**ОПИСАНИЯ ЛАБОРАТОРНЫХ РАБОТ ПО КУРСУ ННСУИД**

[ВВЕДЕНИЕ 3](#_Toc25653765)

[1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ КУРСА 4](#_Toc25653766)

[1.1. Основы искусственных нейронных сетей 4](#_Toc25653767)

[1.2.Основы генетических алгоритмов 7](#_Toc25653768)

[1.3. Основы нечёткой логики 12](#_Toc25653769)

[1.4. Основы нейро-нечётких систем 23](#_Toc25653770)

[2. ОПИСАНИЯ ЛАБОРАТОРНЫХ РАБОТ 33](#_Toc25653771)

[Лабораторная работа №1. Изучение основ применения искусственных нейронных сетей 33](#_Toc25653772)

[Лабораторная работа №2. Изучение основ применения генетических алгоритмов 39](#_Toc25653773)

[Лабораторная работа № 3. Изучение основ применения нечёткой логики 41](#_Toc25653774)

[Лабораторная работа №4. Изучение основ применения нейро-нечётких систем 43](#_Toc25653775)

## ВВЕДЕНИЕ

В настоящее время всё большее распространение приобретают разработки в области управления и обработки информации, основанные на математических аппаратах искусственных нейронных сетей, нечёткой логики, генетических алгоритмов.

Среди задач обработки информации одной из немаловажных задач, которая может быть решена с применением указанными выше подходов является задача диагностики технических систем.

Цель учебного пособия – познакомить читателя с основами применения современных подходов в задачах управления и диагностики технических систем на примере выполнения лабораторных работ.

Выполнение лабораторных работ предусмотрено в программном пакете MatLab, широко распространённом и хорошо себя зарекомендовавшим во многих инженерных приложениях, что позволит получить полезный для практической работы навык.

## 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ КУРСА

# 1.1. Основы искусственных нейронных сетей

**1.1.1. Принципы функционирования**

Под искусственными нейронными сетями (ИНС) подразумеваются вычислительные структуры, которые моделируют простые биологические процессы, обычно ассоциируемые с процессами человеческого мозга. Они представляют собой распараллеленные системы, способные к обучению и адаптации путем анализа положительных и отрицательных воздействий. Элементарным преобразователем в данных сетях является искусственный нейрон или просто нейрон, названный так по аналогии с биологическим.

Нейрон — это составная часть нейронной сети. Нейрон состоит из умножителей (синапсов), сумматора и нелинейного преобразователя.

При помощи синапсов осуществляется связь между нейронами. При этом входной сигнал нейрона умножается на число, характеризующее силу связи, — вес синапса.

Сумматор осуществляет сложение сигналов, поступающих по синаптическим связям от других нейронов, и внешних входных сигналов. Нелинейный преобразователь реализует нелинейную функцию одного аргумента — выхода сумматора. Эта функция называется функцией активации нейрона. Нейрон в целом реализует скалярную функцию векторного аргумента.

Математическая модель нейрона может быть описана следующими соотношениями





В общем случае входной сигнал, весовые коэффициенты и значения смещения могут принимать действительные значения. Выход нейрона *у* определяется видом функции активации *f(s)* и может быть как действительным, так и целым. Во многих практических задачах входы, веса и смещения могут принимать лишь некоторые фиксированные значения

Таким образом, нейрон полностью описывается своими весами *wi* и передаточной функцией *f(s).* Для заданного набора чисел (вектора) *xi* в качестве входов, нейрон формирует некоторое число *у* на выходе.

Процесс функционирования ИНС зависит от значений синаптических связей. Поэтому, задав определенную структуру ИНС, необходимо найти значения всех переменных весовых коэффициентов (некоторые синаптические связи могут быть постоянными).

Этот этап называется обучением ИНС, и от того, насколько качественно он будет выполнен, зависит способность сети решать поставленные перед ней проблемы во время функционирования.

1.1.2.Обучение ИНС

Математически процесс обучения может быть описан следующим образом.

В процессе функционирования нейронная сеть формирует выходной сигнал **Y**, соответствующий входному сигналу **X**, реализуя некоторую функцию **Y = G(X)**.

Пусть решением некоторой задачи является функция **Y = F(X)**, которая представлена известными парами входных-выходных данных (**X1, Y1**), (**X2, Y2**), ..., (**XN, *YN****)*,где ***Yk* = *F(Xk)*** *(k* = 1, 2, ...*N*).

Процесс обучения заключается в поиске функции **G**, близкой к **F** в смысле некоторой функции ошибки ***Е***.

Если выбраны множество обучающих примеров — пар **(X*k*, *Yk)***(где *k* = 1,2,..., *N)* и способ вычисления функции ошибки **Е***,* то обучение нейронной сети превращается в задачу многомерной оптимизации, имеющую очень большую размерность. Т.к. функция **Е**может иметь произвольный вид, обучение в общем случае — многоэкстремальная невыпуклая задача оптимизации.

Для решения этой задачи могут быть использованы следующие (итерационные) алгоритмы:

• алгоритмы локальной оптимизации с вычислением частных производных первого порядка;

• алгоритмы локальной оптимизации с вычислением частных производных первого и второго порядка;

• стохастические алгоритмы оптимизации;

• алгоритмы глобальной оптимизации.

К первой группе относятся: градиентный алгоритм (метод скорейшего спуска); методы с одномерной и двумерной оптимизацией целевой функции в направлении антиградиента; метод сопряженных градиентов; методы, учитывающие направление антиградиента на нескольких шагах алгоритма.

Ко второй группе относятся: метод Ньютона, методы оптимизации с разреженными матрицами Гессе, квазиньютоновские методы, метод Гаусса Ньютона, метод Левенберга-Марквардта и др.

Стохастическими методами являются: поиск в случайном направлении, имитация отжига, метод Монте-Карло (численный метод статистических испытаний).

Задачи глобальной оптимизации решаются с помощью перебора значений переменных, от которых зависит целевая функция (функция ошибки **Е**)*.*

1.1.3. Переобучение и обобщение

В отсутствие полной и бесконечно большой обучающей выборки на самом деле, минимизируется не «истинная» ошибка на поверхности ошибок в заранее неизвестной модели явления, а только ошибка относительно имеющихся данных.

Сильнее всего это различие проявляется в явлении слишком близкой подгонки параметров ИНС под данные обучающей выборки, называемом переобучением. Это явление хорошо иллюстрируется на примере аппроксимации посредством полиномов.

Графики полиномов могут иметь различную форму. При этом, чем выше его степень, тем более сложной может быть эта форма. Полиномиальная кривая (модель) может быть подогнана под конкретные данные и таким образом объяснить существующую зависимость.

При этом данные могут быть зашумлены, поэтому нельзя считать, что самая лучшая модель задается кривой, которая в точности проходит через все имеющиеся точки данных. Полином низкого порядка может быть недостаточно гибким средством для аппроксимации данных, в то время как полином высокого порядка может оказаться чересчур гибким, и будет точно следовать данным, принимая при этом замысловатую форму, не имеющую никакого отношения к форме настоящей зависимости.

Нейронная сеть сталкивается с точно такой же трудностью. ИНС с большим числом весов моделируют более сложные функции и, следовательно, склонны к переобучению. ИНС с небольшим числом весов может оказаться недостаточно гибкой, чтобы смоделировать имеющуюся зависимость.

Для борьбы с переобучением используется механизм контрольной кросс-проверки, при котором часть обучающих наблюдений резервируется и в обучении не используется. Вместо этого она используется для независимого контроля результата в процессе осуществления обучения сети.

В самом начале обучения ошибки сети на обучающем и контрольном множествах будут одинаковыми (если они существенно отличаются, то, вероятно, разбиение всех наблюдений на два множества было неоднородным). По мере того как сеть обучается, ошибка обучения будет убывать, и, пока обучение уменьшает действительную функцию ошибок, ошибка на контрольном множестве также будет убывать. Если же контрольная ошибка перестала убывать или даже стала расти, это означает, что сеть начала слишком близко аппроксимировать данные и обучение следует остановить. Этот эффект слишком точной аппроксимации в процессе обучения и называется переобучением. Если такое случилось, то обычно советуют уменьшить число скрытых элементов и/или слоев, ибо сеть является слишком мощной для данной задачи. Если же сеть, наоборот, была взята недостаточно богатой для того, чтобы моделировать имеющуюся зависимость, то переобучения, скорее всего, не произойдет, и обе ошибки — обучения и проверки — не достигнут достаточного уровня малости.

# 1.2.Основы генетических алгоритмов

1.2.1. Генетические алгоритмы и традиционные методы оптимизации

Генетический алгоритм представляет собой метод, отражающий естественную эволюцию методов решения проблем, и в первую очередь задач оптимизации. Генетические алгоритмы - это процедуры поиска, основанные на механизмах естественного отбора и наследования. В них используется эволюционный принцип выживания наиболее приспособленных особей.

Генетические алгоритмы отличаются от традиционных методов оптимизации несколькими базовыми элементами. В частности, они:

1) обрабатывают не значения параметров самой задачи, а их закодированную форму;

2) осуществляют поиск решения исходя не из единственной точки, а из их некоторой популяции;

3) используют только целевую функцию, а *не* ее производные либо иную дополнительную информацию,

4) применяют вероятностные, а не детерминированные правила вывода.

1.2.2. Основные понятия генетических алгоритмов

В теории генетических алгоритмов используются определения, заимствованные из генетики

Например, речь идет о *популяции особей, а в* качестве базовых понятий применяются *ген, хромосома, генотип, фенотип,* аллель.

Также используются соответствующие этим терминам определения из технического лексикона, в частности, цель, *двоичная последовательность, структура.*

*Популяция -* это конечное множество *особей.*

*Особи, входящие в популяцию*, вгенетических алгоритмах представляются хромосомами с закодированными в них множествами *параметров* задачи, т.е. решений, которые иначе называются точками *в пространстве поиска.* В некоторых работах особи называются организмами.

Хромосомы(другие названия - цепочки или *кодовые последовательности)* - это упорядоченные последовательности *генов.*

*Ген* (также называемый *свойством, знаком* или детектором) - это атомарный элемент *генотипа,* в частности, хромосомы.

*Генотип* или структура - это набор хромосом данной особи. Следовательно, особями популяции могут быть генотипы либо единичные хромосомы (в довольно распространенном случае, когда генотип состоит из одной хромосомы).

*Фенотип -* это набор значений, соответствующих данному генотипу, т.е. *декодированная структура* или множество параметров задачи (решение, точка *пространства поиска).*

*Аллель -* это значение конкретного гена, также определяемое как *значение* свойстваили *вариант свойства.*

*Локус* или п*озиция* указывает место размещения данного гена в хромосоме (цепочке). Множество позиций генов - это *локи.*

Очень важным понятием в генетически алгоритмах считается *функция приспособленности (fitness function),* иначе называемая *функцией оценки:*

Представляет меру приспособленности данной особи в популяции.

Играет важнейшую роль, поскольку позволяет оценить степень приспособленности конкретных особей в популяции и выбрать из них наиболее приспособленные (т.е. имеющие наибольшие значения функции приспособленности) в соответствии с эволюционным принципом выживания «сильнейших» (лучше всего приспособившихся).

Оказывает сильное влияние на функционирование генетических алгоритмов и должна иметь точное и корректное определение.

В задачах оптимизации функция приспособленности, как правило, оптимизируется (точнее говоря, максимизируется) и называется *целевой функцией.* В задачах минимизации целевая функции преобразуется, и проблема сводится кмаксимизации.

В теории управления функция приспособленности может принимать вид *функции погрешности,* а в теории игр - *стоимостной функции.* На каждой итерации генетического алгоритма приспособленность каждой особи данной популяции оценивается при помощи функции приспособленности, и на этой основе создается следующая популяция особей

Очередная популяция в генетическом алгоритме называется поколением, а к вновь создаваемой популяции особей применяется термин «новое поколение» или «поколение потомков».

1.2.3. Классический генетический алгоритм

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

1) инициализация, или выбор исходной популяции хромосом;

2) оценка приспособленности хромосом в популяции;

3) проверка условий остановки алгоритма;

4) селекция хромосом;

5) применение генетических операторов;

6) формирование новой популяции;

7) выбор «наилучшей» хромосомы,

Формирование исходной популяции заключается в случайном выборе заданного количества хромосом (особей), представляемых двоичными последовательностями фиксированной длины.

Оценивание приспособленности хромосом заключается в расчете функции приспособленности для каждой хромосомы этой популяции. Чем больше значение этой функции, тем выше «качество» хромосомы. Форма функции приспособленности зависит от характера решаемой задачи. Предполагается, что функция приспособленности всегда принимает неотрицательные значения и. кроме того, что для решения оптимизационной задачи требуется максимизировать эту функцию. Если исходная форма функции приспособленности не удовлетворяет этим условиям, то выполняется соответствующее преобразование (например, задачу минимизации функции можно легко свести к задаче максимизации).

Определение условий остановки генетического алгоритма зависит от его конкретного применения. В оптимизационных задачах, если известно максимальное (или минимальное) значение функции приспособленности, то остановка алгоритма может произойти после достижения ожидаемого оптимального значения, возможно - с заданной точностью. Остановка алгоритма также может произойти в случае, когда его выполнение не приводит к улучшению уже достигнутого значения. Алгоритм может быть остановлен по истечении определенного времени выполнения либо после выполнения заданного количества итераций. Если условие остановки выполнено, то производится переход к завершающему этапу выбора "наилучшей" хромосомы. В противном случае на следующем шаге выполняется селекция.

Селекция хромосом заключается в выборе (по рассчитанным на втором этапе значениям функции приспособленности) тех хромосом, которые будут участвовать в создании потомков для следующей популяции, т.е. для очередного поколения. Такой выбор производится согласно принципу естественного отбора, по которому наибольшие шансы на участие в создании новых особей имеют хромосомы с наибольшими значениями функции приспособленности. Существуют различные методы селекции. Наиболее популярным считается так называемый метод *рулетки (roulette wheel selection),* который свое название получил по аналогии с известной азартной игрой. Каждой хромосоме может быть сопоставлен сектор колеса рулетки, величина которого устанавливается пропорциональной значению функции приспособленности данной хромосомы. Поэтому чем больше значение функции приспособленности, тем больше сектор на колесе рулетки.

Всё колесо рулетки соответствует сумме значений функций приспособленности всех хромосом рассматриваемой популяции. Каждой хромосоме, обозначаемой *chi*, для *i = 1, 2,..., N* (где *N* обозначает численность популяции) соответствует сектор колеса *v(chi),* выраженный в процентах согласно формуле



где 

причем *F(chi)* - значение функции приспособленности хромосомы *chi*, а *рs(chi)* - *вероятность селекции* хромосомы *chi*. Селекция хромосомы может быть представлена как результат поворота колеса рулетки, поскольку «выигравшая» (т.е. выбранная) хромосома относится к выпавшему сектору этого колеса.

Очевидно, что чем больше сектор, тем больше вероятность «победы» соответствующей хромосомы. Поэтому вероятность выбора данной хромосомы оказывается пропорциональной значению ее функции приспособленности.

Если всю окружность колеса рулетки представить в виде цифрового интервала [0, 100], то выбор хромосомы можно отождествить с выбором числа из интервала [а, b], где а и bобозначают соответственно начало и окончание фрагмента окружности, соответствующего этому сектору колеса; очевидно, что *0 ≤ a < b ≤ 100*. В этом случае выборе помощью колеса рулетки сводится к выбору числа из интервала [0,100], которое соответствует конкретной точке на окружности колеса.

В результате процесса селекции создается *родительская популяция,* также называемая *родительским пулом (mating poof) с* численностью N, равной численности текущей популяции.

**Применение генетических операторов** к хромосомам, отобранным с помощью селекции, приводит к формированию новой популяции потомков от созданной на предыдущем шаге родительской популяции.

В классическом генетическом алгоритме применяются два основных генетических оператора: *оператор скрещивания (crossover)* и *оператор мутации (mutation).* Однако следует отметить, *что* оператор мутации играет явно второстепенную роль по сравнению с оператором скрещивания. Это означает, что скрещивание в классическом генетическом алгоритме производится практически всегда, тогда как мутация - достаточно редко. Вероятность скрещивания, как правило, достаточно велика (обычно 0,5 *< рс <* 1), тогда как вероятность мутации устанавливается весьма малой (чаще всего 0 < рm <0,1). Это следует из аналогии с миром живых организмов, где мутации происходят чрезвычайно редко

В генетическом алгоритме мутация хромосом может выполняться на популяции родителей перед скрещиванием либо на популяции потомков, образованных в результате скрещивания.

На первом этапе скрещивания выбираются пары хромосом из родительской популяции (родительского пула). Это временная популяция, состоящая из хромосом, отобранных в результате селекции и предназначенных для дальнейших преобразований операторами скрещиваний и мутации с целью формирования новой популяции потомков. На данном этапе хромосомы из родительской популяции объединяются в пары. Это производится случайным способом в соответствии с вероятностью скрещивания pc, Далее для каждой пары отобранных таким образом родителей разыгрывается позиция гена (локус) *в* хромосоме, определяющая так называемую *точку скрещивания.*

Если хромосома каждого из родителей состоит из *L* генов, то очевидно, что точка скрещиваний *lk* представляет собой натуральное число, меньшее *L.* Поэтому фиксация точки скрещивания сводится к случайному выбору числа из интервала *[1, L-1]*. В результате скрещивания пары родительских хромосом получается следующая пара потомков:

1) потомок, хромосома которого на позициях от 1 до *lk* состоит из генов первого родителя, *а* на позициях от *lk + 1* до *L* — из генов второго родителя;

*2)* потомок, хромосома которого на позициях от 1 до *lk* состоит из генов второго родителя, а на позициях от *!к* + 1 до *L* - из генов первого родителя.

Оператор мутации с вероятностью *рm,* изменяет значение гена в хромосоме на противоположное (т.е. с *0* на 1 или обратно). Например, если в хромосоме [100110101010] мутации подвергается ген на позиции 7, то его значение, равное 1, изменяется на 0. что приводит к образованию хромосомы [100110001010]. Как уже упоминалось выше, вероятность мутации обычно очень мала, и именно от нее зависит, будет данный ген мутировать или нет. Вероятность *рm* мутации может эмулироваться, например, случайным выбором числа из интервала [0, 1] для каждого гена и отбором для выполнения этой операции тех генов, для которых разыгранное число оказывается меньшим или равным значению *рm*.

Хромосомы, полученные в результате применения генетических операторов к хромосомам временной родительской популяции, включаются в состав новой популяции. Она становится так называемой текущей популяцией для данной итерации генетического алгоритма. На каждой очередной итерации рассчитываются значения функции приспособленности для всех хромосом этой популяции, после чего проверяется условие остановки алгоритма и либо фиксируется результат в виде хромосомы с наибольшим значением функции приспособленности, либо осуществляется переход к следующему шагу генетического алгоритма, т.е. к селекции В классическом генетическом алгоритме вся предшествующая популяция хромосом замещается новой популяцией потомков, имеющей ту же численность.

Если условие остановки алгоритма выполнено, то следует вывести результат работы, т.е. представить искомое решение задачи. Лучшим решением считается хромосома с наибольшим значением функции приспособленности.

# 1.3. Основы нечёткой логики

1.3.1. Введение

Работа Лотфи Заде «Fuzzy Sets», появившаяся в 1965 г. в журнале Information and Control, № 8, заложила основы моделирования интеллектуальной деятельности человека и явилась начальным толчком к развитию новой математической теории.

Л. Заде расширил классическое канторовское понятие множества, допустив, что характеристическая функция (функция принадлежности элемента множеству) может принимать любые значения в интервале [0; 1], а не только значения 0 либо 1. Такие множества были названы им нечеткими (fuzzy). Он определил также ряд операций над нечеткими множествами и предложил обобщение известных методов логического вывода modus ponens и modus tollens.

Введя затем понятие *лингвистической переменной* и допустив, что в качестве ее значений (термов) выступают нечеткие множества, Л. Заде создал аппарат для описания процессов интеллектуальной деятельности, включая нечеткость и неопределенность выражений.

Математическая теория нечетких множеств позволяет описывать множество *Е,* элементы которого удовлетворяют свойству R, в виде упорядоченных пар.

*А* = *{µA(x)/x},* где *µA(x)* — *характеристическая функция,* принимающая значение 1, если *х* удовлетворяет свойству R, и 0 — в противном случае. Нечеткое подмножество отличается от обычного тем, что для элементов *x* из E нет однозначного ответа «да нет» относительно свойства R. В связи с этим нечеткое подмножество *А* универсального множества *Е* определяется как множество упорядоченных пар

*А = {µА(х)/х},* где *µA(x)* — характеристическая *функция принадлежности* (или просто *функция принадлежности)*, принимающая значения в некотором вполне упорядоченном множестве *М* (например, *М* = **[0, 1])**

Функция принадлежности указывает степень (или уровень) принадлежности элемента *х* подмножеству *А.* Множество *М* называют множеством принадлежностей. Если *М* = {0, 1}, то нечеткое подмножество *А* может рассматриваться как обычное или четкое множество

1.3.2. Нечеткая и лингвистическая переменные

Понятие нечеткой и лингвистической переменных используется при описании объектов и явлений с помощью нечетких множеств.

*Нечеткая переменная* характеризуется тройкой (α, X, *А),* где

*α* - наименование переменной;

*X* - универсальное множество (область определения α);

*А* - нечеткое множество на *X*, описывающее ограничения (т.е. *µА(х))* на значения нечеткой переменной α*.*

*Лингвистической* переменной (ЛП) называется набор

<β, Т, X, G, М>, где

β — наименование лингвистической переменной;

*Т* — множество ее значений (терм-множество), представляющих собой наименования нечетких переменных, областью определения каждой из которых является множество *X.* Множество *Т* называется базовым т*ерм-множеством* лингвистической переменной;

G - синтаксическая процедура, позволяющая оперировать элементами терм-множества Т, в частности, генерировать новые термы (значения). Множество T G(T), где *G(T)* — множество сгенерированных термов, называется *расширенным терм-множеством* лингвистической переменной;

*М* - семантическая процедура, позволяющая превратить каждое новое значение лингвистической переменной, образуемое процедурой G, в нечеткую переменную, т.е. сформировать соответствующее нечеткое множество.

Замечание. Чтобы избежать большого количества символов:

1) символ β используют как для названия самой переменной, так и для всех ее значений;

2) пользуются одним и тем же символом для обозначения нечеткого множества и его названия, например терм «Высокая», являющийся значением лингвистической переменной *(*β= «цена», одновременно есть и нечеткое множество *М* («Высокая»).

Присвоение нескольких значений символам предполагает, что условия задачи позволяют разрешить возможные неопределенности.

Пример. Пусть эксперт определяет скорость движения автомобиля с помощью понятий «Малая скорость», «Средняя скорость» и «Большая скорость», при этом минимальная толщина равна 0 км/ч, а максимальная — 100 км/ч.

Формализация такого описания может быть проведена с помощью следующей лингвистической переменной <β, Т, X, G, М>*,* где

β— скорость;

*Т* — {«Малая скорость», «Средняя скорость», «Большая скорость»};

*X —* [0, 100];

G — процедура образования новых термов с помощью связок «и», «или» и модификаторов типа «очень», «не», «немного», «слегка» и т.п. Например: «Малая или средняя скорость», «Очень малая скорость» и т.д.;

*М* — процедура задания на *X* = [0, 100] нечетких подмножеств *А1=* «Малая скорость», *А2 =* «Средняя скорость», *А*3 = «Большая скорость», а также нечетких множеств для термов из G(T) в соответствии с правилами трансляции нечетких связок и модификаторов «и», «или», «не», «очень», «слегка» и других операций над нечеткими множествами вида:

 и т.п.

1.3.3. Нечеткие выводы

Используемый в различного рода экспертных и управляющих системах механизм нечетких выводов в своей основе имеет базу знаний, формируемую специалистами предметной области в виде совокупности нечетких предикатных правил вида:

*FR*1: Если *х* есть *А*1, то *у* есть *C*1,

*FR*2: Если *х* есть *А*2, то *у* есть *C*2,

.......................

*FRn:* Если *х* есть *Аn* то *у* есть Cn

В указанных правилах:

*х* – входная переменная (имя для известных значений данных),

*у* – переменная вывода (имя для значения данных, которое будет вычислено);

*А* и *C* – термы с функциями принадлежности, определенными соответственно на *х* и *у.*

Пример подобного правила:

Если *х* – низко, то *у* – высоко.

Знание эксперта *А* → *C* отражает нечеткое причинное отношение предпосылки и заключения, поэтому его можно назвать нечетким отношением и обозначить через *R:*

*R = A* → *C*,

Где “→” называют нечёткой импликацией.

Импликацией называется вид отношения, имеющего форму правила, используемого при рассуждениях. Различают классическую и нечеткую импликации.

Классическая импликация выражается с помощью соотношения:

Если *р* то *q*,

где *р* – утверждение, называемое антецедентом (условием), *q* – утверждение, называемое консеквентом (заключением).

Нечеткая импликация представляет собой правило, условие (антецедент) и заключение (конceквент) заданы нечеткими множествами, со своими функциями принадлежности и областями определения соответственно.

Отношение R можно рассматривать как нечеткое подмножество прямого произведения *Х* х *Y* полного множества предпосылок *X* и заключений *Y*.

Таким образом, процесс получения (нечеткого) результата вывода *C'* с использованием данного наблюдения *А’* и знания *A* → *C* можно представить в виде формулы



где «○» - операция свертки

Как операцию композиции, так и операцию импликации в алгебре нечетких множеств можно реализовывать по-разному (при этом, естественно, будет разниться и итоговый получаемый результат), но в любом случае общий логический вывод осуществляется за четыре этапа:

1. Фаззификация
2. Логический вывод
3. Композиция
4. Дефаззификация

Фаззификация

Функции принадлежности, определенные для входных переменных применяются к их фактическим значениям для определения степени истинности каждой предпосылки каждого правила.

Логический вывод

Вычисленное значение истинности для предпосылок каждого правила применяется к заключениям каждого правила.

Это приводит к одному нечеткому подмножеству, которое будет назначено каждой переменной вывода для каждого правила.

В качестве правил логического вывода обычно используются только операции min (минимум) или prod (умножение).

В логическом выводе на основе операции минимума функция принадлежности вывода «отсекается» по высоте, соответствующей вычисленной степени истинности предпосылки правила (нечеткая логика «И»).

В логическом выводе на основе операции умножения функция принадлежности вывода масштабируется при помощи вычисленной степени истинности предпосылки правила.

Композиция

Все нечеткие подмножества, назначенные к каждой переменной вывода (во всех правилах), объединяются вместе, чтобы формировать одно нечеткое подмножество для каждой переменной вывода.

При подобном объединении обычно используются операции максимума или суммирования.

При композиции на основе операции максимума комбинированный вывод нечеткого подмножества конструируется как поточечный максимум по всем нечетким подмножествам (нечеткая логика «ИЛИ»).

При композиции на основе операции суммирования комбинированный вывод нечеткого подмножества конструируется как поточечная сумма по всем нечетким подмножествам, назначенным переменной вывода правилами логического вывода.

Дефаззификация

На данном этапе на основе полученного в результате композиции нечёткого множества производится вычисление численного значения нечёткого вывода (четкое значение выходной переменной), т.е. генерация некоторой однозначно определенной численной величины *y*, отображающей своим значением композицию нечетких подмножеств, соответствующих выходной переменной.

В основу синтеза дефаззификатора могут быть положены различные подходы, простейший из которых состоит в генерации такого выходного численного значения *ymax*, которое соответствует достижимому максимуму на множестве нечётких переменных в составе выходного нечёткого множества *B****,*** где заданные значения входных переменных активируют *r* решающих правил *FR*:



Таким образом, выбирается тот элемент нечеткого множества, который имеет наивысшую степень принадлежности этому множеству. Если этот элемент не является единственным, т.е. функция принадлежности имеет несколько локальных максимумов, или если имеется максимальное «плато», то выбор среди элементов, имеющих наивысшую степень принадлежности множеству, осуществляется на основе некоторого критерия. В зависимости от критерия, можно выделить несколько разновидностей данного метода:

1. Метод левого максимума – выбирается наименьшее из чисел y1**,** y2**, .. ,**yn, имеющих наивысшую степень принадлежности выходному нечеткому множеству.



1. Метод правого максимума – выбирается наибольшее из чисел y1**,** y2**, .. ,**yn, имеющих наивысшую степень принадлежности выходному нечеткому множеству.



Метод среднего из максимумов – в качестве искомого четкого значения *yo* принимается среднее арифметическое координат локальных максимумов.

Четкое значение находится по формуле:



где G – подмножество элементов, максимизирующих *С(y)*.

Также чёткое значение может выбираться произвольно среди множества элементов, доставляющих максимум *C*:



Высотная дефаззификация. Элементы области определения Ω, для которых значения функции принадлежности меньше, чем некоторый уровень *α* в расчет не принимаются, и четкое значение рассчитывается по формуле:



где *Y*α — нечеткое множество α-уровня.

Нетрудно заметить, что представленные подходык дефаззификации игнорируют дополнительную доступную информацию по конфигурации нечёткого множества. Например, во многих практических задачах выходное нечёткого множество является существенно асимметричным, и соответствующая информация может представлять определенное значение при генерации численной выходной переменной *y*.

В связи с этим, более эффективным с точки зрения полноты использования априорной информации, представляется подход на основе нечеткого центрирования. При этом *yo*определяетсякак горизонтальная составляющая координат центра физической массы фигуры, распределенной в двухмерном пространстве, которую образует выходное нечёткое множество. Такой подход носит название центроидный метод дефаззификации и использует следующую формулу.



1.3.4. Алгоритм Мамдани

Рассмотрим систему с базой знаний с двумя нечеткими правилами вида:

*FR1*: если *х1* есть *A1* и *x2* есть *B1*, тогда *y* есть *C1*,

*FR2*: если *х1* есть *А2* и *x2* есть *В2*, тогда *y* есть *С2*,

где *х1* и *x2* –входные переменные, *y* – переменная вывода.

*A1, A2, B1, B2, C1, C2* – некоторые заданные термы с функциями принадлежности  соответственно. при этом четкое значение *y0* необходимо определить на основе приведенной информации и четких значений *х10* и x20.

Этапы нечёткого вывода по алгоритму Мамдани могут быть описаны следующим образом.

Фаззификация

Находятся степени истинности для предпосылок каждого правила:  путём подстановки числовых значений входных переменных в соответствующие функции принадлежности.

Нечеткий вывод

Находятся уровни «отсечения» для предпосылок каждого из правил (с использованием операции минимум) где через «» обозначена операция логического минимума:



Затем находятся «усеченные» функции принадлежности заключений правил:



Композиция

С использование операции максимума (далее обозначаемой как «») производится объединение найденных усеченных функций, что приводит к получению итогового нечеткого подмножества для переменной выхода с функцией принадлежности:



Дефаззификация

Нахождение численного значения *y0* проводится, например, центроидным методом по формуле



1.3.5. Алгоритм Ларсена

Рассмотрим систему с базой знаний с двумя нечеткими правилами вида:

*FR1:* если *х1* есть *A1* и *x2* есть *B1*, тогда *y* есть *C1*,

*FR2:* если *х1* есть *А2* и *x2* есть *В2*, тогда *y* есть *С2*,

где *х1* и *x2* – входные переменные, *y* – переменная вывода.

*A1, A2, B1, B2, C1, C2* — некоторые заданные термы с функциями принадлежности  соответственно, при этом четкое значение *y0* необходимо определить на основе приведенной информации и четких значений *х10* и x20

Этапы нечёткого вывода по алгоритму Ларсена могут быть описаны следующим образом.

Фаззификация

Находятся степени истинности для предпосылок каждого правила:  *(аналогично алгоритму Мамдани)*.

Нечеткий вывод

Как в алгоритме Мамдани вначале находятся значения степеней истинности правил:



Однако нечеткая импликация реализуется с использованием оператора умножения (а не минимума):



Композиция

На данном этапе находится итоговое нечеткое подмножество с функцией принадлежности с использованием операции максимума, что приводит к получению итогового нечеткого подмножества для переменной выхода с функцией принадлежности



Дефаззификация

Производится одним из рассмотренных ранее методов. Например, аналогично алгоритму Мамдани, центроидным методом.

1.3.6. Упрощенный алгоритм нечеткого вывода

Исходные правила в данном случае задаются в виде:

*FR1*: если *х1* есть A1 и *x2* есть B1, тогда *y1 = c1*,

*FR2*: если *х1* есть А2 и *x2* есть В2, тогда *y2 = с2*,

где *х1* и *x2* —входные переменные, *y* – переменная вывода.

*A1, A2, B1, B2,* – некоторые заданные термы с функциями принадлежности , *c1* и *с2* – некоторые обычные (четкие) числа, при этом четкое значение *y0* необходимо определить на основе приведенной информации и четких значений *х1 0* и *x*2 0

Этапы нечёткого вывода по упрощённому алгоритму могут быть описаны следующим образом.

Фаззификация

Находятся степени истинности для предпосылок каждого правила:  *(аналогично алгоритму Мамдани)*.

Нечёткий вывод

На втором этапе находятся числа, определяющие степень истинности правил:



Композиция и дефаззификация

На данном этапе находится четкое значение выходной переменной по формуле:



В общем случае для *n* правил значение будет находиться следующим образом:



1.3.7. Алгоритм Цукамото

Рассмотрим систему с базой знаний с двумя нечеткими правилами вида:

*FR1:* если *х1* есть *A1* и *x2* есть *B1*, тогда *y* есть *C1*,

*FR2:* если *х1* есть *А2* и *x2* есть *В2*, тогда *y* есть *С2*,

где *х1* и *x2* – входные переменные, *y* – переменная вывода.

*A1, A2, B1, B2, C1, C2* — некоторые заданные термы с функциями принадлежности  соответственно. При этом четкое значение *y0* необходимо определить на основе приведенной информации и четких значений *х10* и x20.

В алгоритме Цукамото функции задаются монотонными.

Этапы нечёткого вывода по упрощённому алгоритму могут быть описаны следующим образом.

Фаззификация

Находятся степени истинности для предпосылок каждого правила путём подстановки числовых значений входных переменных в соответствующие функции принадлежности.

Нечеткий вывод и композиция

На данном этапе сначала находятся (как в алгоритме Мамдани) уровни «отсечения» *α1* и *α2,* а затем — посредством решения уравнений

 – четкие значения *(yC1* и *yC2)* для каждого из исходных правил.

Дефаззификация

Чёткое значение переменной вывода определяется (как взвешенное среднее *y1* и *y2)*



В общем случае выходное численное значение (дискретный вариант центроидного метода) находится по формуле:



1.3.8. Алгоритм Сугено-Такаги

Рассмотрим систему с базой знаний с двумя нечеткими правилами. Особенностью алгоритма Сугено-Такаги является следующий вид правил

*FR1*: если *х1* есть A1 и *x2* есть B1, тогда *yC1 = a1x + b1y*

*FR2:* если *х1* есть А2 и *x2* есть В*2*, тогда *yC2 = a2x +b2y*

где *х1* и *x2* –входные переменные, *y* –переменная вывода.

*A1, A2, B1, B2, C1, C2* — некоторые заданные термы с функциями принадлежности. При этом четкое значение *y0* необходимо определить на основе приведенной информации и четких значений *х10* и *х20*.

Математически алгоритм может быть описан следующим образом:

Фаззификация

Находятся степени истинности для предпосылок каждого правила: *A1(x0), А2(x0), B1(y0), В2(y0)* (аналогично алгоритму Мамдани).

Нечеткий вывод

Находятся степени истинности для предпосылок каждого из правил (с использованием операции минимум или умножения)



Также для заданных входных значений *х10* и x20 вычисляютсяиндивидуальные выходы правил:



Дефаззификация

Итоговое значение переменной вывода определяется (как взвешенное среднее *yC1\** и *yC2\*)*



# 1.4. Основы нейро-нечётких систем

1.4.1. Введение

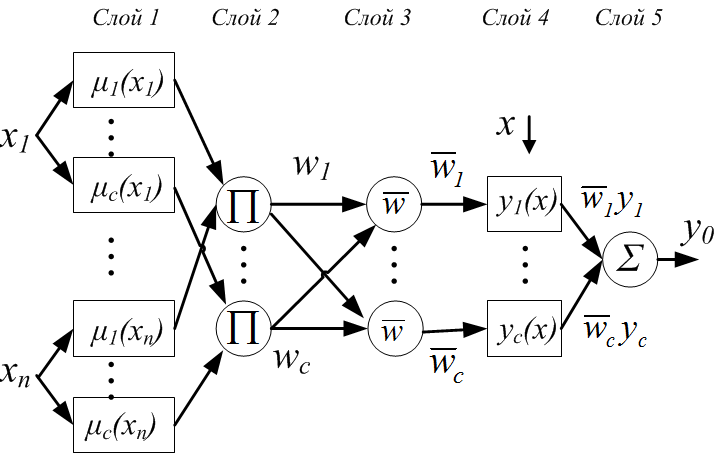
Нечёткие системы обладают определёнными достоинствами. В условиях недостатка формализованных данных за счёт использования осмысленных лингвистически заданных терминов позволяют достаточно просто интерпретировать получаемые результаты. Тем не менее, они обладают низкой гибкостью по отношению к изменениям в исследуемой системе и слабо приспособлены к самообучению, т.е. к автоматической подстройке параметров функций принадлежности на основе вновь получаемой информации.

Вместе с тем, с такой задачей могут успешно справляться искусственные нейронные сети, которые в отличие от систем нечёткого вывода могут настраивать свои параметры (весовые коэффициенты) на основе выборки данных без привлечения эксперта. Однако, для корректной настройки им требуется информативная выборка, и в случае недостатка информации, экспертные оценки не могут быть напрямую учтены при настройке. Также реализуемые ими функции часто не поддаются понятной интерпретации, что может вызвать недоверие к получаемому результату.

Предлагается комбинировать аппараты нечёткой логики и ИНС, чтобы объединить преимущества каждого из них, и при этом компенсировать их недостатки. Самым эффективным способом сочетания нечёткой логики и ИНС являются нейро-нечеткие системы, которые с одной стороны, могут рассматриваться как системы нечёткого вывода (а значит, понятно интерпретировать получаемые результаты), а с другой – как ИНС, состоящими из нейронов специального вида (а значит, могут обучаться). Таким образом, на первоначальном этапе в случае отсутствия достаточной информации такую систему можно рассматривать и конструировать как нечёткую систему на основе экспертных оценок, а в процессе накопления информации донастраивать методами искусственных нейронных сетей.

1.4.2. Структура нейро-нечётких систем

Как уже было сказано ранее, для максимизации достоинств обоих подходов в теории нейро-нечётких систем предложено рассматривать нечёткую систему типа Сугено в виде эквивалентной ей ИНС специального вида, которая изображена на рис.1.1.

****

**Рис.1.1. Структура нейро-нечёткой системы**

**Слой 1** . Содержит настраиваемые нейроны. Входами являются входные лингвистические переменные модели. Функция активации каждого из нейронов является функцией принадлежности терма лингвистической переменной.



**Слой 2** . Является неадаптивным слоем. Выполняет операцию умножения входных сигналов. Выходами слоя являются степени активации каждого правила



**Слой 3** . Состоит из неадаптивных узлов, в которых вычисляются степени активации (веса) *i*-го правила относительно степеней активации правил – средневзвешенную степень активации правила



**Слой 4** состоит из адаптивных узлов, передаточная функция которых представляет собой полином вида  (аналогично алгоритму Сугено).

По сути, в **узлах** данного слоя рассчитывается заключение каждого из правил, которое умножается на средневзвешенную степень активации  соответствующего правила

*, i=1..c*

**Слой 5** **состоит** из единственного неадаптивного узла, который вычисляет конечное значение выходной переменной сети как сумму всех взвешенных, с учётом соответствующей степени активации, заключений каждого из правил

****

При этом, с точки зрения нечётких систем, этапы функционирования подобной системы аналогичны алгоритму Сугено-Такаги.

Рассмотрим систему с базой знаний с двумя нечеткими правилами.

*FR1*: если *х1* есть A1 и *x2* есть B1, тогда *y1 = g1,1x1 + g2,1x2+h1*

*FR2*: если *х1* есть А2 и *x2* есть В*2*, тогда *y2 = g1,2x1 +g2,2x2+h2*

где *х1* и *x2* – входные переменные, *y* – переменная вывода.

*A1, A2, B1, B2,* — некоторые заданные термы с функциями принадлежности, при этом четкое значение *y0* необходимо определить на основе приведенной информации и четких значений *х1 0* и *x*2 0

Математически алгоритм может быть описан следующим образом:

Фаззификация

Находятся степени истинности для предпосылок каждого правила: *A1(x0), А2(x0), B1(y0), В2(y0)* (аналогично алгоритму Мамдани)

Нечеткий вывод

Находятся степени истинности для предпосылок каждого из правил (с использованием операции минимум или умножения)



Также для заданных входных значений *х10* и *x20* *вычисляются* индивидуальные выходы правил





Дефаззификация

Чёткое значение переменной вывода определяется (как взвешенное среднее *y1\** и *y2\*)*



С учётом обозначения , , получим

,

тем самым показав эквивалентность рассмотрения подобных систем, как с точки зрения нечёткой логики, так и искусственных нейронных сетей.

1.4.3. Построение нейро-нечёткой системы

Построение нейро-нечётких систем можно производить как на основе экспертных суждений (аналогично нечётким моделям), так и на основе имеющейся выборки данных (как в случае с ИНС). Встречаются и гибридные подходы для решения указанной задачи, например, когда первоначальная структура задаётся экспертами, а дальнейшая настройка осуществляется методами обучения ИНС.

Рассмотрим более подробно методы настройки ННС на основе выборки данных.

Учитывая, что нейро-нечёткая система представляет собой ИНС специального вида с фиксированным и определённым количеством слоёв, методы их настройки будут иметь некоторые особенности.

Создание таких систем можно разделить на два основных этапа: непосредственно их построение (этап структурной настройки), и их обучение (этап параметрической настройки).

На первом этапе настраивается структура системы – определяется количество решающих правил, термов входных переменных. При этом могут определяться или задаваться начальные значения параметров функций принадлежности термов входных переменных, а также выражений в заключениях правил.

На втором этапе производится настройка значений параметров функций принадлежности термов входных переменных, а также выражений в заключениях правил на основе выборки данных.

Рассмотрим подробнее первый этап.

Введём в рассмотрение выборку данных *Z***={***Z1, Z2…Zn***}**, объёмом n в *M*-мерном пространстве, где первые *L*-координат Z соответствуют вектору *X* входных переменных разрабатываемой модели, а оставшиеся *М-L* – вектору *Y* выходных переменных, т.е. *Zi* = (*Xi,, Yi*), *i = 1..n*.

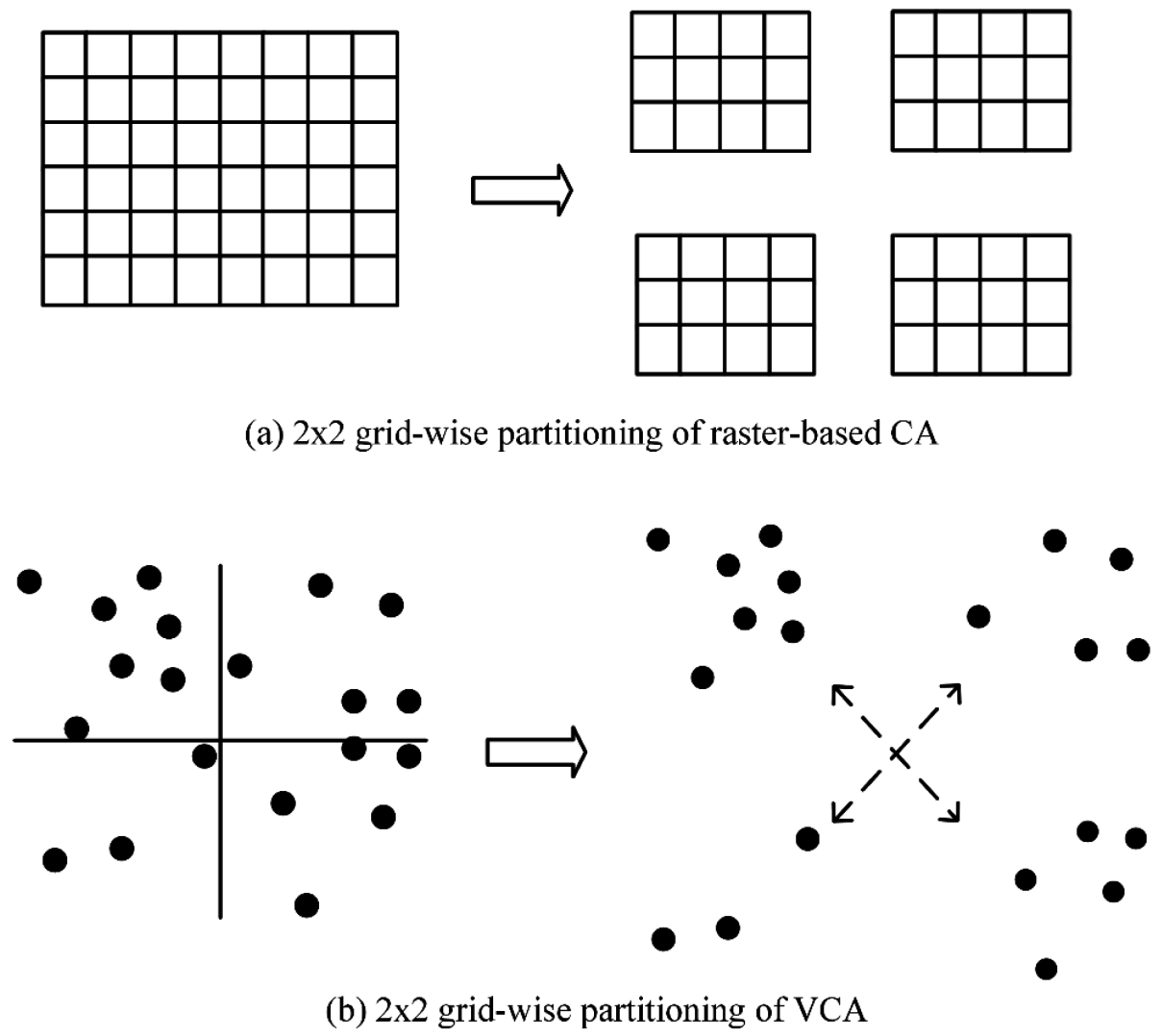
Рассмотрим следующие способы построения нейро-нечёткой системы

- Метод решётки

- Метод субтрактивной кластеризации

**Метод решётки**

Пространство выборки *Z* разделяется на заданное количество равных областей (см. рис. 1.2)



**Рис.1.2. Иллюстрация процедуры разделения данных по методу решётки**

Решающие правила определяются на основании проведённого разделения.

Функции принадлежности задаются из стандартного набора, максимум функций соответствует центрам выделенных областей

**Метод субтрактивной кластеризации**

Метод основан на разбиении имеющейся выборки данных *Z* на кластеры. Идея метода заключается в том, что каждый из кластеров, характеризующийся определёнными значениями, описывает определённое состояние исследуемой системы.

Решающие правила определяются на основании проведённого разделения. При этом максимум функции принадлежности соответствует центру кластера. Выделение кластеров производится на основе расчёта потенциалов точек.

Согласно этому методу, сначала для каждой *i*-той точки выборки *Z* рассчитывается значение потенциала *Pi , i = 1..n*:

,

где  является настраиваемым параметром, определяющим область, охватываемую каждым кластером. Константа ra – это радиус, определяющий область соседства. Таким образом, потенциал каждой точки данных  - это функция её расстояний до других точек данных.

Точка с большим количеством соседей будет иметь высокое значение потенциала. Точки данных вне радиуса ra оказывают небольшое влияние на значение потенциала. После расчёта потенциала каждой из точек выбираем точку с наиболее высоким потенциалом в качестве центра первого кластера. Пусть Z1\* – координаты центра первого кластера, а Р1\* - значение его потенциала. Обычно, точка с максимальным значением потенциала окружена несколькими точками-соседями также с достаточно высокими потенциалами. Поэтому назначение центром следующего кластера точки с максимальным потенциалом среди оставшихся привело бы к выделению большого числа близко расположенных центров кластеров. Чтобы выбрать следующий центр кластера необходимо вначале исключить влияние только что найденного кластера. Поэтому мы корректируем значение потенциала каждой точки по следующей формуле



где  а *rb* – положительная константа. Таким образом, мы вычитаем из потенциала каждой точки функцию расстояния от центра первого кластера. Таким образом, точки вблизи центра первого кластера будут иметь значительно пониженный потенциал и поэтому не будут выбраны в качестве центра следующего кластера.

Константа *rb* определяет область соседних точек, у которых значительно уменьшится потенциал. Для того чтобы избежать близко расположенных центров кластеров необходимо, чтобы *rb > ra*; хорошим выбором является *rb = 1.5ra* ,т.к. такое их соотношение позволяет минимизировать количество выделяемых кластеров при сохранении точности*.*

Когда потенциал всех точек был изменён, точка с наивысшим потенциалом выбирается в качестве центра второго кластера. В общем случае для центра *k*-того кластера формула будет следующей:



Где *Zk\** – положение центра *k*-того кластера, *Pk\** значение его потенциала. Процедура пересчета потенциалов и выделения центров кластеров продолжается до тех пор, пока максимальное значение потенциала превышает некоторый порог. Если на *k*-том шаге , то мы принимаем *Zk\** в качестве центра кластера и продолжаем дальше. Если  то отклоняем *Zk\** и прекращаем итерационный процесс. Если не выполняются оба вышеприведённых неравенства, то проверяем, выполняется ли неравенство:

****

где dmin – наикратчайшее расстояние между Zk\* и центром кластера, найденного на предыдущем шаге. Если выполняется, то принимаем *Zk\** в качестве центра нового кластера и продолжаем поиск. В противном случае отклоняем точку *Zk\** и устанавливаем её потенциал равным 0, выбираем точку со следующим по величине потенциалом и повторяем проверку.

Здесь  задаёт минимальное значение потенциала, при котором мы определённо считаем точку данных центром кластера;  определяет порог, ниже которого мы определённо отбрасываем точку. Общая методика по выбору порогов отсутствует, вместе с тем, в литературе на основе некоторых практических соображений и  приводятся рекомендуемые значения.Неравенство отражает проверку факта достижения компромисса между текущей величиной потенциала и расстоянием от уже определённых центров кластеров. В результате получим множество центров кластеров *Z\*=* {*Z1\*, Z2\*..Zc\**}.

Рассмотрим координаты центров выделенных кластеров. Разложим координаты каждого из центров *Zi\** на два составляющих вектора *Xi\** = (*x1(i)\*.. xL(i)\**) и *Yi\* =* (*y1(i)\*.. y(M-L)(i)\**)*, *, где *Xi\** состоит из первых *L* координат *Zi\** (т.е. из координат центров кластеров в пространстве входов), а *Yi\** – из *M – L* координат (т.е. из координат центров кластеров в пространстве выходов). В результате получили *c* соответствий, в которых для значений переменных на входе *X* = *Xi\** выходные переменные будут принимать значение *Y* = *Yi\**. Эти соответствия можно рассматривать в качестве решающих правил и использовать следующей форме:

«Если *x1* это *x1(i)\** и *x2* это *x2(i)\**…то *y1* это *y1(i)* и *y2* это *y2(i)\**...».

1.4.5. Определение начальных параметров функций принадлежности термов входных и выходных переменных на основе потенциалов выделенных кластеров

В предыдущем пункте показано получение термов и базы решающих правил из выборки данных. На следующем необходимо задать функции принадлежности для термов лингвистических переменных. Опираясь на существующие разработки в данной области, предлагается степень принадлежности к терму определять через введённое ранее понятие потенциала точки.

Если выражение потенциала записать для X из выборки Z, относительно *Хi\** из *Zi\**, то можно определить степень принадлежности *Х* к *i*-тому кластеру как

 e, *i=1..c*.

Указанное выражение определяет степень активации решающего правила, определяемого координатами центра кластера *Zi\**. Соответственно, рассчитать значение выходного вектора *Y* можнокак средневзвешенное значение выходов всех правил:



Для того чтобы получить выражения функций принадлежностей для термов каждой из входных переменных, т.е. для компонент вектора *X* = (*x1,.,xL*), перепишем в скалярной форме:



Полученное выражение соответствует функции принадлежности Гауссового типа для i-того терма, переменной *xj*. Здесь *xj(i)\** – *j-*ая координата *Хi\** (т.е. центра *i*-того кластера), *i=1..c, j = 1..L.* В результате на основе выборки данных с использованием понятия потенциала кластера произведён переход от набора кластеров к системе нечёткого вывода. При этом значение потенциала точки будет соответствовать степени активации решающего правила, расчёт потенциалов по каждой координате точки – фаззификации, а расчёт выходного значения как средневзвешенной суммы центров всех кластеров – дефаззификации, что соответствует нечёткой модели Сугено-Такаги.

1.4.6. Определение начальных параметров функций принадлежности термов выходной переменной

На следующем этапе необходимо определить функции принадлежности для выходной переменной. Алгоритмом Сугено-Такаги для лучшей точности нечёткого вывода предусмотрено задание выхода каждого из правил не в виде константы, а в виде линейной функции от входных переменных:



Здесь Gi – это матрица размерностью (М – L) х L, а hi – вектор-столбец с M – L элементами, i = 1..c. В результате получится модель нечёткого вывода типа Сугено-Такаги 1-ого порядка, где консеквентом каждого правила является линейная комбинация входных переменных. Такая модель позволяет описывать поведение сложных систем при небольшом объёме базы решающих правил. Настройка уравнений может осуществляться на основе имеющейся выборки с данными с помощью метода наименьших квадратов (МНК). Для этого введём следующий параметр:

.

Тогда можно получить следующее уравнение:



Приведя которое к матричному виду, получаем:



Учитывая, что при заданных из выборки значениях X и Y, первая матрица в левой части и матрица в правой части не содержат неизвестных, с помощью МНК получим неизвестные значения элементов матрицы G и вектор-столбца h, т.е. коэффициенты функций принадлежности выходной переменной

Неизвестные значения элементов матрицы *G* и вектор-столбца *h*, т.е. коэффициенты заключений правил находятся с помощью МНК

1.4.7. Настройка параметров нейро-нечёткой системы

На данном этапе производится настройка параметров модели с помощью методов обучения нейронных сетей. Настройка производится на основе метода обратного распространения ошибки. Для использования методов обучения нейронных сетей для настройки нечёткой модели типа Сугено-Такаги представим её в виде эквивалентной ей нейронной сети прямого распространения следующего вида, как показано на рис.

С учётом приведённого в п.1.4.2 описания структуры системы в процессе обучения будут настраиваться параметры нейронов в слое 1 и слое 4.

Настройка нейронов слоя 1, по сути, является настройкой функций принадлежности (параметров сдвига и масштаба) термов входных переменных, а слоя 4 – коэффициентов в выражениях заключений решающих правил.

Процесс обучения сети состоит из прямого и обратного проходов на каждой эпохе. Во время прямого прохода мы предъявляем выборку на вход сети и функциональные сигналы идут по ней в прямом направлении, чтобы рассчитать выход каждого узла сети. После этого определяется мера ошибки относительно данных из обучающей выборки и производная:

 ,

где *p* – номер набора данных из выборки T,  - значение выходной переменной из выборки,  - значение выходной переменной, рассчитанное в ходе прямого прохода по сети.



Во время обратного прохода по сети последовательно идёт расчёт производной меры ошибки на выходе каждого из внутренних слоёв,

,

где  - значение на выходе i-ого узла k-того слоя при предъявлении на вход сети p-ого набора данных из выборки, - количество узлов в *k+1* слое.

Если обозначить корректируемый параметр узла сети , тогда получим



где *S* – множество всех узлов, выходное значение которых зависит от . Тогда значение производной величины ошибки для всей эпохи будет следующим:

,

где *P* – объём обучающей выборки.

Соответственно, параметры 1-ого слоя (параметры функций принадлежности) и 4-ого слоя (параметры коэффициентов заключений нечетких правил) корректируются согласно методу градиента по следующей формуле:



где  – коэффициент шага коррекции, 1

Обучение продолжается до тех пор, пока ошибка обучения превышает допустимое значение:



где *p* – номер набора данных из выборки *T*,  - значение выходной переменной из выборки,  - значение выходной переменной, рассчитанное в ходе прямого прохода по сети.

## 2. ОПИСАНИЯ ЛАБОРАТОРНЫХ РАБОТ

# Лабораторная работа №1. Изучение основ применения искусственных нейронных сетей

Цель работы – изучение основ работы с искусственными нейронными сетями (ИНС) в среде Neural Toolbox ППП Simintech.

Одно из возможных применений ИНС, которое рассматривается в курсе – оценка надёжности технических систем.

Рассмотрим в качестве объекта исследования элемент автоматизированной информационно-измерительной системы коммерческого учёта электроэнергии (АИИС КУЭ) – информационно-вычислительный комплекс электроустановки (ИВКЭ).

Основу информационно-измерительного канала (ИИК) системы составляют измерительные трансформаторы тока (ТТ), измерительные трансформаторы напряжения (ТН), их вторичные цепи, трёхфазный счетчик электрической энергии. Таким образом, измерительные трансформаторы тока и напряжения, входящие в состав системы, предназначены для преобразования большого тока и высокого напряжения сети к уровням, соответствующим входным токам и напряжениям счётчиков электрической энергии, иными словами, они являются масштабирующими устройствами.

Следующим элементом системы, образующим вместе с ТТ и ТН ИИК, является микропроцессорный счётчик электрической энергии. Он предназначен для измерения перетоков активной и реактивной электрической энергии и мощности. Вдобавок, он способен выполнять интегрирование результатов измерений на получасовых интервалах и сохранение полученных значений в памяти с привязкой к текущему времени (профили нагрузки).

На уровне ИВКЭ происходит сбор и объединение результатов измерений по отдельным ИИК в групповые измерения по НПС и подготовка информации для передачи на более высокие уровни АИИС КУЭ. Обычно данные функции выполняет комплексное устройство учёта и автоматизации (КУУиА).

Учитывая нормативные документы, применяемые в области эксплуатации АИИС КУЭ, и тот факт, что как АИИС КУЭ, так и ИВКЭ являются восстанавливаемыми системами, то в качестве показателя, характеризующего её состояние, используется коэффициент готовности.

Исходя из анализа структуры ИВКЭ, для использования для оценивания надёжности в качестве входных переменных можно использовать следующие.

*Срок эксплуатации компонентов*, входящих в ИВКЭ.

Выбор данной переменой связан с тем, что на различных этапах жизненного цикла ИВКЭ показатели надёжности отдельных устройств (например, интенсивность отказов), входящих в него могут принимать не только различные значения, но и менять вид закона распределения (последний фактор часто не учитывается при расчётах обычными методами).

Сложность ИВКЭ предлагается измерять *количеством установленных счётчиков электрической энергии*.

Выбор данного показателя связан с тем, что, с точки зрения теории надёжности, все компоненты, входящие в ИВКЭ представляют собой последовательное соединение элементов. Т.е. надёжность такой системы зависит от количества, входящих в неё устройств и, в значительной степени, определяется самым слабым звеном в цепи элементов. Учитывая то, что счётчики электрической энергии являются одними из наиболее сложных с технической точки зрения и многочисленными устройствами, входящих в ИВКЭ, можно предположить, что они как раз и являются этим слабым звеном.

Максимальное по модулю отклонение от номинального режима величины тока в измерительных цепях компонентов ИВКЭ.

Как показывает практика, отклонение режима работы от номинального для любого устройства ведёт к ухудшению его показателей надёжности. Режим работы компонентов ИВКЭ может характеризоваться температурными условиями и влажностью в помещении, напряжением питания, напряжённостью электромагнитного поля и т.п. Однако, как показывают исследования, на практике чаще всего имеет место влияние отклонения величины тока в измерительных цепях ТТ, ТН и счётчиков электрической энергии. При значениях тока ниже номинального происходит рост суммарной погрешности измерений вплоть до выхода за допустимые пределы. А такая ситуация может расцениваться как отказ, т.к. система перестаёт правильно выполнять свои функции. При значениях тока, выше номинального к росту погрешности добавляется повышенная нагрузка на элементы, которая может привести к выходу устройств из строя.

По результатам эксплуатации подобного рода систем, как правило, накапливается значительный объём статистических данных, который можно использовать для построения ИНС, которая будет представлять собой оценку зависимости показателя надёжности от внешних факторов. В результате, с помощью такой ИНС можно оценить значение коэффициента готовности по значениям параметров сложности, срока эксплуатации и отклонения режима эксплуатации.

В лабораторной работе необходимо построить ИНС, позволяющей оценивать надёжность, на основе выборки с тремя параметрами, характеризующими информационно-измерительные системы коммерческого учёта электроэнергии (коэффициент сложности, среднюю загруженность оборудования, срок эксплуатации оборудования), и значением параметра надёжности (коэффициента готовности), соответствующего данным системам.

Порядок выполнения работы.

1. Ознакомьтесь с принципами работы с искусственными нейронными сетями в ППП Simintech на примере проекта классификации ирисов (\SimInTech\Demo\Нейронные сети\Классификация ирисов\Обучение.prt).
2. Изучите состав схемы, назначение блоков и их свойства. Изучите пункт меню «Параметры расчёта» Проведите обучение данной сети. Сделайте вывод о точности обучения.
3. Проведите исследование работы обученной сети, для этого откройте проект (\SimInTech\Demo\Нейронные сети\Классификация ирисов\Тестирование.prt).
4. Исследуйте влияние увеличения и уменьшения количества слоёв на качество обучения.
5. На основе изученных схем создайте свою схему для классификации систем по степени надёжности. Сохраните созданный проект в своей папке на ПК.
6. Данные, необходимые для выполнения работы, находятся в файлах *inputXX.txt, targetXX.txt*. Для загрузки данных из файлов в среду Simintech воспользуйтесь блоком «Таблица данных из файла».
7. При настройке размерности входных данных укажите, что в них содержится 3 параметра.
8. Проведите обучение и тестирование полученной сети, аналогично примеру с ирисами. Сделайте выводы о точности обучения и влиянии количества слоёв и нейронов в слое на качество обучения.

Вопросы к защите:

1. Искусственный нейрон. Структура и свойства.
2. Свойства искусственных нейронных сетей (ИНС). Классификация искусственных нейронных сетей.
3. Многослойная (двухслойная) сеть прямого распространения. Принципы работы.
4. Обучение нейронных сетей. Алгоритм обратного распространения ошибки
5. Алгоритм обратного распространения ошибки. Геометрическая интерпретация.
6. Переобучение. Причины возникновения. Способы избегания переобучения.
7. Алгоритмы обучения без учителя. Особенности.
8. ИНС в задачах управления. Приведите пример.
9. ИНС в задачах диагностики. Приведите пример.
10. Опишите процесс работы и настройки нейрорегулятора.

# Лабораторная работа №2. Изучение основ применения генетических алгоритмов

Цель работы – изучение основ работы с генетическими алгоритмами в среде Optimization Toolbox ППП Matlab.

В работе рассматривается решение задачи поиска минимума функции на заданном интервале с применением генетических алгоритмов.

Порядок выполнения работы.

1. Запустите пакет Optimization Toolbox с помощью команды *gatool* или *optimtool(‘ga’)* (в зависимости от версии пакета MatLab).
2. Изучите состав главного окна модуля, проанализируйте назначение и параметры отдельных блоков.
3. Создайте функцию в пакете MatLab в соответствии с вариантом (см. табл. 2.2). Постройте её график (с помощью функции *plot*).

Таблица 2.2

|  |  |
| --- | --- |
| № вар. | Функция |
|  | f(x) = 2x2+1 |
|  | f(x) = 2x3+125 |
|  | f(x) = 4x4+16 |
|  | f(x)= |
|  | f(x)= |
|  | f(x) = 2x8+15 |
|  | f(x) = |
|  | f(x) = -4x4+16 |
|  | f(x)= |
|  | f(x) = 2|x7|+11 |

1. Задайте параметры работы алгоритма, соответствующие классическому варианту.
2. Запустите процедуру поиска оптимального значения на отрезке [0;10]. Проанализируйте полученные результаты, оцените точность полученного результата*.*
3. Модифицируйте алгоритм поиска: измените количество точек скрещивания, тип селекции (по указанию преподавателя). Проанализируйте полученные результаты.
4. Полученные результаты и выводы занесите в отчёт.

Вопросы к защите:

1. Основные понятия генетических алгоритмов.
2. Генетические алгоритмы и традиционные методы оптимизации. Достоинства и недостатки
3. Генетические операторы. Кодирование параметров задачи в генетическом алгоритме.
4. Функция приспособленности.
5. Опишите работу классического генетического алгоритма.
6. Основная теорема о генетических алгоритмах. Следствия теоремы.
7. Модификации классического генетического алгоритма
8. Генетические алгоритмы и ИНС. Способы совместного использования

# Лабораторная работа № 3. Изучение основ применения нечёткой логики

Цель работы – изучение основ работы с элементами нечёткой логики в среде Fuzzy Logic Toolbox ППП Matlab.

В работе рассматривается система управления с нечётким регулятором. Объектом управления является бак с водой, к которому подходят две трубы: через одну трубу, снабженную краном, вода поступает в бак, через другую – вытекает. Подачу воды в бак можно регулировать краном. Расход воды является неконтролируемым и зависит от уровня воды в баке. Если понимать под выходной (регулируемой) переменной уровень воды, а под регулирующим элементом – кран, то подобный объект регулирования с точки зрения его математического описания является динамическим и существенно нелинейным.

Цель управления в данном случае заключается в установлении на требуемом, изменяющемся во времени уровне воды в баке. В регулятор, обеспечивающий достижение цели управления, должна поступать информация о несоответствии (разности) требуемого и фактического уровней воды, при этом данный регулятор должен вырабатывать управляющий сигнал, подаваемый на регулирующий элемент.

Порядок выполнения работы.

1. Откройте в Simulink модель системы с регулятором воды в баке командой *sltank*. Изучите её состав, проанализируйте назначение и параметры блоков.

2. Запустите выполнение модели. Проанализируйте графики, выводимые в блоках *scope.* Занесите их в отчёт.

3. Запустите выполнение с ПИД-регулятором в контуре вместо нечёткого. Проанализируйте графики, выводимые в блоках *scope,* и сравните их с полученными в п.2*.* Занесите их в отчёт.

Примечание: для использования ПИД-регулятора в контуре измените управляющий вход в блоке *Switch*.

4. Перейдите в командное окно MatLab и откройте в редакторе Fuzzy Logic Toolbox модель нечёткого регулятора воды в баке командой *fuzzy(tank).* Изучите переменные регулятора, термы и решающие правила

5. Изучите в редакторе состав и параметры нечёткого регулятора, а также используемый тип алгоритма нечёткого вывода. Обратите внимание на возможность настройки способов реализации логических операций «И», «ИЛИ», логического вывода, объединения полученных заключений правил, дефаззификации.

6. Перейдите в редактор функций принадлежности (Membership Function Editor), кликнув по одной из входных или выходной переменных. Обратите внимание на способ описания параметров функций (поле *Params*), а также типа функции (поле *Type*).

Изучите способы создания и удаления термов и функций принадлежности (ФП) через пункты *Edit* → *Add MFs/Remove Selected MF* (Правка → Добавить ФП/Удалить выбранную ФП).

7. Перейдите в редактор решающих правил через пункт Edit → Rules… (Правка → Правила). Проанализируйте представленный набор правил. Обратите внимание на возможные предпосылки правил (способы задания связок между ними), а также заключения. Изучите процесс создания, редактирования, удаления правил.

8. Перейдите в окно просмотра правил (*Rule Viewer*) из редактора правил через пункт View → Rules (Вид → Правила). Задавая различные значения входных параметров, изучите процесс формирования выходной величины. С помощью пункта View → Surface (Вид → Поверхность) постройте поверхность отклика регулятора. Проанализируйте её и занесите в отчёт.

9. Модифицируйте исходную систему в соответствии с вариантами, приведёнными ниже.

Вариант 1: Модифицируйте базу решающих правил. Добавьте учёт скорости изменения уровня воды при высоком и низком уровнях воды в баке.

Вариант 2: Модифицируйте базу решающих правил. Удалите два наименее важных (обоснуйте) решающих правила.

Вариант 3: Измените вид функций принадлежности и границы термов входных переменных.

Вариант 4: Измените способ дефаззификации на метод среднего максимума.

Сохраните получившийся нечёткий регулятор в пункте *File* → *Export to Workspace* (Файл → Экспорт в рабочую область) под именем *tank2.*

7. В модели системы с регулятором воды, в параметрах нечёткого регулятора укажите изменённую модель нечёткого регулятора *tank2*. Запустите моделирование системы. Графики параметров работы системы занесите в отчёт. Проанализируйте работу системы. Что изменилось в работе системы? Выводы занесите в отчёт.

Вопросы к защите:

1. Какие виды функций принадлежности Вам известны? Проанализируйте их преимущества и недостатки.
2. Опишите основные этапы нечёткого вывода.
3. Алгоритм Мамдани. Особенности.
4. Алгоритм Цукамото. Особенности.
5. Алгоритм Ларсена. Особенности.
6. Алгоритм Такаги-Сугено. Особенности.
7. Опишите процедуру фаззификации.
8. Опишите процедуру дефаззификации. Способы дефаззификации.
9. Опишите работу нечёткого регулятора. Приведите структуру нечёткой системы управления.
10. Опишите применение нечётких систем для решения задач диагностики.

# Лабораторная работа №4. Изучение основ применения нейро-нечётких систем

Цель работы – изучение основ работы с нейро-нечёткими системами в среде Neural Toolbox ППП Matlab.

Имеется выборка с тремя параметрами (коэффициент сложности, среднюю загруженность оборудования, срок эксплуатации оборудования), характеризующими информационно-измерительные системы коммерческого учёта электроэнергии (аналогичной рассмотренной в лабораторной работе №1) и значением параметра надёжности (коэффициента готовности), соответствующего данным системам.

Необходимо с помощью представленных выборок создать нейро-нечёткую систему, позволяющую производить оценивание надёжности подобного рода систем.

Порядок выполнения работы.

1. Данные, необходимые для выполнения работы, находятся в файле LR4\_data.mat. Необходимо открыть данный файл в среде MatLab.
2. Запустите в среде Matlab командой *fuzzy* Fuzzy Logic Toolbox. Создайте систему нечёткого вывода Сугено. Задайте с помощью редактора нечётких систем (*Fuzzy Logic Designer*) количество входных переменных у созданной системы равным 3 (что соответствует размерности входных данных), а затем перейдите в пункт *ANFIS* (Редактор нейро-нечёткой системы). Изучите редактор*.*
3. Исследуйте способ создания нейро-нечёткой системы с помощью метода решётки (*grid partition*).
   1. Загрузите данные для дальнейшего построения нейро-нечёткой системы, выбрав варианты загрузки обучающих (*training*) данных из рабочей области (*from workspace*). В качестве данных для обучения используйте массив *teach\_arX1,* где Х – номер варианта. Обратите внимание на визуализацию данных для обучения.
   2. С помощью кнопки *Generate FIS* (Создать систему нечёткого логического вывода) запустите построение системы. В открывшемся окне задайте функции принадлежности Гауссова типа (*gaussmf*) для входных переменных. Число функций принадлежности по каждой переменной задайте равным 6. Выходные функции принадлежности задайте линейного вида.
   3. Занесите в отчёт получившиеся функции принадлежности.
   4. Проведите обучение получившейся нейро-нечёткой системы. Обратите внимание на предлагаемые способы обучения: *backprop* (метод обратного распространения ошибки) и *hybrid* (гибридный метод обратного распространения ошибки и метода наименьших квадратов), а также критерии останова обучения: *Error Tolerance* (Допустимая ошибка) и *Epochs* (Количество эпох). Количество эпох задайте равным 300, значение допустимой ошибки – 0. В дополнение к обучающей выборке загрузите проверочную выборку (*checking data*) *teach\_arX2,* где Х – номер варианта*.* Запустите процесс обучения при помощи кнопки *Train Now* (Обучить сейчас).
   5. Занесите в отчёт и проанализируйте график изменения ошибки в процессе обучения, длительность обучения, а также величину ошибки на момент окончания обучения.
   6. Сравните функции принадлежности до и после обучения.
   7. Загрузите тестовую выборку (*testing data*) *teach\_arX3,* где Х – номер варианта*.* Проведите с её помощью тестирование полученной нейро-нечёткой системы. График ошибки проанализируйте и занесите в отчёт.
   8. Сохраните получившуюся нейро-нечёткую систему в качестве переменной Matlab (*Export to workspace*).
4. Исследуйте способ создания нейро-нечёткой системы с помощью с помощью субтрактивной кластеризации (subtractive clustering).
   1. В качестве параметров кластеризации используйте значения, заданные по умолчанию (табл. 2.3).

Таблица 2.3.

|  |  |  |
| --- | --- | --- |
| Наименование параметра | Значение | Назначение |
| Range of influence  (Диапазон влияния) | 0,5 | Определяет область, охватываемую каждым кластером |
| Squash factor  (Фактор подавления) | 1,25 | Определяет область, которая не должна относиться к кластеру |
| Accept ratio  (Величина принятия) | 0,5 | Уровень принятия точки в качестве центра нового кластера |
| Reject ratio  (Величина отклонения) | 0,15 | Уровень непринятия точки в качестве центра нового кластера |

* 1. Занесите в отчёт получившиеся функции принадлежности.
  2. Проведите обучение получившейся нейро-нечёткой системы. Количество эпох задайте равным 300, значение допустимой ошибки – 0. В дополнение к обучающей выборке загрузите проверочную выборку (*checking data*) *teach\_arX2,* где Х – номер варианта.
  3. Занесите в отчёт и проанализируйте график изменения ошибки в процессе обучения, длительность обучения, а также величину ошибки на момент окончания обучения.
  4. Сравните функции принадлежности до и после обучения
  5. Сохраните получившуюся нейро-нечёткую систему в качестве переменной Matlab (*Export to workspace*).
  6. Загрузите тестовую выборку (*testing data*) *teach\_arX3,* где Х – номер варианта*.* Проведите с её помощью тестирование полученной нейро-нечёткой системы. График ошибки проанализируйте и занесите в отчёт.

1. Сравните две системы, полученные в п.3. и п.4.
2. Повторите создание системы с помощью метода кластеризации с новыми параметрами (табл. 2.4)

Таблица 2.4

**Параметры для метода кластеризации**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № варианта | Диапазон влияния | Фактор подавления | Величина принятия | Величина отклонения |
| 1 | 0,7 | 1,5 | 0,6 | 0,2 |
| 2 | 0,4 | 1 | 0,35 | 0,1 |
| 3 | 0,8 | 1 | 0,55 | 0,2 |
| 4 | 0,3 | 1,5 | 0,2 | 0,1 |
| 5 | 0,55 | 1,2 | 0,4 | 0,2 |
| 6 | 0,35 | 1,2 | 0,2 | 0,1 |
| 7 | 0,6 | 1,4 | 0,4 | 0,2 |
| 8 | 0,45 | 1,4 | 0,3 | 0,1 |
| 9 | 0,75 | 1 | 0,6 | 0,2 |

1. Проведите сравнение получившихся в п.4 и п.6 систем. Как повлияло изменение параметров на количество решающих правил, форму функций принадлежности, точность работы системы. Результаты занесите в отчёт.

Вопросы к защите:

1. Нейро-нечёткие системы. Предпосылки возникновения
2. Нейро-нечёткие системы. Структура системы.
3. Нейро-нечёткие системы. Принципы работы.
4. Нейро-нечёткие системы. Сравнение с нечёткими и нейросистемами.
5. Создание функций принадлежности на основе выборки данных.
6. Использование методов ИНС для обучения нейро-нечётких систем.
7. Модификации нейро-нечётких систем. Особенности.
8. Алгоритмы обучения без учителя. Особенности.
9. Нейро-нечёткие системы в задачах управления. Приведите пример.
10. Нейро-нечёткие системы в задачах диагностики. Приведите пример.

**СПИСОК РЕКОМЕНДОВАННОЙ ЛИТЕРАТУРЫ**

**Основной**

1. Борисов В.В., Круглов В.В., Федулов А.С. Нечеткие модели и сети. – 2-е изд., стереотип. – М.: Горячая линия–Телеком, 2016. – 284 с.

**Дополнительный**

1. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы: Пер. с польск. И.Д. Рудинского. – 2-е изд., стереотип. – М.: Горячая Линия–Телеком, 2013. –452 с.:ил.
2. Пегат А. Нечеткое моделирование и управление: пер. с англ. – 2-е изд.–М.: БИНОМ. Лаборатория знаний, 2017.–798 с.: ил.