Configuration de modèles et expérimentations

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

Professeur: Christian Gagné

Semaine 14

UNIVERSITÉ LAVAL

14.1 Évaluations et comparaisons d'algorithmes

Évaluations et comparaisons d'algorithmes

- Problème de l'évaluation des performances
  - Comment évaluer la performance d'un algorithme de classement sur un problème (en généralisation) ?
  - Grande différence entre performance sur jeu d'entraînement et jeu de test ?
- Problème de comparaison des performances
  - Comment évaluer si un algorithme performe mieux qu'un autre pour un certain problème ?
  - Différents types de comparaisons possibles
    - Différents algorithmes
    - Mêmes algorithmes, différents hyperparamètres
    - Mêmes algorithmes, différentes représentations des données
- Répétitions des mesures nécessaires pour validité statistique
  - Partionnement *aléatoire* pour entraînement/validation
  - Processus d'apprentissage avec résultats variables
    - Algorithme stochastique (ex. initialisation poids PMC)
    - Algorithme sensible aux choix des hyperparamètres (ex. valeurs σ et C du SVM)

Exemple des charlatans (Jensen et Cohen, 2000)

- Évaluation d'un conseiller en investissements
  - À chaque jour, le conseiller doit prédire si les cours boursiers seront à la hausse ou à la baisse
  - Test : prédire les cours boursiers pour 14 jours
  - Critère de sélection : prédiction correcte pour 11 jours ou plus
    - Charlatan fait des prédictions aléatoires (0,5/0,5)
    - Charlatan a donc une probabilité de 0,0287 de réussir le test
  - Bon test pour évaluer les performances d'un conseiller
- Mais n'est pas adapté au choix d'un conseiller parmi n
  - Probabilité qu'un charlatan parmi n passe le test : $1 - (1 - 0{,}0287)^n$
  - Pour $n = 10$, probabilité $\approx 0{,}253$; pour $n = 30$, probabilité $\approx 0{,}583$
  - Pour valeur élevée de n, presque certainement que des charlatans vont passer le test, même s'ils ne font pas mieux que le hasard !

D. Jensen, P. Cohen, *Multiple Comparisons in Induction Algorithms*, Machine Learning, n^o 38, p. 309–338, 2000.

Pathologies en apprentissage

- Surapprentissage
  - Ajouter éléments superflus au modèle (apprendre par cœur)
    - Faible valeur de C avec SVM, trop de vecteurs de support
  - Découvrir des relations inexistantes entre les données
    - Surentraîner PMC : apprendre faux liens entre données
  - Faire des modèles plus complexes n'offrant aucun avantage
- Erreurs dans sélection d'information discriminante
  - Biais dans l'algorithme favorise certains types de données
    - Classement paramétrique avec loi normale multivariée et matrice de covariance diagonale : biais vers discrimination de variables indépendantes
  - Sensibilité aux probabilités *a priori* des données (balances des classes)
  - Sensibilité aux choix des caractéristiques
- Sur-recherche
  - Faire une recherche dans de très vastes espaces de modèles
    - Solution : d'abord espaces de modèles simples, puis augmenter complexité
  - Similaire à augmenter valeur de n avec l'exemple des charlatans
    - Solution : resserrer le critère de sélection lorsque n augmente

Facteurs à considérer (1/2)

- Difficile de généraliser toutes conclusions faites sur un problème particulier à d'autres problèmes
  - Théorème du *No Free Lunch* !
  - Bon algorithme pour un problème : compatibilité entre le biais inductive et le problème
- Partitionnement du jeu de données en partitions entraînement/validation pour tests seulement
  - Bon pour évaluation/comparaison des performances en généralisation d'algorithmes
  - Bon pour choix des hyperparamètres
  - Une fois choix des algorithmes/hyperparamètres fait : utilisation de tout le jeu de données pour l'entraînement

Facteurs à considérer (2/2)

- Partition de validation fait partie des données d'inférence
  - Choix d'hyperparamètre ou critère d'arrêt
    - Chaque utilisation du jeu de validation intègre de l'information dans l'algorithme d'apprentissage
  - Évaluation finale des performances sur jeu de test distinct, **jamais** utilisé dans l'apprentissage
- Autres critères pour évaluation et comparaison d'algorithmes
  - Autres mesures du risque, autres fonctions de perte
  - Complexité de l'entraînement (temps et espace)
  - Complexité de l'évaluation (temps et espace)
  - Interprétabilité des résultats
  - Facilité de programmation

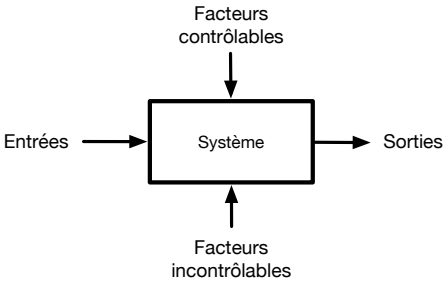
14.2 Plans d'expériences

Expérimentations

- Expérimentation : test ou série de tests où on joue avec des facteurs modifiant la sortie
  - Choix de l'algorithme d'apprentissage
  - Jeu de données d'entraînement
  - Caractéristiques des données
- Objectifs généraux
  - Identifier les facteurs les plus influents
  - Éliminer les facteurs les moins importants
  - Déterminer la configuration des facteurs donnant les meilleurs résultats
- Objectifs en apprentissage
  - Résultats statistiquement significatifs (éliminer effet du hasard)
  - Meilleure performance en généralisation
  - Complexité (temps et espace) réduite
  - Robustesse

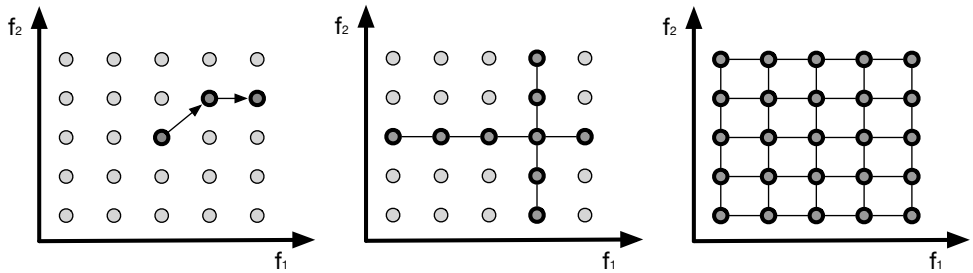
Processus d'expérimentations

- Facteurs contrôlables : éléments que l'on veut étudier
- Facteurs incontrôlables : éléments où on n'a pas le contrôle, mais dont on veut minimiser l'impact sur les décisions



Stratégies d'expérimentations

- Stratégies d'expérimentation possibles
  - Au pif : expérimentation basée sur l'intuition de l'opérateur
  - Un facteur à la fois : configuration de départ, en testant toutes les valeurs d'un facteur séparément
  - Recherche en grille : tester toutes les combinaisons



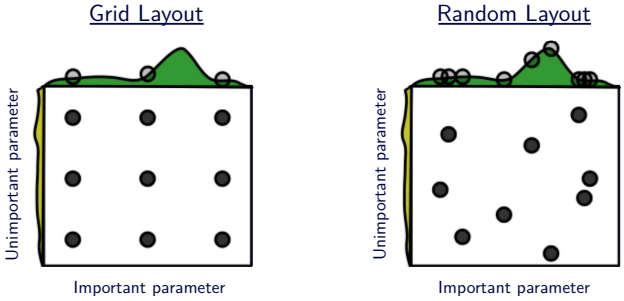
Au pif | Un facteur à la fois | Recherche en grille

Recherche en grille

- Recherche en grille : ajustement de paires (ou triplets) d'hyperparamètres, avec mesure sur ensemble de validation
  1. Partitionner ensemble de données $\mathcal{X}$ en deux sous-ensembles, $\mathcal{X}_T$ et $\mathcal{X}_V$ (généralement 50%-50%)
  2. Entraîner sommairement classifieur avec $\mathcal{X}_T$ pour chaque paire d'hyperparamètres considérés
  3. Sélectionner la paire d'hyperparamètres où l'erreur est minimale sur $\mathcal{X}_V$
  4. Utiliser cette paire d'hyperparamètres pour entraînement sur tout l'ensemble $\mathcal{X}$
- Applicable pour toutes paires d'hyperparamètres dont l'effet conjoint est important dans l'entraînement de classifieurs

Recherche aléatoire

- Sélectionner les valeurs d'hyperparamètres au hasard
  - Permet une meilleure exploration de l'espace en présence de variables sans influence



Tiré de *J. Bergstra et Y. Bengio, Random search for hyper-parameter optimization, Journal of Machine Learning Research, vol. 13, 2012.*
Disponible en-ligne au `https://www.jmlr.org/papers/v13/bergstra12a.html` .

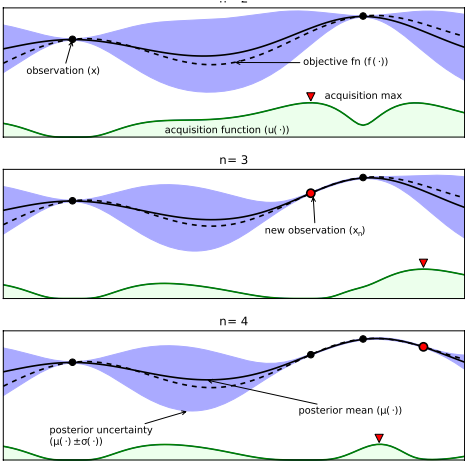
- Rafinnement possible : utilisation de nombres quasi-aléatoires
  - Séquence déterministe avec valeurs uniformément distribuées selon chaque dimension

14.3 Optimisation pour l'ajustement d'hyperparamètres

Optimisation séquentielle à base de modèle

- Idée : bâtir des modèles d'apprentissage pour estimer performance
  - Régression d'une fonction $\mathrm{f}(\mathbf{x})$ donnant la performance estimée selon hyperparamètres $\mathbf{x}$
  - Estimer incertitude des prédictions dans l'espace des hyperparamètres
  - Modèle couramment utilisé : processus gaussiens
    - Processus aléatoire générant une loi normale pour chaque valeur de $\mathbf{x}$
- Compromis exploration-exploitation : sélection de prochains hyperparamètres $\mathbf{x}$ à évaluer
  - Exploitation : sélectionner valeur de $\mathbf{x}$ avec bonne performance
  - Exploration : tester de nouvelle valeur de $\mathbf{x}$ pour acquérir plus d'information sur la fonction à optimiser
- Fonction d'acquisition pour déterminer prochaine valeur de $\mathbf{x}$
  - Fonction typique *Upper Confidence Bound* : $\mathrm{argmax}_{\mathbf{x}}\, \mu(\mathbf{x}) + \sigma(\mathbf{x})$
- Réestimer fonction de régression avec évaluation de la prochaine valeur

Optimisation bayésienne



Tiré de *B. Shahriari, K. Swersky, Z. Wang, R.P. Adams et N. De Freitas, Taking the human out of the loop : A review of bayesian optimization, Proceedings of the IEEE, vol. 104, no. 1, 2016.* Disponible en-ligne au `https://doi.org/10.1109/JPROC.2015.2494218`.

AutoML

- AutoML : automatiser l'apprentissage automatique
  - Permettre l'utilisation de ces techniques par des non experts
  - Permettre déploiement dans des situations inconnues, avec minimum d'intervention
  - Permettre l'adaptation des modèles aux conditions d'opération
- Choix de modèles et prétraitements
  - Au-delà des choix d'hyperparamètres, quel modèle utiliser ?
    - SVM, réseau de neurones, k-plus proches voisins, modèles linéaires, AdaBoost, forêts aléatoires, etc.
  - Raffiner la configuration des modèles
    - Nombre de couches cachées, fonction noyau, mesure de distance, etc.
  - Quels prétraitements faire des données ?
    - Normalisation, standardisation, sélection des variables, etc.
- Hors de l'optimisation des hyperparamètres, encore un sujet de recherche
  - Pas de modèles universels
  - Ressources en calcul requises peuvent être très importantes
  - Taille des jeux de données limite l'ampleur possible de la recherche de modèles

14.4 Organisation de plans d'expériences

Principes de base pour plan d'expériences

- Randomiser : l'ordre d'exécution des expériences doit être aléatoire, afin d'assurer une indépendance dans les résultats
  - Ex. : machine requiert un certain temps pour être à la bonne température
  - Généralement n'est pas un problème lors d'expérimentations avec du logiciel
- Reproduire : moyenner les résultats de plusieurs expériences avec les mêmes valeurs de facteurs contrôlables, pour éliminer l'effet des facteurs incontrôlables
  - En apprentissage : rouler le même algorithme avec différents échantillonnages du jeu de données (ex. validation croisée)
- Colmater : réduire ou éviter les facteurs de nuisance, influençant la sortie, mais n'étant pas d'intérêt
  - En apprentissage : comparer des algorithmes en utilisant les mêmes échantillonnages de données (mêmes partitions)

Directives pour expérimentations en apprentissage

1. Établir l'objectif de l'étude
   - Estimer l'erreur d'une méthode sur problème particulier (erreur en deçà d'une valeur)
   - Comparer deux algorithmes sur un même problème (est-ce qu'un algorithme est meilleur que l'autre ?)
2. Sélectionner la variable de réponse
   - Erreur de classement ou erreur quadratique en régression
   - Fonction de perte arbitraire, mesure de risque, précision, rappel, complexité, etc.
3. Choix des facteurs et des niveaux
   - Valeurs d'hyperparamètres
   - Algorithmes d'apprentissage
   - Jeux de données
4. Choix du plan d'expériences
   - Faire un design factoriel, à moins d'être certain d'aucune interaction
   - Nombre de reproductions d'expériences inversement proportionnel à la taille des jeux (variance des résultats selon taille)
   - Éviter jeux de données synthétiques pour évaluer les performances

Directives pour expérimentations en apprentissage

5. Effectuer les expériences
   - Faire quelques exécutions préliminaires pour s'assurer que tout va comme prévu
   - Pour expériences exigeantes en ressources, sauvegarde d'états intermédiaires (*checkpoints*)
   - Les expériences doivent être reproductibles
   - Faire des comparaisons de bonne foi, en étant juste relativement aux différentes approches testées
6. Faire une analyse statistique des données
   - S'assurer que résultats ne sont pas subjectifs ou un produit du hasard
   - Tester des hypothèses statistiques : est-ce que l'erreur de A est significativement plus basse que B ?
7. Conclusions et recommandations
   - Une fois données obtenues et analysées, tirer des conclusions objectives
   - Conclusion fréquente : faire plus d'expérimentations !
   - Procéder itérativement : ne pas investir toutes les énergies pour compléter une étape du premier coup

14.5 Manipulation des jeux de données

Partitionnement et stratification

- Cas idéal : partitionner jeu $\mathcal{X}$ en K paires distinctes de jeux d'entraînement et de validation
  - Nécessite d'immenses jeux de données
- Solution : faire plusieurs partitions du même jeu de données

$$\{\mathcal{T}_i, \mathcal{V}_i\}_{i=1}^{K}$$

  - Compromis entre taille des jeux et recoupements
    - Grands jeux permettent meilleure inférence des classifieurs
    - Recoupements importants entre ensembles donnent des mesures non statistiquement indépendantes
- Partitionnement avec stratification
  - Respecter les probabilités *a priori* dans le partitionnement en jeux entraînement/validation
  - Évite des variations liées au biais des algorithmes selon proportions entre les classes

Effet de la taille des ensembles d'entraînement

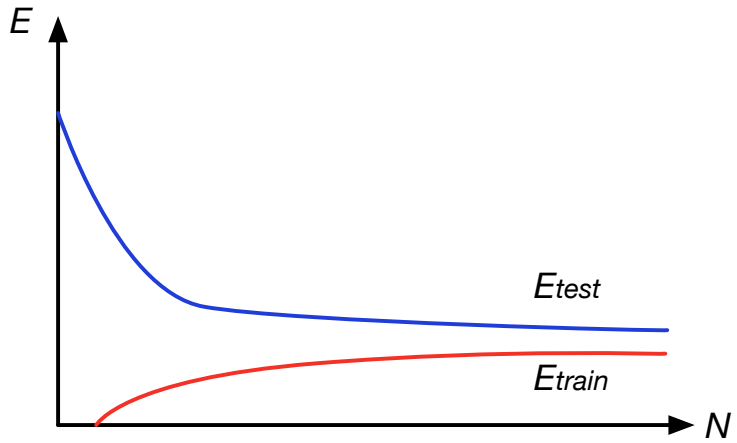
- Pour de vrais problèmes, courant que les taux d'erreurs en entraînement et test suivent des lois de puissance

$$\begin{aligned} E_{entrainement} &= E_{Bayes} - \frac{b}{N^{\beta}} \\ E_{test} &= E_{Bayes} + \frac{a}{N^{\alpha}} \end{aligned}$$

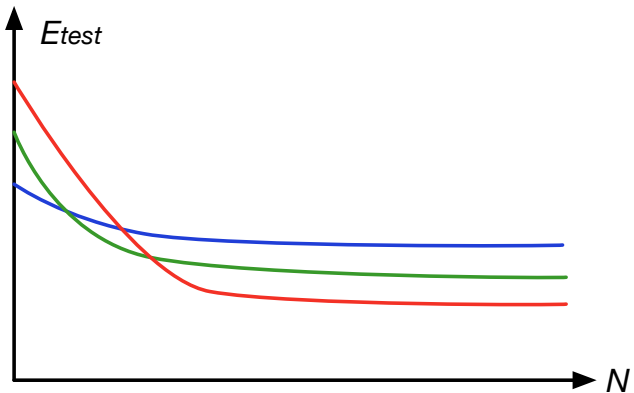
  où a, b, $\alpha \geq 1$ et $\beta \geq 1$ dépendent du classifieur et du problème
- Avec grands ensembles d'entraînement, les taux d'erreur tendent vers le taux bayésien optimal

$$\begin{aligned} \lim_{N \to \infty} E_{entrainement} &= E_{Bayes} \\ \lim_{N \to \infty} E_{test} &= E_{Bayes} \end{aligned}$$

Taux en entraînement et test selon N



Taux en test selon N



Validation croisée à K plis

- Validation croisée à K plis
  - Jeu d'entraînement divisé en K partitions disjointes, $\mathcal{X}_1 \cup \mathcal{X}_2 \cup \cdots \cup \mathcal{X}_K = \mathcal{X}$
  - K entraînements sur $\mathcal{T}_i$ et évaluation sur $\mathcal{V}_i$, $i = 1, \ldots, K$

$$\begin{array}{ll} \mathcal{V}_1 = \mathcal{X}_1 & \mathcal{T}_1 = \mathcal{X}_2 \cup \mathcal{X}_3 \cup \cdots \cup \mathcal{X}_K \\ \mathcal{V}_2 = \mathcal{X}_2 & \mathcal{T}_2 = \mathcal{X}_1 \cup \mathcal{X}_3 \cup \cdots \cup \mathcal{X}_K \\ \vdots & \vdots \\ \mathcal{V}_K = \mathcal{X}_K & \mathcal{T}_K = \mathcal{X}_1 \cup \mathcal{X}_2 \cup \cdots \cup \mathcal{X}_{K-1} \end{array}$$

  - Performance moyenne sur $\mathcal{V}_i$, $i = 1, \ldots, K$
  - $(K-2)/K$ des données partagé par chaque paire de jeux d'entraînement (non-indépendance statistique des résultats)
- *Leave-one-out* : $K = N$
  - Entraînement sur $N-1$ données, performance sur une donnée (répété N fois)
  - Utile pour des algorithmes avec temps d'entraînement réduits ou inexistants (ex. k-PPV), ou très petits jeux de données

Validation croisée 5×2

- Validation croisée 5×2
  - Diviser jeu $\mathcal{X}$ en deux partitions disjointes égales $\mathcal{X}_1^{(1)}$ et $\mathcal{X}_1^{(2)}$
  - Entraîner sur $\mathcal{T}_1 = \mathcal{X}_1^{(1)}$ et évaluer sur $\mathcal{V}_1 = \mathcal{X}_1^{(2)}$
  - Répéter avec entraînement sur $\mathcal{T}_2 = \mathcal{X}_1^{(2)}$ et évaluation sur $\mathcal{V}_2 = \mathcal{X}_1^{(1)}$
  - Répéter cinq fois, pour un total de 10 entraînements/évaluations

$$
\begin{array}{ll}
\mathcal{T}_1 = \mathcal{X}_1^{(1)} & \mathcal{V}_1 = \mathcal{X}_1^{(2)} \\
\mathcal{T}_2 = \mathcal{X}_1^{(2)} & \mathcal{V}_2 = \mathcal{X}_1^{(1)} \\
\mathcal{T}_3 = \mathcal{X}_2^{(1)} & \mathcal{V}_3 = \mathcal{X}_2^{(2)} \\
\mathcal{T}_4 = \mathcal{X}_2^{(2)} & \mathcal{V}_4 = \mathcal{X}_2^{(1)} \\
\vdots & \vdots \\
\mathcal{T}_9 = \mathcal{X}_5^{(1)} & \mathcal{V}_9 = \mathcal{X}_5^{(2)} \\
\mathcal{T}_{10} = \mathcal{X}_5^{(2)} & \mathcal{V}_{10} = \mathcal{X}_5^{(1)}
\end{array}
$$

- Plus de cinq répétitions : trop de dépendances entre les jeux de données
- Moins de dix résultats : pas assez d'échantillons pour estimer une distribution et faire des tests statistiques

Bootstrapping

- *Bootstrapping* : échantillonnage **avec remise**
  - Générer jeu d'entraînement en échantillonnant N données avec remise parmi N données du jeu d'origine
  - Validation sur un jeu d'entraînement différent, généré de la même façon
  - Répéter autant de fois que nécessaire pour évaluer les performances
  - Probabilité d'échantillonner une donnée est $1/N$
    - Pour jeu de N données, probabilité qu'une certaine donnée ne soit pas tirée

$$\left(1 - \frac{1}{N}\right)^N \approx e^{-1} = 0{,}368$$

  - Environ 63,2 % des données originales présentes dans jeu échantillonné
- Plus grande dépendance entre jeux échantillonnés qu'avec validation croisée
  - Tout de même excellent pour évaluer performances avec de petits jeux de données
  - Bon également pour évaluer stabilité d'un algorithme

14.6 Mesures d'erreurs et courbes ROC

Mesures d'erreurs et matrice de confusion

- Matrice de confusion : explication des erreurs effectuées

	Décision	
Vérité	1	0
1	$\|TP\|$	$\|FN\|$
0	$\|FP\|$	$\|TN\|$

- Redéfinition du taux d'erreur : $E = \frac{|FN|+|FP|}{N}$
  - Avec $N = |TP| + |FP| + |TN| + |FN|$
- Pondération selon type d'erreurs (coûts variables)

$$E = \frac{c_{FN}|FN| + c_{FP}|FP|}{N}$$

- Généralisation directe à K classes

Courbes ROC

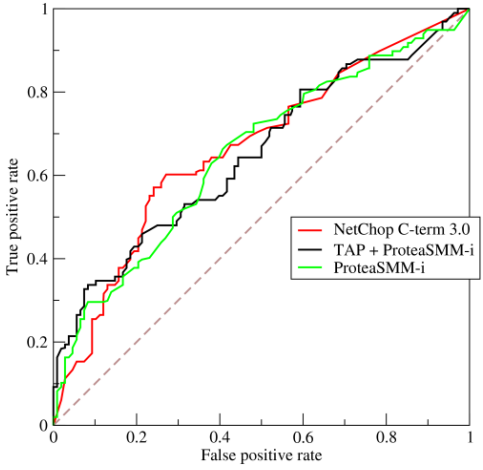
- Courbe ROC (*receiver operator characteristics*)
  - Taux de décisions correctes

$$\frac{|TP|}{|TP| + |FN|}$$

  - Taux de fausses alarmes

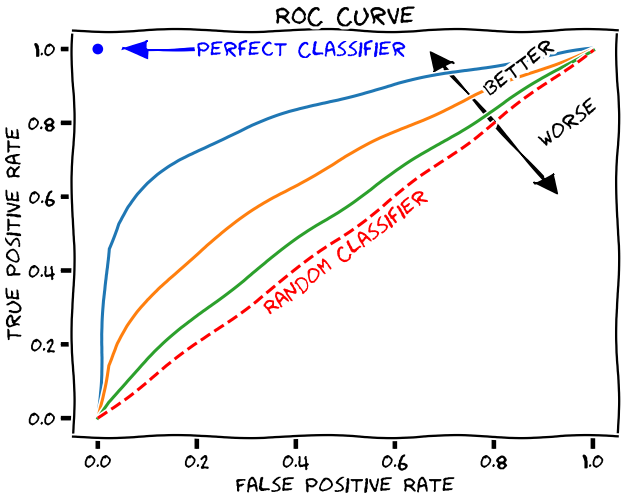
$$\frac{|FP|}{|FP| + |TN|}$$

  - Différents seuils d'acceptation donnent différents points d'opérations sur la courbe



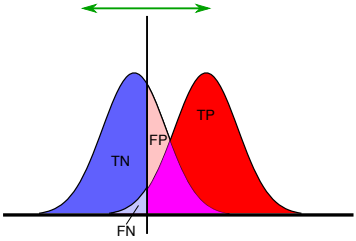
Par BOR, CC-BY-SA 3.0,
https://commons.wikimedia.org/wiki/File:Roccurves.png.

Courbes ROC pour le classement

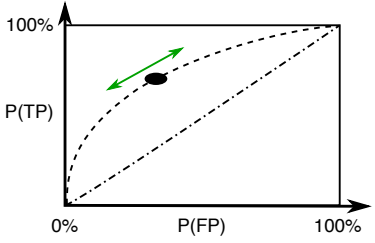


Par MartinThoma, domaine public, https://commons.wikimedia.org/wiki/File:Roc-draft-xkcd-style.svg.

Seuil de décision de courbes ROC



TP	FP
FN	TN



Par Sharpr, CC-BY-SA 3.0, https://commons.wikimedia.org/wiki/File:ROC_curves.svg.

AUC-ROC, sensibilité et spécificité

- Aire sous la courbe ROC (AUC-ROC) : mesure de performance indépendante du seuil
  - Capacité du classifieur à bien discriminer deux classes pour tous les seuils
  - Similarité avec test non paramétrique Wilcoxon-Mann-Whitney
- Sensibilité : nombre de positifs correctement identifiés

$$\text{sensibilité} = \frac{|TP|}{|TP| + |FP|}$$

- Spécificité : nombre de négatifs correctement identifiés

$$\text{spécificité} = \frac{|TN|}{|TN| + |FN|} = 1 - \frac{|FP|}{|TN| + |FN|}$$

Précision et rappel

- Recherche d'information dans des bases de données
  - Entrées extraites suite à une requête : positifs
  - Entrées pertinentes à une requête : vrais positifs + faux négatifs
- Précision : # entrées extraites pertinentes par # entrées extraites

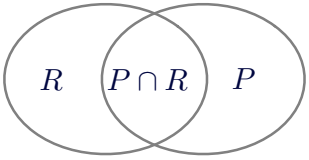
$$\text{précision} = \frac{|TP|}{|TP| + |FP|}$$

  - Précision de 1 : entrées extraites toutes pertinentes, mais peut rester faux négatifs
  - Équivalent à la sensibilité
- Rappel : # entrées extraites pertinentes par # entrées pertinentes

$$\text{rappel} = \frac{|TP|}{|TP| + |FN|}$$

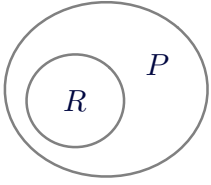
  - Rappel de 1 : toutes les entrées pertinentes sont extraites, mais il y a peut-être des entrées extraites non pertinentes (faux positifs)

Précision et rappel

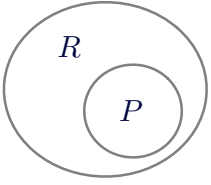


$$\text{Precision} = \frac{|R|}{|P \cap R|}$$

$$\text{Rappel} = \frac{|P \cap R|}{|P|}$$



Précision = 1



Rappel = 1

14.7 Intervalle de confiance et lois statistiques

Intervalle de confiance

- Estimateur (ex. maximum de vraisemblance) : une valeur d'un paramètre
- Intervalle de confiance : la plage de valeurs plausibles d'un paramètre, à un certain degré de confiance
  - Basé sur la densité de probabilité sous-jacente de l'estimateur
- Exemple : estimation de moyenne μ d'une loi normale à partir d'échantillons $\mathcal{X} = \{x^t\}_{t=1}^N$
  - Estimateur par moyenne des échantillons : $m = \sum_t x^t / N$
  - m est une somme de variables normales, et donc également normale, $m \sim \mathcal{N}(\mu, \sigma^2/N)$
  - Selon la loi normale, on a donc confiance à 95 % que $\mu \in [m - 1{,}96\sigma/\sqrt{N}, m + 1{,}96\sigma/\sqrt{N}]$

$$P\left(m - 1{,}96\frac{\sigma}{\sqrt{N}} < \mu < m + 1{,}96\frac{\sigma}{\sqrt{N}}\right) = 0{,}95$$

Intervalle de confiance

- Loi $\mathcal{Z}$: loi normale de moyenne nulle et variance unitaire, $\mathcal{Z} \equiv \mathcal{N}(0,1)$
- Formalisation générale d'intervalle de confiance pour loi normale :
  $Z \sim \mathcal{Z}$, $P(Z > z_\alpha) = \alpha$, $\alpha \in [0,1]$
  - Loi normale de moyenne nulle est symétrique
    - Borne simple : $P(-z_\alpha < Z) = 1-\alpha$, $P(Z < z_\alpha) = 1-\alpha$, $\alpha \in [0,1]$
    - Borne double : $P(-z_{0,5\alpha} < Z < z_{0,5\alpha}) = 1-\alpha$, $\alpha \in [0,1]$
- Estimation de la moyenne de l'échantillon, $m \sim \mathcal{N}(\mu,\sigma^2/N)$, implique

$$\begin{aligned} \sqrt{N}\frac{m-\mu}{\sigma} &\sim \mathcal{Z} \\ P\left(m - z_\alpha \frac{\sigma}{\sqrt{N}} < \mu\right) &= 1-\alpha \\ P\left(\mu < m + z_\alpha \frac{\sigma}{\sqrt{N}}\right) &= 1-\alpha \end{aligned}$$

Loi du χ^2

- Si $Z_i \sim \mathcal{Z}$ sont des variables aléatoires indépendantes, et

$$X = Z_1^2 + Z_2^2 + \cdots + Z_n^2$$

  alors X suit une loi du χ^2 à n degrés de liberté, $X \sim \chi_n^2$
- Espérance de $\mathbb{E}[X] = n$ et variance $\mathrm{Var}(X) = 2n$
- Pour un échantillonnage $x^t \sim \mathcal{N}(\mu,\sigma^2)$
  - Estimation de variance : $s^2 = \frac{\sum_t (x^t - m)^2}{N-1}$
  - $(N-1)\frac{s^2}{\sigma^2} \sim \chi_{N-1}^2$
- Loi du χ^2 excellente pour faire des tests statistiques sur plusieurs variables aléatoires suivant des lois normales
  - Par exemple, plusieurs estimations d'un taux de classement

Loi de Student

- Loi de Student : appropriée pour faire des tests sur des distributions normales où on a peu d'échantillons
- Si $Z \sim \mathcal{Z}$ et $X \sim \chi^2_n$ sont indépendants, alors $T_n \sim t_n$, suit une loi de Student à n degrés de liberté

$$T_n = \frac{Z}{\sqrt{X/n}}$$

  - Avec n grand, distribution a une forme similaire à une distribution normale de moyenne nulle
  - Espérance $\mathbb{E}[T_n] = 0$, variance $\mathrm{Var}(T_n) = \frac{n}{n-2}$, pour $n > 2$

14.8 Tests statistiques

Test d'hypothèses

- Test d'hypothèse : méthode classique pour tester validité statistique de résultats
  - Faire l'hypothèse qu'une variable aléatoire suit une certaine loi de densité
  - Estimer la probabilité que la variable respecte l'hypothèse selon les statistiques obtenues de mesures
  - Si la probabilité est suffisant élevée, le test est positif (hypothèse nulle vérifiée)
- Test-t (loi de Student)
  - Différence entre vraie moyenne μ_0 et moyenne m de N échantillons, ayant une variance s, suit une loi de Student à $N-1$ degrés de liberté

$$\frac{\sqrt{N}(m-\mu_0)}{s} \sim t_{N-1}$$

  - Hypothèse vérifiée avec une probabilité $1-\alpha$ lorsque :

$$\frac{\sqrt{N}(m-\mu_0)}{s} \in [-t_{0,5\alpha,\,N-1},\, t_{0,5\alpha,\,N-1}]$$

Test-t apparié

- Utilisation du test-t pour la validation croisée à K plis
  - K pourcentages d'erreur p_i sur jeux de validation $\mathcal{V}_i$, $i = 1, \ldots, K$

$$p_i = \frac{\sum_{\mathbf{x}^t \in \mathcal{V}_i} \mathbb{I}(r^t, \mathrm{h}(\mathbf{x}^t | \mathcal{T}_i))}{N}$$

  - Moyenne et variance des résultats avec validation croisée à K plis

$$m = \frac{\sum_{i=1}^{K} p_i}{K}, \quad s^2 = \frac{\sum_{i=1}^{K} (p_i - m)^2}{K - 1}$$

  - Test-t apparié effectué selon

$$\frac{\sqrt{K}(m - p_0)}{s} \sim t_{K-1}$$

    où p_0 est le taux d'erreur vérifié par le test d'hypothèse

- Donc, taux d'erreur inférieur à p_0 avec probabilité $1 - \alpha$ si test suivant positif

$$\frac{\sqrt{K}(m - p_0)}{s} < t_{\alpha, K-1}$$

Test-t apparié pour comparaison de résultats

- Comparaison de deux algorithmes entraînés par validation croisée à K plis
  - p_i^1 : erreur classement sur $\mathcal{V}_i$ du premier algorithme entraîné sur $\mathcal{T}_i$
  - p_i^2 : erreur classement sur $\mathcal{V}_i$ du deuxième algorithme entraîné sur $\mathcal{T}_i$
  - Différence de l'erreur classement sur plis i : $p_i = p_i^1 - p_i^2$
  - Test d'hypothèse : valeur moyenne de p_i est nulle
  - Moyenne et variance de la différence de l'erreur

$$m = \frac{\sum_{i=1}^{K} p_i}{K}, \quad s^2 = \frac{\sum_{i=1}^{K} (p_i - m)^2}{K-1}$$

  - La différence d'erreur p_i suit une loi de Student à $K-1$ degrés de liberté

$$\frac{\sqrt{K}(m-0)}{s} = \frac{\sqrt{K}m}{s} \sim t_{K-1}$$

- Algorithme avec performance statistiquement identique, avec probabilité $1-\alpha$, si test suivant positif

$$\frac{\sqrt{K}m}{s} \in [-t_{0,5\alpha,K-1}, t_{0,5\alpha,K-1}]$$

Analyse de la variance (ANOVA)

- ANOVA : comparer plusieurs algorithmes de classement
  - Comment comparer L algorithmes, chacun entraîné et testé sur K paires de partitions différentes ?
  - Hypothèse que chaque résultat $E_{i,j}$ suit une loi normale de moyenne

$$E_{i,j} \sim \mathcal{N}(\mu_j, \sigma^2),\ i = 1, \ldots, K,\ j = 1, \ldots, L$$

    - Moyenne μ_j inconnue et différente pour chaque algorithme
    - Variance σ^2 partagée par tous les plis/algorithmes
  - Hypothèse H_0 : toutes les moyennes μ_j sont égales

$$H_0 : \mu_1 = \mu_2 = \cdots = \mu_L$$

- Approche d'ANOVA : deux estimateurs différents de σ^2
  - Premier estimateur de σ^2 valide seulement lorsque H_0 est vraie
  - Deuxième estimateur de σ^2 valide peu importe la validité de H_0

Premier estimateur de σ^2 avec ANOVA

- Premier estimateur de σ^2 : H_0 est valide
  - Moyenne par algorithme sur les K plis : $m_j = \frac{\sum_{i=1}^{K} e_{i,j}}{K}$
  - Moyenne et variance des m_j

$$m = \frac{\sum_{j=1}^{L} m_j}{L}, \quad s^2 = \frac{\sum_{j=1}^{L}(m_j - m)^2}{L-1}$$

  - Estimateur de σ^2

$$\hat{\sigma}^2 = Ks^2 = K\frac{\sum_{j=1}^{L}(m_j - m)^2}{L-1}$$

  - Comme chaque m_j suit une loi normale, on peut dire

$$\frac{(L-1)s^2}{\sigma^2/K} = \frac{K\sum_{j=1}^{L}(m_j - m)^2}{\sigma^2} \sim \chi^2_{L-1}$$

  - En posant $S_b \equiv K\sum_{j=1}^{L}(m_j - m)^2$, on obtient H_0 est valide lorsque

$$\frac{S_b}{\sigma^2} \sim \chi^2_{L-1}$$

Deuxième estimateur de σ^2 avec ANOVA

- Deuxième estimateur de σ^2 : indépendant de validité de H_0
  - σ^2 : moyenne de la variance s_j^2 des algorithmes

$$
\begin{aligned}
s_j^2 &= \frac{\sum_{i=1}^{K}(e_{i,j} - m_j)^2}{K-1} \\
\hat{\sigma}^2 &= \sum_{j=1}^{L}\frac{s_j^2}{L} = \sum_{j=1}^{L}\sum_{i=1}^{K}\frac{(e_{i,j} - m_j)^2}{L(K-1)}
\end{aligned}
$$

  - En posant $S_w \equiv \sum_{j=1}^{L}\sum_{i=1}^{K}(e_{i,j} - m_j)^2$

$$
(K-1)\sum_{j=1}^{K}\frac{s_j^2}{\sigma^2} = (K-1)\sum_{j=1}^{K}\frac{\sum_{i=1}^{K}(e_{i,j} - m_j)^2}{(K-1)\sigma^2} = \frac{S_w}{\sigma^2} \sim \chi^2_{L(K-1)}
$$

ANOVA

- Loi de Fisher : ratio de deux lois du χ^2 indépendantes

$$F_{n,m} = \frac{X_1/n}{X_2/m}, \quad \text{où } X_1 \sim \chi^2_n \text{ et } X_2 \sim \chi^2_m$$

- ANOVA : rejeter hypothèse H_0 si les deux estimateurs de σ^2 diffèrent significativement

$$\begin{aligned} H_0 : \mu_1 &= \mu_2 = \cdots = \mu_L \\ \frac{\frac{S_b/\sigma^2}{L-1}}{\frac{S_w/\sigma^2}{L(K-1)}} &= \frac{S_b/(L-1)}{S_w/(L(K-1))} = \frac{L(K-1)}{L-1}\frac{S_b}{S_w} \sim F_{L-1,L(K-1)} \end{aligned}$$

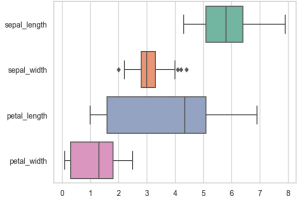
- Donc hypothèse que taux de classement moyens sont égaux pour tous les algorithmes est valide à une probabilité $1-\alpha$ lorsque

$$\frac{L(K-1)}{L-1}\frac{S_b}{S_w} < F_{\alpha,L-1,L(K-1)}$$

14.9 Outils d'expérimentation dans Python

Outils d'expérimentation dans Python

- `sklearn.model_selection.cross_val_score` : validation croisée à K plis
- `scipy.stats.ttest_rel` et `scipy.stats.ttest_ind` : test-t, appariés ou individuels
- `scipy.stats.f_oneway` : analyse de la variance (ANOVA)
- `seaborn.boxplot` : comparaison graphique de plusieurs résultats (requiert librairie Seaborn)



Tiré de `https://seaborn.pydata.org/generated/seaborn.boxplot.html`.

- Auto-sklearn : AutoML avec scikit-learn
  `https://automl.github.io/auto-sklearn/master/`