УДК 007

АНАЛИЗ АЛГОРИТМОВ ДЛЯ СИСТЕМЫ ПРИНЯТИЯ РЕШЕНИЙ

Бошляков А.А.
Доцент,
Московский Государственный Технический Университет им. Н.Э. Баумана,
Москва, Россия

Поярков Г.А.
Студент магистратуры,
Московский Государственный Технический Университет им. Н.Э. Баумана,
Москва, Россия

Аннотация
В данной статье раскрывается проблема управления схватом, имеющим большое количество доступных захватов и рассматриваются различные алгоритмы систем принятия решения, основанные на нечеткой логике, нейронных сетях, нейронечетких методах и классических методах машинного обучения, которые возможны для использования в управлении роботизированным протезом. В ходе работы оцениваются вычислительная сложность, необходимая оперативная память и другие качества методов.

Ключевые слова: Система принятия решений, нейросети, нечеткая логика, алгоритмы, ансамбль алгоритмов, нейронечеткие системы.

ALGORITHM ANALYSIS FOR DECISION MAKING SYSTEM

Boshlyakov A.A.
Docent,
Bauman Moscow State Technical University,
Russia, Moscow

Poyarkov G.A.
Master student
Bauman Moscow State Technical University
Russia, Moscow

Annotation
This article reveals the problem of controlling a grip that has a large number of available grips and discusses various algorithms for decision-making systems based on fuzzy logic, neural networks, neuro-fuzzy methods and classical machine learning methods that are possible for use in controlling of robotic prosthesis. In the course of the work, computational complexity, the necessary RAM, and other qualities of the methods are evaluated.

Keywords: decision-making system, neural networks, fuzzy logic, algorithms, ensemble of algorithms, neural-fuzzy systems.

Введение

В 21 веке производство электромеханических протезов кистей рук перестало быть чем-то фантастическим. Современные технологии позволяют реализовать множество различных конструкций, даже полностью повторяющие функционал человеческой конечности.

Однако существует проблема, решение которой является ключевым фактором, влияющим на качество использования протеза – управление устройством пользователем [1, с.22].

В коммерческих протезах, где для управления используются неинвазивные методы считывания ЭМГ-сигналов с помощью биопотенциальных датчиков, применяются ограниченное количество захватов, заданных заранее и поделенных на группы (Рис. 1). В таком случае задачу выбора движения, которое выполнит рука, можно поделить на выбор группы и выбор захвата внутри группы, что позволяет уменьшить количество используемых датчиков и упростить управление протезом.

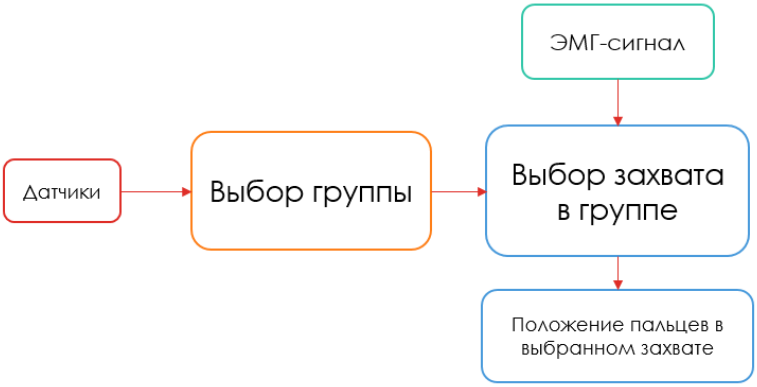

Рис. 1 - Выбор захвата с помощью деления на группы.

На данный момент наиболее распространёнными решениями (Рис. 2-5) этой задачи являются:

1. выбор группы захватов положением большого пальца;
2. выбор группы захватов при помощи чипа;
3. выбор группы захватов комбинацией команд;
4. Выбор группы захватов в мобильном приложении.

Рис.2 - Захват положением большого пальца.

Рис. 3 - Использование чипа.

Рис. 4 - Использование комбинации команд. Рис. 5 Использование приложения.

Чтобы уменьшить количество действий, требуемых от пользователя, или даже свести их практически до нуля, предлагается другая структура (Рис. 6). В нее входят человеко-машинный интерфейс, с которого идут управляющие сигналы от пользователя и осуществляется выбор внутри группы, набор датчиков, которые дают устройству представление о захватываемом предмете и система управления приводами, распределяющая нагрузку и контролирующая мягкость захвата.

Рис. 6 - Структурная схема системы принятия решений.

Чтобы связать все эти данные и сигналы, требуется система принятия решений, которая бы осуществляла выбор группы захватов или, при определенной команде и достаточном объеме информации об объекте, выбирала конкретный захват. Таким образом достигается увеличение количества распознаваемых захватов с небольшим

количеством ЭМГ-датчиков, что упрощает управление по сравнению с рассмотренными выше аналогами.

В протезах, как и во многих робототехнических устройствах, используются процессоры с тактовой частотой на несколько порядков ниже, чем в персональных компьютерах. Поэтому, при реализации сложных встроенных систем, необходимо использовать те методы и алгоритмы, которые укладываются в жесткие ограничения, налагаемые микроконтроллерами.

Целью исследования является выбор наиболее эффективного метода реализации системы принятия решения, работающей в реальном времени во встроенной системе. Для этого понадобятся следующие критерии, оценки:

- Вычислительная сложность алгоритма - это функция зависимости объёма работы, которая выполняется некоторым алгоритмом, от размера входных данных. Для вычисления сложности программа разбивается на элементарные операции или алгоритмы с уже известной сложностью, после чего считается верхняя граница сложности путем отбрасывания всех функций имеющих меньший темп роста.
- Требования к ОЗУ – требования к объему памяти устройства, в которой во время работы хранится выполняемый код программы, а также входные, выходные и промежуточные данные, обрабатываемые процессором.
- Размер обучающей выборки – применительно к алгоритмам машинного обучения и нейросетям, это количество имеющихся примеров для обучения в виде пары: вектор входных параметров – ответ.

В статье будут рассмотрены такие методы как нечеткая логика, нейронные сети, ансамблевые методы машинного обучения и нейронечеткие системы.

1. Система принятия решений с нечеткой логикой.

Нечеткая логика была разработана профессором Лофтисаде из Калифорнийского университета в Беркли в 1965 году. Первое приложение было использовано для выполнения компьютерной обработки данных на основе натуральных значений.

Нечеткая логика - логическая система или система управления n-значной логической системой, которая использует «степени истинности» входов и выдает выходные значения, зависящие от состояний входов и скорости изменения этих состояний. Этот принцип в основном и обеспечивает приблизительное рассуждение с использованием неопределенных и неточных решений, принимающих на вход лингвистические **переменные.**

Например, в **логическом** языке мы можем сказать стакан горячей воды (логическая единица) или стакан холодной воды т. е. (логический ноль), **но в нечеткой логике** мы можем сказать стакан теплой воды (ни горячей, ни холодной).

Система нечеткой логики (Рис. 7) состоит из следующих модулей:

Фаззификатор (**Fuzzifier)** - принимает измеренные переменные в качестве входных данных и преобразует числовые значения в лингвистические переменные. Он отображает физические значения, а также сигналы ошибок в нормированное нечеткое подмножество, которое состоит из интервала для диапазона входных значений и функций принадлежности, описывающих вероятность состояния входных переменных (Рис.8).

Рис. 7 - Система принятия решений с использование нечеткой логики

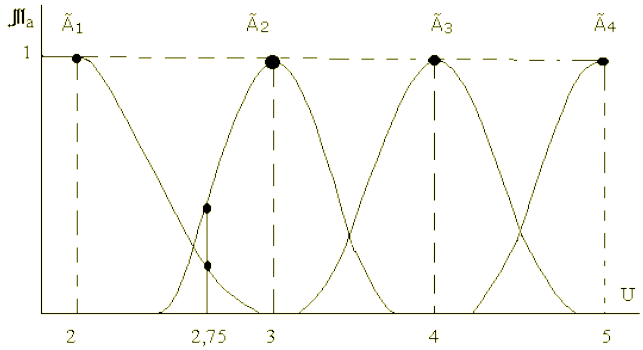

Рис. 8 Функции принадлежности фаззификатора.

Контроллер (Controller): он состоит из базы знаний (knowledge base), а также устройства вывода (inference engine). В базе хранятся функции принадлежности и нечеткие правила, полученные на основе экспертных знаний о работе системы в данной среде. Механизм вывода выполняет обработку полученных функций принадлежности с помощью нечетких правил. Другими словами, механизм вывода назначает выходные данные на основе лингвистической информации.

Дефаззификатор(Defuzzifier): выполняет обратный процесс фаззификации. То есть, он преобразует нечеткие значения в обычные числовые сигналы и отправляет их в физическую систему. Наиболее популярным методом дефаззификации является метод центра тяжести, который для дискретного случая рассчитывается как:

$$z_{cog}(S) = \frac{\sum_{i=1}^{N} \mu_s(z_i) * z_i}{\sum_{i=1}^{N} \mu_s(z_i)} \qquad (1)$$

Где S – нечеткое множество вывода, $\mu_s(z_i)$ – функция принадлежности, z – значение на нечетком множестве, а N – количество разбиений при дискретизации.

Системы, использующие нечеткую логику, применяются в робототехнике для анализа сигналов и систем управления роботами в условиях неопределенности и, в частности, при управлении протезами (например, для решения проблемы управления двумя степенями свободы большого пальца и нахождения оптимального положения в зависимости от пожеланий пользователя и сигналов с силомоментных датчиков [2, с. 22]).

Главными преимуществами систем с нечеткой логикой является их гибкость (возможность модификации в правилах) и отсутствие потребности в обучающей выборке.

Недостатками же являются отсутствие стандартной методики проектирования и расчета нечетких систем и невозможность математического анализа нечетких систем существующими методами.

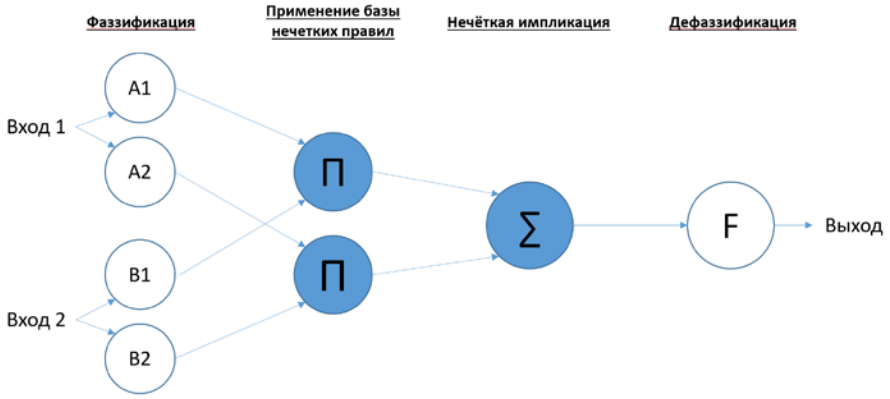

Рис. 9 - Алгоритм работы системы на основе нечеткой логики.

Если взять количество входов за *n*, среднее количество термов взять равным *a*, а количество разбиений за *N*, то алгоритм работы системы с нечеткой логикой можно представить следующим образом (Таблица 1):

Таблица 1 – Анализ сложности реализации нечеткой логики

№	Описание	Вычислительная сложность
1	Фаззификация. На этом этапе входному числовому значению в соответствие ставится лингвистическая переменная. При этом рассчитывается вероятность принадлежности значения к каждой лингвистической переменной, заданной в виде распределений. Соответственно, на первом шаге вычислительная сложность будет равняться количеству входных значений, умноженному на среднее количество термов.	$n * a$
2	Применение базы нечетких правил. В худшем случае количество этих правил будет равно среднему количеству термов в степени количества входов.	a^n
3	Следующим этапов является нечеткая импликация, в которой выводы каждого правила суммируются с помощью нечеткого объединения. Количество операций равно числу правил минус один.	$n - 1$
4	Дефаззификация. Переход от нечеткого логического вывода к искомому результату. Исходя из формулы 1, количество операций будет равно N*2.	$N + 2N$

Итого:

$$F(n) = n * a + a^n + n - 1 + N * 2 = z^n + (a + 1) * n + 3N - 1$$

Вычислительная сложность в худшем случае равна O(n) = a^n.

Для расчета необходимой памяти надо учесть какие именно параметры нам необходимо хранить постоянно и какие переменные будут использоваться при выполнении алгоритма (Таблица 2). Выводы делаются исходя из веса типов float в языке C++ (double – 8 Байт, int и float - 4 Байта, char и bool – 1 Байт).

Таблица 2 – Приблизительная оценка требований к памяти для нечеткой логики

№	Описание	Необходимое количество памяти
1	Нечеткие термы. Удобнее всего хранить такую информацию в виде формул, задающих распределение. Для нормального распределения, имеющего всего два параметра (математическое ожидание и среднеквадратичное отклонение), необходимое количество памяти равно 2*4 Байт, умноженное на количество входов и среднее количество термов.	$2 * n * a$
2	Применение базы нечетких правил. В худшем случае количество этих правил будет равно среднему количеству термов в степени количества входов умноженному на количество символов операций в правилах. Для хранения символов используется тип char весом один байт.	$a^n * n$
3	Дефаззификация. Требуется хранить $N + 2N$ чисел типа double.	$(N + 2N) * 8$

Итого для хранения данных требуется $a^n * n + 2 * n * a + (N + 2N) * 8$.

2. Система принятия решений на основе нейросети

Другим популярным методом, использующимся для систем принятия решений, является использование классификаторов на основе нейронных сетей. Нейросети используют обучение с учителем для настройки коэффициентов под конкретную задачу, то есть для создание такой системы принятия решений потребуется большая обучающая выборка. Являются универсальным алгоритмом, с помощью которого может быть описан любой другой алгоритм.

Нейросетевой классификатор (Рис. 10) состоит из следующих модулей (рассматривается на примере трехслойной нейросети):

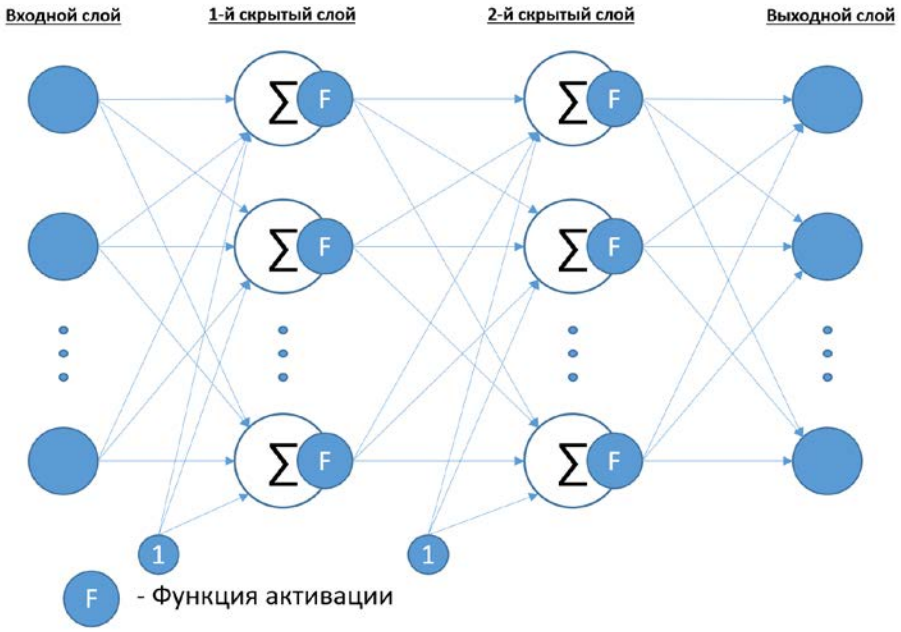

Рис. 10 - Классификатор на основе нейросети.

Входной слой – все элементы данного слоя являются входами для первого скрытого слоя. Они не имеют функций активации, а их количество равно количеству признаков, которые подаются на вход.

Скрытый слой – каждый нейрон скрытого слоя принимает значение, передаваемое ему входным слоем, помноженное на соответствующий весовой коэффициент. Именно эти коэффициенты настраиваются в процессе обучения нейросети. Сумма всех таких сигналов подается на вход функции активации нейрона, после чего он передает полученное значение следующему слою. Для всего слоя это действие можно описать с помощью следующей формулы:

$$Z = X*W+b \qquad (2)$$

Где X – вектор входных значений, W – матрица коэффициентов, а b – вектор смещения.

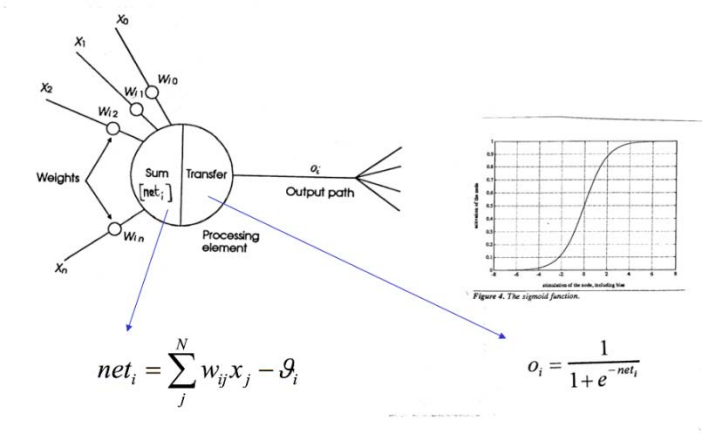

Рис. 11 - Структура нейрона с сигмоидальной функцией активации.

Функции активации - решает, должен ли нейрон (Рис. 11) быть активирован или нет, путем вычисления взвешенной суммы и дальнейшего добавления смещения с ней. Цель функции активации состоит в том, чтобы ввести нелинейность в выход нейрона. Наиболее используемыми из них являются:

- Ступенчатая функция (Рис.13)
- Сигмоидальная (Рис.11)
- ReLU(Рис.12)

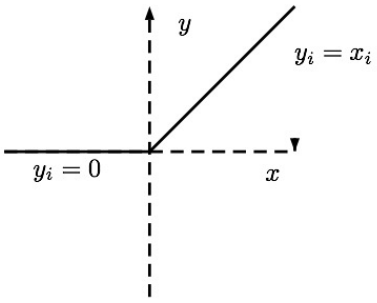

Рис. 12 - Функция ReLU.

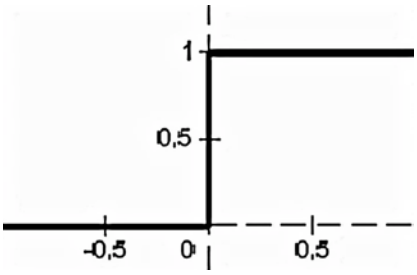

Рис. 13 - Ступенчатая функция.

Выходной слой - слой выводит информацию, полученную сетью, во внешний мир. Количество нейронов соответствует количеству классов. В случае применения сигмоидальной функции на выходном слое, на выходе получается принадлежность объекта к классу. Далее полученные значения сравниваются и ответом будет метка класса с наибольшей вероятностью.

Удачным применением нейросетей, в задачах схожих по тематике с рассматриваемой, является адаптивная «Стратегия классификации для надежного распознавания режима локомоции» [3, с. 22] и «Сочетание биофизического моделирования и глубокого обучения для локализации и классификации нейронов в многоэлектродной матрице» [4, с. 23].

Нейросети имеют очень высокое быстродействие, точность и отказоустойчивость, а наиболее отличительным преимуществом их использования является решение задач при неизвестных закономерностях

Однако нейросети могут слишком сильно подстроиться под обучающую выборку и несмотря на отличные показатели качества, при использовании такая модель будет бесполезна.

Работа нейросети во многом зависит от гиперпараметров. Для однонаправленного многослойного персептрона таковыми будут количество слоев (m) и количество нейронов в каждом слое(n1..nx), а также, в зависимости от задачи будут изменяться функции активации. Предположим, что количество нейронов, для слоя, будет уменьшаться по экспоненте, количество слоев будет равна 4-м (два скрытых слоя), а количество выходов будет равна b, то сложность алгоритма можно будет записать в следующем виде

Алгоритм классификации будет иметь следующий вид (Таблица 3):

Таблица 3 – Анализ сложности реализации нейронной сети

№	Описание	Вычислительная сложность
1	Запись значений во входной вектор признаков (количество операций равно числу признаков).	n

2	Умножение вектора 1xn на матрицу весов nxm (Для вычисления этого шага, можно использовать готовые алгоритмы перемножения матриц, как например.	n^2
3	Прибавление вектора смещения (количество операций равно числу нейронов второго слоя).	n
4	Умножение на функцию активации (При использовании сигмоидальной функции, количество операций можно приравнять к количеству нейронов второго слоя).	n
5	Повторение шагов 2-4 для каждого последующего слоя (Цикл, количество итераций которого равно количеству слоев минус один). В последнем слое вектор смещений не прибавляется.	-
6	Вывод значений.	b

Итого:

$$F(n) = n + (n^2 + 2 * n) + (n^2 + 2 * n) * e^{-1} + n * b + b =$$
$$= (n^2 + 2 * n) * (1 + e^{-1}) + n * (b + 1) + b$$

Таким образом, в худшем случае для нейросети с четырьмя слоями и с количеством нейронов, уменьшающихся экспоненциально для каждого последующего слоя, сложность алгоритма будет составлять O(n) = n^2.

Рассчитаем необходимое количество памяти (Таблица 4):

Таблица 4 - Приблизительная оценка требований к памяти для нейронной сети

№	Описание	Необходимое количество памяти
1	Наибольшее количество памяти необходимо для хранения матриц весов, размеры которых равны n^2e^{-1}. Также ранее было наложено	$(n^2e^{-1} + n^2e^{-3} + n^2e^{-5})$*8

	условие что с каждым слоем количество нейронов уменьшается по экспоненте. Для весов предпочтительно использовать тип double.	
2	Для хранения значений в вычисляемом слое необходим массив чисел с плавающей точкой с двойной точностью.	$n * 8$

Как итог, получаем $n^2(e^{-1} + e^{-3} + e^{-5}) * 8 + n * 8 = 3.4n^2 + 8n$

То есть необходимая память имеет квадратичную зависимость от количества данных на входе нейросети.

3. Классические подходы машинного обучения

В большинстве задач, где человек сам способен описать признаки при решении задачи обучения с учителем, нейронные сети являются неоправданно сложным решением. В таких случаях применяются классические методы машинного обучения. В задачах классификации большинство линейных алгоритмов поддерживает только разделение двух классов и в n-мерном пространстве признаков классификатор представляет собой разделяющую плоскость.

Чтобы разделить большее количество объектов, используется метод один против всех (one versus all), в котором создается m-алгоритмов (по одному на класс) и ответом будет та целевая метка, которая набрала наибольшее количество «очков» при сравнении с остальным набором.

Для увеличения точности и полноты алгоритма, можно использовать ансамблевый метод – градиентный бустинг.

Эффективный способ получить высокую оценку качества классификатора это построить алгоритм, на основе группы более простых алгоритмов, ответ которых будет суммироваться.

Линейный алгоритм представляем собой скалярное произведение вектора признаков на вектор коэффициентов:

$$a(x) = x \times \alpha \qquad (3)$$

В результате получается проекция точки на нормаль плоскости, задаваемой вектором коэффициентов.

Типичный функционал ошибки в регрессии — это среднеквадратичная ошибка:

$$MSE(a, X) = \frac{1}{l}\sum_{i=1}^{l}(a * (x_i) - y_i)^2 \qquad (4)$$

При этом функция потерь, которая измеряет ошибку для одного объекта представляет собой первую производную квадрата ошибки:

$$L(y, z) = (z - y)^2 \qquad (5)$$

$$L'_z(y, z) = 2(z - y) \qquad (6)$$

Алгоритмы обучаются последовательно, причем каждый последующий обучается на ошибках предыдущего, т.е. для следующего алгоритма вектор ответов получается как разность вектора ответов из обучающей выборки и вектора значений предсказанных на предыдущей итерации.

Бустинг помогает уменьшить ошибку простых алгоритмов, таких как линейные классификаторы. Успешным примером использования такой модели является предварительные результаты к естественно контролируемой мульти-синергетической протезной руке [5, с. 23].

Преимуществами этого метода является:

1. Достигается большая точность, так как часть алгоритмов учится на ошибках и как следствие, алгоритм хорошо подстраивается под данные.
2. Базовые алгоритмы обычно имеют малую вычислительную сложность, что снижает требования к используемым аппаратным средствам.

К недостаткам можно отнести быстрое переобучение такой системы. При большом числе последовательных функций алгоритм идеально будет

классифицировать все объекты обучающей выборки, однако на тестовой выборке его точность будет сильно падать.

Пусть количество признаков будет равно n, количество базовых алгоритмов будет равно 4, а количество классов равно b. Так как алгоритм представляет собой несколько алгоритмов, сумма ответов которых и является выходом, отличающим один класс от других, то вычисления сводятся к следующему алгоритму (Таблица 5):
Таблица 5– Анализ сложности реализации ансамбля линейных алгоритмов

№	Описание	Вычислительная сложность
1	Запись значений во входной вектор признаков (количество операций равно числу признаков);	n
2	Умножение вектора признаков на вектор весов (количество операций равно числу признаков, умноженному на число базовых алгоритмов и на количество классов)	b*4*n
3	Сравнение оценок определения различных классов (количество операций равно количеству классов)	b

Сложность алгоритма будет считаться следующим образом:

$$F(n) = n + b * 4 * n + b = (4 * b + 1) * n + b$$

Следовательно, для ансамбля линейных классификаторов, с числом базовых алгоритмов равным четырем, сложность алгоритма будет составлять O(n) = n.

Для хранения каждого классификатора понадобится выделить память для каждого вектора коэффициентов в виде b массивов типа double. Также необходимо выделить один массив под вектор входных значений. Итого получаем: $n * b + n = n(b + 1)$.

4. Нейронечеткие системы принятия решений

В настоящее время большую популярность набирают нейронечеткие системы принятия решений, которые объединяют в себе экспертные системы и обучение с учителем на основе нейросетей. В простейшем случае совместную модель можно рассматривать, как препроцессор, где механизм обучения искусственной нейронной сети (ANN) определяет правила нечеткого вывода (FIS). Как только параметры FIS определяются, ANN работает в обычном режиме. Функции принадлежности обычно аппроксимируются нейронной сетью из обучающих данных.

Принцип функционирования нейро-нечеткой модели в задачах автоматического управления может быть иллюстрирован на примере наиболее распространённой модели ANFIS (Adaptive-Network-Based Fuzzy Inference System). ANFIS – адаптивная сеть нечеткого вывода реализует нечёткую систему Такаги – Сугено и представляет собой пятислойную нейронную сеть прямого распространения сигнала (Рис. 14).

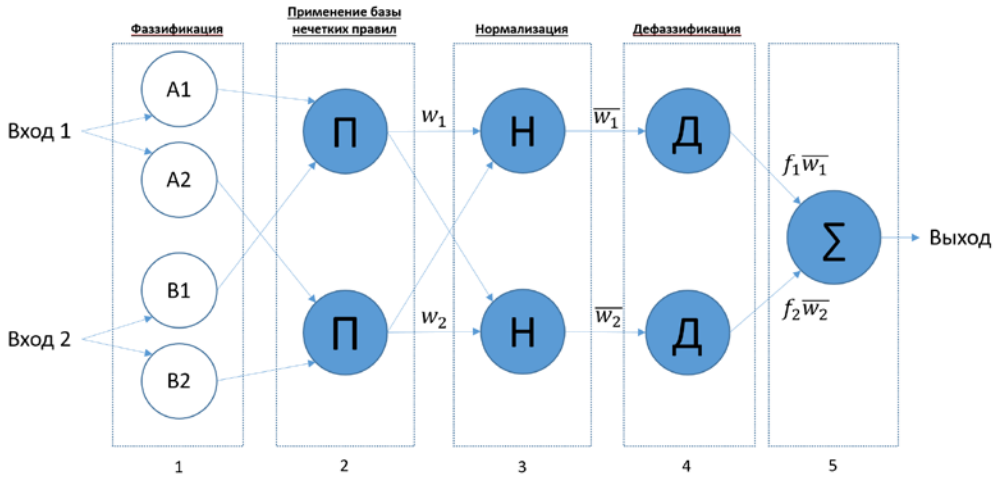

Рис. 14 - Нейронечеткая система ANFIS.

Входные переменные модели – управляемые переменные (на практике чаще всего применяется рассогласование между заданным и текущим значением управляемой переменной), выходные переменные у –оценка принадлежности к классу.

Первый слой определяет нечеткие термы входных параметров. Выходы узлов этого слоя представляют собой значения функции принадлежности при конкретных значениях входов $\mu_i(x_j)$.

Второй слой определяет посылки нечетких правил. Данный слой – неадаптивный. Каждый узел этого слоя соответствует одному нечеткому правилу. Узел второго слоя соединен с теми узлами первого слоя, которые формируют предпосылки соответствующего правила. Выходами узла μ_j является степень выполнения правила, которая рассчитывается как произведение входных сигналов.

Третий слой осуществляет нормализацию степеней выполнения правил

$$\overline{\omega_i} = \frac{\omega_i}{\sum \omega_j} \qquad (7)$$

Неадаптивные узлы этого слоя рассчитывают относительный вес выполнения нечеткого правила.

Четвертый слой определяет вклад каждого нечеткого правила в выход сети. Узел четвертого слоя рассчитывает вклад нечеткого правила v_i в выход сети.

Пятый слой формирует управляющий сигнал

$$y = \sum y_i \quad (8)$$

Выбор нейро-нечетких моделей осуществляется в зависимости от класса решаемых задач. Так, для интеллектуального управления наибольшее применение получили модели ANFIS, FALCON, GARIC, NEFCON, FUN.

Использование нейро-нечетких систем, ввиду их эффективности, находят применение в проектах, где количество входных сигналов сравнительно не велико и обучающая выборка недостаточно велика для обучения нейросети. Примером использования такой системы является Адаптивный нейро-нечеткий логический анализ на основе миоэлектрических сигналов [6, с. 23], Нейро-нечеткая система для характеристики движений рук [7, с. 23].

Нейронечеткие системы имеют более быструю сходимость, чем у обычных нейронных сетей, а для их обучения требуется меньший объем обучающей выборки. Однако с другой стороны получается высокая пространственная сложность, осцилляции при большом числе правил, сильное влияние выбросов и симметричная функция ошибки.

Пусть количество входов равно n, среднее количество термов равно a, а количество разбиений за N, то вычислительная сложность равна Нейронечеткий классификатор ANFIS представляет собой нейросеть, в которой применяются принципы нечеткой логики. Вычисление происходит по слоям (Таблица 6):

Таблица 6 – Анализ сложности реализации нейронечеткой системы

№	Описание	Вычислительная сложность
1	Выполняет раздельную фаззификацию, другими словами, определяет нечеткие термы входных параметров. Настраиваемыми в процессе обучения величинами являются параметры распределения. Соответственно, на первом шаге вычислительная сложность будет равняться количеству входных значений, умноженному на среднее количество термов.	$n * a$
2	В следующем слое выполняется применение нечетких правил, в которых входные переменные суммируются через нечеткую логическую операцию «И». При этом каждый нейрон предыдущего слоя связывается со всеми нейронами следующего слоя. В худшем случае количество этих правил будет равно среднему количеству термов в степени количества входов.	a^n
3	На третьем слое осуществляется нормализация степеней выполнения правил. То есть для каждого нейрона этого слоя нужно поделить	$a^n + 1$

	входное значение от предыдущего нейрона на сумму всего предыдущего слоя.	
4	На четвертом слое рассчитывают вклад каждого нечеткого правила. Количество операций равно числу нейронов предыдущего слоя. При этом вычисления происходят по формуле 1.	$a^n * 3N$
5	На пятом слое производится суммирование всех вкладов этих правил. Количество операций равно	a^n

Итого:

$$F(n) = n * a + a^n + a^n + 1 + a^n * 3N + a^n = 3(N+1) * a^n + 1$$

Вычислительная сложность в худшем случае равна O(n) = z^n. Что равно вычислительной сложности нечеткой логики.

Рассчитаем необходимое количество памяти (Таблица 7):

Таблица 7 - Приблизительная оценка требований к памяти нейронечеткой системы

№	Описание	Необходимое количество памяти
1	Нечеткие термы. Как и в случае с системой принятия решений на нечеткой логике, необходимое количество памяти равно 2*4 Байт, умноженное на количество входов и среднее количество термов.	$2 * n * a * 4$
2	Применение базы нечетких правил. В худшем случае количество этих правил будет равно среднему количеству термов в степени количества входов умноженному на количество символов операций в правилах. Для хранения символов используется тип char бесом один байт.	$a^n * n$
3	Для этапа 3 требуется массив размера a^n для хранения результатов.	$a^n * 8$
4	Дефаззификация. Требуется хранить $N + 2N$ чисел типа double.	$a^n * 3N * 8$

5	Для хранения суммы достаточно только одной переменной типа double.	8

Как итог, получаем $8 * n * a + a^n * n + a^n + a^n * 3N * 8 + 8 = (24N + 9) ** a^n + 8 * a(n + 1)$

Результаты

Сравнение параметров методов представлены в таблице 8.

Таблица 8 – Сравнение методов реализации системы принятия решений

Название метода	Вычислительная сложность	Потребность в обучающей выборке	Требования к памяти
Системы с нечеткой логикой	a^n	Не требуется.	$a^n * n + 2 * n * a + (N + +2N) * 8$
Нейронные сети	n^2	Требуется большая выборка	$3.4n^2 + 8n$
Ансамбль линейных методов	n	Требуется выборка	$n(b + 1)$
Нейронечеткие системы	a^n	Требуется выборка	$(24N + 9) * a^n + 8 * * a(n + 1)$

Наиболее важным фактором выбора алгоритма является сложность реализации алгоритма. Для того чтобы обучить нейросеть требуется большая выборка, так как при увеличении количества признаков и слоев, количество необходимых данных возрастает экспоненциально (проклятье размерности). Методы машинного обучения, в том числе и ансамблевые, требуют немного меньшую выборку за счет создания наиболее простых разделительных поверхностей. Настройка нейронечеткого классификатора потребует еще меньше данных, так как нейросеть в данном случае настраивает параметры нечеткой логики. Нечеткая система принятия решений не

требует обучающей выборки, но для ее корректной работы требуется прописать все правила вручную, что является весьма трудоемкой задачей. Это же относится и к нейронечетким системам.

Наиболее приемлемым вариантом с учетом вычислительной сложности и требований к памяти является использование ансамблевых алгоритмов с линейными алгоритмами или нейросетей. Системы, использующие нечеткую логику используют алгоритмы с вычислительной сложностью, возрастающие по экспоненте.

Наиболее требовательной к памяти является нейронечеткая система принятия решения.

Заключение

В данной работе был проведен анализ четырех систем принятия решения, используемых для управления схватом или в смежных тематиках. Были разобраны принципы работы этих алгоритмов, приведены преимущества и недостатки, а также посчитана вычислительная сложность.

Исходя из сделанных выводов по алгоритмам, наиболее подходящим в случае наличия достаточной выборки и достаточной вычислительной мощности, следует использовать систему принятия решения на основе нейронных сетей или ансамблевые алгоритмы. Они имеют относительно небольшую вычислительную сложность и помогают достичь высокой точности.

Библиографический список

1. Kakoty NM, Hazarika SM. Recognition of grasp types through principal components of DWT based EMG features. [Электронный ресурс], URL: https://www.ncbi.nlm.nih.gov/pubmed/22275601 (дата обращения: 20.12.2019).

2. Bahrami Moqadam S, Elahi SM, Mo A, Zhang W. Hybrid control combined with a voluntary biosignal to control a prosthetic hand. [Электронный ресурс], URL: https://www.ncbi.nlm.nih.gov/pubmed/30294521 (дата обращения: 20.12.2019).
3. Buccino AP, Kordovan M, Ness TV, Merkt B, Häfliger PD, Fyhn M, Cauwenberghs G, Rotter S, Einevoll GT. Combining biophysical modeling and deep learning for multielectrode array neuron localization and classification. [Электронный ресурс] URL: https://www.ncbi.nlm.nih.gov/pubmed/29847231 (дата обращения: 20.12.2019).
4. Ming Liu. An Adaptive Classification Strategy for Reliable Locomotion Mode Recognition. [Электронный ресурс], URL: https://vk.com/away.php?to=https%3A%2F%2Fres.mdpi.com%2Fd_attachment%2Fsensors%2Fsensors-17-02020%2Farticle_deploy%2Fsensors-17-02020.pdf&cc_key= (дата обращения: 20.12.2018).
5. Rossi M, Della Santina C, Piazza C, Grioli G, Catalano M, Biechi A. Preliminary results toward a naturally controlled multi-synergistic prosthetic hand. [Электронный ресурс], URL: https://www.ncbi.nlm.nih.gov/pubmed/28814009 (дата обращения: 21.12.2019).
6. Favieiro GW, Balbinot A. Adaptive neuro-fuzzy logic analysis based on myoelectric signals for multifunction prosthesis control. [Электронный ресурс], URL: https://www.ncbi.nlm.nih.gov/pubmed/22256169 (дата обращения: 21.12.2019).
7. Alexandre Balbinot* and Gabriela Favieiro. A Neuro-Fuzzy System for Characterization of Arm Movements. [Электронный ресурс], URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3649412/ (дата обращения: 21.12.2019).
8. Андриевская Н.В. Резников А.С. Черанев А.А. Особенности применения нейро-нечетких моделей для задач синтеза систем автоматического управления //

Фундаментальные исследования. -2014. - №11. – С.1445-1449. [Электронный ресурс]. - URL: https://www.fundamental-research.ru/ru/article/view?id=35784 (дата обращения: 22.12.2019).

9. Демидова Г.Л., Лукичева Д.В. Регуляторы на основе нечеткой логики в системах управления техническими объектами. - Санкт-Петербург: Университет ИТМО, 2017 г. – 84 с.
10. Орельен Жерон [Aurélien Geron] Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow: пер. с англ. – Москва Санкт-Петербург: ООО «Диалектика» 2019. – 683 с.
11. Саймон Хайкин [Simon Haykin]. Нейронные сети полный курс: пер. с англ. 2-е издание – Москва, Санкт-Петербург, 2018. – 1103 с.
12. Томас Кормен [Thomas Cormen], Чарльз Лейзерсон, Рональд Ривест, Клиффорд Штайн. Алгоритмы: построение и анализ: пер. с англ. 3-е издание – Москва, Санкт-Петербург: ООО «Диалектика», 2019 – 1328 с.
13. Nicholas D. Lane, Sourav Bhattacharya, Akhil Mathur, Petko Georgiev. Squeezing Deep Learning into Mobile and Embedded Devices [Электронный ресурс]. URL: https://www.researchgate.net/publication/318738792_Squeezing_Deep_Learning_into_Mobile_and_Embedded_Devices (дата обращения: 29.04.2020).
14. Sérgio Branco, André G. Ferreira and Jorge Cabral. Machine Learning in Resource-Scarce Embedded Systems, FPGAs, and End-Devices: A Survey. [Электронный ресурс]. URL: https://www.mdpi.com/2079-9292/8/11/1289/htm (дата обращения: 29.04.2020).

Оригинальность 75%