УДК 004.65, 004.032.26 Дата подачи статьи: 11.08.16
DOI: 10.15827/0236-235X.030.1.012-020 2017. Т. 30. № 1. С. 12–20

РАЗРАБОТКА СИСТЕМЫ ХРАНЕНИЯ АНСАМБЛЕЙ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ

Е.В. Пучков, к.т.н., доцент, puchkoff@i-intellect.ru;
С. Терехов, магистрант, isergeiterehov@gmail.com
(Академия строительства и архитектуры
Донского государственного технического университета,
ул. Социалистическая, 162, г. Ростов-на-Дону, 344022, Россия)

Важным инструментом в работе специалиста по анализу данных и машинному обучению является ПО для организации экспериментов. Прежде всего это связано с большим количеством этапов в обработке данных и спецификой их выполнения. В ходе работы был спроектирован и разработан прототип системы хранения ансамблей нейросетевых моделей, обеспечивающий структурированное хранение данных на различных этапах решения задачи прогнозирования временных рядов. Рассмотрены модель данных, архитектура системы хранения и механизмы поступления и перераспределения информации в ней. Разработана модель классов для программного взаимодействия с хранилищем. Для хранения данных об объектах и связей между этими объектами была использована MySQL, а для хранения временных рядов – нереляционная БД InfluxDB. Создан пользовательский интерфейс с возможностями наглядного отображения данных и удобного взаимодействия с хранилищем ансамблей нейросетевых моделей. Апробация системы проводилась на примере задачи прогнозирования солнечной активности за период с января 1700 года по февраль 2015 года. Проведенный эксперимент с применением рекуррентной сети LSTM показал, что ошибка ансамбля нейросетевых моделей ниже ошибки каждой отдельно взятой нейросетевой модели. LSTM построена с применением библиотеки Keras, для формирования ансамбля использован подход Blending.

Результаты проделанной работы показывают перспективность разработки, обеспечивающей высокую степень интеграции в расширяемые программные продукты на языке Python. Разработка полнофункциональной системы позволит не только организовать процесс анализа данных, но и повысить качество результирующих моделей за счет автоматизации процесса формирования ансамблей.

***Ключевые слова**: хранилище, нереляционная БД, рекуррентные нейронные сети, LSTM, ансамбль, stacking, прогнозирование временных рядов.*

В последнее время среди специалистов по анализу данных и машинному обучению все более популярным становится ПО для организации исследований. Прежде всего это связано с большим количеством этапов обработки данных и спецификой их выполнения. Можно выделить такую библиотеку, как Sacred [1], которая позволяет организовать эксперименты без привязки к конкретным моделям, данные параметров моделей и результаты можно сохранить в БД. В библиотеке Hyperopt [2] акцент делается на оптимизации параметров моделей. FGLab [3] позволяет аналитику запускать свои модели на распределенной системе с возможностью сохранять результаты экспериментов и их параметры в БД. Для сложных вычислительных задач с применением Hadoop, которые могут длиться дни или недели, подойдет Luigi [4]. Данный пакет позволяет организовать управление многочисленными вычислительными задачами в одном месте. Последние две системы имеют интерфейс для визуализации результатов и информации по задачам.

Заключительным этапом в решении задачи машинного обучения является построение ансамбля моделей, поскольку в некоторых случаях оптимальное решение может быть получено с применением ансамбля нескольких различных моделей. Большое количество источников показывают практическую значимость применения ансамбля в решении прикладных задач [5–7]. Очень часто в таких ансамблях используют нейросетевые модели. Примечательно, что построение ансамбля только из нейросетевых моделей в некоторых задачах дает преимущество [8–10]. В связи с этим возникает проблема хранения данных на этапах моделирования, в том числе данных самих моделей и построенных с их помощью ансамблей. Проведенный обзор систем организации экспериментов показал, что существующие системы не решают такую проблему в явном виде.

Цель данной работы – проектирование и разработка системы хранения ансамблей нейросетевых моделей, обеспечивающей структурированное хранение данных на различных этапах решения задач прогнозирования временных рядов. Разработка хранилища позволит не только организовать процесс анализа данных, но и повысить качество результирующих моделей за счет автоматизации процесса формирования ансамблей.

Работу можно разделить на следующие основные части:

– разработка модели данных и архитектуры системы хранения;
– разработка пользовательского интерфейса;
– тестирование системы на реальных данных.

Для задач прогнозирования временных рядов принято использовать два типа ИНС: рекуррентные сети (RNN) [11] и различные сети прямого распространения с задержкой по времени (TLFN) [12]. Задержку по времени также можно применять и для рекуррентных сетей [13]. В работе при реше-

нии задачи прогнозирования временного ряда будет использована LSTM (long short-term memory – долгая краткосрочная память). Рекуррентные нейронные сети, основанные на этом подходе, получили большое распространение при решении задач распознавания рукописного текста, моделирования языка, машинного перевода, обработки аудио- и видеоизображений, анализа тональности и классификации текстов, прогнозирования временных рядов.

При решении сложных задач классификации, регрессии, а также прогнозирования временных рядов часто оказывается, что ни один из алгоритмов не обеспечивает желаемого качества восстановления зависимости. В таких случаях имеет смысл строить композиции алгоритмов (ансамбли), в которых ошибки отдельных алгоритмов взаимно компенсируются. Для задачи прогнозирования временных рядов подойдут такие подходы, как голосование и стекинг (stacking). Они подразумевают формирование ансамбля из моделей, полученных на одинаковых данных, что подходит для временных рядов, в отличие от бустинга (boosting) и бэггинга (bagging), где для базовых алгоритмов используются разные данные.

Наиболее известные корректирующие операции голосования:

– простое:

$$b(x) = F\left(b_1(x), \ldots, b_T(x)\right) = \frac{1}{T}\sum_{t=1}^{T} b_t(x);$$

– взвешенное:

$$b(x) = F(b_1(x), ..., b_T(x)) =$$
$$= \sum_{t=1}^{T} w_t b_t(x), \ = \sum_{t=1}^{T} w_t b_t(x), \sum_{t=1}^{T} w_t = 1, \ w_t \geq 0;$$

– смесь экспертов:

$$b(x) = F(b_1(x), ..., b_T(x)) =$$
$$= \sum_{t=1}^{T} w_t(x) b_t(x), \sum_{t=1}^{T} w_t = 1, \ \forall x \in X.$$

Простое голосование – это лишь частный случай взвешенного голосования, а взвешенное является частным случаем смеси экспертов.

Основная идея стекинга и его разновидности блендинга заключается в использовании базовых алгоритмов для получения предсказаний (метапризнаков) и использовании их как признаков для некоторого обобщающего алгоритма (метаалгоритма). Иными словами, основной идеей стекинга является преобразование исходного пространства признаков задачи в новое пространство, точками которого являются предсказания базовых алгоритмов [14].

Разработка модели данных и архитектуры системы хранения

Для реализации поставленных задач необходим следующий набор программных средств:

– реляционная БД для хранения данных об объектах и связей между этими объектами;

– нереляционная БД для хранения временных рядов;

– язык программирования для реализации логики системы хранения;

– сопутствующие программные пакеты, в том числе реализующие LSTM.

В качестве реляционной СУБД была использована MySQL [15]. Для хранения временных рядов современное решение – InfluxDB [16]. Для программирования логики хранилища нейросетевых моделей выбраны Python версии 2.7.11 и следующие свободно распространяемые пакеты:

– numpy (для работы с массивами [17]);

– sklearn (библиотека для анализа данных [18]);

– cherrypy (библиотека, позволяющая реализовать веб-сервер [19]).

Среди многочисленных программных реализаций архитектур нейронных сетей, в частности LSTM, выделим Theano [20], а также созданную на ее основе библиотеку Keras [21]. Библиотека Keras позволяет использовать как Theano, так и TensorFlow [22] в качестве основы вычислений. Keras упрощает процесс создания нейронных сетей, предоставляя для этого специальный конструктор. В основе любого кода с использованием Keras лежит объект model, который описывает то, в каком порядке и какие именно слои содержит ваша нейронная сеть.

Для построения структуры реляционной БД рассмотрим необходимые сущности и их структуру [23].

Проект (project) объединяет ряд исследований над набором данных. В рамках проекта рассматриваются данные из определенного источника (временной ряд, который хранится в InfluxDB). Все действия по преобразованию данных, построению моделей или ансамблей производятся в рамках проекта.

Источник данных (data_source) представляет собой описание данных в источнике. В рамках хранилища рассматривается основная задача – прогнозирование временных рядов. Соответственно, информация об источнике данных включает такую информацию, как начало периода, конец периода, интервал измерений и другие. В связи с тем, что источник данных не фиксирован, то есть данные в нем могут изменяться, дополняться, удаляться, необходимо фиксировать состояние источника данных на момент начала какого-либо исследования или ряда исследований.

Снимок данных (data_snapshot) отражает состояние источника данных на момент времени. Однако сами данные в исходном виде, как правило, не пригодны для построения качественных моделей, поэтому необходимо выполнить ряд преобразований.

Преобразование данных (data_preparation) показывает способ преобразования данных (снимка данных), а также сохраняет преобразованные данные для дальнейшего применения. Преобразованные данные по-прежнему являются временным рядом, но для использования в различных нейросетях должны быть созданы конечные наборы данных в виде матриц X и Y, объясняющие признаки и целевые значения.

Набор данных (data_set) – это конечная выборка данных, отвечающая требованиям той или иной модели. Например, одна модель может использовать для прогнозирования окно в 7 значений, а целевое (прогнозируемое) значение будет отступать на 2 пункта от окна. В этом случае размерность матрицы X – 7, а Y формируется по определенному правилу. При других параметрах выборка будет сформирована иначе, что и объясняет необходимость введения рассматриваемой сущности.

Снимок данных, преобразование данных и набор данных – это отдельные наборы данных, пошагово полученные из предыдущего источника. Эти данные уже необязательно являются временными рядами с точки зрения способа их хранения. В связи с этим необходимо организовать хранение этих данных в унифицированном виде. Наиболее подходящим форматом хранения является CSV (Comma-Separated Values – разделяемые запятыми значения) – текстовый формат, предназначенный для представления табличных данных. Определим также сущность CSV-данных (data_csv) – это зависимая сущность, которая представляет собой только сами данные под уникальным идентификатором.

Другим не менее важным набором сущностей является набор, связанный с нейросетевыми моделями.

Исследование (research) – это группа моделей, полученных по определенным правилам. Такие правила устанавливают порядок преобразования данных, способ построения моделей и их настройки и т.д. В рамках исследования рассматриваются данные, преобразованные определенным образом, поэтому все модели, построенные в ходе исследования, с точки зрения представления работают с одними и теми же данными.

Модель (model) – это представление математической модели. Модель может быть любого типа: как нейросетевой, так и любой другой (случайный лес, логистическая регрессия и др.). Для получения модели данные должны быть подготовлены определенным образом, как говорилось ранее, и сохранены как набор данных. Такой набор и используется далее для обучения и тестирования модели.

В ходе описания настроек или построения модели могут возникать некоторые данные, так или иначе описывающие модель. Они называются метаданными и требуют вынесения в определенную сущность (для реляционной БД).

Метаданные модели (model_meta) – это простое представление данных о модели в виде «ключ-значение». Такие данные могут содержать настройки модели и/или данные о процессе обучения (ошибка, доля правильных ответов, время обучения, алгоритм обучения и другие).

Последним набором сущностей являются сущности, связанные с ансамблями (комитетами).

Ансамбль (ensemble) – это сущность, создаваемая в рамках одного проекта. Такое ограничение вводится для ограничения данных: все модели должны работать на данных одного и того же рода. Здесь описываются такие данные об ансамбле, как метод построения ансамбля, тип метамодели (метаклассификатора), математическая модель ансамбля и т.д.

Ансамбль составляется из моделей. При этом каждая модель может содержать ряд определенных параметров с точки зрения ансамбля, например, вес эксперта для линейной регрессии.

Элемент ансамбля (ensemble_item) – параметризованная модель, используемая в построении ансамбля.

Физическая модель данных в MySQL представлена на рисунке 1. Для связи реляционной БД MySQL и нереляционной InfluxDB необходимо ввести ряд спецификаций:

– измерение (measurement), содержащее экспортируемую информацию, должно иметь имя data_source;

– измерение обязательно должно включать тэг (tag) mysql_id, содержащий идентификатор исходных данных, куда будет произведена привязка;

– тип данных значения (value) должен быть float-числом с плавающей точкой.

Измерение создается автоматически при добавлении новых данных. Рассмотрим пример добавления данных в необходимое измерение по установленным правилам: data_source,mysql_id=234 value==0.55 1422568543702900257. Такой запрос добавит в БД InfluxDB запись в необходимое измерение для источника данных с идентификатором 234. Запись будет содержать значение 0.55 и привязку ко времени со значением 1422568543702900257 (timestamp) – Thu, 29 Jan 2015 21:55:43.702900257 GMT.

Одна из основных задач хранилища – предоставление функционального программного интерфейса для взаимодействия с данными, хранящимися в БД. Поэтому, помимо средств хранения данных (MySQL, InfluxDB), хранилище ансамблей нейросетевых моделей включает внутреннюю логику, определяющую правила функционирования системы.

Рассмотрим каждый пакет из представленных на рисунке 2:

– enstorage (полная библиотека хранилища, включающая в себя основные модели объектов, отражающие сущности БД (ORM, Object-Relational Mapping));

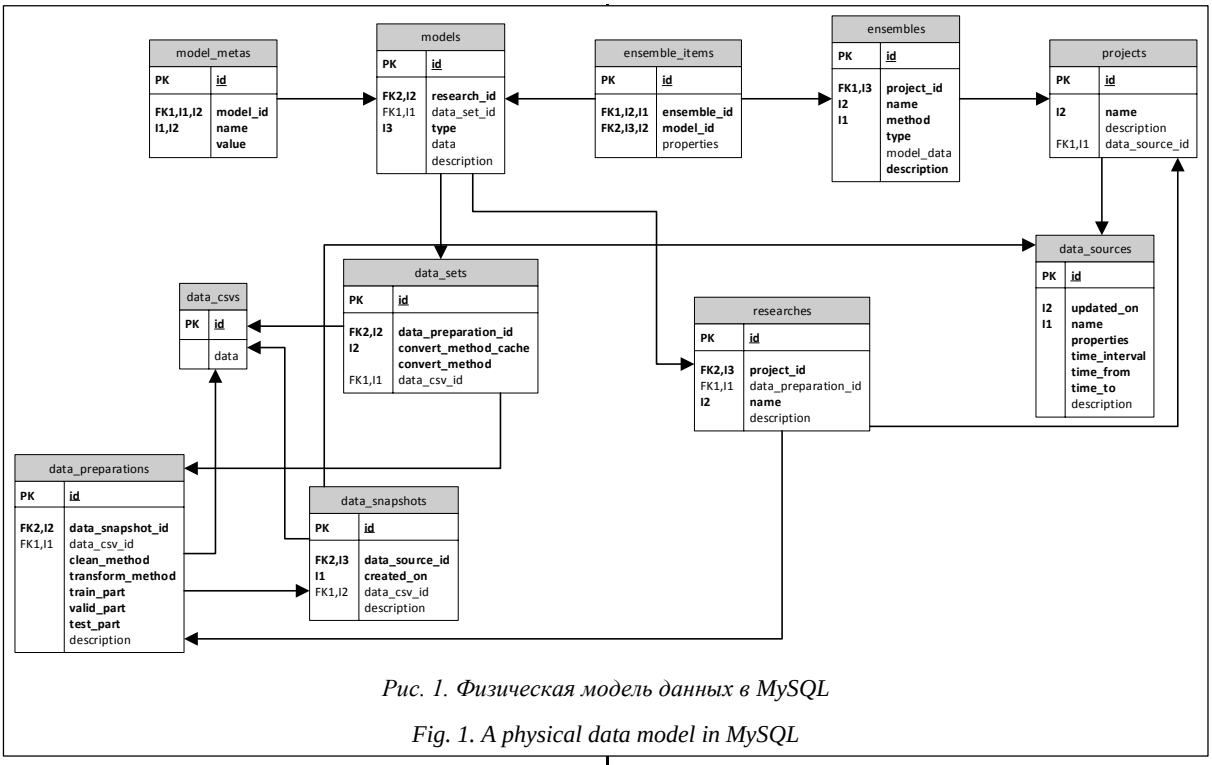

Рис. 1. Физическая модель данных в MySQL

Fig. 1. A physical data model in MySQL

– adapter (библиотека, реализующая методы преобразования данных для дальнейшего использования в моделях);

– enmyadmin (содержит основной функционал встроенной системы администрирования хранилища).

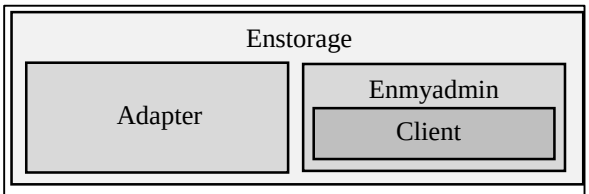

Рис. 2. Общая схема пакетов хранилища

Fig. 2. A common pattern of storage packages

Отдельным классом, который обязательно должен быть использован перед работой с хранилищем ансамблей нейросетевых моделей, является Connector (рис. 3). Он осуществляет подключение к необходимым БД (MySQL и InfluxDB). Все остальные классы являются компонентами ORM и реализуют следующие стандартные public (доступные извне) функции:

– delete – удаление связанного с БД объекта;

– save – сохранение (создание или обновление) объекта БД;

– get – статичный метод, возвращающий объект класса, к которому он относится, по указанному идентификатору записи (id);

– get_list – статичный метод, возвращающий список объектов класса, в котором вызван, по указанным идентификаторам (ids); если идентификаторы не указаны, возвращается полный список всех объектов.

С течением времени исходные данные в InfluxDB могут обновляться и пополняться. Для фиксации определенного набора данных необходимо создать снимок (DataSnapshot) (рис. 4).

Снимок создается при помощи метода DataSource.create_snapshot(), который загружает текущее состояние источника на указанный временной период (DataSource: time_from, time_to). Загрузка данных выполняется с применением агрегирующей функции, группирующей данные по временному интервалу, – DataSource.time_interval. Сохраненный снимок выступает в роли самостоятельных данных, которые могут быть использованы для дальнейших исследований.

Для подготовки данных создается объект DataPreparation, который обеспечивает хранение подготовленных данных, а также преобразование данных DataSnapshot. Создание преобразованных данных выполняется при помощи метода DataSnapshot.create_preparation(clean_method, transform_method, train_part, valid_part, test_part), где параметрами являются (по порядку) метод заполнения пустых значений, метод преобразования значений, доля обучающей выборки, доля валидационной выборки, доля тестовой выборки. Подготовка данных в ручном режиме осуществляется методом DataPreparation.prepare().

После преобразования данных на их основе может быть создано исследование – Research. При создании модели Research не определяет, с каким набором работает модель, это задача сервиса, использующего хранилище.

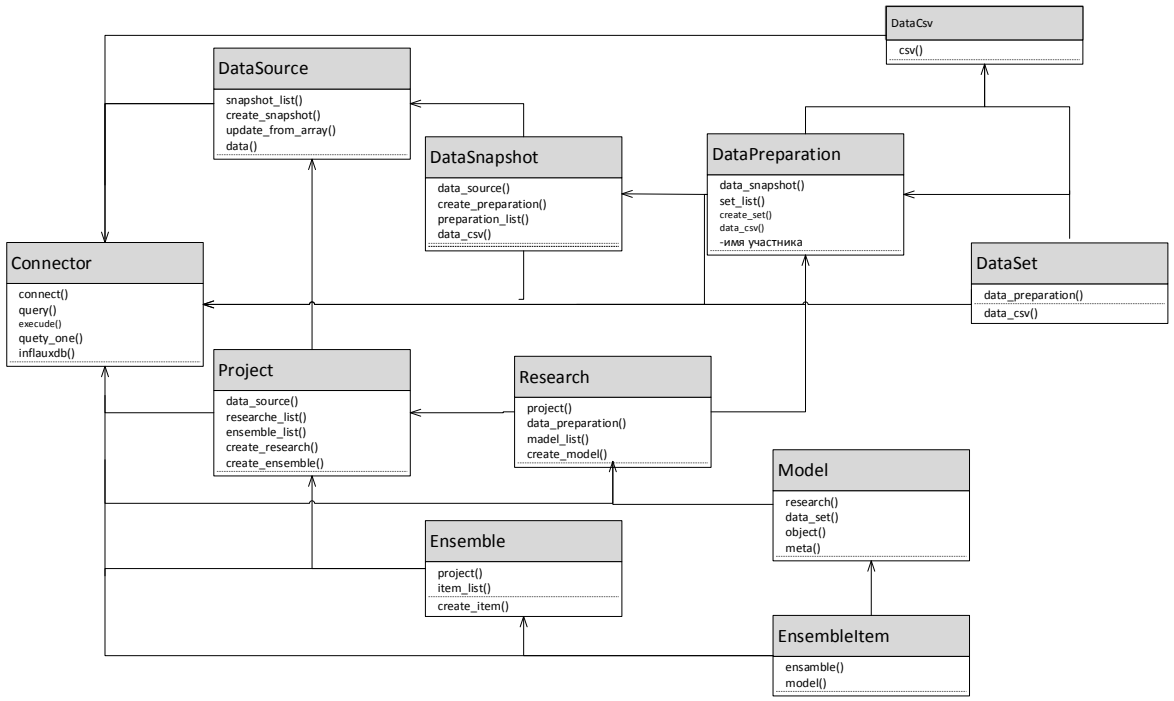

Рис. 3. Структура классов хранилища enstorage

Fig. 3. The structure of enstorage classes

Сервис должен произвести следующие действия:

– создать модель, привязанную к исследованию;

– исходя из типа модели преобразовать нужным адаптером (adapter) данные – DataPreparation.create_set(adapter);

– сообщить модели о созданном наборе данных посредством Model.data_set(created_data_set).

При этом в автоматическом режиме адаптером будут обработаны преобразованные данные DataPreparation. Данный этап является завершающим для серии преобразования исходных данных, полученных из DataSource.

Еще одним важным потоком данных является информация, поступающая в ходе построения моделей и ансамблей. Фиксация таких данных (метаданных) для модели осуществляется методами Model.meta(key, value). Метаданными могут быть абсолютно любые данные, описывающие модель. При построении ансамбля дополнительные параметры фиксируются в свободной форме в EnsembleItem.properties, однако рекомендуется использовать формат JSON.

Особенностью хранилища является то, что хранение объектов конечных реализаций моделей осуществляется благодаря специальному формату. Данные объектов сериализуются и десериализуются при помощи библиотеки pickle. Такой подход обеспечивает возможность сохранения и восстановления объектов целиком, тем самым обеспечивая высокий уровень интеграции пакета enstorage с другими библиотеками. Также, благодаря используемому формату, хранилище может принять не только нейросетевые модели, но и любые другие. Однако из-за использования библиотеки pickle существует ограничение на использование этих дан-

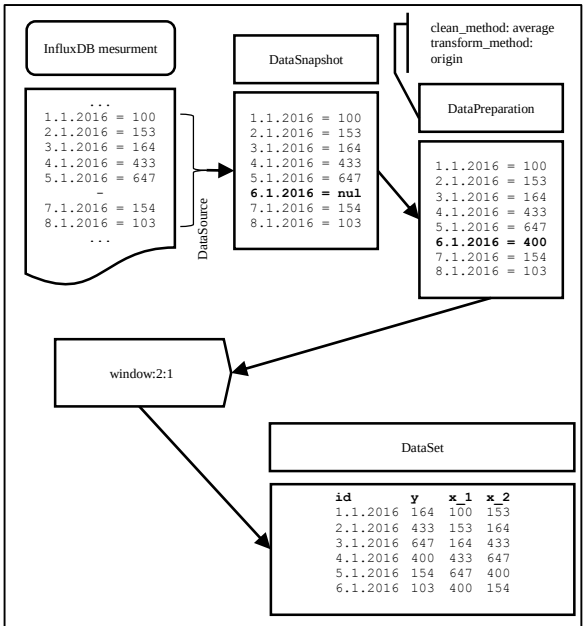

Рис. 4. Поступление и конвертация данных в хранилище

Fig. 4. Data occurrence and conversion in a storage

ных в языках, отличных от Python, так как данные совместимы только с ним.

Разработка пользовательского интерфейса

Для упрощения администрирования хранилища ансамблей нейросетевых решений предусмотрен пакет enmyadmin, входящий в enstorage. Данный пакет представляет собой веб-сервер с основными методами администрирования. Основным шаблоном проектирования веб-приложения является model-view-controller (MVC, «модель-представление-контроллер»). В роли клиентского приложения выступает HTML5-JS-приложение, разработанное с использованием AngularJS [24]. Взаимодействие клиентского и серверного приложений осуществляется по технологии REST (representational state transfer – «передача состояния представления»), обеспечивающей независимость серверной части от клиентского приложения. Фреймворк работает с HTML, включающим дополнительные пользовательские атрибуты, которые описываются директивами, и связывает ввод-вывод области страницы с моделью, состоящей из объектов JavaScript. Значения этих объектов задаются вручную или извлекаются из статических или динамических JSON-данных.

На рисунке 5 представлена форма просмотра информации о проекте. В левой части формы находится фиксированная панель навигации, позволяющая просматривать список проектов и источников данных для быстрого перехода к ним. Для поиска необходимого пункта предусмотрен функционал фильтрации. В правой части окна находится область управления, включающая элементы управления открытым объектом.

Разработаны следующие формы управления объектами: проект, исследование, модель, ансамбль, исходные данные, снимок данных, преобразованные данные, набор данных.

Важно отметить, что благодаря использованию технологии REST клиентское приложение системы администрирования может быть разработано на любой платформе, поддерживающей взаимодействие по HTTP-протоколу.

Тестирование системы на реальных данных

Для тестирования работоспособности хранилища в реальных задачах необходимо реализовать систему построения ансамблей, а также обучения моделей. Конструктор ансамблей – это отдельный функционал, который может быть вынесен в специальный пакет endirector. Данный пакет включает методы построения ансамбля и использует объекты хранилища из пакета enstorage. Ансамбль формируется в автоматическом режиме на основе настроек в Ensemble. Далее приведен пример построения ансамбля из набора моделей, а также применения Conductor для формирования метамодели:

```
project = enstorage.Project.get(1)
models = enstorage.Model.get_list([3,4,5])
# Настройка ансамбля
ensemble = project.create_ensemble('blending', 'LinearRegression', models=models)
# Создание ансамбля
conductor = endirector.Conductor(ensemble)
conductor.render()
```

Порядок действий для построения ансамбля, реализованный в методе Conductor.render():

– инициализация объекта метамодели;
– определение метода создания ансамбля;
– создание пересечения матриц данных (DataSet) всех экспертов;

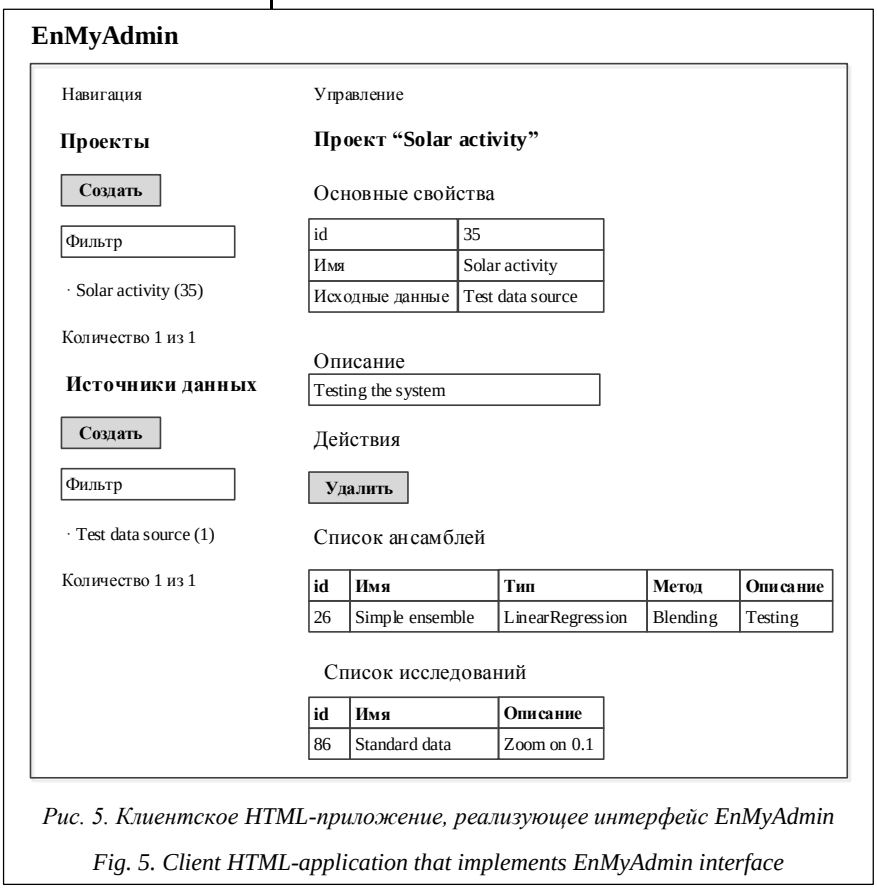

Рис. 5. Клиентское HTML-приложение, реализующее интерфейс EnMyAdmin

Fig. 5. Client HTML-application that implements EnMyAdmin interface

– запуск алгоритма построения ансамбля;
– сохранение метамодели и параметров моделей.

Стоит отметить, что наборы данных различных моделей могут существенно отличаться друг от друга. Так, один набор данных может быть получен с задержкой в 4 значения, а другой – в 10. При этом объем выборок также будет отличаться. Могут использоваться и другие методы конвертации временного ряда, что делает невозможным однозначное получение результата всех моделей для одних данных. Решением данной проблемы является идентификация (*id*) целевых значений. Таким образом, каждый набор [*X, y*] в DataSet также включает и *id* – [*id, X, y*]. Благодаря этому можно получить значения всех моделей для конкретного целевого значения – *y*.

В качестве исходных данных для тестирования системы использованы данные о солнечной активности за период с января 1700 года по февраль 2015 года, всего 303 значения (рис. 6). Для эксперимента построим 3 нейронные сети с задержкой в 5, 7, 13 значений.

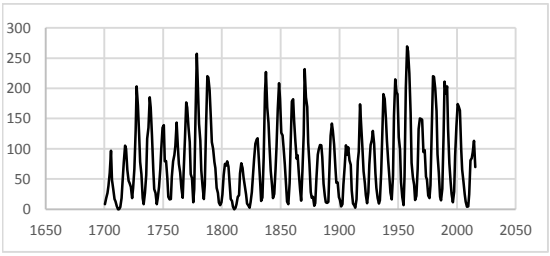

Рис. 6. Среднегодовое количество солнечных пятен

Fig. 6. Sunspot annual average

В ходе выполнения итогового скрипта осуществляются следующие действия:
– подключение к хранилищу;
– получение данных о наборе данных;
– создание снимка данных;
– преобразование данных (масштабирование, выделение тестовой выборки – 30 %);
– создание проекта и исследования;
– создание и инициализация модели;
– обучение моделей;
– создание ансамбля.

Далее приведен код создания модели LSTM с применением библиотеки Keras:

```
model = Sequential()
model.add(LSTM(output_dim=50, input_dim=n_input,
return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(100, return_sequences=False))
model.add(Dropout(0.2))
model.add(Dense(output_dim=1))
model.add(Activation("linear"))
model.compile(loss=item.meta('loss'), optimizer=
'rmsprop')
```

Для оценки качества каждой модели, а также ансамбля рассчитаем среднеквадратическую ошибку (MSE) на тестовой выборке.

На рисунке 7 видим, что наименьшее значение ошибки у ансамбля (на графике – out). Так как в качестве метамодели использовалась линейная регрессия (Linear Regression), вес каждой модели можно оценить в результирующем значении. Данные значения записаны в EnsembleItem.properties: 1.06192747 – keras-1, –0.13015426 – keras-2, 0.17757505 – keras-3.

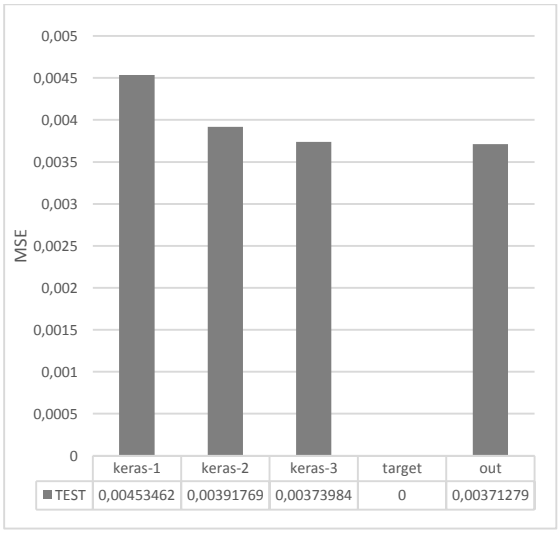

Рис. 7. Среднеквадратическая ошибка моделей и ансамбля на тестовой выборке

Fig. 7. Mean square error of models and an ensemble on a test set

Значения весов можно интерпретировать следующим образом: наибольшим весом обладает первая модель (keras-1), небольшую корректировку вносит третья модель (keras-3), компенсацию оказывает вторая модель (keras-2). На рисунке 8 представлены результаты прогноза, полученные с помощью ансамбля.

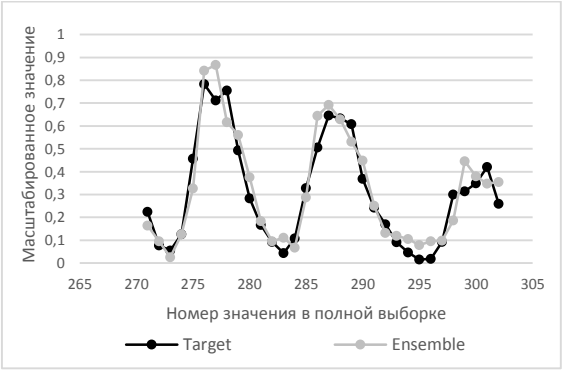

Рис. 8. График прогноза солнечной активности на тестовой выборке

Fig. 8. A solar activity forecast diagram on a test set

Заключение

В ходе выполнения данной работы был создан прототип системы хранения ансамблей нейросетевых моделей. Проведенный эксперимент по прогнозированию солнечной активности показал, что ошибка ансамбля нейросетевых моделей ниже ошибки каждой отдельно взятой нейросетевой модели. Несомненно, для улучшения результатов прогнозирования необходимы дополнительные эксперименты и совершенствование ПО.

Разработаны следующие программные решения:

– пакет для языка Python, обеспечивающий быстрое и упрощенное взаимодействие с БД, реализованный с использованием технологии ORM;

– пакет преобразования временного ряда в конечные выборки, применяемые в моделях;

– интерфейс пользователя в виде HTML-приложения, обеспечивающий наглядное отображение данных и удобное взаимодействие с хранилищем ансамблей нейросетевых моделей.

Результаты проделанной работы показывают перспективность разработанных программных решений и обеспечивают высокую степень интеграции в расширяемые программные продукты на языке Python.

Исследование выполнено при финансовой поддержке РФФИ, проект № 14-01-00579 а.

Литература

1. Sacread's. URL: http://sacred.readthedocs.io/en/latest/ (дата обращения: 9.05.2016).
2. Hyperopt. URL: http://hyperopt.github.io/hyperopt/ (дата обращения: 12.05.2016).
3. FGLab. URL: https://kaixhin.github.io/FGLab/ (дата обращения: 16.05.2016).
4. Luigi. URL: http://luigi.readthedocs.io/en/stable/ (дата обращения: 11.05.2016).
5. Kaggle Ensembling Guide. URL: http://mlwave.com/kaggle-ensembling-guide/ (дата обращения: 11.05.2016).
6. A deep learning approach with an ensemble-based neural network classifier for black box ICML 2013 Contest. URL: http://deeplearning.net/wp-content/uploads/2013/03/LukaszRomaszko_ICML2013_BlackBox.pdf (дата обращения: 10.05.2016).
7. О проблеме генерации разнообразия ансамблей индивидуальных моделей в задачах идентификации. URL: http://vspu2014.ipu.ru/proceedings/prcdngs/3214.pdf (дата обращения: 13.05.2016).
8. Jing Yang, Xiaoqin Zeng, Shuiming Zhong, Shengli Wu. Effective neural network ensemble approach for improving generalization performance. IEEE Transactions on Neural Networks and Learning Systems, 2013, vol. 24, iss. 6, pp. 878–887.
9. Ensemble of deep convolutional neural networks for learning to detect retinal vessels in fundus images. URL: https://arxiv.org/abs/1603.04833 (дата обращения: 5.05.2016).
10. Learning ensembles of convolutional neural networks. URL: http://theorycenter.cs.uchicago.edu/REU/2014/final-papers/chen.pdf (дата обращения: 10.05.2016).
11. Хайкин С. Нейронные сети: полный курс; [пер. с англ.]. М.: Вильямс, 2006. 1104 с.
12. Belyavsky G., Misyura V., Puchkov E. Prediction intervals for time series using neural networks based on wavelet-core. Far East Jour. of Mathematical Sciences, 2016, vol. 100, iss. 3, pp. 413–425.
13. Mohamed Akram Zaytar, Chaker El Amrani. Sequence to sequence weather forecasting with long short-term memory recurrent neural networks. Intern. Jour. of Comp. Applications, 2016, vol. 143, no. 11, pp. 7–11.
14. Li D., Yu Dong, and Platt J. Scalable stacking and learning for building deep architectures. Acoustics, Speech and Signal Processing, 2012.
15. MySQL. URL: https://www.mysql.com/ (дата обращения: 2.05.2016).
16. InfluxDB. URL: https://influxdata.com/ (дата обращения: 2.05.2016).
17. NumPy. URL: http://www.numpy.org (дата обращения: 3.05.2016).
18. Scikit-learn. URL: http://scikit-learn.org (дата обращения: 3.05.2016).
19. CherryPy. URL: http://www.cherrypy.org (дата обращения: 7.05.2016).
20. Theano. URL: http://deeplearning.net/software/theano/ (дата обращения: 17.05.2016)
21. Keras. URL: https://keras.io/ (дата обращения: 10.05.2016).
22. TensorFlow. URL: https://www.tensorflow.org/ (дата обращения: 13.05.2016).
23. Пучков Е.В., Белявский Г.И. Разработка нейроэмулятора для решения задач прогнозирования и классификации. Ростов н/Д: Из-во РГСУ, 2012. 138 с.
24. AngularJS. URL: https://angularjs.org/ (дата обращения: 13.05.2016).

Software & Systems
DOI: 10.15827/0236-235X.030.1.012-020

Received 11.08.16
2017, vol. 30, no. 1, pp. 12–20

NEURAL NETWORK ENSEMBLES STORAGE DEVELOPMENT

E.V. Puchkov[1], *Ph.D (Engineering), Associate Professor, puchkoff@i-intellect.ru*
S. Terekhov[1], *Graduate Student, isergeiterehov@gmail.com*

[1] *DSTU Academy of Civil Engineering and Architecture,*
Socialisticheskaya St. 162, Rostov-on-Don, 344022, Russian Federation

Abstract. An important tool in the work of a data analysis and machine learning expert is software for an experiment organization. This is primarily related to a large number of stages in data processing and the characteristic aspects of their implementation. In the course of this work the authors have designed and developed a prototype of neural network ensemble storage for data structured storing on various stages of time series forecasting. The article considers a data model, data storage architecture and mechanisms of data acquiring and redistribution in the storage. There is also a description of the developed

class model for software-based interaction with the storage. In order to store data on objects and relationships between these objects there has been used MySQL. For storing time series we used non-relational database InfluxDB. There is also user interface with data visualization and easy interaction with the neural network ensembles storage. The system has been tested using solar activity data in the period from January 1700 to February 2015. The experiment (using LSTM recurrent network) showed that an error of a neural network ensemble was lower than an error of each individual neural network model. LSTM was built using the library Keras, the Blending approach was used to form an ensemble.

The results of this work indicate the prospects of the developed software solution and provide a high degree of integration into scalable Python software. The development of a fully functional system will allow not only organizing the data analysis process, but also improving the performance of resulting models due to ensemble formation process automation.

Keywords: storage, non-relational database, recurrent neural network, LSTM, ensemble, stacking, time series forecasting.

Acknowledgements. *The research has been financially supported by RFBR, project no. 14-01-00579 a.*

References

1. *Sacread's*. Available at: http://sacred.readthedocs.io/en/latest/ (accessed May 9, 2016).
2. *Hyperopt*. Available at: http://hyperopt.github.io/hyperopt/ (accessed May 12, 2016).
3. *FGLab*. Available at: https://kaixhin.github.io/FGLab/ (accessed May 16, 2016).
4. *Luigi*. Available at: http://luigi.readthedocs.io/en/stable/ (accessed May 11, 2016).
5. *Kaggle Ensembling Guide*. Available at: http://mlwave.com/kaggle-ensembling (accessed May 11, 2016).
6. *A Deep Learning Approach with an Ensemble-Based Neural Network Classifier for Black Box ICML 2013 Contest*. Available at: http://deeplearning.net/wp-content/uploads/2013/03/LukaszRomaszko_ICML2013_BlackBox.pdf (accessed May 10, 2016).
7. On the problem of individual model assembly variety generation in identification problems. *XII Vseross. soveshchanie po problemam upravleniya VSPU-2014* [All-Russian Conf. on Management Problems VSPU-2014]. Available at: http://vspu2014.ipu.ru/proceedings/prcdngs/3214.pdf (accessed May 13, 2016).
8. Yang J., Zeng X., Zhong Sh., Wu Sh. Effective Neural Network Ensemble Approach for Improving Generalization Performance. *IEEE Trans. on Neural Networks and Learning Systems*. 2013, vol. 24, iss. 6, pp. 878–887.
9. *Ensemble of Deep Convolutional Neural Networks for Learning to Detect Retinal Vessels in Fundus Images*. Available at: https://arxiv.org/abs/1603.04833 (accessed May 5, 2016).
10. *Learning Ensembles of Convolutional Neural Networks*. Available at: http://theorycenter.cs.uchicago.edu/REU/2014/final-papers/chen.pdf (accessed May 10, 2016).
11. Haykin S.O. *Neural Networks: A Comprehensive Foundation*. Prentice Hall Publ., 2nd ed., 1998, 842 p. (Russ.ed.: Moscow, Vilyams Publ., 2006, 1104 p.).
12. Belyavsky G., Misyura V., Puchkov E. Prediction intervals for time series using neural networks based on wavelet-core. *Far East Jour. of Mathematical Sciences*. 2016, vol. 100, iss. 3, pp. 413–425.
13. Mohamed Akram Zaytar, Chaker El Amrani Sequence to Sequence Weather Forecasting with Long Short-Term Memory Recurrent Neural Networks. *Int. Jour. of Computer Applications*. 2016, vol. 143, no. 11, pp. 7–11.
14. Li D., Yu D., Platt J. Scalable stacking and learning for building deep architectures. *Acoustics, Speech and Signal Processing*. 2012.
15. *MySQL*. Available at: https://www.mysql.com/ (accessed May 2, 2016).
16. *InfluxDB*. Available at: https://influxdata.com/ (accessed May 2, 2016).
17. *NumPy*. Available at: http://www.numpy.org (accessed May 3, 2016).
18. *Scikit-learn*. Available at: http://scikit-learn.org (accessed May 3, 2016).
19. *CherryPy*. Available at: http://www.cherrypy.org (accessed May 7, 2016).
20. *Theano*. Available at: http://deeplearning.net/software/theano/ (accessed May 17, 2016)
21. *Keras*. Available at: https://keras.io/ (accessed May 10, 2016).
22. *TensorFlow*. Available at: https://www.tensorflow.org/ (accessed May 13, 2016).
23. Puchkov E.V., Belyavsky G.I. *Razrabotka neyroemulyatora dlya resheniya zadach prognozirovaniya i klassifikatsii* [Neuroemulator Development to Solve Forecasting and Classification Problems]. Rostov-on-Don, RGSU Publ., 2012, 138 p.
24. *AngularJS*. Available at: https://angularjs.org/ (accessed May 13, 2016).

Примеры библиографического описания статьи

1. Пучков Е.В., Терехов С. Разработка системы хранения ансамблей нейросетевых моделей // Программные продукты и системы. 2017. Т. 30. № 1. С. 12–20; DOI: 10.15827/0236-235X.030.1.012-020.

2. Puchkov E.V., Terekhov S. Neural network ensembles storage development. *Programmnye produkty i sistemy* [Software & Systems]. 2017, vol. 30, no. 1, pp. 12–20 (in Russ.); DOI: 10.15827/0236-235X.030.1.012-020.