Algorithmes d'apprentissage supervisé

Table des matières

> 🖊 Note pour plus tard
>
> mettre le schéma avec l'explication des différents algo

Algorithme Dictionnaire

L'algorithme de **classification par dictionnaire** en apprentissage supervisé est une approche relativement simple qui repose sur la création d'un dictionnaire de caractéristiques (souvent des mots, dans le cas d'un texte) associées à des catégories ou des classes. Cet algorithme est souvent utilisé dans des contextes comme la **classification de texte** ou la **détection de spam**.

Principe de base

L'idée de l'algorithme du **dictionnaire** est de créer une sorte de "référence" des caractéristiques présentes dans l'ensemble d'apprentissage. En général, il s'agit d'une **table** ou d'une **liste de mots** associée à des **fréquences** ou des **scores** indiquant à quelle classe chaque mot (ou caractéristique) est associé. Lorsqu'un nouvel exemple est présenté, l'algorithme vérifie les mots ou caractéristiques présents dans l'exemple, et utilise le dictionnaire pour prédire la classe la plus probable.

Étapes de l'algorithme de classification par dictionnaire

1. **Phase d'entraînement : Construction du dictionnaire** :
    - On dispose d'un ensemble de données d'entraînement constitué d'exemples (souvent des textes ou des objets contenant des caractéristiques) **associés à des étiquettes de classe** (spam, non-spam, positif, négatif, etc.).

- Le **dictionnaire** est construit en :
    1. **Collectant** les caractéristiques (comme les mots ou les phrases) de chaque exemple dans l'ensemble d'entraînement.
    2. **Comptant** la fréquence d'apparition de chaque caractéristique pour chaque classe.
    3. **Créant un dictionnaire** où chaque mot ou caractéristique est associé à un score (comme une probabilité) pour chaque classe.

  Exemple : En classification d'e-mails, si un mot comme "gagner" apparaît souvent dans des e-mails marqués comme spam, ce mot sera enregistré dans le dictionnaire avec un score ou une pondération élevée pour la classe "spam".

2. **Phase de prédiction** :
    - Lorsqu'un **nouvel exemple** est présenté (comme un nouvel e-mail), l'algorithme **scanne** ses caractéristiques (mots, termes, etc.).
    - Pour chaque caractéristique présente dans le nouvel exemple, il **regarde dans le dictionnaire** pour voir à quelle classe cette caractéristique est associée.
    - L'algorithme **somme les scores** ou les pondérations des caractéristiques présentes dans le dictionnaire pour chaque classe.
    - Finalement, l'algorithme prédit que le nouvel exemple appartient à la classe qui a obtenu le score le plus élevé.

Avantages de l'approche par dictionnaire

1. **Simple à implémenter** : Le concept de base est facile à comprendre et à mettre en œuvre.
2. **Interprétable** : Le dictionnaire donne une idée claire des caractéristiques les plus importantes pour chaque classe. On peut facilement observer quels mots sont associés à quelles classes.
3. **Rapide à utiliser** : Une fois le dictionnaire construit, la prédiction est rapide car il suffit de consulter un tableau de correspondance et de faire des sommes.

Limites de l'approche par dictionnaire

1. **Hypothèse de dépendance linéaire** : L'algorithme suppose que les caractéristiques (par exemple, les mots) agissent indépendamment les unes des autres, ce qui peut être une simplification trop forte dans de nombreux cas.

2. **Manque de généralisation** : Si un mot ou une caractéristique n'a jamais été rencontré dans l'ensemble d'entraînement, l'algorithme aura du mal à traiter cet exemple (cela peut être résolu par des approches comme Bayes naïf).

3. **Pertinence du contexte** : Les mots ou caractéristiques peuvent avoir des significations différentes en fonction du contexte, mais l'algorithme ne prend pas cela en compte. Par exemple, le mot "argent" peut apparaître dans un e-mail légitime ou un spam, mais le dictionnaire ne prend pas en compte cette nuance contextuelle.

Différence avec d'autres approches

- **Comparé à k-NN** : Le k-NN ne construit pas de dictionnaire. Il compare directement les nouveaux exemples à l'ensemble d'entraînement complet. Le k-NN peut être plus flexible mais plus coûteux en termes de calcul, tandis que le dictionnaire est plus rapide une fois construit.

- **Comparé à Bayes naïf** : Le dictionnaire est similaire à Bayes naïf, mais Bayes utilise un cadre probabiliste basé sur le théorème de Bayes, ce qui lui permet de calculer explicitement les probabilités. L'approche par dictionnaire est moins formelle et ne calcule pas de probabilités précises.

Exemple simple : Classification de spam

Étape 1 : Construction du dictionnaire

On a un ensemble d'e-mails étiquetés comme **spam** ou **non-spam**. On analyse les mots présents dans chaque e-mail :

Mot	Fréquence dans spam	Fréquence dans non-spam
gagner	30	5
argent	25	8
offre	20	2
bonjour	5	40
réunion	3	50

Dans cet exemple, on voit que les mots **gagner** et **argent** apparaissent plus souvent dans des e-mails marqués comme spam, tandis que **bonjour** et **réunion** apparaissent plus fréquemment dans des e-mails non-spam.

Étape 2 : Prédiction

Supposons qu'on souhaite classer un nouvel e-mail contenant les mots : **"gagner argent réunion"**.

- On regarde dans le dictionnaire pour chaque mot présent dans l'e-mail.
- Ensuite, on additionne les scores :
  - Pour la classe **spam** :
    - "gagner" = 30 points
    - "argent" = 25 points
    - "réunion" = 3 points
    - Total : **58 points**
  - Pour la classe **non-spam** :
    - "gagner" = 5 points
    - "argent" = 8 points
    - "réunion" = 50 points
    - Total : **63 points**

Dans cet exemple, le score est plus élevé pour la classe **non-spam** (63), donc l'algorithme prédit que cet e-mail n'est **pas un spam**.

Algorithme des k plus proches voisins (K-NN)

L'algorithme des **k plus proches voisins** (*k-nearest neighbors*, ou k-NN) est un algorithme de classification (et parfois de régression) très simple et intuitif. Il fonctionne en analysant la proximité des données d'entrée par rapport à celles d'un ensemble d'apprentissage.

sources internet :

- *https://fr.wikipedia.org/wiki/M%C3%A9thode_des_k_plus_proches_voisins*
- *voir les deux pdf (TD et cours)*

sources vidéos :

- *https://www.youtube.com/watch?v=9pvbEP1eyNY*
- *https://www.youtube.com/watch?v=VuBvOuNzvMo*

Principe de l'algorithme

- Lorsqu'on veut **classer** un nouvel élément, l'algorithme cherche les **k** points de données les plus proches dans l'ensemble d'entraînement (d'où le nom "k plus proches voisins").
- Ensuite, il attribue la **classe majoritaire** de ces k voisins à l'élément que l'on souhaite classer.

Étapes de l'algorithme k-NN

1. **Choisir un paramètre k** (le nombre de voisins à considérer).
2. **Calculer la distance** entre le point à classer et tous les points de l'ensemble d'apprentissage (utilisation de la distance euclidienne, manhattan, ou autre).
3. **Trier les distances** pour identifier les k plus proches voisins.
4. **Majorité des voisins** : Parmi les k voisins, compter la fréquence de chaque classe.
5. **Prédire la classe** : Assigner au point à classer la classe majoritaire parmi les k plus proches voisins.

Exemple simple

Imagine que l'on souhaite classer un nouveau point en fonction de ses caractéristiques. On regarde les k points les plus proches (par exemple, les 5 plus proches voisins), et si 3 de ces points appartiennent à la classe A et 2 à la classe B, alors le nouvel élément sera classé dans la classe A (car elle est majoritaire parmi les voisins).

Paramètres et variantes

- **Choix de k :** Un petit k peut rendre l'algorithme plus sensible au bruit (i.e., à des données anormales ou erronées), tandis qu'un k trop grand peut lisser trop les frontières entre les classes.
- **Distance :** L'algorithme dépend de la manière dont la distance est mesurée entre les points. La distance euclidienne est courante, mais on peut aussi utiliser d'autres métriques (ex. : distance de Manhattan, distance de Minkowski).

Avantages

- Simple à comprendre et à implémenter.
- Pas besoin d'une phase d'entraînement explicite, ce qui signifie qu'il est souvent qualifié d'algorithme "paresseux" (*lazy learner*).

Inconvénients

- **Complexité temporelle** : k-NN peut devenir coûteux en termes de calcul, car il doit calculer la distance à tous les autres points à chaque nouvelle prédiction.
- **Sensibilité à la dimension** : La performance peut se dégrader avec des jeux de données à haute dimension (phénomène de la **malédiction de la dimensionnalité**).

Algorithme Bayes

L'algorithme de **Bayes naïf** est un classificateur probabiliste basé sur le **théorème de Bayes**. Il est appelé "naïf" car il fait l'hypothèse (souvent irréaliste) que toutes les **caractéristiques** d'un ensemble de données sont **indépendantes les unes des autres**, ce qui simplifie les calculs.

sources internet :

- *https://fr.wikipedia.org/wiki/Classification_na%C3%AFve_bay%C3%A9sienne*

source vidéos :

- à compléter ?

Principe de l'algorithme

L'algorithme de Bayes naïf calcule la **probabilité** qu'un élément appartienne à une classe donnée en fonction de ses caractéristiques, et choisit la classe avec la probabilité la plus élevée.

Le **théorème de Bayes** s'exprime ainsi :

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)} P(C|X) = P(X) P(X|C) \cdot P(C)$$

Où :

- $P(C|X) P(C|X) P(C|X)$ est la **probabilité a posteriori** que l'élément appartienne à la classe C, étant donné les caractéristiques X.
- $P(C) P(C) P(C)$ est la **probabilité a priori** d'observer la classe C.
- $P(X|C) P(X|C) P(X|C)$ est la **vraisemblance** (probabilité d'observer les caractéristiques X, étant donné que la classe est C).
- $P(X) P(X) P(X)$ est la **probabilité totale** d'observer les caractéristiques X (elle est la même pour toutes les classes et peut être ignorée dans le calcul comparatif).

Étapes de l'algorithme

1. **Calculer les probabilités a priori** : Déterminer la probabilité d'occurrence de chaque classe dans l'ensemble d'apprentissage.
2. **Calculer la vraisemblance** $P(X|C) P(X|C) P(X|C)$: En se basant sur l'hypothèse d'indépendance, on calcule la probabilité de chaque caractéristique indépendamment dans chaque classe.
3. **Appliquer le théorème de Bayes** : Combiner les probabilités pour chaque classe avec les caractéristiques du nouvel élément.
4. **Prédire la classe** : La classe ayant la plus haute probabilité a posteriori $P(C|X) P(C|X) P(C|X)$ est celle attribuée au nouvel élément.

Exemple simple

Supposons que tu cherches à classer un e-mail comme **spam** ou **non-spam** en fonction de la présence de certains mots. En utilisant un ensemble d'apprentissage d'emails marqués comme spam ou non-spam, Bayes naïf calcule la probabilité que l'e-mail appartienne à la catégorie spam, en prenant en compte la présence de chaque mot et la probabilité qu'il apparaisse dans des e-mails spam ou non-spam.

Avantages

- **Simple et rapide** : Très efficace avec des jeux de données larges, et peu coûteux en calcul.
- **Robuste avec des données bruitées** : Il fonctionne bien même lorsque les hypothèses d'indépendance ne sont pas strictement respectées.
- **Probabilités explicites** : Il fournit des probabilités explicites pour chaque classe, ce qui permet d'avoir une interprétation claire des résultats.

Inconvénients

- **Hypothèse d'indépendance des caractéristiques** : Dans de nombreux cas, cette hypothèse est irréaliste (par exemple, les mots dans un texte ne sont pas indépendants les uns des autres). Cette limitation peut affecter la précision.
- **Ne gère pas bien les données continues** sans prétraitement.