ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА

Лекции по курсу «Интеллектуальные системы»

Для студентов направления 090401

Очное обучение 2024-2025 учебный год

Лектор: Солдатова Ольга Петровна, к.т.н., доцент

Основные понятия нейронных сетей. Персептроны.

Лекция 3

Глубокое обучение.
Основные свойства нейронных сетей.
Биологические основы.
Модели искусственных нейронов.
Функции активации нейронов.
Однослойный персептрон.
Многослойный персептрон.
Структура двухслойной сети.
Теорема Цыбенко.
Градиентные алгоритмы обучения.
Алгоритм градиентного спуска.
Метод обратного распространения ошибки.
Эвристические алгоритмы.
Стохастические алгоритмы.
Подбор архитектуры многослойного персептрона.

Литература по курсу

1. Куприянов А.В. «Искусственный интеллект и машинное обучение» https://do.ssau.ru/moodle/course/view.php?id=1459
2. Осовский С. Нейронные сети для обработки информации / Пер. с пол. И.Д. Рудинского. – М.: Финансы и статистика, 2002. – 344 с.: ил.
3. Горбаченко, В. И. Интеллектуальные системы: нечеткие системы и сети : учебное пособие для вузов / В. И. Горбаченко, Б. С. Ахметов, О. Ю. Кузнецова. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2018. — 105 с. — (Университеты России). — ISBN 978-5-534-08359-0. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/424887 – Режим доступа: https://urait.ru/bcode/424887
4. 4. Гафаров Ф.М. Искусственные нейронные сети и приложения: учеб. пособие / Ф.М. Гафаров, А.Ф. Галимянов. – Казань: Изд-во Казан. ун-та, 2018. – 121 с. – Текст : электронный. – Режим доступа: https://repository.kpfu.ru/?p_id=187099
5. Хайкин С. Нейронные сети: Полный курс: Пер. с англ. - 2-е изд. – М.: Вильямс, 2006. – 1104 с.: ил.
6. Борисов В.В., Круглов В.В., Федулов А.С. Нечёткие модели и сети. – М.: Горячая линия– Телеком, 2007. -284 с.: ил.
7. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечёткие системы: Пер. с польск. И.Д.Рудинского, - М.: Горячая линия – Телеком, 2007. – 452 с. ил.
8. Николенко С., Кадурин А., Архангельская Е. Глубокое обучение. — СПб.: Питер, 2018. — 480 с.: ил. — (Серия ≪Библиотека программиста≫).
9. Гудфеллоу Я., Бенджио И., Курвилль А. Глубокое обучение / пер. с анг. А. А. Слинкина. – 2-е изд., испр. – М.: ДМК Пресс, 2018. – 652 с.: цв. ил.

Нейросетевые модели

Глубокое обучение

Глубокое обучение — это разновидность машинного обучения на основе искусственных нейронных сетей. *Процесс обучения* называется *глубоким*, так как структура искусственных нейронных сетей состоит из нескольких входных, выходных и скрытых слоев. Каждый слой содержит единицы, преобразующие входные данные в сведения, которые следующий слой может использовать для определенной задачи прогнозирования. Благодаря этой структуре компьютер может обучаться с помощью собственной обработки данных.

Типичным примером модели глубокого обучения является глубокая сеть прямого распространения, или **многослойный персептрон** (MLP). Многослойный персептрон – это просто математическая функция, отображающая множество входных значений на множество выходных. Эта функция является композицией нескольких более простых функций, каждая из которых реализуется в отдельном слое. Каждое применение одной математической функции можно рассматривать как новое представление входных данных. Каждый слой можно представить как состояние памяти компьютера после параллельного выполнения набора инструкций. Чем больше глубина сети, тем больше инструкций она может выполнить последовательно, поскольку более поздние инструкции могут обращаться к результатам выполнения предыдущих.

Глубокое обучение

Один из способов оценить глубину архитектуры сети основан на подсчёте числа последовательных инструкций, которые необходимо выполнить. Можно считать, что это длина самого длинного пути в графе, описывающего вычисление каждого выхода модели по ее входам. Одна и та же функциональность может быть изображена графами с разной длиной пути в зависимости от того, какие функции допускаются в качестве шагов.

Человеческий мозг представляет собой чрезвычайно **сложный, нелинейный, параллельный компьютер**. Он обладает способностью организовывать свои структурные компоненты, называемые нейронами, так, чтобы они могли выполнить конкретные задачи (такие как распознавание образов, обработку сигналов органов чувств, моторные функции) во много раз быстрее, чем могут позволить самые быстродействующие компьютеры. Мозг имеет совершенную структуру, позволяющую строить собственные правила на основе опыта. Опыт накапливается с течением времени.

Понятие развития нейронов мозга связано с понятием **пластичности мозга** – способностью настройки нервной системы в соответствии с окружающей средой. Аналогично в искусственных нейронных сетях производится настройка искусственных нейронов и формируется структура нейронной сети. В общем случае нейронная сеть представляет машину, моделирующую способ обработки мозгом конкретной задачи. Эта сеть обычно реализуется с помощью электронных компонентов или моделируется программой.

Основные свойства нейронных сетей

Таким образом, можно дать следующее определение нейронных сетей, выступающих в роли адаптивной машины:

Нейронная сеть – это громадный распределенный параллельный процессор, состоящий из элементарных единиц обработки информации, накапливающих экспериментальные знания и представляющих их для последующей обработки. Нейронная сеть сходна с мозгом с двух точек зрения:

1. **Знания поступают в нейронную сеть из окружающей среды и используются в процессе обучения;**
2. **Для накопления знаний применяются связи между нейронами, называемые синаптическими весами.**

Основные свойства нейронных сетей

Наиболее существенными свойствами нейронных сетей являются:

- ✓ **Нелинейность.** Поскольку искусственные нейроны могут быть линейными и нелинейными, то нейронные сети позволяют воспроизводить сложные зависимости, как линейные, так и нелинейные. Нейронные сети реализуют нелинейность особого вида, так как она распределена по сети.
- ✓ **Параллельная обработка информации.** Благодаря этой способности при большом количестве межнейронных связей достигается значительное ускорение процесса обработки информации.
- ✓ **Обучение на примерах.** Одной из популярных парадигм обучения является *обучение с учителем*, то есть на основе набора *учебных примеров*. Каждый пример состоит из входного сигнала и соответствующего ему *ожидаемого* выходного сигнала. Нейронная сеть модифицирует синаптические веса для минимизации разности ожидаемого выходного сигнала и *реального* выходного сигнала нейронной сети. Учебные примеры могут быть использованы для обучения снова в таком же или ином порядке.
- ✓ **Адаптивность (adaptivity).** Нейронные сети обладают способностью адаптировать свои синаптические веса к изменениям окружающей среды, то есть нейронные сети могут быть легко переучены.

Основные свойства нейронных сетей

- ✓ **Нечувствительность к ошибкам (faulttolerance).** Очень большое количество межнейронных соединений приводит к тому, что сеть становится нечувствительной к ошибкам, возникающим в отдельных контактах.
- ✓ **Способность к обобщению полученных знаний.** Натренированная на ограниченном множестве обучающих примеров, она обобщает накопленную информацию и вырабатывает ожидаемую реакцию применительно к данным, не обрабатывавшимся в процессе обучения.
- ✓ **Единообразие анализа и проектирования.** Нейронные сети являются универсальным механизмом обработки информации. Одно и тоже проектное решение нейронной сети может быть использовано в разных предметных областях. Это свойство проявляется из-за нескольких причин:
  - нейроны являются стандартными составными частями любой нейронной сети;
  - можно использовать одни и те же алгоритмы обучения в различных нейросетевых приложениях;
  - на основе интеграции целых модулей могут быть построены модульные сети.

Основные свойства нейронных сетей

- ✓ **Аналогия с нейробиологией.** Строение нейронных сетей определяется аналогией с живым мозгом, являющимся доказательством возможности существования отказоустойчивых вычислительных параллельных систем, эффективно решающих поставленные задачи.

Наличие перечисленных свойств вызвало в последние годы огромный рост интереса к нейронным сетям и существенный прогресс в их исследовании.

Искусственные нейронные сети используются для аппроксимации функций, сжатия данных, классификации и распознавания образов, прогнозирования, идентификации, оценивания и ассоциативного управления.

Биологические основы нейронных сетей

Искусственные нейронные сети возникли на основе знаний о функционировании нервной системы живых существ. Нервную систему человека можно представить в виде трёхступенчатой системы.

Центром этой системы является ***мозг***, представляемый сетью нервных клеток, то есть нейронной сетью. ***Рецепторы*** получают информацию от тела и окружающей среды и преобразуют её в электрический импульс, передаваемый в мозг. ***Эффекторы*** преобразовывают электрические импульсы, вырабатываемы мозгом в выходные сигналы.

Нервная клетка является основным элементом нервной системы. У нейрона есть тело (сома), внутри которого располагается ядро. Из сомы нейрона выходят отростки двух видов: тонкие, густо ветвящиеся дендриты и более толстый, расщепляющийся на нервные окончания – колатералы, аксон (рисунок 1).

Выходной сигнал клетки передается через аксон. Колатералы контактируют с сомой и дендритами других нейронов, образуя каналы связи (синапсы) выходных сигналов клетки с входами других клеток. Синапсы могут находиться как на дендритах, так и непосредственно в теле клетки.

Биологические основы нейронных сетей

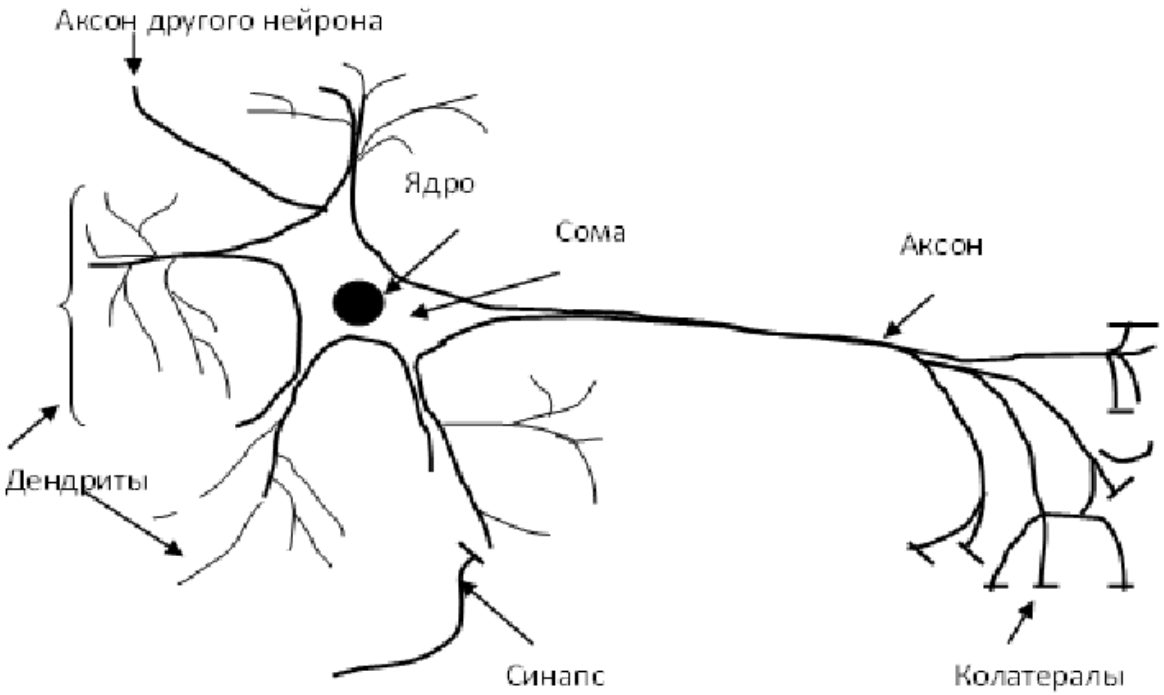
Передача сигналов внутри нервной системы – это очень сложный электрохимический процесс. С большим упрощением можно считать, что передача нервного импульса между двумя клетками основана на выделении особых химических субстанций, называемых ***нейромедиаторами***, которые формируются под влиянием поступающих от синапсов раздражителей. Предсинаптический процесс формирует субстанцию, которая методом диффузии передаётся по синаптическим связям и влияет на постсинаптический процесс. Синапс преобразует предсинаптический электрический сигнал в химический, а затем в постсинаптический - электрический.

Данные субстанции воздействуют на клеточную мембрану, вызывая изменение ее энергетического потенциала, причем величина этого изменения пропорциональна количеству нейромедиатора, попадающего на мембрану.

Синапсы отличаются друг от друга размерами и возможностями концентрации нейромедиатора.

Поэтому импульсы одинаковой величины, поступающие на входы нервной клетки через различные синапсы, могут в разной степени изменять ее энергетический потенциал. Мерой изменения потенциала считается уровень поляризации мембраны, зависящий от суммарного количества нейромедиатора, выделенного на всех синапсах.

Биологические основы нейронных сетей (рисунок 1)



Биологические основы нейронных сетей

В результате поступления входных импульсов происходит изменение электрического потенциала клетки. Если отклонение от состояния электрического равновесия невелико, клетка возвращается в исходное состояние и на ее выходе сигнал не регистрируется. В этом случае считается, что уровень изменения потенциала ниже порога ее срабатывания. Если суммарное изменение потенциала превысило порог активации клетки, значение выходного сигнала начинает нарастать, формируя нервный импульс, пересылаемый аксоном на другие нейроны. Величина этого сигнала не зависит от степени превышения порога срабатывания.

Количество взаимодействующих друг с другом нервных клеток в человеческом мозге оценивается, как 10^{11} - 10^{14}. Каждая нервная клетка выполняет функцию суммирования весовых коэффициентов входных сигналов и сравнивает полученную сумму с пороговым значением. Каждый нейрон имеет свои веса и свои пороговые значения. Громадное количество нейронов и межнейронных связей (до 1000 входов в каждый нейрон) приводит к тому, что ошибка в срабатывании отдельного нейрона остается незаметной в общей массе взаимодействующих клеток.

Существует огромное количество форм и размеров нейронов в зависимости от того, в какой части мозга он находятся. Самыми распространёнными нейронами коры головного мозга являются пирамидальные нейроны.

Модель искусственного нейрона МакКаллока – Питса

Одна из первых моделей искусственного нейрона была предложена Дж. МакКаллоком и У. Питсом в 1943 году. Структурная схема этой модели представлена на рисунке 2. В данной модели выходной сигнал принимает двоичные значения: 0 или 1. Значение 1 соответствует превышению порогового уровня, значение 0 – в противном случае.

Сигналы x_j на входе синапсов j $(j = 1,2,\ldots,N)$, суммируются с учетом соответствующих синаптических весов w_j, после чего результат сравнивается с пороговым значением w_0 .

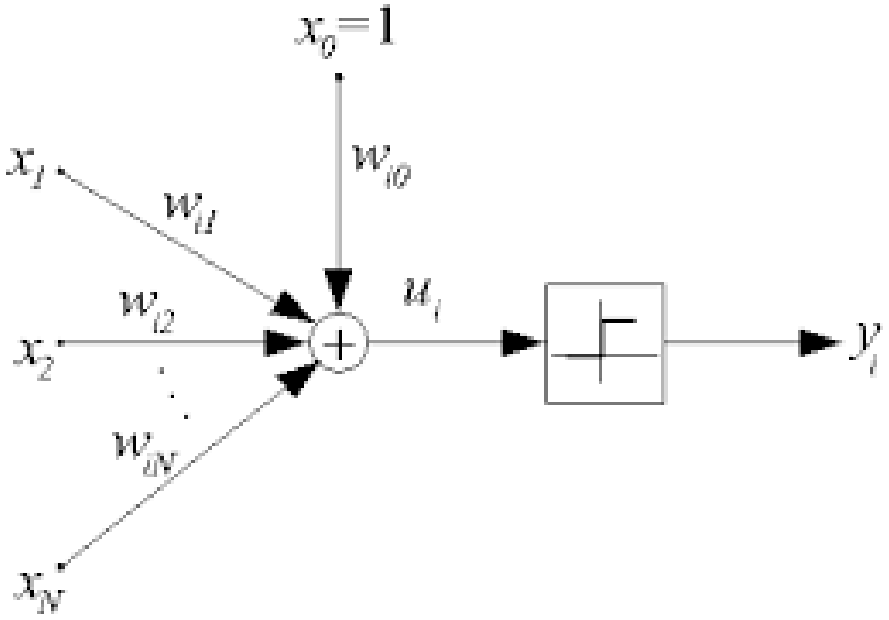
Пороговое значение отражает увеличение или уменьшение входного сигнала, подаваемого на функцию активации, которая ограничивает амплитуду выходного сигнала. Выходной сигнал нейрона y определяется при этом зависимостью:

$$y = f\left(\sum_{J=1}^{N} w_j x_j(t) + w_0\right) \qquad (1)$$

Аргументом функции выступает суммарный сигнал, формируемый сумматором искусственного нейрона.

$$u_i = \sum_{j=1}^{N} w_{ij} x_j(t) + w_{i0} \qquad (2)$$

Модель нейрона МакКаллока – Питса (рисунок 2)



Модель нейрона МакКаллока – Питса

Коэффициенты в формуле (1) представляют веса синапсов. Использование порогового сигнала обеспечивает эффект аффинного преобразования выхода линейного сумматора .

Модель МакКаллока – Питса – это дискретная модель, в которой состояние нейрона в момент (t+1) рассчитывается по значению его входных сигналов в момент времени t.

Функция $f(u)$ называется функцией активации. В модели МакКаллока – Питса это пороговая функция вида:

$$f(u) = \begin{cases} 1 \ для\ u > 0 \\ 0 \ для\ u \le 0 \end{cases} \qquad (3)$$

В общем случае эта функция активации описывается следующим выражением:

$$\mathrm{sgn}(x) = \begin{cases} b, если\ x > 0; \\ c, если\ x \le 0, \end{cases} \qquad (4),$$

где b и c – некоторые постоянные. На практике чаще всего используют две пары постоянных b и c: первая (-1,1); вторая – (0,1). Первая пара коэффициентов определяет так называемую симметричную пороговую функцию, вторая – смещенную.

Персептрон

Ф. Розенблатт в 1958 году ввел понятие персептрона как первой модели обучения с учителем. Перспетрон это, по сути, нейрон МакКаллока-Питса, для которого Розенблатт впервые предложил алгоритм обучения. Обучение персептрона требует наличие учителя и состоит в таком подборе весов, чтобы выходной сигнал был наиболее близок к заданному значению. Каждому обучающему примеру, представленному вектором x поставлено в соответствии ожидаемое значение на выходе i-го нейрона.

Наиболее популярный метод обучения персептрона, называемый **правилом персептрона**, состоит в подборе весовых коэффициентов по следующему алгоритму:

1. При первоначально выбранных (как правило, случайным образом) значениях весов на вход нейрона подается обучающий вектор x и рассчитывается значение выходного сигнала y_i . По результатам сравнения значения с заданным значением d_i уточняются значения весов;
2. Если y_i совпадает с ожидаемым значением d_i , то весовые коэффициенты w_{ij} не изменяются;
3. Если y_i =0 , а соответствующее значение d_i =1 , то значения весов уточняются по формуле $w_{ij}(t+1)=w_{ij}(t)+x_j$, где $(t+1)$– это номер текущего цикла, а t – номер предыдущего цикла;
4. Если y_i=1 , а соответствующее значение d_i =0 , то значения весов уточняются по формуле $w_{ij}(t+1)=w_{ij}(t)-x_j$, где $(t+1)$– это номер текущего цикла, а t – номер предыдущего цикла;

Этот процесс повторяется для всех обучающих примеров.

Требования к функции активации

Входы – внешние данные, а веса, порог (смещение), функция активации – внутренние параметры нейрона. Каждый нейрон имеет свои собственные внутренние параметры. Изменяя параметры нейрона, изменяют его поведение, таким образом, подбирая параметры нейрона можно его обучать.

К функции активации предъявляют следующие требования (которые иногда нарушаются):

- непрерывная;
- монотонная;
- нелинейная;
- дифференцируемая.

Часто используются **сигмоидальные** функции активации. Такой нейрон называется сигмоидальным нейроном.

Достоинства гиперболического тангенса:

✓Современные компьютеры вычисляют функцию гиперболического тангенса быстрее, чем логистическую.

✓Изменяется в диапазоне от –1 до +1.

Часто бывает необходимо нормировать обучающий набор данных таким образом, чтобы среднее значение было равно 0 при единичном стандартном отклонении. Такая нормировка возможна только с функцией активации, которая способна принимать отрицательные значения. Нечетная функция, такая, как гиперболический тангенс, обеспечивает более быстрое обучение, чем несимметричная логистическая функция.

Сигмоидальные функции активации

В формулах сигмоидальных функций параметр k подбирается пользователем. Его значение влияет на форму функции активации. При малых значениях k график функции достаточно пологий, по мере роста значения k крутизна графика увеличивается.

При $k \to \infty$ сигмоидальная функция превращается в пороговую функцию, идентичную функции активации персептрона. На практике чаще всего для упрощения используется значение k =1.

Важным свойством сигмоидальной функции является ее дифференцируемость. Для униполярной функции имеем :

$$\frac{df(x)}{dx} = kf(x)(1 - f(x)) \tag{5}$$

тогда как для биполярной функции:

$$\frac{df(x)}{dx} = k(1 - f^2(x)) \tag{6}$$

И в первом, и во втором случае график изменения производной относительно переменной x имеет колоколообразную форму, а его максимум соответствует значению x=0.

Сигмоидальный нейрон, как правило, обучается с учителем.

Традиционные функции активации нейронов приведены в таблице 1.

Современные функции активации нейронов приведены в таблице 2.

Графики функций приведены на рисунке 3.

Функции активации нейронов (таблица 1)

Название	Формула	Область значений
Линейная	$f(x) = kx$	$(-\infty, \infty)$
Полулинейная	$f(x) = \begin{cases} kx, & x > 0 \\ 0, & x \le 0 \end{cases}$	$(0, \infty)$
Линейная с насыщением	$f(x) = \begin{cases} -1, x \le -1 \\ x, -1 < x < 1 \\ 1, x \ge 1 \end{cases}$	$(-1,1)$
Логистическая	$f(x) = \frac{1}{1 + e^{-kx}}$	$(0,1)$
Гиперболический тангенс	$f(x) = \frac{e^{kx} - e^{-kx}}{e^{kx} + e^{-kx}}$	$(-1,1)$
Рациональная	$f(x) = \frac{x}{k + \lvert x \rvert}$	$(-1,1)$
Синусоидальная	$f(x) = \sin(x)$	$(-1,1)$
Экспоненциальная	$f(x) = e^{-kx}$	$(0, \infty)$
Гаусса	$f(x) = \exp\left(-\frac{\lVert x - c_i \rVert^2}{2\sigma_i^2}\right)$	$(-\infty, \infty)$
Пороговая	$f(x) = \begin{cases} 0, x < 0 \\ 1, x \ge 0 \end{cases}$	$(0,1)$
Модульная	$f(x) = \lvert x \rvert$	$(0, \infty)$
Знаковая (сигнатурная)	$f(s) = \begin{cases} 1, x > 0 \\ -1, x \le 0 \end{cases}$	$(-1,1)$
Квадратичная	$f(x) = x^2$	$(0, \infty)$

Современные функции активации персептронов (таблица 2)

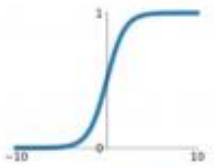
Название функции	Формула $f(x)$	Производная $f'(x)$
Логистическая сигмоида σ	$\frac{1}{1-e^{-x}}$	$f(x)$ $= f(x)(1-f(x))$
Гиперболический тангенс $tanh$	$\frac{e^x - e^{-x}}{e^x + e^{-x}}$	$1 - f^2(x)$
SoftSign	$\frac{x}{1-\lvert x\rvert}$	$\frac{1}{(1-\lvert x\rvert)^2}$
SoftPlus	$log(1+e^x)$	$\frac{1}{1-e^{-x}}$
ReLU	$\begin{cases} 0, & x < 0, \\ x, & x \geq 0. \end{cases}$	$\begin{cases} 0, & x < 0, \\ 1, & x \geq 0. \end{cases}$
LeakyReLU, Parameterized ReLU	$\begin{cases} \alpha x, & x < 0, \\ x, & x \geq 0. \end{cases}$	$\begin{cases} \alpha, & x < 0, \\ 1, & x \geq 0. \end{cases}$
ELU	$\begin{cases} \alpha(e^x - 1), & x < 0 \\ x, & x \geq 0. \end{cases}$	$\begin{cases} f(x) + \alpha, x < 0, \\ 1, & x \geq 0. \end{cases}$

Современные функции активации нейронов (рисунок 3)

Функция активации — определяет, какие нейроны будут активированы и какая информация будет передаваться последующим слоям. Нелинейность этих функций позволяет решать задачу нелинейной разделимости классов объектов.

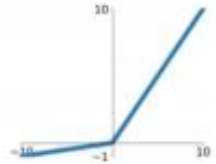
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



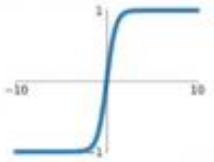
Leaky ReLU

$$\max(0.1x, x)$$



tanh

$$\tanh(x)$$

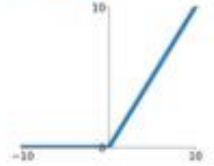


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

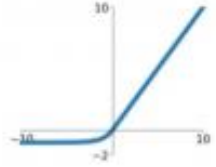
ReLU

$$\max(0, x)$$



ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Однослойный персептрон

В 1958 году **Фрэнк Розенблатт** предложил также первую модель нейронной сети, представляющей собой однослойную структуру с пороговой функцией активации и бинарными или многозначными входами.

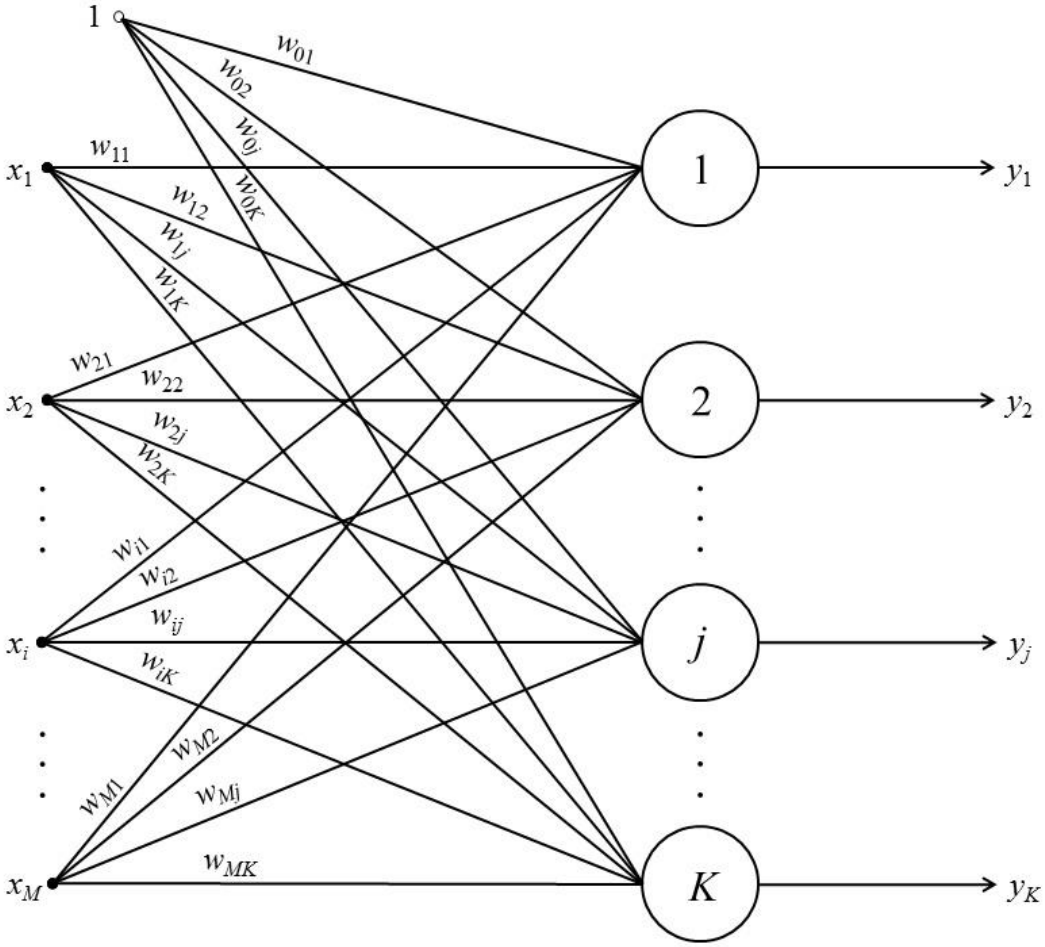
Однослойный персептрон образуют нейроны, расположенные в одной плоскости (рисунок 4). Каждый нейрон имеет поляризатор, то есть единичный сигнал, который с весом w_{i0} поступает на вход нейрона, а также множество связей с весами w_{ij} (первый индекс соответствует номеру нейрона, второй – номеру компонента входного вектора), по которым поступают входные сигналы x_j . В качестве функции активации чаще всего используется сигмоидальная функция.

Значения весов подбираются в процессе обучения сети путём приближения реальных выходных значений y_i к эталонным значениям d_i . Мерой близости является значение целевой функции. При использовании p обучающих векторов $\langle x, d \rangle$ для сети из K нейронов, целевую функцию можно определить следующим образом:

$$E = \frac{1}{2} \sum_{m=1}^{p} \sum_{i=1}^{K} \left(y_i^{(m)} - d_i^{(m)} \right)^2 \qquad (7)$$

Выходные сигналы y_i являются функциями весов сети и уточняются в процессе обучения в соответствии с критерием минимизации целевой функции (7).

Однослойный персептрон (рисунок 4)



Однослойный персептрон

Нейроны функционируют независимо друг от друга, поэтому возможности однослойного персептрона определяются возможностями отдельных нейронов. Каждый нейрон реализует функциональное отображение $y_i = f\left(\sum_{j=0}^{N} w_{ij} x_j\right)$, где f - сигмоидальная функция, поэтому значение выходного сигнала будет зависеть от знака выражения $\sum_{j=0}^{N} w_{ij} x_j$.

Выходной сигнал y_i при фиксированных значениях весов зависит от входного вектора x, который определяет гиперплоскость, разделяющую многомерное входное пространство на два подпространства. Поэтому, задача классификации (приписывание выходному сигналу значения 1 или 0), может быть решена одним нейроном, если это задача линейной разделимости классов. Добавление нейронов не улучшает её функциональные возможности, поэтому однослойная сеть не имеет практического значения. Добавление ещё одного слоя нейронов позволяет существенно расширить возможности сети.

Ф. Розенблаттом была доказана **теорема о сходимости персептрона**, а также были представлены доказательства ряда сопутствующих теорем, следствия из которых позволяли сделать вывод о том, каким условиям должны соответствовать архитектура искусственных нейронных сетей и методы их обучения.

Многослойный персептрон

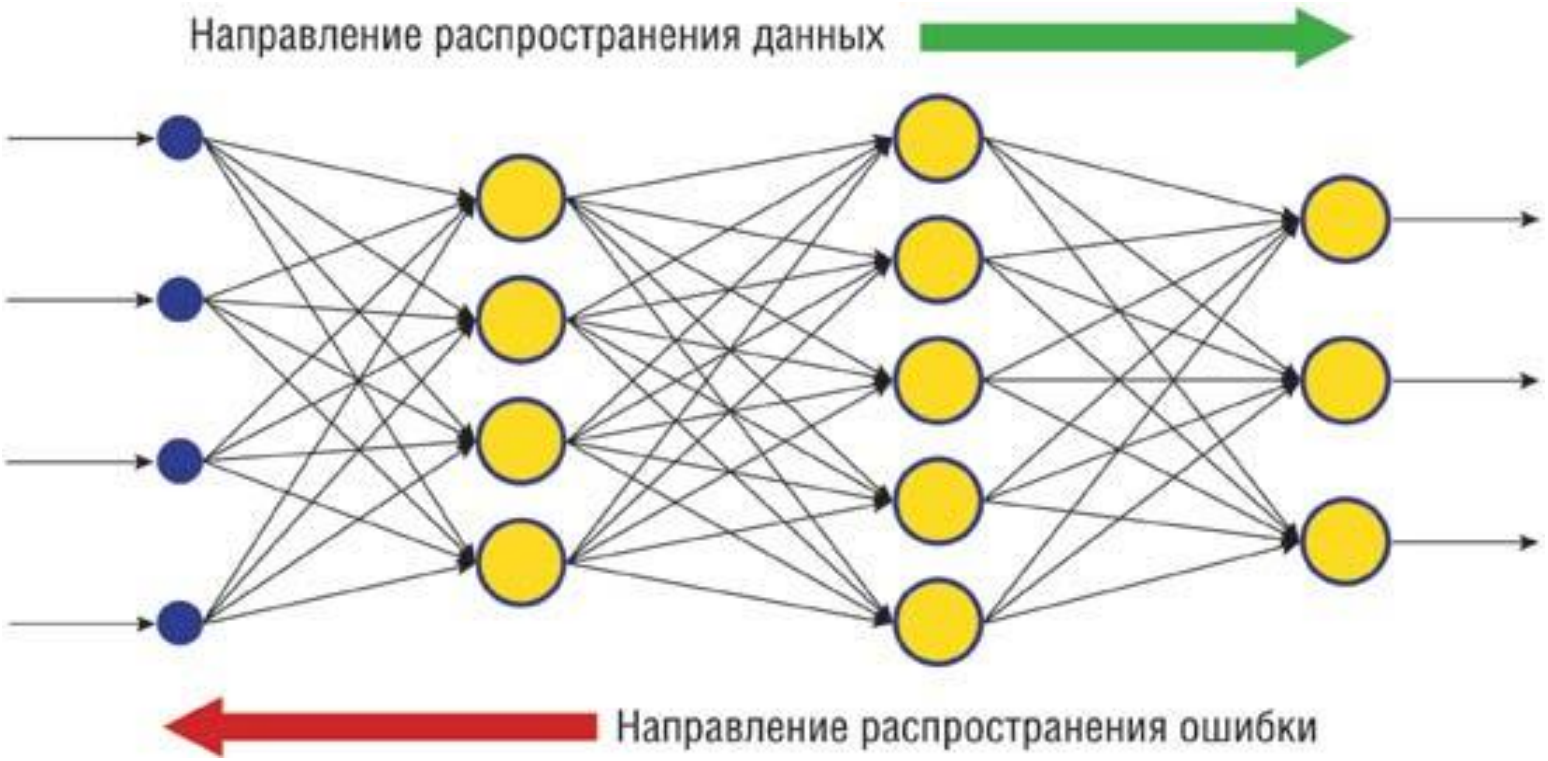
В настоящее время наиболее часто используемой архитектурой нейросети является **многослойный персептрон (MLP),** который представляет собой обобщение однослойного персептрона.

Обычно сеть состоит из множества входных узлов, которые образуют ***входной слой***; одного или нескольких ***скрытых слоев*** вычислительных нейронов и одного ***выходного слоя***. Входной сигнал распространяется по сети в прямом направлении от слоя к слою. Многослойные персептроны успешно применяются для решения разнообразных сложных задач. При этом обучение с учителем выполняется с помощью такого популярного алгоритма, как **алгоритм обратного распространения ошибки**.

Многослойный персептрон имеет три отличительных признака:

1. Каждый нейрон имеет **нелинейную функцию** активации. Данная функция должна быть гладкой (то есть всюду **дифференцируемой**). Самой популярной гладкой функцией активации является сигмоидальная функция.
2. Сеть содержит один или несколько слоев **скрытых нейронов**. Эти нейроны позволяют сети обучаться решению сложных задач, последовательно извлекая наиболее важные признаки из входного вектора.
3. Сеть обладает высокой степенью **связности**, реализуемой посредством синаптических соединений.

Структура многослойного персептрона с двумя скрытыми слоями (рисунок 5)



Многослойный персептрон

Структура многослойного персептрона с двумя скрытыми слоями изображена на рис. 5. Показанная на рисунке сеть является **полносвязной**, что характерно для многослойного персептрона. Это значит, что каждый нейрон любого слоя связан со всеми нейронами предыдущего слоя. **Функциональный сигнал** передается по сети в прямом направлении слева направо, **сигнал ошибки** справа налево.

Выходные нейроны составляют выходной слой сети. Остальные нейроны относятся к скрытым слоям. Первый скрытый слой получает данные из входного слоя. Результирующий сигнал первого скрытого слоя, в свою очередь, поступает на следующий скрытый слой и так далее, до самого конца сети.

В сетях подобного типа используются, в основном, сигмоидальные нейроны. Такую сеть легко можно интерпретировать как модель вход-выход, в которой веса и пороговые значения являются свободными параметрами модели. Такая сеть может моделировать функцию практически любой степени сложности, причем число слоев и число нейронов в каждом слое определяют сложность функции.

Проблемы обучения

В многослойных сетях эталонные значения выходных сигналов известны, как правило, только для нейронов выходного слоя, поэтому сеть невозможно обучить, руководствуясь только величинами ошибок на выходе нейросети. Один из вариантов решения этой проблемы – разработка учебных примеров для каждого слоя нейросети, что является очень трудоемкой операцией и не всегда осуществимо.

Второй вариант – динамическая подстройка весовых коэффициентов синапсов, в ходе которой выбираются, как правило, наиболее слабые связи и изменяются на малую величину в ту или иную сторону, а сохраняются только те изменения, которые повлекли уменьшение ошибки на выходе всей сети. Очевидно, что данный метод "тыка", несмотря на свою кажущуюся простоту, требует громоздких рутинных вычислений.

И, наконец, третий вариант – распространение сигналов ошибки от выходов нейросети к ее входам, в направлении, обратном прямому распространению сигналов в обычном режиме работы. Этот алгоритм обучения нейросети получил название процедуры обратного распространения. Разработка алгоритма обратного распространения для определения весов в многослойном персептроне сделала эти сети наиболее популярными у исследователей и пользователей нейронных сетей.

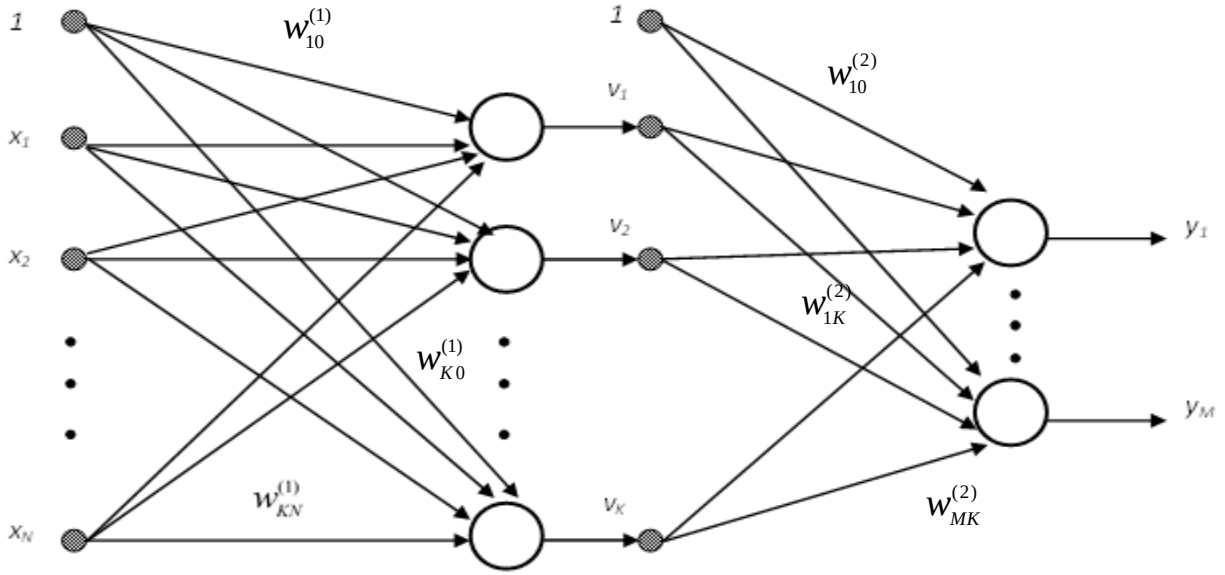
Достоинства и недостатки

Основными достоинствами многослойного персептрона являются простота в использовании, гарантированное получение ответа после прохода данных по слоям, хорошо апробированные и широко применяемые алгоритмы обучения, способность моделирования функции любой степени сложности.

С другой стороны, существует множество спорных вопросов при проектировании сетей прямого распространения. Например, сколько скрытых слоев необходимо для решения данной задачи, сколько следует выбрать нейронов в каждом слое, как сеть будет реагировать на данные, не включенные в обучающую выборку, и какой размер обучающей выборки необходим для достижения "хорошей" обобщающей способности сети.

Структура двухслойной сигмоидальной нейронной сети (рисунок 6)

На рис. 6 представлена сеть с одним скрытым слоем. Все последующие рассуждения относятся к сетям именно такого типа. Обозначения сигналов и весов также будут соответствовать этому рисунку. Веса нейронов скрытого слоя пометим верхним индексом (1), а выходного слоя – верхним индексом (2). Выходные сигналы нейронов скрытого слоя обозначим ($i=0, 1, 2,\ldots, K$), а выходного слоя ($s=1, 2, \ldots, M$).



Основные определения

С вектором *x* связаны два выходных вектора сети: вектор фактических выходных сигналов $y = \left[y_0, y_1, \ldots, y_M \right]^T$ и вектор ожидаемых выходных сигналов $d = \left[d_0, d_1, \ldots, d_M \right]^T$.

Цель обучения состоит в подборе таких значений весов $w_{ij}^{(1)}$ и $w_{si}^{(2)}$ для двух слоев сети, чтобы при заданном входном векторе *x* получить на выходе значения сигналов y_s которые с требуемой точностью будут совпадать с ожидаемыми значениями d_s для *s=1, 2, …, M.*

Если рассматривать единичный сигнал порогового элемента как один из компонентов входного вектора *x*, то веса пороговых элементов можно добавить в векторы весов соответствующих нейронов обоих слоев. При таком подходе выходной сигнал *i*-го нейрона скрытого слоя удается описать функцией

$$v_i = f\left(\sum_{j=0}^{N} w_{ij}^{(1)} x_j \right) \qquad (8)$$

в которой индекс 0 соответствует сигналу и весам пороговых элементов, причем $v_0 \equiv 1$, $x_0 \equiv 1$.

Теорема Цыбенко

В выходном слое s-ый нейрон вырабатывает выходной сигнал, определяемый как

$$y_s = f\left(\sum_{i=0}^{K} w_{si}^{(2)} v_i\right) = f\left(\sum_{i=0}^{K} w_{si}^{(2)} f\left(\sum_{j=0}^{N} w_{ij}^{(1)} x_j\right)\right) \quad (9)$$

Из формулы (9) следует, что на значение выходного сигнала влияют веса обоих слоев, тогда как сигналы, вырабатываемые в скрытом слое, не зависят от весов выходного слоя.

Универсальная теорема аппроксимации — теорема, доказанная Джорджем Цыбенко в 1989 году, которая утверждает, что искусственная нейронная сеть прямой связи (англ. feed-forward; в которых связи не образуют циклов) с одним скрытым слоем может аппроксимировать любую непрерывную функцию многих переменных с любой точностью.

Теорема была независимо доказана Джорджем Цибенко в 1989 году и Куртом Хорником в 1991 году. Доказательство Цибенко конкретно касалось сетей с сигмовидными функциями активации, а работа Хорника распространила результат на более широкий класс функций активации, включая популярный ReLU.

Формулировка теоремы

Пусть *N*- число входных узлов многослойного персептрона, а *M*- число выходных нейронов сети. Тогда теорема об универсальной аппроксимации формулируется следующим образом:

Пусть φ(.)– ограниченная, не постоянная монотонно возрастающая непрерывная функция. Пусть I_N - *N*- мерный единичный гиперкуб $[0,1]^N$.

Пусть пространство непрерывных на I_N функций обозначается символом $C(I_N)$.

Тогда для любой функции $f \in C(I_N)$ и ε>0 существует такое целое число *K* и множество действительных констант w_0 , w_i и w_{ij} , где $i=1, 2,..., K$, $j=1, 2,..., N$, что

$$F(x_1, x_2,..., x_N) = \sum_{i=1}^{K} w_i \varphi\left(\sum_{j=1}^{N} w_{ij} x_j + w_0 \right)$$

является реализацией аппроксимации функции f(.), то есть

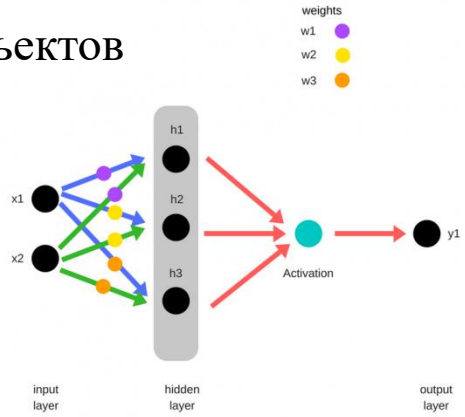
$$\left| F(x_1, x_2,..., x_N) - f(x_1, x_2,..., x_N) \right| < \varepsilon$$

для всех $x_1, x_2,..., x_N$ принадлежащих входному пространству.

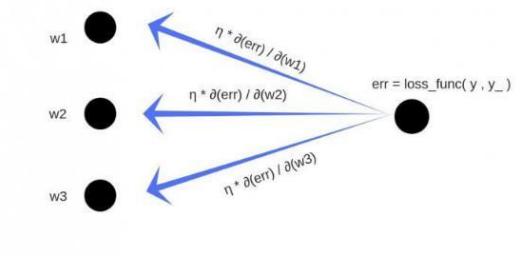
Теорема об универсальной аппроксимации непосредственно применима к многослойному персептрону.

Этапы построения модели многослойного персептрона

- Выбор архитектуры сети
  - Число входов – равно числу признаков объекта
  - Число выходов – равно числу классов объектов
  - **Функции активации нейронов**
  - **Функция ошибки**
- Подбор весов – **обучение сети**



Прямое распространение



Обратное распространение

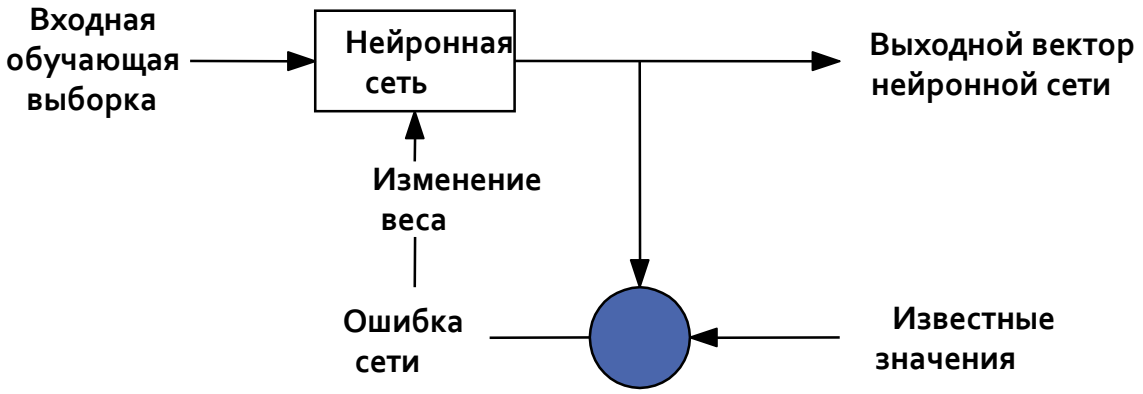
Обучение многослойного персептрона

Основная идея алгоритма обратного распространения ошибки

- Ошибку на последнем слое можно рассчитать явно
- Ошибку на промежуточном слое, распространим с последнего с текущими весами
  - Фактически, сеть запускается «наоборот» и вместо сигнала распространяется ошибка
  - Для её минимизации применяется градиентный спуск

Алгоритм обратного распространения ошибки

- это один из методов обучения многослойных нейронных сетей прямого распространения сигнала



Функция ошибки (loss function)

- Функция ошибки используется для расчёта ошибки между реальными и ожидаемыми ответами.
- Глобальная цель обучения сети — минимизировать эту ошибку. Она может зависеть от таких переменных, как веса и смещения нейронов.
- Функция ошибки одномерна и не является вектором, поскольку она оценивает, насколько хорошо нейронная сеть работает в целом.
- Функция ошибки в нейронной сети должна удовлетворять двум условиям:
  - Функция ошибки должна быть записана как среднее;
  - Функция ошибки не должна зависеть от каких-либо активационных значений нейронной сети, кроме значений, выдаваемых на выходе.

Некоторые известные функции ошибки :

- Среднеквадратичное отклонение (RMSE);
- Бинарная или категориальная кросс-энтропия (BinaryCrossentropy or CategoricalCrossentropy);
- Экспоненциальная (AdaBoost);
- Расстояние Кульбака — Лейблера или прирост информации.
- Квадратичная (MSE) – самая простая функция ошибки и наиболее часто используемая.

Градиентные алгоритмы обучения

Обучение сети состоит в подборе весов межнейронных связей, обеспечивающих наибольшую близость ответов сети к известным правильным ответам.

Задачу обучения нейронной сети будем рассматривать, как требование минимизировать априори определенную целевую функцию *E(w)*. При таком подходе можно применять для обучения алгоритмы, которые в теории оптимизации считаются наиболее эффективными. К ним относятся градиентные алгоритмы, чью основу составляет выявление градиента целевой функции. Они связаны с разложением целевой функции *E(w)* в ряд Тейлора в ближайшей окрестности точки имеющегося решения *w*.

В случае целевой функции от многих переменных $w = [w_1, w_2, ..., w_n]^T$ такое представление связывается с окрестностью ранее определенной точки (в частности, при старте алгоритма это исходная точка) в направлении *p*. Подобное разложение описывается универсальной формулой вида

$$E(w+p) = E(w) + [g(w)]^T p + \frac{1}{2} p^T H(w) p + ..., \qquad (10)$$

где $g(w) = \nabla E = \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, ..., \frac{\partial E}{\partial w_n} \right]^T$ - это вектор градиента,

Градиентные алгоритмы обучения

а симметричная квадратная матрица

$$H(w)=\begin{bmatrix} \frac{\partial^2 E}{\partial w_1 \partial w_1} & \cdots & \frac{\partial^2 E}{\partial w_1 \partial w_n} \\ \cdots & \cdots & \cdots \\ \frac{\partial^2 E}{\partial w_n \partial w_1} & \cdots & \frac{\partial^2 E}{\partial w_n \partial w_n} \end{bmatrix}$$

является матрицей производных второго порядка, называемой гессианом .

В выражении (10) p играет роль направляющего вектора, зависящего от фактических значений вектора w. На практике чаще всего рассчитываются три первых члена ряда (10), а последующие члены ряда просто игнорируются. При этом зависимость (10) может считаться квадратичным приближением целевой функции $E(w)$ в ближайшей окрестности найденной точки w с точностью, равной локальной погрешности отсеченной части $O(h^3)$, где $h = ||p||$. Для упрощения описания значения переменных, полученных в t-ом цикле, будем записывать с нижним индексом t. Точкой решения будем считать точку $w = w_t$, в которой достигается минимум целевой функции $E(w)$ и $g(w_t) = 0$, а гессиан $H(w_t)$ является положительно определенным.

В процессе поиска минимального значения целевой функции направление поиска p и шаг h подбираются таким образом, чтобы для каждой очередной точки выполнялось условие $E(w_{t+1}) < E(w_t)$. Поиск минимума продолжается, пока норма градиента не будет ниже априори заданного значения допустимой погрешности либо пока не будет превышено максимальное время вычислений (количество итераций).

Универсальный градиентный алгоритм обучения

Универсальный оптимизационный алгоритм обучения нейронной сети можно представить в следующем виде (будем считать, что начальное значение оптимизируемого вектора известно и составляет $w_t = w_0$):

w_t

1. Проверка сходимости и оптимальности текущего решения w_t . Если точка w_t отвечает градиентным условиям остановки процесса – завершение вычислений.
   В противном случае перейти к п.2.
2. Определение вектора направления оптимизации p_t для точки w_t .
3. Выбор величины шага η_t в направлении p_t , при котором выполняется условие $E(w_t + \eta_t p_t) < E(w_t)$ (11).
4. Определение нового решения $w_{t+1} = w_t + \eta_t p_t$, а также соответствующих ему значений $E(w_t)$ и $g(w_t)$, а если требуется – то и $H(w_t)$, и возврат к п.1.

Подбор коэффициента обучения

После правильно выбранного направления p_t , в градиентных алгоритмах обучения, следует определить новую точку решения w_{t+1} , в которой будет выполняться условие $E(w_{t+1}) < E(w_t)$. Необходимо подобрать такое значение η_t , чтобы новое решение $w_{t+1} = w_t + \eta_t p$ лежало как можно ближе к минимуму функции $E(w)$ в направлении p_t . Коэффициент обучения подбирается из диапазона [0,1]. Грамотный подбор коэффициента η_t обучения оказывает огромное влияние на сходимость алгоритма оптимизации к минимуму целевой функции. Слишком малое значение η_t не позволяет быстро минимизировать целевую функцию. Слишком большой шаг приводит к «перепрыгиванию» через минимум функции и фактически заставляет возвращаться к нему.

Существуют различные способы подбора значений η_t . Простейший из них основан на фиксации постоянного значения на весь период обучения. Этот способ имеет низкую эффективность, поскольку значение коэффициента обучения никак не зависит от вектора фактического градиента на данной итерации. Величина η подбирается, как правило, раздельно для каждого слоя сети с использованием различных эмпирических зависимостей.

Подбор коэффициента обучения

Один из походов состоит в определении минимального значения η для каждого слоя по формуле

$$\eta \le \min\left(\frac{1}{n_i}\right) \qquad (12),$$

где n_i обозначает количество входов i-го нейрона в слое.

Адаптивная формула для корректировки значений коэффициента обучения в зависимости от итерации обучения:

$$\eta(t) = \frac{\eta_0}{1+\left(\frac{t}{\tau}\right)} \qquad (13),$$

где η_0 - начальное значение коэффициента обучения, τ - константа времени поиска, задаваемые пользователем, а t - номер итерации обучения.

При достаточно больших значениях η_0 и при большом числе итераций, значительно превосходящих константу τ, алгоритм обучения будет вести себя как стохастический алгоритм аппроксимации, а веса будут сходиться к своим оптимальным значениям.

Алгоритм градиентного (наискорейшего) спуска

Если при разложении целевой функции $E(w)$ в ряд Тейлора ограничиться ее линейным приближением, то мы получим алгоритм градиентного спуска. Для выполнения соотношения $E(w_{t+1}) < E(w_t)$ достаточно подобрать $g(w_t)^T p < 0$. Условию уменьшения значения целевой функции отвечает выбор вектора направления

$$p_t = -g(w_t) \tag{14}$$

В этом случае коррекция весовых коэффициентов производится по формуле:

$$w_{ij}(t+1) = w_{ij}(t) + \eta p_t \tag{15}$$

В другом виде формулу коррекции весов по методу наискорейшего спуска можно представить следующим образом:

$$w_{ij}(t+1) = w_{ij}(t) - \eta \cdot \frac{\partial E(t)}{\partial w_{ij}(t)} \tag{16}$$

Ограничение слагаемым первого порядка при разложении функции в ряд Тейлора, не позволяет использовать информацию о ее кривизне. Это обуславливает линейную сходимость метода. Указанный недостаток, а также резкое замедление минимизации в ближайшей окрестности точки оптимального решения, когда градиент принимает очень малые значения, делают алгоритм градиентного спуска низкоэффективным. Тем не менее, простота, невысокие требования к объему памяти и относительно невысокая вычислительная сложность, обуславливают широкое использование алгоритма.

Алгоритм градиентного спуска с моментом

Повысить эффективность удается путем эвристической модификации выражения, определяющего направление градиента. Одна из модификаций получила название алгоритма обучения с моментом. При этом подходе уточнение весов сети производится по формуле:

$$w_{ij}(t+1) = w_{ij}(t) - \eta \cdot \frac{\partial E(t)}{\partial w_{ij}(t)} + \alpha(w_{ij}(t) - w_{ij}(t-1)) \quad (17)$$

где α - это коэффициент момента, принимающий значения в интервале [0, 1].

Первое слагаемое в формуле (17) соответствует алгоритму наискорейшего спуска, а второе слагаемое учитывает последнее изменение весов и не зависит от фактического значения градиента. Чем больше значение коэффициента α, тем большее значение оказывает показатель момента на подбор весов. При постоянном значении коэффициента обучения $\eta(t) = \eta$ приращение весов остается примерно одинаковым, то есть

$$\Delta w_{ij}(t) = \eta p(t) + \alpha \Delta w_{ij}(t)$$

.

При малых значениях градиента показатель момента начинает доминировать, что приводит к увеличению значения целевой функции, позволяющему выйти из зоны локального минимума. Однако показатель момента, не должен доминировать на протяжении всего процесса обучения, поскольку это приводит к расхождению алгоритма. На практике, увеличение целевой функции не допускается больше, чем на 4%. При этом показатель градиента начинает доминировать над показателем момента и процесс развивается в направлении минимизации, заданном вектором градиента.

Алгоритм обратного распространения ошибки

Алгоритм обратного распространения ошибки используется вместе с градиентными методами оптимизации. В настоящее время градиентные алгоритмы считаются наиболее эффективными алгоритмами обучения нейронных сетей. Целевая функция формируется чаще всего в виде квадратичной суммы разностей между фактическими и ожидаемыми значениями выходных сигналов, которая для *P* обучающих выборок определяется по формуле:

$$E(w) = \frac{1}{2}\sum_{t=1}^{P}\sum_{s=1}^{M}\left(y_s^{(t)} - d_s^{(t)}\right)^2 \qquad (18)$$

В случае единичной обучающей выборки *(x,d)* целевая функция имеет вид:

$$E(w) = \frac{1}{2}\sum_{s=1}^{M}\left(y_s - d_s\right)^2 \qquad (19)$$

Уточнение весов может проводиться после предъявления каждой обучающей выборки (так называемый режим «онлайн»), при этом используется целевая функция вида (19), либо однократно после предъявления всех обучающих выборок (режим «оффлайн»), при этом используется целевая функция вида (18). В последующем изложении используется целевая функция вида (19).

Алгоритм обратного распространения ошибки

1. Подать на вход сети вектор x и рассчитать значения выходных сигналов нейронов скрытых слоев и выходного слоя, а также соответствующие производные $\frac{df(u_i^{(1)})}{du_i^{(1)}}, \frac{df(u_i^{(2)})}{du_i^{(2)}}, \ldots, \frac{df(u_i^{(m)})}{du_i^{(m)}}$ функций активации каждого слоя (m – количество слоев).
2. Создать сеть обратного распространения ошибок путем изменения направления передачи сигналов, замены функций активации их производными и подачи на бывший выход сети в качестве входного сигнала разности между фактическими и ожидаемыми значениями.
3. Уточнить веса по формулам (24) и (25) на основе результатов, полученных в п.1 и п.2 для исходной сети и для сети обратного распространения ошибки.
4. Пункты 1, 2, 3 повторить для всех обучающих выборок, вплоть до выполнения условия остановки: норма градиента станет меньше заданного значения ε, характеризующего точность обучения.

Алгоритм обратного распространения ошибки

Рассмотрим основные расчетные формулы для сети с одним скрытым слоем, представленной на рисунке 6.

Используется сигмоидальная функция активации, при этом в случае гиперболического тангенса производная функции активации равна

$$\frac{df(u)}{du} = f(u) \cdot (1 - f(u)) \qquad (20)$$

В случае логистической функции производная равна

$$\frac{df(u)}{du} = 1 - f^2(u) \qquad (21)$$

В формулах (22) и (23) под переменной u будем понимать выходные сигналы сумматоров нейронов скрытого или выходного слоя, представленных формулами (24) и (25).

Алгоритм обратного распространения ошибки

$$u_i^{(1)} = \sum_{j=0}^{N} w_{ij}{}^{(1)} x_j \qquad (22)$$

$$u_s^{(2)} = \sum_{i=0}^{K} w_{si}{}^{(2)} v_i \qquad (23)$$

Уточнение весовых коэффициентов будем проводить после предъявления каждой обучающей выборки. Минимизация ведется методом градиентного спуска, что означает подстройку весовых коэффициентов следующим образом:

$$\Delta w_{ij}^{(m)} = -\eta \cdot \frac{\partial E}{\partial w_{ij}} \qquad (24)$$

$$w_{ij}^{(m)}(t+1) = w_{ij}^{(m)}(t) - \eta \cdot \frac{\partial E}{\partial w_{ij}^{(m)}} \qquad (25)$$

Здесь w_{ij} – весовой коэффициент синаптической связи, соединяющей i-ый нейрон слоя m с j-ым нейроном слоя m-1, η – коэффициент обучения, $0<\eta<1$.

Алгоритм обратного распространения ошибки

С учетом принятых на рисунке 6 обозначений целевая функция для выходного слоя нейронов определяется следующим образом:

$$\frac{\partial E}{\partial w_{si}^{(2)}} = (y_s - d_s) \cdot \frac{df(u_s^{(2)})}{du_s^{(2)}} \cdot v_i \qquad (26)$$

Здесь под y_s , как и раньше, подразумевается выход s –го нейрона.

Компоненты градиента относительно нейронов скрытого слоя описываются более сложной зависимостью:

$$\frac{\partial E}{\partial w_{ij}^{(1)}} = \sum_{s=1}^{M} (y_s - d_s) \cdot \frac{dy_s}{dv_i} \cdot \frac{dv_i}{dw_{ij}^{(1)}} \qquad (27)$$

В другом виде эта зависимость может быть выражена формулой:

$$\frac{\partial E}{\partial w_{ij}^{(1)}} = \sum_{s=1}^{M} (y_s - d_s) \cdot \frac{df(u_s^{(2)})}{du_s^{(2)}} \cdot w_{si}^{(2)} \frac{df(u_s^{(1)})}{du_i^{(1)}} x_j \qquad (28)$$

Эвристический алгоритм RPROP

Простой эвристический алгоритм, демонстрирующий высокую эффективность обучения, - это алгоритм М. Ридмиллера и Х. Брауна, называемый RPROP. В этом алгоритме при уточнении весов учитывается только знак градиентной составляющей, а ее значение игнорируется:

$$\Delta w_{ij}(t) = -\eta_{ij}(t)\,\mathrm{sgn}\left(\frac{\partial E(w(t))}{\partial w_{ij}}\right). \qquad (29)$$

Коэффициент обучения подбирается индивидуально для каждого веса w_{ij} с учетом изменения значения градиента:

$$\eta_{ij}(t) = \begin{cases} \min\left(a\,\eta_{ij}(t-1), \eta_{\max}\right) & для \quad S_{ij}(t)S_{ij}(t-1) > 0 \\ \max\left(b\,\eta_{ij}(t-1), \eta_{\min}\right) & для \quad S_{ij}(t)S_{ij}(t-1) < 0 \\ \eta_{ij}(t-1) & \text{в остальных случаях} \end{cases} \qquad (30)$$

где $S_{ij}(t) = \dfrac{\partial E(w(t))}{\partial w_{ij}}$, a и b – константы: $a=1.2$; $b=0.5$. Минимальное и максимальное значения коэффициента обучения составляют: $\eta_{\min} = 10^{-6}$ и $\eta_{\max} = 50$, а функция sgn( ) принимает значение, равное знаку градиента.

Эвристический алгоритм QuickProp

Алгоритм QuickProp содержит элементы, предотвращающие зацикливание в точке неглубокого локального минимума, возникающего в результате работы нейрона на фазе насыщения сигмоидальной кривой, где из-за близости к нулю производной функции активации процесс обучения практически прекращается.

Вес w_{ij} на k-ом шаге алгоритма изменяется согласно правилу:

$$\Delta w_{ij}(k) = -\eta_k \left[\frac{\partial E(w(k))}{\partial w_{ij}} + \gamma w_{ij}(k) \right] + \alpha_{ij}^k \Delta w_{it}(k-1) \qquad (31)$$

Первое слагаемое (31) соответствует оригинальному алгоритму наискорейшего спуска, последнее – фактору момента, а средний член предназначен для минимизации абсолютных значений весов. Коэффициент γ обычно имеющий малую величину (порядка 10^{-4}), приводит к уменьшению весов вплоть до разрыва соответствующих взвешенных связей. Константа η_k - это коэффициент обучения, который в данном алгоритме может иметь ненулевое значение (как правило, $0{,}01 < \eta_0 < 0{,}6$) на старте процесса обучения, когда $\Delta w_{it}(k-1) = 0$

или когда

$$\Delta w_{ij} \left[\frac{\partial E(w(k))}{\partial w_{ij}} + \gamma w_{ij}(k) \right] > 0$$

либо нулевое значение – в противном случае.

Эвристический алгоритм QuickProp

Важную роль в алгоритме QuickProp играет фактор момента, который приспосабливается к текущим результатам процесса обучения. Этот коэффициент подбирается индивидуально для каждого веса по правилу:

$$\alpha_{ij}^{k} = \begin{cases} \alpha_{\max}, \text{где } \beta_{ij}(k) > \alpha_{\max}, \text{где } S_{ij}(k)\Delta w_{ij}(k-1)\beta_{ij}(k) < 0; \\ \beta_{ij}(k) \end{cases} \quad (32)$$

причем:

$$S_{ij}(k) = \frac{\partial E(w(k))}{\partial w_{ij}} + \gamma w_{ij}(k) \quad (33)$$

$$\beta_{ij}(k) = \frac{S_{ij}(k)}{S_{ij}(k-1) - S_{ij}(k)} \quad (34)$$

Константа $\alpha_{\max}$- это максимальное значение коэффициента момента, которое принимается равным 1,75.

Эвристический алгоритм QuickProp

Также известна упрощенная версия алгоритма QuickProp, в которой значения весов изменяются в соответствии с правилом:

$$\Delta w_{ij}^{k} = \begin{cases} \alpha_{ij}(k)\Delta w_{ij}(k-1) \text{ для } \Delta w_{ij}(k-1) \neq 0 \\ \eta_0(k)\dfrac{\partial E}{\partial w_{ij}} \end{cases} \qquad (35)$$

$$\alpha_{ij}(k) = \min\left(\frac{S_{ij}(k)}{S_{ij}(k-1) - S_{ij}(k)}, a_{\max}\right), \qquad (36)$$

где $S_{ij}(k) = \dfrac{\partial E(w(k))}{\partial w_{ij}}$.

В нем уменьшено количество управляющих параметров и упрощена формула уточнения значений весов.

Стохастический алгоритм имитации отжига

Все представленные ранее алгоритмы обучения нейронных сетей являются локальными. Они ведут к одному из локальных минимумов целевой функции, лежащему в окрестности точки начала обучения. Только в ситуации, когда значение глобального минимума известно, удается оценить, находится ли найденный локальный минимум в достаточной близости от искомого решения. Если локальное решение признается неудовлетворительным, следует повторить процесс обучения при других начальных значениях весов и с другими управляющими параметрами.

При решении реальных задач в общем случае даже приблизительная оценка глобального минимума оказывается неизвестной. По этой причине возникает необходимость применения методов глобальной оптимизации. Рассмотрим метод имитации отжига.

Название данного алгоритма связано с методами имитационного моделирования в статистической физике, основанными на методе Монте-Карло. Исследование кристаллической решетки и поведения атомов при медленном остывании тела привело к появлению на свет стохастических алгоритмов, которые оказались чрезвычайно эффективными в комбинаторной оптимизации.

Стохастический алгоритм имитации отжига

Классический алгоритм имитации отжига:

1. Запустить процесс из начальной точки w при заданной начальной температуре $T = T_{max}$.
2. Пока T>0 повторить L раз следующие действия:

   2.1. Выбрать новое решение w' из окрестности w.

   2.2. Рассчитать значение целевой функции $\Delta = E(w') - E(w)$:

   2.3. Если $\Delta \le 0$ принять $w' = w$, в противном случае принять, что $w' = w$ с вероятностью $\exp(-\Delta / T)$ путем генерации случайного числа R из интервала (0,1) с последующим его сравнением со значением $\exp(-\Delta / T)$; если $\exp(-\Delta / T) > R$, принять новое решение $w' = w$, в противном случае проигнорировать новое решение.
3. Уменьшить температуру $(T \leftarrow rT)$ с использованием коэффициента уменьшения r, выбираемого из интервала (0,1), и вернуться к п.2.
4. После снижения температуры до нулевого значения провести обучение сети любым из представленных выше детерминированных методов, вплоть до достижения минимума целевой функции.

Стохастический алгоритм роя частиц

Метод роя частиц – стохастический алгоритм, хорошо зарекомендовавший себя при решении задач оптимизации большой размерности. Его смысл состоит в следующем. Пусть S – количество частиц в рое, каждая из которых имеет позици $x_i \in W^n$ и скорость $v_i \in W^n$, где W^n – пространство возможных параметров сети. Пусть $p_i \in W^n$ – лучшая позиция i-й частицы, а $g \in W^n$ – лучшая позиция внутри роя.

Тогда:

- для каждой частицы проинициализируем позицию случайным вектором, равномерно распределенным на интервале $(w_{\min}, w_{\max})$;
- для каждой частицы примем $p_i = x_i$; g для роя x_i также определим случайно;
- если $E(p_i) < E(g)$, принимаем $g = p_i$;
- проинициализировать скорость частиц $v_i \approx U(-|w_{\max} - w_{\min}|, |w_{\max} - w_{\min}|)$:
- пока не пройдено заданное число итераций N, выполнить следующее:
- для каждой частицы i и каждой размерности пространства W^n d выбрать случайные числа $r_p, r_g \approx U(0,1)$, пересчитать скорость частицы :

$$v_{id} = \omega v_{id} + \varphi_p r_p (p_{id} - x_{id}) + \varphi_g r_g (g_d - x_{id})$$

- для каждой частицы x_i пересчитать позицию:
- если $E(x_i) < E(p_i)$, принять $p_i = x_i$;
- если $E(p_i) < E(g)$, принять $x_i = x_i + v_i$;
- $g = p_i$ – лучшее решение.

Параметры $\omega, \varphi_p, \varphi$ и число итераций N выбираются экспериментально. В некоторых работах рекомендуется использовать следующие значения:

$$\omega = -0.1618 \qquad \varphi_p = 1{,}8903 \qquad \varphi_g = 2{,}1225,\ N = 500$$

Подбор архитектуры многослойного персептрона

На погрешность тестирования оказывает влияние отношение количества примеров в обучающей выборке к числу нейронов в сети. Небольшой объем обучающей выборки при фиксированном количестве нейронов вызывает хорошую адаптацию сети к элементам обучающей выборки, однако не улучшает способности к обобщению. Фактически задача аппроксимации подменяется в этом случае задачей приближенной интерполяции. В результате всякого рода нерегулярности обучающих данных и шумы могут восприниматься как существенные свойства процесса.

На основе опытных данных существуют следующие рекомендации по выбору числа скрытых нейронов:

1. Не следует выбирать число скрытых нейронов больше, чем удвоенное число входных элементов.
2. Число обучающих данных должно быть по крайней мере в $1/\varepsilon$ раз больше количества весов в сети, где ε - граница ошибки обучения.
3. Следует выявить особенности нейросети, так как в этом случае требуется меньшее количество скрытых нейронов, чем входов. Если есть сведения, что размерность данных может быть уменьшена, то следует использовать меньшее количество скрытых нейронов.

Подбор архитектуры многослойного персептрона

4. При обучении на бесструктурных входах необходимо, чтобы количество скрытых нейронов было больше, чем количество входов. Если набор данных не имеет общих свойств, необходимо использовать больше скрытых нейронов.
5. Имеет место взаимоисключающая связь между обобщающими свойствами (меньше нейронов) и точностью (больше нейронов), которая специфична для каждой задачи.
6. Большая сеть требует большего времени для обучения.

Существуют также практические рекомендации по модификации сети и параметров алгоритмов обучения:

1. Если ошибка обучения мала, а ошибка тестирования велика, следовательно, надо увеличить число скрытых нейронов или уменьшить число эпох обучения (возможно сеть переучена).
2. Если и ошибка обучения, и ошибка тестирования велики, следовательно, надо увеличить число эпох обучения (возможно сеть недоучена).
3. Если все весовые коэффициенты очень большие, следовательно, надо увеличить число скрытых нейронов.
4. Добавление нейронов не панацея; если известно, что нейронов достаточно, следует подумать о других причинах ошибок, например о недостаточном количестве обучающих данных.