Технологические карты разработки библиотеки алгоритмов прогноза временных рядов

А. Н. Фирстенко, Д. С. Кононенко, М. П. Кузнецов, А. А. Морозов, Д. С. Сунгуров, Н. А. Савинов, А. И. Корниенко, Р. Б. Джамтырова, Н. П. Ивкин, Е. Ю. Зайцев, Н. К. Животовский, Д. С. Кононенко, Р. Б. Быстрый
alexnickfirst@gmail.com

В нижеприведенном документы приведены технологическое рекомендации по созданию программных систем интеллектуального анализа данных. Рекомендации были собраны при разработке библиотеки алгоритмов прогнозирования временных рядов. В частности, вошли рекомендации по созданию метаописаний временных рядов, визуализации прогноза, стилевой правке кода, созданию базы временных рядов, unit-тестированию, системному тестированию и профилированию.

Метаописание временных рядов

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. Для прогнозирования временных рядов существует большое количество алгоритмов [4, 3]. 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 [2].

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:

$$a_{\text{opt}} = \arg\min_{a \in A} L(s, a).$$

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[1]. 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.

Таблица 1. Результаты сравнения алгоритмов

	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

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.

Визуализация прогноза временных рядов

Целью данной технологической карты является описание работы функции визуализации PlotTS.

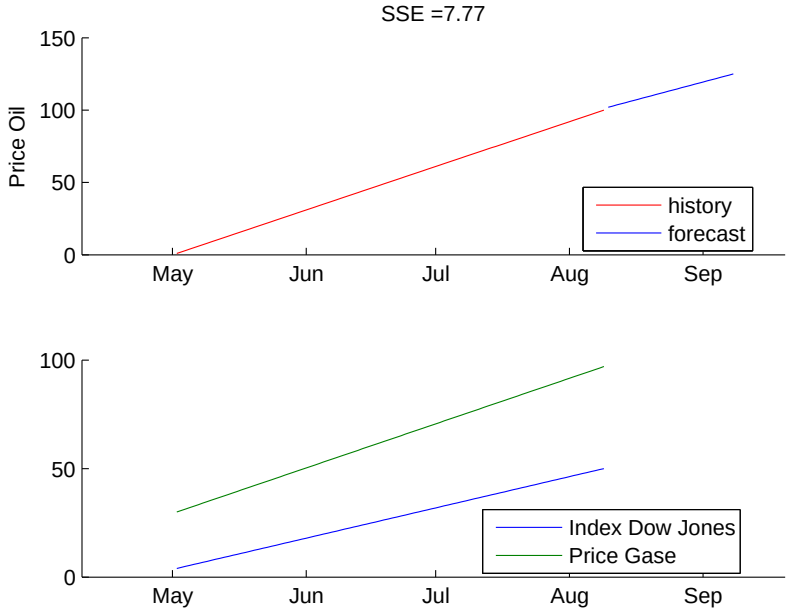

Рис. 1. Пример работы PlotTS

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).

Стилевая правка кода

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);

Создание базы данных многомерных временных рядов для задач прогнозирования

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 http://www.mathworks.com/help/toolbox/datafeed/f9320.html 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.

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).

Типичные синтетические (слабозашумленные) временные ряды

— Константа
https://mlalgorithms.svn.sourceforge.net/svnroot/mlalgorithms
/TSForecasting/TimeSeries/Sources/constants.mat
— Синус
https://mlalgorithms.svn.sourceforge.net/svnroot/mlalgorithms
/TSForecasting/TimeSeries/Sources/sines.mat
— 2 синуса
https://mlalgorithms.svn.sourceforge.net/svnroot/mlalgorithms
/TSForecasting/TimeSeries/Sources/2sines.mat
— Пила
https://mlalgorithms.svn.sourceforge.net/svnroot/mlalgorithms
/TSForecasting/TimeSeries/Sources/saws.mat
— Трапеция
https://mlalgorithms.svn.sourceforge.net/svnroot/mlalgorithms
/TSForecasting/TimeSeries/Sources/trapezia.mat

Высокопериодичные временные ряды

— Потребление электроэнергии
https://mlalgorithms.svn.sourceforge.net/svnroot/mlalgorithms
/TSForecasting/TimeSeries/Sources/tsEnergyConsumption.csv
— Работа машин и механизмов
— Звук
— Музыка
https://mlalgorithms.svn.sourceforge.net/svnroot/mlalgorithms
/TSForecasting/TimeSeries/Sources/tsLedZeppelin.csv

Периодичные зашумленные временные ряды

— Цены на электроэнергию
— Цены на потребительские товары
— Объем сбыта товаров
https://dmba.svn.sourceforge.net/svnroot/dmba/Data/RetialSalesItems.csv
— Цены на сахар
https://mlalgorithms.svn.sourceforge.net/svnroot/mlalgorithms
/TSForecasting/TimeSeries/Sources/tsSugarPrice.csv
— Цены на хлеб
https://dmba.svn.sourceforge.net/svnroot/dmba/Data/WhiteBreadPrices.csv
— Объем потребления напитков
— Погода: температура, влажность, сила ветра
https://mlalgorithms.svn.sourceforge.net/svnroot/mlalgorithms
/TSForecasting/TimeSeries/Sources/tsGermanWeather.csv

— Объем пассажирских (и грузо-) перевозок

Со сложным периодом

— Электрокардиограмма
https://mlalgorithms.svn.sourceforge.net/svnroot/mlalgorithms
/TSForecasting/TimeSeries/Sources/tsEcg.csv
— Пульсовая волна
— Энцефалограмма
— Отраженные волны

Апериодичные временные ряды

— Распространение гриппа
https://mlalgorithms.svn.sourceforge.net/svnroot/mlalgorithms
/TSForecasting/TimeSeries/Sources/tsFluUSA.csv
— Миграция населения
— Миграция птиц

Сильно зашумленные временные ряды

— Цены (объемы) на основные биржевые инструменты
https://mlalgorithms.svn.sourceforge.net/svnroot/mlalgorithms
/TSForecasting/TimeSeries/Sources/tsCSCO.csv
— Биржевые индикаторы
https://mlalgorithms.svn.sourceforge.net/svnroot/mlalgorithms
/TSForecasting/TimeSeries/Sources/tsDJIA.csv
— Цены на опционы (по сетке)

Событийные

— Землетрясения https://mlalgorithms.svn.sourceforge.net/svnroot/mlalgorithms
/TSForecasting/TimeSeries/Sources/tsEarthquakesArkansas.csv
— Финансовые пузыри https://mlalgorithms.svn.sourceforge.net/svnroot/mlalgorithms
/TSForecasting/TimeSeries/Sources/tsFinancialBubbles.csv
— Рекорды

Unit-тестирование

Порядок работы. Необходимо добавить папку xunit к списку просматриваемых MatLab'ом. Для этого необходимо воспользоваться командой 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 ошибкой 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

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 (2) приведена сумма баллов, а также среднее время работы алгоритма на ряде.

Таблица 2. 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 (3).

Таблица 3. 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

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.

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.

Литература

[1] https://mlalgorithms.svn.sourceforge.net/svnroot/mlalgorithms/TSForecasting/TSMetaDescription/.

[2] К. В. Воронцов *Машинное обучение (курс лекций)*, 2009.

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

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