SUPPORT VECTOR MACHINES

1. INTRODUCTION AUX SVM
2. POUR FAIRE SIMPLE...
3. QU'EST-CE QUE C'EST QUE CA ENCORE ?!
4.5. PASSONS DU COTE OBSCUR DE LA FORCE... LES MATHS !
6.7. QUE FAIRE EN CAS DE DONNEES NON LINEAIRES ?
8. LES HYPERPARAMETRES HYPERIMPORTANTS
9. MULTI CLASS CLASSIFICATION
10. PETIT RECAP'

SVMMAIRE

Un peu d'histoire...

Les **SVM** ont été introduites par Vladimir Vapnik et Alexey Chervonenkis dans les années 1960-70.

Ils ont gagné en popularité dans les années 1990 en tant qu'**algorithmes puissants** pour la **classification binaire et la régression**.

Leur efficacité dans la gestion de données à haute dimension et leur capacité à gérer des ensembles de données complexes ont contribué à leur adoption dans divers domaines.

Bien que les SVM aient été largement utilisées et appréciées pendant de nombreuses années, leur popularité relative par rapport à d'autres techniques de machine learning peut varier en fonction des avancées technologiques, des besoins spécifiques des tâches, etc.

Avancées dans les Réseaux de Neurones Profonds (DNN)
Capacité des DNN à Apprendre des Représentations Hiérarchiques
Meilleure Gestion des Ensembles de Données Massif
Sensibilité aux Paramètres des SVM
Émergence de Méthodes d'Ensemble et d'Algorithmes à Base d'Arbres
Complexité des Modèles SVM dans Certains Cas

Quand utiliser les SVM?

CLASSIFICATION BINAIRE

Les SVM sont particulièrement adaptées à la classification binaire, où elles peuvent tirer parti de leur capacité à trouver des hyperplans de décision optimaux.

DONNEES A HAUTE DIMENSION

Lorsque le nombre de caractéristiques est élevé par rapport à la taille de l'ensemble de données, les SVM peuvent être plus performantes que d'autres algorithmes.

PROBLEMES NON LINEAIRES

Les SVM avec des noyaux non linéaires sont utiles pour traiter des données dont la relation entre les caractéristiques et la classe n'est pas linéaire.

APPLICATIONS MEDICALES ET BIOLOGIQUES

Les SVM sont souvent utilisées dans des domaines tels que la bioinformatique

Avantages

Efficacité dans les Espaces de Haute Dimension
même lorsque le nombre de dimensions est supérieur au nombre d'échantillons

Bonne Généralisation
Les SVM sont robustes contre le surajustement, en particulier dans des situations où le nombre de caractéristiques est élevé par rapport à la taille de l'ensemble de données.

Adaptabilité aux Données Non Linéaires
Les noyaux permettent aux SVM de traiter des données non linéaires en les projetant dans un espace de dimension supérieure.

Inconvénients

Sensibilité à l'Échelle des Caractéristiques
Les SVM peuvent être sensibles à l'échelle des caractéristiques, donc **la normalisation des données est souvent nécessaire.**

Choix des Noyaux et Paramètres
Le choix du noyau et des paramètres peut avoir un impact significatif sur les performances, et trouver les valeurs optimales peut nécessiter une recherche approfondie.

Difficulté avec les Grands Ensembles de Données
L'entraînement des SVM peut être lent sur de grands ensembles de données.

Un peu de vocabulaire

Imaginez que vous êtes un superhéros dans le monde des données, et votre mission est de diviser les gentils (les points d'une classe) des méchants (les points d'une autre classe) de la manière la plus nette possible.

Vecteur de Support

(Support Vectors)

Un vecteur de données d'entraînement qui influence la position et l'orientation de la frontière de décision. Ce sont les points les plus proches de la frontière de décision.
Quand on utilise une Soft Margin pour déterminer la position du seuil

Hyperplan

Une frontière de décision qui sépare l'espace des caractéristiques en deux régions. Dans le cas d'une SVM linéaire, l'hyperplan est une ligne ou un plan.

Noyau

(Kernel)

Une fonction qui transforme l'espace des caractéristiques d'origine en un espace de dimension supérieure, permettant de trouver une frontière de décision non linéaire. Les noyaux communs incluent le noyau linéaire, le noyau polynomial et le noyau radial (RBF).

Marge

La distance entre la frontière de décision et les vecteurs de support les plus proches. Une SVM cherche à maximiser ces marges.

Maximum Margin Classifier

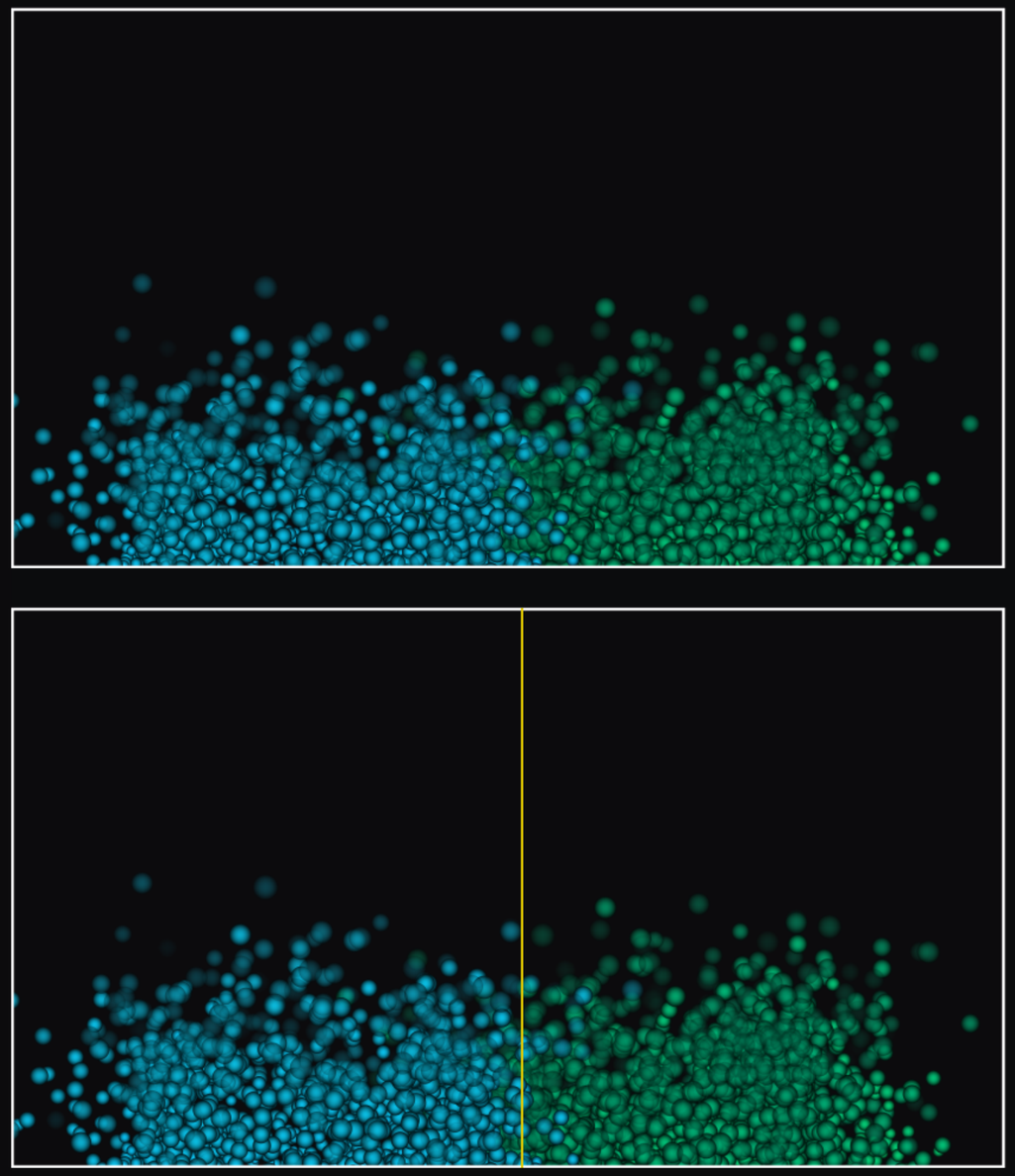
C'est quand le seuil/hyperplan nous donne la marge la plus large pour la classification.
(Attention, très sensible aux outliers!)

Soft Margin

C'est ainsi qu'est appelée la distance entre les observations les plus proches et le seuil lors d'une "misclassification".

Les SVM sont comme votre superpouvoir qui vous aide à dessiner une ligne ou un plan (hyperplan) pour séparer ces deux groupes d'une manière vraiment géniale.

Qu'est-ce que c'est que ça encore?



Machines à Vecteurs de Support (SVM)

Les SVM sont des algorithmes d'**apprentissage supervisé** utilisés pour la **classification** et la **régression** (prédictions sur des variables qualitatives et quantitatives).

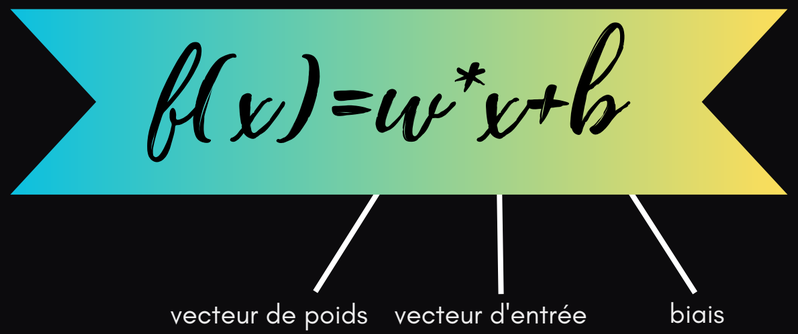
L'objectif principal est de **trouver un hyperplan** **dans un espace de dimension N** (où N est le nombre de caractéristiques) **qui sépare les exemples** de différentes classes de manière optimale.

L'objectif de SVM est de **maximiser la marge** (**distance entre l'hyperplan et les points les plus proches de chaque classe**) tout en **minimisant la classification incorrecte**. Cela se traduit par la résolution d'un problème d'**optimisation quadratique**.

Passons du côté obscur : les maths

HYPERPLAN

Un SVM résout un problème d'optimisation quadratique pour **trouver un hyperplan w*x+b qui maximise la marge entre les exemples positifs et négatifs tout en minimisant la classification incorrecte.**



C'est l'équation de l'hyperplan trouvé après que le SVM ait été entraîné avec succès. Une fois que le SVM a optimisé les poids (**w**) et le terme de biais (**b**), cette équation est utilisée pour classer de nouveaux points en fonction de leur position par rapport à l'hyperplan.

w représente le **vecteur de poids** qui **définit l'orientation de l'hyperplan**. Chaque composante de ce vecteur (*********) est associée à une caractéristique particulière de l'ensemble de données. Ces poids déterminent l'importance relative de chaque caractéristique dans la définition de l'hyperplan.

x représente le **vecteur d'entrée ou** le **vecteur de caractéristiques** d'une observation. Chaque composante *(x1,x2 ,...,xn)* correspond à une caractéristique spécifique de cette observation.

b représente le terme de **biais**, également appelé terme constant. Il **permet de déplacer l'hyperplan parallèlement sans changer son orientation**. Cela est crucial pour ajuster l'hyperplan de manière à bien séparer les deux classes.

*w*x* correspond à la somme pondérée des caractéristiques d'une observation, et l'ajout de **b** ajuste cette somme pour déterminer la position de l'observation par rapport à l'hyperplan. Si **f(x)** est positif, l'observation est d'un côté de l'hyperplan, et si c'est négatif, elle est de l'autre côté. En fonction du signe de **f(x)**, l'observation est classée dans l'une des deux classes.

COMMENT TROUVER LE MEILLEUR HYPERPLAN?

$$\min_{w,b} \frac{1}{2} \| w \|^2 \geq y_i (w^* x_i + b) \geq 1$$

Cette formulation inclut des **contraintes** pour **garantir que les points de chaque classe soient correctement classés et que l'hyperplan de séparation soit optimisé.**

Cela signifie qu'à travers cette formule mathématique, on cherche à trouver les meilleures valeurs pour **w** (vecteur de poids) et **b** (biais) pour séparer les exemples des différentes classes de la manière la plus élégante.

$$\min_{w,b} 1/2 \| w \|^2$$

On cherche à minimiser la norme euclidienne au carré du vecteur de poids **w**.
La minimisation de cette quantité favorise les hyperplans qui maximisent la marge entre les deux classes.

$\min$ indique la minimisation de la fonction qui suit.

w,b représentent les variables que nous cherchons à optimiser.

$1/2$ constante qui simplifie les calculs, elle n'affecte pas la solution optimale.

$||w||^2$ norme euclidienne au carré du vecteur de poids **w**. Cela représente la somme des carrés de ses composantes.

$$\geq y_i (w^* x_i + b) \geq 1$$

Ce sont les contraintes du problème d'optimisation. Ces contraintes garantissent que chaque point **xi** est correctement classé et respecte une marge minimale par rapport à l'hyperplan.

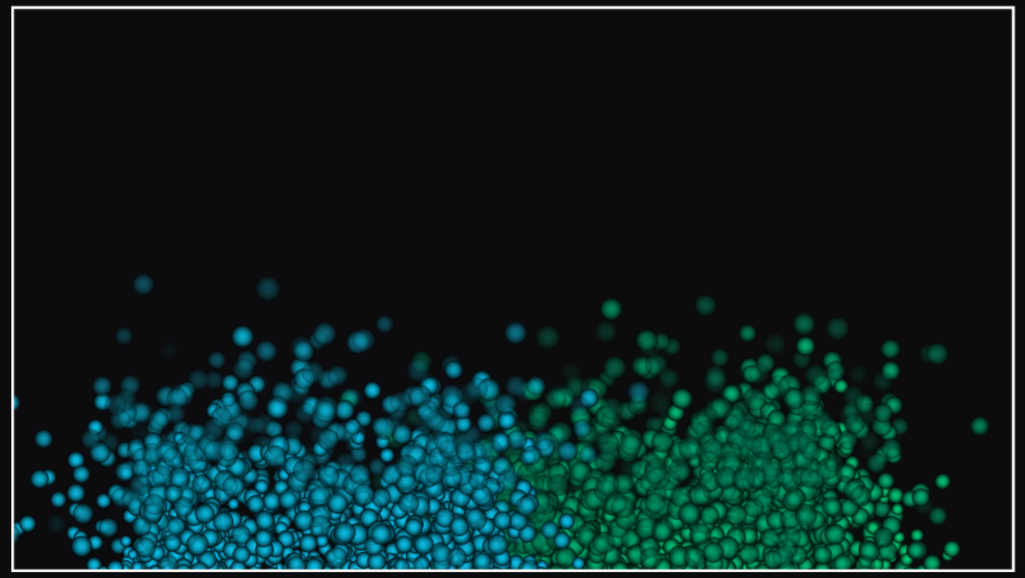
y_i c'est classe associée à l'observation ***xi*yi* = 1** pour la classe positive et ***yi* = -1** pour la classe négative.

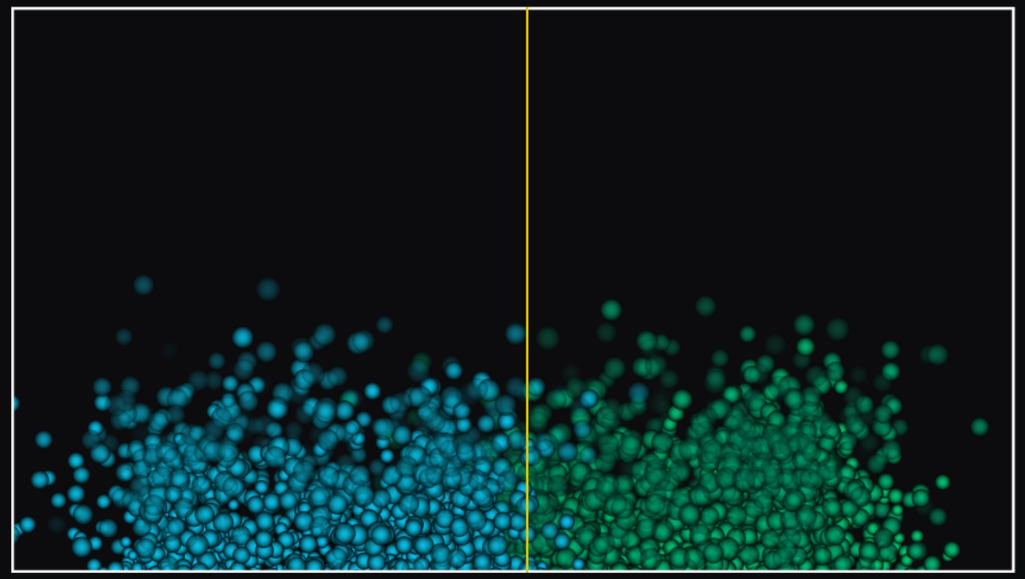
$w^* x_i + b$ c'est l'expression déterminant la position de **xi** par rapport à l'hyperplan défini par **w** et **b**.

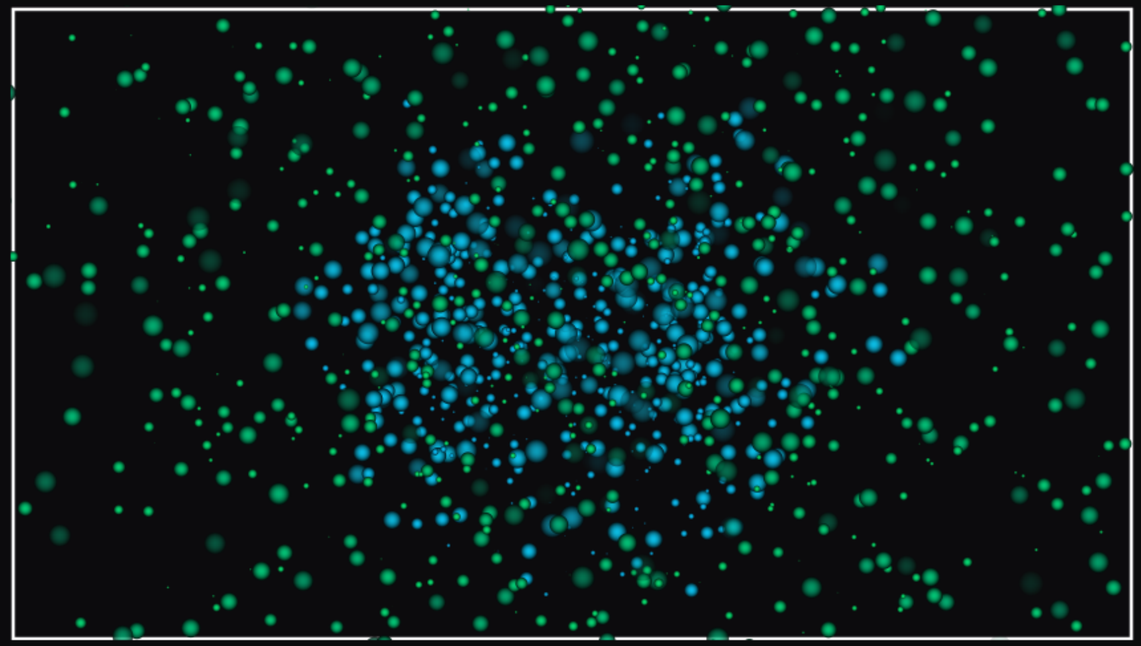
$y_i (w^* x_i + b)$ cette expression s'assure que les points sont correctement classés en garantissant que **yi** a le même signe que **w · xi+b**.

≥ 1 cette contrainte impose une marge minimale de 1 entre chaque point et l'hyperplan.

Que faire en cas de données non-linéaires?







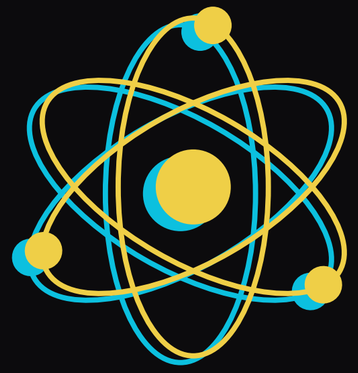
Les SVM peuvent **utiliser des noyaux pour traiter des données non linéaires**.

Un noyau transforme l'espace des caractéristiques pour permettre la séparation non linéaire. Le noyau le plus couramment utilisé est le noyau gaussien (RBF kernel).

NOYAU (**KERNEL**) DANS LES SVM

L'idée est d'appliquer des transformations sur nos données pour pouvoir les séparer linéairement.
Le plus difficile est de trouver la bonne transformation.

Le choix du noyau dépend de la nature des données et du problème. Le noyau linéaire est souvent utilisé lorsque les données sont linéairement séparables, tandis que les noyaux non linéaires, tels que le noyau RBF, sont plus appropriés lorsque les données présentent une structure complexe et non linéaire.



KERNEL TRICK

C'est le fait de calculer les relations entre les données sur des dimensions plus élevées sans transformer les données.

Le Kernel Trick réduit le nombre de calculs demandé pour les SVM en ignorant les maths qui transforment les données de basses dimensions à des hautes dimensions.

linear

La forme la plus simple de SVM, utilisant un hyperplan linéaire pour la séparation des classes.

La fonction de noyau linéaire est simplement le produit scalaire entre les vecteurs d'entrée.

polynomial

Transforme les données dans un espace de dimension supérieure en utilisant une fonction polynomiale.

La fonction de noyau polynomial a deux paramètres : le degré du polynôme et un terme constant.

RBF

Un noyau populaire qui transforme les données dans un espace de dimension infinie.

Il utilise une fonction gaussienne pour mesurer la similarité entre deux vecteurs d'entrée.

La fonction de noyau RBF dépend d'un paramètre appelé la largeur de la bande (gamma).

sigmoidal

Utilise une fonction sigmoïde pour transformer les données. Il peut être utile dans certains cas, mais n'est pas aussi couramment utilisé que d'autres noyaux.

noyau personnalisé

Vous pouvez également définir votre propre fonction de noyau, à condition qu'elle satisfasse les conditions requises (positivité et symétrie).

Les hyperparamètres hyperimportants

Kernel

détermine le type de fonction noyau à utiliser (linéaire, polynomial, RBF, etc.)

C

PARAMETRE DE REGULARISATION

contrôle l'équilibre entre la maximisation de la marge et la minimisation de l'erreur.
Les valeurs plus élevées de C autorisent une marge plus étroite mais minimisent l'erreur de classification.

Ordre de grandeur : Typiquement dans la plage de 0,01 à 100.

- Petites valeurs de C : Pour des marges plus larges, tolérant quelques erreurs de classification. Utile lorsque les données sont bruyantes ou lorsque la priorité est de maximiser la marge.

- Grandes valeurs de C : Pour des marges plus étroites, réduisant les erreurs de classification. Utile lorsque les données sont relativement propres et que la priorité est une classification précise.

Gamma

POUR LES NOYAUX NON LINEAIRES

influence la forme de la frontière de décision. Des valeurs plus élevées rendent la frontière de décision plus complexe.

Ordre de grandeur : Typiquement dans la plage de 0,1 à 100.

degree (ou d)

degré à utiliser pour un noyau polynomial
sa valeur doit être un entier positif tel que 2,3,4...
Lorsque vous augmentez le degré du polynôme, vous augmentez également la complexité du modèle. Cependant, il est essentiel de trouver un équilibre, car une augmentation excessive de la complexité peut entraîner un overfitting

class_weight

permet de donner un poids différent à chaque classe lors de l'apprentissage du modèle (utile lorsque les classes sont déséquilibrées)

- Equilibrage des classes: en définissant **class_weight = "balanced"**, le modèle ajustera automatiquement les poids en fonction de l'inverse du nombre de données dans chaque classe.

- Personnalisation des poids : on peut spécifier des poids personnalisés pour chaque classe
  class_weight = {class_1: weight_1, class_2 : weight_2, ...}

En ajustant les poids, vous modifiez la manière dont le modèle considère les erreurs de chaque classe.

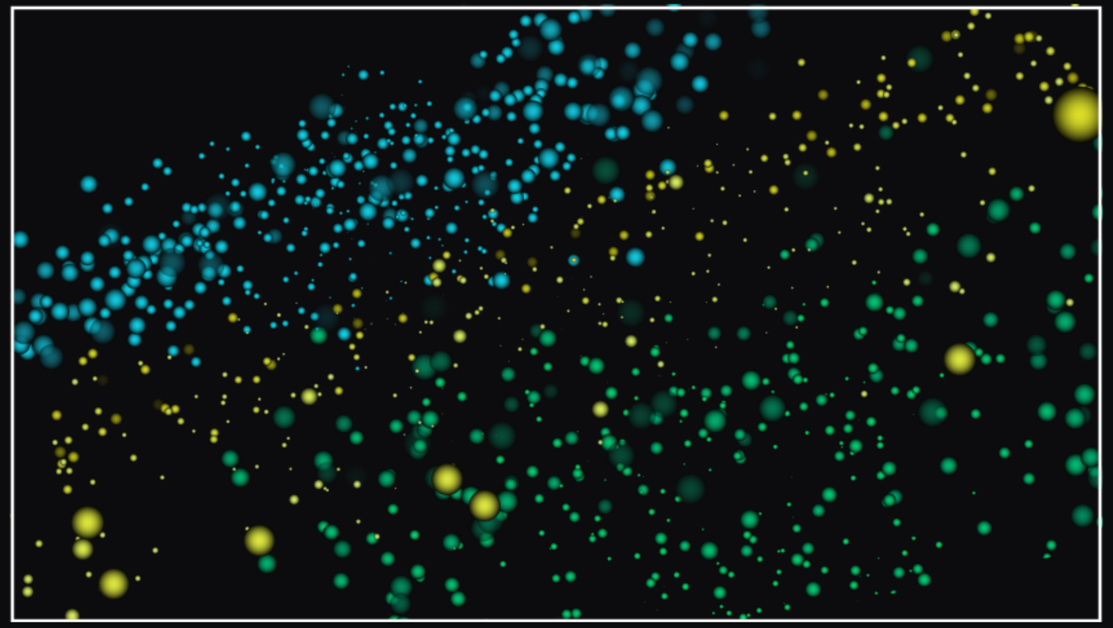
Multi-class classification

La classification multiclasse avec SVM (Support Vector Machines) peut être réalisée de différentes manières.

SVM est intrinsèquement un classificateur binaire, ce qui signifie qu'il est initialement conçu pour résoudre des problèmes de classification à deux classes.

Cependant, il existe des stratégies pour l'étendre à des problèmes multiclasse. Deux des approches les plus courantes sont le schéma "**One-Vs-One**" (**OvO**) et le schéma "**One-Vs-The-Rest**" (**OvR**).

- **One-Vs-One (OvO)** : Dans cette approche, un classificateur binaire est entraîné pour chaque paire possible de classes. Par exemple, si vous avez K classes, vous entraînez K * (K-1) / 2 classificateurs. Pendant la prédiction, chaque classificateur binaire vote pour une classe, et la classe avec le plus de votes est choisie.
- **One-Vs-The-Rest (OvR ou OvA - One-Vs-All)** : Dans cette approche, un classificateur binaire est entraîné pour chaque classe contre toutes les autres classes combinées. Vous avez donc K classificateurs binaires pour K classes. Pendant la prédiction, chaque classificateur binaire indique si l'échantillon appartient à sa classe respective ou non, et la classe avec le classificateur le plus "confiant" est choisie.



```
clf = svm.SVC(decision_function_shape='ovo')
clf = svm.SVC(decision_function_shape='ovr')
```

Petit récap'

2 classes

données linéaires

Import du modèle
from sklearn import svm

Création du classificateur SVM linéaire
clf_linear = svm.SVC(kernel='linear')

Ajout d'une pénalité de régularisation
clf_linear = svm.SVC(kernel='linear', C=0.5)

données non linéaires

Import du modèle
from sklearn import svm

Création du classificateur SVM polynomial
clf_poly = svm.SVC(kernel='polynomial')

(n'oubliez pas que vous avez accès à d'autres types de noyaux)

Ajout d'un degré au noyau
clf_poly = svm.SVC(kernel='polynomial', degree=5)

Multiclass

Imaginez que vous ayez un problème où vous devez classer des animaux en trois catégories : "Chien", "Chat" et "Oiseau".

one vs one

Dans cette approche, SVM crée un classificateur pour chaque paire possible d'animaux. Un classificateur pour "Chien" vs "Chat", un autre pour "Chien" vs "Oiseau", et enfin un pour "Chat" vs "Oiseau". Lorsqu'il doit classer un nouvel animal, chaque classificateur vote pour sa classe préférée, et la classe qui obtient le plus de votes gagne.

On peut garder notre modèle SVM polynomial et y ajouter un paramètre
pour gérer la classification multiclasse.

**clf_poly = svm.SVC(kernel='poly', degree=5,
decision_function_shape='ovo')**

one vs the rest

one vs all

Dans cette approche, SVM crée un classificateur pour chaque animal contre tous les autres. Un classificateur pour "Chien" contre "Chat" et "Oiseau" combinés, un autre pour "Chat" contre "Chien" et "Oiseau" combinés, et enfin un pour "Oiseau" contre "Chien" et "Chat" combinés. Chaque classificateur indique si l'animal est de son type ou non. L'animal est classé dans la catégorie du classificateur le plus "confiant".

decision_function_shape='ovr' **decision_function_shape='ova'**

MERCI POUR VOTRE ATTENTION