layout: post title: 深度学习：参数正规化 description: deep learning基础 category: blog

7.1 概念

深度学习中用以减小测试误差，但可能会增加训练误差的策略称为正规化。

限制：有些正规化策略是机器学习模型上添加限制。有些在模型参数上，有些在目标函数上添加额外项。有些限制或惩罚被设计来对特定先验知识编码的，其余的则是为了提高模型泛化能力。

深度学习中大部分正规化策略都是基于估计正规化，而估计正规化则是通过增加偏置来减少方差。(>比如，神经网络中的神经元都有一个偏置项b<)。 一个估计的正规化的目标是在大幅度减小方差的同时，尽可能小的带来偏置的增加。我们在讨论泛化和过拟合问题时会遇到以下三种情形:

图7.1

- **欠拟合**:实际的正例没有完全被包含在模型预测域。
- **绝佳**：完全匹配了数据生成过程.(>完全匹配了模拟的函数<)
- **过拟合**：模型的预测域包含了全部的正例，同时也包含了负例（）。

正规化的目标就是将模型的**过拟合**情形改善至**绝佳**情形。

现实情况中，即便是极端复杂的模型也没法完全拟合目标函数(>让神经网络学习人对图像的认知能力，模型已经达到一千层，参数几千万<)，因为大部分情况，我们并不知道目标函数具体是如何映射的。 这表明设计一个模型的复杂度极高(模型大小，参数等)。而在近些年的实际实验中，我们发现比较好的模型都是正规化处理过后的大模型。

7.2 参数规范惩罚

目前许多正规化方法，如神经网络、线性回归、logistic回归通过在目标函数J上加一个参数规范惩罚项 $\Omega(\theta)$ 公式如下:

$$ \bar{J}(\theta;X,y) = J(\theta;X,y)+\alpha\Omega(\theta)\\tag {7.1}

其中 \alpha\epsilon [0,\infty) $$

其中，更大的 α 对应更强的正规化处理。

在神经网络中，使用参数规范惩罚，只是对每一层映射转换的权重，不对偏置使用。这是因为权重决定了两个变量如何交互(>神经元如何输出<)，而偏置只作用于单一变量。同时正规化偏置容易引入欠拟合。

预定义

- 向量w代表所有需要被规范惩罚的权重
- 向量 θ代表所有参数，包括w和其他非正规化的参数

尽管每一层使用独立的 α参数的惩罚机制效果可能会更好，但是由于计算量太大。实际中所有层使用相同的权重衰减。

7.2.1 L^2 参数正规化

最简单最常见的正规惩罚莫过于L^2，有时候称为*权重衰减*，同时也称为*岭回归*或者*吉洪诺夫正规*。它是直接在目标函数后面添加一个正规项 $\Omega(\theta)=\frac{1}{2}||w||^2_2$

L2范数是指向量各元素的平方和然后求平方根。我们让L2范数的规则项 $||W||_2$最小，可以使得W的每个元素都很小，都接近于0，但与L1范数不同，它不会让它等于0，而是接近于0。

回头看式子 (7.1)，假设没有偏置参数，因此 θ参数就是w,模型目标函数如下: $$ \bar J(w;X,y) =\frac{\alpha}{2}w^Tw+J(w;X,y) \tag{7.2} $$

对应的参数梯度如下:

$$ \bigtriangledown\bar J(w;X,y)=\alpha w+\bigtriangledown_wJ(w;X,y) $$

使用梯度更新权重时:

$$ w\leftarrow w-\epsilon(\alpha w+\bigtriangledown_wJ(w;X,y)) $$

重写为:

$$ w\leftarrow w-(1-\epsilon\alpha)w -\epsilon\bigtriangledown_wJ(w;X,y)) $$

可以看到新增权重衰减项最终反映出，它通过对每一步的权重向量乘以一个常数因子修改了学习规则。由于 ϵ ， α都是正值，所以它实际是减小了w。这就是它被称为**权重衰减**的原因。

如何防止过拟合:更小的权值w，从某种意义上说，表示网络的复杂度更低，对数据的拟合刚刚好,而在实际应用中，也验证了这一点，L2正则化的效果往往好于未经正则化的效果。

过拟合的时候，拟合函数的系数往往非常大，为什么？如下图(图7.2)所示，过拟合，就是拟合函数需要顾及每一个点，最终形成的拟合函数波动很大。在某些很小的区间里，函数值的变化很剧烈。这就意味着函数在某些小区间里的导数值（绝对值）非常大，由于自变量值可大可小，所以只有系数足够大，才能保证导数值很大。

图7.2

而正则化是通过约束参数的范数使其不要太大，所以可以在一定程度上减少过拟合情况。

7.2.2 L^1正规化处理

对模型参数w上的L^1正归化是增加一个惩罚项

$$ \Omega(\theta) = ||w||_1=\sum_i|w_i| $$

完整公式如下：

$$ \bar J(w;X,y) =\alpha||w||_1+J(w;X,y) $$

对上式求梯度得到：

$$ \bigtriangledown _w \bar J(w;X,y) = \alpha sign(w)+\bigtriangledown _w J(X,y;w) $$

可以看到，L^1正规化对梯度的影响不再与每个w_i线性相关，而是一个常量因子。符号只与w相关，当w为正时，更新后的w变小。当w为负时，更新后的w变大——因此它的效果就是让w往0靠，使网络中的权重尽可能为0，也就相当于减小了网络复杂度，防止过拟合。

当w为0时怎么办？当w等于0时，|W|是不可导的，所以我们只能按照原始的未经正则化的方法去更新w，这就相当于去掉 $\alpha sign(w)$ 这一项，所以我们可以规定$sgn(0)=0$，这样就把w=0的情况也统一进来了

7.3 规范惩罚用作约束优化

考虑一个使用了参数规范惩罚正则化的损失函数 $$ \bar J(\theta;X,y) = J(\theta;X,y)+\alpha \Omega(\theta) $$ 我们可以通过构建一个拉格朗日函数求在约束条件下目标函数的最小值，构建函数即在原始目标函数上添加一些惩罚项。假若我们想要约束条件 $\Omega(\theta)$ 小于一个常量k,可以构建如下拉格朗日函数: $$ L(\theta,\alpha;X,y) =J(\theta;X,y)=\alpha(\Omega(\theta)-k) $$

7.4 正则化和受限问题

机器学习中正则化是十分必要的，许多依赖转置矩阵 X^TX 如线性回归模型，主成分分析(PCA)模型，但是若 X^TX 是奇异的（行列式为0，有无穷个解），就没法实现。当数据在某一些方向上没有方差（所有的值相同）时，矩阵是奇异的。此时需要进行相对应的正则化来保证矩阵是可逆的。

当线性问题的相关矩阵是可逆时才有封闭解。欠定方程也可能没有封闭解。一个例子是，逻辑回归用于线性可分类问题时，如果权重 w 可获得最佳分类结果，那么$2w$ 也可以，而且使用最大似然框架时，似然度更高。进行迭代优化如使用SGD(随机梯度下降)时将会导致 w 不断增加，而算法不会停止（实际中会产生数值溢出）。

大多数正则化可以保证迭代方法收敛，比如权重衰减(L^2 正则化)中似然函数梯度等于权重衰减系数时停止。？

7.5 数据增强

理论上来说，数据越多，模型训练得越充分，模型泛化能力越强。但是现实情况是，数据量总是有限的，解决此问题的一个方法是生成一部分的模拟数据。

这对于分类问题最简单，它需要喂入复杂的高维度数据输入并映射到一个单一分类上。这说明，分类问题主要面临的是对于任意广度的输入其分类结果不变。我们可以简单的生成一个 (x,y) 即可。但是对于很多其他问题，如密度估计问题，很难生成模拟数据，除非已经知道需要解决的密度估计问题。

数据增强用于特定领域分类问题，如图像识别很有效。（举例）但是切记，转换数据的时候不要改变图像的正确分类。比如不要将手写字识别图像中的`6`垂直转换成了`9`，`b`水平翻转成了`d`。

语音识别问题中，网络输入数据中也会注入一些随机噪音干扰，这也是一种数据增强（现实生活中语音环境有噪音）。

神经网络对噪音鲁棒性并不好，所以我们在可以有一种提升网络性能的方法，即在训练时加入随机干扰。

由于数据增强的存在，我们在比较机器学习结果时应当考虑数据增强。手工设计的数据通常可以大幅度减少泛化错误（图像分类，如果知道分类结果，制造些类似的图像，然后给分类训练）。所以，在比较机器学习算法时，需要做对照试验，使用相同算法，一组使用没有应用数据增强的输入A，另一组使用应用了数据增强的B，如果A的性能很差，而B的性能很好，那么说明促使模型性能提升的不是算法而是数据。

7.6 噪声鲁棒

对于使用了数据增强的模型，噪音其实等同于在权重分布上增加了惩罚机制。通常，噪音注入比简单的收缩参数(正则化)更有用，尤其是噪音作用于隐藏神经元时，dropout就是专门在方面的进展。

在RNN中，权重上增加噪声被证明是一种很有效的正则策略（书中论文）。接下来分析下标准前馈神经网络中权重噪音的实际影响。

在回归问题中，假设损失函数是最小平方误差，如下:

$$ J = E_{p(x,y)}[(\bar y(x)-y)^2] $$ 假设输入中包含随机扰动 $\epsilon_w \sim N(\epsilon ;0,\eta I)$ ，此时对应的目标函数变成:

$$ \bar J_W=E_{p(x,y,\epsilon *w)}[(\bar y*{\epsilon_W}(x)-y)^2] \ =E_{p(x,y,\epsilon_W)}[\bar y^2_{\epsilon_W}(x)-2y\bar y_{\epsilon_W}(x)+y^2] $$

若 η 很小,最小化 J 等同于最小化 J 外加一个正则项 $\eta E_{p(x,y)}[||\bigtriangledown_W\bar y(x)||^2]$ 这种正则项使得模型对参数的轻微扰动不再敏感，此时的最优参数不在使得损失函数最小点而是在最小点附近。

7.6.1 在输出目标上注入噪音

模型的错误分类将导致最大似然框架求得的 $logP(y|x)$ 并不是真正最大点。一种办法是在标签上加入噪音，例如常量 θ ，此时训练数据集x，其输出被分类到标签y的概率为 $1-\theta$ ，这种方法很容易并入到惩罚函数，而不需要引入噪声数据。这是一种平滑机制，比如标签正则模型基于有k个输出的 *softmax* ，将分类 `0`, `1` 替换为 $\frac{\theta}{k}$ 和 $1-\frac{k-1}{k}\theta$

7.7 多任务学习

多任务学习可以看做一种从多个模型中抽象出一个汇总模型以提高泛化能力的方法。模型的某个部分的参数在多个任务间共享时，该部分将获得更好的泛化能力。

下图(图7.3)展示了一个多任务学习方法的常见形式，不同的监督学习任务(对于给定输入X预测输出Y)，共享了相同的输入X，以及一些中间表述层 h^{shared} (捕获参数的一些平均特征)。

图7.3

通过提升这些共享参数的统计特性（更稳定），可以提高模型泛化能力和泛化边界。与单任务学习相比，其实是按比例增加了共享参数的输入样本。当然前提是，多个模型之间可以共享参数。

以深度学习的观点来看，这种方法的先验知识是：不同模型的输入数据有些解释了数据变动的参数是在多个任务中共享的。

7.8 提前终止

看一张图(图7.4)：

图7.4

我们可以看到随着时间或迭代次数的增加，训练误差不断减少，而验证误差最后会逐渐上升，呈U型。验证误差开始上升时，已经出现了过拟合。我们应当在验证误差有段时间没有下降时停止迭代，而不是等到验证误差达到某个极小值时。此策略即提前终止，是正则化策略中最常见，最有效的方法。

如何确定何时终止：

- 在算法开始之前，先确定训练次数。
- 在训练过程中定期地运行验证，验证集可以比训练集数据量小。(caffe模型)

优点：

- 几乎不改变算法过程，易于使用
- 提前终止可以很容易地与其他正则化方法结合使用。

代价：

- 需要不断保存最优模型参数，这是可以接收的，可以直接存放在磁盘上。
- 需要一个验证数据集，验证数据集不用于训练。

最大化利用所有数据

为了更好的利用所有数据，可以在算法提前终止后再次训练。此时有两种策略

- 重新初始化模型，在所有数据集上重新训练，但只训练提前终止训练的次数。此时可以增加一些参数，因为数据更多（个人认为，其实是分两步走，第一步

是用提前终止找到最优训练次数，第二次再完全训练）

- 保存第一次训练时的所有参数，并在所有数据上继续训练。此时无法知晓算法何时停止，但是可以观察验证数据集上的平均Loss，当其低于第一次训练的loss时停止。此方法可以避免第一次的重复计算，但是表现一般。

提前终止的内在机制：一些论文认为，它能将参数搜索空间限制在较小区间，进而加快模型训练。实际上在使用均方差作为损失函数和梯度下降更新参数的简单线性回归模型中，L^2 正则等同于提前终止。

有图，是否需要解释？

7.9 参数捆绑和参数共享

前面的部分讲的都是在固定区域或点对参数加约束或惩罚，例如 L^2 从0点开始寻找最优参数。有时候我们的先验知识需要其他表现形式，或者有时候无法确定精确值，但是知道参数之间的依赖。

前面讲到的参数规范惩罚是一种让参数近似另外一个参数的正则化方法，一种更常见的形式是:**强制让一群参数相等**，此方法称为**参数共享**。其优势在于，可以大幅度减少需要存储和更新的参数。该方法在卷积神经网络中尤其有效。

7.10 稀疏表述

权重衰减是通过在权重参数上增加惩罚项，另一种策略是在网络神经元上施加惩罚。

L^1 正则化带来了稀疏项，是通过使得部分参数为0，而稀疏表述则直接让其中的表述元素直接为0.举例线性回归来说明这种差别:

参数稀疏的表述：

$$ \begin{bmatrix} 18\ 5\ 15\ -9\ -3 \end{bmatrix}=\begin{bmatrix} 4\quad0\quad0\quad-2\quad0\quad0\ 0\quad0\quad-1\quad0\quad3\quad0\ 0\quad5\quad0\quad0\quad0\quad0\ 1\quad0\quad0\quad-1\quad0\quad-4 \end{bmatrix}

\quad\begin{bmatrix} 2\ 3\ -2\ -5\ 1\ 4 \end{bmatrix}\ y\epsilon R^m \quad\quad\quad \quad \quad A\epsilon R^{m\times n}\quad\quad\quad\quad x \epsilon R^m $$

表述稀疏的: $$ \begin{bmatrix} -14\ 1\ 19\ 2\ 23 \end{bmatrix}=\begin{bmatrix} 3\quad-1\quad2\quad-5\quad4\quad1\ 4\quad2\quad-3\quad-1\quad1\quad3\ -1\quad5\quad4\quad2\quad-3\quad-2\ 3\quad1\quad2\quad-3\quad0\quad-3\ -5\quad4\quad-2\quad2\quad-5\quad-1\ \end{bmatrix}

\quad\begin{bmatrix} 2\ 3\ -2\ -5\ 1\ 4 \end{bmatrix}\ y\epsilon R^m \quad\quad\quad \quad \quad B\epsilon R^{m\times n}\quad\quad\quad\quad h \epsilon R^m $$

稀疏表述的 规范惩罚是在损失函数 J 上加一个惩罚项 $\Omega (h)$

$$ \bar J(\theta;X,y) = J(\theta;X,y)+\alpha \Omega (h) \ 其中 \alpha \epsilon [0,\infty) $$

7.11 集成学习方法

主要思想：独立训练多个模型然后所有模型对测试样本投票来减少泛化错误。依据是，不同模型不会在测试样本上犯同样错误。

考虑k个回归模型集合，每个模型在每个(单个)样本上误差为 $\epsilon _i$，模型误差来自多变量正太分布，方差为 $E[\epsilon _i^2]=v$ ，协方差期望为 $E[\epsilon _i,\epsilon _j]=c$ ，则所有集成模型的平均误差是 $\frac{1}{k}\sum_i\epsilon _i$,期望方差为:

$$ E[(\frac{1}{k}\sum_i \epsilon _i)]=\frac{1}{k^2}E[\sum(\epsilon *i^2+\sum*{j\ne i}\epsilon _i\epsilon _j)] \ =\frac{1}{k}v+\frac{k-1}{k}c $$ 有些情况下，误差完全相关，$c=v$ ，此时平均模型没用。 集成学习的期望均方误差与集成规模呈线性递减，也就是集成学习最终至少有其中一个模型的性能。

集成学习方法的简单示例:

图7.5

图中第一行是原始数据，进行一些随机替换和重复（如第二行9替换为8，第三行重复9）分别进行训练。其运行机制就是，每个模型使用的数据集大小相同，但是内容不一，会有部分替换和重复。这样来看，每个单独的模型是相对脆弱的，但是平均化输出，整个模型又是健壮的（上图中只有两个模型的输出都是8时，才会有最大置信度）。

平均模型是极其有效可靠的减少泛化误差的方法，它经常被应用到机器学习竞赛中。BVLC的googlenet使用了6个模型，但是不鼓励在论文中使用，因为可以通过以存储空间换取模型泛化能力。

7.12 dropout

dropout提供了一种计算量不大，但是强大的正则化方法。它是对一个**模型**族进行正则化处理。

集成学习方法需要训练多个模型，如果模型巨大，无法实现。dropout训练的是从基础（原始）网络中移除非输出单元构成的全部子网的集合。下图（图7.6）展

示了此过程。

图7.6

集成学习定义了k个模型，k个从训练数据集中抽样的自集。dropout的目标就是模拟这一过程。训练dropout过程中，我们使用了一个基于*mini-batch*的学习算法，如*SGD*(随机梯度下降)。每载入一个样本到*mini-batch*时，对网络中所有输入和隐藏神经元应用一个二进制掩码，决定每个神经元是否被纳入（二进制掩码为1时），每个神经元取掩码值得过程是独立抽样。掩码取值为1的概率在训练之前就固定的，一般取值是，输入神经元0.8，隐藏神经元0.5。下图是一个前馈网络示例:

图7-7

集成学习的模型都是独立的(每个模型参数和训练数据)，dropout的每个模型的参数都是原网络参数的一个子集，这种共享机制使得dropout 网络有能力表述指数级的特征。**集成学习**每个子模型分别在其训练子集中收敛，而在dropout中，大部分模型并没有得到完全训练（不是每个模型都达到了收敛状态），其魔性太大无法穷尽。dropout网络中，子网的某些部分在迭代中一步步训练，同时参数共享机制使得剩余子网的参数达到一个较好的状态。

使用集成学习模型进。行预测时，需要所有模型对预测结果投票，我称此过程为*inference*。无论是集成学习或者dropout，我们都没有要求模型的精确概率,假设模型是输出一个概率分布，那么集成学习即输出所有概率分布的均值。

$$ \frac{1}{k}\sum_{i=1}^kp^{(i)}(y|x) $$ dropout模型中每个由掩码向量 μ 定义的子模型定义了一个概率分布 $p(y|x,\mu)$，其*inference*是所有掩码概率分布的均值。所有掩码的算术均值如下:

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

由于求和公式包含太多项（指数级），当模型经过一些简化之后很难估计其输出期望（目前为止，深度神经网络的都有不可知的简化）。我们可以通过抽样来模拟*inference*，即平均多个掩码的输出。一般10-20个掩码足以获得较好的表现。

然而，有一种更好的方法，一次前向传播即可获得较好的模拟整个集成学习，即使用**几何平均**替换**算术平均**(所有子模型的)。（论文）

多概率分布的几何均值并不一定是概率分布，为保证多概率的结果依然是概率分布，所有子模型不得使任何事件出现的概率为0，然后使结果呈正态分布。几何均值的非标准(*unnormalizalized*)概率分布如下:

$$ \widetilde p_{ensemble}(y|x) =\sqrt[2^d]{\prod_{\mu}p(y|x,\mu)} \ 其中d为可能被dropout的神经元数 $$

此处使用一个 μ 的正太分布简化表述，其实非正太分布也是可能的。如果要预测输出，需要对模型集合重新标准化:

$$ P(y|x) =\frac{\widetilde P(y|x)}{\sum_{y^{'}}\widetilde P(y^{'}|x)} $$ dropout中可以通过估计一个模型的 $p(y|x)$ 来近似 $p_{ensemble}$ ，该模型包含了所有的神经元，但是每个神经元输出要乘以该神经元被保留的概率。此方法可以获得该神经元的期望输出。我们称此方法为*scale inference rule*，此方法只是一种经验上的技巧，并没有学术论证，但是实际效果很好。

7.12.1 dropout的优点和注意

优点一: 计算量小，训练时每次更新每个样本的时间复杂度为 $O(n)$，其中 n 为要生成的随机二进制数

优点二: 对模型类型和训练过程没有太大限制。几乎对所有使用分布式表述(*distribute representation*)并使用SGD的模型都可以很好。

注意: 尽管dropout的每一步代价不高，但是整体权衡下来还是会比较高，作为一种正则化技术，会削弱模型性能，增大模型可以一定程度上抵消这种削弱，但是切记勿得不偿失。一般标签分类任务中，如果样本较少，比如少于5000时，dropout性能一般。