5 神经网络

由固定基函数的线性组合具有一些有用的分析和计算性质，但是其实际应用被维数灾难问题所限制，为了获得更加有效的模型，我们有必要根据数据调节基函数。

- 一种方法是使用支持向量机：
  首先定义以训练数据点为中心的基函数，然后在训练过程中选择一个子集。其优点在于：虽然在训练阶段涉及到非线性优化，但目标函数是凸函数，所以最优问题的解释很直接的。最终模型当中的基函数数目通常远小于训练数据点的数目。相关向量机也选择固定基函数集合的一个子集，通常会产生一个相当稀疏的模型，与支持向量机的不同，相关向量机也产生概率形式的输出，虽然这种输出的产生以训练阶段的非凸优化为代价。
- 另一种方法是实现固定基函数的数量，但基函数是可调节的，也就是采用参数形式的基函数，这些参数可以在训练阶段进行调节。在模式识别中，最成功的模型是前馈神经网络，也叫多层感知机。实际上该模型是由多层的logistic回归模型（具有连续非线性）而非多层的感知机（具有非连续非线性）组成的。与支持向量机相比，在大多数情况下该模型会更加简洁，计算速度更快，但是构成网络训练根基的似然函数不再是模型参数的凸函数。

5.1 前馈神经网络

回归的线性模型和分类的线性模型是基于固定非线性函数$\phi_j(x)$的线性组合，形式可以写作：

$$y(x,w)=f(\sum_{j=1}^{M}w_j\phi_j(x))$$

其中$f(.)$在分类问题中是一个非线性激活函数，在回归问题中则是一个恒等函数。我们的目标是将该模型推广，使得$\phi_j(x)$依赖于参数，从而使得这些参数以及系数$\{w_j\}$能够在训练阶段调节。而神经网络使用的是与上式形式相同的基函数，也就是说，基函数本身就是输入的线性组合的非线性函数，其中线性组合的系数是可调节参数。
所以，我们先构造输入变量$x_1,...,x_D$的M个线性组合，其形式为：

$$a_j=\sum_{i=1}^{D}w_{ji}^{(1)}x_i+w_{j0}^{(1)}$$

其中$j=1,\cdots,M$,而且上标(1)表示的是对应参数是神经网络的第一层.我们将$w_{ji}^{(1)}$称为权，将参数$w_{i0}^{(1)}$称为偏置，a_j被称为激活。每一个激活都使用一个可微的非线性激活函数$h(.)$进行变换，可以得到：

$$z_j=h(a_j)$$

这些量对应着基函数的输出，这些基函数在神经网络中被称为隐含单元。非线性函数h通常被选为S型的函数，例如logistic函数或者双曲正切函数。这些值再次被线性组合，得到输出单元激活

$$a_k=\sum_{j=1}^{M}w_{kj}^{(2)}z_j+w_{k0}^{(2)}$$

其中$k=1,\cdots,K$,K是输出的总数，这个变换对应的是神经网络的第二层，最后使用恰当的激活函数对输出单元激活进行变换，得到神经网络的一组输出$y_1,...,y_K$.激活函数的选择由数据本身以及目标变量假定的分布决定，因此对于标准

的回归问题，激活函数是恒等函数，从而$y_k = a_k$,类似的，对于多个二元分类问题，每一个输出单元激活使用logistic sigmod函数进行变换，即：

$$y_k = \sigma(a_k) = \frac{1}{1+\exp(-a_k)}$$

对于多类问题，我们可以使用softmax激活函数，所以整体的网络函数：

$$y_k(x,w) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\Big(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\Big) + w_{k0}^{(2)}\right)$$

我们将所有的权参数和偏置参数聚集到一起，记作向量w，所以神经网络可以看作由输入到输出的非线性网络，并且可以通过调节参数w进行控制。上面公式的计算过程可以看作信息通过网络的前向传播，然后我们额外定义输入x_0使得$x_0 = 1$,则可以转化为：

$$a_j = \sum_{i=0}^{D} w_{ji}^{(1)} x_i$$

最后得到：

$$y_k(x,w) = \sigma\left(\sum_{j=0}^{M} w_{kj}^{(2)} h\Big(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\Big)\right)$$

神经网络模型与感知机模型的重要区别是神经网络在隐含单元中使用的是连续的sigmod非线性函数，而感知机使用的是阶梯函数这一非线性函数，这意味着神经网络函数关于神经网络参数是可微的。
如果网络中所有的隐含单元激活函数都是线性函数，则对于这样的网络，我们总能找到一个等价的无隐含单元的网络。如果隐含单元数目小于输入单元或者输出单元的数目，则网络产生的变换不是最一般的从输入到输出的线性变换，因为在隐含单元处维度降低造成了信息的丢失。
前馈网络中每一个单元都计算了下面的函数：

$$z_k = h\left(\sum_j w_{kj} z_j\right)$$

其中求和的对象是所有向单元k发送链接的单元（偏置参数也包含在求和式当中）。

5.5.1 权空间对称性

前馈神经网络中的重要性质是，对于多个不同的权向量w的选择，网络可能产生相同的从输入到输出的映射函数。所以在一个输入层，一个隐层，一个输出层的神经网络中，网络中有M个隐含节点，激活函数是双曲正切函数，且两层之间是完全链接。如果我们把作用于某个特定的隐含单元的所有权值以及偏置全部变号，则对于给定的输入模式，隐含单元的激活符号也会发生改变。这是因为双曲正切函数是一个奇函数。这种变换可以通过改变所有从这个隐含单元到输出单元的权值符号的方式进行精确补偿。因此，可以通过改变特定的一组权值产生相同的映射函数。对于M个这样的符号改变对称性，所以对于任意给定的权向量都是2^M个等价的权向量中的一个。
类似的，将与某个特定隐含节点所关联的所有输入输出权值都变为与不同的隐含节点所关联的对应权值。输入输出映射不变，但是对应的权向量不同，对于M个隐含节点，任一权向量都属于交换对称性产生的$M!$个等价权向量中的一个，对应于$M!$个不同隐含单元的顺序。所以网络中有一个整体的权空间对称性因子$M!2^M$。

5.2 网络训练

我们将神经网络看作是输入变量x到输出变量y的参数化非线性函数中的一大类。首先看回归问题，先只考虑一元目标变量t的情形， 其中t可以取任何实数值，我们假定t服从高斯分布，均值与x相关，由神经网络的输出确定，即：

$$p(t|x,w)=\mathcal{N}(t|y(x,w),\beta^{-1})$$

对于这样的条件分布，我们将输出单元激活函数取为恒等函数即可，因为这样的网络可以近似任何从x到y的连续函数。给定N个独立同分布的观测组成的数据集$X=\{x_1,\cdots,x_N\}$,以及相应的目标值$\mathbf{t}=\{t_1,\cdots,t_N\}$,可以构造对应的似然函数：

$$p(\mathbf{t}|X,w,\beta)=\prod_{n=1}^{N}p(t_n|x_n,w,\beta)$$

取负对数，可以得到误差函数：

$$\frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n,w)-t_n\}^2-\frac{N}{2}\ln\beta+\frac{N}{2}\ln(2\pi)$$

首先考虑w,最大化似然函数等价于最小化平方和误差函数：

$$E(w)=\frac{1}{2}\sum_{n=1}^{N}\{y(x_n,w)-t_n\}^2$$

最小化$E(w)$得到w的值w_{ML},因此它对应最大化似然函数。在实际应用中，神经网络函数$y(x_n,w)$的非线性性质导致误差函数$E(w)$不是凸函数，因此实际应用中可能寻找的是似然函数的局部最大值，对应误差函数的局部最小值。
找到w_{ML}之后，β的值可以通过最小化似然函数的负对数的方式得到：

$$\frac{1}{\beta_{ML}}=\frac{1}{N}\sum_{n=1}^{N}\{y(x_n,w_{ML}-t_n)\}^2$$

若我们有多个目标变量，并且假定x 和w的条件下，目标变量之间是相互独立的，而且噪声精度均为β,所以目标变量的条件分布为

$$p(t|x,w)=\mathcal{N}(t|y(x,w),\beta^{-1}I)$$

与一元目标变量的情形相同，可以看到最大似然权值由最小化平方和误差函数确定，于是噪声精度为

$$\frac{1}{\beta_{ML}}=\frac{1}{NK}\sum_{n=1}^{N}||y(x_n,w_{ML})-t_n||^2$$

其中K是目标变量的数量。
在回归问题中，可以将神经网络看成一个有恒等输出激活函数的模型，也就是$y_k=a_k$。平方和误差函数的性质：

$$\frac{\partial E}{\partial a_k}=y_k-t_k$$

再来考虑二分类的情况，我们有一个单一变量t,且$t=1$表示类别C_1,$t=0$表示C_2，所以对于一个具有单一输出的网络，激活函数是 logistic sigmod 函数：

$$y = \sigma(a) = \frac{1}{1 + \exp(-a)}$$

从而$0 \leq y(x, w) \leq 1$,可以将$y(x, w)$表示为条件概率$p(C_1|x)$,这时$p(C_1|x)$为$1 - y(x, w)$，如果给定输入，目标变量的条件概率分布是一个伯努利分布，形式为：

$$p(t|x, w) = y(x, w)^t \{1 - y(x, w)\}^{1-t}$$

所以考虑由独立观测组成的训练集，由负对数似然函数给出的误差函数就是交叉熵误差函数，形式为

$$E(w) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1 - t_n) \ln (1 - y_n)\}$$

这里没有与噪声精度相类似的项，因为假设目标值的标记都是正确的。
对于K个独立的二元分类问题，可以使用带有K个输出的神经网络，每一个输出都是一个 logistic sigmod 激活函数。假定标签独立：

$$p(t|x, w) = \prod_{k=1}^{K} y_k(x, w)^{t_k} [1 - y_k(x, w)]^{(1-t_k)}$$

取负对数，可以得到误差函数：

$$E(w) = -\sum_{n=1}^{N} \sum_{k=1}^{K} \{t_{nk} \ln y_{nk} + (1 - t_{nk}) \ln (1 - y_{nk})\}$$

其中y_{nk}表示$y_k(x_n, w)$。
对于标准的多分类问题，其中每个输入被分到K个互斥类别中，网络输出可以表示为$y_k(x, w) = p(t_k = 1|x)$,因此误差函数为：

$$E(w) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \ln y_k(x_n, w)$$

输出单元激活函数是**softmax**函数：

$$y_k(x, w) = \frac{\exp(a_k(x, w))}{\sum_j \exp(a_j(x, w))}$$

满足$0 \leq y_k \leq 1, \sum_k y_k = 1$，如果给所有的$a_k(x, w)$都加一个常数，则$y_k(x, w)$是不变的，所以误差函数在权空间中某些方向上是常数。如果我们给误差函数加一个恰当的正则化项，则可以避免这种问题。
对于回归问题，使用线性输出和平方和误差函数，对于二元分类问题，使用 logistic sigmod 输出和交叉熵误差函数。对于多分类问题，可以使用**softmax**输出和对应的多分类交叉熵误差函数。对于涉及的两类的分类问题，可以用单一的 logistic sigmod 输出，也可以使用有两个输出的神经网络，输出激活函数为softmax函数。

5.2.2 局部二次近似

考虑$E(w)$在权空间中某点$\hat{w}$处的泰勒展开

$$E(w) \simeq E(\hat{w}) + (w-\hat{w})^T b + \frac{1}{2}(w-\hat{w})^T H(w-\hat{w})$$

其中$b = \nabla E|_{w=\hat{w}}$。
Hessian矩阵$H = \nabla\nabla E$的元素为：

$$H_{ij} = \left.\frac{\partial E}{\partial w_i \partial w_j}\right|_{w=\hat{w}}$$

梯度的局部近似：

$$\nabla E \simeq b + H(w-\hat{w})$$

在误差函数最小值点w^*进行局部二次近似，此时没有线性项，因为在这时$\nabla E = 0$，此时$E(w)$的泰勒展开变成了：

$$E(w) \simeq E(w^*) + \frac{1}{2}(w-w^*)^T H(w-w^*)$$

这里Hessian矩阵在点w^* 处计算，为了用几何形式表示结果，考虑Hessian矩阵的特征值方程：

$$Hu_i = \lambda_i u_i$$

其中特征向量u_i构成了完备的单位正交集合：

$$u_i^T u_j = \delta_{ij}$$

把$(w-w^*)$展开成特征值的线性组合的形式：

$$w - w^* = \sum_i \alpha_i u_i$$

这可以看作是坐标系的变换，坐标系的原点变成了w^*，坐标轴旋转与特征向量对齐。所以使用

$$E(w) \simeq E(w^*) + \frac{1}{2}(w-w^*)^T H(w-w^*)$$
$$w - w^* = \sum_i \alpha_i u_i$$
$$u_i^T u_j = \delta_{ij}$$
$$Hu_i = \lambda_i u_i$$

可以将其转换为：

$$E(w) = E(w^*) + \frac{1}{2}\sum_i \lambda_i \alpha_i^2$$

我们知道，矩阵正定当且仅当：

$$v^T H v > 0 \text{ 对所有} v \neq 0 \text{都成立}$$

由于特征向量$\{u_i\}$组成了一个完备集，因此任意的向量v都可以写作是

$$v = \sum_i c_i u_i$$

所以可以得到：

$$v^T H v = \sum_i c_i^2 \lambda_i$$

所以H是正定的，当且仅当其所有特征值均为正。在新坐标系中，基向量是特征向量$\{u_i\}$，E为常数的轮廓线是以原点为中心的椭圆，对于一维权空间，驻点w^*满足下面条件时取得最小值：

$$\left.\frac{\partial^2 E}{\partial w^2}\right|_{w^*} > 0$$

对应的D维结论是，在w^*处的Hessian矩阵是正定矩阵。

5.2.3 使用梯度信息

我们可以使用误差反向传播的方法高效计算误差函数的梯度，这个梯度的信息可以大幅度加快找到极小值点的速度。
在误差函数的二次近似

$$E(w) \simeq E(\hat{w}) + (w - \hat{w})^T b + \frac{1}{2}(w - \hat{w})^T H (w - \hat{w})$$

误差曲面由b和H确定，它包含了$\frac{W(W+3)}{2}$个独立的元素，其中W是w的维度，即网络中可调节参数的总数。二次近似极小值点位置依赖于$O(W^2)$个参数，并且不奢求在收集到$O(W^2)$条独立信息之前就找到最小值。如果不使用梯度信息，则不得不进行$O(W^2)$次函数求值，每次需要$O(W)$个步骤，因此，使用该方法的最小值需要计算复杂度为$O(W^3)$。
将这种方法与使用梯度信息的方法进行对比，由于每次计算∇E都会带来W条信息，所以我们预计找到函数最小值需要计算$O(W)$次梯度，使用反向传播算法，每一个计算需要$O(W)$步，因此可以在$O(W^2)$步骤内找到极小值。

5.2.4 梯度下降最优化

最简单的使用梯度信息的方法是将$w^{\tau+1} = w^\tau + \Delta w^\tau$中的权值更新方式选择为每次权值更新都是在负梯度上一次小的移动，即：

$$w^{\tau+1} = w^\tau - \eta \nabla E(w^\tau)$$

其中参数$\eta > 0$被称为学习率，每次更新的时候梯度使用新的权值向量重新计算，然后将该过程重复下去，误差函数是关于训练集定义的，因此为了计算∇E，每一次都需要重新处理整个数据集，每一步沿着误差函数下降最快的方向移动，因此该方法被称为梯度下降法。对于梯度下降法的在线方法：基于一组独立观测的最大似然函数的误差函数有一个求和式组成，求和项每一项都对应着一个数据点：

$$E(w) = \sum_{n=1}^{N} E_n(w)$$

在线梯度下降也叫随机梯度下降算法，使得权向量的更新每一次只依赖一个数据点，也就是：

$$w^{\tau+1} = w^\tau - \eta \nabla E_n(w^\tau)$$

这个更新在数据集上循环重复进行，既可以顺序处理数据，也可以随机又重复的选择数据点，也可以每次更新依赖于数据点的一部分。
随机梯度下降的一个优点是更高效的处理数据当中的冗余性。

5.3 误差反向传播

在局部信息传递的思想中，信息可以在神经网络中交替向前向后传播，这种方法被称为误差反向传播，有时简称反传。
大部分训练算法涉及到一个迭代的步骤用于误差函数的最小化，以及通过一系列步骤进行权值调节。在每一个这样迭代的过程中，我们可以区分这两个不同的阶段，在第一个阶段中，误差函数关于权值的导数必须要计算出来，反向传播方法的重要贡献是提供了计算这些导数的高效方法，误差在这个阶段通过网络就可以反向传播。在第二个阶段，导数用于权值的调整量

5.3.1 误差函数导数的计算

许多实际应用中使用的误差函数是由若干项求和式组成的，每一项对应的是训练集的一个数据点，即：

$$E(w) = \sum_{n=1}^{N} E_n(w)$$

在这里需要计算$\nabla E_n(w)$,可以直接使用顺序优化的方法计算，或者使用批处理方法在训练集上进行累加。
首先对于简单的线性模型：

$$y_k = \sum_i w_{ki} x_i$$

对于特定的输入模式n，误差函数的形式为：

$$E_n = \frac{1}{2} \sum_k (y_{nk} - t_{nk})^2$$

其中$y_{nk} = y_k(x_n, w)$，误差函数关于权值w_{ji}的梯度为：

$$\frac{\partial E_n}{\partial w_{ji}} = (y_{nj} - t_{nj}) x_{ni}$$

所以可以表示为与链接w_{ji}相关联的误差信号$y_{nj} - t_{nj}$和与链接的输入端相关联的变量x_{ni}的乘积。
在一个一般的前馈网络中，每一个单元都会计算输入的一个加权和，形式为：

$$a_j = \sum_i w_{ji} z_i$$

其中z_i是一个单元的激活或者是输入，它向单元j发送一个链接，w_{ji}是与这个链接相关联的权值，我们可以将偏置整合到求和式当中，求和式通过一个非线性激活函数$h(.)$进行变换，得到单元j的激活z_j,形式为：

$$z_j = h(a_j)$$

求和式某些z_j可以是输入，类似的，单元j可以是输出。
我们假定给提供了相应的输入向量，反复用上面的公式，计算神经网络中所有隐含但元和输出单元的激活，这个过程叫做

正向传播，因为他可看作网络中前向流动的信息流。
然后计算E_n关于权值w_{ji}的导数，每一个单元的输出依赖于某个特定的输入模式n，我们省略神经网络当中的下标n，首先，E_n只通过单元j经过求和之后的输入a_j对权值w_{ji}产生依赖，使用链式法则：

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j}\frac{\partial a_j}{\partial} w_{ji}$$

我们引入δ:

$$\delta_j = \frac{\partial E_n}{\partial a_j}$$

而对上面的公式可以知道：

$$\frac{\partial a_j}{\partial w_{ji}} = z_i$$

所以可以得到：

$$\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i$$

使用标准链接函数作为输出单元的激活函数，对于输出单元，可以得到：

$$\delta_k = y_k - t_k$$

对于隐含单元的δ值，我们可以使用偏导数的链式法则：

$$\delta_j = \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k}\frac{\partial a_k}{\partial a_j}$$

其中求和式对象是所有向单元j发送链接的单元k，单元k可以包含其他的隐含单元和输出单元，可以得到反向传播公式：

$$\delta_j = h'(a_j)\sum_k w_{kj}\delta_k$$

这表明，一个特定的隐含单元δ值可通过网络中更高层单元的δ进行反向传播实现。反响传播中求和式对于w_{ji}的第一个下标进行求和，而正向传播的求和针对的是第二个下标，所以可以递归使用公式反向传播算法计算前馈网络中所有隐含单元的δ值。所以反向传播算法：

- 对于网络中的一个输入向量x_n,使用正向传播找到所有隐含单元和输出单元的激活
- 使用$\delta_k = y_k - t_k$计算所有输出单元的δ_k
- 使用反向传播算法，计算所有网络中隐含单元的δ_j
- 使用$\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i$计算导数

对于批处理方法，总误差E的导数可以使用对训练集中每一个模式重复以上步骤，对所有的模式求和，得到：

$$\frac{\partial E}{\partial w_{ji}} = \sum_n \frac{\partial E_n}{\partial w_{ji}}$$

这里我们假设网络中每一个隐含单元或输入单元都有共同的激活函数$h(.)$。

5.3.2 一个简单的例子

我们考虑两层神经网络，误差函数的形式为平方和误差函数，输出单元激活函数为线性激活函数，即$y_k = a_k$，隐含单元的激活函数为$h(a) = \tanh(a)$,其中：

$$\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$$

其导数为

$$h'(a) = 1 - h(a)^2$$

对于标准的平方和误差函数，对于模式n，误差为：

$$E_n = \frac{1}{2}\sum_{k=1}^{K}(y_k - t_k)^2$$

对于训练集中的每一个模式，我们先进行正向传播：

$$\begin{aligned} a_j &= \sum_{i=0}^{D} w_{ji}^{(1)} x_i \\ z_j &= \tanh(a_j) \\ y_k &= \sum_{j=0}^{M} w_{kj}^{(2)} z_j \end{aligned}$$

然后计算每一个输出单元的δ值：

$$\delta_k = y_k - t_k$$

然后将δ用下面公式进行反向传播，得到隐含的δ_j的值：

$$\delta_j = (1 - z_j^2)\sum_{k=1}^{K} w_{kj}\delta_k$$

然后得到两层的权值导数：

$$\begin{aligned} \frac{\partial E_n}{\partial w_{ji}^{(1)}} &= \delta_j x_i \\ \frac{\partial E_n}{\partial w_{kj}^{(2)}} &= \delta_k z_j \end{aligned}$$

5.3.3 反向传播的效率

我们需要考察误差函数导数的计算次数与网络中权值和偏置总数W之间的关系，对于给定的输入模式，需要$O(W)$次操作，其中W充分大，因为权值的数量比单元的数量大得多。因为正向传播计算复杂度取决于求和式的计算。
一种计算误差函数导数的反向传播算法是使用有限差，首先令权值有一个扰动，然后近似导数：

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{E_n(w_{ji}+\epsilon) - E(w_{ji})}{\epsilon} + O(\epsilon)$$

通过让ϵ变小，对导数的近似精度可以提升，使用对称的中心差，使得精度大大提升：

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{E_n(w_{ji}+\epsilon) - E(w_{ji}-\epsilon)}{2\epsilon} + O(\epsilon^2)$$

在这种情况下，$O(\epsilon) = 0$,可以通过泰勒展开进行证明，所以修正项为$O(\epsilon^2)$，但实际计算步骤书大约变为了二倍。
计算数值导数的问题在于计算复杂度每次正向传播为$O(W)$,对于网络中的W个权值，每一个权值必需单独施加扰动，因此整体时间复杂度为$O(W^2)$.
当训练网络时，导数应该使用反向传播算法计算，因为其有最高的精度和效率，但是可以使用一些样例将结果与数值导数结果进行对比，检查结果正确性。

5.3.4 Jacobian矩阵

误差反向传播算法也可以用来计算其他形式的导数，这里考虑**Jacobian**矩阵的计算，元素值是网络输出关于输入的导数：

$$J_{ki} = \frac{\partial y_k}{\partial x_i}$$

其中计算每一个导数时，其他输入都固定，假定对于参数w，最小化误差函数E，其导数为：

$$\frac{\partial E}{\partial w} = \sum_{k,j} \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial z_j}\frac{\partial z_j}{\partial w}$$

由于**Jacobian**矩阵度量了输出对于每一个输入变量的改编的敏感性，可以得到：

$$\Delta y_k \simeq \sum_i \frac{\partial y_k}{\partial x_i}\Delta x_i$$

只要Δx_i足够小，关系就成立。由于训练后神经网络中网络映射是非线性的，因此**Jacobian**矩阵元素不是常数，而是依赖于具体的输入向量，因此对于每一个新的输入变量，**Jacobian**矩阵必须重新计算。
Jacobian矩阵的元素可以使用反向传播算法进行计算，对于元素J_{ki}：

$$\begin{aligned} J_{kj} &= \frac{\partial y_k}{\partial x_i} \\ &= \sum_j \frac{\partial y_k}{\partial a_j}\frac{\partial a_j}{\partial x_i} \end{aligned}$$

我们现在用反向传播算法计算导数$\frac{\partial y_k}{\partial a_j}$：

$$\frac{\partial y_k}{\partial a_j} = \sum_l \frac{\partial y_k}{\partial a_l}\frac{\partial a_l}{\partial a_j}$$
$$= h'(a_j)\sum_l w_{lj}\frac{\partial y_k}{\partial a_l}$$

反向传播开始于输出单元，对于输出单元，导数可以直接从输出单元激活函数的形式中得到，比如对于每一个输出单元都有各自的 sigmod 函数，则可以得到：

$$\frac{\partial y_k}{\partial a_l} = \delta_{kl}\sigma'(a_l)$$

对于**softmax**函数，可以得到：

$$\frac{\partial y_k}{\partial a_l} = \delta_{kl}y_k - y_k y_l$$

所以**Jacobian**矩阵的计算方法如下：将输入空间要寻找**Jacobian**矩阵的点映射为一个输入向量，将其作为网络的输入，使用正向传播算法，得到网络的所有隐含单元和输出单元的激活。对于**Jacobian**矩阵的每一行k（对应于输出单元k），使用递归关系进行反向传播，对于所有隐含节点，用上面的公式进行反向传播，最后使用J_{kl}进行输入单元的反向传播，另一种方式是使用正向传播算法，可以使用类似反向传播的方式推导出来。
与之前一样，可以使用数值导数的方法检查正确性：

$$\frac{\partial y_k}{\partial x_i} = \frac{y_k(x_i+\epsilon) - y_k(x_i-\epsilon)}{e\epsilon} + O(\epsilon^2)$$

对于一个有D个输入的网络来说，这种方法需要2D次正向传播。

5.4 Hessian 矩阵

反向传播算法一样可以计算误差函数的二阶导数：

$$\frac{\partial^2 E}{\partial w_{ji}\partial w_{kl}}$$

将所有权值和偏置参数看作一个向量w的元素w_i会更方便，此时二阶导数组成了Hessian矩阵H的元素H_{ij},其中$i,j \in \{1,\cdots,W\}$,其中W为权值和偏置的总数。
Hessian矩阵的重要作用：

- 一些用来训练神经网络的非线性最优算法是基于误差曲面的二阶性质，这些性质由hessian矩阵控制
- 对于训练数据的微小变化，Hessian矩阵构成了快速重新训练前馈网络算法的基础
- Hessian矩阵的逆矩阵用以鉴别神经网络中最不重要的权值，这是神经网络剪枝算法的一部分。
- Hessian矩阵是贝叶斯神经网络的拉普拉斯近似的核心，其逆矩阵用来确定训练过的神经网络的预测分布，其特征值确定了超参数的值，其行列式用来计算模型证据

使用反向传播算法的一个扩展，Hessian矩阵可以精确计算出来。
如果网络中有W个参数，则Hessian矩阵的维度为$W \times W$,因此对于每一个模式，计算Hessian矩阵的计算量为$O(W^2)$.

5.4.1 对角近似

由于很多对于Hessian矩阵的应用需要求出Hessian矩阵的逆矩阵，而不是其本身，所以为了容易计算Hessian矩阵的逆矩阵，我们将非对角线上的元素置为0。
我们考虑有一系列项的求和式组成的误差函数，每一项对应于数据集中的一个模式，即$E=\sum_n E_n$,所以Hessian矩阵可以通过每次考虑一个模式然后对于所有模式求和的方式得到，对于模式n，Hessian矩阵的对角线元素可以写作：

$$\frac{\partial^2 E_n}{\partial w_{ji}^2}=\frac{\partial^2 E_n}{\partial a_j^2}z_i^2$$

由于

$$\frac{\partial E_n}{\partial a_j}=h'(a_j)\sum_k w_{kj}\frac{\partial E_n}{\partial a_k}$$

右侧的二阶导数可以通过递归使用微分的链式法则的方式求出来，可以得到反向传播方程的形式为：

$$\frac{\partial^2 E_n}{\partial a_j^2}=h'(a_j)^2\sum_k\sum_{k'}w_{kj}w_{k'j}\frac{\partial^2 E_n}{\partial a_k\partial_{k'}}+h''(a_j)\sum_k w_{kj}\frac{\partial E_n}{\partial a_k}$$

如果忽略二阶导数的非对角线元素，那么我们可以得到：

$$\frac{\partial^2 E_n}{\partial a_j^2}=h'(a_j)^2\sum_k w_{kj}^2\frac{\partial^2 E_n}{\partial a_k^2}+h''\sum_k w_{kj}\frac{\partial E_n}{\partial a_k}$$

需要计算这个近似所需的计算步骤是$O(W)$,其中W是网络中所有权值和偏置的总数，对于原始的Hessian矩阵，所需要的步骤数是$O(W^2)$。
对角近似的主要问题是，在实际应用中Hessian矩阵通常是强烈非对角化的，因此需要谨慎使用这些近似手段。

5.4.2 外积近似

当神经网络应用于回归问题的时候，通常使用下面形式的平方和误差函数：

$$E=\frac{1}{2}\sum_{n=1}^N(y_n-t_n)^2$$

我们考虑单一输出的情形（很直接就可以推广到多个输出），这样，我们可以将Hessian矩阵写成下面的形式：

$$H=\nabla\nabla E=\sum_{n=1}^N\nabla y_n(\nabla y_n)^T+\sum_{n=1}^N(y_n-t_n)\nabla\nabla y_n$$

如果网络已经在训练集上训练过了，输出y_n已经很接近t_n,那么公式的第二项很小，所以可以忽略。在更一般的情况下，忽略这一项更合适，因为最小化平方和误差的最优函数是目标数据的条件均值。y_n-t_n是一个零均值的随机变量。我们假设它的值与右侧的二阶导数项无关，那么对于n的求和项中，整个项的平均值为0.
忽略公式的第二项，就得到了 Levenberg-Marquardt 近似，或者叫外积近似，形式为：

$$H\simeq\sum_{n=1}^N b_n b_n^T$$

其中$b_n = \nabla a_n = \nabla y_n$,因为输出单元的激活函数为恒等函数。Hessian矩阵的外积近似只涉及到误差函数的一阶导数，这可以通过使用标准反向传播算法在$O(W)$个步骤高效的求出，通过简单乘法，矩阵的元素可以在$O(W^2)$个步骤计算出来。
这样的近似只在网络被恰当的训练出来以后才成立，对于一般的网络映射，右侧的二阶导数项是不能忽略的。
当误差函数是交叉熵误差函数的时候，输出单元的激活函数是 logistic sigmod 函数的神经网络中，对应的近似是：

$$H \simeq \sum_{n=1}^{N} y_n(1-y_n)b_n b_n^T$$

对于输出函数是 softmax 函数的多类神经网络，可以得到类似的结果

5.4.3 Hessian矩阵的逆矩阵

使用外积近似$H_N = \sum_{n=1}^{N} b_n b_n^T$,我们可以高效地计算Hessian矩阵的逆矩阵。
其中，$b_n = \nabla_w a_n$是数据点n产生的输出单元激活对于梯度的贡献。我们每次处理一个数据点。假设我们使用前L个数据点得到的Hessian逆矩阵，可以得到：

$$H_{L+1} = H_L + b_{L+1} b_{L+1}^T$$

为了计算逆矩阵，我们使用下面的恒等式：

$$(M + vv^T)^{-1} = M^{-1} - \frac{(M^{-1}v)(v^T M^{-1})}{1 + v^T M^{-1} v}$$

我们令$H_L = M, b_{L+1} = v$，我们可以得到：

$$H_{L+1}^{-1} = H_L^{-1} - \frac{H_L^{-1} b_{L+1} b_{L+1}^T H_L^{-1}}{1 + b_{L+1}^T H_L^{-1} b_{L+1}}$$

用这样的方式，数据点可以依次使用，直至$L+1=N$。最开始的矩阵H_0被选为αI，其中α是一个较小的值，所以实际上找的是$H + \alpha I$的逆矩阵。
Hessian矩阵有时可以作为神经网络训练算法的一部分被间接计算。

5.4.4 有限差

我们可以使用有限差的方法求二阶导，精度受数值计算的精度限制，如果我们对于每一对可能的权值施加一个扰动，那么

$$\frac{\partial^2 E}{\partial w_{ji} \partial w_{lk}} = \frac{1}{4\epsilon^2}\Big\{E(w_{ji}+\epsilon, w_{lk}+\epsilon) - E(w_{ji}+\epsilon, w_{lk}-\epsilon) - E(w_{ji}-\epsilon, w_{lk}+\epsilon) + E(w_{ji}-\epsilon, w_{lk}-\epsilon)\Big\}$$

使用对称的中心差，我们确保了残留的误差项是$O(\epsilon^2)$而不是$O(\epsilon)$,由于在Hessian矩阵中有W^2个元素，每一个元素要四次正向传播过程，每一个传播过程要$O(W)$次操作（每一个模式），所以计算完整的Hessian矩阵需要$O(W^3)$次操作，所以计算性质较差。
所以更高效的方法是将中心差用于一阶导数，一阶导数可通过反向传播算法计算：

$$\frac{\partial^2 E}{\partial w_{ji} \partial w_{lk}} = \frac{1}{2\epsilon}\Big\{\frac{\partial E}{\partial w_{ji}}(w_{lk}+\epsilon) - \frac{\partial E}{\partial w_{ji}}(w_{lk}-\epsilon)\Big\} + O(\epsilon^2)$$

由于只需要对W个权值施加扰动，且梯度可以通过$O(W)$次操作计算得到，因此可以在$O(W^2)$次操作内得到Hessian矩阵。

5.4.5 Hessian矩阵的精确计算

我们考虑一个具体的例子，即具有两层权值的网络，我们使用下标i, i'表示输入，下标j, j'表示隐含单元，用下标k, k'表示输出，我们定义：

$$\delta_k = \frac{\partial E_n}{\partial a_k}$$
$$M_{k,k'} = \frac{\partial^2 E_n}{\partial a_k \partial a_{k'}}$$

其中E_n是数据点n对误差函数的贡献，于是Hessian矩阵可以看作三个独立模块：

- 两个权值都在第二层。

$$\frac{\partial^2 E_n}{\partial w_{kj}^{(2)} \partial w_{k'j'}^{(2)}} = z_j z_{j'} M_{kk'}$$

- 两个权值都在第一层：

$$\frac{\partial^2 E_n}{\partial w_{ji}^{(1)} \partial w_{j'i'}^{(1)}} = x_i x_{i'} h''(a_{j'}) I_{jj'} \sum_k w_{kj'}^{(2)} \delta_k$$
$$+ x_i x_{i'} h'(a_{j'}) h'(a_j) \sum_k \sum_{k'} w_{k'j'}^{(2)} w_{kj}^{(2)} M_{kk'}$$

- 每一层一个权值：

$$\frac{\partial^2 E_n}{\partial w_{ji}^{(1)} \partial w_{kj'}^{(2)}} = x_i h'(a_j) \Big\{ \delta_k I_{j'j} + z_{j'} \sum_k w_{k'j}^{(2)} M_{kk'} \Big\}$$

其中$I_{jj'}$是单位矩阵的第j, j'个元素，如果权值中有偏置项，那么只需要将激活设为1即可得到相应的表达式。

5.4.6 Hessian矩阵的快速乘法

对于Hessian矩阵的许多应用，我们感兴趣的是H与某些向量v的乘积。可以注意到：

$$v^T H = v^T \nabla (\nabla E)$$

我们使用$\mathcal{R}\{.\}$表示算符$v^T \nabla$ 可以得到：

$$\mathcal{R}\{w\} = v$$

我们使用之前的网络结构和误差函数，考虑数据集里的一个模式对于误差函数的贡献，这样所要求解的向量可以通过求出每个模式各自贡献的求和得到，对于两层神经网络，正向传播方程为：

$$a_j = \sum_i w_{ji} x_i$$
$$z_j = h(a_j)$$

$$y_k = \sum_j w_{kj} z_j$$

使用$\mathcal{R}$作用于这些方程，得到正向传播方程

$$\mathcal{R}\{a_j\} = \sum_i v_{ji} x_i$$
$$\{z_j\} = h'(a_j)\mathcal{R}\{a_j\}$$
$$\mathcal{R}\{y_k\} = \sum_j w_{kj}\mathcal{R}\{z_j\} + \sum_j v_{kj} z_j$$

其中，v_{ji}是向量v中对应于权值w_{ji}的元素。
我们使用的是平方和误差函数，因此得到标准的反向传播表达式：

$$\delta_k = y_k - t_k$$
$$\delta_j = h'(a_j)\sum_k w_{kj}\delta_k$$

与之前一样，可以得到反向传播方程：

$$\mathcal{R}\{\delta_k\} = \mathcal{R}\{y_k\}$$
$$\mathcal{R}\{\delta_j\} = h''(a_j)\mathcal{R}\{a_j\}\sum_k w_{kj}\delta_k + h'(a_j)\sum_k v_{kj}\delta_k + h'(a_j)\sum_k w_{kj}\mathcal{R}\{\delta_k\}$$

最后有误差函数的一阶导数方程：

$$\frac{\partial E}{\partial w_{kj}} = \delta_k z_j$$
$$\frac{\partial E}{\partial w_{ji}} = \delta_j x_i$$

于是可以得到：

$$\mathcal{R}\{\frac{\partial E}{\partial w_{kj}}\} = \mathcal{R}\{\delta_k\}z_j + \delta_k\mathcal{R}\{z_j\}$$
$$\{\frac{\partial E}{\partial w_{ji}}\} = x_i\mathcal{R}\{\delta_j\}$$

所以对于每一个输入模式，这些量可以用上面的结果求出，$v^T H$的结果由上面两式得到，方法的好处是计算$v^T H$的方程和标准的正向传播和反向传播方程相同。

所以该方法可以计算完整的Hessian矩阵：将向量v选为一系列$(0, 0, \cdots, 1 \cdots, 0)$的单位向量，每一个单位向量选出Hessian矩阵的一列，但这种方法会损失一定的计算效率。

5.5 神经网络正则化

神经网络的输入和输出单元的数目是确定的，而隐含单元的数目M是一个自由参数，可以通过调节得到最优预测性能，在最大似然框架下，会得到一个泛华性能最优的M值。
泛化性能与M的关系不是简单的函数关系，因为误差函数中有局部最小值，为了控制神经网络的模型复杂度来避免过拟

合，一个方法是选择一个较大的M值，然后通过给误差函数添加一个正则化项来控制模型复杂度，最简单正则化形式是二次的，然后给出正则化误差函数，形式为：

$$\tilde{E}(w) = E(w) + \frac{\lambda}{2} w^T w$$

这个正则化项也叫权重衰减，模型复杂度可以通过选择正则化系数λ来确定，正则化项可以表示为权值w上的零均值的高斯先验分布的负对数。

5.5.1 相容的高斯先验

简单权值衰减的局限性是，它与网络映射的确定缩放性质不相容。我们考虑一个多层感知机网络，有两层权值和线性输出单元，给出了从输入变量集合$\{x_i\}$到输出变量集合$\{y_k\}$的映射，第一个隐含层的隐含单元的激活形式为：

$$z_j = h\Big(\sum_i w_{ji} x_i + w_{j0}\Big)$$

输出单元的激活为：

$$y_k = \sum_j w_{kj} z_j + w_{k0}$$

如果我们对输入变量进行一个线性变换：

$$x_i \to \tilde{x}_i = a x_i + b$$

然后根据映射对网络进行调节，使得网络映射不变。可以从输入单元到隐层单元的权值和偏置也进行一个对应的线性变换，形式为：

$$w_{ji} \to \tilde{w}_{ji} = \frac{1}{a} w_{ji}$$
$$w_{j0} \to \tilde{w}_{j0} = w_{j0} - \frac{b}{a} \sum_i w_{ji}$$

网络输出变量的线性变换为：

$$y_k \to \tilde{y}_k = c y_k + d$$

对第二层偏置和权值进行线性变换：

$$w_{kj} \to w_{kj} = c w_{kj}$$
$$w_{k0} \to \tilde{w}_{k0} = c w_{k0} + d$$

如果使用原始数据训练网络，并且使用线性变换的数据训练一个网络，则相容性要求两个网络是等价的，差别仅在于上面给出权值的线性变换。任意的正则化项都应该与该性质相容，否则模型会倾向于选择某个值，而忽视某个等价的解，由于简单的权值衰减把权值与偏置同等对待，因此不满足这个性质。
所以我们需要找一个对于权值的重新缩放不变，对于偏置的平移不变的正则化项，这样的正则化项：

$$\frac{\lambda_1}{2} \sum_{w \in W_1} w^2 + \frac{\lambda_2}{2} \sum_{w \in W_2} w^2$$

其中W_1表示第一层权重集合，W_2表示第二层的权重集合，偏置未出现，正则化项在权值的变换下不会发生变化，只要正则化参数进行下面的重新放缩即可：

$$\lambda_1 \to a^{\frac{1}{2}}\lambda_1, \lambda_2 \to c^{-\frac{1}{2}}\lambda_2$$

正则化项对应于下面形式的先验概率分布：

$$p(w|\alpha_1,\alpha_2) \propto \exp\Big(-\frac{\alpha_1}{2}\sum_{w\in W_1}w^2 - \frac{\alpha_2}{2}\sum_{w\in W_2}w^2 + \Big)$$

这样的先验形式是反常的，不能被归一化，因为偏置参数是没有限制的。使用反常的先验会给正则化系数选择带来很大的困难，给贝叶斯框架下的模型选择造成很大的困难，对应的模型证据等于0.因此通常做法是单独包含一个有着自己单独的一套超参数的一套超参数的偏置的先验。
更一般的，我们可以考虑权值被分为任意数量的组W_k情况下的先验，即：

$$p(w) \propto \exp\Big(-\frac{1}{2}\sum_k \alpha_k||w||_k^2\Big)$$

其中$||w||_k^2 = \sum_{j\in W_k} w_j^2$,作为该先验的特殊情况，如果将每一个输入单元关联的权值设为一个分组，并且关于对应参数α_k最优化边缘似然函数，就得到了自动确定方法

5.5.2 早停

非现行网络模型的训练对应着误差函数的迭代减小，误差函数关于训练数据集定义。对于许多用于网络训练的最优化算法，误差函数是一个关于迭代次数的不增函数。在独立数据上的测量误差通常先减小，然后过拟合后增大。于是训练过程可以在关于验证集误差最小的点停止，就得到一个泛化性能较小的限制复杂度的方式。
在这种情况下，网络行为有事可以通过增加网络自由度有效数量来定量描述。自由度有效数量开始时很小，然后训练时增加，对应于模型复杂度持续增加，训练误差达到最小值之前停止训练就得到了限制模型复杂度的方式。

5.5.3 不变性

在许多模式识别应用中，对于输入变量进行了一个或多个变换，预测不应该发生变化，或者说应该具有不变性。
使得可调节模型能表述所需要的不变性（如平移不变性，缩放不变性等）的方法：

- 复制训练模式，根据要求的不变性进行变换，对训练集进行拓展
- 为误差函数加上正则化项，用于惩罚当输入变换时，输出发生的改变。
- 抽取在要求的变换下不发生改变的特征，不变性被整合到预处理过程，任何后续使用这些特征作为输入的回归或者分类系统就会有不变性，这些特征要具有所需的不变性，还不能丢失对于判别有帮助的信息
- 将不变性的性质整合到神经网络的构建过程，或者对于相关向量机的方法，整合到核函数中。一种方法是使用局部接收场和共享权值，例如卷积神经网络

5.5.4 切线传播

通过切线传播的方法，我们可以使用正则化使模型对于输入的变换具有不变性。对于一个特定的输入向量x_n，假设变换是连续的，变换模式会扫过D维输入空间的流形$\mathcal{M}$,当D=2，假设变换由单一参数ξ为参数控制，则被x_n扫过的子空间$\mathcal{M}$是一维的，并且以ξ为参数。令这个变换作用于x_n上产生的向量为$s(x_n,\xi)$,且$s(x,0)=x$，曲线$\mathcal{M}$切线方向有方向导数$\tau = \frac{\partial x}{\partial \xi}$给出，$x_n$处切线向量为

$$\tau_n = \left.\frac{\partial s(x_n, \xi)}{\partial \xi}\right|_{\xi=0}$$

输入变量进行变换之后，输出也会变化，输出k关于ξ的导数为

$$\left.\frac{\partial y_k}{\partial \xi}\right|_{\xi=0} = \sum_{i=1}^{D} \left.\frac{\partial y_k}{\partial x_i}\frac{\partial x_i}{\partial \xi}\right|_{\xi=0} = \sum_{i=1}^{D} J_{ki}\tau_i$$

可以用来修改标准的误差函数，使得数据点的邻域内有不变性，也就是给原始的误差函数E增加一个正则化函数Ω,得到误差函数：

$$\tilde{E} = E + \lambda\Omega$$

其中正则化系数为λ,且

$$\Omega = \frac{1}{2}\sum_n\sum_k\left(\left.\frac{\partial y_{nk}}{\partial \xi}\right|_{\xi=0}\right)^2 = \frac{1}{2}\sum_n\sum_k\left(\sum_{i=1}^{D} J_{nki}\tau_{ni}\right)^2$$

当网络映射函数在每个模式向量的邻域内具有变换不变性的时候，正则化函数为0.
执行时，切线向量τ_n可以使用有限差近似。
若变换由L个参数控制，那么流形$\mathcal{M}$维度为L，对应正则化项由形如上面的公式求和得到，每个变换对应于每个变换分别具有不变性，则对于变换的组合就会具有不变性。
一个相关技术，被称作切线距离，可用来构造基于距离的方法的不变性。

5.5.5 用变换后的数据训练

对于平方和误差函数，对于未经变换的输入，误差函数可以写作：

$$E = \frac{1}{2}\iint\{y(x) - t\}^2 p(t|x)p(x)\mathrm{d}x\mathrm{d}t$$

考虑只有一个输出单元的网络，如果对每一个数据点的无穷个副本，每一个副本都有一个变换施加扰动，变换的参数为ξ，ξ服从概率分布$p(\xi)$，则拓展的误差函数为：

$$\tilde{E} = \frac{1}{2}\iiint\{y(s(x,\xi)) - t\}^2 p(t|x)p(x)p(\xi)\mathrm{d}x\mathrm{d}t\mathrm{d}\xi$$

假设$p(\xi)$均值为0，方差很小，即只考虑对原始输入变量小的变换，对于变换函数关于ξ进行展开，可以得到：

$$\begin{aligned} s(x,\xi) &= s(x,0) + \xi\left.\frac{\partial}{\partial\xi}s(x,\xi)\right|_{\xi=0} + \left.\frac{\xi^2}{2}\frac{\partial^2}{\partial\xi^2}\right|_{\xi=0} + O(\xi^3) \\ &= x + \xi\tau + \frac{1}{2}\xi^2\tau' + O(\xi^3) \end{aligned}$$

可以得到：

$$y(s(x,\xi)) = y(x) + \xi\tau^T\nabla y(x) + \frac{\xi^2}{2}[(\tau')^T\nabla y(x) + \tau^T\nabla\nabla y(x)\tau] + O(\xi^3)$$

代入平均误差函数，可以得到：

$$\begin{aligned}\tilde{E} &= \frac{1}{2}\iint\{y(x)-t\}^2 p(t|x)p(x)\mathrm{d}x\mathrm{d}t \\ &+ E[\xi]\iint\{y(x)-t\}\tau^T\nabla y(x)p(t|x)p(x)\mathrm{d}x\mathrm{d}t \\ &+ E[\xi^2]\iint\left[\{y(x)-t\}\{(\tau')^T\nabla y(x)+\tau^T\nabla\nabla y(x)\tau\}+(\tau^T\nabla y(x))^2\right]p(t|x)p(x)\mathrm{d}x\mathrm{d}t+O(\xi^3)\end{aligned}$$

由于变换分布均值为0，因此$E[\xi]=0$，将$E[\xi^2]$记作λ。省略$O(\xi^3)$，平均误差函数就成了：

$$\tilde{E}=E+\lambda\Omega$$

正则化项Ω的形式:

$$\Omega=\frac{1}{2}\int\left[\{y(x)-E[t|x]\}\left\{(\tau')^T\nabla y(x)+\tau\nabla\nabla y(x)\tau\right\}+(\tau^T\nabla y(x))^2\right]p(x)\mathrm{d}x$$

其中已经对t进行了积分。所以正则化误差函数等于非正则化误差函数加上一个$O(\xi^2)$的项，因此最小化总误差函数的网络函数的形式为：

$$y(x)=\mathbb{E}[t|x]+O(\xi^2)$$

所以正则化项的第一项消失，剩下的为：

$$\Omega=\frac{1}{2}\int(\tau^T\nabla y(x))^2p(x)\mathrm{d}x$$

这等价于切线传播的正则化项。
如果对于特殊情况，即输入变量的变换只是简单添加随机噪声，从而$x\to x+\xi$,所以正则化项的形式为：

$$\Omega=\frac{1}{2}\int||\nabla y(x)||^2p(x)\mathrm{d}x$$

这被称为**Tikhonov**正则化。

5.5 神经网络正则化

神经网络的输入和输出单元的数目是确定的，而隐含单元的数目M是一个自由参数，可以通过调节得到最优预测性能，在最大似然框架下，会得到一个泛华性能最优的M值。
泛化性能与M的关系不是简单的函数关系，因为误差函数中有局部最小值，为了控制神经网络的模型复杂度来避免过拟合，一个方法是选择一个较大的M值，然后通过给误差函数添加一个正则化项来控制模型复杂度，最简单正则化形式是二次的，然后给出正则化误差函数，形式为：

$$\tilde{E}(w)=E(w)+\frac{\lambda}{2}w^Tw$$

这个正则化项也叫权重衰减，模型复杂度可以通过选择正则化系数λ来确定，正则化项可以表示为权值w上的零均值的高斯先验分布的负对数。

5.5.1 相容的高斯先验

简单权值衰减的局限性是，它与网络映射的确定缩放性质不相容。我们考虑一个多层感知机网络，有两层权值和线性输出单元，给出了从输入变量集合$\{x_i\}$到输出变量集合$\{y_k\}$的映射，第一个隐含层的隐含单元的激活形式为：

$$z_j = h\Big(\sum_i w_{ji}x_i + w_{j0}\Big)$$

输出单元的激活为：

$$y_k = \sum_j w_{kj}z_j + w_{k0}$$

如果我们对输入变量进行一个线性变换：

$$x_i \to \tilde{x}_i = ax_i + b$$

然后根据映射对网络进行调节，使得网络映射不变。可以从输入单元到隐层单元的权值和偏置也进行一个对应的线性变换，形式为：

$$w_{ji} \to \tilde{w}_{ji} = \frac{1}{a}w_{ji}$$
$$w_{j0} \to \tilde{w}_{j0} = w_{j0} - \frac{b}{a}\sum_i w_{ji}$$

网络输出变量的线性变换为：

$$y_k \to \tilde{y}_k = cy_k + d$$

对第二层偏置和权值进行线性变换：

$$w_{kj} \to w_{kj} = cw_{kj}$$
$$w_{k0} \to \tilde{w}_{k0} = cw_{k0} + d$$

如果使用原始数据训练网络，并且使用线性变换的数据训练一个网络，则相容性要求两个网络是等价的，差别仅在于上面给出权值的线性变换。任意的正则化项都应该与该性质相容，否则模型会倾向于选择某个值，而忽视某个等价的解，由于简单的权值衰减把权值与偏置同等对待，因此不满足这个性质。
所以我们需要找一个对于权值的重新缩放不变，对于偏置的平移不变的正则化项，这样的正则化项：

$$\frac{\lambda_1}{2}\sum_{w\in W_1} w^2 + \frac{\lambda_2}{2}\sum_{w\in W_2} w^2$$

其中W_1表示第一层权重集合，W_2表示第二层的权重集合，偏置未出现，正则化项在权值的变换下不会发生变化，只要正则化参数进行下面的重新放缩即可：

$$\lambda_1 \to a^{\frac{1}{2}}\lambda_1, \lambda_2 \to c^{-\frac{1}{2}}\lambda_2$$

正则化项对应于下面形式的先验概率分布：

$$p(w|\alpha_1,\alpha_2) \propto \exp\Big(-\frac{\alpha_1}{2}\sum_{w\in W_1} w^2 - \frac{\alpha_2}{2}\sum_{w\in W_2} w^2 + \Big)$$

这样的先验形式是反常的，不能被归一化，因为偏置参数是没有限制的。使用反常的先验会给正则化系数选择带来很大的困难，给贝叶斯框架下的模型选择造成很大的困难，对应的模型证据等于0.因此通常做法是单独包含一个有着自己单独的

一套超参数的一套超参数的偏置的先验。
更一般的，我们可以考虑权值被分为任意数量的组W_k情况下的先验，即：

$$p(w) \propto \exp\Big(-\frac{1}{2}\sum_k \alpha_k ||w||_k^2\Big)$$

其中$||w||_k^2 = \sum_{j\in W_k} w_j^2$,作为该先验的特殊情况，如果将每一个输入单元关联的权值设为一个分组，并且关于对应参数α_k最优化边缘似然函数，就得到了自动确定方法

5.5.2 早停

非现行网络模型的训练对应着误差函数的迭代减小，误差函数关于训练数据集定义。对于许多用于网络训练的最优化算法，误差函数是一个关于迭代次数的不增函数。在独立数据上的测量误差通常先减小，然后过拟合后增大。于是训练过程可以在关于验证集误差最小的点停止，就得到一个泛化性能较小的限制复杂度的方式。
在这种情况下，网络行为有事可以通过增加网络自由度有效数量来定量描述。自由度有效数量开始时很小，然后训练时增加，对应于模型复杂度持续增加，训练误差达到最小值之前停止训练就得到了限制模型复杂度的方式。

5.5.3 不变性

在许多模式识别应用中，对于输入变量进行了一个或多个变换，预测不应该发生变化，或者说应该具有不变性。
使得可调节模型能表述所需要的不变性（如平移不变性，缩放不变性等）的方法：

- 复制训练模式，根据要求的不变性进行变换，对训练集进行拓展
- 为误差函数加上正则化项，用于惩罚当输入变换时，输出发生的改变。
- 抽取在要求的变换下不发生改变的特征，不变性被整合到预处理过程，任何后续使用这些特征作为输入的回归或者分类系统就会有不变性，这些特征要具有所需的不变性，还不能丢失对于判别有帮助的信息
- 将不变性的性质整合到神经网络的构建过程，或者对于相关向量机的方法，整合到核函数中。一种方法是使用局部接收场和共享权值，例如卷积神经网络

5.5.4 切线传播

通过切线传播的方法，我们可以使用正则化使模型对于输入的变换具有不变性。对于一个特定的输入向量x_n，假设变换是连续的，变换模式会扫过D维输入空间的流形$\mathcal{M}$,当D=2，假设变换由单一参数ξ为参数控制，则被x_n扫过的子空间$\mathcal{M}$是一维的，并且以ξ为参数。令这个变换作用于x_n上产生的向量为$s(x_n,\xi)$,且$s(x,0)=x$，曲线$\mathcal{M}$切线方向有方向导数$\tau = \frac{\partial x}{\partial \xi}$给出，$x_n$处切线向量为

$$\tau_n = \left.\frac{\partial s(x_n,\xi)}{\partial \xi}\right|_{\xi=0}$$

输入变量进行变换之后，输出也会变化，输出k关于ξ的导数为

$$\left.\frac{\partial y_k}{\partial \xi}\right|_{\xi=0} = \sum_{i=1}^{D}\left.\frac{\partial y_k}{\partial x_i}\frac{\partial x_i}{\partial \xi}\right|_{\xi=0} = \sum_{i=1}^{D} J_{ki}\tau_i$$

可以用来修改标准的误差函数，使得数据点的邻域内有不变性，也就是给原始的误差函数E增加一个正则化函数Ω,得到误差函数：

$$\tilde{E} = E + \lambda\Omega$$

其中正则化系数为λ,且

$$\Omega = \frac{1}{2}\sum_n\sum_k\Big(\frac{\partial y_{nk}}{\partial\xi}\Big|_{\xi=0}\Big)^2 = \frac{1}{2}\sum_n\sum_k\Big(\sum_{i=1}^{D}J_{nki}\tau_{ni}\Big)^2$$

当网络映射函数在每个模式向量的邻域内具有变换不变性的时候，正则化函数为0.
执行时，切线向量τ_n可以使用有限差近似。
若变换由L个参数控制，那么流形$\mathcal{M}$维度为L，对应正则化项由形如上面的公式求和得到，每个变换对应于每个变换分别具有不变性，则对于变换的组合就会具有不变性。
一个相关技术，被称作切线距离，可用来构造基于距离的方法的不变性。

5.5.5 用变换后的数据训练

对于平方和误差函数，对于未经变换的输入，误差函数可以写作：

$$E = \frac{1}{2}\iint\{y(x)-t\}^2p(t|x)p(x)\mathrm{d}x\mathrm{d}t$$

考虑只有一个输出单元的网络，如果对每一个数据点的无穷个副本，每一个副本都有一个变换施加扰动，变换的参数为ξ，ξ服从概率分布$p(\xi)$，则拓展的误差函数为：

$$\tilde{E} = \frac{1}{2}\iiint\{y(s(x,\xi))-t\}^2p(t|x)p(x)p(\xi)\mathrm{d}x\mathrm{d}t\mathrm{d}\xi$$

假设$p(\xi)$均值为0，方差很小，即只考虑对原始输入变量小的变换，对于变换函数关于ξ进行展开，可以得到：

$$\begin{aligned}s(x,\xi) &= s(x,0) + \xi\frac{\partial}{\partial\xi}s(x,\xi)\Big|_{\xi=0} + \frac{\xi^2}{2}\frac{\partial^2}{\partial\xi^2}\Big|_{\xi=0} + O(\xi^3)\\ &= x + \xi\tau + \frac{1}{2}\xi^2\tau' + O(\xi^3)\end{aligned}$$

可以得到：

$$y(s(x,\xi)) = y(x) + \xi\tau^T\nabla y(x) + \frac{\xi^2}{2}[(\tau')^T\nabla y(x) + \tau^T\nabla\nabla y(x)\tau] + O(\xi^3)$$

代入平均误差函数，可以得到：

$$\begin{aligned}\tilde{E} = &\frac{1}{2}\iint\{y(x)-t\}^2p(t|x)p(x)\mathrm{d}x\mathrm{d}t\\ &+ E[\xi]\iint\{y(x)-t\}\tau^T\nabla y(x)p(t|x)p(x)\mathrm{d}x\mathrm{d}t\\ &+ E[\xi^2]\iint\Big[\{y(x)-t\}\{(\tau')^T\nabla y(x) + \tau^T\nabla\nabla y(x)\tau\} + (\tau^T\nabla y(x))^2\Big]p(t|x)p(x)\mathrm{d}x\mathrm{d}t + O(\xi^3)\end{aligned}$$

由于变换分布均值为0，因此$E[\xi] = 0$，将$E[\xi^2]$记作λ。省略$O(\xi^3)$，平均误差函数就成了：

$$\tilde{E} = E + \lambda\Omega$$

正则化项Ω的形式:

$$\Omega = \frac{1}{2}\int \Big[\{y(x) - E[t|x]\}\big\{(\tau')^T\nabla y(x) + \tau\nabla\nabla y(x)\tau\big\} + (\tau^T\nabla y(x))^2\Big]p(x)\mathrm{d}x$$

其中已经对t进行了积分。所以正则化误差函数等于非正则化误差函数加上一个$O(\xi^2)$的项，因此最小化总误差函数的网络函数的形式为：

$$y(x) = \mathbb{E}[t|x] + O(\xi^2)$$

所以正则化项的第一项消失，剩下的为：

$$\Omega = \frac{1}{2}\int (\tau^T\nabla y(x))^2 p(x)\mathrm{d}x$$

这等价于切线传播的正则化项。
如果对于特殊情况，即输入变量的变换只是简单添加随机噪声，从而$x \to x + \xi$,所以正则化项的形式为：

$$\Omega = \frac{1}{2}\int ||\nabla y(x)||^2 p(x)\mathrm{d}x$$

这被称为**Tikhonov**正则化。

5.6 混合密度网络

有监督学习的目标是对$p(t|x)$建模，对于简单的回归问题，这个分布被选中为高斯分布，然而实际任务中常会遇到与高斯分布差别相当大的分布。例如，在逆问题中概率分布可以是多峰的，这时采用高斯分布会产生相当差的预测结果。
我们寻找一个对条件概率密度建模的一般框架：为$p(t|x)$使用一个混合模型，模型混合系数和每一个分量的概率分布都是输入向量x的灵活的函数，这就构成了混合密度网络。对于任意给定的x，混合模型提供了通用模型，对任意条件概率密度$p(t|x)$进行建模。如果使用一个足够灵活的网络，则可以得到一个近似的任意条件概率密度框架。
我们显式令模型分量为高斯分布，得到：

$$p(t|x) = \sum_{k=1}^{K}\pi_k(x)\mathcal{N}(t|\mu_k(x), \sigma_k^2(x)I)$$

这是异方差模型的一个例子，因为数据中噪声方差时输入向量x的一个函数。同样也可以使用其他分布，比如如果目标变量是二值的而不是连续的，则我们可以使用伯努利分布。
混合密度网络使用相同的函数来预测所有分量概率分布的参数以及混合参数，因此非线性隐含单元被依赖于输入的函数共享。
对于两层的神经网络，网络有S形（双曲正切）隐含单元。如果混合模型有K个分量，且t有L个分量，网络有K个输出单元激活a_k^π，确定混合系数$\pi_k(x)$，K个输出a_k^σ,确定核宽度$\sigma_k(x)$,有$K \times L$个输出记作 a_{kj}^μ,确定核中心$\mu_k(x)$的分量$\mu_{kj}(x)$，网络输出总数$(L+2)K$， 这与通常的网络的L 个输出不同。通常的网络只是简单预测目标变量的条件均值。
混合系数要满足以下性质：

$$\sum_{k=1}^{K}\pi_k(x) = 1,\ 0 \le \pi_k(x) \le 1$$

可以通过一组**softmax**输出来实现

$$\pi_k(x) = \frac{\exp(a_k^\pi)}{\sum_{l=1}^K \exp(a_l^\pi)}$$

类似的，方差要满足$\sigma_k^2(x) \geq 0$,因此可以使用对应网络激活的指数形式：

$$\sigma_k(x) = \exp(a_k^\pi)$$

由于均值$\mu_k(x)$有实数分量，因此可以直接用激活表示：

$$\mu_{kj}(x) = a_{kj}^\mu$$

混合密度网络的可调节参数由权向量和偏置组成，可以通过最大似然法确定，或者可以使用最小化误差函数确定，对于独立数据，误差函数形式为：

$$E(w) = -\sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k(x_n, w) \mathcal{N}(t_n | \mu_k(x_n, w), \sigma_k^2(x_n, w) I) \right\}$$

然后计算误差函数对于w分量的导数，可以通过反向传播算法计算误差函数关于w分量的导数。
可以将混合系数$\pi_k(x)$看作是与x相关的先验概率分布，从而引入对应的后验概率，形式为：

$$\gamma_{nk} = \gamma_k(t_n|x_n) = \frac{\pi_k \mathcal{N}_{nk}}{\sum_{l=1}^K \pi_l \mathcal{N}_{nl}}$$

关于控制混合系数的网络输出激活函数的导数是：

$$\frac{\partial E_n}{\partial a_k^\pi} = \pi_k - \gamma_{nk}$$

控制分量均值的网络输出激活的导数为：

$$\frac{\partial E_n}{\partial a_{kl}^\mu} = \gamma_k \left\{ \frac{\mu_{kl} - t_{nl}}{\sigma_k^2} \right\}$$

控制方差的网络激活导数为：

$$\frac{\partial E_n}{\partial a_k^\sigma} = \gamma_{nk} \left\{ L - \frac{||t_N - \mu_k||^2}{\sigma_k^2} \right\}$$

通过调整混合分布$\pi_k(x)$的大小，模型能产生一个对于某些x是单峰的，对其他的x是多峰的概率分布。
训练结束之后，就可以预测对于任意给定的输入向量的目标数据的条件密度函数。该概率密度可以完整描述生成数据的概率分布，用此可以计算不同应用中更感兴趣的具体的量，一个简单的量就是目标数据的条件均值：

$$\mathbb{E}[t|x] = \int t p(t|x) \mathrm{d}t = \sum_{k=1}^K \pi_k(x) \mu_k(x)$$

由于使用最小平方训练标准的神经网络近似了条件均值，因此混合密度网络可以复制传统的最小平方结果，对于多峰分布，条件均值是一个受限的值。
同样也可以计算密度函数的方差：

$$s^2(x) = \mathbb{E}[||t - \mathbb{E}[t|x]||^2|x] = \sum_{k=1}^{K} \pi_k(x) \left\{ \sigma_k^2(x) + ||\mu_k(x) - \sum_{l=1}^{K} \pi_l(x)\mu_l(x)||^2 \right\}$$

与最小平方结果相比，这个结果更加一般，因为方差是x的一个函数
对于多峰分布使用条件均值描述数据效果很差，条件众数可能更有价值，由于条件均值没有解析解，因此需要数值迭代。简单方法是取x的对应最可能分量的均值。

5.7 贝叶斯神经网络

我们使用最大似然方法确定网络参数，震泽化最大似然方法可看做MAP方法，正则化项可以被看作先验参数分布的对数，在贝叶斯方法中需要对参数的概率分布进行积分或者求和。
后验概率分布可精确计算，预测分布具有解析解。多层神经网络中，网络函数对于参数值的高度非线性意味着精确的贝叶斯方法不可行。后验概率分布的对数是非凸的，对应于误差函数中的多个局部极小值。最完整的贝叶斯方法基于拉普拉斯近似的方法，使用真实的后验概率的众数为中心的高斯分布来近似后验概率分布，假设高斯分布协方差很小，网络函数关于参数空间的区域中参数近似是线性关系。
首先考虑回归问题。

5.7.1 后验参数分布

假设条件概率分布$p(t|x)$是高斯分布，均值与x有关，有输出$y(x,w)$确定，精度β为：

$$p(t|x,w,\beta) = \mathcal{N}(t|y(x,w),\beta^{-1})$$

似然函数为：

$$p(D|w,\beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|y(x_n,w),\beta^{-1})$$

所以最终后验概率

$$p(w|D,\alpha,\beta) \propto p(w|\alpha)p(D|w,\beta)$$

由于$y(x,w)$与w的关系是非线性的，所以后验概率不是高斯分布。
首先找到后验概率分布的最大值，先最大化后验概率分布的对数：

$$\ln p(w|D) = -\frac{\alpha}{2}w^T w - \frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n,w) - t_n\}^2 + \text{常数}$$

对应于正则化的平方和误差函数。假设α,β都是固定的，则可以计算导数，找到后验概率的最大值，将最大值的位置记作w_{MAP}.
然后计算后验概率分布的负对数的二阶导数，建立局部高斯近似，负对数后验概率的二阶导数：

$$A = -\nabla\nabla \ln p(w|D,\alpha,\beta) = \alpha I + \beta H$$

后验概率对应的高斯近似形式为：

$$p(w|D) = \mathcal{N}(w|w_{MAP},A^{-1})$$

预测分布可以积分得到：

$$p(t|x,D)=\int p(t|x,w)q(w|D)\mathrm{d}w$$

由于对后验概率分布高斯近似，积分无法得到解析解，假设与$y(x,w)$发生变化造成的w变化幅度相比，后验概率分布的方差较小，可以在w_{MAP}附近对于网络函数进行泰勒展开，可以得到：

$$y(x,w)\simeq y(x,w_{MAP})+g^T(w-w_{MAP})$$
$$g=\nabla_w y(x,w)\Big|_{w=w_{MAP}}$$

使用该近似，得到一个线性高斯模型，$p(w)$是高斯分布，$p(t|w)$也是高斯分布，均值为w的线性函数，分布形式为：

$$p(t|x,w,\beta)\simeq\mathcal{N}(t|y(x,w_{MAP})+g^T(w-w_{MAP}),\beta^{-1})$$

于是对于边缘分布$p(t)$；

$$p(t|x,D,\alpha,\beta)=\mathcal{N}(t|y(x,w_{MAP}),\sigma^2(x))$$

与输入相关的方差为：

$$\sigma^(x)=\beta^{-1}+g^TA^{-1}g$$

可以看到方差的第一项来自目标变量的固有噪声，第二项与x相关，表示由w的不确定性造成内插的不确定性。

5.7.2 超参数最优化

超参数的边缘似然函数或者是模型证据可通过对于网络权值进行积分得到：

$$p(D|\alpha,\beta)=\int p(D|w,\beta)p(w|\alpha)\mathrm{d}w$$

使用拉普拉斯近似结果，积分后取对数，得到：

$$\ln p(D|\alpha,\beta)\simeq -E(w_{MAP})-\frac{1}{2}\ln|A|+\frac{W}{2}\ln\alpha+\frac{N}{2}\ln\beta-\frac{N}{2}\ln(2\pi)$$

正则化误差函数定义为：

$$E(w_{MAP})=\frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n,w_{MAP})-t_n\}^2+\frac{\alpha}{2}w_{MAP}^2w_{MAP}$$

定义特征值方程：

$$\beta Hu_i=\lambda_iu_i$$

其中H是在$w=w_{MAP}$处计算的Hessian矩阵，由平方和误差函数的二阶导数组成，我们由第三章可以得到：

$$\alpha=\frac{\gamma}{w_{MAP}^Tw_{MAP}}$$

其中γ表示参数的有效数量，定义为：

$$\gamma = \sum_{i=1}^{W} \frac{\lambda_i}{\alpha + \lambda_i}$$

对于非线性神经网络，α的改变会引起Hessian矩阵H的改变，进而改变特征值，于是隐式忽略了涉及到λ_i关于α的项。
类似的，关于β的最大化模型证据，得到重估计公式：

$$\frac{1}{\beta} = \frac{1}{N-\gamma} \sum_{n=1}^{N} \{y(x_n, w_{MAP}) - t_n\}^2$$

所以可以交替进行超参数α和β的重新估计和后验概率分布的更新。对于神经网络，由于后验概率分布的多峰性质，情况更加复杂。所以最终得到的w_{MAP}将依赖于w的初始值。
为比较不同的模型，需要计算模型证据$p(D)$,可以将最优化得到的α, β带入到

$$lnp(D|\alpha, \beta) \simeq -E(w_{MAP}) - \frac{1}{2}\ln|A| + \frac{W}{2}\ln\alpha + \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi)$$

得到模型证据的近似。更仔细的方法是关于α, β进行积分，同时使用高斯近似。拉普拉斯近似基于的是权值的后验概率分布的众数附近的局部二次展开。在5.1.1节，我们已经看到，在两层神经网络中，任意给定的众数都是$M!2^M$ 个等价的众数中的一个，这些等价的众数由网络的互换对称性和符号对称性造成，其中M 是隐含结点的数量。当比较具有不同隐含结点数量的网络时，通过将模型证据乘以因子$M!2^M$ ，就可以考虑到这一点。

5.7.3 用于分类的贝叶斯神经网络

对一个二类问题，使用 **logistic sigmod** 输出，多类时使用 **softmax** 输出。
模型的对数似然函数为：

$$\ln p(D|w) = \sum_{n=1}^{N} \{t_n \ln y_n + (1-t_n)\ln(1-y_n)\}$$

这里没有超参数β，因为假定数据点被正确标记，先验分布是各向同性高斯分布

$$p(w|\alpha) = \mathcal{N}(w|0, \alpha^{-1}I)$$

将拉普拉斯框架用于这个模型的第一个阶段是初始化超参数α，然后通过最大化对数后验概率分布确定参数w，等价于最小化正则化误差函数：

$$E(w) = -\ln p(D|w) + \frac{\alpha}{2} w^T w$$

找到w_{MAP}之后，之后可以精确计算负对数似然函数的二阶导数组成的Hessian矩阵H。
为了优化超参数α，再次最大化边缘似然函数，形式为：

$$\ln p(D|\alpha) \simeq -E(w_{MAP}) - \frac{1}{2}\ln|A| + \frac{W}{2}\ln\alpha$$

正则化误差函数为：

$$E(w_{MAP}) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1-t_n)\ln(1-y_n)\} + \frac{\alpha}{2} w_{MAP}^T w_{MAP}$$

再找预测分布的时候，由于网络函数的非线性的性质，积分无法正确计算。简单的近似方法是假设后验概率非常窄，因此可进行近似：

$$p(t|x,D) \simeq p(t|x,w_{MAP})$$

我们对输出激活函数进行线性近似:

$$a(x,w) = a_{MAP}(x) + b^T(w - w_{MAP})$$

其中向量$b = \nabla a(x,w_{MAP})$。
由于对w的后验概率分布进行了高斯近似，a的模型是w的线性函数，因此神经网络权值分布引出的输出单元激活函数的值的分布：

$$p(a|x,D) = \int \delta(a - a_{MAP}(x) - b^T(x)(w - w_{MAP}))q(w|D)\mathrm{d}w$$

其中$q(w|D)$是对高斯概率分布的高斯近似。我们看到该分布是高斯分布，均值为$a_{MAP} = a(x,w_{MAP})$,方差为:

$$\sigma_a^2(x) = b^T(x)A^{-1}b(x)$$

为了得到预测分布，对于a进行积分，得到：

$$p(t=1|x,D) = \int \sigma(a)p(a|x,D)\mathrm{d}a$$

高斯分布与**logistic sigmod**函数的卷积无法计算，于是由于使用logistic sigmod 函数于高斯的卷积近似：
得到：

$$p(t=1|x,D) = \sigma(\kappa(\sigma_a^2)a_{MAP})$$