LU3I026 - Science des données

Évaluation et optimisation

Olivier Schwander <olivier.schwander@sorbonne-universite.fr>

Sorbonne Université

2025-2026

Réflexes informatiques

Implémentation

- `Numpy` et `Scipy`
- Orienté objet: abstraction (classe `Classifier` par exemple)
- Modules Python: implémentations complètes

Utilisation, analyses et expériences

- `Pandas`
- `Jupyter`: expériences, analyses et petites implémentations
- Fixer la graine du générateur aléatoire (`seed`)

Sortie des expériences

- Sauvegarder les figures (format vectoriel) et les valeurs
- Avec des noms de fichier intelligents
- Les hyper-paramètres et idéalement les modèles aussi

Évaluation

Aussi important que tout ce qui précède

Comment faire les choix ?

- Plein de méthodes, modèles, algorithmes, hyper-paramètres
- Comment les comparer ?

Est-ce que ça marche ?

- Obtenir des scores ou en tout cas des indicateurs
- Comparer à un cahier des charges
- Est-ce que ça marche *suffisament* bien ?

Est-ce que ça marchera dans la vraie vie ?

- Si on sort du contexte académique ou R&D…
- Évolution des données dans le temps
- Utilisateurs pas forcément coopératifs

Évaluation

Comment faire les choix ?

- Difficile

Est-ce que ça marche ?

- Très difficile

Est-ce que ça marchera dans la vraie vie ?

- Extrèmement difficile

De nos jours

- De plus en plus de choses utilisables en production

Différents types d'évaluation

Évaluation quantitative

- Indicateurs chiffrés, dépendants de la tâche, de l'application
- Classification: accuracy, précision, f1-score
- Détection: vrai/faux positifs/négatifs, précision/spécificité
- Recherche d'information: précision/rappel

Évaluation qualitative

- Typique pour les modèles génératifs: bien ou pas bien
- Manuel, donc coûteux, on analyse seulement quelques sorties

Évaluation quantitative humaine

- Si on a les moyens, beaucoup d'analyses
- Info binaire, ou indicateurs plus évolués
- Critique pour faire marcher des modèles de dialogue

Protocole expérimental

Description de l'évaluation

- Formel, précis, sans ambiguité
- Tous les choix faits pour l'évaluation
- Mais pas les choix de modèle/algo/méthode/etc

Objectifs

- Comparer des scores, et donc des modèles
- Assurer la reproductibilité des expériences
- Valider les choix

OBLIGATOIRE — INDISPENSABLE — VITAL

Sur quoi évaluer

Données d'apprentissage

- C'est tricher
- Erreur méthodologique **TRÈS GRAVE**
- Autant apprendre par cœur
- (score intéressant quand même, mais ça n'est pas de l'évaluation)

Données de test

- Sous-ensemble du jeu de donné
- Réservé pour l'évaluation: interdiction de l'utiliser lors de l'apprentissage
- Utilisé seulement pour le calcul du score

Découpage train/test

Dataset complet

- C'est ce qu'on a au départ
- Exemples + leurs étiquettes

Jeu de train

- Sous-ensemble du datase complet
- Utilisé pour l'apprentissage

Jeu de test

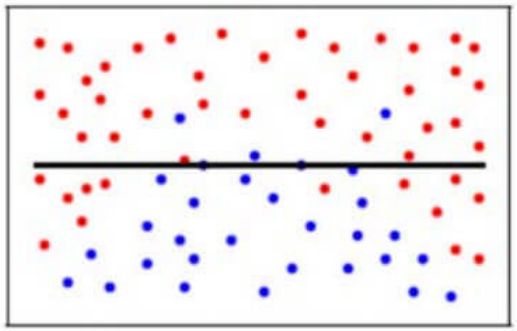
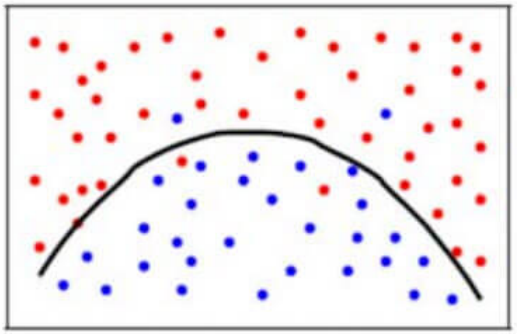
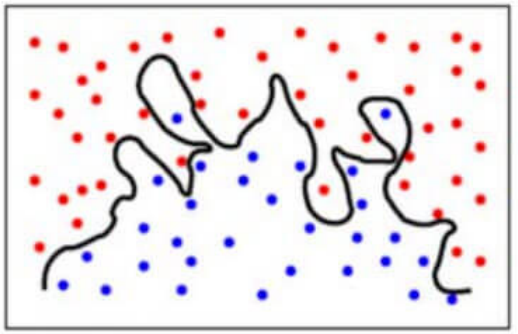
- Sous-ensemble, distinct du précédent
- Utilisé pour le calcule des scores

Découpage fixe

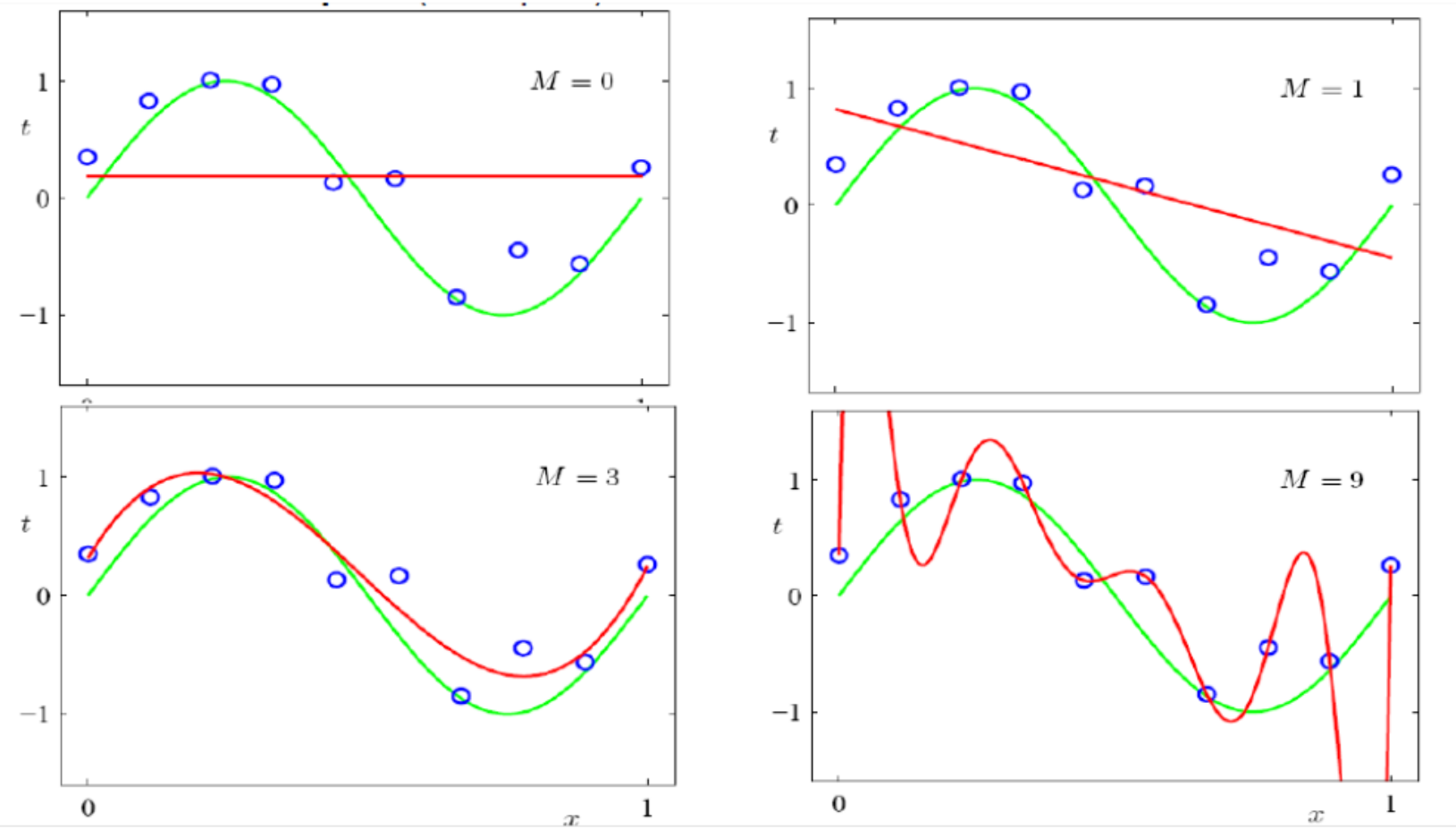
- Par exemple 80% train + 20% test ou 90% train + 10% test

Objectif: généralisation

Quel est le meilleur modèle ?

Objectif: généralisation



Comment savoir ?

Évaluation

- Base de test: données inédites

Sous-apprentissage

- En train: score faible
- En test: score faible
- On a rien appris

Sur-apprentissage

- En train: score élevé
- En test: score faible
- On a appris par cœur

Problème du choix de la base de test

Découpage fixe: 80/20 par exemple

Choix du contenu

- Test facile: généralisation sur-estimée
- Test difficile: génériſation sous-estimée
- Données pas mélangées: une seule classe dans la base de test

Mélange des données

- Absolument indispensable et critique
- Reproductible: stocker le sous-ensemble, ou la graine aléatoire

Autres protocoles de test

Plusieurs sous-ensemble

- Moyenner
- Estimation statisitique de la généralisation

Tous les sous-ensembles ?

- Exponentiel
- Impossible

Validation croisée

[Au tableau]

Mesures de score

Classification

- Binaire: précision (nombre de bonnes réponses / nombre de réponses)
- Multiclasse: accuracy (moyenne des bonnes réponses classe par classe)
- Vrai/Faux positifs/négatifs
- F1-score $\frac{2\mathrm{VP}}{2\mathrm{VP}+\mathrm{FP}+\mathrm{FN}}$

Régression

- Erreur quadratique

Recherche d'information

- Précision et rappel

Optimisation

Problème d'optimisation

$$\min_{\theta} J(\theta)$$

Fonction de coût

- Ce qu'on veut optimiser
- Exemple: traînée d'une aile d'avion

Paramètre

- Ce qu'on fait varier
- Ce qu'on veut récupérer à la fin
- Exemple: forme d'une aile d'avion

Descente de gradient

Minimisation

- Fonction $J : \mathbb{R}^d \to \mathbb{R}$
- Dérivable: ∇J
- Converge vers un minimum local

Algorithme

- Choisir un $\theta_0 \in \mathbb{R}^d$
- **Jusqu'à** la convergence
  - Mise à jour: $\theta_{k+1} \leftarrow \theta_k - \alpha \nabla J(\theta_k)$

Variantes

- Plein

Fonction de coût d'apprentissage

Erreur locale

- Entrée x et étiquette y
- Modèle f et $f(x)$ la prédiction pour l'entrée x
- Erreur locale: erreur faite en essayant de prédire y avec f $\Delta(f(x), y)$

Erreur globale

- Somme des erreurs locales $\sum_i \Delta(f(x_i), y_i)$
- Erreur calculée sur tous les exemples

Choix du Δ

- Descente de **gradient** → dérivable
- Pas vraiment ce qu'on voudrait...

Apprentissage ≠ optimisation

Cadre classique de l'optimisation

- On connaît parfaitement la fonction à optimiser
- Les paramètres sont interprétables
- Les paramètres nous intéressent
- Exemple: traînée et forme d'une aile

Pour l'apprentissage

- On voudrait minimiser l'erreur sur la distribution inconnue (la "réalité")
- On voudrait une erreur binaire: bonne réponse ou mauvaise réponse
- Or on a que la base d'apprentissage...
- Et un Δ dérivable

Descente de gradient stochastique

Cas particulier

- L'erreur est une somme $J(\theta) = \sum_i \Delta(f_\theta(x_i), y_i)$

Nouvel algorithme

- Choisir un $\theta_0 \in \mathbb{R}^d$
- **Jusqu'à** la convergence
  - **Jusqu'à** avoir parcouru tout les exemples
    - Tirer un x_i aléatoire
    - Mise à jour: $\theta_{k+1} \leftarrow \theta_k - \alpha \nabla \Delta(f_\theta(x_i), y_i)$

Version utilisée en apprentissage

- Meilleure convergence en pratique
- Évite de charger tout le dataset en mémoire

Moindre carrés

Loss

- $\Delta(a, b) = (a - b)^2$
- Erreur locale

$$F(\theta) = \sum (f_\theta(x) - y)^2$$

$$\nabla F(\theta) = \sum 2(\langle \theta, x \rangle - y)x$$

Algorithme

- Tirer aléatoirement w
- **Jusqu'à** convergence ou un nombre d'itérations maximum:
  - **Pour** chaque exemple x et son étiquette y:
    $\hat{y} = \langle w, x \rangle$
    $w \leftarrow w + \alpha(y - \hat{y})x$

Ça ne vous rappelle rien ?

Hinge loss

$$\ell(y, f_\theta(x)) = \max(0, 1 - y \cdot f_\theta(x))$$

Intérêt

- Correspond mieux à la loss binaire vrai/faux
- Marge: pénalise les réponses justes mais proches de la frontière

Inconvénient

- Pas dérivable
- Mais dérivée à gauche et à droite

Modèles non-linéaires

Typiquement

- Passage linéaire/affine (ajout de la colonne de 1)
- Polynômes
- Gaussiennes

Réseaux de neurones (deep learning)

- Très grosses fonctions non-linéaires

Modèles polynomiaux

[Au tableau]

Astuce du noyau: kernelisation

[Au tableau]

Importance des pré-traitements

[Au tableau]

Conclusion

Protocole d'évaluation

- À formaliser précisément
- Le plus important, dans la recherche ou dans l'industrie
- Permet de savoir si le système a une chance de marcher
- Objectif: généraliser

Optimisation

- Fondamental pour le perceptron et les réseaux de neurones
- Pas toutes les méthodes
- Mais la quasi-totalité de l'état de l'art
- Besoin d'une fonction de coût dérivable