第七章 深度学习的正则化

Regularization for Deep Learning

让算法对于训练集中的数据和未训练过的数据都有很好的性能是机器学习的一个核心问题（泛化能力）。有很多训练策略是为解决这个问题而设计的，有些策略甚至会为了降低测试集的错误率而牺牲训练集正确率。这些策略统称为正则化（规则化、正规化）。深度学习从业者会面对很多种类的正则化方法。实际上，研发好的正则化方法已经成为深度学习的主要研究方向之一。

第 5 章介绍了泛化、欠拟合、过拟合、偏置（bias）、方差（variance）和正则化的基本概念。如果你对这些术语还不熟悉，请在阅读本章内容之前阅读第 5 章。

本章，我们更详细的阐述正则化，主要专注于深度模型或可以构成深度模型的基础模块的正则化策略。

本章的部分内容主要讲述机器学习的经典概念，如果你对这些经典概念很熟悉可以略过这些章节。但是，本章的大部分内容都是在经典概念基础上面向神经网络的拓展。

在 5.2.2 节，我们曾经定义将正则化定义为："任何以减少泛化错误率而不是训练错误率为目的算法改进"。目前有很多正则化策略。有些事对机器学习模型本身做出限制，例如限定参数取值范围；有些在目标函数中添加额外的附加项，这可以看作是对参数取值范围的柔性限制（soft constraint）。如果选择得当，这些限制和惩罚附加项可以在测试集上面有效的提高性能。有时候，这些附加项是用来表达特定的先验知识；另外一些时候，这些附加项是用来保证模型具有较低的复杂度；还有些时候这些附加项对于将非确定性问题转为确定性问题是必不可少的（are necessary to make an underdetermined problem determined）。集成（Ensemble）方法是另外一种形式的正则化，通过多个假设的组合来解释训练数据。

在深度学习中，大多数正则化策略是基于正则化估计算法（Estimators）。估计算法的正则化是通过增大偏置（bias）和减小方差（variance）的折中（trading）来完成。有效的正则化方法是通过有效的折中偏置和方差来完成的——即在明显减小方差的同时不对增加偏置量进行明显的增加。在第 5 章中我们讨论泛化和过拟合的时候，我们关注了三种情况，即 1）训练后的模型不能表达真实的数据生成过程，对应欠拟合；2）模型与真实的数据生成过程匹配；3）模型除了包含真实的数据生成过程之外，引入了其他可能产生真实数据的生成过程，对应过拟合情况（是方差而不是偏置量决定了估计算法的错误率）。正则化方法的目的就是讲模型从第三种情况变为第二种情况。

在实际应用中，过度复杂的模型不一定能够表达真实目标函数或真实数据的生成过程，有时候甚至前两者的近似都无法表达。我们见不到真实数据生成过程也就无法我们使用的模型是否可以支持该过程。但是对于绝大多数深度学习应用来讲，真实的数据生成过程都无法被我们使用的模型所表达。深度学习算法就是典型的被用来处理极端复杂领域的问题的，例

如图像、音频序列、文本等。这些领域的数据真实生成过程本质上是对整个宇宙过程的仿真和重现。某种程度上，我们总是尝试将一个桩子（真实数据生成过程）放入一个圆形的孔中（我们使用的模型）。

上面的描述的意思就是：在深度学习意义上，控制模型复杂度不再是简单的控制网络参数的数量，而是找到一个经过合理的正则化的复杂的模型。深度学习中要在这个准则下找到最佳模型，最佳的评价标准是具有最小泛化误差。

下面我们回顾一些创建复杂正则化模型的策略。

7.1 参数化范数惩罚

正则化技术出现在深度学习之前很多年。线性模型如线性回归、logistic 回归都有简单有效的正则化策略。

很多正则化方法都是基于限制模型的表达能力（Capacity）的。比如神经网络、线性回归、逻辑回归中在目标函数中加入参数的惩罚项。正则化的目标函数一般表达为：

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

上式中的 A（0 到正无穷）是超参数（不需要训练的参数）。A 的大小决定正则项起的作用大还是损失函数起的作用大。

训练算法会同时减小损失函数和网络的 Size。

在开始研究正则化之前，我们要先阐明一点：一般我们只利用范数正则化权值 W，而不对偏置量 b 进行正则化。理由：1）由于偏置量仅仅控制一个变量，不像权值连接两个变量。因此不对偏置量正则化不会导致方差过大；2）对偏置量进行正则化会导致严重的欠拟合，会导致模型无法调整到正确位置。因此网络参数 Theta 包括 W 和 b 两部分（前者被正则化、后者不被正则化）。

在神经网络中，有的时候我们希望对不同的层采用不同的惩罚项超参数 A，但这样非常耗时，因此在实际应用中我们一般只使用一个超参数来减小搜索空间。

7.1.1 二范数参数正则化 L^2 Parameter Regularization

目标函数如下：

$$\tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \frac{\alpha}{2}\boldsymbol{w}^\top\boldsymbol{w} + J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}), \tag{7.2}$$

参数的梯度：

$$\nabla_{\boldsymbol{w}}\tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \alpha\boldsymbol{w} + \nabla_{\boldsymbol{w}}J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}). \tag{7.3}$$

梯度的单步更新方式：

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \epsilon \left(\alpha \boldsymbol{w} + \nabla_{\boldsymbol{w}} J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) \right). \tag{7.4}$$

换一种写法为：

$$\boldsymbol{w} \leftarrow (1 - \epsilon\alpha)\boldsymbol{w} - \epsilon \nabla_{\boldsymbol{w}} J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}). \tag{7.5}$$

上式可以看到，对于单步更新来讲，在正常的权值更新基础上，又进一步加入了一个固定的乘法性质的权值衰减项。

对于整个训练过程二范数乘法项是会带来哪些影响呢？

我们假定 W*是对没有二范数乘法项的损失函数 J 计算得到的最佳权值，此时对无正则化项的损失函数进行泰勒展开得到其线性化逼近：

$$\hat{J}(\boldsymbol{\theta}) = J(\boldsymbol{w}^*) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^*)^\top \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*), \tag{7.6}$$

注意，上式中没有一阶导数项，这是因为 W*就是最佳权值，是让 J 最小的 W，一阶导数为 0。这里 H 是 J 的 Hessian 矩阵（在 W*处取值）。根据上式，J 的取最小值时的 W 可通过对上式求导并令其=0 获得，即下式=0：

$$\nabla_{\boldsymbol{w}} \hat{J}(\boldsymbol{w}) = \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*) \tag{7.7}$$

此时根据（7.3）加入权值衰减项，可得

$$\alpha\tilde{\boldsymbol{w}} + \boldsymbol{H}(\tilde{\boldsymbol{w}} - \boldsymbol{w}^*) = 0 \tag{7.8}$$

$$(\boldsymbol{H} + \alpha\boldsymbol{I})\tilde{\boldsymbol{w}} = \boldsymbol{H}\boldsymbol{w}^* \tag{7.9}$$

$$\tilde{\boldsymbol{w}} = (\boldsymbol{H} + \alpha\boldsymbol{I})^{-1}\boldsymbol{H}\boldsymbol{w}^*. \tag{7.10}$$

当 A 趋于 0 的时候，权值与无惩罚项的情况看一致。而当 A 增大的时候会怎么样呢？我们将 H 进行分解得到 $\boldsymbol{H} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top$，其中 Q 为归一化特征向量构成的正交阵，A 为特征值构成的对角阵。的进一步得：

$$\tilde{\boldsymbol{w}} = (\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top + \alpha\boldsymbol{I})^{-1}\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top\boldsymbol{w}^* \tag{7.11}$$

$$= \left[\boldsymbol{Q}(\boldsymbol{\Lambda} + \alpha\boldsymbol{I})\boldsymbol{Q}^\top\right]^{-1}\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top\boldsymbol{w}^* \tag{7.12}$$

$$= \boldsymbol{Q}(\boldsymbol{\Lambda} + \alpha\boldsymbol{I})^{-1}\boldsymbol{\Lambda}\boldsymbol{Q}^\top\boldsymbol{w}^*. \tag{7.13}$$

$(\boldsymbol{\Lambda} + \alpha\boldsymbol{I})^{-1}\boldsymbol{\Lambda}$的作用相当于在 H 的坐标轴上，对每个特征值进行压缩，压缩的比例为 $\frac{\lambda_i}{\lambda_i + \alpha}$，（压缩比例从对角阵的逆为元素取倒数，再与原矩阵对应相乘即可获得）。从而看到从整体上 2 范数正则化项的加入压缩了权值。

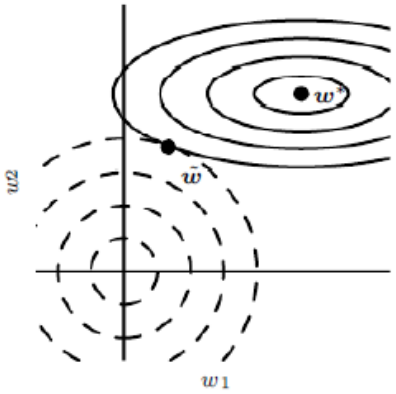


Figure 7.1: An illustration of the effect of L^2 (or weight decay) regularization on the value of the optimal $\boldsymbol{w}$. The solid ellipses represent contours of equal value of the unregularized objective. The dotted circles represent contours of equal value of the L^2 regularizer. At the point $\tilde{\boldsymbol{w}}$, these competing objectives reach an equilibrium. In the first dimension, the eigenvalue of the Hessian of J is small. The objective function does not increase much when moving horizontally away from $\boldsymbol{w}^*$. Because the objective function does not express a strong preference along this direction, the regularizer has a strong effect on this axis. The regularizer pulls w_1 close to zero. In the second dimension, the objective function is very sensitive to movements away from $\boldsymbol{w}^*$. The corresponding eigenvalue is large, indicating high curvature. As a result, weight decay affects the position of w_2 relatively little.

由上图可知，再加入正则项之后，在原来移动 w 就会导致很大梯度的地方，正则化项的影响不大，因此此时不应该让 w 过分衰减，此时减小训练误差主要依靠 w 的这个维度。

对于式（7.2） $\tilde{J}(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y}) = \frac{\alpha}{2}\boldsymbol{w}^\top\boldsymbol{w} + J(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y})$ ，可以看作是如下不等式约束优化问题的拉格朗日表达式。

$$\min_{\mathbf{w}} J(\mathbf{W};\mathbf{X},y), \quad s.t. \quad \mathbf{W}^T\mathbf{W} < C$$

由不等式约束的最优化问题(具体定理可见相关参考资料)可知，上面问题可转化为(7.2)求最小值的问题（即拉格朗日 Lagrangian 表达式）。

不等式约束优化

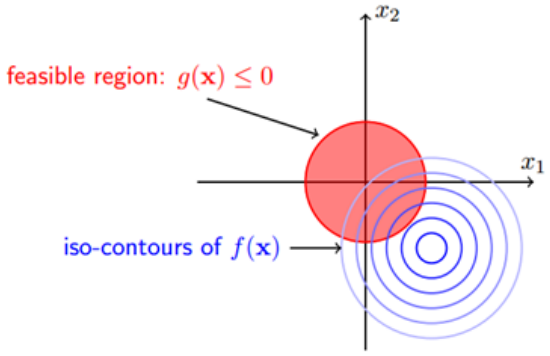
当约束加上不等式之后，情况变得更加复杂，首先来看一个简单的情况，给定如下不等式约束问题：

$$\min_x \ f(x)$$
$$s.t. \ \ g(x) \leq 0$$

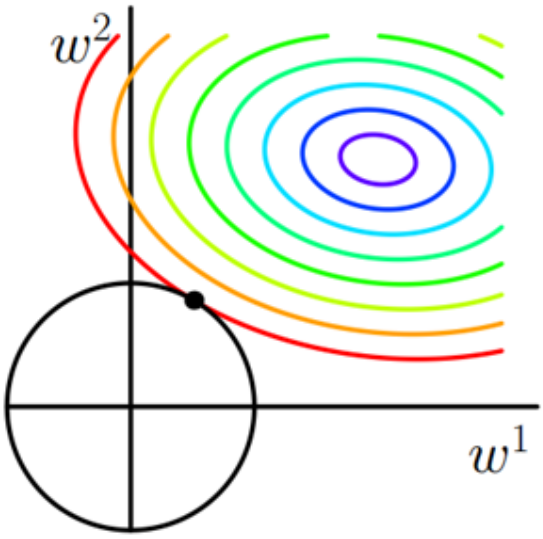
对应的 Lagrangian 与图形分别如下所示：

$$L(x, \lambda) = f(x) + \lambda g(x)$$

这时的可行解必须落在约束区域 $g(x)$ 之内，下图给出了目标函数的等高线与约束：



（7.2）的最小化问题可表达为下图：



(b) ℓ_2-ball meets quadratic function. ℓ_2-ball has no corner. It is very unlikely that the meet-point is on any of axes."

到目前为止，我们讨论的权值衰减的作用都是针对抽象的通用凸优化问题。权值衰减对真正的机器学习问题的影响是什么样的呢？我们可以通过回顾线性回归问题来解读，因为线性回归问题的优化问题是凸的，完全可以应用我们上面的分析。在线性回归中，损失函数平

方误差的和：

$$(\boldsymbol{X}\boldsymbol{w}-\boldsymbol{y})^\top(\boldsymbol{X}\boldsymbol{w}-\boldsymbol{y}). \tag{7.14}$$

When we add L^2 regularization, the objective function changes to

$$(\boldsymbol{X}\boldsymbol{w}-\boldsymbol{y})^\top(\boldsymbol{X}\boldsymbol{w}-\boldsymbol{y})+\frac{1}{2}\alpha\boldsymbol{w}^\top\boldsymbol{w}. \tag{7.15}$$

This changes the normal equations for the solution from

$$\boldsymbol{w}=(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y} \tag{7.16}$$

to

$$\boldsymbol{w}=(\boldsymbol{X}^\top\boldsymbol{X}+\alpha\boldsymbol{I})^{-1}\boldsymbol{X}^\top\boldsymbol{y}. \tag{7.17}$$

$\boldsymbol{X}$	The $m \times n$ matrix with input example $\boldsymbol{x}^{(i)}$ in row $\boldsymbol{X}_{i,:}$

$$\left(\mathbf{X}^T\mathbf{X}+\alpha\mathbf{I}\right)^{-1}=\left(\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T+\alpha\mathbf{I}\right)^{-1}=\left(\mathbf{Q}\left(\mathbf{\Lambda}+\alpha\mathbf{I}\right)\mathbf{Q}^T\right)^{-1}=\mathbf{Q}\left(\mathbf{\Lambda}+\alpha\mathbf{I}\right)^{-1}\mathbf{Q}^T$$

$$\left(\mathbf{X}^T\mathbf{X}\right)^{-1}=\left(\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T\right)^{-1}=\mathbf{Q}\mathbf{\Lambda}^{-1}\mathbf{Q}^T$$

从PCA角度解读上式：

矩阵 $\mathbf{X}^T\mathbf{X}$ 的对角线元素对应的是每一维特征的方差。回想主成分分析 $PCA:\mathbf{Z}=\mathbf{X}\mathbf{Q},\ \mathbf{Z}^T=\mathbf{Q}^T\mathbf{X}^T$，Q是归一化正交阵，$Q^TX^T$相当于PCA之后X投影到新坐标系Q下的系数。上面变为：

$$\left(\mathbf{X}^T\mathbf{X}+\alpha\mathbf{I}\right)^{-1}\mathbf{X}^T=\left(\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T+\alpha\mathbf{I}\right)^{-1}\mathbf{X}^T=\left(\mathbf{Q}\left(\mathbf{\Lambda}+\alpha\mathbf{I}\right)\mathbf{Q}^T\right)^{-1}\mathbf{X}^T=\mathbf{Q}\left(\mathbf{\Lambda}+\alpha\mathbf{I}\right)^{-1}\mathbf{Z}^T$$

$$\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T=\left(\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T\right)^{-1}\mathbf{X}^T=\mathbf{Q}\mathbf{\Lambda}^{-1}\mathbf{Q}^T\mathbf{X}^T=\mathbf{Q}\mathbf{\Lambda}^{-1}\mathbf{Z}^T$$

由于 $\mathbf{Q}\mathbf{Z}^T=\mathbf{X}^T$，

$\mathbf{Q}\mathbf{\Lambda}^{-1}\mathbf{Z}^T$ 相当于把Z中的每个元素都收缩 $1/\lambda_i$ 之后再变换回X。

$\mathbf{Q}\left(\mathbf{\Lambda}+\alpha\mathbf{I}\right)^{-1}\mathbf{Z}^T$ 相当于对Z中的每个元素做了更大的收缩之后再变换回X，收缩量为 $1/\left(\lambda_i+\alpha\right)$。

从奇异值分解角度解读上式

$\mathbf{X}=\mathbf{U}\mathbf{D}\mathbf{V}^T, \mathbf{X}^T=\mathbf{V}\mathbf{D}^T\mathbf{U}^T$

$\mathbf{X}^T\mathbf{X}=\mathbf{V}\mathbf{\Lambda}\mathbf{V}^T, \mathbf{\Lambda}=\mathbf{D}^T\mathbf{D}$:各维特征的协方差（对角阵，协方差都为0）

$\mathbf{X}^T\mathbf{X}+\alpha\mathbf{I}=\mathbf{V}\mathbf{\Lambda}\mathbf{V}^T+\alpha\mathbf{I}=\mathbf{V}\left(\mathbf{\Lambda}+\alpha\mathbf{I}\right)\mathbf{V}^T$

$\left(\mathbf{X}^T\mathbf{X}+\alpha\mathbf{I}\right)^{-1}=\mathbf{V}\left(\mathbf{\Lambda}+\alpha\mathbf{I}\right)^{-1}\mathbf{V}^T$

$\left(\mathbf{X}^T\mathbf{X}+\alpha\mathbf{I}\right)^{-1}\mathbf{X}^T=\mathbf{V}\left(\mathbf{\Lambda}+\alpha\mathbf{I}\right)^{-1}\mathbf{V}^T\mathbf{V}\mathbf{D}^T\mathbf{U}^T=\mathbf{V}\left(\mathbf{\Lambda}+\alpha\mathbf{I}\right)^{-1}\mathbf{D}^T\mathbf{U}^T$

$\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T=\mathbf{V}\left(\mathbf{\Lambda}\right)^{-1}\mathbf{D}^T\mathbf{U}^T$

比较上面最后两行可以发现，原来的计算中每一维特征的方差都在分母中，而加入正则化项后，增大了分母。对于原来方差大的特征维度，α增量占比不大。而对于原来方差较小的特征，α占比很大，压缩大。

结论：

由于α的值是固定的，当 λ_i 越大的时候，α占 $\left(\lambda_i+\alpha\right)$ 的比例越小，α的收缩作用越弱；当 λ_i 越小的时候，α占 $\left(\lambda_i+\alpha\right)$ 的比例越大，收缩作用越强。因此，二范数正则化导致学习算法可以“perceive”（察觉、发现）具有更大方差的特征，对方差更大的特征进行更弱的权值收缩，对方差小的特征进行更强的特征收缩。方差大的特征蕴含的信息量更丰富，对模型的训练和收敛起到关键作用，正则化对这些特征不进行大的收缩。对于那些蕴含信息量较小的特征，正则化对其进行更强的压缩，使模型复杂度得到保证，提高模型的泛化能力。

7.1.2 一范数参数正则化 L^1 Parameter Regularization

目标函数：

$$\tilde{J}(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y})=\alpha||\boldsymbol{w}||_1+J(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y}), \tag{7.19}$$

梯度：

$$\nabla_{\boldsymbol{w}}\tilde{J}(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y})=\alpha\text{sign}(\boldsymbol{w})+\nabla_{\boldsymbol{w}}J(\boldsymbol{X},\boldsymbol{y};\boldsymbol{w}) \tag{7.20}$$

Sign 表示符号。

参考（7.6）式，目标函数在最优 W*上面二阶泰勒展开逼近为：

$$\hat{J}(\boldsymbol{\theta})=J(\boldsymbol{w}^*)+\frac{1}{2}(\boldsymbol{w}-\boldsymbol{w}^*)^\top\boldsymbol{H}(\boldsymbol{w}-\boldsymbol{w}^*)$$

当输入数据进行过 PCA 之后，H 是对角阵（并且元素 $H_{i,i}>0$），此时上式变为

$$J(\boldsymbol{w}^*;\boldsymbol{X},\boldsymbol{y})+\sum_i\frac{1}{2}H_{i,i}(\boldsymbol{w}_i-\boldsymbol{w}_i^*)^2$$

根据（7.19）加入 1 范数正则化项后的目标函数变为：

$$\alpha||\boldsymbol{w}||_1+J(\boldsymbol{w}^*;\boldsymbol{X},\boldsymbol{y})+\sum_i\frac{1}{2}H_{i,i}(\boldsymbol{w}_i-\boldsymbol{w}_i^*)^2$$

根据1范数的定义可得，

$$\hat{J}(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y}) = J(\boldsymbol{w}^*;\boldsymbol{X},\boldsymbol{y}) + \sum_i \left[\frac{1}{2}H_{i,i}(\boldsymbol{w}_i - \boldsymbol{w}_i^*)^2 + \alpha|w_i|\right]. \tag{7.22}$$

上式有如下解析解：

$$w_i = \text{sign}(w_i^*)\max\left\{|w_i^*| - \frac{\alpha}{H_{i,i}}, 0\right\}. \tag{7.23}$$

注意α和H_{ii}都>0。

考虑wi*>0的时候，

当wi*<α/H_{ii}的时候，正则化后的wi的解为0，此时说明经验损失J的作用被完全淹没了，1范数正则化项起主导作用，权值在方向i上被压缩到0；

当wi*>α/H_{ii}的时候，正则化后的wi的解为|wi*|-α/H_i，此时说明经验损失J的作用被部分削弱了，1范数正则化项起一定作用，但没有完全在方向i上将权值压缩到0。

考虑wi*<0的时候，

当|wi*|<α/H_{ii}的时候（此时wi*接近0），正则化后的wi的解为0，此时说明经验损失J的作用被完全淹没了，1范数正则化项起主导作用，权值在方向i上被压缩到0；

当|wi*|>α/H_{ii}的时候（此时wi*是一个离0较远的负值），正则化后的wi的解为-(|wi*|-α/H_i)，即wi*+α/H_i，将wi*向0拉近了。此时说明经验损失J的作用被部分削弱了，1范数正则化项起一定作用，虽然没有完全在方向i上将权值压缩到0。但从负半轴方向将其一定程度上向0压缩了。

由上面分析可以看到，1范数正则化会产生稀疏性，很多w的值被压缩为0。尤其是当α足够大的时候，稀疏性会很明显。

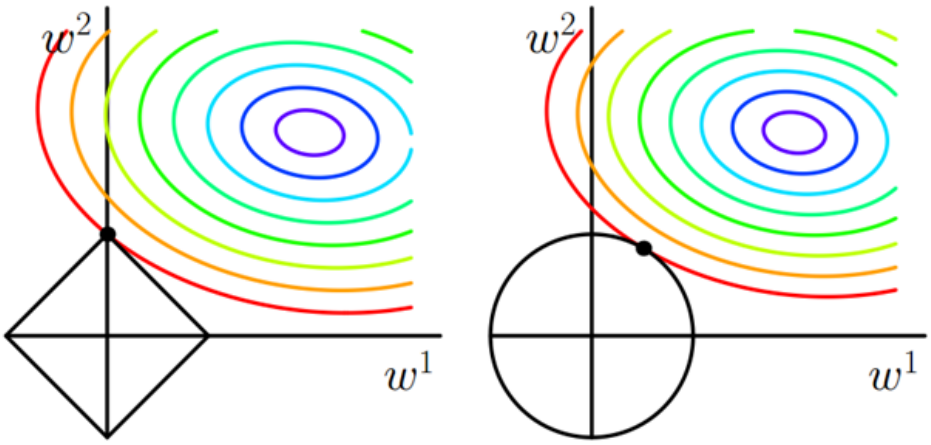
1范数的稀疏性可以用来进行特征选择，找到那些权值不为0的特征。典型做法就是LASSO（Least Absolute Shrink and Selection Operator）模型。

$$Lasso: \quad \min_w \frac{1}{n}\| y - Xw \|^2, \quad s.t. \| w \|_1 \le C$$

LASSO用1范数约束下的线性模型作为目标函数，可以对特征进行筛选，wi=0的特征维度被放弃。与之对应的线性模型是Ridge。

$$Ridge: \quad \min_w \frac{1}{n}\| y - Xw \|^2, \quad s.t. \| w \|_2 \le C$$

两者的区别如下图，可以看到w限制在C中之后，与J的交点极为最优w，而1范数更容易使交点落于坐标轴，从而导致稀疏。



(a) ℓ_1-ball meets quadratic function. ℓ_1-ball has corners. It's very likely that the meet-point is at one of the corners.

(b) ℓ_2-ball meets quadratic function. ℓ_2-ball has no corner. It is very unlikely that the meet-point is on any of axes.

从Bayesian或MAP的角度讲，2范数相当于假设了W的先验概率是0均值高斯分布。而1范数则是假定了W的先验概率是各向相等的拉普拉斯分布（Isotropic Laplace Distribution）（公式3.26）。

$$\text{Laplace}(x;\mu,\gamma) = \frac{1}{2\gamma}\exp\left(-\frac{|x-\mu|}{\gamma}\right) \tag{3.26}$$

优化1范数正则化项，相当于优化Laplace分布W先验概率的对数项，具体如下：

$$\log p(\boldsymbol{w}) = \sum_i \log \text{Laplace}(w_i; 0, \frac{1}{\alpha}) = -\alpha||\boldsymbol{w}||_1 + n\log\alpha - n\log 2.$$

7.2 将范数惩罚转为约束优化问题（4.4 节讲述约束优化，中文可看下载的 pdf 博客）

前面已经涉及了部分最优化约束表达的范数惩罚问题，本节详细展开。把等式和不等式约束都加入到拉格朗日目标函数之后，λ、α 系数统称 KKT 乘子。

$$L(\boldsymbol{x},\boldsymbol{\lambda},\boldsymbol{\alpha}) = f(\boldsymbol{x}) + \sum_i \lambda_i g^{(i)}(\boldsymbol{x}) + \sum_j \alpha_j h^{(j)}(\boldsymbol{x}). \tag{4.14}$$

目标函数极值的必要条件（目标函数导数=0），不等式约束函数与不等式约束函数的 kkt 乘子的积（松弛互补条件）=0，等式约束，不等式约束，不等式约束的 kkt 乘子>=0。共 5 个条件成为 kkt 条件。

$$\nabla_x L(x,\alpha,\beta) = 0 \quad (1)$$
$$\beta_j g_j(x) = 0,\ j = 1,2,\ldots,n \quad (2)$$
$$h_i(x) = 0,\ i = 1,2,\ldots,m \quad (3)$$
$$g_j(x) \le 0,\ j = 1,2,\ldots,n \quad (4)$$
$$\beta_j \ge 0,\ j = 1,2,\ldots,n \quad (5)$$

对拉格朗日目标函数在 kkt 条件下进行求解，即可解决该不等式约束优化问题。

不等式约束正则化与参数化正则化的联系：

使用不等式约束优化问题求解范数惩罚问题的好处在于：这种方法可以显式的指定范数项被约束的程度，即 $\min_{\mathbf{w}} J(\mathbf{W};\mathbf{X},y), \quad s.t. \quad \mathbf{W}^T\mathbf{W} < C$。在参数化范数惩罚方法中，是通过 $\tilde{J}(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y}) = \frac{\alpha}{2}\boldsymbol{w}^\top\boldsymbol{w} + J(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y})$ 中的 α 来控制权值压缩的程度。但是 α 和 C 之间的关联是很难量化描述的，如果我们事先已经确定优化中需要压缩的绝对量 C，那么直接使用不等式优化方法比使用参数化方法更加直观。

7.3 约束不足问题的正则化 Regularization and Under-Constrained Problems

在某些情况下，要合理的定义机器学习问题必须依赖正则化。很多线性机器学习问题，比如线性回归和 PCA 都依赖于矩阵 X^TX 的逆。很多情况下当该矩阵奇异，导致其逆矩阵无法获得。这些情况包括：在某些特征维度上面的方差（对于 0 均值数据而言，X^TX 为协方差阵）为 0，一般由于观测到的样本数量（X 的行数）少于特征维数（X 列数）导致的某些维度上面的方差无法被有效观测（样本太少体现不出方差）。举例如下：

X =

2	1	5	16	32
3	6	4	8	0
10	7	2	8	1

Xt =

2	3	10
1	6	7
5	4	2
16	8	8
32	0	1

Xt*X =

113	90	42	136	74
90	86	43	120	39
42	43	45	128	162
136	120	128	384	520
74	39	162	520	1025

```
[v,d]=eig(Xt*X)

v =

   -0.2992    0.0775   -0.7559    0.5657    0.1141        74
    0.6987    0.4277    0.1821    0.5372    0.0847        39
   -0.6259    0.5912    0.4551    0.1644    0.1570       162
   -0.0989   -0.6454    0.3748    0.4379    0.4913       520
    0.1441    0.2121   -0.2187   -0.4155    0.8449      1025

d = 1.0e+03 *

   -0.0000         0         0         0         0
         0    0.0000         0         0         0
         0         0    0.0200         0         0
         0         0         0    0.2616         0
         0         0         0         0    1.3714
```

```
[U S D]=svd(Xt*X)

U =

-0.1141   -0.5657    0.7559    0.3081    0.0253        74
-0.0847   -0.5372   -0.1821   -0.6155    0.5406        39
-0.1570   -0.1644   -0.4551    0.7176    0.4757       162
-0.4913   -0.4379   -0.3748   -0.0127   -0.6528       520
-0.8449    0.4155    0.2187   -0.1058    0.2336      1025

S = 1.0e+03 *

1.3714         0         0         0         0
     0    0.2616         0         0         0
     0         0    0.0200         0         0
     0         0         0    0.0000         0
     0         0         0         0    0.0000

D =

 -0.1141   -0.5657    0.7559    0.0723    0.3005
 -0.0847   -0.5372   -0.1821   -0.7060   -0.4156
 -0.1570   -0.1644   -0.4551   -0.2275    0.8303
 -0.4913   -0.4379   -0.3748    0.6161   -0.2162
 -0.8449    0.4155    0.2187   -0.2550   -0.0275
```

通过上面的例子可以看到，X 的特征维度大于样本数量的时候，X^TX 的特征值少于 5，不满秩，不可逆。

这种情况下，很多正则化方法都像前面两节一样，改用一定具有逆矩阵的 $X^TX+\alpha I$。

在相应的矩阵可逆的时候，上面自然段中谈到的线性问题是有 Closed 的解的。对于很多欠定问题是没有 Closed 解的。比如逻辑回归问题中，如果一组 W 可以实现 0 错误率的分类，那么 2W、3W、kW 也可以实现完好的分了，并且分类似然度更大。此时如果对 W 不做正则化，则用于求解极大似然的梯度下降法永远不会收敛，并且 W 的绝对值会爆炸增长。在实际的梯度下降法的编程实现中，权值的绝对值总会逐渐增大到会导致数值溢出的程度，这时梯度下降法的算法行为就要依赖于编程者如何处理那些溢出的没有意义的数值了。

大部分的正则化方法都能够保证应用于欠定问题的迭代求解方法的收敛。例如，权值收缩会方法会在似然度的梯度等于权值收缩系数的时候停止增大权值的绝对值。

利用正则化方法解决欠定问题超出了机器学习的范畴，很多基础线性代数问题的求解都依赖于此方法，比如欠定方程组的伪逆求解。回想伪逆的定义：

$$\boldsymbol{X}^{+} = \lim_{\alpha \searrow 0} (\boldsymbol{X}^{\top}\boldsymbol{X} + \alpha \boldsymbol{I})^{-1}\boldsymbol{X}^{\top}. \tag{7.29}$$

我们可以发现线性回归问题中也使用了伪逆来对权值进行收缩，只是在是否对 α 取极限上有微小区别。我们可以将伪逆解读为用正则化方法稳定求解欠定问题的一种方式。

7.4 数据集的增强 Dataset Augmentation

获得机器学习模型泛化能力的最好方法就是在训练中加入更多的有效数据。然而，在实际中我们拥有的数据数量总是有限的。我们可以通过创建伪数据来解决这个问题。

在分类问题中创建伪数据是最简单的。因为分类器需要将一个复杂的高维输入 x 标记为一个类别标号 y。这意味着分类的主要工作是对特征取值在一定范围的同类 x 给出同样的标记 y。我们只需要对 x 的特征值进行各类随机变换并将其标号设为 y，即可获得可用的伪样本（x，y）。

这样的方法对于很多其他应用来说都不像分类问题中这么简单。比如，对于概率密度估计问题就很难创建伪样本，除非这个概率密度已知（不可能，我们要估计的就是它）。

数据集增强对于分类问题中的目标识别问题尤其有效。图像是高维的并且有很多变换参数，而很多变换参数都是非常容易仿真的。比如在每个方向上将训练图像平移几个像素通常可以大大提高泛化能力。这个技巧即使是对已经考虑了平移泛化能力的卷积神经网络（卷积和 pooling）而言仍然是比不可少的。图像的旋转和缩放也被证明是非常有效的。

在应用上述技巧的时候，一定要注意不要改变正确的类别。比如：OCR 任务要区分 b 和 d，6 和 9，所以水平镜像和 180 度旋转就不能再上述任务的数据增强中应用。

还有一些变换是很难去仿真的，比如，非二维平面的旋转就不是输入图像的简单操作能完成的。

数据集增强对语音识别任务也是同样有效的（Jaitly and Hinton, 2013）。

向网络中加入噪声也可看作是一种数据增强的方法。很多分类和回归任务要求数据收到微小随机噪声干扰的情况下仍然可以正常工作。而神经网络对噪声是不够鲁棒的（2010）。

可以在输入中加入随机噪声来增强鲁棒性。向输入数据添加噪声是降噪自动编码器（2008）类的非监督学习方法的组成部分之一。噪声也可以注入到网络隐层，此时可以看作在多个抽象的层次对数据进行增强。Pool et al.（2014）的研究成果证明了在噪声量级精确调整的情况下，这种多层次加噪的方法是非常有效的。Dropout 是另外一种非常有效的正则化策略，我们将在 7.12 节讨论这种方法。这种方法也可以被看作是通过乘以噪声创建新的输入的过程。

当对比机器学习 benchmark 结果的时候，一定要将数据集增强的效果考虑进去。通常，手工设计的数据集增强方法可以大幅降低机器学习的泛化错误率。在对比机器学习算法的时候，一定要使用同样的数据增强策略。

7.5 噪声鲁棒性 Noise Robustness

7.4 节已经提及了输入样本加噪的样本集增强策略。对有些模型，对输入样本加入方差很小的加性噪声等同于对权值加入范数惩罚约束（Bishop，1995）。通常，样本集加噪要比只是收缩权值要更有效，特别是对隐层加噪时候。隐层加噪是一个非常重要的话题，我们将用单独的一个章节讨论该问题，7.12 中讨论的 Dropout 算法是隐层加噪的最主要实现方式。

将噪声加入到权值是加噪方法的另外一个应用。这项技术主要是用于 RNN（1996，2011）。这可以看作是权值上的 Bayesian 推理的随机实现。Bayesian treatment of learning 将模型的权值看作是不确定的，并且可以用概率分布来表达。权值加噪可反应这种不确定性。

权值加噪在某些假设下也可以看作一种更加传统的正则化，可以增加学习的稳定性。考虑回归问题，我们要在最小二乘损失准则下降输入 x 映射到输出 y。

$$J = \mathbb{E}_{p(x,y)}\left[(\hat{y}(\boldsymbol{x}) - y)^2\right]. \tag{7.30}$$

通常可以向权值加入 0 均值随机扰动 $\epsilon \boldsymbol{W} \sim \mathcal{N}(\epsilon; \boldsymbol{0}, \eta \boldsymbol{I})$，加入随机噪声等同于加入一个正则化项 $\eta \mathbb{E}_{p(\boldsymbol{x},y)}\left[\|\nabla_{\boldsymbol{W}} \hat{y}(\boldsymbol{x})\|^2\right]$，（此处没有看懂）。正则化项的作用为：使权值的搜索优先考虑那些随机扰动对输出具有更小的影响的权值取值区域。也可看作：搜索目标函数的极小值，而且周围是平缓区域的极小值，此时 w 的小的变化不会对目标函数带来大的改变。

7.5.1 在样本标号中加入噪声 Injecting Noise at the Output Targets

大多数数据集的标号 y 是有错误的，这对最大化后验概率 p（y|x）是非常不利的。通过对标号的噪声进行概率建模可以从一定程度上解决这个问题。例如，我们可以假设样本标号都是以概率 1-ϵ 正确的。这个概率是可以解析的加入到目标函数中的。比如 Label Smoothing 方法将硬标号 0，1 转为软标号 ϵ/(k-1)和 1-ϵ。标准的交叉熵损失可以用在这类软目标标号上面。采用 Softmax 和硬标号的极大似然估计有可能不会收敛，因为 Softmax 不能对输入预测为精确的 0 或 1，所以分类器会一直尝试将权值大再变大从而做出更加接近 0 和 1 的预

测。权值收缩时应对这种情况的有效方法。Label Smoothing 可以防止权值不停的迭代以满足上述的 hard probabilities。而且这种防止权值不收敛的方式不是以鼓励错分为代价的（discouraging correct classification），是否可理解为？如果要使用硬标号让迭代收敛必须要让一部分样本被错分。

7.6 半监督学习 Semi-Supervised Learning（翻译的不好，直接参考 ppt）

在半监督学习框架下，由分布 P（x）产生的无标号样本和由分布 P（x，y）产生的有标号样本一起用来估计 P(y|x)。

在深度学习场景中，半监督学习通常指学习一个自变量为 x 的表达 h=f（x），让所有来自相同类别的样本具有类似的表达。非监督学习指明了如何在表达空间对数据分组，那些输入空间中聚集紧密的样本应该映射为近似的表达。表达之后的新的特征空间中，线性分类器会具有更好的泛化能力（2002,2003）。这个方法的一个非常经典的应用就是用 PCA 对数据进行分类前的预处理。

监督和非监督方法可以组合到一个模型中，组合后的模型中生成模型 P（x）（或 P（x，y））与判别模型 P(y|x)共享参数。可以通过监督学习标准-logP(y|x)和非监督标准-logP(x)（或-logP（x，y））的折中来确定半监督学习标准。此时生成模型标准表现为监督学习问题的解的特定形式的先验置信度（prior belief），即通过参数化的方式描述了 P（x）和 P（y|x）的关系。通过控制生成模型标准在总标准中的比重，可以获得比单纯使用生成模型或单独使用判别模型都更好的训练标准。

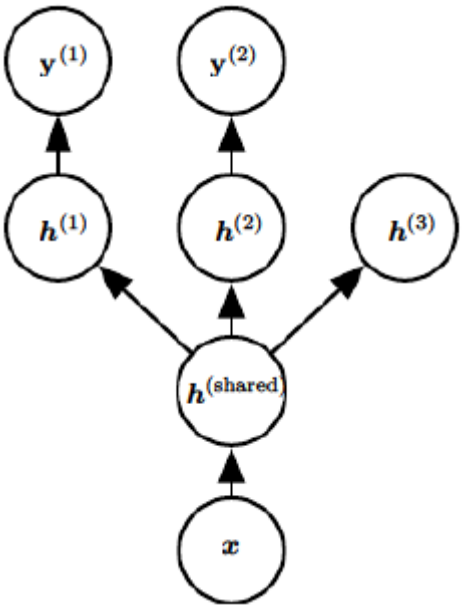
Salakhutdinov and Hinton (2008)描述了一种学习高斯过程的协方差核的方法，通过利用无标号样本对 P(x)进行建模，极大的提高了 P（y|x ）的建模准确性。Using deep belief nets to learn covariance kernels for Gaussian processes。

更多的关于半监督学习的信息可以参考 Chappelle et al.（2006）。

一句话：目标函数中包含监督学习损失和非监督学习损失两部分，细节很多，参见上面的经典书 2006 Oliver Chappelle《Semi-Supervised Learning》 MIT Press。

7.7 多任务学习 Multi-Task Learning

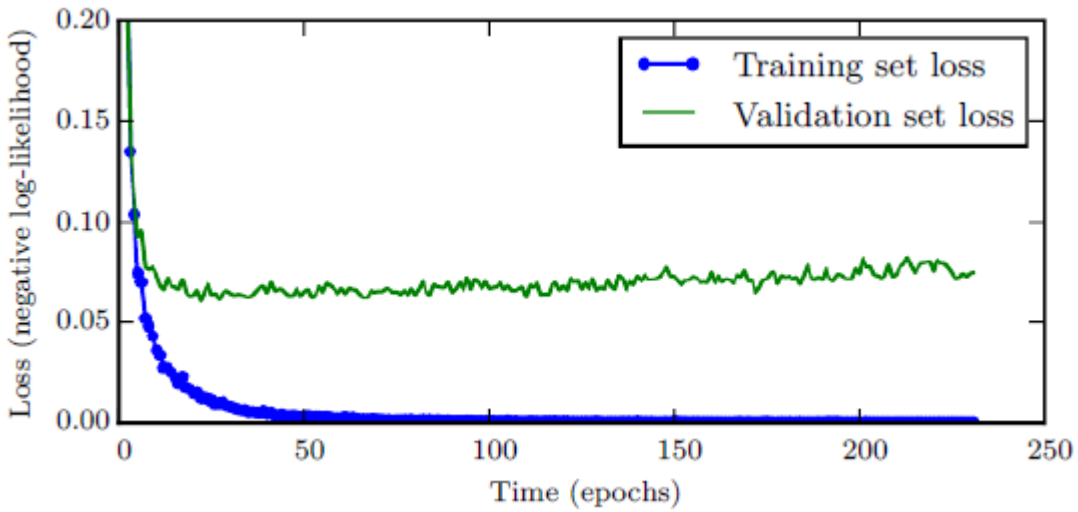
多任务学习（1993）可以通过共用面向多个任务的训练样本提高模型泛化能力。



如果多个任务的模型有一个 h_share 部分是通用的，因此可以把多个任务的数据放到一起来训练，这样 h_share 部分的模型就学习了更多的训练样本，也就提高了泛化能力。当然这种方法应用的前提是各个模型具有通用部分，比如迁移学习？h_share 学习猫和狗的特征，分类的时候一个任务是分类虎和狼、一个任务是分类猫和狗？

7.8 提前停止 Early Stopping

在训练大型模型的时候，训练误差会随着迭代的进行而稳定的下降，但是验证集的误差往往在到达一定程度后会开始上升。如下图所示。



因此在训练中，我们不是记录最后的网络权值，而是在每个时刻都记录到当前为止具有最小验证集训练误差的权值。在网络迭代指定次数之后，我们找到那个记录的具有最小验证集误差的权值作为最终的训练输出模型。

这种想法非常简单也易于实现，是目前深度学习最为广泛采用的正则化方法。

其实这种方法可以理解为：把迭代次数看作一个超参数，而 early stopping 是一种有效的超参数选择。

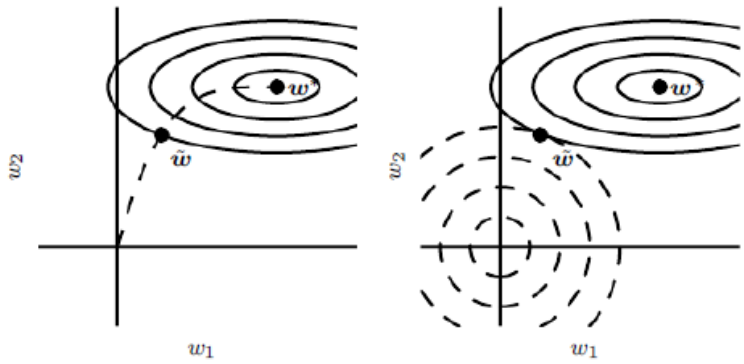
通常，超参数搜索是非常耗时的，但迭代次数或迭代时间是非常特殊的，它的搜索可以在训练的同时完成。多出来的计算代价仅仅是在训练每组参数后计算验证集误差。可以并行计算或减小验证集来加速。

与权值衰减不同，early stopping 不影响网络训练的基础过程。在权值衰减中如果衰减设置的过大，会导致网络陷入一个毫无意义的权值都很小的目标函数局部极小，从而导致训练失败。

Early Stopping 与其他正则化技术一起使用也是没有问题的。其他正则化方法修改了目标函数之后，使用 Early Stopping 依然是很好的选择。

在实际使用中可以先进行一次 Early Stopping，记录训练迭代的次数。之后重新初始化权值为随机数，再重新训练迭代第一次 Early Stopping 记录的最佳次数。

为什么 early Stopping 可以实现正则化



看上左图，如果没有 Early Stopping，权值会一直收敛到 w 处，现在提前停止了，所以在 w~处收敛。这与带有 2 范数惩罚项的正则化方法有异曲同工之妙。

7.9 参数绑定和参数共享 Parameter Tying and Parameter Sharing

有些时候我们虽然不知道权值的值是什么，但是我们知道权值之间的依赖关系。

一个典型依赖关系就是特定参数间的值比较接近。考虑如下情况：两个模型执行高度近似的任务（输入输出的分布都近似），这种情况下两个模型的参数应该具有较高的相似性。这时可以把权值的相似度加入到目标函数中，例如 2 范数距离 $||W_a - W_b||^2$。Lasserre et al. (2006) 用这种方法训练了一个监督学习模型 A，在训练中，作者使用了另外一个非监督方法训练得到的模型 B（表达的是输入观测数据的分布）。网络的架构体现了 A 和 B 两个模型中的参数的对应相似关系。

此外，更为广泛应用的做法是权值共享，典型的例子就是卷积神经网络。

7.10 稀疏表达 Sparse Representations

权值衰减直接在网络权值中加入惩罚项。另外一种策略是在神经元激活输出中加入惩罚

项，促使其神经元激活输出为稀疏的。这间接的向模型权值中加入了一个复杂的惩罚项。

我们已经在 7.1.2 节中讨论了 1 范数正则化如何导致权值的稀疏。另一方面，表达稀疏描述了一种元素大部分是 0 或接近 0 的表达。在线性回归场景中可以看到这种分布的简单情况。

$$\begin{bmatrix} 18 \\ 5 \\ 15 \\ -9 \\ -3 \end{bmatrix} = \begin{bmatrix} 4 & 0 & 0 & -2 & 0 & 0 \\ 0 & 0 & -1 & 0 & 3 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & -4 \\ 1 & 0 & 0 & 0 & -5 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ -2 \\ -5 \\ 1 \\ 4 \end{bmatrix} \tag{7.46}$$

$\boldsymbol{y} \in \mathbb{R}^m$ $\quad \boldsymbol{A} \in \mathbb{R}^{m \times n}$ $\quad \boldsymbol{x} \in \mathbb{R}^n$

$$\begin{bmatrix} -14 \\ 1 \\ 19 \\ 2 \\ 23 \end{bmatrix} = \begin{bmatrix} 3 & -1 & 2 & -5 & 4 & 1 \\ 4 & 2 & -3 & -1 & 1 & 3 \\ -1 & 5 & 4 & 2 & -3 & -2 \\ 3 & 1 & 2 & -3 & 0 & -3 \\ -5 & 4 & -2 & 2 & -5 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \\ 0 \\ 0 \\ -3 \\ 0 \end{bmatrix} \tag{7.47}$$

$\boldsymbol{y} \in \mathbb{R}^m$ $\quad \boldsymbol{B} \in \mathbb{R}^{m \times n}$ $\quad \boldsymbol{h} \in \mathbb{R}^n$

第一个表达式，A 是一个参数稀疏的线性回归模型。

第二个表达式，h 是一个数据 x 的稀疏表达，即 h 是 x 的函数，h 以一种稀疏的方式表达 x 的信息。（**相当于隐层激活稀疏**）

表达方面的正则化可以用与参数正则化类似的方式完成，即向损失函数中添加表达范数惩罚项。

$$\tilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \alpha \Omega(\boldsymbol{h}) \tag{7.48}$$

上式中 α>=0。

与 1 范数参数惩罚项类似，1 范数表达惩罚项引入表达的稀疏性：$\Omega(\boldsymbol{h}) = ||\boldsymbol{h}||_1 = \sum_i |h_i|$。当然 1 范数惩罚项只是能够得到稀疏表达的一种选择。其他可以得到稀疏表达的方法还有：假定表达的先验分布为 Student-t 从而导出的惩罚项（2011）；在表达元素限定于单位间隔情况下极为有效的 KL 散度惩罚项（2008）；Lee（2008）和 Goodfellow（2009）都提出了一种方法稀疏表达正则化方法，方法中让多个样本激活的平均值（向量）尽量接近元素都是 0.01 的向量。

其他可以获得稀疏表达的方法都是对神经元激活的取值加入强力约束。 例如，orthogonal matching pursuit (Pati et al, 1993)通过下面的约束优化问题来求解 x 的稀疏表达 h。

$$\arg\min_{\boldsymbol{h},\|\boldsymbol{h}\|_0<k} \|\boldsymbol{x}-\boldsymbol{W}\boldsymbol{h}\|^2, \tag{7.49}$$

上式中|| h ||$_0$ 是 h 的非 0 元素的个数。上述问题在限定 W 是正交阵后可以有效的求解。这中方法通常称为 OMP-k，k 表示允许的非零特征的个数。

本质上任何具有隐层神经元的模型都可以对其进行稀疏表达操作。纵观整本书，我们将在很多场景中见到许多稀疏正则化的例子。

7.11 Bagging 和其他集成方法 Bagging and Other Ensemble Methods

Bagging（Booststrap aggregating）是通过合并多个模型来减小泛化误差的方法。核心思想是训练多个不同的模型，在通过不同的模型投票来确定输出。Bagging 这是机器学习中的通用策略“model averaging”的典型例子，采用“model averaging”策略的技术统称为“Ensemble Methods”。

Model Averaging 策略之所以能够成功，原因在于不同的模型不太可能在测试集上都犯同样的错误。

考虑一组 k 个回归模型，假定每个模型对样本的预测误差为 ϵi（误差为 0 均值正态分布，误差的方差为 v，不同模型间误差的协方差为 c）。那么对所有的预测取均值后，预测误差变为 $\frac{1}{k}\sum_i \epsilon_i$。Ensemble 后的平方预测误差为：

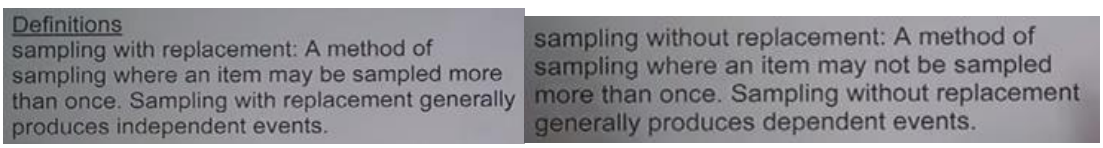
$$\mathbb{E}\left[\left(\frac{1}{k}\sum_i \epsilon_i\right)^2\right] = \frac{1}{k^2}\mathbb{E}\left[\sum_i\left(\epsilon_i^2+\sum_{j\neq i}\epsilon_i\epsilon_j\right)\right] \tag{7.50}$$

$$= \frac{1}{k}v+\frac{k-1}{k}c. \tag{7.51}$$

当各个模型误差完全相关 v=c 时，上述均方差变为 v，此时 ensemble 没有任何作用。当各个模型误差完全不相关 c=0 的时候，均方差变为 v/k。这说明均方差与模型个数线性相关。就是说集成后的模型最起码不会比单个模型变差。

不同的集成方法从不同的角度进行模型集成。例如：参与集成的模型可以是来自不同的算法和目标函数的完全不同类的模型。而 Bagging 是指用集成同样的算法、同样的目标函数训练多次得到的多个同类模型。

Bagging 算法有一个特别的地方，就是要构建 k 个不同的训练样本集。每个训练样本集都与原始数据集大小相同，采用从原始数据集中采用置回式随机抽样来获取（一个样本被采集后，仍然在候选集中，即有可能被采到多次。这可以保证每次抽样的独立性）。



这样每个样本集合都会遗漏部分原始数据集中的数据，并且有些数据被抽样（复制）了

多次。在训练样本集与原始样本集样本数量相同的情况下，通常每个训练样本集会包含原始样本集中的 2/3 样本。这样不同的训练样本会训练处不同的模型。

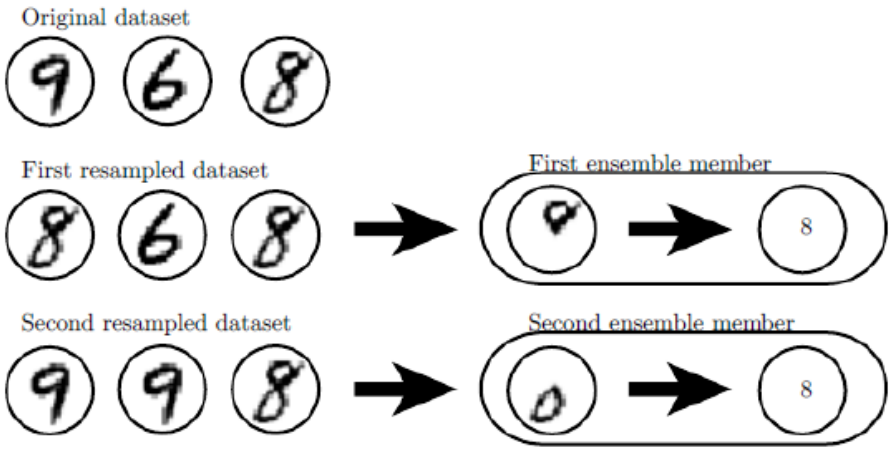


图 7.5 第一个模型会把上边的半个圈识别为 8，第二个模型会把下边的半个圈识别为 8。每个模型单独使用都是不好的，但是把这些模型做 Averaging，效果就很好了。

神经网络的训练通常会收敛到很多不同差异较大的解，通过对这些解取平均可以极大提高网络性能，即使这些解是用相同的样本集训练出来的（Also works）。有很多元素可以支持从相同的数据集训练出不同模型，这些要素包括：不同的随机初始化、不同的随机 minibatch 的选择、不同的超参数、不适定（non-deterministic）问题在求解实现中得到的不同输出。并且这些从相同数据集得到的不同模型的泛化错误具有相当程度的独立性，可以支持 Ensemble。

需要指出：虽然 Model Averaging 策略可以极大提高网络性能，但是通常在对各类方法进行标准对比的时候都是使用单一模型，而不用 average。这时可以体现所有方法本身的性能。但在实际应中一定更要应用这一策略，来确保提高泛化能力。

很多机器学习竞赛中的优胜者都是使用了 Model Average 策略，并且集成的模型种类和个数很多。最近的一个著名例子是 Netflix Grand Prize （Koren,2009）。

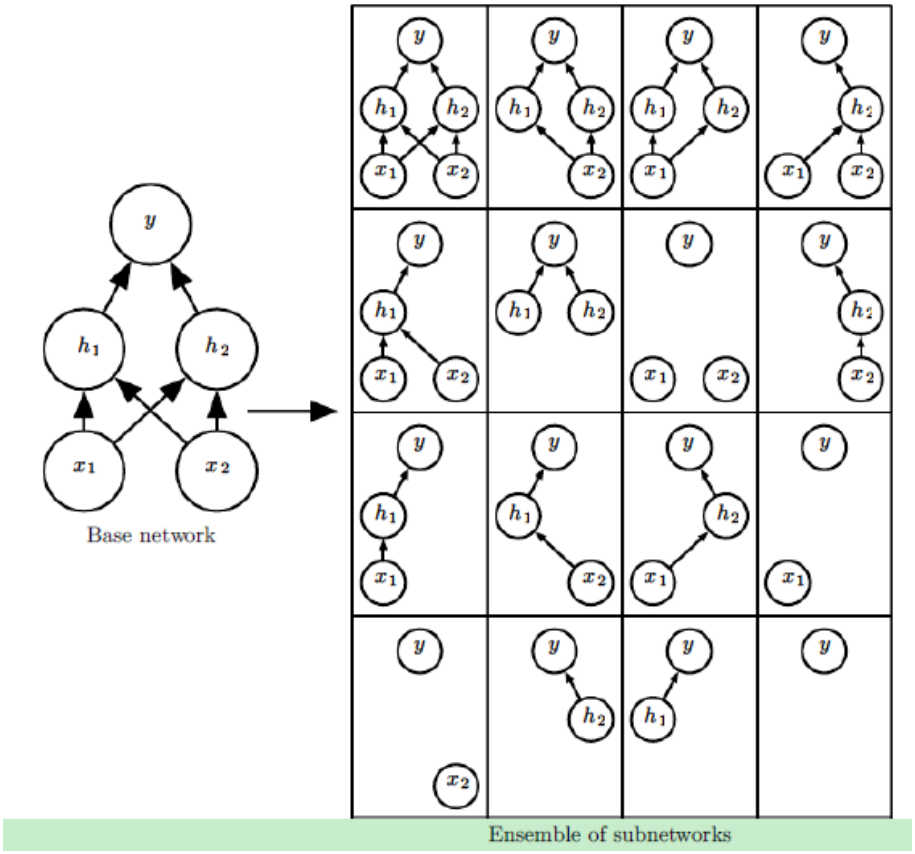
并不是所有的集成策略都是用来进行正则化的（即设计使得多个优于单个）。比如 Boosting （Freund and Schapire,1996b,a）也是一项 Ensemble 技术，但它的目的是增强集成后的模型的 Capacity（对复杂函数的表达能力），而不是用来正则化。该技术曾被用来构建神经网络的集成（Schwenk and Benjio, 1998），还能够被用来解读神经网络架构，即神经网络本质上是隐层节点的 Ensemble。

7.12 Dropout

Dropout 是一种计算量可接受，并且能够集成很多类模型的强大方法。该方法可近似看成是对非常多的大型神经网络的 Bagging。Bagging 策略要分别训练集成所需的每个模型，这对于大型深度神经网络来讲是很难实现的。一般 Bagging 中都是用 5 到 10 个模型（Szegedy et al. 2014a 用 6 个模型的 Bagging 赢得了 ISLVRC）。Dropout 是 Bagging 的近似逼近，其优

点是：1）计算量可控；2）可实现对指数级别数量的神经网络的 Bagging Ensemble。

需要特别指出，Dropout 在训练中集成多个子网络，每个子网络都是通过移除原始网络中的所谓的“non-output”神经元得到的。



我们可以通过让神经元输出乘以 0 的办法来移除很多神经元。这种操作对于某些网络而言需要一些额外的小的处理技巧，比如 RBF 网络（根据神经元与某个设定的参考值的距离来决定输出）。为了简化分析，我们一律以乘以 0 的方式来阐述 Dropout 算法。

Dropout 的训练

在 Dropout 中，因为要模拟指数级别数量的子网络，所以采用基于 minibatch 的学习方法来减少训练迭代次数，比如随机梯度下降法 SGD。每向 minibatch 中添加一个样本，我们对所有输入和隐层神经元都随机生成一个 Mask（取值为 0 或 1），各个神经元的 Mask 的生成都是相互独立的。Mask 取值为 1（神经元不被移除）的概率是模型的超参数，即神经元被保留的比例是作为超参数提前设定好的，而不是网络参数或输入样本的函数。典型的配置为：对输入层神经元保留比例设为 0.8（意味着只有 80%维度的特征被用到），而隐层神经元的保留比例设为 0.5。然后前馈、后向误差传播、学习更新等与普通训练方法一致。图 7.7 给出了 Dropout 训练机制下如何进行前馈。

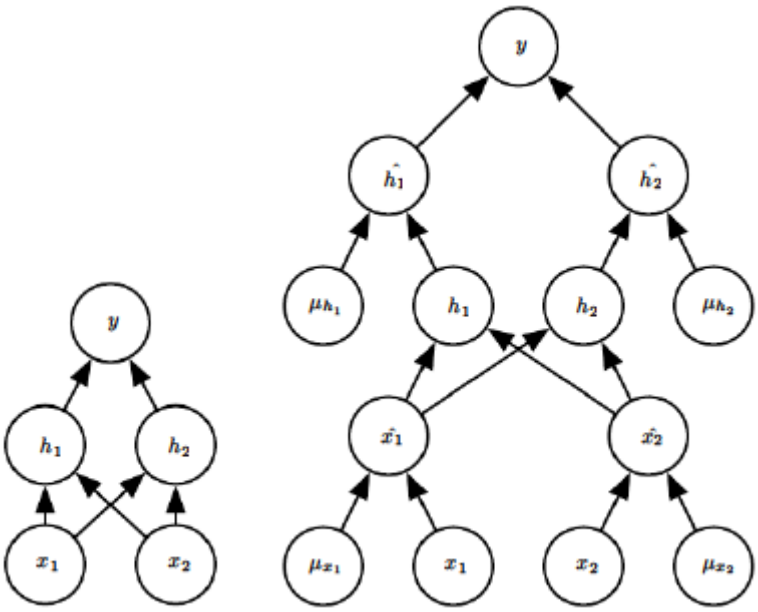


图 7.7 左图为原始网络，右图中为每个神经元加入了一个 Mask，所有 Mask 的值组成网络的筛选向量 μ，Mask 为 1 的时候该神经元才生效，每生成一个 μ 相当于从原始网络中筛选出了一个子网络。

对于筛选向量 μ 和网络参数 θ，子网络的损失函数为 $\boldsymbol{J}(\theta,\mu)$，对网络的整体优化要针对所有筛选向量 μ 取平均损失，即 $\mathbf{E}_{\mu}[\boldsymbol{J}(\theta,\mu)]$。在后向误差传播计算梯度的时候，要求平均损失的梯度，有 2 的指数（指数为神经元个数）数量级的子网络损失要计算，而这是不可能完成的。Dropout 方法采用 Monte Carlo 方法，通过对 μ 采样，来对其梯度进行无偏估计。

如前所述，在每向 Mini-Batch 中加入一个样本就有一次 μ 的采样，假定 Mini-Batch 的 size 是 k。这样当 Mini-Batch 的误差累计之后，要进行一次向的后向传播的时候，针对一个神经元，其更新值来自其之后层与之相连的所有神经元，而这些神经元按照采样好的 k 个（Mini-Batch 样本个数）μ 计算 k 次更新值，再取均值即可得到面向 μ 的平均损失的更新量。这个更新量是平均损失的，这样训练得到的网络就隐含表达了对 2 的指数数量级（神经元数量）个子网络的 Averaging（因其损失函数是面向 μ 的平均损失）。

与 Bagging 不同，Dropout 方法中的各个子网络的参数是共享的，每个子网络的参数是原始网络参数的不同子集。这种参数共享方式使得 Dropout 方法可以在有限的计算时间和内存条件下进行 2 的指数数量级的子网络的 Averaging。在 Bagging 中，每个模型都要独立收敛到其对应的训练集，在有限的时间内，不可能对大型网络遍历其所有子网络。在 Dropout 中，每一步迭代只训练很少的一部分子网络，而参数共享机制利用小部分网络的参数引导其他子网络进行更合理的搜索并获得合理的参数。上述区别是 Bagging 和 Dropout 唯一的区别。其他方面 Dropout 都采用和 Bagging 一样的思路，比如每个子网络的训练集都是从原始数据集中置回采样得到的（因为每个样本在迭代中都对应一个 μ 的采样，相当于一个子网络。所以每个子网络用到的样本都是整个样本集中的一个子集，并且各个 μ 之间是独立采样

获得的，因此子集是 sampling with replacement 的）。

Dropout 的预测

Dropout 在训练中，用共享权值的方式给出了 2 的指数次个子网络的最佳权值。这是总的网络可以看作一个节点和权值的总池子，在使用中，要获得子网络只需要从池子中随机抽取出若干节点及与其连接的权值，就可以获得一个子网络的预测。重复多次这个过程就可以对多个子网络的预测进行 averaging，完成输出。

为了进一步加速预测过程，根据经验（暂无理论证明 Hinton et al. 2012c），可将整个网络中从每个节点出来（下一层传播）的权值都进行压缩，乘以该节点被保留的概率（一般在训练中该超参数设为 0.5 左右，输入层为 0.8）。此时就可以用权值 scale 过的网络仅做一次预测就可以了。直观原理（无证明）：在多次采样预测中，每次每个节点都只接受大概 50% 的上层 unit 的输入（因为大概 50%的被移除了），所以 unit 接受的值是整个网络在该 unit 输入的 50%）。这项技术称为“Weight Scaling Inference Rule，WSIR”。

对于很多没有非线性隐层神经元的模型而言，WSIR 技术是可以精确的解析推导出来的。式（7.56）至式（7.66）给出了 Softmax Regression 分类器的例子。

WSIR 技术对于其他很多线性模型也是可以解析推导出来的，而且对于很多使用非线性隐层神经元的模型也往往具有很好的效果。Goodfellow（2013a）发现在分类问题中其比蒙特卡洛方法（上面黑色字体下面的第一段）效果更好，甚至是使用 1000 次子网络的蒙特卡洛也比不上 WSIR。但是 Gal and Ghahramani（2015）发现在某些分类问题中用 20 次子网络采样的 Monte Carlo 方法就胜过了 WSIR。这说明这种方法的性能还是与具体的问题紧密相关的。

Srivastava et al. (2014)证明了 Dropout 方法比权值衰减、范数约束、稀疏表达等都更有效。并且该方法可以和其他方法结合进一步提高网络的泛化能力。

Dropout 的优点总结

1. 快速性：计算量不大，尤其在预测（inference）阶段，比普通方法多出来的计算量只是权值的缩放。
2. 普适性：任何采用梯度下降法和分布式节点的方法都可以使用 dropout，比如前馈网络、受限玻尔兹曼机（2014）、递归神经网络（2014）。

什么时候不用 Dropout

Dropout 在训练的时候一定要保证网络的 Size 足够大，否则子网络没有足够的 Capacity，网络没有用。因此训练比较耗时。有些时候训练样本集非常完备，不用 dropout 也可以获得较好的泛化能力。此时就要权衡是否还需要用 Dropout 了。

当有标号的样本非常少的时候，Bayesian neural network（1996）比 Dropout 更好。当具有大量无标号数据的时候，非监督分类也比 Dropout 好。

Dropout 与权值衰减的关系

对于线性模型 Dropout 相当于权值衰减，每个特征的衰减系数由该维特征的方差确定。对于深度模型，Dropout 与权值衰减没有关联。

Fast Dropout

Wang and Manning (2013)提出了一种快速 Dropout 方法。性能未见明显提高但速度快些。

不使用{0,1}Mask 的 Dropout

Srivastava et al. (2014)证明 μ 不再使用 Bernoulli 分布，而是使用正态分布 $N(\mathbf{1}, \boldsymbol{I})$的时候，效果更好，此时在预测（inference）阶段不用再进行权值 scale，因为 μ 的均值为 1。

从生物学启事的角度解读 Dropout

提出 Dropout 的 Hinton（2012c）是从生物学的有性繁殖中获得的启发。在繁衍中，后代的基因要来自于父体和母体，而有些父体和母体相同的基因要随机的赋予后代，即后代的某些基因随机的来自于父体或母体。这就要求这些基因本身具有很强的稳定性和代表性。就像 Dropout 网络中的隐层 unit 输出一样，由于每个 unit 都要做好在其他 unit 不在的情况下独立表达数据的准备，因此每个 unit 的输出都是非常好的中层数据表达，稳定而又代表性。

从加噪的角度理解 Dropout

Dropout 本质上是对隐层特征的输出加噪（Bernoulli 或高斯）。对隐层特征输出加噪而不是对原始数据加噪是有道理的。比如人脸识别中，某个隐层输出对应鼻子，其到一定程度上决定了其后的全连接层可以将其识别为人脸。现在这个隐层被屏蔽了，鼻子无法输入到全连接层。网络必须找到其他冗余信息或替代信息（如嘴）来时全连接层仍然认识这是人脸。所以 Dropout 网络具有很强的泛化能力。而原始数据层面的加噪是做不到这一点的。直接对抽象出来的中层特征加噪来提高泛化能力更加直观有力。

还有一个重要方面，Dropout 中加入的噪声是乘性的。如果是加性噪声的话，ReLU 的输出 h 在学习中会变的很大，使噪声 ϵ 与 h 不可比，从而抑制噪声。H 的爆炸式增加对于网络是非常不利的，而乘性噪声不会导致这种情况。

Dropout 与 Batch Normalization 的关系

Batch Normalization 也会对神经元的输出（实际上是在激活之前）加入噪声，并且是乘性和加性噪声都有。其本质上是为了提高目标函数的优化求解过程，但是噪声的加入也具有较高的正则化效果。因此有时候 BN 的应用会使 Dropout 变得不是那么必不可少。关于 BN 会在 8.7.1 中详细探讨。

7.13 竞争性训练 Adversarial Training

由于构建网络的基础单元大多具有一定的线性性质，深度神经网络模型也一般都具有较

强的线性性质。这种性质是的神经网络相对更容易优化，单也是网络变得非常不稳定。比如对样本加入一个小的加性偏移 ϵ，网络的输出为 $\mathbf{W}^T\left(\mathbf{X}+\varepsilon\right)=\mathbf{W}^T\mathbf{X}+\mathbf{W}^T\varepsilon$，最多可以偏移 $\varepsilon\left\|\mathbf{W}\right\|_1=\varepsilon\sum\left|w_i\right|$。当网络的维度很高的时候，网络输出的偏移量非常大。这就导致了相近的输入不能获得相同的输出。

对这种情况进行正则化的方法就是竞争性训练，即人工得到离某个训练样本非常接近的人工竞争性样本，并把竞争性样本加入到训练集中。这可以极大提高网络的能力，相当于对网络输出的局部一致性做了先验约束正则化。

下面例子给出了竞争性样本的实例。

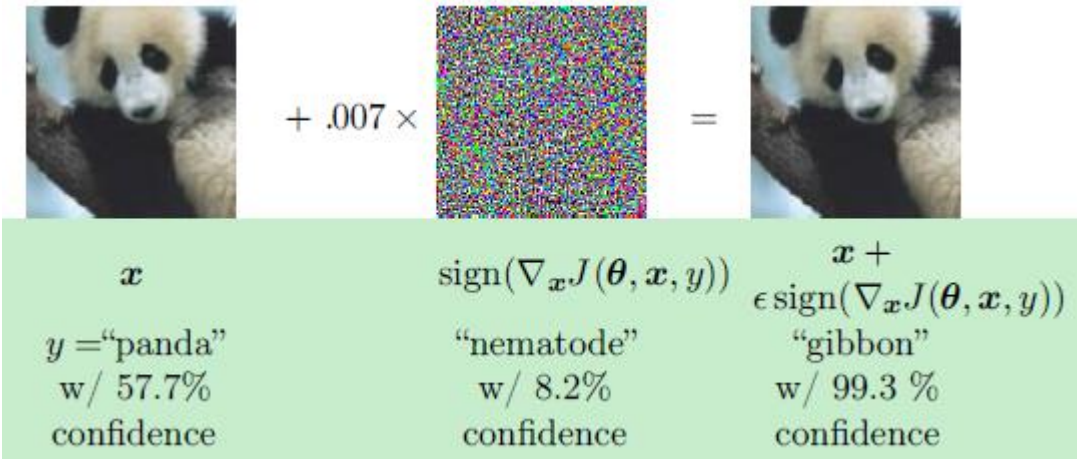


图 7.8 应用于 GoogLeNet（Szegedy et al., 2014a）的竞争性样本，数据集来自 ImageNet。通过在原始样本中加入一个很小的向量（目标函数对样本的导数的 sign 函数值），GoogLeNet 的分类结果发生了剧烈变化。把加偏移后的熊猫识别为了长臂猿。

纯线性方法如逻辑回归是不能用这种方法进行正则化的，因为模型本身就是纯线性的。但神经网络具有很强的弹性，可以接近线性，也可以具有很高的局部一致性（局部稳定性）。因此是既可以学习捕获数据的线性趋势又可以学习竞争性样本来抵抗样本的局部扰动。

竞争性样本也提供了一种半监督学习的方法。对于无标号数据 x，模型对其预测的标号为 y，虽然这个 y 不是真值，但当模型质量较好时，y 是有较高置信度的。此时可以通过微小扰动构建 x 的竞争样本 x'，使当前模型对 x'的预测为 y'（y'不等于 y）。此时的 x'成为 Virtual Adversarial Example（Miyato et al., 2015）。之后模型会调整使得 x 和 x'的预测结果一致。变得更鲁棒。这种做法的初衷是不同的类别通常分布于不连通的数据流形中，微小的扰动不会是 x 从一个类别的流形跳到另外一个类别的流形中。

7.14 正切距离、正切支柱、流形正切分类器 Tangent Distance, Tangent Prop, and Manifold Tangent Classifier

许多机器学习方法假设数据分布在低维流形上以克服维数灾难的影响（5.11.3）。

最早的尝试使用流形假设的方法就是 Tangent Distance（Simard et al., 1993, 1998）。这

是一个非参数最近邻算法。**该算法不用欧氏距离而是使用基于流形的距离。**算法的基本假设是在同样的流形中的样本具有同样的类别，由于分类器的分类要对流形中的微小变化具有不变形（不因流形中的微小移动而导致分类结果变化）。这时，使用点 x1 和 x2 各自所在的流形的最近距离来度量 x1 和 x2 的距离是合理的。虽然这可能计算比较困难（计算连个流形中的最近点对是耗时的优化问题），但是可以用流形在 x 处的切平面来获得简化计算方法。

类似的想法可以用在 Tangent Prop 算法和 Manifold Tangent 分类器上。图 7.9 阐述了这种方法。

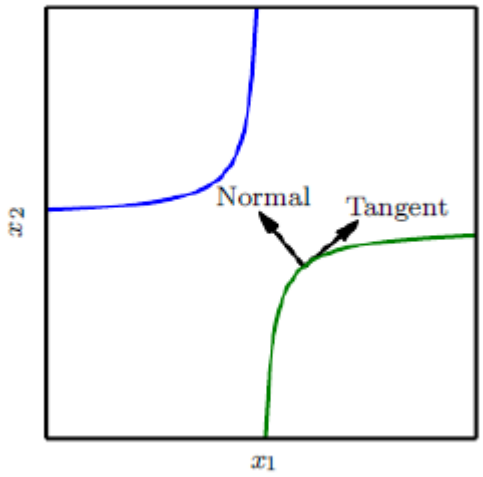


图 7.9 Tangent Prop 和 Manifold Tangent Classifier

神经网络训练的时候，要加入两个正则项，一个约束 x 在沿流形法线方向移动的时候不要是网络输出变化过慢。另外一个约束 x 在沿流形切线方向移动的时候网络输出不要变化过快。

结语：

本章讲述了大部分神经网络正则化的策略。正则化是机器学习的一个核心话题，本书的很多章节会反复的回顾和用到本章的知识。深度学习的另外一个核心话题是优化问题，下一张讨论优化问题。