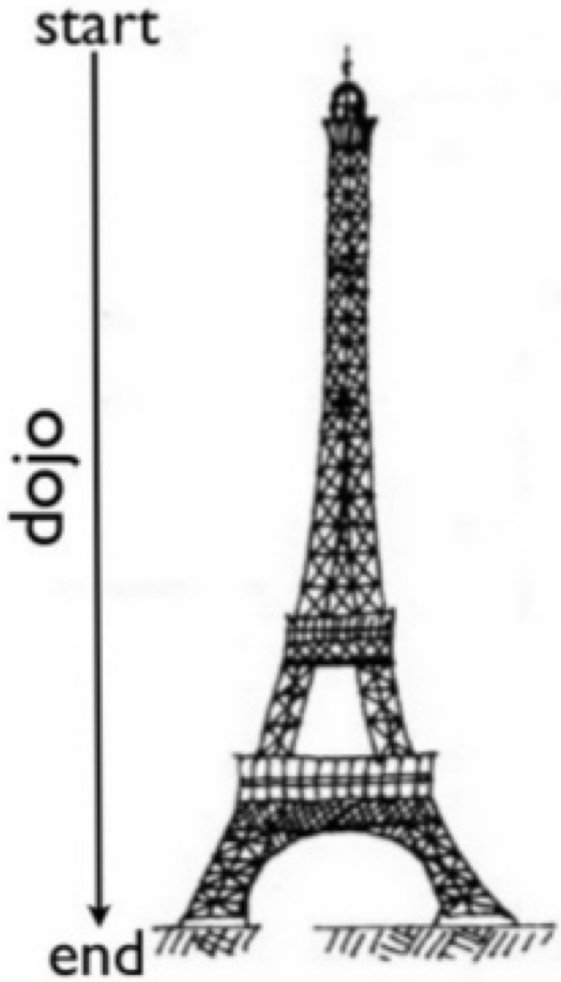


2mins: organise next dojo

30mins: retrospective on last dojo

10mins: decide the topic for this dojo

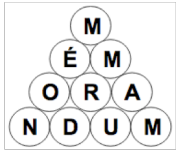
40mins: code (either “prepared” or “randori” kata)

10mins: half time break

40mins: code

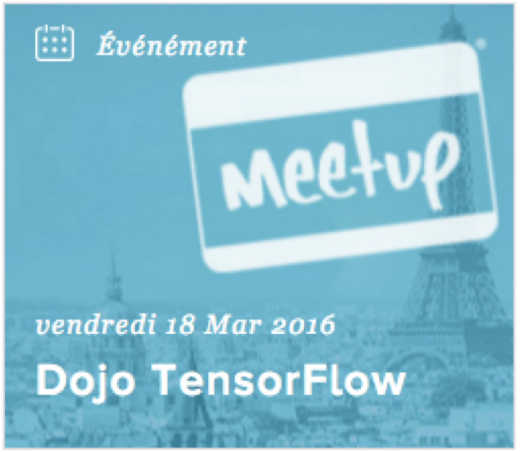
Dojo Tensorflow #1

Chez Sfeir



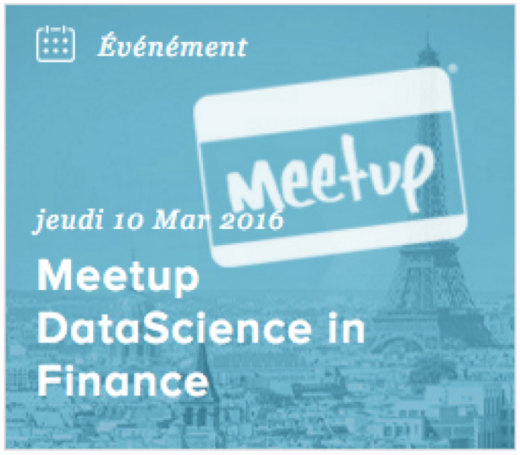
[sfeir] ACTUALITÉS OFFRES LE GROUPE NOUS REJOINDRE CONTACT LE MAG

Le développeur au coeur du digital



@sfeir
Marre de stagner? Envie de vous tester sur de nouveaux challenges techniques? Venez passer les Playoffs!
#SFEIRCode
https://t.co/7xJBzIwRwY

09:05 - 14 Mar 2016



@sfeir

Félicitations @cbalit !!!
Nouveau #gde chez les Sfeiriens !
https://t.co/h6moPbLMIm

14:47 - 11 Mar 2016

Mémorandum

Mémorandum est un cabinet de **conseil en data stratégie**.

Nous intervenons en trois phases :

1. Réflexion sur l'usage de la donnée dans votre entreprise
2. Analyse de vos données
3. Industrialisation de solutions informatiques

Nous apportons :

- ➔ Une méthodologie mélant stratégie et technique.
- ➔ Des preuves de concepts "machine learning" avec les outils en pointe de la communauté open source
- ➔ Des méthodes agiles et de Lean Analytics qui garantissent des résultats adaptés

Romain Jouin - Associé
INT Management 2006 - Télécom Paris 2013
7 ans de commercial
25 ans d'informatique

Denis Oblin - Associé
Centrale 1994 - Télécom Paris 2013
10 ans de conseil en stratégie
7 ans en direction opérationnelle Groupama

Mémorandum a trois expertises majeures :

- ➔ Technique
  - ◆ Big Data
  - ◆ Machine Learning
- ➔ Fonctionnelle
  - ◆ Stratégie de la micro décision
  - ◆ Marketing
- ➔ Métier
  - ◆ Relation client
  - ◆ Force de vente

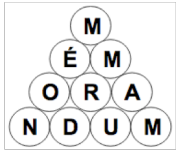
- Installation

Lesson 1

Lesson 2 (to do)

Lesson 3 (to do)

Lesson 4 (to do)



UDACITY

Deep Learning

Take machine learning to the next level

Advanced

Approx. 3 months

Assumes 6hrs/wk (work at your own pace)

Built by Google

Join 61,607 students

MÉMORANDUM

Prerequisites and Requirements

This is an intermediate to advanced level course. Prior to taking this course, and in addition to the prerequisites and requirements outlined for the Machine Learning Engineer Nanodegree program, you should possess the following experience and skills:

- Minimum 2 years of programming experience (preferably in Python)
- Git and GitHub experience (assignment code is in a GitHub repo)
- Basic machine learning knowledge (especially supervised learning)
- Basic statistics knowledge (mean, variance, standard deviation, etc.)
- Linear algebra (vectors, matrices, etc.)
- Calculus (differentiation, integration, partial derivatives, etc.)

scipy, numpy, scikit learn...

Syllabus

Lesson 1: From Machine Learning to Deep Learning

- Understand the historical context and motivation for Deep Learning.
- Set up a basic supervised classification task and train a black box classifier on it.
- Train a logistic classifier "by hand"Optimize a logistic classifier using gradient descent, SGD, Momentum and AdaGrad.

Lesson 2: Deep Neural Networks

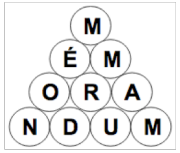
- Train a simple deep network.
- Effectively regularize a simple deep network.
- Train a competitive deep network via model exploration and hyperparameter tuning.

Lesson 3: Convolutional Neural Networks

- Train a simple convolutional neural net.
- Explore the design space for convolutional nets.

Lesson 4: Deep Models for Text and Sequences

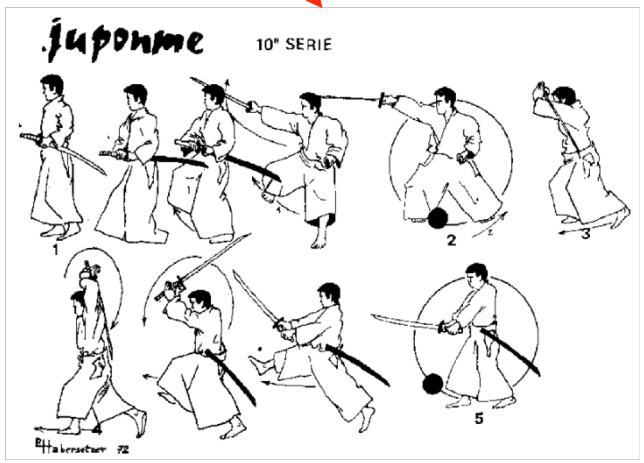
- Train a text embedding model.
- Train a LSTM model.



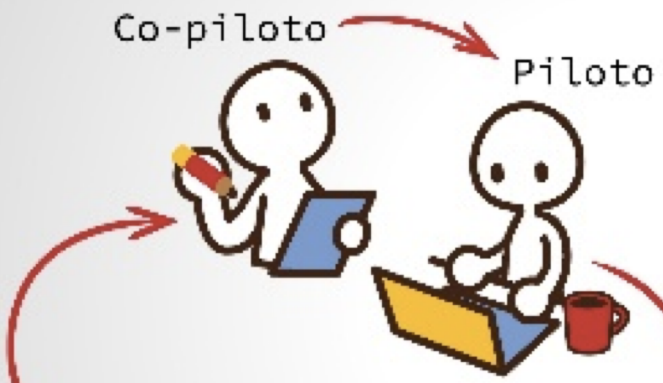
CODING DOJO

Randori

Kata



Dojo Randori

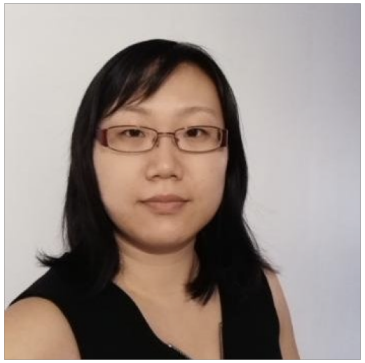



```
private int arma;

private int local;

public Problema(int assassino, int arma, int local) {
    this.assassino = assassino;
    this.arma = arma;
    this.local = local;

}
```

Ce soir 4 intervenants

Alphabet

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



Experte Tensorflow

Nous expliquera les matsh

Animera la soirée

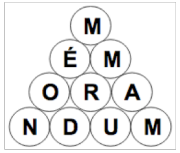
Docker propose un docker file pour lancer l'environnement

docker run -p 8888:8888 -it --rm b.gcr.io/tensorflow-udacity/assignments:0.4.0

https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/udacity/README.md

Ou bien nous avons notre propre dockerfile

```
FROM b.gcr.io/tensorflow/tensorflow
RUN apt-get update
RUN apt-get install -y  gfortran python-pip python-dev git mc vim
RUN apt-get install -y python-numpy python-scipy libatlas-base-dev
RUN pip install --upgrade pip
RUN pip install -U pandas
RUN easy_install scipy
RUN pip install --upgrade scipy
RUN pip install -U scikit-learn
RUN pip install git+git://github.com/tensorflow/skflow.git
RUN pip install --upgrade --user ipython
RUN mkdir /notebook
VOLUME /notebook
EXPOSE 8888 6006
CMD jupyter notebook /notebook
```



Environnement pour la soirée

python™

TensorFlow

THE MNIST DATABASE

of handwritten digits

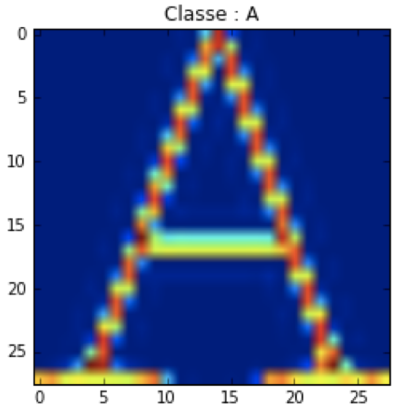
Yann LeCun, Courant Institute, NYU
Corinna Cortes, Google Labs, New York
Christopher J.C. Burges, Microsoft Research, Redmond

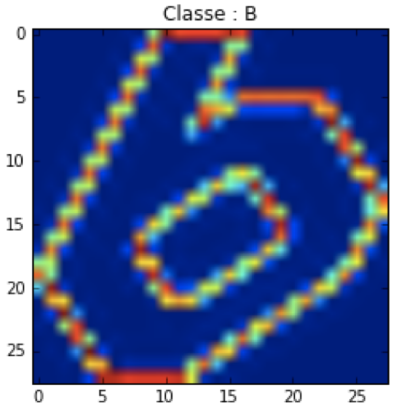
The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

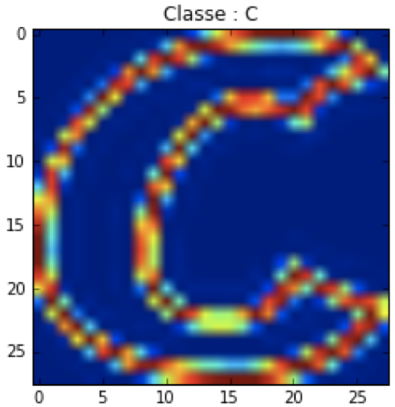
http://yann.lecun.com/exdb/mnist/

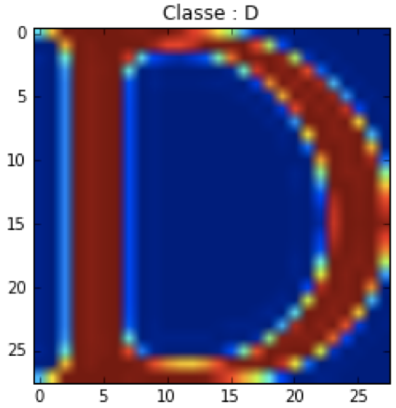
MÉMORANDUM

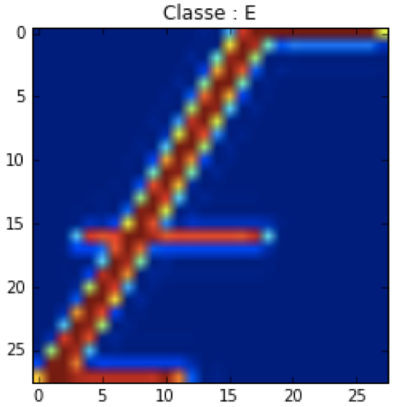
MNIST Database

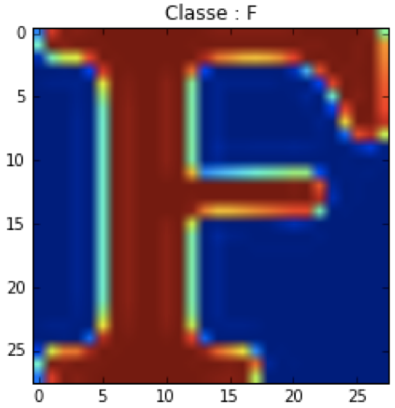




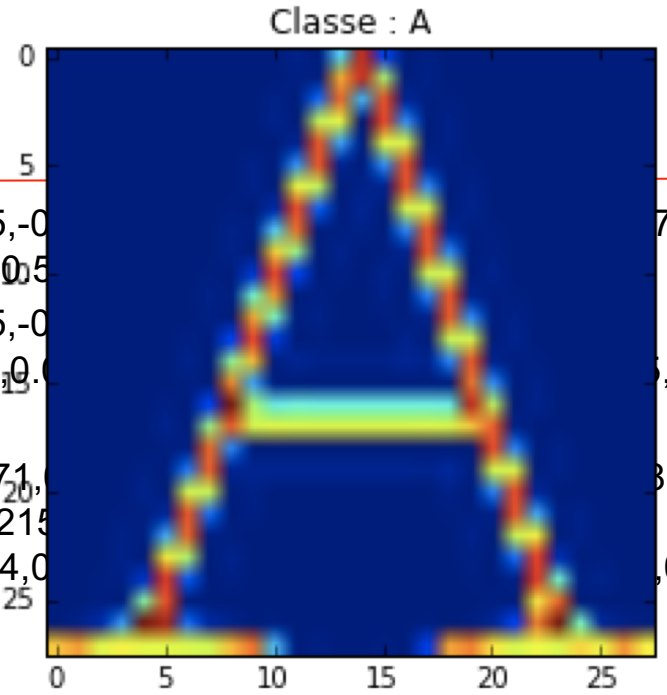








dataset.shape = (52909, 28, 28)
dataset[0].shape = (28, 28)



array(
[
[-0.5,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5,-0 ... 7451,0.4254902,-0.36274511,-0.5,-
0.49215686,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5
[-0.5,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5,-0
0.49607843,0.2254902,0.42156863,0.0 ... ,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5],
...
[0.19411765,0.24901961,0.11176471,0 ... 3,0.10392157,0.0882353,0.20980392,
0.30392158,-0.19019608,-0.5,-0.49215
0.40196079,0.23333333,0.28431374,0 ... ,0.10392157,0.18235295,0.13137256,
0.26078433,0.14705883]
],
dtype=float32)

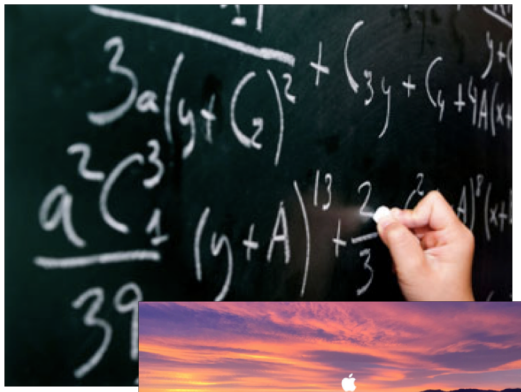
Installation

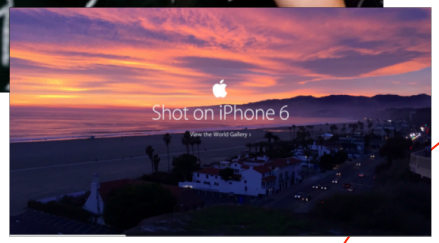
Lesson 1 – classification logistique et descente de gradient

Lesson 2 - Réseau de neurone profond (to do)

Lesson 3 - Réseau de convolution (to do)

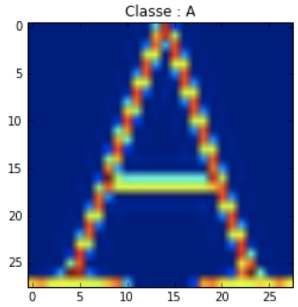
Lesson 4 – Réseau de neurone profond pour l’analyse de texte et de séquences (to do)



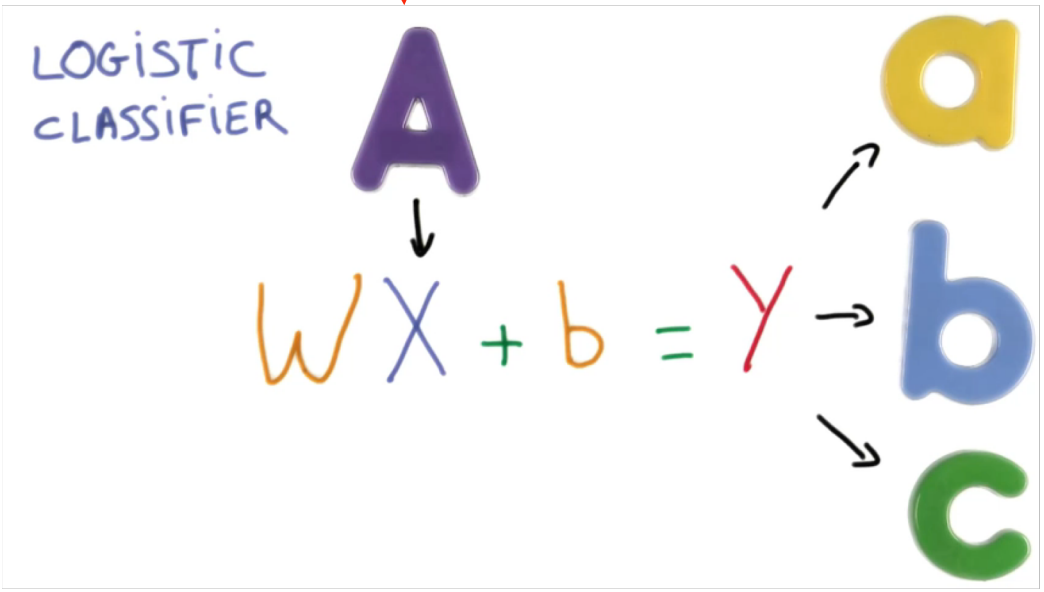


12 millions de paramètres

Logistic classifier et softmax



28 * 28 = 756



randori #1

- Objectif : implémenter la fonction softmax
- Durée prévue : 10 minutes max
- Nombre de codeurs : 1 à 2
- Nombre de lignes dans la réponse : 1 fonction de 3 lignes

Practical issues of softmax: Numeric stability

$$\frac{e^{f_{y_i}}}{\sum_j e^{f_j}} = \frac{Ce^{f_{y_i}}}{C\sum_j e^{f_j}} = \frac{e^{f_{y_i}+\log C}}{\sum_j e^{f_j+\log C}}$$

$$\log C = -\max_j f_j$$

We are free to choose the value of *C.*

Practical issues: Numeric stability

```
f = np.array([123, 456, 789]) # example with 3 classes and each having large scores
p = np.exp(f) / np.sum(np.exp(f)) # Bad: Numeric problem, potential blowup

# instead: first shift the values of f so that the highest number is 0:
f -= np.max(f) # f becomes [-666, -333, 0]
p = np.exp(f) / np.sum(np.exp(f)) # safe to do, gives the correct answer
```

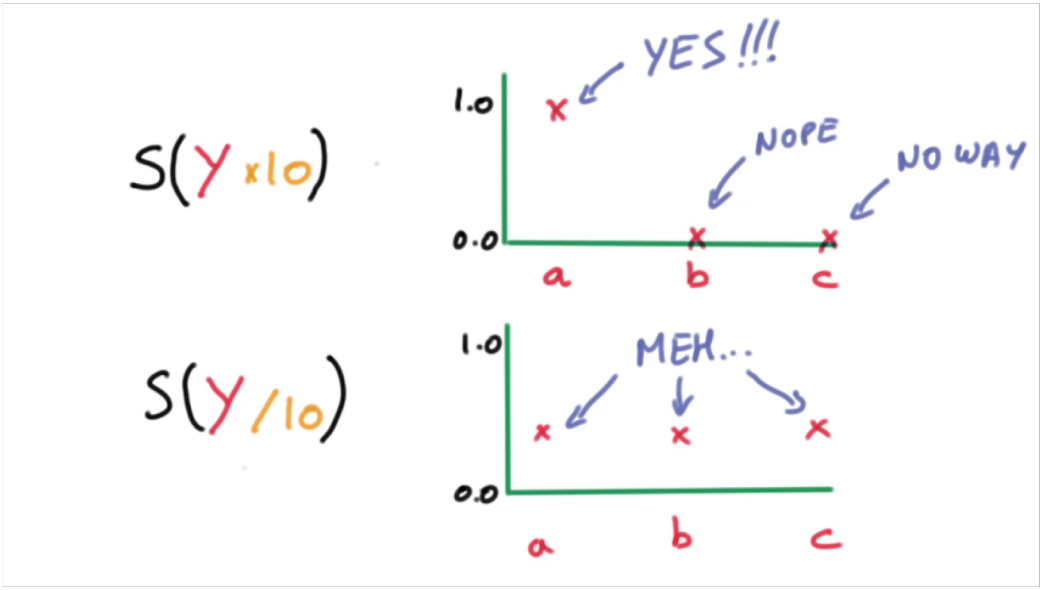
One hot encoding

Pour prendre une décision binaire sur la classification, on peut transformer les probabilités en valeurs "0" ou "1". C'est le "One hot encoding".

Il faut alors un vecteur par classe, qui contiendra plein de zéros, et une seule valeur "1".

Ainsi chaque classe a son vecteur signature qui lui permet d'être reconnaissable.

Le problème du one-hot encoding c'est que ça créé des matrices très "creuses" s'il y a beaucoup de classes : on a quasiment que des zéros, pour un seul "1" par colonne.



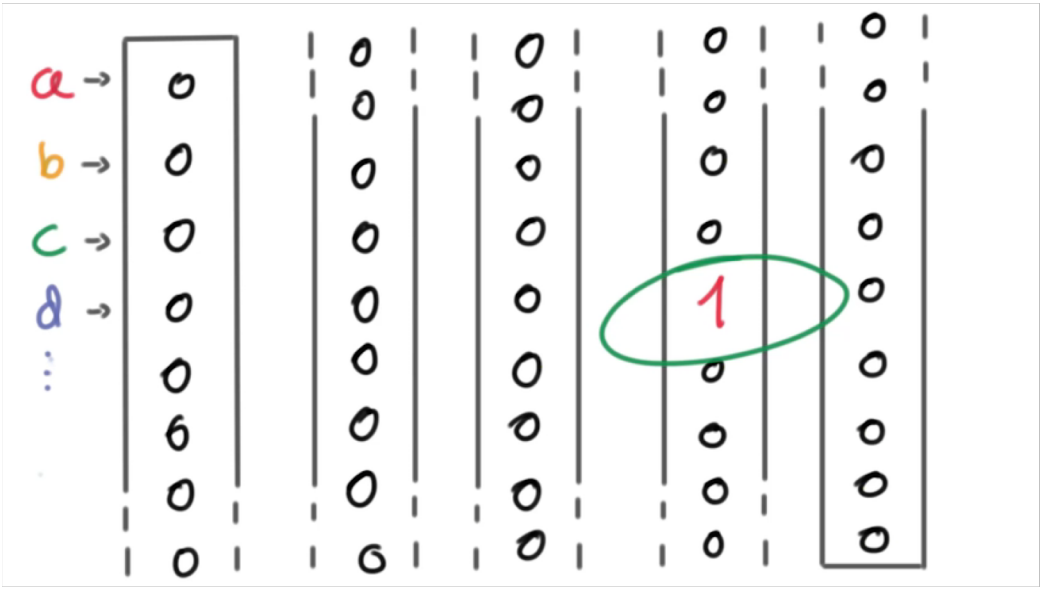
Cross entropy

Cette fonction permet de transformer l'ensemble des probabilités en un vecteur binaire avec un seul "1", qui sera la classe élue.

On a donc :

input => modèle => score => softmax =>probabilité => cross-entropy => classe choisie.

C'est la classification logistique multinomiale.



Schotastic gradient descent

Comment trouver la matrice des poids [W] et la matrice de biais [b] qui nous permettent d'atteindre notre objectif, à savoir :

1) d'avoir une petite distance aux classes correctes

2) d'avoir une grande distances aux classes incorrectes

En apprentissage supervisé on peut calculer la distance entre notre classification logistique (multinomiale) et les classes connues. La différence est l'erreur, appelée en anglais "loss".

On définit donc une fonction de mesure de l'erreur ; la "loss function", que l'on définit selon les cas comme l'on veut.

Par exemple ici on peut la définir comme étant une moyenne des erreurs de distance :

loss = sum (cross_entropy (classification logistique, valeur réelle)) / nombre_de_lignes_connues

loss = 1/n * SUM[i](D(S(W*Xi + b), Li))

=> W est la matrice des poids

=> Xi est une ligne connue, avec Li la réponse connue (le label / la classe)

=> D est la fonction de cross-entropy qui va convertir les score en proba

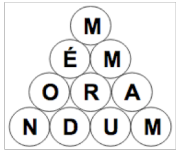
C'est à dire que la loss function ici est la moyenne des erreurs.

Pour réduire la cross entropy on va essayer de changer W de manière à réduire la fonction de perte. Comment faire ? on essaie en changeant les valeurs dans W ;)

Mais il y a des manières intelligentes de changer W : on peut utiliser la méthode de la "descente de gradient".

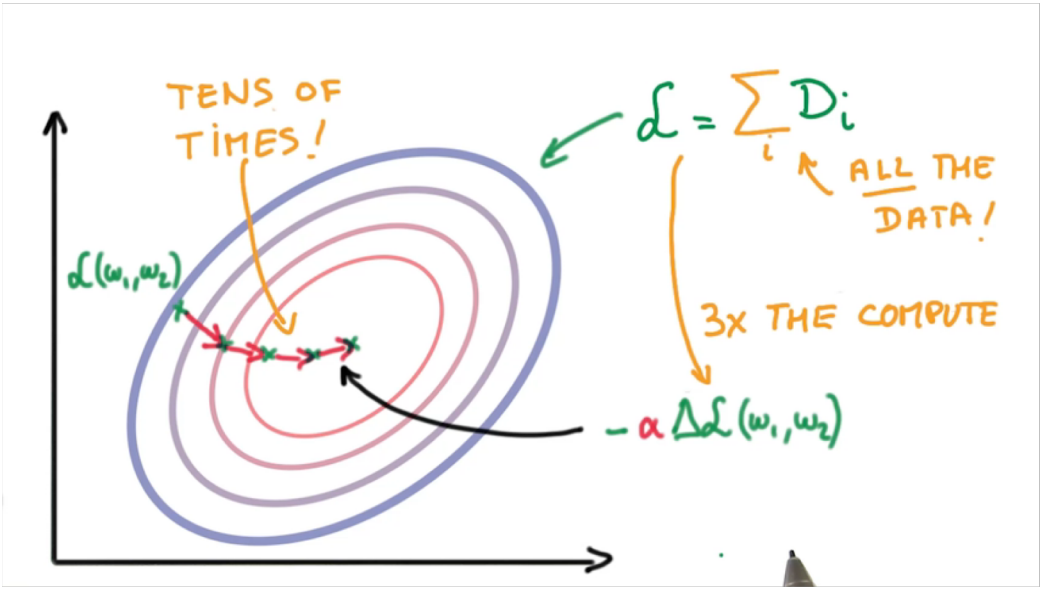
La descente de gradient fonctionne un peu comme quelqu'un sur une coline qui voudrait descendre en bas : il pose son pied à plusieurs endroits et avance là où il sent que la coline descend.

En termes plus mathématiques, on calcule la dérivée de la fonction de perte en accord avec les paramètres, et on avance d'un pas dans cette direction.



Schotastic gradient descent (vidéo 36)

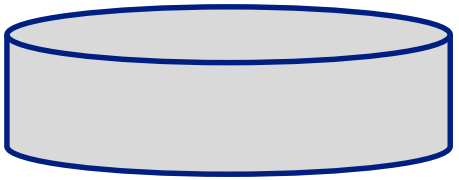
Comment trouver la matrice des poids [W] et la matrice de biais [b] qui nous permettent d'atteindre notre objectif, à savoir :
1) d'avoir une petite distance aux classes correctes
2) d'avoir une grande distances aux classes incorrectes



Jeu de donnée utilisé pendant le MOOC

NOT

Téléchargement
10 classes : A -> J



training set : 500k images
train_datasets

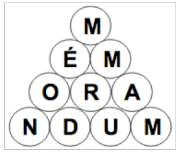
```
notMNIST_large/A.pickle
notMNIST_large/B.pickle
notMNIST_large/C.pickle
notMNIST_large/D.pickle
notMNIST_large/E.pickle
notMNIST_large/F.pickle
notMNIST_large/G.pickle
notMNIST_large/H.pickle
notMNIST_large/I.pickle
notMNIST_large/J.pickle
```

Test set : 19k images
test_datasets

```
notMNIST_small/A.pickle
notMNIST_small/B.pickle
notMNIST_small/C.pickle
notMNIST_small/D.pickle
notMNIST_small/E.pickle
notMNIST_small/F.pickle
notMNIST_small/G.pickle
notMNIST_small/H.pickle
notMNIST_small/I.pickle
notMNIST_small/J.pickle
```

Chaque répertoire
contient environ
50 000 images

```
['notMNIST_small/A.pickle',
 'notMNIST_small/B.pickle',
 'notMNIST_small/C.pickle',
 'notMNIST_small/D.pickle',
 'notMNIST_small/E.pickle',
 'notMNIST_small/F.pickle',
 'notMNIST_small/G.pickle',
 'notMNIST_small/H.pickle',
 'notMNIST_small/I.pickle',
 'notMNIST_small/J.pickle']
```



Installation

Lesson 1 – classification logistique et descente de gradient

Lesson 2 - Réseau de neurone profond (to do)

Lesson 3 - Réseau de convolution (to do)

Lesson 4 – Réseau de neurone profond pour l'analyse de texte et de séquences (to do)

randori #2

- Objectif : afficher trois images de chaque classe
- Durée prévue : 7 minutes, 10 minutes maximum
- Nombre de codeurs : 1 à 2
- Nombre de lignes dans la réponse : 8

randori #3

- Objectif : vérifier que les 10 classes sont « balancées »
- Durée prévue : 5 minutes, 10 minutes maximum
- Nombre de codeurs : 1 à 2
- Nombre de lignes dans la réponse : 5

randori #4

- Objectif : vérifier que les 10 classes sont « balancées » après shuffling
- Durée prévue : 5 minutes, 10 minutes maximum
- Nombre de codeurs : 1 à 2
- Nombre de lignes dans la réponse : 5

SKIP

randori #5

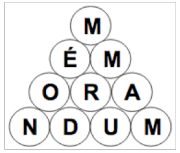
- Objectif : trouver les images dupliquées entre les jeux de données
- Durée prévue : 20 minutes à 30 minutes maximum
- Nombre de codeurs : 4 à 6
- Nombre de lignes dans la réponse : une quinzaine

- Options :
  - Quid des images « presques dupliquées » ?
  - Créer des datasets sans duplication

MÉMORANDUM

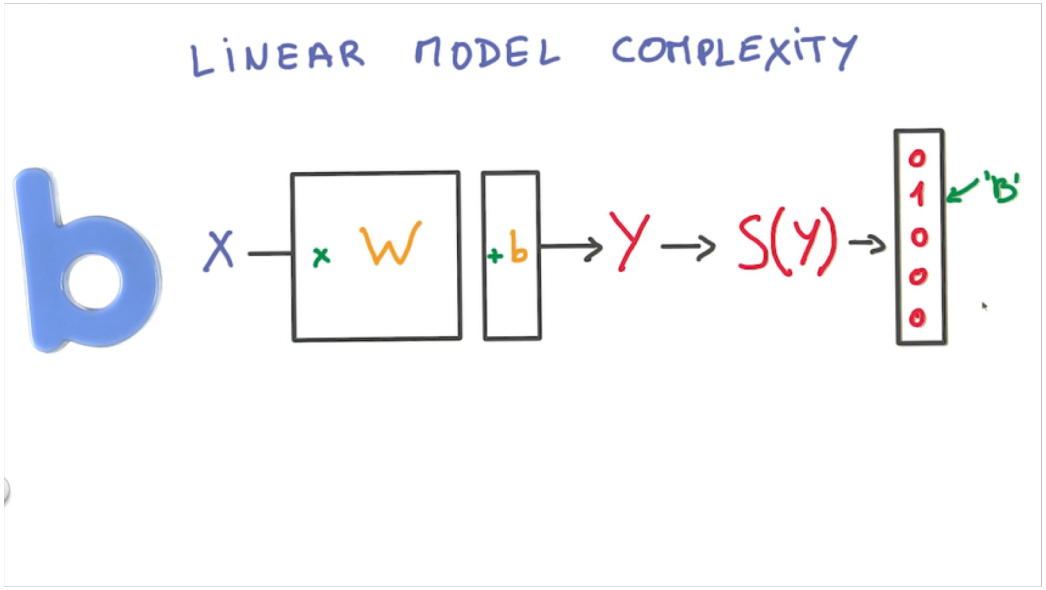
randori #6

- Objectif :
  - créer un classifieur avec scikit learn pour une régression logistique
  - tester sur des sous ensemble de 50 / 100 / 1000 / 5000
  - calculer le temps d'éxécution
- Durée prévue : 10 minutes, 15 minutes maximum
- Nombre de codeurs : 2 à 3
- Nombre de lignes dans la réponse : 5



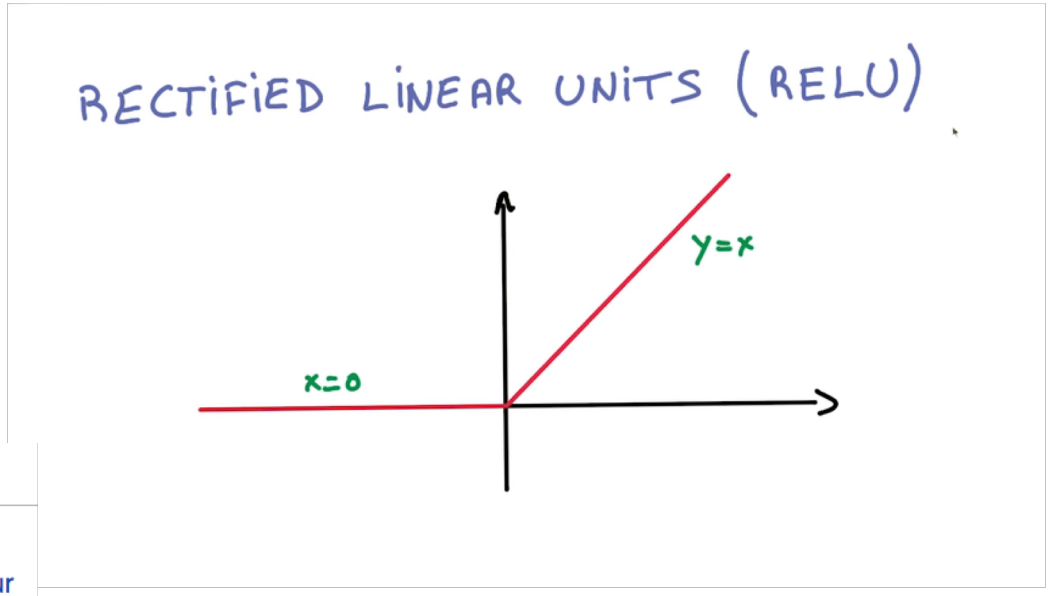
Nombre de paramètres (vidéo 2)

Les fonctions linéaires sont très stables, et se calculent bien sur GPU. On voudrait donc les conserver.
Par contre elles ne permettent pas de faire tout type de calculs.
Donc on va chercher à les modifier pour étendre le champ des possibles.



RELU (vidéo 5)

Le « relu » est une fonction très simple : c'est le maximum entre 0 et x ....



Rappel - dérivée :

Définition formelle [modifier | modifier le code]

Soit f une fonction réelle à valeurs réelles définie sur une réunion quelconque d'intervalles non triviaux, et x_0 appartenant à l'intérieur de l'ensemble de définition $\mathcal{D}_f$.

Pour tout $h \in \mathbb{R}^*$ tel que $[x_0, x_0 + h] \subset \mathcal{D}_f$, on appelle *taux d'accroissement de f en x_0 et avec un pas de h* la quantité :

$$t_{x_0}(h) = \frac{f(x_0 + h) - f(x_0)}{h}$$

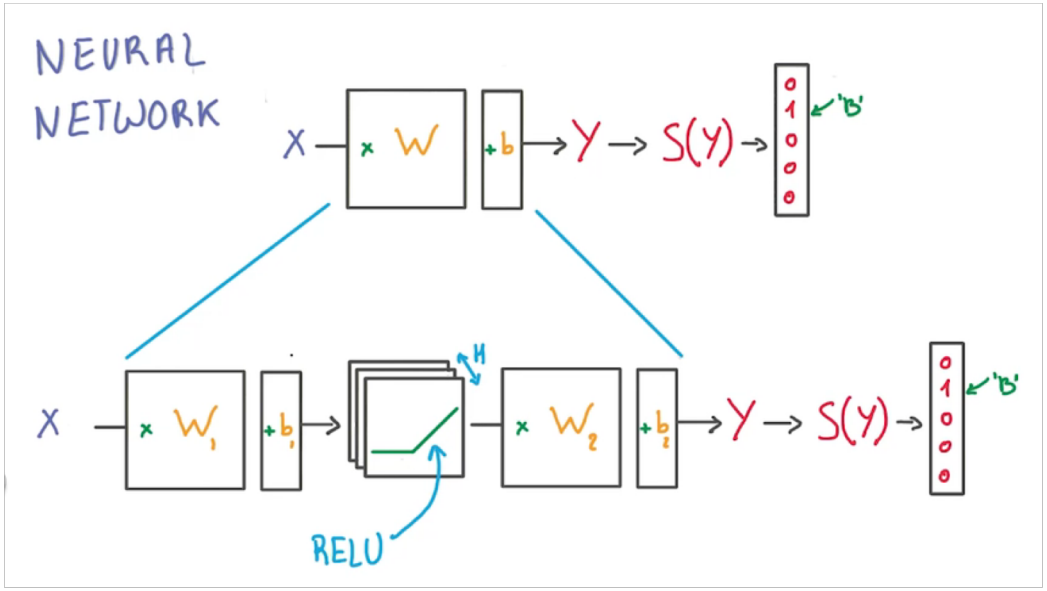
Réseau de neurone profond (vidéo 7)

On rend les fonctions non linéaires en insérant une fonction « relu » au milieu de la fonction...

Les calculs des dérivées restent alors très simple :

$[g(f(x))]' = g'(f(x)) * f'(x)$

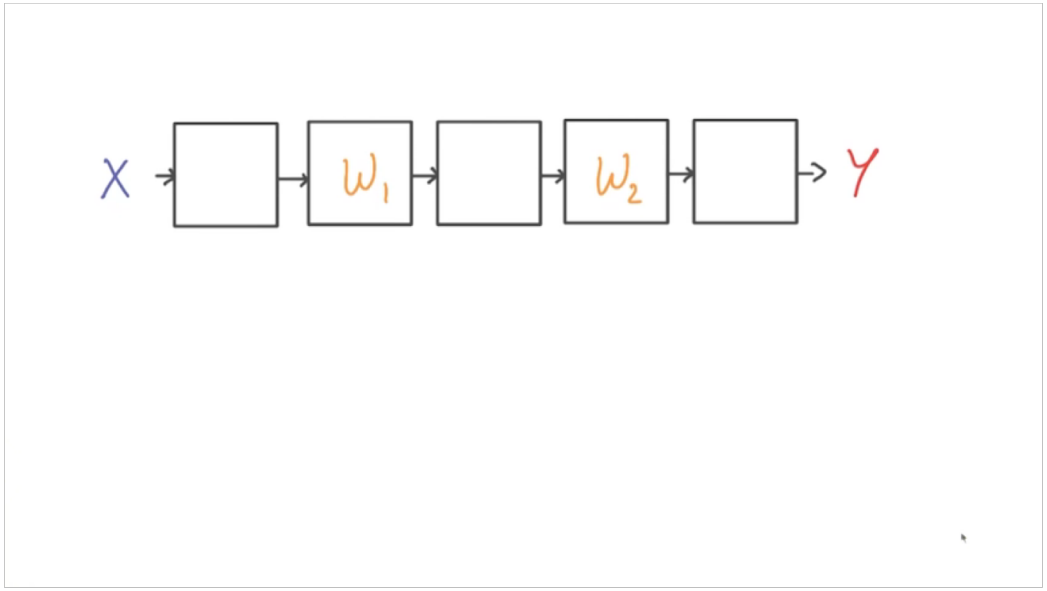
Comme la descente de gradient fait principalement du calcul de dérivée, il est important que cela reste simple.



Back Propagation

Le calcul du gradient pour un ensemble de poids **W** donné se fait en deux étapes :

1) **Forward propagation**
    1) relu -> transformation linéaire -> relu -> transformation linéaire
        1) On obtient alors une première prédiction Y
        2) Pour affiner cette prédiction et faire la descente de gradient, il faut trouver les dérivées. Pour trouver les dérivées, on va passer par l'étape 2



2) Back propagation

Le graph de back-propagation est calculé par Tensorflow (il est déduit du premier graph), on n'a donc pas en le programmer.
Il calcule des dérivées pour de la matrice de poids **W** que l'on va soustraire de **W** pour créer une nouvelle matrice qui la remplacera.

3) **on applique un coefficient alpha sur la dérivée de W avant la soustraction**

https://www.tensorflow.org/versions/r0.7/api_docs/python/state_ops.html#variables

Building Graphs:	26
Constants, Sequences, and Random Values:	14
Variables:	32
Tensor Transformations:	31
Math:	80
Control Flow:	26
Images:	35
Sparse Tensors:	11
Inputs and Readers:	30
Data IO (Python functions):	2
Neural Network:	34
Running Graphs:	20
Training:	35
Wraps python functions:	1
Testing:	6
Layers (contrib):	12
Utilities (contrib):	2

```
#Program that makes up some data in two dimensions, and then fits a line to it.

import tensorflow          as tf
import numpy               as np

# Create 100 phony x, y data points in NumPy, y = x * 0.1 + 0.3
x_data = np.random.rand(100).astype(np.float32)
y_data = x_data * 0.1 + 0.3

# Try to find values for W and b that compute y_data = W * x_data + b
W = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
b  = tf.Variable(tf.zeros([1]))
y   = W * x_data + b

loss        = tf.reduce_mean(tf.square(y - y_data)) # Minimize the mean squared
errors.
optimizer = tf.train.GradientDescentOptimizer(0.5)
train        = optimizer.minimize(loss)

init        = tf.initialize_all_variables()    # Before starting, initialize the variables
sess       = tf.Session()                          # Launch the graph.
sess.run(init)

for step in xrange(201):                          # Fit the line.
    sess.run(train)
    if step % 20 == 0: print(step, sess.run(W), sess.run(b))

# Learns best fit is W: [0.1], b: [0.3]
```

Librairie 0.7 (1/4)

Building Graphs

add_to_collection as_dtype bytes control_dependencies convert_to_tensor
convert_to_tensor_or_indexed_slices device Dimension DType get_collection get_default_graph
get_seed Graph GraphKeys import_graph_def load_op_library name_scope NoGradient op_scope
Operation register_tensor_conversion_function RegisterGradient RegisterShape reset_default_graph
Tensor TensorShape

Constants, Sequences, and Random Values

constant fill linspace ones ones_like random_crop random_normal random_shuffle
random_uniform range set_random_seed truncated_normal zeros zeros_like

Variables

all_variables assert_variables_initialized assign assign_add assign_sub constant_initializer count_up_to device
get_checkpoint_state get_variable get_variable_scope IndexedSlices initialize_all_variables
initialize_variables latest_checkpoint make_template moving_average_variables
random_normal_initializer random_uniform_initializer Saver scatter_add scatter_sub scatter_update
sparse_mask trainable_variables truncated_normal_initializer

Tensor Transformations

boolean_mask cast concat depth_to_space dynamic_partition dynamic_stitch
expand_dims gather pack pad rank reshape reverse reverse_sequence shape
shape_n size slice space_to_depth split squeeze string_to_number tile
to_bfloat16 to_double to_float

Librairie 0.7 (2/4)

Math

abs	accumulate_n	add	add_n	argmax	argmin	batch_cholesky	
batch_matmul	batch_matrix_determinant	batch_matrix_inverse	batch_matrix_solve	batch_matrix_solve_ls			
batch_matrix_triangular_solve	batch_self_adjoint_eig	ceil	cholesky	complex	complex_abs	conj	
cos	cross	diag	div	edit_distance	erf	erfc	

Control Flow

add_check_numerics_ops	Assert	check_numerics	cond	count_up_to	equal	greater	
greater_equal	group	identity	is_finite	is_inf	is_nan	less	less_equal
logical_and	logical_not	logical_or	logical_xor	no_op	not_equal	Print	select	tuple
verify_tensor_all_finite	where						

Images

adjust_brightness	adjust_contrast	adjust_hue	adjust_saturation	convert_image_dtype		
crop_to_bounding_box	decode_jpeg	decode_png	draw_bounding_boxes	encode_jpeg	encode_png	extract_glimpse
flip_left_right	flip_up_down	grayscale_to_rgb	hsv_to_rgb	pad_to_bounding_box	per_image_whitening	
random_brightness	random_contrast	random_flip_left_right	random_flip_up_down	random_hue		
random_saturation	resize_area	resize_bicubic				

Sparse Tensors

shape	sparse_concat	sparse_fill_empty_rows	sparse_reorder	sparse_retain
sparse_split	sparse_tensor_to_dense	sparse_to_dense	sparse_to_indicator	SparseTensor
SparseTensorValue				

Librairie 0.7 (3/4)

Inputs and Readers

batch batch_join decode_csv decode_json_example decode_raw FIFOQueue FixedLenFeature
FixedLengthRecordReader FixedLenSequenceFeature IdentityReader limit_epochs match_filenames_once
matching_files parse_example parse_single_example placeholder QueueBase RandomShuffleQueue
range_input_producer read_file ReaderBase shuffle_batch shuffle_batch_join size slice_input_producer
string_input_producer

Data IO (Python functions)

tf_record_iterator TFRecordWriter

Neural Network

avg_pool bias_add compute_accidental_hits conv2d conv2d_transpose depthwise_conv2d dropout
elu embedding_lookup fixed_unigram_candidate_sampler in_top_k l2_loss l2_normalize
learned_unigram_candidate_sampler local_response_normalization log_uniform_candidate_sampler max_pool
max_pool_with_argmax moments nce_loss relu relu6 sampled_softmax_loss separable_conv2d
sigmoid sigmoid_cross_entropy_with_logits

Running Graphs

AbortedError AlreadyExistsError CancelledError DataLossError DeadlineExceededError
FailedPreconditionError get_default_session InteractiveSession InternalError InvalidArgumentError
NotFoundError OpError OutOfRangeError PermissionDeniedError ResourceExhaustedError Session
UnauthenticatedError UnavailableError UnimplementedError UnknownError

Librairie 0.7 (4/4)

Training

AdagradOptimizer	AdamOptimizer	add_queue_runner	AggregationMethod	clip_by_average_norm
clip_by_global_norm	clip_by_norm	clip_by_value	Coordinator	exponential_decay
ExponentialMovingAverage	export_meta_graph	FtrlOptimizer	generate_checkpoint_state_proto	
global_norm	global_step	GradientDescentOptimizer	gradients	histogram_summary
image_summary	import_meta_graph	LooperThread	merge_all_summaries	merge_summary
MomentumOptimizer	Optimizer			

Wraps python functions

py_func

Testing

assert_equal_graph_def	compute_gradient	compute_gradient_error	get_temp_dir	is_built_with_cuda	main

Layers (contrib)

assert_same_float_dtype	convolution2d	fully_connected	l1_regularizer	l2_regularizer
summarize_activation	summarize_activations	summarize_collection	summarize_tensor	summarize_tensors
xavier_initializer	xavier_initializer_conv2d			

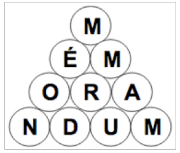
Utilities (contrib)

constant_value	make_tensor_proto

randori # 7

- Objectfs : Reformat into a shape that's more adapted to the models we're going to train:
  - data as a flat matrix,
  - labels as float 1-hot encodings.

randori



Practical issues of cross entropy:

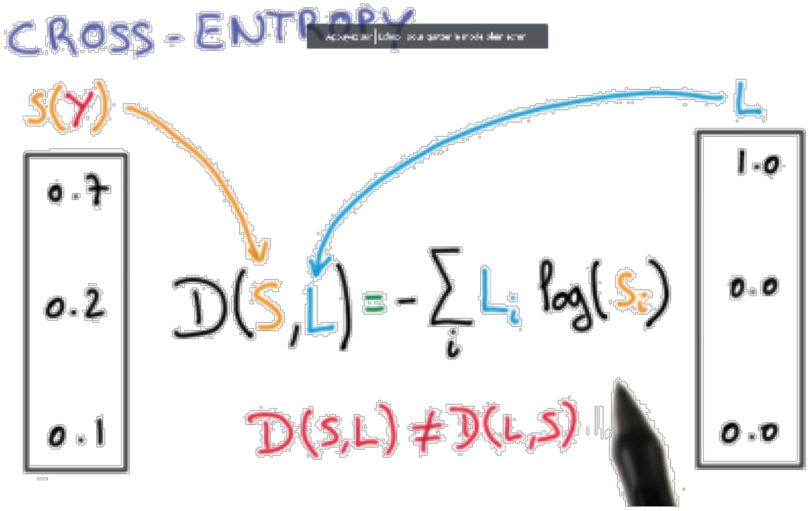
Tensorflow NaN bug? - Stack Overflow
stackoverflow.com/questions/.../**tensorflow-nan**-bug ▾ Traduire cette page
14 nov. 2015 - I'm using **TensorFlow** and I modified the tutorial example to take my RGB ... **cross_entropy** = -tf.reduce_sum(y_*tf.log(tf.clip_by_value(y_conv ...

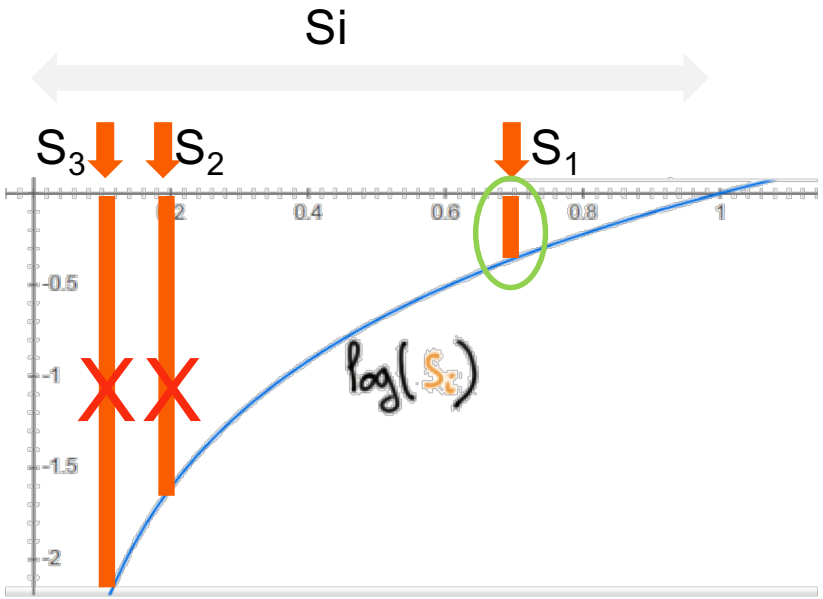
python - Why does TensorFlow return [[nan nan]] instead of ...
stackoverflow.com/.../why-does-**tensorflow**-return-**na**... ▾ Traduire cette page
25 nov. 2015 - Why does **TensorFlow** return [[**nan nan**]] instead of probabilities from a CSV file? ..... + 1e-50)) ## avoid **nan** due to 0*log(0) **cross_entropy** = tf.

python - Why does TensorFlow example fail when ...
stackoverflow.com/.../why-does-**tensorflow**-example-... ▾ Traduire cette page
10 nov. 2015 - I was looking at the **Tensorflow** MNIST example for beginners and found that in this part: ... here: stackoverflow.com/questions/33712178/**tensorflow-nan**-bug ... **cross_entropy** = -tf.reduce_sum(y_*tf.log(y)) **cross_entropy** = tf.

Practical issues of cross entropy:

```
cross_entropy = -tf.reduce_sum(y_*tf.log(y_conv))

Replace with:
cross_entropy = -tf.reduce_sum(y_*tf.log(tf.clip_by_value(y_conv,1e-10,1.0)))
```





Seule la probabilité calculée de la bonne classe est prise en compte (le coefficient L des autres classes est à 0)
Pour la bonne classe l'erreur est calculée comme $-\text{Log}(S_1)$

$$D(S,L) = -\log(S_1)$$

Test de progression significative

3000 EXAMPLES

80% → 81%

1%*3000= 30 OK !

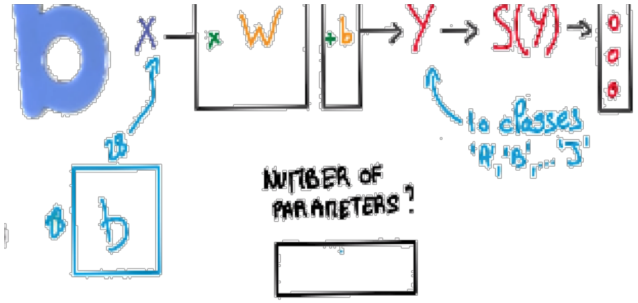
80% → 80.5%

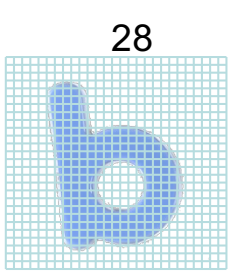
0,5%*3000=15 KO !

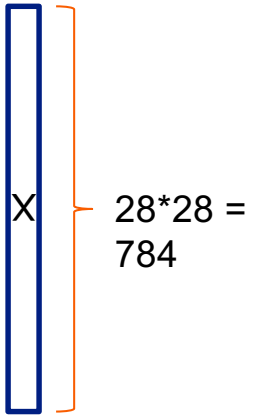
80% → 80.1%

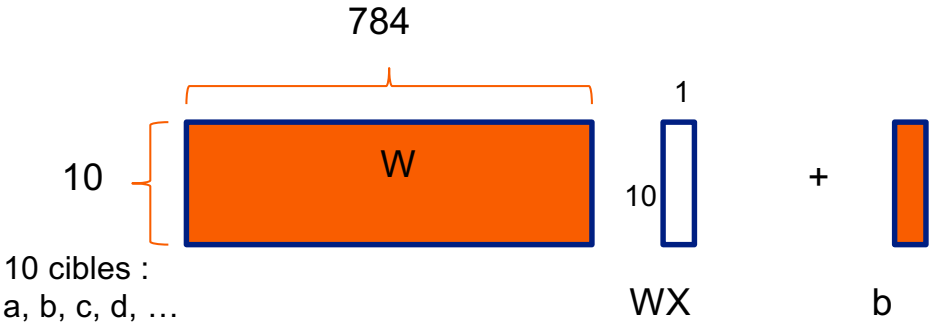
0,1%*3000=3 KO !

Une progression portant sur plus de 30 unités considéré comme signiufication
→ La taille de son test set défini le niveau de précision de réglage du modèle que vous pouvez chercher



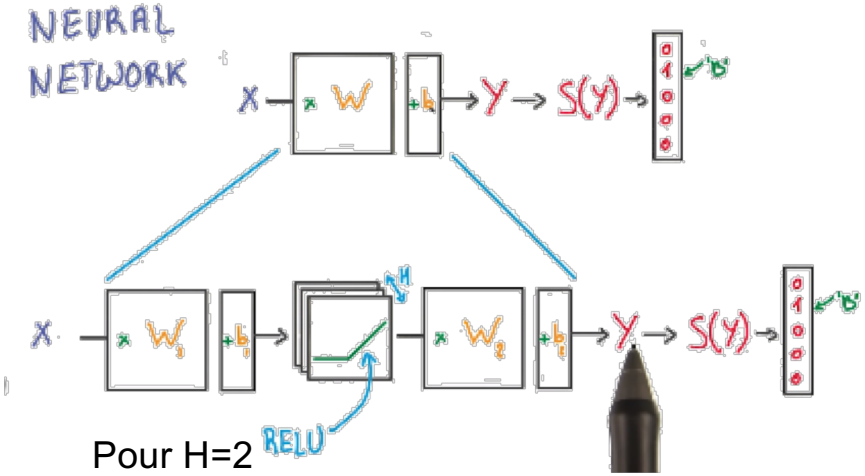






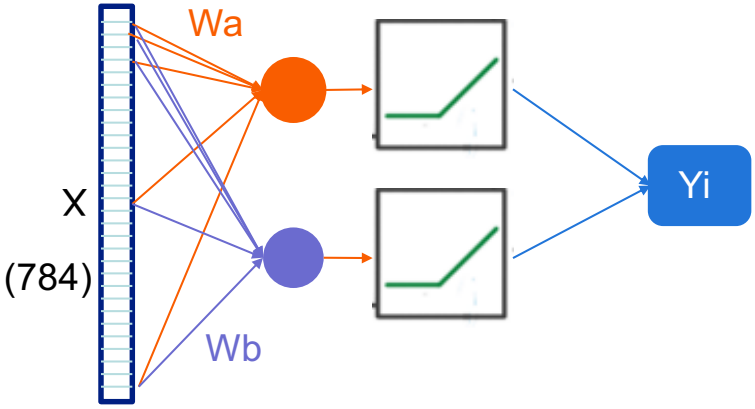
Vecteur de
28*28=784

Nombre de parametres :
784*10 (pour w) + 10 (pour b) : 7850



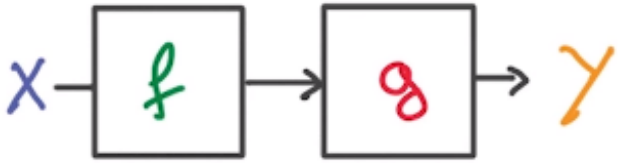
Pour chaque cible

Pour H=2

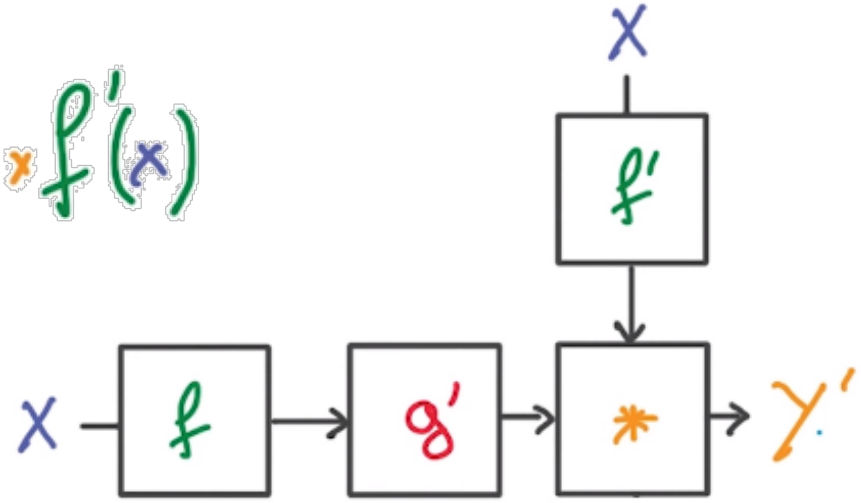


Il y a donc 784*10*2 paramètres W pour la première couche

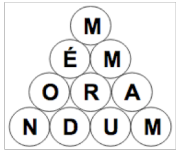
Il y a 2*10 paramètres W pour la 2è couche



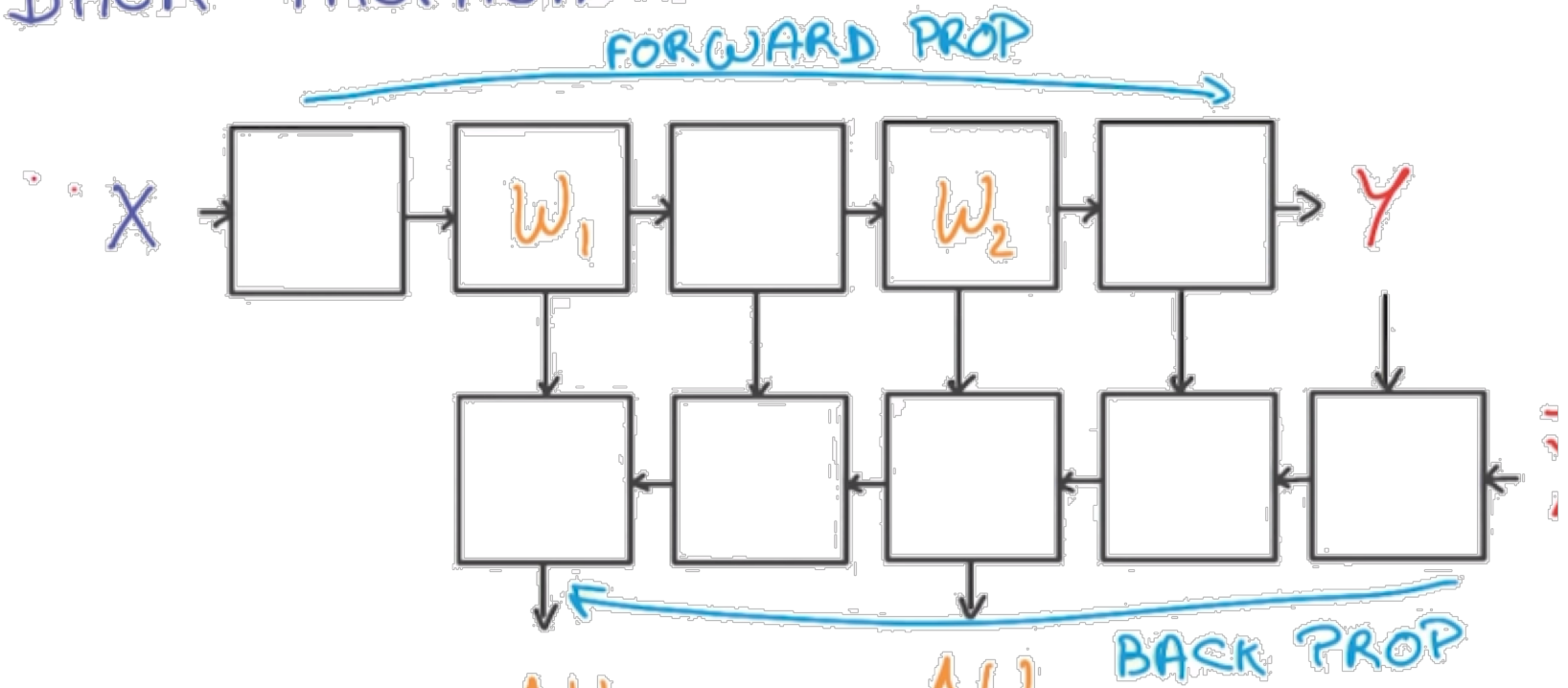
$$[g(f(x))]' = g'(f(x)) \cdot f'(x)$$



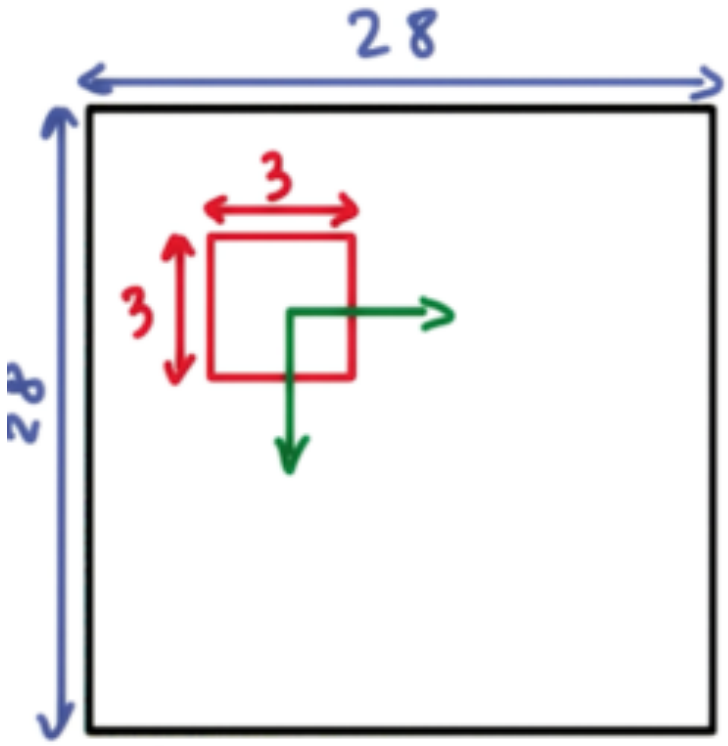
WX



BACK - PROPAGATION



STRIDES, DEPTH & PADDING



INPUT DEPTH = 3

OUTPUT DEPTH = 8

PADDING	STRIDE	OUTPUT WIDTH	OUTPUT HEIGHT	OUTPUT DEPTH
'SAME'	1	28	28	8
'VALID'	1	26	26	8
'VALID'	2	13	13	8