



Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра математических методов прогнозирования

Никишин Евгений Сергеевич

**Снижение размерности больших массивов
данных**

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:
д.ф.-м.н., профессор
А.Г. Дьяконов

Москва, 2017

Содержание

1	Введение	3
1.1	Популярные алгоритмы	3
1.2	Связь понижения размерности с поиском матричной аппроксимации	5
2	Стабильная матричная аппроксимация	6
3	Анализ стабильности	7
4	Модель	9
4.1	Предложенная архитектура	9
4.2	Выбор подмножеств Ω_k	11
4.3	Итоговый алгоритм	11
5	Эксперименты	13
6	Заключение	13

1 Введение

Снижение размерности данных является одной из главных задач в машинном обучении, на практике использующейся для ряда целей:

- получение информативного описания элементов выборки;
- визуализация данных;
- предобработка данных с целью уменьшения затрат по памяти и времени.

Однако из-за роста объёмов информации многие классические алгоритмы оказываются неприменимы даже для предобработки данных. Наиболее плодотворной идеей в таком случае является постановка задачи снижения размерности как задачи оптимизации с последующим использованием аппарата стохастической оптимизации.

Кроме того, если есть необходимость в обучении правила снижения размерности с последующим его применением к данным, не участвовавшим в обучении, возникает вопрос об обобщающей способности такого правила. Реальные данные зачастую оказываются зашумлёнными и разреженными, что приводит к плохому качеству работы методов на данных, не участвующих в обучении.

[2do: переход]

Постановка задачи выглядит следующим образом. Отметим сразу, что в дальнейшем мы несколько отойдём от неё. Дана матрица $R \in \mathbb{R}^{m \times n}$. Предполагается, что существует некоторое представление данных размерности k (где $k < n$), вложенное в исходное. Методы снижения размерности осуществляют преобразование выборки, получая новое представление $Y \in \mathbb{R}^{m \times k}$, как можно больше при этом сохраняя структуру исходных данных.

1.1 Популярные алгоритмы

РСА

Метод главных компонент (Principal Component Analysis) является классическим методом снижения размерности. Основным предположением метода является существование линейного многообразия, на котором или около которого лежат данные.

Алгоритм находит k ортогональных осей, вдоль которых данные имеют наибольшую дисперсию и осуществляет проекцию данных на них.

Нахождение таких осей тесно связано с нахождением сингулярного разложения матрицы:

$$R = U \cdot \Sigma \cdot V^T,$$

где $U \in \mathbb{R}^{m \times m}$ и $V \in \mathbb{R}^{n \times n}$ — ортогональные матрицы, $\Sigma \in \mathbb{R}^{m \times n}$ — диагональная. При этом столбцы матриц U и V являются собственными векторами матриц RR^T и $R^T R$, а числа на диагонали Σ — корни из собственных значений этих двух матриц (наборы собственных значений у RR^T и $R^T R$ совпадают), называемые сингулярными числами.

Если центрировать исходные данные, то матрица $R^T R$ будет ковариационной матрицей, собственные вектора и собственные значения которой соответствуют осям и дисперсиям вдоль них. Как было сказано выше, метод находит k осей, вдоль которых дисперсия наибольшая. Поэтому, для получения представления размерности k необходимо умножить исходную матрицу R на матрицу, полученную из V выбором k вектор-столбцов, соответствующих наибольшему сингулярным числам:

$$Y = R \cdot V_k, \quad V_k \in \mathbb{R}^{n \times k}$$

Если количество признаков меньше длины выборки (как и происходит обычно на практике), то сложность метода главных компонент составляет $O(mn^2)$. Однако существуют эффективные итерационные методы нахождения усечённого сингулярного разложения (т.е. сохраняющего лишь первые k максимальных сингулярных чисел и соответствующих им сингулярных векторов), сложность которых $O(kmn)$.

Несмотря на то, что метод был предложен Карлом Пирсоном в 1901 году, РСА остаётся до сих пор одним из самых популярных методов снижения размерности.

Автокодировщик

Автокодировщиком называется нейронная сеть, выучивающая приближение для тождественной функции. Накладывая различные ограничения на архитектуру, мы можем обнаружить структурные закономерности в данных. К примеру, такими ограничениями могут быть ограничение на число нейронов в скрытых слоях сети или ограничение разреженности на нейроны скрытого слоя. На Рис. 1 схематично изображена типичная архитектура сети.

Одним из главных достоинств автокодировщиков является возможность обучения стохастическими градиентными методами, итерация ко-

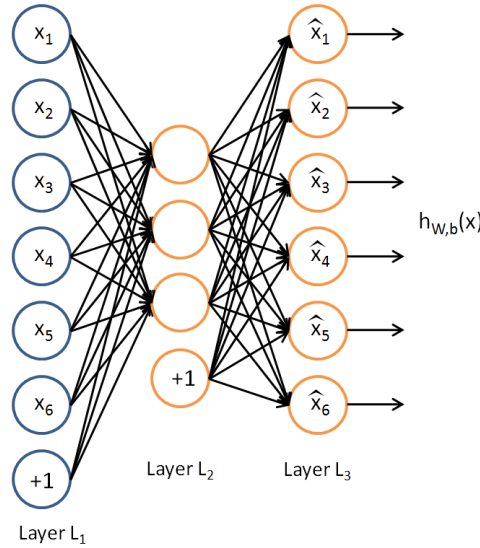


Рис. 1: Пример архитектуры автокодировщика

торого не зависит от одной из размерностей исходной матрицы, позволяя использовать для обучения большие массивы данных.

1.2 Связь понижения размерности с поиском матричной аппроксимации

Отметим, что и усечённое сингулярное разложение, и автокодировщик осуществляют аппроксимацию матрицы R . Для получения нового представления при использовании сингулярного разложения необходимо умножить исходную матрицу на один из факторов разложения, а при использовании автокодировщика — построчно применить к исходной матрице нейронную сеть и собрать представления на выходе кодирующей части.

В ряде задач машинного обучения матрица R не допускает представления объекты-признаки. Ярким примером является задача построения рекомендательной системы, в которой строкам матрицы R соответствуют пользователи, а столбцам — товары. На пересечении строки i и столбца j стоит рейтинг R_{ij} в случае, если пользователь i оценил товар j . В таком случае объектами являются и пользователи, и товары, а их признаковое описание в матрице R отсутствует.

Обычно [2do: Цитата] тогда пытаются обучать скрытые представления для пользователей и товаров. Можно сказать, что одновременно ре-

шаются две задачи снижения размерности в предположении, что объектами являются пользователи, а признаками — товары, и наоборот. То есть строятся две матрицы $U \in \mathbb{R}^{m \times k_1}$ и $V \in \mathbb{R}^{n \times k_2}$, на основе которых находится аппроксимация \hat{R} исходной матрицы R . Часто $k_1 = k_2$ и $\hat{R} = UV^T$.

Как было отмечено выше, реальные данные зачастую оказываются зашумлёнными и разреженными, что приводит к плохой обобщающей способности методов. Именно поиску аппроксимаций, не обладающих данным недостатком, и посвящена данная работа.

2 Стабильная матричная аппроксимация

Будем искать аппроксимацию матрицы в виде решения следующей задачи оптимизации:

$$\hat{R} = \arg \min_{X \in \text{Domain}(X)} D(R, X),$$

где $D(\cdot, \cdot)$ — некоторая функция потерь, а $\text{Domain}(X)$ — некоторый класс матриц, в котором ищется аппроксимация.

Например, если $D(R, X) = \|R - X\|_F^2 = \sum_{i,j} (R_{ij} - X_{ij})^2$ — квадрат нормы Фробениуса разности матриц, а класс $\text{Domain}(X)$ — множество всех матриц ранга k соответствующего размера, то решением задачи оптимизации \hat{R} является матрица, получаемая с помощью усечённого сингулярного разложения исходной матрицы R . [2do: об SVD в введении]

Другим примером может служить случай, когда $\text{Domain}(X) = \mathbb{R}^{m \times n}$ — множество всех действительных матриц соответствующего размера. В таком случае для большинства адекватных функций потерь аппроксимация \hat{R} будет совпадать с исходной матрицей R . Однако нас подобные решения интересовать не будут, так как обобщающая способность такой аппроксимации нулевая. Схожий эффект в обучении с учителем называется переобучением: если семейство, в котором мы ищем решения, допускает произвольно сложные функции, а функция потерь никак за эту сложность не штрафует, то алгоритм по сути лишь запоминает обучающую выборку, выдавая на новых объектах далёкие от истины ответы.

Таким образом, мы приходим, как минимум, к двум способам повышения обобщающей способности аппроксимации матрицы: штраф за сложность и ограничение класса решений. Рассмотренное выше ограничение на ранг матрицы представляет собой одновременно оба способа регуляризации. Это излишне упрощает получаемые аппроксимации, приводя к низкому качеству предсказания ненаблюдаемых ранее элементов матрицы.

[2do: Негладкий переход]

Введём теперь в некотором смысле «прямой» способ повышения обобщающей способности. Допустим, что для оценки качества мы используем k-fold кросс-валидацию. Если дисперсия эмпирического риска (среднего значения функции потерь по объектам) для разных подвыборок получается большой, то с высокой вероятностью на тестовых данных такой алгоритм продемонстрирует плохое качество. То есть, если рассматривать ошибку как случайную величину, то такая случайная величина должна иметь низкую дисперсию. Более формально ([2do: Цитата]):

Определение 1 Назовём ошибку аппроксимации δ -стабильной, если для заданного числа ϵ и для случайно выбранного подмножества элементов Ω выполнено следующее неравенство:

$$\mathbb{P}(|D(\hat{R}) - D_{\Omega}(\hat{R})| \leq \epsilon) \geq 1 - \delta,$$

где $D_{\Omega}(\hat{R})$ — ошибка на выбранном подмножестве. Например, если $D(\hat{R}) = \sqrt{\frac{1}{mn} \sum_{i=1}^m \sum_{j=1}^n (R_{ij} - \hat{R}_{ij})^2}$, то $D_{\Omega}(\hat{R}) = \sqrt{\frac{1}{|\Omega|} \sum_{(i,j) \in \Omega} (R_{ij} - \hat{R}_{ij})^2}$.

Далее будет неоднократно упоминаться «стабильная аппроксимация». Данные термины являются взаимозаменяемыми. Если выполнено свойство стабильности, то минимизация ошибки на тренировочном множестве объектов с высокой вероятностью влечёт низкую ошибку на тестовом множестве.

Существуют другие способы формализации понятия стабильности, которые можно найти, например, в [2do: Цитата]

Подобная нотация позволяет производить сравнение различных аппроксимаций. Для примера рассмотрим два подмножества элементов Ω_1 и Ω_2 . Если соответствующие им аппроксимации δ_1 - и δ_2 -стабильны, то можно говорить, что $D_{\Omega_1}(\hat{R})$ более стабильна, чем $D_{\Omega_2}(\hat{R})$, если $\delta_1 < \delta_2$. То есть, $D_{\Omega_1}(\hat{R})$ ближе к $D(\hat{R})$ с большей вероятностью, чем $D_{\Omega_2}(\hat{R})$. Это означает, что минимизация $D_{\Omega_1}(\hat{R})$ будет приводить к аппроксимации с лучшей обобщающей способностью, чем минимизация $D_{\Omega_2}(\hat{R})$. На основе подобного анализа и будет построено дальнейшее изложение.

3 Анализ стабильности

В обучении с учителем широко известной практикой является дообучение алгоритма на объектах, ошибка предсказания которых велика (если известно, что эти объекты не являются шумовыми / выбросами). Делать это можно разными способами, например, придавать больший

вес таким объектам в оптимизируемом функционале. В работе [2do: Цитата] это и предлагается делать. Если на текущем шаге у нас есть аппроксимация \hat{R} , то «прямым» способом повышения стабильности будет добавление к оптимизируемому функционалу слагаемых, дополнительно штрафующих за ошибку на выделенных элементах матрицы. Посредством такой операции ошибки на различных подмножествах элементов становятся более равномерными, приводя к более стабильным аппроксимациям в соответствии с определением выше.

Может показаться, что предложенный способ повышения стабильности является сугубо эвристическим. Однако справедливы следующие две теоремы, обосновывающие данную процедуру:

Теорема 1 Пусть Ω — множество всех наблюдаемых элементов матрицы R . Обозначим за ω множество таких элементов, что выполнено неравенство $|R_{ij} - \hat{R}_{ij}| < D_\Omega(\hat{R}) \forall (i, j) \in \omega$. Обозначим $\Omega' = \Omega \setminus \omega$. Тогда для любого $\epsilon > 0$ и любого набора весов λ_0, λ_1 ($0 < \lambda_0, \lambda_1 < 1$, $\lambda_0 + \lambda_1 = 1$) $\lambda_0 D_\Omega(\hat{R}) + (1 - \lambda_0) D_{\Omega'}(\hat{R})$ и $D_\Omega(\hat{R})$ являются δ_1 - и δ_2 -стабильными, причём $\delta_1 \leq \delta_2$.

Теорема 2 Пусть Ω — множество всех наблюдаемых элементов матрицы R . Обозначим за ω множество таких элементов, что выполнено неравенство $|R_{ij} - \hat{R}_{ij}| < D_\Omega(\hat{R}) \forall (i, j) \in \omega$. Разобьём это множество на K подмножеств $\omega_k: \bigsqcup_{k=1}^K \omega_k = \omega$, и обозначим $\Omega' = \Omega \setminus \omega$, $\Omega_k = \Omega \setminus \omega_k$. Тогда для любого $\epsilon > 0$ и любого набора весов $\lambda_0, \dots, \lambda_K$ ($0 < \lambda_k < 1 \forall k \in \overline{0, K}$, $\sum_{k=0}^K \lambda_k = 1$) $\lambda_0 D_\Omega(\hat{R}) + \sum_{k=1}^K \lambda_k D_{\Omega_k}(\hat{R})$ и $\lambda_0 D_\Omega(\hat{R}) + (1 - \lambda_0) D_{\Omega'}(\hat{R})$ являются δ_1 - и δ_2 -стабильными, причём $\delta_1 \leq \delta_2$.

Разберёмся со смыслом данных теорем. В множество ω входят элементы, ошибка на которых меньше, чем в среднем. Тогда, неформально говоря, в разнице множества всех наблюдаемых элементов Ω и множества «легкопредсказуемых» элементов ω лежат «труднопредсказуемые» элементы матрицы R , а при разбиении ω на подмножества ω_k в множествах $\Omega_k = \Omega \setminus \omega_k$ лежат «труднопредсказуемые» элементы R с некоторым количеством «легкопредсказуемых».

В первой теореме формулируется утверждение о том, что лучше минимизировать выпуклую комбинацию функционала на всём множестве Ω и функционала на множестве элементов Ω' , ошибка на которых в среднем выше, чем просто на множестве Ω . Вторая же теорема гласит, что ещё большей стабильности можно достичь, если совместно с функционалом на Ω минимизировать функционал сразу на нескольких множествах элементов, не являющимися лёгкими для предсказания.

Таким образом, в контексте поиска стабильных матричных аппроксимаций, подход с дообучением на объектах с высокой ошибкой предсказания обоснован не только эмпирически, но и теоретически.

4 Модель

До сих пор мы обсудили способ повышения стабильности аппроксимации, но упомянули, как именно её искать. В оригинальной статье предлагается находить аппроксимацию в виде произведения UV^T двух матриц $U \in \mathbb{R}^{m \times k}$ и $V \in \mathbb{R}^{n \times k}$, где $k < \min(m, n)$. Поэтому результатом будет матрица низкого ранга. Кроме того, к оптимизируемому функционалу прибавляются значения с коэффициентами норм этих матриц:

$$(\hat{U}, \hat{V}) := \arg \min_{U, V} \sum_{k=1}^K \lambda_k D_{\Omega_k}(UV^T) + \lambda_0 D_{\Omega}(UV^T) + \mu_1 \|U\|_F^2 + \mu \|V\|_F^2$$

То есть используются сразу три способа повышения стабильности: «прямой» способ, ограничение на класс матриц и штраф за сложность.

Поиск низкоранговой аппроксимации действительно приводит к стабильной ошибке, однако излишне ограничивает множество допустимых решений оптимизационной задачи, что означает недостаточно высокое качество аппроксимации.

Отметим, что в данной модели элементы R_{ij} находятся в виде линейной комбинации элементов матриц $U = U(R)$ и $V = V(R)$. Тогда естественным расширением будет поиск элементов R_{ij} в виде нелинейной функции $f = f(R)$. Одним из самых гибких подходов к аппроксимации нелинейных функций в настоящее время зарекомендовали себя нейронные сети. Параметризуя функцию с помощью нейронной сети $f = f_{\theta}$, мы можем применять для поиска аппроксимации аппарат стохастической оптимизации, который позволяет решать задачу для больших массивов данных.

В последующих разделах будет изложено описание архитектуры сети, способ поиска подмножеств Ω_k и итоговый алгоритм для поиска аппроксимации.

4.1 Предложенная архитектура

На рисунке 2 схематично изображена архитектура сети. Допустим, мы хотим получить элемент R_{ij} . Тогда на вход сети подаётся конкатенация векторов $R_{i\bullet}$ и $R_{\bullet j}$. Далее следует один или несколько слоёв, на

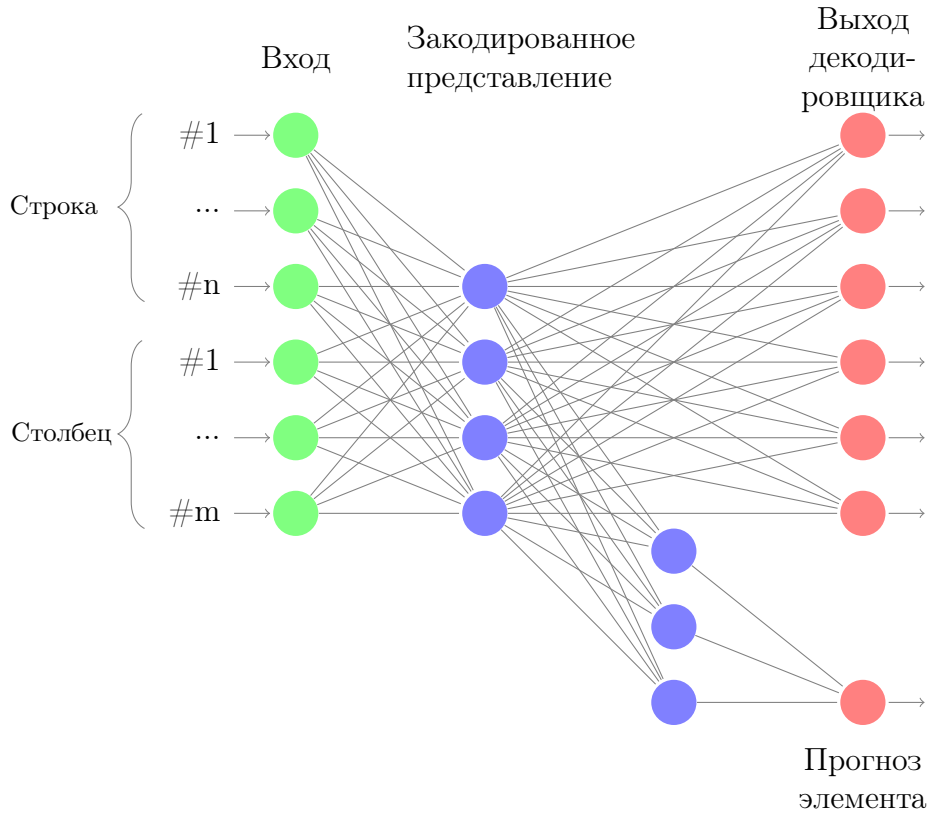


Рис. 2: Архитектура нейронной сети для поиска элементов аппроксимации

выходе которых получается скрытое совместное представление соответствующих строки и столбца матрицы. Затем архитектура разделяется на две ветви: первая соответствует декодировщику, который пытается приблизить тождественную функцию от входа, вторая же ветвь выдаёт предсказание элемента R_{ij} . Вклад каждой из ветвей в функционал регулируется весами.

Ключевым предположением при использовании такой сети является наличие структуры по строкам и столбцам в матрице R . Поясним на примере, что это означает. Как отмечалось, при решении задачи построения рекомендательной системы возникает матрица, строки которой соответствуют пользователям, столбцы — товарам. На пересечении строки i и столбца j стоит рейтинг R_{ij} в случае, если данный пользователь оценил данный товар. Такая матрица будет обладать структурой по строкам и по столбцам. Отметим, что поиск R в виде UV^T соответствует поиску R_{ij} в виде скалярного произведения векторов скрытых представлений

пользователей $U_{i\bullet}$ и товаров $V_{j\bullet}$.

[2do: о регуляризации]

Данная архитектура обладает рядом особенностей:

- она обобщает алгоритм поиска аппроксимации в виде произведения двух матриц низкого ранга;
- в работе [2do: Цитата] предложена модель, в которой либо по строкам, либо по столбцам обучается автокодировщик. Данная архитектура также обобщает эту модель;
- соответствующие \hat{R}_{ij} элементы встречаются в выходных слоях, вообще говоря, 3 раза: 2 раза в соответствующих позициях векторов $\hat{R}_{i\bullet}$ и $\hat{R}_{\bullet j}$ выхода декодировщика и третий раз в выходе сети непосредственно для прогноза R_{ij} . Итоговый элемент аппроксимации \hat{R}_{ij} можно полагать равным их выпуклой комбинации.

4.2 Выбор подмножеств Ω_k

Процедура дообучения на элементах, ошибка предсказания которых велика, предполагает подсчёт этой ошибки на основе имеющейся аппроксимации. Находить её можно с помощью стандартных методов, подбор которых стоит производить на основе априорной информации и функции потерь. Например, для $D(\hat{R}) = \sqrt{\frac{1}{mn} \sum_{i=1}^m \sum_{j=1}^n (R_{ij} - \hat{R}_{ij})^2}$ хорошим методом будет алгоритм RSVD, описанный в работе [2do: Цитата].

После подсчёта ошибки на каждом элементе происходит дополнительная рандомизация, в ходе которой решается, включить ли данный элемент во множество ω , на основе которого будут выбираться подмножества Ω_k . Пусть $p > 0.5$, тогда элементы, удовлетворяющие неравенству $|R_{i,j} - \hat{R}_{ij}| > D_{\Omega}(\hat{R})$, будем включать с вероятностью p , а не удовлетворяющие – с вероятностью $1 - p$.

Теперь всё готово для изложения алгоритма от начала до конца.

4.3 Итоговый алгоритм

Здесь за $[f_{\theta}(R_{i\bullet}, R_{\bullet j})]_{i=1, j=1}^{m,n}$ обозначена матрица, элементами которой являются предсказания элементов \hat{R}_{ij} , то есть $D_{\Omega_k}([f_{\theta}(R_{i\bullet}, R_{\bullet j})]_{i=1, j=1}^{m,n}) = \sqrt{\frac{1}{|\Omega_k|} \sum_{(i,j) \in \Omega_k} (R_{ij} - f_{\theta}(R_{i\bullet}, R_{\bullet j}))^2}$, а за $\|g_{\theta}(R_{i\bullet}, R_{\bullet j}) - [R_{i\bullet}^T, R_{\bullet j}^T]\|^2$ – штраф за разницу между выходом декодировщика и конкатенацией исходных векторов.

¹Код доступен на github.com/nikishin-evg/bachelors-diploma

Алгоритм 1 Нахождение стабильной матричной аппроксимации ¹

Вход: Исходная матрица R , вероятность выбора элемента p ,
начальная аппроксимация \bar{R} , число подмножеств K ,
набор коэффициентов $\{\lambda_k\}_{k=0}^K, \mu$;

Выход: Аппроксимация \hat{R} ;

- 1: Ω = множество пар индексов наблюдаемых элементов R ;
 - 2: $\omega = \emptyset$;
 - 3: **Для всех** $(i, j) \in \Omega$
 - 4: $\rho \sim U[0, 1]$;
 - 5: **Если** $(|R_{ij} - \bar{R}_{ij}| \geq D_\Omega(\bar{R}) \text{ и } \rho \leq p)$ **или**
 $(|R_{ij} - \bar{R}_{ij}| < D_\Omega(\bar{R}) \text{ и } \rho > p)$ **то**
 - 6: $\omega = \omega \cup (i, j)$;
 - 7: разбить ω на ω_k : $\bigsqcup_{k=1}^K \omega_k = \omega$
 - 8: положить $\Omega_k = \Omega \setminus \omega_k \ \forall k \in \overline{1, K}$;
 - 9: положить $\Omega_0 = \Omega$; {для удобства}
 - 10: $\sum_{k=0}^K \lambda_k (D_{\Omega_k}([f_\theta(R_{i\bullet}, R_{\bullet j})]_{i=1, j=1}^{m, n}) +$
 $\mu \sum_{(i, j) \in \Omega_k} \|g_\theta(R_{i\bullet}, R_{\bullet j}) - [R_{i\bullet}^T, R_{\bullet j}^T]\|^2) \rightarrow \min_\theta$
 - 11: **Для всех** (i, j)
 - 12: **Если** $(i, j) \in \Omega$ **то**
 - 13: $\hat{R}_{ij} = R_{ij}$;
 - 14: **иначе**
 - 15: $\hat{R}_{ij} = f_\theta(R_{i\bullet}, R_{\bullet j})$;
-

5 Эксперименты

6 Заключение