МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ
Северо-Кавказский филиал
ордена Трудового Красного Знамени федерального государственного бюджетного образовательного учреждения высшего образования
«Московский технический университет связи и информатики»

МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ
для проведения лабораторных работ по дисциплине «Системы искусственного интеллекта»

Кафедра **«Информатика и вычислительная техника»**

Направление подготовки **09.03.01. Информатика и вычислительная техника**

Профиль **Вычислительные машины, комплексы, системы и сети, Программное обеспечение и интеллектуальные системы**

Формы обучения **очная, заочная**

Разработала:
Доцент кафедры ИВТ Швидченко С.А.

Содержание

Лабораторная работа №1.
Задача линейного разделения двух классов. Линейное разделение классов. Алгоритм обучения персептрона по отдельным примерам. Геометрическая интерпретация линейного разделения классов. Настройка весового вектора.

Аннотация: Рассматриваются многослойные рекуррентные сети (персептроннаяbreak сеть с обратной связью, рекуррентная сеть Эльмана, сеть RTRN) и их использование для идентификации динамических объектов.

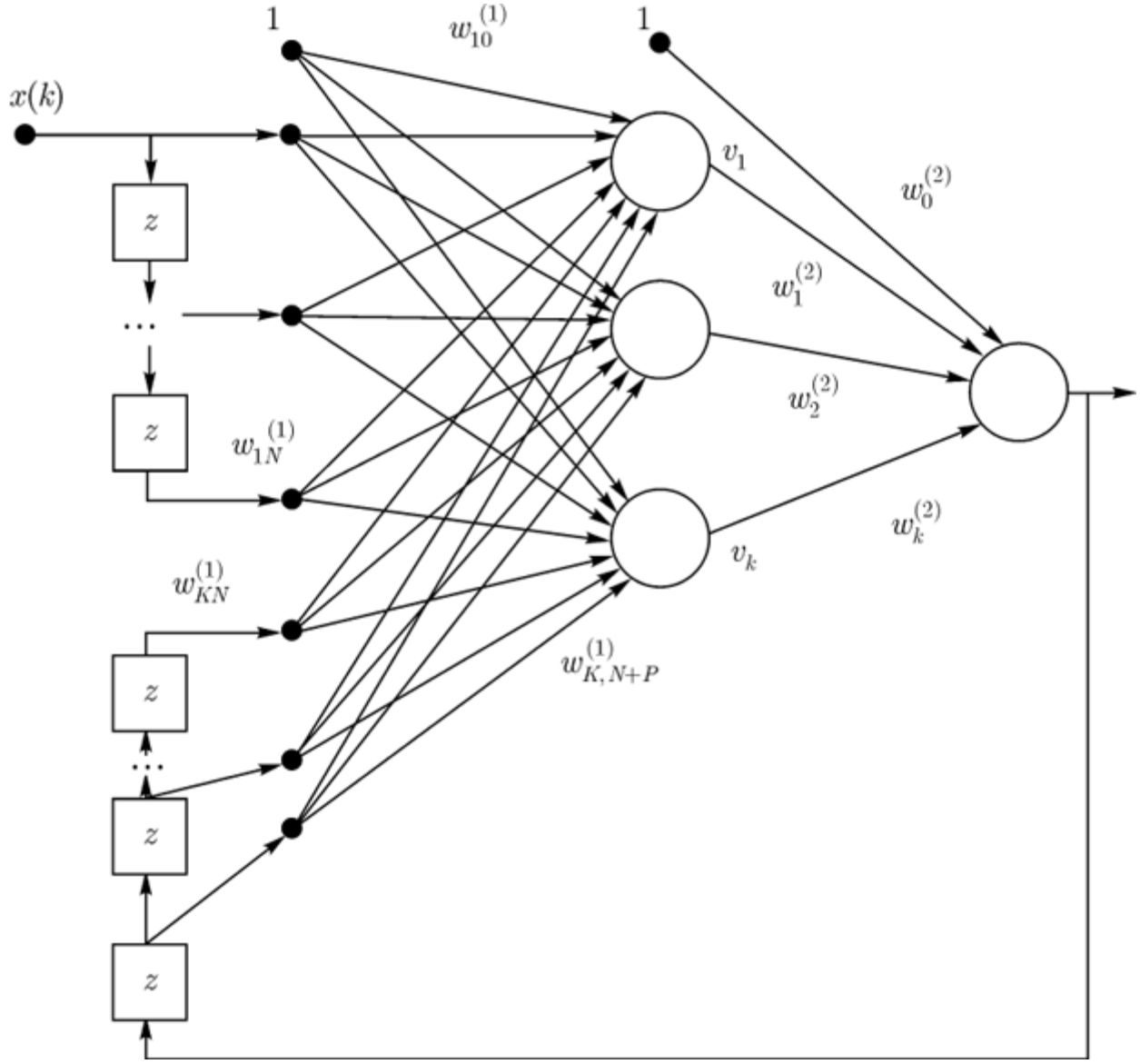
Введение

Многослойные *рекуррентные сети* представляют собой развитие однонаправленных сетей персептронного типа за счет добавления в них соответствующих обратных связей. Обратная *связь* может исходить либо из выходного, либо из скрытого слоя нейронов. В каждом контуре такой связи присутствует элемент единичной задержки, благодаря которому *поток* сигналов может считаться однонаправленным (выходной сигнал предыдущего временного *цикла* рассматривается как априори заданный, который просто увеличивает *размерность* входного вектора сети). Представленная подобным образом *рекуррентная сеть*, с учетом способа формирования выходного сигнала, функционирует как однонаправленная персептронная *сеть*. Тем не менее, *алгоритм* обучения такой сети, адаптирующий значения синаптических весов, является более сложным из-за зависимости сигналов в момент времени t от их значений в предыдущие моменты и соответственно из-за более громоздкой формулы для расчета вектора градиента.

При обсуждении *рекуррентных сетей*, в которых в качестве выходного элемента используется многослойный *персептрон*, рассмотрим наиболее известные структуры сетей *RMLP*, *RTRN*, Эльмана.

Персептронная сеть с обратной связью

Один из простейших способов построения рекуррентной сети на базе однонаправленной НС состоит во введении в персептронную *сеть* обратной связи. В дальнейшем мы будем сокращенно называть такую *сеть RMLP* (англ.: *Recurrent MultiLayer Perceptron* - рекуррентный многослойный *персептрон*). Ее обобщенная структура представлена на рис. 1 (z - единичные элементы *запаздывания*).



увеличить изображение

Рис. 1. Структура сети RMLP

Это *динамическая сеть*, которая характеризуется запаздыванием входных и выходных сигналов, объединяемых во *входной вектор* сети. Рассуждения будут касаться только одного входного узла $x(k)$ и одного выходного нейрона, а также одного скрытого слоя. Такая система реализует *отображение*:

$$y(k+1) = f(x(k), x(k-1), \ldots, x(k-(N-1)), y(k-1), \ldots, y(k- \quad (1) \quad 1)$$

где $N-1$ - количество задержек входного сигнала, а P - количество задержек выходного сигнала. Обозначим K количество нейронов в скрытом слое. В этом случае *сеть RMLP* можно характеризовать тройкой чисел (N, P, K). Подаваемый на вход сети *вектор* x имеет вид:

$$x(k) = [1, x(k), x(k-1), \ldots, x(k-(N-1)),$$
$$y(k-P), y(k-P+1), \ldots, y(k-1)]^T.$$

Допустим, что все нейроны имеют сигмоидальную функцию активации. Обозначим u_i взвешенную сумму сигналов i -го нейрона скрытого слоя, а g - взвешенную сумму сигналов *выходного нейрона*. При введенных обозначениях выходные сигналы конкретных нейронов описываются зависимостями

$$u_i = \sum_{j=0}^{N+P} w_{ij}^{(1)} x_j$$
$$v_i = f(u_i)$$
$$g = \sum_{i=0}^{K} w_i^{(2)} v_i$$
$$y = f(g)$$

Сеть RMLP повсеместно применяется для моделирования динамических процессов в режиме "*онлайн*". Типичным примером ее приложения может служить имитация нелинейных динамических объектов, для которых *сеть RMLP* выступает в роли модели, а *алгоритм* уточнения весов - в роли процедуры идентификации параметров этой модели (рис. 2). Идентифицированная модель может в последующем использоваться для управления данным объектом. Именно по этой причине сети *RMLP* наиболее популярны для имитации систем управления машинами, устройствами и динамическими процессами.

В результате сравнения выходного сигнала модели $y(k)$ с выходным сигналом динамического объекта $d(k)$ рассчитывается *значение* погрешности $e(k) = y(k) - d(k)$, управляющей процессом уточнения параметров нейронной сети. Символом M на рис. 2 обозначен коэффициент усиления модуля, масштабирующего выходной сигнал сети $y(k)$ таким образом, чтобы его динамический уровень лежал в том же диапазоне, что и уровень выходного сигнала динамического объекта $d(k)$.

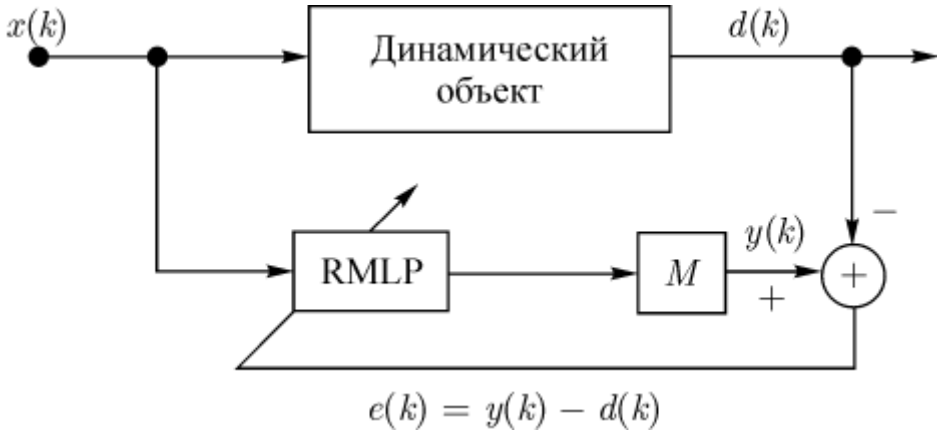


Рис. 2. Схема включения сети RMLP при решении задачи идентификации

Рекуррентная сеть Эльмана

Рекуррентная сеть Эльмана характеризуется частичной рекуррентностью в форме обратной связи между скрытым и входным слоем, реализуемой с помощью единичных элементов *запаздывания* z. Обобщенная структура этой сети представлена на рис. 3.

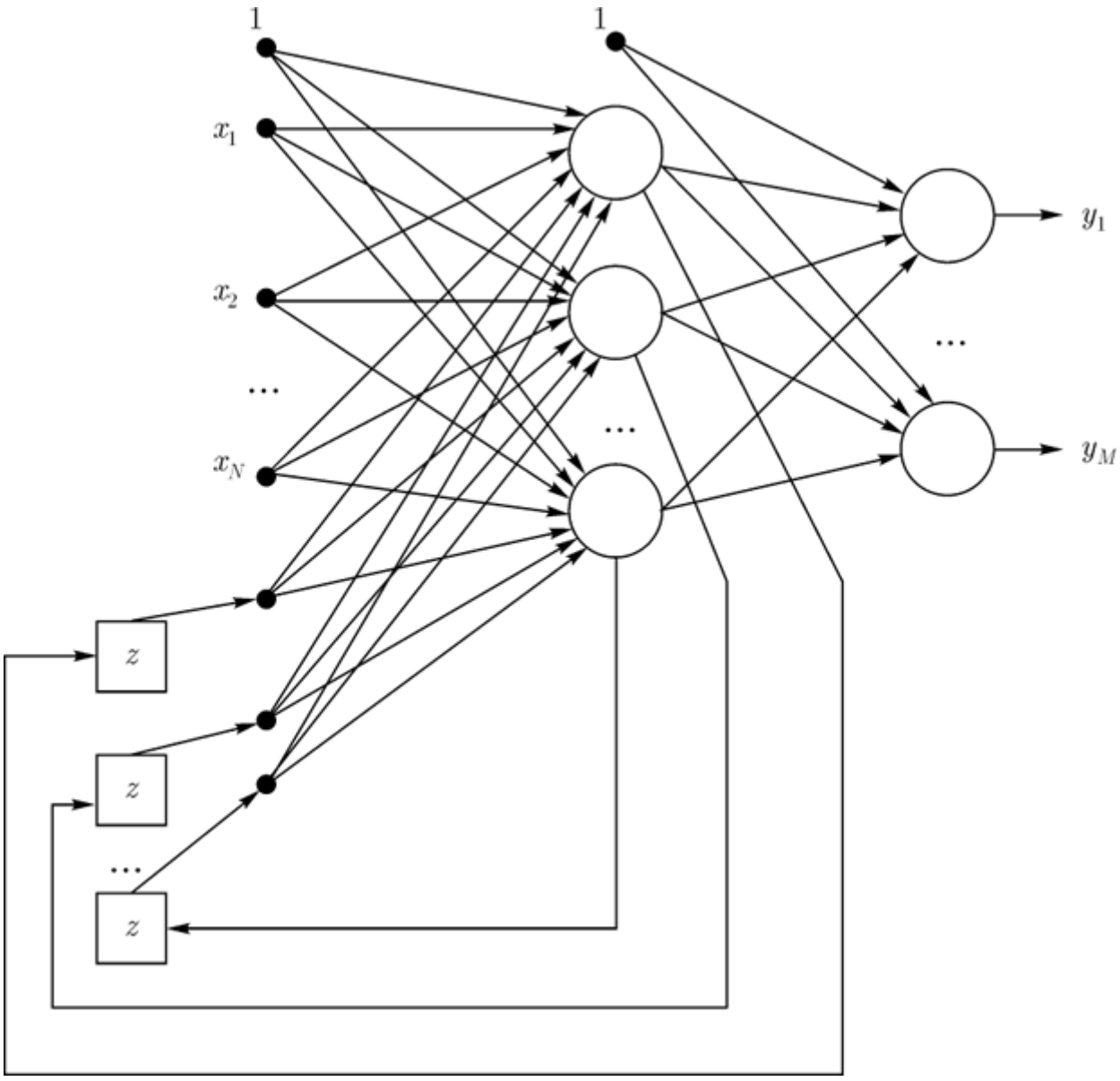
Каждый скрытый *нейрон* имеет свой аналог в контекстном слое, образующем совместно с внешними входами сети *входной* слой. *Выходной слой* состоит из нейронов, однонаправленно связанных только с нейронами скрытого слоя, подобно сети *RMLP*. Обозначим внутренний *вектор* возбуждения сети x (в его состав входит также единичный сигнал поляризации), состояния скрытых нейронов - $v \in R^K$, а выходные сигналы сети - $y \in R^M$. При таких обозначениях *входной вектор* сети в момент t имеет форму

$$X(k) = [x_0(k), x_1(k), \ldots, x_N(k), v_1(k-1), v_2(k-1), \ldots, v_K(k-1)].$$

Веса синаптических связей первого (скрытого) слоя сети обозначим $w_{ij}^{(1)}$, а второго (выходного) слоя - $w_{ij}^{(2)}$. Если взвешенную сумму i -го нейрона скрытого слоя обозначить u_i, а его выходной сигнал - v_i, то

$$u_i(k) = \sum_{j=0}^{N+K} w_{ij}^{(1)} x_j(k),$$

$$v_i(k) = f_1(u_i(k)).$$



увеличить изображение

Рис. 3. Структура сети Эльмана

Веса $w_{ij}^{(1)}$ образуют матрицу $W^{(1)}$ синаптических связей скрытого слоя, а $f_1(u_i)$ - *функция активации* i -го нейрона этого слоя. Аналогично можно обозначить взвешенную сумму i -го нейрона выходного слоя g_i, а соответствующий ему выходной сигнал сети - y_i. Эти сигналы описываются формулами

$$g_i(k) = \sum_{i=0}^{K} w_{ij}^{(2)} v_j(k),$$

$$y_i(k) = f_2(g_i(k)).$$

В свою *очередь*, веса $w_{ij}^{(2)}$ образуют матрицу $W^{(2)}$, описывающую синаптические связи нейронов выходного слоя; $f_2(g_i)$ - *функция* активации i -го нейрона выходного слоя.

Сеть Эльмана естественным образом предназначена для моделирования временных рядов. В частности, она решает задачу прогнозирования амплитуды сигнала на основе текущего значения входного сигнала и запомненных значений из предыдущего временного *цикла*. Задача прогноза временных рядов возникает в финансовой области: прогноз котировок товаров и ценных бумаг на бирже, курсов валют, показателей банковской деятельности. В экономике прогноз может быть связан, например, с анализом уровня производства в различных отраслях промышленности и сельского хозяйства, процента трудовой занятости населения, роста средней заработной платы. Проблема прогноза временных рядов возникает в многочисленных экологических задачах и самых разнообразных технических приложениях.

Для прогноза временных рядов могут применяться *статистические методы*. В этом случае должна быть построена *динамическая модель* данных (например, регрессионная модель) изучаемого явления. Для простейших задач такая модель может быть построена известными методами. Однако для практических задач, примеры которых приведены выше, построение подобной динамической модели представляет собой сложную аналитическую задачу. Эти приложения связаны обычно не со скалярными, а с векторными временными рядами. Например, в финансовой сфере прогноз котировок товара зависит от вектора динамических данных, которые включают цены открытия и закрытия торговой сессии, среднюю и максимальную цены торговой сессии, суммарный уровень заявок, *валютные курсы* и пр.

В том случае, когда адекватной математической модели изучаемых временных рядов не существует, удобным инструментом для решения задачи прогноза является нейросетевой экстраполятор динамических данных.

Задача прогноза векторного временного ряда ставится следующим образом:

- задана реализация временного ряда $x_j = (x_{j1}, x_{j2}, \ldots, x_{jM})$, $j = 1, 2, \ldots, T$, на интервале времени $[\Delta, T\Delta]$ с постоянным интервалом дискретности Δ;
- требуется построить оценку значения временного ряда (обычно одной его координаты) в момент времени $(T + t_{pr})\Delta$, где $t_{pr}\Delta$ - заданное время прогноза.

Из логических соображений или путем статистического анализа имеющейся реализации можно установить, сколько предшествующих значений относительно произвольного текущего момента времени $j\Delta$ определяюще связаны с прогнозируемым значением. Это означает, что если представить прогнозируемое *значение* $y_j = x_{j+t,m}$ m -ой *координаты* вектора x как функцию его предшествующих измерений:

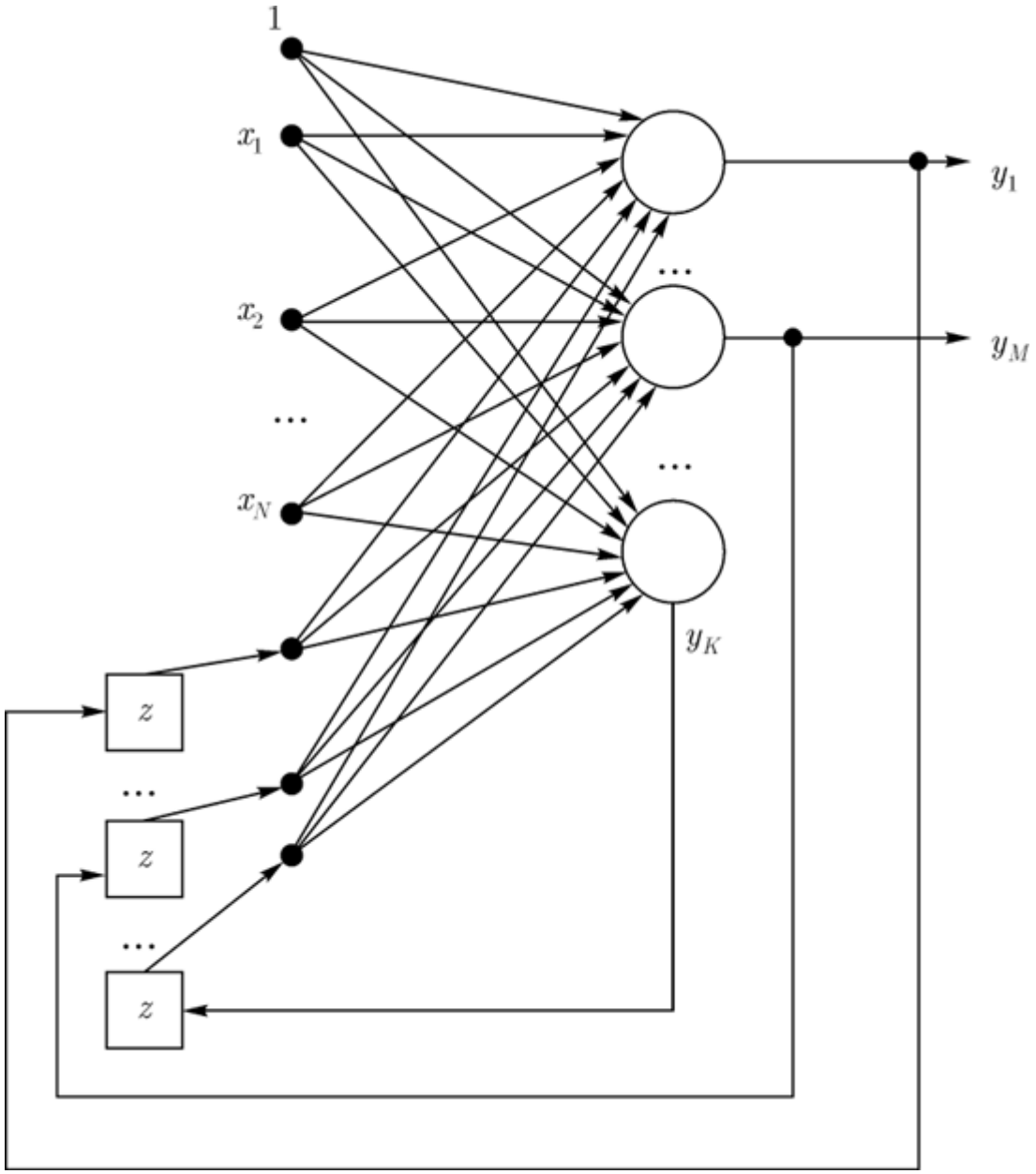
$$y_j = \Phi(x_j, x_{j-1}, \ldots, x_{j-q+1}),$$

то выбор значения q устанавливает "*память*" экстраполятора. *Значение* q определяет также *входной вектор* для нейронной сети, которая строится для решения задачи прогноза. *Размерность* этого вектора равна $M * q$.

Таким образом, задача прогноза данных на нейронной сети сведена к задаче воспроизведения функции многих переменных $\Phi(x_j, x_{j-1}, \ldots, x_{j-q+1})$ по данным обучающей выборки.

Сеть RTRN

Среди *рекуррентных сетей* особого внимания заслуживает *сеть* типа *RTRN* (англ.: *Real Time Recurrent Network*), предложенная Р.Вильямсом и Д.Зипсером и предназначенная для обработки сигналов в реальном времени. *Сеть RTRN* - частный случай сети Эльмана.



увеличить изображение

Рис. 4. Структура сети RTRN

Обобщенная структура сети представлена на рис. 4. *Сеть* содержит N входных узлов, K *скрытых нейронов* и K соответствующих им узлов контекстного слоя. Из K *скрытых нейронов* только M составляют *выход* сети. Обозначим взвешенную сумму i - го нейрона скрытого слоя u_i, а *выход* этого нейрона - y_i. *Вектор* $x(k)$ и смещенный (задержанный) на один цикл *вектор* $y(k-1)$ образуют расширенный *вектор* активации $x(k)$, возбуждающий нейроны сети:

$$x(k) = [1, x_1(k), x_2(k), \ldots, x_N(k), y_1(k-1), \ldots, y_K(k-1)]^T.$$

После описания входного вектора сети в момент t можно определить состояние всех нейронов согласно зависимостям:

$$u_i(k) = \sum_{j=0}^{N+K} w_{ij} x \quad 2)$$

$$y_i(k) = f(u_i(k \quad 3)$$

причем $f()$ обозначает непрерывную функцию активации нейрона (как правило, сигмоидальную). На рис. 4 видно, что *сеть RTRN* представляет собой частный случай сети Эльмана, в которой веса выходного слоя постоянны и равны дельте Кронекера, т.е. $w_{ij} = \delta_{ij} = 1$ для $i = j$ или 0 для $i \neq j$. В этом случае можно применять *алгоритм* обучения Вильяма-Зипсера.

1. Выбрать случайные начальные значения весов сети, составляющих матрицу W и равномерно распределенных в заданном интервале (обычно в диапазоне от -1 до 1).

2. Рассчитать состояние всех K нейронов для очередного момента $t = 0, 1, 2, \ldots$ с использованием формул (1) и (2). На этой основе можно определить *входной вектор* $x(k)$, возбуждающий нейроны в момент t .

3. Рассчитать значения

$$dy_i(k)/dw_{ab} = (df_1(u_i)/du_i)[\delta_{ja} x_b + \sum_{k=1}^{K} (dy_i(k-1)/dw_{ab}) w_{i,k+N}]$$

4. Уточнить значения весов по алгоритму наискорейшего спуска согласно формуле

$$w_{ab}(k+1) = w_{ab}(k) - \alpha \sum_{i=1}^{K} [y_i(k) - d_i(k)](dy_i(k)/dw_{ab})$$

для $a = 1, 2, \ldots, K$ и $b = 0, 1, 2, \ldots, N + K$.

Шаги (2-4) повторять вплоть до стабилизации значений всех весов сети.

Контрольные вопросы ЛР (ПК-4):

1. Функциональная структура использования СИИ.
2. Модели и методы решения задач.
3. Классификация представления задач.
4. Логические модели.
5. Нейрофизиологическая аналогия.

6. Реализация булевых функций нейронными сетями.
7. Каково назначение основных элементов управления Lazarus?
8. Какие существуют типы элементов управления Lazarus?
9. Каково назначение свойств элементов управления в Lazarus?
10. Как можно изменить свойства элементов управления?
11. Как влияют свойства на интерфейс программы?
12. Понятие информационной системы. Классы ИС.
13. Структура однопользовательской и многопользовательской, малой и корпоративной ИС.
14. Основные особенности современных проектов ИС.
15. Как называются информационные системы, ориентированные на коллективное использование информации членами рабочей группы и чаще всего строящиеся на ЛВС.
16. Как называются информационные системы, основанные гипертекстовых документах и мультимедиа.

Лабораторная работа №2.
Виды нейронных сетей и способы организации их функционирования. Виды сетей. функционирование сетей. настройка нейронных сетей для решения задач. Предобработка данных. Интерпретация ответов сети. оценка способности сети решить задачу. Сдача теста № 2.

Адаптивная резонансная теория (APT)

Серьезная проблема для нейронных сетей - правильное соотношение стабильности и *пластичности* при запоминании образов. Существуют наборы эталонов (даже состоящие всего из 4-х векторов), которые при циклическом предъявлении в обучении дают никогда не сходящиеся наборы параметров сети. Предъявление всего одного нового образа в обучающем множестве часто приводит к долгому *переобучению*. Если *сеть* работает в реальном времени, например, обрабатывает сенсорную информацию, то обучающее множество может все время меняться. Для большинства моделей нейронных сетей это приводит к отсутствию обучения вообще.

Человеческая *память*, напротив, эффективно хранит и корректирует запоминаемые образы. Ни предъявление нового образа, ни изменение старых не приводит к уничтожению памяти или невозможности запоминания. Даже удаление части нервной ткани чаще всего не прерывает работу сети и не стирает запомненные образы, а лишь делает их менее четкими.

Сеть APT - попытка приблизить механизм запоминания образов в искусственных НС к биологическому. Результатом работы *APT* является устойчивый набор запомненных образов и возможность выборки "похожего" вектора по произвольному предъявленному на входе вектору. Важное качество *APT* - динамическое запоминание новых образов без полного *переобучения* и отсутствие потерь уже запомненных образов при предъявлении новых.

Сеть APT-1

Сеть APT-1 предложена Карпентером и Гроссбергом в 1986 г. Она представляет собой векторный *классификатор* и обучается без учителя, лишь на основании предъявляемых входных векторов. *APT*-1 работает только с двоичными векторами, состоящими из нулей и единиц. Позже было предложено много разновидностей этой модели. *APT*-2 запоминает и классифицирует непрерывные входные векторы. *Группа* моделей с суффиксом "*MAP*" (ARTMAP и др.) классифицирует и входные, и выходные вектора, а также строит связи между ними.

Архитектура и работа

Структура сети *APT*-1 (далее *APT*) представлена на рис. 1. *Входной вектор* сети $x = x_1, \ldots, x_n, \ldots, x_N$ имеет N *компонент*. В слое распознавания запоминается M классов образов, по одному классу на каждый *нейрон* $m = 1, \ldots, M$.

Основную работу по классификации производят слой сравнения и слой распознавания. Схемы приемников (Прм1, Прм2) и схема сброса управляют режимом работы сети и могут быть реализованы в виде обычных логических схем или в виде нейронов.

Работа блоков *APT* определяется следующими формулами:

$$\text{Прм1} : G1 = (\vee_n x_n) \wedge \neg(\vee_m R_m).$$

Выход Прм1 обеспечивает единичный сигнал для слоя сравнения, если на вход сети подан *вектор* (нулевой *вектор* на входе недопустим) и если *выход* слоя распознавания равен нулю.

$$\text{Прм2} : G2 = \vee_n x_n$$

Если на вход подан *вектор* x, то блок Прм2 формирует на выходе единичный сигнал и тем самым разрешает работу слоя распознавания.

Схема сброса: $G3 = (\sum_n C_n / \sum_n x_n) < \rho$.

Проверяет критерий сходства для векторов x и C. Критерий состоит в сравнении количества единиц в векторах x, C. Количества единиц сравниваются в виде отношения с некоторым пороговым уровнем сходства ρ. Если порог не превышен, то сходство считается плохим и схема сброса вырабатывает сигнал торможения для нейрона в слое распознавания. *Выход* схемы сброса - двоичный *вектор* с M компонентами. Схема сброса является динамической и "помнит" свое состояние в течение одной классификации. Порог ρ является *внешним параметром* по отношению к сети и задается пользователем в интервале от 0 до 1. Чем меньше ρ, тем менее похожие векторы будут отнесены сетью к одному классу.

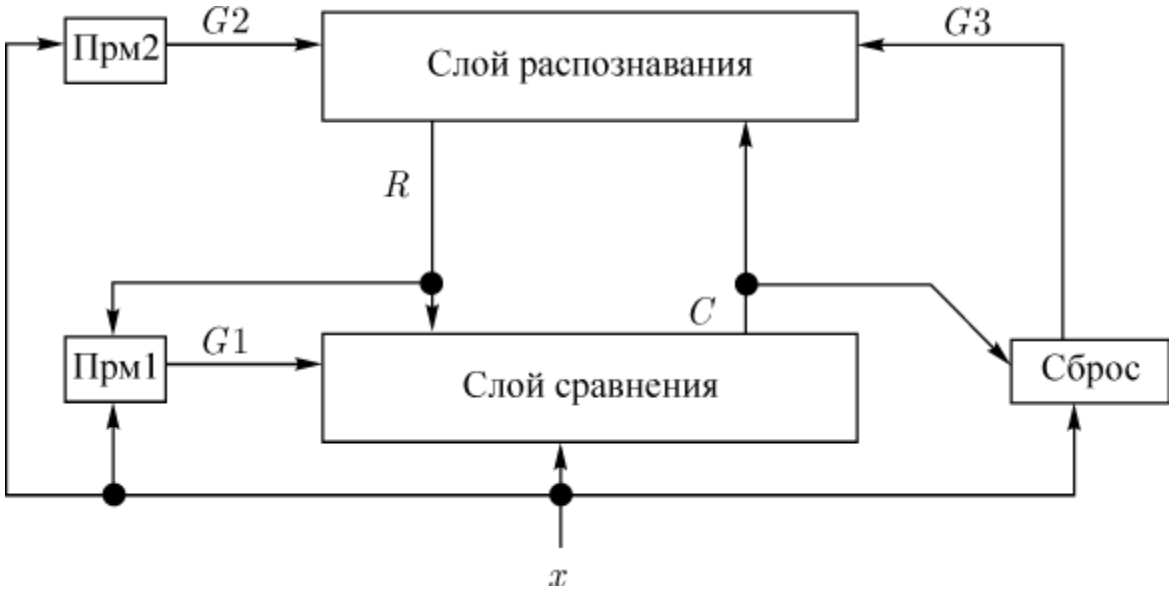


Рис. 1. Структурная схема АРТ

Слой сравнения

Каждый *нейрон* в слое сравнения имеет порог, равный двум. На вход одного нейрона в слое сравнения подаются: сигнал $G1$ с единичным весом, одна компонента x^n с единичным весом и все выходы слоя распознавания, M *компонент* с вектором весов T^n, где n - номер нейрона в слое сравнения. Весовые *коэффициенты* T - двоичные. В нейроне используется нелинейность в виде жесткой ступеньки: если активация нейрона NET_n превышает порог $\Theta = 2$, то на выходе нейрона будет *единица*, иначе - ноль. Это "правило 2/3": для активации нейрона достаточно два сигнала из трех.

Работа слоя определяется формулами:

$$P_n = T^n R = \sum_m T_m^n R_m$$

$$NET_n = P_n + x_n + G1$$

$$C_n = \{0, \text{ если } NET_n < 2; 1, \text{ если } NET_n \geqslant 2\}$$

Работой слоя управляет сигнал $G1$. Если $G1 = 1$, то x проходит без изменений на *выход* слоя сравнения, благодаря лишнему единичному сигналу $G1$ на входе нейрона. Если $G1 = 0$, то на выходе имеем $C = x \wedge P$, т.е. *вектор* C будет логическим произведением двоичных векторов x и P.

Слой распознавания

Каждый *нейрон* в слое распознавания имеет следующие входы: один сигнал $G2$ с единичным весом, одна компонента $G3_m$ с большим отрицательным весом (m - номер нейрона) и N сигналов со слоя сравнения с вектором весов B^m (у вектора B^m всего N *компонент*, $B_1^m, \ldots, B_N^m$).

Нейроны слоя распознавания не содержат нелинейных элементов, но обладают следующей особенностью. Каждый *нейрон* в слое связан со всеми остальными нейронами этого же слоя обратными тормозящими связями и положительной обратной связью - с самим собой (как во втором слое сети Хеммингa, см. Лекцию 10).

Такой способ связности называется латеральным торможением. Это приводит к тому, что только один *нейрон* в слое распознавания может быть активирован. Между нейронами существует конкуренция, и *нейрон* с максимальным выходом "подавляет" все остальные нейроны в слое, выигрывая "состязание". Его *выход* становится равным единице, остальных нейронов - нулю, т.е. *вектор* R имеет только одну единичную компоненту, остальные - нули.

Веса B^m имеют действительные значения. Работа слоя определяется формулой:

$$R_m = f(B^m, C) \wedge G2 \wedge \neg(G3_m),$$

где $f(B^m, C)$ - *выход* m -го нейрона, равный нулю или единице.

Отсюда видно, что сигнал $G2$ "разрешает" работу слоя распознавания, а сигнал $G3$ позволяет выборочно затормозить любые нейроны в слое.

Работа сети АРТ

Решение задачи классификации с помощью *АРТ* содержит следующие этапы: *инициализация*, *распознавание*, сравнение, *поиск*, обучение.

1. *Инициализация*.

а) выбираем *параметр* ρ, исходя из требуемой детальности классификации;

б) создаем *сеть* в памяти. Количество нейронов должно быть достаточным, чтобы запомнить все ядра классов (до M). Изначально все нейроны слоя распознавания считаются "невыделенными", их веса приравниваются к одинаковым небольшим значениям:

$$B_n^m = B_{\text{нач}} < L/(L+N-1),$$

где $L > 1$ - некоторая константа (обычно $L = 2$). Веса в слое сравнения также выбираются одинаковыми, равными единице: $T_m^n = 1$.

Такой выбор весов обеспечивает остановку поиска на невыделенном нейроне, если нет подходящих выделенных нейронов, и правильное обучение.

2. *Распознавание.*

а) предъявляем *вектор* x на входе. До этого момента $G2 = 0$ и *выход* слоя распознавания равен нулю: $R = 0$.

б) у вектора x есть ненулевые компоненты, поэтому $G1$ становится равным единице, т.к. $R = 0$. Сигнал $G1$ "подпитывает" нейроны слоя сравнения и x без изменений проходит через слой сравнения: $C = x$.

в) весовые *коэффициенты* B^m имеют смысл нормированных ядер классов. В слое распознавания активируется несколько нейронов, но благодаря латеральному торможению остается один *нейрон* с выходом $R_{m0} = 1$, а остальные тормозятся. m_0 - номер выигравшего нейрона.

3. Сравнение.

а) *выход* $R \neq 0$ приводит к $G1 = 0$, что снимает "подкачку" нейронов в слое сравнения. Весовые *коэффициенты* T^n имеют смысл ненормированных двоичных ядер классов. На вход слоя сравнения передается один ненулевой *выход* слоя распознавания, $R_{m0} = 1$. Эта *единица* умножается на весовые *коэффициенты*, давая в сумме сигнал

$$NET_n = x_n + T_{m0}^n.$$

Порог всех нейронов равен 2, поэтому *выход* слоя сравнения равен

$$C_n = x_n \wedge T_{m0}^n.$$

Следовательно, *выход* слоя сравнения на этом этапе - логическое *произведение* входного сигнала и двоичного ядра класса из слоя сравнения.

б) *модуль* сброса вычисляет второй критерий сходства (первый - *максимум* произведения (B^m, x) в слое распознавания). Если количества единиц в векторе C и векторе x близки, то сходство считается хорошим и выносится решение о принадлежности вектора x к классу $m0$.

4. *Поиск*.

а) если критерий сходства не выполняется, схема сброса вырабатывает сигнал $G3_{m0} = 1$, который тормозит *нейрон* m_0 в слое распознавания. Сигнал $G3_{m0} = 1$ остается равным 1 до окончания данной классификации. *Выход* нейрона m_0 становится равным 0, а, следовательно, и весь *вектор* $R = 0$. Сигнал $G1$ становится равным нулю и *вектор* x снова проходит через слой сравнения без изменений, вызывая новый цикл поиска (шаги 2в-3б), пока критерий сходства не будет удовлетворен.

При соответствующем выборе начальных значений весов B *поиск* всегда закончится на нераспределенном нейроне слоя распознавания. Для него будет выполнен критерий сходства, т.к. все веса T равны 1. Если все нейроны выделены и критерий сходства не выполняется, следует аварийная остановка либо расширение сети введением нового нейрона в слое распознавания и новых входов в слое сравнения.

5. Обучение.

Независимо от того, найден ли на этапе поиска распределенный *нейрон* или нераспределенный, обучение протекает одинаково. Корректируются лишь веса выигравшего нейрона m_0 в слое распознавания и веса T^n_{m0} для всех n в слое сравнения.

Различают быстрое и медленное обучение. При быстром обучении коррекции весов имеют вид:

$$B_n^{m0} = LC_n/(L + \sum_n C_n - 1) \equiv \Delta B_n^{m0},$$

где $L > 1$ - константа.

Веса в слое сравнения - двоичные: T^n_{m0}.

В результате такого алгоритма обучения ядра T изменяются, несущественные компоненты обнуляются в процессе обучения. Если какая-то компонента вектора T^n стала нулевой на какой-то итерации обучения, она никогда не вернется к единице. В этом проявляется *асимметрия АРТ* по отношению к значениям 0 и 1. Эта *асимметрия* имеет серьезные отрицательные последствия для модели, приводя к деградации ядер классов в случае зашумленных входных векторов.

Медленное обучение меняет ядра малыми коррекциями:

$$B_n^{m0} \longrightarrow \beta \Delta B_n^{m0} + (1 - \beta) B_n^{m0},$$
$$T^n_{m0} \longrightarrow \beta C_n + (1 - \beta) T^n_{m0},$$

где β мало и характеризует скорость обучения.

В результате каждой итерации обучения ядра меняются незначительно.

Видно, что веса B в любой момент времени могут быть однозначно рассчитаны через веса T, таким образом, *кодирование* информации о ядрах в *АРТ* в рассмотренной модели является избыточным в смысле расхода памяти.

Необходимость поиска

В сети *APT* используются два критерия "похожести" векторов. Первый - *максимум* скалярного произведения $max_m(B^m, x)$ при выборе "победителя" в слое распознавания. Второй - критерий сходства в блоке сброса:

$$\left(\sum_n C_n / \sum_n x_n\right)\Big|_{x,C} \geqslant \rho.$$

Таким образом, задача классификации в сети *APT* состоит в том, чтобы найти *ядро* с максимальным скалярным произведением (B^m, x), соблюдая при этом условие выполнения критерия сходства. Эти два критерия не являются эквивалентными, поэтому и фаза поиска, и фаза распознавания являются необходимыми и не могут быть опущены.

Положительные качества и недостатки APT

Сеть APT решает дилемму стабильности-*пластичности* и позволяет быстро запоминать новые образы без утраты старых. Как и в случае других моделей НС, на обычных машинах фон-неймановского типа сети работают медленно и неэффективно. Для решения задачи нужно найти *максимум* скалярного произведения, что требует около $3NM$ операций с плавающей запятой, и вычислить в худшем случае M критериев сходства. Для этого необходимы существенные вычислительные *затраты*. На параллельном компьютере *операции* расчета скалярных произведений могут быть распараллелены, но расчет критериев сходства все равно выполняется последовательно. Таким образом, даже на параллельной машине *сеть APT* является требовательной к ресурсам.

Тем не менее, одна *итерация* для запоминания каждого входного вектора - редкая экономичность для нейронных сетей. Вспомним, что многослойный *персептрон* для запоминания нового вектора требует полного *переобучения*.

У сети *APT* есть несколько существенных недостатков.

1. Чувствительность к порядку предъявления векторов. Большинство разновидностей *APT* весьма чувствительны к порядку предъявления входных векторов x. Картины ядер классов, сформированные сетью, принципиально меняются при различных видах упорядочения.
2. Невозможность классификации зашумленных векторов. Пусть входные векторы содержат шум.

Если компонента незашумленного входного вектора равна x_n, то предъявленные сети значения будут определяться вероятностным законом:

$$p(x_n) = 1 - \varepsilon,$$
$$p(-x_n) = \varepsilon,$$

где ε - малое положительное число, характеризующее уровень шума.

Если такие данные будут предъявлены *APT*, то будет наблюдаться деградация и размножение классов. Если сетью сформировано правильное *ядро* для класса, к которому

относится *вектор* x , то как только компонента x_n примет нулевое *значение* за счет шума (если векторы предъявляются не однократно), соответствующая компонента ядра также будет обнулена. Т.к. случайное нулевое *значение* может принять любая компонента x , то с течением времени все компоненты ядра будут обнулены, запомненная *информация* об этом классе - утрачена. Если после этого предъявить незашумленный вариант вектора x , то для него будет выделен новый *нейрон*, т.е. сформирован новый *класс*. Это явление называется размножением классов. Через некоторое время в сети будет множество нейронов с нулевыми весами, и все нейроны будут распределены. Работа сети прекратится. Это явление определяется исходной асимметрией алгоритмов *APT* относительно значений 0 и 1. Существуют методы для устранения асимметрии и предотвращения размножения классов.

Контрольные вопросы ЛР (ПК-4):

1. Виды нейронных сетей и способы организации их функционирования.
2. Виды сетей. Функционирование сетей.
3. Настройка нейронных сетей для решения задач.
4. Интерпретация ответов сети.
5. Оценка способности сети решить задачу.
6. Цели и задачи предпроектной стадии создания ИС.
7. Модели деятельности организации ("как есть" и "как должно быть").
8. Типовое проектирование ИС. Понятие типового проекта, предпосылки типизации.
9. Объекты типизации. Методы типового проектирования
10. Оценка эффективности использования типовых решений.
11. Типовое проектное решение (ТПР).
12. Классы и структура ТПР.
13. Основные понятия организационного бизнес-моделирования.
14. Миссия компании, дерево целей и стратегии их достижения.
15. Статическое описание компании: бизнес-потенциал компании.
16. Функционал компании.

Лабораторная работа №3
Задача нелинейного разделения двух классов. Метод максимума правдоподобия. нейрофизиологическая аналогия. Реализация булевых функций нейронными сетями. Выделение выпуклых областей сдача теста № 3..

Значимость параметров и сигналов. Сокращение описания (контрастирование) сетей.

Сокращение *множества* параметров и входных сигналов обученной нейронной сети преследует цели:

1. упрощение специализированных устройств;
2. сокращение объема используемой памяти и увеличение быстродействия;
3. удешевление сбора данных;
4. обеспечение (или облегчение) интерпретации результатов обработки данных.

Существует два способа сокращения (редукции) описания:

- редукция "снизу вверх" - постепенное удаление параметров от наименее значимых к более значимым;
- редукция "сверху вниз" - выделение наиболее значимых параметров и постепенное дополнение их менее значимыми.

Способ редукции "снизу вверх":

1. определяются наименее значимые параметры и устраняются вместе с соответствующими элементами системы;
2. оставшиеся параметры модифицируются так, чтобы наилучшим способом решить задачу;
3. циклически повторять пп. 1-2 до тех пор, пока задача не будет решаться с удовлетворительной точностью.

Определение значимости параметров на основании функции оценки

Есть набор x^i, $i = 1, \ldots, n$ размерности N, M - мерный *вектор* параметров w и *функция* оценки $H(x, w)$, оценивающая работу системы с параметрами w на векторе x (например, *расстояние* от вектора выходных сигналов системы до нужного ответа или до *множества* правильно интерпретируемых ответов). Требуется выделить наименее значимые параметры $w_k, k \in \{1, \ldots, M\}$ и компоненты данных x_j и модифицировать систему, отбрасывая наименее значимые параметры. Процедура отбрасывания неоднозначна. Простейший вариант - обращение в ноль - не всегда лучший: он не учитывает корреляции между данными. Учитывая корреляцию, следует отбрасываемые компоненты заменять на функции остающихся *компонент*.

Пусть для каждого w_k определено фиксированное *значение* w_k^0. Отбрасывание j -ой компоненты для i -го примера означает приравнивание $x_j := x_j^0$. В качестве простейшего варианта примем $w_k^0 = 0$ и для любого i полагаем

$$x_j^0 = (1/n)\sum_{p=1}^{n} x_j^p$$

(параметры обращаются в ноль, данные заменяются средним *по* выборке). Более тонкие методы предполагают замену отбрасываемых параметров и сигналов на некоторые функции оставшихся.

Показатели значимости вычисляются в два этапа: сначала они оцениваются для одного вектора (примера), потом для всей выборки.

1. Для данного x^p значимости w_k и x_j оцениваются как

$$\chi(w_k|x^p) = |\partial H(x^p, w)/\partial w_k| \times |w_k - w_k^0|,$$
$$\chi(x_j^p|x^p) = |\partial H(x^p, w)/\partial x_j^p| \times |x_j^p - x_j^{p0}|.$$

Здесь χ - вычисленные в линейном приближении абсолютные величины изменения H при сокращении описания. Оценка на всей выборке $x^p, p = 1, \ldots, n$ может проводиться *по*-разному. Например, может использоваться одна из следующих норм:

1. Сумма модулей:

$$\chi(w_k) = \sum_p \chi(w_k|x^p),$$
$$\chi(x_j) = \sum_p \chi(x_j|x^p).$$

2. *Максимум* модуля

$$\chi(w_k) = \max_p \chi(w_k|x^p),$$
$$\chi(x_j) = \max_p \chi(x_j|x^p).$$

Часто приходится иметь дело с системой, которая меняет свои параметры (например, в ходе обучения). Тогда к моменту принятия решения о значимости может быть накоплена *информация* о частных производных H в разных точках $w \in \{w_1, \ldots, w_q\}$. Ее можно использовать следующим образом.

Обозначим угловыми скобками процедуру усреднения *по* множеству параметров $\{w^1, \ldots, w^q\}$:

$$< f(w, \ldots) >= (1/q)\sum_{i=1}^{q} f(w^s, \ldots)$$

положим

$$\chi(w_k|x^p) =< |\partial H(x,w)/\partial w_{k|\chi=\chi^p} > \cdot w_k - |w_k^0|,$$
$$\chi(x_j|x^p) =< |\partial H(x,w)/\partial x_{j|\chi=\chi^p} > \cdot x_j^p - |x_j^{p0}|.$$

Усредняются абсолютные значения производных, а приращения берутся в тех точках, в которых будет проводиться процедура сокращения описания. Усреднение параметров w *по* нескольким значениям важно для нелинейных систем, в которых *производные* H могут сильно меняться от точки к точке.

Главная задача при сокращении описания - сохранить качество работы системы, оцениваемое с помощью H. Для этого требуется знать назначение системы и иметь способ оценки ее соответствия своему назначению.

Возможен другой подход, не предполагающий никакого знания о способах оценки. Ставится задача сохранить описание, минимально изменяя функционирование системы. В этом случае роль оценки играет изменение выходного сигнала системы после сокращения.

Определение значимости параметров по изменению выходных сигналов системы

Значимость параметров определяется практически так же, как и с помощью функции оценки. Пусть x - *вектор* данных, а w - *вектор* параметров. Пусть задан набор векторов $\{x^p\}$, на которых будет оцениваться функционирование системы, и определены значения x_j^{p0}, w_k^0 (в простейшем случае $w_k^0 = 0, x_j^{p0} = (1/n)\sum_{p=1}^{n} x_j^p$).

Вычислим в линейном приближении изменение вектора F при обращении w_k в w_k^0 и x_j^p в x_j^{p0}:

$$\Delta F = \partial F(x,w)/\partial w_{k|\chi=\chi^p} \times (w_k^0 - w_k),$$
$$\Delta F = \partial F(x,w)/\partial x_{j|\chi=\chi^p} \times (x_j^p - x_j^{p0}).$$

Пусть в пространстве выходных сигналов системы задана некоторая *норма* (например, евклидова). Тогда положим:

$$\chi(w_k|x^p) = \|\partial F(x,w)/\partial w_k\|_{\chi=\chi^p} \times |w_k^0 - w_k|,$$
$$\chi(x_j|x^p) = \|\partial F(x,w)/\partial x_j\|_{\chi=\chi^p} \times |x_j^p - x_j^{p0}|.$$

Таким образом, для каждого w_k и любого x_j определен *вектор* показателей значимости. *Координаты* вектора соответствуют точкам x^p. Теперь нужно вычислить норму этого вектора и объявить ее показателем значимости (можно в качестве нормы взять *максимум* модуля или сумму модулей). При использовании евклидовой нормы в пространстве выходных сигналов бывает удобно и далее выбирать такую же норму, полагая:

$$\chi(w_k) = (\sum_p \chi(w_k|x^p)^2)^{1/2},$$
$$\chi(x_j) = (\sum_p \chi(x_j|x^p)^2)^{1/2}.$$

Подход к определению значимости через изменение выходного сигнала не имеет альтернатив в том случае, когда рассматриваемая система является лишь подсистемой в некоторой системе (например, *сумматор* или *нейрон* в нейронной сети). Тогда при изменении параметров этой подсистемы приходится ограничиться требованием: выходной сигнал подсистемы должен изменяться как можно меньше, чтобы не нарушать функционирование всей системы.

Сокращение числа выходов в адаптивном линейном сумматоре (путь "снизу вверх")

Рассмотрим адаптивный линейный *сумматор*, вычисляющий линейную функцию $F(x, w) = w_0 + (x, w)$.

Решим задачу о сокращении числа выходных сигналов. Рассмотрим *определение* значимости *по* изменению выходного сигнала. Заметим, что:

$$\partial F/\partial w_0 = 1, \quad \partial F/\partial w_{i|\chi=\chi^p} = x_i^p, \quad \partial F/\partial w_i|_{\chi=\chi^p} = w_i, i = 1, \ldots, N.$$

Уничтожить i -й выходной сигнал можно двумя способами:

- заменой параметра w_i на 0;
- заменой x_i на постоянную величину не зависящую от p.

В последнем случае получаем новую функцию

$$F_{1i} = w_0 + x_i^0 w_i + \sum_{j=1, j\neq i}^{N} x_j w_j$$

Такое преобразование означает, что одновременно с уничтожением i -й выходной связи w_0 приобретает новое *значение*:

$$w_0 := w_0 + x_i^0 w_i.$$

При этом можно добиться меньшего изменения $F(x, w)$, чем просто при приравнивании w_i к нулю. Поэтому остановимся на замене i -го выходного сигнала на постоянную величину x_i^0. *Значение* этой постоянной определим исходя из минимизации изменения $F(x^p, w)$. *Минимизация* этого изменения, вычисленного в евклидовой норме, дает:

$$x_i^0 = (1/n) \sum_{p=1}^{n} x_i^p$$

Таким образом, оптимальной является замена x_i на его среднее *значение по* исходной выборке. В обозначениях теории вероятностей:

$$x_i^0 = M(x_i), \chi(x_i) = n^{1/2} w_i \sigma(x_i).$$

где $\sigma(x_i)$ - *среднеквадратичное отклонение* от x_i^0 на выборке $\{x^p\}$.

Значимость замены оценивается как

$$\chi(x_i) = |w_i| \sigma(x_i).$$

При исключении сигналов *по* одному, они сортируются в соответствии со значениями $\chi(x_i)$ и отбрасываются (заменяются средним) сначала те, что соответствуют меньшим $\chi(x_i)$. Заметим, что поэтому *путь* "снизу вверх" универсален, но не оптимален. В частности, для сумматоров и других элементов, линейных *по* параметрам (например, квадратичных сумматоров), существует учитывающий все корреляции *путь* исключения "сверху вниз" с ортогонализацией. Далее ограничимся оценкой значимости *по* изменению выходного сигнала.

Показатели значимости для нейрона с дифференцируемым нелинейным элементом

Эти показатели ищутся почти так же как для сумматора. Пусть

$$F(x, w) = \varphi(w_0 + (x, w))$$

тогда

$$\partial F / \partial x_{i|x=x^p} = \varphi'(w_0 + (x^p, w)) \cdot w_i.$$

В евклидовой норме (что соответствует методу наименьших квадратов) получаем:

$$\chi(x_i) = |w_i| [\sum_p (\varphi'(w_0 + (x^p, w)))^2 (x_i^p - M(x_i))^2]^{1/2},$$

т.е. *произведение* модуля параметра w_i на среднеквадратичное отклонение с весами. Роль веса играет квадрат производной функции в точке $w_0 + (x^p, w)$.

Показатели значимости для нейрона с пороговым нелинейным элементом (персептрона)

Эти показатели требуют для своего вычисления еще одного эвристического хода, т.к. прямо воспользоваться предыдущими формулами невозможно. Пусть *функция*, вычисляемая нейроном, имеет вид

$$F(x,w) = h(w_0 + (x,w)),$$
$$h(x) = \{1, x > 0,$$
$$0, x \leqslant 0\}$$

Для каждого вектора данных x^p необходимо оценить *значимость* изменения аргумента функции h при замене x_i^p на x_i^0. *Значение* $h(a)$ меняется только тогда, когда a меняет знак, поэтому естественно оценивать *значимость* изменения Δa переменной a в масштабе, определенном текущем значением переменной, т.е. *значимость* Δa оценивается, как $|\Delta a/a|$. Из этого эвристического рассуждения получаем:

$$(x_i|x^p) = |w_i| \cdot |x_i^p - x_i^0|/|w_0 + (x^p, w)|.$$

Если положить $x_i^0 = M(x_i)$ и воспользоваться евклидовой мерой, то вновь получим *произведение* модуля параметров w_i на среднеквадратичное отклонение с весом. В качестве весов выступают квадраты величин, обратных выходным сигналам сумматоров:

$$\chi(x_i) = |w_i| \cdot [\sum_p (x_i^p - M(x_i))^2/(w_0 + (x^p, w))^2]^{1/2}.$$

Полученные *выражение* для показателей значимости позволяют уп\-рощать основные элементы НС "снизу вверх", начиная с исключения самых малозначимых параметров.

Сокращение описания "сверху вниз" - набор достаточного семейства наиболее значимых параметров

Метод исключения параметров "сверху вниз" с ортогонализацией применим не ко всяким функциям $F(x,w)$, а только к таким, которые имеют вид:

$$F(x,w) = \varphi(\sum_i w_i f_i(x)).$$

Достоинство метода - автоматический учет корреляции между $f_i(x)$. Рассмотрим устройства, вычисляющие функции

$$F(x,w) = \sum_i w_i f_i(x).$$

К ним относятся линейные сумматоры, квадратичные сумматоры и др.

Пусть заданы векторы данных:

$$x^p = (x_1^p, \ldots, x_i^p, \ldots, x_N^p), p = 1, \ldots, n, i = 1, \ldots, N.$$

Поставим задачу сокращения описания следующим образом: так определить некоторое наименьшее возможное множество индексов J и набор чисел $\beta_j, j \in J$, чтобы *норма* отклонения $\|\Delta F\| = \|F - F'\|$, где $F' = \sum_{j \in J} \beta_j f_j(x)$, не превышала некоторой наперед заданной величины ε. Все функции рассматриваются на конечном множестве $\{x^p\}$. Для любой функции $\varphi(x)$ евклидова *норма*:

$$\|\varphi\| = [\sum_p \varphi^2(x^p)]^{1/2}.$$

С каждой функцией $f(x)$ связан n -мерный *вектор* f с компонентами $f^p = f(x^p)$. *Вектор* F с координатами $F^p = \sum_i w_i f_i(x^p)$ является линейной комбинацией векторов f_i с координатами $f_i^p = f_i^p(x^p)$. Линейную оболочку семейства векторов f_i обозначим $L = L(\{f_i\})$. Построим в пространстве L ортонормированный *базис* с помощью последовательной ортогонализации векторов f_i. Каждый следующий шаг ортогонализации выполним так, чтобы величина проекции F на новый *вектор* базиса была максимальной из возможных. Процесс ортогонализации продолжим, пока $\|F\|^2 - \|F_\perp\|^2 > \xi^2$, где $F_\perp$ - *проекция* F на построенную ортогональную систему. *По* окончании процесса полагаем $F' = F_\perp$.

Опишем вычисления более детально.

1. Проводим нормировку: для любого i полагаем $g_i^0 = f_i / \|f_i\|$.

2. Вычисляем $|(F, g_i^0)|$ (*модуль* проекции вектора F на *вектор* g_i^0), $i = 1, \ldots, N$. Находим среди этих чисел максимальное (пусть его номер $i_{0,max}$), полагаем $e_1 = g_{i0max}^0$, исключаем g_{i0max}^0 из *множества* $\{g_i^0\}$, получаем $g_i^1 = g_i^0 - (g_i^0, e_1)e_1$, исключаем из *множества* $\{g_i^1\}$ нулевые векторы, если таковые существуют, проводим нормировку $g_i^1 = g_i^1 / \|g_i^1\|$.

3. Пусть определены векторы $e_1, \ldots, e_l$ и не более чем $n - l$ нормированных векторов g_i^l. Среди векторов g_i^l ищем такой $g_{i,max}^l$, для которого $|(F, g_i^l)|$ принимает максимальное *значение*, полагаем $e_{l+1} = g_{i,max}^l$, исключаем $g_{i,max}^l$ из *множества* векторов $\{g_i^l\}$; полагаем $g_i^{l+1} = g_i^l - (g_i^l, e_{l+1})e_{l+1}$, исключаем из этого *множества* нулевые векторы, если таковые существуют, нормируем: $g_i^{l+1} = g_i^{l+1} / \|g_i^{l+1}\|$.

Вычисления проводим, пока $\|F\|^2 - \sum_{i=1,l}(F, e_i)^2 > \varepsilon^2$.

После завершения вычислений имеем набор ортонормированных векторов $e_1, \ldots, e_l, l \leqslant n$. Они являются линейными комбинациями векторов $f_{i,1max}, \ldots, f_{i,lmax}$. *Коэффициенты* разложения e_j *по* набору $f_{i,1max}, \ldots, f_{i,lmax}$ могут быть вычислены и сохранены в ходе ортогонализации. Полагаем $J = \{i_{1max}, \ldots, i_{lmax}\}$. Тогда $F' = \sum_{j=1,l}(F, e_j)e_j = \sum_{j \in J} \beta_j f_j$. Это и есть решение задачи. Числа β_j выражаются через *коэффициенты* разложения векторов $e_i, i = 1, \ldots, l$ *по* $f_j, j \in J$ и скалярные произведения (F, e_j): если $e_i = \sum_{j \in J} q_{ij} f_j$, то $\beta_j = \sum_{i=1,l}(F, e_i)q_{ij}$.

Разложение e_i *по* $f_j, j \in J$ имеет рекурсивную форму:

$$e_1 = f_{i,1max}/\|f_{i,1max}\|,$$
$$e_2 = [f_{i,2max} - (f_{i,2max}, e_1)e1]/\|f_{i,2max} - (f_{i,2max}, e_1)e_1\|,$$
$$\ldots,$$
$$e_j = [f_{i,jmax} - \sum_{r=1,j-1}(f_{i,jmax}, e_r)e_r]/\|f_{i,jmax} - \sum_{r=1,j-1}(f_{i,jmax}, e_r)e_r\|,$$
$$\ldots$$

Для функций вида $F(x, w) = \varphi(\sum_i w_i f_i(x))$ с дифференцируемой функцией процедура аналогична с точностью до замены скалярного произведения: используется скалярное *произведение* с весами $(f, g) = \sum_p V_p f^p g^p$, где $V_p = ((\sum_i w_i f_i(x^p)))^2$. В этом скалярном произведении вычисляются все нормы и проводится ортогонализация.

Для функций с пороговой нелинейностью на выходе используем скалярное *произведение* с весами $V_p = |\sum_i w_i f_i(x^p)|^2$.

Описанная процедура сокращения "сверху вниз" с ортогонализацией особенно важна для упрощения элементов сложных сетей, в структуре которых и *вектор* входных сигналов элемента может быть далек от исходных данных, и его выходной сигнал далек от оцениваемого выхода всей сложной системы.

Процедуры анализа значимости и сокращения описания выделяют наиболее важные параметры и связи в НС. *По* аналогии с обработкой изображения их называют процедурами контрастирования или редукции.

Роль контрастирования (редукции) не сводится только к сокращению описания: более общая задача - привести параметры системы к выделенному набору значений, в частности, уменьшить *разрядность*, что важно для удешевления специализированных устройств, экономии памяти и т.д.

Рекурсивное контрастирование и бинаризация

Рекурсивное контрастирование состоит в модификации параметров системы - одного за другим. Для этого параметры должны быть как-то линейно упорядочены $w_1, \dots, w_N$. При модификации w_i используются модифицированные значения $w_1, \dots, w_{i-1}$ и немодифицированные $w_{i+1}, \dots, w_N$.

Пусть для сумматора задана *обучающая выборка* входных векторов $x_1, \dots, x_n$ и соответствующих выходных сигналов $f_1, \dots, f_n$, а также известны значения параметров, которые реализуют *сумматор*: $f_i = w_0 + (x^i, w)$. Требуется произвести бинаризацию сумматора, т.е. найти числа a, b и *вектор* β с координатами 0 или 1, чтобы значения функции $\varphi(x) = a + b(x, \beta)$ на выборке $\{x^i\}$ как можно меньше отличались от f_i. Критерием такого отличия будем считать $H = \sum_{i=1,n}(f_i - \varphi(x^i))^2$. Построим *координаты* вектора β *по* порядку $\beta_1, \beta_2, \dots$.

Пусть построены $\beta_1, \dots, \beta_{i-1}$. Обозначим $\beta^{0i} = (\beta_1, \dots, \beta_{i-1}, 0, \dots, 0)$ (последние $N - i + 1$ координат - нули), $\beta^{1i} = (\beta_1, \dots, \beta_{i-1}, 1, 0, \dots, 0)$ (последние $N - i$ координат - нули), $\alpha^i = (0, \dots, 0, \alpha_{i+1}, \dots)$ (первые i координат - нули).

Введем функции:

$$\varphi_0^i(x) = a_{0i} + b_{0i}(x, \beta^{0i}) + (x, \alpha^i),$$
$$\varphi_1^i(x) = a_{1i} + b_{1i}(x, \beta^{1i}) + (x, \alpha^i),$$
$$H_{0i} = \sum_{j=1,n} (f_j - \varphi_0^i(x^j))^2,$$
$$H_{1i} = \sum_{j=1,n} (f_j - \varphi_1^i(x^j))^2.$$

Определим параметры $a_{0i}, b_{0i}, a_{1i}, b_{1i}$ из условий $H_{0i} \to min, H_{1i} \to min$, минимизируя функции H_{0i} и H_{1i}. Пусть $h_{0i} = minH_{0i}$ и $h_{1i} = minH_{1i}$. Если $h_{1i} \geqslant h_{0i}$, то полагаем $\beta_i = 0$, в противном случае $\beta_i = 1$.

После того как построены все $\beta_i, i = 1, \dots, N$ (N - *размерность* вектора данных), автоматически определяются a и b: если $N = 0$, то полагаем $a = a_{0N}, b = b_{0N}$, иначе $a = a_{1N}, b = b_{1N}$.

Если бинаризация проведена, а необходимая *точность* не достигнута, то можно построить второй бинаризованный *сумматор*, корректирующий ошибку первого --- так, чтобы в сумме они хорошо аппроксимировали работу исходного сумматора на элементах обучающей выборки. В

описанной процедуре делаем замену $f_i := f_i - \varphi(x^i)$ и для этих исходных данных вновь строим бинаризованный *сумматор по* алгоритму рекурсивной бинаризации. Повторяем такое построение, пока не будет достигнута удовлетворительная *точность*. В результате получим набор бинаризованных сумматоров, которые в совокупности (т.е. в результате суммирования выходных сигналов) достаточно точно аппроксимируют исходный. При появлении весов, определяющих *значимость* отдельных примеров из обучающей выборки, рекурсивная бинаризация проводится точно так же, только в функциях H появляются веса.

Если требуется тем же путем упростить любой другой элемент, линейный *по* параметрам, $F(x,w) = \sum_i w_i f_i(x)$, то вместо обучающей выборки $\{x^j\}$ берем семейство векторов $\{y^j\}$ с координатами $y_i^j = f_i(x^j)$. После такого $y_i^j = f_i(x^j)$ преобразования рассматриваемый элемент превращается в обычный *сумматор*, для которого последовательность действий уже описана.

Контрольные вопросы ЛР (ПК-4):

1. Метод максимума правдоподобия.
2. Нейрофизиологическая аналогия.
3. Реализация булевых функций нейронными сетями.
4. Выделение выпуклых областей.
5. Особенности задачи оптимизации, возникающей при обучении нейронных сетей.
6. Учет ограничений при обучении.
7. Выбор направления минимизации.
8. Партан-методы.
9. Одношаговый квазиньютоновский метод и сопряженные градиенты.
10. Комплексная схема нечеткого планирования.
11. Процессный подход к организации деятельности организации.
12. Связь концепции процессного подхода с концепцией матричной организации.
13. Основные элементы процессного подхода.
14. Границы процесса, ключевые роли, дерево целей, дерево функций, дерево показателей.
15. Выделение и классификация процессов.

Лабораторная работа №4.
Градиентные алгоритмы обучения сети. Универсальный путь обучения. Особенности задачи оптимизации, возникающей при обучении нейронных сетей. Учет ограничений при обучении. Выбор направления минимизации. партан-методы. Одношаговый квазиньютоновский метод и сопряженные градиенты. Сдача теста № 4.

Электронная реализация нейронных сетей

В качестве единицы производительности нейросетей принято "число соединений в секунду" - *CPS* (connections *per second*). Под соединением здесь понимается *умножение* входного сигнала на весовой коэффициент и *сложение* с накопленной суммой.

Анализ нейросетевых алгоритмов позволяет сделать следующие выводы:

1. При решении плохо формализованных задач моделирования, прогнозирования и распознавания, которые обычно сводятся к конструированию областей многомерного пространства, достаточно малоразрядных представлений входов и весов и операций с фиксированной точкой. Это обусловлено тем, что входные сигналы нормируются и количество их значений невелико.
2. При решении хорошо формализованных задач (например, задач комбинаторной оптимизации) существенна точность вычислений, что требует полноразрядных представлений чисел и операций с плавающей точкой.

Электронные нейронные сети обычно используются в качестве *акселераторов* для персональных ЭВМ при решении соответствующих классов задач (обработки сигналов и изображений, распознавания образов и т.п.).

Нейрочипы

Нейрочипы подразделяются на цифровые, аналоговые и гибридные. Они могут включать в себя схемы настройки весов при обучении или предусматривать внешнюю загрузку весов. Наибольшую проблему при создании нейрочипов представляют схемы умножения, так как именно они лимитируют скорость вычислений.

Аналоговые реализации используют простые физические эффекты для выполнения нейросетевых преобразований. Обеспечение заданной точности требует тщательного проектирования и изготовления.

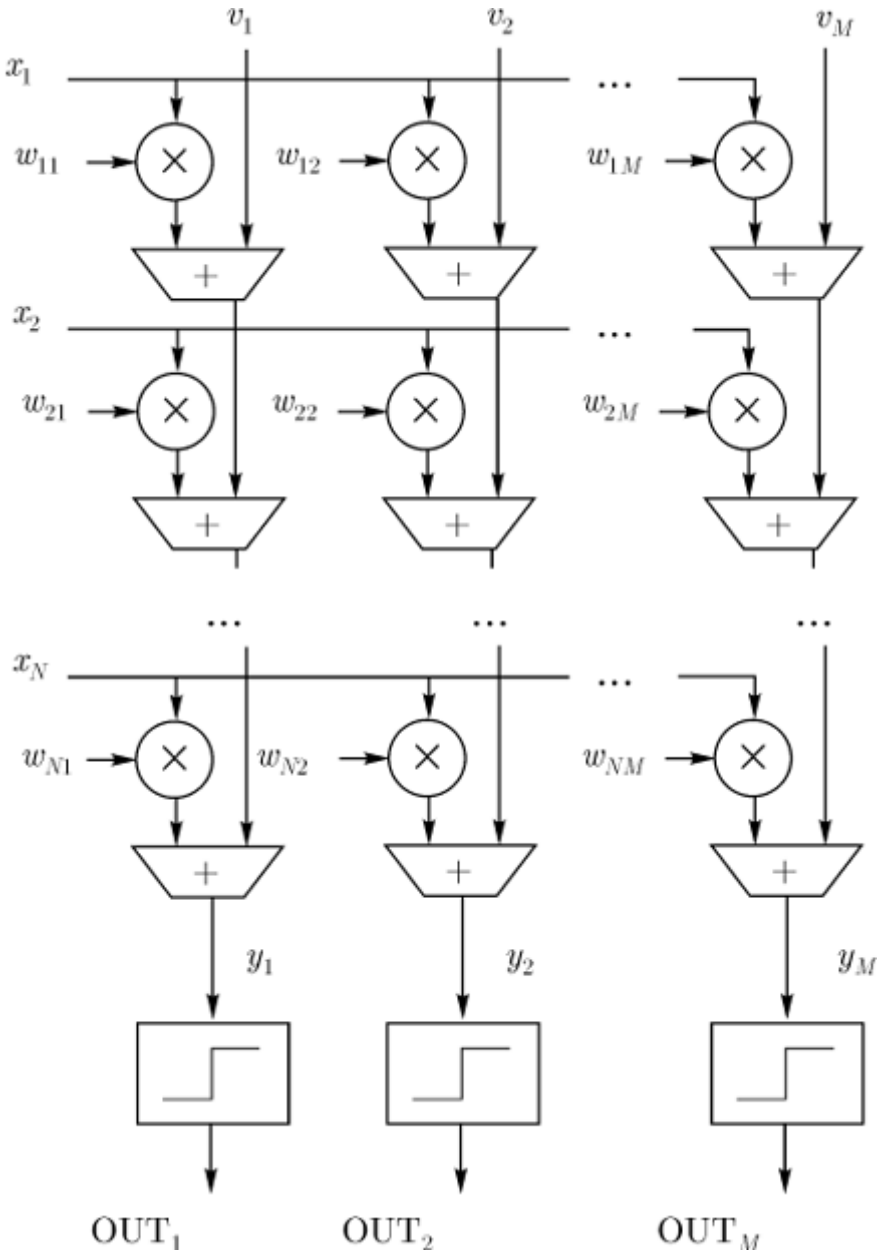
Гибридные нейрочипы используют комбинацию аналогового и цифрового подходов. Например, входы могут быть аналоговыми, веса могут загружаться как цифровые и выходы могут быть цифровыми. Существуют нейрочипы, в которых используется *представление данных* частотой или шириной импульсов.

Нейропроцессор NM6403

Отечественный нейропроцессор NM6403, разработанный в НПО "*Модуль*" (www.*module*.ru), имеет скалярный *процессор* (скалярное RISC-ядро) и *векторный процессор* для обработки двоичных векторов произвольной разрядности в пределах 1-64 битов. Скалярный *процессор* выполняет всю подготовку данных для работы векторного процессора.

Модель слоя нейронов, эмулируемого процессором NM6403, показана на рис. 1. В общем случае слой имеет N входов и состоит из M нейронов. M -й *нейрон* выполняет операцию умножения - накопления над N данными $x_1, \ldots, x_N$, поданными на соответствующие входы нейрона.

Нейропроцессор NM6403 имеет два встроенных линка, совместимых с линками сигнального микропроцессора TMS320C40. Кроме того, интерфейсы локальной и глобальной памятей позволяют без дополнительного оборудования подсоединять два нейропроцессора к общему блоку памяти.



увеличить изображение

Рис. 1. Модель слоя нейронов

Мультипроцессорная система (рис. 2) из K нейропроцессоров NM6403 эмулирует нейронную *сеть* в K раз быстрее, чем один нейропроцессор.

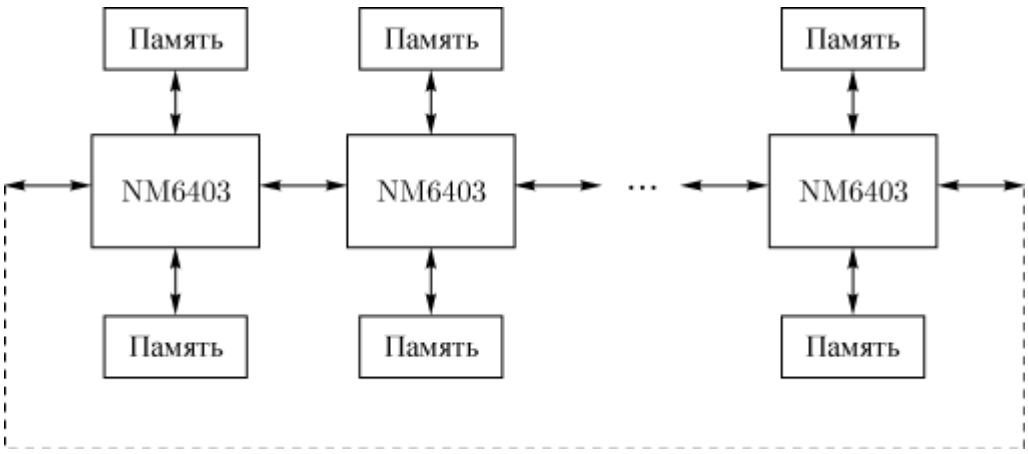


Рис. 2. Линейная (кольцевая) система из нейропроцессоров NM6403

Оптическая реализация нейронных сетей

Мощность нейронной сети определяется большим количеством связей: отдельные элементы имеют относительно малые вычислительные мощности. Обеспечение требуемой связности в электронных цепях остается серьезной проблемой, особенно при реализации нейронных сетей с полным графом соединений. Электронные интегральные цепи являются существенно планарными с рельефностью, обусловленной множеством слоев.

Проблему связей можно решить при использовании оптических систем для реализации НС. Взаимное соединение нейронов с помощью световых лучей не требует изоляции между сигнальными путями: световые потоки могут пересекаться, не влияя друг на друга, и сигнальные пути могут располагаться в трех измерениях. Плотность путей передачи ограничена только размерами источников и детекторов. Все сигнальные пути могут работать одновременно, тем самым обеспечивая огромную *скорость передачи* данных.

В оптических НС величины оптических весов могут запоминаться в голограммах с высокой степенью плотности (до 10^{12} *бит* на куб. см.). Веса могут модифицироваться в процессе работы сети.

К сожалению, возникает множество практических проблем при попытках оптической реализации нейронных сетей. Оптические устройства имеют собственные физические характеристики, часто не соответствующие требованиям искусственных нейронных сетей. Хотя они в действительности пригодны для обработки изображений, все же изображения от оптических нейронных сетей, полученные до настоящего времени, были разочаровывающе плохими. Однако достаточно взглянуть на первые пробы телевизионных передач, чтобы понять, какой огромный прогресс возможен в повышении качества изображения. Несмотря на эти трудности, а также на такие проблемы, как *стоимость*, размеры и критичность к ориентации, потенциальные возможности оптических систем побуждают попытки проведения интенсивных и широких исследований. В этой области происходят стремительные изменения, и в ближайшее время ожидаются важные улучшения.

Конфигурации оптических НС в основном подразделяются на две категории: векторно-матричные умножители и голографические корреляторы.

Векторно-матричные умножители

В качестве *матрицы весов* (рис. 3) используется фотопленка, у которой прозрачность каждого квадрата пропорциональна весу. *Выход* каждого фотодетектора является сверткой между входным вектором и соответствующим столбцом *матрицы весов*. *Умножение* выполняется параллельно. При использовании соответствующих высокоскоростных светодиодов и фотодетекторов *умножение* вектора на матрицу может быть выполнено менее, чем за наносекунду. Более того, скорость умножения практически не зависит от размерности массива. Это позволяет наращивать сети без существенного увеличения времени вычислений. Возможность менять веса основана на использовании жидкокристаллического клапана вместо фотографического негатива.

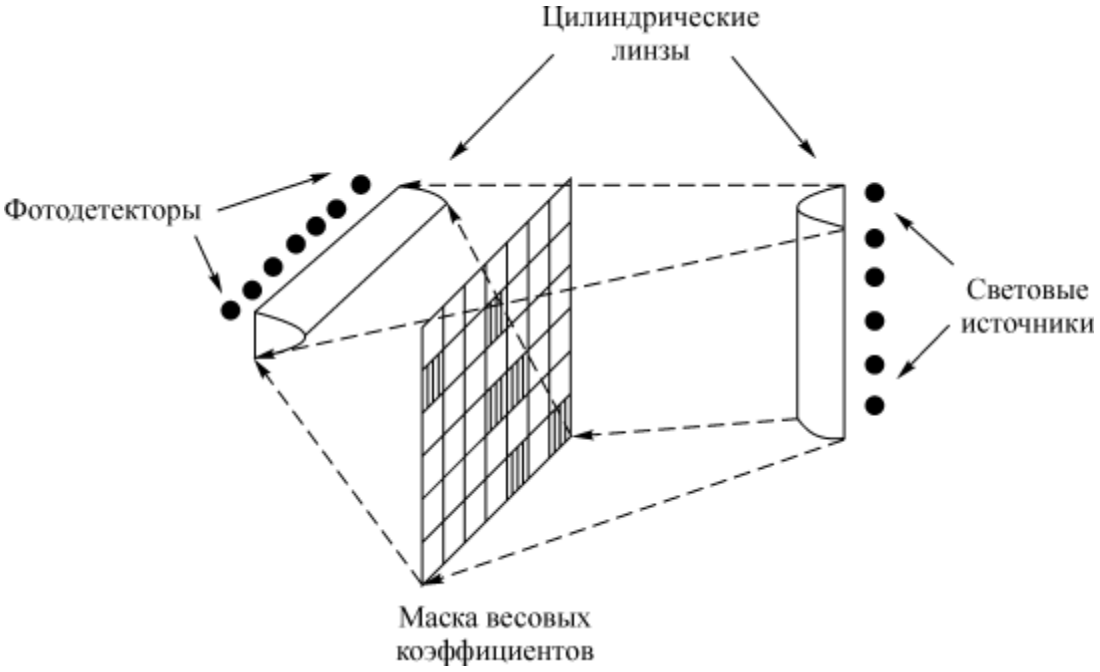


Рис. 3. Электронно-оптический векторно-матричный умножитель

Голографические корреляторы

В голографических корреляторах образцы изображений запоминаются в виде голограммы (плоской или объемной) и восстанавливаются при когерентном освещении в петле обратной связи.

Входное изображение (возможно, зашумленное или неполное) коррелируется оптически одновременно со всеми запомненными изображениями. Корреляции обрабатываются пороговой функцией и подаются на вход системы, где наиболее сильные корреляции усиливают (и, возможно, корректируют или завершают) входное изображение. Этот процесс повторяется многократно, и усиленное изображение при каждом проходе изменяется, пока система не стабилизируется на требуемом изображении.

Оптические нейронные сети предлагают огромные выгоды с точки зрения скорости и плотности внутренних связей. Они могут быть использованы (в той или иной форме) для реализации сетей фактически с любой архитектурой.

В настоящее время ограничения электронно-оптических устройств создают множество серьезных проблем, которые должны быть решены прежде, чем оптические нейронные сети получат широкое применение. Однако, учитывая, что большое количество превосходных исследователей работает над этой проблемой, а также большую поддержку со стороны военных, можно надеяться на быстрый прогресс в этой области.

Контрольные вопросы ЛР (ПК-4):

1. Особенности задачи оптимизации, возникающей при обучении нейронных сетей.
2. Учет ограничений при обучении.
3. Выбор направления минимизации.
4. Партан-методы.
5. Одношаговый квазиньютоновский метод и сопряженные градиенты.
6. Максимизация консенсуса.
7. Синхронное и асинхронное функционирование машины Больцмана.
8. Решение задачи коммивояжера машиной Больцмана.
9. Свойства, определяемые пользователем (UDP).
10. Диаграммы потоков данных (Data Flow Diagramming): работы.
11. Диаграммы потоков данных (Data Flow Diagramming): внешние сущности (ссылки.
12. Диаграммы потоков данных (Data Flow Diagramming): потоки работ.
13. Диаграммы потоков данных (Data Flow Diagramming): хранилища данных.
14. Внемашинное информационное обеспечение.
15. Основные понятия классификации информации.

Лабораторная работа №5. Рекуррентные сети как ассоциативные запоминающие устройства. Автоассоциативная сеть Хопфилда. Обучение сети Хопфилда по правилу Хебба. Сеть Хемминга. Двунаправленная ассоциативная память.

Аннотация: Рассматриваются: нейронная сеть Хопфилда как автоассоциативная память, обучаемая с использованием метода Хебба и проекционного метода; гетероассоциативная память на основе сети Хемминга и двунаправленная ассоциативная память.

Введение

Отдельную группу нейронных сетей составляют *сети с обратной связью* между различными слоями нейронов. Это так называемые рекуррентные сети. Их общая черта состоит в передаче сигналов с выходного либо скрытого слоя на *входной* слой.

Благодаря обратной связи при подаче сигнала на входы сети, в ней возникает переходный процесс, который завершается формированием нового устойчивого состояния, отличающегося в общем случае от предыдущего. Если функцию активации нейрона обозначить $f(u)$, где u - взвешенная сумма его возбуждений, то состояние нейрона можно определить выходным сигналом $y = f(u) = f(w_1x_1 + \ldots + w_Nx_N)$. Изменение состояния i -го нейрона можно описать системой дифференциальных уравнений

$$\tau_1(\partial u_i/\partial t) = w_{i1}f(u_1) + \ldots + w_{iN}f(u_N) - u_i - b_i$$

для $i = 1, 2, \ldots, N$, где b_i - пороговое *значение*.

Рекуррентной сети можно поставить в соответствие энергетическую функцию Ляпунова

$$E = -(1/2)\sum_j \sum_{i \neq j} w_{ij}y_iy_j + \sum_{i=1,N}(1/R_i)\int_0^x f_i^{-1}(y_i)dy_i + \sum_{i=1,N} b_iy_i.$$

Изменение состояния какого-либо нейрона инициализирует изменение энергетического состояния сети в направлении минимума ее энергии вплоть до его достижения. В пространстве состояний локальные энергетические минимумы E представлены точками стабильности, называемыми аттракторами из-за тяготения к ним ближайшего окружения. Благодаря наличию аттракторов, рекуррентные сети могут быть использованы как устройства ассоциативной памяти.

Ассоциативная память играет роль системы, определяющей взаимную зависимость векторов. В случае, когда на взаимозависимость исследуются компоненты одного и того же вектора, говорят об автоассоциативной памяти. Если же взаимозависимыми оказываются два различных вектора, можно говорить о памяти гетероассоциативного типа. К первому классу относится *сеть* Хопфилда, а ко второму - *сеть* Хемминга и *сеть* типа BAM (*Bidirectional Associative Memory - двунаправленная ассоциативная память*).

Задача ассоциативной памяти сводится к запоминанию обучающих векторов, чтобы при представлении нового вектора система могла сгенерировать ответ - какой из запомненных ранее векторов наиболее близок к вновь поступившему образу. Часто в качестве меры близости отдельных множеств применяется *расстояние* Хемминга.

При использовании двоичных значений (0,1) *расстояние* Хемминга между двумя векторами $y = (y_1, y_2, \ldots y_n)$ и $d = (d_1, d_2, \ldots, d_n)$ определяется в виде

$$d_H(y,d) = \sum_{i=1,n} (d_i(1-y_i) + (1-d_i)y_i)$$

При биполярных значениях элементов обоих векторов *расстояние* Хемминга рассчитывается по формуле

$$d_H(y,d) = (1/2)(n - \sum_{i=1,n} d_i y_i)$$

Мера Хемминга равна числу несовпадающих *компонент* двух векторов. Она равна нулю, когда $y = d$.

Автоассоциативная сеть Хопфилда

Структура *сети Хопфилда* представляется в виде системы с непосредственной обратной связью выхода со входом (рис. 1). Выходные сигналы нейронов являются одновременно входными сигналами сети: $x_i(k) = y_i(k-1)$. В классической *сети Хопфилда* отсутствует автосвязь (*связь* выхода нейрона с его собственным входом), что соответствует $w_{ii} = 0$, а *матрица* весов является симметричной: $W = W^T$. Отсутствие автосвязи и *симметричность матрицы весов* являются достаточными (но не необходимыми!) условиями сходимости итерационных (переходных) процессов в *сети Хопфилда*.

Далее в данной лекции предполагаем, что каждый *нейрон* имеет биполярную ступенчатую функцию активации со значениями ± 1. Это означает, что выходной сигнал i -го нейрона определяется функцией

$$y_i = \mathrm{sgn}(\sum_{j=0,N} w_{ij}x_j + b_i)$$

где N обозначает количество нейронов, $N = n$.

Далее допустим, что порог срабатывания является компонентой вектора x. Тогда основную зависимость, определяющую *сеть Хопфилда*, можно представить в виде

$$y_i(k) = \mathrm{sgn}(\sum_{j=0,N} w_{ij}y_j(k-1)) \qquad 1)$$

с начальным условием $y_j(0) = x_j$.

В процессе функционирования *сети Хопфилда* можно выделить два режима: обучения и классификации. В режиме обучения на основе известных векторов подбираются весовые *коэффициенты* сети. В режиме классификации при фиксированных значениях весов и вводе конкретного начального состояния нейронов возникает переходный процесс вида (1), завершающийся в одном из локальных минимумов, для которого $y(k) = y(k-1)$.

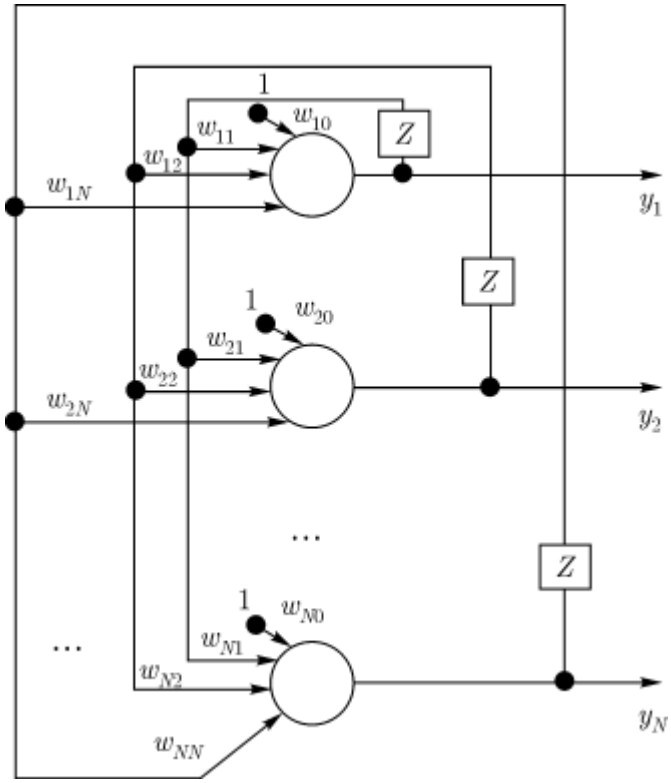


Рис. 1. Структура сети Хопфилда

Обучение сети Хопфилда по правилу Хебба

Для одного обучающего вектора x значения весов могут быть вычислены по правилу Хебба

$$w_{ij} = (1/N)x_i x_j,$$

поскольку тогда

$$(1/N)(\sum_{j=1}^{N} x_i x_j x_j) = x_i$$

(вследствие биполярных значений элементов вектора x всегда $x_j^2 = (\pm 1)^2 = 1$).

При вводе большего количества обучающих векторов $x(k), k = 1, 2, \ldots, p$ веса w_{ij} подбираются согласно обобщенному правилу Хебба

$$w_{ij} = (1/N) \sum_{k=0}^{p} x_i^{(k)} x_j^{(k)}.$$

Важным параметром ассоциативной памяти является ее емкость. Под емкостью понимается максимальное число запомненных образов, которые классифицируются с допустимой

погрешностью ε_{max}. Показано, что при использовании для обучения правила Хебба и при $\varepsilon_{max} = 0.01$ (1% компонентов образа отличается от нормального состояния) максимальная *емкость памяти* составит всего лишь около 13,8% от количества нейронов, образующих *ассоциативную память*. Столь малая емкость обусловлена тем, что *сеть* Хебба хорошо запоминает только взаимно ортогональные векторы или близкие к ним.

Обучение сети Хопфилда методом проекций

Лучшие результаты, чем при использовании правила Хебба, можно получить, если для обучения использовать псевдоинверсию. В основе этого подхода лежит предположение, что при правильно подобранных весах каждый поданный на вход сети *вектор* вызывает генерацию самого себя на выходе сети. В матричной форме это можно представить в виде

$$WX = X,$$

где W - *матрица весов* сети размерностью $N \times N$, а X - прямоугольная *матрица* размерностью $N \times p$, составленная из p обучающих векторов $x^{(i)}, i = 1, 2, \ldots, p$. Решение такой линейной системы уравнений имеет вид

$$W = XX^{+},$$

где знак + обозначает псевдоинверсию.

Если обучающие векторы линейно независимы, последнее *выражение* можно упростить и представить в виде

$$W = X(X^T X)^{-1} X^T. \qquad 2)$$

Здесь псевдоинверсия заменена обычной инверсией квадратной матрицы $X^T X$ размерностью $p \times p$.

Выражение (2) можно записать в итерационной форме, не требующей расчета обратной матрицы. В этом случае (2) принимает вид итерационной зависимости от последовательности обучающих векторов $x^{(i)}$, $i = 1, 2, \ldots p$:

$$y^{(i)} = (W^{(i-1)} - E)x^{(i)},$$
$$W^{(i)} = W^{(i-1)} - (y^{(i)} y^{(i)T}) / (y^{(i)T} y^{(i)})$$

при начальных условиях $W^{(0)} = 0$. В результате предъявления p векторов *матрица* весов сети принимает *значение* $W = W^{(p)}$. Описанный здесь метод называется методом проекций. Применение его увеличивает максимальную емкость сети Хопфилда до $N - 1$. Увеличение емкости обусловлено тем, что в методе проекций требование ортогональности векторов заменено гораздо менее жестким требованием их линейной независимости.

Модифицированный вариант метода проекций - метод Δ -проекций — градиентная форма алгоритма минимизации. В соответствии с этим методом веса подбираются с помощью процедуры, многократно повторяемой на всем множестве обучающих векторов:

$$W \leftarrow W + (h/N)(x^{(i)} - Wx^{(i)})x^{(i)T}, h \in (0.7, 0.9).$$

Обучающие векторы предъявляются многократно вплоть до стабилизации значений весов.

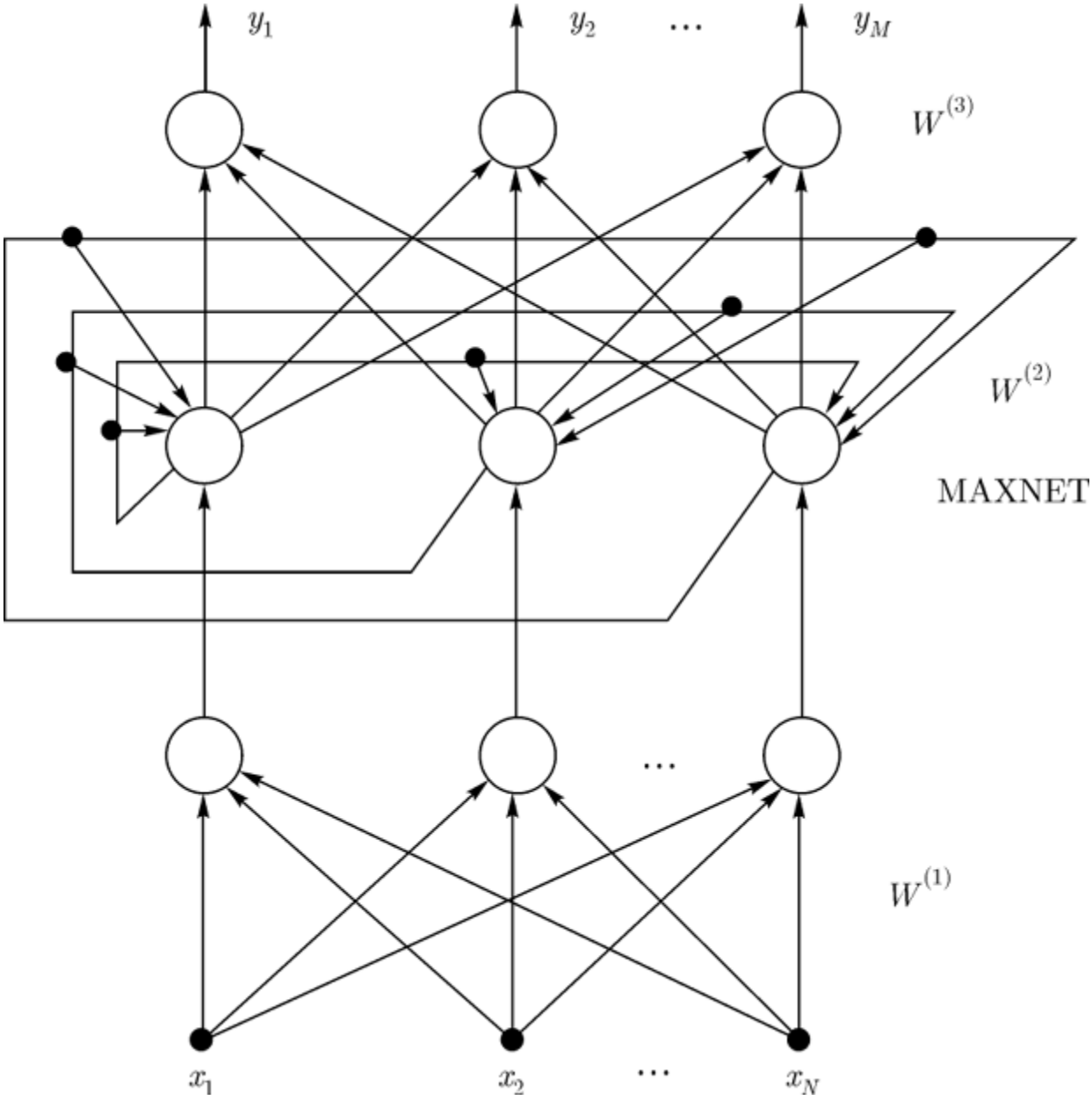
Сеть Хемминга

Сеть Хемминга включает в себя три слоя (рис.2).

Первый слой имеет однонаправленное распространение сигналов от входа к выходу и фиксированные значения весов.

Второй слой состоит из нейронов, связанных обратными связями по принципу "каждый с каждым", при этом в каждом нейроне слоя существует автосвязь (*связь* входа нейрона со своим собственным выходом). Разные нейроны в слое связаны отрицательной (тормозящей) обратной связью с весом $-\varepsilon$, при этом величина ε обычно обратно пропорциональна количеству образов. С собственным входом *нейрон* связан положительной (возбуждающей) обратной связью с весом, равным +1. Пороговые веса нейронов приняты равными нулю. Нейроны этого слоя функционируют в режиме WTA, при котором в каждой фиксированной ситуации активизируется только один *нейрон*, а остальные пребывают в состоянии покоя.

Выходной однонаправленный слой формирует выходной *вектор*, соответствующий входному вектору.



увеличить изображение

Рис. 2. Структура сети Хемминга

Сеть Хемминга считается гетероассоциативным запоминающим устройством с парой связанных между собой векторов (x, y), где x и y - *входной* и выходной биполярные векторы сети.

Веса первого слоя соответствуют векторам $x_i, i = 1, \ldots, p$, т.е.

$$w_{ij}^{(1)} = x_{ij}.$$

Аналогично, веса выходного слоя соответствуют векторам образов y_i, связанных с x_i:

$$w_{ij}^{(3)} = y_{ij}.$$

Во втором слое (MAXNET), функционирующем в режиме *WTA* (Winner Takes *ALL* - "Победитель забирает все"), каждый *нейрон* должен усиливать собственный сигнал и ослаблять сигналы остальных нейронов. Для этого принимается

$$w_{ij}^{(2)} = 1,$$

а также

$$-1/(p-1) < w_{ij}^{(2)} < 0, \quad i \neq j.$$

Для обеспечения сходимости итерационного процесса во втором слое веса

$$w_{ij}^{(2)} = -1/(p-1) + \xi,$$

где ξ - достаточно малая *случайная величина*, $|\xi| \ll 1/(p-1)$.

Нейроны первого слоя рассчитывают расстояния Хемминга $d_H(x,y)$ между поданным на вход сети вектором x и векторами весов $w_i = x_i, i = 1, 2, \ldots, p$ нейронов этого слоя. Значения выходных сигналов нейронов первого слоя определяются по формуле

$$y_i^{(1)} = 1 - d_H(x,y)/N,$$

где N - число *компонент* вектора x.

Сигналы $y_i^{(1)}$ становятся начальными состояниями нейронов второго слоя. Этот слой определяет "победителя", т.е. *нейрон*, выходной сигнал которого близок к 1. Такой *нейрон* указывает на *вектор* образа с минимальным расстоянием Хемминга до входного вектора x. *Функция* активации для нейронов второго слоя задается выражением

$$f(y) = \begin{cases} y, & \text{если } y > 0, \\ 0, & \text{если } y < 0. \end{cases}$$

Итерационный процесс во втором слое завершается, когда активным остается только один *нейрон* (победитель), тогда как остальные нейроны пребывают в нулевом состоянии. Победитель через веса $w_{ij}^{(3)}$ линейных нейронов выходного слоя представляет *вектор* y_i, который соответствует вектору x_i, признанному вторым слоем ближайшим к входному вектору x.

Достоинством сети Хемминга считается небольшое количество взвешенных связей между нейронами. Многочисленные эксперименты доказали, что *сеть* Хемминга дает лучшие результаты, чем *сеть Хопфилда*. Единственная проблема, связанная с сетью Хемминга, проявляется в случае, когда зашумленные образы находятся на одинаковом (в смысле Хемминга) расстоянии от двух или более эталонов. В этом случае выбор сетью Хемминга одного из эталонов становится случайным.

Двунаправленная ассоциативная память

Обобщением *сети Хопфилда* на случай двухслойной рекуррентной структуры, позволяющей кодировать *множества* двух взаимосвязанных векторов, считается двунаправленное ассоциативное *запоминающее устройство*, называемое BAM (*Bidirectional Associative Memory*) (рис. 3). Сигналы распространяются в двух направлениях. Если в первом цикле сигналы вначале проходят в одну сторону для задания состояний нейронов-получателей, то в следующем цикле эти нейроны сами становятся источниками, высылающими сигналы в обратную сторону. Процесс повторяется до достижения состояния равновесия.

Функция активации нейронов имеет пороговый характер. Для обеспечения лучших характеристик сети на этапе обучения используются только биполярные сигналы. *Матрица весов* W, связывающая обе части сети, является действительной и в общем случае несимметричной. При прямом распространении сигналов веса описываются матрицей W, а при обратном — матрицей W^T.

Пусть входные обучающие данные представляют собой множество пар $\{(x_i, y_i), i = 1, 2, \ldots, m\}$ биполярных векторов. На основе этого *множества* формируется *матрица*

$$W = \sum_{i=1}^{n} x_i^T y_i.$$

В результате процесса двунаправленной обработки сигналов формируются два стабильных вектора x_f и y_f, удовлетворяющих уравнениям

$$y_f = f(x_f W),$$
$$x_f = f(y_f W^T) = f(W_y f^T)$$

Каждой промежуточной точке (x_k, y_k) можно сопоставить энергетическую функцию

$$E_k = -x_k W y_k^T,$$

которая убывает при каждом изменении состояния вплоть до достижения локального минимума

$$E_{min} = -x_f W y_f^T, \quad f \in 1, 2, \ldots, m.$$

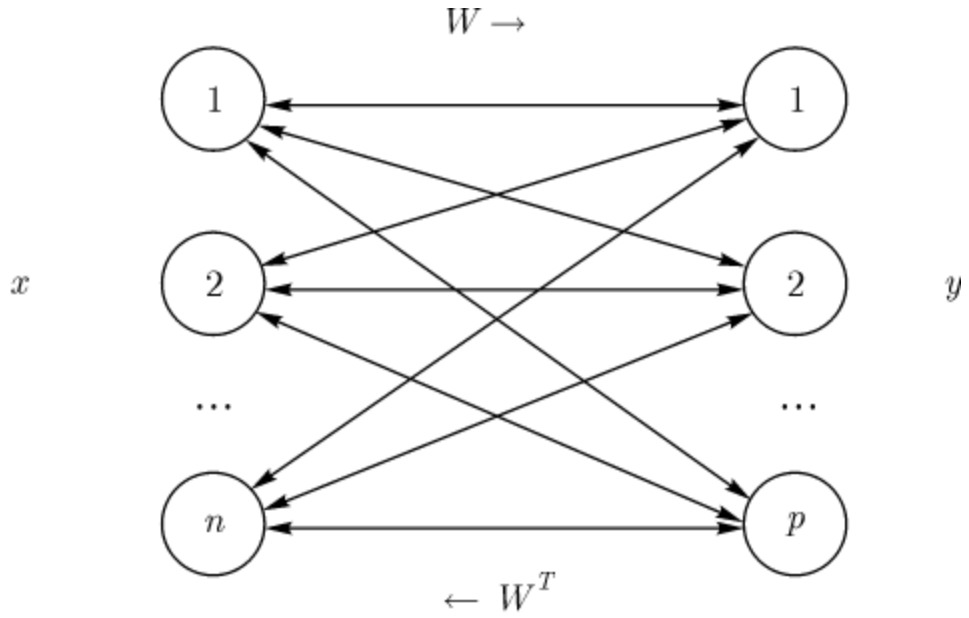


Рис. 3. Структура сети BAM

В режиме распознавания при начальных значениях векторов, совпадающих с использованными при обучении, *сеть* распознает их безошибочно. При искажении векторов x и y *сеть* BAM не всегда способна откорректировать эти векторы и распознает их с определенными погрешностями. Если размерности векторов x и y обозначить соответственно n и p, то удовлетворительное качество распознавания можно получить при выполнении зависимости

$$m < \sqrt{\min(n, p)},$$

где m - число запоминаемых в сети BAM пар векторов.

Контрольные вопросы ЛР5(ПК-4):

1. Автоассоциативная сеть Хопфилда.
2. Обучение сети Хопфилда по правилу Хебба.
3. Сеть Хемминга. Двунаправленная ассоциативная память.
4. Инструментальный Комплекс Для Создания Статических Экспертных Систем.
5. Средства представления знаний и стратегии управления.
6. Инструментальный комплекс для создания экспертных систем реального времени.
7. Понятия и основные требования к системе кодирования информации.
8. Внутримашинное информационное обеспечение.
9. Проектирование экранных форм электронных документов.
10. Информационная база и способы ее организации.
11. Моделирование информационного обеспечения.
12. Моделирование данных. Метод IDEFI.
13. Уровни отображения модели. Создание физической модели.
14. Уровни физической модели; таблицы; правила валидации и значение по умолчанию.
15. Уровни физической модели; индексы; триггеры и хранимые процедуры.
16. Уровни физической модели; проектирование хранилищ данных; вычисление размера БД.
17. Уровни физической модели; таблицы; прямое и обратное проектирование

Лабораторная работа №6. Решение задач комбинаторной оптимизации рекуррентными сетями. Решение задачи коммивояжера сетью Хопфилда. Машина Больцмана. Функция консенсуса. Максимизация консенсуса. Синхронное и асинхронное функционирование машины Больцмана. Решение задачи коммивояжера машиной Больцмана.

Аннотация: Рассматривается решение задачи коммивояжера сетью Хопфилда и машиной Больцмана. Оцениваются параметры функции энергии нейронных сетей, обеспечивающие решение задачи коммивояжера.

Решение задачи коммивояжера сетью Хопфилда

Рассмотрим *задачу коммивояжера* для n городов. Известны расстояния d_{XY} между каждой парой городов X, Y; коммивояжер, выходя из одного города, должен посетить $n-1$ других городов, заходя *по* одному разу в каждый, и вернуться в исходный. Требуется определить порядок обхода городов, при котором общее пройденное *расстояние* минимально.

Пусть *сеть Хопфилда* состоит из $N = n^2$ нейронов, а состояние нейронов описывается двойными индексами v_{Xi}, где *индекс* X связан с именем города, i - с позицией города в маршруте коммивояжера. Запишем функцию вычислительной энергии для сети, предназначенной решать задачу коммивояжера. В ней состояние с наименьшей энергией должно соответствовать самому короткому маршруту. *Функция* энергии должна удовлетворять следующим требованиям:

1) должна поддерживать устойчивое состояние в форме матрицы

$$V = \{v_{Xi}\}, \quad (1)$$

в которой строки соответствуют городам, столбцы - их номерам в маршруте; в каждой строке и каждом столбце только одна *единица*, остальные нули;

2) из всех решений вида (1) *функция* энергии должна поддерживать те, которые соответствуют коротким маршрутам.

Таким требованиям удовлетворяет *функция* энергии в виде:

$$E = (A/2)\sum_X \sum_i \sum_{j\neq i} v_{Xi}v_{Xj} + (B/2)\sum_i \sum_X \sum_{Y\neq X} v_{Xi}v$$

$$+(C/2)(\sum_X \sum_i v_{Xi} - n)^2 + (D/2)\sum_X \sum_{X\neq Y} \sum_i d_{XY}v_{Xi}(v_{Y,i+1} - v_{Y,i-1}), \quad (2)$$

где первые три члена поддерживают первое требование, четвертый член — второе. Первый член равен нулю, если каждая строка X содержит не более одной единицы. Второй равен нулю, если каждый столбец i содержит не более одной единицы. Третий равен нулю, если в матрице всего n единиц. Короткие маршруты поддерживает четвертый член. В нем индексы i берутся *по* модулю n для того, чтобы показать, что n -й город соседствует в маршруте с $(n-1)-$ м, т.е. $v_{Y,n+j} = v_{Y,j}$. Четвертый член численно равен *длине маршрута*. Каноническое *выражение* для функции вычислительной энергии имеет вид

$$E = -(1/2)\sum_X \sum_i \sum_Y \sum_j W_{Xi,Yj} v_{Xi} v_{Xj} - \sum_{xi} I_X \quad 3)$$

Из (2) и (3) получаем веса сети Хопфилда:

$$W_{Xi,Yj} = -A\delta_{XY}(1-\delta_{ij}) - B\delta_{ij}(1-\delta_{XY}) - C - Dd_{XY}(\delta_{j,i+1} + \delta_{j,i-1}),$$
$$I_{Xi} = Cn.$$

Здесь δ - символ Кронекера.

Моделирование работы сети Хопфилда показало, что лучшее *по* качеству решение дает *сеть*, нейроны которой имеют сигмовидную характеристику, а *сеть*, в которой нейроны имеют ступенчатые переходы, приходила к финальным состояниям, соответствующим маршрутам немного лучшим, чем случайные. Многочисленные исследования показывают, что качество решения задачи минимизации функции энергии (2) существенно зависит от выбора производной сигмовидной униполярной *функции активации нейрона* в окрестности нуля. При малой величине производной минимумы энергии оказываются в центре *гиперкуба* решений (некорректное решение), при большой величине производной *сеть* Хопфилда попадает в вершину *гиперкуба*, соответствующую локальному минимуму функции энергии. Кроме того, на качество решения существенное влияние оказывает выбор коэффициентов A, B, C, D. *Поиск* методов оптимального выбора этих коэффициентов является в настоящее время предметом интенсивных исследований.

Машина Больцмана

Математической основой для решения комбинаторных оптимизационных задач на *машине Больцмана* является *алгоритм*, моделирующий затвердевание жидкостей или расплавов (*алгоритм* имитации отжига). Он базируется на идеях из двух различных областей: статистической физики и комбинаторной оптимизации. *Машина Больцмана* (МБ) способна реализовать этот *алгоритм* параллельно и асинхронно. МБ задается четверкой $B = (N, E, W, V_0), N$ - число нейронов, $E = \{(i,j)\}$ - множество связей между нейронами, при этом все автосвязи принадлежат этому множеству, т.е. $(i,i) \in E$. Каждый *нейрон* может иметь состояние 0 или 1. Состояние V_k МБ определяется состояниями нейронов $V_k = (v_1^k, \ldots, v_N^k), V_0$ - начальное состояние. Каждая *связь* (i,j) имеет *вес* w_{ij} - *вещественное число*, множество связей - W. *Связь* (i,j) называется активной в состоянии V_k, если $v_i^k v_j^k = 1$. *Вес* связи (i,j) интерпретируется как количественная *мера* желательности, чтобы эта *связь* была активной. При $w_{ij} \gg 0$ - *активность* очень желательна, при $w_{ij} \ll 0$ - *активность* очень нежелательна. Как и в модели Хопфилда, связи в МБ симметричны, т.е. $w_{ij} = w_{ji}$.

Функция консенсуса

Для состояния V_k МБ вводится понятие консенсуса

$$C_k = \sum_{i,j} w_{ij} v_i^k v_j^k.$$

Каждая *связь* в этой сумме учитывается один раз. Консенсус C_k интерпретируется как количественная *мера* желательности, чтобы все связи (i, j) в состоянии V_k были активны. Для состояния V_k определяется множество соседей $V^{(k)}$. Соседнее состояние $V_{k(i)} \in V^{(k)}$ получается из V_k при изменении состояния нейрона i,

$$V_j^{k(i)} = \begin{cases} v_j^k \text{ если } j \neq i \\ 1 - v_j^k \text{ если } j = i \end{cases}$$

Разница консенсусов соседних состояний V_k и $V_{k(i)}$ равна

$$\Delta C_{kk(i)} = C_{k(i)} - C_k = (1 - 2v_i^k)(\sum_{(i,j) \in E(i)} w_{ij} v_i^k + w_{ii}),$$

где $E(i)$ - множество связей нейрона i. Видно, что $\Delta C_{kk(i)}$ для всех $V_{k(i)} \in V^{(k)}$ могут вычисляться параллельно.

Максимизация консенсуса

Переход МБ *из одного состояния в другое с максимизацией консенсуса* происходит путем выполнения пошаговой процедуры. На каждом ее шаге выполняется *испытание*, состоящее из двух частей:

1. для данного состояния V_k генерируется соседнее $V_{k(i)}$,
2. оценивается, может ли быть принято состояние $V_{k(i)}$, если может, то результат испытания - $V_{k(i)}$, иначе V_k.

Состояние $V_{k(i)}$ принимается с вероятностью

$$P_{kk(i)}(t) = 1/[1 + \exp(\Delta C_{kk(i)}/t)], \qquad 4)$$

где $t \geq 0$ - управляющий *параметр* ("температура").

Процесс максимизации консенсуса начинается с высокого значения t_0 параметра t и случайно выбранного начального состояния V_0. В течение процесса *параметр* t уменьшается от t_0 до 0. *По* мере того как t приближается к нулю, нейроны все реже изменяют свои состояния, и наконец, МБ стабилизируется в финальном состоянии. Практически, МБ стабилизируется в состоянии, соответствующем *локальному максимуму* консенсуса, который близок (или равен) глобальному. Сходимостью МБ управляют следующие параметры:

1. Начальное *значение* параметра t для каждого нейрона i

$$t_0^{(i)} = \sum_{(i,j)\in E(i)} |w_{ij}| + |w_{ii}|.$$

2. Правило понижения t

$$t_{j+1}^{(i)} = \alpha t_j^{(i)},$$

где α - положительное число, меньшее единицы, но близкое к ней.

3. Число L испытаний, которые проводятся без изменения t (L — *функция* от N).

4. Число M последовательных испытаний, не приводящих к изменению состояния машин (M - *функция* от N), как критерий завершения процесса.

Синхронное и асинхронное функционирование машины Больцмана

Для выполнения синхронного процесса все множество нейронов разбивается на непересекающиеся подмножества $\{M_1, \ldots, M_m\}$, такие, что нейроны, попавшие в одно *подмножество*, не связаны друг с другом. Тогда на каждом такте синхронизации элементы случайно выбранного подмножества M_i могут одновременно изменять свои состояния в соответствии с заданной вероятностью.

В асинхронном параллельном процессе все нейроны могут изменять свои состояния только в зависимости от величины вероятности. Практически асинхронный *параллелизм* может быть выполнен следующим образом. Случайно выбирается *подмножество* M, содержащее $q = 2N/3$ нейронов. Для каждого нейрона из этого подмножества устанавливается состояние в соответствии с $P_{kk(i)}(t)$. Получившееся в результате состояние есть результат одного асинхронного шага.

Решение задачи коммивояжера машиной Больцмана

Общий подход к программированию комбинаторных оптимизационных задач состоит в следующем:

каждое решение представляется набором $\{x_1, \ldots, x_N\}, x_i \in \{0, 1\}$, N — число нейронов в сети, x_i - состояние нейрона. Структура связей и веса выбираются так, что:

$R1$. Все локальные максимумы функции консенсуса соответствуют приемлемым решениям задачи;

$R2$. Чем лучше приемлемое решение, тем больше консенсус соответствующего состояния *машины Больцмана*.

Перефразируем для МБ *задачу коммивояжера*.

$R1$. Состояние МБ соответствует *локальному максимуму* функции консенсуса, если и только если это состояние соответствует приемлемому маршруту.

$R2$. Чем короче *маршрут*, тем выше консенсус соответствующего состояния МБ.

Каждый *нейрон* соответствует элементу матрицы $n \times n$, состояния нейронов обозначаются v_{Xi} (n - число городов). *Функция* консенсуса

$$C_k = \sum_{(Xi,Yj)} w_{Xi,Yj} v_{Xi}^k v_{Yj}^k.$$

Множество связей в сети определяется как *объединение* трех непересекающихся подмножеств:

E_d - множество связей, несущих информацию о расстояниях между городами,

$$E_d = \{(Xi, Yj) | (X \neq Y) \wedge (i = (j+1) mod n)\};$$

E_i - множество ингибиторных (запретительных) связей,

$$E_i = \{(Xi, Yj) | (i \neq j) \wedge (X = Y) \vee (i = j) \wedge (X \neq Y)\};$$

E_b - множество связей смещений,

$$E_b = \{(Xi, Yj) | (X = Y) \wedge (i = j)\}.$$

Здесь $X, Y, i, j = 1, \ldots, n$. Общее число связей равно $2n^3 - n^2$.

Ингибиторные связи гарантируют, что, в конце концов, ни в одной строке и ни в одном столбце не будет более одной единицы. Связи смещений гарантируют, что хотя бы *по* одной единице есть в каждом столбце и в каждой строке. Таким образом, связи E_i и E_b гарантируют выполнение ограничений в задаче и все их дают одинаковые вклады в консенсусы для всех приемлемых маршрутов.

Связь $(Xi, Yj) \in E_d$ активна только в том случае, когда в маршруте есть *прямой путь* из города X в город Y. *Вес* связи $(Xi, Yj) \in E_d$ равен расстоянию между городами X и Y с отрицательным знаком. Следовательно, для данного маршрута отрицательный вклад связи из E_d в консенсус пропорционален длине пути, поэтому максимизация функции консенсуса соответствует минимизации длины маршрута.

Доказано, что для консенсуса C_k выполняются требования $R1$ и $R2$, если и только если веса связей выбраны следующим образом:

$$\forall (Xi, Yj) \in E_d : w_{Xi,Yj} = -d_{XY},$$
$$\forall (Xi, Yj) \in E_i : w_{Xi,Yj} < -\min(\mu_X, \mu_Y),$$

$$\forall(Xi, Yj) \in E_b : w_{Xi,Yj} > \mu_X,$$

где

$$\mu_X = \max\{d_{XP} + d_{XQ} | P, Q = 1, \ldots, n \wedge (P \neq Q)\}.$$

При $d = 0,95, L = 10, M = 100$ было проведено 100 испытаний для $n = 10$ и 25 испытаний для $n = 30$ при различных начальных состояний МБ. Для $n = 10$ получено оптимальное решение, для $n = 30$ получено решение на 14% хуже оптимума. Вероятностный механизм функционирования МБ дает возможность получать на ней несколько лучшие результаты, чем на модели Хопфилда.

Контрольные вопросы ЛР6(ПК-4):

1. Решение задачи коммивояжера сетью Хопфилда.
2. Машина Больцмана.
3. Функция консенсуса.
4. Максимизация консенсуса.
5. Синхронное и асинхронное функционирование машины Больцмана.
6. Решение задачи коммивояжера машиной Больцмана.
7. Мягкая экспертная система.
8. Определение мягкой экспертной системы.
9. Сравнение нечеткой и мягкой экспертных систем.
10. Представление знаний в мягкой экспертной системе.
11. Содержание баз знаний и данных мягкой экспертной системы.
12. Основные элементы диаграмм взаимодействия — объекты, сообщения.
13. Диаграммы состояний.
14. Диаграммы начального состояния, конечного состояния, переходы.
15. Вложенность состояний.
16. Диаграммы внедрения: подсистемы, компоненты, связи.

Лабораторная работа №7. Самоорганизация (самообучение) нейронных сетей. Классификация без учителя. Метод динамических ядер в классификации без учителя. Алгоритмы обучения сетей с самоорганизацией. Алгоритм Кохонена. Применение сетей с самоорганизацией. Компрессия данных. Прогнозирование нагрузок энергетической системы.

Аннотация: Рассматриваются: метод динамических ядер в классификации без учителя, алгоритмы обучения сетей с самоорганизацией и их применение к компрессии данных и прогнозированию.

Ключевые слова: вектор, мера, евклидово расстояние, расстояние, определение, значение, класс, число классов, выборка, пространство, ядро, разбиение, минимум, решающее правило, сумматор, поиск, максимум, норма, сеть, погрешность, вес, vector, quantization, алгоритм, WTA, ALL, активность, нейрон, скалярное произведение, MOST, функция, feature, MAP, самоорганизующаяся карта, Размещение, кластер, кадр, компонент, знание

Классификация без учителя

Задан набор объектов, каждому объекту поставлен в соответствие *вектор* значений признаков (строка таблицы). Требуется разбить эти объекты на классы эквивалентности. Для каждого нового объекта нужно:

- Найти класс, к которому он принадлежит.
- Использовать новую информацию, полученную об этом объекте, для исправления (коррекции) правил классификации.

Отнесение объекта к классу проводится путем его сравнения с типичными элементами разных классов и выбора из них ближайшего.

Простейшая *мера* близости объектов - квадрат *евклидова расстояния* между векторами значений их признаков (чем меньше *расстояние*, тем ближе объекты). Соответствующее *определение* признаков типичного объекта - среднее арифметическое *значение* признаков по выборке, представляющей *класс*. Другая *мера* близости, возникающая при обработке сигналов, изображений и т.п. - квадрат коэффициента корреляции (чем он больше, тем ближе объекты). Возможны и иные варианты.

Если *число классов* m заранее определено, то задачу классификации без учителя можно поставить следующим образом.

Метод динамических ядер в классификации без учителя

Пусть задана *выборка* предобработанных векторов данных $\{x\} \subseteq E, E$ - *пространство* векторов данных. Каждому классу будет соответствовать некоторое *ядро* $w \subseteq W, W$ - *пространство* ядер.

Для любых $x \in E$ и $w \in W$ определим меру близости $d(x, w)$, а для каждого набора из k ядер $w_1, \dots w_k$ и любого разбиения $\{x\}$ на k классов $\{x\} = P_1 \smile P_2 \smile \dots \smile P_k$ определим критерий качества

$$D = D(w_1, \ldots, w_k, P_1, \ldots, P_k) = \sum_{i=1}^{k} \sum_{x \in P_i} d(x, w_i \quad 1)$$

Требуется найти набор $w_1, \ldots, w_k$ и *разбиение* $P_1, \ldots, P_k$, минимизирующие D. Шаг алгоритма разбиваем на 2 этапа:

1) Для фиксированного набора ядер $w_1, \ldots, w_k$ ищем минимизирующее D *разбиение* $P_1, \ldots, P_k$; оно дается следующим решающим правилом: $x \in P_i$, если $d(x, w_i) < d(x, w_j)$ при $i \neq j$ (когда для x *минимум* $d(x, w_i)$ достигается при нескольких значениях i, выбор между ними может быть сделан произвольно).

2) Для каждого $P_i, i \in 1, \ldots, k$, полученного на первом этапе, отыскивается $w_i \in W$, минимизирующее критерий качества

$$D_i = \sum_{x \in P_i} d(x, w_i).$$

Начальные значения $w_1, \ldots, w_k$, $P_1, \ldots, P_k$ выбираются произвольно либо по какому-нибудь эвристическому правилу. Если ядру w_i ставится в соответствие элемент сети, вычисляющей по входному сигналу x функцию $d(x, w_i)$, то *решающее правило* для классификации дается интерпретатором "проигравший забирает все": элемент x принадлежит классу P_i, если выходной сигнал i -го элемента $d(x, w_i)$ меньше всех остальных. *Мера* близости d выбирается такой, чтобы легко можно было найти *ядро* w_i, минимизирующее D_i для данного P_i.

В простейшем случае *пространство* ядер W совпадает с E, а $d(x, w_i)$ - положительно определенная квадратичная форма от $x - w_i$, например, квадрат *евклидова расстояния*. Тогда *ядро* w_i, минимизирующее D_i, есть центр масс класса P_i:

$$w_i = (1/|P_i|) \sum_{x \in P_i} x,$$

где $|P_i|$ - число элементов в P_i.

Пусть векторы пространства E нормированы. Тогда

$$(x, x) = (w_i, w_i) = 1. \quad 2)$$

Так как $d(x, w_i) = (x - w_i, x - w_i) = (x, x) - 2(x, w_i) + (w_i, w_i)$, то с учетом (2) упрощается *решающее правило*, разделяющее классы:

$$x \in P_i, \text{ если } (x, w_i) > (x, w_j) \text{ при } i \neq j,$$

поскольку *минимум* $d(x, w_i)$ достигается при максимуме (x, w_i). Такое решающее правило реализуется с помощью k сумматоров, вычисляющих (x, w_i), и интерпретатора, выбирающего *сумматор* с максимальным выходным сигналом. Номер этого сумматора и есть номер класса, к которому относится x.

Задача поиска ядра w_i для класса P_i превращается в *поиск* вектора w, максимизирующего

$$D_i = \sum_{x \in P_i} (x, w).$$

Этот *максимум* достигается в точке

$$w = \sum_{x \in P_i} x / \| \sum_{x \in P_i} x \|$$

где $\| \dots \|$ - евклидова *норма*.

В тех простейших случаях, когда *ядро* класса точно определяется как среднее арифметическое (или нормированное среднее арифметическое) элементов класса, а *решающее правило* основано на сравнении выходных сигналов линейных адаптивных сумматоров, нейронную *сеть*, реализующую метод динамических ядер, называют сетью Кохонена. В *определение* ядра w_i для сетей Кохонена входят суммы $\sum_{x \in P_i} x$. Это позволит накапливать новые динамические ядра, обрабатывая по одному примеру и пересчитывая w_i после получения в P_i нового примера.

Если *число классов* заранее не определено, то полезен критерий слияния классов: классы Y_i и Y_j сливаются, если *расстояние* между их ядрами меньше, чем среднее *расстояние* от элемента класса до ядра в одном из них:

$$(y^i, y^j) < \max[(1/|Y_i|) \sum_{x \in Y_i} \rho(x, y^i), (1/|Y_j|) \sum_{x \in Y_j} \rho(x, y^j)],$$

где $|Y|$ - число элементов в Y. Использовать критерий слияния классов можно так: сначала принимаем гипотезу о достаточном числе классов, строим их, минимизируя D, затем некоторые Y_i объединяем, повторяем минимизацию D с новым числом классов и т.д.

Алгоритмы обучения сетей с самоорганизацией

Целью обучения сети с самоорганизацией на основе конкуренции нейронов считается такое упорядочение нейронов (подбор значений их весов), которое минимизирует *значение* ожидаемого искажения, оцениваемого погрешностью аппроксимации входного вектора x значениями весов нейрона-победителя. При p входных векторах x и применении евклидовой метрики эта *погрешность*, называемая также погрешностью квантования, может быть выражена в виде

$$E = (1/p)\sum_{i=1}^{p} \|x^i - w_{win}\|^2, \qquad 3)$$

где w_{win} - *вес* нейрона-победителя при предъявлении вектора x^i.

Этот подход также называется векторным квантованием (англ. *Vector Quantization* - VQ) или кластеризацией. Номера нейронов-победителей при последовательном предъявлении векторов x^i образуют так называемую кодовую таблицу. При классическом решении задачи кодирования применяется *алгоритм* K -усреднений (англ. K-means), носящий имя обобщенного алгоритма Ллойда.

Для нейронных сетей аналогом алгоритма Ллойда считается *алгоритм WTA* (англ.: Winner Takes *All* - "победитель получает все"). В соответствии с ним после предъявления вектора x рассчитывается *активность* каждого нейрона. Победителем признается *нейрон* с самым сильным выходным сигналом, т.е. тот, для которого *скалярное произведение* (x, w) оказывается наибольшим. В предыдущем разделе было показано, что при использовании нормализованных векторов это равнозначно наименьшему эвклидову расстоянию между входным вектором и вектором весов нейронов. Победитель получает право уточнить свои веса в направлении вектора x согласно правилу

$$w_{win} \longleftarrow w_{win} + \alpha(x - w_{win}),$$

где α - коэффициент обучения. Веса остальных нейронов уточнению не подлежат. *Алгоритм* позволяет учитывать усталость нейронов путем подсчета количества побед каждого из них и поощрять элементы с наименьшей активностью для выравнивания их шансов. Такая модификация применяется чаще всего на начальной стадии обучения с последующим отключением после активизации всех нейронов. Подобный способ обучения реализован в виде режима CWTA (Conscience Winner Takes *All*) и считается одним из лучших и наиболее быстрых алгоритмов самоорганизации.

Помимо алгоритмов *WTA*, в которых в каждой итерации может обучаться только один *нейрон*, для обучения сетей с самоорганизацией широко применяются алгоритмы типа WTM (англ.: Winner Takes *Most* - "победитель получает больше"), в которых, кроме победителя, уточняют значения своих весов и нейроны из его ближайшего окружения. При этом, чем дальше какой-либо *нейрон* находится от победителя, тем меньше изменяются его веса. Процесс уточнения вектора весов может быть определен обобщенной зависимостью, которая здесь представляется в виде

$$w_i \longleftarrow w_i + \alpha G(i, x)[x - w_i]$$

для $G(i,x)$ всех нейронов, расположенных в окрестности победителя. Если *функция* определяется в форме

$$G(i,x) = \{1 \text{ для } i = I, 0 \text{ для } i \neq I\},$$

где I обозначает номер победителя, то мы получаем классический *алгоритм WTA*. Существует множество вариантов алгоритма WTM, отличающихся прежде всего формой функции $G(i,x)$. Для дальнейшего изучения выберем классический *алгоритм* Кохонена.

Алгоритм Кохонена

Алгоритм Кохонена относится к наиболее старым алгоритмам обучения сетей с самоорганизацией на основе конкуренции, и в настоящее время существуют различные его версии. В классическом алгоритме Кохонена *сеть* инициализируется путем приписывания нейронам определенных позиций в пространстве и связывания их с соседями на постоянной основе. Такая *сеть* называется самоорганизующейся картой признаков (*сеть* SOFM - Self-Organizing *Feature Map*). В момент выбора победителя уточняются не только его веса, но также и веса его соседей, находящихся в ближайшей окрестности. Таким образом, *нейрон*-победитель подвергается адаптации вместе со своими соседями. В классическом алгоритме Кохонена *функция* соседства $G(i,x)$ определяется в виде

$$G(i,x) = \{1 \text{ для } d(i,I) \leqslant L, 0 \text{ для } d(i,I) > L\}.$$

В этом выражении $d(i,I)$ обозначает эвклидово *расстояние* между векторами весов нейрона-победителя I и i -го нейрона. Коэффициент L выступает в роли уровня соседства, его *значение* уменьшается в процессе обучения до нуля. Соседство такого рода называется прямоугольным.

Другой тип соседства, часто применяемый в картах Кохонена, - это соседство гауссовского типа, при котором *функция* $G(i,x)$ задается формулой

$$G(i,x) = exp(-d^2(i,x)/2\lambda^2).$$

Степень адаптации нейронов-соседей определяется не только евклидовым расстоянием между i -м нейроном и победителем (I -м нейроном), но также и уровнем соседства λ. В отличие от соседства прямоугольного типа, где каждый *нейрон*, находящийся в окрестности победителя, адаптировался в равной степени, при соседстве гауссовского типа уровень адаптации различен и зависит от значения функции Гаусса. Как правило, гауссовское соседство дает лучшие результаты обучения и обеспечивает лучшую организацию сети, чем прямоугольное соседство.

Самоорганизующаяся карта признаков проходит два этапа обучения. На первом этапе элементы упорядочиваются так, чтобы отражать *пространство* входных элементов, а на втором происходит уточнение их позиций. Как правило, процесс представляется визуально путем использования двумерных данных и построения соответствующей поверхности. Например, входные векторы выбираются случайным образом на основе однородного распределения в некотором квадрате, и начинается обучение карты. В определенные моменты в ходе обучения строятся изображения карты путем использования соответствия, показанного на рис. 1. Элементы соединяются линиями, чтобы показать их относительное *размещение*. Сначала карта выглядит сильно "измятой", но постепенно в ходе обучения она разворачивается и расправляется. Конечным

результатом обучения является карта, покрывающая все входное *пространство* и являющаяся достаточно регулярной (т.е. элементы оказываются распределенными почти равномерно). Для примера была рассмотрена карта с топологией квадрата из 49 элементов, и для 250 точек данных, взятых из единичного квадрата, было проведено ее обучение, которое начиналось со случайного набора весовых значений, задающих *размещение* кластерных элементов в центре входного пространства, как показано на рис. 1. На рис. 2 и 3 иллюстрируется процесс разворачивания карты с течением времени. Как и для других типов сетей, в данном случае результат обучения зависит от учебных данных и выбора параметров обучения.

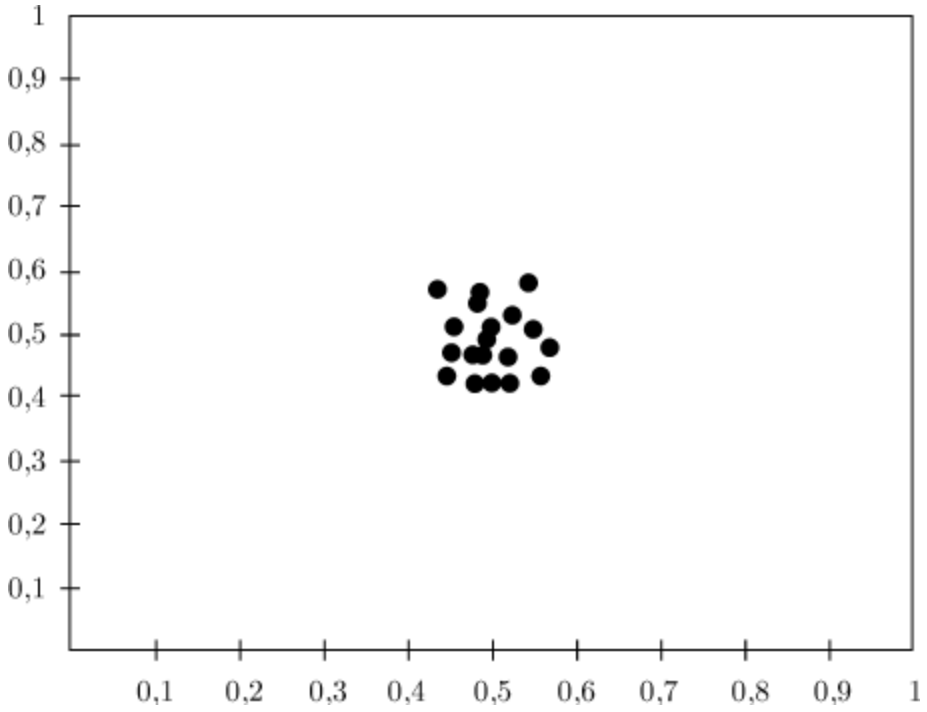


Рис. 1. Весовые векторы инициализируются случайными значениями из диапазона 0.4-0.6

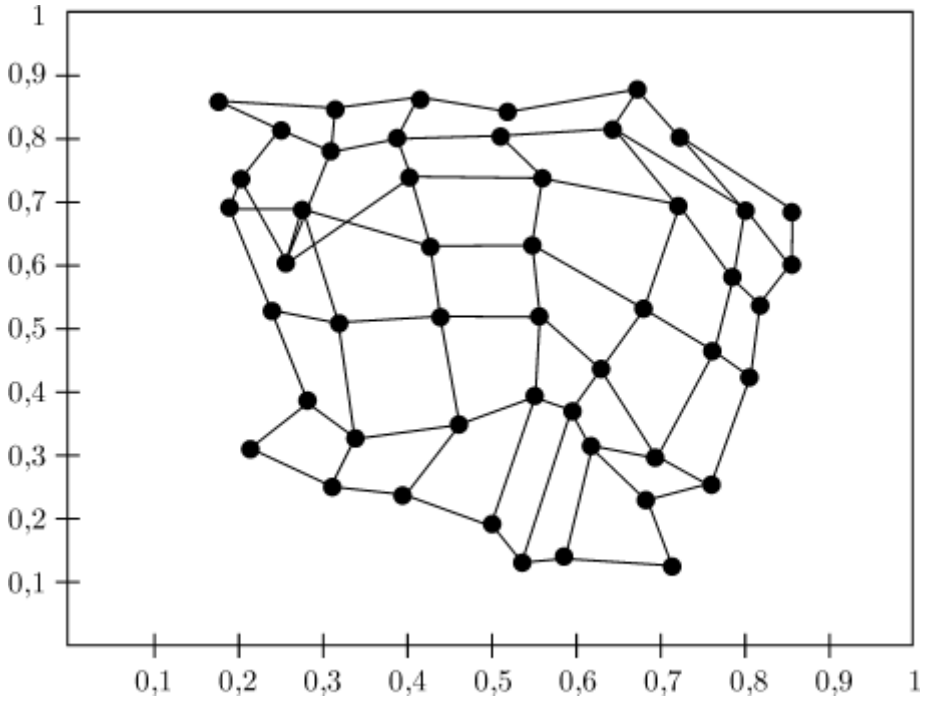


Рис. 2. Карта по прошествии 20 итераций

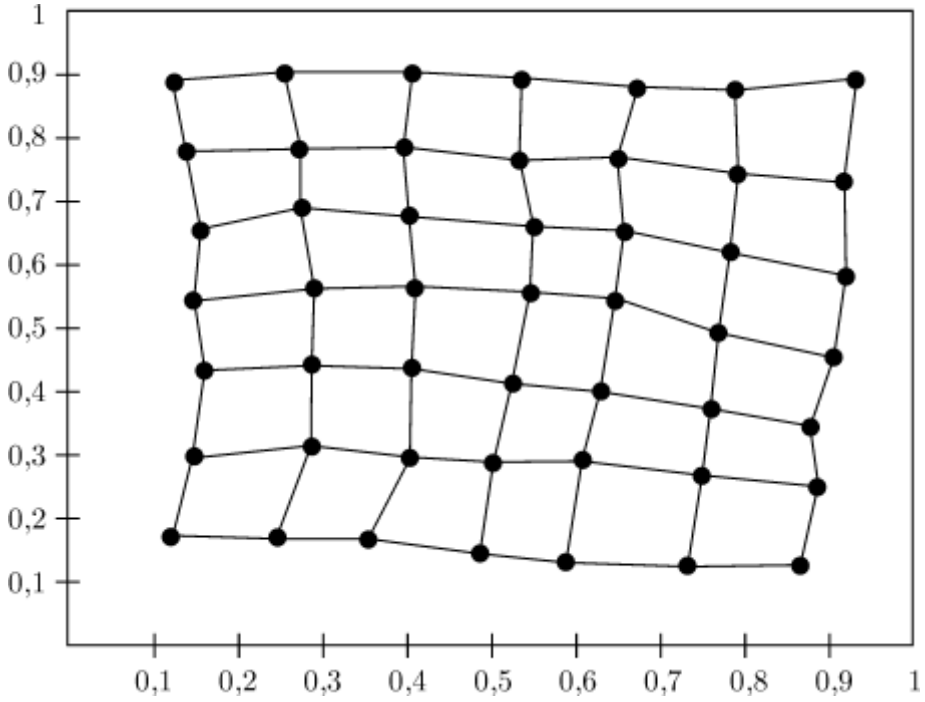


Рис. 3. Карта незадолго до окончания обучения. Элементы теперь упорядочены, и карта станет еще более регулярной по окончании финальной фазы сходимости

Применение сетей с самоорганизацией

Главным свойством сети Кохонена считается компрессия данных, состоящая в том, что образующие *кластер* группы данных представляются единственным вектором весов нейрона-победителя. При разделении данных на кластеры и представлении каждого кластера одним из нейронов достигается значительное сокращение объема используемой под данные памяти, которое и называется компрессией. Это компрессия с потерей информации, которая сопровождается определенной погрешностью квантования.

Компрессия данных

Примером использования компрессионных свойств сети Кохонена может считаться сжатие изображений, предназначенное для уменьшения количества информации, представляющей конкретный образ, при сохранении погрешности восстановления на заданном уровне.

Пусть изображение разделяется на одинаковые кадры размером $n_x \times n_y$ пикселов. Образующие *кадр* пикселы представляют собой компоненты входного вектора x.

Сеть с самоорганизацией содержит n нейронов, каждый из которых имеет входом *вектор* x. Обучение сети при помощи одного из алгоритмов самоорганизации состоит в подборе таких весов конкретных нейронов, при которых минимизируется *погрешность* квантования (3). В результате обучения формируется структура сети, при которой вектору x каждого кадра соответствует *вектор* весов нейрона победителя. В процессе предъявления очередного кадра выбирается номер нейрона-победителя. Номера нейронов-победителей образуют кодовую таблицу, а веса этих нейронов представляют средние значения, соответствующим конкретным компонентам вектора x (т.е. уровням интенсивности пикселов, составляющих *кадр*).

Поскольку количество нейронов обычно намного меньше количества кадров, то можно получить существенное сокращение объема данных, описывающих исходное изображение. В итоге коэффициент компрессии изображения равен

$$K = N \cdot n_x n_y T/(N \cdot \lg_2 n + n \cdot n_x n_y t),$$

где n_x и n_y - размеры кадра в осях x и y, N - количество кадров, n - количество нейронов, а T и t - количество битов для представления соответственно градаций интенсивности пиксела и значений весов. Этот подход позволяет получить степень компрессии изображений порядка 16 при значении коэффициента сигнал/шум (PSNR) около 26-28 дБ.

Прогнозирование нагрузок энергетической системы

Рассмотрим решение задачи прогнозирования часовых нагрузок в элктроэнергетической системе на 24-часовом интервале. Пусть имеется база данных, содержащая векторы профильных нагрузок дня

$$p_j = [p(j,1), p(j,2), \ldots, p(j,24)],$$

где *компонент* $p(j,k)$ соответствует действительной нагрузке в k -й час суток. Множество профильных векторов подается на вход сети Кохонена, состоящей из n нейронов. Процесс самоорганизации сети приводит к автоматической кластеризации данных и к сопоставлению каждому кластеру одного из нейронов сети. Этот *нейрон* считается победителем, а его веса наилучшим образом адаптируются к усредненным весам профильных векторов, составляющих *кластер*. Характерная особенность состоит в том, что соседние векторы имеют сходные профильные характеристики.

Близость весов нейронов, расположенных недалеко друг от друга, объясняется тем, что один и тот же день в разные годы при небольших отличиях в часовых нагрузках может возбуждать различные нейроны, которые образуют кластеры, группирующие данные сходных классов.

Знание таблицы распределения побед конкретных нейронов сети позволяет относительно легко предвидеть профили часовых нагрузок для произвольного дня года. С этой целью создаются таблицы принадлежности каждого дня года к области доминирования определенного нейрона с обозначением количества его побед для всех дней в прошлом. Для выбора прогнозируемого профиля нагрузок актуального дня в требуемом месяце рассчитываются усредненные значения весов нейронов победителей, которые указывали в прошлом на требуемый день. Если количество побед i -го нейрона, соответствующего j -му дню, обозначить k_{ji}, а соответствующие векторы весов класса - w_i, то прогнозируемый профильный *вектор* j -го дня рассчитывается по формуле

$$p_j = \sum_{i=1}^{n} k_{ji} w_i / \sum_{i=1}^{n} k_{ji}$$

Контрольные вопросы ЛР7(ПК-4):

1. Классификация без учителя.
2. Метод динамических ядер в классификации без учителя.
3. Алгоритмы обучения сетей с самоорганизацией.
4. Алгоритм Кохонена.
5. Применение сетей с самоорганизацией.
6. Компрессия данных.
7. Прогнозирование нагрузок энергетической системы.

8. Модуль лингвистической обработки.
9. Лингвистический анализ.
10. Формирование просодических характеристик.
11. Синтезатор русской речи.
12. Язык формальной записи правил синтеза.
13. Интонационное обеспечение.
14. Аллофонная база данных.
15. Лингвистический анализ.
16. Инструментарий синтеза русской речи.
17. Система распознавания речи.
18. Акустическая и лингвистическая модели.
19. Классификация систем распознавания.

Лабораторная работа №8. Нечеткие и гибридные нейронные сети. Интеллектуальные информационные системы в условиях неопределенности и риска. Нечеткие множества. Лингвистические переменные. Нечеткие правила вывода. Системы нечеткого вывода Мамдани-Заде. Фазификатор. Дефазификатор. Модель Мамдани-Заде как универсальный аппроксиматор. Нечеткие сети TSK (Такаги-Сугено-Канга). Гибридный алгоритм обучения нечетких сетей. Мягкая экспертная система. Определение мягкой экспертной системы. Сравнение нечеткой и мягкой экспертных систем. Представление знаний в мягкой экспертной системе. Содержание баз знаний и данных мягкой экспертной системы.

Аннотация: Рассматриваются: математические основы нечетких систем, преимущества и алгоритмы обучения нечетких нейронных сетей, нечеткие сети с генетической настройкой, экспертные системы на основе гибридных НС.

Интеллектуальные информационные системы в условиях неопределенности и риска

С помощью *символьной обработки информации* не удается решить прикладные задачи многих предметных областей, если для них невозможно получить полную информацию и если их *определение* недостаточно полно. Такая ситуация характерна для:

- сложных технических систем;
- систем экономического планирования;
- социальных систем большой размерности;
- систем принятия решений и т.п.

Выходом является использование систем, основанных на мягких вычислениях, которые включают в себя:

- нечеткую логику и вероятностные вычисления;
- нейрокомпьютинг - обучение, адаптация, классификация, системное моделирование и идентификация;
- генетические вычисления - синтез, настройка и оптимизация с помощью систематизированного случайного поиска и эволюции.

Эти составные части не конкурируют друг с другом, а создают эффект взаимного усиления (гибридные системы). Наряду с термином "мягкие вычисления" используется термин "вычислительный *интеллект*" - научное направление, где решаются задачи искусственного интеллекта на основе теории нечетких систем, нейронных сетей и эволюционных (генетических) вычислений.

Нечеткие нейронные сети с генетической настройкой параметров (гибридные системы) демонстрируют взаимное усиление достоинств и нивелирование недостатков отдельных методов:

1. Представление знаний в нейронных сетях в виде *матриц весов* не позволяет объяснить результаты проведенного распознавания или прогнозирования, тогда как в системах вывода на базе нечетких правил результаты воспринимаются как ответы на вопросы "почему?".
2. Нейронные сети обучаются с помощью универсального алгоритма, т.е. трудоемкое *извлечение знаний* заменяется сбором достаточной по объему обучающей выборки. Для нечетких систем вывода *извлечение знаний* включает в себя сложные процессы формализации понятий, определение функций принадлежности, формирование правил вывода.

3. Нечеткие нейронные сети обучаются как нейронные сети, но их результаты объясняются как в системах нечеткого вывода.

Нечеткие множества

Понятие нечетких множеств (fuzzy sets) как *обобщение* обычных (четких) множеств было введено Л.Заде в 1965 г.. Традиционный способ представления элемента *множества* A состоит в применении характеристической функции $\mu_A(x)$, которая равна 1, если элемент принадлежит множеству A, или равна 0 в противном случае. В нечетких системах элемент может частично принадлежать любому множеству. Степень принадлежности множеству A, представляющая собой *обобщение* характеристической функции, называется функцией принадлежности $\mu_A(x)$, причем $\mu_A(x) \in [0,1]$, и $\mu_A(x) = 0$ означает отсутствие принадлежности x множеству A, а $\mu_A(x) = 1$ - полную принадлежность. Конкретное *значение* функции принадлежности называется степенью или коэффициентом принадлежности.

Лингвистические переменные

В теории нечетких множеств, помимо переменных цифрового типа, существуют лингвистические переменные с приписываемыми им значениями.

Пусть x обозначает температуру. Можно определить нечеткие *множества* "отрицательная", "близкая к нулю", "положительная", характеризуемые функциями принадлежности $\mu_{\text{отриц}}(x)$, $\mu_{\text{бнул}}(x)$, $\mu_{\text{полож}}(x)$. Лингвистическая *переменная* "температура" может принимать значения "отрицательная" "близкая к нулю", "положительная". *Функция* нечеткой принадлежности является непрерывным приближением пороговой функции точной принадлежности.

Нечеткие правила вывода

Правило вывода

если x это A, то y это B

называется нечеткой импликацией $A \longrightarrow B$, если A и B - лингвистические значения (значения *лингвистической переменной*), идентифицированные нечетким способом через соответствующие функции принадлежности для переменных.

Часть "x это A" называется условием (предпосылкой), а "y это B" - следствием (заключением).

Обобщение для N-мерного вектора x:

если x_1 это A_1 и x_2 это A_2 и ... и x_N это A_N, то y это B, $A_1, A_2, \ldots, A_N, B$ обозначают величины соответствующих коэффициентов принадлежности $\mu_A(x_i), i = 1,2,\ldots,N, \mu_B(y)$.

Возможна *интерпретация* $\mu_A(x)$

- в форме логического произведения

$$\mu_A(x) = min_{i=1,\dots,N}\mu_A(x_i)$$

- в форме алгебраического произведения

$$\mu_A(x) = \prod_{i=1,\dots,N} \mu_A(x_i)$$

(агрегирование предпосылки).

Каждой импликации $A \to B$ можно приписать *значение* функции принадлежности $\mu_{A\to B}(x, y)$:

- форма логического произведения

$$\mu_{A\to B} = min\{\mu_A(x), \mu_B(y)\}$$

- форма алгебраического произведения

$$\mu_{A\to B} = \mu_A(x)\mu_B(y)$$

агрегирование на уровне импликации).

Системы нечеткого вывода Мамдани-Заде

Элементы теории нечетких множеств, правила импликации и нечетких рассуждений образуют систему нечеткого вывода. В ней можно выделить:

- множество используемых нечетких правил;
- базу данных, содержащую описания функций принадлежности;
- механизм вывода и агрегирования, который формируется применяемыми правилами импликации.

В случае технической реализации в качестве входных и выходных сигналов выступают измеряемые величины, однозначно сопоставляющие входным значениям соответствующие выходные значения.

Для обеспечения взаимодействия этих двух видов вводится нечеткая система с так называемым фазификатором (преобразователем множеств входных данных в *нечеткое множество*) на входе и дефазификатором (преобразователем нечетких множеств в конкретное *значение* выходной переменной) на выходе.

Фазификатор преобразует точное множество входных данных в не\-четкое множество, определенное с помощью функции принадлежности, а~дефазификатор решает обратную задачу - формирует однозначное решение относительно *входной* переменной на основании многих нечетких выводов, вырабатываемых исполнительным модулем нечеткой системы.

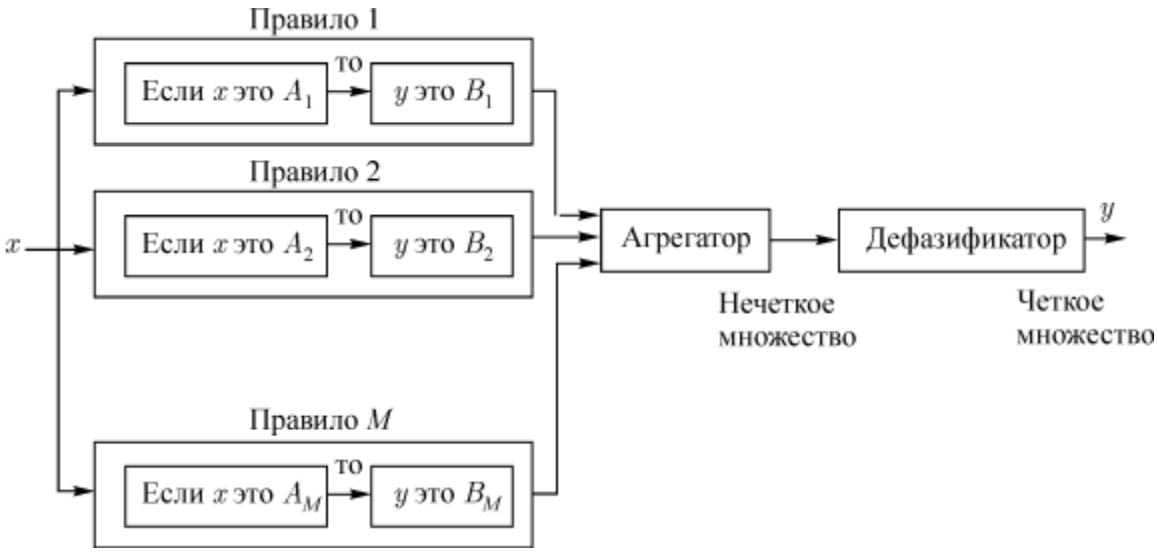


Рис. 1. Вывод в нечеткой системе при наличии M правил

Выходной сигнал модуля вывода может иметь вид M нечетких множеств, определяющих *диапазон* изменения выходной переменной. Дефазификатор преобразует этот *диапазон* в одно конкретное *значение*, принимаемое в качестве выходного сигнала всей системы.

В модели вывода Мамдани-Заде присутствуют следующие *операторы*:

- оператор логического или арифметического произведения для определения результирующего уровня активации, в котором учитываются все компоненты вектора условия;
- оператор логического или арифметического произведения для определения значения функции принадлежности для всей импликации $A \longrightarrow B$;
- оператор логической суммы как агрегатор равнозначных результатов импликации многих правил;
- оператор дефазификации, трансформирующий нечеткий результат $\mu(y)$ в четкое значение y .

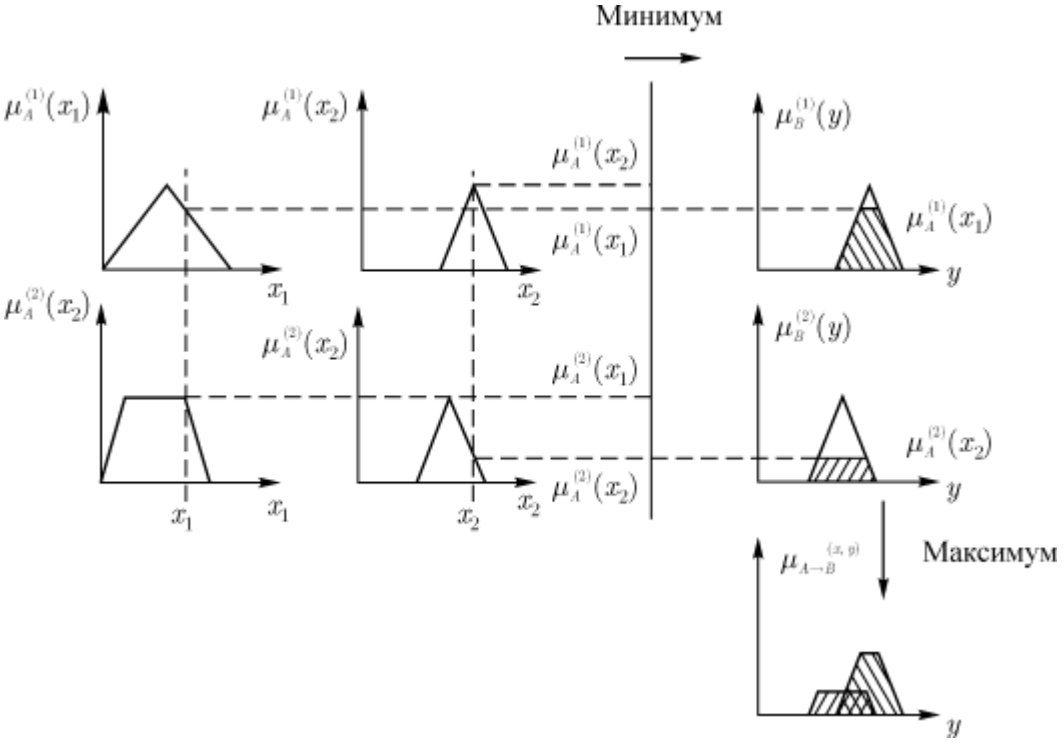


Рис. 2. Пример системы вывода Мамдани-Заде

На рис. 2 представлен способ агрегирования при двух входных переменных x_1, x_2.

Логическое *произведение* (оператор min) используется как для агрегирования нечетких правил относительно конкретных переменных x_i, $i = 1, 2$, образующих *вектор* x, так и на уровне импликации $A \longrightarrow B$ для одиночных правил вывода. *Агрегирование* импликаций, касающихся правил 1 и 2, проводится с использованием логической суммы (оператор max).

Фазификатор

Фазификатор преобразует N -мерный *вектор* $x = [x_1, x_2, \ldots, x_N]$ в нечеткое множество A, характеризуемое функцией принадлежности $\mu_A(x)$.

Наибольшей популярностью пользуются функции гауссовского типа, треугольные и трапецеидальные функции:

- Общая форма гауссовской функции

$$\mu_A(x) = \exp[-(x-c)^2/\sigma^2]$$

c - центр нечеткого множества,

σ - коэффициент широты.

- Симметричная треугольная функция

$$\mu_A(x) = \{1 - |x-c|/d, \text{ при } x \in [c-d, c+d]; \\ 0, \text{ для остальных } x\},$$

c - центр,

d - ширина.

- Трапецеидальная функция

$$\mu_A(x) = \{0, \text{ при } x/gtz \text{ и } x/lty; \\ 1, \text{ при } c - t/2 \leqslant x \leqslant c + t/2; \\ s(z-x), \text{ при } c + t/2 \leqslant x \leqslant z; \\ s(x-y), \text{ при } y \leqslant x \leqslant c - t/2; \},$$

s - угол наклона.

При $t = 0$ получаем треугольную функцию.

Дефазификатор

Трансформировать *нечеткое множество* $\mu(y) = \mu_{A \longrightarrow B}(y)$ в точечное решение y можно многими способами:

1. Дефазификация относительно центра области

$$y_c = \int \mu(y) \cdot y \cdot dy / \int \mu(y) dy$$

или

$$y_c = \sum_i \mu(y_i) \cdot y_i / \sum_i \mu(y_i)$$

2. Дефазификация относительно среднего центра

$$y_c = \sum_{i=1,M} \mu(c_i) \cdot c_i / \sum_{i=1,M} \mu(c_i)$$

где c_i - центр i -го нечеткого правила,

$\mu(c_i)$ - соответствующая *функция* принадлежности.

3. Дефазификация относительно среднего максимума

$$y_M = \sum_{i=1,m} y_i / m,$$

где m - количество точек, в которых $\mu(y_i)$ достигает максимального значения. Если *функция* $\mu(y)$ имеет максимальное *значение* только в одной точке, то

$$y_M = y_m ax.$$

4. выбирается минимальное из максимальных значений y :

y_s - наименьшее из y , для которых $\mu(y) = max$.

5. выбирается максимальное из максимальных значений:

y_l - наибольшее из y , для которых $\mu(y) = max$.

Модель Мамдани-Заде как универсальный аппроксиматор

Модели нечеткого вывода позволяют описать выходной сигнал многомерного процесса как нелинейную функцию входных переменных x_i, $i = 1, 2, \ldots, N$ и параметров нечеткой системы, например, при использовании в качестве агрегатора оператора алгебраического произведения с последующей дефазификацией относительно среднего центра. В модели Мамдани-Заде каждое из M правил определяется уровнем активации условия

$$\mu(y_i) = \prod_{j=1}^{M} \mu_{Ai}(x_j)$$

где y_i - *значение* y, при котором *значение* $\mu(y_i)$ максимально. Пусть y_i — центр C_i нечеткого *множества* заключения i -го правила вывода. Тогда дефазификация относительно среднего центра дает

$$y = (\sum_{i=1}^{M} C_i[\prod_{j=1}^{N} \mu_{Ai}(x_j)])/\sum_{i=1}^{M}\prod_{j=1}^{N} \mu_{Ai}(x_j)$$

Приведенные формулы модели Мамдани-Заде имеют модульную структуру, которая идеально подходит для системного представления в виде многослойной структуры, напоминающей структуру классических нейронных сетей. Такие сети мы будем называть нечеткими нейронными сетями. Характерной их особенностью является возможность использования нечетких правил вывода для расчета выходного сигнала. Обучение таких сетей сводится к расчету параметров функции фазификации.

Нечеткие сети TSK (Такаги-Сугено-Канга)

Схема вывода в модели TSK при использовании M правил и N переменных x_j имеет вид $(i = 1, 2, \ldots M)$

$$\text{if } (x_1 \text{ is } A_1^{(i)})\ (x_2 \text{ is } A_2^{(i)})\ \ldots\ (x_N \text{ is } A_N^{(i)})$$
$$\text{then } y_i = p_{i0} + \sum_{j=1}^{N} p_{ij} x_j.$$

Условие $(x_i \text{ is } A_i)$ реализуется функцией фазификации

$$\mu_A(xi) = 1/(1 + ((x_i - c_i)/\sigma_i)^{2bi}).$$

При M правилах агрегированный выходной результат сети имеет вид

$$y(x) = \sum_{i=1}^{M} w_i y_i(x) / \sum_{i=1}^{M}$$

$$y_i(x) = p_{i0} + \sum_{j=1}^{N} p_{ij} x_j. \quad (1)$$

Веса w_i интерпретируются как *значимость* компонентов $\mu_A^{(i)}(x)$. Тогда формуле (1) можно поставить в соответствие многослойную нейронную *сеть* рис. 3.

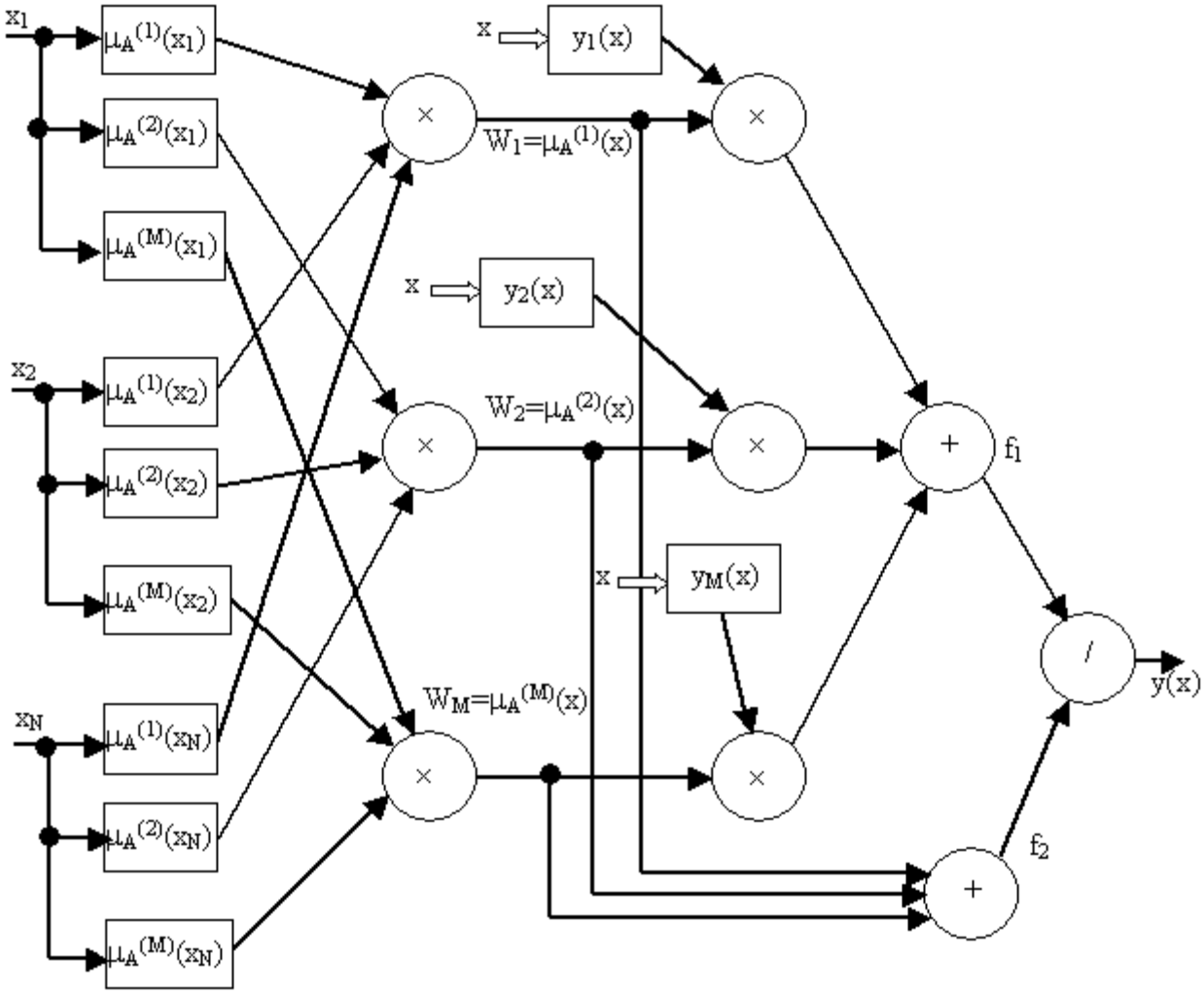


Рис. 3. Нечеткая нейронная сеть TSK

1. Первый слой выполняет фазификацию каждой переменной. Это параметрический слой с параметрами $c_j^{(i)}, \sigma_j^{(i)}, b_j^{(i)}$, подлежащими адаптации в процессе обучения.

2. Второй слой выполняет *агрегирование* отдельных переменных, определяя результирующее *значение* коэффициента принадлежности $w_i = \mu_A^i(x)$ для вектора x (непараметрический слой).

3. Третий слой - *генератор функции* TSK, рассчитывает значения

$$y_i(x) = p_{i0} + \sum_{j=1}^{N} p_{ij} x_j.$$

В этом слое также производится *умножение* $y_i(x)$ на w_i, сформированные в предыдущем слое. Здесь адаптации подлежат веса $p_{ij}, i = 1, 2, \ldots, M, j = 1, 2, \ldots, N$, определяющие функцию следствия модели TSK.

4. Четвертый слой составляют два нейрона-сумматора, один из которых рассчитывает взвешенную сумму сигналов $y_k(x)$, а второй - сумму весов $w_i, i = 1, 2, \ldots M$ (непараметрический слой).

5. Пятый слой из одного нейрона - это нормализующий слой, в котором выходной сигнал сети агрегируется по формуле (1).

Таким образом, в процессе обучения происходит уточнение параметров только первого (нелинейного) и третьего (линейного) слоев.

Гибридный алгоритм обучения нечетких сетей

Параметры, подлежащие адаптации, разделяются на две группы:

- первая состоит из параметров p_{ij} линейного третьего слоя;
- вторая состоит из параметров нелинейной функции принадлежности первого слоя.

Уточнение параметров проводится в два этапа.

На первом этапе при фиксации определенных значений параметров функции принадлежности путем *решения системы линейных уравнений* рассчитываются параметры p_{ij} полинома TSK.

При известных значениях функции принадлежности преобразование, реализуемое сетью, можно представить в виде

$$y(x) = \sum_{i=1}^{M} w_i (p_{i0} + \sum_{j=1}^{N} p_{ij} x_j).$$

$$w_i = [\prod_{j=1}^{N} \mu_A^{(i)}(x_j)] / \sum_{k=1}^{N} [\prod_{j=1}^{N} \mu_A^{(k)}(x_j)] = const.$$

При p обучающих выборках $(x^{(l)}, d^{(l)}), l = 1, 2, \ldots, p$ и замене выходного сигнала сети *ожидаемым значением* $d^{(l)}$ получим систему из p линейных уравнений вида

$$W \cdot P = d,$$

где

$$W = \begin{Vmatrix} w'_{11} & w'_{11}x_1^{(1)} & \dots & w'_{11}x_N^{(1)} & \dots & w'_{1M} & w'_{1M}x_1^{(1)} & \dots & w'_{1M}x_N^{(1)} \\ w'_{21} & w'_{21}x_1^{(2)} & \dots & w'_{21}x_N^{(2)} & \dots & w'_{2M} & w'_{2M}x_1^{(2)} & \dots & w'_{2M}x_N^{(2)} \\ & \dots & \dots & \dots & \dots & \dots & \dots & & \\ w'_{p1} & w'_{p1}x_1^{(p)} & \dots & w'_{p1}x_N^{(p)} & \dots & w'_{pM} & w'_{pM}x_1^{(p)} & \dots & w'_{pM}x_N^{(p)} \end{Vmatrix}$$

$$P = \|p_{10} \dots p_{1N} \dots p_{M0} \dots p_{MN}\|^T,$$

w'_{ki} - уровень активации (*вес*) i -го правила при предъявлении k -го входного вектора $x^{(k)}$.

Размерность матрицы W равна $p \times (N+1)M$, при этом обычно количество строк (количество выборок) значительно больше количества столбцов. Решение этой системы уравнений можно получить за один шаг при помощи псевдоинверсии матрицы W:

$$P = W^+ d.$$

Псевдоинверсия матрицы заключается в решении задачи минимизации

$$\min \|W^+ W - E\|,$$

где E - *единичная матрица*.

На втором этапе (линейные параметры $p_{ij}, i = 1, \dots M$ - фиксированы) рассчитываются фактические выходные сигналы y_k, $k = 1, 2, \dots, p$:

$$y = Wp,$$

вектор ошибки

$$\varepsilon = y - d,$$

и *градиент* целевой функции $E(n)$ по параметрам первого слоя. Если применяется *метод наискорейшего спуска*, то формулы адаптации принимают вид

$$c_j^{(i)}(n+1) = c_j^{(i)}(n) - \alpha_c \partial E(n)/\partial c_j^{(i)}$$
$$\sigma_j^{(i)}(n+1) = \sigma_j^{(i)}(n) - \alpha_\sigma \partial E(n)/\partial \sigma_j^{(i)}$$
$$b_j^{(i)}(n+1) = b_j^{(i)}(n) - \alpha_b \partial E(n)/\partial b_j^{(i)}$$

где n обозначает номер очередной итерации.

После уточнения нелинейных параметров вновь запускается процесс адаптации линейных параметров TSK (первый этап) и нелинейных параметров (второй этап). Этот цикл повторяется вплоть до стабилизации всех параметров процесса.

Мягкая экспертная система

Рассмотрим архитектуру и основные структурно-функциональные решения мягкой экспертной системы (МЭС). Для определения МЭС сопоставим понятия нечеткой и мягкой экспертных систем. В описании архитектуры МЭС будем использовать три признака: способ *извлечения знаний*; *представление* знаний; обработку знаний. Перечисленные признаки создают общую "координатную" сетку описания.

Определение мягкой экспертной системы. Сравнение нечеткой и мягкой экспертных систем

Нечеткие экспертные системы (ЭС) используют *представление* знаний в форме нечетких продукций и *лингвистических переменных*. Основу представления *лингвистической переменной* составляет *терм* с функцией принадлежности. Способ обработки знаний в нечетких ЭС - это *логический вывод* по нечетким продукциям. Особенностью нечеткой ЭС является способ извлечения функций принадлежности, который сводится либо к статистическим методам построения, либо к методу экспертных оценок. Мягкой ЭС (МЭС) будем называть нечеткую ЭС, которая обладает следующими особенностями:

- использует статистические данные, которые интерпретирует как обучающие выборки для нечетких нейронных сетей;
- представляет знания в виде *лингвистических переменных* (функций принадлежности - ФП), нечетких продукций и обученных нейронных сетей. Редукция множества нечетких продукций,настройка ФП и базы правил выполняется с помощью генетических алгоритмов (ГА).

Мягкими называют вычисления, сочетающие теорию нечетких систем, нейронные сети, вероятностные рассуждения и *генетические алгоритмы*, и обладающие синергическим эффектом; следовательно, мягкой экспертной системой называют ЭС, сочетающую перечисленные теории ради того же эффекта взаимного усиления.

Рассмотрим возможные применения МЭС в автоматизированном проектировании. Обобщенной *моделью проектирования* является иерархически-блочный метод, сущность которого сводится к декомпозиции функций с последующим выделением иерархий систем и подсистем. Проектируемая система формируется с помощью синтеза таких подсистем. *Анализ* в ходе автоматизированного проектирования обычно заключается в том, что необходимо рассмотреть условия эксплуатации будущей системы или ее окружения, которое является сложной системой (например, для экономических информационных систем окружающая среда - это социально-экономическая среда). Кроме анализа окружающей среды в ходе проектирования приходится выполнять *анализ* результатов физических или численных экспериментов и имитационного моделирования. Можно выделить два основных принципа экспертной деятельности в ходе проектирования.

1. Исходные данные для анализа представляются в виде качественного описания структурно-функционального решения и в виде совокупности временных рядов системных переменных окружения.

Принцип "конструктивной неопределенности" утверждает, что *точность* и смысл противоречат друг другу, начиная с некоторого момента анализа. Если в технике важными являются все более точные измерения, то в ходе анализа эксперт отказывается от точных цифр в

пользу нечетких, но содержательных оценок, которые осмыслены и позволяют принять проектное или управленческое решение.

Мягкая *экспертная система* должна предоставить инструментальную и информационную среду для экспертной деятельности в ходе проектирования. Инструменты для разработки МЭС должны представлять собой совокупность различных программных продуктов, объединенных логикой работы. Покажем, что МЭС, являющаяся инструментальной средой проектировщика, позволяет выполнить в автоматизированном режиме все этапы экспертной деятельности. Если рассматривать экспертную *деятельность* как управление объектом, то *инструментарий* экспертизы можно использовать как систему управления, а именно - нечеткий *контроллер*.

Представление знаний в мягкой экспертной системе. Содержание баз знаний и данных мягкой экспертной системы

Если использовать нечеткую НС на этапе *извлечения знаний*, то, кроме функций принадлежности и нечетких продукций, порождается совокупность обученных НС, которые входят в базу знаний МЭС. *Оптимизация* (*редукция*) *множества* извлеченных правил выполняется на основе генетического алгоритма.

База знаний МЭС должна содержать следующие части:

- функции принадлежности;
- нечеткие продукции;
- обученные нечеткие нейронные сети;
- процедуры интерпретации хромосом генетических алгоритмов;
- функции оптимальности.

Рассмотрим проблему представления перечисленных составных частей в компьютерных интеллектуальных системах. Если *функция* принадлежности характеризуется такими математическими свойствами, как непрерывность, выпуклость (унимодальность), то *функция* принадлежности может быть представлена параметризованной функцией формы. Наибольшее распространение получили четыре вида функций формы: треугольная, трапециевидная, колоколообразная и сигмоидальная, которые определяются тройкой, четверкой и двойкой параметров соответственно. Некоторые *операции* нечеткой алгебры сохраняют унимодальность при использовании трапециевидного представления функций принадлежности, поэтому результаты *операции* также являются четверкой параметров. *Представление* нечетких продукций упрощается в связи с тем, что порядок обработки нечетких продукций не важен и не влияет на ход вывода результата. *Представление* нечеткой нейронной сети является более сложной проблемой, так как описание структуры ННС не имеет смысла без нейроимитатора соответствующей архитектуры нечетких нейронных сетей, т.е. нейроимитатор определяется как составляющая часть механизма вывода мягкой ЭС. Для организации хранения знаний МЭС можно использовать как *СУБД*, так и специальные форматы.

Контрольные вопросы ЛР8(ПК-4):

1. Интеллектуальные информационные системы в условиях неопределенности и риска.
2. Нечеткие множества.
3. Лингвистические переменные.
4. Нечеткие правила вывода.
5. Системы нечеткого вывода Мамдани-Заде.
6. Фазификатор. Дефазификатор.

7. Модель Мамдани-Заде как универсальный аппроксиматор.
8. Распознавание рукописных текстов.
9. Состояние и тенденции развития искусственного интеллекта.
10. Успехи систем искусственного интеллекта и их причины
11. Моделирование бизнес-прецедентов.
12. Разработка модели бизнес-объектов.
13. Разработка концептуальной модели данных.
14. Разработка требований к системе.
15. Анализ требований и предварительное проектирование системы
16. Гибридный алгоритм обучения нечетких сетей.
17. Мягкая экспертная система.
18. Определение мягкой экспертной системы.
19. Сравнение нечеткой и мягкой экспертных систем.
20. Представление знаний в мягкой экспертной системе.
21. Содержание баз знаний и данных мягкой экспертной системы.
22. Синтезатор русской речи.
23. Язык формальной записи правил синтеза.
24. Интонационное обеспечение.
25. Аллофонная база данных. Лингвистический анализ.
26. Инструментарий синтеза русской речи.
27. Система распознавания речи.
28. Акустическая и лингвистическая модели.
29. Классификация систем распознавания. Системы машинного зрения.
30. Основные принципы или целостность восприятия.
31. Распознавание символов.
32. Состояние и тенденции развития искусственного интеллекта.
33. Успехи систем искусственного интеллекта и их причины

ЛИТЕРАТУРА

1. Агаханян Т.М., Плеханов С.П. Интегральные триггеры устройств автоматики. – М.: Машиностроение, 1978, - 368 с.
2. Шило В.Л. Популярные цифровые микросхемы: Справочник. – Челябинск: Металлургия, Челябинское отделение, 1988, - 352 с.
3. Микроэлектронные устройства автоматики: Учеб. Пособие для вузов / А. А. Сазонов, А. Ю. Лукичёв, В. Т. Николаев, и др./ Под ред. А. А. Сазонова. – М.: Энергоатомиздат, 1991. – 384 с.
4. Токхейм Р. Основы цифровой электроники: Пер. с англ. – М.: Мир, 1988. – 392 с.
5. Программа Electronic Workbench для анализа электронных схем: Учеб.- метод. пособие. В.М. Чухонцев. Самар. гос. тех. ун-т. Самара, 1999.- 92с.