异常点检测算法（三）

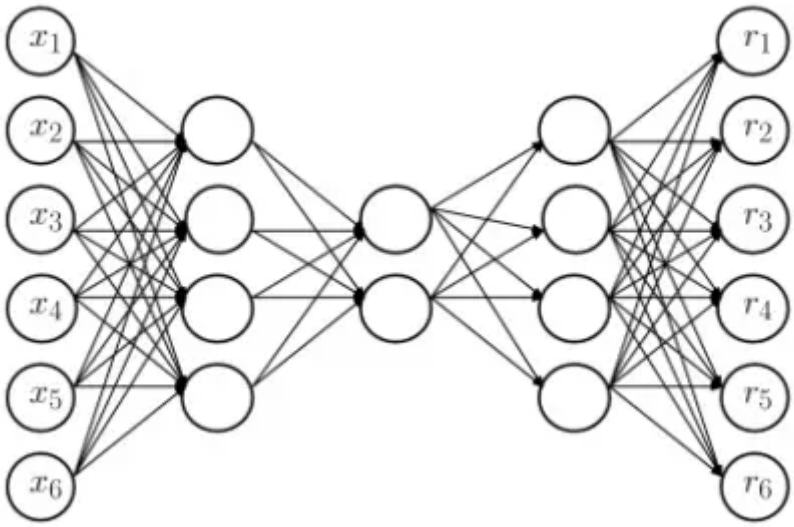
原创 2016-08-03 张戎 数学人生

异常值检测算法在数据挖掘的诸多领域有着应用场景，例如金融领域，信息传输领域，图像领域等。在研究过程中，有学者给出了异常点的一个定义：

An outlier is an observation that deviates so much from other observations as as to arouse suspicion that it was generated by a different mechanism.

RNN 算法的主要思想

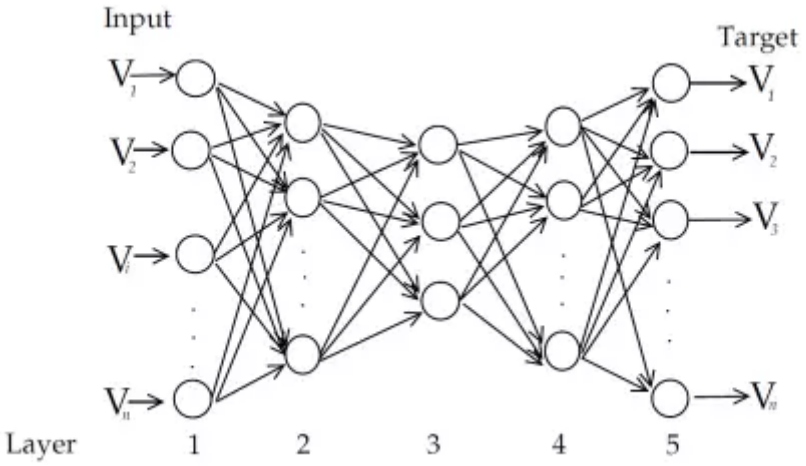
在这篇文章中，我们将会介绍一个多层的前馈神经网络，该神经网络可以用来进行异常值的检测。这个神经网络模拟的是一个恒等映射，输入层的神经元个数和输出层的神经元个数是一样的。这类的神经网络被称为 Replicator Neural Networks (RNNs)，请注意这里的 RNN 算法指的并不是 Recurrent Neural Networks（RNNs），而是 Replicator Neural Networks，尽管它们拥有着同样的缩写名字 RNNs。具体来说， Replicator Neural Networks (RNNs)，或者说自编码器，是一个多层前馈的神经网络 (multi-layer feed-forward neural networks)。在 Replicator Neural Networks 中，输入的变量也是输出的变量，模型中间层节点的个数少于输入层和输出层节点的个数。这样的话，模型就起到了压缩数据和恢复数据的作用。



如图所示，这里的 RNNs 有三个隐藏层，输入层和输出层的节点个数都是6，第一个隐藏层和第三个隐藏层的节点个数（图中是4个节点）少于输入层，第二个隐藏层的节点个数是最少的（图中是2个节点）。在神经网络传输的时候，中间使用了 tanh 函数和 sigmoid 函数。这个神经网络是训练一个从输入层到输出层的恒等函数（identity mapping），传输的时候从输入层开始压缩数据，然后到了第二个隐藏层的时候开始解压数据。训练的目标就是使得整体的输出误差足够小，整体的误差是由所有的样本误差之和除以样本的个数得到的。由于图中只画出了6个特征，因此第 i 个样本的误差是

如果使用已经训练好的 RNN 模型，异常值的分数就可以定义为重构误差（reconstruction error）。

下面简要介绍一下 RNN 模型是如何构建的：



根据上图所示，左边的是输入层，右边的输出层。假设第 k 层中第 i 个神经元的输出是 $S_k(I_{ki})$，其中 I_{ki} 表示第 k 层中第 i 个神经元的输入，S_k 表示第 k 层使用的激活函数。那么

$$\theta = I_{ki} = \sum_{j=0}^{L_{k-1}} w_{kij} Z_{(k-1)j}$$

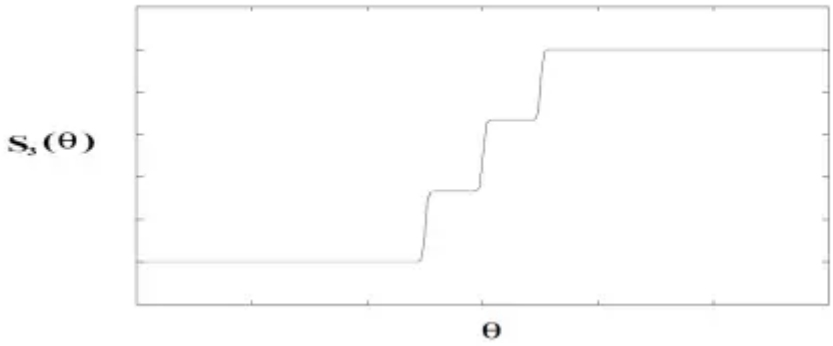
其中 Z_{kj} 是第 k 层中第 j 个神经元的输出，L_k 是第 k 层神经元的个数。对于第二层和第四层而言 (k=2,4)，激活函数选择为

$$S_k(\theta) = tanh(a_k\theta) \text{ for } k = 2 \text{ or } 4,$$

这里的 a_k 是一个参数，通常假设为1。对于中间层 (k=3) 而言，激活函数是一个类阶梯 (step-like) 函数。有两个参数 N 和 a_3，N 表示阶梯的个数，a_3 表示从这一层到下一层的提升率 (transition rate)：

$$S_3(\theta) = \frac{1}{2}+\frac{1}{4}\sum_{j=1}^{N-1} tanh(a_3(\theta - \frac{j}{N})).$$

在这里可以假设 $a_3 = 100$，$N = 4$. 那么 $S_3(\theta)$ 就如下图所示。



第三层的激活函数的输出就变成了 N 个离散的变量：0, 1/(N-1), 2/(N-1),...,1。这个阶梯型的激活函数是把第三层的连续输入值变成了一批离散的值。也就意味着把样本映射到了 N 个簇，那么 RNN 就可以计算出单个的异常点和一小簇的异常点。

备注：

根据上面的分析，可以看出如果按照以上算法，则不能使用反向传播算法来训练模型，原因是 的导数不能够通过它的取值来表示。这一点与 tanh 函数， 函数是不一致的，因为 和 。因此有学者指出 [1]，使用三个隐藏层是没有必要的，使用1个或者2个隐藏层的神经网络也能够得到类似的结果；同样，没有必要使用 这样类型的阶梯函数，使用传统的 激活函数也能够得到类似的结果。并且 是一个 step-like 函数，很多地方的导数取值都是接近于零的。

后向传播算法：

一般来说，为了训练神经网络模型，需要使用后向传播算法（back propagation），也简称为 BP 算法，或者误差逆传播算法（error back propagation）。在本文中，仅针对最简单的 RNN 模型介绍如何使用 BP 算法进行模型训练，至于多层的神经网络模型或者其他的神经网络模型，方法则是完全类似的。

给定训练集合 ，其中有 m 个样本，并且输入和输出是一样的值。换句话说，也就是 n 维向量

.

换句话说，输入样例是由 n 个属性描述，输出的结果也是 n 个属性。隐藏层只有一个，隐藏层的神经元个数是 q=(n+1)/2，这里的 [] 表示 Gauss 取整函数。输出层第 j 个神经元的阈值使用 表示，隐藏层第 h 个神经元的阈值使用 表示。输入层第 i 个神经元与隐藏层第 h 个神经元之间的连接权重是 ，隐藏层第 h 个神经元与输出层第 j 个神经元之间的连接权重是 其中

记隐藏层第 h 个神经元接收到的输入为

写成矩阵形式就是：

记输出层第 j 个神经元接收到的输入为

$$\beta_j = \sum_{h=1}^{q} w_{hj} b_h \text{ for all } 1 \leq j \leq n,$$

其中 b_h 是隐藏层第 h 个神经元的输出，$b_h = f(\alpha_h - \gamma_h) \text{ for all } 1 \le h \le q$, f 是激活函数。写成矩阵形式就是：

$$(\beta_1, \cdots, \beta_n) = (b_1, \cdots, b_q) \begin{bmatrix} w_{11} & \dots & w_{1n} \\ \dots & \dots & \dots \\ w_{q1} & \dots & w_{qn} \end{bmatrix}.$$

输出层第 j 个神经元的输出是 $f(\beta_j - \theta_j)$, 其中 $1 \le j \le n$.

下面可以假定激活函数都使用 $f(x) = 1/(1\exp(-x))$, 那么直接通过导数计算可以得到 $f'(x) = f(x)(1 - f(x))$.

对于训练集 $(\mathbf{x}_k, \mathbf{y}_k)$, 通过神经网络得到的输出是 $\hat{\mathbf{y}}_k = (\hat{y}_{k1}, \ldots, \hat{y}_{kn})$,并且 $\hat{y}_{kj} = f(\beta_j - \theta_j)$ 对于 $1 \le j \le n$ 都成立。那么神经网络在训练集 $(\mathbf{x}_k, \mathbf{y}_k)$ 的均方误差是

$$E_k = \frac{1}{2}\sum_{j=1}^{n}(\hat{y}_{kj} - y_{kj})^2,$$

其中 $\mathbf{y}_k = (y_{k1}, \ldots, y_{kn})$. 整体的误差是

$$E = \frac{1}{m}\sum_{k=1}^{m} E_k = \frac{1}{2m}\sum_{k=1}^{m}\sum_{j=1}^{n}(\hat{y}_{kj} - y_{kj})^2$$

标准 BP 算法：

网络中有 个参数需要确定：输入层到隐藏层的 n*q 个权重值，隐藏层到输出层的 n*q 个权重值，q个隐层神经元的阈值，n 个输出层神经元的阈值。BP 算法是一个迭代学习算法，在迭代的每一轮采用了梯度下降法来进行参数的更新。任意参数的更新规则是

+

标准 BP 算法是根据每一个 来获得更新规则，下面来推导每一个参数的更新规则。对于 计算梯度

注意到 先影响到第 j 个输出层神经元的输入值 再影响到第 j 个输出层神经元的输出值 ，最后影响到 ，根据高等数学的链式法则可以得到

根据定义 可以得到 对于 都成立。

根据定义 可以得到 .

根据定义 和 可以得到

所以可以计算出对于 有

$$\frac{\partial E_k}{\partial w_{hj}} = (\hat{y}_{kj} - y_{kj}) \cdot \hat{y}_{kj} \cdot (1 - \hat{y}_{kj}) \cdot b_h$$

如果假设

$$g_j = -\frac{\partial E_k}{\partial \beta_j} = -\frac{\partial E_k}{\partial \hat{y}_{kj}} \cdot \frac{\hat{y}_{kj}}{\partial \beta_j}$$

那么可以得到

$$g_j = \hat{y}_{kj} \cdot (1 - \hat{y}_{kj}) \cdot (y_{kj} - \hat{y}_{kj})$$

因此对于 $1 \leq h \leq q, 1 \leq j \leq n,$ 可以得到$\Delta w_{hj} = \eta g_j b_h.$

根据类似的想法，有

$$\Delta\theta_j = -\eta \cdot \frac{\partial E_k}{\partial \theta_j}, \Delta v_{ih} = -\eta \cdot \frac{\partial E_k}{\partial v_{ih}}, \Delta\gamma_h = -\eta \cdot \frac{\partial E_k}{\partial \gamma_h}.$$

逐个计算：

$$\frac{\partial E_k}{\partial \theta_j} = \frac{\partial E_k}{\partial \hat{y}_{kj}} \cdot \frac{\partial \hat{y}_{kj}}{\partial \theta_j} = (\hat{y}_{kj} - y_{kj}) \cdot (-1) \cdot f'(\beta_j - \theta_j) = (y_{kj} - \hat{y}_{kj}) \cdot \hat{y}_{kj} \cdot (1 - \hat{y}_{kj}) = g_j$$

$$\frac{\partial E_k}{\partial v_{ih}} = \frac{\partial E_k}{\partial \alpha_h} \cdot \frac{\partial \alpha_h}{\partial v_{ih}} = \frac{\partial E_k}{\partial b_h} \cdot \frac{\partial b_h}{\partial \alpha_h} \cdot \frac{\partial \alpha_h}{\partial v_{ih}}$$

由于

$$\frac{\partial \alpha_h}{\partial v_{ih}} = x_{ki}$$

$$\frac{\partial b_h}{\partial \alpha_h} = f'(\alpha_h - \gamma_h) = f(\alpha_h - \gamma_h) \cdot (1 - f(\alpha_h - \gamma_h)) = b_h \cdot (1 - b_h)$$

$$\frac{\partial E_k}{\partial b_h} = \sum_{j=1}^{n} \frac{\partial E_k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} = \sum_{j=1}^{n} (-g_j) \cdot w_{hj}$$

所以，

$\Delta v_{ih} = \eta(\sum_{j=1}^{n} g_j w_{hj}) \cdot b_h \cdot (1 - b_h) x_{ki} = \eta e_h x_{ki}$, 其中
$e_h = -\partial E_k / \partial \alpha_h = (\sum_{j=1}^{n} g_j w_{hj}) \cdot b_h \cdot (1 - b_h)$.

$$\Delta\gamma_h = (-\eta) \cdot \frac{\partial E_k}{\partial \gamma_h} = (-\eta) \cdot \frac{\partial E_k}{\partial b_h} \cdot \frac{\partial b_h}{\partial \gamma_h} = \eta \cdot (\sum_{j=1}^{n} g_j w_{hj}) \cdot (-1) \cdot$$
$$f^{'}(\alpha_h - \gamma_h) = (-\eta) \cdot (\sum_{j=1}^{n} g_j w_{hj}) \cdot b_h \cdot (1 - b_h) = (-\eta) \cdot e_h.$$

整理之后，任意参数 v 的更新式子是 $v \leftarrow v+\Delta v$, 并且更新的规则如下：

$$\Delta w_{hj} = \eta g_j b_h \text{ for all } 1 \leq j \leq n, 1 \leq h \leq q,$$

$$\Delta \theta_j = -\eta g_j \text{ for all } 1 \leq j \leq n,$$

$$\Delta v_{ih} = \eta e_h x_{ki} \text{ for all } 1 \leq i \leq n, 1 \leq h \leq q,$$

$$\Delta \gamma_h = -\eta e_h \text{ for all } 1 \leq h \leq q,$$

其中学习率 $\eta \in (0, 1)$ 控制着算法每一轮迭代中的更新步长，若步长太大则容易振荡，太小则收敛速度过慢，需要人工调整学习率。 对每个训练样例，BP 算法执行下面的步骤：先把输入样例提供给输入层神经元，然后逐层将信号往前传，直到计算出输出层的结果；然后根据输出层的误差，再将误差逆向传播至隐藏层的神经元，根据隐藏层的神经元误差来对连接权和阈值进行迭代（梯度下降法）。该迭代过程循环进行，直到达到某个停止条件为止。

标准 BP 算法的训练流程：
输入：训练集合 $D = (\mathbf{x}_k, \mathbf{y}_k)_{k=1}^{m}$ 和学习率 η.
过程：
1. 在 (0,1) 范围内随机神经网络中的所有连接权重和阈值
2. repeat
for all $(\mathbf{x}_k, \mathbf{y}_k)$ do
根据当前参数，计算出当前的样本输出 $\mathbf{y}_k$
计算输出层神经元的梯度项 g_j
计算隐藏层神经元的梯度项 e_h
更新连接权重 w_{hj}, v_{ih} 与阈值 θ_j, γ_h
end for
3. 达到停止条件
输出：链接权与阈值都确定的神经网络模型

累积 BP 算法：

BP 算法的目的是最小化训练集上的累计误差 $E = \sum_{k=1}^{m} E_k/m$, 其中 m 是训练集合中样本的个数。不过，标准的 BP 算法每次仅针对一个训练样例更新连接权重和阈值，也就是说，标准 BP 算法的更新规则是基于单个的 E_k 推导而得到的。通过类似的计算方法可以推导出累计误差的最小化更新规则，那就得到了累计误差逆传播（accumulate error backpropagation）算法。标准 BP 算法需要进行更多次的迭代，并且参数的更新速度快，累积 BP 算法必须扫描一次训练集合才会进

行一次参数的更新，而且累计误差下降到一定的程度以后 ，进一步下降就会明显变慢，此时标准 BP 算法往往会更快的得到较好的解，尤其是训练集合大的时候。

训练方法：

（1）把数据集合的每一列都进行归一化；

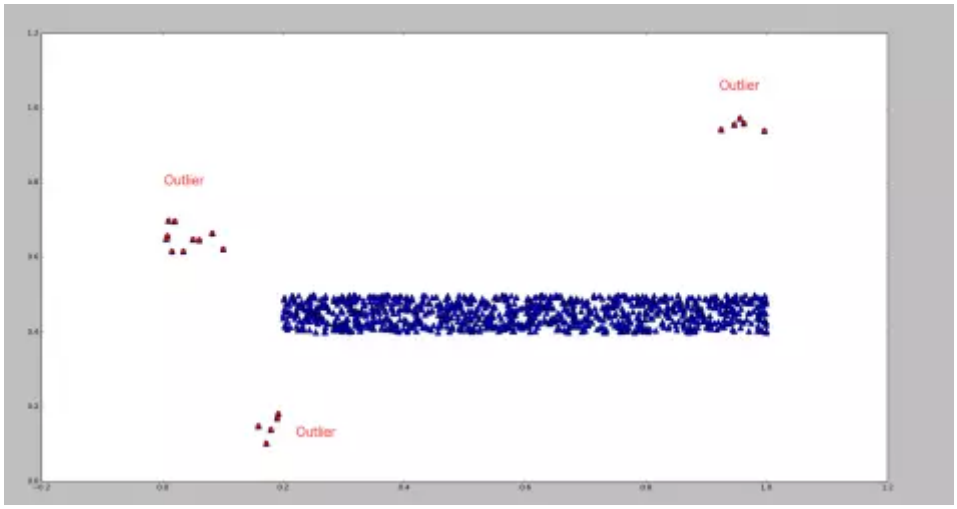
（2）选择 70% 的数据集合作为训练集合，30% 的数据集合作为验证集合。或者 训练集合 : 验证集合 = 8 : 2，这个需要根据情况而定。

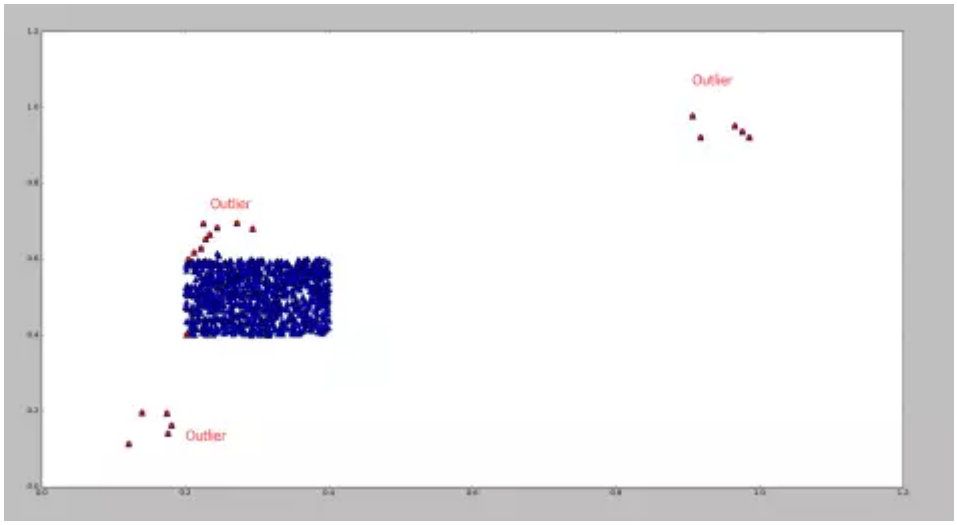
（3）随机生成一个三层的神经网络结构，里面的权重都是随机生成，范围在 [0,1] 内。输入层的数据和输出层的数据保持一致，并且神经网络中间层的节点个数是输入层的一半。

（4）使用后向传播算法（back-propagation）来训练模型。为了防止神经网络的过拟合，通常有两种策略来防止这个问题。（i）第一种策略是“早停”（early stopping）：当训练集合的误差降低，但是验证集合的误差增加时，则停止训练，同时返回具有最小验证集合误差的神经网络；（ii）第二种策略是“正则化”（regularization）：基本思想是在误差目标函数中增加一个用于描述网络复杂度的部分，例如链接权和阀值的平方和。

测试效果：

其中蓝色的点表示正常点，红色的点表示被 RNN 算法标记的异常点。





参考文献：

[1] Anomaly Detection Using Replicator Neural Networks Trained on Examples of One Class, Hoang Anh Dau, Vic Ciesielski, Andy Song

[2] Replicator Neural Networks for Outlier Modeling in Segmental Speech Recognition, Laszlo Toth and Gabor Gosztolya

[3] Outlier Detection Using Replicator Neural Networks, Simon Hawkins, Honxing He, Graham Williams and Rohan Baxter

END

相关文章推荐：

1. 量子计算（一）
2. 特征工程简介
3. 聚类算法（一）
4. 异常点检测算法（一）
5. 异常点检测算法（二）


数学人生