Bayes **Inference, S.A.**
c/ Barco, Nº10, 1ºD, E-28004 MADRID
Tfn (91) 532.74.40, Fax (91) 532.26.36

Informe sobre la implementación de la estimación ARIMA

Copyright	2003, ***Bayes*** Inference, S.A.
Título	Informe sobre la implementación de la estimación ARIMA
Asunto	Estimación máximo-verosímil de modelos ARIMA de función de transferencia con interrupciones
Clave	Función de verosimilitud, modelos función de transferencia, interrupciones
Archivo	C:\users\vdebuen\prj\tolp\trunk\doc\modeling\ARIMA_Estimate\InformeImplementaciónDeLaEstimacionARIMA_Mar2004.doc
Edición	Función de verosimilitud, modelos función de transferencia, interrupciones
Impresión	Función de verosimilitud, modelos función de transferencia, interrupciones
Distribución	Interna

1.La función objetivo

En el documento estconint3.doc se da una propuesta de estimación máximo-verosímil de modelos ARIMA de función de transferencia con interrupciones. El modelo implementado para la función Estimate de TOL no parte exactamente del modelo ARIMA con función de transferencia de Box-Jenkins

$$r_t = z_t - \sum_i \frac{\omega_i(B)}{\delta_i(B)} x_{i,t} = \frac{\theta(B)}{\phi(B)} a_t$$

sino de la restricción a transferencias lineales

$$r_t = z_t - \sum_i \omega_i(B) x_{i,t} = \frac{\theta(B)}{\phi(B)} a_t$$

Se trata de maximizar la función de verosimilitud,

$$Max\left\{ f(Z|\alpha,\sigma^2) = \left(2\pi\sigma^2\right)^{\frac{-N}{2}} |\Sigma|^{-\frac{1}{2}} e^{\frac{-1}{2\sigma^2} R^T \Sigma^{-1} R} \right\}$$ o bien su logaritmo

$$Max\left\{ \log(f(Z|\alpha,\sigma^2)) = \frac{-N}{2}(\log 2\pi + \log\sigma^2) - \frac{1}{2}\log|\Sigma| - \frac{1}{2\sigma^2} R^T \Sigma^{-1} R \right\}$$

donde $R \in \Re^N$ es el ruido diferenciado y $\sigma^2 \Sigma \in \Re^{N \times N}$ su matriz de covarianzas.

Maximizando respecto de σ^2, tenemos que

$$\frac{-N}{2\sigma^2} + \frac{R^T \Sigma^{-1} R}{2\sigma^4} = 0 \Rightarrow \sigma^2 = \frac{R^T \Sigma^{-1} R}{N}$$

$$Max\left\{ f(Z|\alpha) = \left(2\pi \frac{R^T \Sigma^{-1} R}{N} \right)^{\frac{-N}{2}} |\Sigma|^{-\frac{1}{2}} e^{\frac{-N}{2}} = \left(\frac{2\pi e}{N} \right)^{\frac{-N}{2}} \left(R^T \Sigma^{-1} R \right)^{\frac{-N}{2}} |\Sigma|^{-\frac{1}{2}} \right\}$$

lo cual es equivalente a minimizar esta otra función mucho más sencilla

$$Min\left\{ R^T \Sigma^{-1} R \ \ |\Sigma|^{\frac{1}{N}} \right\}$$

a la que llamaremos función objetivo del problema. Hasta ahora, se había utilizado como función objetivo del modelo la suma de cuadrados de los residuos, lo cual equivale a minimizar su norma euclídea

$$Min\left\{ \sqrt{\tfrac{1}{N} A^T A} \right\}$$

La norma que aparecerá ahora como objetivo será

$$Min\left\{ \sqrt{\tfrac{1}{N} R^T \Sigma^{-1} R \ \ |\Sigma|^{\frac{1}{N}}} \right\}$$

Se había detectado que las raíces MA tendían a la unidad, especialmente en series cortas. Esto se debía a dos defectos de la versión utilizada:

- No se tenía en cuenta el determinante de la matriz de autocovarianzas con lo que la aproximación a la función de verosimilitud empeora si dicho determinante se aleja de 1, lo cual ocurre cuando las raíces de los polinomios se acercan al círculo unidad.
- Además, a pesar de que se calculaban bien los residuos iniciales, aunque se hacía condicionando al ruido diferenciado, luego no se tenían en cuenta y se seguía minimizando la suma de cuadrados de los residuos en lugar de la suma ampliada.

Los residuos se pueden calcular en toda la muestra abarcada por el ruido diferenciado, como su esperanza condicionada a dicho ruido diferenciado, puesto que se conoce la distribución conjunta de ambos, mediante la ecuación

$$A = \Psi^T \Sigma^{-1} R = \Psi^T R^* \wedge R^* = \Sigma^{-1} R$$

El algoritmo de Levinson permite a un tiempo calcular el determinante de Σ y resolver el sistema $\Sigma R^* = R$ con $4N^2$ operaciones pero existen métodos que realizan ambas tareas con complejidad $O(N\log(N))$. El producto $\Psi^T R^*$ es en realidad una ecuación en diferencias hacia atrás con valores de inicialización nulos por lo que la complejidad es sólo $O(N(p+q))$.

Para poder seguir utilizando Marquardt y el jacobiano analítico se minimiza la siguiente suma de cuadrados

$$Min\left\{ F(\alpha) = A^{*T}A^* = |\Sigma|^{\frac{1}{N}} R^T\Sigma^{-1}R = |\Sigma|^{\frac{1}{N}} R^T R^* \right\} \wedge A^* = \lambda A$$

$$\lambda = |\Sigma|^{\frac{1}{2N}} \sqrt{\frac{R^T\Sigma^{-1}R}{A^TA}} \wedge \frac{R^T\Sigma^{-1}R}{A^TA} = \frac{R^T\Sigma^{-1}R}{R^TM^TMR} \approx \frac{|\Sigma^{-1}|}{|M|^2} \Rightarrow \frac{\partial a_t^*}{\partial \varpi_{ik}} \approx \lambda \frac{\partial a_t}{\partial \varpi_{ik}} \wedge M = \Psi^T\Sigma^{-1}$$

puesto que la matriz de autocovarianzas Σ es independiente de las funciones de transferencia y el cociente de normas casi lo es también. Las derivadas parciales con respecto a los valores ARMA se calculan siempre numéricamente sin recalcular el ruido R que sólo depende de los ω_{ik}.

Estas aproximaciones atañen sólo al cálculo del jacobiano que, en cualquier caso, se utiliza para dar una aproximación del hessiano, con lo cual no afecta a la convergencia de Marquardt. En el cálculo de la función de verosimilitud no hay ninguna aproximación, luego se trata de máxima verosimilitud exacta. En la traza de seguimiento del algoritmo de Marquardt se presentan los valores correspondientes a las componentes de la función objetivo

$$\sqrt{\tfrac{1}{N}A^{*T}A^*} = \sqrt{\tfrac{1}{N}A^TA}\ |\Sigma|^{\frac{1}{2N}}\ \sqrt{R^T\Sigma^{-1}R\left(A^TA\right)^{-1}}$$

Se ha modificado la función TOL `ARIMACalcResiduals` y renombrado como `ARIMALevinsonEval` para que efectúe todos los cálculos relacionados con la evaluación de los residuos máximo verosímiles y de todos los cálculos anteriores. Esta función ha sido comprobada usando un programa TOL de simulación dando muy buenos resultados. Mediante esta función el usuario puede implementar en TOL estimadores personalizados de modelos estocásticos con componentes ARIMA que no se puedan reducir al caso aquí tratado.

2.La restricción de estacionariedad de los polinomios

En un modelo ARIMA los polinomios AR y MA deben cumplir la restricción de estacionariedad de polinomios de retardo, consistente en que todas sus raíces deben estar fuera del círculo unidad. Cuando esto no es así la función de covarianzas no existe si se trata de raíces AR o no es invertible si son raíces MA.

Además basta con que las raíces se aproximen demasiado al círculo unidad, para que el cálculo de los residuos iniciales, así como de la matriz de covarianzas y su determinante, dé lugar a graves errores de cálculo. Debido a esto, cuando un método iterativo de optimización se acerca a estas situaciones corre el riesgo de atascarse, razón por la cual se estiman a menudo raíces sospechosamente cercanas a la unidad, incluso aunque el autocorrelograma del ruido diferenciado no lo justifique.

Esta restricción supone un grave problema para los métodos iterativos de optimización, que funcionan bien sólo en problemas no restringidos y con las condiciones de continuidad y derivabilidad aseguradas.

A continuación, y en virtud de la propia definición de polinomio estacionario, se va a construir una transformación biunívoca, continua y derivable que convierte el problema inicial en un problema sin restricciones.

Un polinomio cualquiera expresado como

$$P(x) = \sum_{k=0}^{n} a_k x^{n-k}$$

es estacionario, es decir, tiene todas sus raíces fuera del círculo unidad, si, y sólo si, el polinomio de coeficientes en orden inverso

$$Q(x) = \sum_{k=0}^{n} a_k x^k$$

las tiene todas dentro del círculo unidad, o sea, es de Schur. Según la proposición debida a Duffin, [Bandorf-Nielsen, Schou, 1973], $Q(x)$ es un polinomio de Schur si y sólo si

$|a_0| < |a_n|$ y el polinomio $Q'(x) = \sum_{k=0}^{n-1} a'_k x^k = \sum_{k=0}^{n-1} (a_n a_{k+1} - a_0 a_{n-k-1}) x^k = \frac{1}{x}(a_n Q(x) - a_0 P(x))$

es de Schur. Por un lado, se tiene que

$$a'_{n-1} = a_n a_n - a_0 a_0 = a_n^2 - a_0^2 > 0$$

y por otro lado, se tienen las ecuaciones en ambos sentidos

$$\left.\begin{aligned} a'_k &= a_n a_{k+1} - a_0 a_{n-k-1} \\ a'_{n-k-2} &= a_n a_{n-k-1} - a_0 a_{k+1} \end{aligned}\right\} \forall\, k = 0 \ldots n-2$$

Multiplicando la primera por a_n y la segunda por a_0 resulta

$$\left.\begin{array}{rrrr} a'_k a_n = & a_n^2 a_{k+1} & - a_0 a_n a_{n-k-1} \\ a'_{n-k-2} a_0 = & a_0 a_n a_{n-k-1} & - a_0^2 a_{k+1} \end{array}\right\} \forall\, k = 0 \ldots n-2$$

y sumándolas

$$a'_k a_0 + a'_{n-k-2} a_n = \left(a_n^2 - a_0^2\right) a_{k+1} = a'_{n-1} a_{k+1} \forall\, k = 0 \ldots n-2$$

$$a_{k+1} = \frac{a'_k a_0 + a'_{n-k-2} a_n}{a'_{n-1}} \forall\, k = 0 \ldots n-2$$

Puesto que no hay ninguna pérdida de generalidad en obligar a que sea $a_n = 1$, dados un polinomio de Schur $Q'(x)$ y un número b cualquiera, se puede construir el polinomio de Schur $Q(x)$ mediante la fórmula

$$a_0 = \frac{b}{\sqrt{1+b^2}} \Rightarrow |a_0| < 1 = a_n$$

$$a_{k+1} = \frac{a'_k a_0 + a'_{n-k-2}}{a'_{n-1}} \forall\, k = 0 \ldots n-2$$

Para evitar problemas numéricos se puede truncar a_0 por $\pm|1-\varepsilon|$. Así pues dado un vector $(b_1, b_2, \ldots b_n)$ de números cualesquiera se puede construir unívocamente un polinomio de Schur de forma recursiva partiendo inicialmente del polinomio 1.

Recíprocamente, dado un polinomio de Schur, se puede calcular el vector generador de forma recursiva en orden decreciente tomando en cada iteración

$$b = a_0 \Big/ \sqrt{1 - a_0^2}$$

Previamente a cada paso se debe dividir el polinomio por el coeficiente de grado mayor para obligar a que sea $a_n = 1$.De este modo se consigue una transformación biunívoca, continua y derivable que convierte el problema inicial en un problema sin restricciones y mucho mejor condicionado en el caso de raíces cercanas al círculo unitario.

Lógicamente, esta transformación sólo es un cambio de variable válido si todos los coeficientes del polinomio desde el grado 1 hasta el n entran en juego. Si se quiere estimar modelos con polinomios incompletos, el sistema dará un mensaje de aviso y sólo efectuará la transformación sobre los que estén completos. Esto no afecta a los polinomios estacionales, es decir, polinomios en B^s, pues en tal caso la transformación es perfectamente válida reduciendo previamente el polinomio, a no ser que falten términos en B^{ks} para cierto $k = 1 \ldots n-1$.

Evidentemente, al implementar este algoritmo en un ordenador con aritmética discreta, vuelve a convertirse en un problema restringido pero con mayor margen operativo para aprovechar la precisión de la máquina, es decir, no se evita por completo el problema pero se reducen mucho sus consecuencias nocivas para la optimización. Al final del proceso de minimización, a efectos de construir la matriz de información y los estadísticos correspondientes, se pueden recalcular las derivadas parciales con respecto a los parámetros ARMA sin transformar.

3.El paso lineal en la optimización no lineal

El método de Marquardt, al igual que otros métodos iterativos de optimización no lineal, requiere la solución de sistemas lineales, que frecuentemente, y sobre todo cuando crece el número n de variables, pueden estar mal condicionados debido a menudo a la presencia de altas correlaciones múltiples.

Debido a ello, el sistema lineal a resolver en cada iteración de Marquardt

$$\left(J^T J - \lambda^2 I_n\right)\nabla x = -J^T a$$

aunque es regular por definición, puede presentar graves problemas numéricos para valores pequeños de λ , y forzar a que aparezcan valores grandes que no conllevan avances significativos.

Para resolver el problema se parte de la descomposición de valor singular

$J = UDV^T \in \Re^{N\times n}$ donde $U \in \Re^{N\times n} \wedge D \in \Re^{n\times n} \wedge V \in \Re^{n\times n}$; siendo

$U^T U = V^T V = I_n$ y D es diagonal. Es fácil demostrar que

$\left(J^T J + \lambda^2 I_n\right)^{-1} = V\left(D^2 + \lambda^2 I_n\right)^{-1} V^T$ puesto que

$$\left(D^2 + \lambda^2 I_n\right)^{-1} = VD'V^T \Leftrightarrow\ I_n = \left(VD'V^T\right)\left(\lambda^2 I_n + VD^2V^T\right) = \lambda^2 VD'V^T - VD'D^2V^T = VD'\left(D^2 + \lambda^2 I_n\right)V^T \Leftrightarrow$$
$$I_n = V^T I_n V = V^T VD'\left(D^2 + \lambda^2 I_n\right)V^T V = D'\left(D^2 + \lambda^2 I_n\right) \Leftrightarrow\ D' = \left(D^2 + \lambda^2 I_n\right)^{-1}$$
Por lo tanto $\nabla x = -V\left(D^2 + \lambda^2 I_n\right)^{-1} V^T VDU^T a = -V\left(D^2 + \lambda^2 I_n\right)^{-1} DU^T a = -V\left(D + \lambda^2 D^{-1}\right)^{-1} U^T a$

Obsérvese que Marquardt requiere la solución del sistema para diferentes valores de λ^2 , lo cual en este caso no ocasiona apenas coste adicional. Ahora bien, la descomposición de valor singular es muy costosa cuando crece el número de variables, y más aún si el sistema está mal condicionado. Para evitar este problema, se tendrá en cuenta, que de una iteración a otra el jacobiano es distinto pero, puesto que suele tener un importante peso de variables lineales, hay una parte del mismo que apenas varía. Por ello si se guardan las matrices U' y V' de la iteración anterior, la matriz $D' = U'^T JV' \in \Re^{n\times n}$ no será diagonal, pero se acercará bastante,

estará mucho mejor condicionada y además se reduce la dimensión del problema, todo lo cual reduce sustancialmente el cálculo, dispuesto del siguiente modo:

$$D' = U''DV''^T \Rightarrow J = U'D'V'^T = U'U''DV''^TV'^T \Rightarrow U = U'U'';V = V'V''$$

Esto a su vez introduce errores de redondeo por lo que debe buscarse un compromiso entre la eficiencia y la calidad de los resultados mediante las pertinentes pruebas de simulación. Por el momento no se hace uso de esta posibilidad de aceleración.

Por otro lado, aunque la dirección dada por Marquardt sea una dirección de descenso, no deja de ser una aproximación lineal de la función objetivo real, por lo que la magnitud del paso puede no ser la más adecuada, y es conveniente encontrar el valor de $h > 0$ que da el mínimo de la función objetivo

$$\underset{h>0}{Min}\left\{ F(\alpha + h\nabla\alpha) = \sum_{t=1}^{N-q} f'(\alpha + h\nabla\alpha)^2 \right\}$$

Esta búsqueda unidireccional se puede implementar de un modo sencillo por el método de interpolación polinómica de orden cúbico con sólo una evaluación extra.

Aún cuando en esta dirección no se encuentre una mejora significativa, es posible que exista una dirección mejor en situaciones de alta correlación. Dada la descomposición de valor singular expuesta anteriormente se puede aplicar el método del gradiente conjugado (CG Conjugate Gradient), minimizando sucesivamente en las direcciones de los autovectores de J^TJ, que son las columnas de la matriz V. Si los ordenamos previamente en cuanto a la cantidad de información que ofrecen sobre los residuos, es decir, del valor absoluto del vector DU^Ta, entonces, en principio basta con explorar sólo unas pocas direcciones para comprobar si hay alguna dirección de descenso significativo.

El usuario dispone de la variable global `Real CGMaxIter` que indica el número máximo de direcciones conjugadas a explorar cuando la iteración de Marquardt no produce un avance significativo. Si se toma 0 entonces no se utiliza el método del gradiente conjugado. La razón por la que se ha desarrollado es la de tener un método que permita dilucidar si se alcanzado realmente el mínimo, pues tomando `CGMaxIter:=n`, se examinan todas las direcciones y, si ninguna ofrece avances, se está sin duda ante un mínimo. Esto ha sido de gran utilidad en la depuración del código.

Además, no hay que olvidar que el orden de convergencia del método de Marquardt desciende cuando está cerca del óptimo, y con estas mejoras se puede aumentar el orden de convergencia y se aprovecha al máximo la información del jacobiano, que puede conllevar un alto coste computacional cuando crece el número de variables.

En cualquier caso, este apartado se encuentra todavía en pruebas y hay que desarrollar la teoría general del método CG para aplicarla a este caso concreto, aunque en principio no se observa que sea prioritario.

Otros problemas que aparecen en algunas simulaciones son los relativos a la posible excesiva no linealidad de los parámetros ARMA, que pueden presentar una alta curvatura y torsión, lo cual ralentiza la convergencia, en especial cuando además concurre un alto número de inputs que esconden esta no linealidad. El problema se detecta por la presencia de altos valores en el autocorrelograma de los residuos. En estos casos se puede hacer una segunda estimación partiendo del ruido y utilizar los valores estimados de los inputs del primer modelo y los ARMA del segundo en una tercera estimación conjunta que suele tomar tan sólo una iteración.

4.Las interrupciones

Se ha detectado un problema que ocurre cuando existen interrupciones y variables inputs que toman valor no nulo en sus proximidades. Cuantas más interrupciones y más inputs de tipo pulso hay, más fácilmente aparecen los problemas, que se pueden detectar como altos errores del ruido diferenciado cercanos a una interrupción o más probablemente a un grupo de interrupciones próximas.

Según las pruebas de simulación realizadas, si el valor inicial tomado para las interrupciones no tiene un buen nivel de aproximación, aparecen relaciones espúreas entre interrupciones cercanas entre sí o con éstas y los inputs que toman valor en sus proximidades, y la optimización no es capaz de sortear los problemas que esto ocasiona.

El método de cálculo de las interrupciones introduce cambios irreversibles en la propia serie output, lo cual implica que no se puede volver atrás cuando en una iteración de Marquardt o de cualquier otro método de optimización no lineal, se prueba el posible avance en una dirección. Si el paso es infructuoso, al intentar retroceder y retomar los antiguos valores no se obtienen los mismos resultados porque no se aplican a la misma serie output. Se ha intentado mitigar este desastroso efecto pero una solución definitiva no es sencilla de implementar.

En estos casos, la única forma de solucionarlo hallada hasta ahora es la de estimar previamente alguna aproximación de las interrupciones, y posteriormente estimar el modelo partiendo de dichas aproximaciones.

Se ha introducido la posibilidad de especificar al estimador los valores iniciales en las interrupciones de la serie output, de forma que no hubiera que modificar la estructura `ModelDef` y sin ampliar el número de parámetros opcionales de la función `Estimate`. Para ello el usuario debe incluir una serie input que se llame obligatoriamente `InterruptionsInitialValues_` y cuya función de transferencia sea 1 y que sólo tome valores en cada una de esas interrupciones, de forma que se elimine como input y no haya ambigüedades. Esto se ha hecho para poder llevar a cabo las pruebas pertinentes sin modificar el interfaz de la función `Estimate` y con los mínimos cambios del código C++. Si se demuestra realmente que éste es el camino se puede cambiar la forma de introducir estos valores iniciales o hacer que el propio sistema los estime de algún modo.

Para obtener una buena aproximación inicial se puede estimar un modelo en el que se sustituyen las interrupciones de la serie output por 0 y se añade un pulso por cada interrupción.

Estos problemas han aparecido en los tests de simulación aleatoria de modelos ARIMA con función de transferencia e interrupciones, pero en los casos reales disponibles no se ha hecho necesario, aunque realmente hay muy pocos de estos casos reales.