第六周

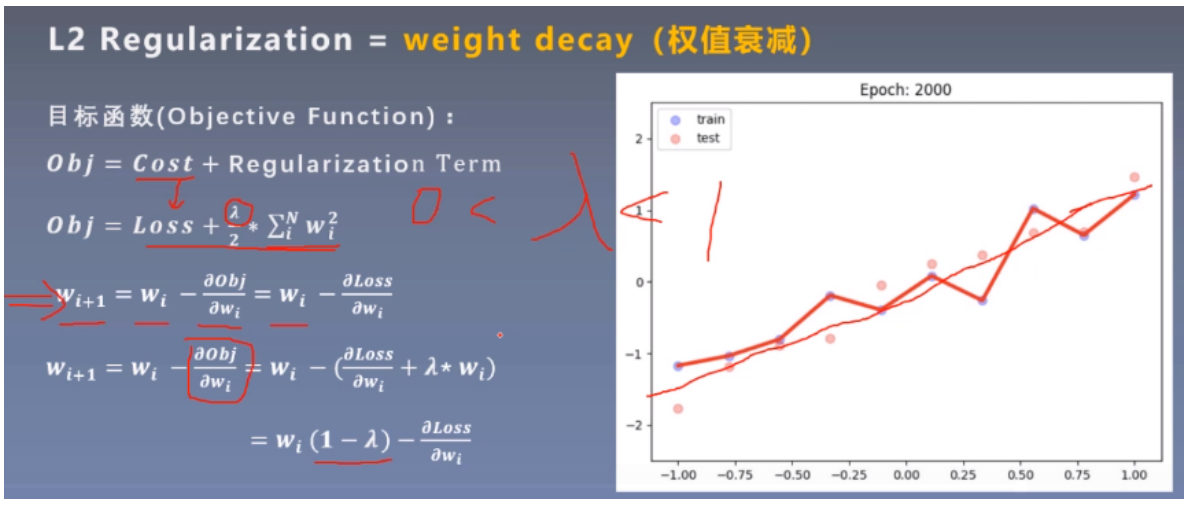
1. 正则化

1.1 正则化与偏差、方差分解

- 正则化：减小方差的策略，高方差过拟合
- 误差 = 偏差 + 方差 + 噪声
  - 偏差度量了学习算法的期望预测与真实结果的偏离程度，即刻画了学习算本身的拟合能力
  - 方差度量了相同大小的训练集的变动所导致的学习性能的变化，即刻画了数据扰动所造成的影响
  - 噪声则表达了在当前任务上任何学习算法所能达到的期望泛化误差的下界

- L1、L2正则化
  - 目标函数：$\text{Obj} = \text{Cost} + \text{Regularization Term}$
  - L1 Regularization Term：$\sum_i^N |w_i|$
  - L2 Regularization Term：$\sum_i^N {w_i}^2$

1.2 pytorch中的L2正则项 —— weight decay



注：在优化器中实现正则化

```
torch.optim.SGD(weight_decay=1e-2)
```

1.3 Dropout概念

- Dropout：随机失活，有一定概率使得神经元权重置零

```
torch.nn.Dropout(p=0.5,     # 失活概率
        inplace=False)
```

1.4 Dropout注意事项

- 数据尺度变化：
  - 模型训练时，由于部分神经元失活，使得参数尺度发生了变化
  - 模型测试时，对应的，所有权重需要乘以 $1-p$ 保持尺度一致
  - 在pytorch中，在模型训练时所有权重乘以 $\frac{1}{1-p}$，模型测试时便不需要乘以 $1-p$，降低了测试的复杂度

```
# 模型进入测试模式
module_name.eval()
# 模型进入恢复训练模式
module_name.train()
```

1.5 作业

- weight decay在pytorch的SGD中实现代码是哪一行？它对应的数学公式为？
  - torch/optim/sgd.py93行
  - $dp = dp + weight\_decay * p.data$
- PyTorch中，Dropout在训练的时候权值尺度会进行什么操作？
  - 模型训练时，由于部分神经元失活，使得参数尺度发生了变化
  - 模型测试时，对应的，所有权重需要乘以 $1-p$ 保持尺度一致
  - 在pytorch中，在模型训练时所有权重乘以$\frac{1}{1-p}$，模型测试时便不需要乘以 $1-p$，降低了测试的复杂度

2. Batch Normalization

2.1 概念

- 批标准化 批：一批数据，通常为mini-batch 标准化：使得数据分布符合0均值、1方差
- 优点 可以使用更大的学习率，加速模型收敛 可以不用精心设计权值初始化 可以不用dropout或使用较小的dropout 可以不用L2正则化或使用较小的weight decay 可以不用LRN（local response normalization）
- 计算方法

2.2 Batch Normalization的1d/2d/3d实现

```
class _BatchNorm():
    __init__(self, num_features,     # 一个样本特征数量
        eps=1e-5,     # 分母修正项
        momentum=0.1,     # 指数加权平均估计当前均值/方差
        affine=True,     # 是否需要affine transform
        track_running_stats=True)     # 训练状态/测试状态
```

```
nn.BatchNorm1d
nn.BatchNorm2d
nn.BatchNorm3d
# 主要属性
running_mean：均值
running_var：方差
weight：affine transform中的gamma
bias：affine transform中的beta
# ------------------ #
训练：均值和方差采用指数加权平均计算
测试：当前统计值
running_mean = (1-momentum)*pre_running_mean + momentum*mean_t
running_var = (1-momentum)*pre_running_var + momentum*var_t
```

```
nn.BatchNorm1d input = B*特征数*1d特征
nn.BatchNorm2d input = B*特征数*2d特征
nn.BatchNorm3d input = B*特征数*3d特征
```

3. 其他标准化方法

3.1 标准化的作用

- Batch Normalization的提出是为了解决ICS问题
  - Internal Covariate Shift：数据尺度/分布异常，导致训练困难

3.2 常见标准化方法

- Batch Normalization(BN)
- Layer Normalization(LN)
  - BN不适用于变长的网络，如RNN
  - 逐层计算均值和方差
  - 无 running_mean和 running_var
  - gamma和 beta是逐元素的

```
nn.LayerNorm(normalized_shape,    # 该层特征形状
        eps=1e-05,    # 分母修正项
        elementwise_affine=True)    # 是否需要affine transform
```

- Instance Normalization(IN)
  - BN不适用于图像生成
  - 逐Instance(channel)计算均值和方差

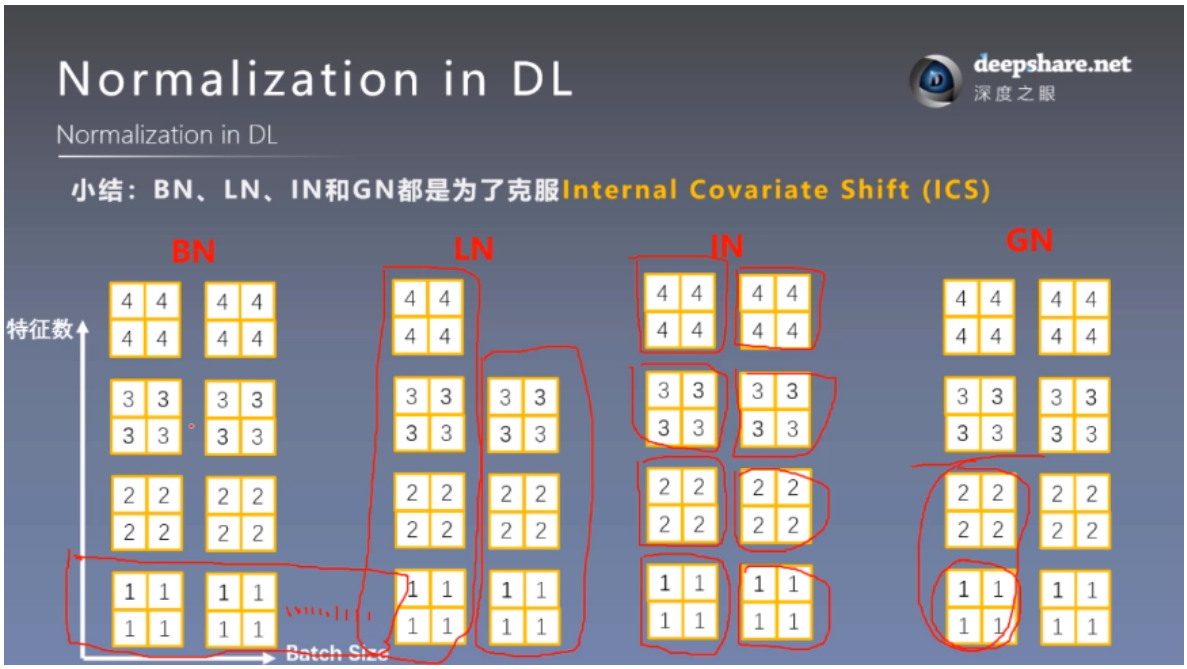
```
nn.InstanceNorm2d(num_features,    # 一个样本特征数量
        eps=1e-05,    # 分母修正项
        momentum=0.1,    # 指数加权平均估计当前均值与方差
        affine=False,    # 是否需要affine transform
        track_runng_stats=False)    # 训练状态/测试状态
```

- Group Normalization(GN)
  - 小batch样本中，BN估计的值不准
  - 用于大模型（小batch size）任务
  - 数据不够，通道来凑
  - 无 running_mean和 running_var
  - gamma和 beta是通道的

```
nn.GroupNorm(num_groups,    # 分组数，2/4/8/16
        num_channels,    # 通道数（特征数）
        eps=1e-05,    # 分母修正项
        affine=True)    # 是否需要affine transform
```

- **注：不同之处在于均值和方差的求取方式**

3.3 小结



3.4 作业

- Batch Normalization 的4个重要参数分别是什么？BN层中采用这个四个参数对X做了什么操作?
  - running_mean：均值
  - running_var：方差
  - weight：affine transform中的gamma
  - bias：affine transform中的beta

- 课程结尾的"加减乘除"是什么意思?
  - 减 μ_B
  - 除 $1/\sqrt{\sigma_B^2+\epsilon}$
  - 乘 γ
  - 加 β

- Layer Normalizatoin、Instance Normalizatoin和Group Normalizatoin的应用场景分别是什么?

- Layer Normalizatoin：用于变长的网络，如RNN
- Instance Normalizatoin：用于图像生成
- Group Normalizatoin：用于batch_size小模型大的任务