NOVAGEN
GÉNÉRATEUR D'INNOVATION

Choisir son notebook

Qu'est ce qu'un notebook ?

Outil de prédilection des *Data Scientists,* et dont l'usage se démocratise auprès des *Data Analystes*, le notebook les aide à concevoir des programmes dans l'ordre qui leur semble le plus adapté (programmation lettrée).

Il possède 3 grandes fonctionnalités :

- le mode *Markdown* qui procure les fonctionnalités d'un traitement de texte (Formules LaTeX, Structuration, typographie, hyperliens, etc.) via une syntaxe très simple (langage de balisage léger), pour un rendu en html ;
- le mode *Code*, qui permet d'interpréter des langages, de se connecter à des SGBD, à des moteurs de calcul ou encore à des interfaces système ;
- l'affichage de rendu visuels.

Typiquement un notebook est un client qui s'utilise à partir d'un navigateur internet et qui nécessite le lancement d'un service à partir d'un serveur.

Le notebook n'a pas vocation à servir à développer des programmes complexes, (pour cet usage on préfèrera un IDE), mais se révèle très utile au *Data Scientist* pour explorer les données et comparer les algorithmes.

On pourra également l'utiliser pour la présentation de travaux,
il s'agit d'un outil immédiatement opérationnel et très intuitif.

Jupyter VS Zeppelin

Jupyter et Zeppelin sont actuellement les deux notebooks les plus utilisés,
Qu'est-ce qui distingue l'un de l'autre et quels sont leurs avantages comparatifs ?

Généralités

Version actuelle
sortie le 28 septembre 2016 **2.4.0**

Depuis
fork du projet IPython **2014**

Licence
BSD Modifiée

Version actuelle
sortie le 15 octobre 2016 **0.6.2**

Depuis
2015

Licence
Apache

Partenaires

Sponsor

Google Microsoft

THE APACHE SOFTWARE FOUNDATION

Partenaires institutionnels

Bloomberg Berkeley UNIVERSITY OF CALIFORNIA ANACONDA

Sociétés et institutions qui l'utilisent

NASA O'REILLY SOUNDCLOUD
Google Bloomberg
Microsoft IBM
Berkeley UNIVERSITY OF CALIFORNIA SOCIETE GENERALE
NEW YORK UNIVERSITY

Distributions qui l'intègrent

Hortonworks

Amazon EMR

Communauté

Documentation

Très complète
http://jupyter-notebook.readthedocs.io/en/latest/

Très sommaire
https://zeppelin.apache.org/docs/latest/

Support

Communauté 3 fois plus active
http://stackoverflow.com/questions/tagged/jupyter

Activité relativement faible
http://stackoverflow.com/questions/tagged/apache-zeppelin

Implémentation

Mode service

Très simple
Installer anaconda, et lancer la commande `jupyter notebook` dans un shell.

Très simple,
Télécharger le .tgz sur le site de zeppelin, décompresser, lancer la commande `./zeppelin-daemon.sh start` dans un shell à partir du dossier bin et taper `localhost:8080` dans un navigateur internet

Mode hub

Il est possible d'utiliser Jupyter et Zeppelin en mode Hub au moyen de **JupyterHub** et **ZeppelinHub**. Chaque utilisateur d'une même société pourra ainsi créer son propre compte sur le hub via un portail commun. Ceci lui permettra d'avoir accès à un outil en silo qui hébergera tous ses notebooks.

JupyterHub

Possibilité d'installer Jupyterhub soi-même
(composant d'Anaconda dont les sources sont disponibles)

ZeppelinHub

Licence propriétaire
(solution hérbergée par ZEPL Inc)

Possibilité d'utiliser les notebooks en mode collaboratif
(gestion des droits et des utilisateurs)

Possibilité de publier les notebooks
(en mode public ou privé)

Interopérabilité

- **Interpréteurs et connecteurs ne sont pas natifs**
- **Un notebook ne peut interpréter qu'un seul langage**

Pour lancer les notebooks dans d'autres langages, il est nécessaire d'installer des noyaux (*kernels*) additionnels. La liste des kernels (https://github.com/ipython/ipython/wiki/IPython-kernels-for-other-languages/) comprend, les **Interpréteurs de langage** (liste non exhaustive) :

- **C#**
- **Erlang**
- **ghc**
- **Go**
- **gnuplot**
- **julia**
- **Lua**
- **Matlab**
- **nodejs**
- **Octave**
- **Perl**
- **PHP**
- **Processing**
- **R**
- **ruby**
- **Scala**
- **Scilab**
- **Torch**

Les **connecteurs** :

- SGBD distribuées : Redis
- Moteurs de calculs distribués : **Scala Spark**
- Interface système :
  - **bash**
  - **windows**

- **Interpréteurs et connecteurs natifs**
- **Un notebook peut interpréter plusieurs langages à la fois**

Pour activer tous les interpréteurs et langages disponibles, il suffit de lancer `./install-interpreter.sh -a` depuis le dossier bin.
Les **interpréteurs de langages** disponibles sont :

- **Python**
- **Scala**
- **R**

Les **connecteurs** :

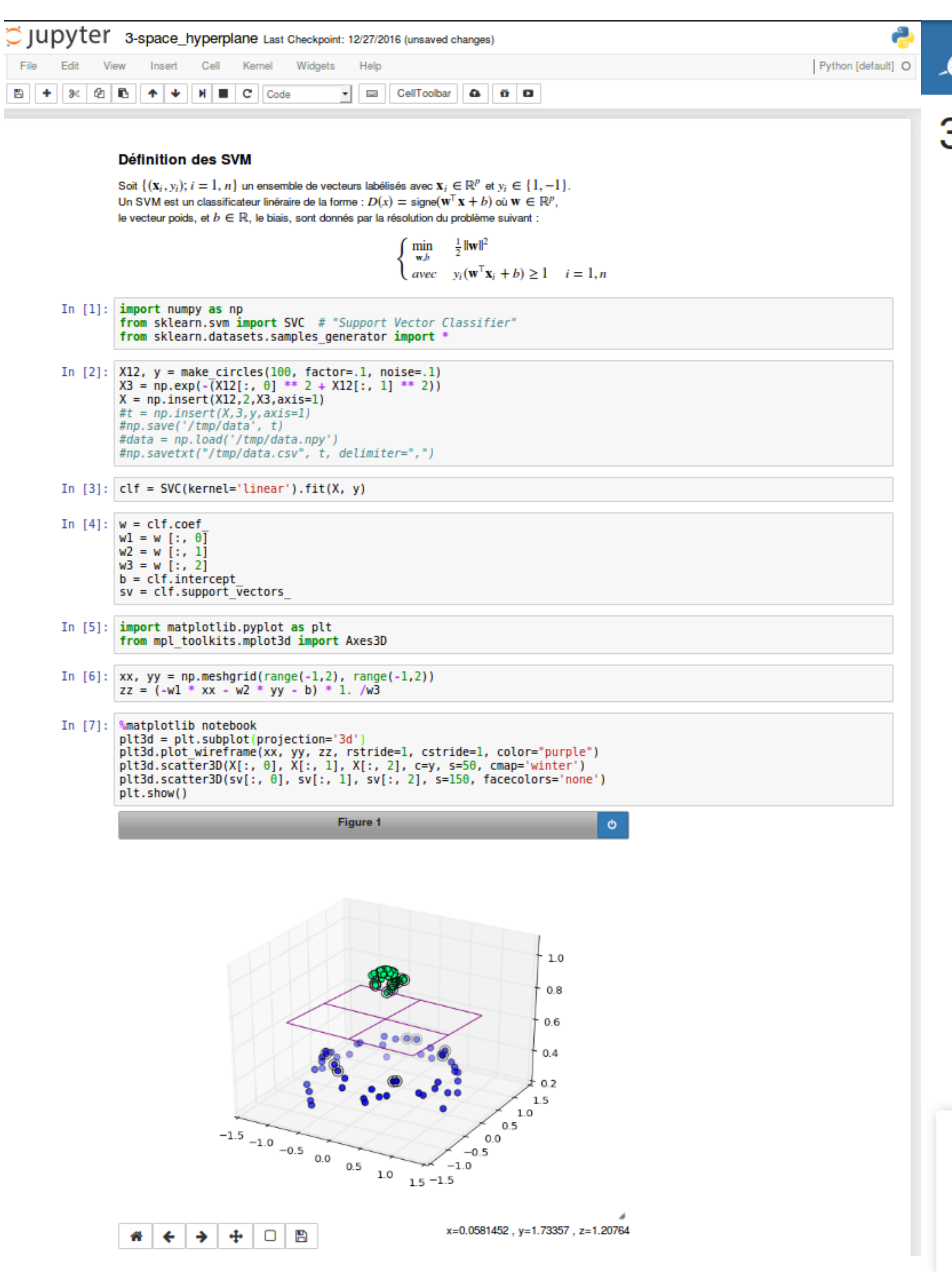
- SGBD distribuées
  - bigquery
  - cassandra
  - Elasticsearch
  - hbase
  - hive
  - Jdbc
  - kylin
  - lens
  - postgresql
- Moteurs de calculs distribués
  - **Spark**
  - Fink
  - Ignite
- Interface système
  - **Shell**
  - Alluxio
  - HDFS file interpreter

Interface

Ergonomie

- Supporte l'**auto-complétion** (TAB)
- Les **graphiques** peuvent s'afficher de manière **interactive ou non interactive** (images) à l'intérieur des *notebook.*
- **Formules LaTeX** gérées par le mode *Markdown*

- **Auto-complétion très limitée** (`Ctl + .`)
- Les **graphiques** s'ouvrent dans une nouvelle fenêtre et sont **interactifs**.
- Chaque **fenêtre** du *notebook* est **dimensionnable**.
- Supporte ***Angular***



jupyter 3-space_hyperplane Last Checkpoint: 12/27/2016 (unsaved changes)

Définition des SVM

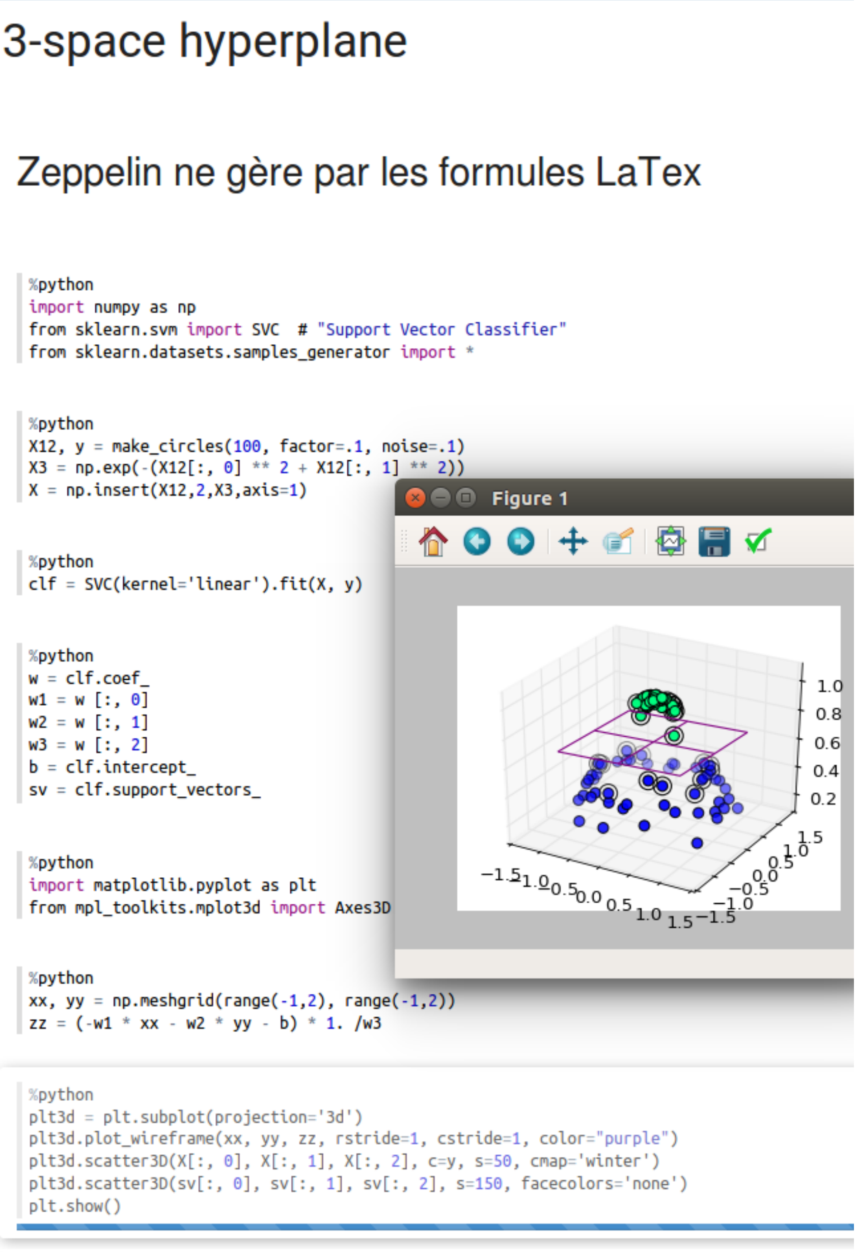
Soit $\{(\mathbf{x}_i, y_i); i = 1, n\}$ un ensemble de vecteurs labélisés avec $\mathbf{x}_i \in \mathbb{R}^p$ et $y_i \in \{1, -1\}$.
Un SVM est un classificateur linéraire de la forme : $D(x) = \text{signe}(\mathbf{w}^\top \mathbf{x} + b)$ où $\mathbf{w} \in \mathbb{R}^p$,
le vecteur poids, et $b \in \mathbb{R}$, le biais, sont donnés par la résolution du problème suivant :

$$\begin{cases} \min\limits_{\mathbf{w},b} & \frac{1}{2}\|\mathbf{w}\|^2 \\ avec & y_i(\mathbf{w}^\top\mathbf{x}_i + b) \geq 1 \quad i = 1, n \end{cases}$$

```
In [1]: import numpy as np
        from sklearn.svm import SVC  # "Support Vector Classifier"
        from sklearn.datasets.samples_generator import *

In [2]: X12, y = make_circles(100, factor=.1, noise=.1)
        X3 = np.exp(-(X12[:, 0] ** 2 + X12[:, 1] ** 2))
        X = np.insert(X12,2,X3,axis=1)
        #t = np.insert(X,3,y,axis=1)
        #np.save('/tmp/data', t)
        #data = np.load('/tmp/data.npy')
        #np.savetxt("/tmp/data.csv", t, delimiter=",")

In [3]: clf = SVC(kernel='linear').fit(X, y)

In [4]: w = clf.coef_
        w1 = w [:, 0]
        w2 = w [:, 1]
        w3 = w [:, 2]
        b = clf.intercept_
        sv = clf.support_vectors_

In [5]: import matplotlib.pyplot as plt
        from mpl_toolkits.mplot3d import Axes3D

In [6]: xx, yy = np.meshgrid(range(-1,2), range(-1,2))
        zz = (-w1 * xx - w2 * yy - b) * 1. /w3

In [7]: %matplotlib notebook
        plt3d = plt.subplot(projection='3d')
        plt3d.plot_wireframe(xx, yy, zz, rstride=1, cstride=1, color="purple")
        plt3d.scatter3D(X[:, 0], X[:, 1], X[:, 2], c=y, s=50, cmap='winter')
        plt3d.scatter3D(sv[:, 0], sv[:, 1], sv[:, 2], s=150, facecolors='none')
        plt.show()
```

Figure 1

x=0.0581452 , y=1.73357 , z=1.20764



Zeppelin

3-space hyperplane

Zeppelin ne gère par les formules LaTex

```
%python
import numpy as np
from sklearn.svm import SVC  # "Support Vector Classifier"
from sklearn.datasets.samples_generator import *

%python
X12, y = make_circles(100, factor=.1, noise=.1)
X3 = np.exp(-(X12[:, 0] ** 2 + X12[:, 1] ** 2))
X = np.insert(X12,2,X3,axis=1)

%python
clf = SVC(kernel='linear').fit(X, y)

%python
w = clf.coef_
w1 = w [:, 0]
w2 = w [:, 1]
w3 = w [:, 2]
b = clf.intercept_
sv = clf.support_vectors_

%python
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

%python
xx, yy = np.meshgrid(range(-1,2), range(-1,2))
zz = (-w1 * xx - w2 * yy - b) * 1. /w3

%python
plt3d = plt.subplot(projection='3d')
plt3d.plot_wireframe(xx, yy, zz, rstride=1, cstride=1, color="purple")
plt3d.scatter3D(X[:, 0], X[:, 1], X[:, 2], c=y, s=50, cmap='winter')
plt3d.scatter3D(sv[:, 0], sv[:, 1], sv[:, 2], s=150, facecolors='none')
plt.show()
```

Figure 1

Formats

Sauvegarde

Ipynb

JSON

Export

- **Notebook (ipynb)**
- **PDF via LaTeX**
- **Presentation**
- **Python**
- **HTML**
- **Markdown**

JSON

Récapitulatif

Avantages comparatifs

Projet **plus mature**
Plus utilisé
Documentation plus complète
Beaucoup **plus d'interpréteurs** de langages
Plus de formats d'export
Auto-completion
JupyterHub est une solution **open source** qu'on peut installer soi-même.
Formules LaTeX

Plus de connecteurs
Possibilité d'interpréter **plusieurs Langages dans un même notebook**
Chaque **fenêtre** du notebook est **dimensionnable**
Possibilité d'avoir des **graphiques dynamiques** avec Angular
ZeppelinHub permet d'avoir des **notebooks collaboratifs et publiables**

Verdict

Data science

Data Engineering