感知机代码说明

作者：蔡中恒
2018.06.03

一、原理讲解

详见李航《统计学习方法》第 2 章感知机。

二、原型代码设计思路

本次使用 matlab 来写感知机的算法原型代码，设计思路如下：

1、 使用 matlab 自带函数来产生二维平面上均匀分布的随机数。
2、 在二维平面上随意画一条直线 1，将第 1 步里面产生的随机数划分为两个区域，将这两个区域的随机数分别标志为 0 和 1。这个步骤的目的是确保数据一定线性可分。
3、 编写感知机代码，采用自动搜索的方式，让代码自动搜索到新的直线 2，将随机数划分为两个区域。如果代码无错，根据感知机理论，我们一定可以找到一条直线满足需求。
4、 对比直线 1 和直线 2，观察两条直线的差异。

三、代码讲解

代码名为 perceptron_demo.m。

首先进行初始定义，定义保证线性可分基准直线的 w 和 b 系数，然后将感知机的 w 和 b 系数初始化都为 0，然后定义更新步长为 1e-3。接下来定义数据范围是±100，一共 90 个二位数据，同时初始化 lost_func 的值，Sum_lost_func 的值初始化为 1，目的是为了让代码运行到感知机的 while 循环中。

```
13      %% setting
14 -    base_coe_w    = [3,8];
15 -    base_coe_b    = -5;
16 -    est_coe_w     = [0,0];
17 -    est_coe_b     = 0;
18 -    upd_mu        = 1e-3;
19
20 -    plot_en       = 1;
21 -    uni_scale     = 100;
22 -    line_vec      = (-uni_scale:0.01:uni_scale).';
23 -    data_len      = 90;
24 -    lost_func     = zeros(data_len,1);
25 -    sum_lost_func = 1;
```

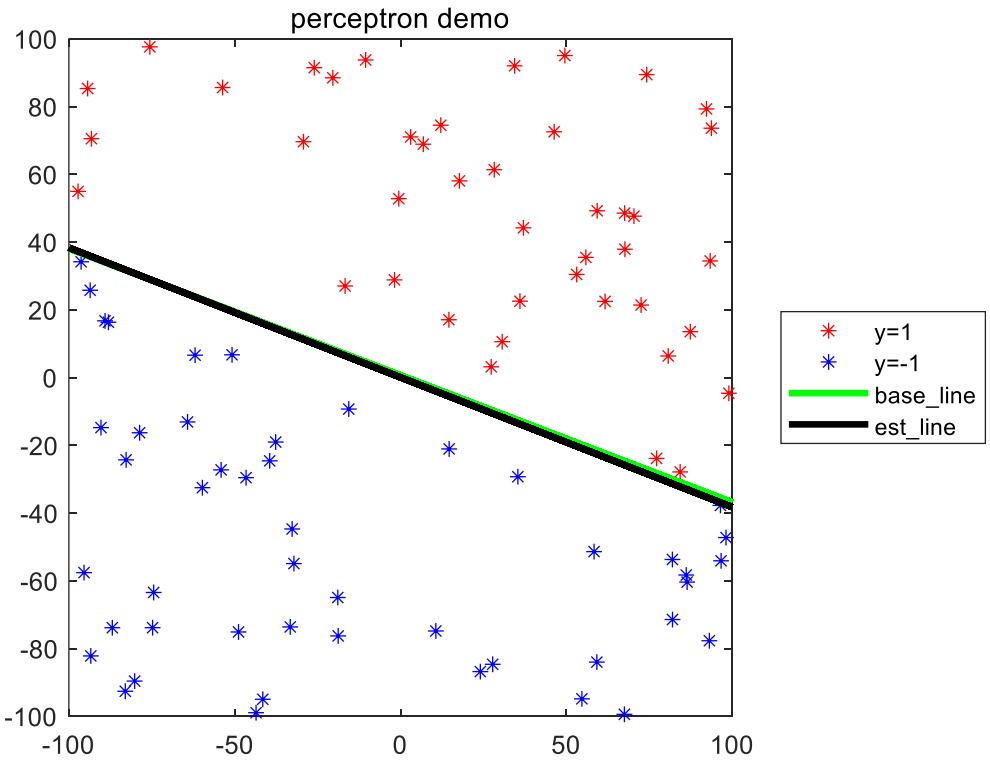
接下来使用 matlab 自带函数，产生±100 以内均匀分布的随机数，且使用基准直线，将其划分为两个区域，分别赋值+1 和-1，保存在 test_data_y 中。

```
27      %% creating test data
28 —    test_data_x1 = unifrnd(-uni_scale,uni_scale,[data_len,1]);
29 —    test_data_x2 = unifrnd(-uni_scale,uni_scale,[data_len,1]);
30
31      %{...%}
42
43      %% using a baseline to divide the data into two parts.
44 —    test_data_y = base_coe_w(1)*test_data_x1+base_coe_w(2)*test_data_x2+base_coe_b;
45 —    test_data_y = 2*(double(test_data_y>0)-1/2);
46
47      %{...%}
```

最后编写梯度下降法的代码，使用一个 while 循环和一个 for 循环，对测试数据进行遍历验证，一旦发现有不满足分类标准的数据，就更新 est_w 和 est_b，直到所有数据的 lost_func 都大于 0。

```
70      %% using perceptron method to find the estimation line
71 —    while(sum_lost_func>0)
72 —        for idx=1:data_len
73 —            lost_func(idx) = test_data_y(idx)*(est_coe_w(1)*test_data_x1(idx)+est_coe_w(2)*test_data_x2(idx)+est_coe_b);
74 —            if(lost_func(idx)<=0)
75 —                est_coe_w = est_coe_w + upd_mu*test_data_y(idx)*[test_data_x1(idx),test_data_x2(idx)];
76 —                est_coe_b = est_coe_b + upd_mu*test_data_y(idx);
77 —                break;
78 —            else
79 —            end
80 —        end
81 —        sum_lost_func = sum(double(lost_func<=0));
82 —    end
```

代码运行结果如图所示，可以看到，est_line 和 base_line 基本重合，达到了我们对数据进行分类的目的。



四、后续思考

1、对 est_w 和 est_b 设定不同的初始值，最后出来的 est_line 应该会略微有差异。毕竟目前感知机的代码是只要正确分为两类就可以收手，并不一定要求 lost_func 最小。

2、对于不能做二分类的数据（比如有一类数据就是有散点存在于另一类中，从理论上来说就没法做到完全可分）。我们应该允许有一定误差，这个时候就需要设定迭代次数，且需要编写不同的测试用例，设定不同初始，让每条测试用例各自进行仿真，最后比较各用力的 lost_func，选择当中的最小值。