Faculté des sciences Rabat

Licence fondamentale en Mathématiques et applications

Analyse des images par machine learning

Encadré par
Mr ZIANI

Réalisé par :
Mr Ayoub OUAALA

2021 - 2022

Remerciements

je tiens à saisir cette occasion et adresser mes profonds remerciements et mes profondes reconnaissances à :

Mon encadrant Mr Ziani , pour l'orientation, la confiance, la patience qui as constitué un apport considérable sans lequel ce travail n'aurait pas pu être mené au bon port. Son œil critique m a été très précieux pour structurer le travail et pour améliorer la qualité des différentes sections. j espère Qu'il trouve dans ce travail un hommage vivant à sa haute personnalité.

Mes vifs remerciements vont également aux membres du jury pour l'intérêt qu'ils ont porté à ma recherche en acceptant d'examiner mon travail et de l'enrichir par leurs propositions.

Mes remerciements s'étendent également à tous nos enseignants durant ces années d'études.
J'oublie pas mes parents pour leur contribution, leur soutien et leur patience. mes proches et amis, qui m ont toujours encouragée au cours de la réalisation de ce projet .

Enfin, je tiens également à remercier toutes les personnes qui ont participé de près ou de loin à la réalisation de ce travail.

Table des matières

Introduction

Le traitement d'images est une discipline de l'informatique et des mathématiques appliquées qui étudie les images numériques et leurs transformations, dans le but d'améliorer leur qualité ou d'en extraire de l'information.

Il s'agit d'un sous-ensemble du traitement du signal dédié aux images et aux données dérivées comme la vidéo (par opposition aux parties du traitement du signal consacrées à d'autres types de données : son et autres signaux mono dimensionnels notamment), tout en opérant dans le domaine numérique (par opposition aux techniques analogiques de traitement du signal, comme la photographie ou la télévision traditionnelles).

Dans le contexte de la vision artificielle, le traitement d'images se place après les étapes d'acquisition et de numérisation, assurant les transformations d'images et la partie de calcul permettant d'aller vers une interprétation des images traitées. Cette phase d'interprétation est d'ailleurs de plus en plus intégrée dans le traitement d'images, en faisant appel notamment à l'intelligence artificielle pour manipuler des connaissances, principalement sur les informations dont on dispose à propos de ce que représentent les images traitées (connaissance du ■ domaine ■).

La compréhension du traitement d'images commence par la compréhension de ce qu'est une image. Le mode et les conditions d'acquisition et de numérisation des images traitées conditionnent largement les opérations qu'il faudra réaliser pour extraire de l'information. En effet, de nombreux paramètres entrent en compte.

Chapitre 1

État de l'art

1.1 L'image numérique

L'image est définie comme étant une fonction $f(x, y)$ à deux dimensions, où x et y sont les coordonnées spatiales, et l'amplitude à tout point (x, y) correspond à l'intensité ou au niveau de gris.
Lorsque les points (x, y) et l'amplitude sont discrétisés, on parle d'image numérique ou digitale. Dans ce dernier cas la fonction f est remplacée par la lettre I et le couple (x, y) par le couple (i, j). Ainsi, une image numérique est constitué d'un ensemble de points appelés **pixels** (abréviation de **PIC**ture **El**ement). Les pixels sont approximativement rectangulaires, parfois carrés. Le pixel représente alors le plus petit élément constitutif d'une image numérique. L'ensemble de ces pixels est contenu dans un tableau à deux dimensions constituant l'image, voir la figure 1.1.

1.1.1 Types d'images numériques

Selon la représentation des couleurs, nous distinguons quatre types d'images :

Images binaires

Pour une image binaire, chaque pixel est soit blanc soit noir, voir figure 1.2. Puisqu'il y a uniquement deux valeurs pour chaque pixel, un seul bit est utilisé pour le coder.

Images en niveaux de gris

Pour une image en niveaux de gris, chaque pixel est un niveau de gris, allant de 0 (noir) à 255 (blanc). Cet intervalle de valeurs signifie que chaque

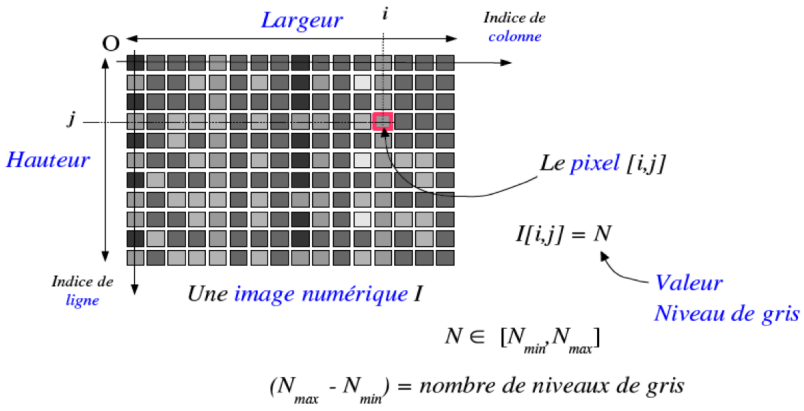


FIGURE 1.1 – Image numérique

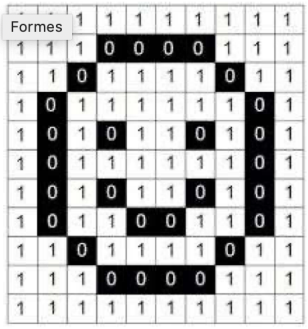


FIGURE 1.2 – Image binaire

pixel est codé sur 8 bits (un octet), voir figure 1.3.

FIGURE 1.3 – Image en niveaux de gris

Images en couleurs (ou RGB)

Pour une image en couleurs, chaque pixel possède une couleur décrite par la quantité de rouge (R), vert (G) et bleu (B), voir figure 1.4. Chacune de ces trois composantes est codée sur l'intervalle $[0, 255]$, ce qui donne $255^3 = 16\ 777\ 216$ couleurs possibles. Il faut donc 24 bits pour coder un pixel.

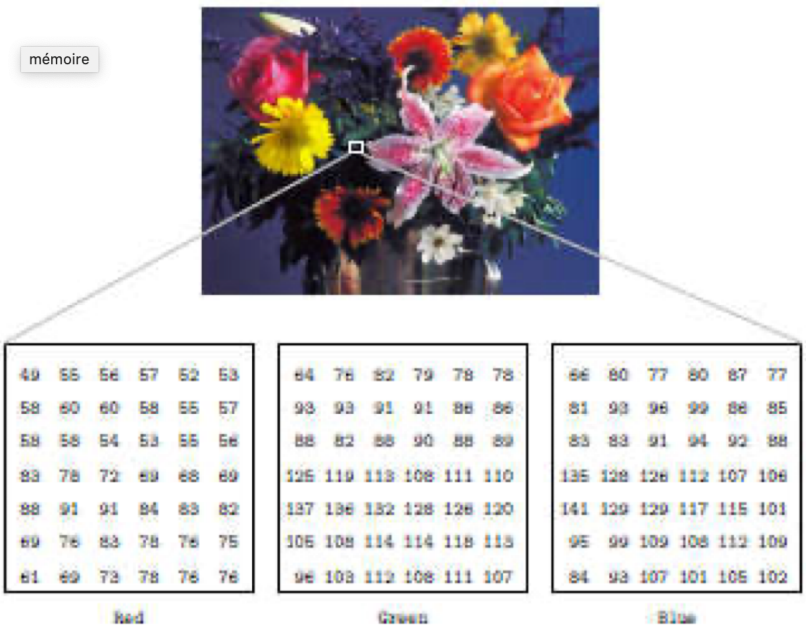


FIGURE 1.4 – Image en couleurs

1.1.2 Caractéristiques d une image

Les pixels

Une image numérique est constituée d'un ensemble de points appelés pixels (abréviation de PICture Element) pour former une image.

Le pixel représente ainsi le plus petit élément constitutif d'une image numérique.

L'ensemble de ces pixels est contenu dans un tableau à deux dimensions constituant l'image :

Le codage des couleurs

En plus de sa définition, une image numérique utilise plus ou moins de mémoire selon le codage des informations de couleur qu'elle possède.

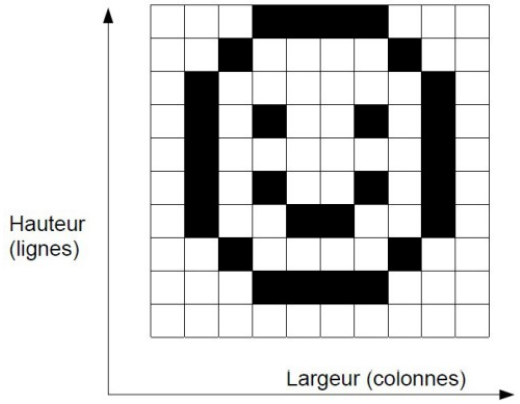


FIGURE 1.5 – Exemple d une image matricielle ■ 10 x 11 ■.

C'est ce que l'on nomme le codage de couleurs ou profondeur des couleurs, exprimé en bit par pixel (bpp) : 1, 4, 8, 16 bits...

En connaissant le nombre de pixels d'une image et la mémoire nécessaire à l'affichage d'un pixel, il est possible de définir exactement le poids que va utiliser le fichier image dans une mémoire informatique.

Formule pour calculer le poids d'une image en octet :

Nombre de pixel total × codage couleurs (octet) = Poids (octet)

Petit rappel du code binaire, utilisé par l'ordinateur pour enregistrer des informations.

■ On sait que : ■

— 1bit = 2 possibilitées ; (0 ou 1) possibilité de coder 2 couleurs par pixel

— 2bits = 4 possibilitées (00 / 01 / 10 / 11) possibilité de coder 4 couleurs par pixel

— 4bits = 16 possibilitées (0000 / 0001 / 0011 / 0111 / 1111 / 0010 /0110 / 1010 / ....) possibilité de coder 16 couleurs par pixel

— 8bits = 256 possibilités —- codage de 256 couleurs par pixel

Histogramme d une image numérique

Définition

En imagerie numérique, l'histogramme représente la distribution des intensités (ou des couleurs) de l'image. C'est un outil fondamental du traite-

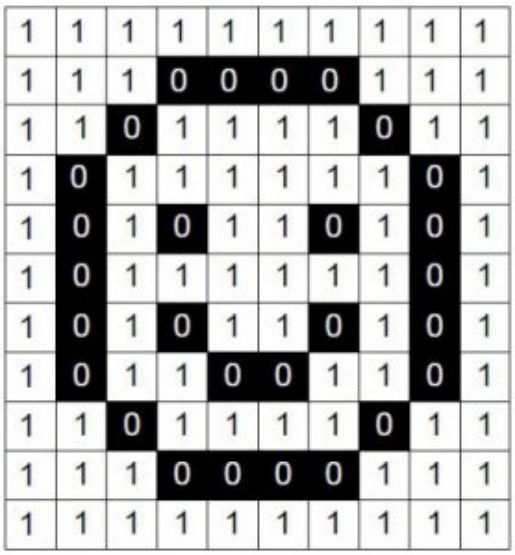


FIGURE 1.6 – Exemple d'image noir et blanc ou chaque pixel et codé avec un seul bit

ment d'images, avec de très nombreuses applications. Les histogrammes sont aussi très utilisés en photographie et pour la retouche d'images.

Les histogrammes sont en général normalisés, en divisant les valeurs de chaque classe par le nombre total de pixels de l'image **n** . La valeur d'une classe varie alors entre 0 et 1, et peut s'interpréter comme la probabilité d'occurrence de la classe dans l'image. L'histogramme peut alors être vu comme une densité de probabilité1. Pour une image x en niveaux de gris codée sur L niveaux, on définit n_k le nombre d'occurrences du niveau x_k. La probabilité d'occurrence d'un pixel de niveau x_k dans l'image est :

$$p_x(x_k) = p(x = x_k) = \frac{n_k}{n}, \quad 0 \leq k < L$$

avec **n** le nombre total de pixels de l'image, et p_x définit alors l'histogramme normalisé sur [0,1].

Propriétés d'histogramme :

Les histogrammes sont résistants à un certain nombre de transformations sur l'image. Ils sont invariants aux rotations et aux translations, ainsi que

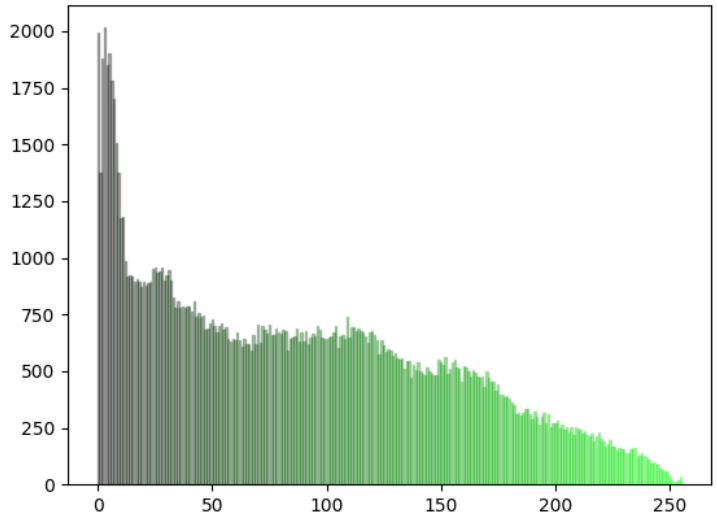


FIGURE 1.7 – Histogramme des degrés du vert dans une image

dans une moindre mesure aux changements de point de vue, et aux changements d'échelle. Les histogrammes sont en revanche sensibles aux changements d'illumination et aux conditions d'éclairage.

Pour un histogramme normalisé, la valeur de chaque classe s'interprète comme la probabilité d'occurrence de la classe dans l'image. Un histogramme normalisé somme à 1.

1.2 Machine Learning

Le Machine Learning ou apprentissage automatique est un domaine scientifique, et plus particulièrement une sous-catégorie de l'intelligence artificielle. Elle consiste à laisser des algorithmes découvrir des "patterns", à savoir des motifs récurrents, dans les ensembles de données. Ces données peuvent être des chiffres, des mots, des images, des statistiques, ...

Les algorithmes de Machine Learning apprennent de manière autonome à effectuer une tâche ou à réaliser des prédictions à partir de données et améliorent leurs performances au fil du temps. Une fois entraîné, l'algorithme pourra retrouver les patterns dans de nouvelles données.

Les algorithmes de l'apprentissage automatique se classent en deux catégories principales ; on distingue entre un algorithme supervisé et un algo-

rithme non supervisé.

1.2.1 Apprentissage supervisé

Pour l'apprentissage supervisé la machine s'apprend sur des données dont la nature des résultats est connue au préalable, c'est exactement le principe d'un enseignant supervisant le processus d'apprentissage de ses élèves ou il connait les réponses correctes et il est là pour corriger les erreurs qu'ils commettent. Cet type d'apprentissage est généralement utile dans les problèmes de **classification** et **régression**.

Régression

En mathématiques, la régression recouvre plusieurs méthodes d'analyse statistique permettant d'approcher une variable à partir d'autres qui lui sont corrélées. Par extension, le terme est aussi utilisé pour certaines méthodes d'ajustement de courbe.

En apprentissage automatique, on distingue les problèmes de régression des problèmes de classification. Ainsi, on considère que les problèmes de prédiction d'une variable quantitative sont des problèmes de régression tandis que les problèmes de prédiction d'une variable qualitative sont des problèmes de classification.

La résolution de problèmes de régression est l'une des applications les plus courantes des modèles d'apprentissage automatique, en particulier dans l'apprentissage automatique supervisé. Les algorithmes sont formés pour comprendre la relation entre les variables indépendantes et un résultat ou une variable dépendante. Le modèle peut ensuite être exploité pour prédire le résultat de données d'entrée nouvelles et inédites, ou pour combler une lacune dans les données manquantes.

Parmi les types de régression, on pourra citer :

- Le modèle de régression le plus connu est le modèle de **régression linéaire.**
- Lorsque le modèle n'est pas linéaire, on peut effectuer une régression approchée par des algorithmes itératifs, on parle de **régression non linéaire**.
- Si on s'intéresse au quantile conditionnel de la distribution de la variable aléatoire y sachant le vecteur de variables aléatoires y, on utilise un modèle de **régression quantile**.
- Si la variable expliquée est une variable aléatoire binomiale, il est courant d'utiliser **une régression logistique ou un modèle probit.**

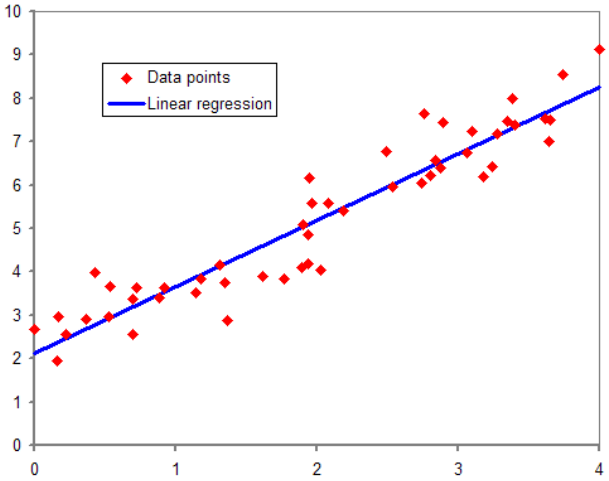


FIGURE 1.8 – Exemple de l'application de la régression

— Si la forme fonctionnelle de la régression est inconnue, on peut utiliser un modèle de **régression non paramétrique.**

Classification

La classification est une tâche qui nécessite l'utilisation d'algorithmes d'apprentissage automatique qui apprennent à attribuer une étiquette de classe aux exemples du domaine du problème. Un exemple facile à comprendre consiste à classer les e-mails comme "spam" ou "non spam".

Il existe de nombreux types de tâches de classification que vous pouvez rencontrer dans l'apprentissage automatique et des approches spécialisées de la modélisation qui peuvent être utilisées pour chacune.

1.2.2 Apprentissage non supervisé

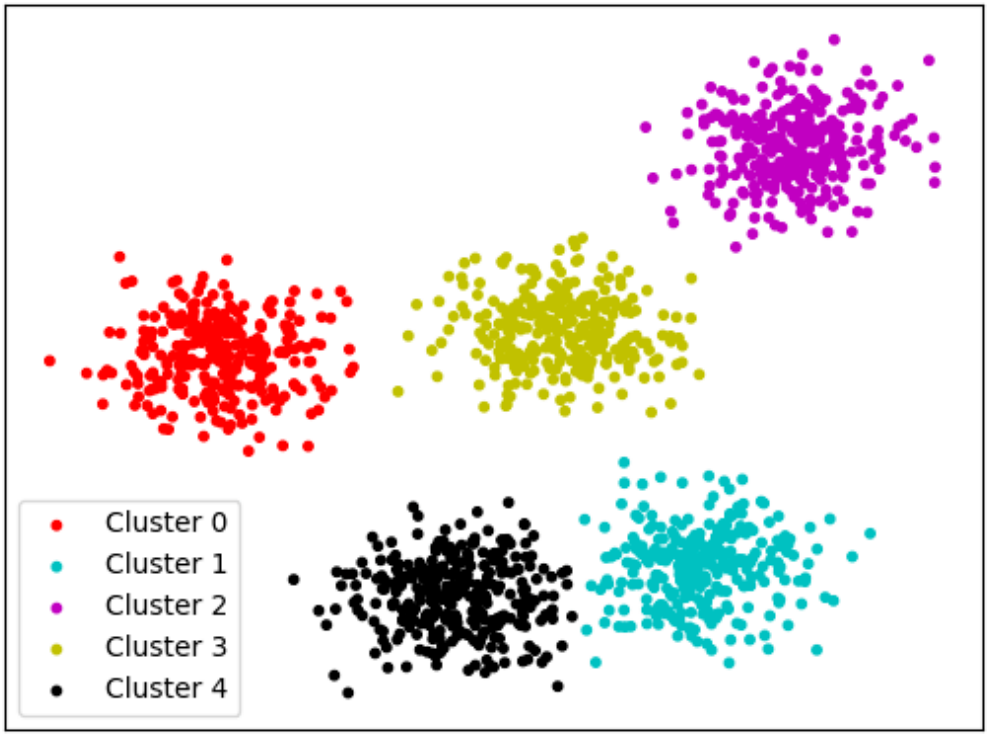
Dans le cas de l'apprentissage non supervisé, l'algorithme apprend d'une façon totalement autonome, c'est à dire on communique des données à la machine sans lui révéler la nature des résultats, et donc c'est à l'algorithme lui même de modéliser et de représenter la structure intéressante des données.
On peut réduire les problèmes d'apprentissage non supervisés à deux problèmes ; le problème de "**Clustering**" pour lequel on attend de la machine à faire des regroupements des données les plus homogènes possibles, et le problème de "**Association**" qui consiste à trouver des liens entre les données.

Clustering

Définition

Le partitionnement de données (ou data clustering en anglais) est une méthode en analyse des données. Elle vise à diviser un ensemble de données en différents ■ paquets ■ homogènes, en ce sens que les données de chaque sous-ensemble partagent des caractéristiques communes, qui correspondent le plus souvent à des critères de proximité (similarité informatique) que l'on définit en introduisant des mesures et classes de distance entre objets.
Pour obtenir un bon partitionnement, il convient d'à la fois :

— minimiser l'inertie intra-classe pour obtenir des grappes (cluster en anglais) les plus homogènes possibles
— maximiser l'inertie inter-classe afin d'obtenir des sous-ensembles bien différenciés.



Par ailleurs, nous venons de voir les trois grandes familles de catégories du Machine Learning (Régression, Classification, et Clustering). Désormais, vous saurez plus intuitivement de quelle catégorie il s'agit quand vous êtes confrontés à un problème d'apprentissage automatique,

mais au long de ce projet de rechercher on s intéresse specialement a la methode du Clustering et ces applications sur l analyse des images en utilisant le language de programmation Python.

Chapitre 2

Algorithme K-means et Analyse des images

2.1 Algorithme K-means

2.1.1 Étapes

L'algorithme k-moyennes à 3 pas essentiels :

1. **Initialisation :** une fois le nombre de groupes, k choisi, k barycentres sont établis dans l'espace de données, par exemple en les choisissant au hasard.
2. **Affectation des objets aux barycentres :** chaque objet des données est affecté à son barycentre le plus proche.
3. **Mise à jour des barycentres :** La position du barycentre de chaque groupe est mise à jour en prenant comme nouveau barycentre la position moyenne des objets appartenant audit groupe.

répéter étape 2 et 3 jusqu'à ce que les barycentres ne bougent pas ou se déplacent en dessous d'une distance seuil à chaque étape.

Cela s'interprète mathématiquement a un problème d'optimisation et la fonction a optimiser (minimiser) est la somme des distances quadratiques de chaque objet à son barycentre du groupe qu'y appartient.

$$\sum e_j = \sum_{X_j \in S_i} \|x_j - \mu_i\|^2$$

Les objets sont représentés avec **d** vecteurs de dimension **n**, $x = (x_1, x_2, \ldots, x_n)$ et le résultat de l'algorithme est **k** groupes ou la distance entre chaque vec-

teur et le barycentre du groupe qu'y appartient est minimisée l'ensemble des groupes étant $\mathbf{S} = \{S_1, S_2, \ldots, S_k\}$. le problème peut se formuler :

$$\min_{\mathbf{s}}(E(\mu_i)) = \min_{\mathbf{s}} \sum_{i=1}^{k} \sum_{X_j \in S_i} \|\boldsymbol{x_j} - \boldsymbol{\mu_i}\|^2 = \min \sum_{i=1}^{k} \sum_{j=1}^{n} a_{ij} \|x_j - \mu_i\|^2$$

$$a_{ij} = 1 \quad si \quad x_j \in S_i, \ avec \sum_{i=1}^{k} a_{ij} = 1$$

Dans chaque mise à jour du centroïde, du point de vue mathématique, nous imposons la condition nécessaire extrême (minimale, dans ce cas) à la fonction $E(\mu_i)$ que, pour cette fonction quadratique est :

$$\frac{\partial E}{\partial \mu_i} = 0 \Longrightarrow {\mu_i}^{(t+1)} = \frac{1}{|{S_i}^{(t)}|} \sum_{x_j \in {S_i}^{(t)}} \boldsymbol{x_j}$$

et la solution consiste à prendre la moyenne de chaque élément de groupe comme nouveau centroide(barycentre). Nous avons utilisé la méthode de descente de gradient.

Les principaux avantages de la méthode des k-moyennes sont qu'elle est simple et rapide. Mais il est obligatoire de décider la valeur de **k** et le résultat final dépend de l'initialisation des barycentres. De plus, il ne converge pas nécessairement vers le minimum global mais vers un minimum local.

2.1.2 Convergence de l'algorithme :

Il y a un nombre fini de partitions possibles à **k** classes. De plus, chaque étape de l'algorithme fait strictement diminuer la fonction de coût, positive, et fait découvrir une meilleure partition. Cela permet d'affirmer que l'algorithme converge toujours en temps fini, c'est-à-dire termine.

Le partitionnement final n'est pas toujours optimal. De plus le temps de calcul peut être exponentiel en le nombre de points, même dans le plan. Dans la pratique, il est possible d'imposer une limite sur le nombre d'itérations ou un critère sur l'amélioration entre itérations.

comme on peut voir sur la figure on obtient un résultat efficace en seulement 6 opérations en appliquant kmeans a une certaines de points $(x_1, x_2, \ldots, x_n)$

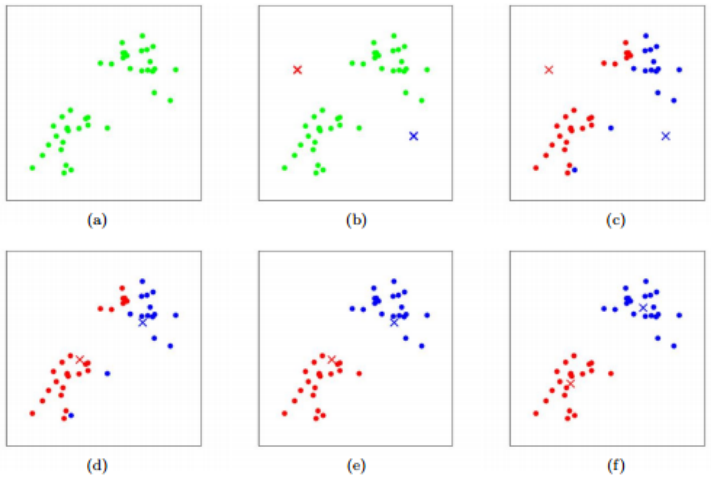


FIGURE 2.1 – exemple d'application de Kmeans sur des points de $\mathbb{R}^2$

avec $x_i \in \mathbb{R}^2$
Mais cette efficacité dépend d'une grande degré au choix des barycentres initiales ce qui est très difficile surtout si on as pas d'informations sur les données qu'on doit travailler avec

2.1.3 Avantages et inconvénients pour l'apprentissage

Un inconvénient possible des k-moyennes pour le partitionnement est que les clusters dépendent de l'initialisation et de la distance choisie.

Le fait de devoir choisir a priori le paramètre k peut être perçu comme un inconvénient ou un avantage. Dans le cas du calcul des sac de mots par exemple, cela permet de fixer exactement la taille du dictionnaire désiré. Au contraire, dans certains partitionnements de données, on préférera s'affranchir d'une telle contrainte.

Avantages

- -Simple
- -Flexible
- -Convient aux gros data sets
- -Efficace
- -Clusters proches
- -Facile à interpréter
- -Faible coût de calcul

Inconvénients

— -Ensemble non optimal de clusters
— -Ordre des valeurs
— -Limitation des calculs
— -Traiter les données numériques
— -Problèmes de prédiction
— -Spécifiez les valeurs K

Conclusion

La classification K-means est une technique largement utilisée pour l'analyse par clusters de données. Cet algorithme est simple à comprendre. En outre, il fournit des résultats d'entraînement rapidement. Cependant, ses performances ne sont généralement pas aussi compétitives que celles des autres techniques de classification sophistiquées, car de légères variations dans les données pourraient entraîner une variance des résultats élevée.

2.2 Application de K-means pour l'analyse des images

2.2.1 Outils utilisé pour l'analyse des images

Normalement on ne calcule pas les étapes d'un algorithme avec la main pour cela on utilise des outils scientifiques programmés en certains langages de programmation , heureusement pour le machine learning y' as plusieurs langage pour faciliter le travail en ce domaine, on trouve parmi eux :

— **R :** R est un langage de programmation et un logiciel libre destiné aux statistiques et à la science des données soutenu par la R Foundation for Statistical Computing

— **Matlab :** MATLAB (■ matrix laboratory ■) est un langage de script émulé par un environnement de développement du même nom , il est utilisé à des fins de calcul numérique. Développé par la société The MathWorks, MATLAB permet de manipuler des matrices, d'afficher des courbes et des données, de mettre en œuvre des algorithmes, de créer des interfaces utilisateurs, et peut s'interfacer avec d'autres langages comme le C, C++, Java, et Fortran.

— **Python :** Python est un langage de programmation qui peut s'utiliser dans de nombreux contextes et s'adapter à tout type d'utilisation grâce à des bibliothèques spécialisées. Il est cependant particulièrement utilisé comme langage de script pour automatiser des tâches simples mais fastidieuses,

2.2.2 Installation dans l'ordinateur

heureusement pour notre algorithme Kmeans on est issu d un package open source nommé **scikit** (science kit) ou plusieurs outils du machine learning sont prédéfinies en plus ces outils sont faciles a interpréter dans un programme PYTHON pour résoudre les problèmes qu'on veut.
parmi ces outils on a la classe **KMeans** :

— Pour installer le package **scikit** dans votre ordinateur vous devez insérer une des commandes suivantes.

— sur **windows** :

```
pip install -U scikit-learn
```

— sur **linux** :

```
pip3 install -U scikit-learn
```

— sur **macOS** :

```
pip install -U scikit-learn
```

— pour importer le package sur votre projet :

```
from sklearn.* import *(Classe Voulu)
```

exemple pour importer spécifiquement seulement la classe **KMeans** après avoir installé le package **scikit** bien sur on insère le code suivant dans le script python qu'on travaille sur :

```
from sklearn.cluster import KMeans
```

2.2.3 Utilisation de la Classe KMeans

Voici le minimum code python pour que Kmeans nous donne des résultats :

```python
from sklearn.cluster import KMeans

#important fixer le k
k = 3
kmeans = KMeans(n_clusters=k)

#la methode KMeans.fit() pour inserer les données ,X une liste des vecteur de R² par exemple,
kmeans.fit(X)
centroids = kmeans.cluster_centers_
labels = kmeans.labels_
```

FIGURE 2.2 – interprétation de KMeans sur python

On commence d'abord par initialiser **k** les nombre de groupes voulues(clusters), et puis on crée un objet de classe **KMeans** avec le paramètre **k** déjà initialisé, la méthode `KMeans.fit()` est où tout le calcul ce passe , et finalement on retourne les résultats sous forme de tableaux ou Listes ou seulement une seule valeur réel :

- L'attribut `cluster_centers` : nous retournes les **k** barycentres résultantes de l'application de l'algorithme sur certaines données.
- L'attribut `labels_` : nous retournes une liste qui contient le groupes résultantes de après exécution de l'algorithme mais d'une manière indirecte par exemple :
  si le **i**-ème vecteur appartient au **n**-ème groupe , le **i**-ème élément de `labels_` aura pour valeur **n** au lieu du **i**-ème vecteur pour des raisons d'optimisation et de réduction du temps d'exécution.
  et ainsi de suite ...
- L'attribut `inertia_` nous donne le pourcentage de l'erreur accumulée après la fin de l algorithme .
- On peut aussi spécifier le nombre d'itérations maximale qu'on veut avec le paramètre `max_iter =` dans la déclaration de l'objet du classe **KMeans**
- Si on veut exécuter l'algorithme plusieurs fois avant de donner le résultat il suffit de ajouter le paramètre `n_init =` dans la déclaration et automatiquement le programme retournera le meilleur résultat avec le minimum d'erreur.

2.2.4 La reconnaissance de l image

L'émergence de l'intelligence artificielle fait réfléchir sur le potentiel d'évolution de nos industries et de nos métiers. De plus en plus, les entreprises ont recours à la Computer Vision (vision par ordinateur), et en particulier à la reconnaissance d'image, pour améliorer leurs processus et augmenter leur productivité. on va donc vous expliquer en quelques mots ce qu'est la reconnaissance d'image, son fonctionnement ainsi que ses différents usages.

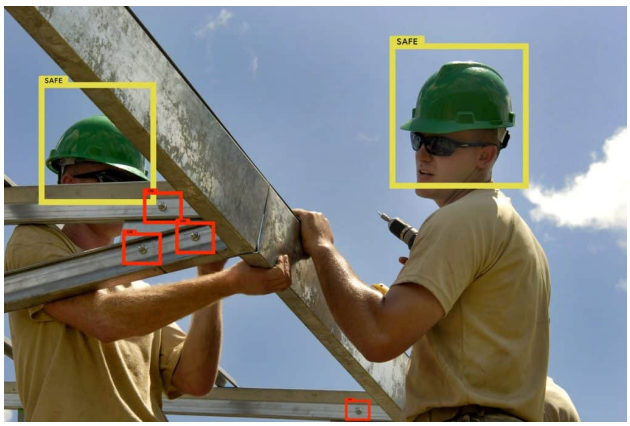


FIGURE 2.3 – Exemple d'application de la reconnaissance de l image pour assurer la sécurité des ouvriers

2.2.5 Définition

La reconnaissance d'image, sous-catégorie de la Computer Vision et de l'Intelligence Artificielle, représente un ensemble de méthodes de détection et d'analyse d'images pour permettre l'automatisation d'une tâche spécifique. Il s'agit d'une technologie qui est capable d'identifier des lieux, des personnes, des objets et plusieurs autres types d'éléments au sein d'une image et d'en tirer des conclusions en les analysant.

La reconnaissance photo ou vidéo peut être réalisée à des degrés de précision différents. Il existe donc différentes "tâches" que la reconnaissance d'image peut
effectuer :

— **La classification.** Il s'agit de l'identification de la "classe", autrement dit la catégorie, à laquelle une image appartient. Une image ne peut avoir qu'une seule classe.

— **Le tagging.** Aussi appelé “labellisation”, il s’agit d’une tâche de classification mais à un degré de précision plus élevé. Cela va permettre de reconnaître la présence de plusieurs concepts ou objets au sein d’une image. Il est par conséquent possible d’attribuer un ou plusieurs tags à une image en particulier.

— **La détection.** Cela est nécessaire lorsque l’on souhaite situer un objet dans une image. Une fois la localisation réussie, un rectangle, appelée en anglais bounding box, va encadrer l’objet en question.

— **La segmentation.** Elle représente également une tâche de détection. La segmentation peut situer au pixel près un élément sur une image. Car dans certains cas, la précision ne peut être négligée, tel que pour le développement des voitures autonomes.

Chapitre 3

Résultats numériques

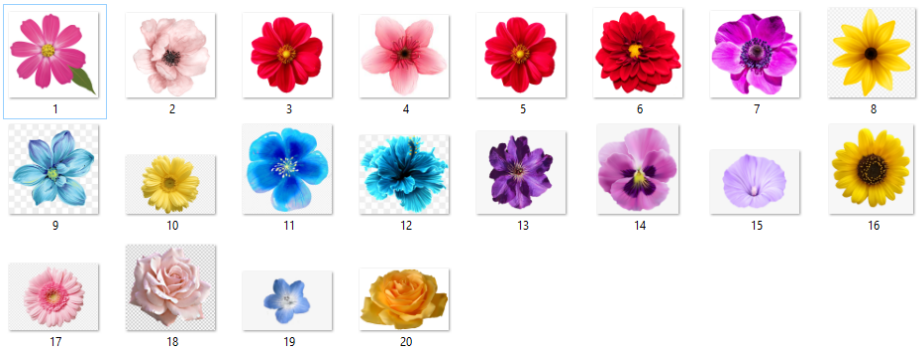
3.1 Labellisation des images en utilisant K-means

Définition

La labellisation (ou classification) est l'application directe du Kmeans dans le domaine de l'analyse des images numériques, cette dernière consiste tous simplement a grouper une base de données des images an **k** groupes en précisant le **k** a priori , on peut faire cela en entrant des données sous formes de vecteur qui ont une relation directe avec les images qu'on veut classer appelées features.

Application et exemple

On peut appliquer la classification sure les différents images suivantes :



On prends en compte un vecteur de $\mathbb{R}^6$: ou les trois premiers composantes sont les moyennes des valeurs de R(rouge), G(vert), B(bleu) de chaque image, et les trois autres composantes sont les l'écart type(Standard deviation) aussi

```python
def getFeatureMean(path):
    img = Image.open(path)
    img = img.convert('RGB')
    a = np.array(img)
    w, h, c = a.shape
    a1 = a.reshape(w * h, c)
    mr, mg, mb = np.mean(a1[:, 0]), np.mean(a1[:, 1]), np.mean(a1[:, 2])
    return [mr, mg, mb]
def getStandardDev(path):
    img = Image.open(path)
    img = img.convert('RGB')
    a = np.array(img)
    r, g, b = img.split()
    r, g, b = np.array(r), np.array(g), np.array(b)
    stdr, stdg, stdb = np.std(r), np.std(g), np.std(b)

    return [stdr, stdg, stdb]
```

de chaque R(rouge), G(vert), B(bleu) de chaque image.
et voici les codes pour extraire ce vecteur sur Python :

On fusionne les résultats pour obtenir notre vecteur :

```python
def getFeatures(path):
    m = getFeatureMean(path)
    s = getStandardDev(path)
    features = m + s
    return features
```

Après on applique Kmeans avec le $\mathbf{K} = 5$:

```python
n = 5
kMeans = KMeans(n_clusters=n)
v = np.array(list(features.values()))
print(v.shape)
#print(v)
kMeans.fit(v)
centroids = kMeans.cluster_centers_
labels = kMeans.labels_
```

On organise les résultats et on les affiche utilisant l'interface d'affichage `matplotlib.pyplot` en utilisant une simple boucle et la fonction `view_clusters` que j'ai déjà défini :

```
clusters = []
for i in range(n):
    cluster = []
    for j in range(len(data)):
        if labels[j] == i:
            cluster.append(data[j])
    clusters.append(cluster)
```

```
for i in range(n):
    view_cluster(clusters[i])
```

les résultats affichées sont :

FIGURE 3.1 – Cluster 1 :

FIGURE 3.2 – Cluster 2 :

FIGURE 3.3 – Cluster 3 :

FIGURE 3.4 – Cluster 4 :

FIGURE 3.5 – Cluster 5 :

3.2 Quantification d'images numériques avec k-means

Définition

La quantification est une technique de compression avec perte qui consiste à regrouper toute une groupe de valeurs en une seule. Si on quantifie la couleur

d'une image, on réduit le nombre de couleurs nécessaires pour la représenter et la taille du fichier diminue.

Application et exemple

D'abord on transforme l image qu'on veux quantifier en une matrice où chaque composante est égale a un pixel est la valeur stockée dans ce dernier est un vecteur de $\mathbb{R}^3$ représentant les niveaux de (R, G, B) , et après en prend la matrice et on la linéarise sous forme d'un tableau qui as un longueur égale a la définition de cette image où chaque élément de ce tableau est aussi un vecteur de $\mathbb{R}^3$.

```
img = Image.open(path)
img = img.convert('RGB')
a  = np.asarray(img, dtype=np.float32)/255
x, y, z = a.shape
a1 = a.reshape(x*y, z)
```

Après on applique Kmeans sur le tableau qu'on a extrait , et on change la valeur de chaque couleur par le barycentre du cluster qu'y appartient , et on reconstruit l'image résultante.

```
n = n_clusters
kMeans = KMeans(n_clusters=n)
kMeans.fit(a1)
centroids = kMeans.cluster_centers_
labels = kMeans.labels_
a2 = centroids[labels]
a3 = a2.reshape(x,y,z)
a4 = np.floor(a3*255)
a5 = a4.astype(np.uint8)
I1 = Image.fromarray(a5)
I1.save("test.jpg")
```

Exemple

FIGURE 3.6 – l'image avant la quantification.

FIGURE 3.7 – l'image après la réduction de ces couleurs en 3.

3.3 Segmentation d'image avec kmeans

Définition

la segmentation divise une image en régions avec des propriétés internes cohérentes. Vous pouvez segmenter une image en utilisant la couleur.

Le processus est similaire à la quantification d'image. La différence est le but du regroupement de pixels : en segmentation, on regroupe les pixels pour séparer les éléments significatifs d'une image et ainsi, pouvoir extraire certaines informations quantitatives. Par exemple, calculer la taille d'une tumeur à partir d'images médicales, le pourcentage de mica dans une roche granitique, la superficie d'un lac à partir d'une photo aérienne.

Application et exemple

On va utiliser la segmentation pour calculer la surface d'un lac depuis une photo aérienne.
Pour commencer, d'abord on transforme l'image tout d'abord en la rendre en mode L c.a.d qu'elle contient que les dégrées du gris et après en réduisant ces couleurs en utilisant la quantification citée dans la dernière section.

Comme cela on distingue le lac des autres élément, ce dernier est représenté par la surface noire.

FIGURE 3.8 – l'image du lac avant la quantification.

FIGURE 3.9 – l'image du lac après la quantification.

```
a3 = a2.reshape(x, y)
a4 = (a3 - np.min(a3))/(np.max(a3)-np.min(a3))*255
a5 = a4.astype(np.uint8)
I2 = Image.fromarray(a5)
w, h =I2.size
colors = I2.getcolors(w * h)

float(totalArea) * float(colors[0][0])/float(w * h)
```

ou `totalArea` représente la surface totale en Km et `colors[0][0]` est combien de pixels occupe le noir en la divisant par `w * h` on obtient le pourcentage du noir par rapport a l'image totale.

les résultat finale est obtenu en Km , est représente la surface du lac .

Bibliographie

[1] https ://fr.wikipedia.org/wiki/Apprentissage_non_supervisé

[2] https ://mrmint.fr/machine-learning-applications

[3] https ://fr.wikipedia.org/wiki/Traitement_d'images

[4] https ://fr.wikipedia.org/wiki/Histogramme_(imagerie_numérique)

[5] https ://www.lossendiere.com/2016/08/31/caracteristiques-dune-image-numerique/

[6] https ://fr.wikipedia.org/wiki/K-moyennes

[7] https ://analyticsinsights.io/k-means/

[8] https ://www.unioviedo.es/compnum/labs/new/kmeans.html# :˜ :text=K-means%20is%20an%20unsupervised,the%20group%20or%20cluster%20centroid

[9] https ://deepomatic.com/fr/quest-ce-que-la-reconnaissance-dimage

[10] https ://fr.wikipedia.org/wiki/Histogramme_(imagerie_numérique)

lien pour le code python et la base de données :

[11] https ://github.com/ouaala-ayoub/PFE