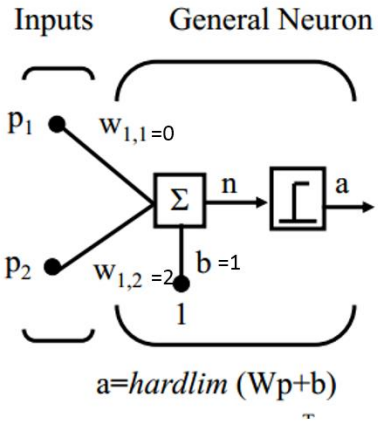
HW1 Report

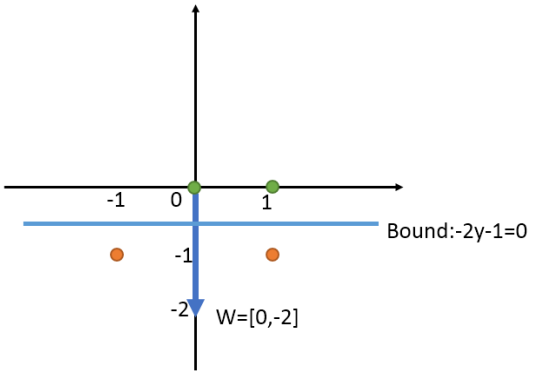
姓名：缪佳宇

学号：515021910492

1.

(1.1) 设计的求解这个问题的单神经元感知机如下图所示





(1.2) a = hardlim($w^T p$+ b) = hardlim($w_{1,1}$*p_1+ $w_{1,2}$ p_2) + b

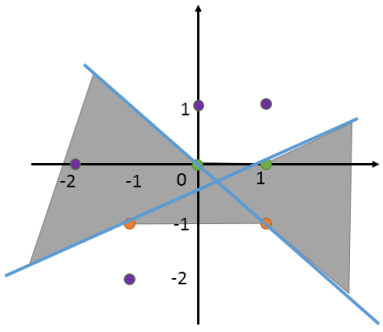
P1: a = hardlim(-2*(-1)-1) = hardlim(1)=1
P2: a = hardlim(-2*(-1)-1) = hardlim(1)=1
P3: a = hardlim(-2*0-1)= hardlim(-1)=0
P4: a = hardlim(-2*0-1)= hardlim(-1)=0

验证结果符合

(1.3)

P5: a = hardlim(-2*0-1) = hardlim(-1)=0
P6: a = hardlim(-2*1-1) = hardlim(-3)=0
P7: a = hardlim(-2*1-1)= hardlim(-3)=0
P8: a = hardlim(-2*(-2)-1)= hardlim(3)=1

(1.4)



P5-p8 为紫色的点。

分界线的范围我已经用灰色区域表示出来了，由于权值向量 w 中 $w_2<0$，所以在灰色区域上方的点计算结果一定小于 0，在灰色区域下方的点一定大于 0，而在灰色区域中间的点则根据分界线的取值而定。

所以 P5 的分类是与 W,b 的选择有关的，而 P6，P7，P8 是无关的。

(1.5)

感知机规则：

Unified Learning Rule

If $t = 1$ and $a = 0$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}$

If $t = 0$ and $a = 1$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}$

If $t = a$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old}$

$$e = t - a$$

If $e = 1$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}$

If $e = -1$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}$

If $e = 0$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old}$

$$ {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + e\mathbf{p} = {}_1\mathbf{w}^{old} + (t-a)\mathbf{p}$$

$$b^{new} = b^{old} + e$$

因为感知机规则是收敛的，所以设计的算法主要是一个循环，不断的应用学习规则直到四个点的分类都不变。

程序主要部分截图和结果：

```
20      while count!=3:
21          e = labels[i]-hardlim(w.dot(node[i])+b) #calculate e
22          if e==0:
23              count += 1
24          else:
25              count = 0
26          w = w + e * node[i]
27          b = b + e
28          i = (i+1) % 4
29          print("w1=%d, w2=%d, b=%d"%(w[0],w[1],b))
```

```
/home/mg2015started/anaconda2/envs/py35_env/bin/python /home/mg2015started/4j/HW1/ques1.py
w1=0, w2=0, b=0
w1=0, w2=0, b=0
w1=0, w2=0, b=-1
w1=0, w2=0, b=-1
w1=1, w2=-1, b=0
w1=1, w2=-1, b=0
w1=1, w2=-1, b=-1
w1=0, w2=-1, b=-2
w1=1, w2=-2, b=-1
w1=1, w2=-2, b=-1
w1=1, w2=-2, b=-1
w1=0, w2=-2, b=-2
w1=0, w2=-2, b=-2
w1=0, w2=-2, b=-2
w1=0, w2=-2, b=-2

Process finished with exit code 0
```

另外我注意到检验点的顺序不同，最后得到的分界线也是不同的，但是都是正确的分界

线。

新的分界线：-2y-2=0 即-y-1=0

P5: a = hardlim(-1*0-1) = hardlim(-1)=0

P6: a = hardlim(-1*1-1) = hardlim(-2)=0

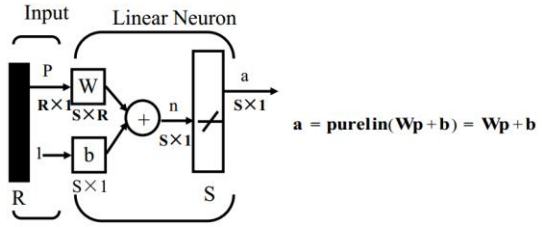
P7: a = hardlim(-1*1-1)= hardlim(-2)=0

P8: a = hardlim(-1*(-2)-1)= hardlim(1)=1

2.

我用 Numpy 生成了两类点，第一类在[-1:3，-1:3]的范围内，第二类在[-5:-1,-5:-1]的范围内，都是均匀分布，为了三次实验采用统一的数据，我新建了 gen_data.py 生成了 Numpy 数组并且保存到 npz 文件中。

ADALINE 网络和 LMS 算法：



a=purelin(Wp+b)

$a_i = purelin(n_i) = purelin({}_i\mathbf{w}^T\mathbf{p} + b_i) = {}_i\mathbf{w}^T\mathbf{p} + b_i$ $\quad {}_i\mathbf{w} = \begin{bmatrix} w_{i,1} \\ w_{i,2} \\ \vdots \\ w_{i,R} \end{bmatrix}$

LMS Learning Rule

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla F(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_k}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + 2\alpha e(k)\mathbf{z}(k)$$

$${}_1\mathbf{w}(k+1) = {}_1\mathbf{w}(k) + 2\alpha e(k)\mathbf{p}(k)$$

$$b(k+1) = b(k) + 2\alpha e(k)$$

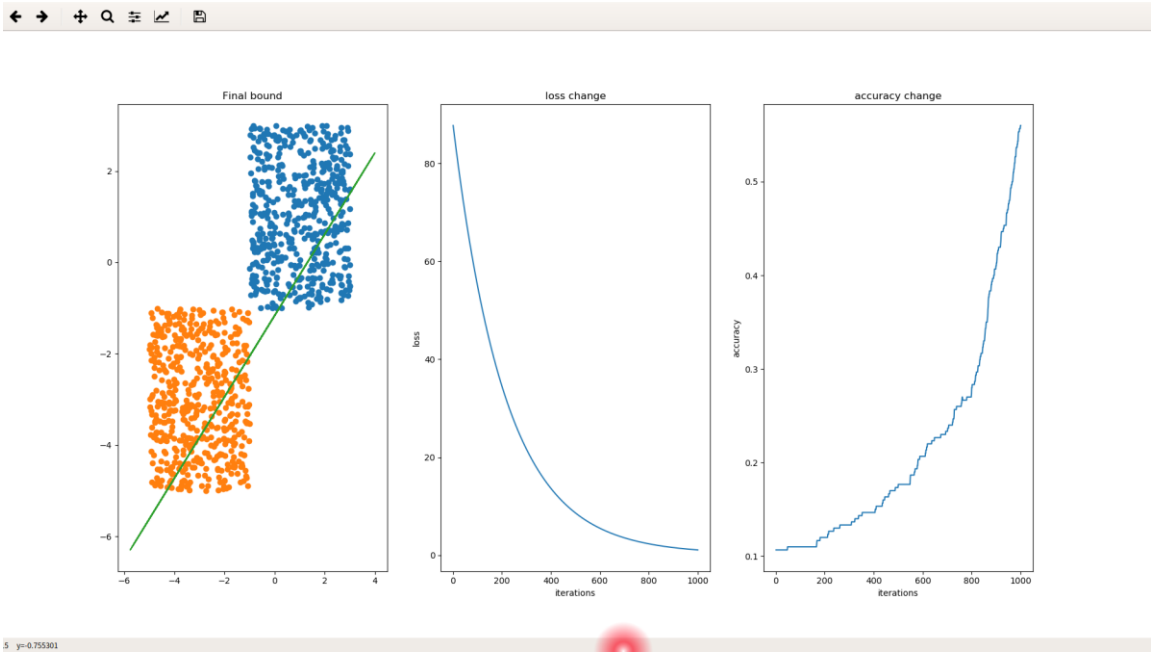
用 tensorflow shuffle 了数据然后分出训练集和测试集，然后实现了 LMS 算法并用不同的学习率 a 来进行实验，程序主要部分和测试结果如下。

```
model_output=tf.matmul(X,W)+B
loss = tf.reduce_mean(tf.square(Y-model_output))
train_op = tf.train.GradientDescentOptimizer(0.1).minimize(loss)  # construct optimizer
accuracy = tf.reduce_mean(tf.cast(tf.equal(tf.sign(model_output), Y),tf.float32)) #construct accuracy
loss_array = []
acc_array = []
with tf.Session() as sess:
    with tf.device("/gpu:0"):
        tf.global_variables_initializer().run()
        for i in range(iteration):
            #print(i)
            _w = sess.run(W)
            _b = sess.run(B)
            sess.run(train_op, feed_dict={X: train[:,:2], Y: train[:,2:]})
            temp_loss = sess.run(loss, feed_dict={X: train[:,:2], Y: train[:,2:]})
            acc = sess.run(accuracy, feed_dict={X: test[:,:2], Y: test[:,2:]})
            print(temp_loss, acc)#, _w, _b)
            loss_array.append(temp_loss)
            acc_array.append(acc)
```

Iterations=1000

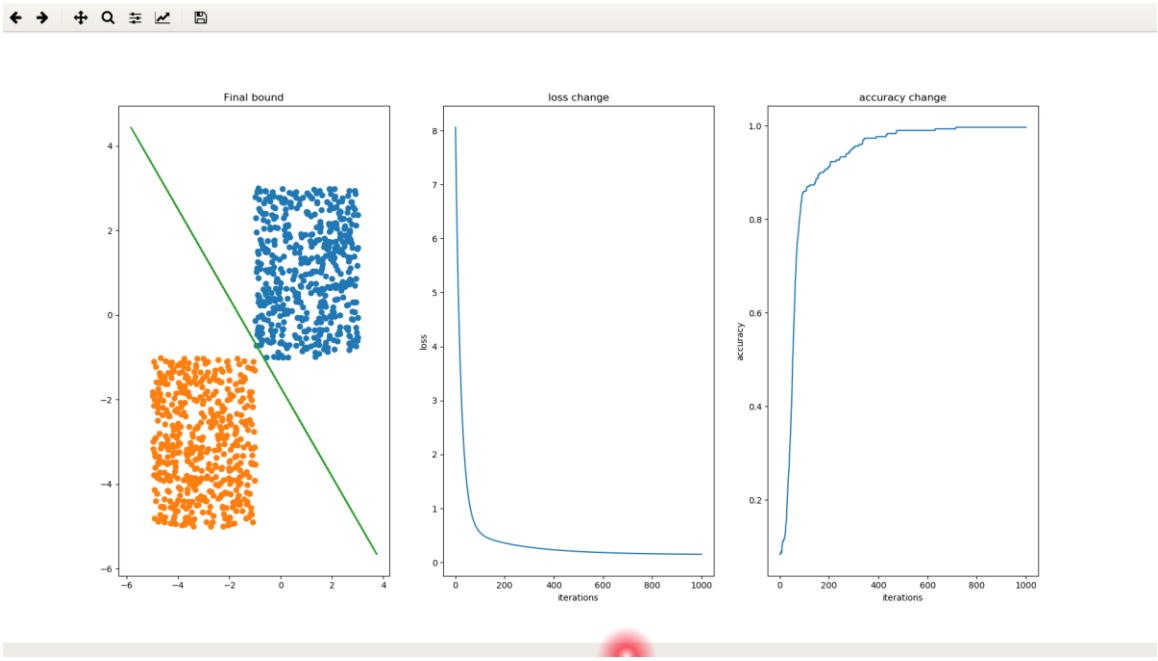
三张图分别为分类图和训练出的分界线、误差(损失)随着 iteration 的变化曲线、准确度 acc 随着 iteration 的变化曲线。

a=0.0001:



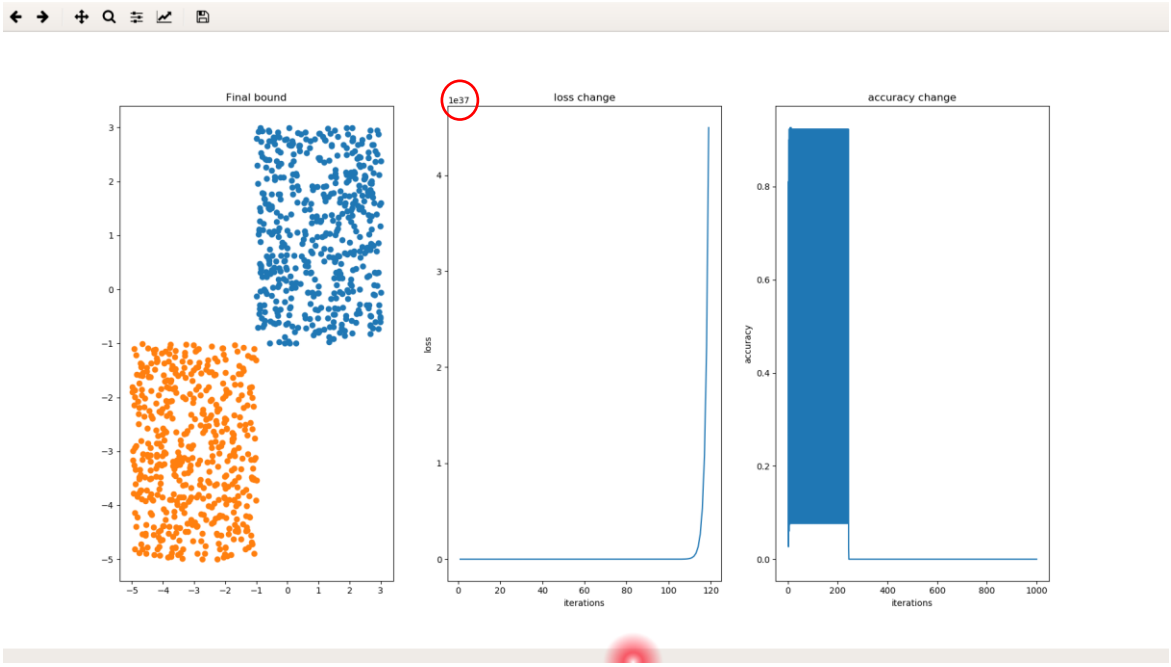
可以看到 1000 iteration 虽然误差不断在下降，但最后还没有收敛完全，accuracy 才达到 0.6 左右，训练出的分界线也不是很完美。

a=0.001:



可以看到模型收敛很快，大概 600iterations 就大概收敛了，accuracy 也达到了 1.0。分界线也十分合理。模型的效果不错。

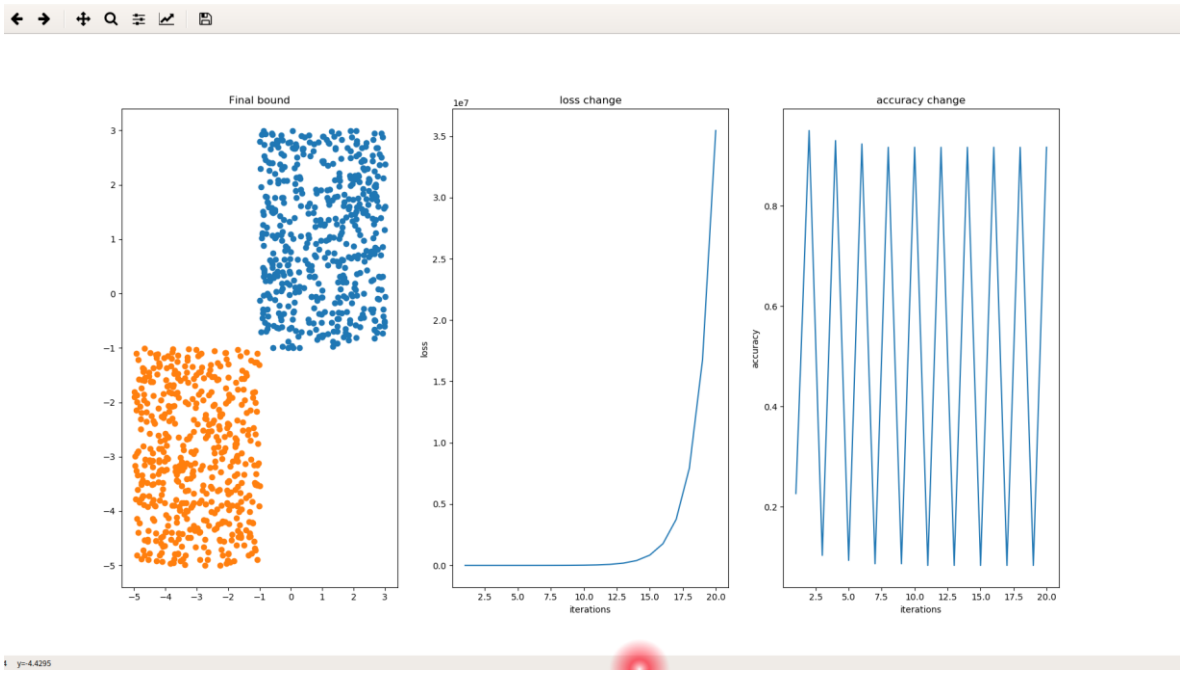
a=0.1:



可以看到训练过程十分不稳定，最后 w,b 都变成了 nan 无法计算，accuracy 感觉十分不稳定，分界线也没有画出来。

我把 iterations 调整到 20（下图）看细节，可以看到训练过了几步之后 w,b 都变成了 nan，而 accuracy 则是在不断的抖动之中。

原因就是学习率太大导致模型无法达到全局最小点从而在不断的来回波动之中。



综上，学习率 a 为 0.001 时比较合适，过小会收敛很慢，过大抖动很大，模型无法收敛。