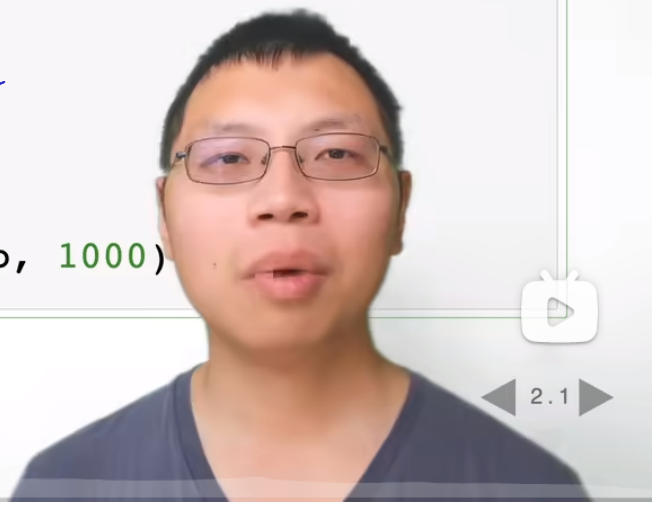
DIVE INTO DEEP LEARNING

根据带有噪声的线性模型构造一个人造数据集。 我们使用线性模型参数 $\mathbf{w} = [2, -3.4]^\top$、$b = 4.2$和噪声项$\epsilon$生成数据集及其标签：

$$\mathbf{y} = \mathbf{X}\mathbf{w} + b + \epsilon$$

```
In [33]: def synthetic_data(w, b, num_examples):
             """生成 y = Xw + b + 噪声。"""
             X = torch.normal(0, 1, (num_examples, len(w)))
             y = torch.matmul(X, w) + b
             y += torch.normal(0, 0.01, y.shape)
             return X, y.reshape((-1, 1))

         true_w = torch.tensor([2, -3.4])
         true_b = 4.2
         features, labels = synthetic_data(true_w, true_b, 1000)
```



实验一下超参数不同的选择会有什么样的效果

```
lr = 0.03
num_epochs = 3
net = linreg
loss = squared_loss
for epoch in range(num_epochs):
    for X, y in data_iter(batch_size, features, labels):
        l = loss(net(X, w, b), y)
        l.sum().backward()
        sgd([w, b], lr, batch_size)
    with torch.no_grad():
        train_l = loss(net(features, w, b), labels)
        print(f'epoch{epoch + 1}, loss{float(train_l.mean()):f}')
```

```
epoch1, loss0.030892
epoch2, loss0.000117
epoch3, loss0.000049
```

```
lr = 0.001
num_epochs = 3
net = linreg
loss = squared_loss
for epoch in range(num_epochs):
    for X, y in data_iter(batch_size, features, labels):
        l = loss(net(X, w, b), y)
        l.sum().backward()
        sgd([w, b], lr, batch_size)
    with torch.no_grad():
        train_l = loss(net(features, w, b), labels)
        print(f'epoch{epoch + 1}, loss{float(train_l.mean()):f}')
```

```
epoch1, loss13.550470
epoch2, loss11.083100
epoch3, loss9.065393
```

tips:当改变学习率的时候，如果你不从新随机初始化w,b.那么新的训练就会接着上一次训练得到的w,b继续更新，这样就没办法观察“单纯改变学习率”对收敛速度和结果的影响

当学习率比较小的时候他的误差会变大，当然你也可以增加他的epochs

```
lr = 10
num_epochs = 3
net = linreg
loss = squared_loss
for epoch in range(num_epochs):
    for X, y in data_iter(batch_size, features, labels):
        l = loss(net(X, w, b), y)
        l.sum().backward()
        sgd([w, b], lr, batch_size)
    with torch.no_grad():
        train_l = loss(net(features, w, b), labels)
        print(f'epoch:{epoch + 1}, loss:{float(train_l.mean()):f}')
```

```
epoch:1, loss:nan
epoch:2, loss:nan
epoch:3, loss:nan
```

太大了，超出了浮点运算的计算范围了