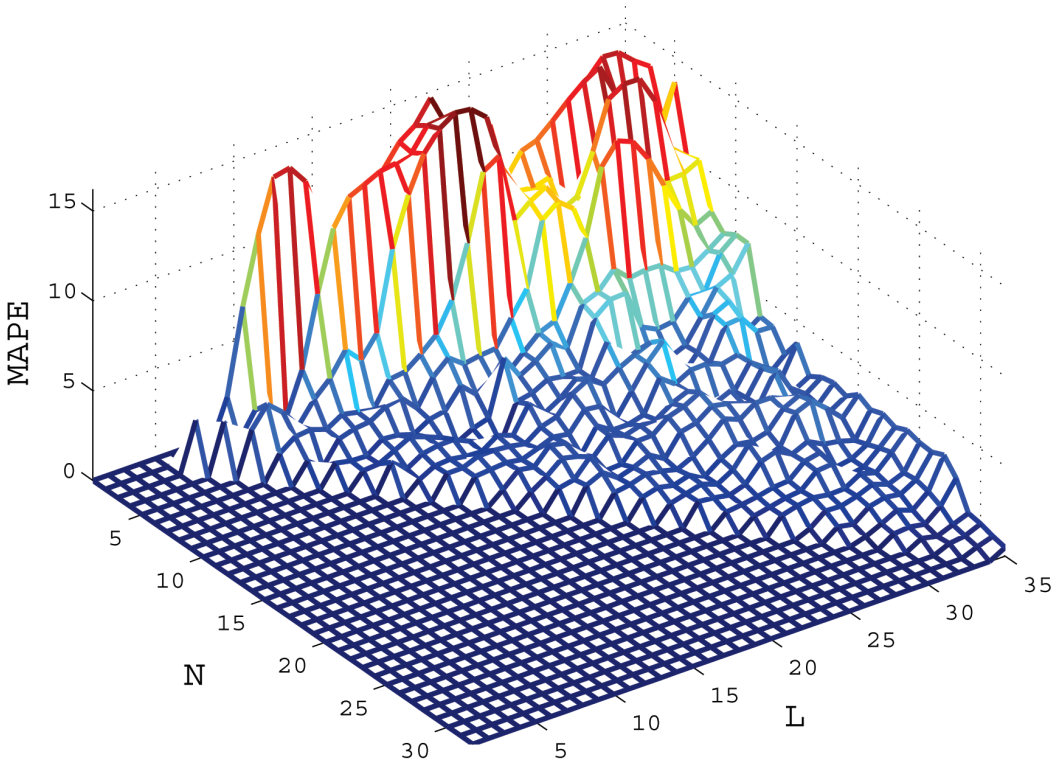
ISSN 2223-3792

Машинное обучение и анализ данных

Июнь 2011 **Том 1, номер 1**



Машинное обучение и анализ данных

Journal of Machine Learning and Data Analysis
ISSN 2223-3792 Rus

Журнал публикует статьи, содействующие развитию теоретических и прикладных методов машинного обучения и интеллектуального анализа данных. Журнал, прежде всего, предназначен для публикации результатов работ аспирантов и студентов, изучающих курс «Численные методы машинного обучения» и занимающихся теоретическими и эмпирическими исследованиями свойств алгоритмов регрессии и классификации. Приветствуются также обзорные, фундаментальные и методические статьи исследователей, работающих в области машинного обучения.

Тематика журнала:

- регрессионный анализ,
- классификация,
- кластеризация,
- многомерный статистический анализ,
- байесовские методы регрессии и классификации,
- методы прогнозирования временных рядов,
- методы оптимизации в задачах машинного обучения и анализа данных,
- методы визуализации данных,
- обработка и распознавание речи и изображений,
- анализ и понимание текста, информационный поиск,
- прикладные задачи анализа данных.

Научный редактор: В. В. Стрижов (stijov@ccas.ru)
Вчрстка: А. А. Мафусалов, П. А. Сечин

Московский физико-технический институт
Факультет управления и прикладной математики
Кафедра «Интеллектуальные системы»

Москва, 2011

От редактора

Первый номер журнала содержит результаты семестровых работ студентов кафедры «Интеллектуальные системы» третьего и четвертого курсов.

Задачей третьекурсников являлось исследование свойств алгоритмов прогнозирования. В процессе выполнения работ им требовалось изучить методы и технику написания научных статей и проведения вычислительных экспериментов, а также сопутствующие этому процессу технологии — язык разметки научных текстов LaTeX, формат представления библиографической записи BibTeX, язык программирования высокого уровня m-code. Для выполнения работ были использованы следующие инструменты: WinEdit/TeXnicCenter — набор текста, MikTex — компиляция статьи, JabRef — создание библиографический базы, SciLab/Octave/Matlab — проведение вычислительных экспериментов.

Работа включала следующие этапы: сбор и анализ литературы, математическая постановка задачи, описание метода решения задачи и исследование его свойств, проведение вычислительного эксперимента. Каждому студенту предлагалась персональная тема, по которой он анализировал публикации отечественных и зарубежных исследователей за последние десять лет, ставил задачу и делал доклад для группы о предварительно полученных результатах. Далее выполнялось математическое описание метода, делался промежуточный доклад о состоянии работ. Последним шагом работы был вычислительный эксперимент, иллюстрирующий свойства метода и использующий синтетические или реальные данные. Каждая статья рецензировалась одногруппниками автора, синхронизация работ выполнялась на сайте SourceForge.org, проект «MLalgorithms». По результатам работ делался двадцатиминутный доклад.

Студенты четвертого курса изучали методы руководства научно-исследовательскими проектами. При этом каждый получал две роли: руководителя и технолога. Ответственность за выполнение проекта возлагалась на руководителя. Руководители должны были поставить и решить одну из задач прогнозирования, выполнить вычислительный эксперимент, подтверждающий адекватность прогноза и написать отчет. Размер отчета ограничивался двумя страницами. Результатом проекта являлся код прогностического алгоритма. Технологи выполняли следующие работы, принятые в промышленном программировании: разработка архитектуры программных систем, общая для всех проектов; контроль стиля написания кода и документации; юнит-тестирование отдельных модулей; системное тестирование; оптимизация и профилирование кода; создание тестовых библиотек; синхронизация работ; верстка данного журнала. Результатом выполнения работ стала программная система прогнозирования многомерных временных рядов, имеющая унифицированный интерфейс и содержащая тринадцать алгоритмов. Оценка объема кода — более десяти тысяч строк.

Подробные материалы проведения вышеперечисленных работ и ссылки на сами работы и код находятся на сайте MachineLearning.ru, в статьях «Численные методы обучения по прецедентам», «Автоматизация и стандартизация научных исследований» и «Руководство исследовательскими проектами».

Успехов в научных исследованиях!
Вадим Викторович Стрижов

Содержание

Многомерная гусеница, выбор длины и числа компонент

Л. Н. Леонтьева
Московский физико-технический институт, ФУПМ, каф. «Интеллектуальные системы»

В работе описывается метод гусеницы (SSA) и его применение для прогнозирования временных рядов. Алгоритм основан на выделении из изучаемого временного ряда некоторого набора его главных компонент и последующего построения прогноза по выбранному набору. Исследуется зависимость точности прогноза от выбора длины гусеницы и числа ее компонент. В вычислительном эксперименте приводятся результаты работы алгоритма на периодических рядах с разным рисунком внутри периода, на рядах с нарушением периодичности, а так же на реальных рядах почасовой температуры в Москве.
Ключевые слова: *прогнозирование, singular spectrum analysis, сингулярное разложение.*

Введение

Данная работа посвящена методу анализа и прогноза временных рядов “Гусеница”, в зарубежной литературе этот метод также называется Singular Spectrum Analisys (SSA). Данному методу посвящены книги [1, 2, 3, 4]. Мы использовали книгу [1] как основной источник сведений о методе гусеницы и придерживались используемых там обозначений и понятий. Одной из основных задач данной работы является исследование зависимости качества прогноза, построенного с помощью метода гусеницы, от длины гусеницы и числа ее компонент [3, 5]. В конце раздела 3 сделаны некоторые теоретические выводы по этому вопросу, а в разделе 4 представлены результаты работы алгоритма на данных почасовой температуры в зависимости от выбора параметров. Метод гусеницы применяется для решения довольно широкого круга задач [8] таких как: разбиение ряда на интерпретируемые составляющие [6], подавление шумов и сглаживание, заполнение пропущенных значений в данных [7] и многих других задач.

Постановка задачи

Дан временной ряд $T = \{x_i\}_{i=1}^{n}$, где n — длина временного ряда, i — номер отсчета. Требуется, задав параметр l, $1 < l < n$ (длину гусеницы) разложить ряд в сумму компонент (используя метод главных компонент), выбрать часть из них и построить по ним продолжение ряда $\{x_i\}_{i=1}^{n+\tau}$.

Предполагаем, что в рассматриваемом временном ряду нет пропущенных значений и он имеет периодическую составляющую с периодом τ, на который и производится прогноз.

Для контроля качества алгоритма прогноза разбиваем множество индексов $N = 1, \ldots, n$ на два подмножества $J_1 = n - \tau + 1, \ldots, n$ и $J_2 = 1, \ldots, n - \tau$. Выделяем во временном ряде T последовательных значений $\{x_i | i \in J_1\}$ (контрольную выборку), которые с помощью алгоритма прогнозируем по предыдущим значениям $\{x_i | i \in J_2\}$. В качестве критерия качества прогноза использовались два функционала: SSE (сумма квадратов ошибок) и MAPE (средняя абсолютная процентная ошибка):

$$SSE = \sum_{i=1}^{\tau} |\tilde{x}_i - x_i|^2,$$

$$MAPE = \frac{1}{\tau} \sum_{i=1}^{\tau} 100 \frac{|\tilde{x}_i - x_i|}{|x_i|},$$

где $\tilde{x}_i$ — спрогнозированное значение в точке i, x_i — фактическое значение в точке i.

Описание алгоритма

Анализ временного ряда

Для последующего разложение ряда по главным компонентам преобразуем ряд в траекторную матрицу X, которую строим следующим образом:

$$X = \begin{pmatrix} x_1 & x_2 & \ldots & x_k \\ x_2 & x_3 & \ldots & x_{k+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_l & x_{l+1} & \ldots & x_n \end{pmatrix}. \tag{1}$$

где $k = n - l + 1$, k — время жизни гусеницы. Матрицу (1) будем называть нецентрированной траекторной матрицей, порожденной гусеницей длины l.

Замечание 1. Проводимый в дальнейшем анализ главных компонент может проводиться как по центрированной, так и по нецентрированной выборкам. Для упрощения выкладок рассмотрим простейший нецентрированный вариант.

Построим ковариационную матрицу следующим образом:

$$C = \frac{1}{k} X^T X.$$

Выполним её сингулярное разложение:

$$C = V \Lambda V^T,$$

где $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_l)$ — диагональная матрица собственных чисел, $V = [v^{(1)}, \ldots, v^{(l)}]$ — ортогональная матрица собственных векторов. При этом будем предполагать, что собственные векторы упорядочены по убыванию соответствующих собственных чисел, т. е. $\lambda_1 > \lambda_2 > \ldots > \lambda_l$. Вычислим матрицу U нецентрированных главных компонент:

$$U = V^T X = (U_1, \ldots, U_l)^T.$$

Восстановим траекторную матрицу по некоторому поднабору главных компонент, т. е. для $\tilde{V} = [v^{(i_1)}, \ldots, v^{(i_r)}]$, $r \leqslant l$ и $\tilde{U} = \tilde{V}^T X$ вычисляется матрица $\tilde{X} = \tilde{V}\tilde{U}$.

После восстановления матрицы $\tilde{X}$ исходная последовательность восстанавливается усреднением по побочным диагоналям матрицы $\tilde{X}$:

$$\tilde{x}_s = \begin{cases} \frac{1}{s} \sum\limits_{j=1}^{s} \tilde{x}_{j,s-j+1} & 1 \leqslant s \leqslant l, \\ \frac{1}{l} \sum\limits_{j=1}^{l} \tilde{x}_{j,s-j+1} & l \leqslant s \leqslant k, \\ \frac{1}{n-s+1} \sum\limits_{j=1}^{n-s+1} \tilde{x}_{j+s-k,k-j+1} & k \leqslant s \leqslant n. \end{cases}$$

С геометрической точки зрения операция получения главных компонент есть изображение исходной выборки в базисе, составленном из выбранных собственных векторов, а восстановление — проектирование исходной выборки на гиперплоскость, порожденную выбранным набором собственных векторов ковариационной матрицы.

Прогноз временного ряда Перейдем к прогнозированию временных рядов методом гусеницы. Для начала определимся с тем, что мы будем понимать под продолжением ряда.

Определение 1. *Числовой ряд $\{x_i\}_{i=1}^{n+1}$ называется продолжением ряда $\{x_i\}_{i=1}^{n}$, если порождаемая им при гусеничной обработке выборка лежит в той же гиперплоскости, что и у исходного ряда.*

Рассмотрим систему уравнений:

$$\begin{cases} \sum\limits_{j=1}^{s} h_j v_1^j = x_{n-l+1}, \\ \qquad \ldots \\ \sum\limits_{j=1}^{s} h_j v_{l-1}^j = x_n. \end{cases} \tag{2}$$

Введем следующие обозначения: $\mathbf{v} = (v_l^{(i_1)}, v_l^{(i_2)}, \ldots, v_l^{(i_r)})$, где $0 < i_1 < \ldots < i_r < l$, и

$$V^* = \begin{pmatrix} v_1^{(i_1)} & v_1^{(i_2)} & \ldots & v_1^{(i_r)} \\ v_2^{(i_1)} & v_2^{(i_2)} & \ldots & v_2^{(i_r)} \\ \vdots & \vdots & \ddots & \vdots \\ v_{l-1}^{(i_1)} & v_{l-1}^{(i_2)} & \ldots & v_{l-1}^{(i_r)} \end{pmatrix}.$$

Заметим, что

$$\tilde{V} = \begin{pmatrix} V^* \\ \mathbf{v} \end{pmatrix}.$$

Также пусть $Q = (q_i)_{i=1}^{l-1} = (x_{n-l+2}, \ldots, x_n)^T$ и $\bar{h} = (h_1, \ldots, h_r)^T$. В этих обозначениях система (2) запишется как

$$V^* \bar{h} = Q. \tag{3}$$

Определение 2. *Обобщенным решением системы (3) назовем решение системы*

$$(V^*)^T V^* \bar{h} = (V^*)^T Q.$$

Определение 3. *Величину*

$$b = \mathbf{v}\bar{h}^*,$$

где $\bar{h}^*$ — решение системы (3), назовем обобщенным продолжением рассматриваемого ряда.

Учитывая (3), можно записать для прогнозируемого значения x_{n+1} следующую формулу:

$$x_{n+1} = \mathbf{v}((V^*)^T V^*)^{-1}(V^*)^T Q. \tag{4}$$

Выбор параметров В этом разделе мы обсудим роль параметров базового метода SSA и принципа их выбора. В базовом методе SSA есть два параметра. Первый — это целое число l, длина гусеницы, а второй параметр является структурным — это способ группировки главных компонент.

Дадим несколько рекомендаций по выбору длины гусеницы:

— Сингулярные разложения одного и того же ряда длины n, соответствующие выбору длины гусеницы l и $n - l + 1$ эквивалентны. Следовательно, для анализа структуры временного ряда не имеет смысла брать длину гусеницы, большую чем половина длины ряда.
— Чем больше длина гусеницы, тем более детальным получается разложение исходного ряда. Таким образом, наиболее детальное разложение достигается при выборе длины гусеницы, приблизительно равной половине длины ряда ($l \sim n/2$). Причем, чем больше длина гусеницы, тем более детальным получается разложение исходного ряда.
— Маленькая длина гусеницы может привести к смешиванию интерпретируемых компонент ряда.
— При решении задачи выделения периодической компоненты с периодом τ следует выбирать длину гусеницы l кратной τ.
— В общем метод гусеницы устойчив относительно изменения длины гусеницы. Эффект проявляется не столько в количественном, сколько в качественном смысле.

Теперь обсудим важные моменты связанные с отбором главных компонент. Пусть длина гусеницы l фиксирована и мы уже имеем сингулярное разложение траекторной матрицы исходного ряда. Тогда следующим шагом является группировка членов сингулярного разложения:

— Если мы восстановили компоненту ряда только с помощью одной собственной тройки (собственное значение, собственный вектор и главная компонента) и оба сингулярных вектора имеют похожую форму, то восстановленная компонента будет иметь примерно такую же форму. Это правило означает, что, имея дело с единственной собственной тройкой, часто можно предсказать поведение соответствующей компоненты временного ряда. Например, если оба сингулярных вектора собственной тройки похожи на линейные ряды, то соответствующая составляющая ряда также будет близкой к линейной. Если сингулярные векторы имеют экспоненциальную форму, то и компонента ряда будет такой же. Монотонные сингулярные векторы соответствуют монотонной компоненте ряда. Синусоидальные векторы порождают гармоническую составляющую ряда.
— Чем больше собственное значение главной компоненты, тем больше вклад соответствующей восстановленной компоненты ряда.

Вычислительный эксперимент

Сперва протестируем работу алгоритма на простых периодических рядах. Строим прогноз на период для зашумленного синуса с периодом 50. Даже для длины гусеницы не кратной периоду ряда результат работы алгоритма хороший.

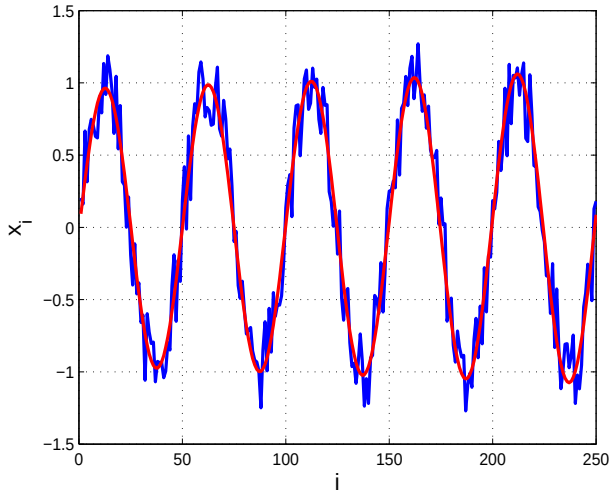


Рис. 1. Прогнозирование зашумленного синуса по двум первым компонентам

На рис. 1 показан прогноз, сделанный по двум компонентам и длине гусеницы равной 80.

А теперь мы возьмем только одну первую главную компоненту, а длину гусеницы оставим прежней.

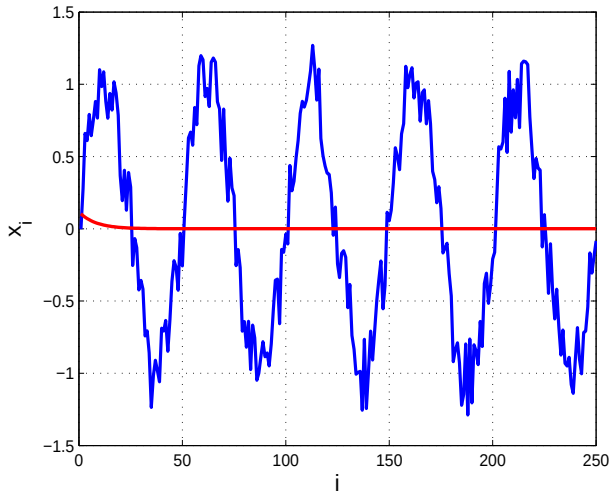


Рис. 2. Прогнозирование зашумленного синуса по первой компоненте

Мы существенно не угадали период и длина гусеницы ему не кратна. Первая ГК не отражает основной период, а пытается отразить тренд, которого нет.

Теперь вместо зашумления внесем в тот же синус с периодом 50 два сильных выброса и спрогнозируем его также по двум компонентам с длинной гусеницы 80.

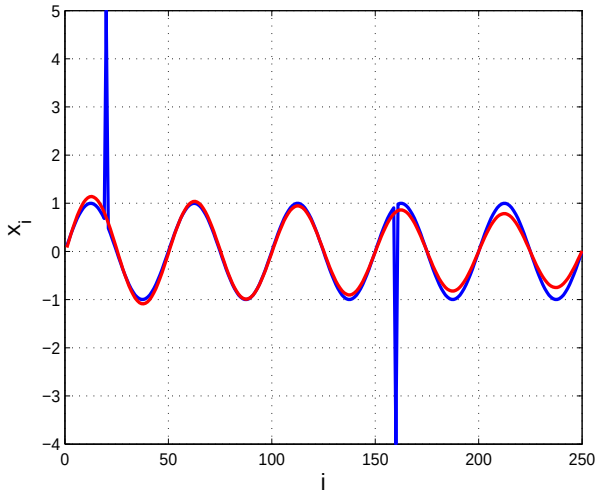


Рис. 3. Прогнозирование синуса с двумя сильными выбросами по двум компонентам

Результат работы показан на рис. 3, из которого можно сделать вывод, что алгоритм в некоторой степени устойчив к выбросам. Однако при увеличении числа компонент регрессионные остатки значительно увеличиваются. На рис. 4 показан результат работы алгоритма на том же ряде с той же длинной гусеницы, но число компонент выбрано равное 3.

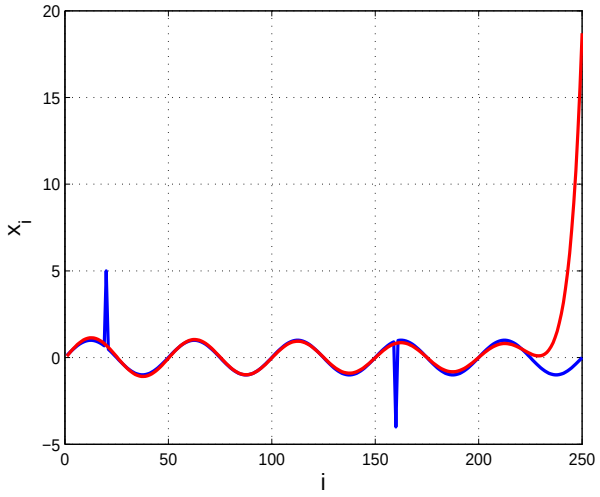


Рис. 4. Прогнозирование синуса с двумя сильными выбросами по трем компонентам

Далее рассмотри ряд являющийся суммой двух периодических зашумленных рядов с периодами 57 и 70.

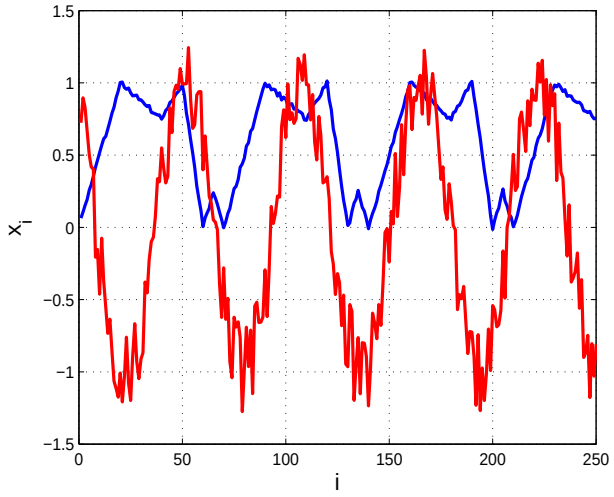


Рис. 5. Два зашумленных несинфазных ряда с разными периодоми

То есть суммарный ряд имеет два периода, которые сильно отличаются по длине, не синфазны и не кратны. Чтобы их корректно восстановить мы взяли первые четыре компоненты и длину гусеницы равную 80.

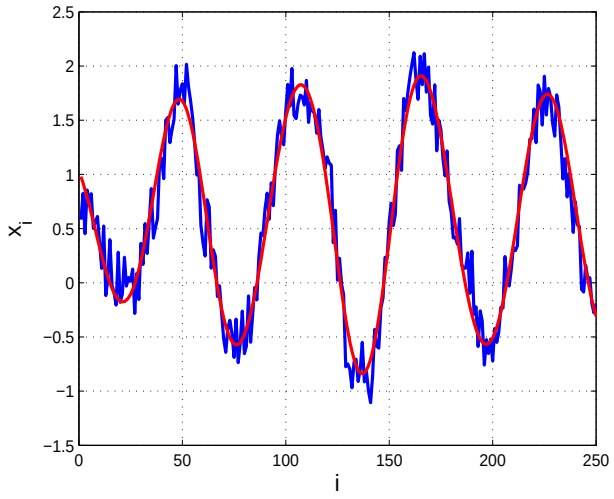


Рис. 6. Прогнозирование суммы двух периодических рядов

Для проведения следующего эксперимента были использованы данные почасовой погоды в Москве в апреле 2011 года. Мы прогнозируем погоду на ближайший час по трем предыдущим дням.

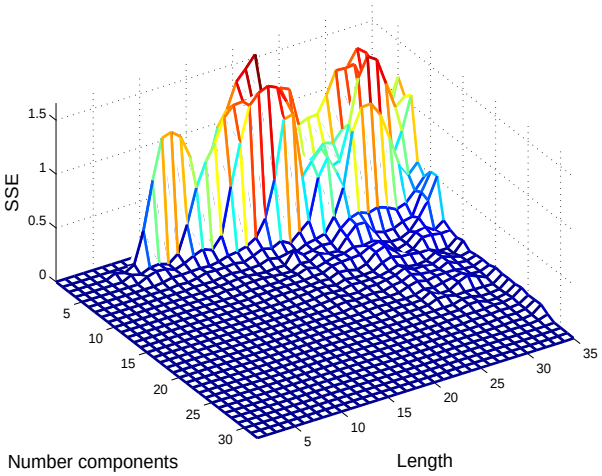


Рис. 7. График зависимости SSE от длины и числа компонент на обучении

Как видно ошибка на обучении мало зависит от длины гусеницы, но резко уменьшается при увеличении числа компонент.

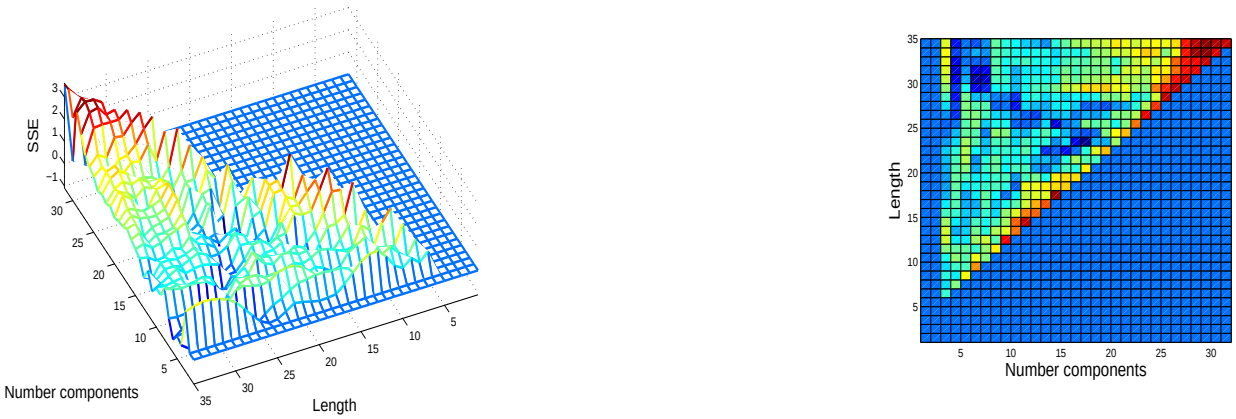


Рис. 8. График зависимости SSE от длины и числа компонент на прогнозе в логарифмическом масштабе

Наименьшая ошибка наблюдается на длине гусеницы от 22 до 26 и числе компонент от 13 до 18. При числе компонент порядка длинны гусеницы (то есть при почти полном наборе компонент) ошибка резко возрастает, тем самым иллюстрируется переобучение нашего алгоритма. Чем меньше длина гусеницы, тем раньше (при меньшем числе компонент) начинается переобучение.

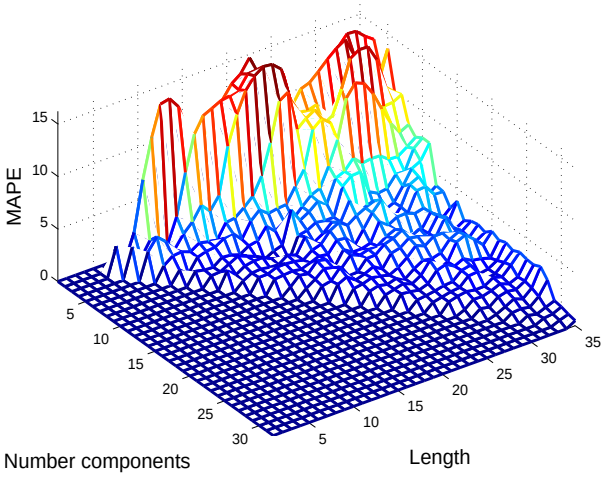


Рис. 9. График зависимости MAPE от длины и числа компонент на обучении

Таким образом, MAPE также как и SSE на обучении мало зависит от длины гусеницы и резко уменьшается при увеличении числа компонент.

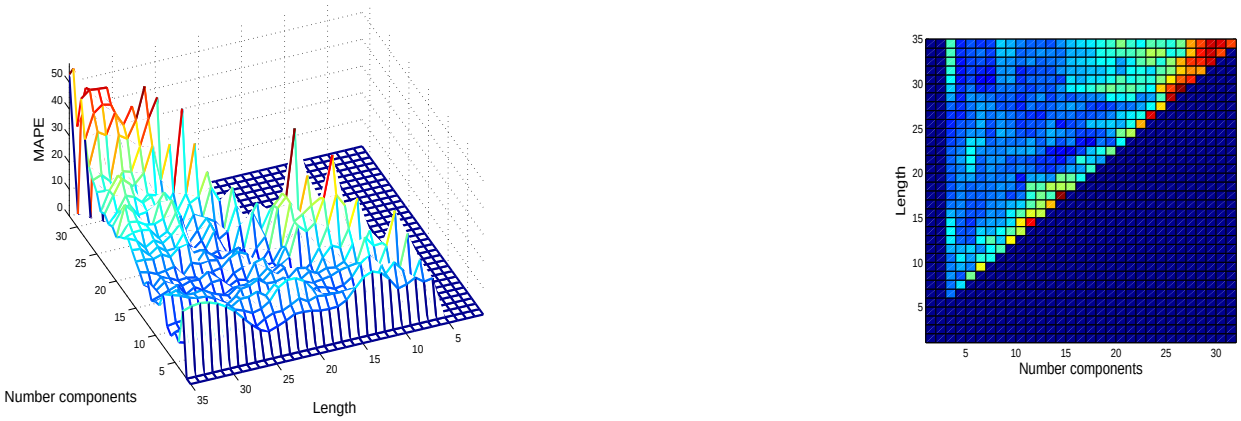


Рис. 10. График зависимости MAPE от длины и числа компонент на прогнозе

Из рис. 10 можно заключить, что MAPE почти всюду, кроме областей переобучения, принимает значения меньше 10%. За счет переобучения при выборе числа компонент близкого по величине к длине гусеницы значение MAPE достигает $40 - 50\%$.

Заключение

В данной работе была исследована зависимость SSE и MAPE при прогнозировании временных рядов методом “Гусеница” в зависимости от значений входных параметров, то есть длины и числа компонент гусеницы. Результаты вычислительного эксперимента показали, что для минимизации SSE необходимо выбирать длину гусеницы кратной (или почти кратной) периоду ряда. Переобучение, связанное с большим числом компонент по которым строится прогноз, наступает (для рядов значений почасовой температуры) примерно на 10 компонентах. Помимо этого была исследована эффективность работы алгоритма на зашумленных рядах и рядах с выбросами. Зашумленные ряды метод гусеницы сглаживает и строит хороший прогноз. При наличии в рядах выбросов метод неустойчив к выбору числа компонент.

Код, необходимый для повторения вычислительного эксперимента расположен на сайте:
`https://mlalgorithms.svn.sourceforge.net/svnroot/mlalgorithms/GaterpillarLearning/`

Литература

[1] В. Н.Солнцев, Д. Л. Данилов, А. А. Жиглявский. *Главные Компоненты Временных Рядов: Метод "Гусеница"*, С.-Петербургский государственный университет, 1997.

[2] Н. Э. Голяндина. *Метод "Гусеница"-SSA: анализ временных рядов*, С.-Петербургский государсвенный университет, 2004.

[3] Н. Э. Голяндина. *Метод "Гусеница"-SSA: прогноз временных рядов*, С.-Петербургский государсвенный университет, 2004.

[4] N. Golyadina, V. Nekrutkin. *Analysis of Time Series Structure: SSA and Related Techniques*, Chapman&Hall/CRC, 2001.

[5] Nina Golyandina. *On the choice of parameters in Singular Spectrum Analysis and related subspace-based methods*, Statistics and Its Interface, 2010, №3: 259-279.

[6] Ф. И. Александров. *Выделение аддитивных компанент временного ряда на основе метода "Гусеница"*, С.-Петербургский государсвенный университет, 2003.

[7] Н. Э. Галядина, Е. В. Осипов. *Метод "Гусеница"-SSA для анализа временных рядов с пропусками, С.-Петербургский государсвенный университет.*

[8] Hossein Hassani. *Singular Spectrum Analysis: A Relatively New and Powerful Technique for Time series Analysis and Forecasting*, The 31st Annual International Symposium on Forecasting, 2011.

Выбор функции активации при прогнозировании нейронными сетями

Г. И. Рудой

Московский физико-технический институт, ФУПМ, каф. «Интеллектуальные системы»

Целью работы является исследование зависимости качества прогнозирования временных рядов нейронными сетями от параметров нейронной сети. В частности, анализируется зависимость от выбранной функции активации нейронов в сети, а также от параметров этой функции. Функция активации описывает выходное значение нейрона в зависимости от взвешенной суммы его входов и порогового значения срабатывания. Рассматриваются сети с прямым распространением сигналов (без обратной связи). Приводятся результаты вычислительного эксперимента по прогнозированию нейронными сетями различных временных рядов и анализируется качество прогнозов при различных функциях активации и прочих параметрах сети.

Ключевые слова: *нейронные сети с прямым распространением сигналов, линейные нейронные сети, многослойные нейронные сети, гетерогенные нейронные сети, алгоритм обратного распространения ошибки.*

Введение

Нейронная сеть с прямым распространением сигналов [2] [3] [4] — такая сеть, в которой сигнал распространяется только в одном направлении, от слоя к слою. Каждый элемент сети (нейрон) имеет один или несколько входов и один выход. Нейрон представляет собой систему из двух элементов — сумматора и функции активации. Функция активации определяет выходное значение нейрона в зависимости от результата сумматора (взвешенной суммы входов) и некоторого порогового значения. Пороговое значение также может быть представлено как еще один (неявный) вход нейрона, который не соединен ни с одним другим нейроном.

Постановка задачи

Пусть дан временной ряд $x = x(n) \mid n = 1, \ldots, m$, состоящий из некоторых числовых признаков. Требуется построить значение ряда на неизвестном промежутке, то есть, определить $x(m+1)$, $x(m+2)$ и так далее, и минимизировать среднеквадратичную ошибку прогнозирования.

Задача сводится к выбору и сравнительному анализу различных функций активации нейросети, а также различных методов обучения.

После вычисления значения в момент времени $m+1$ (и, возможно, последующие) вычисляется среднеквадратичная ошибка, являющаяся показателем качества прогнозирования, и исследуется ее зависимость от функции активации, числа нейронов, метода обучения и прочих параметров сети.

Пути решения

Совокупность известных значений временного ряда образует обучающую выборку.

Для прогнозирования временных рядов используется метод скользящего окна: выбирается p последовательных элементов, составляющих обучающую выборку и формирующих *образ*, который подаются, соответственно, на p распределительных нейронов сети. P выходных нейронов характеризуют значения функции в момент времени $p+1, p+2, \ldots, p+P$. Тогда конкретные методы прогнозирования различаются архитектурой сети, ее организацией и способом подбора и настройки весовых коэффициентов нейронной сети.

Линейная сеть

Нейронная сеть, состоящая из распределительных нейронов и одного выходного нейрона (1), имеющего линейную функцию активации, называется адаптивным нейронным элементом [5]. Выходное значение такой сети:

$$y = \sum_{i=1}^{n} w_{i1} x_i - T,$$

где T — пороговое значение указанного нейрона, а w_{j1} — весовой коэффициент, соответствующий j-ому распределительному нейрону.

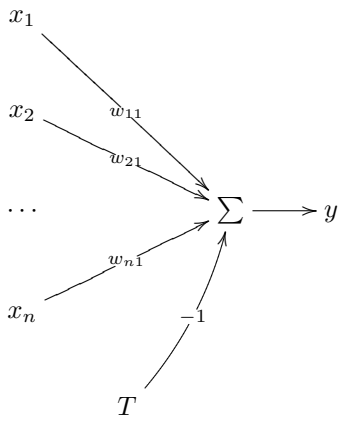


Рис. 1. Нейронная сеть из распределительных нейронов и одного нейрона с линейной функцией активации

Заметим, что нейронную сеть с несколькими выходными элементами можно представить как суперпозицию соответствующего числа нейронных сетей с одним выходным элементом, ввиду независимости выходных элементов друг от друга. В этом разделе для простоты мы будем рассматривать сети с одним выходным элементом.

Для линейной нейронной сети указанный ранее метод скользящего окна соответствует линейной авторегрессии и описывается выражением:

$$\hat{x}(n) = \sum_{k=1}^{p} w_k x(n-p+k-1), \tag{1}$$

где $w_k, k = 1, \dots, p$ — весовые коэффициенты нейронной сети, $\hat{x}(n)$ — оценка значения ряда $x(n)$ в момент времени n. Тогда ошибка прогнозирования определяется выражением

$$e(n) = x(n) - \hat{x}(n).$$

Правило Видроу-Хоффа

Для обучения такой сети используется правило Видроу-Хоффа, известное также под названием *дельта-правила*. Оно предполагает минимизацию среднеквадратичной ошибки нейронной сети, которая для L входных образов определяется следующим образом:

$$E = \sum_{k=1}^{L} E(k) = \frac{1}{2} \sum_{k=1}^{L} (y_1^k - t^k)^2,$$

где $E(k)$ — среднеквадратичная ошибка сети для k-го образа, а y_1^k и t^k — соответственно выходное и эталонное значения нейронной сети для k-го образа.

Правило обучения Видроу-Хоффа базируется на методе градиентного спуска. Согласно этому правилу весовые коэффициенты и пороги нейронной сети необходимо изменять на $t+1$-ой итерации по следующим выражениям:

$$w_{j1}(t+1) = w_{j1}(t) - \alpha \frac{\partial E(k)}{\partial w_{j1}(t)}, \tag{2}$$

$$T(t+1) = T(t) - \alpha \frac{\partial E(k)}{\partial T(t)}, \tag{3}$$

где $j = 1, \dots, n$, α — скорость обучения.

Производные среднеквадратичной ошибки E по данным параметрам:

$$\frac{\partial E}{\partial w_{j1}(t)} = (y_1^k - t^k) x_i^k, \tag{4}$$

$$\frac{\partial E}{\partial T(t)} = -(y_1^k - t^k), \tag{5}$$

где x_i^k — j-ая компонента k-го образа.

Подставляя (4) и (5) в (2) и (3), получаем выражения для обучения нейронной сети по дельта-правилу:

$$w_{j1}(t+1) = w_{j1}(t) - \alpha(y_1^k - t^k)x_i^k, \tag{6}$$

$$T(t+1) = T(t) + \alpha(y_1^k - t^k). \tag{7}$$

Б. Видроу и М. Хофф доказали [5], что данный закон всегда позволяет находить весовые коэффициенты нейронного элемента таким образом, чтобы минимизировать среднеквадратичную ошибку сети независимо от начальных значений весовых коэффициентов.

Алгоритм обучения, основанный на дельта-правиле, состоит из следующих шагов:

1. Задается скорость обучения α $(0 < \alpha < 1)$ и минимальная среднеквадратичная ошибка сети E_m, которую необходимо достичь в процессе обучения.
2. Случайным образом инициализируются весовые коэффициенты и порог нейронной сети.
3. Подаются входные образы на нейронную сеть и вычисляются выходные значения сети.
4. Осуществляется изменение весовых коэффициентов согласно (6) и (7).
5. Алгоритм работает до тех пор, пока суммарная среднеквадратичная ошибка не станет меньше заданной $(E \leqslant E_m)$.

Использование псевдообратной матрицы

Для настройки весовых коэффициентов линейной нейронной сети с целью минимизации среднеквадратичной ошибки можно использовать матричное решение системы линейных уравнений.

Пусть размерность обучающей выборки — L, число выходных нейронов — m, число входных нейронов — n. Тогда матрицы выходных значений, входных значений и весовых коэффициентов, соответственно, имеют вид:

$$Y = \begin{bmatrix} y_1^1 & y_1^2 & \dots & y_1^L \\ y_2^1 & y_2^2 & \dots & y_2^L \\ \dots & \dots & \dots & \dots \\ y_m^1 & y_m^2 & \dots & y_m^L \end{bmatrix} \text{ — матрица выходных значений,}$$

$$X = \begin{bmatrix} x_1^1 & x_1^2 & \dots & x_1^L \\ x_2^1 & x_2^2 & \dots & x_2^L \\ \dots & \dots & \dots & \dots \\ x_n^1 & x_n^2 & \dots & x_n^L \end{bmatrix} \text{ — матрица входных значений,}$$

$$W = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1m} \\ w_{21} & w_{22} & \dots & w_{2m} \\ \dots & \dots & \dots & \dots \\ w_{n1} & w_{n2} & \dots & w_{nm} \end{bmatrix} \text{ — матрица весовых коэффициентов.}$$

Тогда уравнение линейной нейронной сети можно представить в виде:

$$Y = XW$$

В случае использования порогов нейронных элементов в матрицу X добавляется строка, содержащая значения -1, а в матрицу W — строка, в которой находятся пороговые значения нейронных элементов T.

В работе [6] показано, что наилучшим приближенным решением системы линейных уравнений, при котором среднеквадратичная ошибка сети достигает своего наименьшего значения, является выражение

$$W = X^+Y,$$

где X^+ — псевдообратная матрица для матрицы X, определяемая следующим образом:

$$X^+ = (X^TX)^{-1}X^T$$

Это решение единственно [6] и минимизирует среднеквадратичную ошибку сети: $E = \|Y - WX\|^2$.

Существуют различные эффективные способы нахождения псевдообратной матрицы [6].

Заметим, что при использовании псевдообратной матрицы возникают проблемы, когда матрица $X^T X$ является вырожденной.

Многослойная нейронная сеть

Архитектура многослойной нейронной сети (рис. 2) состоит из множества слоев нейронных элементов. *Входной слой* нейронных элементов выполняет распределительные функции, *выходной слой* служит для обработки информации от предыдущих слоев и выдачи результата. Слои, расположенные между входным и выходным слоями, называются *промежуточными* или *скрытыми*. И выходной, и скрытые слои являются обрабатывающими. Выход каждого нейронного элемента предыдущего слоя нейронной сети соединен со всеми входами нейронных элементов следующего слоя.

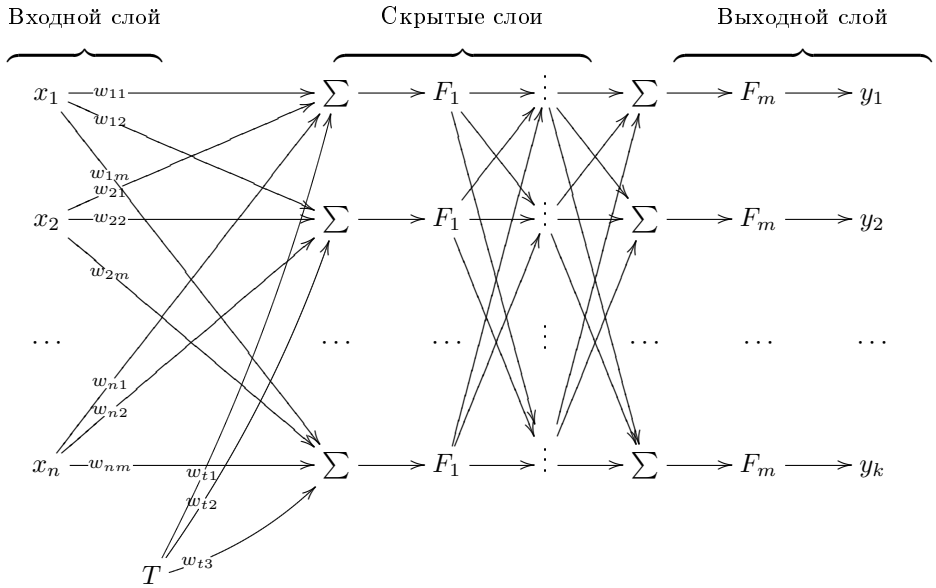


Рис. 2. Нейронная сеть с $m-1$ скрытыми слоями, n входами и k выходами

В качестве функции активации нейронных элементов обычно используются гиперболический тангенс или сигмоидная функция. Если соответствующие элементы имеют одинаковую функцию активации, то сеть называется *гомогенной*, иначе — *гетерогенной*.

Пусть $W^{(i)}$ — матрица весовых коэффициентов i-го слоя. Тогда, например, для гомогенной нейронной сети с двумя скрытыми слоями матрица выходных значений определяется как

$$Y = F_3(F_2(F_1(XW^{(1)})W^{(2)})W^3),$$

где $X = (x_1, x_2, \ldots, x_n)$ — вектор-строка входных сигналов, F_i — определяемый функцией активации оператор нелинейного преобразования для i-го слоя. Для гомогенной нейронной сети $F_i = F_j \mid \forall i, j$, для гетерогенной сети некоторые (либо все) из операторов могут быть различны.

Число слоев в многослойной нейронной сети определяет, каким образом входное пространство образов может быть разбито на подпространства меньшей размерности [2]. Так, двухслойная нейронная сеть с одним слоем нелинейных нейронов разбивает входное пространство образов на классы при помощи гиперплоскости [7]. Трехслойная нейронная сеть, где в качестве двух последних слоев используются нейронные элементы с нелинейной функцией активации, позволяет формировать любые выпуклые области в пространстве решений [7] [8]. Четырехслойная нейронная сеть с тремя нелинейными слоями дает возможность получать область решений любой формы и сложности, в том числе и невыпуклой.

А. Н. Колмогоров показал [9], что любую непрерывную функцию n переменных на единичном отрезке $[0;1]$ можно представить в виде суммы конечного числа одномерных функций:

$$f(x_1, x_2, \ldots, x_n) = \sum_{p=1}^{2n+1} g\left(\sum_{i=1}^{n} \lambda_i \varphi_p(x_i)\right),$$

где функции g и φ_p являются одномерными и непрерывными, $\lambda_i = const$ для всех i.

Данная теорема легла в основу построения многослойных нейронных сетей для аппроксимации функций. Из нее следует, что любую непрерывную функцию $f : [0;1]^n \to [0;1]$ можно аппроксимировать при помощи трехслойной нейронной сети, имеющей n входных, $2n+1$ скрытых и один выходной нейрон. Данный результат был затем обобщен на многослойную сеть с алгоритмом обратного распространения ошибки [10], [3], [11].

Алгоритм обратного распространения ошибки

Алгоритм обратного распространения ошибки был предложен в [12] и является эффективным методом для обучения нейронных сетей. Данный алгоритм минимизирует среднеквадратичную ошибку нейронной сети. Покажем основные идеи, использованные при построении этого алгоритма [2].

Согласно методу градиентного спуска изменение весовых коэффициентов и порогов нейронной сети происходит по следующему правилу:

$$w_{ij}(t+1) = w_{ij}(t) - \alpha \frac{\partial E}{\partial w_{ij}(t)} \tag{8}$$

$$T_j(t+1) = T_j(t) - \alpha \frac{\partial E}{\partial T_j(t)}, \tag{9}$$

где $E = \frac{1}{2}\sum_j (y_j - t_j)^2$ — среднеквадратичная ошибка нейронной сети для одного образа.

В [2] показано, что

$$\frac{\partial E}{\partial w_{ki}} = \gamma_i F'(S_i)\gamma_k \tag{10}$$

$$\frac{\partial E}{\partial T_i} = -\gamma_i F'(S_i) \tag{11}$$

где γ_i — выходное значение i-го нейрона.

Подставляя (10) и (11) в (8) и (9), получаем следующие выражения, показывающие, что для минимизации среднеквадратичной ошибки сети весовые коэффициенты и пороги нейронных элементов должны изменяться следующим образом:

$$w_{ij}(t+1) = w_{ij}(t) - \alpha\gamma_j F'(S_j) y_i \tag{12}$$

$$T_j(t+1) = T_j(t) + \alpha\gamma_i F'(S_j) \tag{13}$$

Эти выражения определяют правило обучения многослойных нейронных сетей в общем виде, которое называется *обобщенным дельта-правилом*.

Определим выражения (12) и (13) для типичных функций активации нейронных элементов. Примем обозначение $S_j = \sum_i w_{ij} y_i - T_j$.

— Для *сигмоидной функции* выходное значение j-го элемента определяется следующим образом:

$$y_j = \frac{1}{1 + e^{-S_j}}$$

В результате обобщенное дельта-правило можно представить в виде:

$$w_{ij}(t+1) = w_{ij}(t) - \alpha\gamma_j y_j (1 - y_j) y_i \tag{14}$$

$$T_j(t+1) = T_j + \alpha\gamma_j y_j (1 - y_j)$$

Ошибка для j-го нейрона выходного слоя:

$$\gamma_j = y_j - t_j,$$

для j-го нейрона скрытого слоя

$$\gamma_j = \sum_{i=1}^{m} \gamma_i y_i (1 - y_i) w_{ji},$$

где m — число нейронов следующего слоя по отношению к слою j.

— Для *биполярной сигмоидной функции* выходное значение j-го элемента определяется по формуле

$$y_j = \frac{2}{1 + e^{-S_j}} - 1$$

Обобщенное дельта-правило:

$$w_{ij}(t+1) = w_{ij}(t) - \frac{1}{2}\alpha\gamma_j(1 - y_j^2)y_i$$

$$T_j(t+1) = T_j(t) + \frac{1}{2}\alpha\gamma_j(1 - y_j^2)$$

Ошибка для j-го нейрона выходного и скрытого слоев, соответственно:

$$\gamma_j = y_j - t_j$$

$$\gamma_j = \frac{1}{2}\sum_i \gamma_i(1 - y_i^2)w_{ij}$$

— Для *гиперболического тангенса* выходное значение j-го нейрона определяется как

$$y_j = \mathrm{th}(S_j) = \frac{e^{S_j} - e^{-S_j}}{e^{S_j} + e^{-S_j}}$$

Производная этой функции имеет вид $F'(S_j) = 1 - y_j^2$, поэтому правило обучения можно представить в виде:

$$w_{ij}(t+1) = w_{ij}(t) - \alpha\gamma_j(1 - y_j^2)y_i \tag{15}$$

$$T_j(t+1) = T_j(t) + \alpha\gamma_j(1 - y_j^2) \tag{16}$$

Ошибка для j-го нейрона выходного и скрытого слоев соответственно:

$$\gamma_j = y_j - t_j$$

$$\gamma_j = \sum_i \gamma_i(1 - y_i^2)w_{ij}$$

Сам алгоритм обратного распространения представляет следующую последовательность шагов.

1. Задаются шаг обучения α $(0 < \alpha < 1)$ и желаемая среднеквадратичная ошибка нейронной сети E_m.
2. Инициализируются случайным образом весовые коэффициенты и пороговые значения нейронной сети.
3. Подаются последовательно образы из обучающей выборки на вход нейронной сети. При этом для каждого входного образа выполняются следующие действия:

   (а) Производится фаза прямого распространения образа по сети, при этом вычисляется выходная активность всех нейронных элементов сети:

   $$y_j = F(\sum_i w_{ij}y_i - T_j),$$

   где индекс j характеризует нейроны следующего слоя по отношению к слою i.

   (б) Осуществляется фаза обратного распространения сигнала, в результате которой определяется ошибка γ_j нейронных элементов для всех слоев сети. При этом, соответственно, для выходного и скрытого слоев:

   $$\gamma_j = y_j - t_j$$

   $$\gamma_j = \sum_i \gamma_i F'(S_i)w_{ji}$$

В последнем выражении i характеризует нейронные элементы следующего слоя по отношению к слою j.

(в) Происходит изменение весовых коэффициентов и порогов нейронных элементов для каждого слоя нейронной сети в соответствии с (12) и (13).

4. Вычисляется суммарная среднеквадратичная ошибка нейронной сети E:

$$E = \frac{1}{2}\sum_{k=1}^{L}\sum_{j}(y_j^k - t_j^k)^2,$$

где L — размерность обучающей выборки.

5. Если $E > E_m$, то происходит переход к шагу 3, иначе алгоритм заканчивается.

Таким образом, алгоритм функционирует до тех пор, пока суммарная среднеквадратичная ошибка сети не станет меньше заданной.

Алгоритм обратного распространения ошибок наделен следующими проблемами:

— Неизвестность выбора числа слоев и количества нейронных элементов в слое.
— Медленная сходимость градиентного метода с постоянным шагом обучения.
— Сложность выбора скорости обучения α. Слишком малая скорость увеличивает время обучения и приводит к нахождению лишь локального минимума, в то время как большая скорость может привести к расхождению процесса.
— Невозможность определения точек локального и глобального минимумов, так как градиентный метод их не различает.
— Влияние случайной инициализации весовых коэффициентов на поиск минимума (неустойчивость).

Заметим, что важную роль играет порядок величин случайно инициализируемых синаптических связей [13] [4]. Так, для сигмоидной функции активации нейронных элементов, при больших по модулю значениях весовых коэффициентов выходная активность будет близка к единице или нулю, и тогда значение выражения $y_j(1 - y_j)$ будет близко к нулю, и, согласно (14), весовые коэффициенты будут изменяться незначительно. Это приведет к тому, что процесс обучения остановится в ближайшем локальном минимуме от стартовой точки. В [13] рекомендуется случайно выбирать значения весовых коэффициентов, имеющих порядок $w_{ij} \approx \frac{1}{\sqrt{n(i)}}$, где $n(i)$ — число элементов в слое i. Другие авторы рекомендуют инициализировать весовые коэффициенты случайными числами в диапазоне порядка $[-0.05; 0.05]$.

Другим важным вопросом является число нейронных элементов в скрытых слоях. С одной стороны, при возрастании их числа растет точность, с другой, при слишком большой размерности скрытых слоев возникает явление *перетренировки сети*, ухудшающее обобщающие способности нейронных сетей. Поэтому число нейронных элементов в скрытом слое должно быть меньше числа тренировочных образцов.

Для нейтрализации застревания метода градиентного спуска в нежелательных минимумах применяется метод тяжелого шарика [13]. В этом случае модификация синаптических связей нейронной сети происходит в соответствии с выражением

$$\Delta w_{ij}(t+1) = -\alpha\gamma_j F'(S_j)y_i + \xi\Delta w_{ij}(t),$$

где ξ — постоянная величина, называемая *моментным параметром*. Значение моментного параметра выбирается из диапазона $[0; 1]$, на практике часто используется значение $\xi = 0.9$ [13].

Алгоритм послойного обучения

Как отмечалось ранее, алгоритм обратного распространения ошибки имеет ряд недостатков, среди которых зависимость от начальной инициализации синаптических связей. Это происходит из-за наличия локальных минимумов в целевой функции, и в результате не всякая попытка приводит к обучению. В данном разделе приводится альтернативный алгоритм обучения сети, проявляющий устойчивость при решении некоторых задач.

Рассмотрим алгоритм на примере нейронной сети с одним скрытым слоем, где в качестве функции активации используется гиперболический тангенс. Вводя адаптивный шаг и используя алгоритм обратного распространения ошибки, выражения для обучения нейронной сети можно представить следующим образом:

$$w_{ij}(t+1) = w_{ij}(t) - \alpha(t)\gamma_j(1 - y_j^2)y_i \tag{17}$$

$$T_j(t+1) = T_j(t) + \alpha(t)\gamma_j(1-y_j^2), \tag{18}$$

где

$$\alpha(t) = \frac{\sum_j \gamma_j^2(1-y_j^2)}{(1+\sum_i y_i^2)\sum_j \gamma_j^2(1-y_j^2)}$$

Алгоритм послойного обучения предполагает также проводить модификацию выходов нейронных элементов скрытых слоев. Такая модификация осуществляется в соответствии с методом градиентного спуска:

$$y_j(t+1) = y_j(t) - \alpha(t)\frac{\partial E}{\partial y_j(t)}$$

Здесь $\alpha(t)$ находится из метода наискорейшего спуска:

$$\alpha(t) = \min\left\{E(y_j(t) - \alpha(t)\frac{\partial E}{\partial y_j(t)})\right\}$$

Показывается [2], что оптимальное значение адаптивного шага обучения:

$$\alpha = \frac{\sum_j (y_i - t_j)\sum_i \gamma_i w_{ij}}{\sum_j (\sum_i \gamma_i w_{ij})^2}$$

Тогда выходы нейронных элементов скрытого слоя должны модифицироваться на основе выражения:

$$y_i(t+1) = y_i(t) - \alpha\gamma_i \tag{19}$$

Тогда сам алгоритм послойного обучения состоит из следующих шагов:

1. Случайная инициализация весовых коэффициентов нейронной сети и задание минимальной среднеквадратичной ошибки E_m.
2. На вход сети последовательно подаются L входных образов, и для последнего слоя весовые коэффициенты и пороги модифицируются в соответствии с (17) и (18).
3. На вход сети последовательно подаются L входных образов, и выходные значения y_i нейронов скрытого слоя модифицируются в соответствии с (19).
4. Шаги 2 и 3 повторяются до тех пор, пока суммарная среднеквадратичная ошибка сети не станет меньше, чем заданная.
5. Для L входных образов модифицируются весовые коэффициенты и пороги следующего слоя нейронной сети. При этом ошибка i-го нейронного элемента $\overline{\gamma_i} = y_i - \overline{y_i}$.
6. Процедура повторяется с шага 2, пока суммарна я среднеквадратичная ошибка сети не станет меньше заданной.

Гетерогенные нейронные сети

Рассмотрим простейшую гетерогенную сеть, состоящую из одного скрытого слоя с нелинейной функцией активации нейронных элементов и выходного линейного нейрона. Тогда выходное значение сети:

$$y = \sum_i \nu_i y_i - T, \tag{20}$$

где ν_i — i-ый весовой коэффициент выходного нейрона, y_i — выходные значения нейронных элементов скрытого слоя:

$$y_i = F(S_i) = F(\sum_l w_{li}x_l - T) \tag{21}$$

Для различных слоев нейронной сети необходимо использовать разные выражения для вычисления адаптивного шага обучения. Адаптивный шаг для выходного слоя вычисляется по формуле:

$$\alpha_1 = \frac{1}{1+\sum_i y_i^2} \tag{22}$$

Адаптивный шаг для скрытого слоя:

$$\alpha_2 = \frac{\sum_i C_i \sum_j (y_j - t_j) w_{ij}}{F_1'(0) F_2'(0) \sum_j (\sum_i C_i w_{ij})^2} \tag{23}$$

Показывается [2], что обучающие правила для выходного слоя и для скрытого слоя соответственно записываются как:

$$v_i(t+1) = v_i(t) - \alpha_1(t)(y - t)y_i \tag{24}$$

$$T(t+1) = T(t) + \alpha_1(t)(y - t) \tag{25}$$

$$w_{li}(t+1) = w_{lj}(t) - \alpha_2(t)\gamma_i F'(S_i)x_l \tag{26}$$

$$T_i(t+1) = T_i(t) + \alpha_2(t)\gamma_i F'(S_i) \tag{27}$$

Для обучения может использоваться *алгоритм многократного распространения ошибки* в связи с нестабильностью процесса обучения из-за использования различных функций активации.

Алгоритм обратного распространения ошибки предполагает для каждого тренировочного набора модификацию синаптических связей всех слоев нейронной сети. При этом изменение весовых коэффициентов одного слоя нейронной сети происходит без учета изменения остальных слоев. Это может привести к нестабильности процесса обучения, который характеризуется отсутствием тенденции к снижению среднеквадратичной ошибки сети. В гетерогенных сетях поэтому может возникнуть разсинхронизация процесса обучения между разными слоями сети. Алгоритм многократного распространения ошибки, в свою очередь, предполагает на каждой итерации модификацию синаптических связей только одного слоя нейронной сети. В соответствии с этим каждый образ будет последовательно подаваться на нейронную сеть столько раз, сколько настраиваемых слоев имеет сеть.

Пусть p — число настраиваемых слоев нейронной сети. Тогда алгоритм многократного распространения ошибки состоит из следующих шагов:

1. Задается желаемая среднеквадратичная ошибка E_m.
2. Синаптические связи инициализируются случайным образом.
3. Счетчик числа настраиваемых слоев инициализируется числом p.
4. Подается первый тренировочный набор на вход нейронной сети. Производится фаза прямого и обратного распространения сигнала. В результате осуществляется модификация весовых коэффициентов и порогов нейронных элементов только для p-го слоя нейронной сети:

$$w_{ip}(t+1) = w_{ip}(t) - \alpha_2(t)\gamma_p F_p'(S_p)y_i$$

$$T_p(t+1) = T_p(t) + \alpha(t)\gamma_p F_p'(S_p),$$

   где $i = p - 1$.
5. Счетчик декрементируется.
6. Если $p \neq 0$, перейти к шагу 4, иначе перейти к шагу 7.
7. Повторяется процесс обучения, начиная с шага 3, для всех тренированных наборов обучающей выборки.
8. Вычисляется суммарная среднеквадратичная ошибка E, и если $E > E_m$, происходит переход к шагу 3, иначе процесс обучения завершается.

Логарифмическая функция активации

Рассмотрим такой нейрон, выходное значение которого представляется в виде выражения ($S_i = \sum_l w_{li}x_l - T$):

$$y_i = \ln(S_i + \sqrt{S_i^2 + 1}) \tag{28}$$

Такой нейрон имеет логарифмическую функцию активации, возрастающую монотонно от $(-\infty; \infty)$ и имеющую точку перегиба в начале координат.

Рассмотрим гетерогенную нейронную сеть, имеющую скрытый слой нейронов с логарифмической функцией активации и один линейный выходной нейрон. Такая нейронная сеть используется для решения задач прогнозирования.

Выражения для модификации настраиваемых параметров сети можно представить в следующем виде:

$$w_{li}(t+1) = w_{li}(t) - \alpha_2(t)(y-t)y_i\nu_i x_l y_i^l$$

$$T_i(t+1) = T_i(t) + \alpha_2(t)(y-t)y_i\nu_i y_i^l$$

При этом

$$\alpha_2 = \frac{\sum_i \nu_i^2 y_i}{(1+\sum_l x_l^2)\sum_i \nu_i^2 (y_i')^2}$$

Обучение происходит согласно выведенным ранее формулам для обучения гетерогенных сетей (24), (25) и (22).

Численный эксперимент

В численном эксперименте нейронные сети с различными архитектурами и конфигурациями тестируются для выявления оптимальных конфигураций параметров и характерных особенностей поведения. Используются как синтетические данные, так и реальные:

— Прогнозирование функции
$$y = 0.1\sin(3\mathbf{x}) + 0.5 \tag{29}$$

— Прогнозирование функции
$$y = 10\sin\mathbf{x} + 5\sin(3\mathbf{x}) \tag{30}$$

— Прогнозирование функции
$$y = 10\sin\mathbf{x} + 5\sin(3\mathbf{x}) + 2\sin(30\mathbf{x}) + \sin(50\mathbf{x}) \tag{31}$$

— Данные по потреблению электроэнергии в Турции.

В ходе численного эксперимента сначала задавался размер одной выборки данных, эффективно определяющий количество входов нейронной сети, и размер части выборки, используемой для обучения. Также задавалось количество точек, которые необходимо спрогнозировать. Нейронная сеть обучалась до достижения заданной среднеквадратичной ошибки на обучающей выборке, затем производился эксперимент по прогнозированию данных на P шагов вперед и вычислялась среднеквадратичная ошибка на спрогнозированных данных. Если архитектура нейронной сети подразумевала одновременное вычисление всех P точек, то они вычислялись сразу, в противном случае вычислялась лишь одна или несколько следующих точек, они добавлялись в вектор уже известных данных и снова подавались на вход нейронной сети.

Линейная сеть с правилом обучения Видроу-Хоффа

Линейная сеть при обучении до среднеквадратичной ошибки в $E_m = 0.001$ демонстрирует хорошие результаты при прогнозировании периодических функций как на короткий промежуток времени, так и на несколько периодов. Однако, при помощи линейной сети, обучаемой по правилу Видроу-Хоффа, удалось достичь требуемой E_m только для первых двух периодических функций (29) и (30). Для функции (31) метод либо не сходился, либо за критическое число итераций, установленное в 10^5, не достигалась требуемая точность, и процесс прерывался.

Из 3 видно, что среднеквадратичная ошибка прогнозирования как функция от числа нейронов (и, то есть, от числа входных данных) имеет один или несколько явно выраженных минимумов: при недостаточном числе нейронов нейронная сеть не может точно спрогнозировать следующие точки, а при избыточном числе наступает переобучение. Также из этого графика видно, что если предсказывание осуществляется на основе тех точек, которые уже были в обучающей выборке (как в случае 25 точек, для зеленого графика), то среднеквадратичная ошибка существенно меньше, чем в обратном случае (для 20 точек, красная линия).

На графике 4 приведены результаты прогнозирования для двух нейронных сетей: с 4 и 14 входными нейронами соответственно. Видно, что прогноз нейронной сети из 4 элементов практически совпадает с эталоном, в то время как для прогноза нейронной сети с 14 входными элементами прогноз существенно

отклоняется от реальных значений. Среднеквадратичные ошибки равны 0.0452432 и 0.2324168 соответственно — то есть, для нейронной сети из 14 элементов наблюдается переобучение. Аналогичная картина наблюдается на графике ??, на котором представлены результаты прогнозирования зашумленного синуса (30) нейросетями с 5 и 15 входными нейронами.

Вид зависимости среднеквадратичной ошибки от числа итераций при прогнозировании (30) указан на ??. Заметим дополнительно, что для сети с 15 входными элементами потребовалось 2303 итерации для достижения требуемой среднеквадратичной ошибки в 0.001, а для сети с 5 нейронами — 1520 итераций.

Заметим, что данным методом не удалось предсказать (31) и реальные данные: заданная точность не достигалась за установленное предельное число итераций при достаточно малой скорости обучения α, либо метод расходился с увеличением α.

Линейная сеть с использованием псевдообратной матрицы

При использовании псевдообратной матрицы следует заботиться, чтобы $L - p - P \geqslant k$, где L — размерность выборки, p — число входов, k — зависящее от ряда целое число, например, 2 для (29), 3 для (30) и $6 - 7$ для (31), иначе метод перестает работать.

На графике 5 приведены прогнозы сетью с 8 входными нейронами на 10 и 11 точек вперед соответственно. В первом случае прогноз совпадает с реальными значениями, в то время как во втором случае наглядно демонстрируется ошибочный прогноз, возникающий из-за недостаточной размерности выборки относительно желаемого числа предсказанных точек и входных нейронов.

Графики 7 и 9 демонстрируют возможности по предсказанию реальных временных рядов. Заметим, что среднеквадратичная ошибка, деленная на число предсказанных точек, на три порядка меньше абсолютных значений, и приблизительно в 50 раз меньше локальных разбросов значений. График 11 показывает, что нейронная сеть не в состоянии предсказать случайные девиации: в реальных данных на этом участке случился небольшой провал, который не был предсказан нейронной сетью. График 13 показывает, что нейронная сеть успешно справляется с прогнозированием на число точек, сопоставимое с размерностью обучающей выборки.

Итак, метод демонстрирует хорошую точность предсказания и высокую применимость даже при недостаточной размерности входных данных относительно желаемого числа входов и размерности предсказываемых данных: достаточно, чтобы соблюдалось вышеупомянутое условие. Кроме того, при наличии эффективных методов расчета псевдообратной матрицы метод соответственным образом гораздо быстрее метода, использующего правило обучения Видроу-Хоффа. Заметим так, что этим методом удалось эффективно предсказать (31) и реальные данные, что было невозможно при обучении по правилу Видроу-Хоффа. Заметим так же, что для продолжения временного ряда из реальных данных по 5000-7000 точкам на 500 точек вперед, пришлось увеличить размер стека в Scilab до 10 миллионов слов двойной точности, а для продолжения на 2200 точек по 4000 точкам с размерностью обучающей выборки 6500 — до 50 миллионов.

Гомогенная многослойная сеть

Анализировались гомогенные нейронные сети с одним скрытым слоем и различным числом нейронов в входном, выходном и скрытом слоях, а также различными функциями активации. Для обучения во всех случаях использовался алгоритм обратного распространения ошибки.

При использовании биполярной сигмоидной функции в качестве функции активации ни в одном из случаев не удалось достигнуть среднеквадратичной ошибки в 0.001 за число итераций, не превышающее 10^5, поэтому все дальнейшие экспериментальные данные приведены для нейронной сети, использующей гиперболический тангенс как функцию активации. Заметим, что, в связи с ограниченной областью значений гиперболического тангенса, необходимо масштабировать обучающую выборку к соответствующему диапазону значений.

График 14 демонстрирует характерный вид зависимости среднеквадратичной ошибки от номера итерации.

Качество прогнозирования зависит также от числа нейронов в скрытом слое, которое определялось как $k \lfloor p + P \rfloor$, где k варьировалось в ходе эксперимента. Графики зависимости приведены на 15. По оси абсцисс отложено число нейронов во входном слое, разные линии соответствуют разным числам k, которые приведены на легенде. На 16 те же графики представлены как трехмерная поверхность для наглядности. Видно, что, начиная с $k = 2$, изменение числа нейронов в скрытом слое практически не влияет на качество прогнозирования.

Также стоит отметить, что с ростом числа нейронов во входном слое одновременно растет среднеквадратичная ошибка прогнозирования и возрастает число итераций, необходимых для достижения заданной среднеквадратичной ошибки на обучающей выборке, поэтому в реальных приложениях имеет смысл

ограничиваться меньшим числом нейронов во входном слое. Однако, при слишком малом числе входных нейронов сеть демонстрирует недостаточную обобщающую способность (17, приведены графики для сетей с 3 и 4 нейронами во входном слое).

С задачей прогнозирования функции (31) сеть справляется, в отличие от однослойной сети, обучаемой по правилу Видроу-Хоффа, но имеет существенно большую среднеквадратичную ошибку, чем однослойная же сеть, но обучаемая методом псевдообратной матрицы (18).

Заключение

В работе рассмотрены различные варианты архитектур нейронных сетей без обратной связи, протестированы на синтетических и реальных данных различные сети с различными методами обучения, проанализирована зависимость качества прогнозирования и скорости обучения от параметров сетей.

В частности:

— Наблюдается зависимость качества прогнозирования от размера окна, с одним или несколькими минимумами, и ухудшением качества с чрезмерным ростом размера окна (явление переобучения).
— Метод Видроу-Хоффа для обучения однослойной линейной нейронной сети перестает работать для достаточно сложных зависимостей.
— Метод обучения однослойной линейной нейронной сети при помощи псевдообратной матрицы позволяет быстро (за время, необходимое для вычисления псевдообратной матрицы) получить минимально возможную ошибку прогнозирования на данных.
— Сходимость алгоритма обратного распространения ошибки обеспечивается лишь при использовании гиперболического тангенса.
— Существует оптимальное значение входного числа нейронов для каждого типа временных рядов.
— Повышение сложности нейронной сети путем введения дополнительных нейронов в скрытый слой не является эффективным методом повышения качества прогнозирования для всех типов временных рядов.

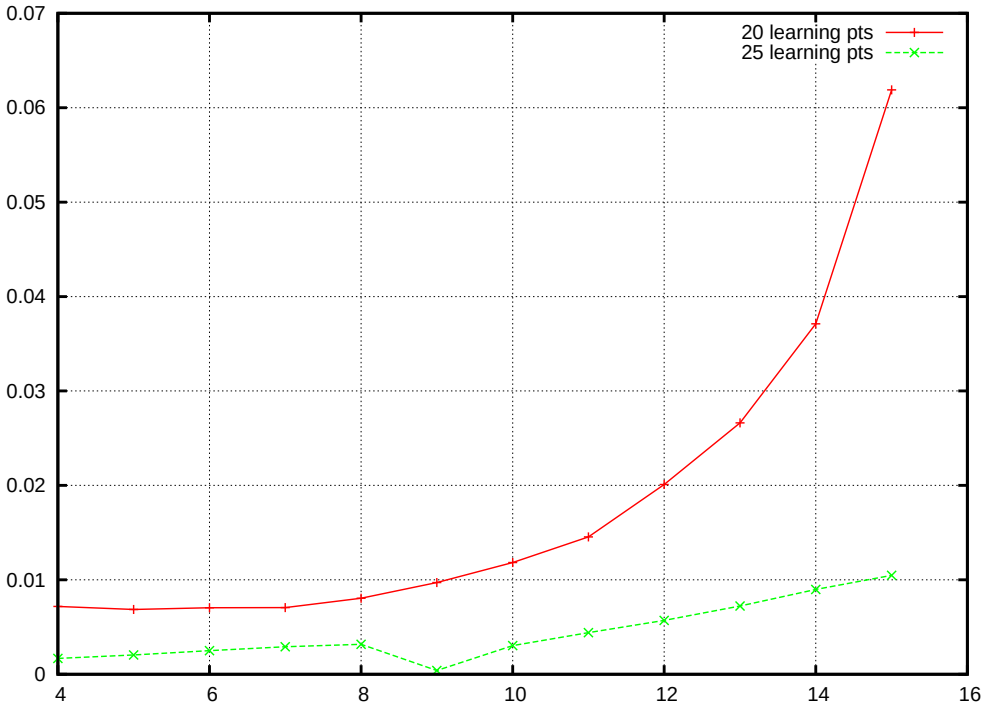


Рис. 3. Зависимость среднеквадратичной ошибки прогнозирования от числа входных нейронов при прогнозировании (29)

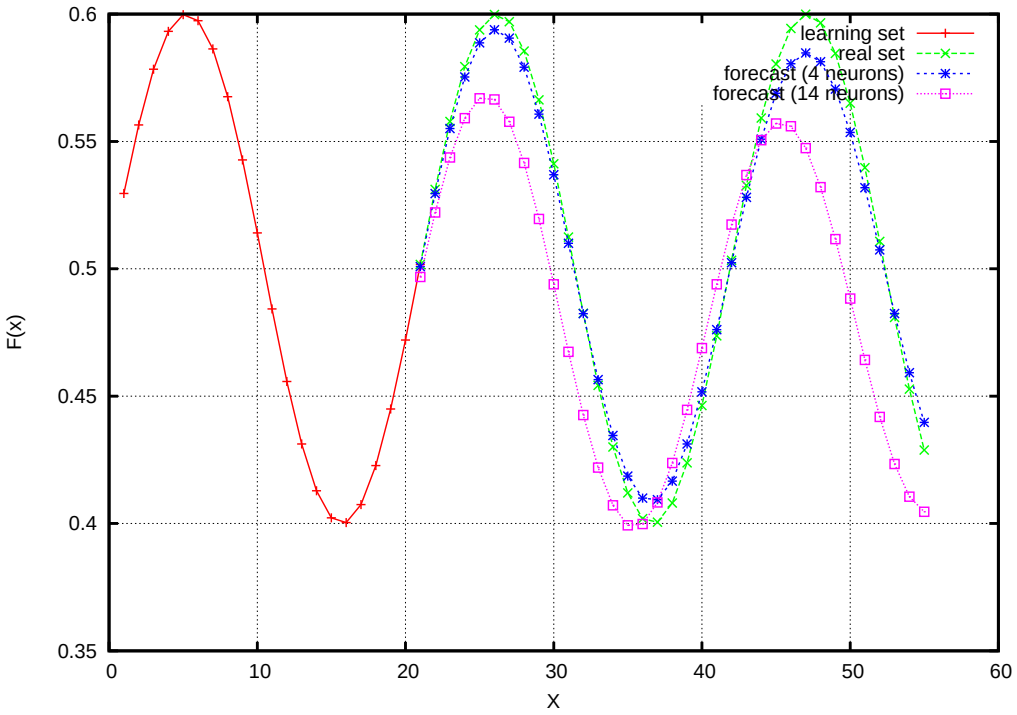


Рис. 4. Прогноз для функции (29) с 20 точками обучающей выборки: сети с 4 и 14 входными нейронами и 35 предсказанными точками, обучение по правилу Видроу-Хоффа

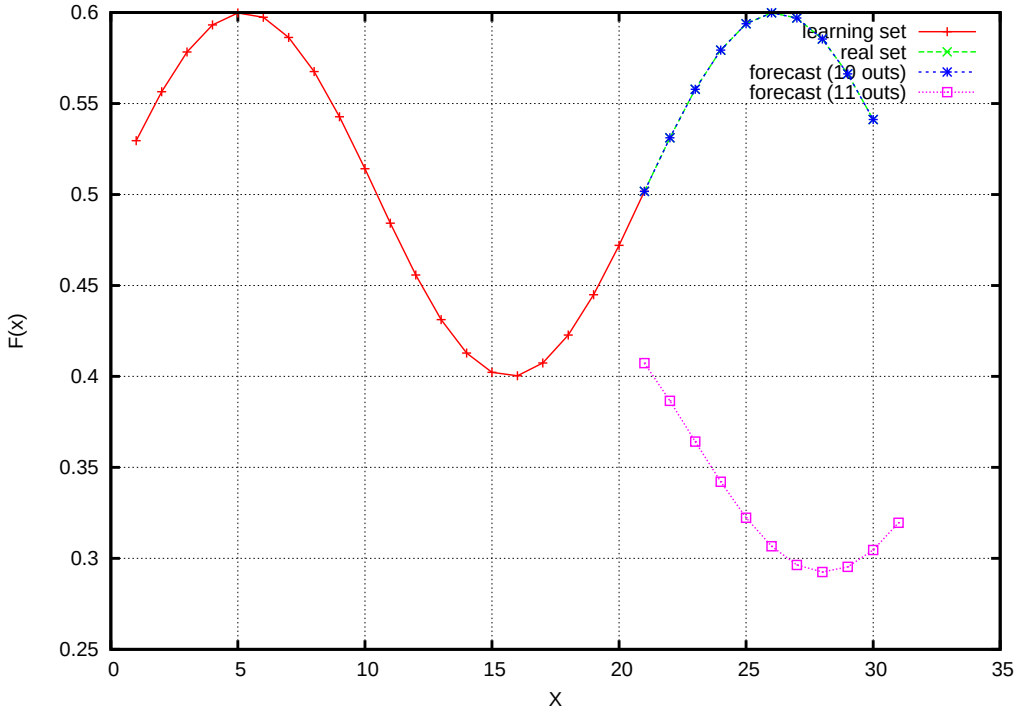


Рис. 5. Прогноз для функции (29) с 20 точками обучающей выборки: сети с 8 входными нейронами и 10 и 11 предсказанными точками, обучение методом псевдообратной матрицы

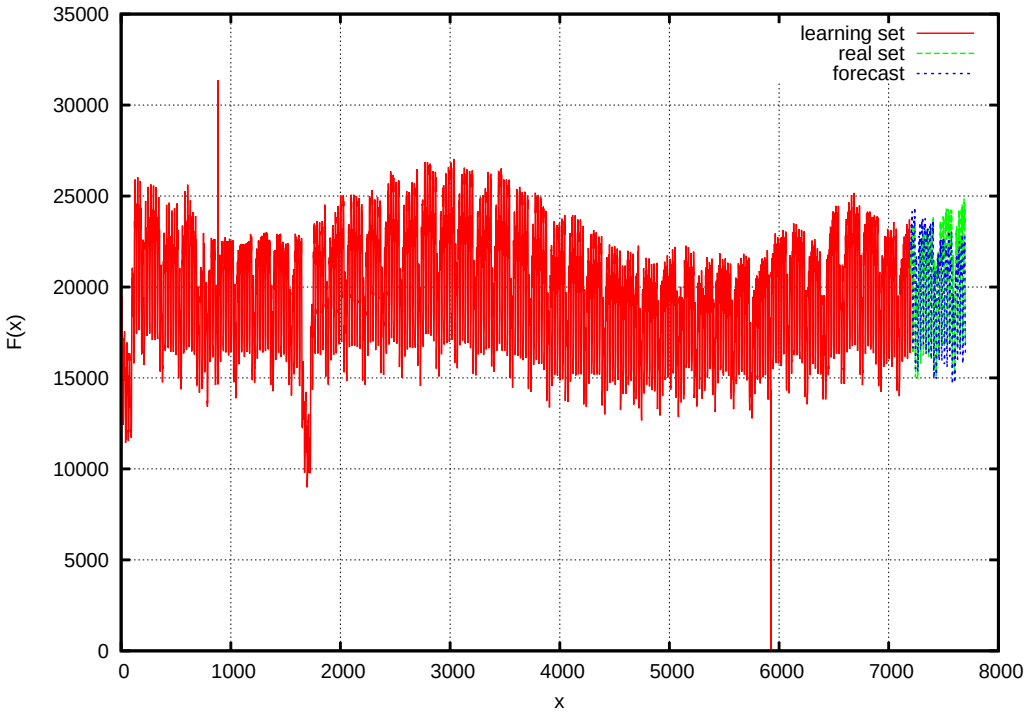


Рис. 6. Предсказанные значения для реальных данных, 7200 точек в обучающей выборке, 6600 входов, предсказание на 500 точек

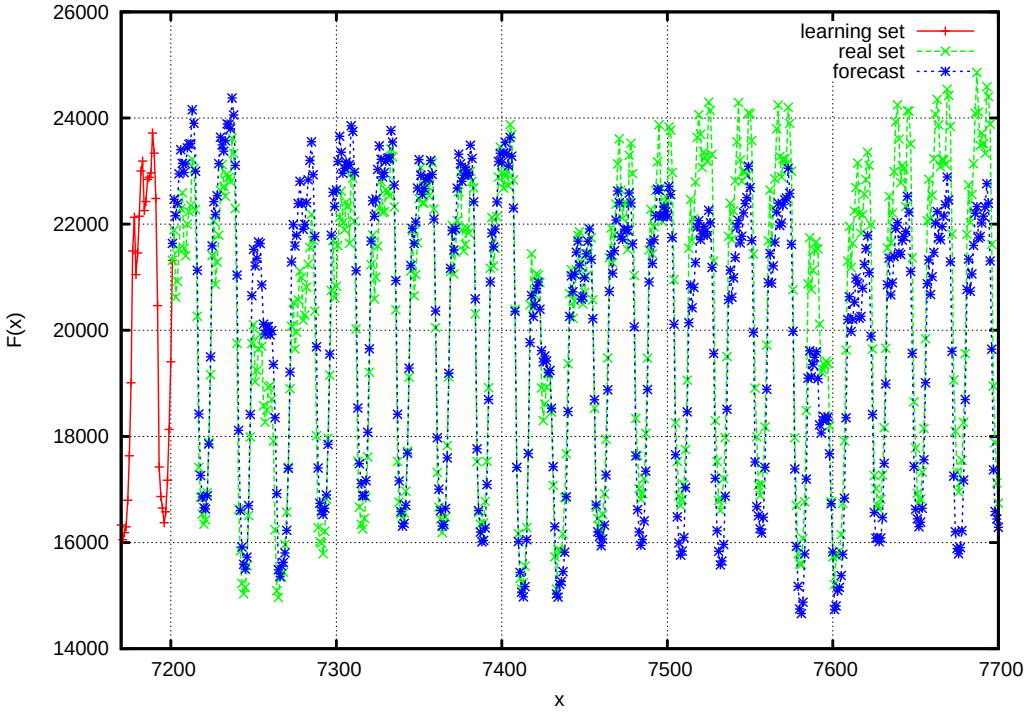


Рис. 7. Предсказанные значения для реальных данных, 7200 точек в обучающей выборке, 6600 входов, предсказание на 500 точек, увеличенная предсказанная часть

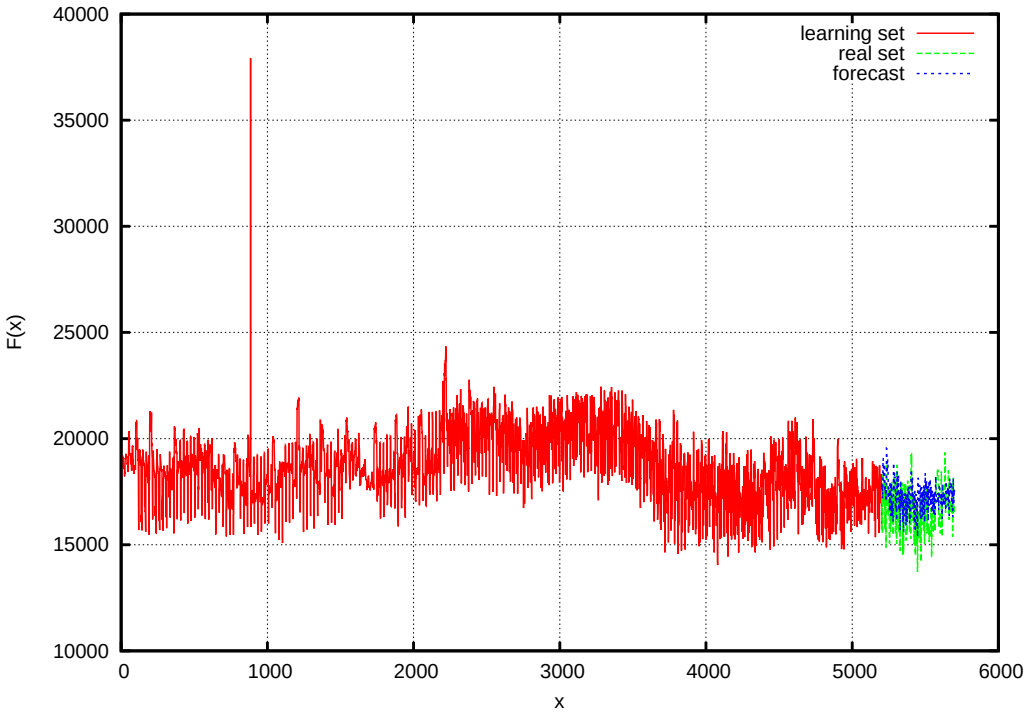


Рис. 8. Предсказанные значения для реальных данных, 5200 точек в обучающей выборке, 4600 входов, предсказание на 500 точек

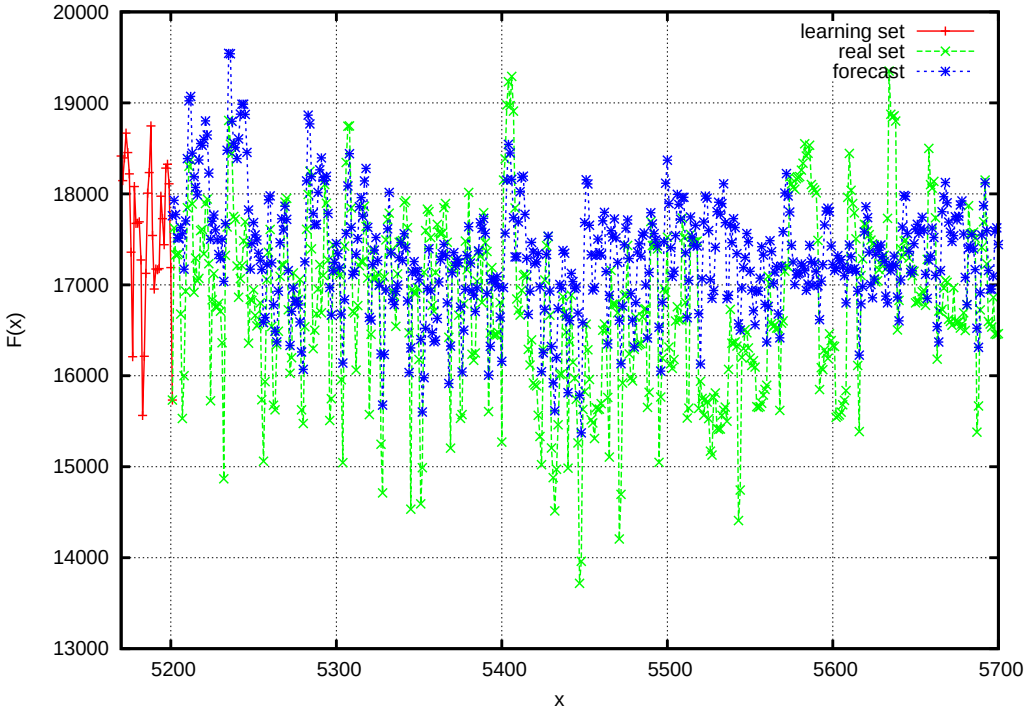


Рис. 9. Предсказанные значения для реальных данных, 5200 точек в обучающей выборке, 4600 входов, предсказание на 500 точек, увеличенная предсказанная часть

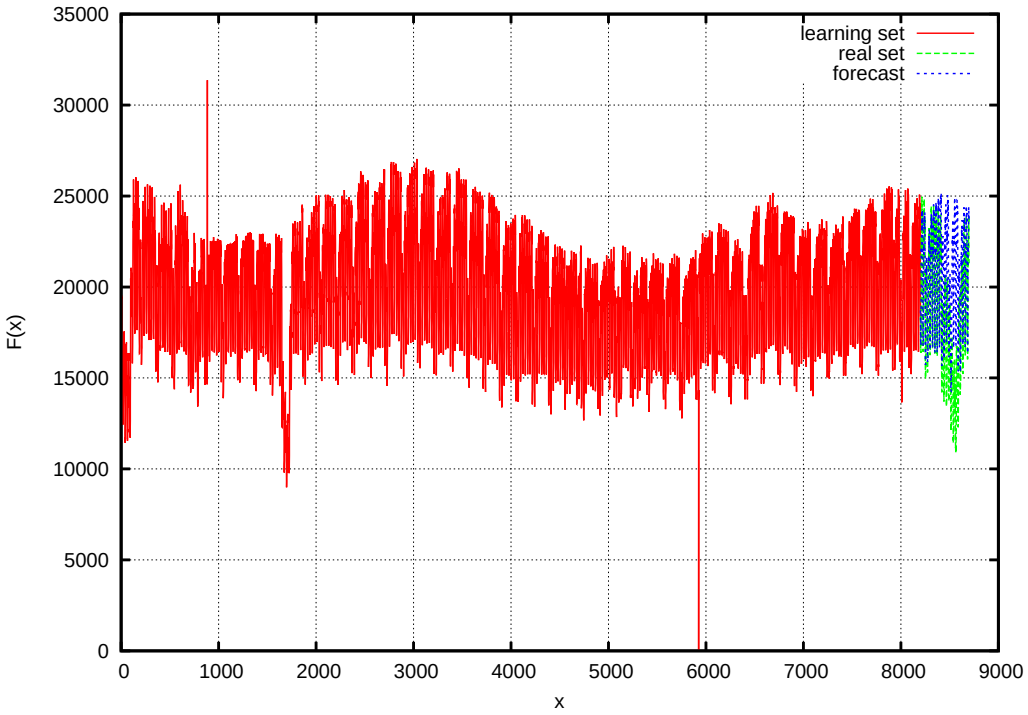


Рис. 10. Предсказанные значения для реальных данных, 8200 точек в обучающей выборке, 7600 входов, предсказание на 500 точек

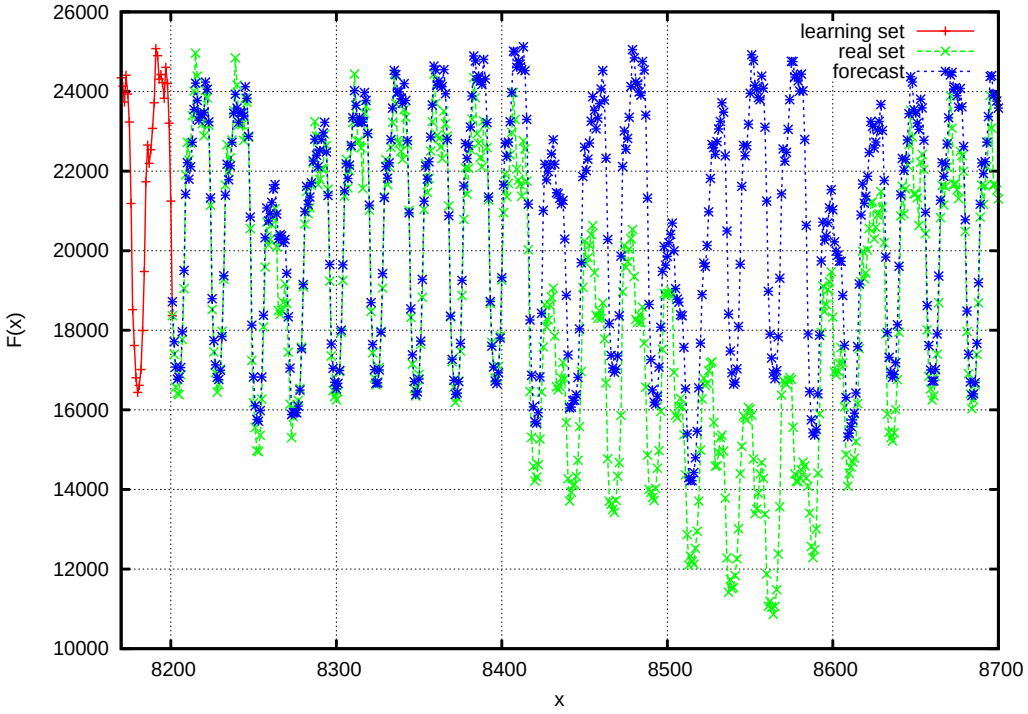


Рис. 11. Предсказанные значения для реальных данных, 8200 точек в обучающей выборке, 7600 входов, предсказание на 500 точек, увеличенная предсказанная часть

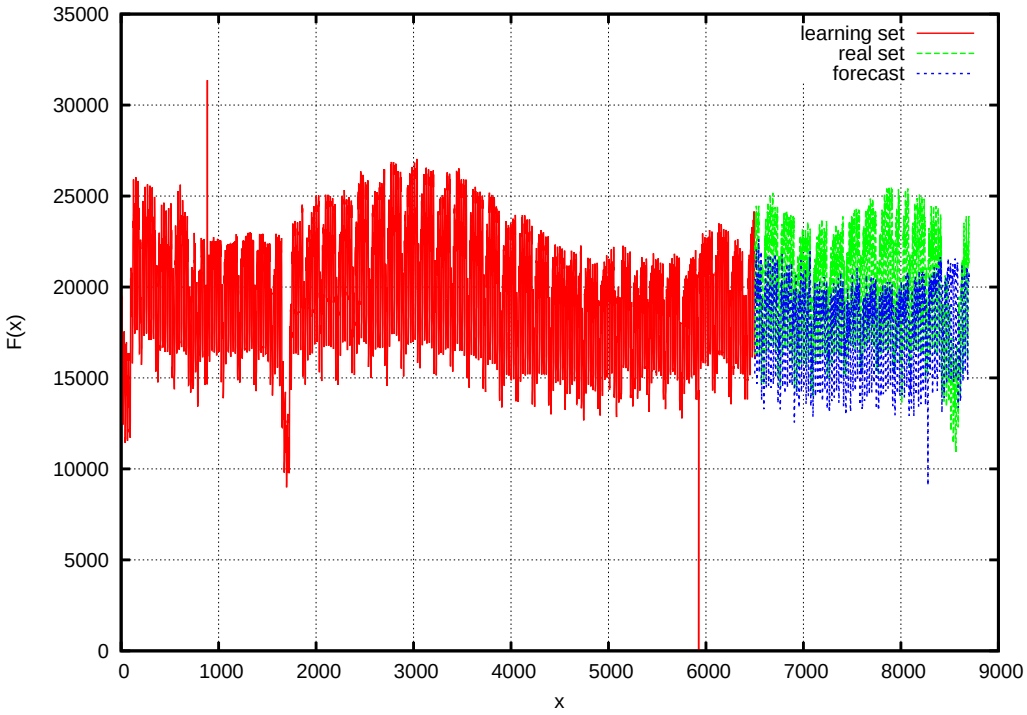


Рис. 12. Предсказанные значения для реальных данных, 6500 точек в обучающей выборке, 4000 входов, предсказание на 2200 точек

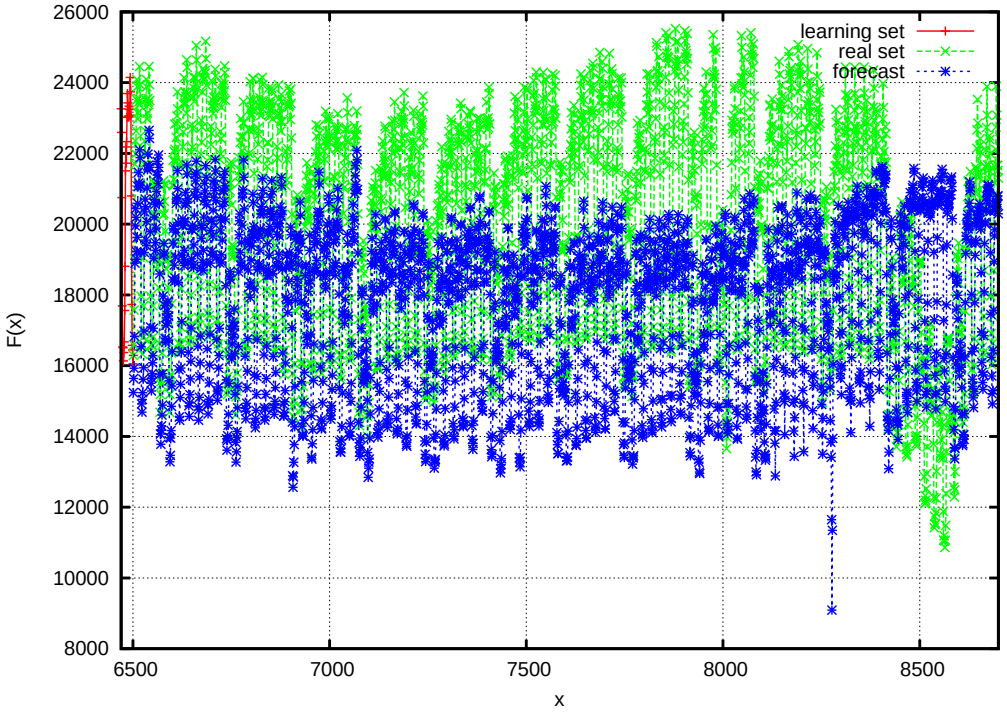


Рис. 13. Предсказанные значения для реальных данных, 6500 точек в обучающей выборке, 4000 входов, предсказание на 2200 точек, увеличенная предсказанная часть

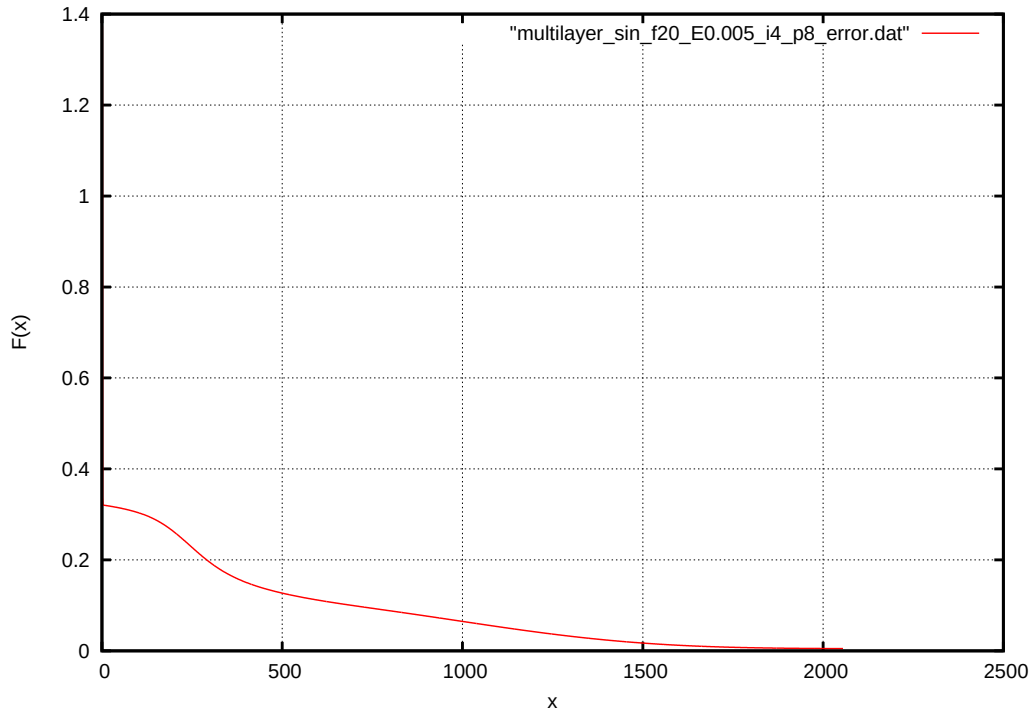


Рис. 14. График убывания ошибки на обучающей выборке для функции (29) с 20 точками обучающей выборки и 8 предсказанными точками

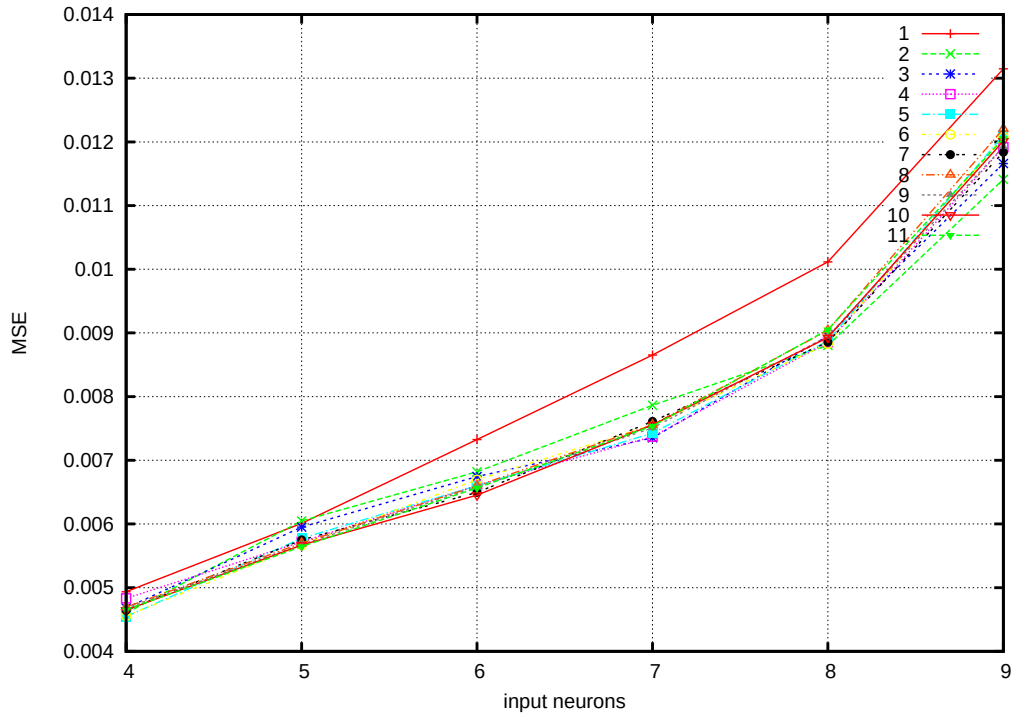


Рис. 15. Зависимость приведенной среднеквадратичной ошибки прогнозирования функции (29) от числа нейронов во входном и скрытом слоях

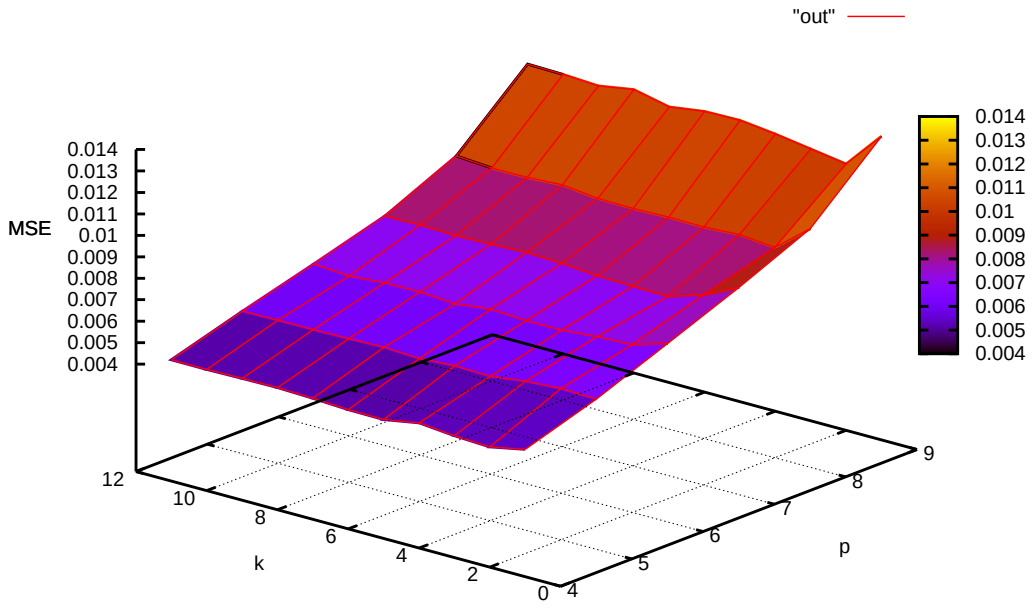


Рис. 16. Зависимость приведенной среднеквадратичной ошибки прогнозирования функции (29) от числа нейронов во входном и скрытом слоях (трехмерная визуализация)

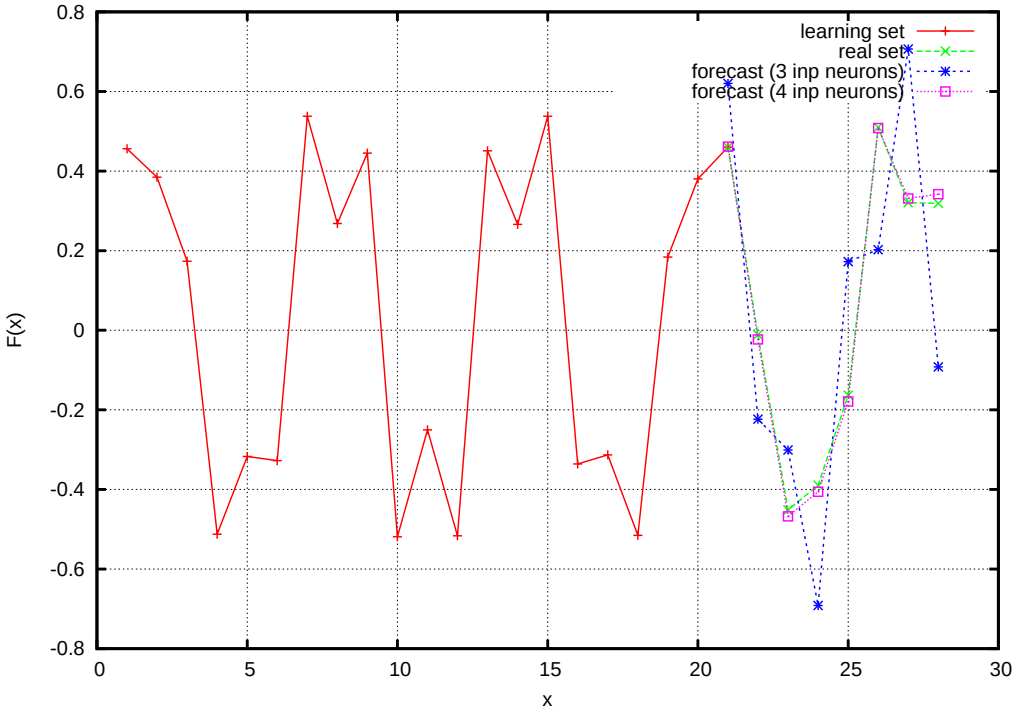


Рис. 17. Прогноз для функции (30) с 20 точками обучающей выборки: многослойные сети с 3 и 4 входными нейронами

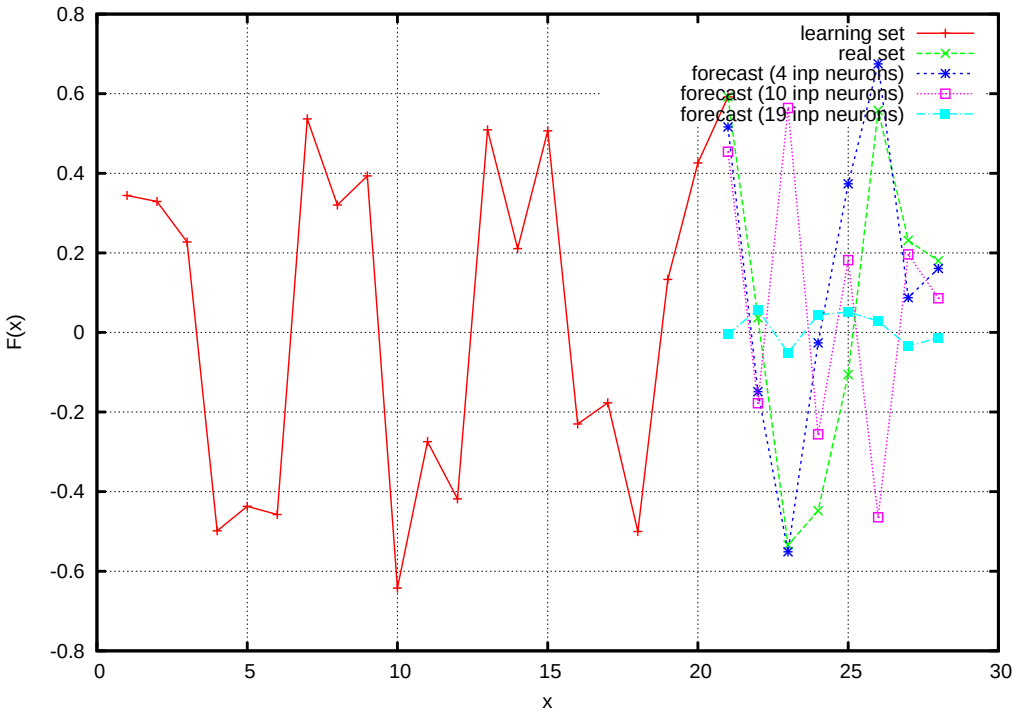


Рис. 18. Прогноз для функции (31) с 20 точками обучающей выборки: многослойные сети с 4, 10 и 19 входными нейронами

Литература

[1] В. Н.Солнцев, Д. Л. Данилов, А. А. Жиглявский. *Главные Компоненты Временных Рядов: Метод "Гусеница"*, С.-Петербургский государственный университет, 1997.

[2] В.А. Головко. Нейронные сети обучение, организация и применение. 2001.

[3] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[4] Ben J. A. Kroese and P. Patrick van der Smagt. An introduction to neural networks. 1993.

[5] B. Widrow and M. E. Hoff. Adaptive switching circuits. pages 96–104, 1960.

[6] Ф. Гантмахер. Теория матриц. 1988.

[7] Richard P. Lippmann. An introduction to computing with neural nets. *IEEE ASSP Magazine*, April 1987.

[8] А. Н. Скурихин. Нейронные сети: определения, концепции, применение. 1991.

[9] А. Н. Колмогоров. Представление непрерывных функций многих переменных суперпозицией функций одной переменной и сложением. *ДАН*, 5:953–956, 1958.

[10] Raul Rojas. Theorie der neuronalen netze: Eine systematische einfuehrung. 1996. 4., korrigierter Nachdruck.

[11] T. Maxwell, C. L. Giles, Y. C. Lee, and H. H. Chen. Nonlinear dynamics of artificial neural systems. 1986.

[12] D. Rumelhart, G. Hinton, and R. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, October 1986.

[13] J. Hertz, A. Krogh, and R. Palmer. Introduction to the theory of neural computation. 1991.

Выделение периодической компоненты из временного ряда

А. А. Токмакова
Московский физико-технический институт, ФУПМ, каф. «Интеллектуальные системы»

В проекте исследуется временной ряд на наличие периодической компоненты. На основе теории о рядах Фурье строится тригонометрическая интерполяция предложенных временных рядов методом наименьших квадратов. Также производится оценка параметров функции метода наименьших квадратов в зависимости от качества прогнозирования. В вычислительном эксперименте приводятся результаты работы корреляционной функции и метода наименьших квадратов на зашумлённом модельном синусе и реальном временном ряде электрокардиограммы.

Ключевые слова: *корреляционная функция, тригонометрическая интерполяция, метод наименьших квадратов, периодическая компонента.*

Введение

Определение 1. *Временной ряд — последовательно измеренные через некоторые (зачастую равные) промежутки времени данные.*

При прогнозировании некоторых временных рядов, например временных рядов продаж, потребления энергии или электрокардиогрммы, мы сталкиваемся с тем, что данные ряды обладают периодической компонентой. Существует несколько методов выявления периода. В данной работе рассмотрены алгоритмы атвокорреляционной функции и метода наименьших квадратов.

Автокорреляционная функция исследует временной ряд на наличие периодической компоненты, сдвигая ряд на несколько временных отсчетов и сравнивая с самим собой. Более подробно алгоритм автокорреляционной функции представлен в книгах [1, 5, 4].

Метод наименьших квадратов(МНК) вычисляет тригонометрическую аппроксимацию данного на вход ряда. Так как любая последовательность, обладающая периодичностью может быть разложена в ряд Фурье [2], необходимо принять коэффициенты перед синусами и косинусами за коэффициенты регрессии [6] и оценить их величину. Если найденная корреляция (коэффициент при определенном синусе или косинусе) велика, то можно заключить, что существует строгая периодичность на соответствующей частоте в данных.

Далее будет рассмотрена работа алгоритмов на модельных данных, а также на реальном временном ряде электрокардиограммы. Будет исследована зависимость коэффициента корреляции от различных характеристик временного ряда, а также рассмотрена возможность примененения метода наименьших квадратов для прогнозирования данных.

Постановка задачи

Дан временной ряд f_t, где n — длина временного ряда, $t \in \{1, ..., n\}$ — номер отсчета. Предполагаем, что в рассматриваемом временном ряде нет пропущенных значений, и он имеет периодические составляющие с периодом $T = \{\tau_1, ..\tau_p\}$. Работа состоит из трех следующих ступеней.

Во-первых, *тестирование на модельной задаче.* Дан зашумлённый синус с известным периодом. Необходимо исследовать изменение коэффициента корреляции в следующих ситуациях:

— при увеличении шума;
— при уменьшении числа отсчётов на период;
— при сокращении длины временного ряда.

Во-вторых, *тестирование на реальном временном ряде.* Дан временной ряд электрокардиограммы, включающий периодическую компоненту со сложным строением. Необходимо исследовать его на наличие временных периодичностей, используя алгоритмы автокорреляционной функции и МНК.

В-третьих, необходимо выяснить пригодность метода наименьших квадратов для *прогнозирования временных рядов.*

Для контроля качества алгоритма прогноза будем выделять во временном ряде l последовательных значений (контрольную выборку), которые алгоритм будет прогнозировать по предыдущим значениям. В качестве критерия качества прогноза будем минимизировать следующий функционал:

$$Q = \sum_{t=1}^{l} |\hat{f}_t - f_t|,$$

где $\hat{f}_t$ — прогнозируемое значение в t-ый момент времени, f_t — фактическое значение.

Пути решения задачи

Опишем используемые в работе алгоритмы.

Автокорреляционная функция

Определение 2. *Автокорреляционная функция — это характеристика временного ряда, которая помогает находить его повторяющиеся участки, скрытые из-за наложений шума или других помех.*

Для дискретного временного ряда $X_1, X_2, ..., X_n$ с известными матожиданием μ и дисперсией σ автокорреляционную функцию можно рассчитать по следующей формуле:

$$R(w) = \frac{1}{(n-w)\sigma^2} \sum_{t=1}^{n-w} [X_t - \mu][X_{t+w} - \mu],$$

где n —длина временного ряда, w —текущая задержка во времени. Таким образом получим функцию $R(w)$, зависящую от лагов (задержек во времени). Исследуя ее на экстремальные значения, получим искомые значения периодов $T = \{\tau_1, ..\tau_p\}$.

Оценка периода осложняется тем, что в некоторой окрестности оцениваемого периода, наблюдаются локальные максимумы коэффициентов корреляции. Следовательно, необходимо усреднение коэффициентов корреляции, а также удаление "близких"и кратных периодов ("близкими"в работе считаются периоды, отличающиеся друг от друга менее, чем на величину δ).

Ряд Фурье

Сделаем предположение о наличии периодики в предлагаемом ряде и обратимся к теореме [2].

Теорема 1. *Если некоторая периодическая функция с периодом $2j$ на интервале $[-j, j]$ удовлетворяет условиям Дирихле (имеет конечное число экстремумов и точек разрыва I рода), то она может быть представлена в виде суммы ряда Фурье (разложена в ряд Фурье).*

Таким образом, рассматриваемые в данной работе временные ряды могут быть представлены в виде бесконечного ряда Фурье. Построим регрессионную модель следующего вида [7].

Можно заметить, что разложение временной последовательности в ряд Фурье позволяет отыскать скрытые периодичности. Одним из возможных способов определения автокорреляционной зависимости является разложение временного ряда в функции синусов и косинусов и нахождение линейной множественной регрессии [6].

$$X_t = \frac{a_0}{2} + \sum_{k=1}^{n} a_k \cos(\lambda_k t) + b_k \sin(\lambda_k t),$$

где $\lambda_k = 2\pi\eta_k$, $\eta_k = \frac{k}{n}$, $k = 1, 2, .., n$.

Коэффициенты a_k и b_k определяются следующими рядами:

$$a_k = \frac{2}{n} \sum_{i=1}^{n} X_i \cos(\lambda_k t), k = 0, 1, 2.., n;$$

$$b_k = \frac{2}{n} \sum_{i=1}^{n} X_i \sin(\lambda_k t), k = 1, 2.., n.$$

Коэффициенты при косинусах и синусах — это коэффициенты регрессии. Они показывают степень, с которой соответствующие функции коррелируют с данными. Необходимо заметить, что сами синусы и косинусы на различных частотах ортогональны. Будем рассматривать не более чем n различных синусов и косинусов.

В итоге определяется корреляция функций синусов и косинусов различной частоты с наблюдаемыми данными. Если найденная корреляция (коэффициент при определенном синусе или косинусе) велика, то можно заключить, что существует строгая периодичность на соответствующей частоте в данных.

Данный метод даёт точный результат только тогда, когда длина временного ряда (то есть параметр n) кратен искомому периоду. В противном случае мы получим некую суперпозицию синусов и косинусов, которую достаточно сложно интерпретировать. Поэтому воспользуемся методом тригонометрической интерполяции с помощью метода наименьших квадратов.

Тригонометрическая интерполяция методом наименьших квадратов

Требуется построить кривую, которая воспроизводила бы график исходной экспериментальной закономерности, то есть была бы максимально близка к экспериментальным точкам, но в то же время была бы нечувствительна к случайным отклонениям измеряемой величины.

Введем непрерывную функцию $\varphi(x)$ для аппроксимации дискретной зависимости $g(x_i)$, $i = 1, ..., n$. Будем считать, что $\varphi(x)$ построена при условии наилучшего квадратичного приближения, если:

$$Q = \sum_{i=1}^{n} (\varphi(x_i) - g(x_i))^2 = \min. \tag{1}$$

Рассмотрим случай линейной аппроксимации:

$$\varphi(x) = c_0\varphi_0(x) + c_1\varphi_1(x) + ... + c_m\varphi_m(x),$$

где $\varphi_0, ..., \varphi_m$ — произвольные базисные функции, $c_0, ..., c_m$ — неизвестные коэффициенты. Количество базисных функций должно быть меньше количества заданных точек для того, чтобы их суперпозиция определялась единственным образом.

Для решения задачи линейной аппроксимации в общем случае следует найти условия минимума суммы квадратов отклонений (1). Это можно свести к задаче поиска корня системы уравнений $\frac{\partial Q}{\partial c_k} = 0$, $k = 1, ..., m$. Вычисление данных производных, при учёте равенства (1) приведёт к следующей системе алгебраических уравнений:

$$\begin{cases} \sum_{i=1}^{n} (c_0\varphi_0(x) + c_1\varphi_1)(x) + \ldots + c_m\varphi_m(x) - f_i)\varphi_0(x) = 0; \\ \sum_{i=1}^{n} (c_0\varphi_0(x) + c_1\varphi_1)(x) + \ldots + c_m\varphi_m(x) - f_i)\varphi_1(x) = 0; \\ \cdots \\ \sum_{i=1}^{n} (c_0\varphi_0(x) + c_1\varphi_1)(x) + \ldots + c_m\varphi_m(x) - f_i)\varphi_m(x) = 0. \end{cases} \tag{2}$$

Далее следует решить полученную СЛАУ относительно коэффициентов $c_0, ..., c_m$. Для решения СЛАУ обычно составляется расширенная матрица коэффициентов, которую называют матрицей Грама, элементами которой являются скалярные произведения базисных функций и столбец свободных коэффициентов:

$$\begin{pmatrix} (\varphi_0, \varphi_0) & (\varphi_0, \varphi_1) & \ldots & (\varphi_0, \varphi_m) & (\varphi_0, f) \\ (\varphi_1, \varphi_0) & (\varphi_1, \varphi_1) & \ldots & (\varphi_1, \varphi_m) & (\varphi_1, f) \\ \ldots & \ldots & \ddots & \ldots & \ldots \\ (\varphi_m, \varphi_0) & (\varphi_m, \varphi_1) & \ldots & (\varphi_m, \varphi_m) & (\varphi_m, f) \end{pmatrix}.$$

$$(\varphi_j, \varphi_k) = \sum_{i=1}^{n} \varphi_j(x_i)\varphi_k(x_i)(\varphi_j f) = \sum_{i=1}^{n} \varphi_j(x_i) f(x_i),$$

где $j = 0, ..., m$, $k = 0, ..., m$.

После того как с помощью, например, метода Гаусса найдены коэффициенты $c_0, ..., c_m$, можно построить аппроксимирующую кривую или вычислить координаты заданной точки. Таким образом, задача аппроксимации решена.

Для того чтобы сформировать ортонормированный базис, коэффициенты нормировки будут следующими: для a_0 — $(4/n)^{0.5}$; для a_k и b_k — $(2/n)^{0.5}$, где a_k — коэффициенты нормировки для косинусов, $k = 0, ..., m$; b_k — коэффициенты нормировки для синусов, $k = 1, ..., m$; n — количество отсчетов.

При работе с временными рядами необходимо учесть возможное наличие тренда или присутствие постоянного слагаемого. Обе эти составляющие исключим из данных, поскольку они могут привести к большим погрешностям при подсчёте функционала $Q = \sum_{t=1}^{l} |\hat{f}_t - f_t|$. Пользуясь методом наименьших квадратов мы учтём и тренд и наличие постоянного слагаемого. Необходимо заметить, что тригонометрическая интерполяция основана на разложении в ряд Фурье, поэтому она также не годится для выявления периодической компоненты ряда. В данной работе она используется для нахождения тренда, содержащего периодическую компоненту. Оставшиеся точки исследуются при помощи автокорреляционной функцией.

Вычислительный эксперимент

Исследование модельного зашумленного синуса В качестве модельных данных будем использовать функцию $f(t) = 0.3\sin(2\pi t/7)$. Количество отсчетов $n = 100$. При исследовании временных рядов автокорреляционной функцией для поиска "близких"периодов будем считать $\delta = 1\%$.

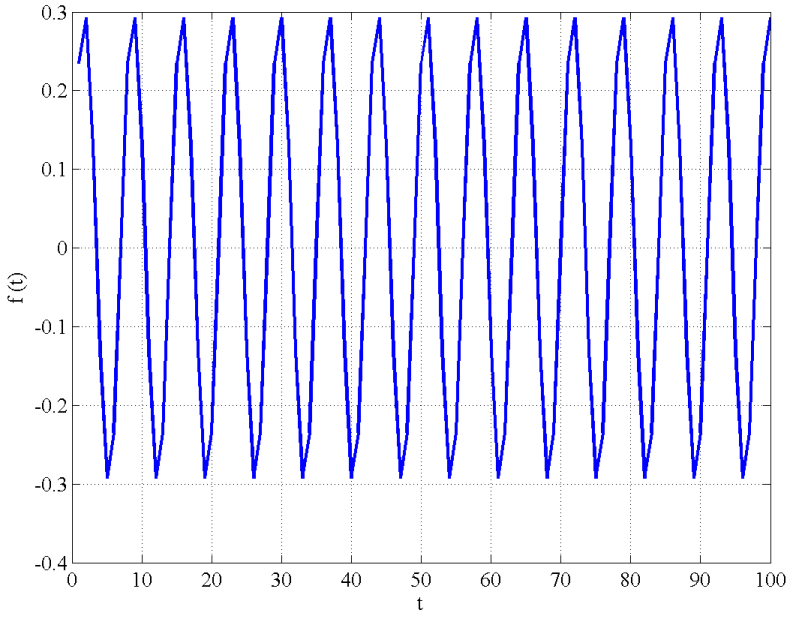


Рис. 1. Модельный временной ряд $f(t) = 0.3\sin(2\pi t/7)$

Исследование величины коэффициента корреляции в зависимости от накладываемого шума. Была определена зависимость коэффициента корреляции в зависимости от шума (величина шума определялась в процентах от максимального значения функции). Распределение шума — равномерное. Результаты представлены на рис. 2.

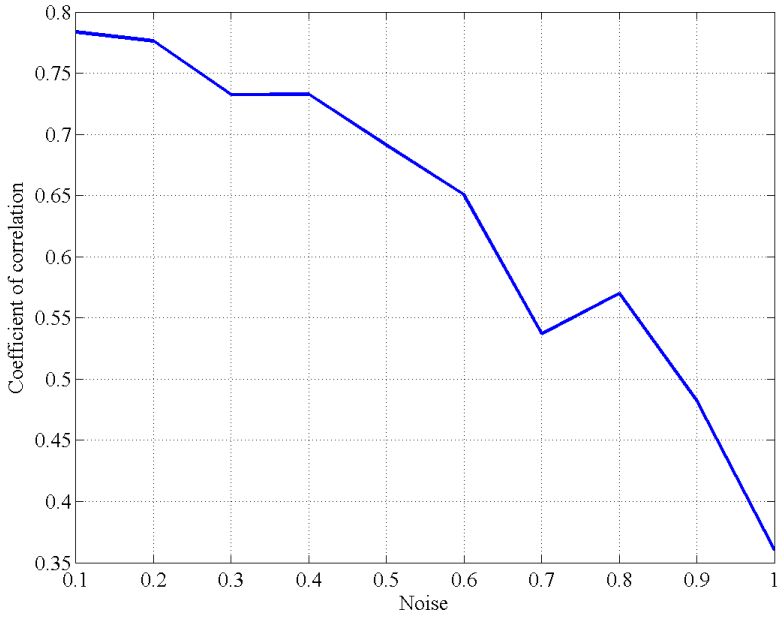


Рис. 2. Зависимость коэффициента корреляции от шума

Максимальное значение функции $f_{max} = 0.2925$. Максимальный коэффициент корреляции $R_{max} = 0.9234$ соответствует 10%-ому зашумлению. Минимальный коэффициент корреляции $R_{min} = 0.6499$ соответствует 100%-ому зашумлению.

Исследование величины коэффициента корреляции при уменьшении числа отсчетов за период. Были вычислена зависимость коэффициента корреляции от уменьшения количества отсчетов за период. Результаты представлены на рис. 3.

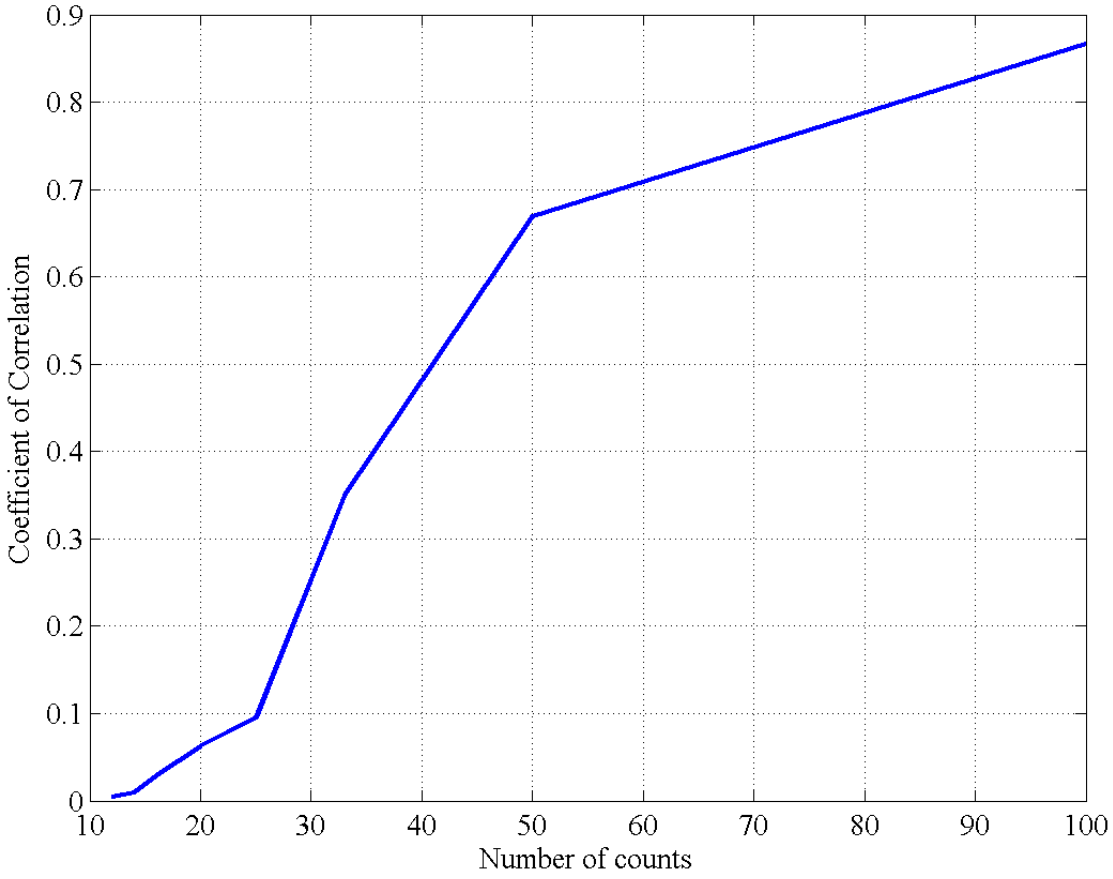


Рис. 3. Зависимость коэффициента корреляции от количества отсчетов за период

Максимальный коэффициент корреляции $R_{max} = 0.9295$ соответствует 100-ти отсчетам за период. Минимальный коэффициент корреляции $R_{min} = 0.4355$ соответствует 7-ми отсчетам за период.

Исследование величины коэффициента корреляции при сокращении временного ряда. Расчет был произведен начиная с длины ряда $n = 100$ до $n = 10$. Величина шага составляла 10 временных отсчетов.

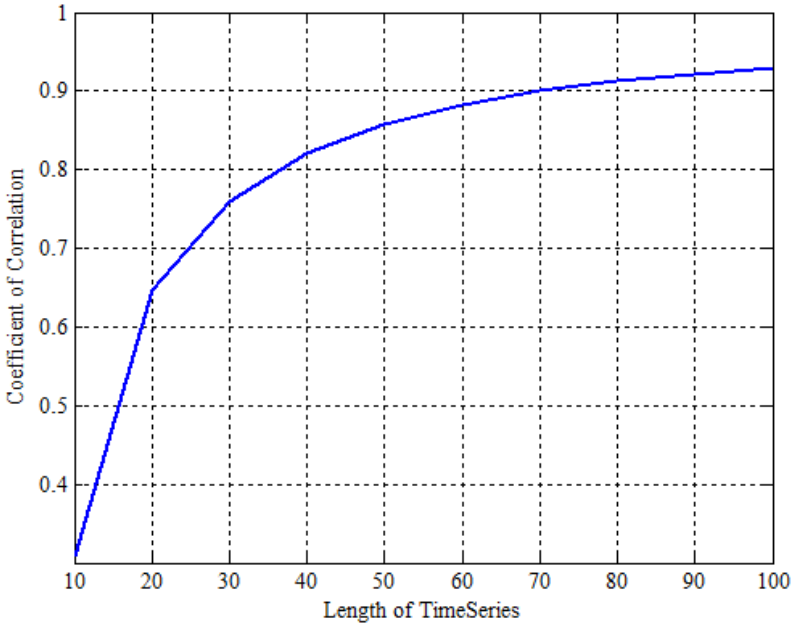


Рис. 4. Зависимость коэффициента корреляции от длины временного ряда

Максимальный коэффициент корреляции $R_{max} = 0.9295$ соответствует $n = 100$. Минимальный коэффициент корреляции $R_{min} = 0.3085$ соответствует $n = 10$, где n — длина ряда.

Вывод:

1. при увеличении шума коэффициент корреляции уменьшается;
2. при уменьшении количества отсчётов за период коэффициент корреляции уменьшается;
3. при уменьшении длины временного ряда коэффициент корреляции уменьшается.

Данные выводы напрямую следуют из устройства корреляционной функции.

Исследование реального временного ряда электрокардиограммы

Рассмотрим реальный временной ряд сложной периодики. Количество отсчетов $n = 4170$.

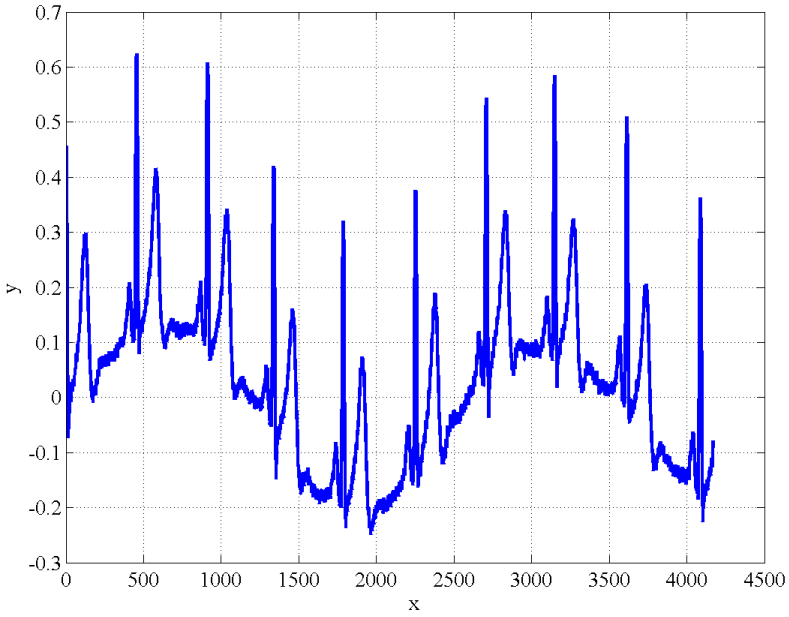


Рис. 5. Временной ряд электрокардиограммы

После применения корреляционной функции к первоначальному ряду получим следующие коэффициенты корреляции:

Корреляция	0.6	0.6	0.4	0.4	0.4	0.3	0.2	0.1	0.1
Период	43	121	453	163	205	288	2371	2089	2811

При анализе таблицы и графика получим, что корреляционная функция определила "локальный"период временного ряда, тогда как глобальная периодика осталась не выявленной. Стоит заметить, что локальный период также определён не точно и коэффициенты корреляции очень малы.

Применим метод наимеьнших квадратов для получения тригонометрического тренда данного временного ряда. Аппроксимация происходит с помощью двух гармоник.

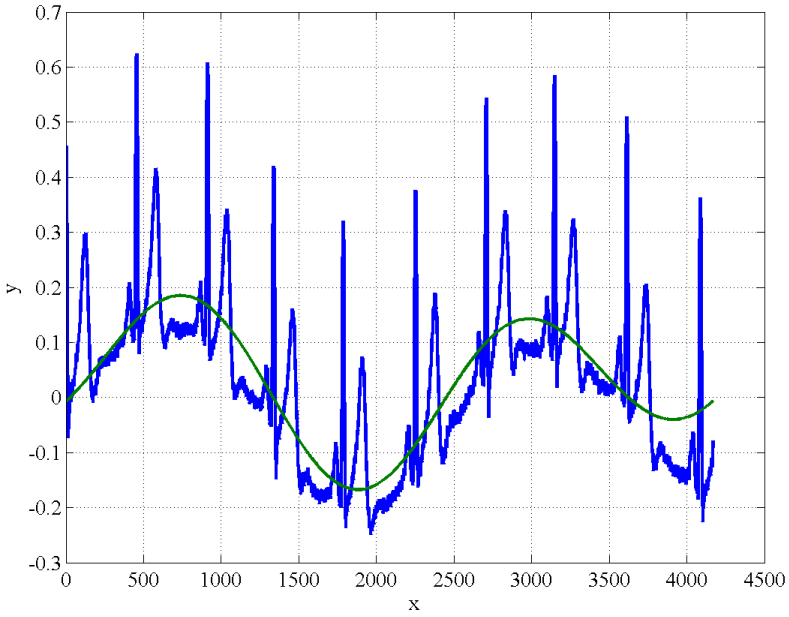


Рис. 6. Выделение тригонометрического тренда

Для выявления скрытой периодики необходимо вычесть из исходного ряда его тригонометрическую интерполяцию.

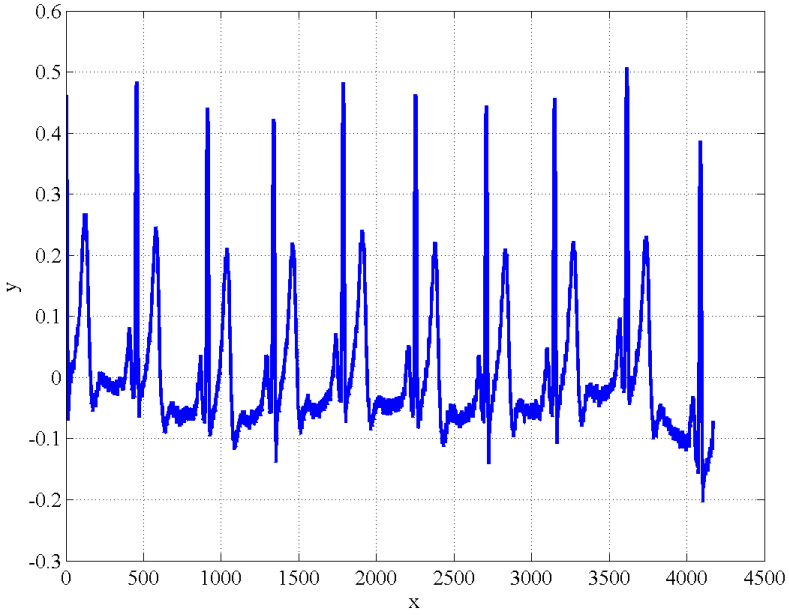


Рис. 7. Ряд электрокардиограммы с исключенным тригонометрическим трендом

После обработки полученного временного ряда автокорреляционной функцией получим следующие скрытые периодики и коэффициенты корреляции:

Корреляция	0.5	0.2	0.2	0.1	0.1
Период	459	2700	3160	1320	868

Заметим, что хотя коэффициенты корреляции малы, периодика ряда определена верно. Малость коэффициентов объясняется сложным строением ряда.

Исследование применения МНК для прогнозирования реальных временных рядов

Метод наименьших квадратов при тригонометрической интерполяции основывается на добавлении новых гармоник для лучшего совпадения реальной функции и её аппроксимации. Критерием схожести служил функционал:

$$Q = \sum_{t=1}^{l} |\hat{f}_t - f_t|,$$

где $\hat{f}_t$ — прогнозируемое значение в t-ый момент времени, f_t — фактическое значение.

В данной работе расчет велся начиная с двух гармоник. Остановка алгоритма происходила в двух случаях:

— номер гармоники больше половины длины ряда;
— относительная ошибка на один отсчёт составляет менее 1%.

Результат работы алгоритма для реального временного ряда приведён на рис. 8.

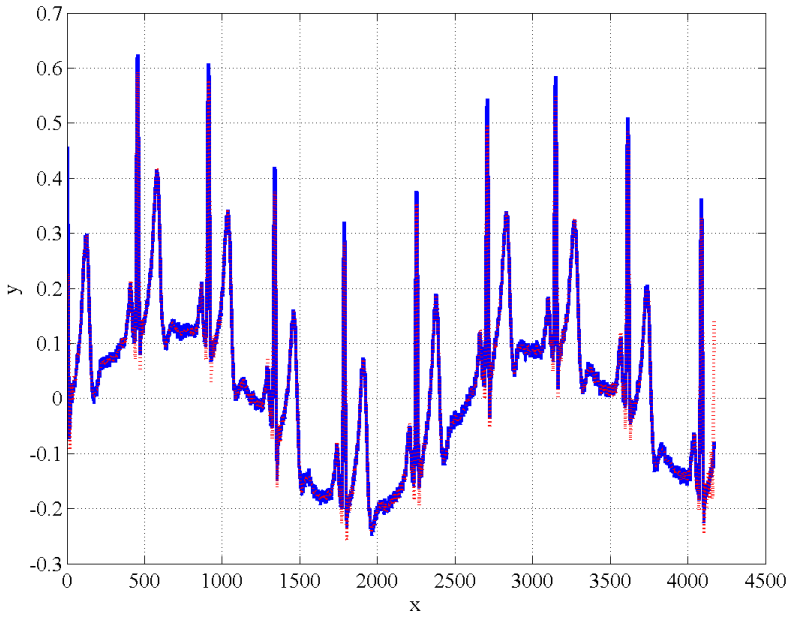


Рис. 8. Тригонометрическая аппроксимация временного ряда

Результат прогнозирования ряда приведён на рис. 9.

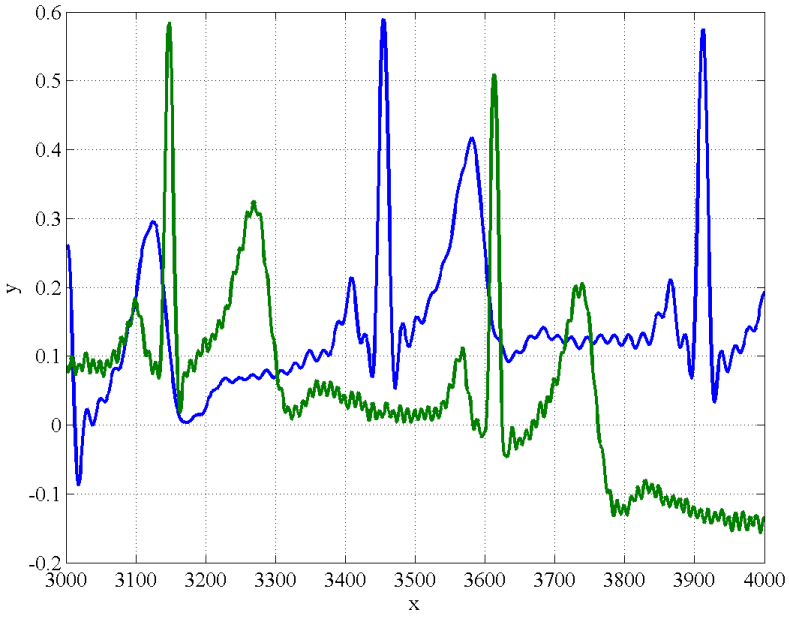


Рис. 9. Реальный и спрогнозированный временной ряд

Длина входного ряда 3000. Прогноз на 1000 точек. Остановка произошла на 148 гармонике. Функционал качества составил $Q = 163.6817$ на 1000 временных отсчётов, что составляет 16.3%.

Плохие результаты объясняются тем, что метод наименьших квадратов берёт за период количество отсчётов равное длине ряда. Следовательно, при прогнозировании происходит копирование первоначальных точек.

Прогнозировать методом наименьших квадратов можно только ряды, которые обладают строгой периодичностью. Однако, метод подходит для интерполяции временного ряда внутри отрезка (создание непрерывных рядов), а также для аналитического описания (представления в виде ряда Фурье) рядов, обладающих периодической компонентой.

Заключение

В работе рассмотрена зависимость коэффициента корреляции от различных входных параметров. Исследованы способы получения периодичности временных рядов, а также метод нахождения скрытых периодичностей. Рассмотрена возможность применения метода наименьших квадратов для прогнозирования временного ряда с периодической компонентой. Анализ проводился на модельных данных и на реальном временном ряде электрокардиограммы.

Необходимый для повторения вычислительного эксперимента код можно найти на сайте:
`https://mlalgorithms.svn.sourceforge.net/svnroot/mlalgorithms/PeriodicComponents/`

Литература

[1] D. Gujarati. *Basic Econometrics, 4th ed*, The McGraw-Hill Companies, 2004.

[2] А.М. Тер-Крикоров, М.И. Шабунин. *Курс математического анализа*, ФИЗМАТ-ЛИТ, 2001.

[3] В.В. Стрижов, Г.О. Пташко. *Алгоритмы поиска суперпозиций при выборе оптимальных регрессинных моделей*, Вычислительный центр им. А.А. Дородницына РАН, 2007.

[4] В.В. Витязев. *Спектрально-корреляционный анализ равномерных временных рядов*, С.-Петербургский государственный университет, 2001.

[5] А.И. Орлов Прикладная статистика, «Экзамен», 2004, №3: 259-279.

[6] Ю.В. Попов *О выделении периодической компоненты из временного ряда показателя количества катастроф*, "Проблемы безопасности полетов 2008.

[7] В.В. Стрижов *Методы индуктивного порождения регрессионных моделей*, Вычислительный центр им. А.А. Дородницына Российской Академии наук, 2008, 37.

Использование теста Гренджера при прогнозировании временных рядов

А. П. Мотренко
Московский физико-технический институт, ФУПМ, каф. «Интеллектуальные системы»

Работа посвящена исследованию возможностей применения теста Гренджера в прогнозировании временных рядов. В основе теста Гренджера лежат статистические тесты и использование линейных регрессионных моделей. Исследуется зависимость качества прогноза от порядка модели, способа обработки данных. В вычислительном эксперименте приводятся результаты работы алгоритма на различных временных рядах: стационарных, нестационарных, с обратной связью, независимых по Гренджеру.
Ключевые слова: *тест Гренджера, casual connectivity, выбор порядка регрессионной модели.*

Введение

Назовем временным рядом последовательность значений некоторой величины $\mathbf{x}(t)$, измеренной через равные промежутки времени. Основываясь на этих данных, а также исследуя другие временные ряды, можно спрогнозировать значения ряда $\mathbf{x}(t)$ в будущем. В частности, возникают ситуации, когда история других рядов лучше помогает сделать прогноз, чем история самого $\mathbf{x}(t)$. Определить, существует ли зависимость такого рода, дает возможность тест Гренджера – статистический метод, предложенный Клайвом Гренджером (Clive Granger) в 60-х годах [1].

Постановка задачи

Использование теста Гренджера подразумевает прогнозирование с помощью линейных регрессионных моделей. Пусть $\mathbf{x}_1(t)$ и $\mathbf{x}_2(t)$ — исследуемые временные ряды, тогда пример такой модели

$$\mathbf{x}_1(t) = \sum_{j=1}^{p} a_{11}(j)\mathbf{x}_1(t-j) + \sum_{j=1}^{p} a_{12}(j)\mathbf{x}_2(t-j) + E_1(t),$$

$$\mathbf{x}_2(t) = \sum_{j=1}^{p} a_{21}(j)\mathbf{x}_1(t-j) + \sum_{j=1}^{p} a_{22}(j)\mathbf{x}_2(t-j) + E_2(t).$$

Здесь p – количество предыдущих значений, принимаемых во внимание, матрица $A(j)$ с коэффицентами $a_{ik}(j)$ содержит веса узлов, а $E_1(t)$ и $E_1(t)$ – ошибки прогнозирования. Будем считать, что $\mathbf{x}_1$ следует из $\mathbf{x}_2$, если ошибка прогнозирования $\mathbf{x}_1(t)$ уменьшается при включении в модель значений ряда $\mathbf{x}_2$ (то есть если коэфты $A_{12,j}$ заметно отличаются от нуля). Результат чувсвителен к изменению параметра p; эта зависимость исследуется в работе.

Тест Гренджера применим к рядам, обладающим постоянными (не зависящими от времени) матожиданием и дисперсией. Если исследуемый ряд не обладает этими свойствами, необходимо привести его к соответсвующему виду. Предполагается провести исследование зависимости результата от способа преобразования данных.

Для оценки качества прогнозирования выделим контрольную выборку длины m и построим функционал

$$Q = \sum_{i=1}^{m} (\tilde{\mathbf{x}}(i) - \mathbf{x}(i))^2,$$

где $\tilde{\mathbf{x}(i)}$ — спрогнозированное значение элемента выборки, $\mathbf{x}(i)$ — его истинное значение.

Пути решения

Подготовка данных

Прежде всего, необходимо удостовериться, что ряды стационарны. Для оценки этого свойства можно использовать ADF тест [2], основанный на нулевой гипотезе об отсутствии стационарности или, дополнительно к ADF, KPSS тест [3], предполагающий ее наличие. Если проверка дала отрицательный результат, необходимо модифицировать ряды. Существуют различные способы:

— дифференцирование ряда — т.е. переход непосредственно от значений к их изменениям. Эта операция увеличивает вероятность успеха и может повторяться несколько раз, однако каждая итерация затрудняет интерпретацию полученных данных, поэтому в данной работе дифференцирование про-

водится лишь один раз. С данными, для которых однократное дифференцирование не приводит к положительному результату (то есть не удается получить стационарный ряд), алгоритм не работает.

— метод окна – использовать лишь часть известного ряда. Подход основан на идее, что чем короче ряд тем больше он похож на стационарный.

Выбор параметров

Ключевым параметром при использовании теста Гренджера является порядок модели (порядок лагирования), т.е. количество предыдущих измерений (значений ряда), учтенных при прогнозировании очередного значения. Если этот параметр не может быть выбран на основе априорного знания, то могут быть использованы информационные критерии Акаике (Akaike information criterion, AIC) [4] или Байеса (Bayesian information criterion, BIC) [5], позволяющие сравнивать модели с различным числом параметров. Рассмотрим их подробнее.

В данной работе рассматривается только линейная регрессия, поэтому критерий Акаике может быть представлен в виде:

$$AIC = 2p + m \ln \frac{Q}{m}.$$

Аналогичный вид в случае линейной регрессии имеет и байесовский критерий, однако функция штрафа за сложность модели (т.е за ее порядок) здесь жестче:

$$BIC = p \ln m + m \ln \frac{Q}{m}.$$

В обоих случаях наилучшей модели соответсвует минимальное значение критерия. Важно, что модели сравниваюстя по выборкам одинаковой длины.

Прогнозирование временных рядов

В данном эксперименте регрессия строится следующим образом: пусть $\mathbf{x}_{K\times T}$ — матрица, строки которой содержат элементы временных рядов, а столбцы соответсвуют моментам времени. В ней выделим последние p столбцов. Чтобы определить матрицу коэффицентов регрессии, составим матрицу R размером $(T-p)\times Kp$, полученную из $\mathbf{x}$, со строками r_i вида:

$$r_i = \begin{pmatrix} \mathbf{x}_1(T-2p+k-1) \\ \mathbf{x}_1(T-2p+k) \\ \cdots \\ \mathbf{x}_1(T-p+k-1) \\ \mathbf{x}_2(T-2p+k-1) \\ \cdots \\ \cdots \\ \mathbf{x}_K(T-p+k-1) \end{pmatrix}^T . \tag{1}$$

Тогда

$$\tilde{\mathbf{x}_i} = R\beta_i,$$

где $\tilde{\mathbf{x}_i}$ — вектор, составленный из элементов ряда $\mathbf{x}_i$, от $T-p$ до T, b_i – - вектор коэффицентов регрессии. Таким образом можем определить β_i:

$$\beta_i = \tilde{\mathbf{x}_i} R^{-1}.$$

Пусть мы хотим выяснить, зависит ли ряд $\mathbf{x}(t)$ от $y(t)$. Действуем по следующей схеме [6, гл.17]:

1. Прогнозируем ряд $\mathbf{x}$ линейной регрессией с порядком p. При этом используем значения этого ряда и любых других известных рядов, кроме $y(t)$. По полученным данным вычисляем

$$RSS_R = \sum_{i=1}^{n} (\tilde{\mathbf{x}}(t-i) - \mathbf{x}(t-i))^2.$$

Здесь RSS_R — обозначение для residual sum of squares (restricted), $\tilde{\mathbf{x}}(t-i)$ — по-прежнему, спрогнозированное значение элемента ряда, $\mathbf{x}(t-i)$ — его истинное значение.

2. Прогнозируем ряд $\mathbf{x}(t)$ линейной регрессией, но уже с использованием элементов ряда $y(t)$. По полученным данным вычисляем
$$RSS_{UR} = \sum_{i=1}^{n} (\hat{\mathbf{x}}(t-i) - \mathbf{x}(t-i))^2.$$
Здесь RSS_{UR} — обозначение для residual sum of squares (unrestricted).
3. Нулевая гипотеза: элементы Y не участвуют в регрессии.
4. Определим величину F следующим образом:
$$F = \frac{(RSS_R - RSS_{UR})/p}{RSS_{UR}/(n-m)}.$$
Обозначения: p — количество элементов ряда Y, задействованных в регрессии п.2, совпадающее с порядком модели; m — длина контрольной выборки.
5. Если вычисленное значение F превосходит некоторое критическое значение, отвергаем нулевую гипотезу, т.е $\mathbf{x}$ зависит от y.
6. Аналогичные действия проводим на случай существования обратной связи.

Вычислительный эксперимент

Работа алгоритма на модельных данных

В ходе эксперимента использовались синтетические данные(рис. 1), состоящие из пяти рядов, для получения которых поступим следующим образом: сгенерируем матрицу $\mathbf{x}_{5\times T}$, каждый элемент которой — нормальная случайная величина; затем переопределим каждый столбец, начиная с пятого, чтобы каждый элемент такого столбца удовлетворял формуле

$$\mathbf{x}_1(i) = 1.6\mathbf{x}_1(i-1) + 0.65\mathbf{x}_2(i-2),$$

$$\mathbf{x}_2(i) = 1.5\mathbf{x}_2(i-1) - 0.3\mathbf{x}_2(i-2) - 0.3\mathbf{x}_3(i-4) + 0.6\mathbf{x}_4(i-1),$$

$$\mathbf{x}_3(i) = 1.8\mathbf{x}_3(i-1) - 0.7\mathbf{x}_3(i-2) - 0.1\mathbf{x}_5(i-3),$$

$$\mathbf{x}_4(i) = 1.5\mathbf{x}_4(i-1) + 0.9\mathbf{x}_3(i-2) + 0.4\mathbf{x}_5(i-2),$$

$$\mathbf{x}_5(i) = 1.7\mathbf{x}_5(i-1) - 0.5\mathbf{x}_5(i-2) - 0.2\mathbf{x}_3(i-1).$$

Таким образом каждый ряд определяется не только своей историей, но и прошлыми значениями других рядов. Например, ряд $\mathbf{x}_5$ зависит от $\mathbf{x}_1$, а изменения рядов $\mathbf{x}_3$ и $\mathbf{x}_4$ повлекут за собой изменение $\mathbf{x}_2$. Эта зависимость продемонстрирована на рисунках 2 и 3.

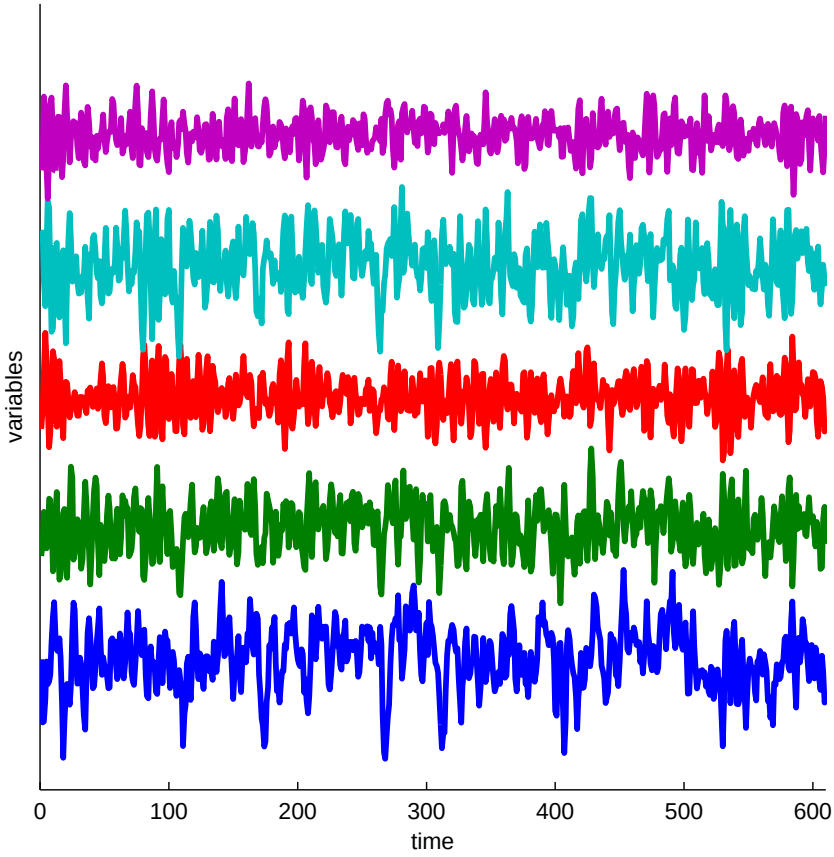


Рис. 1. Вид исследуемых рядов.

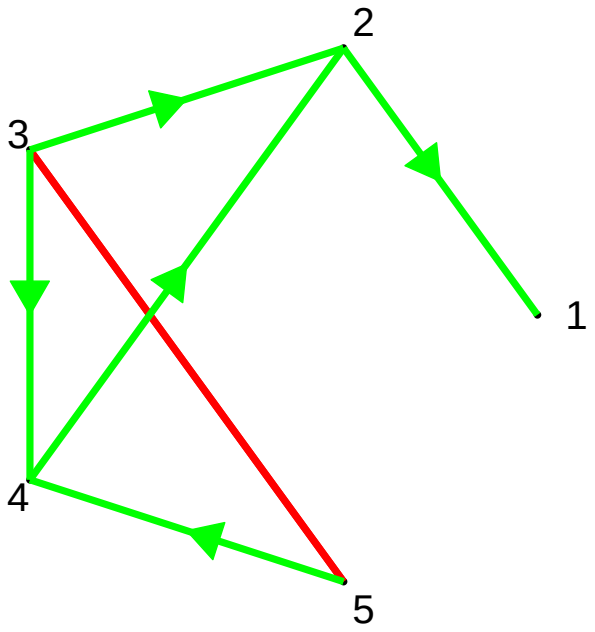


Рис. 2. Схема зависимостей между рядами. Цифры соответствуют номерам рядов, например, ряд $\mathbf{x}_2$ зависит от $\mathbf{x}_3$ и $\mathbf{x}_4$. Красной линией обозначена двусторонняя связь.

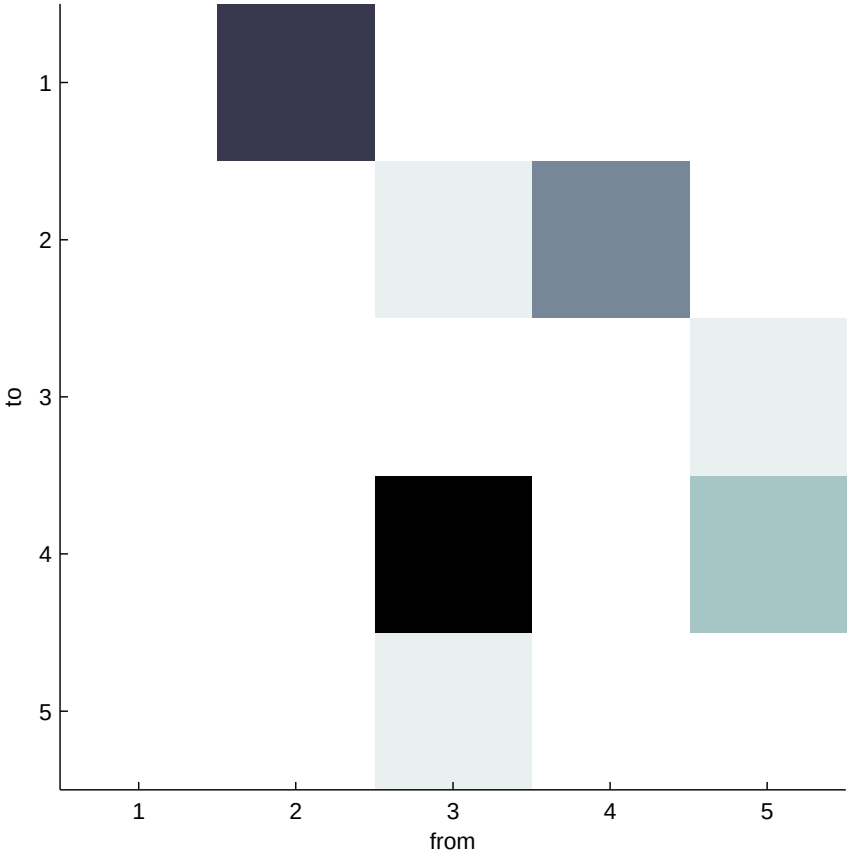


Рис. 3. Альтернативный способ представления результатов. Здесь закрашенная клетка на пересечении строки и столбца означает зависимость элемента слобца от элемента строки. Например, 2 влечет за собой 1.

В данном случае все ряды оказались стационарными, дополнительная обработка не потребовалась. Критерии AIC и BIC дали оптимальное значение порядка модели, равное 2. Этот результат можно проверить, постороив зависимость функционала качества от порядка модели. Для наглядности будем использовать так называемый относительный функционал качества:

$$\tilde{Q} = \sum_{i=1}^{n} \left(\frac{\tilde{\mathbf{x}}(i) - \mathbf{x}(i)}{\mathbf{x}(i)} \right)^2 .$$

Рис. 4 и 5 демонстрируют, что при увеличении порядка модели ($p > 2$) функционал резко возрастает, в соответствии с критериями Аикаке и Байеса.

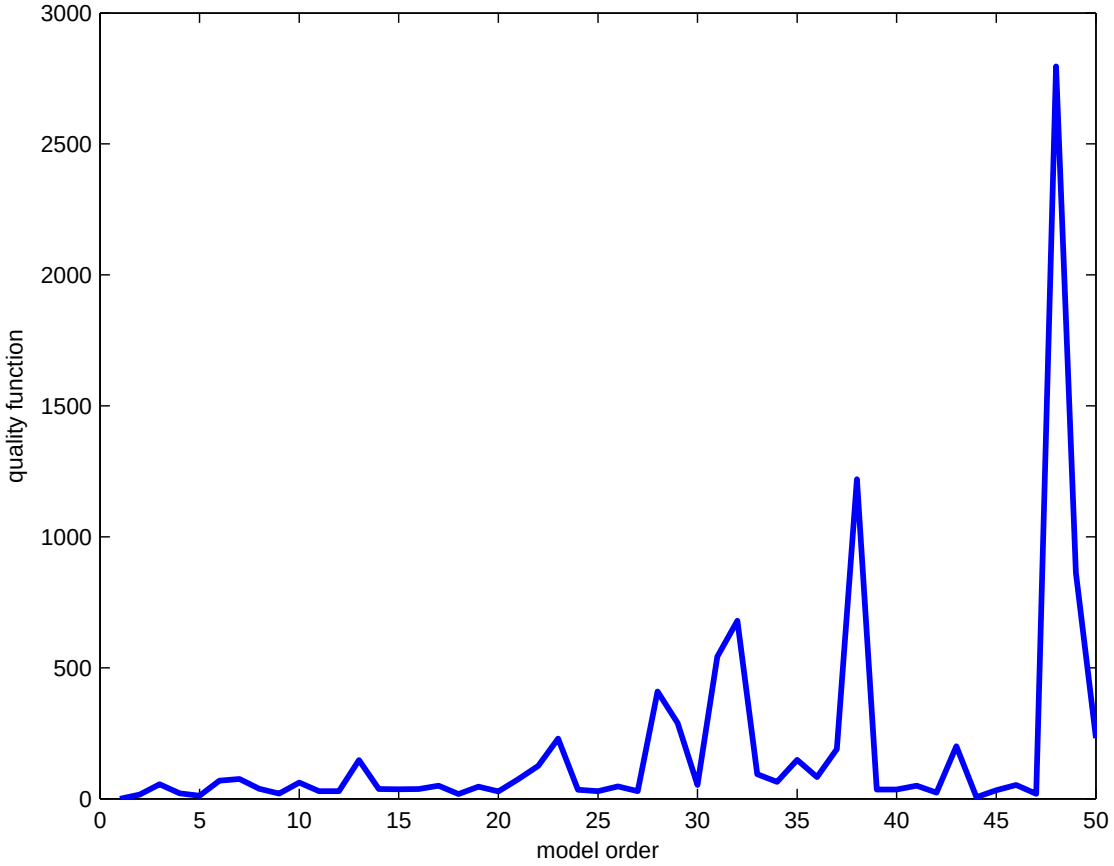


Рис. 4. Зависимость относительного функционали качества от порядка модели p для $\mathbf{x}_2$.При p больших 5 значение ошибок значительно возрастает.

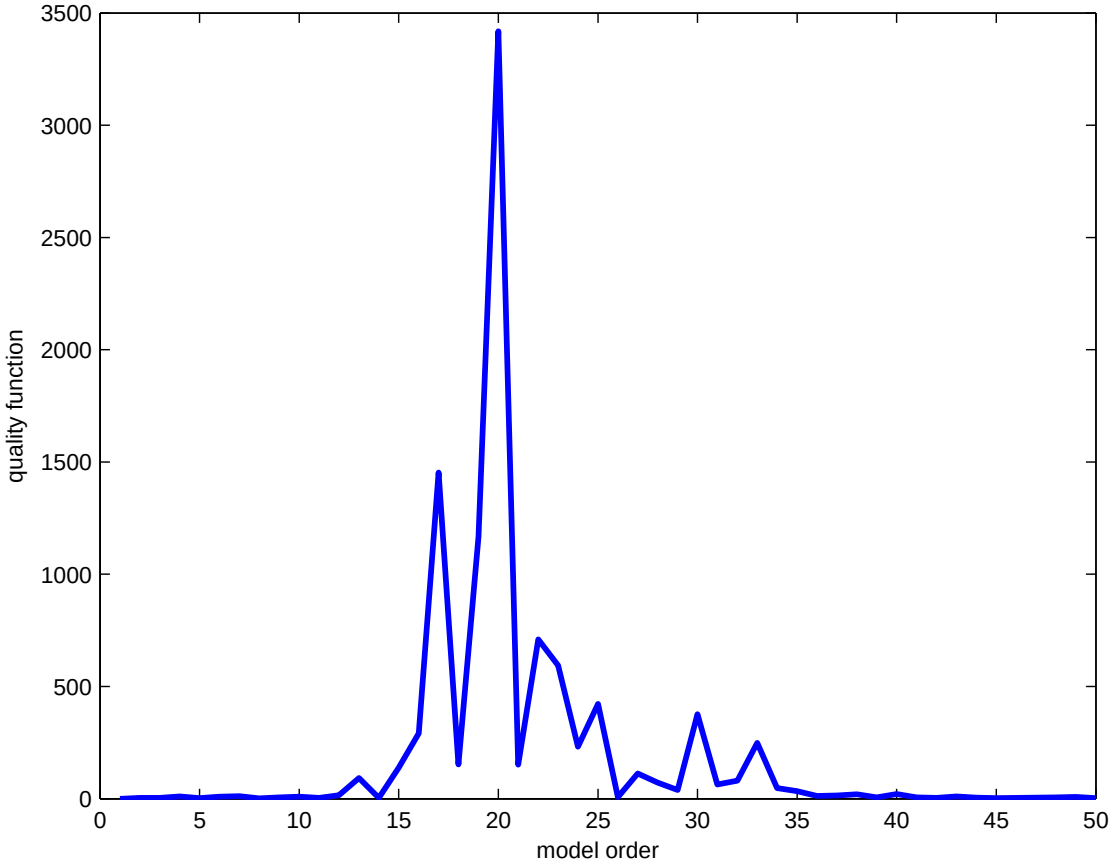


Рис. 5. Аналогичная зависимость для $\mathbf{x}_3$.Здесь качество прогнозирования ухудшается уже при p больших 2, что и соответсвует значениям, найденным с помощью BIC и AIC.

Работа алгоритма на реальных данных

В работе так же использовались реальные данные — цены на различные виды товаров за каждый месяц с 1960 по 2010 год [9]. В частности, были исследованы

1. зависимость цен на сахар в США от цен на сахар в Европе 6.
2. связь между ценами на природный газ в США и Европе и ценами на энергию (World Bank) **??**.

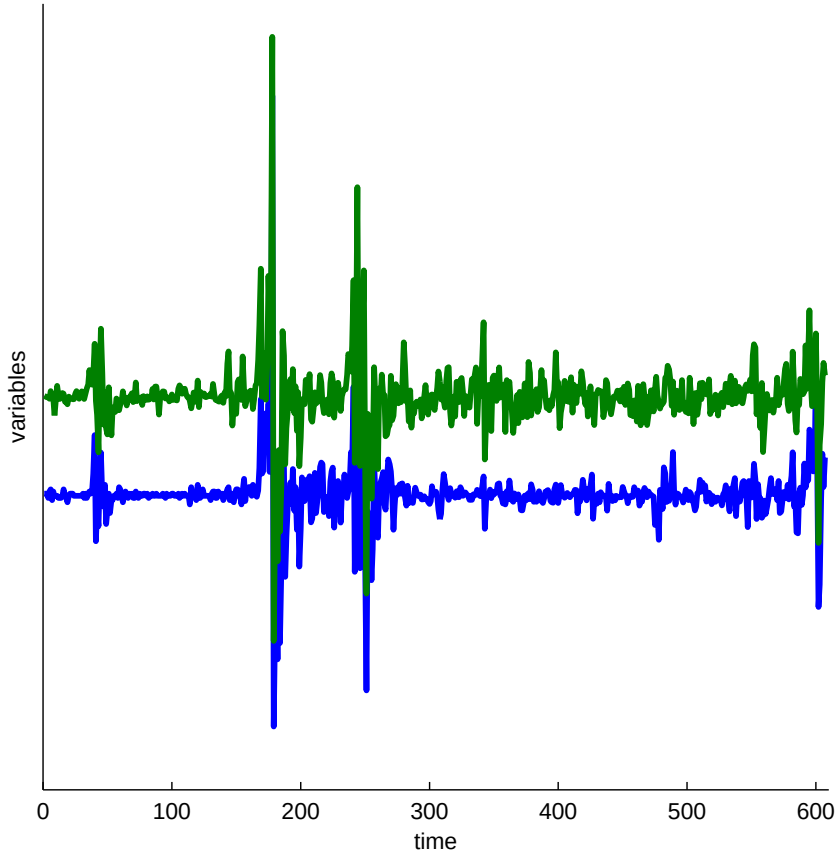


Рис. 6. Вид зависимостей цен на сахар от времени: синим в — Европе, зеленым — в США.

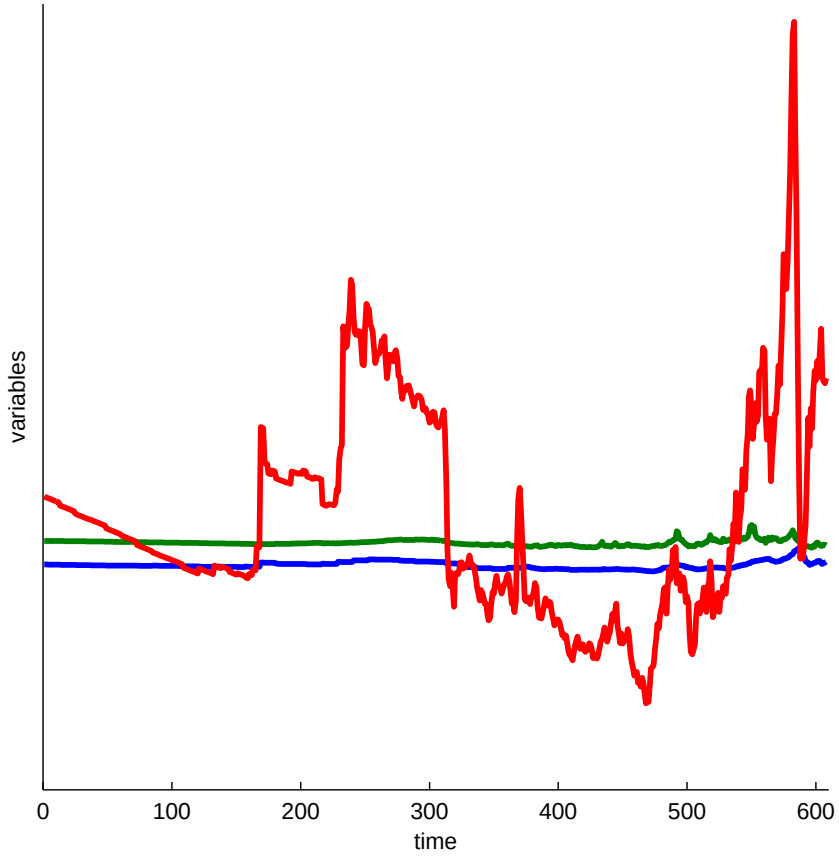


Рис. 7. Вид зависимостей для данных (2). Синим и зеленым цвет соответствует ценам на газ в Европе и Америке, красным — ценам на энергию.

В первом случае алгоритм выявил зависимость вида "цены на сахар в Европе влияют на цены на сахар в США". Такого результата можно было ожидать, изучив вид рис. 6. Чтобы убедиться в правильности результата, построим также зависимость относительных функционалов качества от порядка модели прогноза цен в США с использованием данных о ценах (назовем его unrestricted) в Европе и без (restricted) 8. Видно, что оценка, полученная при учете дополнительной информации, более точна.

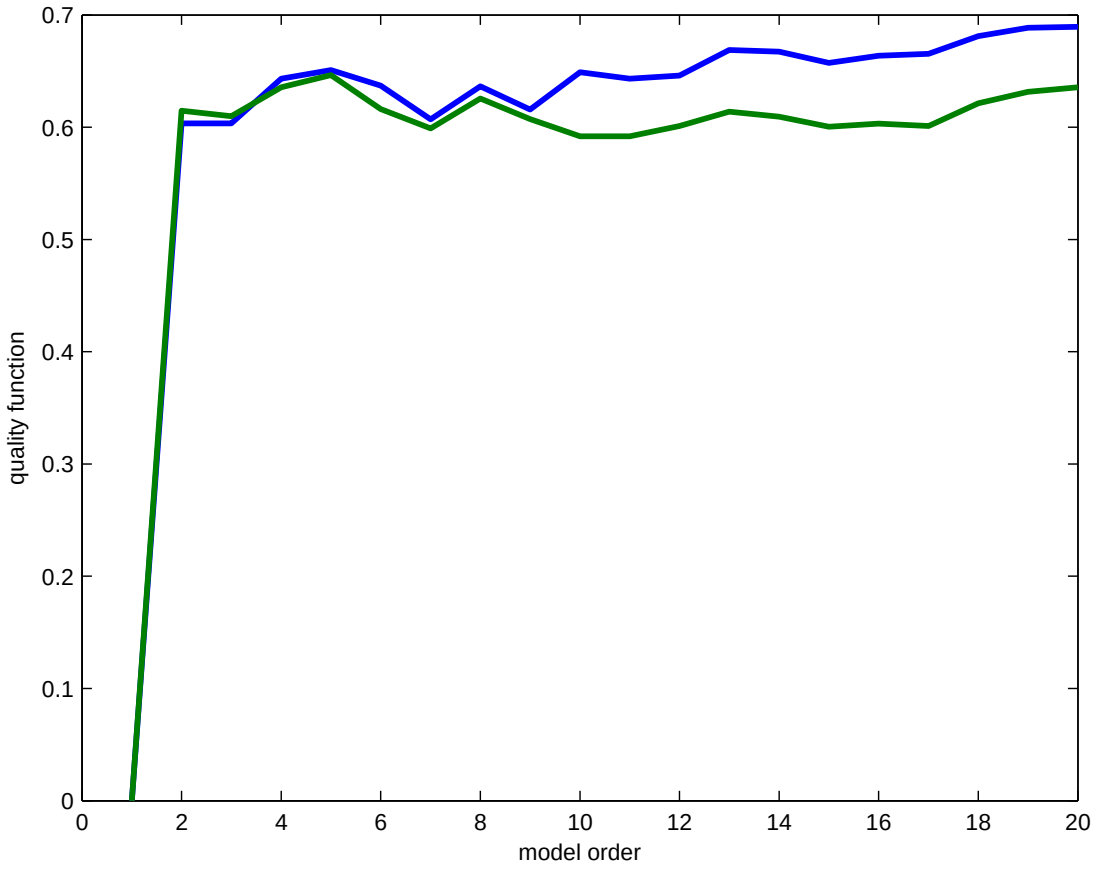


Рис. 8. Зависимость относительных функционалов качества от порядка модели: синим цветом — для прогнозирования по собственной истории(цены на сахар в США), зеленым — по истории цен на сахар в США и Европе.

Результаты, полученные во втором случае, представлены на рисунке 9. Чтобы продемонстрировать справедливость результата, на этот раз представим на одном рисунке графики зависимостей исходного ряда и его предсказанных значений от времени 10. Как и в предыдущем случае, построим для ряда 1 unrestricted и restricted прогнозы. Из рис. 11, отражающего ту же зависимость, но в другом масштабе, видно что использование вспомогательных данных привело к более точному прогнозу.

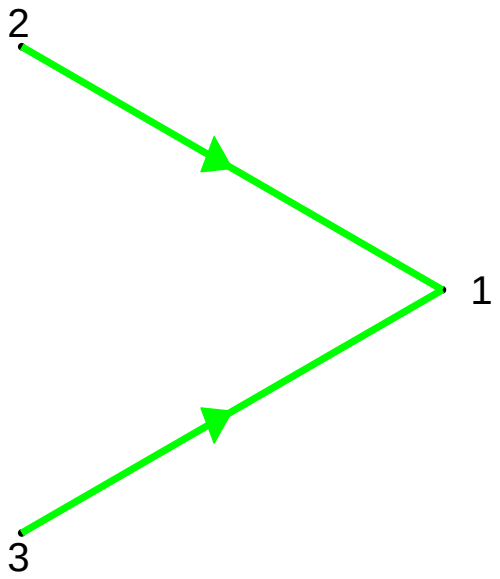


Рис. 9. 1 — цены на природный газ в Европе,2 — в США, 3 — цены на энергию. Здесь, например, 1 зависит от 3 и 2.

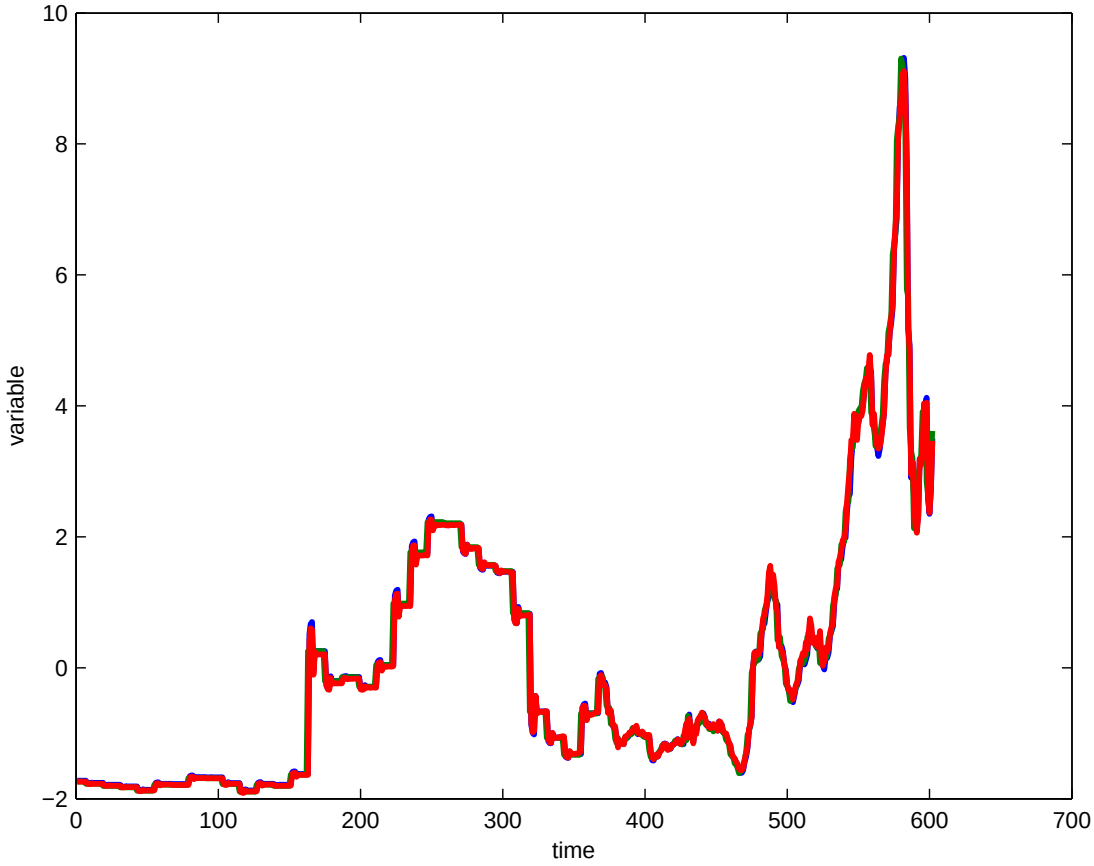


Рис. 10. Прогноз для цен на естественный газ в Европе; зеленым цветом обозначены исходные данные, синим — полученные в результате работы алгоритма.Ввиду нестационарности данных, на графиках отображены не сами значения исследуемых величин, а их изменения.

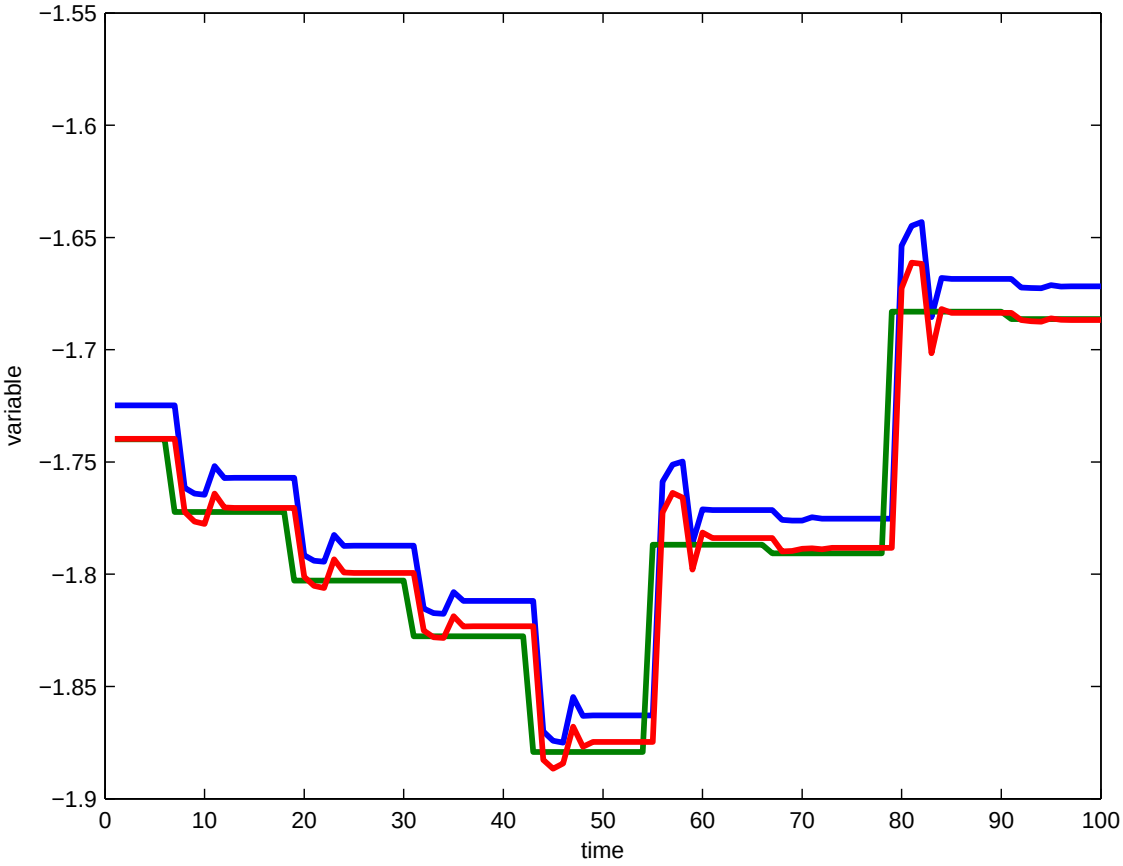


Рис. 11. Прогноз цен на естественный газ в Европе, в увеличенном масштабе. Здесь зеленым цветом обозначены исходные данные, синим — полученные в результате прогнозирования с использованием вспомогательных данных, красным — прогноз по истории самого ряда.

В заключение данного раздела отметим, что из среди приведенных примером стационарностью не обладали временные ряды во втором случае. Из указанных в разделе "Подготовка данных-методов обработки наиболее действенным оказалось дифференцирование. В действительности, ни в одном из рассмотренных случаев метод окна не привел к положительному результату. Что касается применения алгоритма к заведомо нестационарным данным, возможны как точный прогноз, так и полное несоответствие с действительностью.

Заключение

В работе рассмотрена возможность использования теста Гренджера при составлении прогнозов. Исследована зависимость качества работы алгоритма от различных параметров, выбор которых проанализирован с точки зрения качества прогнозирования; также рассмотрены способы обработки входных данных. Анализ проводился на модельных и реальных данных.

Необходимый для построения вычислительного эксперимента код можно найти найти на сайте:
`https://mlalgorithms.svn.sourceforge.net/svnroot/mlalgorithms/GrangerForecasting/`

Литература

[1] C. W. J. Granger. *Investigating Causal Relations by Econometric Models and Cross-spectral Methods*, Econometrica, vol.37,424 - 432, 1969.

[2] J. D. Hamilton. *Time series analysis*, Princeton University Press, 1994.

[3] D. Kwiatskowski & Peter C. B. Phillips and Peter Schmidt & Yongcheol Shin. *Testing the null hypothesis of stationarity against the alternative of a unit root*, Journal of Econometrics, vol. 54, 159-178, 1992.

[4] H. Akaike. *A new look at the statistical model identification*, IEEE Trans. Autom.Control, vol. 19, 716-723, 1974.

[5] *Информационный критерий Байеса на MacnineLearning.ru*

[6] D. Gujarati. *Basic Econometrics, 4th ed*, The McGraw-Hill Companies, 2004.

[7] A. K. Seth. *A MATLAB toolbox for Granger causal connectivity analysis*,Journal of Neuroscience Methods,vol. 186, 262 - 273, 2010.

[8] A. K. Seth. *Granger causality*, Scolarpedia, 2(7), 2007.

[9] `https://mlalgorithms.svn.sourceforge.net/svnroot/mlalgorithms/TSForecasting/TimeSeries/Sources/tsEarthquakesArkansas.csv`

Исследование сходимости при прогнозировании нейронными сетями с обратной связью

Н. П. Балдин
baldin01@gmail.com

Исследуется зависимость скорости сходимости при прогнозировании временных рядов от параметров нейронной сети с обратной связью. В качестве модели нейронной сети используется сеть Джордана. Предлагается проанализировать скорость сходимости в зависимости от выбора функции активации (сигмоидной, гиперболического тангенса), от числа нейронов в промежуточном слое и от ширины скользящего окна. Также разбирается способ повышения скорости сходимости при использовании обобщенного дельта-правила.
Ключевые слова: *машинное обучение, нейронные сети, сеть Джордана, градиентный спуск, обобщенное дельта-правило.*

Введение

Первые в шаги в области нейронных сетей были сделаны В. Мак-Калохом (W. MCculloch) и В. Питсом (W. Pitts). Они показали, что при помощи нейронных элементов можно реализовать исчисление любых логический функций [1]. В 1949 г. Хебб (D. Hebb) предложил правило обучения, которое стало математической основой для обучения ряда нейронных сетей. В 1957—1962 гг. Ф. Розенблат (F. Rosenblatt) предложил и исследовал модель нейронной сети, которую он назвал персептроном. Она была популярной до 1969 г., когда М. Минский (M. Minsky) и С. Пайперт (S. Pappert) опубликовали монографию [2], в которой были доказаны ограниченные возможности персептрона. В 1986 г. ряд авторов (D. Rumerhalt, G. Hinton, R. Williams) предложили алгоритм обратного распространения ошибки для обучения многослойной нейронной сети [3].

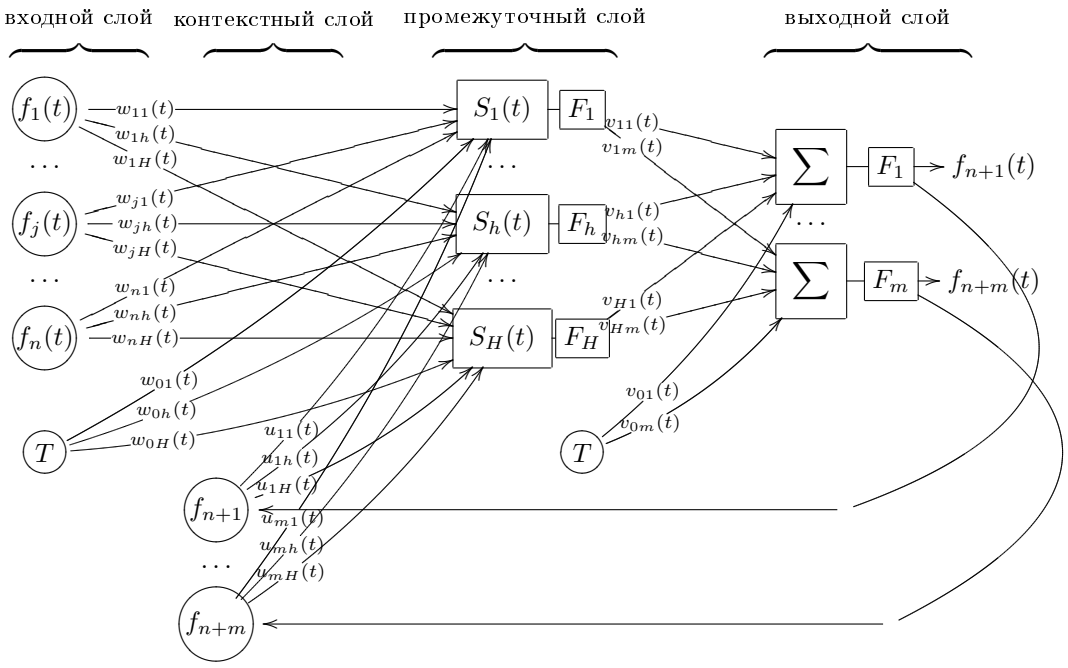


Рис. 1. Архитектура сети Джордана

Рекуррентные сети — нейронные сети, в которых выходы нейронных элементов последующих слоев имеют соединение с нейронами предшествующих слоев (Рис. 1). Выходное значение нейрона промежуточного слоя:

$$S_i(t) = \sum_{j=1}^{n} w_{ij}(t)x_j + \sum_{k=1}^{p} u_{ki}(t)y_k(t-1) - T_i,$$

где w_{ij}—весовой коэффициент между j-м нейроном входного и i-м нейроном промежуточного слоев; p — число нейронов выходного слоя; u_{ki}— весовой коэффициент между k-м контекстным нейроном и i-м нейроном промежуточного слоя; T_j —пороговое значение i-го нейрона промежуточного слоя; n — размерность входного вектора. Это приводит к возможности учета результатов преобразования нейронной сетью информации на предыдущем этапе для обработки входного вектора на следующем этапе функционирования сети. Существует множество архитектур нейронных сетей [4, 5, 6, 7]. Для изучения сходимости в данной работе используется простейшая архитектура рекуррентной сети — сеть Джордана, представленная в [7]. В качестве модельных данных используются тригонометрические функции (периодический, зашумленный и апериодический временные ряды).

Постановка задачи

Дано:

$$f_i : X \to R, i = 1, ..., n \text{ —числовые признаки,}$$
$$\langle \mathbf{w}, \mathbf{x_j} \rangle = \sum_{i=1}^{n} f_i(x) w_i - w_0 \text{ — линейная комбинация признаков}$$
$$w_0, w_1, ..., w_n \in R.$$

Определение 1. *Нейрон — вершина взвешенного ориентированного графа (нейронной сети).*

Определение 2. *Функция активации — оператор не ного преобразования входных данных нейрона промежуточного слоя.*

Совокупность известных значений временного ряда образуют обучающую выборку. Обозначим ее размерность L. Архитектура нейронной сети: n входных нейронов (в дальнейшем, ширина окна), p нейронов промежуточного слоя, один выходной нейрон, характеризующий значение в момент времени $n + 1$. Требуется по известным значениям ряда определить значение в последующий момент времени.
Вычисляется среднеквадратичная ошибка.

$$E = \sum_{i}^{m} (y_i - t_i)^2,$$

где y_i — значение на i-м выходе, t_i — эталонное значение. Затем запускается алгоритм обратного распространения ошибки. Исследуется сходимость от числа нейронов в промежуточном слое, от функции активации, и ширины окна. Для этого нужно для каждого набора параметров найти число итераций алгоритма. Далее найдем оптимальный набор параметров для каждого временного ряда. Проанализируем один из способов ускорения сходимости.

Решение задачи

Описание архитектуры

Выходное значение j-го нейрона последнего слоя определяется по формуле

$$y_j(t) = \sum_{i=1}^{m} v_{ij} p_i(t) - T_j,$$

где v_{ij} —весовой коэффициент между i-м нейроном промежуточного слоя и j-м нейроном выходного слоев. $p_i(t)$— выходное значение i-го нейрона промежуточного слоя; T_j— пороговое значение j-го нейрона выходного слоя.
Взвешенная сумма i-го нейрона промежуточного слоя определяется следующим образом:

$$S_i(t) = \sum_{j=1}^{n} w_{ij}(t) x_j + \sum_{k=1}^{p} u_{ki}(t) y_k(t-1) - T_i.$$

Тогда выходное значение i-го нейрона промежуточного слоя $p_i(t) = F(S_i(t))$, где F — функция активации. В качестве F используется гиперболический тангенс или сигмоидальная функция.

Алгоритм обучения

Алгоритм обратного распространения ошибок был предложен в [3] и является эффективным средством для обучения многослойных нейронных сетей. Алгоритм обучения реккурентной сети в общем случае состоит из следующих шагов:

1. В начальный момент времени $t = 1$ все контекстные нейроны устанавливаются в нулевое состояние, т. е. их выходные значения равняются нулю.
2. Входной образ подается на сеть и происходит прямое распространение его в нейронной сети.
3. В соответствии с алгоритмом обратного распространения ошибки модифицируются весовые коэффициенты и пороговые значения нейронов.
4. Устанавливается $t = t + 1$ и осуществляется переход к шагу 2.

Определяется среднеквадратичная ошибка нейронной сети:

$$E = \frac{1}{2}\sum_j (y_j - t_j)^2,$$

где t_j — эталонное выходное значение j-го нейрона. Согласно методу градиентного спуска изменение весовых коэффициентов и порогов нейронной сети происходит по следующему правилу:

$$w_{ij}(t+1) = w_{ij}(t) - a\frac{\partial E}{\partial w_{ij}(t)},$$
$$T_j(t+1) = T_j(t) - a\frac{\partial E}{\partial T_j(t)}.$$

Ошибка j-го выходного слоя:

$$h_j = y_j - t_j.$$

Теорема 1. *Для любого промежуточного слоя i ошибка i-го нейрона определяется рекурсивным образом через ошибки нейронов следующего слоя j:*

$$h_i = \sum_{j=1}^{m} h_j F'(S_j) w_{ij},$$

где m — число нейронов следующего слоя по отношения к слою i; w_{ij} — вес между i-м и j-м нейронами различных слоев; S_j— взвешенная сумма j-го нейрона.

Используя результаты данной теоремы, сформулированной и доказанной в [7], можно определить ошибки нейронов промежуточного слоя через ошибки нейронов следующего слоя по отношению к промежуточному слою.

Теорема 2. *Производные среднеквадратичной ошибки по весовым коэффициентам и порогам нейронов для любых двух слоев i и j многослойной сети определяются следующим образом:*

$$\frac{\partial E}{\partial w_{ij}} = -h_j F'(S_j) y_j;$$
$$\frac{\partial E}{\partial T_j} = h_j F'(S_j).$$

Следствие 1. *Для минимизации среднеквадратичной ошибки сети весовые коэффициенты и пороги нейронов должны изменяться с течением времени следующим образом*

$$w_{ij}(t+1) = w_{ij}(t) - a h_j F'(S_j) y_j;$$
$$T_j(t+1) = t_j(t) + a h_j F'(S_j),$$

где a - скорость обучения. Доказательства можно посмотреть, например, в [7].

Обучение рекуррентной сети

Пусть среднеквадратичная ошибка одного входного объекта
$E(t) = \frac{1}{2}(y(t) - e)^2$, где e — эталонное значение для соответствующего входного объекта. Для определения алгоритма обратного распространения ошибки найдем производные функции среднеквадратичной ошибки по настраиваемым параметрам сети. Тогда

$$\frac{\partial E}{\partial v_i} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial v_i} = (y(t) - e)p_i(t);$$
$$\frac{\partial E}{\partial T} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial T} = -(y(t) - e);$$
$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial p_i}\frac{\partial p_i}{\partial S_i}\frac{\partial S_i}{\partial w_{ij}} = (y(t) - e)v_i F'(S_i)x_j(t);$$

$$\frac{\partial E}{\partial u_{li}} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial p_i}\frac{\partial p_i}{\partial S_i}\frac{\partial S_i}{\partial u_{li}} = (y(t)-e)v_i F'(S_i)y(t-1);$$
$$\frac{\partial E}{\partial T_i} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial p_i}\frac{\partial p_i}{\partial S_i}\frac{\partial S_i}{\partial T_i} = -(y(t)-e)v_i F'(S_i),$$

где $F'(S_i)$ — производная функции активации промежуточного слоя

Недостатки алгоритма обратного распространения ошибки

Алгоритм обратного распространения ошибки, в основе которого лежит градиентный метод, создает ряд проблем при обучении многослойных нейронных сетей. Существуют следующие проблемы:

— неизвестность выбора числа слоев и количества нейронных элементов в слое для многослойной сети,
— медленная сходимость градиентного метода с постоянным шагом обучения,
— сложность выбора подходящей скорости обучения,
— невозможность отделения точек локального и глобального минимума (градиентный метод их не различает).

Способ повышения скорости обучения

В алгоритме обратного распространения ошибки чем больше параметр обучения a, тем больше корректировка весов и, следовательно, тем выше скорость обучения. Однако, в этом случае система может перейти в неустойчивое состояние. Простейшим способом повышения скорости обучения без потери устойчивости является добавление зависимости текущего изменения весов от изменений весов на предыдущих шагах алгоритма с какими-то коэффициентами η (так называемое обобщенное дельта-правило).

$$w_{ij}(t+1) = w_{ij}(t) + \eta\Delta w_{ij}(t-1) - a\frac{\partial E(t)}{\partial w_{ij}(t)},$$

где

$$\Delta w_{ij}(t-1) = \eta\Delta w_{ij}(t-2) - a\frac{\partial E(t-1)}{\partial w_{ij}(t-1)}.$$

Особенности обобщенного дельта-правила:

1. Текущее значение коррекции весов $\Delta w_{ij}(t)$ представляет собой сумму экспоненциально взвешенного ряда. Для того чтобы ряд сходился, постоянная η должна находиться в диапазоне $0 < |\eta| < 1$ (см. [4]). Она может быть и отрицательной, хотя это не рекомендуется на практике.
2. Если частная производная $\frac{\partial E(t)}{\partial w_{ij}(t)}$имеет один и тот же знак на нескольких последовательных итерациях , то экспоненциально взвешенная сумма $\Delta w_{ij}(t)$ возрастает по абсолютному значению, поэтому веса могут меняться на значительную величину(ускорение спуска).
3. Если частная производная $\frac{\partial E(t)}{\partial w_{ij}(t)}$ на нескольких итерациях меняет знак, то экспоненциально взвешенная сумма $\Delta w_{ij}(t)$ уменьшается по абсолютной величине, поэтому веса $w_{ij}(t)$ меняются незначительно.

Вычислительный эксперимент

Данные

В качестве периодического, зашумленного и апериодического временных рядов используются модельные данные:

— $y = 0.4\sin(5x) + 0.5$,
— $y = 0.2\sin(20x) + 0.3\cos(15x) + 0.1\cos(5x) + 0.3\sin(3x)$,
— $y = 0.1x + 0.1\sin(x) + 0.05\cos(30x)$.

Размерность обучающей выборки — L, число признаков у объекта — n, число нейронов в промежуточном слое — p, функция активации — F, шаг обучения — a, минимум ошибки — E_{min}, коэффициент в обобщенном дельта-правиле —η.
Исследуются две функции активации (Рис. 2):

1. $F(S) = \frac{1}{1+e^{-S}}$ (сигмоидная),
2. $F(S) = \frac{1-e^{-2S}}{1+e^{-2S}}$ (гиперболический тангенс).

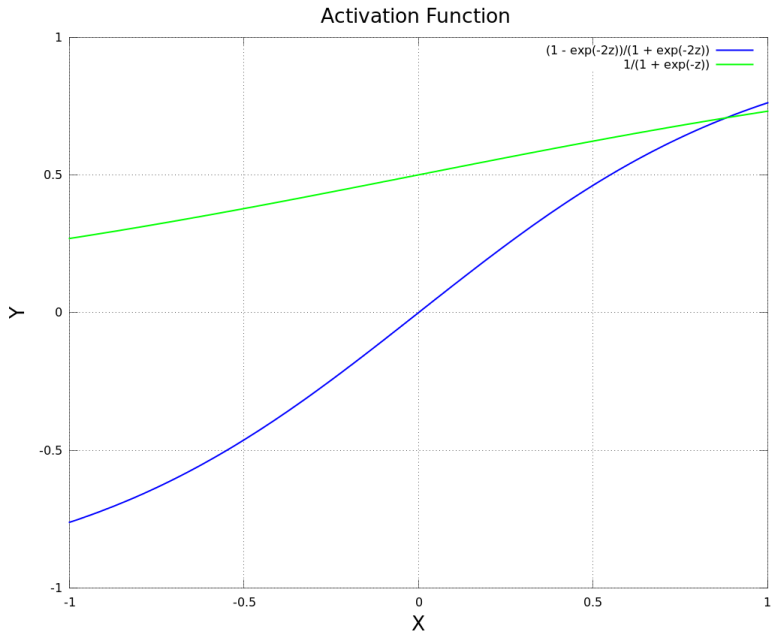


Рис. 2. Функции активации

Результаты

1. $y = 0.4\sin(5x) + 0.5$ (Рис. 2)

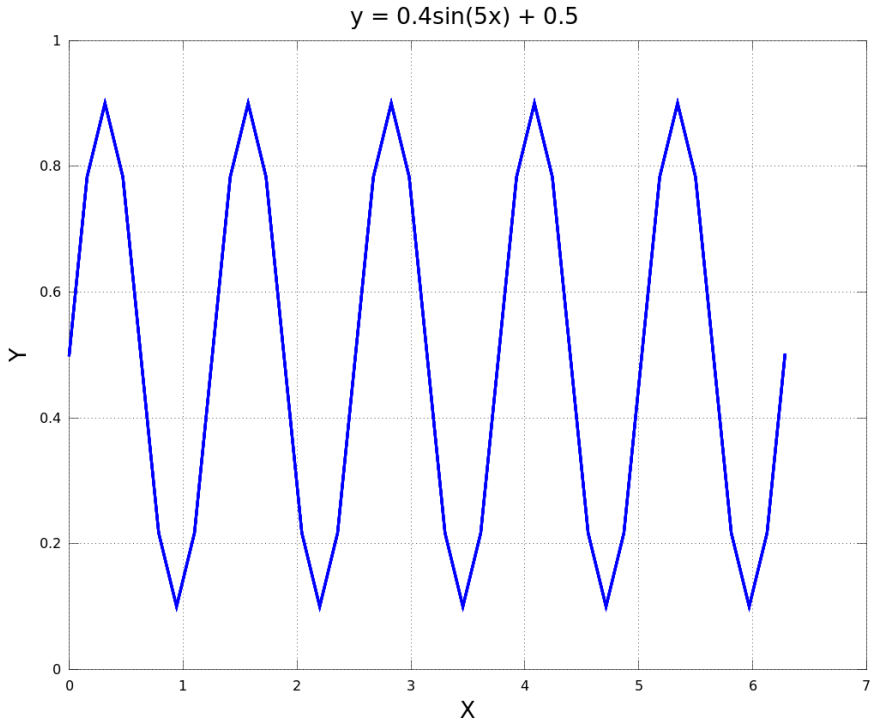


Рис. 3. График функции $y = 0.4\sin(5x) + 0.5$

Минимум ошибки $E_{min} = 0.0001$, шаг обучения $a = 0.03$. Размерность обучающей выборки $L = 40$ со значениями, равномерно расположенными на отрезке $[0, 2\pi]$. Результаты: Таб. 1.

Таблица 1. Число итераций и ошибка на контрольной выборке при прогнозировании временного ряда $y = 0.4\sin(5x) + 0.5$ в зависимости от параметров

		Число нейронов промежуточного слоя p				
		3	6	9	12	15
Ширина окна n	3	146(0.0001) 44(0.002)	81(0.031) 19(0.007)	83(0.014) 27(0.0001)	95(0.013) 21(0.002)	76(0.021) 17(0.002)
	6	626(0.000012) 31(0.002)	159(0.00001) 28(0.0009)	120(0.0001) 24(0.0005)	85(0.007) 27(0.0004)	79(0.007) 26(0.0005)
	9	53(0.05) 33(0.0002)	82(0.01) 20(0.0009)	69(0.02) 22(0.0012)	66(0.016) 23(0.0001)	64(0.017) 21(0.0002)
	12	54(0.05) 18(0.004)	67(0.02) 21(0.0024)	68(0.02) 22(0.002)	62(0.02) 22(0.0024)	71(0.02) 21(0.0024)
	15	42(0.06) 9(0.003)	81(0.03) 11(0.001)	72(0.03) 18(0.006)	67(0.02) 7(0.0041)	67(0.02) 8(0.003)
	21	65(0.03) 21(0.003)	105(0.03) 17(0.002)	88(0.03) 16(0.002)	89(0.02) 18(0.001)	78(0.02) 16(0.005)

Проиллюстрируем данные в таблице на графиках зависимости числа итераций от числа нейронов в промежуточном слое для каждой ширины окна (рис. 4).

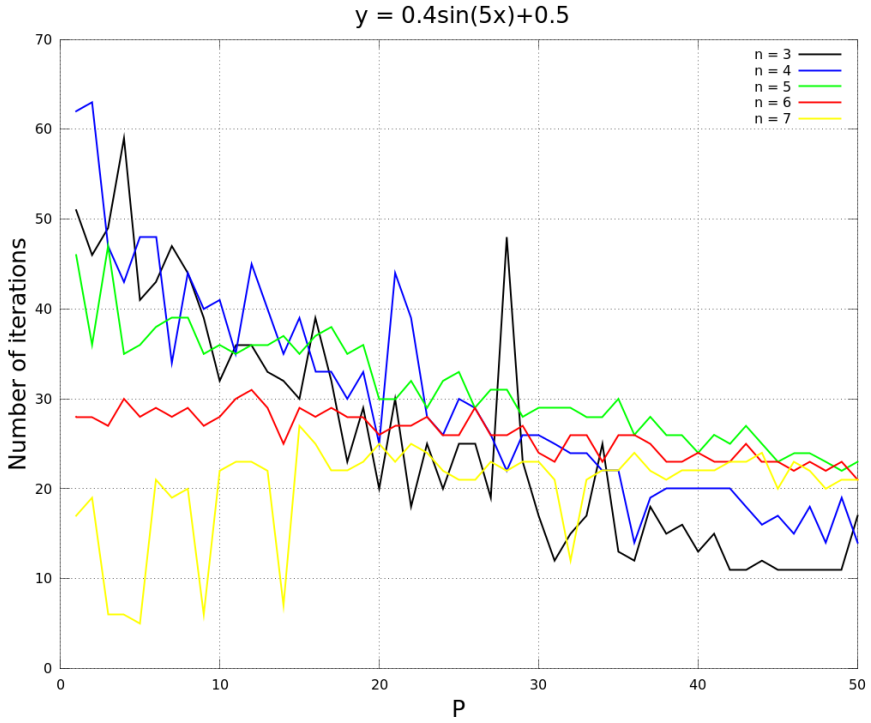


Рис. 4. Результаты для функции $y = 0.4\sin(5x) + 0.5$ (ширина окна 3 - 7).

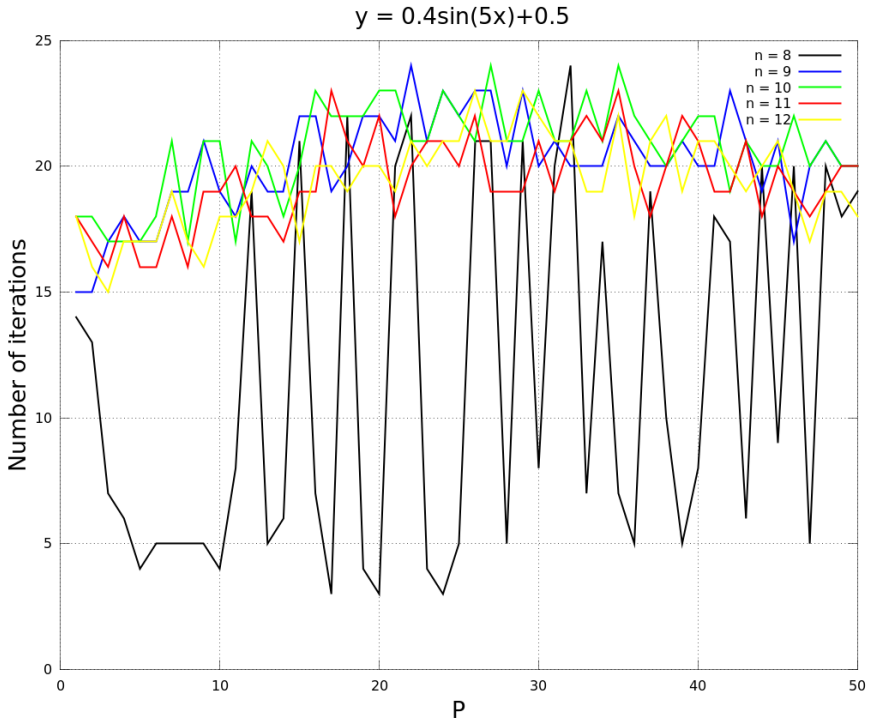


Рис. 5. Результаты для функции $y = 0.4\sin(5x) + 0.5$(ширина окна 8 - 12).

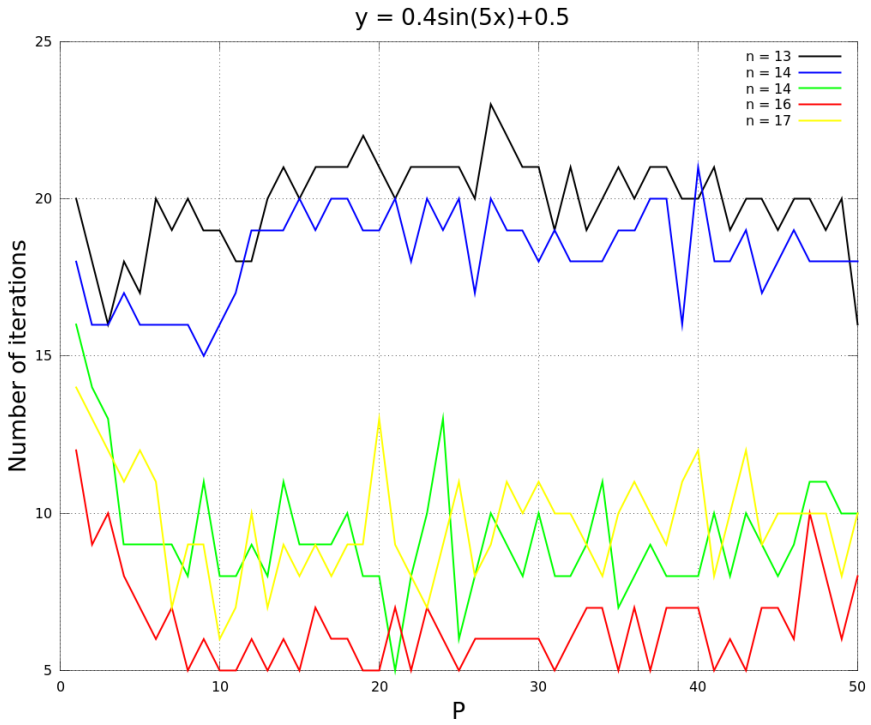


Рис. 6. Результаты для функции $y = 0.4\sin(5x) + 0.5$(ширина окна 13 - 17).

На графиках 4, 6 наблюдаем уменьшение числа итераций при увеличении числа нейронов промежуточного слоя. Результаты работы алгоритма при $n = 15$, $p = 15$, $F(S) = \frac{1-e^{-2S}}{1+e^{-2S}}$ и числе выходных нейронов 5 — Рис. 3.

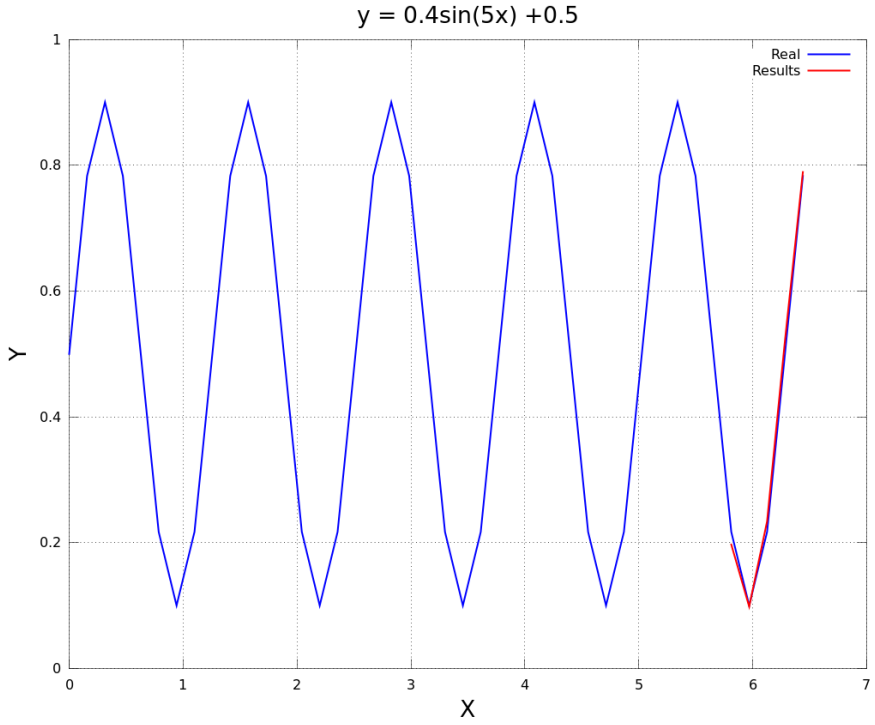


Рис. 7. Результаты для функции $y = 0.4\sin(5x) + 0.5$.

2. $y = 0.2\sin(20x) + 0.3\cos(15x) + 0.1\cos(5x) + 0.3\sin(3x)$(Рис. 4).

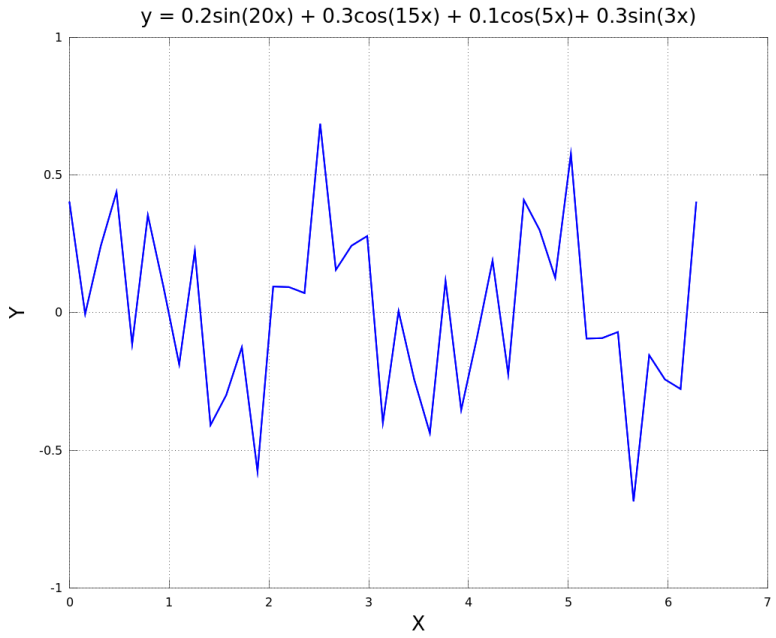


Рис. 8. График функции$y = 0.2\sin(20x) + 0.3\cos(15x) + 0.1\cos(5x) + 0.3\sin(3x)$.

Минимум ошибки $E_{min} = 0.0001$, шаг обучения $a = 0.03$. Размерность обучающей выборки $L = 40$ со значениями, равномерно расположенными на отрезке $[0, 2\pi]$. Результаты: Таб. 2

Таблица 2. Число итераций и ошибка на контрольной выборке при прогнозировании временного ряда $y = 0.2\sin(20x) + 0.3\cos(15x) + 0.1\cos(5x) + 0.3\sin(3x)$ в зависимости от параметров.

		Число нейронов промежуточного слоя p				
		3	6	9	12	15
Ширина окна n	3	»2000 »2000	»2000 »2000	»2000 »2000	»2000 »2000	»2000 »2000
	6	426(0.006) 98(0.006)	358(0.007) 118(0.005)	322(0.008) 102(0.005)	225(0.01) 127(0.006)	181(0.017) 124(0.005)
	9	665(0.0003) 190(0.0003)	687(0.0002) 191(0.0001)	666(0.0002) 186(0.0001)	773(0.00009) 187(0.0001)	781(0.00005) 185(0.0002)
	12	427(0.0005) 99(0.00026)	411(0.0003) 117(0.00059)	390(0.0004) 105(0.0002)	405(0.0004) 106(0.0002)	421(0.0004) 93(0.00001)
	15	559(0.00006) 130(0.00005)	537(0.00006) 142(0.00005)	540(0.00003) 144(0.00005)	560 (0.00002) 170(0.00006)	534(0.00001) 151(0.00005)
	21	644(0.00004) 119(0.0001)	510(0.00005) 113(0.00005)	492(0.00003) 115(0.00004)	490 (0.000006) 117(0.0002)	517(0.0000001) 124(0.0003)

Проиллюстрируем данные в таблице на графиках зависимости числа итераций от числа нейронов в промежуточном слое для каждой ширины окна (рис. 9).

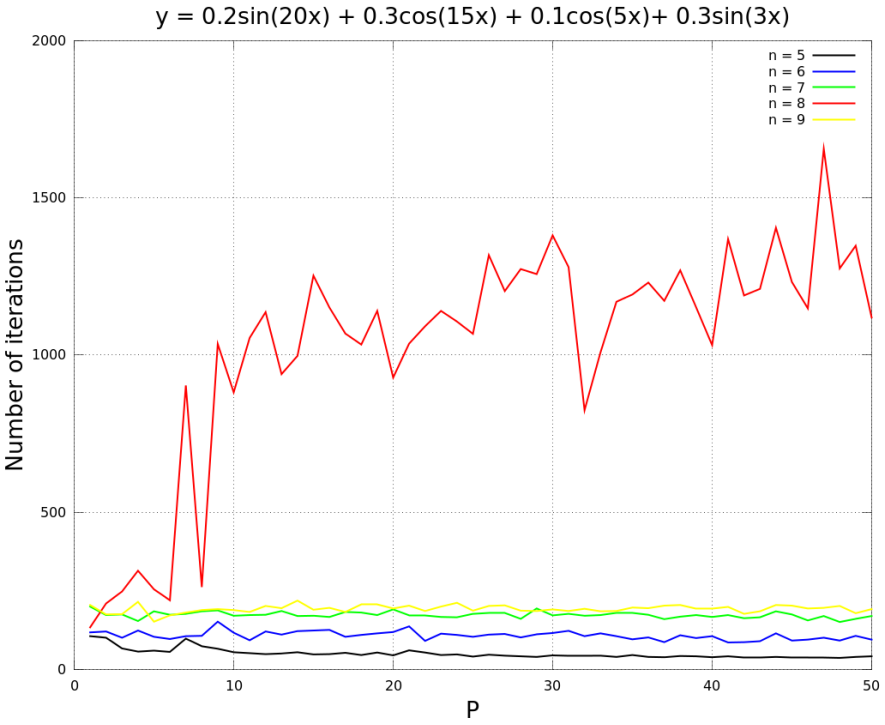


Рис. 9. Результаты для функции $y = 0.2\sin(20x) + 0.3\cos(15x) + 0.1\cos(5x) + 0.3\sin(3x)$ (ширина окна 5 - 9).

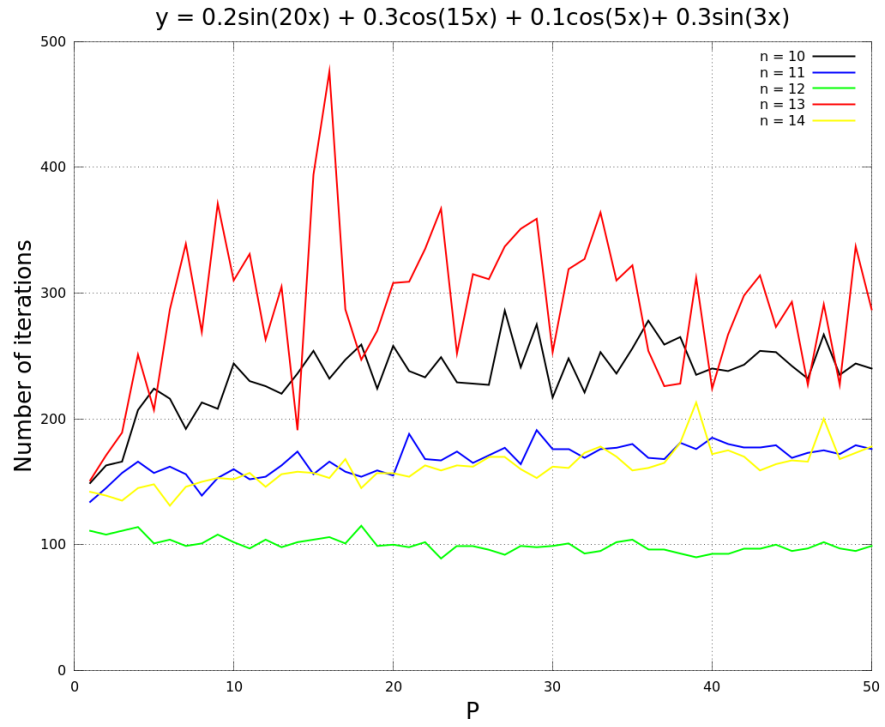


Рис. 10. Результаты для функции $y = 0.2\sin(20x) + 0.3\cos(15x) + 0.1\cos(5x) + 0.3\sin(3x)$ (ширина окна 9 - 14).

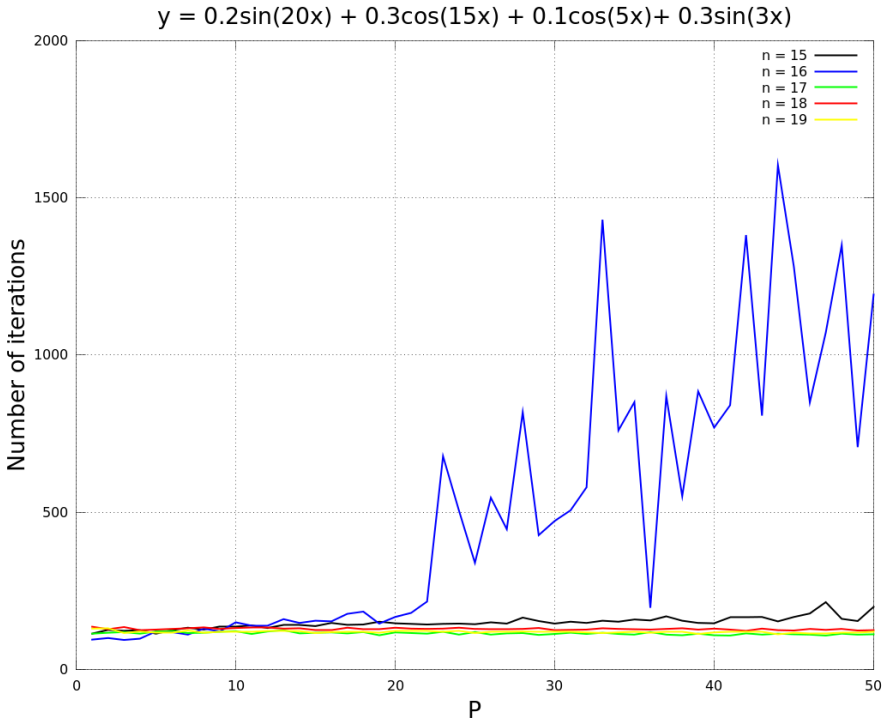


Рис. 11. Результаты для функции $y = 0.2\sin(20x) + 0.3\cos(15x) + 0.1\cos(5x) + 0.3\sin(3x)$ (ширина окна 15 - 19).

Не удалось выявить зависимости числа итераций как от числа нейронов промежуточного слоя, так и от ширины окна. Результаты работы алгоритма при $n = 12$, $p = 15$, $F(S) = \frac{1-e^{-2S}}{1+e^{-2S}}$ и числе выходных нейронов 5 — Рис. 12.

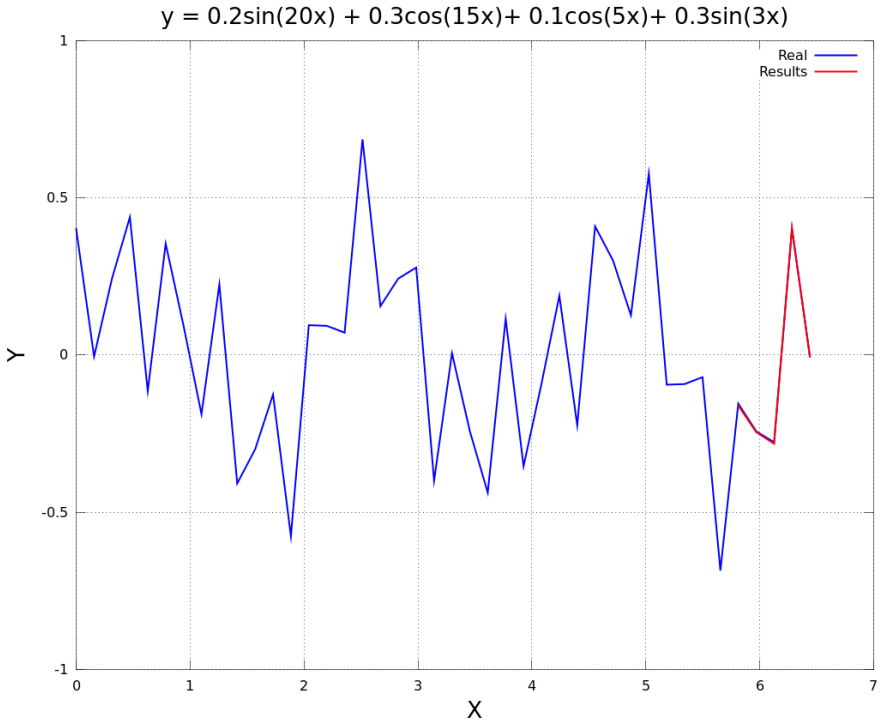


Рис. 12. Результаты алгоритма для функции $y = 0.2\sin(20x) + 0.3\cos(15x) + 0.1\cos(5x) + 0.3\sin(3x)$.

3. $y = 0.1x + 0.1\sin(x) + 0.05\cos(30x)$(Рис. 13).

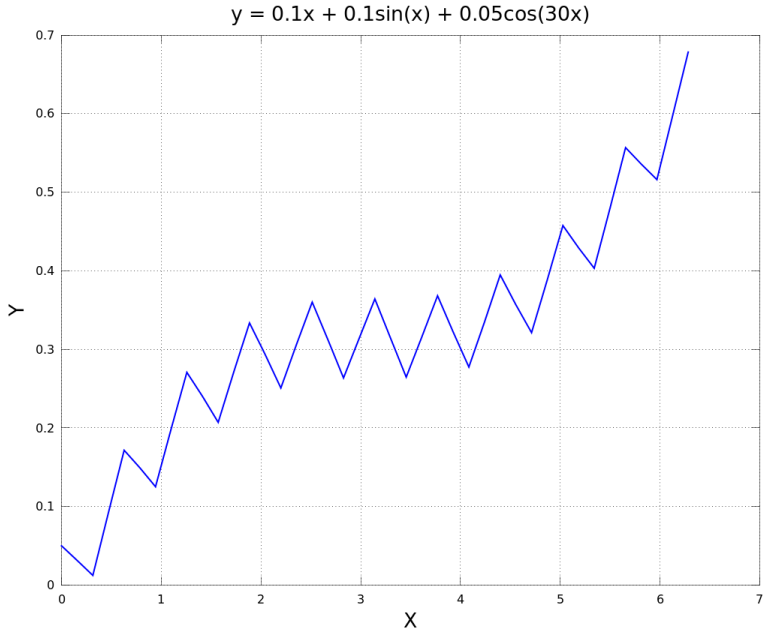


Рис. 13. График функции $y = 0.1x + 0.1\sin(x) + 0.05\cos(30x)$.

Минимум ошибки $E_{min} = 0.0001$, шаг обучения $a = 0.03$. Размерность обучающей выборки $L = 40$ со значениями, равномерно расположенными на отрезке $[0, 2\pi]$. Результаты: Таб. 3

Таблица 3. Число итераций и ошибка на контрольной выборке при прогнозировании временного ряда $y = 0.1x + 0.1\sin(x) + 0.05\cos(30x)$ в зависимости от параметров

		Число нейронов промежуточного слоя p				
		3	6	9	12	15
Ширина окна n	3	»2000 »2000	»2000 »2000	»2000 »2000	»2000 »2000	»2000 »2000
	6	»2000 687(0.00005)	»2000 703(0.0002)	»2000 636(0.0001)	»2000 786(0.00002)	»2000 653(0.0002)
	8	»2000 720(0.00005)	»2000 827(0.0002)	»2000 811(0.0001)	»2000 825(0.00002)	»2000 831(0.0002)
	9	»2000 675(0.0003)	»2000 626(0.0001)	»2000 685(0.0002)	»2000 744(0.0004)	»2000 793(0.0002)
	12	2591(0.0001) 392(0.00086)	2432(0.0002) 412(0.00069)	2260(0.0004) 424(0.0002)	2134(0.0001) 489(0.0008)	2128(0.0005) 471(0.0006)
	15	2337(0.0001) 343(0.0004)	1994(0.00004) 325(0.0006)	2052(0.00003) 334(0.0005)	2026(0.00003) 358(0.0006)	2150(0.00005) 382(0.0006)
	21	4057(0.00001) 539(0.0004)	2929(0.00003) 501(0.0001)	2972(0.00003) 548(0.0001)	2972(0.00005) 529(0.00015)	2999(0.00005) 567(0.0002)

Проиллюстрируем данные в таблице на графиках зависимости числа итераций от числа нейронов в промежуточном слое для каждой ширины окна (рис. 14).

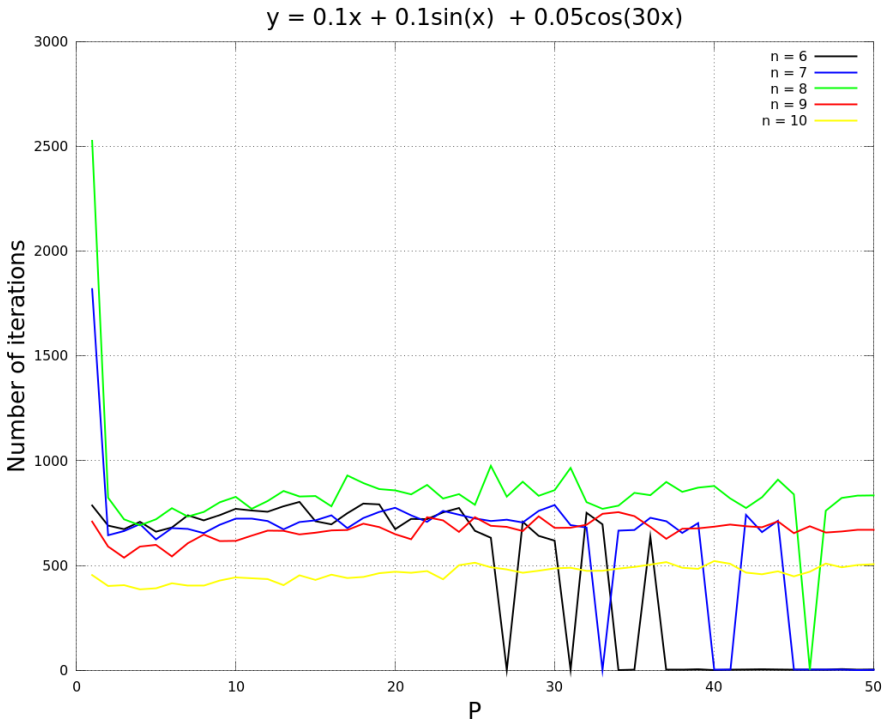


Рис. 14. Результаты для функции $y = 0.1x + 0.1\sin(x) + 0.05\cos(30x)$ (ширина окна 6 - 10).

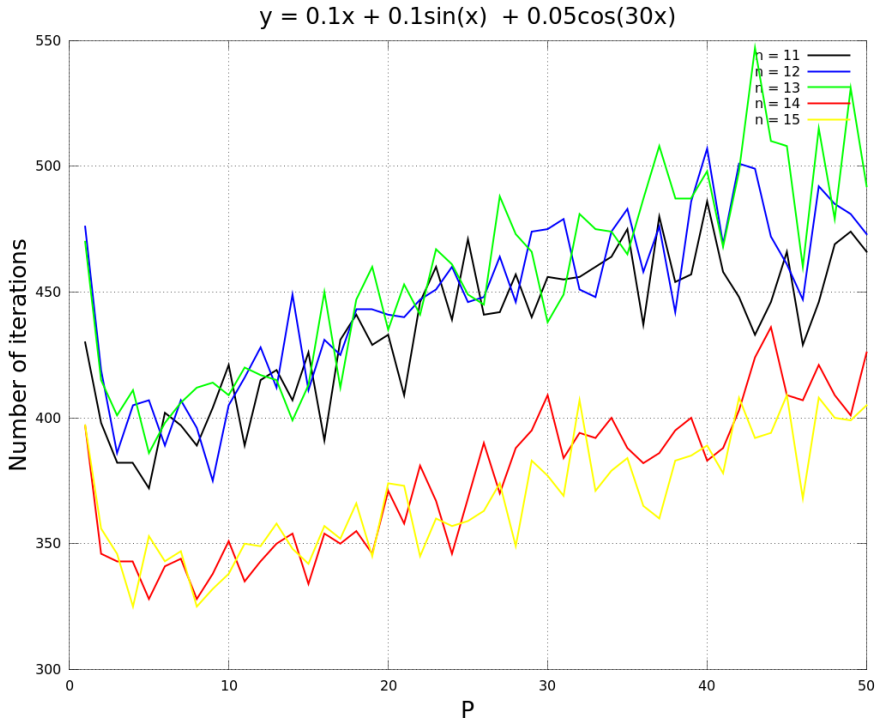


Рис. 15. Результаты для функции $y = 0.1x + 0.1\sin(x) + 0.05\cos(30x)$ (ширина окна 11 - 15).

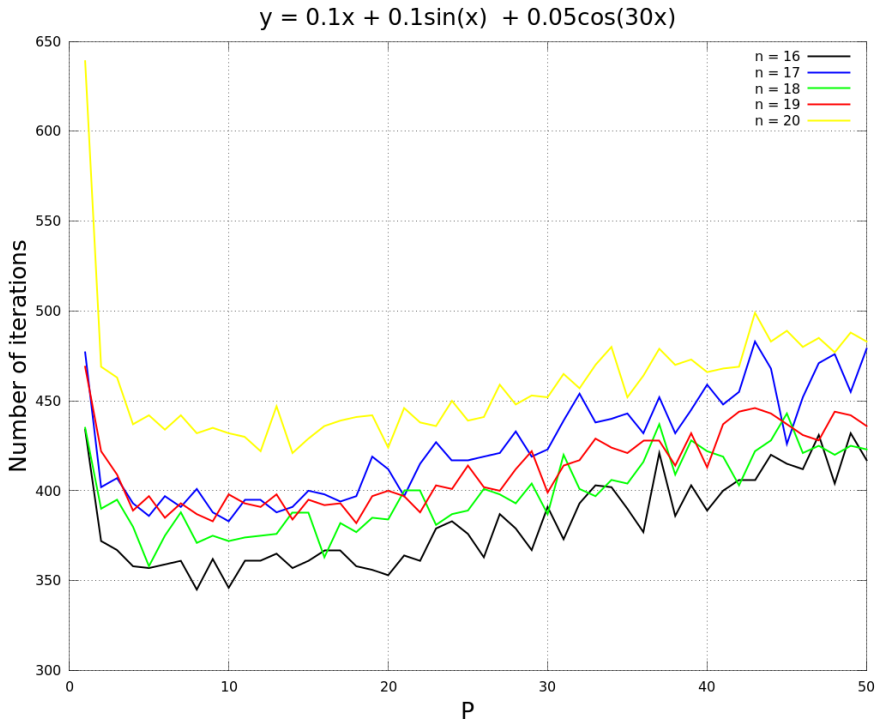


Рис. 16. Результаты для функции $y = 0.1x + 0.1\sin(x) + 0.05\cos(30x)$ (ширина окна 16 - 20).

На графиках 14,15,16 видно, что при увеличении числа нейронов в промежуточном слое число итераций сначала убывало, затем возрастало. Результаты алгоритма при $n = 15$, $p = 6$, $F(S) = \frac{1-e^{-2S}}{1+e^{-2S}}$ и числе выходных нейронов 5 — Рис. 17.

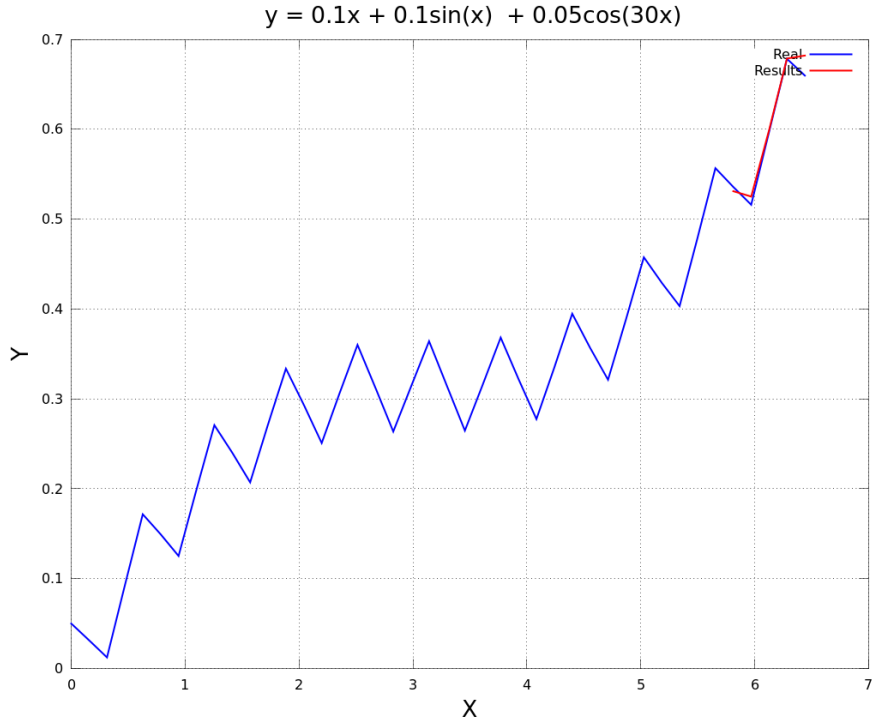


Рис. 17. Результаты алгоритма для функции y = 0.1x + 0.1sin(x) + 0.05cos(30x) (8600 итераций).

Для лучших (в смысле скорости сходимости) параметров каждой функции применим обобщенное дельта-правило с параметром $\eta = 0.2$ и сравним с первыми результатами (Таб. 4).

Таблица 4. Число итераций при использовании обобщенного дельта-правила в обучении сети.

	первые результаты	обобщенное дельта-правило
$y = 0.4\sin(5x) + 0.5$	8(0.003)	6(0.04)
$y = 0.2\sin(20x) + 0.3\cos(15x) + 0.1\cos(5x) + 0.3\sin(3x)$	91(0.00001)	81(0.0004)
$y = 0.1x + 0.1\sin(x) + 0.05\cos(30x)$	325(0.0006)	276(0.0006)

Заключение

В работе рассмотрена классическая архитектура рекуррентной нейронной сети с обратной связью (сеть Джордана), проанализированы на модельных данных зависимости скорости сходимости алгоритма обратного распространения ошибки от функций активации, числа нейронов промежуточного слоя и ширины окна, исследовано обобщенное дельта-правило. Получены результаты:

— использование в качестве функции активации тангенциальной функции повышает скорость сходимости,
— большая размерность промежуточного слоя повышает скорость сходимости,
— применение обобщенного дельта-правила повышает скорость сходимости,
— не найдено зависимости между скоростью сходимости и шириной окна,
— для каждого временного ряда существует некий оптимальный выбор, максимизирующий скорость сходимости.

Необходимый для повторения вычислительного эксперимента код можно найти на сайте:
`https://mlalgorithms.svn.sourceforge.net/svnroot/mlalgorithms/FNNForecasting/`

Литература

[1] W. S. McCulloch and W. Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[2] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, 1969.

[3] D. Rumelhart, G. Hinton, and R. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[4] С. Хайкин. *Нейронные сети*. Вильямс, 2006.

[5] К. Воронцов. Лекции по искусственным нейронным сетям. page 20, 2009.

[6] David MacKay. *Information Theory, Inference, and Learning Algorithms*. 2003.

[7] В. Головко. *Нейронные сети: обучение, организация и применение*. ИПРЖР, 2001.

Выравнивание временных рядов: прогнозирование с использованием DTW

А. А. Романенко
Московский физико-технический институт, ФУПМ, каф. «Интеллектуальные системы»

Временной ряд — это повсеместно встечающаяся форма представления данных во многих научных дисциплинах. Задача, сопутствующая появлению временных рядов, — сравнение одной последовательности данных с другой. Dynamic time warping (DTW) представляет собой технику эффективного выравнивая временных рядов. Методы DTW используются при распознавании речи, при анализе информации в робототехнике, в промышленности, в медицине и других сферах. Предлагается классический алгоритм DTW и упоминаются его возможные модификации. В работе описывается алгоритм поиска в последовательности подпоследовательности, «больше всего похожей» на данную последовательность. Приведены результаты работы алгоритма.
Ключевые слова: *выравнивание временных рядов, DTW алгоритм, time warping, warping path*

Введение

Временной ряд — это повсеместно встечающаяся форма представления данных во многих научных дисциплинах. Распространенная задача, связанная с временными рядами, это сравнение одной последовательности с другой. Например, прогнозирование цен на акции базируется на сравнивании текущей подпоследовательности ряда с найденной подпоследовательностью ряда, сохраненного в истории [1]. Для распознавания речи, рукописного текста и подписи успешно применяется методы, требующие сравнения двух временных рядов [2, 3]. В некоторых случаях достаточно в качестве расстояния между последовательностями выбрать евклидово расстояние. Но часто сравнение таким простым путем дает ошибочные результаты. Dynamic time warping (DTW) представляет собой технику эффективного выравнивая временных рядов. Методы DTW используются в перечисленных выше отраслях, при анализе информации в робототехнике [4], в медицине [5, 6], в биоинформатике [7] и других сферах.

Далее будет описан классический DTW алгоритм. Он дает неплохие результаты, но конечно же он не позволяет достичь наилучшего выравнивания. Это связано с тем, что этот алгоритм очень чувствителен к искажениям временного ряда по оси Y. Также описан алгоритм поиска в последовательности подпоследовательности, «больше всего похожей» на данную последовательность. Приведены результаты работы алгоритма на искусственных и реальных временных рядах.

Постановка задачи

Классический DTW алгоритм

Пусть даны две последовательности Q и C (временных ряда) длиной n и m соответственно:

$$Q = q_1, q_2, \ldots, q_n,\ C = c_1, c_2, \ldots, c_m.$$

Классический DTW алгоритм по этим последовательностям строит *путь наименьшей стоимости*. Поясним, что это значит.

Определим матрицу $\Omega^{n\times m}$ так, чтобы её элемент (i, j) соответствовал расстоянию между i-ым и j-ым элементами последовательностей Q и C, то есть соответствовал выравниванию между q_i и c_j. Мы будем брать евклидово расстояние:

$$d(q_i, c_j) = (q_i - c_j)^2.$$

В качестве метрики можно взять и другие функции, например:

$$d(q_i, c_j) = |q_i - c_j|.$$

По матрице Ω построим некоторый *путь* W. Этот путь выражает соответствие между Q и C. k-ый элемент W определяется как $w_k = (i, j)$. Далее под $d(w_k)$, где $w_k = (i, j)_k$, будем понимать $d(q_i, c_j)$, т. е.

$$d(w_k) = d(q_i, c_j) = (q_i - c_j)^2.$$

Итак, мы имеем

$$W = w_1, w_2, \ldots, w_k, \ldots, w_K,$$

где K — длина пути. K очевидно удовлетворяет следующему условию:

$$\min(m,n) \leqslant K < m+n-1.$$

Пусть путь W удовлетворяет следующим условиям:

— Граничные условия
Обычно предполагают, что $w_1 = (1,1)$ и $w_K = (n,m)$, т. е. начало и конец W находятся на диагонали в противоположных углах Ω.
— Непрерывность
Пусть $w_k = (a,b)$ и $w_{k-1} = (p,q)$. Тогда

$$a - p \leqslant 1, b - q \leqslant 1$$

Это ограничение нужно, чтобы в шаге пути W участвовали только соседние элементы матрицы (включая соседние по диагонали).
— Монотонность
Пусть $w_k = (a,b)$ и $w_{k-1} = (p,q)$. Тогда

$$a - p \geqslant 1, b - q \geqslant 1$$

Это ограничение нужно, чтобы точки W монотонно перемещались во времени.

Путей, удовлетворяющих этим трем условиям, может быть очень много. Однако нам нужен путь, на котором достигается минимум *стоимости пути*:

$$DTW(Q,C) = \min\left\{\frac{1}{K}\sqrt{\sum_{k=1}^{K} d(w_k)}\right\}.$$

Знаменатель K нужен для того, чтобы учесть различную длину W.

Таким образом, *путь наименьшей стоимости (выравнивающий путь)* для последовательностей Q и C это путь W, на котором достигается минимум стоимости пути $DTW(Q,C)$.
Классичесский DTW алгоритм поиска пути минимальной стоимости рекурсивно находит длину пути наименьшей стоимости $\gamma_{i,j}$ до каждого элемента матрицы Ω :

$$\gamma_{i,j} = d(w_{i,j}) + \min(\gamma_{i,j-1}, \gamma_{i-1,j}, \gamma_{i-1,j-1})$$

О других способах вычисления $\gamma_{i,j}$ можно изнать из [8, 9, 10].
Заметим, что евклидово расстояние между Q и C — это частный случай алгоритма DTW, когда путь наименьшей стоимости W ограничен условием

$$w_k = (i,j)_k,\ i = j = k.$$

Постановка задачи
Теперь пусть даны две последовательности Q и C длиной n и m соответственно ($n \gg m$):

$$Q = q_1, q_2, \ldots, q_n,\ C = c_1, c_2, \ldots, c_m.$$

Требуется, используя классический DTW алгоритм, найти такую подпоследовательность Q' последовательности Q, которая «больше всего похожа» на C, т. е. ту подпоследовательность, на которой достигается значение функции

$$DTW(Q',C) = \min_{Q',W}\left\{\frac{1}{K}\sqrt{\sum_{k=1}^{K} d(w_k)}\right\}. \quad (1)$$

Описание алгоритма поиска «похожей» подпоследовательности

Идея алгоритма состоит в том, чтобы с помощью первого применения классического DTW алгоритма отсеять подпоследовательности, которые не могут удовлетворить (1). А затем обычным перебором из

оставшихся подпоследовательностей выбрать оптимальную с помощью классического DTW алгоритма. Код алгоритма с более подробным описанием можно взять из [11].

Вычислительный эксперимент

Алгоритм тестировался как на искусственных временных рядах, так и на реальных.

Поиск подпоследовательности в синусоиде

Возьмем в качестве Q синусоиду на отрезке $[0; \pi]$, длина Q: $n = 1001$. А в качестве C возьмем ее подпоследовательность длины $m = 350$. С помощью нашего алгоритма найдем подпоследовательность Q' и путь W,на которых выполняется (1). (Рис. 1–3)

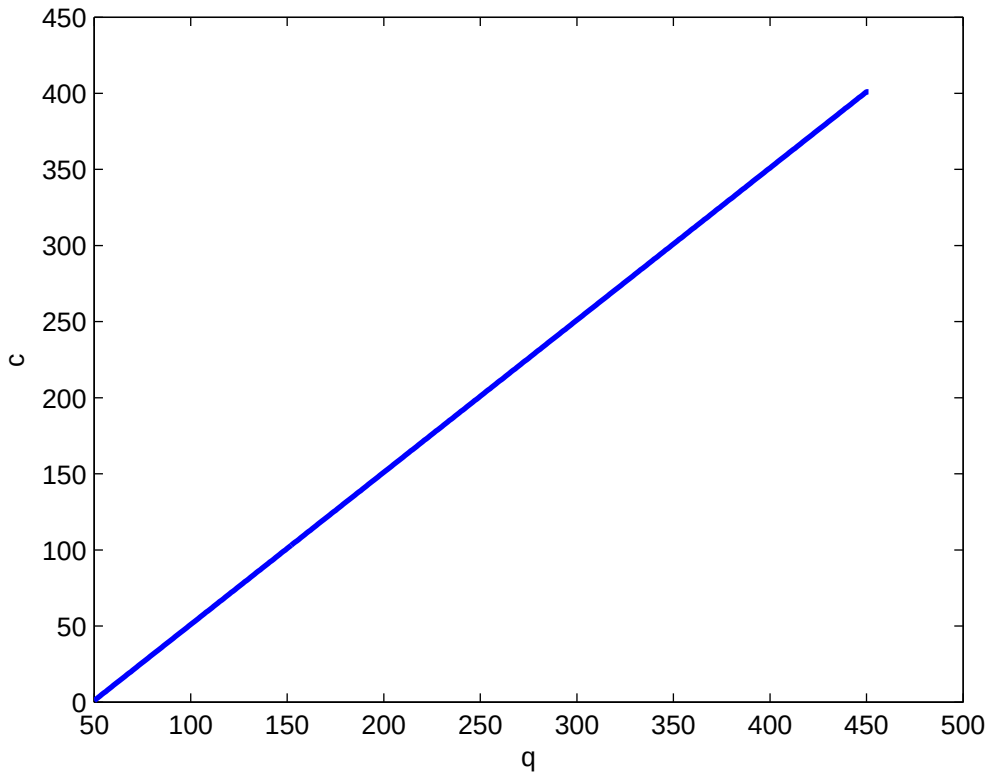


Рис. 1. Выравнивающий путь W.

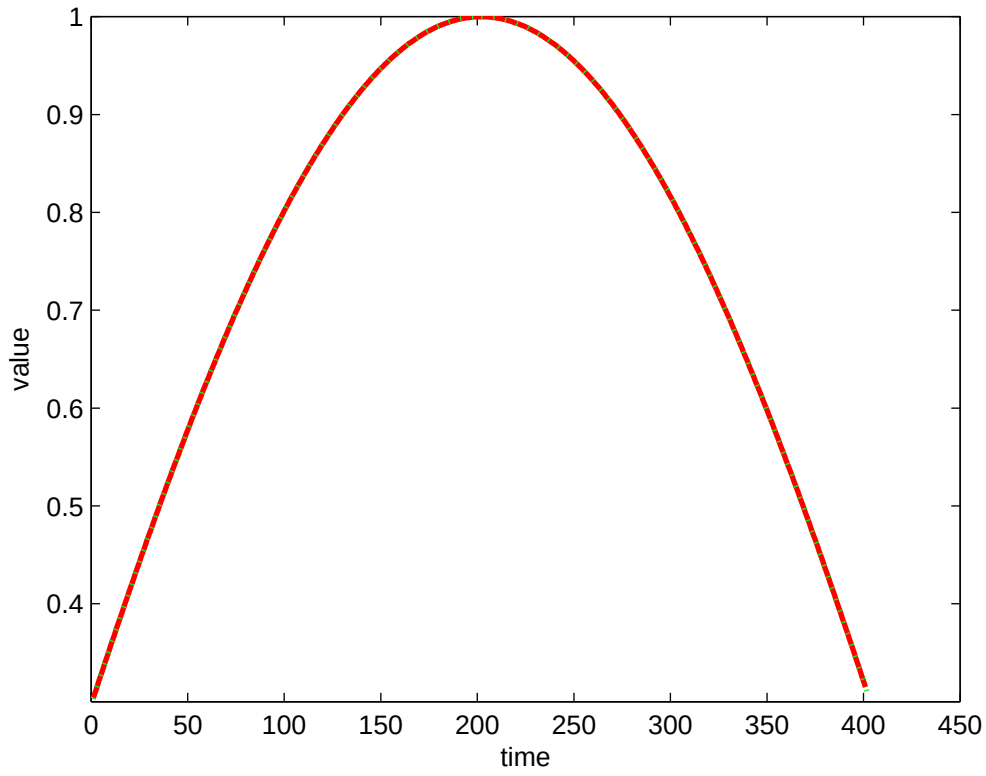


Рис. 2. Последовательность C (красным цветом) и найденная подпоследовательность Q' (зеленым цветом).

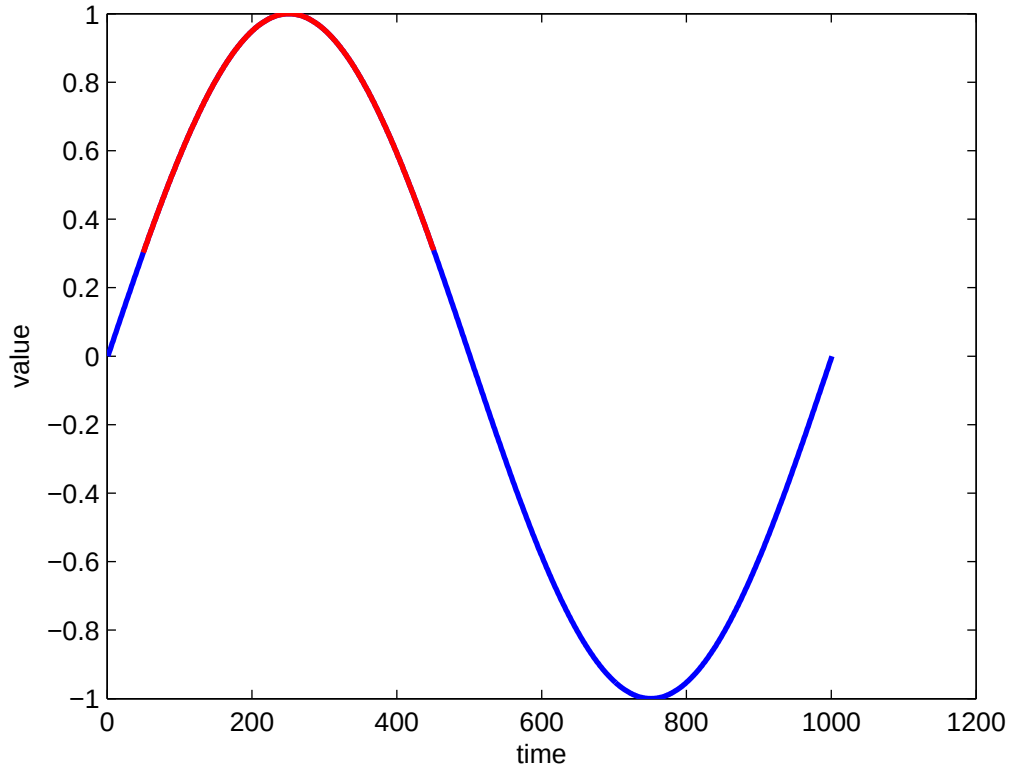


Рис. 3. Последовательность Q и подпоследовательность Q' в ней.

Как видно из графиков, алгоритм нашел оптимальную подпоследовательность Q' и на простейших модельных данных работает правильно.

Поиск подпоследовательности в зашумленной синусоиде Проведем тест подобный предыдущему, только в качестве Q возьмем зашумленную синусоиду, а в качестве C — часть гладкой синусоиды. (Рис. 4–6)

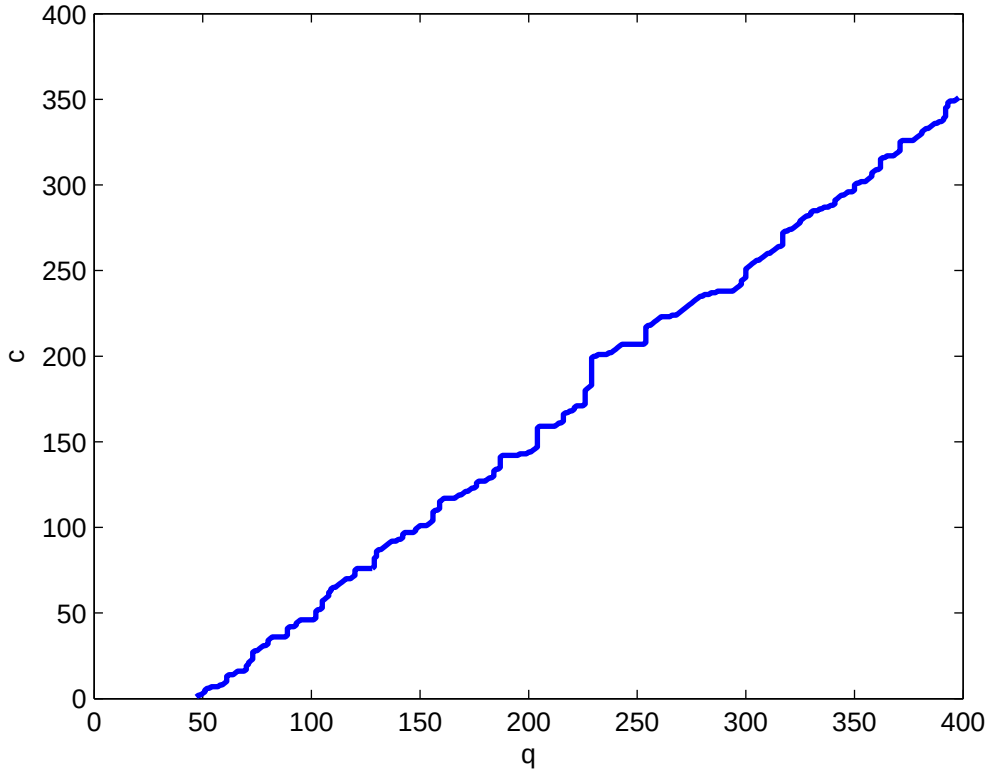


Рис. 4. Выравнивающий путь W.

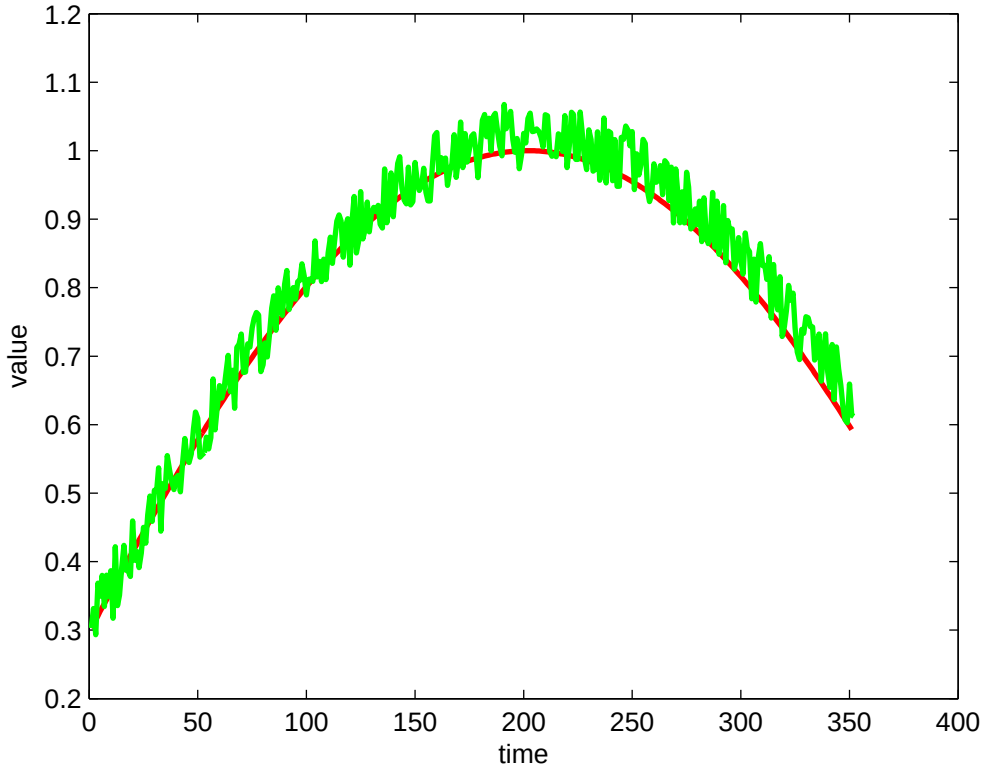


Рис. 5. Последовательность C (красным цветом) и найденная подпоследовательность Q' (зеленым цветом).

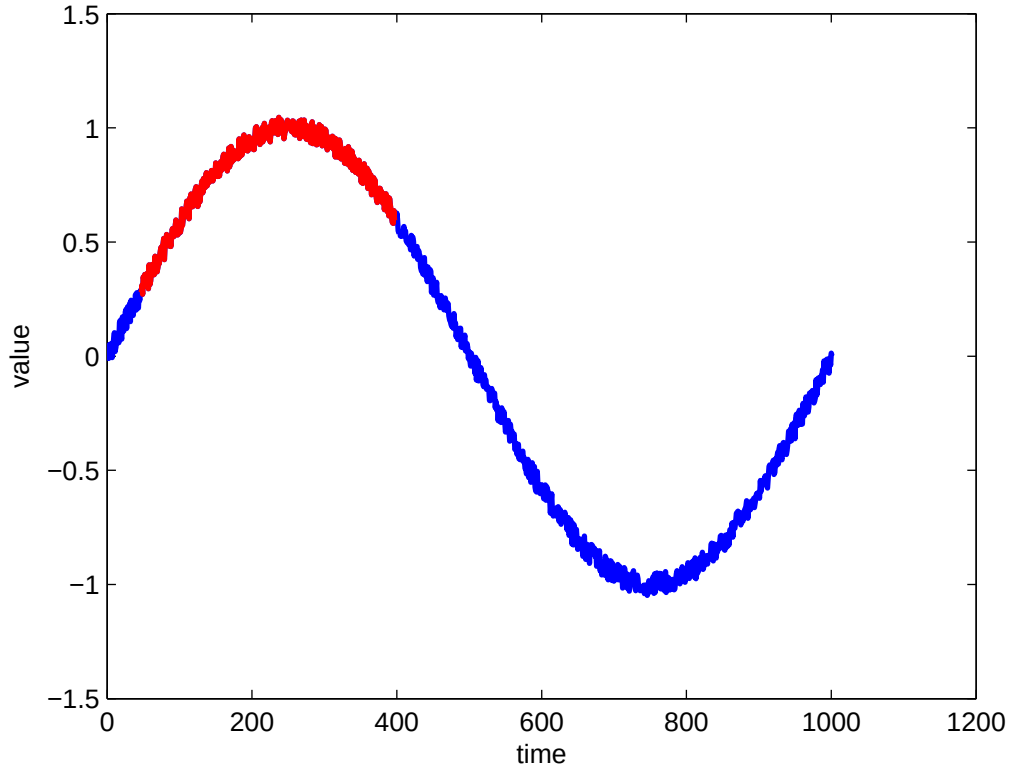


Рис. 6. Последовательность Q и подпоследовательность Q' в ней.

Видно, что алгоритм работает правильно, подпоследовательность найдена верно.
Стоит отметить, что эксперимент проводился при 5% шуме. При больших шумах алгоритм прекращал работать.

Работа на реальных данных

Далее предоставлены результаты работы алгоритма на реальных данных. В качестве Q возьмем зависимость продаж некого товара от времени, в качестве C небольшую последовательность, не являющуюся подпоследовательностью Q. (Рис. 7–9)

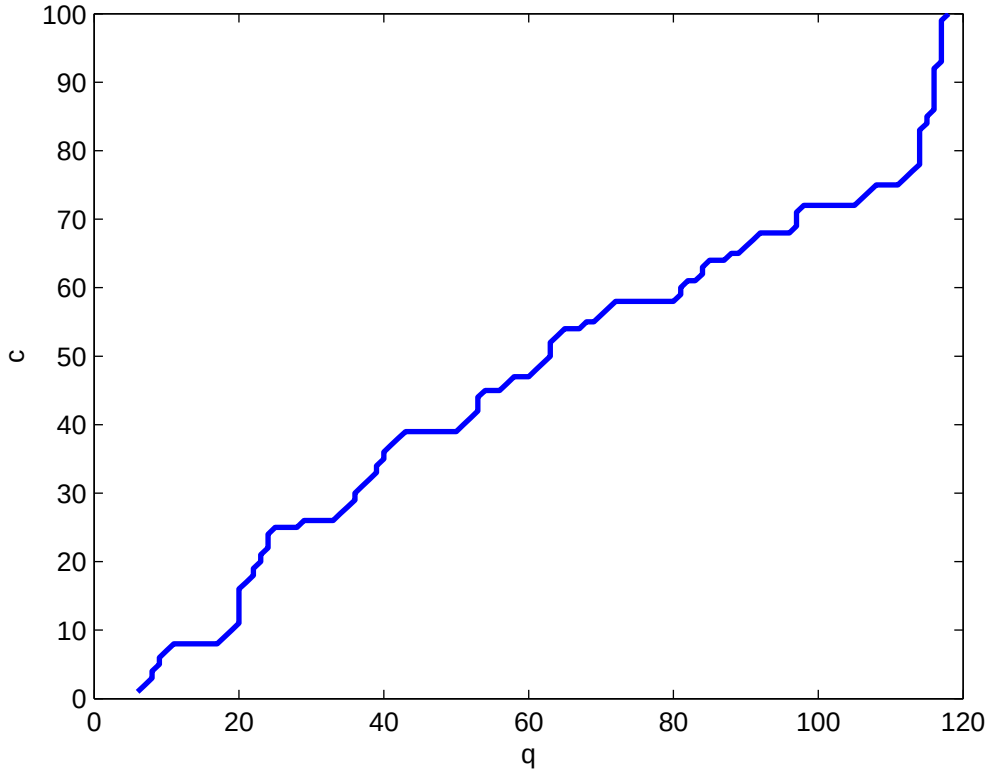


Рис. 7. Выравнивающий путь W.

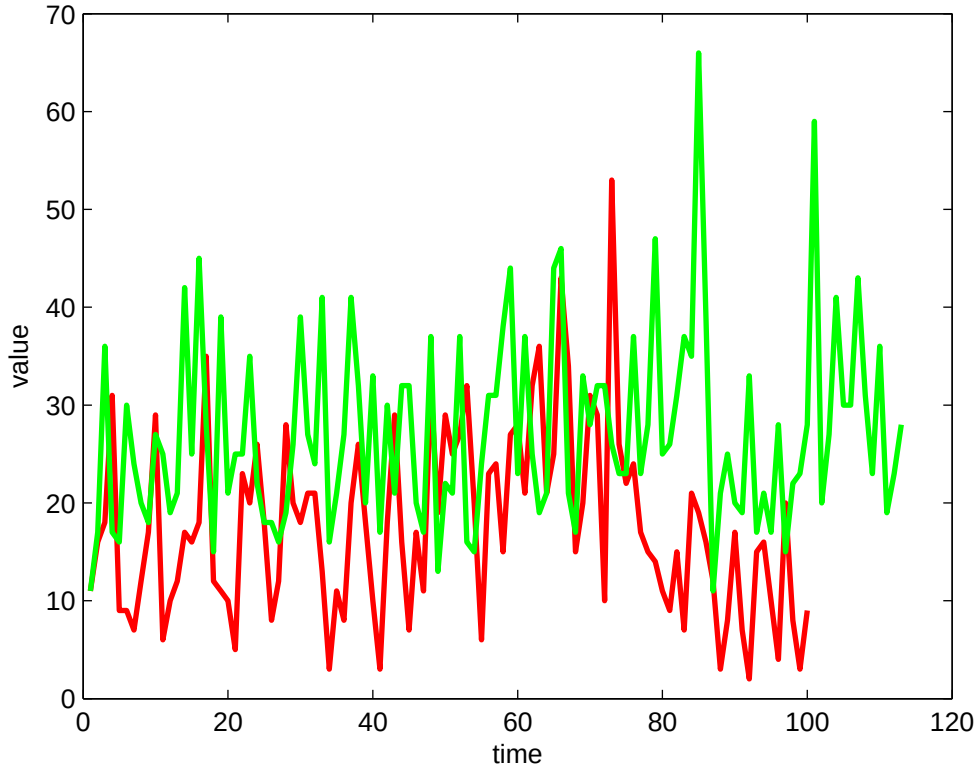


Рис. 8. Последовательность C (красным цветом) и найденная подпоследовательность Q' (зеленым цветом).

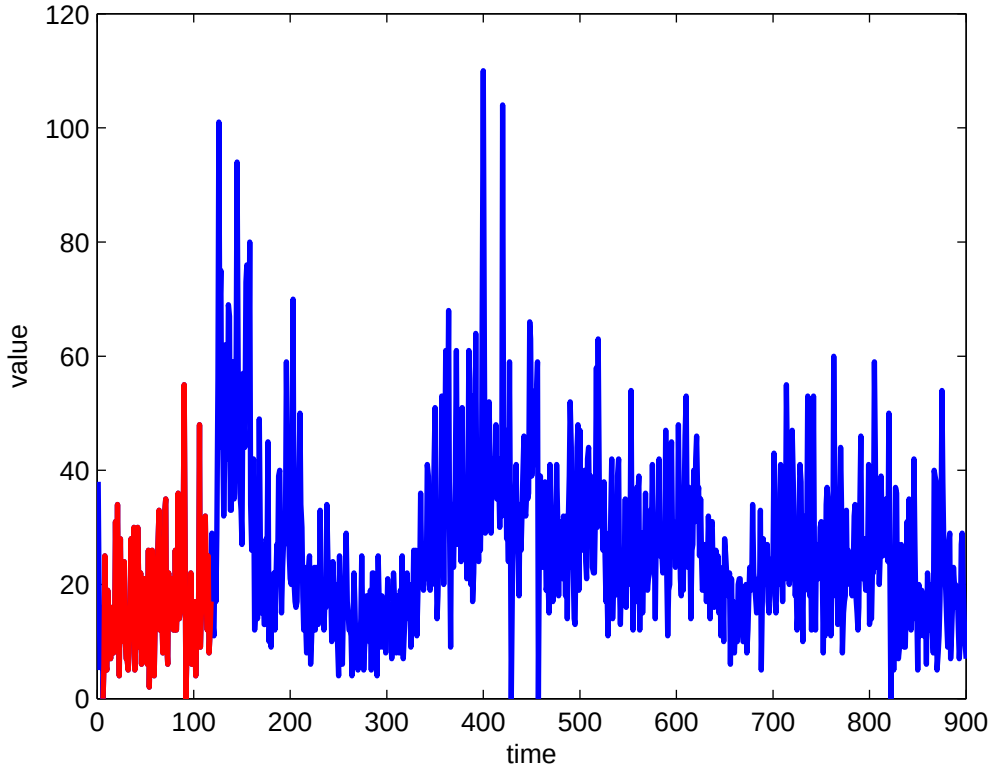


Рис. 9. Последовательность Q и подпоследовательность Q' в ней.

На Рис. 7 конец выравнивающего пути практически вертикален. Это означает, что хвост последовательности C нужно немного растянуть по оси времени, что подтверждает Рис. 8.

Заключение

В работе рассматривается классический DTW алгоритм. Об его улучшениях и модификациях можно узнать из [12, 8, 9, 10, 13]. Также рассмотрим алгоритм поиска «похожей» подпоследовательности, основывающийся на классическом DTW алгоритме. Приведены примеры работы этого алгоритма.

Литература

[1] *Banavas G. N., Denham S., Denham M. J.* Fast nonlinear deterministic forecasting of segmented stock indices using pattern matching and embedding techniques: Computing in Economics and Finance 2000 64: Society for Computational Economics, 2000.

[2] *Niels R.* Dynamic time warping: An intuitive way of handwriting recognition? — 2004.

[3] *Sakoe H., Chiba S.* Readings in speech recognition / Ed. by A. Waibel, K.-F. Lee. — San Francisco, CA, USA, 1990. — Pp. 159–165.

[4] *Oates T., Schmill M. D., Cohen P. R.* A method for clustering the experiences of a mobile robot that accords with human judgments // Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence. — AAAI Press, 2000.

[5] *Vullings H. J. L. M., Verhaegen M. H. G., Verbruggen H. B.* Ecg segmentation using time-warping // Proceedings of the Second International Symposium on Advances in Intelligent Data Analysis, Reasoning about Data. — IDA '97. — London, UK: Springer-Verlag, 1997. — Pp. 275–285.

[6] Warped-average template technique to track on a cycle-by-cycle basis the cardiac filling phases on left ventricular volume. / E. Caiani, A. Porta, G. Turiel et al. // IEEE Computers in Cardiology. — Vol. 25. — 1998.

[7] *Aach J., Church G. M.* Aligning gene expression time series with time warping algorithms. — 2001.

[8] *Keogh E. J., Pazzani M. J.* Derivative dynamic time warping // In First SIAM International Conference on Data Mining (SDM'2001). — 2001.

[9] *Chotirat A., Eamonn R., Keogh.* Everything you know about dynamic time warping is wrong. — 2004.

[10] *Keogh E. J., Pazzani M. J.* Scaling up dynamic time warping to massive datasets. — Springer, 1999. — Pp. 1–11.

[11] *Romanenko A.* https://mlalgorithms.svn.sourceforge.net/svnroot/mlalgorithms/dtwforecasting.

[12] *Berndt D. J., Clifford J.* Using dynamic time warping to find patterns in time series // KDD Workshop. — 1994. — Pp. 359–370.

[13] Iterative deepening dynamic time warping for time series / S. Chu, E. Keogh, D. Hart, M. Pazzani // In Proc 2 nd SIAM International Conference on Data Mining. — 2002.

Прогнозирование функциями дискретного аргумента

Е. А. Будников
Московский физико-технический институт, ФУПМ, каф. «Интеллектуальные системы»

В работе исследуются короткие временные ряды на примере монофонических музыкальных мелодий. Происходит прогнозирование одной ноты экспоненциальным сглаживанием, локальным методом, а также методом поиска постоянных закономерностей.
Вычислительный эксперимент проводится на двух мелодиях, одна из которых имеет точно повторяющиеся фрагменты.
Ключевые слова: *временной ряд, прогнозирование мелодий, машинное обучение, прогнозирование функциями дискретного аргумента, локальные методы прогнозирования, поиск постоянных закономерностей.*

Введение

В отчете представлена попытка прогнозирования таких специфических временных рядов, как монофонические мелодии. Были осуществлены три различных подхода: экспоненциальное сглаживание, локальное прогнозирование и поиск постоянных закономерностей. Первый из них — хорошо разработанный метод, описанный в [1, 2]. Второй описан, например, в [9]. Третий предлагается в полной мере в работах [7, 6].

Предлагается опробовать первый метод в традиционной его форме, чтобы ответить на вопрос, пригоден ли он для решения данной задачи. Затем предлагается во втором методе проверить работоспособность коэффициента корреляции Пирсона в качестве меры сходства. Третий будет использоваться в упрощенном варианте.

Постановка задачи

Мелодия есть функция $m: T \to X \times Y$, где $T = 0, 1, 2, ...$ — позиция ноты, $X = 0, 1, 2, ...$ — конечное множество нот, занумерованных в порядке увеличения тона, Y — длительность ноты, в секундах. Таким образом, будем работать с пучком из двух временных рядов.

Предполагается, что мелодия дана законченная, но без нескольких финальных нот(без ограничения общности одной). Необходимо их предсказать.

Пути решения задачи

Экспоненциальное сглаживание
Пусть $X = \{x_1, ...x_T\}$ — временной ряд.
Экспоненциальное сглаживание ряда осуществляется по рекуррентной формуле:

$$S_t = \alpha x_t + (1 - \alpha)\, S_{t-1},\ \alpha \in (0, 1).$$

Чем меньше α, тем в большей степени фильтруются, подавляются колебания исходного ряда и шума.
Если последовательно использовать рекуррентное это соотношение, то экспоненциальную среднюю S_t можно выразить через значения временного ряда X.

$$S_t = \alpha x_t + (1 - \alpha)\,(\alpha x_{t-1} + (1 - \alpha) S_{t-2}) = \cdots = \alpha \sum_{i=0}^{t-1} (1 - \alpha)^i x_{t-i} + (1 - \alpha)^t S_0.$$

После появления работ Р. Брауна экспоненциальное сглаживание часто используется для решения задачи краткосрочного прогнозирования временных рядов следующим способом.
Пусть задан временной ряд: $y_i \cdots y_t,\ y_i \in R$.
Необходимо решить задачу прогнозирования временного ряда, т.е. найти
$\hat{y}_{t+d} = f_{t,d}\,(y_1 ... y_t)\,,\ d \in \{1, 2, ...D\},\ D$ — горизонт прогнозирования, необходимо, чтобы
$Q_T = \sum_{i=1}^{T} (y_i - \hat{y}_i) \to min$
Предположим, что D - невелико (краткосрочный прогноз), то для решения такой задачи используют модель Брауна.
$\hat{y}_{t+d} = \alpha y_t + (1 - \alpha)\hat{y}_t,\ \hat{y}_0 = y_0,\ \alpha \in (0, 1)$.
Если рассматривать прогноз на 1 шаг вперед, то $(y_t - \hat{y}_t)$ — погрешность этого прогноза, а новый прогноз $\hat{y}_{t+1}$ получается в результате корректировки предыдущего прогноза с учетом его ошибки — суть адаптации.

При краткосрочном прогнозировании желательно как можно быстрее отразить новые изменения и в то же время как можно лучше "очистить"ряд от случайных колебаний. Т.о. следует увеличивать вес более свежих наблюдений: $\alpha \to 1,\ \hat{y}_{t+d} \to y_t$.

С другой стороны, для сглаживания случайных отклонений, α нужно уменьшить: $\alpha \to 0,\ \hat{y}_{t+1} \to \bar{y}_t$.

Т.о. эти два требования находятся в противоречии. Мы будем брать α из интервала (0,0.5).

Локальные методы прогнозирования

Музыкальный временной ряд отличается от обычного хаотического: он почти не хаотичен (для специалистов, я думаю, слово "почти" можно убрать). В нем встречаются похожие, повторяющиеся и прочие регулярные структуры.

Определение 1. *Регулярной структурой назовем кусок временного ряда, обладающий автономностью по отношению к остальному временному ряду, склонный к повторению в немного искаженной форме*

Очевидно, что "немного" должно определяться некой функцией близости. В работе использовался вариант коэффициента корреляции Неймана-Пирсона:

$$k(f,g) = \frac{\int fg}{\sqrt{\int f^2} \cdot \sqrt{\int g^2}},$$

где интеграл понимается в смысле суммы в силу дискретности функций.

Прогноз будет строиться на естественном предположении компактности регулярных структур: у похожих кусков временного ряда должны быть похожие продолжения.

Воспользуемся самым простым локальным алгоритмом, который ищет ближайшего соседа к прогнозируемому участку.

Поиск постоянных закономерностей

Рассмотрим один из подходов к поиску закономерностей в пучках временных рядов, который предполагает отсутствие изменений в закономерностях с течением времени. Для простоты будем рассматривать единственный временной ряд длины T вместо пучка.

Маской ω на отрезке назовем булеву строку длины N (здесь параметр N определяет максимальный отступ по времени). Число единиц в маске ω будем называть весом маски и обозначать $H(\omega)$. Элемент маски, находящийся на i-ом месте будем обозначать $\omega(i)$ или ω_i. Закономерностью R назовем пару $(\omega; f)$, где маска ω указывает на значения ряда, являющиеся аргументами функции f, а частично-определенная функция f задает зависимость значений целевого ряда от переменных, на которые указывает маска ω.

$$f : X^{H(\omega)} \to X \cup \{\lambda\},$$

где λ означает, что функция не определена на соответствующем наборе переменных.

Зафиксировав теперь маску $\omega = [1,1,1]$, построим множество пар (α_t, v_t), где $\alpha_t = [m(t), m(t+1), m(t+2)]$, а $v_t = m(t+3)$, $t \in \{1,2,\ldots,T-3\}$.

Полученное множество пар записывается в виде таблицы частот $\|\nu_{\alpha,v}\|$ с числом строк, равным числу всех возможных наборов из $X^{H(\omega)} = x^3$, и числом столбцов, равным $|X|$. Элемент таблицы частот $\|\nu_{\alpha,v}\|$ $(0 \leqslant \alpha \leqslant |X|^3 - 1,\ 0 \leqslant v \leqslant |X| - 1)$ — это число раз, которое значение v встречается во входных данных на наборе $\tilde{\alpha}$ с номером α из X^3.

(Предполагается, что наборы расположены в лексикографическом порядке.)

Обозначим $\nu_{\alpha,max} = \max\limits_{v \in \{1,2,\ldots,|X|-1\}} \nu_{\alpha,v}$ и $v_m = \arg \max\limits_{v \in \{1,2,\ldots,|X|-1\}} \nu_{\alpha,v}$ (в случае если максимум достигается на нескольких значениях, v_m выбирается среди этих значений произвольным образом).

Обозначим также $\nu_{\alpha,max-1} = \max\limits_{v \in \{1,2,\ldots,|X|-1\}, v \neq v_m} \nu_{\alpha,v}$ и $\nu_\alpha = \sum_{v=0}^{|X|-1} \nu_{\alpha,v}$.

На основе таблицы частот порождается закономерность $(\omega; f)$, где частично- определенная функция f задается на каждом наборе $\tilde{\alpha}$ из X^3 следующим образом:

$$f(\tilde{\alpha}) = \begin{cases} v_m, & \text{если } \nu_{\alpha,max} - \nu_{\alpha,max-1} \geqslant k \cdot \nu_\alpha \\ \lambda, & \text{иначе} \end{cases}$$

Здесь символ λ обозначает отсутствие значения на данном наборе, а k — параметр алгоритма, $0 < k < 1$.

Вычислительный эксперимент

Экспоненциальное сглаживание

В качестве исследуемой мелодии сначала был взят файл Гуси.mid.

Для экспоненциального сглаживания была выбрана α, при которой среднеквадратичное отклонение ряда от его сглаживания было минимальным.

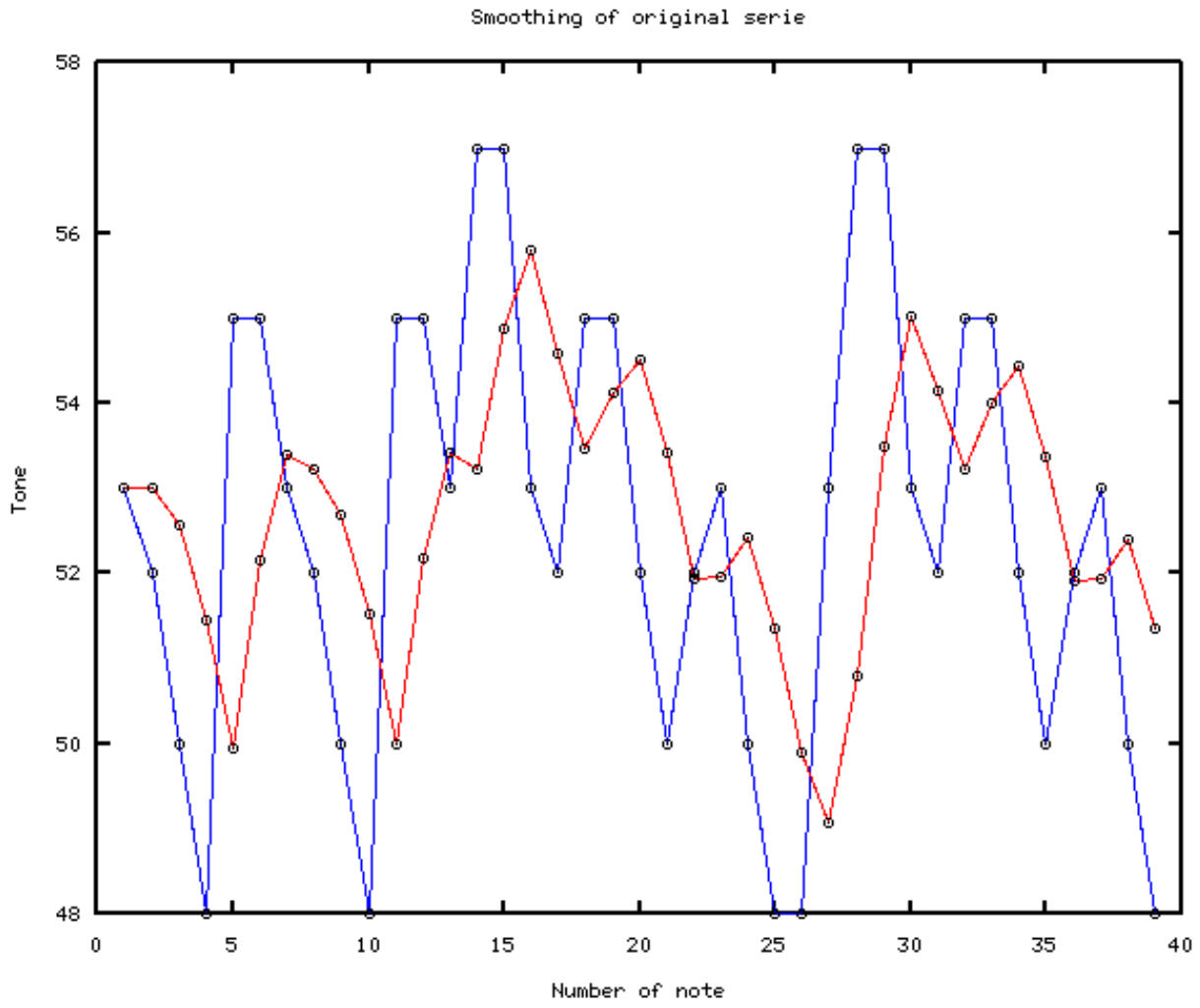


Рис. 1. Изначальный временной ряд Гуси.mid и его экспоненциальное сглаживание

Это значение оказалось равным $\alpha = 0.4361$.

Ну и рисунок с предсказанной последней нотой:

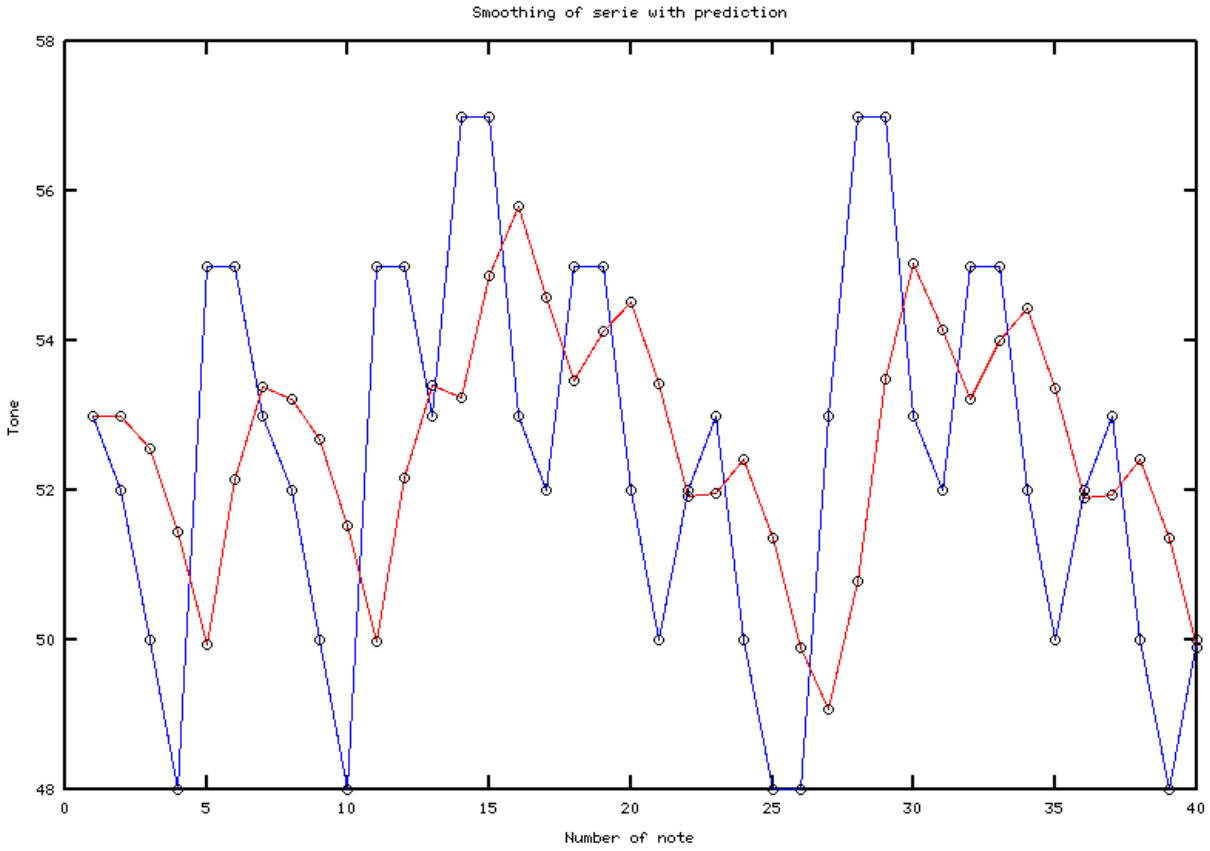


Рис. 2. Временной ряд с предсказанной точкой

Теперь исследуем другую мелодию, еще более короткую, китайское.mid

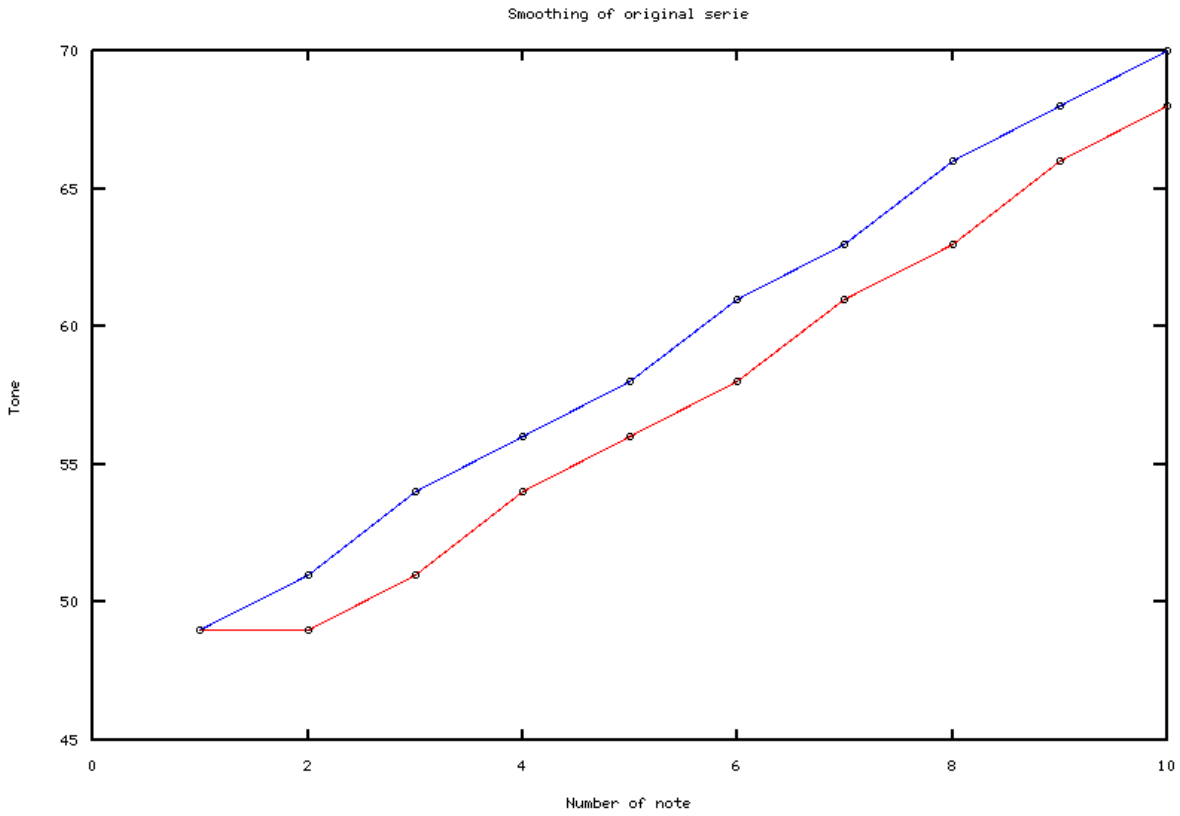


Рис. 3. Изначальный временной ряд китайское.mid и его экспоненциальное сглаживание

Значение параметра здесь оказалось равным $\alpha = 1$. И все равно экспоненциальное сглаживание не может "догнать" ряд.

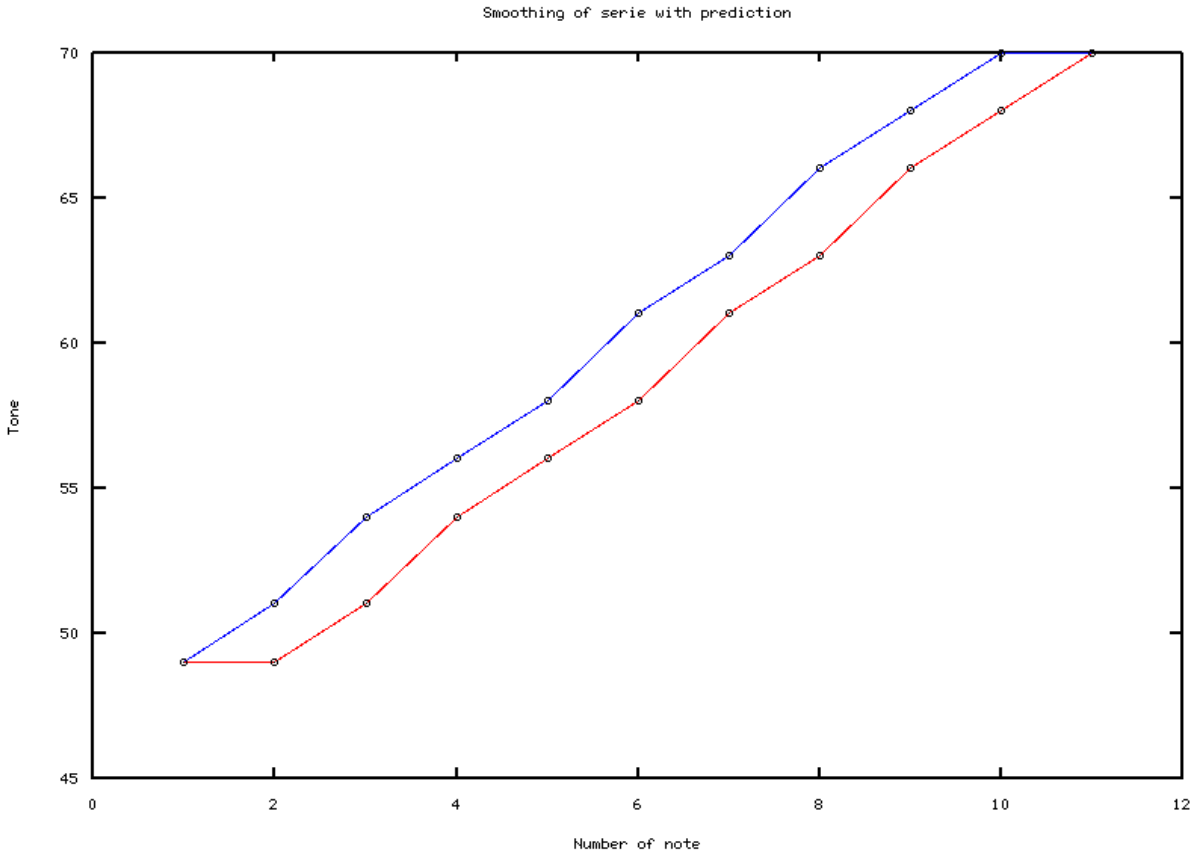


Рис. 4. Временной ряд с предсказанной точкой

Локальное прогнозирование

Локальное прогнозирование оказалось успешней. В случае первой мелодии оно не только правильно нашло похожий кусок мелодии и правильно предсказало финальную ноту, но и в случае многократной прогонки алгоритма повторяло последнее предложение мелодии. Результаты работы алгоритма представлены в файлах out1.mid и out2.mid.

Поиск постоянных закономерностей

В качестве исследуемой мелодии снова был взят файл Гуси.mid. Метод оказался удачным. Результаты работы алгоритма представлены в файле out3.mid.

Заключение

В заключение можно сказать, что проделана совсем небольшая работа, но уже точно можно сказать, что экспоненциальное сглаживание не может быть использовано для успешного решения поставленной задачи. Локальные же методы представляются более перспективными в том плане, что на мелодиях, обладающих некими похожими повторяющимися фрагментами, есть смысл использовать именно такие методы. К сожалению, в работе были проделаны исследования лишь с одной метрикой, имеет смысл исследовать поведение других метрик.

Метод поиска постоянных закономерностей оказался также удачным. Тем не менее, я склонен рассматривать это следствием того, что мелодия имела точно повторяющиеся куски. То есть для прогноза был найден идентичный кусок, ранее встречавшийся в мелодии, а в качестве прогноза была просто взята следующая за упомянутым куском нота. В силу малой длины мелодии построенная функция оказалась крайне мало определенной на множестве всевозможных наборов из X^3. Возможным выходом из данной ситуации, кроме очевидного увеличения длины известной части мелодии, я полагаю выделение при помощи эксперта либо методами кластеризации примитивных элементов мелодии, дальнейшее ее кодирование в терминах этих примитивов. Этот шаг позволит снизить мощность множества допустимых значений временного ряда, а следовательно и множества всевозможных наборов из него же. И искомая частично-

определенная функцая будет определен плотнее.

Необходимый для повторения вычислительного эксперимента код можно найти на сайте:
`https://mlalgorithms.svn.sourceforge.net/svnroot/mlalgorithms/DiscreteForecasting/`

Литература

[1] *Brown, R.G.*. Smoothing forecasting and prediction of discrete time series / Brown, R.G.—N.Y., 1963.

[2] *Brown, R.G. and Meyer, R.F.*. The fundamental theorem of exponential smoothing / Brown, R.G. and Meyer, R.F.—Oper. Res., 1961.

[3] *Kohonen, T.*. The Self-Organising Map / T. Kohonen.—Proceedings of the IEEE, 1990.

[4] *Бокс Дж., Дженкинс Г.*. Анализ временных рядов, прогноз и управление. Том 1 / Бокс Дж., Дженкинс Г.—1969.

[5] *Е.М.Четыркин*. Статические методы прогнозирования / Е.М.Четыркин.—Издательство "Статистика 1977.

[6] *Николай Филипенков*. Об алгоритмах прогнозирования процессов с плавно меняющимися закономерностями / Николай Филипенков. –2010.

[7] *Николай Филипенков*. О задачах анализа пучков временных рядов с изменяющимися закономерностями / Николай Филипенков.—2006.

[8] *Ю.П.Лукашин*. Адаптивные методы краткосрочного прогнозирования временных рядов / Ю.П.Лукашин.—М.: Финансы и статистика, 2003.

[9] *James McNames*. Local Modeling Optimization for Time Series Prediction / James McNames.—European Symposium on Artificial Neural Networks Bruges (Belgium). D-Facto public, 2000.

Многомерная авторегрессия

Р. Б. Джамтырова

Аннотация

Многомерная авторегрессия является эконометрической моделью, которую используют для пронозирования временными рядами. Многомерная авторегрессия является обобщением модели авторегрессии (AR).

Постановка задачи

Даны временной ряд $\mathbf{s}_1 = [x_1, \ldots, x_{T-1}]^T$, $x_i \in \mathbb{R}^1$ и матрица признаков, столбцами которой являются временные ряды $\mathbf{s}_2, \mathbf{s}_3 \cdots \mathbf{s}_m$. Необходимо спрогнозировать следующую величину x_T ряда $\mathbf{s}_1$.

Предполагается, что

— отсчеты сделаны через равные промежутки времени,
— ряд имеет периодическую составляющую,
— ряд не имеет пропущенных значений,
— длина ряда кратна периоду k.

Составляется $(m \times k)$ -матрица значений временного ряда:

$$S = \left(\begin{array}{c|ccc} x_T & x_{T-1} & \ldots & x_{T-k+1} \\ \hline x_{(m-1)k} & x_{(m-1)k-1} & \ldots & x_{(m-2)k+1} \\ \ldots & \ldots & \ldots & \ldots \\ x_{nk} & x_{nk-1} & \ldots & x_{n(k-1)+1} \\ \ldots & \ldots & \ldots & \ldots \\ x_k & x_{k-1} & \ldots & x_1 \end{array}\right).$$

Введем обозначения

$$\left(\begin{array}{c|c} x_T & \mathbf{x}^T \\ \hline \mathbf{y} & X \end{array}\right).$$

В случае, когда учитываются временные ряды $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_m$, для каждого j-го временного ряда строится авторегрессионная матрица S_j и присоединяется справа. Полученная матрица

$$A = \begin{bmatrix} S_1 & S_2 & \cdots & S_m \end{bmatrix}.$$

Пусть задан набор функций $G = \{g_1, \ldots, g_r\}$, например, $g_1 = 1$, $g_2 = \sqrt{x}$, $g_3 = x$, $g_4 = x\sqrt{x}$. Матрица порождённых признаков имеет вид

$$A = \left(\begin{array}{c|ccccccc} x_T & g_1 \circ x_{T-1} & \ldots & g_r \circ x_{T-1} & \ldots & g_1 \circ x_{T-k+1} & \ldots & g_r \circ x_{T-k+1} \\ \hline x_{(m-1)k} & g_1 \circ x_{(m-1)k-1} & \ldots & g_r \circ x_{(m-1)k-1} & \ldots & g_1 \circ x_{(m-2)k+1} & \ldots & g_r \circ x_{(m-2)k+1} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ x_{nk} & g_1 \circ x_{nk-1} & \ldots & g_r \circ x_{nk-1} & \ldots & g_1 \circ x_{n(k-1)+1} & \ldots & g_r \circ x_{n(k-1)+1} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ x_k & g_1 \circ x_{k-1} & \ldots & g_r \circ x_{k-1} & \ldots & g_1 \circ x_1 & \ldots & g_r \circ x_1 \end{array}\right).$$

Пути решения задачи Требуется решить задачу линейной регрессии $\|X\mathbf{w} - \mathbf{y}\|^2 \longrightarrow \min$. Искомый вектор параметров имеет вид:

$$\mathbf{w} = (X^T X)^{-1}(X^T \mathbf{y}).$$

В терминах линейной регрессии

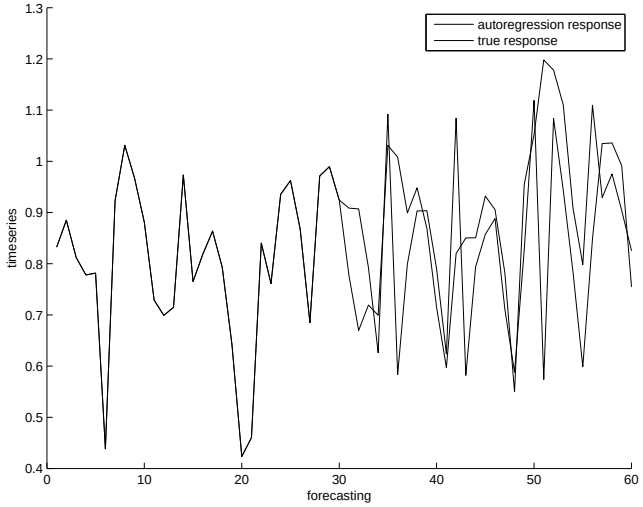
$$\mathbf{y} = X\mathbf{w}$$

$$y_T = < \mathbf{x}^T, \mathbf{w} >$$

Возможны два подхода к нахождению значения $\mathbf{x}_T$. Первый заключается в последовательном построении прогнозируемых значений. В качестве вектора $\mathbf{x}^T$ выбираются $K - 1$ предыдущих значений временного ряда, и для каждого следующего значения $x_{T+1}, x_{T+2}, ..., x_{T+l}$ необходимо заново строить авторегрессионную матрицу X. Второй подход заключается в прогнозировании периода ряда по истории..

В качестве вектора $\mathbf{x}^T$ выбираются $K-1$ значений ряда предшествующего периода. При данном подходе перестраивать авторегрессионную матрицу X не нужно.

Результат работы алгоритма



Литература

[1] Стрижов В. В. Методы выбора регрессионных моделей, 2010.

[2] V. Strijov. Model Generation and its Applications in Financial Sector, 2009.

Прогнозирование продаж групп товаров

Е. Ю. Зайцев
zaytsev.yevgen@gmail.com
Москва, Вычислительный Центр РАН

Постановка задачи

Заданы временные ряды продаж товаров $x_{ij}(t) \in \mathbb{R}$ — продажи i-го товара в j-том магазине за время t (где $i \in I$, I — множество товаров; $j \in J$, J — множество магазинов; $t \in \mathbb{N}$). Значения продаж известны при $t_0 \leqslant t \leqslant t_1$. Также задан товарный классификатор, который разбивает товары на группы, образующие иерархическую стуктуру.

Требуется спрогнозировать продажу заданного товара в заданном магазине на следующий день после t_1.

Математическое описание

1. Найти все группы нижнего уровня — разбиение множества I на непересекающиеся подмножества $I_k \subset I$.
   Для всех групп нижнего уровня I_k повторять следующие шаги.
2. Найти суммарные продажи товаров из I_k во всех магазинах

$$s_{ij}(w) = \sum_{t=t_1-w+1}^{t1} x_{ij}(t),$$

   где w - размер обучающей выборки.
   Обозначим

$$S_w = \sum_{i \in I_k} \sum_{j \in J} s_{ij}(w).$$

3. Определить доли продаж отдельных товаров из группы

$$D_w(i) = \frac{1}{S_w} \sum_{j \in J} s_{ij}(w).$$

   Повторять для всех $j \in J$ и всех групп нижнего уровня I_k следующие шаги.
4. Вычислить по методу скользящего среднего прогноз продаж товаров из I_k в магазине j на следующий день:

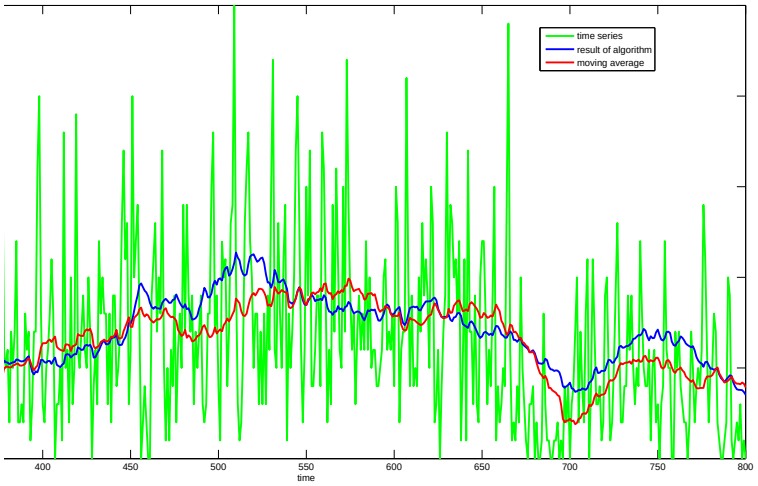
$$S_{vj}(I_k) = \frac{1}{v} \sum_{i \in I_k} s_{ij}(v).$$

5. Распределить предсказанное число товаров согласно величинам $D_w(i)$:

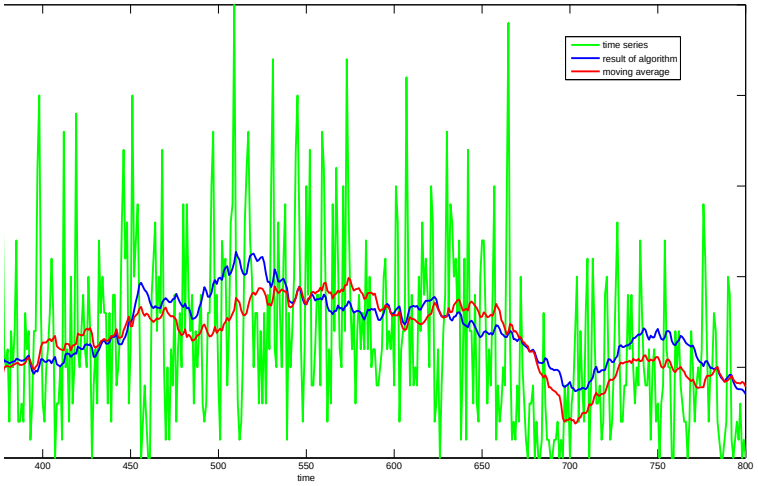
$$\overline{y_{ij}} = S_{vj}(I_k) D_w(i).$$

Результаты

Результат работы алгоритма приведен на графике.



Также применим данный алгоритм не к товарам, а к группам товаров нижнего уровня



На обоих графиках синяя линия прогноза описанного выше алгоритма и красная линия прогноза методом скользящего среднего опережают график продаж (зеленая кривая) на один шаг.

Литература

[1] https://mlalgorithms.svn.sourceforge.net/svnroot/mlalgorithms/tsforecasting/groupforecast/.

Авторегрессионное интегрированное скользящее среднее

Н. П. Ивкин
ivkinnikita@gmail.com
Москва, Вычислительный Центр РАН

Целью проекта является прогноз временного ряда на несколько отсчетов алгоритмом ARIMA (autoregressive integrated moving average).

Постановка задачи

Решается задача прогнозирования временных рядов. Обучающей выборкой является временной ряд, называемый историей. Требутся спрогнозировать дальнейшее поведение некоторого показателя данного временного ряда.

Пути решения задачи

Для нахождения решения предлагается использовать модель ARIMA[1]. Модель ARIMA является обобщением модели авторегрессионного скользящего среднего (ARMA). Модель ARMA является синтезом авторегрессионной модели и модели скользящего среднего.

Пусть задан временной ряд X_t, где t — целое число и X_t — вещественные числа.

Авторегрессионная модель порядка p задается следующим образом:

$$X_t = c + \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t, \tag{1}$$

где φ_i — параметры модели, c — константа, ε_t — белый шум.

Модель скользящего среднего порядка q задается следующим образом:

$$X_t = \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t, \tag{2}$$

где θ_i — параметры модели и $\varepsilon_t \propto N(0, \sigma^2)$.

Модель ARMA(p,q) задается следующим образом:

$$X_t = \sum_{i=1}^{p} \varphi_i X_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + c + \varepsilon_t. \tag{3}$$

Модель ARMA работает в предположение, что ряд стационарен. Приводить ряд к стационарному предлагается методом взятия последовательной разности. Данный метод не обязательно приводит исходный ряд к стационарному, но неплохо решает проблему наличия тренда. Метод заключается в построение нового ряда Z_t такого, что:

$$Z_t = \Delta X_t = X_t - X_{t-1}. \tag{4}$$

Модель ARIMA(p,q,d) заключается в приведение ряда к стационарному виду взятием последовательной разности d раз и в применение к новому временному ряду модели ARMA(q,d).

Определение параметров p,q,d

Для определения параметра d будем брать последовательную разность до тех пор, пока тест стационарности Дики-Фуллера [1] не опровергнет нулевую гипотезу для получившегося временного ряда.

Параметры p и q предлагается находить минимизирующими функционал качества SSE. На практике значения этих параметров редко бывают больше трех.

Результаты работы алгоритма (одномерный случай)

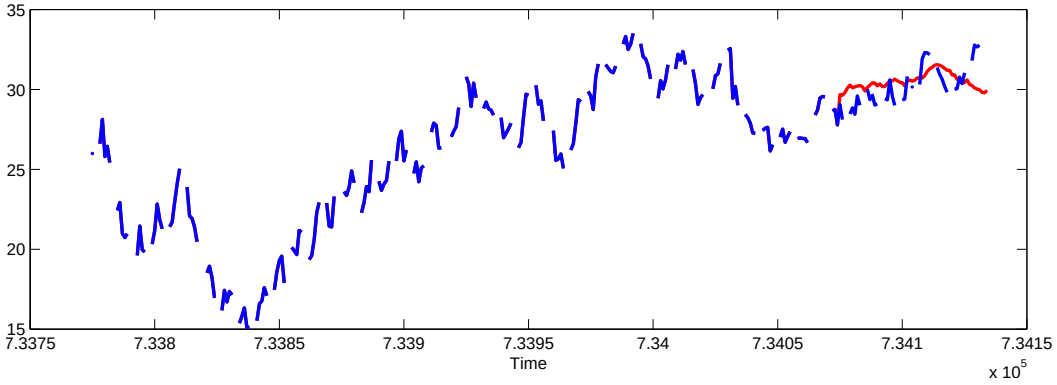


Рис. 1. Одномерный случай

Литература

[1] Магнус Я.Р.,Эконометрика.Начальный курс:Учеб.-6 изд.—М.:Дело,2004.—576 с. ISBN 5-7749-0055-X

Локальные методы прогнозирования временных рядов

А. И. Корниенко
aleksej.kornienko@phystech.edu
Москва, Вычислительный центр РАН

Исследуется алгоритм прогнозирования временных рядов. Алгоритм основан на локальных методах с использованием различных функционалов качества.

Введение

В работе исследуется метод прогнозирования временных рядов. Все методы построения прогноза временных рядов можно разделить на локальные и глобальные. Глобальные методы используют всю известную историю для построения прогноза. Локальные используют только часть истории.

Мы исследуем алгоритм, основанный на методе k ближайших соседей. Мы решаем задачу настройки параметров алгоритма и параметров метрики.

К параметрам алгоритма относятся:

1. Число ближайших соседей.
2. Метрика.
3. Длина рассматриваемого вектора предыстории.

Рассмотренные методы были ранее предложены в работах [1, 2].

Постановка задачи прогнозирования

Заданы несколько временных рядов, один из которых требуется спрогнозировать на h шагов вперед. Предполагается, что прогнозируемый отрезок лежит сразу же за окончанием известных нам значений рядов, т.е. за концом истории. Будем считать, что в истории нет пропусков. Тогда в качестве объекта будем рассматривать последовательности точек длины l.

Требуется найти k наиболее похожих объектов на окончание истории в смысле выбранной метрики. Прогноз строится как линейная комбинация прогнозов соседей.

Описание алгоритма

Основную сложность составляет определение метрики, в смысле которой определяется близость объектов.

Схема алгоритма:

1. Выделяем объект длины l.
2. Нормализуем выделенный объект. Вычитаем минимум и делим на разницу максимального и минимального значений по всем данным рядам.
3. Далее, находим расстояние между выделенным и эталонным объектами:

$$dist(x, y) = \min_{A,b} \rho(x, Ay + b).$$

В данной работе рассматриваются две метрики:

$$\rho(x, y, \omega) = \sum_{k=1}^{l} \omega^{2k} (x(k) - y(k))^2$$

и

$$\rho(x, y, \omega) = \sum_{k=1}^{l} \omega^{2k} |x(k) - y(k)|.$$

В случае, когда ряд — многомерный, вычисляется метрика для каждого ряда со своим параметром ω, и затем значения складываются. Параметр ω настраивается в ходе обучения модели.

4. Если объект оказался среди k ближайших на данный момент, запоминаем полученные параметры и расстояние. Выбрасываем из списка самый далекий от эталонного объект.
5. По результатам работы выделяем k лучших объектов и выделяем за ними ряд длиной достаточной, чтобы покрыть историю.
6. Из полученных рядов строим выпуклую линейную комбинацию с равными весами.
7. Выбираем те элементы спрогнозированного ряда, по которым необходим прогноз.

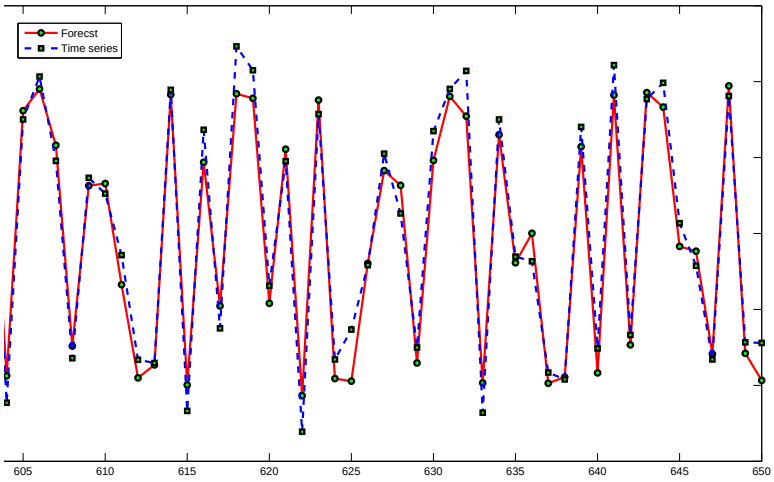


Рис. 1. Результаты прогнозирования

Вычислительный эксперимент

Для примера использовался модельный ряд

$$\boldsymbol{x} = \sin(8\sin(\boldsymbol{a})) + \xi,$$

где $\boldsymbol{a}$ — вектор арифметической прогрессии из 1000 точек, у которой $a_1 = 1$, a_{1000}=500; $\xi \in \mathcal{N}(0, 0.1)$ — нормально распределенная случайная величина.

Обучение — первые 600 элементов ряда. Прогноз строится для 50 следующих временных отсчетов, причем для всех точек сразу.

На рисунке: синий (Time series) — исходный ряд, красный (Forecast) — спрогнозированные значения.

Заключение

В данной работе исследован алгоритм прогнозирования временного ряда, основанный на локальных методах. Проведен вычислительный эксперимент.

Литература

[1] Федорова В. П. Локальные методы прогнозирования временных рядов. Master's thesis, МГУ им. М.В.Ломоносова, 2009.

[2] J. McNames. *Innovations in local modeling for time series prediction.* PhD thesis, Stanford University, 1999.

Ядерное сглаживание

М. П. Кузнецов
mikhail.kuznecov@phystech.edu
Москва, Вычислительный Центр РАН

Целью проекта является прогноз временного ряда на несколько отсчетов методом ядерного сглаживания. Для достижения наилучшего качества прогноза используется выбор параметров модели.

Постановка задачи

Решается задача восстановления регрессии. Задано пространство объектов X и множество возможных ответов $Y = \mathbb{R}$. Существует неизвестная целевая зависимость $y^* : X \to Y$, значения которой известны только на объектах обучающей выборки $X^l = (x_i, y_i)_{i=1}^l$. Требуется построить алгоритм $a : X \to Y$, аппроксимирующий целевую зависимость y^*.

Пути решения задачи

Определение 1. *Гладкая, невозрастающая, ограниченная функция $K : [0, +\infty) \to [0, +\infty)$ называется ядром.*

С помощью функции ядра зададим веса объектов:

$$w_i(x) = K\left(\frac{\rho(x, x_i)}{h}\right). \tag{1}$$

Определение 2. *Параметр h в формуле называется шириной окна сглаживания.*

Определим функционал качества:

$$Q(\alpha, X^l) = \sum_{i=1}^{l} w_i(x)(\alpha - y_i)^2 \to \min_{\alpha \in \mathbb{R}},$$

где w_i — веса обучающих объектов.

Приравняв к нулю $\frac{\partial Q}{\partial \alpha} = 0$, получим формулу Надарая-Ватсона:

$$a_h(x, X^l) = \frac{\sum\limits_{i=1}^{l} y_i K\left(\frac{\rho(x,x_i)}{h}\right)}{K\left(\frac{\rho(x,x_i)}{h}\right)}.$$

Существует теорема, которая утверждает, что для широкого класса ядер оценка Надарая-Ватсона сходится к ожидаемому значению восстанавливаемой зависимости при неограниченном увеличении длины выборки l и одновременном уменьшении ширины окна h.

Выбор ширины окна

Для выбора ширины окна производится минимизация функционала Leave-One-Out:

$$LOO(h, X^l) = \sum_{i=1}^{l} (a_h(x_i, X^l \backslash \{x_i\} - y_i)^2 \to \min_h.$$

Выбор ядра

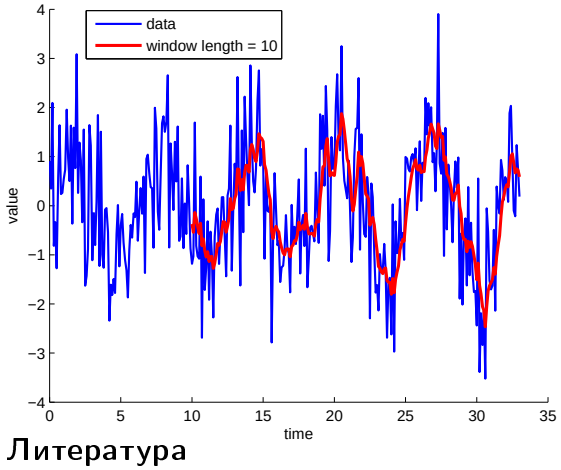
На практике используется несколько видов ядер:

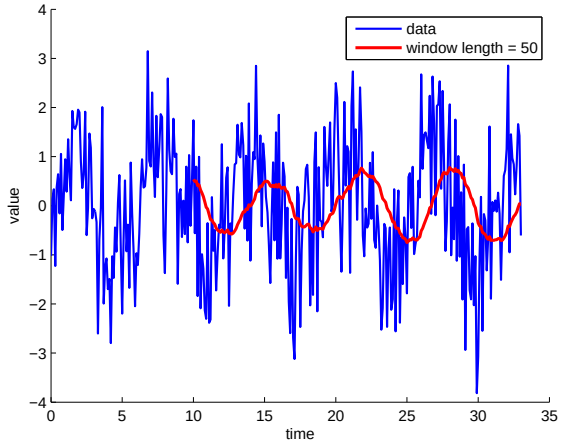
Квартическое	$K(u) = \frac{15}{16}(1 - u^2)^2 I(\lvert u\rvert \leqslant 1)$
Епанечникова	$K(u) = 0.75(1 - u^2) I(\lvert u\rvert \leqslant 1)$
Гаусса	$K(u) = (2\pi)^{-1/2} \exp(-u^2/2)$
Треугольное	$K(u) = (1 - \lvert u\rvert) I(\lvert u\rvert \leqslant 1)$
Прямоугольное	$K(u) = 1/2 I(\lvert u\rvert \leqslant 1)$

Из 5 доступных ядер выбирается то, которое дает минимум SSE при оптимальной ширине окна.

Результат работы алгоритма (одномерный случай)

Исходная зависимость — синус с гауссовским шумом. На левой картинке приведен результат для ширины окна, равной 10, на правой — для ширины окна, равной 50. Используемое ядро — квартическое.





Литература

[1] В. Хардле. *Прикладная непараметрическая регрессия.* 1989.

[2] К. В. Воронцов. *Лекции по алгоритмам восстановления регрессии.*

Метод гибких наименьших квадратов

Н. А. Савинов
nikolay.savinov@phystech.edu
Москва, Вычислительный Центр РАН

Общая постановка задачи и алгоритм решения

Решается задача восстановления нестационарной регрессии. Пусть задана линейная нормальная модель сигнала со скрытой компонентой:

$$\mathbf{x}_t = \mathbf{V}_t\mathbf{x}_{t-1} + \xi_t, \qquad t = 2\dots T, \tag{1}$$

$$y_t = \mathbf{c}_t^T\mathbf{x}_t + \eta_t, \qquad t = 1\dots T, \tag{2}$$

где $\mathbf{x}_t \in \mathbb{R}^n$ — скрытая компонента (вектор коэффициентов регрессии), $y_t \in \mathbb{R}$ — наблюдаемая компонента (прогнозируемая переменная); $\xi_t \in \mathbb{R}^n$ — внутренний шум, $\eta_t \in \mathbb{R}$ — внешний шум, $cov\,(\xi_t) = \frac{1}{\lambda}\mathbf{U}_t^{-1}$, λ — отношение интенсивностей внешнего и внутреннего шумов (гиперпараметр модели). Параметры $\mathbf{V}_t \in \mathbb{R}^{n\times n}$, $\mathbf{U}_t \in \mathbb{R}^{n\times n}$ задаются априори, а $\mathbf{c}_t \in \mathbb{R}^n$ — значение многомерного временного ряда, по которому выполняется прогнозирование.

Алгоритм получает на вход выборку $\{y_t, \mathbf{c}_t\}$, $t = 1\dots T$, на выходе выдает набор значений коэффициентов регрессии $\{\mathbf{x}_t\}$, $t = 1\dots T$. Для оценки коэффициентов используется квадратичная функция потерь:

$$J(\mathbf{x}_1,\dots,\mathbf{x}_T) = \sum_{t=1}^{n}\psi_t(\mathbf{x}_t|y_t) + \lambda\sum_{t=2}^{n}\gamma_t(\mathbf{x}_{t-1},\mathbf{x}_t), \text{ где} \tag{3}$$

$$\psi_t(\mathbf{x}_t|y_t) = (y_t - \mathbf{c}_t{}^T\mathbf{x}_t)^2 = (\mathbf{x}_t - \mathbf{x}_t^0)^T\mathbf{Q}_t^0(\mathbf{x}_t - \mathbf{x}_t^0) \text{ — отклонение от модели сигнала } y_t, \tag{4}$$

$$\mathbf{x}_t^0 = \frac{y_t}{\mathbf{c}_t^T\mathbf{c}_t}\mathbf{c}_t,\ \mathbf{Q}_t^0 = \mathbf{c}_t\mathbf{c}_t^T \text{ — точка минимума и матрица квадратичной формы } \psi_t(\mathbf{x}_t|y_t), \tag{5}$$

$$\gamma_t(\mathbf{x}_{t-1},\mathbf{x}_t) = (\mathbf{x}_t - \mathbf{V}_t\mathbf{x}_{t-1})^T\mathbf{U}_t(\mathbf{x}_t - \mathbf{V}_t\mathbf{x}_{t-1}) \text{ — отклонение от модели скрытых переменных.} \tag{6}$$

Данная функция потерь объединяет 2 противоречивых требования к коэффициентам регрессии $\mathbf{x}_t$: точность восстановления сигнала y_t (первое слагаемое) и гладкость изменения самих коэффициентов регрессии (второе слагаемое). Компромисс между этими требованиями достигается выбором гиперпараметра λ.

Процедура оценивания сводится к применению фильтра-интерполятора Калмана-Бьюси и выполняется в 2 этапа.

1. Фильтрация:
   (а) $\mathbf{x}_{1|1} = \mathbf{x}_1^0,\ \mathbf{Q}_{1|1} = \mathbf{Q}_1^0, \quad t = 1,$
   (б) $\mathbf{Q}_{t|t} = \mathbf{Q}_t^0 + (\mathbf{V}_t\mathbf{Q}_{t-1|t-1}\mathbf{V}_t^{-1} + \frac{1}{\lambda}\mathbf{U}_t^{-1}),\ t = 2\dots T,$
   (в) $\mathbf{k}_t = \mathbf{Q}_{t|t}^{-1}\mathbf{c}_t,\ t = 2\dots T,$
   (г) $\mathbf{x}_{t|t} = V_t\mathbf{x}_{t-1|t-1} + \mathbf{k}_t(y_t - \mathbf{c}_t^T\mathbf{V}_t\mathbf{x}_{t-1|t-1}),\ t = 2\dots T.$
2. Интерполяция:
   (а) $\mathbf{H}_t = (\mathbf{I} + \frac{1}{\lambda}\mathbf{U}_{t+1}^{-1}(\mathbf{V}_{t+1}^{-1})^T\mathbf{Q}_{t|t})^{-1},\ t = T-1\dots 1,$
   (б) $\hat{\mathbf{x}}_t = \mathbf{x}_{t|t} + \mathbf{H}_t(\hat{\mathbf{x}}_{t+1} - \mathbf{x}_{t|t}),\ t = T-1\dots 1.$

Для оценки гиперпараметра λ используется метод скользящего контроля LOO. В статье [1] получены следующие соотношения для значений параметров $\hat{\mathbf{x}}_t$ на основании данных с пропущенным значением y_t:

$$\hat{\mathbf{x}}_t(\mathbf{Y}^{(t)},\lambda) = (\mathbf{Q}_{t|T}^{(t)})^{-1}(\mathbf{Q}_{t|T}\mathbf{x}_{t|T} - \mathbf{Q}_t^0\mathbf{x}_t^0), \tag{7}$$

$$\mathbf{Q}_{t|T}^{(t)} = \mathbf{Q}_{t|T} - \mathbf{Q}_t^0, \text{где } (t) \text{ означает выброшенное измерение } y_t; \tag{8}$$

$$\mathbf{Q}_{t|T} = (\mathbf{H}_t\mathbf{V}_{t+1}^{-1}\mathbf{Q}_{t+1|T}^{-1}(\mathbf{V}_{t+1}^{-1})^T\mathbf{H}_t^T + (\mathbf{Q}_{t|t} + \lambda\mathbf{V}_{t+1}^T\mathbf{U}_{t+1}\mathbf{V}_{t+1})^{-1})^{-1}. \tag{9}$$

Тогда внешний критерий LOO вычисляется по определению:

$$LOO(\lambda) = J(\hat{\mathbf{x}}_1(\mathbf{Y}^{(1)},\lambda),\dots,\hat{\mathbf{x}}_T(\mathbf{Y}^{(T)},\lambda)).$$

При решении задачи значение λ выбирается из условия минимизации внешнего критерия:

$$\hat{\lambda} = \arg\min_{\lambda} LOO(\lambda).$$

Процедура обладает линейной сложностью по длине T временного ряда.

Для применения алгоритма следует задать $\mathbf{U}_t$, $t = 1 \dots T$ (на практике можно использовать диагональные матрицы), а также $\mathbf{V}_t$, $t = 1 \dots T$, которые определяют модель изменения скрытой компоненты. В исходной работе [1] используются единичные матрицы.

Применение для прогнозирования

Сначала производится оценка λ по известной истории временных рядов *hist*. Далее выполняется прогнозирование в заданном временном интервале *frc*.

1. Прогнозируется $\mathbf{c}_t$, $t \in frc$ (с помощью экспоненциального сглаживания).
2. Прогнозируется предварительное значение $y_{t|t-1}$ (с помощью экспоненциального сглаживания).
3. FLS оценивает сглаженный коэффициент регрессии $\mathbf{x}_{t|t}$.
4. Вычисляется сглаженное значение $y_{t|t} = \mathbf{c}_t^T \mathbf{x}_{t|t}$.
5. $t := t + 1$.

См. исходный код: [3].

Тестовый прогноз

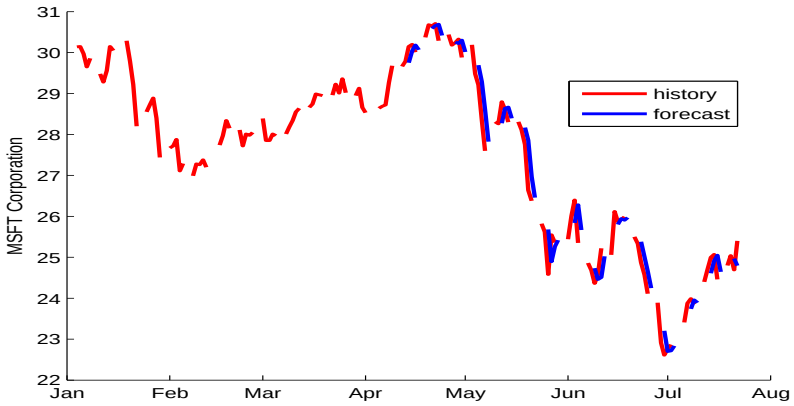


Рис. 1. Цена акций компании Microsoft

Литература

[1] Vadim Mottl, Olga Krasotkina, Michael Markov, and Ilya Muchnik. Time-varying regression model with unknown time-volatility for nonstationary signal analysis. 2006.

[2] R. Kalaba and L. Tesfatsion. Time-varying linear regression via flexible least squares. 1989.

[3] Nikolay Savinov. Flexible least squares. `https://mlalgorithms.svn.sourceforge.net/svnroot/mlalgorithms/TSForecasting/FlexibleLeastSquares/`, 2011.

[4] Vadim Mottl, Olga Krasotkina, Michael Markov, and Ilya Muchnik. Dynamic analysis of hedge funds. 2006.

[5] Vadim Mottl. Lectures on statistical methods of signal and data array analysis.

Прогнозирование временного ряда с помощью приближения производными рядами

А. А. Мафусалов
mafusalov@gmail.com
Москва, Вычислительный Центр РАН

Целью проекта является прогнозирование временного ряда на основании прогнозов для производных от него рядов.

Постановка задачи

Пусть дан временной ряд $S = (s_1, \dots, s_T)$, при значениях времени $T = (t_1, \dots, t_T)$. Пусть также даны производные от него ряды

$$\{S_i = (s_1^i, \dots, s_T^i)\}_{i=1}^n,$$

при тех же значениях времени, для которых существует прогноз

$$\{\bar{S}_i = (s_{T+1}^i, \dots, s_{T+\tau}^i)\}_{i=1}^n,$$

при значениях времени $(t_{T+1}, \dots, t_{T+\tau})$. Например, производными являются ряды, значения которых в каждой точке равны средним значениям исходного ряда в некоторой временной окрестности соответствующей точки.

Требуется построить прогноз $\{\bar{S} = (s_{T+1}, \dots, s_{T+\tau})\}_{i=1}^n$ на интервале значений времени $(t_{T+1}, \dots, t_{T+\tau})$.

Будем строить прогноз как функцию прогнозов производных рядов:

$$\bar{s}_t = F(s_t^1, \dots, s_t^n), \quad F : \mathbb{R}^n \to \mathbb{R}.$$

Постановка подзадачи

Требуется найти $\{\hat{F}^j(S_1, \dots, S_n)\}_{j=1}^{|P|} = \{F^{k_j}(\mathbf{w}, S_1, \dots, S_n)\big|_{\mathbf{w}=\mathbf{w}_j^*}\}_{k_j \in P}$, где $\hat{F} \in \mathfrak{F}$, $\mathfrak{F}$ — множество n-местных непрерывных функций действительной переменной, $F^k \in \mathfrak{F}_1$, $\mathfrak{F}_1$ — множество функций из $\mathfrak{F}$, зависящих дополнительно от вектора параметров, $k \in \mathfrak{K}$ — множество индексов функций множества $\mathfrak{F}_1$, $\mathbf{w}$ — настраиваемый вектор параметров, $\mathbf{w} \in \mathfrak{W}(k)$, $\mathfrak{W}(k)$ — множество допустимых векторов параметров функции F^k.

Множество $\{F^{k_j}\}_{k_j \in P}$ — парето-оптимальное множество по совокупности критериев качества:

$$P = \{k \in \mathfrak{K} | POF(F^k) = 1\},$$

где $POF(F^k)$ — номер парето-слоя, в котором лежит модель с индексом k и вектором параметров, настроенным по минимизации суммы квадратов регрессионных остатков на обучающей подвыборке:

$$\mathbf{w}^* = \arg \min_{\mathbf{w} \in \mathfrak{W}(k)} R_{J_l}(F^k).$$

Для каждой F^{k_j} вектор параметров находится как

$$\mathbf{w}_j^* = \arg \min_{\mathbf{w} \in \mathfrak{W}(k_j)} R_{J_l}(F^{k_j}).$$

Подготовка к решению задачи

1. Найдём множество производных рядов $\{S_i = (s_1^i, \dots, s_T^i)\}_{i=1}^n$ как значения функций от производного ряда: $S_i = H_i(S), H_i : \mathbb{R}^n \to \mathbb{R}^n$.
2. Продлим каждый из производных рядов каким-либо методом прогнозирования до момента времени $T + \tau$, получив прогнозы: $\{\bar{S}_i = (s_{T+1}^i, \dots, s_{T+\tau}^i)\}_{i=1}^n$.

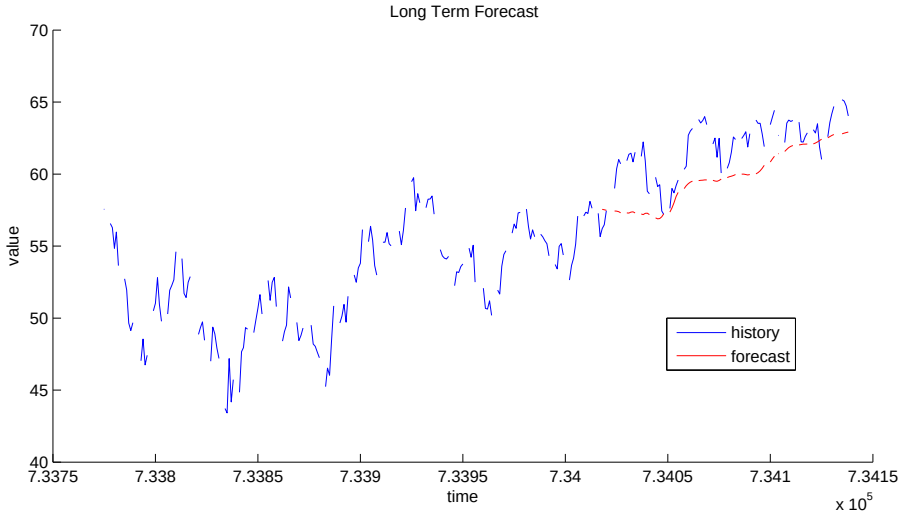
Метод решения задачи

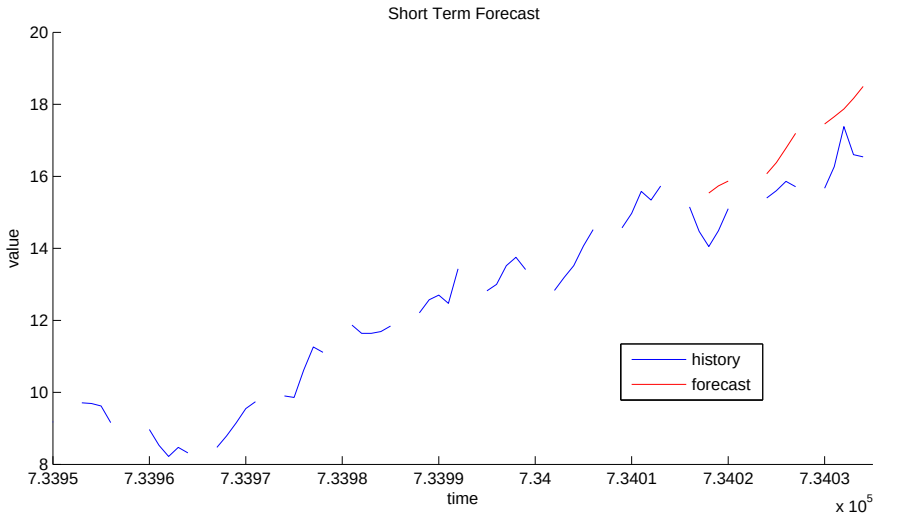
Один из методов решения подзадачи описан здесь [1, 2]. Опишем метод решения задачи, опирающийся на метод, решающий подзадачу.

1. Зададим алфавит функций: $\{f_i(\mathbf{w}, \mathbf{x})\}_{i=1}^{k}$, где $\mathbf{x}$ — вектор-переменная, $\mathbf{w}$ — вектор параметров, — для порождения суперпозиций.
2. Выберем подмножества производных рядов $\{S_{i_1(\alpha)}, \ldots, S_{i_{n(\alpha)}(\alpha)}\}_{\alpha \in A}$ и для каждого $\alpha \in A$ построим суперпозиции $F_1^{\alpha}, \ldots, F_{m(\alpha)}^{\alpha}$, приближающие исходный ряд и оптимальные в смысле постановки подзадачи.
3. Для каждого из производных рядов вычислим количество вхождений в суперпозиции по всему множеству полученных функций.
4. Выберем несколько производных рядов $S_{i_1}, \ldots, S_{i_d}$, которые наибольшее число раз были использованы в оптимальных суперпозициях.
5. Решим подзадачу нахождения суперпозиций, приближающих ряд S производными рядами $S_{i_1}, \ldots, S_{i_d}$, получим функции $F_1, \ldots, F_m$.
6. На основании спрогнозированных значений производных рядов $\{\bar{S}_i = (s_{T+1}^i, \ldots, s_{T+\tau}^i)\}_{i=1}^{n}$ по функциям зависимости $F_1, \ldots, F_m$ получим прогнозы $\{\hat{S}^j = (\hat{s}_{T+1}, \ldots, \hat{s}_{T+\tau})\}_{j=1}^{m}$.

Результат работы алгоритма

Ниже представлены результаты двух прогнозов на разных временных рядах, выполненных алгоритмом.





Литература

[1] Порождение неинейных регрессионных моделей

[2] Поиск нелинейной поверхности Мохоровичича

Выбор моделей в задачах прогнозирования

Д. С. Сунгуров
e1ekt@bk.ru
Москва, Вычислительный Центр РАН

Целью проекта является моделирование многомерных временных рядов. Алгоритм основан на расширении пространства признаков с помощью порождающих функций и последующем выборе линейных моделей.

Постановка задачи и алгоритм решения

Решается задача восстановления регрессии. Задана выборка $D = \{(\xi^i, y^i)\}_{i=1}^m$ — множество пар вектора значений n свободных переменных $\xi^i = (\xi_u^i)_{u=1}^U \in \mathbb{R}^U$ и значения одной зависимой переменной $y^i \in \mathbb{R}^1$. Задано множество порождающих функций $G = \{g_v\}_{v=1}^V$.

Обозначим $a_i = g_v(\xi_u)$, где $i = (v-1)U + u$. Рассмотрим декартово произведение $\Xi \times G$ (Ξ — множество свободных переменных). Элементу (g_v, ξ_u) поставим в соответствие суперпозицию $g_v(\xi_u)$, однозначно определяемую индексами u, v.

В качестве модели, описывающей отношение между зависимой переменной y и свободными переменными a_i, используется полином Колмогорова-Габора:

$$y = \beta_0 + \sum_{l=1}^{UV} \beta_l a_l + \sum_{l=1}^{UV}\sum_{\zeta=1}^{UV} \beta_{l\zeta} a_l a_\zeta + ..., \tag{1}$$

где $\beta = (\beta_0, \beta_l, \beta_{l\zeta}, ...)_{l,\zeta=1,2,...}$ — вектор коэффициентов.

Запишем этот полином в виде линейной комбинации порожденных переменных, где j — номер члена линейной комбинации:

$$y = \sum_{j \in J} \beta_j x_j, \quad |J| = n.$$

Переменные x_i поставлены в однозначное соответствие мономам полинома. В матричной форме соотношение имеет вид:

$$y = \mathbf{X}\beta,$$

где $\mathbf{X}$ — матрица признаков со столбцами $\mathbf{x}_1, \cdots, \mathbf{x}_n$, $\beta = (\beta_1, \cdots, \beta_n)$ — вектор параметров.

Предполагается, что векторы $x_j = (x_j^1, ..., x_j^m)$ линейно независимы. Линейно зависимые вектора исключаются из дальнейшего рассмотрения. Для обнаружения мультиколлинеарности используется метод Белсли [1].

Пусть матрица признаков X имеет размер $m \times n$, центрирована и нормирована. Проводится сингулярное разложение:

$$X = U\Lambda V^T,$$

где U, V — ортогональные матрицы, Λ — диагональная матрица с сингулярными числами на диагонали, упорядоченными по убыванию.

Индекс обусловленности с номером i — отношение максимального сингулярного числа к i-му сингулярному числу:

$$\eta_i = \frac{\lambda_{max}}{\lambda_i}.$$

Ковариационная матрица метода наименьших квадратов может быть записана следующим образом:

$$V(\beta) = \sigma^2 (X^T X)^{-1} = \sigma^2 V \Lambda^{-2} V^T$$

Обозначим $V = (v_{ij})$ и перепишем предыдущее выражение:

$$\sigma^{-2} V(\beta_i) = \frac{v_{i1}^2}{\lambda_{i1}^2} + \frac{v_{i2}^2}{\lambda_{i2}^2} + \cdots + \frac{v_{in}^2}{\lambda_{in}^2} = (q_{i1} + q_{i2} + \cdots + q_{in}) \sum_{j=1}^{n} \frac{v_{ij}^2}{\lambda_{ij}^2},$$

где q_{ij} — дисперсионные доли, т.е. отношения соответствующего слагаемого в разложении $\sigma^{-2} V(\beta_i)$ ко всей сумме.

Большие значения индексов обусловленности означает наличие мультиколлинеарности. Большие значения q_{ij}, соответствующие большим индексам обусловленности относятся к признакам, между которыми эта зависимость существует. Алгоритм отбрасывает признаки, находящиеся в зависимости с индексами обусловленности превышающими заданное значение.

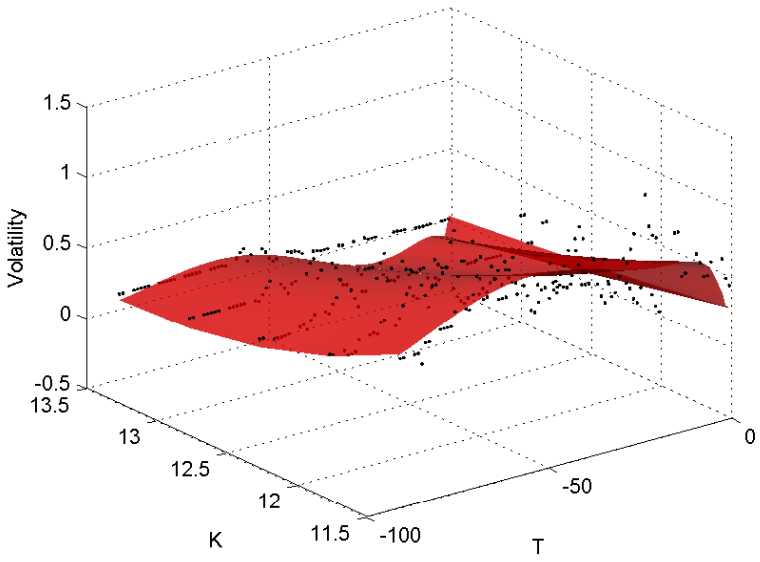
Отбор признаков осуществляется одним из следующих методов(см. [2]).

1. Полный перебор моделей.
2. Генетический алгоритм;
3. Метод группового учета аргументов;
4. Алгоритм Лассо;
5. LARS;
6. Последовательное добавление признаков с ортогонализацией;
7. Случайный поиск с адаптацией;
8. Гребневая регрессия;
9. Поиск самой длинной проекции вектора ответов на биссекторы векторов признаков.

По отобранным признакам выполняется восстановление линейной регрессии.

Пример работы алгоритма

На графике показано работа алгоритма на данных биржевых опционов.



Литература

[1] David A. Belsley. A Guide to using the collinearity diagnostics. *Computational Economics*, 4(1):33–50, February 1991.

[2] Стрижов В.В. Методы выбора регрессионных моделей, ВЦ РАН 2010.

Метод SSA для прогнозирования временных рядов*

И. В. Фадеев
Московский физико-технический институт, ФУПМ, каф. «Интеллектуальные системы»

Описание алгоритма

Постановка задачи

Наблюдается система функций дискретного аргумента $(f_i^{(k)})_{i=1}^N$, где k = 1, ..., s. s — число временных рядов, k — номер ряда, N — длина временного ряда, i — номер отсчета. Требуется разложить ряд в сумму компонент (используя метод главных компонент, см. описание алгоритма), интерпретировать каждую компоненту, и построить продолжение ряда $(f_i^{(k)})_{i=1}^{N+M}$ по выбранным компонентам.

Построение матрицы наблюдений

Рассмотрим сначала одномерный временной ряд $(f_i)_{i=1}^N$. Выберем n такое, что $0 < n \leqslant N-1$ — время жизни многомерной гусеницы. Пусть $\sigma = N - n + 1$ — длина гусеницы. Построим последовательность из n векторов в R^σ следующего вида:

$$Y^{(l)} \in R^\sigma,$$
$$Y^{(l)} = (f_{i+l-1})_{i=1}^\sigma$$

Обозначим

$$Z = (Y^{(1)}, \ldots, Y^{(n)}) :$$

Будем называть Z нецентрированной матрицей наблюдений, порождённой гусеницей со временем жизни n.

В случае многомерного временного ряда матрицей наблюдения называется столбец из матриц наблюдений, соответствующих каждой из компонент.

Проводимый в дальнейшем анализ главных компонент может проводиться как по центрированной, так и по нецентрированной выборкам. Для упрощения выкладок рассмотрим простейший нецентрированный вариант.

Анализ главных компонент

Рассмотрим ковариационную матрицу полученной выборки:

$$C = \frac{1}{n} Z Z^T.$$

Выполним её svd-разложение:

$$C = V \Lambda V^T,$$

где $\Lambda = diag(\lambda_1, \ldots, \lambda_\tau)$ — диагональная матрица собственных чисел, $V = (v^{(1)}, \ldots, v^{(\tau)})$, $(v^{(i)})^T v^{(j)} = \delta_{ij}$ — ортогональная матрица собственных векторов.

Далее рассмотрим систему главных компонент:

$$U = V^T Z, U = (U^{(1)}, \ldots, U^{(\tau)})^T.$$

После проведения анализа главных компонент обычно предполагается проведение операции восстановления исходной матрицы наблюдений по некоторому поднабору главных компонент, т. е. для $V' = (v^{(i_1)}, \ldots, v^{(i_r)})$ и $U' = V'^T Z$ вычисляется матрица $Z' = V' U'$.

Далее восстанавливаются исходные последовательности. В одномерном случае i-ая компонента восстановленного ряда есть среднее значение по i-ой диагонали восстановленной матрицы наблюдений Z'.

В многомерном случае усреднение проводится с учётом того, что матрица наблюдений состоит из подматриц, соответствующих каждой компоненте ряда:

Научный руководитель В. В. Стрижов

$$f'^{(k)}_m = \left\{ \begin{array}{ll} \frac{1}{m}\sum_{i=1}^{m} x_i^{(m-i+1,k)} & 1 \leqslant m \leqslant \sigma, \\ \frac{1}{\sigma}\sum_{i=1}^{\sigma} x_i^{(m-i+1,k)} & \sigma \leqslant m \leqslant n, \\ \frac{1}{N-m+1}\sum_{i=1}^{N-m+1} x_{i+m-n}^{(n-i+1,k)} & n \leqslant m \leqslant N. \end{array} \right\}$$

Прогноз

Числовой ряд $(f_i)_{i=1}^{N+1}$ называется продолжением ряда $(f_i)_{i=1}^{N}$, если порождаемая им при гусеничной обработке выборка лежит в той же гиперплоскости, что и у исходного ряда. Пусть у нас есть некоторый набор выбранных главных компонент $i_1, i_2, \ldots, i_r$. Определим

$$w = \begin{pmatrix} v_\sigma^{(i_1)} & v_\sigma^{(i_2)} & \ldots & v_\sigma^{(i_r)} \\ v_{2\sigma}^{(i_1)} & v_{2\sigma}^{(i_2)} & \ldots & v_{2\sigma}^{(i_r)} \\ \vdots & \vdots & \ddots & \vdots \\ v_\tau^{(i_1)} & v_\tau^{(i_2)} & \ldots & v_\tau^{(i_r)} \end{pmatrix}$$

и

$$V_* = \begin{pmatrix} v_1^{(i_1)} & v_1^{(i_2)} & \ldots & v_1^{(i_r)} \\ \vdots & \vdots & \ddots & \vdots \\ v_{\sigma-1}^{(i_1)} & v_{\sigma-1}^{(i_2)} & \ldots & v_{\sigma-1}^{(i_r)} \\ v_{\sigma+1}^{(i_1)} & v_{\sigma+1}^{(i_2)} & \ldots & v_{\sigma+1}^{(i_r)} \\ \vdots & \vdots & \ddots & \vdots \\ v_{2\sigma-1}^{(i_1)} & v_{2\sigma-1}^{(i_2)} & \ldots & v_{2\sigma-1}^{(i_r)} \\ \vdots & \vdots & \ddots & \vdots \\ v_{\tau-1}^{(i_1)} & v_{\tau-1}^{(i_2)} & \ldots & v_{\tau-1}^{(i_r)} \end{pmatrix}$$

Также положим

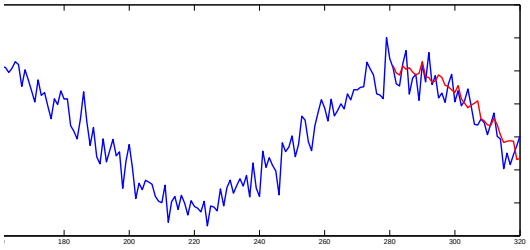
$$Q = \left(f_{N-\sigma+2}^{(1)}, \ldots, f_N^{(1)}, f_{N-\sigma+2}^{(2)}, \ldots, f_N^{(2)}, \ldots, f_{N-\sigma+2}^{(s)}, \ldots, f_N^{(s)}\right)^T$$

Тогда прогнозируемые значения системы в точке $N+1$ вычисляются по формуле:

$$f_{N+1} = w(V_*^T V_*)^{-1} V_*^T Q.$$

Тестовый прогноз

Протестируем алгоритм на тестовом временном ряде, синтезируемом как сумма линейной компоненты (тренда), двух сезонных компонент (синусоид разной частоты) и нормального шума. На графике представлен исходный временной ряд (синий) и прогнозируемый с помощью алгоритма SSA (красный):



Метаописание временных рядов

А. Н. Фирстенко
alexnickfirst@gmail.com,
Москва, Вычислительный Центр РАН

Исследуется возможность метаописания временных рядов с целью последующей их классификации. Под метаописанием ряда понимается некоторый набор признаков, характеризующих временной ряд. В качестве метки класса для некоторого временного ряда выступает название того алгоритма из наперед заданного множества алгоритмов прогнозирования, который прогнозирует этот ряд наилучшим образом.

Введение

Для прогнозирования временных рядов существует большое количество алгоритмов [1, 2]. Результат работы каждого алгоритма зависит от свойств прогнозируемого ряда, поэтому возникает задача автоматического выбора наилучшего алгоритма из некоторого заданного семейства.

Данную задачу можно рассматривать как задачу классификации. Объектами классификации являются временные ряды. Для метаописания временного ряда создается набор признаков. В качестве признаков были использованы длина ряда, число вспомогательных рядов, максимальное и минимальное значения ряда, число пропущенных данных, среднее значение временного ряда. Метками классов являются названия алгоритмов прогнозирования. Временной ряд относится к некоторому классу, если соответствующий этому классу алгоритм работает на временном ряде наилучшим образом по заданному функционалу качества.

Для классификации был использован алгоритм k взвешенных ближайших соседей [1].

Постановка задачи метоописания временных рядов

Задано множество многомерных временных рядов $S = \{s_i\}_{i=1}^{n}$. Задано множество алгоритмов прогнозирования $A = \{a_k\}_{k=1}^{l}$. Задан функционал качества работы алгоритма $L(s, a)$. Требуется построить алгоритм классификации, который выбирает для нового временного ряда s алгоритм a_{opt}, такой что:

$$a_{\text{opt}} = \arg\min_{a \in A} L(s, a).$$

Описание алгоритма

Алгоритм решения поставленной задачи.

1. Для каждого ряда $s_i \in S$ составить его признаковое описание $\mathbf{x}_i = (x_{i1}, ..., x_{im})$.
2. Для каждого ряда $s_i \in S$ определить метку $y_i = k$, где $a_k = \arg\min_{a \in A} L(s_i, a)$.
3. Настроить классификатор по X, $\mathbf{y}$.
4. Классифицировать s по его признаковому описанию.

Вычислительный эксперимент

Был проведен вычислительный эксперимент на выборке из 120 рядов, по результатам прогнозирования их тремя алгоритмами — SSA, ARIMA, локальными методами прогнозирования. Исходные данные доступны здесь[4]. По результатам прогнозирования каждый ряд был отнесен к классу, соответствующему алгоритму, который на данном ряде работает наилучшим образом. Было проведена 1000 экспериментов, в ходе каждого из которых выборка случайным образом делилась на тестовую и обучающую с сохранением пропорций по классам. Результаты эксперимента приведены в таблице.

	SSA	ARIMA	LOCAL
Ряды в исходной выборке, шт.	14	49	57
Ряды в обучающей выборке, шт.	4	16	19
Ряды в тестовой выборке, шт.	10	33	38
В среднем неправильно классифицировано, шт.	9.7	20.7	24.1

Выводы

Исходя из результатов вычислительного эксперимента, можно утверждать, что для более успешного автоматического выбора алгоритмов прогнозирования требуется более тщательный выбор признаков для метаописания рядов, а также, возможно, следует использовать другие алгоритмы классификации.

Литература

[1] *Kalaba R., Tesfatsion L.* Time-varying linear regression via flexible least squares. — 1989.

[2] *McNames J.* Innovations in local modeling for time series prediction: Ph.D. thesis / Stanford University. — 1999.

[3] *Воронцов.* Машинное обучение (курс лекций).

[4] https://mlalgorithms.svn.sourceforge.net/svnroot/mlalgorithms/tsforecasting/tsmetadescription/data/.

Прогнозирование событий

Д. С. Кононенко
daniilkononenko@gmail.com
Москва, кафедра «Интеллектуальные системы», ФУПМ МФТИ

Рассматривается метод прогнозирования событий на основании информации о наличии событий во временных рядах, данной экспертом. Предлагается способ порождения и отбора признаков, описывающих временной ряд. Алгоритм основан на разметке интервалов роста и падения временного ряда и применении логистической регрессии для настройки параметров линейной модели и оценки ее качества.

Постановка задачи

Дана выборка временных рядов, размеченных экспертом:

$$\{\mathbf{x}^i, y^i\}, \mathbf{x}^i = (x_1^i, \ldots, x_T^i) \in R^T, i = 1 \ldots l.$$

Здесь целевая переменная:

$$y^i = \begin{cases} 1, & \text{в данном ряде есть событие;} \\ 0, & \text{в данном ряде нет события.} \end{cases}$$

Необходимо предложить признаковое описание временного ряда:

$$g(x^i) = (g_1(x^i), \ldots, g_n(x^i)).$$

На основании этого описания требуется решить задачу классификации — построить модель $f : R^w \times R^T \to \{0, 1\}$, где R^w — пространство параметров модели.

Задача разбивается на три этапа.

1. Порождение множества числовых признаков $(g_1(x), \ldots, g_N(x))$, описывающих временной ряд.
2. Предложение критерия качества модели.
3. Выбор наилучшей модели.

Алгоритм решения

Порождение множества признаков.

Зафиксируем множество меток $\mathcal{M} = \{'U', 'D'\}$: '$U$' — повышение значения ряда, 'D' — понижение (см. (1)). Размечаем временной ряд $(m_1, \ldots, m_T)$:

$$m_k = \begin{cases} 'U', & \text{если } x_k - x_{k-1} > 0, \\ 'D', & \text{иначе.} \end{cases} \tag{1}$$

Таким образом, для каждого временного ряда $\mathbf{x}^i$ получаем строчку из $(T-1)$ символа $'U'$ или $'D'$: $\mathbf{m}^i = m_1^i \ldots m_{T-1}^i$ — слово длины $(T-1)$ в алфавите $\mathcal{M}$.

Рассмотрим теперь все слова длины не более w в алфавите $\mathcal{M}$. Таких слов $N = (2^w - 1)$ (исключаем пустое слово). Пронумеруем их: $R = (r_1, \ldots, r_N)$. Слову $r_k = r_k^1 \ldots r_k^{n_k}$ поставим в соответствие 2 числовых признака.

1. Бинарный признак
$$g_{2k-1}(x_i) = \begin{cases} 1, & \text{если в строке } \mathbf{m}^i \text{ есть подстрока } r_k; \\ 0, & \text{иначе.} \end{cases}$$
2. Действительный признак
$$g_{2k}(x_i) = \begin{cases} x_{j+n^k}^i - x_j^i, & \text{если в строке } \mathbf{m}^i \text{ есть подстрока } r_k : (m_{j+1}^i, \ldots, m_{j+n_k}^i) = (r_k^1, \ldots, r_k^{n_k}); \\ 0, & \text{если в строке } \mathbf{m}^i \text{ нет подстроки } r_k. \end{cases}$$

Если комбинация встречается в ряде несколько раз, то мы рассматриваем только ее последнее по времени вхождение (и записываем в действительный признак суммарное изменение цены на нем). Таким образом,

мы получаем $2N = 2(2^w - 1)$ числовых признаков и матрицу объект-признак размера $l \times 2N$:

$$G = \begin{pmatrix} g_1(x^1) & \dots & g_{2N}(x^1) \\ \vdots & \ddots & \vdots \\ g_1(x^l) & \dots & g_{2N}(x^l) \end{pmatrix}.$$

Отбор наилучшей модели.

Строить классификатор и оценивать качество модели будем с помощью логистической регрессии [1]. За оценку качества модели примем площадь под ROC-кривой данного классификатора.

Отбор наилучшей модели будем производить с помощью генетического алгоритма [2].

Фиксируем четное число — длину модели n (количество признаков в ней). Ранее мы построили множество слов $R = (r_1, \dots, r_N)$ длины не более w в алфавите $\mathcal{M}$ и каждому слову r_k поставили в соответствие признаки g_{2k-1} и g_{2k}. Активное множество признаков $\mathcal{A}$ — признаки, на основании которых строится текущая модель, $|\mathcal{A}| = n$. Поставим ему в соответствие активное множество индексов $\mathcal{I} = \{i_1, \dots, i_{\frac{n}{2}}\}$, $i_k \in \{1, \dots, N\}$, $|\mathcal{I}| = \frac{n}{2}$, следующим образом:

$$\mathcal{I} = \{i_1, \dots, i_{\frac{n}{2}}\} \Leftrightarrow \mathcal{A} = \{g_{2i_1-1}, g_{2i_1}, \dots, g_{2i_{\frac{n}{2}}-1}, g_{2i_{\frac{n}{2}}}\}.$$

Таким образом, признаки g_{2k-1} и g_{2k} всегда используются или не используются в модели вместе. Модель полностью описывается активным множеством индексов, поэтому далее генетический алгоритм опишем в терминах этих множеств.

Генетический алгоритм содержит следующие параметры для отбора моделей: F — число лучших моделей в популяции, F_1 — число моделей для скрещивания, P_2 — вероятность выбора модели для мутации.

Начальное множество моделей (популяцию) выбираем случайным образом. Делим все множество индексов $\{1, \dots, N\}$ случайным образом на подмножества размера $\frac{n}{2}$ (для простоты считаем, что $2N$ делится на n):

$$\{1, \dots, N\} = \{(k_1, \dots, k_{\frac{n}{2}}), (k_{\frac{n}{2}+1}, \dots k_n), \dots\} = (\mathcal{I}_1, \dots, \mathcal{I}_P),$$

где $P = \frac{2N}{n}$ — размер популяции.

Итеративно выполняются следующие операции.

1. Отбор.
   Модели сортируются по убывания качества (площадь под ROC-кривой), и в следующее поколение попадают F лучших.
2. Случайным образом выбираются F_1 моделей для скрещивания и мутации.
3. Скрещивание.
   Выбранные модели случайным образом делятся на пары. В каждой паре активные множества индексов $\mathcal{I}_p = (k_1^p, \dots, k_{\frac{n}{2}}^p)$ и $\mathcal{I}_q = (k_1^q, \dots, k_{\frac{n}{2}}^q)$ разбиваются точкой кроссинговера m, выбираемой случайно из множества $\{1, \dots \frac{n}{2}\}$, на 2 части. Происходит обмен элементов множеств $\mathcal{I}_p$ и $\mathcal{I}_q$ так, чтобы в новых множествах не было одинаковых индексов:

$$\begin{cases} (k_1^p, \dots, k_m^p, k_{m+1}^p, \dots, k_{\frac{n}{2}}^p) \\ (k_1^q, \dots, k_m^q, k_{m+1}^q, \dots, k_{\frac{n}{2}}^q) \end{cases} \mapsto \begin{cases} (k_1^p, \dots, k_m^p, k_{m+1}^q, \dots, k_{\frac{n}{2}}^q) \\ (k_1^q, \dots, k_m^q, k_{m+1}^p, \dots, k_{\frac{n}{2}}^p) \end{cases}.$$

4. Мутация.
   Каждая модель $\mathcal{I} = (k_1, \dots, k_{\frac{n}{2}})$ из полученного множества с вероятностью P_2 подвергается мутации: случайным образом из множества $\{1, \dots \frac{n}{2}\}$ выбирается число j, и значение k_j меняется на случайно выбранный индекс из множества $\{1, \dots N\}$. При этом контролируется, чтобы в новом множестве индексов не было одинаковых индексов.
5. Оценка полученных моделей.
   Новые модели настраиваются и оцениваются (см. пункт Оценивание качества модели).

После заданного числа шагов генетического алгоритма выбирается лучшая модель, таким образом получаем набор признаков $(g_1, \dots, g_n)$, на основании которых строится итоговый классификатор.

Литература

[1] К. В. Воронцов. *Машинное обучение (курс лекций).*

[2] В. В. Стрижов. *Методы выбора регрессионных моделей.* ВЦ РАН, 2010.

Экспоненциальное сглаживание

Н. К. Животовский
nikita.zhivotovskiy@phystech.edu
Москва, Вычислительный Центр РАН

Целью проекта является прогноз временного ряда на несколько отсчетов методом экспоненциального сглаживания. Для достижения наилучшего качества прогноза используется выбор параметра модели.

Постановка задачи

Решается задача прогнозирования временного ряда. Пусть задано пространство объектов X и задан временной ряд $\{x_t\}_{t=1}^{L}$ $x_i \in X$, причем известными считаются первые N отсчетов. Требуется построить алгоритм $a : \{1, \ldots, N+D\} \to X$, аппроксимирующий зависимость x_i, $i = \{1, \ldots, N+D\}$. Параметр D называется горизонтом прогнозирования.

Для решения этой задачи используется техника экспоненциального сглаживания.

Описание метода

Определение 1. *Рядом экспоненциальных средних, построенным по временному ряду $\{x_t\}_{t=0}^{T}$, называется ряд $\{S_t\}$, построенный индуктивно:*

$$S_t = \alpha x_t + (1-\alpha)S_{t-1}, \quad t > 1.$$

В определении ряда S_0 — выбранное начальное значение. Величина $\alpha \in (0,1)$ называется параметром сглаживания.

Значение S_{t-1} будем использовать как прогноз для x_t. Замкнутая форма для S_t получается преобразованием суммы:

$$S_t = \alpha \sum_{i=0}^{N} (1-\alpha)^i x_{t-i} + (1-\alpha)^N S_0,$$

где N — число членов ряда. Таким образом, S_t — взвешенная сумма членов ряда. Заметим, что чем «старше» член ряда, тем меньше его вес.

В задачах краткосрочного прогнозирования для повышения значимости «свежих» наблюдений параметр сглаживания следует увеличивать. Однако для сглаживания выбросов α нужно уменьшать. Таким образом, задача сводится к поиску компромиссного значения параметра.

Выбор значения S_0

Значение S_0 должно задаваться с учетом априорной информации. Часто используется среднее значение ряда или среднее значение наблюдений в предыстории, если она доступна. Также в качестве S_0 может быть положено значение x_1.

В данной работе в качестве S_0 используется среднее значение наблюдаемого ряда.

Рассчитаем, сколько последних членов ряда вносят существенный вклад в прогноз. Пусть x_T - текущее наблюдение. Назовём возрастом наблюдения x_t относительно наблюдения x_T число $T - t$. В частности, возраст текущего наблюдения равен 0, предыдущего — 1.

Определение 2. *Средним возрастом членов ряда называется взвешенная сумма возрастов с коэффициентами $\{\alpha(1-\alpha)^i\}_{i=0}^{\infty}$.*

Средний возраст сходится при длине ряда, стремящейся к бесконечности, к величине

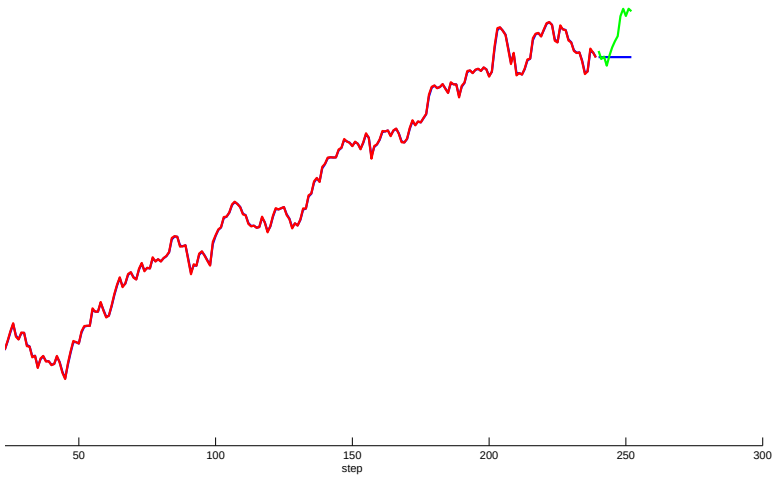
$$0 \cdot \alpha + 1 \cdot \alpha(1-\alpha) + 2 \cdot \alpha(1-\alpha)^2 \ldots = \frac{1-\alpha}{\alpha}.$$

Значит, малые значения α увеличивают средний возраст информации.

Выбор параметра сглаживания

Значение α минимизирует Leave One Out Cross Validation: $\alpha = \arg\min_{\alpha} \sum_{i=0}^{N-1} (S_i - x_{i+1})^2$.

Результат работы алгоритма



Литература

[1] Лукашин Ю. П., Адаптивные методы краткосрочного прогнозирования временных рядов, Финансы и статистика, Москва, 2003.

Визуализация

Д. С. Кононенко
daniilkononenko@gmail.com
Москва, кафедра «Интеллектуальные системы», ФУПМ МФТИ

Целью данной технологической карты является описание работы функции визуализации PlotTS.

Описание возможностей

Реализованы следующие опции:

1. Построение графика истории временного ряда.
2. Построение графика прогноза.
3. Форматирование диапазонов по осям.
4. Вывод легенды.
5. Отображение заданного в параметрах значения функции оценки качества прогноза.
6. Вывод на дополнительном графике вспомогательных временных рядов, использованных при прогнозе.
7. Сохранения графика в файл.

Для детального описания входных и выходных параметров функции и примеров работы см. файл PlotTS.m. Формат входных параметров стандартный, все параметры построения графика передаются в структуре plotOptions. В одной папке с файлом PlotTS.m должны лежать файлы defaultPlotOptions.mat and uniteStructures.m. В файле defaultPlotOptions.mat можно посмотреть пример задания параметров построения графика, также оттуда подгружаются параметры, не указанные пользователем. Пример выводимого графика — рис. (1).

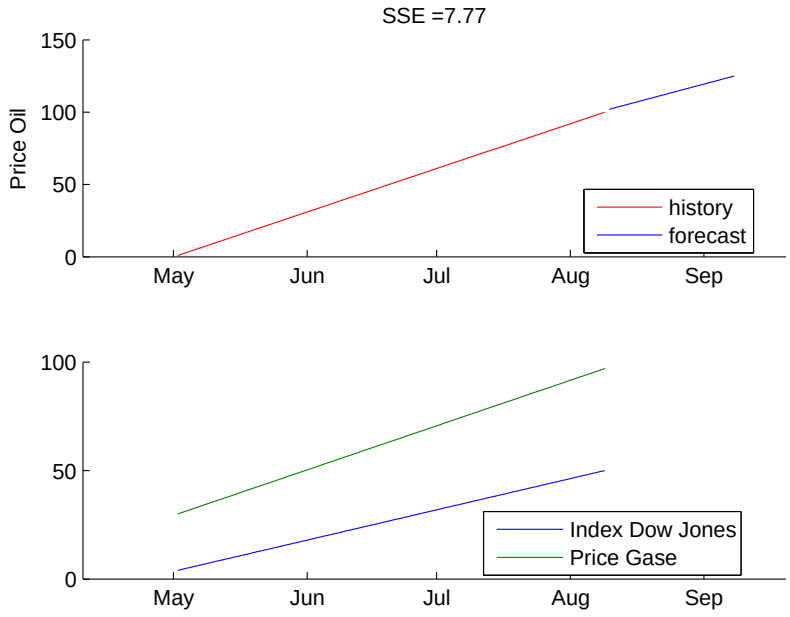


Рис. 1. Пример работы PlotTS

Стилевая правка

М. П. Кузнецов, А. А. Морозов, Д. С. Сунгуров
mikhail.kuznecov@phystech.edu, morozovmipt@gmail.com, e1ekt@bk.ru
Москва, Вычислительный Центр РАН

Целью данной технологической карты является указание последовательности действий при проверке кода корректором и основные ошибки руководителей проектов.

Порядок работы

1. Открыть код, посмотреть основную функцию программы. В ней должно быть немного строк, вызовы некоторых процедур, связанных с подбором параметров. После просмотра функции должно стать понятно понятно, как работает алгоритм, и на какие блоки разбит. Если это непонятно, то, скорее всего, программа спроектирована криво.

2. Смотреть каждую функцию, следуя логике работы программы.

3. Посмотреть на входные/выходные параметры. Если выходных параметров много, это подозрение на то, что функция, скорее всего, должна быть разбита на несколько маленьких. Каждая из подфункций должна выполнять свою работу.

4. Прочитать документацию и example. Из них, опять же, должно быть понятно, как будет работать функция. Если пункты 1-4 выполнены добросовестно, то это сильно облегчит дальнейшую проверку кода.

5. Приступить к проверке конкретной функции. Основная задача корректора — сделать так, чтобы код стал понятным. Соответственно, если код понятен корректору сразу, скорее всего, грубых замечаний в нем не будет, и наоборот. Ниже приведены стандартные замечания, на которые следует обращать внимание при написании и проверке кода.

Стандартные замечания.

1. Названия переменных: надо придумывать переменной название, содержащее в себе ее смысл, не боясь перегромоздить код словами. Даже если эта переменная — просто индекс небольшого цикла.

2. В случае, когда переменная — индекс, следует приписывать ей префикс «i» или «j», например, iFunction. Если переменная обозначает количество (размер) ч.-либо, следует приписывать ей префикс «n». Например,

for iFunction = 1:nFunctions

3. Все операторы типа «=», «==», «<»,... следует выделять пробелами.

4. Не должно быть булевских переменных с названиями типа «flag». Название любого флага должно передавать его смысл. Лучше ставить префикс «is», например, «isFound».

5. В описание всех функций надо вставлять example.

6. Большую функцию всегда лучше разделить на несколько маленьких, каждая из которых выполняет конкретное действие. Если так не делать, то следует вставлять в тело функции комментарии, раскрывающие смысл каждого отдельного куска.

7. Цифр в коде должно быть по минимуму, они должны выноситься в начало функции константами. Константы следует писать с большой буквы, через подчеркивание, например, NUMBER_OF_ELEMENTS.

8. Названия структур должны начинаться с заглавной буквы.

9. При перечислении аргументов функции, следует ставить пробелы после запятых:

result = Function(arg1, arg2, ..., argN);

Обнаруженные ошибки (Сунгуров).

1. Ядерное сглаживание (Михаил Кузнецов). Замечаний нет.

2. Flexible Least Squares (Николай Савинов). Пара ошибок в названиях переменных и констант. Ошибки исправлены.

3. Локальные алгоритмы (Алексей Корниенко). Несколько ошибок в названиях переменных, не выделенные пробелами операторы, отсутствие пробелов при перечислении аргументов функций, отсутствие example в описании функций. Ошибки исправлены.

4. Групповой прогноз (Евгений Зайцев). Множество ошибок в названиях переменных, не выделенные пробелами операторы, отсутствие пробелов при перечислении аргументов функций, отсутствие example в описаниях функций, гигантские функции. Ошибки исправлены.

Обнаруженные ошибки (Морозов Алексей).

1. Параллельные вычисления (Роман Быстрый).

Вынести константы в отдельные переменные. Нет примеров в функции, не описаны входные и выходные данные.

2. Метаописание временных рядов (Александр Фирстенко).

В функции tsAlgLabel переменные m, best не используются, удалить или заменить на ~. Переменная n (nCol), Y (numLabels), i (iCol, index) нужно переименовать на более понятные. В функции tsClassify на выходе присутствует переменная predictions, а она никак не описана. Предлагается переименовать переменные X (featureDescription) и Y (Weight). В функции tsClassLearn нужно переименовать названия структуры PP (paramOfMethod) и переменных XL, YL. В функции tsFeatureDescription нужно переименовать переменную X (featureDescription), i (index). В функции TSFeatureGeneration изменить название функции tsFeatureGeneration.

3. Приближения производными рядами (Александр Мафусалов).

Было предложено заменить русскоязычные комментарии на англоязычные. Нужно откомментировать функции — входные, выходные данные.

4. Многомерная авторегрессия (Раиса Джамтырова).

Переменные должны быть написаны с маленькой буквы, например numberOfPoints. Нужно откомментировать функции — входные данные, выходные, что делают, примеры: getTreeDimentionalMatrix, CatNewMatrix, ForecastOnePeriod, GetLinearCoefficients. В функции ForecastOnePeriod заменить переменные, которые не используются Переменная l в 3й строчке не несет смысл. Замени на . В функции ManyDimentionalAR на выходе функции MatrixOfLinearCoefficients, а не par. В функции modul исправить в первой строчке forecastinf на forecasting. Эту функцию нужно описать. Предлагается все операции с рисованием графиков вынести в отдельную функцию. В функции params нужно откомментировать функции — входные данные, выходные. Заменить переменную params на paramsOfAlg.

Создание базы данных многомерных временных рядов для задач прогнозирования

Н. А. Савинов
savinov.nikolay@phystech.edu
Москва, Вычислительный Центр РАН

Целью данной работы является объяснение принципов поиска временных рядов в Интернете. Приводится также технологический отчет о создании базы данных и средств автоматического скачивания.

Задача поиска временных рядов

Временные ряды бывают 2 основных типов: природные и финансово-экономические. *Требуется найти ряды обоих типов, желательно организовать автоматическое скачивание.*

Наибольший объем информации можно получить от следующих информационно-финансовых систем (они являются главными мировыми поставщиками этих данных):

1. Yahoo finance
2. Google finance
3. Bloomberg
4. Datastream Thomson Reuters
5. World Bank

Начинать поиск таких систем нужно хотя бы с одного крупного источника финансовой информации. Вводим google finance в Wikipedia, получаем ссылки на yahooFinance, Bloomberg, Reuter (в разделе См. Также Wiki-статьи). Это почти полный список всех крупных информационных источников в Интернете.

При этом бесплатные данные предоставляют только 1, 2, 5.

Для скачивания данных в этих системах выбираем индекс/акцию, historical prices, download to spreadsheet. Данные сохраняются в формате .csv.

Можно организовать автоматическом скачивание, задавая соответствующий запрос в адресной строке. Также для автоматического скачивания можно использовать Matlab, в состав которого включен Datafeed Toolbox. В документации разобран пример с подключением к базе Bloomberg (есть возможность работать с бесплатными данными через Yahoo finance).

Имеется аналог UCI для временных рядов: UCR Classification/Clustering Page by Eamonn Keogh (22 ряда). Однако для скачивания оттуда требуется регистрация с указанием личных данных . Другие источники информации можно найти с помощью Google. При этом могут быть полезны следующие запросы:

1. time series db (time series database)
2. time series repository
3. time series download
4. time series data library
5. (time series) + subject area (meteorology, weather, precipitation, quakes, tide, wind, temperature, solar radiation) - природные временные ряды

Однако данные, полученные из большого числа разнородных источников, могут различаться по формату и способу доступа к ним, поэтому организовать автоматическое скачивание таких рядов затруднительно.

Задача создания базы данных в заданном формате

При создании системы прогнозирования важно использовать определенный формат данных, с которым будут работать все участники проекта. Поэтому необходимо *создать процедуру, приводящую ряды из формата скачивания в установленный формат* (в данном случае, формат ts).

Работа технолога

Выполнение пунктов, выделенных курсивом. После создания базы данных остальные участники проекта тестируют свои алгоритмы на этих данных, при необходимости объединяя ряды из базы, следуя стандарту хранения данных.

Результаты работы технолога

1. Написана программа, позволяющая скачивать ряды с Yahoo Finance по заданному в текстовом файле набору индексов (в заданном промежутке времени): downloadTS.m.
2. Вручную собран список из 210 индексов (потому что нет открытых источников индексов на данном ресурсе): securitiesA.txt.

3. Написана программа, приводящая ряды, скачанные с ресурса, к установленному формату ts: table2ts.m.
4. В результате получены 210 многомерных временных рядов в нужном формате ts. Они загружены на Source Forge: Sources.
5. Создана база в формате .csv, которая включает 16 рядов из разных предметных областей. В базе находится сам ряд и описание в формате .txt. В описании есть ссылка на источник, что может быть полезно для дальнейшего поиска информации. Страница базы на ML: Временной ряд (библиотека примеров).

Подробная классификация временных рядов (с примерами из базы на ML)

Типичные синтетические (слабозашумленные)

— Константа Constant
— Синус Sine
— 2 синуса 2Sines
— Пила Saw
— Трапеция Trapezium

Высокопериодичные

— Потребление электроэнергии EnergyConsumption
— Работа машин и механизмов
— Звук
— Музыка LedZeppelin

Периодичные зашумленные

— Цены на электроэнергию
— Цены на потребительские товары
— Объем сбыта товаров RetailSalesItems
— Цены на сахар SugarPrice
— Цены на хлеб WhiteBreadPrices
— Объем потребления напитков
— Погода: температура, влажность, сила ветра GermanWeather
— Объем пассажирских (и грузо-) перевозок

Со сложным периодом

— Электрокардиограмма ECG
— Пульсовая волна
— Энцефалограмма
— Отраженные волны

Апериодичные

— Распространение гриппа FluUSA
— Миграция населения
— Миграция птиц

Сильно зашумленные

— Цены (объемы) на основные биржевые инструменты Cisco
— Биржевые индикаторы DowJonesIndustrialAverage
— Цены на опционы (по сетке)

Событийные

— Землетрясения ArkansasEarthquakes
— Финансовые пузыри FinancialBubbles
— Рекорды

Unit-тестирование

А. И. Корниенко, Р. Б. Джамтырова, Н. П. Ивкин, Е. Ю. Зайцев
ivkinnikita@gmail.com, raisad90@yandex.ru, zaytsev.yevgen@gmail.com,
aleksej.kornienko@phystech.edu
Москва, Вычислительный Центр РАН

Целью ланной технологической карты является указание последовательности действий при написание систем unit-тестирования и типичные ошибки руководителей проектов.

Порядок работы

Первым делом необходимо добавить папку xunit к списку просматриваемых MatLab'ом. Для этого необходимо воспользоваться коммандой addpath('путь к папке'). Собственно сам процесс, требуемый от нас:

1. Создать отдельную папку для всех unit-тестов для данного проекта.
2. Каждый создаваемый тест должен называться в стиле testfunction или TestFunction (само собой, на место function ставим истинное имя тестируемой функции).
3. Идейная суть задания — подавать на вход функции различные контрольные значения и поверять правильность результата.
4. Для этих целей достаточно использовать всего одну функцию: assertEqual(A, B, message): если A != B — выкрикивает message. В качестве message предлагается взять такую форму — 'FunctionName::ResultError:: FunctionName(A)!=B'. Иногда функции assertEqual будет недостаточно, но для такой ситуации существует документация к пакету MatLab x-unit.
5. Для запуска всех тестов сразу напишем коротенькую функцию:

```
function StartUnitTests
suite = TestSuite.fromName('Название папки со всеми тестами');
suite.run
end
```

6. Запускаем и проверяем.

Типичные ошибки

Основной ощибкой было то, что все руководители перегружали смыслом почти каждую функцию, в то время как unit-тестирование создано для проверки работоспособности простейших, идейно почти не нагруженных технических функций. Второй основной ошибкой было то, что почти все подавали свой код на unit-тестирование, когда работа над кодом была уже завершена, в то время как unit-тестирование предполагается использовать именно во время разработки, а не после.

Системное тестирование

Н. К. Животовский, Д. С. Кононенко
nikita.zhivitovskiy@phystech.edu, daniilkononenko@gmail.com
Москва, Вычислительный Центр РАН

Целью данной технологической карты является указание последовательности действий при выполнении системного тестирования, а также описание основных ошибок руководителей проектов.

Порядок работы

1. Открыть скрипт **algtest-new.m**.
2. В соответствующие поля внести названия алгоритмов, временных рядов, метрик качества.
3. Добавить файлы из п.2 в **path Matlab**
4. Задать **FRC-PROPORTION** — долю объектов выборок, участвующую в обучении.
5. Запустить скрипт. Все найденные числовые значения будут находиться в трехмерной матрице **final-qual**.

Стандартные замечания.

1. Функция не проходит простые тесты.
2. Нет соответствия стандартным интерфейсам.
3. На вход функция требует дополнительные параметры, не имеющие значений по умолчанию.
4. Программа работает настолько долго, что тестирование на большом числе тестовых рядов проблематично.

Результаты тестирования

Семь алгоритмов были протестированы на 121 реальном финансовом ряде. Рассматривалось краткосрочное прогнозирование: длина прогноза примерно 1% длины обучения. Длина обучения в среднем 400 отсчетов. Алгоритмы сравнивались по средней ошибке MSE. На каждом из рядов лучший из алгоритмов получал 6 баллов, следующий — 5, ..., худший — 0 баллов. В таблице (1) приведена сумма баллов, а также среднее время работы алгоритма на ряде.

Таблица 1. Сравнение работы алгоритмов на краткосрочном прогнозировании

Автор	Название алгоритма	Сумма баллов	Среднее время работы на ряде
Илья Фадеев	SSA	198	менее секунды
Дмитрий Сунгуров	Model Selection	92	менее секунды
Алексей Корниенко	Local Forecasting	463	менее секунды
Никита Ивкин	ARIMA	462	менее секунды
Александр Мафусалов	Subseries Superposition Producing	295	35 секунд
Михаил Кузнецов	Kernel Smoothing	503	120 секунд
Никита Животовский	Exponential Smoothing	528	менее секунды

Видно, что по сумме баллов вперед вырываются четыре алгоритма с примерно одинаковыми результатами: Local Forecasting, ARIMA, Kernel Smoothing, Exponential Smoothing. Три из этих алгоритмов также работают быстро.

Также четыре алгоритма были протестированы на долгосрочном прогнозировании. На тех же финансовых рядах длина прогноза составляла примерно 10% обучения. Баллы — от 3 до 0. Результаты — таблица (2).

Таблица 2. Сравнение работы алгоритмов на долгосрочном прогнозировании

Автор	Название алгоритма	Сумма баллов	Среднее время работы на ряде
Илья Фадеев	SSA	85	менее секунды
Алексей Корниенко	Local Forecasting	221	менее секунды
Никита Ивкин	ARIMA	205	менее секунды
Никита Животовский	Exponential Smoothing	215	менее секунды

На долгосрочном прогнозировании алгоритмы Local Forecasting, ARIMA и Exponential Smoothing работают значительно лучше, чем SSA.

Профайлер MATLAB

Р. Б. Быстрый

Профайлер в Matlab

Инструкция по использованию профайлера в Matlab:
`http://www.mathworks.com/help/techdoc/ref/profile.html`.

Перечислим основные моменты, над которыми стоит задуматься при использовании профайлера.

1. Необходимо запускать профайлер на примерах время работы которых составляет хотя бы пару минут.
2. Полезно оценить зависимость времени исполнения функции от длины входных данных, а также оценить распределение времени выполнения между модулями.
3. Необходимо следить, что бы основные функции тестировались на ненулевых данных (так как операции с 0 происходят несравненно быстрее).

Поиск способа ускорения кода

Профайлер укажет «проблемные» места, но не подскажет как их ускорить. Поиск путей ускорения следует основывать на двух идеях. Во-первых, Matlab — интерпретируемый язык, а значит лишен стандартной оптимизации, которая выполняется на этапе компиляции. Во-вторых, нужно обратить внимание на стандартные ошибки программиста Matlab.

Стандартный набор оптимизационных шагов компилятора можно прочитать в документации к gcc или icc:

`http://gcc.gnu.org/onlinedocs/`.

Основные ошибки программиста (основываясь на опыте просмотренных работ):

— необходимо контролировать операции стоящие в циклах и выносить за цикл всё что можно вычислить вне него.

— следует избегать динамических изменений размеров структур. Рекомендуется сначала создать структуру нужного размера, заполненную нулями, а потом её заполнять, а не динамически добавлять элементы.

— не использовать сложные cell() структуры без необходимости.

Работа технолога

Далее приведена последовательность действия технолога.

1. Запускаем unit-тест.
2. Запускаем профайлер, находим «проблемные» модули.
3. Исправляем код.
4. Запускаем профайлер, оцениваем результаты своей деятельности.
5. Запускаем unit-тест, контроль корректности работы программы.