Computer Science and Application 计算机科学与应用, 2020, 10(6), 1224-1233
Published Online June 2020 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2020.106126

Hans汉斯

Regularization Methods in Deep Learning

Guoning Wu, Huifeng Hu, Mengmeng Yu

College of Science, China University of Petroleum-Beijing, Beijing
Email: wuguoning@163.com, 1303400551@qq.com, 939716685@qq.com

Received: May 27th, 2020; accepted: Jun. 5th, 2020; published: Jun. 12th, 2020

Abstract

The neural network with millions of parameters can easily be overfitting by large dataset. A wide range of regularization methods have been proposed. In this paper, L^1, L^2 and Dropout regularization methods are reviewed. Finally, MNIST handwriting recognition experiments using the above regularization methods are conducted for comparisons.

Keywords

DNN, Overfitting, L^1 Regularization, L^2 Regularization, Dropout, MNIST

深度学习中的正则化方法研究

武国宁，胡汇丰，于萌萌

中国石油大学(北京)，理学院数学系，北京
Email: wuguoning@163.com, 1303400551@qq.com, 939716685@qq.com

收稿日期：2020年5月27日；录用日期：2020年6月5日；发布日期：2020年6月12日

摘　要

带有百万个参数的神经网络在大量训练集的训练下，很容易产生过拟合现象。一些正则化方法被学者提出以期达到对参数的约束求解。本文总结了深度学习中的L^1，L^2和Dropout正则化方法。最后基于上述正则化方法，进行了MNIST手写体识别对比数值试验。

关键词

深度神经网络，过拟合，L^1正则化，L^2正则化，Dropout，MNIST

文章引用: 武国宁, 胡汇丰, 于萌萌. 深度学习中的正则化方法研究[J]. 计算机科学与应用, 2020, 10(6): 1224-1233.
DOI: 10.12677/csa.2020.106126

1. 引言

计算机通过多层次的网络结构，构建简单的“认知”来学习复杂的概念，这种方法被称为 AI 深度学习[1]。它的另外一种解释是一种以人工神经网络为架构，对数据进行表征学习的算法。在当前人工智能的发展中，深度学习起到了中流砥柱的作用。我们一般通过构建神经网络来进行深度学习。现如今已有很多种深度学习的框架模型，例如深度神经网络(DNN) [2]、卷积神经网络(CNN) [3]、置信神经网络(DBN) [4]和递归神经网络(RNN) [5]。它们被应用在计算机视觉、自然语言处理、语音识别与生物信息学等领域并获得极好的效果[6] [7] [8]。

深度神经网络是一种具备至少一个隐藏层的神经网络。我们在构建多层次的网络模型时，经常会遇到过拟合的问题。如何防止过拟合成为深度学习的关键问题之一。减弱过拟合的方法有 L^1、L^2 和 Dropout 等正则化方法[8]。L^1 正则化通过稀疏化权重，而 L^2 正则化通过缩小权重，从而达到减小过拟合现象。Dropout 正则化方法通过“模型平均”和减小神经元之间的共适应性从而达到减弱过拟合现象。

最后本文构建了 784-1000-500-10 的深度前馈全连接神经网络，基于 L^1，L^2 和 Dropout 正则化进行 MNIST 手写体实验。

2. 正则化

一般正则化方法都是通过对目标函数 J 添加一个参数惩罚项 $\Omega(\theta)$，来限制神经网络模型的学习能力[9]。我们将正则化的目标函数记为 $\tilde{J}$：

$$\tilde{J}(\boldsymbol{\theta};\boldsymbol{X},\boldsymbol{y}) = J(\boldsymbol{\theta};\boldsymbol{X},\boldsymbol{y}) + \alpha\Omega(\boldsymbol{\theta}) \tag{1}$$

其中 $\alpha \in [0,\infty)$ 是平衡范数惩罚系数项 Ω 和标准目标函数 $J(\boldsymbol{\theta};\boldsymbol{X},\boldsymbol{y})$ 相对贡献的超参数。当我们的训练算法最小化正则化后的目标函数 $\tilde{J}$ 时，它会减少标准目标函数 $J(\boldsymbol{\theta};\boldsymbol{X},\boldsymbol{y})$ 关于训练数据的误差并同时减少参数 $\boldsymbol{\theta}$ 的规模。选择不同的参数范数 Ω 会偏好不同的解法。

2.1. L^1 正则化

L^1 参数正则化是通过向目标函数函数添加正则项 $\Omega(\theta) = \|\boldsymbol{w}\|_1$，使权重更加靠近坐标轴。我们可以将 L^1 参数正则化目标函数的二次近似分解成关于参数的求和：

$$\hat{J}(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y}) = J(\boldsymbol{w}^*;\boldsymbol{X},\boldsymbol{y}) + \sum_i \left[\frac{1}{2}H_{i,i}\left(w_i - w_i^*\right)^2 + \alpha|w_i|\right] \tag{2}$$

其中 w^* 是最优的目标解，H 是海森矩阵。最小化近似代价函数的解析解是：

$$w_i = sign\left(w_i^*\right)\max\left\{|w_i^*| - \frac{\alpha}{H_{i,i}}, 0\right\} \tag{3}$$

从这个解可以得到两种结果：若 $|w_i^*| \le \frac{\alpha}{H_{i,i}}$，$L^1$ 正则化使得 w_i 趋向 0；若 $|w_i^*| > \frac{\alpha}{H_{i,i}}$，$L^1$ 正则化使得 w_i^* 增加了 $\frac{\alpha}{H_{i,i}}$。

下面借助一张图来解释 L^1 正则化的思想。如图 1 所示，坐标轴右上方的同心椭圆表示原始目标函数 $J(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y})$ 的等值线，中心点 $\boldsymbol{w}^*$ 是没有正则化的原始最优解。图中(虚线)菱形表示 L^1 正则化项的等值线。最小化新的目标函数 $\tilde{J}(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y})$，需要让 $\alpha\|\boldsymbol{w}\|_1$ 和 $J(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y})$ 都尽可能小。在 $\hat{\boldsymbol{w}}$ 点处，这两个竞争目标达到平衡($\hat{\boldsymbol{w}}$ 点为新的目标函数的最优解)。α 很大时，$\hat{\boldsymbol{w}}$ 直接等于 0；α 较小时，$\hat{\boldsymbol{w}}$ 被拉向 0。并且由于 L^1 正则化项图像的特殊性，$\hat{\boldsymbol{w}}$ 很容易就会出现在坐标轴上，即 L^1 正则化会让权重矩阵变得稀疏，使得网络复杂度降低，这也是为什么 L^1 正则化能够防止过拟合。

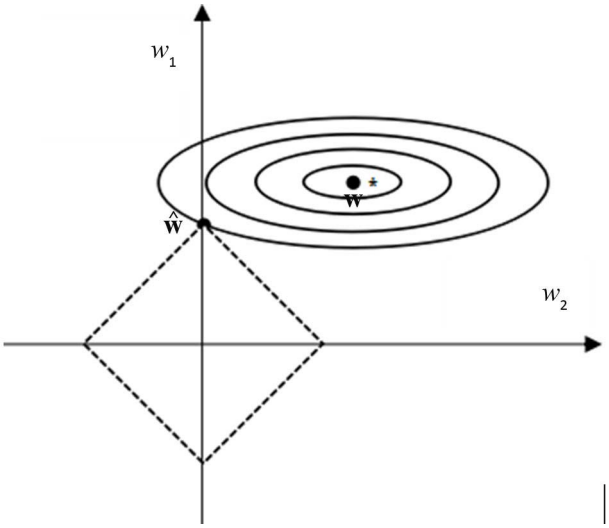

Figure 1. L^1 regularization
图 1. L^1 正则化

2.2. L^2 正则化

L^2 参数正则化是通过向损失函数添加正则项 $\Omega(\boldsymbol{\theta})=\frac{1}{2}\|\boldsymbol{w}\|_2^2$，使权重更加接近原点。可以得到加入 L^2 正则化项的总的目标函数的梯度为：

$$\nabla_{\boldsymbol{w}}\tilde{J}(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y})=\alpha\boldsymbol{w}+H(\boldsymbol{w}-\boldsymbol{w}^*) \tag{4}$$

使用梯度下降法更新权重，过程如下：

$$\boldsymbol{w}\leftarrow(1-\varepsilon\alpha)\boldsymbol{w}-\varepsilon\nabla_{\boldsymbol{w}}J(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y}) \tag{5}$$

每次更新梯度前，都会先对权重向量乘以一个小于 1 的常数因子，这也就是 L^2 正则化称被为权重衰减的原因。记加入正则化项之后的最优解为 $\hat{\boldsymbol{w}}$，有：

$$\nabla_{\boldsymbol{w}}\tilde{J}(\hat{\boldsymbol{w}};\boldsymbol{X},\boldsymbol{y})=\alpha\hat{\boldsymbol{w}}+H(\hat{\boldsymbol{w}}-\boldsymbol{w}^*)=0 \tag{6}$$

其中可以通过特征分解将海森矩阵 H 分解成一个对角阵 Λ 和一组特征向量的标准正交基 Q，即 $H=Q\Lambda Q^{\mathrm{T}}$。解得：

$$\hat{W}=Q(L+\alpha I)^{-1}LQ^{\mathrm{T}}W^* \tag{7}$$

由上面的情况可以看出，海森矩阵的特征值大小决定这权重的缩放程度。而海森矩阵的特征值表示的意义是该点附近特征向量方向上的凹凸性，特征值越大，对应的凸性越强。目标函数下降快的方向对应于训练样本的通用的特征方向，而下降的慢的方向则是会造成过拟合的特征方向。下面借助一张图来

更形象的理解一下 L^2 正则化的效果。

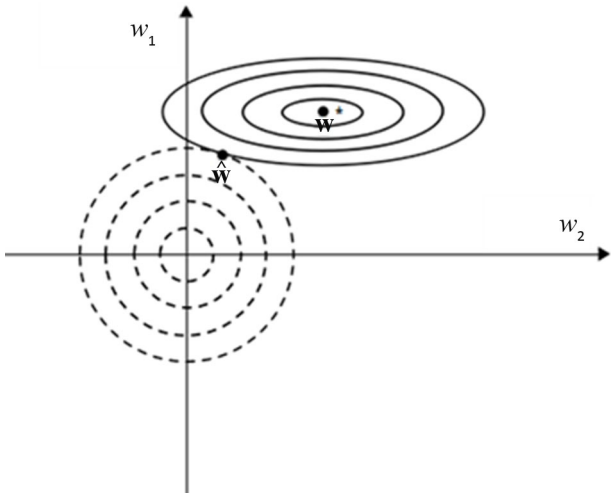

Figure 2. L^2 regularization
图 2. L^2 正则化

如图 2，最小化新的目标函数 $\tilde{J}(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y})$，需要让 $\frac{\alpha}{2}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}$ 和 $J(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y})$ 都足够的小。在 $\hat{\boldsymbol{w}}$ 点处，两者达到平衡。$\hat{\boldsymbol{w}}$ 点为新的目标函数的最优解。当正则化系数 α 越大时，$\hat{\boldsymbol{w}}$ 越接近零点；α 越小时，$\hat{\boldsymbol{w}}$ 越接近 $\boldsymbol{w}^*$。我们看到，目标函数 $J(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y})$ 的海森矩阵的 w_2 方向上的特征值很小，由图 2，我们看到正则化项将 w_2 拉向零。再看，代价函数对于沿着 w_1 所在方向的移动较为敏感,因为对应海森矩阵的特征值比较大，表现为高曲率。因此，权重衰减对 w_1 所在方向影响较小。

通过上面的分析我们发现，保留的相对完整往往是有助于减小目标函数方向上的参数 w_i。而无助于目标函数减小的方向上的参数会在训练中逐渐的衰减掉。这也就是说，在目标函数添加 L^2 正则化项会使模型的参数倾向于比较小的值，针对参数减小了模型拟合各种函数的能力，从而减弱模型的过拟合现象。

2.3. Dropout

下面通过两张图来简单了解一下标准神经网络和应用 Dropout 之后的差异[10]：

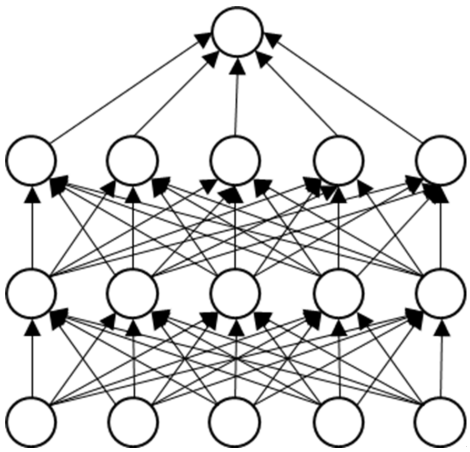

Figure 3. Standard neural network
图 3. 标准神经网络

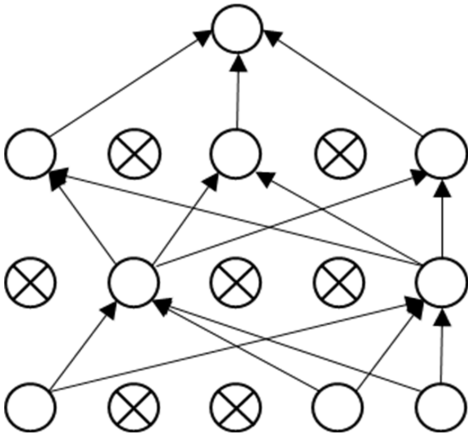

Figure 4. Network with Dropout
图 4. 应用 Dropout 的神经网络

图 3 为一个含有两个隐藏层的标准神经网络。而图 4 则是图 3 的神经网络应用 Dropout 之后的产生的稀疏网络，其中带叉的神经元已经被剔除。

下面介绍 Dropout 的具体工作流程，假设我们要训练图 5 所示神经网络：

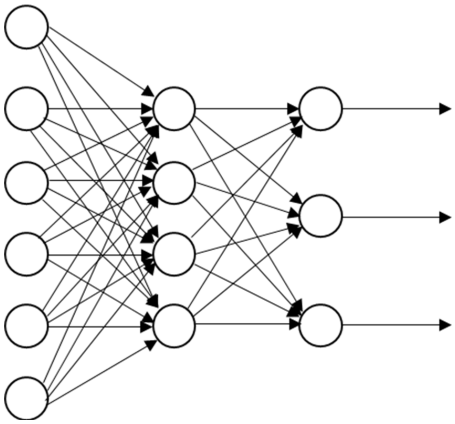

Figure 5. Schematic diagram of neural network
图 5. 神经网络示意图

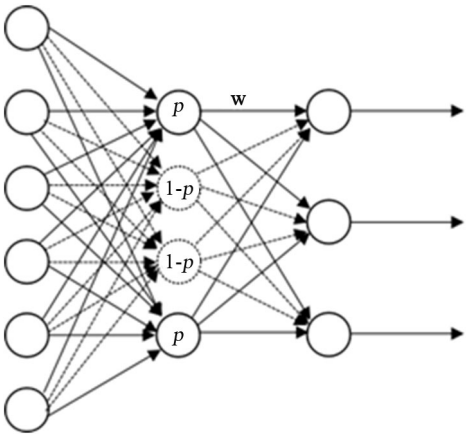

Figure 6. Network with Dropout
图 6. 应用 Dropout 的神经网络

步骤一：遍历网络所有的隐藏层，随机删除掉网络中隐藏层的部分神经元，输入层和输出层保持不变，如图 6 所示，在最简单的情况下，每个单元都以固定的概率 p 保留；

步骤二：接着，输入 x 通过图 3、图 4 所示的神经网络传播，然后反向传播。按照随机梯度下降法更新没有被删除的神经元对应的参数 $\boldsymbol{w}$ 和 b；

步骤三：最后重复如下过程：恢复被删除的神经元，随机删除的神经元的参数不会更新，没有被删除的神经元的参数得到更新。再从隐藏层随机删除一部分神经元，并备份被删除神经元的参数。在划分的小的训练集执行完这个操做之后，按照随机梯度下降法更新没有被删除的神经元对应的参数 $\boldsymbol{w}$ 和 b。被删除的神经元的参数保持原来的结果。而在测试的时候，网络的神经单元一直存在，而权值要乘于 p。这样做是为了保证测试时的输出与训练时的输出期望相同。

下面是这两种情况的示意(图 7)：

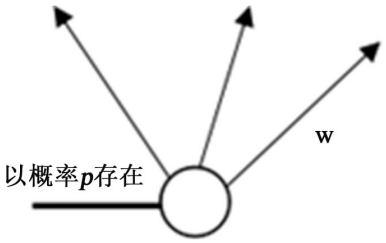

Figure 7. At training time
图 7. 在训练时

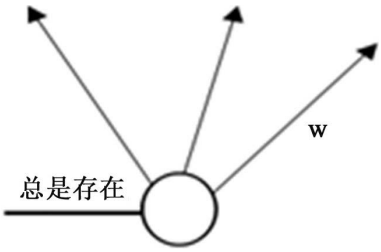

Figure 8. At test time
图 8. 在测试时

将 Dropout 应用到神经网络中相当于从神经网络抽取一个稀疏网络。稀疏网络由所有 Dropout 存活的单元组成(见图 6)。一个有 n 个单元(隐藏层单元)的神经网络，可以看作是有 2^n 个可能的神经网络的集合。这些网络共享权重，因此参数的总数仍然是 $O\left(n^2\right)$ 或更少。对于每次训练集的输出，将对一个新的稀疏网络重新训练。因此，训练一个 Dropout 神经网络可以看做是训练 2^n 个具有大量权重共享的稀疏网络的集合，每个稀疏网络很少接受训练，如果这种情况发生的话(训练单独的稀疏网络)。在测试时，使用单一的神经网络而不 Dropout。该网络的权值是训练权值的“缩减版”，这样就类似于 L^2 正则化的权重缩减。如果一个单元在训练过程中以概率 p 保留，则该单元输出的权值在测试时乘于 p，如图 8 所示。这可以确保对于任何隐藏单元，预期输出(在训练时用于删除单元的分布下)与测试时的实际输出相同。在测试使用时，经过这样的交织， 2^n 个共享权值的网络可以组成一个单独的神经网络。

那么 Dropout 是如何防止过拟合，从而实现正则化的呢？总结为以下两点：

平均的作用：在标准的神经网络模型(没有用任何正则化方法)中，我们利用相同的训练集去训练 m 个不同的神经网络，一般会得到 m 种结果。如果我们采用取均值的方式来决定最终的模型，那么这种综合多个模型取均值的策略可以有效的减小过拟合。因为不同的网络可能会产生不同程度的拟合效果，取

平均值会在一定程度上使过拟合和欠拟合相互抵消。Dropout 在训练时忽略部分神经元，训练不同的稀疏网络，并让这些网络共享权重，这样付出的代价要低。在测试的时候，恢复所有的神经元,相当于很多不同的神经网络取平均。这样在一定程度上使过拟合和欠拟合相互抵消达到整体上减小过拟合。

降低了神经元之间的适应性：一个 100 个人一起完成的计划，要比 100 个人均分为 20 组完成 20 个任务要困难的多。因为前者需要 50 个人的默契配合，后者显然提高了这方面的容错率。Dropout 在训练过程中，两个神经元不一定每次都会在一个 Dropout 网络中出现。这样权值的更新不再依赖于有固定关系的隐藏层神经元的共同作用，一定程度上避免了一些特征只有在特定特征下才有效果的情况，迫使网络学习更加鲁棒(指系统的健壮性)的特征，从而降低了神经元之间的适应性，达到减小过拟合的效果。

3. 正则化在 MNIST 手写体识别中的应用

MNIST 手写体是 NIST 提供的用来进行神经网络实验的数据集，包含 60,000 张手写数字的二进制数据组成的训练集和 10,000 张相同规格的二进制图像组成的测试集。在本文实验中，我们训练和测试用的数据，用的是 MNIST 数据集的 784 维的二进制数据，输入的像素是灰度级的，值为 0 表示白色，值为 1.0 表示黑色，中间值表示逐渐暗淡的灰色。因此我们构建的全连接前馈神经网络的输入层的神经元的个数为 784 个。为了实现更好的拟合效果，我们把第一个隐藏层的神经元的个数设置为 1000，第二个隐藏层的神经元的个数设置为 500。最后是输出层，因为手写体上包含 0~9 十个数字，因此我们的输出层的神经元的个数设置为 10。网络的示意图如下(图 9)：

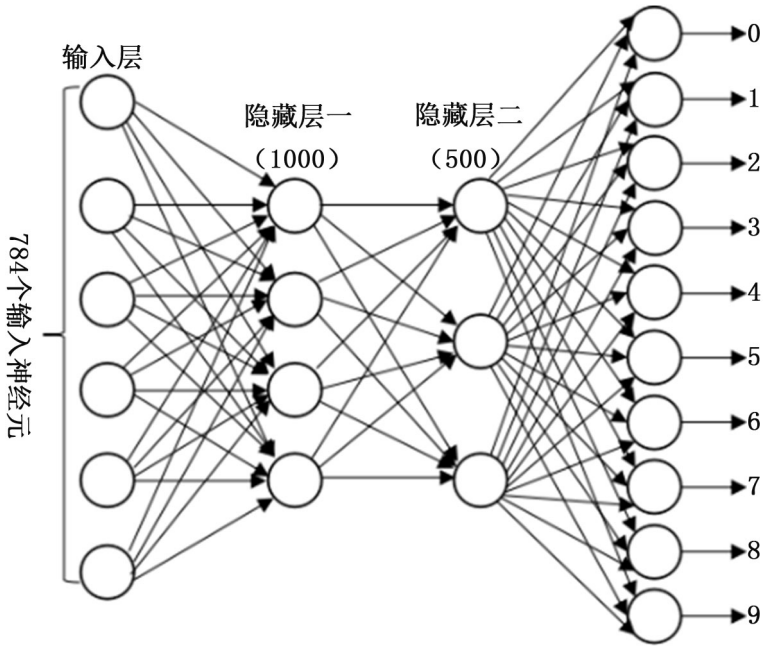

Figure 9. 784-1000-500-10 net
图 9. 784-1000-500-10 网络

隐藏层激活函数我们选用双曲正切函数：$f(z)=\frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$。输出层的激活函数我们选用 Softmax 函数，它的主要作用是是将多个神经元的输出，映射到 0~1 的区间上，按照概率的高低进行分类，各个输出神经元的概率之和为 1。形式如下：$a_j^L=\frac{e^{z_j^L}}{\sum_k e^{z_k^L}}$。目标函数们采用交叉熵函数，在犯错的情况下能够学习

的更快，其具体形式如下：

$$C(\boldsymbol{\theta};\boldsymbol{X},\boldsymbol{y})=-\frac{1}{n}\sum_{x}\left[y(x)\ln\hat{y}(x)+(1-y(x))\ln(1-\hat{y}(x))\right]$$

在本文实验中，我们采用的学习速率 $\varepsilon=0.5$，Dropout 的存活概率为 0.5，L^1,L^2 正则化的惩罚系数 $\alpha=0.0005$。本文实验的平台是 Ubuntu，运用的 python 版本是 3.6.7，TensorFlow 的版本是 1.13.1。下面是运行结果：

我们分别取上述四种情况的测试准确率进行比对，如下(图 10)：

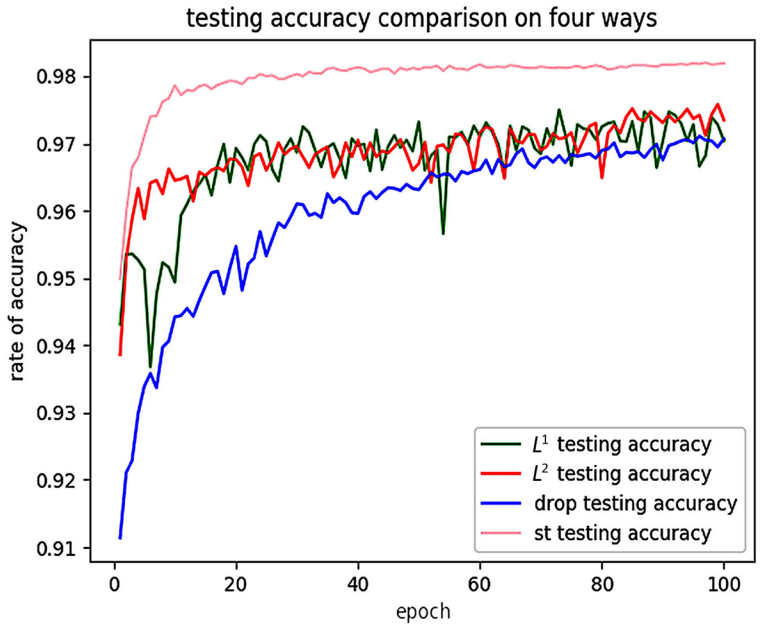

Figure 10. Comparison of test accuracy in four cases
图 10. 四种情况下的测试准确率对比

对应的表格如下(表 1)：

Table 1. Four cases test accuracy rate partial training cycle results
表 1. 四种情况测试准确率部分训练周期结果

测试准确率(%)	Epoch 20	Epoch 40	Epoch 60	Epoch 80	Epoch 100
标准网络	97.92	98.13	98.17	98.14	98.19
L^1 正则化	96.93	96.97	97.11	97.25	97.04
L^2 正则化	96.76	97.05	97.16	96.49	97.35
Dropout	95.47	95.96	96.61	96.75	97.07

由图 10，我们看到 L^1,L^2 正则化测试准确率上下震荡比较厉害，而 Dropout 只有小幅度的震荡，并且由表 4.5，在 80~100 个 epoch，三者的测试准确率都在 97%左右，由此看出 Dropout 作用后拟合效果(相对于 L^1,L^2)比较好。我们分别取上述四种情况的训练准确率进行比对，如图 11。

对应的表格如表 2。

下面我们对比部分周期四种情况下的训练误差和测试误差的差值。

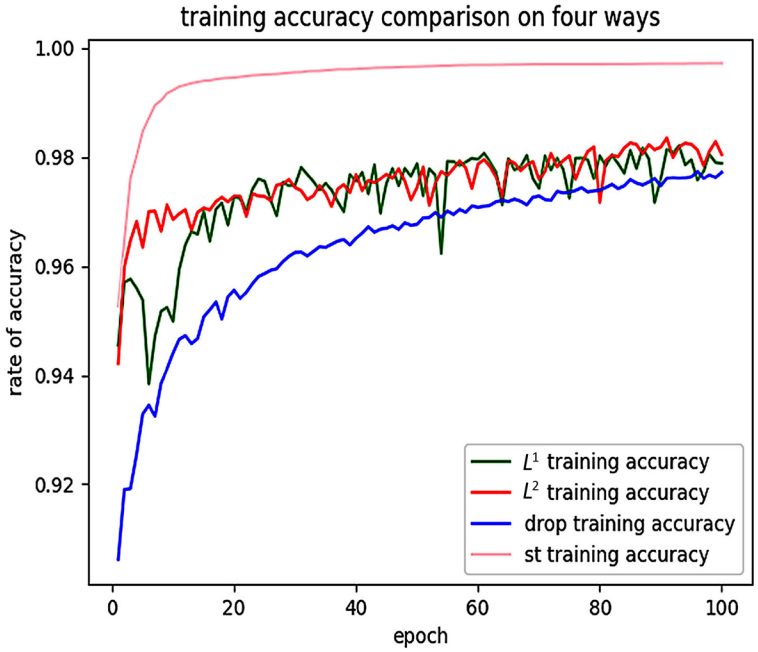

Figure 11. Comparison of train accuracy in four cases
图 11. 四种情况下的训练准确率对比

Table 2. Four cases train accuracy rate partial training cycle results
表 2. 四种情况训练准确率部分训练周期结果

测试准确率(%)	Epoch 20	Epoch 40	Epoch 60	Epoch 80	Epoch 100
标准网络	99.45	99.61	99.69	99.70	99.71
L^1 正则化	97.26	97.55	97.95	98.02	97.88
L^2 正则化	97.27	97.68	97.87	97.16	98.04
Dropout	95.56	96.51	97.07	97.46	97.72

Table 3. Four cases of accuracy rate difference part of the training cycle results
表 3. 四种情况准确率差值部分训练周期结果

准确率差值(%)	Epoch 20	Epoch 40	Epoch 60	Epoch 80	Epoch 100
标准网络	1.53	1.48	1.52	1.56	1.52
L^1 正则化	0.33	0.58	0.84	0.77	0.84
L^2 正则化	0.51	0.63	0.71	0.67	0.69
Dropout	0.09	0.55	0.46	0.71	0.65

由表 3，我们发现 Dropout 最终将准确率的差值缩小到 0.65%，相比 L^1,L^2 正则化的 0.69%，0.84%效果较好。综上对于 784-1000-500-10 的深度神经网络，Dropout 的正则化效果相对较好。

4. 结论

1) L^1 参数正则化会趋于生成少量的权重，而其他权重都变为 0。如图 1，由于 L^1 正则项函数的特性，参数的最优值很大概率会出现在坐标轴上。这样就会导致 $\boldsymbol{w}$ 的某一维变为 0，使得权重矩阵变得稀疏，并且网络复杂度降低，从而一定程度上减小过拟合。

2) L^2 参数正则化则会保留更多(相对于 L^1 参数正则化)的权重，但是这些权重都会在不同程度上逼近于 0。如图 2，L^2 参数正则化的最优的参数只有很小概率会出现在坐标轴上，因此 $\boldsymbol{w}$ 的每一维基本都不会是 0，而是在正则化下逼近于 0。它通过衰减参数减小了模型拟合各种函数的能力，从而减弱模型的过拟合现象。

3) Dropout 则是在训练时忽略部分神经元，训练不同的稀疏网络，并让这些网络共享权重。在测试的时候，恢复所有的神经元，即所有的稀疏网络交织在一起，相应的权重乘以概率 p，相当于很多不同的神经网络取平均。这样在一定程度上使过拟合和欠拟合相互抵消达到整体上减小过拟合。并且在训练不同稀疏网络时，两个神经元不一定每次都会在一个 Dropout 网络中出现。这样权重的更新不再依赖于有“逻辑关系”的隐藏层的神经元的共同作用，一定程度上避免了一些特征只有在特定特征下才有效果的情况，迫使网络学习更加鲁棒(指系统的健壮性)的特征，从而降低了神经元之间的适应性，达到减小过拟合的效果。

4) 在 MNIST 手写体实验中，我们构建 784-1000-500-10 的深度神经网络，并进行正则化处理。结果表明 Dropout 的正则化效果会更好，最终的准确率稳定在 97%，训练误差与测试误差的差值为 0.65%。

参考文献

[1] LeCun, Y., Bengio, Y. and Hinton, G. (2015) Deep Learning. *Nature*, **521**, 436-444. https://doi.org/10.1038/nature14539

[2] Schmidhuber, J. (2015) Deep Learning in Neural Network: An Overview. *Neural Networks*, **61**, 85-117. https://doi.org/10.1016/j.neunet.2014.09.003

[3] Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2012) ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, 1097-1105.

[4] Hinton, G.E., Osindero, S. and Teh, Y.W. (2006) A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, **18**, 1527-1554. https://doi.org/10.1162/neco.2006.18.7.1527

[5] Pearlmutter (1989) Learning State Space Trajectories in Recurrent Neural Networks. *International* 1989 *Joint Conference on Neural Networks*, Washington DC, **2**, 365-372. https://doi.org/10.1109/ijcnn.1989.118724

[6] Bengio, Y. (2009) Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning*, **2**, 1-127. https://doi.org/10.1561/2200000006

[7] Bengio, Y., Courville, A. and Vincent, P. (2013) Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **35**, 1798-1828. https://doi.org/10.1109/tpami.2013.50

[8] Najafabadi, M.M., Villanustre, F., Khoshgoftaar, T.M., *et al.* (2015) Deep Learning Applications and Challenges in Big Data Analytics. *Journal of Big Data*, **2**, 1. https://doi.org/10.1186/s40537-014-0007-7

[9] LeCun, Y., Bottou, L., Bengio, Y., *et al.* (1998) Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, **86**, 2278- 2324. https://doi.org/10.1109/5.726791

[10] Srivastava, N., Hinton, G., Krizhevsky, A., *et al.* (2014) Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, **15**, 1929-1958.