Méthodes de Simulation Informatique

Amaya Nogales Gómez
amaya.nogales-gomez@univ-cotedazur.fr

Licence 3 Informatique
Université Côte d'Azur

4 février 2022

Le cours

À quoi s'attendre?

Ce cours a pour objectif s'initier à la démarche scientifique: expliquer, prevoir et valider l'étude d'un phénomène à l'aide d'un base de données. Savoir présenter un problème, discuter les approches de résolution possibles, défendre des conclusions devant des pairs, et réciproquement savoir évaluer les résultats d'autres études sur des sujets connexes.

Plan du cours

1. Initiation à l'Intelligence Artificielle: techniques d'apprentissage automatique.
   - Préliminaires et bases d'apprentissage automatique.
   - Algorithmes d'apprentissage supervisé.
2. Extraction de connaissances a partir de données: méthodes descriptives et prédictives.
3. Initiation aux techniques de validation.
4. Initiation aux techniques de communication scientifique.
   - LaTeX
   - Présentation et posters en beamer.

Aperçu du cours

1. Logistique
   - 10 seances: CM (2H) + TP (3H).
   - Récapitulatif et/ou QCM au début de chaque cours.

Aperçu du cours

1. Logistique
   - 10 seances: CM (2H) + TP (3H).
   - Récapitulatif et/ou QCM au début de chaque cours.
2. Contrôle des connaissances
   - Contrôle continu: (30%), très probablement le 18 mars.
   - Contrôle terminal: projet et présentation final par groupes (70%).

Aperçu du cours

1. Logistique
   - 10 seances: CM (2H) + TP (3H).
   - Récapitulatif et/ou QCM au début de chaque cours.
2. Contrôle des connaissances
   - Contrôle continu: (30%), très probablement le 18 mars.
   - Contrôle terminal: projet et présentation final par groupes (70%).
3. Contact
   - E-mail: amaya.nogales-gomez@univ-cotedazur.fr
   - Moodle.
   - Bureau: 421, Templiers 1, Polytech'Nice Sophia.

Plan du cours

1. Introduction
   - Préliminaires
   - Python: numpy, pandas
2. Base de données
   - Generation des données synthétiques
   - Base de données reels
3. Analyse descriptive
4. Techniques d'apprentissage supervisée
   - Support Vector Machines
   - Régression logistique
5. Contrôle de connaissances
6. Techniques de validation
7. Elements de la méthodologie scientifique
8. LaTeX
   - Écriture de textes scientifiques
   - Beamer: présentations et posters scientifiques

NumPy

Qu'est-ce que c'est NumPy

- Numpy est une bibliothèque open-source pour travailler efficacement avec des tableaux.
- Développé en 2005 par Travis Oliphant, le nom signifie Numerical Python.
- En tant que bibliothèque principal de science des données en Python, de nombreuses autres bibliothèques en dépend.

Pourquoi NumPy est-il si populaire ?

NumPy améliore considérablement la facilité et performances de travail avec des tableaux multidimensionnels.

Avantages de NumPy

- Les opérations mathématiques sur les tableaux de NumPy sont jusqu'à 50 fois plus rapides que sur des listes avec de boucles.
- NumPy stocke les termes du tableau dans un emplacement unique ordonné dans la mémoire, éliminant les redondances en faisant en sorte que tous les éléments soient du même type et en utilisant pleinement les processeurs modernes.
- Les avantages en termes d'efficacité deviennent particulièrement évidents lorsque vous travaillez sur des tableaux avec des milliers ou des millions d'éléments.

Avantages de NumPy

- Il offre une syntaxe d'indexation pour accéder facilement à des portions de données dans un tableau.
- Il contient des fonctions intégrées qui améliorent la qualité de vie lorsque vous travaillez avec des tableaux et des mathématiques, telles que des fonctions pour l'algèbre linéaire, les transformations de tableaux et les mathématiques matricielles.
- Il nécessite moins de lignes de code pour la plupart des opérations mathématiques que avec les listes Python natives.

Quelle est la relation entre NumPy, Scikit-learn et Pandas?

- NumPy fournit une base sur laquelle d'autres packages de science des données sont construit, y compris SciPy, Scikit-learn et Pandas.
- Scikit-learn étend NumPy et SciPy avec un apprentissage automatique avancé algorithmes. (CM 4 et 5)
- Pandas étend NumPy en fournissant des fonctions d'exploration analyse de données, statistiques et visualisation de données.

Liste de fonctions utiles de NumPy

```
>>> import numpy as np
ModuleNotFoundError: No module named 'numpy'
>>> pip install numpy
```

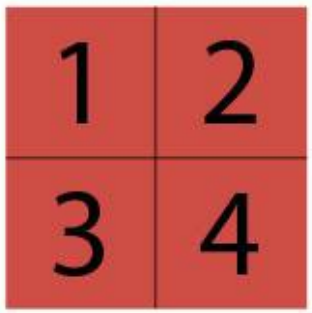
- Creation de tableaux: arange, array, copy, empty, eye, identity, linspace, logspace, mgrid, ogrid, ones, zeros
- Manipulations: concatenate, diagonal, repeat, reshape, transpose
- Questions: in, all, any, nonzero, where
- Ordernnation: argmax, argmin,max, min, sort
- Opérations: cumprod, cumsum, prod, real, sum
- Statistique: cov, mean, std, var
- Algèbre linéaire: dot, outer, vdot

Tableaux NumPy (*array*)

- Objet central du package NumPy.
- Un tableau NumPy unidimensionnel: un vecteur.
- Bidimensionnel: une matrice.
- Tridimensionnel: un tenseur (ensemble de matrices).

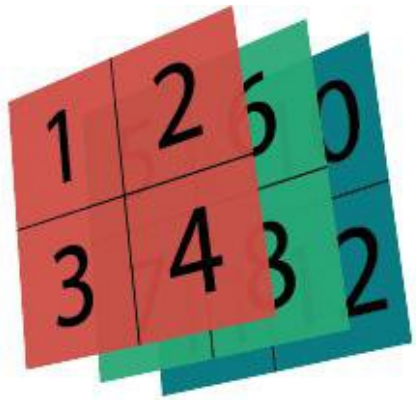
Vector

```
np.array([1, 2])
```



Matrix

```
np.array([[1, 2], [3, 4]])
```



3D Matrix

```
np.array([[[1, 2], [3, 4]],
          [[5, 6], [7, 8]],
          [[9, 10], [11, 12]]])
```

Image source: `https://www.learndatasci.com/`

- Similaires aux listes en Python, sauf que chaque élément d'un tableau doit être du même type, généralement *float* ou *int*.
- Les tableaux rendent les opérations avec de grandes quantités de données numériques très rapides et sont généralement beaucoup plus efficace que les listes.

```
>>> from numpy import *
>>> v=array([1,2,3,4])
>>> type(v)
<class 'numpy.ndarray'>
>>> import numpy as np
>>> w=np.array([1,2,3,4])
>>> type(w)
<class 'numpy.ndarray'>
```

```
#La fonction array prend deux arguments : la liste
#a  convertir en tableau et le type de chaque terme
>>> a = np.array([[1, 2, 3], [4, 5, 6]], float)
>>> a
array([[ 1., 2., 3.],[ 4., 5., 6.]])
#Les differents axes sont accessibles a  l'aide de
#virgules a  l'interieur du crochets
>>> a[0,1]
2.0
```

- Le découpage fonctionne comme d'habitude.
- L'utilisation d'un seul ":" dans une dimension indique utilisation de tout le long de cette dimension.

```
#a[rangee,colonne]
>>> a = np.array([[1, 2, 3], [4, 5, 6]], float)
>>> a[1,:]
array([ 4., 5., 6.])
>>> a[:,2]
array([ 3., 6.])
>>> a[-1:,-2:]
array([[ 5., 6.]])
```

```
>>> a = np.array([[1, 2, 3], [4, 5, 6]], int)
#shape retourne un tuple avec la taille de chaque
#dimension du tableau:
>>> a.shape
(2, 3)
#dtype indique le type de valeurs stockees dans
# le tableau:
>>> a.dtype
dtype('int')
```

La fonction *in* peut être utilisée pour tester si certain valeurs sont présentes dans un tableau:

```
>>> s = np.array([[1,1,1],[2,2,2],[3,3,3]], float)
>>> 2 in s
True
>>> 99 in s
False
```

Les tableaux peuvent être remodelés à l'aide de tuples qui spécifient de nouvelles dimensions.

```
>>> a = np.array([ 0., 1., 2., 3., 4., 5., 6., 7., 8., 9.]
>>> a.reshape((5, 2))
array([[0., 1.],
       [2., 3.],
       [4., 5.],
       [6., 7.],
       [8., 9.]])
```

Copier des tableaux

```
>>> x = np.array([-1, 0, 1], float)
>>> y = x
>>> z = x.copy()
>>> x[0] = 99
>>> x
array([99., 0., 1.])
>>> y
array([99., 0., 1.])
>>> z
array([-1., 0., 1.])
```

Transposer et aplatir des tableaux

```
>>> s = np.array([[1,1,1],[2,2,2],[3,3,3]], float)
>>> s.transpose()
array([[1., 2., 3.],
       [1., 2., 3.],
       [1., 2., 3.]])
>>> s.flatten()
array([1., 1., 1., 2., 2., 2., 3., 3., 3.])
```

Enchaîner des tableaux

```
>>> a = np.array([[1, 2], [3, 4]])
>>> b = np.array([[5, 6]])
>>> np.concatenate((a, b), axis=0)
array([[1, 2],
       [3, 4],
       [5, 6]])
>>> np.concatenate((a, b.T), axis=1)
array([[1, 2, 5],
       [3, 4, 6]])
```

Renvoie des nombres régulièrement espacés sur un intervalle spécifié.

```
>>> np.linspace(2.0, 3.0, num=5)
array([2.  , 2.25, 2.5 , 2.75, 3.  ])
```

Similaire à *linspace*, mais utilise une taille de pas (au lieu du nombre d'échantillons).

```
>>> np.arange(3, dtype=float)
array([ 0.,  1.,  2.])
>>> np.arange(3,7)
array([3, 4, 5, 6])
>>> np.arange(3,7,2)
array([3, 5])
```

zeros et *ones*

- Les fonctions *zeros* et *ones* créent de nouveaux tableaux de dimensions spécifiées remplis de ces valeurs.
- Tres utilisées pour créer de nouveaux tableaux:

```
>>> np.ones((4,3), dtype=float)
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
>>> np.zeros(6, dtype=int)
array([0, 0, 0, 0, 0, 0])
```

Matrice d'identité

```
>>> np.identity(4, dtype=float)
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]])
>>> np.eye(4, k=2, dtype=float)
array([[0., 0., 1., 0.],
       [0., 0., 0., 1.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

Mathématiques du tableau

- Les opérations mathématiques sont appliquées élément par élément.
- Les tableaux doivent avoir la même taille:

```
>>> x = np.array([-1,7,-3], float)
>>> y = np.array([1,2,0], float)
>>> x+y
array([ 0.,  9., -3.])
>>> x-y
array([-2.,  5., -3.])
```

```
>>> x = np.array([-1,7], float)
>>> y = np.array([1,2,0], float)
>>> x+y
---------------------------------------------------------
ValueError                                Traceback
(most recent call last)
/tmp/ipykernel_62514/2163207258.py in <module>
----> 1 x+y
ValueError: operands could not be broadcast together
with shapes (2,) (3,)
```

Les fonctions floor, ceil et rint donnent l'entier inférieur, supérieur ou le plus proche (arrondi):

```
>>> b = np.array([9.1, 9.3, 9.9], float)
>>> np.floor(b)
array([9., 9., 9.])
>>> np.floor(b)
array([10., 10., 10.])
>>> np.rint(b)
array([9., 9., 10.])
```

Opérations sur les tableaux

```
>>> a=np.array([10,20,30],float)
a.sum()
60.0
>>> a.prod()
6000.0
>>> np.sum(a)
60.0
>>> a.mean()
66.66666666666667
```

Terme minimal et maximal

```
>>> a = np.array([22, 12, 97], float)
>>> a.min()
12.0
>>> a.max()
97.0
```

Les fonctions *argmin* et *argmax* renvoient les indices des termes minimal et maximal:

```
>>> a.argmin()
1
>>> a.argmax()
2
```

Produit de matrices

```
>>> M = np.array([[1, 2, 3], [4, 5, 6],[7,8,9]])
>>> M * M # Multiplication element par element
array([[ 1,  4,  9],
       [16, 25, 36],
       [49, 64, 81]])
>>> np.dot(M,M) #Multiplication de matrices
array([[ 30,  36,  42],
       [ 66,  81,  96],
       [102, 126, 150]])
```

Nombres aléatoires: germe ou graine

- Une partie importante de toute **simulation** est la possibilité de tirer des nombres aléatoires.
- On utilise de routines de génération de nombres pseudo-aléatoires intégrées de NumPy dans le sous-module *random*.
- Ils sont générés à partir d'un nombre appelé germe qu'est une valeur entière.
- Tout programme qui démarre avec le même germe générera exactement la même séquence de nombres aléatoires à chaque exécution.
- Il n'est pas nécessaire de spécifier le germe.

```
>>> np.random.seed(293423)
```

Génération de nombres aléatoires

Nombres aléatoires d'une distribution uniforme.

```
>>> np.random.rand(2,3)
array([[ 0.50431753, 0.48272463, 0.45811345],
[ 0.18209476, 0.48631022, 0.49590404]])
>>> np.random.random()
0.70110427435769551
>>> np.random.randint(5, 10)
9
```

Pandas

- Outil le plus important à la disposition des Data Scientists.
- **Pandas** est dérivé du terme "panel data", un terme économétrique pour les ensembles de données qui incluent des observations sur plusieurs périodes de temps pour les mêmes individus.

- A travers des pandas on travaille avec nos données en les nettoyant, les transformant et les analysant.
- Par exemple, si on souhaite explorer une base de données stocké dans un CSV, Pandas extraira les données de ce CSV dans un DataFrame et on pourra:
  - Calculer des statistiques comme la moyenne, la médiane, le maximum ou le minimum de chaque colonne.
  - A quoi ressemble la distribution des données dans la colonne C ?
  - Nettoyer les données: suppression ou remplacement des valeurs manquantes.
  - Visualisez les données (*Matplotlib*): diagramme à barres, histogrammes et plus.
  - Stocker les données nettoyées et transformées dans un fichier ou une base de données.

Pandas: premiers pas

```
>>> import pandas as pd
ModuleNotFoundError: No module named 'pandas'
>>> pip install pandas
```

Series et DataFrames

- Principaux composants: *Series* et *DataFrame*.
- Series est essentiellement une colonne.
- DataFrame est un tableau multidimensionnel composé d'une collection de Series.

Series

	apples
0	3
1	2
2	0
3	1

+

Series

	oranges
0	0
1	3
2	7
3	2

=

DataFrame

	apples	oranges
0	3	0
1	2	3
2	0	7
3	1	2

Image source: `https://www.learndatasci.com/`

Créer des DataFrames à partir de zéro

```
import pandas as pd
data = {
    'cas positifs': [9, 2, 0, 1],
    'cas contact': [30, 13, 17, 5]
}
covid = pd.DataFrame(data)

covid
```

	cas positifs	cas contact
0	9	30
1	2	13
2	0	17
3	1	5

Créer des DataFrames à partir de zéro

L'indice (*index*) du DataFrame *covid* par defaut sont des nombres 0-3.
On peut créer nôtre propre *index*.

```
covid = pd.DataFrame(data, index=['Licence 1', 'Licence 2',
                                  'Licence 3', 'Master 1'])

covid
```

	cas positifs	cas contact
Licence 1	9	30
Licence 2	2	13
Licence 3	0	17
Master 1	1	5

Créer des DataFrames à partir de zéro

```
df = pd.read_csv('covid.csv')

df
```

	Unnamed: 0	cas positifs	cas contact
0	Licence 1	9	30
1	Licence 2	2	13
2	Licence 3	0	17
3	Master 1	1	5

Créer des DataFrames à partir de zéro

```
df = pd.read_csv('covid.csv', index_col=0)

df
```

	cas positifs	cas contact
Licence 1	9	30
Licence 2	2	13
Licence 3	0	17
Master 1	1	5

Reconvertir en fichier csv ou json

- Après un travail de nettoyage de vos données, on peut les enregistrer dans un fichier.
- Pandas fournit des commandes tres intuitives commandes pour l'enregistrer:

```
>>>df.to_csv('new_covid.csv')
>>>df.to_json('new_covid.json')
```

Opérations avec DataFrames

```
compas_df = pd.read_csv("propublica.csv", index_col=0)
```

```
compas_df.head()
```

	Age	C_charge_degree	Race	Age_cat	Score_text	sex	priors_count	days_screening_arrest (before)	decile_sc
1	69	F	Other	Greater than 45	Low	Male	0	-1	
2	34	F	African-American	25 - 45	Low	Male	0	-1	
3	24	F	African-American	Less than 25	Low	Male	4	-1	
4	44	M	Other	25 - 45	Low	Male	0	0	
5	41	F	Caucasian	25 - 45	Medium	Male	14	-1	

Opérations avec DataFrames

```
compas_df.tail(2)
```

	Age	C_charge_degree	Race	Age_cat	Score_text	sex	priors_count	days_screening_arrest (before)	decile
6171	33	M	African-American	25 - 45	Low	Female	3	-1	
6172	23	F	Hispanic	Less than 25	Low	Female	2	-2	

```
compas_df.shape
```

```
(6172, 13)
```

Gérer les répétitions

```
temp_df = compas_df.append(compas_df)

temp_df.shape
```

```
(12344, 13)
```

```
temp_df = temp_df.drop_duplicates()

temp_df.shape
```

```
(6172, 13)
```

Inplace

- On peut eviter d'assigner des DataFrames à la même variable chaque fois on fait des operations.
- Pandas a l'argument *inplace* sur de nombreux ses méthodes.
- *inplace=True* modifiera l'objet DataFrame en question:

```
>>>temp_df.drop_duplicates(inplace=True)
```

Nettoyage de colonnes

```
compas_df.columns
```

```
Index(['Age', 'C_charge_degree', 'Race', 'Age_cat', 'Score_text', 'sex',
       'priors_count', 'days_screening_arrest (before)', 'decile_score',
       'is_recid', 'two_year_recid', 'c_jail (in)', 'c_jail (out)'],
      dtype='object')
```

```
compas_df.rename(columns={
        'days_screening_arrest (before)': 'days_screening_arrest',
        'c_jail (in)': 'c_jail_in',
        'c_jail (out)': 'c_jail_out',
    }, inplace=True)


compas_df.columns
```

```
Index(['Age', 'C_charge_degree', 'Race', 'Age_cat', 'Score_text', 'sex',
       'priors_count', 'days_screening_arrest', 'decile_score', 'is_recid',
       'two_year_recid', 'c_jail_in', 'c_jail_out'],
      dtype='object')
```

```
compas_df.columns = [col.lower() for col in compas_df]

compas_df.columns
```

```
Index(['age', 'c_charge_degree', 'race', 'age_cat', 'score_text', 'sex',
       'priors_count', 'days_screening_arrest', 'decile_score', 'is_recid',
       'two_year_recid', 'c_jail_in', 'c_jail_out'],
      dtype='object')
```

Valeurs manquants

```
compas_df.isnull()
```

	age	c_charge_degree	race	age_cat	score_text	sex	priors_count	days_screening_arrest	decile_sc
1	False	False	False	False	False	False	False	False	Fa
2	False	False	False	False	False	False	False	False	Fa
3	False	False	False	False	False	False	False	False	Tr
4	False	False	False	False	False	False	False	False	Fa
5	False	False	False	False	False	False	False	False	Fa
...	...	...	...	...	...	...	...	...	
6168	False	False	False	False	False	False	False	False	Fa
6169	False	False	False	False	False	False	False	False	Fa
6170	False	False	False	False	False	False	False	False	Fa
6171	False	False	False	False	False	False	False	False	Fa
6172	False	False	False	False	False	False	False	False	Tr

6172 rows × 13 columns

```
compas_df.isnull().sum()
```

```
age                        0
c_charge_degree            0
race                       0
age_cat                    0
score_text                 0
sex                        0
priors_count               0
days_screening_arrest      0
decile_score               9
is_recid                   0
two_year_recid             0
c_jail_in                  0
c_jail_out                 0
dtype: int64
```

Suppression des valeurs nulles

```
compas_df.dropna()
```

	age	c_charge_degree	race	age_cat	score_text	sex	priors_count	days_scr
1	69	F	Other	Greater than 45	Low	Male	0	
2	34	F	African-American	25 - 45	Low	Male	0	
4	44	M	Other	25 - 45	Low	Male	0	
5	41	F	Caucasian	25 - 45	Medium	Male	14	
7	39	M	Caucasian	25 - 45	Low	Female	0	

Paramètre *axis=1*

- Le paramètre axis=1 affect les colonnes.
- L'explication vient d'output de *.shape*:

```
>>>compas_df.shape
(6172,13)
```

- Il s'agit d'un tuple qui représente la forme du DataFrame, soit 6172 rangees et 13 colonnes.
- Commes dans NumPy, l'index 0 represent les rangees et l'index 1 les colonnes.

Imputation

```
decile = compas_df['decile_score']
```

```
decile.head()
```

```
1    1.0
2    3.0
3    NaN
4    1.0
5    6.0
Name: decile_score, dtype: float64
```

```
decile_mean = decile.mean()

decile_mean
```

```
4.418465033263021
```

```
decile.fillna(decile_mean, inplace=True)
```

```
compas_df.isnull().sum()
```

```
age                        0
c_charge_degree            0
race                       0
age_cat                    0
score_text                 0
sex                        0
priors_count               0
days_screening_arrest      0
decile_score               0
is_recid                   0
two_year_recid             0
c_jail_in                  0
c_jail_out                 0
dtype: int64
```

Sélection de colonnes

```
age_col = compas_df['age']

type(age_col)
```

```
pandas.core.series.Series
```

Sélection de rangées

```
compas_df.head(2)
```

	age	c_charge_degree	race	age_cat	score_text	sex	priors_count	days_screening_arrest	decile_score	is_recid	t
1	69	F	Other	Greater than 45	Low	Male	0	-1	1.0	0	
2	34	F	African-American	25 - 45	Low	Male	0	-1	3.0	1	

```
Afr_34 = compas_df.loc[2]

Afr_34
```

```
age                              34
c_charge_degree                   F
race                 African-American
age_cat                     25 - 45
score_text                      Low
sex                            Male
priors_count                      0
days_screening_arrest            -1
decile_score                    3.0
```

Sélection de rangées

	age	c_charge_degree	race	age_cat	score_text	sex	priors_count	days_screening_arrest	decile_score	is_recid	tv
1	69	F	Other	Greater than 45	Low	Male	0	-1	1.0	0	
2	34	F	African-American	25 - 45	Low	Male	0	-1	3.0	1	

```
Afr_34 = compas_df.iloc[1]

Afr_34
```

```
age                          34
c_charge_degree               F
race                 African-American
age_cat                   25 - 45
score_text                    Low
sex                          Male
priors_count                    0
days_screening_arrest          -1
decile_score                  3.0
is_recid                        1
two_year_recid                  1
```

Sélections conditionnelles

```
condition = (compas_df['race'] == "African-American")

condition.head()
```

```
1    False
2     True
3     True
4    False
5    False
Name: race, dtype: bool
```

```
compas_df[compas_df['decile_score'] >=5].head(3)
```

	age	c_charge_degree	race	age_cat	score_text	sex	priors_count	days_screening_arrest	decile_score	is_recid
5	41	F	Caucasian	25 - 45	Medium	Male	14	-1	6.0	1
9	23	M	African-American	Less than 25	Medium	Male	3	0	6.0	1
15	25	F	African-American	25 - 45	High	Male	3	-1	10.0	0

Application de fonctions

```
def fonction_coupable(x):
    if x >= 5.0:
        return "coupable"
    else:
        return "non-coupable"
```

```
compas_df["decision"] = compas_df["decile_score"].apply(fonction_coupable)

subset=compas_df[['decile_score','decision']]
subset.head(3)
```

	decile_score	decision
1	1.000000	non-coupable
2	3.000000	non-coupable
3	4.418465	non-coupable

```
compas_df["decision"] = compas_df["decile_score"].apply(lambda x: 'coupable'
                                                         if x >= 5.0 else 'non-coupable')

subset=compas_df[['decile_score','decision']]
subset.head(3)
```

	decile_score	decision
1	1.000000	non-coupable
2	3.000000	non-coupable
3	4.418465	non-coupable

Projet

L'un des principaux objectifs de ce cours est de vous préparer à appliquer des algorithmes d'apprentissage automatique à des tâches du monde réel. Le projet final est destiné à vous lancer dans ces directions.

Thèmes du projet

Votre première tâche consiste à choisir un problème sur lequel travailler.

Livrables du projet

Proposition, rapport d'avancement, livrable final et présentation

Evaluation

Les projets seront évalués en fonction de:

- La qualité technique du travail : Le contenu technique a-t-il du sens ? Le travail proposé est-il intéressant et raisonnable ? Votre travail propose-t-il un nouvel aperçu du problème?
- Importance: le problème choisi est-il un problème "réel" et intéressant? ou est-ce juste un problème de jouet? Ce travail pourrait-il être utile et avoir un impact ?
- La nouveauté du travail: le projet applique-t-il une technique commune à un problème bien étudié, ou certaines de ces techniques sont-elles relativement inexplorées?

Afin de mettre en évidence ces composants, il est important que vous présentiez une discussion sur l'apprentissage acquis pendant le développement de votre projet et que vous résumiez comment votre travail se compare aux approches existantes.

Proposition de projet

Dans la proposition de projet, vous choisirez une idée de projet sur laquelle travailler.

- **Format**: Votre proposition doit être un document PDF, donnant le titre du projet, les noms complets de tous les membres de votre équipe, et une description de 300 à 500 mots de ce que vous envisagez de faire.
- **Contenu**: Votre proposition de projet doit inclure les informations suivantes:
  1. Motivation : à quel problème vous attaquez-vous ? Est-ce une application ou un résultat théorique ?
  2. Méthode: quelles techniques d'apprentissage automatique prévoyez-vous d'appliquer ou d'améliorer?
  3. Expériences prévues: quelles expériences prévoyez-vous d'exécuter? Comment comptez-vous évaluer votre algorithme d'apprentissage automatique?

Livrable final

Votre livrable final comprendra le rapport de projet ainsi que le code produit pour le projet. L'utilisation d'un site Github pour ce dernier est fortement recommandée.

- **Format**: Le rapport de projet final ne peut pas dépasser 5 pages (y compris les figures). Les pages supplémentaires ne sont autorisées que pour des références. Vous pouvez utilisez les directives du cours d'apprentissage automatique de Standford sur la façon de rédiger le rapport final (lien).
- **Contributions**: veuillez inclure une section décrivant sur ce que chaque membre de l'équipe a travaillé et contribué au projet.

Livrable final

- **Code**: veuillez inclure un lien vers un référentiel Github ou un fichier zip avec le code de votre projet final.
- **Evaluation** : Le rapport final sera evalué sur la clarté du rapport, la pertinence du projet aux sujets enseignés dans le cours, et la qualité technique.

Présentations finales

- Les projets seront présentés lors d'une session avec un jury composé au moins pour les deux enseignants du cours. Chaque équipe doit préparer une presentation de leur travail sous forme de poster ou diapositives en LaTeX.
- **Evaluation**: les présentations seront notées en fonction de leur qualité et de leur clarté, du contenu technique, et la connaissance démontrée par l'équipe lors de la discussion du travail avec les membres du jury.
- **Duration**: à déterminer.
- **Date**: à déterminer (le plus tard possible).

Inspiration pour les sujets

Vous pouvez trouver de l'inspiration dans la liste des projets du cours *Machine Learning and Intelligent Systems* d'Eurecom (lien).

- Qlearnkit: A Python toolkit for quantum machine learning
- Machine Learning in Predictice Maintenance
- Predicting the authenticity of tweets from Twitter during a disaster
- Country Detection by Image Classification
- Heart Stroke Prediction
- Wine quality prediction
- Eyes images segmentation using Active Learning for data annotation
- Machine Learning for Autism Spectrum Disorder Diagnosis
- Analysis of the MAGIC Gamma Telescope Data Set
- Twitter Sentiment Analysis
- Music Genre Classification

Machine Learning and Intelligent Systems
MALIS - Fall 2021 Fri 13:30 - 16:45

List of Projects - Fall 2021

Posted on November 1, 2021

Invited Jury Members

François Bremond, INRIA
Fréderic Cazals, INRIA
Alix Lhéritier, Amadeus
Amaya Nogales Gómez, Université Côte d'Azur

Best project presentation award

Qlearnkit: A Python toolkit for quantum machine learning
G. Corallo, M. Pronesti, F. Tiblias

Honorable mentions

Machine Learning in Predictice Maintenance
B. Salon, Z Thiry, M Ramon

Image Captcha simulation and labeling using Machine Learning
D Sivasankaran, DT Johnson, A Burse

Eyes images segmentation using Active Learning for data annotation
F Germinario, A Ghiglione, G Gioetto

Analysis of the MAGIC Gamma Telescope Data Set
F Capano, D Falcetta, S Papicchio

Comic book classification: DC vs Marvel
I Panagiotis Pitsiorlas, H Rechatin, E Øien

Automatic colorization of archives using Machine Learning
Y Moudere, L Marcadet, J de Mathan

All projects

Qlearnkit: A Python toolkit for quantum machine learning

Exemple I

La Liga games predictions

Nour Jamoussi, Marco Klepatzky

Introduction

This project's aim is to **predict** the odds (W/L/D) between two different teams from the football Spanish League "La Liga" based on the 2020-2021 season input data, which was scraped from a sports statistics website. Afterwards, it was curated, processed and used to run experiments to validate the accuracy of the models implemented.

Features

The data consists in **37 features**. The logic behind the model is tied to not knowing certain variables (i.e. yellow/red cards, strikes) as the odds should be ran before a match. So we proceeded by doing for each score, **the average of the three last matches** and the **average of all the matches** played **before the targeted one.** Then to emphasize the difference between opponent teams, for each team and for each score we computed a **subtraction of the score of the team by the score of the opponent team**.

Table 1. Explanatory example of the data used

Match	Team	Last3AvgScore	Avg Score
Alaves vs Sevilla	1	-0.47	-0.3
Alaves vs Sevilla	2	0.47	0.3

Methods

The classification models used were **KNN**, **random forests**, **SVM**, **logistic regression** and **logistic regression** with **lasso** regularization. Additional feature engineering, recursive feature elimination and normalization/standardization were executed on the inputs before the model testing, so as to preserve the **distribution** and reduce possible induced **bias** caused by the magnitude difference of the independent variables.

Overfitting was prevented by validating the test and train accuracies. The "curse of **dimensionality**" was avoided by running instances of RFE to select the optimal number of features which prevented the classifier's performance degradation.

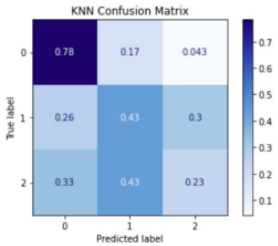


Fig 1. KNN Confusion Matrix

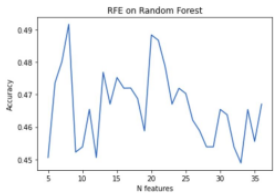


Fig 2. RFE on Random Forest

Conclusion

The highest accuracy on the general data was **52.63%**, given by the **logistic regression with lasso regularization**. The initial approach of the problem was achieved, by developing an algorithm that could determine the outcome of a football match between two teams with a decent accuracy.

Table 2. The accuracy of the logistic regression with lasso regularization.

Experiment	Accuracy
All matches	52.63%
Real Madrid matches	75%

Future Work

Additional ideas to pursue progressive **increments** on the existing **accuracy**:

-Retrieving a larger and more curated database, considering **exogenous** features (injuries, total play hours, etc.)

-Implement specific custom hyperparameter tuning for the models described.

-On the commercial level, deploying an application for model **benchmarking** and bet assistance, showing the benefits and weakness of each model on any given input match.

Exemple II

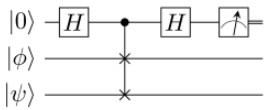
What is Qlearnkit?

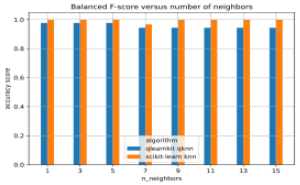
- Tested, documented Python package
- Built with Qiskit
- Implements K-Means, K-NN, SVM, Ridge

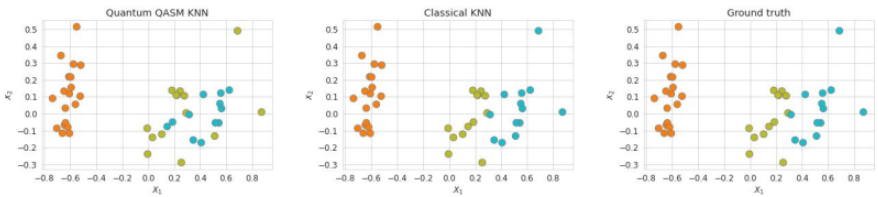
Why quantum in machine learning?

- Quantum speedup
  (linear algebra routines, distance calculation...)
- Faster algorithms based on distance
  (K-nn, SVM, K-Means, ...)

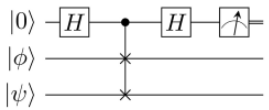
How does it perform? QKNN





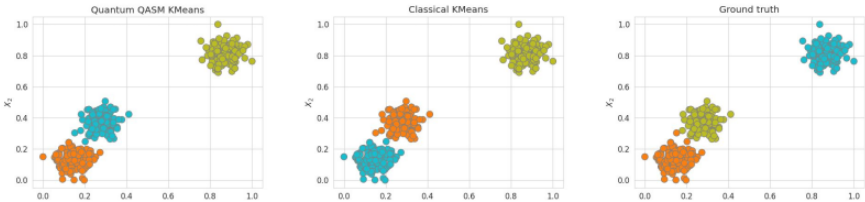


How does it perform? QKMeans



$$P(|1\rangle) = \frac{1 - \langle\psi,\phi\rangle^2}{2}$$

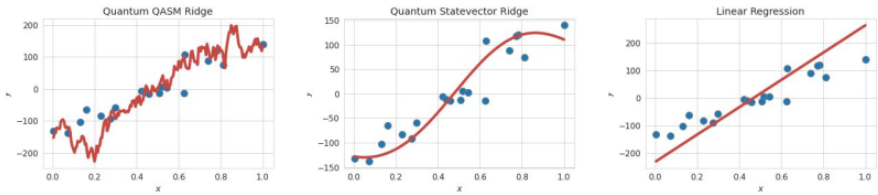
$$P(|0\rangle) = \frac{1 + \langle\psi,\phi\rangle^2}{2}$$



Exemple II

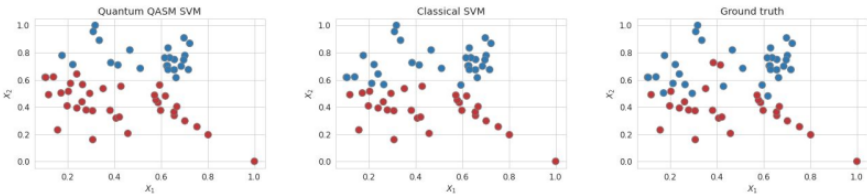
How does it perform? QRidge

$$\alpha = (K + \lambda I)^{-1} y$$



How does it perform? QSVM

$$Ax = b \longrightarrow \begin{bmatrix} 0 & 1_M^T \\ 1_M & K + \gamma^{-1} I_M \end{bmatrix} \begin{bmatrix} \beta \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ Y \end{bmatrix}$$



Where can I learn more?

qlearnkit

```
pip install qlearnkit
```

Qlearnkit contains more...

- Quantum Data Encodings
- Quantum K-NN Regressor
- The project is still maintained
- More algorithms to explore! (PCA, NNs, ...)
- Feel free to contribute!