Apprentissage sur un petit ensemble de données

Bousquet Jérémie, Hmamouche Youssef

Apprentissage automatique - problématique

Algorithmes permettant de résoudre un problème, une tâche, en apprenant des données - lorsqu'il n'est pas possible ou complexe d'écrire un algorithme pour résoudre la tâche.

$$X = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \dots & \dots & \dots & \dots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{pmatrix} \text{ et } y = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{pmatrix}$$

- Chaque ligne de X est une donnée structurée (un exemple), composée de p variables (features)
- A chaque exemple est associée une cible y_i, ce que l'on souhaite prédire
- $y = f(X) + \epsilon$ est ce que l'algorithme cherche à approximer au mieux (ϵ l'erreur)
- Si les y_i sont continus on parle de tâche de régression, s'ils sont discrets on parle de tâche de classification

Apprentissage automatique - problématique

Classification: reconnaître un objet dans une image (chaque ligne de X est une image, les p variables représentant les pixels de l'image, et par exemple $y \in \{chien, chat, ours, ...\}$)

Régression: prédiction du prix de vente d'une maison en fonction de critères (les p variables sont les critères, superficie, nombre de chambres, etc, et y est le prix de vente)

Les programmes utilisant ces algorithmes ont deux phases:

- Apprentissage: on approxime $f(X) = y$, X et y étant connus (apprentissage supervisé), obtenant $\hat{f}$
- Inférence: on prédit y à partir de f apprise et de nouvelles données $X^{'}$ pour lesquelles $y^{'}$ peut être connu (pour tester l'apprentissage) ou pas (problème réel).

D'autres formes d'apprentissage existent, par ex. non supervisé (pas de y, par exemple le clustering qui vise à détecter des groupements au sein des X)

Apprentissage automatique - dilemme biais-variance

- Biais inductif: on commence par faire une hypothèse dans le choix d'une famille de fonctions $\mathbb{H}$ pour f (ex. pour la régression: polynomiale, logistique..., pour la classification: Support Vector Machines, Bayésien, réseau de neurones ...)
- L'erreur commise sur plusieurs jeux de données peut se décomposer en biais, variance, et une erreur irréductible liée au bruit dans les données du problème.

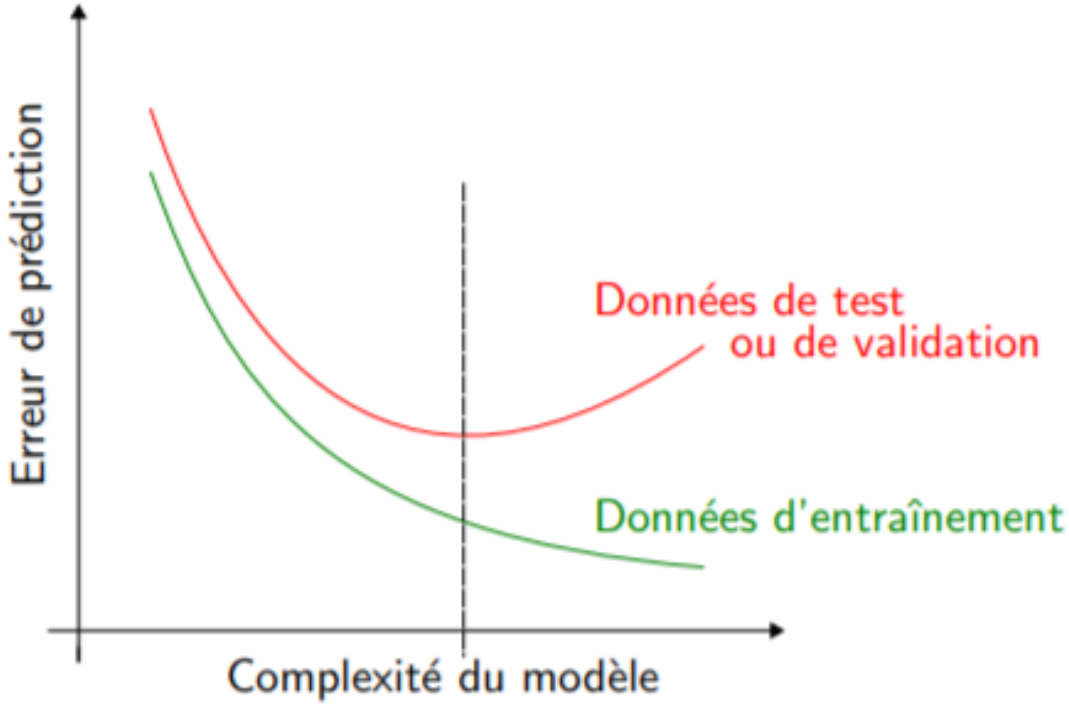
$$E[(y - \hat{f}(x))^2] = Biais[\hat{f}(x)^2] + Var[\hat{f}(x)] + \sigma^2$$

- Le biais est l'écart entre la fonction de prédiction moyenne $\hat{f}$ apprise sur plusieurs jeux de données, et la fonction recherchée f
- La variance est l'écart entre la fonction de prédiction moyenne apprise sur plusieurs jeux de données, et une fonction de prédiction apprise sur un seul jeu de données

Apprentissage automatique - dilemme biais-variance

Le biais et la variance sont liés:

- Si la complexité du modèle $\mathbb{H}$ choisi est faible, le biais sera important et la variance faible
- Si la complexité du modèle $\mathbb{H}$ choisi est forte, le biais sera faible mais la variance importante



Si le biais est trop important on parle de sous-apprentissage. Si la

Overfitting - comment le mesurer ?

Le terme “overfitting” correspond au cas où le prédicteur modélise trop étroitement les données d'apprentissage (jusqu'à les apprendre “par coeur”). Le biais est minimal mais la forte variance pénalise la capacité de généralisation à de nouvelles données.

- Evaluer la différence entre les erreurs des modèles sur les données de validation (entrainement) et les données de test. Exemple, utiliser des tests de signification statistiques.

Généralement on découpe les données en trois ensembles distincts : données d'apprentissage (le modèle ajuste ses paramètres internes sur ces données), données de validation (pour choisir les meilleurs hyper-paramètres d'un modèle, on compare les performances obtenues sur cet ensemble suivant les hyper-paramètres utilisés), et données de test (pour évaluer la capacité de généralisation du modèle sur des données totalement nouvelles pour lui).

Overfitting - comment le mesurer ?

- Eviter les biais méthodologiques (cf. par ex. Cawley et al. (2017)) Exemples:
  - Si des données sont utilisées pour apprendre ou sélectionner un modèle, alors elles ne doivent pas servir à évaluer ce modèle (risque d'overfitting sur l'apprentissage mais aussi sur la sélection du modèle).
  - Etre attentif aux métriques permettant d'évaluer objectivement le modèle (ex. pour la classification, ne pas se limiter à la précision qui pénalise les faux positifs, mais aussi le rappel qui pénalise les faux négatifs, ou le f-score qui combine les deux).
  - Choisir une méthode adaptée pour la découpe en train/valid/test. Par exemple si les classes sont déséquilibrées utiliser une découpe stratifiée (qui respecte le ratio entre classes dans chaque ensemble). Un modèle aléatoire peut avoir une très bonne précision sur des données aux classes fortement déséquilibrées !

Overfitting - comment l'éviter ?

- Des méthodes existent, généralement très dépendantes de la famille de modèles choisie ($\mathbb{H}$)
- Contraindre les valeurs d'hyper-paramètres recherchés suivant la connaissance de la famille de modèle.

Par exemple des valeurs extrêmes pour C et γ pour les SVMs peuvent encourager l'overfitting (mais il n'y a pas de règle absolue) - les hyper-paramètres peuvent en fait influer sur la complexité du modèle (et donc le dilemme biais-variance). Les connaissances empiriques et théoriques sur chaque famille de modèle peuvent permettre d'ajuster la recherche d'hyper-paramètres dans l'espoir d'éviter l'overfitting (mais aussi l'underfitting bien sûr).

Overfitting - comment l'éviter ?

- Régulariser l'apprentissage (lorsque cela est possible). Suivant la famille de modèles, la régularisation consiste à ajouter un terme de contrainte dans la fonction de perte, l'objectif étant (généralement) de pénaliser les modèles trop complexes lors de l'apprentissage. Note: la fonction de perte (ou coût) est une fonction que l'on cherche à minimiser lors de l'apprentissage.

Régression Ridge:

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \boxed{\lambda \sum_{j=1}^{p} \beta_j^2}$$

Les β_j forment la matrice que l'algorithme cherche à apprendre en minimisant cette fonction de coût. La présence du terme L1 contraint l'algorithme à tendre vers des solutions β_j "petites", donc à limiter l'espace de recherche des solutions, donc à limiter la complexité du modèle.

Overfitting - comment l'éviter ?

- "Augmenter" les données

On entend ici ajouter de nouvelles données synthétisées à partir des données existantes et par ajout d'un "bruit" suivant différentes méthodes (pour des images, on peut ajouter de nouvelles images en appliquant des transformations de type flip, rotations, bruit gaussien, recadrages, etc).

Paradoxalement en augmentant le bruit dans les données, à complexité équivalent le modèle aura plus de difficultés à s'adapter étroitement aux données, la variance aura tendance à être plus faible ainsi que l'overfitting.

Application - Prédiction de l'activité cérébrale en fonction des signaux multimodaux

- Méthode utilisée : k-fold cross-validation avec ensemble de test et de validation.
- Problèmatique : la cross validation pose quelques problèmes pour les données séquentielles car elle ne tient pas en compte l'ordre chronologiques des données, mais on peut la faire marcher dans notre cas si on considère chaque conversation comme un sous-ensemble sous l'hypothèse que l'ordre des conversations n'est pas important.

Application - Prédiction de l'activité cérébrale en fonction des signaux multimodaux

- Première stratégie : construire un seul modèle pour toutes les conversations.
- Dans ce cas, on peut découper les données en 4 blocks (comme découpé lors de l'expérience d'IRMf), chaque block contient 6 conversations. Sur chaque block on peut appliquer une k-fold cross-validation en gardant une seule conversation comme données de test. Pour les autres, on change aléatoirement l'ordre des conversations à chaque fois et fixant une conversation pour la validation, et on répète ce processus, jusqu'à ce que chaque conversation des données d'entrainement est utilisée une fois pour la validation.
- Deuxième stratégie : deux modèles séparés, un pour les conversations humain-humain (HH) et l'autre pour les conversations humain-robot (HR).

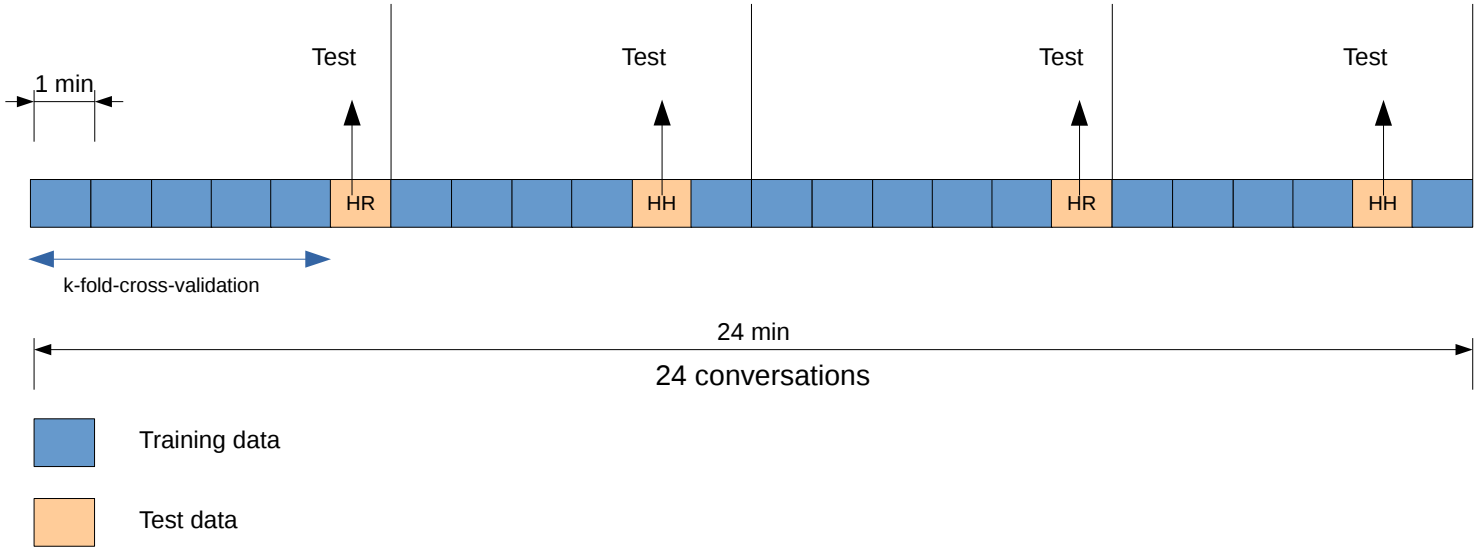
Application - Prédiction de l'activité cérébrale en fonction des signaux multimodaux

Modélisation 1

- 20 conversations pour l'entrainement, et 4 conversations de test (2 HH et 2 HR).
- 5-fold cross-validation avec ensemble de test et de validation sur les 4 blocks :
  - Division des conversations en 4 blocks.
  - Sur chaque block, une conversation est extraite comme données de test.
  - Application d'une 5-fold cross-validation sur le reste des conversations.

Application - Prédiction de l'activité cérébrale en fonction des signaux multimodaux

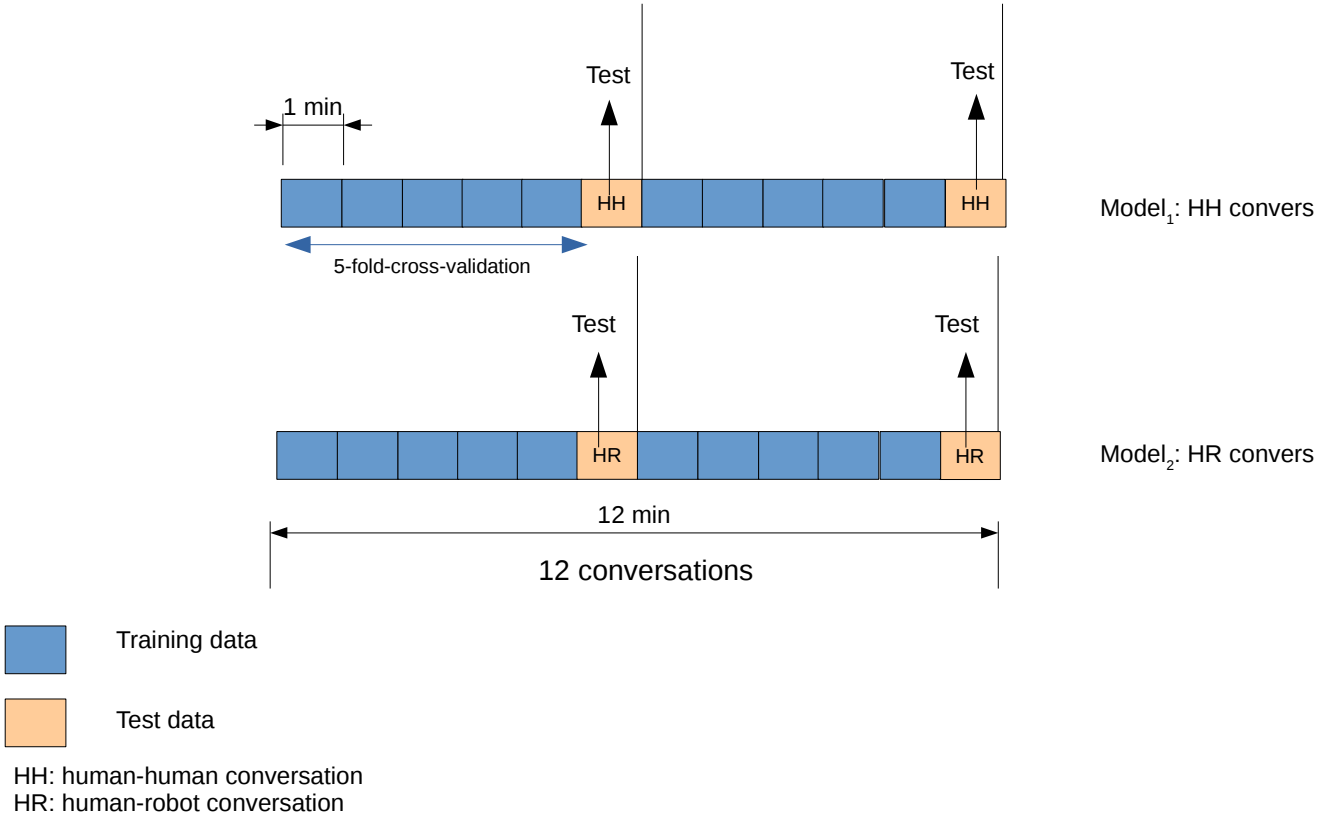
Modélisation 1 : un modèle pour toutes les conversations



HH: human-human conversation
HR: human-robot conversation

Application - Prédiction de l'activité cérébrale en fonction des signaux multimodaux

Modélisation 2 : deux modèles selon le type des conversations

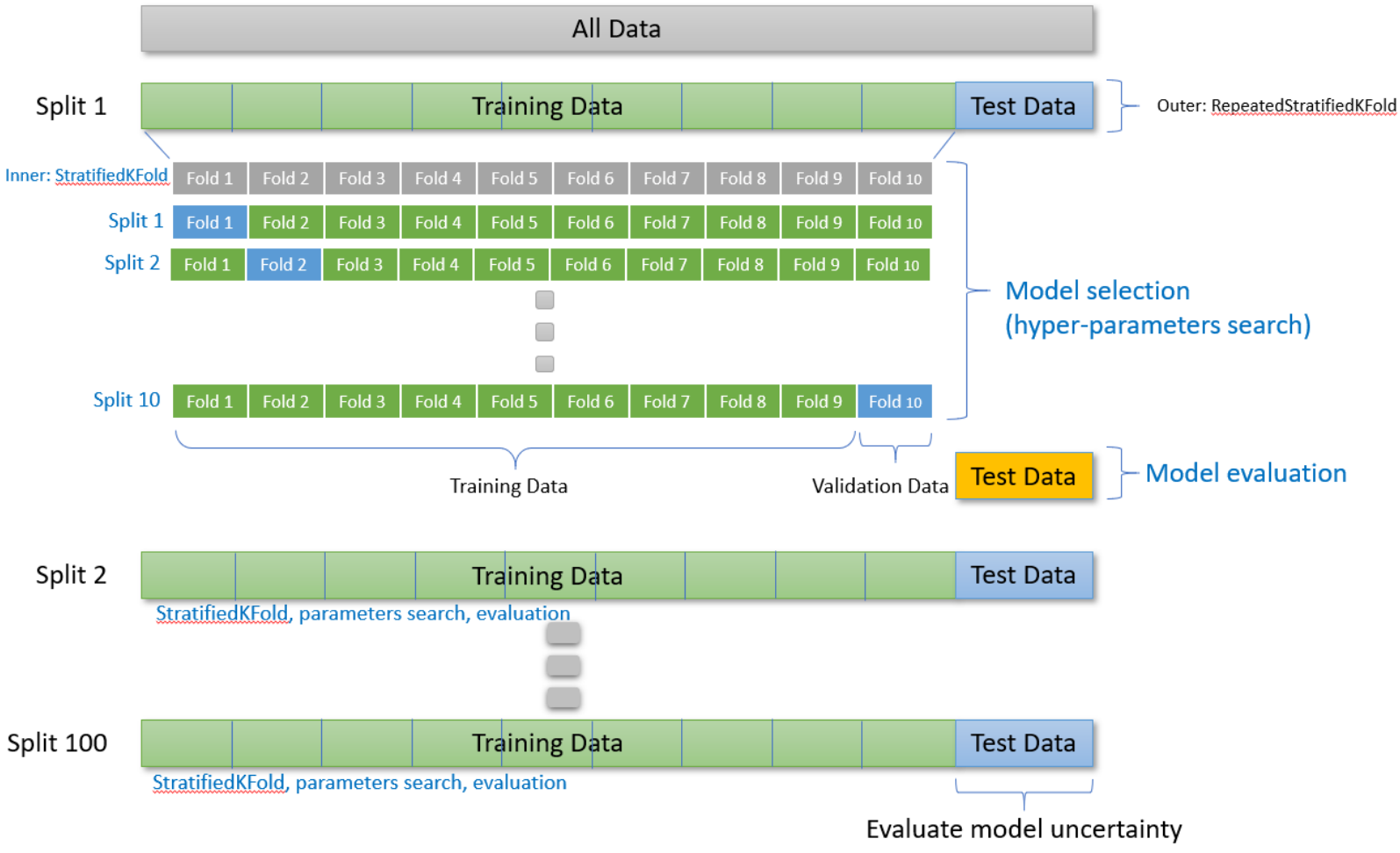


Application - Prédiction du sentiment de (co)présence

Méthodologie:

- discrétisation des scores de présence/co-présence en problèmes de classification binaire
- 10x 10-folds (90% train / 10% test), splits aléatoires stratifiés (= conservant les proportions de chaque classe)
  - 10-fold cross-validation sur l'ensemble train (du coup train+validation) pour la recherche d'hyper-paramètres du modèle
  - évaluation de la capacité de prédiction sur l'ensemble test (non vu lors de l'apprentissage)
  - moyennage des scores de test sur les 10x10 splits (avec calcul de l'intervalle de confiance)

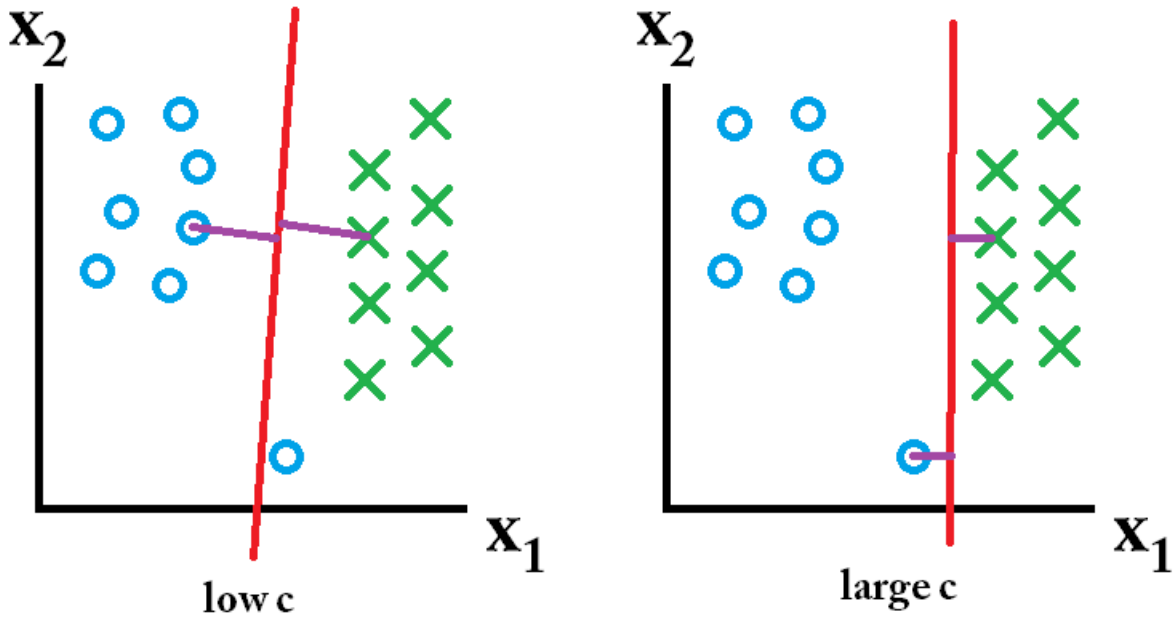
Application - Prédiction du sentiment de (co)présence



Application - Prédiction du sentiment de (co)présence

Overfitting: cas du Support Vector Machines (SVM) (1/2)

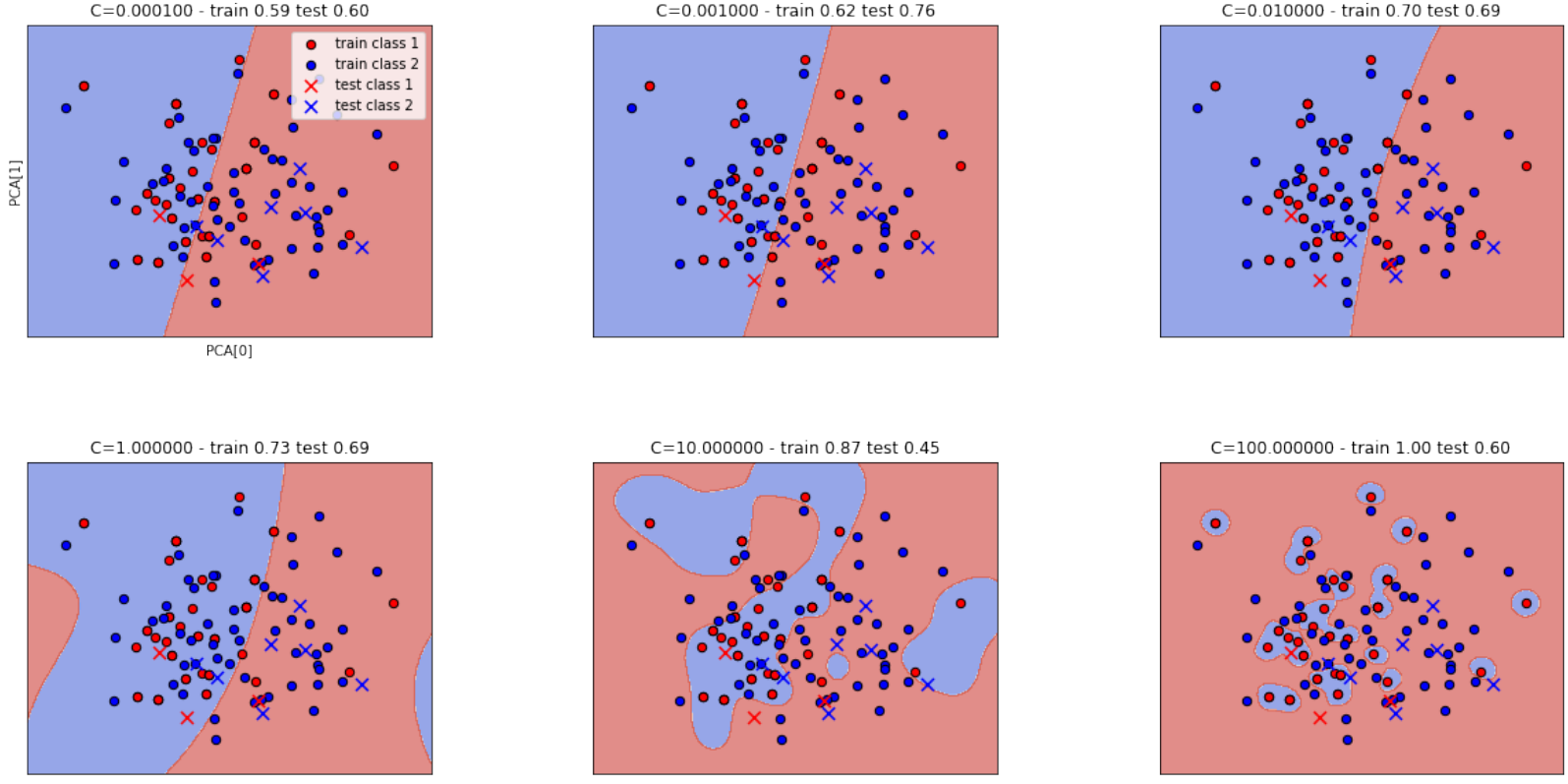
Le classifieur SVM tente de séparer les données par un hyper-plan, avec la contrainte d'avoir une marge minimale (entre l'hyper-plan et les données) qui soit maximale. Le paramètre C détermine un compromis entre maximiser cette marge, et autoriser la mauvaise classification de certains points : plus C est grand, plus les points mal classés sont exclus, et plus la marge aura tendance à être petite.



Application - Prédiction du sentiment de (co)présence

Overfitting: cas du Support Vector Machines (SVM) (2/2)

Pour une valeur de C grande, SVM tentera de classer correctement chaque exemple d'apprentissage, au prix de la marge et possiblement d'une meilleure capacité de généralisation:



Méthodes et techniques adaptées aux petits ensembles de données

- Utiliser des algorithmes empiriquement plus adaptés aux petits ensembles de données (non exhaustif: Random Forests, Naïve Bayes ...)
- Augmenter les données, pour:
  - obtenir un dataset plus grand
  - et/ou réduire les déséquilibres entre classes
- Diminuer l'influence du bruit / le biais, retirer les outliers
  - auditer les exemples d'apprentissage existants
  - régulariser l'apprentissage

Sur un faible volume de données le bruit, les outliers, peuvent avoir un impact important.

Techniques pour la génération de nouvelles données

- Random sampling

Re-sampling (tirage avec remise), utilisé notamment pour résoudre les déséquilibres de classe en répliquant des échantillons de la classe minoritaire.

- SMOTE, ADASYN

Synthèse de nouveaux examples par interpolation d'exemples existants.

- Réseaux de neurones dont les réseaux antagonistes génératifs (GAN)

Un réseau apprend à générer des données, et est corrigé par un réseau apprenant à discriminer vraie donnée et donnée générée (apprentissage souvent difficile).

Application - Prédiction de l'activité cérébrale en fonction des signaux multimodaux

Génération de nouvelles données

- Il serait intéréssant d'ajouter de nouvelles données si cela permet d'améliorer la qualité des prédictions.
- Pour le moment, pour chaque sujet, nous avons 1200 observations, où chaque conversation contient 50 observations.
- Ces observations présentent des auto-corrélations pour la plupart des variables.
- Par conséquent, il faut générer des données de manière à tenir en compte ces auto-corrélations.

Application - Prédiction du sentiment de (co)présence

- GAN
  - peut nécessiter de grands volumes de données
    - approche “fine-tuning”, mais possible seulement si le domaine/la tâche est semblable, compliqué ici
  - qualité / pertinence des données générées difficile à évaluer

Application - Prédiction du sentiment de (co)présence

- Random Sampling, SMOTE, ADASYN, ...
  - Faciles à mettre en oeuvre, variables continues interpolables
  - Variables catégorielles: méthodes pour déterminer la catégorie de la nouvelle donnée (plus proches voisins ...)
  - expérimentalement pas de réel avantage mesuré du moment que les métriques prennent en compte le déséquilibre de classes: