Intelligence artificielle (IA)

Algorithmes des k plus proches voisins et des k moyennes

Prof d'info

Lycée Thiers

1. Intelligence artificielle
   - Apprentissage
   - Notion de partition
2. Apprentissage supervisé
   - Algorithme des k plus proches voisins
   - Matrice de confusion
3. Apprentissage non supervisé
   - Algorithme des K-moyennes

Crédits

- Wikipedia : K-moyennes

Crédits

- Wikipedia : K-moyennes
- Wikipedia Intelligence artificielle

Crédits

- Wikipedia : K-moyennes
- Wikipedia Intelligence artificielle
- Informatique Tronc Commun (Serge Bays - Ellipse)

Concept

- *Intelligence artificielle* (IA) : ensemble des théories et des techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence humaine

Concept

- *Intelligence artificielle* (IA) : ensemble des théories et des techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence humaine
- Concept flou dont les bornes varies selon les époques. Ainsi, la recherche de plus court chemin était considéré comme de l'IA dans les années 50 mais n'est plus considéré que comme un programme d'algorithmique.

Concept

- *Intelligence artificielle* (IA) : ensemble des théories et des techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence humaine
- Concept flou dont les bornes varies selon les époques. Ainsi, la recherche de plus court chemin était considéré comme de l'IA dans les années 50 mais n'est plus considéré que comme un programme d'algorithmique.
- Une origine probable : un article d'Allan turing ≪ Computing Machinery and Intelligence ≫ (1950) dans lequel il propose une expérience visant à trouver à partir de quand une machine deviendrait ≪ consciente ≫

Concept

- *Intelligence artificielle* (IA) : ensemble des théories et des techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence humaine
- Concept flou dont les bornes varies selon les époques. Ainsi, la recherche de plus court chemin était considéré comme de l'IA dans les années 50 mais n'est plus considéré que comme un programme d'algorithmique.
- Une origine probable : un article d'Allan turing ≪ Computing Machinery and Intelligence ≫ (1950) dans lequel il propose une expérience visant à trouver à partir de quand une machine deviendrait ≪ consciente ≫
- Une autre origine problable : en 1949, Warren Weaver publie un mémorandum sur la traduction automatique des langues qui suggère qu'une machine peut faire une tâche qui relève typiquement de l'intelligence humaine.

Apprentissage automatique

- *Apprentissage automatique* (AA) : Domaine de l'IA fondé sur des techniques mathématiques et statistiques pour donner aux ordinateurs la capacité d'≪ apprendre ≫ à partir de données : améliorer leurs performances à résoudre des tâches sans être explicitement programmés pour chacune.

Apprentissage automatique

- *Apprentissage automatique* (AA) : Domaine de l'IA fondé sur des techniques mathématiques et statistiques pour donner aux ordinateurs la capacité d'≪ apprendre ≫ à partir de données : améliorer leurs performances à résoudre des tâches sans être explicitement programmés pour chacune.
- L'américain Arthur Samuel est le premier à faire usage de l'expression ≪ machine learning ≫. Il concernait son programme de jeu de dame (écrit en 1952 pour IBM) qui s'améliorait en jouant.

Principe

- Construire un *modèle* à partir de *données*.

Principe

- Construire un *modèle* à partir de *données*.
- Phase d'*apprentissage* : la machine explore une base de données.

Principe

- Construire un *modèle* à partir de *données*.
- Phase d'*apprentissage* : la machine explore une base de données.
  - Apprentissage *supervisé* : chaque donnée explorée est accompagnée d'une *étiquette* (par exemple, pour chaque photo, on indique si un chat est présent ou non).

Principe

- Construire un *modèle* à partir de *données*.
- Phase d'*apprentissage* : la machine explore une base de données.
  - Apprentissage *supervisé* : chaque donnée explorée est accompagnée d'une *étiquette* (par exemple, pour chaque photo, on indique si un chat est présent ou non).
  - Apprentissage *non supervisé* : la machine effectue elle-même une classification ; elle identifie la structure (plus ou moins cachée) des données.

Principe

- Construire un *modèle* à partir de *données*.
- Phase d'*apprentissage* : la machine explore une base de données.
  - Apprentissage *supervisé* : chaque donnée explorée est accompagnée d'une *étiquette* (par exemple, pour chaque photo, on indique si un chat est présent ou non).
  - Apprentissage *non supervisé* : la machine effectue elle-même une classification ; elle identifie la structure (plus ou moins cachée) des données.
- Dans une seconde phase, dite *phase de test*, on compare la prédiction faite par la machine et la véritable réponse.

Régression VS Classement

- Méthode de *classement* : il s'agit de prédire une valeur qualitative.
  Exemple : c'est un chat, ce n'est pas un chat.

Régression VS Classement

- Méthode de *classement* : il s'agit de prédire une valeur qualitative.
  Exemple : c'est un chat, ce n'est pas un chat.
- Méthode de *régression* : il s'agit de déterminer une valeur quantitative, le plus souvent continue.
  Par exemple, une probabilité pour une machine de tomber en panne (15 %, 20 %, etc.) ou le prix de vente idéal d'un appartement en fonction de critères comme la surface, le quartier, etc.

Partition

- Notion associée aux méthodes de classement.

Partition

- Notion associée aux méthodes de classement.
- Si les données sont contenues dans un ensemble E on décompose E en classes d'équivalences. À l'arrivée d'une nouvelle donnée d, le programme doit trouver à quelle classe de E appartient d.

Partition

- Notion associée aux méthodes de classement.
- Si les données sont contenues dans un ensemble E on décompose E en classes d'équivalences. À l'arrivée d'une nouvelle donnée d, le programme doit trouver à quelle classe de E appartient d.
- On construit donc une *partition* de E, c'est à dire une famille non vide d'ensembles non vides telle que :

Partition

- Notion associée aux méthodes de classement.
- Si les données sont contenues dans un ensemble E on décompose E en classes d'équivalences. À l'arrivée d'une nouvelle donnée d, le programme doit trouver à quelle classe de E appartient d.
- On construit donc une *partition* de E, c'est à dire une famille non vide d'ensembles non vides telle que :
  - la réunion de tous les éléments de la partition constitue E

Partition

- Notion associée aux méthodes de classement.
- Si les données sont contenues dans un ensemble E on décompose E en classes d'équivalences. À l'arrivée d'une nouvelle donnée d, le programme doit trouver à quelle classe de E appartient d.
- On construit donc une *partition* de E, c'est à dire une famille non vide d'ensembles non vides telle que :
  - la réunion de tous les éléments de la partition constitue E
  - l'intersection de deux éléments de la partition est vide.

Partition

- Notion associée aux méthodes de classement.
- Si les données sont contenues dans un ensemble E on décompose E en classes d'équivalences. À l'arrivée d'une nouvelle donnée d, le programme doit trouver à quelle classe de E appartient d.
- On construit donc une *partition* de E, c'est à dire une famille non vide d'ensembles non vides telle que :
  - la réunion de tous les éléments de la partition constitue E
  - l'intersection de deux éléments de la partition est vide.
- Rappel : le nombre de parties d'un ensemble de cardinal n est 2^n

Partition

- Notion associée aux méthodes de classement.
- Si les données sont contenues dans un ensemble E on décompose E en classes d'équivalences. À l'arrivée d'une nouvelle donnée d, le programme doit trouver à quelle classe de E appartient d.
- On construit donc une *partition* de E, c'est à dire une famille non vide d'ensembles non vides telle que :
  - la réunion de tous les éléments de la partition constitue E
  - l'intersection de deux éléments de la partition est vide.
- Rappel : le nombre de parties d'un ensemble de cardinal n est 2^n
- Remarque : dans ce cours nous appelons *classes* les éléments de la partition. C'est justifié par le fait que cette notion mathématique est utilisée en IA pour effectuer des classements.

Algorithme de détermination de l'ensemble des parties

- Principe 1 : pour un élément x de E, si on sait construire toutes les parties de E qui ne contiennent pas x, alors on sait construire $\mathcal{P}(E)$. C'est la réunion de :

Il s'agit d'un algorithme de *programmation dynamique*.

Algorithme de détermination de l'ensemble des parties

- Principe 1 : pour un élément x de E, si on sait construire toutes les parties de E qui ne contiennent pas x, alors on sait construire $\mathcal{P}(E)$. C'est la réunion de :
  - l'ensemble $\mathcal{E}_1$ des parties de E qui ne contiennent pas x

Il s'agit d'un algorithme de *programmation dynamique*.

Algorithme de détermination de l'ensemble des parties

- Principe 1 : pour un élément x de E, si on sait construire toutes les parties de E qui ne contiennent pas x, alors on sait construire $\mathcal{P}(E)$. C'est la réunion de :
  - l'ensemble $\mathcal{E}_1$ des parties de E qui ne contiennent pas x
  - avec l'ensemble $\mathcal{E}_2$ obtenu en ajoutant x à chaque élément de $\mathcal{E}_1$.

Il s'agit d'un algorithme de *programmation dynamique*.

Algorithme de détermination de l'ensemble des parties

- Principe 1 : pour un élément x de E, si on sait construire toutes les parties de E qui ne contiennent pas x, alors on sait construire $\mathcal{P}(E)$. C'est la réunion de :
  - l'ensemble $\mathcal{E}_1$ des parties de E qui ne contiennent pas x
  - avec l'ensemble $\mathcal{E}_2$ obtenu en ajoutant x à chaque élément de $\mathcal{E}_1$.

  Il s'agit d'un algorithme de *programmation dynamique*.
- Principe 2 : Si $|E| = n$, il y a une bijection entre $\mathcal{P}(E)$ et $[\![0, 2^n - 1]\!]$. On considère que $E = \{e_0, \ldots, e_{n-1}\}$ (on se donne une numérotation des éléments de E)

Algorithme de détermination de l'ensemble des parties

- Principe 1 : pour un élément x de E, si on sait construire toutes les parties de E qui ne contiennent pas x, alors on sait construire $\mathcal{P}(E)$. C'est la réunion de :
  - l'ensemble $\mathcal{E}_1$ des parties de E qui ne contiennent pas x
  - avec l'ensemble $\mathcal{E}_2$ obtenu en ajoutant x à chaque élément de $\mathcal{E}_1$.

  Il s'agit d'un algorithme de *programmation dynamique*.
- Principe 2 : Si $|E| = n$, il y a une bijection entre $\mathcal{P}(E)$ et $[\![0, 2^n - 1]\!]$. On considère que $E = \{e_0, \ldots, e_{n-1}\}$ (on se donne une numérotation des éléments de E)
  - Pour chaque entier $k \in [\![0, 2^n - 1]\!]$, son code binaire tient sur n bits. Ce code est interprété comme suit : s'il y a un 0 en place i du code, alors l'élément e_i de E n'est pas dans le sous-ensemble associé à k, mais il y est si on trouve un 1 en position i.

Algorithme de détermination de l'ensemble des parties

- Principe 1 : pour un élément x de E, si on sait construire toutes les parties de E qui ne contiennent pas x, alors on sait construire $\mathcal{P}(E)$. C'est la réunion de :
  - l'ensemble $\mathcal{E}_1$ des parties de E qui ne contiennent pas x
  - avec l'ensemble $\mathcal{E}_2$ obtenu en ajoutant x à chaque élément de $\mathcal{E}_1$.

  Il s'agit d'un algorithme de *programmation dynamique*.
- Principe 2 : Si $|E| = n$, il y a une bijection entre $\mathcal{P}(E)$ et $[\![0, 2^n - 1]\!]$. On considère que $E = \{e_0, \ldots, e_{n-1}\}$ (on se donne une numérotation des éléments de E)
  - Pour chaque entier $k \in [\![0, 2^n - 1]\!]$, son code binaire tient sur n bits. Ce code est interprété comme suit : s'il y a un 0 en place i du code, alors l'élément e_i de E n'est pas dans le sous-ensemble associé à k, mais il y est si on trouve un 1 en position i.
  - Exemple avec $n = 4$ et $k = 11$. Alors $|\mathcal{P}(E)| = 16$, le code de k est 1011 et k est associé à l'ensemble $\{e_0, e_2, e_3\}$.

Appartenance d'un élément à une classe

- Soit E un ensemble à n éléments et $E_1, \ldots, E_k$ les k classes d'une partition de E. Et soit $x \in E$. On veut trouver i tel que $x \in E_i$.

Appartenance d'un élément à une classe

- Soit E un ensemble à n éléments et $E_1, \ldots, E_k$ les k classes d'une partition de E. Et soit $x \in E$. On veut trouver i tel que $x \in E_i$.
- Si on parcourt tous les éléments de tous les E_j pour y trouver x, la complexité est en $O(n)$.

Appartenance d'un élément à une classe

- Soit E un ensemble à n éléments et $E_1, \ldots, E_k$ les k classes d'une partition de E. Et soit $x \in E$. On veut trouver i tel que $x \in E_i$.
- Si on parcourt tous les éléments de tous les E_j pour y trouver x, la complexité est en $O(n)$.
- On peut décrire E comme une dictionnaire dans lequel les clés sont les éléments de E et les valeurs sont les numéros des parties. Par exemple

  $\{\text{x1} : 4\ ,\ \ \text{x2}\ \ : 12\ , \ldots\}$

  signifie que $x_1 \in E_4$ et $x_2 \in E_{12}$. On obtient alors la partie à laquelle appartient un élément en temps constant ($O(1)$).

Principe

k Nearest Neighbours (KNN)

- On considère

Principe

k Nearest Neighbours (KNN)

- On considère
  - un ensemble E de n points à p coordonnées représentant des données étiquetées.

Principe

k Nearest Neighbours (KNN)

- On considère
  - un ensemble E de n points à p coordonnées représentant des données étiquetées.
  - un entier $k < n$ et $x \notin E$

Principe

k Nearest Neighbours (KNN)

- On considère
  - un ensemble E de n points à p coordonnées représentant des données étiquetées.
  - un entier $k < n$ et $x \notin E$
- On cherche les k points de E les plus $\ll$ proches $\gg$ de x. Le plus souvent, on attribue ensuite à x la caractéristique la plus fréquemment partagée parmi ces k points.

Principe

k Nearest Neighbours (KNN)

- On considère
  - un ensemble E de n points à p coordonnées représentant des données étiquetées.
  - un entier $k < n$ et $x \notin E$
- On cherche les k points de E les plus « proches » de x. Le plus souvent, on attribue ensuite à x la caractéristique la plus fréquemment partagée parmi ces k points.
- La notion de *proximité* amène à définir une *distance* entre deux points. Cette distance est calculée à partir des coordonnées des points. Exemple : distance euclidienne (dans ce cas, on utilise plutôt son carré)

Analyse des besoins

- Les points sont donnés comme des tuples de r coordonnées.

Analyse des besoins

- Les points sont donnés comme des tuples de r coordonnées.
- Hypothèse : on dispose d'une liste `E` de points, et d'un dictionnaire `partition` tel que `partition[x]` indique la classe de x dans une certaine partition de E.

Analyse des besoins

- Les points sont donnés comme des tuples de r coordonnées.
- Hypothèse : on dispose d'une liste `E` de points, et d'un dictionnaire `partition` tel que `partition[x]` indique la classe de x dans une certaine partition de E.
- Il faut :

Analyse des besoins

- Les points sont donnés comme des tuples de r coordonnées.
- Hypothèse : on dispose d'une liste `E` de points, et d'un dictionnaire `partition` tel que `partition[x]` indique la classe de x dans une certaine partition de E.
- Il faut :
  - Une fonction `d(x,y)` qui calcule le carré de la distance (par exemple euclidienne) entre deux points x, y.

Analyse des besoins

- Les points sont donnés comme des tuples de r coordonnées.
- Hypothèse : on dispose d'une liste `E` de points, et d'un dictionnaire `partition` tel que `partition[x]` indique la classe de x dans une certaine partition de E.
- Il faut :
  - Une fonction `d(x,y)` qui calcule le carré de la distance (par exemple euclidienne) entre deux points x, y.
  - Une fonction `knn(E,p,d,k)` qui retourne la liste des k points de E qui sont les plus proches du point p au sens de la distance d.

Analyse des besoins

- Les points sont donnés comme des tuples de r coordonnées.
- Hypothèse : on dispose d'une liste `E` de points, et d'un dictionnaire `partition` tel que `partition[x]` indique la classe de x dans une certaine partition de E.
- Il faut :
  - Une fonction `d(x,y)` qui calcule le carré de la distance (par exemple euclidienne) entre deux points x, y.
  - Une fonction `knn(E,p,d,k)` qui retourne la liste des k points de E qui sont les plus proches du point p au sens de la distance d.
  - Une fonction `classe_maj(pts,partition))` qui retourne la classe majoritaire parmi les éléments de l'ensemble `pts` considérant le dictionnaire des classes `partition`.
    Algorithme : Etablir la liste des nombres d'occurrences des différentes classes des points de `pts`. Choisir une classe parmi celles d'occurrences maximum.

Classification de l'algorithme KNN

Apprentissage supervisé paresseux

- On classe un nouveau point en observant les catégories de ses plus proches voisins. Les voisins étant déjà catalogués, cet algorithme fait partie des *algorithmes d'apprentissage supervisé*.

Classification de l'algorithme KNN

Apprentissage supervisé paresseux

- On classe un nouveau point en observant les catégories de ses plus proches voisins. Les voisins étant déjà catalogués, cet algorithme fait partie des *algorithmes d'apprentissage supervisé*.
- Cependant, un véritable algorithme d'apprentissage se fait en deux parties : phase d'apprentissage et phase de test.

Classification de l'algorithme KNN

Apprentissage supervisé paresseux

- On classe un nouveau point en observant les catégories de ses plus proches voisins. Les voisins étant déjà catalogués, cet algorithme fait partie des *algorithmes d'apprentissage supervisé*.
- Cependant, un véritable algorithme d'apprentissage se fait en deux parties : phase d'apprentissage et phase de test.
- En l'occurrence, l'algorithme KNN n'apprend rien : les données de référence sont déjà cataloguées et la méthode de classement (prendre la classe majoritaire parmi les k plus proches voisins) lui est donnée sans qu'il ait besoin de la découvrir lui même.
  Pour cette raison, on parle d'*apprentissage paresseux.*

Définition

- La *matrice de confusion* est un tableau qui mesure la qualité d'un système de classification.

Définition

- La *matrice de confusion* est un tableau qui mesure la qualité d'un système de classification.
- Par convention, chaque ligne correspond à une classe *réelle*, chaque colonne correspond à une classe *estimée*. La cellule ligne L, colonne C contient le nombre d'éléments de la classe réelle de numéro L qui ont été estimés comme appartenant à la classe numéro C.

Définition

- La *matrice de confusion* est un tableau qui mesure la qualité d'un système de classification.
- Par convention, chaque ligne correspond à une classe *réelle*, chaque colonne correspond à une classe *estimée*. La cellule ligne L, colonne C contient le nombre d'éléments de la classe réelle de numéro L qui ont été estimés comme appartenant à la classe numéro C.
- Dans l'idéal, la matrice de confusion est diagonale. Cela signifie que le système de classement automatique donne exactement le bon résultat pour chaque point.

Exemple (Wikipedia)

- Objectif : On veut mesurer la qualité d'un système automatique de classement de mails (2 classes : légitime ou spam).

Exemple (Wikipedia)

- Objectif : On veut mesurer la qualité d'un système automatique de classement de mails (2 classes : légitime ou spam).
- Données : 200 mails dont 100 sont légitimes.

Exemple (Wikipedia)

- Objectif : On veut mesurer la qualité d'un système automatique de classement de mails (2 classes : légitime ou spam).
- Données : 200 mails dont 100 sont légitimes.
- On veut savoir :

Exemple (Wikipedia)

- Objectif : On veut mesurer la qualité d'un système automatique de classement de mails (2 classes : légitime ou spam).
- Données : 200 mails dont 100 sont légitimes.
- On veut savoir :
  - la quantité de mails légitimes identifiés comme spam.

Exemple (Wikipedia)

- Objectif : On veut mesurer la qualité d'un système automatique de classement de mails (2 classes : légitime ou spam).
- Données : 200 mails dont 100 sont légitimes.
- On veut savoir :
  - la quantité de mails légitimes identifiés comme spam.
  - la quantité de spams identifiés comme légitimes.

Exemple (Wikipedia)

- Objectif : On veut mesurer la qualité d'un système automatique de classement de mails (2 classes : légitime ou spam).
- Données : 200 mails dont 100 sont légitimes.
- On veut savoir :
  - la quantité de mails légitimes identifiés comme spam.
  - la quantité de spams identifiés comme légitimes.
- On obtient par exemple le tableau suivant :

		classe estimée	
		légitime	spam
classe réelle	légitime	95 (vrais légitimes)	5 (faux spam)
	spam	3 (faux légitimes)	97 (vrais spams)

Exemple (Wikipedia)

- Objectif : On veut mesurer la qualité d'un système automatique de classement de mails (2 classes : légitime ou spam).
- Données : 200 mails dont 100 sont légitimes.
- On veut savoir :
  - la quantité de mails légitimes identifiés comme spam.
  - la quantité de spams identifiés comme légitimes.
- On obtient par exemple le tableau suivant :

		classe estimée	
		légitime	spam
classe réelle	légitime	95 (vrais légitimes)	5 (faux spam)
	spam	3 (faux légitimes)	97 (vrais spams)

  - Classe estimée : identifiée comme telle par le classificateur de mail.

Exemple (Wikipedia)

- Objectif : On veut mesurer la qualité d'un système automatique de classement de mails (2 classes : légitime ou spam).
- Données : 200 mails dont 100 sont légitimes.
- On veut savoir :
  - la quantité de mails légitimes identifiés comme spam.
  - la quantité de spams identifiés comme légitimes.
- On obtient par exemple le tableau suivant :

		classe estimée	
		légitime	spam
classe réelle	légitime	95 (vrais légitimes)	5 (faux spam)
	spam	3 (faux légitimes)	97 (vrais spams)

  - Classe estimée : identifiée comme telle par le classificateur de mail.
  - Classe réelle : identifiée comme telle par l'algorithme.

Exemple (suite)

Analyse

- Ligne 1 : sur 100 mails légitimes (95+5), 5 sont identifiés à tort comme spams

Exemple (suite)

Analyse

- Ligne 1 : sur 100 mails légitimes (95+5), 5 sont identifiés à tort comme spams
- Ligne 2 : sur 100 spams (3+97), 3 sont identifiés à tort comme légitimes

Exemple (suite)

Analyse

- Ligne 1 : sur 100 mails légitimes (95+5), 5 sont identifiés à tort comme spams
- Ligne 2 : sur 100 spams (3+97), 3 sont identifiés à tort comme légitimes
- Colonne 1 : sur 98 mails (95+3) identifiés comme légitimes par le classificateur, 3 sont en fait des spams.

Exemple (suite)

Analyse

- Ligne 1 : sur 100 mails légitimes (95+5), 5 sont identifiés à tort comme spams
- Ligne 2 : sur 100 spams (3+97), 3 sont identifiés à tort comme légitimes
- Colonne 1 : sur 98 mails (95+3) identifiés comme légitimes par le classificateur, 3 sont en fait des spams.
- Colonne 2 : sur 102 mails (97+5) identifiés comme spams par le classificateur, 5 sont en fait légitimes.

Exemple (suite)

Analyse

- Ligne 1 : sur 100 mails légitimes (95+5), 5 sont identifiés à tort comme spams
- Ligne 2 : sur 100 spams (3+97), 3 sont identifiés à tort comme légitimes
- Colonne 1 : sur 98 mails (95+3) identifiés comme légitimes par le classificateur, 3 sont en fait des spams.
- Colonne 2 : sur 102 mails (97+5) identifiés comme spams par le classificateur, 5 sont en fait légitimes.
- Diagonalement : sur les 200 mails reçus (95+5+3+97), 192 (95+97) sont estimés correctement par le classificateur.

Vocabulaire

La matrice de confusion permet d'estimer le type d'erreurs commises et le taux de ces erreurs.

- Le taux d'erreur : nombre de prédictions incorrectes sur nombre de prédictions. Exemple du classificateur : $\frac{5+3}{95+5+3+97} = \frac{8}{200}$. On vise une valeur proche de 0.

Vocabulaire

La matrice de confusion permet d'estimer le type d'erreurs commises et le taux de ces erreurs.

- Le taux d'erreur : nombre de prédictions incorrectes sur nombre de prédictions. Exemple du classificateur : $\frac{5+3}{95+5+3+97} = \frac{8}{200}$. On vise une valeur proche de 0.
- La précision : nombre de prédictions correctes sur le nombre total de prédictions. Exemple du classificateur : $\frac{95+97}{95+5+3+97} = \frac{192}{200}$. On vise une valeur proche de 1.

Vocabulaire

La matrice de confusion permet d'estimer le type d'erreurs commises et le taux de ces erreurs.

- Le taux d'erreur : nombre de prédictions incorrectes sur nombre de prédictions. Exemple du classificateur : $\frac{5+3}{95+5+3+97} = \frac{8}{200}$. On vise une valeur proche de 0.
- La précision : nombre de prédictions correctes sur le nombre total de prédictions. Exemple du classificateur : $\frac{95+97}{95+5+3+97} = \frac{192}{200}$. On vise une valeur proche de 1.
- D'autres taux sont également porteurs d'informations. Citons : taux de vrais positifs ($\frac{95}{100}$), taux de vrais négatifs ($\frac{97}{100}$), taux de faux positifs ($\frac{3}{100}$), taux de vrais positifs ($\frac{5}{100}$).

Calcul de la matrice de confusion

- On dispose d'un ensemble E de données représentés par des vecteurs (des tuples d'un nombre fixé de coordonnées, toujours le même nombre) et d'une classification C (on sait à quelle classe appartient tel vecteur).

Calcul de la matrice de confusion

- On dispose d'un ensemble E de données représentés par des vecteurs (des tuples d'un nombre fixé de coordonnées, toujours le même nombre) et d'une classification C (on sait à quelle classe appartient tel vecteur).
- On décompose l'ensemble des données en 2 parties : un ensemble d'apprentissage A (par exemple 75% des données), et un ensemble de tests T (par exemple 25% des données).

Calcul de la matrice de confusion

- On déclare

```
1 def matrice(A,T,D,k,d):#retourne mat. de confusion
```

`A` (données d'apprentissage) et `T` (données de tests) sont des listes de tuples, `D` est un dictionnaire

`point:numéro de classe`

, `k` est le nombre de plus proches voisins utilisés pour l'algorithme KNN et `d` une fonction distance au carrée.

Calcul de la matrice de confusion

- On déclare

```
1 def matrice(A,T,D,k,d):#retourne mat. de confusion
```

`A` (données d'apprentissage) et `T` (données de tests) sont des listes de tuples, `D` est un dictionnaire

`point:numéro de classe`

, `k` est le nombre de plus proches voisins utilisés pour l'algorithme KNN et `d` une fonction distance au carrée.

- Algorithme :

Calcul de la matrice de confusion

- On déclare

```
1 def matrice(A,T,D,k,d):#retourne mat. de confusion
```

`A` (données d'apprentissage) et `T` (données de tests) sont des listes de tuples, `D` est un dictionnaire

```
point:numéro de classe
```

, `k` est le nombre de plus proches voisins utilisés pour l'algorithme KNN et `d` une fonction distance au carrée.

- Algorithme :
  - Créer une matrice carrée nulle M avec autant de lignes que de classes.

Calcul de la matrice de confusion

- On déclare

```
1 def matrice(A,T,D,k,d):#retourne mat. de confusion
```

`A` (données d'apprentissage) et `T` (données de tests) sont des listes de tuples, `D` est un dictionnaire

```
point:numéro de classe
```

, `k` est le nombre de plus proches voisins utilisés pour l'algorithme KNN et `d` une fonction distance au carrée.

- Algorithme :
  - Créer une matrice carrée nulle M avec autant de lignes que de classes.
  - pour chaque point de l'ensemble de tests :

Calcul de la matrice de confusion

- On déclare

```
1 def matrice(A,T,D,k,d):#retourne mat. de confusion
```

`A` (données d'apprentissage) et `T` (données de tests) sont des listes de tuples, `D` est un dictionnaire

`point:numéro de classe`

, `k` est le nombre de plus proches voisins utilisés pour l'algorithme KNN et `d` une fonction distance au carrée.

- Algorithme :
  - Créer une matrice carrée nulle M avec autant de lignes que de classes.
  - pour chaque point de l'ensemble de tests :
    - déterminer sa classe estimée (c.a.d un numéro de colonne j) en utilisant A comme ensemble de points dans `knn` (si l'algo étudié est KNN).

Calcul de la matrice de confusion

- On déclare

```
1 def matrice(A,T,D,k,d):#retourne mat. de confusion
```

`A` (données d'apprentissage) et `T` (données de tests) sont des listes de tuples, `D` est un dictionnaire

`point:numéro de classe`

, `k` est le nombre de plus proches voisins utilisés pour l'algorithme KNN et `d` une fonction distance au carrée.

- Algorithme :
  - Créer une matrice carrée nulle M avec autant de lignes que de classes.
  - pour chaque point de l'ensemble de tests :
    - déterminer sa classe estimée (c.a.d un numéro de colonne j) en utilisant A comme ensemble de points dans `knn` (si l'algo étudié est KNN).
    - Déterminer sa classe véritable (information contenue dans T). Cela donne un numéro de ligne i.

Calcul de la matrice de confusion

- On déclare

```
1 def matrice(A,T,D,k,d):#retourne mat. de confusion
```

`A` (données d'apprentissage) et `T` (données de tests) sont des listes de tuples, `D` est un dictionnaire

`point:numéro de classe`

, `k` est le nombre de plus proches voisins utilisés pour l'algorithme KNN et `d` une fonction distance au carrée.

- Algorithme :
  - Créer une matrice carrée nulle M avec autant de lignes que de classes.
  - pour chaque point de l'ensemble de tests :
    - déterminer sa classe estimée (c.a.d un numéro de colonne j) en utilisant A comme ensemble de points dans `knn` (si l'algo étudié est KNN).
    - Déterminer sa classe véritable (information contenue dans T). Cela donne un numéro de ligne i.
    - Incrémenter de 1 le coefficient $M_{i,j}$

Taux d'erreur et précision

Une fois la matrice de confusion connue :

- Calcul du taux d'erreur : faire la somme des coefficients non diagonaux ; diviser par la somme de tous les coefficients.

Taux d'erreur et précision

Une fois la matrice de confusion connue :

- Calcul du taux d'erreur : faire la somme des coefficients non diagonaux ; diviser par la somme de tous les coefficients.
- Calcul de la précision : faire la somme des coefficients diagonaux ; diviser par la somme de tous les coefficients.

Présentation et historique

- Il s'agit encore d'un algorithme de *classification* (ex : la photo représente un adorable chaton, un mignon petit chien, Vladimir Poutine...) comme pour KNN.

Présentation et historique

- Il s'agit encore d'un algorithme de *classification* (ex : la photo représente un adorable chaton, un mignon petit chien, Vladimir Poutine...) comme pour KNN.
- Le terme « k-means » (K-moyennes) a été utilisé pour la première fois par James MacQueen en 1967. L'idée originale a été proposée par Hugo Steinhaus en 1957.

Décomposer en *clusters*

- On dispose d'un ensemble E de n vecteurs de mêmes dimensions.
  On veut décomposer E en k groupes, souvent appelés *clusters*, de façon à minimiser une certaine fonction dite *de coût*.
  L'ensemble des clusters constitue une partition de E.

Décomposer en *clusters*

- On dispose d'un ensemble E de n vecteurs de mêmes dimensions.
  On veut décomposer E en k groupes, souvent appelés *clusters*, de façon à minimiser une certaine fonction dite *de coût*.
  L'ensemble des clusters constitue une partition de E.
- Le barycentre de chaque cluster est appelé un *centroïde*. La fonction à minimiser est la somme des « distances » de chaque point au centroïde du cluster auquel il appartient. La notion de distance varie selon le problème étudié.

Décomposer en *clusters*

- On dispose d'un ensemble E de n vecteurs de mêmes dimensions.
  On veut décomposer E en k groupes, souvent appelés *clusters*, de façon à minimiser une certaine fonction dite *de coût*.
  L'ensemble des clusters constitue une partition de E.
- Le barycentre de chaque cluster est appelé un *centroïde*. La fonction à minimiser est la somme des « distances » de chaque point au centroïde du cluster auquel il appartient. La notion de distance varie selon le problème étudié.
- Différence fondamentale avec l'apprentissage supervisé :

Décomposer en *clusters*

- On dispose d'un ensemble E de n vecteurs de mêmes dimensions.
  On veut décomposer E en k groupes, souvent appelés *clusters*, de façon à minimiser une certaine fonction dite *de coût*.
  L'ensemble des clusters constitue une partition de E.
- Le barycentre de chaque cluster est appelé un *centroïde*. La fonction à minimiser est la somme des « distances » de chaque point au centroïde du cluster auquel il appartient. La notion de distance varie selon le problème étudié.
- Différence fondamentale avec l'apprentissage supervisé :
  - Dans le cadre de l'apprentissage supervisé, la machine connaît déjà la liste des réponses possibles qu'on attend d'elle. Elle travaille à partir de données étiquetées (ex : KNN).

Décomposer en *clusters*

- On dispose d'un ensemble E de n vecteurs de mêmes dimensions. On veut décomposer E en k groupes, souvent appelés *clusters*, de façon à minimiser une certaine fonction dite *de coût*. L'ensemble des clusters constitue une partition de E.
- Le barycentre de chaque cluster est appelé un *centroïde*. La fonction à minimiser est la somme des « distances » de chaque point au centroïde du cluster auquel il appartient. La notion de distance varie selon le problème étudié.
- Différence fondamentale avec l'apprentissage supervisé :
  - Dans le cadre de l'apprentissage supervisé, la machine connaît déjà la liste des réponses possibles qu'on attend d'elle. Elle travaille à partir de données étiquetées (ex : KNN).
  - Les réponses que l'on cherche à prédire dans un apprentissage non supervisées ne sont pas disponibles dans les jeux de données. L'algorithme utilise un jeu de données non étiquetées. On demande alors à la machine de créer ses propres réponses.

Algorithme

Entrées : Un ensemble E de n données (vecteurs de mêmes dimensions) ; un entier k désignant le nombre de clusters voulus.

- Initialisation possible : choisir (souvent aléatoirement) les centres de chaque cluster : prendre k points parmi les n du nuage.

Algorithme

Entrées : Un ensemble E de n données (vecteurs de mêmes dimensions) ; un entier k désignant le nombre de clusters voulus.

- Initialisation possible : choisir (souvent aléatoirement) les centres de chaque cluster : prendre k points parmi les n du nuage.
- Répéter les deux étapes :

Algorithme

Entrées : Un ensemble E de n données (vecteurs de mêmes dimensions) ; un entier k désignant le nombre de clusters voulus.

- Initialisation possible : choisir (souvent aléatoirement) les centres de chaque cluster : prendre k points parmi les n du nuage.
- Répéter les deux étapes :
  - Affecter chaque donnée au cluster dont le centroïde est le plus proche.

Algorithme

Entrées : Un ensemble E de n données (vecteurs de mêmes dimensions) ; un entier k désignant le nombre de clusters voulus.

- Initialisation possible : choisir (souvent aléatoirement) les centres de chaque cluster : prendre k points parmi les n du nuage.
- Répéter les deux étapes :
  - Affecter chaque donnée au cluster dont le centroïde est le plus proche.
  - Recalculer le centroïde de chaque cluster (isobarycentre des points du cluster)

Algorithme

Entrées : Un ensemble E de n données (vecteurs de mêmes dimensions) ; un entier k désignant le nombre de clusters voulus.

- Initialisation possible : choisir (souvent aléatoirement) les centres de chaque cluster : prendre k points parmi les n du nuage.
- Répéter les deux étapes :
  - Affecter chaque donnée au cluster dont le centroïde est le plus proche.
  - Recalculer le centroïde de chaque cluster (isobarycentre des points du cluster)
- Sortie de boucle, deux possibilités :

Algorithme

Entrées : Un ensemble E de n données (vecteurs de mêmes dimensions) ; un entier k désignant le nombre de clusters voulus.

- Initialisation possible : choisir (souvent aléatoirement) les centres de chaque cluster : prendre k points parmi les n du nuage.
- Répéter les deux étapes :
  - Affecter chaque donnée au cluster dont le centroïde est le plus proche.
  - Recalculer le centroïde de chaque cluster (isobarycentre des points du cluster)
- Sortie de boucle, deux possibilités :
  - Soit on s'arrête après un certain nombre d'itérations fixé à l'avance.

Algorithme

Entrées : Un ensemble E de n données (vecteurs de mêmes dimensions) ; un entier k désignant le nombre de clusters voulus.

- Initialisation possible : choisir (souvent aléatoirement) les centres de chaque cluster : prendre k points parmi les n du nuage.
- Répéter les deux étapes :
  - Affecter chaque donnée au cluster dont le centroïde est le plus proche.
  - Recalculer le centroïde de chaque cluster (isobarycentre des points du cluster)
- Sortie de boucle, deux possibilités :
  - Soit on s'arrête après un certain nombre d'itérations fixé à l'avance.
  - Soit on s'arrête lorsque les centroïdes calculés lors d'un tour sont égaux aux centroïdes calculés au tour précédent.

Retour sur l'initialisation

L'initialisation est un facteur déterminant dans la qualité des résultats (minimum local).

- Il existe deux méthodes très utilisées :

Retour sur l'initialisation

L'initialisation est un facteur déterminant dans la qualité des résultats (minimum local).

- Il existe deux méthodes très utilisées :
  - La *méthode de Forgy* : choisir aléatoirement k points initiaux dans le nuage. Ces k points sont les k premiers centroïdes et servent à initialiser les classes
    Le calcul des centroïdes est alors fait avant celui des clusters

Retour sur l'initialisation

L'initialisation est un facteur déterminant dans la qualité des résultats (minimum local).

- Il existe deux méthodes très utilisées :
  - La *méthode de Forgy* : choisir aléatoirement k points initiaux dans le nuage. Ces k points sont les k premiers centroïdes et servent à initialiser les classes
    Le calcul des centroïdes est alors fait avant celui des clusters
  - La méthode du *partitionnement aléatoire* affecte aléatoirement chaque point à un cluster (par exemple le point 1 au cluster 12, le point 5 au cluster 1...). Elle calcule ensuite le centroïde de chaque cluster.
    La détermination des clusters se fait alors avant le calcul des centroïdes.

Retour sur l'initialisation

L'initialisation est un facteur déterminant dans la qualité des résultats (minimum local).

- Il existe deux méthodes très utilisées :
  - La *méthode de Forgy* : choisir aléatoirement k points initiaux dans le nuage. Ces k points sont les k premiers centroïdes et servent à initialiser les classes
    Le calcul des centroïdes est alors fait avant celui des clusters
  - La méthode du *partitionnement aléatoire* affecte aléatoirement chaque point à un cluster (par exemple le point 1 au cluster 12, le point 5 au cluster 1...). Elle calcule ensuite le centroïde de chaque cluster.
    La détermination des clusters se fait alors avant le calcul des centroïdes.
- Importance du choix de k :

Retour sur l'initialisation

L'initialisation est un facteur déterminant dans la qualité des résultats (minimum local).

- Il existe deux méthodes très utilisées :
  - La *méthode de Forgy* : choisir aléatoirement k points initiaux dans le nuage. Ces k points sont les k premiers centroïdes et servent à initialiser les classes
    Le calcul des centroïdes est alors fait avant celui des clusters
  - La méthode du *partitionnement aléatoire* affecte aléatoirement chaque point à un cluster (par exemple le point 1 au cluster 12, le point 5 au cluster 1...). Elle calcule ensuite le centroïde de chaque cluster.
    La détermination des clusters se fait alors avant le calcul des centroïdes.
- Importance du choix de k :
  - si k est trop petit, les données dans chaque groupe risquent de ne pas avoir de similarité forte,

Retour sur l'initialisation

L'initialisation est un facteur déterminant dans la qualité des résultats (minimum local).

- Il existe deux méthodes très utilisées :
  - La *méthode de Forgy* : choisir aléatoirement k points initiaux dans le nuage. Ces k points sont les k premiers centroïdes et servent à initialiser les classes
    Le calcul des centroïdes est alors fait avant celui des clusters
  - La méthode du *partitionnement aléatoire* affecte aléatoirement chaque point à un cluster (par exemple le point 1 au cluster 12, le point 5 au cluster 1...). Elle calcule ensuite le centroïde de chaque cluster.
    La détermination des clusters se fait alors avant le calcul des centroïdes.
- Importance du choix de k :
  - si k est trop petit, les données dans chaque groupe risquent de ne pas avoir de similarité forte,
  - si k est trop grand, les groupes ne permettent pas de découvrir de caractéristiques intéressantes.

Retour sur l'initialisation

L'initialisation est un facteur déterminant dans la qualité des résultats (minimum local).

- Il existe deux méthodes très utilisées :
  - La *méthode de Forgy* : choisir aléatoirement k points initiaux dans le nuage. Ces k points sont les k premiers centroïdes et servent à initialiser les classes
    Le calcul des centroïdes est alors fait avant celui des clusters
  - La méthode du *partitionnement aléatoire* affecte aléatoirement chaque point à un cluster (par exemple le point 1 au cluster 12, le point 5 au cluster 1...). Elle calcule ensuite le centroïde de chaque cluster.
    La détermination des clusters se fait alors avant le calcul des centroïdes.
- Importance du choix de k :
  - si k est trop petit, les données dans chaque groupe risquent de ne pas avoir de similarité forte,
  - si k est trop grand, les groupes ne permettent pas de découvrir de caractéristiques intéressantes.
  - Il est sage d'essayer plusieurs valeurs de k lorsque $|E|$ est grand.

Convergence

Recherche de k pour un ensemble E de n données.

- Dans chaque cluster, les données présentes minimisent la somme des carrés des distances entre chaque donnée et le centroïde (variance). On cherche donc la valeur de k qui permet de trouver des groupes qui minimisent les variances.

Convergence

Recherche de k pour un ensemble E de n données.

- Dans chaque cluster, les données présentes minimisent la somme des carrés des distances entre chaque donnée et le centroïde (variance). On cherche donc la valeur de k qui permet de trouver des groupes qui minimisent les variances.
- Dans le groupe/cluster G_i, notant C_i le centroïde, la somme $\sum_{i=1}^{k} \sum_{P \in G_i} d(P, C_i)^2$ est minimisée. Cette somme fait intervenir exactement les n distances (Point/centroïde du point).

Convergence

Recherche de k pour un ensemble E de n données.

- Dans chaque cluster, les données présentes minimisent la somme des carrés des distances entre chaque donnée et le centroïde (variance). On cherche donc la valeur de k qui permet de trouver des groupes qui minimisent les variances.
- Dans le groupe/cluster G_i, notant C_i le centroïde, la somme $\sum_{i=1}^{k} \sum_{P \in G_i} d(P, C_i)^2$ est minimisée. Cette somme fait intervenir exactement les n distances (Point/centroïde du point).
- Pour le point numéro j du nuage la quantité $\min_{i=0,\ldots,k-1} d(P_j, C_i)^2$ est la distance de P_j au centre du cluster qui le contient.

Convergence

Recherche de k pour un ensemble E de n données.

- Dans chaque cluster, les données présentes minimisent la somme des carrés des distances entre chaque donnée et le centroïde (variance). On cherche donc la valeur de k qui permet de trouver des groupes qui minimisent les variances.
- Dans le groupe/cluster G_i, notant C_i le centroïde, la somme $\sum_{i=1}^{k} \sum_{P \in G_i} d(P, C_i)^2$ est minimisée. Cette somme fait intervenir exactement les n distances (Point/centroïde du point).
- Pour le point numéro j du nuage la quantité $\min_{i=0,\dots,k-1} d(P_j, C_i)^2$ est la distance de P_j au centre du cluster qui le contient.
- On veut donc minimiser la somme $\sum_{j=0}^{n-1} \min_{i=0,\dots,k-1} d(P_j, C_i)^2$.

Convergence

Recherche de k.

- On veut minimiser la somme

$$\sum_{j=0}^{n-1} \min_{i=0,\ldots,k-1} d(P_j, C_i)^2$$

Convergence

Recherche de k.

- On veut minimiser la somme

$$\sum_{j=0}^{n-1} \min_{i=0,\ldots,k-1} d(P_j, C_i)^2$$

- Comme on s'arrête lorsque les centres ne sont plus modifiés, il y a convergence du procédé. La preuve de cette propriété (admise) exclu notamment tout phénomène d'oscillation. Le résultat trouvé n'étant pas toujours optimal (cela dépend des centroïdes initiaux), on parle de convergence vers des *minimas locaux*.

Convergence

Recherche de k.

- On veut minimiser la somme

$$\sum_{j=0}^{n-1} \min_{i=0,\ldots,k-1} d(P_j, C_i)^2$$

- Comme on s'arrête lorsque les centres ne sont plus modifiés, il y a convergence du procédé. La preuve de cette propriété (admise) exclu notamment tout phénomène d'oscillation. Le résultat trouvé n'étant pas toujours optimal (cela dépend des centroïdes initiaux), on parle de convergence vers des *minimas locaux*.
- Pour un k donné, exécuter plusieurs fois l'algorithme en changeant à chaque fois aléatoirement les centroïdes initiaux.

Analyse des besoins (1)

`E` est une liste de vecteurs de mêmes tailles représentant les données.

- Une fonction `genere_centre(E,k)` retourne une liste de k numéros de points de E choisis aléatoirement. Par exemple si $k = 3$ et $n = 50$, elle peut renvoyer `[2,36,24]`. C'est la méthode de Forgy, on a mis au hasard un point dans chacun des k clusters.

Analyse des besoins (1)

`E` est une liste de vecteurs de mêmes tailles représentant les données.

- Une fonction `genere_centre(E,k)` retourne une liste de k numéros de points de E choisis aléatoirement. Par exemple si $k = 3$ et $n = 50$, elle peut renvoyer `[2,36,24]`. C'est la méthode de Forgy, on a mis au hasard un point dans chacun des k clusters.
- Une fonction `barycentre(cluster)` calcule l'isobarycentres des éléments contenus dans un cluster (liste de points).

Analyse des besoins (1)

`E` est une liste de vecteurs de mêmes tailles représentant les données.

- Une fonction `genere_centre(E,k)` retourne une liste de k numéros de points de E choisis aléatoirement. Par exemple si $k = 3$ et $n = 50$, elle peut renvoyer `[2,36,24]`. C'est la méthode de Forgy, on a mis au hasard un point dans chacun des k clusters.
- Une fonction `barycentre(cluster)` calcule l'isobarycentres des éléments contenus dans un cluster (liste de points).
- Une fonction `ind_minimum(dist)` prend en paramètre la liste des distances entre un point et les différents centroïdes. Elle retourne le numéro du cluster dont le centroïde est le plus proche du point. En cas d'égalité de plus petite distance, choisir un cluster aléatoirement. **En pratique**, pour un point donné on calcule ses distances aux différents centroïdes et on appelle `ind_minimum` pour déterminer le centroïde le plus proche.

Analyse des besoins (2)

- Une fonction `partitionne(E,C,d)` crée (en particulier) la partition initiale (méthode du partitionnement aléatoire).
  Paramètres : une liste de points `E`, une liste `C` de k numéros de points (les centroïdes) et une fonction distance `d`. `C` représente la liste des centroïdes.

Analyse des besoins (2)

- Une fonction `partitionne(E,C,d)` crée (en particulier) la partition initiale (méthode du partitionnement aléatoire).
  Paramètres : une liste de points `E`, une liste `C` de k numéros de points (les centroïdes) et une fonction distance `d`. `C` représente la liste des centroïdes.
- Retourne un triplet `clusters,new_centers,classes` :

Analyse des besoins (2)

- Une fonction `partitionne(E,C,d)` crée (en particulier) la partition initiale (méthode du partitionnement aléatoire).
  Paramètres : une liste de points `E`, une liste `C` de k numéros de points (les centroïdes) et une fonction distance `d`. `C` représente la liste des centroïdes.
- Retourne un triplet `clusters,new_centers,classes` :
  - `clusters` : liste de listes d'entiers. Ainsi, **clusters[i]** est la liste des numéros des points de E contenus dans le cluster i.

Analyse des besoins (2)

- Une fonction `partitionne(E,C,d)` crée (en particulier) la partition initiale (méthode du partitionnement aléatoire).
  Paramètres : une liste de points `E`, une liste `C` de k numéros de points (les centroïdes) et une fonction distance `d`. `C` représente la liste des centroïdes.
- Retourne un triplet `clusters,new_centers,classes` :
  - `clusters` : liste de listes d'entiers. Ainsi, **clusters[i]** est la liste des numéros des points de E contenus dans le cluster i.
  - `new_centers` : liste des centroïdes de **clusters**. `new_centers[i]` : liste des coordonnées du centroïde du cluster i.

Analyse des besoins (2)

- Une fonction `partitionne(E,C,d)` crée (en particulier) la partition initiale (méthode du partitionnement aléatoire).
  Paramètres : une liste de points `E`, une liste `C` de k numéros de points (les centroïdes) et une fonction distance `d`. `C` représente la liste des centroïdes.
- Retourne un triplet `clusters,new_centers,classes` :
  - `clusters` : liste de listes d'entiers. Ainsi, **clusters[i]** est la liste des numéros des points de E contenus dans le cluster i.
  - `new_centers` : liste des centroïdes de **clusters**. `new_centers[i]` : liste des coordonnées du centroïde du cluster i.
  - `classes` : dictionnaire. Clés : numéros de points (de 0 à $n-1$) ; Valeurs : numéros de clusters (entre 0 et $k-1$).
    `classes[i]` retourne le numéro du cluster qui contient le point numéro i.

Analyse des besoins (3)

Cas du partitionnement aléatoire

La fonction `kmeans(ens,centres,d,m)` prend en paramètres, un ensemble de points `ens`, un ensemble de numéros de centroïdes `centres` (de la partition initiale), une distance `d` et un nombre `m` maximal d'itérations.

Principe :

- Affecter chaque point à son cluster (par la fonction `partitionne`).

Analyse des besoins (3)

Cas du partitionnement aléatoire

La fonction `kmeans(ens,centres,d,m)` prend en paramètres, un ensemble de points `ens`, un ensemble de numéros de centroïdes `centres` (de la partition initiale), une distance `d` et un nombre `m` maximal d'itérations.

Principe :

- Affecter chaque point à son cluster (par la fonction `partitionne`).
- Tant que

Faire :

Analyse des besoins (3)

Cas du partitionnement aléatoire

La fonction `kmeans(ens,centres,d,m)` prend en paramètres, un ensemble de points `ens`, un ensemble de numéros de centroïdes `centres` (de la partition initiale), une distance `d` et un nombre `m` maximal d'itérations.

Principe :

- Affecter chaque point à son cluster (par la fonction `partitionne`).
- Tant que
  1. le maximum d'itérations n'est pas atteint

  Faire :

Analyse des besoins (3)

Cas du partitionnement aléatoire

La fonction `kmeans(ens,centres,d,m)` prend en paramètres, un ensemble de points `ens`, un ensemble de numéros de centroïdes `centres` (de la partition initiale), une distance `d` et un nombre `m` maximal d'itérations.

Principe :

- Affecter chaque point à son cluster (par la fonction `partitionne`).
- Tant que
  1. le maximum d'itérations n'est pas atteint
  2. et qu'on peut trouver un point dont le numéro de cluster a changé par rapport au tour précédent,

  Faire :

Analyse des besoins (3)

Cas du partitionnement aléatoire

La fonction `kmeans(ens,centres,d,m)` prend en paramètres, un ensemble de points `ens`, un ensemble de numéros de centroïdes `centres` (de la partition initiale), une distance `d` et un nombre `m` maximal d'itérations.

Principe :

- Affecter chaque point à son cluster (par la fonction `partitionne`).
- Tant que
  1. le maximum d'itérations n'est pas atteint
  2. et qu'on peut trouver un point dont le numéro de cluster a changé par rapport au tour précédent,

  Faire :
  - Recalculer les coordonnées des centroïdes.

Analyse des besoins (3)

Cas du partitionnement aléatoire

La fonction `kmeans(ens,centres,d,m)` prend en paramètres, un ensemble de points `ens`, un ensemble de numéros de centroïdes `centres` (de la partition initiale), une distance `d` et un nombre `m` maximal d'itérations.

Principe :

- Affecter chaque point à son cluster (par la fonction `partitionne`).
- Tant que
  1. le maximum d'itérations n'est pas atteint
  2. et qu'on peut trouver un point dont le numéro de cluster a changé par rapport au tour précédent,

  Faire :
  - Recalculer les coordonnées des centroïdes.
  - Mettre à jour le contenu des k clusters en adaptant la correspondance (numero de point/numéro de son cluster) (c.a.d. pour chaque point déterminer le centroïde le plus proche : le cluster correspondant devient le cluster du point).

Analyse des besoins (4)

Cas de la méthode de Forgy

La fonction `kmeans(ens,clusters,d,m)` prend en paramètres, un ensemble de points `ens`, un dictionnaire indiquant pour chaque point son cluster dans la partition initiale, une fonction distance `d` et un nombre `m` maximal d'itérations.
Principe :

- Calculer les centroïdes des clusters.

Analyse des besoins (4)

Cas de la méthode de Forgy

La fonction `kmeans(ens,clusters,d,m)` prend en paramètres, un ensemble de points `ens`, un dictionnaire indiquant pour chaque point son cluster dans la partition initiale, une fonction distance `d` et un nombre `m` maximal d'itérations.
Principe :

- Calculer les centroïdes des clusters.
- Tant que

Faire :

Analyse des besoins (4)

Cas de la méthode de Forgy

La fonction `kmeans(ens,clusters,d,m)` prend en paramètres, un ensemble de points `ens`, un dictionnaire indiquant pour chaque point son cluster dans la partition initiale, une fonction distance `d` et un nombre `m` maximal d'itérations.
Principe :

- Calculer les centroïdes des clusters.
- Tant que
  1. le maximum d'itérations n'est pas atteint

  Faire :

Analyse des besoins (4)

Cas de la méthode de Forgy

La fonction `kmeans(ens,clusters,d,m)` prend en paramètres, un ensemble de points `ens`, un dictionnaire indiquant pour chaque point son cluster dans la partition initiale, une fonction distance `d` et un nombre `m` maximal d'itérations.

Principe :

- Calculer les centroïdes des clusters.
- Tant que
  1. le maximum d'itérations n'est pas atteint
  2. et qu'on peut trouver un point dont le numéro de cluster a changé par rapport au tour précédent,

  Faire :

Analyse des besoins (4)

Cas de la méthode de Forgy

La fonction `kmeans(ens,clusters,d,m)` prend en paramètres, un ensemble de points `ens`, un dictionnaire indiquant pour chaque point son cluster dans la partition initiale, une fonction distance `d` et un nombre `m` maximal d'itérations.
Principe :

- Calculer les centroïdes des clusters.
- Tant que
  1. le maximum d'itérations n'est pas atteint
  2. et qu'on peut trouver un point dont le numéro de cluster a changé par rapport au tour précédent,

  Faire :
  - Mettre à jour le contenu des k clusters en adaptant la correspondance numero de point/numéro de son cluster (c.a.d. pour chaque point déterminer le centroïde le plus proche).

Analyse des besoins (4)

Cas de la méthode de Forgy

La fonction `kmeans(ens,clusters,d,m)` prend en paramètres, un ensemble de points `ens`, un dictionnaire indiquant pour chaque point son cluster dans la partition initiale, une fonction distance `d` et un nombre `m` maximal d'itérations.
Principe :

- Calculer les centroïdes des clusters.
- Tant que
  1. le maximum d'itérations n'est pas atteint
  2. et qu'on peut trouver un point dont le numéro de cluster a changé par rapport au tour précédent,

  Faire :
  - Mettre à jour le contenu des k clusters en adaptant la correspondance numero de point/numéro de son cluster (c.a.d. pour chaque point déterminer le centroïde le plus proche).
  - Recalculer les coordonnées des centroïdes.