Информационные процессы, Том 21, № 1, 2021, стр. 50–64

МАТЕМАТИЧЕСКИЕ МОДЕЛИ, ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ

Онлайн обучение многокомпонентных прогнозирующих систем

В. В. Вьюгин, В. К. Калмыков, В. Г. Трунов

Институт проблем передачи информации им. А.А.Харкевича,
Российская академия наук, Москва, Россия
e-mail: vyugin@iitp.ru

Поступила в редколлегию 15.03.2021

Аннотация—Рассматривается подход, при котором обучение прогнозирующей системы производится во все моменты времени. Этот подход называется в современной литературе Lifelong Machine Learning. В качестве примера применения такого подхода решается задача прогнозирования потока данных, поступающих в режиме онлайн. При вычислении очередного прогноза не используется никаких предположений о природе источника, генерирующего поток данных – источник может быть алгоритмическим или вероятностным, смена его параметров может происходить в случайные моменты времени. В этих условиях прогнозирующие системы должны обучаться непрерывно по мере поступления входной информации. Предлагается два вида адаптивных алгоритмов машинного обучения. Первый алгоритм производит агрегацию в режиме онлайн локальных моделей потока данных, которые автоматически создаются в процессе его изучения. Вторая группа методов основана на применении искусственных нейронных сетей, обучающихся в режиме онлайн. Проводится сравнительный анализ эффективности предложенных методов.

КЛЮЧЕВЫЕ СЛОВА: непрерывное машинное обучение, Lifelong Machine Learning, адаптивные алгоритмы прогнозирования в режиме онлайн, предсказания с использованием экспертных стратегий, регрет, агрегирующий алгоритм, искусственные нейронные сети.

1. ВВЕДЕНИЕ

В работе решаются задачи машинного обучения в режиме онлайн. Рассматривается подход, при котором обучение и построение прогнозирующей системы производится во все моменты времени. Этот подход называется в современной литературе Lifelong Machine Learning [1].

Рассматривается задача обучения с учителем, при которой в каждый момент времени t некоторая прогнозирующая система, используя наблюдаемые прошлые значения (исходы) $(x_1, y_1), \ldots, (x_{t-1}, y_{t-1})$ и текущий объект x_t, выдает прогноз γ_t неизвестного будущего значения y_t.

После того как истинное значение y_t будет предъявлено, вычисляются потери $\lambda(\gamma_t, y_t)$ для этого предсказания, где λ есть некоторая функция потерь, например, $\lambda(\gamma_t, y_t) = (\gamma_t - y_t)^2$. Кумулятивные потери за первые t шагов равны сумме потерь на этих шагах: $\sum_{s=1}^{t} \lambda(\gamma_s, y_s)$.

Под прогнозирующей системой мы понимаем функцию, зависящую от набора параметров, которая на каждом шаге t при данном объекте x_t выдает прогноз γ_t.

Мы будем рассматривать следующие прогнозирующие системы: регрессоры – линейные функции

$$f(\mathbf{x}_t) = (\mathbf{w}_t \cdot \mathbf{x}_t)$$

и их комбинации, где $\mathbf{w}_t, \mathbf{x}_t \in \mathcal{R}^n$, а также функции, задаваемые искусственными нейронными сетями различной архитектуры.

Источник, порождающий данные, может быть вероятностным или алгоритмическим или даже “адаптивно враждебным” по отношению к прогнозирующей системе. Природа источника, порождающего реальные исходы, может быть неизвестна предсказателю и не будет использоваться при построении и корректировке прогнозирующей системы.

В разделе 2 приводятся основные понятия теории предсказаний с использованием экспертных стратегий, представлены некоторые детали агрегирующего алгоритма Вовка [2], [3].

В разделе 3 будут рассматриваться линейные прогнозирующие системы, построенные по фрагментам временного ряда, а также их комбинации. Будет рассматриваться постановка, при которой несколько конкурирующих методов (линейных функций) дают прогнозы в режиме онлайн. Эти прогнозы могут привести к большим или меньшим потерям. Задача агрегирующего алгоритма состоит в том, чтобы объединить эти прогнозы в один агрегированный прогноз таким образом, чтобы минимизировать потери.

Задача агрегации прогнозов решается в рамках теории предсказаний с использованием экспертных стратегий (Prediction with Expert Advice – PEA). Каждая экспертная стратегия использует свою специфическую предсказательную модель, прогнозирующую наблюдаемые данные. Экспертные стратегии автоматически строятся в режиме онлайн в зависимости от реальных данных. На каждом шаге определяется новая экспертная прогнозирующая стратегия, выражающая локальные свойства временного ряда, которая будет использоваться на всех шагах в дальнейшем. В каждый момент времени прогнозы всех построенных к этому моменту локальных прогнозирующих стратегий объединяются с помощью одного из методов агрегации.

Факторы, влияющие на релевантность экспертных моделей, могут быть скрыты от предсказателей, поэтому задача предсказания решается в рамках теоретико-игрового подхода, при котором не используется никаких предположений о том, какая из возможных моделей релевантна в данный момент времени.

Агрегирующий алгоритм в режиме онлайн объединяет прогнозы экспертных моделей в зависимости от их потерь в прошлом. Разность между кумулятивными потерями агрегирующего алгоритма и кумулятивными потерями произвольного экспертного алгоритма, накопленными за весь период прогнозирования, называется регретом. Цель агрегирующего алгоритма заключается в том, чтобы минимизировать регрет относительно любой экспертной стратегии.

В работах [4], [5], [2], [6] были развиты методы агрегирования в режиме онлайн точечных прогнозов. В этом случае эксперты выдают вещественные прогнозы в виде чисел или векторов, агрегирующий алгоритм выдает прогноз такого же вида. В разделе 3 представлен метод вычисления агрегированного предсказания на основе прогнозов, представленных экспертами, в частности, представлены формулы для прямого расчета агрегированного прогноза на основе прогнозов, представленных экспертами.

В этой работе мы получаем теоретическую верхнюю границу регрета для специального случая, когда число экспертов с течением времени неограниченно возрастает.

В разделе 4 та же самая задача прогнозирования решается с помощью искусственных нейронных сетей. В этом разделе представлены различные методы прогнозирования с помощью искусственных нейронных сетей.

Будет рассматриваться постановка в стиле онлайн выпуклой оптимизации для нейронных сетей. Прогноз будет производиться с помощью нейронных сетей различной архитектуры. На каждом шаге параметры нейронной сети будут оптимизироваться методом онлайн градиентного спуска. На каждом шаге нейронная сеть дообучается на основе определенной информации о прошлом изучаемых данных.

Сравнительный анализ эффективности предложенных методов проводится в разделе 4.2. Проведенные численные эксперименты показывают, что оба подхода приводят к схожей средней точности прогнозов, при этом, метод локальных экспертов дает большую точность в том случае, когда информация, предоставляемая локальным экспертам и нейронным сетям для обучения, ограничена.

2. ОСНОВНЫЕ ПОНЯТИЯ И УТВЕРЖДЕНИЯ

В этом разделе даются необходимые определения и вспомогательные результаты теории предсказаний с использованием экспертных стратегий.

2.1. Предсказания с использованием экспертных стратегий

В этом разделе мы напомним основные идеи теории предсказаний с использованием экспертных стратегий. Пусть Ω – множество исходов и Γ – множество предсказаний, $\lambda(\gamma, y)$ – функция потерь, где $\gamma \in \Gamma$ и $y \in \Omega$.

Будет также использоваться дополнительная или входная информация (сигналы). Пусть $X \subseteq \mathcal{R}^n$ – множество сигналов. Предполагаем, что исходы $y_1, y_2, \dots \in \Omega$ и сигналы $\mathbf{x}_1, \mathbf{x}_2, \dots \in X$ появляются постепенно по шагам – в режиме онлайн.

Общая схема предсказания с использованием экспертных стратегий указана в следующем протоколе.

Алгоритм 1

FOR $t = 1, \dots, T$

1. Получаем входную информацию (сигнал) $\mathbf{x}_t$.
2. Вычисляем предсказания $f_{i,t}$ экспертов $i \in \mathcal{N}$.
3. Вычисляем предсказание γ_t агрегирующего алгоритма.
4. Получаем исход y_t и вычисляем потери $l_{i,t} = \lambda(f_{i,t}, y_t)$ экспертов и потери $h_t = \lambda(\gamma_t, y_t)$ алгоритма.

ENDFOR

В дальнейшем, эксперты также будут строиться по шагам в режиме онлайн в зависимости от наблюдаемых данных и сигналов. Каждый эксперт i определяется (инициализируется) в момент времени i и отождествляется с прогнозирующей стратегией – функцией $f_i(\mathbf{x})$. В дальнейшем, на шагах $t > i$ этот эксперт вычисляет свой прогноз с помощью этой функции в зависимости от текущего сигнала $f_{i,t} = f_i(\mathbf{x}_t)$.

Для удобства предполагаем, что на каждом шаге t имеется бесконечное множество экспертов $\mathcal{N} = \{1, 2, \dots\}$, при этом, стратегии $f_i(\cdot)$ экспертов $1 \le i \le t$ уже инициализированы, а стратегии экспертов $i > t$ временно являются виртуальными.[1]

Процесс обучения идет по шагам $t = 1, 2, \dots,$. В каждый момент времени t эксперты $i \in \mathcal{N}$ наблюдают сигнал $\mathbf{x}_t$ и представляют свои предсказания $f_{i,t} = f_i(\mathbf{x}_t)$ при $i \le t$. Агрегирующий алгоритм вычисляет свое предсказание γ_t. При $i > t$ в качестве предсказания эксперта i искусственным образом используем предсказание γ_t агрегирующего алгоритма.

После того, как все предсказания будут сделаны, предъявляется значение y_t и эксперты и агрегирующий алгоритм несут потери – $l_{i,t} = \lambda(f_{i,t}, y_t)$, при $i \in \mathcal{N}$, и $h_t = \lambda(\gamma_t, y_t)$.

[1] Эти стратегии не используются при вычислении агрегированного предсказания.

Кумулятивные потери $L_{i,T}$ каждого эксперта i и потери H_T агрегирующего алгоритма, которые они несут за первые T шагов, определяются в виде

$$L_{i,T} = \sum_{t=1}^{T} l_{i,t} \quad \text{и} \quad H_T = \sum_{t=1}^{T} h_t,$$

соответственно. Эффективность агрегирующего алгоритма относительно эксперта i измеряется с помощью кумулятивного регрета $R_{i,T} = H_T - L_{i,T}$.

Задача агрегирования прогнозов экспертов заключается в том, чтобы минимизировать регрет $R_{i,t}$ относительно каждого эксперта i. Для того, чтобы достичь этой цели, агрегирующий алгоритм на каждом шаге t оценивает эффективность экспертов, приписывая им веса $\mathbf{w}_t = \{w_{i,t} : i \in \mathcal{N}\}$, где $w_{i,t} \geq 0$ для всех i и t. Например, можно определить начальные значения этих весов на первом шаге $w_{i,1} = \frac{1}{i(i+1)}$ при $i \in \mathcal{N}$, а затем перестраивать их в зависимости от потерь экспертов с помощью метода экспоненциального взвешивания (см. [5], [2]):

$$w_{i,t+1} = w_{i,t} e^{-\eta l_{i,t}} \tag{1}$$

для каждого $i \in \mathcal{N}$, где η – параметр обучения. Перед использованием веса нормируются

$$w^*_{i,t} = \frac{w_{i,t}}{\sum\limits_{j \in \mathcal{N}} w_{j,t}}.$$

2.2. Агрегирующие алгоритмы – **AA** *и* **WA**

Агрегирующий алгоритм **AA**, предложенный в работах [2], [3], лежит в основе наших приложений.

Будем использовать смешиваемые функции потерь $\lambda(\gamma, y)$, где γ – прогноз и y – исход. В работе [3] введена функцию суперпредсказания

$$g(y) = -\frac{1}{\eta} \ln \sum_{i \in \mathcal{N}} e^{-\eta \lambda(g_i, y)} p_i,$$

где $\mathbf{p} = (p_i : i \in \mathcal{N})$ – распределение вероятностей на множестве всех экспертов, $\mathbf{g} = (g_i : i \in \mathcal{N})$ – последовательность прогнозов, представленных экспертами.

Функция потерь λ называется η-смешиваемой, если для любого вероятностного распределения $\mathbf{p}$ на множестве всех экспертов и для любой последовательности $\mathbf{g}$ их предсказаний существует предсказание γ такое, что

$$\lambda(\gamma, y) \leq g(y) \tag{2}$$

для всех y.

Частным случаем этого понятия является понятие экспоненциальной вогнутости. Функция потерь $\lambda(\gamma, y)$ является η-экспоненциально вогнутой, если для любого y функция $e^{-\eta\lambda(\gamma,y)}$ является вогнутой по γ. По определению в этом случае неравенство (2) имеет место, т.е. любая η-экспоненциально вогнутая функция является η-смешиваемой.

Фиксируем некоторое правило $\gamma = \mathrm{Subst}(\mathbf{g}, \mathbf{p})$ для вычисления прогноза, удовлетворяющего (2). Функция Subst называется функцией подстановки.

Квадратичная функция потерь $\lambda(\gamma, y) = (y - \gamma)^2$ является η-смешиваемой для любого η такого, что

$$0 < \eta \leq \frac{2}{(b-a)^2},$$

где y и γ – вещественные числа и $y \in [a, b]$ для некоторых $a < b$ (см. [2], [3]). Согласно [3] в этом случае соответствующее предсказание γ может быть вычислено по формуле

$$\gamma = \mathrm{Subst}(\mathbf{g}, \mathbf{p}) = \frac{a+b}{2} + \frac{1}{2\eta(b-a)} \ln \frac{\sum\limits_{i \in \mathcal{N}} p_i e^{-\eta(b-c_i)^2}}{\sum\limits_{i \in \mathcal{N}} p_i e^{-\eta(a-g_i)^2}}. \tag{3}$$

Для η-экспоненциально вогнутой функции потерь для вычисления γ можно использовать более простое выражение

$$\gamma = \mathrm{Subst}(\mathbf{g}, \mathbf{p}) = \sum_{i \in \mathcal{N}} g_i p_i. \tag{4}$$

Квадратичная функция потерь является η-экспоненциально вогнутой при $0 < \eta \leq \frac{1}{2(b-a)^2}$. При этом, определение (4) приводит к оценке регрета, которая в четыре раза хуже, чем определение (3). Неравенство (2) также имеет место для всех y. Проведенные эксперименты подтверждают это различие, см. раздел 4.2.

2.3. Регрет алгоритма **AA**

Оценка регрета для алгоритма **AA** дается в следующем утверждении. (см. [3]).

Предложение 1. *Допустим, что функция потерь $\lambda(f, y)$ является η-смешиваемой. Пусть H_T – кумулятивные потери алгоритма* **AA** *и $L_{i,T}$ – кумулятивные потери эксперта i, $w_{i,1}$ – его начальный вес. Тогда*

$$H_T \leq L_{i,T} + \frac{1}{\eta} \ln \frac{1}{w_{i,1}}$$

для любого T.

Доказательство. Пусть $\mathbf{w}_t^* = (w_{i,t}^* : i \in \mathcal{N})$ – нормированные веса и $\mathbf{g}_t = (f_{i,t} : i \in \mathcal{N})$ – прогнозы всех экспертов на шаге t и $f_t = \mathrm{Subst}(\mathbf{g}_t, \mathbf{w}_t^*)$ – прогноз **AA**.

По свойству смешиваемости

$$h_t = \lambda(f_t, y_t) \leq g_t(y_t) = -\frac{1}{\eta} \ln \sum_{i \in \mathcal{N}} e^{-\eta \lambda(f_t, y_t)} w_{i,t}^* = -\frac{1}{\eta} \ln \frac{W_{t+1}}{W_t} \tag{5}$$

для всех t, где $W_t = \sum\limits_{i \in \mathcal{N}} w_{i,t}$ и $W_1 = 1$. По (1) имеем

$$w_{i,T+1} = w_{i,1} e^{-\eta L_{i,T}}.$$

Складываем неравенство (5) при $1 \leq t \leq T$ и получаем оценку

$$H_T \leq \sum_{t=1}^{T} g_t(y_t) = -\frac{1}{\eta} \ln W_{T+1} \leq L_{i,T} + \frac{1}{\eta} \ln \frac{1}{w_{i,1}}$$

для всех T. □

3. АГРЕГАЦИЯ ЛОКАЛЬНЫХ ЭКСПЕРТОВ

Будем использовать сигналы $\mathbf{x}_t \in \mathcal{R}^n$ для построения прогнозов экспертов. Допустим, что на каждом шаге i вводится в действие (инициализируется) новый эксперт. Этот эксперт представлен прогнозирующей стратегией – функцией $f_i(\cdot)$ от сигнала. Функция f_i может определяться по предыдущим членам временного ряда. Прогноз эксперта i на шаге $t \geq i$ равен $f_{i,t} = f_i(\mathbf{x}_t)$, где $\mathbf{x}_t$ – сигнал на шаге t.

В экспериментах далее эксперт i будет использовать стратегию $f_i(\mathbf{x}) = (\mathbf{w}_i \cdot \mathbf{x})$, где $\mathbf{w}_i \in \mathcal{R}^n$ – весовой вектор, $\mathbf{x} \in \mathcal{R}^n$ – аргумент линейной функции. Весовой вектор $\mathbf{w}_i$ будет определен на шаге i методом гребневой регрессии по скользящему окну $((\mathbf{x}_{i-h}, y_{t-h}), \ldots, (\mathbf{x}_{i-1}, y_{i-1}))$. На каждом шаге $t \geq i$ эксперт i будет выдавать прогноз $f_{i,t} = f_i(\mathbf{x}_t) = (\mathbf{w}_i \cdot \mathbf{x}_t)$.

На шагах $t < i$, когда эксперт i еще не инициализирован, введем для него виртуальные прогнозы – будем предполагать, что его прогноз равен γ_t – прогнозу агрегирующего алгоритма. Такое определение содержит логический круг, так как прогноз γ_t определяется путем агрегации прогнозов всех экспертов, в том числе и эксперта i. Это противоречие будет разрешено с помощью метода неподвижной точки, предложенного в работе [7].

Допустим, что прогноз агрегирующего алгоритма γ_t известен. Определим предсказания всех экспертов $i = 1, 2, \ldots$ на шаге t: $f_{i,t} = f_i(\mathbf{x}_t)$ при $i \leq t$, и $f_{i,t} = \gamma_t$ при $i > t$.

Потери агрегирующего алгоритма равны $h_t = \lambda(\gamma_{t-1}(\mathbf{x}_t), y_t)$, потери экспертов $1 \leq i \leq t-1$ равны $l_{i,t} = \lambda(f_{i,t}, y_t)$, $l_{i,t} = h_t$ при $i > t$.

Наша цель – определить прогноз γ_t так, чтобы

$$e^{-\eta\lambda(\gamma_t, y)} \geq \sum_{i=1}^{\infty} e^{-\eta\lambda(\tilde{f}_{i,t}, y)} w^*_{i,t} \tag{6}$$

для каждого y, где $w^*_{i,t}$ – нормированный вес i-го эксперта. Заменим это условие на эквивалентное ему условие, в котором суммирование производится по конечному множеству экспертов.

Так как $f_{i,t} = f_i(\mathbf{x}_t)$ при $i \leq t$ и $f_{i,t} = \gamma_t$ при $i > t$, перепишем условие (6) на γ_t в более детальной форме:

$$e^{-\eta\lambda(\gamma_t, y)} \geq \sum_{i=1}^{t} w^*_{i,t} e^{-\eta\lambda(f_i, y)} + e^{-\eta\lambda(\gamma_t, y)} \left(1 - \sum_{i=1}^{t} w^*_{i,t}\right). \tag{7}$$

Таким образом, неравенство (6) эквивалентно неравенству

$$e^{-\eta\lambda(\gamma_t, y)} \geq \sum_{i=1}^{t} w^p_{i,t} e^{-\eta\lambda(f_i, y)}, \tag{8}$$

где

$$w^p_{i,t} = \frac{w_{i,t}}{\sum_{j=1}^{t} w_{j,t}}. \tag{9}$$

Согласно правилу вычисления прогноза для **AA**, можно определить

$$\gamma_t = \mathrm{Subst}(\mathbf{g}_t, \mathbf{w}^p_t).$$

Здесь Subst – функция подстановки,

$$\mathbf{w}^p_t = (w^p_{1,t}, \ldots, w^p_{t,t}) \quad \text{и} \quad \mathbf{g}_t(s) = (f_1(\mathbf{x}_t), \ldots, f_t(\mathbf{x}_t)).$$

Запишем алгоритм в виде протокола.

Алгоритм 2

Определим начальные веса $w_{i,1} = \frac{1}{i(i+1)}$ при $i = 1, 2, \ldots$.

FOR $t = 1, \ldots, T$

1. Инициализируем эксперта $f_t(\cdot)$. Эксперты $f_1(\cdot), \ldots, f_{t-1}(\cdot)$ были инициализированы на предыдущих шагах.

3. Получаем сигнал $\mathbf{x}_t$.

4. Вычисляем прогнозы экспертов $f_{i,t} = f_i(\mathbf{x}_t)$ при $1 \le i \le t$.

5. Вычисляем вспомогательные веса экспертов $1 \le i \le t$:

$$w^p_{i,t} = \frac{w_{i,t}}{\sum_{j=1}^{t} w_{j,t}}. \tag{10}$$

6. Вычисляем прогноз алгоритма $\gamma_t = \mathrm{Subst}(\mathbf{g}_t, \mathbf{w}^p_t)$, где $\mathbf{g}_t = (f_{1,t}, \ldots, f_{t,t})$ и $\mathbf{w}^p_t = (w^p_{1,t}, \ldots, w^p_{t,t})$.

7. Получаем исход y_t и вычисляем потери алгоритма $h_t = \lambda(\gamma_t, y_t)$, а также потери экспертов $1 \le i \le t$

$$l_{i,t} = \begin{cases} \lambda(f_{i,t}, y_t) \text{ если } i \le t, \\ h_t \text{ если } i > t. \end{cases}$$

8. Корректируем веса экспертов $w_{i,t+1} = w_{i,t} e^{-\eta l_{i,t}}$ при $1 \le i < \infty$.

ENDFOR

В практических приложениях на каждом шаге t нам необходимо вычислить веса $w_{i,t}$ только при $i \le t$. Согласно алгоритму

$$w_{i,t+1} = \begin{cases} w_{i,t} e^{-\eta l_{i,t}} \text{ если } i < t, \\ \frac{1}{i(i+1)} e^{-\eta(\sum_{s<t} h_s + l_{i,t})} \text{ если } i = t. \end{cases}$$

Это определение можно упростить до следующей рекуррентной формулы

$$w_{i,t+1} = \begin{cases} w_{i,t} e^{-\eta l_{i,t}} \text{ если } i < t, \\ w_{t-1,t}\big(1 - \frac{2}{t+1}\big) e^{-\eta(h_{t-1} - l_{t-1,t-1} + l_{t,t})} \text{ если } i = t. \end{cases}$$

Веса $w_{i,t}$ остаются неопределенными при $i > t$.

Верхняя оценка кумулятивного регрета $R_{i,T} = H_T - L_{i,t}$ относительно эксперта i представлена в следующей теореме.

Согласно предложению 1 верхняя оценка кумулятивного регрета $R_{i,T} = H_T - L_{i,t}$ относительно произвольного эксперта i имеет вид

$$R_{i,T} \le \frac{1}{\eta} \ln(i(i+1))$$

для всех T.

3.1. Эксперименты по агрегации локальных экспертов

Многие временные ряды показывают строгую зависимость своих членов от самой последней информации. В этом случае, в качестве экспертов полезно использовать регрессию со скользящим окном [8]. Применим Алгоритм 2, в котором экспертные стратегии строятся по скользящему окну. Прогноз каждого эксперта i построен на основе предположения о зависимости очередного исхода от членов временного ряда, составляющих окно в прошлое на шаге i. В стационарном случае такая зависимость относительно регулярная и метод экспоненциального взвешивания будет выделять наиболее релевантную из этих зависимостей.

Пусть $f_t(\mathbf{x}) = (\mathbf{w}_t \cdot \mathbf{x})$ – функция гребневой регрессии, где $\mathbf{w}_t = (\sigma I + X_t' X_t)^{-1} X_t' \mathbf{y}_t$ при $t > h$. Здесь X_t – матрица, у которой столбцы образованы векторами $\mathbf{x}_{t-h}, \ldots, \mathbf{x}_{t-1} \in R^k$, где X_t' – транспонированная матрица X_t, I – единичная матрица, σ – параметр, h – размер окна и $\mathbf{y}_t = (y_{t-h}, \ldots, y_{t-1})$. При $t \le h$ в качестве $\mathbf{w}_t$ берем некоторый фиксированный вектор.

Мы используем квадратичную функцию потерь и предполагаем, что $y_t \in [-b, b]$ для всех t, где $b > 0$ – граница интервала. В каждый момент времени t определим предсказание γ_{t+1} по формуле (11) используя функции регрессии f_i для $h < i \le t$, где каждая такая функция определена по скользящему окну $(\mathbf{x}_{i-h}, y_{i-h}), \ldots, (\mathbf{x}_{i-1}, y_{i-1})$.

Для квадратичной функции потерь

$$\lambda(\gamma, y) = (\gamma - y)^2,$$

где $y \in [-b, b]$, согласно (3), прогноз алгоритма **AA** равен

$$\gamma_t = \frac{1}{4\eta b} \ln \frac{\sum\limits_{i=1}^{t} w_{i,t}^p e^{-\eta((\mathbf{w}_i \cdot \mathbf{x}_t) - b)^2}}{\sum\limits_{i=1}^{t} w_{i,t}^p e^{-\eta((\mathbf{w}_i \cdot \mathbf{x}_t) + b)^2}}, \tag{11}$$

для алгоритма **WA** прогноз равен

$$\gamma_t = \left(\left(\sum_{i=1}^{t} w_{i,t}^p \mathbf{w}_t \right) \cdot \mathbf{x}_t \right), \tag{12}$$

где веса $w_{i,t}^p$ определены по (10).[2]

Синтетические данные генерируются с помощью нескольких генераторов. Число генераторов и их параметры неизвестны предсказателям. Последовательно генерируем последовательность $\mathbf{x}_1, \ldots, \mathbf{x}_T$ 20-мерных сигналов, независимо и одинаково распределенных согласно многомерному нормальному распределению. Всего было сгенерировано $T = 3000$ таких векторов.

Величина y_t генерируется следующим образом. Используется четыре типа линейных зависимостей, определяемых весовыми векторами $\hat{\mathbf{w}}_1, \hat{\mathbf{w}}_2, \hat{\mathbf{w}}_3, \hat{\mathbf{w}}_4$, т.е. $y_t = (\hat{\mathbf{w}}_i \cdot \mathbf{x}_t)$ при $i = 1, 2, 3, 4$. Временная шкала $[1, T]$ делится на $K = 9$ случайных последовательных интервалов, в каждом из которых используется одна из четырех функций регрессии $y_t = (\hat{\mathbf{w}}_i \cdot \mathbf{x}_t) + \epsilon$, где $i = 1$, $i = 2$, $i = 3$ или $i = 4$, а ϵ – стандартный нормальный шум. Таким образом, на временном интервале $[1, T]$ зависимость y от x переключается 9 раз.

Прогнозирующая стратегия (функция регрессии) $f_i(\mathbf{x})$ каждого эксперта i определяется по окну $(\mathbf{x}_{i-h}, y_{i-h}), \ldots, (\mathbf{x}_i, y_i)$ длины $h = 20$ (а также $h = 10, 15$ при других вариантах).

[2] Наиболее подходит значение $\eta = \frac{1}{2b^2}$ для правила (11) или $\eta = \frac{1}{8b^2}$ для правила (12). Заметим, что более простое определение (12) ведет к верхней оценке регрета, которая в четыре раза хуже.

3D-изображение весовых векторов действующих в данный момент времени генерирующих моделей, весовых векторов локальных экспертов в момент их инициализации, а также весовых векторов агрегирующей модели **WA** приведено на рис. 1.

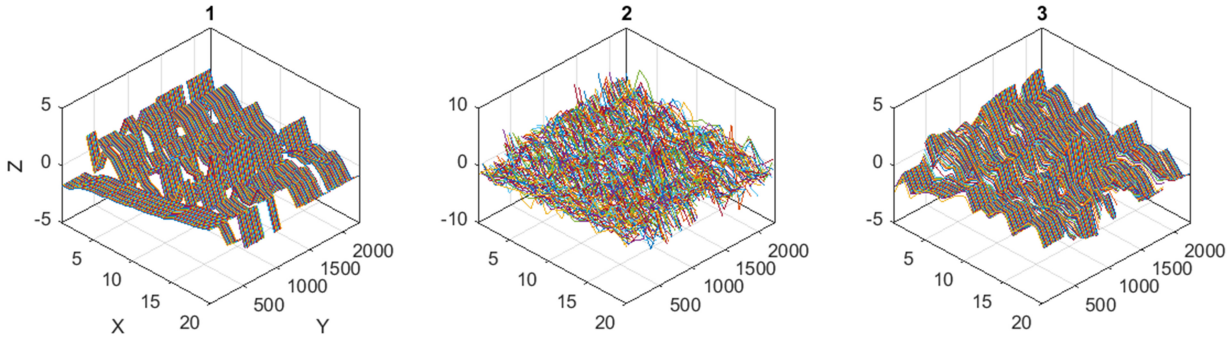

Рис. 1. 3D-изображение: (1) весовые векторы генерирующих моделей, (2) весовые векторы локальных экспертов в момент их инициализации, (3) весовые векторы агрегирующей модели **WA**.

Результаты эксперимента представлены на рис. 2, где представлены графики регрета $H_t - L_{i,t}$ Алгоритма 2 относительно экспертов, инициализированных в моменты времени $i \leq t$.

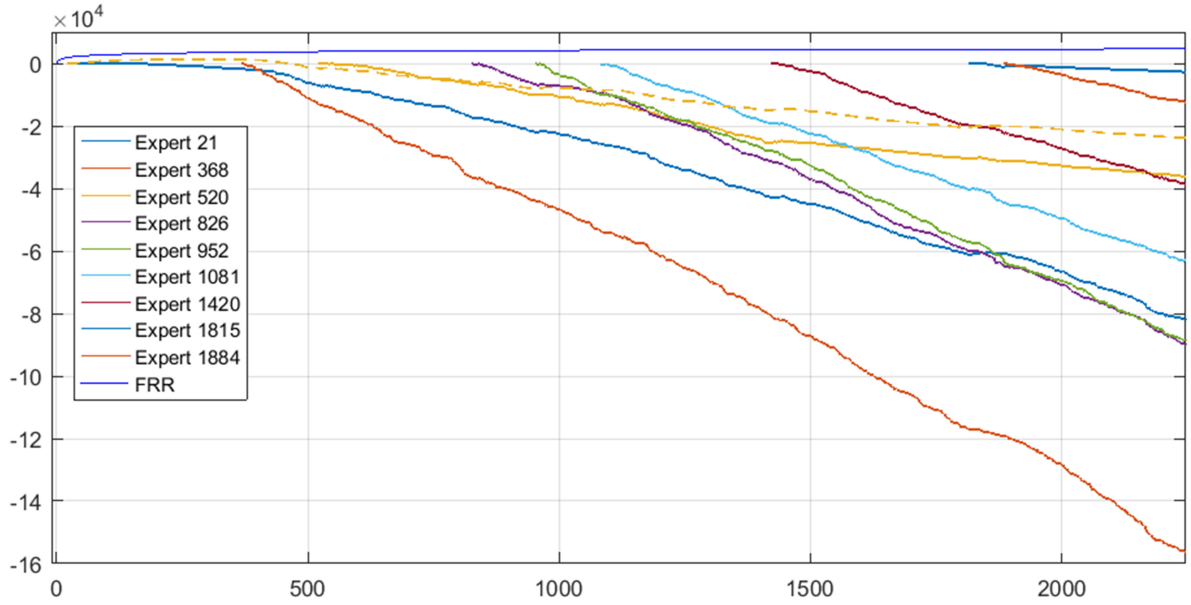

Рис. 2. Графики регрета **AA** относительно экспертов инициализированных в моменты времени, выбранных случайным образом. Теоретическая граница для регрета представлена линией, расположенной выше всех линий. Регрет относительно онлайн гребневой регрессии представлен пунктирной линией (обозначен FRR).

4. ОНЛАЙН ОБУЧЕНИЕ НЕЙРОННЫХ СЕТЕЙ

В этом разделе та же самая задача прогнозирования потока данных в режиме онлайн решается с помощью искусственных нейронных сетей различной архитектуры. При обучении нейронной сети в режиме онлайн нет необходимости заводить локальных предсказателей, а

затем агрегировать их прогнозы, мы просто дообучаем нейронную сеть на каждом шаге на основе данных из окна в прошлое.

В теории выпуклой оптимизации процесс обучения представляется в виде игры с полной информацией. На каждом шаге $t = 1, 2, \ldots$ представляется прогноз – вектор весов $\mathbf{w}_t$ и зависящая от него функция $h_{\mathbf{w}_t,t}(\cdot)$, которая задается с помощью нейронной сети и ее весов $\mathbf{w}_t$. Затем наблюдается сигнал x_t и вычисляется значение прогноза $f_t = h_{\mathbf{w}_t,t}(\mathbf{x}_t)$. После этого, объявляется исход y_t и вычисляются потери $\lambda(f_t, y_t)$. На основе этой информации производится переопределение вектора весов $\mathbf{w}_{t+1}$. Может также корректироваться архитектура нейронной сети. В результате получается прогноз $h_{\mathbf{w}_{t+1},t+1}(\cdot)$ для использования на следующем раунде обучения.

Общая схема обучения нейронной сети в режиме онлайн имеет следующий вид. Конкретные применения отличаются техническими деталями.

Алгоритм 2

Задано выпуклое подмножество S пространства $\mathcal{R}^n$. Пусть также задана функция потерь $\lambda(f, y)$ – выпуклая по f. Задана нейронная сеть фиксированной архитектуры, зависящая от вектора параметров $\mathbf{w}_t \in S$.

FOR $t = 1, \ldots, T$

- (1) Открываем прогноз – нейронная сеть $h_{\mathbf{w}_t,t}(\cdot)$, где $\mathbf{w}_t \in S$ – вектор параметров нейронной сети.
- (2) Получаем сигнал $\mathbf{x}_t \in \mathcal{R}^n$.
- (3) Вычисляем значение прогноза $f_t = h_{w_t,t}(\mathbf{x}_t)$.
- (4) Наблюдаем исход y_t и вычисляем текущие $\lambda(f_t, y_t) = \lambda(h_{\mathbf{w}_t,t}(\mathbf{x}_t), y_t)$ и кумулятивные $L_t = L_{t-1} + \lambda(h_{\mathbf{w}_t,t}(\mathbf{x}_t), y_t)$ потери.
- (5) Вычисляем параметры нейронной сети (прогноза) для следующего раунда $\mathbf{w}_{t+1} = P(\mathbf{w}_t - \eta\nabla_{\mathbf{w}_t}\lambda(h_{\mathbf{w}_t,t}(\mathbf{x}_t), y_t))$, где P – оператор проекции на множество S.

ENDFOR

4.1. Используемая архитектура нейронных сетей

Архитектура FICNN (Полностью выпуклая от входа нейронная сеть). Данная архитектура изображена на рис. 3. Она аналогична архитектуре из статьи [9] и характерна тем, что если потребовать, чтобы все веса нейронной сети были неотрицательными, то данная нейросеть будет представлять собой выпуклую от входа функцию при фиксированных значениях весов. В статье [9] описывается эксперимент, в котором такие нейросети использовались для решения задачи обучения с подкреплением. Архитектура сети представлена на рис. 3.

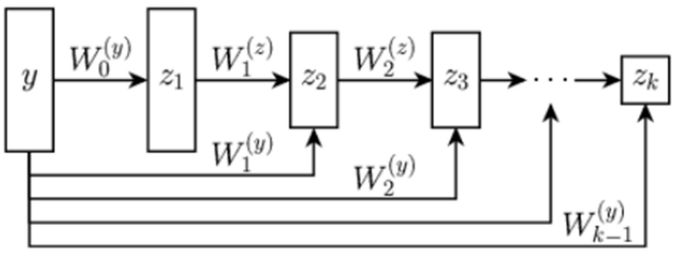

Рис. 3. Архитектура нейронной сети FICNN (Полностью выпуклая от входа нейронная сеть). Количество слоёв $K = 10$, в качестве функции активации использовалась *ReLU*. Рисунок заимствован из работы [9].

Пусть $\mathbf{y}$ – вход нейросети[3], K – количество слоёв. Обозначим за $\mathbf{z}_i$ выход i-го слоя, g_i – функцию активации i-го слоя. Тогда

$$\mathbf{z}_i = g_i\left(W_i^{(z)} \cdot \mathbf{z}_{i-1} + W_i^{(y)} \cdot \mathbf{y} + b_i\right),$$

где $W_i^{(y)}, W_i^{(z)}$ – матрицы весов i-го слоя, b_i – вектор смещения i-го слоя.

В проведённых экспериментах в архитектуре нейросети были сделаны следующие изменения: каждый слой $\mathbf{z}_i$ является произведением конкатенации $\mathbf{y}$, $\mathbf{z}_{i-1}$ и матрицы весов W_i. Количество слоёв $K = 10$, в качестве функции активации использовалась $ReLU$.

Архитектура LSTM (долгая краткосрочная память). Данная архитектура изображена на рис. 4. В этой архитектуре используются так называемые **LSTM** – юниты, описанные в статье [10]. Обычно они используются для генерации и прогнозирования последовательностей, и характерны тем, что на практике в большинстве случаев могут использовать зависимости между членами последовательности.

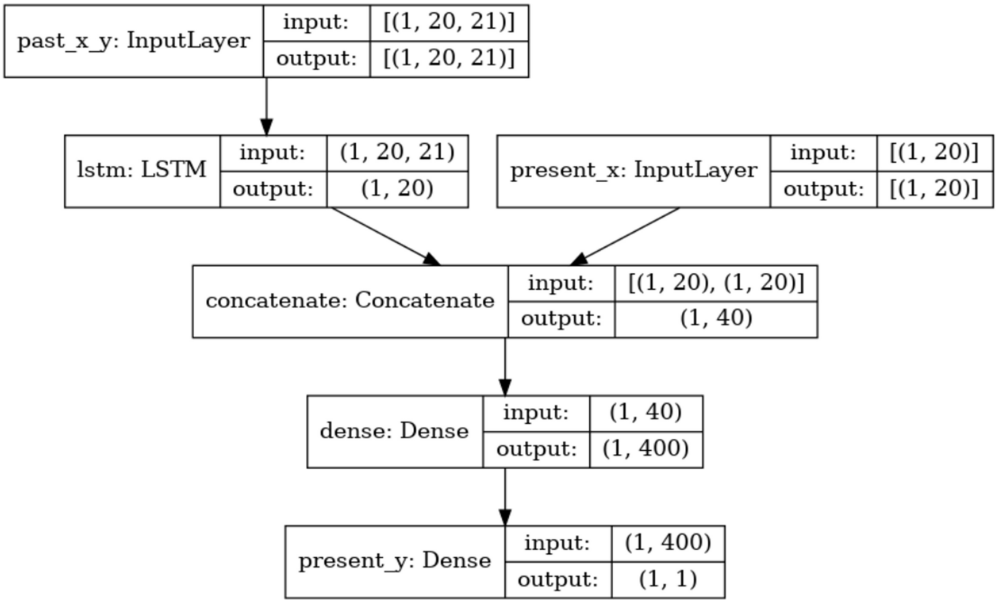

Рис. 4. Архитектура нейронной сети LSTM. Здесь **past_x_y** – 20 предыдущих пар $(\mathbf{x}_t, y_t)$. **lstm** – LSTM-слой **present_x** – текущий наблюдаемый сигнал $\mathbf{x}_t$. **present_y** – линейный слой, прогнозирующий значение y. Рисунок является модификацией рисунка из работы [10].

Каждый LSTM-юнит на вход принимает один элемент последовательности и выход предыдущего LSTM-юнита, если он есть. Выходов у LSTM-юнита тоже два: текущее состояние обработки последовательности и результат обработки последовательности вплоть до текущего элемента. Первый выход передаётся на вход следующему юниту, а второй является частью общего выхода LSTM-слоя, который состоит из LSTM-юнитов. LSTM-слой принимает на вход несколько элементов последовательности.

Данная архитектура состоит из двух частей: LSTM – анализирует последние 20 пар $(\mathbf{x}_t, y_t)$ и выдаёт в некотором виде результат анализа, и двух линейных слоёв, которые по выходу LSTM и текущему $\mathbf{x}_t$ пытаются предсказать y_t. После каждого линейного слоя в качестве функции активации использовалась $ReLU$. Таким образом, на каждом шаге Алгоритма 3 на

[3] В Алгоритме 3 на каждом шаге t берем $\mathbf{y} = \mathbf{x}_t$.

вход нейронной сети подаётся 20 предыдущих элементов последовательности $(\mathbf{x}_t, y_t)$, и новое наблюдаемое значение $\mathbf{x}_t$.

past_x_y – этот слой принимает 20 предыдущих пар $(\mathbf{x}_t, y_t)$.

lstm – в этом слое по 20 парам $(\mathbf{x}_t, y_t)$ получается некоторое промежуточное представление. Слой состоит из 20-ти LSTM-юнитов.

present_x – текущий наблюдаемый сигнал $\mathbf{x}_t$.

Всё это подаётся на вход двум линейным слоям, на выходе получается одно число – прогноз y_t.

Архитектура Hedge Backpropagation – HBP изображена на рис. 5. Архитектура взята из работы [11] и переделана под задачу регрессии. Основная проблема при онлайн обучении нейронной сети: в каждый момент времени нужно разное число слоев для эффективной работы. Авторы статьи предлагают применить алгоритм **Hedge** (см. [4]) к слоям нейронной сети, имитируя таким образом нейросеть переменной глубины.

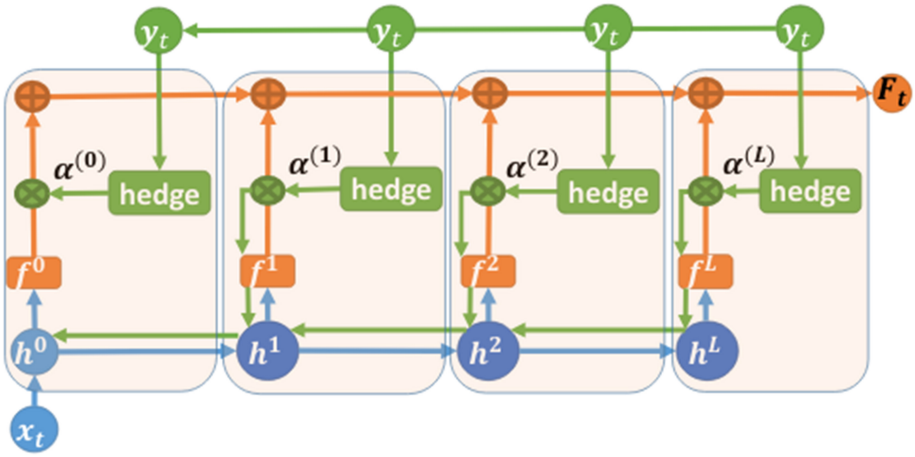

Рис. 5. Схема Hedge Backpropagation. Рисунок заимствован из работы [11].

Обозначим через $\mathbf{x}_t$ вход нейросети, а через $\mathbf{h}^{(k)}$ – выход k-го слоя. Также обозначим через f^k "предиктор" k-го слоя. Это линейный слой, который по $h^{(k)}$ выдаёт прогноз для y. Тогда нейросеть можно записать следующим образом:

1. $\mathbf{h}^{(0)}(\mathbf{x}_t) = ReLU(W_0 \cdot \mathbf{x}_t)$
2. $\mathbf{h}^{(k)} = ReLU(W_k \cdot \mathbf{h}^{(k-1)} + a_t)$
3. $f^{(k)} = \Theta_k \cdot \mathbf{h}^{(k)} + b_t$

Для каждого слоя поддерживается его текущий вес $\alpha^{(k)}$, аналогично алгоритму **Hedge**.

Итоговый прогноз нейронной сети вычисляется следующим образом:

$$F^{(t)} = \sum_{k=0}^{K-1} f^k \cdot \alpha^{(k)}$$

Обновление весов $\alpha^{(k)}$ происходит аналогично алгоритму **Hedge** после запуска обратного распространения ошибки.

Основная идея состоит в следующем: выход каждого линейного слоя $\mathbf{h}^{(i)}$ подаётся на вход не только следующему линейному слою $\mathbf{h}^{(i+1)}$, но и слою, который считает ответ – $f^{(i)}$. Этот слой тоже линейный, размерность выхода 1 (так как надо получить прогноз $\widehat{y}_t \in \mathbb{R}$). Все

прогнозы суммируются с весами $\alpha^{(i)}$, и эти веса обновляются на шаге **observe**. Архитектура нейронной сети приведена на рис. 5. Приводим описание алгоритма обучения:

Алгоритм Hedge Backpropagation (HBP)

1. Сравниваем выход каждого $f^{(i)}$ с наблюдаемым y_t, подсчитываем ошибку и градиент ошибки для каждого $f^{(i)}$.
2. Умножаем градиенты на $\alpha^{(i)}$
3. Запускаем backpropagation
4. Обновляем $\alpha^{(i)} \leftarrow \alpha^{(i)} \cdot \beta^{\lambda(y_t, f^{(i)})}$
5. Нормируем все $\alpha^{(i)}$ (их сумма должна быть равна 1)

β – гиперпараметр, характеризующий скорость обучения.

В проведённых экспериментах использовалось $K = 20$ слоёв.

Поскольку целесообразность умножения градиента на $\alpha^{(i)}$ под вопросом, эксперименты были проведены также для аналогичной модели, но без умножения градиента на $\alpha^{(i)}$.

Архитектура Hedge Backpropagation + AA (HBP+AA). Архитектура полностью аналогична предыдущей за исключением метода обновления $\alpha^{(i)}$ – обновление происходит как в алгоритме **AA**. В проведённых экспериментах использовалось $K = 15$ слоёв для данной архитектуры.

4.2. Результаты экспериментов

Эксперименты со всеми приведенными выше нейронными сетями проводились на тех же синтетических данных, что и эксперименты с агрегированием прогнозов локальных экспертов алгоритмами **AA** и **WA** в разделе 3.1.

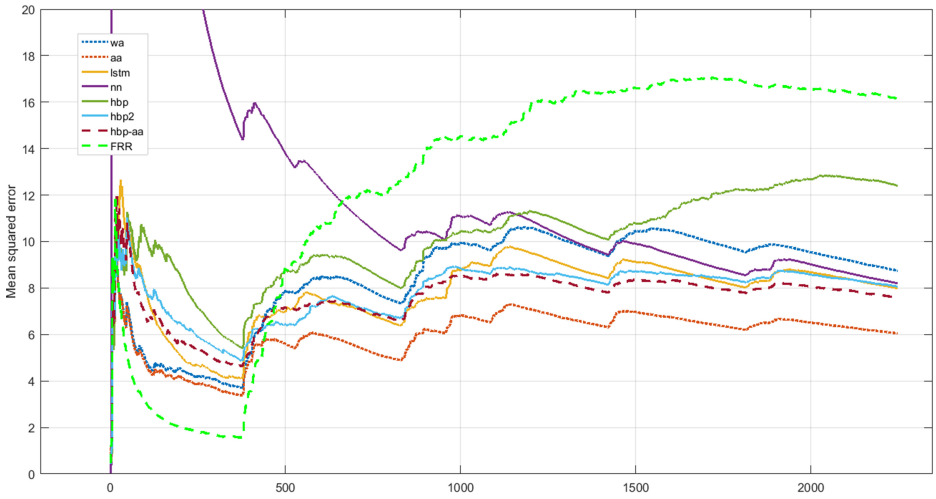

Рис. 6. Средние потери H_T/T предсказания различных прогнозирующих систем. Обозначения на графике: nn – **FICNN**, aa – алгоритм **AA**, wa – алгоритм **WA**, lstm – **LSTM**, hbp – Hedge Backpropagation, hbp2 – Hedge Backpropagation (второй вариант), hbp2_aa – Hedge Backpropagation+**AA**, FRR – онлайн гребневая регрессия на отрезке $[1, T]$. По горизонтальной оси отложено время $T = 1, \ldots, 3000$.

Сравнительный анализ средних потерь H_T/T, при $T = 1, \ldots, 3000$, всех видов нейронных сетей: lstm, nn, hbp, hbp2, hbp-aa, а также алгоритмов **AA** и **WA** (на графике обозначены как aa и wa), агрегирующих прогнозы локальных экспертов, приведены на рис. 6. На рисунке также приведены средние потери полной гребневой регрессии FRR (Full Ridge Regression), которая в каждый момент времени T строится по данным из временного интервала $[1, T]$.

Результаты этих экспериментов показывают, что на всем интервале $[1, 3000]$ алгоритм **AA** (aa) дает большую точность предсказания чем остальные методы, что ведет к наименьшим средним потерям. Следующие близко расположенные кривые представляют средние потери нейросети hpb-aa, затем идут ltsm и hbp2, от них отстает нейросеть nn. Алгоритм **WA** (обозначение – wa) и нейросеть hbp завершают список кривых. Самые большие средние потери на всем интервале имеет метод онлайн гребневой регрессии FRR.

На первом интервале генерации (примерно $[1, 400]$) метод FRR быстрее всех адаптируется к типу данных, однако при смене генератора он быстро теряет свою релевантность, так как при обучении использует всю предысторию данных.

Локальные эксперты лучше всех моделей адаптируется к изменению типа генератора, а алгоритм **AA** быстрее **WA** понижает вес локальных экспертов, которые были обучены на прошлых сегментах данных, и поэтому имеет наименьшие средние потери.

Отметим также, что Hedge Backpropagation в комбинации с алгоритмом **AA** (hbp+aa) для обновления весов работает немного лучше, чем авторский вариант hbp с алгоритмом **Hedge** из работы [11]. В целом, можно заключить, что большая часть нейросетевых архитектур показывают эффективность схожую с **AA** и **WA**.

Алгоритм **AA** дает большую среднюю точность предсказания чем остальные методы в условиях, когда окно в прошлое имеет ограниченную длину (20 точек). При увеличении размера окна может происходить потеря релевантности локальных моделей. При увеличении размера окна, представленного для дообучения нейронной сети, точность прогнозов **AA** и нейросетей **LSTM** и **HBP**, **HBP+AA** сближаются.

5. ЗАКЛЮЧЕНИЕ

Рассматривается подход, при котором обучение прогнозирующей системы производится во все моменты времени. Этот подход называется в современной литературе Lifelong Machine Learning.

Решается задача прогнозирования потока данных в режиме онлайн. При вычислении очередного прогноза не использовали никаких предположений о природе источника, генерирующего элементы временного ряда – источник может быть алгоритмическим или вероятностным, смена его параметров может происходить в случайные моменты времени.

Предлагается два адаптивных метода машинного обучения для прогнозирования потока данных.

Первый метод основан на изучении и запоминании локальных свойств временного ряда. В каждый момент времени автоматически строится очередная локальная модель временного ряда, все эти модели сохраняются в памяти и используются в дальнейшем для построения агрегирующей прогнозирующей стратегии.

Второй метод основан на применении искусственных нейронных сетей, обучающихся в режиме онлайн. По мере поступления данных в режиме онлайн, искусственная нейронная сеть заданной архитектуры непрерывно обновляет свои параметры в зависимости от изменения характера временного ряда.

Проводится сравнительный анализ эффективности предложенных методов. Показано, что алгоритм первого метода, который использует более конкретную модель источника данных, приводит к меньшим потерям от предсказания, чем нейронная сеть, которая использует более общий подход к построению такой модели. Это преимущество заметно только тогда, когда обоим методам доступна ограниченная информация при их модификации. При увеличении доступной предыстории нейронная сеть дает предсказания, которые приводят к меньшим потерям.

ФИНАНСИРОВАНИЕ

Работа частично поддержана грантом РФФИ 20-01-00203.

СПИСОК ЛИТЕРАТУРЫ

1. German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, Stefan Wermter *Continual Lifelong Learning with Neural Networks: A Review*. 2018. arXiv:1802.07569 [cs.LG].
2. V. Vovk, Aggregating strategies. In M. Fulk and J. Case, editors, *Proceedings of the 3rd Annual Workshop on Computational Learning Theory*, 371–383. San Mateo, CA, Morgan Kaufmann, 1990.
3. V. Vovk, A game of prediction with expert advice. *Journal of Computer and System Sciences*. 56(2), 153–173, 1998.
4. N. Cesa-Bianchi, G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.
5. Y. Freund, R.E. Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*. 55:119–139, 1997.
6. N. Littlestone, M. Warmuth. The weighted majority algorithm. *Information and Computation*. 108:212–261, 1994.
7. A. Chernov and V. Vovk. Prediction with expert evaluators advice. In Algorithmic Learning Theory, ALT 2009, *LNCS*, 5809:8–22, 2009.
8. Alexander Korotin, Vladimir V'yugin, Evgeny Burnaev; Mixing past predictions. *Proceedings of Machine Learning Research (PMLR)*. 128:171–188, 2020.
9. Brandon Amos, Lei Xu, J.Zico Kolter: *Input Convex Neural Networks*: `https://arxiv.org/pdf/1609.07152.pdf`
10. Sepp Hochreiter, Jurgen Schmidhuber: *Long Short-Term Memory*: `https://www.bioinf.jku.at/publications/older/2604.pdf`
11. Doyen Sahoo, Quang Pham, Jing Ju, Steven C.H. Hoi: *Online Deep Learning: Learning Deep Neural Networks on the Fly*: `https://arxiv.org/pdf/1711.03705.pdf`

Online Learning of Multicomponent Forecasting Systems

V.V. V'yugin, V. Kalmykov, V.G. Trunov

Institute for Information Transmission Problems, Russian Academy of Sciences, Moscow, Russia

An approach is considered in which the learning of the predictive system is carried out at all points in time. This approach is called Lifelong Machine Learning in modern literature. As an example of the application of this approach, the problem of forecasting a time series online is being solved. When calculating the next forecast, no assumptions about the nature of the source generating elements of the time series are used – the source can be algorithmic or probabilistic, and its parameters can change at random times. Under these conditions, predictive systems must be trained continuously as input information arrives. Two types of adaptive machine learning algorithms are proposed. The first algorithm performs online aggregation of local time series models, which are automatically generated in the process of studying it. The second group of methods is based on the use of artificial neural networks, students online. Comparative analysis of efficiency is carried out proposed methods.

KEYWORDS: Lifelong Machine Learning, online adaptive prediction algorithms, prediction with expert strategies, regret, aggregating algorithm, artificial neural networks.