在notebook里设置超参，可以从这些里面选

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（）：