Tensorflow Baby Step

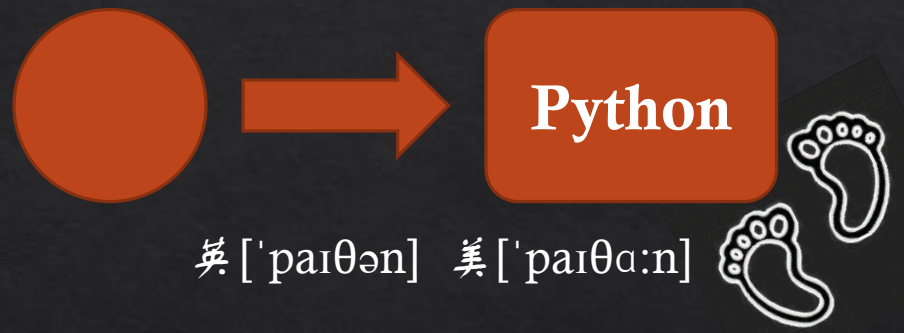
AI-Fundamental Course Slides © 2018

chendian@ict.ac.cn

Dian Chen

Supervisor: Ping Luo

Python Zero to Hero



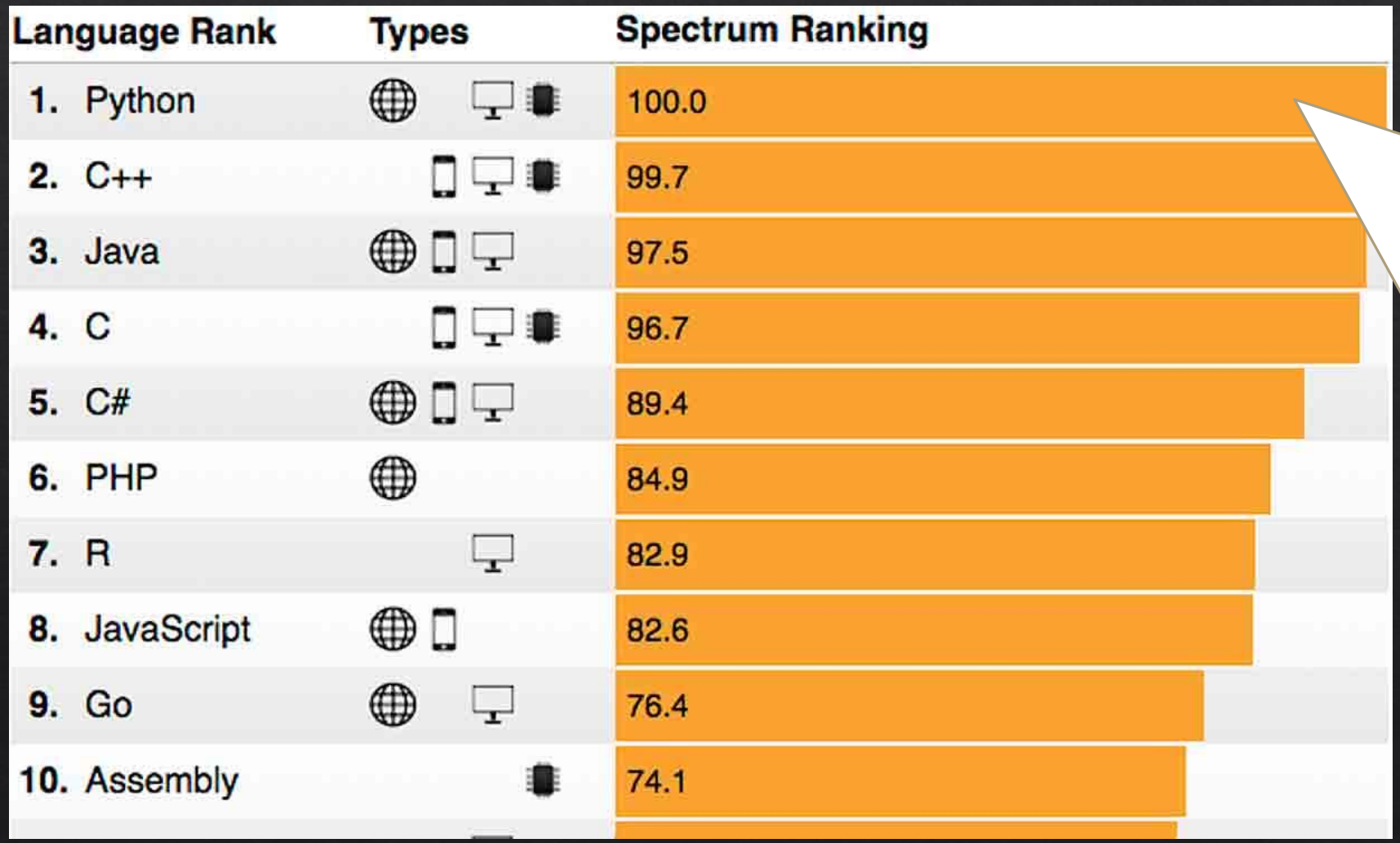

```python
def post_data(self, inp, url):
    try:
        if url in self.url:
            url = self.url[url]
        ret = requests.post(url, json=inp)
        code = ret.status_code
    except Exception as e:
        logging.warning("POST failed for {}".format(e))
        return False
    info = "POST Finished, status: {}".format(
        status_dict[code]
    )
    return code, info
```

游标卡尺

无需预先定义数据类型

注意：numpy库支持的Python版本为 >=2.7.* or >=3.4.*；
推荐 2.7.6 / 2.7.10 / 2.7.12 / 3.6.0 及以上

Why Python



根据 IEEE Spectrum 发布的研究报告显示：在 2016 年排名第三的 Python 已经连续两年已经成为世界上最受欢迎的语言，C++ 和 Java 分列第二和第三位。

2018-07-31 IEEE Spectrum

Numpy Quickstart

```python
import numpy as np
matrix = [[1,2],[3,4],[5,6]]

print (matrix)
print ("==============")
print (np.array(matrix, dtype=float))
```

.r_
.c_
.copy
.empty
.arange

.ravel
.shape
.reshape
.hstack
.vstack

```
[[1, 2], [3, 4], [5, 6]]
==============
[[ 1.  2.]
 [ 3.  4.]
 [ 5.  6.]]
```

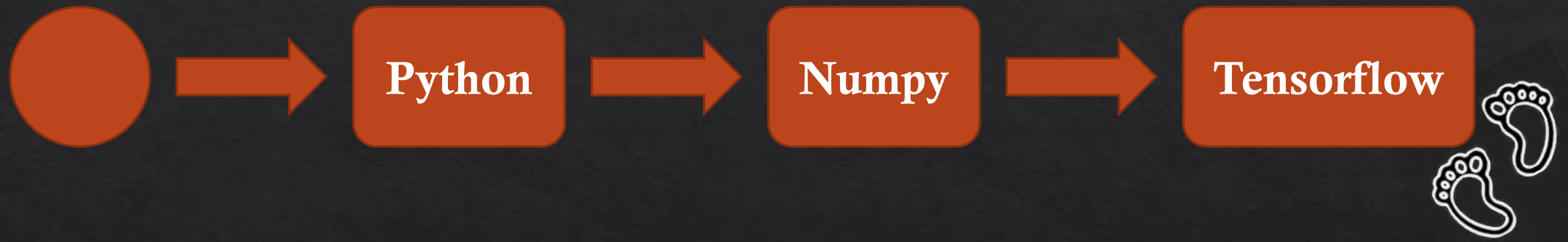
注意：numpy中需要是一个规则、对齐的“矩阵”
如果是类似 [[1,2,3], [4,5], [6]] 这样不对齐的list，
是无法转化为numpy.array的。

NumPy（Numeric Python）

- Import numpy as np
- Import tensorflow as tf

- 提供了许多高级的数值编程工具，如：矩阵数据类型、矢量处理，以及精密的运算库。
- 是一个用python实现的科学计算包。

Tensorflow Tutorial



Tensorflow

- TensorFlow™ 是一个开放源代码软件库，用于进行高性能数值计算。
- 借助其灵活的架构，用户可以轻松地将计算工作部署到多种平台（CPU、GPU、TPU）和设备（桌面设备、服务器集群、移动设备、边缘设备等）。

- TensorFlow™ 最初是由 Google Brain 团队（隶属于 Google 的 AI 部门）中的研究人员和工程师开发的，可为机器学习和深度学习提供强力支持，并且其灵活的数值计算核心广泛应用于许多其他科学领域。

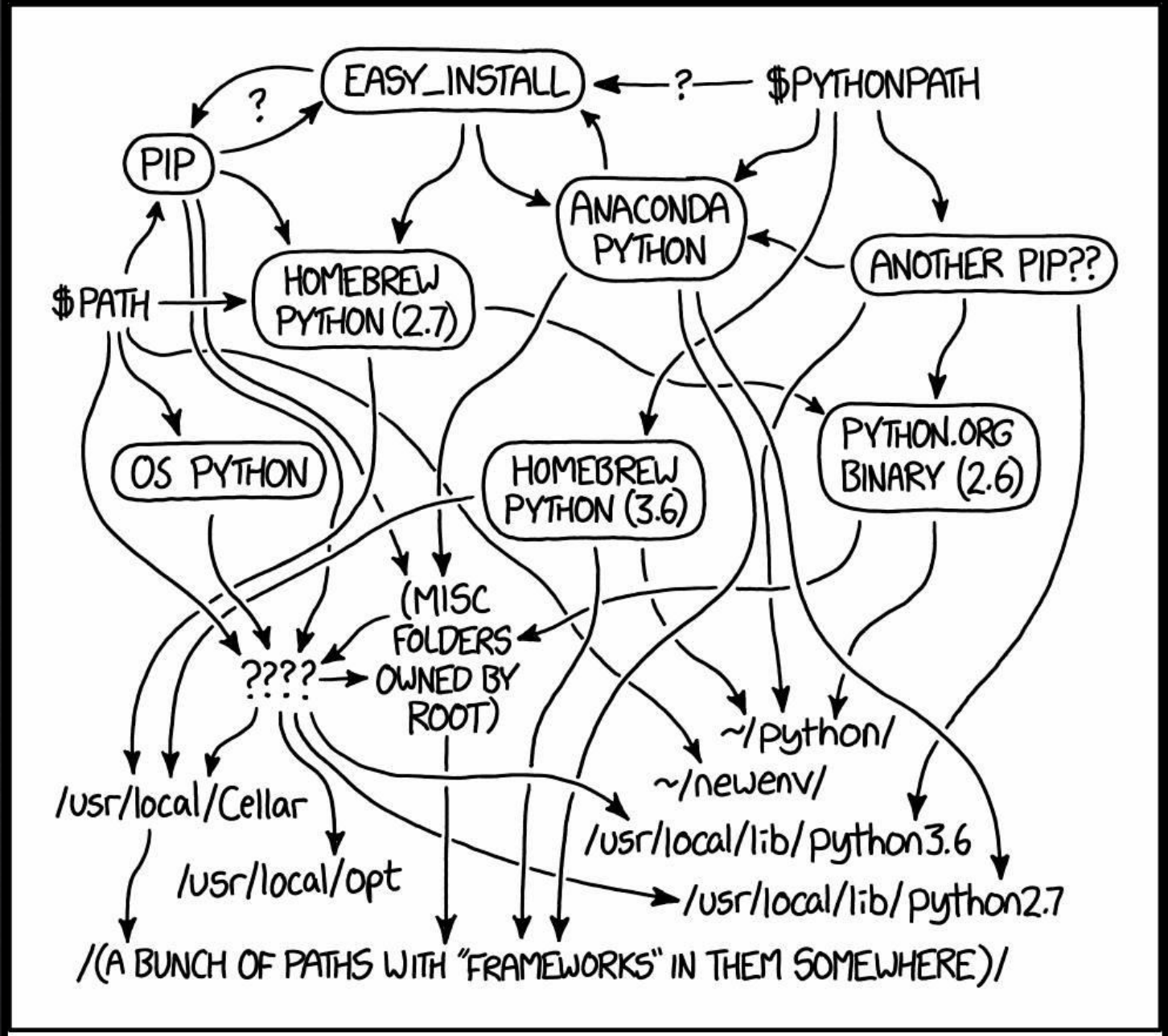
- 平台共有特性：自动求导
- 底层转化：Cuda-C语言

Python

Numpy

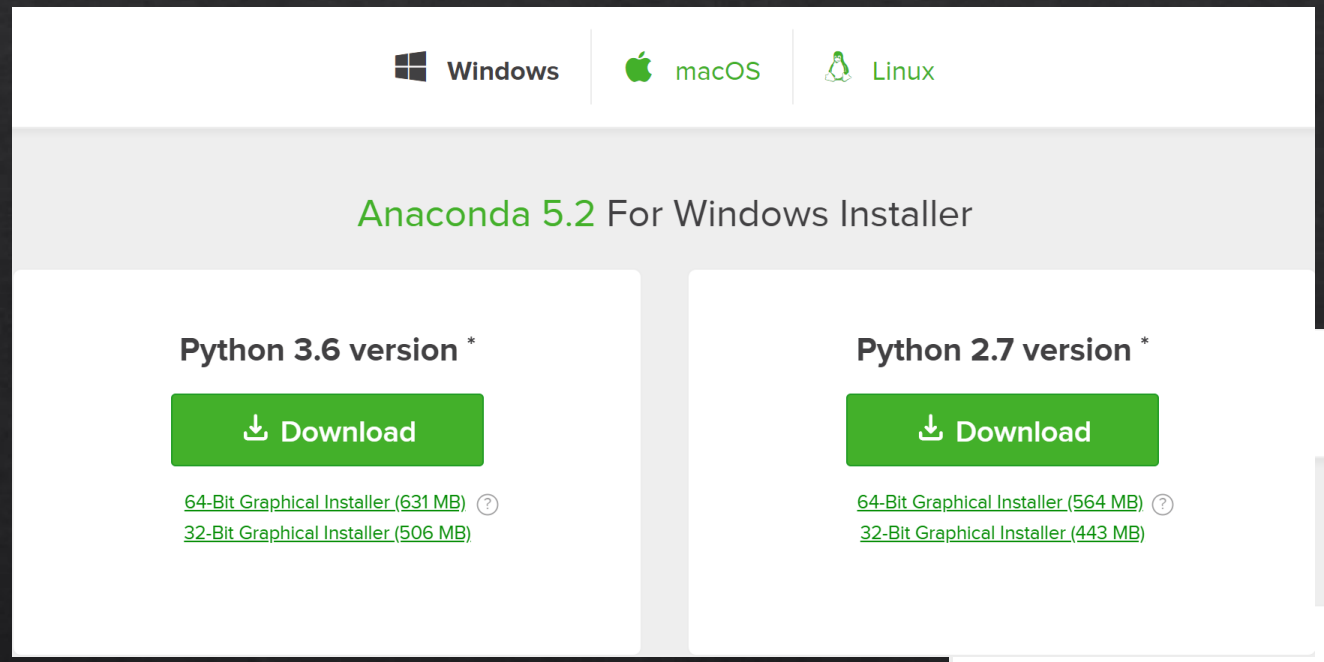
Tensorflow

Crazy Environment Configuration



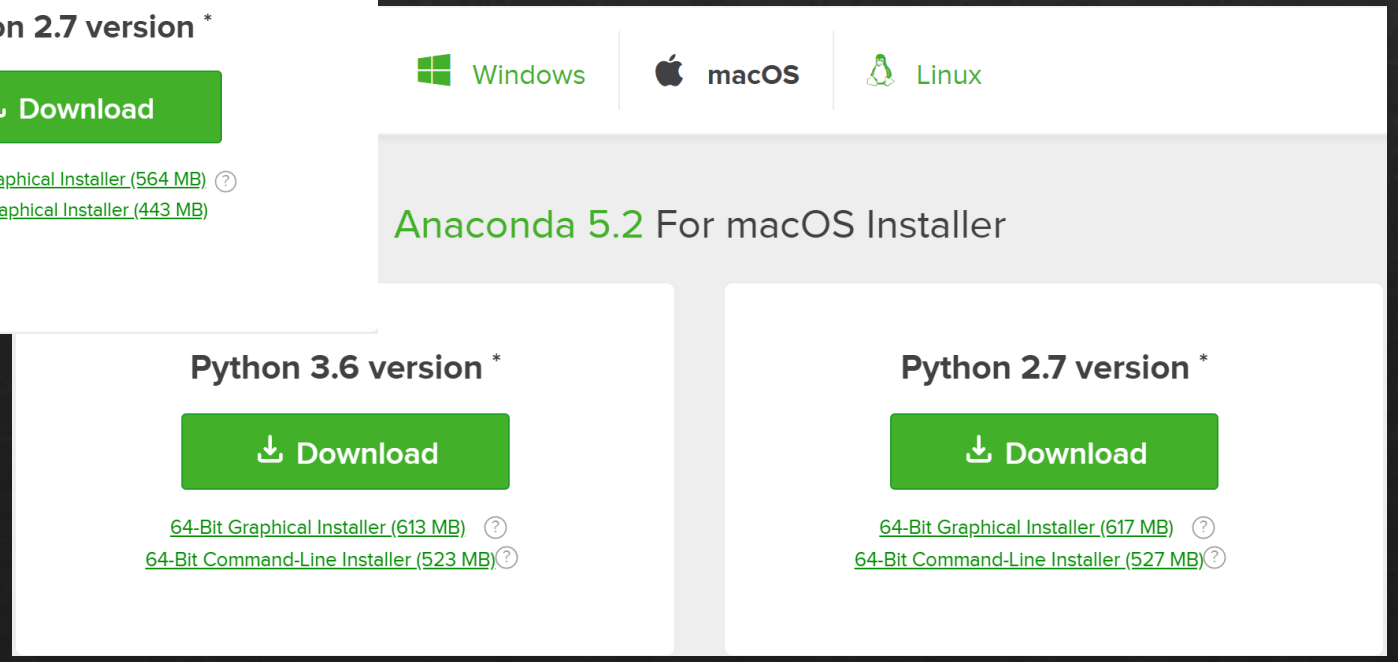
MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

https://www.anaconda.com/download/



ANACONDA.

/ ˌanəˈkändə/



Conda是一个开源的包、环境管理器，可以用于在同一个机器上安装不同版本的软件包及其依赖，并能够在不同的环境之间切换。

Installation on Linux

- Install `Anaconda`: download package from https://anaconda.org/
- （for Python2）$ conda create -n *my_env* python=2.7
- （for Python3）$ conda create -n *my_env* python=3.6

- $ source activate my_env
- (my_env) $ pip install tensorflow

To deactivate this environment, use:

- (my_env) $ source deactivate

Installation on windows

- Install `Anaconda`: download package from https://anaconda.org/
- （for Python2）cmd> conda create -n *my_env* python=2.7
- （for Python3）cmd> conda create -n *my_env* python=3.6

- cmd> activate my_env
- (my_env) cmd> pip install tensorflow

To deactivate this environment, use:

- (my_env) cmd> deactivate

Installation with Docker

- https://hub.docker.com/r/tensorflow/tensorflow/

- # Download latest image
- docker pull tensorflow/tensorflow

- # Start a Jupyter notebook server
- docker run -it -p 8888:8888 tensorflow/tensorflow

And of course

- Use your search engine!

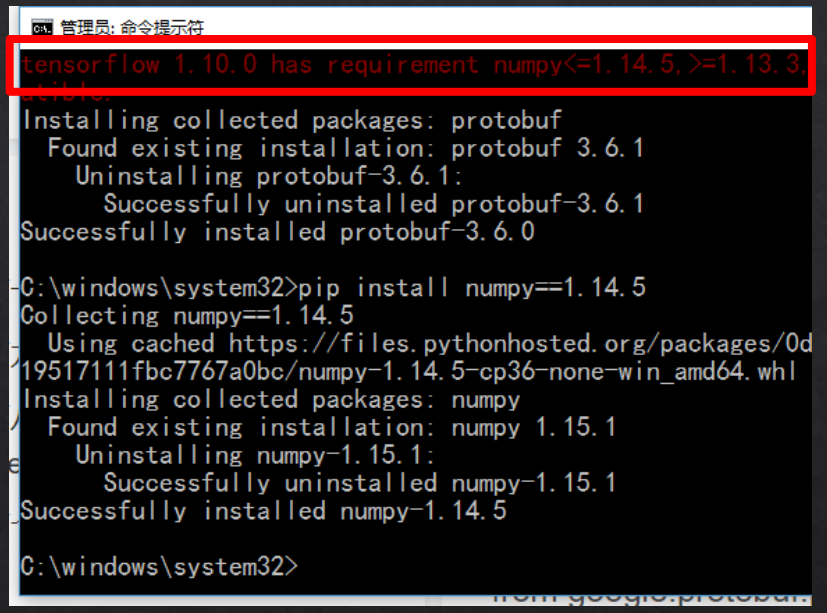
- If something goes wrong,

First,

- Check version (i.e. numpy <=1.14.5, >=1.13.3)

Then,

- Stack-overflow-oriented programming
- CSDN-oriented programming
- Classmate-oriented programming



Environment Configuration

```
C:\windows\system32>python
Python 3.6.0 |Anaconda 4.3.1 (64-bit)| (default, Dec 23 2016, 11:57:41) [MSC v.1900 64 bit (AMD64)] on
 win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> import numpy as np
>>>
```

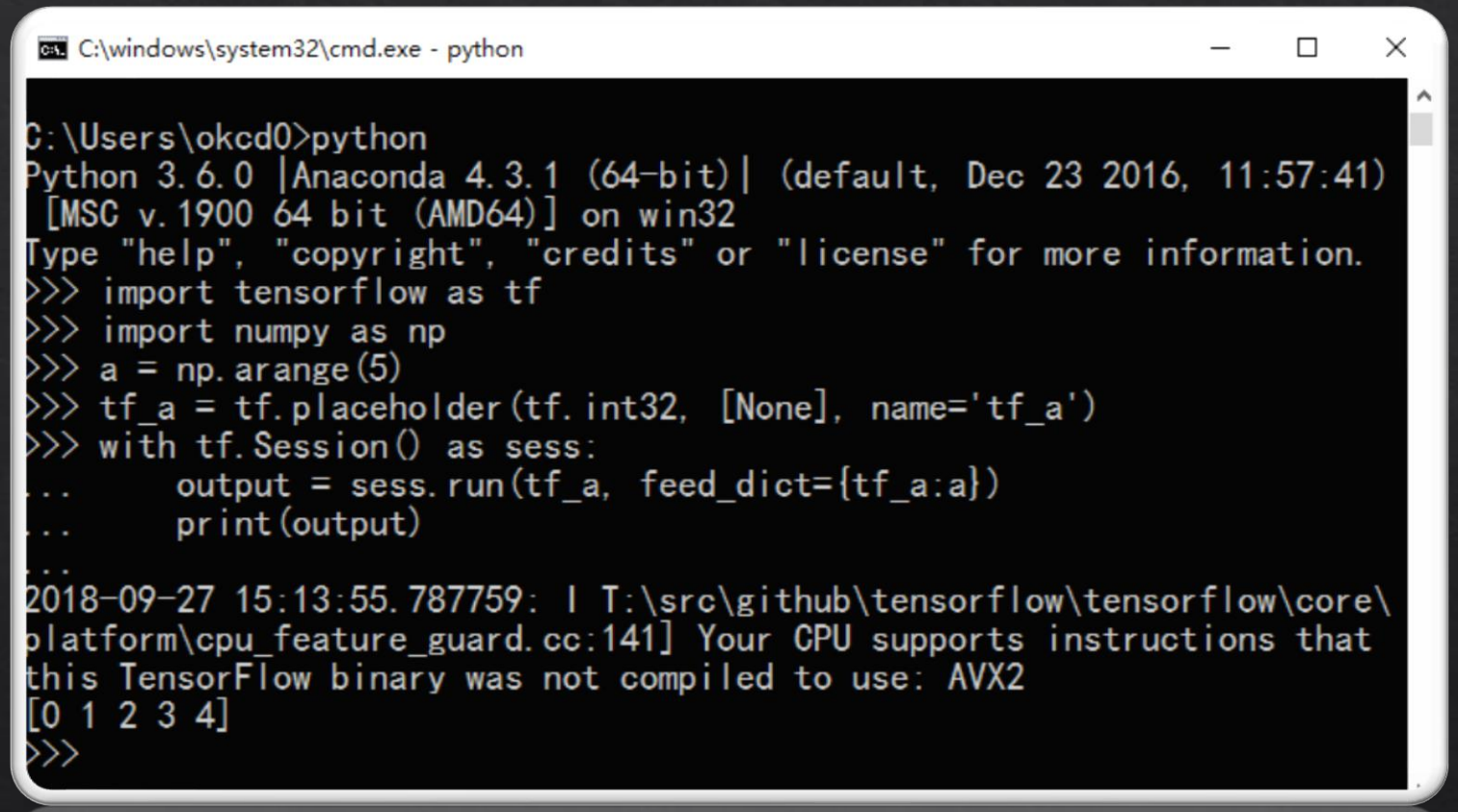
↑ **Windows**

Linux →

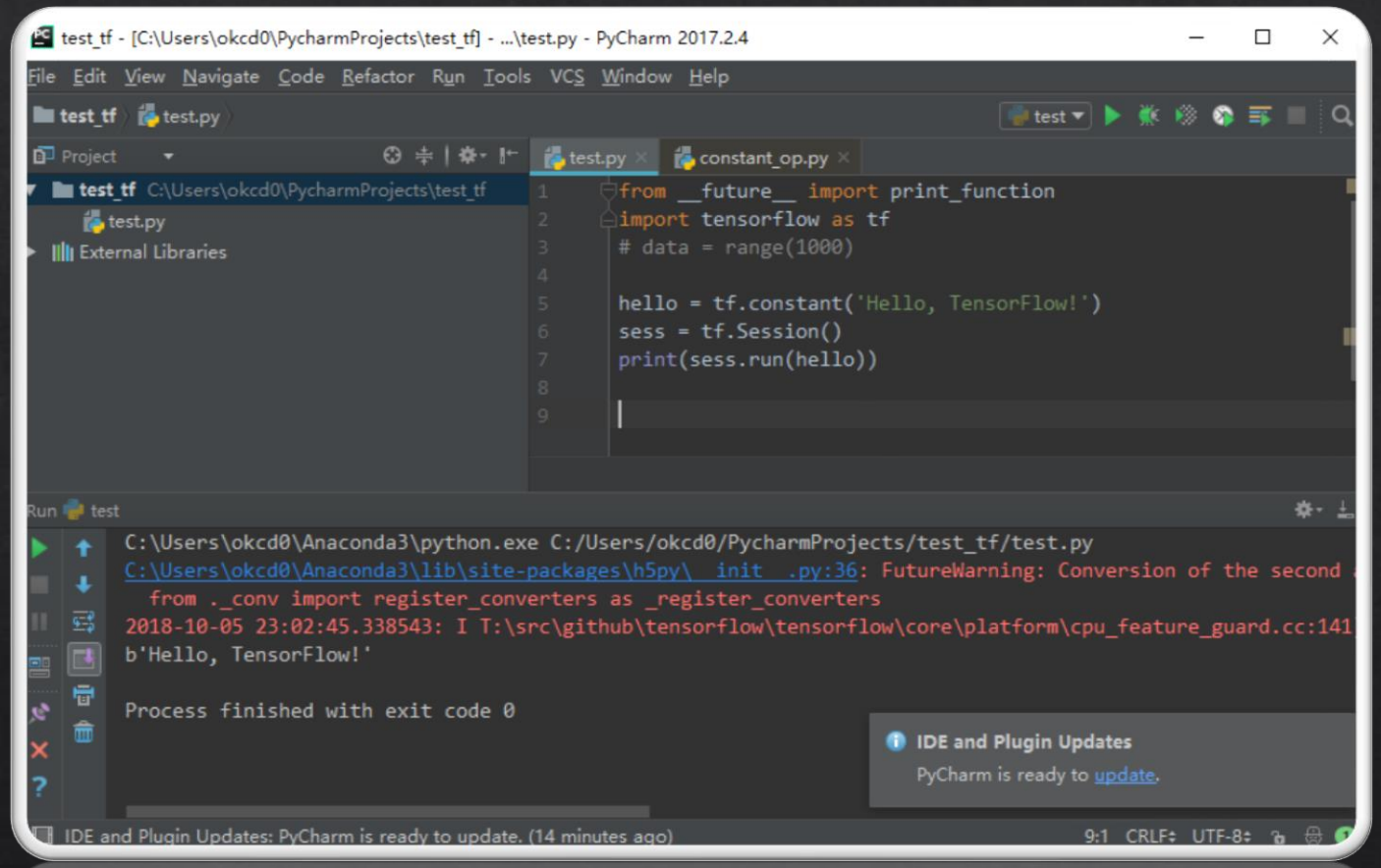
```
[chend@c9 AutoDoc]$ python
Python 2.7.13 |Anaconda custom (64-bit)| (default, Dec 20 2016, 23:09:15)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://anaconda.org
>>> import numpy as np
>>> import tensorflow as tf
>>> np.zeros([3,4])
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
>>>
```

It means the painful environment configuration is over (for Windows / Linux)

Coding in command_line



Coding in local IDE

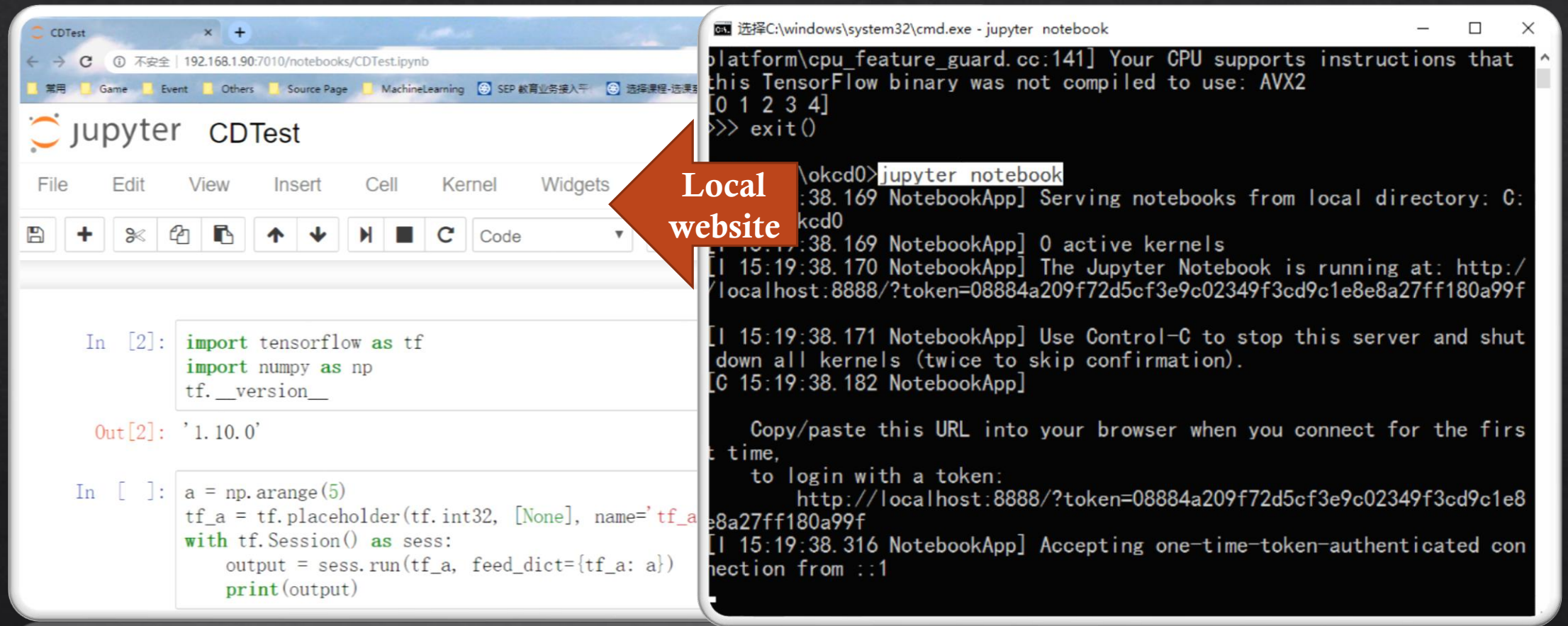


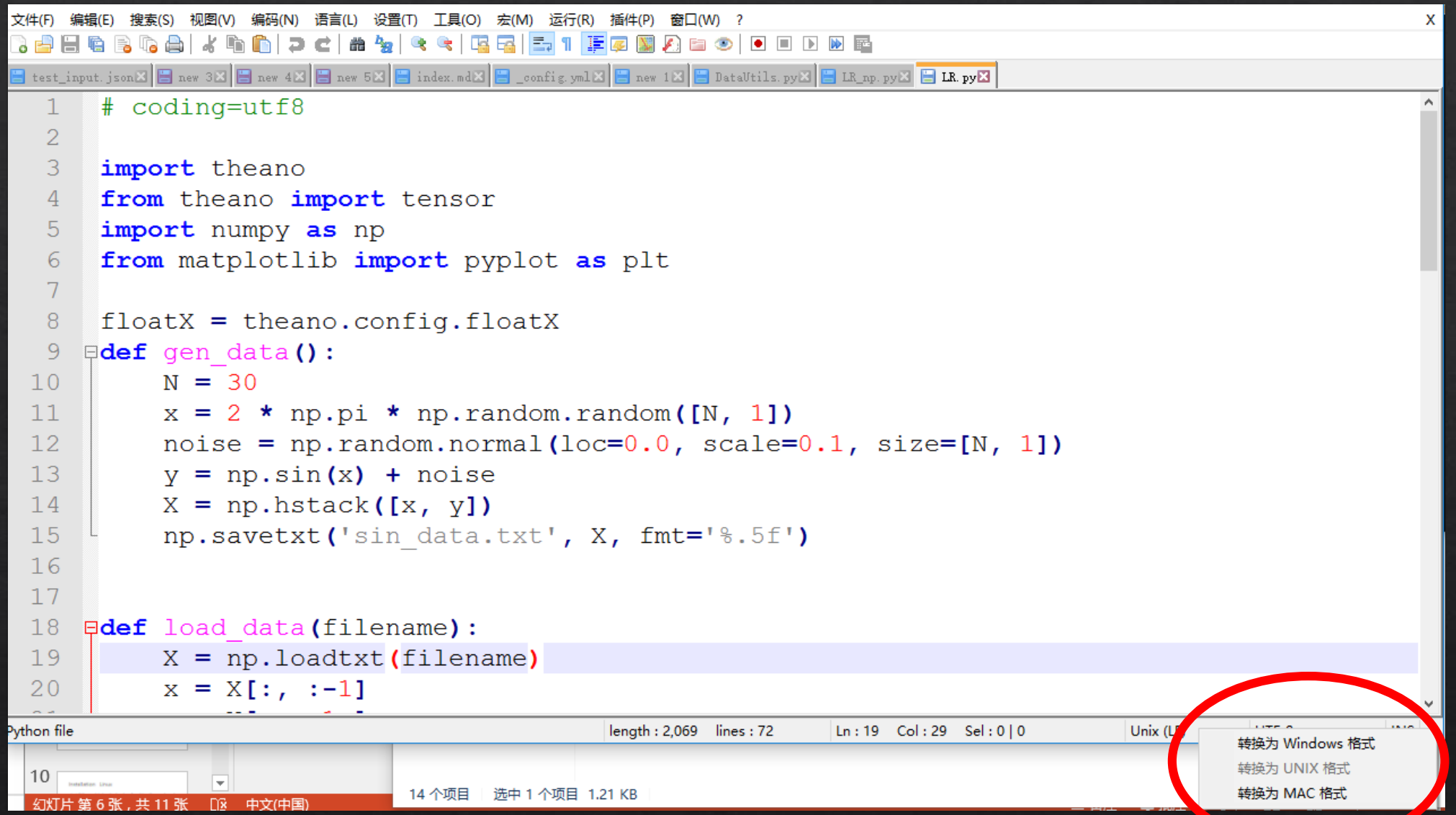
Pycharm IDE

What's more:

Spyder
Sublime Text
Eclipse + PyDev
Visual Studio + PTVS

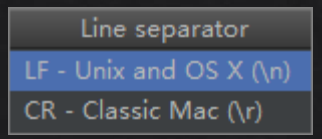
Coding in Jupyter Notebook





SyntaxError: encoding problem: utf8

文件格式： Windows (CR LF) / Unix (LF) / Mac (CR)



General Ideas

```
# 1. define input for type/size/name
tf_a = tf.placeholder(tf.int32, [None], name='tf_a')

# 2. build calculation network with input
tf_output = tf.Sigmoid(tf_a)

# 3. prepare real_data
a = np.arange(5)

# 4. run the network with feed_data
with tf.Session() as sess:
    output = sess.run(tf_output, feed_dict={tf_a: a})
    print(output)
```

1. Define input for type/size/name

import tensorflow as tf

```
# 1. define input for type/size/name
x_scalar = tf.placeholder(tf.int32,
                          name='my_x1')  # feeds = 1.0
x_vector = tf.placeholder(tf.int32, [None],
                          name='my_x2')  # feeds = np.ones(3)
x_matrix = tf.placeholder(tf.int32, [None, 5],
                          name='my_x3')  # feeds = np.ones(3, 5)
x_tensor = tf.placeholder(tf.int32, [None, 5, 7],
                          name='my_x4')  # feeds = np.ones(3, 5, 7)
```

Hints: small prime number as matrix shape

DataType	DataSize	CustomName

```
x = tf.placeholder(tf.int32, [None, 5], name='my_x')
```

数据类型

tf.int32	int32
tf.int64	int64
tf.byte	byte
tf.float32	float32
tf.float64	float64

数据计算

matmul	矩阵乘法
+ - * /	逐元素计算
concat	拼接操作
log	自然对数
grad	求导

What's more:
https://tensorflow.google.cn/api_docs/python

数据维度

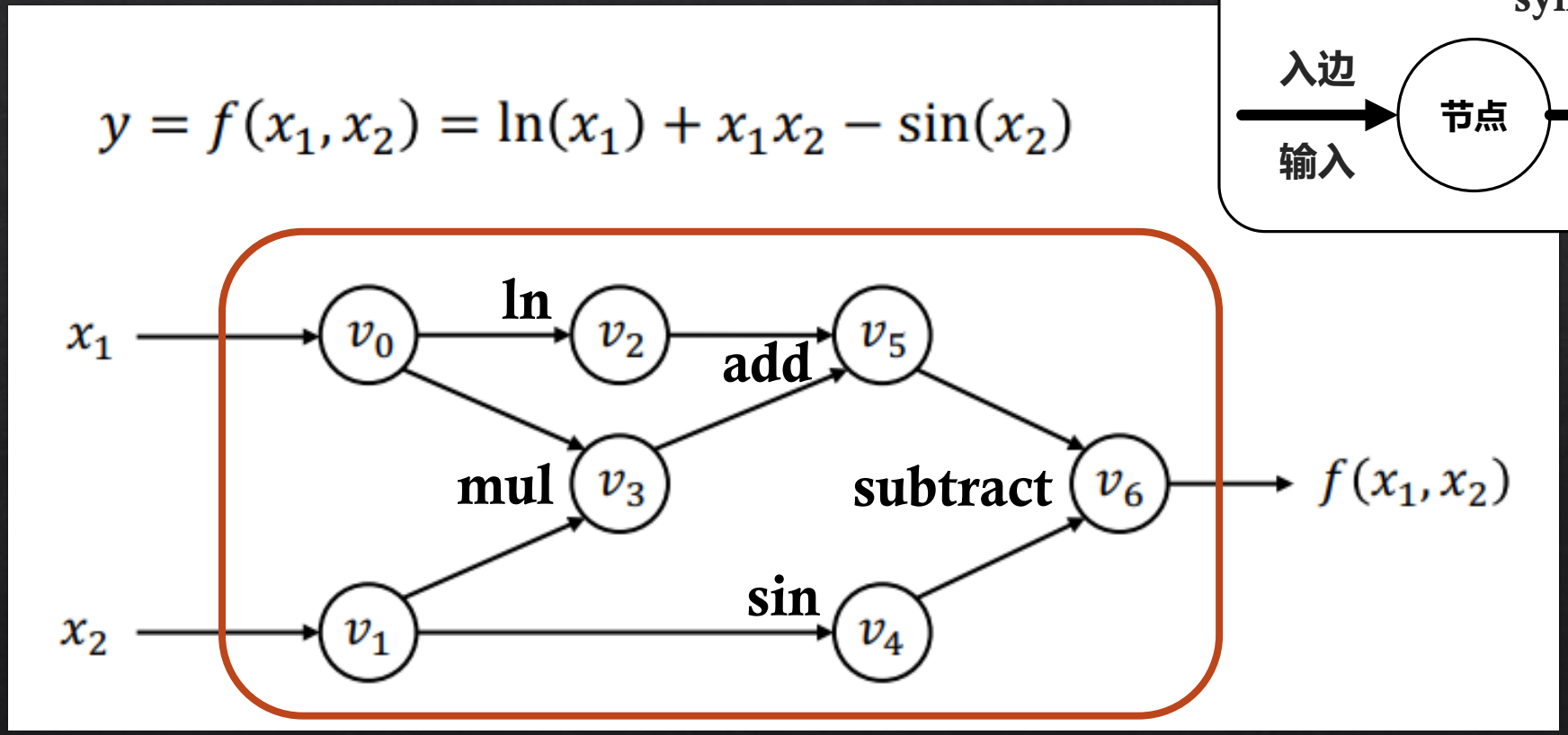
```
x1 = tf.placeholder(tf.int32, 'x1')
x2 = tf.placeholder(tf.int32, [None], 'x2')
x3 = tf.placeholder(tf.int32, [None, 5], 'x3')
```

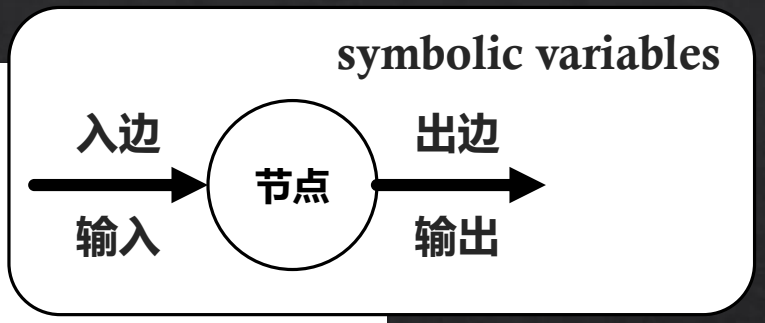
C code Transfer

```
int x1
int x2[]
int x3[][5]
```

2. Build calculation network with input

symbolic variables





Calculation Network/Graph

```
# x, y are both in shape [2, 2]
add_calc = tf.add(x_symb, y_symb)  # the same as x+y
element_product = tf.mul(x_symb, y_symb)  # the same as x*y
matrix_product = tf.matmul(x_symb, y_symb)
sigmoid_calc = 1.0 / (1.0 + tf.exp(-x_symb))
```

symbolic variables can be fed with any data in correct shape.

$$if\ x_{symb} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix},$$

$$y_{symb} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$sum: \begin{bmatrix} 2 & 2 \\ 3 & 5 \end{bmatrix}$$

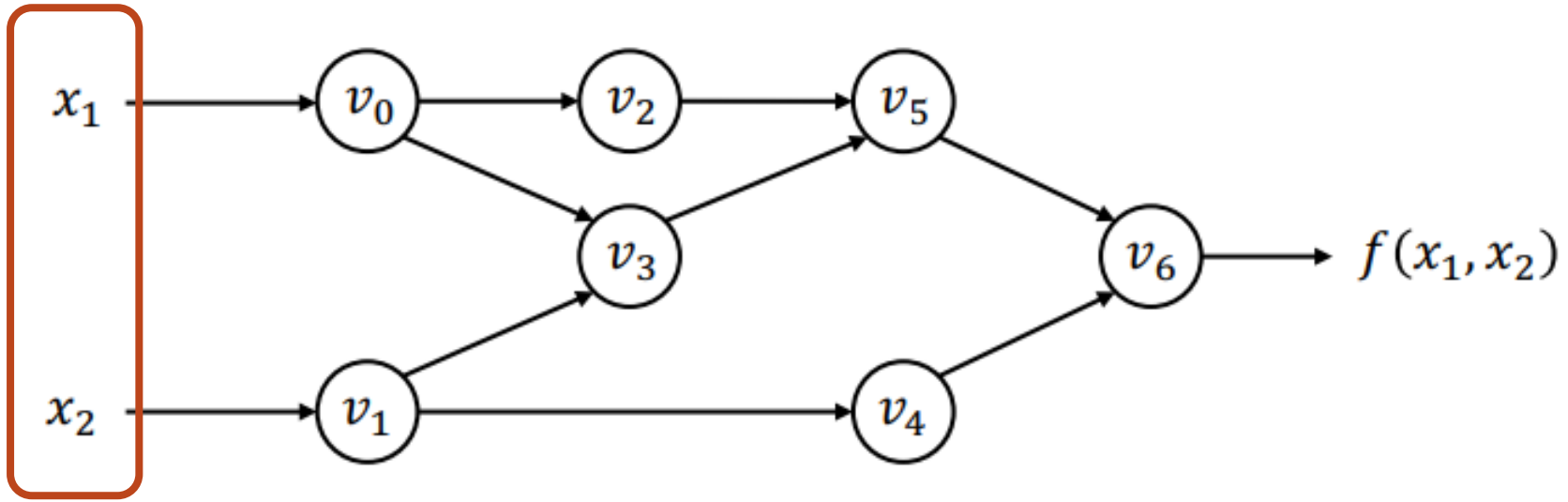
$$matrix\ product: \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

$$element-wise\ product: \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}$$

3. Prepare real data (feed data)

import numpy as np

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$



Generate feed data with np

- From python list:
  - np.array([1,2,3,4,5])
  - np.array([[1,0],[0,1]])
- Directly generate:
  - np.ones(3)
  - np.eye(3)
  - np.zeros([2,5])
  - np.arange(6).reshape([2,3])
- Convenient calculation/operation on <np.array>

```
np.ones(3):
[ 1.  1.  1.]

np.eye(3):
[[ 1.  0.  0.]
 [ 0.  1.  0.]
 [ 0.  0.  1.]]

np.zeros([2,5]):
[[ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]]

np.ones([2,3,4]).ndim
3

np.ones([2,3,4]).shape
(2, 3, 4)

np.arange(8):
[0 1 2 3 4 5 6 7]

np.arange(6).reshape([2,3]):
[[0 1 2]
 [3 4 5]]
```

What's more:
github.com/rougier/numpy-100

Generate feed data with tf

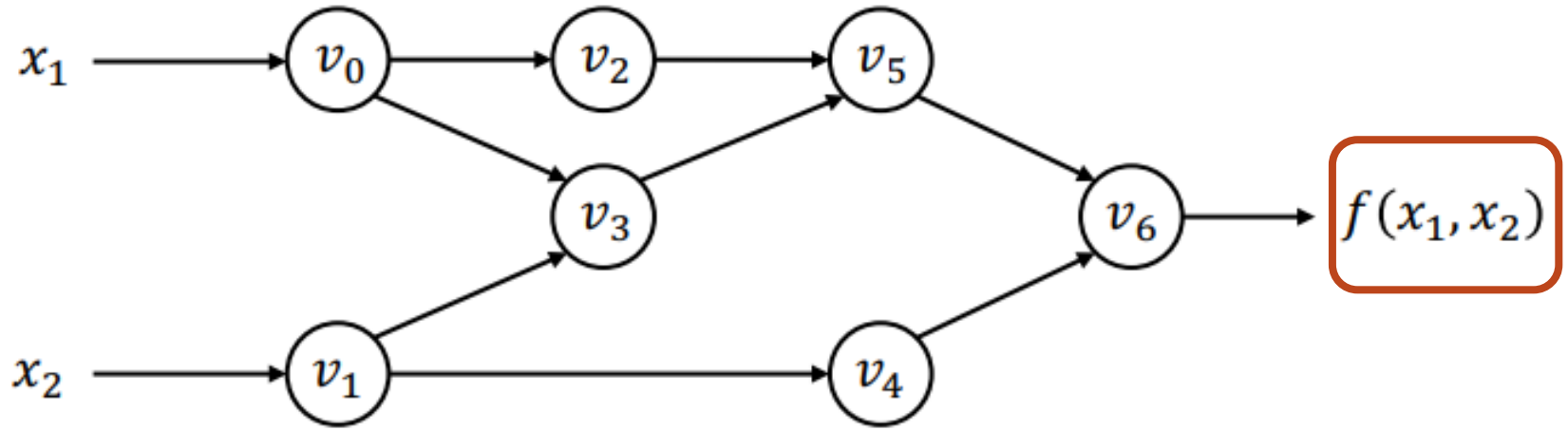
- Shape from an existing array:
  - tf.ones_like(array)
  - tf.zeros_like(array)
- Directly generate:
  - tf.eye(3)
  - tf.ones(3)
  - tf.zeros([2,5])
  - tf.reshape(tf.range(6))
  - tf.constant([1,2,3,4,5])
  - tf.constant(-1.0, shape=[2, 3])

```
Tensor("a/MatrixDiag:0", shape=(3, 3), dtype=float32)
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]

Tensor("b:0", shape=(3,), dtype=float32)
[1. 1. 1.]

Tensor("c:0", shape=(2, 5), dtype=float32)
[[0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]]

Tensor("d:0", shape=(2, 3), dtype=int32)
[[0 1 2]
 [3 4 5]]

Tensor("e:0", shape=(5,), dtype=int32)
[1 2 3 4 5]

Tensor("f:0", shape=(2, 3), dtype=float32)
[[-1. -1. -1.]
 [-1. -1. -1.]]
```

4. Run the network with feed_data

for symbolic variables

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$



4. Run the network with feed_data

for symbolic variables

```cpp
#include <cstdio>
#include <cstring>
#include <iostream>
using namespace std;

void add(int a[], int b[], int c[], int size)
{
    memset(c, 0, sizeof(c[0]));
    for(int i=0; i<size; i++)
    {
        c[i] = a[i] + b[i];
        //cout<<c[i]<<endl;
    }
}

int main()
{
    int n=5;
    int a[n] = {1,1,1,1,1};
    int b[n] = {2,2,2,2,2};
    int c[n];
    add(a, b, c, n);

    for(int i=0; i<n; i++)
    {
        cout<<c[i]<<' ';
    }
    return 0;
```

```
C:\Users\okcd0\Documents\Temp\未命名2.ex
3 3 3 3 3
--------------------------------
Process exited after 0.05206 seconds
请按任意键继续. . .
```

```python
a = tf.placeholder(tf.int32, [None], name='a')
b = tf.placeholder(tf.int32, [None], name='b')
c = tf.add(a, b)  # c = a+b is also OK

data_a = np.ones(5)
data_b = np.ones(5) * 2

with tf.Session() as sess:
    output = sess.run(
        c,
        feed_dict={
            a: data_a,
            b: data_b
        }
    )
```

- Now, you can use ‘tensorflow’ to calculate any formula you can write on paper.
- But, why we use tensorflow?

Automatic derivation

`Autograd`

tf.gradients(ys, xs)

- You can get the gradients easily from this function.

- **tf.gradients(ys, xs)**
- Constructs symbolic derivatives of sum of ys w.r.t. x in xs.
- ys and xs are each a Tensor or a list of tensors.

- $\frac{\partial(a\cdot\left(2a+\frac{1}{a}\right))}{\partial a} = 4a = 12.0$
- $\frac{\partial(a\cdot b)}{\partial b} = a = 3.0$

```python
import tensorflow as tf

a = tf.constant(3.)
b = 2 * a + 1. / a
g = tf.gradients(
    ys = a * b,
    xs = [a, b],
    name='gradients')

with tf.Session() as sess:
    print (sess.run(g))
```

```
[12.000001, 3.0]
```

How to run on GPU (optional)

- $ pip install tensorflow-gpu
- Maybe you need to setup CUDA environment.

```
import tensorflow as tf
tf_config = tf.ConfigProto()
tf_config.allow_soft_placement = True  # auto select gpu
tf_config.gpu_options.allow_growth = True  # Set growth
tf_config.gpu_options.allocator_type = 'BFC'  # Set as BFC mode
tf_config.gpu_options.visible_device_list = '0'  # use first GPU
```

- **with tf.Session(config = tf_config) as sess:**
  - **# do something.**
  - **sess.run(……)**

```
Every 1.0s: gpustat -cpu --color                                        Fri S

c1  Fri Sep 28 11:42:45 2018
[0] TITAN X (Pascal) | 44'C,   0 % | 11629 / 12189 MB | chend:python/57021(11619M)
[1] TITAN X (Pascal) | 49'C,   0 % | 11589 / 12189 MB | chend:python/57021(11579M)
[2] TITAN X (Pascal) | 49'C,   0 % | 11589 / 12189 MB | chend:python/57021(11579M)
[3] TITAN X (Pascal) | 46'C,   0 % | 11589 / 12189 MB | chend:python/57021(11579M)
```

Without tf_config

```python
import tensorflow as tf
tf_config = tf.ConfigProto()
tf_config.allow_soft_placement = True  # auto select gpu
tf_config.gpu_options.allow_growth = True  # Set growth
tf_config.gpu_options.allocator_type = 'BFC'  # Set as BFC mode
tf_config.gpu_options.visible_device_list = '0'  # use first GPU
```

tf.Session()
---
tf.Session(config=tf_config)

With tf_config

```
Every 1.0s: gpustat -cpu --color

c1  Fri Sep 28 11:41:17 2018
[0] TITAN X (Pascal) | 36'C,   0 % |   203 / 12189 MB | chend:python/54684(193M)
[1] TITAN X (Pascal) | 41'C,   0 % |    10 / 12189 MB |
[2] TITAN X (Pascal) | 41'C,   0 % |    10 / 12189 MB |
[3] TITAN X (Pascal) | 40'C,   0 % |    10 / 12189 MB |
```

How to debug?

```
a = tf.constant(3.)
b = 2 * a + 1. / a
print(b)
g = tf.gradients(
    ys = a * b,
    xs = [a, b],
    name='gradients')
for each in g:
    print (each)

with tf.Session(config=tf_config) as sess:
    print (sess.run(g))
```

How can I get the value of them?

Shape is another important point.

```
Tensor("add_2:0", shape=(), dtype=float32)
Tensor("gradients_2/AddN:0", shape=(), dtype=float32)
Tensor("gradients_2/mul_5_grad/Reshape_1:0", shape=(), dtype=float32)
[12.000001, 3.0]
```

Solution A: add it into sess.run()

```
a = tf.constant(3.)
b = 2 * a + 1. / a
print(b)
g = tf.gradients(
    ys = a * b,
    xs = [a, b],
    name='gradients')
for each in g:
    print (each)

with tf.Session(config=tf_config) as sess:
    print (sess.run([b, g]))
```

```
Tensor("add_3:0", shape=(), dtype=float32)
Tensor("gradients_3/AddN:0", shape=(), dtype=float32)
Tensor("gradients_3/mul_7_grad/Reshape_1:0", shape=(), dtype=float32)
[6.3333335, [12.000001, 3.0]]
```

Solution B: use eval()

```
with tf.Session() as sess:
    # build network
    xxxxxxxxxxxxx
```

SAME 上下两种表达等效

```
sess = tf.Session()
# build network
xxxxxxxxxxxxx
sess.close()
```

```
# the same as `with ... as sess:`
sess = tf.Session(config=tf_config)

a = tf.placeholder(tf.float32, name='a')
b = 2 * a + 1. / a
print(b.eval(session=sess, feed_dict={a: 3.0}))
g = tf.gradients(
    ys = a * b,
    xs = [a, b],
    name='gradients')
for each in g:
    print (each.eval(session=sess, feed_dict={a: 1.0}))

# the same as `with ... as sess:`
sess.close()  # remember to close
```

if feed_dict is required

```
6.33333
4.0
1.0
```

Variable

- **symbolic** => used in symbolic expression
- **non-symbolic** => has internal value
- A 'variable' is a buffer that stores a numerical value for a tensorflow variable.
- Usually used to store model parameters.

```
init_value_weight = scale * np.random.standard_normal(size=(n_input, n_output))
weight = tf.Variable(
    initial_value=init_value_weight,
    dtype=tf.float32, name='weight')

init_value_bias = tf.constant(0.0, shape=[output_dim])
bias = tf.Variable(
    initial_value=init_value_bias,
    dtype=tf.float32, name='bias')
```

Variable

How does it change?

- Var = function(Var)

How to use it for learning?

- Var' = Var – learning_rate * $\frac{\partial f(Var)}{\partial Var}$

It's too complex! (for all parameters)

- opt=tf.train.GradientDescentOptimizer(lr)
- opt.minimize(loss)

In [2]:
```
import tensorflow as tf
import numpy as np

# setup tf_config
tf_config = tf.ConfigProto()
tf_config.allow_soft_placement = True  # auto select gpu
tf_config.gpu_options.allow_growth = True  # Set growth
tf_config.gpu_options.allocator_type = 'BFC'  # Set as BFC mode
tf_config.gpu_options.visible_device_list = '0'  # use first GPU
```

In [3]:
```
init_value_weight = np.random.standard_normal(size=(1,))
weight = tf.Variable(
    initial_value=init_value_weight,
    dtype=tf.float32, name='weight')

init_value_bias = tf.constant(0.0, shape=(1,))
bias = tf.Variable(
    initial_value=init_value_bias,
    dtype=tf.float32, name='bias')
```

In [4]:
```
init = tf.global_variables_initializer()
# sess = tf.Session()
sess = tf.Session(config=tf_config)
sess.run(init)  # Variables needs init before using.
```

In [5]:
```
for i in range(5):
    weight = weight + tf.constant(1.0)
    bias = bias + tf.constant(1.0)
    y = weight * tf.constant(1.0) + bias
    weight, bias, y = sess.run([weight, bias, y])
    print y, '=', weight, '* 1 +', bias
```

```
[ 13.08324337] = [ 6.08324337] * 1 + [ 7.]
[ 15.08324337] = [ 7.08324337] * 1 + [ 8.]
[ 17.08324432] = [ 8.08324337] * 1 + [ 9.]
[ 19.08324432] = [ 9.08324337] * 1 + [ 10.]
[ 21.08324432] = [ 10.08324337] * 1 + [ 11.]
```

In [6]:
```
sess.close()  # remember to close
```

Try something

- Logistic Regression
- Jupyter notebook
- Run blocks step by step
- Get your hands dirty,
- Just try!

Define network

> 此处给出的是逻辑回归(Logistic Regression)的神经网络结构

对于输入向量x，其属于类别i的概率为：

$$P(Y=i \mid x, W, b) = softmax_i(Wx+b) = \frac{e^{W_i x + b_i}}{\sum_j e^{W_j x + b_j}}$$

模型对于输入向量x的预测结果y_pred是所有类别的预测中概率值最大的，即

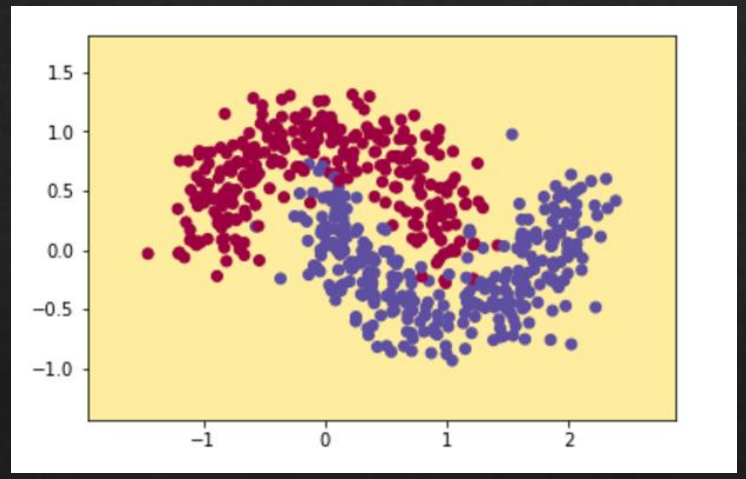
$$y_{pred} = argmax_i P(Y=i \mid x, W, b)$$

在LR模型中，需要求解的参数为权重矩阵W和偏置向量b，为了求解模型的两个参数，首先必须定义损失函数。对于上述的多类别Logistic回归，可以藉由Log似然函数作为其损失函数（注意取负）：

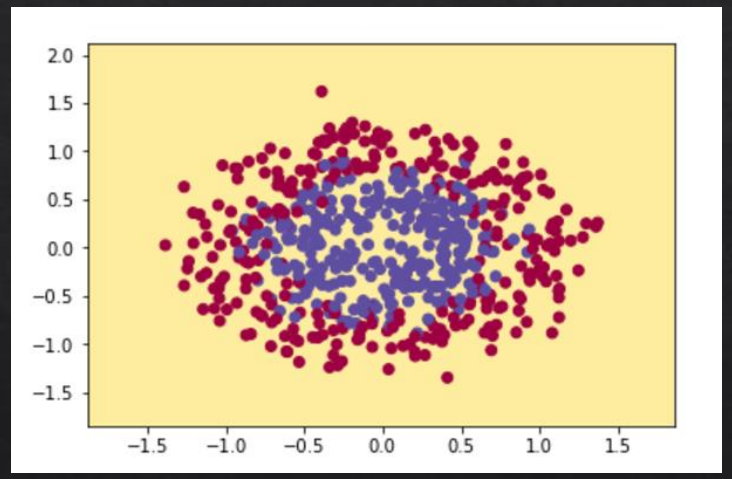
$$L(\theta=\{W,b\}, D) = \sum_{i=0}^{|D|} log\left(P\left(Y=y^{(i)} \mid x^{(i)}, W, b\right)\right)$$

Given Datasets

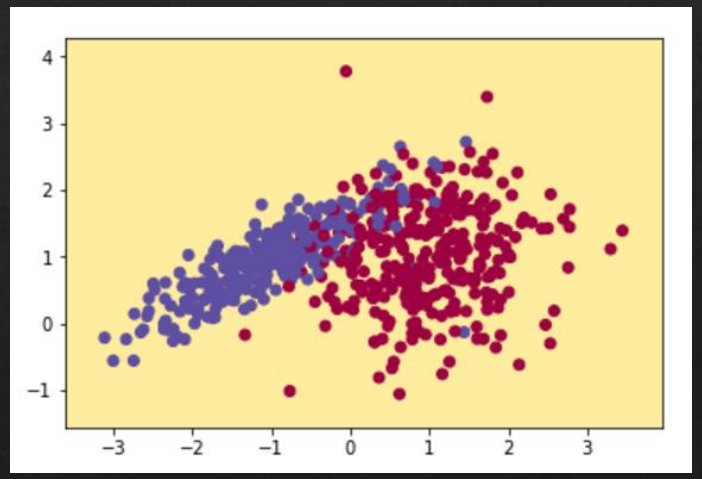
Moons



Circles



Linear



What's more

- Python 3 Cheat Sheet
  - https://perso.limsi.fr/pointal/_media/python:cours:mementopython3-english.pdf（原表）
  - https://mp.weixin.qq.com/s?__biz=MzA3MzI4MjgzMw==&mid=2650749623&idx=2&sn=c75401fc24e591da968367a146951eb4&chksm=871afec9b06d77dfe84e6221759449aa71d0c3d90888e71c059902b34ee3052e8b49731d7851&scene=0#rd（讲解 by 机器之心）

- 100 Numpy exercises.（练习与答案， Numpy 的“从入门到精通”）

  https://github.com/rougier/numpy-100/blob/master/100_Numpy_exercises_with_hint.ipynb

  https://github.com/rougier/numpy-100/blob/master/100_Numpy_exercises.ipynb

What's more

- 课程主页（搭建中）

  https://www.ai-fundamental.com

- TensorFlow Dev Summit 2018（中文）

  https://list.youku.com/albumlist/show/id_51692237.html

- 令人困惑的 Tensorflow（Google Brain 工程师亲手编写的指南）

  https://mp.weixin.qq.com/s/JVSxFFIyW4yCuV1LoIKM3g

  https://mp.weixin.qq.com/s/P8oJV1UUr0cHQ9iulcPAmw

  https://jacobbuckman.com/post/ （原文）

Thanks

Best wishes from okcd00