The Third Week Report

Lu Guorui

2018.6.10

Contents

Python Learning

1 Complier

1.1 Pycharm

The installation of Pycharm is relatively simple. Just be aware of the following points:

- Make sure you have installed pip3.
- Make sure your have set interpreter to python3.5
- There is no numpy when Pycharm is downloaded from official website, you should add it in "$Settings->Project->Projectinterpreter$".

1.2 Vim

Although Pycharm is powerful, I prefer the vim's concision and efficiency. But you are supposed to configure all the functions by yourself.

1.2.1 The steps of configuring vim [1]

[1] Main Source:

1. https://blog.csdn.net/u012450329/article/details/52539058
2. https://blog.csdn.net/hang916/article/details/79652645

1. Make sure the catalog, **/.vim/bundle**, exist. If not, creat a new one.

2. Install the Vundle:

   - The old version: **git clone https://github.com/g /.vim/bundle/Vundle.vim**
   - The new version: **git clone https://github.com/ /.vim/bundle/Vundle.vim**

   I used the old version.

3. New the **.vimrc** file

4. Add the following contents in your new file:

```
set nocompatible              " required
filetype off                  " required

" set the runtime path to include Vundle and initialize
set rtp+=~/.vim/bundle/Vundle.vim
call vundle#begin()

" alternatively, pass a path where Vundle should install plugins
" call vundle#begin('~/some/path/here')

" let Vundle manage Vundle, required
Plugin 'gmarik/Vundle.vim'
Plugin 'vim-scripts/indentpython.vim'
Plugin 'tmhedberg/SimpylFold'
"Plugin 'aralla/completor.vim'
```

```
Plugin 'scrooloose/syntastic'
Plugin 'nvie/vim-flake8'
Plugin 'jnurmine/Zenburn'
Plugin 'altercation/vim-colors-solarized'


Plugin 'tell-k/vim-autopep8'


Plugin 'scrooloose/nerdtree'

Plugin 'Xuyuanp/nerdtree-git-plugin'

"Plugin 'Lokaltog/vim-powerline'Valloric/
    YouCompleteMe'
"Plugin 'maralla/completor.vim'
"Plugin 'scrooloose/syntastic'
"Plugin 'nvie/vim-flake8'
"Plugin 'jnurmine/Zenburn'
"Plugin 'altercation/vim-colors-solarized'

"Plugin 'scrooloose/nerdtree'

"Plugin 'Xuyuanp/nerdtree-git-plugin'

"Plugin 'Lokaltog/vim-powerline'

Plugin 'Yggdroot/indentLine'


Plugin 'kien/ctrlp.vim'

Plugin 'jiangmiao/auto-pairs'


```

```

" Add all your plugins here (note older versions of Vundle used Bundle instead of Plugin)

" All of your Plugins must be added before the following line
call vundle#end()            " required

filetype plugin indent on    " required

let g:completor_python_binary = '/usr/bin/python3.5'
let Tlist_Auto_Highlight_Tag=1
let Tlist_Auto_Open=1
let Tlist_Auto_Update=1
let Tlist_Display_Tag_Scope=1
let Tlist_Exit_OnlyWindow=1
let Tlist_Enable_Dold_Column=1
let Tlist_File_Fold_Auto_Close=1
let Tlist_Show_One_File=1
let Tlist_Use_Right_Window=1
let Tlist_Use_SingleClick=1
nnoremap <silent> <F8> :TlistToggle<CR>

filetype plugin on
autocmd FileType python set omnifunc=pythoncomplete#Complete
autocmd FileType javascrpt set omnifunc=javascriptcomplete#CompleteJS
autocmd FileType html set omnifunc=htmlcomplete#CompleteTags
autocmd FileType css set omnifunc=csscomplete#CompleteCSS
autocmd FileType xml set omnifunc=
xmlcomplete#CompleteTags
autocmd FileType php set omnifunc=phpcomplete#CompletePHP
autocmd FileType c set omnifunc=ccomplete#Complete

let g:pydiction_location='~/.vim/tools/pydiction/complete-dict'
set autoindent
set expandtab
set tabstop=4
set shiftwidth=4
set number
set lines=35 columns=118


set number "
set nowrap     "
set showmatch     "
set scrolloff=3          "3"
set encoding=utf-8   "
set fenc=utf-8        "
set mouse=v          "
set hlsearch          "

let python_highlight_all=1
syntax on     "


hi BadWhitespace guifg=gray guibg=red ctermfg=gray ctermbg=red
au BufRead,BufNewFile *.py,*.pyw,*.c,*.h
\ set tabstop=4   "tab
\ set softtabstop=4
\ set shiftwidth=4
\ set textwidth=79  "
"\ set expandtab      "tab
\ set autoindent      "
\ set fileformat=unix  "


map <F5> :call RunPython()<CR>
func! RunPython()
    exec "W"
    if &filetype == 'python'
        exec "!time python2.7 %"
    endif
endfunc

"split navigations
nnoremap <C-J> <C-W><C-J>
nnoremap <C-K> <C-W><C-K>
nnoremap <C-L> <C-W><C-L>
nnoremap <C-H> <C-W><C-H>

set foldmethod=indent
set foldlevel=99

let g:SimpylFold_docstring_preview=1



au BufNewFile,BufRead *.js, *.html, *.css
\ set tabstop=2
\ set softtabstop=2
\ set shiftwidth=2


au BufRead,BufNewFile *.py,*.pyw,*.c,*.h
    match BadWhitespace /\s\+$/
```

```


"python with virtualenv support
"py << EOF
"import os
"import sys
"if 'VIRTUAL_ENV' in os.environ:
"   project_base_dir = os.environ['VIRTUAL_ENV']
"   activate_this = os.path.join(project_base_dir, 'bin/activate_this.py')
"   execfile(activate_this, dict(__file__=activate_this))
"EOF


if has('gui_running')
  set background=dark
  colorscheme solarized
else
  colorscheme zenburn
endif

map <C-n> :NERDTreeToggle<CR>

hi MatchParen ctermbg=DarkRed guibg=lightblue


autocmd FileType python noremap <buffer> <F7> :call Autopep8()<CR>


map <F8> :call FormartSrc()<CR>
```

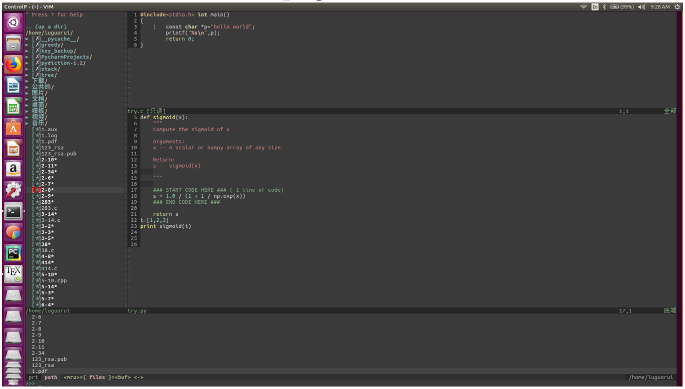
```

"FormartSrc()
func FormartSrc()
exec "w"
if &filetype == 'c'
exec "!astyle --style=ansi --one-line=keep-statements -a --suffix=none %"
elseif &filetype == 'cpp' || &filetype == 'hpp'
exec "r !astyle --style=ansi --one-line=keep-statements -a --suffix=none %> /dev/null 2>&1"
elseif &filetype == 'perl'
exec "!astyle --style=gnu --suffix=none %"
elseif &filetype == 'py'||&filetype == 'python'
exec "r !autopep8 -i --aggressive %"
elseif &filetype == 'java'
exec "!astyle --style=java --suffix=none %"
elseif &filetype == 'jsp'
exec "!astyle --style=gnu --suffix=none %"
elseif &filetype == 'xml'
exec "!astyle --style=gnu --suffix=none %"
endif
exec "e! %"
endfunc
"FormartSrc
```

5. After step4, my vim gives an error whic is roughly said that it can't find the Zenburn. So we should copy the folder from **/.vim/bundle** to **/usr/share/vin**

6. Then you can open the vim and input:"**:PluginInstall**". But this command failed on my computer. Unfortunately, there is no method that can solve my problem, and yet I found a command "**vim +PluginInstall +qall**" has the same effect as "**:PluginInstall**" in vim. What's more, I saw what the log of errors said was "**Permission denied**", so tried to input "**sudo vim +PluginInstall +qall**" and succeed.

Let's have a look of what my vim looks like after configuration through the figure 1:

09-28-03.png 09-28-03.bb



```
ControlP - (~) - VIM
" Press ? for help

.. (up a dir)
/home/luguorui/
▸ [✗]__pycache__/
▸ [✗]greedy/
▸ [✗]key_backup/
▸ [✗]PycharmProjects/
▸ [✗]pydiction-1.2/
▸ [✗]stack/
▸ [✗]tree/
▸ 下载/
▸ 公共的/
▸ 图片/
▸ 文档/
▸ 桌面/
▸ 模板/
▸ 视频/
▸ 音乐/
  [★]1.aux
  [★]1.log
  [★]1.pdf
  [★]123_rsa
  [★]123_rsa.pub
  [★]2-10*
  [★]2-11*
  [★]2-34*
  [★]2-6*
  [★]2-7*
  [★]2-8*
  [★]2-9*
  [★]283*
  [★]283.c
  [★]3-14*
  [★]3-14.c
  [★]3-2*
  [★]3-3*
  [★]3-5*
  [★]38*
  [★]38.c
  [★]4-8*
  [★]414*
  [★]414.c
  [★]5-10*
  [★]5-10.cpp
  [★]5-14*
  [★]5-3*
  [★]5-7*
  [★]6-4*
/home/luguorui
```

```c
#include<stdio.h> int main()
{
        const char *p="hello world";
        printf("%s\n",p);
        return 0;
}
```

```
try.c [只读]                                   1,1        全部
```

```python
def sigmoid(x):
    """
    Compute the sigmoid of x

    Arguments:
    x -- A scalar or numpy array of any size

    Return:
    s -- sigmoid(x)

    """

    ### START CODE HERE ### ( 1 line of code)
    s = 1.0 / (1 + 1 / np.exp(x))
    ### END CODE HERE ###

    return s
t=[1,2,3]
print sigmoid(t)
```

```
try.py                                         17,1       底端
  2-6
  2-7
  2-8
  2-9
  2-10
  2-11
  2-34
  123_rsa.pub
  123_rsa
  1.pdf
 prt  path  <mru>={ files }=<buf> <->                 /home/luguorui
>>>
```

Figure 1: My Vim

Practical aspects of DeepLearning

1 Normalizing inputs

1.1 Explanation

The essence of normalization is a kind of linear transformation which compresses and parallel moves the data. Linear transformation has many good properties, one of which is that it won't change the relative order of data. These properties ensure our data won't be invalidated while we transform them.

We can also have a intuitive view of normalization from geometrical point of view. Let's look at the pictures below.

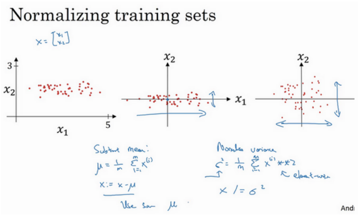

Normalizing training sets
x_1
x_2
Subtract mean:
$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$
$x := x - \mu$
Normalize variance
$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} ** 2$
element-wise
$x /= \sigma^2$

Figure 2: Data

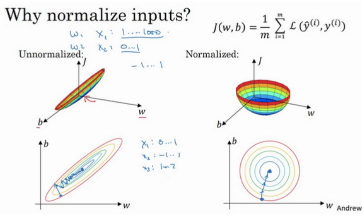

Why normalize inputs?
$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$
Unnormalized:
Normalized:
J
w
b
Andrew

Figure 3: Const function

Obviously, normalization makes data more concentrated.

1.2 Steps of normalizing inputs

1. $\mu = \frac{1}{m} \sum_{i=1}^{m} x^{(i)}$

2. $\sigma^2 = \frac{1}{m} \sum_{i=1}^{m} \left(x^{(i)}\right)^2$

As once the data has been determined, μ and σ are both constants , so it can be seen as linear transformation, which can makes the process more efficient.

2 Mini-batch gradient descent

We can divided our training set into many smaller mini-bitch while training the neural network, which can makes the process more efficienct.

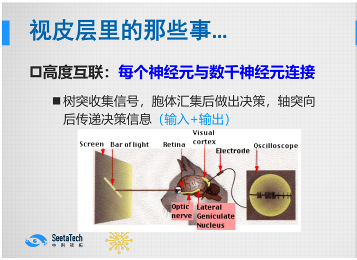
Mini-batch gradient descent allows our neural network carry out more gradient decents while it can only count one gradient descent if we don't use mini-batch. Considering the way that computer store, we usually set the mini-batch's size to be an integer multiple of 2.

Some thoughts from SeetaTech's videos

Apart from deeplearning.ai, I also watched some videos produced by SeetaTech, which gave me a deeper understanding.

1 How can we see an object

There are many nerve cells in our brains. From the experiment in figure 4 we can know that each of them speciallized in judging a specific condition and transmit the information to the cell which judges a more complex condition.



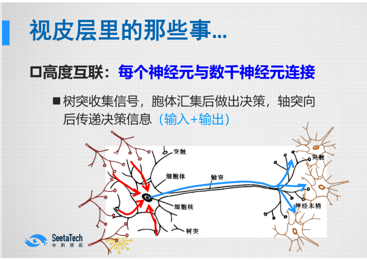
视皮层里的那些事...

□高度互联：每个神经元与数千神经元连接

■树突收集信号，胞体汇集后做出决策，轴突向后传递决策信息（输入+输出）

Screen
Bar of light
Retina
Visual cortex
Electrode
Oscilloscope
Optic nerve
Lateral Geniculate Nucleus
SeetaTech
中 科 视 拓

Figure 4: The experiment



视皮层里的那些事...

□高度互联：每个神经元与数千神经元连接

■树突收集信号，胞体汇集后做出决策，轴突向后传递决策信息（输入+输出）

突触
细胞体
轴突
细胞核
树突
突触
神经末梢
SeetaTech
中 科 视 拓

Figure 5: How nerve cells work

2 How can we imitate our brains

In Deeplearning, every neuron represent a function which count the prossibility of a specific incident happening and they transmit the results to the next neuron which is used to count a more complex prossibility. By repeat the step above we can judge what we want precisely.

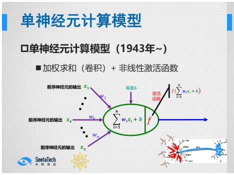


Figure 6: Model

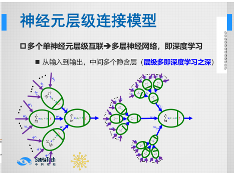

神经元层级连接模型
□多个单神经元层级互联→多层神经网络，即深度学习
■从输入到输出，中间多个隐含层（层级多即深度学习之深）
SeetaTech 中科视拓

Figure 7: Neural net work

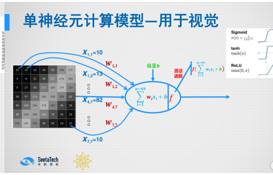


Figure 8: Applications in the vision field

Learning summary

This week I spend a lot of in configuring complier and studying python grammar. Now I can understand a little more about the codes in videos and homework. And videos from SeetaTech also helps a lot. I'll continue to watch it.

In this course, Practical aspects of DeepLearning, the knowledges points are much more mathematical, which makes it harder for me to understand. I'll spend more time in practicing instead of watching the next week so that I can grasp the technology expertly.