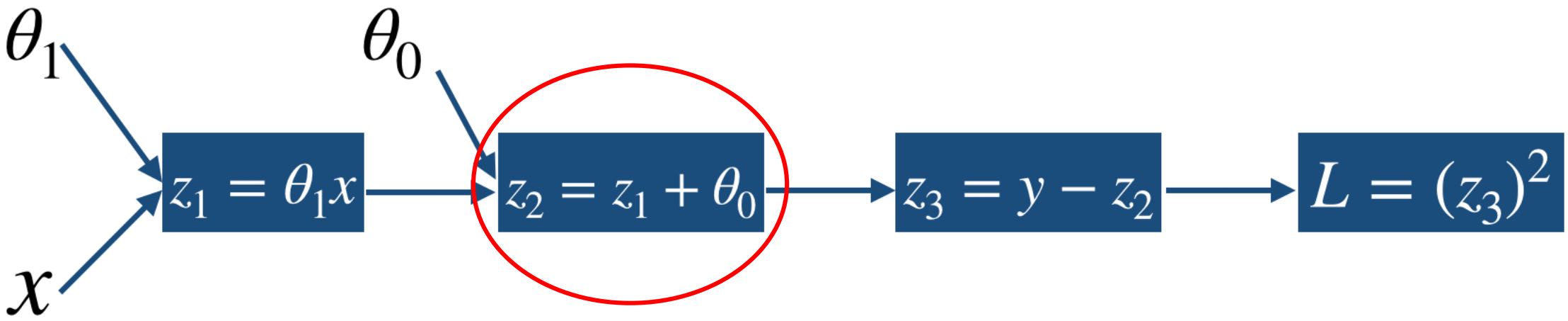
$$y = \theta_1 x + \theta_0$$

Forward and Backward

Propagation

Forward propagation



Node

```python
class mul_node():
    def __init__(self):
        self.x, self.y, self.z = None, None, None

    def forward(self, x, y):
        self.x, self.y, self.z = x, y, x*y
        return self.z
    def backward(self, dL):
        return self.y*dL, self.x*dL
```

```python
class plus_node():
    def __init__(self):
        self.x,self.y,self.z = None,None,None
    def forward(self,x,y):
        self.x,self.y,self.z = x,y,x+y
        return self.z
    def backward(self,dL):
        return dL, dL
```

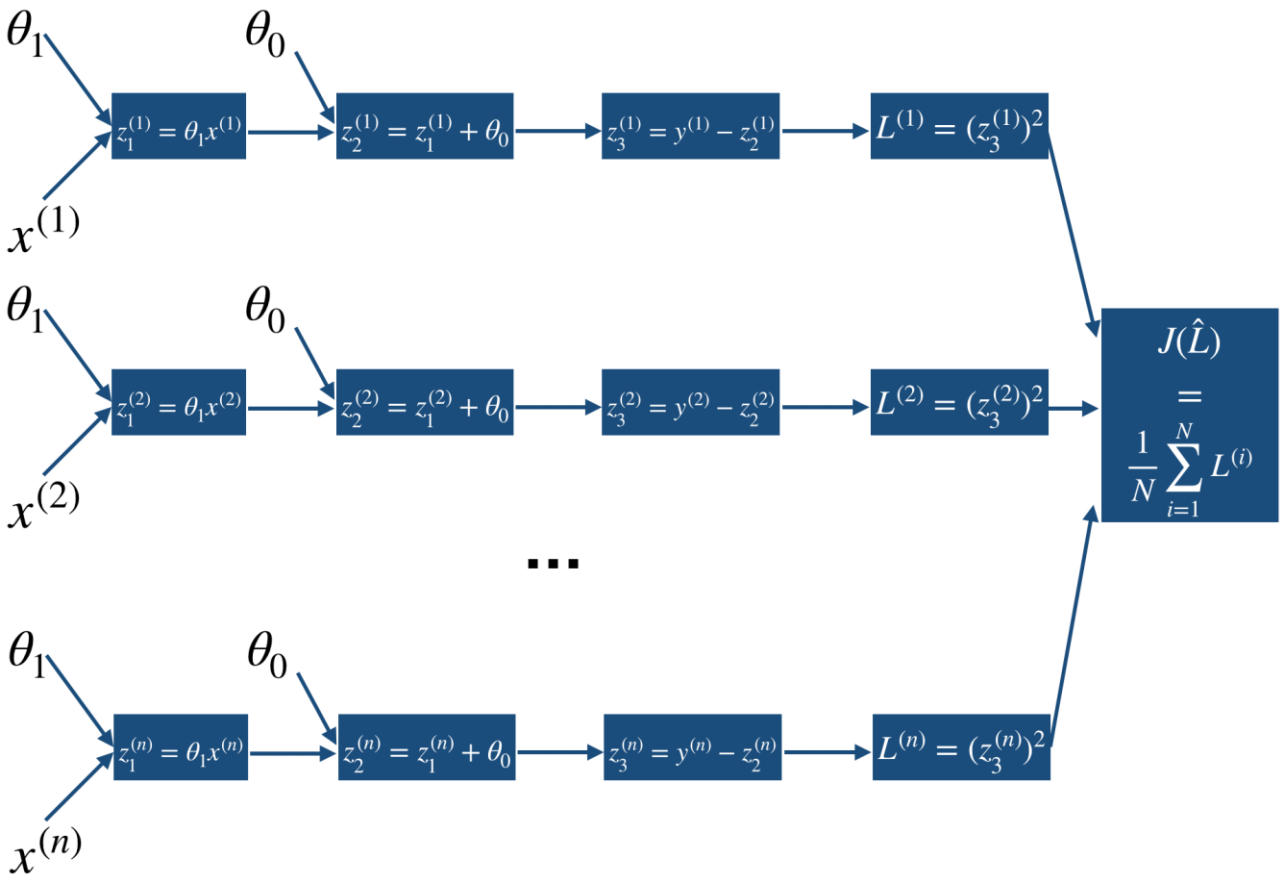
```python
class minus_node():
    def __init__(self):
        self.x,self.y,self.z = None,None,None
    def forward(self,x,y):
        self.x, self.y, self.z = x, y, x - y
        return self.z
    def backward(self,dL):
        return dL, -1 * dL
```

```python
class square_node():
    def __init__(self):
        self.x, self.z = None, None

    def forward(self, x):
        self.x, self.z = x, x*x
        return self.z

    def backward(self, dL):
        return 2*self.x*dL
```

Batch gradient descent



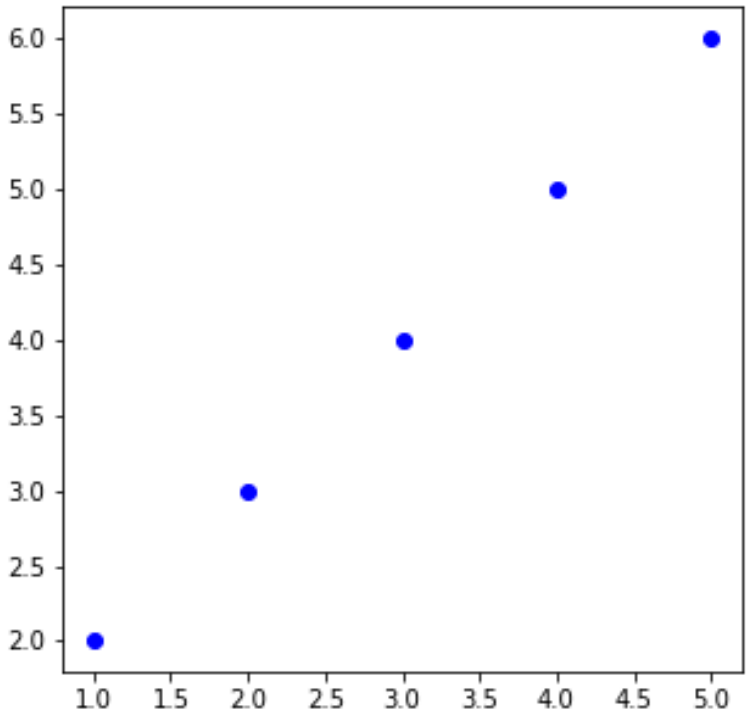

```
class cost_node():
    def __init__(self):
        self.x, self.z = None, None

    def forward(self, x):
        self.x = x
        self.z = np.mean(self.x)
        return self.z
    def backward(self):
        return 1/len(self.x)*np.ones(shape = (len(self.x)))
```

Data

Data Index	Study Hour	Math Score
0	1	2
1	2	3
2	3	4
3	4	5
4	5	6

```
x_data = np.arange(1,6)
y_data = x_data +1
```



Forward and Backward

```
z1 = z1_node.forward(theta1,x_data)
z2 = z2_node.forward(z1,theta0)
z3 = z3_node.forward(y_data,z2)
z4 = z4_node.forward(z3)
C = c_node.forward(z4)
```

Forward Propagation

```
dz4 = c_node.backward()
dz3 = z4_node.backward(dz4)
dy,dz2 = z3_node.backward(dz3)
dz1,dTheta0 = z2_node.backward(dz2)
dTheta1,dx = z1_node.backward(dz1)
```

Backward Propagation

Theta update

```
theta1 = theta1 - lr*np.sum(dTheta1)
theta0 = theta0 - lr*np.sum(dTheta0)

loss_list.append(C)
theta1_list.append(theta1)
theta0_list.append(theta0)
```

	0
0	1
1	2
2	3
3	4
4	5

X_data

	0
0	2
1	3
2	4
3	5
4	6

Y_data

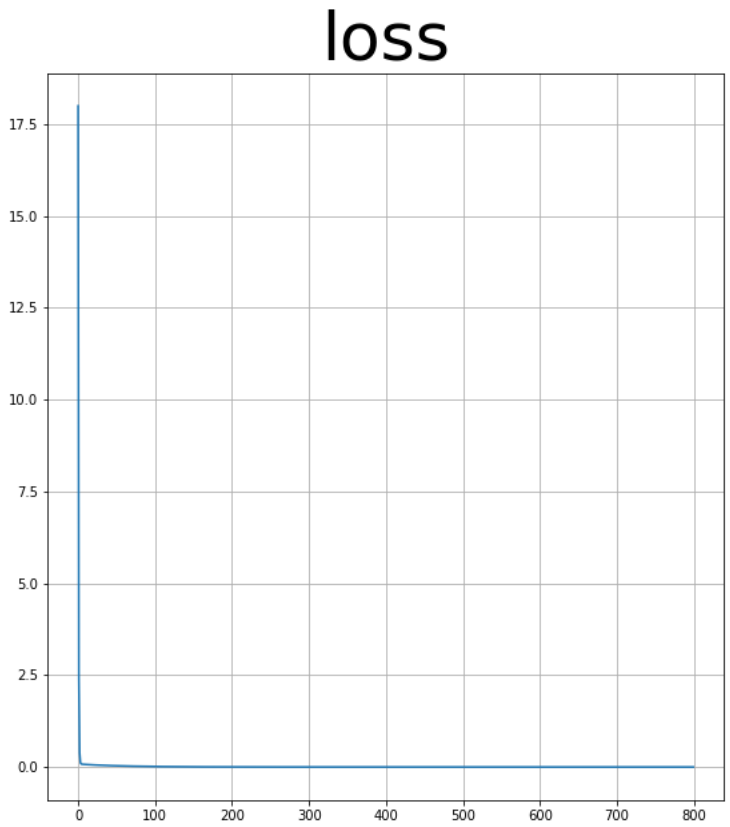
	0
0	-0.000127665
1	-7.8757e-05
2	-2.98493e-05
3	1.90585e-05
4	6.79663e-05

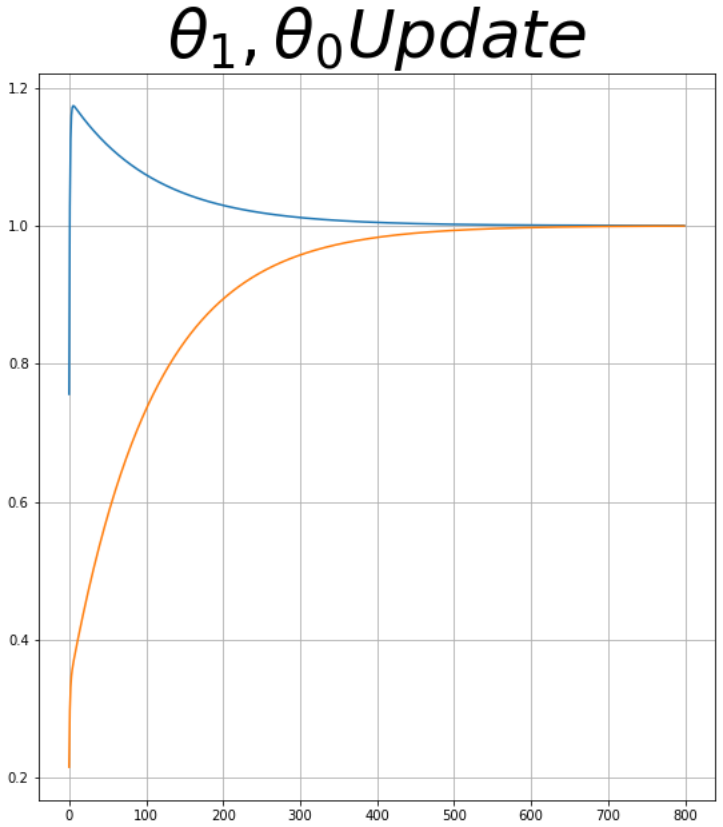
dTheta0

	0
0	-0.000127665
1	-0.000157514
2	-8.95478e-05
3	7.62341e-05
4	0.000339831

dTheta1

Result


loss
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
100
200
300
400
500
600
700
800


$\theta_1, \theta_0 Update$
1.2
1.0
0.8
0.6
0.4
0.2
0
100
200
300
400
500
600
700
800