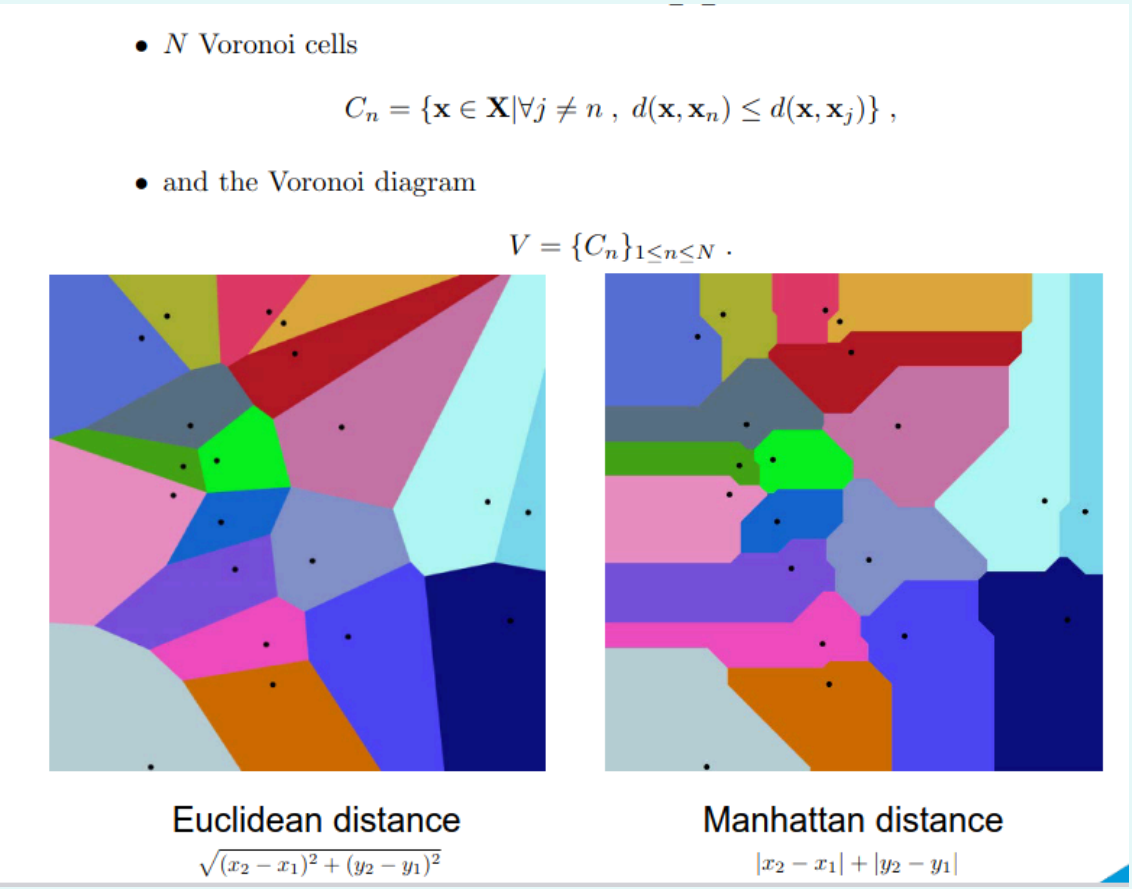
KNN (K Nearest Neighbors)

On prend le voisin le plus proche et on le choisit comme résultat.

Voronoi Cells

On créé des cellules autour de chaque point de telle sorte à ce qu'on puisse voir visuellement.



K-nearest neighbors

- on trouve les k voisins les plus proches
- on prend la majorité parmis ces voisins pour décider

Améliorer son dataset

Data reduction

On veut absolument réduire le nombre de données de notre ensemble de données quand on applique les K-NN parce qu'on doit à chaque fois comparer avec **tous** les voisins. Pour ça on choisit des représentants (des **prototypes**). Mais il y a plusieurs techniques pour faire ça :

- avec le centre de gravité (mais ça ne marche pas tjrs, par exemple si on a un cercle rouge entouré par un cercle vert, ils ont le même centre de gravité !)
- avec l'algorithme des **condensed** nearest neighbors (de meilleurs frontières **et** plus rapide à exécuter car moins de comparaisons) :
  - on a un training set 1, 2, (bleu) 3, 4, (rouge) 5 (vert).
  - On initialise `P = {1}` (random).
  - Ensuite, on choisit p. ex. le 2. Le plus près de 2 dans P est 1, qui a la même classe, donc on jette le 2.
  - On choisit le 3. Quel est le plus près du 3 dans `P` ? C'est 1, qui a une classe différente, donc on garde C.
  - On choisit le 4. Quel est le plus près du 4 dans `P` ? C'est 3, qui a une classe indentique, donc on jette 4.
  - etc. cet algo n'a pas toujours de sens.

Normalisation

On applique le KNN à un dataset comme ceci :
• Age: Ranges from 0 to 100
• Income: Ranges from $0 to $1,000,000
• Binary Gender: Encoded as 0 or 1

S'il y a une diff de $1000 entre A et B, le modèle va considérer ça plus important qu'une diff de 20 ans entre les deux ! On doit donc normaliser (garder la même distribution mais faire un rescaling).

Corriger unbalanced dataset

- on peut enlever des points pour rétablir l'équilibre
- on peut ajouter un poids plus fort aux points
- on peut ajouter des points synthétiques pour compenser

Greedy k-NN Graph construction

Idée: connecter tout le monde à quelques personnes (randomly) et regarder les amis des amis.
p. exemple :

- on connecte Alice à 3 étrangers.
- on regarde dans les amis des amis d'Alice et on compute leur similarity score.
- on prend les premiers et on créé un nouveau "voisinage" à partir de ça.

- et on répète tant que le nombre de changements est $> \varepsilon$.

Dans les espaces très grands, il faut bcp de données pour remplir tout l'espace

→ euclidean distance peu adaptée ? en fait pas vraiment, parce que les données vivent dans un low dimensional manifold (elles sont rassemblées dans un petit endroit du grand espace multi-dimensionnel).