Из цикла «ШАМАНСТВО В АНАЛИЗЕ ДАННЫХ»

Прогнозирование рядов соревнования «Tourism Forecasting Part Two» (http://kaggle.com/)

А.Г. Дьяконов

http://alexanderdyakonov.narod.ru/ e-mail: djakonov(собака)mail(точка)ru

Просьба сообщать обо всех найденных неточностях!!!

Рассмотрим простые методы прогнозирования временных рядов. Для примера используем ряды соревнования «Tourism Forecasting Part Two», проводившегося в рамках проекта KAGGLE осенью 2010 года. Дано 793 временных рядов из туристической области (они могут описывать число клиентов турфирмы, число свободных мест в отелях, спрос на какие-то услуги и т.д.). Первые 366 рядов – «месячные», т.е. каждое значение соответсвует значению некоторого показателя за месяц, большинство из них имеют период 12 (естественно не в строгом смысле, просто каждый следующий год «похож» на предыдущий, см. рис. ниже). Остальные 427 рядов – квартальные, большинство из них имеют период 4.

Все эксперименты будем проводить в среде MATLAB.

Нулевой шаг – загрузка данных.

Данные записаны в csv-файле (в этом текстовом файле через запятую перечислены значения). Сначала загрузим данные стандартным средством «Import Data».

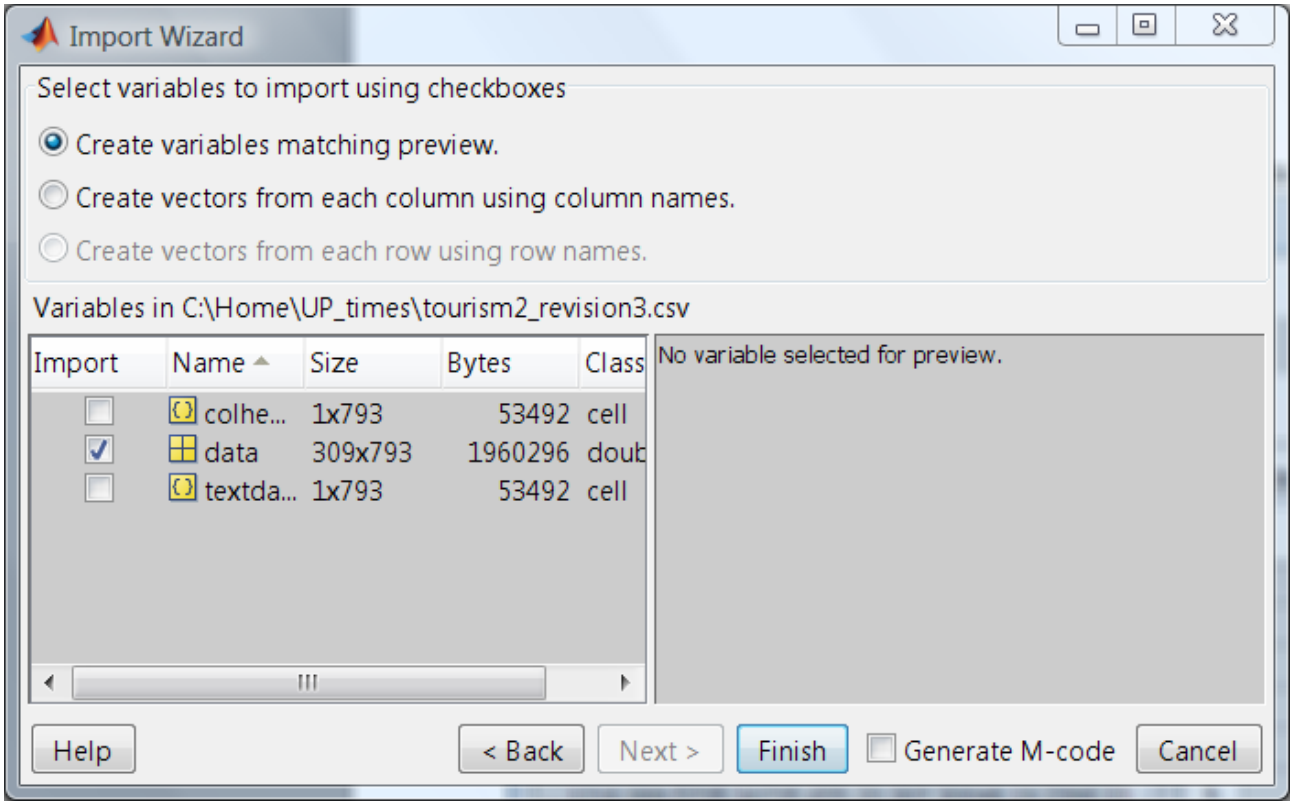

Рис. 1. Компонент Import Wizard.

В файле с рядами есть текстовая информация – названия рядов (первая строка csv-файла). Нам понадобятся только сами ряды, поэтому выбираем лишь переменную data (см. рис. 1). Ряды записаны по столбцам в матрице размера 309×793. Они разной длины, поэтому некоторые столбцы начинаются с пропусков[1]. В созданной переменной data (это матрица) пропуски «превратились» в значения NaN (см. рис. 2).

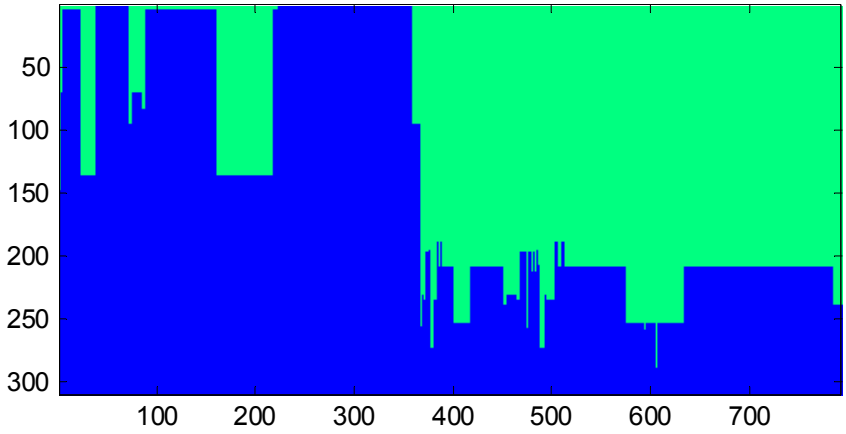

Рис. 2. Результат imagesc(isnan(data)); colormap(winter).

Первый шаг – создание программы для визуализации.

```
% нарисовать график

% x - ряд
% l - период
function X = myplot(x, l)

x(isnan(x)) = []; % убрать лишнее

if (l==0)
    plot(x);
else
    k = ceil(length(x)/l);
    x(end:k*l) = NaN; % добавить до прямоугольной матрицы
    X = reshape(x, l, k); % сделать такую матрицу
    plot(X);
end;
```

Поскольку мы будем иметь дело с периодическими рядами, то неплохо было бы иметь возможность выводить их по кускам (равным этому периоду). Ниже показана работа нашей функции…

[1] Это стандартный способ представления рядов. Иногда выравнивают по началам рядов, тогда короткие ряды заканчиваются NaNами.

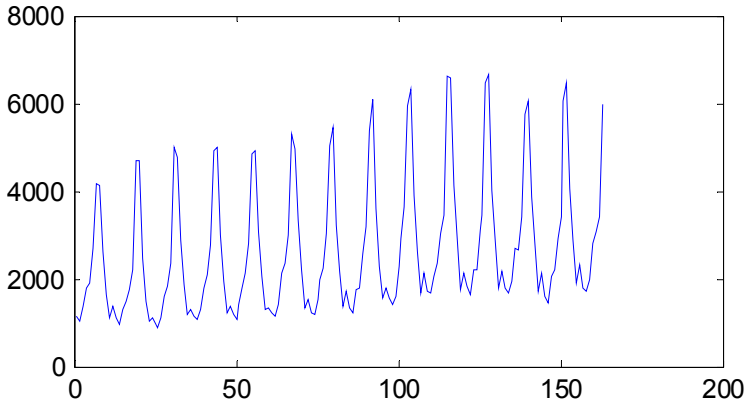

Рис. 3. Сам ряд – myplot(data(:,1), 0).

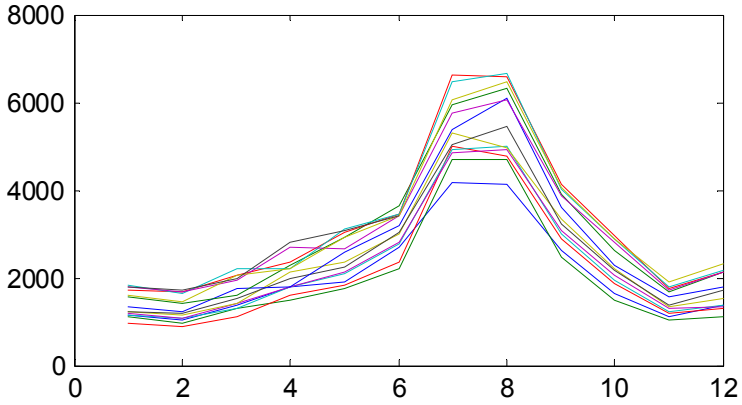

Рис. 4. Вывод по годам – X = myplot(data(:,1), 12).

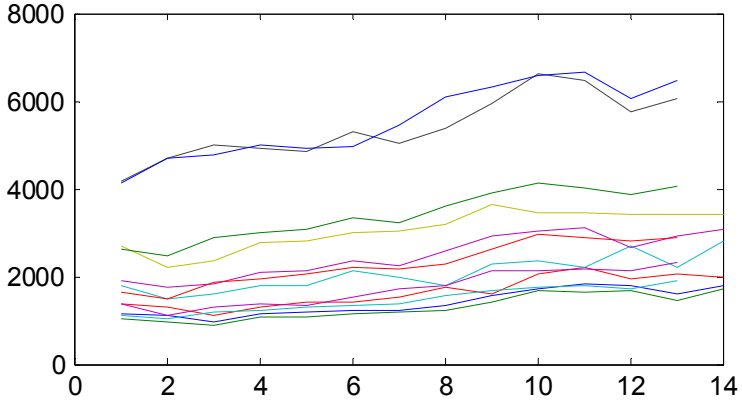

Рис. 5. Вывод эволюции годов – plot(X').

Обратите внимание, что функция возвращает матрицу X (это ряд, записанный «по столбцам»). Эту матрицу можно использовать для визуализации (см. рис. 3, 4, 5). При визуализации видно, что годы «похожи».

Второй шаг – оценка качества прогноза.

Поскольку мы будем заниматься прогнозированием, необходимо написать функцию, которая оценивает качество прогнозирования. В нашем случае это

ошибка MASE. Для ряда $\tilde{f} = (f_1, \ldots, f_n)$ периода l и прогноза $\tilde{z} = (z_1, \ldots, z_t)$ его последнего участка $(f_{n-t+1}, \ldots, f_n)$ ошибка вычисляется по формуле

$$\frac{\frac{1}{t}\sum_{i=1}^{t} |z_i - f_{n-t+i}|}{\frac{1}{n-l}\sum_{i=1}^{n-l} |f_i - f_{i+l}|}.$$

```
% ошибка MASE

% f - ряд
% z - прогноз конца
% l - период

function e = mase(f, z, l)

f = f(:); % просто вытянуть в вектор-столбец
z = z(:);

f(isnan(f)) = []; % на всякий случай

e = mean(abs(z - f((end-length(z)+1):end)))/
mean(abs(f(1:(end-l))-f((l+1):end)));
```

Давайте запустим примитивный алгоритм прогнозирования. Попробуем отрезать с конца каждого ряда два года и угадать их по оставшемуся ряду (по предыдущим годам). Угадываем следующим «наивным методом»: как было в самый последний год (известный нам), так будет и дальше (просто дублируем отрезок последнего известного нам года 2 раза). Следующий код запускает наивный метод и получает его среднюю MASE-ошибку.

```
% "Наивный" метод
E = nan([1 size(data,2)]); % ошибки прогноза
for i = 1:size(data,2)
    if i<=366
        l = 12; % период ряда
        %t = 24; % прогноз на 24 точки
    else
        l = 4; % период ряда
        %t = 8; % на 8
    end;
    f = data(:,i);
    z = f(end+1-3*l:end-2*l); % последний известный нам
год
    z = [z;z]; % дублируем его
```

```
    E(i) = mase(f, z, l);
    i % для вывода "счётчика"
end;
mean(E)
```

Результат выполнения кода:

```
    1.8525
```

Обратите внимание, что у нас две группы рядов: «месячные» – с периодом 12 и «квартальные» – с периодом 4. Вообще говоря, надо отслеживать ошибки на обеих группах:

```
[mean(E(1:366)) mean(E(366:end)) mean(E)]
ans =
    1.8077    1.8883    1.8525
```

Третий шаг – написание своего алгоритма

Теперь придумаем менее тривиальный метод. Поскольку каждый следующий год у нас «незначительно» отличается от предыдущего, то разумно предположить, что существует некоторая зависимость следующего года от текущего. Пусть это зависимость линейная, т.е. постулируем существование линейного оператора, который переводит i-й год в $(i+1)$-й. Если $\tilde{y}_i$ – l-мерный вектор-столбец (l равно 12 или 4) i-го года, то надо найти матрицу A размера $l \times l$, для которой

$$A\tilde{y}_i = \tilde{y}_{i+1},$$

т.е.

$$A[\tilde{y}_1\tilde{y}_2 \dots \tilde{y}_{k-1}] = [\tilde{y}_2\tilde{y}_3 \dots \tilde{y}_k], \qquad (1)$$

если в ряде представлена статистика за k лет. Уравнение (1) обычное матричное уравнение вида

$$AX = Y,$$

которое часто возникает в задачах линейной регрессии и имеет решение

$$A = YX^{\mathrm{T}}(XX^{\mathrm{T}})^{-1},$$

если матрица XX^{T} обратима. В противном случае используют регуляризацию (её, как мы убедимся в экспериментах, имеет смысл использовать в любом случае):

$$A = YX^{\mathrm{T}}(XX^{\mathrm{T}} + \lambda I)^{-1},$$

где матрица I имеет единицы только на главной диагонали, а остальные элементы равны нулю (регрессия в таком виде называется «гребневой»).

```
% Простейший линейный метод

% f - ряд
% l - размер окна (период ряда)
% t - сколько (точек) прогнозировать
```

```
% z - прогноз
% h - модель ряда

function [z h] = forecA(f, l, t)

% очистить от последних NaNов
f(isnan(f)) = [];

f = f(:);

% нормировка данных
minf = min(f);
f = f - minf;
maxf = max(f);
f = f./maxf;

n = length(f); % длина ряда

k = ceil(n/l); % кусков для нарезки
N = k*l;
f = [nan([N-n 1]); f]; % пополнение
F = reshape (f, l, k);
F(isnan(F(:,1)),1) = F(isnan(F(:,1)),2);
%   первый год ~ второй год

% вот такое уравнение
% A*F(:,1:(end-1)) = F(:,2:end);

F1 = F(:,1:(end-1));
F2 = F(:,2:end);

reg = 0.3; % коэф. регуляризации

A = (F2*F1')/(F1*F1'+ reg*eye(size(F1,1)));

% формирование прогноза
k2 = ceil(t/l); % кусков для прогноза
z = [];
Flast = F(:,end);
for i=1:k2
    Flast = A*Flast;
    z = [z; Flast];
end;
z = z(1:t);

z = z.*maxf;
```

```
z = z + minf;

% "модель"
h = [F(:,1) A*F(:,1:(end-1))];
h = h(:);
h = h(end-n+1:end);
h = h.*maxf;
h = h + minf;
```

Первый нетривиальный шаг здесь нормировка. Мы приводим значения ряда на отрезок [0,1], а после прогноза применяем обратное преобразование. Это нужно,

1) чтобы не получались гигантские числа при перемножении матриц (некоторые ряды имеют достаточно большие значения),
2) для «единого» формирования коэффициента регуляризации (см. дальше),
3) в дальнейшем при других способах построения матрицы A это будет удобно.

Дальше мы (уже привычно) превращаем ряд в матрицу. Однако теперь мы добавили фиктивные значения в начало (конец ряда очень ценен – его портить вообще нельзя). Нам нужна статистика ровно за k лет, но за первый год она может быть дана не полностью, поэтому эти фиктивные значения мы заменяем значениями соответствующих месяцев второго года (будем считать, что первый год очень похож на второй, раз уж нам не хватило информации обо всех его месяцах).

Дальше стандартный код линейной регрессии с регуляризацией (т.е. гребневой регрессии). Вот здесь надо, чтобы для всех рядов в получаемых матрицах были числа одного порядка. Это позволит подбирать параметр **`reg`**.

Дальше идёт сам прогноз. Мы стартуем с последней недели и применяем оператор A столько раз, сколько нужно (зависит от числа значений, которые надо спрогнозировать).

К куску кода с заголовком «модель» мы ещё вернёмся.

Следующий код запускает линейный метод прогнозирования. Как видим, результат оказался лучше наивного.

```
% Запуск линейного метода
E = nan([1 size(data,2)]); % ошибки прогноза
for i = 1:size(data,2)
    if i<=366
        l = 12; % период ряда
    else
```

```
        l = 4; % период ряда
    end;
    f = data(:,i);
    z = forecA (f(1:end-2*l), l, 2*l);
    E(i) = mase(f, z, l);
    i % для вывода "счётчика"
end;
[mean(E(1:366)) mean(E(366:end)) mean(E)]
```

Результат выполнения кода:

```
    1.6161    1.6844    1.6538
```

Четвёртый шаг – визуализация.

Неплохо бы «посмотреть, как работает наш метод». Для этого пригодится кусок кода «модель» (см. выше). По сути, это ряд, который получается, если идти по нашему ряду и пытаться по каждому году прогнозировать следующий год, т.е. это ряд прогнозов. Не совсем корректно употреблять термин «модель», поскольку эта функция зависит не только от параметров (в данном случае от матрицы A), но и от самого ряда, но мы позволили себе такую вольность. Сравнение этой модели с рядом позволит понять, как мы настроились на ряд. А сравнение прогноза с истинными значениями – качество нашего прогноза.

Следующий код выводит необходимые графики.

```
% визуализация
i = 2;
l = 12;
f = data(:,i);
f(isnan(f)) = [];
[z h] = forecA (f(1:end-2*l), l, 2*l);
figure(1);
clf;
plot(f, 'LineWidth', 1.5);
hold on;
plot([h;z],'k')
```

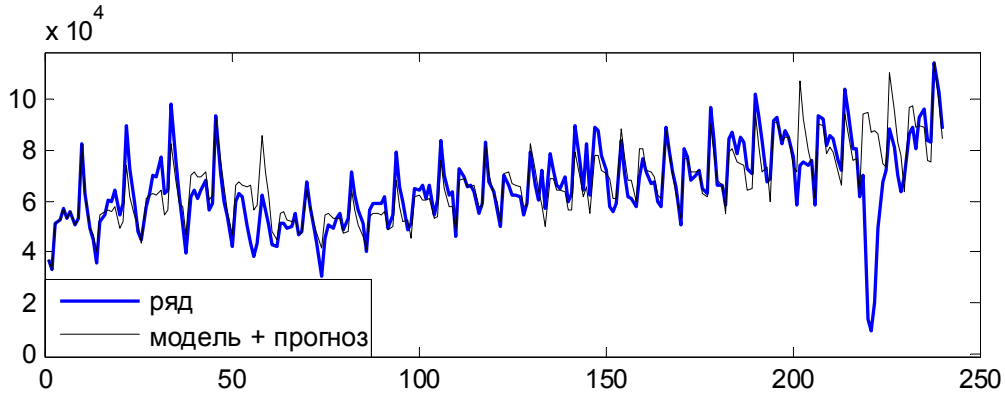

Рис. 6. Вывод ряда, модели и прогноза.

Нетрудно видеть, что «неожиданный выброс» наш метод не предсказал (что как раз вполне предсказуемо), кроме того, не везде произошла настойка на ряд (график ряда и модели не совпадают). Если уменьшить коэффициент регуляризации, то настройка будет точнее, но это скажется на качестве прогнозирования (см. рис. 7).

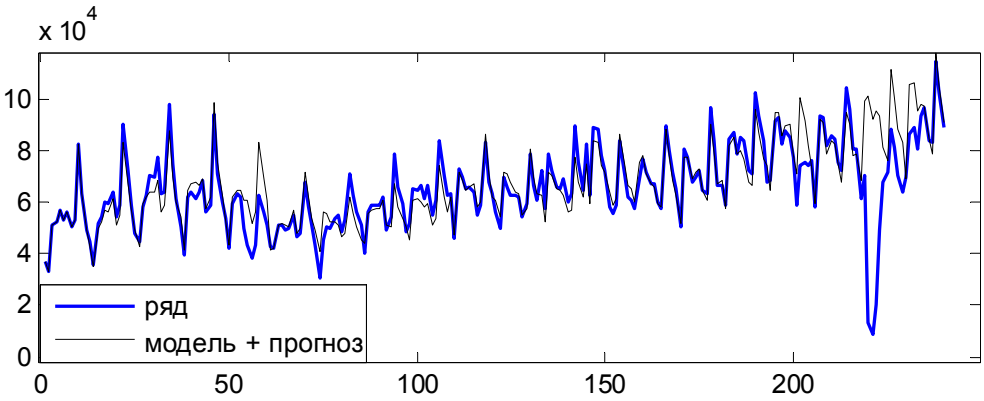

Рис. 7. Вывод ряда, модели и прогноза с коэффициентом регуляризации 0.1.

Пятый шаг – эксперименты.

Собственно говоря, это не совсем пятый шаг… это то, что теперь придётся делать постоянно. И начать можно уже сейчас. В методе есть параметр **reg**, который нуждается в настройке. Напрашивается запустить методы с разным значением этого параметра и оценить качество работы. Это делает следующий код.

```
% Настройка коэффициента регуляризации
regs = 0.1:0.02:0.6;
E = nan([length(regs) size(data,2)]); % ошибки прогноза
for j=1:length(regs)
    for i = 1:size(data,2)
        if i<=366
            l = 12;
```

```
        else
            l = 4;
        end;
        f = data(:,i);
        f(isnan(f)) = [];
        z = forecAreg (f(1:end-2*l), l, 2*l, regs(j));
        E(j, i) = mase(f, z, l);
    end;
    j % для вывода "счётчика"
end;
% визуализация
figure(2);
clf;
plot(regs, mean(E'), 'LineWidth', 1.5);
hold on;
plot(regs, mean(E(:,1:366)'), 'k');
plot(regs, mean(E(:,366:end)'), 'r');
legend('все ряды', '1я группа', '2я группа');
```

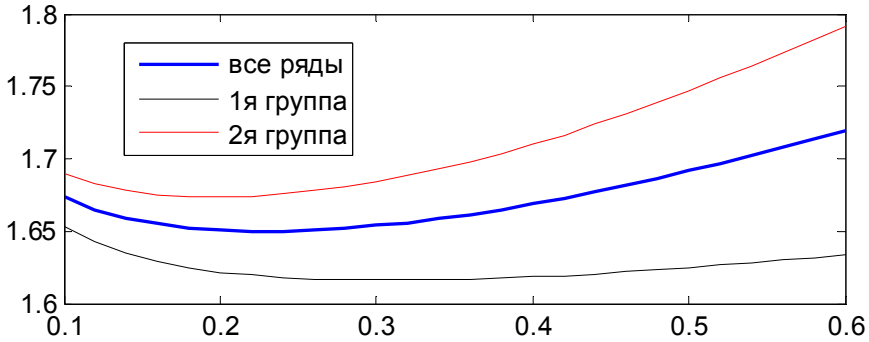

Рис. 8. Качество прогноза от коэффициента регуляризации.

Видна достаточно типичная ситуация:

1) для разных групп разные оптимальные коэффициенты регуляризации (1я группа – 0.3, вторая – 0.2),
2) качество прогноза на разных группах по-разному зависит от коэффициента регуляризации (для первой группы, например, при любом коэффициенте от 0.14 до 0.6 качество лежит в достаточно узком интервале [1.6161, 1.6347], при таких же значениях коэффициентов качество для второй группы меняется от 1.6734 до 1.7915).

Необходимо сделать важное замечание. Выбор таким образом оптимальных коэффициентов регрессии может привести к переобучению. Правильнее их подбирать при настройке модели! Но здесь есть подвох… заменим в предыдущем коде строчку вычисления **`E(j,i)`** на

```
E(j,i) = mean(abs(h - f(1:end-2*l)))./mean(f(1:end-2*l));
```

Получим оценку настройки модели (здесь конечно нужно выбрать адекватный функционал качества настройки, мы взяли «с потолка»). Получившиеся

графики качества показаны на рис. 9. Всё примитивно: чем точнее решаем уравнение настройки (меньше коэффициент регуляризации), тем точнее настраиваемся. Использовать эту информацию для оценки коэффициента регуляризации нельзя, а сравнение рисунков иллюстрирует эффект переобучения (который возникает при близких к нулю значениях коэффициента регуляризации).

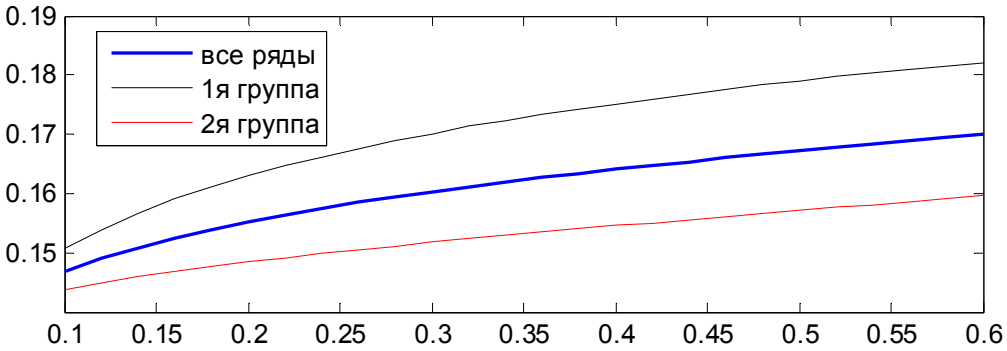

Рис. 9. Качество настройки модели от коэффициента регуляризации.

Следующий код запускает метод с «нужными» коэффициентами регуляризации (код функции **forecAreg** не приводится, поскольку это «естественное» изменение функции **forecA** с помощью добавления параметра **reg**).

```
% Линейный метод с оптимальными коэф. регрессии
E = nan([1 size(data,2)]); % ошибки прогноза
for i = 1:size(data,2)
    f = data(:,i);
    if i<=366
        l = 12; % период ряда
        z = forecAreg (f(1:end-2*l), l, 2*l, 0.3);
    else
        l = 4; % период ряда
        z = forecAreg (f(1:end-2*l), l, 2*l, 0.2);
    end;
    E(i) = mase(f, z, l);
    i % для вывода "счётчика"
end;
[mean(E(1:366)) mean(E(366:end)) mean(E)]
```

Результат:

```
    1.6161    1.6735    1.6479
```

Шестой шаг – усовершенствование алгоритма.

Этот шаг тоже проделывается многократно… Например, в нашем алгоритме мы прогнозируем следующий год, а затем по нему ещё следующий. Если прогноз

осуществляется с ошибкой (а так и происходит), то второй прогноз мы делаем, опираясь на ошибочные данные. Можно настраивать два линейных оператора: первый прогнозирует следующий год, а второй осуществляет прогноз «через год». Тогда не придётся опираться на ошибочные данные – только на последний год. Эту идею реализует следующая функция:

```
% Линейный A,B-метод (прогноз только на 2 периода)

% f - ряд
% l - размер окна (период ряда)
% t - сколько (точек) прогнозировать

% z - прогноз
% h - модель ряда

function [z h] = forecAB2(f, l)

% очистить от последних NaNов
f(isnan(f)) = [];

f = f(:);

% нормировка данных
minf = min(f);
f = f - minf;
maxf = max(f);
f = f./maxf;

n = length(f); % длина ряда

k = ceil(n/l); % кусков для нарезки
N = k*l;
f = [nan([N-n 1]); f]; % пополнение
F = reshape (f, l, k);
F(isnan(F(:,1)),1) = F(isnan(F(:,1)),2);
%   первый год ~ второй год

% вот такое уравнение
% A*F(:,1:(end-1)) = F(:,2:end);

F1 = F(:,1:(end-1));
F2 = F(:,2:end);

regA = 0.3; % коэф. регуляризации

A = (F2*F1')/(F1*F1'+ regA*eye(size(F1,1)));
```

```
% вот такое уравнение
% A*F(:,1:(end-2)) = F(:,3:end);
F1 = F(:,1:(end-2));
F2 = F(:,3:end);

regB = 0.3; % коэф. регуляризации

B = (F2*F1')/(F1*F1'+ regB*eye(size(F1,1)));

% формирование прогноза
Flast = F(:,end);
z = [A*Flast; B*Flast];

z = z.*maxf;
z = z + minf;

% "модель"
h = [F(:,1) A*F(:,1:(end-1))];
h = h(:);
h = h(end-n+1:end);
h = h.*maxf;
h = h + minf;
```

Для простоты функция возвращает прогноз только на два периода (аргумента **t**, который был раньше, теперь нет).

Удивительно, но такая функция работает хуже…

```
% Линейный метод (с прогнозом через год)
E = nan([1 size(data,2)]); % ошибки прогноза
for i = 1:size(data,2)
    if i<=366
        l = 12; % период ряда
        %t = 24; % прогноз на 24 точки
    else
        l = 4; % период ряда
        %t = 8; % на 8
    end;
    f = data(:,i);
    z = forecAB2 (f(1:end-2*l), l);
    E(i) = mase(f, z, l);
    i % для вывода "счётчика"
end;
[mean(E(1:366)) mean(E(366:end)) mean(E)]
```

Результат:

```
    1.6328    1.6825    1.6605
```

На самом деле это очень ценная идея, и она потом сработает для других алгоритмов.

Рассмотрим такое усовершенствование алгоритма. Пусть линейную регрессию вычисляет следующий блок строк

```
F1 = F(:,1:(end-1));
F2 = F(:,2:end);
reg = 0.78; % коэф. регуляризации
A = (F2*F1')/(F1*F1'+ reg*eye(size(F1,1)));
for j = 1:3
    F1 = [F(:,1:(end-1)) A*F(:,1:(end-2))];
    F2 = [F(:,2:end) F(:,3:end)];
    A = (F2*F1')/(F1*F1'+ reg*eye(size(F1,1)));
end;
```

Таким образом, мы хотим построить оператор, который не только переводит год в следующий:

$$A\tilde{y}_i = \tilde{y}_{i+1},$$

но и прогноз предыдущего года переводит в следующий:

$$A(A\tilde{y}_{i-1}) = \tilde{y}_{i+1},$$

т.е. он устойчив к своим ошибкам. Идея построения такого оператора кажется немного надуманной, но на самом деле мы чуть-чуть увеличиваем статистику с помощью такого приёма. В уравнении (1) число столбцов может быть небольшим (например, $k-1<l$ и тогда матрица XX^{T} заведомо вырождена), а мы немного искусственно увеличиваем число столбцов:

$$A[\tilde{y}_1\tilde{y}_2\ldots\tilde{y}_{k-1}A(\tilde{y}_1)A(\tilde{y}_2)\ldots A(\tilde{y}_{k-2})] = [\tilde{y}_2\tilde{y}_3\ldots\tilde{y}_k\tilde{y}_3\tilde{y}_4\ldots\tilde{y}_k].$$

Решаем это уравнение мы итерационно.

Число итераций равно трём, поскольку при этом значении наступает стабилизация качества (в идеале мы должны взять бесконечное число итераций). Самое интересное, что именно при трёх итерациях качество прогноза наилучшее (дальше оно чуть растёт[2]).

[2] Это тоже достаточно типичная ситуация!

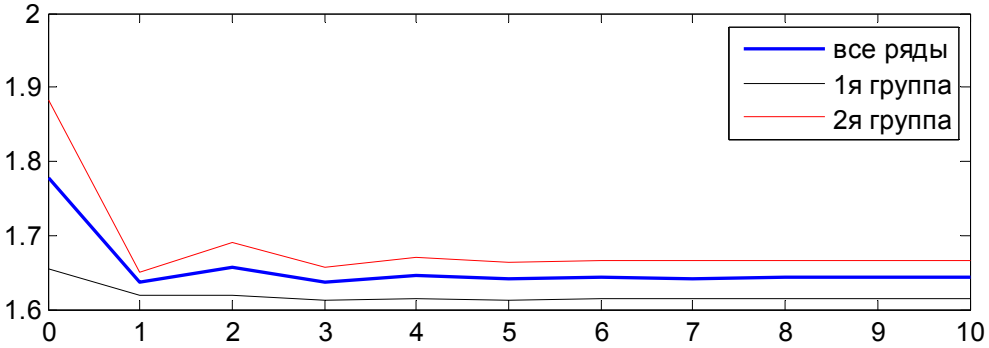

Рис. 10. Качество прогноза от числа итераций при пересчёте.

Коэффициент регрессии тут больше, чем для простейшего линейного метода, поскольку качество по-другому зависит от коэффициента регрессии (см. рис. 11). Запомните это: чуть изменяем метод – сразу меняются его оптимальные параметры (иногда значительно).

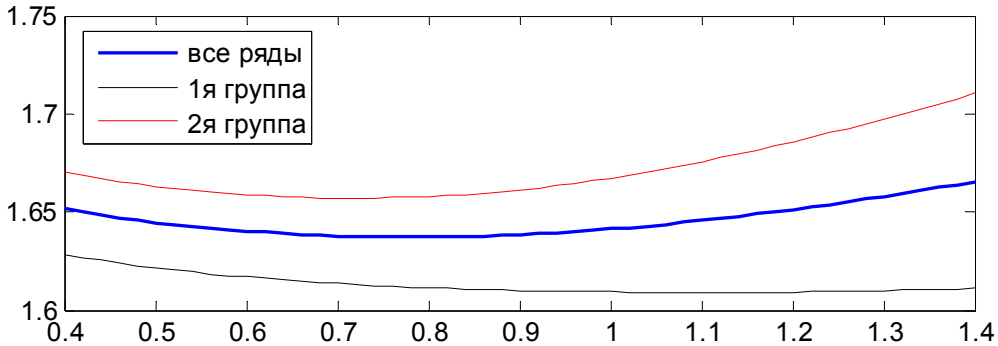

Рис. 11. Качество прогноза от коэффициента регуляризации в методе пересчёта.

Итак, качество прогноза предложенного метода следующее

```
[mean(E(1:366)) mean(E(366:end)) mean(E)]

    1.6118    1.6577    1.6373
```

Особенно заметно улучшение на второй группе рядов.

Если теперь строить два линейных оператора (для прогноза следующей недели и через неделю), то качество прогноза улучшается. Таким образом, пересчёт «возрождает» «здравую» идею применения двух операторов. Ниже представлен код, получившегося алгоритма:

```
% Линейный A,B-метод (прогноз только на 2 периода) +
ПЕРЕСЧЁТ

% f - ряд
% l - размер окна (период ряда)
% t - сколько (точек) прогнозировать
```

```
% z - прогноз
% h - модель ряда

function [z h] = forecAB2P(f, l)

% очистить от последних NaNов
f(isnan(f)) = [];

f = f(:);

% нормировка данных
minf = min(f);
f = f - minf;
maxf = max(f);
f = f./maxf;

n = length(f); % длина ряда

k = ceil(n/l); % кусков для нарезки
N = k*l;
f = [nan([N-n 1]); f]; % пополнение
F = reshape (f, l, k);
F(isnan(F(:,1)),1) = F(isnan(F(:,1)),2);
%   первый год ~ второй год

% вот такое уравнение
% A*F(:,1:(end-1)) = F(:,2:end);

F1 = F(:,1:(end-1));
F2 = F(:,2:end);
regA = 0.78; % коэф. регуляризации

A = (F2*F1')/(F1*F1'+ regA*eye(size(F1,1)));
for j = 1:3
    F1 = [F(:,1:(end-1)) A*F(:,1:(end-2))];
    F2 = [F(:,2:end) F(:,3:end)];
    A = (F2*F1')/(F1*F1'+ regA*eye(size(F1,1)));
end;

% вот такое уравнение
% A*F(:,1:(end-2)) = F(:,3:end);
F1 = F(:,1:(end-2));
F2 = F(:,3:end);
regB = 0.78; % коэф. регуляризации

B = (F2*F1')/(F1*F1'+ regB*eye(size(F1,1)));
for j = 1:3
```

```
    F1 = [F(:,1:(end-2)) B*F(:,1:(end-4))];
    F2 = [F(:,3:end) F(:,5:end)];
    B = (F2*F1')/(F1*F1'+ regB*eye(size(F1,1)));
end;

% формирование прогноза
Flast = F(:,end);
z = [A*Flast; B*Flast];

z = z.*maxf;
z = z + minf;

% "модель"
h = [F(:,1) A*F(:,1:(end-1))];
h = h(:);
h = h(end-n+1:end);
h = h.*maxf;
h = h + minf;
```

А вот его качество

```
[mean(E(1:366)) mean(E(366:end)) mean(E)]

    1.5977    1.6351    1.6186
```

Седьмой шаг – сравнение алгоритмов.

Полезно вести табличку эффективности методов. В нашем случае получается следующая таблица.

Метод	Результат		
«Наивный»	1.8077	1.8883	1.8525
Линейный **forecA**	1.6161	1.6844	1.6538
Линейный «оптимальный», **forecAreg**[3]	1.6161	1.6735	1.6479
Линейный A,B-метод, **forecAB2**	1.6328	1.6825	1.6605
Метод пересчёта	1.6118	1.6577	1.6373
A,B-пересчёт, **forecAB2P**	1.5977	1.6351	1.6186
Итоговый метод[4]	1.5475	1.5813	1.5665

Если есть возможность контроля на валидационной выборке[5], то можно контролировать эффект переобучения.

[3] Скорее всего переобучен.
[4] Его код будет приведён ниже.
[5] На соревновании «Tourism Forecasting Part Two» была возможность загрузки решений на сайт, в результате которой отображалось качество прогноза на 20% всех рядов (эта

Отметим, что последний рассмотренный алгоритм имеет достаточно много параметров (несмотря на тривиальность): два коэффициента регуляризации, два числа пересчёта (их можно сделать параметрами). При усложнении алгоритмов надо уметь эффективно перебирать параметры (и помнить о переобучении).

Восьмой шаг – идеи.

Самый главный шаг, после которого делается усовершенствование алгоритма, проводятся эксперименты и производится сравнение.

Прежде чем перечислить идеи, которые «приходят в голову» для усовершенствования линейного метода приведём код алгоритма, который был получен в результате наших небольших экспериментов. В таблице показано качество его прогноза, а после кода приведены иллюстрации его работы. Кстати, несмотря на заголовок «самый лучший», качество его работы легко улучшить (например, повышение точности на 0.1 достигается при выборе разных параметров метода для двух групп рядов).

```
% МЕТОД ПОКА САМЫЙ ЛУЧШИЙ
% f - ряд
% l - размер окна (период ряда)

% z - прогноз
% h - модель ряда

function [z h] = forecPmybest(f, l, isGraph)

f = f(:); % вытянуть в вектор-столбец
f(isnan(f)) = []; % очистить от NaNов
t = 2*l; % на сколько прогноз

f0 = f;

% ПАРАМЕТРЫ АЛГОРИТМА
regA   = 0.7; % первая регресси
acA    = sqrt(12); % первый "довесок"
regB   = 0.6; % вторая регрессия
acB    = sqrt(4); % второй "довесок"
fshift = 0.6; % сдвиг ряда на отрезок [fshift, fshift+1]

% нормировка данных
 minf = min(f);
```

валидационная группа была заранее выбрана и фиксирована). Итоговый контроль производился на оставшихся 80% рядов.

```
 f = f - minf;
 maxf = max(f);
 f = f./maxf;
 f = f + fshift;

if (isGraph)
    % отбросить последний кусочек
    endf = f(end-t+1:end);
    f(end-t+1:end) = [];
end;

n = length(f); % длина ряда

k = ceil(n/l); % кусков для нарезки
N = k*l;
f = [nan([N-n 1]); f]; % пополнение
F = reshape (f, l, k);
isnaninF = isnan(F(:,1));
F(isnaninF,1) = F(isnaninF,2);

% вот такое уравнение
% A*F(:,1:(end-1)) = F(:,2:end);

% первая регрессия A
F1 = F(:,1:(end-1));
F2 = F(:,2:end);

X = F1;
Y = F2;
A = (Y*X')/(X*X'+regA*eye(size(X,1)));
for j = 1:3
    X1 = [ones([size(X,1) 1]).*acA A*X(:,2:(end-1))
A*A*X(:,2:(end-2)) X(:,2:end)];
    Y1 = [ones([size(X,1) 1]).*acA   Y(:,3:end)
Y(:,4:end) Y(:,2:end)];
    A = (Y1*X1')/(X1*X1'+ regA*eye(size(X1,1)));
end;

% вторая регрессия B
F1 = F(:,1:(end-2));
F2 = F(:,3:end);

X = F1;
Y = F2;
B = (Y*X')/(X*X'+regB*eye(size(X,1)));
for j = 1:3
    X1 = [ones([size(X,1) 1]).*acB B*X(:,2:(end-2))
```

```
B*B*X(:,2:(end-4)) X(:,2:end)]; %A*A*A*X(:,2:(end-6))
    Y1 = [ones([size(X,1) 1]).*acB Y(:,4:end) Y(:,6:end)
Y(:,2:end)]; %Y(:,8:end)
    B = (Y1*X1')/(X1*X1'+regB*eye(size(X1,1)));
end;

% прогноз
Fend = F(:,end); % последний "год"
z = [A*Fend; B*Fend];

% "модель"
h = [F(:,1) A*F(:,1:(end-1))];
h = h(:);

% визуализация
if (isGraph)
    figure(1);
    clf;
    hold on;
    plot([f; endf],'LineWidth', 1.25); % ряд
    plot([h; z],'k'); % "модель" и прогноз
    plot(h, 'k'); % "модель"
    h2 = [F(:,1) F(:,2) B*F(:,1:(end-2))];
    h2 = h2(:);
    plot(h2, 'g'); % вторая "модель"
    legend('ряд', 'прогноз', 'модель-A', 'модель-B')

    figure(2);
    clf;
    hold on;
    plot([h; z] - [f; endf], 'LineWidth', 1.25);
    % вывод ошибки (до окончательной нормироки)
    mase([f; endf], z, 1)
    plot(h2 - f(1:length(h2(:))), 'r');
    legend('ошибка A-модели и прогноза', 'ошибка B-
модели');
end;

% разнормировка модели
h = h(end-n+1:end);
h = h - fshift;
h = h.*maxf;
h = h + minf;

% разнормировка прогноза
z = z - fshift;
```

```
z = z.*maxf;
z = z + minf;

z = round(z);
z(z<0) = 0;

 % если значения кратны
if (isequal(fix(f0./1000).*1000, f0))
    z = round(z./1000).*1000;
elseif (isequal(fix(f0./100).*100, f0))
    z = round(z./100).*100;
elseif (isequal(fix(f0./10).*10, f0))
    z = round(z./10).*10;
end;

% визуализация
if (isGraph)
    figure(3);
    clf;
    hold on;
    plot([f0(1:(end-t)); z], 'r');
    plot(f0, 'k');
    mase(f0, z, 1)
    legend('прогноз', 'ряд');
end;
```

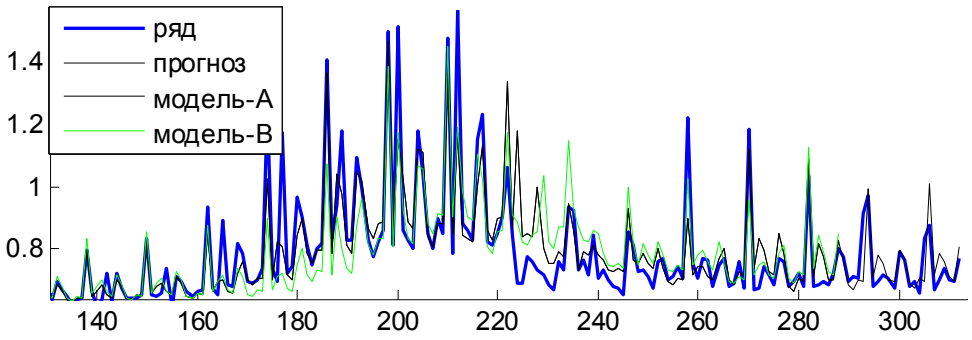

Рис. 12. figure(1) при z = forecPmybest(data(:,9), 12, true).

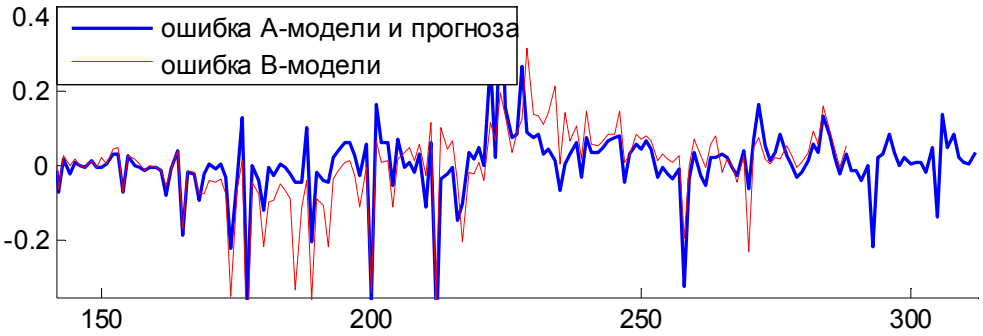

Рис. 13. figure(2) при z = forecPmybest(data(:,9), 12, true).

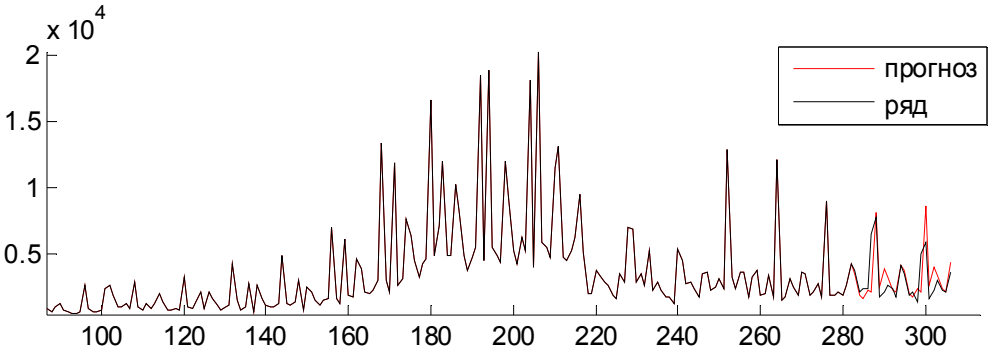

Рис. 14. figure(3) при z = forecPmybest(data(:,9), 12, true).

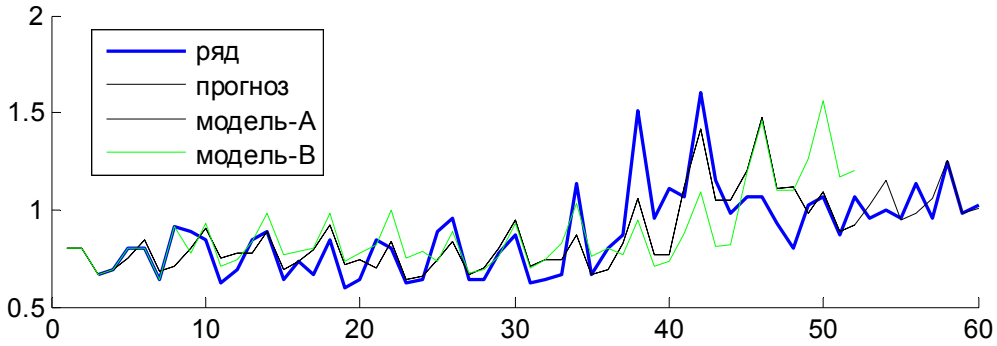

Рис. 15. figure(1) при z = forecPmybest(data(:,400), 4, true).

Отметим, что у функции появился аргумент **`isGraph`**, при его значении **`true`** функция сама обрубает кусочек ряда и пытается его прогнозировать. При этом выводятся следующие иллюстрации: ряд, прогноз, модель и модель, построенная с помощью матрицы B (рис. 12 и 15); разница между рядом и моделями (рис. 13); ряд и прогноз уже в истинном масштабе (рис. 14).

А теперь идеи…

1. Предобработка ряда

Не обязательно работать с заданными значениями ряда. Например, можно прогнозировать логарифмы значений или разности между соседними значениями. Последняя идея (в MATLABe реализуется функцией diff) применяется, чтобы сделать ряд «стационарным» (см. литературу). Иногда предварительно из ряда вычитают линейный или полиномиальный тренд (а потом добавляют к прогнозу).

2. Нормировка ряда

Это тоже связано с предобработкой. Не обязательно переводить ряд (значения ряда) именно на отрезок [0, 1], можно и на другой отрезок или стандартизовать ряд (вычесть выборочное матожидание и поделить на выборочную дисперсию).

3. Борьба с выбросами

Этот этап можно включить в предобработку: находить «неожиданные» значения ряда и заменять их «ожидаемыми», чтобы наша модель лучше подстраивалась. Можно при настройке самой модели посмотреть, на какие

значения она плохо подстраивается, и исключить их. Или в нашем методе делать прогноз не по предыдущему году, а по медиане (поэлементной) нескольких предыдущих лет. Медиана автоматически отфильтровывает выбросы. Если выброс портит последний год, то можно попробовать прогнозировать по предпоследнему через один.

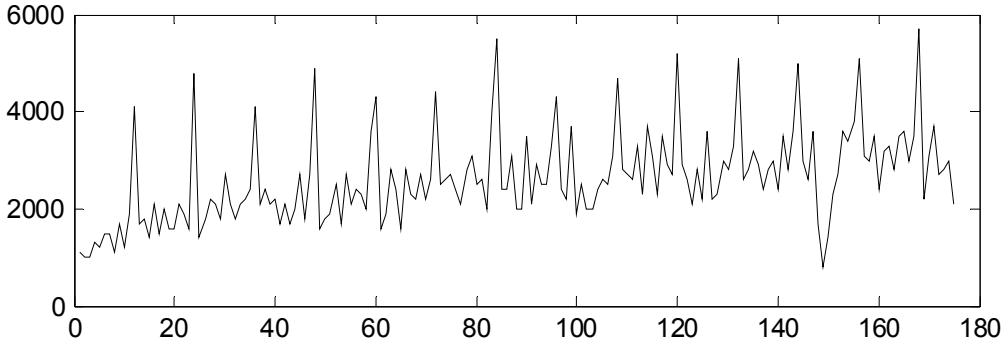

Рис. 16. Пример выброса в ряде (myplot(data(:,33),0)).

4. Учёт «режимов»

Неформально «режим» – участок со стабильным поведением ряда. Часто происходит смена таких режимов: ряд меняет поведение (часто достаточно резко). Возможно, имеет смысл настраивать модель только на режиме, который соответствует концу ряда, или не настраивать на сменах режимов (или прогнозировать смены режимов).

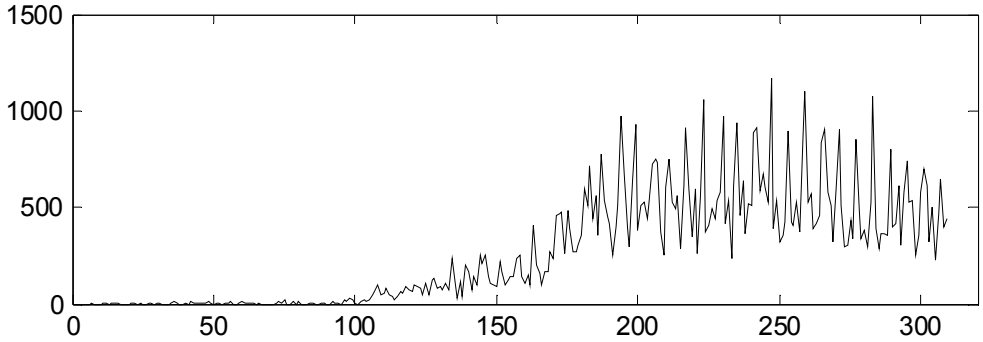

Рис. 17. Пример смены режимов (myplot(data(:,45),0)).

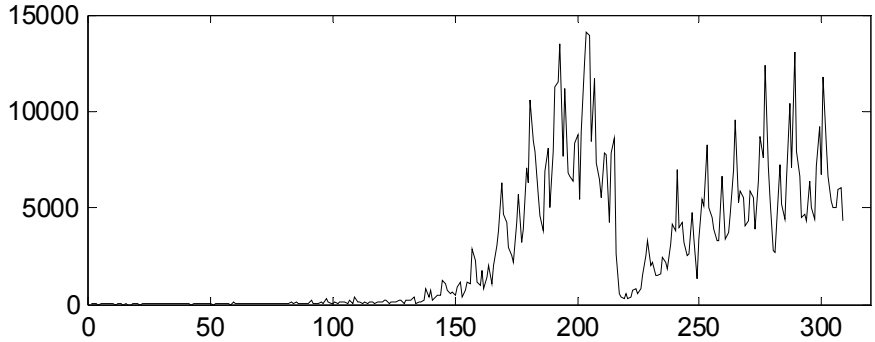

Рис. 18. Пример смены режимов (myplot(data(:,254),0)).

5. Построение методов «под ряд»

Выше мы разбили ряды на две группы «естественным образом», однако в выборке ряды могут сильно различаться. Например, одни демонстрируют

периодическое поведение, другие – хаотическое, есть ряды с выбросами и т.д. Обычно делят на группы однотипных рядов и на каждой группе настраивают метод (или на каждой группе настраивают свой метод). При настройке метода (модели) на каждый ряд в отдельности обычно происходит переобучение. Иногда строят несколько разнотипных методов, смотрят какой из них на каком ряде работает лучше.

6. Пополнение модели дополнительными параметрами

До сих пор мы искали значение, которое соответствует каждому месяцу следующего года, в виде линейной комбинации значений, соответствующих месяцем этого:

$$(\tilde{y}_{r+1})_i = \sum_{j=1}^{l} a_{ij} (\tilde{y}_r)_j ,$$

где $A = \| a_{ij} \|_{l \times l}$. В линейную комбинацию можно добавить другие регрессионные переменные, например время:

$$(\tilde{y}_{r+1})_i = \sum_{j=1}^{l} a_{ij} (\tilde{y}_r)_j + a_{i,l+1} r ,$$

т.е. решать уравнение вида

$$A\left[\begin{pmatrix} \tilde{y}_1 \\ 1 \end{pmatrix} \dots \begin{pmatrix} \tilde{y}_{k-1} \\ k-1 \end{pmatrix}\right] = [\tilde{y}_2 \dots \tilde{y}_k] .$$

7. Усложнение модели

Вместо линейной комбинации можно искать полиномиальную комбинацию (по сути, это переход к обобщённой линейной регрессии). Можно искать модель в виде отношения двух линейных комбинаций (например, поочерёдно настраивая в цикле числитель и знаменатель).

8. Другая регуляризация

Кроме другой регуляризации (см. литературу) можно вообще по-другому искать матрицы A и B. Решение вида $A = YX^{\mathrm{T}}(XX^{\mathrm{T}})^{-1}$ является решением задачи

$$\| A\tilde{y}_1 - \tilde{y}_2 \|^2 + \dots + \| A\tilde{y}_{k-1} - \tilde{y}_k \|^2 \to \min ,$$

т.е. для настройки модели мы применяем метод наименьших квадратов. Можно применять метод взвешенных квадратов (последние годы имеют больший вес):

$$w_1 \| A\tilde{y}_1 - \tilde{y}_2 \|^2 + \dots + w_{k-1} \| A\tilde{y}_{k-1} - \tilde{y}_k \|^2 \to \min$$

или метод наименьших модулей. Иногда срабатывает идея пополнения матриц квадратами (поэлементными) годов.

9. Другой метод

Естественно на фиксированных данных надо опробовать совершенно разнотипные методы. Наш можно существенно модифицировать следующим образом. Можно считать, что линейный оператор, который переводит год в следующий меняется со временем. В простейшем случае – с помощью домножения на фиксированный оператор:

$$A\tilde{y}_1 = \tilde{y}_2 ,\ AC\tilde{y}_2 = \tilde{y}_3 , \dots ,\ AC^{k-2}\tilde{y}_{k-1} = \tilde{y}_k .$$

10. Преобразование прогноза

Некоторые ряды принимают значения только из небольшого конечного множества (см. рис. 19). Значения прогноза можно «округлять» к ближайшим значениям ряда (правда, не всегда это улучшает функционал качества прогноза). См. в коде выше фрагмент «если значения кратны».

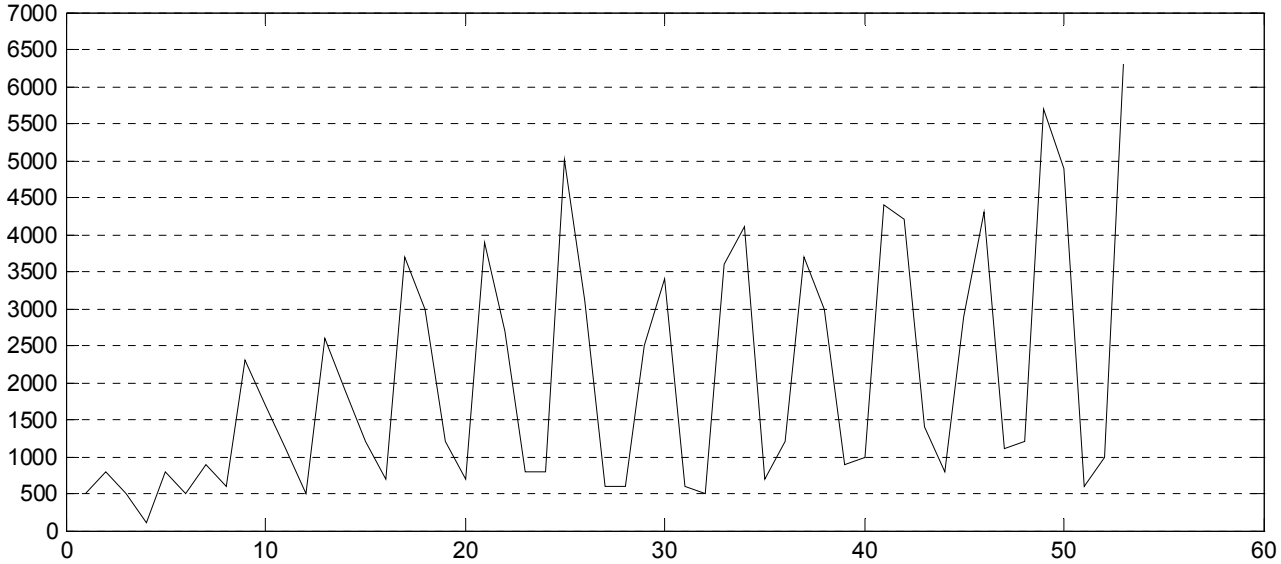

Рис. 19. Ряд с небольшим множеством значений (myplot(data(:,593),0)).

Возможно полученная модель как-то единым образом ошибается на всех рядах или на некоторых группах рядах (например занижает прогноз). Тогда имеет смысл искать преобразование (в простейшем случае опять линейное), которое «выправляет прогноз»:

$$\sum_{\tilde{y}} \left\| D_s \begin{bmatrix} A\tilde{y}_k \\ B\tilde{y}_k \end{bmatrix} - \begin{bmatrix} \tilde{y}_{k+1} \\ \tilde{y}_{k+2} \end{bmatrix} \right\|^2 \to \min,$$

здесь суммирование идёт по всем рядам s-й группы, для которых ищем преобразование D_s.