Применение Recurrent Sigmoid Piecewise нейрона для прогнозирования временных рядов

Введение

Необходимость прогнозировать временные ряды возникает в множестве сфер и контекстов: экономика, социология, метеорология, как составляющая часть различных систем принятия решений и др. На основе прогнозов строятся планы и принимаются решения, а неправильный прогноз может привести к неправильным планам и решениям. Именно поэтому важность и актуальность задачи прогнозирования сложно переоценить. При этом существует множество факторов, значительно усложняющих решение этой задачи: нестационарный характер прогнозируемого объекта/процесса, большой уровень шума в имеющихся данных, малое количество имеющихся данных и другие.

Класические подходы к решению задачи прогнозирования в основном основываются на применении математической статистики и теории вероятности и обычно предполагают как минимум слабую стационарность прогнозируемого объекта или некоторый конкретный вид нестационарности. Использование методов на основе искусственного интеллекта позволяет избавиться от отдельных предположений и недостатков классических подходов. На сегодня наиболее перспективным направлением интеллектуальных технологий считаются искусственные нейронные сети, в том числе благодаря их универсальности и поразительным результатам, полученным в различных сферах их использования, таких как: анализ изображень, видео, текста, языка и другие. Однако, большинство современных методов прогнозирования на основе нейронных сетей и/или других интеллектуальных подходов не решают проблему непостоянного характера прогнозируемого ряда: в случае неожиданного изменения "поведения" процесса качество прогноза зачастую значительно ухудшается. Данная работа предлагает схему построения прогнозирующей модели, способной динамически корректировать свои текущие прогнозы исходя из ошибок предыдущих прогнозов.

Постановка задачи

Имеется временной ряд $x_1, ..., x_N$, сгенерированный некоторым вероятностным процессом $\{X_t\}$ с неизвестными совместными распределениями:

$$p(x_{t+k}, x_t, x_{t-1}, ..., x_{t-n}).$$

Процесс $\{X_t\}$ может быть как стационарным так и нестационарным. Если процесс нестационарный - в общем случае данная задача прогнозирования не имеет решения, так как вероятностные распределения нестационарного ряда в теории могут "меняться" как угодно, и распределения в будущем могут не иметь ничего общего с распределениями, на основе которых был сгенерирован имеющийся временной ряд. Однако на практике изменение вероятностных распределений нестационарных процессов с течением времени не происходит совершенно случайным образом. Поэтому прогнозирующие модели, оцененные на имеющемся в наличии временном ряде, обычно работают удовлетворительно на протяжении определенного периода времени даже при условии нестационарности соответствующего процесса.

Необходимо использовать данный временной ряд для нахождения прогнозирующей модели вида:

$$\hat{x}_{t+k} = f^*(x_t, ..., x_{t-n}),$$

которая минимизирует математическое ожидание ошибки:

$$f^* = argmin_f\{E_{p(x_{t+k}, x_t, x_{t-1}, ..., x_{t-n})}[L(f(x_t, ..., x_{t-n}), x_{t+k})]\},$$

где:

- $L : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_{\geq 0}$ - функция ошибки, часто используется либо квадратическая $L(x, \hat{x}) = (x - \hat{x})^2$ либо абсолютная ошибка $L(x, \hat{x}) = |x - \hat{x}|$
- k - горизонт прогнозирования, т.е. на сколько "шагов" вперед необходимо выполнить прогноз; данное значение задается исходя из конкретной потребности в прогнозе и является частью постановки задачи
- n - размер предыстории, т.е. количество предыдущих значений временного ряда, которые используются для получения прогноза.

Поскольку рассчитать настоящее математическое ожидание невозможно (соответствующие вероятностные распределения неизвестны), вместо него используется среднее значение функции ошибки на тестовой подвыборке временного ряда:

$$f^* = argmin_f\{\frac{1}{M} \sum_{t=N-k-M+1}^{N-k} L(f(x_t, ..., x_{t-n}), x_{t+k})\},$$

где M - количество примеров, входящих в тестовую подвыборку.

Основные существующие методы прогнозирования

Линейные модели на основе модели ARIMA.
В основе модели ARIMA лежит более простая модель ARMA, использующаяся для описания временного ряда X_t следующим образом:

$$X_t = c + \sum_{i=1}^{p} a_i X_{t-i} + \sum_{j=1}^{q} b_j \varepsilon_{t-j} + \varepsilon_t,$$

где:

- ε_t - стационарный временной ряд, представляющий собой шум с нулевым математическим ожиданием;
- $c, a_1, ..., a_p, b_1, ..., b_q$ - параметры модели.

Модель ARIMA(p,d,q) обобщает модель ARMA(p, q) используя для этого оператор разности временного ряда порядка Δ^d:

$$\Delta^d X_t = c + \sum_{i=1}^{p} a_i \Delta^d X_{t-i} + \sum_{j=1}^{q} b_j \varepsilon_{t-j} + \varepsilon_t,$$

где Δ^d - оператор разности временного ряда порядка d (последовательное взятие d раз разностей первого порядка — сначала от временного ряда, затем от полученных разностей первого порядка, затем от второго порядка и т. д.).

Существуют разные методы для построения прогнозирующих ARIMA моделей, среди наиболее известных - линейная регрессия, методология Бокса-Дженкинса, опорная векторная регрессия. Линейная регрессия заключается в построении частного случая ARIMA модели вида:

$$\Delta^d X_t = \sum_{i=1}^{p} a_i \Delta^d X_{t-i} + a_0,$$

где вектор параметров $\vec{a} = [a_0, a_1, ..., a_p]$ оценивается с помощью метода наименьших квадратов:

$$\vec{a} = (A^T A)^{-1} A^T y,$$

где A, y - матрица и вектор получаемые из исходного временного ряда применяя оператор Δ^d и скользящее окно размера p.
Методология Бокса-Дженкинса позволяет оценивать полную ARIMA модель, но часто требует "экспертного вмешательства" для определения параметров p, q, d, так как существуют различные тесты для их определения, и необходимо выбирать тот либо иной тест и критические значения выбранного теста.
Метод опорной векторной регрессии применяет идеи метода опорных векторов к задаче регрессии: для заданных пар $x_i, y_i; x_i \in \mathbb{R}^n, y_i \in \mathbb{R}, i =$

$1, ..., N$ (полученных после применения скользящего окна к исходному временному ряду) оптимальная линейная модель $\hat{y}_i = f(x_i; w) = \sum_{i=1}^{n} w_i x_i + w_0$ находится путем решения оптимизационной задачи:

$$minimize : \frac{1}{2}\|w\|^2$$

$$where : |y_i - f(x_i; w)| \leq \varepsilon, i = 1, ..., N,$$

где $\varepsilon \geq 0$ - порог допустимой ошибки модели. В случае наличия случайных выбросов в обучающих данных вводятся дополнительные переменные ξ_i, ξ_i^* и регуляризационный параметр $C > 0$ (выбираемый "вручную") и решается новая задача:

$$minimize : \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*)$$

$$where :$$

$$y_i - f(x_i; w) \leq \varepsilon + \xi_i^*, i = 1, ..., N$$

$$f(x_i; w) - y_i \leq \varepsilon + \xi_i, i = 1, ..., N$$

$$\xi_i, \xi_i^* \geq 0, i = 1, ..., N.$$

Также существуют нелинейные варианты метода опорной регрессии на основе применения функций-ядер.

Искусственные нейронные сети.

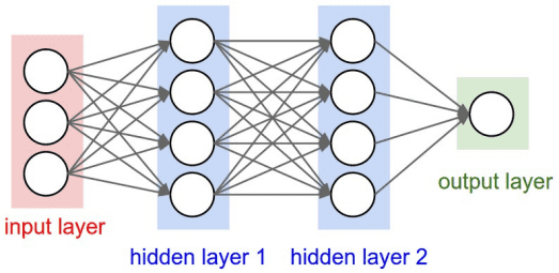
Искусственные нейронные сети представляют собой систему взаимосвязанных искусственных нейронов, где каждый нейрон обычно реализует простую функцию вида:

$$f(x; w) = u(w \cdot x); x, w \in \mathbb{R}^n$$

где $u : \mathbb{R} \to \mathbb{R}$ - некоторая нелинейная функция, называемая функцией активации; w - вектор весовых коэффициентов нейрона, настраиваемых в процессе обучения сети. Наиболее популярные функции активации:

- $ReLU(x; w) = max(0, w \cdot x)$
- $\sigma(x; w) = \frac{1}{1+e^{-w \cdot x}}$
- $tanh(x; w) = tanh(w \cdot x) = \frac{2}{1+e^{-2w \cdot x}} - 1$

В контексте задачи прогнозирования популярными являются искусственные нейронные сети прямого распространения (feedforward neural network) со структурой вида:



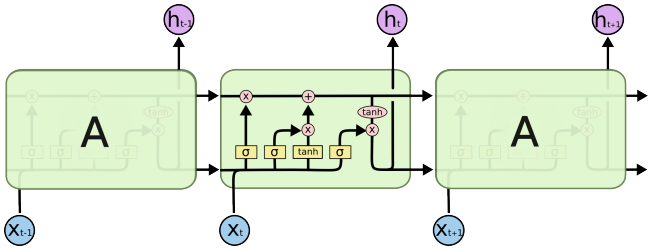
которые по сути являются нелинейной вариацией AR-модели: $X_t = f(X_{t-1}, ..., X_{t-p})$ - где f - функция, реализуемая нейронной сетью. Настройка параметров нейронной сети, также называемая обучением, обычно производится применением некоторой вариации алгоритма градиентного спуска:

- Для относительно небольших сетей и временных рядов можно применять алгоритм Левенберга-Марквардта, который на практике часто находит значение параметров, близкое к оптимальному, и делает это значительно быстрее других алгоритмов (опять же, при условии небольшого количества параметров и длины временного ряда).
- Для больших сетей но относительно коротких временных рядов часто применяют "стандартный" алгоритм градиентного спуска либо его модификации типа алгоритма Adam, LBFGS, где градиент рассчитывается сразу для всего временного ряда.
- Для больших сетей и длинных временных рядов используются "пакетные" вариации алгоритмов из предыдущего пункта, в которых на каждой итерации градиент рассчитывается только для определенного под-множества - "пакета" данных.

Кроме оптимизации непосредственно параметров нейронной сети с заданной структурой также необходимо определить саму структуру сети. Кроме варианта применения определенных эвристик для "ручного" задания структуры также возможно применение алгоритмов автоматического подбора структуры: наиболее популярными являются алгоритмы обучения с подкреплением, эволюционные алгоритмы и алгоритмы семейства МГУА. Основные алгоритмы семейства МГУА были созданы до того, как искусственные нейронные сети стали популярными, поэтому их часто рассматривают как отдельные методы. Однако их также можно рассматривать как различные способы оптимизации структуры и параметров полиномиальных нейронных сетей, в которых используются нейроны с функциональной моделью в виде полиномов Колмогорова-Габора:

$$f(x; w) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i}^{n} w_{ij} x_i x_j + \sum_{i=1}^{n} \sum_{j=i}^{n} \sum_{k=j}^{n} w_{ijk} x_i x_j x_k + ...$$

Рекуррентные нейронные сети.
В задачах обработки естественного языка, таких как построение языковых моделей, автоматический перевод текста и пр. хорошо себя "зарекомендовали"рекуррентные нейронные сети на основе Long Short Term Memory (LSTM) и/или Gated Recurrent Unit (GRU) нейронов. Данные нейроны имеют схожую структуру, которая позволяет уменьшить влияние проблемы затухающего/взрывающегося градиента при обучении рекуррентных моделей с использованием Backpropagation Through time (BPTT) алгоритма на длинных последовательностях. За счет этого, на практике, сети с этими нейронами более стабильны в обучении и имеют большую "точность"при работе на длинных последовательностях. LSTM-нейрон имеет следующую структуру:



Полное математическое описание классического LSTM нейрона:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

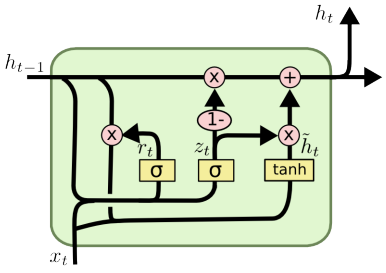
$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * tanh(c_t)$$

GRU нейрон это, по сути, упрощенная версия LSTM нейрона:



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

В данном нейроне вектор выходов h_t так же "выполняет" роль вектора контекста, и используются следующие блоки:

- Блок обновления $z_t(x_t, h_{t-1}; W_z)$, рассчитывающий веса в диапазоне $(0, 1)$, которые применяются для рассчета нового вектора выходов (и, одновременно, контекста) h_t исходя из вектора-кандидата $\tilde{h}_t$ и предыдущего вектора h_{t-1}
- Блок "релевантности" $r_t(x_t, h_{t-1}; W_r)$, рассчитывающий веса в диапазоне $(0, 1)$, которые определяют "релевантность"/"важность" значений предыдущего выходного вектора h_{t-1} при рассчете вектора-кандидата для нового выходного вектора $\tilde{h}_t$
- Блок рассчета вектора-кандидата новых выходов $\tilde{h}_t(x_t, h_{t-1}, r_t; W)$
- Блок рассчета нового вектора выходов $h_t(h_{t-1}, \tilde{h}_t, z_t)$ как взвешенной суммы соответствующих значений из предыдущего вектора h_{t-1} и нового вектора-кандидата $\tilde{h}_t$, где веса для значений под индексом i выбираются как $1 - z_t[i]$ и $z_t[i]$ соответственно.

Регрессионные деревья, boosting, bagging, гибридные подходы. Регрессионные деревья — это метод регрессии, основанный на древовидном рекурсивном разбиении входного пространства на регионы [1]. Используя обучающую выборку выполняется рекурсивная настройка правил, разбивающих пространство $\mathbb{R}^n$ на регионы, называемые конечными листьями. Получившийся набор правил можно представить в виде дерева решений, которые рекурсивно применяются к входному вектору x и определяют конечный лист, к региону которого принадлежит этот вектор: $d(x) = i, i = 1, ..., L$ где L - количество конечных листьев дерева. Финальный прогноз для заданного вектора определяется как среднее выходное значение всех примеров из обучающей выборки, которые принадлежат соответствующему региону:

$$f(x) = \frac{\sum_{j:d(x_j)=d(x)} y_j}{\sum_{j:d(x_j)=d(x)} 1}$$

Настройка правил/решений производится итеративно, где на каждой итерации выбирается правило, оптимизирующее определенный критерий. Наиболее известные и используемый критерии: критерий Джинни, информационный выигрыш, понижение дисперсии.
При настройке и использовании единственной прогнозирующей модели часто возникают две в определенном смысле противоположных проблемы: переобучение и недостаточная точность прогноза, вызванная недостаточной сложностью выбранной модели. Эти проблемы в той либо иной степени свойственны всем вышеперечисленным методам. Для борьбы с этими проблемами часто используют подходы bagging и boosting, как по отдельности так и одновременно.
Метод bagging (сокращение от bootstrap aggregatting) заключается в:

- генерации так называемых bootstrap тренировочных наборов вида $D_b = \{x_i^{(b)}, y_i^{(b)}\}, b = 1, ..., B, i = 1, ..., N$ путем случайной равномерной выборки с возвратом (random sampling with replacement) из оригинального тренировочного набора $D = \{x_i, y_i\}, i = 1, ...N$, где B - количество bootstrap выборок

- обучении/настройки отдельной прогнозирующей модели $f_b(x)$ для каждой bootstrap выборки D_b

- финальная прогнозирующая модель является усреднением выходов всех отдельных bootstrap моделей: $f(x) = \frac{1}{B}\sum_{b=1}^{B} f_b(x)$.

В результате применения описанной процедуры уменьшается проблема переобучения и повышается робастность финальной прогнозирующей модели, что позволяет уменьшить дисперсию ошибки прогноза как в теории [2] так и на практике [3]. Применение bagging подхода к регрессионным деревьям также известно как метод случайного леса [4], и существует множество различных модификаций ([5][6]).
Метод boosting также использует несколько прогнозирующих моделей для рассчета финального прогноза, но другим образом. Существует большое количество различных boosting алгоритмов, большинство из них можно описать следующей схемой:

- Построение прогнозирующих моделей происходит итеративно, каждая следующая модель обучается прогнозировать разницу между суммой прогнозов уже обученных моделей и реальным значением.

- Соответственно, самая первая модель обучается прогнозировать исходный временной ряд.

- В некоторых вариациях, при новой итерации обучения модели также используют т.н. "перевзвешивание"примеров из обучающей выборки, когда в функции ошибки для обучения новой модели используется взвешенная сумма ошибок для отдельных примеров $L(f) = \sum_{i=1}^{N} \theta_i l(f(x_i), y_i)$ и веса для "хороших" примеров с малой ошибкой на предыдущей итерации уменьшаются а веса для "плохих" примеров увеличиваются, таким образом "фокусируя" новую модель на улучшение прогноза именно для этих примеров. Вместо использования взвешенной суммы ошибок также возможно использования семплирования примеров в новую обучающую выборку, где вероятность семплирования конкретного примера обратно пропорциональна значению ошибки прогноза на предыдущей итерации.

- Остановка итераций осуществляется при выполнении определенного критерия, после чего финальная прогнозирующая модель является взвешенной суммой прогнозов отдельных моделей:

$$f(x) = \sum_{i=1}^{M} w_i f_i(x),$$

где M - общее количество итераций, $f_i(x)$ - модель, обученная на i-ой итерации.

Применение boosting также уменьшает дисперсию ошибки и влияние проблемы overfitting. Кроме этого, boosting позволяет использовать простые и не очень точные модели на каждой итерации (например небольшие регрессионные деревья), т.н. weak learners, для получения сложной и точной финальной модели, т.н. strong learner, таким образом адресуя проблему недостаточной сложности отдельных прогнозирующих моделей.
Многие современные работы, посвященные задаче прогнозирования, фокусируются на использовании гибридов нескольких из описанных подходов:

- В работе [7] авторы предлагают комбинацию прогнозирующей модели ARIMA и искусственных нейронных сетей, где в нейронную сеть передаются предыдущие значения временного ряда $x_t, x_{t-1}, ..., x_{t-n}$, текущий прогноз ARIMA модели $\hat{L}_t$ и ошибки предыдущих прогнозов ARIMA модели $r_t, r_{t-1}, ..., r_{t-n}; r_i = x_i - \hat{L}_i$.

- Похожий подход предложен в [8], где для моделирования линейной составляющей временного ряда используется сезонная ARIMA модель, а для моделирования нелинейной составляющей используется метод опорной векторной регрессии с применением нелинейного ядра, модифицированного критерия оптимизации и алгоритма светлячка (firefly algorithm) для оптимизации этого критерия.

- В работе [9] предлагается использование искусственной нейронной сети состоящие из нео-нечетких нейронов и применение МГУА-подобного алгоритма для автоматического подбора структуры и параметров сети. Нео-нечеткий нейрон является системой нечеткого вывода и реализует следующую функцию:

$$y(x_1, x_2) = f_1(x_1) + f_2(x_2); f_i(x_i) = \sum_{j=1}^{n} w_{ji}\mu_{ji}(x_i),$$

  где μ_{ji} - функция принадлежности к нечеткому множеству (наиболее популярными вариантами являются гауссова ф.п., треугольная ф.п. и пр.), w_{ji} - настраиваемый параметр.

- тест

Recurrent Sigmoid Piecewise (RSP) нейрон

В данной работе предлагается новая модель рекуррентного нейрона Recurrent Sigmoid Piecewise (RSP), в основе которой лежит Sigmoid Piecewise (SP) нейрон со следующей математической моделью:

$$SP(x; w_+, w_-, s, k) = \frac{w_+ \cdot x}{1 + e^{-k(s \cdot x)}} + \frac{w_- \cdot x}{1 + e^{k(s \cdot x)}}$$

Используя обозначение сигмоидального нейрона:

$$\sigma(x; s) = \frac{1}{1 + e^{s \cdot x}}$$

и $k = 1$ получаем:

$$SP(x; w_+, w_-, s) = \sigma(x; s)(w_+ \cdot x) + \sigma(x; -s)(w_- \cdot x)$$

Используя равенство $\sigma(x; -s) = 1 - \sigma(x; s)$:

$$SP(x; w_+, w_-, s) = (1 - \sigma(x; s))(w_- \cdot x) + \sigma(x; s)(w_+ \cdot x)$$

Если вместо одного SP нейрона описывается слой из N нейронов, то вместо векторов w_+, w_-, s будут использоваться матрицы W_+, W_-, S:

$$SP(x; W_+, W_-, S) = (1 - \sigma(x; S)) * (W_- \cdot x) + \sigma(x; S) * (W_+ \cdot x)$$

Введя обозначения $z = \sigma(x; S)$, $a = W_- \cdot x$ и $b = W_+ \cdot x$ получаем:

$$SP(x) = (1 - z) * a + z * b$$

Что очень похоже на блок рассчета нового вектора выходов в нейроне GRU:

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

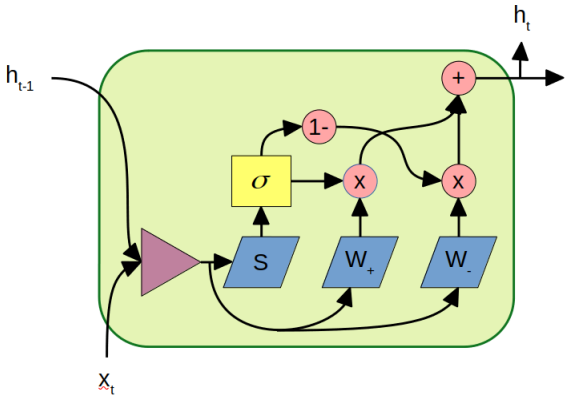
Таким образом, слегка изменив SP нейрон, можно получить его рекуррентную версию, Recurrent Sigmoid Piecewise (RSP) нейрон, который принимает на вход вектор $p_t = [h_{t-1}, x_t]$ и выдает h_t:

$$h_t = RSP(p_t; W_+, W_-, S) = (1 - \sigma(p_t; S)) * (W_- \cdot p_t) + \sigma(p_t; S) * (W_+ \cdot p_t)$$

Либо же, по аналогии с LSTM/GRU нейронами, мат. модель RSP нейрона можно записать в несколько этапов/блоков:

$$z_t = \sigma(S \cdot [h_{t-1}, x_t])$$
$$q_t = W_- \cdot [h_{t-1}, x_t]$$
$$\tilde{h}_t = W_+ \cdot [h_{t-1}, x_t]$$
$$h_t = (1 - z_t) * q_t + z_t * \tilde{h}_t$$

И представить их в виде структурной схемы:



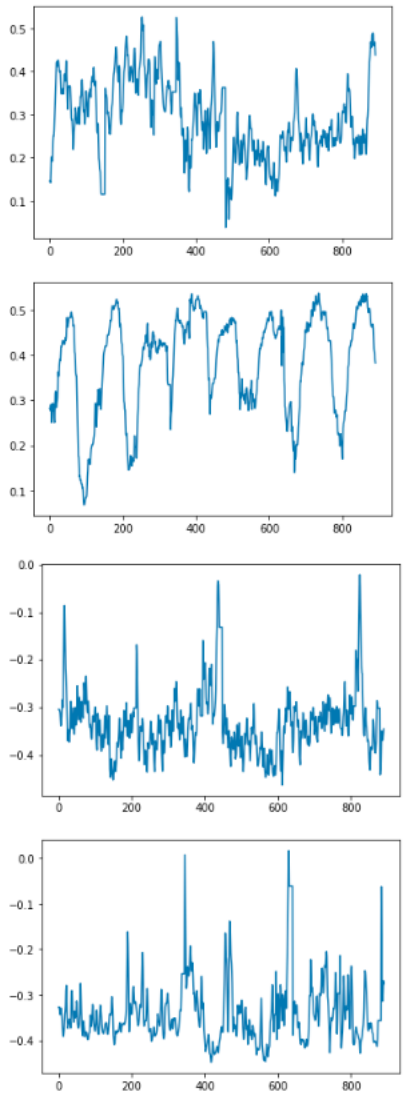
Поверхностно сравнив RSP нейрон с LSTM и GRU нейронами можно сделать следующие наблюдения:

- Математическая модель RSP нейрона проще (используется лишь один нелинейный сигмоидальный блок) чем модели LSTM и GRU нейронов. В задаче прогнозирования временных рядов более простые модели часто предпочтительны на практике.
- При этом, RSP так же как и LSTM и GRU нейроны позволяет забывать определенные значения в векторе контекста при необходимости.

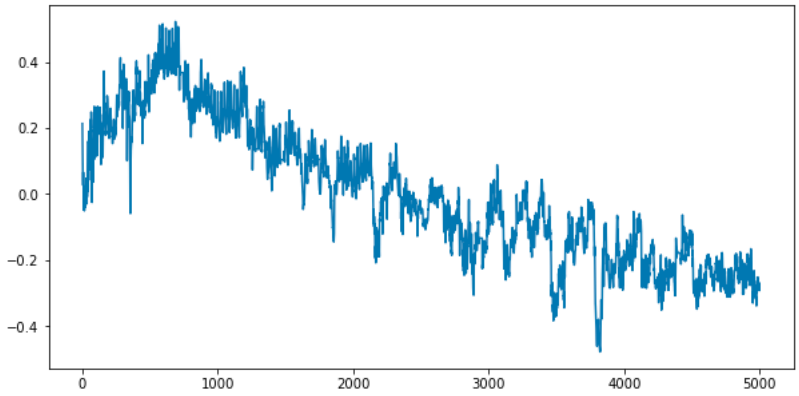
Практическое сравнение нейронов RSP, LSTM, GRU

Для оценки эффективности применения RSP нейрона в моделировании прогнозируемого временного ряда были использованы два реальных временных ряда:

1. Показатели сердечного ритма 4 разных пациентов в различных состояниях [x]:



2. Временной ряд OT (oil temperature, температура масла в электрическом трансформере) из набора данных Electricity Transformer Dataset, который часто используют для оценки методов прогнозирования [x]:



Указанные ряды разбивались на обучающую и тестовую выборки, после чего параметры моделей настраивались на обучающей выборке и рассчитывались метрики на тестовой выборке. Для приведения исходного временного ряда к "более стационарному виду" для каждой обучающей выборки сначала был подобран оптимальный линейный предиктор, после чего перед соответствующей рекуррентной сетью "ставилась" задача прогнозирования отклонения реального значения временного ряда от прогноза линейного предиктора. В качестве моделей для сравнения использовались 2-слойные сети с 1 рекуррентным нейроном в каждом слое - либо LSTM, либо GRU либо RSP, с размерностью вектора скрытого состояния равной 6. Выполнив данную процедуру получаем следующие значения метрик разных моделей:

Табл. 1: Метрики моделей с разными нейронами на тестовой подвыборке показателей сердечного ритма

Тип нейрона	MSE	MAE	MAPE
LSTM	0.0004795	0.0123	0.05
GRU	0.000463	0.0124	0.051
RSP	0.000464	0.0119	0.0496

Табл. 2: Метрики моделей с разными нейронами на тестовой подвыборке временного ряда ETTH OT

Тип нейрона	MSE	MAE	MAPE
LSTM	0.00037	0.0137	0.0668
GRU	0.000361	0.01349	0.0664
RSP	0.000359	0.01345	0.0665

Как видно из рассчитанных тестовых метрик для обоих рядов, сеть с RSP нейронами имеет значения метрик либо очень близкие либо лучше чем у LSTM и GRU сетей, при этом она имеет меньшее количество параметров и более простую структуру нейрона.

Применение рекуррентных сетей на основе RSP нейронов для прогнозирования временных рядов

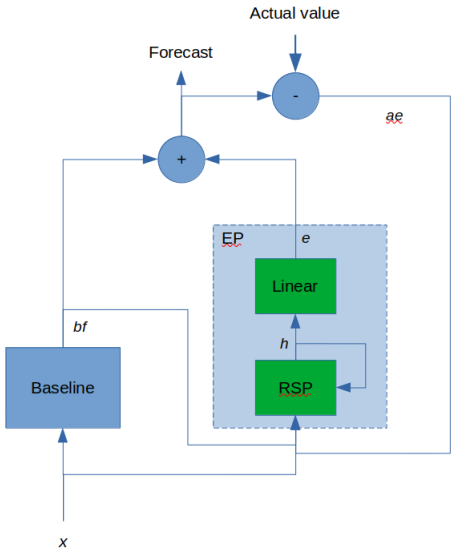
Стандартную линейную ARMA модель можно обобщить следующим образом:

$$X_t = \varepsilon_t + f(X_{t-1}, ..., X_{t-p}) + g(\varepsilon_{t-1}, ..., \varepsilon_{t-q}),$$

где $f : \mathbb{R}^p \to \mathbb{R}, g : \mathbb{R}^q \to \mathbb{R}$ - некоторые функции, в общем случае нелинейные. Наиболее общее описание нелинейной ARMA модели будет иметь вид:

$$X_t = \varepsilon_t + f(X_{t-1}, ..., X_{t-p}, \varepsilon_{t-1}, ..., \varepsilon_{t-q}),$$

где $f : \mathbb{R}^{p+q} \to \mathbb{R}$ - нелинейная функция. На основе этого обобщения предлагается следующая схема прогнозирования временных рядов:



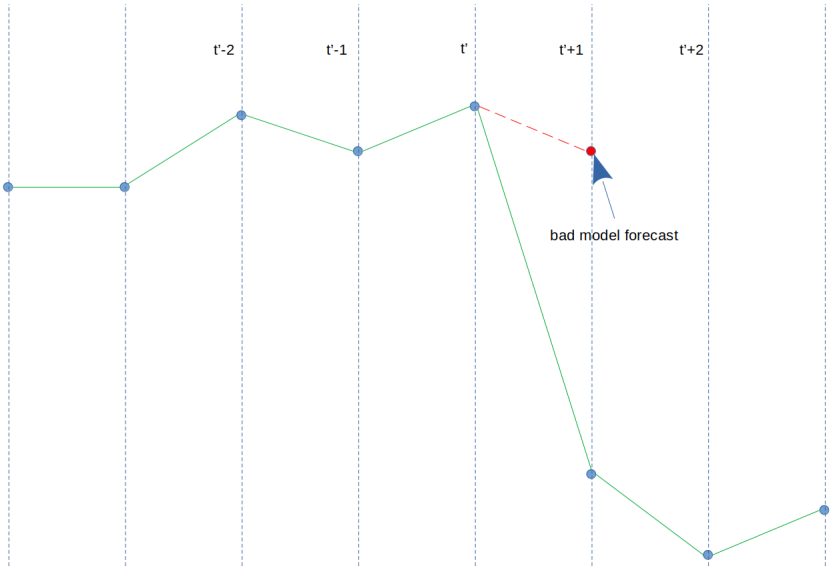
где:

- x - входной вектор, предыдущие значения часового ряда (либо нескольких рядов)
- **Baseline** - "базисная" прогнозирующая модель, рассчитывающая начальную оценку прогноза, например линейная регрессия; соответственно b_t - базисное значение прогноза в момент времени t
- **EP** - error prediction блок, рассчитывающий оценку ошибки прогноза $\hat{e}_t$ базисного метода на основе: входного вектора, самого значения базисного прогноза и настоящей ошибки прогноза с предыдущего шага
- Базисный прогноз b_t и оценка ошибки $\hat{e}_t$ складываются для получения финального прогноза: $f_t = b_t - \hat{e}_t$

- На следующем шаге прогнозирования $t+1$ также рассчитывается настоящая ошибка прогноза с предыдущего шага $e_{t+1} = f_{t+1} - x_{t+1}$ и передается в блок рассчета ошибки прогноза текущего шага

- Блок рассчета ошибки состоит из RSP нейрона и простого линейного слоя. По своей математической модели RSP нейрон может "естественным"способом рассчитывать новое значение коррекции как взвешенную сумму предыдущей ошибки и нового значения контекста.

Основным отличием данной прогнозирующей схемы от обычного использования прогнозирующей модели является блок предсказания ошибки. По сути, данный блок является нелинейной вариацией MA блока в модели ARMA. Использование этого блока позволяет схеме реагировать на изменения в качестве прогноза базисной модели. Например, пускай для некоторого момента времени t' получена достаточно большая ошибка прогноза $e_{t'+1} = f_{t'+1} - x_{t'+1}, e_{t'+1} > 0$, то есть прогноз модели оказался значительно больше реального значения. Одной из возможных причин может быть неожиданный для модели скачок "вниз"временного ряда:



В таком случае можно ожидать, что в момент времени $t'+1$ прогноз базисной модели также окажется больше, и блок ПО сможет его скорректировать путем предсказания оценки ошибки $\hat{e}_{t'+2} > 0$. И наоборот - при неожиданном скачке "вверх"на шаге t' будет получена большая негативная ошибка прогноза $e_{t'+1} < 0$ - тогда на шаге $t'+1$ прогноз базисной модели может также быть меньше реального значения, и блок ПО сможет его скорректировать путем предсказания оценки ошибки $\hat{e}_{t'+2} < 0$. Преимущества данной схемы:

- В качестве базисной модели можно брать прогнозирующую модель, полученную в результате применения любого существующего ме-

тода прогнозирования, и таким образом в процессе обучения EP блок будет пытаться только улучшать прогноз базисной модели.

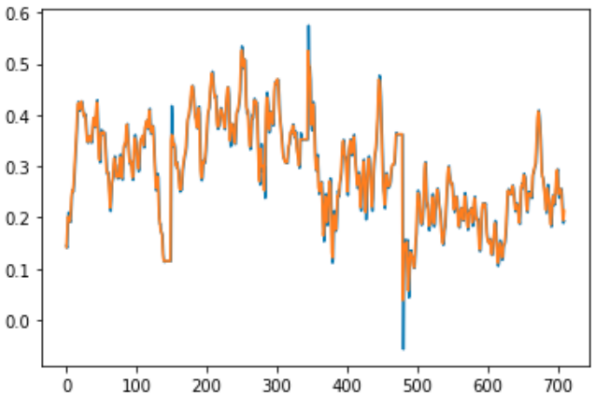
- Рассчет настоящего значения ошибки прогноза с предыдущего шага (шагов) в теории дает возможность EP блоку динамически корректировать текущий прогноз основываясь на значениях предыдущих ошибок прогноза.
- Мат. модель RSP нейрона естественным образом подходит для рассчета некоторой ошибки прогноза.
- В теории возможно поэтапное обучение EP и базисного блоков - на первом этапе обучаем параметры EP блока, на втором - фиксируем их и обучаем параметры базисного блока и т.д.

Практические примеры использования подхода с блоком коррекции на основе рекуррентной RSP сети

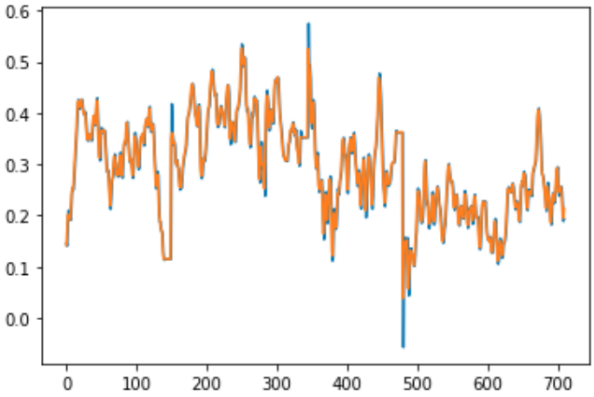
Для первого тестирования используем показатели сердечного ритма 4 разных пациентов в разных состояниях. В качестве базисных моделей выберем простую линейную регрессию и ансамбль регрессионных деревьев, настраиваемый с помощью градиентного бустинга, используя пакет XGBoost. Построив соответствующие базисные модели получаем следующие значения метрик качества прогноза на обучающей и тестовой выборке (усредненные по всем 4 пациентам):

Метод	MSE	MAE	MAPE
Линейная регрессия	train=0.000473 test=0.00048	train=0.01283 test=0.01228	train=0.058 test=0.05
XGBoost	train=0.00022 test=0.000479	train=0.00728 test=0.0128	train=0.03818 test=0.0506

Пример прогнозов линейного предиктора для одного пациента на обучающей:



и тестовой:



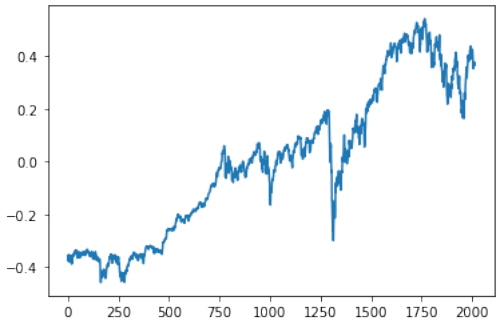
выборках.

После "фиксации" базисных предикторов, добавления блока коррекции на основе RSP нейрона и обучения его параметров получаем следующие значения метрик:

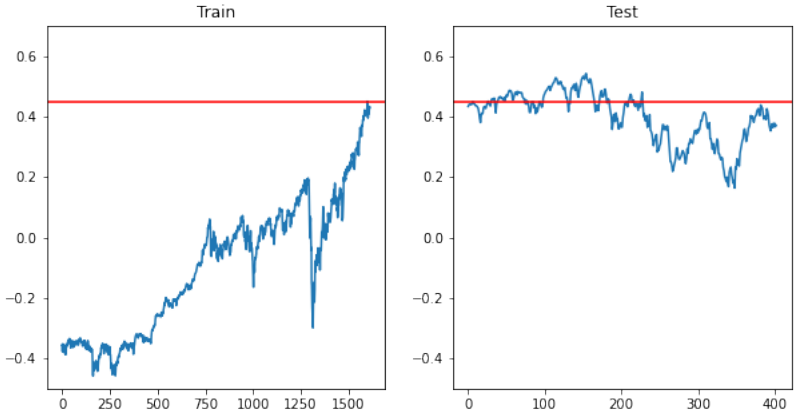
Метод	MSE	MAE	MAPE
Линейная регрессия с блоком коррекции	train=0.00041 test=0.000411	train=0.01178 test=0.01145	train=0.0565 test=0.047
XGBoost с блоком коррекции	train=0.000213 test=0.000464	train=0.00716 test=0.0125	train=0.038 test=0.0502

где значения всех метрик для обоих методов улучшились.

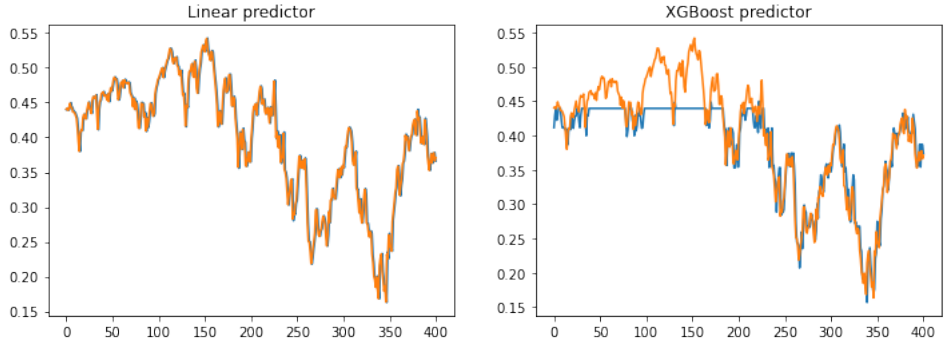
В качестве второго примера возьмем временной ряд дневных значений индекса DJI за период 2015/1/1 - 2023/1/1:



Разбив данный временной ряд на обучающую и тестовую выборки:

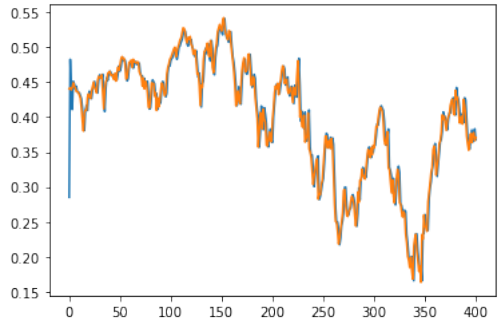


можно заметить, что в тестовой выборке встречаются периоды, в которых значение временного ряда превышает максимальное значение, присутствующее в обучающей выборке. После настройки параметров базисных предикторов - линейной регрессии и ансамбля регрессионных деревьев - на обучающей выборке получаем следующие результаты прогнозов на тестовой выборке:

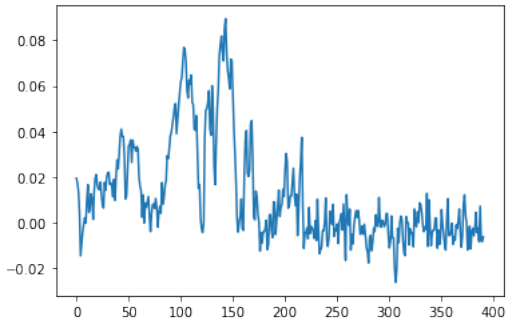


Очевидно, что ансамбль регрессионных деревьев очень плохо справился с прогнозированием значений временного ряда в соответствующих периодах. Причиной таких прогнозов является принцип построения таких деревьев, где значение прогноза для определенного входного вектора x является усредненное значение выходного значения для всех примеров в листе дерева, куда попадает вектор x. Существуют различные методы исправления данного недостатка, однако такой случай можно рассматривать как конкретный пример проблемы непостоянного характера прогнозируемого временного ряда. Механизм коррекции, предлагаемый в данной статье, в теории способен "реагировать" на ухудшение качества прогноза базисного предиктора в таких ситуациях и корректировать его по возможности. Для этого, при настройке параметров блока коррекции также можно добавлять шум в прогнозы базисного предиктора, чтобы "учить" блок реагировать необходимым образом. Применяя этот подход

получаем такой прогноз базисного предиктора XGBoost с обученным блоком коррекции на тестовой выборке:



где хорошо видно, что добавление блока коррекции позволило существенно улучшить качество прогноза в проблемных регионах. Построив график прогнозов отдельно блока коррекции можно увидеть, как он реагирует на плохой прогноз базисного предиктора и "исправляет" его:



Значения метрик для базисных предикторов и вариантов с коррекцией:

Метод	MSE	MAE	MAPE
Линейная регрессия	train=0.000128 test=0.0002515	train=0.0073 test=0.012	train=0.1654 test=0.0336
XGBoost	train=$2.03*10^{-5}$ test=0.001	train=0.00328 test=0.02476	train=0.09 test=0.06
Линейная регрессия с блоком коррекции	train=0.0001276 test=0.0002514	train=0.00725 test=0.012	train=0.165 test=0.0336
XGBoost с блоком коррекции	train=0.00011 test=0.0003	train=0.00686 test=0.0127	train=0.155 test=0.035

Выводы и дальнейшие направления работы

Ссылки

1. Breiman L. Classification and Regression Trees. Boca Raton, FL: Chapman & Hall; 1993.
2. BÜHLMANN, P., YU, B. Analyzing bagging. In: Annals of Statistics 30 (4), 2002, pp. 927–961.
3. BREIMAN, L. Bagging predictors. In: Machine Learning, 24(2), 1996, pp. 123–140.
4. BREIMAN, L. Random forest. In: Machine Learning 45 (1), 2001, pp. 5–32.
5. HASTIE, T., TIBSHIRANI, R., FRIEDMAN, J. The Elements of Statistical Learning. 2nd ed. New York: Springer, 2009.
6. JAMES, G., WITTEN, D., HASTIE, T., TIBSHIRANI, R. An introduction to statistical learning. In Springer 112, 2013, pp. 18.
7. Fawzy, Haitham, EL Houssainy A. Rady, and Amal Mohamed Abdel Fattah. "A Comparative Simulation Study of ARIMA and Computational Intelligent Techniques for Forecasting Time Series Data." Journal of Statistics Applications & Probability Letters 11, no. 1 (2022): 1-7.
8. Ngo, Ngoc-Tri, Anh-Duc Pham, Thi Thu Ha Truong, Ngoc-Son Truong, and Nhat-To Huynh. "Developing a hybrid time-series artificial intelligence model to forecast energy use in buildings."Scientific Reports 12, no. 1 (2022): 15775.
9. Zaychenko, Yuriy, Helen Zaichenko, and Oleksii Kuzmenko. "Investigation of Artificial Intelligence Methods in the Short-Term and Middle-Term Forecasting in Financial Sphere."In System Analysis and Artificial Intelligence, pp. 307-322. Cham: Springer Nature Switzerland, 2023.

ДИНАМИЧЕСКИЙ ВЕС, КОТОРЫЙ ПРИМЕНЯЕТСЯ К ВЫХОДУ БЛОКУ КОРРЕКЦИИ - ЧЕМ МЕНЬШЕ ПРЕДЫДУЩАЯ ОШИБКА, ТЕМ МЕНЬШЕ НЕОБХОДИМОСТЬ В КОРРЕКЦИИ НОВОГО ПРОГНОЗА И НАОБОРОТ