Санкт-Петербургский государственный университет
Прикладная математика и информатика
Вычислительная стохастика и статистические модели

Бакшинская Екатерина Олеговна

Применение методов Монте–Карло для решения многокритериальных задач

Выпускная квалификационная работа

Научный руководитель:
к.ф.-м.н., доцент П. В. Шпилев

Рецензент:
к. ф.-м. н., доцент Н. П. Алексеева

Санкт-Петербург
2018

Saint Petersburg State University
Applied Mathematics and Computer Science
Computational Stochastics and Statistical Models

Bakshinskaia Ekaterina

Applying Monte-Carlo methods to a solution of multi-criteria problems

Graduation Project

Scientific Supervisor:
Associate professor P. V. Shpilev

Reviewer:
Associate professor N. P. Alekseeva

Saint Petersburg
2018

Оглавление

Введение

Многокритериальная задача — это математическая модель принятия оптимального решения одновременно по нескольким критериям. Эти критерии могут отражать оценки различных качеств объекта (или процесса), по поводу которых принимается решение, или оценки одной и той же его характеристики, но с различных точек зрения. В данной работе под многокритериальными задачами мы будем понимать задачи теории игр.

Математическая теория игр — дисциплина, возникшая на стыке 2 наук: математики и экономики. В настоящее время область ее практического применения достаточно широка: политические и дипломатические отношения, биология, информатика и другие. Предметом изучения данной дисциплины являются игры и связанные с ними стратегии (т.е. множества доступных игрокам действий), а целью — выработка методов нахождения оптимальных стратегий. Под игрой, как правило, понимается некоторая конфликтная ситуация, т.е. процесс, в котором принимают участие от двух и более сторон преследующих собственные цели (см., например, [1]).

Игры принято различать по типу. Например, кооперативные и некооперативные игры, с нулевой суммой и ненулевой суммой, параллельные и последовательные, метаигры, игры с полной или неполной информацией [2], [10]. Последние, с одной стороны, наиболее сложны для построения моделей прогнозирования, с другой - представляют значительный практический интерес, т.к., достаточно часто в реальных конфликтных ситуациях, участники не обладают полной информацией о действиях оппонентов [7], [9], [5].

Для дальнейших рассуждений нам понадобится следующее определение: *ход игрока* — это выбор действия из некоторого множества возможных действий игрока, определяемых правилами игры.

В нашей работе мы будем рассматривать игры, включающие в себя следующие элементы:

1. Чередование ходов, которые могут быть как личными, так и случайными;
2. Недостаточность информации;
3. Функция выигрыша [4].

Как уже было отмечено ранее, во многих реальных конфликтных ситуациях участники не обладают всей информацией об истинных состояниях процесса и могут оценивать результаты действий соперников только по некоторым ограниченным наборам данных. В этих условиях принятие решений может зависеть от нескольких факторов, в том числе — случайных. Один из возможных подходов к нахождению оптимальной стратегии в таких задачах это использование для описания исследуемой ситуации вероятностных моделей.

В данной статье мы будем использовать *Скрытые Марковские Модели* (СММ) для описания состояний игры в различные моменты времени. СММ зарекомендовали себя как мощное

средство обработки данных в самых различных областях, с примерами применения можно ознакомиться в [8]. Преимущества СММ перед другими моделями следующие:

1. СММ обладают простой математический структурой;
2. Структура СММ позволяет моделировать сложную цепочку наблюдений;
3. Параметры модели могут быть автоматически выбраны таким образом, чтобы описать имеющийся набор данных для обучения (под обучением СММ понимается определение оценок параметров модели).

В недавно опубликованной работе [17] был предложен метод нахождения оптимальной стратегии для игры с двумя участниками, основанный на применении СММ. В статье был описан подход к решению стратегических игр, в которых игроки могут менять стратегии в течении игры. Целью было увеличить шансы игрока, т.е. узнать какую стратегию принимает соперник в каждый момент времени. Предложенный подход основан на использовании алгоритма Баума-Велша, решения марковской игры поиска смешанного равновесия Нэша.

В нашей работе мы хотим развить данный подход, предложив альтернативный метод, основанный на применении алгоритма Витерби для получения цепочки скрытых состояний и выбора наиболее вероятного хода вместо решения марковской игры.

Работу нашего алгоритма мы проиллюстрируем на двух демонстрационных примерах: игры в теннис и игры в “Камень, ножницы, бумага”.

Данная работа организована следующим образом: в Главе 1 дано теоретическое описание Скрытых Марковских Моделей, а так же предложена модель решения игры двух лиц с неполной информацией; в Главе 2 тестируется предложенная модель решения по игре, описанной в [17], также алгоритм тестируется по другой игре со схожими характеристиками.

Связанные работы

Основополагающей работой в теории игр является книга Джона фон Неймана и Оскара Моргенштерна “Теория игр и экономическое поведение” в 1944 г.[2] Большая часть данной книги посвящена играм с нулевой суммой. Игры с нулевой суммой — игры, в которых выигрыш одной стороны равен проигрышу другой. Игры с нулевой суммой с участием 2–х игроков называются *антагонистическими*. К таким играм также можно отнести игры с постоянной суммой, при которых сумма общего выигрыша всех игроков фиксирована, и поэтому увеличение выигрыша одного из них возможно только за счет уменьшения выигрышей других игроков.

В конце 1960–х годов Джон Харшаньи ввел понятие игр с неполной информацией и разработал концепцию байесовских равновесий. Д. Харшаньи рассматривал ситуации, когда у одного игрока нет информации о возможных выигрышах другого игрока, и выигрыши приходится оценивать вероятностно.

На практике часто встречаются ситуации, которые в теории игр принято называть *повторной игрой* (repeated game) т.е. игрой, которая состоит в некотором числе повторений некоторых игровых стадий (stage game). Впервые повторные игры с неполной информацией были введены Ауманом и Машлером в 1960г.[7] Рассматривались игры, в которых начальное состояние выбиралось с вероятностью p, и только игрок 1 знает это состояние, в то время как игрок 2 знает все возможные состояния, но не знает фактического состояния. После каждого хода, оба игрока знают исход предыдущего хода и продолжают играть.

В [16] рассматриваются игры с двумя игроками с нулевой суммой, заданные цепью Маркова над конечным множеством состояний и семейством матричных игр. Последовательность состояний подчиняется цепи Маркова. В начале каждого этапа только игроку 1 сообщается о текущем состоянии, затем воспроизводится соответствующая матричная игра, после чего выбранные действия наблюдаются обоими игроками. Такие игры можно охарактеризовать играми с отсутствием информации с одной стороны.

В [17] описана модель скрытой марковской игры (СМИ), при которой второй игрок может наблюдать исходы того или иного хода противника, знает множество возможных скрытых состояний соперника, но не знает скрытых состояний в определенные моменты времени и матрицу переходов из состояния в состояние.

В нашей работе мы рассмотрим альтернативный метод нахождения оптимальной стратегии, разработанный на основе предложенного в статье [17].

Глава 1

Теоретическое описание

1.1. Описание модели СММ

Скрытая марковская модель (СММ) с параметрами $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$ — это модель марковского процесса, в котором неизвестно, в каком состоянии находится система (состояния скрыты), в то же время каждое состояние может с некоторой вероятностью произвести некоторое событие, которое можно наблюдать (наблюдение).

В нашей работе мы рассматриваем однородные цепи маркова, то есть такие, матрица переходных вероятностей которых не зависит от номера шага: $\mathbf{A}_{ij}(t) = \mathbf{A}_{ij}$.

Формальное определение СММ следующее:

Определение 1. *Скрытая марковская модель* с параметрами $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$ описывается следующим образом:

1. конечное множество состояний $\mathbf{S} = \{s_1, \ldots, s_n\}$;

2. состояние вероятностей переходов, стохастическая матрица $\mathbf{A}$:

$$\mathbf{A} = \{a_{ij} | a_{ij} = P(\mathbf{S}_T = s_j | \mathbf{S}_{T-1} = s_i)\},\ 1 \leq i, j \leq n,$$

   где $\mathbf{S}_T$ и $\mathbf{S}_{T-1}$ состония в моменты времени T и $(T-1)$ соответственно.

   Вероятностные характеристики марковского процесса в следующий момент времени зависят только от вероятностных характеристик в текущий момент времени;

3. конечное множество наблюдений $\mathbf{O} = \{o_1, \ldots, o_k\}$;

4. состояние конкретного распределения вероятностей, матрица $\mathbf{B}$:

$$\mathbf{B} = \{b_{il} | b_{il} = P(\mathbf{O}_T = o_l | \mathbf{S}_T = s_i)\},\ 1 \leq i \leq n,\ 1 \leq l \leq k,$$

   где $\mathbf{O}_T$ — наблюдение в момент времени T, $\mathbf{S}_T$ — состояние в момент времени T.

   Значение наблюдаемого вектора $\mathbf{O}_T$, взятого в момент времени T, зависит только от скрытого состояния в момент времени T;

5. вектор начала состояний π

$$\pi = \{\pi_i | \pi_i = P(\mathbf{S}_1 = s_i)\}\ 1 \leq i \leq n,$$

   где $\mathbf{S}_1$ — состояние в нальный момент времени.

Популярным примером, иллюстрирующим использование СММ, является “Задача о нечестном казино”:

Имеется 2 кубика: один честный с вероятностью выпадения каждой грани $= \frac{1}{6}$ и нечестный с вероятностью выпадения 6–ки $= \frac{1}{2}$, а остальных граней $= \frac{1}{10}$. Крупье подменяет кубики с определенной вероятностью в зависимости от того, какой кубик был брошен в данный момент. Мы можем наблюдать, что выпадает на кубиках, но мы не знаем, какой из кубиков был использован. Таким образом, требуется определить последовательность скрытых состояний.

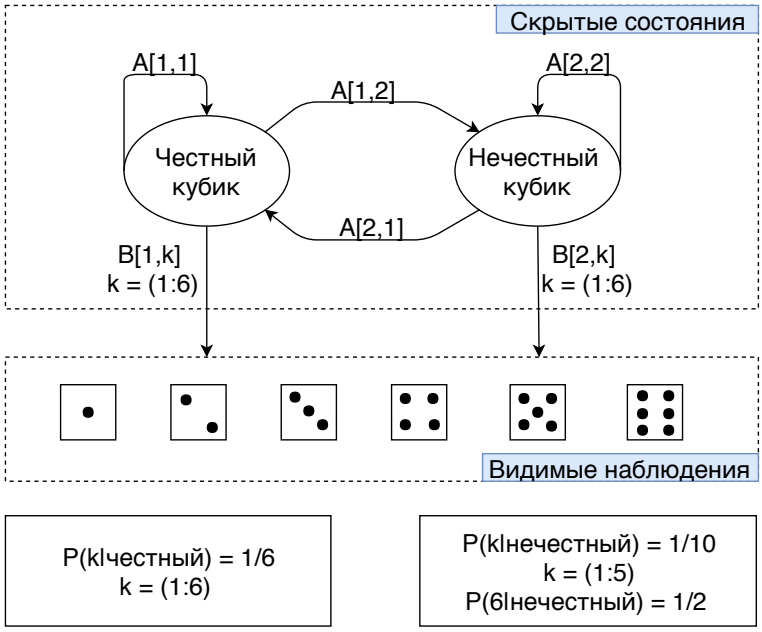

Рис. 1.1. Иллюстрация модели для задачи о нечестном казино.

1.2. Постановка задачи и пошаговый алгоритм решения

В рассматриваемой нами статье ([17]) ставилась задача: для игрока i найти вероятность того, что $\mathbf{O}_{t_{l+1}} = \mathbf{O}_j$ при $1 \leq j \leq k$, используя последовательность наблюдений $\mathbf{O}_{t_0}, \ldots, \mathbf{O}_{t_l}$. В то же время, в рамках рассматриваемой задачи, нам неизвестны ни последовательности состояний, ни элементы матриц $\mathbf{A}$ и $\mathbf{B}$.

Авторы рассматриваемой статьи на каждом шаге игры выводили оценки параметров модели СММ с помощью алгоритма Баума-Велша, и затем решали марковскую игру.

В нашей работе мы предлагаем вместо решения марковской игры использовать алгоритм Витерби для нахождения состояния другого игрока в текущий момент времени, после чего делать выбор наиболее вероятного хода.

Таким образом, в данной работе предлагается использовать следующий пошаговый алгоритм для решения поставленной задачи:

Входные данные:

- 2 игрока;
- множество доступных игрокам действий (наблюдений): $\mathbf{O} = \{o_1, \ldots, o_k\}$;

- множество состояний 1-го игрока: $\mathbf{S} = \{s_1, \ldots, s_n\}$;
- матрицы выплат для каждого из состояний $\{U_1, \ldots, U_k\}$ размерами $k \times k$.

Замечание: поведение 1-го игрока описывается с помощью СММ с параметрами $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$, где $\mathbf{A}$ — матрица вероятностей переходов из состояния i $(i = 1 : n)$ в состояние j $(j = 1 : n)$ размера $n \times n$, $\mathbf{B}$ — матрица вероятностей переходов из состояния i $(i = 1 : n)$ в наблюдение l $(l = 1 : k)$ размера $n \times k$, π — вектор вероятностей появления каждого из состояний в начальный момент времени длины n. Эти параметры являются скрытыми для 2-го игрока. В данной работе они являются входными данными, т.к. по ним моделируем действия 1-го игрока для оценки работы алгоритма.

Результат: на каждом шаге игры алгоритм выдает наиболее вероятную последовательность скрытых состояний 1-го игрока со всеми параметрами его СММ. По итогу считается количество побед 2-го игрока по матрицам выплат (пример можно будет увидеть в 2.1.2).

1. вычисляем элементы матрицы $\mathbf{B}$ размера $n \times k$, находя смешанные стратегии для первого игрока по матрицам выплат (описание метода нахождения смешанных стратегий в 2.1.1);
2. первые 3 хода игры второй игрок выбирает случайным образом;
3. формируется последовательность наблюдений (ходов первого игрока): $\tilde{\mathbf{O}} = \{\mathbf{O}_1, \mathbf{O}_2, \mathbf{O}_3\}$, где $\mathbf{O}_1$, $\mathbf{O}_2$ и $\mathbf{O}_3$ — какие-либо наблюдения из множества наблюдений $\mathbf{O}$, в моменты времени 1, 2 и 3 соответственно;
4. используем алгоритм Баума–Велша для вывода параметров СММ первого игрока:

   Входные данные:

   - последовательность наблюдений $\tilde{\mathbf{O}}$;
   - СММ с параметрами $\lambda = (\mathbf{A}^p, \mathbf{B}, \pi^p)$, где матрица $\mathbf{A}^p$ и вектор π^p произвольные.

   Результат: СММ с параметрами $\lambda^* = \arg\max_\lambda P(\tilde{\mathbf{O}}|\lambda)$ (подробное описание алгоритма и используемый пакет языка R в 1.3.1);
5. находим скрытые состояния 1-го игрока, используя алгоритм Витерби:

   Входные данные:

   - последовательность наблюдений $\tilde{\mathbf{O}}$;
   - СММ с параметрами λ^*.

   Результат: последовательность скрытых состояний противника: $\tilde{\mathbf{S}} = \{\mathbf{S}_1, \ldots, \mathbf{S}_t\}$, где $\mathbf{S}_1, \ldots, \mathbf{S}_t$ — какие-либо состояния из множества состояний $\mathbf{S}$, в моменты времени $1, \ldots, t$ соответственно (подробное описание алгоритма и используемый пакет языка R в 1.3.2);

6. предполагая определенным текущее состояние противника, выбираем для 2-го игрока вариант хода, при котором с наибольшей вероятностью максимизируется его выигрыш: выбираем наиболее вероятное состояние в следующий момент времени находим по матрице $\mathbf{B}$ наиболее вероятное наблюдение из этого состояния 1-го игрока. Делаем ход, который по правилам игры является выигрышным;

7. каждый игрок делает свой ход. Сравниваем наблюдения и вычисляем результат хода по имеющимся матрица выплат;

8. добавляем результат последнего хода 1-го игрока к цепочке наблюдений $\tilde{\mathbf{O}}$;

9. повторяем шаги (4)–(8) после каждого хода до окончания игры;

10. вычисляем результат игры по матрицам выплат.

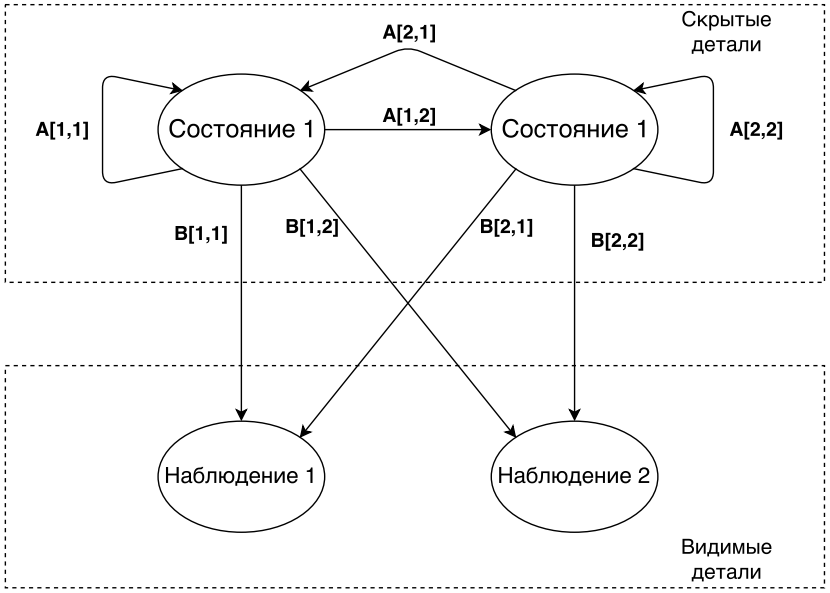

Рис. 1.2. Пример модели с двумя состояниями и двумя наблюдениями

1.3. Описание алгоритмов и инструментарий

Все описанные выше методы реализованы в языке программирования R. Выбор данного языка обусловлен наличием большого количества пакетов с открытым кодом, позволяющих решать задачи описанные скрытыми марковскими моделями. Для реализации предложенного нами алгоритма был использован пакет “HMM” [12] (код пакета “HMM”можно посмотреть в [11]), а конкретно функции из него:

initHMM — инициализация СММ

simHMM — генерация цепочки наблюдений и состояний по заданной СММ

baumWelch — алгоритм Баума-Велша

viterbi — алгоритм Витерби

Рассмотрим подробнее алгоритмы Баума-Велша и Витерби, использующиеся в данной работе:

1.3.1. Алгоритм Баума-Велша

Входными данными алгоритма являются: последовательность наблюдений $\tilde{\mathbf{O}}$ длины t и СММ с параметрами $\lambda = (\mathbf{A}^p, \mathbf{B}, \pi^p)$, где матрица $\mathbf{A}^p$ и вектор π^p произвольные, а матрица $\mathbf{B}$ в нашей задаче известна заранее (мы ее вычисляем, находя смешанные стратегии). В результате получим СММ с параметрами $\lambda^* = \arg\max_\lambda P(\tilde{\mathbf{O}}|\lambda)$, т.е. алгоритм пытается настроить модель на данную последовательность наблюдений $\tilde{\mathbf{O}}$.

Сам алгоритм состоит из следующих этапов:

1. Алгоритм прямого-обратного хода: используется для вычисления вероятности $P(\tilde{\mathbf{O}}|\lambda)$.

   Прямая процедура: вычисляем следующую вероятность:

   $$\alpha_r(i) = P(\mathbf{O}_1, \ldots, \mathbf{O}_r, \mathbf{S}_r = s_i|\lambda),\ 1 \le i \le n, \tag{1.1}$$

   это вероятность получения последовательности наблюдений $\tilde{\mathbf{O}}_r$, т.е. начальной части нашей, подаваемой на вход последовательности $\tilde{\mathbf{O}}$ (длина которой t), где $1 < r < t$, при условии, что в момент времени r мы будем находиться в состоянии s_i.

   Для этого:

   Инициализация:

   $$\alpha_1(i) = \pi_i b_{i,1},\ 1 \le i \le n. \tag{1.2}$$

   Рекурсия:

   $$\alpha_{r+1}(i) = P(\mathbf{O}_1, \ldots, \mathbf{O}_{r+1}, \mathbf{S}_{r+1} = s_i|\lambda) = \sum_{j=1}^{n} P(\mathbf{O}_1, \ldots, \mathbf{O}_r, \mathbf{O}_{r+1}, \mathbf{S}_r = s_j, \mathbf{S}_{r+1} = s_i|\lambda) = {}^{1}$$
   $$= \sum_{j=1}^{n} P(\mathbf{O}_1, \ldots, \mathbf{O}_r, \mathbf{S}_r = s_j, \mathbf{S}_{r+1} = s_i|\lambda) P(\mathbf{O}_{r+1}|\mathbf{S}_{r+1} = s_i) = {}^{2}$$
   $$= b_{i,r+1} \sum_{j=1}^{n} P(\mathbf{O}_1, \ldots, \mathbf{O}_r, \mathbf{S}_r = s_j|\lambda) P(\mathbf{S}_{r+1} = s_i|\mathbf{S}_r = s_j) = {}^{3} b_{i,r+1} \sum_{j=1}^{n} \alpha_r(j) a_{j,i},$$
   $$1 \le i, j \le n,\ 1 \le r \le t-1.$$

   Теперь, используя рекурсию, сможем найти $\alpha_t(i)$.

[1] Равенство следует из определения СММ

[2] $P(\mathbf{O}_{r+1}|\mathbf{S}_{r+1} = s_i) = b_{i,r+1}$

[3] $P(\mathbf{S}_{r+1} = s_i|\mathbf{S}_r = s_j) = a_{j,i}$

Таким образом, получаем: $P(\tilde{\mathbf{O}}|\lambda) = \sum_{i=1}^{n} \alpha_t(i)$.

Обратная процедура: вычисляем следующую вероятность:

$$\beta_r(i) = P(\mathbf{O}_{r+1}, \ldots, \mathbf{O}_t | \mathbf{S}_r = s_i, \lambda),\ 1 \le i \le n, \tag{1.3}$$

это вероятность получения последовательности наблюдений $\tilde{\mathbf{O}}_r$, которая в данном случае уже является "концом" нашей последовательности $\tilde{\mathbf{O}}$ (размера t), подаваемой на вход, при условии, что мы начинаем в момент времени $r+1$ из исходного состояния s_i и заканчиваем в момент времени t.

Аналогично с прямой процедурой:

Инициализация:

$$\beta_t(i) = 1,\ 1 \le i \le n. \tag{1.4}$$

Рекурсия:

$$\beta_r(i) = P(\mathbf{O}_{r+1}, \ldots, \mathbf{O}_t | \mathbf{S}_r = s_i, \lambda) = {}^{4} \sum_{j=1}^{n} P(\mathbf{O}_{r+1}, \ldots, \mathbf{O}_t, \mathbf{S}_{r+1} = s_j | \mathbf{S}_r = s_i, \lambda) = {}^{5}$$

$$= \sum_{j=1}^{n} P(\mathbf{O}_{r+2}, \ldots, \mathbf{O}_t | \mathbf{S}_r = s_i, \lambda) P(\mathbf{O}_{r+1} | \mathbf{S}_{r+1} = s_j) P(\mathbf{S}_{r+1} = s_j | \mathbf{S}_r = s_i) = {}^{6}$$

$$= \sum_{j=1}^{n} \beta_{r+1}(j) b_{i,r+1} a_{i,j},\ 1 \le i, j \le n,\ 1 \le r \le t-1.$$

Используя рекурсию, мы можем вычислить: $\beta_1(i)$.

Из 1.2 и 1.3 можем получить: $P(\tilde{\mathbf{O}}|\lambda) = \sum_{i=1}^{n} \alpha_1(i)\beta_1(i) = \sum_{i=1}^{n} \pi_i b_{i,1} \beta_1(i)$.

В то же время, из 1.1 и 1.3 следует: $P(\tilde{\mathbf{O}}, \mathbf{S}_r = s_i | \lambda) = \alpha_r(i)\beta_r(i),\ 1 \le i \le n,\ 1 \le r \le t$. Из полученной вероятности, мы теперь можем выразить искомую вероятность $P(\tilde{\mathbf{O}}|\lambda)$ следующим образом:

$$P(\tilde{\mathbf{O}}|\lambda) = \sum_{i=1}^{n} P(\tilde{\mathbf{O}}, \mathbf{S}_r = s_i | \lambda) = \sum_{i=1}^{n} \alpha_r(i)\beta_r(i),\ r \in [1, \ldots, t]. \tag{1.5}$$

2. Вычисление вспомогательных переменных:

   Через переменную ξ_r обозначим вероятность того, что модель CMM, при нашей последовательности наблюдений $\tilde{\mathbf{O}}$, в моменты времени r и $r+1$ будет находиться в состояниях s_i и s_j соответственно.

   Используя 1.1 и 1.3 можем вычислить данную вероятность следующим образом:

[4] Равенство следует из определения CMM

[5] Равенство следует из определения CMM

[6] $P(\mathbf{O}_{r+1}|\mathbf{S}_{r+1} = s_j) = b_{i,r+1},\ P(\mathbf{S}_{r+1} = s_j | \mathbf{S}_r = s_i) = a_{i,j}$

$$\xi_r(i,j) = P(\mathbf{S}_r = s_i, \mathbf{S}_{r+1} = s_j|\tilde{\mathbf{O}}, \lambda) = \frac{P(\mathbf{S}_r = s_i, \mathbf{S}_{r+1} = s_j, \tilde{\mathbf{O}}|\lambda)}{P(\tilde{\mathbf{O}}|\lambda)} = \\ = \frac{\alpha_r(i)a_{i,j}\beta_{r+1}(j)b_{j,r+1}}{\sum_{i_s=1}^{n}\sum_{j_s=1}^{n}\alpha_t(i_s)a_{i_s,j_s}\beta_{r+1}(j_s)b_{j_s,r+1}},\ 1 \le r \le t-1. \quad (1.6)$$

Через переменную γ_r обозначим вероятность того, что модель СММ, при нашей последовательности наблюдений $\tilde{\mathbf{O}}$, в момент времени r будет находиться в состоянии s_i.

Для получения γ используем формулы 1.1, 1.3 и 1.5:

$$\gamma_r(i) = P(\mathbf{S}_r = s_i|\tilde{\mathbf{O}}, \lambda) = \frac{P(\tilde{\mathbf{O}}, \mathbf{S}_r = s_i|\lambda)}{P(\tilde{\mathbf{O}}|\lambda)} = \frac{\alpha_r(i)\beta_r(i)}{\sum_{i_s=1}^{n}\alpha_r(i_s)\beta_r(i_s)},\ 1 \le r \le t. \quad (1.7)$$

Заметим, что перемнные $\xi_r(i,j)$ и $\gamma_r(i)$ связаны следующим образом:

$$\gamma_r(i) = \sum_{j=1}^{n}\xi_r(i,j),\ 1 \le i \le n,\ 1 \le r \le t-1. \quad (1.8)$$

3. Перевычисление параметров модели λ.

   На данном этапе происходит перевычисление параметров модели с целью увеличения $P(\tilde{\mathbf{O}}|\lambda)$. Новые параметры определяются следующим образом:

$$\pi_i^* = \gamma_1(i),\ 1 \le i \le n, \quad (1.9)$$

$$a_{ij}^* = \frac{\sum_{r=1}^{t-1}\xi_r(i,j)}{\sum_{r=1}^{t-1}\gamma_r(i)},\ 1 \le i \le n,\ 1 \le j \le n. \quad (1.10)$$

   Напоминаем, что в нашей постановке задачи нам не требуется перевычисление элементов матрицы $\mathbf{B}$, так как она нам дана точная. Поэтому в алгоритме Баума-Велша этот шаг опускаем.

4. Проверка сходимости.

   После перевычисления параметров, мы получили следующую СММ: $\lambda^* = (\mathbf{A}^*, \mathbf{B}, \pi^*)$. Для нее вычисляем $P(\tilde{\mathbf{O}}|\lambda^*)$ по одной из формул, представленных ранее на этапе 1.

   Замечание 1. Алгоритм Баума-Велша обеспечивает неубывание этой вероятности.

   На этом шаге проверяем выполняется ли следующее условие:

$$P(\tilde{\mathbf{O}}|\lambda^*) - P(\tilde{\mathbf{O}}|\lambda) < \varepsilon, \quad (1.11)$$

где ε — некоторая заданная величина.

В случае отрицательного исхода, переобозначаем $\lambda = \lambda^*$, и итерации продолжаются (повторяются все этапы).

Если неравенство выполнено, алгоритм заканчивает работу и мы получаем на выход λ^*.

Описание алгоритма Баума-Велша можно также посмотреть в [13], [15].

Реализация алгоритма Баума-Велша, осуществленная в пакете "HMM" не предусматривает оценку вектора π. В связи с этим, код алгоритма Баума-Велша был доработан с учетом специфики нашей задачи. А именно, была добавлена оценка вектора π и убрана оценка матрицы $\mathbf{B}$ в целях уменьшения количества свободных переменных.

1.3.2. Алгоритм Витерби

Позволяет построить наиболее вероятную последовательность предыдущих состояний скрытой марковской модели на основе последовательности наблюдений.

Входные данные: последовательность наблюдений $\tilde{\mathbf{O}}$ и СММ с параметрами $\lambda^* = (\mathbf{A}^*, \mathbf{B}, \pi^*)$, полученными в алгоритме Баума-Велша. Результатом работы алгоритма будет последовательность скрытых состояний $\tilde{\mathbf{S}} = \{\mathbf{S}_1, \ldots, \mathbf{S}_t\}$.

Алгоритм состоит из следующих этапов:

1. Вычисление максимальной вероятности того, что при последовательности наблюдений $\tilde{\mathbf{O}}$, СММ в момент времени r будет находиться в состоянии s_i.

   Обозначим эту вероятность следующим образом:

$$\delta_r(i) = \max_{\mathbf{S}_1,\ldots,\mathbf{S}_{r-1}} P(\mathbf{S}_1, \ldots, \mathbf{S}_{r-1}, \mathbf{S}_r = s_i, \mathbf{O}_1, \ldots, \mathbf{O}_{r-1}|\lambda). \tag{1.12}$$

   Инициализация:

$$\delta_1(j) = \pi_j b_{j,1},\ 1 \le j \le n. \tag{1.13}$$

   Рекурсия:

$$\delta_{r+1}(j) = b_{j,r+1} \max_{1\le i\le n}(\delta_r(i)a_{i,j}),\ 1 \le i, j \le n,\ 1 \le r \le t-1. \tag{1.14}$$

   Но в данном случае следует запоминать еще и состояния $\psi_r(j)$ (s_j в момент времени r). Это те состояния, при которых $\delta_r(j)$ достигает максимума:

$$\psi_{r+1}(j) = \arg\max_{1\le i\le n}(\delta_r(i)a_{i,j}),\ 1 \le j \le n,\ 1 \le r \le t-1. \tag{1.15}$$

   Используя данную рекурсию сможем вычислить $\delta_t(i)$, $1 \le i \le n$.

2. На следующем этапе следует вычислять наиболее вероятное конечное состояние системы $\mathbf{S}_r$ после шага $t-1$:

$$\mathbf{S}_t = \arg\max_{1\le i\le n}(\delta_t(i)). \tag{1.16}$$

В этой формуле $\max_{1\le i\le n}(\delta_t(i))$ — это наибольшая из максимальных вероятностей нахождение системы в момент времени t в состоянии s_i.

3. На последнем этапе вычисляется наиболее вероятная последовательность состояний системы $\tilde{\mathbf{S}}$.

Вычисляется она обратным проходом по массиву состояний ψ_r (при которых вероятности $\delta_r(i)$ были максимальны, начиная с наиболее вероятного конечного состояния $\mathbf{S}_t$).

$$\mathbf{S}_r = \psi_{r+1}(\mathbf{S}_{r+1}),\ 1 \le r \le t-1. \tag{1.17}$$

Описание алгоритма Витерби можно также посмотреть в [13], [15].

Глава 2

Тестирование

2.1. Тестирование на примере игры “теннис”

Протестируем наш алгоритм на примере игры описанной в [17], сравним с результатами, полученными авторами статьи, а также проведем дополнительные тесты.

В рассматриваемой работе приведен пример игры в теннис, в которой есть 2 игрока (подающий и принимающий). Набор стратегий для каждого из них:

1. подающий может подать мяч в центр (Центр) или в открытую зону (Открытый);
2. принимающий должен быть готов к одному из этих действий.

У подающего есть 3 состояния: Агрессивный, Спокойный или Защитный. Для каждого состояния имеются свои вероятности подать мяч в центр или в открытую зону (состояния меняются согласно цепи маркова). Наша задача состоит в том, чтобы предсказать направление подачи (центр или открытая зона).

Прежде чем перейти к непосредственно к вычислениям и тестированию, введем несколько определений:

Определение 2. *Матрица выплат* — это таблица, в которой отображены выплаты каждому из участников в игре двух лиц за определенный ход. Строки таблицы показывают результаты каждого выбора хода первым игроком, а столбцы — результаты выбора второго игрока.

Определение 3. *Смешанная стратегия* — заданное вероятностное распределение ходов.

Определение 4. *Равновесие в смешанных стратегиях* — ситуация, когда ни одному из игроков не выгодно отклоняться от своей смешанной стратегии.

Определение 5. *Чистая стратегия* — частный случай смешанной стратегии, в котором одна из вероятностей равна 1, а остальные 0.

Определение 6. *Седловая точка* — вид равновесия в чистых стратегиях, характеризующий тем, что соответствующий элемент матрицы является одновременно наибольшим в своем столбце и наименьшим в своей строке [4].

Матрицы выплат для каждого из состояний в предложенной игре использовались следующие:

Обозначения:

“Открытый”— открытая зона

“Центр”— центральная зона

Таблица 2.1. Матрица выплат (Агрессивный игрок)

	Открытый	Центр
Открытый	0.35 , 0.65	0.89 , 0.11
Центр	0.98 , 0.02	0.15 , 0.85

Таблица 2.2. Матрица выплат (Умеренный игрок)

	Открытый	Центр
Открытый	0.15 , 0.85	0.8 , 0.2
Центр	0.9 , 0.1	0.15 , 0.85

Таблица 2.3. Матрица выплат (Защитный игрок)

	Открытый	Центр
Открытый	0.1 , 0.9	0.55 , 0.45
Центр	0.85 , 0.15	0.05 , 0.95

2.1.1. Вычисление элементов матрицы B

Для вычисления элементов матрицы **B** ищем равновесие в смешанных стратегиях. Подробное решение будет приведено для матрицы выплат агрессивного игрока (для остальных матриц выплат аналогично).

Рассмотрим нашу матрицу выплат:

Таблица 2.4. Матрица выплат (Агрессивный игрок)

	Открытый	Центр
Открытый	0.35 , 0.65	0.89 , 0.11
Центр	0.98 , 0.02	0.15 , 0.85

Представленная игра является игрой с постоянной суммой. При игре с постоянной суммой на каждом шаге игры общая сумма выплат игрокам фиксирована. То есть, зная выигрыш первого игрока на каждом шаге игры, мы с точностью можем сказать о выигрыше второго игрока. В таком случае для поиска равновесия в смешанных стратегиях достаточно рассматривать матрицу выплат в которой представлены выплаты только для первого игрока:

Таблица 2.5. Матрица выплат первого игрока (Агрессивный игрок)

	Открытый	Центр
Открытый	0.35	0.89
Центр	0.98	0.15

Ситуация равновесия для игры с постоянной суммой описывается следующими неравенствами:

$$H(x, \bar{y}) \leq H(\bar{x}, \bar{y}) \leq H(\bar{x}, y), \ \forall x \in S_1, \ \forall y \in S_2, \tag{2.1}$$

где

- S_1, S_2 — множество стратегий (наблюдений в нашем случае) для игрока 1 и игрока 2;
- $\bar{x} = \{x_1, x_2\}$, $\bar{y} = \{y_1, y_2\}$ — смешанные стратегии, где x_1, x_2 — вероятности подать мяч в открытую и центральную зоны первым игроком, а y_1, y_2 — вероятности принять мяч в открытой и центральной зонах вторым игроком;
- $H(x, y)$ — функция выигрыша игрока 1 в ситуации (x, y), которая определяется как математическое ожидание его выигрыша [6].

Неравенства (2.1) можно интерпретировать следующим образом: игрок 1 стремится найти стратегию, которая будет приносить ему максимальный выигрыш. В то же время игрок 2 ищет такую стратегию, при которой у него будет минимальный проигрыш.

Прежде чем искать равновесие в смешанных стратегиях, необходимо определить: есть ли равновесие в чистых стратегиях (есть ли седловая точка). В нашем случае ситуации равновесия в чистых стратегиях нет.

Таким образом, нам необходимо найти равновесие в смешанных стратегиях для игрока 1, для чего необходимо решить задачу линейного программирования:

$$\mu \longrightarrow sup$$

$$x_1, x_2 \geq 0$$

$$x_1 + x_2 = 1$$

$$0.35x_1 + 0.98x_2 - \mu \geq 0$$

$$0.89x_1 + 0.15x_2 - \mu \geq 0$$

Для решения задачи был использован пакет языка R “Rglpk”[14]. Полученное решение для матрицы выплат агрессивного игрока: $x_1 = 0.61$, $x_2 = 0.39$.

Аналогично посчитав для состояний “умеренного” и “защитного” игрока получаем следующую матрицу $\mathbf{B}$:

Таблица 2.6. Матрица $\mathbf{B}$

	Открытая зона	Центральная зона
Агрессивный игрок	0.61	0.39
Умеренный игрок	0.54	0.46
Защитный игрок	0.64	0.36

Для сравнения представим матрицу $\mathbf{B}$, полученную в рассматриваемой статье ([17]) без учета равновесия смешанных стратегий:

Таблица 2.7. Матрица **B** из статьи “Using HMM in Strategic Games”

	Открытая зона	Центральная зона
Агрессивный игрок	0.9	0.1
Умеренный игрок	0.6	0.4
Защитный игрок	0.2	0.8

Оценим работу нашего алгоритма на примере двух матриц: посчитанной нами и представленной в статье.

2.1.2. Тестирование на данных, представленных в статье

Рассмотрим для начала сценарий “Агрессивный игрок”.

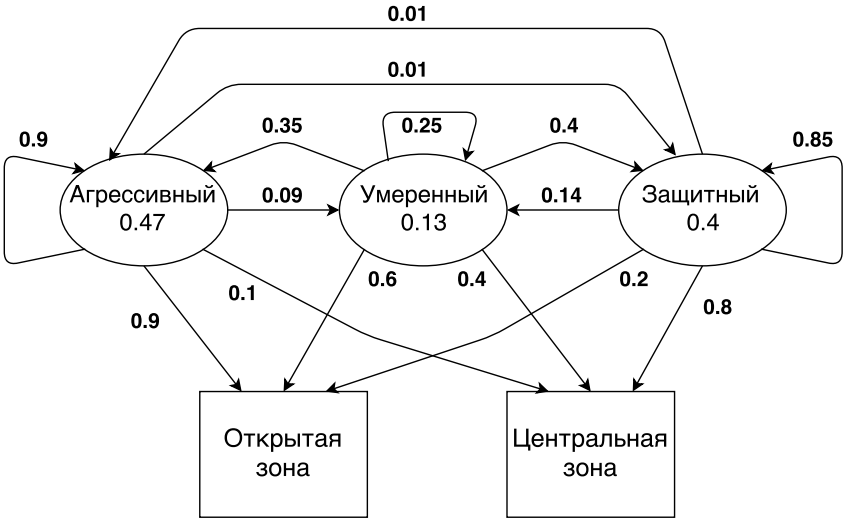

Рис. 2.1. Оригинальная CMM (Агрессивный игрок)

На каждом шаге игры применялся алгоритм Баума-Велша. Результат его работы на 1000 шаге можно видеть на рис.2.2:

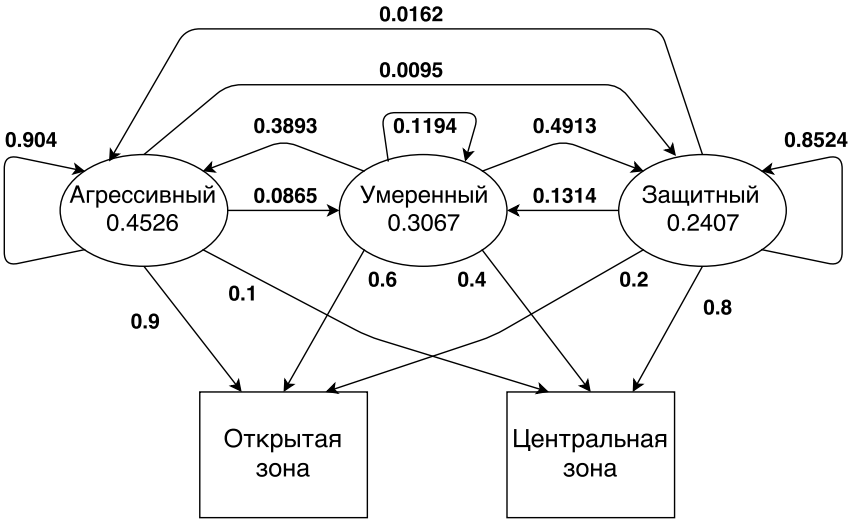

Рис. 2.2. Тренированная CMM (Агрессивный игрок) на 1000 шаге игры

После чего применив алгоритм Витерби для полученной СММ, получаем последовательность предполагаемых состояний первого игрока. Теперь, зная все необходимые сведения об игроке 1, выбираем наиболее вероятное наблюдение для игрока 2. Сравнив полученные наблюдения, считаем результат хода и повторяем алгоритм Баума-Велша и Витерби, но уже на наблюдениях, количество которых на единицу больше.

Используя матрицы выплат, мы посчитали результаты игры после каждых 200 ходов из 10000 (то есть в 50-ти партиях) и получили 47 выигрышей, что составляет 94% от общего количества игр.

Таблица 2.8. Сравнение результатов (сценарий агрессивный игрок)

	Предложенная нами модель	Модель, предложенная в статье [17]
Количество побед	94%	78%

Рассмотрим теперь аналогичным образом сценарий “Защитный игрок”. Оригинальная СММ совпадает с представленной на рис. 2.1 за исключением вектора начальных состояний: $\pi = (0.3, 0.13, 0.57)$. Тренированная СММ, после применения алгоритма Баума-Велша:

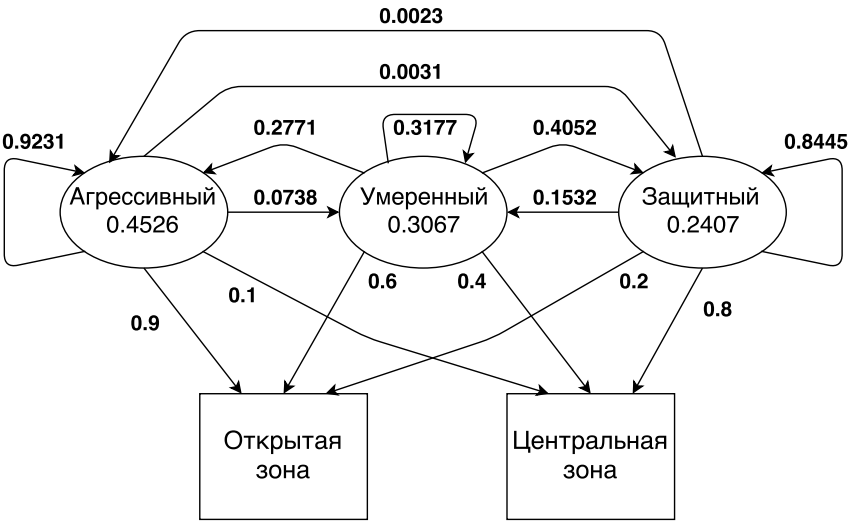

Рис. 2.3. Тренированная СММ (Защитный игрок) на 1000 шаге игры

Результат представлен в таблице:

Таблица 2.9. Сравнение результатов (сценарий защитный игрок)

	Предложенная нами модель	Модель, предложенная в статье [17]
Количество побед	92%	71%

Из полученных результатов, видим, что наш алгоритм сработал лучше, чем представленный в [17].

Особый интерес представляет количество ударов, отраженных принимающим игроком, т.е. сколько ходов из 10000 оказались выигрышными для него. Результат представлен в следующей таблице:

Таблица 2.10. Количество отраженных ударов

	Сценарий агрессивный игрок	Сценарий защитный игрок
Количество отраженных ударов	74,53%	73,67%

Также были построены 95-ти% доверительные интервалы для двух сценариев:

Для сценария агрессивного игрока: $71,42\% < x < 76,24\%$.

Для сценария защитного игрока: $71,18\% < x < 76,64\%$.

2.1.3. Тестирование на данных, найденных нами

Аналогичным образом алгоритм был протестирован для этой же игры, при тех же сценариях (агрессивный игрок и защитный игрок), но уже с матрицей $\mathbf{B}$, найденной нами. Оригинальная СММ для агрессивного игрока:

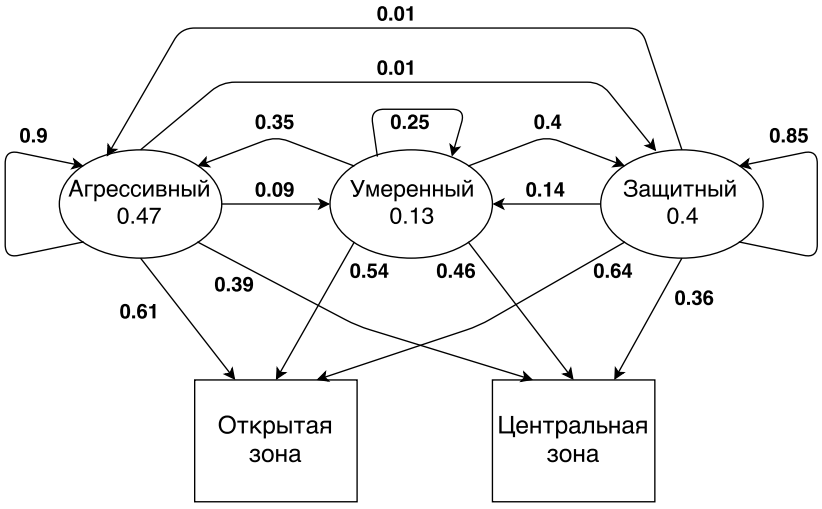

Рис. 2.4. Оригинальная СММ (агрессивный игрок)

Оригинальная СММ защитного игрока совпадает с представленной на рис. 2.4 за исключением вектора начальных состояний: $\pi = (0.3, 0.13, 0.57)$.

Результаты представлены в следующей таблице:

Таблица 2.11. Результаты при тестировании на наших данных

	Результат, посчитанный по матрицам выплат	Количество отраженных ударов
Сценарий агрессивный игрок	62%	59.24%
Сценарий защитный игрок	62%	61.34%

95-ти% доверительные интервалы для количества отраженных ударов:

Для сценария агрессивного игрока: $56,16\% < x < 63,88\%$.

Для сценария защитного игрока: $55,79\% < x < 63,52\%$.

Видим, что результаты, полученные при тестировании алгоритма на данных, использующих матрицу выплат, найденную в разделе 2.1.1, оказались значительно хуже результатов, полученных при тестировании алгоритма на данных из статьи [17]. Это можно объяснить тем, что в матрице $\mathbf{B}$, посчитанной в данной работе, вероятности подать в открытую зону и центральную очень близки друг к другу. При таких условиях достаточно большую роль в игре играет "случай", который невозможно контролировать. Таким образом, можно сделать следующий вывод: чем равномернее матрица $\mathbf{B}$, тем хуже работает алгоритм, и наоборот.

2.1.4. Сравнение с другими подходами решения поставленной задачи

В данном разделе мы сравним эффективность (количество отраженных ударов) нашего алгоритма решения поставленной задачи с результатами, полученными с использованием других подходов, а именно:

1. игрок 2 использует равномерную стратегию (его поведение описывается СММ с равномерными матрицами $\mathbf{A}$ и $\mathbf{B}$);

2. игрок 2 повторяет последний ход противника.

Эффективность предложенных подходов будет зависеть от равномерности стратегии соперника, так же как и в нашем алгоритме. Таким образом, протестируем наш алгоритм решения, а так же и предложенные подходы на нескольких различных СММ первого игрока.

Случай 1

Проверяем на СММ, представленной на рис.2.1 (считаем это неравномерным случаем по матрице $\mathbf{A}$ и матрице $\mathbf{B}$).

Результаты сравнения с предложенными подходами представлены в таблице:

Таблица 2.12. Результаты для случая 1

	Наш алгоритм	Равномерный подход	Выбор последнего действия
Отраженные удары	75.2%	49.8%	68.4%

Случай 2

Проверяем на СММ, представленной на рис.2.4 (считаем это неравномерным случаем по матрице $\mathbf{A}$ и равномерным по матрице $\mathbf{B}$).

Результаты сравнения с предложенными подходами:

Таблица 2.13. Результаты для случая 2

	Наш алгоритм	Равномерный подход	Выбор последнего действия
Отраженные удары	61%	50.9%	52.8%

Случай 3

Рассмотрим также почти равномерную матрицу **A** и неравномерную матрицу **B**. Используем следующую СММ:

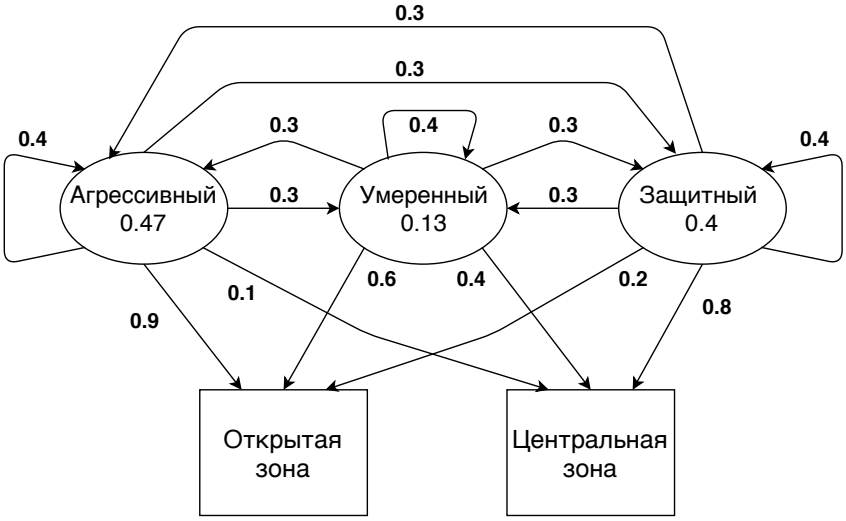

Рис. 2.5. СММ для случая 3

Результаты сравнения с предложенными подходами представлены в таблице:

Таблица 2.14. Результаты для случая 3

	Наш алгоритм	Равномерный подход	Выбор последнего действия
Отраженные удары	63.1%	50.6%	53.4%

Случай 4

И последним случаем рассмотрим почти равномерную матрицу **A** и почти равномерную матрицу **B**:

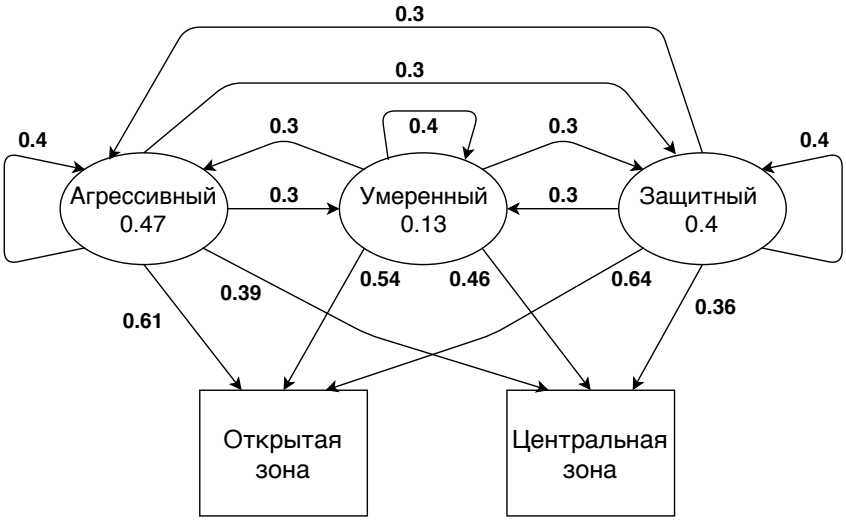

Рис. 2.6. СММ для случая 4

Результаты сравнения с предложенными подходами представлены в таблице:

Таблица 2.15. Результаты для случая 4

	Наш алгоритм	Равномерный подход	Выбор последнего действия
Отраженные удары	59.7%	50.8%	51.4%

Сравнение результатов работы предложенного алгоритма с результатами применения других подходов к решению поставленной задачи показало, что алгоритм, предложенный в данной работе, показывает лучшую эффективность работы по сравнению с рассмотренными альтернативными подходами.

2.2. Применение модели к другой игре со схожими характеристиками

В данном разделе, в порядке демонстрации неплохой эффективности и универсальности нашего алгоритма, проиллюстрируем его работу на примере другой игры со схожими характеристиками. В качестве примера возьмем известную игру “Камень, ножницы, бумага”. Правила игры следующие:

1. игроки одновременно делают ход, в котором могут поставить камень, ножницы или бумагу;
2. побеждает в ходе тот, кто выставил более выигрышную фигуру;
3. камень побеждает ножницы, ножницы побеждают бумагу, бумага побеждает камень. Рис.2.7.;
4. считается результат хода;
5. далее ход может повторяться, то есть игра является повторной.

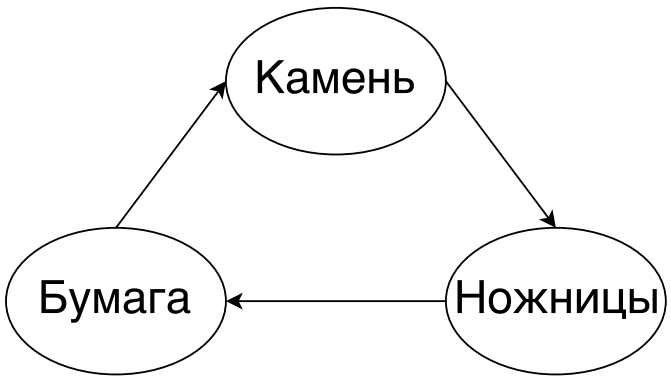

Рис. 2.7. Игра “Камень, ножницы, бумага”

Опишем рассматриваемую модель игры:

1. два игрока, один из которых не знает скрытых состояний и переходов между состояниями и наблюдениями другого;
2. первый игрок может принимать одно из трех состояний:
   . выгоднее ставить “камень”;
   . выгоднее ставить “ножницы”;
   . выгоднее ставить “бумагу”;
3. вектор начала состояний: $\pi = (\frac{1}{3}; \frac{1}{3}; \frac{1}{3})$;
4. вероятности переходов между состояниями и наблюдениями представлены на рис.2.8.

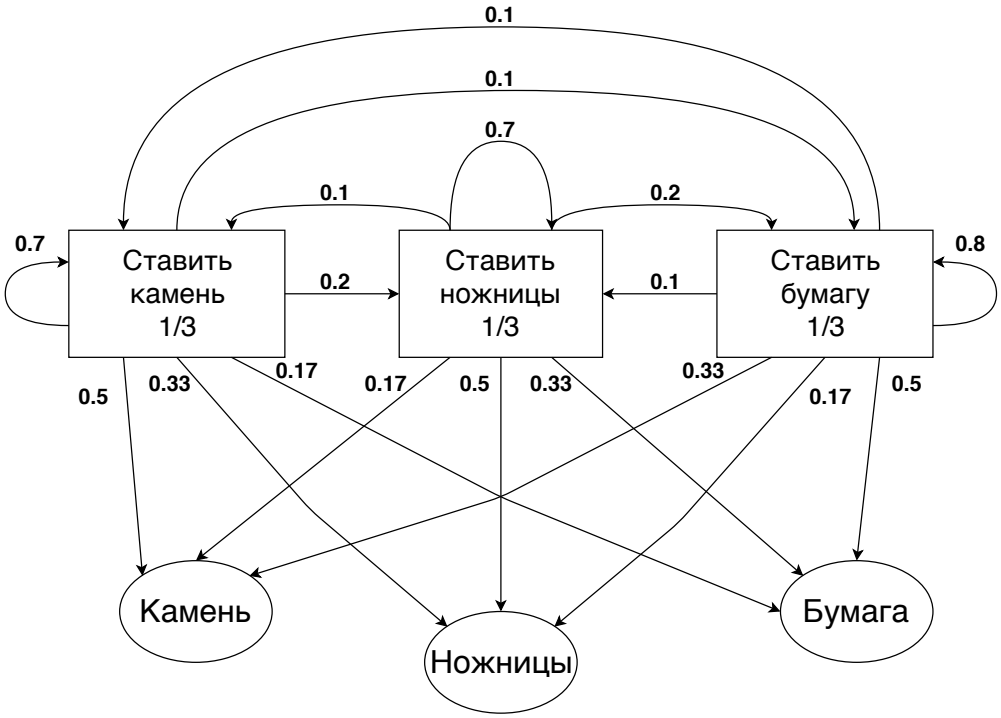

Рис. 2.8. Оригинальная СММ игры “Камень, ножницы, бумага”.

Использованы следующие матрицы выплат:

Таблица 2.16. Матрица выплат (Выгодно ставить камень)

	Камень	Ножницы	Бумага
Камень	0.5 , 0.5	1 , 0	0 , 1
Ножницы	0.5 , 0.5	0.5 , 0.5	1 , 0
Бумага	1 , 0	0 , 1	0.5 , 0.5

Таблица 2.17. Матрица выплат (Выгодно ставить ножницы)

	Камень	Ножницы	Бумага
Камень	0.5 , 0.5	1 , 0	0 , 1
Ножницы	0 , 1	0.5 , 0.5	1 , 0
Бумага	1 , 0	0.5 , 0.5	0.5 , 0.5

Таблица 2.18. Матрица выплат (Выгодно ставить бумагу)

	Камень	Ножницы	Бумага
Камень	0.5 , 0.5	1 , 0	0.5 , 0.5
Ножницы	0 , 1	0.5 , 0.5	1 , 0
Бумага	1 , 0	0 , 1	0.5 , 0.5

Множество наблюдений состоит из трех элементов: $\mathbf{O} = \{$Камень, Ножницы, Бумага$\}$. Тренированная СММ, после применения алгоритма Баума-Велша на 1000 шаге игры:

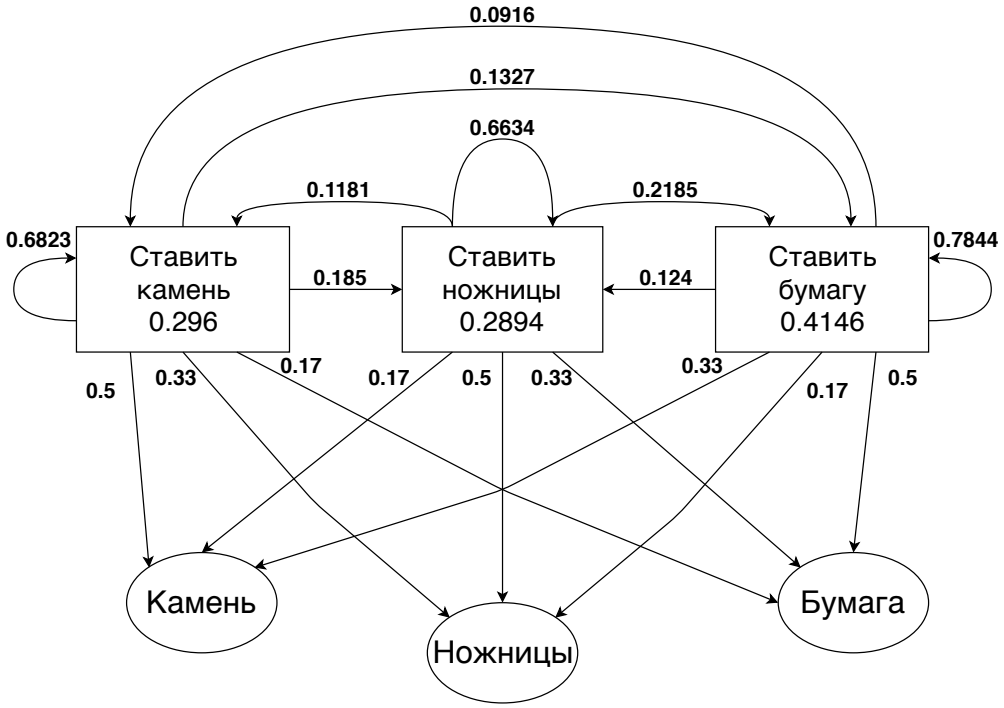

Рис. 2.9. Тренированная СММ игры “Камень, ножницы, бумага”на 1000 шаге игры

После применения алгоритма Витерби сравнив полученные последовательности состояний для оценки эффективности алгоритма Витерби, количество “угадываний” составило 79%.

Далее повторяем действия из раздела 2.1 с разницей в том, что мы для игрока 2 выбираем не наиболее вероятный ход (фигуру), а тот ход, который побеждает наиболее вероятный (побеждает наиболее вероятную фигуру).

Был посчитан результат игры по матрицам выплат после каждых 200 шагов (т.е. в 50-ти партиях), и так-же посчитано количество выигрышей (количество ситуаций, при которых игрок 2 выставил более выигрышную фигуру), и количество ситуаций “ничья” (игроки выставили одинаковые фигуры). Все результаты представлены в следующей таблице:

Таблица 2.19. Результаты работы алгоритма в игре “Камень, ножницы, бумага”

	по матрицам выплат	Количество "выигрышей"	Количество ситуаций "ничья"
Результат	76.7%	74.3%	15.2%

Полученные результаты, оказались неплохими, но можем ли мы на этом основании утверждать, что наш алгоритм примени к любой игре со схожими характеристиками? Для ответа на этот вопрос был проведен еще один тест на модели игры “Камень, ножницы, бумага” со следующими шагами:

- игра была запущена 100 раз;
- при каждом запуске игры генерировалась рандомная матрица $\mathbf{A}$, матрицы выплат по которым вычислялись элементы матрицы $\mathbf{B}$ и вектор начала состояний π;
- каждая игра содержала 10000 шагов;
- после каждой игры вычислялось количество выигрышей и ситуаций “ничья”.

По результатам проведенного теста были построены гистограммы (рис.2.10 и рис.2.11), в которых:

Среднее количество выигрышей составило: 72.3%

Среднее для ситуации “ничья” составило: 14.9%

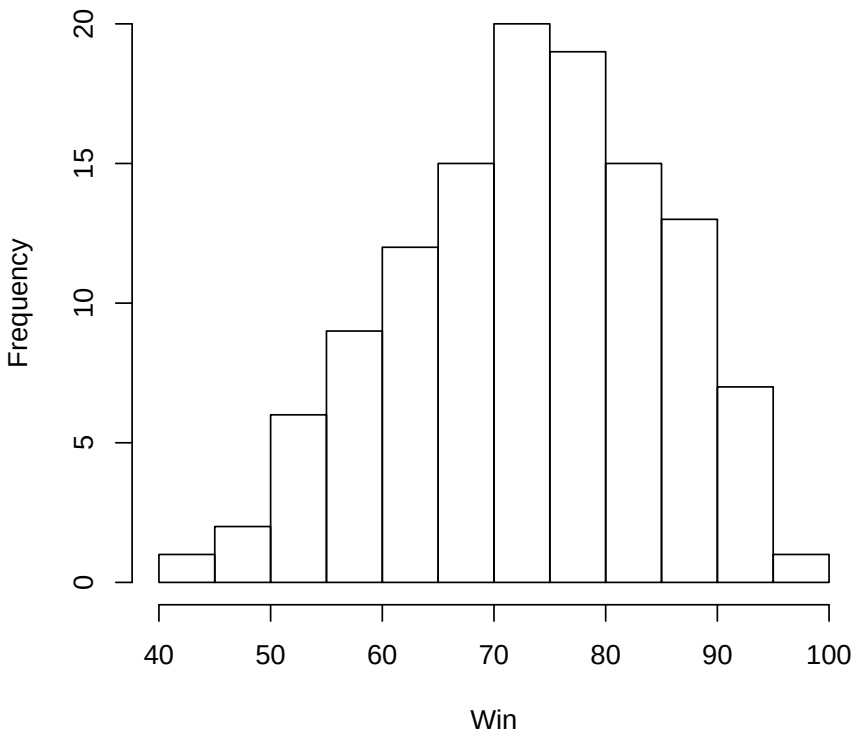

Рис. 2.10. Количество выигрышей

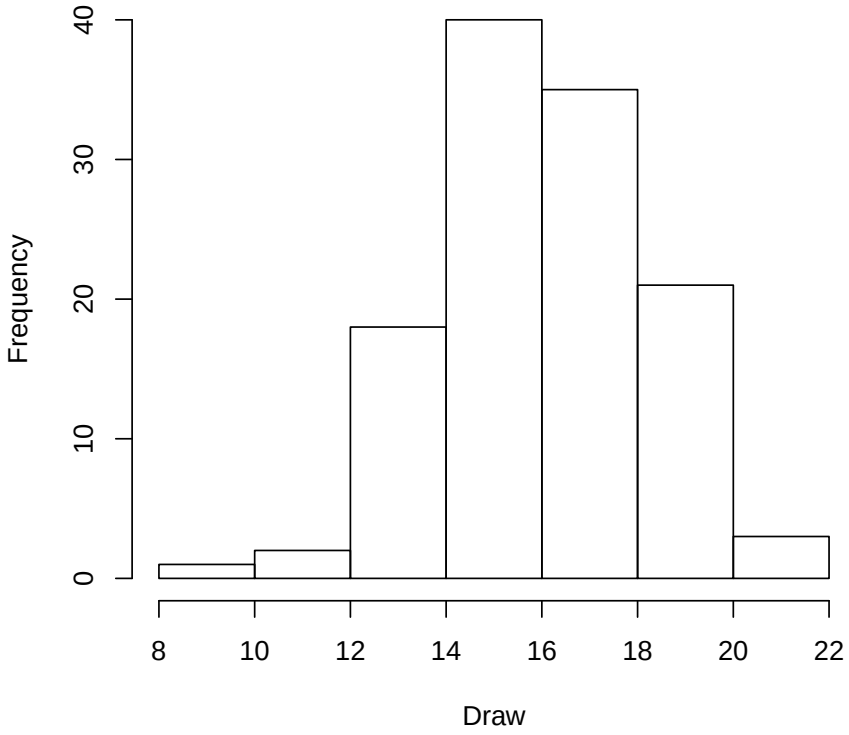

Рис. 2.11. Количество ситуаций “ничья”

Заключение

Таким образом, в данной работе:

1. представлен алгоритм решения частных случаев стохастической игры с отсутствием полной информации;
2. использованы СММ (скрытые марковские модели) в качестве инструмента описания данного процесса (стохастической игры);
3. проведен обзор литературы на близкие темы;
4. изучены алгоритмы Баума-Велша и Витерби как инструмент решения задач, описанных СММ;
5. в практической части:
   - алгоритм был протестирован на гипотетическом примере игры в “теннис”, описанном в [17]. Результаты выполнения алгоритма, предложенного в данной работе, оказались лучше результатов, представленных в [17];
   - было проведено сравнение результатов работы предложенного алгоритма с результатами применения других подходов к решению поставленной задачи. В качестве альтернативных подходов были взяты следующие: использование равномерной стратегии нашим игроком и выбор последнего хода противника. Было рассмотрено 4 случая СММ противника (комбинаций почти равномерных и неравномерных матриц $\mathbf{A}$ и $\mathbf{B}$). В каждом из случаев предложенный алгоритм отработал лучше, чем альтернативные подходы;
   - предложенный алгоритм был протестирован на произвольных данных другой игры со схожими характеристиками: “камень, ножницы, бумага”. В ходе экспериментов алгоритм показал высокую эффективность работы алгоритма, что подтверждает возможность использования предложенного алгоритма в любой игре со схожими характеристиками.

Возможные направления дальнейшей деятельности:

- применение алгоритма к играм с большим количеством игроков;
- адаптация и применение других подходов к решению такого класса игр.

Список литературы

1. Петросян Л, Зенкевич Н, Шевкопляс Е. Теория игр. — 2-е изд. — Санкт-Петербург : БХВ-Петербург, 2012. — 432 с.
2. Фон-Нейман Дж, Моргенштерн О. Теория игр и экономическое поведение. Пер. с англ. под ред. и с доб. Н. Н. Воробьева. — М. : Наука, 1970.
3. Данилов В.И. Лекции по теории игр // М.: Российская экономическая школа. — 2002. — Т. 5.
4. Оуэн Гильермо. Теория игр. Пер. с англ. И. Н. Врублевский, Г. Н. Дюбина, А. Н. Ляпунова / Под ред. А. А. Корбута. — 2-е изд. — М. : Едиториал УРСС, 2004. — 216 с.
5. Харшаньи Дж. Игры с неполной информацией // Мировая экономическая мысль. Всемирное признание (лекции нобелевских лауреатов). — 2005. — Т. 5. — С. 44–63.
6. Григорьева КВ. Методические указания Часть 1. Бескоалиционные игры в нормальной форме. // Факультет ПМ-ПУ СПбГУ. — 2007.
7. Aumann Robert J, Maschler Michael, Stearns Richard E. Repeated games with incomplete information. — Cambridge, MA. : MIT press, 1995.
8. Dymarski Przemysław. Hidden Markov models, theory and applications. — InTechOpen, 2011.
9. Fudenberg Drew, Maskin Eric. The folk theorem in repeated games with discounting or with incomplete information // A Long-Run Collaboration On Long-Run Games. — World Scientific, 2009. — P. 209–230.
10. Gibbons Robert. A primer in game theory. — Harvester Wheatsheaf, 1992.
11. Himmelmann Lin. — 2010. — URL : https://github.com/cran/HMM/blob/master/R/HMM.r.
12. Himmelmann Lin. Package 'HMM'. — 2010. — URL : https://cran.r-project.org/web/packages/HMM/HMM.pdf.
13. Martin James H, Jurafsky Daniel. Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition. — Pearson/Prentice Hall, 2009. — P. 122–141.
14. Theussl Stefan, Hornik Kurt, Buchta Christian, Schwendinger Florian. R/GNU Linear Programming Kit Interface. — 2017. — URL : https://cran.r-project.org/web/packages/Rglpk/Rglpk.pdf.
15. Rabiner Lawrence R. A tutorial on hidden Markov models and selected applications in speech recognition // Proceedings of the IEEE. — 1989. — Vol. 77, no. 2. — P. 257–286.
16. Renault Jérôme. The value of Markov chain games with lack of information on one side // Mathematics of Operations Research. — 2006. — Vol. 31, no. 3. — P. 490–512.
17. Using HMM in Strategic Games / Mario Benevides, Isaque Lima, Rafael Nader, Pedro Rougemont // Proceedings 9th International Workshop on Developments in Computational Models, Buenos Aires, Argentina, 26 August 2013 / Ed. by Mauricio Ayala-Rincón, Eduardo Bonelli, Ian Mackie. — Vol. 144 of Electronic Proceedings in Theoretical Computer Science. — Open Publishing Association, 2014. — P. 73–84.

Приложение А

Реализация алгоритма Баума-Велша из пакета “HMM”

Код взят из пакета “HMM” [11]. Приведенная ниже реализация была уже модифицирована с учетом специфики нашей задачи, а именно: была убрана оценка матрицы B и добавлена оценка вектора π.

```
1 baumWelch = function(hmm, observation, maxIterations=100, delta=1E-9,
      pseudoCount=0)
2 {
3   tempHmm = hmm
4   tempHmm$transProbs[is.na(hmm$transProbs)] = 0
5   tempHmm$emissionProbs[is.na(hmm$emissionProbs)] = 0
6   diff = c()
7   for(i in 1:maxIterations)
8   {
9     # Expectation Step (Calculate expected Transitions and Emissions)
10    bw = baumWelchRecursion(tempHmm, observation)
11    T  = bw$TransitionMatrix
12    S = bw$startProbs
13    # E  = bw$EmissionMatrix
14    # Pseudocounts
15    T[!is.na(hmm$transProbs)] = T[!is.na(hmm$transProbs)] +
          pseudoCount
16    # E[!is.na(hmm$emissionProbs)] = E[!is.na(hmm$emissionProbs)] +
          pseudoCount
17    # Maximization Step (Maximise Log-Likelihood for Transitions and
          Emissions-Probabilities)
18    T = (T/apply(T,1,sum))
19    d = sqrt(sum((tempHmm$transProbs-T)^2)) #+
          sqrt(sum((tempHmm$emissionProbs-E)^2))
20    diff = c(diff, d)
21    tempHmm$transProbs = T
22    tempHmm$startProbs = S
23    #tempHmm$emissionProbs = E
24    if(d < delta)
25    {
26      break
27    }
```

```
28    }
29    tempHmm$transProbs[is.na(hmm$transProbs)] = NA
30    tempHmm$emissionProbs[is.na(hmm$emissionProbs)] = NA
31    return(list(hmm=tempHmm, difference=diff))
32 }
33
34 baumWelchRecursion = function(hmm, observation)
35 {
36    TransitionMatrix = hmm$transProbs
37    TransitionMatrix[,] = 0
38    EmissionMatrix = hmm$emissionProbs
39    EmissionMatrix[,] = 0
40    f = forward(hmm, observation)
41    b = backward(hmm, observation)
42    probObservations = f[1,length(observation)]
43    for(i in 2:length(hmm$States))
44    {
45       j = f[i,length(observation)]
46       if(j > -Inf)
47       {
48          probObservations = j + log(1+exp(probObservations-j))
49       }
50    }
51    for(x in hmm$States)
52    {
53       for(y in hmm$States)
54       {
55          temp = f[x,1] + log(hmm$transProbs[x,y]) +
56             log(hmm$emissionProbs[y,observation[1+1]]) + b[y,1+1]
57          for(i in 2:(length(observation)-1))
58          {
59             j = f[x,i] + log(hmm$transProbs[x,y]) +
60                log(hmm$emissionProbs[y,observation[i+1]]) + b[y,i+1]
61             if(j > -Inf)
62             {
63                temp = j + log(1+exp(temp-j))
64             }
65          }
66          temp = exp(temp - probObservations)
```

```
67          TransitionMatrix[x,y] = temp
68        }
69      }
70      bwa <- backward(hmm, observation)
71      fwa <- forward(hmm, observation)
72      pia <- bwa[,1]*fwa[,1]
73      pia <- pia/sum(bwa[,1]*fwa[,1])
74      return(list(TransitionMatrix=TransitionMatrix,EmissionMatrix=EmissionMatrix
          startProbs=pia))
75  }
```

Приложение Б

Реализация алгоритма Витерби из пакета “HMM”

Код взят из пакета “HMM” [11]. Данная реализация алгоритма использовалась без изменений.

```
viterbi = function(hmm, observation)
{
  hmm$transProbs[is.na(hmm$transProbs)] = 0
  hmm$emissionProbs[is.na(hmm$emissionProbs)] = 0
  nObservations  = length(observation)
  nStates = length(hmm$States)
  v = array(NA,c(nStates,nObservations))
  dimnames(v) = list(states = hmm$States, index = 1:nObservations)
  # Init
  for(state in hmm$States)
  {
    v[state, 1] = log(hmm$startProbs[state] *
      hmm$emissionProbs[state, observation[1]])
  }
  # Iteration
  for(k in 2:nObservations)
  {
    for(state in hmm$States)
    {
      maxi = NULL
      for(previousState in hmm$States)
      {
        temp = v[previousState,k-1] +
          log(hmm$transProbs[previousState, state])
        maxi = max(maxi, temp)
      }
      v[state, k] = log(hmm$emissionProbs[state, observation[k]]) +
        maxi
    }
  }
  # Traceback
  viterbiPath = rep(NA, nObservations)
  for(state in hmm$States)
```

```
31   {
32     if(max(v[,nObservations]) == v[state,nObservations])
33     {
34       viterbiPath[nObservations] = state
35       break
36     }
37   }
38   for(k in (nObservations-1):1)
39   {
40     for(state in hmm$States)
41     {
42       if(max(v[,k]+log(hmm$transProbs[,viterbiPath[k+1]]))==v[state,
             k]+log(hmm$transProbs[state, viterbiPath[k+1]]))
43       {
44         viterbiPath[k] = state
45         break
46       }
47     }
48   }
49   return(viterbiPath)
50 }
```