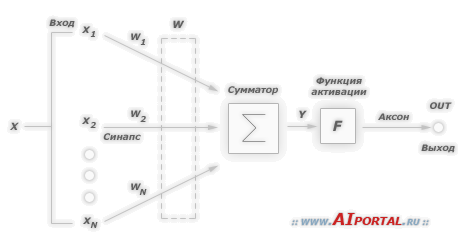
Вопросы к контрольной работе

По курсу «Нейросистемы и генетические алгоритмы»

1. **Искусственный нейрон. Виды функции активации.**

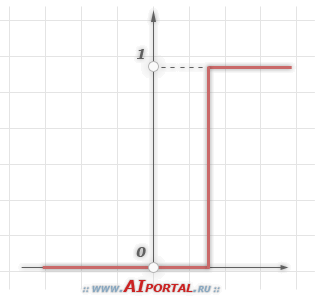
Для того, чтобы определиться с условными обозначениями, приведем ниже следующую модель нейрона:



Функция активации (активационная функция, функция возбуждения) – функция, вычисляющая выходной сигнал искусственного нейрона. В качестве аргумента принимает сигнал Y, получаемый на выходе входного сумматора Sigma. Наиболее часто используются следующие функции активации.

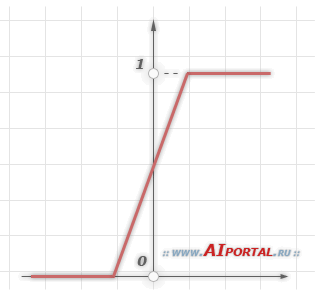
1. Единичный скачок или жесткая пороговая функция

Простая кусочно-линейная функция. Если входное значение меньше порогового, то значение функции активации равно минимальному допустимому, иначе – максимально допустимому.



2. Линейный порог или гистерезис

Несложная кусочно-линейная функция. Имеет два линейных участка, где функция активации тождественно равна минимально допустимому и максимально допустимому значению и есть участок, на котором функция строго монотонно возрастает.



3. Сигмоидальная функция или сигмоид

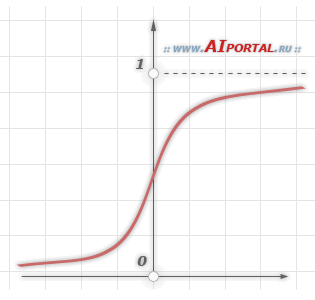
Монотонно возрастающая всюду дифференцируемая S-образная нелинейная функция с насыщением. Сигмоид позволяет усиливать слабые сигналы и не насыщаться от сильных сигналов. Гроссберг (1973 год) обнаружил, что подобная нелинейная функция активации решает поставленную им дилемму шумового насыщения.

Слабые сигналы нуждаются в большом сетевом усилении, чтобы дать пригодный к использованию выходной сигнал. Однако усилительные каскады с большими коэффициентами усиления могут привести к насыщению выхода шумами усилителей, которые присутствуют в любой физически реализованной сети. Сильные входные сигналы в свою очередь также будут приводить к насыщению усилительных каскадов, исключая возможность полезного использования выхода. Каким образом одна и та же сеть может обрабатывать как слабые, так и сильные сигналы?

Примером сигмоидальной функции активации может служить логистическая функция, задаваемая следующим выражением:

OUT~=~{1}/{1~+~exp(- alpha Y)}

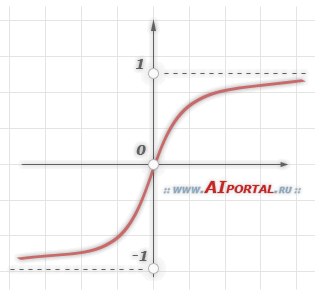
где alpha – параметр наклона сигмоидальной функции активации. Изменяя этот параметр, можно построить функции с различной крутизной.



Еще одним примером сигмоидальной функции активации является гиперболический тангенс, задаваемая следующим выражением:

OUT~=~th{(~~Y/alpha)}

где alpha – это также параметр, влияющий на наклон сигмоидальной функции.



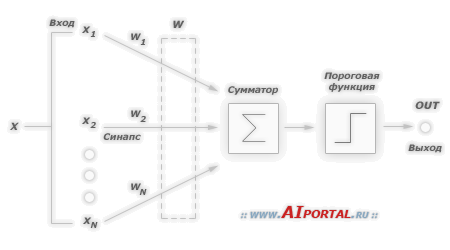
В заключение отметим, что функции активации типа единичного скачка и линейного порога встречаются очень редко и, как правило, используются на учебных примерах. В практических задач почти всегда применяется сигмоидальная функция активации.

1. **Персептрон. Обучение однослойного персептрона.**

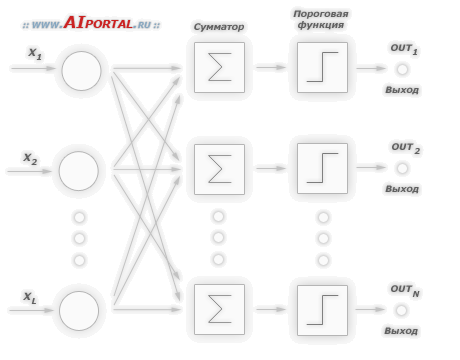
Персептрон

В качестве научного предмета искусственные нейронные сети впервые заявили о себе в 40-е годы. Стремясь воспроизвести функции человеческого мозга, исследователи создали простые аппаратные (а позже программные) модели биологического нейрона и системы его соединений, которые получили название персептроны. Когда нейрофизиологи достигли более глубокого понимания нервной системы человека, эти ранние попытки стали восприниматься как весьма грубые аппроксимации. Тем не менее, именно на персептронах были достигнуты первые впечатляющие результаты, стимулировавшие дальнейшие исследования, приведшие к созданию более изощренных сетей.

Первое систематическое изучение искусственных нейронных сетей было предпринято Маккалокком и Питтсом в 1943г. Простая нейронная модель, показанная на рисунке ниже, использовалась в большей части их работы. На вход поступает только двоичный сигнал, т.е. либо 0 либо 1. Элемент Sigma умножает каждый вход x_{N} на вес w_{N} и суммирует взвешенные входы. Если эта сумма больше заданного порогового значения, выход равен единице, в противном случае – нулю.



Именно такие системы и множество им подобных называются – персептронами. Персептроны состоят из одного слоя (т.е. количество слоев нейронов между входом X и выходом OUT равно одному) искусственных нейронов, соединенных с помощью весовых коэффициентов с множеством входов (см. рис. ниже).



Вершины-круги в левой части рисунка служат лишь для распределения входных сигналов. Они не выполняют каких- либо вычислений, и поэтому не считаются слоем. По этой причине они обозначены в виде круга, чтобы отличать их от вычисляющих нейронов (сумматоров), обозначенных квадратами.

В 60-е годы персептроны вызвали большой интерес и оптимизм. Розенблатт доказал замечательную теорему об обучении персептронов. Уидроу дал ряд убедительных демонстраций систем персептронного типа, и исследователи во всем мире стремились изучить возможности этих систем. Первоначальная эйфория сменилась разочарованием, когда оказалось, что персептроны не способны обучиться решению ряда простых задач. Минский строго проанализировал эту проблему и показал, что имеются жесткие ограничения на то, что могут выполнять однослойные персептроны, и, следовательно, на то, чему они могут обучаться. Так как в то время методы обучения многослойных сетей не были известны, исследователи перешли в более многообещающие области, и исследования однослойных персептронов пришли в упадок. Недавнее открытие методов обучения многослойных сетей в большей степени, чем какой-либо иной фактор, повлияло на возрождение интереса и исследовательских усилий.

Работа Минского, возможно, и охладила пыл энтузиастов персептрона, но обеспечила время для необходимой консолидации и развития лежащей в основе теории. Важно отметить, что анализ Минского не был опровергнут. Он остается важным исследованием и должен изучаться, чтобы ошибки 60-х годов не повторились.

Теория персептронов является основой для многих других типов искусственных нейронных сетей, а сами персептроны являются логической исходной точкой для изучения искусственных нейронных сетей.

1. **Линейная разделимость. Проблема функции "исключающее или" (XOR).**

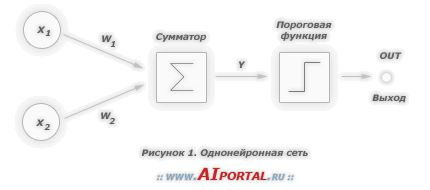
Представляемость персептрона. Проблема XOR - исключающего ИЛИ

Доказательство теоремы обучения персептрона показало, что персептрон способен научиться всему, что он способен представлять. Важно при этом уметь различать представляемость и обучаемость. Понятиепредставляемости относится к способности персептрона (или другой сети) моделировать определенную функцию.

Для иллюстрации проблемы представляемости допустим, что есть множество карт, помеченных цифрами от 0 до 9. Допустим также, что существует гипотетическая машина, способная отличать карты с нечетным номером от карт с четным номером и зажигающей индикатор на своей панели при предъявлении карты с нечетным номером. Представима ли такая машина персептроном? То есть, может ли быть сконструирован персептрон и настроены его веса (неважно каким образом) так, чтобы он обладал такой же разделяющей способностью? Если это так, то говорят, что персептрон способен представлять желаемую машину. Возможности представления однослойными персептронами весьма ограниченны. Имеется много простых машин, которые не могут быть представлены персептроном независимо от того, как настраиваются его веса. Для примера рассмотрим проблему XOR – исключающего ИЛИ.

Проблема функции XOR – исключающего ИЛИ

Один из самых пессимистических результатов Минского показывает, что однослойный персептрон не может воспроизвести такую простую функцию, как XOR. Это функция от двух аргументов, каждый из которых может быть нулем или единицей. Она принимает значение 1, когда один из аргументов равен единице, но не оба, иначе 0. Проблему можно проиллюстрировать с помощью однослойной однонейронной системы с двумя входами, показанной на рисунке ниже.



Обозначим один вход через x_{1}, а другой через x_{2}, тогда все их возможные комбинации будут состоять из четырех точек на плоскости, как показано на рисунке.

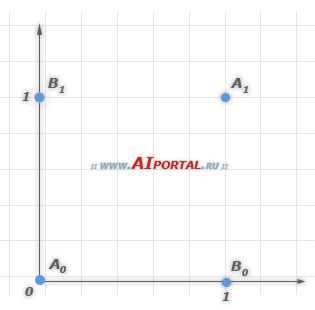


Таблица ниже показывает требуемую связь между входами и выходом, где входные комбинации, которые должны давать нулевой выход, помечены A_{0} и A_{1}, единичный выход – B_{0} и B_{1}.

|  |  |  |  |
| --- | --- | --- | --- |
| Точки | Значение x_{1} | Значение x_{2} | Требуемый выход |
| A_{0} | 0 | 0 | 0 |
| B_{0} | 1 | 0 | 1 |
| B_{1} | 0 | 1 | 1 |
| A_{1} | 1 | 1 | 0 |

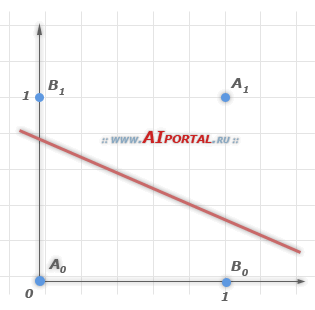
В модели сети, приведенной на рисунке выше, в качестве активационной функции применяется обычная пороговая функция: OUT принимает значение ноль, когда Y меньше 0.5, и единица в случае, когда Yбольше или равно 0,5. Нейрон выполняет следующее вычисление:

x_{1}w_{1}~+~x_{2}w_{2}~=~Y

Никакая комбинация значений двух весов не может дать соотношения между входом и выходом, заданного в таблице выше. Чтобы понять это ограничение, зафиксируем Y на величине порога 0.5. Сеть в этом случае описывается следующим уравнением:

x_{1}w_{1}~+~x_{2}w_{2}~=~0.5

Это уравнение линейно по x_{1} и x_{2}, т. е. все значения по x_{1} и x_{2}, удовлетворяющие этому уравнению, будут лежать на некоторой прямой в плоскости. Любые входные значения для x_{1} и x_{2} на этой линии будут давать пороговое значение 0.5. Прямая разделяет плоскость на две полуплоскости. Входные значения в одной полуплоскости обеспечат значения y больше порога, следовательно, OUT~=~1. Входные значения во второй полуплоскости обеспечат значения Y меньше порогового значения, делая OUT~=~0. Изменения значения w_{1}, w_{2} и порога будут менять наклон и положение прямой. Для того, чтобы сеть реализовала функцию XOR – исключающего ИЛИ, заданную таблицой выше, нужно расположить прямую так, чтобы точки A были с одной стороны прямой, а точки B – с другой. Попытавшись нарисовать такую прямую на рисунке ниже, убеждаемся, что это невозможно. Это означает, что какие бы значения ни приписывались весам и порогу, сеть неспособна воспроизвести соотношение между входом и выходом, требуемое для представления функции XOR – исключающего ИЛИ.



1. **Общий алгоритм решения задачи с помощью нейронной сети.**

1) определить какой смысл вкладывается в компоненты входного вектора (Х). Входной вектор должен содержать всю информацию, необходимую для получения ответа.

2) выбрать входной вектор У, т.о чтобы его компоненты содержали полный ответ поставленной задачи

3) выбрать вид нелинейности в нейронах

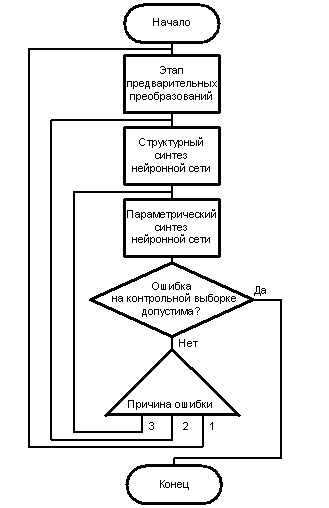
4) выбрать число слове и нейронов в каждом слое (задать структуру НС)

5) задать диапазон изменения входов /выходов весовых коэффициентов, учитывая множество значений выбранной функции активации)

6) присвоить начальные значения весовым коэффициентам

7) провести обучение (подобрать параметры сети так, чтобы задача решалась наилучшим образом). По окончанию обучения сеть готова решать задачи того типа, которым она обучена

8) подобрать на вход сети условия задачи в виде вектора Х, рассчитать выходной вектор У, который даст формализованное решение задачи.



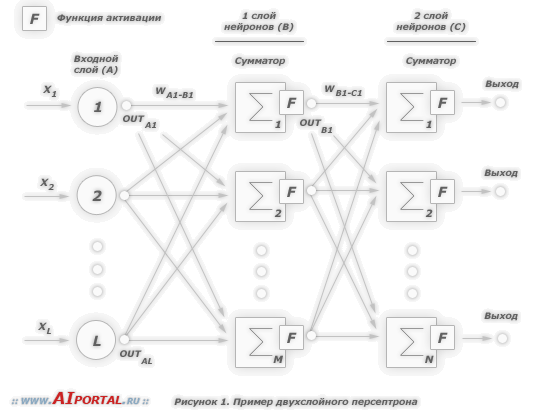
1. **Многослойный персептрон. Метод обратного распространения ошибки.**

Многослойный персептрон

Многослойными персептронами называют нейронные сети прямого распространения. Входной сигнал в таких сетях распространяется в прямом направлении, от слоя к слою. Многослойный персептрон в общем представлении состоит из следующих элементов:

* множества входных узлов, которые образуют входной слой;
* одного или нескольких скрытых слоев вычислительных нейронов;
* одного выходного слоя нейронов.

Многослойный персептрон представляет собой обобщение однослойного персептрона Розенблатта. Примером многослойного персептрона является следующая модель нейронной сети:



Количество входных и выходных элементов в многослойном персептроне определяется условиями задачи. Сомнения могут возникнуть в отношении того, какие входные значения использовать, а какие нет. Вопрос о том, сколько использовать промежуточных слоев и элементов в них, пока совершенно неясен. В качестве начального приближения можно взять один промежуточный слой, а число элементов в нем положить равным полусумме числа входных и выходных элементов.

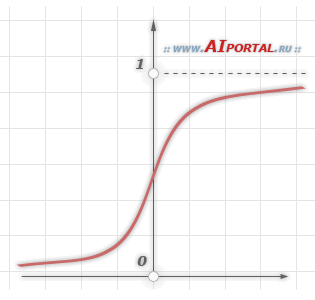
Многослойные персептроны успешно применяются для решения разнообразных сложных задач и имеют следующих три отличительных признака.

Свойство 1. Каждый нейрон сети имеет нелинейную функцию активации

Важно подчеркнуть, что такая нелинейная функция должна быть гладкой (т.е. всюду дифференцируемой), в отличие от жесткой пороговой функции, используемой в персептроне Розенблатта. Самой популярной формой функции, удовлетворяющей этому требованию, является сигмоидальная. Примером сигмоидальной функции может служить логистическая функция, задаваемая следующим выражением:

OUT~=~{1}/{1~+~exp(- alpha Y)}

где alpha – параметр наклона сигмоидальной функции. Изменяя этот параметр, можно построить функции с различной крутизной.



Наличие нелинейности играет очень важную роль, так как в противном случае отображение «вход-выход» сети можно свести к обычному однослойному персептрону.

Свойство 2. Несколько скрытых слоев

Многослойный персептрон содержит один или несколько слоев скрытых нейронов, не являющихся частью входа или выхода сети. Эти нейроны позволяют сети обучаться решению сложных задач, последовательно извлекая наиболее важные признаки из входного образа.

Свойство 3. Высокая связность

Многослойный персептрон обладает высокой степенью связности, реализуемой посредством синаптических соединений. Изменение уровня связности сети требует изменения множества синаптических соединений или их весовых коэффициентов.

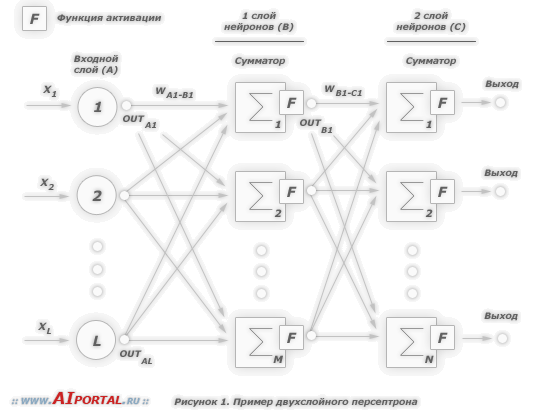
Комбинация всех этих свойств наряду со способностью к обучению на собственном опыте обеспечивает вычислительную мощность многослойного персептрона. Однако эти же качества являются причиной неполноты современных знаний о поведении такого рода сетей: распределенная форма нелинейности и высокая связность сети существенно усложняют теоретический анализ многослойного персептрона.

Алгоритм обратного распространения ошибки

Алгоритм обратного распространения ошибки является одним из методов обучения многослойных нейронных сетей прямого распространения, называемых также многослойными персептронами. Многослойные персептроны успешно применяются для решения многих сложных задач.

Обучение алгоритмом обратного распространения ошибки предполагает два прохода по всем слоям сети: прямого и обратного. При прямом проходе входной вектор подается на входной слой нейронной сети, после чего распространяется по сети от слоя к слою. В результате генерируется набор выходных сигналов, который и является фактической реакцией сети на данный входной образ. Во время прямого прохода все синаптические веса сети фиксированы. Во время обратного прохода все синаптические веса настраиваются в соответствии с правилом коррекции ошибок, а именно: фактический выход сети вычитается из желаемого, в результате чего формируется сигнал ошибки. Этот сигнал впоследствии распространяется по сети в направлении, обратном направлению синаптических связей. Отсюда и название – алгоритм обратного распространения ошибки. Синаптические веса настраиваются с целью максимального приближения выходного сигнала сети к желаемому.

Рассмотрим работу алгоритма подробней. Допустим необходимо обучить следующую нейронную сеть, применив алгоритм обратного распространения ошибки:



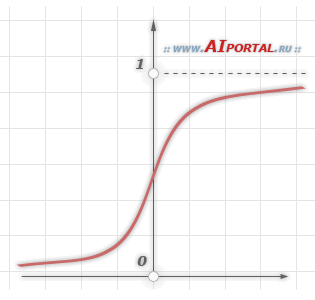
На приведенном рисунке использованы следующие условные обозначения:

* каждому слою нейронной сети соответствует своя буква, например: входному слою соответствует буква A, а выходному – C;
* все нейроны каждого слоя пронумерованы арабскими цифрами;
* w_{A1-B1} – синаптический вес между нейронами A1 и B1;
* OUT_{A1} – выход нейрона A1.

В качестве активационной функции в многослойных персептронах, как правило, используется сигмоидальная активационная функция, в частности логистическая:

OUT~=~1/{1~+~exp(- alpha Y)}

где alpha – параметр наклона сигмоидальной функции. Изменяя этот параметр, можно построить функции с различной крутизной. Оговоримся, что для всех последующих рассуждений будет использоваться именно логистическая функция активации, представленная только, что формулой выше.



Сигмоид сужает диапазон изменения так, что значение OUT лежит между нулем и единицей. Многослойные нейронные сети обладают большей представляющей мощностью, чем однослойные, только в случае присутствия нелинейности. Сжимающая функция обеспечивает требуемую нелинейность. В действительности имеется множество функций, которые могли бы быть использованы. Для алгоритма обратного распространения ошибки требуется лишь, чтобы функция была всюду дифференцируема. Сигмоид удовлетворяет этому требованию. Его дополнительное преимущество состоит в автоматическом контроле усиления. Для слабых сигналов (т.е. когда OUT близко к нулю) кривая вход-выход имеет сильный наклон, дающий большое усиление. Когда величина сигнала становится больше, усиление падает. Таким образом, большие сигналы воспринимаются сетью без насыщения, а слабые сигналы проходят по сети без чрезмерного ослабления.

Целью обучения сети алгоритмом обратного распространения ошибки является такая подстройка ее весов, чтобы приложение некоторого множества входов приводило к требуемому множеству выходов. Для краткости эти множества входов и выходов будут называться векторами. При обучении предполагается, что для каждого входного вектора существует парный ему целевой вектор, задающий требуемый выход. Вместе они называются обучающей парой. Сеть обучается на многих парах.

Алгоритм обратного распространения ошибки следующий:

1. Инициализировать синаптические веса маленькими случайными значениями.
2. Выбрать очередную обучающую пару из обучающего множества; подать входной вектор на вход сети.
3. Вычислить выход сети.
4. Вычислить разность между выходом сети и требуемым выходом (целевым вектором обучающей пары).
5. Подкорректировать веса сети для минимизации ошибки (как см. ниже).
6. Повторять шаги с 2 по 5 для каждого вектора обучающего множества до тех пор, пока ошибка на всем множестве не достигнет приемлемого уровня.

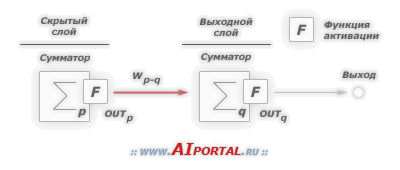
Операции, выполняемые шагами 2 и 3, сходны с теми, которые выполняются при функционировании уже обученной сети, т.е. подается входной вектор и вычисляется получающийся выход. Вычисления выполняются послойно. На рис. 1 сначала вычисляются выходы нейронов слоя B (слой A входной, а значит никаких вычислений в нем не происходит), затем они используются в качестве входов слоя C, вычисляются выходы OUT_{CN} нейронов слоя C, которые и образуют выходной вектор сети OUT. Шаги 2 и 3 образуют так называемый «проход вперед», так как сигнал распространяется по сети от входа к выходу.

Шаги 4 и 5 составляют «обратный проход», здесь вычисляемый сигнал ошибки распространяется обратно по сети и используется для подстройки весов.

Рассмотрим подробней 5 шаг – корректировка весов сети. Здесь следует выделить два нижеописанных случая.

Случай 1. Корректировка синаптических весов выходного слоя

Например, для модели нейронной сети на рис. 1, это будут веса имеющие следующие обозначения: w_{B1-C1} и w_{B2-C1}. Определимся, что индексом p будем обозначать нейрон, из которого выходит синаптический вес, а q – нейрон в который входит:



Введем величину delta, которая равна разности между требуемым T_{q} и реальным OUT_{q} выходами, умноженной на производную логистической функции активации (формулу логистической функции активации см. выше):

delta_{q}~=~OUT_{q}(1~-~OUT_{q})(T_{q}~-~OUT_{q})

Тогда, веса выходного слоя после коррекции будут равны:

w_{p-q}(i+1)~=~w_{p-q}(i)~+~eta delta_{q}OUT_{p}

где:

* i – номер текущей итерации обучения;
* w_{p-q} – величина синаптического веса, соединяющего нейрон p с нейроном q;
* eta (греческая буква «эта») – коэффициент «скорости обучения», позволяет управлять средней величиной изменения весов;
* OUT_{p} – выход нейрона p.

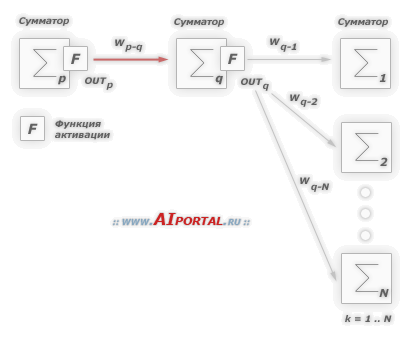
Приведем пример вычислений для синаптического веса w_{B1-C1} :

delta_{C1}~=~OUT_{C1}(1~-~OUT_{C1})(T_{1}~-~OUT_{C1})

w_{B1-C1}(i+1)~=~w_{B1-C1}(i)~+~eta delta_{C1}OUT_{B1}

Случай 2. Корректировка синаптических весов скрытого слоя

Для модели нейронной сети на рис. 1, это будут веса соответствующие слоям A и B. Определимся, что индексом p будем обозначать нейрон из которого выходит синаптический вес, а q – нейрон в который входит (обратите внимание на появление новой переменной k):



Введем величину delta, которая равна:

delta_{q}~=~OUT_{q}(1~-~OUT_{q})sum{k=1}{M}{delta_{k}w_{q-k}}

где:

* sum{k=1}{N}{} – сумма от 1 по N.

Тогда, веса скрытых слоев после коррекции будут равны:

w_{p-q}(i+1)~=~w_{p-q}(i)~+~eta delta_{q}OUT_{p}

Приведем пример вычислений для синаптического веса w_{A1-B1}:

delta_{B1}~=~OUT_{B1}(1~-~OUT_{B1})(delta_{C1}w_{B1-C1}~+~...~+~delta_{CN}w_{B1-CN}

w_{A1-B1}(i+1)~=~w_{A1-B1}(i)~+~eta delta_{B1}OUT_{A1}

Для каждого нейрона в скрытом слое должно быть вычислено delta и подстроены все веса, ассоциированные с этим слоем. Этот процесс повторяется слой за слоем по направлению к входу, пока все веса не будут подкорректированы.

Появление алгоритма обратного распространения ошибки стало знаковым событием в области развития нейронных сетей, так как он реализует вычислительно эффективный метод обучения многослойного персептрона. Будет неправильно утверждать, что алгоритм обратного распространения ошибки предлагает действительно оптимальное решение всех потенциально разрешимым проблем, однако он развеял пессимизм относительно обучения многослойных машин, воцарившийся в результате публикации Минского в 1969 году.

1. **Алгоритмы обучения однонаправленных многослойных нейронных сетей (детерменированные)**

Детерминистский метод обучения шаг за шагом осуществляет процедуру коррекции весов сети, основанную на использовании их текущих значений, а также величин входов, фактических выходов и желаемых выходов. Обучение персептрона является примером подобного детерминистского метода.

Представленные ранее выражения для целевой функции E(W) характеризуются, во-первых, аналитическим видом, во-вторых, дважды дифференцируемостью. Это дает возможность использовать для отыскания минимума E(W) практически весь арсенал универсальных методов поисковой оптимизации. Кроме того, специально для целей обучения ИНС разработаны модификации "канонических" методов.

Среди универсальных методов для обучения сетей наибольшее распространение получили следующие.

1. Метод градиента. Очередное приближение к оптимальному решению определяется по формуле

Wk+1=Wk-nu\*gradE(Wk),

где nu - коэффициент обучения, адаптивно подбираемый в ходе поиска так, чтобы Ek+1<Ek.

2. Метод наискорейшего спуска.

Wk+1=Wk-nu\*gradE(Wk),

где коэффициент обучения nu на каждом шаге поиска таков, что гарантированно обеспечивает достижение минимума Ek+1 в направлении -gradE(Wk). Расчет оптимального значения nu реализуется, как правило, методом полиномиальной аппроксимации (одномерная оптимизация в антиградиентом направлении).

3. Метод сопряженных градиентов.

Wk+1=Wk+nu\*pk,  
pk=-gradE(Wk)+bk-1\*pk-1.

В этом методе направление поиска на текущем шаге определяется как градиентом целевой функции в точке Wk, так и направлением поиска на предыдущем шаге pk-1, умноженном на коэффициент сопряжения bk-1. Существуют различные правила расчета bk-1, например, популярно такое:

bk-1=(gradE(Wk)T\*(gradE(Wk)-gradE(Wk-1)))/(gradE(Wk-1)T\*gradE(Wk-1)).

4. Ньютоноподобные методы. Напомним, формула метода Ньютона имеет следующий вид:

Wk+1=Wk-H-1(Wk)\*gradE(Wk),

где H-1(Wk) - матрица Гессе (матрица вторых производных целевой функции). В силу трудоемкости рассчета матрицы Гессе метод Ньютона как таковой на практике не используется, но служит прототипом для ряда весьма эффективных методов, предлагающих экономичные способы расчета приближенных эквивалентов матрицы Гессе и вектора градиента на каждом шаге поиска.

Специализированным методом обучения (минимизации целевой функции E(W)), широко используемым в практике обучения ИНС, следует считать метод обратного распространения ошибки, описываемый ниже в отдельном разделе.

Примером эвристического адгоритма, демонстрирующего неплохие результаты при подборе весовых коэффициентов ИНС, служит алгоритм RPROP (Resilent back PROPagation), формула которого имеет следующий вид:

wijk+1=wijk-nuijk\*sign(дE(Wk)/дwij)

В этом алгоритме используется только знак производной. Коэффициент обучения nuijk индивидуален для каждого веса wij и выбирается следующим образом:

nuijk=min(a\*nuijk-1, numax), если gijk\*gijk-1>0;

nuijk=max(b\*nuijk-1, numin),если gijk\*gijk-1<0;

nuijk=nuijk-1 в остальных случаях.

Здесь gijk=дE(Wk)/дwij, a=1,2, b=0,5 - эмпирически подобранные константы изменения коэффициента обучения; минимальное и максимальной допустимые значения коэффициента обучения выбраны так: numin=10-6 и numax=50. Явное достоинтство алгоритма - ускоренное обучение на пологих участках целевой функции (там, где величина градиента мала).

Существует много других эвристических алгоритмов, предложенных для разных типов сетей, классов решаемых задач, предметных областей.Однако, необходимо всегда учитывать, что эвристические методы строгого теоретического обоснования не имеют.

Для задач обучения ИНС, как и для задач параметрической оптимизации в целом, актуальной является проблема полимодальности целевых функций и, следовательно, отыскания глобальных экстремумов.

К сожалению, все упоминавшиеся ранее методы - это методы локального поиска, они способны отыскать только локальный экстремум, в область притяжения которого попадает начальный вектор весов. Среди методов глобального поиска перспективными с точки зрения обучения ИНС считаются метод имитации отжига и генетические алгоритмы.

Многие из упоминавшихся выше методов допускают функционирование в двух режимах: online и offline. В первом режиме ("онлайн") уточнение весов осуществляется после предъявления каждой обучающей выборки при использовании целевой функции вида:

E(W)=(1/2)\*sum[i=1:M]((yi-di)2).

Второй режим ("оффлайн") подразумевает корректировку весов только после обработки всех пар <Xk, Dk> в обучающей последовательности с использование целевой функции в виде:

E(W)=(1/2)\*sum[k=1:p](sum[i=1:M]((yi-di)2)).

Многие из методов обучения многослойного персептрона могут быть приспособлены для обучения сетей и других типов.

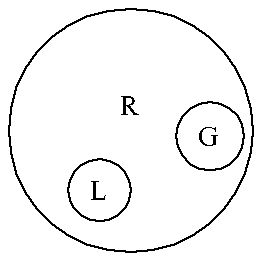
На эффективность процесса обучения ИНС оказывает влияние не только выбор метода подбора весовых коэффициентов, но и начальные значения этих коэффициентов. Понятно, что в лучшем случае начальные значения wij должны располагаться в области притяжения глобального минимума целефой функции E(W). К сожалению, в большинстве реальных задач такой выбор значений wij невозможен. Поэтому в практических реализациях ИНС чаще всего используется случайный выбор весов с равномерным распределением значений в заданном интервале.

Выбор этого диапазона слишком широким может привести к чрезмерно большим значениям ui=sum[j](wij\*xj), что, в свою очередь, вызовет глубокое насыщение сигмоидальной функции нейрона, что скажется либо на продолжительности обучения, либо "умертвит" нейрон.

Предлагаются различные оценки диапазона начальных значений, но все они лежат в пределах (по абсолютной величине) (0, 1). Одна из таких оценок такова: веса всех нейронов должны выбираться в диапазоне (-2/sqrt(Nin), 2/sqrt(Nin)), где Nin - количество входов нейрона; веса поляризации нейронов выходного слоя - нулевые; веса поляризации всех остальных слоев выбираются в диапазоне (-sqrt(Nin)/2, sqrt(Nin)/2).

До сих пор процесс обучения сети рассматривался как процесс минимизации погрешности обучения EL, определяемой на множестве L эталонных пар <Xk, Dk>, k=1, 2, ..., p. Однако, необходимо понимать, что истинная цель обучения - придание сети способности адекватно реагировать на все множество данных R, "типичным" подмножеством которого является множество L. Такая способность сети называется способностью к обощению.

Для оценки возможности к обобщению используется тестовое подмножество G (см. рисунок ниже) и определенная на нем погрешности обобщения EG.



Показано, что погрешность обощения зависит от мощности множества L (количества обучающих выборок p) и сложности ИНС. Однако, строго мера сложности сети не определена, и в качестве ее приближения предлагается использовать просто общее количество весовых коэффициентов сети. Разнообразные численные эксперименты показали, что для достижения сетью приемлемого уровня способности к обобщению необходимо, чтобы количество обучающих выборок в несколько раз превышало общее количество весов сети. Действительно, для обретения способности обобщать информацию сеть должна тренероваться на избыточном множестве данных, поскольку тогда веса будут адаптироваться не к уникальным выборкам, а к их усредненным совокупностям.

1. **Стохастические методы обучения нейронных сетей (обучение Больцмана, обучение Коши).**

Стохастические методы обучения выполняют псевдослучайные изменения величин весов, сохраняя те изменения, которые ведут к улучшениям. Чтобы показать это наглядно, рассмотрим [рис. 7.1](http://www.intuit.ru/studies/courses/88/88/lecture/20539?page=1#image.7.1), на котором изображена типичная сеть, где нейроны соединены с помощью весов. Выход нейрона является здесь взвешенной суммой его входов, которая преобразована с помощью нелинейной функции. Для обучения сети могут быть использованы следующие процедуры:

1. Выбрать вес случайным образом и подкорректировать его на небольшое случайное число. Предъявить множество входов и вычислить получающиеся выходы.
2. Сравнить эти выходы с желаемыми выходами и вычислить величину разности между ними. Общепринятый метод состоит в нахождении разности между фактическим и желаемым выходами для каждого элемента обучаемой пары, возведение разностей в квадрат и нахождение суммы этих квадратов. Целью обучения является минимизация этой разности, часто называемой целевой функцией.
3. Выбрать вес случайным образом и подкорректировать его на небольшое случайное значение. Если коррекция помогает (уменьшает целевую функцию), то сохранить ее, в противном случае вернуться к первоначальному значению веса.
4. Повторять шаги с 1 по 3 до тех пор, пока сеть не будет обучена в достаточной степени.

Больцмановское обучение

Этот стохастический метод непосредственно применим к обучению искусственных нейронных сетей:

1. Определить переменную T, представляющую искусственную температуру. Придать T большое начальное значение.
2. Предъявить сети множество входов и вычислить выходы и целевую функцию.
3. Дать случайное изменение весу и пересчитать выход сети и изменение целевой функции в соответствии со сделанным изменением веса.
4. Если целевая функция уменьшилась (улучшилась), то сохранить изменение веса.

Если изменение веса приводит к увеличению целевой функции, то вероятность сохранения этого изменения вычисляется с помощью распределения Больцмана:

P(c)=\exp(-c/kT),

где P(c) — вероятность изменения c в целевой функции; k — константа, аналогичная константе Больцмана, выбираемая в зависимости от задачи; T — искусственная температура.

Выбирается случайное число r из равномерного распределения от нуля до единицы. Если P(c) больше, чем r, то изменение сохраняется, в противном случае величина веса возвращается к предыдущему значению. Это позволяет системе делать случайный шаг в направлении, портящем целевую функцию, и дает ей тем самым возможность вырываться из локальных минимумов, где любой малый шаг увеличивает целевую функцию.

Для завершения больцмановского обучения повторяют шаги 3 и 4 для каждого из весов сети, постепенно уменьшая температуру T, пока не будет достигнуто допустимо низкое значение целевой функции. В этот момент предъявляется другой входной вектор, и процесс обучения повторяется. Сеть обучается на всех векторах обучающего множества, с возможным повторением, пока целевая функция не станет допустимой для всех них.

Величина случайного изменения веса на шаге 3 может определяться различными способами. Например, подобно тепловой системе, весовое изменение w может выбираться в соответствии с гауссовским распределением:

P(w)=\exp(-w^2/T^2),

где P(w) — вероятность изменения веса на величину w, Т — искусственная температура.

Так как требуется величина изменения веса \Delta w, а не вероятность изменения веса, имеющего величину w, то метод Монте-Карло может быть использован следующим образом:

1. Найти кумулятивную вероятность, соответствующую P(w). Это есть интеграл от P(w) в пределах от 0 до w.Поскольку в данном случае P(w) не может быть проинтегрирована аналитически, она должна интегрироваться численно, а результат необходимо затабулировать.
2. Выбрать случайное число из равномерного распределения на интервале (0,1). Используя эту величину в качестве значения P(w), найти в таблице соответствующее значение для величины изменения веса.

Свойства машины Больцмана широко изучены. Скорость уменьшения температуры должна быть обратно пропорциональна логарифму времени, чтобы была достигнута сходимость к глобальному минимуму. Скорость охлаждения в такой системе выражается следующим образом:

T(t)=\frac{T_0}{\log(1+t)},

где T(t) — искусственная температура как функция времени; T_0 — начальная искусственная температура; t — искусственное время.

Этот разочаровывающий результат предсказывает очень медленную скорость охлаждения (и вычислений). Вывод подтвержден и экспериментально. Машины Больцмана часто требуют для обучения очень большого ресурса времени.

Обучение Коши

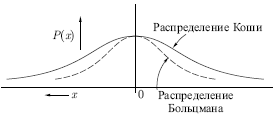


Рис. 7.3.

В этом методе при вычислении величины шага распределение Больцмана заменяется на распределение Коши. Распределение Кошиимеет, как показано на [рис. 7.3](http://www.intuit.ru/studies/courses/88/88/lecture/20539?page=4#image.7.3), более длинные "хвосты", увеличивая тем самым вероятность больших шагов. В действительности,распределение Коши имеет бесконечную (неопределенную) дисперсию. С помощью такого простого изменения максимальная скорость уменьшения температуры становится обратно пропорциональной линейной величине, а не логарифму, как для алгоритма обучения Больцмана. Это резко уменьшает время обучения. Зависимость может быть выражена следующим образом:

T(t)=\frac{T_0}{1+t}.

Распределение Коши имеет вид

P(x)=\frac{T(t)}{T(t)^2+x^2},

где P(x) есть вероятность шага величины x.

В данном уравнении P(x) может быть проинтегрирована стандартными методами. Решая относительно x, получаем

x_c=\rho T(t)\tg (P(x)),

где \rho — коэффициент скорости обучения; x_c — изменение веса.

Теперь применение метода Монте-Карло становится очень простым. Для нахождения x в этом случае выбирается случайное число из равномерного распределения на открытом интервале (-\pi/2, \pi/2) (необходимо ограничить функцию тангенса). Оно подставляется в формулу выше в качестве P(x), и с помощью текущей температуры вычисляется величина шага.

Трудности с алгоритмом обучения Коши

Несмотря на улучшение скорости обучения, даваемое машиной Коши по сравнению с машиной Больцмана, время сходимости все еще может в 100 раз превышать время для алгоритма обратного распространения. Отметим, что сетевой паралич особенно опасен для алгоритма обучения Коши, в особенности для сети с нелинейностью типа логистической функции. Бесконечная дисперсияраспределения Коши приводит к изменениям весов до неограниченных величин. Далее, большие изменения весов будут иногда приниматься даже в тех случаях, когда они неблагоприятны, часто приводя к сильному насыщению сетевых нейронов с вытекающим отсюда риском паралича.

Комбинирование обратного распространения с shape обучением Коши. Коррекция весов в комбинированном алгоритме, использующем обратное распространение и обучение Коши, состоит из двух компонент: (1) направленной компоненты, вычисляемой с использованием алгоритма обратного распространения, и (2) случайной компоненты, определяемой распределением Коши. Эти компоненты вычисляются для каждого веса, и их сумма является величиной, на которую изменяется вес. Как и валгоритме Коши, после вычисления изменения веса вычисляется целевая функция. Если происходит улучшение, изменение сохраняется безусловно. В противном случае, оно сохраняется с вероятностью, определяемой распределением Больцмана. Коррекция веса вычисляется с использованием представленных ранее уравнений для каждого из алгоритмов:

w_{mn,k}(n+1) =w_{mn,k}(n) + \eta [\alpha \Delta w_{mn,k}(n) + (1 - \alpha) \delta_{n,k} OUT_{m,j}] + (1 - \eta) x_c,

где \eta — коэффициент, управляющий относительными величинами Коши и обратного распространения в компонентах весового шага. Если \eta приравнивается нулю, система становится полностью машиной Коши. Если \eta приравнивается единице, система становится машиной обратного распространения. Изменение лишь одного весового коэффициента между вычислениями весовой функции неэффективно. Оказалось, что лучше сразу изменять все веса целого слоя, хотя для некоторых задач может стать выгоднее иная стратегия.

1. **Самоорганизующаяся карта признаков Кохонена (структура, обучение WTA, WTM).**

Самоорганизующиеся карты Кохонена используются для решения таких задач, как моделирование, прогнозирование, выявление наборов независимых признаков, сжатие информации, а также для поиска закономерностей в больших массивах данных. Наиболее часто описываемый алгоритм применяется для кластеризации данных.

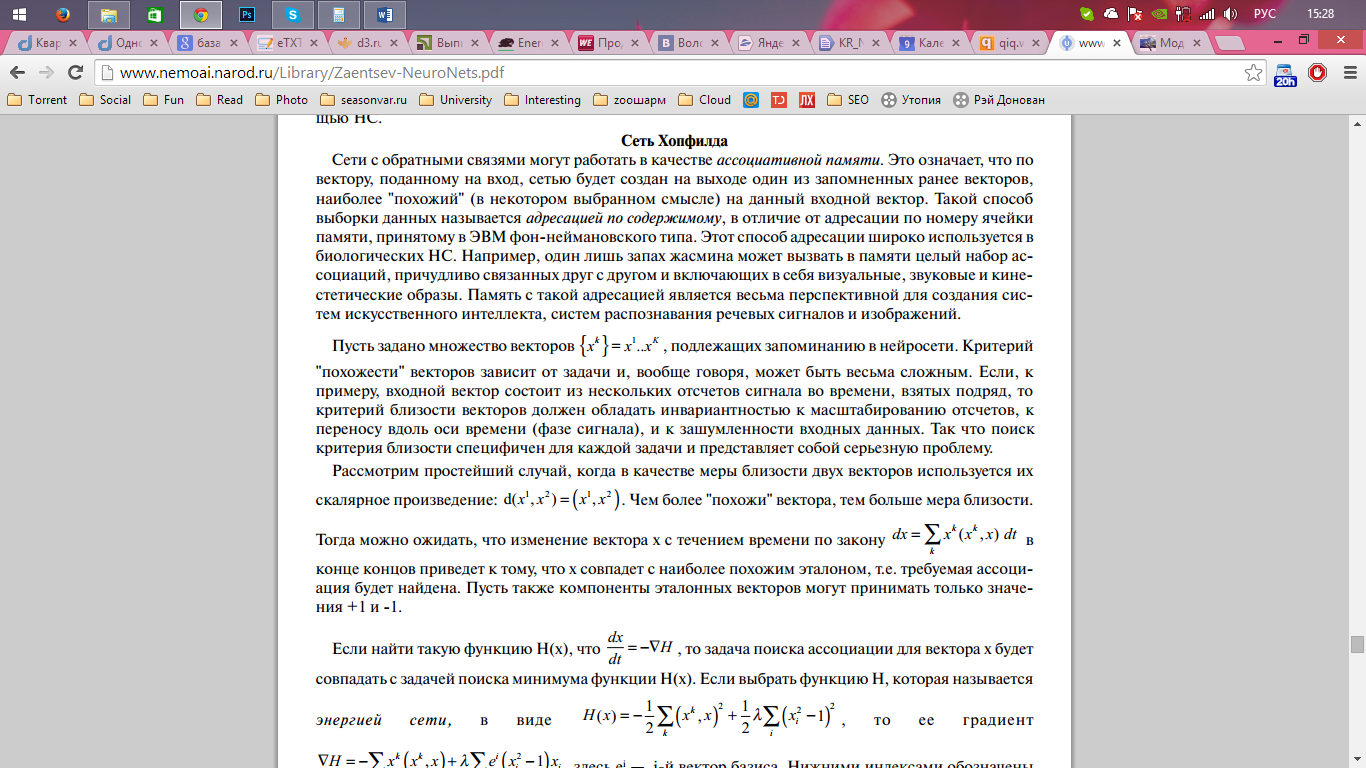
Самоорганизующаяся карта состоит из компонентов, называемых узлами или нейронами. Их количество задаётся [аналитиком](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B0%D0%BB%D0%B8%D1%82%D0%B8%D0%BA). Каждый из узлов описывается двумя векторами. Первый — т. н. вектор веса m, имеющий такую же размерность, что и входные данные. Второй — вектор r, представляющий собой координаты узла на карте. Карта Кохонена визуально отображается с помощью ячеек прямоугольной или шестиугольной формы; последняя применяется чаще, поскольку в этом случае расстояния между центрами смежных ячеек одинаковы, что повышает корректность визуализации карты.

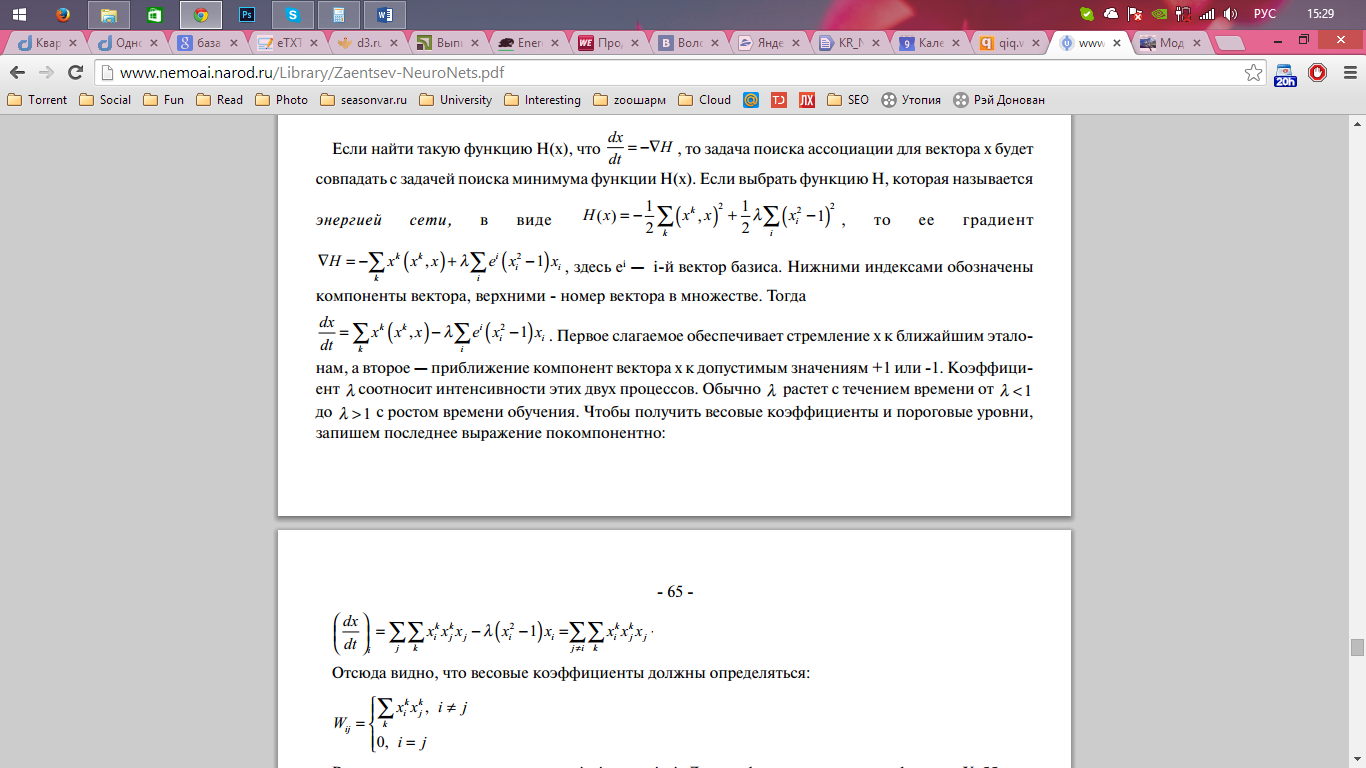
Изначально известна размерность входных данных, по ней некоторым образом строится первоначальный вариант карты. В процессе обучения векторы веса узлов приближаются к входным данным. Для каждого наблюдения (семпла) выбирается наиболее похожий по вектору веса узел, и значение его вектора веса приближается к наблюдению. Также к наблюдению приближаются векторы веса нескольких узлов, расположенных рядом, таким образом если в множестве входных данных два наблюдения были схожи, на карте им будут соответствовать близкие узлы. Циклический процесс обучения, перебирающий входные данные, заканчивается по достижении картой допустимой (заранее заданной аналитиком) погрешности, или по совершении заданного количества итераций. Таким образом, в результате обучения карта Кохонена классифицирует входные данные на кластеры и визуально отображает многомерные входные данные в двумерной плоскости, распределяя векторы близких признаков в соседние ячейки и раскрашивая их в зависимости от анализируемых параметров нейронов.

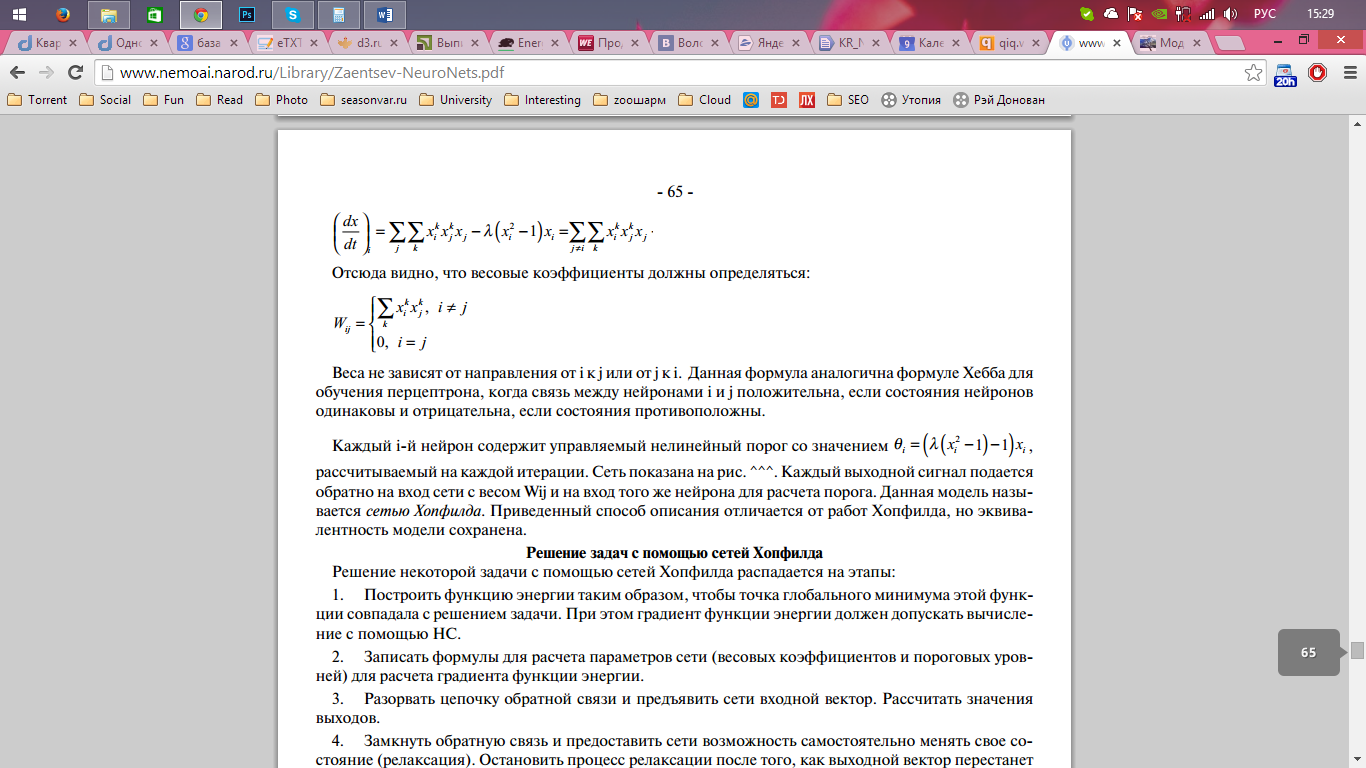
В результате работы алгоритма получаются следующие карты:  
карта входов нейронов — визуализирует внутреннюю структуру входных данных путем подстройки весов нейронов карты. Обычно используется несколько карт входов, каждая из которых отображает один из них и раскрашивается в зависимости от веса нейрона. На одной из карт определенным цветом обозначают область, в которую включаются приблизительно одинаковые входы для анализируемых примеров.  
карта выходов нейронов — визуализирует модель взаимного расположения входных примеров. Очерченные области на карте представляют собой кластеры, состоящие из нейронов со схожими значениями выходов.  
специальные карты — это карта кластеров, полученных в результате применения алгоритма самоорганизующейся карты Кохонена, а также другие карты, которые их характеризуют.

* Инициализация карты, то есть первоначальное задание векторов веса для узлов.
* Цикл:
  + Выбор следующего наблюдения (вектора из множества входных данных).
  + Нахождение для него лучшей единицы соответствия (best matching unit, BMU, или Winner) — узла на карте, вектор веса которого меньше всего отличается от наблюдения (в метрике, задаваемой аналитиком, чаще всего, евклидовой).
  + Определение количества соседей BMU и обучение — изменение векторов веса BMU и его соседей с целью их приближения к наблюдению.
  + Определение ошибки карты.

1. **Нейронная сеть Хопфилда (структура, функционирование, обучение).**

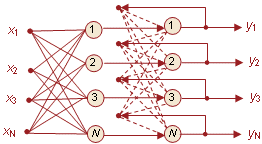






1. **Нейронная сеть Хемминга (структура, функционирование, обучение).**

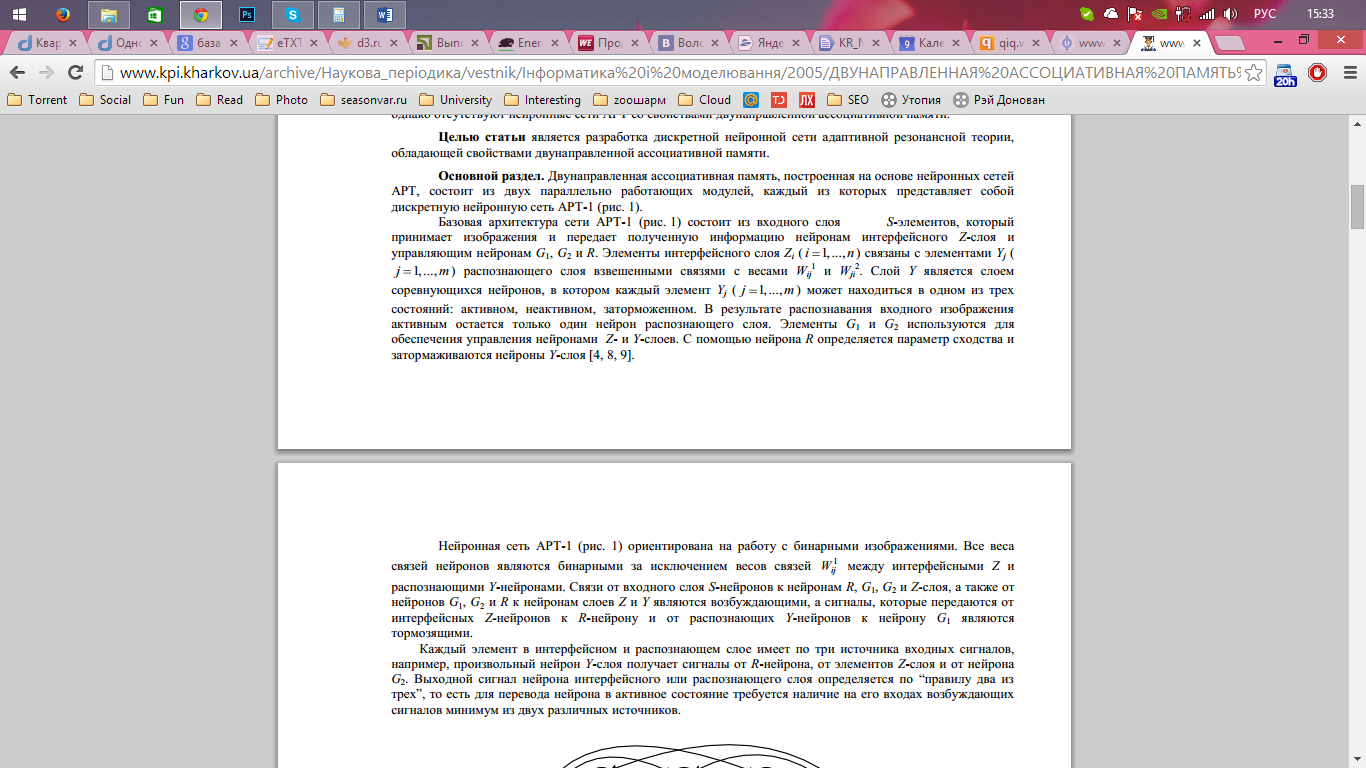
Нейронная сеть, состоящая из двух слоев, каждый из которых содержит число нейронов M, равное числу хранящихся образов. Нейроны первого слоя имеют N связей, соединенными со входами сети (образующими фиктивный нулевой слой). Нейроны второго слоя связаны между собой отрицательными обратными связями. Единственную положительную [обратную связь](http://www.basegroup.ru/glossary/definitions/feedback/) каждый нейрон имеет с собственным выходом.



Нейронные сети Хемминга можно использовать для реализации [ассоциативной памяти](http://www.basegroup.ru/glossary/definitions/associative_memory/) в тех случаях, когда нет необходимости, чтобы сеть выдавала на выходе образ в явном виде, а достаточно только его номер (или код). По сравнению с сетью Хопфилда сеть Хемминга имеет меньшие затраты на память и объем требуемых вычислений.

Принцип работы сети Хемминга основан на вычислении расстояния Хемминга от входного образа до всех образов, хранимых сетью. Под расстоянием Хемминга понимается число различающихся бит в двух бинарных векторах. Сеть должна выбрать образ с минимальным хемминговым расстоянием до входного. При этом на выходе сети формируется не сам этот образ, а выполняется активизация выхода, ассоциированного с ним.

1. **Двунаправленная ассоциативная память.**



**12 ГА, основные понятия ГА (Хромосома, популяция и т.д.)**

Генети́чний алгори́тм ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) genetic algorithm) — це [еволюційний алгоритм](https://uk.wikipedia.org/wiki/%D0%95%D0%B2%D0%BE%D0%BB%D1%8E%D1%86%D1%96%D0%B9%D0%BD%D0%B8%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) пошуку, що використовується для вирішення задач оптимізації і моделювання шляхом послідовного підбору, комбінування і варіації шуканих параметрів з використанням механізмів, що нагадують [біологічну](https://uk.wikipedia.org/wiki/%D0%91%D1%96%D0%BE%D0%BB%D0%BE%D0%B3%D1%96%D1%8F) [еволюцію](https://uk.wikipedia.org/wiki/%D0%95%D0%B2%D0%BE%D0%BB%D1%8E%D1%86%D1%96%D1%8F).

Особливістю генетичного алгоритму є акцент на використання оператора «схрещення», який виконує операцію рекомбінацію рішень-кандидатів, роль якої аналогічна ролі схрещення в живій природі.

Популяція - набір особей

Хромосома – упорядковані гени. Бітова строка

Ген - отамарний елемент хромосоми

Генотип - набір хомосом даної особи

Особи популяці - генотип

Фенотип - набір значення віівідаючим даному генотипу

Алель - значення гена це нулі або 1

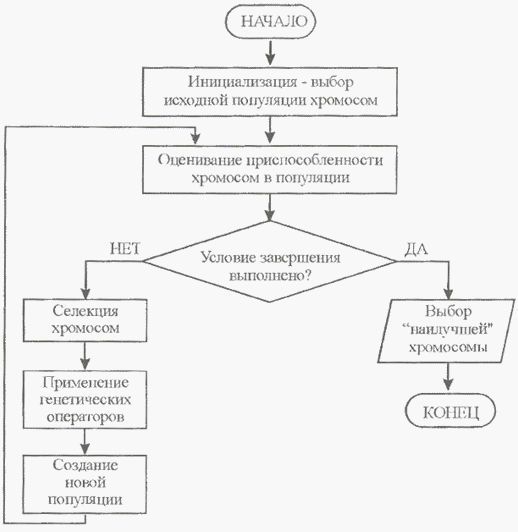
Локус - позиція генів в хромосомі

Функція пристосування - міра якості триманих рішень

**13 Классический ГА (Основные шаги, блок-схема)**

Классический генетический алгоритм (также называемый элементарным или простым генетическим алгоритмом) состоит из следующих шагов:

* инициализация, или выбор исходной популяции хромосом;
* оценка приспособленности хромосом в популяции – расчет функции приспособленности для каждой хромосомы;
* проверка условия остановки алгоритма;
* селекция хромосом – выбор тех хромосом, которые будут участвовать в создании потомков для следующей популяции;
* применение генетических операторов – мутации и скрещивания;
* формирование новой популяции;
* выбор «наилучшей» хромосомы.



**14. Кодирование параметров в задачи ГА**

Выбор исходной популяции связан с представлением параметров задачи в форме хромосом, т.е. с так называемым хромосомным представлением. Это представление определяется способом кодирования. В классическом генетическом алгоритме применяется двоичное кодирование, т.е. аллели всех генов в хромосоме равны 0 или 1. Длина хромосом зависит от условий задачи, точнее говоря - от количества точек в пространстве поиска.

**15. Операторы ГА, стратегии отбора родительских пар для скрещивания**

Генетический оператор по аналогии с оператором алгоритма – средство отображения одного множества на другое. Другими словами, это конструкция, представляющая один шаг из последовательности действий генетического алгоритма.

Оператор репродукции (селекция) – это процесс, посредством которого хромосомы (альтернативные решения), имеющие более высокое значение целевой функции (с «лучшими» признаками), получает большую возможность для воспроизводства (репродукции) потомков, чем «худшие» хромосомы. Элементы, выбранные для репродукции, обмениваются генетическим материалом, создавая аналогичных или различных потомков.

Существует большое число видов операторов репродукции. К ним относятся следующие:

Селекция на основе рулетки – это простой и широко используемый в простом генетическом алгоритме метод. При его реализации каждому элементу в популяции соответствует зона на колесе рулетки, пропорционально соразмерная с величиной целевой функции. Тогда при повороте колеса рулетки каждый элемент имеет некоторую вероятность выбора для селекции. Причем элемент с большим значением целевой функции имеет большую вероятность для выбора.

Селекция на основе заданной шкалы. Здесь популяция предварительно сортируется от «лучшей» к «худшей» на основе заданного критерия. Каждому элементу назначается определенное число и тогда селекция выполняется согласно этому числу.

Элитная селекция. В этом случае выбираются лучшие (элитные) элементы на основе сравнения значений целевой функции. Далее они вступают в различные преобразования, после которых снова выбираются элитные элементы. Процесс продолжается аналогично до тех пор, пока продолжают появляться элитные элементы.

Турнирная селекция. При этом некоторое число элементов (согласно размеру «турнира») выбираются случайно или направленно из популяции, и лучшие элементы в этой группе на основе заданного турнира определяются для дальнейшего эволюционного поиска.

Оператор репродукции считается эффективным, если он создает возможность перехода из одной подобласти альтернативных решений области поиска в другую. Это повышает вероятность нахождения глобального оптимума целевой функции. Выделяют два основных типа реализации оператора репродукции: случайный выбор хромосом; выбор хромосом на основе значений целевой функции.

Выбор родителей:

Размножение в генетических алгоритмах обычно половое — чтобы произвести потомка, нужны несколько родителей, обычно два.

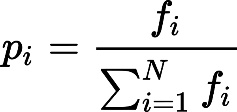
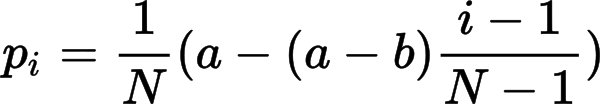
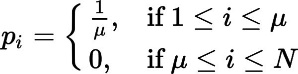
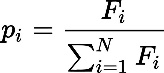
Можно выделить несколько операторов выбора родителей:

* Панмиксия - оба родителя выбираются случайно, каждая особь популяции имеет равные шансы быть выбранной
* Инбридинг - первый родитель выбирается случайно, а вторым выбирается такой, который наиболее похож на первого родителя
* Аутбридинг - первый родитель выбирается случайно, а вторым выбирается такой, который наиболее не похож на первого родителя

Инбридинг и аутбридинг бывают в двух формах: фенотипной и генотипной. В случае фенотипной формы похожесть измеряется в зависимости от значения функции приспособленности (чем ближе значения целевой функции, тем особи похожее), а в случае генотипной формы похожесть измеряется в зависимости от представления генотипа (чем меньше отличий между генотипами особей, тем особи похожее).

**16. Парасетры ГА. Способы отбора особей в новую популяцию.**

ГА представляет из себя итерационный процесс, в котором особи сначала отбираются для скрещивания, потом скрещиваются, затем из их потомков формируется новое поколение и все начинается сначала. Стратегии отбора являются составной частью ГА и определяют "достойных" для скрещивания особей.

* Турнирная селекция - сначала случайно выбирается установленное количество особей (обычно две), а затем из них выбирается особь с лучшим значением функции приспособленности
* Метод рулетки - вероятность выбора особи тем вероятнее, чем лучше её значение функции приспособленности , где Pi - вероятность выбора i особи, fi- значение функции приспособленности для i особи, N - количество особей в популяции
* Метод ранжирования - вероятность выбора зависит от места в списке особей отсортированном по значению функции приспособленности , где a є [1,2], b =2-a, I – порядковый номер особи в списке особей отсортированном по значению функции приспособленности ( т.е. C:\Users\zvegi\Downloads\b68fe0560a6ca0ba307557bbdc2277475d78d2d3.jpg – если мы минимизируем значение функции приспособленности)
* Равномерное ранжирование - вероятность выбора особи определяется выражением:  , где C:\Users\zvegi\Downloads\ebbe149aa9e438d5aa5859eacc4a87df493eea40.jpg параметр метода
* Сигма отсечение - для предотвращения преждевременной сходимости генетического алгоритма используются методы, масштабирующие значение целевой функции. Вероятность выбора особи тем больше, чем оптимальнее значение масштабируемой целевой функции , где C:\Users\zvegi\Downloads\93f77ad17707dba52ff39ddb15f6fa7defd2d8c4.jpg – среднее значение целевой функции для всей популяции, σ - среднеквадратичное отклонение значения целевой функции

**17. Модели ГА**

1. Canonical GA (J. Holland)

Данная модель алгоритма является классической. Она была предложена Джоном Холландом в его знаменитой работе "Адаптация в природных и исусственных средах" (1975). Часто можно встретить описание простого ГА (Simple GA, D. Goldberg), он отличается от канонического тем, что использует либо рулеточный, либо турнирный отбор. Модель канонического ГА имеет следующие характеристики:

● Фиксированный размер популяции.

● Фиксированная разрядность генов.

● Пропорциональный отбор.

● Особи для скрещивания выбираются случайным образом.

● Одноточечный кроссовер и одноточечная мутация.

● Следующее поколение формируется из потомков текущего поколения без "элитизма". Потомки занимают места своих родителей.

2. Genitor (D. Whitley)

В данной модели используется специфичная стратегия отбора. Вначале, как и полагается, популяция инициализируется и её особи оцениваются. Затем выбираются случайным образом две особи, скрещиваются, причем получается только один потомок, который оценивается и занимает место наименее приспособленной особи. После этого снова случайным образом выбираются 2 особи, и их потомок занимает место особи с самой низкой приспособленностью. Таким образом на каждом шаге в популяции обновляется только одна особь. Подводя итоги можно выделить следующие характерные особенности:

● Фиксированный размер популяции.

● Фиксированная разрядность генов.

● Особи для скрещивания выбираются случайным образом.

● Ограничений на тип кроссовера и мутации нет.

● В результате скрещивания особей получается один потомок, который занимает место наименее приспособленной особи.

3. Hybrid algorithm (L. "Dave" Davis)

Использование гибридного алгоритма позволяет объединить преимущества ГА с преимуществами классических методов. Дело в том, что ГА являются робастными алгоритмами, т.е. они позволяют находить хорошее решение, но нахождение оптимального решения зачастую оказывается намного более трудной задачей в силу стохастичности принципов работы алгоритма. Поэтому возникла идея использовать ГА на начальном этапе для эффективного сужения пространства поиска вокруг глобального экстремума, а затем взяв лучшую особь, применить один из "классических" методов оптимизации. Характеристики алгоритма:

● Фиксированный размер популяции.

● Фиксированная разрядность генов.

● Любые комбинации стратегий отбора и формирования следующего поколения

● Ограничений на тип кроссовера и мутации нет.

● ГА применяется на начальном этапе, а затем в работу включается классический метод оптимизации.

4. Island Model GA

Представим себе следующую ситуацию. В некотором океане есть группа близкорасположенных островов, на которых живут популяции особей одного вида. Эти популяции развиваются независимо и только изредка происходит обмен представителями между популяциями. Островная модель ГА использует описанный принцип для поиска решения. Вариант, безусловно, интересный и является одной из разновидностей параллельных ГА. Данная модель генетического алгоритма обладает следующими свойствами:

● Наличие нескольких популяций, как правило одинакового фиксированного размера.

● Фиксированная разрядность генов.

● Любые комбинации стратегий отбора и формирования следующего поколения в каждой популяции. Можно сделать так, что в разных популяциях будут использоваться разные комбинации стратегий, хотя даже один вариант дает разнообразные решения на различных "островах".

● Ограничений на тип кроссовера и мутации нет.

● Случайный обмен особями между "островами". Если миграция будет слишком активной, то особенности островной модели будут сглажены и она будет не очень сильно отличаться от моделей ГА без параллелизма.

5. CHC (Eshelman)

CHC расшифровывается как Cross-population selection, Heterogenous recombination and Cataclysmic mutation. Данный алгоритм довольно быстро сходится из-за того, что в нем нет мутаций, используются популяции небольшого размера, и отбор особей в следующее поколение ведется и между родительскими особями, и между их потомками. В силу этого после нахождения некоторого решения алгоритм перезапускается, причемлучшая особь копируется в новую популяцию, а оставшиеся особи являются сильной мутацией (мутирует примерно треть битов в хромосоме) существующих и поиск повторяется. Ещу одной специфичной чертой является стратегия скрещивания: все особи разбиваются на пары, причем скрещиваются только те пары, в которых хромосомы особей существенно различны (хэммингово расстояние больше некоторого порогового плюс возможны ограничения на минимальное расстояние между крайними различающимися битами). При скрещивании используется так называемый HUX-оператор (Half Uniform Crossover) - это разновидность однородного кроссовера, но в нем к каждому потомку попадает ровно половина битов хромосомы от каждого родителя. Таким образом, модель обладает следующими свойствами:

● Фиксированный размер популяции.

● Фиксированная разрядность генов.

● Перезапуск алгоритма после нахождения решения.

● Небольшая популяция.

● Особи для скрещивания разбиваются на пары и скрещиваются при условии существенных отличий.

● Отбор в следующее поколение проводится между родительскими особями и потомками.

● Используется половинный однородный кроссовер (HUX).

● Макромутация при перезапуске.