Метод оптимального нелинейного растяжения симметричных матриц в задачах распознавания

И. Л. Мазуренко, А. А. Петюшко

В данной работе рассматриваются матрицы самосравнения одномерного сигнала (в частности, речевого). Предлагается метод нелинейного растяжения этих симметричных матриц для нахождения оптимального расстояния между ними в смысле похожести сигналов.

Ключевые слова: одномерный сигнал, матрица самосравнения, нелинейное растяжение.

Введение

Практический интерес представляет задача автоматического распознавания речевых команд. Традиционный подход решения этой задачи заключается в сравнении речевого сигнала с эталонами команд, фонем и т.п. Такое сравнение делается с помощью метода динамического программирования (см. [1] и [2]). Распространёнными походами к реализации автоматического распознавания речи являются методы скрытых марковских моделей (см. [3] и [4]) и динамической деформации времени [5].
К недостаткам данных подходов относится их неустойчивая работа в условиях сильных, в том числе нестационарных и низкочастотных, шумов. В данной работе рассмотрен подход, позволяющий повысить надёжность традиционных алгоритмов распознавания.
Кратко этот метод был упомянут в статье [6], в данной работе приводится его полное описание.

Постановка задачи

К уже существующим методам распознавания предлагается добавить ещё один, вычисляющий некоторым образом функцию расстояния между звуковыми сигналами (командами), чтобы сократить перебор в словаре и повысить надёжность результатов распознавания в условиях нестационарного шума. Отметим, что поскольку предполагается использовать этот метод в связке с другими, то от него не требуется распознавания как такового, т. е. расстояние может быть маленьким не только для сигналов, соответствующих одной и той же команде, но и для разных команд.
Для увеличения надёжности распознавания предлагается применить метод динамического программирования для нахождения расстояния не между одномерными сигналами, а между матрицами их самосравнения.
Дадим конструктивное определение **матрицы самосравнения** S для одномерного сигнала $s \in \mathbb{R}^N$:

1) Сигнал разбивается на n окон одинаковой длины (возможно, частично перекрывающихся).
2) В каждом окне с номером $i = 1 \dots n$ вычисляется вектор признаков $v_i(s) \in \mathbb{R}^k$ (спектральных, кепстральных, коэффициентов линейного предсказания и т.п. — набор признаков выбирается в зависимости от конкретной реализации).
3) На основе некоторой меры близости $\rho_k(.,.)$ (например, евклидовой метрики в пространстве $\mathbb{R}^k$) вычисляется расстояние для каждой пары векторов из соответствующих окон.
4) На место (i, j) в матрице самосравнения заносится вычисленное в п. 3 расстояние между i-м и j-м вектором признаков: $S(i, j) = \rho_k(v_i(s), v_j(s))$.

Таким образом, для сигнала s, которому соответствуют n векторов признаков $v_i(s)$ для $i = 1 \dots n$, получим матрицу самосравнения S размера $n \times n$.
Для примера прилагаются изображения команды "один" (Рис. 1) и ее матрицы самосравнения (Рис. 2; на рисунке более тёмные области соответствуют маленькому значению расстояния $\rho_k(.,.)$, а более светлые — наоборот, большому $\rho_k(.,.)$).
На функцию расстояния между матрицами самосравнения сигнала налагаются следующие содержательные ограничения:

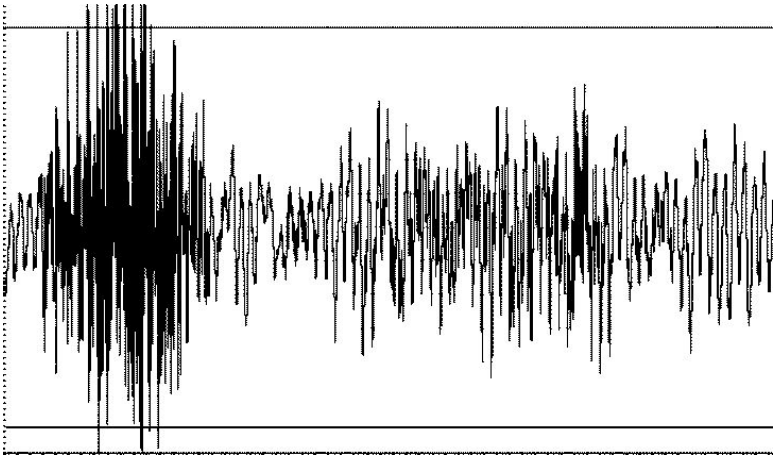

Рис. 1: Команда “один” в амплитудно-временной области

1) Расстояние инвариантно относительно изменения длины сигналов (и, как следствие, размеров квадратных матриц, которые мы сравниваем).

2) Полученная функция расстояния соответствует визуальной похожести матриц, определяемой человеком.

3) Расстояние между матрицами, являющимися самосравнениями произнесений одних и тех же команд, *в среднем не больше*, чем расстояние между матрицами из разных классов.

4) Описанное в п. B) условие “инвариантно” относительно сравниваемых пар классов. Это значит, что если у двух команд матрицы визуально разные, то и расстояния между этими матрицами должны быть в среднем большими, и наоборот: если у двух матриц из разных классов расстояние мало, то и матрицы самосравнения визуально похожи.

5) Желательно, чтобы вышеописанные требования выполнялись не только для матриц самосравнений сигналов с одним уровнем шума, но и для матриц, вычисленных по сигналам для разных уровней шума.

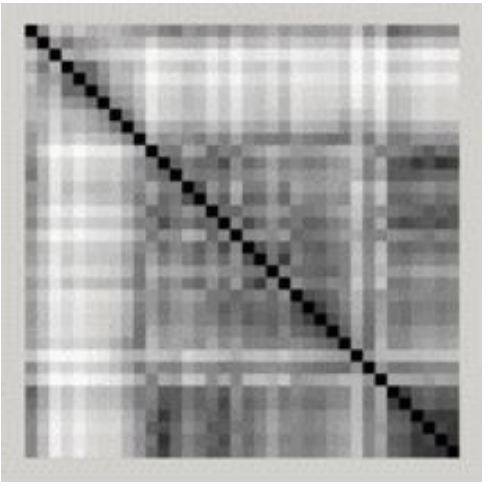

Рис. 2: Матрица самосравнения команды “один”

Описание метода

По аналогии с методом динамической деформации времени для одномерного сигнала, авторами статьи предлагается использовать метод нелинейного сжатия-растяжения матриц. Необходимость использования нелинейного преобразования состоит в том, что, как и в случае с одномерным сигналом, два произнесения последнего обычно находятся в нелинейном соответствии друг с другом (например, один звук растянут по времени в 1.5 раза, а другой, наоборот, сжат в 2 раза). Соответственно, и матрицы самосравнения одной команды также нелинейно зависят друг от друга.

Содержательно, необходимо таким образом увеличить две сравниваемые матрицы (будем именно *растягивать* матрицы) до одинакового размера, чтобы, с одной стороны, это растяжение минимизировало расстояние между этими матрицами, а с другой стороны, после растяжения можно было бы использовать некоторую функцию расстояния между матрицами одинаковой размерности.

Опишем процедуру для растяжения двух матриц самосравнения S_1, S_2 (размера $n_1 \times n_1$ и $n_2 \times n_2$ соответственно) для нахождения расстояния между ними более строго. Для этого строится квадратная таблица — **таблица растяжений** (или **таблица пути**) R размера $n_1 \times n_2$, элементом (i, j) которой является пара: 1) последовательность растяжений $Rp(i, j)$, минимизирующая расстояние

между квадратными подматрицами, левый верхний угол которых совпадает с верхним левым углом для исходных матриц S_1, S_2, размера $i \times i$ для S_1 и $j \times j$ для S_2 соответственно, и 2) само это расстояние $Rd(i, j)$ (см. Рис. 3).

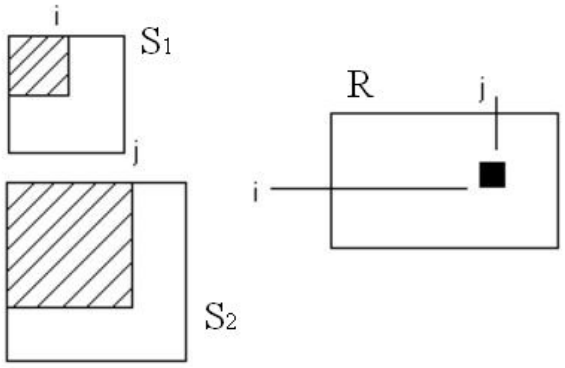

Рис. 3: Построение матрицы самосравнения

Отметим, что расстояние $Rd(i, j)$ между квадратными подматрицами размера $i \times i$ для S_1 и $j \times j$ для S_2 — это действительное число, т. е. $Rd(i, j) \in \mathbb{R}$. Последовательность же растяжений $Rp(i, j)$ — это упорядоченный набор пар $Rp(i, j) = ((a_1, b_1), \ldots, (a_m, b_m))$ таких, что $(a_1, b_1) = (1, 1)$, $(a_m, b_m) = (i, j)$, и $(a_k, b_k) \neq (a_{k+1}, b_{k+1})$, причём $0 \leq a_{k+1} - a_k \leq 1$ и $0 \leq b_{k+1} - b_k \leq 1$ для любого $k = 1, \ldots, m-1$. Предположим, что мы используем меру близости $f_k(.,.)$ для нахождения расстояния между двумя векторами размерности k (заметим, что в общем случае $f_k(.,.)$ и $\rho_k(.,.)$ — разные функции). Также будем использовать обозначения $e_1, e_2, \ldots$ для обозначения базисных векторов в многомерном пространстве. Тогда первая строка и первый столбец таблицы растяжений R формируются следующим

образом:

$$Rd(1,1) = f_1(S_1(1,1), S_2(1,1)),$$
$$Rp(1,1) = (1,1),$$
$$Rd(1,k) = f_k\left(S_1(1,1)\sum_{i=1}^{k} e_i,\ \sum_{i=1}^{k} S_2(1,i)e_i\right) + Rd(1,k-1),$$
$$k = 2,\dots,n_2, Rp(1,k) = ((1,1),\dots,(1,k)),\ k = 2,\dots,n_2$$
$$Rd(m,1) = f_m\left(\sum_{i=1}^{m} S_1(1,i)e_i,\ S_2(1,1)\sum_{i=1}^{m} e_i\right) + Rd(m-1,1),$$
$$m = 2,\dots,n_1, Rp(m,1) = ((1,1),\dots,(m,1)),\ m = 2,\dots,n_1.$$

Содержательно это значит, что левый верхний элемент матрицы S_1 копируется на все координаты вектора длины k, и на место $(1,k)$ таблицы R в качестве Rd заносится сумма расстояния между этим вектором и вектором такой же длины, являющемся началом k-ой строки второй матрицы S_2 длины k, и числа $Rd(1,k-1)$ из клетки слева (для первого столбца все аналогично с точностью до замены S_1 на S_2). Одновременно в каждую клетку R в качестве Rp заносится путь, по которому мы в неё пришли, т.е. последовательность таких пар чисел, в которых первое число отвечает за номер строки (или столбца, что то же самое, т.к. матрицы самосравнения симметричны) первой матрицы S_1, а второе — за номер строки (столбца) второй матрицы S_2.
Для остальных клеток таблицы растяжений используем следующую итеративную процедуру:

1) Пусть надо посчитать путь и расстояние для клетки с номером (i,j). Предположим, что для всех других клеток с индексами (i_1,j_1) таких, что $(i_1,j_1) \neq (i,j)$, $1 \le i_1 \le i, 1 \le j_1 \le j$, мы знаем $Rp(i_1,j_1), Rd(i_1,j_1)$ (как заполнить первую строку и первый столбец, было показано выше).

2) Рассмотрим три соседние уже заполненные клетки: с индексами $(i-1,j-1)$, $(i,j-1)$ и $(i-1,j)$. Рассчитаем три значения $d_{1,1}$, $d_{0,1}$ и $d_{1,0}$, которые будут соответствовать расстоянию между квадратными подматрицами размера $i \times i$ для S_1 и $j \times j$ для S_2 при разных последовательностях растяжений этих подматриц, а именно:

- $d_{1,1}$: одновременное растяжение на 1 клетку обеих подматриц S_1 и S_2,
- $d_{0,1}$: растяжение на 1 клетку только подматрицы S_2,
- $d_{1,0}$: растяжение на 1 клетку только подматрицы S_1.

3) Для примера рассчитаем значение $d_{1,1}$ ($d_{0,1}$ и $d_{1,0}$ рассчитываются аналогично). Пусть $Rp(i-1, j-1) = (a_1, b_1), \ldots, (a_m, b_m)$. Тогда, если обозначить через

$$v_1 = S_1(i,i)e_{m+1} + \sum_{k=1}^{m} S_1(i, a_k)e_k,$$
$$v_2 = S_2(j,j)e_{m+1} + \sum_{k=1}^{m} S_2(j, b_k)e_k,$$

то расстояние между квадратными подматрицами размера $i \times i$ для S_1 и $j \times j$ для S_2 при одновременном растяжении на 1 клетку обеих подматриц будет равно

$$d_{1,1} = Rd(i-1, j-1) + f_{m+1}(v_1, v_2).$$

Заметим, что с помощью такой процедуры расчёта значений расстояний между квадратными подматрицами разного размера мы учитываем все прошлые растяжения, которые были записаны в предыдущую клетку пути, и вариация происходит, по сути, только в выборе растяжения последней клетки (растягиваем обе подматрицы сразу на 1 клетку, или только одну подматрицу; предыдущая последовательность растяжений сохраняется).

4) Пусть $\min(d_{1,1}, d_{0,1}, d_{1,0}) = d_{a,b}$, где $0 \le a, b \le 1$.
Тогда

$$Rd(i,j) = d_{a,b},$$
$$Rp(i,j) = (Rp(i-a, j-b), (i,j)).$$

В итоге в последней (правой нижней) ячейке таблицы растяжений мы получим путь $Rp(n_1, n_2)$ — последовательность растяжений, которая минимизирует расстояние между матрицами самосравнения S_1, S_2.

Как следствие из вышеописанной процедуры нахождения оптимального растяжения симметричных матриц имеем следующую теорему:

Теорема 1. *Сложность алгоритма (в терминах количества операций взятия минимума и нахождения расстояния между векторами) нахождения оптимального растяжения двух симметричных матриц самосравнения S_1, S_2 размеров $n_1 \times n_1$ и $n_2 \times n_2$ соответственно равна $O(n_1 n_2)$.*

Опишем процедуру построения растянутых матриц T_1, T_2, которые получаются применением последовательности растяжений $Rp(n_1, n_2)$ к изначальным матрицам самосравнения S_1, S_2. Пусть $Rp(n_1, n_2)$ содержит m пар индексов: $Rp(n_1, n_2) = ((a_1, b_1), \ldots, (a_m, b_m))$. Тогда матрицы T_1, T_2 имеют одинаковый размер $m \times m$ и их элементы определяются следующим образом $(1 \leq i, j \leq m)$:

$$T_1(i, j) = S_1(a_i, a_j),$$
$$T_2(i, j) = S_2(b_i, b_j).$$

Напоследок заметим, что в качестве расстояния $r(S_1, S_2)$ между симметричными матрицами самосравнения S_1, S_2 предлагается брать не значение $Rd(n_1, n_2)$, а некоторую матричную норму $g(.)$ от разности растянутых матриц T_1, T_2 одного размера:

$$r(S_1, S_2) = g(T_1 - T_2).$$

Такой матричной нормой может служить, например, матричная норма Фробениуса.

Усовершенствование метода

Чтобы описанный выше метод удовлетворял поставленным нами содержательным ограничениям на функцию расстояния между матрицами $r(.,.)$, то можно, во-первых, в качестве расстояния между векторами $f_k(.,.)$ брать взвешенную евклидовую метрику, т. е.

$$f_k(v_1, v_2) = \sqrt{\frac{1}{k} \sum_{i=1}^{k} (v_1(i) - v_2(i))^2},$$

и, во-вторых, матричную норму делить на некоторый эквивалент длины оптимального пути — например, норму Фробениуса на площадь матрицы $T_1 - T_2$ размера $m \times m$:

$$r(S_1, S_2) = \frac{g(T_1 - T_2)}{m^2}.$$

Эти соображения позволяют достичь инвариантности от размера как при вычислении расстояния между векторами (более длинные вектора не будут давать бОльшее расстояние), так и при вычислении матричной нормы (большие матрицы не дадут бОльшее расстояние, чем маленькие, при прочих равных).
При сравнении растянутых матрицы поэлементно (норма Фробениуса) мы считаем равноправными все пары окон звука внутри сигналов. На практике ситуация выглядит сложнее. Например, если произнесённое слово достаточно длинное, и в нем есть два одинаковых звука в начале и конце слова, то начало и конец слова могут произноситься по-разному хотя бы потому, что шумовой фон за это время успел измениться, или голос человека изменил окраску и т.п. Соответственно, при вычислении матрицы самосравнения может получиться так, что соответствующие этим звукам элементы матрицы имеют существенно разные значения.
Таким образом, возможно, что два одинаковых звука в одном слове могут произноситься по-разному. При этом мы хотим, чтобы два разных звука в одном и том же слове не могли быть близкими в смысле используемой меры близости.
Чтобы учесть это обстоятельство, в определение матричной нормы нами были введены дополнительно весовые коэффициенты, которые тем меньше, чем дальше по времени отстоят сравниваемые звуки друг от друга. Пусть симметричная матрица T с нулями на диагонали имеет размер $n \times n$. Тогда в качестве функции $g(T)$ будем использовать:

$$g(T) = 2\sqrt{\sum_{j<i}^{n} \left(T(i,j) \cdot \frac{j}{i} \right)^2}.$$

Программная реализация метода

Описанный в работе метод был реализован в виде программной системы распознавания речевых сигналов в условиях шума. Реализованная система распознавания работает в реальном времени в операционной системе MS Windows ©.
Входные команды будут задаваться отдельными wav-файлами. Для вычисления параметров будем использовать: шаг анализа wav-файлов — 128 отсчётов; размер окна анализа — 512 (т. о., получаем

перекрытие в 3/4 окна); число коэффициентов линейного предсказания — 14; количество компонент спектра линейного предсказания — 256.

Пользователю предлагается выбрать как файл, содержащий первую команду, так и файл, содержащий вторую команду. Внешний вид программы при выборе файла команды изображён на Рис. 4.

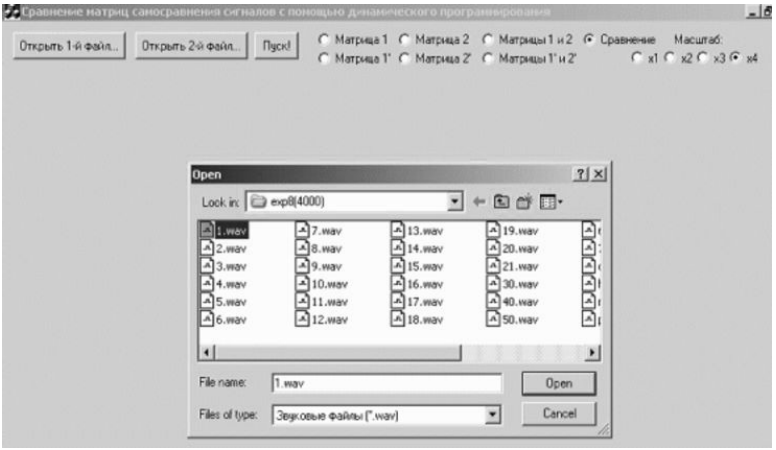

Рис. 4: Программный интерфейс

После загрузки файлов для обеих команд можно посмотреть и оценить близость двух матриц самосравнения (см. Рис. 5).

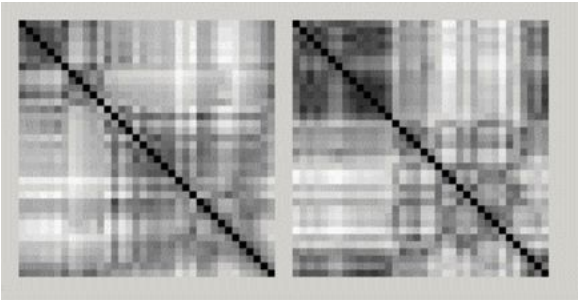

Рис. 5: Анализируемые матрицы самосравнения

После нажатия кнопки “Пуск!” программа вычисляет расстояние между матрицами (которое сразу выводится на экран — см. Рис. 6) и оптимальное растяжение матриц.

Рис. 6: Расстояние между матрицами самосравнения

Можно посмотреть на обе растянутые матрицы (см. Рис. 7 — как видно, поскольку изначально файлы были произнесением одной команды, то и растяжения практически нет), а также таблицу растяжений Rd с помеченным в ней минимизирующим путём (см. Рис. 8).

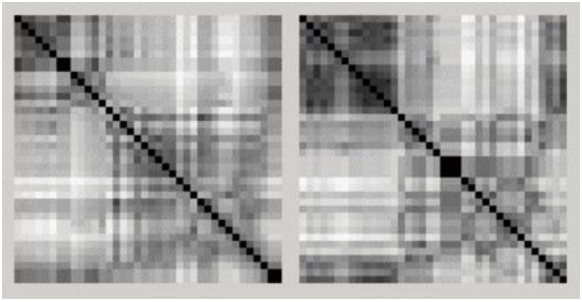

Рис. 7: Растянутые матрицы самосравнения

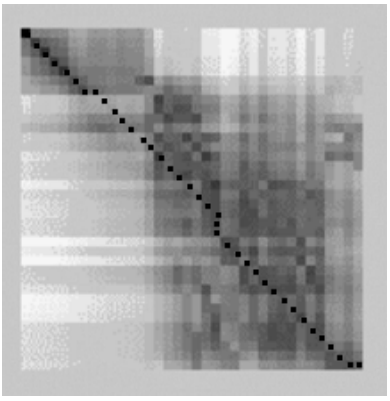

Рис. 8: Оптимальный путь в таблице растяжений

В данном случае использовались команды для слова “четыре”, записанные в автомобиле с 4000 и 800 оборотами двигателя в минуту соответственно. Для разных команд, например, “один” и “четыре”, получаются матрицы самосравнения, изображённые на Рис. 9, оп-

тимальным образом растянутые - на Рис. 10, расстояние между ними — 0.3128 (сравните с 0.2415).

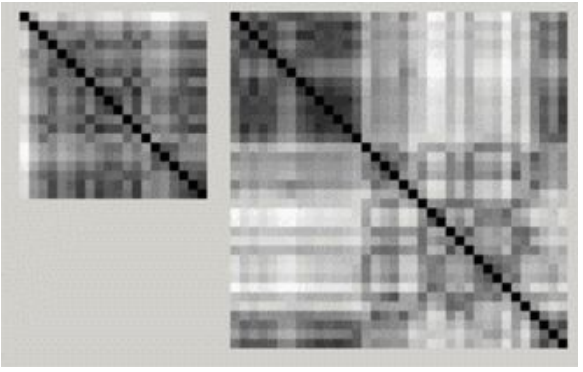

Рис. 9: Матрицы самосравнения для разных команд

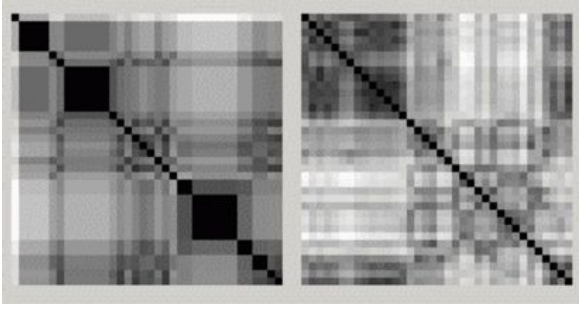

Рис. 10: Растянутые матрицы самосравнения для разных команд

Проведенные эксперименты.

Команды произносились в автомобиле с 4000 и 800 оборотами двигателя в минуту (именно для этих значений наблюдается самая большая разница в отношении сигнал/шум). Использовались следующие речевые команды (всего $L = 29$): “один”, “два”, “три”, “четыре”, “пять”, “шесть”, “семь”, “восемь”, “девять”, “десять”, “одиннадцать”, “двенадцать”, “тринадцать”, “четырнадцать”, “пятнадцать”, “шестнадцать”, “семнадцать”, “восемнадцать”, “девятнадцать”, “двадцать”, “тридцать”, “сорок”, “пятьдесят”, “шестьдесят”, “семьдесят”, “да”, “нет”, “хорошо”, “понял”.

В качестве эталона для каждой команды считалось произнесение слова при 800 оборотов двигателя в минуту (с минимальным уровнем шума). Затем этот эталон сравнивался описанным выше методом со всеми командами, записанными при 4000 оборотов двигателя в минуту (т. е. максимальное значение уровня шума). На Рис. 11 показана соответствующая матрица расстояний $D(i,j), 1 \leq i,j \leq L$.

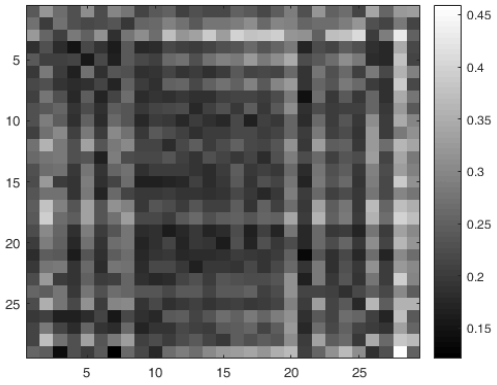

Рис. 11: Матрица попарных расстояний для 29 команд с разным уровнем шума

Поскольку цель - не создать новый метод распознавания, а дополнить уже существующие для повышения надёжности, то для нас важно, чтобы расстояние между матрицами, являющимися самосравнениями произнесений одних и тех же команд, было *в среднем не больше*, чем расстояние между матрицами из разных классов. Поэтому для каждой команды с номером $i, 1 \leq i \leq L$, будем оценивать следующую величину:

$$c(i) = \frac{\frac{1}{L}\sum_{j=1}^{L} D(i,j) - D(i,i)}{\frac{1}{L}\sum_{j=1}^{L} D(i,j)}.$$

Будем считать распознавание “очень надёжным” для команды с номером i, если значение $c(i) \geq 0.05$; “надёжным”, если $-0.05 \leq c(i) < 0.05$; и “ненадёжным”, если $c(i) \leq -0.05$. На Рис. 12 показан график оценки надёжности $c(i), 1 \leq i \leq L$.
Как видим, только для 3 команд (“восемь”, “сорок” и “хорошо”; отмечены кружками) предложенный метод даёт неудовлетворитель-

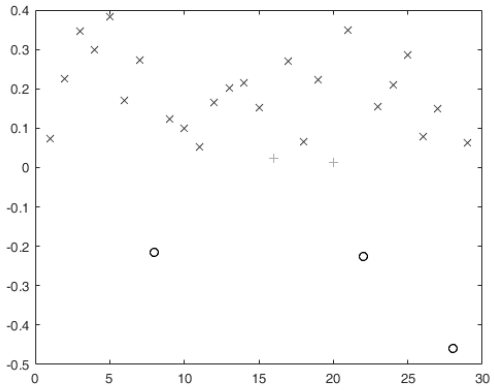

Рис. 12: Оценка надежности распознавания

ные результаты, и для 2 (“шестнадцать” и “двадцать”; отмечены плюсами) надёжность не слишком высокая. Тем не менее, 26 из 29 команд распознаны с удовлетворительной надёжностью, что в результате даёт около 90% правильных распознаваний и позволяет утверждать, что предложенный метод вполне годится для совместного применения с другими методами распознавания для повышения их надёжности.

Заключение

В качестве возможного будущего улучшения предложенного метода можно предложить следующие идеи. Во-первых, перед применением незашумлённого эталона его следует сначала как-то специально “зашумить”, используя текущую шумовую обстановку (либо, наоборот, удалить шумовые помехи из анализируемого сигнала — см. работы [7] и [8]). Также при создании оптимального растяжения возможно делать некий аналог сглаживания соседних строк, чтобы не копировать строки матриц без обработки.

Список литературы

[1] Р. Беллман. Динамическое программирование. – М.: Изд-во иностранной литературы, 1960.

[2] R. Bellman. On the theory of dynamic programming. // Proceedings of the National Academy of Sciences. – 1952. – Vol. 38. – № 8. – P. 716–719.

[3] J. K. Baker. The DRAGON system — An overview. // Acoustics, speech and signal processing, IEEE transactions on. – 1975. – Vol. 23. – № 1. – P. 24–29.

[4] F. Jelinek, L. R. Bahl and R. L. Mercer. Design of a linguistic statistical decoder for the recognition of continuous speech. // Information Theory, IEEE Transactions on. – 1975. – Vol. 21. – № 3. – P. 250–256.

[5] Т. К. Винцюк. Распознавание слов устной речи методами динамического программирования. // Кибернетика. – 1968. – № 1. – С. 81–88.

[6] Д. Н. Бабин, В. В. Дементиенко, И. Л. Мазуренко, В. И. Миргородский, Д. С. Михайлов, А. Б. Холоденко и А. В. Уранцев. Система речевого контроля состояния машиниста. // Интеллектуальные системы. Теория и приложения. – 2014. – Т. 18. – № 4. – С. 69–75.

[7] И. Л. Мазуренко. Об одном подходе к линейной адаптивной цифровой обработке сигналов. // Интеллектуальные системы. – 2013. – Т. 17. – № 1–4. – С. 245–248.

[8] А. А. Петюшко. Анализ распознавания одномерных сигналов в шуме. // Молодежь в науке: Сборник докладов Четвертой научно-технической конференции, Саров, 1-3 ноября 2005 г. – ФГУП “РФЯЦ-ВНИИЭФ” Саров. – 2006. – С. 131–137.