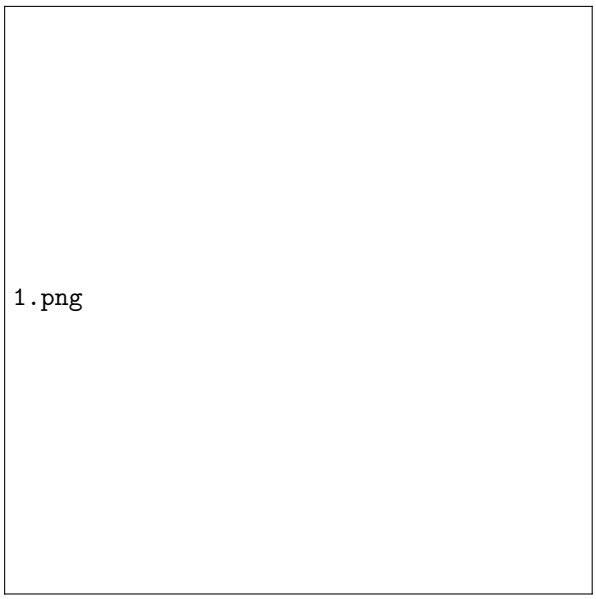
APprentissage STAtistique
2024-2025

Introduction

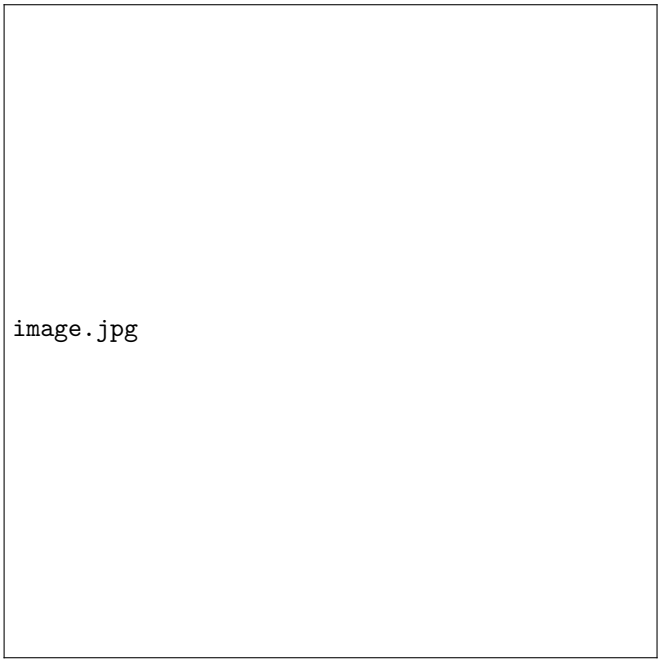
L'intelligence artificielle (IA) représente la théorie et le développement de systèmes informatiques capables d'effectuer des tâches nécessitant normalement l'intelligence humaine.

L'apprentissage automatique, c'est la capacité d'un ordinateur à apprendre sans avoir été explicitement programmé.

0.1 Exemple d'un modèle supervisé :



0.2 La fonction de risque dans l'apprentissage :



Idéalement :

$$f^*_{Bayes} = \arg\min_{f \in \mathcal{F}_{all}} R(f)$$

$$R(f) = \int L(y, f(x)) dP(x, y)$$

Mais f est un certain type de fonction :

$$f^* = \arg\min_{f \in \mathcal{F}} R(f)$$

Cependant, nous n'avons pas accès à la distribution :

$$\hat{f}_{erm} = \arg\min_{f \in \mathcal{F}} R_{erm}(f)$$

$$R_{erm}(f) = \frac{1}{N} \sum_{i=1}^{N} L(y, f(x))$$

0.3 Décomposition biais-variance :

$$\mathbb{E}\left[(\hat{y} - y)^2\right] = \mathbb{E}\left[(\hat{y} - \mathbb{E}[\hat{y}] + \mathbb{E}[\hat{y}] - y)^2\right]$$
$$= \mathbb{E}\left[(\hat{y} - \mathbb{E}[\hat{y}])^2 + 2(\hat{y} - \mathbb{E}[\hat{y}])(\mathbb{E}[\hat{y}] - y) + (\mathbb{E}[\hat{y}] - y)^2\right]$$
$$= \mathbb{E}\left[(\hat{y} - \mathbb{E}[\hat{y}])^2\right] + \mathbb{E}\left[2(\hat{y} - \mathbb{E}[\hat{y}])(\mathbb{E}[\hat{y}] - y)\right] + \mathbb{E}\left[(\mathbb{E}[\hat{y}] - y)^2\right]$$
$$= \mathbb{E}\left[(\hat{y} - \mathbb{E}[\hat{y}])^2\right] + 2(\mathbb{E}[\hat{y}] - y)\mathbb{E}\left[(\hat{y} - \mathbb{E}[\hat{y}])\right] + (\mathbb{E}[\hat{y}] - y)^2$$
$$= \mathbb{E}\left[(\hat{y} - \mathbb{E}[\hat{y}])^2\right] + (\mathbb{E}[\hat{y}] - y)^2$$
$$= \mathrm{var}(\hat{y}) + \mathrm{bias}(\hat{y}, y)^2$$

0.4 Dataset split

- Ensemble d'entraînement : pour adapter les paramètres du modèle.
- Ensemble de validation : pour ajuster la complexité du modèle.
- Ensemble de test : pour évaluer la performance du modèle.

1 Méthodes principales

1.1 Apprentissage supervisé

Utilisé pour des données étiquetées, où l'objectif est de prédire y à partir de x. Il peut être divisé en :

- **Régression**: prédiction de valeurs continues.
- **Classification**: attribution de labels à des exemples.

1.2 Apprentissage non supervisé

Utilisé pour découvrir des structures dans des données non étiquetées, comprenant :

- **Clustering**: regroupement de données similaires.
- **Réduction de dimension**: élimination des caractéristiques non pertinentes.

1.3 Apprentissage par renforcement

Basé sur un agent qui interagit avec un environnement pour maximiser une récompense cumulative, souvent à travers le Q-learning ou les DNN.

1.4 Autres classifications

Par nature :

- **Symbolistes** : Règles et arbres de décision
- **Bayésiens** : Naive Bayes ou Markov
- **Connectionnistes** : Réseaux neuronaux
- **Évolutionnistes** : Programmes génétiques
- **Analogistes** : Vecteurs de support

Par Taxonomie :

- **Discriminatif vs Génératif** :
  - Modèles spécialisés pour distinguer (discriminatif).
  - Modèles capables d'expliquer comment générer de nouvelles données (génératif).
- **Basé instance vs. Basé modèle** :
  - On stocke les données d'entraînement (e.g., K-NN, SVM).
  - On résume les données avec un modèle (e.g., modèles paramétriques).

2 Régression

2.1 Régression linéaire

Objectif : Trouver une fonction linéaire $y = \mathbf{w}^T\mathbf{x}+b$ qui minimise l'erreur quadratique entre les valeurs observées y_i et les prédictions $\hat{y}_i$.

Formulation :

$$L_{\text{SE}} = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \left(\mathbf{w}^T\mathbf{x}_i + b\right)\right)^2$$

où $\mathbf{w}$ est le vecteur des coefficients, et b est le biais. La solution analytique est donnée par la formule des moindres carrés :

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Points forts : Facile à comprendre et rapide pour des petites dimensions.

Points faibles : Sensible aux valeurs aberrantes et nécessite des données linéaires.

2.2 Régression Ridge/Lasso

Le modèle peut être régularisé avec les méthodes Ridge (ℓ_2) ou Lasso (ℓ_2) pour éviter l'overfitting :

$$L_{\text{Ridge}} = L_{\text{SE}} + \lambda\|\mathbf{w}\|_2, \quad L_{\text{Lasso}} = L_{\text{SE}} + \lambda\|\mathbf{w}\|_1$$

La pénalisation ℓ_1 favorise la parcimonie alors que ℓ_2 favorise la dérivabilité.

Démonstration KRR

1. Transformer la fonction de perte

La fonction de perte pour la régression ridge à noyau (KRR) est :

$$L(w) = \frac{1}{2}\|y - \Phi w\|^2 + \frac{\lambda}{2}\|w\|^2$$

On substitue $w = \sum_{i=1}^{n} a_i\phi(x_i) = \Phi^T a$:

$$L(a) = \frac{1}{2}\|y - Ka\|^2 + \frac{\lambda}{2}a^T Ka$$

où $K = \Phi\Phi^T$ et $K_{ij} = k(x_i, x_j)$.

2. Dériver et résoudre pour a

On prend la dérivée de $L(a)$:

$$\frac{\partial L(a)}{\partial a} = -K^T(y - Ka) + \lambda Ka$$

On annule la dérivée :

$$Ky = K(K + \lambda I)a$$

En supposant K inversible :

$$a = (K + \lambda I)^{-1}y$$

3. Prédire pour un nouveau point x_{new}

La prédiction est :

$$f(x_{\text{new}}) = \sum_{i=1}^{n} a_i k(x_i, x_{\text{new}})$$

Sous forme matricielle :

$$f(x_{\text{new}}) = k_{\text{new}}^T a$$

où $k_{\text{new}} = [k(x_1, x_{\text{new}}), \ldots, k(x_n, x_{\text{new}})]^T$.

4. KRR vs SVM

Similitudes :

- Les deux utilisent l'astuce du noyau pour modéliser des relations non linéaires.
- Les deux intègrent une régularisation.
- Les deux reposent sur des formulations duales.

Différences :

- KRR minimise l'erreur quadratique ; SVM minimise la perte hinge.
- Les solutions SVM sont parcimonieuses ; les solutions KRR ne le sont pas.
- SVM a une interprétation géométrique (maximisation de la marge) ; KRR n'en a pas.

3 Classification

3.1 Régression logistique

Objectif : Modéliser la probabilité qu'une observation x appartienne à la classe $y = 1$ via la fonction sigmoïde :

$$P(y = 1|x) = \sigma(\mathbf{w}^T x) = \frac{1}{1 + e^{-(\mathbf{w}^T \mathbf{x} + b)}}$$

Fondement : La régression logistique repose sur la distribution de Bernoulli pour modéliser la probabilité de la classe positive :

$$P(y \mid \mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x})^y (1 - \sigma(\mathbf{w}^T \mathbf{x}))^{1-y}$$

La fonction de coût à minimiser est la log-vraisemblance négative (ou cross-entropy) :

$$L_{\text{CE}} = -\frac{1}{N} \sum_{i=1}^{N} (y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i))$$

Solution : Utilisation de la descente de gradient pour minimiser cette fonction de coût.

Points forts : probabiliste, coût calculatoire faible, facile à implémenter, entraîner et interpréter.

Points faibles : Ne gère pas bien les relations non-linéaires, nécessite de bonnes features.

Extension aux dimensions supérieures : on étend sigmoid à SoftMax et cross-entropy à categorical cross entropy :

$$P(y = k|\mathbf{x}) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$$

$$L_{\text{CE}} = -\sum_{i=1}^{N} \sum_{k=1}^{K} 1_{\{y_i = k\}} \log(P(y_i = k|\mathbf{x}_i))$$

Log-Odds Ratio (réciproque de sigmoid) :

$$\text{logit}(P(y = 1 \mid \mathbf{x})) = \ln \left(\frac{P(y = 1 \mid \mathbf{x})}{P(y = 0 \mid \mathbf{x})} \right)$$

- $w_i > 0$: Les cotes augmentent.
- $w_i < 0$: Les cotes diminuent.
- $w_i = 0$: Les cotes restent inchangées.

Cette interprétation est particulièrement utile pour évaluer l'influence des caractéristiques.

3.2 K-Nearest Neighbors (KNN)

Soit un point de données x et un ensemble d'entraînement $\{(x_i, y_i)\}_{i=1}^n$. Pour prédire la classe ou la valeur de x :

1. Calculer les distances $d(x, x_i)$ (e.g., distance euclidienne).
2. Sélectionner les k plus proches voisins $\mathcal{N}_k(x)$.
3. Pour la classification :

$$\hat{y} = \text{mode}(\{y_i \mid x_i \in \mathcal{N}_k(x)\})$$

4. Pour la régression :

$$\hat{y} = \frac{1}{k} \sum_{x_i \in \mathcal{N}_k(x)} y_i$$

Avantages :

- Solution multi-classes réelle.
- Classificateur intuitif.
- L'entraînement consiste uniquement à stocker l'ensemble d'entraînement.
- Flexibilité dans le choix du voisinage et de la distance.

Inconvénients :

- Besoin en mémoire important.
- Coût computationnel élevé.

3.3 Support Vector Machines (SVM)

Objectif : Trouver un hyperplan optimal qui sépare deux classes en maximisant la marge entre les points les plus proches (vecteurs de support). Pour des données non linéairement séparables, une marge souple (soft margin) est introduite pour tolérer les erreurs de classification.

3.3.1 Hyperplan séparateur optimal

Problème : Maximiser la marge $\frac{1}{\|\mathbf{w}\|}$ tout en classant correctement les données :

$$\min_{\mathbf{w}, w_0} \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{sous} \quad y_i(\mathbf{w}^T \mathbf{x}_i + w_0) \geq 1 \quad \forall i$$

Formulation duale : Le problème est reformulé en termes de multiplicateurs de Lagrange α_i :

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

sous les contraintes :

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \quad \text{et} \quad \alpha_i \geq 0 \quad \forall i$$

Vecteurs de support : Seuls les points avec $\alpha_i > 0$ (vecteurs de support) contribuent à la solution :

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \quad \text{et} \quad w_0 = y_{\text{SV}} - \mathbf{w}^T \mathbf{x}_{\text{SV}}$$

3.3.2 Marge souple (Soft Margin)

Pour les données non séparables, des variables de relaxation ξ_i sont introduites :

$$\min_{\mathbf{w}, w_0, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \xi_i$$

sous les contraintes :

$$y_i(\mathbf{w}^T \mathbf{x}_i + w_0) \geq 1 - \xi_i \quad \text{et} \quad \xi_i \geq 0 \quad \forall i$$

où C contrôle le compromis entre la marge et les erreurs.

3.3.3 Kernel Trick

Pour les données non linéairement séparables, les données sont projetées dans un espace de grande dimension via une fonction $\phi(\mathbf{x})$. Le produit scalaire $\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$ est remplacé par un noyau $K(\mathbf{x}_i, \mathbf{x}_j)$.

Formulation duale avec noyau :

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

sous les mêmes contraintes que précédemment.

Exemples de noyaux :

- Linéaire : $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
- Polynomial : $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^d$
- RBF : $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$

3.3.4 Prédiction

Pour un nouveau point $\mathbf{x}_{\text{new}}$, la décision est donnée par :

$$f(\mathbf{x}_{\text{new}}) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_{\text{new}}) + w_0 \right)$$

3.3.5 Points forts et faibles

Points forts :

- Efficace pour les données linéairement et non linéairement séparables.
- Généralisation robuste grâce à la maximisation de la marge.
- Flexibilité grâce aux noyaux pour modéliser des relations complexes.

Points faibles :

- Choix du noyau et des hyperparamètres (e.g., C, γ) critique.
- Coût computationnel élevé pour les grands jeux de données.
- Limité aux problèmes binaires (nécessite des extensions pour le multi-classe).

3.4 Arbre de Décision

Objectif : construire un arbre qui minimise l'impureté des nœuds tout en maximisant le gain d'information à chaque étape de division.

3.4.1 Structure d'un Nœud

Un nœud v dans l'arbre est défini par une fonction de décision f_v et un seuil τ_v. La fonction de décision est généralement linéaire :

$$f_v(\mathbf{x}) = \mathbf{w}_v^T \mathbf{x},$$

où $\mathbf{w}_v \in \mathbb{R}^{D+1}$ est le vecteur de poids et $\tau_v \in \mathbb{R}$ est le seuil. Le point de données $\mathbf{x}$ est dirigé vers le nœud enfant gauche si $f_v(\mathbf{x}) < \tau_v$ et vers le nœud enfant droit sinon.

3.4.2 Nœuds Feuilles

Un nœud feuille correspond à une cellule C_z dans la partition. Les probabilités a posteriori des classes dans C_z sont estimées comme suit :

$$p(k|\mathbf{x}) = \frac{|\{\mathbf{x}_i \in C_z : y_i = k\}|}{|C_z|},$$

où $|C_z|$ est le nombre de points dans C_z, et k est l'étiquette de classe.

3.4.3 Entraînement

L'arbre est construit récursivement en optimisant la division à chaque nœud. La qualité d'une division S est mesurée à l'aide de métriques d'impureté telles que l'indice de Gini ou l'entropie :

$$\text{Gini}(S) = 1 - \sum_{k=1}^{K} p(k|\mathbf{x})^2,$$

$$\text{Entropie}(S) = -\sum_{k=1}^{K} p(k|\mathbf{x}) \log p(k|\mathbf{x}).$$

Le gain d'information est utilisé pour choisir la meilleure division :

$$\text{Gain d'Information}(S) = H_q - \frac{N_{\text{gauche}}}{N_q} H_{\text{gauche}} - \frac{N_{\text{droit}}}{N_q} H_{\text{droit}},$$

où H_q est l'entropie du nœud parent, H_{gauche} et H_{droit} sont les entropies des nœuds enfants, et N_{gauche} et N_{droit} sont le nombre de points dans les nœuds enfants.

3.4.4 Algorithme CART

L'algorithme CART (Classification and Regression Trees) utilise une optimisation aléatoire des nœuds pour générer des candidats de division. Les étapes clés sont :

1. Générer plusieurs candidats de division aléatoires.
2. Choisir la meilleure division en fonction de la mesure de qualité (Gini ou gain d'information).
3. Répéter le processus jusqu'à ce que les critères d'arrêt soient atteints (profondeur maximale, nombre minimal de points, etc.).

3.4.5 Prédiction

Pour un point de test $\mathbf{x}$, il est parcouru dans l'arbre jusqu'à atteindre un nœud feuille. La classe prédite est :

$$\hat{y} = \arg\max_{k \in \mathcal{K}} p(k|\mathbf{x}).$$

3.4.6 Régression

Dans les tâches de régression, la valeur prédite pour une région C_z est la moyenne des valeurs cibles des instances dans cette région :

$$\hat{y} = \frac{1}{|C_z|} \sum_{\mathbf{x}_i \in C_z} y_i.$$

La qualité de la division est mesurée en minimisant l'erreur quadratique moyenne (MSE) :

$$\text{MSE}(S) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2.$$

3.4.7 Avantages et Inconvénients

- **Avantages :** Apprentissage et évaluation rapides, utilisation efficace de la mémoire, et gestion naturelle des problèmes multi-classes.
- **Inconvénients :** Sensible au surapprentissage avec des arbres profonds, et la performance peut varier significativement avec la profondeur de l'arbre et la qualité des divisions.

3.5 Méthodes d'Ensemble

Les méthodes d'ensemble combinent plusieurs modèles d'apprentissage faible pour améliorer les performances globales. Elles visent à réduire le biais, la variance et à améliorer la précision des prédictions.

3.5.1 Stratégies Principales

- **Bagging (Bootstrap Aggregating) :** Entraîne plusieurs modèles sur des sous-ensembles bootstrap de l'ensemble d'entraînement et agrège leurs prédictions (par exemple, par vote majoritaire).
- **Boosting :** Entraîne séquentiellement des modèles faibles en ajustant les poids des exemples mal classés pour améliorer les performances.
- **Stacking :** Combine les prédictions de plusieurs modèles de base en utilisant un méta-modèle pour apprendre la meilleure combinaison.

3.6 Forêts Aléatoires

Les forêts aléatoires sont une méthode d'ensemble basée sur le bagging, utilisant des arbres de décision comme modèles de base. Elles introduisent de l'aléatoire dans la construction des arbres pour améliorer la généralisation.

3.6.1 Entraînement

1. Créer T sous-ensembles bootstrap de l'ensemble d'entraînement.
2. Entraîner un arbre de décision sur chaque sous-ensemble, en sélectionnant aléatoirement un sous-ensemble de caractéristiques à chaque division.
3. Répéter jusqu'à obtenir T arbres.

3.6.2 Prédiction

Pour un point de test $\mathbf{x}$, les prédictions de tous les arbres sont agrégées. Les règles d'agrégation courantes incluent :

- **Vote majoritaire :** $\hat{y} = \arg\max_{k \in \mathcal{K}} \sum_{t=1}^{T} \mathbb{I}(\hat{y}_t = k)$
- **Moyenne :** $\hat{y} = \frac{1}{T} \sum_{t=1}^{T} \hat{y}_t$

3.6.3 Avantages et Inconvénients

- **Avantages :** Réduction du surapprentissage, gestion naturelle des problèmes multi-classes, et performances élevées.
- **Inconvénients :** Complexité accrue et temps de calcul plus long par rapport à un seul arbre.

4 Clustering

4.1 K-Means Clustering

Objectif : Partitionner N points de données en K clusters en minimisant la variance intra-cluster :

$$\min_{\mathbf{C}} \sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - \mu_k\|^2$$

où μ_k est le centre du cluster C_k, et la minimisation est effectuée par des étapes itératives d'assignation des points aux clusters et recalcul des centres de clusters.

Solution : Algorithme itératif avec deux étapes principales : assignation des points aux clusters et mise à jour des centres de clusters.

Points forts : Simple à comprendre, rapide pour des jeux de données modérés.

Points faibles : Sensible aux initialisations (risque de convergence vers des solutions locales), choix du K.

5 Méthodes de Sélection de Caractéristiques

5.1 Méthodes de Filtrage

Objectif : Sélectionner les caractéristiques en fonction de leur pertinence statistique, indépendamment du modèle de prédiction.

Approche :

- **Corrélation de Pearson** : Mesure la corrélation linéaire entre une caractéristique x_i et la cible y :

$$\text{Corr}(x_i, y) = \frac{\text{Cov}(x_i, y)}{\sigma_{x_i} \sigma_y}$$

- **Test F** : Mesure la ratio de variance inter-classe sur variance intra-classe pour une caractéristique x_i :

$$F(x_i) = \frac{\text{Var}_{\text{inter-classe}}(x_i)}{\text{Var}_{\text{intra-classe}}(x_i)}$$

- **Information Mutuelle (MI)** : Mesure la dépendance statistique entre x_i et y :

$$\text{MI}(x_i, y) = \sum_{x_i, y} p(x_i, y) \log \frac{p(x_i, y)}{p(x_i)p(y)}$$

Avantages :

- Faible coût computationnel.
- Peu de risque de surajustement (overfitting).

Inconvénients :

- Ignore les interactions entre caractéristiques.
- Ne tient pas compte du modèle de prédiction.

5.2 Méthodes Embarquées

Objectif : Intégrer la sélection de caractéristiques directement dans le processus d'apprentissage du modèle.

Approche :

- **Régression Ridge** : Ajoute une pénalité L_2 sur les coefficients $\mathbf{w}$:

$$\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2$$

- **Régression Lasso** : Ajoute une pénalité L_1 sur les coefficients $\mathbf{w}$ pour favoriser la parcimonie :

$$\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_1$$

- **Elastic Net** : Combine les pénalités L_1 et L_2 :

$$\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda_1 \|\mathbf{w}\|_1 + \lambda_2 \|\mathbf{w}\|_2^2$$

Avantages :

- Interaction directe avec le modèle.
- Moins de risque de surajustement que les méthodes de wrapper.

Inconvénients :

- Coût computationnel plus élevé que les méthodes de filtrage.
- Dépend du modèle utilisé.

5.3 Méthodes de Wrapper

Objectif : Évaluer des sous-ensembles de caractéristiques en utilisant un modèle de prédiction pour mesurer leur performance.

Approche :

- **Sélection Séquentielle** :
  - **Forward Selection** : Ajoute itérativement la caractéristique qui améliore le plus la performance.
  - **Backward Elimination** : Supprime itérativement la caractéristique qui dégrade le moins la performance.
- **Élimination Récurrente de Caractéristiques (RFE)** : Supprime itérativement les caractéristiques les moins importantes selon un modèle (e.g., SVM, régression).

Avantages :

- Prend en compte les interactions entre caractéristiques.
- Adapté au modèle de prédiction utilisé.

Inconvénients :

- Coût computationnel élevé.
- Risque de surajustement.

5.4 Comparaison des Méthodes

- **Filter Methods** : Rapides, indépendantes du modèle, mais ignorent les interactions.
- **Embedded Methods** : Équilibre entre performance et coût, intégrées au modèle.
- **Wrapper Methods** : Performantes mais coûteuses, adaptées au modèle.

6 Réduction de dimensionalité

6.1 Analyse en Composantes Principales (PCA)

Objectif : Projeter les données sur un sous-espace de dimension réduite en maximisant la variance des données projetées.

Formulation :

- Soit $\mathbf{X} \in \mathbb{R}^{N \times D}$ la matrice de données centrée (moyenne nulle).
- La matrice de covariance est $\mathbf{C} = \frac{1}{N}\mathbf{X}^T\mathbf{X}$.
- L'ACP cherche les vecteurs propres $\mathbf{v}_i$ de $\mathbf{C}$ associés aux plus grandes valeurs propres λ_i.
- Les composantes principales sont les vecteurs propres $\mathbf{v}_i$, et la projection des données est donnée par :
$$\mathbf{X}' = \mathbf{X}\mathbf{V}_d$$
où $\mathbf{V}_d$ contient les d premiers vecteurs propres.

Choix de la dimension d : Sélectionner d tel que :
$$\frac{\sum_{i=1}^{d} \lambda_i}{\sum_{i=1}^{D} \lambda_i} \geq \tau$$
où τ est un seuil de variance conservée (e.g., 95%).

6.2 Analyse Discriminante Linéaire (LDA)

Objectif : Trouver une projection qui maximise la séparation entre les classes tout en minimisant la variance intra-classe.

Formulation :

- Soit $\mathbf{S}_B$ la matrice de dispersion inter-classe :
$$\mathbf{S}_B = \sum_{c=1}^{C} N_c(\mathbf{m}_c - \mathbf{m})(\mathbf{m}_c - \mathbf{m})^T$$
où $\mathbf{m}_c$ est la moyenne de la classe c, et $\mathbf{m}$ la moyenne globale.
- Soit $\mathbf{S}_W$ la matrice de dispersion intra-classe :
$$\mathbf{S}_W = \sum_{c=1}^{C} \sum_{\mathbf{x} \in \mathcal{C}_c} (\mathbf{x} - \mathbf{m}_c)(\mathbf{x} - \mathbf{m}_c)^T$$
- Le critère de Fisher est :
$$J(\mathbf{w}) = \frac{\mathbf{w}^T\mathbf{S}_B\mathbf{w}}{\mathbf{w}^T\mathbf{S}_W\mathbf{w}}$$
- La solution est donnée par les vecteurs propres de $\mathbf{S}_W^{-1}\mathbf{S}_B$.

6.3 t-SNE (t-Distributed Stochastic Neighbor Embedding)

Objectif : Projeter les données dans un espace de faible dimension en préservant les similarités locales.

Formulation :

- Les similarités dans l'espace original sont modélisées par une distribution gaussienne :
$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2/2\sigma_i^2)}$$
- Les similarités dans l'espace réduit sont modélisées par une distribution de Student :
$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}}$$
- L'objectif est de minimiser la divergence de Kullback-Leibler :
$$\mathrm{KL}(P\|Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

Paramètre : La perplexité contrôle le nombre effectif de voisins considérés.

6.4 Autres Méthodes

- **MDS (Multidimensional Scaling)** : Préserve les distances entre les points.
- **ICA (Independent Component Analysis)** : Trouve une transformation linéaire telle que les composantes soient statistiquement indépendantes.

6.5 Comparaison des Méthodes

- **ACP** : Linéaire, globale, sensible aux outliers.
- **LDA** : Supervisée, optimisée pour la séparation des classes.
- **t-SNE** : Non linéaire, adaptée à la visualisation, coûteuse en calcul.

7 Deep Learning

7.1 Réseaux de Neurones Convolutifs

7.1.1 Composants des CNN

- **Couches Convolutives** :
  - Appliquent des filtres convolutifs pour extraire des caractéristiques locales.
  - Poids partagés et connectivité locale réduisent le nombre de paramètres.
  - Formule de convolution pour un filtre $\mathbf{w}$:
    $$h_i = (\mathbf{x} * \mathbf{w})_i + b$$
    où $\mathbf{x}$ est l'entrée, $\mathbf{w}$ le filtre, et b le biais.
- **Couches de Pooling** :
  - Réduisent la dimension spatiale (e.g., max-pooling, average-pooling).
  - Max-pooling : $h_i = \max(\mathbf{x}_{\text{fenêtre}})$.
  - Augmente l'invariance aux translations.
- **Couches d'Activation** :
  - Introduisent de la non-linéarité (e.g., ReLU : $f(x) = \max(0, x)$).
- **Couches Fully Connected (FC)** :
  - Chaque neurone est connecté à tous les neurones de la couche précédente.
  - Peuvent être vues comme des convolutions 1×1 avec des filtres de la taille de l'entrée.

7.1.2 Architectures Classiques

- **LeNet (1998)** :
  - Première architecture CNN pour la reconnaissance de chiffres manuscrits.
  - Utilise des convolutions 5×5, du pooling non chevauchant, et des couches FC.
- **AlexNet (2012)** :
  - Popularise les CNN avec des convolutions 11×11, ReLU, et dropout.
- **VGG (2014)** :
  - Utilise des blocs de convolutions 3×3 empilées pour une meilleure expressivité.
- **GoogleNet (Inception, 2015)** :
  - Introduit des modules Inception avec des convolutions 1×1 pour réduire la dimensionnalité.
- **ResNet (2015)** :
  - Utilise des connexions résiduelles pour entraîner des réseaux très profonds.
  - Formule d'un bloc résiduel :
    $$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{\mathbf{W}_i\}) + \mathbf{x}$$
    où $\mathcal{F}$ est une fonction de transformation.

7.2 Optimisation des Réseaux de Neurones Profonds

7.2.1 Hyperparamètres

- **Taille du lot (Batch Size)** :
  - Affecte la stabilité et la vitesse de convergence.
- **Taux d'apprentissage (Learning Rate)** :
  - Contrôle la taille des pas dans la descente de gradient.
  - Trop élevé : divergence. Trop faible : convergence lente.
- **Planification du Taux d'Apprentissage** :
  - Réduire le taux d'apprentissage au fil du temps (e.g., décroissance exponentielle).

7.2.2 Initialisation des Poids

- **Xavier/Glorot Initialization** :
  - Pour des activations tanh : $W \sim \mathcal{N}(0, \frac{1}{n_{\text{in}}})$.
- **He Initialization** :
  - Pour des activations ReLU : $W \sim \mathcal{N}(0, \frac{2}{n_{\text{in}}})$.

7.2.3 Régularisation

- **Dropout** :
  - Désactive aléatoirement des neurones pendant l'entraînement pour éviter le surajustement.
  - Probabilité de dropout p (e.g., $p = 0.5$).
- **Weight Decay** :
  - Ajoute une pénalité L_2 sur les poids :
    $$E(\mathbf{w}) = E_{\text{data}}(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

7.2.4 Normalisation

- **Batch Normalization (BN)** :
  - Normalise les activations par lot pour stabiliser l'entraînement.
  - Formule :
    $$h' = \frac{h - \mu_B}{\sigma_B}, \quad h'' = \gamma h' + \beta$$
    où μ_B et σ_B sont la moyenne et l'écart-type du lot, γ et β sont des paramètres appris.

7.2.5 Méthodes d'Optimisation

- **SGD avec Momentum** :
  - Accumule une vitesse pour accélérer la convergence :

$$\mathbf{v}^{(t)} = \gamma \mathbf{v}^{(t-1)} + \lambda \nabla E(\mathbf{w}^{(t)})$$

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \mathbf{v}^{(t)}$$

- **Adam** :
  - Combine momentum et adaptation du taux d'apprentissage :

$$\mathbf{v}^{(t)} = \gamma_1 \mathbf{v}^{(t-1)} + (1-\gamma_1)\nabla E(\mathbf{w}^{(t)})$$

$$\mathbf{s}^{(t)} = \gamma_2 \mathbf{s}^{(t-1)} + (1-\gamma_2)(\nabla E(\mathbf{w}^{(t)}))^2$$

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \frac{\lambda}{\sqrt{\mathbf{s}^{(t)}}}\mathbf{v}^{(t)}$$

7.3 Lutte contre le Surajustement

- **Augmentation des Données** :
  - Applique des transformations géométriques / photométriques pour diversifier les données.
- **Transfer Learning** :
  - Utilise un modèle pré-entraîné et adapte les dernières couches au nouveau problème.

7.4 Courbes d'Apprentissage

- **Surajustement (Overfitting)** :
  - Écart important entre les pertes d'entraînement et de validation.
  - Solutions : régularisation, dropout, early stopping.
- **Sous-ajustement (Underfitting)** :
  - Perte d'entraînement et de validation élevées.
  - Solutions : augmenter la capacité du modèle, entraîner plus longtemps.

8 Métriques d'évaluation

8.1 Métriques

- **TP** : Nombre de prédictions correctes positives.
- **TN** : Nombre de prédictions correctes négatives.
- **FP** : Nombre de prédictions positives incorrectes.
- **FN** : Nombre de prédictions négatives incorrectes.
- Accuracy $= \frac{TP+TN}{TP+FP+TN+FN}$
- Error rate $= 1 -$ Accuracy
- Sensitivity (recall) $= \frac{TP}{TP+FN}$
- Specificity $= \frac{TN}{TN+FP}$
- Precision $= \frac{TP}{TP+FP}$
- FNR $= 1 -$ Sensitivity
- FPR $= 1 -$ Specificity

Confusion matrix

Prediction	Ground Truth	
	Class A	Class B
Class A	True positive	False positive Type I Error (α)
Class B	False negative Type II Error (β)	True negative

8.2 ROC

Compare pour tous les seuils :

- **recall/TPR/sensitivity** (y-axis)
- **false positive rate** (x-axis)
- **Chaque point** sur la courbe est calculé à partir d'une matrice de confusion, générée avec un seuil différent.

Caractéristiques

- visualise la balance entre l'objectif (sensitivity) et le coût (FPR)
- La ligne allant du coin inférieur gauche au coin supérieur droit représente un **classifieur aléatoire**.
- Un **classifieur parfait** est représenté par la ligne passant par les points (0,0), (0,1), et (1,1).
- Le **point indiquant les meilleures performances** sur le graphe est celui le plus proche du coin supérieur gauche.

Limites :

- Les courbes ROC ne fonctionnent pas pour les données déséquilibrées.
- Les courbes ROC ne sont pas adaptées à la détection d'objets (on ne peut pas compter les TN).
- Les courbes Precision-Recall sont calculées de manière similaire aux courbes ROC.
- Les courbes Precision-Recall visualisent le compromis entre les faux négatifs et les faux positifs.