УДК: 004.8, 004.94, 51-74, 621.37

# Разработка и исследование методики построения нейронных сетей на основе адаптивных элементов

Е.Н. Ефимов, Т.Я. Шевгунов

**Аннотация**: В работе рассмотрены искусственные нейронные сети прямого распространения сигнала, обучаемые по методу обратного распространения ошибки. Для представления такой сети используется системный подход, описывающий её в форме структурного графа взаимосвязанных элементов, в которых осуществляются преобразования сигналов, распространяющихся в сети в прямом и обратном направлениях. Для практической реализацией такого подхода был разработан прототип программного обеспечения, частью которого является библиотека классов, реализующая основные адаптивные элементы и связи между ними. Также в настоящей работе представлены результаты численного моделирования решения двух практических задач: аппроксимации радиолокационного изображения и классификации двух случайных процессов.

**Ключевые слова**: нейронная сеть; обратное распространение ошибки; адаптивный элемент; сигналы и системы; математические методы моделирования; метод градиентного спуска; *SageMath*; *Python*.

## 1. Введение

В данной работе рассматриваются искусственные нейронные сети – модели, в основе которых лежат принципы, заимствованным из сетей нервных клеток живых организмов. Продуктивное развитие методов, использующих принципы нейронных сетей, позволило за последние 20 лет получить успешные результаты в области, получившей неформальное название «интеллектуальной» обработки данных [1].

Традиционная нейронная сеть прямого распространения представляет собой систему взаимодействующих адаптивных элементов – нейронов [2–4], каждый из которых выполняет определенное функциональное преобразование над сигналами. Существуют два основных

методологических подхода к описанию процессов нейросетевой обработки сигналов: математический, основанный на описании сети в терминах функциональных преобразований, и системный, представляющий сеть в форме взаимодействующих подсистем, в которых происходит преобразование входных сигналов в выходные. В настоящем исследовании используется системный подход, на основе которого предложена методика, предполагающая разбиение всей нейронной сети на блоки простых адаптивных элементов.

Так в работе [5] было впервые предложено представить процесс обратного распространения ошибки с помощью функциональной схемы, которая получила название системной диаграммы представления обратного распространения (*backpropagation diagrammatic representation*). Такая диаграмма является наглядным средством, позволяющим описать функционирование алгоритма обратного распространения в удобной для анализа форме. Однако в своей работе авторы используют её как вспомогательный инструмент для упрощения вывода необходимых выражений при анализе динамических нейронных сетей – НС, предназначенных для обработки сигналов, являющихся функциями времени. Этот инструмент также был заимствован впоследствии другими авторами, например [4, 6], как наглядный способ представления правил обратного распространения при изучении предмета нейронных сетей.

В настоящей работе сделана попытка развить и систематизировать предложенный способ описания. Для этого авторы предлагают построить нейронную сеть на основе адаптивных элементов, которые предполагается сохранять отдельными друг от друга при построении математической модели сети. Между элементами организуются двунаправленными связи, которые в целом образуют два совмещенных графа для описания прямого и обратного распространения сигнала. Каждый элемент осуществляет обработку сигналов в прямом и обратном направлении, а также выполняет подстройку своих адаптируемых параметров в фазе обучения сети с использованием одного из известных методов обучения [7].

Оставшаяся часть работы составлена следующим образом. Во второй части вводится общее описание адаптивного элемента и рассматриваются принципы его взаимодействия с другими элементами. В третьей части рассматривается пример синтеза простых адаптивных элементов, реализующих желаемые методы обучения. В четвёртой части представлена реализация методики адаптивных элементов на языке программирования высокого уровня *Python* с использованием объектно-ориентированного подхода. В пятой части представлены выполненные с помощью разработанного прототипа программного обеспечения численные

эксперименты решения двух задач: аппроксимации и классификации. В заключении приводятся выводы и обсуждаются направления дальнейшего развития метода.

## 2. Теория адаптивных элементов

Рассмотрим некоторый элемент, реализующий известное преобразование входного вектора **x** в выходной вектор **y**. В самом общем виде такое преобразование можно записать в форме некоторого преобразования известной формы **T**:

$$\mathbf{y} = \mathbf{T}(\mathbf{x}, \boldsymbol{\theta}), \tag{1}$$

где под вектором $\boldsymbol{\theta}$ будет подразумеваться вектор параметров этого преобразования. Форма преобразования **T** определяется назначением и внутренней структурой элемента, а конкретное значение вектора его параметров $\boldsymbol{\theta}$ определяется в процессе его обучения.

Ключевой идеей, лежащей в основе функционирования адаптивного элемента, является понятие полного дифференциала функции, который для общей записи выражения (1) имеет вид:

$$d\mathbf{y} = \frac{\partial \mathbf{T}(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}} \cdot d\mathbf{x} + \frac{\partial \mathbf{T}(\mathbf{x}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \cdot d\boldsymbol{\theta} \quad . \tag{2}$$

Формула (2) с точки зрения обучения может быть интерпретирована следующим образом. Желаемое изменение выходного вектора $d\mathbf{y}$ может быть достигнуто как за счёт изменения параметров элемента $d\boldsymbol{\theta}$, так и за счёт изменения входного сигнала $d\mathbf{x}$. Частные производные определяют силу влияния изменений входного вектора и изменений параметров на изменение выходного вектора, или, говоря по другому, чувствительность выхода элемента к его входу и к его параметрам. Из формулы (2) естественно следует, например, частное свойство, состоящее в том, что если элемент не является адаптивным, т.е. не содержит параметров, которые можно адаптировать, то изменение выходного вектора возможно только за счёт изменения входного сигнала.

Формально можно считать, что нейронная сеть представляет совокупность адаптивных элементов и сигнальных связей между ними. Между элементами нейронной сети сигналы распространяются в двух направлениях, которые традиционно называются прямым (*feedforward*) и обратным (*backpropagation*). Прямое прохождение сигнала используется для получения выходного сигнала $\mathbf{y}_F$ элемента при известном входном сигнале $\mathbf{x}_F$, форме преобразования **T** и значениях его параметров $\boldsymbol{\theta}$. Обратное прохождение необходимо для формирования локального градиента вектора параметров адаптируемого элемента и сигнала ошибки $\mathbf{y}_B$. Локальный градиент используется для изменения параметров самого элемента, а сигнал ошибки $\mathbf{y}_B$ передаётся тем элементам, выходные сигналы которых подаются на вход

рассматриваемого элемента при прямом прохождении. Входным сигналом обратного прохождения $\mathbf{x}_B$ служит сигнал, подаваемый с элементов, которые используют выходной сигнал прямого прохождения $\mathbf{y}_F$ в качестве своих входных сигналов. Данный принцип проиллюстрирован рис. 1.

Для того, что бы обеспечить подобное прохождение сигналов, каждый элемент сети должен обрабатывать сигналы как при прямом, так и при обратном их прохождении, а каждая связь между элементами фактически должна представлять собой двунаправленный канал, как показано на рис. 1 двунаправленными стрелками.

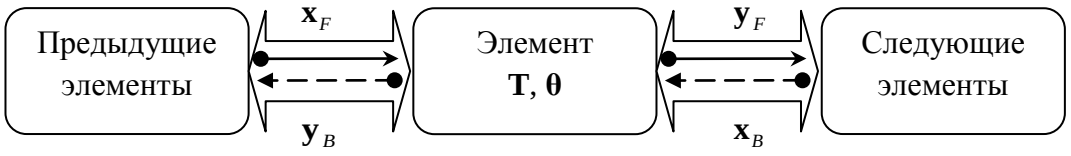



Рис 1. Структура сигнальных связей адаптивного элемента.

Будем считать, что нейронная сеть может находиться в двух рабочих режимах: режиме эксплуатации и режиме обучения. В режиме эксплуатации сеть считается обученной и выполняет свою основную задачу, для которой она была разработана. Например, её задачей может быть вычисление функции для подаваемых на её вход значений аргументов в задачах аппроксимации или выдача индикаторного сигнала отнесения к одному из классов в задачах классификации. В режиме эксплуатации параметры элементов сети не изменяются.

В режиме обучения сеть по некоторому набору правил, формирующих парадигму обучения, изменяет значения параметров своих элементов. В этом режиме, как правило, сеть не используется для решения своей основной задачи, а основное внимание уделяется контролю внутренних процессов сети. В том случае, если используется обучение с учителем (*supervised learning*), настраиваемой сети последовательно предлагаются примеры из обучающего множества, каждый из которых состоит из пары: входной набор значений и соответствующий ему выходной набор.

Данный процесс повторяется циклически для всех элементов обучающего множества. На каждой итерации выполняются последовательно два основных шага: расчёт сигналов прямого распространения и расчёт сигналов обратного распространения. Оба этих шага выполняются при неизменных значениях адаптируемых параметров. В результате расчёта прямого распространения определяется текущий отклик каждого элемента сети на входное воздействие. Отклики элементов соединенных непосредственно с выходом сети формируют реакцию цепи на входное воздействие при фиксированных адаптируемых параметрах. Для выполнения второго шага, обратного распространения, требуется знать желаемое значение

отклика сети и все промежуточные значения составляющих её элементов. Результатом этого шага является вычисление т.н. локальных градиентов, рассчитываемых для каждого из адаптируемых параметров. Локальный градиент представляет собой коэффициент чувствительности итогового отклика сети к изменению параметров при:

- фиксированных значениях адаптируемых параметров;
- используемом на данной итерации элементе обучающего множестве.

Вычисленные локальные градиенты используются для последующего изменения значений адаптируемых параметров. Их использование определяется выбранными режимом и методом обучения. Основные известные режимы обучения [4, 8] – это последовательный режим, при котором подстройка параметров происходит после каждого примера, и пакетный (*batch*), при котором подстройка осуществляется на основе кумулятивного локального градиента, как правило, суммы локальных градиентов по всем итерациям примеров из обучающего множества. В обоих режимах полный цикл представления множества шаблонов обучения, завершающийся подстройкой параметров, называется эпохой обучения сети.

Чтобы придать формальную запись рассмотренным выше положениям, необходимо определить способ для количественной оценки того, насколько качественно сеть выполняет своё целевое предназначение. Для оценки качества работы сети вводится функция потерь, традиционно [4] имеющая смысл среднеквадратической ошибки (СКО):

$$E = \frac{1}{2N}\sum_{n=1}^{N}\left(t_n - z_n\right)^2 = \frac{1}{2N}\sum_{n=1}^{N}\left(\Delta z_n\right)^2 = \frac{1}{2N}\sum_{n=1}^{N} e_n\,, \qquad (3)$$

где $N$ – количество представленных шаблонов; $t_n$ – желаемый, или целевой (*target*) выходной сигнал сети в $n$-ом шаблоне; $z_n$ –выходной сигнал, вычисляемый сетью при подаче входного сигнала из $n$-ого шаблона; $\Delta z_n$ и $e_n$ – выходная и квадратичная выходная ошибки соответственно; коэффициент 1/2 перед дробью выбран для удобства последующих операций [8], хотя может быть положен равным единице или любому другому неотрицательному числу.

Фактически отклик сети $z$ является функцией от вектора входных сигналов $\mathbf{x}$ и вектора адаптируемых параметров $\boldsymbol{\theta}$. Полагая, что входные сигналы фиксированы, требуется решить задачу минимизации СКО на множестве определения вектора $\boldsymbol{\theta}$. Поскольку зависимость функции z от вектора $\boldsymbol{\theta}$ в общем случае нелинейная, то вывод аналитического выражения для оптимального значения $\boldsymbol{\theta}_{opt}$, минимизирующего СКО, может оказаться трудно разрешимой задачей, не поддающейся чёткой алгоритмизации при анализе даже топологически похожих сетей.

Однако поиск оптимального значения $\boldsymbol{\theta}_{opt}$ может быть выполнен на основе итеративных приближений. Для этого необходимо, начав с некоторого начального значения $\boldsymbol{\theta}_0$, изменять каждый из элементов вектора $\boldsymbol{\theta}$ в том направлении, которое приводит к убыванию функции СКО $E$, заданной в (3). Если считать примеры, подаваемые для обучения сети независимыми друг от друга, то локальный градиент при подаче единственного примера может быть вычислен по формуле:

$$\delta\theta_m = \frac{\partial e}{\partial \theta_m} = \frac{\partial e}{\partial z} \cdot \frac{\partial z}{\partial \theta_m} = \Delta z \cdot \frac{\partial z}{\partial \theta_m}, \tag{4}$$

при этом выходная ошибка $\Delta z$ оказывается общим для всех параметров коэффициентом, вычисляемой вне самой сети. Если рассмотреть всю сеть как некий адаптивный элемент (1), то $\Delta z$ играет для него роль входного сигнала ошибки $x_B$ (см. рис. 1).

Рассмотрим здесь один полезный пример. Предположим, что $z$, как функция, зависит от некоторого переменного $u$, сигнала или параметра, опосредовано, т.е. через другие функции:

$$z = z\left(g_1(u), g_2(u), \ldots, g_K(u)\right), \tag{5}$$

тогда вычисление частной производной функции по параметру $u$ подчиняется двум правилам: правилу дифференцирования сложной функции и правилу вычисления полной производной:

$$\frac{\partial z}{\partial u} = \sum_{k=1}^{K}\left(\frac{\partial z}{\partial g_k}\frac{\partial g_k}{\partial u}\right). \tag{6}$$

Такое правило позволяет вычислить локальные градиенты для всех адаптируемых параметров, выводя их последовательно через те сигналы, которые зависят от них непосредственно. В более традиционном описании нейронных сетей, например в [4], непосредственное вычисление по формуле (4) оказывается возможным для параметров выходного слоя сети, а вычисление параметров скрытых слоёв сети проводится с использованием правила «цепного» дифференцирования, ключевой фрагмент которого и представлен формулой (6).

## 3. Синтез простых адаптивных элементов

Одним из шагов синтеза адаптивного элемента является задание метода его обучения. Далее мы проведём классификацию методов обучения по критерию положения источника используемых в них данных.

Автономными методами обучения мы будем называть такие, которые для подстройки вектора параметров адаптивного элемента используют только те сигналы, которые

присутствуют в самом рассматриваемом элементе. Реализация автономного метода обучения может быть заключена внутри адаптивного элемента в рамках модели двунаправленной связи, представленной ранее и проиллюстрированной рис. 1. При использовании автономных методов корректирующее изменение $\Delta\theta_m$ для $m$-го элемента $\theta_m$ из вектора адаптируемых параметров $\boldsymbol{\theta}$ вычисляется на основе информации, полученной только из его локального градиента $\delta\theta_m$ и истории изменения этого локального градиента с течением эпох.

В неавтономных методах обучения для подстройки вектора параметров некоторого элемента требуются сигналы, существующие в других элементах. Как правило, для реализации таких методов требуется дополнительный блок на уровне всей нейронной сети, который не участвует в прямом распространении сигнала, а задействуется в конце каждой эпохи, в фазе изменения параметров. Задача этого блока состоит в том, чтобы сформировать корректирующие изменение $\Delta\boldsymbol{\theta}_m$ для вектора параметров $\boldsymbol{\theta}_m$ на основе обработки значений локальных градиентов всех адаптируемых элементов, а не только того локального градиента, который был сформирован для $m$-го элемента.

Рассмотрим кратко некоторые автономные методы первого порядка, использованные в практической части настоящей работы.

1. Метод **градиентного спуска** является самым простым методом обучения сети. Вектор параметров адаптируемых параметров $\boldsymbol{\theta}$ корректируется на величину $\Delta\boldsymbol{\theta}$

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \Delta\boldsymbol{\theta}_i, \tag{7}$$

где $\boldsymbol{\theta}_i$ и $\boldsymbol{\theta}_{i+1}$ – значения вектора в текущую и следующую эпоху соответственно. Величина коррекции в текущую эпоху $\Delta\boldsymbol{\theta}_i$ вычисляется с использованием локального градиента: кумулятивного или текущего (в зависимости от режима обучения):

$$\Delta\boldsymbol{\theta}_i = -\varepsilon \cdot \delta\boldsymbol{\theta}_i, \tag{8}$$

где $\varepsilon$ – коэффициент скорости обучения сети, а $\delta\boldsymbol{\theta}$ – вектор локальных градиентов для элементов вектора $\boldsymbol{\theta}$. Метод градиентного спуска является базовым методом, на основе которого строятся другие автономные методы первого и так называемого квазивторого порядка.

2. Добавление эффекта **инерции** (*momentum*) при изменении параметров расширяет метод градиентного спуска так, что на подстройку параметра в текущей эпохе, оказывает влияние подстройки параметров в предыдущие эпохи. Мера влияния предыдущих подстроек на текущую определяется специальным параметром – коэффициентом инерции $\mu$: $0 \le \mu < 1$. Выражение для вычисления коррекции (7) преобразуется в формулу:

$$\Delta\boldsymbol{\theta}_i = \mu \cdot \Delta\boldsymbol{\theta}_{i-1} - (1-\mu) \cdot \varepsilon \cdot \delta\boldsymbol{\theta}_i. \tag{9}$$

Цель применения инерции состоит в том, чтобы ускорить достижение элементами вектора адаптируемых параметров **θ** своих оптимальных значений.

3. В работе [8] подробно рассмотрен метод, известный под названием **Delta-Bar-Delta**. Принципиальное расширение данного метода заключается в том, что для каждого адаптивного параметра вводится индивидуальный коэффициент скорости обучения. После каждой эпохи обучения происходит как подстройка адаптируемых параметров, так и подстройка коэффициента скорости обучения. Для упрощения выкладок рассмотрим один из адаптируемых параметров, который обозначим через $\theta$. Для корректировки его скорости изменения вводится вспомогательный параметр $f$, также изменяющийся с течением номера эпохи по правилу:

$$f_i = \gamma f_{i-1} + (1-\gamma)\cdot\delta\theta_{i-1}. \qquad (10)$$

где коэффициент $\gamma\left(0 \le \gamma < 1\right)$ определяет «глубину памяти» накопления истории предыдущих значений градиента. Для текущей эпохи вспомогательный параметр $f$ определяет градиент, накопленный за предыдущие эпохи. Если знак величины градиента текущей эпохи $\delta\boldsymbol{\theta_i}$ совпадет со знаком коэффициента $f_i$, то скорость обучения увеличивается, иначе – уменьшается: согласно правилу:

$$\varepsilon_i = \begin{cases} \varepsilon_{i-1} + k, & d_i f_i > 0; \\ \varepsilon_{i-1}\cdot\varphi, & d_i f_i \le 0. \end{cases} \qquad (11)$$

Параметры $\varphi$ и $\kappa$, выбираемые в диапазоне от нуля до единицы, определяют, насколько велико будет изменение скорости обучения при каждой подстройке. Вектор, составленный из значений скоростей $\boldsymbol{\varepsilon}_i$, вычисляемых по формуле (11), и значение градиента $\delta\boldsymbol{\theta_i}$ используются в формуле (8) для вычисления нового значения вектора адаптируемых параметров.

Метод адаптации индивидуальных скоростей Delta-Bar-Delta может быть также использован совместно с инерцией, как это предложено, например, там же, в [8]. Для этого при вычислении новых значений вектора адаптируемых параметров следует использовать формулу (9). Использование автономных методов обучения позволяет построить сравнительно простые модели элементов, простейшие из которых показаны на рис. 2 а)–г).

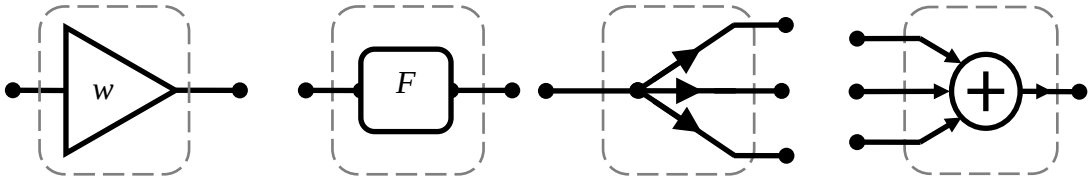



Рис 2. Простейшие элементы нейронной сети: а) адаптивный умножитель, б) безынерционный функциональный преобразователь, в) разветвитель, г) сумматор.

Рассмотрим эти элементы более подробно. Адаптивный умножитель, представленный на рис. 2, а), представляет собой элемент, реализующий при прямом прохождении сигнала преобразование

$$y_F = w \cdot x_F , \tag{12}$$

где $y_F$ и $x_F$ – входной и выходной сигналы, принятые здесь для упрощения скалярными, а $w$ – единственный адаптируемый параметр.

Выходной сигнал ошибки, формируемый данным элементом, определяется выражением, в котором значение производной функции $y_F$ из (2) не зависит от значения подаваемого входного сигнала $x_F$:

$$y_B = x_B \cdot \frac{\partial y_F}{\partial x_F} = x_B \cdot w , \tag{13}$$

что с точностью до обозначения переменных соответствует выражению прямой зависимости (12). Локальный градиент $\delta w$ для параметра $w$ адаптивного умножителя вычисляется по формуле:

$$\delta w = x_B \cdot \frac{\partial y_F}{\partial w} = x_B \cdot x_F , \tag{14}$$

Функциональный безынерционный элемент, изображенный на рис. 2, б), осуществляет преобразование текущего значения входного сигнала в выходной сигнал, представимое в форме функциональной зависимости:

$$y_F = F\left(x_F\right). \tag{15}$$

Это преобразование не содержит адаптируемых параметров. Выходной сигнал обратного распространения вычисляется по формуле:

$$y_B = x_B \cdot \frac{\partial y_F}{\partial x_F} = x_B \cdot F'\left(x_F\right), \tag{16}$$

Это означает, что при обратном распространении функциональный элемент представляет собой умножитель с коэффициентом, равным производной функции преобразования, вычисленной при текущем значении входного сигнала $x_F$.

Сумматор представляет собой элемент с одним выходом и несколькими входами, количество которых обозначим через $N$. Выходной сигнал прямого прохождения определяется суммой:

$$y_F = \sum_{n=1}^{N} (\mathbf{x}_F)_n = (\mathbf{x}_F)_0 + (\mathbf{x}_F)_1 + \ldots + (\mathbf{x}_F)_N \,. \tag{17}$$

Выходной сигнал ошибки каждого из входов будет одинаков и определяется выражением:

$$(y_B)_n = x_B \cdot \frac{\partial y_F}{\partial (\mathbf{x}_F)_n} = x_B \,, \tag{18}$$

поскольку частная производная суммы по любому из слагаемых равна единице.

Разветвитель представляет собой элемент с одним входом и $N$ выходами, реализующим при прямом прохождении повторение значение входного сигнала на каждом из выходов:

$$(\mathbf{y}_F)_n = x_F \,. \tag{19}$$

Тогда выходной сигнал ошибки будет определяться выражением:

$$y_B = \sum_{n=1}^{N} \left[ (\mathbf{x}_B)_n \cdot \frac{\partial (\mathbf{y}_F)_n}{\partial x_F} \right] = (\mathbf{x}_B)_0 + (\mathbf{x}_B)_1 + \ldots + (\mathbf{x}_B)_N \,. \tag{20}$$

Сопоставляя результаты анализа для сумматора и разветвителя можно сделать следующее пояснение. В отношении прямого и обратного прохождения сигналов сумматор и разветвитесь имеют дуальную природу, а именно: при обратном прохождении разветвитель выполняет роль сумматора для обратно распространяющегося сигнала ошибки, а сумматор – роль разветвителя.

Классический базовый элемент сети, искусственный нейрон, может быть построен в виде композиции простейших элементов, как показано на рис. 3 на примере нейрона с тремя входами. Такое построение сложного элемента связывает между собой соответствующие входы и выходы простых элементов, обеспечивая распространение сигналов в прямом и обратном направлениях.

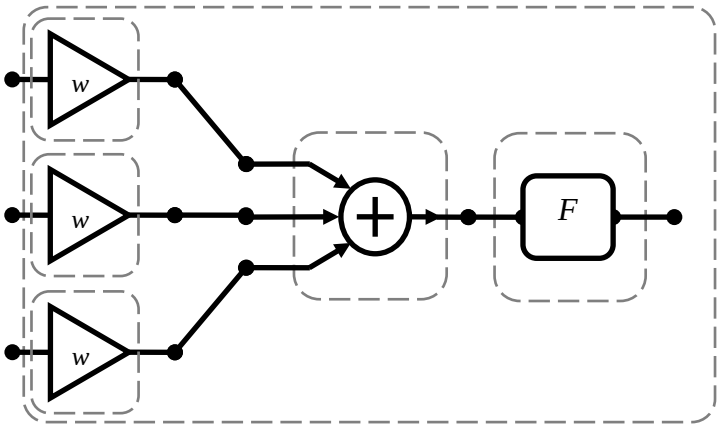



Рис 3. Классический нейрон как композиция простейших элементов.

Детальное представление нейрона в форме такой декомпозиции позволяет отслеживать сигналы внутренних связей, что может быть целесообразно тогда, когда используются сети из малого числа нейронов, и в том случае, когда каждый из входных сигналов имеет определенный физический смысл, который необходимо учитывать при построении модели.

В том случае, когда внутренние сигналы нейрона не представляют интереса, весь нейрон целиком может быть представлен адаптивным элементом с $N$ входами и одним выходом. Вектор адаптируемых параметров нейрона $\boldsymbol{\theta}$ будет описывать упорядоченный набор коэффициентов адаптивных умножителей:

$$\theta = \{w_1, w_2, \ldots, w_N\}. \tag{21}$$

Представленный способ синтеза может быть распространен и на синтез более сложных адаптивных элементов с векторными входами и выходами, например, для матричного умножителя и нейронного слоя. Однако в настоящей работе синтез был ограничен пятью элементами, представленными на рис. 2 и рис. 3.

## 4. Реализация

Структура сети, организованная на основе адаптивных элементов, может быть реализована в рамках парадигмы объектно-ориентированного программирования. Для реализации прототипа программного обеспечения в качестве языка программирования выбран высокоуровневый язык программирования общего назначения *Python*, предоставляющий с одной стороны простую форму записи математических выражений, с другой – широкие возможности в области объектно-ориентированного программирования. Кроме этого данный язык обладает свободной лицензией и большой библиотекой готовых разработок для решения ряда вспомогательных задач моделирования.

Программная реализация разделена на две части – базовую и дополнительную. Базовая часть выполнена в качестве пакета и включает необходимую для работы приложения иерархию классов. Дополнительная часть предназначена для работы в интерактивном режиме в составе системы компьютерной математики *Sage* [9], представляющей собой пакет программ со свободной лицензией, объединенный единым пользовательским интерфейсом. Дополнительная часть использует возможности *Sage* для ввода и вывода данных: генерация обучающих последовательностей, отображение графов, построение графиков и таблиц. Отметим ещё одно преимущество системы *Sage*, состоящее в том, что она распространяется под свободной лицензией.

Иерархия классов базовой части показана на рис. 4 с использованием нотации универсального языка моделирования *UML* [10, 11]. Основным рабочим классом при работе с пакетом является класс *NeuralNetwork,* объекты которого непосредственно представляют нейронную сеть. Другим важным классом является абстрактный класс *Node,* который описывает простейший элемент сети и требует реализации в своих потомках необходимых методов прямого и обратного прохождения. Наследниками класса *Node* являются классы, реализующие простейшие элементы, описанные ранее: адаптивный умножитель, безынерционный функциональный преобразователь, разветвитель и сумматор. Эти классы используются в классе *Neuron,* представляющим собой классический нейрон. Его поведение при прямом и обратном прохождении определяется функцией активации, а так же ее первой производной. Исходный класс *Neuron* – абстрактный класс, пакет базовой части представляет несколько его наследников, которые реализуют наиболее широко распространенные функции активации, а именно нейроны с гауссовой функцией активации, линейной и сигмоидной функциями.

Для выполнения моделирования необходимо создать экземпляр класса *NeuralNetwork,* необходимое количество элементов (например, нейронов), установить связи между ними и обучить нейронную сеть. Обучение сети выполняется при помощи метода *Train* класса *NeuralNetwork*, на вход которого поступает объект типа *TrainingSet.* Данный объект содержит в себе всю необходимую информацию для обучения, а именно: набор шаблонов обучения, выбранный метод обучения и набор критериев, по которым определяется необходимость прервать обучение. Набор критериев остановки обучения представлен экземпляром класса *TrainingLimits*, поля которого представляют ограничения на количество эпох, время на обучение и требуемую производительность сети (минимальную СКО).

Представленные в следующем разделе настоящей работы результаты численного моделирования – это данные, полученные при помощи базовой части разработанного

программного прототипа и визуализированные при помощи дополнительной части. Дополнительная часть предлагает вместо класса *NeuralNetwork* использовать класс-наследник *VisualNeuralNetwrok*, в котором был разработан ряд методов для визуализации данных в удобной форме. Отображение результатов: построение графиков функций, связных графов и т.д. – происходит при помощи пакета компьютерной математики *Sage* и входящих в него компонентов, например, такого как *matplotlib*.

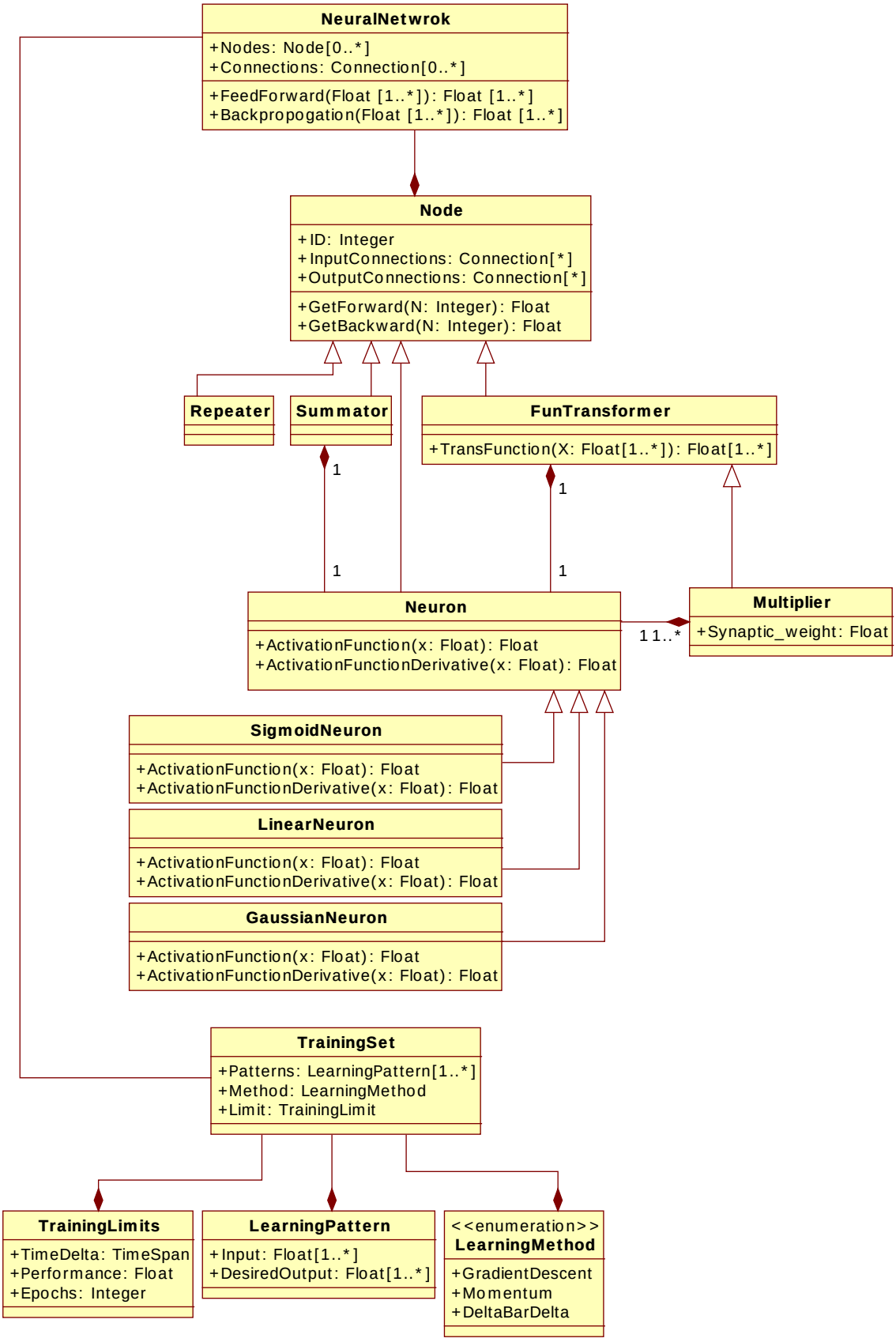



Рис. 4 Иерархия классов базовой части программного прототипа.

(Диаграмма построена при помощи пакета StarUML [11] со свободной лицензией)

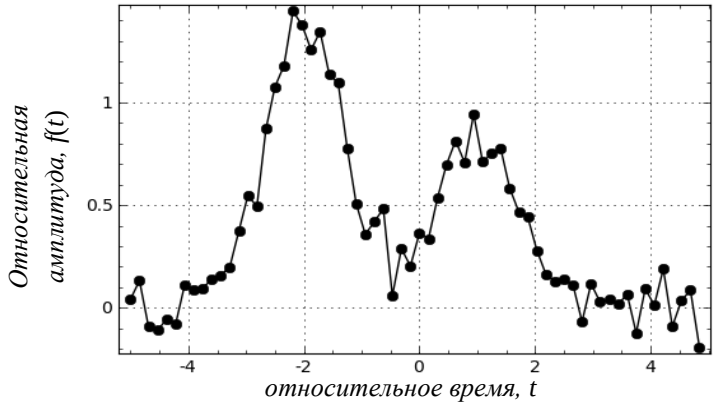
## 5. Экспериментальные результаты

Разработанный программный прототип был использован для численного моделирования применения нейронной сети для решения задачи аппроксимации и классификации входных данных.

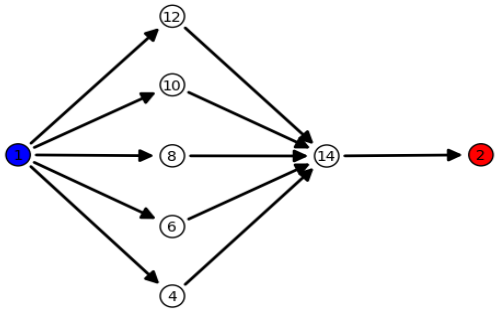
В качестве входного сигнала в задаче аппроксимации была использована аддитивная смесь суммы импульсов в форме функции Гаусса и шума:

$$f(t) = \sum_{i=0}^{M} \left\{ A_i \frac{1}{\sqrt{2\pi} \cdot \rho_i} \cdot \exp\left( -\frac{(t-\tau_i)^2}{2\rho_i^2} \right) \right\} + n(t), \qquad (22)$$

где $A_i$ – амплитуда импульса, $\rho_i$ – параметр ширины импульса, $\tau_i$ – временная задержка прихода импульса, $M$ – количество импульсов, $n(t)$ – белый гауссовский шум. Практическое использование модель такого сигнала находит, например, в задачах идентификации точечных рассеивателей в сверхкороткоимпульсной радиолокации [12, 13].




а)




б)

Рис 5. а) входные данные для решения задачи аппроксимации,
б) конфигурация нейронной сети, построенная прототипом ПО.

Входные данные показаны на рис. 5, а) точками, которые аппроксимированы ломанными для полноты восприятия. На рис. 5, б) представлен граф нейронной сети, построенный непосредственно с помощью средств разработанного программного прототипа. Номера в кружках на рисунке соответствуют порядковым идентификационным номерам элементов в списке и выводятся в качестве отладочной информации.

Изображенная на рис. 5, б) конфигурация сети фактически отображает структуру, собранную из разветвителей и нейронов. Последние в свою очередь построены на основе простых адаптивных элементов, рассмотренных ранее. Подробно данная сеть представлена на рис. 6, где пунктирными блоками выделены основные элементы.

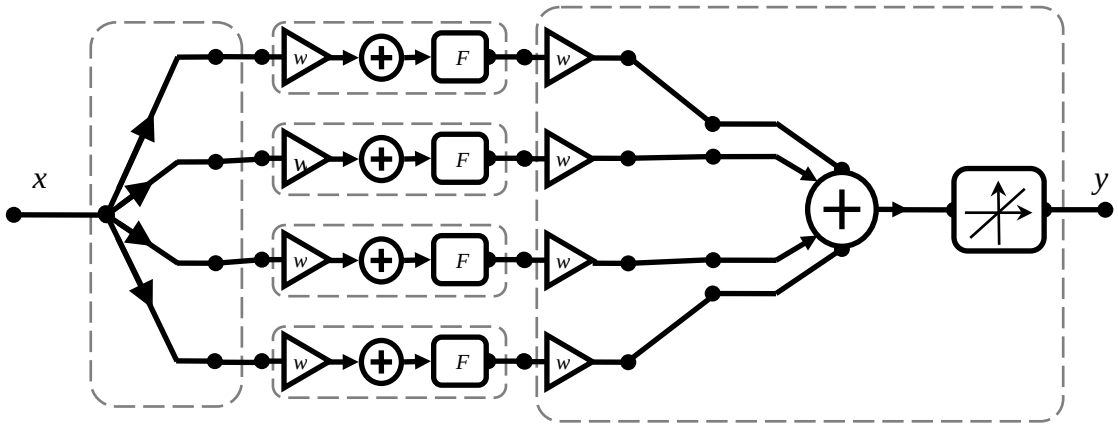



Рис 6. Структура сети в задаче одномерной аппроксимации.

Для обучения сети применялись методы первого порядка, кратко рассмотренные ранее, а именно: метод простого градиентного спуска, метод спуска с инерцией и *Delta-Bar-Delta*. Параметры методов выбирались в процессе исследования различными, но в настоящей статье представлены только некоторые из них.

Зависимость среднеквадратической ошибки (СКО) за время обучения для метода градиентного спуска показана на рис. 7, как видно из рисунка, увеличение исходного коэффициента скорости обучения с 0,3 (непрерывная линия) до 0,6 (пунктирная линия) позволяет несколько ускорить процесс обучения и добиться лучших результатов. Однако дальнейшее увеличение коэффициента скорости обучения приводит к слишком быстрой корректировке синаптических весов. Это приводит к тому, что веса осциллируют вокруг оптимальных значений, не достигая их, – аналогичный колебательный характер принимает и СКО.

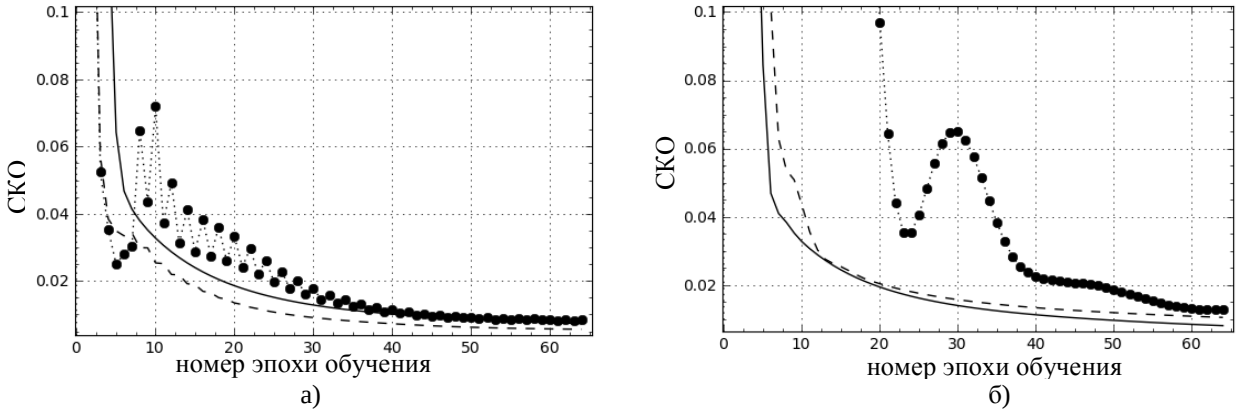
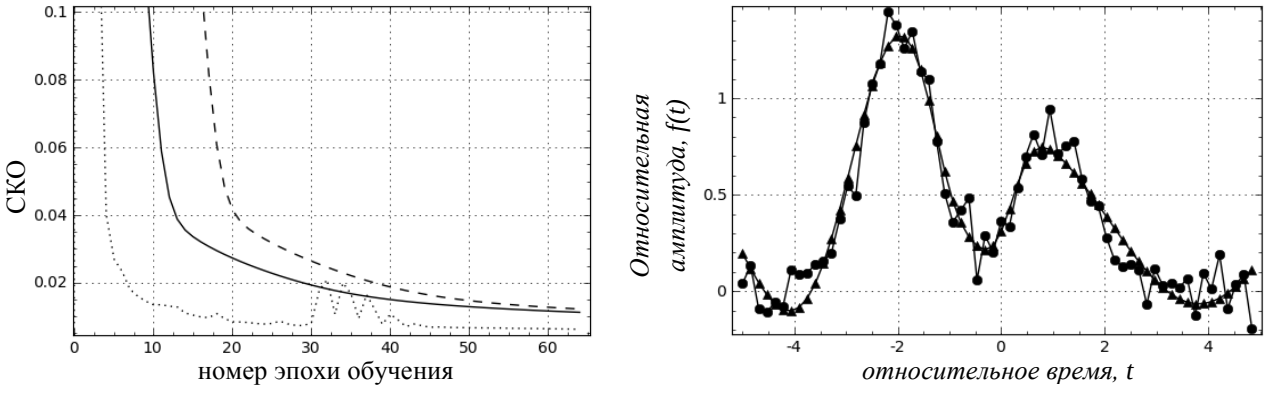



Рис 7. а) Метод градиентного спуска с коэффициентом скорости обучения: 0,3 для непрерывной линии, 0,6 для пунктирной линии, 0,9 для точечной линии, б) Метод градиентного спуска с коэффициентом скорости обучения 0,3 и коэффициентом инерции (momentum) 0,3 для непрерывной линии, 0,6 для пунктирной линии, 0,9 для точечной линии.

Для демонстрации эффекта инерции выбран коэффициент скорости обучения 0,3, затем проведено обучение сети при трех различных коэффициентах инерции. Эти результаты показаны на рис. 7. Увеличение коэффициента инерции приводит к незначительному изменению скорости обучения, причем дальнейшее увеличение вплоть до 0,9 приводит к негативному эффекту – дестабилизации процесса обучения сети в целом.

Наибольший интерес представляет метод *Delta-Bar-Delta,* результаты применения которого показаны на рис. 8, а). В [8] высказано предположение, что в большинстве случаев оптимальным набором параметров для данного метода являются следующие значения параметров формул (10, 11): $\gamma = 0{,}3$, $\varphi = 0{,}7$, $\kappa = 0{,}5$, которым на рис. 8 соответствует непрерывная линия. Как видно из этого графика, отклонение параметров от оптимальных значений могут в конкретном случае приводить как к положительным, так и к отрицательным результатам. В случае чрезмерного уменьшения коэффициента скорости обучения (пунктирная линия) наблюдается значительное отставание в обучении. Уменьшение коэффициента $\gamma$, определяющего степень влияния градиентов ошибки, полученных в предыдущих эпохах и, одновременное, увеличение коэффициента $\kappa$, и, как следствие, ускорение роста коэффициента скорости, позволяет получить значительное улучшение качества обучения.

В зависимости от использованного метода обучения за некоторое количество итераций синаптические веса нейронной сети приближаются к оптимальным значениям. На рис. 8, б) показан график аппроксимации входного сигнала после завершения процесса обучения с использованием метода *Delta-Bar-Delta* с оптимальными параметрами.




а) б)

Рис 8. а) Изменение СКО по мере обучения сети для метода *Delta-Bar-Delta* при различных параметрах обучения: $\gamma=0.7$, $\kappa=0.01$, $\varphi=0.5$ – непрерывная линия, $\gamma=0.7$, $\kappa=0.01$, $\varphi=0.7$ – пунктирная линия, $\gamma=0.3$, $\kappa=0.7$, $\varphi=0.5$ – точечная линия, б) результат аппроксимации.

Под классификацией будем понимать процедуру отнесения объекта (одного примера входных данных) к одному из двух или более классов. Для демонстрации применения нейронных сетей для решения этой задачи в работе рассматривается двумерный случай классификации. Практическое приложение этой задачи следующее. Посредством квадратурного демодулятора наблюдается один из двух узкополосных случайных процессов (СП). Известно, что плотность вероятности каждого из процессов описывается выражением:

$$p(i,q) = \frac{1}{\sqrt{2\pi}\sigma_I\sigma_Q}\exp\left\{-\left[\frac{(i-m_I)^2}{2\sigma_I^2}+\frac{(q-m_Q)^2}{2\sigma_Q^2}\right]\right\}, \qquad (23)$$

где $\sigma_I$ , $\sigma_Q$ – дисперсии, а $m_I$ , $m_Q$ – математические ожидания составляющих СП.

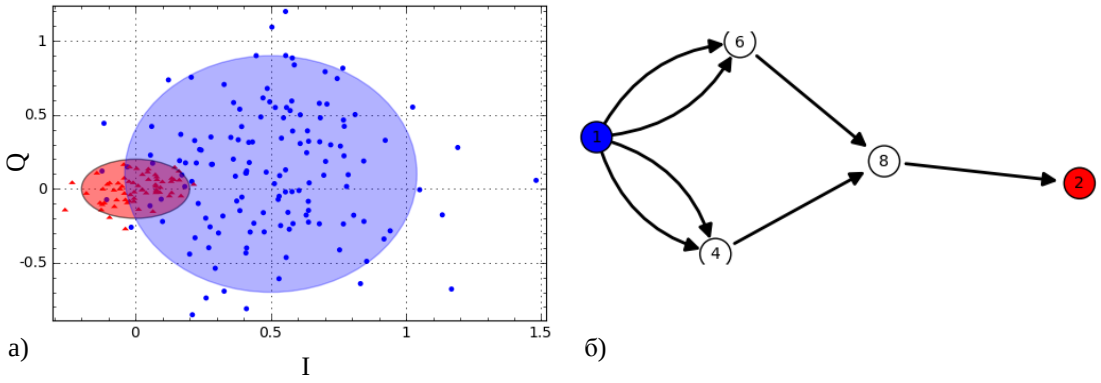



Рис 9. а) данные двух классов, б) нейронная сеть для решения задачи классификации.

На координатной плоскости $\{I, Q\}$ размещаются точки, принадлежащие одному из двух различаемых классов, соответствующих двум случайным процессам, параметры выражения (23) для которых различны. Сгенерированные таким образом примеры показаны на рис. 9, а).

В качестве входных данных для нейронной сети выступают координаты точки. Выход сети должен определять принадлежность точки к первому или второму классу. Для двумерного случая можно создать нейронную сеть с двумя входными нейронами, в скрытом слое которой размещаются два нейрона с сигмоидными функциями активации. Выходной нейрон сети так же имеет сигмоидную функцию активации, что необходимо для получения ограниченного выходного сигнала – индикатора класса. Созданная нейронная сеть показана на рис. 10, а её представление – на рис. 9, б).

Предполагается, что выходной сигнал сети будет близок к нулю, если точка принадлежит к классу «А» и близок к единице, если точки принадлежит к классу «В». Итоговое решение о принадлежности точки к классу принимается по уровню 0,5. Качество работы сети оценивается отдельно для каждого класса как доля успешного определения принадлежности к этому классу точки.

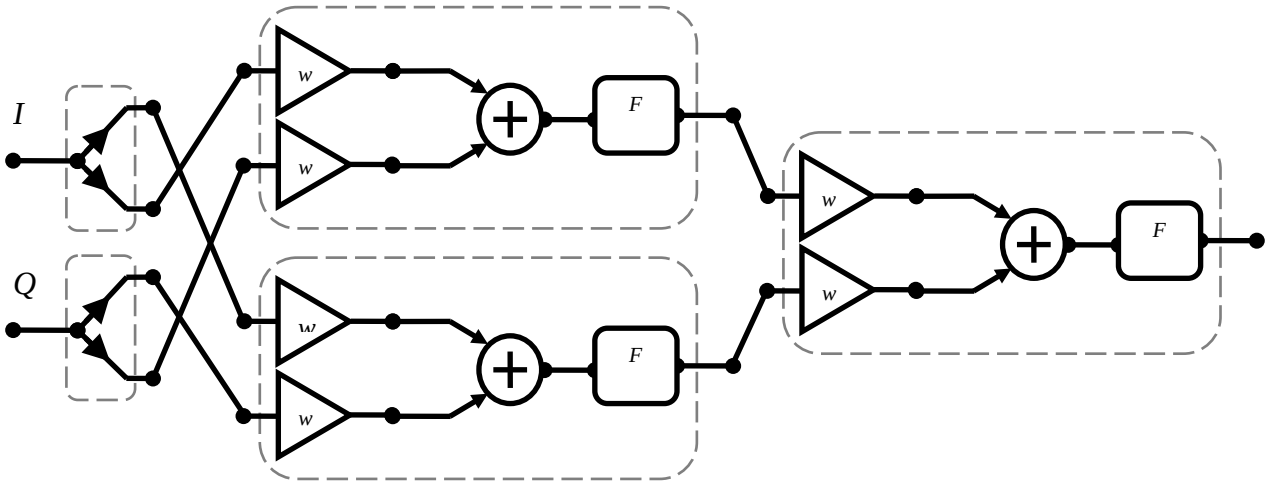



Рис 10. Структура сети в задаче двумерной классификации.

Обучение сети проведено методом *Delta-Bar-Delta* с коэффициентом скорости обучения 0.4, без коэффициента инерции и параметрами $\gamma$ =0.3, $\varphi$=0.7, $\kappa$ =0.5. В процессе обучения, после каждой эпохи оценивалось качество работы сети, при этом использовалось два отдельных набора данных, сгенерированных по тому же закону.

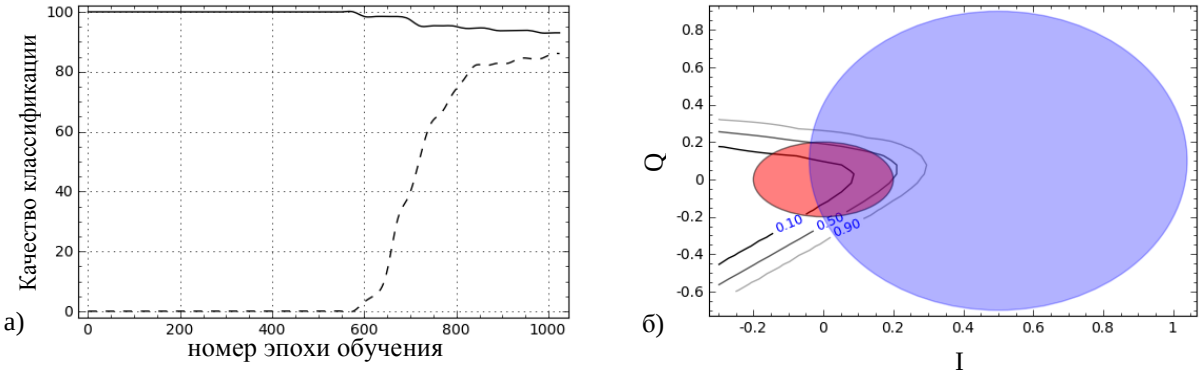



Рис 11. а) изменение качества классификации в процессе обучения сети: непрерывная линия – точки класса A, пунктирная – класса B, б) граничная кривая между точками классов A и B – изолинии соответствуют подписанным на них значениям выходного сигнала.

На рис. 11, а) показано изменение качества работы сети в процессе обучения. Видно, что вначале обучения все точки плоскости классифицировались как точки, принадлежащие к классу «A», следовательно, все точки, действительно принадлежащие к этому классу, были распознаны корректно. Однако для точек класса «B» ситуация была иной – все точки этого класса распознавались неверно и относились к классу «A». Это несоответствие порождало сигнал ошибки, который был использован для корректировки синаптических весов, что в свою очередь приводило к изменению выходного сигнала сети. В процессе обучения сеть стала правильно классифицировать большую часть данных тестового набора. Однако, как видно из рис. 11 б), носители распределений вероятностей, по которым генерировались

данные, пересекаются. Это пересечение обуславливает неизбежную ошибку в работе сети, так что некоторые точки по-прежнему распознаются некорректно. На рис. 11, б) показана итоговая граница по уровню 0,5, которая принята при классификации. Подробное теоретическое решение этой задачи методом оптимального Байесовского классификатора представлено в монографии [4].

## 6. Заключение

В настоящей работе представлена методика синтеза нейронной сети на основе простых адаптивных элементов. Был разработан прототип программного обеспечения, реализующий адаптивные элементы в рамках объектно-ориентированного подхода. Была разработана специализированная библиотека классов на языке Python для работы в среде Sage. В работе представлены численные эксперименты, проведенные с данным ПО. Примеры включают в себя обучение сетей, предназначенных для решения задач аппроксимации и классификации. Разработанный прототип ПО позволяет визуализировать все необходимые зависимости в процессе обучения и строить графы сетей.

Одним из преимуществ предложенного подхода к синтезу нейронной сети является возможность строить масштабируемые сети, включающие в свою структуру элементы, производящие требуемые функциональные преобразования. Такие сети могут быть организованы иерархически в виде подсетей, для которых могут быть определены свои собственные правила обучения и структура связей их элементов. Дальнейшее развитие данной методики состоит в расширении списка адаптируемых элементов, включение в него элементов с многомерными входами и выходами. Также большой практический интерес, по мнению авторов, представляет использование в адаптивных элементах методов второго порядка [7], которое, однако, потребует некоторого расширения и последующего обобщения рассмотренных подходов.

## Библиографический список


1. Д. Рутковская, М. Пилиньский, Л. Рутковский, Нейронные сети, генетические алгоритмы и нечёткие системы: Пер. с польск. – М.: Горячая линия – Телеком, 2006, 452 стр.
2. M. H. Hassoun, Fundamentals of Artificial Neural Networks, The MIT Press, 1995, 511 p.
3. А. И. Галушкин, Нейронные сети. Основы теории, М.: Горячая линия – Телеком, 2010, 496 стр.

4. С. Хайкин, Нейронные сети: полный курс, 2-е изд., испр. : Пер. с англ. – М.: ООО «И.Д. Вильямс», 2006, 1104 стр.
5. K. S. Narendra, K. Parthasarathy, Identification and Control of Dynamical Systems Using Neural Networks // IEEE Transactions on Neural Networks, Vol. 1, No. 1, March 1990, pp. 4–27.
6. M.T. Hagan, H.B. Demuth, M.H. Beale, Neural Network Design, Martin Hagan, 2002, 736 p.
7. R. Battiti, First- and Second-Order Methods for Learning: Between Steepest Descent and Newton's Method // Neural Computation, MIT, Vol. 4, No. 2, March 1992, pp. 141–166
8. S. Samarasinghe, Neural Networks for Applied Sciences and Engineering: From Fundamentals to Complex Pattern Recognition, Auerbach Publications; 1 ed., 2006, 570 p.
9. Sage: Open source mathematics software, Официальный сайт, http://www.sagemath.org/
10. Г. Буч, Дж. Рамбо, А. Джекобсон, Язык UML. Руководство пользователя, Изд-во: «ДМК Пресс», 2007, 496 стр.
11. StarUML – The Open Source UML/MDA Platform, Официальный сайт, http://staruml.sourceforge.net/en/
12. Коновалюк М.А., Кузнецов Ю.В., Баев А.Б., Определение параметров многоточечных целей по спектру радиолокационного изображения Вестник МАИ, том 17, № 3, стр. 193-198, 2010 г.
13. Коновалюк М.А., Горбунова А.А., Кузнецов Ю.В., Баев А.Б., Алгоритм извлечения информации из комплексного радиолокационного изображения сложной цели, 4-я всероссийская конференция «Радиолокация и радиосвязь», Москва, ИРЭ им. ак. В.А. Котельникова РАН, декабрь 2010 г.

## Сведения об авторах

ЕФИМОВ Евгений Николаевич, студент Московского авиационного института (национального исследовательского университета).
МАИ, Волоколамское ш., 4, Москва, А-80, ГСП-3, 125993;
тел.: (499) 158-40-47;
e-mail: omegatype@gmail.com

ШЕВГУНОВ Тимофей Яковлевич, доцент Московского авиационного института (национального исследовательского университета), к.т.н.
МАИ, Волоколамское ш., 4, Москва, А-80, ГСП-3, 125993;
тел.: (499) 158-40-47;
e-mail: shevgunov@mai-trt.ru