortie H : un vote majoritaire pondéré des hypothèses

Boosting probabiliste

- L'idée de base consiste donc à affecter des poids pour chaque exemple dans l'ensemble d'apprentissage.
- Les exemples mal classés vont avoir leur poids augmenté.
- Les exemples correctement classés vont avoir leur poids diminué.

→ Le classificateur se concentre donc sur les données mal classées dans chaque étape du boosting.

❑ Exemple

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

L'exemple 4 est difficile a classé → il est plus probable il sera choisi dans les prochaines étapes du boosting

AdaBoost (adaptive boosting)

- L'idée principale est de définir à chacune de ses étapes i : $1 \leq i \leq k$, une nouvelle distribution de probabilité *a priori* sur les exemples d'apprentissages en fonction des résultats de l'algorithme à l'étape précédente.

- Le poids à l'étape i d'un exemple x_j est noté w_i

- Initialement, tous les exemples ont un poids identique, puis à chaque étape, les poids des exemples mal classés par l'apprenant sont augmentés.

AdaBoost

Entraînement séquentiel AdaBoost avec modification des poids des observations.

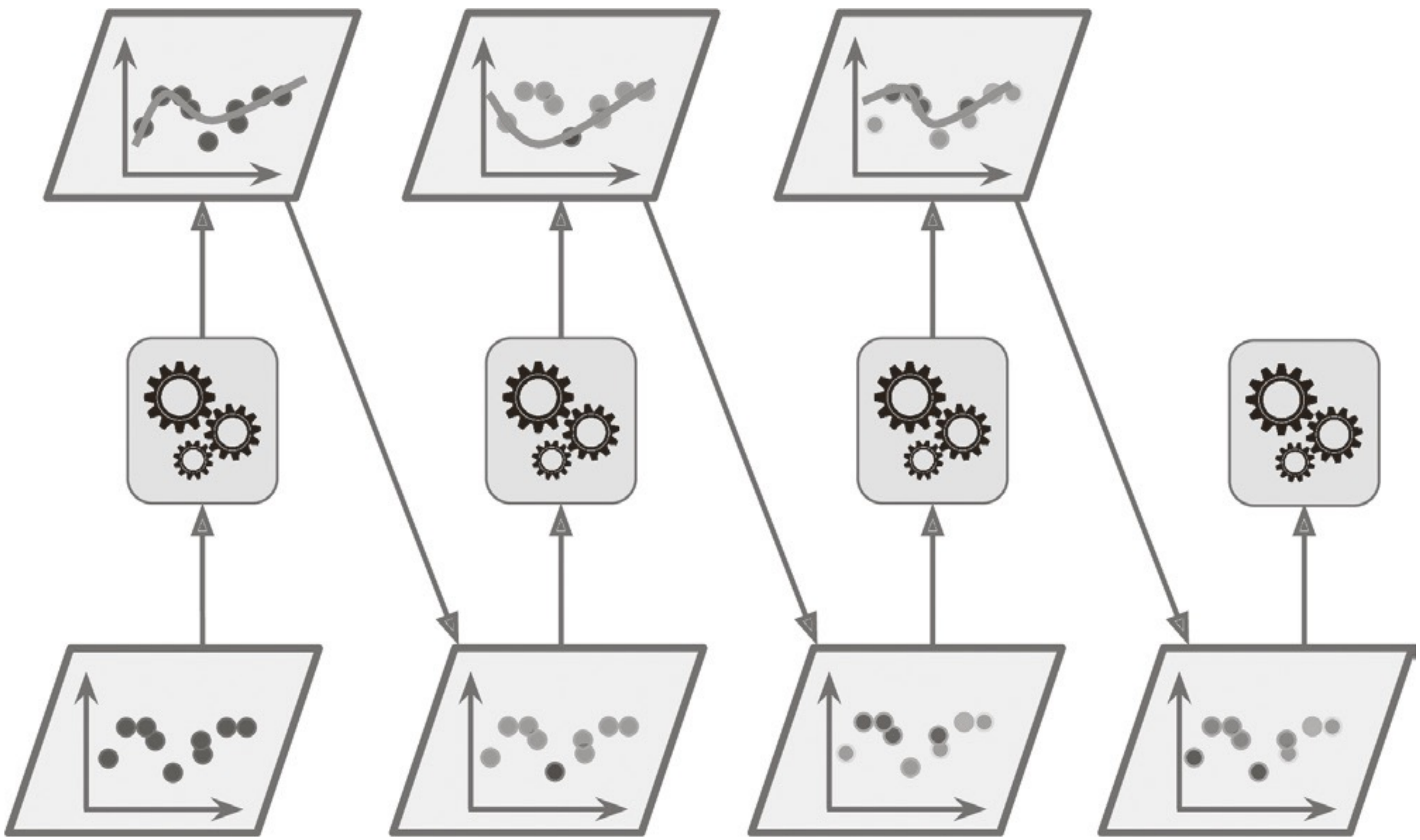
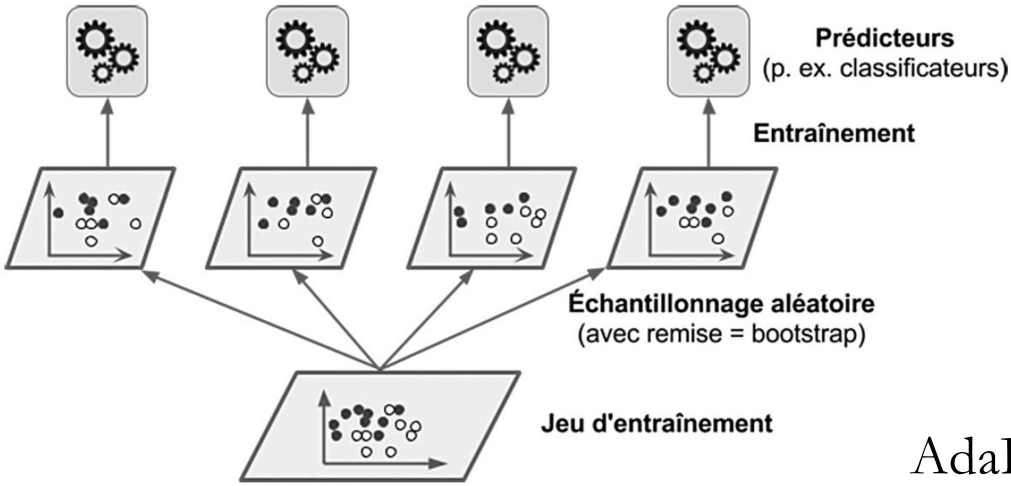


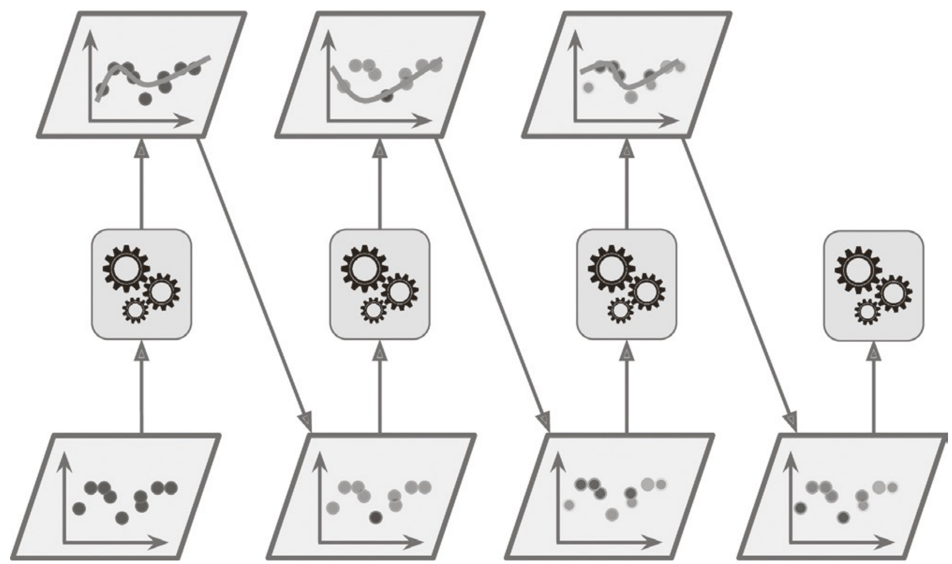
Figure tirée de : Aurélien Géron. Machine Learning avec Scikit-Learn : Mise en œuvre et cas concrets. Dunod, 2019.

Bagging vs AdaBoost

Bagging.



AdaBoost.



Figures tirées de : Aurélien Géron. Machine Learning avec Scikit-Learn : Mise en œuvre et cas concrets. Dunod, 2019.

AdaBoost (adaptive boosting)

- Soit $\{(x_j, y_j) \mid j = 1, \ldots, n\}$ un ensemble de n exemples
- À chaque i ($1 \leq i \leq k$) de AdaBoost, la performance de l'apprenant h_i dépend des poids w_i des exemples x_j sur lesquels h_i est entraîné. L'erreur de h_i est définie comme :

$$\varepsilon_i = \frac{1}{n}\left[\sum_{j=1}^{n} w_j \times I(h_i(x_j) \neq y_j)\right]$$

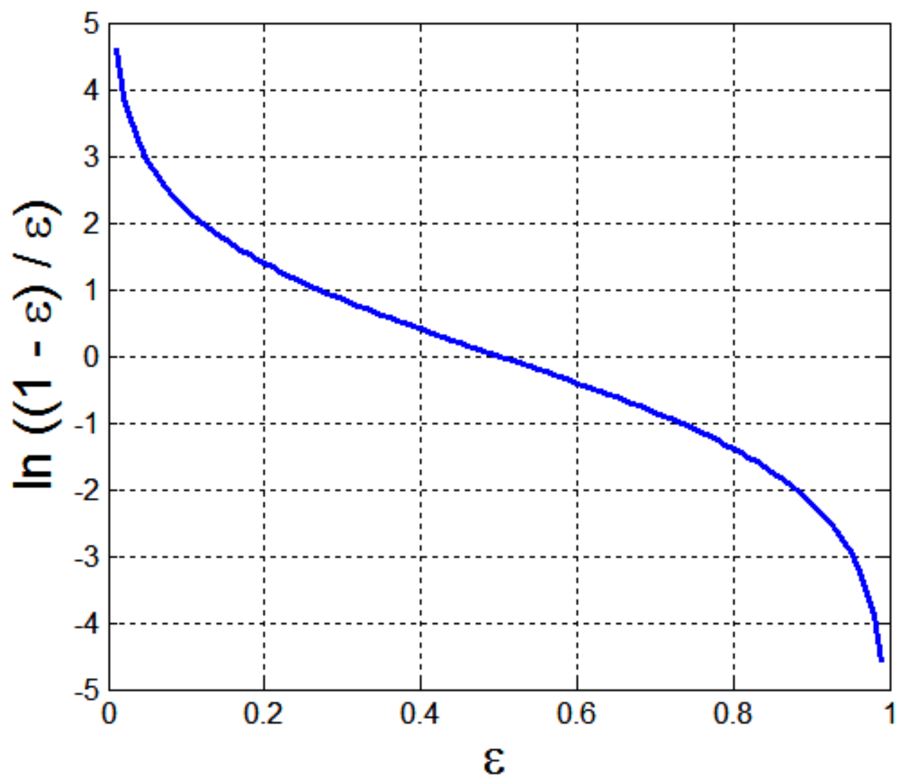
I(p) = 1 si p est vrai, sinon I(p) = 0.

AdaBoost

- L'importance de h_i est déterminée par le paramètre suivant

$$\alpha_i = \frac{1}{2}\ln\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$

- **si** $\varepsilon_i \leq 0.5$ alors $\alpha_i > 0$ **sinon** $\alpha i < 0$



AdaBoost

- α_i est utilisé pour le calcul des mises à jour des poids w_j de chaque exemple x_j pour l'étape $i+1$ du boosting.

$$w_j^{(i+1)} = \frac{w_j^i}{Z_i} \times \begin{cases} \exp^{-\alpha_i} \; si\ h_i(x_j) = y_j \longleftarrow x_j \text{ bien classé par } h_i \\ \exp^{\alpha_i} \; si\ h_i(x_j) \neq y_j \longleftarrow x_j \text{ mal classé par } h_i \end{cases}$$

Z_i est un facteur de normalisation utilisé pour assurer que

$$\sum_{j=1}^{n} w_j^{(i+1)} = 1$$

Avec la formule d'estimation des w_j, les éléments mal classés vont avoir un poids élevé alors que les éléments correctement classés vont avoir un poids faible.

AdaBoost

1. $w = \{w_j = 1/n \mid j = 1, \ldots, n\}$ //initialisation des poids
2. **pour** $i = 1, \ldots, k$ **faire** //k le nombre de round
    2.1. Tirer un échantillon d'apprentissage S_i de S selon les poids w.
    2.2. Apprendre une règle de classification h_i sur S_i par un algorithme d'apprentissage A.
    2.3. Calculer l'erreur ε_i.
    2.4. Calculer le paramètre α_i.
    2.5. **pour** j = 1, …, n **faire** Calculer $w_j^{(i+1)}$ **fin pour**
3. **fin pour**
4. Fournir en sortie l'hypothèse finale : $H(x) = sign\left(\sum_{i=1}^{k} \alpha_i h_i(x)\right)$

AdaBoost

À la fin de cet algorithme, chaque hypothèse h_i est pondérée par une valeur α_i calculée en cours de route. La classification d'un nouvel exemple (ou des points de S pour obtenir l'erreur apparente) se fait en utilisant la règle :

$$H(x) = sign\left(\sum_{i=1}^{k} \alpha_i h_i(x)\right)$$

→ le boosting construit l'hypothèse finale comme une série additive dans une base de fonctions, dont les éléments sont les hypothèses h_i.

Exemple

- Soit l'ensemble S suivant :

	x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
données →	x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
classe →	y	1	1	1	-1	-1	-1	-1	1	1	1

- Les tableaux suivants montrent les échantillons S_i générés durant 3 rounds de boosting

Boosting Round 1:

x	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1
y	1	-1	-1	-1	-1	-1	-1	-1	1	1

Boosting Round 2:

x	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
y	1	1	1	1	1	1	1	1	1	1

Boosting Round 3:

x	0.2	0.2	0.4	0.4	0.4	0.4	0.5	0.6	0.6	0.7
y	1	1	-1	-1	-1	-1	-1	-1	-1	-1

*Source : Tan et. Al "Introduction to Data Mining", 2nd edition, Pearson, 2018.

Exemple (suite)

- Les poids estimés dans chaque étape de boosting :

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2	0.311	0.311	0.311	0.01	0.01	0.01	0.01	0.01	0.01	0.01
3	0.029	0.029	0.029	0.228	0.228	0.228	0.228	0.009	0.009	0.009

Boosting Round 1:

x	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1
y	1	-1	-1	-1	-1	-1	-1	-1	1	1

Boosting Round 2:

x	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
y	1	1	1	1	1	1	1	1	1	1

Boosting Round 3:

x	0.2	0.2	0.4	0.4	0.4	0.4	0.5	0.6	0.6	0.7
y	1	1	-1	-1	-1	-1	-1	-1	-1	-1

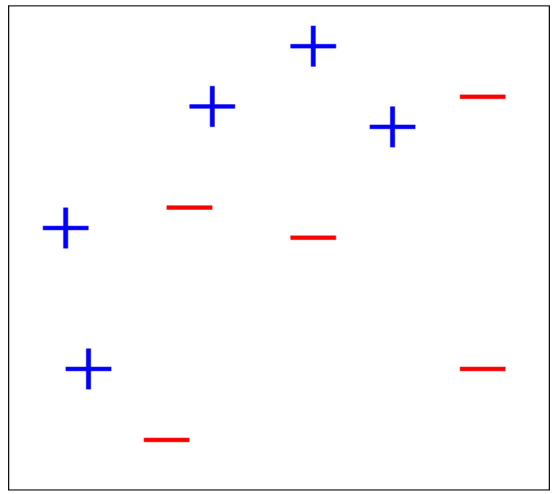
Exemple (suite et fin)

Round	Split Point	Left Class	Right Class	alpha
1	0.75	-1	1	1.738
2	0.05	1	1	2.7784
3	0.3	1	-1	4.1195

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	-1	-1	-1	-1	-1	-1	-1	1	1	1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
Sum	5.16	5.16	5.16	-3.08	-3.08	-3.08	-3.08	0.397	0.397	0.397
Sign	1	1	1	-1	-1	-1	-1	1	1	1

→ À la fin de l'algorithme AdaBoost, tous les exemples sont correctement classés.

Illustration graphique de AdaBoost



- Étape 1

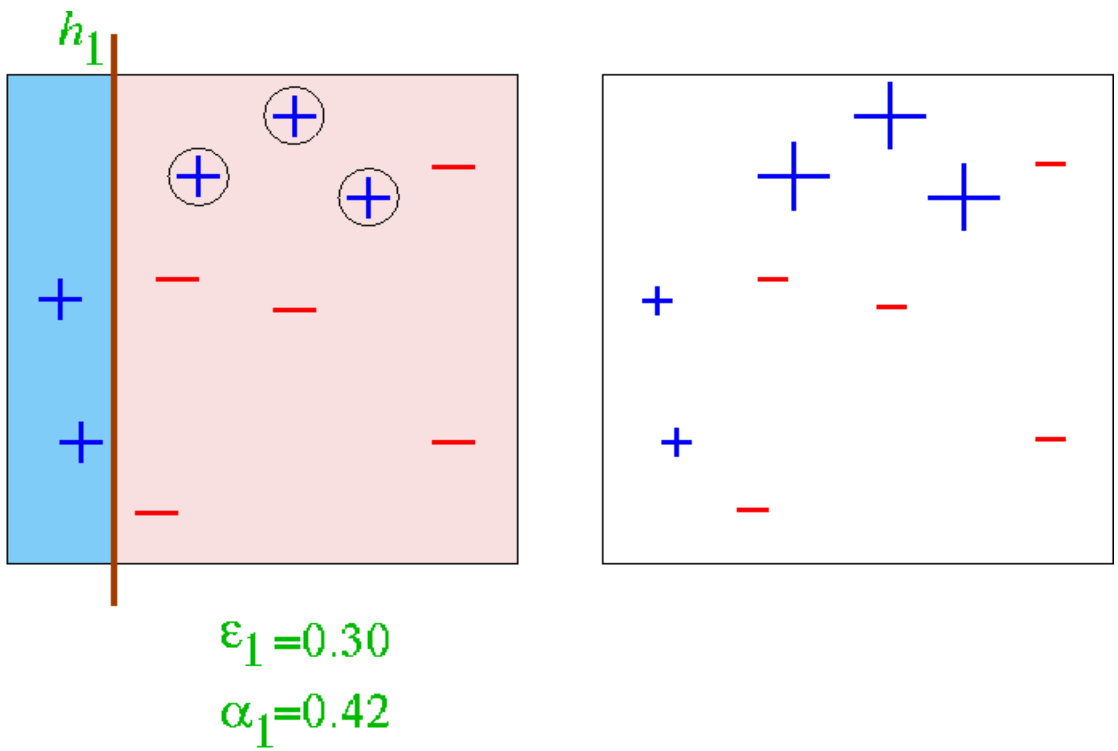


Illustration graphique de AdaBoost

- Étape 2

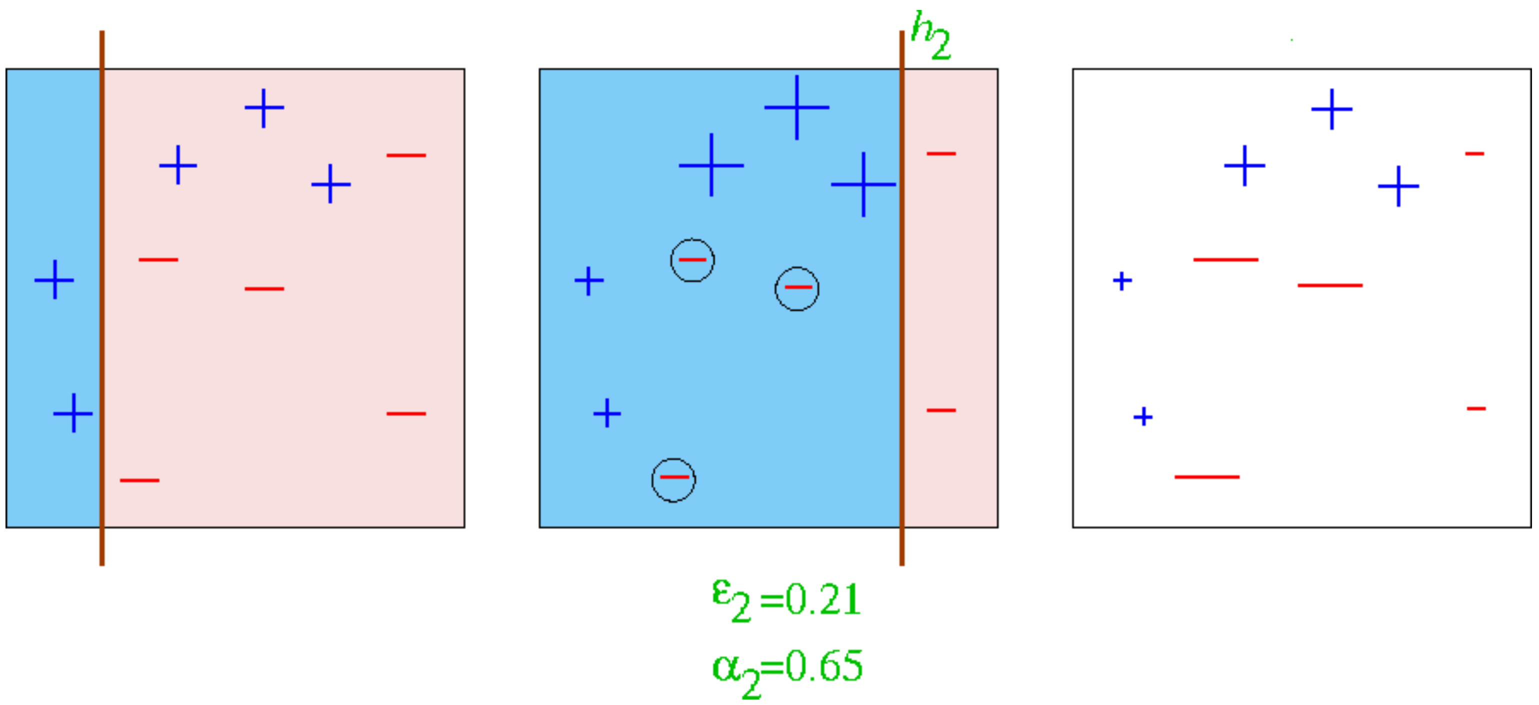


Illustration graphique de AdaBoost

- Étape 3

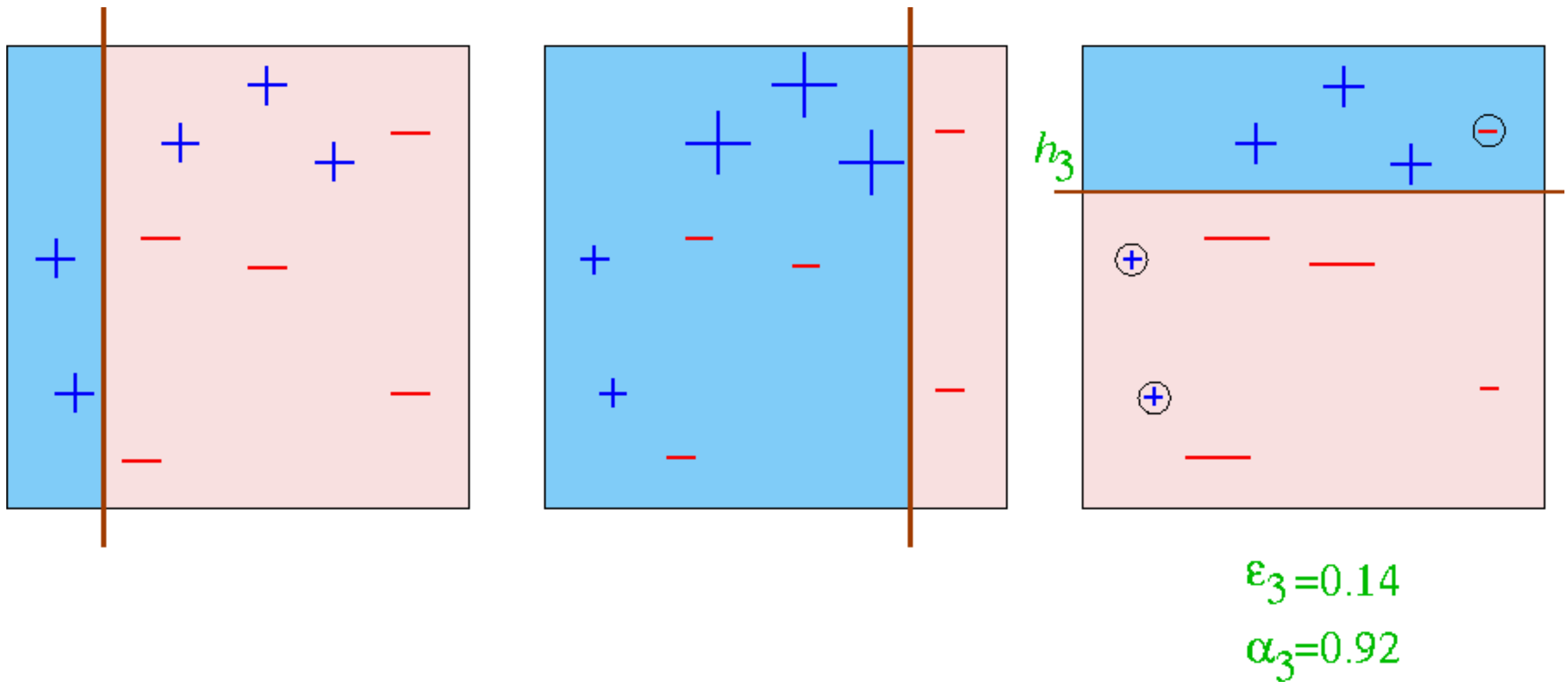
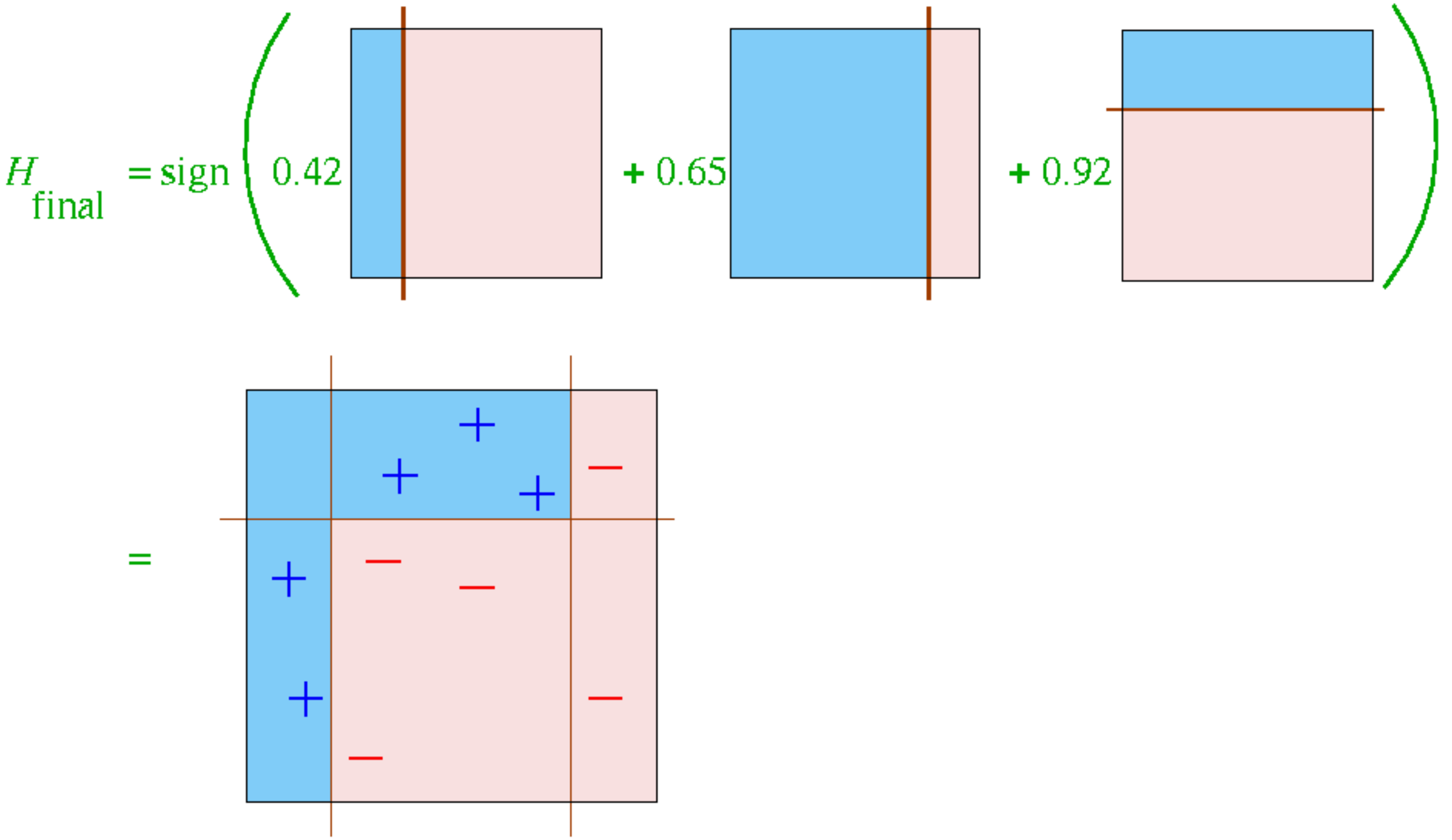


Illustration graphique de AdaBoost

❑ Hypothèse finale



Boosting : résumé

- La prédiction finale est issue d'une combinaison (vote pondéré) de plusieurs prédictions
- **Méthode** :
  - Itérative
  - Chaque classificateur dépend des précédents (les classificateurs ne sont donc pas indépendants comme dans d'autres méthodes de vote)
  - Les exemples sont pondérés différemment
  - Le poids des exemples reflète la difficulté des classificateurs précédents à les apprendre

Plan

1 – Apprentissage par combinaison de classificateurs

2 – Bagging

3 – Boosting

4 – Forêt d'arbres décisionnels (Random Forest)

Random Forest

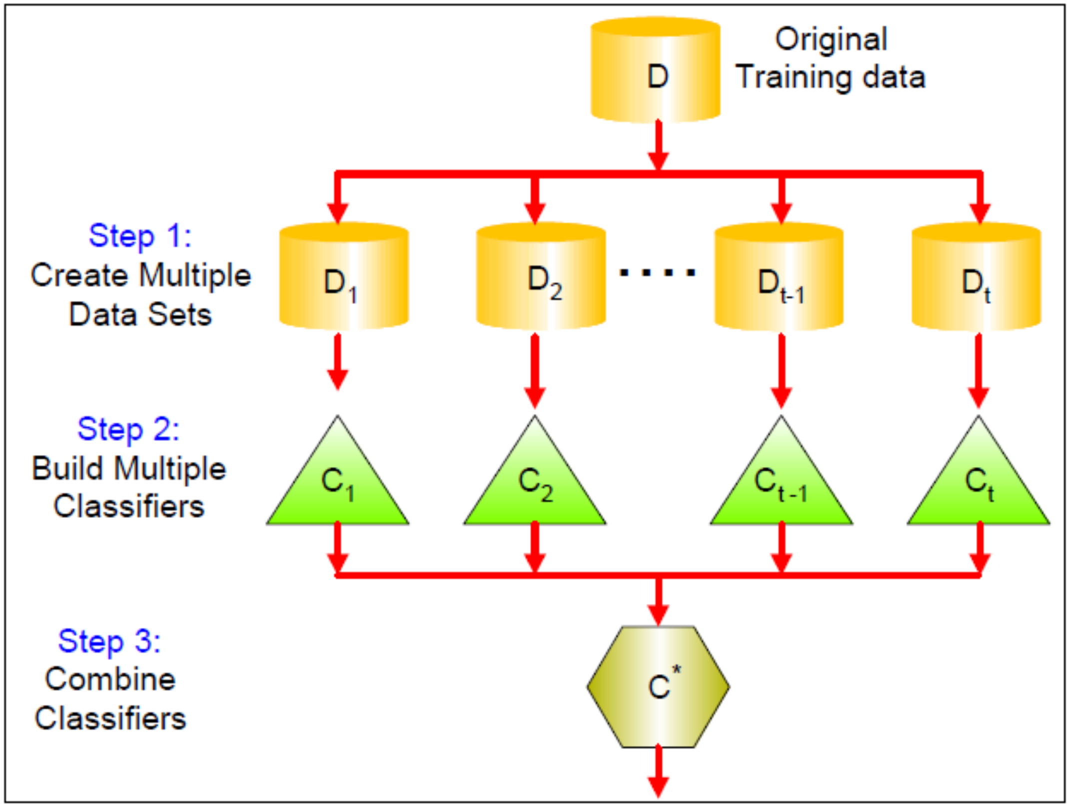
- Random Forest combine la sélection aléatoire d'enregistrements avec la sélection aléatoire des attributs.
- L'algorithme de base : les arbres de décision.
- La forêt = un ensemble d'arbres de décision.

→ Apprentissage d'un ensemble d'arbres de décision.

→ Combinaison des prédictions de l'ensemble des arbres appris pour prédire la (ou les) classes d'un exemple

Apprentissage par combinaison de classificateurs

❑ Idée générale



Algorithme : Random Forests

- Soit l'ensemble S comportant n exemples et d attributs.
- Soit $m << d$ le nombre d'attributs à utiliser à chaque nœud.
- Soit k le nombre d'arbres à générer
- Créer un échantillon bootstrap S_i de taille n_i partir de S.
- Apprentissage d'un arbre de décision sur S_i :
  - ➢ Pour chaque nœud de l'arbre, choisir aléatoirement m attributs parmi d (tirage sans remise) à partir desquels le meilleur attribut sera sélectionné.
- Itérer le processus k fois.

Random Forests

- Lors de la phase de prédiction, l'exemple à classer est propagé dans chaque arbre de la forêt et étiqueté en fonction des règles identifiées pour chaque arbre.
- La prédiction globale de la forêt est fournie par un vote à la majorité simple des attributions de classe des arbres individuels.

Random Forests

- Pour de nombreux jeux de données, le classificateur résultant fournit des prédictions de meilleure qualité.
- Permets de gérer des jeux de données ayant de grandes dimensions.
- Permets de déterminer l'importance des attributs dans le processus de classification (sélection d'attributs / feature selection)

Exemple

Le code complet se trouve dans Moodle : Section Arbre de décision/Exemple – Iris dataset

Entrée [38]:
```
from sklearn.ensemble import RandomForestClassifier
```

Entrée [39]:
```
rnd_forest  = RandomForestClassifier()
```

Entrée [40]:
```
#Entraînement avec Random Forest
rnd_forest.fit(train_features,train_labels)
```

Out[40]:
```
RandomForestClassifier()
```

Entrée [41]:
```
# Prédiction sur les données de test en utilisant rnd_forest
predicted_labels = rnd_forest.predict(test_features)
```

Entrée [42]:
```
#Calcul de l’exactitude (Accuracy) de la classification
from sklearn.metrics import accuracy_score
acc = accuracy_score(test_labels, predicted_labels)
print("Random Forest Accuracy Score = ", acc)
```

```
Random Forest Accuracy Score =  1.0
```

Entrée [43]:
```
#Mesure de l'importance des quatre attributs du dataset Iris en utlisant la mesure Mean Dcrease in Impurity
#À noter que à la fin de l’entraînement du Random Forest,
#                                            Scikit-Learn calcule automatiquement cette valeur pour chaque attribut
#Cela est possible grâce à la variable feature_importances_
#Le code suivant imprime l'importance de chaque attribut : une grande valeur indique une plus grande importance
for name, score in zip(iris_data[features], rnd_forest.feature_importances_):
    print(name,score)
```

```
sepal.length 0.08712563579243131
sepal.width 0.0255705736201352
petal.length 0.4536786713054945
petal.width 0.43362511928193903
```

Comparaison entre arbre de décision et techniques d'apprentissage par ensemble

Data Set	Number of (Attributes, Classes, Records)	Decision Tree (%)	Bagging (%)	Boosting (%)	RF (%)
Anneal	(39, 6, 898)	92.09	94.43	95.43	95.43
Australia	(15, 2, 690)	85.51	87.10	85.22	85.80
Auto	(26, 7, 205)	81.95	85.37	85.37	84.39
Breast	(11, 2, 699)	95.14	96.42	97.28	96.14
Cleve	(14, 2, 303)	76.24	81.52	82.18	82.18
Credit	(16, 2, 690)	85.8	86.23	86.09	85.8
Diabetes	(9, 2, 768)	72.40	76.30	73.18	75.13
German	(21, 2, 1000)	70.90	73.40	73.00	74.5
Glass	(10, 7, 214)	67.29	76.17	77.57	78.04
Heart	(14, 2, 270)	80.00	81.48	80.74	83.33
Hepatitis	(20, 2, 155)	81.94	81.29	83.87	83.23
Horse	(23, 2, 368)	85.33	85.87	81.25	85.33
Ionosphere	(35, 2, 351)	89.17	92.02	93.73	93.45
Iris	(5, 3, 150)	94.67	94.67	94.00	93.33
Labor	(17, 2, 57)	78.95	84.21	89.47	84.21
Led7	(8, 10, 3200)	73.34	73.66	73.34	73.06
Lymphography	(19, 4, 148)	77.03	79.05	85.14	82.43
Pima	(9, 2, 768)	74.35	76.69	73.44	77.60
Sonar	(61, 2, 208)	78.85	78.85	84.62	85.58
Tic-tac-toe	(10, 2, 958)	83.72	93.84	98.54	95.82
Vehicle	(19, 4, 846)	71.04	74.11	78.25	74.94
Waveform	(22, 3, 5000)	76.44	83.30	83.90	84.04
Wine	(14, 3, 178)	94.38	96.07	97.75	97.75
Zoo	(17, 7, 101)	93.07	93.07	95.05	97.03