Оглавление

Теоретическое введение

Фильтры нижних частот (ФНЧ) часто используются для анализа временных рядов: для того, чтобы избавиться от шума, выделить низкочастотную составляющую из изначальных данных, или же избавиться от сезонной компоненты.

KZ-фильтр

По сравнению с другими ФНЧ, KZ-фильтр имеет только два параметра - длину окна и количество итераций. Фильтр Колмогорова-Журбенко легко справляется с проблемой недостающих данных и обеспечивает высокое частотное разрешение [7], [8].

Определение

Пусть $X(t), t = \pm 1, \pm 2, \cdots$ – вещественнозначный временной ряд. KZ-фильтр с параметрами m (длина окна) и k (количество итераций) можно определить через итерации фильтра скользящего среднего следующим образом, первая итерация MA-фильтра, применённая к m точкам временного ряда:

$$KZ_{m,k=1}[X(t)] = \sum_{s=-(m-1)/2}^{(m-1)/2} X(t+s) \times \frac{1}{m}. \tag{1}$$

Вторая итерация:

$$KZ_{m,k=2}[X(t)] = \sum_{s=-(m-1)/2}^{(m-1)/2} KZ_{m,k=1}[X(t+s)] \times \frac{1}{m}. \tag{2}$$

В общем случае k-я итерация представляет собой применение фильтра скользящего среднего к $(k-1)$-й итерации:

$$KZ_{m,k}[X(t)] = \sum_{s=-(m-1)/2}^{(m-1)/2} KZ_{m,k-1}[X(t+s)] \times \frac{1}{m}. \tag{3}$$

С другой стороны, фильтр можно определить явно, пользуясь тем, что импульсная характеристика композиции нескольких фильтров представляет собой свертку их импульсных характеристик. Пусть $h[n]$ - импульсная характеристика фильтра скользящего среднего, а m нечетное число. Из определения фильтра следует, что

$$h[n] = \begin{cases} \frac{1}{m}, n \in [-\frac{m-1}{2}, \frac{m-1}{2}], \\ 0, n \notin [-\frac{m-1}{2}, \frac{m-1}{2}]. \end{cases} \tag{4}$$

По принципу итеративного расчета имеем, что k раз выполняется свертка имеющегося сигнала функцией h. В виду ассоциативности операции свертки формулу можно переруппировать следующим образом:

$$\underbrace{(h * ... * h)}_{\text{k раз}} * X(t). \tag{5}$$

В результате, импульсная характеристика фильтра имеет следующий вид:

$$\underbrace{(h * ... * h)}_{\text{k раз}}[s] = \frac{a_s^{m,k}}{m^k}, \tag{6}$$

а сам фильтр:

$$KZ_{m,k}[X[t]] = \sum_{s=-k(m-1)/2}^{k(m-1)/2} \frac{a_s^{m,k}}{m^k} X(t+s). \tag{7}$$

Коэффициенты $a_s^{m,k}$ определяются коэффициентами многочлена $(1+z+\cdots+z^{m-1})^k$, полученного из уравнения

$$\sum_{s=-k(m-1)/2}^{k(m-1)/2} z^{s+k(m-1)/2} a_s^{m,k} = (1+z+\cdots+z^{m-1})^k. \tag{8}$$

Докажем это.

Утверждение 1. Пусть есть дискретная функция f, принимающая значения, равные 1, на отрезке $[0, m-1]$ и 0 вне этого отрезка.

Произведем k раз свертку функции f:

$$f_2[s] = (f * f)[s] = \sum_{i=0}^{m-1} f[i] f[s-i],$$

$$f_3[s] = (f * f * f)[s] = (f * f_2) = \sum_{i=0}^{m-1} f[i] f_2[s-i],$$

$$\ldots$$

$$f_k[s] = \underbrace{(f * ... * f)}_{\text{k раз}}[s] = (f * f_{k-1}) = \sum_{i=0}^{m-1} f[i] f_{k-1}[s-i].$$

Получим, что значение $f_k[s]$, соответствует коэффициенту при s-той степени в многочлене $(1 + z + ... + z^{m-1})^k$.

Доказательство.

Выполнение утверждения для $k = 1$ очевидно.

Предположим, что на k-й итерации утверждение верно и значение в т. $x = s$ равняется $b_s^{m,k}$. Докажем для $k + 1$ итерации.

В случае свертки для произвольного значения s получаем:

$$f_{k+1}[s] = (f * f_k)[s] = \sum_{i=0}^{m-1} f[i] f_k[s-i] = \sum_{i=0}^{m-1} f_k[s-i] = \sum_{i=0}^{m-1} b_{s-i}^{m,k}. \tag{9}$$

Расмотрим случай многочлена. Пусть он будет равен $p_k(z)$. Тогда $p_{k+1}(z)$ равен, по нашему определению,

$$p_k(z) * (1 + z + ... + z^{m-1}) = p_k(z) + z * p_k(z) + ... + z^{m-1} * p_k(z).$$

Из данного представления становится ясно, что коэффициент $b_s^{m,k+1}$ при степени s в полученном многочлене будет равен:

$$\sum_{i=0}^{m-1} b_{s-i}^{m,k},$$

что полностью эквивалентно значению коэффициента в формуле (9).

Утверждение 2. Пусть g - дискретная функция, равная 1 на на отрезке $[-\frac{m-1}{2}, \frac{m-1}{2}]$ и 0 в противном случае, m - нечетное натуральное число, f и f_k - функции из Утв. 1, а g_k - функция, полученная из g аналогичным для f_k способом.

Тогда

$$g_k[s] = f_k[s + \frac{k(m-1)}{2}]. \tag{10}$$

Доказательство.

Ясно, что для $k = 1$ утверждение выполняется.

Пусть утверждение верно для k, то есть $g_k[s] = f_k[s + \frac{k(m-1)}{2}]$. Для $k + 1$:

$$g_{k+1}[s] = \sum_{i=-(m-1)/2}^{(m-1)/2} g[i]g_k[s-i] = \sum_{i=-(m-1)/2}^{(m-1)/2} f[i+\frac{(m-1)}{2}]g_k[s-i] =$$

$$= \sum_{i=0}^{m-1} f[i]g_k[s-i+\frac{(m-1)}{2}] = \sum_{i=0}^{m-1} f[i]f_k[s-i+\frac{(m-1)}{2}+\frac{k(m-1)}{2}] =$$

$$= \sum_{i=0}^{m-1} f[i]f_k[s-i+\frac{(k+1)(m-1)}{2}] = \sum_{i=0}^{m-1} f[i]f_k[(s+\frac{(k+1)(m-1)}{2})-i] =$$

$$= f_{k+1}[s+\frac{(k+1)(m-1)}{2}].$$

Утверждение 3. Пусть итоговая импульсная характеристика h_k определена через h аналогичным указанному выше способом. Тогда

$$h_k[s] = \frac{g_k[s]}{m^k}. \tag{11}$$

Доказательство.

Следует из определения h, согласно которому можно сделать вывод, что

$$h[s] = \frac{g[s]}{m}.$$

В результате проделанных выкладок, получаем

$$\underbrace{(h * ... * h)}_{\text{k раз}}[s] = h_k[s] = \frac{g_k[s]}{m^k} = \frac{f_k[s+\frac{k(m-1)}{2}]}{m^k} = \frac{b^{m,k}_{s+\frac{k(m-1)}{2}}}{m^k},$$

откуда следует, что если коэффициент $a^{m,k}_s$ из (6) определить равным $b^{m,k}_{s+\frac{k(m-1)}{2}}$, то можно убедиться в корректности (8), что и требовалось доказать.

Применение и преимущества

KZ-фильтр использовался для удаления шума перед определением связи между двумя временными рядами [9] и проведением анализа по методу главных компонент [10]. Он также использовался для отделения низкочастотных компонентов, вызванных деятельностью человека, для изучения их влияния на загрязнение воздуха [11].

Фильтр Колмогорова-Журбенко применим для сглаживания периодограмм. Для класса стохастических процессов, при наихудшем раскладе, когда единственной доступной информацией о процессе является его спектральная плотность и гладкость, определяемая показателем Липшица, Журбенко была получена оптимальная ширина спектрального окна, зависящая только от гладкости спектральной плотности [1]. При сравнении с другими, часто используемыми оконными функциями (треугольным (Бартлетта), Тьюки-Хэмминга, Пуссена), Журбенко был сделан вывод о практической оптимальности окна, используемого в фильтре Колмогорова-Журбенко. Графики сравнения и соответствующие вычисления представлены в [1].

Фильтр хорошо работает в условиях недостающих данных, особенно в многомерных временных рядах, где эта проблема возникает из-за пространственной разреженности. Фильтр является устойчивым к выбросам [3].

Однако KZ-фильтр имеет тенденцию сглаживать резкие разрывы (включая всплески и впадины), что может привести к затруднению оценки свойств временного ряда.

KZA-фильтр

Адаптивная версия KZ-фильтра, или же KZA-фильтр, была разработана для поиска разрывов в непараметрических, сильно зашумленных сигналах. KZA-фильтр сначала определяет потенциальные временные интервалы, в которых происходит разрыв. Затем проводится более тщательное исследование этих временных интервалов и уменьшение размера окна таким образом, что качество сглаженного результата улучшается [3].

Определение

Пусть $X(t), t = \pm 1, \pm 2, \cdots -$ вещественнозначный временной ряд. Перепишем определение для фильтра скользящего среднего через параметр q, где q - половина длины окна скользящего среднего:

$$Y(t) = \frac{1}{2q+1} \sum_{s=-q}^{q} X(t+s). \tag{12}$$

Аналогично перепишем определение KZ-фильтра как $KZ(q, k)$, q - полудлина окна, k - количество итераций MA-фильтра. Первым шагом адаптивного фильтра Колмогорова-Журбенко мы применяем обычный KZ-фильтр к исходному временному ряду X(t):

$$Z(t) = KZ_{q,k}[X(t)]. \tag{13}$$

Далее определим абсолютное изменение $D(t)$ внутри окна как:

$$D(t) = |Z(t+q) - Z(t-q)|, \tag{14}$$

а скорость его изменения как

$$D'(t) = D(t+1) - D(t). \tag{15}$$

Основная идея KZA-фильтра - переопределить границы окна скользящего среднего по следующему принципу. Когда точка данных t находится в области возрастания $D(t)$, то есть, $D'(t) > 0$, полудлина q_H окна MA-фильтра перед ней (начало окна MA-фильтра) уменьшается пропорционально $D(t)$, а полудлина q_T окна после нее (конец окна MA-фильтра) остается неизменной. В области убывания $D(t)$ ($D'(t) < 0$) будет просиходить обратное: q_T уменьшается, а q_H останется неизменным.

То есть, адаптивный фильтр можно определить следующим образом, первая итерация:

$$Y^{(1)} = \frac{1}{q_T(t) + q_H(t) + 1} \sum_{s=-q_T(t)}^{q_H(t)} X(t+s), \tag{16}$$

где

$$q_H(t) = \begin{cases} q, D'(t) > 0, \\ f(D(t))q, D'(t) \leq 0, \end{cases} \tag{17}$$

$$q_T(t) = \begin{cases} q, D'(t) < 0, \\ f(D(t))q, D'(t) \geq 0, \end{cases} \tag{18}$$

а $f(D(t))$ определяется следующим образом:

$$f(D(t)) = 1 - \frac{D(t)}{\max D(t)}. \tag{19}$$

k-я итерация:

$$Y^{(k)} = \frac{1}{q_T(t) + q_H(t) + 1} \sum_{s=-q_T(t)}^{q_H(t)} Y^{(k-1)}(t+s). \tag{20}$$

Применение и преимущества

Исследования показали, что KZA-фильтр хорошо работает как с одномерными, так и с многомерными данными.[12], [13]. Фильтр KZA можно использовать и для обнаружения разрывов в высокоразмерных данных, например, для выявления резких изменений цвета в изображениях [5]. Для трехмерных данных (например трехмерные пространственные данные или трехмерные медицинские изображения) KZA также может быть применен для обнаружения разрывов (границ между слоями) [6].

Префиксные суммы

Определние.

Префиксными суммами массива $[a_0, a_1, .., a_{n-1}]$ называется массив $[0, s_1, .., s_n]$, где $s_0 = 0, s_j = \sum_{i=0}^{j-1} a_i = s_{j-1} + a_{j-1}, j = 1, ...n.$

Префиксные суммы обычно используются для быстрого нахождения суммы на отрезке $[l, r]$ массива a, значения которого остаются постоянными. Для этого берется разность $(s_r - s_{l-1})$ двух его элементов.

Использование префиксных сумм требует линейных затрат как по памяти, так и по времени на их расчет. При применении такой оптимизации к расчетам в фильтре Колмогорова-Журбенко, сложность по времени работы составляет $O(n)$ (n - размер входных данных), в отличие от наивного решения, где она равна $O(n \cdot m)$ (m - размер окна). Однако такая оптимизация может привести к снижению точности вычислений, что обусловлено особенностями работы чисел с плавающей точкой.

Оптимизации и распараллеливание

Оптимизация с помощью префиксных сумм

Рассмотрим применение префиксных сумм для подсчета значения фильтра в фиксированном окне.

Пусть значения расположены в массиве data. Построим массив префиксных сумм $pref\_sum$, как было объяснено выше, с той лишь разницей, что будем игнорировать элементы, которые бесконечны или NaN, по правилам

- $pref\_sum[0] = 0$,
- $pref\_sum[i] = pref\_sum[i-1] + data[i-1]$.

А также построим массив префиксного количества элементов $pref\_finite\_cnt$, для того чтобы знать количество элементов, не равных inf и NaN:

$$pref\_finite\_cnt[i] = pref\_finite\_cnt[i-1] + 1 \cdot is\_finite\_flag,$$

для $i = 1, 2, ..$ ($is\_finite\_flag = 0$, если $data[i-1] = inf$ или $data[i-1] = NaN$, и 1 - иначе).

В приведеннном ниже коде представлен процесс построения описанных выше массивов

```
static void calc_prefix_sum(const double *data, int size, double
    *pref_sum, int *pref_finite_cnt)
{
    int i;

    pref_sum[0] = 0;
    pref_finite_cnt[0] = 0;
    for (i = 1; i <= size; i++) {
        pref_sum[i] = pref_sum[i-1];
        pref_finite_cnt[i] = pref_finite_cnt[i-1];
        if (isfinite(data[i-1])) {
            pref_sum[i] += data[i-1];
            ++pref_finite_cnt[i];
        }
    }
```

```
}
```

Листинг 1. Построение префиксных сумм и количеств конечных элементов

Предположим, мы зафиксировали длину окна w слева и справа от центра $window\_center$ и z - количество элементов в окне. Тогда мы хотим посчитать

$$(data[window\_center - w + 1] + data[window\_cneter - w + 2] + ...$$
$$+data[window\_center + w - 1] + data[window\_center + w])/z.$$

Заметим, что числитель выражения может быть представлен через предподсчитанные нами префиксные суммы как

$$\frac{pref\_sum[window\_center + w + 1] - pref\_sum[window\_center - w]}{pref\_cnt[window\_center + w + 1] - pref\_cnt[window\_center - w]}.$$

Таким образом, для того, чтобы подсчитать значение фильтра в фиксированном окне, нам достаточно предподсчитать оба массива, а затем за $O(1)$ мы можем отвечать на данный запрос. В приведенном ниже коде представлена функция, возвращающая значение фильтра в окне:

```
static double mavg1d(const double *pref_sum, const int *pref_finite_cnt, int length, int col, int w)
{
    double s;
    int z;
    int start_idx, end_idx;

    /* window length is 2*w+1 */
    start_idx = (window_center+1) - w; /* the first window value index */
    if (start_idx < 0)
        start_idx = 0;

    end_idx = (window_center+1) + w; /* the last window value index */
    if (end_idx > data_size)
        end_idx = data_size;

    /* (window sum) = (sum containig window) - (sum before window) */
    s = (pref_sum[end_idx] - pref_sum[start_idx-1]);
    z = (pref_finite_cnt[end_idx] - pref_finite_cnt[start_idx-1]);
```

```
    /*
    if (z == 0)
        return nan("");
    */
    return s/z;
}
```

Листинг 2. Расчет значения фильтра в окне с помощью префиксных сумм и количеств конечных элементов

Распараллеливание программного кода

Заметим, что

1. Результат применения MA-фильтра на текущей итерации зависит только от значений на предыдущей.

2. Значения, полученные на предыдущей итерации, остаются неизменными во время расчета.

Следовательно, расчет значений может происходить независимо, а значит параллельно. Однако, использование очень большого количества потоков бессмысленно, поскольку

1. Создание потока является дорогостоящей операцией, которая может занять не меньше времени, чем сами расчеты.

2. Количество потоков, которые могут выполняться одновременно, ограничено числом ядер процессора.

Таким образом, можно установить число потоков равным количеству ядер процессора, n_{ker}. Поскольку число значений, которые потребуется рассчитать, на практике сильно больше n_{ker}, то необходимо распределить их вычисление таким образом, чтобы загруженность потоков оказалась равномерной. Наиболее простой, и при этом эффективный способ распареллеливания - разделить массив данных на n_{ker} равных непересекающихся отрезков, и каждому потоку сопоставить один из них.

Далее можно заметить, что на каждой итерации выполняются одни и те же действия, а также что размер данных, как и объем вычислений, остается прежним. То есть, можно разделить между потоками выполнение сразу всех итераций, а не только одной, в результате чего будут сэкономлены ресурсы, которые были бы направлены на закрытие и создание потоков между рассчетами каждой из них. Подобная оптимизация требует использование барьера (объекта синхронизации), чтобы расчет следующей итерации ни в одном потоке не начинался раньше окончания расчета текущей.

Проиллюстрируем работу одного потока кодом из программы. Здесь `perform_single_iteration` осуществляет расчет выходного значения для отрезка [`start_idx`, `end_idx`], а `sync_iteration` является барьером, который после завершения ожидания всех потоков дополнительно осуществляет подготовку данных к выполнению следующей итерации:

```
for (SizeT i = 0; i < iterations - 1; ++i) {
    this->perform_single_iteration(start_idx, end_idx);
    // waiting for other threads to complete iteration
    sync_iteration.arrive_and_wait();
}
// perform last iteration and finish
this->perform_single_iteration(start_idx, end_idx);
```

Список литературы

1. Zurbenko, I. (1986) The Spectral Analysis of Time Series, North-Holland Series in Statistics and Probability. Elsevier, Amsterdam.

2. Yang, W., Zurbenko, I. (2010). Nonstationarity. In WIREs Computational Statistics (Vol. 2, Issue 1, pp. 107–115). Wiley. https://doi.org/10.1002/wics.64

3. Zurbenko, I., Porter, P. S., Rao, S. T., Ku, J. Y., Gui, R., Eskridge, R. E. (1996). Detecting Discontinuities in Time Series of Upper-Air Data: Development and Demonstration of an Adaptive Filter Technique. Journal of Climate, 9(12), 3548–3560. http://www.jstor.org/stable/26201469

4. G Zurbenko, I., Sun, M. (2017). Applying Kolmogorov-Zurbenko Adaptive R-Software. In International Journal of Statistics and Probability (Vol. 6, Issue 5, p. 110). Canadian Center of Science and Education. https://doi.org/10.5539/ijsp.v6n5p110

5. Yang, W., Zurbenko, I. (2010). Kolmogorov–Zurbenko filters. In WIREs Computational Statistics (Vol. 2, Issue 3, pp. 340–351). Wiley. https://doi.org/10.1002/wics.71

6. Zurbenko, I. G., Smith, D. (2017). Kolmogorov–Zurbenko filters in spatiotemporal analysis. In WIREs Computational Statistics (Vol. 10, Issue 1). Wiley. https://doi.org/10.1002/wics.1419

7. Eskridge, R. E., Ku, J. Y., Rao, S. T., Porter, P. S., Zurbenko, I. G. (1997). Separating Different Scales of Motion in Time Series of Meteorological Variables. In Bulletin of the American Meteorological Society (Vol. 78, Issue 7, pp. 1473–1483). American Meteorological Society. https://doi.org/10.1175/1520-0477(1997)078<1473:sdsomi>2.0.co;2

8. Yang, W., Zurbenko, I. (2010). Kolmogorov–Zurbenko filters. In WIREs Computational Statistics (Vol. 2, Issue 3, pp. 340–351). Wiley. https://doi.org/10.1002/wics.71

9. Zurbenko, I. G. AND M. Sowizral. RESOLUTION OF THE DESTRUCTIVE EFFECT OF NOISE ON LINEAR REGRESSION OF TWO TIME SERIES. (R825260). BULLETIN OF THE AMERICAN METEOROLOGICAL SOCIETY. Allen Press, Inc., Lawrence, KS, 3:1-17, (1999).

10. Tsakiri, Katerina. (2010). Effect of noise in principal component analysis with an application to ozone pollution.

11. Milanchus, M. L., Rao, S. T., Zurbenko, I. G. (1998). Evaluating the Effectiveness of Ozone Management Efforts in the Presence of Meteorological Variability. In Journal of the Air amp; Waste Management Association (Vol. 48, Issue 3, pp. 201–215). https:

//doi.org/10.1080/10473289.1998.10463673

12. Civerolo, K. L., Brankov, E., Rao, S. T., Zurbenko, I. G. (2001). Assessing the impact of the acid deposition control program. In Atmospheric Environment (Vol. 35, Issue 24, pp. 4135–4148). Elsevier BV. https://doi.org/10.1016/s1352-2310(01)00200-x

13. Close B, Zurbenko I. Time series analysis by KZA of the fatal analysis reporting system. In: Proceedings of the Joint Statistical Meeting; Denver, Colorado; 2008.

Исходники

Ссылка на исходники кода: https://github.com/jakosv/kza

Ссылка на исходники отчета: https://www.overleaf.com/read/bzptxqgznptr#6ac86c