А.А.ШАРИПБАЕВ, М.Х.КАРАБАЛАЕВА

ПРИМЕНЕНИЕ МНОГОСЛОЙНОЙ НЕЙРОННОЙ СЕТИ ОБРАТНОГО РАСПРОСТРАНЕНИЯ ДЛЯ РАСПОЗНАВАНИЯ РЕЧИ

1. Целесообразность использования нейронных сетей для распознавания речи

Какие проблемы возникают при построении системы распознавания речи? Главная особенность речевого сигнала в том, что он очень сильно варьируется по многим параметрам: длительность, темп, высота голоса, искажения, вносимые большой изменчивостью голосового тракта человека, различными эмоциональными состояниями диктора, сильным различием голосов разных людей. Два временных представления одного и того же фрагмента речи даже для одного и того же человека, записанные в разное время, не будут совпадать. Необходимо искать такие параметры речевого сигнала, которые полностью описывали бы его (т.е. позволяли бы отличить один звук речи от другого), но были бы в какой-то мере инвариантны относительно указанных выше вариаций речи. Полученные таким образом параметры должны затем сравниваться с образцами, причем это должно быть не простое сравнение на совпадение, а поиск наибольшего соответствия. Это вынуждает искать нужную форму расстояния в найденном параметрическом пространстве.

Далее, объем информации, которую может хранить система, не безграничен. Каким образом запомнить практически бесконечное число

вариаций речевых сигналов? Очевидно, здесь не обойтись без какой-либо формы статистического усреднения.

Ещё одна проблема – это скорость поиска в базе данных. Чем больше её размер, тем медленнее будет производиться поиск – это утверждение верно для всех последовательных вычислительных машин.

В поисках аппарата, способного разрешить вышеперечисленные проблемы, мы обратимся к нейронным сетям.

Классификация – это одна из «любимых» для нейросетей задач. Причем нейросеть может выполнять классификацию даже при обучении без учителя (правда, при этом образующиеся классы не имеют смысла, но ничто не мешает в дальнейшем ассоциировать их с другими классами, представляющими другой тип информации – фактически наделить их смыслом). Любой речевой сигнал можно представить как вектор в какомлибо параметрическом пространстве, затем этот вектор может быть запомнен в нейросети. Нейросетевые алгоритмы обладают способностью к статистическому усреднению, т.е. решается проблема с вариативностью речи. Кроме того, они осуществляют параллельную обработку информации, т.е. одновременно работают все нейроны. Тем самым решается проблема со скоростью распознавания – обычно время работы нейросети составляет несколько итераций.

Далее, на основе нейросетей легко строятся иерархические многоуровневые структуры, при этом сохраняется их прозрачность (возможность их раздельного анализа). Так как фактически речь является составной, т.е. разбивается на фразы, слова, буквы, звуки, то было бы логично и систему распознавания речи строить иерархическую.

Наконец, ещё одним важным и перспективным свойством нейронных сетей является гибкость архитектуры. Под этим термином мы подразумеваем, что фактически алгоритм работы нейросети определяется её архитектурой. Автоматическое создание алгоритмов – это мечта уже нескольких десятилетий. Но создание алгоритмов на языках программирования пока под силу только человеку. Конечно, созданы специальные языки, позволяющие выполнять автоматическую генерацию алгоритмов, но и они не намного упрощают эту задачу. А для нейросети генерация нового алгоритма достигается простым изменением ее архитектуры. При этом возможно получить совершенно новое решение задачи. Установив правила модификации нейросети и корректное правило отбора, определяющее, лучше или хуже новая нейросеть справляется с задачей, можно в конце концов получить нейросеть, которая решит задачу верно. Все нейросетевые модели, объединенные такой парадигмой, образуют множество генетических алгоритмов. Таким образом, существует возможность создания таких нейросетей, которые не были изучены исследователями или не поддаются аналитическому изучению но, тем не менее успешно справляются с задачей.

Таким образом, задача распознавания речи может быть решена при помощи нейронных сетей[1].

2. Выбор структуры нейронной сети

Эффективность нейронных вычислений проистекает от соединений нейронов в сетях. Широкий круг задач, решаемых нейросетями, не позволяет в настоящее время создавать универсальные сети, вынуждая разрабатывать специализированные нейросети, функционирующие по различным алгоритмам.

Выбор структуры нейросети осуществляется в соответствии с особенностями и сложностью задачи. Для решения некоторых отдельных типов задач уже существуют оптимальные, на сегодняшний день, конфигурации. Если же задача не может быть сведена ни к одному из известных типов, разработчику приходится решать сложную проблему синтеза новой конфигурации. При этом следует руководствоваться несколькими основополагающими принципами:

a) возможности сети возрастают с увеличением числа ячеек сети, плотности связей между ними и числом выделенных слоев;

b) введение обратных связей наряду с увеличением возможностей сети поднимает вопрос о динамической устойчивости сети;

c) сложность алгоритмов функционирования сети (в т.ч., например, введение нескольких типов синапсов – возбуждающих, тормозящих и др.) также способствует усилению мощи нейросети.

Вопрос о необходимых и достаточных свойствах нейросети для решения того или иного рода задач представляет собой целое направление

нейрокомпьютерной науки. Так как проблема синтеза нейросети сильно зависит от решаемой задачи, получить общие подробные рекомендации затруднительно. В большинстве случаев оптимальный вариант получается на основе интуитивного подбора. Единственное жесткое требование, предъявляемое архитектурой к элементам сети, это соответствие размерности вектора входных сигналов числу ее входов [2].

Рассмотрим свойства и особенности некоторых наиболее распространенных конфигураций.

Сети прямого распространения (одно- и многослойные перцептроны) – представляют интерес и широко используются в качестве классификаторов. При правильном проектировании позволяют эффективно проводить классификацию входных неявно выраженных состояний, объединенных в нечеткие множества. Отсутствие обратных связей гарантирует устойчивость таких сетей.

Сети с обратными связями (рекуррентные) – имеют более широкие возможности. Однако обучение систем с большим числом обратных связей может стать нестабильным и занять непозволительно долгое время

Популярной разновидностью нейросетей с обратными связями являются ***самоорганизующиеся карты Кохонена***. Они успешно применяются для распознавания речи, обработки изображений, в робототехнике и задачах управления. Отличительной особенностью этих сетей является то, что при их обучении используется метод обучения без учителя, то есть результат обучения зависит только от структуры входных данных.

Стоит уделить отдельное внимание двум основным парадигмам обучения нейронных сетей: «с учителем» и «без учителя».

Когда идет речь об использовании нейросетей и нейросетевых алгоритмов, обычно подразумеваются определенные процедуры их обучения. Нейросеть представляет собой адаптивную систему, жизненный цикл которой состоит из двух независимых фаз – обучения и работы сети. Обучение считается законченным, когда сеть правильно выполняет преобразование на тестовых примерах и дальнейшее обучение не вызывает значительного изменения настраиваемых весовых коэффициентов. Далее сеть выполняет преобразование ранее неизвестных ей данных на основе сформированной ею в процессе обучения нелинейной модели процесса. Сеть успешно работает до тех пор, пока существенно не изменится реальная модель отображаемого явления (например, в случае возникновения ситуации, информация о которой никогда не предъявлялась сети при обучении). После этого сеть может быть дообучена с учетом новой информации, причем при дообучении предыдущая информация не теряется, а обобщается с вновь поступившей. При «повреждении» части весовых коэффициентов нейросети ее свойства могут быть полностью восстановлены в процессе дообучения.

От того, насколько качественно будет выполнен этап обучения нейросети, зависит способность сети решать поставленные перед ней проблемы во время эксплуатации.

Обучение с учителем предполагает, что для каждого входного вектора существует целевой вектор, представляющий собой требуемый выход. Вместе они называются обучающей парой. Сеть обучается на некотором числе таких обучающих пар.

Когда в сети только один слой, алгоритм ее обучения с учителем довольно очевиден, так как правильные выходные состояния нейронов единственного слоя заведомо известны, и подстройка синаптических связей идет в направлении, минимизирующем ошибку на выходе сети.

В многослойных же сетях оптимальные выходные значения нейронов всех слоев, кроме последнего, как правило, неизвестны, и такую сеть уже невозможно обучить, руководствуясь только величинами ошибок на выходах этой сети. Наиболее широко используемым вариантом решения этой проблемы является алгоритм обратного распространения – когда сигналы ошибки распространяются от выходов нейросети к ее входам, в направлении, обратном прямому распространению сигналов в обычном режиме работы. Отметим, что этот алгоритм не согласуется с данными нейробиологии, где показано, что сигналы в биологических нейронных сетях могут распространяться только в одном, прямом направлении.

Обучение без учителя является более правдоподобной моделью обучения в биологической системе. Процесс обучения заключается в подстраивании весов синапсов, пока выходные значения сети не стабилизируются с заданной точностью. Некоторые алгоритмы предусматрива-

ют также изменение и структуры сети, т.е. количества нейронов и их взаимосвязи (самоорганизация). Результат обучения зависит только от структуры входных данных. Поэтому нейросети данного типа удобно применять для восстановления пропусков в данных, для анализа данных и поиска закономерностей. Подстраиваемые значения синапсов фактически не ограничены.

Обучение без учителя гораздо более чувствительно к выбору оптимальных параметров, чем обучение с учителем. Его качество сильно зависит от начальных весовых величин синапсов, характера изменения коэффициента обучения и др. Поэтому потребуется провести предварительную работу по подбору оптимальных параметров. Но несмотря на сложности реализации, алгоритмы обучения без учителя находят успешное применение [2].

Описанные нейронные сети автоматически приобретают знания. Но процесс их обучения зачастую проходит достаточно медленно, а анализ обученной сети весьма сложен (как правило, обученная сеть – черный ящик для пользователя). При этом какую-либо априорную информацию (знания эксперта) для ускорения процесса обучения в нейросеть ввести невозможно.

Попытки устранить эти свойственные нейросетям недостатки, используя в полной мере их преимущества, привели к возникновению аппарата ***нечетких нейронных (гибридных) сетей***. В них выводы делаются на основе аппарата нечеткой логики, но соответствующие функции принадлежности подстраиваются с использованием алгоритмов обучения нейросетей. Такие системы используют априорную информацию, могут приобретать новые знания и для пользователя являются логически прозрачными [3].

3. Нейронные сети обратного распространения

Рассмотрим подробнее структуру искусственных нейронных сетей и их применение в конкретных задачах [4].

Нейронные сети – самообучающиеся системы, имитирующие деятельность человеческого мозга. Несмотря на большое разнообразие вариантов нейронных сетей все они имеют общие черты. Так все они, также как и мозг человека, состоят из большого числа однотипных элементов – нейронов, которые имитируют нейроны головного мозга, связанных между собой. На рис.1 показана схема нейрона.

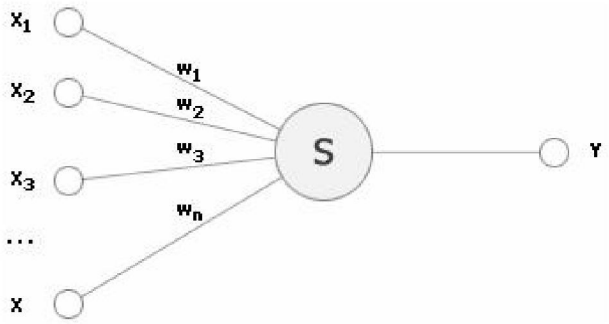

Рис.1

Из рисунка видно, что искусственный нейрон, так же как и живой, состоит из синапсов (w_n), связывающих входы нейрона (x_i) с ядром, ядра нейрона (S), которое осуществляет обработку входных сигналов, и аксона (Y), который связывает нейрон с нейронами следующего слоя. Каждый синапс имеет вес, который определяет, насколько соответствующий вход нейрона влияет на его состояние. Состояние нейрона определяется по формуле

$$S = \sum_{i=1}^{n} x_i w_i , \qquad (1)$$

где n – число входов нейрона, x_i – значение i-го входа нейрона, w_i – вес i-го синапса.

Затем определяется значение аксона нейрона по формуле

$$Y = f(S), \qquad (2)$$

где f – некоторая функция, называемая активационной. Наиболее часто в качестве активационной функции используется сигмоид

$$f(x) = \frac{1}{1+e^{-\alpha x}} . \qquad (3)$$

Основное достоинство этой функции в том, что она дифференцируема на всей оси абсцисс и имеет очень простую производную

$$f'(x) = f(x)\ (1-f(x)) \qquad (4)$$

При уменьшении параметра сигмоид становится более пологим, вырождаясь в горизонтальную линию на уровне 0,5 при = 0. При увеличении сигмоид все больше приближается к функции единичного скачка.

Нейронные сети обратного распространения – это мощнейший инструмент поиска закономерностей, прогнозирования, качественного

анализа. Такое название – сети обратного распространения (back propagation) они получили из-за используемого алгоритма обучения, в котором ошибка распространяется от выходного слоя к входному, т.е. в направлении, противоположном направлению распространения сигнала при нормальном функционировании сети.

Нейронная сеть обратного распространения состоит из нескольких слоев нейронов, причем каждый нейрон слоя i связан с каждым нейроном слоя $i+1$, т.е. речь идет о полносвязной нейронной сети.

В общем случае задача обучения нейросети сводится к нахождению некой функциональной зависимости $Y=F(X)$, где X – вектор входной, а Y – выходной. В общем случае такая задача, при ограниченном наборе входных данных, имеет бесконечное множество решений. Для ограничения пространства поиска при обучении ставится задача минимизации целевой функции ошибки нейросети, которая находится по методу наименьших квадратов:

$$E(w) = \frac{1}{2}\sum_{j=1}^{p}(y_j - d_j)^2, \qquad (5)$$

где y_j – значение j-го выхода нейросети, d_j – целевое значение j-го выхода, p – число нейронов в выходном слое.

Обучение нейросети производится методом градиентного спуска, т.е. на каждой итерации изменение веса производится по формуле

$$\Delta w_{ij} = -\eta \cdot \frac{\partial E}{\partial w_{ij}}, \qquad (6)$$

где – параметр, определяющий скорость обучения.

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_j}{dS_j} \cdot \frac{\partial S_j}{\partial w_{ij}}, \qquad (7)$$

где y_j – значение выхода j-го нейрона, S_j – взвешенная сумма входных сигналов, определяемая по формуле (1).

При этом множитель

$$\frac{\partial S_j}{\partial w_{ij}} \equiv x_i, \qquad (8)$$

где x_i – значение i-го входа нейрона.

Далее рассмотрим определение первого множителя формулы (7):

$$\frac{\partial E}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{dS_k} \cdot \frac{\partial S_k}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{dS_k} \cdot w_{jk}^{(n+1)}, \qquad (9)$$

где k – число нейронов в слое $n+1$.

Введем вспомогательную переменную

$$\delta_j^{(n)} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_j}{dS_j}. \qquad (10)$$

Тогда мы сможем определить рекурсивную формулу для определения $\delta_j^{(n)}$ n-го слоя, если нам известно $\delta_k^{(n+1)}$ следующего $(n+1)$-го слоя.

$$\delta_j^{(n)} = \left[\sum_k \delta_k^{n+1} \cdot w_{jk}^{n+1}\right] \cdot \frac{dy_j}{dS_j} \qquad (11)$$

Нахождение же $\delta_j^{(n)}$ для последнего слоя нейросети не представляет трудности, так как нам известен целевой вектор, т.е. вектор тех значений, которые должна выдавать нейросеть при данном наборе входных значений.

$$\delta_j^{(N)} = (y_i^{(N)} - d_i) \cdot \frac{dy_i}{dS_i} \qquad (12)$$

И, наконец, запишем формулу (6) в раскрытом виде:

$$\Delta w_{ij}^{(n)} = -\eta \cdot \delta_j^{(n)} \cdot x_i^n \qquad (13)$$

Рассмотрим теперь полный алгоритм обучения нейросети:

1. Подать на вход нейросети один из требуемых образов и определить значения выходов нейронов нейросети.

2. Рассчитать $\delta^{(N)}$ для выходного слоя нейросети по формуле (12) и рассчитать изменения весов $\Delta w_{ij}^{(N)}$ выходного слоя N по формуле (13).

3. Рассчитать по формулам (11) и (13) соответственно $\delta^{(n)}$ и $\Delta w_{ij}^{(n)}$ для остальных слоев нейросети, $n = N-1, \ldots, 1$.

4. Скорректировать все веса нейросети:

$$w_{ij}^{(n)}(t) = w_{ij}^{(n)}(t-1) + \Delta w_{ij}^{(n)}(t). \qquad (14)$$

5. Если ошибка существенна, то перейти на шаг 1.

На следующем этапе сети поочередно в случайном порядке предъявляются вектора из обучающей последовательности.

Простейший метод градиентного спуска, рассмотренный выше, очень неэффективен в случае, когда производные по различным весам сильно отличаются. Это соответствует ситуации, когда значение функции S для некоторых нейронов близко по модулю к 1 или когда модуль некоторых весов много больше 1. В этом случае для плавного уменьшения ошибки надо выбирать очень маленькую скорость обучения, но при этом обучение может занять непозволительно много времени.

Простейшим методом усовершенствования градиентного спуска является введение момента β, когда влияние градиента на изменение весов изменяется со временем. Тогда формула (13) примет вид

$$\Delta w_{ij}^{(n)}(t) = -\eta \cdot \delta_j^{(n)} \cdot x_i^n + \mu \Delta w_{ij}^{(n)}(t-1). \quad (13.1)$$

Дополнительным преимуществом от введения момента является способность алгоритма преодолевать мелкие локальные минимумы [4].

4. Представление входных данных

Основное отличие нейронных сетей в том, что в них все входные и выходные параметры представлены в виде чисел с плавающей точкой обычно в диапазоне [0..1]. В тоже время данные предметной области часто имеют другое кодирование. Это могут быть числа в произвольном диапазоне, даты, символьные строки. Таким образом, данные о предметной области могут быть как количественными, так и качественными. Рассмотрим сначала преобразование качественных данных в числовые, а затем рассмотрим способ преобразования входных данных в требуемый диапазон [4].

Качественные данные можно разделить на две группы: упорядоченные и неупорядоченные. Рассмотрим задачу о прогнозировании успешности лечения какого-либо заболевания. Примером упорядоченных данных могут являться данные, например, о дополнительных факторах риска при данном заболевании:

Опасность каждого фактора возрастает в таблицах при движении слева направо.

В первом случае мы видим, что у больного может быть несколько факторов риска одновременно. Значит, нам необходимо использовать такое кодирование, при котором отсутствует ситуация, когда разным комбинациям факторов соответствует одно и то же значение. Наиболее распространен способ кодирования, когда каждому фактору ставится в соответствие разряд двоичного числа. «1» в этом разряде говорит о наличии фактора, а «0» — о его отсутствии. Очевидно, для представления всех факторов достаточно 4-разрядного двоичного числа. Например, число $1010_2 = 10_{10}$ означает наличие у больного гипертонии и употребления алкоголя, а числу 0000_2 соответствует отсутствие у больного факторов риска. Таким образом, факторы риска будут представлены числами в диапазоне [0..15].

Во втором случае мы также можем кодировать все возрастные значения двоичными весами, но это будет нецелесообразно, т.к. набор возможных значений будет слишком неравномерным. В этом случае более правильным будет установка в соответствие каждому значению своего веса, отличающегося на 1 от веса соседнего значения. Так число 3 будет соответствовать возрасту 50-59 лет. Таким образом, возраст будет закодирован числами в диапазоне [0..4].

Аналогично можно поступать и для неупорядоченных данных, поставив в соответствие каждому значению какое-либо число. Однако это вводит нежелательную упорядоченность, которая может исказить данные и сильно затруднить процесс обучения. В качестве одного из способов решения этой проблемы можно предложить поставить в соответствие каждому значению один из входов нейросети. В этом случае при наличии этого значения соответствующий ему вход устанавливается в 1, в противном случае — в 0.. К сожалению, данный способ не является панацеей, т.к. при большом количестве вариантов входных значений число входов нейросети разрастается до огромного количества. Это резко увеличит затраты времени на обучение. В этом случае можно использовать несколько иное

Нет	Ожирение	Алкоголь	Курение	Гипертония
Другой пример — возраст больного:				
До 25 лет	25-39 лет	40-49 лет	50-59 лет	60 лет и старше

решение. В соответствие каждому значению входного параметра ставится бинарный вектор, каждый разряд которого соответствует отдельному входу нейросети. Например, если число возможных значений параметра 128, то можно использовать 7-разрядный вектор. 1-му значению будет соответствовать вектор {0000000}, 128-му – вектор {1111111}, а, например, 26-му значению – {0011011}. Тогда число требуемых для кодирования параметров входов можно определить как

$$N = \log_2 n, \qquad (14)$$

где n — количество значений параметра, N — количество входов.

Для нейросети необходимо чтобы входные данные лежали в диапазоне [0..1], в то время как данные предметной области могут лежать в любом диапазоне. Предположим, что данные по одному из параметров лежат в диапазоне [Min..Max]. Тогда наиболее простым способом нормирования будет

$$\tilde{x} = \frac{x - Min}{Max - Min}, \qquad (15)$$

где x — исходное значение параметра, $\tilde{x}$ — значение, подаваемое на вход НС.

К сожалению, этот способ кодирования не лишен недостатков. Так в случае если $Max >> \tilde{x}$, то распределение данных на входе может принять вид, как на рис.2.

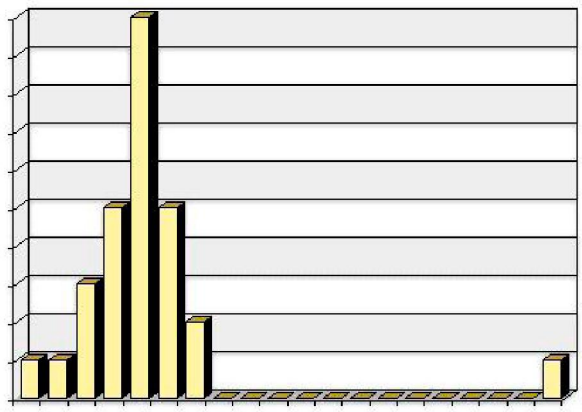

Рис.2

Т.е. распределение входных параметров будет крайне неравномерным, что приведет к ухудшению качества обучения. Поэтому в подобных ситуациях, а также в случае, когда входные данные лежат в диапазоне [0,], можно использовать нормировку с помощью функции вида

$$\tilde{x} = \frac{1}{1 + e^{-x}}. \qquad (16)$$

5. Построение многослойной нейронной сети обратного распространения для решения задачи распознавания речи

При построении многослойной нейронной сети обратного распространения для решения задачи распознавания речи особое внимание следует уделить структуре входных данных. Очевидно, что на вход подобной системы должен подаваться речевой сигнал в его цифровом представлении (например, частотный спектр звука). Следовательно, первым этапом создания нейросетевого распознавателя речи является формирование звуковой базы.

В случае распознавания слитной речи целесообразно использовать отдельные фонемы для формирования обучающей выборки нейросети, а целые слова и предложения — для тестовой.

Возьмем файл с записью речи диктора, начитывающего какой-либо текст. Вооружимся программой вроде Sonic Foundry Sound Forge 6.0, просмотрим и прослушаем звуковой файл в поисках таких фрагментов речи, где те или иные фонемы наиболее ярко выражены (рис.3). Сохраним эти фонемы в отдельные файлы.

Помимо фонем, также сохраним в отдельные звуковые файлы целые слова, произносимые диктором. Множество нарезанных таким способом «кусочков» и составят звуковую базу системы.

Затем преобразуем звук в число, т.е. приведем элементы нашей звуковой базы к виду, удобному для потребления нейросетью. Каждый звуковой фрагмент, каждая фонема должны быть представлены в виде набора числовых данных, с приемлемой точностью описывающего исходный сигнал. Можно использовать для этого, например, спектральный анализ. Тогда амплитуды гармоник спектра послужат входными параметрами нейросети. (Проведение спектрального анализа цифрового сигнала выходит за рамки данной статьи, для получения информации по этому вопросу можно обратиться, например, к работе [5].)

Теперь займемся формированием на основе звуковой базы собственно обучающей выборки будущей многослойной нейронной сети обратного распространения.

Фонема «А» во временном представлении

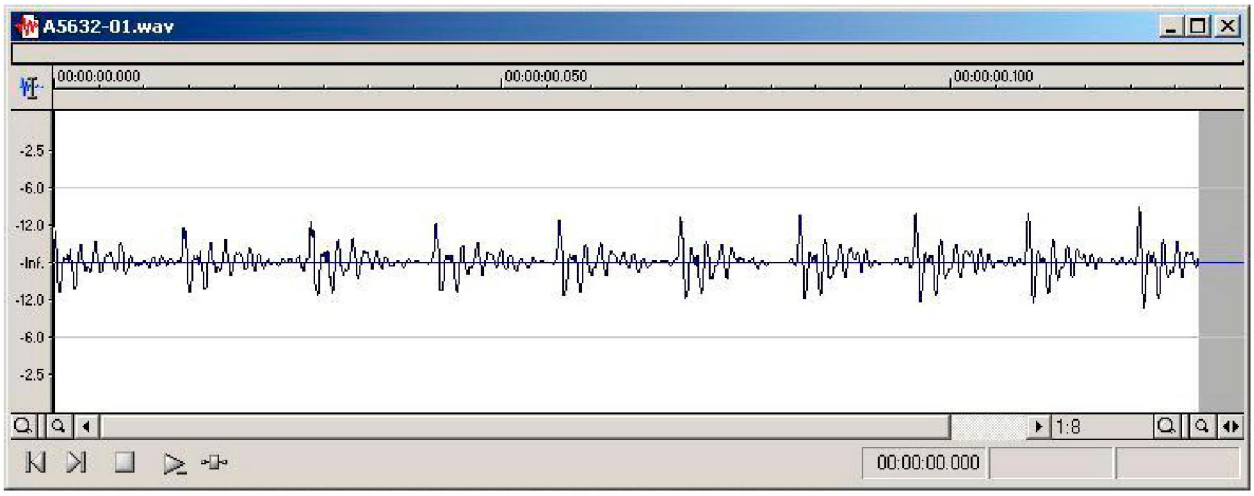

Рис.3

Фонема «А» в числовом представлении

66 A512-01.wav

iID	cName	dMaxA
3268	Отрезок1	4,2216940442

iID	iFrequency	iAmplitude
163400	86,1328125	0,7430103159
163401	172,265625	1,378564123
163402	258,3984375	1,1240167032
163403	344,53125	0,9984864322
163404	430,6640625	1,0211314805
163405	516,796875	1,2547260214
163406	602,9296875	2,2496768785
163407	689,0625	4,0641973854
163408	775,1953125	4,2216940442
163409	861,328125	1,8495932501
163410	947,4609375	1,0986734925
163411	1033,59375	1,8977002185
163412	1119,7265625	2,9624924518
163413	1205,859375	2,4971362277

Рис. 4

К подготовке данных для нейросети нужно подходить очень серьезно. От этого зависит 90% успеха. Количество примеров должно быть достаточно большим. При этом необходимо, чтобы выборка была репрезентативной и непротиворечивой.

Если просмотреть таблицы спектральных частот и амплитуд, привлекает внимание тот факт, что частоты за рубежом 4000 Гц становятся малоинформативными — их амплитуды близки к нулю. Поэтому чтобы не засорять нейросеть неинформативными данными, можно ограничить количество гармоник спектра первыми несколькими десятками.

Неравномерное распределение амплитуд, свойственное спектрограмме речевого сигнала, может привести к ухудшению качества обучения. Во избежание этого целесообразно подвергнуть спектр сжатию в пространстве амплитуд, используя формулу для перевода отношения в логарифмическую шкалу [6]:

$$X_{new} = 20 * lg\,(X / X_N), \qquad (17)$$

где X_{new} — амплитуда в новом сжатом амплитудном пространстве, X — текущее значение амплитуды, X_N — номинальное значение.

В качестве номинального значения можно использовать, например, максимальную для данной спектрограммы амплитуду.

Полученные новые значения X_{new} нормируем в диапазоне [0..1] и сохраним в базе данных. Это и есть входные данные для нашей нейросети.

В подобной предобработке нуждаются и выходные данные.

Так как мы используем парадигму обучения с учителем, то для каждого входного вектора мы должны указать целевой вектор, представляющий собой требуемый выход. В случае распознавания слитной речи, на вход нейросети мы подаем количественные данные (числовые значения амплитуд), а на выходе ожидаем получить символьное значение (распознанный звук). Нейросеть умеет оперировать только числами, следовательно, мы должны представить символьную информацию в числовом виде, т.е. кодировать.

Если кодировать каждый звук его порядковым номером: «А» = 1, «Е» = 2, «О» = 3, и т.д., – это окажется неэффективным, так как даже небольшие ошибки нейросети будут приводить к сильному искажению результата.

Можно попробовать увеличить расстояние между символами: «А» = 0, «Е» = 40, «О» = 80, и т.д., а результат определять по расстоянию между значением на выходе нейросети и кодом символа. Например, если нейросеть выдала результат 50, то это звук «Е». На практике этот подход оказался достаточно плодотворным и показал прекрасные результаты для пяти символов. Однако с увеличением количества распознаваемых символов привнесение нежелательной упорядоченности выходных данных стало затруднять процесс обучения и искажать результат.

Для решения этой проблемы увеличим число нейронов в выходном слое, с тем чтобы назначить каждому символу один выходной нейрон. Если совместить этот подход с идеей увеличения расстояния между объектами, то получится примерно следующее решение: «А» = {100; 0; 0; 0; 0; …}, «Е» = {0; 100; 0; 0; 0; …}, «О» = {0; 0; 100; 0; 0; …}, и т.д.

Подготовив обучающую выборку, можно конфигурировать нейросеть и запускать ее обуче-

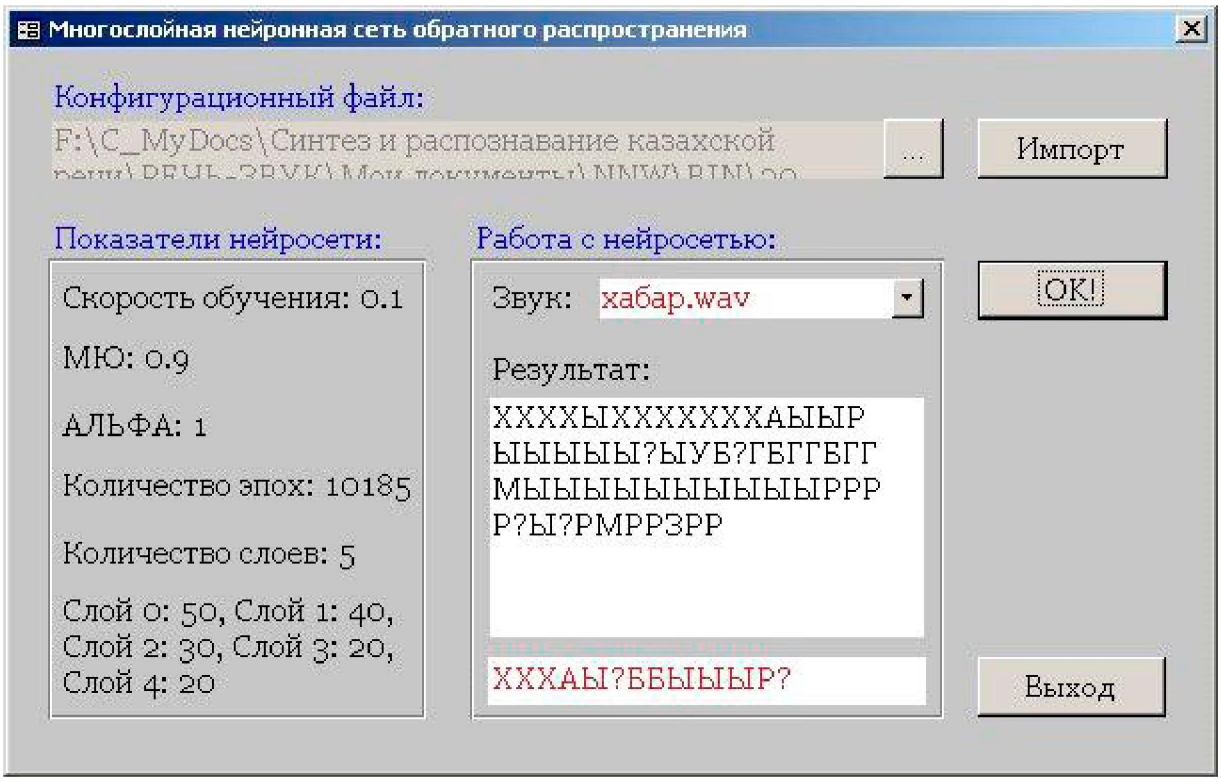

Рис.5

Результаты тестирования нейросети приведены в таблице.

		Распознано																					Всего испытаний	Успех, %	С учетом родственных звуков
		А	Е	О	Ұ	Ы	Б	Х	Г	К	С	З	Р	Л	Ш	Й	Ж	Т	Н	ң	М	?			
Предложено	А	7				19		1					2								1	8	38	18	68
	Е		9											1		1						3	14	64	71
	О			6	8																		14	43	100
	Ұ				2								1									2	5	40	40
	Ы					11																5	16	69	69
	Б						3		3													3	9	33	67
	Х							4															4	100	100
	Г								2													4	6	33	33
	К				1					2												4	7	29	29
	С										2	2										5	9	22	44
	З											4											4	100	100
	Р					1							12	1								6	20	60	60
	Л													9							2	2	13	69	69
	Ш															7							7	100	100
	Й		1													7							8	88	100
	Ж																4					1	5	80	80
	Т								1					1			6					9	17	35	35
	Н												1				10				2	7	20	50	50
	ң				1			2														7	10	0	20
	М																				3	2	5	60	60

ние. Под конфигурацией нейросети в данном случае понимается определение количества ее слоев и количества нейронов в каждом слое.

6. Реализация и тестирование

Руководствуясь вышеизложенным, я предприняла попытку реализовать на практике алгоритм автоматического распознавания слитной казахской речи с применением нейросети обратного распространения.

Моя нейронная сеть содержит 5 слоев: входной, выходной и 3 внутренних. Во входном слое 50 входов, в выходном — 20 нейронов, в 1-м, 2-м и 3-м внутренних слоях, соответственно, 40, 30 и 20 нейронов.

В качестве активационной используется функция $f(x) = \dfrac{1}{1+e^{-x}}$.

Скорость обучения (амплитуда коррекции весов на каждом шаге обучения) равна 0,1.

Момент (степень воздействия i-ой коррекции весов на $(i+1)$-ую) равен 0,9.

В качестве входных данных использовались первые 50 гармоник спектра фонем, нормированные в логарифмическом масштабе. В качестве выходных данных — 20-разрядные векторы, заполненные значениями «0» и «100».

Сеть была обучена распознавать 20 звуков казахского языка: «А», «Е», «О», «Ө», «Ы», «Б», «Х», «Г», «К», «С», «З», «Р», «Л», «Ш», «Й», «Ж», «Т», «Н», «ң», «М», — с настройкой на голос конкретного диктора. Обучающая выборка содержала 800 примеров, по 40 на каждый звук. Различия между твердыми и мягкими согласными не производилось. Так, например, нейросеть была обучена распознавать символ «Р» как в звуке «Р», так и в звуке «Рь».

При обучении 80% выборки использовалось как обучающее множество, 20% — как тестовое. Обучение заняло около десяти тысяч эпох и завершилось, когда более 95% тестового множества примеров было распознано верно.

Затем было проведено тестирование обученной нейросети на реальных данных: словах и предложениях звуковой базы.

Нейросеть получает на входе спектры звуковых фрагментов и рассчитывает набор выходных значений. Набору выходных значений сопоставляется некий символ. При сопоставлении используются следующие правила:

– если амплитуды всех гармоник < 0,1, то на выходе определяется символ «_», т.е. пауза;

– если на одном из выходов получено значение > 75, то на выходе определяется символ (буква казахского алфавита), назначенный именно этому выходу нейросети;

– если ни на одном из выходов не получено значение > 75, то на выходе определяется символ «?», т.е. неизвестный символ;

– если на двух и более выходах получено значение > 50, то программа сигнализирует о неоднозначном решении.

Согласный определяется при стабилизации формы сигнала в 3-х и более последовательных звуках, гласный — при стабилизации в 4-х и более последовательных звуках.

Приведенные результаты дают повод как для размышлений, так и для оптимизма. Система явно различает большинство звуков, хотя и бессистемно путает родственные звуки, с трудом отличает «А» от «Ы», а «О» – от «Ұ». Наибольшие сложности для распознавания представляют короткие по длительности согласные «Б», «Г», «К», «Т» и (как ни странно) согласный «Р», который система путает не столько с другими согласными, сколько с различными гласными. Звук «?» вообще ни разу не был правильно распознан. Шипящие же и шумовые согласные («Х», «Ж», «Ш», «З») система распознает очень уверенно; сонорные звуки («Л», «М») — с достаточной уверенностью. Особенно радуют пары «Бол – Бұл», «Қыр – Хыр», «Ойын – Ойн», «За – Зын», «Тұр – Тұрр», «Қамыр – Хымыр», «Ала – Ала», «Жұмыс – Жұмыз» и др.

ЛИТЕРАТУРА

1. *Москаленко А.* Использование нейросетей для анализа звуковой информации. Краснодар: Кубанский государственный университет, 2000.

2. *Комашинский В.И., Смирнов Д.А.* Нейронные сети и их применение в системах управления и связи. М.: Горячая линия – Телеком, 2002.

3. *Круглов В.В., Дли М.И.* Интеллектуальные информационные системы: компьютерная поддержка систем нечеткой логики и нечеткого вывода. М.: Издательство физико-математической литературы, 2002.

4. Информационный портал компании BaseGroup Labs http://www.basegroup.ru/

5. *Лукин А.* Введение в цифровую обработку сигналов (математические основы): Учебно-методическое пособие. М.: МГУ, 2002.

6. *Деньгуб В.М., Смирнов В.Г.* Единицы величин. М.: Издательство стандартов, 1990.