在 notebook 里设置超参，参考以下选项

```
Learning_Rate = 0.0005        #please be greater than 0
Epoch = 50                    #please be greater than 0
Batch_Size = 32               #please be greater than 0
Dropout_Rate = 0.3            #please be 0 to 1 (inclusive)
Weight_Decay = 0              #please be 0 to 1 (inclusive)


Regularizer = None            #please be L1 or L2 in string
Batch_Normalization = True    #please be boolean
OPtimization = "adam"         #please be optimization in string
#Available optimization include:
# Adam, AdaDelta, RMSProp, AdaGrad, Nesterov, Momentum


Training_Rate = 1             #please be 0 to 1 (inclusive)
Cross_Validate_Rate = 0       #please be 0 to 1 (inclusive)
Test_Rate = 0                 #please be 0 to 1 (inclusive)


Plot_Loss = True              #please be boolean
Plot_Accuracy = True          #please be boolean
Print_Info = True             #please be boolean
Print_At = 1                  #please be int and be greater than 0
```

Notebook 内容（run 模块）：

数据处理，split

(train_X, train_Y, cv_X, cv_Y, test_X, test_Y) = data

初始化 model = MLP（）

传入 drop ratio， regularizer object， normalizier object，optimizer object

可以以初始化参数或者 set 来传入

用 model.add_layer() 来构筑整个网络结构， 四个参数 in out activation keep_prob

有后面两个是因为最后一层的 activation 和 dropout rate 是不一样的，所以在加层时需要说明

model.fit(data，label，epoch，learning_rate， batchsize)

model.predict(test_x)

model.evaluate (test_X, test_Y)

这个还没实现

model.plot(config.Plot_Loss, config.Plot_Accuracy)

MLP 模块

需要存储的变量:

Batch 数据 （在 fit 时填入）

Batch size

m 数据量　用于计算

dims 每一层的 dimension（list）

learning rate

epoch

layers list of layer object

optimizer

keep rate for drop out

regularizer

batch normalizer

cost

方法:

加层 add_layer(in, out , acti, drop):

实例化一个 layer

给 layer Set activation

Set batch normalizer

Set dropout

Set optimiser

都统一继承模型的就行，这些都是每层独有的

Regularizer 不需要每层都有，在这里不用传给 layer

把 out 加进 dims 中

Layer 加进 layers 中

Reset regularizer：

每一个 epoch 需要 call 一次来清空之前的 regularizer loss

forward(input, mode = True)：

重置 regularizer

逐层 forward，需要传入 input， train_mode, 模型 regularizer

Backward（）：

逐层跑 layer 的 back，带 regularizer 因为要算 regularizer 的 loss

Update（）：

逐层按模型 lr update params

fit（）：

shuffle data

for every epoch

for every batch：

run forward，record loss， accuracy， run backward， update

计算平均 loss，acc

predict（）：在预测集 run forward

evaluate（）：预测然后计算 acc

Layer 模块：

Attribute：

In

Out

激活函数 object

Batchnormalizer 对象

Optimizer 对象

m 数据量

z linear combination

z_norm　normalized z

a 激活值

a_dropout dropout 之后的激活值

keep_rate

input

方法

一些 set 来继承 model 中的对象，set dropout activation， batchnormalizer， optimizer

Forward（input， train_mode, regularizer）： 因为 regularizer 需要记录 W

此时给 m，input 赋值

计算 z

根据情况 normalize z

Activate a

如果 training：需要 regularizer forward，dropout forward

　　Return a_drop

Else： return a

Backward

Update (lr)：optimizer update，batchnormalizer update