

Филиал Московского Государственного Университета  
имени М.В. Ломоносова в г. Ташкенте

Факультет прикладной математики и информатики

Кафедра прикладной математики и информатики

Мурадасилов Руслан Серверович

## ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

на тему: «ЕМ-алгоритм и его применение в статистических  
задачах»

по направлению 01.03.02 «Прикладная математика и  
информатика»

ВКР рассмотрена и рекомендована к защите

зав. кафедрой «ПМиИ», к.ф.-м.н., доцент \_\_\_\_\_ Строгалов А. С.

Научный руководитель:

д.ф.-м.н., профессор \_\_\_\_\_ Абдушукуров А. А.

«\_\_\_\_\_» \_\_\_\_\_ 2021 г.

Ташкент 2021

## **Аннотация**

В данной работе рассматривается ЕМ-алгоритм и его применение в статистических задачах.

В первой части работы описан сам ЕМ-алгоритм, его свойства и модификации. Вторая часть работы представляет собой исследование по применению ЕМ-алгоритма к цензурированной выборке на языке программирования Python с помощью модуля GaussianMixture библиотеки Scikit-Learn и модуля Stats библиотеки SciPy.

**Ключевые слова:** цензурированная выборка, ЕМ-алгоритм, GaussianMixture

## **Abstract**

This paper is devoted to EM-algorithm and its application in statistical problems.

The first part of the paper describes EM-algorithm, its properties and modifications. The second part of the paper represents a research on application of EM-algorithm to censored data on Python programming language using the GaussianMixture module of Scikit-Learn library and Stats module of SciPy library.

**Keywords:** censored data, EM-algorithm, GaussianMixture

# Содержание

<b>1</b>	<b>Введение</b>	<b>4</b>
<b>2</b>	<b>Историческая часть вопроса</b>	<b>5</b>
<b>3</b>	<b>Основные результаты</b>	<b>6</b>
3.1	Описание ЕМ-алгоритма . . . . .	6
3.1.1	Задача разделения смеси распределений . . . . .	6
3.1.2	ЕМ-алгоритм . . . . .	6
3.1.3	Смеси нормальных распределений . . . . .	10
3.2	Применение ЕМ-алгоритма . . . . .	15
3.2.1	Проверка работоспособности GaussianMixture . . . .	15
3.2.2	Применение ЕМ-алгоритма к цензурированной вы- борке . . . . .	22
<b>4</b>	<b>Заключение</b>	<b>28</b>
<b>5</b>	<b>Список литературы</b>	<b>29</b>

# 1 Введение

ЕМ-алгоритм в математической статистике используется для нахождения оценок максимального правдоподобия параметров вероятностной модели, в случае, когда модель зависит от некоторых скрытых данных. Как правило, ЕМ-алгоритм применяется при решении задач двух типов.

К первому типу относятся задачи с *действительно* неполными данными, когда некоторые статистические данные отсутствуют в силу каких-либо причин. Ко второму же типу можно отнести задачи, в которых удобно вводить скрытые переменные для упрощения подсчета функции правдоподобия. Примером такой задачи может служить кластеризация.

В данной работе приведено описание ЕМ-алгоритма и его свойства, а также предложен пример с его применением к задаче первого типа.

В первой части работы дан теоретический материал, в котором подробно изложены Е-шаг и М-шаг в общем случае, а также на примере разделения смеси нормальных распределений.

Вторая часть работы посвящена применению ЕМ-алгоритма с использованием модуля GaussianMixture библиотеки Scikit-Learn на языке Python. В первую очередь были сгенерированы разные выборки с помощью модуля Stats библиотеки SciPy, на основе которых была проведена проверка работоспособности и степень точности оценки алгоритма на данных разных распределений. Далее алгоритм был применен к цензурированной выборке, предварительно предобработанной с помощью оценки её функции распределения  $F_n^{RR}$ , предложенной Абдушукуровым А. А.

## 2 Историческая часть вопроса

Одним из первых ЕМ-алгоритм был предложен McKendrick (1926) для медицинских приложений. Затем после довольно большого перерыва эта идея вновь возникла в работах Healy and Westmacott (1956), Шлезингера (1965, 1968), Day (1969), Wolfe (1970), а затем развита и систематически исследована в работе Dempster, Laird and Rubin (1977) [1]. Само название *ЕМ-алгоритм* было предложено в работе [1], в которой показана высокая общность алгоритма. Возможно, поэтому зарубежные источники традиционно ссылаются на эту статью, как на первую работу по ЕМ-алгоритму.

Основные свойства ЕМ-алгоритма описаны еще в работе Шлезингера (1965) [2]. Позднее в работах Dempster, Laird and Rubin (1977), Everitt and Hand (1981), Wu (1983), Boyles (1983), Redner and Walker (1984) эти свойства были передоказаны и развиты.

Литература по ЕМ-алгоритму и его применениям к решению задач из конкретных областей обширна. Среди них можно выделить книги, посвященные собственно ЕМ-алгоритму Литтла и Рубина (1991), McLachlan and Krishnan (1997), книги, в которых ЕМ-алгоритму уделено значительное место Айвазяна и др. (1989) [3], Tanner (1993), а также двух обстоятельных работ Bilmes (1998) и Figueiredo (2004).

## 3 Основные результаты

### 3.1 Описание ЕМ-алгоритма

#### 3.1.1 Задача разделения смеси распределений

В тех случаях, когда «форму» класса не удаётся описать каким-либо одним распределением, можно попробовать описать её смесью распределений.

Пусть плотность распределения на  $X$  имеет вид смеси  $k$  распределений:

$$p(x) = \sum_{j=1}^k w_j p_j(x), \quad \sum_{j=1}^k w_j = 1, \quad w_j \geq 0,$$

где  $p_j(x)$  — функция правдоподобия  $j$ -ой компоненты смеси,  $w_j$  — её априорная вероятность. Функции правдоподобия принадлежат параметрическому семейству распределений  $\varphi(x; \theta)$  и отличаются только значениями параметра,  $p_j(x) = \varphi(x; \theta_j)$ .

Иными словами, «выбрать объект  $x$  из смеси  $p(x)$ » означает сначала выбрать  $j$ -ю компоненту смеси из дискретного распределения  $\{w_1, \dots, w_k\}$ , затем выбрать объект  $x$  согласно плотности  $p_j(x)$ .

*Задача разделения смеси* заключается в том, чтобы, имея выборку  $X = (X_1, \dots, X_n)$  —  $n$  случайных независимых наблюдений размерности  $m$  из смеси  $p(x)$ , зная число  $k$  — количество распределений в смеси, называемых кластерами, и функцию  $\varphi$ , оценить вектор параметров  $\Theta = (w_1, \dots, w_k, \theta_1, \dots, \theta_k)$ .

#### 3.1.2 ЕМ-алгоритм

К сожалению, попытка разделить смесь, используя принцип максимума правдоподобия «в лоб», приводит к слишком громоздкой оптимизационной задаче. Обойти эту трудность позволяет алгоритм ЕМ (expectation-maximization). Идея алгоритма заключается в следующем. Искусственно вводится вспомогательный вектор скрытых (hidden) переменных  $G$ , обладающий двумя замечательными свойствами. С одной

стороны, он может быть вычислен, если известны значения вектора параметров  $\Theta$ . С другой стороны, поиск максимума правдоподобия сильно упрощается, если известны значения скрытых переменных.

ЕМ-алгоритм состоит из итерационного повторения двух шагов. На Е-шаге вычисляется ожидаемое значение (expectation) вектора скрытых переменных  $G$  по текущему приближению вектора параметров  $\Theta$ . На М-шаге решается задача максимизации правдоподобия (maximization) и находится следующее приближение вектора  $\Theta$  по текущим значениям векторов  $G$  и  $\Theta$ .

**Е-шаг (expectation).** Обозначим через  $p(x, \theta_j)$  плотность вероятности того, что объект  $x$  получен из  $j$ -й компоненты смеси. По формуле условной вероятности

$$p(x, \theta_j) = p(x)P(\theta_j | x) = w_j p_j(x).$$

Введём обозначение  $g_{ij} \equiv P(\theta_j | x_i)$ . Это неизвестная апостериорная вероятность того, что обучающий объект  $x_i$  получен из  $j$ -й компоненты смеси. Возьмём эти величины в качестве скрытых переменных. Обозначим  $G = (g_{ij})_{m \times k} = (g_1, \dots, g_j)$ , где  $g_j$  —  $j$ -й столбец матрицы  $G$ . Каждый объект обязательно принадлежит какой-то компоненте, поэтому справедлива формула полной вероятности:

$$\sum_{j=1}^k g_{ij} = 1, \quad i = 1, \dots, m$$

Зная параметры компонент  $w_j, \theta_j$ , легко вычислить  $g_{ij}$  по формуле Байеса:

$$g_{ij} = \frac{w_j p_j(x_i)}{\sum_{s=1}^k w_s p_s(x_i)} \quad \forall i, j. \quad (1)$$

В этом и заключается Е-шаг алгоритма ЕМ.

**М-шаг (maximization).** Покажем, что знание значений скрытых переменных  $g_{ij}$  и принцип максимума правдоподобия приводят к оптимизационной задаче, допускающей эффективное численное (или даже

аналитическое) решение. Будем максимизировать логарифм правдоподобия

$$Q(\Theta) = \ln \prod_{i=1}^m p(x_i) = \sum_{i=1}^m \ln \sum_{j=1}^k w_j p_j(x_i) \rightarrow \max_{\Theta}$$

при ограничении  $\sum_{j=1}^k w_j = 1$ . Запишем лагранжиан этой оптимизационной задачи:

$$L(\Theta; X^m) = \sum_{i=1}^m \ln \left( \sum_{j=1}^k w_j p_j(x_i) \right) - \lambda \left( \sum_{j=1}^k w_j - 1 \right).$$

Приравняем к нулю производную лагранжиана по  $w_j$ :

$$\frac{\partial L}{\partial w_j} = \sum_{i=1}^m \frac{p_j(x_i)}{\sum_{s=1}^k w_s p_s(x_i)} - \lambda = 0, \quad j = 1, \dots, k. \quad (2)$$

Умножим левую и правую части на  $w_j$ , просуммируем все  $k$  этих равенств, и поменяем местами знаки суммирования по  $j$  и по  $i$ :

$$\sum_{i=1}^m \underbrace{\sum_{j=1}^k \frac{w_j p_j(x_i)}{\sum_{s=1}^k w_s p_s(x_i)}}_{=1} = \lambda \underbrace{\sum_{j=1}^k w_j}_{=1},$$

откуда следует  $\lambda = m$ .

Теперь снова умножим левую и правую части (2) на  $w_j$ , подставим  $\lambda = m$ , и, замечая сходство с формулой (1), получим выражение весов компонент через скрытые переменные:

$$w_j = \frac{1}{m} \sum_{i=1}^m \frac{w_j p_j(x_i)}{\sum_{s=1}^k w_s p_s(x_i)} = \frac{1}{m} \sum_{i=1}^m g_{ij}, \quad j = 1, \dots, k. \quad (3)$$

Легко проверить, что ограничения-неравенства  $w_j > 0$  будут выполнены на каждой итерации, если они выполнены для начального приближения.

Приравняем к нулю производную лагранжиана по  $\theta_j$ , помня, что  $p_j(x) \equiv \varphi(x; \theta_j)$ :



$$\begin{aligned}\frac{\partial L}{\partial \theta_j} &= \sum_{i=1}^m \frac{w_j}{\sum_{s=1}^k w_s p_s(x_i)} \frac{\partial}{\partial \theta_j} p_j(x_i) = \sum_{i=1}^m \frac{w_j p_j(x_i)}{\sum_{s=1}^k w_s p_s(x_i)} \frac{\partial}{\partial \theta_j} \ln p_j(x_i) = \\ &= \sum_{i=1}^m g_{ij} \frac{\partial}{\partial \theta_j} \ln p_j(x_i) = \frac{\partial}{\partial \theta_j} \sum_{i=1}^m g_{ij} \ln p_j(x_i) = 0, \quad j = 1, \dots, k.\end{aligned}$$

Полученное условие совпадает с необходимым условием максимума в задаче максимизации взвешенного правдоподобия

$$\theta_j = \underset{\theta}{argmax} \sum_{i=1}^m g_{ij} \ln \varphi(x_i, \theta), \quad j = 1, \dots, k \quad (4)$$

Таким образом, М-шаг сводится к вычислению весов компонент  $w_j$  как средних арифметических (3) и оцениванию параметров компонент  $\theta_j$  путём решения  $k$  независимых оптимизационных задач (4). Отметим, что разделение переменных оказалось возможным благодаря удачному введению скрытых переменных. Условия сходимости алгоритма ЕМ рассматриваются в работах [1], [6].

**Критерий останова.** Итерации останавливаются, когда значения функционала  $Q(\Theta)$  или скрытых переменных  $G$  перестают существенно изменяться. Удобнее контролировать скрытые переменные, так как они имеют смысл вероятностей и принимают значения из отрезка  $[0, 1]$ . На Е-шаге вычисляется матрица скрытых переменных  $G$  по формуле (1). На М-шаге решается серия из  $k$  задач максимизации взвешенного правдоподобия (4), каждая из них — по полной выборке  $X$  с вектором весов  $g_j$ .

**Проблема выбора начального приближения.** Хотя алгоритм ЕМ сходится при достаточно общих предположениях, скорость сходимости может существенно зависеть от «удачности» начального приближения. Сходимость ухудшается в тех случаях, когда делается попытка поместить несколько компонент в один фактический сгусток распределения, либо разместить компоненту посередине между сгустками.

Стандартная (но далеко не самая лучшая) эвристика заключается в том, чтобы выбрать параметры компонент случайным образом. Более разумная идея — найти в выборке  $k$  объектов, максимально удалённых друг от друга, и именно в этих точках разместить компоненты.

**Проблема выбора числа компонент  $k$ .** До сих пор предполагалось, что число компонент  $k$  известно заранее. На практике это, как правило, не так. Иногда число компонент удаётся оценить визуально, спроецировав выборку на плоскость каким-либо способом и определив число сгустков точек на полученном графике. Однако визуальный подход обладает очевидными недостатками: проецирование искажает структуру выборки, а необходимость обращаться к эксперту исключает возможность автоматического анализа данных.

Существует ещё один приём — решить задачу несколько раз при последовательных значениях  $k$ , построить график зависимости правдоподобия выборки  $Q(\Theta)$  от  $k$ , и выбрать наименьшее  $k$ , при котором график претерпевает резкий скачок правдоподобия. Это называется критерием «крутого склона». К сожалению, он также не лишён недостатков. Во-первых, существенно увеличиваются затраты времени. Во-вторых, если данные плохо описываются моделью компонент  $\varphi(x; \theta)$ , то «крутой склон» может не наблюдаться. Наличие крутого склона свидетельствует о том, что модель компонент была выбрана удачно.

### 3.1.3 Смеси нормальных распределений

Пусть компоненты смеси имеют  $n$ -мерные нормальные распределения  $\varphi(x; \theta_j) = \mathcal{N}(x; \mu_j, \Sigma_j)$  с параметрами  $\theta_j = (\mu_j, \Sigma_j)$ , где  $\mu_j \in \mathbf{R}^n$  — вектор математического ожидания,  $\Sigma_j \in \mathbf{R}^{n \times n}$  — ковариационная матрица,  $j = 1, \dots, k$ .

Тогда стационарная точка оптимизационной задачи (4) имеет вид

$$\hat{\mu}_j = \frac{1}{mw_j} \sum_{i=1}^m g_{ij} x_i, \quad j = 1, \dots, k$$

$$\hat{\sum}_j = \frac{1}{mw_j} \sum_{i=1}^m g_{ij}(x_i - \hat{\mu}_j)(x_i - \hat{\mu}_j)^T, \quad j = 1, \dots, k$$

Таким образом, М-шаг сводится к вычислению выборочного среднего и выборочной ковариационной матрицы для каждой компоненты смеси. При этом для каждой компоненты используется своё распределение весов объектов. Вес  $i$ -го объекта для  $j$ -й компоненты равен  $g_{ij}$  — оценке принадлежности данного объекта данной компоненте, вычисленной на Е-шаге.

Смеси многомерных нормальных распределений позволяют приближать любые непрерывные плотности вероятности. Они являются универсальными аппроксиматорами плотностей, подобно тому, как полиномы являются универсальными аппроксиматорами непрерывных функций. В практических задачах это позволяет восстанавливать функции правдоподобия классов даже в тех случаях, когда на первый взгляд для выполнения исходных условий нет содержательных оснований.

Недостатком гауссовских смесей является необходимость обращаться ковариационные матрицы. Это трудоёмкая операция. Кроме того, ковариационные матрицы нередко оказываются вырожденными или плохо обусловленными. Тогда возникает проблема неустойчивости выборочных оценок плотности и самого классификатора. Стандартные приёмы (регуляризация, метод главных компонент) позволяют справиться с этой проблемой. Но есть и другой выход — использовать для описания компонент более простые распределения, например, сферические.

### Частные случаи

1.  $m = 1$  — одномерные наблюдения, т. е. точки,  $k = 2$  — два гауссовых распределения  $a$  и  $b$  с неизвестными параметрами  $(\mu, \sigma^2)$ .

Оценка этих параметров была бы тривиальной, если бы мы знали, какому из распределений принадлежит каждое из наблюдений:

$$\mu_a = \frac{x_1 + x_2 + \dots + x_{n_a}}{n_a}, \quad \sigma_a^2 = \frac{(x_1 - \mu_a)^2 + (x_2 - \mu_a)^2 + \dots + (x_{n_a} - \mu_a)^2}{n_a}$$

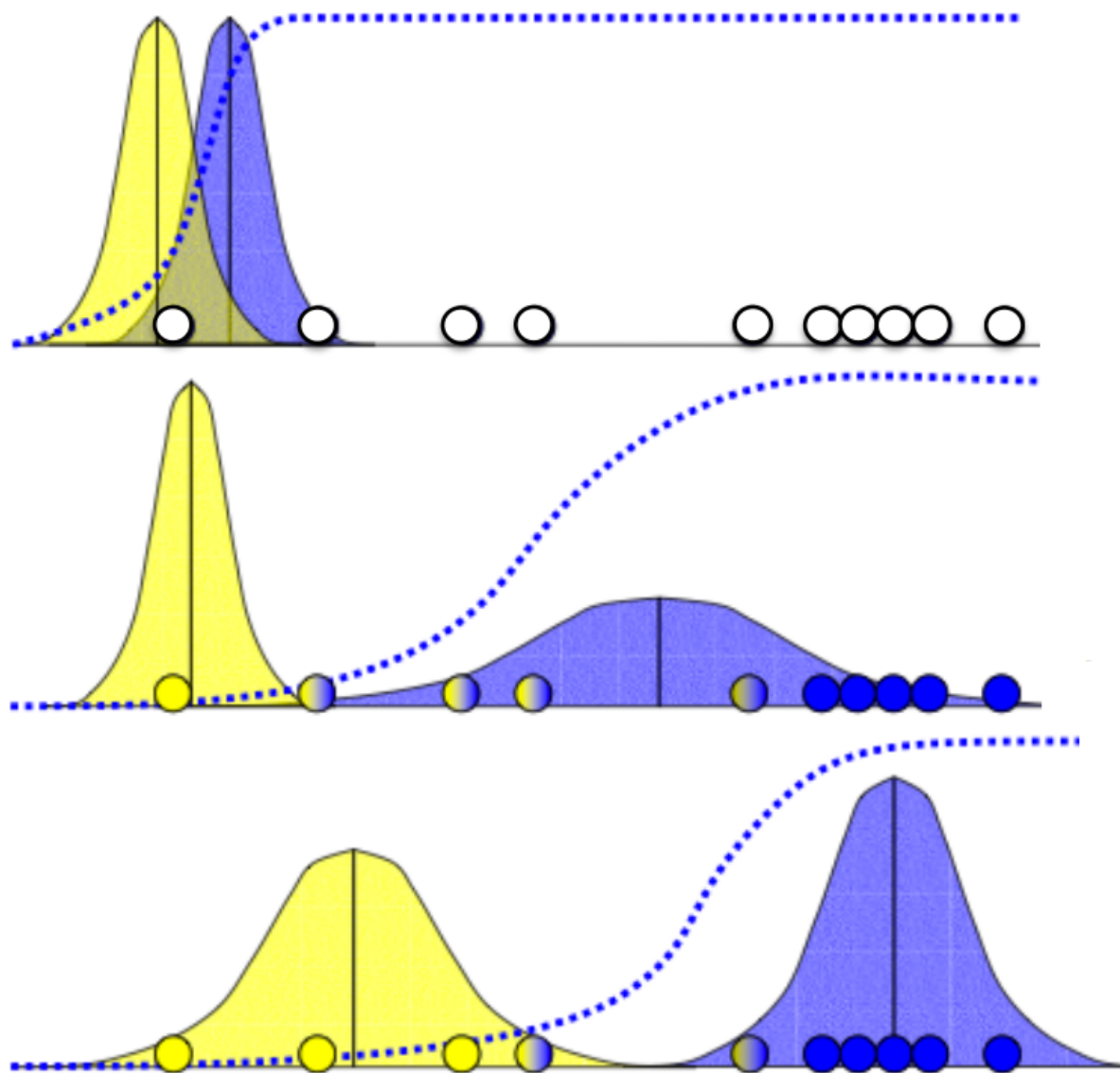


Рис. 1

Аналогично для  $(\mu_b, \sigma_b^2)$ .

И, наоборот, зная параметры  $(\mu, \sigma^2)$  мы определяем, какова вероятность принадлежности наблюдения этому распределению по формуле Байеса:

$$b_i = P(b | x_i) = \frac{P(x_i | b)P(b)}{P(x_i | b)P(b) + P(x_i | a)P(a)}, \quad a_i = P(a | x_i) = 1 - b_i$$

$$\text{где } P(x_i | b) = \frac{1}{\sqrt{2\pi\sigma_b^2}} e^{-\frac{(x_i - \mu_b)^2}{2\sigma_b^2}}.$$

Следовательно, в нашей задаче необходимо знать параметры  $(\mu, \sigma^2)$  для оценки принадлежности наблюдений распределениям и в то же время знать распределение для оценки параметров. Тут и приходит на помощь ЕМ-алгоритм (Рис. 1):

- (а) возьмем случайным образом два гауссовых распределения  $(\mu_a, \sigma_a^2)$  и  $(\mu_b, \sigma_b^2)$ ;
- (б) **Е-шаг:** вычислим  $a_i$  и  $b_i$  для каждого наблюдения;
- (с) **М-шаг:** пересчитываем  $(\mu_a, \sigma_a^2)$  и  $(\mu_b, \sigma_b^2)$  по формулам:

$$\mu_a = \frac{a_1 x_1 + \dots + a_n x_n}{a_1 + \dots + a_n}, \quad \sigma_a^2 = \frac{a_1(x_1 - \mu_a) + \dots + a_n(x_n - \mu_a)}{a_1 + \dots + a_n}$$

Аналогично для  $(\mu_b, \sigma_b^2)$ ;

- (д) продолжаем итеративно вплоть до сходимости;

Также в М-шаге можно было пересчитывать априорные вероятности  $P(b) = \frac{b_1 + \dots + b_n}{n}$  и  $P(a) = 1 - P(b)$ .

2.  $m > 1, k > 2$ .

$$P(c) = \frac{1}{n} \sum_{i=1}^n P(c | \vec{x}_i) \text{ для кластера } c$$

$$\mu_{c,j} = \sum_{i=1}^n \left( \frac{P(c | \vec{x}_i)}{nP(c)} \right) x_{i,j}$$

$$\begin{aligned}
(\Sigma_c)_{j,k} &= \sum_{i=1}^n \left( \frac{P(c | \vec{x}_i)}{nP(c)} \right) (x_{i,j} - \mu_{c,j}) (x_{i,k} - \mu_{c,k}) \\
P(c | \vec{x}_i) &= \frac{P(\vec{x}_i | c)P(c)}{\sum_{c'=1}^k P(\vec{x}_i | c')P(c')} \\
P(\vec{x}_i | c) &= \frac{1}{\sqrt{2\pi} |\Sigma_c|} \exp\left(-\frac{1}{2} \underbrace{(\vec{x}_i - \vec{\mu}_c)^T \Sigma_c^{-1} (\vec{x}_i - \vec{\mu}_c)}_{\sum_{j=1}^d \sum_{k=1}^d (x_{i,j} - \mu_{c,j}) (\Sigma_c^{-1})_{jk} (x_{i,k} - \mu_{c,k})}\right)
\end{aligned}$$

**Преимущества ЕМ-алгоритма.** По сравнению с широко известными градиентными методами настройки нейронных сетей, ЕМ-алгоритм более устойчив к шуму и быстрее сходится. Кроме того, он описывает каждый класс как совокупность компонент или кластеров, что позволяет лучше понимать внутреннюю структуру данных. В частности, центры сферических гауссовских компонент  $\mu_{yj}$  можно интерпретировать как виртуальные эталонные объекты, с которыми сравниваются классифицируемые объекты. Во многих прикладных задачах виртуальным эталонам удаётся подыскать содержательную интерпретацию. Например, в медицинской дифференциальной диагностике это может быть определённая ( $j$ -я) форма данного ( $y$ -го) заболевания. Информация о том, что классифицируемый объект близок именно к этому эталону, может быть полезной при принятии решений.

## 3.2 Применение ЕМ-алгоритма

В данной работе ЕМ-алгоритм представлен классом `GaussianMixture` [7], предложенной библиотекой `scikit-learn`. Этот класс позволяет оценивать параметры гауссовских распределений.

Метод `fit` этого класса оценивает параметры модели с помощью ЕМ-алгоритма. Метод `fit` итерируется между Е-шагом и М-шагом указанное количество раз до тех пор, пока изменение правдоподобия или нижней границы не станет меньше некоторого числа. В противном случае возникнет сообщение о том, что метод не сходится.

При создании экземпляра класса указывается количество компонент смеси.

### 3.2.1 Проверка работоспособности `GaussianMixture`

С помощью модуля `stats` [8] библиотеки `Scipy` сгенерированы выборки разных распределений.

Для ясности введём следующие обозначения:

- `cdf` — cumulative distribution function — функция распределения;
- `pdf` — probability density function — плотность вероятности распределения;
- `std` — standard deviation of the distribution — среднеквадратическое отклонение распределения.

Проверка работоспособности проводится в несколько этапов.

1. Напомним формулу гауссовского распределения

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

где параметр  $\mu$  — математическое ожидание и  $\sigma$  среднеквадратическое отклонение распределения. Квадрат среднеквадратического отклонения,  $\sigma^2$ , называется дисперсией.

Генерирование одного одномерного гауссовского распределения (Рис. 2):

```
mu = 0
variance = 1
sigma = math.sqrt(variance)
x = np.linspace(mu - 3*sigma, mu + 3*sigma, 100)
data = norm.pdf(x, mu, sigma)
```

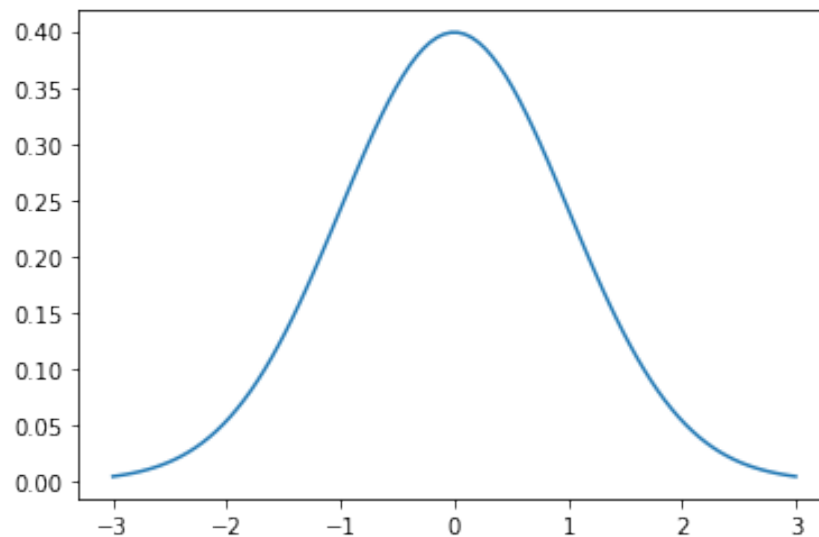


Рис. 2

Проверка оценки параметров соответствующей выборки с помощью GaussianMixtureModel:

```
k = 1
model = gmm(n_components=k, covariance_type='full')
model.fit(data)
```

Вывод программы:

```
data_mean: 0.1645975096425618
data_covariance: 0.13947206450268224
model_mean: 0.16459751
model_covariance: 0.13947565
```



В результате видим, что оценка параметров GaussianMixtureModel в точности совпадает с реальными параметрами.

2. Генерирование двух одномерных гауссовских распределений (Рис. 3):

```
mu_1 = 2
variance_1 = 4
sigma_1 = math.sqrt(variance_1)

mu_2 = 10
variance_2 = 1
sigma_2 = math.sqrt(variance_2)

x_1 = np.linspace(mu_1 - 3*sigma_1, mu_1 + 3*sigma_1, 100)
x_2 = np.linspace(mu_2 - 3*sigma_2, mu_2 + 3*sigma_2, 100)
data_1 = norm.pdf(x_1, mu_1, sigma_1)
data_2 = norm.pdf(x_2, mu_2, sigma_2)
```

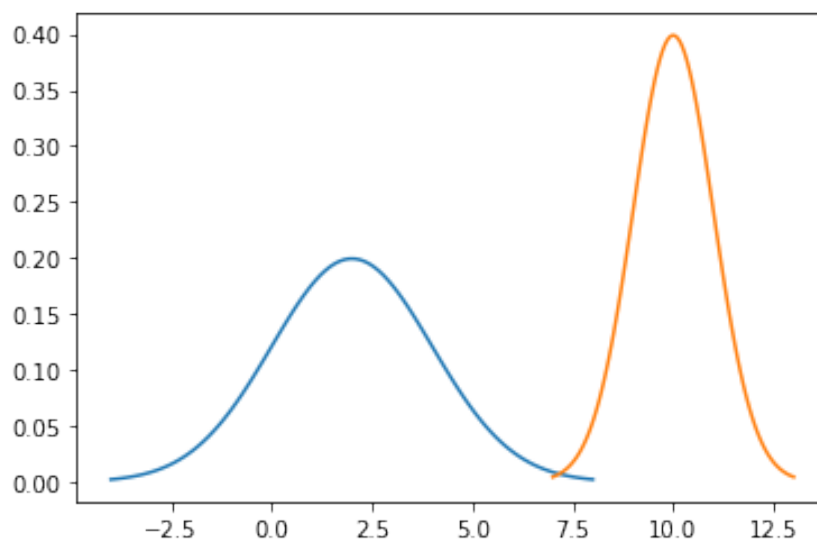


Рис. 3

Генерирование одной выборки для двух распределений и проверка оценки её параметров с помощью GaussianMixtureModel:

```

new_data = np.concatenate((data_1, data_2), axis=0)
np.random.shuffle(new_data)

k = 2
model = gmm(
    n_components=k,
    max_iter=1000,
    covariance_type='full'
)
model.fit(new_data)

```

Вывод программы:

```

data_mean_1: 0.0822987548212809
data_covariance_1: 0.06973603225134112
data_mean_2: 0.16459750964256176
data_covariance_2: 0.1394720645026822

model_mean_1: 0.01832714
model_covariance_1: 0.01409637
model_mean_2: 0.18062076
model_covariance_2: 0.10954001

```

Здесь во время многократных экспериментов точность результата варьировалась в зависимости от исходных распределений.

3. Генерирование одного двумерного гауссовского распределения (Рис. 4):

```

x = np.linspace(0, 10, 100)
y = np.linspace(10, 20, 100)
X, Y = np.meshgrid(x, y)

```

```

mu_x = np.mean(x)
sigma_x = np.std(x)
mu_y = np.mean(y)
sigma_y = np.std(y)
norm = multivariate_normal(
    [mu_x, mu_y],
    [[sigma_x, 0],
     [0, sigma_y]]
)

pos = np.empty(X.shape + (2,))
pos[:, :, 0] = X
pos[:, :, 1] = Y
data = norm.pdf(pos)

```

Проверка оценки параметров соответствующей выборки с помощью GaussianMixtureModel:

```

k = 2
model = gmm(
    n_components=k,
    max_iter=10000,
    covariance_type='full'
)
model.fit(new_data)

```

Вывод программы:

```

data_mean_x: 5.0
data_covariance_x: 2.9157646512850626
data_mean_y: 15.0

```

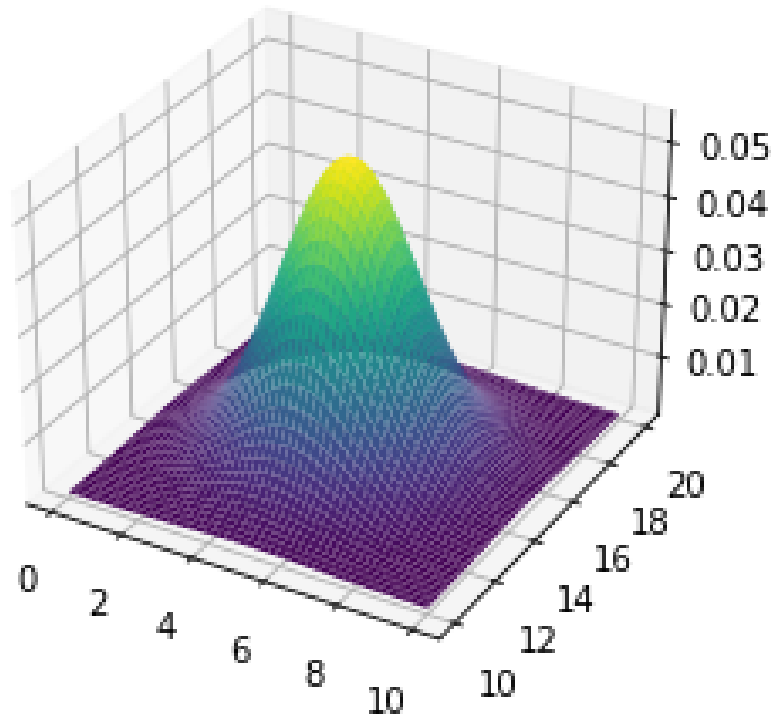


Рис. 4

data\_covariance\_y: 2.9157646512850626

model\_mean\_x: 5.12451189

model\_covariance\_x: 3.16980297

model\_mean\_y: 14.7726552

model\_covariance\_y: 3.22703713

Здесь во время многократных экспериментов точность результата варьировалась в зависимости от исходных распределений.

4. Напомним формулу равномерного распределения

$$p(x) = \frac{1}{b - a},$$

принимая значения  $p(x)$  в промежутке  $[a, b)$ , и ноль — вне промежутка.

Генерирование одного одномерного равномерного распределения (Рис. 5):

```
a = 0
b = 5
size = 100
uni = uniform(loc=a, scale=b)
x = np.linspace(uni.ppf(0), uni.ppf(1), size)
data = uni.pdf(x)
```

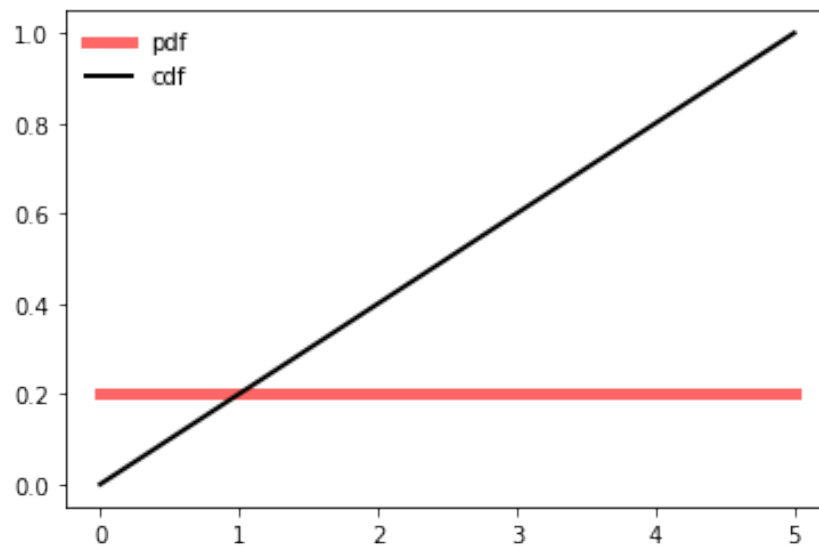


Рис. 5

Проверка оценки параметров соответствующей выборки с помощью GaussianMixtureModel:

```
k = 1
model = gmm(n_components=k, covariance_type='full')
model.fit(data)
```

Вывод программы:

```
data_mean: 0.19999999999999996
data_covariance: 5.551115123125783e-17
model_mean: 0.2
model_covariance: 0.001
```

В результате видим, что оценка параметров равномерной выборки с помощью GaussianMixtureModel далека от истинного значения.

### 3.2.2 Применение ЕМ-алгоритма к цензурированной выборке

Дана выборка объема  $n = 97$  вида  $\{(z_{(i)}, \delta_{(i)}), i = \overline{1, n}\}$  (Рис. 6):

(777;1), (781;0), (843;0), (866;0), (869;1), (872;1), (876;1), (893;1), (894;1), (895;0), (898;1), (906;0), (907;1), (909;1), (911;1), (911;0), (914;0), (927;1), (932;1), (936;0), (940;0), (942,5;0), (943;0), (945;1), (945;0), (948;1), (951;0), (953;0), (956;0), (957;1), (957;0), (959;0), (960;0), (966;1), (966;0), (969;1), (970;0), (971;1), (972;0), (973;0), (977;0), (983;1), (984;0), (985;1), (989;1), (992,5;1), (993;1), (996;1), (998;1), (1001;0), (1002;0), (1005;0), (1006;0), (1009;1), (1011,5;1), (1012;1), (1012;0), (1013;0), (1015;0), (1016;0), (1018;0), (1022;1), (1023;0), (1025;1), (1027;0), (1029;1), (1031;1), (1031;0), (1031,5;0), (1033;1), (1036;1), (1043;1), (1043;0), (1044;1), (1044;0), (1045;0), (1047;0), (1053;1), (1055;1), (1058;0), (1059;1), (1060;1), (1060;0), (1064;0), (1070;0), (1073;0), (1080;1), (1085;1), (1093;0), (1093,5;1), (1094;1), (1106;0), (1107;0), (1118;0), (1128;1), (1139;1), (1153;0).

Данные представлены в месяцах, причем число 1 в парах означает нецензурирование (т.е. смерть), а 0 - цензурирование. При этом 46 человек умерли с начала открытия центра в 1964 году по 1 июля 1975 года ко дню сбора данных. Это нецензурированные данные. Из остальных 51 человек 5 были выписаны из центра, а 46 ещё были живы к 1 июля 1975 года. Это цензурированные данные.

Построим оценку, предложенную Абдушукуровым А. А., по следу-

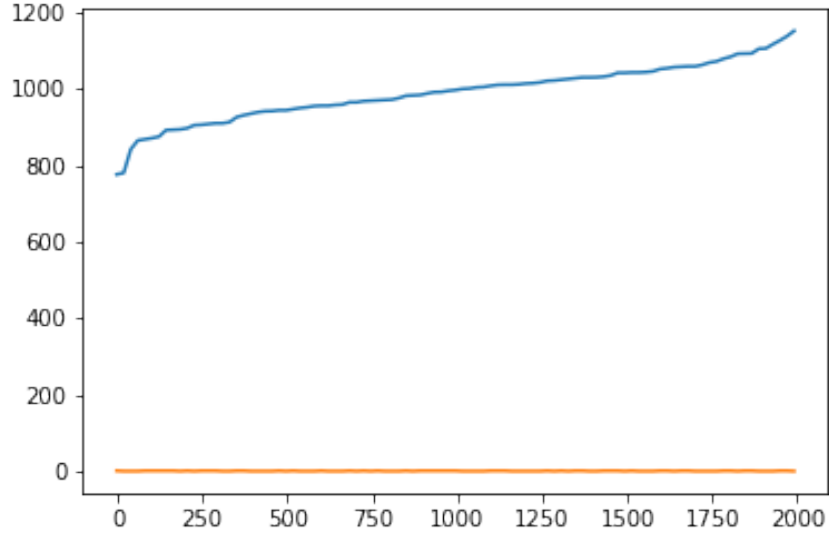


Рис. 6

ющим формулам:

$$F_n^{RR}(x) = 1 - (1 - H_n(x))^{\frac{\Lambda_{1n}(x)}{\Lambda_n(x)}} =$$

$$= \begin{cases} 0, & x < z_{(1)} \\ 1 - (1 - \frac{k}{n})^{\frac{\Lambda_{1n}(x)}{\Lambda_n(x)}}, & z_{(k)} \leq x < z_{(k+1)}, k = \overline{1, n} \\ 1, & x \geq z_{(n)} \end{cases},$$

где

$$1 - H_n(x) = \frac{1}{n} \sum_{j=1}^n I(z_{(j)} > x),$$

$$\Lambda_{1n}(x) = \int_{-\infty}^x \frac{dH_{1n}(u)}{1 - H_{1n}(u-)} =$$

$$= \frac{1}{n} \sum_{j=1}^n \frac{\delta_{(j)} I(z_{(j)} \leq x)}{\frac{1}{n} \sum_{i=1}^n I(z_{(i)} \geq z_{(j)})} = \sum_{j=1}^n \frac{\delta_{(j)} I(z_{(j)} \leq x)}{\sum_{i=1}^n I(z_{(i)} \geq z_{(j)})},$$

$$\Lambda_n(x) = \int_{-\infty}^x \frac{dH_n(u)}{1 - H_n(u-)} = \sum_{j=1}^n \frac{I(z_{(j)} \leq x)}{\sum_{i=1}^n I(z_{(i)} \geq z_{(j)})}.$$

Приведём код программы, которая реализует вышеизложенные формулы и строит оценку  $F_n^{RR}$  (Рис. 7):

```

def one_minus_H_n(x, data):
    n = len(data)
    result = 0
    for i in range(n):
        result += 1 if data[i][0] > x else 0
    result /= n
    return result

def Lambda_1n(x, data):
    n = len(data)
    result = 0
    numerator = 0
    denominator = 0
    for i in range(n):
        denominator = 0
        numerator = data[i][1] * (1 if data[i][0] <= x else 0)
        for j in range(n):
            denominator += (1 if data[j][0] >= data[i][0] else 0)
        result += numerator / denominator
    return result

def Lambda_n(x, data):
    n = len(data)
    result = 0
    numerator = 0
    denominator = 0
    for i in range(n):
        denominator = 0
        numerator = 1 if data[i][0] <= x else 0
        for j in range(n):
            denominator += (1 if data[j][0] >= data[i][0] else 0)
        result += numerator / denominator
    return result

```



```
def FnRR(x, data):
    result = 1 - one_minus_H_n(x, data)**(Lambda_1n(x, data) /
                                           Lambda_n(x, data))

    return result
```

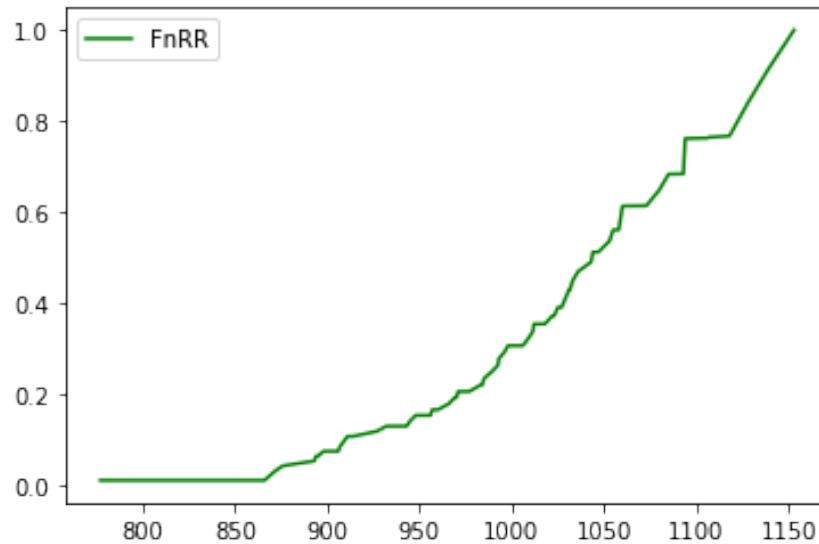


Рис. 7

По следующим формулам вычислим математическое ожидание и дисперсию для цензурированной выборки:

$$\hat{\mu}_n = \int_{z(1)}^{z(n)} t dF_n^{RR}(t) = \sum_{i=1}^n z_{(i)} \Delta F_n^{RR}(z_i)$$

$$\hat{\sigma}_n^2 = \int_{z(1)}^{z(n)} (t - \hat{\mu}_n)^2 dF_n^{RR}(t) = \sum_{i=1}^n (z_{(i)} - \hat{\mu}_n)^2 \Delta F_n^{RR}(z_i),$$

где  $\Delta F_n^{RR}(z_i)$  — скачок функции.

Приведём код программы, которая реализует вышеизложенные формулы в одной функции:

```
def calculate_mu_and_sigma(z, data):
    n = len(data)
    mu = z[0] * FnRR(z[0], data)
```

```

for i in range(2, n):
    mu += z[i] * (FnRR(z[i], data) - FnRR(z[i-1], data))
variance = (z[0] - mu) ** 2 * FnRR(z[0], data)
for i in range(2, n):
    variance += (z[i] - mu) ** 2 * (FnRR(z[i], data)
                                   - FnRR(z[i-1], data))

sigma = sqrt(variance)
return mu, sigma

```

Далее центрируем и нормируем исходные данные (Рис. 8):

```

z = [data[i][0] for i in range(97)]
mu, sigma = calculate_mu_and_sigma(z, data)

data_processed = (data[:, 0] - mu) / sigma
data_processed = np.c_[data_processed, data[:, 1]]

z = [data_processed[i][0] for i in range(97)]
mu, sigma = calculate_mu_and_sigma(z, data_processed)

```

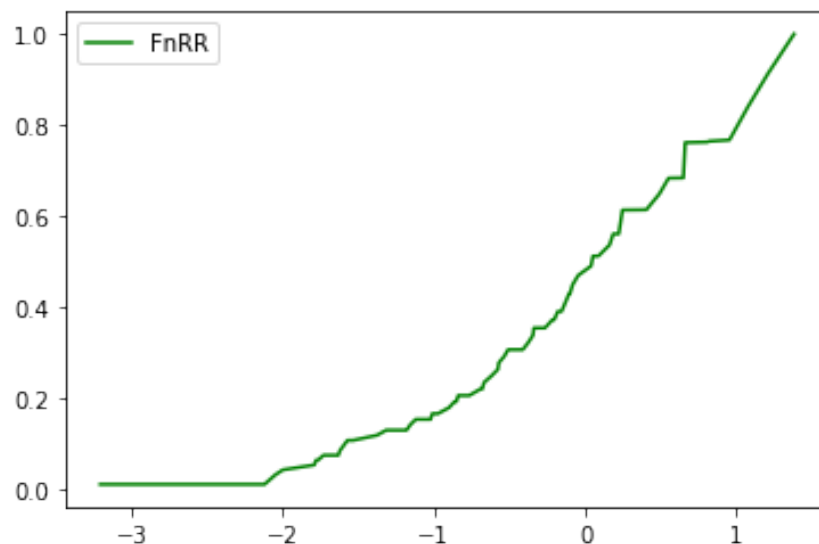


Рис. 8

Вывод программы:

```
mu_before: 1039.7833741995091
sigma_before: 81.9647172814403
mu_after: 3.5417227168543786e-06
sigma_after: 0.9999999999937279
```

Проверка оценки параметров выборки, состоящей из компоненты  $z$ , с помощью GaussianMixtureModel:

```
k = 1
model = GMM(
    n_components=k,
    max_iter=10000,
    covariance_type='full'
)
model.fit(data_processed)
```

Вывод программы:

```
data_mean: 3.5417227168543786e-06
data_covariance: 0.9999999999937279
model_mean: -0.58832044
model_covariance: 0.8946689
```

В результате видим, что оценка параметров цензурированной выборки после центрирования и нормирования с помощью GaussianMixtureModel очень близка к истинному значению.

## 4 Заключение

В данной работе было собрано методическое пособие по знакомству с ЕМ-алгоритмом на примере смеси гауссовых распределений. Было дано описание самого алгоритма, которое позволит быстро ознакомиться с его теоретической частью.

Также был выполнен следующий прикладной вклад. Во-первых, была наглядно продемонстрирована генерация выборок разных распределений и исследованы результаты способностей GaussianMixture оценивать параметры таких выборок. Во-вторых, был сделан новый шаг по вычислению математического ожидания и дисперсии цензурированных выборок на основе оценки функции распределения таких выборок  $F_n^{RR}$ , предложенной Абдушукуровым А. А., и был применён ЕМ-алгоритм к реальным цензурированным данным.

В дальнейшем предполагается увеличение компонент смеси распределений и расширение семейства самих распределений, подаваемых на вход ЕМ-алгоритму. К примеру, разработка GeneralMixture, позволяющей оценивать параметры любого семейства распределений.

Автор выражает благодарность Абдушукурову А. А. за научное руководство.

## 5 Список литературы

- [1] Dempster A. P., Laird N. M., Rubin D. B. Maximum likelihood from incomplete data via the EM algorithm // J. of the Royal Statistical Society, Series B. — 1977.
- [2] Шлезингер М. И. О самопроизвольном различении образов // Читающие автоматы. — Киев, Наукова думка, 1965.
- [3] Айвазян С. А., Бухштабер В. М., Енюков И. С., Мешалкин Л. Д. Прикладная статистика: классификация и снижение размерности.
- [4] В. Ю. Королёв. EM-алгоритм, его модификации и их применение к задаче разделения смесей вероятностных распределений
- [5] К. В. Воронцов. Лекции по статистическим (байесовским) алгоритмам классификации
- [6] Wu C. F. G. On the convergence properties of the EM algorithm // The Annals of Statistics. — 1983.
- [7] Документация GaussianMixture  
<https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html#sklearn.mixture.GaussianMixture>
- [8] Документация к модулю stats библиотеки Scipy  
<https://docs.scipy.org/doc/scipy/reference/tutorial/stats.html>
- [9] GitHub-репозиторий выпускной квалификационной работы  
<https://github.com/ruslanmuradasilov/diploma>