Оглавление

Глава 1. Введение

Временной ряд - это серия или последовательность данных, собранных за определенное время, например, цены на акции, погодные данные или показания датчиков спутника. Для прогноза временных рядов зачастую используются статистические методы, например ARIMA и ETS, однако с развитием машинного обучения появился интерес в использовании нейронных сетей для предсказания временного ряда.

Нейронные сети - это модели, которые имитируют работу мозга и способны обучаться на больших объемах данных. Одним из типов нейронных сетей являются рекуррентные нейронные сети (RNN), которые специально разработаны для работы с последовательными данными и могут учитывать долгосрочные зависимости во времени.

В данной работе сравниваются общепринятые статистические методы ARIMA и ETS с рекуррентными нейронными сетями для прогнозирования временных рядов на различных наборах данных и анализируются их достоинства и недостатки. С помощью данной информации далее в выпускной квалификационной работе будут спрогнозированы временные ряды, полученные с помощью спутника LASP.

Глава 2. Теоретическая часть.

2.1. Временные ряды.

Главное отличие временного ряда от случайной выборки заключается в распределении - во временном ряде может присутствовать несколько зависимых с различными распределениями, которые зависят от выборки, в то время как случайная выборка состоит из независимых величин с одинаковым распределением. Определим временной ряд как Y_t - значение переменной в период t. Тогда $LY_t = Y_{t-1}$ - лаг (первый) переменной с помощью лагового оператора L. В таком случае $LLL...LY_t = L^kY_t = Y_{t-k}$ - k-ый лаг переменной. Введем понятие станционарности. Временной ряд называется строго станционарным, если никакой сдвиг во времени не меняет ни одну функцию плотности распределения ряда. В случае если определение станционарности вводить для функции распределения, то такой ряд имеет сильную станционарность. В случае строгой станционарности математическое ожидание временного ряда не зависит от времени t, так как по определению матиматического ожидания $E(Y_t) = \int_{-\infty}^{\infty} \xi \cdot f_t(\xi)d\xi = \mu < \infty$, а при строгой станционарности $f_t(\xi)$ не меняется. Аналогичное следствие справедливо и для дисперсии временного ряда, $Var(Y_t) = \gamma_0$.

Из строгой станционарсти также следует что благодаря одним и тем же значениям матожидания и дисперсии сдвиг на τ значений вперед или назад во времени не повлияет на ковариацию, т.е.

$$Cov(Y_t, Y_{t-\tau}) = \iint (Y_t - \mu) \cdot (Y_{t-\tau} - \mu) \cdot f(Y_t, Y_{t-\tau})dY_t dY_{t-\tau} = \gamma_\tau \tag{1}$$

Совокупностью таких значений ковариаций при всевозможных значениях расстояния между моментами времени τ называется автоковариационной функцией случайного процесса и обозначается γ_τ. Введем также понятие коэффициента автокорреляции:

$$\rho_\tau = Corr(Y_t, Y_{t-\tau}) = \frac{Cov(Y_t, Y_{t-\tau})}{\sqrt{Var(Y_t)Var(Y_{t-\tau})}} = \frac{\gamma_\tau}{\gamma_0} \tag{2}$$

Данный коэффициент показывает статистическую зависимость значений временного ряда со сдвигом во времени τ.

Также существует понятие слабой станционарности. Матожидание и дисперсия также не зависят от времени, однако автокорреляция зависит только от сдвига τ. Каждый строго станционарный ряд слабо танционарен, но обратное не всегда верно. [1]

Выделяют следующие важные простейшие примеры временных рядов.

[1] Например, третий центральный момент $u_3 = \int_{-\infty}^{\infty}(x - E(X))^3 f(x)dx$ может зависить от времени.

1. Белый шум. Белым шумом называется последовательность независимых и одинаково распределенных случайных величин, таких, что $E(Y_t) = 0, Var(Y_t) = \sigma^2, Cov(Y_i, Y_j) = 0$ при $i \neq j$, поэтому данный процесс слабо станционарен. В случае если временной ряд распределен нормально, то такой процесс становится строго станционарен. Определяется как ϵ_t.

2. Процесс авторегрессии первого порядка - AR(1). Данный процесс задается как $Y_t = \delta + \theta Y_{t-1} + \epsilon_t$. Вычислим математичесское ожидание, дисперсию и автоковариацю.

$$\begin{aligned} E(Y_t) = E(\delta + \theta Y_{t-1} + \epsilon_t) = \delta + \theta E(Y_{t-1}) + E(\epsilon_t) = \delta + \theta\mu + 0 \\ \implies \mu = \frac{\delta}{1-\theta} \\ Var(Y_t) = Var(\delta + \theta Y_{t-1} + \epsilon) = \theta^2 Var(Y_{t-1}) + Var(\epsilon_t) = \theta^2\gamma_0 + \sigma^2 \\ \implies \frac{\sigma^2}{1-\theta^2} \\ Cov(Y_t, Y_{t-1}) = Cov(\delta + \theta Y_{t-1} + \epsilon, Y_{t-1}) = Cov(\theta Y_{t-1} + \epsilon_t, Y_{t-1}) = \\ = \theta Cov(Y_{t-1}, Y_{t-1}) + Cov(\epsilon_t, Y_{t-1}) = \theta\gamma_0 + 0 \\ \implies \gamma_1 = \theta\frac{\sigma^2}{1-\theta^2} \\ \implies \gamma_k = Cov(Y_t, Y_{t-\tau}) = \theta^\tau\frac{\sigma^2}{1-\theta^2} \end{aligned} \tag{3}$$

3. Процесс скользящего среднего - MA(q), является частным случаем разложения Волда [2]. Определяется как $Y_t\delta + \epsilon_t + \alpha\epsilon_{t-1}$. Так как это линейная комбинация белых шумов, то математическое ожидание, ковариация и дисперсия определяются достаточно просто.

$$\begin{aligned} E(Y_t) = 0 \\ Var(Y_t) = \sigma^2\sum_{i=1}^{q}\alpha_i^2 \\ Cov(Y_t, Y_{t+\tau}) = \sigma^2\sum_{i=0}^{q-k}\alpha_i\alpha_i + \tau, \tau = 0, 1, 2...q \end{aligned} \tag{4}$$

2.2. ARIMA и ETS.

Определив авторегрессию и скользящее среднее можно получить обобщенную модель. Такой общий процесс авторегрессии-скользящего среднего называется ARMA(p, q).

$$Y_t = \delta + \alpha_1 Y_{t-1} + ... + \alpha_p Y_{t-p} + \epsilon_t + \beta_1\epsilon_{t-1} + ... + \beta_q\epsilon_{t-q} \tag{5}$$

[2]Волд доказал, что любой недетерменированный случайный слабо станционарный процесс может быть представлен в виде линейной комбинации белых шумов с разными коэффициентами: $Y_t - \mu = \sum_{\tau=0}^{\infty}\alpha_\tau \cdot \epsilon_{t-\tau}$

Запись с помощью оператора лага.

$$\alpha_p(L)X_t = \beta_q(L)\epsilon_t \tag{6}$$

Введем условия станционарности ARIMA. Так, в случае если все корни полинома $\alpha_p(L)$ по модулю меньше единицы то существует обратный оператор, с помощью которого становится возможным аписать временной ряд как $X_t = [\alpha_p(L)]^{-1}\beta_q(L)\epsilon_t$

Обратный оператор возможно представить как сумму простых дробей, каждая из которых эквивалентна бесконечной убывающей геометрической прогрессии с операторными коэффициентами. Таким образом, имеется дело с бесконечным операторным полиномом. Если полином умножить на конечный полином, то результат будет также бесконечным полиномом. Это выражение имеет смысл только в том случае, если все корни характеристического полинома по модулю меньше единицы. Тогда это выражение является разложением Волда и процесс стационарен. Следовательно, стационарность процесса ARMA зависит только от его AR-компоненты. А значит, что процесс ARMA стационарен, если корни характеристического уравнения AR части по модулю меньше единицы.

Для получения процесса ARIMA необходимо ввести понятие приращения или первую разность (first difference). Так, уравнение $Y_t - Y_{t-1} = \epsilon_t$ можно переписать в виде $\Delta Y_t = \epsilon_t$. Такая операция способна привести нестанционарный ряд к станционарному так как степень полинома тренда при каждом взятии первой разности понижается, усиливая влияние скользящего среднего на процесс [1].

В работе Бокса и Дженкинса [2] предложен класс нестанционарных временных рядов, которые с помощью взятия последовательных первых разностей ряд приводится к станционарному виду ARMA. Так, ARIMA(p, d, q) означает, что исходный ряд с помощью d последовательных разностей приведен в станционарный вид ARMA(p, q) и записывается в операторном виде следующим уравнением.

$$\alpha_p(L)\Delta^d x_t = \beta_q(L)\epsilon_t \tag{7}$$

ETS же использует совершенно иной подход, учитывая что более новые поступаемые данные являются более важными для предсказания.

Метод экспоненциального сглаживания объединяет компоненты тренда, сезонности и остатков в расчете сглаживания. Каждый член может быть объединен либо аддитивно, либо мультипликативно, либо не учитываться в модели вовсе. Эти три термина (остаток, тренд и сезонность) называются ETS[3]. Так, в случае модели с аддитивным остатком, без тренда и без сезонности используется обознаяение ETS(A, N, N), ETS(A, A, N) - Линейный метод

Хольтца с аддитивным остатком, а ETS(M, A, N) - линейный метод Холтца с мультипликативным остатком.

Для определения модели ETS сезонность, тренд и остаток получаются благодаря декомпозиции временного ряда с помощью LOESS, или же STL[4].

Рассмотрим различные методы для определения типа поведения компонентов временного ряда: аддитивного или мультипликативного. В целом, аддитивные компоненты имеют линейную или неизменную динамику. Если компонент тренда линейно возрастает или убывает, то он аддитивный. Если же тренд демонстрирует экспоненциальное изменение, положительное или отрицательное, то он мультипликативный. Сезонный компонент подчиняется подобной логике, но для дисперсии значений. Если амплитуда сезонных колебаний не меняется, то она аддитивная. В обратном случае, если амплитуда зависит от времени, она мультипликативная. То же касается и компонента остатков - если амплитуда варьируется во времени, то также используется мультипликативное поведение.

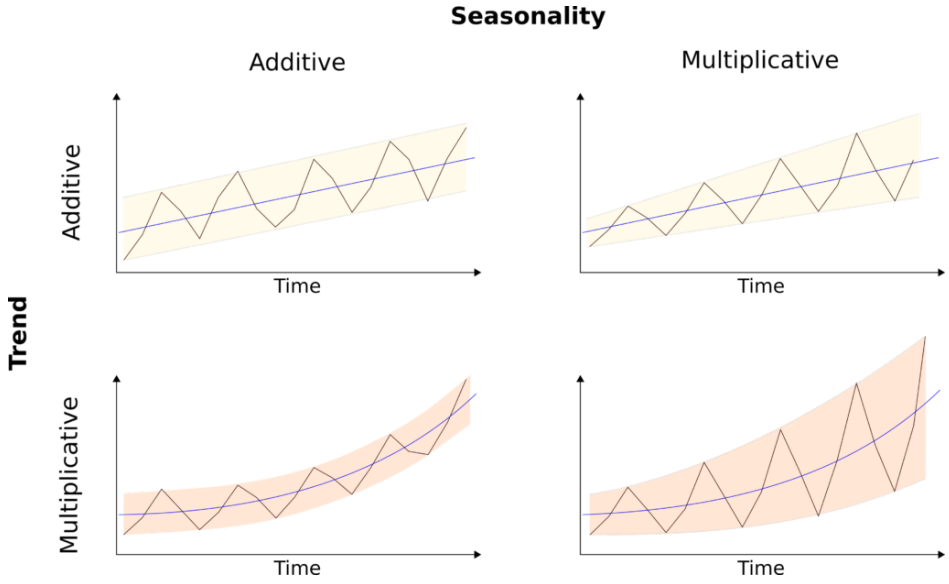

Рис. 2.1. Различные поведения компонент.

Рассмотрим простейшую аддитивную остаточную модель ETS(A, N, N). Тогда данное уравнение Холтца имеет вид

$$\begin{aligned} Y_t &= Y_{t-1} + e_t \\ l_t &= l_{t-1} + \alpha e^t \end{aligned} \tag{8}$$

Где Y_t - предсказание основанное на предыдущем шаге, l_t - экспоненциальная сглаживающая модель. Второе уравнение сглаживания рассчитывает следующий уровень как средневзвешенное значение предыдущего уровня и предыдущего наблюдения.

Данный метод экспоненциального сглаживания может быть модифицирован для учи-

тывания в модели тренда и сезонности. В аддитивном методе Холта-Уинтерса сезонный компонент добавляется к остальным. Эта модель соответствует модели ETS(A, A, A) и имеет следующую формулировку пространства состояний:

$$\begin{aligned} Y_t &= l_{t-1} + b_{t-1} + s_{t-1} + e_t \\ l_t &= l_{t-1} + b_{t-1} + \alpha e_t \\ b_t &= b_{t-1} + \beta e_t \\ s_t &= s_{t-1} + \gamma e_t \end{aligned} \tag{9}$$

2.3. Рекурентные нейронные сети.

Рекуррентная нейронная сеть - это нейронная сеть, в которой активность нейрона зависит от активности других нейронов на предыдущих временных шагах в дополнение к его текущему входу. Это позволяет рекуррентной нейронной сети сохранять состояние во времени, что может быть использовано для обработки данных временных рядов, таких как цены на акции или погода. Эта способность делает рекуррентные нейронные сети особенно полезными для задач, в которых существует основная временная структура. Рекуррентные нейронные сети могут применяться для решения широкого круга задач и зарекомендовали себя как эффективный инструмент машинного обучения и искусственного интеллекта.

Это делает РНС полезными для анализа временных рядов данных, поскольку они могут учиться на исторических данных для создания прогнозов на будущее. Сочетание временных рядов и RNN может дать аналитикам более точные прогнозы событий, обеспечивая мощный инструмент для принятия решений и оценки рисков. Базовую модель нейронной сети можно получить с помощью общего нелинейного неоднородного дифференциального уравнение первого порядка.

$$\frac{d\vec{s}(t)}{dt} = \vec{f}(t) + \vec{\phi} \tag{10}$$

Где $\vec{s}$ - значение d-мерного вектора сигналов состояния, $\vec{\phi}$ - постоянный d-мерный вектор (bias), а $\vec{f}(t)$ - "Аддитивная модель" в научной литературе по динамике мозга[5], определяемая как следующее уравнение.

$$\vec{f}(t) = \vec{a}(t) + \vec{b}(t) + \vec{c}(t) \tag{11}$$

В данном уравнении $\vec{a}(t), \vec{b}(t)$ и $\vec{c}(t)$ определяются следующим образом:

$$\vec{a}(t) = \sum_{k=0}^{K_s-1} \vec{a}_k(\vec{s}(t - \tau_s(k))) \tag{12}$$

$$\vec{b}(t) = \sum_{k=0}^{K_r-1} \vec{b}_k(\vec{r}(t - \tau_r(k))) \tag{13}$$

$$\vec{c}(t) = \sum_{k=0}^{K_x-1} \vec{c}_k(\vec{x}(t - \tau_x(k))) \tag{14}$$

$$\vec{r}(t - \tau_r(k)) = G(\vec{s}(t - \tau_s(k))) \tag{15}$$

Где $\vec{a}(t)$ - "аналоговый" компонент динамической системы, представляет из себя набор смещенных по времени функций, $\vec{b}$ - компоненты $a(t)$ с примененной на них заданной функцией активации G, $\vec{c}$ - внешний вход, смещенный на константу. Задержка по времени необходима для предоставления данной динамичесской системе "памяти". Тогда подставив уравнения в исходную систему и учитывая, что a, b и c являются линейными функциями векторов, мы получим новое уравнение:

$$\frac{d\vec{s}(t)}{dt} = \sum_{k=0}^{K_s-1} A_k(\vec{s}(t - \tau_s(k))) + \sum_{k=0}^{K_r-1} B_k(\vec{r}(t - \tau_r(k))) + \sum_{k=0}^{K_x-1} C_k(\vec{x}(t - \tau_x(k))) \tag{16}$$

Упрощенная и дискретизированная форма аддитивной модели сыграла ключевую роль в связывании нелинейных динамических систем, управляющих морфогенезом, одним из фундаментальных аспектов биологии развития, с обобщенной версией сети Хопфилда, и применении ее к инженерной проблеме обработки изображений.

После заданных упрощений [6] данное нелинейное диффуренциальное уравнение решается и приходит к следующей системе:

$$\begin{cases} \vec{s} = W_s\vec{s}[n+1] + W_r\vec{r}[n-1] + W_x\vec{x}[n] + \vec{\theta}_s \\ \vec{r} = G(\vec{s}[n]) \end{cases} \tag{17}$$

Где W_r и W_x - внешние и скрытые весовые матрицы, $\vec{\theta}_s$ - константый вектор.

Данную систему можно представить в виде следующей схемы

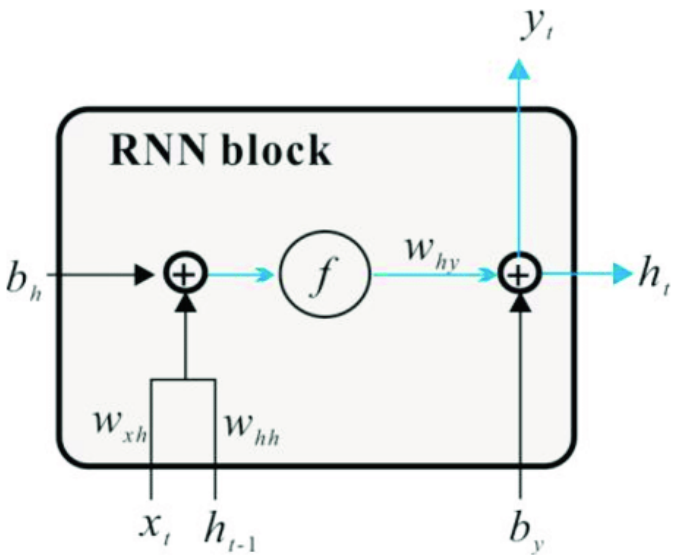

Рис. 2.2. Блок RNN.

Главное отличие нейрона РНС от нейрона обычной глубинной сети состоит в том, что помимо обычных скрытых весов между нейронами также существуют и внешние веса, которые подаются остальным весам со сдвигом времени τ, минуя функцию активации G. Именно благодаря новым внешним весам РНС способна более быстро обнаружить паттерны в информации и обучиться на них эффективнее.

Однако, несмотря на свою более быструю сходимость, системы RNN являются проблематичными на практике - во время обучения они страдают от хорошо документированных проблем, известных как "исчезающие градиенты" и "взрывающиеся градиенты".

Эти трудности становятся очевидными, когда зависимости в целевой подпоследовательности охватывают большое количество выборок, что требует, чтобы окно модели развёрнутой РНС было соразмерно широким, чтобы уловить эти дальние зависимости. Для решения данной проблемы в РНС вводится новая функция акцивации состоящего из аффинного преобразования с последующим простой поэлементной нелинейности с использованием управляющего рекурентного блока[7].

Один из таких блоков называется "долгая краткосрочная память" (LSTM). Так, если отдельный рекурентный блок вычисляет только взвешанную сумму входного сигнала и применяет нелинейную функцию, то каждый j-тый $LSTM$ блок также содержит ячейку памяти c_t^j в момент времени t.Тогда функция активации считается как

$$h_t^j = \sigma_t^j tanh(c_t^j) \tag{18}$$

Где σ_t^j - выходной управляющий блок, который определяет объем содержания памяти.

$$\sigma_t^j = \sigma(W_o x_t + U_o h_{t-1} + V_o c_t)^j \tag{19}$$

Сам же рекурентный управляющий блок при каждом обновлении частично забывает и запомниает новую информацию $\tilde{c}_t^j = tanh(W_c x_t + U_c h_{t-1}^j$.

$$c_t^j = f_t^j c_{t-1}^j + i_t^j \tilde{c}_t^j \tag{20}$$

Где f_t^j - управляющий блок "забывания", i_t^j - "запоминающий" управляющий блок. Именно эти два управлябщих блока определяют количество забывающейся и запоминающейся информации.

$$\begin{aligned} f_t^j &= \sigma(W_f x_t + U_f h_{t-1} + V_f c_{t-1})^j \\ i_t^j &= \sigma(W_i x_t + U_i h_{t-1} + V_i c_{t-1})^j \end{aligned} \tag{21}$$

В отличие от традиционного рекуррентного блока, который перезаписывает свое содержимое на каждом временном шаге, блок LSTM способен решать, сохранять ли существующую память с помощью введенных управляющих блоков. Интуитивно понятно, что если блок LSTM обнаруживает важную особенность во входной последовательности на ранней стадии, он легче переносит эту информацию (о существовании признака) на большое расстояние, таким образом, улавливая больше потенциальных зависимостей. Именно такой подход позволяет LSTM справляться с взрывающимеся и затухающими градиентами.

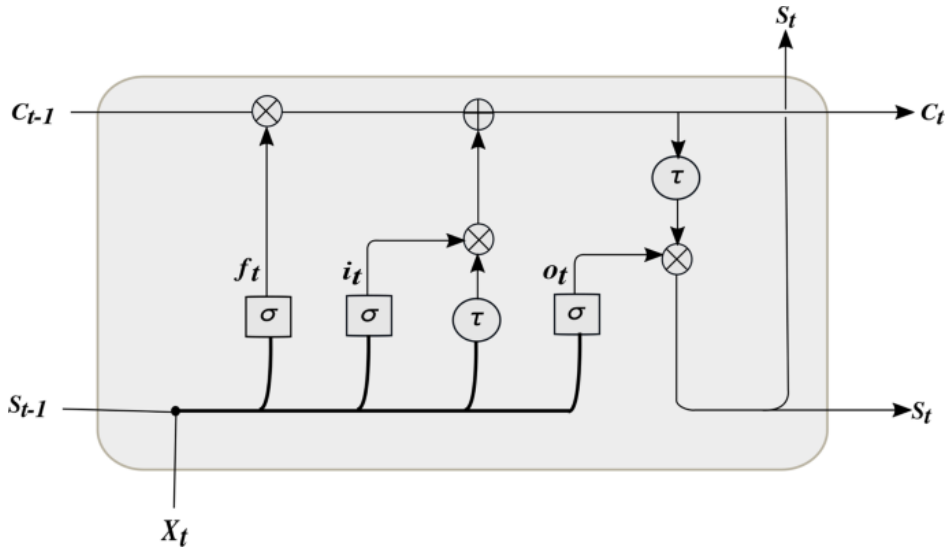

Рис. 2.3. Блок LSTM

Помимо LSTM подхода также существует механизм управляемой рекурентной сети (GRU),

который позволяет каждому рекурентному блоку адаптивно улавливать новые признаки на раличных временных промежутках. GRU имеет схожее строение с LSTM, однако не имеет отдельных ячеек памяти c_t^j. Забывающий и запоминабщий управляющие блоки меняются одним управляющим блоком.

$$z_t^j = \sigma(W_z x_t + U_z h_{t-1})^j \tag{22}$$

Тогда функция активации вычисляется как

$$h_t^j = (1 - z_t^j) h_{t-1}^j + z_t^j \tilde{h_t^j} \tag{23}$$

Где $\tilde{h}_t^j = tanh(Wx_t + U(r_t \cdot h_{t-1}))^j$ - кандидат активации, а $r_t^j = \sigma(W_r x_t + U_r h_{t-1})^j$ - блок стирания.

Главное преимущество GRU перед LSTM в более быстрой и стабильной сходимости, однако некоторая информация может теряться в процессе обучения.

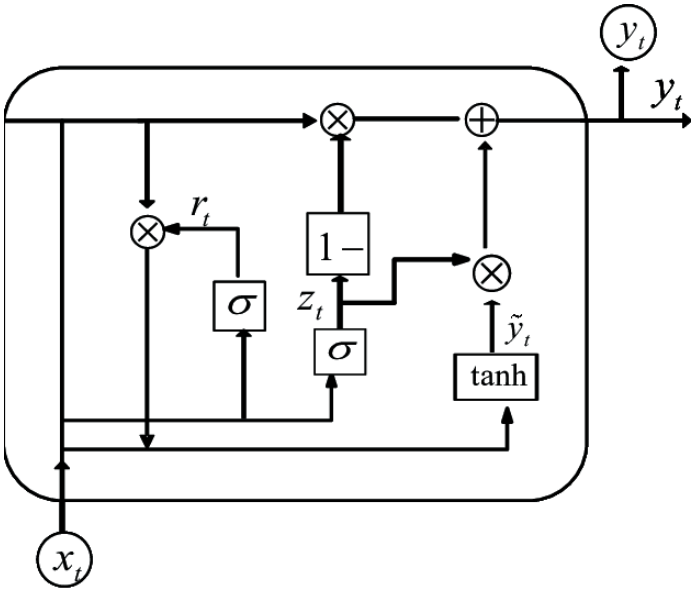

Рис. 2.4. Блок GRU.

Однако даже не смотря на данные нововведения в рекурентную модель, во время обучения нейронной сети на большом количестве слоев может произойти взрыв градиента. Для избежания этого вводятся дополнительные пространственные кратчайшие путь от одних слоев к другим для эффективного обучения глубоких сетей с несколькими слоями, така НС называется остаточной[8]. Остаточная сеть обеспечивает отображение идентичности через кратчайшие пути. Поскольку отображение идентичности всегда существует, функция на выходе должна изучать только остаточное отображение. Формулировка этого отношения

может быть выражена как:

$$\hat{y} = G(\vec{x}, W_x) + \vec{x} \tag{24}$$

Тогда схематически остаточные нейронные сети можно представить следующим образом.

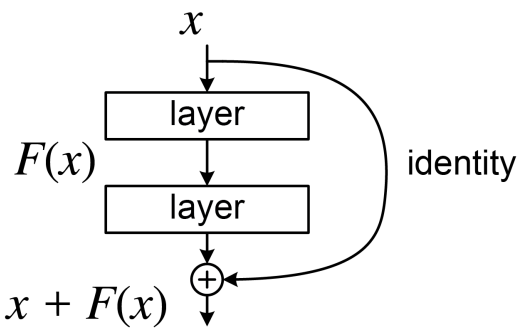

Рис. 2.5. Слои остаточной нейронной сети

Несмотря на свою простоту, такой подход позволяет делать сети более глубокими без потери качества обучения[8].

Помимо использования одной нейронной сети для прогноза временного ряда возможно использовать сразу несколько для более точного предсказания, то есть использовать ансамбль нейронных сетей.

Одним из таких подходов является блендинг.

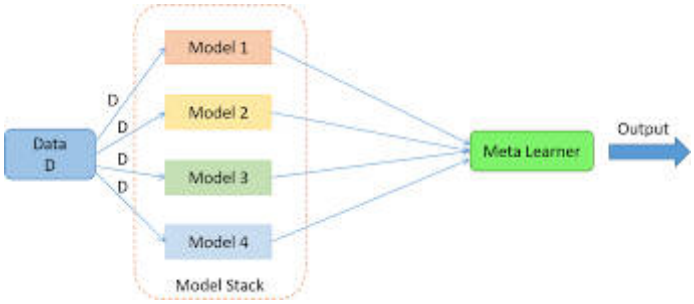

Рис. 2.6. Схематическая модель блендинга с использованием метапризнаков на обученных моделях.

Идея состоит в том, чтобы использовать уже обученные нейронные сети и использовать их прогнозы как новые признаки (метапризнак). Далее с помощью полученных метапризнаков можно обучить метамодель, которая Обучится на полученных метапризнаках и даст новое предсказание.

Глава 3. Практическая часть.

3.1. Наборы данных

В данной работе использовались следующие датасеты:

1. Прогнозирование денежных доходов населения, ВШЭ. Наблюдается явный тренд в виде роста и сезонность, временной ряд не станционарен.

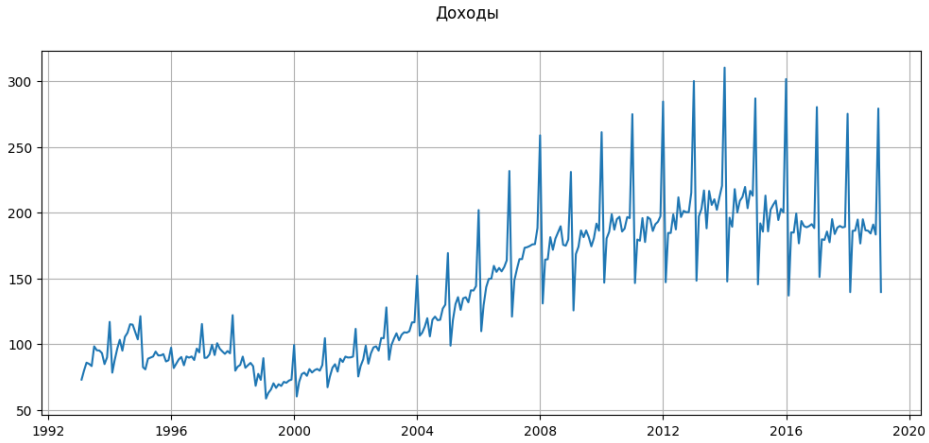

Рис. 3.1. Временной ряд доходов населения

2. Временной ряд цены золота, данные Бундесбанка. Наблюдается тренд, ряд не станционарен.

Рис. 3.2. Временной ряд цены золота.

3. Аномальные температуры, NASA. Наблюдается явный тренд и сезонность, ряд не станционарен.

Рис. 3.3. Временной ряд измерений аномальной температуры NASA

Прежде чем строить прогнозы на нейронных сетях, каждый временной ряд был спрогнозирован с помощью ARIMA/ETS, а затем оценены с помощью метрик MAE и RMSE.

3.2. Построение рекурентных моделей с помощью PyTorch

Для построения моделей рекурентной нейронной сети используется библиотека PyTorch. Все стадии обучения были произведены в Google Colab.

Прежде чем начать обучение РНС на данных были написаны вспомогательные классы обучения. Например, для автоматизации обучения различных моделей был написан общий класс Trainer, который позволяет также автоматически сохранять веса при каждой эпохе обучения. Полный код представлен в ноутбуках google colab.

Листинг 1. Вспомогательный класс для автоматизации обучения нейронной сети. На вход принимает непосредственно модель, функцию потерь и оптимизатор.

```
class Trainer:
    def __init__(self, model, loss_fn, optimizer):
        self.model = model
        ...
    def train_step(self, x, y):
        self.model.train()
        yhat = self.model(x)
        loss = self.loss_fn(y, yhat)
        loss.backward()
        self.optimizer.step()
        self.optimizer.zero_grad()
        return loss.item()
    def evaluate(self, test_loader, batch_size=1, n_features=1):
      with torch.no_grad():
          predictions = []
          values = []
          for x_test, y_test in test_loader:
              x_test = x_test.view([batch_size, -1, n_features]).
  to(device)
              y_test = y_test.to(device)
              self.model.eval()
              yhat = self.model(x_test)
              predictions.append(yhat.detach().cpu().numpy())
              values.append(y_test.detach().cpu().numpy())

      return predictions, values
    def plot_losses(self):
        plt.plot(self.train_losses, label="Training loss")
        plt.plot(self.val_losses, label="Validation loss")
        plt.legend()
        ...
```

Также напишем непосредственно сами моделей РНС с помощтю готовых управляющих блоков PyTorch. Например, стандартную сеть RNN можно реализовать следующим образом.

Листинг 2. Слои RNN с помощью модулей PyTorch.

```
class RNNModel(nn.Module):
    """
    Classic RNN model
    """
    def __init__(self, input_dim, hidden_dim, layer_dim, output_dim, dropout_prob):
        super(RNNModel, self).__init__()
        self.hidden_dim = hidden_dim
        self.layer_dim = layer_dim

        # RNN layers
        self.rnn = nn.RNN(
            input_dim, hidden_dim, layer_dim, batch_first=True, dropout=dropout_prob
        )
        # Fully connected layer
        self.fc = nn.Linear(hidden_dim, output_dim)

    def forward(self, x):
        h0 = torch.zeros(self.layer_dim, x.size(0), self.hidden_dim).to(device)
        out, h0 = self.rnn(x, h0.detach())
        out = out[:, -1, :]
        out = self.fc(out)
        return out
```

Также были написаны функции для нормализации данных, создания датасетов для обучения с помощью moving window и отдельный класс для Residual и blending моделей, для некоторых временных рядов был использован grid search для нахождения оптимальных параметров. Весь исходный код доступен в прилагающихся ноутбуках jupyter.

После обучения различных моделей, в том числе используя residual модели и блендинг, оценки MAE оказались следующими:

1. Наилучший результат для данных доходов населения оказался у модели LSTM и GRU, причем хуже всего модель обучилась на статистическом методе ARIMA.

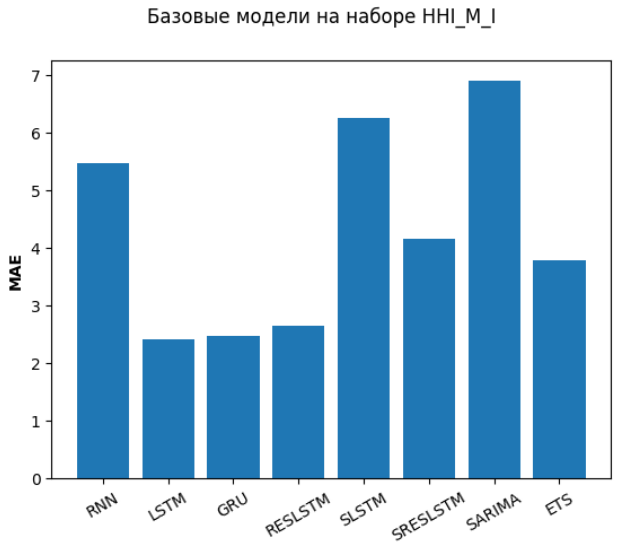

Рис. 3.4. MAE моделей для прогноза доходов населения

2. Наилучший результат для данных цен на золото оказался у модели ETS и LSTM, причем хуже всего модель обучилась на статистическом методе ARIMA и блендинге ResLSTM.

Рис. 3.5. MAE моделей для прогноза индекса цен на золото

3. Наилучший результат для данных цен на золото оказался у модели ResLSTM и блендинга, причем хуже всего модель обучилась на статистическом методе ARIMA и стандартной модели RNN.

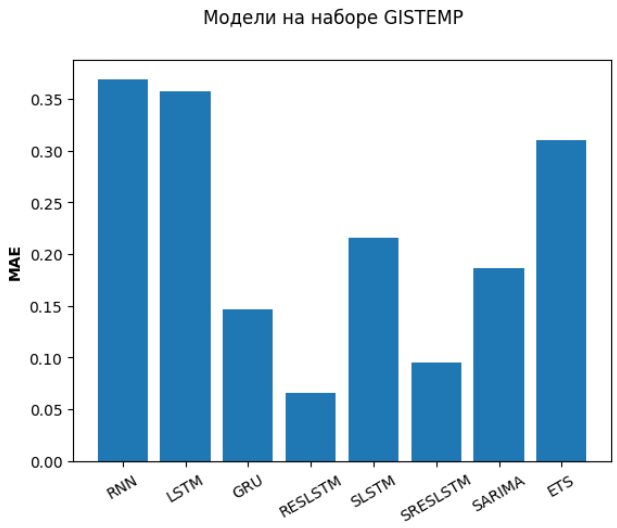

Рис. 3.6. MAE моделей для прогноза аномальных температур

Глава 4. Заключение

В данной работе различные временные ряды были спрогнозированны с помощью классических общепринятых статистических методов (ARIMA, ETS) и различных моделей рекурентных нейронных сетей. Оценивания результатов были произведены с помощью метрики MAE, и нейронные сети показали очень хороший результат, лишь в одном опыте статистический прогноз ETS превзошел прогноз от нейронной сети. Причем данный результат можно улучшить. Так, помимо блендинг-ансамбля можно использовать беггинг ResLSTM. Именно этот подход был использован в работе Slawek Smyl для соревнования по предсказанию временных рядов. Данный подход позволил рекурентным нейронным сетям впервые превзойти[9] статистические методы, которые до сих пор повсеместно используются для прогноза временных рядов.

Учитывая полученный опыт, в выпускной квалификационной работе будет произведен прогноз различных временных рядов из датасета аномальных наблюдений датчиков космических аппаратов WebTCAD: показания температуры баттареи, сила сигнала, общая температура спутника и т.д. В случае успешного прогноза таких временных рядов станет возможным использовать полученные модели для предотвращения аварий космических аппаратов и для улучшения связи со спутником.

Список литературы

1. Канторович Г. Г. Анализ временных рядов // Экономический журнал ВШЭ. — 2002. — Т. 6, № 1. — С. 85–116. — Режим доступа: https://ej.hse.ru/2002-6-1/26549758.html.

2. Box George E. P. Time series analysis; forecasting and control. — Сан-Франциско : Holden-Day, 1970. — Т. 1.

3. Makridakis Spyros, Hibon Michèle. Exponential smoothing: The effect of initial values and loss functions on post-sample forecasting accuracy // International Journal of Forecasting. — 1991. — Т. 7. — С. 317–330.

4. Cleveland Robert B; Cleveland William S; Terpenning Irma. STL: A Seasonal-Trend Decomposition Procedure Based on Loess // Journal of Official Statistics. — 1990. — Т. 6. — С. 3–73.

5. S. Saeid J.A. Chambers. EEG Signal Processing. — Нью-Джерси : John Wiley and Sons Ltd, 2007.

6. Sherstinsky Alex. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network // Physica D: Nonlinear Phenomena. — 2020. — Т. 404.

7. Chung Junyoung. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling // NIPS 2014 Workshop on Deep Learning. — Нью-Йорк : Нью-Йоркский университет, 2014.

8. Kim Jaeyoung. Residual LSTM: Design of a Deep Recurrent Architecture for Distant Speech Recognition // Interspeech 2017. — Дублин : INTERNATIONAL SPEECH COMMUNICATION ASSOCIATION., 2017. — Режим доступа: https://arxiv.org/pdf/1701.03360.pdf.

9. Smyl Slawek. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting // International Journal of Forecasting. — 2020. — Т. 36. — С. 75–85.

10. Hansika Hewamalage Christoph Bergmeir Kasun Bandara. Recurrent Neural Networks for Time Series Forecasting: Current Status and Future Directions // International Journal of Forecasting. — 2021. — Т. 37. — С. 388–427.