作业 2：

一、对于给定的模型，比如二阶模型 $y=w_1*x^2+w_2*x+b$，尝试找出该 model 中的最佳 function。

分别验证 Data Scaling 技术、学习率 lr、BGD 及其两种变体 SGD 和 MBGD、不同的梯度优化方法、训练数据量，对损失函数 L(f)的收敛速度和准确度的影响。其中，

- 收敛速度：指，是否能很快找到最佳 function？即，是否能很快找到 Loss 的极小值？
- 收敛准确度：指，找到的最佳 function 是否正确？即是否能找到 Loss 函数的全局极小值？（若损失函数 L(f)为凸函数，极小值只有一个）

（对应的代码为：AI_StuAsssign_OrderMore.py）

1、验证数据归一化技术的效果。采用 BGD。超参数设置如下：

training_epochs	Learning Rate	N	TestingDataRatio	OrderNum	Seed	BatchSize
int(3000)	1*1e-3	50	0.2	2	18225209	N*(1-TestingDataRatio)

a) 设置 DataScaleFlag=0（表示未使用数据归一化技术）

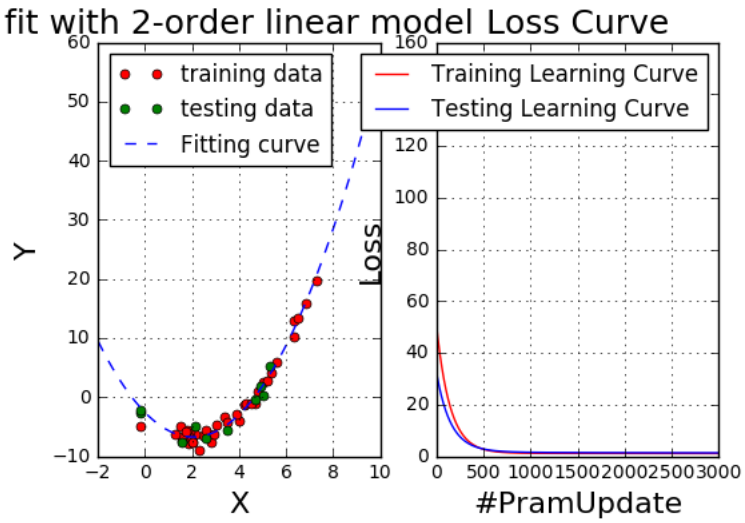
b) 设置 DataScaleFlag=1（表示使用数据归一化技术）

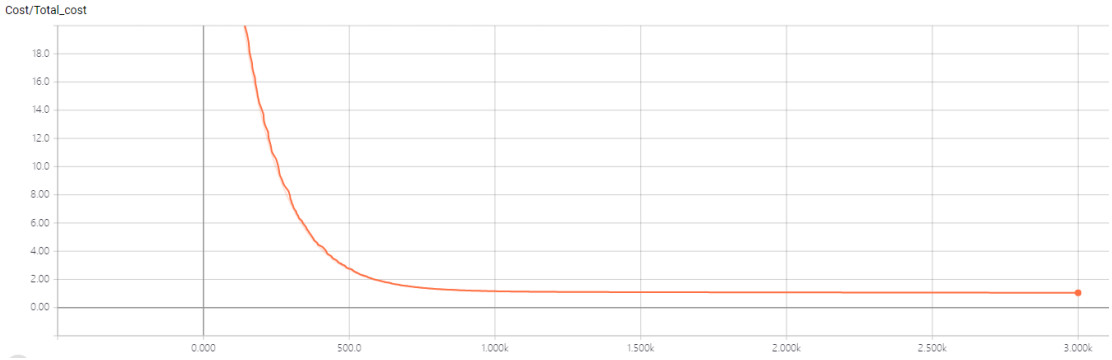
对比观察 tensorboard 中以上两种情况在训练数据集上的 Total_cost 的 learning Curve 图（即观察 Loss 在不同 epoch 时有何不同），并尝试归纳总结出数据归一化技术对 Total_cost 的收敛速度的影响。

答：（因为按照老师给出的原来的超参数进行训练时误差太大，所以我将学习率改为 0.001、training epochs 改为 3000 来进行训练）

a）不使用特征归一化技术时运行结果如下所示：

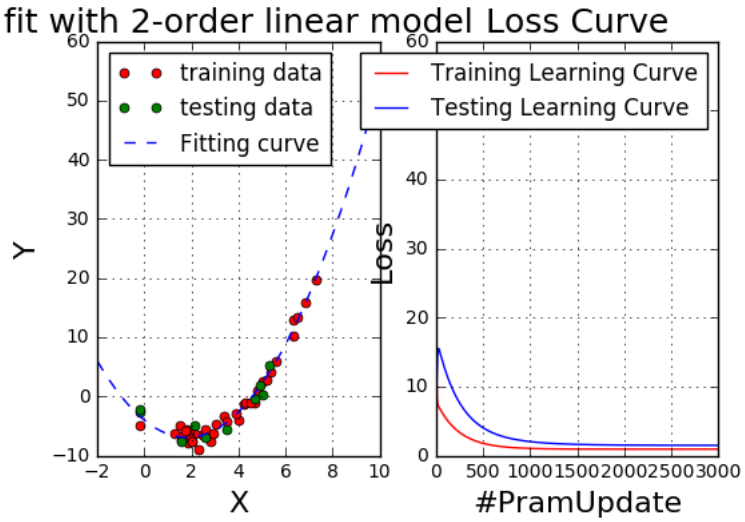
```
[[ 0.  0.]] [[ 1.  1.]]
Initial w,b: [array([[-0.36522242],
       [ 0.11645161]], dtype=float32), array([ 0.53994095], dtype=float32)]
TrainingDataNum|TestingDataNum= 40.0 | 10.0
fit with 2-order linear model...
最佳参数 W,b: [[ 0.99097008]
 [-4.05390358]] [-2.60380578]
23.540085792541504 秒
Training...Cost= 1.04822
Testing...Cost= 1.2925
```

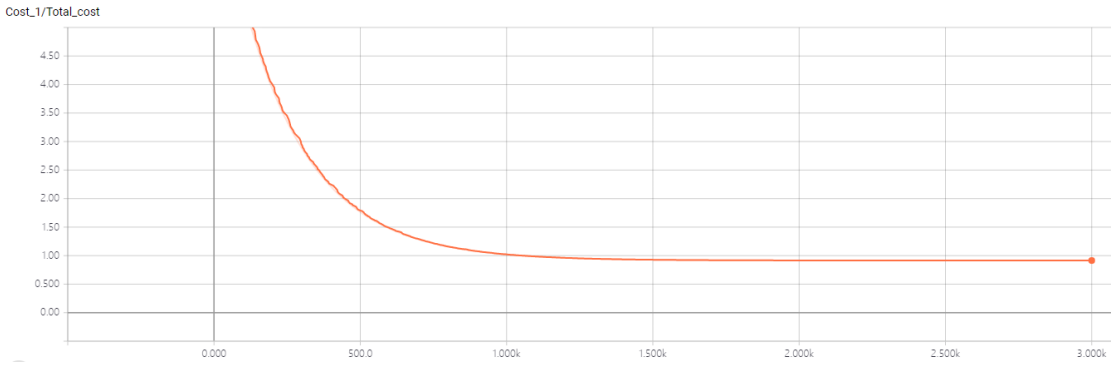




b）使用特征归一化技术时运行结果如下所示：

```
[[ 14.66708353   3.38105886]] [ 2.22623785  0.28440836]
Initial w,b: [array([[ 0.27158165],
       [-0.25995949]], dtype=float32), array([ 0.11009897], dtype=float32)]
TrainingDataNum|TestingDataNum= 40.0 | 10.0
fit with 2-order linear model...
最佳参数 W,b: [[ 0.89213812]
 [-3.21691632]] [-4.0488019]
23.267666339874268 秒
Training...Cost= 0.914476
Testing...Cost= 1.47065
```





从两种条件下可以观察到的结果是：在其他超参数保持一定时，使用特征归一化时训练的 Total_cost 的收敛速度会比不使用要稍快一点，所以在训练模型时，最好使用特征归一化技术，这样能减少不同特征间值域范围相差太大对收敛速度造成的影响。

2、学习率对收敛速度的影响。

2.1）采用 BGD。超参数设置如下：

training_epochs	BatchSize	N	TestingDataRatio	OrderNum	Seed	DataScaleFlag
int(2000)	N*(1-TestingDataRatio)	500	0.2	2	18225209	0

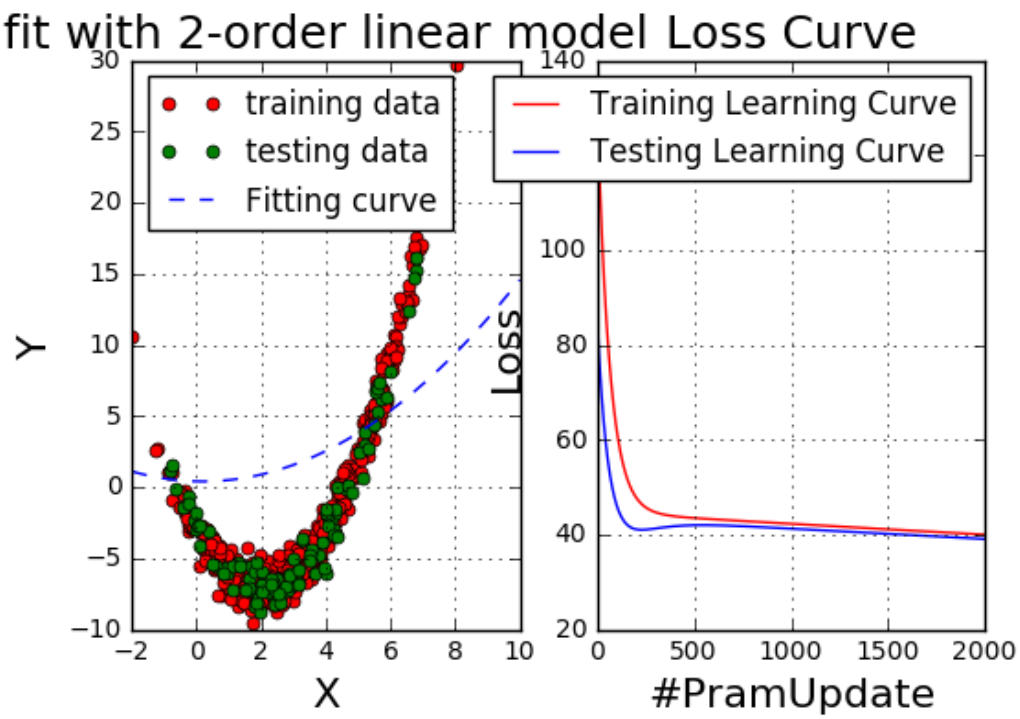
当学习率 lr 分别为以下 4 种取值时，观察 tensorboard 中训练数据集上的 Total_cost 的 learning curve 图。尝试总结：不同学习率 lr，对 Total_cost 的收敛速度的影响。

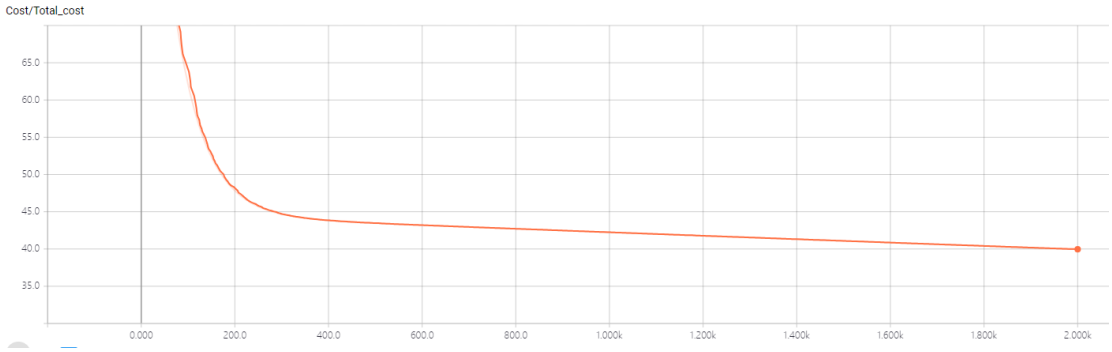
a) 设置：lr = 0.00001
b) 设置：lr = 0.0001
c) 设置：lr = 0.001
d) 设置：lr = 0.01

答：

a）

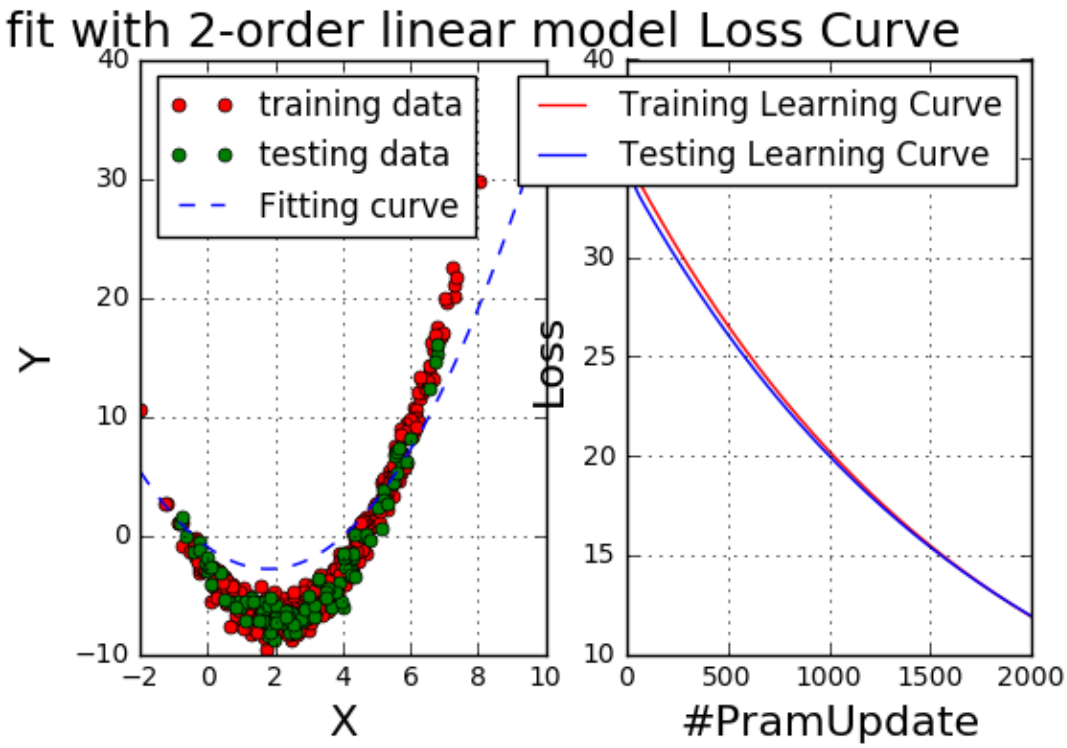
```
[[ 0.  0.]] [[ 1.  1.]]
Initial w,b: [array([[-0.36522242],
       [ 0.11645161]], dtype=float32), array([ 0.53994095], dtype=float32)]
TrainingDataNum|TestingDataNum= 400.0 | 100.0
fit with 2-order linear model...
最佳参数 W,b: [[ 0.14715022]
 [-0.0546229 ]] [ 0.39737201]
19.09967017173767 秒
Training...Cost= 39.957
Testing...Cost= 39.0372
```

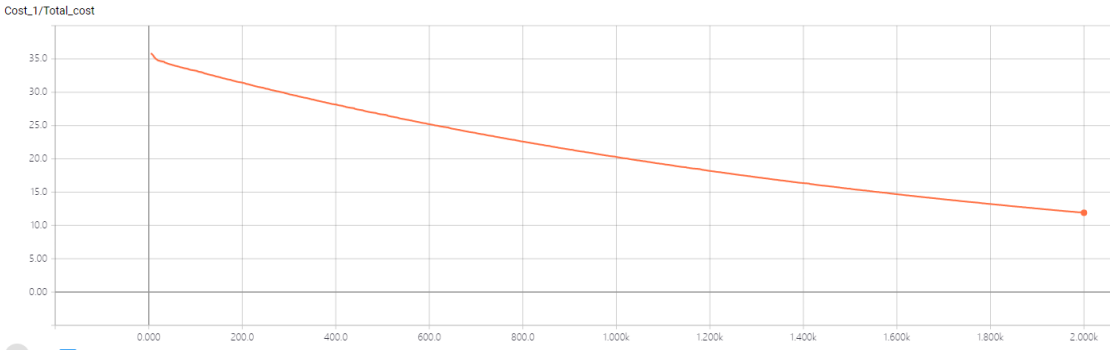




b)

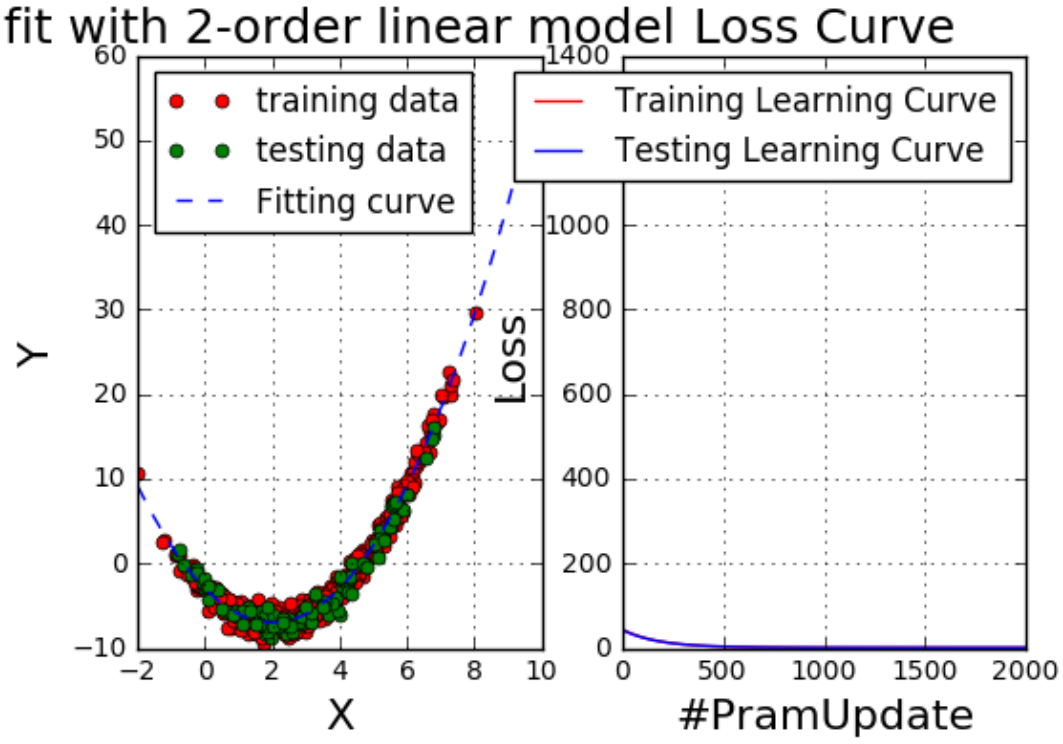
```
[[ 0.  0.]] [[ 1.  1.]]
Initial w,b: [array([[ 0.27158165],
       [-0.25995949]], dtype=float32), array([ 0.11009897], dtype=float32)]
TrainingDataNum|TestingDataNum= 400.0 | 100.0
fit with 2-order linear model...
最佳参数 W,b: [[ 0.56776732]
 [-2.04342675]] [-1.00498044]
16.884270191192627 秒
Training...Cost= 11.8827
Testing...Cost= 11.9057
```

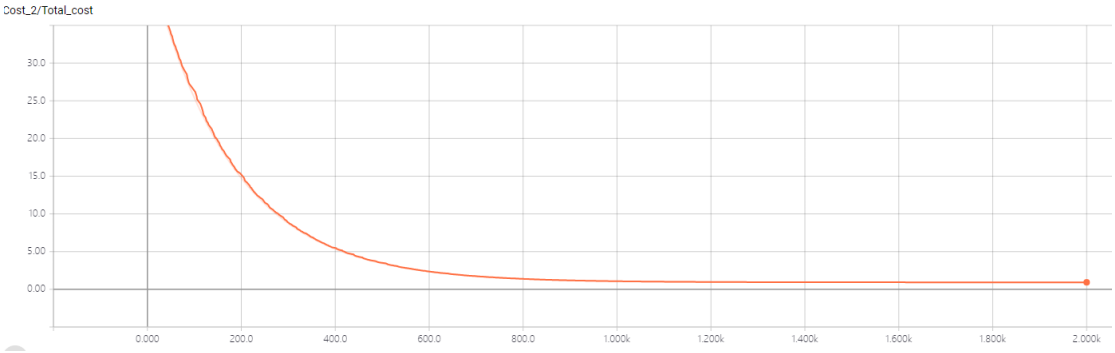




c)

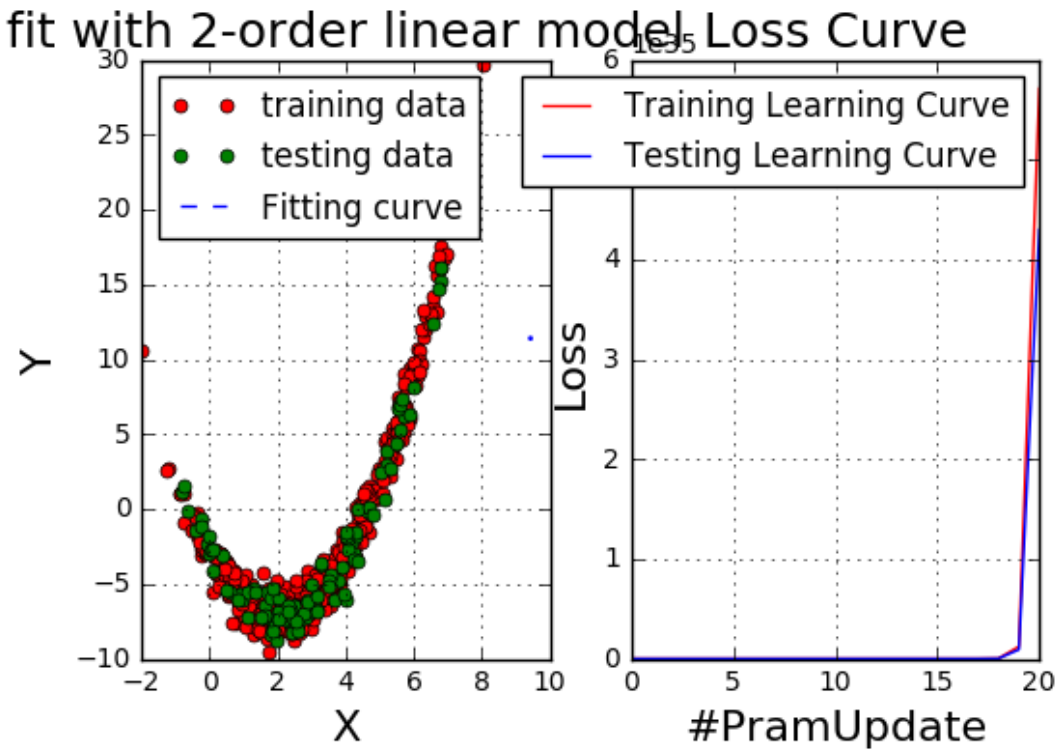
```
[[ 0.  0.]] [[ 1.  1.]]
Initial w,b: [array([[ 1.85635781],
       [ 0.91206825]], dtype=float32), array([-0.05565428], dtype=float32)]
TrainingDataNum|TestingDataNum= 400.0 | 100.0
fit with 2-order linear model...
最佳参数 W,b: [[ 1.00869727]
 [-4.0298748 ]] [-2.96259284]
17.005727767944336 秒
Training...Cost= 0.924187
Testing...Cost= 1.0212
```

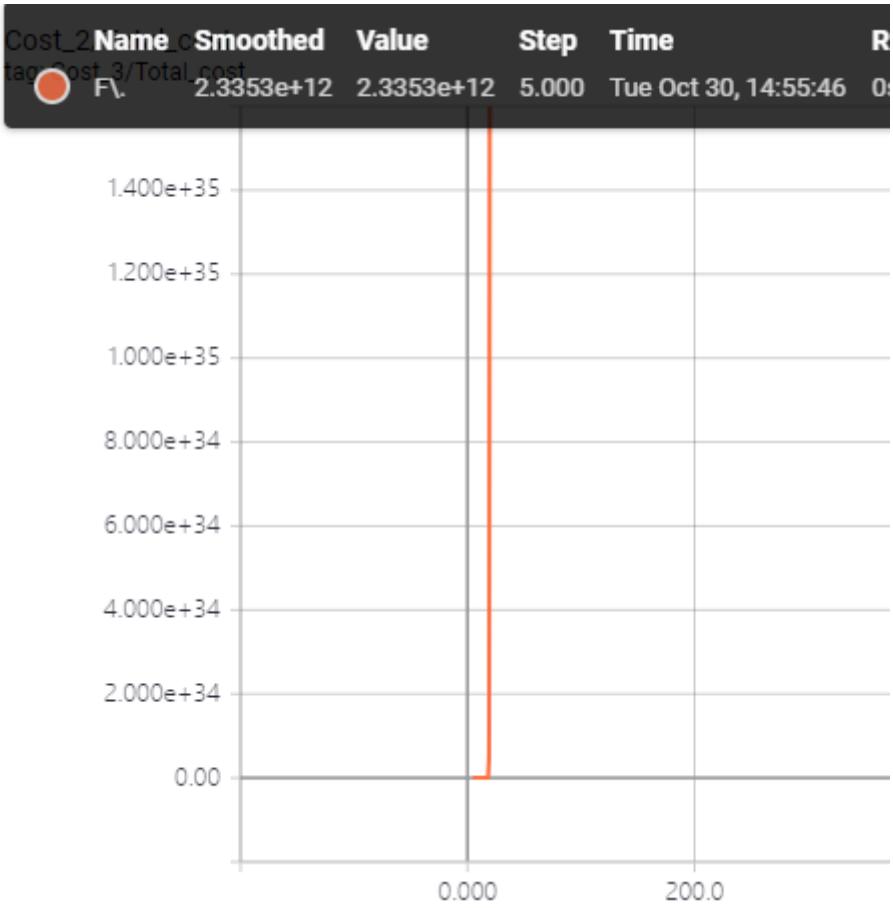




d)

```
[[ 0.  0.]] [[ 1.  1.]]
Initial w,b: [array([[-0.66750389],
       [-0.0548335 ]], dtype=float32), array([ 1.74707508], dtype=float32)]
TrainingDataNum|TestingDataNum= 400.0 | 100.0
fit with 2-order linear model...
最佳参数 W,b: [[ nan]
 [ nan]] [ nan]
18.233057022094727 秒
Training...Cost= nan
Testing...Cost= nan
```





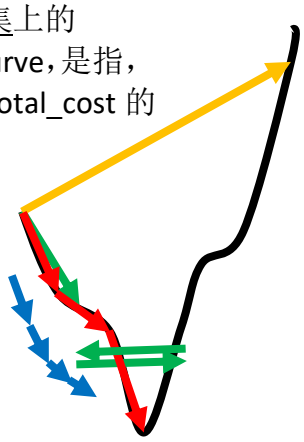
从上面的结果可以很显然的看出来了，当学习率较小（1*e-5，1*e-4）时，因为梯度下降的步伐太小，Total_cost 的收敛速度明显要慢得多了，2000 个 epoch 内曲线完全还没有呈现出快要收敛的意思，结束时拟合出来的曲线也是很差的。Lr=1*e-3 最终的训练误差和曲线的收敛情况都是比较满意的了，到了 1*e-2 时，学习率就有点偏大了，曲线已经不能正常梯度下降收敛了。所以在选择学习率时不可过大也不可以过小，1 从 1*e-5 开始，3 或 10 倍一选直到训练不出来的上一个学习率就很好。

2.2）采用 SGD。超参数设置如下：

training_epochs	BatchSize	N	TestingDataRatio	OrderNum	Seed	DataScaleFlag
int(1000)	1	50	0.2	2	18225209	0

当学习率 lr 分别是以下 4 种取值，观察 tensorboard 中训练数据集上的 Batch_Cost 和 Total_cost 的 learning curve（Batch_Cost 的 learning curve，是指，Loss 在不同 Batch 时的变化）。尝试总结：不同学习率 lr，对于 Total_cost 的收敛速度和准确度的影响。

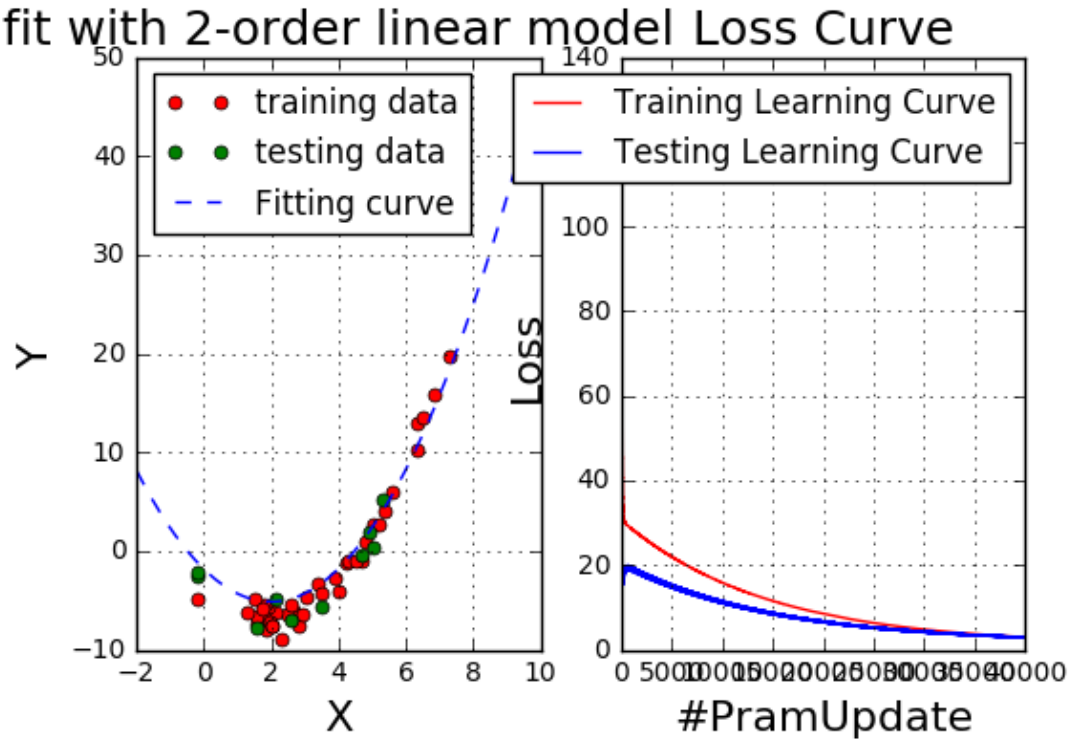
a) 设置：lr = 0.00001
b) 设置：lr = 0.0001
c) 设置：lr = 0.001
d) 设置：lr = 0.01

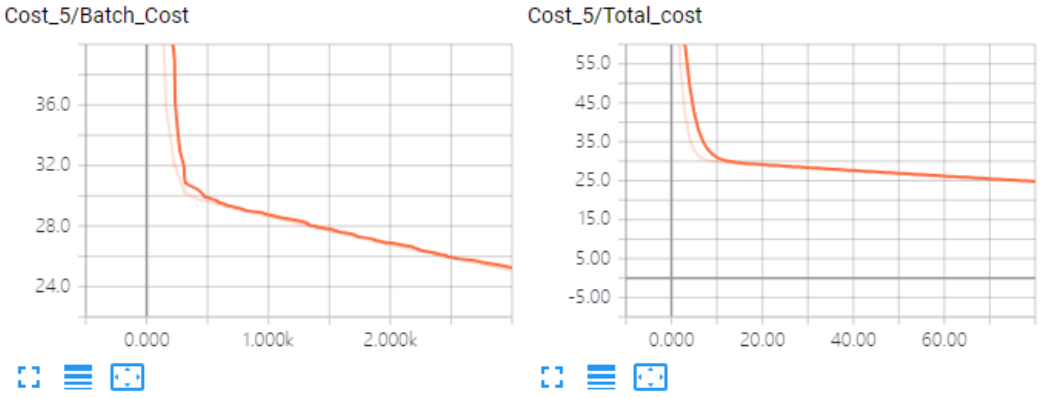


答：

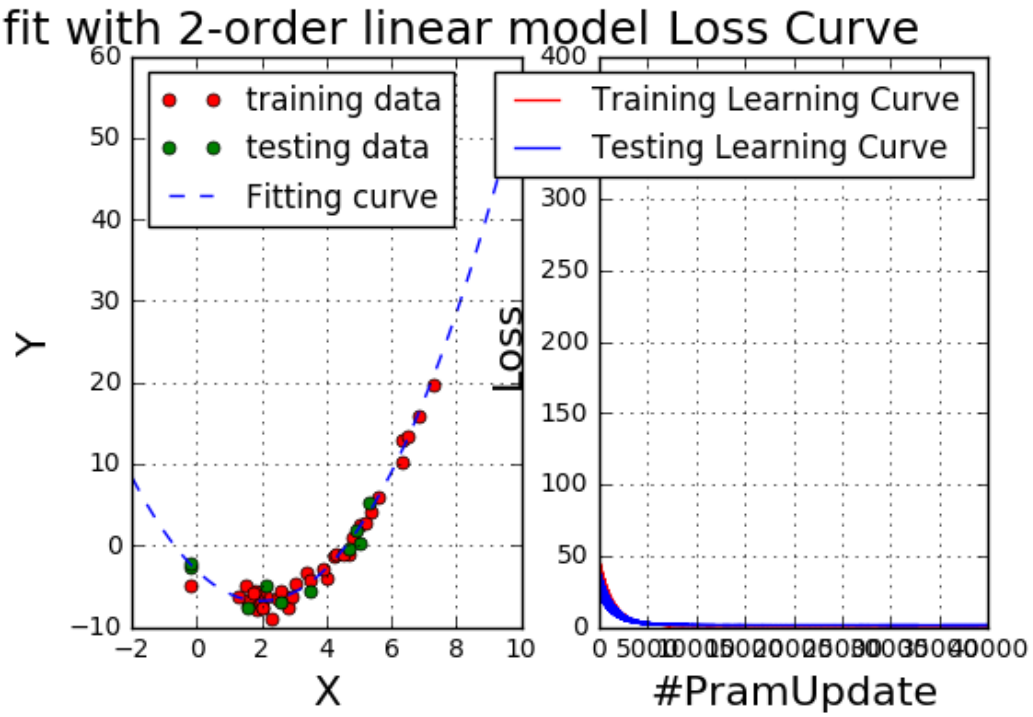
```
[[ 0.  0.]] [[ 1.  1.]]
Initial w,b: [array([[-0.16364539],
       [-0.80650234]], dtype=float32), array([-0.24529502], dtype=float32)]
TrainingDataNum|TestingDataNum= 40.0 | 10.0
fit with 2-order linear model...
最佳参数 W,b: [[ 0.83277822]
 [-3.31590652]] [-1.8471334]
238.63065195083618 秒
Training...Cost= 2.93664
Testing...Cost= 2.93396
```

a)

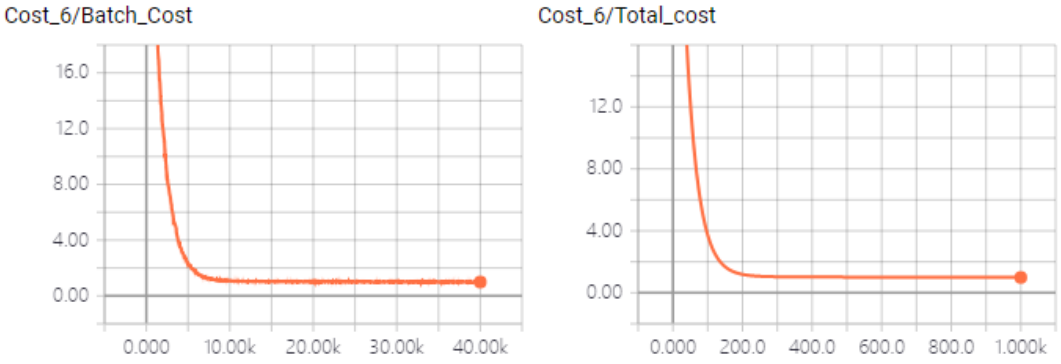




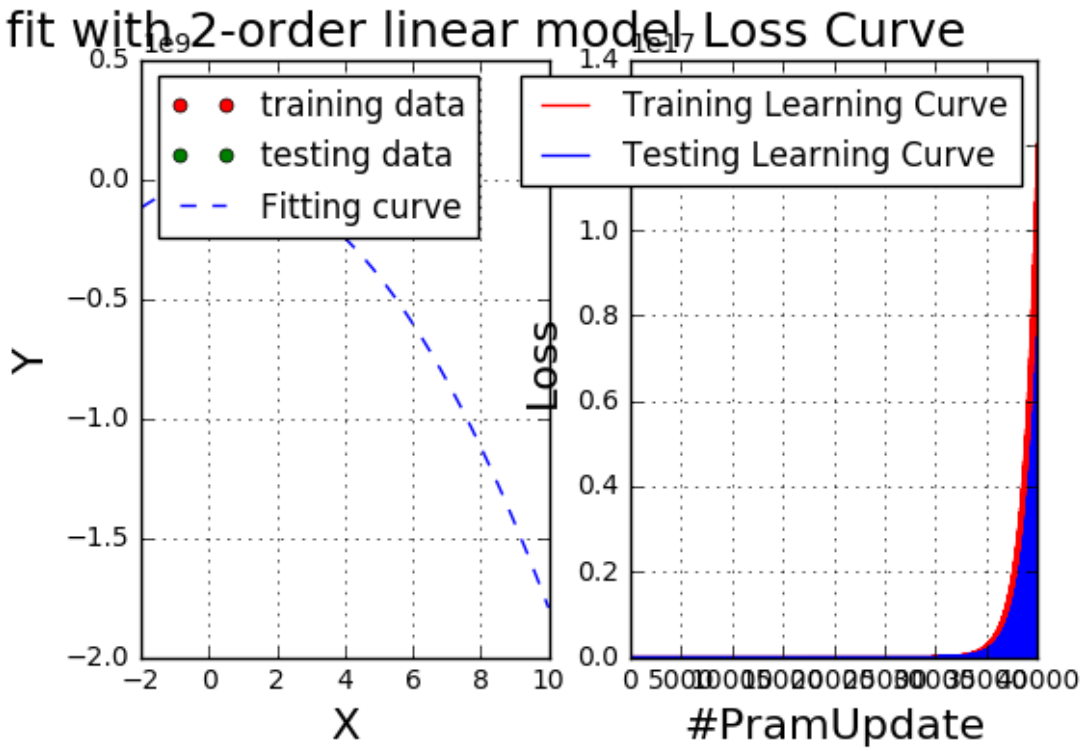
```
[[ 0.  0.]] [[ 1.  1.]]
Initial w,b: [array([[-0.75776762],
       [ 0.02883373]], dtype=float32), array([-0.21769717], dtype=float32)]
TrainingDataNum|TestingDataNum= 40.0 | 10.0
fit with 2-order linear model...
最佳参数 W,b: [[ 0.96649152]
 [-3.78835559]] [-3.09820914]
259.45737385749817 秒
Training...Cost= 0.987046
Testing...Cost= 1.25211
```



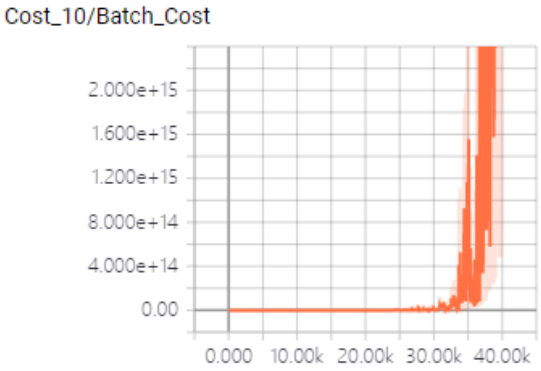
b)

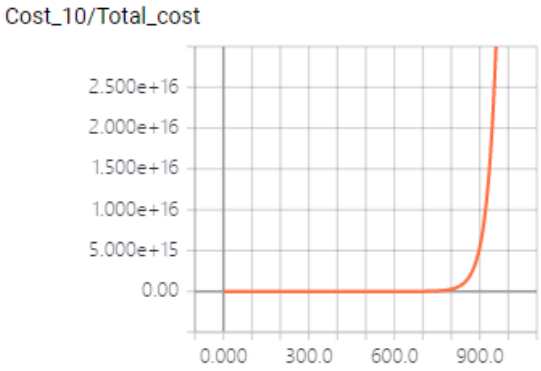


```
[[ 0.  0.]] [[ 1.  1.]]
Initial w,b: [array([[-0.53485155],
       [ 2.72216821]], dtype=float32), array([-0.86181062], dtype=float32)]
TrainingDataNum|TestingDataNum= 40.0 | 10.0
fit with 2-order linear model...
最佳参数 W,b: [[-19727386.]
 [ 18394880.]] [-5080825.5]
219.74335169792175 秒
Training...Cost= 1.14311e+17
Testing...Cost= 7.10711e+16
```

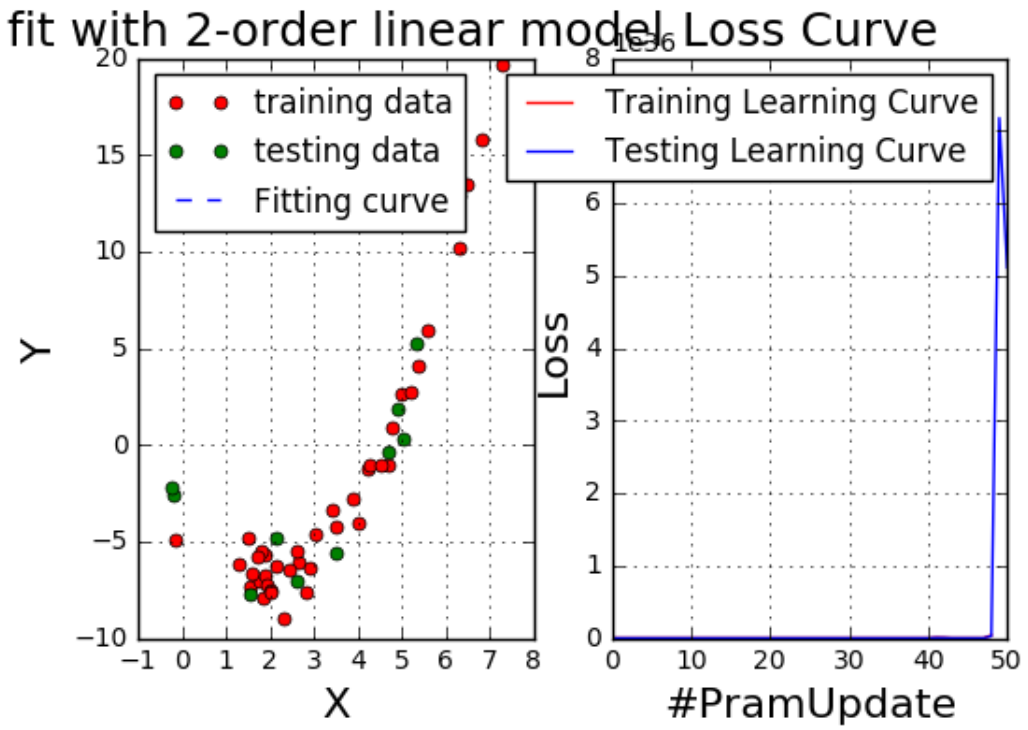


c)

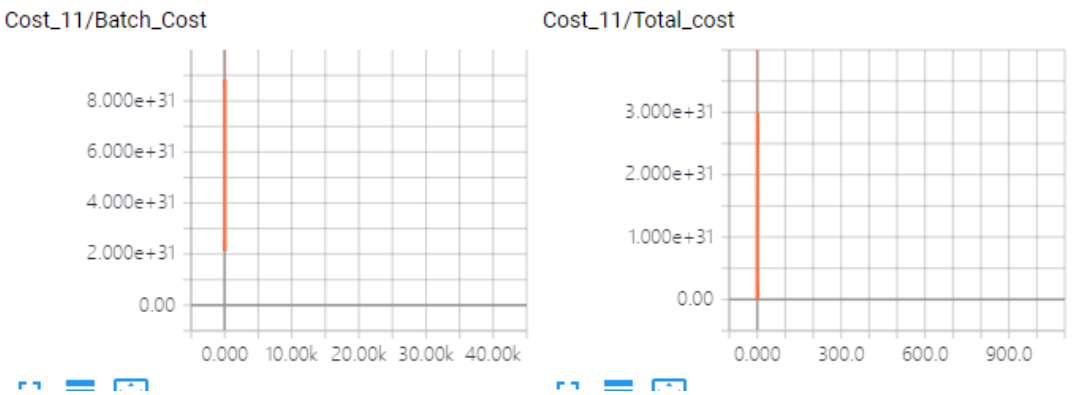




```
[[ 0.  0.]] [[ 1.  1.]]
Initial w,b: [array([[ 0.78543472],
       [ 1.5923593 ]], dtype=float32), array([ 1.42320824], dtype=float32)]
TrainingDataNum|TestingDataNum= 40.0 | 10.0
fit with 2-order linear model...
最佳参数 W,b: [[ nan]
 [ nan]] [ nan]
225.59048342704773 秒
Training...Cost= nan
Testing...Cost= nan
```



d)



当学习率较小时，因为梯度下降的步伐太小，损失的收敛速度要慢一些，1000 个 epoch 内 lr=0.00001 的曲线还没有收敛到最小值，训练误差也较大，lr=0.0001 的情况收敛速度和误差大小情况都要好一些。到了 Lr=1*e-3, 1*e-2 时，学习率就有点偏大了，从 batchcost 曲线可以看出来曲线振荡得厉害，步伐太大了，已经无法正常的进行梯度下降训练了。所以在选择学习率时不可过大也不可以过小，从 1*e-5 开始，3 或 10 倍一选直到曲线振荡训练不出来的上一个学习率就很好。

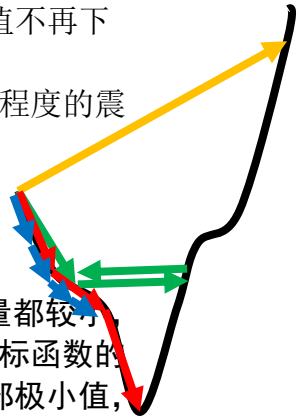
3、对比 BGD，SGD 和 MBGD，并进一步观察分析 MBGD 中 BatchSize 对收敛稳定性和收敛速度的影响。使用基本的 GD 方法，超参数设置如下：

training_epochs	Learning Rate	N	TestingDataRatio	OrderNum	Seed	DataScaleFlag
int(2000)	1*1e-3	500	0.2	2	18225209	0

a) 设置 BatchSize = N*(1-TestingDataRatio)（表示使用 BGD）
b) 设置 BatchSize =1（表示使用 SGD）
c) 设置 BatchSize = 10（表示使用 MBGD，且一个 echo 中包含 40 个 batch）
d) 设置 BatchSize = 40（表示使用 MBGD，且一个 echo 中包含 10 个 batch）
e) 设置 BatchSize = 80（表示使用 MBGD，且一个 echo 中包含 5 个 batch）

在 tensorboard 中对比以上 5 种情况在训练数据集上的 Total_cost 和 Batch_Cost 的 learning curve 图，观察不同 BatchSize 取值时对 Total_cost 收敛速度和 Batch_Cost 稳定性（出现震荡，即意味着不稳定）的影响。运行代码，将观察到如下现象：

1） Batchsize=80 时，Total Loss 收敛最快。
2） BGD 下，Total Loss 稳定的下降，不断朝极小值接近，但收敛速度慢。
3） SGD 和 Batchsize=10 这两种情况下，很早就出现了 Total Loss 值不再下降的现象，但实际上此时并没有找到 Total Loss 的极小值。
4） MBGD 下，随着 Batchsize 的取值不同，Batch_Cost 出现了不同程度的震荡。



Q1：现象 3）中，为何会出现 Total Loss 值不再更新的现象呢？
Q2：现象 4）中，为何会出现震荡？
Q3：你觉得以上 5 种不同的 BatchSize 中，哪种取值最好？为什么？
A1：因为在 SGD 和 BGD 的 Batchsize=10 这两种情况训练的样本数量都较多，SGD 频繁的执行更新所伴随的高方差（a high variance）会导致目标函数的剧烈波动，一方面波动能够使损失函数跳到一个全新并且更优的局部极小值，但另一方面每个训练样本中高方差的参数更新会导致损失函数大幅波动，因此我们可能无法获得给出损失函数的最小值。
A2：因为 MBGD 每次迭代的样本数量较少，虽然收敛速度较快，但训练出来的模型只是使得当前训练样本的损失最小，但对于总体样本来说，模型和目标函数还未达到最优解。所以在每次迭代过程中， batch_cost 会出现较大变化震荡，不同 batch_size 情况下，震荡的程度也不同。
A3：BatchSize = 80 的 MBGD 的取值是最好的，综上比较，此种取值是既收敛速度快，同时测试误差又最小的。

4、验证常用梯度优化算法的效果。
超参数设置如下：

training_epochs	Learning Rate	BatchSize	N	TestingDataRatio	OrderNum	Seed	DataScaleFlag
int(2000)	1e-3	N*(1-TestingDataRatio)	500	0.2	2	18225209	0

采用不同的梯度优化方法，比如 AdaGrad, Adam, Moment 算法，在 tensorboard 中对比这 3 类优化方法在训练数据集上的 Total_cost 的 learning curve 图，并对观察到的结果进行解释。**注意，**对于 AdaGrad 和 Adam 算法，可以增加 Learning Rate，将 Learning Rate 设置为 0.1，将可加快收敛速度。

Q：通过观察分析，你认为，对于当前代码，哪种梯度优化算法最好？为什么？

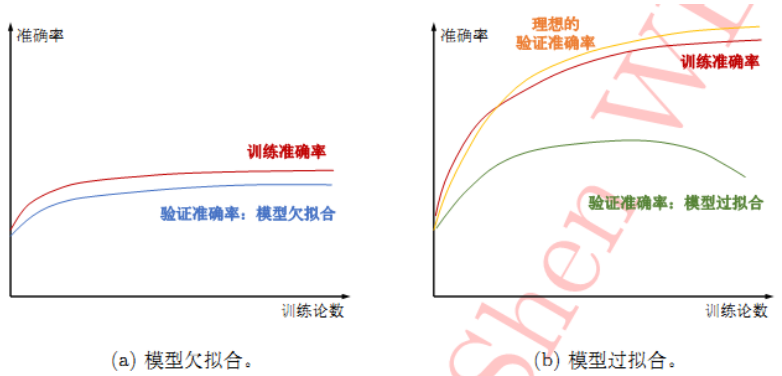
A：对比几个实验结果，Adam 算法的收敛速度是最快的，Cost 也是最小的。

5、数据集和验证集 CV 划分对“过拟合”的影响。采用 BGD 作为梯度下降方法。超参数设置如下：

training_epochs	Learning Rate	BatchSize	N	OrderNum	Seed	DataScaleFlag
int(2000)	0.001	N*(1-TestingDataRatio)	50	3	18225209	1

对于以下三种 CV 情况，在 tensorboard 中分别对比观察训练数据和测试数据上 Total_cost 的 Learning Curve 图。请指出哪种情况下出现了过拟合（模型过拟合的判断依据），并尝试总结训练数据集的数据量对过拟合的影响。

a) 设置：TestingDataRatio = 0.2
b) 设置：TestingDataRatio = 0.5
c) 设置：TestingDataRatio = 0.8



(a) 模型欠拟合。　(b) 模型过拟合。

图 10.5: 模型欠拟合和过拟合。

答：在 mbgd 情况下，TestingDataRatio=0.8 的时候出现过拟合了，这个时候每个 batch 内的数据量是比较少的，验证了数据集相对参数数量太少的情况会出现过拟合的情况。增加数据量是解决过拟合最有效的方法，只要给足够多的数据，让模型「看见」尽可能多的「例外情况」，它就会不断修正自己，从而得到更好的结果。

如何获取更多数据，可以有以下几个方法：

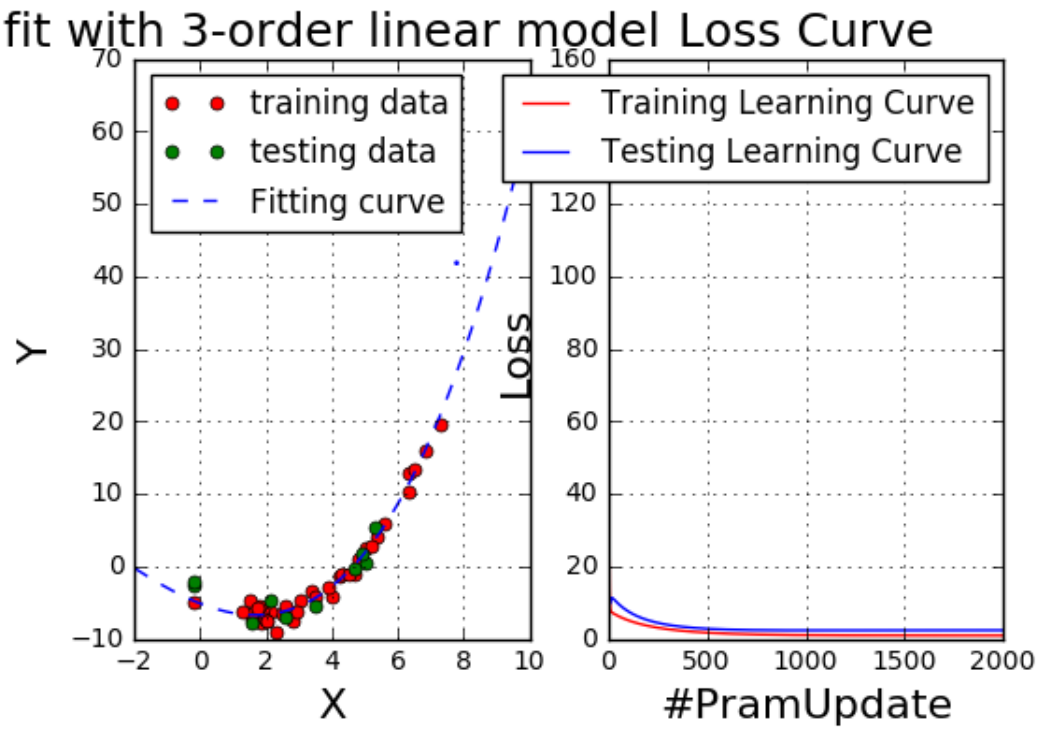
从数据源头获取更多数据：这个是容易想到的，例如物体分类，我就再多拍几张照片好了；但是，在很多情况下，大幅增加数据本身就不容易；另外，我们不清楚获取多少数据才算够；

根据当前数据集估计数据分布参数，使用该分布产生更多数据：这个一般不用，因为估计分布参数的过程也会代入抽样误差。

数据增强（Data Augmentation）：通过一定规则扩充数据。如在物体分类问题里，物体在图像中的位置、姿态、尺度，整体图片明暗度等都不会影响分类结果。我们就可以通过图像平移、翻转、缩放、切割等手段将数据库成倍扩充

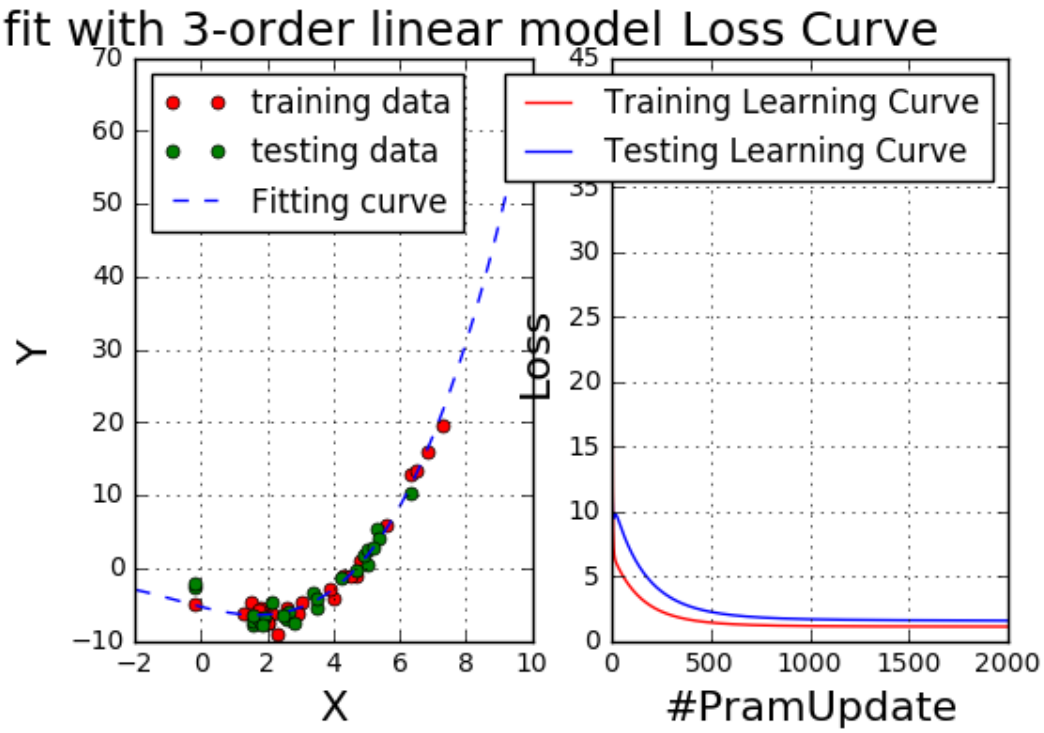
a)

```
[[ 74.20705902  14.66708353   3.38105886]] [ 15.43379651   2.22623785   0.28440836]
Initial w,b: [array([[-0.36522242],
       [ 0.11645161],
       [-0.62009418]], dtype=float32), array([ 0.53994095], dtype=float32)]
TrainingDataNum|TestingDataNum= 40.0 | 10.0
fit with 3-order linear model...
最佳参数 W,b: [[ 0.04256451]
 [ 0.42131323]
 [-1.77146296]] [-5.15023232]
16.5740647315979 秒
Training...Cost= 0.876082
Testing...Cost= 2.32759
```



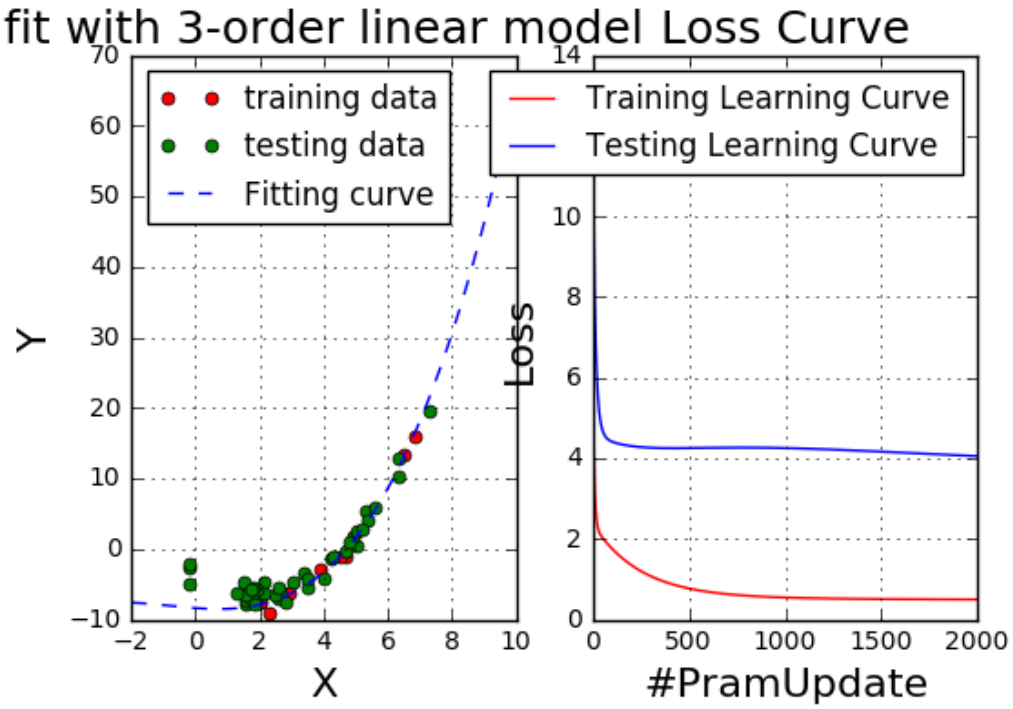
b)

```
[[ 82.02629373  15.50636931   3.42575479]] [ 21.84027233   3.07381189   0.38835929]
Initial w,b: [array([[ 0.27158165],
       [-0.25995949],
       [ 0.35058597]], dtype=float32), array([ 0.11009897], dtype=float32)]
TrainingDataNum|TestingDataNum= 25.0 | 25.0
fit with 3-order linear model...
最佳参数 W,b: [[ 0.06773576]
 [ 0.16386892]
 [-1.12678278]] [-5.35638714]
15.706478357315063 秒
Training...Cost= 1.09009
Testing...Cost= 1.55275
```



c)

```
[[ 90.51358041  17.18045343   3.75837468]] [ 34.1582629    4.75341704   0.55272717]
Initial w,b: [array([[ 1.85635781],
       [ 0.91206825],
       [ 0.47756279]], dtype=float32), array([-0.05565428], dtype=float32)]
TrainingDataNum|TestingDataNum= 9.999999999999998 | 40.0
fit with 3-order linear model...
最佳参数 W,b: [[ 0.0612242 ]
 [ 0.15811248]
 [-0.32204666]] [-8.39381123]
16.06514835357666 秒
Training...Cost= 0.495501
Testing...Cost= 4.04957
```



二、跨越不同模型进行比较，从而找出终极的最佳 function。（对应的代码为：AI_StuAsssign_CrossModel.py）

使用 Adam 作为梯度下降优化方法。

超参数设置如下：

training_epochs	BatchSize	N	TestingDataRatio	Learning Rate	Seed	DataScaleFlag
int(1e20)	N*(1-TestingDataRatio)	500	0.2	0.1	用自己的学号的数字部分	1

1、分别定义 1 阶、2 阶、3 阶、4 阶和 5 阶模型，找出各个模型下的最佳 function，并按照以下示例的表格模板来记录最佳参数组合、最佳 function 在训练数据和测试数据上的损失值。

注意，请使用自己学号的数字部分作为随机数生成器的 seed，从而保证每位同

学所使用的参数 W 和 b 的初值不同，以保证每个人最终记录的表结果不一样。
采用 seed=18225209，则可得到如下结果：

Model & Optimized function	Training Loss	Testing Loss
1 阶: y=2.39731148x--9.18679619	21.1233	20.0794
2 阶： y=1.00681202*x^2-4.003224*x-3.04389644	0.922675	1.0286
3 阶： y=7.17437352e-04*x^3+ 0.999737128x^2 -3.98628310*x-3.04875731	0.922699	1.02898
4 阶： y= 9.79180118e-04*x^4 -1.17568628e-02*x^3+ 1.04489496x^2- 4.01995696*x-3.07472181	0.92165	1.0314
5 阶： y= -5.52100529e-04*x^5 +9.60560863e-03*x^4 -5.33197147e-02*x^3 + 1.09674622x^2 -3.96587831*x-3.1385417	0.921736	1.04088

2、 观察上表中你的记录结果，试找出跨模型的最佳 function，并说出你的选择理由。

答：从测试误差来看 2 阶 function 拟合得最好，1 阶误差是最大的，3、4、5 阶模型的>2 阶的系数其实拟合出来都是极小的，

```
tf.set_random_seed(Seed)
#Construct N data samples(including training data set
x = np.linspace(0, 6, N) + np.random.randn(N)
x = np.sort(x)
y = x ** 2 - 4 * x - 3 + np.random.randn(N)
x.shape = -1, 1
```

与初始定义的就是一个 2 阶的函数也是蛮相符。