# 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. 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
```

[1]Main Source:

https://blog.csdn.net/u012450329/article/details/52539058

https://blog.csdn.net/hang916/article/details/79652645

```
" 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'
```

```
43
44
45 Plugin 'kien/ctrlp.vim'
46
47 Plugin 'jiangmiao/auto-pairs'
48
49
50
51 " Add all your plugins here (note older
       versions of Vundle used Bundle instead
       of Plugin)
52
53 " All of your Plugins must be added before
       the following line
54 call vundle#end()            " required
55
56 filetype plugin indent on    " required
57
58 let g:completor_python_binary = '/usr/bin/
       python3.5'
59 let Tlist_Auto_Highlight_Tag=1
60 let Tlist_Auto_Open=1
61 let Tlist_Auto_Update=1
62 let Tlist_Display_Tag_Scope=1
63 let Tlist_Exit_OnlyWindow=1
64 let Tlist_Enable_Dold_Column=1
65 let Tlist_File_Fold_Auto_Close=1
66 let Tlist_Show_One_File=1
67 let Tlist_Use_Right_Window=1
68 let Tlist_Use_SingleClick=1
69 nnoremap <silent> <F8> :TlistToggle<CR>
70
71 filetype plugin on
72 autocmd FileType python set omnifunc=
       pythoncomplete#Complete
```

```
73 autocmd FileType javascrpt set omnifunc=
       javascriptcomplete#CompleteJS
74 autocmd FileType html set omnifunc=
       htmlcomplete#CompleteTags
75 autocmd FileType css set omnifunc=
       csscomplete#CompleteCSS
76 autocmd FileType xml set omnifunc=
       xmlcomplete#CompleteTags
77 autocmd FileType php set omnifunc=
       phpcomplete#CompletePHP
78 autocmd FileType c set omnifunc=ccomplete#
       Complete
79
80 let g:pydiction_location='~/.vim/tools/
       pydiction/complete-dict'
81 set autoindent
82 set expandtab
83 set tabstop=4
84 set shiftwidth=4
85 set number
86 set lines=35 columns=118
87
88
89 set number "
90 set nowrap     "
91 set showmatch     "
92 set scrolloff=3         "3"
93 set encoding=utf-8  "
94 set fenc=utf-8       "
95 set mouse=v          "
96 set hlsearch         "
97
98 let python_highlight_all=1
99 syntax on     "
100
```

```
101
102 hi BadWhitespace guifg=gray guibg=red
      ctermfg=gray ctermbg=red
103 au BufRead,BufNewFile *.py,*.pyw,*.c,*.h
104 \ set tabstop=4    "tab
105 \ set softtabstop=4
106 \ set shiftwidth=4
107 \ set textwidth=79   "
108 "\ set expandtab        "tab
109 \ set autoindent       "
110 \ set fileformat=unix   "
111
112
113 map <F5> :call RunPython()<CR>
114 func! RunPython()
115     exec "W"
116     if &filetype == 'python'
117        exec "!time python2.7 %"
118     endif
119 endfunc
120
121 "split navigations
122 nnoremap <C-J> <C-W><C-J>
123 nnoremap <C-K> <C-W><C-K>
124 nnoremap <C-L> <C-W><C-L>
125 nnoremap <C-H> <C-W><C-H>
126
127 set foldmethod=indent
128 set foldlevel=99
129
130 let g:SimpylFold_docstring_preview=1
131
132
133
134 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
```

```
164
165
166 autocmd FileType python noremap <buffer> <
    F7> :call Autopep8()<CR>
167
168
169 map <F8> :call FormartSrc()<CR>
170
171 "FormartSrc()
172 func FormartSrc()
173 exec "w"
174 if &filetype == 'c'
175 exec "!astyle --style=ansi --one-line=keep
    -statements -a --suffix=none %"
176 elseif &filetype == 'cpp' || &filetype ==
    'hpp'
177 exec "r !astyle --style=ansi --one-line=
    keep-statements -a --suffix=none %> /
    dev/null 2>&1"
178 elseif &filetype == 'perl'
179 exec "!astyle --style=gnu --suffix=none %"
180 elseif &filetype == 'py'||&filetype == '
    python'
181 exec "r !autopep8 -i --aggressive %"
182 elseif &filetype == 'java'
183 exec "!astyle --style=java --suffix=none
    %"
184 elseif &filetype == 'jsp'
185 exec "!astyle --style=gnu --suffix=none %"
186 elseif &filetype == 'xml'
187 exec "!astyle --style=gnu --suffix=none %"
188 endif
189 exec "e! %"
190 endfunc
191 "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:

# 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

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

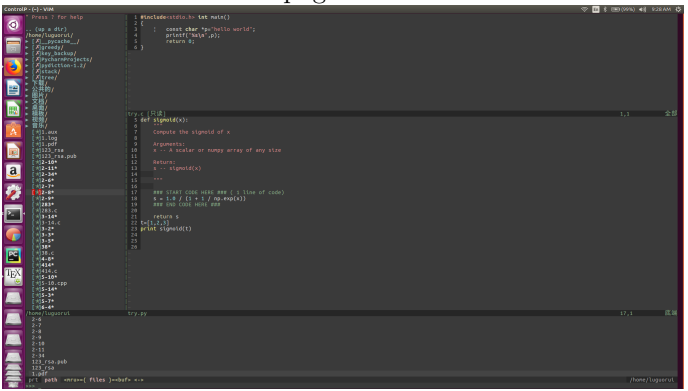

Figure 1: My Vim

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.

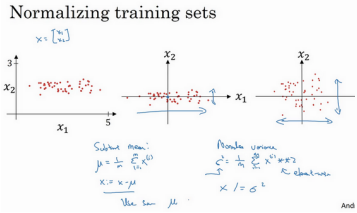


Figure 2: Data

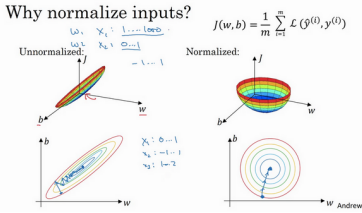


Figure 3: Const function

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, $\mu$ and $\sigma$ 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.

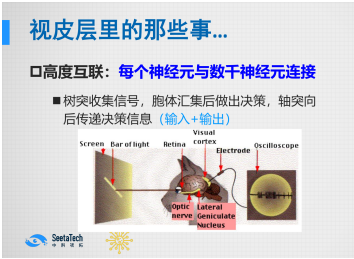


Figure 4: The experiment

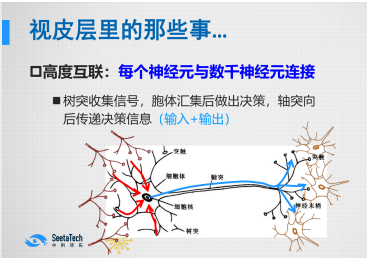


Figure 5: How nerve cells work

## 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.

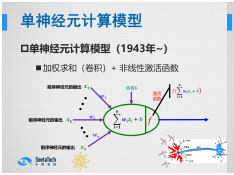


Figure 6: Model

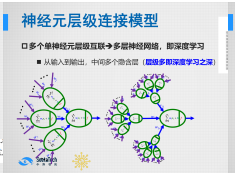


Figure 7: Neural net work

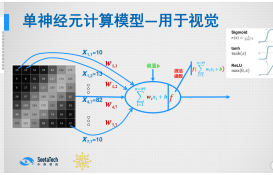


Figure 8: Applications in the vision field

# 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.