Sklearn, Généralisation & Overfitting

Jour 2 — Après-midi

Julien Rolland

Formation M2 Développement Fullstack

Jour 2

Plan du module

Le Vrai Objectif : Généraliser

Le Piège

- Minimiser $J(\Theta, X_{\text{train}})$ = problème résolu
- La performance sur le train **ne compte pas**

Définition

Généralisation : capacité à performer sur des exemples jamais vus pendant l'entraînement.

Ce qu'on veut vraiment

- Données **non-vues** (unseen data)
- Capter la loi sous-jacente
- Ignorer le bruit du dataset

Underfitting vs Overfitting

Underfitting

- Modèle trop **simple**
- Passe à côté de la tendance
- Biais élevé

Overfitting

- Modèle trop **complexe**
- Relie les points, délire entre eux
- Variance élevée

Mémorisation vs Généralisation

Théorème de Cover

- Dans $\mathbb{R}^D$, $N \leq D$ points sont toujours linéairement séparables
- Un modèle assez complexe peut **mémoriser** n'importe quel dataset
- Accuracy train $= 100\%$ ne signifie rien

Analogie MNIST

- Mémoriser les pixels exacts de chaque «3»
- Incapable de reconnaître un «3» écrit avec un stylo différent
- $\Rightarrow$ 0% de généralisation

Biais élevé — Underfitting

- Erreur systématique
- Modèle trop rigide (trop peu de paramètres)
- Train error $\approx$ Val error, les deux élevées

Variance élevée — Overfitting

- Sensible aux variations du train
- Train error $\ll$ Val error
- Modèle trop riche (trop de paramètres)

Levier : les Hyperparamètres

- Profondeur d'arbre, nombre de neurones
- Coefficient de régularisation λ
- Taille du dataset

3 ensembles distincts

1. **Train set** — ajuster W, b
2. **Validation set** — choisir les hyperparamètres
3. **Test set** — coffre-fort, ouvert **une seule fois** à la fin

Règle d'or : Data Leakage

Ne **jamais** entraîner sur le test set.
Ne **jamais** utiliser le test pour choisir les hyperparamètres.
Le test set doit rester **invisible**.

K-Fold Cross-Validation

Le Problème

- Peu de données $\Rightarrow$ split instable
- Un seul val set $\Rightarrow$ variance élevée de l'estimation

Solution : K-Fold

- Découper le train en K morceaux (*folds*)
- K rotations : chaque fold sert de validation
- Score final = moyenne sur K runs
- Utilisation maximale des données disponibles

L'API Estimator de sklearn

Interface universelle

- `.fit(X, y)` — entraînement
- `.predict(X)` — inférence
- `.score(X, y)` — évaluation

Modèles clés

- `LogisticRegression` (softmax)
- `StandardScaler`
- `cross_val_score`

Pipeline

- Enchaîner Scaler → Modèle
- Évite le Data Leakage sur le scaling
- `Pipeline([('scaler', StandardScaler()), ('clf', LogisticRegression())])`

Dataset MNIST

- 70 000 images 28×28 ($= 784$ features)
- 10 classes (chiffres 0–9)
- Split standard : 60 000 train / 10 000 test

Baseline

- `LogisticRegression` (Softmax)
- Normalisation `StandardScaler`
- Métriques : accuracy, matrice de confusion

Observer l'Overfitting

- Entraîner sur $N = 500$ images
- Train accuracy $\approx 100\%$, Test accuracy $\approx 60\%$
- Le modèle mémorise

Remèdes

- **Plus de données** — augmenter N
- **Régularisation L2** (`C` dans sklearn) — pénalise les grands poids
- **Régularisation L1** — sparse weights

Matrice de Confusion

- Quels chiffres sont confondus ?
- Ex. : 4 vs 9, 3 vs 8, 5 vs 6
- Orienter les efforts d'amélioration

Aller plus loin

- `cross_val_score` pour estimer la variance
- Grid search sur C (régularisation)