



温州大學
WENZHOU UNIVERSITY

深度学习-深度学习实践

黄海广 副教授

2023年03月

- 01** 数据集划分
- 02** 数据集制作
- 03** 数据归一化/标准化
- 04** 正则化
- 05** 偏差和方差

数据集划分

3

训练集 (Training Set) : 帮助我们训练模型, 简单的说就是通过训练集的数据让我们确定拟合曲线的参数。

验证集 (Validation Set) : 也叫做开发集 (Dev Set), 用来做模型选择 (model selection), 即做模型的最终优化及确定的, 用来辅助我们的模型的构建, 即训练超参数, 可选;

测试集 (Test Set) : 为了测试已经训练好的模型的精确度。



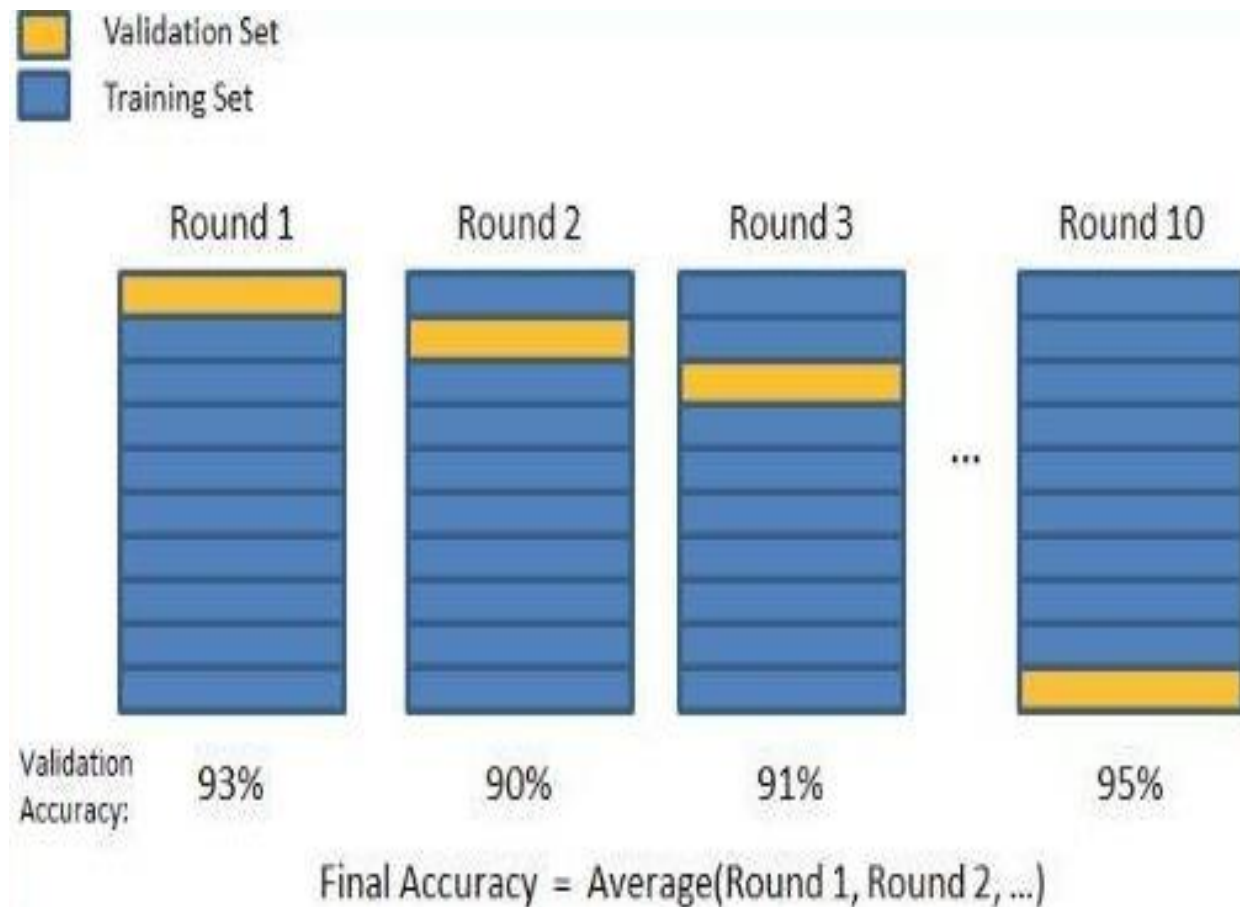
三者划分: 训练集、验证集、测试集

机器学习: 60%, 20%, 20%; 70%, 10%, 20%

深度学习: 98%, 1%, 1% (假设百万条数据)

交叉验证

4



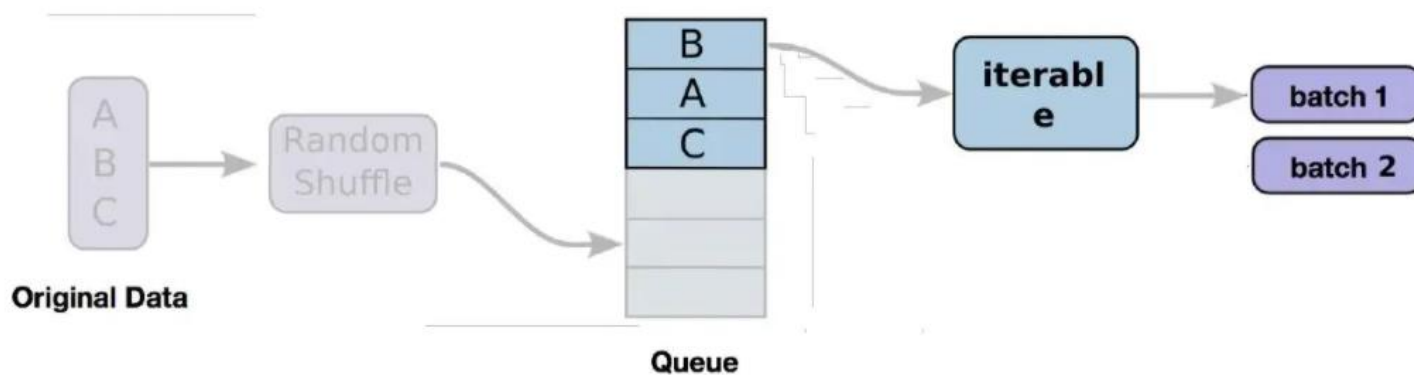
1. 使用训练集训练出10个模型
2. 用10个模型分别对交叉验证集计算得出交叉验证误差（代价函数的值）
3. 选取代价函数值最小的模型
4. 用步骤3中选出的模型对测试集计算得出推广误差（代价函数的值）

数据集制作

5

PyTorch的dataloader是用于读取训练数据的工具，它可以自动将数据分割成小batch，并在训练过程中进行数据预处理。

DataLoader



```
for i, data in enumerate(train_loader, 0):  
    # get the inputs  
    inputs, labels = data  
  
    # wrap them in Variable  
    inputs, labels = Variable(inputs), Variable(labels)  
  
    # Run your training process  
    print(epoch, i, "inputs", inputs.data, "labels", labels.data)
```

数据集制作

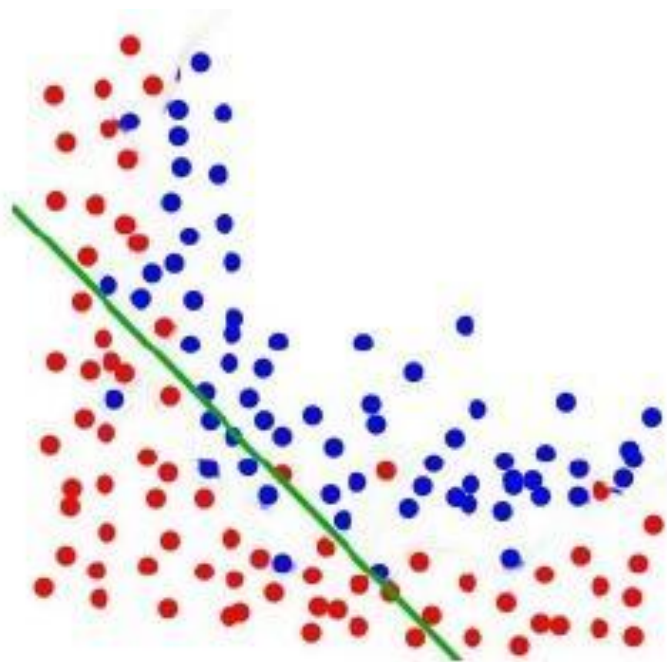
6

```
class MyDataset(Dataset):  
    def __init__(self, data):  
        self.data = data  
  
    def __len__(self):  
        return len(self.data)  
  
    def __getitem__(self, index):  
        # 返回第index个数据样本  
        return self.data[index]
```

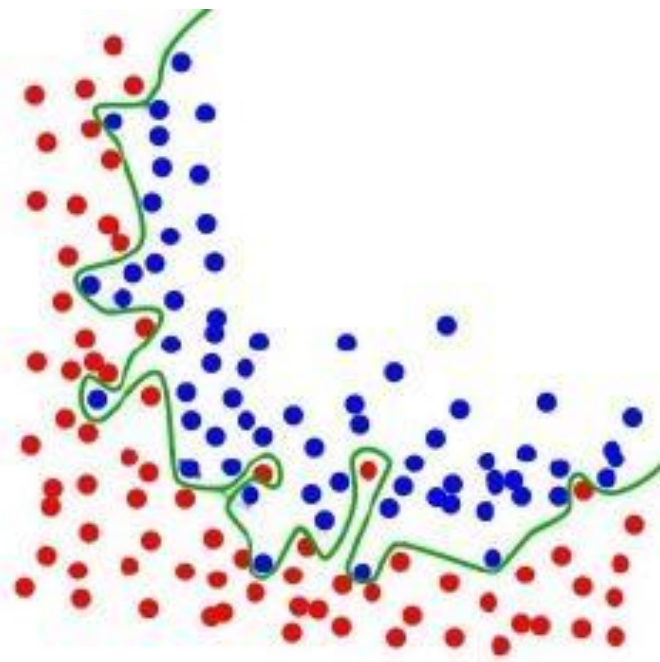
自定义一个继承自torch.utils.data.Dataset的类，在该类中实现__len__和__getitem__方法。

过拟合和欠拟合

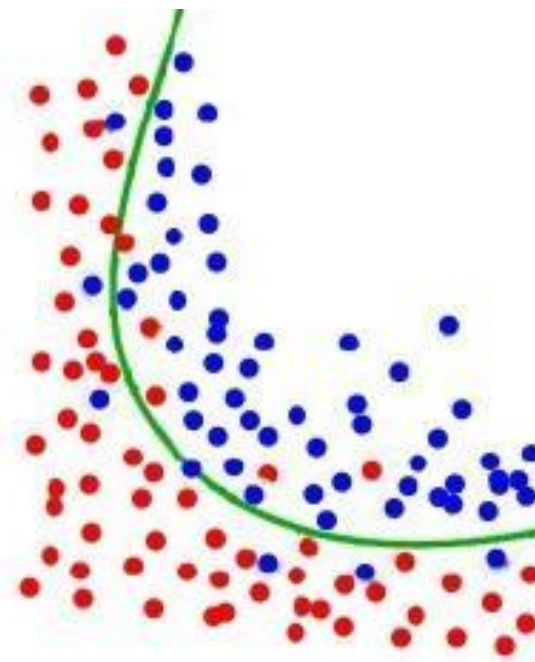
7



欠拟合



过拟合



正合适

过拟合的处理

8

1. 获得更多的训练数据

使用更多的训练数据是解决过拟合问题最有效的手段，因为更多的样本能够让模型学习到更多更有效的特征，减小噪声的影响。

2. 降维

即丢弃一些不能帮助我们正确预测的特征。可以是手工选择保留哪些特征，或者使用一些模型选择的算法来帮忙（例如PCA）。

3. 正则化

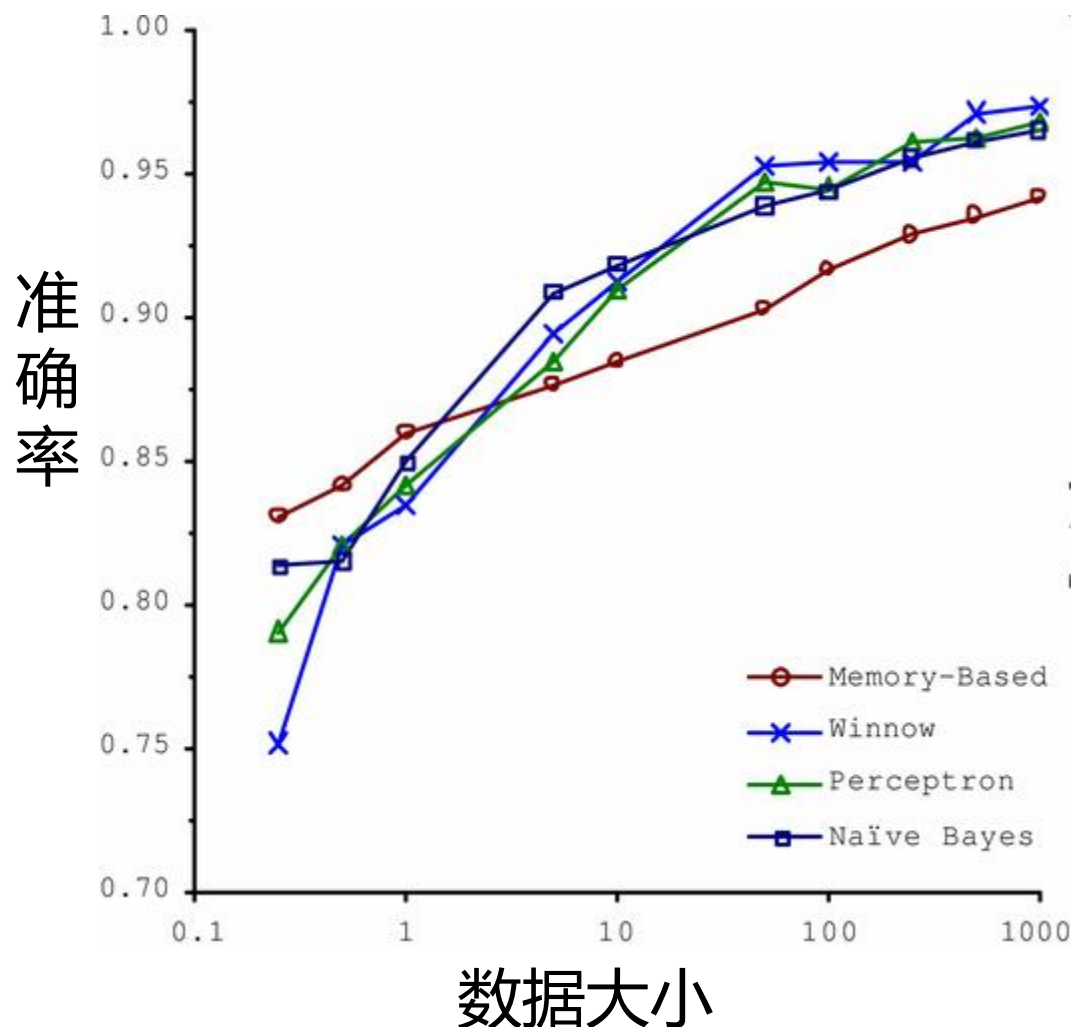
正则化(regularization)的技术，保留所有的特征，但是减少参数的大小（magnitude），它可以改善或者减少过拟合问题。

4. 集成学习方法

集成学习是把多个模型集成在一起，来降低单一模型的过拟合风险。

数据决定一切

9



通过这张图可以看出，各种不同算法在输入的数据量达到一定级数后，都有相近的高准确度。于是诞生了机器学习界的名言：

成功的机器学习应用不是拥有最好的算法，而是拥有最多的数据！

欠拟合的处理

10

1. 添加新特征

当特征不足或者现有特征与样本标签的相关性不强时，模型容易出现欠拟合。通过挖掘组合特征等新的特征，往往能够取得更好的效果。

2. 增加模型复杂度

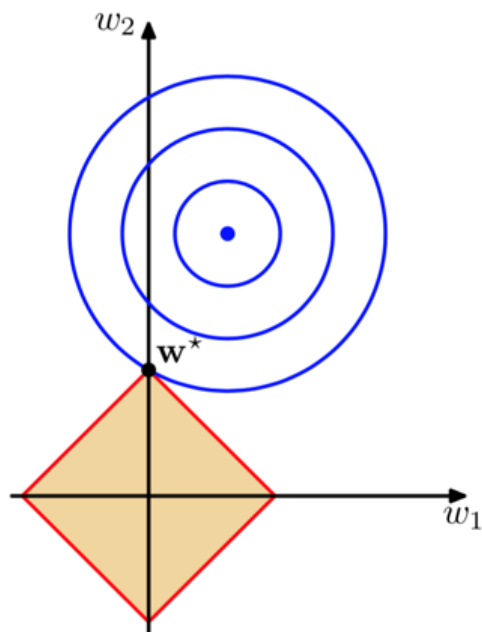
简单模型的学习能力较差，通过增加模型的复杂度可以使模型拥有更强的拟合能力。例如，在线性模型中添加高次项，在神经网络模型中增加网络层数或神经元个数等。

3. 减小正则化系数

正则化是用来防止过拟合的，但当模型出现欠拟合现象时，则需要有针对性地减小正则化系数。

正则化

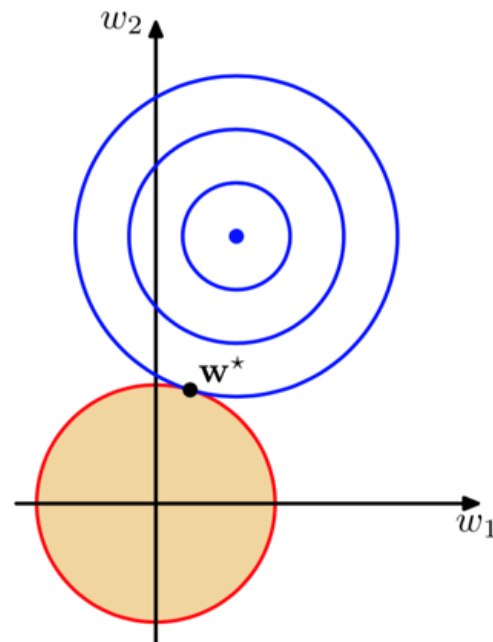
11



L_1 正则化是指在损失函数中加入权值向量 w 的绝对值之和, L_1 的功能是使权重稀疏

L_1 正则化可以产生稀疏模型

$$J(w) = \frac{1}{m} \sum_{i=1}^m L(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m} \sum_{j=1}^n |w_j|$$



在损失函数中加入权值向量 w 的平方和, L_2 的功能是使权重平滑。

L_2 正则化可以防止过拟合

$$J(w) = \frac{1}{m} \sum_{i=1}^m L(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

图上面中的蓝色轮廓线是没有正则化损失函数的等高线, 中心的蓝色点为最优解, 左图、右图分别为 L_1 、 L_2 正则化给出的限制。

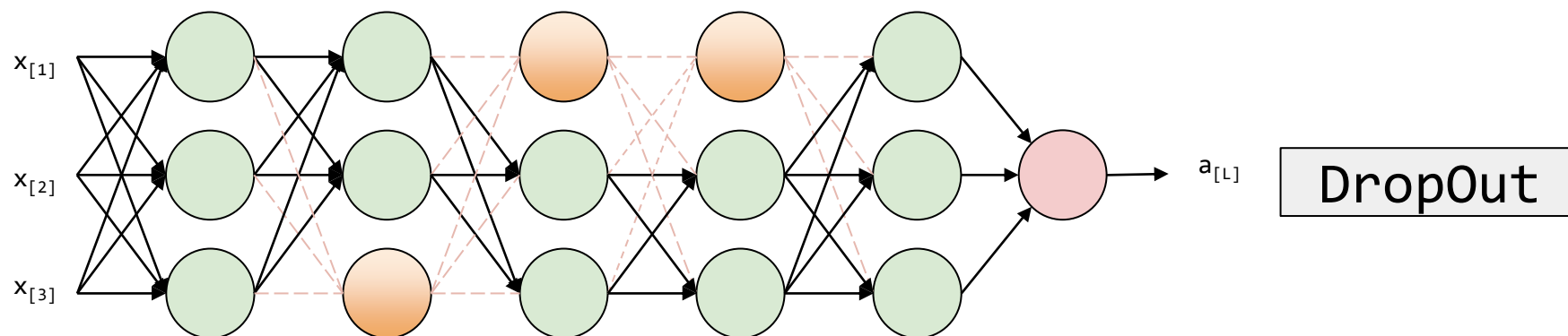
可以看到在正则化的限制之下, L_2 正则化给出的最优解 w^* 是使解更加靠近原点, 也就是说 L_2 正则化能降低参数范数的总和。

L_1 正则化给出的最优解 w^* 是使解更加靠近某些轴, 而其它的轴则为0, 所以 L_1 正则化能使得到的参数稀疏化。

正则化

12

Dropout正则化



Dropout的功能类似于 $L2$ 正则化，与 $L2$ 正则化不同的是，被应用的方式不同，**dropout**也会有所不同，甚至更适用于不同的输入范围

keep-prob=1(没有dropout) **keep-prob=0.5**(常用取值，保留一半神经元)

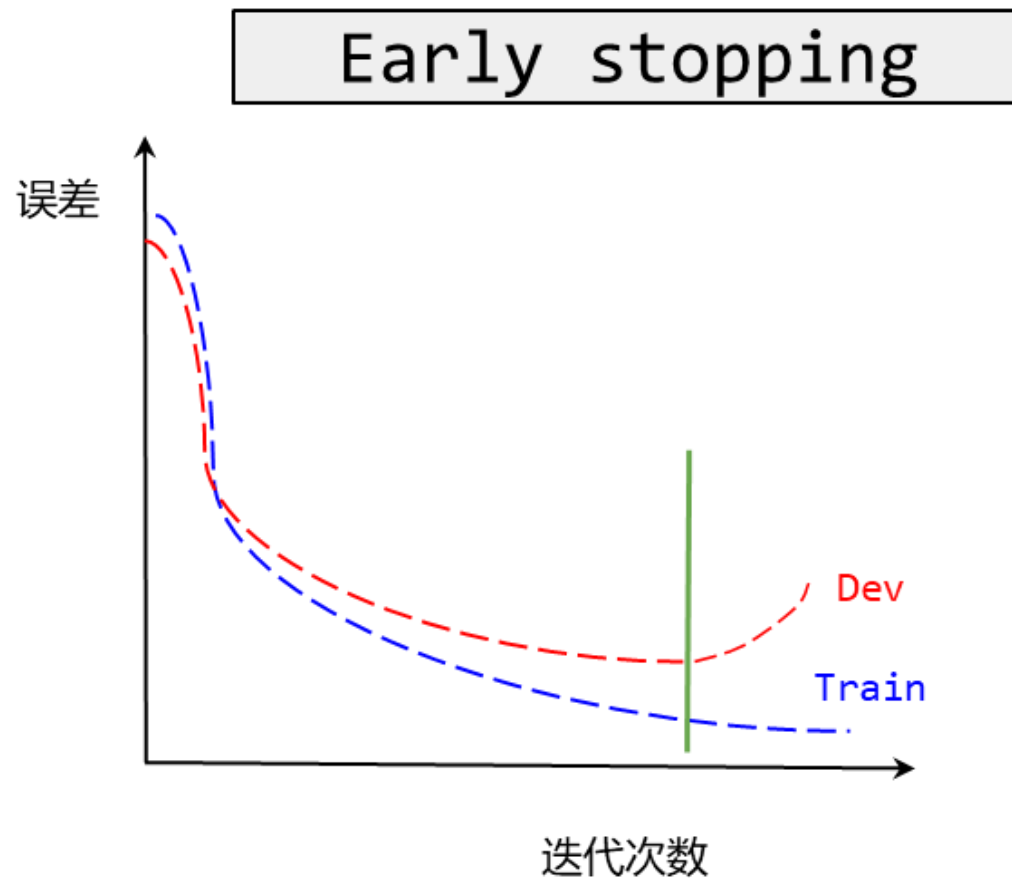
在训练阶段使用，在测试阶段不使用！

正则化

13

Early stopping代表提早停止训练神经网络

Early stopping的优点是，只运行一次梯度下降，你可以找出 w 的较小值，中间值和较大值，而无需尝试 $L2$ 正则化超参数 λ 的很多值。

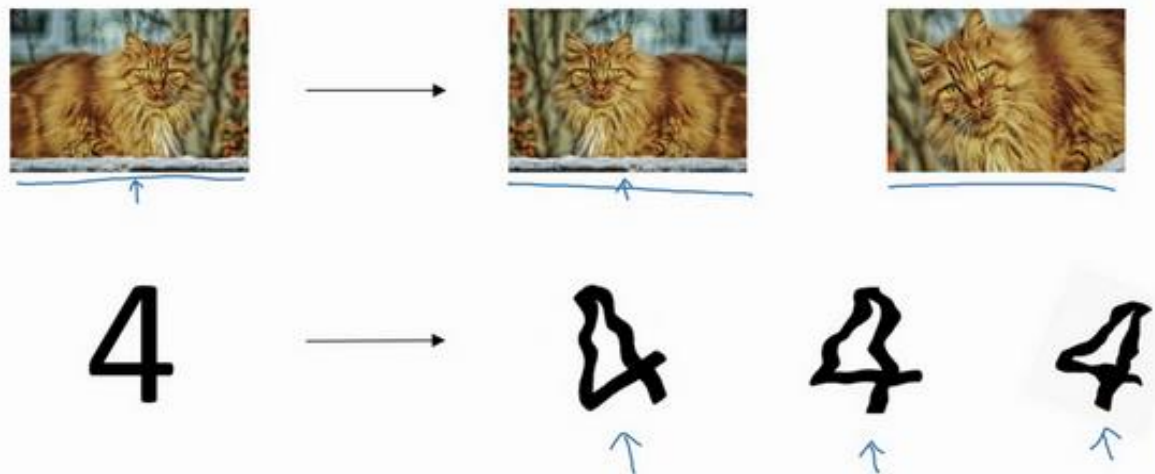


正则化

14

数据增强：随意翻转和裁剪、扭曲变形图片

Data augmentation



数据增强的PyTorch实现

15

```
import torch
from torchvision import transforms

# 定义数据增强的方法
transform = transforms.Compose([
    transforms.RandomResizedCrop(224), # 随机裁剪
    transforms.RandomHorizontalFlip(), # 随机翻转
    transforms.ColorJitter(brightness=0.3, contrast=0.3, saturation=0.3, hue=0.3), # 随机改变颜色
    transforms.RandomRotation(30), # 随机旋转
    transforms.ToTensor(), # 转换为Tensor类型
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]) # 标准化
])

# 加载图像数据
img = Image.open('image.jpg').convert('RGB')

# 对图像进行数据增强
img_aug = transform(img)

# 可以将数据增强的过程添加到数据集的加载器中
dataset = datasets.ImageFolder('data', transform=transform)
dataloader = torch.utils.data.DataLoader(dataset, batch_size=32, shuffle=True)
```

其中RandomResizedCrop是随机裁剪方法

RandomHorizontalFlip是随机翻转方法

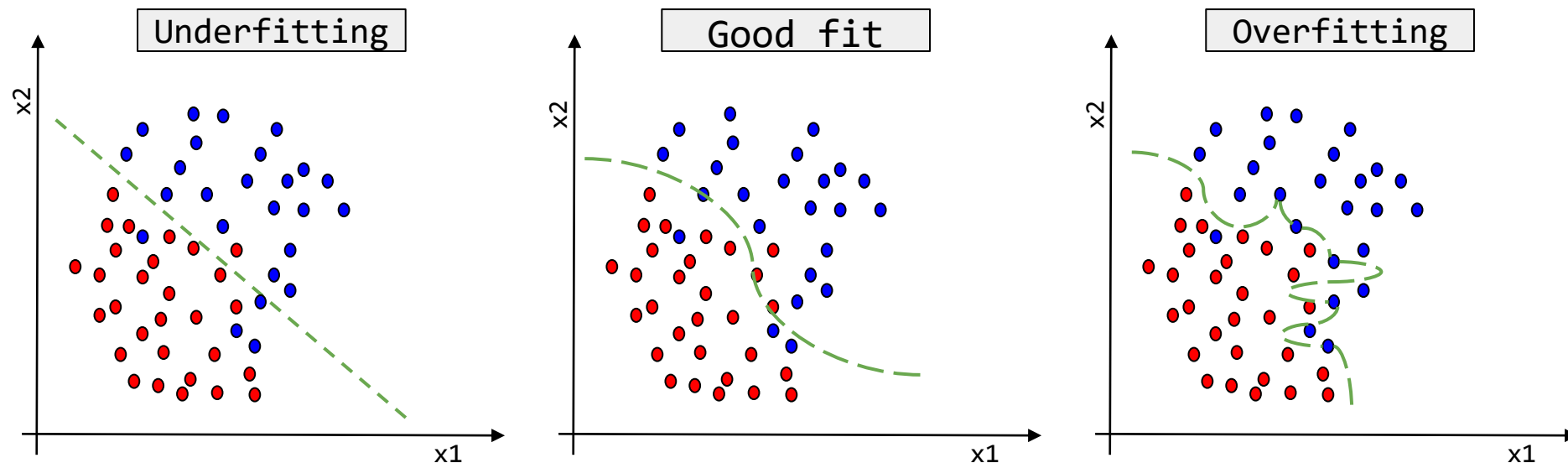
ColorJitter是随机改变颜色方法

RandomRotation是随机旋转方法。最后将图像转换为Tensor类型并进行标准化。

可以将以上方法添加到数据集加载器中进行批量的数据增强。

偏差和方差

16

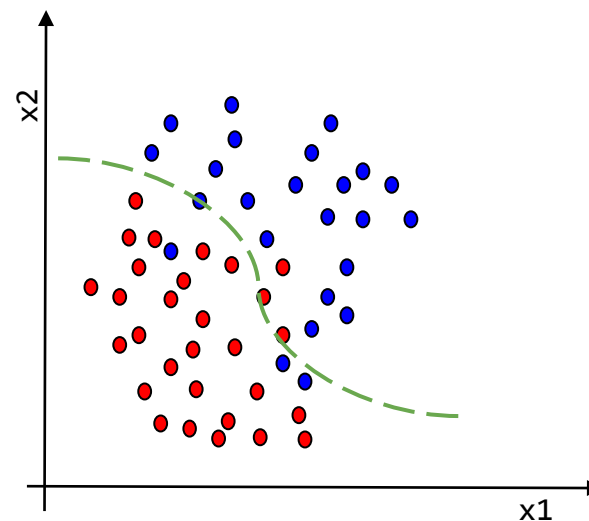


训练集误差和交叉验证集误差近似时：偏差/欠拟合
交叉验证集误差远大于训练集误差时：方差/过拟合

偏差和方差

17

1. 获得更多的训练实例——解决高方差
2. 尝试减少特征的数量——解决高方差
3. 尝试获得更多的特征——解决高偏差
4. 尝试增加多项式特征——解决高偏差
5. 尝试减少正则化程度 λ ——解决高偏差
6. 尝试增加正则化程度 λ ——解决高方差



1. IAN GOODFELLOW等, 《深度学习》, 人民邮电出版社, 2017
2. Andrew Ng, <http://www.deeplearning.ai>

谢谢!

