Федеральное государственное автономное образовательное
учреждение высшего образования
«Национальный исследовательский университет
Высшая Школа Экономики»

Факультет компьютерных наук

Курсовая работа

Прогнозирование хаотических временных рядов: алгоритм Markov Chain для прогнозирования с помощью цепей Маркова

Приготовлен студентом:
Группа мНОД21_ИССА
Цыренов Баяр Солбонович

Руководитель КР
Д.ф.-м.н., профессор:
Громов В.А.

Москва
2022

Содержание

1. Введение

Хаотические временные ряды встречаются во многих областях деятельности человека. Такие, как медицина, физика, финансы и других областях. Хаотические ряды являются таковыми в связи их неустойчивости по Ляпунову. Одним из характеристик таких рядов является старший показатель Ляпунова. Это означает высокую чувствительность к начальным данным. Малое отклонение приводит к экспоненциальному росту ошибки с течением времени. Еще одним показателем таких систем является горизонт предсказуемости T. Этот показатель показывает среднее время предсказуемости системы. Обыно его считают с помощью старшего показателя Ляпунова: $T \sim \frac{1}{\lambda_0}$, где λ_0 - старший показатель Ляпунова. В вопросе прогнозирования хаотических временных рядов возникает еще один показатель - это горизонт прогнозирования h, и обычно его определяют сильно меньшим числом $T >> h$. Горизонт прогнозирования - это количество шагов возможного прогнозирования хаотических рядов, где ошибка не превышает заранее заданную планку ошибки, который линейно зависит от горизонта предсказуемости. Но существуют алгоритмы, позволяющие прогнозировать за горизонтом прогнозирования, такие как метод кластеризации.

Задача данной работы - предложить новый алгоритм прогнозирования за горизонтом прогнозирования и сравнить предложенный способ с алгоритмом кластеризации о объяснить основные отличия и сильные стороны того или иного алгоритма.

2. Существующие методы прогнозирования рядов

На данный момент существуют множество различных алгоритмов прогнозирования хаотических временных рядов. Некоторые из них посвящены теме прогнозирования на один или несколько шагов. Такие как метод опорных векторов [1] или бустинг.

В работе [2] рассматриваются и сравниваются 5 различных алгоритмов прогнозирования на множество шагов вперед. Такие как итеративная стратегия, прямая стратегия, стратегии DirRec, MIMO, DIRMO.

Но в этой работе мы сконцентрируемся на алгоритме прогнозирования с помощью кластеров [3]. Задача прогнозирования сформулировано в виде задачи оптимизации, где и будут сформулированы основные понятия и определения.

Пусть $Y = \{y_0, y_1, \dots, y_t, \dots\}$ - хаотический временной ряд, динамический процесс завершен и данные нормализованы. Этот ряд разделен на две непересекающиеся части: обучающую и тестовую $Y = Y_1 \cap Y_2, Y_1 \cup Y_2 = \emptyset$. Алгоритм вычисляет возможные прогнозное значения $S_{t+h} = \{y_{t+h}^{(1)}, \dots, y_{t+h}^{(N_{t+h})}\}$, t - последний член ряда Y_2, h - горизонт прогнозирования, $y_{t+h}^{(i)}$ - i-ое возможное значение, N_{t+h} - количество возможных значении. Определим оператор прогнозирования:

$$\zeta(S_{t+h}) = \begin{cases} 1, & \text{если точка прогнозируемая} \\ 0, & \text{иначе} \end{cases}$$

Оператор g вычисляет объединенное значение из множества возможных значении: $g(S_{t+h}) = \hat{y}_{t+h}$. Тогда задача оптимизации будет выглядеть:

1. $\min \sum_{t+h \in Y_2} (1 - \zeta(S_{t+h}))$ - минимизация количества непрогнозируемых точек,

2. $\min \frac{1}{|Y_2|} \sum_{t+h \in Y_2} \zeta(S_{t+h}) ||g(S_{t+h}) - y_{t+h}||$ - минимизация ошибки.

Алгоритм кластеризации включает в себя несколько шагов:

1. Формирование обучающей выборки:
   z-вектор - это последовательность наблюдении, взятый в соответствии некоторому паттерну. Паттерн - это вектор расстоянии между наблюдениями. Для примера возьмем z-вектор (y_1, y_3, y_6) или (y_2, y_4, y_7), соответствующий паттерну (2, 2, 3).
   Пусть $\aleph(L, k_{max})$ - множество всех возможных паттернов длины L и максимальным расстоянием между наблюдениями k_{max}. β - процент паттернов, которые будут использованы. Для каждого паттерна $\alpha \in \beta\aleph(L, k_{max})$ вычисляется множество z-векторов и мотивы (центры кластеров). Для кластеризации используются алгоритмы DBScan [4] и

Wishart [5].

2. Вычисление вектора возможных прогнозных значении:
   Пусть Ξ_α - множество мотивов для паттерна α. Для прогнозируемой точки $t+h$ для каждого паттерна $\alpha \in \beta\aleph(L, k_{max})$ для каждого мотива $C_\alpha \in \Xi_\alpha$ вычисляется вектор C возможных значении. Если евклидово расстояние между вектором C и мотивом $C_\alpha = (\eta_1, \ldots, \eta_{L-1})$ меньше некоторого значения ε, то значение $(C_\alpha - \eta_L)$ включается в вектор C.

3. Определение непрогнозируемых точек:
   Здесь приведены те алгоритмы, что были использованы в работе:

   (a) Принудительное прогнозирование (ForcedPredictionNPM):
   Алгоритм объявляет все точки прогнозируемыми, если вектор прогнозируемых значении непусто.

   (b) Быстрый рост разброса (RapidGrowthNPM):
   Алгоритм основан на предположении, что если разброс возрастет монотонно на трех или более точках подряд, то точки после третьей включительно являются непрогнозируемыми.

   (c) Быстрый рост числа кластеров (RapidGrowthDBSCAN):
   Аналогичен предыдущему, только вместо роста разброса считается количество кластеров, найденные алгоритмом DBScan.

4. Вычисление объединеного значения:
   Для вычисления единого прогнозного значения использовались следующие алгоритмы:

   (a) Среднее значение ('a' - average):
   Вычислялось среднее значение вектора C.

   (b) Центр наибольшего кластера DBScan, Wishart, OPTICS ('db', 'wi', 'op'): Единое значение вычислялось путем усреднения значении кластера с наибольшим числом точек.

3. Алгоритм Markov Chain

В алгоритме Markov Chain, многие шаги совпадают с таковыми в методе кластеризации. Цепь Маркова - это последовательность случайных событии с конечным числом исходов. На данный момент, событиями называются значения, полученные из множества обучающей выборки. В этой работе использовалось событие Difference - это разница между парой последующей друг за другом точек. Пусть $s_1^L = (y_1, \ldots, y_{1+L}) \in Y_1$ - последовательность точек из обучающей выборки длины L. Тогда имеем последовательность событии $\theta_1^{L-1} = (\theta_1^1, \ldots, \theta_L^1)$, где $\theta_i^1 = y_{i+1} - y_i$.

Теперь необходимо построить ориентированный граф событии $G = (V, E)$. Изначально граф содержит только стартовую вершину $e_0 \in E$. Далее имеем некоторое событие θ_1^k и для него необходимо среди доступных вершин найти соответствующее ему вершину $e_1^{\theta_1^k}$ и увеличить вес ребра $v(e_0) = e_1^{\theta_1^k}$ на единицу. Если такового нету, то соответствующую вершину необходимо создать. Таким образом для последовательности точек s_k^L и событии θ_k^{L-1} имеем путь графа $e(s_k^l) = (e^{\theta_1^k}, \ldots, e^{\theta_L^k})$.

Предлагается несколько определении доступной вершины:

1. Union: Вершина $e_i^{\theta_i^k}$ для вершины $e_{i-1}^{\theta_{i-1}^k}$ называется доступной, если длина пути до этих вершин совпадает.

2. Individual: Вершина $e_i^{\theta_i^k}$ для вершины $e_{i-1}^{\theta_{i-1}^k}$ называется доступной, если существует грань $v(e_{i-1}^{\theta_{i-1}^k}) = e_i^{\theta_i^k}$.

Другими словами, метод Union ищет среди всех вершин в уровне i, а Individual ищет только среди соседей вершины. В дальнейшем будет использоваться метод Individual для всех экспериментов.

Таким образом при объявленной длине последовательности L и обучающей выборкой Y_1 мощности $|Y_1|$ мы определяем $|Y_1| - L$ последовательностей и генерируем граф событии.

Следующим шагом будет формирование прогнозных значении. Для этого берется некоторая последовательность точек из тестовой выборки $(y_k, \ldots, y_{k+l})$ длины $l < L$. Далее для этой последовательности вычисляется последовательность событии $(\theta_1^k, \ldots, \theta_{l-1}^k)$. Далее в графе среди доступных вершин для начальной вершины e_0 вычисляется вершина с наименьшей

ошибкой $\min_{\theta_1^i \in E_1}(\theta_1^k - \theta_1^i)^2$. И так далее для каждого события θ^k. Итого мы придем к вершине глубины l в то время, когда граф имеет максимальную глубину $L-1$. Таким образом мы можем продолжить последовательность $(y_k, \ldots, y_{k+l})$ длины l до последовательности $(y_k, \ldots, y_{k+l}, \check{y}_{k+l+1}, \ldots, \check{y}_{k+L})$, где $\check{y}_{k+l+1} = y_{k+l} - \theta_{k+l}^i$, где θ_{k+l}^i соответствует вершине, который был выбран случайно с учетом весов граней (чем больше вес, тем больше вероятность). И так далее до конца графа.

Таким образом начиная с двух точек тестового множества Y_2 мы прогнозируем последовательность $(y_1, y_2, \check{y}_3, \ldots, \check{y}_{L-1})$. Далее увеличиваем длину последовательности из тестового множества на единицу получаем предсказанную последовательность $(y_1, \ldots, y_3, \check{y}_4, \ldots, \check{y}_{L-1})$. В итоге мы получим последовательность прогнозных векторов $C = (C_1, \ldots, C_{L-1})$, $|C_i| = L - i$. Таким образом заранее задав такую глубину графа $L > h$, то тогда появляется возможность прогнозировать за горизонтом прогнозирования.

Далее ранее оговоренными алгоритмами определяем непрогнозируемые точки и вычисляем объединенные значения точек. Таким образом мы имеем 2 шага в прогнозировании:

1. Создание графа (Produce Graph):

Algorithm 1: ProduceGraph

Input: Train set Y_1, length L

Output: Graph G

while $i < (|Y_1| - L)$ **do**

    current_node = e_0;

    while $j <$ *L - 1* **do**

        $\theta_j = y_{i+j} - y_{i+j+1}$;

        if θ_j *in possible nodes* **then**

            increase a weight by 1;

            current_node = possible_node

        else

            create a new node with θ_j;

            current_node = new_node

        end

    end

end

2. Прогноз (Predict):

Algorithm 2: Predict

Input: Test set Y_2

Output: Predicted vector $\hat{y}$

while $1 < i < L - \max(L - |Y_2|, 0)$ **do**

  true_array $= (y_1, \ldots, y_i)$;

  continue true_array in graph and get current_node;

  from current_node continue array and get array $(y_1, \ldots, y_i, \check{y}_{i+1}, \ldots, \check{y}_L)$;

  get predicted array C_i;

end

predicted vector $C = (C_1, \ldots, C_L)$;

for $i < L$ **do**

  if C_i *is predictable* **then**

    $\hat{y}_i = g(C_i)$;

  end

end

4. Результаты экспериментов

Для проверки работы алгоритма был использован хаотический ряд Лоренца с параметрами $\sigma = 10, r = 28, b = \frac{8}{3}$:

$$\begin{cases} \dot{x} = \sigma(y - x) \\ \dot{y} = x(r - z) - y \\ \dot{z} = xy - bz \end{cases}$$

Для численного решения был применен алгоритм Рунге-Кутты 4 порядка. В качестве одномерного ряды были выбраны значения ряда x.

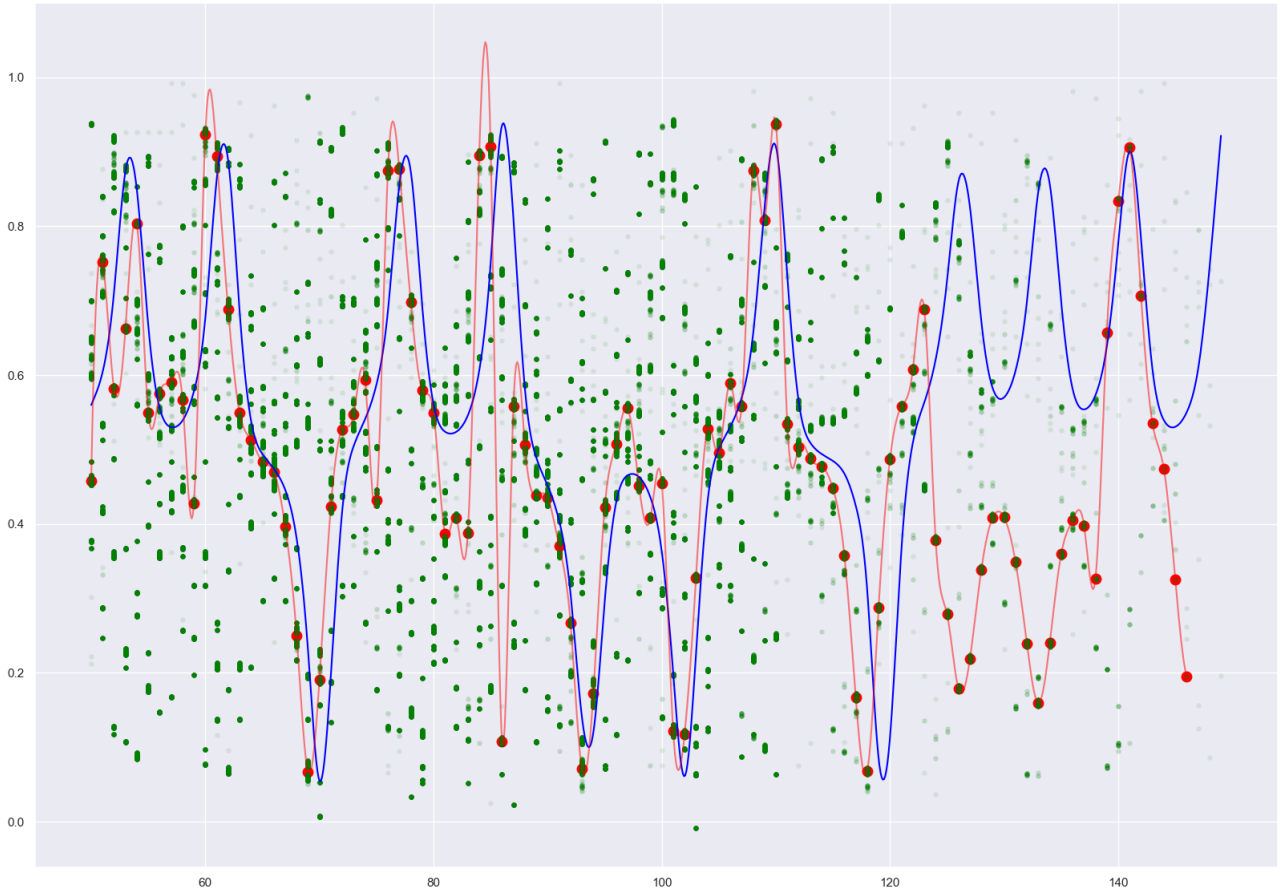

Рис. 1: Пример работы алгоритма. Синий - истинная траектория, красный - предсказанная объединенная траектория, зеленые точки - предсказанные значения точек. Алгоритм определения непрогнозируемых точек - ForcedPredictionNPM. Алгоритм объединения - 'db'. Тестовая выборка - 50. Длина траектории - 100.

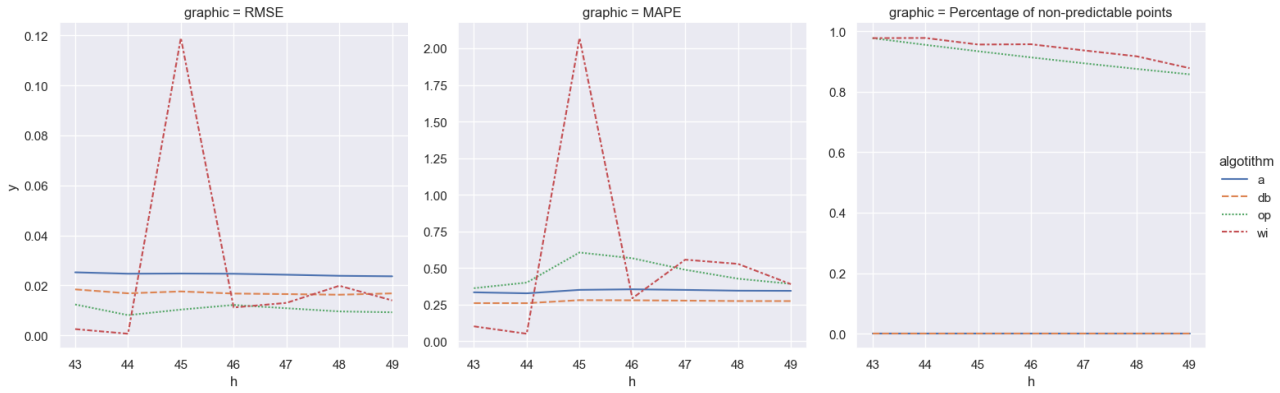

Рис. 2: Графики зависимостей RMSE, MAPE и процента непрогнозируемых точек в зависимости от алгоритма объединения прогнозов. Алгоритм определения непрогнозируемых точек - ForcedPredictionNPM. Тестовая выборка - 50. Длина траектории - 100.

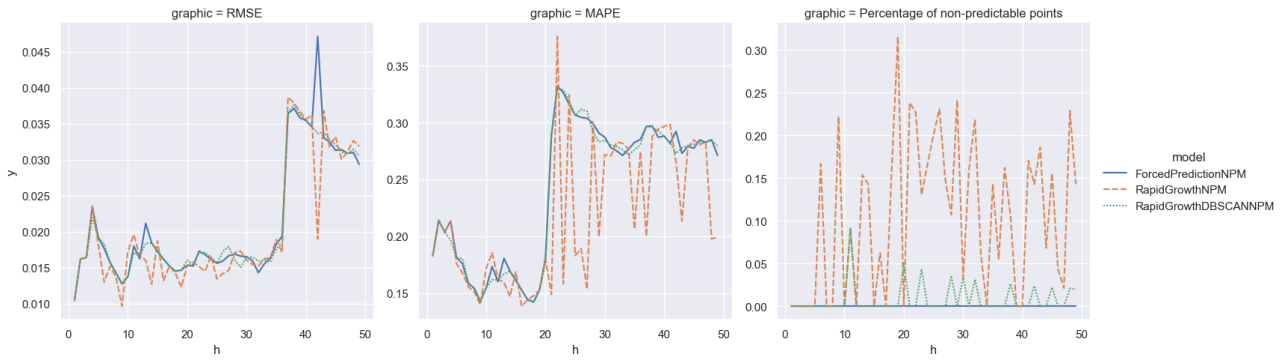

Рис. 3: Графики зависимостей RMSE, MAPE и процента непрогнозируемых точек в зависимости от алгоритма определения непрогнозируемых точек. Алгоритм объединения прогнозов - 'db'. Тестовая выборка - 50. Длина траектории - 100.

Видно на Рис.2, что лучше всего себя показал алгоритм 'db'. Если 'db' и 'op' и показывают себя лучше в плане ошибки, то это достигается лишь за счет малого количества прогнозируемых точек. Алгоритм 'a' лучше использовать при малых и не осциллирующих обучающих данных. Ибо все этот алгоритм выравнивает все пики. В дальнейшем все сравнения будут проведены с алгоритмом 'db'.

Их Рис.3 можно сделать вывод, что сильной разницы между ForcedPrediction и RapidGrowthDBScan нету. Но второй алгоритм обеспечивает чуть лучший результат в плане ошибки, выкидывая некоторые точки. Но алгоритм RapidGrowth ведет себя несколько иначе. Он отбраковывет больше точек, но значительного преимущества не достигает. Соответственно, рекомендуется использовать RapidGrowthDBScan.

.

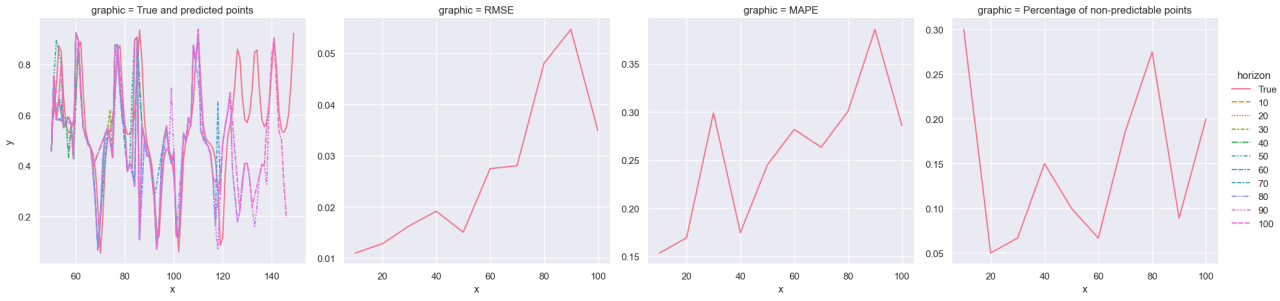

Рис. 4: Предсказанные траектории в зависимости от длины траектории. Графики RMSE, MAPE и процент непрогнозируемых точек. Алгоритм определения непрогнозируемых точек - RapidGrowth. Алгоритм объединения - 'db'. Тестовая выборка - 50. Длина траектории - 100.

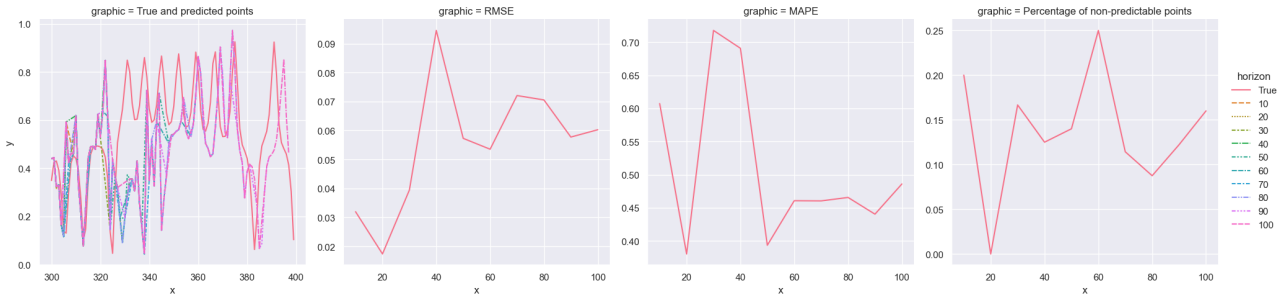

Рис. 5: Предсказанные траектории в зависимости от длины траектории. Графики RMSE, MAPE и процент непрогнозируемых точек. Алгоритм определения непрогнозируемых точек - RapidGrowth. Алгоритм объединения - 'db'. Тестовая выборка - 300. Длина траектории - 100.

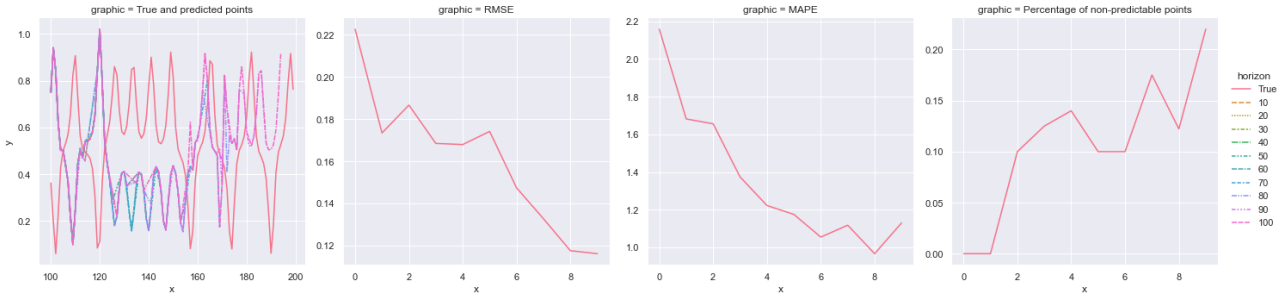

Рис. 6: Предсказанные траектории в зависимости от длины траектории. Графики RMSE, MAPE и процент непрогнозируемых точек. Алгоритм определения непрогнозируемых точек - ForcedPrediction. Алгоритм объединения - 'db'. Тестовая выборка - 100. Длина траектории - 100.

Как можно увидеть на Рис.5 и Рис.6 точность в плане ошибки никак не зависит от размера тестовой выборки. Более тем, ошибка не растет монотонно с ростом длины прогнозируемой траектории.

Далее на Рис.7, Рис.8, Рис.9 будут приведены графики сравнения предложенныого алгоритма Markov Chain и алгоритма Clustering. Во всех экспериментах был использован алгоритм определения непрогнозируемых точек - ForcedPrediction. RMSE и MAPE считались только на тех точках, которые были прогнозированы и не были помечены как непрогнозируемые.

Откуда видно, что глобально преимущества одного или иного алгоритма в плане ошибки нету. На деле все будет меняться от выборки к выборке. Но что сразу видно, то это значительное преимцщество по проценту непрогнозируемых точек. При одних и тех же входных условиях алгоритм Markov Chain демонстрирует значительное преимущество по данному показателю без ущерба в точености.

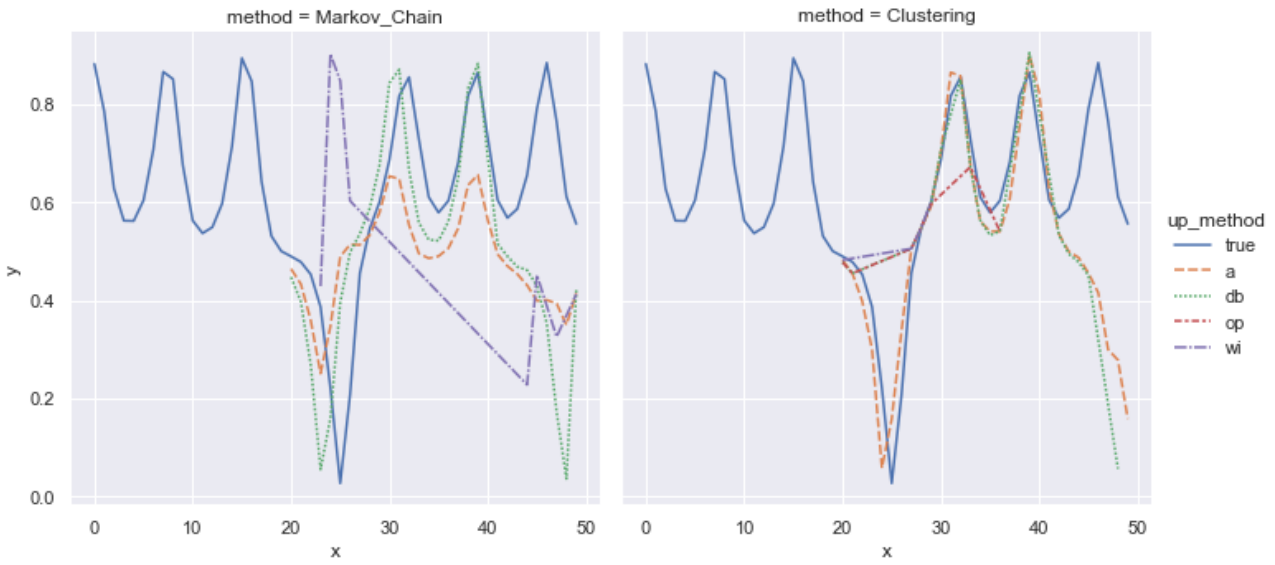

Рис. 7: Различные прогнозируемые траектории для алгоритмов Markov Chain и Clustering. Цветом обозначены алгоритмы объединения значении. Синии - истинная траектория, Оранжевый - 'a', Зеленый - 'db', Красный - 'op', Фиолетовый - 'wi'

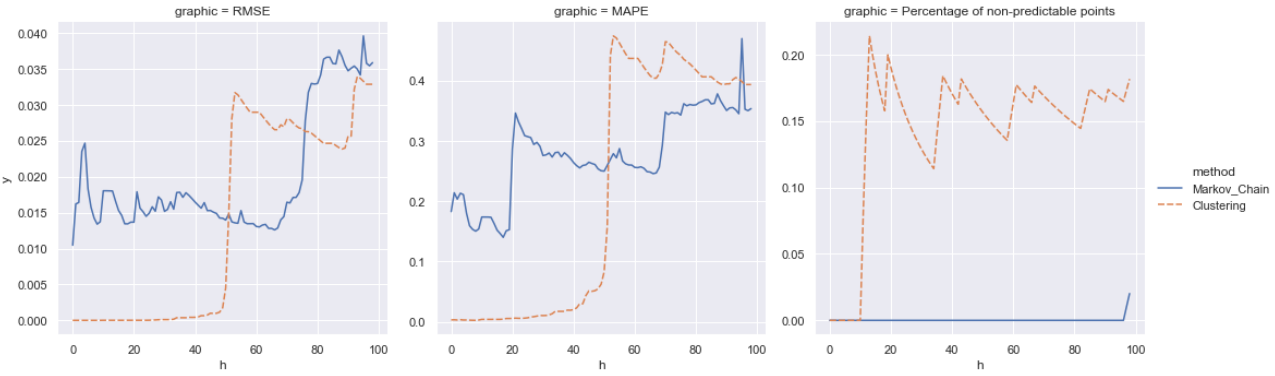

Рис. 8: Сравнение Markov Chain и Clustering. Предсказанные траектории. График зависимости RMSE, MAPE и процент непрогнозируемых точек. Алгоритм объединения - 'db'. Тестовая выборка - 50. Длина траектории - 100.

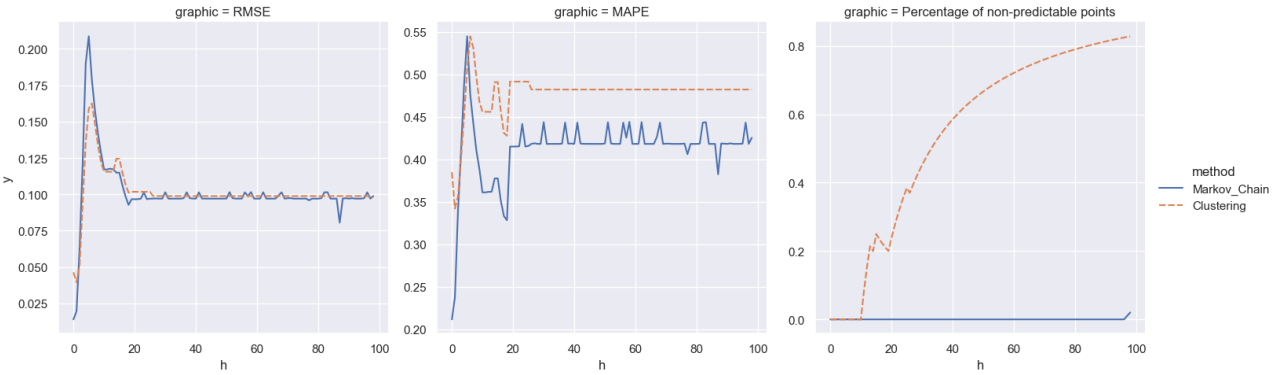

Рис. 9: Сравнение Markov Chain и Clustering. Предсказанные траектории. График зависимости RMSE, MAPE и процент непрогнозируемых точек. Алгоритм объединения - 'db'. Тестовая выборка - 300. Длина траектории - 100.

5. Заключение

В этой работе было предложен новый способ прогнозирования за горизонтом прогнозирования. И по итогам экспериментов алгоритм проявил себя с лучшей стороны. А именно обеспечил гораздо меньший процент непрогнозируемых точек без ущерба к точности. К тому же этот алгоритм позволяет проводить прогнозы при тестовой выборке от 2-х точек. То для алгоритма Clustering для анологичной точности необходимо минимум 25 точек.

В будущем можно развить алгоритм с использованием события Gradient - разница между разностью текущей и предыдущей парой. Но на данный момент этот алгоритм является неустойчивым и требует дальнейшего развития.

Список литературы

1. Y. Bao, T. Xiong, Z. Hu, Multi-step-ahead time series prediction using multiple-output support vector regression / Y. Bao, T. Xiong, Z. Hu // Neurocomputing — 2014. — 129. — 482-493.
2. Ben Taieb, Souhaib and Bontempi, Gianluca and Atiya, Amir, Sorjamaa, Antti. A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition / Ben Taieb, Souhaib and Bontempi, Gianluca and Atiya, Amir, Sorjamaa, Antti. // Expert Systems with Applications. — 2011. — 39.
3. V.A. Gromov, P.S. Baranov, A. Tsybakin, Prediction After a Horizon of Predictability: Non-Predictable Points and Partial Multi- Step Prediction for Chaotic Time Series / V.A. Gromov, P.S. Baranov, A. Tsybakin // Neurocomputing — 2020.
4. Ester, M., H. P. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. / Ester, M., H. P. Kriegel, J. Sander, and X. Xu // In: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, Portland, OR — AAAI Press — pp. 226-231 — 1996.
5. D. Wishart, A numerical classification methods for deriving natural classes / Nature 221 — 1969 — pp. 97–98.