正则化

1. 正则化常用于缓解模型过拟合。过拟合发生的原因是模型的容量过大，而正则化可以对模型施加某些限制，从而降低模型的有效容量。
2. 目前有多种正则化策略。
   - 有些正则化策略是向模型添加额外的约束，如增加对参数的限制。这是对参数的硬约束。
   - 有些正则化策略是向目标函数增加额外项。这是对参数的软约束。
3. 正则化策略代表了某种先验知识，即：倾向于选择简单的模型。
4. 在深度学习中，大多数正则化策略都是基于对参数进行正则化。正则化以偏差的增加来换取方差的减少，而一个有效的正则化能显著降低方差，并且不会过度增加偏差。
5. 在深度学习的实际应用中，不要因为害怕过拟合而采用一个小模型，推荐采用一个大模型并使用正则化。

一、参数范数正则化

1. 一些正则化方法通过对目标函数 J 添加一个参数范数正则化项 $\Omega(\vec{\theta})$ 来限制模型的容量`capacity` 。

   正则化之后的目标函数为 $\tilde{J}$：$\tilde{J}(\vec{\theta};\mathbf{X},\vec{\mathbf{y}})=J(\vec{\theta};\mathbf{X},\vec{\mathbf{y}})+\alpha\Omega(\vec{\theta})$ 。
   - $\alpha\in[0,\infty)$ 为正则化项的系数，它衡量正则化项 $\Omega(\vec{\theta})$ 和标准目标函数 J 的比重。
     - $\alpha=0$ 则没有正则化。
     - α 越大则正则化项越重要。
   - 如果最小化 $\tilde{J}$，则会同时降低 J 和参数 $\vec{\theta}$ 的规模。
2. 参数范数正则化可以缓解过拟合。

   如果 α 设置的足够大，则参数 $\vec{\theta}$ 就越接近零。这意味着模型变得更简单，简单的模型不容易过拟合（但是可能欠拟合）。

   对于神经网络，这意味着很多隐单元的权重接近0，于是这些隐单元在网络中不起任何作用。此时大的神经网络会变成一个小的网络。

   在 α 从 零逐渐增加的过程中存在一个中间值，使得参数 $\vec{\theta}$ 的大小合适，即一个合适的模型。
3. 选择不同的 Ω 的形式会产生不同的解，常见的形式有 L_2 正则化和 L_1 正则化。

1.1 L2 正则化

1. L_2 正则化通常被称作岭回归或者`Tikhonov`正则化。
   - 正则化项为 $\Omega(\vec{\theta})=\frac{1}{2}||\vec{\theta}||_2^2$。系数 $\frac{1}{2}$ 是为了使得导数的系数为 1。
   - 该正则化形式倾向于使得参数 $\vec{\theta}$ 更接近零。
2. 假设 $\vec{\theta}$ 参数就是权重 $\vec{\mathbf{w}}$，没有偏置参数，则：$\tilde{J}(\vec{\mathbf{w}};\mathbf{X},\vec{\mathbf{y}})=J(\vec{\mathbf{w}};\mathbf{X},\vec{\mathbf{y}})+\frac{\alpha}{2}\vec{\mathbf{w}}^T\vec{\mathbf{w}}$ 。

   对应的梯度为：$\nabla_{\vec{\mathbf{w}}}\tilde{J}(\vec{\mathbf{w}};\mathbf{X},\vec{\mathbf{y}})=\nabla_{\vec{\mathbf{w}}}J(\vec{\mathbf{w}};\mathbf{X},\vec{\mathbf{y}})+\alpha\vec{\mathbf{w}}$ 。

   使用梯度下降法来更新权重，则权重的更新公式为：$\vec{\mathbf{w}}\leftarrow\vec{\mathbf{w}}-\epsilon(\nabla_{\vec{\mathbf{w}}}J(\vec{\mathbf{w}};\mathbf{X},\vec{\mathbf{y}})+\alpha\vec{\mathbf{w}})$ 。即：

   $$\vec{\mathbf{w}}\leftarrow(1-\epsilon\alpha)\vec{\mathbf{w}}-\epsilon\nabla_{\vec{\mathbf{w}}}J(\vec{\mathbf{w}};\mathbf{X},\vec{\mathbf{y}})$$

   L_2 正则化对于梯度更新的影响是：每一步执行梯度更新之前，会对权重向量乘以一个常数因子来收缩权重向量。因此`L2`正则化也被称作“权重衰减”。

1.1.1 整体影响

1. 令 $\vec{\mathbf{w}}^* = \arg\min_{\vec{\mathbf{w}}} J(\vec{\mathbf{w}})$，它就是无正则化项时使得目标函数最小的权重向量。

   根据极小值的条件，有 $\nabla_{\vec{\mathbf{w}}} J(\vec{\mathbf{w}}^*) = \vec{\mathbf{0}}$。于是在 $\vec{\mathbf{w}}^*$ 的邻域内泰勒展开 $J(\vec{\mathbf{w}})$：

   $$\hat{J}(\vec{\mathbf{w}}) = J(\vec{\mathbf{w}}^*) + \vec{\mathbf{0}} + \frac{1}{2}(\vec{\mathbf{w}} - \vec{\mathbf{w}}^*)^T \mathbf{H}(\vec{\mathbf{w}} - \vec{\mathbf{w}}^*), \quad \vec{\mathbf{w}} \in \mathbb{N}(\vec{\mathbf{w}}^*)$$

   其中：$\mathbf{H}$ 为 $J(\vec{\mathbf{w}})$ 在 $\vec{\mathbf{w}}^*$ 处的海森矩阵；$\mathbb{N}(\vec{\mathbf{w}}^*)$ 为 $\vec{\mathbf{w}}^*$ 处的一个邻域。

   则 $\hat{J}(\vec{\mathbf{w}})$ 的梯度为：$\nabla_{\vec{\mathbf{w}}} \hat{J}(\vec{\mathbf{w}}) = \mathbf{H}(\vec{\mathbf{w}} - \vec{\mathbf{w}}^*), \quad \vec{\mathbf{w}} \in \mathbb{N}(\vec{\mathbf{w}}^*)$。

2. 令 $\tilde{\vec{\mathbf{w}}}^* = \arg\min_{\vec{\mathbf{w}}} \tilde{J}(\vec{\mathbf{w}})$，它就是有正则化项时使得目标函数最小的权重向量。

   假设 $\tilde{\vec{\mathbf{w}}}^* \in \mathbb{N}(\vec{\mathbf{w}}^*)$，即 $\tilde{\vec{\mathbf{w}}}^*$ 在 $\vec{\mathbf{w}}^*$ 的一个邻域内，则有：$\nabla_{\vec{\mathbf{w}}} J(\tilde{\vec{\mathbf{w}}}^*) = \mathbf{H}(\tilde{\vec{\mathbf{w}}}^* - \vec{\mathbf{w}}^*)$。

   根据极小值条件：$\nabla_{\vec{\mathbf{w}}} \tilde{J}(\tilde{\vec{\mathbf{w}}}^*) = \nabla_{\vec{\mathbf{w}}} J(\tilde{\vec{\mathbf{w}}}^*) + \alpha \tilde{\vec{\mathbf{w}}}^* = \vec{\mathbf{0}}$，则有：

   $$\mathbf{H}(\tilde{\vec{\mathbf{w}}}^* - \vec{\mathbf{w}}^*) + \alpha \tilde{\vec{\mathbf{w}}}^* = \vec{\mathbf{0}} \rightarrow (\mathbf{H} + \alpha \mathbf{I})\tilde{\vec{\mathbf{w}}}^* = \mathbf{H}\vec{\mathbf{w}}^*$$
   $$\rightarrow \tilde{\vec{\mathbf{w}}}^* = (\mathbf{H} + \alpha \mathbf{I})^{-1}\mathbf{H}\vec{\mathbf{w}}^*$$

   当 $\alpha \rightarrow 0$ 时，$\tilde{\vec{\mathbf{w}}}^* \rightarrow \vec{\mathbf{w}}^*$。

3. 因为 $\mathbf{H}$ 是实对称矩阵，对其进行特征值分解：$\mathbf{H} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$。其中特征值组成对角矩阵 $\mathbf{\Lambda}$，对应的特征向量组成正交矩阵 $\mathbf{Q}$：

   $$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}$$

   于是有：

   $$\tilde{\vec{\mathbf{w}}}^* = (\mathbf{H} + \alpha \mathbf{I})^{-1}\mathbf{H}\vec{\mathbf{w}}^* = (\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T + \alpha \mathbf{I})^{-1}\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T\vec{\mathbf{w}}^*$$
   $$= [\mathbf{Q}(\mathbf{\Lambda} + \alpha \mathbf{I})\mathbf{Q}^T]^{-1}\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T\vec{\mathbf{w}}^* = \mathbf{Q}(\mathbf{\Lambda} + \alpha \mathbf{I})^{-1}\mathbf{\Lambda}\mathbf{Q}^T\vec{\mathbf{w}}^*$$

   其中：

   $$(\mathbf{\Lambda} + \alpha \mathbf{I})^{-1}\mathbf{\Lambda} = \begin{bmatrix} \frac{\lambda_1}{\lambda_1+\alpha} & 0 & \cdots & 0 \\ 0 & \frac{\lambda_2}{\lambda_2+\alpha} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\lambda_n}{\lambda_n+\alpha} \end{bmatrix}$$

4. L_2 正则化对模型整体的影响：沿着 $\mathbf{H}$ 的特征向量所定义的轴来缩放 $\vec{\mathbf{w}}^*$。
   - $\mathbf{H}$ 的第 i 个特征向量对应的 $\vec{\mathbf{w}}^*$ 分量根据 $\frac{\lambda_i}{\lambda_i+\alpha}$ 因子缩放。
   - 沿着 $\mathbf{H}$ 特征值较大的方向受到正则化的影响较小。
   - 当 $\lambda_i \ll \alpha$ 的方向对应的权重分量将被缩小到几乎为零。

1.1.2 物理意义

1. 如下所示：实线椭圆表示 J 的等值线，虚线圆表示正则化项 $\frac{\alpha}{2}\vec{\mathbf{w}}^T\vec{\mathbf{w}}$ 的等值线。

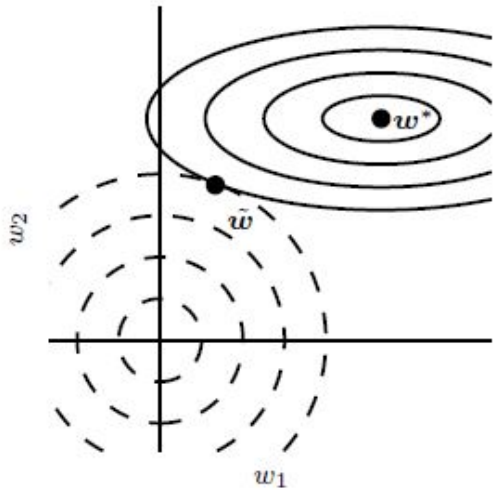
在 $\vec{\mathbf{w}}^*$ 点， J 取得最小值；在 $\tilde{\vec{\mathbf{w}}}^*$ 点（也就是图中的 $\tilde{w}$ 点），J 和正则化项达到平衡（使得二者之和最小）。

沿着 w_1 方向（横向）的 J 的曲率半径较大；曲率半径越大，曲率越小，特征值越小。

- 曲率刻画曲线的弯曲程度。弯曲越厉害，则表示曲率半径越小、曲率越大。

  直线的曲率半径为 $+\infty$ ，曲率为0。

- 曲率半径是曲率的倒数。对于椭圆 $\frac{w_1^2}{a^2}+\frac{w_2^2}{b^2}=1$：
  - 在左右顶点：沿着 w_2 方向（纵向）的曲率半径为 $\frac{b^2}{a}$ 。
  - 在上下顶点：沿着 w_1 方向（横向）的曲率半径为 $\frac{a^2}{b}$ 。
  - 海森矩阵的特征值为： $\lambda_1=\frac{2}{a^2},\lambda_2=\frac{2}{b^2}$ 。



2. 在上图中：

   - J 的海森矩阵第一维 （w_1 ）的特征值很小。

     所以当从 $\vec{\mathbf{w}}^*$ 点水平移动时， J 不会增加太多。因为 J 对这个方向没有强烈的偏好。所以正则化项对于该轴具有强烈的影响：正则化项将 w_1 拉向零。

   - J 的海森矩阵第二维的特征值较大。

     J 对于 w_2 的变化非常敏感，因此正则化项对于该轴影响较小。

   - 因为沿着水平方向，一个较大的偏移只会对 J 产生一个较小的变化。因此正则化项倾向于从 $\vec{\mathbf{w}}^*$ 点水平向零点移动。

3. L_2 正则化表明：

   - 只有显著减小目标函数 J 的那个方向的参数会相对保留下来。
   - 无助于减小目标函数 J 的方向（该方向上$\mathbf{H}$ 特征值较小，或者说该方向上 J 的曲率较小，或者说该方向上 J 的曲线更接近于直线），因为在这个方向上移动不会显著改变梯度，因此这个不重要方向上的分量会因为正则化的引入而被衰减掉。

1.1.3 示例

1. 考虑线性回归的 L_2 正则化，采用平方误差作为代价函数：

$$J=(\mathbf{X}\vec{\mathbf{w}}-\vec{\mathbf{y}})^T(\mathbf{X}\vec{\mathbf{w}}-\vec{\mathbf{y}})$$

$$\tilde{J}=J+\frac{\alpha}{2}\vec{\mathbf{w}}^T\vec{\mathbf{w}}=(\mathbf{X}\vec{\mathbf{w}}-\vec{\mathbf{y}})^T(\mathbf{X}\vec{\mathbf{w}}-\vec{\mathbf{y}})+\frac{\alpha}{2}\vec{\mathbf{w}}^T\vec{\mathbf{w}}$$

这里忽略了线性回归的 $\vec{\mathbf{b}}$ 的影响，这是为了便于说明解的性质。

2. $\vec{\mathbf{w}}^* = \arg\min_{\vec{\mathbf{w}}} J(\vec{\mathbf{w}})$ 的解析解为：$\vec{\mathbf{w}}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\vec{\mathbf{y}}$ 。

   $\tilde{\vec{\mathbf{w}}}^* = \arg\min_{\vec{\mathbf{w}}} \tilde{J}(\vec{\mathbf{w}})$ 的解析解为：$\tilde{\vec{\mathbf{w}}}^* = (\mathbf{X}^T\mathbf{X} + \alpha\mathbf{I})^{-1}\mathbf{X}^T\vec{\mathbf{y}}$ 。

   样本的协方差矩阵为 $\mathbf{\Sigma} = \frac{1}{N}\mathbf{X}^T\mathbf{X}$ （这里已经将样本进行了标准化：减去了均值）， N 为样本数量。因此 $\mathbf{X}^T\mathbf{X}$ 的对角线对应于每个输入特征的方差，$\mathbf{X}^T\mathbf{X} + \alpha\mathbf{I}$ 在对角线上增加了 α。

   因此，L_2 正则化使得：

   - 方差较小的特征对应的权重被收缩。
   - 方差远大于 α 的特征受影响较小。
   - 只有方差接近甚至小于 α 的特征受影响较大。

1.2 L1 正则化

1. 模型参数 $\vec{\mathbf{w}}$ 的 L_1 的正则化形式为：$\Omega(\vec{\theta}) = ||\vec{\mathbf{w}}||_1 = \sum_i |w_i|$ 。即各个参数的绝对值之和。
2. L_1 正则化后的目标函数 $\tilde{J}(\vec{\mathbf{w}};\mathbf{X},\vec{\mathbf{y}})$：$\tilde{J}(\vec{\mathbf{w}};\mathbf{X},\vec{\mathbf{y}}) = J(\vec{\mathbf{w}};\mathbf{X},\vec{\mathbf{y}}) + \alpha||\vec{\mathbf{w}}||_1$ 。

   对应的梯度为 $\nabla_{\vec{\mathbf{w}}}\tilde{J}(\vec{\mathbf{w}};\mathbf{X},\vec{\mathbf{y}}) = \nabla_{\vec{\mathbf{w}}}J(\vec{\mathbf{w}};\mathbf{X},\vec{\mathbf{y}}) + \alpha\text{sign}(\vec{\mathbf{w}})$ 。其中 $\text{sign}(\cdot)$ 函数取自变量的符号：

   如果自变量大于零，则取值为 1；如果自变量小于零，则取值为 -1；如果自变量为零，则取值为零。

   使用梯度下降法来更新权重，给出权重的更新公式为：

$$\begin{aligned}\vec{\mathbf{w}} &\leftarrow \vec{\mathbf{w}} - \epsilon(\nabla_{\vec{\mathbf{w}}}J(\vec{\mathbf{w}};\mathbf{X},\vec{\mathbf{y}}) + \alpha\text{sign}(\vec{\mathbf{w}}))\\ &= (\vec{\mathbf{w}} - \epsilon\alpha\text{sign}(\vec{\mathbf{w}})) - \epsilon\nabla_{\vec{\mathbf{w}}}J(\vec{\mathbf{w}};\mathbf{X},\vec{\mathbf{y}})\end{aligned}$$

   L_1 正则化对于梯度更新的影响是：不再是线性地缩放每个 w_i （L_2 正则化项的效果），而是减去与 $\text{sign}(w_i)$ 同号的常数因子。

1.2.1 整体效果

1. 令 $\vec{\mathbf{w}}^* = \arg\min_{\vec{\mathbf{w}}} J(\vec{\mathbf{w}})$，它就是无正则化项时使得目标函数最小的权重向量。

   和 L_2 正则化中的推导相同，在 $\vec{\mathbf{w}}^*$ 的邻域内泰勒展开：

$$\hat{J}(\vec{\mathbf{w}}) = J(\vec{\mathbf{w}}^*) + \vec{\mathbf{0}} + \frac{1}{2}(\vec{\mathbf{w}} - \vec{\mathbf{w}}^*)^T\mathbf{H}(\vec{\mathbf{w}} - \vec{\mathbf{w}}^*), \quad \vec{\mathbf{w}} \in \mathbb{N}(\vec{\mathbf{w}}^*)$$

   其中： $\mathbf{H}$ 为 J 在 $\vec{\mathbf{w}}^*$ 处的海森矩阵； $\vec{\mathbf{w}}$ 在 $\vec{\mathbf{w}}^*$ 的邻域 $\mathbb{N}(\vec{\mathbf{w}}^*)$内。

2. 由于 L_1 正则化项在一般的海森矩阵情况下无法得到直接的代数表达式。

   因此我们进一步假设海森矩阵是对角矩阵。即：

$$\mathbf{H} = \begin{bmatrix} H_{1,1} & 0 & \cdots & 0 \\ 0 & H_{2,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & H_{n,n} \end{bmatrix}$$

   其中 $H_{i,i} > 0, i = 1,2,\cdots,n$

> 如果用于线性回归问题的数据已经被预处理（如使用 PCA），去除了输入特征之间的相关性，则这一假设成立。

于是：

$$\hat{J}(\vec{\mathbf{w}}) = J(\vec{\mathbf{w}}^*) + \frac{1}{2}(\vec{\mathbf{w}} - \vec{\mathbf{w}}^*)^T \mathbf{H}(\vec{\mathbf{w}} - \vec{\mathbf{w}}^*)$$
$$= J(\vec{\mathbf{w}}^*) + \sum_i \left[\frac{1}{2}H_{i,i}(w_i - w_i^*)^2\right], \quad \vec{\mathbf{w}} \in \mathbb{N}(\vec{\mathbf{w}}^*)$$

3. 考虑定义式，有：

$$\tilde{J}(\vec{\mathbf{w}}) = J(\vec{\mathbf{w}}) + \alpha||\vec{\mathbf{w}}||_1 = \hat{J}(\vec{\mathbf{w}}) + \alpha||\vec{\mathbf{w}}||_1$$
$$= J(\vec{\mathbf{w}}^*) + \sum_i \left[\frac{1}{2}H_{i,i}(w_i - w_i^*)^2 + \alpha|w_i|\right], \quad \vec{\mathbf{w}} \in \mathbb{N}(\vec{\mathbf{w}}^*)$$

对于 $\vec{\mathbf{w}}$ 来讲， $J(\vec{\mathbf{w}}^*)$ 为常量。因此 $\tilde{J}(\vec{\mathbf{w}})$ 的最小值由 $\sum_i \left[\frac{1}{2}H_{i,i}(w_i - w_i^*)^2 + \alpha|w_i|\right]$ 决定。

考虑每一个维度 i，可以考虑最优化目标：

$$\tilde{w}_i^* = \arg\min_{w_i} \left[\frac{1}{2}H_{i,i}(w_i - w_i^*)^2 + \alpha|w_i|\right]$$

得到解析解：$\tilde{w}_i^* = \text{sign}(w_i^*)\max\left\{|w_i^*| - \frac{\alpha}{H_{i,i}}, 0\right\}$ 。

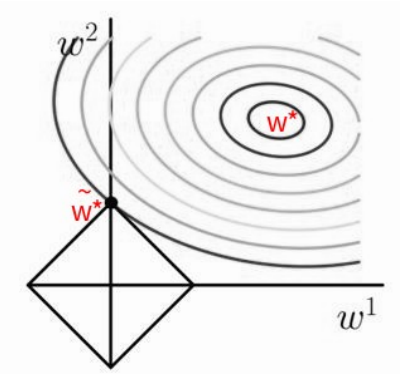
4. 考虑 $w_i^* > 0$ 的情况。此时有两种可能：
   - $w_i^* \le \frac{\alpha}{H_{i,i}}$：则 $\tilde{w}_i^* = 0$。表示 L_1 正则化项将 w_i 推向 0 。
   - $w_i^* > \frac{\alpha}{H_{i,i}}$：则 $\tilde{w}_i^* = w_i^* - \frac{\alpha}{H_{i,i}}$。此时 L_1 正则化项并不会将 w_i 推向 0，而是向零的方向推动了 $\frac{\alpha}{H_{i,i}}$ 的距离。
5. 考虑 $w_i^* < 0$ 的情况。此时有两种可能：
   - $w_i^* \ge -\frac{\alpha}{H_{i,i}}$：则 $\tilde{w}_i^* = 0$。表示 L_1 正则化项将 w_i 推向 0 。
   - $w_i^* < -\frac{\alpha}{H_{i,i}}$：则 $\tilde{w}_i^* = w_i^* + \frac{\alpha}{H_{i,i}}$。此时 L_1 正则化项并不会将 w_i 推向 0，而是向零的方向推动了 $\frac{\alpha}{H_{i,i}}$ 的距离。

> 如果使用 L_2 正则化，则解为 $\tilde{w}_i^* = \frac{H_{i,i}}{H_{i,i}+\alpha}w_i^*$ 。

1.2.2 物理意义

1. 如下所示：实线椭圆表示 J 的等值线，实线菱形表示正则化项 $\alpha||\vec{\mathbf{w}}||_1$ 的等值线。

   在 $\vec{\mathbf{w}}^*$ 点， J 取得最小值；在 $\tilde{\vec{\mathbf{w}}}^*$ 点（也就是图中的 $\tilde{w}$ 点），J 和正则化项达到平衡（使得二者之和最小）。



可以看到 J 的等值线更容易与 L_1 正则化项的等值线在坐标轴相交从而取得整体极小值。

2. L_1 正则化项更容易产生稀疏(sparse)解，而 L_2 正则化并不会导致稀疏解。
   - 在 L_1 正则化中， w_i^* 的绝对值越小，该维的特征越容易被稀疏化。
   - L_1 正则化的这一性质已经被广泛地用作特征选择： L_1 正则化使得部分特征子集的权重为零，表明相应的特征可以被安全地忽略。

1.3 L1/L2正则化与最大后验估计

1. 许多正则化策略可以被解释为最大后验估计 MAP ：

$$\vec{\theta}_{MAP}=\arg\max_{\vec{\theta}}p(\vec{\theta}\mid\vec{\mathbf{x}})=\arg\max_{\vec{\theta}}p(\vec{\mathbf{x}}\mid\vec{\theta})+\log p(\vec{\theta})$$

   最大化后验估计等价于最小化代价函数。
   - L_2 正则化项：参数的先验分布为高斯分布：

$$\log p(\vec{\mathbf{w}})=\log\mathcal{N}(\vec{\mathbf{w}};0,\frac{1}{\alpha}\mathbf{I})=-\frac{\alpha}{2}\vec{\mathbf{w}}^T\vec{\mathbf{w}}+\frac{n}{2}\log\frac{\alpha}{2\pi}$$

     忽略 $\frac{n}{2}\log\frac{\alpha}{2\pi}$ 项，因为它们与 $\vec{\mathbf{w}}$ 无关。
   - L_1 正则化项：参数的先验分布为各向同性拉普拉斯分布 ：

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

     忽略 $n\log\frac{\alpha}{2}$ 项，因为它们与 $\vec{\mathbf{w}}$ 无关。
2. 更复杂的正则化项可以通过先验分布为混合高斯分布得到。

二、显式约束正则化

1. 可以通过添加一个显式约束来实现正则化：$\min_{\vec{\theta}}J(\vec{\theta};\mathbf{X},\vec{\mathbf{y}}),\quad st.\ \Omega(\vec{\theta})<k$ 。其中 k 为一个常数。
2. 可以通过构建广义拉格朗日函数来求解该约束最优化问题。

   定义广义拉格朗日函数： $\mathcal{L}(\vec{\theta},\alpha)=J(\vec{\theta})+\alpha(\Omega(\vec{\theta})-k)$ 。则上述约束最优化问题的解由下式给出：

$$\vec{\theta}^*=\arg\min_{\vec{\theta}}\max_{\alpha,\alpha>0}\mathcal{L}(\vec{\theta},\alpha)$$

   假设 α 的解为 α^*，固定 α^* 则：$\vec{\theta}^*=\arg\min_{\vec{\theta}}J(\vec{\theta})+\alpha^*\Omega(\vec{\theta})$ 。

   这和参数范数正则化是相同的，因此可以将参数范数正则化视为对参数强加的约束：
   - 如果 Ω 是 L_2 范数，则权重就是被约束在一个 L_2 球中。
   - 如果 Ω 是 L_1 范数，则权重就是被约束在一个 L_1 限制的区间中。
3. 也可以通过重投影来求解该约束最优化问题。此时需要修改梯度下降算法：首先计算 $J(\vec{\theta})$ 的下降步，然后将 $\vec{\theta}$ 投影到满足 $\Omega(\vec{\theta})<k$ 的最近点。
4. 使用显式约束，而不是使用范数正则化有两个好处：
   - 采用范数正则化后，当 $\vec{\theta}$ 较小时容易使得非凸优化的过程陷入局部极小值。
     - 当使用权重范数的正则化时，较小的权重可能是局部最优的。
     - 当使用显式约束时，算法不鼓励权重接近原点，因此工作的较好。
   - 使用显式约束对优化过程增加了一定的稳定性。

如：当使用了较高的学习率时，很可能进入了正反馈：较大的权重产生了较大的梯度，较大的梯度诱发权重的更大的更新。

如果这些更新持续增加了权重的大小，则 $\vec{\theta}$ 就会迅速增大直到溢出。显式约束可以防止这种反馈环引起的权重的无限制持续增加。

5. `Srebro and Shraibman` 提供了一种正则化策略：约束神经网络的权重矩阵每列的范数，而不是限制整个权重矩阵的 `Frobenius` 范数。分别限制每一列的范数可以防止某一个隐单元有非常大的权重。

   在实践中，列范数的限制总是通过重投影的显式约束来实现。

三、数据集增强

1. 提高模型泛化能力的一个最直接的方法是采用更多的数据来训练。但是通常在现实任务中，我们拥有的数据量有限。

   解决该问题的一种方法是：创建一些虚拟的数据用于训练。

2. 数据集增强仅仅用于模型的训练，而不是用于模型的预测。即：不能对测试集、验证集执行数据集增强。

3. 当比较机器学习算法基准测试的结果时，必须考虑是否采用了数据集增强。

   通常情况下，人工设计的数据集增强方案可以大大减少模型的泛化误差。当两个模型的泛化性能比较时，应该确保这两个模型使用同一套人工设计的数据集增强方案。

4. 注意数据集增强和预处理的区别：数据集增强会产生更多的输入数据，而数据预处理产生的输入数据数量不变。

3.1 线性变换

1. 对于某些任务来说，创建虚拟数据非常困难。如：在密度估计任务中，除非预先知道了密度函数，否则无法产生新的虚拟数据。

   对于分类问题来说，创建虚拟数据非常简单。对于一个分类器，它将高维的输入 $\vec{\mathbf{x}}$ 映射到类别 y 。这意味着这种映射规则是不随坐标系的改变而改变的。因此可以通过线性变换，将训练集中的 $(\vec{\mathbf{x}}, y)$ 变换为 $(\vec{\mathbf{x}}', y)$ 从而产生了新的数据 $(\vec{\mathbf{x}}', y)$ 。

   对图像分类问题，数据集增强特别有效。数据集增强也可以应用于语音识别任务。

2. 常见的图片数据集增强方法：
   - 将训练图像沿着每个方向平移几个像素产生新的图像。
   - 对训练图像进行旋转、翻转或者缩放。
   - 对训练图像进行随机裁剪。

     > 实际上，随机裁剪图像的操作也可以被认为是预处理步骤，而不是数据集增强。

   - 对训练图像进行颜色抖动：调整饱和度、调整亮度、调整对比度、调整锐度。
     - 对比度：图像画面的明暗反差程度。对比度越高，则图片亮的地方更亮，暗的地方越暗。
     - 亮度：图像的明暗程度。亮度越高，则图像整体越亮。
     - 饱和度：图像颜色种类的多少。饱和度越高，则图像的颜色种类越多，图像越鲜艳。
     - 锐度：图像的边缘轮廓的锐利程度。锐度越高，则图像的边缘越清晰。
3. 在使用线性变换执行数据集增强时需要注意：
   - 某些线性变换会改变正确的类别。

     如：字符识别任务中， `b/d` 以及 `6/9` 的图像， 不能执行水平翻转变换和旋转 180 度变换。

- 某些线性变换难以执行。

  如：平面外的绕轴旋转（类似于翻页）难以通过简单的几何运算在输入图片上实现。

3.2 输入噪声注入

1. 在神经网络的输入层注入噪声也可以视作数据增强的一种形式。如：在图像识别任务中，对训练图像注入高斯噪声。

   事实上输入噪声注入也可以用于无监督学习，如：降噪自动编码器。

2. 通常一个训练好的神经网络对噪声鲁棒性较差，改善其噪声鲁棒性的常用方法是：简单地将随机噪声施加到输入上，再进行训练。

   - `Poole et al.(2014)` 表明：当仔细调整噪声的幅度之后，该方法非常高效。
   - 噪声被添加到每层隐单元的输入（而不仅仅是整个网络的输入）也是可行的，这被视为在多个抽象层上进行数据集增强。

     本章后面的 `dropout` 正则化策略可以被看作是通过对隐单元的输入乘上噪声。

四、噪声鲁棒性

1. 有三种添加噪声的策略：输入噪声注入、权重噪声注入、输出噪声注入。

4.1 输入噪声注入

1. 输入噪声注入：将噪声作用于输入的数据集，这也是前文介绍的一种数据集增强方法。
2. 对于某些模型，在输入上注入方差极小的噪音等价于对权重施加参数范数正则化（`Bishop,1995a,b`）。

   但是输入噪声注入远比简单地收缩参数强大，尤其是噪声被添加到隐单元的输入上时。

4.2 权重噪声注入

1. 权重噪声注入：将噪音作用于权重。这项技术主要用于循环神经网络。
2. 权重噪声注入可以解释为：将权重视作不确定的随机变量（拥有某个概率分布），向权重注入噪声是对该随机变量采样得到的一个随机值。
3. 在某些假设下，权重噪声注入等价于传统的参数正则化形式。
4. 假设有一个 l 层的标准的深度前馈神经网络，我们将噪声注入到该网络的权重。

   假设 $p_{model}(y\mid\vec{\mathbf{x}};\vec{\theta})=\mathcal{N}(y;f(\vec{\mathbf{x}};\vec{\theta}),\mathbf{I})$，则有：

   $$J(\vec{\theta})=-\mathbb{E}_{\vec{\mathbf{x}},y\sim\hat{p}_{data}}\log p_{model}(y\mid\vec{\mathbf{x}};\vec{\theta})=\frac{1}{2}\mathbb{E}_{\vec{\mathbf{x}},y\sim\hat{p}_{data}}||y-f(\vec{\mathbf{x}};\vec{\theta})||^2+\text{const}$$

   常数项包含了高斯分布的方差（与 $\vec{\theta}$ 无关）。

   于是目标函数重写为：$J(\vec{\theta})=\frac{1}{2}\mathbb{E}_{\vec{\mathbf{x}},y\sim\hat{p}_{data}}||y-f(\vec{\mathbf{x}};\vec{\theta})||^2$ 。

5. 假设每个权重添加一个随机扰动 $\epsilon_{\mathbf{W}}\sim\mathcal{N}(\epsilon;\vec{\mathbf{0}},\eta\mathbf{I})$，它是一个均值为0、方差为 η 的标准正态分布。

   假设添加扰动之后的模型为 $\tilde{p}_{model}(y\mid\vec{\mathbf{x}};\vec{\theta},\epsilon_{\mathbf{W}})=\mathcal{N}(y;\tilde{f}(\vec{\mathbf{x}};\vec{\theta},\epsilon_{\mathbf{W}}),\mathbf{I})$。

   假设有 $\mathbb{E}_{p(\epsilon_{\mathbf{W}})}\tilde{f}(\vec{\mathbf{x}};\vec{\theta},\epsilon_{\mathbf{W}})=f(\vec{\mathbf{x}};\vec{\theta})$，即：模型对于增加扰动之后的期望等于原来的模型。

   于是：

$$\begin{aligned}\tilde{J}(\vec{\theta}) &= \frac{1}{2}\mathbb{E}_{p(\vec{\mathbf{x}},y,\epsilon_{\mathbf{W}})}||y-\tilde{f}(\vec{\mathbf{x}};\vec{\theta},\epsilon_{\mathbf{W}})||^2 = \mathbb{E}_{p(\epsilon_{\mathbf{W}})}\left[\frac{1}{2}\mathbb{E}_{\vec{\mathbf{x}},y\sim\hat{p}_{data}}||y-\tilde{f}(\vec{\mathbf{x}};\vec{\theta},\epsilon_{\mathbf{W}})||^2\right]\\ &= \frac{1}{2}\mathbb{E}_{p(\epsilon_{\mathbf{W}})}\mathbb{E}_{\vec{\mathbf{x}},y\sim\hat{p}_{data}}||y-f(\vec{\mathbf{x}};\vec{\theta})+f(\vec{\mathbf{x}};\vec{\theta})-\tilde{f}(\vec{\mathbf{x}};\vec{\theta},\epsilon_{\mathbf{W}})||^2\\ &= \frac{1}{2}\mathbb{E}_{p(\epsilon_{\mathbf{W}})}\mathbb{E}_{\vec{\mathbf{x}},y\sim\hat{p}_{data}}\Big[(y-f(\vec{\mathbf{x}};\vec{\theta}))^2+(f(\vec{\mathbf{x}};\vec{\theta})-\tilde{f}(\vec{\mathbf{x}};\vec{\theta},\epsilon_{\mathbf{W}}))^2+\\ &\qquad 2(y-f(\vec{\mathbf{x}};\vec{\theta}))(f(\vec{\mathbf{x}};\vec{\theta})-\tilde{f}(\vec{\mathbf{x}};\vec{\theta},\epsilon_{\mathbf{W}}))\Big]\\ &= \frac{1}{2}\mathbb{E}_{p(\epsilon_{\mathbf{W}})}\mathbb{E}_{\vec{\mathbf{x}},y\sim\hat{p}_{data}}(y-f(\vec{\mathbf{x}};\vec{\theta}))^2+\frac{1}{2}\mathbb{E}_{p(\epsilon_{\mathbf{W}})}\mathbb{E}_{\vec{\mathbf{x}},y\sim\hat{p}_{data}}(f(\vec{\mathbf{x}};\vec{\theta})-\tilde{f}(\vec{\mathbf{x}};\vec{\theta},\epsilon_{\mathbf{W}}))^2\\ &\quad+\mathbb{E}_{\vec{\mathbf{x}},y\sim\hat{p}_{data}}(y-f(\vec{\mathbf{x}};\vec{\theta}))\mathbb{E}_{p(\epsilon_{\mathbf{W}})}(f(\vec{\mathbf{x}};\vec{\theta})-\tilde{f}(\vec{\mathbf{x}};\vec{\theta},\epsilon_{\mathbf{W}}))\end{aligned}$$

根据：

$$\frac{1}{2}\mathbb{E}_{p(\epsilon_{\mathbf{W}})}\mathbb{E}_{\vec{\mathbf{x}},y\sim\hat{p}_{data}}(y-f(\vec{\mathbf{x}};\vec{\theta}))^2 = \frac{1}{2}\mathbb{E}_{\vec{\mathbf{x}},y\sim\hat{p}_{data}}(y-f(\vec{\mathbf{x}};\vec{\theta}))^2 = J(\vec{\theta})$$

$$\mathbb{E}_{p(\epsilon_{\mathbf{W}})}(f(\vec{\mathbf{x}};\vec{\theta})-\tilde{f}(\vec{\mathbf{x}};\vec{\theta},\epsilon_{\mathbf{W}})) = f(\vec{\mathbf{x}};\vec{\theta})-\mathbb{E}_{p(\epsilon_{\mathbf{W}})}\tilde{f}(\vec{\mathbf{x}};\vec{\theta},\epsilon_{\mathbf{W}}) = 0$$

于是有：

$$\tilde{J}(\vec{\theta}) = J(\vec{\theta})+\frac{1}{2}\mathbb{E}_{p(\epsilon_{\mathbf{W}})}\mathbb{E}_{\vec{\mathbf{x}},y\sim\hat{p}_{data}}(f(\vec{\mathbf{x}};\vec{\theta})-\tilde{f}(\vec{\mathbf{x}};\vec{\theta},\epsilon_{\mathbf{W}}))^2+0$$

6. 将 $\tilde{f}$ 在 $\mathbf{W}$ 处泰勒展开，有：$\tilde{f}(\vec{\mathbf{x}};\vec{\theta},\epsilon_{\mathbf{W}}) = f(\vec{\mathbf{x}};\vec{\theta})+\nabla_{\mathbf{W}}f(\vec{\mathbf{x}};\vec{\theta})^T\epsilon_{\mathbf{W}}$ 。则有：

   $$\mathbb{E}_{p(\epsilon_{\mathbf{W}})}[(\tilde{f}(\vec{\mathbf{x}};\vec{\theta},\epsilon_{\mathbf{W}})-f(\vec{\mathbf{x}};\vec{\theta}))^2] = \mathbb{E}_{p(\epsilon_{\mathbf{W}})}(\nabla_{\mathbf{W}}f(\vec{\mathbf{x}};\vec{\theta})^T\epsilon_{\mathbf{W}})^2 = ||\nabla_{\mathbf{W}}f(\vec{\mathbf{x}};\vec{\theta})||^2\eta$$

   于是有：$\tilde{J}(\vec{\theta}) = J(\vec{\theta})+\frac{\eta}{2}\mathbb{E}_{\vec{\mathbf{x}},y\sim\hat{p}_{data}}||\nabla_{\mathbf{W}}f(\vec{\mathbf{x}};\vec{\theta})||^2$ 。

   这说明：权重噪声注入的代价函数等于非权重噪声注入的代价函数加上一个参数正则化项。

   - 该正则化项就是 $\frac{\eta}{2}\mathbb{E}_{\vec{\mathbf{x}},y\sim\hat{p}_{data}}||\nabla_{\mathbf{W}}f(\vec{\mathbf{x}};\vec{\theta})||^2$，其中 η 为噪声的方差。

     噪声方差越大，则正则化项越大。

   - 该形式的正则化将鼓励参数进入对小扰动不敏感的区域。即：找到的点不仅是极小点，还是由平坦区域包围的极小点。

     平坦区域意味着梯度很小，意味着对小扰动不敏感。

7. 如果是简单的线性回归，即 $f(\vec{\mathbf{x}};\vec{\theta}) = \vec{\mathbf{w}}^T\vec{\mathbf{x}}+b$，则权重噪声注入等价的参数正则化项退化为 $\frac{\eta}{2}\mathbb{E}_{\vec{\mathbf{x}},y\sim\hat{p}_{data}}||\vec{\mathbf{x}}||^2$ 。

   该正则化项与模型的参数无关，因此对 $\tilde{J}(\vec{\theta})$ 关于 $\mathbf{W}$ 的梯度没有贡献，因此目标函数可以重写为：$\tilde{J}(\vec{\theta}) = J(\vec{\theta})$ 。

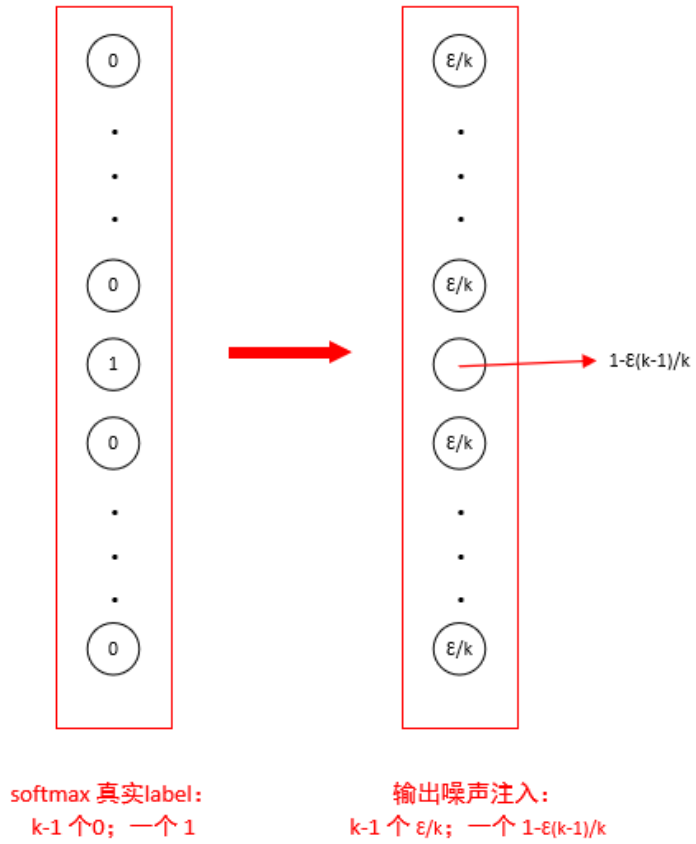
4.3 输出噪声注入

1. 有些数据集存在少量的 y 标签是错误的，此时通过最大似然准则来最大化 $\sum\log p(y\mid\vec{\mathbf{x}})$ 是不正确的。

   输出噪声注入显式地对标签上的噪音进行建模：假设某个很小的常数 ϵ， 标签 y 是正确的概率为 $1-\epsilon$ 、是错误的概率为 ϵ 。

2. 基于 k 个输出的 `softmax` 单元的标签平滑正则化 `label smoothing regularize`：将真实的标签从 $\{0,1\}$ 替换为 $\{\frac{\epsilon}{k},1-\frac{k-1}{k}\epsilon\}$ 。

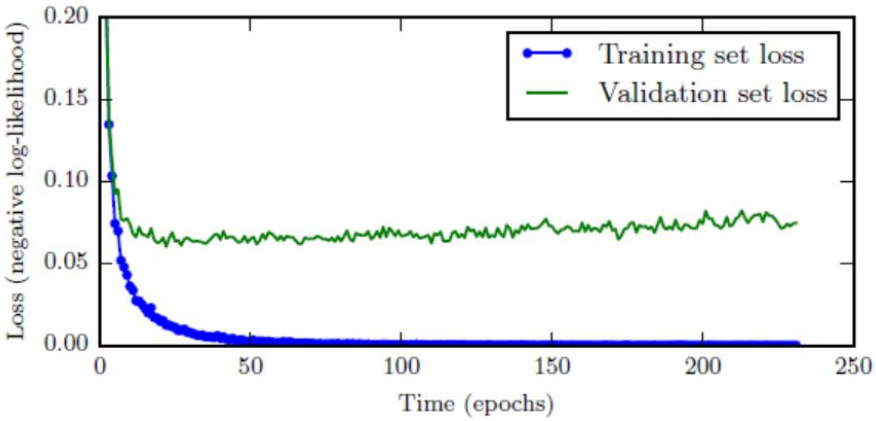
原始的标签：$k-1$ 个为 `0`，一个为 `1`。注入噪声之后的标签：$k-1$ 个为 $\frac{\epsilon}{k}$，一个为 $1-\frac{k-1}{k}\epsilon$。



五、早停

1. 当训练一个容量较大的模型时会经常发现：训练误差逐渐降低，但是验证误差先下降后上升。

   当验证误差没有进一步改善时，算法就提前终止。这种策略被称作早停 `early stopping`。



2. 早停是深度学习中最常用的正则化形式，因为它简单、有效。
3. 当训练终止时，返回的不是最新的模型参数，而是验证误差最小的模型参数，因此需要频繁存储模型参数。

5.1 早停算法

1. 早停算法：
   - 输入：

        - 当前验证集的误差非最小值的次数 p
        - 验证集验证的间隔 n
        - 初始参数 $\vec{\theta}_0$
    - 输出：
        - 最佳参数 $\vec{\theta}^*$
        - 获得最佳参数时迭代的步数 i^*
    - 算法步骤：
        - 初始化：
            - 参数变量 $\vec{\theta}=\vec{\theta}_0$
            - 迭代步数变量 $i=0$
            - 验证次数变量 $j=0$
            - 验证集的最小误差 $v=\infty$
            - 最佳参数 $\vec{\theta}^*=\vec{\theta}$
            - 最佳迭代步数 $i^*=i$
        - 循环，循环条件为： $j<p$:
            - 学习模型 n 步（每隔 n 步验证一次）
            - 更新 $i=i+n$
            - 记录最新的验证集误差 $v'=\text{ValidationSetError}(\vec{\theta})$
            - 如果 $v'<v$，则： $j=0,\vec{\theta}^*=\vec{\theta},i^*=i,v=v'$ 。

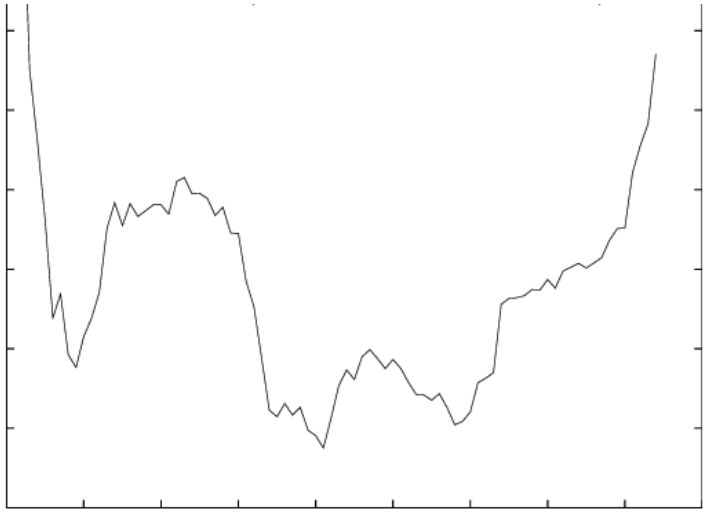
              如果 $v'\geq v$，则： $j=j+1$ 。

              > 若当前验证集误差是最小的，则 j 清零。这意味着必须连续 p 次 $v'<v$ ，才说明算法到达终止条件。

2. 可以认为早停是一个非常高效的超参数选择算法：训练步数是一个超参数，该超参数在验证误差上具有 `U` 形曲线。
    - 早停策略通过控制训练步数来控制模型的有效容量 `capacity` 。
    - 早停策略只需要跑一轮训练就能够得到很多的超参数（即：训练步数）及其对应的验证误差。
3. 早停策略的代价有两个：
    - 需要在训练期间定期评估验证集。
        - 可以通过并行的执行训练和验证来加速这一过程。
        - 也可以选取一个较小的验证集、或者不那么频繁地评估验证集来减小评估代价。
    - 需要保持最佳的参数的副本。

      这种代价一般可以忽略不计。
4. 早停是正则化的一种非常不起眼的形式，其优点有：
    - 它几乎不需要干涉基本的训练过程，适合任何模型。
    - 可以单独使用，或者与其他的正则化策略相结合。
    - 早停不仅有正则化的好处，还有降低计算成本的好处。
5. 以泛化误差的偏差方差分解角度来看，早停试图同时解决偏差和方差，其结果很可能导致得到的模型并不是一个最优的模型。

   如下图中所示的验证误差曲线。因为提前停止了训练，所以使得代价函数的值可能不够小，即：偏差可能还可以继续降低。方差（即：验证误差与训练误差之间的 `gap` ）虽然处于上升趋势，但是它们叠加的结果可能导致验证误差呈现波动走势。



5.2 二次训练

1. 早停需要验证集，这意味着某些样本不能用于模型的训练过程，这会造成数据的浪费。

   为了更好地利用验证集的样本，可以在早停之后进行额外的训练。在第二轮额外的训练中，所有的训练数据都被包括在内（包括验证集）。

   有两个基本的策略可以用于第二轮训练过程：

   - 保留迭代步：再次初始化模型，然后使用所有数据再次训练。此时使用第一轮早停确定的最佳步数作为第二轮的迭代步数。

     该策略重新训练模型，成本较高，但是效果较好。

   - 保留参数：保持从第一轮训练中获得的参数，然后使用全部的数据继续训练。此时观察原始验证集的损失函数，直到它低于第一轮停止时的原始训练集的损失函数值。

     > 根据早停策略，第一轮结束时原始验证集的损失函数值是较大的

     该策略避免了重新训练模型的高成本，但是表现一般。

     这是因为一旦将 $(\mathbf{X}^{(valid)},\mathbf{y}^{(valid)})$ 合并到训练集，则对它们评估的结果就是训练误差（而不再是验证误差）。新的训练误差小于原来的验证误差，并不能说明模型的泛化能力得到了提升。

2. 保留迭代步二次训练算法：

   - 输入：
     - 训练集 $\mathbf{X}^{(train)}$ ， $\mathbf{y}^{(train)}$
   - 步骤：
     - 将 $\mathbf{X}^{(train)}$ 和 $\mathbf{y}^{(train)}$ 分割为 $(\mathbf{X}^{(subtrain)},\mathbf{y}^{(subtrain)})$ 和 $(\mathbf{X}^{(valid)},\mathbf{y}^{(valid)})$
     - 随机选择参数的初始化值 $\vec{\theta}_0$，将 $(\mathbf{X}^{(subtrain)},\mathbf{y}^{(subtrain)})$ 作为训练集， 将 $(\mathbf{X}^{(valid)},\mathbf{y}^{(valid)})$ 作为验证集，运行早停算法，返回最佳训练步数 i^*
     - 再次选择参数的另一个初始化值 $\vec{\theta}_1$，在 $(\mathbf{X}^{(train)},\mathbf{y}^{(train)})$ 上再次训练 i^* 步

3. 保留参数二次训练算法：

   - 输入：
     - 训练集 $\mathbf{X}^{(train)}$ ， $\mathbf{y}^{(train)}$
   - 步骤：
     - 将 $\mathbf{X}^{(train)}$ 和 $\mathbf{y}^{(train)}$ 分割为 $(\mathbf{X}^{(subtrain)},\mathbf{y}^{(subtrain)})$ 和 $(\mathbf{X}^{(valid)},\mathbf{y}^{(valid)})$

- 随机选择参数的初始化值 $\vec{\theta}_0$，将 $(\mathbf{X}^{(subtrain)},\mathbf{y}^{(subtrain)})$ 作为训练集， 将 $(\mathbf{X}^{(valid)},\mathbf{y}^{(valid)})$ 作为验证集，运行早停算法，返回算法停止时的目标函数的值 $\epsilon=J(\vec{\theta},\mathbf{X}^{(valid)},\mathbf{y}^{(valid)})$
- 迭代： $while\ J(\vec{\theta},\mathbf{X}^{(valid)},\mathbf{y}^{(valid)})>\epsilon\quad do$
  - 在 $\mathbf{X}^{(train)},\mathbf{y}^{(train)}$ 上训练 n 步 (每隔 n 步检查一次，为了降低评估代价)

5.3 早停与 L2 正则化

1. 早停将优化过程的参数空间限制在初始参数值 $\vec{\theta}_0$ 的一个小的邻域内。
2. 假设参数 $\vec{\theta}=\vec{\mathbf{w}}$ 。令 $\vec{\mathbf{w}}^*=\arg\min_{\vec{\mathbf{w}}}J(\vec{\mathbf{w}})$，它就是无正则化项时使得目标函数最小的权重向量。

   和 L_2 正则化中的推导相同，有：$\nabla_{\vec{\mathbf{w}}}\hat{J}(\vec{\mathbf{w}})=\mathbf{H}(\vec{\mathbf{w}}-\vec{\mathbf{w}}^*),\quad \vec{\mathbf{w}}\in\mathbb{N}(\vec{\mathbf{w}}^*)$ 。

   $\mathbf{H}$ 为 J 在 $\vec{\mathbf{w}}^*$ 处的海森矩阵， 其中 $\vec{\mathbf{w}}$ 在 $\vec{\mathbf{w}}^*$ 的一个邻域内。

   - 根据梯度下降法，参数的迭代过程为：

     $$\vec{\mathbf{w}}^{(\tau)}=\vec{\mathbf{w}}^{(\tau-1)}-\epsilon\nabla_{\vec{\mathbf{w}}}\hat{J}(\vec{\mathbf{w}}^{(\tau-1)})=\vec{\mathbf{w}}^{(\tau-1)}-\epsilon\mathbf{H}(\vec{\mathbf{w}}^{(\tau-1)}-\vec{\mathbf{w}}^*)$$
     $$\rightarrow \vec{\mathbf{w}}^{(\tau)}-\vec{\mathbf{w}}^*=(\mathbf{I}-\epsilon\mathbf{H})(\vec{\mathbf{w}}^{(\tau-1)}-\vec{\mathbf{w}}^*)$$

     注意：这里并没有加入任何正则化项，而是使用 J

   - 将 $\mathbf{H}$ 进行特征分解： $\mathbf{H}=\mathbf{Q}\Lambda\mathbf{Q}^T$ ，其中 Λ 为对角矩阵， $\mathbf{Q}$ 为特征向量的一组标准正交基。则有：

     $$\vec{\mathbf{w}}^{(\tau)}-\vec{\mathbf{w}}^*=\mathbf{Q}(\mathbf{I}-\epsilon\Lambda)\mathbf{Q}^T(\vec{\mathbf{w}}^{(\tau-1)}-\vec{\mathbf{w}}^*)$$
     $$\rightarrow \mathbf{Q}^T(\vec{\mathbf{w}}^{(\tau)}-\vec{\mathbf{w}}^*)=(\mathbf{I}-\epsilon\Lambda)\mathbf{Q}^T(\vec{\mathbf{w}}^{(\tau-1)}-\vec{\mathbf{w}}^*)$$

   - 令参数向量的初始值为原点： $\vec{\mathbf{w}}^{(0)}=\vec{\mathbf{0}}$ ，并且选择 ϵ 使得 $|1-\epsilon\lambda_i|<1$（λ_i 为 $\mathbf{H}$ 的特征值）。则经过 τ 次参数更新之后：$\mathbf{Q}^T\tilde{\mathbf{w}}^{(\tau)}=[\mathbf{I}-(\mathbf{I}-\epsilon\Lambda)^\tau]\mathbf{Q}^T\vec{\mathbf{w}}^*$ 。
3. 根据 L_2 正则化项中的推导结果，有：

   $$\tilde{\vec{\mathbf{w}}}^*=\mathbf{Q}(\mathbf{\Lambda}+\alpha\mathbf{I})^{-1}\mathbf{\Lambda}\mathbf{Q}^T\vec{\mathbf{w}}^*$$
   $$\rightarrow \mathbf{Q}^T\tilde{\vec{\mathbf{w}}}^*=(\mathbf{\Lambda}+\alpha\mathbf{I})^{-1}\mathbf{\Lambda}\mathbf{Q}^T\vec{\mathbf{w}}^*=[\mathbf{I}-(\Lambda+\alpha\mathbf{I})^{-1}\alpha]\mathbf{Q}^T\vec{\mathbf{w}}^*$$

   > 通过直接写出逆矩阵的形式可以证明等式

   如果超参数 ϵ,α,τ 满足：$(\mathbf{I}-\epsilon\Lambda)^\tau=(\Lambda+\alpha\mathbf{I})^{-1}\alpha$ ，则 L_2 正则化等价于早停。
4. 为了求得三个超参数满足的条件，求解：

   $$\begin{bmatrix}(1-\epsilon\lambda_1)^\tau & 0 & \cdots & 0\\ 0 & (1-\epsilon\lambda_2)^\tau & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & (1-\epsilon\lambda_n)^\tau\end{bmatrix}=\begin{bmatrix}\frac{\alpha}{\lambda_1+\alpha} & 0 & \cdots & 0\\ 0 & \frac{\alpha}{\lambda_2+\alpha} & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & \frac{\alpha}{\lambda_n+\alpha}\end{bmatrix}$$

   则有：$(1-\epsilon\lambda_i)^\tau=\frac{\alpha}{\lambda_i+\alpha}$ 。

   两边取对数，然后使用 $\log(1+x)$ 的级数展开有（假设 $0<\epsilon\lambda_i\ll 1$，且 $\frac{\lambda_i}{\alpha}\ll 1$）： $\tau\epsilon\lambda_i=\frac{\lambda_i}{\alpha}$ 。

   > 由于 $\vec{\mathbf{w}}^*$ 是 $J(\vec{\mathbf{w}})$ 的最小点，因此海森矩阵 $\mathbf{H}$ 是半正定的，因此其特征值 $\lambda_i\geq 0$

   则有：$\tau\epsilon=\frac{1}{\alpha}$ 。即： $\tau\epsilon$ 的倒数与 L_2 正则化系数的作用类似。

   在给定 ϵ 的条件下：

- τ 越小，则正则化系数越大。这与以下事实吻合： $\vec{\mathbf{w}}$ 从原点出发开始迭代，如果 τ 越小，则 $\vec{\mathbf{w}}$ 越靠近原点。
- τ 越大，则正则化系数越小。这与以下事实吻合：此时 $\vec{\mathbf{w}}$ 迭代足够多步之后会逐渐远离原点。

5. 假设用学习率 ϵ 进行了 τ 个优化步骤（对应于 τ 个训练迭代），$\epsilon\tau$ 可以视作模型的有效容量 `effective capacity` 的度量。

   假设梯度有界，则限制迭代的次数和学习率，会限制 $\vec{\theta}$ 从 $\vec{\theta}_0$ 能够到达的范围。$\epsilon\tau$ 的行为就像是 L_2 正则化项的系数的倒数。

6. 早停比 L_2 正则化更有优势：
   - 早停能够监控验证误差，从而自动在某个较好的参数解的位置终止。

     训练一次就相当于得到了多个超参数 α 的结果。
   - 采用 L_2 正则化需要多次训练，从而选择合适的超参数 α 的值。

     这种方式的计算成本太高。

六、参数相对约束

1. 通常对参数添加约束时，是固定于相对的某个点。如 L_2 正则化：将参数对于偏离零点来进行惩罚。

   如果需要表达两个模型之间参数的相对关系时，则使用参数的相对约束。

   假设模型 `A` 的参数为 $\vec{\mathbf{w}}^{(A)}$，模型 `B` 的参数为 $\vec{\mathbf{w}}^{(B)}$ 。如果两个模型非常相似，则给定下列形式的惩罚：

   $$\Omega(\vec{\mathbf{w}}^{(A)},\vec{\mathbf{w}}^{(B)})=||\vec{\mathbf{w}}^{(A)}-\vec{\mathbf{w}}^{(B)}||_2^2$$

   这里使用 L_2 惩罚，也可以使用其他形式的正则化形式。

   这种方法由 `Lasserre et al.(2006)` 提出，它使得一个有监督的分类模型的参数接近于另一个无监督的数据分布模型的参数。

2. 还有一种方案：强迫 $\vec{\mathbf{w}}$ 的某个子集相等，这称作参数共享 `parameter sharing`。

   如：要求 $w_i=w_{i+K}$ ，即参数子集为：$\{w_1,w_2,\cdots,w_{K-1}\}$ ，其剩余参数都与该子集相等。这就是卷积神经网络中使用的方案。

   参数共享的优点：能显著降低参数的数量、减少模型占用的内存。

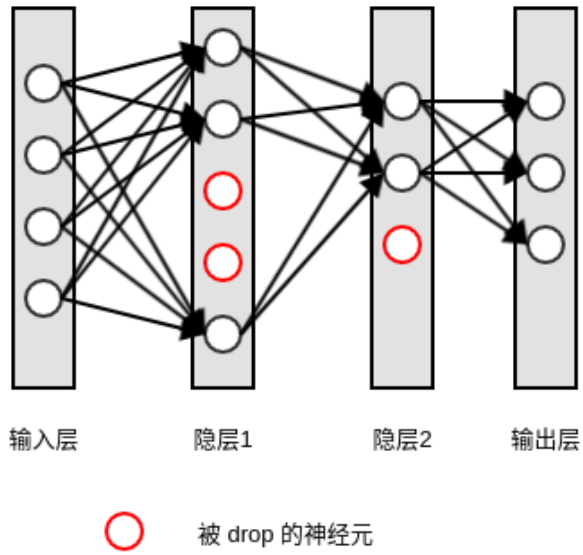
七、dropout

1. `dropout`：在前向传播过程中，对网络中的每个隐层，每个隐单元都以一定的概率 p_{drop} 被删除，最后得到一个规模更小的网络。在反向传播过程中，仅仅针对该小网络进行权重更新。
   - 所谓的删除，即指定该该隐单元的输出都为 0。

     一旦隐单元的权重为0，则该隐单元对后续神经元的影响均为 0 。
   - 输入层和输出层的神经元不会被删除，因为这两个层的神经元的数量是固定的。

     理论上可以对输入层应用 `dropout` ，使得可以有机会删除一个或者多个输入特征。但实际工程中，通常不会这么做。
   - 隐单元删除发生在一个训练样本的训练期间。
     - 不同的训练样本，其删除的隐单元的集合是不同的，因此裁剪得到的小网络是不同的。
     - 不同的训练样本，隐单元被删除的概率 p_{drop} 都是相同的。
     - 在不同 `batch` 之间的同一个训练样本，其删除的隐单元的集合也是不同的。

- 在不同的梯度更新周期，会从完整的网络中随机删除不同的神经元，因此裁剪得到的小网络是不同的。但是在这个过程中，隐单元被删除的概率是相同的。
- 可以指定某一个隐层或者某几个隐层执行`dropout`，而没有必要针对所有的隐层执行`dropout`。
- 可以对网络的每个隐单元指定不同的删除概率，但实际工程中，通常不会这么做。



2. 定义一个掩码向量 $\vec{\mu}$，它给出了哪些隐单元被保留哪些隐单元被删除：掩码为`0`的位置对应的隐单元被删除，掩码为`1`的位置对应的隐单元被保留。

   定义 $J(\vec{\theta},\vec{\mu})$ 为参数 $\vec{\theta}$ 和掩码 $\vec{\mu}$ 共同定义的模型代价，`dropout`的目标是最小化 $\mathbb{E}_{\vec{\mu}}J(\vec{\theta},\vec{\mu})$。

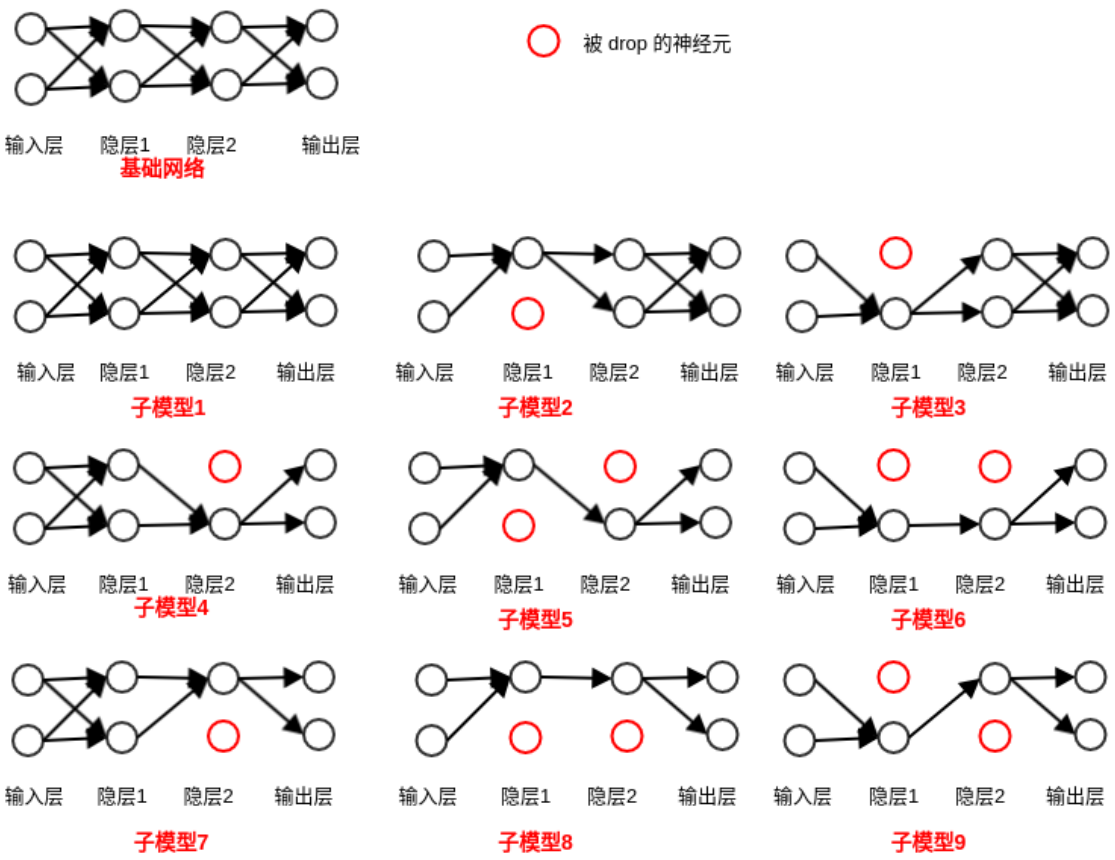
   - 这里采用期望，因为掩码向量 $\vec{\mu}$ 是一个随机向量，对于每个训练样本 $\vec{\mu}$ 都可能不同。
   - 因为掩码向量具有指数多个，因此期望包含了指数多项。实际应用中，可以通过抽样 $\vec{\mu}$ 来获得期望的无偏估计。

7.1 dropout 与 bagging

1. `dropout`可以视作集成了非常多的神经网络的`bagging`集成模型，这些网络包含了所有从基础网络中删除隐单元形成的子网络。

   `bagging`涉及训练多个模型，并且在每个测试样本上评估多个模型。当每个模型都是一个大型神经网络时，这种`bagging`计算量非常庞大，实际不可行。

   `dropout`提供了一种方便的`bagging`近似，它能够训练和评估指数级别的神经网络的集成。



2. `dropout` 训练与 `bagging` 训练不同：
   - `bagging` 中，假设所有的子模型都是独立的。

     `dropout` 中，所有的子模型是共享参数的，每个子模型继承了基础神经网络的不同子集。

     参数共享使得在有限的内存下训练指数数量的子模型变得可能。
   - `bagging` 中，每个子模型在其相应的训练集上训练到收敛。

     `dropout` 中，大部分子模型都没有显式的被训练（因为指数量级的神经网络不可能被训练完成）。

     我们只是对子模型的某些部分训练单个 `step` ，同时参数共享使得子模型的剩余部分能够有较好的参数值。

   除此之外，二者相同。比如每个子模型使用的训练集是原始训练集的一个有放回重复采样而来。

7.2 模型推断

1. `dropout` 仅仅用于神经网络的训练阶段，在网络的测试阶段并不会删除神经元，而是使用所有的神经元。因为在测试期间，不希望输出是随机的。

   如果在测试阶段使用 `dropout` ，则理论上需要运行多次 `dropout` 测试过程，然后对输出结果取加权平均（或者几何平均）。这种做法的结果与不采用 `dropout` 的测试结果相同，反而计算效率更低。

2. 在 `dropout` 情况下，每个子模型的输出为概率分布 $p(y \mid \vec{\mathbf{x}}, \vec{\mu})$ ，不同的掩码 $\vec{\mu}$ 就定义了不同的子模型。

   集成模型的输出由所有子模型输出的加权平均给出：

$$p_{ensemble}(y \mid \vec{\mathbf{x}}) = \sum_{\vec{\mu}} p(\vec{\mu}) p(y \mid \vec{\mathbf{x}}, \vec{\mu})$$

其中 $p(\vec{\mu})$ 是训练时 $\vec{\mu}$ 的概率分布。之所以不采用 $\frac{1}{k}$ 这种代数平均（`k` 为子模型数量），是因为 $\vec{\mu}$ 掩码生成的概率不一定是均匀的。

上式包含了指数量级的项，不可能计算出来。但是可以通过采样来近似推断，即对多个掩码对应的输出进行平均。通常 10-20 个掩码就足以获取不错的表现。

3. 除了子模型输出的加权平均之外，还可以采用几何平均：

$$\tilde{p}_{ensemble}(y \mid \vec{\mathbf{x}}) = \sqrt[2^d]{\prod_{\vec{\mu}} p(y \mid \vec{\mathbf{x}}, \vec{\mu})}$$

其中 d 为所有的可以被丢弃的单元的数量，其丢弃/保留组合有 2^d 种。

- 这里采用的是均匀分布的 $\vec{\mu}$（也可以采用非均匀分布），设丢弃和保留的概率都是 $\frac{1}{2}$ 。
- 多个概率分布的几何平均不一定是个概率分布，因此为了保证结果是一个概率分布，概率归一化为：

$$p_{ensemble}(y \mid \vec{\mathbf{x}}) = \frac{\tilde{p}_{ensemble}(y \mid \vec{\mathbf{x}})}{\sum_{y'} \tilde{p}_{ensemble}(y' \mid \vec{\mathbf{x}})}$$

其中要求：没有任何一个子模型给出所有事件的概率为都为 0 （否则会导致 $\tilde{p}_{ensemble}(y' \mid \vec{\mathbf{x}})$ 都为 0 ）。

4. 实际应用中，并不需要直接求解 $p_{ensemble}(y \mid \vec{\mathbf{x}})$ 。通常通过评估某个模型的 $p(y \mid \vec{\mathbf{x}})$ 来近似 $p_{ensemble}$ 。该模型这样产生：该模型与`dropout` 的基础模型具有相同的单元；该模型单元 i 输出的权重需要乘以保留该单元 i 的概率。

这种策略叫做权重比例推断法则。目前这种做法在深度非线性网络上工作良好，但是没有任何理论上的证明。

7.3 示例

1. 考虑`softmax` 单元。假设 $\vec{\mathbf{v}}$ 表示 n 个输入变量，则有：$p(y \mid \vec{\mathbf{v}}) = softmax(\mathbf{W}^T\vec{\mathbf{v}} + \vec{\mathbf{b}})_y$ 。其中 $y \in \{1, 2, \cdots, K\}$ 表示分类的类别， $softmax(\cdot)_y$ 表示输出的第 y 个分量。

假设给定掩码 $\vec{\mathbf{d}} = (d_1, d_2, \cdots, d_n)$ 。其中 $d_i \in \{0, 1\}$ ，为二值随机变量：若它取值为 `0` 则表示输入 v_i 被遗忘；若它取值为 `1` 则表示输入 v_i 被保留。且假设 $p(d_i = 0) = \frac{1}{2}$ 。

则：$p(y \mid \vec{\mathbf{v}}; \vec{\mathbf{d}}) = softmax(\mathbf{W}^T(\vec{\mathbf{v}} \odot \vec{\mathbf{d}}) + \vec{\mathbf{b}})_y$ 。其中 $\odot$ 表示逐元素的相乘。

2. `softmax` 单元的输出为：$p_{ensemble}(y \mid \vec{\mathbf{v}}) = \frac{\tilde{p}_{ensemble}(y|\vec{\mathbf{v}})}{\sum_{y'} \tilde{p}_{ensemble}(y'|\vec{\mathbf{v}})}$ 。其中：

$$\tilde{p}_{ensemble}(y \mid \vec{\mathbf{v}}) = \sqrt[2^n]{\prod_{\vec{\mathbf{d}} \in \{0,1\}^n} p(y \mid \vec{\mathbf{v}}; \vec{\mathbf{d}})}$$

化简上式：

$$\sqrt[2^n]{\prod_{\vec{\mathbf{d}} \in \{0,1\}^n} softmax(\mathbf{W}^T(\vec{\mathbf{v}} \odot \vec{\mathbf{d}}) + \vec{\mathbf{b}})_y} = \sqrt[2^n]{\prod_{\vec{\mathbf{d}} \in \{0,1\}^n} \frac{\exp(\mathbf{W}_{y,:} \cdot (\vec{\mathbf{v}} \odot \vec{\mathbf{d}}) + \vec{\mathbf{b}}_y)}{\sum_{y'} \exp(\mathbf{W}_{y',:} \cdot (\vec{\mathbf{v}} \odot \vec{\mathbf{d}}) + \vec{\mathbf{b}}_{y'})}}$$

其中 $\mathbf{W}_{y,:}$ 表示矩阵$\mathbf{W}$ 的第 y 列；$\cdot$ 表示向量点乘。

忽略对 y 不变的项，有：

$$\tilde{p}_{ensemble}(y \mid \vec{\mathbf{v}}) \propto \sqrt[2^n]{\prod_{\vec{\mathbf{d}} \in \{0,1\}^n} \exp(\mathbf{W}_{y,:}^T(\vec{\mathbf{v}} \odot \vec{\mathbf{d}}) + \vec{\mathbf{b}}_y)}$$

$$= \exp\left(\frac{1}{2^n} \sum_{\vec{\mathbf{d}} \in \{0,1\}^n} \mathbf{W}_{y,:}^T(\vec{\mathbf{v}} \odot \vec{\mathbf{d}}) + \vec{\mathbf{b}}_y\right)$$

3. 考虑 $\frac{1}{2^n}\sum_{\vec{\mathbf{d}} \in \{0,1\}^n} \mathbf{W}_{y,:}^T(\vec{\mathbf{v}} \odot \vec{\mathbf{d}})$，它就是 $\mathbb{E}_{d_i \sim uniform}[\mathbf{W}_{y,:}^T \vec{\mathbf{v}}] = \frac{1}{2}\mathbf{W}_{y,:}^T \vec{\mathbf{v}}$。则有：

$$\tilde{p}_{ensemble}(y \mid \vec{\mathbf{v}}) \propto \exp\left(\frac{1}{2^n} \sum_{\vec{\mathbf{d}} \in \{0,1\}^n} \mathbf{W}_{y,:}^T(\vec{\mathbf{v}} \odot \vec{\mathbf{d}}) + \vec{\mathbf{b}}_y\right) = \exp\left(\frac{1}{2}\mathbf{W}_{y,:}\vec{\mathbf{v}} + \vec{\mathbf{b}}_y\right)$$

因此，`softmax`单元采用`dropout`的结果就是权重为 $\frac{1}{2}\mathbf{W}$ 的`softmax`单元。

4. 权重比例推断法在`softmax`以外的情况也是精确的，但是它对于非线性的深度模型仅仅是一个近似。

虽然尚未有理论上的分析，但是在实践中往往效果非常好。

7.4 性质

1. `dropout`不限制使用的模型或者训练过程，另外其计算非常方便。
   - 训练过程中使用`dropout`时，产生 n 个随机二进制数与每个权重相乘。

     `dropout`对于每个样本每次更新只需要 $O(n)$ 的计算复杂度。根据需要也可能需要 $O(n)$ 的存储空间来保存这些二进制数（直到反向传播阶段）。
   - 在预测过程中，计算代价与不使用`dropout`是一样的。
2. `dropout`的缺点是：代价函数 J 不再被明确定义，因为每次迭代时都会随机移除一部分隐单元。
3. 虽然`dropout`在模型训练的每一步上所做的工作是微不足道的（仅仅是随机地保留某些单元），但是在完整的训练结果上，该策略的效果非常显著。
4. `dropout`不仅是一种高效的近似`bagging`的方法，它还是共享隐单元的集成模型，它通过参数共享来实现`bagging`。
   - 这种参数共享策略不一定必须是`包括`和`排除`（也就是二进制的 0 和1）。原则上任何一种随机的修改都可以接受。
   - 随机向量 $\vec{\mu}$ 可以具有任何形式，而不一定要求它是掩码向量（取值为 0/1）。

     `Srivastava et al.(2014)`表明：将权重乘以 $\vec{\mu} \sim \mathcal{N}(\vec{\mathbf{1}}, \mathbf{I})$ 要比基于二值掩码`dropout`表现的更好。

     由于此时 $\mathbb{E}[\vec{\mu}] = 1$，则网络自动实现了集成方法的近似推断，因此不需要应用权重比例推断。
5. 网络中的不同隐层的删除概率可以是变化的：
   - 对于某些层，如果隐单元数量较少，过拟合的程度没有那么严重，则该层的 p_{drop} 值可以小一些。
   - 对于一些层，如果隐单元数量较多，过拟合较严重，则该层 p_{drop} 的值可以较大。
   - 对于一些层，如果不关心其过拟合问题，则该层的 p_{drop} 可以为`0`。

   > 对一个隐层设定一个删除概率`p`，意思是对该层中所有隐单元设定同一个删除概率`p`
6. 使用`dropout`有两点注意：
   - 在训练期间如果对某个单元的输入神经元执行了`dropout`，则推断期间该单元的输出要进行调整。

     假设该单元的输出为 h，则需要调整为 $\frac{h}{1-p_{drop}}$，从而保证不影响该单元的输出值。

- 在测试期间，不必使用`dropout`。

7.5 dropout 与正则化

1. 由于针对每个训练样本，训练的都是一个规模极小的网络，因此`dropout`也是一种正则化策略，它减少了模型的有效容量`effective capacity`。

   为了抵抗模型有效容量的降低，必须增大模型的规模，此时需要更多的训练迭代次数。

   - 对于非常大的数据集，`dropout`正则化带来的泛化误差减少得很小。此时使用`dropout`带来的更大模型的计算代价可能超过了正则化带来的好处。
   - 对于极小的数据集，`dropout`不会很有效。
2. 除非模型发生过拟合，否则不要轻易使用`dropout`。

   因为计算机视觉领域通常缺少足够的数据，所以非常容易发生过拟合。因此`dropout`在计算机视觉领域大量应用。
3. `Srivastava et al.(2014)`显示：`dropout`比其他标准的、计算开销较小的正则化项（如 L_2 正则化）更加有效。

   `dropout`也可以与其他形式的正则化合并，进一步提升模型的泛化能力。
4. `dropout`训练时随机丢弃隐单元：
   - 这种随机性并不是产生正则化效果的必要条件，它仅是近似所有可能的子模型总和的一种方法。
   - 这种随机性也不是产生正则化效果的充分条件。

     `Warde-Farley et al.(2014)`使用`dropout boosting`策略（类似于`boosting`）设计了一个对照试验，结果表明`dropout boosting`几乎没有正则化的效果。
5. `dropout`正则化效果主要来源于施加到隐单元的掩码噪声。这可以看作是对输入信息自适应破坏的一种形式。
   - `dropout`是对输入内容的某个结构进行了修改，而不是传统的噪声注入。
   - `dropout`的噪声是乘性的，而传统的噪声注入是加性的。

八、对抗训练

1. 神经网络在预测过程中，可以故意人工构造这样的一种样本：对输入点 $\vec{\mathbf{x}}$，我们搜寻它的附近的一个人眼看起来没有区别、网络预测结果差异很大的样本 $\vec{\mathbf{x}}'$。
   - 输入点 $\vec{\mathbf{x}}$ 来自于训练集，称作原始样本；而 $\vec{\mathbf{x}}'$ 是人工构造的，称作对抗样本`adversarial example`。
   - 我们可以通过将对抗样本加入训练集来改善模型的泛化能力，这称作对抗训练`adversarial training`。其中：对抗样本$\vec{\mathbf{x}}'$的真实标签要求和 $\vec{\mathbf{x}}$ 相同，但是其预测结果要求和 $\vec{\mathbf{x}}$ 不同。
2. `Goodfellow et al(2014b)`表明：存在这些对抗样本的主要原因是高度线性。

   神经网络主要基于线性块构建的，因此模型是高度线性的。对于一个线性函数，如果它是高维的，那么其函数值可能对扰动非常敏感。
3. 对抗训练通过鼓励网络在训练数据附近的局部区域保持稳定来限制函数对输入扰动的高度敏感性。它可以视作一种先验知识：模型是局部稳定的。
4. 对抗训练也可以用于实现半监督学习：
   - 首先根据规模小的、监督样本中学习模型。

- 然后根据模型，预测大量的、未监督的样本。假设未监督样本 $\vec{\mathbf{x}}$ 预测的标签为 $\hat{y}$（虽然其真实标签可能不是 $\hat{y}$，但是如果模型质量非常好，则真实标签是 $\hat{y}$ 的概率非常大）。
- 然后在 $\vec{\mathbf{x}}$ 的一个很小的领域内寻找它的对抗样本 $\vec{\mathbf{x}}'$。由于不是采用真实的标签，而是模型预测的标签 $\hat{y}$，因此 $\vec{\mathbf{x}}'$ 也称作虚拟对抗样本。
- 重新训练模型，使得模型在 $\vec{\mathbf{x}},\vec{\mathbf{x}}'$ 上的预测标签分类都相同。

这种策略鼓励模型沿着未监督样本所在流形上的任意微小变化都是稳定的。其基本假设是：不同的分类通常位于分离的流形上，并且小的扰动不能从一类的流形跳到另一个流形上。

九、正切传播算法

1. 切面距离算法是最近邻算法的一种，其中的度量使用的不是通用的欧几里得距离，而是流形距离。
2. 假设分类样本有这样的性质：在同一个流形上的样本分类相同，不同流形上的样本分类不同。则分类器会有这样的特点：对于样本的局部变化不敏感。即：样本在流形上的移动时，分类器的输出不变。
3. 假设样本 $\vec{\mathbf{x}}_1$ 的流形为 M_1，样本 $\vec{\mathbf{x}}_2$ 的流形为 M_2。则这两个点的距离定义为：流形 M_1 到流形 M_2 的距离。

   问题是计算两个流形的距离代价很高，因为它要求解两个集合的最近点。一个替代方案：用 $\vec{\mathbf{x}}_i$ 点的切平面来近似 M_i， 然后测量两个切平面的距离（或者一个切平面到一个点的距离）。
4. 受切面距离算法的启发，正切传播算法训练神经网络分类器的策略是：使得神经网络的输出 $f(\vec{\mathbf{x}})$ 沿着同样类别样本所在的流形上移动时，保持局部不变性。
   - 局部不变性：要求输出 $f(\vec{\mathbf{x}})$ 沿着流形的方向变化很小。即： $\nabla_{\vec{\mathbf{x}}}f(\vec{\mathbf{x}})$ 在流形的切向的分量为零。
   - 对于样本点 $\vec{\mathbf{x}}_i$，其流形切向量为 $\vec{\mathbf{v}}_i$。则局部不变性要求 $\nabla_{\vec{\mathbf{x}}}f(\vec{\mathbf{x}}_i)$ 与 $\vec{\mathbf{v}}_i$ 正交。
5. 正切传播算法的正则化惩罚项 Ω 要求 f 在 $\vec{\mathbf{x}}$ 的 $\vec{\mathbf{v}}$ 方向的导数是较小的：

$$\Omega(f)=\sum_i\left(\nabla_{\vec{\mathbf{x}}}f(\vec{\mathbf{x}}_i)^T\vec{\mathbf{v}}_i\right)^2$$

   - 这里并没有严格要求其为零，而是比较小即可。
   - $\nabla_{\vec{\mathbf{x}}}f(\vec{\mathbf{x}})$ 是这样算出的：
     - 首先根据样本的类别来分成不同的流形。
     - 然后找出 $\vec{\mathbf{x}}$ 在本流形的近邻点来计算出梯度。
6. 正切传播有两个主要缺点：
   - 模型的正则化只能抵抗无穷小的扰动。因为若扰动较大，则可能切平面变化较大。

     作为对比：显式的数据集增强能够抵抗较大扰动。
   - 梯度沿着切向方向分量很小的假设对于采用了 `relu` 单元的模型是困难的。因为 `relu` 单元的导数与 $\vec{\mathbf{w}}$ 无关，是个常数。

     `sigmoid` 或者 `tanh` 单元就没有这个问题，因为可以通过采用大权重在饱和区来收缩导数。
7. 正切传播算法和人工数据集增强（如：沿着 x 轴的平移）都要求模型对于输入变化的某些特定的方向是不变的。
   - 正切传播算法要求模型对于输入位于流形切平面上的所有方向上的小变化都是不变的。
   - 正切传播算法能够抵抗较大扰动。
8. 流形正切分类器：使用自编码器通过无监督学习来学习流形的结构，然后使用正切传播算法来正则化神经网络分类器。

十、其它相关

10.1 稀疏表达

1. 稀疏表达：鼓励 $\vec{\mathbf{x}}$ 的表达 $\vec{\mathbf{h}}$ 为稀疏的，即：目标函数为：$\tilde{J}(\vec{\theta};\mathbf{X},\vec{\mathbf{y}})=J(\vec{\theta};\mathbf{X},\vec{\mathbf{y}})+\alpha\Omega(\vec{\mathbf{h}})$ 。
   - 它与普通正则化区别是：它对 $\vec{\mathbf{x}}$ 的表达 $\vec{\mathbf{h}}$ 进行限制，而不是参数 $\vec{\theta}$ 进行限制。
   - 通常采用 L_1 正则化可以获取稀疏表达：$\Omega(\vec{\mathbf{h}})=||\vec{\mathbf{h}}||_1$ 。

     也有其他方法可以获取稀疏表达，如：正交匹配追踪 `orthogonal matching pursuit`：

     $$\arg\min_{\vec{\mathbf{h}},||\vec{\mathbf{h}}||_0<k}||\vec{\mathbf{x}}-\mathbf{W}\vec{\mathbf{h}}||^2$$

     其中 $||\vec{\mathbf{h}}||_0$ 是 $\vec{\mathbf{h}}$ 中非零项的个数。

     当 $\mathbf{W}$ 被限定为正交矩阵时，该问题可以被高效解决。

10.2 半监督学习

1. 在深度学习中，半监督学习指的是学习一个表达 `representation`：$\vec{\mathbf{h}}=f(\vec{\mathbf{x}})$ 。学习的目标是：使得同类中的样例有类似的表达。
2. 通常可以构建两个模型：生成模型 $p(\vec{\mathbf{x}})$（或者 $p(\vec{\mathbf{x}},y)$） 与判别模型 $p(y\mid\vec{\mathbf{x}})$，其中生成模型与判别模型共享参数。
   - 生成模型 $p(\vec{\mathbf{x}})$（或者 $p(\vec{\mathbf{x}},y)$） 表达了对于监督学习问题解的先验知识。

     即：$p(\vec{\mathbf{x}})$ 的结构通过共享参数的方式连接到 $p(y\mid\vec{\mathbf{x}})$ 。
   - 不需要将无监督学习和监督学习部分进行分离，二者位于同一个网络结构中。
   - 最终需要对监督准则 $-\log p(y\mid\vec{\mathbf{x}})$ 与无监督准则 $-\log p(\vec{\mathbf{x}})$（或者 $-\log p(\vec{\mathbf{x}},y)$） 进行权衡。

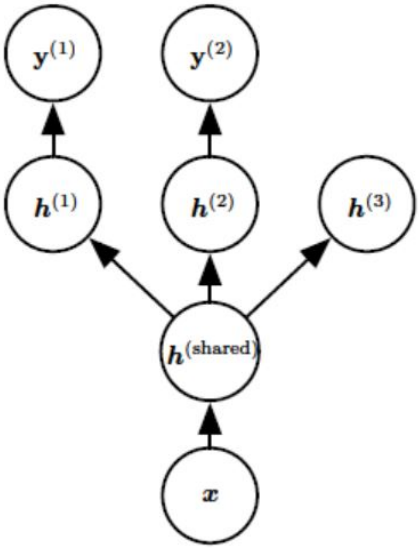
     这样可以得到比单纯的生成模型或者单纯的判别模型更好的模型，从而提高了泛化能力。

10.3 多任务学习

1. 多任务学习是指几个任务共享相同的样本集。这可以视作一种参数上的软约束。

   下图给出了多任务学习中的一个非常普遍的形式：有两个监督任务和一个无监督任务。所有的任务都共享相同的输入 $\vec{\mathbf{x}}$ 和第一级中间层 $\vec{\mathbf{h}}^{(share)}$，具体的任务使用了具体的表示层。

   因为共享参数，其统计强度大大提高因此能改进泛化能力。但是前提条件是：确实有某些信息在不同任务之间共享了。这要求不同任务之间存在某些统计关系。



2. 多任务学习刻画了一个先验知识：这些不同的任务中，能解释数据变化的因子是跨任务共享的。

10.4 正则化和欠定问题

1. 机器学习的许多线性问题（包括线性回归和`PCA`），都依赖于求 $\mathbf{X}^T\mathbf{X}$ 的逆矩阵。
   - 当 $\mathbf{X}^T\mathbf{X}$ 是可逆时，该问题有解析解。
   - 当 $\mathbf{X}^T\mathbf{X}$ 是奇异矩阵时，该问题是欠定的。此时可以考虑正则化形式：求解 $\mathbf{X}^T\mathbf{X}+\alpha\mathbf{I}$ 的逆矩阵。
2. 大多数形式的正则化能够用于欠定问题。

   如：`Moore-Penrose`求解欠定线性方程， $\mathbf{X}$ 伪逆的一个定义：$\mathbf{X}^+=\lim_{\alpha\to 0}(\mathbf{X}^T\mathbf{X}+\alpha\mathbf{I})^{-1}\mathbf{X}^T$ 。
3. 使用正则化来解决欠定问题的思想超出了机器学习的范畴。