《金融系统仿真》实验报告：
基于 EMD 和 SVR 的汇率预测

141292018 桑梓洲

摘要

我们选取的为 Premanode and Toumazou[1] 在 *Expert System with Applications* 的一篇文章。文章主要采用了 EMD 和 SVR 结合的方法将多个相关金融时间序列作为自变量以试图改进对欧元-美元汇率的预测效果，并通过与其他当今主流预测方法准确性对比有力表明所提出的方法确实能提升预测效率。我们则在原作者的基础上对文章的核心理论支持向量机进行了更深一步的研究并指出了原文中存在的一些不足之处。

1 文献回顾

1.1 文章贡献及概述

动机与贡献

外汇市场是重要的全球金融市场之一，汇率序列具有传统金融时间序列非线性、高频率、高波动特性。综合各领域成果，当今主流的处理此类特征数据的模型有：

1. MS-GARCH 计量中常用到的马尔科夫转移 GARCH 模型
2. EMD 广泛应用于各领域去噪声的经验模态分解算法
3. SVR 统计学习理论中的支持向量回归理论

作者创造性地提出一种由 EMD 平滑和 SVR 预测相结合的算法试图改进对汇率的预测效果，与目前主流的 MS-GARCH 和 MSR 方法对比后作者发现在以预测方向一致性为指标的检测上提出的算法效果明显更优。

文章概述

原文结构简述如下：

- 第一部分作者概述了本文的核心工作：EMD 结合 SVR 改进汇率预测，并对 EMD、SVR 和 MS-GARCH、MSR 算法的背景做了简要介绍。
- 第二部分作者给出了数据的来源。作者选取欧元-美元汇率作为待预测变量 y 并选取了涵盖股票、利率、汇率、商品期货四大类 21 个与之相关的金融序列作为输入的自变量 x；时间选取上从 1998 年到 2010 年供 3130 个观测日数据，以时间上前 70% 为测试集并以后 30% 为训练集检测效果。
- 第三部分作者简要说明了什么是 EMD 以及说明了在此基础上作者提出的 dEMD, 并对 SVR 理论做了简要说明。dEMD 是 differential EMD 的意思，这里的 differential 是指微分，因为作者的 IMF 是从汇率收益率序列提取的，收益率是原序列的一阶微分。
- 第四部分为作者实证研究及结果。通过 EMD 作者提取了最近似正态分布的一个 IMF（考虑到高斯白噪声服从正态分布）作为白噪声并将之从原始汇率序列中减去，通过对比去噪后序列结合 SVR 预测、原始数据结合 SVR 预测、MS-GARCH、MSR 四种方法预测效率对比发现 dEMD 去噪后得到的预测效果最好，衡量标准是预测序列和真实序列上下变动的一致性。
- 第五部分是结论。

本文的特点为：相较于金融方面的知识更侧重信号处理的技术应用，优点是方法具有普适性，容易迁移到其他领域，这也是机器学习理论的特点；但另一方面作为一篇金融文章来讲则缺乏研究的细致程度。

1.2 文章可能存在的一些问题

尽管我们从作者的研究成果中得到许多收获和启迪，但经过思考仍有以下不敢苟同之处：

1. 数据处理上，全文没有一处提到“归一化”。不仅是二十余个自变量未提到，作者将用 dEMD 算法从收益率序列提取的“准噪声序列”直接

从原始汇率中减去，我们认为即使欧元-美元汇率可以直接减掉，那么其他任意两个国家之间汇率也可以这样操作吗？

2. 降噪算法上，作者在经典的 EMD 算法上自创了“dEMD”，但在我们后来用的人民币-美元汇率试图照做一遍时从收益率序列中试图提取时发现所有 IMF 都无法通过正态检验，即无法提取“准噪声序列”，因此该方法目前看来缺乏理论依据且可能不具备普适性。

3. 核心算法上，作者只是列出理论简单运用了 SVR 却并未对真正核心性、有价值的的核函数选取、参数调优等做足够工作和说明，浅尝辄止；甚至参数选取上存在明显错误，比如 $\varepsilon = 1$ 即是说回归管道的宽度为 1，欧元-美元汇率预测值落在真实值 ± 1 范围均可。

4. 衡量标准上，单纯的变动方向预测正确可能意义不大。试想一个随机游走的序列即使全部预测向上可能也有 50% 的正确率；事实上我们的实验结果也表明 SVR 预测真实外汇序列变动并不如文中描述让人看起来的那样理想。

2 核心理论简介

2.1 经验模态分解 EMD

EMD (Empirical Mode Decomposition) 算法是由著名海洋学家，美国国家工程院黄锷 (Norden Huang) 院士于 1998 年作为 HHT (Hilbert-Huang Transform) 的一个重要组成部分提出的。HHT 广泛应用于生物医疗、神经科学、气象学、地震学、天体粒子物理学、化学化工等各领域，截止 2017 年 12 月原文 [2] 被引逾 15988 次。

EMD 算法的基本思想是将一个不规则的波化为多个单一频率的波加上残波的形式，即原波形 $=\Sigma$ IMFs $+$ 余波。

IMF 即 Intrinsic Mode Function，中文为本征模函数。判断 IMF 需满足以下**两个条件**：

1. 信号极值点的数目于零点的数目相等或最多相差不超过一个

2. 信号局部由极大值形成的包络和由极小值形成的包络的均值为零

EMD 算法的具体步骤

1. 找出信号 $x(t)$ 所有的极大（小）值点并将其用三次样条函数差值成为原数据序列的上（下）包络，上包络线和下包络线的平均线为 $m_1(t)$.

2. 将原数据序列减去平均包络后即可得到一个去掉低频的新数据序列 $h_1(t)$，即 $h_1(t) = x(t) - m_1(t)$，检查是否符合 IMF 条件，若不符合将 $h_1(t)$ 看作新的 $x(t)$ 重复得 $h_{11}(t) = h_1(t) - m_{11}(t)$，直到 k 次选出第一个符合 IMF 的分量 $c_1(t)$.

3. 原始信号减去得到的 IMF 分量，将残差序列 $res_1(t) = x(t) - c_1(t)$ 看作新的信号重复上述操作，直至残差序列不能再提取出 IMF 为止.

以上是经验模态分解算法的理论部分，具体实践中黄锷院士带领所在的台湾国立中央大学团队打造了基于 Matlab 的 EEMD 算法包，可对任意一个信号序列进行一键分解，正宗性自不必说。

2.2 支持向量回归 SVR

提到支持向量回归就不得不从它的母类支持向量机说起。支持向量机 (Support Vector Machine, SVM) 可分为支持向量分类机 (Support Vector Machines for Classification, SVC) 和支持向量回归机 (Support Vector Machines for Regression, SVR)。SVM 是一种基于统计学习理论十分年轻的机器学习方法，由 Corinna Cortes 和 Vapnik 等人于 1995 年首次提出，适合解决小样本、高维和非线性问题。

为了更好地理解 SVR 我们先简要介绍一下 SVC，要理解 SVC 首先要清楚什么是分类问题。分类 (classification) 是机器学习和数据挖掘中最重要、最频繁使用的算法，其基本作用就是：从一组已经带有分类标记的训练样本数据集来预测一个测试样本的分类结果。生活中的分类问题有很多，比如给定身高和体重区分男生还是女生，给定葡萄酒的化学成分区分酒的品种，再比如判断一句话的情感到底是积极消极还是中性，事实上给定足够的训练样本以上问题均可以由 SVC 实现。

了解了分类问题的概念后，我们从最简单的线性可分问题说起。图 1 中有两类待分类数据，想取得最好的分类效果代表我们试图找到一条直线让两组样本间的间隔 (margin) 最大。那么我们假设这条直线的表达式是 $w^T x + b = 0$, 两类样本的标签分别是 $+1$ 和 -1，我们想让分类为 $+1$ 的样本

都在直线 $w^Tx+b=1$ 之上，同理分类为 -1 的样本都在直线 $w^Tx+b=-1$ 之下，处在直线上的点就是支持向量，支持向量机的核心平面只由支持向量决定，这也是其得名的原因。经过解析几何知识我们不难得到两条直线之间的距离是 $\frac{2}{||w||}$，最大化间隔也即变成了以下约束：

$$\min \frac{1}{2}||w||^2, s.t.\ y_i(w^Tx+b) \geqslant 1$$

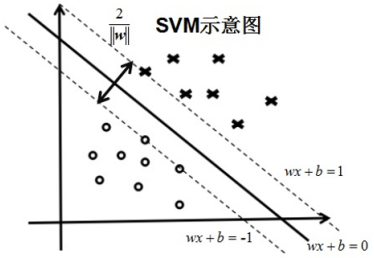

图 1: SVC 示意图

其最优求解可用凸优化理论中的拉格朗日对偶问题结合 KKT 条件或 SMO 算法得到，因为对数学要求比较高，所以我干脆也不费力去打自己都看不懂的数学公式了。值得注意的是因为是转化为二次优化求解，原问题维数不会影响算法复杂度，算法复杂度由支持向量个数决定，因此适合小样本、高维数据。

对于线性不可分的数据，支持向量机通过把它们映射到高维线性可分的空间，然后依然寻找超平面让两组样本之间间隔最大，但这样的一大问题是高维会使计算复杂度升高导致维数灾难，但神奇的是可以通过核函数简化计算 (kernel trick)，从而大大降低计算复杂度。此外上述公式最简单的版本，松弛变量的引入可以允许存在适当的误差防止过度拟合，罚函数的存在使得控制误差大小成为可能。

对于 SVR 其本质我们可以理解为非线性回归，依旧从最简单的线性情形开始，SVR 希望找到一条直线让样本点尽可能多地落在直线 $w^Tx+b=0$ 为中心宽度为 ε 的管道内。相比于最小二乘法点到直线的距离平方和最小化，SVR 试图最小化点到 ε 宽度管道的距离。对于非线性回归依然通过映射到高维并用核函数简化计算得到。对于简单情形的 ε-SVR 其数学表达式如下：

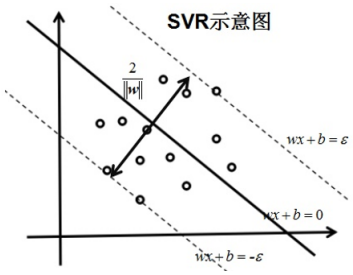

图 2: SVR 示意图

$$\min \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*), s.t. \begin{cases} y_i - w^T x_i - b \leqslant \varepsilon + \xi_i \\ w^T x_i + b - y \leqslant \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geqslant 0 \end{cases}$$

更多的详细证明过程请您参见机器学习相关书籍 [3] 及课件。

3 实证研究

此部分中，实验整体上我们沿袭了作者的思路：试图通过相关金融序列数据预测人民币-美元汇率，训练-测试集划分拟采用 70 : 30 比例分配。但相比原作者我们在 SVR 预测上增加了参数调优的工作，衡量指标改为 MSE 而非预测波动的一致性，我们认为这样能更好地反映真实预测效果。且尽管使用遗传算法调优后的参数可能因为过度拟合而导致效果不如不进行调优操作，但我们仍认为这是不可或缺的步骤，无论因效果不好绝口不提或忘记此步骤都有失妥当。

3.1 操作概述

我们的实验操作步骤先简要概述如下：

1. **获取数据**.Wind 上获取人民币-美元汇率作为因变量，选取主要相关国债利率、股票指数、商品期货为自变量，受制于 Wind 数据我们样本期为 2009.6.1-2017.11.30 共 2076 组观测值。

2. **EMD 分解**. 下载安装台湾国立中央大学 EEMD 软件包进行分解。与原作者得到的结论不同，我们将汇率收益率序列分解之后得到的 12 个 IMF 几大正态检验均未通过，因此 dEMD 方法失效。

3. **SVR 训练**. 下载安装国立台湾大学林智仁教授的 LibSVM 软件包；将各自变量进行归一化操作再通过 Java 程序转为要求格式，并放入 LibSVM 中进行回归训练。

4. **参数调优**. 下载安装 MatlabSky 论坛 SVM 达人 Faruto 的 UltimateLibSVM 包，手动粗调 c, g 值大体确定范围之后，通过包内的遗传算法实现参数精调，可使 MSE 达到 10^{-6} 级别

5. **SVR 预测**. 用之前最优训练得到的模型进行预测，返回预测效果并于不进行参数调优的结果进行对比，最终以 MSE 和 R^2 为衡量标准的检验中遗传算法参数调优甚至不如不操作。

3.2 详细说明

3.2.1 EMD 分解收益率序列

安装添加好中央大学软件包后，如需对某个信号序列 y 进行 EMD，只需输入以下代码:

```
result=eemd(y,1,0);
```

注意 EEMD 是 EMD 的升级版，当第二个参数为 1，第三个参数为 0 时就是 EMD。得到的结果会返回一个矩阵，其中第一列时原数据，其余为 IMF。我们想把若干阶 IMF 放在一张图中展示，以下是我们写的将第 1、3、5 个 IMF 和剩下的残差放在一起的 Matlab 代码：

```
rslt=eemd(y,1,0);
hold on;
for i=2:2:6;
% the 1st is original so 2nd is the 1st IMF
    % plot 2,4,6 column->1st,3rd,5th IMF
    plot(date,rslt(:,i),-0,005*(i-1));
```

```
7           % make offset to display them in the same …
               frame
8       end;
9       plot(date,sum(rslt(:,7,12),2)-0.03);
10      % sum of all other IMF, totalled 12
11      set(gca,'yTickLabel',[]);
12      % let y label be blank
13      datetick('x',12,'keepticks');
14      xlim([date(1),date(end)]);
```

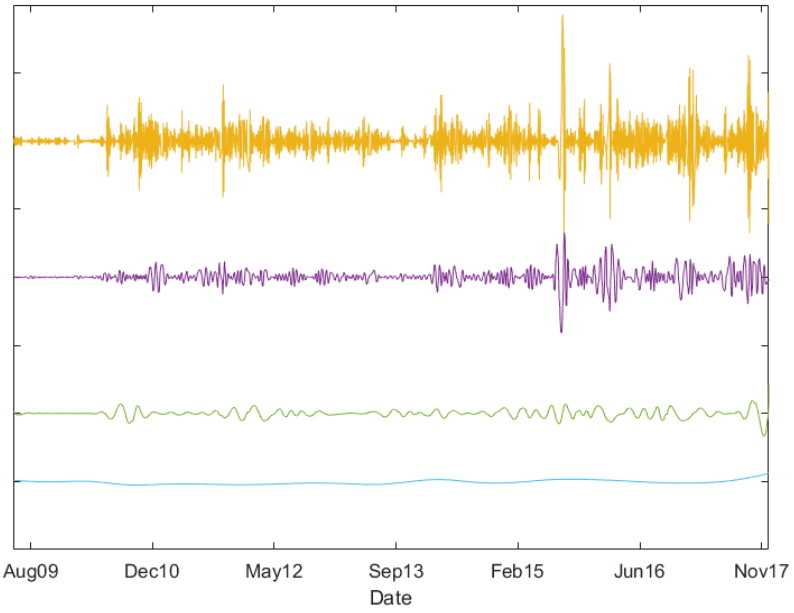

图 3: EMD 效果图

3.2.2 LibSVM 格式数据预处理

因为 SVM 常用于解决高维问题，我想 LibSVM 的作者考虑到了这点故要求了其数据输入格式必须为:

$$\begin{array}{cccccc} y_1 & 1:x_{11} & 2:x_{12} & \dots & n:x_{1n} \\ y_2 & 1:x_{21} & 2:x_{22} & \dots & n:x_{2n} \end{array}$$

在 Matlab 中存储也为稀疏矩阵的形式，这样在处理文本分类是面对可能上千维的数据就不会浪费过多空间和计算时间。对于我们从 Wind 上得到的 2076×7 维 Excel 数据，我们先将其保存为 txt 格式并用 Matlab 的 `price2ret` 和 `ret2price` 两步操作实现归一化处理，对于归一化处理好的数据我们通过以下 Java 代码将其转为 LibSVM 需要的格式：

```
FileReader fr=new FileReader("f:/all.txt");
BufferedReader br = new BufferedReader(fr);
FileWriter fw=new FileWriter("f:/asvm.txt");
BufferedWriter bw =new BufferedWriter(fw);
String s;
String[] str;
int i=0;
while ((s=br.readLine())!=null){
    str=s.split(" ");
    bw.flush();
    StringBuilder newStr=new …
       StringBuilder(str[0]);
    for(int j=1;j≤7;++j){
        str[j]=" "+j+":"+str[j];
        newStr.append(str[j]);
    }
    bw.write(newStr.toString());
    bw.newLine();
}
fr.close();
fw.close();
```

3.2.3 LibSVM 使用

LibSVM 包 [4] 针对 SVR 主要有以下几个可调参数：

- -s: SVM 类型，其中 0 为分类常用的 C-SVC，3 为回归常用的 ε-SVR
- -t: 核函数类型，0 为线性，1 为多项式，2 为较常用的径向基函数

(RBF)

- -c: 惩罚函数，越大则对误差要求越严格
- -g: 核函数系数，类似 $y = kx + b$ 中的 k, 核函数具体形式在自带文档中给出
- -p: ε-SVR 中的 ε 值，越大则允许的回归管道宽度越大
- -v: 交叉验证次数，随机分成 n 份交互检验分类/回归结果

此外，LibSVM 设计十分科学，各项参数填错或不填均按默认值处理，因此不必担心崩溃。预测效果衡量上，回归返回均方误差 MSE，分类返回分类精度 Accuracy。对于 LibSVM 使用可简洁概括为以下三步：

```
[y,x]=libsvmread(filePath); % 读取数据
model=svmtrain(y,x,'Parameters'); %训练模型
[py,acc,¬]=svmpredict(y,x,model); % …
    得到结果和精度，py为predicted y,acc代表accuracy
```

注意第三步中的 svmpredict 函数中的 y 值并非用于预测，而是用于和预测值对比检验模型效果，当然此检验只对训练集起作用，对于未知分类的测试集可以让 y 全为某一数值，反正返回的精度也是无意义的。

3.2.4 SVM 参数调优

在参数选取的早期阶段，固定了基本的参数如'-s 3' 之后，我们对 $t, c, g\ p$ 参数进行了手动的、大范围的粗调，通过对比 MSE 和 R^2 和直观上的拟合效果除了确定 RBF 核函数比较合适外还确定了各参数大体范围，后又通过调用 Faruto[5] 写的遗传算法包进行参数的调优：

```
ga_option=struct('maxgen',200,'sizepop',20, …
    'ggap',0.9,'cbound',[0,10],      …
    'gbound',[0,10],'pbound',[0.001,0.02],'v',5);
[bmse,bd,bg,bp]=gaSVMcgpForRegress(y,x,ga_option);
```

最终经过数个小时的计算之后得到的最优拟合效果 MSE 为 $3.0426e-06$, R^2 为 0.997965，对应的最优参数为 $c=4.7265, g=2.7770, p=0.001$, 图 4 为参数调优前后拟合效果对比。

图 4: 参数调优效果图

3.2.5 最终测试效果

然而尽管我们的训练集可以通过参数调优无限逼近原始数据，看起来就像真的一样，但当我们拿样本的后 30% 去测试的时候，预测的效果还是比较惨不忍睹的：

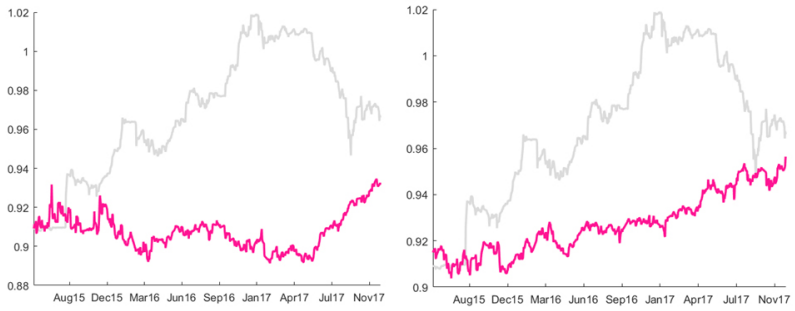

图 5: 最终测试效果图

图中灰色代表真实值，粉色代表预测值，左图为经过遗传算法参数调优后的结果，右图为"`-s 3 -p 0.01`" 简单参数调优后的效果；MSE 两者分别为 0.0061 和 0.0023, R^2 两者分别为 0.2608 和 0.4055，可见无论直观上还是具体的数据上都反映出了预测效果。

附画图部分代码如下：

```
    clf;
    model=svmtrain(y,x,'-s 3 -c 4.73 -t 2 -g 2.78 -p …
        0.001');
    py=svmpredict(y,x,model);
    hold on;
    plot(testdate,y,'Color',[0.86 0.86 …
        0.86],'LineWidth',2);
    plot(testdate,py,'Color',[1 0.078 …
        0.576],'LineWidth',2);

    datetick('x',12,'keepticks');
    xlim([testdate(1),testdate(end)]);
    set(gcf,'color','w');
```

4 实验反思

尽管我们的实验最终预测的效果没有那么理想，但这也恰恰说明了高频金融序列的难以预测性，同时我们的实验也存在以下两大方面的不足：

- 广度上，作者采用了多种方法证明 SVR 的相对有效性，而我们则因时间精力有限未能够和其他当今主流方法对比。
- 深度上，SVM 理论对我们现阶段水平而言仍较为复杂，我们现阶段只停留在表面应用层次。

针对以上不足我们也提出以下一些设想作为未来可能的改进空间：

- 去噪声算法上，或许可以用某一阶 IMF 作为趋势项从原始序列中减去达到平滑目的，而不必苛求正态性；且原作者服从正态分布就认为可以当作白噪声减去这种逻辑是完全站不住脚的
- 参数寻优上，或许可以依赖随机划分 (random split) 样本训练以得到最优参数，而不是直接划分前后两部分，这样或许从统计意义上能得到更优结果。

- 本文的算法无论 EMD 还是 SVR 都只是基于一次这样的实验，效果好很可能具有偶然性；若要证明方法的普适性可能还需要对多个类似的实验进行研究，尤其是 dEMD 算法很可能就不是普适的。
- 尽管 SVR 结果看起来不尽如人意，但很可能这是由于金融数据本身的难以预测行使然，需要其他主流方法对比来确定相对意义上的最优或许更能说明作者的方法是否具有更好的效果。

致谢

最后特别特别感谢瞿老师!!! 不仅是您教给我的 Matlab（用了一整个学期现在根本停不下来!）和领我入门的 SVM(要不然后来的大数据文本情感分类和风险管理论文都不知道做什么！)，不过还有比以上更重要但在此并不太方便透露哈哈，总之非常幸运选上您的课并从凭感觉选的 SVM 中收获多多！祝瞿老师天天开心!!

参考文献

[1] Bhusana Premanode and Chris Toumazou. Improving prediction of exchange rates using differential emd. *Expert Systems with Applications*, 40(1):377–384, 2013.

[2] Norden E. Huang, Zheng Shen, Steven R. Long, Manli C. Wu, Hsing H. Shih, Quanan Zheng, Nai-Chyuan Yen, Chi Chao Tung, and Henry H. Liu. The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 454(1971):903–995, 1998.

[3] Simon S Haykin. *Neural networks and learning machines*. China Machine Press, 2009.

[4] Chih Chung Chang and Chih Jen Lin. Libsvm: A library for support vector machines. *http://www.csie.ntu.edu.tw/cjlin/libsvm*, 2(3):1–27, 2001.

[5] 王小川. *MATLAB* 神经网络 *43* 个案例分析. 北京航空航天大学出版社, 2013.