Introduction au Machine Learning
Deep Learning
Natural Language Processing

Formateurs :

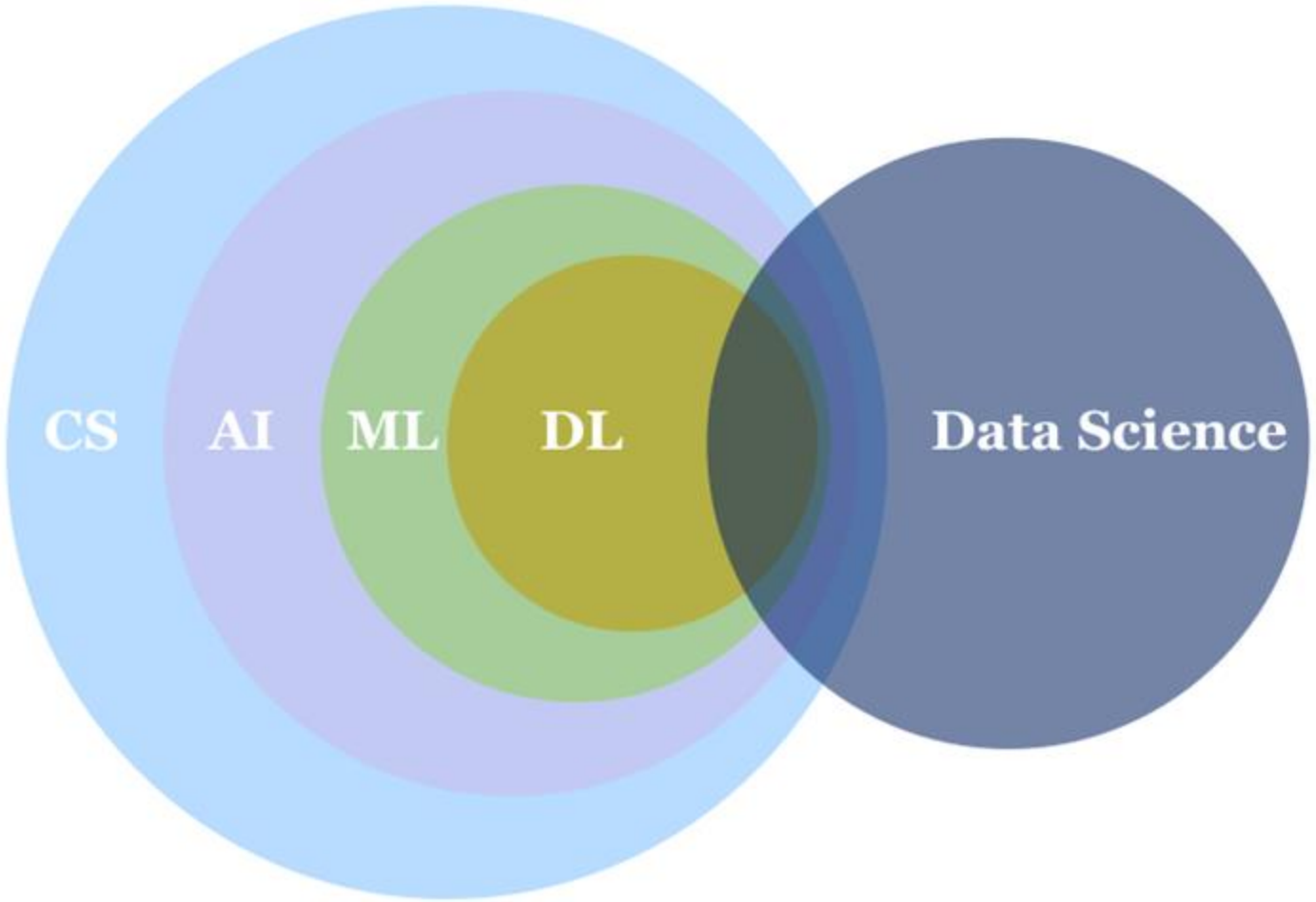
Albérick Euraste DJIRE
Maïmouna OUATTARA

Agenda

- Fondamentaux de l'Intelligence Artificielle et du Machine Learning
- Machine Learning : Apprentissage Supervisé
- Machine Learning : Apprentissage Non-Supervisé
- Deep Learning
- Natural Language Processing

Fondamentaux de l’Intelligence Artificielle et du Machine Learning

Taxonomie de l'IA



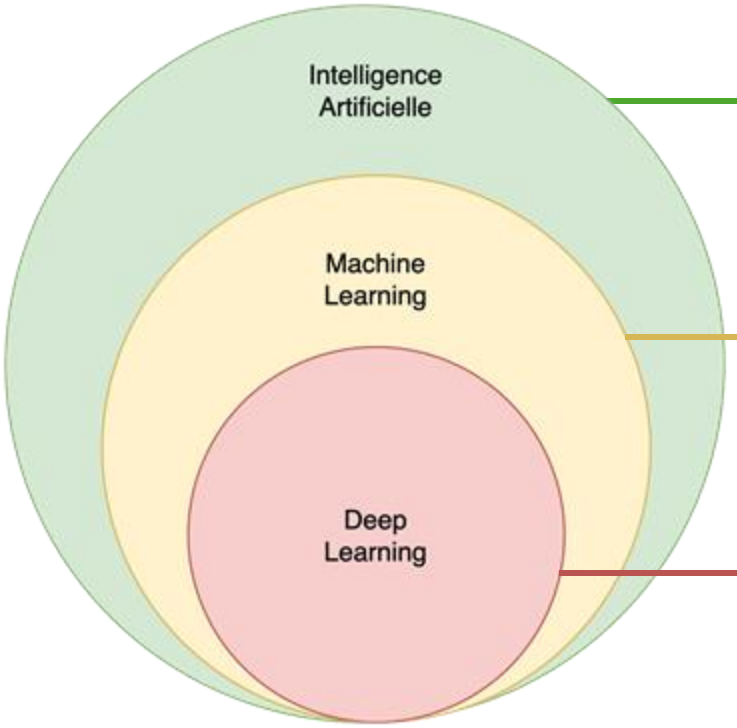
CS: Computer science

AI: Artificial Intelligence

ML: Machine Learning

DL: Deep Learning

Définitions IA – ML - DL



Capacité d'une machine à imiter le comportement humain

Application de l'IA qui permet à un systeme d'apprendre automatiquement et de s'améliorer au fur et à mesure

Application du Machine Learning qui utilise des algorithme complexes et des réseaux de neurones profonds pour entrainer des modèles

Paradigme d’apprentissage

🎯 Apprentissage supervisé

- **Principe** : Apprendre à partir de données étiquetées (paires entrée-sortie)
- **Exemples** : Classification d'emails (spam/non-spam), prédiction de prix immobiliers
- **Algorithmes** : Régression linéaire, SVM, Random Forest, Réseaux de neurones

🔍 Apprentissage non-supervisé

- **Principe** : Découvrir des structures cachées dans des données non étiquetées
- **Exemples** : Segmentation de clients, détection d'anomalies, réduction de dimension
- **Algorithmes** : K-means, PCA, Autoencodeurs

🕹️ Apprentissage par renforcement

- **Principe** : Un agent apprend à prendre des décisions par essai-erreur en maximisant une récompense
- **Exemples** : Jeux (AlphaGo), robotique, véhicules autonomes
- **Algorithmes** : Q-Learning, Policy Gradient, Actor-Critic

Pipeline d'un projet de Machine Learning

Collecte de données → Nettoyage de données → Feature Engineering → Entrainement → Validation → Déploiement

Collecte et néttoyage des données

- ✓ Identification des sources de données
- ✓ Gestion des valeurs manquantes
- ✓ Suppression des doublons
- ✓ Détection des valeurs aberrantes
- ✓ Validation des données

Ingénierie des caractéristiques et entraînement

- ✓ Sélection des caractéristiques pertinentes
- ✓ Normalisation/mise à l'échelle des données
- ✓ Division des ensembles d'entraînement et de test
- ✓ Entraînement du modèle
- ✓ Optimisation des hyperparamètres

Validation et déploiement

- ✓ Validation croisée
- ✓ Indicateurs de performance
- ✓ Analyse des erreurs
- ✓ Déploiement du modèle
- ✓ Surveillance

Applications de l'IA

Text to Speech /
Speech to Text

Cartographie

Reconnaisance facial /
Detection d'objet

Analyse et recommandation

Traduction

Robotique

Régression

Objectif : Prédire une valeur continue

Exemples :

- Prix d'un bien immobilier
- Température future
- Chiffre d'affaires

Algorithmes : Régression linéaire, régression polynomiale, SVR, réseaux de neurones

Type de modèle

Objectif : Assigner une catégorie/classe à une observation

Types :

- Binaire : 2 classes (spam/non-spam, malade/sain)
- Multi-classe : >2 classes exclusives (chiffre manuscrit 0-9)
- Multi-label : Plusieurs classes simultanées (tags d'un article)

Algorithmes : Régression logistique, SVM, Random Forest, CNN

Clustering

Objectif : Regrouper des observations similaires sans labels préexistants

Exemples :

- Segmentation de clients
- Groupement de documents similaires
- Détection de communautés dans un réseau social

Algorithmes : K-means, DBSCAN, Hierarchical Clustering

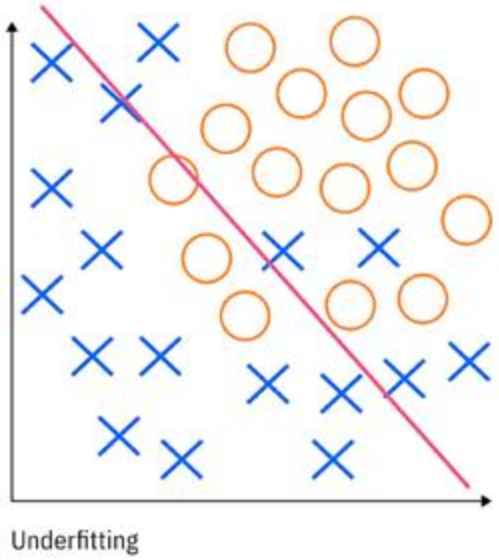
Overfitting vs Underfitting

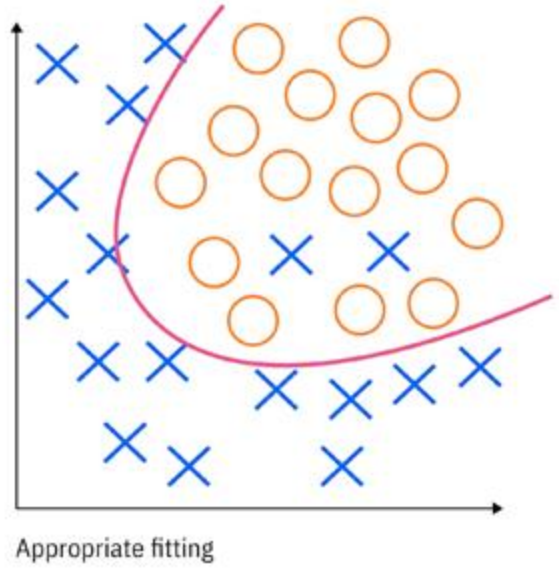
Underfitting (sous-apprentissage)

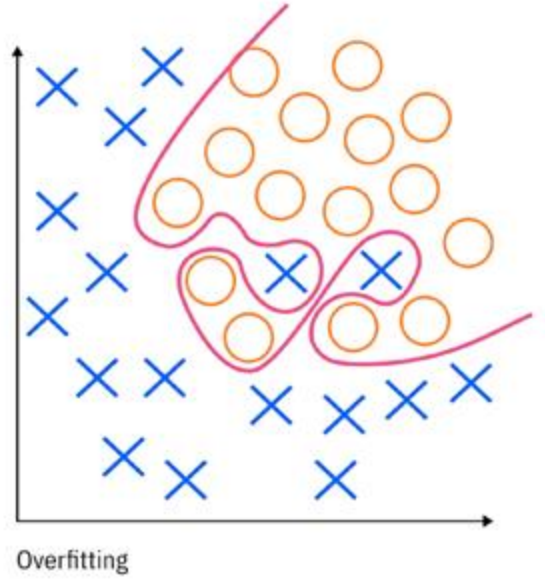
- Le modèle est trop simple
- Ne capture pas les patterns des données
- Biais élevé
- Solution : modèle plus complexe, plus de features

Overfitting (sur-apprentissage)

- Le modèle "mémorise" les données d'entraînement
- Capture le bruit au lieu du signal
- Variance élevée
- Solution : régularisation, plus de données, simplifier le modèle







[1] https://www.ibm.com/think/topics/overfitting-vs-underfitting

Bias-Variance Tradeoff

Biais (Bias) : Erreur due à des hypothèses simplificatrices du modèle

- ✓ Biais élevé → underfitting
- ✓ Le modèle rate systématiquement certains patterns

Variance : Sensibilité du modèle aux fluctuations dans les données d'entraînement

- ✓ Variance élevée → overfitting
- ✓ Le modèle change drastiquement avec de nouvelles données

Compromis : Il faut trouver l'équilibre optimal

$$Erreur\ total = Biais^2 + Variance + Bruit\ irréductible$$

[PLACEHOLDER: Graphique en U montrant l'évolution du biais et de la variance en fonction de la complexité du modèle]

Cross-Validation

Définition : Technique pour évaluer la capacité de généralisation d'un modèle en utilisant plusieurs divisions différentes des données.

K-Fold Cross-Validation

```
Fold 1: [Test][Train][Train][Train][Train]
Fold 2: [Train][Test][Train][Train][Train]
Fold 3: [Train][Train][Test][Train][Train]
Fold 4: [Train][Train][Train][Test][Train]
Fold 5: [Train][Train][Train][Train][Test]
```

Processus :

1. Diviser les données en K parties égales
2. Pour chaque fold, entraîner sur K-1 parties et tester sur la partie restante
3. Calculer la moyenne des performances sur tous les folds

Avantages :

- Utilisation efficace des données
- Estimation plus robuste de la performance
- Réduit le risque de biais lié à une division particulière

Mesures de performance - Régression

Mean Absolute Error (MAE)

$$MAE = \frac{1}{n}\sum |y_i - \hat{y}_i|$$

- ✓ Facile à interpréter
- ✓ Même unité que la variable cible

Root Mean Squared Error (RMSE)

$$RMSE = \sqrt{\left[\frac{1}{n}\sum (y_i - \hat{y}_i)^2\right]}$$

- ✓ Pénalise davantage les grandes erreurs
- ✓ Sensible aux outliers

R^2 (Coefficient de détermination)

$$R^2 = 1 - \left(\frac{SS_{res}}{SS_{tot}}\right)$$

- ✓ Varie entre 0 et 1 (ou négatif si très mauvais)
- ✓ Indique la proportion de variance expliquée

Mesures de performance - Classification

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- ✓ Simple mais peut être trompeuse avec des classes déséquilibrées

$$Precision = \frac{TP}{TP + FP}$$

- ✓ "Parmi les prédictions positives, combien sont correctes ?"
- ✓ Important quand le coût des faux positifs est élevé

$$Recall = \frac{TP}{TP + FN}$$

- ✓ "Parmi tous les positifs réels, combien ont été détectés ?"
- ✓ Important quand le coût des faux négatifs est élevé

$$F1_{score} = 2 * \frac{Precision\ *.Recall}{Precision+.Recall}$$

- ✓ Moyenne harmonique de précision et rappel
- ✓ Utile pour les données déséquilibrées

Matrice de confusion

	Vraiment Positif	Vraiment Négatif
Prédiction Positif	Vrai Positif TP	Faux Positif FP
Prédiction Négatif	Faux Négatif FN	Vrai Négatif TN

Mesures de performance - Classification

Courbe ROC (Receiver Operating Characteristic)

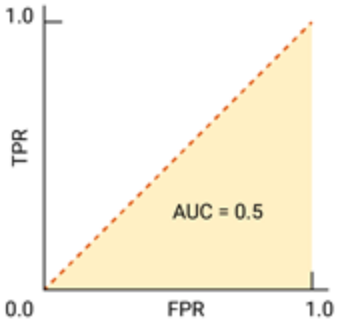
- ✓ Représente le taux de vrais positifs vs faux positifs
- ✓ AUC (Area Under Curve) : métrique agrégée (0.5 = aléatoire, 1.0 = parfait)

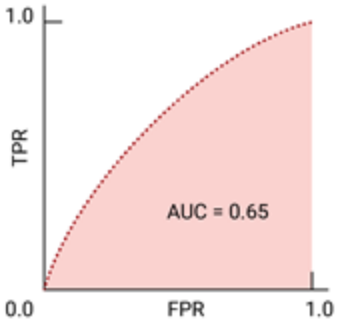
$$Spécificité = \frac{TN}{TN + FP}$$

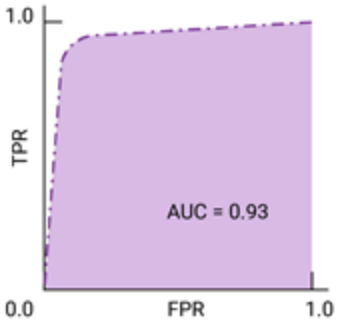
- ✓ Capacité à identifier correctement les négatifs

Choix de la métrique :

- ✓ **Accuracy** : classes équilibrées
- ✓ **F1-Score** : classes déséquilibrées
- ✓ **Recall** : minimiser faux négatifs (cancer, fraude)
- ✓ **Precision** : minimiser faux positifs (spam detection)





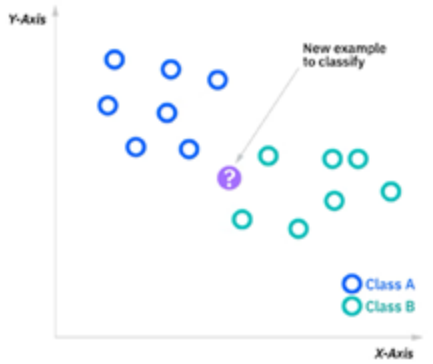


Machine Learning : Apprentissage Supervisé

K-Nearest Neighbours (KNN) (1/2)

Principe

- ✓ Algorithme d'apprentissage paresseux (lazy learning)
- ✓ Classification basée sur la proximité
- ✓ "Dis-moi qui sont tes voisins, je te dirai qui tu es"



Fonctionnement

1. Choisir K (nombre de voisins)
2. Calculer la distance entre le point à classifier et tous les points d'entraînement
3. Sélectionner les K plus proches voisins
4. Assigner la classe majoritaire parmi ces voisins

Distances courantes

- ✓ Euclidienne : $\sqrt{\sum((x_i - y_i)^2)}$
- ✓ Manhattan : $\sum|x_i - y_i|$
- ✓ Minkowski : $(\sum|x_i - y_i|^p)^{\frac{1}{p}}$

[1] https://www.ibm.com/fr-fr/think/topics/knn

K-Nearest Neighbours (KNN) (2/2)

Avantages :

- Simple à comprendre et implémenter
- Pas d'entraînement (ou très rapide)
- Naturellement multi-classe

Inconvénients :

- Prédiction lente (calcul de distances)
- Sensible à l'échelle des features
- Performance dégradée en haute dimension (curse of dimensionality)
- Sensible au bruit

Régression Linéaire (1/2)

Intuition géométrique

Trouver la droite (ou hyperplan) qui minimise l'erreur entre prédictions et valeurs réelles.

Équation :

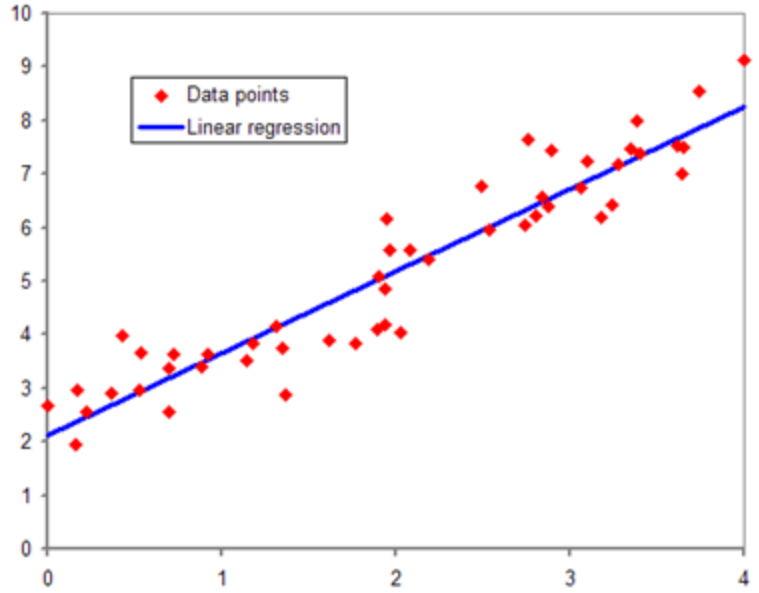
$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ ... + \beta_n x_n + \ \varepsilon$$

Fonction de coût

Erreur quadratique moyenne (Mean Squared Error) :

$$MSE = \frac{1}{n}\sum (y_i - \hat{y}_i)^2$$

L'objectif est de minimiser cette fonction en ajustant les paramètres β.



Régression Linéaire (2/2)

Méthodes de résolution

1. Solution analytique : Équations normales
    - Rapide pour petits datasets
    - Calcul matriciel
2. Descente de gradient
    - Itératif
    - Scalable pour grands datasets

Hypothèses importantes

✓ Linéarité de la relation

✓ Indépendance des résidus

✓ Homoscédasticité (variance constante)

✓ Normalité des résidus

Régression Logistique

Principe

Prédit la probabilité d'appartenance à une classe en utilisant la fonction sigmoïde.

Fonction sigmoïde :

$$\sigma(z) = \frac{1}{(1 + e^{(-z)})} \text{ où } z = \beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n$$

Interprétation

- ✓ Sortie entre 0 et 1 (probabilité)
- ✓ Seuil de décision (généralement 0.5)
- ✓ Si P(y=1|x) > 0.5 → classe 1, sinon classe

Fonction de coût

Log-loss (entropie croisée binaire) :

$$Cost = -\frac{1}{n} \Sigma[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

Extensions

- ✓ **Régression logistique multi-classe** : Softmax
- ✓ **Régularisation** : L1 (Lasso) ou L2 (Ridge)

Support Vector Machine (1/2)

Intuition des marges

Trouver l'hyperplan qui sépare au mieux les classes en **maximisant la marge** (distance minimale entre l'hyperplan et les points les plus proches).

Concepts clés

Support Vectors : Points les plus proches de l'hyperplan (les seuls qui comptent vraiment)

Hyperplan optimal :

$w \cdot x + b = 0$

où la marge est maximale : 2/||w||

Marge dure vs marge douce :

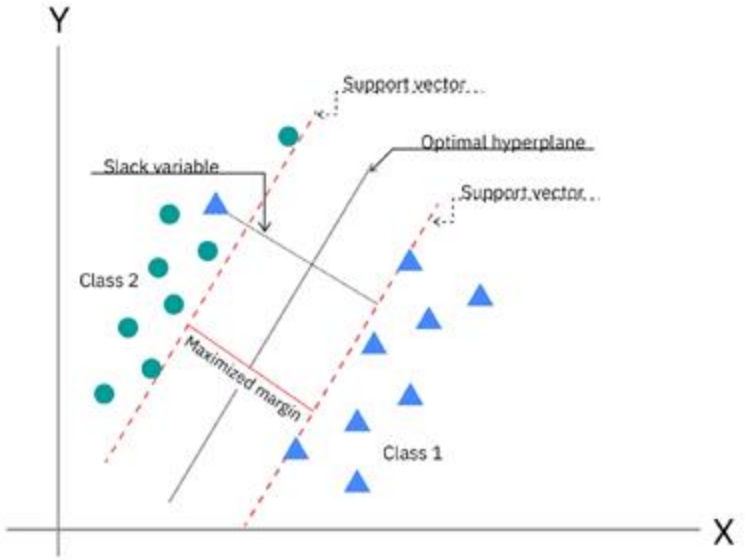
- ✓ **Marge dure** : Aucune erreur tolérée (données linéairement séparables)
- ✓ **Marge douce** : Permet quelques erreurs (paramètre C)

Kernel Trick

Pour les données non linéairement séparables, projection dans un espace de dimension supérie

Kernels courants :

- ✓ **Linéaire** : K(x, y) = x·y
- ✓ **Polynomial** : K(x, y) = (x·y + c)^d
- ✓ **RBF (Gaussian)** : K(x, y) = exp(-γ||x-y||^2)



[1] https://www.ibm.com/fr-fr/think/topics/support-vector-machine

Support Vector Machine (2/2)

Avantages :

- Efficace en haute dimension
- Versatile (grâce aux kernels)
- Robuste au sur-apprentissage

Inconvénients :

- Coûteux en temps de calcul ($O(n^2)$ ou $O(n^3)$)
- Choix du kernel et des hyperparamètres délicat
- Pas de probabilités directes

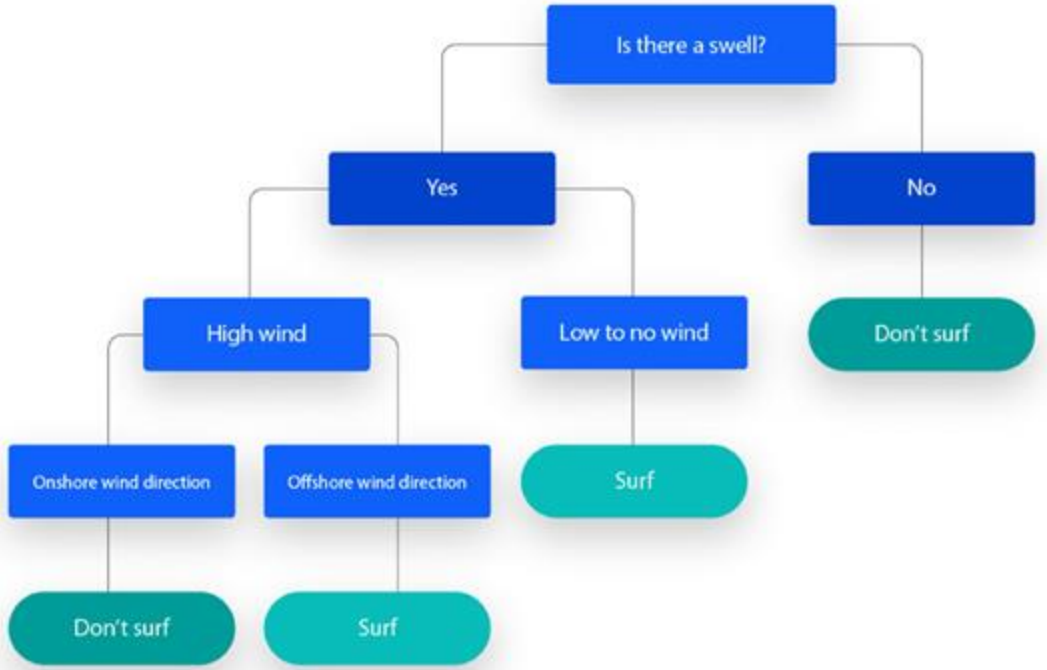
Arbre de Décision (1/2)

Principe

Succession de tests binaires sur les features menant à une décision.

Construction

1. Sélectionner la meilleure feature pour séparer les données
2. Créer des branches basées sur les valeurs de cette feature
3. Répéter récursivement pour chaque branche
4. Arrêter selon un critère (profondeur, nombre minimal d'échantillons, etc.



Critères de séparation

Pour la classification :

- ✓ **Gini Impurity** : $1 - \Sigma(p_i)^2$
- ✓ **Entropie** : $-\Sigma(p_i \log(p_i))$

Pour la régression :

- ✓ Réduction de variance

Arbre de Décision (2/2)

Avantages :

- Facile à interpréter et visualiser
- Gère automatiquement les interactions
- Pas besoin de normalisation
- Gère variables numériques et catégorielles

Inconvénients :

- Tendance au sur-apprentissage
- Instables (petites variations → arbre très différent)
- Biais vers features avec plus de valeurs

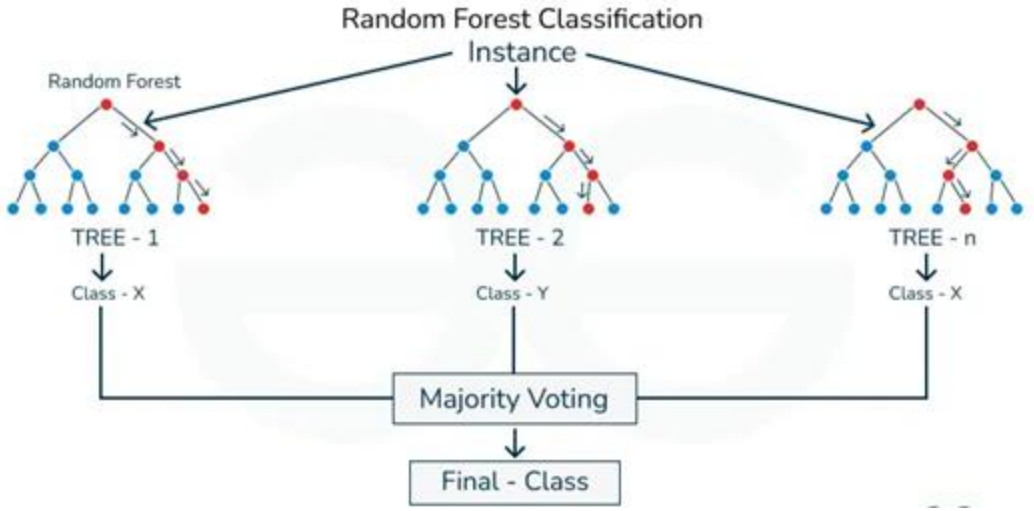
Random Forest (1/2)

Principe

Ensemble d'arbres de décision entraînés sur des sous-échantillons aléatoires.

Processus :

1. Bootstrap : Créer N échantillons aléatoires avec remise
2. Pour chaque échantillon, entraîner un arbre
3. À chaque nœud, sélectionner aléatoirement un sous-ensemble de features
4. Prédiction finale = vote majoritaire (classification) ou moyenne (régression



Bagging et Feature Randomness

- **Bagging** : Bootstrap Aggregating
- Réduit la variance sans augmenter le biais
- Décorrélation des arbres grâce à la sélection aléatoire de features

[1] https://www.geeksforgeeks.org/dsa/random-forest-classifier-using-scikit-learn/

Random Forest (2/2)

Hyperparamètres importants

- ✓ n_estimators : Nombre d'arbres
- ✓ max_depth : Profondeur maximale
- ✓ max_features : Nombre de features considérées à chaque split
- ✓ min_samples_split : Nombre minimal d'échantillons pour split

Feature Importance

Random Forest permet de calculer l'importance de chaque feature.

Avantages :

- ✓ Très performant out-of-the-box
- ✓ Robuste au sur-apprentissage
- ✓ Gère bien les données manquantes
- ✓ Parallélisable

Inconvénients :

- ✓ Moins interprétable qu'un seul arbre
- ✓ Peut être lent sur de grandes données
- ✓ Modèle lourd à sauvegarder

Machine Learning : Apprentissage Non-Supervisé

K-Means Clustering (1/2)

Principe

Partitionner n observations en K clusters où chaque observation appartient au cluster avec la moyenne la plus proche.

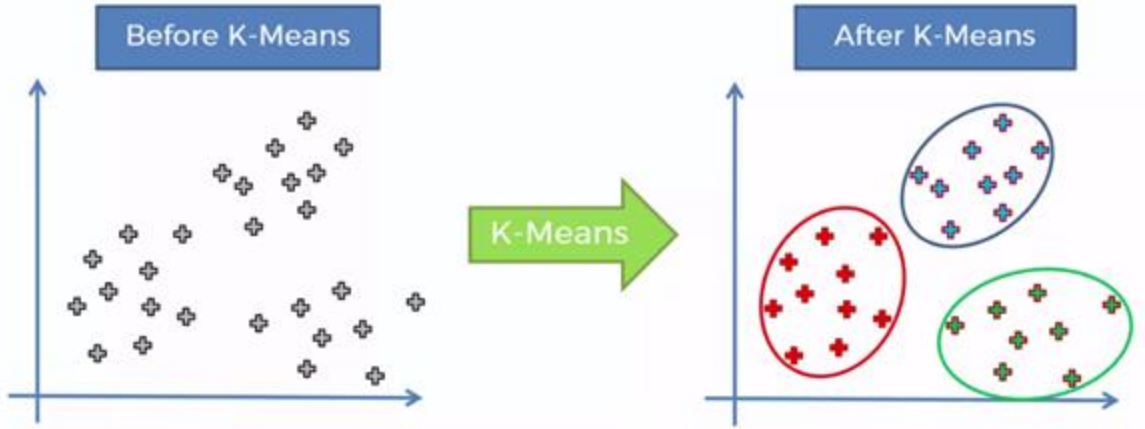
Algorithme

1. Initialiser K centroïdes aléatoirement
2. **Assignation** : Assigner chaque point au centroïde le plus proche
3. **Mise à jour** : Recalculer les centroïdes comme moyenne des points assignés
4. Répéter 2-3 jusqu'à convergence

Fonction objectif

Minimiser l'inertie intra-cluster :

$$J = \Sigma\Sigma \left|\left|x_i - \mu_j\right|\right|^2$$

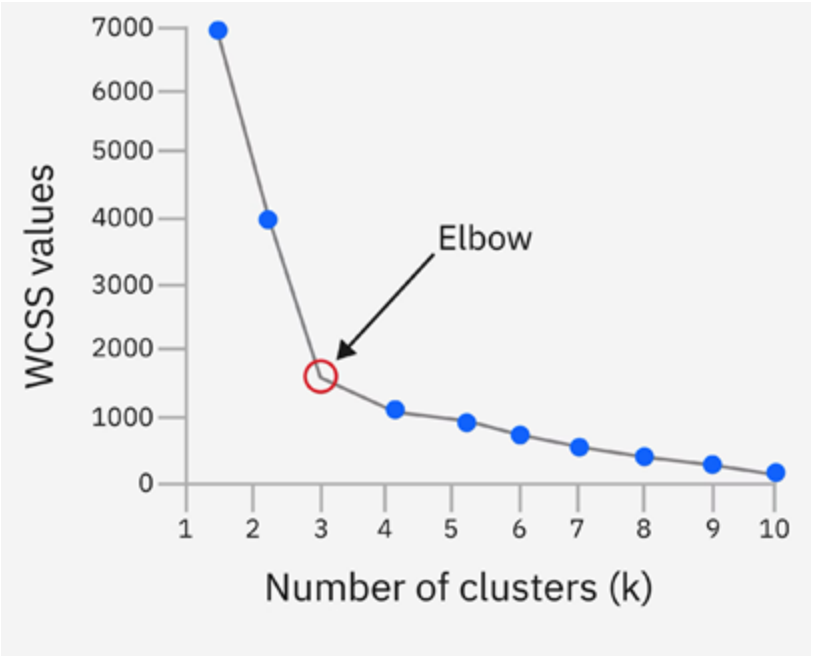


K-Means Clustering (2/2)

Choix de K

Méthode du coude (Elbow Method) :

- Tracer l'inertie en fonction de K
- Chercher le "coude" où l'amélioration ralentit



Silhouette Score :

- Mesure de cohésion des clusters
- Varie entre -1 et 1

Limites

- Sensible à l'initialisation (solution : K-means++)
- Suppose des clusters sphériques de tailles similaires
- Nécessite de spécifier K à l'avance
- Sensible aux outliers

PCA (Principal Component Analysis) (1/2)

Objectif

Réduire la dimension en projetant les données sur un sous-espace de variance maximale.

Principe

Trouver les directions (composantes principales) qui capturent le maximum de variance.

Mathématiquement :

1. Centrer les données (moyenne = 0)
2. Calculer la matrice de covariance
3. Trouver les vecteurs propres et valeurs propres
4. Sélectionner les K premiers vecteurs propres

Variance expliquée

Chaque composante principale explique une certaine proportion de la variance totale.

Variance expliquée = $\lambda_i / \Sigma\lambda_j$

PCA (Principal Component Analysis) (2/2)

Applications

- **Visualisation** : Réduire à 2D ou 3D
- **Pré-traitement** : Réduire le bruit
- **Compression** : Réduire l'espace de stockage
- **Accélération** : Réduire le temps de calcul

Avantages :

- Réduit la dimensionnalité efficacement
- Élimine la multicolinéarité
- Améliore la visualisation

Inconvénients :

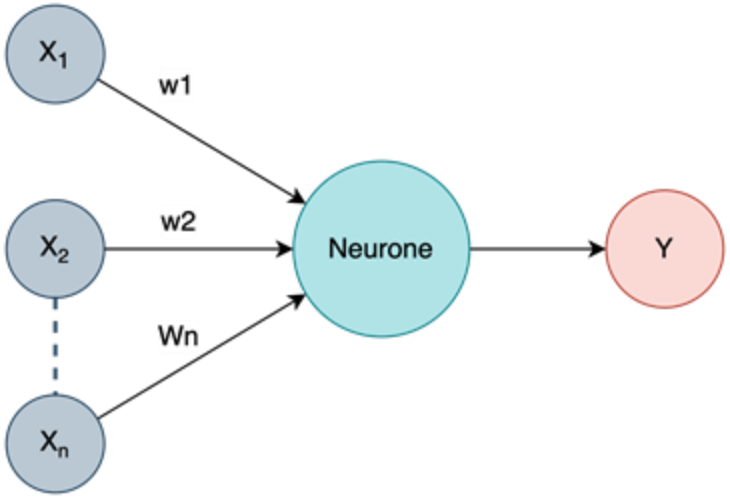
- Perte d'interprétabilité (composantes = combinaisons linéaires)
- Suppose des relations linéaires
- Sensible à l'échelle (nécessite normalisation)

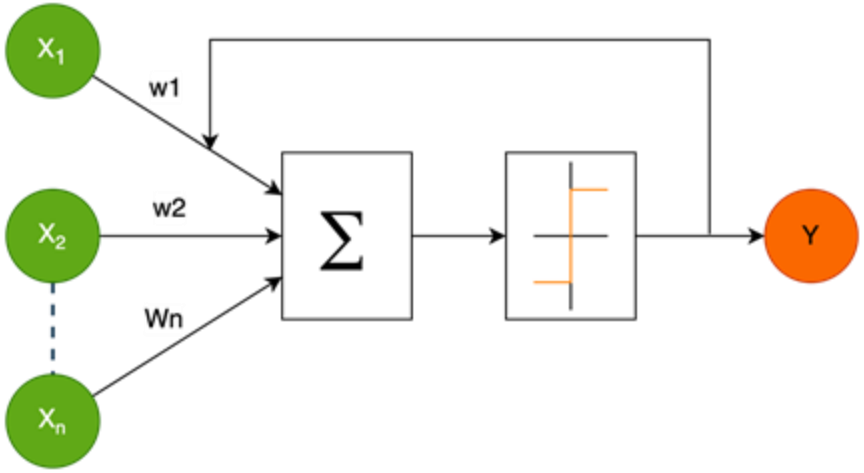
Deep Learning : Types de réseaux de nerones

Réseau de Neurones à Action Direct / Feed forward Neural Network (1/3)

1. Perceptron

- Un perceptron est la partie de base d'un réseau neuronal. Il représente un neurone unique d'un cerveau humain et est utilisé pour les classificateurs binaires.



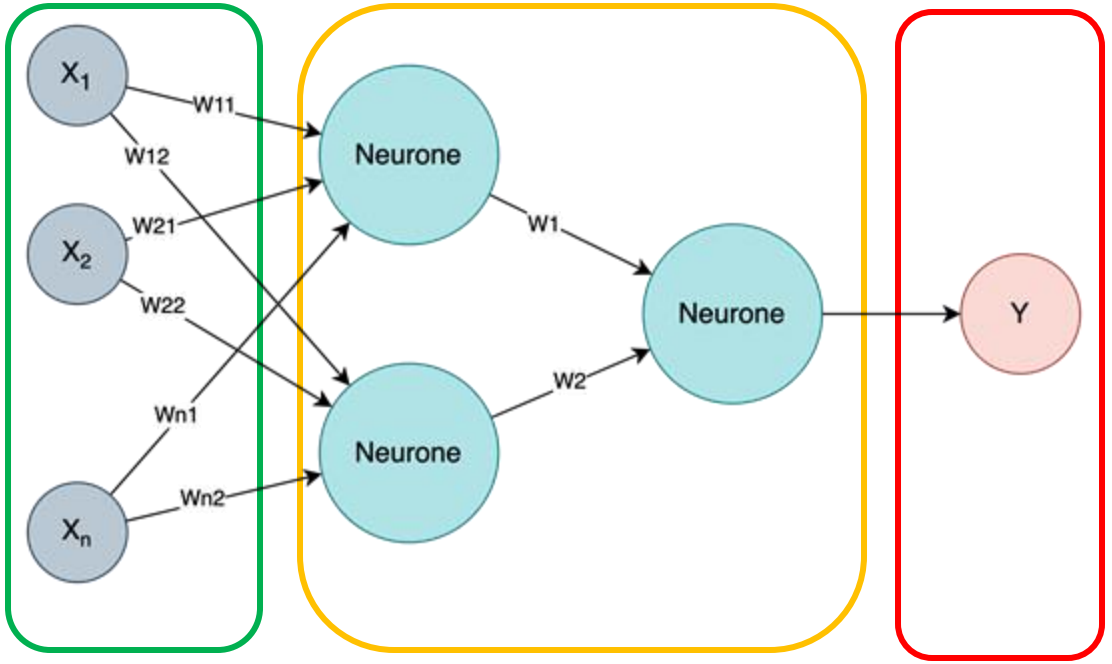


Réseau de Neurones à Action Direct / Feed forward Neural Network (2/3)

2. Perceptron multicouche (reseaux de neuronne)

- Type de réseau neurones artificiels organisés en plusieurs couche de perceptron
- Comporte au moins 03 couches :

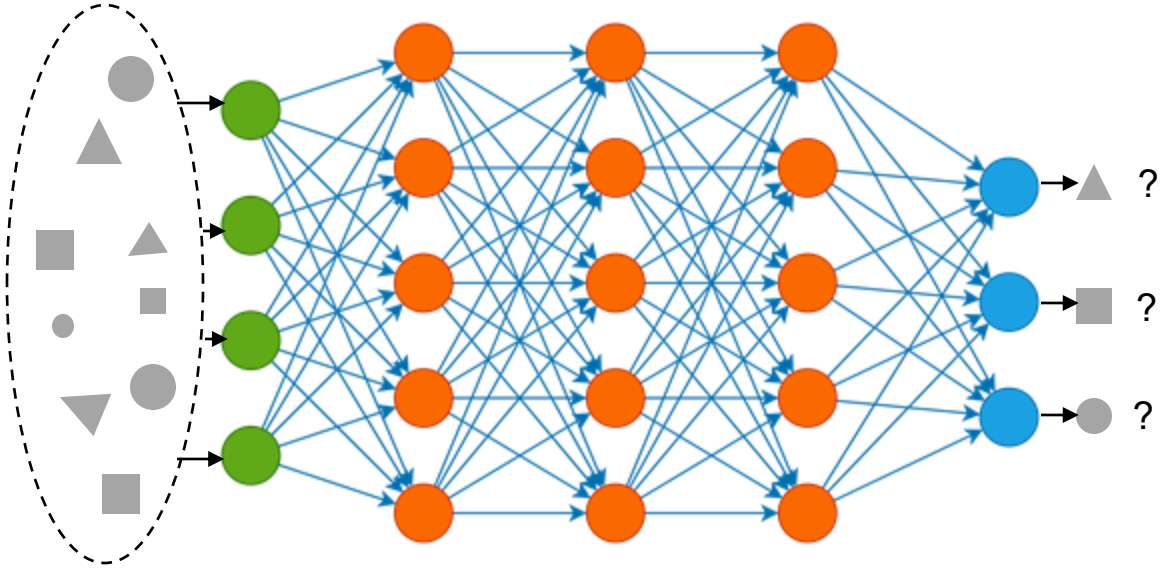
  - **Couche d'entrée** : Capte les signaux d'entrée et les transmet à la couche suivante
  - **Couche cachée** : Effectue toutes sortes de calculs et d'extraction de caractéristiques
  - **Couche de sortie** : livre le résultat final

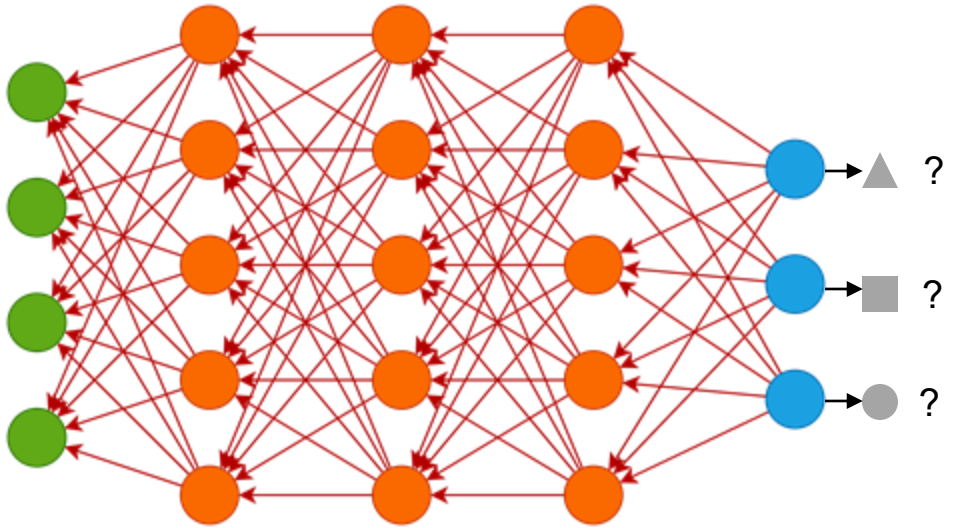


Réseau de Neurones à Action Direct / Feed forward Neural Network (1/3)

3. A grande échelle ?

- Dans les réseaux de neurones classique le flux d'information est unidirectionnel (feed forward)
- Le backward propagation permet de réajuster les paramètres interne du modèle.





$$cost = \frac{1}{2}(Y_{pred} - Y_{true})^2$$

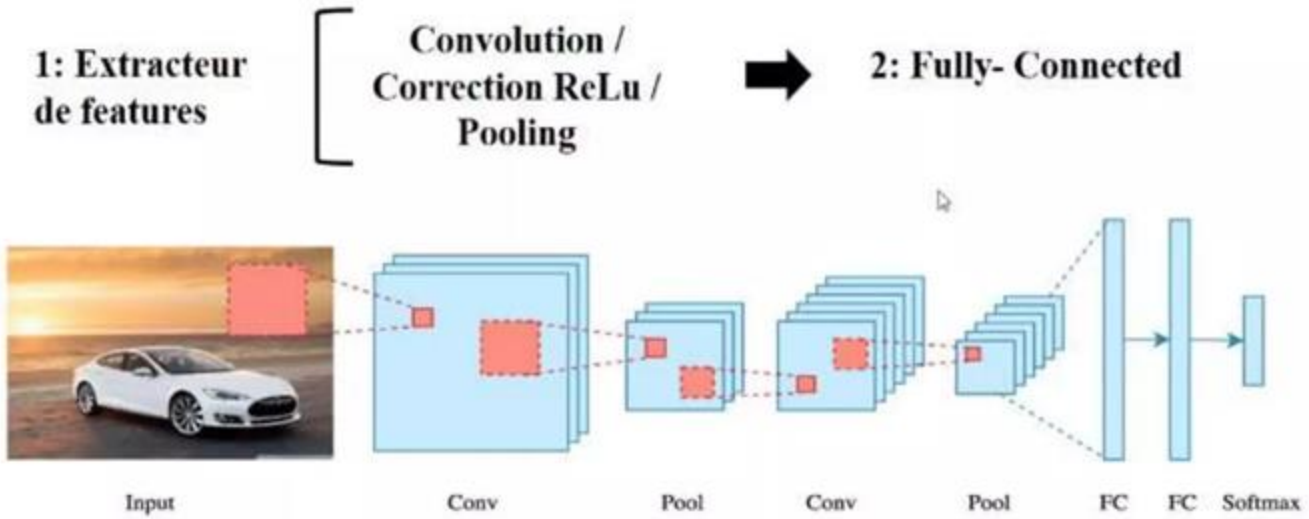
Réseau Neuronal de Convolution / Convolutional Neural Network (1/6)

1. Définition

- Type de réseau neurones artificiels conçu spécifiquement pour le traitement des données ayant une structure de type grille, comme les images.
- Utilisés dans des tâches comme la reconnaissance d'images, la vision par ordinateur et, plus récemment, le traitement du langage naturel.
- Composé de plusieurs couches successives :
  - Couches convolutionnelles : Effectuent des convolutions sur l'entrée en appliquant des filtres
  - Couches de pooling : Réduisent la dimension des cartes de caractéristiques
  - Couches entièrement connectées : Agissent comme dans les réseaux de neurones classiques.

Réseau Neuronal de Convolution / Convolutional Neural Network (2/6)

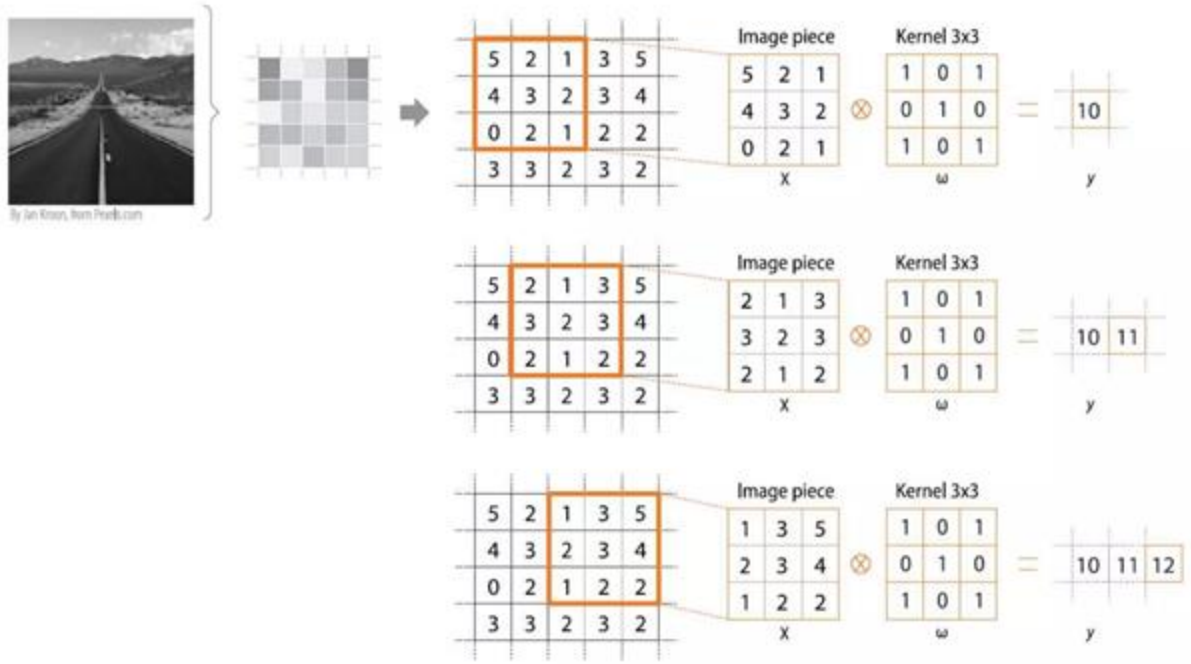
2. Fonctionnement



Réseau Neuronal de Convolution / Convolutional Neural Network (3/6)

2.1. Couche de convolution

- La convolution, d'un point de vue simpliste, est le fait d'appliquer un filtre mathématique à une image. D'un point de vue plus technique, il s'agit de faire glisser une matrice par-dessus une image, et pour chaque pixel, utiliser la somme de la multiplication de ce pixel par la valeur de la matrice. Cette technique nous permet de trouver des parties de l'image qui pourraient nous être intéressantes.



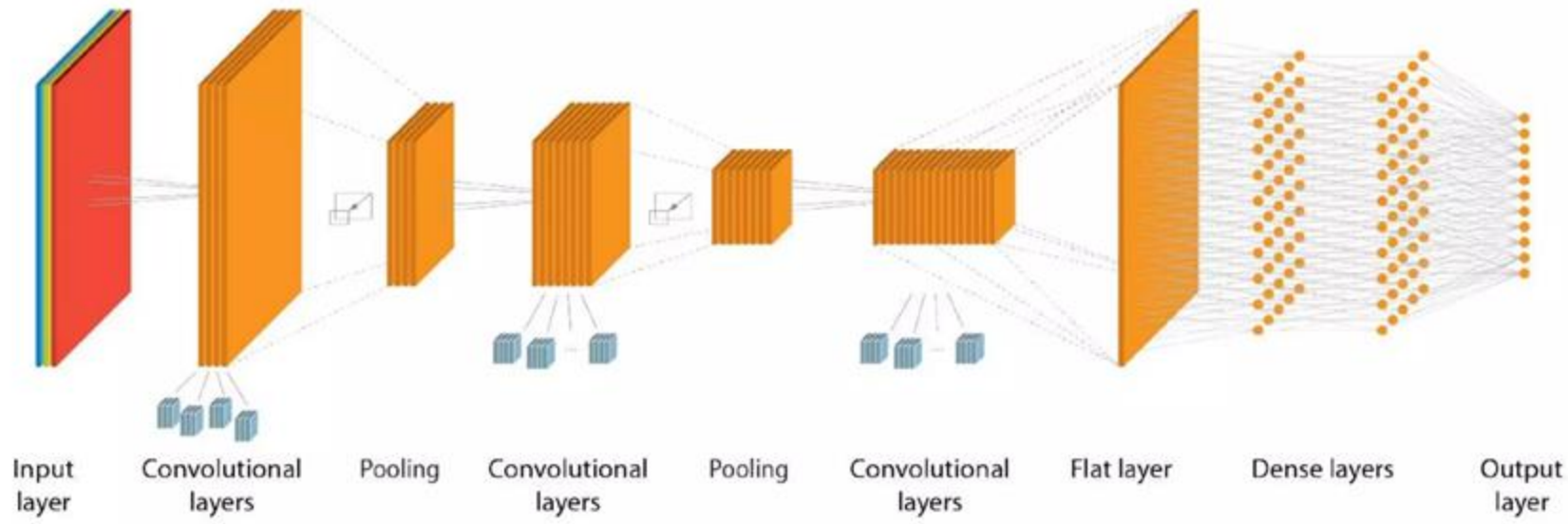
Réseau Neuronal de Convolution / Convolutional Neural Network (4/6)

2.2. Couche de pooling

- Opération utilisée pour réduire la dimension en recherchant des détails plus « grossiers », de plus grandes « structures » dans l'image
  - **Max-pooling** : on prend l'élément maximal de chaque sous-tableau
  - **Sum-pooling** : on fait la somme de tous les éléments de chaque sous tableau

Réseau Neuronal de Convolution / Convolutional Neural Network (5/6)

2.3. Recapitulatifs



Réseau Neuronal de Convolution / Convolutional Neural Network (6/6)

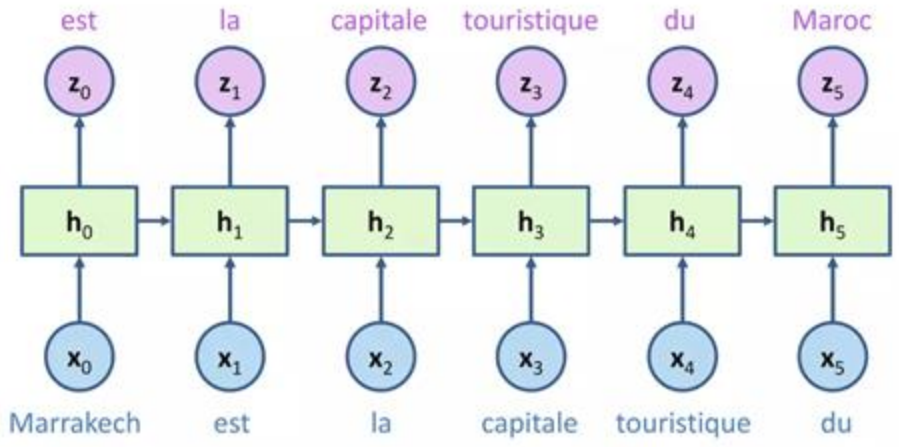
3. Applications

- Reconnaissance d'images (par exemple, ImageNet, Google Photos).
- Détection d'objets (voitures, visages, etc.).
- Analyse de vidéos.Traitement d'images médicales (diagnostic basé sur IRM, radiographies).
- Applications créatives (transfert de style artistique, génération d'images).

Réseau de Neurone Récurrent / Recurrent Neural Network (1/6)

1. Définition

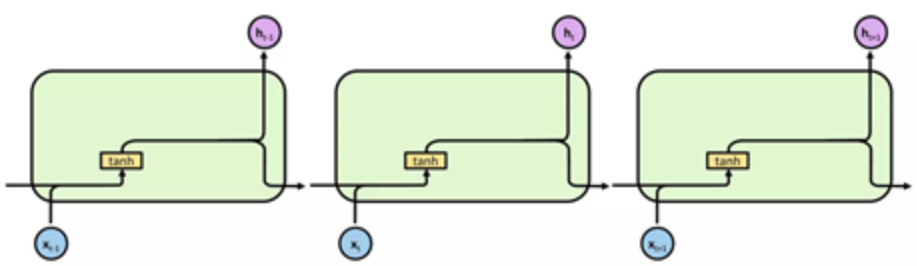
- Type de réseau de neurones conçus de manière à reconnaître les caractéristiques séquentielles et pour prédire le scénario suivant le plus probable. Les RNN sont ainsi en mesure de traiter des données séquentielles et temporelles.
- Considère à la fois l'entrée textuelle actuelle et une «unité de contexte» construite sur ce qu'ils ont vu précédemment pour prédire l'unité suivante

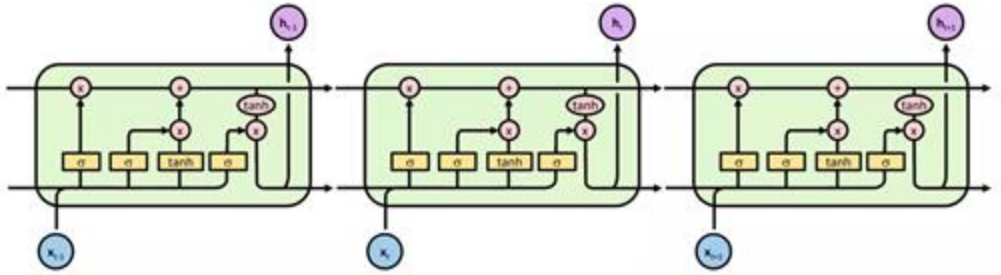


Réseau de Neurone Récurrent / Recurrent Neural Network (2/6)

2.1. Réseau RNN vs Réseau LSTM

- Le plus souvent le réseau de neurone récurent classique ne gère pas bien les dépendances à long terme à cause de la disparition du gradient (vanishing gradient) comme conséquence de l'utilisation de *tanh*.
- Les LSTM sont explicitement conçus pour éviter le problème de dépendance à long terme.



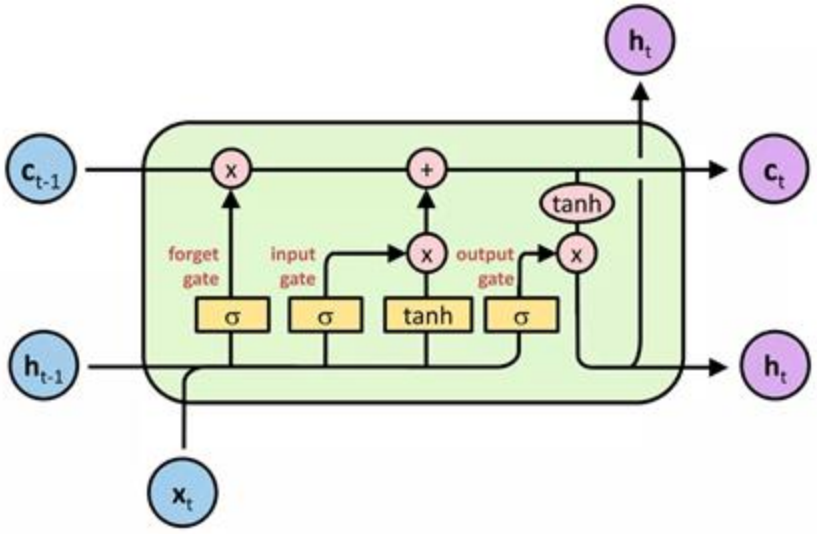


Réseau de Neurone Récurrent / Recurrent Neural Network (3/6)

2.2. Réseau LSTM

- Le plus souvent le réseau de neurone récurent classique ne gère pas bien les dépendances à long terme à cause de la disparition du gradient (vanishing gradient) comme conséquence de l'utilisation de *tanh*.
- Les LSTM sont explicitement conçus pour éviter le problème de dépendance à long terme.

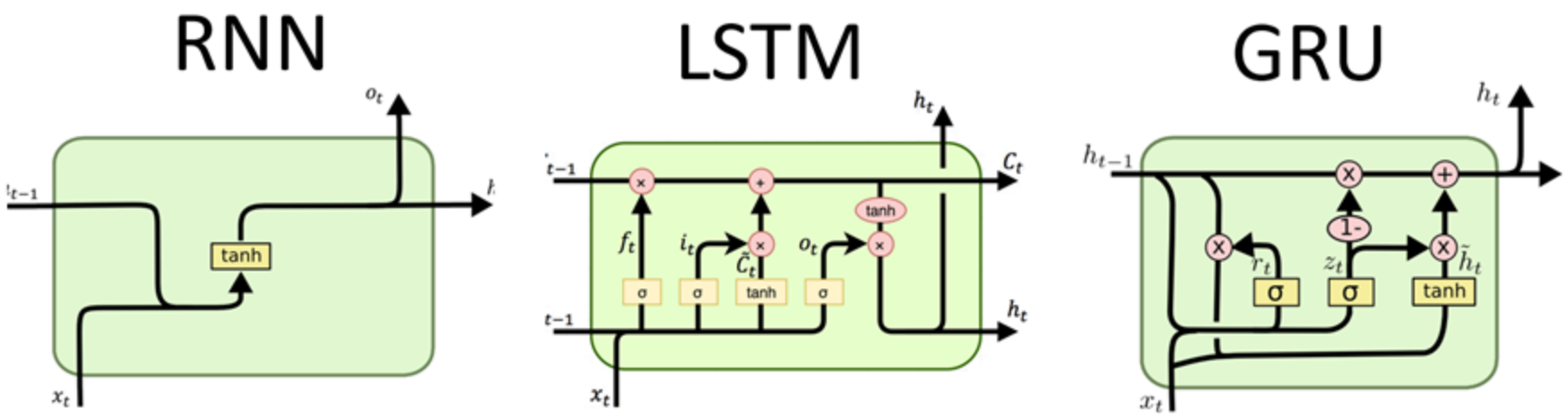
  - **Forget gate** : indique à l'état de la cellule les informations à oublier (multiplication par 0) ou à conserver (multiplication par 1).
  - **Input gate** : déterminer quelles informations produite par la couche tanh doivent entrer dans l'état de la cellule (donc à sauvegarder dans la mémoire à long terme).
  - **Output gate** : Indique les informations qui doivent passer au prochain état caché h.
  - **ct** : Etat de la cellule qui permettra de se souvenir des informations pendant de longues périodes.
  - **ht** : Etat caché



Réseau de Neurone Récurrent / Recurrent Neural Network (4/6)

2.3. Réseau GRU

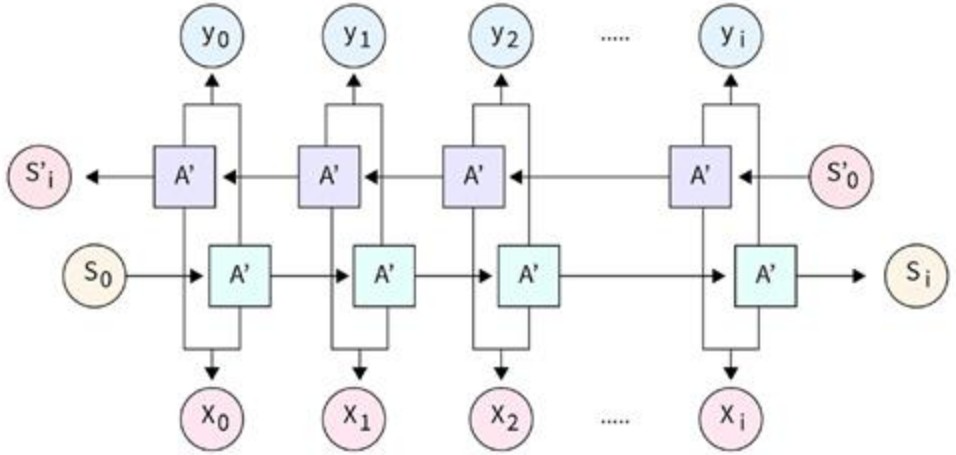
- Version simplifiée des réseaux LSTM
- Plus simples que les LSTM car ils n'ont pas de cellule de mémoire distincte : l'état caché joue directement ce rôle.
- Utilisent moins de paramètres que les LSTM, ce qui les rend plus rapides à entraîner et à exécuter, tout en conservant une performance compétitive dans de nombreux cas.



Réseau de Neurone Récurrent / Recurrent Neural Network (5/6)

2.3. Unidirectionnel vs Bidirectionnel

- **Unidirectionnel** : L'information est traitée uniquement dans **un seul sens**. À chaque instant t, le RNN unidirectionnel ne dispose que de l'information présente **dans les étapes précédentes**.
- **Bidirectionnel** : L'information est traitée dans **les 02 seul sens**. À chaque instant t, le modèle a accès aux informations **du passé et du futur** dans la sequence. Il combine 02 RNN distincts :
  - Un réseau traite la séquence dans l'ordre naturel (passé → futur).
  - L'autre traite la séquence dans l'ordre inverse (futur → passé).



Réseau de Neurone Récurrent / Recurrent Neural Network (6/6)

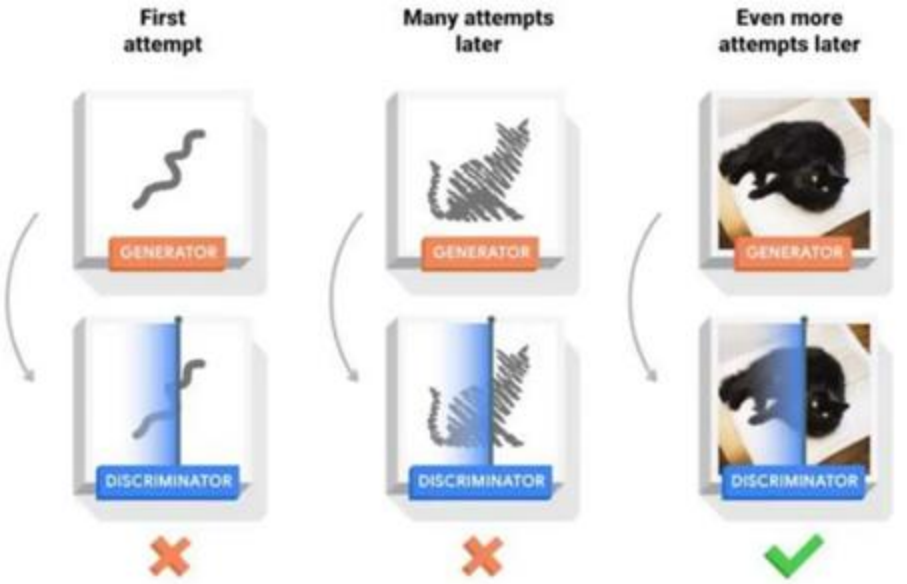
3. Applications

- Données séquentielles Les RNN traitent efficacement les données de séries temporelles pourdes prévisions.
- Analyse vidéo : analyse des cadres de vidéos pour la classification de la reconnaissance d'actions.
- Langage Naturel utilisé pour les tâches de traduction linguistique et d'analyse des sentiments.
- Génération musicale : creation des séquences musicales originales dans divers styles.
- Prévisions météorologiques : prédiction des schémas météorologiques sur la base des tendances des données historiques.
- Reconnaissance vocale : convertion le langage parlé en texte pour les applications vocales.
- Détection d'anomalies : identification des valeurs aberrantes dans des données séquentielles pour la détection de fraude.

Réseau Antagoniste Génératif / Generative Adversarial Network (1/5)

1. Définition

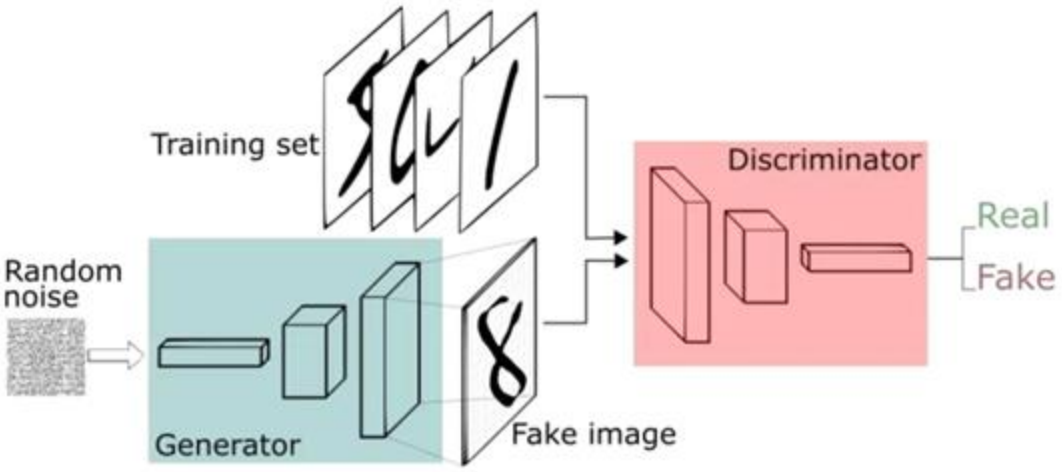
- Type de réseau de neurones conçu pour générer des données nouvelles qui ressemblent à des données réelles.
- Entrent dans la catégorie de l'apprentissage non supervisé.
- L' architecture se compose de deux réseaux neuronaux en compétition dans un scénario de théorie des jeux :
  - Le Générateur : Son rôle est de créer des exemples artificiels.
  - Le Discriminateur : Son rôle est de distinguer les données réelles des données générées.



Réseau Antagoniste Génératif / Generative Adversarial Network (2/5)

2. Principe de fontionnement

- Le générateur crée un exemple (par exemple, une image synthétique)
- Le discriminateur évalue cet exemple en le comparant à de vraies données.
- Le discriminateur donne un retour au générateur :
  - Si l'exemple semble trop faux, le générateur ajuste ses paramètres pour améliorer ses créations.
  - Si l'exemple est convaincant, le discriminateur affine sa capacité à détecter les fraudes.
- Ces deux réseaux s'entraînent en opposition jusqu'à ce que le générateur produise des données si réalistes que le discriminateur n'arrive plus à faire la distinction.



Réseau Antagoniste Génératif / Generative Adversarial Network (3/5)

2.1. Le Générateur

- Réseau de neurones artificiel conçu pour créer des données synthétiques qui ressemblent à des données réelles à partir d'un vecteur aléatoire.
- Le bruit est passé à travers plusieurs couches de réseau de neurones qui transforment progressivement ce vecteur abstrait en une donnée structurée.
- Les paramètres du générateur sont ajustés pendant l'entraînement pour produire des sorties de plus en plus réalistes.
- Structure du générateur :
  - Réseaux convolutionnels génératifs (de type CNN, mais inversés)
  - Réseaux récurrent

Réseau Antagoniste Génératif / Generative Adversarial Network (4/5)

2.2. Le discriminateur

- Joue le rôle d'un "détecteur de fraude". Son objectif principal est de distinguer entre les données réelles et les données générées par le générateur.
- Pousse le générateur a produire des données de plus en plus réaliste.
- Structure du discriminateur :
  - Réseaux convolutionnels génératifs (de type CNN, mais inversés)
  - Réseaux récurrent

Réseau Antagoniste Génératif / Generative Adversarial Network (5/5)

3. Applications

- **Création d'images réalistes** (deepfakes, portraits synthétiques, images de haute résolution)
- **Amélioration d'images** (réduction du bruit, super-résolution)
- **Génération de contenu audio ou musical**
- **Synthèse de texte**
- **Simulation / Augmentation de données** pour les jeux vidéo, la recherche scientifique, etc.

Modèle de diffusion avec débruitage / Denoising Diffusion Model (1/2)

1. Définition

- Classe de modèles génératifs utilisés pour créer ou restaurer des données (generation d'image, inpainting, …)
- Reposent sur le concept de diffusion de bruit et de débruitage progressif pour transformer une distribution de bruit aléatoire en une distribution cible de données.
  - **Apprentissage :** Pendant l'entraînement, le modèle apprend à prédire comment passer d'une image bruitée à une image moins bruitée en utilisant une fonction de perte.
  - **Génération :** Une fois entraîné, le modèle peut partir d'un bruit aléatoire (par exemple, une image "blanche" remplie de bruit) et générer une image réaliste ou plausible en débruitant progressivement.

Modèle de diffusion avec débruitage / Denoising Diffusion Model (2/2)

3. Applications

- Génération d'images (comme DALL-E 2 ou Stable Diffusion).
- Restauration d'images bruitées.
- Synthèse audio ou vidéo.
- Amélioration de la qualité des données (denoising, super-résolution, etc.).

Natural Language Processing (NLP)

Définition

Le **Natural Language Processing (NLP)** ou en français T**raitement du Langage Naturel (TAL)** est une discipline qui porte essentiellement sur la compréhension, la manipulation et la génération du langage naturel par les machines.

Domaines:

- Branche de l'IA (sous-domaine du ML) dédiée à l'interaction entre ordinateurs et langage humain.
- Interface entre la science informatique et la linguistique.

Objectifs :

- Comprendre le langage naturel
- Générer du langage naturel
- Extraire de l'information
- Traduire entre langues

Applications du NLP

- **Compréhension**
  - ➢ Classification de texte (sentiment, spam, catégories)
  - ➢ Named Entity Recognition (NER)
  - ➢ Question-Answering
  - ➢ Extraction d'information

- **Génération**
  - ➢ Traduction automatique
  - ➢ Résumé de texte
  - ➢ Chatbots et assistants virtuels
  - ➢ Génération de contenu

- **Autres**
  - ➢ Reconnaissance vocale (Speech-to-Text)
  - ➢ Synthèse vocale (Text-to-Speech)
  - ➢ Correction grammaticale
  - ➢ Autocompletion

Pourquoi le NLP est unique ? - Les défis computationnels du texte

- **Le texte comme donnée**
  - ➢ Donnée non structurée: Problème central → Comment encoder efficacement du texte pour des algorithmes ?
  - ➢ Taille variable (séquences)
  - ➢ Alphabet discret mais espace combinatoire énorme: Contrairement aux images (pixels continus), le texte est composé d'unités discrètes (tokens) issues d'un vocabulaire fini mais très large (10K-100K tokens).

- **Séquentialité et dépendances longues**

L'ordre des mots compte ! *"Le chat mange la souris" ≠ "La souris mange le chat"*

Problème majeur: Dépendances syntaxiques et sémantiques peuvent s'étendre sur plusieurs centaines de tokens → challenge pour la mémoire des modèles.

- **Ambiguïté multidimensionnelle**

Lexicale : "Il a acheté une souris" (animal ou périphérique ?)

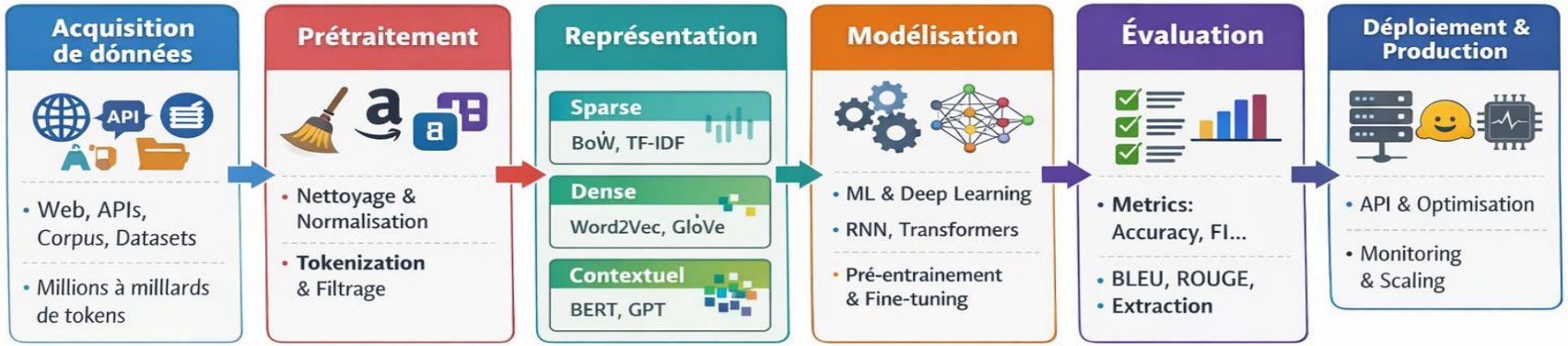
Syntaxe: "J'ai vu l'homme avec le télescope" (qui a le télescope?)

Sémantique : "Je vais à la banque" (institution financière ou bord de rivière ?)

Pragmatique: "Il fait froid ici" → requête implicite (fermer fenêtre) (Nécessite de comprendre le lien causal)

Pipeline NLP : Architecture système

Du texte brut au modèle déployé en production



Acquisition de données

- **Sources**

➢ Web scraping (BeautifulSoup, Scrapy)

➢ APIs (Twitter, Reddit), corpus académiques (Common Crawl, Wikipedia dumps)

➢ Datasets publics (Hugging Face Hub)

- **Volume**: Millions à milliards de tokens pour pré-entraînement

Prétraitement - du texte aux données **(1/2)**

Exemple de cas pratique: Supposons que vous vouliez être capable de déterminer si un mail est un spam ou non, uniquement à partir de son contenu. À cette fin, il est indispensable de transformer les données brutes (le texte du mail) en des données exploitables.
Parmi les principales étapes, on retrouve:

1- Nettoyage : Variable selon la source des données, cette phase consiste à réaliser des tâches telles que la suppression d'urls, d'emoji, etc.

2- Normalisation des données:

- **Tokenisation**, ou découpage du texte en plusieurs pièces appelés tokens.

Exemple : « Vous trouverez en pièce jointe le document en question » ; « Vous », «trouverez », « en pièce jointe », « le document », « en question ».

- **Stemming** : un même mot peut se retrouver sous différentes formes en fonction du genre (masculin féminin), du nombre (singulier, pluriel), la personne (moi, toi, eux…) etc. Le stemming désigne généralement le processus heuristique brut qui consiste à découper la fin des mots afin de ne conserver que la racine du mot.

Exemple : « trouverez » -> « trouv »

Prétraitement - du texte aux données **(2/2)**

2- Normalisation des données:

- **Lemmatisation** : cela consiste à réaliser la même tâche mais en utilisant un vocabulaire et une analyse fine de la construction des mots. La lemmatisation permet donc de supprimer uniquement les terminaisons inflexibles et donc à isoler la forme canonique du mot, connue sous le nom de lemme.

Exemple : « trouverez » -> trouvez

- **Autres opérations** : suppression des chiffres, ponctuation, symboles et stopwords, passage en minuscule.

Note: Impact direct sur :

- ➢ taille du vocabulaire
- ➢ performances
- ➢ complexité mémoire

Représentation (Feature Engineering)

Étape centrale du NLP : C’est ici que le texte devient des nombres

Approches :

- **Sparse**: Bag-of-Words, TF-IDF
- **Dense**: Word Embeddings (Word2Vec, GloVe)
- **Contextuel**: Transformers (BERT), GPT (dimension = 768-1024+)

Feature Engineering - Bag of Words

Principe

Représenter un document par le comptage de mots, sans tenir compte de l'ordre.

Exemple :

Doc1: "Le chat mange la souris" Doc2: "Le chien mange le chat" Vocabulaire: [le, la, chat, chien, mange, souris] Doc1: [2, 1, 1, 0, 1, 1] Doc2: [2, 0, 1, 1, 1, 0]

Avantages

- ✓ Simple et rapide
- ✓ Fonctionne bien pour classification de documents

Inconvénients

- ✓ Perte de l'ordre et du contexte
- ✓ Vocabulaire potentiellement énorme
- ✓ Aucune notion de similarité sémantique

Feature Engineering - TF-IDF

Term Frequency - Inverse Document Frequency

Pondère les mots selon leur importance dans un document par rapport au corpus.

TF (Term Frequency) :

$$TF(t,d) = (Nombre\ d'occurrences\ de\ t\ dans\ d) / (Nombre\ total\ de\ mots\ dans\ d)$$

IDF (Inverse Document Frequency) :

$$IDF(t) = \log(Nombre\ total\ de\ documents / Nombre\ de\ documents\ contenant\ t)$$

TF-IDF :

$$TF - IDF(t,d) = TF(t,d) \times IDF(t)$$

Intuition

- ✓ Mots fréquents dans un document mais rares dans le corpus → score élevé
- ✓ Mots très fréquents partout (stopwords) → score faible

Application

Utilisé pour :

- ✓ Classification de documents
- ✓ Recherche d'information
- ✓ Extraction de mots-clés

Feature Engineering - Word Embedding **(1/2)**

Limitation des représentations précédentes

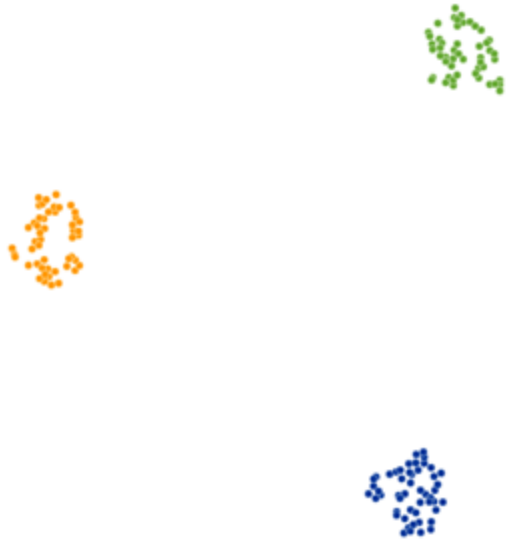
- ✓ Représentations creuses (sparse)
- ✓ Pas de notion de similarité sémantique
- ✓ Dimension = taille du vocabulaire

Principe des embeddings

Représenter les mots comme des vecteurs denses dans un espace de dimension réduite où les mots similaires sont proches.

Exemple :

"roi" - "homme" + "femme" ≈ "reine"



Feature Engineering - Word Embedding **(2/2)**

- **Word2Vec**

Deux architectures principales :

CBOW (Continuous Bag of Words) :

✓ Prédire un mot à partir de son contexte

Contexte: [le, chat, la, souris] → Prédire: "mange"

- **Skip-gram** :

✓ Prédire le contexte à partir d'un mot

Mot: "mange" → Prédire: [le, chat, la, souris]

- **GloVe (Global Vectors)**

Méthode basée sur la co-occurrence globale des mots dans le corpus.

Combine :

✓ Statistiques globales (comme matrice de co-occurrence)

✓ Apprentissage contextuel (comme Word2Vec)

Modélisation

Choix dépendant du problème

Méthodes:

- **ML** : rapide, explicable
- **DL** : contexte, séquences
- **LLM** : généralisation, zéro-shot

➡️ Le modèle consomme uniquement des vecteurs

Modélisation – Machine Learning classique

Principe : NLP vu comme ML sur données vectorisées

Tâches typiques

- Classification de texte
- Filtrage de spam
- Analyse de sentiment

Algorithmes utilisés

- Naive Bayes
- Régression logistique
- SVM

Limites

- Perte de la sémantique
- Mots synonymes non liés
- Pas de notion de contexte

Modélisation - Deep Learning

Pourquoi le DL ?

Données séquentielles

Contexte dépendant de la position

Réseaux Récurrents (RNN)

- **Motivation** : Traitement séquentiel (Texte = séquence), Besoin de capturer les dépendances temporelles; Les CNN et MLP ne gèrent pas naturellement les séquences

- **Applications**

Modélisation de langue	Traduction automatique (seq2seq)
Génération de texte	Analyse de sentiment

- **Limitations des RNNs classiques**

Vanishing gradient (difficile d'apprendre dépendances longues)

Séquentiel (pas parallélisable)

- **Solutions** : LSTM, GRU (Meilleure gestion du long terme)

Modélisation – Transformers **(1/2)**

Limites des RNN
Séquentialité
Parallélisation difficile

Self-Attention
Chaque token regarde les autres
Complexité $O(n^2)$

Modèles modernes
- BERT (encodeur)
- GPT (décodeur)
- T5

➡️ Paradigme : **Pre-training + Fine-tuning**

Modélisation – Transformers **(1/3)**

Limites des RNN
Séquentialité
Parallélisation difficile

Self-Attention
Chaque token regarde les autres
Complexité $O(n^2)$

Modèles modernes
- BERT (encodeur)
- GPT (décodeur)
- T5

➡️ Paradigme : **Pre-training + Fine-tuning**

Modélisation – Transformers **(2/3)**

Révolution "Attention is All You Need" (2017)

Abandon des RNN au profit d'un mécanisme d'**attention** pur.

Mécanisme d'attention

Intuition : Permettre au modèle de se concentrer sur les parties pertinentes de l'entrée.

Exemple :

Traduction: "The cat sat on the mat" → "Le chat s'est assis sur le tapis" Pour traduire "chat", le modèle attend "cat" (attention élevée).

Self-Attention : Chaque mot peut attendre tous les autres mots de la séquence.

Modélisation – Transformers **(3/3)**

Architecture Transformer

Encodeur :

- Stack de couches (généralement 6 ou 12)
- Chaque couche : Multi-Head Self-Attention + Feed-Forward

Décodeur :

- Similaire mais avec attention masquée (pas d'accès au futur)
- Attention croisée avec l'encodeur

Avantages

- **Parallélisation** : Toute la séquence traitée en une fois
- **Dépendances longues** : Pas de limite de distance
- **Interprétabilité** : Visualisation des poids d'attention

Modélisation – Large Language Models (LLM) **(1/5)**

Qu'est-ce qu'un LLM ?

- Modèle de langage très grande taille (milliards de paramètres)
- Basé sur l'architecture Transformer
- Entraîné sur des corpus massifs de texte

Objectif fondamental : prédire le token suivant

Pré-entraînement des LLM (Apprendre les représentations générales du langage) :

- Auto-supervisé
- Données non annotées
- Tâche : next-token prediction

Avantage : apprentissage des représentations générales du langage, de la syntaxe et du raisonnement implicite

Modélisation – Large Language Models (LLM) **(2/5)**

Zero-shot Learning

Accomplir une tâche sans exemple, juste avec des instructions.

Prompt: "Traduis en français : Hello world" Output: "Bonjour le monde"

Few-shot Learning

Fournir quelques exemples dans le prompt.

Prompt: "Traduire en français: Hello → Bonjour Goodbye → Au revoir Thank you → ?" Output: "Merci"

Fine-tuning

Entraîner sur un dataset spécifique pour une tâche précise.

Applications

Chatbots : Assistants conversationnels

Résumé : Articles, documents, reunions

Extraction d'information : Entités, relations

Question-Answering : Sur documents, connaissances générales

Génération de code : GitHub Copilot, etc.

Analyse de sentiment : Opinions, reviews

Traduction : Tous types de langues

Modélisation – Large Language Models (LLM) **(3/5)**

Architectures principals:

BERT (Bidirectional Encoder Representations from Transformers) :

- Encodeur uniquement
- Pré-entraîné avec Masked Language Modeling (MLM)
- Excellent pour la compréhension (classification, NER, Q&A)

GPT (Generative Pre-trained Transformer) :

- Décodeur uniquement
- Pré-entraîné avec prédiction du mot suivant
- Excellent pour la generation

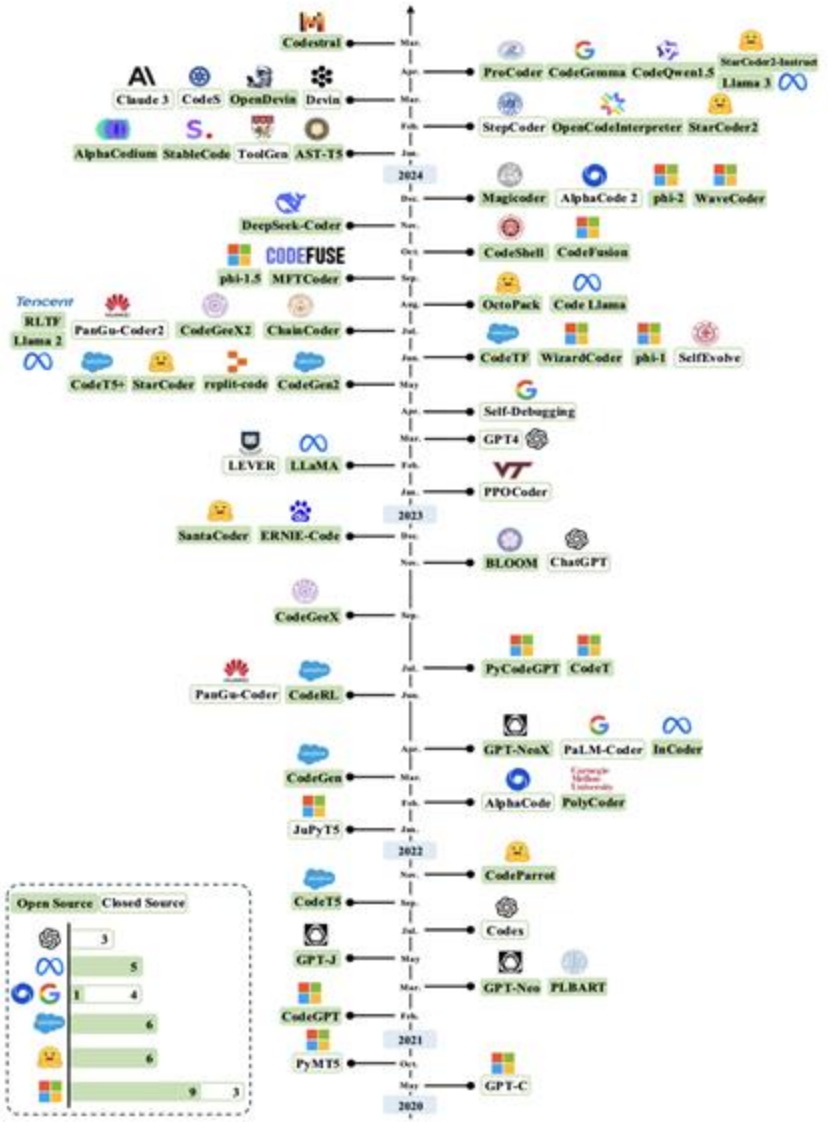
T5 (Text-to-Text Transfer Transformer) :

- Encodeur-décodeur
- Toutes les tâches formulées en texte-à-texte
- Versatile

Modélisation – Large Language Models (LLM) **(4/5)**

Modèles modernes

- GPT-3, GPT-4 (OpenAI)
- LLaMA (Meta)
- Claude (Anthropic)
- Gemini (Google)
- Mistral (Mistral AI)



Modélisation – Large Language Models (LLM) **(4/5)**

Limitations techniques

- **Hallucinations** : Génération d'informations fausses avec confiance
- **Longueur de contexte** : Limite fixe (bien qu'en augmentation)
- **Coût computationnel** : Entraînement et inférence coûteux
- **Biais** : Reflètent les biais présents dans les données d'entraînement

Défis éthiques

- **Désinformation** : Génération de fake news
- **Vie privée** : Mémorisation de données sensibles
- **Propriété intellectuelle** : Reproduction de contenu protégé
- **Impact environnemental** : Empreinte carbone de l'entraînement

Défis de recherche

- **Raisonnement** : Logique, mathématiques
- **Connaissance du monde** : Compréhension causale
- **Multimodalité** : Intégration vision + langage + audio
- **Efficacité** : Modèles plus petits et performants

NLP Multimodal - Extension au-delà du texte

Vision + Language :

- Image captioning
- Visual Question Answering (VQA)
- Text-to-Image (DALL-E, Stable Diffusion, Midjourney)

Audio + Language :

- Speech-to-Text (Whisper)
- Text-to-Speech (naturel et expressif)
- Traduction vocale simultanée

Signes + Language :

- Reconnaissance de langue des signes
- Génération d'avatars signeurs
- Traduction texte ↔ signes

Modèles multimodaux récents

- CLIP (OpenAI) : Vision + Language
- Whisper (OpenAI) : Speech recognition
- Flamingo (DeepMind) : Vision-Language
- GPT-4V : Vision-Language-Reasoning

Evaluation

Métriques courantes :

- Classification : Accuracy, F1, précision, rappel
- Génération : BLEU, ROUGE, perplexité
- Compréhension : Exact Match, F1 (Q&A)

Évaluation humaine : Souvent nécessaire pour qualité subjective

Exemple : **Mean Opinion Score (MOS)** en TTS

Evaluation

Métriques courantes :

- Classification : Accuracy, F1, précision, rappel
- Génération : BLEU, ROUGE, perplexité
- Compréhension : Exact Match, F1 (Q&A)

Évaluation humaine : Souvent nécessaire pour qualité subjective

Exemple : **Mean Opinion Score (MOS)** en TTS

Note : Dépoiement à voir dans un module dedié

Bonne pratiques NLP

Préparation des données

- Nettoyage : HTML tags, URLs, mentions
- Normalisation : Casse, accents, unicode
- Tokenization : Adapter au modèle et à la langue
- Augmentation : Synonymes, back-translation

Choix du modèle

Règle générale :

- Peu de données → Modèles pré-entraînés + fine-tuning
- Beaucoup de données → Possibilité d'entraîner from scratch
- Ressources limitées → Modèles plus petits (DistilBERT, MobileBERT)
- Production → Compromis performance/latence

References

- Nelson, Hala. *Essential Math for AI : : Next-Level Mathematics for Efficient and Successful AI Systems .*
- Maxime Labonne. Hands-On Graph Neural Networks Using Python
- David Foster. Generative Deep Learning, 2nd Edition
- Suhas Pai. Designing Large Language Model Applications
- https://datascientest.com/graph-neural-networks-tout-savoir
- Youtube, Google, ChatGPT 🙂