УДК 004.4'2 + 004.032.26

Модульный программный нейрокомпьютер SNC: реализация и применение

Мисуно И.С., Рачковский Д.А., Ревунова Е.Г., Слипченко С.В., Соколов А.М., Тетерюк А.Е.

Приведено описание реализованных в программном нейрокомпьютере SNC блоков предобработки и классификации данных. (Описание концепции и архитектуры SNC см. в УСиМ N3 2004). SNC позволяет пользователю упростить создание произвольных алгоритмов путем их визуального конфигурирования. Приведены примеры применения SNC для серийных экспериментов классификации данных. Рассмотрены пути практического применения и дальнейшего развития.

Modular Software NeuroComputer SNC: implementation and application

Misuno I.S., Rachkovskij D.A., Revunova E.G., Slipchenko S.V., Sokolov A.M., Teteryuk A.E.

Description of implemented in Software NeuroComputer data pre-processing and classification blocks is given. Software NeuroComputer allows end users to visually configure arbitrary data processing algorithms with processing and control blocks. Examples of configuring SNC for real-valued vectors and handwritten digits classification are given with a discussion of practical applications and evolution of SNC.

Модульний програмний нейрокомп’ютер SNC: реалізація і застосування

Місуно І.С., Рачковський Д.А., Ревунова Е.Г., Сліпченко С.В., Соколов А.М., Тетерюк О.Е.

Наведено опис реалізованих в програмному нейрокомп’ютері SNC блоків передобробки і класифікації даних. (Опис концепції і архітектури SNC див. в УСиМ N3 2004). SNC дозволяє користувачу спростити створення довільних алгоритмів шляхом їх візуального конфігурування. Наведені приклади застосування SNC до серійних експериментів класифікації даних. Розглянуто шляхи практичного застосування і подальшого розвитку.

Модульный программный нейрокомпьютер SNC: реализация и применение

Мисуно И.С., Рачковский Д.А., Ревунова Е.Г., Слипченко С.В., Соколов А.М., Тетерюк А.Е.

Международный научно-учебный центр информационных технологий и систем
Киев, Украина

ВВЕДЕНИЕ

В настоящее время важнейшими задачами развития современных информационных технологий являются расширение сферы применения, с одной стороны, и расширение круга пользователей, с другой. Пути решения этих задач – интеллектуализация информационных технологий, а также унификация методов и средств их создания.

Интеллектуализация информационных технологий позволяет расширить сферу их применения за счет появляющихся возможностей использования в более сложных приложениях. Она же позволяет расширить круг пользователей, облегчая использование информационных технологий за счет возможности самостоятельного решения интеллектуальной системой рутинных задач и обеспечения более простого взаимодействия с пользователем. Унификация методов и средств создания интеллектуальных информационных технологий позволяет снизить трудоемкость их разработки, а также создает возможность интеграции систем различного применения.

Активизация исследований в области нейронных сетей и расширение сферы их применения привели к появлению в конце 80-х годов нейрокомпьютеров. В работах [1], [2] представлены концепция и архитектура программного нейрокомпьютера SNC, который разработан с учетом потребностей современных информационных технологий и предоставляет пользователю максимально дружественную среду визуального конфигурирования задач.

Основное назначение нейрокомпьютера SNC – создание, реализация и использование новых нейросетевых и других прогрессивных технологий обработки информации. Разработанная и реализованная архитектура SNC призвана облегчить этот процесс для потенциального пользователя. Такая архитектура обеспечивает высокую степень повторного использования разработанных компонент и позволяет гибко конфигурировать включенные в состав нейрокомпьютера блоки. Указанные возможности облегчают построение конфигураций *проектов* для решения существующих и новых задач обработки информации. *Проект SNC* – это совокупность обрабатывающих и управляющих блоков и связей между ними, созданная пользователем для решения некоторой задачи обработки данных. (см. также раздел 1 и [2]).

Благодаря гибкой модульной программной архитектуре, основанной на технологии COM, нейрокомпьютер позволяет легко создавать и использовать новые и существующие блоки. Унифицированные интерфейсы обрабатывающих блоков упрощают создание новых типов блоков и реализацию новых алгоритмов обработки информации. Функциональность блоков не ограничена только имеющимися интерфейсами и может быть расширена введением новых интерфейсов. Новые модули можно включать в состав SNC без необходимости перекомпиляции пакета. При этом разработчики получают доступ к реализованным в системе SNC блокам и вспомогательным механизмам, таким как удобная система конфигурирования, мощная поддержка баз данных, и возможность использования скриптовых технологий (JScript). Блоки реализуют нейросетевые алгоритмы обработки информации, методы предобработки информации, другие необходимые методы. Имеющиеся модули классификации реализуют как известные алгоритмы, так и алгоритмы, построенные на основе оригинальной нейросетевой парадигмы.

Визуальная среда конфигурирования проектов является отличительной особенностью нейрокомпьютера SNC. Используя визуальную среду, пользователь может комбинировать элементы конфигурации, визуально изменять порядок выполнения операций и направления потоков данных, а также производить индивидуальную настройку каждого элемента и конфигурации в целом. Указанные возможности обеспечивают удобство, гибкость и универсальность данной реализации нейрокомпьютера. Средства записи параметров и результатов в базу данных, построения отчетов и серийных запусков упрощают процесс проведения исследований и анализа результатов.

Предполагается использование SNC в качестве базового средства для решения актуальных задач интеллектуализации информационной обработки. К ним в первую очередь относятся задачи эффективного доступа и использования текстовой информации. Это обусловлено ростом объема электронных документов в компьютерных базах документов и Интернет и необходимостью извлекать релевантные знания из этой информации.

1 ОБРАБАТЫВАЮЩИЕ БЛОКИ

В рамках SNC реализован ряд обрабатывающих блоков, ориентированных на решение некоторых задач интеллектуальной обработки данных. В этом разделе описаны блоки классификации и предобработки данных, которые применяются для создании проектов и решения задач с использованием нейросетевых информационных технологий.

1.1 Блоки-форматы

"Форматами" называются блоки чтения-записи данных разных форматов.

Блок DgenFormat. Блок реализует считывание данных, полученных с помощью программы DataGen [3]. Программа DataGen генерирует и записывает в файл данные - векторы признаков с компонентами, нормированными к диапазону 0.0 - 1.0, и целочисленную метку класса. Можно задавать число признаков; число классов; сложность генерируемых областей; число образцов данных в классе, и другие параметры.

Блок FormatMNIST. Блок реализует чтение изображений из файлов базы данных MNIST [4]. Это наборы изображений рукописных цифр, собранные для исследований систем распознавания изображений символов. Предоставляемые данные используется последующими обрабатывающими блоками, работающими с полутоновыми изображениями (блоками предобработки LIRADistort, кодировщиком LiRA и др).

Блоки ImageSampleReader/ImageSampleWriter. Реализуют запись на диск и чтение изображений во внутреннем формате SNC. Применюется для записи результатов промежуточной обработки изображений и для дальнейшего доступа к ним других блоков. Обеспечивают простое побайтовое копирование данных изображения в файл и произвольный доступ к записанным данным.

Блоки WeightMatrixReader/WeightMatrixSaver. Реализуют запись на диск и чтение матриц связей нейронных сетей. Применяются совместно с блоком персептрона, позволяют использовать ранее обученную матрицу связей персептрона в других экспериментах.

Блоки LongBinaryVectorsReader/LongBinaryVectorsWriter. Реализуют запись на диск и чтение массивов многомерных бинарных векторов. Блоки оптимизированы для эффективной работы с большими массивами данных (порядка нескольких гигабайт). Такие массивы появляются в результате предобработки и кодирования больших обучающих выборок.

1.2 Блоки предобработки

Блок трансформации изображений LIRADistort. Реализует предварительную обработку входных изображений, позволяя искусственно увеличивать объем обучающей выборки добавлением трансформированных изображений. Текущая версия SNC реализует следующие типы трансформаций: MoveTransform – сдвиг изображения на заданное количество точек вдоль любой оси координат, SkewTransform – вертикальный скос изображения (эмулирующий наклон).

Трансформированные изображения предназначены для подачи на вход последующих обрабатывающих блоков – классификаторов. Параметры, величины и общее количество трансформаций задаются в параметрах блока, позволяя исследовать их влияние на результат классификации. Пример трансформированных изображений показан на Рис. 1, где а) – исходное изображение, б) – после MoveTransform, в) – после SkewTransform.

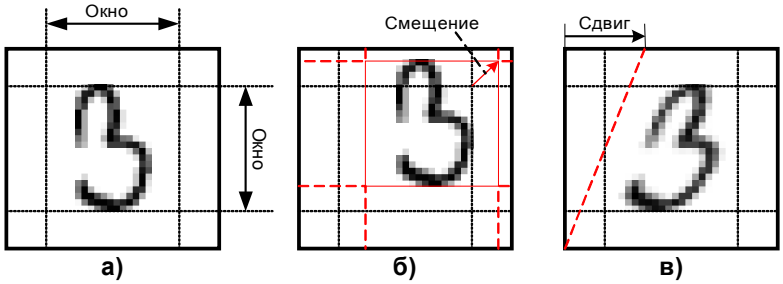


Рис. 1. Типы трансформаций исходного изображения

Блок бинаризации изображений ImageBinarization. Реализует преобразование полутонового изображения в бинарное. При преобразовании яркость пикселя полутонового изображения сравнивается со значением порога, и в зависимости от результата сравнения, пиксель выходного бинарного изображения устанавливается в 0 или 1. Порог определяется как среднее значение яркости всех пикселей исходного полутонового изображения, или устанавливаться пользователем.

Блоки-конверторы ImageToBinaryVector/ImageToDataSample. Реализуют преобразование бинарного/полутонового изображения в вектор с бинарными/целочисленными компонентами. При этом каждый пиксель с координатами (x,y) изображения отображается на y*M+x элемент вектора, где M – ширина изображения. Размерность полученного вектора равна количеству пикселей исходного изображения.

1.3 Блоки кодирования

Блок кодирования CodingRSC. Реализует кодирование входных данных с помощью алгоритма Random Subspace Coding [5], [6].

При кодировании A признаков, соответствующих входным атрибутам (вещественным компонентам входного вектора), преобразуются в N бинарных выходных признаков (Рис. 2). Бинарный признак устанавливается в 1 при попадании входной точки в его рецептивное поле - (гипер)прямоугольник в многомерном входном подпространстве размерностью S. Размеры и положение рецептивного поля выбираются случайно и затем фиксируются. Параметрами являются максимальный размер рецептивного поля G в одном измерении и его максимальная размерность S.

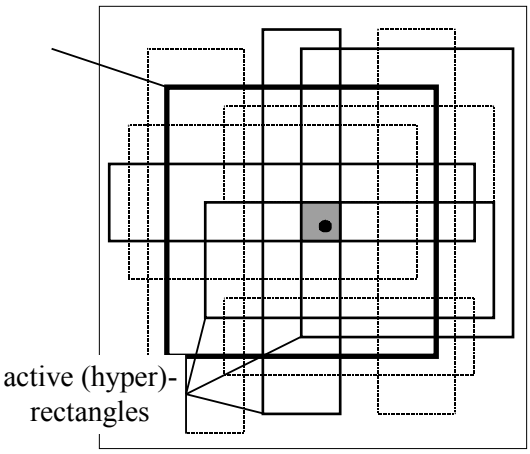


Рис. 2. Кодирование данных блоком RSC

Для блока CodigRSC задаются следующие параметры:

- S – размерность под-пространства рецептивных полей;
- G – размер рецептивного поля.

Блоки LIRACoding/LIRADecode. Блок **LIRACoding** осуществляет кодирование изображений, преобразовывая их в многомерные бинарные векторы. Алгоритм такого кодирования подробно описан в [7], [8].

Бит многомерного бинарного вектора ("вторичный признак") устанавливается в 1 при наличии и отсутствии единичных пикселей в определенных местах изображения - рецептивных точках (Рис. 3). Положения рецептивных точек для каждого вторичного признака выбираются случайно внутри рецептивного поля и затем фиксируются. Параметры кодирования - размер рецептивного поля, количество рецептивных точек в поле.

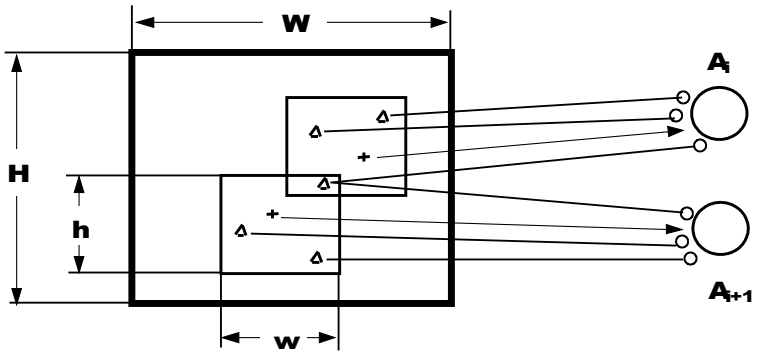


Рис. 3. Кодирование признаков входного изображения

Для блока кодирования LIRA задаются следующие параметры:

- MaskRows, MaskCols – размер окна признака (ширина w, высота h);
- MaskPositive, MaskNegative – количество «положительных» и «отрицательных» точек признаков;
- MaskBorder – минимальное расстояние от границы окна признака до границы изображения;

Блок **LIRADecode** реализует декодирование и визуализацию изображений, полученных в результате работы блока кодирования **LIRACoding**. При декодировании векторов "положительные" рецептивные точки признаков в графической консоли в виде красных точек, а "отрицательные" - в виде зеленых. Отсутствие рецептивных точек представлено синим цветом. В зависимости от количества признаков, активируемых некоторым пикселем, меняется его яркость при визуализации. Таким образом, можно зрительно оценить качество (полноту) кодирования изображений. На Рис. 4 представлен пример вывода блока, буква T обозначает действительный класс, R – распознанный, N – номер примера.

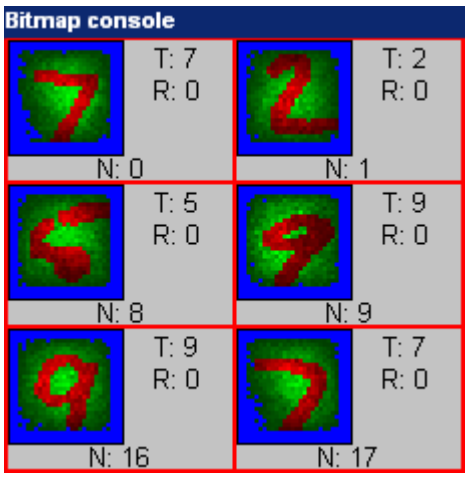


Рис. 4. Пример вывода LIRADecode

1.4 Блоки классификации

В описываемой версии SNC реализованы наиболее известные и эффективные алгоритмы классификации бинарных и вещественных данных: различные варианты персептрона, «*K* ближайших соседей» и SVM (Support Vector Machine). Все они в настоящее время активно применяются для решения практически важных задач распознавания образов.

Блоки Binary / Gradual Perceptron. Блоки реализуют алгоритмы классификации с помощью персептрона. Нейросетевая архитектура персептрона была предложена в конце 1950х годов [9]. Процедура обучения персептрона находит линейную разделяющую поверхность между двумя классами данных, если она существует. Бинарный и градуальный персептроны работают, соответственно, с входными данными - бинарными или вещественными векторами.

Данные поступают на входные нейроны. Входные нейроны связаны с выходными нейронами, которые соответствуют классам, с помощью матрицы связей. Результат классификации – вектор активации классов – определяется путем умножения вектора входных данных на матрицу связей.

Блок классификатора функционирует в двух основных режимах – обучения и тестирования (распознавания). Обучение блоков персептрона производится по обычному алгоритму обучения персептрона [10], [9], [11].

Номер класса определяется по выходному вектору активации классов, как индекс элемента с максимальным значением. Для персептрона в режиме обучения, веса связей при ошибке классификации изменяются следующим образом:

$w_{ij}(t+1) = w_{ij}(t) - \alpha x_i$, для неверно распознанного класса;

$w_{ik}(t+1) = w_{ik}(t) + \alpha x_i$, для истинного класса,

где (t) и $(t+1)$ – номера текущей и следующей итераций; α – коэффициент скорости обучения (обычно равен 1 для бинарного персептрона); x_i – значение сигнала на выходе i-го входного нейрона (для бинарного нейрона – 0 или 1); j и k – номера неверно распознанного и истинного классов соответственно.

Для персептронов задаются следующие параметры:

- Размер сети NetSize;
- Параметр D – «защитный коэффициент», уменьшающий активацию истинного класса (умножением на величину $1–D$), "затрудняя" обучение персептрона [6];
- Параметр α, определяющий скорость изменения весов связей градуального персептрона.

Сформированную в процессе обучения матрицу связей персептрона можно сохранить в файле для последующего использования. Персептрон может также использовать ранее обученную матрицу для распознавания.

Блоки Binary / Gradual KNN. Блоки реализуют метод ближайшего соседа, который является одним из самых известных алгоритмов классификации [12]. Алгоритм позволяет классифицировать произвольные данные с самыми сложными границами классов и шумом. Основным недостатком метода является его ресурсоемкость - большое время классификации и требуемый объем памяти.

В SNC реализованы две модификации блока: KNNBinary и KNNGradual, работающие с бинарными и вещественными векторами соответственно.

Блоки производят поиск K ближайших соседей к тестовому вектору среди всех векторов обучающей выборки. Если K=1, то результатом классификации тестового вектора считается класс ближайшего к нему вектора. Если K>1, то результатом классификации считается тот класс, представителей которого больше всего среди найденных K векторов (процедура «голосования»). Близость векторов определяются как косинус угла или Эвклидово расстояние между ними, или по другой метрике близости.

Блоки Binary / Gradual SVM. Реализуют классификатор на основе методов SVM (Support Vector Machines) [13].

Метод SVM получил широкое распространение в 1990х годах ([14], [15], [16], [17] и ссылки в этих работах), хотя его основы были заложены гораздо раньше. Метод позволяет классифицировать произвольные данные и является одним из самых точных универсальных методов классификации.

При экспериментах (раздел 2.4) варьировались стандартные параметры SVM: C и ε. Использовалось RBF-ядро с параметром Gamma.

1.5 Дополнительные блоки

Блок визуализации ImageViewer. Реализует визуализацию изображений в графической консоли. В основном используется для просмотра ошибочно распознанных изображений базы MNIST. Отображается изображение, номер изображения в базе данных (N), номер правильного класса изображения T (True), а также номер распознанного класса изображения R (Recognized) (Рис. 5). Номера изображений, которые необходимо отобразить, задаются как параметры блока в процессе конфигурирования, либо в процессе работы конфигурации.

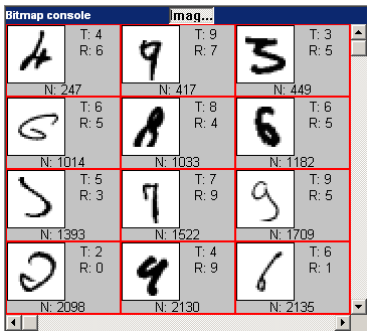


Рис. 5. Пример вывода ImageViewer

Блок оптимизации векторов LBVectorOptimizer. Блок реализует фильтрацию массивов бинарных признаков. При этом удаляются бинарные признаки, которые редко и/или часто встречаются в обучающей выборке. Пороговые значения фильтрации (минимальная и максимальная частота встречаемости) задаются пользователем в конфигурации проекта.

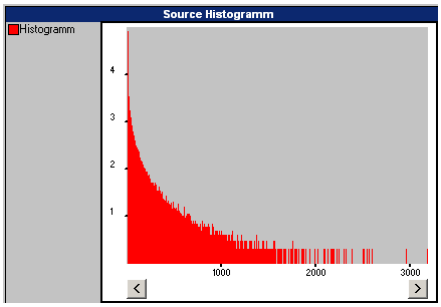


Рис. 6. Пример вывода LBVectorOptimizer

Блок статистики MatrixStatistics. Блок реализует подсчет и вывод статистической информации (в виде гистограммы распределения весов) матрицы связей персептрона. Полученная статистика отображается в виде диаграммы для каждого класса в ActiveX-консоли (MS Web Chart). Пример вывода блока представлен на Рис. 7, где по оси абсцисс выведены веса признаков, по оси ординат – частоты встречаемости признаков.

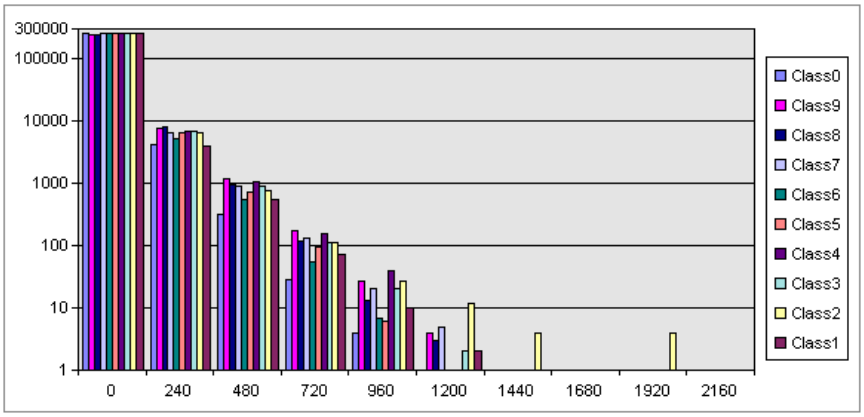


Рис. 7. Пример вывода MatrixStatistics

2 ПРИМЕРЫ ПРОЕКТОВ SNC: КЛАССИФИКАЦИЯ ДАННЫХ

SNC позволяет создавать проекты для решения задач классификации данных - отнесения данных к одному из заданного набора классов. Обучение классификации производится с использованием обучающей выборки, где имеются образцы данных с известной принадлежностью к классу. В качестве примера использования SNC для решения задач классификации рассмотрены проекты для классификации векторов вещественных признаков и изображений рукописных символов.

2.1 Проекты классификации векторов вещественных признаков

В этом типе задач данные представлены в виде набора признаков. Набор признаков формирует вектор, компоненты которого являются вещественными числами - значениями признака. С помощью проектов SNC проводилось экспериментальное сравнение разных алгоритмов классификации на стандартных тестовых наборах вещественных данных. Исследовались широко распространенные классификаторы: К ближайших соседей, метод опорных векторов (SVM), персептроны, а также оригинальные нейросетевые классификаторы RSC/RTC. Использовались известные тестовые выборки с нелинейными границами классов – "две спирали" (DoubleSpiral) [18], [19], и "задача Леонарда-Крамера" (Leonard-Kramer) [20].

Классификатор RTC/RSC. В SNC реализован оригинальный алгоритм классификации, основанный на "кодировании со случайными порогами" - RTC/RSC [6], [5] и его модификациях. Такой способ кодирования данных позволяет создавать классификаторы с высоким быстродействием и качеством распознавания и работать с данными, имеющими сложные границы классов.

В рамках реализованной в SNC модели классификатор RTC/RSC состоит из двух объектов – кодировщика RSC и собственно классификатора, в роли которого выступает бинарный персептрон. Вместе они составляют классификатор с тремя нейронными слоями: первый (входной) слой - *A* нейронов, соответствующих количеству атрибутов входных данных (элементов вектора); второй (промежуточный) слой - *N* двоичных нейронов; и третий (выходной) слой - *C* нейронов, соответствующих классам.

В конфигурации проекта участвуют следующие объекты (Рис. 8): DGENFormat, CodingRSC, BinaryPerceptron, WeightMatrixSaver.

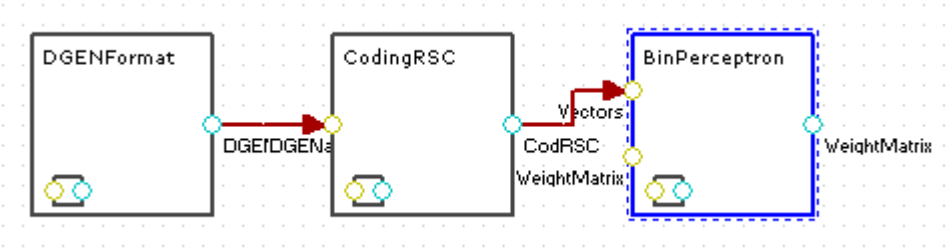


Рис. 8. Конфигурация проекта классификатора RSC

На вход классификатора подаются обучающая и тестирующая выборки в виде векторов с вещественными компонентами. Векторы представлены в формате программы DataGen [3] и находятся во внешних файлах.

Блок DGENFormat производит чтение данных из файла. С его выхода данные поступают на вход блока CodingRSC, где подвергаются кодированию и преобразуются в многомерные бинарные векторы. Для кодирования входных данных должны быть установлены следующие параметры: количество N бинарных "вторичных" признаков, параметр G (максимальный размер рецептивного поля) кодирования RSC [5], и некоторые дополнительные параметры - тип и начальное значение генератора случайных чисел, и др.

Закодированные данные, представленные в виде бинарных векторов, подаются на вход блока персептрона BinaryPerceptron. Персептрон работает в одном из двух режимов – обучения или тестирования. Переход в нужный режим осуществляется с помощью двух управляющих блоков control_Processing, присутствующих в блоке персептрона BinaryPerceptron. Матрица связей, с которой оперирует персептрон, периодически сохраняется в файл на диске с помощью объекта WeightMatrixSaver.

В режиме обучения на вход классификатора многократно подается обучающая выборка. Блок BinaryPerceptron изменяет матрицу связей персептрона в соответствии с алгоритмом обучения. Ход процесса обучения контролируется классификацией тестовой выборки. В параметрах проекта классификатора задается количество проходов обучения, после которого выполняется тестирование. В описываемых далее экспериментах этот параметр равен 1 – после каждого прохода обучающей выборки проводился контроль результатов на тестовой выборке.

Другие проекты классификаторов для векторов признаков с вещественными компонентами:

- ***С использованием классификаторов бинарных векторов.*** Реализация других типов классификаторов возможна путем замены блока классификатора бинарных векторов в проекте, описанном выше. Общая схема проекта при этом не меняется. В SNC реализован и испытан ряд блоков классификации бинарных векторов: BinaryPerceptron, BinaryKNN (метод ближайшего соседа), BinarySVM (метод Support Vector Machines).
- ***Без использования классификаторов бинарных векторов.*** Более традиционная схема классификации не использует предварительного выделения бинарных признаков из входной информации. Классификаторы работают непосредственно с входными данными. При этом проекты состоят из блока-формата для чтения входных данных и собственно блока классификатора (Рис. 9). Реализован и испытан ряд блоков классификации векторов с вещественными числами: GradualPerceptron, GradualKNN,

GradualSVM. Как и в случае бинарных классификаторов с кодированием, блоки классификации являются взаимозаменяемыми.

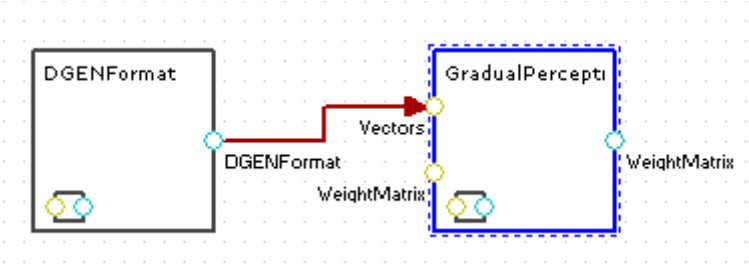
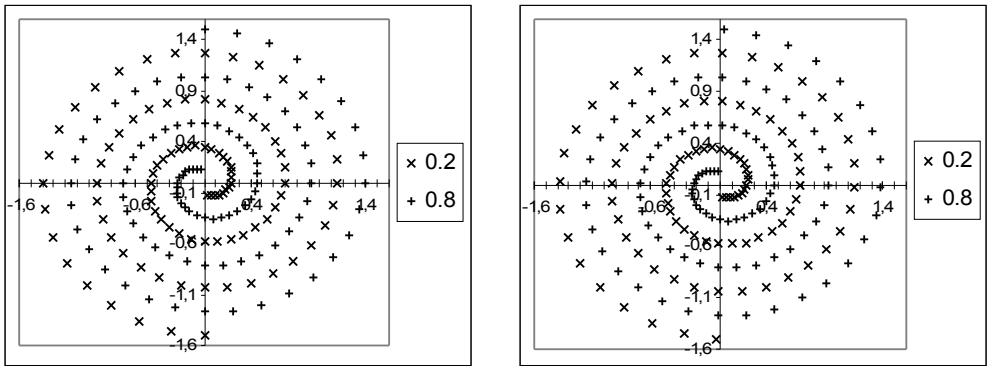


Рис. 9. Конфигурация проекта однослойного персептрона.

2.2 Тестовые задачи

DoubleSpiral (исходная формулировка). Эта известная тестовая задача [19], [21] является классическим примером задачи нелинейной классификации. В таких задачах классы входных данных не являются линейно разделимыми в исходном пространстве.

В экспериментах по решению задачи DoubleSpiral (traditional) использованы две независимые выборки данных – обучающая и тестовая. Каждая из выборок содержит по 192 образца данных. В графическом виде исходные данные представлены на Рис. 10.

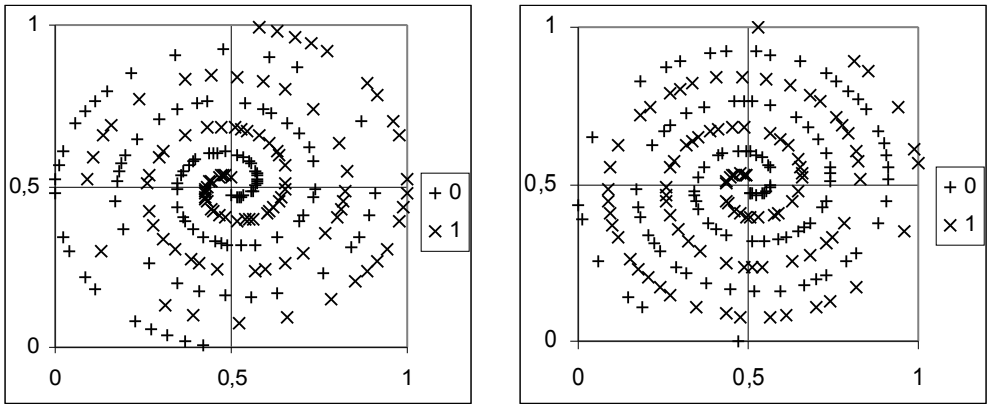


а) Обучающая выборка б) Тестирующая выборка

Рис. 10. Исходные данные DoubleSpiral (traditional)

- Применение классификатора **RTC/RSC.** При экспериментальном исследовании точности и скорости классификации варьировались следующие параметры: размер сети (NetSize); параметр G кодировщика RSC; параметр D бинарного персептрона (Defense Coefficient). Постоянным параметром кодировщика RSC являлось количество подпространств каждого нейрона входного слоя S=2.
- Применение классификатора **Gradual KNN.** Тестирование метода «*K* ближайших соседей» производилось с разным числом соседей (*K*), и разными мерами сходства между векторами: Евклидово расстояние (Euclidean); сумма модулей разности (“Manhattan”). В качестве "опорных точек" использовалась вся обучающая выборка.
- Применение классификатора **Binary KNN** с кодированием RSC**.** Проводилось тестирование классификатора «*K* ближайших соседей» для бинарных векторов, полученных кодированием исходных данных методом RSC. Изменяемые параметры – число бинарных признаков NetSize, максимальный размер рецептивного поля в одном измерении G, количество соседей K. Мера сходства между бинарными векторами определялась как косинус угла (Cosine).

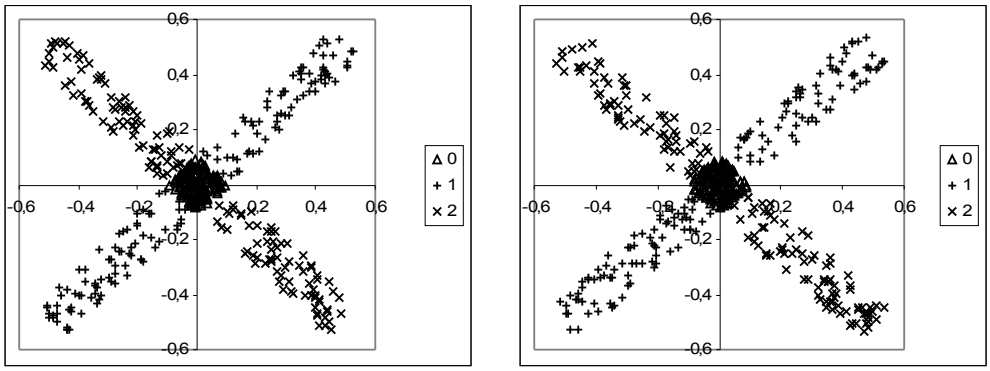
DoubleSpiral (случайное разбиение). Данные для этой задачи получены путем случайного разбиения 384 точек исходной спирали на две выборки – обучающую и тестирующую, как предложено в [21]. Таким образом, некоторые точки обучающей и тестирующей выборок не находятся в непосредственной близости друг от друга. Исходные данные для этой задачи представлены на Рис. 11.



а) Обучающая выборка б) Тестирующая выборка

Рис. 11. Исходные данные для задачи DoubleSpiral (случайное разбиение)

Задача Leonard-Kramer. Эта задача также является известной тестовой задачей нелинейной классификации. В обучающей и тестовой выборках имеется по 450 образцов данных (Рис. 12).



а) Обучающая выборка б) Тестирующая выборка

Рис. 12. Исходные данные задачи Leonard-Kramer

2.3 Результаты экспериментальных исследований на тестовых задачах

В Табл. 1 представлены результаты применения реализованных в SNC схем классификации RTC/RSC, Gradual KNN и Binary KNN с кодированием RSC для решения трех описанных выше тестовых задач. Там же приведены результаты работы других классификаторов и известных нейропакетов при решении этих задач ([21], [22]).

Табл. 1. Результаты работы классификаторов, реализованных в SNC.

	Задача		
Классификатор	**DS (traditional)**	**DS (random)**	**LK**
	Количество ошибок распознавания, %		
RSC + Bin Perceptron	0	2,08	0
RSC + Binary KNN	0	2,08	0,67
Gradual KNN	0	4,27	0,39
Cascade Backprop	-	5,7	-
Cascade associative	-	7,3	-
Kernel associative	-	5,8	-
NeuroLand	-	-	1,33
NeuralWorks	-	-	2
BrainMaker	-	-	19,17
	Время обучения (кол-во проходов обучения):		

RSC + Bin Perceptron	100 мс - 20 с	100 мс - 20 с	250 мс - 60 с
Cascade Backprop	-	35 с	-
Cascade associative	-	1,1 с	-
Kernel associative	-	0,46 с	-
NeuroLand	-	-	7 (400)
NeuralWorks	-	-	15 (250)
BrainMaker	-	-	600 (5500)
	Время распознавания		
RSC + Bin Perceptron	10 мс - 200 мс	10 мс - 200 мс	25 мс - 500 мс
RSC + Binary KNN	10 мс - 13 с	10 мс - 13 с	25 мс - 60 с
Gradual KNN	~10 мс	~10 мс	~25 мс

Сравнение результатов работы разных классификаторов на описанных трех тестовых задачах показывает, что лучшие результаты по точности распознавания, скорости обучения, и по скорости классификации получены с помощью классификаторов RSC и Gradual KNN.

2.4 Проекты классификация изображений рукописных символов

С помощью проектов SNC проводилось экспериментальное сравнение наиболее известных алгоритмов классификации на стандартном наборе изображений рукописных цифр (база MNIST [23]). Исследовались оригинальные нейросетевые классификаторы LiRA и известные классификаторы других типов: *K* ближайших соседей, метод опорных векторов (SVM), персептроны.

Проект классификатора LiRA. Классификатор по алгоритму LIRA [8], [24] является разновидностью классификатора RSC для случая бинарных изображений. Он также может рассматриваться как развитие персептрона Розенблатта [25]. Обучение нейронов выходного слоя реализовано по правилу обучения простого однослойного персептрона с помощью блока BinaryPerceptron.

Реализованный в рамках SNC проект классификатора LiRA (Рис. 13) содержит следующие блоки: FormatMNIST, ImageBinarization, LIRADistort, LIRACoding, BinaryBerceptron, WeightMatrixSaver.

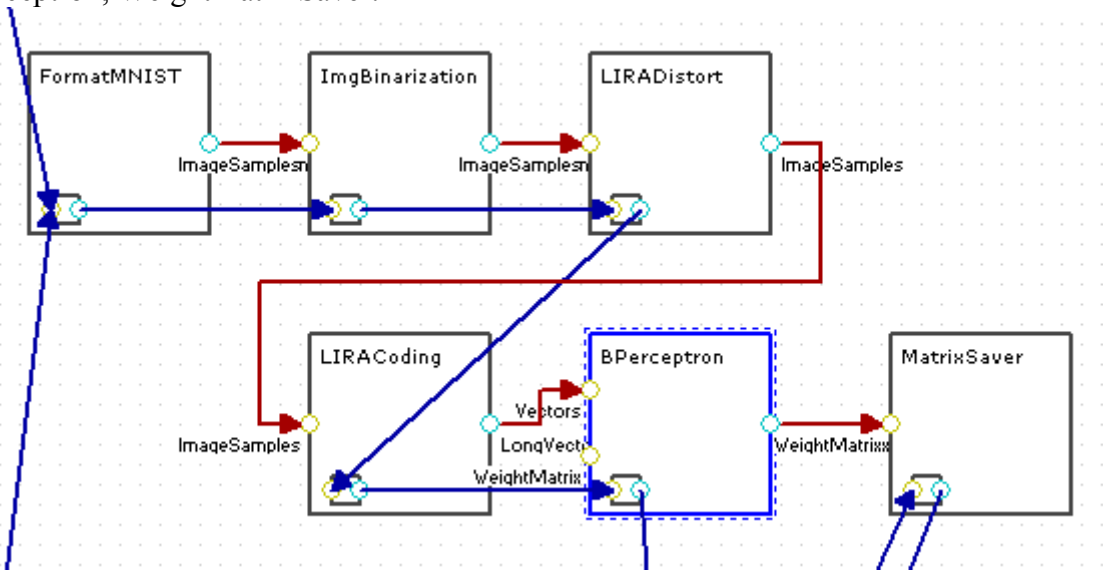


Рис. 13. Конфигурация проекта классификатора LiRA

Входные данные читаются из файлов базы MNIST с помощью объекта FormatMNIST. Прочитанные изображения бинаризируются с помощью блока ImageBinarization. Бинаризованные изображения подаются на вход блока LIRADistort, где

к ним применяются трансформации – перемещения и сдвиги. Количество трансформаций N_{dist} задается параметрами проекта (при N_{dist} = 0 трансформации отсутствуют). При проведении экспериментов с трансформациями на вход классификатора подаются исходные изображения и несколько версий трансформированных изображений. Тип и параметры трансформаций специфичны для каждого номера трансформации. Далее изображения кодируются блоком LIRACoding, и сформированные многомерные бинарные векторы поступают на вход блока BinaryPerceptron, реализующего процедуру обучения и классификации.

Обучение производится с разными параметрами (количеством нейронов промежуточного слоя и параметрами кодировщика LiRA), в течение максимум 40 проходов обучающей выборки, после каждой из которых качество обучения проверяется на тестовой выборке. Матрица связей персептрона периодически сохраняется в файл с помощью блока WeightMatrixSaver.

Другие проекты классификаторов рукописных цифр. Реализованные в SNC блоки классификаторов также использованы для задачи распознавания полутоновых изображений. Возможность такого применения достигается в SNC путем замены в конфигурации проекта блока персептрона блоками соответствующих классификаторов. Были созданы и протестированы ряд проектов классификаторов с применением бинарного кодирования LiRA. В ряде проектов полутоновые изображения базы MNIST непосредственно подавались на входы блоков классификации Perceptron, KNN, SVM.

Результаты распознавания рукописных символов базы MNIST. Классификация полутоновых изображений производилась на базе MNIST [23]. База содержит набор из 60000 обучающих и 10000 тестовых примеров полутоновых изображений рукописных цифр размером 28x28 пикселей.

В Табл. 2 приведены результаты классификации (процент ошибок), времена обучения и распознавания классификатора Binary LiRa на 10000 образцов тестирующей части базы MNIST.

Табл. 2. Результаты работы классификатора LiRA на базе MNIST

Ndist	**Размер сети**		
	128k	**256k**	**512k**
Ошибки классификации, %			
0	1,57	1,45	1,33
4	1,06	1,07	0,92
16	0,85	0,70	0,65
Время обучения			
0	313 с	537 с	1065 с
4	0.43 ч	1.4 ч	2.7 ч
16	3.3 ч	9.7 ч	11.4 ч
Время распознавания			
–	2 с	3.9 с	11 с

В Табл. 3 приведены результаты распознавания (процент ошибок), времена обучения и распознавания на обучающей и тестирующей выборках базы MNIST с помощью классификатора SVM (раздел 1.4).

Табл. 3. Результаты работы классификатора SVM на базе MNIST

Ndist	**Epsilon**	**Gamma**	**C**	**Результат, %**	**Время обучения, ч**	**Время распозн., с**
0	0.001	0,00000047	5	**1,41**	5,4	
3	0.001	0,0000004768	4	**1,07**	15	
4	0.001	0,000000476837	5	**0,96**	27,3	150
6	0.001	0,000000476837	5	**0,96**	49,2	
7	0.001	0,000000476837	5	**0,85**	51,5	

Как видно из Табл. 2, количество ошибок распознавания уменьшалось с увеличением количества N нейронов второго слоя. Например, количество ошибок E на 10000 тестовых примерах было E=1150 при N=4000; E=330 при N=32000; и E=175 при N=256000 (см. также [25]). Время обучения при N=256000 составляло менее 10 мин на 2,500-МГц процессоре.

Некоторое уменьшение количества ошибок было достигнуто за счет увеличения параметра D персептрона. Наилучший результат без расширения обучающей выборки трансформациями входных изображений составил 133 ошибки при N=512000.

Для дальнейшего уменьшения количества ошибок использовался расширенный обучающий набор, полученный с помощью добавления трансформированных версий оригинальных изображений в базе данных. Трансформации производились с помощью объекта LIRADistort, и осуществлялись сдвигами и наклонами исходного изображения. С помощью обучения на таком расширенном тестовом наборе были получены результаты в 65 ошибок на 10000 тестовых примерах, что является одним из лучших результатов, показанных в мире на базе MNIST [4].

Исследования по распознаванию рукописных символов базы MNIST с помощью алгоритма SVM (без предварительной бинаризации и перекодирования изображений) показали хорошую динамику улучшения результатов с увеличением числа искажений. Однако не удалось испытать алгоритм на большем числе искажений из-за того, что для используемой версия алгоритмов не хватило имеющегося объема памяти 4GB.

ЗАКЛЮЧЕНИЕ

Текущая версия программного нейрокомпьютера SNC использовалась для создания и отладки оригинальных нейросетевых классификаторов на основе алгоритмов RSC и LiRA. Были созданы конфигурации проектов для проведения одиночных и масштабных серийных экспериментов с применением этих, а также других известных классификаторов. Это позволило провести сравнение работы различных алгоритмов классификации на различных типах данных.

Реализованные алгоритмы использовались для решения задач классификации данных – рукописных цифр и искусственно сгенерированных векторов признаков. Созданные прототипы систем классификации показали результаты на уровне лучших известных алгоритмов, превосходя их по ряду показателей (скорость обучения и работы, точность распознавания). Программный нейрокомпьютер SNC показал себя достаточно удобным в работе, позволяя создавать и выполнять масштабные серийные эксперименты с классификаторами различных типов и получать результаты в виде информативных специализированных отчетов.

Таким образом, разработанный и реализованный программный нейрокомпьютер SNC представляет собой эффективное средство для создания нейросетевых технологий решения разноплановых прикладных задач. Это достигнуто использованием оригинальных нейросетевых архитектур, а также реализацией нейросетевых блоков обработки информации и других, используемых в системе блоков, в виде COM модулей. Системная архитектура SNC позволяет гибко конфигурировать нейрокомпьютер для решения требуемых задач, а также легко включать в его состав новые блоки, разработанные с учетом системных требований.

Текущая версия системы рассчитана на квалифицированных пользователей, поскольку создание проекта требует знания не только специфики решаемой задачи, но и программной структуры SNC и программирования на JScript. Поэтому важной задачей является упрощение использования SNC для пользователей, в частности, за счет автоматизации составления нейросетевых конфигураций для типовых задач – например, включением шаблонных конфигураций, и.т.п.

Дальнейшее развитие SNC предполагается в следующих основных направлениях:

- реализация новых алгоритмов обработки информации;
- расширение функциональности и набора блоков в рамках существующей архитектуры;
- перепроектирование некоторых системных решений.

В более отдаленной перспективе планируется:

- реализация и поддержка распределенных вычислений;
- переход к кросс-платформенной реализации;
- включение в состав SNC средств поддержки решения задач информационной обработки.

БЛАГОДАРНОСТИ

Авторы выражают благодарности Л.М.Касаткиной и А.М.Касаткину за обсуждения рукописи этой статьи, Алексею Попову за участие в реализации разработки, Михаилу Куссулю и Дмитрию Новицкому - за консультации. Работа частично поддерживалась за счет финансирования по договору с МПП № 92289/48-2001 и № 92211/07-42-00.

1. *Misuno I.S., Rachkovskij D.A., Revunova E.G., Sokolov A.M.* SNC: The Software Neurocomputer With Modular Architecture. // *Проблемы Нейрокибернетики*, 2002. – 2 – P. 109-113.
2. *Гриценко В.И., Мисуно И.С., Рачковский Д.А., Ревунова Е.Г., Слипченко С.В., Соколов А.М.* Концепция и архитектура программного нейрокомпьютера SNC. // *УСиМ*, 2004 – N 3 – С. … (в печати)
3. *Rachkovskij D.A., Kussul E.M.* DataGen: a generator of datasets for evaluation of classification algorithms. // *Pattern Recognition Letters*, 1998. - P. 537-544.
4. *LeCun Y.* The MNIST database of handwritten digits. – http://yann.lecun.com/exdb/mnist/
5. *Kussul E.M., Rachkovskij D.A., Wunsch D.* The Random Subspace coarse coding scheme for real-valued vectors. // *International Joint Conference on Neural Networks*, 1999. - P. 6.
6. *Kussul E.M., Baidyk T.N., Lukovich V.V., Rachkovskij D.A.* Adaptive High Performance Classifier Based on Random Threshold Neurons. // *Twelfth European Meeting on Cybernetics and Systems Research (EMCSR- 94)* - Singapore: World Sci. Publ, 1994. - P. 1687-1694.
7. *Kussul E.M., Baidyk T.N.* Improved Method of Handwritten Digit Recognition. - UNAM, Centro de Instrumentos, 2001.
8. *Куссуль Э.М., Касаткина Л.М., Лукович В.В.* Нейросетевые классификаторы для распознавания рукописных символов. // *Управляющие системы и машины (УСиМ)*, 1999. - P. 77-86.
9. *Rosenblatt F.* The perceptron: A probabilistic model for information storage and organization in the brain. // *Psychological Review*, 1958. - N 65 – P. 386-408.
10. *Widrow B., Hoff M. E.* Adaptive switching circuits. - IRE, 1960. - P. 96-104.
11. *Розенблатт, Ф.* Принципы нейродинамики (персептроны и теория механизмов мозга). - М.: "Мир", 1965.
12. *Дуда Р., Харт П.* Распознавание образов и анализ сцен. - М., «Мир», 1967. – 512 с.
13. *Библиотека* libsvm. http://www.csie.ntu.edu.tw/~cjlin/libsvm/
14. *Vapnik V.* The Nature of Statistical Learning Theory. - New York: Springer-Verlag, 1995.
15. *Schoelkopf B.* Support Vector Learning. - Munich, Germany: Oldenbourg-Verlag, 1997.
16. *Vapnik V.* Statistical Learning Theory. - New York: Wiley, 1998.
17. *Schoelkopf B., Burges C., Smola A.* Advances in Kernel Methods - Support Vector Learning. - Cambridge, MA: MIT Press, 1999.
18. *Lang K. J., Witbrock M. J.* Learning to Tell Two Spirals Apart. // *1988 Connectionist Models Summer School*, 1988.

19. *Fahlman S. E.* Faster-Learning Variations on Back-Propagation: An Empirical Study. // *1988 Connectionist Models Summer School*, 1988.
20. *Leonard J., Kramer M., Ungar L.* Using Radial Basis Functions to Approximate a Function and Its Error Bounds. // *IEEE Transactions on Neural Networks*, vol. 3, 1992. - P. 624-627.
21. *Новицкий Д.* Ассоциативная память на основе ядерных сетей. // (в печати) – 2003.
22. *Резник А.М., Калина Е.А., Садовая Е.Г., Дехтяренко А.К., Сичев А.С., Галинская А.А.* Многофункциональный нейрокомпьютер NeuroLand // Математичні машини і системи. – 2003. – N 1. – С. 36-45.
23. *LeCun Y., Bottou L., Bengio Y., Haffner P.* Gradient-based Learning Applied to Document Recognition. // *IEEE*, 1998, vol. 86. - P. 2278-2344.
24. *Kussul E.M., Baidyk T.N., Rachkovskij D.A.* Application of Neural Network Classifiers for the OCR of Printed Texts. // *Second International Symposium on Neuroinformatics and Neurocomputers*, 1995. - P. 1-6.
25. *Kussul E.M., Baidyk T.N., Kasatkina L.M., Lukovich V.V.* Neural network system for continuous handwritten words recognition. // *Intern. Joint Conference on Neural Networks*, 2001.