## 问题求解—数学理论

观察数据，根据其特性，我们可以利用梯度下降进行参数更新，然后再进行预测

数学理论：

梯度下降算法在机器学习中是很普遍的算法，不仅可以用于线性回归问题，还可以应用到其他很多的机器学习的问题中。

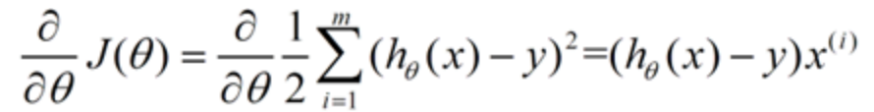
梯度下降算法是一种求局部最优解的方法，对于F(x)，在a点的梯度是F(x)增长最快的方向，那么它的相反方向则是该点下降最快的方向。

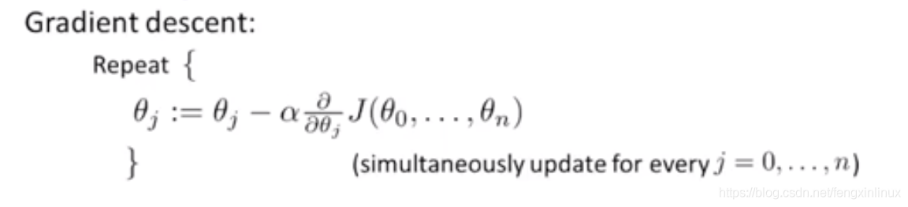
原理：将函数比作一座山，我们站在某个山坡上，往四周看，从哪个方向向下走一小步，能够下降的最快；

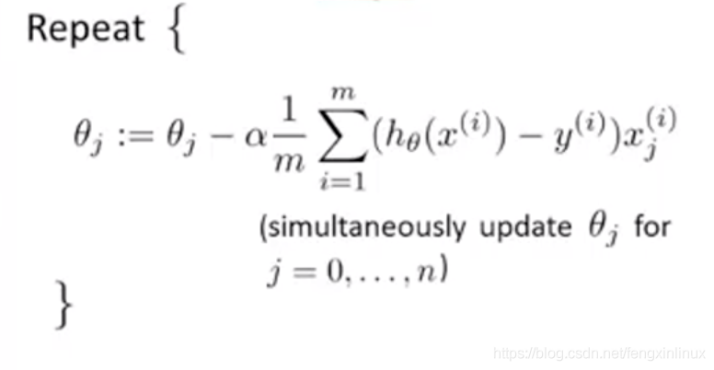
注意：当变量之间大小相差很大时，应该先将他们做处理，对特征值进行相应的缩放，使得他们的值尽量在同一个范围，这样会收敛的更快些，我们称之为特征值缩放。

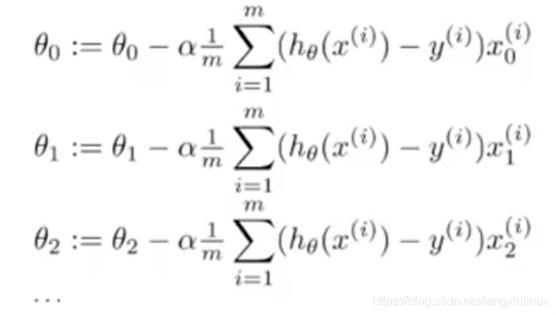
1）首先对θ赋值，这个值可以是随机的，也可以让θ是一个全零的向量。

2）改变θ的值，使得J(θ)按梯度下降的方向进行减少。









## 问题求解—程序分析与设计

1. 开发环境说明

python

1. 系统分析

创建了main函数，里面包括了data\_iter, linreg, squared\_loss, sgd函数

（1）main函数的功能是数据预处理以及调用函数，属于执行函数。

（2）data\_iter功能是批量生成训练数据；参数是batch\_size,features,labels，分别表示批量大小，特征与标签。

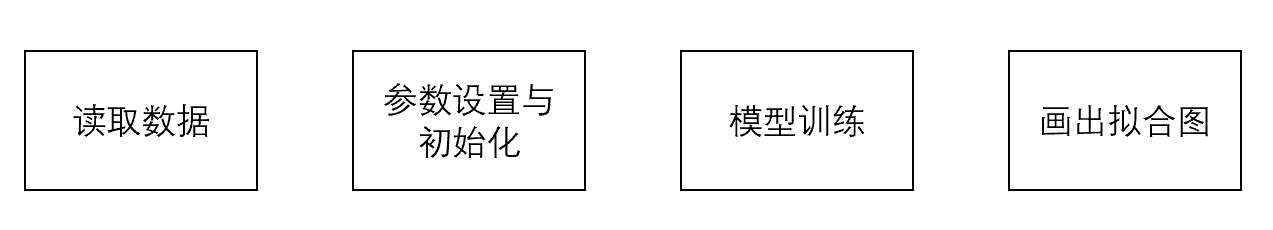
（3）linreg功能是计算并返回回归值；参数是x，w，b。

（4）squared\_loss功能是计算损失；参数是y\_hat与y，分别表示预测值和真实值。

（5）sgd功能是梯度下降更新参数；参数是batch\_size, lr, params，分别表示批量大小、学习率、以及需要更新的参数

1. 函数设计

main流程：



1. # 数据预处理
2. data = pd.read\_csv('实验数据.csv',encoding='utf-8',header=None)
3. features = torch.Tensor(data.iloc[:,0]).unsqueeze(1)/100
4. labels\_1 = torch.Tensor(data.iloc[:,1])/10000
5. labels\_2 = torch.Tensor(data.iloc[:,2])/10000
6. # 参数设置与初始化
7. lr = 0.001
8. num\_epochs = 100
9. batch\_size = 5
10. net = linreg
11. loss = squared\_loss
12. w = torch.ones(1, requires\_grad = True)
13. b = torch.zeros(1, requires\_grad = True)
14. # 第一个拟合模型训练
15. **for** epoch in range(num\_epochs):
16. **for** X, y in data\_iter(batch\_size, features, labels\_1):
17. l = loss(net(X, w, b), y)
18. l.sum().backward()
19. sgd([w, b], lr, batch\_size)
20. with torch.no\_grad():
21. **if** epoch == num\_epochs-1:
22. print('k值为：',w.detach().numpy())
23. print('b值为：',b.detach().numpy())
24. train\_l = loss(net(features, w, b), labels\_1)
25. #print(f'epoch {epoch+1}, loss {**float**(train\_l.mean()):f}')
27. print(f'2021年国内生产总值预测值为：{(w.detach().numpy()\*20.21+b.detach().numpy())\*10000}亿元')
28. # 画出拟合图
29. plt.figure(1)
30. plt.scatter(features[:, (0)].detach().numpy(), labels\_1.detach().numpy())
31. plt.grid()
32. x1=np.linspace(2010,2021,12)/100
33. y1=(w.detach().numpy()\*(x1)+b.detach().numpy())
34. plt.plot(x1,y1)

data\_iter流程：获取数据的index，打乱后再抽取数据



1. # 定义数据生产器
2. def data\_iter(batch\_size, features, labels):
3. num\_examples = len(features)
4. indices = list(range(num\_examples))
5. random.shuffle(indices)
7. **for** i in range(0, num\_examples, batch\_size):
8. batch\_indices = torch.tensor(indices[i:min(i+batch\_size, num\_examples)])
9. yield features[batch\_indices], labels[batch\_indices]

linreg计算回归值

1. # 定义模型
2. def linreg(X, w, b):
3. **return** torch.matmul(X, w)+b

squared\_loss计算损失函数

1. # 定义损失函数
2. def squared\_loss(y\_hat, y):
3. **return** (y\_hat - y.reshape(y\_hat.shape))\*\*2/2

sgd：定义优化方法，这里采用小批量梯度下降

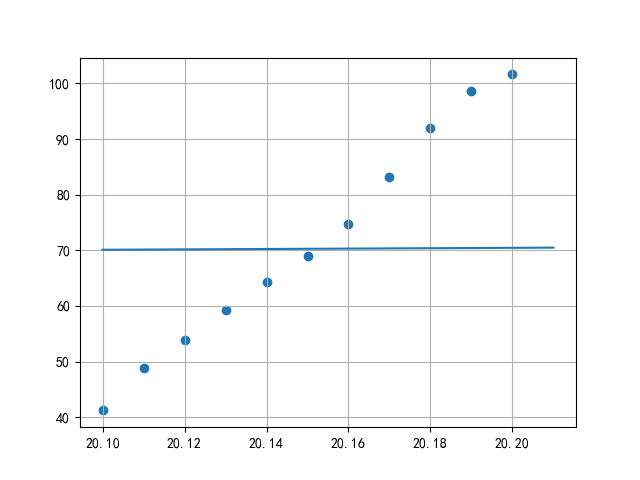
1. # 定义优化器，这里采用SGD
2. def sgd(params, lr, batch\_size):
3. with torch.no\_grad():
4. **for** param in params:
5. param -= lr\*param.grad/batch\_size
6. param.grad.zero\_()

## 问题求解结果

1. 程序测试（即程序操作流程）

运行main即可

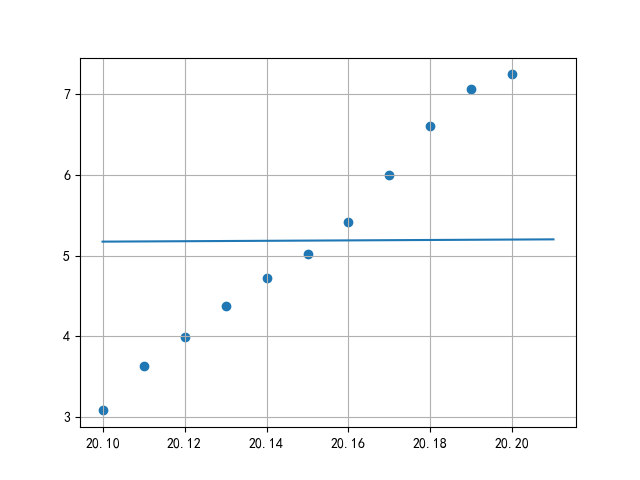
1. 结果分析（即问题求解结果分析）



k值为： [3.5809267]

b值为： [0.12126588]

2021年国内生产总值预测值为：[724917.94]亿元



k值为： [0.25665507]

b值为： [0.01227086]

2021年人均国内生产总值为：[51992.7]元

1. 结论

利用sgd对该数据进行优化拟合，效果不是特别好，从两幅图可知，样本点与曲线的贴合度很低。