Hurst 指数和金融市场可预测性

Bo Qian　　Khaled Rasheed
计算机科学系
佐治亚大学
Athens， GA 30601
USA
[qian,khaled]@cs.uga.edu

摘　要

Hurst 指数（H）是一个统计学测量用来分类时间序列。当 H=0.5 时，表示一个完全随机的序列。而当 H>0.5 时，表示了一个具有保持趋势倾向能力的序列。H 的值越大，这个序列的倾向也越强。我们接下来将要研究如何利用 Hurst 指数来将不同时期的金融序列数据进行分类。BP 神经网络的实验表明，具有高 Hurst 指数的序列比那些 Hurst 指数接近于 0.50 的序列能够被更加精确的预测。因此，Hurst 指数提供了一种预测方法。

关键词：Hurst 指数，时间序列分析，神经网络，蒙特卡诺模拟，预测

1．介绍

Hurst 指数是 H. E. Hurst [1] 提出用来作分形分析的 [2][3] ，现在已经被用在许多研究领域。最近，由于 Peter 的相关工作 [7][8] ，它在金融领域也变的十分热门 [4][5][6] 。Hurst 指数为长期记忆和时间序列的分形提供了一种方法。由于它是高鲁棒性的基本系统的几个假设，现在已经被广泛用于时间序列分析。Hurst 指数的值在 0 和 1 之间。基于 Hurst 指数 H，一个时间序列能够被分为三种类型：（1）H=0.5 表明了序列可以用随机游走来描述。（2）0<H<0.5 表明了序列具有反持续性。（3）0.5<H<1 表明序列具有持续性。一个反持续性序列具有均值回复的特性，即意味着一个上升的值更有可能紧接着一个下降的值，反之亦然。H 的值越接近于 0.0，序列均值回复的能力也越强。而一个持续性序列具有保持倾向的能力，即下一时刻的值相对于现在值的变化，更有可能与这一时刻相对于上一时刻值的变化一致。H 的值越接近于 1.0，序列保持倾向的能力也越强。大多数的经济和金融时间序列具有持续性，即 H>0.5。

在时间序列的预测当中，我们首先需要解决的问题是我们想要研究的这个时间序列是否可以被预测。如果这个时间序列是随机的，一切的方法都是无效的。我们想要确定这些序列具有一定的可预测等级。我们知道一个具有很高 H 值的时间序列是具有很强的倾向性的，所以我们自然地认为这样的时间序列要比那些 H 值接近于 0.5 的时间序列更可能被预测。接下来，我们将要使用神经网络来测试这个假设。

神经网络是无参数的通用函数逼近 [9] ，可以无假设地从数据中进行学习。在过去的十年里，神经网络预测模型已经被广泛应用于金融时间序列分析 [10][11][12] 。

神经网络可以被用来代替通用函数逼近，进行预测。在同样的条件下，一个时间序列如果比另外一个时间序列具有更小的预测误差，我们便说它更容易被预测。从 1930 年 1 月 2 日到 2004 年 5 月 14 日，我们研究每日的道琼斯指数，计算每 1024 交易日的 Hurst 指数。从当中选出 30 个具有最大的 Hurst 指数与 30 个 Hurst 指数接近于随机序列的周期，然后我们用这些数据来训练我们的神经网络。我们对比这两组数据的预测误差，发现他们的预测误差完全不同。这个研究是通过 Matlab 来实现的，这篇文章所有的 Matlab 程序生成的结果都可以从 www.arches.uga.edu/~qianbo/research 下载。

在这篇论文接下来的部分：第二部分将会详细描述 Hurst 指数，第三部分我们将利用蒙特卡洛模拟过程来构造一个类似让我们感兴趣的金融序列，第四部分描述了一个我们模拟生成的混乱序列用来验证根据样本顺序构造的模型，第五部分描述了神经网络和他们用来验证高 Hurst 指数的序列能够比低 Hurst 值的序列更加准确地被学习和预测。最后，这篇论文将在第六部分作出结论。

2. Hurst 指数与 R/S 分析

Hurst 指数能够通过重标极差分析(R/S)分析。对于一个时间序列，$X = X_1, X_2, \ldots, X_n$，R/S 分析方法如下：

1) 计算平均值 m

$$m = \frac{1}{n}\Sigma_{i=1}^{n} X_i$$

2)计算均值调整序列 Y

$$Y_t = X_t - m \qquad t=1, 2, \cdots, n$$

3)计算累计偏离序列 Z

$$Z_t = \Sigma_{i=1}^{t} Y_i \qquad t=1, 2, \cdots, n$$

4)计算序列范围 R

$$R_t = \max(Z_1, Z_2, \ldots, Z_t) - \min(Z_1, Z_2, \ldots, Z_t)$$

t=1, 2, …, n

5)计算标准差序列 S

$$S_t = \sqrt{\frac{1}{t}\Sigma_{i=1}^{t}(X_i - u)^2}$$

t=1, 2, …, n

在这里，u 是从X_1到X_t的平均值

6)计算重标极差序列(R/S)

$$(R/S)_t = R_t/S_t \qquad t=1, 2, \cdots, n$$

我们把 $(R/S)_t$ 记为区间平均值 $[X_1, X_t], [X_{t+1}, X_{2t}]$ 直到 $[X_{(m-1)t+1}, X_{mt}]$ 其中 m=floor(n/t)。事实上，为了计算所有的数据，t 的值是可以整除 n 的。

Hurst 发现(R/S)随着时间的增加具有指数增长的规律，研究表明

$$(R/S)_t = c * t^H$$

在这里 c 是一个常数，H 被称为 Hurst 指数。为了分析 Hurst 指数，我们画出(R/S)

随着 t 变化的 log 图。这条回归直线的斜率便可以用来估计 Hurst 指数。当 t<10 时，$(R/S)_t$是不准确的。所以我们将利用至少 10 个值来计算重标极差。图 2.1 便是一个 R/S 分析的例子。

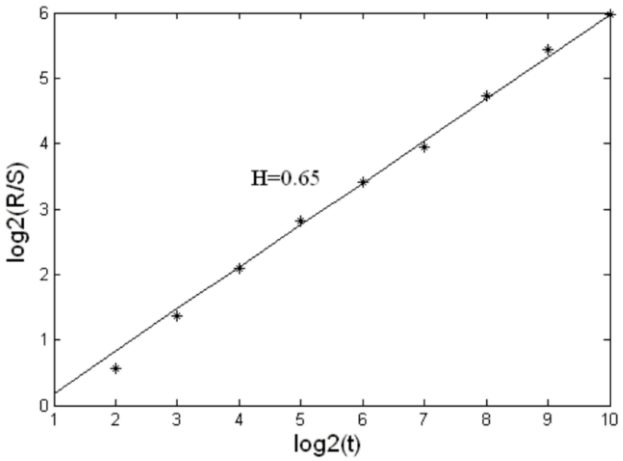

Figure 2.1. R/S analysis for Dow-Jones daily return from 11/18/1969 to 12/6/1973

在我们的实验中，我们计算了一个 1024 交易日周期的 Hurst 指数。我们用$t = 2^4, 2^5, \ldots, 2^{10}$来进行回归计算。在金融领域，采用对数变化率的计算每日收益的方法十分普遍。对于累积变化对应的累积收益率，这在 R/S 分析中是非常有意义的。图 2.2 展示的是道琼斯指数从 1930 年 1 月 2 日至 2004 年 5 月 14 日的日收益率。图 2.3 展示的是在这个期间内对应的 Hurst 指数。在这个期间内，Huest 指数从 0.4200 至 0.6804 波动。我们同样也想知道什么样的 Hurst 指数能够满足我们的条件。

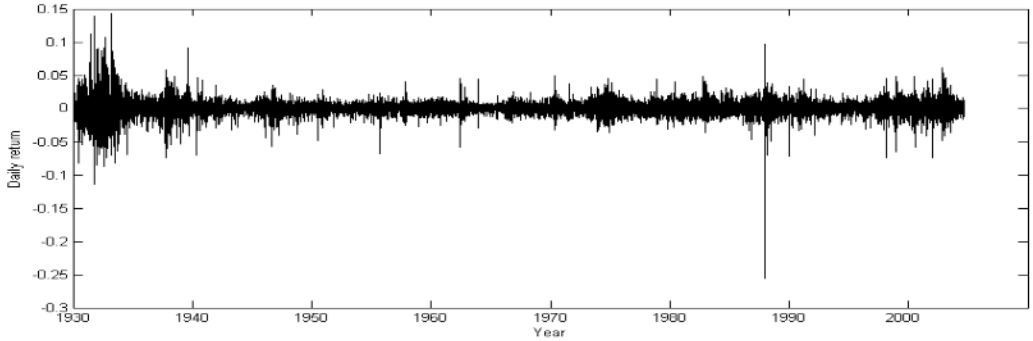

Figure 2.2. Dow-Jones daily return from 1/2/1930 to 5/14/2004

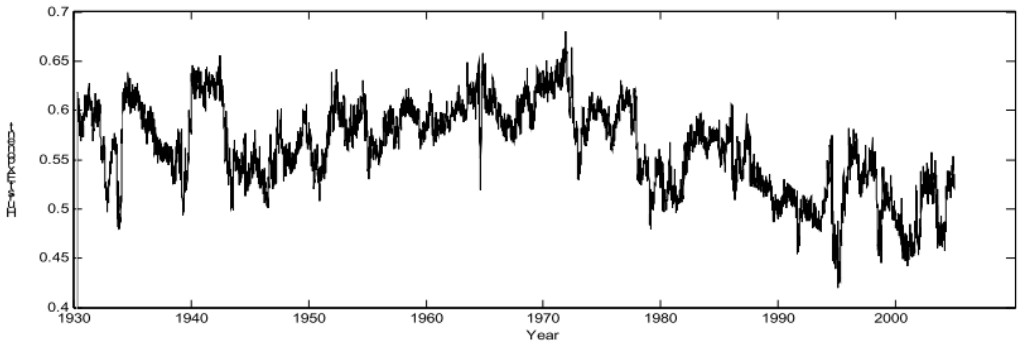

Figure 2.3. Hurst exponent for Dow-Jones daily return from 1/2/1930 to 5/14/2004

3. 蒙特卡洛模拟

对于一个随机序列，Feller [13] 给出了一个预计的(R/S)t 公式:

$$E((R/S)_t) = (n * \pi/2)^{0.50} \quad (3.1)$$

然而，这是一个近似的关系并且只在 t 很大时才有效。Anis 和 Lloyd [14] 为了在 t 很小时克服这个误差提供了下面这个公式：

$$E((R/S)_t) = \left(\frac{\Gamma(0.5*(t-1))}{\sqrt{\pi}*\Gamma(0.5*t)}\right) * \Sigma_{r=1}^{t-1}\sqrt{(t-r)/r} \quad (3.2)$$

当 t>300 时，大多数的计算机都很那来计算 gamma 函数了。利用 Sterling 的方程，此公式还能够被近似为：

$$E((R/S)_t) = (t * \pi/2)^{-0.50} * \Sigma_{r=1}^{t-1}\sqrt{(t-r)/r} \quad (3.3)$$

Peter 又给出了公式 3.2 的一个修正(公式 3.4)

$$E((R/S)_t) = \left(\frac{t-0.5}{t}\right) * (t * \pi/2)^{-0.50} * \Sigma_{r=1}^{t-1}\sqrt{(t-r)/r} \quad (3.4)$$

我们计算了对应$t = 2^4, 2^5, \ldots, 2^{10}$期望的(R/S)值并且在$\alpha = 0.05$的置信水平下作了平方回归。结果如表格 3.1 所示。

log2(t)	*log2(E(R/S))*		
	Feller	*Anis*	*Peters*
4	0.7001	0.6059	0.5709
5	0.8506	0.7829	0.7656
6	1.0011	0.9526	0.9440
7	1.1517	1.1170	1.1127
8	1.3022	1.2775	1.2753
9	1.4527	1.4345	1.4340
10	1.6032	1.5904	1.5902
Regression Slope (H)	**0.5000** ±5.5511e-016	**0.5436** ± 0.0141	**0.5607** ± 0.0246

Table 3.1. Hurst exponent calculation from Feller, Anis and Peters formula

从表格 3.1 可以看出，对于 Feller, Anis 和 Peter 之间的公式有不少的差别。更进一步，他们的公式都是基于大量的数据点的计算。我们现在的数据量固定在 1024 个点。所以在我们这种情况下，随机序列的 Hurst 指数是怎样的？
幸运地，我们可以利用蒙特卡洛估计法来得到结果。我们生成了 10000 个高斯随机序列。每个序列都有 1024 个值。我们计算了每个序列的 Hurst 指数和平均值。我们希望这个平均值尽量地接近实际的值。然后重复这个过程 10 次。下面的表格 3.2 给出了这个模拟结果。

	Simulated Hurst Exponent	*Standard deviation (Std.)*
1	0.5456	0.0486
2	0.5452	0.0487
3	0.5449	0.0488
4	0.5454	0.0484
5	0.5456	0.0488
6	0.5454	0.0481
7	0.5454	0.0487
8	0.5457	0.0483
9	0.5452	0.0484
10	0.5459	0.0486
Mean	**0.5454**	**0.0485**
Std.	**2.8917e-004**	

Table 3.2. Monte Carlo simulations for Hurst exponent of random series

从表格 3.2，我们可以看出在我们实验的情况下，可以计算出蒙特卡洛模拟的 Hurst 指数为 0.5454，标准差为 0.0485.这个结果是非常接近于 Anis 的公式的。通过以上的模拟，在 95%置信度的情况下，Husrt 指数处在 0.5454± 1.96*0.0485 的区间，即 0.4503 到 0.6405 之间。我们选择一个 Hurst 指数大于 0.65 的区间，希望找到这些区间一些不同于随机序列的规律。然而，从这些大样本(总共 17651 个周期)中选出的样本中，我们想知道是否存在或碰巧存在这些时期的真实模型。为了达到这个目的，我们进行了一个混乱测试。

4．混乱测试

为了测试是否存在一个真实的模型能满足 Hurst 指数大于 0.65 时期的样本，我们从中随机选择了 10 个样本。对于每一个样本，我们打乱这个序列，然后计算这个混乱序列的 Hurst 值。被打乱的序列与非随机序列的原始样本具有相同的分布。如果对于这些序列存在某些模型，在打乱顺序之后，这些模型将会被破坏并且计算的 Hurst 值也会接近于随机序列的 Hurst 值。在我们的实验当中，我们将每个样本打乱了 500 次，然后计算了平均的 Hurst 值。结果如表格 4.1 所示。

	Hurst exponent after scrambling	*Standard deviation*
1	0.5492	0.046
2	0.5450	0.047
3	0.5472	0.049
4	0.5454	0.048
5	0.5470	0.048
6	0.5426	0.048
7	0.5442	0.051
8	0.5487	0.048
9	0.5462	0.048
10	0.5465	0.052
Mean	**0.5462**	**0.048**

Table 4.1. The average Hurst exponent on 500 scrambling runs

从表格4.1我们可以看出,在样本被打乱顺序之后,Hurst指数都非常接近于0.5454，这与我们的模拟随机序列一致。通过这个结果，我们可以得出在这些时期内必然存在某些模型使得时间序列不同于随机序列，并且会被颠倒顺序被破坏。我们希望这个模型能够被用来进行预测。神经网络作为一个通用方程的近似器，提供了一个强大的工具用来学习这个潜在的模型。当潜在的规则未知时，他们变得尤为有用。我们希望用神经网络来发现这个模型并从中受益。我们用神经网络来控制错误率在可控制的范围内。接下来，我们对比 Hurst 指数大于 0.65 的时期与处于 0.54 和 0.55 之间的时期的预测误差。

5. 神经网络

在 1943 年，McClloch 和 Pitts [15] 提出了一个模拟神经估算模型。这项工作被普遍认为是人工神经网络研究的开端。Rosenblatt [16][17] 普及了感知器的概念并创造了很多感知器的学习规则。然而，在 1969 年，Minsky 和 Papert [18] 发现感知器无法解决一些非线性可分的问题。人们认识到多层感知器(MLP)能够模拟非线性可分方程，但是没人知道如何训练他们。神经网络的研究在 1986 年几乎停止了。在 1986 年，Rumelhart [19] 使用了反向传播算法来训练 MLP，终于解决了这个长时间困扰人们的问题。从那以后，神经网络在很多领域重新得到了重视。神经网络开始在金融领域变得热门，并且在金融方面的神经网络研究投入资金排名第二 [20]。

一个神经网络是一系列简单的相互联系的处理节点。每个节点计算加权输入，然后输出其传递函数对其他节点的值。前馈反向传播网络是最广泛使用的网络范式。利用反向传播算法训练算法，神经网络将调整权重，使它减少所观察到的输出之间的平方差（误差）和他们的目标值。反向传播算法使用梯度下降法寻找局部极小值误差曲面。它对每一个权重计算平方差的偏导数。偏导数（梯度）的相反数给出了使误差减小的方向。这个方向被称为最速下降方向。标准反向传播算法调整权重沿最速下降方向。虽然沿最速下降方向的偏差减少的最快，它通常收敛缓慢，并且可能收敛于局部最小值并振荡。因此，许多反向传播变种算法被发明，他们通过优化方向和步长来提高性

能。比如说几个有名的，我们有反向传播动量，共轭梯度，准牛顿以及 LM［21］。经过训练，我们可以使用这个网络来预测给定的不可见的输入。在神经网络预测中，第一步是数据的准备和预处理。经过训练后，我们可以用神经网络来预测给定的不可见的输入。

5.1 数据的准备和预处理

对于道琼斯日收益率数据，从 1930 年 1 月 2 日至 2004 年 5 月 14 日，我们计算每 1024 交易日的 Hurst 指数。在总共 17651 的时期中，有 65 个时期的 Hurst 指数大于 0.65,1152 个时期的 Hurst 指数处于 0.54 至 0.55 之间。图 5.3 展示了所有周期的条形图。

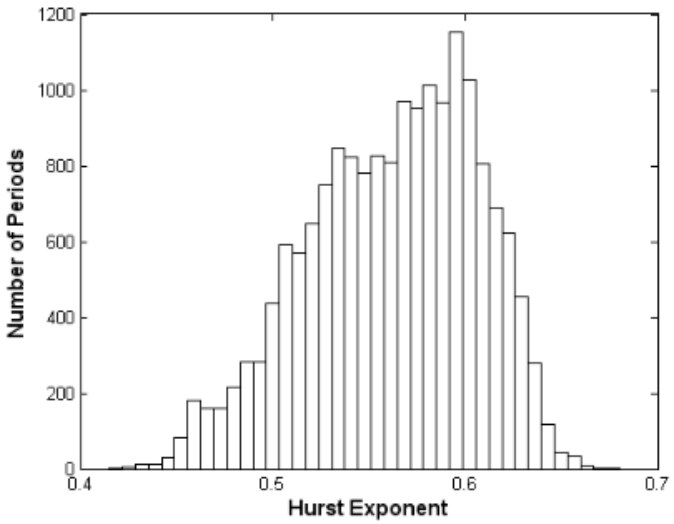

Figure 5.3. Histogram of all calculated Hurst exponents

我们从 Hurst 指数大于 0.65 的样本中随机选择了 30 个周期，并从 Hurst 指数处于 0.54 至 0.55 之间的也选择了 30 个周期。两组总共 60 个样本作为我们的原始数据集。

给定一个时间序列$x_1, x_2, \ldots, x_i, x_{i+1}$，基于$x_1, x_2, \ldots, x_i$我们应该如何构造一个向量$X_1$用来预测$x_{i+1}$呢？Taken 的理论［22］告诉我们，如果我们有合适的 d 和τ，我们可以通过延时向量$X_i = (x_i, x_{i+\tau}, x_{i+2\tau}, \ldots, x_{i+(d-1)\tau})$来重构一个潜在的系统。这里 d 被称为嵌入维度，$\tau$叫做分离。使用自动同步信息和假最近邻方法［23］，我们可以估算 d 和τ。我们用 TSTOOL［24］包对我们的 60 个数据集来运行自动同步信息和假最近邻方法。建议使用自动同步信息将所有数据集的分离信息的方法。这符合我们的直觉，因为我们没有理由使用分离的值。至于嵌入维数，我们的数据集建议从 3 到 5 来取。我们将在后面进行检验。

在构造好了时间延迟向量X_i和目标值x_{i+1}，我们将输入X_i和输出x_{i+1}规范化到平均值

为 0，标准差为 1 的分布。我们没有必要将输出规范化至一个有限的区间里，比如说 -0.85 到 0.85，以避免出现饱和的现象。因为我们在输出层使用了一个线性的转换方程。

我们使用了一个通常用来解决过拟合问题的神经网络。我们将数据集分成了三部分以用来进行训练，生效和测试。训练的数据集通过误差的反向传播来调整权重。生效集是通过判定此集合中的均方差开始增加时，停止训练。神经网络的预测性能是通过测试集来评定的。我们一般使用前 60%的数据来进行训练，接下来的 20%用来生效，最后的 20%作为测试数据。在这种方式下，我们能够对最后通过神经网络模型进行预测的结果更加有信心。

5.2 神经网络的构造

尽管神经网络是通用函数的近似器，我们仍然需要注意一下他们的构型。我们到底需要多少层？我们应该使用哪一种算法？实际上，到目前为止还没有直接的证据表明多隐含层的神经网络比单隐含层的神经网络更有优势，大多数在构建过程中都只使用了一层隐含层。因此，在我们的研究中将使用单隐含层的结构。对于学习算法，我们测试了 Levenberg-Marquardt， 共轭梯度法，以及包含动量算法的反向传播算法。我们发现 Levenberg-Marquardt 比其它的算法具有非常明显的优势。我们因此选择了 Lvenberg-Marquardt 算法，隐含层选用了 Sigmoid 传递函数，输出层则选用了一个线性函数。现在，我们需要确定隐藏的维数和隐含层的节点数。有一个探索式的规则用来确定隐含层的节点数，即神经网络总的自由度需要等于 1.5 倍的总数据量的平方根。根据这一条规则，我们得到以下的方程：

(#输入节点数+1)*(#隐含层节点数)+(#隐含层节点数+1)*(#输出节点数)=1.5*sqrt(#数据总数)　　(5.2.1)

方程得到隐含层节点数的解是 10.类似的，我们发现维度为 4 和 5 的网络隐含层的节点数为 8 和 7.对于每一个维度，根据建议的隐含层节点数，我们测试了 5 种神经网络构型。例如，对于 3 维度的 8,9,10,11,12 的隐含层节点数的神经网络。我们从每一组中(Hurst 指数大于 0.65 的组和 Hurst 指数处于 0.54 和 0.55 之间的组)随机选择了 5 个时期用来训练神经网络。每一个神经网络被训练 100 次，然后最小的规范误差均方根 NRMSE 被记录下来。NRMSE 被定义为：

$$\mathrm{NRMSE} = \frac{\sqrt{\Sigma_i (O_i - T_i)^2}}{\sqrt{\Sigma_i (T_i - T)}} \qquad (5.2.2)$$

在(5.2.2)中，O 是输出的值，T 是目标值。NRMSE 给出了一个对于平均预测的性能比较方法。如果我们经常使用这个平均值去预测，NRMSE 将会变成 1.当 NRMSE 的值为 0 时表明所有的预测都是正确的。

表格 5.1-5.3 给出了对于不同的神经网络的训练结果。

	Dimension 3, Hidden nodes					MIN	Std.
	8	9	10	11	12		
1	0.9572	0.9591	0.9632	0.9585	**0.9524**	0.9524	0.0039
2	0.9513	0.9531	0.9560	**0.9500**	0.9523	0.9500	0.0023
3	0.9350	0.9352	0.9332	**0.9328**	0.9359	0.9328	0.0013
4	0.9359	0.9426	0.9383	0.9351	**0.9313**	0.9313	0.0042
5	0.9726	0.9686	0.9652	0.9733	**0.9647**	0.9647	0.0040
6	0.9920	0.9835	0.9892	**0.9793**	0.9931	0.9793	0.0059
7	0.9831	0.9813	**0.9725**	0.9845	0.9825	0.9725	0.0048
8	0.9931	0.9852	**0.9832**	0.9877	0.9907	0.9832	0.0040
9	**0.9684**	0.9790	0.9773	0.9815	0.9862	0.9684	0.0066
10	**0.9926**	1.0044	1.0047	1.0014	1.0039	0.9926	0.0051
Mean	**0.9681**	0.9692	0.9683	0.9684	0.9693	0.9627	0.0042
Std.	0.0225	0.0215	0.0221	0.0232	0.0255	0.0207	0.0015

Table 5.1. NRMSE for dimension 3 with hidden nodes 8, 9, 10, 11 and 12

	Dimension 4, Hidden nodes					MIN	Std.
	6	7	8	9	10		
1	0.9572	0.9572	**0.9557**	0.9558	0.9633	0.9557	0.0031
2	0.9534	0.9554	0.9523	**0.9518**	0.9574	0.9518	0.0023
3	**0.9373**	0.9406	0.9414	0.9437	0.9404	0.9373	0.0023
4	0.9419	0.9471	0.9392	**0.9332**	0.9376	0.9332	0.0052
5	0.9691	0.9678	**0.9597**	0.9669	0.9662	0.9597	0.0037
6	0.9907	0.9939	**0.9836**	0.9948	0.9876	0.9836	0.0046
7	0.9902	0.9816	0.9872	**0.9766**	0.9855	0.9766	0.0053
8	0.9852	0.9842	**0.9802**	0.9865	0.9878	0.9802	0.0029
9	0.9809	0.9729	0.9722	**0.9669**	0.9741	0.9669	0.0050
10	**0.9916**	0.9957	0.9991	1.0025	0.9959	0.9916	0.0041
Mean	0.9698	0.9696	**0.9671**	0.9679	0.9696	0.9637	0.0038
Std.	0.0210	0.0193	0.0204	0.0225	0.0203	0.0196	0.0012

Table 5.2. NRMSE for dimension 4 with hidden nodes 6, 7, 8, 9 and 10

	Dimension 5, Hidden nodes					MIN	Std.
	5	6	7	8	9		
1	0.9578	**0.9560**	0.9617	0.9589	0.9622	0.9560	0.0026
2	0.9441	0.9466	0.9456	0.9456	**0.9427**	0.9427	0.0015
3	0.9410	**0.9395**	0.9449	0.9428	0.9396	0.9395	0.0023
4	0.9546	0.9453	0.9414	**0.9409**	0.9479	0.9409	0.0056
5	0.9659	**0.9501**	0.9671	0.9653	0.9653	0.9501	0.0071
6	0.9906	0.9919	0.9898	0.9901	**0.9891**	0.9891	0.0010
7	**0.9803**	0.9819	0.9805	0.9840	0.9837	0.9803	0.0017
8	**0.9912**	0.9980	1.0009	0.9991	1.0049	0.9912	0.0050
9	0.9770	0.9747	0.9742	0.9761	**0.9689**	0.9689	0.0032
10	**0.9909**	0.9984	0.9975	0.9977	0.9951	0.9909	0.0031
Mean	0.9693	**0.9682**	0.9704	0.9701	0.9699	0.9650	0.0033
Std.	0.0194	0.0233	0.0220	0.0226	0.0227	0.0217	0.0020

Table 5.3. NRMSE for dimension 5 with hidden nodes 5, 6, 7, 8, and 9

从表格 5.1 到 5.3 我们可以看出，对于每一个维度，不同的隐含层的节点数，NRMSE 的差异非常小。对于 3,4,5 维度的网络，拥有最小平均 NRMSE 值的隐含层节点数是 8,8,6.然后我们利用 8,8,6 的隐含层节点数的神经网络来对 3,4,5 维度的数据进行预测。每一个神经网络将会被训练 100 次，最小的 NRMSE 将会被记录下来。最后 NRMSE 将会是维度 3 的最小值。表格 5.4 给出了我们最初 60 个样本分成两组后的 NRMSE 值。

	H>0.65	*0.55>H>0.54*		*H>0.65*	*0.55>H>0.54*
1	0.9534	0.9863	16	0.93	0.9747
2	0.9729	0.9784	17	0.9218	0.9738
3	0.9948	0.9902	18	0.9256	0.9635
4	0.9543	0.9754	19	0.9326	0.957
5	0.9528	0.9773	20	0.937	0.9518
6	0.9518	0.9477	21	0.9376	0.9542
7	0.9466	0.9265	22	0.9402	0.9766
8	0.9339	0.9598	23	0.9445	0.9901
9	0.9299	0.9658	24	0.9498	0.9777
10	0.9435	0.9705	25	0.948	0.9814
11	0.9372	0.9641	26	0.9486	0.9778
12	0.9432	0.9824	27	0.9451	0.9968
13	0.9343	0.9557	28	0.9468	0.9966
14	0.9338	0.9767	29	0.9467	0.9977
15	0.9265	0.9793	30	0.9542	0.9883
Mean	**0.9439**	**0.9731**			
Std.	**0.0145**	**0.0162**			

Table 5.4. NRMSE for two groups

我们将两组不同的学生测试作为零假设，即两个组的平均值相同。再计算之后发现，t 的统计量是 7.369，p 的值是 7.0290e-010.这表示两个平均值是完全不同的，或者说他们相同的概率为 0.这个结论证明了具有高 Hurst 值的时间序列更容易被预测精准。

6. 结论

在这篇论文当中，我们分析了从 1930 年 1 月 2 日至 2004 年 5 月 14 日道琼斯股票所有 1024 交易日周期的 Hurst 指数。我们发现具有高 Hurst 值的数据比那些 Hurst 值接近于随机序列的数据更加容易被预测精确。这表明了股票市场并不是在所有时刻都是随机的。有一些时期会有很强的走势倾向模型，这些模型能够被神经网络学习并用于预测。

自从 Hurst 指数提供了一个评价可预测性的方法，在预测之前，我们能够使用这个值来指导我们进行数据的筛选。我们可以选择具有高 Hurst 值的模型来进行预测。更进一步，我们可以只关心具有高 Hurst 值的时期。这样能够大大地节约物力财力，指导

我们更有效率低进行预测。

引用:

[1] H.E. Hurst, Long-term storage of reservoirs: an experimental study, Transactions of the American society of civil engineers, 116, 1951, 770-799.
[2] B.B. Mandelbrot & J. Van Ness, Fractional Brownian motions, fractional noises and applications, SIAM Review, 10, 1968, 422-437
[3] B. Mandelbrot, The fractal geometry of nature (New York: W. H. Freeman, 1982).
[4] C.T. May, Nonlinear pricing : theory & applications (New York : Wiley, 1999)
[5] M. Corazza & A.G. Malliaris, Multi-Fractality in Foreign Currency Markets, Multinational Journal, 6(2), 2002, 65-98.
Finance
[6] D. Grech & Z. Mazur, Can one make any crash prediction in finance using the local Hurst exponent idea? Physica A: Statistical Mechanics and its Applications, 336, 2004, 133-145
[7] E.E. Peters, Chaos and order in the capital markets: a new view of cycles, prices, and market volatility (New York: Wiley, 1991).
[8] E.E. Peters, Fractal market analysis: applying chaos theory to investment and economics (New York: Wiley, 1994).
[9] K. Hornik, M. Stinchcombe & H. White, Multilayer feedforward
networks are requirements for universal
Neural networks, 2(5), 1989, 259-366 [10] S.
Walczak, An empirical analysis of financial forecasting with
approximators, data neural
networks, Journal of management information systems, 17(4), 2001, 203-222
[11] E. Gately, Neural networks for financial forecasting (New York: Wiley, 1996)
[12] A. Refenes, Neural networks in the capital markets (New York: Wiley, 1995)
[13] W. Feller, The asymptotic distribution of the range of sums of independent
mathematical statistics, 22, 1951, 427-432
[14] A.A. Anis & E.H. Lloyd, The expected value of the adjusted rescaled hurst
summands, Biometrika, 63, 1976, 111-116
[15] W. McCulloch and W. Pitts, A logical calculus of the ideas immanent in Mathematical Biophysics, 7, 1943,:115 - 133.
[16] F. Rosenblatt, The Perceptron: a probabilistic model for Information storage and organization in the brain, Psychological Review, 65(6), 1958,

386-408.
[17] F. Rosenblatt,, Principles of neurodynamics, (Washington D.C.: Spartan Press, 1961).
[18] M. Minsky & S. Papert, Perceptrons (Cambridge, MA: MIT Press, 1969)
[19] D.E. Rumelhart, G.E. Hinton & R.J. Williams, Learning internal representations by error propagation, in Parallel distributed processing, 1, (Cambridge, MA: MIT Press, 1986)
[20] J. Yao, C.L. Tan & H. Poh, Neural networks for technical analysis: a study on LKCI, International journal of theoretical and applied finance, 2(3), 1999, 221-241
[21] T. Masters, Advanced algorithms for neural networks : a C++ sourcebook (New York: Wiley, 1995)
[22] F. Takens, Dynamical system and turbulence, lecture notes in mathematics, 898(Warwick 1980), edited by A. Rand & L.S. Young (Berlin: Springer, 1981)
[23] A.S. Soofi & L. Cao, Modelling and forecasting financial data: techniques of nonlinear dynamics
(Norwell, Massachusetts: Kluwer academic publishers, 2002)
[24] C. Merkwirth, U. Parlitz, I. Wedekind & W. Lauterborn, TSTOOL user manual, http://www.physik3.gwdg.de/tstool/manual.pdf, 2002