

L₁因子分析及其在宏观经济中的应用

答辩人：蒯强 导师：孔新兵 教授

南京审计大学统计与数学学院

March 18, 2021



宏观经济和政府调控

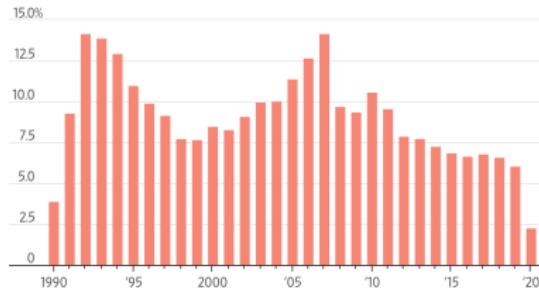
- 政府是指导和调控经济运行的主体。国民经济的发展和政府的指导和调控紧密相关。
- 政府部门需要时刻把握国民经济的方方面面，要研究这些经济变量发生变化的原因以及各种经济变量之间的关系。
- 只有这样，政府才能发现经济中存在的问题，并且给出针对性的指导和调控手段。



把控宏观经济数据

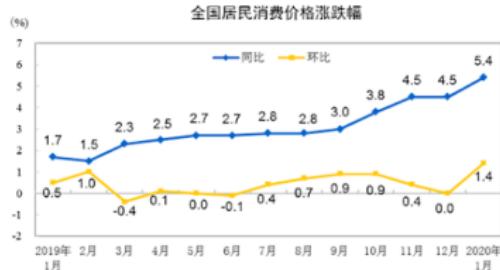
- 政府需要时刻把控宏观经济指标的当前水平。

中国年度GDP同比变化



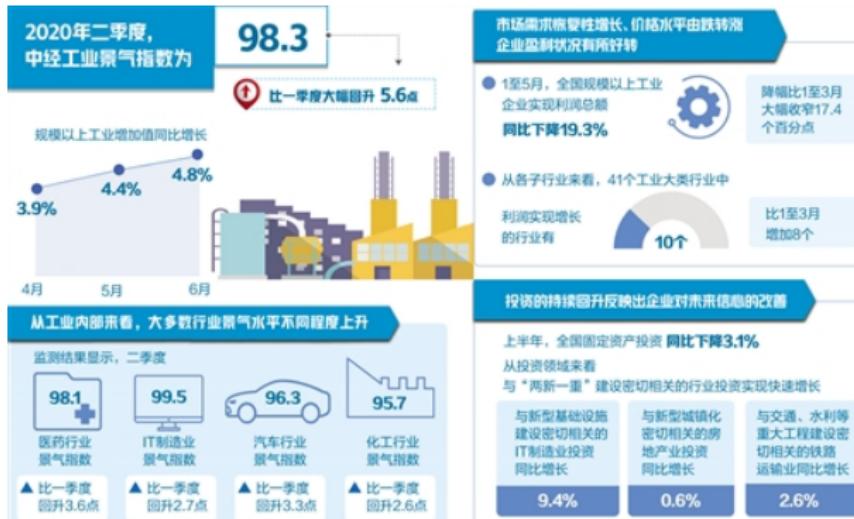
数据来源：中国国家统计局，经由万得资讯

全国居民消费价格涨跌幅



宏观经济现状分析

- 通过对当前宏观经济指标数据的综合分析，观察当前宏观经济的运行状况，便于行使见机行事的宏观经济政策。



宏观经济预测

- 政府和研究机构常常对重要经济指标做出预测

圖 1 IMF對全球經濟預測

國家	2019 年	2020 年(預測)	2021 年(預測)
全 球	2.9%	-3%	5.8%
台 灣	2.7%	-4%	3.5%
新 加 坡	0.7%	-3.5%	3%
韓 國	2%	-1.2%	3%
香 港	-1.2%	-4.8%	3.9%
日 本	0.7%	-5.2%	3%
中 國	6.1%	1.9%	7.4%
澳 洲	1.8%	-6.7%	6.1%
紐 西 蘭	2.2%	-7.2%	5.9%

資料來源：IMF, "World Economic Outlook 2020APR"



宏观经济预测

	2020年預測值	
	全年	
	預測值	年增率 (%)
實質GDP	19,360.24	1.15
民間消費	9,698.94	-0.30
政府消費	2,646.81	2.86
固定資本形成	4,439.10	4.11
民間投資	3,594.48	2.05
公營投資	264.33	21.56
政府投資	580.47	10.90
存貨變動	-11.33	-
貿易順差	2,553.50	-1.49
商品及服務輸出	12,103.47	-3.72
商品及服務輸入	9,567.24	-4.13
物價		
消費者物價指數	102.37	-0.18
躉售物價指數	95.88	-6.19
貨幣供給		
M 1B	19,851.12	7.43
M 2	47,115.94	4.36



高维宏观经济数据

- 宏观经济指标往往是高维的，难以采用传统模型进行分析。

工业

- 工业增加值（亿元）（2005年止）
- 工业增加值比上年同期增长（%）
- 主要工业产品产量
 - 铁矿石原矿产量（万吨）
 - 磷矿石（折含P2O5 30%）产量（万吨）
 - 原盐产量（万吨）
 - 饲料产量（万吨）
 - 精制食用植物油产量（万吨）
 - 成品糖产量（万吨）
 - 鲜、冷藏肉产量（万吨）
 - 乳制品产量（万吨）
 - 白酒（折65度，商品量）产量（万千升）
 - 啤酒产量（万千升）
 - 葡萄酒产量（万千升）
 - 软饮料产量（万吨）
 - 卷烟产量（亿支）
 - 织物产量（万吨）
 - 布产量（亿米）
 - 蚕丝及交织机织物产量（万米）
 - 机制纸及纸板（外购原纸加工除外）产量（万吨）
 - 汽油产量（万吨）（2013年止）
 - 煤油产量（万吨）（2013年止）
 - 柴油产量（万吨）（2013年止）
 - 焦炭产量（万吨）（2013年止）
 - 硫酸（折100%）产量（万吨）
 - 烧碱（氢氧化钠）（折100%）产量（万吨）
 - 纯碱（碳酸钠）产量（万吨）
 - 农用氮磷钾化肥（折纯）产量（万吨）
 - 化学农药原药（折有效成分100%）产量（万吨）
 - 乙烯产量（万吨）
 - 加工形态的塑料产量（万吨）

货物运输及沿海主要港口货物吞吐量

- 货运量总计（亿吨）
- 货运量总计比上年同期增长（%）
- 货物周转量
- 货物周转量同比增长
- 旅客运输
 - 客运量总计（亿人）
 - 客运量总计比上年同期增长（%）
 - 旅客周转量总计（亿人公里）
 - 旅客周转量总计比上年同期增长（%）
- 邮电业务
 - 邮电业务总量（亿元）

能源

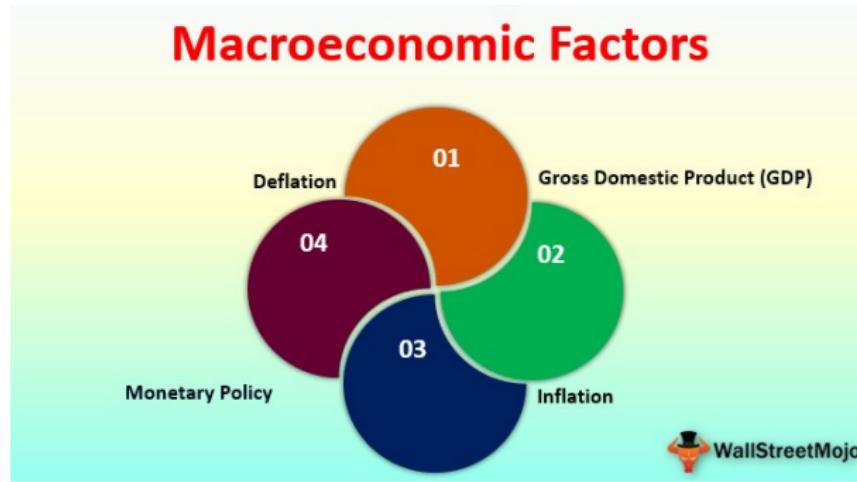
- 能源生产总量（万吨）
 - 原煤产量（万吨）
 - 天然原油产量（万吨）
 - 天然气产量（亿立方米）
 - 发电量（亿千瓦小时）
 - 液化天然气产量（万吨）
 - 汽油产量（万吨）
 - 煤油产量（万吨）
 - 柴油产量（万吨）
 - 燃料油产量（万吨）
 - 石脑油产量（万吨）
 - 液化石油气产量（万吨）
 - 石油焦产量（万吨）
 - 焦炭产量（万吨）
 - 煤气产量（亿立方米）
 - 石油沥青产量（万吨）
- 能源生产总量同比增长（%）

国内贸易

● 社会消费品零售总额 / (%)



因子分析是分析高维宏观经济数据的常用手段



扩散指数模型进行宏观经济预测

令 y_t 为待预测经济变量 y 在时间 t 的水平， X_t 为 p 维随机向量，假设 (X_t, y_t) 服从近似因子模型并且 X_t 和 y_t 具有相依性，若 X_t, y_t 有 m 维共同因子 F_t ，即

$$X_t = AF_t + e_t \quad (1)$$

则可以通过式(8)对 y_{t+h} 进行预测，

$$y_{t+h} = \beta(L)F_t + \alpha(L)y_t + c + e_{t+h} \quad (2)$$

其中滞后算子 $\beta(L)$ 反映了共同因子滞后项的影响，而 $\alpha(L)$ 表示 y_t 自身的滞后项的影响。



扩散指数模型进行宏观经济预测

Stock and Watson: Macroeconomic Forecasting Using Diffusion Indexes

151

Table 1. Simulated Out-of-Sample Forecasting Results: Real Variables, 12-Month Horizon

Forecast method	Industrial production		Personal income		Mfg & trade sales		Nonag. employment	
	Rel. MSE	$\hat{\alpha}$	Rel. MSE	$\hat{\alpha}$	Rel. MSE	$\hat{\alpha}$	Rel. MSE	$\hat{\alpha}$
<i>Benchmark models</i>								
AR	1.00		1.00		1.00		1.00	
LI	.86 (.27)	.57 (.13)	.97 (.21)	.52 (.15)	.82 (.25)	.63 (.17)	.89 (.23)	.56 (.14)
VAR	.97 (.07)	.75 (.68)	.98 (.05)	.68 (.34)	.98 (.04)	.73 (.58)	1.05 (.09)	.22 (.41)
<i>Full dataset (N = 215)</i>								
DI-AR, Lag	.57 (.27)	.76 (.13)	.77 (.14)	.76 (.13)	.48 (.25)	.99 (.15)	.91 (.13)	.63 (.18)
DI-AR	.63 (.25)	.71 (.12)	.86 (.16)	.61 (.12)	.57 (.24)	.84 (.18)	.99 (.31)	.51 (.20)
DI	.52 (.26)	.88 (.17)	.86 (.16)	.61 (.12)	.56 (.23)	.94 (.20)	.92 (.26)	.55 (.20)
<i>Balanced panel (N = 149)</i>								
DI-AR, Lag	.67 (.25)	.70 (.13)	.82 (.15)	.70 (.13)	.56 (.23)	.91 (.16)	.88 (.14)	.68 (.18)
DI-AR	.67 (.25)	.70 (.12)	.92 (.14)	.57 (.12)	.61 (.23)	.80 (.17)	.88 (.22)	.58 (.17)
DI	.59 (.25)	.81 (.17)	.92 (.14)	.57 (.12)	.57 (.23)	.91 (.18)	.84 (.21)	.62 (.16)
<i>Stacked balance panel</i>								
DI-AR	.65 (.25)	.70 (.12)	.93 (.15)	.56 (.12)	.61 (.22)	.89 (.19)	1.02 (.30)	.49 (.14)
DI	.62 (.25)	.81 (.18)	.93 (.15)	.56 (.12)	.66 (.21)	.85 (.20)	.95 (.24)	.53 (.14)
<i>Full dataset; m = 1, p = BIC, k fixed</i>								
DI-AR, k = 1	1.06 (.11)	.27 (.34)	1.03 (.08)	.34 (.41)	.98 (.06)	.63 (.46)	1.01 (.09)	.49 (.24)
DI-AR, k = 2	.63 (.25)	.76 (.14)	.78 (.14)	.77 (.14)	.53 (.24)	.93 (.15)	.77 (.13)	.82 (.15)
DI-AR, k = 3	.56 (.26)	.84 (.14)	.77 (.15)	.77 (.13)	.52 (.23)	.99 (.16)	.84 (.14)	.75 (.20)
DI-AR, k = 4	.54 (.26)	.85 (.14)	.76 (.15)	.78 (.14)	.51 (.23)	1.01 (.16)	.83 (.15)	.73 (.19)
<i>Full dataset; m = 1, p = 0, k fixed</i>								
DI, k = 1	1.03 (.07)	.30 (.49)	1.01 (.09)	.46 (.34)	.98 (.05)	.67 (.49)	1.01 (.09)	.48 (.24)
DI, k = 2	.55 (.25)	.89 (.15)	.78 (.14)	.76 (.13)	.57 (.24)	.95 (.17)	.78 (.13)	.83 (.16)
DI, k = 3	.51 (.25)	1.00 (.16)	.77 (.15)	.77 (.13)	.60 (.21)	1.02 (.19)	.84 (.14)	.76 (.19)
DI, k = 4	.49 (.25)	1.00 (.16)	.76 (.15)	.78 (.14)	.59 (.22)	1.03 (.20)	.82 (.15)	.75 (.18)
RMSE, AR Model	.049		.027		.045		.017	



扩散指数模型的因子估计

为了得到因子序列，需要对静态因子进行估计，Stock和Watson采用主成分分析作为非参数估计。

2.2 Estimation

In “small- N ” dynamic factor models, forecasts are generally constructed using a three-step process (see, e.g., Stock and Watson 1989). First, parametric models are postulated for the joint stochastic process $\{y_{t+h}, X_t, w_t, e_t\}$, and the sample data $\{y_{t+h}, X_t, w_t\}_{t=1}^{T-h}$ are used to estimate the parameters of this process, typically using a Gaussian Maximum likelihood estimator (MLE). Next, these estimated parameters are used in signal extraction algorithms to estimate the unknown value of F_T . Finally, the forecast of y_{T+h} is constructed using this estimated value of the factor and the estimated parameters. When N is large, this process requires estimating many parameters using iterative nonlinear methods, which can be computationally prohibitive. We therefore take a different approach and estimate the dynamic factors nonparametrically using the method of principal components.

Consider the nonlinear least squares objective function,

$$V(\tilde{F}, \tilde{\Lambda}) = (NT)^{-1} \sum_i \sum_t (x_{it} - \tilde{\lambda}_i \tilde{F}_i)^2, \quad (5)$$

written as a function of hypothetical values of the factors (\tilde{F}) and factor loadings ($\tilde{\Lambda}$), where $\tilde{F} = (\tilde{F}_1 \tilde{F}_2 \dots \tilde{F}_T)'$ and $\tilde{\lambda}_i$ is the i th row of $\tilde{\Lambda}$. Let \hat{F} and $\hat{\Lambda}$ denote the minimizers of $V(\tilde{F}, \tilde{\Lambda})$.

After concentrating out \hat{F} , minimizing (5) is equivalent to



主成分分析法

- 主成分分析法是常见的降维方法。



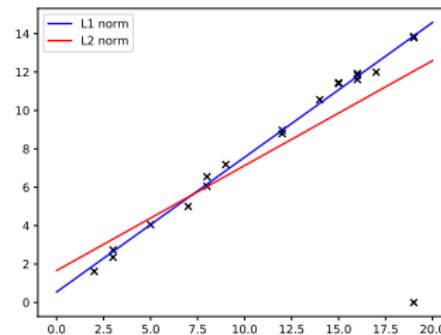
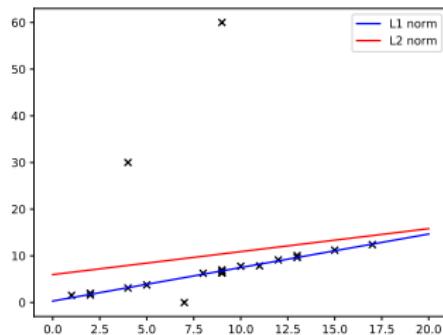
L₂主成分分析不稳健

- L₂主成分分析对离群值不稳健。



L₁和L₂的稳健性对比

- L₁范数能够提供更好的稳健性。



L₂主成分分析问题描述

- L₂主成分分析问题描述

$$P_1 : \hat{A}_{p \times m}, \hat{F}_{m \times n} = \arg \min_{A, F} \|X - AF\|_{L_2} = \arg \min_{A, F} \sum_{i=1}^p \sum_{j=1}^m (x_{ij} - a_i^T f_i)^2, \quad (3)$$

其中X为 $p \times n$ 的高维数据矩阵，A的列构成了X的m维线性子空间的基，这个子空间也称为特征空间。

F为一系数矩阵，给出了X各列元素在特征空间中的坐标，根据矩阵投影理论，在给定A的条件下， $F = A^T X$ 。

问题P₁可以理解为，需要找到一个合适的投射矩阵，使得数据在低维的投影上升回高维空间后和原矩阵各元素的误差平方和最小。

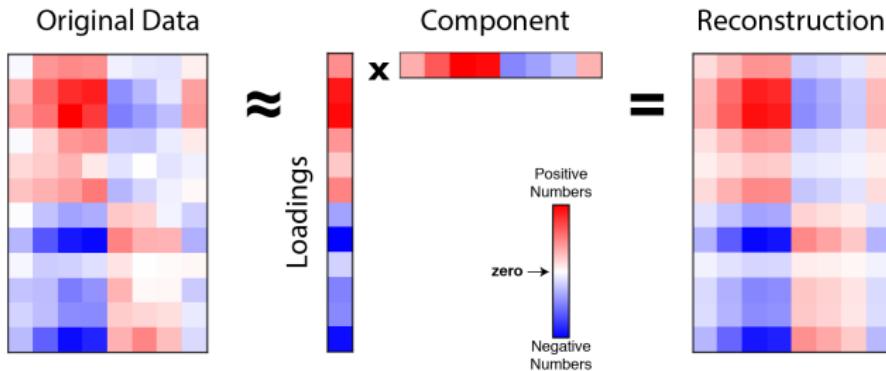
对于问题P₁常用奇异值分解法求解。同样地我们也可考虑其对偶问题P₂，

$$P_2 : \hat{A} = \arg \max_A \|A^T X\|_{L_2}, \text{ 其中 } A^T A = I_m. \quad (4)$$

问题P₂可以理解为，需要找到一个合适的投射矩阵，使得数据在低维空间的投影有最大的方差。在统计学上，数据的方差反映了数据中信息的多少，因此在特征空间中选取方差最大的方向作为主成分是很恰当的。



重构误差最小化



L₁主成分分析问题描述

- L₁主成分分析问题描述

$$P_3 : \hat{A}_{p \times m}, \hat{F}_{m \times n} = \arg \min_{A, F} \|X - AF\|_{L_1} = \arg \min_{A, F} \sum_{i=1}^p \sum_{j=1}^m |x_{ij} - a_i^T f_i|. \quad (5)$$

$$P_4 : \hat{A} = \arg \max_A \|A^T X\|_{L_1} = \arg \max_A \sum_{i=1}^n \sum_{k=1}^m \left| \sum_{j=1}^p a_{jk} x_{ij} \right|, \text{ 其中 } A^T A = I_m \quad (6)$$

一种L₁主成分分析的交替凸优化算法

- Qifake & Kande于2005年提出一种交替凸优化算法。

交替凸优化求解L₁ 主成分分析 (ACP, Alternate Convex Programming)

- 初始化：给出A, Σ 的初始值 $A^{(0)}$, $\Sigma^{(0)} = I$, (其中 Σ 为一对角矩阵, I为单位矩阵) ;
- 交替凸优化：对于迭代次数 $t = 1, \dots, \tau$:

$$F^{(t)} = \arg \min_F \|X - A^{(t-1)} \Sigma^{(t-1)} F^T\|_{L_1}$$

$$A^{(t)} = \arg \min_A \|X - A \Sigma^{(t-1)} (F^{(t)})^T\|_{L_1}$$

归一化:

$$\left\{ \begin{array}{l} N_a = \text{diag}((A^{(t)})^T A^{(t)}) \\ N_f = \text{diag}((F^{(t)})^T F^{(t)}) \\ F^{(t)} \leftarrow F^{(t)} N_f^{-1} \\ A^{(t)} \leftarrow A^{(t)} N_a^{-1} \\ \Sigma^{(t)} \leftarrow N_a \Sigma^{(t-1)} N_f \end{array} \right.$$

- 输出结果: $A \leftarrow A^{(\tau)} \Sigma^{1/2}$, 对A进行QR分解取正交矩阵得到 \hat{A} ; $\hat{F} \leftarrow \hat{A}^T X$ 。



模拟实验准备

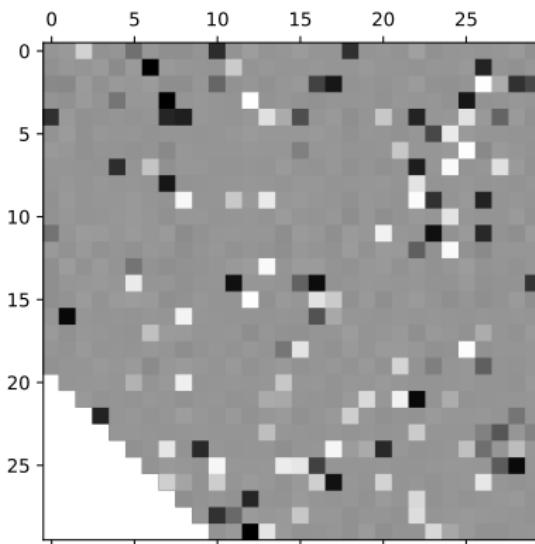
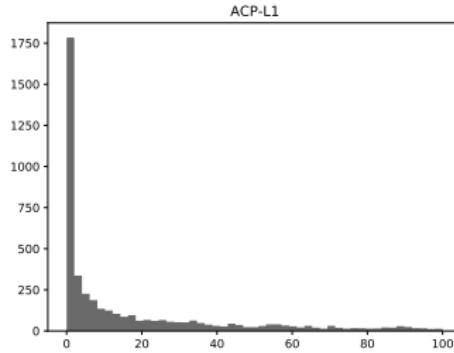
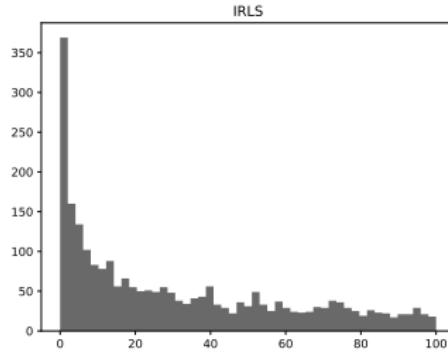
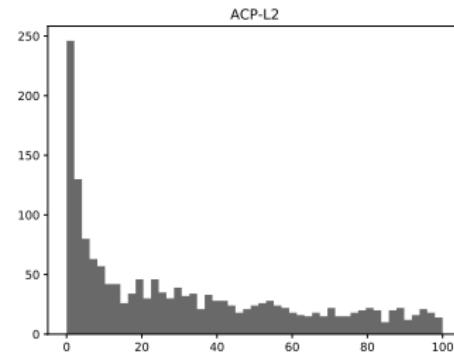
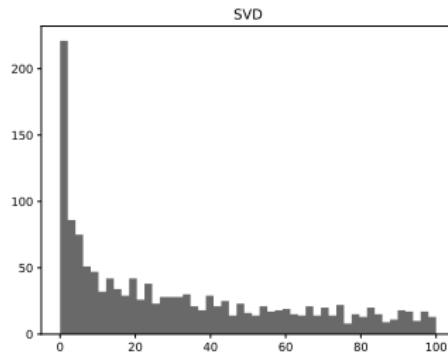


Figure: 30 × 30 的模拟矩阵



模拟实验结果



基于国内主要月度宏观经济数据的实证研究

- 数据来源：EPS数据平台-全国月度宏观经济数据

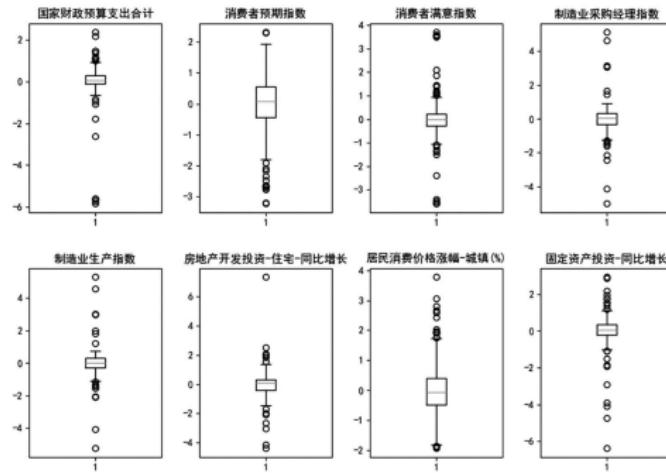
The screenshot shows the EPS Data Platform website. At the top, there is a navigation bar with links for '帮助中心' (Help Center), 'English', and '个人登录' (Personal Login). Below the navigation bar, there is a search bar with the placeholder '请输入关键字' (Please enter keyword) and a dropdown menu labeled '已选指标(0)' (0 selected indicators). The main interface displays a sidebar with categories like '中国宏观经济数据库' (China Macroeconomic Database), '行维度' (Dimension), '列维度' (Dimension), '固定维度' (Fixed Dimension), and '查询' (Query). The '查询' section is currently active, showing a list of economic indicators. Some indicators have checkboxes next to them, such as '工业增加值 (亿元) (2005年止)' (Industrial output value (billion yuan) (2005 year end)) and '社会消费品零售总额 (亿元)' (Social consumer goods retail total value (billion yuan)). At the bottom right of the search area, there are '重置' (Reset) and '确定' (Confirm) buttons.



基于国内主要月度宏观经济数据的实证研究

- 宏观经济数据的重尾性

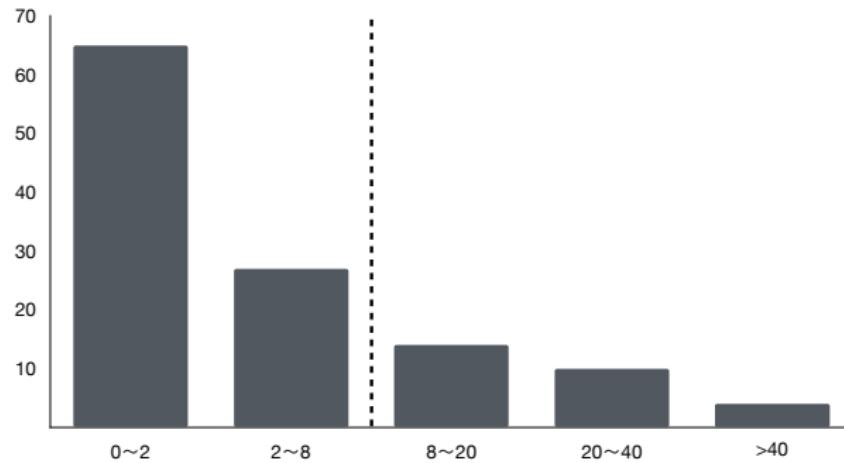
部分指标箱形图



基于国内主要月度宏观经济数据的实证研究

- 宏观经济数据的重尾性

120项指标的峰度分布



利用扩散指数模型进行预测

- 扩散指数模型预测

$$X_t = AF_t + e_t \quad (7)$$

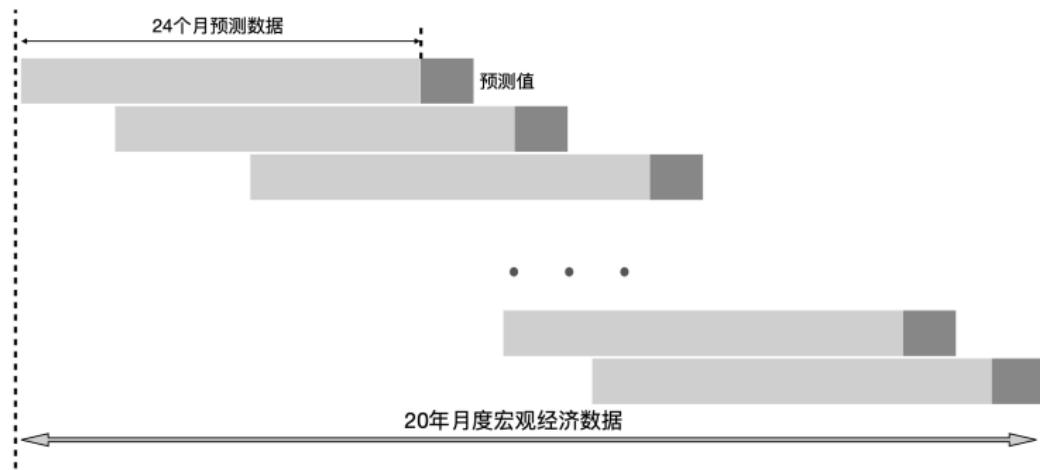
$$y_{t+h} = \beta(L)F_t + \alpha(L)y_t + c + e_{t+h} \quad (8)$$

- 因子个数选择，Bai&Ng信息准则。



利用扩散指数模型进行预测

- 滑动窗口预测



实证结果

- 预测结果表明：L₁因子在短期预测中比L₂因子更准确。
长期预测，表现也良好。

Table: 向前一个月预测结果

	MSE (L1 PCA)	MSE (IM)	MAE (L1 PCA)	MAE (IM)	MPAE (L1 PCA)	MPAE (IM)
消费者满意指数*	0.81	1.49	0.87	1.67	0.43	1.55
工业生产者出厂价格指数*	0.70	1.98	0.75	1.54	0.96	1.60
货币供应量M2*	0.76	1.25	0.90	1.19	0.45	0.97
固定资产投资总额*	0.89	1.03	0.97	0.99	0.81	1.32
房地产开发投资总额*	0.79	0.98	0.95	1.00	0.91	1.20
社会消费品零售总额*	0.84	1.25	0.87	1.12	0.45	1.17
制造业采购经理指数	1.24	1.69	1.06	1.43	0.90	1.50
住宅新开工面积总数*	0.89	2.40	0.85	1.98	0.45	2.22
股票流通市值*	0.99	3.99	0.98	2.84	0.81	4.51
消费者信心指数*	0.93	1.01	0.98	0.99	0.98	1.56



交替凸优化算法的计算性能问题

- 交替凸优化算法需要求解多个最小绝对值回归问题，在变量较多情况下，线性规划算法计算性能较差。

$$\mathbf{F}^{(t)} = \arg \min_{\mathbf{F}} \|\mathbf{X} - \mathbf{A}^{(t-1)} \mathbf{F}\|_{L_1} \quad (2.12)$$

$$\mathbf{A}^{(t)} = \arg \min_{\mathbf{A}} \|\mathbf{X} - \mathbf{A} \mathbf{F}^{(t)}\|_{L_1} \quad (2.13)$$

我们改写式(2.12)中的目标函数，

$$\xi(\mathbf{F}) = \|\mathbf{X} - \mathbf{A}^{(t-1)} \mathbf{F}\|_{L_1} = \sum_{j=1}^n |\mathbf{x}_j - \mathbf{A}^{(t-1)} \mathbf{f}_j| \quad (2.14)$$

其中 \mathbf{x}_j 是矩阵 \mathbf{X} 的第 j 列， \mathbf{f}_j 是 \mathbf{F} 的第 j 列。于是式(2.12)问题可以分解为 n 个独立的子优化问题，求解相应的 \mathbf{f}_j ：

$$\mathbf{f}_j^{(t)} = \arg \min_{\theta} |\mathbf{A}^{(t-1)} \theta - \mathbf{x}_j| \quad (2.15)$$

同样地，(2.13)可以转化为下面的 p 个独立的子优化问题，

$$\mathbf{a}_i^{(t)} = \arg \min_{\theta} |\mathbf{x}_i - \theta^T \mathbf{F}^{(t-1)}| \quad (2.16)$$

其中 \mathbf{a}_i 为 \mathbf{A} 的第 i 行，而 \mathbf{x}_i 为 \mathbf{X} 的第 i 行。



最小绝对值回归的优化1: 聚类——迭代拆解算法

- Park等人于2016年提出一种聚类——迭代拆解算法，通过减小问题规模来提升计算性能。

算法 3.1 聚类——迭代拆解算法 (Aggregate and Iterative Disaggregate, AID)

输入：原始数据集 $\mathbf{X}_{n \times p}$, 样本点的下标集合 $I = \{1, 2, \dots, n\}$, 数据的特征下标集合 $J = 1, 2, \dots, p$, 原优化问题 P 。

初始化：对原始数据集 \mathbf{X} 聚类，然后按某种规则产生新的优化数据集 $\mathbf{X}^{(1)}$ 。

对于 $t = 1, \dots, \tau$:

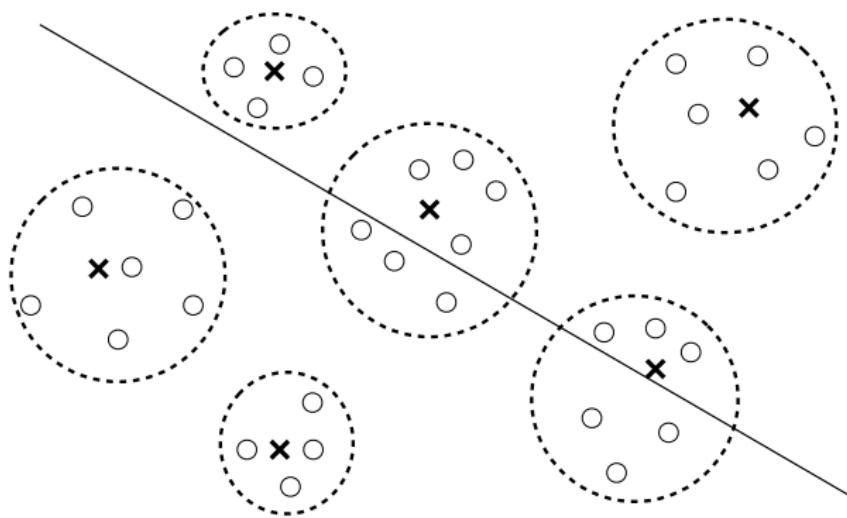
记 $C^{(t)} = \{C_1^{(t)}, \dots, C_K^{(t)}\}$ 为聚类的集合, $K^{(t)} = \{1, \dots, |K^{(t)}|\}$ 为当前聚类的下标,

1. 根据当前聚类情况 $C^{(t)}$, 构造新的数据集 $\mathbf{X}^{(t)}$, 求解相应的优化问题 $P^{(t)}$;
2. 检查解 $s^{(t)}$ 是否达到最优条件;
3. 如果不满足条件, 拆解当前聚类。



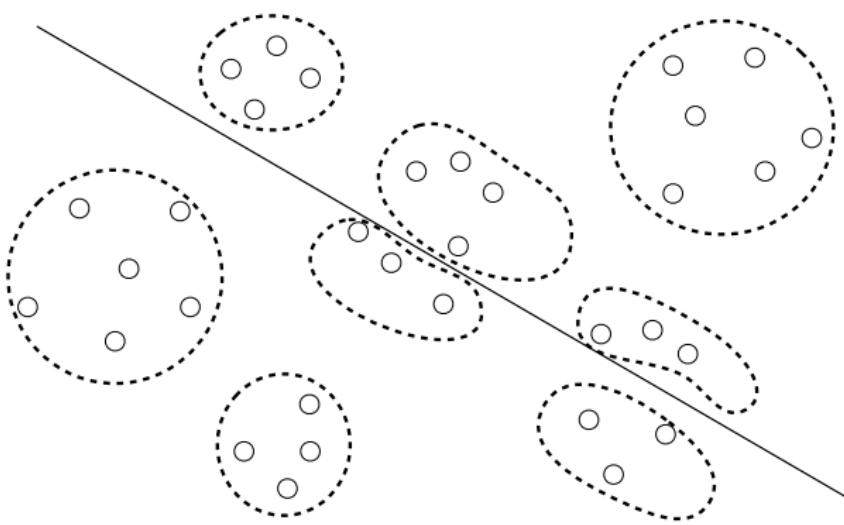
最小绝对值回归的优化1: 聚类——迭代拆解算法

- 初始聚类，使用任意聚类方法，我们这里使用K-means。



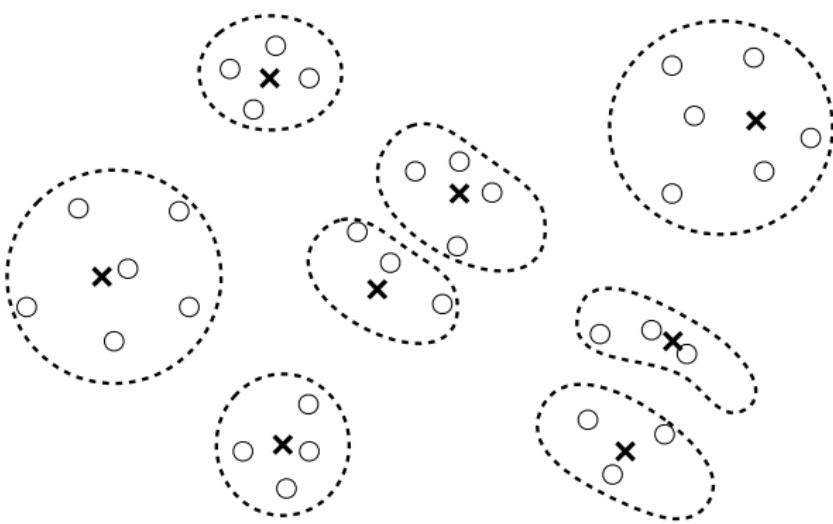
最小绝对值回归的优化1: 聚类——迭代拆解算法

- 计算 β , 按规则对聚类进行拆解。



最小绝对值回归的优化1: 聚类——迭代拆解算法

- 在新的聚类上重新计算 β 。



最小绝对值回归的优化1: 聚类——迭代拆解算法

- 可以证明，该方法最终获得原问题的最优解（和在全部数据上解原优化问题有相同的解）。

下面证明最后一次迭代的解 $\hat{\beta}^{(t)}$ 就是(3.4)的解 β^* ，

$$\begin{aligned}\xi^* &= \sum_{i \in I} |y_i - \sum_{j \in J} x_{ij} \beta_j^*| = \sum_{k \in K^{(t)}} \sum_{i \in C_k^{(t)}} |y_i - \sum_{j \in J} x_{ij} \beta_j^*| \\ &\geq \sum_{k \in K^{(t)}} \left| \sum_{i \in C_k^{(t)}} (y_i - \sum_{j \in J} x_{ij} \beta_j^*) \right| = \sum_{k \in K^{(t)}} |C_k^{(t)}| |y_k^{(t)} - \sum_{j \in J} x_{kj}^{(t)} \beta_j^*| \\ &\geq \sum_{k \in K^{(t)}} |C_k^{(t)}| |y_K^{(t)} - \sum_{j \in J} x_{kj}^{(t)} \hat{\beta}_j^{(t)}| = \sum_{k \in K^{(t)}} \left| \sum_{i \in C_k^{(t)}} y_i - \sum_{i \in C_k^{(t)}} \sum_{j \in J} x_{ij} \hat{\beta}_j^{(t)} \right| \\ &= \sum_{k \in K^{(t)}} \sum_{i \in C_k^{(t)}} |y_i - \sum_{j \in J} x_{ij} \hat{\beta}_j^{(t)}| = \sum_{i \in I} |y_i - \sum_{j \in J} x_{ij} \hat{\beta}_j^{(t)}| = \xi^{(t)}\end{aligned}$$

因为 $\hat{\beta}_j^{(t)}$ 是(3.4)的可行解，又显然 $\xi^* \leq \xi^{(t)}$ ，这就证明了 $\xi^* = \xi^{(t)}$ ，注意到 $\sum_{k \in K^{(t)}} |C_k^{(t)}| |y_K^{(t)} - \sum_{j \in J} x_{kj} \hat{\beta}_j^{(t)}|$ 就是 $F^{(t)}$ ，因此 $\xi^{(t)} = F^{(t)}$ 。因此最后一次迭代 $F^{(t)}$ 的最优解 $\hat{\beta}^{(t)}$ 就是原问题的最优解。



最小绝对值回归的优化2: 基于替代变量的估计算法

- Weidong Liu等人于2020年提出了一种基于替代变量的估计方法。
- 该方法可以优化一般的分位数回归问题，最小绝对值回归就是中位数回归，因此我们针对该特例给出步骤和推导。
- 这是一种拟牛顿法。该方法利用替代变量，转化为最小二乘问题，简化计算。



最小绝对值回归的优化2: 基于替代变量的估计算法

- 从牛顿-拉弗森迭代出发，对问题进行探究。

一般地，我们使用牛顿迭代法求解某随机优化问题：

$$\boldsymbol{\beta}^* = \arg \min_{\boldsymbol{\beta} \in \mathbb{R}^p} \mathbb{E}[G(\boldsymbol{\beta}; \mathbf{X}, Y)] \quad (3.11)$$

其中 $G(\boldsymbol{\beta}; \mathbf{X}, Y)$ 是损失函数， \mathbf{X} 和 Y 分别是 p 维自变量和一元响应变量， $\boldsymbol{\beta}$ 为回归系数。使用牛顿-拉弗森迭代来求解，

$$\tilde{\boldsymbol{\beta}}_1 = \boldsymbol{\beta}_0 - \mathbf{H}(\boldsymbol{\beta}_0)^{-1} \mathbb{E}[g(\boldsymbol{\beta}; \mathbf{X}, Y)] \quad (3.12)$$

其中 $\boldsymbol{\beta}_0$ 是一个初始估计， $g(\boldsymbol{\beta}; \mathbf{X}, Y)$ 为损失函数 $G(\boldsymbol{\beta}; \mathbf{X}, Y)$ 关于 $\boldsymbol{\beta}$ 的梯度。



最小绝对值回归的优化2: 基于替代变量的估计算法

- 当 β_0 和 β^* 十分接近时，可以做一点近似。

$\mathbf{H}(\boldsymbol{\beta}) := \partial \mathbb{E}[g(\boldsymbol{\beta}; \mathbf{X}, Y)] / \partial \boldsymbol{\beta}$ 表示 $\mathbb{E}G(\boldsymbol{\beta}; \mathbf{X}, Y)$ 的海森矩阵 (Hessian Matrix)。特别地，我们这里考虑损失函数为 L_1 损失的特殊情形，

$$G(\boldsymbol{\beta}; \mathbf{X}, Y) = \rho(Y - \mathbf{X}^T \boldsymbol{\beta}) \quad (3.13)$$

在(3.13)的情形下， $g(\boldsymbol{\beta}; \mathbf{X}, Y) = \mathbf{X}(\mathbb{I}[Y - \mathbf{X}^T \boldsymbol{\beta} < 0] - 0.5)$ 。

并且 $\mathbf{H}(\boldsymbol{\beta}) = \mathbb{E}(\mathbf{X} \mathbf{X}^T f(\mathbf{X}^T (\boldsymbol{\beta} - \boldsymbol{\beta}^*)))$ ，这里 $f(x)$ 是噪声 e 的密度函数。当初始估计量 β_0 和 β^* 很接近时， $\mathbf{H}(\beta_0)$ 就会很接近 $\mathbf{H}(\beta^*) = \Sigma f(0)$ ，这里 $\Sigma = \mathbb{E} \mathbf{X} \mathbf{X}^T$ 是 \mathbf{X} 的协方差矩阵。使用 $\mathbf{H}(\beta^*)$ 替换 $\mathbf{H}(\beta_0)$ ，可得

$$\beta_1 = \beta_0 - \mathbf{H}(\beta^*)^{-1} \mathbb{E}[g(\boldsymbol{\beta}; \mathbf{X}, Y)] = \beta_0 - \Sigma^{-1} f^{-1}(0) \mathbb{E}[g(\beta_0; \mathbf{X}, Y)] \quad (3.14)$$



最小绝对值回归的优化2: 基于替代变量的估计算法

- 在这种近似下，牛顿迭代依然可以逼近最优解。

在 β^* 处对 $\mathbb{E}[g(\beta_0; \mathbf{X}, Y)]$ 进行泰勒展开，

$$\begin{aligned}\mathbb{E}[g(\beta_0; \mathbf{X}, Y)] &= \mathbf{H}(\beta^*)(\beta_0 - \beta^*) + O(|\beta_0 - \beta^*|_2^2) \\ &= \Sigma f(0)(\beta_0 - \beta^*) + O(|\beta_0 - \beta^*|_2^2)\end{aligned}$$

结合(3.14)，可以得到

$$\begin{aligned}|\beta_1 - \beta^*|_2 &= |\beta_0 - \Sigma^{-1}f^{-1}(0)(\Sigma f(0)(\beta_0 - \beta^*) + O(|\beta_0 - \beta^*|_2^2)) - \beta^*|_2 \\ &= O(|\beta_0 - \beta^*|_2^2)\end{aligned}$$

因此，如果我们得到一个 β^* 的一致估计量 β_0 ，我们就可以通过(3.14)得到偏误更小的估计。



最小绝对值回归的优化2: 基于替代变量的估计算法

- 利用替代变量，转化为一个最小二乘问题。

$$\begin{aligned}\beta_1 &= \Sigma^{-1}(\Sigma\beta_0 - f^{-1}(0)\mathbb{E}[g(\beta_0; X, Y)]) \\ &= \Sigma^{-1}\mathbb{E}[X\{X^T\beta_0 - f^{-1}(0)(\mathbb{I}[Y \leq X^T\beta_0] - 0.5)\}]\end{aligned}\tag{9}$$

这里我们定义一个新的响应变量 \tilde{Y} ,

$$\tilde{Y} = X^T\beta_0 - f^{-1}(0)(\mathbb{I}[Y \leq X^T\beta_0] - 0.5)\tag{10}$$

那么 $\beta_1 = \Sigma^{-1}\mathbb{E}(X\tilde{Y})$ 就是线性回归问题 $\tilde{Y} = X^T\beta$ 的最优回归系数，即

$$\beta_1 = \arg \min_{\beta \in \mathbb{R}^p} \mathbb{E}(\tilde{Y} - X^T\beta)^2\tag{11}$$



最小绝对值回归的优化2: 基于替代变量的估计算法

使用替代变量迭代算法方法求解最小绝对值回归问题(SVN, Substitue Variable Newton method)

输入: Y 和 X 的样本 $Y = (Y_1, Y_2, \dots, Y_n)$, $X = (X_1^T, X_2^T, \dots, X_n^T)$, 迭代次数 τ , 核函数 $K(x)$, 依赖于迭代次数的带宽 $h^{(t)}$, ($t = 1, \dots, \tau$)。

初始化: 给出初始相合估计量 $\hat{\beta}^0$, 将样本划分成 J 个均等子集, 样本量均为 m , 为了给出最小绝对值回归估计量的相合估计, 我们取任一子集里面的数据, 直接使用最小一乘法估计 $\hat{\beta}^0$ 。

对于 $t = 1, \dots, \tau$: 取新的数据子集, 在该数据集上

1. 计算 $\hat{f}^{(t)}(0)$,

$$\hat{f}^{(t)}(0) = \frac{1}{mh} \sum_{i=1}^m K\left(\frac{Y_i - X_i^T \hat{\beta}^{(t-1)}}{h^{(t)}}\right)$$

2. 计算 $\tilde{Y} = (\tilde{Y}_1, \tilde{Y}_2, \dots, \tilde{Y}_m)$,

$$\tilde{Y}_i = X_i^T \hat{\beta}^{(t-1)} - \hat{f}^{(t)}(0)^{-1} (\mathbb{I}[Y_i \leq X_i^T \hat{\beta}^{(t-1)}] - 0.5)$$

3. 计算 $\hat{\beta}^{(t)}$,

$$\hat{\beta}^{(t)} = \arg \min_{\beta \in \mathbb{R}^p} \frac{1}{m} \sum_{i=1}^m (\tilde{Y}_i - X_i^T \beta)^2$$

输出: $\hat{\beta}^{(\tau)}$



算法的性能比较实验

- 利用模拟数据，观察和测试算法的运算性能。
- SVN算法为拟牛顿法，这里先比较不设置终止条件下的结果。

Table: 高斯噪声下三种估计算法的性能比较

n	p	AID			SVN		LP	
		r ⁰ (%)	r ^T (%)	τ	Time(sec)	τ	Time(sec)	Time(sec)
20000	10	0.05	2.90	6	2	3	0.38	0.20
20000	50	0.15	12.00	12	13	7	6	3
20000	100	0.15	12.00	12	38	11	27	21
20000	200	0.66	21.55	9	118	8	76	126
20000	500	0.80	28.00	11	535	3	322	813
40000	10	0.05	2.90	6	3	3	0.74	0.23
40000	50	0.15	12.00	12	31	7	14	7
40000	100	0.15	12.00	12	84	8	56	48
40000	200	0.66	21.55	9	210	8	154	258
40000	500	0.80	28.00	11	1965	3	1143	3446
200000	100	0.80	11.00	11	134	3	361	496
200000	200	0.80	28.00	11	2476	3	1631	3329



算法的性能比较实验

- 比较SVN算法设置终止条件下的结果。

Table: 部分性能结果对比

s	n	p	τ	Time(SVN)	Time(SVN-full)	Time(LP)	Time(AID)	l_2 -error(SVN)	l_2 -error(LS)	l_2 -error(LP)
5	20000	10	12	0.12	0.42	0.17	2	0.0023	0.7304	0.0024
5	20000	50	8	1	7	3	13	0.0039	3.1548	0.0020
5	20000	100	14	10	28	19	41	0.0059	15.0373	0.0033
5	20000	200	27	81	130	170	124	0.0188	151.4039	0.0028
5	20000	500	40	437	437	1119	698	0.0259	245.0070	0.0026



算法的性能比较实验

- 观察 SVN 算法的收敛性

$$h^{(t)} = \sqrt{\frac{s \log n}{n}} + s^{-1/2} \left(\frac{s^2 \log n}{10m} \right)^{(t+1)/2},$$

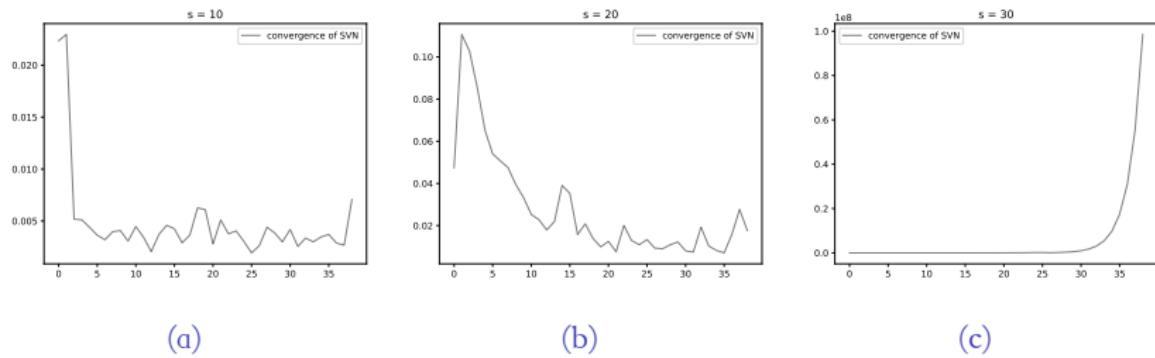


Figure: SVN 算法的收敛情况



算法的性能比较实验—总结

- SVN算法在计算性能上很有优势。
- SVN算法作为拟牛顿法不能保证最终收敛到最优解。
可以设置适当的终止条件。
- AID算法最后获得与对全部数据求线性规划相同的解，
它在数据量较大时也有计算性能的优势。



改进交替凸优化算法

- 可利用SVN算法改进交替凸优化算法。
- 可利用AID算法改进交替凸优化算法。



改进交替凸优化算法—模拟实验

- 产生一个服从因子模型的模拟数据集。
- 比较ACP-LP, ACP-SVN, ACP-AID的计算时间和重构误差中位数。

Table: 三种算法的比较结果

			ACP		ACP-SVN		ACP-AID		PCA-L2	
m	n	p	Time	l2-error	Time	l2-error	Time	l2-error	Time	l2-error
5	1000	50	47	0.0083	19	0.0115	129	0.0083	0.39	
10	1000	50	68	0.0064	17	0.0096	104	0.0064	0.58	
5	5000	50	119	0.0382	31	0.1232	278	0.0382	1.24	
10	5000	50	107	0.0259	79	0.1352	321	0.0259	0.85	
5	5000	100	689	0.0725	362	0.1480	1782	0.0725	1.26	
10	5000	100	792	0.0664	510	0.2394	1440	0.0664	2.99	
5	2000	200	3567	0.0992	1135	0.4903	3290	0.0992	2.07	
10	2000	200	3893	0.1033	1650	0.5790	5033	0.1033	3.12	



改进交替凸优化算法—结论

- SVN算法改进交替凸优化算法具有较明显的计算性能改进。
- 可利用AID算法改进交替凸优化算法，但是在数据规模较小时不建议使用。
- 当观测和维数都很大时，交替凸优化算法的计算开销仍非常大。



一种最大化特征空间投影L₁范数的贪婪算法

- 交替凸优化算法可以解决问题P₃，问题P₄也是L₁主成分分析。
- 回顾P₄问题的表述：

$$P_4 : \hat{A} = \arg \max_{A} \|A^T X\|_{L_1} = \arg \max_{A} \sum_{i=1}^n \sum_{k=1}^m \left| \sum_{j=1}^p a_{jk} x_{ij} \right|, \text{ 其中 } A^T A = I_m \quad (12)$$

- 当m不为1时，是一个非常复杂的问题。



一种最大化特征空间投影L₁范数的贪婪算法

- Kwak 于2009年提出一种贪心算法，可以求解该问题。
- 首先考虑m为1的情况，在该条件下可以给出一个求局部最优解的方法。

算法 4.2 $m = 1$ 时的 P_4 求解算法

初始化：取 \mathbf{a} 的任意初始值； $\mathbf{a}^{(0)} \leftarrow \mathbf{a}^{(0)} / \|\mathbf{a}^{(0)}\|_{L_2}$ 。

对于 $t = 1, \dots, \tau$ ：

1. 对所有的 $i \in \{1, \dots, n\}$ ，若 $(\mathbf{a}^{(0)})^T \mathbf{x}_i < 0$ ，令 $p_i^{(t)} = -1$ ，否则 $p_i^{(t)} = 1$ 。
2. 令 $t \leftarrow t + 1$ 并且 $\mathbf{a}^{(t)} = \sum_{i=1}^n p_i^{(t-1)} \mathbf{x}_i$ ； $\mathbf{a}^{(t)} \leftarrow \mathbf{a}^{(t)} / \|\mathbf{a}^{(t)}\|_{L_2}$ 。
3. 检查是否收敛：
 - 1) 如果 $\mathbf{a}^{(t)} \neq \mathbf{a}^{(t-1)}$ ，回到步骤 2；
 - 2) 若存在某个 i 使得 $\mathbf{a}^T \mathbf{x}_i = 0$ ，令 $\mathbf{a}^{(t)} \leftarrow (\mathbf{a}^{(t)} + \Delta \mathbf{a}) / \|\mathbf{a}^{(t)} + \Delta \mathbf{a}\|_{L_2}$ 并回到步骤 2，这里 $\Delta \mathbf{a}$ 为一个很小的随机向量；
 - 3) 若前面两个都不符合，令 $\mathbf{a}^* = \mathbf{a}^{(t)}$ ，算法返回。

- 可以证明该算法在有限步骤内收敛到一个局部最优解。



一种最大化特征空间投影L₁范数的贪婪算法

- 利用贪婪策略，每次求解m为1的子问题，最终得出n个主成分。

算法 4.3 求解特征空间 L₁ 范数最大化的贪婪算法

输入： $\mathbf{a}^0, \{\mathbf{x}_i^0 = \mathbf{x}_i\}_{i=1}^n$ 。

对于 $j = 1, \dots, m$:

对所有的 $i \in \{1, \dots, n\}$, $\mathbf{x}_i^j = \mathbf{x}_i^{j-1} - \mathbf{a}^{j-1}(\mathbf{a}^{(j-1)T} \mathbf{x}_i^{j-1})$,

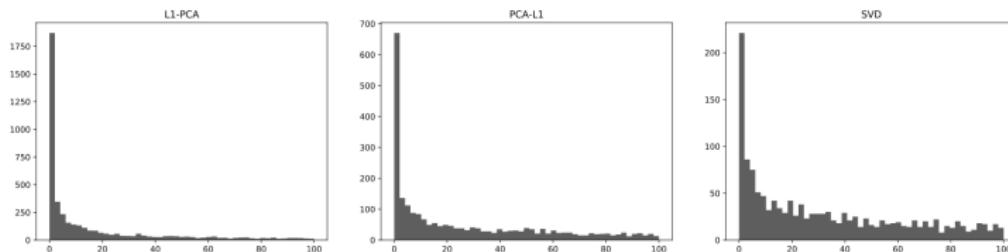
通过算法 4.2 求解 \mathbf{a}_j , 令 $\mathbf{X}^j = [\mathbf{x}_1^j, \mathbf{x}_2^j, \dots, \mathbf{x}_i^j]$ 。

- 可以证明，算法中每次得到的主成分之间正交。



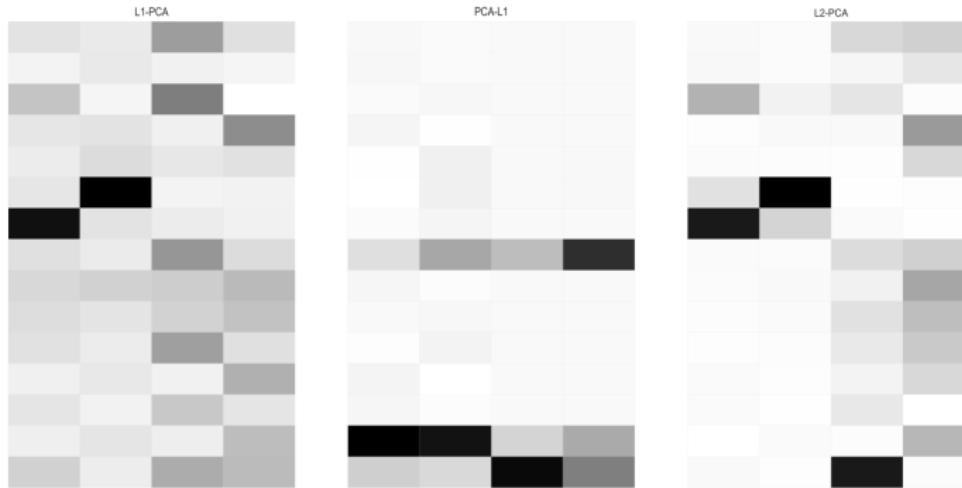
数值模拟实验

- 通过数值模拟实验，希望观察两种 L_1 主成分分析和 L_2 主成分分析在稳健性和载荷矩阵上的对比。
- 可以发现，无论是哪一种 L_1 主成分分析都是对离群值稳健的。



数值模拟实验

- 三种主成分分析得到的主成分是不同的。
- 将得到的因子载荷矩阵进行最大方差旋转，可更易看出区别。



实证研究——扩散因子模型进行宏观经济预测

- 将两种不同的L₁主成分分析都应用于扩散指数模型。
- 比较预测的准确性，结论是两种L₁因子都有良好的预测效果，均适用于稳健因子分析。

Table: 向前一个月预测结果

	MSE (L1 PCA)	MSE (PCA L1)	MAE(L1 PCA)	MAE (PCA L1)	MPAE(L1 PCA)	MPAE (PCA L1)
消费者满意指数	0.81	0.78	0.87	0.83	0.43	0.67
工业生产者出厂价格指数	0.70	0.83	0.75	0.92	0.96	0.90
货币供应量M2	0.76	1.01	0.90	1.00	0.45	0.92
固定资产投资总额	0.89	1.11	0.97	1.03	0.81	1.05
房地产开发投资总额	0.79	0.94	0.95	0.97	0.91	1.10
社会消费品零售总额	0.84	0.62	0.87	0.78	0.45	0.76
制造业采购经理指数	1.24	0.99	1.06	1.14	0.90	0.82
住宅新开工面积总数	0.89	1.20	0.85	1.12	0.45	1.19
股票流通市值	0.99	1.23	0.98	1.12	0.81	0.98
消费者信心指数	0.93	0.70	0.98	0.80	0.98	0.99

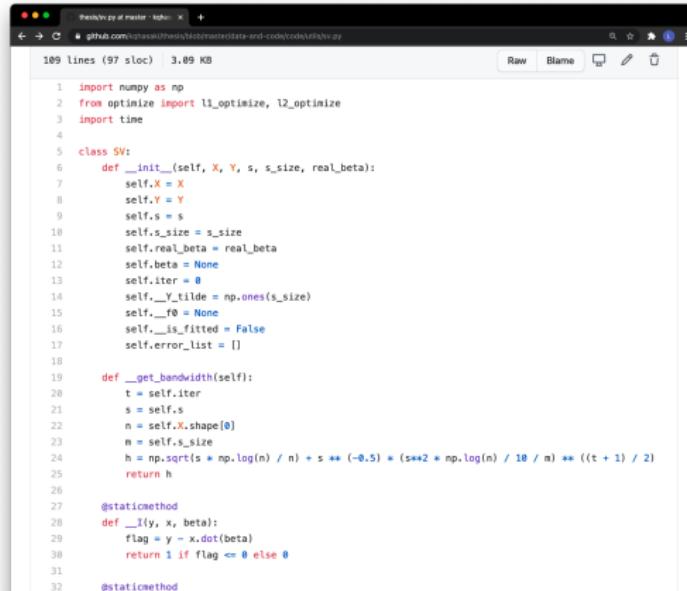


注意

- 本文所有实验均用Python编程实现了各算法伪代码。

算法的计算性能与实现程序的质量关联很大，本文的运行时间仅做实验对比，不代表算法的标准性能。

以SVN算法的一次实现为例给出部分示例代码：



```

1 import numpy as np
2 from optimize import l1_optimize, l2_optimize
3 import time
4
5 class SV:
6     def __init__(self, X, Y, s, s_size, real_beta):
7         self.X = X
8         self.Y = Y
9         self.s = s
10        self.s_size = s_size
11        self.real_beta = real_beta
12        self.beta = None
13        self.iter = 0
14        self._Y_tilde = np.ones(s_size)
15        self._r0 = None
16        self._is_fitted = False
17        self.error_list = []
18
19    def _get_bandwidth(self):
20        t = self.iter
21        s = self.s
22        n = self.X.shape[0]
23        m = self.s_size
24        h = np.sqrt(s * np.log(n) / n) + s ** (-0.5) * (s**2 * np.log(n) / 10 / m) ** ((t + 1) / 2)
25        return h
26
27    @staticmethod
28    def __l1y(x, beta):
29        flag = y - x.dot(beta)
30        return 1 if flag <= 0 else 0
31
32

```



注意

```

  thesis.py at master · kohn/...
  https://github.com/kohn/thesis/blob/master/src-code/code/thesis.py
  66     X = self.X
  67     Y = self.Y
  68     if (size(t) >= X.shape[0]):
  69         return (None, None)
  70     if (size((t+1)) > X.shape[0]):
  71         return (X[size*t:], Y[size*t:])
  72     else:
  73         return (X[size*t:size*(t+1)], Y[size*t:size*(t+1)])
  74
  75     def __init_beta(self):
  76         X, Y = self.__get_curXY()
  77         ret = l2_optimize(X, Y.T)
  78         beta = ret.x
  79         self.beta = beta
  80         self.itev += 1
  81         return beta
  82
  83     def __call_stop(self, new_beta):
  84         a = new_beta
  85         b = self.beta
  86         cosangle = a.dot(b)/(np.linalg.norm(a) * np.linalg.norm(b))
  87         return np.arccos(cosangle) < 1e-3
  88
  89     def fit(self):
  90         if self.__is_fitted:
  91             return
  92         self.__init_beta()
  93         start_time = time.time()
  94         while True:
  95             X, Y = self.__get_curXY()
  96             if (X is None): break
  97
  98             self.__get_f0()
  99             self.__get_Y_tilde()
 100             new_beta = l2_beta is not None:
 101                 self.error_list.append(sum(np.square(new_beta - self.real_beta)))
 102             if self.__call_stop(new_beta):
 103                 break
 104             self.beta = new_beta
 105             self.itev += 1
 106             end_time = time.time()
 107             self.__is_fitted = True
 108             self.time = end_time - start_time
 109

```



论文的创新点

- 扩散指数模型进行经济预测时，使用 L_2 主成分估计因子；本文创新性地使用 L_1 主成分估计得到 L_1 因子，使得在处理重尾宏观经济数据时，预测更加准确、稳健性更好。
- 本文介绍并检验了两种最新的最小绝对值回归优化算法，并给出了使用建议。
- 利用最小绝对值回归优化算法，本文改进了经典的采用线性规划的交替凸优化算法，起到了一定的效率改进作用。



结束语

感谢到场的各位专家、教授、老师和同学！

