МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
Кафедра компьютерных технологий и систем

ИВАНКОВИЧ Арсений Валерьевич

# ОБУЧЕНИЕ В НАПРАВЛЕННЫХ ВЕРОЯТНОСТНЫХ ГРАФОВЫХ МОДЕЛЯХ

Дипломная работа

Научный руководитель:
Доктор педагогических наук,
кандидат физико-математических
наук, профессор
В.В.Казаченок

Допущена к защите
«___» _____2025 г.
Заведующий кафедрой компьютерных технологий и систем
доктор педагогических наук, кандидат физико-математических наук,
профессор В.В.Казаченок

Минск, 2025

# РЕФЕРАТ

Дипломная работа, 60 страниц, 20 рисунков, 2 приложения, 13 источников.

Ключевые слова: НАПРАВЛЕННЫЕ ВЕРОЯТНОСТНЫЕ ГРАФОВЫЕ МОДЕЛИ, БАЙЕСОВСКАЯ СЕТЬ, СТРУКТУРНОЕ ОБУЧЕНИЕ.

Объект исследования – методы обучения направленных вероятностных графовых моделей.

Предмет исследования – алгоритмы обучения направленных вероятностных графовых моделей, их применение на примере байесовских сетей.

Цель работы – анализ различных подходов к обучению направленных вероятностных графовых моделей, исследование их преимуществ и недостатков.

В процессе работы были изучены и проанализированы подходы и конкретные алгоритмы обучения направленных вероятностных графовых моделей, которые могут быть использованы для решения задачи поиска структуры графа и параметров модели. Эти подходы были сравнены на практических задачах построения байесовских сетей в сфере медицинской диагностики и метеорологии. Было проанализировано влияние на качество работы алгоритмов размерности задачи, количества тренировочных примеров и наличия шума в данных.

# РЭФЕРАТ

Дыпломная праца, 60 старонак, 20 малюнкаў, 2 дадаткі, 13 крыніц.

Ключавыя словы: НАКІРАВАНЫЯ ІМАВЕРНАСНЫЯ ГРАФАВЫЯ МАДЭЛІ, БАЙЕСАЎСКАЯ СЕТКА, СТРУКТУРНАЕ НАВУЧЭННЕ.

Аб’ект даследавання – метады навучання накіраваных імавернасных графавых мадэляў.

Прадмет даследавання – алгарытмы навучання накіраваных імавернасных графавых мадэляў, іх прымяненне на прыкладзе байесаўскіх сетак.

Мэта працы – аналіз розных падыходаў да навучання накіраваных імавернасных графавых мадэляў, даследаванне іх пераваг і недахопаў.

У працэсе працы былі вывучаны і прааналізаваны падыходы і канкрэтныя алгарытмы навучання накіраваных імавернасных графавых мадэляў, якія могуць быць выкарыстаны для вырашэння задачы пошуку структуры графа і параметраў мадэлі. Гэтыя падыходы былі параўнаны на практычных задачах пабудовы байесаўскіх сетак у сферы медыцынскай дыягностыкі і метэаралогіі. Было праналізавана ўздзеянне на якасць працы алгарытмаў размернасці задачы, колькасці трэніровачных прыкладаў і наяўнасці шуму ў дадзеных.

# ABSTRACT

Graduation thesis, 60 pages, 20 figures, 2 attachments, 13 sources.

Keywords: DIRECTED PROBABILISTIC GRAPHICAL MODELS, BAYESIAN NETWORK, STRUCTURAL LEARNING.

The object of research is methods for learning directed probabilistic graphical models.

The subject of research is an analysis of learning algorithms for directed probabilistic graphical models and their application using Bayesian networks as an example.

The aim of the work is a review of different approaches to learning directed probabilistic graphical models, a research of their advantages and disadvantages.

In the process of work various approaches and specific algorithms for learning directed probabilistic graphical models were studied and analyzed, which can be used to solve the problem of finding the graph structure and model parameters. These approaches were compared in practical tasks of constructing Bayesian networks in the fields of medical diagnostics and meteorology. The influence of problem dimensionality, the number of training examples, and the presence of noise in the data on the performance of the algorithms was analyzed.

# ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ....7
ГЛАВА 1 ОСНОВНЫЕ ПОНЯТИЯ ВЕРОЯТНОСТНЫХ ГРАФОВЫХ МОДЕЛЕЙ....9
1.1 Вероятностные графовые модели....9
1.2 Представление, обучение, логический вывод....11
1.3 Теорема Байеса в машинном обучении....12
1.4 Граф в направленных вероятностных графовых моделях....12
1.5 Модель регрессии....13
1.6 EM-алгоритм....15
ГЛАВА 2 ОБУЧЕНИЕ ПАРАМЕТРОВ НАПРАВЛЕННЫХ ВЕРОЯТНОСТНЫХ ГРАФОВЫХ МОДЕЛЕЙ....17
2.1 Общий подход к обучению параметров в направленных вероятностных графовых моделях....17
2.2 Байесовская сеть....18
2.2.1 Обучение параметров байесовской сети с частично не наблюдаемыми данными....19
2.2.2 Наивный байесовский классификатор....20
2.2.3 Марковская цепь....22
2.3 Скрытая марковская модель....23
2.4 Выводы....25
ГЛАВА 3 ОБУЧЕНИЕ СТРУКТУРЫ НАПРАВЛЕННЫХ ВЕРОЯТНОСТНЫХ ГРАФОВЫХ МОДЕЛЕЙ....26
3.1 Поиск структуры графа....26
3.2 Алгоритмы, основанные на ограничениях....27
3.3 Алгоритмы, основанные на оценке....28
3.3.1 Метрика оценки MDL....28
3.3.2 Метрики оценки K2 и BIC....29
3.3.3 Поиск структуры графа с определенной оценкой....31
3.4 Алгоритмы, основанные на регрессии....33
3.5 Гибридные алгоритмы....33
3.6 Выводы....34
ГЛАВА 4 ПРИМЕНЕНИЕ НАПРАВЛЕННЫХ ВЕРОЯТНОСТНЫХ ГРАФОВЫХ МОДЕЛЕЙ В ПРАКТИЧЕСКИХ ЗАДАЧАХ....35
4.1 Использованные технологии....35

4.2 Рассматриваемые задачи....36
4.3 Анализ методов поиска структуры графа....39
4.3.1 Сеть Asia....40
4.3.2 Сеть Child....41
4.3.3 Сети Hailfinder и Pathfinder....43
4.4 Анализ на разном количестве тренировочных примеров....44
4.5 Анализ результатов при добавлении шума....47
4.6 Выводы....49
ЗАКЛЮЧЕНИЕ....51
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ....53
ПРИЛОЖЕНИЕ А Код программы демонстрации разных подходов к обучению НВГМ на примере байесковских сетей....55
ПРИЛОЖЕНИЕ Б Код программы демонстрации разных подходов на разных объемах данных с возможностью добавления шума....57

# ВВЕДЕНИЕ

Направленные вероятностные графовые модели (НВГМ) представляют собой мощный инструмент для моделирования сложных зависимостей между переменными во множестве задач в различных областях, включая компьютерное зрение, обработку естественного языка, биоинформатику и финансы. Эти модели позволяют эффективно представлять вероятностные зависимости и использовать априорные знания для анализа многомерных данных и учета неопределенности. Их способность использовать вероятностные зависимости между наблюдаемыми и скрытыми переменными позволяет моделям эффективно обрабатывать данные с неполной информацией и высокой степенью неопределенности. Однако создание и обучение НВГМ представляет собой сложную задачу из-за необходимости правильного построения структурных зависимостей между переменными, выбора подходящих методов оптимизации и учета вычислительных ограничений.

Данная работа посвящена анализу алгоритмов обучения направленных вероятностных графовых моделей. Кроме обучения, акцент будет сделан на получении результатов из НВГМ, а также на их применении в реальных задачах. Само обучение можно условно разделить на два этапа: поиск параметров модели для заданной структуры графа, лежащего в ее основе, а также непосредственно поиск структуры.

Для достижения поставленных целей работа разбита на 4 части (главы). В первой проводится краткий обзор вероятностных графовых моделей и их применения в различных областях. Подробно объясняется принцип работы направленных вероятностных моделей, лежащие в его основе технологии. Эта глава содержит описание понятий, используемых в дальнейшем.

Во второй главе рассматривается обучение параметров направленных вероятностных графовых моделей. Приводится общий принцип обучения на основе метода максимального правдоподобия и EM-алгоритма, затем анализируется его применение на конкретных моделях этого типа.

Третья глава посвящена структурному обучению направленных вероятностных графовых моделей на примере байесовских сетей. Анализируются три основных подхода, приводятся теоретические обоснования их эффективности, преимуществ и недостатков.

В четвертой главе показаны результаты экспериментов с алгоритмами структурного обучения на практической задаче обучения байесовских сетей различного размера, анализируются ранее сделанные теоретические выводы.

В заключении подводятся итоги проделанной работы, делаются выводы о применимости направленных вероятностных графовых моделей и подходах к их обучению. Приложения содержат исходный код программ, позволяющих повторить проведенные в четвертой главе эксперименты.

# ГЛАВА 1
# ОСНОВНЫЕ ПОНЯТИЯ ВЕРОЯТНОСТНЫХ ГРАФОВЫХ МОДЕЛЕЙ

## 1.1 Вероятностные графовые модели

Вероятностная графовая модель (ВГМ) – это математическая структура, которая объединяет теорию вероятностей и графы для создания статистических моделей. ВГМ предоставляет условные зависимости между компонентами многомерных данных в виде графа, где вершины представляют собой случайные величины, а ребра – непосредственно вероятностные взаимосвязи между этими случайными величинами [2].

Вероятностная графовая модель – это компактное представление совместного распределения вероятностей, из которого можно получить частное распределение вероятностей и условные вероятности. Основная идея вероятностных графовых моделей заключается в том, чтобы вместо составления зависимостей между всеми переменными и их комбинациями, рассматривать только те отношения, которые зависимы между собой в рамках конкретной задачи, и включении только этих отношений в вероятностную модель для уменьшения требуемых объемов памяти и времени вычислений.

Вероятностная графовая модель определяется двумя компонентами:

1) графом, который определяет структуру модели;

2) набором локальных функций $f(Y_i)$, которые определяют параметры.

Совместное распределение вероятностей вычисляется как произведение локальных функций:

$$P(X_{1,}X_{2,\dots}X_N)=K\prod_{i=1}^{M}f(Y_i). \qquad (1.1)$$

где $K$ – константа нормализации (приводит сумму вероятностей к единице).

Такое представление в форме графа и набор локальных функций являются основой для логического вывода и обучения в ВГМ. Под логическим выводом понимается получение частного распределения вероятностей или условных вероятностей на любом подмножестве переменных Z с учетом любого другого подмножества Y, то есть по сути получение вероятности какого-либо события Z из модели при условии того, что произошло Y. Под

обучением же понимается оценка структуры графа и параметров (локальных функций) модели при заданном наборе тренировочных данных.

Вероятностные графовые модели можно классифицировать по следующим характеристикам:

1) направленные или ненаправленные (ориентированные или не ориентированные);

2) статические или динамические;

Первая характеристика определяет тип графа, используемого в модели. Ненаправленные ВГМ используют неориентированные графы и представляют симметричные отношения, тогда как направленные используют ориентированные ациклические графы и представляют отношения, в которых направление важно. В случае ненаправленных моделей формула (1.1) принимает такой вид:

$$P(X_{1,}X_{2,\dots}X_N)=\frac{1}{Z}\prod_{c\in C}\phi_c(X_c). \tag{1.2}$$

где $C$ – клики графа, то есть подмножества вершин графа, образующие полный граф,

$Z$ – нормировочная константа.

Функции в произведении называются потенциальными и определяют зависимости.

В случае же направленных моделей (1.1) имеет следующий вид:

$$P(X_{1,}X_{2,\dots}X_N)=\prod_{i}^{n}P(X_i|\pi(X_i)). \tag{1.3}$$

где $\pi(X_i)$ – родители вершины $X_i$.

Нормировочная константа в данном случае не нужна, так как формула работает с вероятностями.

Таким образом, ненаправленные модели представляют корреляцию между переменными, а направленные – причинно-следственные связи, и очевидны сферы применения каждой из них: ненаправленные модели для симметричных данных вроде картинок, направленные для задач диагностики и анализа.

Вторая характеристика определяет, рассматривается ли конкретный момент времени (статическая модель) или различные его интервалы (динамическая модель). Это означает, что в первом случае все тренировочные данные независимы и одинаково распределены, что актуально для таких задач

как медицинская диагностика и класификация спама. В случае динамических моделей данные зависят от предыдущих наблюдений, что делает такие модели применимыми в задачах, связанных, например, с временными рядами или видео.

## 1.2 Представление, обучение, логический вывод

Для каждого класса вероятностной графовой модели существуют три аспекта: представление, логический вывод и обучение. Представление (representation) – это основное свойство каждой модели, которое определяет, какие сущности (объекты) образуют модель и как эти сущности связаны между собой. Например, все вероятностные графовые модели могут быть представлены как графы, которые определяют структуру модели, и локальные функции, описывающие параметры модели. Но вид графа и локальные функции различны для разных типов моделей.

Как уже упоминалось, логический вывод заключается в ответах на разнообразные вероятностные запросы и основан на самой модели. К логическому выводу можно отнести, например, получение апостериорного распределения вероятностей переменной или набора переменных, при условии, что другие переменные в этой модели известны. Главная трудность заключается в том, чтобы сделать это как можно более вычислительно эффективно.

Для создания ВГМ существует два основных способа: создание вручную с помощью экспертов в рассматриваемой предметной области или вывод модели алгоритмически из данных. Очевидно, что на задачах большой размерности построение вручную сложно и дорогостояще, не говоря уже о субъективности оценок, выставляемых экспертами. Поэтому немало исследовании проводилось для нахождения методов вывода модели алгоритмически, зачастую с использованием уже известных методик машинного обучения. Важным свойством этих методик с практической прикладной точки зрения является возможность отделения процессов логического вывода и обучения от конкретных моделей и данных. В результате алгоритмические методы, разработанные для логического вывода в каждом классе вероятностных графовых моделей, можно напрямую применять на разнообразных задачах.

Данная работа фокусируется на направленных ВГМ. Проанализированные далее представления, логические выводы и методы

обучения применимы только на моделях этого типа, хотя некоторые идеи используются и в ненаправленных ВГМ.

## 1.3 Теорема Байеса в машинном обучении

Многие рассуждения как в этой работе, так и во всем машинном обучении опираются теорему Байеса. Действительно, практически все задачи машинного обучения заключаются в том, чтобы по данным D подобрать описывающие их параметры θ. Математическая модель обычно определяется как задание распределения вероятностей на параметрах и данных p(θ, D) [1]. Но смоделировать такое распределение тяжело, поэтому прибегают сначала к $P(\theta, D)=P(D|\theta)P(\theta)$ , а затем к теореме Байеса:

$$P(\theta|D)=\frac{P(D|\theta)P(\theta)}{P(D)}. \quad (1.4)$$

где $P(\theta|D)$ – апостериорная вероятность,
$P(D|\theta)$ – правдоподобие,
$P(\theta)$ – априорная вероятность,
$P(D)$ – вероятность по данным.

Если максимизировать правдоподобие, получится метод максимального правдоподобия, в машинном же обучении идут на шаг дальше и максимизируют апостериорное распределение. Заметим, что вероятность по данным не зависит от параметров, поэтому задача максимизации (максимальная апостериорная гипотеза) выглядит следующим образом:

$$arg\,max_{\theta} P(\theta|D)=arg\,max_{\theta} P(D|\theta)P(\theta). \quad (1.5)$$

## 1.4 Граф в направленных вероятностных графовых моделях

Один из ключевых компонентов, определяющих структуру НВГМ – это граф G(V, E), определяющий структуру модели.

Здесь V = $\{u_i\}$ – вершины графа, каждая из которых представляет случайную величину, причем каждая такая величина может принимать

значения из множества $Val(u_i)$, а $E = \{(u_i, u_j) : i \neq j\}$ – дуги. Далее, если существует дуга из $u_i$ в $u_j$, будем говорить, что $u_i$ – родитель для $u_j$, $\pi(u_i)$ – множество родителей вершины. Ребра E, в свою очередь, представляют причинно-следственные связи между вершинами, например в виде чисел, которые затем переводятся в вероятности.

Для такого графа вводится понятие марковского покрытия вершины –это все родители и дети вершины, а также все остальные родители детей. Вершины из покрытия предоставляют всю информацию, необходимую для предсказания значения вершины.

Еще одно понятие, которое будет использоваться в дальнейшем повествовании – d-разделенные множеством вершин вершины. В неориентированном графе это значит, что любая цепь, соединяющая две вершины, содержит вершины из множества. Для ориентированного графа сначала вводится понятие d-разделенной цепи, и уже на его основании d-разделения вершин. Цепь (не учитывает направление дуг) между u и v называется d-разделенной множеством вершин Z, если выполняется любое из 3 условии:

- Существует цепь P из u в v или наоборот, которая содержит u … → m → … v или u … ← m ← … v, где m – вершина из Z.
- P содержит «вилку» u … ← m → … v, в которой m – вершина из Z.
- P содержит «коллайдер» u … → m ← … v, в котором m – вершина не из Z.

В свою очередь две вершины называются d-разделенными, если все цепи между ними d-разделены. В вероятностных направленных графовых моделях d-разделение двух вершин множеством Z означает, что если значения вершин из Z известны, то две вершины независимы. Это, в свою очередь, означает, что для каждой вершины вводится условное распределение $P(u_i|\pi(u_i))$, где $\pi(u_i)$ – родители $u_i$, и вероятностная направленная графовая модель позволяет однозначно получить совместное распределение таким образом:

$$P(u_1,\dots,u_n)=\prod_{i=1}^{n} P(u_i|\pi(u_i)). \qquad (1.6)$$

## 1.5 Модель регрессии

Модель регрессии решает такую задачу: имеется N тренировочных примеров $(x_i, y_i)$ и заранее определенная функция Ф(w, x), необходимо найти

такой набор w, называемых весами, чтобы функция Ф выдавала наилучшие предсказания $y_i$ при данном $x_i$. Для решения задачи используют так называемую функцию потерь, которая зависит от весов и принимает тем большие значения, чем больше ошибка в предсказании. Минимизировав эту функцию относительно весов получают решение. Некоторые функции можно минимизировать напрямую, но, как правило, используют итеративный процесс постепенной коррекции весов, а именно градиентный спуск. Градиентный спуск основывается на том факте, что градиент функции направлен в сторону ее наискорейшего роста, а антиградиент – наоборот, убывания. Тогда, постепенно корректируя веса прибавлением к ним части антиградиента, можно прийти близко к минимиму, но не всегда глобальному [3].

В данной работе будет рассматриваться линейная регрессия, когда целевую переменную представляют в виде взвешенной суммы признаков плюс еще одной обучаемой переменной, называемой смещением, цель которой заключается в том, чтобы позволить модели не проходить через нулевую координату.

$$y = \sum_{i=1}^{N} w_i x_i + b. \tag{1.7}$$

где $y$ – целевая переменная,
$x_i$ – признаки примера,
$w_i$ – соответствующие признакам веса,
$b$ – смещение.

На практике, если достигается глобальный минимум функции потерь на тренировочных данных, то это означает переобучение. Переобучение – это ситуация, когда модель слишком подстраивается под тренировочные данные, которые как правило могут содержать неточности, и плохо обобщается на другие, а ведь конечная цель обучения модели регрессии – это ее использование на не виденных ранее данных для предсказания.

Один из признаков переобучения – большие абсолютные значения весов, откуда следует способ предотвращения переобучения. Чтобы ограничить рост весов, в функцию потерь вносят дополнительное слагаемое – регуляризатор. Регуляризатор чаще всего имеет форму $\lambda\|w\|_p$, где $\lambda$ – число меньше единицы, определяющее влияние регуляризатора на функцию потерь, а $\|w\|_p$ – $L^p$ норма:

$$\|w\|_p = \left(\sum_i x_i^p\right)^{\frac{1}{p}}. \tag{1.8}$$

Чем меньше значение p, тем больше регуляризатор поощряет зануление некоторых весов. На практике для простоты вычислении работают с натуральными значениями p, чаще всего 1 и 2.

В контексте НВГМ наибольший интерес будет представлять именно L1 регуляризация (регуляризация Лассо), так как она дает наибольшее зануление весов. Это явление подтверждено исследованиями, и объясняется формой изолиний (линия, в каждой точке которой функция имеет одинаковое значение). Чем меньше число p, тем менее округлую форму принимают изолинии и, соответственно, тем больше шанс пересечения изолинии регуляризатора и функции потерь на оси координат. Это явление графически представлено на рисунке 1.1.

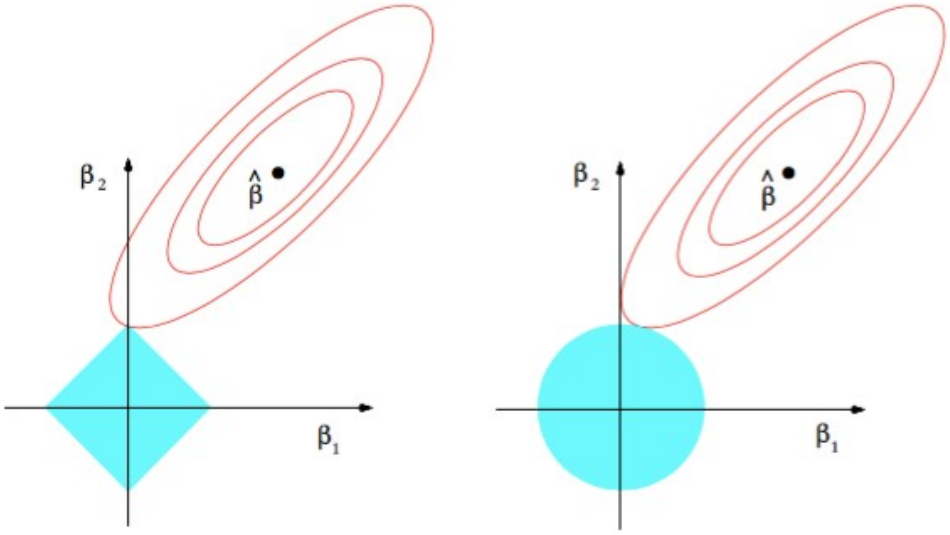



**Рисунок 1.1 – Сравнение изолинии L1 (слева) и L2 регуляризации в двумерном случае**

## 1.6 EM-алгоритм

Этот алгоритм используется в нескольких методах обучения, описанных в следующих разделах. Его идея заключается в том, чтобы добавить в метод максимального правдоподобия некие латентные переменные, знание которых значительно упростит вычисления. Зачастую в качестве латентных переменных выбирается непосредственно искомый ответ, например, номер кластера в задаче кластеризации. Алгоритм подразумевает случайную инициализацию на

первом этапе, затем итеративное повторение двух шагов, E-шага (от expectation, ожидание) и M-шага (от maximization, максимизация).

На E-шаге для фиксированных параметров модели, полученных из M-шага, подсчитывается ожидание латентных переменных, на M-шаге для подсчитанных в E-шаге латентных переменных пересчитываются параметры модели.

Более формально, изначальная задача подразумевает поиск $\max_\theta L(\theta|X)$, где L – функция правдоподобия, X – набор тренировочных примеров, θ – искомые параметры. Ее можно представить таким образом:

$$L(\theta|X)=\int p(X|Z,\theta)\,p(Z|\theta)\,dZ. \quad (1.9)$$

где $Z$ – это латентные переменные.

Тогда E-шаг – это вычисление условного ожидания логарифма правдоподобия при условии X и оценок Z на этом шаге:

$$Q(\theta|\theta^{(t)})=E[logp(X,Z|\theta)]. \quad (1.10)$$

А новые параметры получаются на M-шаге максимизацией этой функции.

На каждом шаге правдоподобие не убывает, что гарантирует сходимость к максимуму, хотя необязательно глобальному.

Этот алгоритм широко применяется в машинном обучении благодаря своей способности работать с неполными и ненаблюдаемыми данными. У него есть недостатки вроде медленной сходимости и требования к возможности аналитического вычисления ожиданий на E-шаге, но во-первых, это базовый алгоритм, и для него есть модификации, например обобщенный EM-алгоритм, который упрощает M-шаг, а во-вторых, он даже в самом базовом виде все равно работает на огромном количестве задач, например в кластеризации, тематическом моделировании, компьютерном зрении и обработке изображении.

В контексте направленных вероятностных графовых моделей он занимает важную роль в работе с ненаблюдаемыми данными и используется в таких представителях моделей этого класса, как динамические байесовские сети и скрытые марковские модели.

# ГЛАВА 2
# ОБУЧЕНИЕ ПАРАМЕТРОВ НАПРАВЛЕННЫХ ВЕРОЯТНОСТНЫХ ГРАФОВЫХ МОДЕЛЕЙ

В этой главе анализируется обучение параметров НВГМ при заданной структуре сети. Приводится общий подход к обучению и показывается его применение на конкретных моделях.

## 2.1 Общий подход к обучению параметров в направленных вероятностных графовых моделях

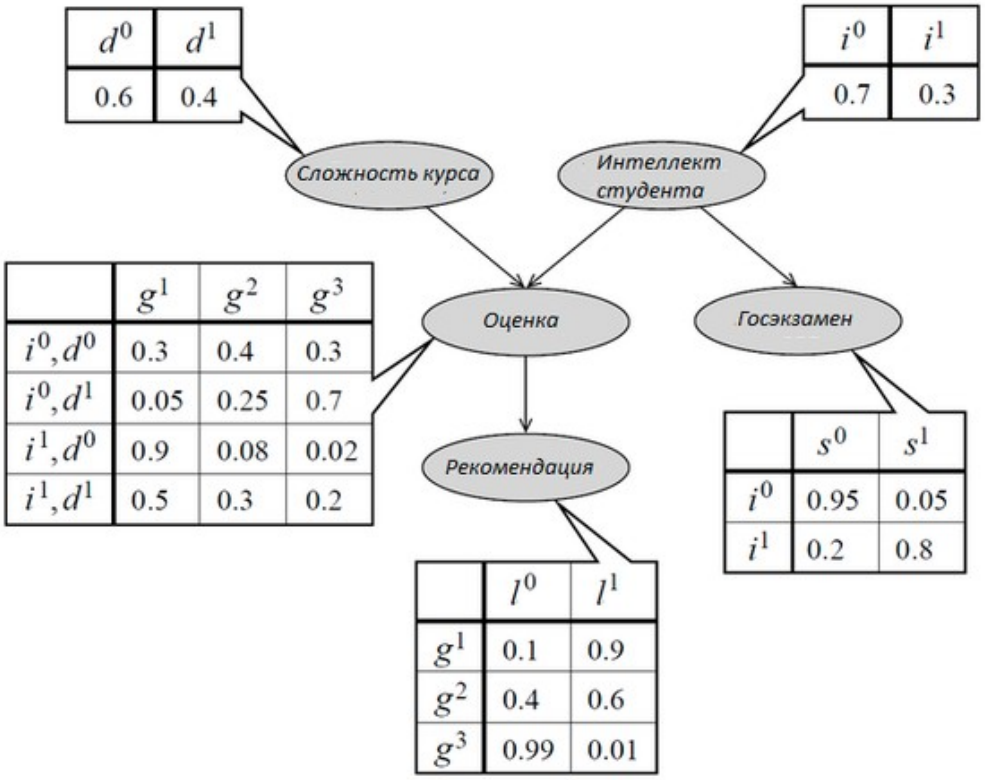
Существует много моделей, относящихся к классу НВГМ. Тем не менее, для каждой из них актуальна задача нахождения параметров из имеющихся наблюдении (тренировочных данных) и, благодаря тому, что их объединяет ряд черт, описанных в предыдущей главе, существует общий подход, применимый к любым моделям этого типа.

По сути, для обучения параметров достаточно всего два метода: метод максимального правдоподобия и EM-алгоритм (который следует из метода максимального правдоподобия). Первый из них применяется для задач, в которых все переменные наблюдаемы. Стоит обратить внимание, что зачастую для улучшения результатов используется не сам метод максимального правдоподобия, а его улучшение с помощью априорных вероятностей, то есть максимизируется апостериорная вероятность, как это было описано в разделе 1.3. EM-алгоритм же применяется для моделей, где часть переменных скрыта (не наблюдается) [7].

Таким образом, эти два метода полностью покрывают все виды направленных вероятностных графовых моделей. Для конкретных типов моделей уже заранее рассчитаны формулы, решающие эти задачи, и вся глава будет посвящена их рассмотрению и анализу.

## 2.2 Байесовская сеть

Байесовская сеть представляет собой компактное представление совместных распределений случайных величин, входящих в модель. Наивная альтернатива такому представлению в случае дискретных распределений – таблица всех совместных распределений, большая часть полей которой является неинформативной или дублирует друг друга. В байесовской же сети, чтобы получать условные распределения, используют различные формулы, зависящие от распределений вершин (значений, которые они могут принимать). Например, для дискретной величины вероятность каждого значения может определяться как взвешенная сумма параметров родителей, а в случае независимости браться непосредственно из априорного распределения. На рисунке 2.1 представлен пример байесовской сети с дискретными случайными величинами.




| $d^0$ | $d^1$ |
|---|---|
| 0.6 | 0.4 |

| $i^0$ | $i^1$ |
|---|---|
| 0.7 | 0.3 |

| | $g^1$ | $g^2$ | $g^3$ |
|---|---|---|---|
| $i^0, d^0$ | 0.3 | 0.4 | 0.3 |
| $i^0, d^1$ | 0.05 | 0.25 | 0.7 |
| $i^1, d^0$ | 0.9 | 0.08 | 0.02 |
| $i^1, d^1$ | 0.5 | 0.3 | 0.2 |

| | $s^0$ | $s^1$ |
|---|---|---|
| $i^0$ | 0.95 | 0.05 |
| $i^1$ | 0.2 | 0.8 |

| | $l^0$ | $l^1$ |
|---|---|---|
| $g^1$ | 0.1 | 0.9 |
| $g^2$ | 0.4 | 0.6 |
| $g^3$ | 0.99 | 0.01 |

**Рисунок 2.1 – Пример байесовской сети**

Одним из преимуществ байесовской сети является то, что она благодаря своей структуре хорошо справляется с ситуациями, когда часть данных отсутствует, так как она находит зависимости между всеми параметрами. Например, если два признака скоррелированы, то классические модели

обучения с учителем при отсутствии одного из них не выдадут правильный результат, в то время как байесовская сеть кодирует такие зависимости в своей структуре. Также байесовскую сеть можно использовать для поиска вероятностей событии при заданных параметрах и причинных связях. Еще одно привлекательное для анализа данных свойство – интерпретируемость. Кроме этого, как будет видно из приведенных далее способов обучения, к байесовской сети легко применить априорные знания о решаемой проблеме, что особенно полезно в условиях ограниченного количества тренировочных данных.

Байесовские сети могут решать несколько задач. Одна из них – это прогнозирование (прямой вывод), то есть определение вероятности события по другим наблюдаемым событиям (причинам). Задача сводится к поиску вероятности события при условии других, что является простой задачей при известных значениях параметров сети благодаря свойству факторизации.

Менее очевидная задача называется обратным выводом и заключается в определении вероятности причин при наблюдаемых следствиях. Для получения ответа используется теорема Байеса.

Еще одна задача, которую могут решать байесовские сети – это межпричинный (смешанный) вывод, когда определяется вероятность одной из причин наступившего события при условии наступления одной или нескольких других причин того же события. В качестве примера можно привести такую задачу: трава мокрая, это сильно повышает вероятность как прошедшего дождя, так и работавшей поливалки, хотя они не влияют друг на друга. Но если станет известно, что шел дождь, вероятность работы поливалки сильно уменьшится, влажность уже объяснена одной из возможных причин, и вторая становится менее вероятной.

Модели этого типа позволяют интерпретировать результаты своей работы, что делает их актуальными для ряда сфер, где решения не могут приниматься «вслепую»: медицина, финансы и многие другие. Но их обучение, как и у других НВГМ имеет свои особенности, и именно на примере байесовской сети в главе 3 будут проанализированы методы обучения НГВМ.

### 2.2.1 Обучение параметров байесовской сети с частично не наблюдаемыми данными

Пусть часть вершин $X = \{x_1, \ldots, x_k\}$ в сети наблюдаема, а оставшиеся $Z = \{z_1, \ldots, z_m\}$ нет. Тогда обычная задача поиска параметров $\theta$, максимизирующих правдоподобие данных, принимает вид:

$$\hat{\theta} = argmax_{\theta} P(X|\theta) = argmax_{\theta} \sum_{Z} P(X, Z|\theta). \quad (2.1)$$

К такой задаче можно применить EM-алгоритм. На E-шаге просиходит поиск апостериорного распределения скрытых переменных для каждого $x_i$:

$$P(Z = z|X = x_i, \theta^{(t)}) = \frac{P(X = x_i|Z = z_i, \theta^{(t)}) P(Z = z|\theta^{(t)})}{P(X = x_i|\theta^{(t)})}. \quad (2.2)$$

Это выражение подставляется в общую формулу $Q(\theta|\theta^{(t)})$, которая максимизируется на M-шаге. Максимизация производится путем переоценки вероятностей через частоты:

$$P^{(t+1)}(X = k|Z = z) = \frac{\sum_{i: x_i = k} P(Z = z|x_i, \theta^{(t)})}{\sum_{i=1}^{N} P(Z = z|x_i, \theta^{(t)})}, P^{(t+1)}(Z = z) = \frac{\sum_{i=1}^{N} P(Z = z|x_i, \theta^{(t)})}{N}. \quad (2.3)$$

Для начальной инициализации можно использовать случайные значения или простейший алгоритм кластеризации k-means. Алгоритм останавливает итерации, когда разница в правдоподобии между двумя последовательными шагами станет меньше заданного порога.

### 2.2.2 Наивный байесовский классификатор

Наивный байесовский классификатор является частным случаем байесовской сети и основан на предположении о том, что все признаки являются независимыми по отношению к переменной класса, то есть каждый признак $A_i$ является условно независимым от всех прочих относительно класса: P ($A_i|A_j$ , C) = P ($A_i$|C), ∀ j ≠ i. Таким образом, наивный байесовский классификатор всегда имеет одинаковую структуру: все признаки являются детьми переменной класса, других связей нет [12].

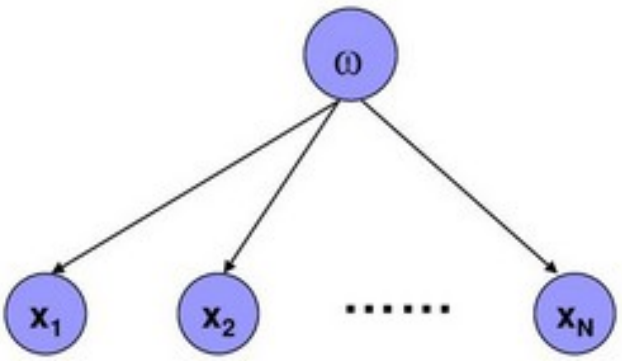



**Рисунок 2.2 – Вид наивного байесовского классификатора**

На рисунке 2.2 представлен общий вид наивного байесовского классификатора, где w – переменная класса, $x_i$ – признаки.

Благодаря простоте структуры вычисление вероятности класса простое, с учетом свойства факторизации получается следующая формула:

$$P(C|A_{1,}A_{2,\dots}A_N)=\frac{P(C)P(A_1|C)P(A_2|C)\dots P(A_n|C)}{P(A)}. \quad (2.4)$$

В уравнении (2.4) P(A), вероятность по данным, используется в качестве постоянной нормализации, все остальные члены формулы получаются в результате обучения.

Обучение наивного байесовского классификатора заключается в определении оценки априорной вероятности класса P(C) и условной вероятности каждого признака относительно класса P($A_i$|C). Эти значения можно получить с помощью субъективных оценок эксперта или алгоритмически с использованием метода максимального правдоподобия. Вероятности оцениваются по тренировочным данным по фиксированным формулам. Например, априорную вероятность класса можно вычислять как долю элементов данного класса среди всех примеров, а условные вероятности зависят от того, непрерывен или дискретен признак. Для категориального признака $A_j$ условные вероятности могут оцениваться по следующей формуле:

$$P(A_j|c_i)\sim\frac{N_{ji}}{N_i}. \quad (2.5)$$

где $N_{ji}$ – количество примеров класса $i$, в которых атрибут $A_j$ принимает значение j,

$N_i$ – количество элементов класса $c_i$ в рассматриваемом наборе данных.

После вычисления оценок параметров апостериорную вероятность можно получить простым умножением априорной вероятности на правдоподобие каждого признака. Таким образом, с учетом значений для m атрибутов $a_{1,}a_{2,\dots}a_m$ для каждого класса $c_i$ апостериорная вероятность пропорциональна следующему произведению:

$$P(c_i|a_{1,}a_{2,\dots}a_m) \sim P(c_i)P(a_1|c_i)\dots P(a_m|c_i). \quad (2.6)$$

В качестве результата классификации будет выбран класс $c_k$, который максимизирует значение этого выражения.

У такой модели много минусов: невозможность находить нелинейные зависимости, предположение о независимости признаков, что на реальных задачах выполняется редко. Кроме этого, если есть дисбаланс классов, то необходимо модифицировать формулы, используемые в обучении. В то же время, простота наивного байесовского классификатора делает его хорошим вариантом для решения простых задач и кандидатом на место минимально работающей модели, которую используют для оценки результатов работы более сложных моделей.

### 2.2.3 Марковская цепь

Марковские цепи – это класс вероятностных графовых моделей, который представляет изменение состояния процесса во времени, то есть это динамическая НВГМ. У графа в моделях этого типа переменные связываются в цепь с дугой от одного состояния к следующему [9]. При этом подразумевается выполнение марковского свойства: состояние в $X_{t+1}$ зависит не от всех предыдущих, а только от предшествующего ему $X_t$ :

$$P(X_{T+1}|X_1, X_2, \dots, X_T) = P(X_{T+1}|X_T). \quad (2.7)$$

Таким образом, в марковской цепи основным параметром является вероятность состояния с учетом вероятности предыдущего состояния. Для дискретных марковских цепей эти параметры задаются, по аналогии с другими НГВМ, в виде матрицы. Такую матрицу A называют стохастической:

$$A_{ij} = P(X_{T+1} = j | X_T = i), \sum_j A_{ij} = 1. \quad (2.8)$$

Такая модель, хотя и является динамической, вследствие своей простоты по сути является частным случаем байесовских сетей и может обучаться так же, как статические модели. Но существуют и более сложные модели.

Марковская цепь предполагает, что каждое состояние является наблюдаемым. Но это предположение не всегда истинно. Во многих практических задачах нет возможности непосредственного наблюдения за состоянием процесса, поэтому необходимо перейти к модели, в которой состояние скрыто. Такая модель называется скрытой марковской.

## 2.3 Скрытая марковская модель

Скрытая марковская модель – вероятностная модель, позволяющая анализировать последовательности данных, в которых некие наблюдаемые события зависят от ненаблюдаемых (скрытых) состоянии. Модели этого типа также подразумевают выполнение марковского свойства, и в общем виде выглядят так:

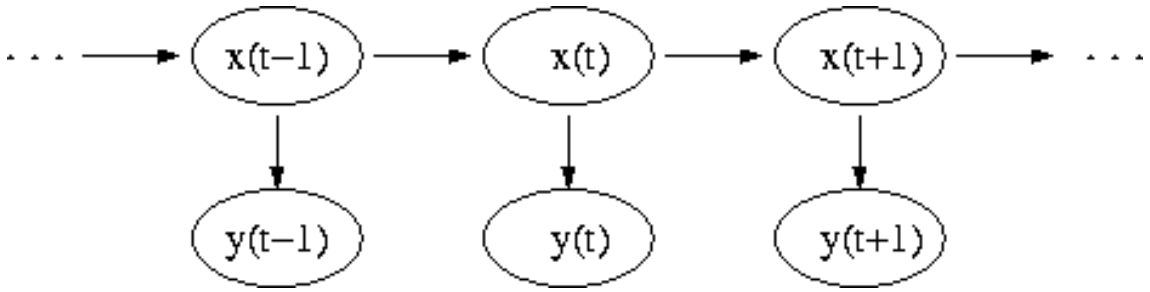



**Рисунок 2.3 – Граф скрытой марковской модели**

На рисунке 2.3 показан общий вид скрытой марковской модели, где X(t) – скрытые состояния, y(t) – наблюдаемые.

Параметры модели ограничиваются начальными распределениями $\pi_i = P(X_1 = i)$, матрицей переходов A, аналогичной рассмотренной в марковской цепи, $A_{ij} = P(X_{t+1} = j | X_t = i)$, а также так называемой матрицей эмиссии B, определяющей переходы между скрытыми состояниями и наблюдениями, $B_{ik} = P(Z_t = k | X_t = i)$ [10].

Такая модель может использоваться для определения вероятностей наблюдении при заданных параметрах (задача оценки), но наибольший интерес представляет задача декодирования, когда по наблюдениям необходимо найти последовательность скрытых состоянии. Такая постановка задачи позволяет

решать такие практические проблемы, как, например, распознавание речи: здесь скрытыми переменными являются фонемы, а наблюдаемыми – звуковые сигналы.

Обучение параметров производится с помощью EM-алгоритма. Для удобства обозначим набор параметров (π, A, B) как λ. Тогда на E-шаге происходит вычисление апостериорных распределений скрытых состоянии. Для этого сначала считают прямым проходом по сети:

$$\alpha_t(i) = P(Z_1, \ldots, Z_t, X_t = i | \lambda). \tag{2.9}$$

И обратным проходом:

$$\beta_t(i) = P(Z_1, \ldots, Z_t | X_t = i, \lambda). \tag{2.10}$$

Используя полученные значения, можно посчитать искомые апостериорные вероятности:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(Z|\lambda)}. \tag{2.11}$$

$$\epsilon_t(i,j) = \frac{\alpha_t(i) A_{ij} B_j(Z_{t+1}) \beta_{t+1}(j)}{P(Z|\lambda)}. \tag{2.12}$$

На M-шаге снова заранее получены формулы вычисления значении, максимизирующих функцию, и параметры получаются определенным образом. Начальные вероятности – это $\pi_i = \gamma_1(i)$, элементы матрицы переходов получаются как:

$$A_{ij} = \frac{\sum_{t=1}^{T-1} \epsilon_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}. \tag{2.13}$$

А элементы матрицы эмиссии:

$$B_{ik} = \frac{\sum_{t=1}^{T} \gamma_t(i) I(Z_t = k)}{\sum_{t=1}^{T} \gamma_t(i)}. \quad (2.14)$$

где $I$ – индикаторная функция, которая принимает значение единицы, когда ее аргумент верен, значение ноля иначе.

Значения для первого шага алгоритма можно выбрать случайным образом.

## 2.4 Выводы

Был проанализирован общий подход к обучению параметров в НВГМ, а также его реализации на конкретных моделях этого класса. В зависимости от выбора модели поиск значений параметров может быть как вычислительно сложным так и наоборот, каждая из проанализированных моделей может решать разные задачи. Таким образом, хотя обучение параметров и является решаемой задачей, специалисту, который хочет воспользоваться их функционалом, стоит тщательно подбирать модель, подходящую именно под его задачу. Кроме этого, некоторые модели не имеют фиксированной структуры сети, и ее нужно обучать отдельными методами. Вся следующая глава будет посвящена именно структурному обучению и там уже не будет общего подхода, который бы хорошо работал на любых моделях.

# ГЛАВА 3
# ОБУЧЕНИЕ СТРУКТУРЫ НАПРАВЛЕННЫХ ВЕРОЯТНОСТНЫХ ГРАФОВЫХ МОДЕЛЕЙ

В самых простых задачах структуру модели могут составить вручную эксперты. Такой подход субъективен, плохо масштабируем, и для сложных задач алгоритмически происходит поиск структуры сети и, возможно после, возможно одновременно, ее параметров (условных распределений).

## 3.1 Поиск структуры графа

Различают 3 основных подхода к поиску структуры направленных вероятностных графовых моделей: основанные на ограничениях, на оценке (score) качества структуры и на регрессии [13].

Первый из этих подходов подразумевает поиск ограничении, то есть условно независимых между собой вершин, и затем построения структуры, удовлетворяющей данным ограничениям. Минус этого подхода заключается в том, что здесь нет ясной метрики качества, поиск критериев производится для вершин попарно и не учитывает многомерные зависимости. Этот подход работает для любых моделей, где условная независимость кодируется графом.

Подходы, основанные на оценке, вводят метрику качества, определяющую, насколько хорошо данная структура подходит под имеющиеся данные. Таким образом, перебирая разные структуры, можно определить лучшую из них. Так как возможных структур экспоненциально много, используют эвристики, например, жадный поиск. Этот подход работает для любых моделей, где можно задать функцию правдоподобия или информационный критерий.

Наконец, последний подход, появившийся позже остальных, основан на регрессии. Регрессионные методы универсальны для моделей, где зависимости можно выразить функционально, особенно с непрерывными данными или нелинейностями. Привлекательным его делает то, что он использует готовые алгоритмы из других областей машинного обучения.

Также возможны гибридные подходы, комбинирующие несколько вышеперечисленных.

Далее приведены описания нескольких самых часто используемых алгоритмов поиска структуры сети. Чтобы была возможность показать конкретные формулы, все алгоритмы анализируются на примере байесовских сетей.

## 3.2 Алгоритмы, основанные на ограничениях

Один из самых простых алгоритмов – это SGS (Sprites, Glymour and Scheines). На начальный момент работы алгоритма граф считается полным ненаправленным, далее, итеративно для каждой пары вершин i и j, если существует подмножество вершин такое, что i d-отделено от j этим подмножеством, то ребро между этими двумя вершинами удаляется, а некоторым ребрам рядом с удаленным присваивается направление согласно определенному правилу. Когда все возможные ребра удалены, ставится направление оставшимся ненаправленным ребрам, тоже согласно логическим правилам.

Этот алгоритм страдает от большой размерности задачи – во-первых, экспоненциально растет количество вычислений, во-вторых, с увеличением подмножеств, используемых в проверке на независимость, количество примеров для данного случая уменьшается, а погрешность используемых при этом статистических методов увеличивается.

Алгоритм PC (Peter Sprites and Clark Glymour) – это усовершенствование SGS. Теперь поиск независимостей происходит от меньших подмножеств к большим, причем максимальный размер подмножества ограничен. Это не решает проблему полностью, но позволяет работать с графами с большим количеством вершин, чем SGS.

Принципиальная проблема предыдущих двух алгоритмов заключается в том, что они ищут зависимости в полном графе. Следующий алгоритм, GS (Grow-Shrinkage), использует другой подход: сначала происходит поиск надмножества марковского покрытия для каждой вершины, и затем только между вершинами из этого покрытия ищутся зависимости. Марковское покрытие ищется приближенно – сначала в полном графе итеративно к покрытию добавляют вершины, которые зависимы с данной относительно всех вершин уже в марковском покрытии, затем, чтобы минимизировать эффект от неточностей в предыдущем шаге, итеративно удаляют вершины, которые независимы с данной относительно всех вершин в покрытии. После этого для каждых двух вершин рассматривается условная зависимость относительно всех

подмножеств, но не полного графа, как в SGS и PC, а только марковского покрытия. Такой алгоритм быстро работает при условии марковских покрытии малых размеров (существует приближенная оценка $O(n^2)$, n – количество вершин в графе), но это условие выполняется не в любой практической задаче. Например, байесовские сети можно использовать для описания пользователей в социальных сетях, и в такой постановке марковские покрытия как правило будут больших размеров.

## 3.3 Алгоритмы, основанные на оценке

Как уже было упомянуто, такие алгоритмы вводят некую метрику качества сети и затем выбирают ту структуру сети, которая получает лучшее качество на этой метрике. Как правило, метрики выбираются таким образом, чтобы поощрять малое количество ребер, что делает задачи оптимизации невыпуклыми и трудноразрешимыми, поэтому выбор структуры происходит перебором.

Существует множество способов оценить качество байесовской сети, речь пойдет про несколько самых часто используемых.

### 3.3.1 Метрика оценки MDL

Одна из таких оценок базируется на принципе минимальной длины описания (далее MDL, minimal description length). Он гласит, что лучшая гипотеза, то есть в данном случае случайные величины (вершины графа), структуру графа и условные вероятности, – это та, которая позволяет сжать данные наилучшим образом. Описание вершин в оценке не участвует, так как вершины одинаковы для любой структуры графа, для остальных же компонентов используется длина битов, необходимых для описания.

Чтобы полностью описать структуру графа, хватает описания родителей; для этого можно использовать количество родителей и индекс такого набора родителей в множестве всевозможных комбинации вершин данного размера, тогда длина описания графа:

$$DL_{graph} = \sum_i \log n + \log\left(C_n^{|\pi(i)|}\right). \tag{3.2}$$

где $n$ – количество вершин в графе,
$|\pi_i|$ – количество родителей вершины $i$.

Чтобы описать условные распределения, используют таблицы. Количество элементов в таблице – это $|\pi_i| * (|x_i| - 1)$, длину каждого элемента таблицы можно оценить как logN / 2, где N – число обучающих примеров. Тогда:

$$DL_{table} = \frac{1}{2}|\pi(i)|(|x_i|-1)\log N. \quad (3.3)$$

И наконец, для описания длины тренировочных данных с использованием сети используют код Хаффмана, в котором длина описания примера основывается на вероятности ее появления:

$$DL_{data} = -\sum_{i}^{N} \sum_{x_i, x_{|\pi(i)|}} count\left((x_i, x_{|\pi(i)|})\right)\log P\left(x_i | x_{|\pi(i)|}\right). \quad (3.4)$$

где $count((x_i, x_{\pi(i)|}))$ – количество появлении $(x_i, x_{\pi(i)|})$ в тренировочных примерах, вероятность можно получить из таблицы.

Итоговая формула длины описания суммирует эти 3 компонента:

$$DL = DL_{graph} + DL_{table} + DL_{data}. \quad (3.5)$$

Интуитивно, здесь первое слагаемое штрафует за большие степени вершин в графе, второе за излишнюю объемность правил, третье отражает, насколько неожиданны данные для модели – чем хуже сеть предсказывает данные, тем больше длина их описания.

### 3.3.2 Метрики оценки K2 и BIC

Очевидный критерий качества модели, который можно максимизировать – это совместное распределение P(D, G), где D – данные, G – структура графа, по теореме Байеса это совместное распределение кратно P(G)P(D|G). В предположении, что тренировочные данные доступны в больших количествах, отбрасывается априорная вероятность, и критерий сводится к P(D|G). Для упрощения вычислении алгоритм K2 предполагает, что параметры сети имеют равномерное распределение Дирихле. Тогда вероятность наблюдать данные при данной структуре вычисляется как:

$$P(D|G)=\prod_{i=1}^{n}\prod_{j=1}^{q_i}\int\prod_{k=1}^{r_i}\theta_{ijk}^{N_{ijk}}\,Dir(\theta_{ij}|\alpha_{ij})\,d\,\theta_{ij}. \qquad (3.6)$$

где $q_i$ – количество возможных конфигурации родителей вершины $X_i$,
$r_i$ – количество возможных значений вершины $X_i$,
$N_{ijk}$ – количество тренировочных примеров, где $X_i=k$ при определенных значениях родителей,
$\alpha_{ij}$ всегда 1, так как распределение равномерно.

На практике для упрощения вычислении от этого выражения берут логарифм.

Метрика BIC (Bayesian Information Criterion) призвана упростить вычисление этой условной вероятности и бороться с переобучением на маленьком количестве данных. LogP(D|G) можно представить в виде:

$$logP(D|G)=\log\int_{\theta}P(D|G,\theta)\,P(\theta|G)\,d\,\theta. \qquad (3.7)$$

Часть выражения под интегралом обозначим как g(θ). Из монотонности интеграла следует, что максимизация левой части выражения эквивалентна максимизации g(θ). Тогда, найдя оптимальное значение параметров $\hat{\theta}$ , можно с помощью разложения Тейлора второго порядка оценить g(θ) как:

$$g(\theta)\approx g(\hat{\theta})-\frac{1}{2}(\theta-\hat{\theta})\,A\,(\theta-\hat{\theta})^{T}. \qquad (3.8)$$

где $A$ – отрицательный гессиан (матрица вторых производных) в $\hat{\theta}$ .

В результате искомое выражение принимает вид:

$$logP(D|G) \approx \log P(D|\hat{\theta}, G) + \log P(\theta 1|G) + \frac{d}{2}\log(2\pi) - \frac{1}{2}\log|A|. \quad (3.9)$$

где $d$ – количество параметров.

Это выражение называется приближением Лапласа, и его относительная ошибка составляет $O(N^2)$, где N – количество примеров в D. Вычисление гессиана связано с большими вычислительными затратами, поэтому может вводиться предположение о независимости между параметрами, что позволяет использовать только диагональные элементы A. Тем не менее, даже так выисления слишком затратны, поэтому предполагают, что тренировочных данных достаточно много и из суммы исключают слагаемые, не зависящие от N, и меняют вид слагаемого с определителем гессиана, так как оно растет логарифмически с увеличением N, тогда остается:

$$logP(D|G) \approx P(D|\hat{\theta}, G) - \frac{d}{2}\log(N). \quad (3.10)$$

Это приближение называется BIC, и так же как для MDL, можно объяснять слагаемые интуитивно: первое оценивает, насколько хорошо сеть предсказывает данные, второе штрафует структуру графа за сложность. В практических задачах важно помнить, что при большом количестве данных регуляризующий член BIC начинает иметь слишком большой вес и склоняет модель к плохим предсказаниям.

Таким образом, K2 – более надежная чем BIC оценка при большом количестве тренировочных данных, но при маленьком она склонна к переобучению, и стоит использовать BIC.

### 3.3.3 Поиск структуры графа с определенной оценкой

Определившись с оценкой, необходимо провести поиск в пространстве возможных структур и определить ту, у которой будет наилучшая оценка.

Стоит упомянуть, что вышеперечисленные оценки обладают полезным на практике свойством, а именно, при модификации структуры нужно заново вычислять оценки только для той части структуры, которая была изменена. Вследствие этого свойства очевидно напрашиваются жадные стратегии, например, так называемый hill-climbing search, когда генерируется некий граф, а затем итеративно на каждом шаге путем ровно одного добавления, удаления или переворота дуги строится множество соседних графов, из которых

выбирается тот, который имеет наилучшую оценку. Процесс повторяется, пока все графы в множестве не окажутся хуже текущего. Проблема данного алгоритма в том, что на каждом шаге нужно вычислять оценки для $O(n^2)$ графов, кроме того, при манипуляциях с ребрами необходимо проверять граф на ацикличность.

Чтобы уменьшить количество оценок, используют модифицированный алгоритм под названием «sparse candidate» (разреженный кандидат). Его суть заключается в том, чтобы вычислять оценку только для наиболее перспективных кандидатов. Таким образом, алгоритм имеет две фазы: на первой (фаза ограничения) для каждой вершины ищутся k наиболее перспективных вершин-кандидатов в родители, на второй (фаза максимизации) вычисляются оценки. Алгоритм останавливается либо когда на следующем шаге все оценки оказались не лучше, чем на предыдущем, либо когда множество кандидатов не изменилось [6].

Для выбора кандидатов, как правило, рассматривают некую оценку, оценивающую зависимость между вершиной и каждой другой. Если вершина Y – родитель X, то X и Y не могут быть независимы. Попарные оценки не могут найти все зависимости, но на практике работают достаточно хорошо и экономят вычислительные ресурсы. Итак, одна из очевидных оценок – это формула взаимной информации:

$$I(X,Y)=\sum_{x,y}\hat{P}(x,y)\log\frac{\hat{P}(x,y)}{\hat{P}(y)\hat{P}(x)}. \qquad (3.11)$$

где $\hat{P}$ – частота встречаемости в тренировочном датасете.

Но такая оценка не учитывает уже имеющуюся на предыдущем шаге структуру сети, поэтому есть несколько подходов, ее расширяющих. Один из них использует дивергенцию Кульбака-Лейблера между частотой и уже имеющейся сетью:

$$measure(X_i,X_j|B)=D_{KL}(\hat{P}(X_i,X_j)\|P_B(X_i,X_j)). \qquad (3.12)$$

где $\hat{P}$ – частота встречаемости в тренировочном датасете,
$P_B$ – значение из сети, если оно имеется, иначе произведение частот $X_i$, $X_j$.

Другой подход – использовать то свойство байесовской сети, что вершина зависима только со своим марковским покрытием. Если вершина $X_i$ зависима от $X_j$ при родителях вершины $X_i$, то $X_j$ – родитель или ребенок Xi. Это позволяет учесть такая формула, как условная взаимная информация:

$$I(X_i, X_j|\pi(X_i)) = \sum_{\pi(X_i)} \hat{P}(\pi(X_i)) D_{KL}(\hat{P}(X_i, X_j|\pi(X_i)) \| \hat{P}(X_i|\pi(X_i))\hat{P}(X_j|\pi(X_i))). \quad (3.13)$$

где $\pi(X_i)$ – родители вершины $X_i$. В случае, когда родителей нет, приходим к обычной взаимной информации.

Алгоритм разреженного кандидата обладает несколькими проблемами. Во-первых, поиск перспективных кандидатов все еще вычислительно затратен, хотя и работает быстрее, чем перебор всех (рассматривается O(n) графов вместо $O(n^2)$). Во-вторых, количество кандидатов в родители – гиперпараметр, значение которого не очевидно. Кроме этого, из-за того, что алгоритм ограничивает количество родителей вершин, построенная им сеть будет плохо аппроксимировать довольно часто встречающуюся на практике ситуацию, когда несколько вершин имеют большую степень, а большинство очень малую.

## 3.4 Алгоритмы, основанные на регрессии

Алгоритмы, основанные на регрессии, не оценивают зависимости между вершинами напрямую, как это делали алгоритмы, основанные на ограничениях и оценке, а моделируют значение вершины в виде функции, зависящей от ее родителей. Такая функция может иметь вид, например, взвешенной суммы значений родителей.

Методы, основанные на регрессии, в основном различаются функцией потерь. Один из типов функции потерь – это такие функции, которые призваны найти соседей для каждой вершины. Для этого каждая вершина моделируется в виде комбинации всех остальных, и только те вершины, которые оказывают наибольшее влияние на предсказание считаются соседями. Для этого можно, например, использовать линейную комбинацию вершин и L1 регуляризацию (регуляризацию Лассо), чтобы занулить веса мало влияющих из них:

$$Objective = argmin_\theta \frac{1}{n} \|(x_i - \theta^T x)\|_2^2 + \lambda \|\theta\|_1. \quad (3.14)$$

Здесь используется Ln норма, а в первом слагаемом считается, что вес при $\theta_i$ всегда 0.

В качестве функции потерь можно использовать и те же функции, что и в методах, использующих оценки, например длину минимального описания, добавляя к ним дополнительные ограничения, в частности регуляризаторы.

## 3.5 Гибридные алгоритмы

Некоторые алгоритмы комбинируют несколько подходов в попытке использовать их сильные стороны. Примером такого алгоритма может служить Max-min hill-climbing алгоритм. Аналогично hill-climbing алгоритму, он оценивает соседние структуры с помощью оценки, но выбирает эти структуры с помощью другого алгоритма, Max-min Parents and Children (MMPC), который относится к классу алгоритмов, основанных на оценке.

## 3.6 Выводы

После описания трех подходов, можно сделать выводы о том, в каких ситуациях использовать какие или их гибриды.

Методы, основанные на ограничениях, для хорошего результата требуют много тренировочных данных и не слишком сложных зависимостей, так как они строят структуру поэтапно. Одно из больших преимуществ этого подхода – интерпретируемость результатов их работы.

Методы, основанные на оценке, оценивают всю структуру, а не только ее части, как методы с ограничениями, но это происходит за счет больших вычислительных затрат. Также здесь необходимо подбирать два подалгоритма – функцию оценки и способ перебора структур, что усложняет задачу. Алгоритмы этого типа часто используют вместе с другими, чтобы упростить перебор.

К преимуществам алгоритмов, основанных на регрессии, можно отнести то, что они тривиально работают как с непрерывными, так и с дискретными величинами, а также возможность использования их для поиска нелинейных зависимостей. Один из недостатков же напрямую вытекает из преимущества – такие алгоритмы требуют сильных предположений о виде зависимости.

Гибридные подходы комбинируют сильные стороны алгоритмов, но требуют более сложной настройки, из-за чего их использование может быть необоснованно.

# ГЛАВА 4
# ПРИМЕНЕНИЕ НАПРАВЛЕННЫХ ВЕРОЯТНОСТНЫХ ГРАФОВЫХ МОДЕЛЕЙ В ПРАКТИЧЕСКИХ ЗАДАЧАХ

## 4.1 Использованные технологии

Практическая часть была выполнена с использованием языка программирования Python. Этот язык является одним из самых эффективных инструментов в сфере машинного обучения вследствие большого количества реализации разнообразных алгоритмов и дополнительных инструментов, применяемых в этой сфере. Эти реализации поступают в виде импортируемых в программу пакетов (библиотек), и далее будут описаны те, которые использовались в данной работе.

Центральным пакетом в ходе проведения экспериментов стал pgmpy, в котором реализованы основные алгоритмы для каузального и вероятностного моделирования с использованием байесовских сетей. Она предоставляет интерфейс для построения, обучения и анализа направленных вероятностных графовых моделей. В частности, в этом пакете реализованы основные алгоритмы обучения структуры и параметров, описанные в предыдущей главе, причинного и вероятностного прямого вывода, генерации данных при заданной сети. Код данного пакета находится в открытом доступе.

Для связанных с машинным обучением операции использовался пакет scikit-learn. Он реализует алгоритмы для обработки данных, построения моделей и их оценки. В ходе рассмотрения практической задачи для данной работы использовались некоторые методы и модель линейной регрессии с регуляризацией из данного пакета.

Для визуализации данных и результатов экспериментов использовался пакет matplotlib, позволяющий создавать статические, анимированные и интерактивные графики в Python. Matplotlib используют для визуализации данных любой сложности. Пакет позволяет строить разные варианты графиков через объектно-ориентированный интерфейс, когда каждая фигура или ее часть является отдельным объектом, что позволяет выборочно менять их свойства и отображение.

## 4.2 Рассматриваемые задачи

Чтобы протестировать различные алгоритмы структурного обучения байесовских сетей, были использованы несколько задач разного размера. Все эти задачи имеют эталонный вариант байесовской сети, разработанный экспертами, что позволяет оценить качество полученной в результате использования алгоритмов сети. В качестве тренировочного датасета использовались данные, сгенерированные из эталонной сети. Для генерации была проведена топологическая сортировка вершин, затем в отсортированном порядке генерация значений вершин из априорного распределения для узлов без родителей и из условного для остальных [4]. Итоговый тренировочный датасет содержит заданное количество примеров, каждый из которых представляет из себя вектор из значений вершин.

Первая, самая маленькая задача моделирует медицинскую диагностику заболевании легких в связи с поездкой в Азию. Как видно на рисунке 4.1, она состоит из 8 бинарных признаков, связанных причинно-следственными отношениями. Эта задача (далее сеть Asia) часто используется для демонстрации методов байесовских сетей вследствие своей простоты, и не должна представлять сложностей для алгоритмов структурного обучения.

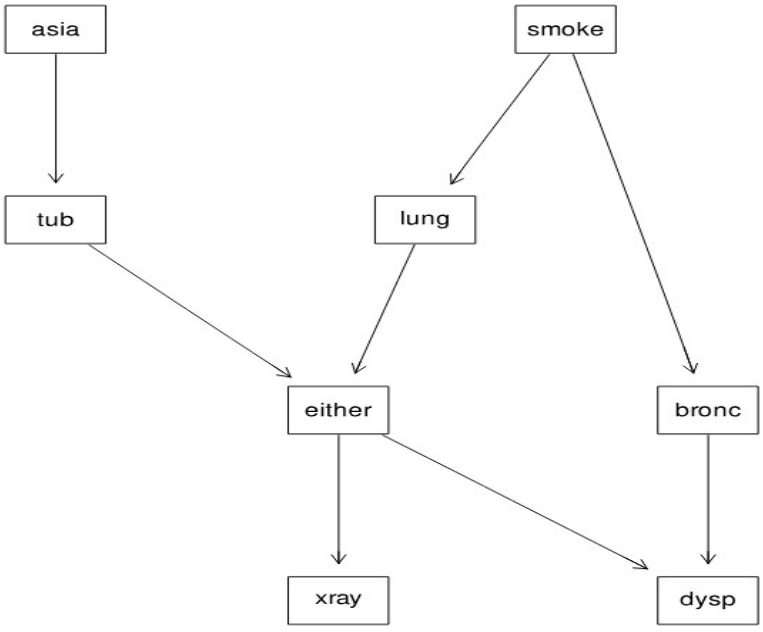



**Рисунок 4.1 – Эталонная структура сети Asia**

Следующая задача имеет больше переменных, в частности 20. Разница с предыдущей задачей может показаться небольшой, но стоит учитывать, что вычислительная сложность многих методов структурного обучения растет экспоненциально с количеством вершин. Итак, следующая эталонная сеть (далее сеть Child) моделирует медицинскую диагностику у детей, включая различные факторы здоровья и симптомы заболевании. Переменные здесь категориальные, но, в отличие от предыдущей задачи, не обязательно бинарные. Эта сеть представлена на рисунке 4.2.

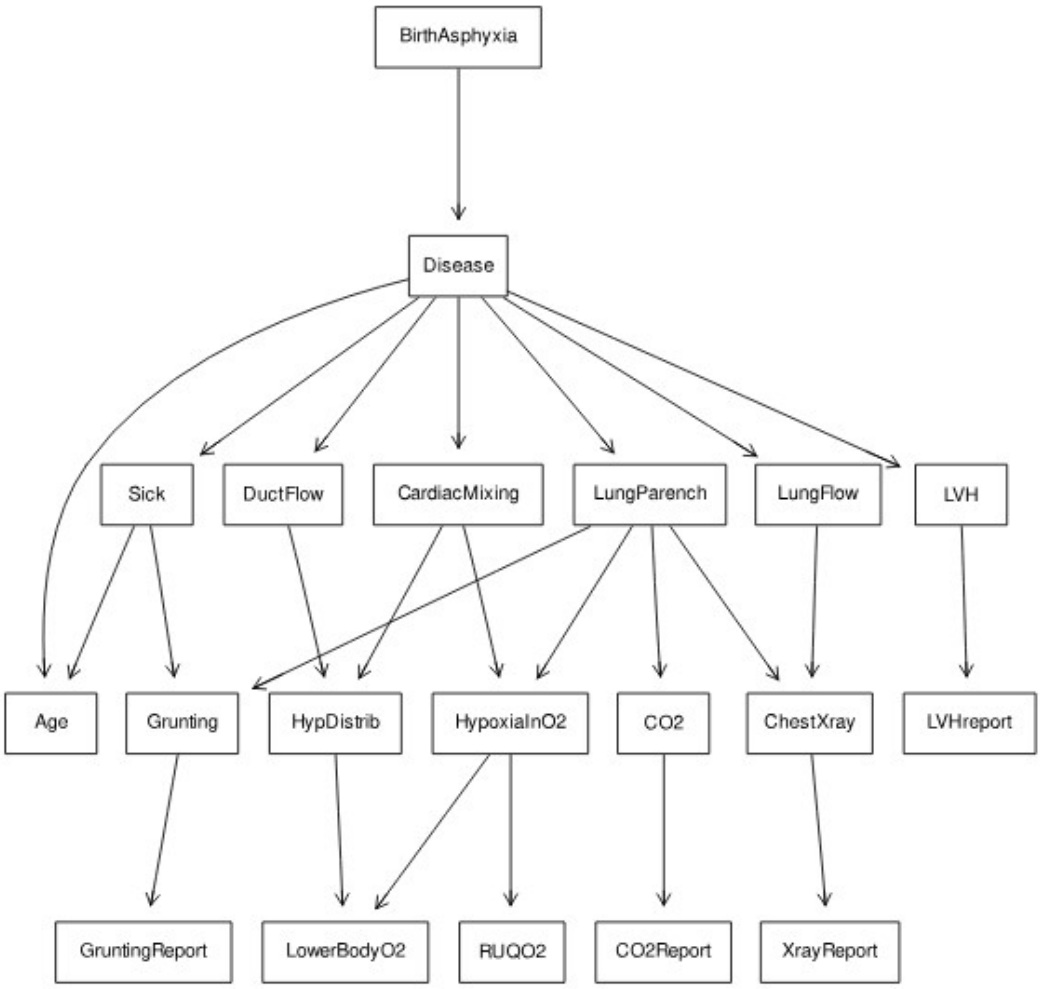



**Рисунок 4.2 – Эталонная структура сети Child**

Следующая сеть (далее датасет Hailfinder) была создана в 1990-х годах метеорологами и специалистами по байесовским сетям для моделирования прогнозирования градовых бурь. Эта задача использовалась в ряде исследований, посвященных байесовским сетям, так как она предоставляет объем переменных, достаточный для решения многих реальных задач, что делает использующие ее работы актуальными. Содержит 56 категориальных вершин, в основном бинарных. Включает атмосферные условия,

прогнозируемые события и другие параметры, которые могут давать информацию о возможности появления градовых бурь. Эта сеть представлена на рисунке 4.3.

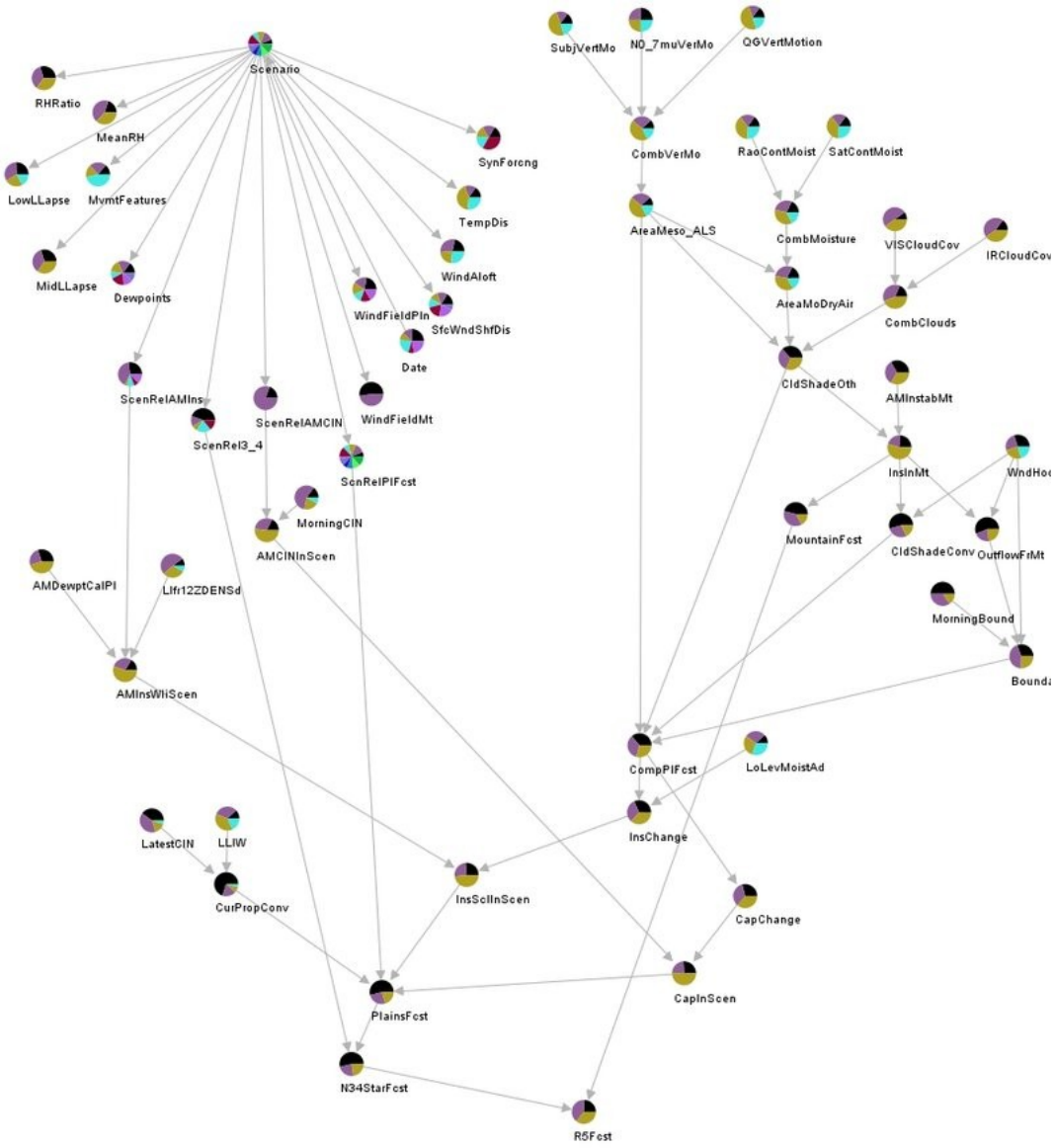



**Рисунок 4.3 – Эталонная структура сети Hailfinder**

Наконец, самая большая задача, использованная в данной работе – это сеть под названием Pathfinder. Она была создана в Стэнфордском университете для помощи патологоанатомам в диагностике лимфопролиферативных заболеваний, и в оригинальном исследовании показала точность 92%, сопоставимую с результатами работы врачей. Имеет уровневую структуру – сначала рассматриваются наблюдаемые признаки, например результаты анализов или возраст пациента, от них идут связи к промежуточным гипотезам, и только после этого ставится финальный диагноз. Эта сеть содержит 109

вершин, из которых 103 бинарных и 6 категориальных. Эта сеть представлена на рисунке 4.4.

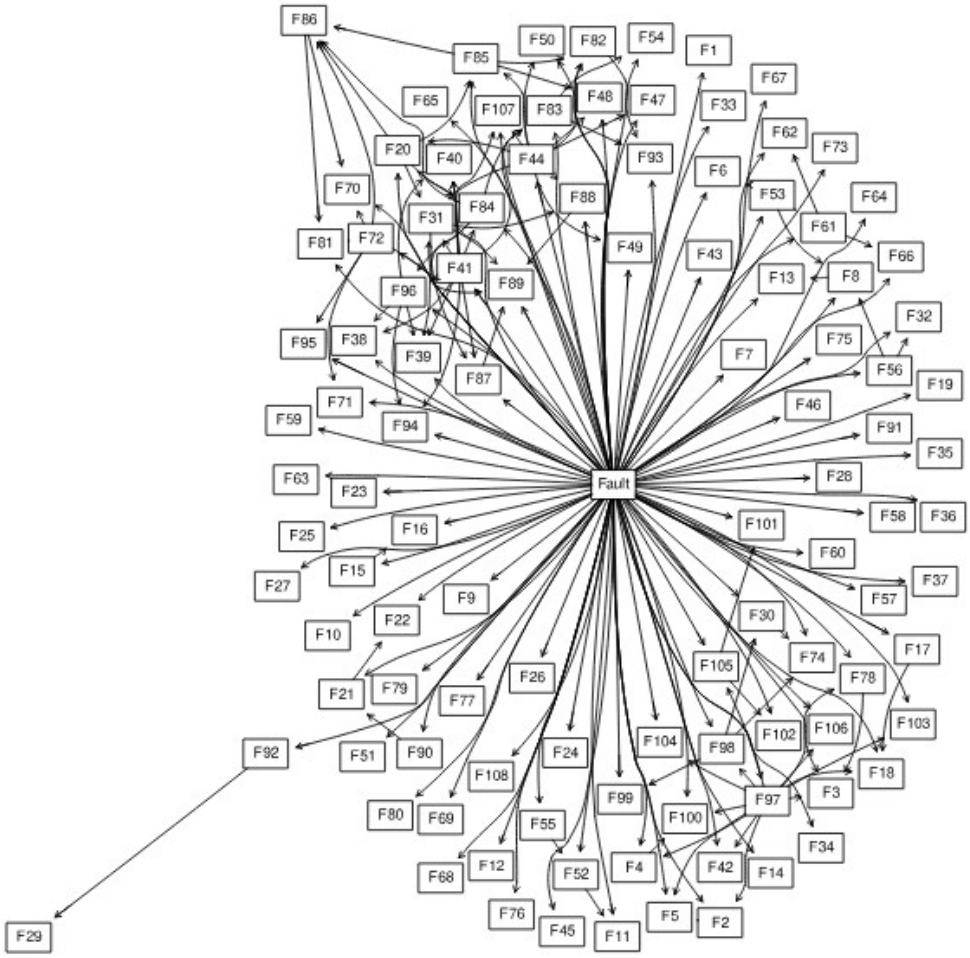



**Рисунок 4.4 – Эталонная структура сети Pathfinder**

Разные размеры задач помогают оценить эффективность анализируемых алгоритмов структурного обучения. Чтобы добиться максимальной объективности оценки, задачи запускались на разном количестве тренировочных примеров, а также с добавлением шума к данным.

## 4.3 Анализ методов поиска структуры графа

В качестве примеров разных подходов к обучению структуры сети были выбраны несколько алгоритмов. Алгоритм PC как представитель методов, основанных на ограничениях, hill-climbing алгоритм с метриками BIC и K2 как представители метода, основанного на оценке, регрессия Лассо как представитель подходов, основанных на регрессии.

Обучение параметров в фиксированной структуре сети было произведено методом максимального правдоподобия при каждом из подходов. Данные для тренировки были сгенерированы из эталонной сети согласно описанному в разделе 4.2 алгоритму.

Еще одним важным шагом был выбор метрики оценки результатов. В силу того, что анализировались алгоритмы обучения структуры графа, ошибку было решено считать с использованием предсказаний ребер. К ним была применена F1-score – это метрика, обычно используемая в задачах бинарной классификации. Она опирается на такой инструмент оценки моделей, как матрица путаницы. Матрица путаницы состоит из четырех элементов – количества правильных и неправильных предсказании каждого из классов. F1-score можно адаптировать для оценки структуры байесовской сети относительно эталонной, используя предсказания существования ребер для заполнения матрицы путаницы:

$$F1-score=\frac{TP}{TP+\frac{1}{2}(FP+FN)}. \tag{4.1}$$

где $TP$ – количество правильно предсказанных существующих ребер,
$FP$ – количество неправильно предсказанных существующих,
$FN$ – количество неправильно предсказанных отсутствующих.

Выбор именно f1-score, а не просто доли правильных предсказании (точности) обусловлен тем, что в рассматриваемых задачах будет преобладать отсутствие ребер, и модель, предсказывающая всегда отсутствие ребра, будет иметь большую точность. F1-score же присвоит такой модели низкую оценку вследствие акцентирования на полноте (доле правильных предсказаний существующего ребра из всех существующих ребер) предсказания.

Кроме f1-score для каждого метода измерялось время работы.

Код для последующих экспериментов приведен в приложениях.

### 4.3.1 Сеть Asia

Методы запускались на 1000, 10000 и 50000 примеров. В ходе проведения экспериментов были подтверждены тезисы из выводов предыдущей главы.

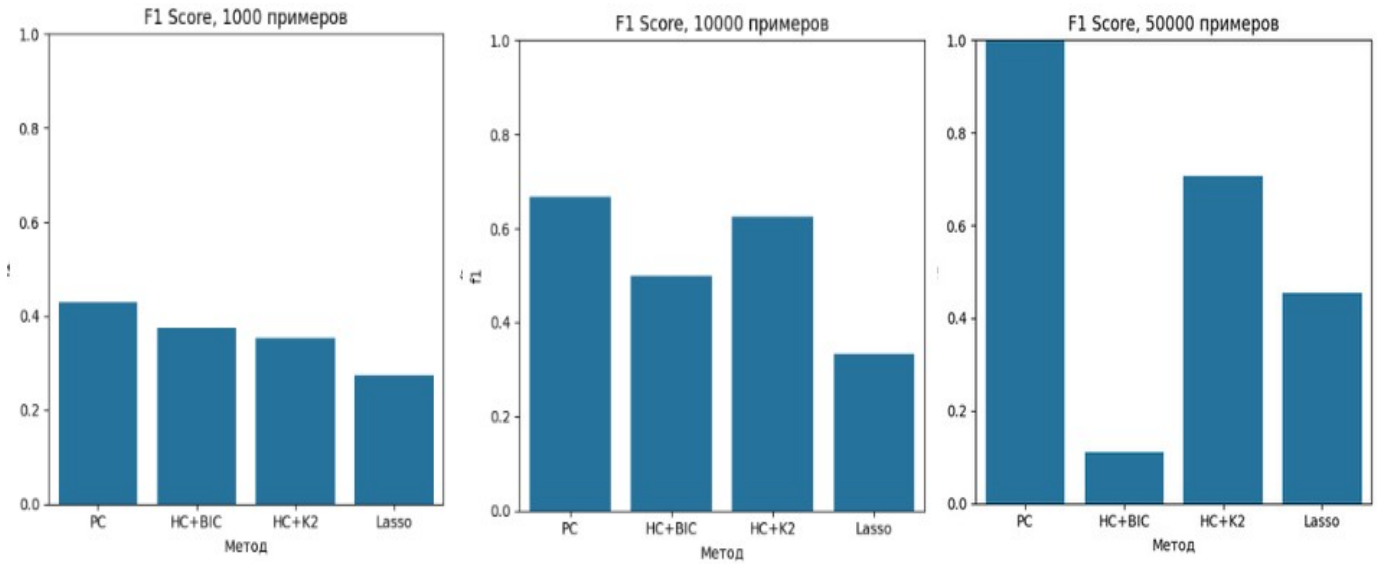



**Рисунок 4.5 – Результаты работы методов, сеть Asia**

Из результатов, представленных на графиках на рисунке 4.5, можно сделать несколько выводов. PC, метод, основанный на ограничениях, работает лучше остальных при условии наличия большого количества данных, но результаты ухудшаются при уменьшении объема тренировочной выборки. Он работает дольше остальных, но на задаче малого размера эта разница не заметна, в отличии от разницы в качестве предсказаний.

Методы, основанные на оценке, довольно стабильны, но оценка K2 всегда дает лучшие результаты на большом количестве данных, а BIC наоборот, начинает работать хуже.

Регрессионный метод может работать неплохо, устойчив к малым количествам тренировочных данных, но требует настройки гиперпарметров и подбора нелинейной функции, что делает его использование необоснованно сложным.

### 4.3.2 Сеть Child

Метод, основанный на регрессии, снова сработал намного хуже остальных и работал при этом значительно дольше остальных. Из этого эксперимента и эксперимента с сетью Asia можно сделать вывод, что методы, основанные на регрессии значительно уступают двум другим подходам. Судя по отсутствию готовых реализации алгоритмов, основанных на регрессии, серьезные исследования в этом направлении не проводились. Хотя метод и выглядит перспективно и позволяет не перебирать экспоненциально большое

количество структур, что в менее наивной реализации позволило бы работать быстрее, чем остальные методы, такой подход лишает направленные вероятностные графовые модели главного преимущества – интерпретабельности. Далее Лассо не будет участвовать в графиках, чтобы не искажать масштаб разницы между другими методами. На рисунке 4.6 представлены результаты работы методов на разном количестве примеров.

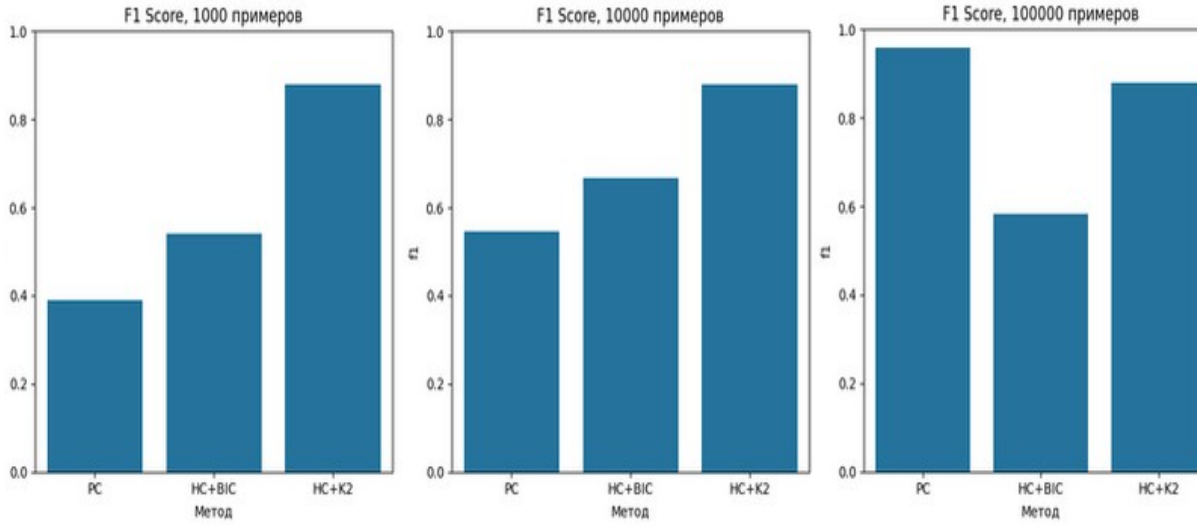



**Рисунок 4.6 – Результаты работы методов, сеть Child**

Как видно на графиках, представленных на рисунке 4.7, здесь значительна разница во времени работы метода, основанного на ограничениях, и методов, основанных на оценке. Интересно, что хотя у алгоритмов, основанных на оценке, асимптотика примерно оценивается факториалом, на таком размере эвристики помогают очень значительно сократить время обучения.

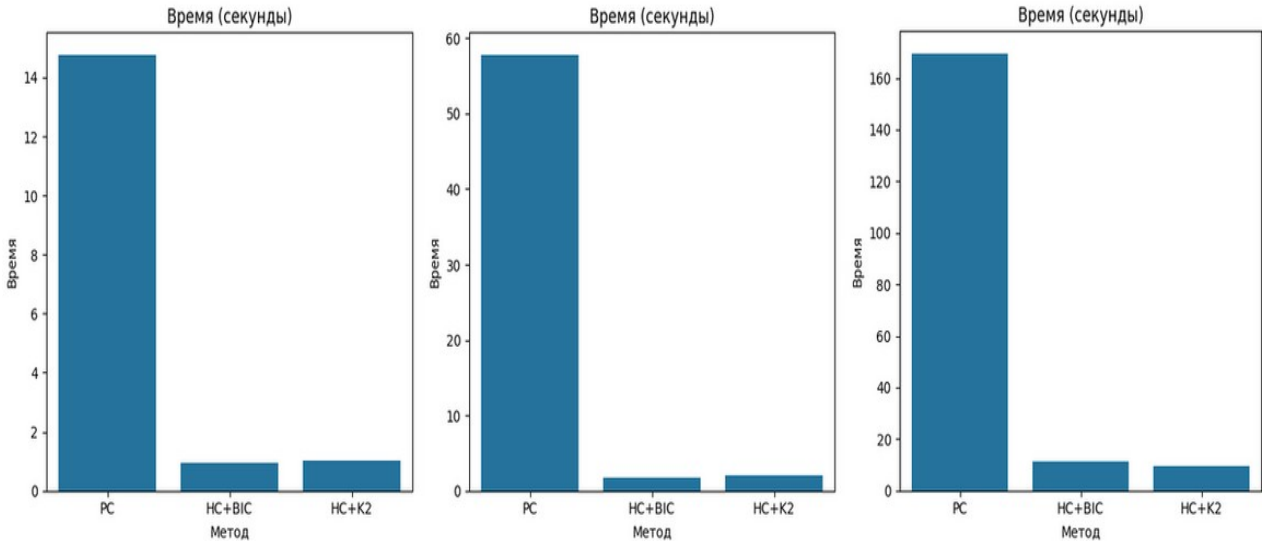



**Рисунок 4.7 – Время работы методов, сеть Child**

### 4.3.3 Сети Hailfinder и Pathfinder

Результаты экспериментов для сети Hailfinder представлены на графиках на рисунках 4.8 и 4.9. При увеличении количества вершин обучение алгоритмов занимает все больше времени: на сети из 56 узлов с 10000 тренировочных примеров алгоритмы работали больше 10 часов. Это время можно значительно сократить, выставив ограничение на максимальное количество родителей у методов, основанных на оценке, и уменьшив размеры рассматриваемых условных множеств в методах, основанных на ограничениях. В частности, использовав в обоих значение 3, были получены следующие результаты.

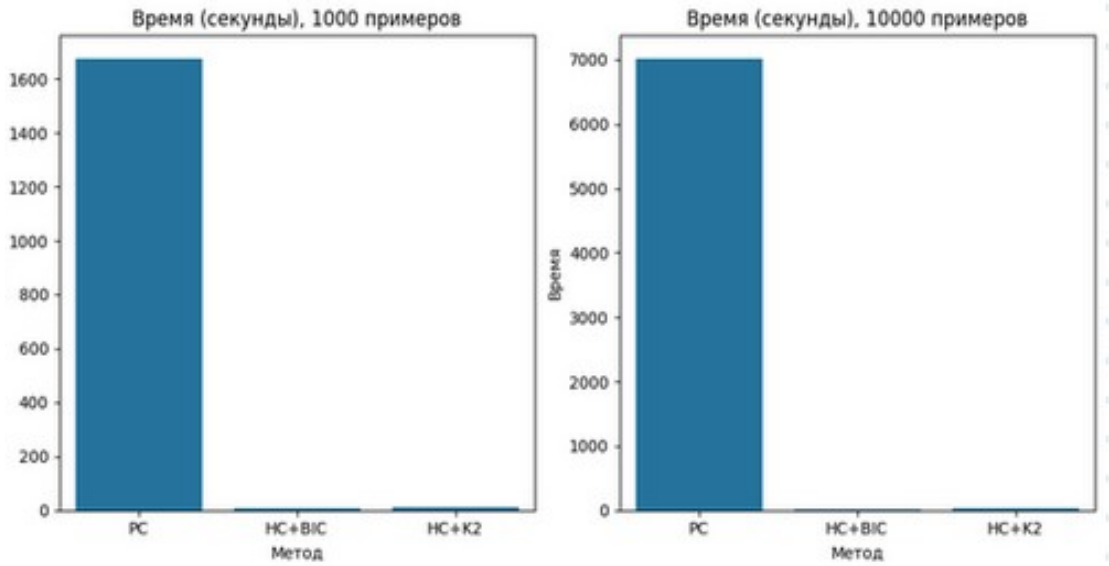



**Рисунок 4.8 – Время работы методов, сеть Hailfinder**

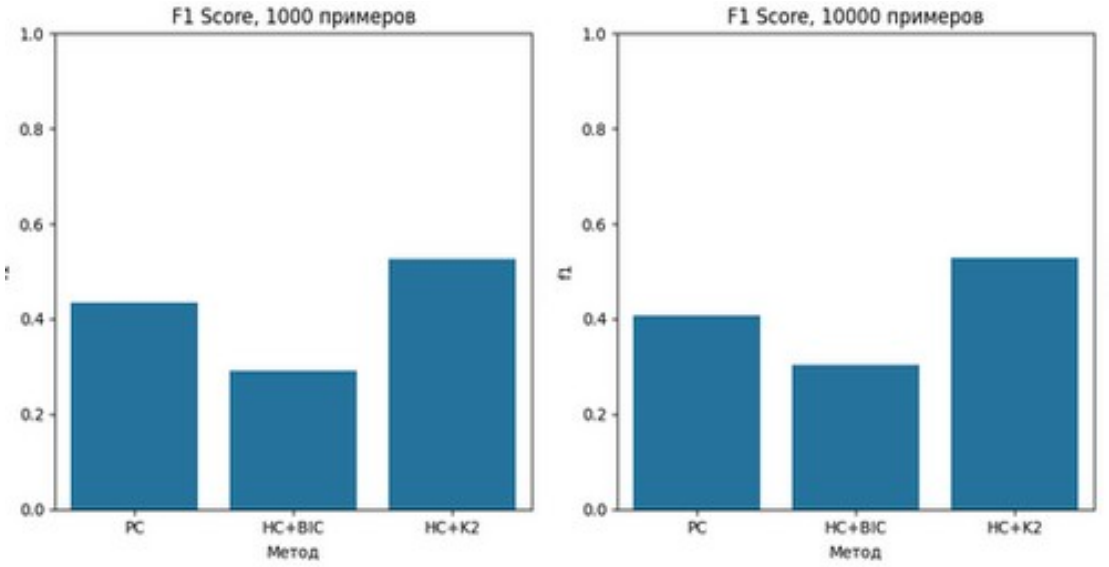



**Рисунок 4.9 – Результаты работы методов, сеть Hailfinder**

Такие ограничения ухудшают качество работы алгоритмов. Кроме того, хотя в большинстве задач, на которые есть эталонная сеть, максимальное количество родителей не превышает 6, а средний размер марковского покрытия 4, нет гарантий, что на новой задаче эти значения будут такими же.

Обучение для сети pathfinder запускалось на 5000 тренировочных примеров с теми же ограничениями на количество родителей, что и для hailfinder.

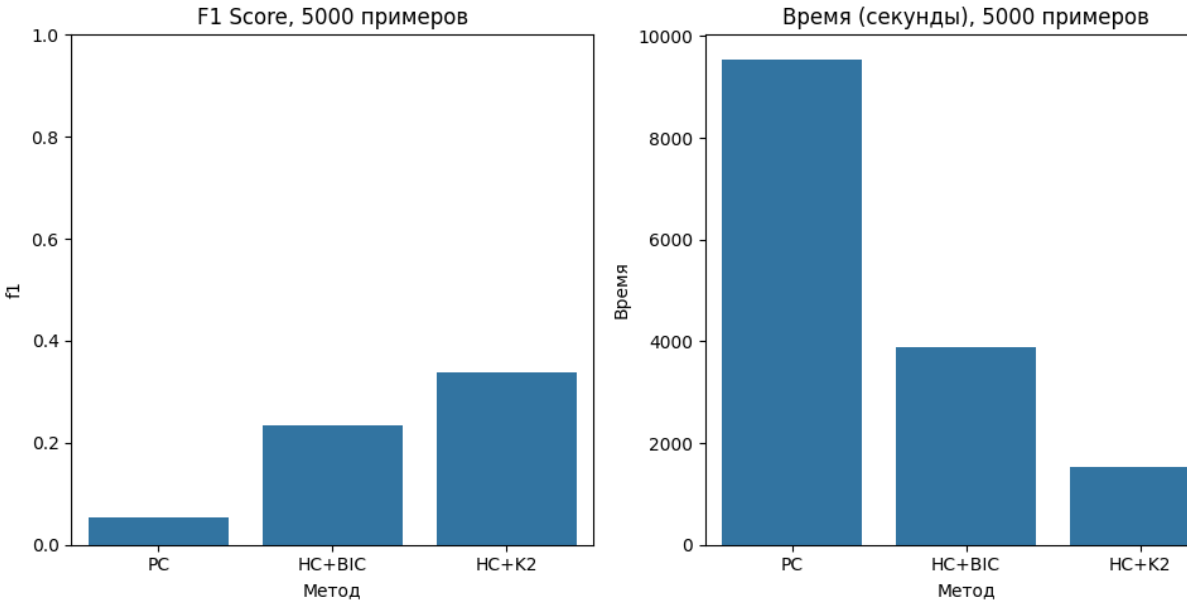



**Рисунок 4.10 – результаты для сети Pathfinder**

Результаты работы этих сетей, представленные на графике на рисунке 4.10, подтверждают теоретический вывод, что из-за экспоненциального роста сложности алгоритмов, их применение на больших задачах проблематично. Стоит отметить, что хотя рассмотренные задачи и эффективно обучали с использованием анализируемых методов на более мощных вычислительных машинах, в реальных задачах количество параметров может быть гораздо больше.

## 4.4 Анализ на разном количестве тренировочных примеров

В реальных задачах, где нет эталонной сети и, соответственно, возможности сгенерировать сколько угодно тренировочных данных, количество примеров может значительно варьироваться. Чтобы лучше продемонстрировать сильные и слабые стороны рассматриваемых методов,

были проанализированы результаты работы на сети Asia с количеством тренировочных примеров от 1000 до 100000 с шагом в 2000 вплоть до 50000 и шагом в 5000 после. Для каждого количества примеров проводилось по 5 запусков, каждый с разными данными.

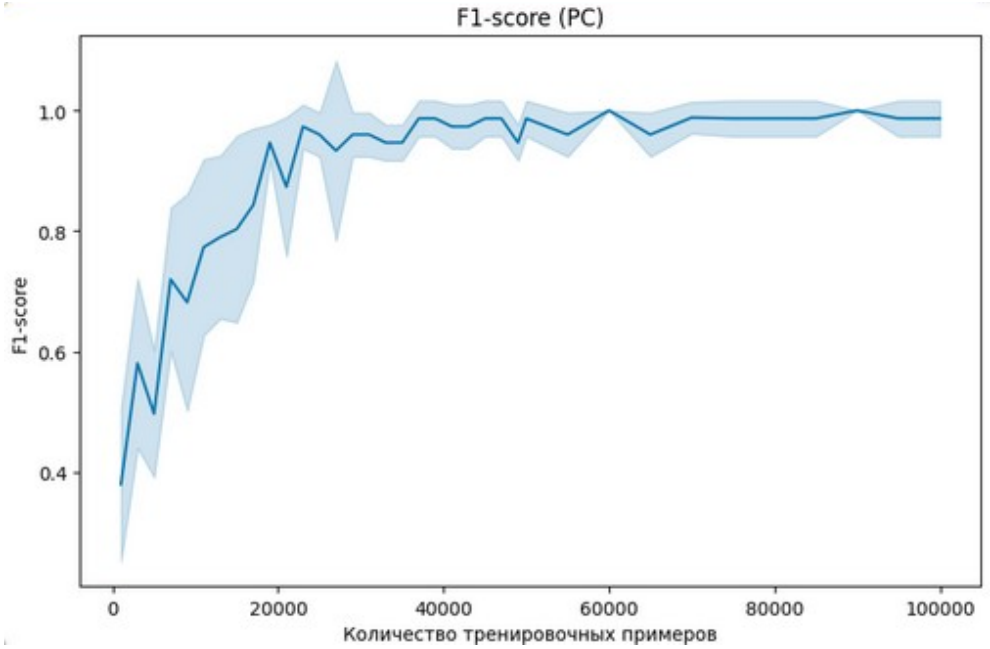



**Рисунок 4.11 – Запуск PC на разном количестве примеров**

Как видно на графике на рисунке 4.11, качество предсказаний алгоритма PC растет с увеличением примеров и может достигать очень хороших результатов, но для этого нужно значительно больше тренировочных примеров, чем для остальных алгоритмов.

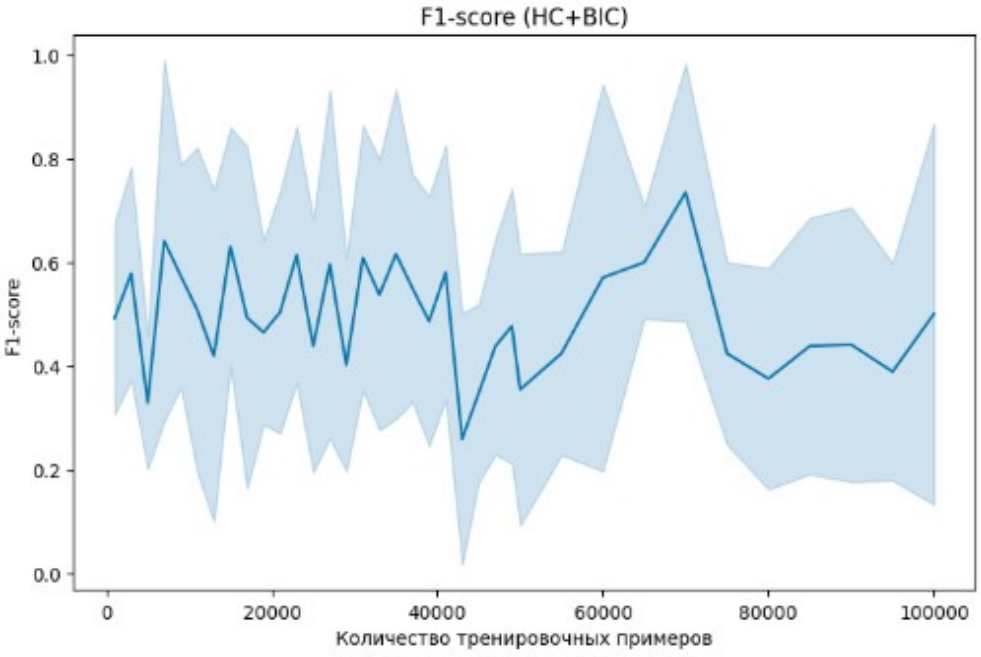



**Рисунок 4.12 – Запуск BIC на разном количестве примеров**

Из графика, представленного на рисунке 4.12, видно, что метод, использующий критерий оценки BIC, как и ожидалось, довольно стабилен при маленьком количестве примеров и страдает от слишком большого, когда регуляризационный член начинает перевешивать остальную часть формулы.

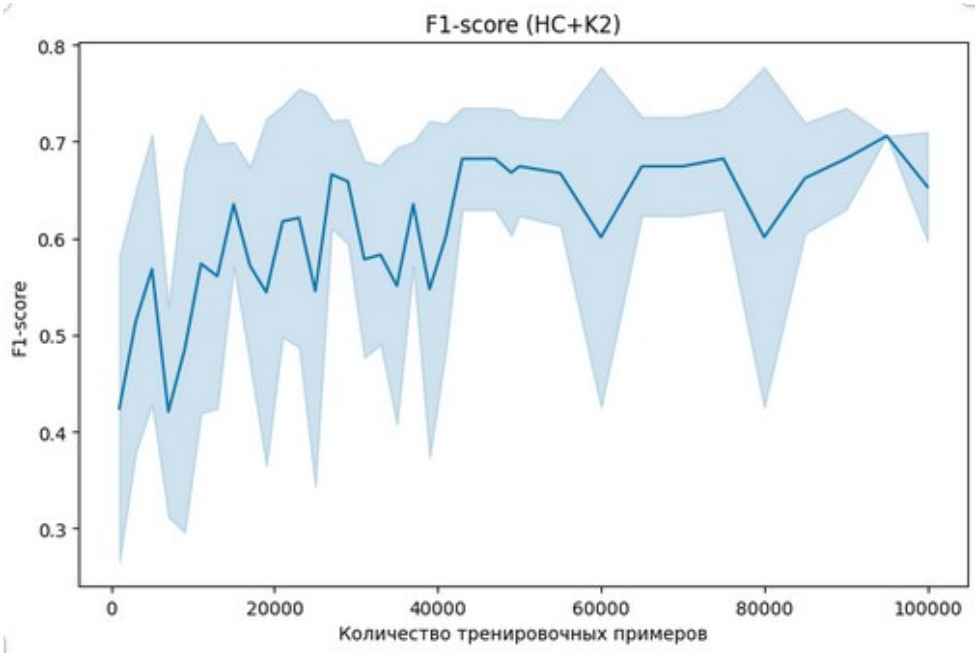



**Рисунок 4.13 – Запуск K2 на разном количестве примеров**

Как видно на графике на рисунке 4.13, оценка работы K2 быстро растет при очень маленьком количестве примеров, после этого замедляется, но все еще получает прибавку в качестве с увеличением количества данных.

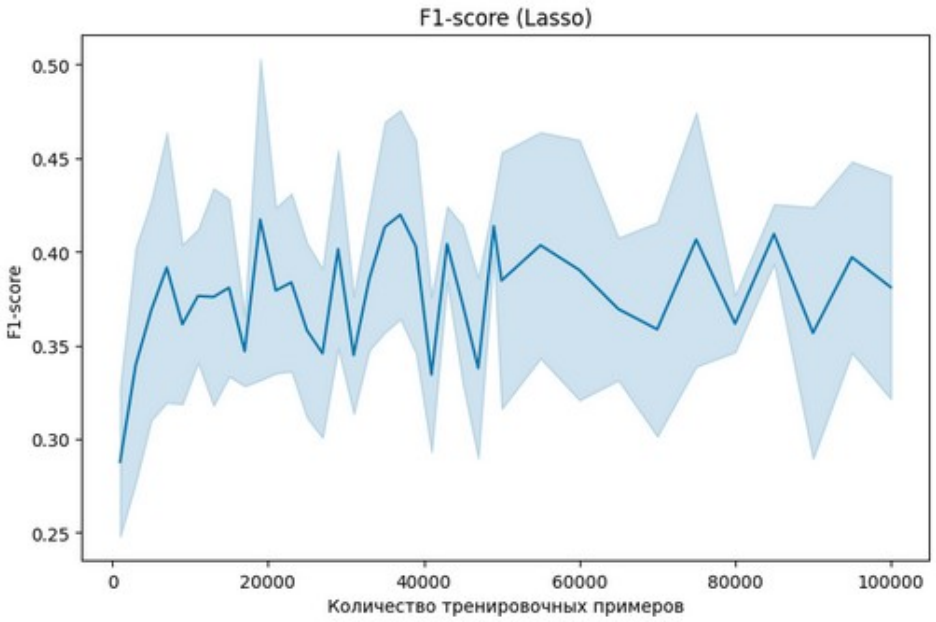



**Рисунок 4.14 – Запуск Lasso на разном количестве примеров**

Результаты эксперимента, представленные на рисунке 4.14, подтверждают, что метод, основанный на регрессии, мало улучшает свои результаты с увеличением количества тренировочных примеров после небольшого порога. Стоит обратить внимание, что он работает хуже остальных методов на любом количестве примеров.

Из графиков видно, что PC работает стабильнее остальных алгоритмов. Это можно объяснить тем, что во-первых, PC в принципе хорошо работает на разреженных задачах, каковой является Asia. Кроме того, жадный алгоритм перебора структур у методов, основанных на оценке, может попадать в локальные минимумы, а также сильно зависеть от того, насколько удачно была выбрана начальная точка.

## 4.5 Анализ результатов при добавлении шума

В практических задачах, в результате несовершенности измерительных приборов, человеческого фактора и других причин, данные зачастую

поступают с ошибками. Таким образом, для применимости в реальных задачах важно, чтобы методы были устойчивы к аномалиям в данных.

Тесты с добавлением шума производились на сети Asia, все вершины в ней бинарны. Чтобы сымитировать шум, для каждого из признаков в фиксированной доле тренировочных примеров значения менялись на противоположные.

По результатам экспериментов, представленным на рисунках 4.15 и 4.16, можно увидеть, что хотя предсказания на шумных данных менее стабильные и в среднем хуже, алгоритмы не перестают работать и показывают результаты, сопоставимые с полученными на идеальных данных. На графиках приведены сравнения работы с шумом и без, причем уровень шума довольно большой, и можно ожидать, что в большинстве реальных задач он не будет превышать использованного, что свидетельствует о том, что алгоритмы применимы на практике.

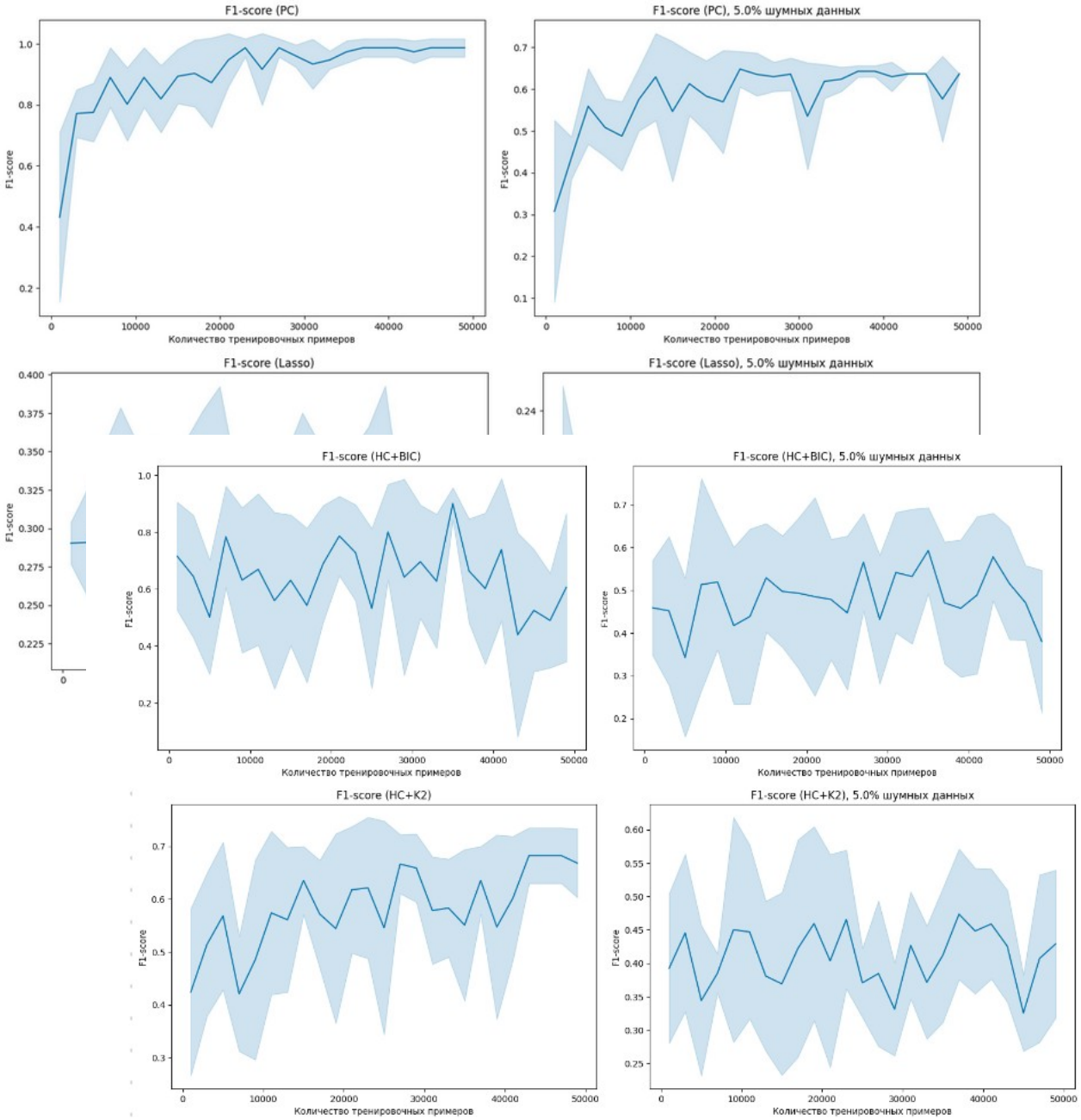



**Рисунок 4.16 – Запуск BIC и K2 на разном количестве примеров с добавлением шума**

## 4.6 Выводы

Проведенные эксперименты позволяют сделать ряд выводов, как о подходах к обучению байесовских сетей, так и о сфере использования моделей этого типа.

Все проанализированные методы страдают от высокой размерности задач, причем реальные задачи могут быть значительно больше рассмотренных. Таким образом, байесовские сети стоит применять, только если необходима высокая интерпретируемость результатов работы, иначе будет лучше использовать другие методы машинного обучения. Если размерность очень большая, то обучение структуры приведенными алгоритмами невозможно. Большой проблемой является то, что, в то время как обучение многих других моделей машинного обучения вследствие большого количества матричных операции можно ускорять с помощью видеокарт, проанализированные алгоритмы не распараллеливаются и не получают прироста в скорости работы. В частности, алгоритмы, основанные на ограничениях, должны проверять наборы родителей последовательно, а основанные на оценке используют жадные алгоритмы, которые так же не могут работать параллельно.

Алгоритм, основанный на регрессии, показал худшие из всех алгоритмов результаты. Он предлагает способ избавиться от экспоненциального роста сложности, присущего другим алгоритмам структурного обучения, но теряет в интерпретабельности результатов. Также для этого метода нужно подбирать большое количество гиперпараметров, что значительно замедляет процесс разработки. Если объяснимость результатов не в приоритете, можно использовать другие методы машинного обучения, которые будут работать надежнее и быстрее. О его несовершенстве на практических задачах также свидетельствует тот факт, что в библиотеках, посвященных байесовским сетям, присутствуют стандартные реализации алгоритмов, основанных на ограничениях и оценке, но не на регрессии.

Алгоритм, основанный на ограничениях, показывают хорошие результаты на задачах с разреженными данными, но требует очень много тренировочных данных, причем с увеличением тренировочных данных он работает все дольше. Таким образом, это хороший вариант для задач небольшой размерности, но на больших задачах тренировочных данных может быть гораздо меньше, чем нужно, а экспоненциальный рост сложности будет требовать уменьшения количества тестов, что негативно скажется на конечном результате.

Алгоритмы, основанные на оценке, показали себя самыми стабильными. Они довольно достигают неплохого результата на относительно небольших объемах данных и после этого мало улучшаются в качестве с увеличением количества примеров., что делает их идеальными кандидатами для задач большой размерности. Тем не менее, хотя они и способны работать на задачах большей размерности, чем алгоритмы основанные на ограничениях, их применимость все равно ограничена размерами сетей, и в ряде реальных задач их использование невозможно.

Резюмируя, использование байесовских сетей благоразумно только на задачах небольшой размерности и только если необходима интерпретабельность результатов, иначе стоит использовать другие методы машинного обучения, например метод опорных векторов или случайный лес. На разреженных задачах малой размерности, где доступно большое количество тренировочных данных, стоит использовать алгоритмы, основанные на ограничениях, иначе основанные на оценке.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения работы был проведен анализ направленных вероятностных графовых моделей, их применимости на практических задачах. После этого был проанализирован общий подход к обучению параметров и конкретные алгоритмы для таких представителей НГВМ как байесовские сети и скрытые марковские модели. Наибольшее внимание было уделено структурному обучению НГВМ на примере байесовских сетей: были подробно проанализированы разные подходы, их сильные и слабые стороны. Наконец, были проведены эксперименты с данными подходами на практических задачах, подтверждены теоретические выводы о них.

Опишем главные выводы, которые были сделаны в результате проведения теоретического и практического анализа НГВМ.

Сфера применения НГВМ – это задачи диагностики и анализа, где явно присутствуют причинно-следственные связи. НВГМ при этом делятся на статические и динамические в зависимости от того, учитывается время или нет. Также НВГМ могут иметь наблюдаемыми все переменные или только их часть, и оставшиеся скрытые. Таким образом, НВГМ являются гибким инструментом, подходящим под множество задач. При этом способы обучения меняются незначительно – существуют общие подходы, которые можно оптимизировать в зависимости от конкретной задачи и типа модели.

Так, для обучения параметров используется метод максимального правдоподобия в случае, когда все переменные наблюдаемы, и EM-алгоритм, когда присутствуют скрытые. Эти методы применяются на практике и хорошо себя показывают, оптимизации как правило присутствуют в вычислении вероятностей из данных.

В случае же обучения структуры все сложнее. Выделяют три подхода и их гибриды, каждый из этих подходов имеет преимущества и недостатки, которые стоит учитывать при выборе алгоритма для конкретной задачи. Алгоритмы, основанные на ограничениях, хорошо работают на разреженных данных, но требуют много тренировочных данных. Алгоритмы, основанные на оценке, показывают хорошие результаты и на меньшем количестве данных. Алгоритмы, основанные на регрессии, теряют главное преимущество НГВМ – интерпертируемость, и вместо них стоит использовать принципиально другие модели, которые работают быстрее и эффективнее. Все подходы страдают от увеличения размерности задач, что делает их применение невозможным на большом количестве практических задач.

Итак, направленные вероятностные графовые модели являются эффективными инструментами для решения задач в различных областях. Для улучшения результатов можно пробовать разные подходы к их обучению. Нет универсального алгоритма, который работал бы одинаково хорошо на любой задаче – необходимо проводить работу по анализу данных, чтобы выбрать самый подходящий алгоритм и его гиперпараметры. Таким образом, дальнейшее развитие направленных вероятностных графовых моделей и интеграция с новыми подходами машинного обучения открывают новые возможности для решения более сложных и разнообразных задач в области искусственного интеллекта.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ


1. Николенко, С. И. Глубокое обучение / С. Николенко, А. Кадурин, Е. Архангельская. – СПб. : Питер, 2020. – 53 с.
2. Сукар, Л. Э. Вероятностные графовые модели. Принципы и приложения. / Л. Э. Сукар – М. : ДМК Пресс, 2021. – 383 с.
3. Федотов, С. Байесовский подход к оцениванию [Электронный ресурс] / С. Федотов. – Mode of access: https://education.yandex.ru/handbook/ml/article/bajesovskij-podhod-k-ocenivaniyu. – Date of access: 15.03.2025.
4. Ankar, A. Bayesian Model Sampling [Electronic resource] / A. Ankar. – Mode of access: https://pgmpy.org/infer/bn_sampling.html. – Date of access: 20.03.2025.
5. Bishop, C. M. Pattern Recognition and Machine Learning / C. M. Bishop. – USA : Springer, 2006 – P. 359-422.
6. Friedman, N. Learning Bayesian Network Structure from Massive Datasets: The "Sparse Candidate" Algorithm [Electronic resource] / N. Friedman, I. Nachman, D. Peer. – Mode of access: https://arxiv.org/pdf/1301.6696. – Date of access: 25.03.2025.
7. Heckerman, D. A Tutorial on Learning With Bayesian Networks [Electronic resource] / D. Heckerman. – Mode of access: https://arxiv.org/pdf/2002.00269. – Date of access: 28.03.2025.
8. Information Pursuit: A Bayesian Framework for Sequential Scene Parsing [Electronic resource] / E. Jahangiri [et al.]. – Mode of access: https://arxiv.org/pdf/1701.02343. – Date of access: 10.02.2025.
9. Kuntz, J. Markov chains revisited [Electronic resource] / J. Kuntz. - Mode of access: https://arxiv.org/pdf/2001.02183. - Date of access: 04.05.2025.
10. Learning Hidden Markov Models Using Conditional Samples. [Electronic resource] / S. M. Kakade [et al.]. – Mode of access: https://arxiv.org/pdf/2302.14753. – Date of access: 03.04.2025.
11. Mulgrave, J. J. Regression-Based Bayesian Estimation and Structure Learning for Nonparanormal Graphical Models [Electronic resource] / J. J. Mulgrave, S. Ghosal ; Department of Statistics, North Carolina State University, North Carolina, USA. – Mode of access: https://arxiv.org/pdf/1812.04442. – Date of access: 12.03.2025.
12. Vijaykumar, B. Bayes and Naive-Bayes Classifier [Electronic resource] / B. Vijaykumar ; Rajiv Gandhi University of Knowledge Technologies Andhra Pradesh, India. – Mode of access: https://arxiv.org/pdf/1404.0933. – Date of access: 12.04.2025.

13. Zhou, Y. Structure Learning of Probabilistic Graphical Models: A Comprehensive Survey [Electronic resource] / Y. Zhou. – Mode of access: https://arxiv.org/pdf/1111.6925. – Date of access: 20.03.2025.

## ПРИЛОЖЕНИЕ А

# КОД ПРОГРАММЫ ДЕМОНСТРАЦИИ РАЗНЫХ ПОДХОДОВ К ОБУЧЕНИЮ НВГМ НА ПРИМЕРЕ БАЙЕСКОВСКИХ СЕТЕЙ

```
import pandas as pd
from pgmpy.utils import get_example_model
from sklearn.preprocessing import LabelEncoder
from pgmpy.estimators import PC, HillClimbSearch, BicScore, K2Score
from pgmpy.models import BayesianNetwork
from sklearn.linear_model import LogisticRegression
import time
import matplotlib.pyplot as plt
import seaborn as sns

N_SAMPLES = 5000
asia_model = get_example_model('pathfinder') #'asia', 'child', 'hailfinder'
asia_data = asia_model.simulate(n_samples=N_SAMPLES, seed=4)

def constraint_based_pc(data):
    """Constraint-based PC algorithm"""
    print('pc')
    start = time.time()
    est = PC(data)
    model = est.estimate(significance_level=0.05, max_cond_vars=3)
    return model, time.time() - start

def score_based_hc(data, score_method):
    """Score-based Hill Climb search"""
    print('score based')
    start = time.time()
    est = HillClimbSearch(data)
    model = est.estimate(scoring_method=score_method, max_indegree=3)
    return model, time.time() - start

def evaluate_performance(true_model, learned_model):
    true_edges = set(true_model.edges())
    learned_edges = set(learned_model.edges() if learned_model else [])

    tp = len(true_edges & learned_edges)
    fp = len(learned_edges - true_edges)
    fn = len(true_edges - learned_edges)

    precision = tp / (tp + fp) if (tp + fp) > 0 else 0
    recall = tp / (tp + fn) if (tp + fn) > 0 else 0
    f1 = 2 * (precision * recall) / (precision + recall) if (precision + recall) > 0 else 0

    return {
```

```
        'precision': precision,
        'recall': recall,
        'f1': f1,
        'tp': tp,
        'fp': fp,
        'fn': fn,
        'edge_error': fp + fn
    }

# Применяемые методы
methods = {
    'Lasso': lambda data: learn_with_lasso(data),
    'PC': lambda data: constraint_based_pc(data),
    'HC+BIC': lambda data: score_based_hc(data, BicScore(data)),
    'HC+K2': lambda data: score_based_hc(data, K2Score(data)),

}

results = []
for method_name, method in methods.items():
    try:
        learned_model, elapsed = method(asia_data)
        metrics = evaluate_performance(asia_model, learned_model)
        metrics.update({
            'method': method_name,
            'time': elapsed
        })
        results.append(metrics)
    except Exception as e:
        print(f"Error in {method_name}: {str(e)}")
        continue

results_df = pd.DataFrame(results)
plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)
sns.barplot(x='method', y='f1', data=results_df)
plt.xlabel('Метод')
plt.title(f'F1 Score, {N_SAMPLES} примеров')
plt.ylim(0, 1)

plt.subplot(1, 3, 2)
sns.barplot(x='method', y='time', data=results_df)
plt.xlabel('Метод')
plt.ylabel('Время')
plt.title(f'Время (секунды), {N_SAMPLES} примеров')
plt.show()
```

## ПРИЛОЖЕНИЕ Б

# КОД ПРОГРАММЫ ДЕМОНСТРАЦИИ РАЗНЫХ ПОДХОДОВ НА РАЗНЫХ ОБЪЕМАХ ДАННЫХ С ВОЗМОЖНОСТЬЮ ДОБАВЛЕНИЯ ШУМА

```
import numpy as np
import pandas as pd
from pgmpy.utils import get_example_model
from pgmpy.estimators import PC, HillClimbSearch, BicScore, K2Score
from pgmpy.models import BayesianNetwork
import matplotlib.pyplot as plt
import seaborn as sns
from tqdm import tqdm
from sklearn.linear_model import LogisticRegression
import logging
logging.getLogger("pgmpy").setLevel(logging.WARNING)

# Конфигурация эксперимента
ADD_NOISE = True
METHOD = 'PC'  # Выбор метода: 'PC', 'HC+BIC', 'HC+K2', 'Lasso'
SAMPLE_SIZES = [i for i in range(1000, 50000, 2000)] #[250, 500, 750, 1000, 1250, 1500,
1750, 2000, 3000, 4000, 5000, 10000, 30000, 50000, 75000, 100000]  # Объемы данных
для тестирования
# SAMPLE_SIZES.extend([i for i in range(50000, 100001, 5000)])
N_RUNS = 5  # Количество запусков для каждого объема данных
NOISE_LEVEL = 0.05

# Загрузка модели
asia_model = get_example_model('asia')

def add_noise(data, noise_level=0.01):
    """
    Добавляет шум переключением значений с вероятностью noise_level
    """
    noisy_data = data.copy()
    for col in noisy_data.columns:
        # Выбираем случайные индексы для изменения
        flip_idx = np.random.choice([False, True], size=len(data),
                                    p=[1-noise_level, noise_level])
        # Меняем 'yes' на 'no' и наоборот для выбранных индексов
        noisy_data.loc[flip_idx, col] = noisy_data.loc[flip_idx, col].apply(
            lambda x: 'no' if x == 'yes' else 'yes'
        )
    return noisy_data

def learn_with_lasso(data):
```

```
    """Regression-based approach with categorical data support"""
    edges = []
    variables = data.columns

    # One-hot кодирование
    data = data.apply(lambda x: x.map({'yes': 1, 'no': 0}))

    for target in variables:
        predictors = [var for var in variables if var != target]
        X = data[predictors]
        y = data[target]

        model = LogisticRegression(penalty='l1', solver='liblinear', random_state=0,
max_iter=1000)
        model.fit(X, y)

        for i, coef in enumerate(model.coef_[0]):
            if abs(coef) > 0.1:
                edges.append((predictors[i], target))

    lasso_model = BayesianNetwork()
    for node in variables:
        lasso_model.add_node(node)
    for u, v in edges:
        if u in variables and v in variables:
            try:
                lasso_model.add_edge(u, v)
            except:
                continue

    if lasso_model.edges():
        data_str = data.apply(lambda x: x.map({1: 'yes', 0: 'no'}))
        lasso_model.fit(data_str)

    return lasso_model, 1

def evaluate_method(method, data, true_model):
    """Запускает метод и возвращает метрики"""
    try:
        if method == 'PC':
            est = PC(data)
            learned_model = est.estimate(significance_level=0.01)
        elif method == 'HC+BIC':
            est = HillClimbSearch(data)
            learned_model = est.estimate(scoring_method=BicScore(data))
        elif method == 'HC+K2':
            est = HillClimbSearch(data)
            learned_model = est.estimate(scoring_method=K2Score(data))
        elif method == 'Lasso':
            learned_model, _ = learn_with_lasso(data)
```

```
        else:
            raise ValueError("Unknown method")

        # Вычисление метрик
        true_edges = set(true_model.edges())
        learned_edges = set(learned_model.edges())
        tp = len(true_edges & learned_edges)
        fp = len(learned_edges - true_edges)
        fn = len(true_edges - learned_edges)

        precision = tp / (tp + fp) if (tp + fp) > 0 else 0
        recall = tp / (tp + fn) if (tp + fn) > 0 else 0
        f1 = 2 * (precision * recall) / (precision + recall) if (precision + recall) > 0 else 0

        return {
            'f1': f1,
            'precision': precision,
            'recall': recall,
            'edge_error': fp + fn,
            'n_edges': len(learned_edges)
        }
    except Exception as e:
        print(f"Error in {method}: {str(e)}")
        return None

# Запуск экспериментов
results = [{
            'f1': 0,
            'precision': 0,
            'recall': 0,
            'edge_error': 0,
            'n_edges': 0
        }]
results_noise = [{
            'f1': 0,
            'precision': 0,
            'recall': 0,
            'edge_error': 0,
            'n_edges': 0
        }]
for sample_size in tqdm(SAMPLE_SIZES, desc="Sample sizes"):
    for run in range(N_RUNS):
        # Генерация данных
        data = asia_model.simulate(n_samples=sample_size, seed=run)
        if ADD_NOISE:
            data_noise = add_noise(data, NOISE_LEVEL)
        # Оценка метода
        metrics = evaluate_method(METHOD, data, asia_model)
        if metrics:
            metrics.update({
```

```
            'sample_size': sample_size,
            'run': run
        })
        results.append(metrics)

    metrics_noise = evaluate_method(METHOD, data_noise, asia_model)
    if metrics_noise:
        metrics_noise.update({
            'sample_size': sample_size,
            'run': run
        })
        results_noise.append(metrics_noise)

# Создание DataFrame с результатами
results_df = pd.DataFrame(results)

# Построение графиков
plt.figure(figsize=(15, 10))

# График F1-score
plt.subplot(2, 2, 1)
sns.lineplot(x='sample_size', y='f1', data=results_df, errorbar='sd')
plt.title(f'F1-score ({METHOD})')
plt.xlabel('Количество тренировочных примеров')
plt.ylabel('F1-score')

if ADD_NOISE:
  results_df_noise = pd.DataFrame(results_noise)
  plt.subplot(2, 2, 2)
  sns.lineplot(x='sample_size', y='f1', data=results_df_noise, errorbar='sd')
  plt.title(f'F1-score ({METHOD}), {NOISE_LEVEL * 100}% шумных данных')
  plt.xlabel('Количество тренировочных примеров')
  plt.ylabel('F1-score')

plt.tight_layout()
plt.show()
```