ФГБОУ ВО «Петрозаводский государственный университет»
Институт математики и информационных технологий
Кафедра математического анализа

(подпись соискателя)

Сафонов Георгий Романович

Выпускная квалификационная работа бакалавра

Определение оптимальной стратегии в интернет-аукционе

Направление 01.03.01 — Математика

Научный руководитель:
к.ф.-м.н., А. А. Ивашко

(подпись руководителя)

Петрозаводск — 2021

Содержание

Введение

Системы облачных вычислений созданы для осуществления удаленного доступа к вычислительным ресурсам по средствам их временной аренды. Такой подход к получению ресурсов позволяет экономить на создании и поддержании собственных центров обработки данных. В совокупности с возможностью быстро изменять характеристики используемого аппаратного обеспечения, системы облачных вычислений быстро заняли лидирующие позиции на цифровом рынке. В современном мире, с возросшим интересом к анализу большого объема данных и применению методов машинного и глубокого обучения, а также с увеличением спроса на удаленный формат работы, популярность и востребованность облачных вычислений продолжает уверенно расти.

Сервисы облачных вычислений предоставляют разнообразные схемы использования и приобретения облачных ресурсов. Пользователю достаточно только выбрать ресурс или их совокупность и оплатить аренду, после чего он может приступать к работе с ресурсами сразу или с заведомо согласованного момента времени. Одним из крупнейших веб-сервисов по предоставлению вычислительных ресурсов является платформа Amazon EC2.

Платформа предлагает пользователям арендовать вычислительные мощности по нескольким различным правилам. Одним из наиболее доступных по цене форматов приобретения является предоставление ресурсов через аукцион. На аукцион попадают неиспользуемые в данный момент мощности платформы, а выгода при приобретении ресурса таким методом достигает 90% в сравнении с другими форматами. Во время аукциона пользователи делают ставки на доступные мощности Amazon EC2. Цены покупки инстансов постоянно меняются, а пользователям доступен просмотр истории цен за предыдущее периоды времени. Поэтому актуальной является задача оптимального определения ценовой ставки в аукционе на аренду данного вычислительного ресурса. Для определения ставки была использована модель, предложенная в [1], которая заключается в применении задачи наилучшего выбора. В данной задаче учитывается распределение динамики цен, доступных для просмотра пользователям. Для определения оптимального значения ставки для выигрыша в аукционе важным является вопрос о выборе вида распределения и определения его параметров.

Цель данной работы — исследовать модель определения ставки на аренду вычислительного ресурса с помощью задачи наилучшего выбора и применить ее, используя распределения цен в аукционе платформы Amazon EC2.

Исходя из цели были поставлены и решены следующие задачи:

- Изучено применение модели наилучшего выбора для определения оптимальной стратегии для выигрыша аукциона на аренду вычислительного ресурса.

- Собранны и исследованы данные об истории цен вычислительных ресурсов на платформе Amazon EC2.
- Построена функция распределения цен по статистическим данным и проведена оценка ее параметров с помощью EM-алгоритма.
- Проведена программная реализация итеративного алгоритма оценки параметров смеси нормальных распределений, а также алгоритмов моделирования оптимальной и конкурирующих стратегий на языке программирования *Python 3.8*.
- Проведено сравнение результатов применения стратегий на нескольких временных интервалах.

В данной работе были собраны и исследованы статистические данные по динамике цен, доступные для просмотра пользователям. Была проверена гипотеза о том, что цены подчиняются закону распределения, представляющего собой смесь нормальных распределений, и проведена оценка параметров этого распределения. Рассмотрена модель наилучшего выбора для определения оптимальной ставки на аренду вычислительного ресурса. Выполнено моделирование оптимальной и конкурирующих стратегий для реальных статистических данных платформы Amazon EC2.

1 Аукцион Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2)[6] является одной из наиболее популярных и ранних платформ по предоставлению облачных вычислительных ресурсов. Amazon EC2 представляет собой веб-сервис, позволяющий пользователям получать в аренду и использовать вычислительные ресурсы системы облачных вычислений AWS (Amazon Web Services). Система AWS размещена в нескольких центрах обработки данных, в различных регионах мира, что позволяет пользователю подобрать наиболее технически доступный ресурс. Ресурсы предоставляются пользователям в виде инстансов (виртуальных машин), клиент платит определенную сумму за час пользования инстансом, также он в праве выбрать инстансы по типу необходимых вычислений и определить для себя один из трех способов их приобретения. **Инстансы по требованию** предоставляются пользователю за полную цену и сразу при приобретении готовы к использованию, также данный вид инстансов не может быть отозван самой системой AWS. **Зарезервированные инстансы** приобретаются пользователем за более низкую цену (до 75%), чем инстансы по требованию. В пользование такой инстанс переходит в срок указанный при покупке, и данный вид инстансов также не может быть отозван. **Спотовые инстансы** (SI) являются самым дешевым видом инстансов на платформе EC2, позволяют экономить до 90% от цены на инстансы по требованию, но данный тип может быть отозван по требованию AWS.

Спотовые инстансы нашли большой спрос среди пользователей, которым не требуется постоянное использование инстанса, но необходимы вычислительные ресурсы повышенной мощности, например: проведение единоразовых вычислений, тестирование программ и их разработка. Аренда SI осуществляется с помощью аукциона на неиспользованные мощности. Пользователи делают ставки на максимальную стоимость часа пользования инстансом, которые они готовы заплатить. Если сделанная ставка выше спотовой цены на SI, то она является выигрышной. Все победившие пользователи платят одинаковую цену, которая равна значению минимальной выигравшей ставки. Спотовые инстансы обрели высокую популярность среди пользователей, которым не требуется длительное использование ресурсов, например: проведение единоразовых вычислений, обучение алгоритмов основанных на нейронных сетях, тестирование программ, хранение временных данных большого объема.

2 Модель наилучшего выбора

В данной работе мы рассматриваем спотовые инстансы. Для определения оптимальной ставки для победы на аукционе в работе [1] была предложена модель наилучшего выбора. Задачи наилучшего выбора изучаются в теории оптимальной остановки. Теория оптимальной остановки имеет дело с задачами выбора времени принятия какого-либо решения на основе последовательного наблюдения за случайными величинами с целью максимизации выигрыша. Одним из классов задач с оптимальной остановкой являются задачи наилучшего выбора, возникшие на основе реальных процессов выбора (подробнее [2], [3]). Задача наилучшего выбора имеет прямое сходство с задачей определения наилучшей ставки на облачном аукционе. Пользователь, желающий арендовать инстанс в определенный момент времени, как и в задаче наилучшего выбора, последовательно просматривает спотовые цены (случайные величины) $X_1, X_2, X_3, ...$, которые имеют распределение от p_{min} до p_{max}, с известной непрерывной функцией распределения $F(x)$. p_{min} соответствует минимальной возможной стоимости инстанса на аукционе, а p_{max} максимальной не превышающей цену данного инстанса по требованию. Пользователь имеет n периодов для того, чтобы выиграть аукцион и получить инстанс. Стратегия пользователя будет иметь пороговый вид. Он устанавливает ставку τ_i перед i-м шагом, $i = 1, ..., n$. Его цель — минимизировать ожидаемую стоимость аренды инстанса за весь период времени.

Рассматриваемая задача — это известная задача наилучшего выбора с полной информацией, решаемая методом обратной индукции. Не выиграв инстанс за период n, пользователь должен купить его по максимальной цене p_{max} . Поэтому любая цена, меньшая или равная p_{max} подходит пользователю: $\tau_n = p_{max}$. Так как спотовые цены — это независимые одинаково распределенные случайные величины с известным непрерывным распределением $F(x)$, то ожидаемая стоимость инстанса на шаге n равна $\int\limits_{p_{min}}^{\tau_n} xdF(x)$.

На $n-1$ шаге уровень цены пользователя — это минимум между текущей и ожидаемой стоимостью инстанса на n-м шаге:

$$\tau_{n-1} = E[\min\{\tau_n, x\}] = \int\limits_{p_{min}}^{p_{max}} \min\{\tau_n, x\}dF(x) = \int\limits_{p_{min}}^{\tau_n} xdF(x) + \int\limits_{\tau_n}^{p_{max}} \tau_n dF(x).$$

Здесь $E[X]$ обозначает математическое ожидание случайной величины X. Продолжая обратную индукцию, получается система рекуррентных уравнений, позволяющая

определить оптимальное значение ставки τ_i для каждого шага i для заданного периода n:

$$\begin{cases} \tau_n = p_{max}, \\ \tau_i = \int\limits_{p_{min}}^{p_{max}} \min\{\tau_{i+1}, x\} dF(x) = \int\limits_{p_{min}}^{\tau_{i+1}} x f(x) dx + \int\limits_{\tau_{i+1}}^{p_{max}} \tau_{i+1} f(x) dx; \end{cases} \tag{1}$$

где $f(x)$ — плотность распределения $F(x)$.

Если ставка τ_1 не выиграла, следующая ставка τ_2 используется на следующем шаге. Продолжая процесс, пользователь гарантированно получает инстанс за период n с минимальной ожидаемой стоимостью.

3 Построение функции распределения цен на аренду вычислительного ресурса

3.1 Исследование динамики цен

Для применения модели наилучшего выбора для определения ставки в аукционе на аренду вычислительного ресурса необходимо знать распределение цен на ресурс. Для этого была исследована динамика спотовых цен инстансов на основе исторических данных, собранных в период с 10 ноября 2018 по 20 октября 2020. По полученным значениям цен строились гистограммы относительных частот, всего было построено 204 гистограммы. Чаще всего встречаются гистограммы с одним и двумя пиками, примеры представлены на рисунках 1.1, 1.2 и 2.1, 2.2 соответственно. Реже встречаются гистограммы с тремя и более пиками (рисунки 3.1, 3.2).

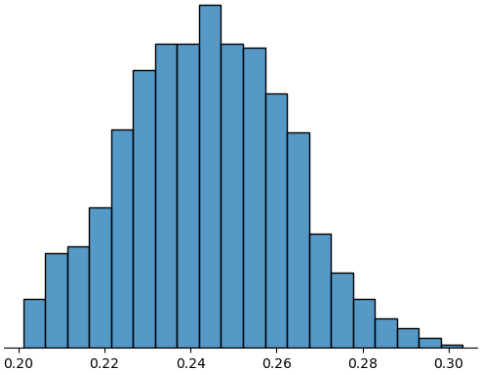

Рис. 1.1

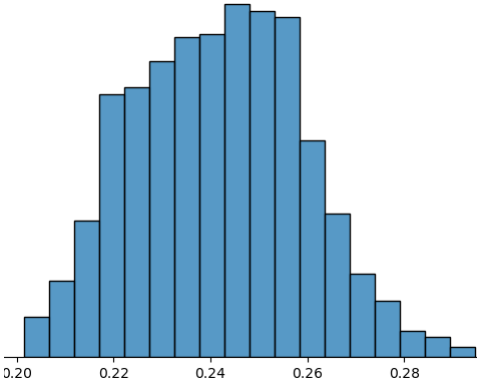

Рис. 1.2

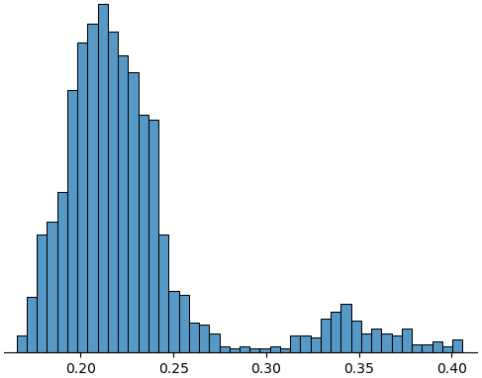

Рис. 2.1

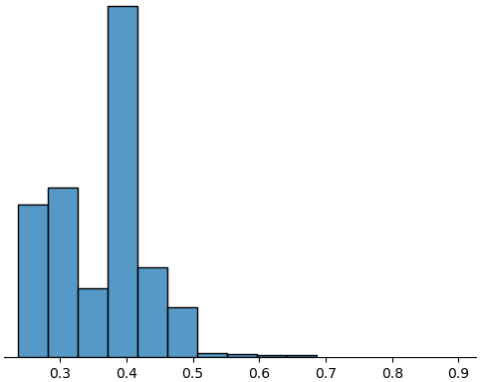

Рис. 2.2

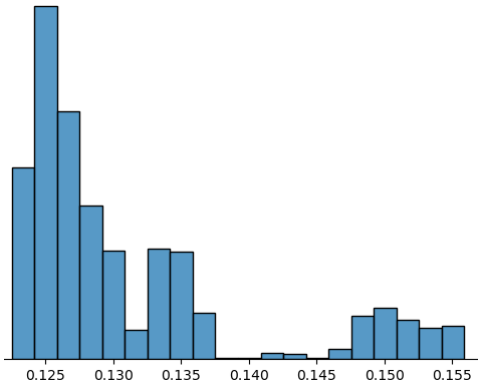

Рис. 3.1

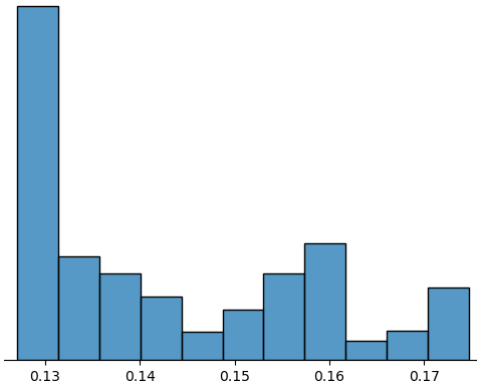

Рис. 3.2

Приведем в качестве примера построения функции распределения инстанса *us-east-1a c1.xlarge SUSE/Linux*. Для проведения моделирования стратегий мы использовали статистику динамики цен инстанса в период с 30 декабря 2019 года по 7 июля 2020 года. Всего было получено $n = 900$ значений спотовых цен, график которых приведен на рисунке 4. Из полученных значений первые 800 применяются для определения функции распределения и моделирования стратегий, а оставшиеся 100 для апробации стратегий на реальных статистических данных.

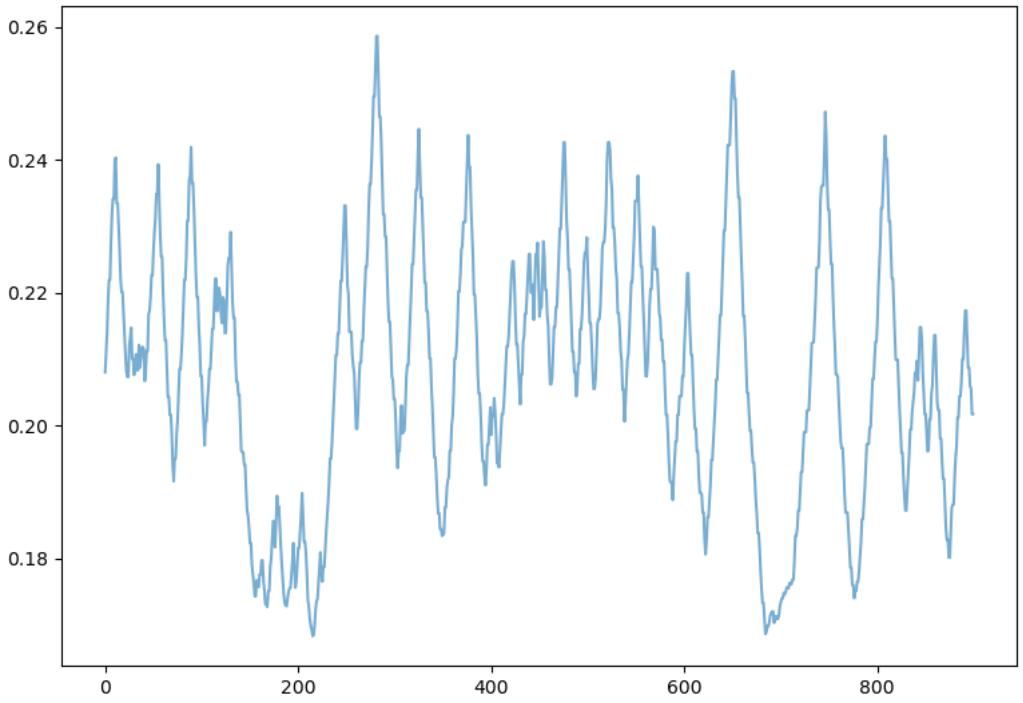

Рис. 4: Статистика цен

По полученным данным была построена гистограмма относительных частот представленная на рисунке 5:

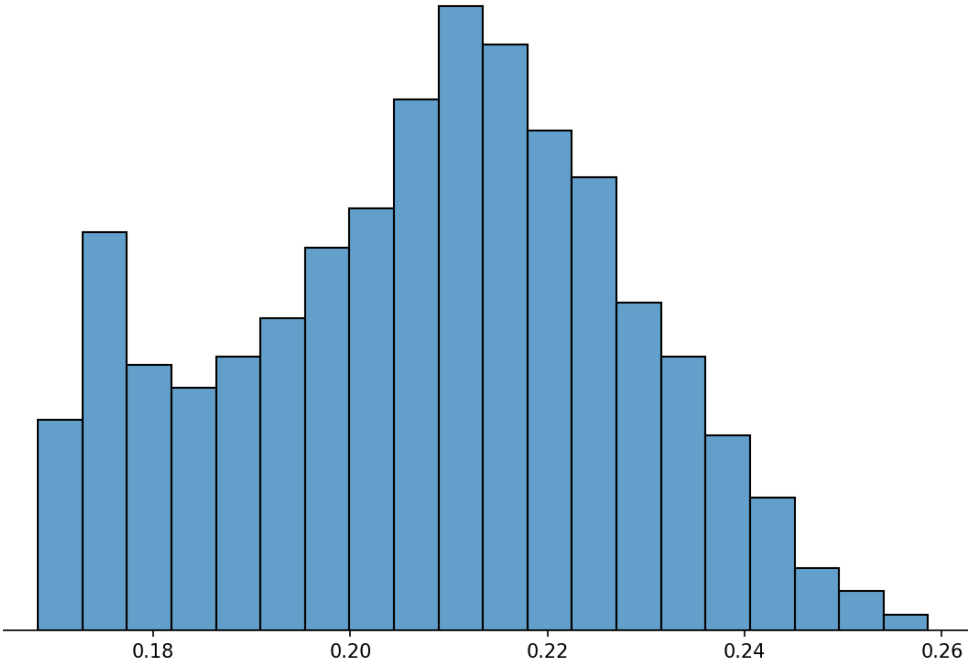

Рис. 5: Гистограмма относительных частот

В результате анализа полученной гистограммы была выдвинута гипотеза о соответствии цены на инстанс усеченной на отрезке $[p_{min}, p_{max}]$ смеси двух нормальных распределений, где $p_{min} = 0.1683$, $p_{max} = 0.2583$ - минимальное и максимальное значения цены на выбранном временном интервале. Плотность рассматриваемого распределения представлена следующей формулой:

$$f_{b_1,b_2}(x) = \begin{cases} \omega_1 \frac{C}{\sqrt{2\pi}\sigma_1} e^{\frac{-(x-a_1)^2}{2\sigma_1^2}} + \omega_2 \frac{C}{\sqrt{2\pi}\sigma_2} e^{\frac{-(x-a_2)^2}{2\sigma_2^2}}, x \in [p_{min}, p_{max}], \\ 0, x \notin [p_{min}, p_{max}]; \end{cases} \tag{2}$$

где $\omega_1 + \omega_2 = 1$, $a_1, a_2, \sigma_1, \sigma_2$ – параметры функции распределения каждой из двух компонент смеси,

$$C = \frac{1}{\int\limits_{p_{min}}^{p_{max}} \left(\omega_1 \frac{1}{\sqrt{2\pi}\sigma_1} e^{\frac{-(y-a_1)^2}{2\sigma_1^2}} + \omega_2 \frac{1}{\sqrt{2\pi}\sigma_2} e^{\frac{-(y-a_2)^2}{2\sigma_2^2}} \right) dy}$$

.

3.2 Оценка параметров распределения

Оценка неизвестных параметров $a_1, \sigma_1, a_2, \sigma_2, \omega_1, \omega_2$ проведена с помощью применения EM-алгоритма. EM-алгоритм [8] — алгоритм, использующийся для нахождения оценок максимального правдоподобия вероятностных моделей, в тех случаях, когда модель зависит от некоторых скрытых переменных. EM-алгоритм применяется для решения задач, в которых вид функции правдоподобия не допускает удобных аналитических методов

исследования, но допускает упрощения с помощью введения дополнительных скрытых переменных. К таким задачам относятся задачи кластерного анализа и задачи разделения смесей вероятностных распределений. EM-алгоритм является итерационным алгоритмом, каждая итерация состоит из двух последовательных шагов. На E-шаге (Expectation) вычисляется ожидаемое значение функции правдоподобия. На данном шаге скрытые переменные рассматриваются как наблюдаемые. На M-шаге (Maximization) вычисляется оценка максимального правдоподобия, увеличивающая ожидаемое значение, вычисляемого на шаге E.

Пусть имеется $\Theta = (\omega_1, \ldots, \omega_k, \theta_1, \ldots, \theta_k)$ — вектор искомых параметров, где k количество компонент смеси, ω_i — веса соответствующих компонент, θ_i — параметры компонент. Основной алгоритм заключается в следующем. Искусственно вводится вектор скрытых переменных G, На E шаге вводится $p(x, \theta_j)$ плотность вероятности того, что объект x был получен из j-й компоненты смеси. В соответствии с формулой условной вероятности получаем:

$$p(x, \theta_j) = p(x)\mathbf{P}(\theta_j|x) = \omega_j p_j(x),$$

где $p_j(x) = \phi(x, \theta_j)$ — это функция правдоподобия j-й компоненты смеси.

Обозначим вероятность того, что известный объект x_i был получен из j-й компоненты, как $g_{ij} = \mathbf{P}(\theta_j|x_i)$. Данные величины используются в качестве скрытых переменных. Также справедливо следующее равенство: $\sum\limits_{j=1}^{k} g_{ij} = 1$ для $i = 1, \ldots, m$, где m — количество элементов в выборке. Это равенство имеет смысл полной вероятности того, что объект x_i принадлежит одной из k компонент смеси. Из формулы Байеса получается:

$$g_{ij} = \frac{w_j p_j(x_i)}{\sum\limits_{s=1}^{k} \omega_s p_s(x_i)}.$$

Далее осуществляется переход к M-шагу, на котором решается задача максимизации правдоподобия и находится следующее приближение вектора Θ по текущим значениям векторов G и Θ. Производится максимизация логарифма полного правдоподобия:

$$Q(\Theta) = \ln \prod_{i=1}^{m} p(x_i) = \sum_{i=1}^{m} \ln \left[\sum_{j=1}^{k} \omega_j p_j(x_i) \right] \to \max_{\Theta}.$$

Далее решается оптимизационная задача Лагранжа с ограничением на то, что $\sum\limits_{j=1}^{k} \omega_j = 1$, и определяются формулы для весов ω_j компонент и соответствующих им параметров θ_j:

$$\omega_j = \frac{1}{m} \sum_{i=1}^{m} g_{ij};$$

$$\theta_j = \underset{\Theta}{\operatorname{argmax}} \sum_{i=1}^{m} g_{ij} \ln \phi(x_i, \theta).$$

Итерации последовательного выполнения шагов E и M продолжаются до тех пор, пока не будет достигнут заданный уровень сходимости.

Продемонстрируем применение EM-Алгоритма для оценки параметров смеси двух нормальных распределений.
Рассматривается вектор оцениваемых параметров $\theta = (\omega_1, \omega_2; a_1, a_2; \sigma_1, \sigma_2)$.
Плотность нормального распределения представлена в виде:

$$N(x, a_j, \sigma_j) = \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(x-a_j)^2}{2\sigma_j^2}}, \; j = 1, 2.$$

Рассмотрим шаги EM-алгоритма:

E-шаг:

$g_{ij} = \frac{\omega_j N(x_i, a_j, \sigma_j)}{\sum\limits_{s=1}^{2} \omega_s N(x_i, a_s, \sigma_s)}, \; i = 1, ..., m;$

M-шаг:

$\omega_j = \frac{1}{m} \sum\limits_{i=1}^{m} g_{ij},$

$a_j = \frac{1}{m\omega_j} \sum\limits_{i=1}^{m} g_{ij} x_i,$

$\sigma_j^2 = \frac{1}{m\omega_j} \sum\limits_{i=1}^{m} g_{ij}(x_i - a_j)^2, \quad j = 1, 2,$

где m — количество элементов в выборке, а g_{ij} — вероятность того, что элемент x_i выборки принадлежит j-ой компоненте смеси распределений.

Последовательные итерации шагов E и M повторяются до тех пор пока разность значений a_1, a_2 на шаге h с a_1, a_2 на шаге $h-1$ не будет меньше 10^{-5}. Программная реализация EM-алгоритма представленна в приложении 1. Полученные в результате применения EM-алгоритма оценки параметров распределения приведены в таблице 1.

ω_1	ω_2	a_1	a_2	σ_1	σ_2
0.1309	0.8691	0.1762	0.2125	0.0042	0.0165

Таблица 1: Полученные значения параметров

На рисунке 6 представлена гистограмма относительных частот цен и график функции плотности полученного распределения:

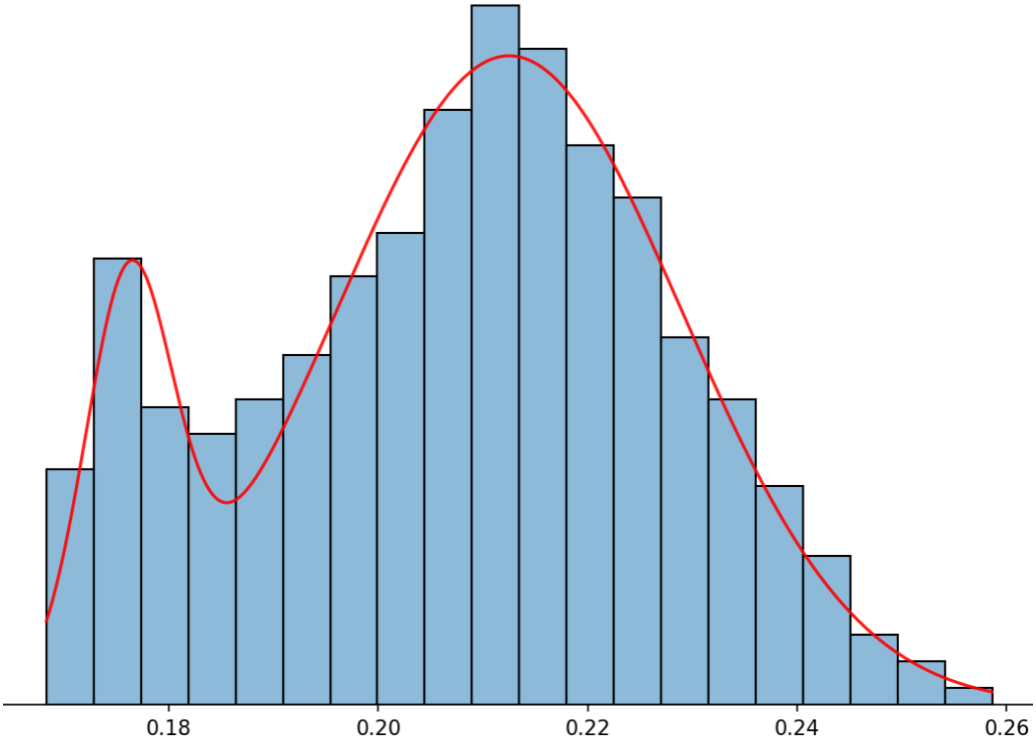

Рис. 6: Гистограмма относительных частот и плотность распределения

3.3 Проверка гипотезы о виде распределения

Проверка статистической гипотезы о виде распределения проводилась с помощью критерия Пирсона (критерий-χ^2). Нулевая гипотеза H_0 состоит в том, что частотное распределение событий, наблюдаемых в выборке (с использованием гистограммы относительных частот), согласуется со смесью двух нормальных распределений. Соответственно, альтернативная гипотеза H_1 состоит в том, что частотное распределение не имеет вида смеси двух усеченных нормальных распределений. Вариационный ряд из значений цен был разбит на $k = 20$ интервалов, $v = 6$ — количество параметров распределения, l, α — степени свободы для распределения χ^2, где $l = k - v - 1$, α — уровень значимости, в нашем случае $\alpha = 0.01$. В результате было определено значение наблюдаемой статистики критерия: $\chi^2_{\text{набл.}} = 12.67$, а значение статистики критерия со степенями свободы l и α имеет значение: $\chi^2_{13,0.01} = 27,69$. Так как $\chi^2_{\text{набл.}} < \chi^2_{13,0.01}$, то оснований отвергать гипотезу H_0 нет.

4 Моделирование стратегий

4.1 Построение оптимальной стратегии

Построим оптимальные ставки для победы в аукционе на инстанс *us-east-1a c1.xlarge SUSE/Linux* с 11 по 15 июня 2020. Предположим что для победы в аукционе пользователю доступно $n = 20$ шагов. Используя формулы (1),(2) и значения параметров распределения из таблицы 1, были получены оптимальные значения ставок для каждого шага. Полученные значения приведены в таблице 2.

Шаг	Значение ставки
1	0.177178
2	0.177498
3	0.177850
4	0.178241
5	0.178676
6	0.179166
7	0.179719
8	0.180351
9	0.181077
10	0.181920
11	0.182906
12	0.184073
13	0.185467
14	0.187153
15	0.189226
16	0.191835
17	0.195248
18	0.200035
19	0.207913
20	0.258600

Таблица 2: Значение ставок для оптимальной стратегии

4.2 Построение конкурирующих стратегий

Для оценки полезности предложенной оптимальной стратегии сравним ее с тремя другими возможными стратегиями с помощью численного моделирования. Рассмотрим следующие три стратегии:

1. **Постоянная стратегия**

   Пользователь определяет значение ставки постоянное для всех шагов. Эта ставка дает минимальное значение ожидаемой стоимости инстанса. Если за n шагов пользователь не получает инстанс, то он покупает инстанс по максимальной цене. Значение

ставки x^* определяется по формуле: $x^* = \text{argmin}(P(y))$, где

$$P(x) = \sum_{i=1}^{n} \left[\int_{x}^{p_{max}} f(y)dy \right]^{i-1} \int_{p_{min}}^{x} yf(y)dy + \left[\int_{x}^{p_{max}} f(y)dy \right]^{n} p_{max}. \quad (3)$$

В формуле (3) вычисляется ожидаемая стоимость инстанса при постоянном пороге x. Для $n = 20$ мы получили следующее значение ставки: $x^* = 0.195558$.

2. **Линейная стратегия**

Пользователь использует ставки, имеющие линейный вид: $\tau_n = ki + b$, где i номер шага, а значения k и b зависят от общего числа шагов n. Для $n = 20$ были определены значения коэффициентов: $k = 0.0045, b = 0.1638$ и получены следующие значения ставок:

Шаг	Значение ставки
1	0.168300
2	0.172815
3	0.177330
4	0.181845
5	0.186360
6	0.190875
7	0.195390
8	0.199905
9	0.204420
10	0.208935
11	0.213450
12	0.217965
13	0.222480
14	0.226995
15	0.231510
16	0.236025
17	0.240540
18	0.245055
19	0.249570
20	0.254085

Таблица 3: Значение ставок для линейной стратегии

3. **Случайная стратегия**

Ставка пользователя является случайной величиной имеющей распределение вида смеси нормальных распределений с определенными ранее параметрами. Пользователь определяет значение ставки как псевдослучайное число, которое может быть вычислено следующим образом:

- Генерируется псевдослучайное число из равномерного распределения с параметрами 0 и 1. На основании полученного значения определяется компонента смеси распределений.
- Генерируется псевдослучайное число из нормального распределения с параметрами, соответствующими выбранной компоненте. Полученное число представляет собой значение ставки для выбранного шага.

Таким образом, генерируются значения для каждого из n шагов. Для $n = 20$ были получены ставки, которые представлены в таблице 4.

Шаг	Значение ставки
1	0.236835
2	0.216236
3	0.221074
4	0.179145
5	0.202665
6	0.227695
7	0.249903
8	0.227412
9	0.198986
10	0.201799
11	0.224349
12	0.164940
13	0.212460
14	0.206461
15	0.212841
16	0.224941
17	0.200646
18	0.208014
19	0.170434
20	0.215569

Таблица 4: Значение ставок для случайной стратегии

Программная реализация алгоритмов построения каждой из стратегий приведена в приложении 2.

5 Сравнение стратегий

Для демонстрации применения стратегий рассмотрим временной промежуток с 11.06.2020 по 15.06.2020 с количеством шагов $n = 20$. На рисунке 7 представлены графики цен и полученных ставок стратегий для соответствующих шагов. Как видно по графикам значение ставки оптимальной стратегии превышает значение цены на шаге $i = 15$, а значение ставки на данном шаге следует установить равной $\tau = 0.1872$. По данной цене пользователь купит инстанс и покинет аукцион.

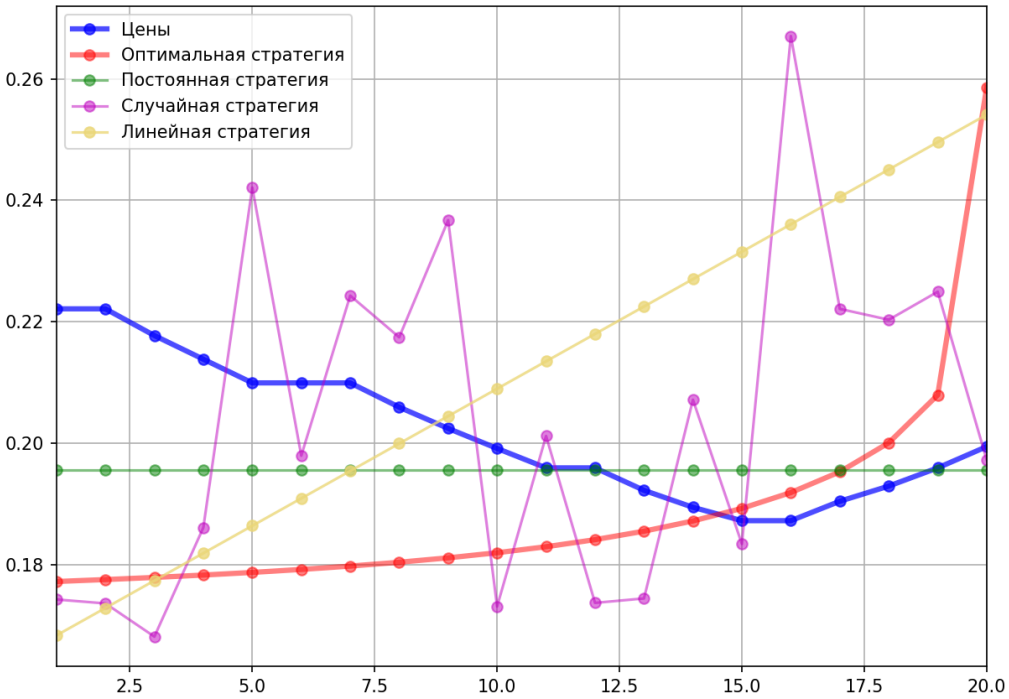

Рис. 7: График значений цен и ставок

Значения цен по котором пользователь купит инстанс в соответствии с каждой из стратегий представлены в таблице 5:

Стратегия	Шаг выхода	Цена инстанса
Оптимальная стратегия	15	0.1872
Постоянная стратегия	13	0.1922
Линейная стратегия	9	0.2024
Случайная стратегия	7	0.2099

Таблица 5: Шаги и цены выхода из аукциона

Из полученных значений видно, что оптимальная стратегия дает наименьшее значение цены выхода в сравнении с остальными рассматриваемыми стратегиями Случайная же стратегия имеет наименьший шаг выхода, однако при этом дает наибольшую цену покупки инстанса. Следовательно, оптимальная стратегия действительно дает наилучший результат с точки зрения финансовой выгоды пользователя.

Было проведено моделирование рассматриваемых стратегий на других промежутках времени состоящих из $n = 20$ шагов, в таблице 6 приведены средние цены и шаги выхода для каждой из стратегий.

Стратегия	Средний шаг выхода	Средняя цена инстанса
Оптимальная стратегия	14.6	0.1933
Линейная стратегия	9.0	0.2016
Случайная стратегия	5.0	0.2024
Постоянная стратегия	10.5	0.2099

Таблица 6: Шаги и цены выхода из аукциона

Оптимальная стратегия дает наименьшее значение средней цены в сравнении с остальными стратегиями, наихудшее значение цены покупки дает постоянная стратегия.

Заключение

В результате выполнения работы была изучена модель наилучшего выбора для определения оптимальной ставки в интернет аукционе. Для проведения моделирования стратегии была собрана и исследована история динамики цен на вычислительные ресурсы платформы Amazon EC2, на примере одного из инстансов было сделано предположение о том, что цены имеют распределение в виде смеси нормальных распределений. Применяя EM–алгоритм были получены оценки параметров распределения и проведена проверка гипотезы о виде распределения с помощью критерия χ^2 Пирсона. Для проверки полезности оптимальной стратегии было проведено сравнение с конкурирующими стратегиями на реальных значениях цен, в результате сравнения оптимальная стратегия дала наилучшие результаты.

Проведена программная реализация итеративного алгоритма оценки параметров смеси нормальных распределений, а также алгоритмов моделирования оптимальной и конкурирующих стратегий на языке программирования *Python 3.8*. Код программы приведен в приложениях 1 и 2.

Список литературы

1. Ивашко Е.Е. Эффективная по цене стратегия аренды облачных ресурсов при неопределенности цены [Текст] / Е.Е. Ивашко, А.А. Ивашко, Г.Р. Сафонов, А. Черных. // Математическая Теория Игр и ее Приложения. - г. Петрозаводск, 2019. - Т.11, вып.3. - С.5-30. (ВАК, РИНЦ)

2. Ивашко А.А. Дискретные задачи оптимальной остановки : учебное пособие для обучающихся по математическим направлениям подготовки / А. А. Ивашко ; М-во образования и науки Рос. Федерации, Федер. гос. бюджет. образоват. учреждение высш. образования Петрозавод. гос. ун-т. — Петрозаводск : Издательство ПетрГУ, 2017. — 47, [1] с.

3. Мазалов В.В., Фалько А.А. Задача наилучшего выбора и ее применение в рекламных кампаниях поисковой системы Яндекс // "Интернет-математика 2007". – Екатеринбург: Изд-во Урал. ун-та, 2007. – С. 126-134.

4. Письменный Д.Т. Конспект лекций по теории вероятностей, математической статистике и случайным процессам / Дмитрий Письменный. – 6-е изд. – М.: Айрис-пресс, 2013. – 288 с. –(Высшее образование).

5. Ивашко А.А. Теория вероятностей и математическая статистика : учеб- ное пособие для студентов физико-технического факультета / А. А. Ивашко ; М-во образования и науки Рос. Федерации, Федер. гос. бюджет. образоват. учреждение высш. образования Петрозавод. гос. ун-т. — Петрозаводск : Издательство ПетрГУ, 2016. — 75 с.

6. Amazon EC2 [Электронный ресурс]. – Режим доступа:https://aws.amazon.com/ru/ec2/. – (Дата обращения 04.03.2021).

7. Amazon EC2 Pricing History [Электронный ресурс].
– Режим доступа: https://console.aws.amazon.com/ec2sp/v2/home?region=us-east-1/spot. – (Дата обращения 10.03.2021).

8. EM-алгоритм, его модификации и обобщения [Электронный ресурс]. – Режим доступа: http://www.machinelearning.ru/wiki/index.php?title=EM-алгоритм/. – (Дата Обращения 03.20.2020)

9. Mazalov, V.V., Ivashko, A.A.: Online Auction and Optimal Stopping Game with Imperfect Observation. Intelligent Information and Database Systems. ACIIDS 2020, LNCS, volume 12033. Springer, 145–156 (2020).

10. Ivashko, E.E., Tchernykh, A., Ivashko, A.A., Safonov, G.R.: Cost-efficient strategy in clouds with spot price uncertainity. Automation and Remote Control **81**(4), 731–745 (2020). (WoS, Scopus)

11. Ivashko A., Safonov G. Optimal Strategy Modelling in an Online Auction for the Rent of Computing Resources. Proceedings of The Second International Workshop on Stochastic Modeling and Applied Research of Technology (SMARTY 2020) (Scopus).

Приложение 1

Используемые методы и программные библиотеки

```
import numpy as np # Пакет мат. функций
import random # Пакет функций для генерации псевдослучайных чисел
import scipy.integrate as integrate # Функции численного интегрирования
from scipy.optimize import minimize # Функция минимизации
from scipy.optimize import Bounds # Константные ограничения аргументов
from scipy.optimize import  LinearConstraint # Линейные ограничения аргументов
from scipy.stats import norm # Функционал для работы с нормальным распределением
```

Вычисление значения функции смеси нормальных распределений

```
def sum_normal(x,weights,means,SD):
# x-элемент выборки, weights/means/SD-параметры
    sum_ = 0
    for i in range(len(weights)):
        sum_ += weights[i] * norm.pdf(x,means[i],SD[i])
        # norm.pdf - зн-е плотности норм. распр. в точке x
        # с параметрами means[i] и SD[i]
    return(sum_)
```

Функция реализации EM-алгоритма

```
def EM(data, NOC=2):
# data - выборка, NOC - количество компонент в смеси
    if len(data) > 0 and NOC >=2:
        weights = [1/NOC for i in range(NOC)] # Список весов
        means = [random.random() for i in range(NOC)] # Список мат.ожиданий
        SD = [random.random() for i in range(NOC)] # Список СКО
        indicator = 1 # Индикатор прекращения итеративного процесса
        h_last = [[1 for i in range(len(data))],[1 for i in range(len(data))]]
        # Матрица скрытых переменных предыдущей итерации
        #(в момент 1-й итерации заполнена единицами)
        while indicator == 1:
            #E-шаг
            h = [] # Матрица скрытых переменных
            for j in range(NOC):
                h_j = []
                for i in range(len(data)):
```

```
                h_ij = (weights[j] * norm.pdf(data[i],means[j],SD[j]))/
                sum_normal(data[i],weights,means,SD)
                # Определение значений матрицы скрытых переменных
                h_j.append(h_ij)
            h.append(h_j)
        m = len(data)
        #M-шаг
        for j in range(NOC):
            weights[j] = (1/m)*sum(h[j])
            means[j] =
            (1/(weights[j]*m))*sum([h[j][i]*data[i] for i in range(m)])
            SD[j] =
            pow((1/(weights[j]*m))*sum([h[j][i]*pow(data[i]-means[j],2)
            for i in range(m)]),0.5)
        max_ = 0
        #* Критерий остановки
        # Поиск max(|h_ij-h_last_ij|)
        for i in range(len(h)):
            for j in range(len(h[i])):
                temp = abs(h[i][j]-h_last[i][j])
                if temp > max_:
                    max_ = temp
        if max_ < pow(10,-5): # Проверка критерия остановки
            indicator = 0
        else:
            h_last = h
    return(weights,means,SD)
    # Функция возвращает найденные параметры тремя списками
```

Приложение 2

Функция определения ставок для оптимальной стратегии

```
def optimal_strategy(steps,weights,means,SD, p_max, p_min, c):
# steps-кол-во шагов, weights/means/SD-параметры распределения
# p_max, p_min-макс. и мин. цена за рассматриваемый промежуток
# c-усекающая константа
    res = [0 for i in range(steps)]
    res[0] = p_max
    for i in range(1,steps):
        f1 = 0
        f2 = 0
        for j in range(len(weights)):
            f1 += integrate.quad(lambda x:
            c*x*(weights[j]*(1/(SD[j]*pow(2*np.pi,0.5)))*
            np.exp(-(pow(x-means[j],2))/(2*pow(SD[j],2)))), p_min, res[i-1])[0]
            f2 += integrate.quad(lambda x:
            c*res[i-1]*(weights[j]*(1/(SD[j]*pow(2*np.pi,0.5)))*
            np.exp(-(pow(x-means[j],2))/(2*pow(SD[j],2)))), res[i-1], p_max)[0]
        res[i] = f1+f2
    res.reverse()
    return(res)
    # Функция возвращает список значений ставок
```

Построение функции $P(x)$ для постоянной стратегии

```
def constant_sum(steps, weights,means,SD,y, p_max, p_min,c):
# steps-кол-во шагов, weights/means/SD-параметры распределения, y-параметр
# p_max, p_min-макс. и мин. цена за рассматриваемый промежуток
# c-усекающая константа
    f1 = 0
    f2 = 0
    for j in range(len(weights)):
        f1 += integrate.quad(lambda x:
        c*x*(weights[j]*(1/(SD[j]*pow(2*np.pi,0.5)))*
        np.exp(-(pow(x-means[j],2))/(2*pow(SD[j],2)))), p_min, y)[0]
        f2 += integrate.quad(lambda x:
        c*(weights[j]*(1/(SD[j]*pow(2*np.pi,0.5)))*
        np.exp(-(pow(x-means[j],2))/(2*pow(SD[j],2)))), y, p_max)[0]
```

```
    s = 0
    for i in range(1,steps):
        s += pow(f2,i-1)*f1
    s += pow(f2,steps)
    return(s)
    # Функция возвращает значение функции P(x)
```

Определение ставки для постоянной стратегии

```
def constant_strategy(steps,weights,means,SD,data,c):
# steps-кол-во шагов, weights/means/SD-параметры распределения
#data-выборка
# p_max, p_min-макс. и мин. цена за рассматриваемый промежуток
# c-усекающая константа
    res = minimize(lambda y:
    constant_sum(steps, weights,means,SD,y,max(data), min(data),c),
    min(data)+0.00001, method='TNC', bounds=[(0, max(data))], tol=0.0001)
    # Определение argmin(P(x))
    return([float(res.x[0]) for i in range(steps)])
    # Функция возвращает список заполненный зн-ем постоянной ставки
```

Построение функции $P(a, b)$ для линейной стратегии

```
def linear_sum(steps, weights, means, SD, params, p_max, p_min, c):
# steps-кол-во шагов, weights/means/SD-параметры распределения
#params-список параметров (a и b из ур-я t=a*i+b)
# p_max, p_min-макс. и мин. цена за рассматриваемый промежуток
# c-усекающая константа
    s = 0
    for step in range(1,steps):
        f1 = 0
        f2 = 0
        for j in range(len(weights)):
            f1 += integrate.quad(lambda x:
            c*x*(weights[j]*(1/(SD[j]*pow(2*np.pi,0.5)))*
            np.exp(-(pow(x-means[j],2))/(2*pow(SD[j],2)))),
            p_min, params[0]*step+params[1])[0]
            f2 += integrate.quad(lambda x:
            c*(weights[j]*(1/(SD[j]*pow(2*np.pi,0.5)))*
            np.exp(-(pow(x-means[j],2))/(2*pow(SD[j],2)))),
            params[0]*step+params[1], p_max)[0]
```

```
        s += pow(f2,step)*f1
    f2 = 0
    for j in range(len(weights)):
            f2 += integrate.quad(lambda x:
            c*(weights[j]*(1/(SD[j]*pow(2*np.pi,0.5)))*
            np.exp(-(pow(x-means[j],2))/(2*pow(SD[j],2)))),
            params[0]*step+params[1], p_max)[0]
    s += pow(f2,steps)
    # Возвращается значение функции P(a,b)
```

Определение ставок для линейой стратегии

```
def linear_strategy(steps, weights,means, SD, data,c):
# steps-кол-во шагов, weights/means/SD-параметры распределения
#data-выборка
# p_max, p_min-макс. и мин. цена за рассматриваемый промежуток
# c-усекающая константа
    linear = LinearConstraint([[1,1],[steps,1]],[min(data),
    min(data)+(max(data)-min(data))/2],
    [min(data)+(max(data)-min(data))/2, max(data)])
    # Линейные ограничения для минимизации
    bnds = ((0,(max(data)-min(data))/steps),(0,max(data)))
    x_00 = bnds[0][0]+(bnds[0][1] - bnds[0][0])/2
    x_01 = bnds[1][0]+(bnds[0][1] - bnds[1][0])/2
    x0 = [x_00,x_01]
    # Константные ограничения для минимизации
    res = minimize(lambda vect:
    linear_sum(steps,weights,means,SD,vect,max(data),min(data),c)
    ,x0, method='SLSQP', constraints=linear,bounds= bnds, tol=0.001)
    # Определение значений параметров a и b
    # при которых P(a,b) достигает своего минимума
    k = res.x[0]
    b = res.x[1]
    res_bits = []
    for step in range(steps):
        print(step+1)
        res_bits.append(k*(step+1)+b)
    return(res_bits)
    # Функция возвращает зн-я ставок для линейной стратегии
```

Геренерация псевдослучайного числа из смеси нормальных распрелений

```
def random_mixture_value(weights, means, SD):
# weights/means/SD-параметры распределения
    p = random.random()
    element = 0
    weights_sort = sorted(weights)
    for weight in weights_sort:
        if p <= weight:
            element = weights.index(weight)
            break
    num = random.gauss(means[element],SD[element])
    return(num)
    # Функция возвращает сгенерированное число
```

Определение ставок для случайной стратегии

```
def random_strategy(steps,weights,means,SD):
# steps-кол-во шагов
# weights/means/SD-параметры распределения
    res = []
    for i in range(steps):
        res.append(random_mixture_value(weights,means,SD))
    return(res)
    # Функция возвращает список зн-й для ставок
```