Санкт-Петербургский государственный университет
Кафедра компьютерного моделирования и многопроцессорных систем

Толмачев Виктор Александрович

Выпускная квалификационная работа бакалавра

Определение скрытых демографических характеристик пользователя социальной сети

Направление 02.03.02
Фундаментальная информатика и информационные технологии

Научный руководитель,
доктор физ.-мат. наук,
профессор
Богданов А.В.

Санкт-Петербург
2019

Содержание

Введение

Стремительный рост социальных сетей привел к огромным ежедневно генерируемым пользователями потокам данных. И, как оказалось, информация, извлекаемая из большого количества свободно доступного публичного контента потенциально может выявить многие черты, предпочтения и мнения владельца профиля.

Подъем сервисов социальных сетей привел к растущему потенциалу для персонализации в компьютерных системах, начиная от интеллектуальных пользовательских интерфейсов или диалоговых агентов и систем рекомендаций до крупномасштабной аналитики здравоохранения, опроса в режиме реального времени, онлайн-рекламы и маркетинга. Исследователи начали добывать массивные объемы персонализированных и разнообразных данных, полученных в социальных сетях, с целью изучения демографических характеристик пользователей, таких как пол, возраст, политические предпочтения, пользовательские интересы, а также эмоции, психодемографический профиль и мнения, которые они выражают. В результате было реализовано несколько интеллектуальных аналитических услуг в социальных сетях [1, 2]. Эти службы принимают на вход профиль из социальной сети и выводят прогнозы о личности, эмоциях, настроениях и демографических характеристиках человека, владеющего профилем.

Вывод демографических характеристик из социальных сетей является полезным механизмом, позволяющим лучше понять свою аудиторию и облегчить взаимодействие с этой аудиторией. На сегодняшний день, общим подходом к определению демографических характеристик является использование методов машинного обучения с учителем, обученных по текстовым признакам. Однако, основным ограничением этого подхода является то, что он мало использует топологию сети. Поэтому для борьбы с ограничениями этого подхода предложены методы, базирующиеся на векторном представлении вершин графов и подходы, которые используют нейронные сети для изучения общей структуры социального графа.

Постановка задачи

Целью данной работы является изучение и применение методов машинного обучения на основе профиля пользователя, методов базирующихся на векторном представлении вершин социального графа, а также методов, использующих архитектуру глубокой нейронной сети социального графа для предсказания таких скрытых демографических атрибутов пользователя как пол и возраст

Для достижения поставленной цели предлагается решить следующие задачи:

1. Выбор источника данных
2. Предварительная обработка данных
3. Реализация методов машинного обучения на основе профиля пользователя
4. Реализация методов машинного обучения с использованием векторного представления вершин социального графа
5. Реализация глубокой нейронной сети для социального графа
6. Оценка и сравнение построенных моделей

Обзор литературы

Большинство работ, на сегодняшний день, используют данные из Twitter и Facebook и основываются на содержимом сообщений пользователей. Например, в работах [3–5] используется техника базирующая на мешке слов (bag of words - BOW). В этой технике предполагается, что текст пользователя представляется как неупорядоченный набор слов и в качестве признаков используются n-граммы и для каждой такой n-граммы подсчитывается абсолютная или относительная частота.

В работе [6] также представлен интересный подход, который основывается на эмоциональном тоне пользователя и эмоциональном контрасте пользовательской среды. Эксперименты показали, что добавления этих признаков к обычным признакам, извлеченным из текстов сообщений, дает неплохое улучшение качества модели.

Однако, проблема подходов, которые базируются только на текстовой информации пользователя в том, что они не учитывают взаимоотношений между пользователями. Поэтому следуя идее обучения представлению (representation learning) и успеху в применении векторных представлений слов [7], был предложен метод DeepWalk [8], который рассматривается как первый метод графовых векторных представлений. Похожие подходы, такие как node2vec [9], LINE [10] и SDNE [11], также достигли прорывов. Однако эти методы могут быть вычислительно дорогими и неоптимальными для больших графов. Примером, использования вышеупомянутых подходов, является метод, описанный в статье [12], который использует векторные представления пользователей с помощью алгоритма DeepWalk и применяет к этим представлениям модель линейной регрессии.

Для решения проблем, которым подвержены графовые векторные представления стали разрабатываться графовые нейронные сети (graph neural network - GNN) [13]. Они основываются на сверточных нейронных сетях (convolutional neural network - CNN) и векторных представлениях графов и предназначены для коллективного агрегирования информации из структуры графа. Примером алгоритма, который внес свой вклад в данную работу может послужить метод GraphSAGE, который описан в статье [14].

Глава 1. Сбор и подготовка данных

В данной главе будет рассмотрен способ сбора данных, краткая характеристика этих данных, а также предобработка полученных данных.

1.1. Сбор данных

Для сбора информации о пользователях в качестве социальной сети была выбрана социальная сеть ВКонтакте. Данная сеть является наиболее популярной в русскоязычном сегменте и ее главным преимуществом является удобный интерфейс прикладного программирования (application programming interface - API) для разработчиков. API - это интерфейс, предоставляющий набор готовых функций, процедур и классов, для удобного взаимодействия разработчиков с сервисом.

В качестве атрибутов пользователей были выбраны следующие признаки, извлекаемые из профилей:

1. Пол
2. Возраст
3. Город проживания
4. Страна проживания
5. Текущая деятельность
6. Политическая принадлежность
7. Количество детей
8. Семейное положение
9. Город расположения школы
10. Год окончания школы
11. Город расположения университета
12. Год окончания университета
13. Уровень образования

Стоит сделать замечание по поводу используемых признаков. Так, например, под городом проживания, страной проживания, городом расположения школы и университета здесь понимаются целочисленные идентификаторы. Возраст, количество детей, год окончания школы и университета принимают целочисленное значение. Все остальные атрибуты являются категориальными, а именно:

- Пол $\in$ {мужской, женский}
- Текущая деятельность $\in$ {работа, среднее образование, высшее образование}
- Политические предпочтения $\in$ {коммунистические, социалистические, умеренные, либеральные, консервативные, монархические, ультраконсервативные, индифферентные, либертарианские}
- Семейное положение $\in$ {женат/замужем, не женат/не замужем, неизвестно}
- Уровень образования $\in$ {среднее образование, высшее образование, неизвестно}

Для того, чтобы загрузить информацию о профилях пользователей, а также информацию о социальном графе, был реализован программный модуль, посредством языка python3.6, который использует API ВКонтакте [15]. Главным ограничением API ВКонтакте является то, что он позволяет делать не более 3 запросов в секунду.

Реализованный программный модуль выполняет следующие задачи:

1. Случайным образом генерируются пользовательские идентификаторы
2. Для полученного списка пользователей проверяется, указал ли он пол и возраст с помощью метода API «users.get»
3. Собирается информация о списке друзей, путем обращения к методу API «friends.get» и затем исходная выборка фильтруется на предмет того, чтобы у каждого пользователя было не менее 25 друзей
4. Для итогового списка пользователей собирается информация о профилях с помощью метода API «users.get»

В итоге были получены социальные связи для 1000 пользователей и демография для 125 000.

Все данные было принято хранить в формате CSV, поскольку использование данного формата удобнее для библиотек машинного обучения и эффективнее по памяти, чем в JSON.

1.2. Предобработка данных

Поскольку многие методы машинного обучения, используемые в данной работе чувствительны к масштабу, а также имеется много категориальных переменных, то перед применение алгоритмов машинного обучения и нейронных сетей, исходные данные, полученные из профилей пользователей, необходимо предобработать.

One Hot Encoding

One Hot Encoding – это способ представления категориальных переменных в виде бинарных векторов. Его суть заключается в том, чтобы сопоставить каждому уникальному значению категориальной переменной двоичный столбец.

Однако главным недостатком такого представления является то, что размер векторов растет с ростом числа уникальных значений в категориальной переменной.

Избавиться от вышеуказанного недостатка могут помочь методы уменьшения размерности.

На Рис. 1 продемонстрировано использование данного метода к категориальной переменной «Семейное положение». Другие категориальные признаки обрабатывались аналогично. Обработка была выполнена с помощью метода «get_dummies» библиотеки pandas [16]

Масштабирование

Масштабирование – это процесс изменения диапазона значений признака. Способ масштабирования, при котором каждый из признаков приводится к диапазону от 0 до 1 называется нормализацией. Другой способ масштабирования – стандартизация. Под стандартизацией подразумевается такая предобработка данных, после которой каждый признак имеет математическое ожидание равное 0 и дисперсию равную 1.

user_id	relation
6729089	неизвестно
4118708	не женат/не замужем
4242339	неизвестно
4356777	женат/замужем
5834258	не женат/не замужем

(а) Данные до преобразования

user_id	женат/замужем	не женат/не замужем	неизвестно
6729089	0	0	1
4118708	0	1	0
4242339	0	0	1
4356777	1	0	0
5834258	0	1	0

(б) Данные после преобразования

Рис. 1: Ohe Hot Encoding

В данной работе в качестве масштабирования был выбран процесс стандартизации, поскольку это один из самых употребляемых способов масштабирования признаков.

Стандартизация элемента выборки x происходит следующим образом:

$$x' = \frac{x - \mu}{\sigma}$$

где μ - среднее значение признака, σ - среднеквадратичное отклонение признака

1.3. Формирование выборок

Для решения поставленных задач были сформированы следующие выборки:

- Исходная – выборка с исходными атрибутами пользователей (без преобразования)
- Scaled – выборка, к которой ко всем признакам была применена стандартизация
- OHE – выборка, к которой к категориальным признакам был использова алгоритм One Hot Encoding
- OHE + Scaled – выборка, к которой сначала применили One Hot Encdoing, а затем проведена стандартизация

При этом в качестве тренировочной выборки было выбрано 80% данных и 20% соответственно для тестовой.

Для методов машинного обучения с учителем, которые будут описаны в следующей главе, существует проблема переобучения. Это явление при котором построенная модель сильно подстраивается под тренировочную выборку и хорошо классифицирует примеры из этого набора, но плохо классифицирует любые другие примеры из тестовых. Это происходит вследствие того, что в процессе обучения модель выявляет закономерности в тренировочном наборе, которые отсутствуют в генеральной совокупности.

Способы борьбы с данной проблемой зависят от выбранной модели. Однако, одним из способов, который применим ко всем методам, является кросс-валидация (перекрестная проверка). Его суть заключается в том, что имеющиеся данные разбиваются на k частей. После этого на $k-1$ частях производится обучение модели, а оставшаяся часть данных используется для тестирования. Данная процедура повторяется k раз, гарантирующая, что каждая из k частей данных будет использоваться для тестирования.

В данной работе для предотвращения переобучения применена кросс-валидация к тренировочным данных с разбиением на 5 частей.

Глава 2. Методы машинного обучения с учителем

В данной главе будут рассмотрены методы машинного обучения, используемые в работе, дана краткая характеристика задач, для которых эти методы используются, а также получены результаты по поставленным задачам.

2.1. Виды машинного обучения с учителем

Как правило, выделяют несколько видов машинного обучения с учителем. В данной работе используются:

1. Классификация. При решении задачи классификации предполагается, что имеется множество объектов разделенных некоторым образом на классы и, при этом, количество классов известно заранее.

2. Регрессия. В задаче регрессии требуется по заданному набору признаков объекта спрогнозировать целевую переменную, которая, как правило, может принимать любое вещественное значение.

2.2. Метрики качества

В данной работе для оценки качества определения возраста используется средняя абсолютная ошибка, которая определяется следующим образом:

$$MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n}$$

где y_i - точное значение, $\hat{y}_i$ - предсказанное значение

Для оценки качества классификации в задаче определения пола используются метрики Accuracy, Precision, Recall и F1, которая является гармоническим среднем Precision и Recall. Более формально эти величины определяются следующим образом:

$$Accuracy = \frac{T_p + T_n}{T_p + F_p + F_n + T_n}$$

$$Precision = \frac{T_p}{T_p + F_p}$$

$$Recall = \frac{T_p}{T_p + F_n}$$

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

где

T_p (true positives) – количество объектов которым модель присвоила первый класс и они на самом деле принадлежат первому классу;

F_p (false positives) – количество объектов, которые классификатор отнес к первому классу, но принадлежащие второму классу;

F_n (false negatives) – количество объектов, которые классификатор отнес ко второму классу, но принадлежащие первому классу;

T_n (true negatives) – количество объектов которым модель присвоила второй класс и они действительно принадлежат второму классу.

Recall показывает способность алгоритма обнаруживать данный класс, а Precision – демонстрирует способность отличать этот класс от других. Accuracy же является просто долей правильно классифицируемых объектов.

2.3. Линейная регрессия

Линейная регрессия [17] – это линейная модель, которая устанавливает связь между зависимой переменной $y \in \mathbb{R}$ и одной или несколькими независимыми переменными $x \in \mathbb{R}^d$. Формально данная модель определяется следующим образом:

$$f(x) = w_0 + \sum_{j=1}^{d} x^j w_j$$

$x = (x^1, x^2, ..., x^d)$ – входной вектор признаков
$w = (w_0, w_1, ..., w_d)$ – параметры модели

В задаче машинного обучения с учителем мы как правило имеем набор тренировочных данных $(x_1, y_1)...(x_l, y_l)$ с помощью которых мы оцениваем неизвестные параметры модели w. Самый популярный метод оценки неизвестных параметров модели линейной регрессии – это метод наименьших квадратов, суть которого состоит в том, чтобы подобрать неизвестные параметры модели w так, чтобы минимизировать функционал ошибки, ко-

торый представляет собой среднеквадратичную ошибку алгоритма:

$$Q(f, x) = \frac{1}{l} \sum_{i=1}^{l} (f(x_i) - y_i) \tag{1}$$

Минимизация данного функционала происходит с помощью оптимизационного подхода, а именно – метода градиентного спуска.

К преимуществам линейной регрессии можно отнести то, что она хорошо работает на больших объемах данных и на большом числе признаков. Однако существенным недостатком является то, что качество работы алгоритма может быть низким, если зависимость целевого значения от признаков нелинейная.

Для решения задачи определения возраста пользователя используется библиотека scikit learn [18], которая имеет в своем составе уже различные реализации метода линейной регрессии, такие как Ridge, Lasso и ElasticNet. Данные 3 метода являются модификацией исходной модели линейной регрессии, к которой добавляются соответственно L1, L2 и комбинация L1 и L2 регуляризации. Добавление регуляризации помогает при борьбе с переобучением.

Для автоматизации подбора параметров регуляризации из пакета scikit learn используется класс GridSearchCV с кросс-валидацией на 5 частей.

Полученные результаты определения возраста продемонстрированы в Таблице 1.

Выборка	Исходная	Scaled	OHE	OHE + Scaled
MAE	5.4882	5.4871	3.6179	3.6127
Регуляризация	Ridge	Ridge	Ridge	Ridge
Коэффициент регуляризации	6.251	323.746	0.001	0.001

Таблица 1: Результаты линейной регрессии для определения возраста

Как видно из Таблицы 1, лучше всего себя показала Ridge регрессия на масштабированной выборки с OHE преобразованием. Ridge регрессия заключается в том, что к оптимизированному функционалу (1) добавляет-

ся штраф – L2 регуляризатор, штрафующий модель за слишком большие веса, которые свидетельствуют о переобучении. В итоге задача оптимизации сводится к следующему виду:

$$Q(f, x) = \frac{1}{l} \sum_{i=1}^{l} (f(x_i) - y_i) + \lambda ||w||_2$$

2.4. Логистическая регрессия

Логистическая регрессия [19] – это метод классификации, который позволяет прогнозировать апостериорные вероятности принадлежности объектов к классам с помощью линейной разделяющей гиперплоскости.

Для того, чтобы оценивать апостериорные вероятности вводят так называемое отношение шансов, которое представляет из себя в случае бинарной классификации отношение вероятности принадлежности объекта x к классу 1 ($P(G = 1|x)$) к вероятности принадлежности объекта x к классу 0 ($P(G = 0|x)$). Логарифм данного отношения пытаются приблизить с помощью линейной функции

$$\log \frac{P(G = 1|x)}{1 - P(G = 1|x)} = w_0 + \sum_{j=1}^{d} x^j w_j \tag{2}$$

Делая некоторые преобразования в формуле (2) можно получить апостериорную оценку вероятности

$$P(G = 1|x) = \frac{1}{1 + e^{-z}}$$

где $z = w_0 + \sum_{j=1}^{d} x^j w_j$

Параметры модели $w = (w_0, w_1, ..., w_d)$ подбираются с помощью метода максимального правдоподобия.

Логистическая регрессия, как и любая другая линейная модель, обладает теми же преимуществами и недостатками, а именно: эффективная работа на больших объемах данных и низкое качество на нелинейных зависимостях.

В данной работе логистическая регрессия используется для оценки пола пользователя социальной сети. Данный метод находится в модуле linear_models пакета scikit learn. Результаты, полученные этим методом

к задаче определения пола можно наблюдать в Таблице 2.

Выборка	Исходная	Scaled	OHE	OHE + Scaled
Accuracy	0.69	0.685	0.71	0.7
Precision	0.66	0.5	0.77	0.636
Recall	0.0317	0.317	0.11	0.11
F1	0.061	0.059	0.194	0.189
Регуляризация	L1	L1	L1	L2
Коэффициент регуляризации	0.0336	0.234	0.234	3792.69

Таблица 2: Результаты логистической регрессии для определения пола

Как видно из таблицы данный метод имеет приемлемое значение по метрикам Accuracy и Precision, однако по метрикам Recall и F1 наблюдается низкое качество по всем выборкам.

2.5. Метод опорных векторов

Метод опорных векторов (support vector machine - SVM) – один из методов машинного обучения с учителем, который применяется для решения задач классификации [20] и регрессии [21].

Его идея заключается в том, что каждый объект данных представляется как вектор в d-мерном пространстве, которое нужно разделить на 2 класса (случай бинарной классификации) разделяющей гиперплоскостью. Однако, поскольку гиперплоскостей может быть множество, то для получения оптимальной гиперплоскости решается задача максимизации зазора между классами для более уверенной классификации. В итоге задача сводится к поиску такой гиперплоскости, чтобы расстояние от нее до ближайшей точки было максимальным. Стоит заметить, что данный метод также расширяется и на случай многоклассовой классификации.

Хотя этот метод относится к семейству линейных алгоритмов классификации, однако также можно получить и нелинейную классификацию, используя трюк ядра (kernel trick), который основывается на предположении о том, что существует пространство большей размерности, в котором выборка линейно разделима.

SVM для задачи регрессии использует те же принципы, что и SVM

для классификации, лишь с небольшими отличиями. Прежде всего, поскольку целевая переменная может принимать действительные числа, то становится очень трудно предсказать целевое значение имеющихся данных, которое может принимать бесконечное множество число значений. В случае регрессии, задается параметр $\epsilon > 0$, который определяет допустимое отклонение для результата.

К плюсам данного метода можно отнести то, что он находит максимальную ширину полосы разделения, вследствие чего производится уверенная классификация. Минусом же является большая чувствительность к шумам и отсутствие общего подхода к выбору ядра в случае линейной неразделимости классов.

Для решения задачи оценки возраста и пола пользователя используется реализация метода SVM, которая находится в пакете scikit learn. Гиперпараметры моделей подбираются с помощью кросс-валидации на 5 частей. Результаты применения данного метода, продемонстрированы в Таблице 3 и в Таблице 4.

Выборка	Исходная	Scaled	OHE	OHE + Scaled
MAE	4.84	3.53	3.98	5.52
Коэффициент регуляризации	2.0235	1000	10	10

Таблица 3: Результаты метода опорных векторов для определения возраста

Выборка	Исходная	Scaled	OHE	OHE + Scaled
Accuracy	0.685	0.7	0.685	0.71
Precision	0.38	0.8	0.57	0.73
Recall	0.05	0.06	0.06	0.13
F1	0.08	0.12	0.11	0.22
Коэффициент регуляризации	0.0001	1.757	1	0.5689

Таблица 4: Результаты метода опорных векторов для определения пола

Как видно из Таблицы 3, в задаче определения возраста SVM лучше всего сработал на отмасштабированных исходных признаках. Что касается задачи определения пола, то здесь по-прежнему наблюдается невысокое

качество метрик Recall и F1.

2.6. Метод k-ближайших соседей

Метод k-ближайших соседей (k-nearest neighbors - KNN) [22] – это метрический алгоритм, который может быть применим как к задаче классификации, так и к задаче регрессии. Его суть заключается в том, что объекту присваивается класс, который преобладает среди его соседей (задача классификации). Либо объекту присваивается значение, которое представляет собой среднее значение целевого атрибута среди соседей (задача регрессии).

Данный алгоритм бывает довольно эффективным если удачно подобрать гиперпараметры, такие как: число соседей, метрику близости и веса соседей. При этом в качестве метрики близости используется расстояние Минковского, которое вводится следующим образом:

$$\rho(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}$$

Если использовать взвешенный способ, то во внимание будет приниматься не только количество объектов определённых классов попавших в область, но а также их удалённость, что может повысить эффективность данного алгоритма.

Преимуществом метода k-ближайших соседей является то, что результаты данного алгоритма легко интерпретировать путём предъявления пользователю нескольких ближайших объектов. Недостатком же является нееэффективный расход памяти, вследствии необходимости хранения обучающей выборки целиком, а также необходимость сравнение классифицируемого объекта со всеми объектами выборки.

Подбор параметров, как и ранее, был выполнен с помощью кросс-валидации и класса GridSearchCV библиотеки scikit learn.

По полученным результатам из Таблицы 5 и Таблицы 6 можно сделать вывод, что данный метод не является оптимальным для обеих задач. Использование различных метрик и алгоритмов вычисления соседей не дало значимых улучшений.

Выборка	Scaled	OHE + Scaled
MAE	4.33	4.63
Число соседей	9	8
Веса соседей	Одинаковые	Одинаковые
Метрика	Минковского p = 1	Минковского p = 1
Алгоритм вычисления соседей	ball tree	brute force
Размер листа	30	–

Таблица 5: Результаты KNN для определения возраста

Выборка	Scaled	OHE + Scaled
Accuracy	0.7	0.71
Precision	0.5	0.86
Recall	0.11	0.1
F1	0.18	0.17
Число соседей	49	41
Веса соседей	Одинаковые	Одинаковые
Метрика	Минковского p = 3	Минковского p = 3
Алгоритм вычисления соседей	ball tree	ball tree
Размер листа	20	25

Таблица 6: Результаты KNN для определения пола

2.7. Случайный лес

Случайный лес [23] является одним из алгоритмов, который основывается на ансамбле решающих деревьев. Данный алгоритм базируется на бэггинге и методе случайных подпространств. Суть бэггинга заключается в том, что обучение базовых алгоритмов происходит на случайных подвыборках обучающей выборки. При этом базовые алгоримты получаются более независимыми при уменьшении размера случайной подвыборки. Что касается метода случайных подпространств, то тут выбирается случайное подмножество исходных признаков и очередной базовый алгоритм использует для обучения только эти признаки. При этом доля выбираемых

признаков является гиперпараметром модели.

В задаче классификации данный метод определяет целевой класс как наиболее распространенный предсказанный класс среди всех деревьев. Что касается регрессии, то целевая переменная определятся как среднее значение среди всех деревьев.

Данный метод имеет много достоинств, в числе которых эффективная обработка данных с большим числом классов, нечувствительность к масштабу признаков, возможность распараллеливания и масштабируемость, а также возможность выявление наиболее информативных признаков.

Однако, помимо всех заявленных ранее преимуществ алгоритм построения случайного леса имеет и ряд недостатков. Обучение очень глубоких деревьев требует, как правило, много вычислительных ресурсов, особенно, если мы имеем дело с большой выборкой или большим числом признаков. Но если ограничить глубину решающих деревьев в случайном лесе, то деревья решений не смогут улавливать сложные закономерности в данных. Другой проблемой можно считать то, что процесс построения деревьев является ненаправленным: каждое следующее дерево в композиции никак не зависит от предыдущих. Из-за этого для решения сложных задач необходимо огромное количество деревьев.

Выборка	Исходная
MAE	2.7
Число деревьев	800
Максимальная глубина дерева	90
Минимальное число объектов для разделения узла	5
Минимальное число объектов в листовом узле	4
Максимальное число признаков в методе случайных подпространств	4

Таблица 7: Результаты случайного леса для определения возраста

При решении поставленных задач использовался класс GridSearchCV пакета scikit learn для подборки следующих параметров: число деревьев (max_depth), максимальная глубина деревьев (max_depth), минимальное

число объектов для разделения узла (min_samples_split), минимальное число объектов в листовом узле (min_samples_leaf) и максимальное число признаков в дереве (max_features).

Результаты для задач определения возраста и пола представлены в Таблице 7 и в Таблице 8. Стоит отметить, что обучение проходило только на исходной выборке, поскольку методы основанные на деревья решений нечувствительны к масштабированию и к любым монотонным преобразованиям в данных.

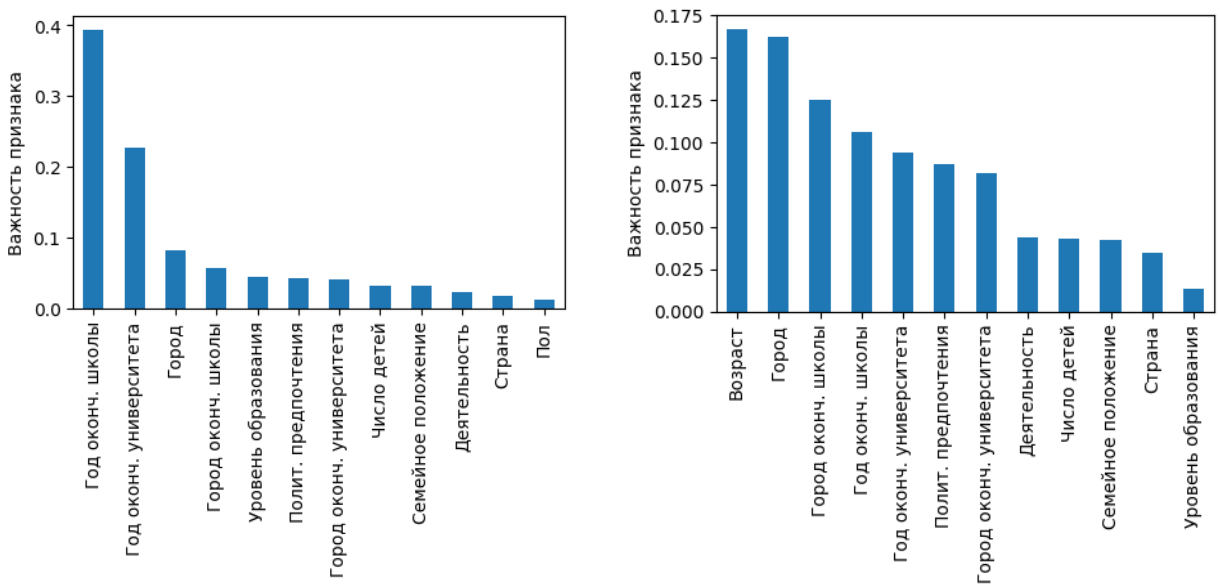
Выборка	Исходная
Accuracy	0.685
Precision	0.5
Recall	0.11
F1	0.18
Число деревьев	1600
Максимальная глубина дерева	10
Минимальное число объектов для разделения узла	5
Минимальное число объектов в листовом узле	1
Максимальное число признаков в методе случайных подпространств	4

Таблица 8: Результаты случайного леса для определения пола

При этом также были получены оценки важности признаков, которые представлены на Рис. 2. Из данных графиков можно сделать вывод, что наибольшее значение для случайного леса имеют год окончания школы и университета для определения возраста пользователя. Что касается определения пола пользователя, то большинство признаков имеют приблизительно равную важность для алгоритма.

2.8. Градиентный бустинг

Градиентный бустинг [24] – это техника машинного обучения для задач классификации и регрессии, которая строит модель в форме ансамбля слабых предсказывающих моделей, обычно деревьев решений. Главное от-



(а) Возраст (б) Пол

Рис. 2: Оценка важности признаков случайного леса в задачах определения возраста и пола

личие от случайного леса заключается в том, что в бустинге базовые модели строятся последовательно и каждый следующий алгоритм строится таким образом, чтобы исправить ошибки уже построенной композиции.

Данный метод имеет как преимущества, так и недостатки. Во-первых, главным плюсом является рассмотрение любого семейства базовых алгоритмов. А это дает широкие возможности для учета особенностей исходной задачи. Как правило, бустинг над решающими деревьями считается одним из эффективных вариантов использования данной модели. Помимо этого, данный алгоритм является довольно простым и имеет четкое математическое обоснование, что позволяет в каждой конкретной вариации бустинга сделать некоторые математические и алгоритмические оптимизации, которые могут ускорить работу алгоритма.

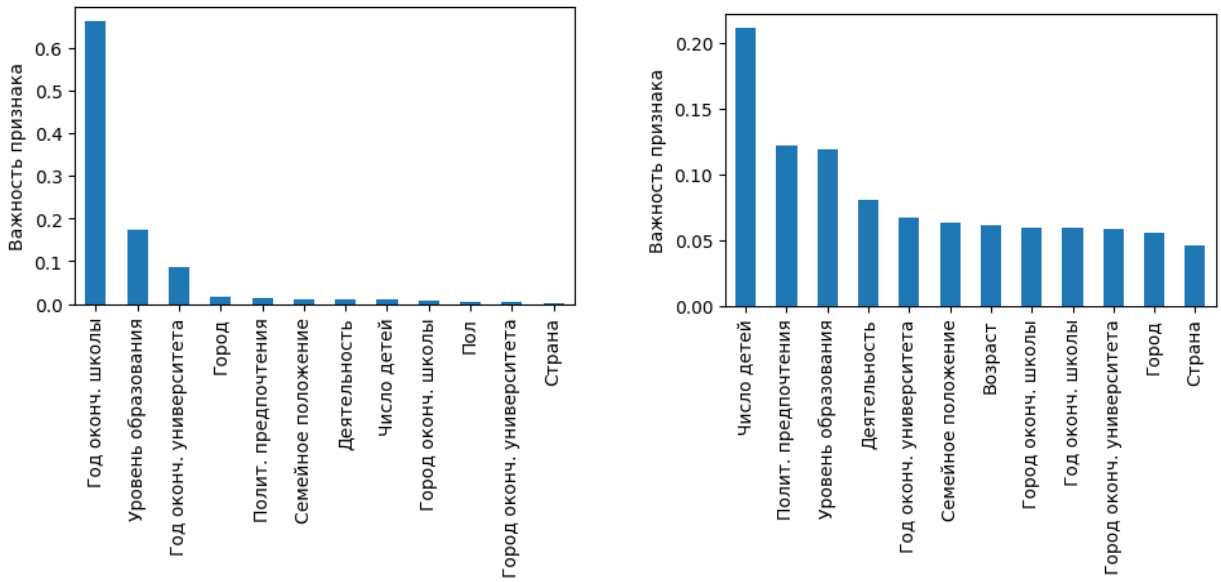
Выборка	Исходная
MAE	2.29
Число деревьев	1473
Максимальная глубина дерева	3
L1 регуляризация	138.94
L2 регуляризация	0.0167

Таблица 9: Результаты градиентного бустинга для определения возраста

Выборка	Исходная
Accuracy	0.715
Precision	0.636
Recall	0.222
F1	0.329
Число деревьев	4314
Максимальная глубина дерева	3
L1 регуляризация	0.022
L2 регуляризация	1.0

Таблица 10: Результаты градиентного бустинга для определения пола

Что касается недостатков, то как правило бустинг работает достаточно медленно, поскольку требуется построение сотен или даже тысяч базовых методов, которые входят в композицию. Также, данный метод имеет свойство очень сильно подстраиваться под данные, в том числе под выбросы в данных, что ведет в свою очередь к переобучению. И, наконец, результаты работы бустинга довольно сложно интерпретировать, особенно если в композицию входят десятки алгоритмов. Помимо этого, поскольку каждый последующий базовый алгоритм пытается исправить ошибки предыдущего, то гралиентный бустинг в целом сложно распараллелить.



(а) Возраст

(б) Пол

Рис. 3: Оценка важности признаков градиентного бустинга в задачах определения возраста и пола

Для решения поставленных задача, а именно определение возраста и пола пользователя используется библиотека XGBoost [25], которая является одной из самых популярных и эффективных реализаций градиентного бустинга на решающих деревьях. Подбор параметров выполнен с помощью кросс-валидации тренировочного множества. Полученные результаты можно наблюдать в Таблице 9 и в Таблице 10.

Графики оценки важности признаков представлены на Рис. 3. В отличие от случайного леса, помимо года окончания школы для градиентного бустинга имеет значение также уровень образования пользователя для определения возраста, а для определения пола – число детей, политические взгляды и уровень образования.

Глава 3. Графовые эмбеддинги

Под графовыми эмбеддингами понимается преобразование графов в вектор или набор векторов. Эмбеддинги должны охватывать топологию графа, отношение между вершинами и другую соответствующую информацию о графах, подграфах и вершинах.

В данной части будут рассмотрены наиболее значимые алгоритмы векторизации вершин социального графа, а также их преимущества и недостатки.

3.1. DeepWalk

DeepWalk [8] – это один из самых первых алгоритмов, который создает векторное представление узлов в графе с помощью случайных блужданий, а именно – идея заключается в переходе от структуры графа к некоторой последовательности вершин, которая сохранит основные свойства и информацию о графе.

Данный метод является простым обобщением концепции word2vec [26] на графы, поскольку вершины в данном алгоритме можно рассматривать как слова, а случайные блуждания - как предложения. А поскольку было установлено, что степени вершин, появляющихся в случайных блужданиях подчиняются такому же закону распределения (степенному), что и распределение слов в естественном языке, то в итоге при обучении к сгенерированным случайным блужданиям можно применять языковые модели, а именно – word2vec и SkipGram модели.

Решение, предложенное авторами оригинальной статьи [8] состоит из двух основных компонент: генерации случайной последовательности вершин (случайных блужданий) W_{v_i} для каждой вершины v_i и процедуры обновления скрытого представления.

Генерация случайного блуждания начинается с выбора начальной вершины v_1 среди всех вершин V графа G. На i-ом шаге выбирается новая вершина $v_{i+1} \in N(v_i)$ среди соседей предыдущей вершины v_i. Данная процедура выполняется, пока длина последовательности не достигнет заданного порога: $|W_{v_i}| = t$. При этом, данная процедура генерации случайных блужданий для каждой вершины выполняется γ раз.

Поскольку целью языковой модели является оценка правдоподобия

появления последовательности слов в корпусе, а случайные блуждания можно рассматривать как предложения в конкретном языке, то задача сводится к оценке правдоподобия вершины v_i $P(v_i|v_0, v_1, ..., v_{i-1})$, если до этого мы встретили $v_1, v_2, ..., v_{i-1}$ в случайном блуждании.

Так как на выходе мы должны получить векторное представление вершин графа, то вводится функция $\Phi : v \in V \mapsto R^{|V| \times d}$, которая сопоставляет каждой вершине v графа V его скрытое d-мерное представление. В итоге задача сводится к оценке вероятности:

$$P(v_i|\Phi(v_1), \Phi(v_2), ..., \Phi(v_{i-1}))$$

Исследования алгоритма word2vec показали, что намного эффективнее, во-первых, предсказывать не слово по контексту, а контекст по слову (в случае исходной задачи под словом понимается вершина графа, а под контекстом последовательность вершин в случайном блуждании). Во-вторых, можно не учитывать порядок слов в контексте, что в итоге сводит нас к задаче минимизации следующего вида:

$$\min_{\Phi} -\log P(\{v_{i-w}, ..., v_{i_1}, v_{i+1}, ..., v_{i+w}\}|\Phi(v_i))$$

После генерации очередной последовательности вершин требуемой длины t выполняется второй этап алгоритма – обновление модели с помощью алгоритма SkipGram.

SkipGram - это языковая модель, которая максимизирует вероятность совместного появления слов, в заданном окне w. Общая схема метода представлена в Алгоритме 1:

Алгоритм 1: SkipGram(Φ, W_{v_i}, w)

1 для всех $v_j \in W_{v_i}$ **выполнять**
2 | **для всех** $u_k \in W_{v_i}[j - v : j + w]$ **выполнять**
3 | | $J(\Phi) = -\log P(u_k|\Phi(v_j))$
4 | | $\Phi = \Phi - \alpha * \frac{\partial J}{\partial \Phi}$

Оценить вероятности $P(u_k|\Phi(v_j))$ можно используя либо логистический функционал, либо иерархический софтмакс.

Оценка с помощью логистического функционала выглядит следую-

щим образом:

$$P(u_k|\Phi(v_j)) = \frac{e^{\Phi(v_j)^T \cdot \Phi(u_k)}}{\sum_{v \in V} e^{\Phi(v)^T \cdot \Phi(u_k)}}$$

Что сводит задачу оптимизации к следующему виду:

$$\max_{\Phi} \sum_{u \in V} -\log Z_u + \sum_{v \in N_w(u)} \Phi(v)^T \cdot \Phi(u)$$

где

$$Z_u = \sum_{v \in V} e^{\Phi(v)^T \cdot \Phi(u)} \tag{3}$$

Однако проблема данного подхода заключается в том, что вычисление знаменателя требует значительных вычислительных ресурсов, поэтому для ускорения времени обучения используют иерархический софтмакс:

$$P(u_k|\Phi(v_j)) = \prod_{l=1}^{\lceil log|V| \rceil} P(b_l|\Phi(v_j))$$

Суть иерархического софтмакса заключается в том, что строится бинарное дерево, в листьях которого находятся вершины исходного графа. Поэтому вычисление требуемой вероятности эквивалентно вычислению вероятности пути в дереве от корня до вершины.

В вышележащей формуле путь до вершины u_k обозначается как (b_0, b_1, ..., $b_{\lceil log|V| \rceil}$), где $b_0 = root$, $b_{\lceil log|V| \rceil} = u_k$. А вероятности $P(b_l|\Phi(v_j))$ моделируются с помощью бинарных классификаторов на вершине b_{l-1}, которые указывают в какую сторону и с какой вероятностью нужно идти по бинарному дереву. Таким образом вычислительная сложность оценки вероятности снижается с $O(|V|)$ до $O(log|V|)$.

Однако, главным недостатком метода DeepWalk является то, что он требует больших вычислительных ресурсов, если поданный на вход граф является большим (более 100 000 узлов). С другой стороны, данный алгоритм можно распараллелить при генерации случайных блужданий, но это не дает значимых улучшений производительности. Другой недостаток – обучение векторных представлений происходит без взаимодействия с целевым значением исходной задачи.

Для решения задачи векторного представления вершин графа была

использована одноименная библиотека deepwalk [27]. При этом использование данного алгоритма проводилось со следующими параметрами: $\gamma = 30$ (число случайных блужданий), $t = 40$ (длина случайных блужданий), $w = 10$ (размер окна в алгоритме SkipGram), $d = 64$ (размерность полученных векторов). Результаты применения данного метода для задач определения возраста и пола можно наблюдать в Таблице 11 и в Таблице 12.

По полученным результатам можно заметить, что уже использование одних эмбеддингов в алгоритмах машинного обучения дает неплохие результаты, а при совместном использовании исходных признаков и эмбеддингов результаты становятся намного лучше. Так, например, использование градиентного бустинга в задаче определения возраста на исходных признаках и эмбеддингах дало внушительные результаты по средней абсолютной ошибке (MAE < 2).

В задаче определения возраста были также достигнуты результаты, превосходящие базовые алгоритмы на исходных данных.

	MAE	
Выборка	Только эмбеддинги	Признаки + эмбеддинги
Линейная регрессия	5.538	3.75
Случайный лес	4.36	2.6
Метод опорных векторов	3.9	4.28
k-ближайших соседей	4.87	4.42
Градиентный бустинг	4.68	1.37

Таблица 11: Результаты применения DeepWalk для определения возраста

Выборка	Только эмбеддинги					Признаки + эмбеддинги				
Метод	1	2	3	4	5	1	2	3	4	5
Accuracy	0.665	0.735	0.685	0.72	0.715	0.715	0.775	0.715	0.765	0.75
Precision	0.17	0.678	0.5	0.57	0.63	0.6	0.821	0.64	0.68	0.81
Recall	0.016	0.301	0.1	0.43	0.22	0.29	0.36	0.22	0.48	0.27
F1	0.03	0.417	0.16	0.49	0.33	0.39	0.51	0.33	0.56	0.4

Таблица 12: Результаты применения DeepWalk для определения пола (1 - логистическая регрессия, 2 - случайный лес, 3 - метод опорных векторов, 4 - k-ближайших соседей, 5 - градиентный бустинг)

3.2. Node2Vec

Поскольку метод DeepWalk выполняет случайные блуждания полностью случайным образом, то это означает, что эмбеддинги не сохраняют информации о локальной окрестности узла. Подход Node2vec [9] призван исправить это.

Алгоритм Node2Vec – является модификацией DeepWalk с небольшой разницей в генерации случайных блужданиях и в обучении этих блужданий с помощью SkipGram модели. В данном подходе вводятся параметры p и q, которые отвечают за баланс между обходом в глубину и в ширину графа. Параметр q определяет, насколько вероятно, что случайное блуждание будет отдаляться от исходной вершины, а параметр p отвечает насколько вероятно вернуться к предыдущему узлу.

Вероятность перехода от вершины u к вершине v задается следующим образом:

$$P(v|u) = \begin{cases} \frac{\pi_{uv}}{Z}, & \text{если } (u,v) \in E \\ 0, & \text{иначе} \end{cases}$$

где π_{uv} – это ненормализованная вероятность перехода из вершины u в вершину v, а Z – нормировочная константа. Эта ненормализованная вероятность задается с помощью марковского процесса второго порядка, суть которого заключается в том, что вероятность перехода зависит от того, откуда мы пришли.

Обозначая за w_{uv} вес ребра (u, v), а за d_{uv} – количество ребер в кратчайшем пути из вершины u в вершину v , которое может принимать значения только из множества $\{0,\ 1,\ 2\}$, то тогда искомую вероятность π_{uv} можно ввести как $\pi_{uv} = \alpha_{uv} \cdot w_{uv}$, где

$$\alpha_{uv} = \begin{cases} \frac{1}{p}, & \text{если } d_{uv} = 0 \\ 1, & \text{если } d_{uv} = 1 \\ \frac{1}{q}, & \text{если } d_{uv} = 2 \end{cases}$$

Если параметр $p > \max(1, q)$, то мы с меньшей вероятностью выбе-

рем уже посещенный узел. С другой стороны, если $p < min(1, q)$, то сгенерированные случайные блуждания будут близки к стартовой вершине. Аналогичны рассуждения для параметра q. Если $q > 1$, то случайное блуждание будет тяготиться к соседним вершинам и такие обходы будут близки к поиску в ширину. В противном случае, когда $q < 1$, случайные блуждания будут стремиться уйти от текущей вершины как можно дальше. Такое поведение отражает посик в глубину, который поощряет внешние исследования.

Другим отличием метода Node2Vec от DeepWalk является то, что вместо использования иерархического софтмакса для ускорения вычисления вероятности $P(u|\Phi(v))$ в SkipGram модели, он использует отрицательное сэмплирование. Идея заключается в том, чтобы уйти от подсчета суммы в формуле (3). Обозначая за E множество, элементами которого являются пары слово-контекст (узел-случайное блуждание), а за D – сгенерированные случайным образом пары из распределения отрицательного сэмплирования, задача оптимизации в итоге сводится к максимизации следующего выражения:

$$\sum_{(u,v)\in E} \log(\frac{1}{1+e^{-\Phi(u)^T\Phi(v)}}) + \sum_{(u,v)\in D} \log(\frac{1}{1+e^{\Phi(u)^T\Phi(v)}})$$

Путем оптимизирования данного функционала, модель учится отличать зависимости между вершинами от шума в виде случайно выбранных пар вершин.

	MAE	
Выборка	Только эмбеддинги	Признаки + эмбеддинги
Линейная регрессия	5.77	4.23
Случайный лес	6.21	2.8
Метод опорных векторов	5.56	5.59
K-ближайших соседей	4.7	4.7
Градиентный бустинг	5.82	2.7

Таблица 13: Результаты применения Node2Vec для определения возраста

Для нахождения векторных представлений вершин была выбрана одноименная библиотека node2vec [28], которая реализована на языке про-

граммирования python. В качестве параметров модели Node2Vec были взяты: $\gamma = 30$, $t = 40$, $w = 10$, $d = 64$, $p = 0.8$, $q = 0.5$. Результаты применения данного метода к задачам определения возраста и пола можно наблюдать в Таблице 13 и в Таблице 14.

По полученным результатам можно сделать вывод, что в целом совместное использование признаков и эмбеддингов улучшает результаты в сравнении с использованием одних лишь эмбеддингов, однако результаты применения DeepWalk оказались в целом лучше, чем у Node2Vec.

Выборка	Только эмбеддинги					Признаки + эмбеддинги				
Метод	1	2	3	4	5	1	2	3	4	5
Accuracy	0.64	0.615	0.71	0.735	0.715	0.665	0.59	0.71	0.765	0.73
Precision	0.36	0.375	1.0	0.61	0.64	0.4	0.38	0.75	0.79	0.71
Recall	0.19	0.33	0.08	0.43	0.22	0.13	0.48	0.1	0.35	0.24
F1	0.25	0.35	0.15	0.5	0.33	0.19	0.42	0.17	0.48	0.36

Таблица 14: Результаты применения Node2Vec для определения пола (1 - логистическая регрессия, 2 - случайный лес, 3 - метод опорных векторов, 4 - k-ближайших соседей, 5 - градиентный бустинг)

Глава 4. Графовые нейронные сети

Графовые нейронные сети (Graph Neural Network - GNN) – это методы глубокого обучения, основанные на графах. Как правило, они базируются на следующих фундаментальных аспектах: сверточных нейронных сетях (Convolutional Neural Network - CNN) и графовых эмбеддингах.

Сверточные нейронные сети могут работать только с евклидовыми данными, такими как изображения (двумерная сетка) и эту структуру данных можно рассматривать как граф. Ключевыми принципами сверточных нейронных сетей являются: локальная связь, разделяемые веса и использование многослойности. И эти принципы могут иметь большое значение при решении задач в области графов, поскольку графы являются наиболее типичной локально связной структурой, а общие веса снижают вычислительные затраты.

Главной проблемой для обобщения сверточных нейронных сетей на графы является то, что графы, как правило, в общем случае представляют собой нерегулярную неевклидову структуру данных (число узлов и связей между ними произвольно). Эту проблему и призвана решить графовая нейронная сеть, поскольку CNN не может должным образом обрабатывать ввод графа из-за того, что она обрабатывает узлы в определенном порядке. Однако, в графе нет естественного порядка, поэтому, чтобы полностью представить граф, мы должны пройти через все возможные порядки узлов в качестве входных данных модели, которая будет очень избыточна при вычислениях. Для решения этой проблемы GNN распространяются на каждом узле соответственно, игнорируя порядок ввода узлов. Другими словами, вывод GNN является инвариантным для порядка ввода узлов.

Другим важным аспектом является использование графовых эмбеддингов, которые учатся представлять узлы, ребра или подграфы графа в низкоразмерных векторах. Однако, использование методов графовых эмбеддингов имеет два серьезных недостатка. Во-первых, никакие параметры не разделяются между узлами при кодировании узлов в вектора, что приводит к неэффективности вычислений, поскольку это означает, что число параметров растет линейно с количеством узлов. Во-вторых, методы векторного представления, такие как DeepWalk и Node2Vec не имеют возможности обобщения, что означает, что они не могут иметь дело с динамиче-

скими графами или обобщаться на новые графы.

Поэтому на основе сверточных нейронных сетях и графовых эмбеддингов были введены графовые нейронные сети коллективно агрегирующие информацию из структуры графов, при это делая это вычислительно более эффективно.

4.1. Оригинальная концепция графовой нейронной сети

В задаче классификации узлов графа, каждый узел v, как правило, характеризуется вектором-признаков x_v и меткой класса t_v, к которому принадлежит узел. Имея частично размеченный граф G, целью графовой нейронной сети является изучение скрытого представления каждого узла v, т.е. $h_v \in \mathbb{R}^d$, зная информация о самом узле (признаки узла) и об окрестности узла. Полученное представление является d-мерным вектором и используется в дальнейшем для получения целевого значения o_v. Более формально:

$$h_v = f(x_v, x_{co[v]}, h_{ne[v]}, x_{ne[v]})$$

где x_v - признаки узла v, $x_{co[v]}$ - признаки ребер, соединенных с узлом v, $h_{ne[v]}$ - векторное представление узлов, находящихся в окрестности узла v, и $x_{ne[v]}$ - признаки узлов, находящихся в окрестности узла v. Функция f является функцией перехода, которая проецирует входные данные в d-мерное пространство.

Уравнение выше представляется как итеративный процесс, используя Банахову теорему о неподвижной точке:

$$H^{t+1} = F(H^t, X)$$

H и X обозначают конкатенацию всех h и x соответственно.

Целевое значение графовой нейронной сети вычисляется путем передачи векторного представления h_v и признаков x_v в выходную функцию g:

$$o_v = g(h_v, x_v)$$

При этом, f и g, могут быть интерпретированы как полносвязные ней-

ронные сети прямого распространения. И параметры этих функций могут быть найдены как решение задачи минимизации функции потерь:

$$loss = \sum_{i=1}^{p} (t_i - o_i)$$

которая может быть решена с помощью метода градиентного спуска.

Однако, оригинальный подход использования концепции графовой нейронной сети имеет существенные ограничения:

1. Она использует одни и те же параметры на итерации, в то время как большинство популярных нейронных сетей используют разные параметры в разных слоях.

2. Она не может обрабатывать информацию о ребрах (например, разные типы ребра в графе могут указывать на разные отношения между узлами)

3. Выбор эффективной функции, которая будет преобразовывать узлы в векторные представления.

Поэтому для решения некоторых из этих проблем были предложены более улучшенные вариации исходной концепции графовой нейронной сети.

4.2. GraphSAGE

Большинство существующих подходов векторного представления узлов графа являются трансдуктивными, т.е. они подходят только для графов с фиксированной структурой. Однако при добавлении новых узлов в граф трансдуктивные методы приходится полностью переучивать, что является вычислительно затратным и неэффективным способом для больших графов.

Метод GraphSAGE [14] успешно решает вышеупомянутую проблему. Этот подход базируется на сверточных нейронных сетях, обобщая их от обработки простых решеток до произвольных графов.

Ключевое отличие от трансдуктивных подходов является то, что вместо того, чтобы обучать векторное представление для каждого узла, мы обучаем набор агрегирующих функций, которые учатся собирать информацию о признаках из окрестности узла. При этом, каждая агрегирую-

щая функция собирает информацию из окрестности, которая находится на определенном расстоянии от целевого узла.

Более формально, процесс получения векторного представления узлов представлен в Алгоритме 2:

Алгоритм 2: Алгоритм генерации эмбеддингов GraphSAGE

Вход: Граф $G(V, E)$; признаки узлов $x_v, \forall v \in V$; весовая матрица $W^k, \forall k \in \{1, 2, ..., K\}$; агрегирующие функции $AGGREGATE_k, \forall k \in \{1, 2, ..., K\}$; нелинейная функция активации σ

Выход: Векторное представление $z_v, \forall v \in V$

$h_v^0 = x_v, \forall v \in V$

цикл *k=1...K* **выполнять**

  для всех $v \in V$ **выполнять**

    $h_{N(v)}^k = AGGREGATE_k(h_u^{k-1}, \forall u \in N(v))$

    $h_v^k = \sigma(W^k \cdot CONCAT(h_v^{k-1}, h_{N(v)}^k))$;

  $h_v^k = \frac{h_v^k}{||h_v^k||_2}, \forall v \in V$

конец

$z_v = h_v^K, \forall v \in V$

Каждый агрегатор во внешнем цикле отвечает за обработку узлов, находящихся на глубине k от исходного узла и при этом обрабатываются векторные представления узлов, полученные на предыдущем шаге. После того, как агрегатор заканчивает свою работу мы получаем вектор $h_{N(v)}^k$, который конкатенируется с векторным представлением текущего обрабатываемого узла h_v^{k-1}, которое получено на предыдущем шаге. Полученный вектор проходит через полно-связный слой с нелинейной функцией активации σ, на выходе которой получается новое векторное представление h_v^k, которое используется на следующем шаге цикла. Стоит заметить, что на первой итерации агрегирующая функция $AGGREGATE_1$ в качестве эмбеддингов использует признаки узла x_v.

Важным моментом является выбор агрегирующей функции и авторы исходной статьи говорят, что такая функция должна обладать свойством симметрии, поскольку это гарантирует, что наша модель нейронной сети может быть обучена и применена к произвольно упорядоченному набору узлов. В качестве агрегирующих функций может выбрана одна из следующих трех функций:

1. Mean агрегатор. Суть данного агрегатора заключается в том, что берется поэлементное среднее векторов.

2. LSTM агрератор. Данный агрегатор использует архитектуру, которая используется в нейронных сетях с долгой краткосрочной памятью, но поскольку данная архитектура не является симметричной, т.к. она обрабатывает фиксированную последовательность в определенном порядке, то при использовании данного метода применяется случайная перестановка.

3. Pooling агрегатор. При таком подходе векторное представление каждого узла проходит через полно-связный слой нейронной сети и после этого преобразования применяется поэлементная операция max или mean пулинга.

4.3. Реализация графовой нейронной сети

Архитектура нейронной сети GraphSAGE, описанная раннее, была реализована посредством языка программирования Python 3.6, а также библиотек Keras [29] и StellarGraph [30]. Выбор данного языка обусловлен тем, что он широко распространен в области машинного обучения и имеет в наличии множество библиотек для реализации нейронных сетей.

В задачах предсказания возраста и пола пользователя архитектура нейронной сети состоит из 3 скрытых слоев и каждый слой преобразует поданный вектор признаков в 32-мерный эмбеддинг. При этом к каждому слою применен дропаут с вероятностью 0.5. Для решения задачи оптимизации выбран алгоритм RMSprop, а в качестве агрегатора - MeanAgregator.

В качестве выборки, которая подается на вход нейронной сети была выбрана выборка, к которой предварительно применены One Hot Encoding и стандартизация.

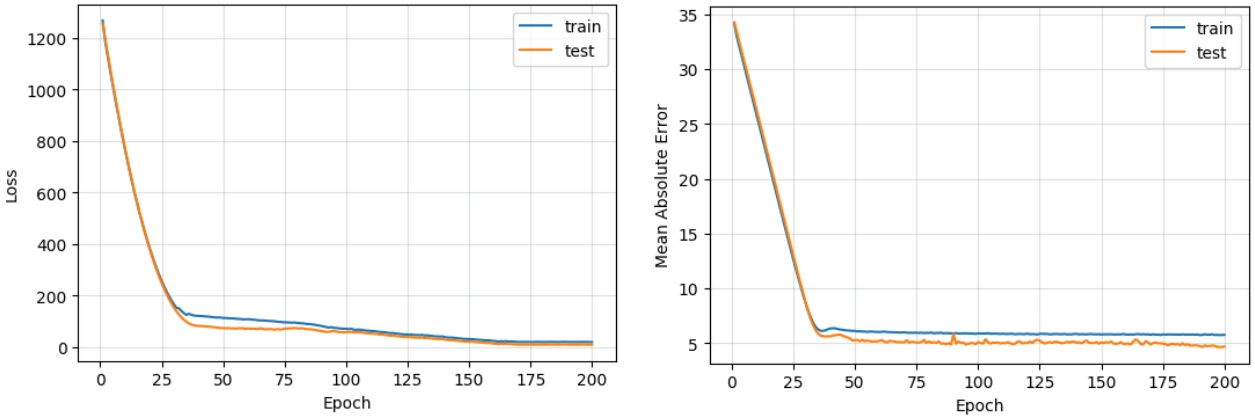
Полученные результаты можно наблюдать в Таблице 15 и в Таблице 16, а также процесс обучения отражен на Рис. 4 и Рис. 5. В задаче определения возраста нейронная сеть сработала хуже базовых алгоритмов

Метод	**MAE**
GraphSAGE	4.7

Таблица 15: Результаты работы метода GraphSAGE для определения возраста

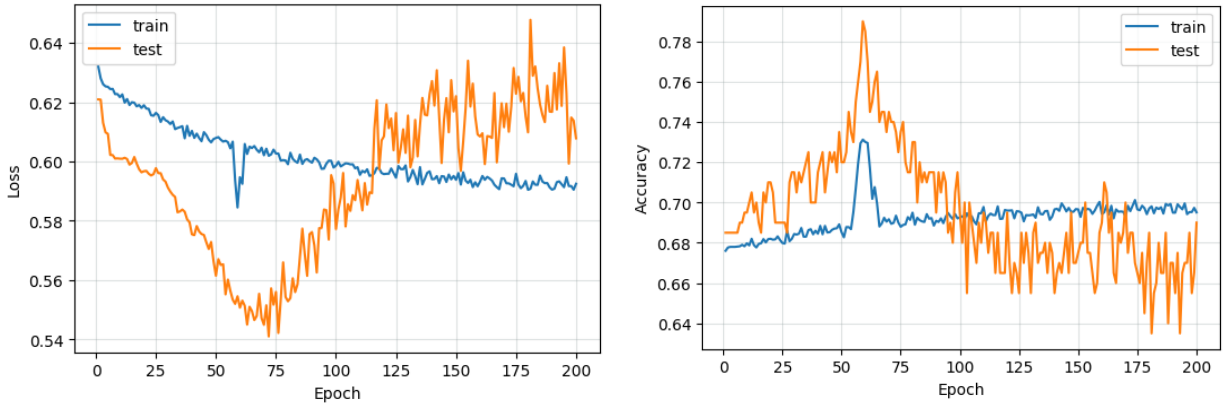
Метод	Accuracy	Precision	Recall	F1
GraphSAGE	0.79	0.72	0.92	0.81

Таблица 16: Результаты работы метода GraphSAGE для определения пола



(а) Loss (б) MAE

Рис. 4: Результаты алгоритма GraphSAGE в задаче определения возраста



(а) Loss (б) Accuracy

Рис. 5: Результаты алгоритма GraphSAGE в задаче определения пола

машинного обучения, однако при определении возраста можно наблюдать неплохой прирост значений по всем метрикам, в особенности F1 меры.

По полученным графикам обучения можно сделать вывод, что оптимальное число эпох обучения в задаче определения пола находится в диапазоне от 50 до 75, а именно – наилучшие результаты показала нейронная сеть при обучении на 59 эпохах. При этом использование большего числа эпох приводит к переобучению модели. Что касается задачи определения возраста, то после 30 эпохи идет незначительное улучшение качества алгоритма.

Стоит отметить, что обучение нейронной сети проходило как на CPU (на 8 ядерном процессоре Intel Scalable), так и на GPU (видеокарта NVIDIA

TESLA K80). На CPU время обучения нейронной сети для задач определения возраста и пола заняло порядка 20000 секунд, в то время как на GPU порядка 14000 секунд. В целом можно наблюдать незначительный прирост в скорости обучения.

Анализ результатов

В данная разделе представлена сравнительная характеристика методов на основе проведенных экспериментов. Полученные лучшие результаты по всем методам представлены в Таблице 17 и в Таблице 18.

Метрика	MAE		
Выборка	Только признаки	Только эмбеддинги	Признаки + эмбеддинги
Линейная регрессия	3.6127	5.538	3.75
Случайный лес	2.7	4.36	2.6
Метод опорных векторов	3.53	3.9	4.28
K-ближайших соседей	4.33	4.7	4.42
Градиентный бустинг	2.29	4.68	1.37
GraphSAGE	4.7	–	–

Таблица 17: Результаты всех методов для определения возраста

Выборка	Только признаки						Только эмбеддинги					Признаки + эмбеддинги				
Метод	1	2	3	4	5	6	1	2	3	4	5	1	2	3	4	5
Accuracy	0.71	0.685	0.71	0.71	0.715	**0.79**	0.64	0.735	0.71	**0.735**	0.715	0.715	**0.775**	0.715	0.765	0.75
Precision	**0.77**	0.5	0.73	0.86	0.63	0.72	0.36	0.68	**1.0**	0.61	0.63	0.6	**0.821**	0.64	0.68	0.81
Recall	0.11	0.11	0.13	0.1	0.22	**0.92**	0.19	0.3	0.08	**0.43**	0.22	0.29	0.36	0.22	**0.48**	0.27
F1	0.194	0.18	0.22	0.17	0.32	**0.81**	0.25	0.42	0.15	**0.5**	0.33	0.39	0.51	0.33	**0.56**	0.4

Таблица 18: Результаты всех методов для определения пола (1 - логистическая регрессия, 2 - случайный лес, 3 - метод опорных векторов, 4 - k-ближайших соседей, 5 - градиентный бустинг, 6 - GraphSAGE)

В задаче определения возраста лучше всего себя показали методы, базирующиеся на ансамблях решающих деревьев. На существенный прирост качества в данной задаче повлияло добавление к исходным признакам векторных представлений вершин графа. Это можно наблюдать по результатам градиентного бустинга. Однако, стоит отметить, что подбор параметров решающих деревьев довольно вычислительно затратный процесс и в данной работе это занимало порядка 8 часов для каждой из выборок. Также к недостатком можно отнести время работы алгоритмов DeepWalk и Node2Vec, поскольку на 8 ядерной машине генерация эмбеддингов занимала порядка 6-8 часов для графа из 125 000 вершин. Лучшие результаты

для всех методов продемонстрированы на Рис. 6.

Что касается результатов задачи определения пола, то можно сделать вывод, что использование нейронной сети и векторных представлений вершин уместно в данной задаче. Графовая нейронная сеть показала себя лучше всего по метрикам Accuracy, F1 и Recall. Также стоит отметить, что использование случайного леса и алгоритма k-ближайших соседей показали высокое качество на эмбеддингах и на совместном использовании эмбеддингов и исходных признаков.

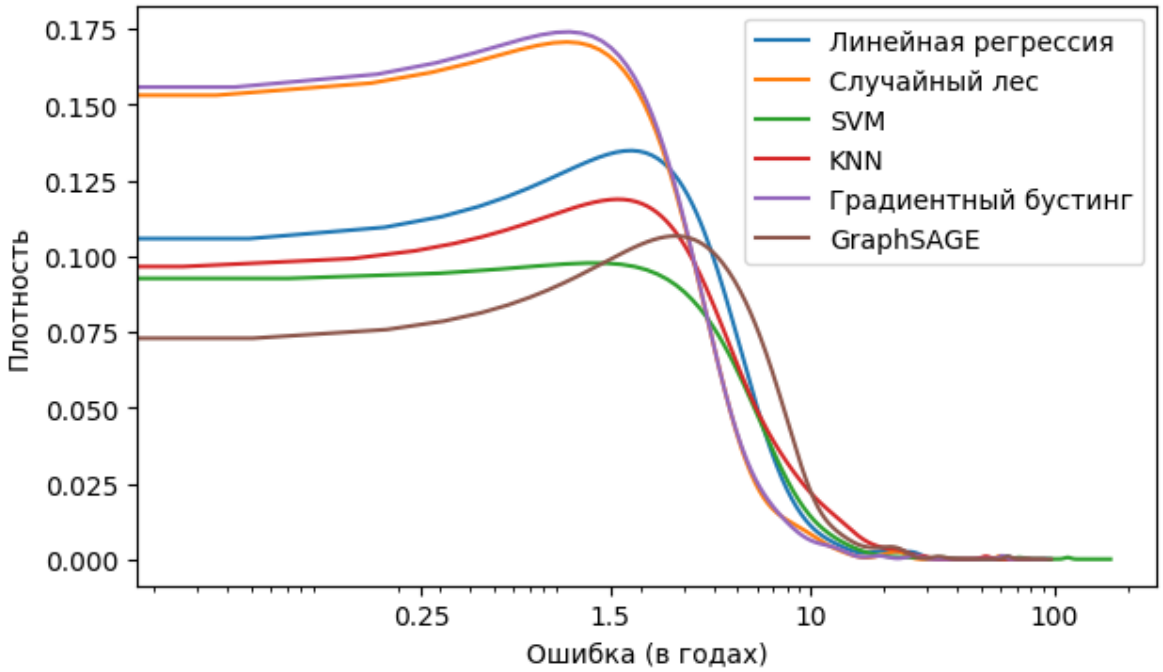


Рис. 6: Плотность распределения ошибки для всех методов в задаче определения возраста

Заключение

В рамках данной работы были рассмотрены и применены методы машинного обучения с учителем к задаче определения неизвестных демографических атрибутов пользователя. Эксперименты были проведены для таких характеристик как возраст и пол.

Также были рассмотрены и использованы алгоритмы для агрегирования информации из структуры социального графа, которая представляется в виде конечномерных векторов.

В данной работе было показано, что векторные представления вершин социального графа могут являться довольно значимыми признаками для определения демографических атрибутов и следовательно есть основания применять их для данной задачи, в том числе совместно с признаками, извлекаемыми из профилей пользователей. Однако проблемой использования данных признаков является то, что их вычисление, как правило, вычислительно затратно, поэтому при изменении структуры социального графа переобучение модели занимает продолжительное время.

С вышеупомянутой проблемой прекрасно справляется графовая нейронная сеть, которая не стремится обучать векторное представления для каждого узла, а напротив обучает набор агрегирующих функций, которые способны преобразовать исходные данные узла в векторное представление.

В качестве продолжения данной работы можно рассмотреть применение текстовой информации из постов пользователей в дополнении к информации, получаемой из структуры социального графа. Также можно сформулировать и решить задачу одновременного определения нескольких атрибутов пользователей, поскольку между разными демографическими атрибутами часто существуют неявные связи, которые могут улучшить качество определения скрытого атрибута.

Список литературы

[1] ApplyMagicSauce URL: https://applymagicsauce.com/demo (дата обращения: 11.04.2019).

[2] PersonalityInsights URL: https://personality-insights-demo.ng.bluemix.net (дата обращения: 11.04.2019).

[3] Benton A., Arora R., Dredze M. Learning multiview embeddings of twitter users //Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). – 2016. – Т. 2. – С. 14-19.

[4] Sap M. et al. Developing age and gender predictive lexica over social media //Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). – 2014. – С. 1146-1151.

[5] Jaika K., Guntuku S. C., Ungar L. H. Facebook vs. twitter: Cross-platform differences in self-disclosure and trait prediction. – 2018.

[6] Volkova S., Bachrach Y. Inferring perceived demographics from user emotional tone and user-environment emotional contrast //Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). – 2016. – Т. 1. – С. 1567-1578.

[7] Mikolov T. et al. Efficient estimation of word representations in vector space //arXiv preprint arXiv:1301.3781. – 2013.

[8] Perozzi B., Al-Rfou R., Skiena S. Deepwalk: Online learning of social representations //Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. – ACM, 2014. – С. 701-710.

[9] Grover A., Leskovec J. node2vec: Scalable feature learning for networks //Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. – ACM, 2016. – С. 855-864.

[10] Tang J. et al. Line: Large-scale information network embedding //Proceedings of the 24th international conference on world wide web. – International World Wide Web Conferences Steering Committee, 2015. – С. 1067-1077.

[11] Wang D., Cui P., Zhu W. Structural deep network embedding //Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. – ACM, 2016. – С. 1225-1234.

[12] Perozzi B., Skiena S. Exact age prediction in social networks //Proceedings of the 24th International Conference on World Wide Web. – ACM, 2015. – С. 91-92.

[13] Zhou J. et al. Graph neural networks: A review of methods and applications //arXiv preprint arXiv:1812.08434. – 2018.

[14] Hamilton W., Ying Z., Leskovec J. Inductive representation learning on large graphs //Advances in Neural Information Processing Systems. – 2017. – С. 1024-1034.

[15] API ВКонтакте URL: https://vk.com/dev (дата обращения: 11.04.2019).

[16] Pandas URL: https://pandas.pydata.org/ (дата обращения: 11.04.2019).

[17] Yan X., Su X. Linear regression analysis: theory and computing. – World Scientific, 2009.

[18] scikit-learn URL: https://scikit-learn.org/ (дата обращения: 11.04.2019).

[19] Hastie T., Tibshirani R., Friedman J. The elements of statistical learning: data mining, inference, and prediction, Springer Series in Statistics. – 2009.

[20] Cortes C., Vapnik V. Support-vector networks //Machine learning. – 1995. – Т. 20. – №. 3. – С. 273-297.

[21] Smola A. J., Schölkopf B. A tutorial on support vector regression //Statistics and computing. – 2004. – Т. 14. – №. 3. – С. 199-222.

[22] Метод ближайших соседей URL: http://www.machinelearning.ru/wiki/index.php?title=Метод_ближайшего_соседа (дата обращения: 18.04.2019).

[23] Breiman L. Random forests //Machine learning. – 2001. – Т. 45. – №. 1. – С. 5-32.

[24] Friedman J. H. Greedy function approximation: a gradient boosting machine //Annals of statistics. – 2001. – С. 1189-1232.

[25] XGBoost URL: https://xgboost.readthedocs.io/en/latest (дата обращения: 18.04.2019).

[26] Mikolov T. et al. Efficient estimation of word representations in vector space //arXiv preprint arXiv:1301.3781. – 2013.

[27] deepwalk library URL: https://github.com/phanein/deepwalk (дата обращения: 18.04.2019).

[28] node2vec library URL: https://github.com/eliorc/node2vec (дата обращения: 18.04.2019).

[29] Keras URL: https://keras.io (дата обращения: 25.04.2019).

[30] SteallarGraph URL: https://www.stellargraph.io (дата обращения: 25.04.2019).