Федеральное государственное бюджетное учреждение науки
Институт водных проблем Российской академии наук (ИВП РАН)

«Методы машинного обучения и гибридные модели в гидрологическом прогнозировании»

Практикум

Авторы:

Д.П. Соломатин, IHE Delft Institute for Water Education

В.М. Морейдо, Институт водных проблем РАН

Москва 2020

Оглавление

Часть I. Теоретические основы машинного обучения для гидрологических задач 3
Введение 3
Машинное обучение 4
Обучение, кросс-валидация и верификация 6
Искусственные нейронные сети 7
Линейная регрессия как простейшая ИНС 7
Добавление сложности: слои и связи 8
Обучение ИНС МСП 11
Практические аспекты обучения 12
Нормализация данных для предотвращения паралича сети 13
Количество скрытых узлов 13
Выбор функций активации 13
Выбор алгоритма обучения 13
Критерии эффективности ИНС 13
Выбор входных переменных 14
Часть II. Интерактивное учебное пособие по машинному обучению для гидрологических задач ...15
Введение 15
Требования к обучающемуся 15
Технические требования 15
Подключение данных 15
Обратите внимание 16
1. Первая модель машинного обучения: множественная линейная регрессия 16
1.1 Объект исследования 16
Исходные данные 17
Задача 18
1.2 Подготовка данных для моделирования 18
1.3 Построение модели множественной линейной регрессии 21
2. Вторая модель машинного обучения: искуственная нейронная сеть (ИНС) 22
3. Оптимизация параметров ИНС с помощью сеточного поиска с кросс-валидацией 25

Часть I. Теоретические основы машинного обучения для гидрологических задач

Введение

Модель можно определить как упрощенное представление реальности с целью ее описания или предсказания. В зависимости от цели, желаемой точности, имеющихся данных и других факторов в моделях могут использоваться самые разные способы представления.

С точки зрения расчетных задач термин «модель» традиционно используется в одном из двух значений:

- (a) математическая модель, основанная на описании изучаемого явления или системы, именуемая также концептуальной или физико-математической моделью;
- (b) физико-статистические модели, основанные на собранных фактических данных, описывающих некоторые аспекты поведения системы.

Отдельный класс моделей основан на данных, собранных на основе длительных наблюдений за естественной системой или процессом. Физико-статистические модели попадают в эту категорию; они были и широко используются в различных науках. Они обладают как определенными преимуществами, так и серьезными недостатками. Одним из наиболее важных недостатков является то, что они основаны на определенных предположениях (часто довольно строгих) о подчинении исходных данных определенным вероятностным распределениям, которые зачастую не выполняются. В 1950-60-е годы начали набирать популярность новые подходы в моделировании. Они не основывались на строгих допущениях статистических моделей и в зависимости от области применения получили названия «анализ данных», «распознавание образов» и «автоматическая классификация».

В результате исследования механизмов, которыми человеческий мозг обрабатывает данные и информацию, были достигнуты важные результаты в области создания искусственного интеллекта. Среди наиболее важных следует отметить теорию обработки неопределенностей в данных не вероятностным способом (нечеткие множества) и способы работы с функциями, которые не выражаются аналитически, но могут быть вычислены - их аппроксимация (искусственные нейронные сети) и оптимизация (глобальная оптимизация и эволюционные вычисления). Важные достижения произошли в решении проблем классификации (теория статистического обучения и байесовские методы).

Разработанные методы позволяют «интеллектуально» обрабатывать данные. В областях, где требуется обрабатывать очень большие наборы данных (в первую очередь, экономика, управление отношениями с клиентами и финансовые услуги, а также медицина, электроника, военная промышленность и т.д.), были введены термины «интеллектуальный анализ данных» и «поиск знаний» для обозначения группы методов, позволяющих проводить такой анализ, выявлять основные закономерности в данных и строить прогнозы на их основании. С другой стороны, многие представители сообщества ИИ перешли в, может быть, менее амбициозную, но более практичную область «машинного обучения».

В настоящее время собирается огромное количество данных о многих природных и социальных процессах и системах, включая речные системы. В то же время разработано множество методов интеллектуальной обработки данных и разработки оптимизированных решений. Во многих случаях эти методы имеют прочную теоретическую основу. Наличие теории, методик и данных

дает возможность говорить о новой парадигме моделирования – моделировании на основании данных или data-driven modelling (DDM).

Модель системы на основании данных о ней – это модель, соединяющая переменные состояния системы (входные, внутренние и выходные переменные) и построенная на основе собранных данных об этих состояниях. Это не обязательно означает, что отсутствуют сведения о деталях процессов, лежащих в основе поведения системы, однако часто модели на основе данных строятся, когда такие знания отсутствуют или недостаточны.

С другой стороны, модели, основанные на описании процессов (также называемые концептуальными, или физически обоснованными, или основанными на знаниях, или имитационными моделями) основаны на понимании основных процессов в системе, например физических, социальных, экономических или политических.

Зачастую, специалистов не интересует, какой тип модели используется - они хотят иметь эффективный аппарат, улучшающий принятие решений. Во многих случаях имеется понимание моделируемых процессов, но не очень подробное, поэтому точные концептуальные модели не могут быть построены. В этом случае неизбежно строится несколько моделей разной сложности и уровня представления. Если такая составная модель сочетает в себе подходы, основанные на процессах и данных, ее можно назвать «гибридной моделью», в которой различные компоненты дополняют друг друга. Такой подход в настоящее время является наиболее популярным в области методов машинного обучения.

Данное пособие основано на использовании и предназначено для изучения наиболее широко распространенной методики машинного обучения - искусственной нейронной сети (ИНС) для построения гидрологической модели речной системы с преимущественно дождевым питанием.

Машинное обучение

Метод машинного обучения (ML) — это алгоритм, который оценивает неизвестное до сих пор взаимоотношение (или зависимость) между входами системы и ее выходами на основе имеющихся данных (рис. 1). Под данными мы понимаем выборки, которые представляют собой ряды наблюдений за входами и выходами системы. Когда такая зависимость обнаружена, ее можно использовать для прогнозирования (или анализа) будущих выходных данных системы на основе других известных входных значений.

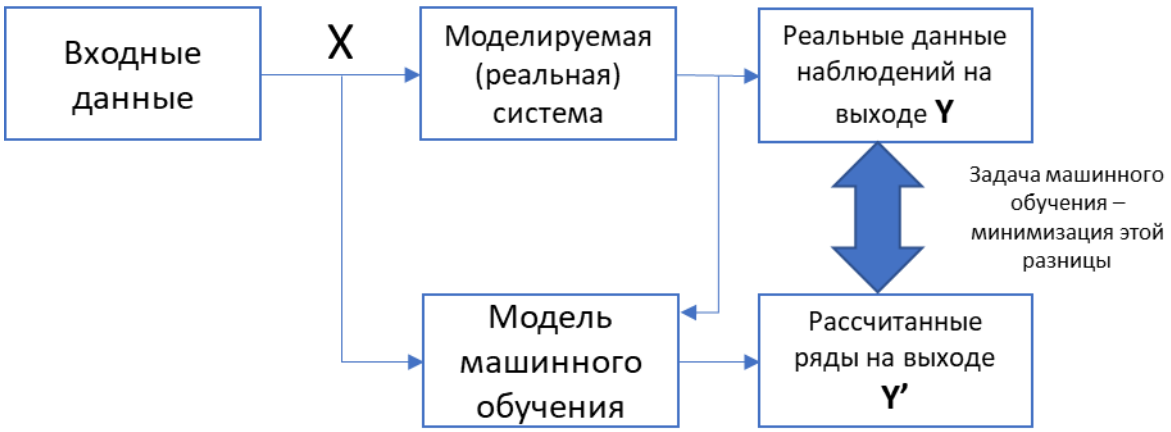


Рисунок 1 – Схема модели машинного обучения

Обозначим набор входных данных как *X* и набор выходных данных как *Y*. Предполагается, что *i*-й элемент *X* (обозначенный как x_i) имеет соответствующий выходной элемент y_i. Пара $\{x_i, y_i\}$ называется тогда *экземпляром*. Элементы *X* также можно назвать входными экземплярами.

Экземпляры могут быть охарактеризованы несколькими параметрами или атрибутами, которые представлены в виде точек в пространстве параметров — это пространство часто называют *пространством состояний*.

На рис. 1 показан процесс машинного обучения. Модель машинного обучения на основе экземпляров данных пытается идентифицировать целевую функцию $Y = f(X)$, описывающую, как ведет себя реальная система. Обучение (или тренинг) здесь — это процесс минимизации разницы между наблюдаемыми данными и выходными данными модели (ошибка модели). Данные, используемые для обучения, называются набором данных для обучения.

После обучения, модель ML, снабженная новыми входными экземплярами, может выполнять прогнозирование, то есть для нового входа x_i генерировать выход y_i, значение которого, как предполагается, близко к тому, которое генерирует реальная система.

Вообще говоря, проблема машинного обучения содержит три основных компонента: архитектура модели (или класс функции) *M*, которая имеет набор изменяемых параметров *P* и измеренные данные *D*. Примером *M* может быть полином заданного порядка, где *P* - коэффициенты. Цель состоит в том, чтобы определить архитектуру модели *M* и значения параметров *P*, которые приводят к наилучшему возможному согласованию между предсказаниями модели и данными *D*.

Одним из простейших примеров модели машинного обучения является модель линейной регрессии (*M* - линейная функция); в нем на основе наблюдений строится (обучается) формула (алгоритм), приблизительно воспроизводящая реальность (рис. 2). Методы машинного обучения позволяют строить модели, которые намного точнее линейных.

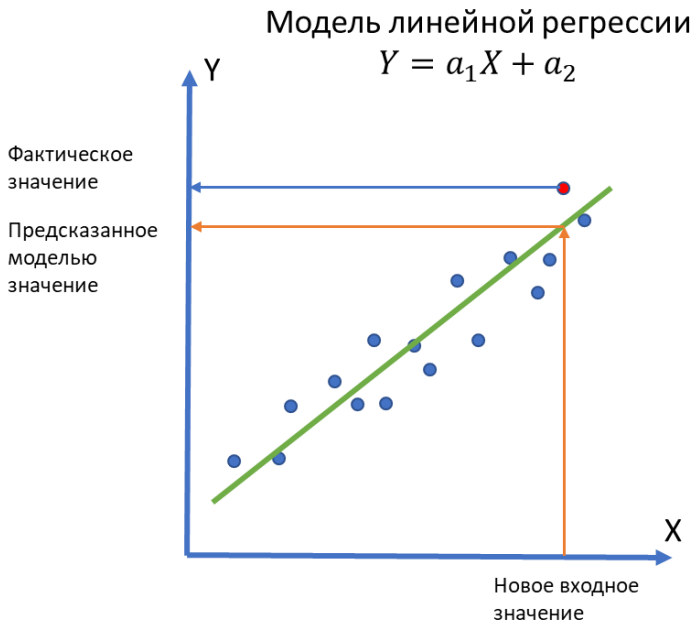


Рисунок 2 – Модель машинного обучения на основе модели линейной регрессии

Однако, входные и выходные переменные могут быть разных типов, наиболее распространенными из которых являются числовые (действительные) а также категориальные (символические). Числовые выходные данные соответствуют задаче оценки регрессии (или

непрерывной функции), в то время как категориальные выходные данные представляют собой класс проблем, известный как распознавание образов и классификация.

Обучение, кросс-валидация и верификация

Минимизация разницы между наблюдаемыми и прогнозируемыми результатами на самом деле имеет двойное значение. Следует различать

• минимизацию ошибки при обучении модели;

• минимизацию погрешности при работе модели.

Модель может быть обучена и хорошо работать (с низкой ошибкой) на наборе обучающих данных, но, когда в модель загружаются новые экземпляры, которые ранее не были использованы для обучения, ошибка может быть высокой. Это означает, что необходимо иметь отдельный набор данных (называемый набором для тестирования или кросс-валидации), который не содержит экземпляров из обучающего набора и используется для проверки ошибки модели. Обучающий набор иногда называют выборочным набором данных. Данные, не содержащие экземпляров из обучающего набора, также называются вневыборочным набором данных.

Этот эффект «слишком долгого обучения» называется «переобучением» - модель пытается описать все точки данных выборки (возможно, содержащими шум) слишком тщательно, не воспроизводя на самом деле об основной тенденции, наблюдаемой в данных. Под *робастностью* алгоритма машинного обучения мы понимаем точность, с которой модель способна воспроизводить за пределами обучающих данных. На рис. 3 показано использование кросс-валидации для выявления момента переобучения.

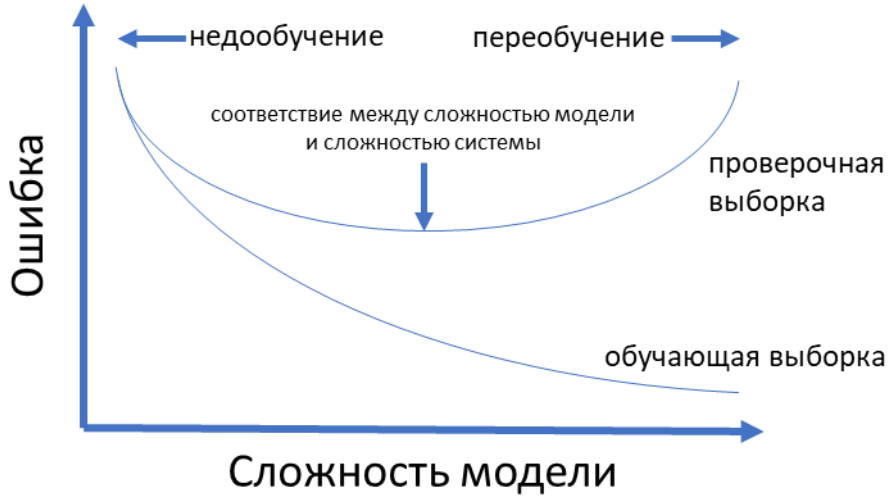


Рисунок 3 – Использование кросс-валидации в процессе обучения

Чтобы контролировать точность модели на данных вне выборки, необходимо иметь еще один набор данных (кросс-валидация) наряду с обучающим. На самом деле обучение должно прекратиться, когда ошибка в наборе данных при кросс-валидации начинает увеличиваться — это показатель начала переобучения.

Когда модель машинного обучения вводится в эксплуатацию, она сталкивается с экземплярами ввода, которые никогда раньше не «видела». Чтобы проверить ее работоспособность

(измеренную по величине ошибки) перед вводом в эксплуатацию, ее необходимо проверить. Для этой цели используется еще один набор данных, называемый верификационным набором данных (не путать с набором тестирования). Он позволяет проверить, как модель будет работать, если в нее будут загружены новые данные.

На рис. 4 показано использование наборов данных для обучения, кросс-валидации и верификации в машинном обучении.

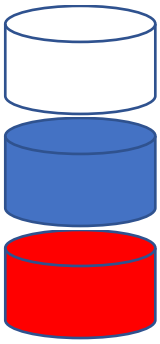


Рисунок 4 – Обучение, кросс-валидация и верификация в машинном обучении

Искусственные нейронные сети

Искусственные нейронные сети (ИНС), создание которых первоначально связно с исследованием искусственного интеллекта, в настоящее время стали одной из основных технологий в машинном обучении и основной технологией моделирования на основе данных. Для кластеризации, классификации и прогнозирования используются различные типы сетей. ИНС в общих чертах имитируют функционирование нейронов в мозгу человека, и оказалось, что можно объединить эти нейроны таким образом, чтобы сеть воспроизводила любую многомерную многозначную функцию при наличии достаточного количества точек и значений этой функции.

Линейная регрессия как простейшая ИНС

Возможно, самый простой алгоритм машинного обучения — это уравнение линейной регрессии, которое можно «обучить», чтобы воспроизвести соотношение ввода-вывода (то есть найти коэффициенты линейной функции). На рис. 5 показана «сеть», состоящая из одного узла с линейным расчетным элементом (РЭ).

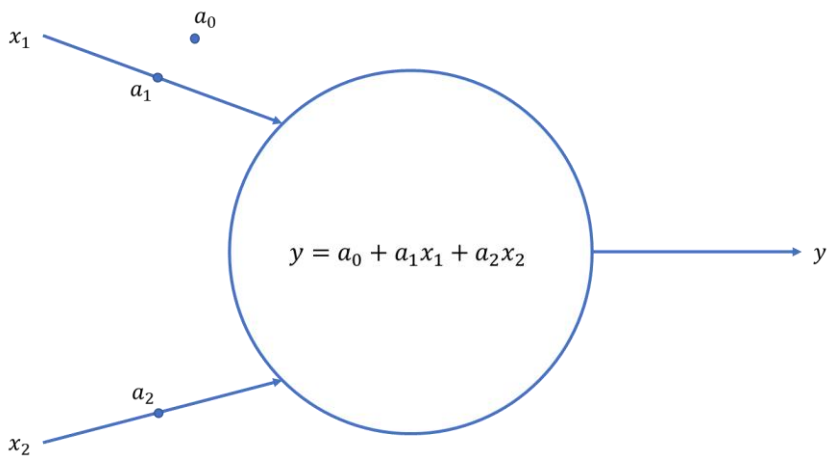


Рисунок 5 – Расчетный элемент (РЭ) на основании линейной регрессии (2 входа)

В одномерном случае (один вход *x*), при наличии вектора входа длиной *T* экземпляров

$$\{x^t,\ y^t\},\ \ t = 1,\dots T$$

коэффициенты уравнения

$$y = f(x) = a_1 X + a_2$$

могут быть найдены и затем для новых данных длиной *V*

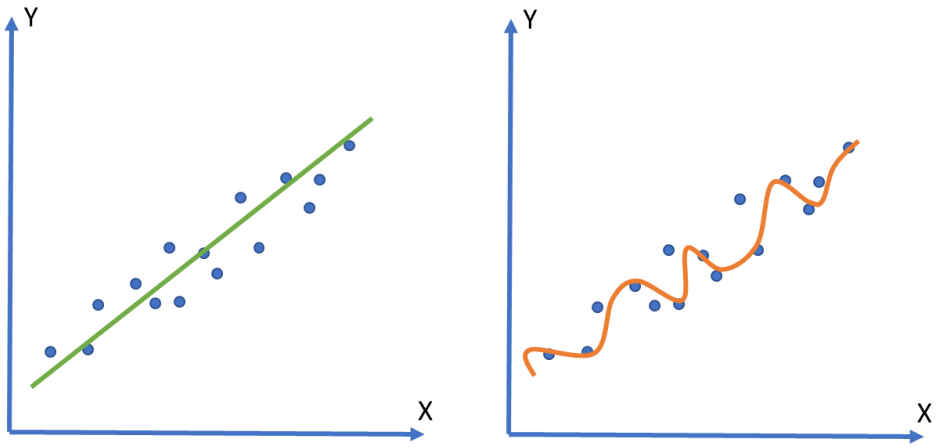
$$\{x^v\},\ \ v = 1,\dots V$$

модель может воспроизвести данные на выходе с определенной точностью

$$\{y^v\},\ \ v = 1,\dots V.$$

Добавление сложности: слои и связи

Представленная выше регрессионная модель не слишком точна - она просто представляет линейную зависимость, хотя большинство процессов, с которыми сталкиваются специалисты в процессе работы, очень нелинейны (рис. 6). Чтобы сделать модель нелинейной, необходимо ввести нелинейные компоненты, чтобы воспроизвести (аппроксимировать) значительно более сложные соотношения с несколькими входами и выходами.



Линейная регрессия
$Y = a_1X + a_2$

Искусственная нейронная сеть
$Y = f(X, a_1, \dots, a_n)$

Рисунок 6 – Схематическое сравнение моделей линейной регрессии и ИНС

Для N_{out} выходных функций, для каждой из которых определены N_{inp} независимых входных величин, длиной T экземпляров

$$\{ x_1^{(t)}, x_2^{(t)}, \dots, x_{N_{inp}}^{(t)}, f_1^{(t)}, f_2^{(t)}, \dots, f_{N_{out}}^{(t)} \}, \; t = 1, \dots, T$$

то на основании этих данных ИНС может быть *обучена*, тогда при подаче на вход другого набора векторов V

$$\{ x_1^{(v)}, x_2^{(v)}, \dots, x_{N_{inp}}^{(v)} \}, \quad v = 1, \dots, V$$

она способна воспроизвести значения соответствующих функций с определенной точностью

$$\{ f_1^{(v)}, f_2^{(v)}, \dots, f_{N_{out}}^{(v)} \}, \quad v = 1, \dots, V$$

Одним из широко используемых типов ИНС является многослойный перцептрон (multi-layer perceptron, MLP), схема которого изображена на рис. 7. Он состоит из ряда взаимосвязанных узлов (называемых нейронами, также, как и РЭ), организованных в уровни трех типов: входной, скрытый (их может быть несколько) и выходной. Линии представляют собой взвешенные связи между РЭ.

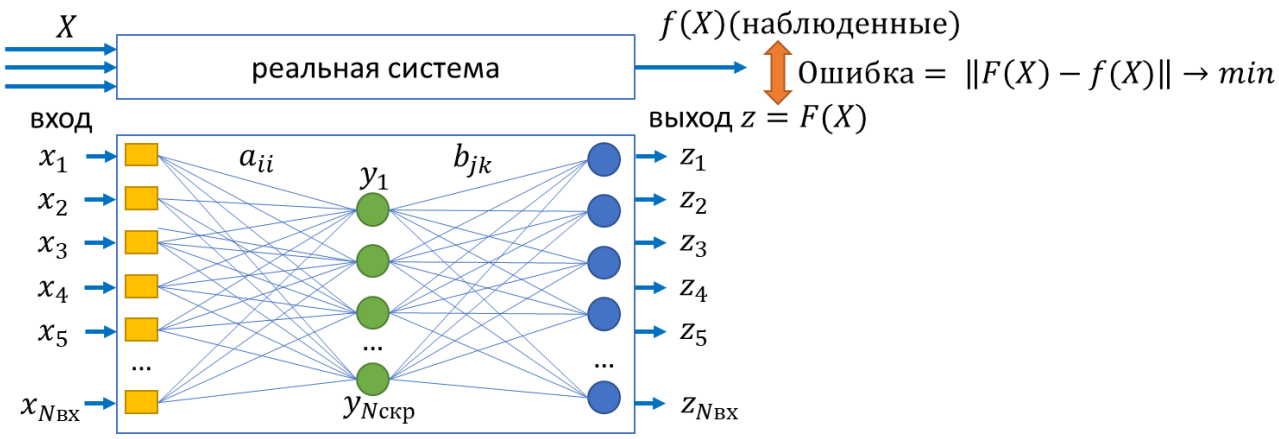


Рисунок 7 – Многослойный перцептрон с одним скрытым слоем

Входной слой на рисунке 7 не выполняет никаких операций с входным сигналом, а просто отправляет x_i в нейроны в скрытом слое. Элемент обработки просто умножает входные данные на набор весов и линейно или нелинейно преобразует результат в выходное значение. Результат может быть входом для других РЭ в любых слоях, кроме входного. Адаптируя свои веса, нейронная сеть работает над оптимальным решением на основе измерения ее производительности.

Входы в ИНС описываются как:

$$x_i, \quad i = 1, \dots, N_{inp}$$

Узел в скрытом слое получает сигналы (значения) от узлов входного слоя и преобразует их в сигналы, которые отправляются на все выходные узлы, которые, в свою очередь, преобразуют их в выходы. Выход *j*-го узла скрытого слоя:

$$y_j = g\,(\,a_{0j} + \sum_{i=1}^{N_{inp}} a_{ij}\,x_i\,)\ ,\quad j = 1,\ldots,N_{hid}$$

Выходные узлы получают сигналы из скрытого слоя и очень похожи. У них также есть две составляющие: первая - линейная, а вторая - обычно нелинейная (однако для задач регрессии вторая составляющая сводится к линейной).

Выходы *k*-ых выходных узлов:

$$z_k = g\,(\,b_{0k} + \sum_{j=1}^{N_{hid}} b_{jk}\,y_j\,)\ ,\quad k = 1,\ldots,N_{out}$$

Функция активации *g* в скрытом слое обычно является нелинейной, ограниченной и кусочно-дифференцируемой функцией (обычно сигмоидальной формы). Широко используемая сигмовидная (логистическая) функция ограничена между 0 и +1 (рис. 8):

$$g\,(u) = \frac{1}{1 + e^{-\alpha\,u}}$$

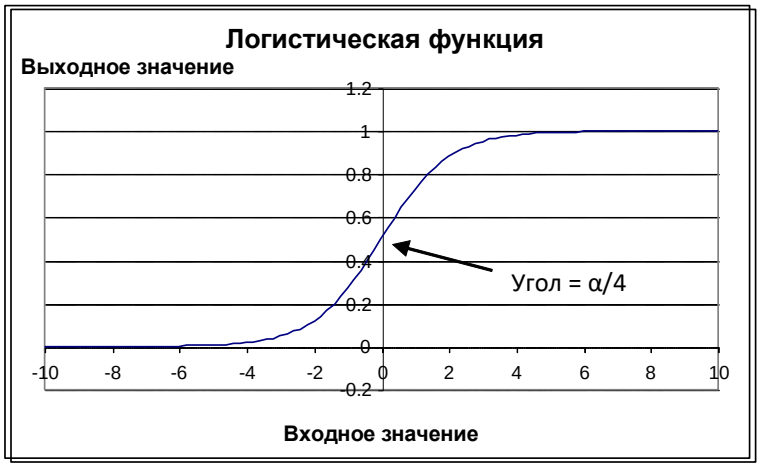


Рисунок 8 – Логистическая функция (здесь для α=1) – нелинейная функция активации, зачастую используемая в ИНС

Другой часто используемой функцией является гиперболический тангенс (рис. 9), ограниченный от –1 до +1:

$$g(u) = \tanh(\alpha\,u) = \frac{e^{\alpha u} - e^{-\alpha u}}{e^{\alpha u} + e^{-\alpha u}}$$

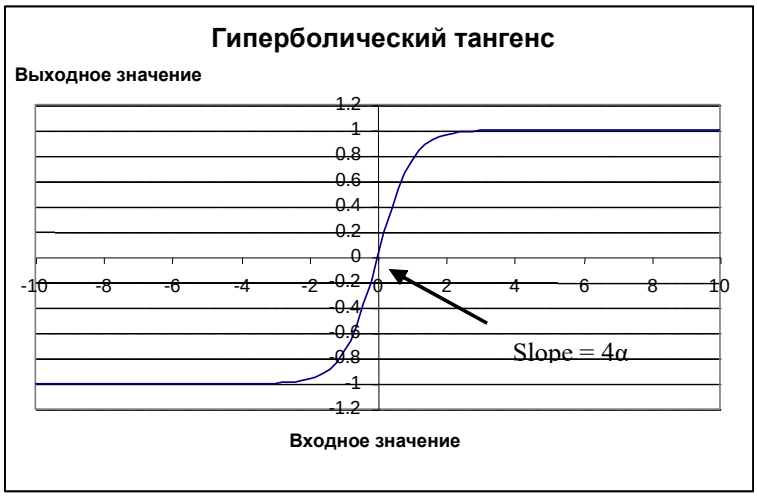


Рисунок 9 – Гиперболический тангенс (здесь для α=1) – еще один пример функции активации

Обратите внимание, что для задач классификации (когда значения находятся в ограниченном диапазоне) обе функции активации в скрытом и выходном слое являются сигмоидальными. Однако для задач регрессии, чтобы не слишком сильно ограничивать выходные значения, функция активации в выходном слое часто (но не всегда) выбирается линейной с единичным наклоном, так что выходные узлы становятся просто узлами суммирования. Нелинейность ИНС обеспечивается сигмоидальными функциями в узлах скрытого слоя.

Обучение ИНС МСП

В задаче обучения ИНС существуют $(N_{inp}+1)N_{hid}+(N_{hid}+1)N_{out}$ параметров (весов ***a*** и ***b***), оптимальные значения которых достигаются путем *калибровки*, или *обучения*, таким образом, чтобы минимизировать целевую функцию ошибок.

Предположим, что для обучения используются векторы длиной *T* экземпляров

$$\{\ x_1^{(t)}, x_2^{(t)}, \ldots, x_{N_{inp}}^{(t)},\ f_1^{(t)}, f_2^{(t)}, \ldots, f_{N_{out}}^{(t)}\ \},\ t = 1, \ldots, T$$

Это означает, что когда вектор $x^{(t)}$ длиной N_{inp} (также называемый *t*-м входом) был введен в систему в качестве входных данных, были сгенерированы фактические (измеренные) значения *k*-й выходной переменной $f_k^{(t)}$, составляющие вектор $f^{(t)}$ (также называемый *t*-м фактическим выводом). Для того же входа ИНС сгенерировала *k*-ое значение выходной переменной, обозначенное как $z_k^{(t)}$, которые вместе составляют вектор $z^{(t)}$ (также называемый *t*-м выходом).

Следует перечислить используемые переменные и индексы:

- *t* для экземпляров (ввода и вывода) - обозначается в скобках как надстрочный индекс, начиная с 1 до *T*;
- *i* для входных переменных x, от 1 до N_{inp};
- *j* для скрытых узлов (на выходе *y*), от 1 до N_{hid};
- *k* для выходных узлов и выходных переменных *z* и значений меры *f*, от 1 до N_{out};
- a_{ij} для весов между входным и скрытым слоями;
- b_{jk} для весов между скрытым и выходным слоями;
- w_s в качестве общей ссылки на веса *a* и *b*, которые необходимо найти.

Результат расчета ИНС по одному входному экземпляру отличается от фактического значения на некоторую величину ошибки - эта разница возведена в квадрат. Для выхода k это

$$E_k^{(t)} = (f_k^{(t)} - z_k^{(t)})^2$$

Оценка ошибки для всех выходов может быть представлена по-разному, но обычно ее можно выразить с помощью метода наименьших квадратов:

$$E^{(t)} = \frac{1}{2}\sum_k (f_k^{(t)} - z_k^{(t)})^2$$

Итоговая величина ошибки — это сумма ошибок для всех выходных узлов для всех экземпляров T:

$$E_{tot} = \frac{1}{2}\sum_t \sum_k (f_k^{(t)} - z_k^{(t)})^2$$

Функция ошибки может быть представлена в следующем виде:

$$\begin{aligned} E_{tot} &= \frac{1}{2}\sum_t \sum_k (f_k^{(t)} - z_k^{(t)})^2 \\ &= \frac{1}{2}\sum_t \sum_k [f_k^{(t)} - g^{out}(b_{0k} + \sum_j b_{jk} y_j)]^2 \\ &= \frac{1}{2}\sum_t \sum_k [f_k^{(t)} - g^{out}(b_{0k} + \sum_j b_{jk}\ g^{hid}(a_{0j} + \sum_i a_{ij} x_i^{(t)}))]^2 \end{aligned}$$

Таким образом, задача оптимизации сводится к нахождению таких значений весов a и b, которые приводят значение E_{tot} к минимуму.

Поскольку функция *E* известна аналитически и является дифференцируемой, можно использовать методы на основе градиентов, такие как градиентный спуск или более эффективный метод сопряженных градиентов. Процесс решения этой проблемы оптимизации называется обратным распространением ошибки – поскольку он включает в себя вычисление ошибок ИНС и распространение ошибок обратно по сети для соответствующего обновления весов.

Практические аспекты обучения

С использованием ИНС связаны три аспекта: количество данных и их нормализация, количество скрытых узлов ИНС, выбор алгоритмов и функций активации, а также целевые функции.

Подготовка данных

ИНС является одним из методов машинного обучения, и здесь также актуальны все рассмотренные вопросы подготовки данных, выбора обучающей, кросс-валидационной и верификационной выборок, проверки робастности и т.д. Набор обучающих данных должен охватывать весь диапазон возможных изменений предсказываемой величины. Очевидно, желательно, чтобы обучающие данные включали максимальное и минимальное значения за весь период наблюдений. Единственные общие правила - использовать много репрезентативных данных. На этом этапе принципиально важно понимание решаемой проблемы. Все эти аспекты более подробно обсуждались в предыдущих разделах, посвященных машинному обучению и подготовке данных.

Нормализация данных для предотвращения паралича сети

Всегда полезно нормализовать данные до интервала [0;1] - в этом случае диапазоны весов не будут сильно отличаться, и это желательное условие для алгоритмов оптимизации.

Другой тип масштабирования может потребоваться из-за характеристик логистической функции, которая очень быстро достигает насыщения:

$$g(1,0) = 0,762$$

$$g(2,0) = 0,964$$

$$g(3,0) = 0,995$$

$$g(4,0) = 0,999$$

Это означает, что диапазон входных переменных должен быть достаточно узким, часто между –3 и +3. Если этого не сделать, может наблюдаться так называемый паралич нейронной сети, когда входы в логистические функции настолько высоки, что их выходы очень близки к –1 или +1, а процесс обучения блокируется.

Еще одно полезное преобразование — это преобразование выходных значений в диапазон [0.1;0.9] (в случае использования логистических выходных узлов). Причина в том, что узлы вывода сигмовидной формы не могут производить значения за пределами диапазона [0;1], и, если мы попытаемся обучить сеть, чтобы соответствовать целевым значениям за пределами этого диапазона, градиентный спуск заставит веса неограниченно расти. С другой стороны, значения 0.1 и 0.9 достижимы при использовании сигмовидного узла с конечными весами. Большинство инструментов автоматически выполняют указанные типы масштабирования.

Количество скрытых узлов

Число скрытых узлов определяет способность сети к воспроизведению ряда наблюдений – чем больше число, тем мощнее сеть. Однако, если это число слишком велико, робастность может ухудшиться. Это связано с переобучением на обучающей выборке, которое можно решить с помощью перекрестной проверки. Наилучшее количество скрытых узлов может быть получено методом проб и ошибок. Одно из «правил» - количество скрытых узлов должно быть примерно $N_{hid} \approx \sqrt{N_{inp}\ N_{out}}$. Многие программные библиотеки для машинного обучения позволяют автоматизировать «метод проб и ошибок».

Выбор функций активации

Как упоминалось ранее, в задачах регрессии наилучшие результаты достигаются с нелинейными сигмоидальными функциями в скрытом слое и линейными функциями в выходном слое. Многие библиотеки для машинного обучения позволяют использовать очень широкий выбор функций.

Выбор алгоритма обучения

В современных библиотеках машинного обучения, таких как scikit-learn, есть возможность выбрать один или несколько алгоритмов минимизации ошибки, которых насчитывается более 15, например алгоритм Леверберга-Марквардта или алгоритмы сопряженных градиентов.

Критерии эффективности ИНС

Типичным критерием производительности ИНС является ее наименьшая квадратичная ошибка (LSE), а ее квадратичный характер явно используется в правилах обновления обратного распространения ошибок для весов. Однако могут использоваться и другие критерии, но для этого может потребоваться изменение алгоритмов оптимизации (обучения) сети.

Выбор входных переменных

Выбор релевантных задаче переменных важен для любого моделирования, включая моделирование на основе ИНС. Если входные переменные выбраны неправильно, они не будут иметь большой корреляции с выходными рядами, и такая модель не будет иметь никакого смысла. Их выбор может быть основан на

- экспертном заключении;
- корреляционном анализе (попытке найти набор входных данных, которые имели бы высокую корреляцию с выходными данными);
- так называемом выборы на основе модели, когда для каждой возможной (или разумной) комбинации входных данных строится модель, оцениваются ее характеристики (на обучающем наборе или, предпочтительно, на множестве перекрестной проверки), и выбирается лучшая модель.
- сочетании трех предыдущих.

Например, при построении модели дождевого стока в качестве входных переменных входными переменными, также называемыми *предикторами*, могут быть осадки и сток (а иногда и температура воздуха), измеренные в предшествующие моменты времени (другими словами, с разным запаздыванием). Результатом обычно является сток заданной заблаговременностью. Выбор этих запаздываний для осадков, стока и температуры приведет к построению оптимального набора (релевантных) входных данных, которые потенциально могут гарантировать построение наилучшей возможной модели.

Часть II. Интерактивное учебное пособие по машинному обучению для гидрологических задач

Введение

Цель настоящего интерактивного учебного пособия - представить основные элементы построения модели, основанной на данных наблюдений, в том числе:

- Подготовка данных: подготовка гидрологических данных для обучения и тестирования моделей
- Выбор соответствующих входных переменных моделей
- Построение простой модели, основанной на данных - модели линейной регрессии
- Тестирование модели на независимых данных (набор проверочных данных), имитирующих краткосрочное гидрологическое прогнозирование.

Требования к обучающемуся

Для прохождения обучения требуются следующие навыки:

- Базовое образование в области наук о Земле (гидрология, метеорология, экология и т.п.).
- Базовые навыки программирования в `python` с использованием библиотек `numpy`, `pandas`, `matplotlib`.

Технические требования

Все основные материалы представлены на Google Drive Школы для молодых ученых «Новое поколение моделей, методов и технологий для противодействия современным угрозам водной безопасности».

Обучающие материалы представлены в документе Jupyter Notebook, размещенном в облачном хранилище общего доступа Google Drive, с программным кодом, исполняемым на сервере облачных вычислений Google Colab. Благодаря этому, никаких дополнительных программ на компьютере пользователя устанавливать не требуется (при условии наличия интернет-браузера). Все используемые библиотеки также уже установлены на вычислительном сервере.

Подключение данных

Для использования исходных данных и расчетов, пользователь должен быть авторизован в учетной записи Google.

Перед началом работы нажмите на выпадающее меню рядом с названием папки IWP HYDRO DDM и нажмите "Добавить ярлык на диск".

Далее выполните следующий кусок кода. Если вы авторизовались в учетной записи Google, вам будет выдан код для подключения по ссылке. Выполните следующий кусок кода и следуйте указаниям. Перейдите по появившейся ссылке, скопируйте авторизационный код и вставьте его в поле ниже.

```
from google.colab import drive
drive.mount('/content/drive')
```

Обратите внимание

В случае, если пользователь пожелает использовать свои входные данные или скачать код и использовать его на локальном компьютере, он делает это на свой страх и риск. Приведенный ниже код и данные, используемые для демонстрации, были проверены только в Google Colab и авторы не несут ответственность за исполнение программного кода в других средах.

1. Первая модель машинного обучения: множественная линейная регрессия

В этом ipynb ноутбуке вы познакомитесь с тем, как построить простую прогнозную модель множественной линейной регрессии.

Далее необходимо загрузить несколько библиотек и модулей. Мы будем использовать библиотеки `numpy` для алгебраических вычислений, `pandas` для манипулирования табличными данными, `matplotlib` для графики. Для создания моделей машинного обучения будет использоваться библиотека `scikit-learn`.

```
# import sys
# sys.path.append('/content/drive/My Drive/IWP HYDRO DDM/')
from time import time
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.dates import DateFormatter
from sklearn.metrics import r2_score, mean_squared_error, make_scorer
from sklearn.linear_model import LinearRegression
from sklearn.multioutput import MultiOutputRegressor
from sklearn.preprocessing import StandardScaler
import warnings
```

1.1 Объект исследования

В качестве объекта исследования используются среднесуточные расходы воды р. Уссури на гидрометрическом посту Кировский в Приморском крае. Уссури - приток Амура; площадь водосбора до устья - 193 000 км^2, в створе поста Кировский - 24400 км^2.

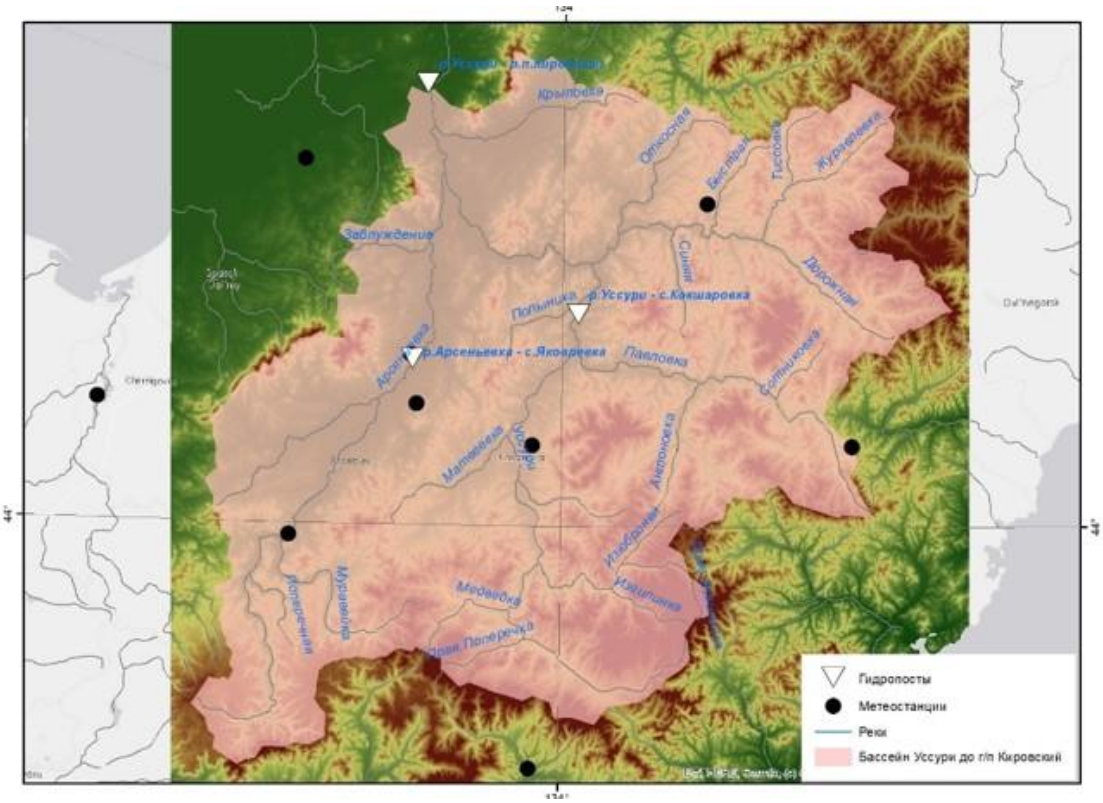


Схема водосбора р. Уссури до г/п Кировский

Климат бассейна определяется его географическим положением на стыке Евразийского материка с Тихим океаном, в области распространения восточно-азиатского внетропического муссона. Уссури относится к рекам с дальневосточным (паводковым) типом водного режима с преимущественно дождевым питанием. Основная часть стока проходит в летне-осенний период (95% от годового объема стока), половодье выражено слабо. Зимой наблюдается устойчивая межень (всего 2–5% годового стока). Среднемноголетний расход воды в замыкающем створе р.Уссури – р.п.Кировский равен 220 м^3/с.

Исходные данные

Метеорологическая информация

Набор метеорологической информации для бассейна р.Уссури – р.п. Кировский представляет собой ряды данных о среднесуточных температурах воздуха (^{o}C) и суммах осадков (мм), измеряемых на 3 метеостанциях за период 1978 – 1986 гг. Данные были взяты из базы ВНИИГМИ-МЦД.

Таблица 1 - Метеостанции

Индекс ВМО	**Название**
31939	Чугуевка
31942	Самарка
31981	Анучино

Гидрологическая информация

Ряды данных о расходах воды, взятые с гидрологических постов для бассейна р.Уссури за период 1978 – 2015 г. Данные за период до 2000 г. взяты из гидрологических ежегодников, за период после 2000 г. частично из гидрологических ежегодников, а также из базы данных АИС ГМВО. На территории бассейна р.Уссури-р.п.Кировский расположено 15 гидропостов, на которых имеются данные о расходах и осадках за период 1978 – 2015 гг. Причем данные о температурах и дефицитах влажности воздуха на гидропостах отсутствуют. Информация об осадках с гидропостов была предоставлена отделом гидропрогнозов Приморского управления.

Задача

Мы настроим несколько моделей машинного обучения для прогнозирования среднесуточного расхода воды р. Уссури заблаговременностью от 1 до 7 дней в летне-осенний паводочный период (с мая по октябрь).

Далее загружаем файл с исходными данными по осадкам и стоку и исследуем его.

```
# путь до исходного файла с данными
df = pd.read_excel('/content/drive/My Drive/IWP HYDRO DDM/data/ussuri_data
.xlsx', index_col='date')
df
```

В файле 4 колонки: значения среднесуточных расходов воды и суточных сумм осадков по трем метеостанциям. Нанесем их на хронологический график.

Как мы видим, выдающиеся осадки на всех станциях приводят к выдающимся паводкам.

```
plt.figure()
ax = df.plot(secondary_y=[31939,  31942,  31981], mark_right=False,
figsize=(20,8))
ax.set_xlabel('Дата')
ax.set_ylabel('Расход воды, м$^3$/с')
ax.set_ylim(0, 3000)
ax.right_ax.set_ylim(0, 200)
ax.right_ax.set_ylabel('Осадки, мм')
ax.right_ax.invert_yaxis()
ax.legend(loc='lower right')
ax.right_ax.legend(loc='upper right')
plt.plot()
```

1.2 Подготовка данных для моделирования

Подготовка данных будет состоять из следующих этапов:

- выбор входных и выходных переменных путем анализа взаимозависимостей между переменными и сдвигами во времени ("лагами");
- преобразование данных, если необходимо;
- разделение данных на обучащую (train) и проверочную (test) выборки;
- сохранение преобразованных данных в виде файлов xlsx.

1.2.1 Сдвиг данных во времени

Так как данные есть только для одного замыкающего створа, мы можем смоделировать только время реакции водосбора на выпадение осадков, т.е. время бассейнового добегания. В общем виде такая модель представляет собой зависимость:

$$Q_{t+1}=f(Q_{t,t-1,\ldots,t-n};R_{t,t-1,\ldots,t-n}),$$

где *Q* - расход воды, *R* - осадки, *t* - текущий момент времени, *i = 1, ..., n* - шаги во времени в сутках.

```
# исходная длина всех данных
init_length = len(df)
# величина сдвига назад по времени расходов воды и осадков, сутки
shift = 10
# заблаговременность прогноза, сутки
steps_predict = 7
# новый пустой датафрейм для записи данных со сдвигом
df_ivs = pd.DataFrame()

# сдвиг предикторов назад
for st in df.columns:
    for i in range(shift + 1):
        colname = str(st) if i == 0 else str(st) + 't-' + str(i)
        series = df[st][shift-i:init_length-1-steps_predict-
i].reset_index(drop=True)
        df_ivs[colname] = series

# сдвиг предиката вперед
ycolname = 'Qt+' + str(steps_predict)
series = df['Q'][steps_predict+shift:].reset_index(drop=True)
df_ivs[ycolname] = series

# сохраняем полученный датафрейм
df_ivs.to_excel('/content/drive/My Drive/IWP HYDRO DDM/data/ussuri_shift_d
ata_test.xlsx')

# размер полученного массива
print('Data rows/cols:', df_ivs.shape)

# название колонок с предикторами
xcols = [col for col in df_ivs.columns if 'Qt+' not in col]
print('Features: ', len(xcols), xcols)

# название колонки (колонок) с предикатом
ycols = [col for col in df_ivs.columns if 'Qt+' in col]
print('Target: ', len(ycols), ycols)

# смотрим на полученный датафрейм
df_ivs.head(10)
```

Наша задача - построить модель для прогнозирования расходов воды заблаговременностью от 1 до 7 суток. Для этого, сначала необходимо определить время добегания осадков до замыкающего створа, осредненное для всего бассейна.

1.2.2 Определение времени добегания

Определение времени добегания производится с помощью анализа автокорреляционной функции.

```
# Вычисляем корреляционную функцию между всеми данными
corr = round(df_ivs.corr(), 2)
# В виде графика - для заблаговременности 1 сутки
```

```
corr[ycols][:-
7].plot(kind='bar', figsize=(30, 10), ylim=(0, 1), grid=True)

# В виде таблицы
corr.loc[ycols, xcols].style.background_gradient(cmap='coolwarm').format("
{:.3}")
```

Судя по корреляционной функции, наиболее сильная автокорреляция расходов воды наблюдается на шаге *t-1*. Время добегания осадков по водосбору (максимальная корреляция с осадками) составляет 7 суток (*t-7*).

1.2.3 Разделение данных на обучащую (train) и проверочную (test) выборки

Длина всей полученной после сдвигов выборки составляет 2904 шага (суток). Для обучающей выборки мы возьмем первые 2192 значения, для проверочной - оставшиеся 712.

```
# определяем размер обучающей выборки
train_size = 2192

# выбираем предикторы из данных со сдвигом
X = df_ivs.loc[:, xcols]

# выбираем предикат
y = df_ivs.loc[:, ycols]

X_train = X[:train_size].reset_index(drop=True)
X_test = X[train_size:].reset_index(drop=True)

y_train = y[:train_size].reset_index(drop=True)
y_test = y[train_size:].reset_index(drop=True)

print('Train:', X_train.shape, y_train.shape)
print('Test: ', X_test.shape, y_test.shape)
```

```
Train: (2192, 44) (2192, 1)
Test:  (712, 44) (712, 1)
```

1.2.4 Проверочные целевые функции

В качестве целевых функций для проверки качества моделирования используем

- среднеквадратическую ошибку:

$$RMSE = \sqrt{\frac{1}{n}(Q_i, p - Q_i, o)^2}$$

где обозначения *p* и *o* означают прогнозный (predicted) и наблюденный (observed) расходы.

- критерий Нэша-Сатклиффа:

$$NSE = 1 - \frac{\sum_{i=1}^{n}(Q_i, m - Q_i, o)^2}{\sum_{i=1}^{n}\left(\overline{Qo} - Q_i, o\right)^2}$$

Для расчета среднеквадратической ошибки используем готовую функцию `mean_squared_error` из библиотеки `sklearn`, а для расчета *NSE* используем готовую функцию `r2_score`, поскольку формально *NSE* аналогичен коэффициенту детерминации R^2.

1.3 Построение модели множественной линейной регрессии

Для построения модели мы используем функцию `LinearRegression()` библиотеки `sklearn`. Чтобы сразу прогнозировать на 7 шагов (суток) вперед, дополнительно используем оператор `MultiOutputRegressor()`.

1.3.1 Построение, обучение и проверка модели

```
# инициализируем модель
lm = LinearRegression()
# обучаем модель
lm.fit(X_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
normalize=False)
```

```
# прогнозные ряды для обучающего массива
train_pred = lm.predict(X_train)
# прогнозные ряды для проверочного массива
test_pred = lm.predict(X_test)
# ряды наблюдений (obs) и прогнозов (pred) для данной (i, y) заблаговремен
ности

# рисуем два гидрографа для обучения и проверки
fig, ax = plt.subplots(nrows=3, ncols=2, figsize=(10,10), dpi=100)
ax[0,0].plot(range(len(y_train)), y_train, label='Наблюдения', linewidth=3
.0)
ax[0,0].plot(range(len(y_train)), train_pred, label='Прогноз', c='orangere
d')
ax[0,0].set(xlabel='Сутки', ylabel='Расход воды (м$^3$/с)', title='Обучени
е')
ax[0,0].grid()
ax[0,0].set_ylim(0, 2000)
ax[0,1].plot(range(len(y_test)), y_test, label='Наблюдения', linewidth=3.0
)
ax[0,1].plot(range(len(y_test)), test_pred, label='Прогноз', color='orange
')
ax[0,1].set(xlabel='Сутки', ylabel='Расход воды (м$^3$/с)', title='Проверк
а')
ax[0,1].grid()
ax[0,1].set_ylim(0, 2000)

# Диаграмма рассеяния (точки)
ax[1,0].scatter(y_train, train_pred, label='Обучение')
ax[1,0].set_xlim(0, 2000)
ax[1,0].set_ylim(0, 2000)
ax[1,0].set(xlabel='Наблюдения (м$^3$/с)', ylabel='Прогноз (м$^3$/с)', tit
le='Обучение')
ax[1,0].grid()
ax[1,1].scatter(y_test, test_pred, color='orange', label='Проверка')
ax[1,1].set_xlim(0, 2000)
ax[1,1].set_ylim(0, 2000)
ax[1,1].set(xlabel='Наблюдения (м$^3$/с)', ylabel='Прогноз (м$^3$/с)', tit
le='Проверка')
ax[1,1].grid()
```

```
# сравниваем метрики для обучения и теста
train_rmse = mean_squared_error(y_train, train_pred, squared=False)
test_rmse = mean_squared_error(y_test, test_pred, squared=False)
train_nse = r2_score(y_train, train_pred)
test_nse = r2_score(y_test, test_pred)

ax[2,0].bar(['Обучение', 'Проверка'], [train_rmse, test_rmse], label='Обуч
ение')
ax[2,0].set(xlabel='', ylabel='RMSE', title='RMSE')
ax[2,0].annotate(str(round(train_rmse, 3)), (0, train_rmse*0.85), ha='cent
er', va='baseline', size=20)
ax[2,0].annotate(str(round(test_rmse, 3)), (1, test_rmse*0.85), ha='center
', va='baseline', size=20)
ax[2,1].bar(['Обучение', 'Проверка'], [train_nse, test_nse], color='orange
', label='Проверка')
ax[2,1].set(xlabel='', ylabel='NSE', title='NSE')
ax[2,1].annotate(str(round(train_nse, 3)), (0, train_nse*0.85), ha='center
', va='baseline', size=20)
ax[2,1].annotate(str(round(test_nse, 3)), (1, test_nse*0.85), ha='center',
 va='baseline', size=20)
ax[2,1].set_ylim(0, 1)
fig.tight_layout()
plt.legend()
plt.show()
```

1.3.2 Задания для самостоятельного выполнения

Для выполнения заданий добавьте блок кода ниже этого блока текста.

1. Поменяйте заблаговременность прогноза с 1 на 3, 5, 7 дней. Как меняется качество прогноза? При какой заблаговременности коэффициент Нэша-Сатклиффа на проверке становится менее 0.5?
2. Уберите осадки из предикторов. Как меняется качество прогноза?
3. Уберите расходы воды из предикторов. Как меняется качество прогноза?

1.3.3 Основные полученные результаты

Построив модель множественной линейной регрессии заданной заблаговременности, мы получили представление о том, что:

- такие модели используют автокорреляцию для прогнозирования будущих расходов воды;
- на малых заблаговременностях (1-2 суток) такие модели достаточно эффективны;
- чем больше заблаговременность, тем хуже прогноз;
- предыдущие значения расходов воды имеют большую предиктивную силу, чем осадки.

2. Вторая модель машинного обучения: искуственная нейронная сеть (ИНС)

```
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPRegressor
```

2.1.1 Преобразование данных

Исходные данные будут нами использованы те же, что и для построения первой модели.

Для входа модели многослойного перцептрона требуется стандартизовать ряды предикторов в пределах значений $[0,1]$:

$$\hat{X} = \frac{X - E[X]}{\sigma[X]}$$

Для этого используется встроенная функция библиотеки `sklearn - StandardScaler`. Стандартизация необходима во избежание сбоя при нелинейной трансформации в функции активации перцептрона.

Ряд предиката стандартизировать не требуется, но нужно изменить его размерность с 2D на 1D.

```
# создаем объект скейлера
Scaler = StandardScaler()
# стандартизируем (скейлим) данные
X_scaled = Scaler.fit_transform(X)
# разделяем отскейленные данные на обучение...
X_scaled_train = X_scaled[:train_size]
# ...и проверку
X_scaled_test = X_scaled[train_size:]
# изменяем размерность массива предиката
y_train_1d = y_train.values.flatten()
# смотрим на размеры массивов
print(X_scaled_train.shape, y_train_1d.shape)
print(X_scaled_test.shape, y_test.shape)
```

```
 (2192, 44) (2192,)
(712, 44) (712, 1)
```

2.1.2 Задание параметров ИНС

```
# задаем некоторые параметры модели
# mlp_params = {
#     'hidden_layer_sizes': (30, ),      # размер скрытого слоя нейронов,
может содержать несколько слоев, разделенных запятыми, например (5, 5,) ил
и (50, 5,)
#     'activation': 'relu',             # функция активации, варианты 'ide
ntity', 'logistic', 'tanh', 'relu'
#     'solver': 'adam',                 # алгоритм поиска глобального опти
мума функции ошибки, варианты 'lbfgs, 'sgd', 'adam'
#     'alpha': 0.00005,                 # регуляризационный член для алгор
итма поиска решений
#     'max_iter': 5000,                 # количество расчетных итераций
#     'early_stopping': True,           # если True, то на каждом расчетно
м шаге выделяется 10% данных для валидации, и если валидационная метрика н
е снижается в течение 10 шагов, алгоритм останавливается
#     'verbose': True,                 # вывод метрик на каждом шаге
#     'random_state': 123               # начальное число для генератора с
лучайных чисел - для воспроизводимости результатов
# }

mlp_params = {
    'activation':'logistic',
    'alpha': 5e-05,
    'batch_size': 'auto',
    'beta_1': 0.9,
    'beta_2':0.999,
    'early_stopping':True,
    'epsilon':1e-08,
    'hidden_layer_sizes':(50,),
    'learning_rate':'constant',
    'learning_rate_init':0.001,
```

```
    'max_fun':15000,
    'max_iter':15000,
    'momentum':0.9,
    'n_iter_no_change':10,
    'nesterovs_momentum':True,
    'power_t':0.5,
    'random_state':123,
    'shuffle':True,
    'solver':'adam',
    'tol':0.0001,
    'validation_fraction':0.1,
    'verbose':True,
    'warm_start':False
}
# создаем объект
ann = MLPRegressor(**mlp_params)
# обучаем
ann.fit(X_scaled_train, y_train_1d)
# прогноз по массиву данных для обучения
train_pred_ann = ann.predict(X_scaled_train)
# прогноз по массиву данных для проверки
test_pred_ann = ann.predict(X_scaled_test)
# нарисуем график эволюции метрик на обучении и проверке
ax = pd.DataFrame({'RMSE': np.sqrt(ann.loss_curve_),
'NSE': ann.validation_scores_}).plot(secondary_y = ['NSE'],
figsize=(10, 6),
mark_right=False)

ax.set_xlabel('Шаг обучения (эпоха)')
ax.set_ylabel('RMSE на обучении')
ax.right_ax.set_ylabel('NSE на проверке')

# рисуем два гидрографа для обучения и проверки
fig, ax = plt.subplots(nrows=3, ncols=2, figsize=(10,10), dpi=100)
ax[0,0].plot(range(len(y_train)), y_train, label='Наблюдения', linewidth=3
.0)
ax[0,0].plot(range(len(y_train)), train_pred_ann, label='Прогноз', c='oran
gered')
ax[0,0].set(xlabel='Сутки', ylabel='Расход воды (м$^3$/с)', title='Обучени
е')
ax[0,0].set_ylim(0, 2000)
ax[0,0].grid()
ax[0,1].plot(range(len(y_test)), y_test, label='Наблюдения', linewidth=3.0
)
ax[0,1].plot(range(len(y_test)), test_pred_ann, label='Прогноз', color='or
ange')
ax[0,1].set(xlabel='Сутки', ylabel='Расход воды (м$^3$/с)', title='Проверк
а')
ax[0,1].set_ylim(0, 2000)
ax[0,1].grid()

# Диаграмма рассеяния (точки)
ax[1,0].scatter(y_train, train_pred_ann, label='Обучение')
ax[1,0].set_xlim(0, 2000)
ax[1,0].set_ylim(0, 2000)
ax[1,0].set(xlabel='Наблюдения (м$^3$/с)', ylabel='Прогноз (м$^3$/с)', tit
le='Обучение')
ax[1,0].grid()
ax[1,1].scatter(y_test, test_pred_ann, color='orange', label='Проверка')
ax[1,1].set_xlim(0, 2000)
```

```
ax[1,1].set_ylim(0, 2000)
ax[1,1].set(xlabel='Наблюдения (м$^3$/с)', ylabel='Прогноз (м$^3$/с)', tit
le='Проверка')
ax[1,1].grid()

# сравниваем метрики для обучения и теста
train_rmse = mean_squared_error(y_train, train_pred_ann, squared=False)
test_rmse = mean_squared_error(y_test, test_pred_ann, squared=False)
train_nse = r2_score(y_train, train_pred_ann)
test_nse = r2_score(y_test, test_pred_ann)

ax[2,0].bar(['Обучение', 'Проверка'], [train_rmse, test_rmse], label='Обуч
ение')
ax[2,0].set(xlabel='', ylabel='RMSE', title='RMSE')
ax[2,0].annotate(str(round(train_rmse, 3)), (0, train_rmse*0.85), ha='cent
er', va='baseline', size=20)
ax[2,0].annotate(str(round(test_rmse, 3)), (1, test_rmse*0.85), ha='center
', va='baseline', size=20)
ax[2,1].bar(['Обучение', 'Проверка'], [train_nse, test_nse], color='orange
', label='Проверка')
ax[2,1].set(xlabel='', ylabel='NSE', title='NSE')
ax[2,1].annotate(str(round(train_nse, 3)), (0, train_nse*0.85), ha='center
', va='baseline', size=20)
ax[2,1].annotate(str(round(test_nse, 3)), (1, test_nse*0.85), ha='center',
 va='baseline', size=20)
ax[2,1].set_ylim(0,1)
fig.tight_layout()
plt.legend()
plt.show()
```

2.1.3 Задания для самостоятельного выполнения

1. Поменяйте последовательно заблаговременность прогноза с 1 до 7 суток. На какой заблаговременности критерий NSE становится меньше 0.6?
2. Поменяйте функцию активации и алгоритм поиска локального оптимума. Как это влияет на количество эпох в обучении?
3. Поменяйте количество нейронов в скрытом слое ИНС в пределах 5 - 100 и количество слоев (1 - 5). Как это влияет на количество эпох в обучении? Как это влияет на критерии?

2.1.4 Основные полученные результаты

- у ИНС МСП значительно больше параметров, чем у модели линейной регрессии, которые обусловливают нелинейные преобразования предикторов для результата
- количество нейронов на скрытых слоях влияет на точность воспроизведения тренингового набора данных и во много определяет ошибки модели и переобучение
- эффективность модели МСП выше, чем линейной регрессии, на всех заблаговременностях

3. Оптимизация параметров ИНС с помощью сеточного поиска с кросс-валидацией

Параметров ИНС достаточно много, и не всегда есть четкое понимание как они влияют на результат. Для оптимальной настройки параметров модели имеет смысл использовать автоматические процедуры оптимизации, такие, как сеточный поиск с кросс-валидацией.

```
from sklearn.model_selection import KFold, GridSearchCV
import pickle

cv_split_method = KFold(n_splits=10)
cv_scorings = {
    'MSE': make_scorer(mean_squared_error),
    'NSE': make_scorer(r2_score),
    'default': make_scorer(r2_score),
```

```
    }
param_grid = \
{
    'activation':['identity', 'logistic', 'tanh', 'relu'],
    # 'alpha':[5e-05],
    # 'batch_size':['auto'],
    # 'beta_1':[0.9],
    # 'beta_2':[0.999],
    'early_stopping':[True],
    # 'epsilon':[1e-08],
    'hidden_layer_sizes':[(10,), (10, 10, ), ],
    # 'learning_rate':['constant'],
    # 'learning_rate_init':[0.001],
    'max_fun':[15000],
    'max_iter':[15000],
    # 'momentum':[0.9],
    # 'n_iter_no_change':[10],
    # 'nesterovs_momentum':[True],
    'power_t':[0.1, 0.5, 0.9],
    # # 'random_state':[123],
    # 'shuffle':[False, True],
    'solver':['lbfgs', 'sgd', 'adam'],
    # 'tol':[0.0001],
    # 'validation_fraction':[0.1],
    # 'verbose':[False],
    # 'warm_start':[False]
}

# gs_ann = MLPRegressor(random_state=123)
# start = time()
# grid_search_ann =  GridSearchCV(gs_ann,
#                                     param_grid=param_grid,
#                                     cv=cv_split_method,
#                                     scoring=cv_scorings,
#                                     n_jobs=-1,
#                                     refit='NSE',
#                                     verbose=True)
# grid_search_ann.fit(X_scaled_train, y_train_1d)
# print("GridSearchCV took %.2f seconds." % (time() - start))
# сохраняем лучшую модель из поиска
trained_ann = grid_search_ann.best_estimator_
filename = '/content/drive/My Drive/IWP HYDRO DDM/models/MLP_model_Qt+%d.p
kl' % steps_predict
print(filename)
pickle.dump(trained_ann, open(filename, 'wb'))
print(grid_search_ann.best_estimator_)
print(grid_search_ann.cv_results_['mean_test_NSE'][grid_search_ann.best_in
dex_])
```

```
MLPRegressor(activation='identity', alpha=0.0001, batch_size='auto',
beta_1=0.9,
             beta_2=0.999, early_stopping=True, epsilon=1e-08,
             hidden_layer_sizes=(10,), learning_rate='constant',
             learning_rate_init=0.001, max_fun=15000, max_iter=15000,
             momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
             power_t=0.1, random_state=123, shuffle=True, solver='lbfgs',
             tol=0.0001, validation_fraction=0.1, verbose=False,
             warm_start=False)
-0.11976018158872206
```

```
# прогноз по массиву данных для обучения
train_pred_ann = trained_ann.predict(X_scaled_train)
# прогноз по массиву данных для проверки
```

```
test_pred_ann = trained_ann.predict(X_scaled_test)
# рисуем два гидрографа для обучения и проверки
fig, ax = plt.subplots(nrows=3, ncols=2, figsize=(10,10), dpi=100)
ax[0,0].plot(range(len(y_train)), y_train, label='Наблюдения', linewidth=3
.0)
ax[0,0].plot(range(len(y_train)), train_pred_ann, label='Прогноз', c='oran
gered')
ax[0,0].set(xlabel='Сутки', ylabel='Расход воды (м$^3$/с)', title='Обучени
е')
ax[0,0].set_ylim(0, 2000)
ax[0,0].grid()
ax[0,1].plot(range(len(y_test)), y_test, label='Наблюдения', linewidth=3.0
)
ax[0,1].plot(range(len(y_test)), test_pred_ann, label='Прогноз', color='or
ange')
ax[0,1].set(xlabel='Сутки', ylabel='Расход воды (м$^3$/с)', title='Проверк
а')
ax[0,1].set_ylim(0, 2000)
ax[0,1].grid()

# Диаграмма рассеяния (точки)
ax[1,0].scatter(y_train, train_pred_ann, label='Обучение')
ax[1,0].set_xlim(0, 2000)
ax[1,0].set_ylim(0, 2000)
ax[1,0].set(xlabel='Наблюдения (м$^3$/с)', ylabel='Прогноз (м$^3$/с)', tit
le='Обучение')
ax[1,0].grid()
ax[1,1].scatter(y_test, test_pred_ann, color='orange', label='Проверка')
ax[1,1].set_xlim(0, 2000)
ax[1,1].set_ylim(0, 2000)
ax[1,1].set(xlabel='Наблюдения (м$^3$/с)', ylabel='Прогноз (м$^3$/с)', tit
le='Проверка')
ax[1,1].grid()

# сравниваем метрики для обучения и теста
train_rmse = mean_squared_error(y_train, train_pred_ann, squared=False)
test_rmse = mean_squared_error(y_test, test_pred_ann, squared=False)
train_nse = r2_score(y_train, train_pred_ann)
test_nse = r2_score(y_test, test_pred_ann)

ax[2,0].bar(['Обучение', 'Проверка'], [train_rmse, test_rmse], label='Обуч
ение')
ax[2,0].set(xlabel='', ylabel='RMSE', title='RMSE')
ax[2,0].annotate(str(round(train_rmse, 3)), (0, train_rmse*0.85), ha='cent
er', va='baseline', size=20)
ax[2,0].annotate(str(round(test_rmse, 3)), (1, test_rmse*0.85), ha='center
', va='baseline', size=20)
ax[2,1].bar(['Обучение', 'Проверка'], [train_nse, test_nse], color='orange
', label='Проверка')
ax[2,1].set(xlabel='', ylabel='NSE', title='NSE')
ax[2,1].annotate(str(round(train_nse, 3)), (0, train_nse*0.85), ha='center
', va='baseline', size=20)
ax[2,1].annotate(str(round(test_nse, 3)), (1, test_nse*0.85), ha='center',
 va='baseline', size=20)
ax[2,1].set_ylim(0,1)
fig.tight_layout()
plt.legend()
plt.show()
```