一、正则化（Regularization）

1. 总体损失形式
   对训练集上的数据损失（data loss）加上正则化项（regularizer）：

$$L(W) = \frac{1}{N} \sum_{i=1}^{N} L_{\text{data}}(f(x_i; W), y_i) + \lambda R(W),$$

   其中 λ 是正则化强度（超参数）。
2. 常见正则化方法
   - L2（权重衰减 / Ridge）：

$$R(W) = \sum_k \sum_l W_{k,l}^2$$

     缩小权重，倾向于将所有特征都保留，但降低影响力
   - L1（Lasso）：

$$R(W) = \sum_k \sum_l |W_{k,l}|$$

     产生稀疏解，即让部分权重精确为零
   - Elastic Net（L1 + L2）：

$$R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$$

     平衡L1和L2的特点，既能产生稀疏解，又能处理特征相关性
3. 为什么需要正则化

- 减少过拟合（减少模型对训练噪声的拟合）。
- 让模型更简单，以便于在测试集上更好的工作
- 通过增加“额外曲率”的方式促进正则化。

二、优化（Optimization）

1. 最优化策略：
   i. Random search，随机尝试很多不同的权重，然后看其中哪个最好。
   ii. Follow the slope，在多维空间中，梯度是沿每个维度的（偏导数）向量。任意方向的坡度是该方向与梯度的点积。最陡下降的方向是负梯度。跟随梯度找到损失最小的方法
2. 梯度下降(Gradient Descent)
   i. 程序重复地计算损失函数的梯度，然后对参数进行更新，直到结果不再变化

```
while True:
  weights_grad = evaluate_gradient(loss_fun, data, weights)
  weights += - step_size * weights_grad # perform parameter update
```

3. 随机梯度下降(Stochastic Gradient Descent) (SGD)
   i. 挑选数据集中的一批数据来进行训练

```
while True:
  data_batch = sample_training_data(data, 256) # sample 256 examples
  weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
  weights += - step_size * weights_grad # perform parameter update
```

   ii. 会产生一些问题，例如实际进展很慢，会产生抖动；出现梯度为0的情况，梯度下降被卡住，梯度来自小批量数据，可能会有噪声
4. SGD + Momentum 动量

i.

SGD

$$x_{t+1} = x_t - \alpha \nabla f(x_t)$$

```
while True:
  dx = compute_gradient(x)
  x -= learning_rate * dx
```

SGD+Momentum

$$v_{t+1} = \rho v_t + \nabla f(x_t)$$
$$x_{t+1} = x_t - \alpha v_{t+1}$$

```
vx = 0
while True:
  dx = compute_gradient(x)
  vx = rho * vx + dx
  x -= learning_rate * vx
```

- Build up “velocity” as a running mean of gradients
- Rho gives “momentum”; typically rho=0.9 or 0.99