TensorFlow

GDG Lille
28/06/2016
Manuel Verriez

Définition ?

“TensorFlow™ is an open source software library for numerical computation using data flow graphs.”

https://www.tensorflow.org

Objectif(s)?

"TensorFlow was originally developed for the purposes of conducting ***machine learning*** and ***deep neural networks*** research"

https://www.tensorflow.org

Machine Learning

Un peu de théorie...

Field of study that gives computers

the ability to learn without

being explicitly programmed

Arthur Samuel 1959

Deep Learning Isn’t a Dangerous Magic Genie. It’s Just Math.

Oren Etzioni 2016

http://www.wired.com/2016/06/deep-learning-isnt-dangerous-magic-genie-just-math

Apprentissages

Supervisés

On cherche une réponse à une question bien précise, en fonction de réponses passées

Non supervisés

On cherche à créer des groupes de données similaires

Par renforcement

On cherche à apprendre via l' expérience, ce qu' il convient de faire en différentes situations

Apprentissages

Supervisés

- Estimation prix de vente
- Classification de photos (catégories connues)
- Prévision des stocks
- Détection de fraude
- …

Non supervisés

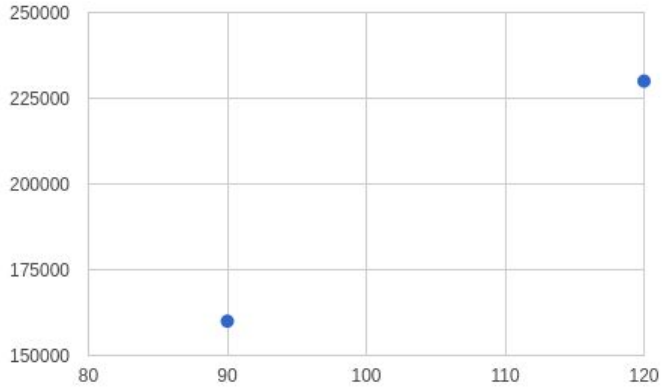
- Classification de photos (catégories non connues)
- Regroupement d'articles de presse
- Détection de fraude
- …

Par renforcement

- Jeu de go !
- Contrôle de robots
- …

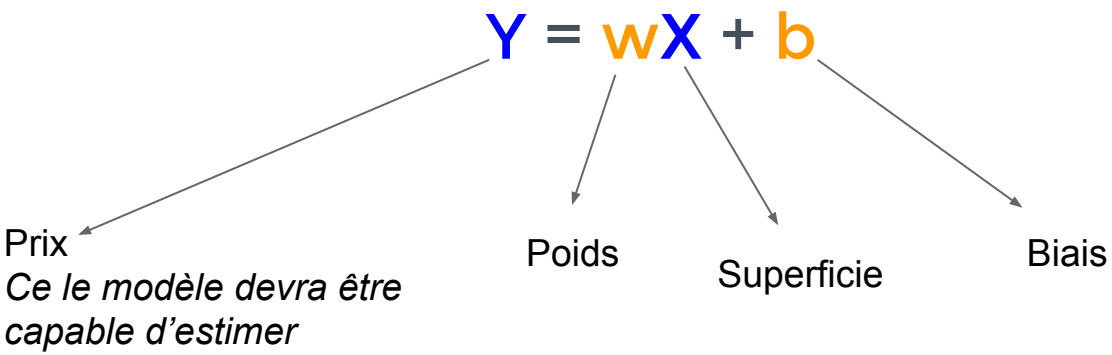
Apprentissage supervisé

Superficie (m2)	Prix de vente (€)
90	**160 000**
120	**230 000**

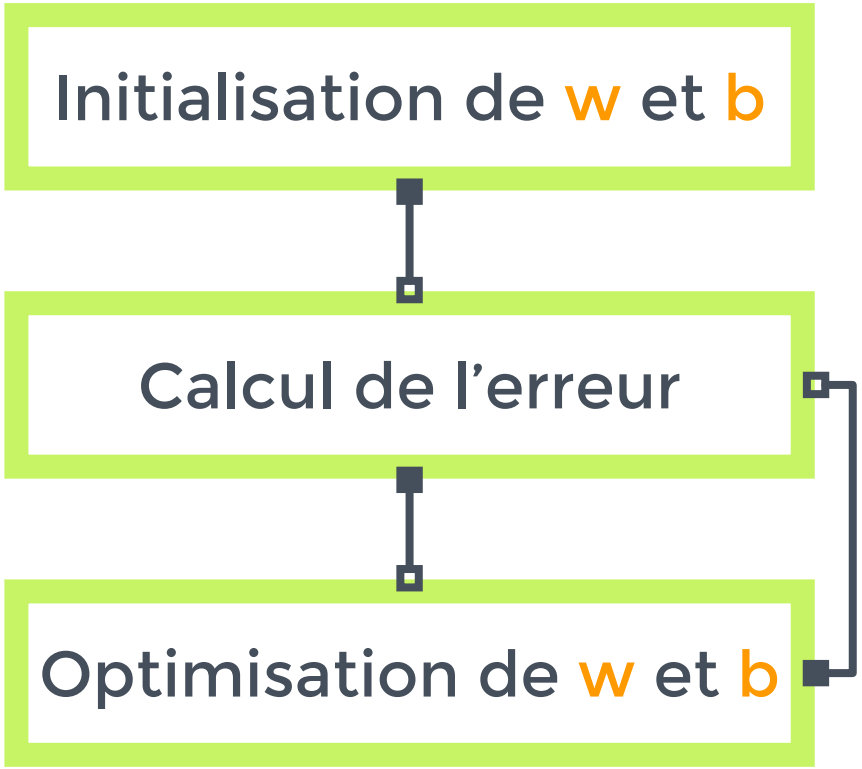


Apprentissage supervisé

Régression linéaire :



Apprentissage supervisé



Apprentissage supervisé

Calcul de l'erreur

- Mean squared error
- Cross-Entropy
- Hellinger distance
- Itakura-Saito
- ...

Optimisation

- Gradient descent
- Adadelta
- Adagrad
- Momentum
- Adam
- FTRL
- RMS prop
- ...

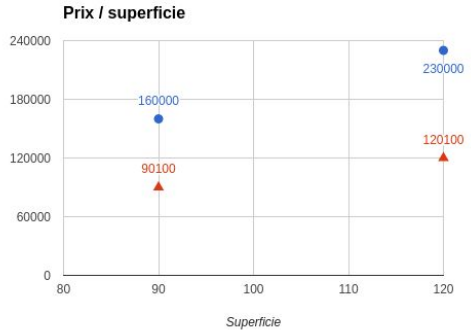
Régression linéaire

Initialisation : w = 1000; b=100

Formule MSE : $$Error_{(w,b)} = \frac{1}{N}\sum_{i=1}^{N}(y_i - (w * x_i + b))^2$$

Première itération :

*Error = ((160 000 - (1 000 * 90))² + (230 000 - (1 000 * 120))²)/2 =* ***8 482 010 000***



Régression linéaire

Algorithme Gradient descent :

- Calcul des dérivées partielles (exemple avec w et b):

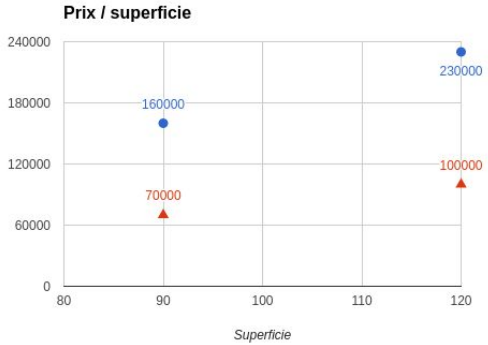
$$\frac{\partial}{\partial b}J(w,b) = \frac{1}{m}\sum_{i=1}^{m}(b + wx^{(i)} - y^{(i)})$$

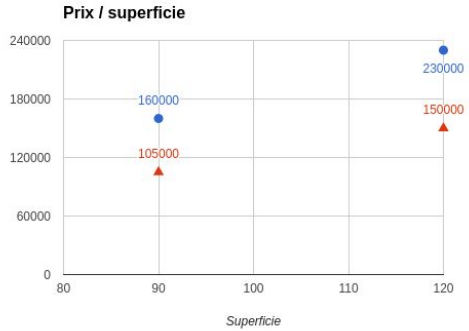
$$\frac{\partial}{\partial w}J(w,b) = \frac{1}{m}\sum_{i=1}^{m}(b + wx^{(i)} - y^{(i)})x^{(i)}$$

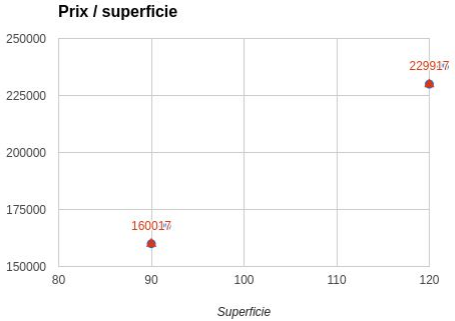
- On répète (avec θ = (w,b)) : $\theta := \theta - \alpha\frac{\delta}{\delta\theta}J(\theta)$
- On arrête quand la réduction d'erreur n'est plus significative

Régression linéaire

Après plusieurs itérations







Régression linéaire

```
sizes=[90,120]
prices=[160000, 230000]

#Calculate cost
def cost(X,y,w,b):
    return sum([( w*X[i] + b - y[i])**2  for i in range(m)]) / (2*m)

#Update weights
def update_weights(X,y,w,b):
    m=len(X)
    grad_b = (1.0/m) * sum([(b + w*X[i] - y[i])for i in range(m)])
    grad_w = (1.0/m) * sum([(b + w*X[i] - y[i])*X[i] for i in range(m)])
    w = w - alpha * grad_w
    b = b - alpha * grad_b
    return (w,b)

#Train
def train(X,y):
    w = 0.5
    b = 0.5
    alpha = 0.0001
    j = cost(X,y,w,b)
    while(j>1000):
        w,b = update_weights(X,y,w,b)
        j = cost(X,y,w,b)
    print "Weight=%f"%w
    print "Bias=%f"%b
    return (w,b)

#Predict
def predict(size):
    return (size*w+b)

def pretty_print(size):
    print "For size=%d, price=%f"% (size, predict(size) )

(w,b) = train(sizes, prices)

pretty_print(120)
pretty_print(150)
pretty_print(100)
```

Régression linéaire

Mais dans la réalité:

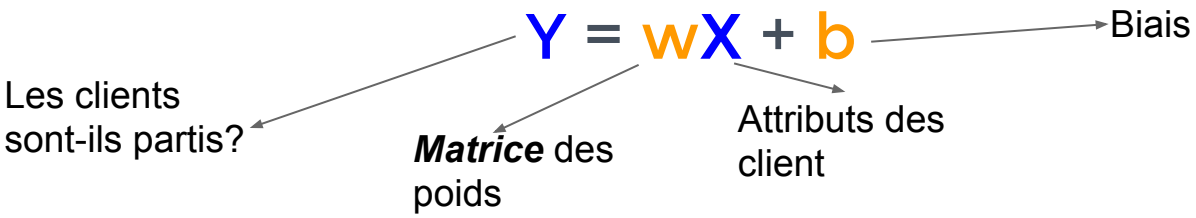
- d'autres critères nombre de pièces, distance/école, proximité commerces, etc.
- nombre de données important

Superficie (m2)	Nombre de pièces	Distance centre-ville	Prix de vente (€)
90	4	800	**160 000**
120	5	900	**230 000**

Apprentissage supervisé

- Exemple de la prédiction du churn (abonné opérateur téléphonique):

```
KS, 128, 415, 382-4657, no, yes, 25, 265.1, 110, 45.07, 197.4, 99, 16.78, 244.7, 91, 11.01, 10, 3, 2.7, 1, False.
OH, 107, 415, 371-7191, no, yes, 26, 161.6, 123, 27.47, 195.5, 103, 16.62, 254.4, 103, 11.45, 13.7, 3, 3.7, 1, False.
NJ, 137, 415, 358-1921, no, no, 0, 243.4, 114, 41.38, 121.2, 110, 10.3, 162.6, 104, 7.32, 12.2, 5, 3.29, 0, False.
OH, 84, 408, 375-9999, yes, no, 0, 299.4, 71, 50.9, 61.9, 88, 5.26, 196.9, 89, 8.86, 6.6, 7, 1.78, 2, False.
OK, 75, 415, 330-6626, yes, no, 0, 166.7, 113, 28.34, 148.3, 122, 12.61, 186.9, 121, 8.41, 10.1, 3, 2.73, 3, False.
AL, 118, 510, 391-8027, yes, no, 0, 223.4, 98, 37.98, 220.6, 101, 18.75, 203.9, 118, 9.18, 6.3, 6, 1.7, 0, False.
...
```



TensorFlow

Let’s go!

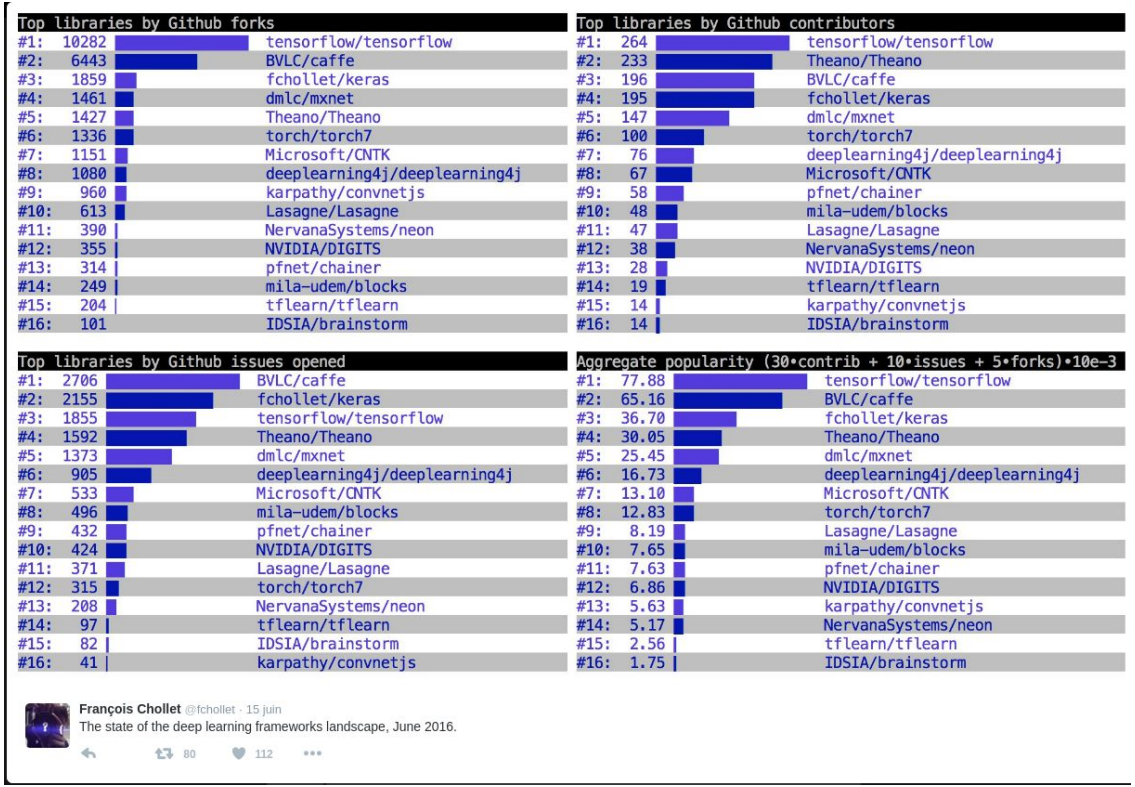
Une histoire récente

- Développé par l'équipe "Google Brain" depuis 2011
- Open source depuis 11/2015
- Utilisé dans "Google photo", "Google voice search", "Google Translate",...

Une histoire récente

- Nov. 2015 : Version pour CPU's, GPU's
- Fév. 2016 : Version distribuée (gRPC)
- Mai 2016 : Google annonce un circuit intégré (ASIC : processeur(s), interfaces, mémoires) conçu pour TensorFlow
- Plateformes:
  - Linux & Mac OS (64 bits)
  - Android / iOS

Une histoire récente



Top libraries by Github forks

#1:	10282	tensorflow/tensorflow
#2:	6443	BVLC/caffe
#3:	1859	fchollet/keras
#4:	1461	dmlc/mxnet
#5:	1427	Theano/Theano
#6:	1336	torch/torch7
#7:	1151	Microsoft/CNTK
#8:	1080	deeplearning4j/deeplearning4j
#9:	960	karpathy/convnetjs
#10:	613	Lasagne/Lasagne
#11:	390	NervanaSystems/neon
#12:	355	NVIDIA/DIGITS
#13:	314	pfnet/chainer
#14:	249	mila-udem/blocks
#15:	204	tflearn/tflearn
#16:	101	IDSIA/brainstorm

Top libraries by Github contributors

#1:	264	tensorflow/tensorflow
#2:	233	Theano/Theano
#3:	196	BVLC/caffe
#4:	195	fchollet/keras
#5:	147	dmlc/mxnet
#6:	100	torch/torch7
#7:	76	deeplearning4j/deeplearning4j
#8:	67	Microsoft/CNTK
#9:	58	pfnet/chainer
#10:	48	mila-udem/blocks
#11:	47	Lasagne/Lasagne
#12:	38	NervanaSystems/neon
#13:	28	NVIDIA/DIGITS
#14:	19	tflearn/tflearn
#15:	14	karpathy/convnetjs
#16:	14	IDSIA/brainstorm

Top libraries by Github issues opened

#1:	2706	BVLC/caffe
#2:	2155	fchollet/keras
#3:	1855	tensorflow/tensorflow
#4:	1592	Theano/Theano
#5:	1373	dmlc/mxnet
#6:	905	deeplearning4j/deeplearning4j
#7:	533	Microsoft/CNTK
#8:	496	mila-udem/blocks
#9:	432	pfnet/chainer
#10:	424	NVIDIA/DIGITS
#11:	371	Lasagne/Lasagne
#12:	315	torch/torch7
#13:	208	NervanaSystems/neon
#14:	97	tflearn/tflearn
#15:	82	IDSIA/brainstorm
#16:	41	karpathy/convnetjs

Aggregate popularity (30•contrib + 10•issues + 5•forks)•10e-3

#1:	77.88	tensorflow/tensorflow
#2:	65.16	BVLC/caffe
#3:	36.70	fchollet/keras
#4:	30.05	Theano/Theano
#5:	25.45	dmlc/mxnet
#6:	16.73	deeplearning4j/deeplearning4j
#7:	13.10	Microsoft/CNTK
#8:	12.83	torch/torch7
#9:	8.19	Lasagne/Lasagne
#10:	7.65	mila-udem/blocks
#11:	7.63	pfnet/chainer
#12:	6.86	NVIDIA/DIGITS
#13:	5.63	karpathy/convnetjs
#14:	5.17	NervanaSystems/neon
#15:	2.56	tflearn/tflearn
#16:	1.75	IDSIA/brainstorm

François Chollet @fchollet · 15 juin
The state of the deep learning frameworks landscape, June 2016.

80 112 •••

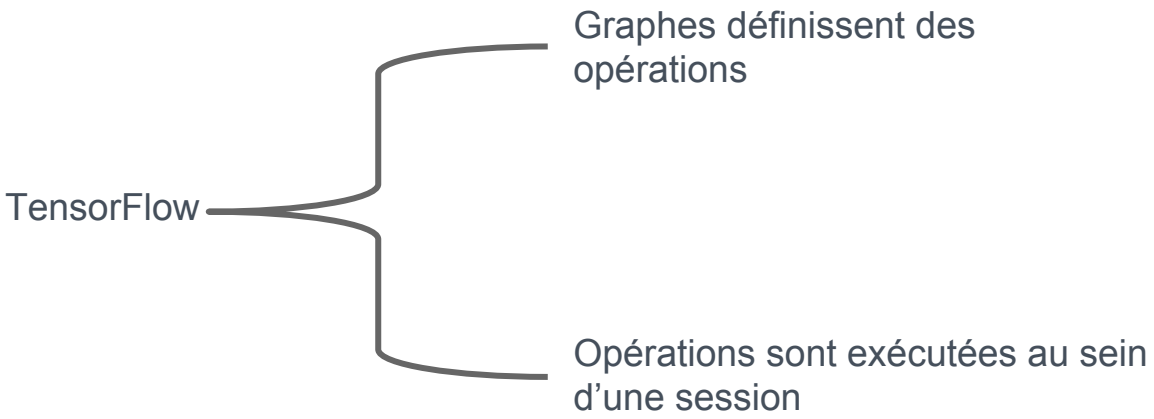
https://twitter.com/fchollet/status/743116684026777601

import tensorflow as tf

Python 2.7 / 3.3

on aurait pu aussi le faire en c++

API Python


TensorFlow
Graphes définissent des opérations
Opérations sont exécutées au sein d'une session

Notebook

In [ ]:
```
import tensorflow as tf
```

In [ ]:
```
tf.reset_default_graph()
graph = tf.get_default_graph()
```

In [ ]:
```
graph
```

In [ ]:
```
graph.get_operations()
```

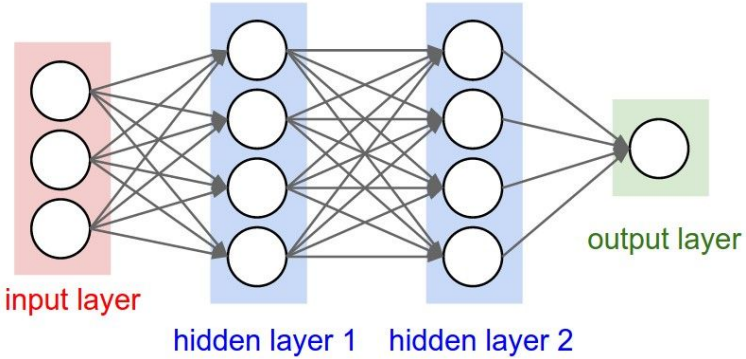
In [ ]:
```
input_value = tf.constant(1.0, name="first_constant")
```

In [ ]:
```
input_value
```

In [ ]:
```
graph.get_operations()
```

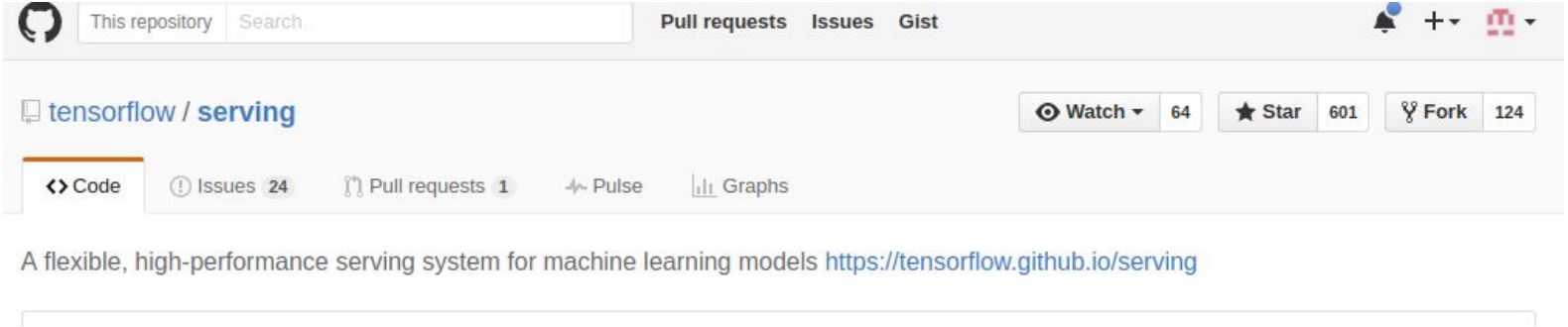
In [ ]:
```
for op in graph.get_operations():
    print op.node_def
```

Réseaux de neurones



Tensorflow serving

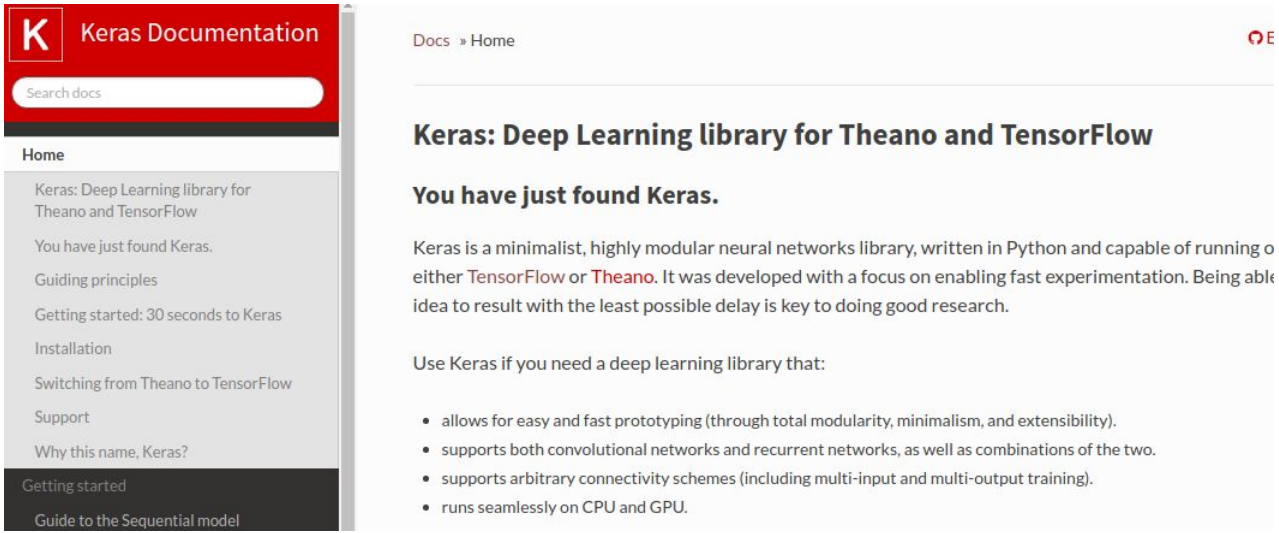
- Publier le modèle dans un service gRPC



- Fonctionne, mais pas encore industrialisé

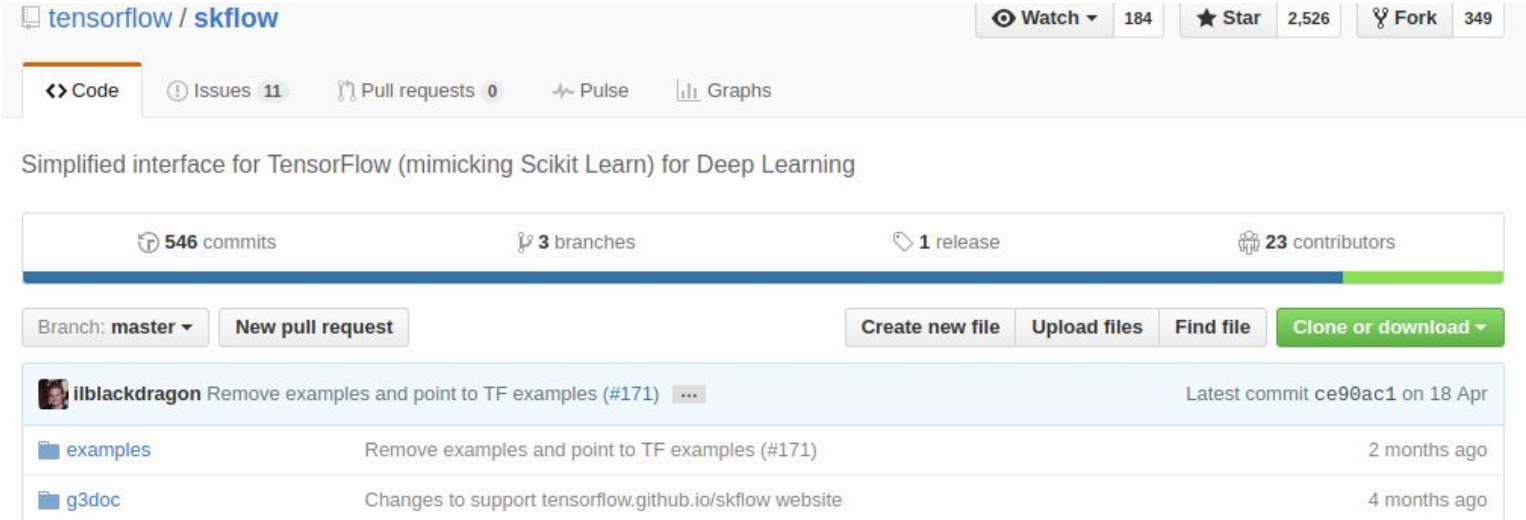
Ecosystème

keras



Ecosystème

skflow



Exemples

- Reinforcement learning

https://github.com/coreylynch/async-rl

Exemples

- Prédiction demande
  - https://cloud.google.com/blog/big-data/2016/05/how-to-forecast-demand-with-google-bigquery-public-datasets-and-tensorflow

Exemples

Classification image: http://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

Distribution

- Avec tensorflow
- Avec spark (https://github.com/databricks/tensorframes)

Extensibilité

https://www.tensorflow.org/versions/r0.9/how_tos/adding_an_op/index.html#adding-a-new-op

Ressources utiles

https://fr.coursera.org/learn/machine-learning

https://www.udacity.com/course/deep-learning--ud730

http://www.jorditorres.org/first-contact-with-tensorflow/

https://www.oreilly.com/learning/hello-tensorflow

Merci.