

第七章 深度学习中的正则化

正则化

- 对学习算法的修改——旨在减少泛化误差而不是训练误差

正则化策略

- 向机器学习模型添加限制参数值的额外约束
- 向目标函数增加额外项来对参数值进行软约束
- 对估计进行正则化，以偏差的增加换取方差的减少

模型族训练的3个情形

- 不包括真实的数据生成过程——对应欠拟合和含有偏差的情况
 - 匹配真实数据生成过程
 - 除了包括真实的数据生成过程，还包括许多其他可能的生成过程——方差（而不是偏差）主导的过拟合
-
- 正则化的目标是使模型从第三种情况转化为第二种情况

7.1 参数范数惩罚

对目标函数 J 添加一个参数范数惩罚 $\Omega(\theta)$

- $\tilde{J}(\theta; \mathbf{X}, \mathbf{y}) = J(\theta; \mathbf{X}, \mathbf{y}) + \alpha\Omega(\theta),$
 - $\alpha \in [0, +\infty)$, 是超参数
 - 正则化后, 减低目标函数训练误差的同时减少参数 θ 的规模
-
- 通常只对权重做惩罚而不对偏置做正则惩罚
 - 正则化偏置参数可能会导致明显的欠拟合
 - 在神经网络的情况下, 为了减少搜索空间, 会在所有层使用相同的权重衰减

7.1.1 L^2 参数正则化

权重衰减

- $\Omega(\theta) = \frac{1}{2} \|w\|_2^2$
 - 使权重更加接近原点
-
- 只有在显著减小目标函数方向上的参数会保留得相对完好
 - 不重要方向对应的分量会在训练过程中因正则化而衰减掉

7.1.2 L^1 参数正则化

L^1 正则化

- $\Omega(\theta) = \|w\|_1 = \sum_i |w_i|$
- 各个参数的绝对值之和

L^1 和 L^2 正则化之间的差异

- L^1 正则化对梯度的影响不再是线性地缩放每个 w_i
- 添加了一项与 $\text{sign}(w_i)$ 同号的常数
- L^1 正则化会产生更稀疏的解
- L^1 正则化导出的稀疏性质已经被广泛地用于特征选择，选择出有意义的特征，化简机器学习问题

7.2 作为约束的范数惩罚

构造一个广义Lagrange 函数来最小化带约束的函数

- $\mathcal{L}(\theta, \alpha; \mathbf{X}, \mathbf{y}) = J(\theta; \mathbf{X}, \mathbf{y}) + \alpha(\Omega(\theta) - k)$
- 求解: $\theta^* = \arg \min_{\theta} \max_{\alpha, \alpha \geq 0} \mathcal{L}(\theta, \alpha)$
- 固定 α^* , $\theta^* = \arg \min_{\theta} \mathcal{L}(\theta, \alpha^*) = \arg \min_{\theta} J(\theta; \mathbf{X}, \mathbf{y}) + \alpha^* \Omega(\theta)$
- 如果 Ω 是 L^2 范数, 那么权重就是被约束在一个 L^2 球中
- 如果 Ω 是 L^1 范数, 那么权重就是被约束在一个 L^1 范数限制的区域中

显式约束和重投影

- 先计算 $J(\theta)$ 的下降步, 然后将 θ 投影到满足 $\Omega(\theta) < k$ 的最近点
- 不想花时间寻找对应于此 k 处的 α 值
- 惩罚可能会导致目标函数非凸
- 对优化过程增加了一定的稳定性

7.3 正则化和欠约束问题

线性回归和PCA

- 正规方程中需要对 $X^T X$ 求逆
 - 如果 $X^T X$ 是奇异的，这些方法就会失效
 - 正则化之后，对应矩阵变成了 $X^T X + \alpha I$ ，可以保证是可逆的
-
- 没有闭式解的问题也可能是欠定的
 - 正则化能够保证应用于欠定问题的迭代方法收敛
 - 可以将伪逆解释为使用正则化来稳定欠定问题

7.4 数据集增强

让机器学习模型泛化得更好的最好办法是使用更多的数据进行训练，因此可以是创建假数据并添加到训练集中

数据集增强对一个具体的分类问题来说是特别有效的方法
对象识别

- 图像每个方向平移几个像素
- 旋转图像或者缩放图像

神经网络

- 在神经网络的输入层注入噪声
- 将随机噪声添加到输入再进行训练

7.5 噪声的鲁棒性

添加噪声，作为数据集增强策略

- 向输入添加方差极小的噪声等价于对权重施加范数惩罚
- 将噪声直接加到权重，加于权重的噪声可以被解释为与更传统的正则化形式等同

向输出目标注入噪声

- 大多数数据集的标签都有一定错误
- 显式地对标签上的噪声进行建模
- 标签平滑通过把确切分类目标从0 和1替换成 $\frac{\epsilon}{k-1}$ 和 $1 - \epsilon$
- 标签平滑的优势是能够防止模型追求确切概率而不影响模型学习正确分类

7.6 半监督学习

深度学习背景下的半监督学习

- 半监督学习通常指的是学习一个表示 $\mathbf{h} = f(\mathbf{x})$
- 学习表示的目的是使相同类中的样本有类似的表示

构建模型

- 生成模型 $P(\mathbf{x})$ 或 $P(\mathbf{x}; y)$ 与判别模型 $P(\mathbf{y} | \mathbf{x})$ 共享参数
- 权衡监督模型准则 $-\log P(\mathbf{y} | \mathbf{x})$ 和无监督或生成模型准则（如 $-\log P(\mathbf{x})$ 或 $-\log P(\mathbf{x}; y)$ ）
- 通过控制在总准则中的生成准则，可以获得比纯生成或纯判别训练准则更好的权衡

7.7 多任务学习

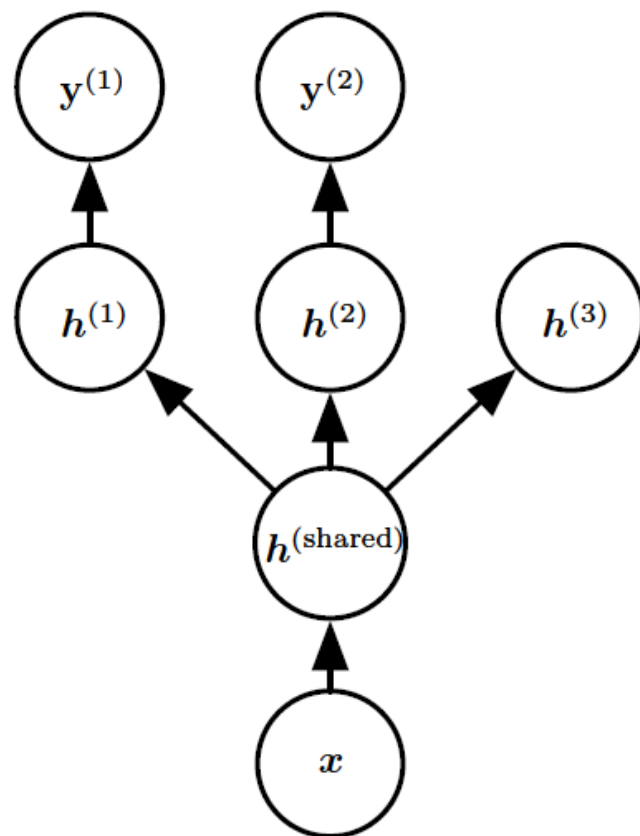
合并几个任务中的样例，提高泛化

当模型的一部分被多个额外的任务共享时，这部分将被约束为良好的值

该模型通常可以分为两类相关的参数

- 具体任务的参数
- 所有任务共享的通用参数

深度网络的较低层可以跨这样的任务共享而任务特定的参数（分别与从 $h^{(1)}$ 和 $h^{(2)}$ 进入和发出的权重）可以在共享表示 $h^{(\text{shared})}$ 上学习。在该示例中，额外假设顶层隐藏单元 $h^{(1)}$ 和 $h^{(2)}$ 专用于每个任务（分别预测 $y^{(1)}$ 和 $y^{(2)}$ ），而一些中间层表示 $h^{(\text{shared})}$ 在所有任务之间共享。



7.8 提前终止

当验证集上的误差在事先指定的循环次数内没有进一步改善时，算法就会终止

算法 7.1 用于确定最佳训练时间量的提前终止元算法。这种元算法是一种通用策略，可以很好地在各种训练算法和各种量化验证集误差的方法上工作。

令 n 为评估间隔的步数。

令 p 为“耐心 (patience)”，即观察到较坏的验证集表现 p 次后终止。

令 θ_0 为初始参数。

$\theta \leftarrow \theta_0$

$i \leftarrow 0$

$j \leftarrow 0$

$v \leftarrow \infty$

$\theta^* \leftarrow \theta$

$i^* \leftarrow i$

while $j < p$ do

 运行训练算法 n 步，更新 θ 。

$i \leftarrow i + n$

$v' \leftarrow \text{ValidationSetError}(\theta)$

 if $v' < v$ then

$j \leftarrow 0$

$\theta^* \leftarrow \theta$

$i^* \leftarrow i$

$v \leftarrow v'$

 else

$j \leftarrow j + 1$

 end if

end while

最佳参数为 θ^* ，最佳训练步数为 i^*

7.8 提前终止

提前终止是非常高效的超参数选择算法

- 训练步骤仅仅是另一个超参数

提前终止自动选择超参数的代价

- 训练期间要定期评估验证集
- 额外代价是需要保持最佳的参数副本

提前终止是一种非常不显眼的正则化形式，无需破坏学习动态就能很容易地使用提前终止

提前终止需要验证集，这意味着某些训练数据不能被馈送到模型

- 再次初始化模型，然后使用所有数据再次训练
- 保持从第一轮训练获得的参数，然后使用全部的数据继续训练

提前终止可以将优化过程的参数空间限制在初始参数值 θ_0 的小邻域内

7.8 提前终止

提前终止需要验证集，这意味着某些训练数据不能被馈送到模型

- 再次初始化模型，然后使用所有数据再次训练
- 保持从第一轮训练获得的参数，然后使用全部的数据继续训练

算法 7.2 使用提前终止确定训练步数，然后在所有数据上训练的元算法。

令 $\mathbf{X}^{(\text{train})}$ 和 $\mathbf{y}^{(\text{train})}$ 为训练集。

将 $\mathbf{X}^{(\text{train})}$ 和 $\mathbf{y}^{(\text{train})}$ 分别分割为 $(\mathbf{X}^{(\text{subtrain})}, \mathbf{X}^{(\text{valid})})$ 和 $(\mathbf{y}^{(\text{subtrain})}, \mathbf{y}^{(\text{valid})})$ 。

从随机 θ 开始，使用 $\mathbf{X}^{(\text{subtrain})}$ 和 $\mathbf{y}^{(\text{subtrain})}$ 作为训练集， $\mathbf{X}^{(\text{valid})}$ 和 $\mathbf{y}^{(\text{valid})}$ 作为验证集，运行 (算法 7.1)。这将返回最佳训练步数 i^* 。

将 θ 再次设为随机值。

在 $\mathbf{X}^{(\text{train})}$ 和 $\mathbf{y}^{(\text{train})}$ 上训练 i^* 步。

7.8 提前终止

提前终止需要验证集，这意味着某些训练数据不能被馈送到模型

- 再次初始化模型，然后使用所有数据再次训练
- 保持从第一轮训练获得的参数，然后使用全部的数据继续训练

算法 7.3 使用提前终止确定将会过拟合的目标值，然后在所有数据上训练直到再次达到该值的元算法。

令 $\mathbf{X}^{(\text{train})}$ 和 $\mathbf{y}^{(\text{train})}$ 为训练集。

将 $\mathbf{X}^{(\text{train})}$ 和 $\mathbf{y}^{(\text{train})}$ 分别分割为 $(\mathbf{X}^{(\text{subtrain})}, \mathbf{X}^{(\text{valid})})$ 和 $(\mathbf{y}^{(\text{subtrain})}, \mathbf{y}^{(\text{valid})})$ 。

从随机 θ 开始，使用 $\mathbf{X}^{(\text{subtrain})}$ 和 $\mathbf{y}^{(\text{subtrain})}$ 作为训练集， $\mathbf{X}^{(\text{valid})}$ 和 $\mathbf{y}^{(\text{valid})}$ 作为验证集，运行 (算法 7.1)。这会更新 θ 。

$\epsilon \leftarrow J(\theta, \mathbf{X}^{(\text{subtrain})}, \mathbf{y}^{(\text{subtrain})})$

while $J(\theta, \mathbf{X}^{(\text{valid})}, \mathbf{y}^{(\text{valid})}) > \epsilon$ **do**

 在 $\mathbf{X}^{(\text{train})}$ 和 $\mathbf{y}^{(\text{train})}$ 上训练 n 步。

end while

7.9 参数绑定和参数共享

有些分类任务会足够相似

- 或许具有相似的输入和输出分布
- 我们认为模型参数应彼此靠近，可以通过正则化利用此信息
- 正则化一个模型（监督模式下训练的分类器）的参数，使其接近另一个无监督模式下训练的模型（捕捉观察到的输入数据的分布）的参数
- 构造的这种架构使得分类模型中的许多参数能与无监督模型中对应的参数匹配

参数共享

- 强迫某些参数相等
- 只有参数（唯一一个集合）的子集需要被存储在内存中

卷积神经网络

- CNN通过在图像多个位置共享参数来处理自然图像
- 参数共享显著降低了CNN模型参数数量

7.10 稀疏表示

惩罚神经网络中的激活单元，稀疏化激活单元

- 表示的正则化可以使用参数正则化中同种类型的机制实现
- $\tilde{J}(\theta; \mathbf{X}, \mathbf{y}) = J(\theta; \mathbf{X}, \mathbf{y}) + \alpha \Omega(\mathbf{h})$
- \mathbf{h} 是 \mathbf{x} 的一个函数
- L^1 惩罚诱导稀疏的表示: $\Omega(\mathbf{h}) = \|\mathbf{h}\|_1 = \sum_i |h_i|$

其他方法:

- 从表示上的Student-t 先验导出的惩罚(Olshausen and Field, 1996; Bergstra, 2011)
- KL 散度惩罚(Larochelle and Bengio, 2008b)
- 正交匹配追踪Pati et al., 1993)

7.11 Bagging 和其他集成方法

Bagging通过结合几个模型降低泛化误差的技术

- 分别训练几个不同的模型
- 让所有模型表决测试样例的输出
- 模型平均
- 采用这种策略的技术被称为集成方法
- 不同的模型通常不会在测试集上产生完全相同的误差
- 集成平方误差的期望会随着集成规模增大而线性减小

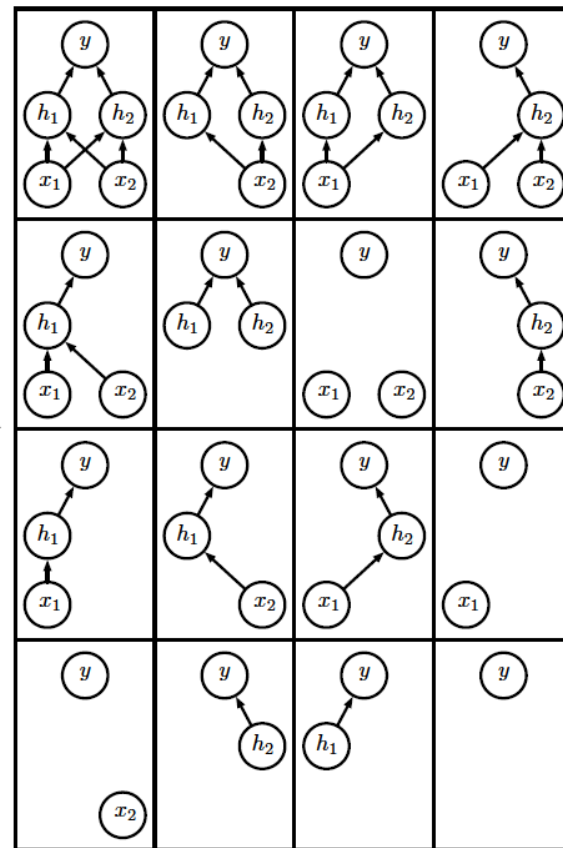
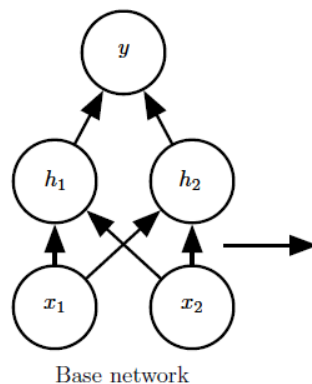
Bagging

- 构造 k 个不同的数据集
- 模型 i 在数据集 i 上训练
- 神经网络中随机初始化的差异、小批量的随机选择、超参数的差异或不同输出的非确定性实现往往足以使得集成中的不同成员具有部分独立的误差

7.12 Dropout

Dropout

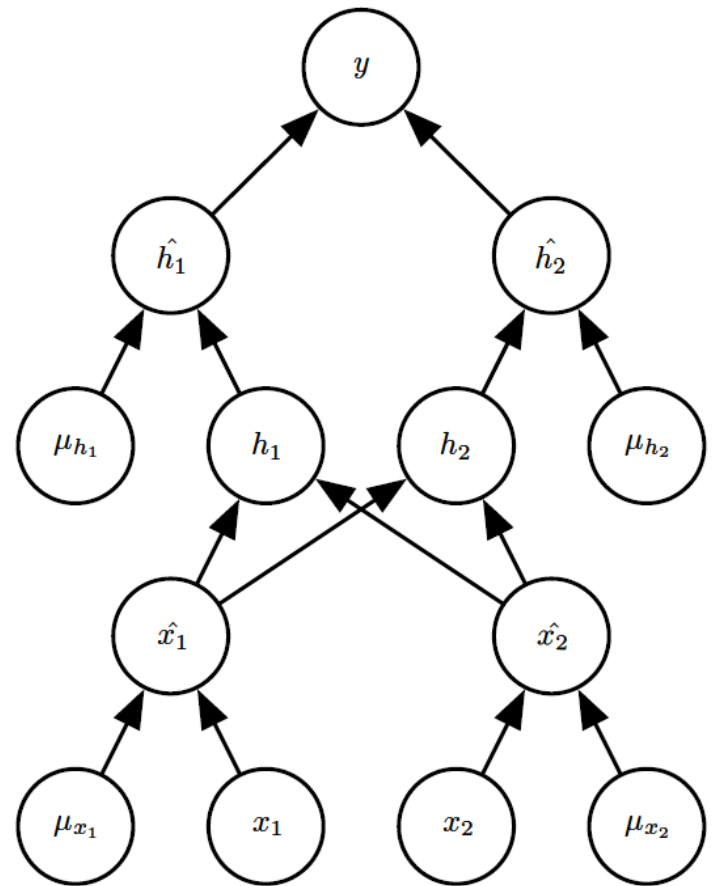
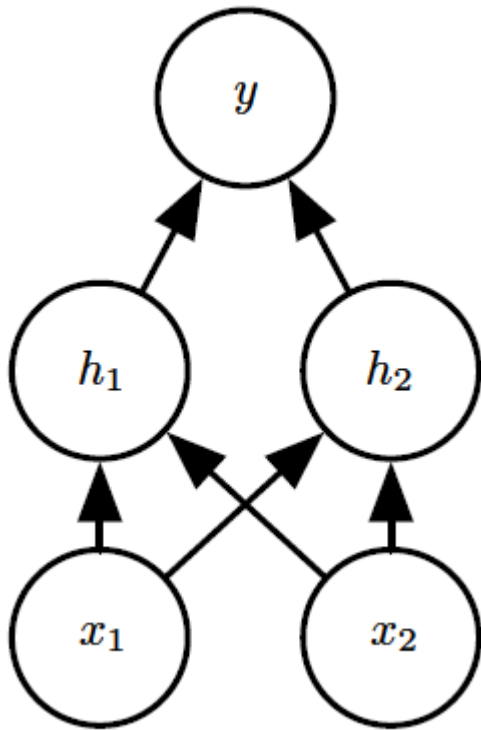
- 提供了正则化一大类模型的方法
- 是集成大量深层神经网络的实用Bagging方法
- 提供了一种廉价的Bagging集成近似，能够训练和评估指数级数量的神经网络
- 子网络通过从基本网络中删除非输出单元构建



7.12 Dropout

Dropout

- 为了执行具有Dropout的前向传播，我们随机地对向量进行采样，其中网络中的每个输入或隐藏单元对应一项



7.12 Dropout

Dropout和Bagging

- 在Bagging的情况下，所有模型都是独立的
- 在Dropout的情况下，所有模型共享参数
- 在Bagging的情况下，每一个模型在其相应训练集上训练到收敛
- 在Dropout的情况下，通常大部分模型都没有显式地被训练
- 在Bagging的情况下，每个模型 i 产生一个概率分布 $p^{(i)}(y | \mathbf{x})$ ，集成的预测由这些分布算术平均值给出

$$\frac{1}{k} \sum_{i=1}^k p^{(i)}(y | \mathbf{x})$$

- 在Dropout的情况下，通过掩码 $\boldsymbol{\mu}$ 定义每个子模型的概率分布 $p(y | \mathbf{x}; \boldsymbol{\mu})$ 。所有掩码的算术平均值由下式给出

$$\sum_{\boldsymbol{\mu}} p(\boldsymbol{\mu}) p(y | \mathbf{x}, \boldsymbol{\mu}).$$

其中 $p(\boldsymbol{\mu})$ 是训练时采样 $\boldsymbol{\mu}$ 的概率分布

7.12 Dropout

权重比例推断规则

- 模型具有所有单元
- 将单元 i 的输出的权重乘以单元 i 的被包含概率
- 通常使用 $\frac{1}{2}$ 的包含概率
- 权重比例规则一般相当于在训练结束后将权重除2，然后像平常一样使用模型

权重比例推断规则在其他设定下也是精确的，包括条件正态输出的回归网络以及那些隐藏层不包含非线性的深度网络

7.12 Dropout

Dropout优点

- 比其他标准的计算开销小的正则化方法更有效
- 计算方便，每个样本每次更新只需 $O(n)$ 的计算复杂度
- 不怎么限制适用的模型或训练过程




Dropout缺点

- 在一个完整的系统上使用Dropout的代价可能非常显著
- 减少了模型的有效容量，为了抵消这种影响，必须增大模型规模
- 只有极少的训练样本可用时，Dropout不会很有效

Dropout强大的原因

- 大部分原因来自施加到隐藏单元的掩码噪声
- 噪声是乘性的

7.13 对抗训练

	$+ .007 \times$		$=$	
x		$\text{sign}(\nabla_x J(\theta, x, y))$		$x + \epsilon \text{sign}(\nabla_x J(\theta, x, y))$
$y = \text{"panda"}$		"nematode"		"gibbon"
w/ 57.7%		w/ 8.2%		w/ 99.3 %
confidence		confidence		confidence

通过优化过程故意构造数据点，对抗样本

通过对抗训练减少原有独立同分布的测试集的错误率

由训练好的模型提供标签产生的对抗样本被称为虚拟对抗样本

7.14 切面距离、正切传播和流形正切分类器

切面距离

- 一种非参数的最近邻算法
- 假设尝试分类的样本和同一流形上的样本具有相同的类别
- 流形之间的最近距离计算起来很困难，可以使用切平面距离代替

正切传播

- 训练带有额外惩罚的神经网络分类器，使神经网络的每个输出 $f(x)$ 对已知的变化因素是局部不变的
- 要求 $\nabla_x f(x)$ 与已知流形的切向 $v^{(i)}$ 正交
- 只有极少的训练样本可用时，Dropout 不会很有效

流形正切分类器

- 无需知道切线向量的先验
- 算法相当简单
 - 使用自编码器通过无监督学习来学习流形的结构
 - 如正切传播一样使用这些切面正则化神经网络分类器