

Математические Методы Прогнозирования

Кафедра Интеллектуальных Систем

Отчёты о лабораторной работе №2

Оглавление

	Page
Глава 1. Muzalevskiy	5
1.1. Введение	5
1.2. Постановка задачи	7
1.3. Модели	7
1.4. Вычислительный эксперимент и анализ ошибки	9
Глава 2. Zharov	11
2.1. Введение	11
2.2. Постановка задачи	11
2.3. Модель	11
2.3.1. Canonical Polyadic Decomposition	11
2.3.2. Tucker Decomposition	12
2.3.3. Tensor Train Decomposition	12
2.4. Вычислительный эксперимент и анализ ошибки	13
Глава 3. kornilov	17
3.1. Введение	17
3.2. Математическая постановка задачи	17
3.3. Вычислительные эксперименты	20
3.4. Выводы	24
3.5. Литература	24
Глава 4. molozhavenco	25
4.1. Введение	25
4.2. Постановка задачи	25
4.3. Метод	25
4.4. Вычислительный эксперимент и анализ ошибки	27
4.4.1. Эксперимент 1	29
4.4.2. Эксперимент 2	31
4.5. Выводы	32
Глава 5. Solodyankin	33
5.1. Введение	33
5.2. Постановка задачи	33
5.3. Модель	33
5.3.1. Tucker Decomposition	33
5.3.2. HOSVD	34
5.3.3. Tensor Train Decomposition	35

5.4. Вычислительный эксперимент и анализ ошибки	35
Глава 6. Vladimirov	
6.1. Введение	36
6.2. Постановка задачи	36
6.2.1. HOSVD	36
6.2.2. ССМ	38
6.3. Вычислительный эксперимент	38
6.4. Литература	40

Глава 1

Muzalevskiy

1.1. Введение

Тензор - это многомерный массив. Следовательно, практически все геометрические структуры данных, с которыми мы работаем, являются тензорами. До размерности $d = 2$ эти тензоры имеют специфические названия: скаляр, вектор, матрица. Данные структуры представлены на рис.5.2

Декомпозиция называют процесс разбиения на составные элементы. По сути, это означает факторизацию тензора размерности d . Данный процесс заключается в нахождении оптимального разбиения общей структуры на элементы. В общем случае декомпозиция мотивируется необходимостью получить более простую совокупность составляющих элементов, которые могут наилучшим образом представить данную систему (или данные) [**decomposition**].

Прежде чем описать трехмерное разложение, опишем для начала принципы двумерного разложения (т.е. разложения матрицы). Подходы к двумерному разложению хорошо известны и включают анализ главных компонент (PCA), анализ независимых компонент (ICA), неотрицательную матричную факторизацию (NMF) и анализ разреженных компонент (SCA). Эти методы стали стандартными инструментами, например, для разделения источников (BSS), извлечения признаков или классификации [**matrixdecomp**].

$$X \approx M = \sum_{r=1}^R a_r \cdot b_r^T = a_r \circ b_r = A \cdot B^T X \in \mathbb{R}^{I \times J}, \mathbf{a} \in \mathbb{R}^I, \mathbf{b} \in \mathbb{R}^J$$

Где R - новая (уменьшенная) размерность наших данных, часто называемая рангом. Эта операция представляет собой простое суммирование внешних про-

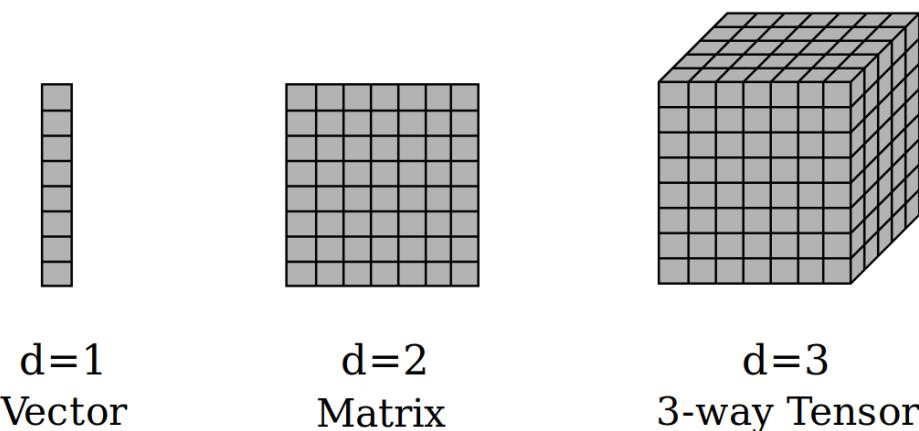


Рис. 1.1. "Структуры"

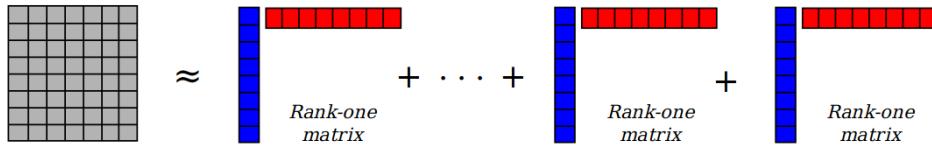


Рис. 1.2. "Факторизация"

изведений каждого столбца A и B , где индекс столбца задается r , как показано ниже на рис.1.6

Такая декомпозиция известна как факторизация. Приведенная выше формулировка страдает от проблемы, называемой проблемой вращения(rotation problem). То есть, мы можем вставить любую несингулярную матрицу вращения, Z , в приведенную выше формулировку, и в итоге получим то же самое приближение X .

$$X \approx M = \sum_{r=1}^R a_r \circ z_r^T \circ z_r^{-1} \circ b_r^T = A \cdot Z^T \cdot Z^{-1} \cdot B^T$$

Следовательно, если приведенная выше формулировка не ограничена, то она приводит к бесконечному множеству комбинаций A и B . Стандартные методы факторизации матриц в линейной алгебре, такие как QR-факторизация, разложение по собственным значениям (EVD) и разложение по сингулярным значениям (SVD), являются лишь частными случаями вышеприведенной формулировки и обязаны своей единственностью ограничениям, таким как треугольность и ортогональность.

Трехмерная декомпозиция - это просто расширение двумерной декомпозиции. Однако, хотя в двумерном случае для получения уникального решения на задачу должны быть наложены явные ограничения, высокая размерность тензорного формата имеет свои преимущества - это возможность получения компактных представлений, уникальность разложения, гибкость в выборе ограничений и общность компонентов, которые могут быть идентифицированы.

$$X \approx M = \sum_{r=1}^R a_r \circ b_r \circ c_r X \in \mathbb{R}^{I \times J \times K}, \mathbf{a} \in \mathbb{R}^I, \mathbf{b} \in \mathbb{R}^J, \mathbf{c} \in \mathbb{R}^K$$

В результате такого разложения мы получим три матрицы A с размерностью (IR), B с размерностью (JR) и C с размерностью (KR). Эта операция является простым суммированием внешнего произведения каждого столбца A , B и C , где индекс столбца задан r , как показано на рис.1.3

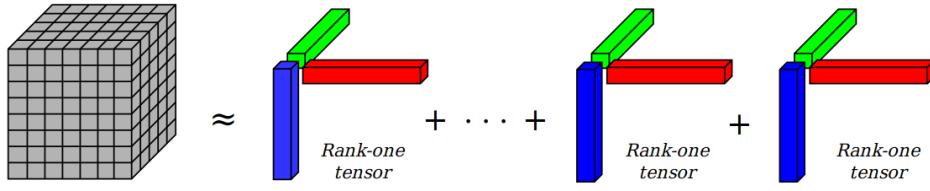


Рис. 1.3. "Разложение"

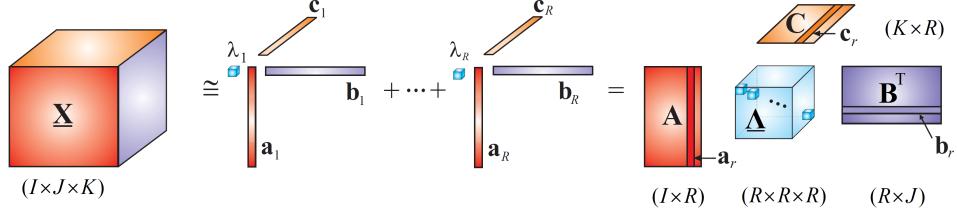


Рис. 1.4. "Canonical Polyadic Decomposition"

1.2. Постановка задачи

В данной работе, мы рассмотрим основные алгоритмы для тензорной декомпозиции, а также их реализации в библиотеке *hottbox* [**hottbox**]. В секции "Вычислительный эксперимент и анализ ошибки мы проведем сравнение ошибки данных методов, используя отношения норм Фробениуса [**matrixnorm**]

1.3. Модели

В рамках данной секции, мы рассмотрим две модели, представленные в пакете *hottbox* - это модель Canonical Polyadic Decomposition (*CPD*) [**CPD**] и модель Higher Order Singular Value Decomposition (*HOSVD*) [**HOSVD**].

Canonical Polyadic Decomposition (*CPD*) (также называемое *PARAFAC* или *CANDECOMP*) - это алгоритм, который факторизует тензор 3-го порядка $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$ в линейную комбинацию членов $\underline{\mathbf{X}}_r = \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$. Другими словами, тензор раскладывается следующим образом:

$$\begin{aligned}\underline{\mathbf{X}} &\simeq \sum_{r=1}^R \lambda_r \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \\ &= \underline{\Lambda} \times {}_1\mathbf{A} \times {}_2\mathbf{B} \times {}_3\mathbf{C} \\ &= [\underline{\Lambda}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\end{aligned}$$

В *hottbox* алгоритм *CPD* (с использованием метода Alternating Least Squares) реализован классом *CPD()*. Он выводит экземпляр класса *TensorCPD()* после каждого вызова метода *decompose()*. Этот метод принимает объект класса *Tensor*

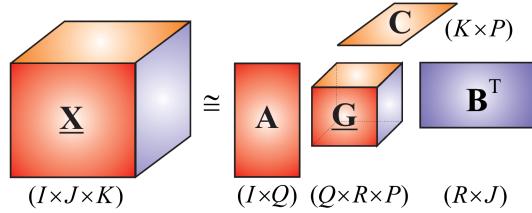


Рис. 1.5. "Tucker Decomposition"

и желаемое значение ранга Крускала, передаваемое в виде кортежа длины 1. Ранг Крускала передается в виде кортежа, чтобы сохранить одинаковый формат с другими алгоритмами тензорных разложений.

Перед тем, как перейти к рассмотрению алгоритма Higher Order Singular Value Decomposition (*HOSVD*), опишем принцип, на котором базируется данный алгоритм. Данный принцип носит название Разложение Такера (Tucker Decomposition).

Разложение Такера представляет заданный тензор $\underline{X} \in \mathbb{R}^{I \times J \times K}$ в виде тензора с плотным ядром \underline{G} и набором факторных матриц $\mathbf{A} \in \mathbb{R}^{I \times Q}$, $\mathbf{B} \in \mathbb{R}^{J \times R}$ и $\mathbf{C} \in \mathbb{R}^{K \times P}$. Другими словами, тензор \underline{X} может быть представлен в форме Такера следующим образом:

$$\begin{aligned}\underline{X} &\simeq \sum_{q=1}^Q \sum_{r=1}^R \sum_{p=1}^P g_{qrp} \mathbf{a}_q \circ \mathbf{b}_r \circ \mathbf{c}_p \\ &= \underline{G} \times {}_1\mathbf{A} \times {}_2\mathbf{B} \times {}_3\mathbf{C} \\ &= [\underline{G}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\end{aligned}$$

Higher Order Singular Value Decomposition (*HOSVD*) - частный случай разложения Такера, в котором все матрицы факторов ограничены ортогональностью. Они вычисляются как усеченная версия сингулярных матриц всех возможных поворотов тензора.

$$\begin{aligned}\mathbf{X}_{(1)} &= \mathbf{U}_1 \boldsymbol{\Sigma}_1 \mathbf{V}_1^T \quad \rightarrow \quad \mathbf{A} = \mathbf{U}_1 [1 : R_1] \\ \mathbf{X}_{(2)} &= \mathbf{U}_2 \boldsymbol{\Sigma}_2 \mathbf{V}_2^T \quad \rightarrow \quad \mathbf{B} = \mathbf{U}_2 [1 : R_2] \\ \mathbf{X}_{(3)} &= \mathbf{U}_3 \boldsymbol{\Sigma}_3 \mathbf{V}_3^T \quad \rightarrow \quad \mathbf{C} = \mathbf{U}_3 [1 : R_3]\end{aligned}$$

Причем, мы рассматриваем трехмерный тензор $\underline{X} \in \mathbb{R}^{I \times J \times K}$ как следующее представление (после применения разложения Такера):

$$\underline{X} = \underline{G} \times {}_1\mathbf{A} \times {}_2\mathbf{B} \times {}_3\mathbf{C}$$

Для тензора общего порядка - кортеж N -tuple (R_1, \dots, R_N) называется мультилинейным рангом и обеспечивает гибкость при сжатии и аппроксимации исходного тензора. После получения матриц коэффициентов, основной тензор вычисля-

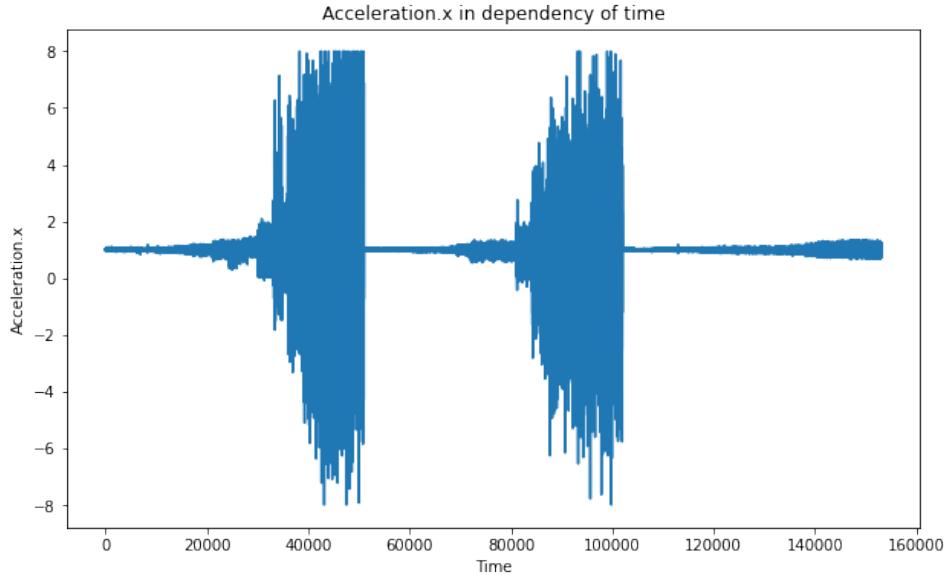


Рис. 1.6. "Акселерометр"

ется как

$$\underline{\mathbf{G}} = \mathbf{X} \times {}_1\mathbf{A}^T \times {}_2\mathbf{B}^T \times {}_3\mathbf{C}^T$$

1.4. Вычислительный эксперимент и анализ ошибки

Ссылка на код работы

Для задачи работы использовался датасет Accelerometer, который представляет собой зафиксированные значения колебаний вентилятора охладителя с грузами на лопастях.

Ошибка считалась несколькими методами: с помощью встроенного метода "residual rel error" который рассчитывает отношение норм Фробениуса двух тензоров (изначального и полученного в результате применения алгоритма), а также с применением ошибок *MSE* и *MAPE* (реализованы в *hottbox* функциями *mse()* и *mape()* соответственно).

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

$$MAPE = \frac{100}{n} \sum_{i=1}^n \frac{|y_i - \hat{y}_i|}{y_i}$$

Для оценки анализа работы двух алгоритмов применялось также время работы алгоритма декомпозиции. Данные ошибок и времени работы алгоритмов доступны в таблице:

	CPD	HOSVD
mse	0.995	0.999
mape	4.43	4.79
rel error	0.485	0.5
time	0.3	0.07

Таблица 1.1. Результаты работы алгоритмов

В данной работе мы видим, что алгоритм CPD показывает лучшие результаты в плане меньшей величины ошибок, однако проигрывает HOSVD по времени работы. Данное наблюдение можно использовать при работе с другими датасетами схожей природы (акселерометры), для того, чтобы получить алгоритм, который дает разумное сочетание качества и быстроты выполнения. Разумеется, остается большой простор для сравнения различных датасетов и применения алгоритмов для решения практических задач.

Глава 2

Zharov

2.1. Введение

В различных задачах машинного обучения ставится вопрос нахождения оптимальной сложности модели и поиска компромисса между точностью и сложностью модели. Так мы можем предобрабатывать исходные данные, уменьшая размерность признакового пространства и удаляя сильно скореллированные признаки. В данной работе исследуются различные методы сжатия снимков функциональной магнитно-резонансной томографии (*fMRI*), представленных в виде трехмерных тензоров. Необходимо исследовать способы получения малоранговых приближений данных тензоров.

2.2. Постановка задачи

Формально задача ставится следующим образом. Имеется исходное объемное изображение, представленное трехмерным тензором X . Рассматривается его приближение X^* и ошибка аппроксимации $\varepsilon = \frac{\|X - X^*\|_F}{\|X\|_F}$. Исследуется поведение ошибки при увеличении ранга аппроксимации тензора (ранг Крускала k для *CPD*, мультиранг k_1, k_2, k_3 для разложения Таксера, количество сверточных слоев для автоэнкодера). Достаточная точность приближения выбирается следующим образом: мы прекращаем увеличивать ранг, когда дисперсия ошибки перестает значительно изменяться на нескольких последних шагах алгоритма.

2.3. Модель

В работе рассматривались следующие модели: Canonical Polyadic Decomposition (*CPD*), Tucker Decomposition в двух вариантах *HOSVD* и *HOOI*, Tensor Train Decomposition и нейросетевой автоэнкодер. Реализация первых трех алгоритмов была взята из библиотеки *hottbox*. Ниже подробнее про алгоритмы.

2.3.1. Canonical Polyadic Decomposition

Canonical Polyadic Decomposition (*CPD*) это алгоритм факторизации трехмерного тензора $\underline{X} \in \mathbb{R}^{I \times J \times K}$ в виде комбинации $\underline{X}_r = \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$, компоненты которой являются тензорами ранга 1. Иначе говоря тензор \underline{X} раскладывается как,

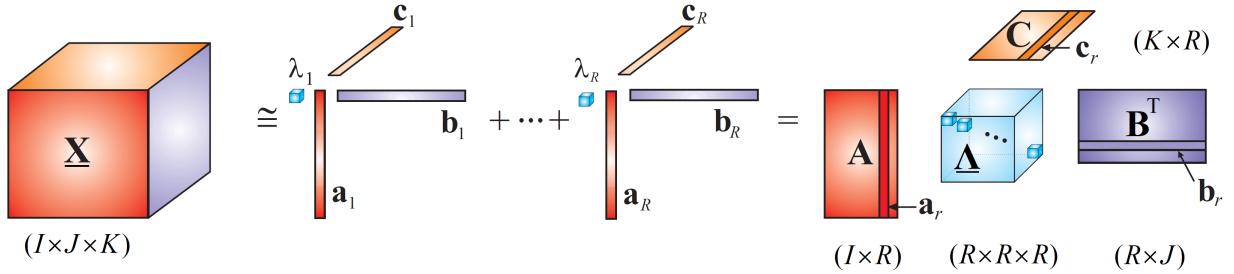


Рис. 2.1. Canonical Polyadic Decomposition

$$\underline{\mathbf{X}} \simeq \sum_{r=1}^R \lambda_r \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r, = \underline{\Lambda} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}, = [\underline{\Lambda}; \mathbf{A}, \mathbf{B}, \mathbf{C}]$$

,

где

$-\underline{\Lambda}$ это трехмерное тензорное ядро с значениями λ_r на позициях $\underline{\Lambda}[i, j, k]$, where $i = j = k$, и нулями в остальных местах,

$-\mathbf{A}, \mathbf{B}, \mathbf{C}$ это фактор-матрицы, полученные как конкатенация фактор-векторов, т.е. $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_R]$,

2.3.2. Tucker Decomposition

Tensor Train Decomposition

Разложение Такера заключается в представлении тензора $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$ в виде тензора-ядра, заполненного элементами \mathbf{G} и набора фактор-матриц $\mathbf{A} \in \mathbb{R}^{I \times Q}, \mathbf{B} \in \mathbb{R}^{J \times R}$ и $\mathbf{C} \in \mathbb{R}^K \times P$ Иначе говоря, тензор $\underline{\mathbf{X}}$ представляется в форме Такера следующим образом:

$$\begin{aligned} \underline{\mathbf{X}} &\simeq \sum_{q=1}^Q \sum_{r=1}^R \sum_{p=1}^P g_{qrp} \mathbf{a}_q \circ \mathbf{b}_r \circ \mathbf{c}_p \\ &= \mathbf{G} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} \\ &= [\mathbf{G}; \mathbf{A}, \mathbf{B}, \mathbf{C}] \end{aligned}$$

В пакете hottbox разложение Такера реализовано в двух видах: *HOSVD* и *HOOI*, о них речь пойдет ниже.

2.3.3. Tensor Train Decomposition

Tensor train decomposition представляет данный тензор в виде малосвязанных тензоров меньшего порядка и фактор-матриц. Другими словами, алгоритм пред-

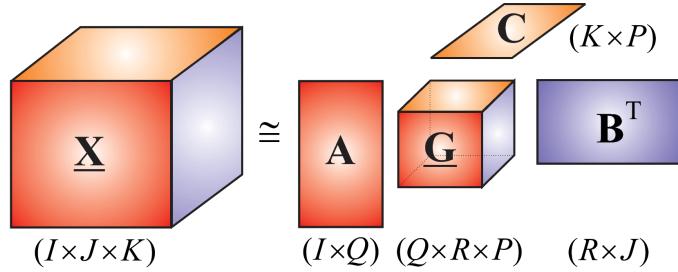


Рис. 2.2. Tucker Decomposition

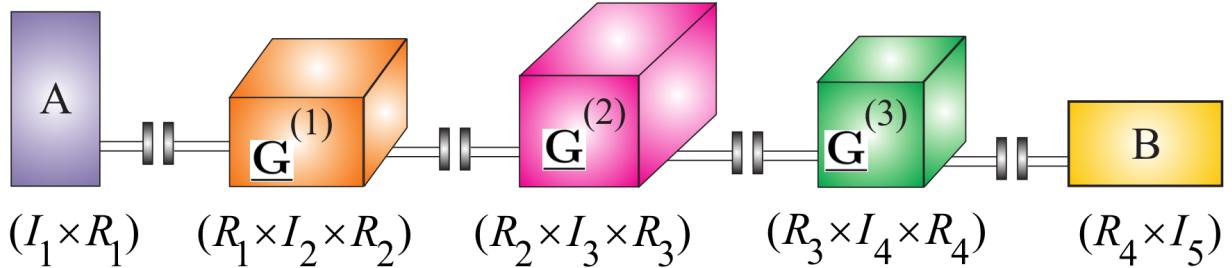


Рис. 2.3. Tensor Train Decomposition

ставляет тензор N -го порядка $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ следующим образом

$$\underline{\mathbf{X}} = [\mathbf{A}, \underline{\mathbf{G}}^{(1)} \underline{\mathbf{G}}^{(2)}, \dots, \underline{\mathbf{G}}^{(N-1)}, \mathbf{B}] = \mathbf{A} \times_2^1 \underline{\mathbf{G}}^{(1)} \times_3^1 \underline{\mathbf{G}}^{(2)} \times_3^1 \dots \times_3^1 \underline{\mathbf{G}}^{(N-1)} \times_3^1 \mathbf{B},$$

,

2.4. Вычислительный эксперимент и анализ ошибки

Код основного вычислительного эксперимента можно найти по ссылке. Для эксперимента был использован датасет CC-359, который содержит fMRI изображения головного мозга человека, представленные в виде трехмерных тензоров. Также для работы с расширением снимков будет полезна библиотека.

Были реализованы алгоритмы *CPD* и два варианта разложения Такера: *HOSVD* и *HOOI*. По результатам применения алгоритмов к снимкам, можно сделать вывод, что алгоритмы *HOSVD* и *HOOI* дают лучшее приближение тензора (лучше визуальное качество картинки и меньшая величина ошибки) нежели *CPD* при аналогичных значениях ранга Крускала для *CPD* и мультиранга для *HOSVD* и *HOOI*. На графике ошибки *HOOI* и *HOSVD* мультиранг равен (k, k, k) .

Относительно времени работы алгоритмов можно сделать следующие выводы. *HOSVD* показал себя лучше всего, выполнив 100 итераций за 2 : 24 минуты,

HOOI выполнил также 100 итераций за 6 : 19 минут. *CPD* отработал 21 итерацию за 12 : 19 минут.

Исходя из результатов эксперимента, можно сделать вывод, что использование алгоритма *HOSVD* является оптимальным для данной задачи.

Теперь немного о грустном. Рассматривалась также реализация разложения Tensor Train, ее можно увидеть в коде. Данный алгоритм удалось запустить только для маленьких синтетических данных. Для запуска хотя бы одной итерации алгоритма на реальном изображении fMRI не хватило вычислительных ресурсов. Также был написан автоэнкодер, но для него не было получено адекватных результатов на снимках fMRI. С черновым вариантом кода можно ознакомиться по ссылке.

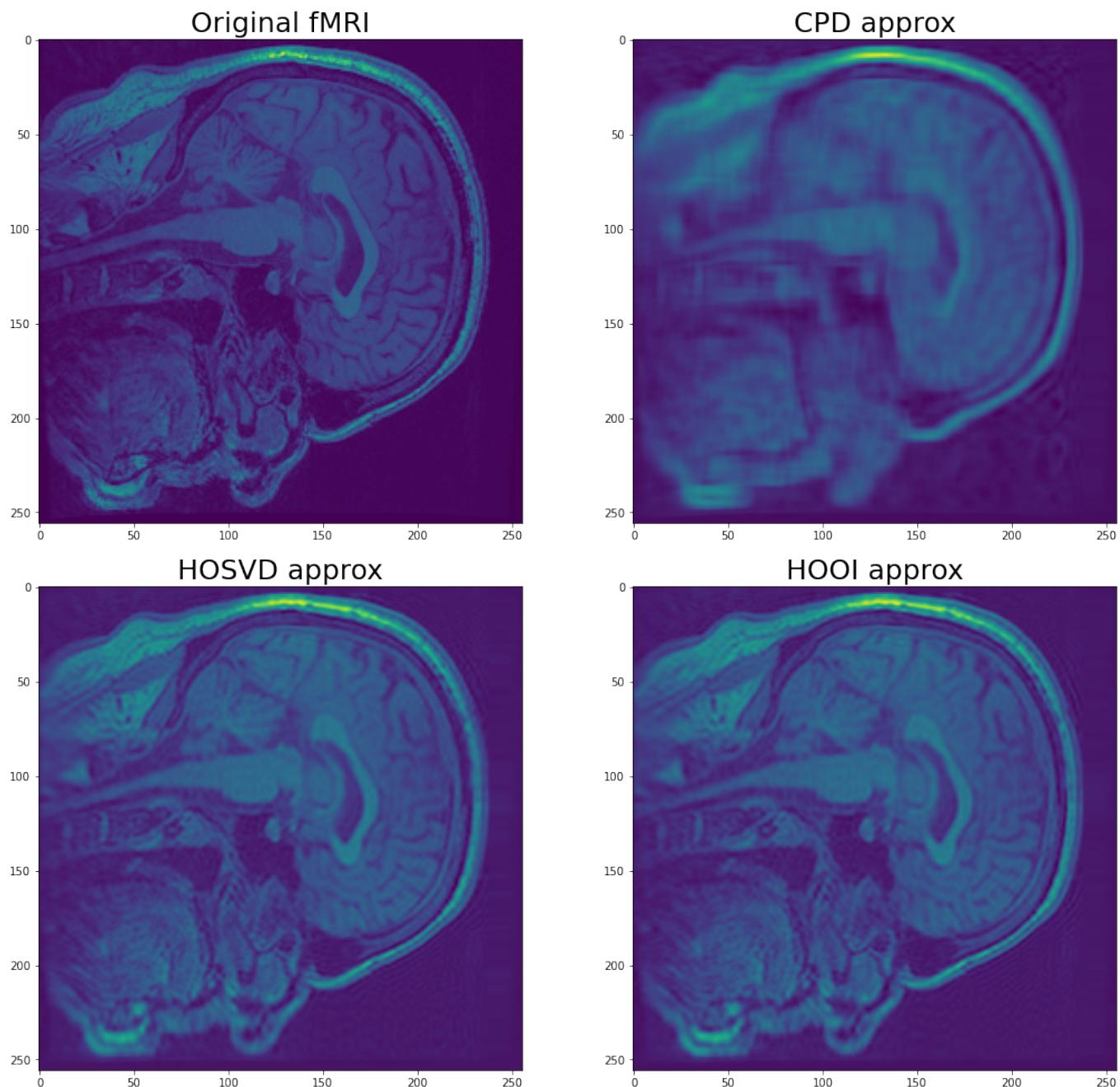


Рис. 2.4. Оригинальный снимок $fMRI$ и результаты применения CPD , $HOSVD$ и $HOOI$ соответственно

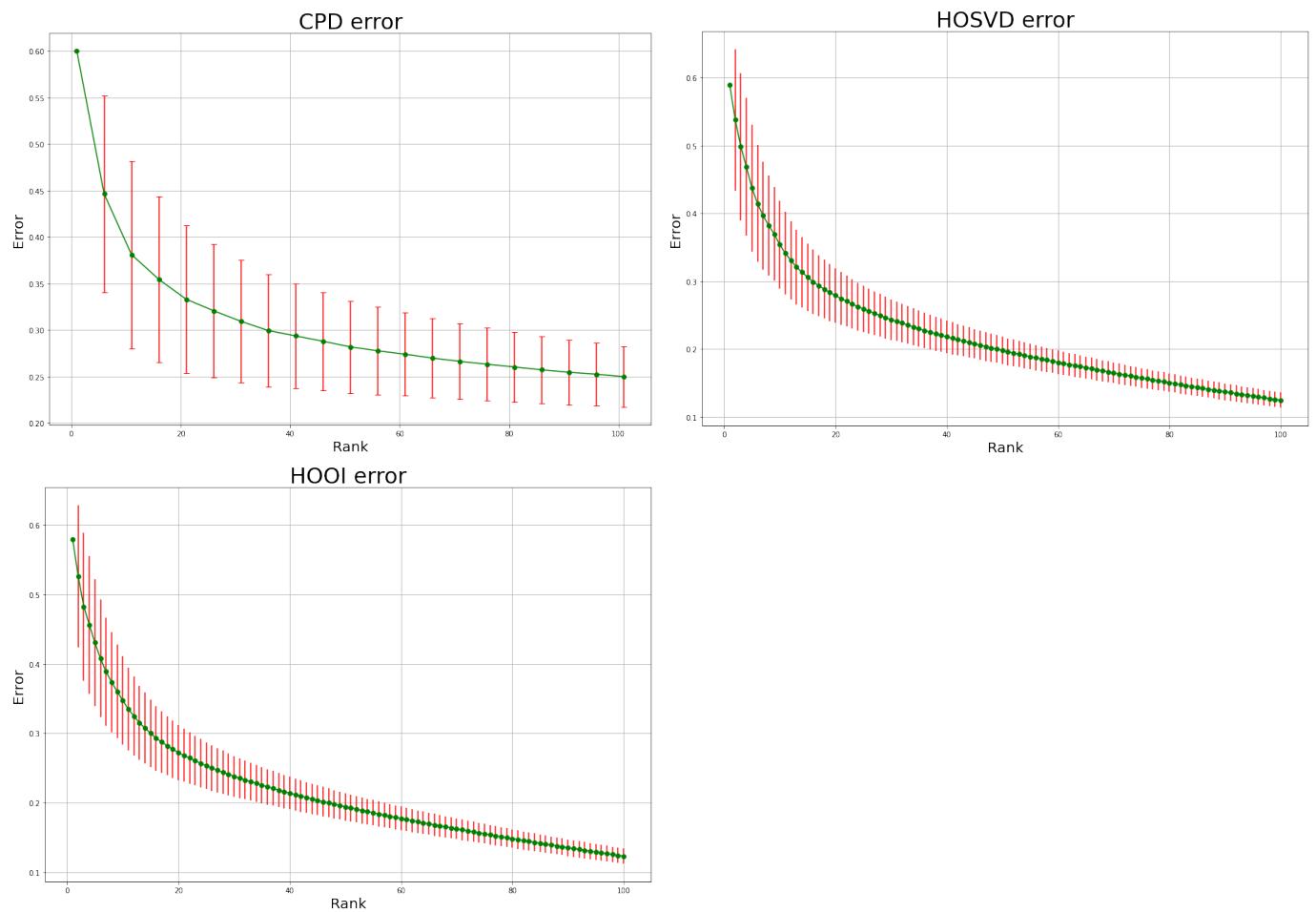


Рис. 2.5. График зависимости ошибки соответствующих алгоритмов от ранга

Глава 3

kornilov

3.1. Введение

В работе исследуются 2-х и 3-х мерные фазовые пространства, которые используются для предсказания и анализа свойств многомерных временных рядов. В частности рассматриваются алгоритмы Higher Order Singular Spectrum Analysis для предсказания и HOSVD для получения фазовых пространств.

3.2. Математическая постановка задачи

Здесь сохранены обозначения из книги [Golyandina2001AnalysisOT].

Пусть $F = (f_0, \dots, f_{N-1})$ - доступный временной одноцветный видеоряд длины N , где $f_i \in \mathbb{R}^{n \times m}$ - фрейм в момент времени i , аналогично $G = (g_1, \dots, g_M)$, где $g_j \in \mathbb{R}^{n \times m}$ - целевая видеоряд, который необходимо предсказать.

Необходимо найти такую оптимальную модель $model : \mathbb{R}^{N \times (n \times m)} \rightarrow \mathbb{R}^{M \times (n \times m)}$ из множества моделей U , что функция ошибки \mathcal{L} между предсказанным и истинным рядом была бы минимальной.

$$model_* = \arg \min_{model \in U} \mathcal{L}(model(F), G)$$

В этой работе мы ограничимся лишь моделями типа SSA, описанными ниже, а в качестве ошибки \mathcal{L} будем рассматривать норму Фробениуса.

Опишем, как работает предсказательная SSA модель $model \in U$.

Гиперпараметрами модели являются ширина окна L , количество lagged векторов K и фазовое подпространство ($K > L$), определяемое числом и k главных компонент в HOSVD разложении.

- На первом шаге из ряда F строится траекторная матрица X_{tr} размера $(L \times K \times (n \times m))$

$$X_{tr} = \begin{pmatrix} f_0 & f_1 & f_2 & \dots & f_{K-1} \\ f_1 & f_2 & f_3 & \dots & f_K \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f_{L-1} & f_L & f_{L+1} & \dots & f_{N-1} \end{pmatrix}$$

Эта матрица имеет одинаковые элементы X_{ij} , когда $i + j = const$. Матрицы с таким свойством называются Hankel matrix.

Заметим, что любую матрицу размера $(L \times K \times (n \times m))$ можно диагонализовать с помощью формулы

$$s = i + j$$

$$\hat{x}_{ij} = \begin{cases} \frac{1}{s-1} \sum_{l=1}^{s-1} x_{l,s-l}, & 2 \leq s \leq L-1 \\ \frac{1}{L} \sum_{l=1}^L x_{l,s-l}, & L \leq s \leq K-1 \\ \frac{1}{K+L-s+1} \sum_{l=s-K}^L x_{l,s-l}, & K+2 \leq s \leq K_L \end{cases}$$

Что также можно записать через оператор H :

$$\hat{X} = H X$$

А также из любой матрицы можно выделить временной ряд по формуле

$$g_k = \begin{cases} \frac{1}{k+1} \sum_{m=1}^{k+1} x_{m,k-m+2}, & 0 \leq k < L-1 \\ \frac{1}{L} \sum_{m=1}^L x_{m,k-m+2}, & L-1 \leq k \leq K-1 \\ \frac{1}{N-K} \sum_{m=k-K+2}^{N-K+1} x_{m,k-m+2}, & K \leq k \leq N \end{cases}$$

2. На следующем шаге производится HOSVD разложение матрицы X_{tr} и выбираются наибольшие по модулю значения λ из core tensor. Так чтобы в итоге осталось всего k столбцов матрицы из U размера $L \times s_L$. Получается, что для каждого пикселя из $n \times m$ матрицы, есть матрица P_{nm} размера $L \times k$ из ортогональных векторов, которые образуют базис в фазовом подпространстве \mathcal{L}_{nm}^k в траекторном ряде пикселя nm .

Также можно восстановить с помощью отобранных главных компонент матрицу X_{rec} и извлечь из неё ряд F_{rec} .

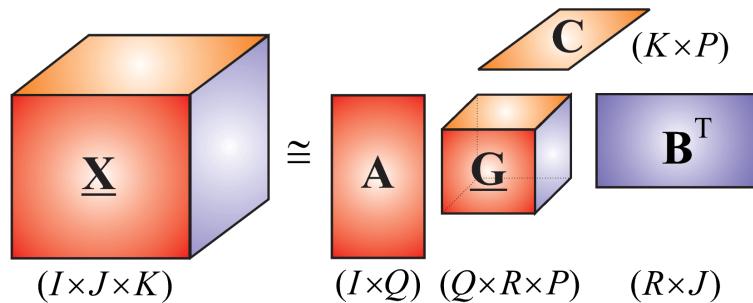


Рис. 3.1. HOSVD

3. Далее рассматриваем каждый пиксель n, m отдельно. Для него есть траекторная матрица X_{tr}^{nm} (та же что и в первом пункте) и подпространство P_{nm} . Для удобства индекс nm опустим.

4. Проецируем каждый вектор из X_{tr} на фазовое пространство \mathcal{L}^k

$$\tilde{X}_{tr} = \sum_{i=1}^k P_i P_i^T X_{tr}$$

5. Далее делаем из неё матрицу Hankel через оператор H и извлекаем из неё ряд \hat{F}

$$\hat{X}_{tr} = H \tilde{X}_{tr}$$

6. Для каждого вектора $Y \in \mathbb{R}^L$ обозначим через $Y_\Delta \in \mathbb{R}^{L-1}$ вектор, состоящий из последних $L - 1$ компонент Y , а через $Y^\nabla \in \mathbb{R}^{L-1}$ вектор, состоящий из первых $L - 1$ компонент Y .
7. Положим $\nu^2 = \sum_{i=1}^k \pi_i^2$, где π_i - последняя компонента P_i . По определению ν^2 — квадрат косинуса угла между e_L и пространством \mathcal{L}^k .
8. Если $e_L \notin \mathcal{L}^k$, то $\nu^2 < 1$. Тогда можно доказать, что для любого $y = (y_1, \dots, y_L)^T \in \mathcal{L}^k$ последняя компонента y_L выражается через линейную комбинацию первых $L - 1$ компонент (свойство LRF)

$$y_L = a_1 y_{L-1} + \dots + a_{L-1} y_1$$

Вектор $R = (a_{L-1}, \dots, a_1)^T$ можно выразить через формулу

$$R = \frac{1}{1 - \nu^2} \sum_{i=1}^k \pi_i P_i^\nabla$$

Причём вектор R не зависит от базиса в \mathcal{L}^k

9. Определим ряд $G_{N+M}^{pred} = (g_0, \dots, g_{N+M-1})$ как

$$g_i = \begin{cases} \hat{f}_i, & i = 0, \dots, N-1 \\ \sum_{j=1}^{L-1} a_j g_{i-j}, & i = N, \dots, N+M-1 \end{cases}$$

Это и будет объединение ряда F и его предсказания $G_{N+M}^{pred} - F$ для пикселя nm

10. Сделаем из ряда G_{N+M}^{pred} траекторную матрицу X_{tr}^{pred} и найдём фазовые координаты её ортогональной проекции на \mathcal{L}^k в каждый из K моментов сдвига

$$\phi = P^T X_{tr}^{pred}$$

Класс рядов F , для которых выполняется свойство LRF крайне широк. Это гармонические, экспоненциальные ряды, многочлены и их суммы.

3.3. Вычислительные эксперименты

Collab Notebook

Для предсказания использован gif-видеоролик F в нескольких вариациях:

1. Обычный

2. Зашумленный

Полный видеоролик Видеоролик с пониженным разрешением

Размерность ролика $(168, 300)$, а длина 200 кадров, предсказать нужно было на 200 кадров.

Рассматривались фазовые пространства размерности 2, 3.

Реализация алгоритма HOSVD взята из библиотеки [**hottbox**].

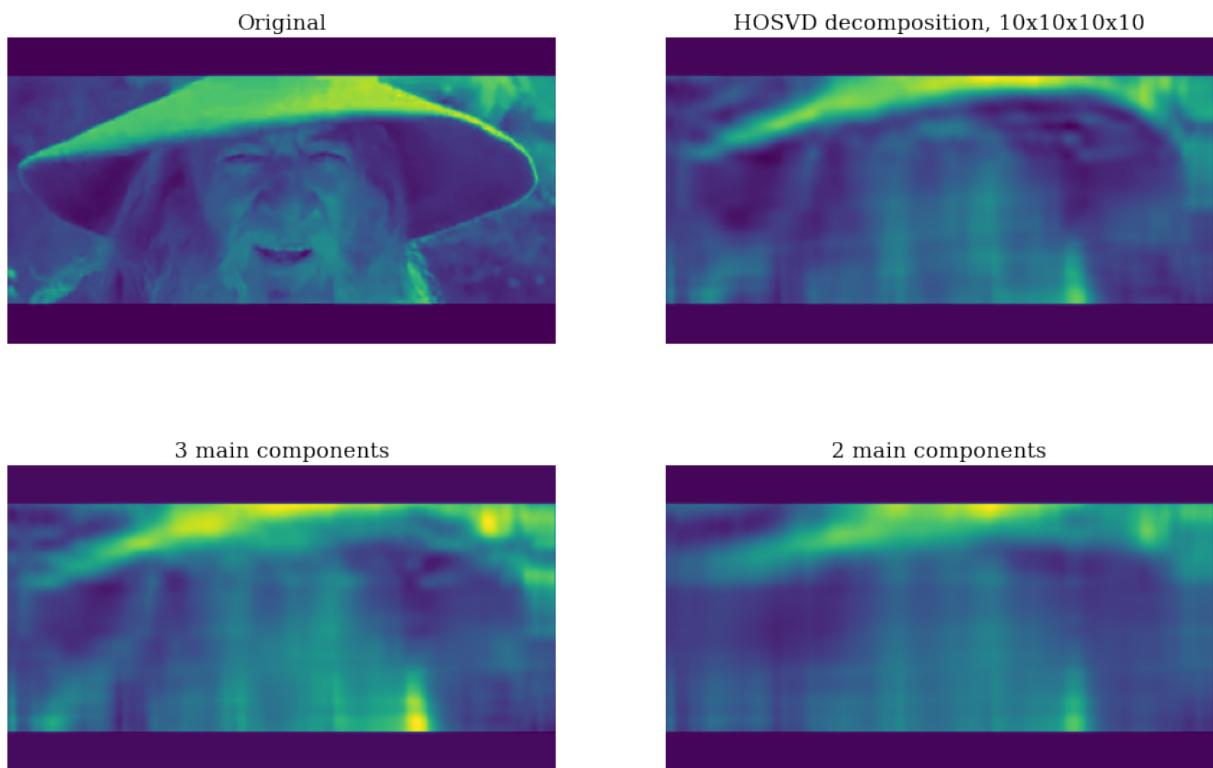


Рис. 3.2. Результат применения HOSVD разложения

Траектории в фазовых пространства строились для сильно колебающейся точки из середины кадра, для более стационарной из угла и для неизменной точки из рамки.

Ошибка считалась как средняя норма разности матриц.

Если размер окна не покрывает период видео, то фазовая траектория получается затухающей.

Когда длина достаточна для предсказания, ошибка становится куда лучше. Заметим, что 2—мерного фазового пространства для качественного предсказания в случае этого видео не достаточно.

Видео с отсчётом 1

Видео с отсчётом 2

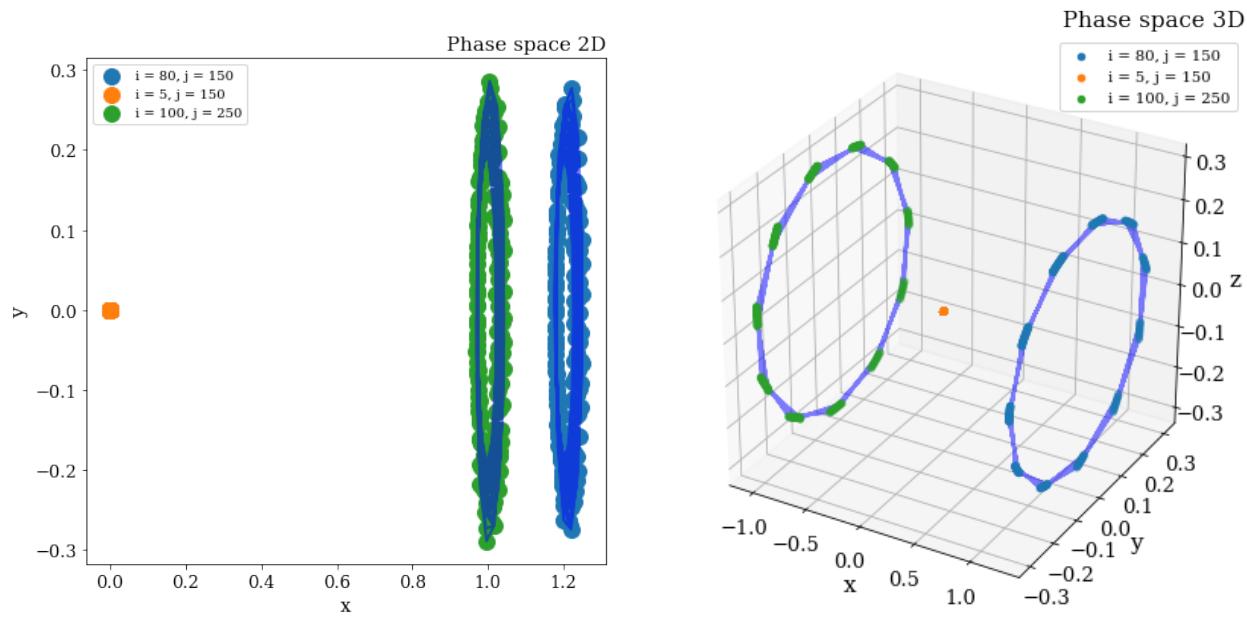


Рис. 3.3. 2-х и 3-х мерные фазовые пространства без шума

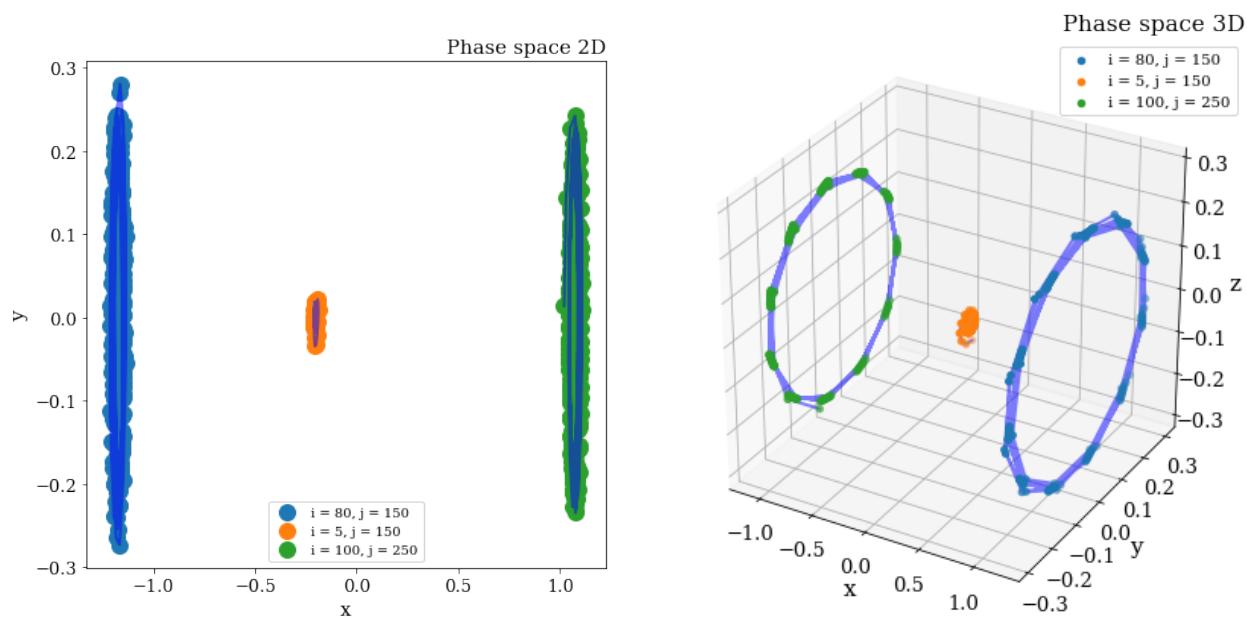


Рис. 3.4. 2-х и 3-х мерные фазовые пространства с шумом

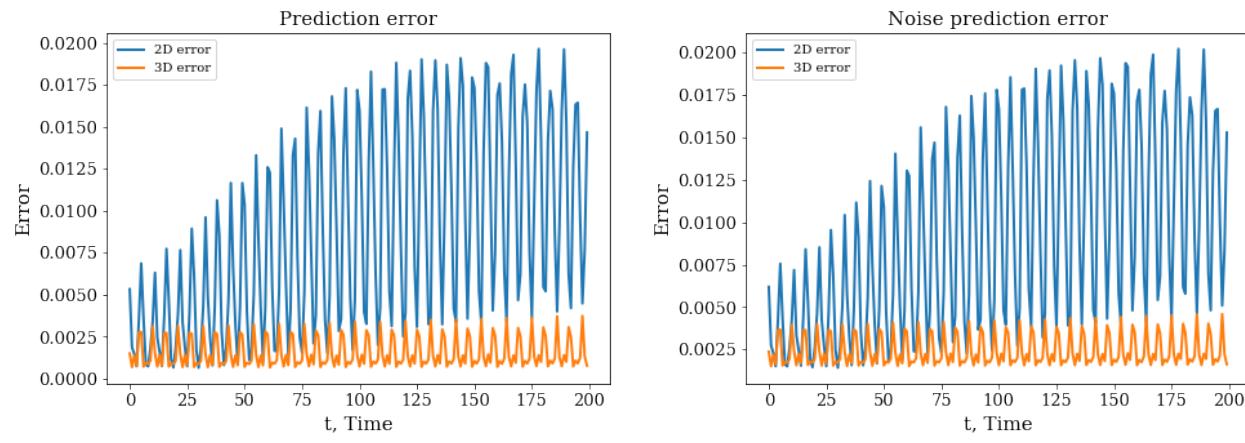


Рис. 3.5. Ошибки предсказания без и с шумом

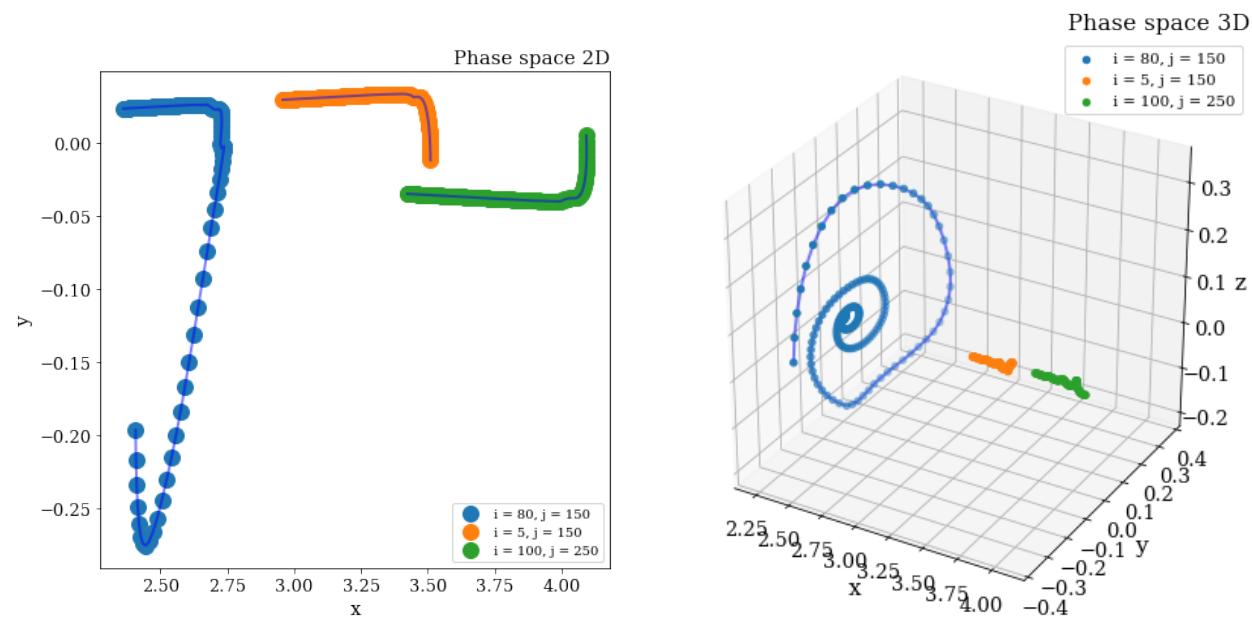


Рис. 3.6. 2-х и 3-х мерные затухающие фазовые траектории

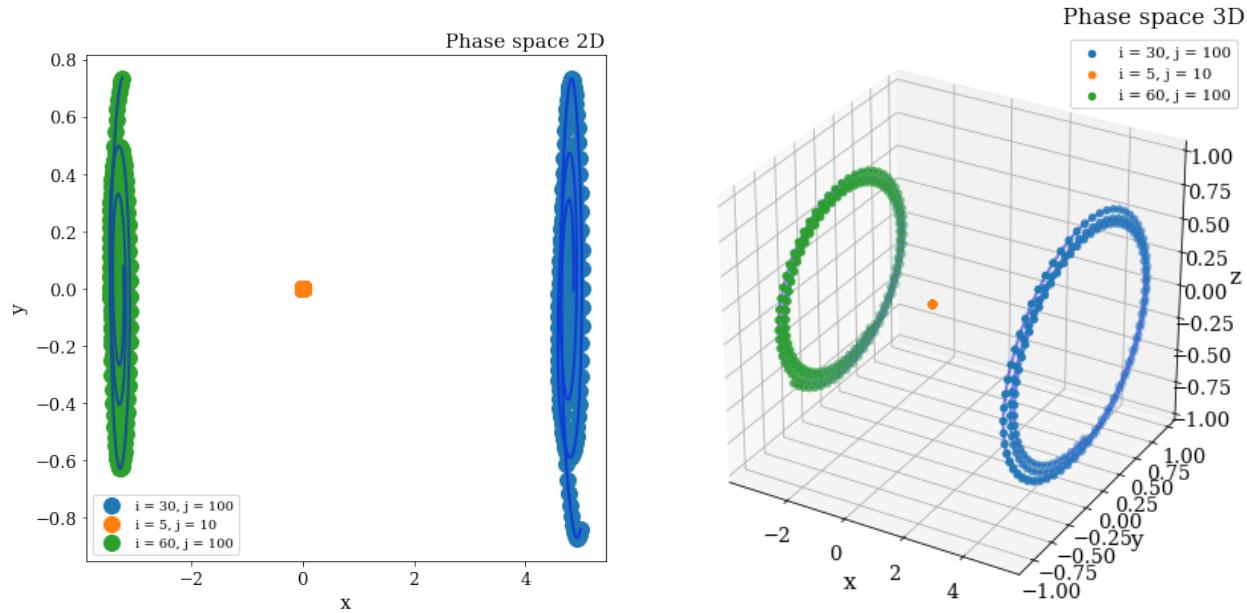


Рис. 3.7. 2-х и 3-х мерные исправленные траектории

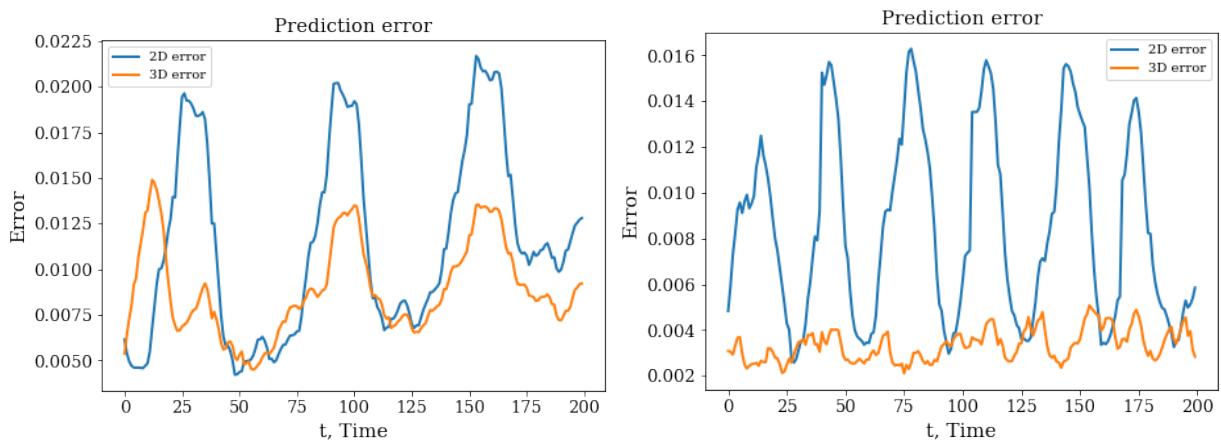


Рис. 3.8. Ошибки предсказания с малым L и с большим L

3.4. Выводы

Для анализа большого количества многомерных временных рядов вполне достаточно 2-х и 3-х мерных фазовых пространств. Алгоритм SSA при всей своей простоте позволяет эффективно выделять тренды, удалять шумы из данного временного ряда, а также предсказывать их на любой заданный горизонт. Алгоритм HOSVD работает быстро и стабильно, при этом он учитывает связь между всеми элементами в матрице, взятой в определенный момент времени.

3.5. Литература

main

Глава 4

molozhavenco

4.1. Введение

Целью данной работы является проведение экспериментов по прогнозированию временных рядов с помощью *HOPLS*.

Эксперименты проводились на данных Walk accelerometer.

4.2. Постановка задачи

Имеется датасет $\mathcal{D} = (\mathbf{X}, \mathbf{Y})$, где \mathbf{X} — матрица, составленная из значений временного ряда с периодом p , а \mathbf{Y} — матрица целевых значений.

Предполагается, что между данными имеется некоторая линейная зависимость, которую и нужно научиться предсказывать.

$$\mathbf{Y} = \mathbf{X} \cdot \Theta + \varepsilon$$

В терминах теории оптимизации:

$$\|\mathbf{X}\Theta - \mathbf{Y}\|^2 \rightarrow \min_{\Theta}$$

Данная задача имеет множество методов решения, в зависимости от размеров данных. В случае, если $\mathbf{X} \in \mathbb{R}^{m \times n}$, $\mathbf{Y} \in \mathbb{R}^{m \times r}$, поставленная оптимизационная задача запишется более конкретно:

$$\sum_{i=1}^m \|\mathbf{X}_i\Theta - \mathbf{Y}_i\|_2^2 \rightarrow \min_{\Theta}$$

Полученная задача, может быть решена методом *Partial Least Squares (PLS)*, принцип работы, которого вкратце будет освещен ниже.

Однако, не совсем становится понятно, что делать, когда, к примеру, $\mathbf{X} \in \mathbb{R}^{(J_1 \times J_2 \times \dots \times J_n)}$, а $\mathbf{Y} \in \mathbb{R}^{(J_1 \times J_2 \times \dots \times J_m)}$. В таких случаях как раз и применяется *Higher-Order Partial Least Squares (HOPLS)* метод.

4.3. Метод

a) Двумерный случай.

Если $\mathbf{X} \in \mathbb{R}^{m \times n}$, $\mathbf{Y} \in \mathbb{R}^{m \times r}$, то для задачи регрессии можно применить обычный метод *PLS*, который заключается в реализации следующей схемы:

- 1) Операторами вращения перевести $x \in \mathbb{R}^n$ в латентное пространство \mathbb{R}^l , $l < n$

- 2) Операторами вращения перевести $y \in \mathbb{R}^r$ в латентное пространство $\mathbb{R}^l, l < r$
- 3) Подобрать, такое латентное пространство, в котором образы x и y имеют наибольшую корреляцию

$$\begin{aligned} \boxed{\mathbf{X}}_{(I \times J)} &= \boxed{\mathbf{T}}_{(I \times R)} \boxed{\mathbf{P}^T}_{(R \times J)} + \boxed{\mathbf{E}}_{(I \times J)} = \sum_{r=1}^R \boxed{\mathbf{t}_r} \boxed{\mathbf{p}_r^T}_{(I \times J)} + \boxed{\mathbf{E}}_{(I \times J)} \\ \boxed{\mathbf{Y}}_{(I \times M)} &= \boxed{\mathbf{T}}_{(I \times R)} \boxed{\mathbf{Q}^T}_{(R \times M)} + \boxed{\mathbf{F}}_{(I \times M)} = \sum_{r=1}^R \boxed{\mathbf{t}_r} \boxed{\mathbf{q}_r^T}_{(I \times M)} + \boxed{\mathbf{F}}_{(I \times M)} \end{aligned}$$

Рис. 4.1. Иллюстрация метода *PLS*

Это можно записать, как:

$$\begin{aligned} X &= TP^\top + E \\ Y &= UQ^\top + F \end{aligned}$$

Где E и F — соответствующие матрицы ошибок, $T = [\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_R] \in \mathbb{R}^{I \times R}$ — состоит из R ортонормированных латентных переменных для X , $U = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_R] \in \mathbb{R}^{I \times R}$ — латентные переменные для Y , причем столбцы матрицы U имеют максимальную корреляцию со столбцами матрицы T . Матрицы P и Q — соответствующие матрицы поворота.

Для того, чтобы отыскать матрицу T

$$\max_{w,q} [w^\top X^\top Y q], \text{ s. t } w^\top w = 1, q^\top q = 1.$$

Затем латентная переменная оценивается как $\mathbf{t} = \mathbf{X}\mathbf{t}$. Основываясь на предположении о линейной зависимости \mathbf{X} и \mathbf{Y} , предсказание строится следующим образом:

$$Y \approx TDQ^\top$$

Где D — диагональная матрица, с элементами $d_{ii} = \frac{\mathbf{u}_i^\top \mathbf{t}_i}{\mathbf{t}_i^\top \mathbf{t}_i}$

- b) Если матрицы X и Y имеют большее количество индексов, то может быть применен *HOLPS*.

Идея его работы точно такая же, как и у *PLS*, вопрос состоит лишь в том, как получить разложения. Ответ — блоки Такера:

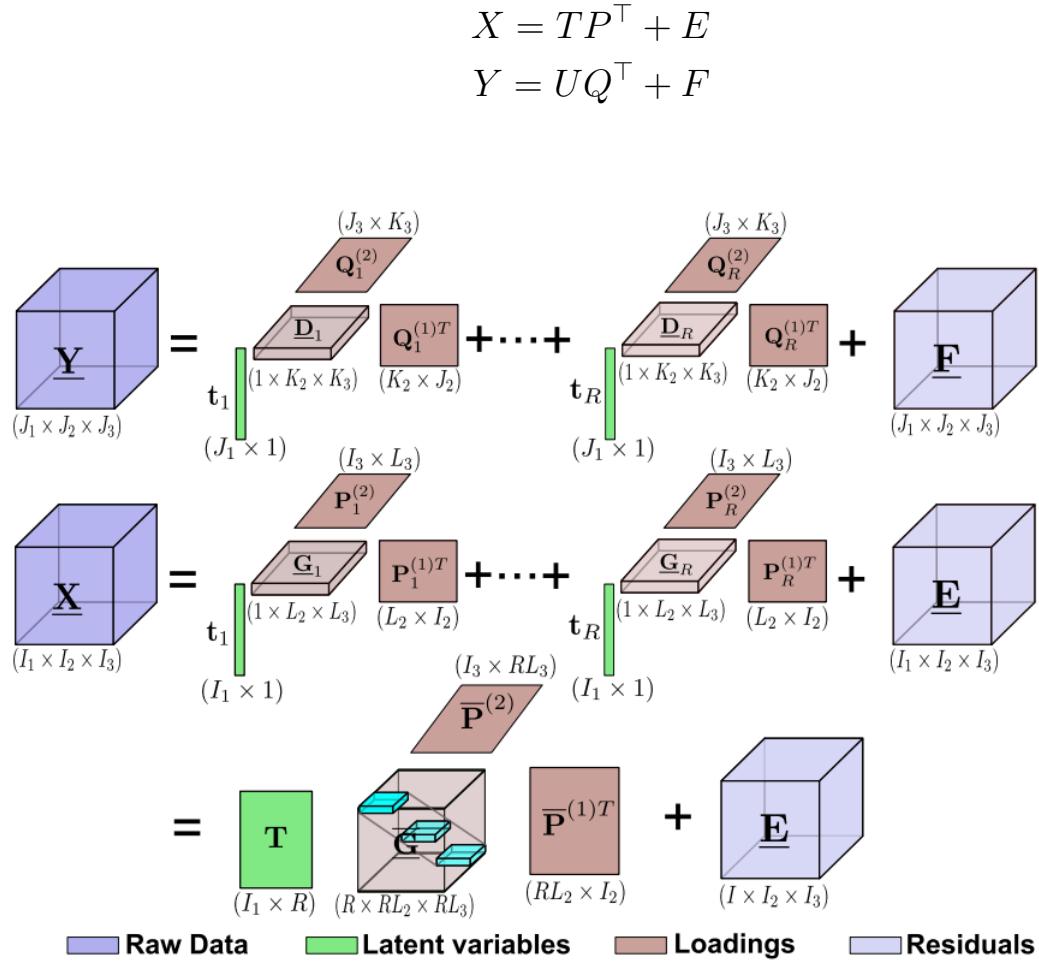


Рис. 4.2. Иллюстрация метода *HOPLS*

4.4. Вычислительный эксперимент и анализ ошибки

Датасет Walk accelerometer представляет собой зафиксированные значения ускорения и показателей гироскопа при выполнении человеком определенных действий таких как ходьба, бег и проч.

Эксперименты были проведены с помощью языка программирования python и таких пакетов как numpy, sklearn, и доступны по ссылке.

Главная трудность в проведении эксперимента, с которой я столкнулся, — отсутствие алгоритма *HOLPS* в *HOTTOBOX*. Поэтому пришлось искать другие готовые решения, одно из которых я и использовал. Данное решение было представлено 4 года назад и уже успело сильно устареть, пришлось копаться в коде и переписывать часть модулей на новые версии библиотек.

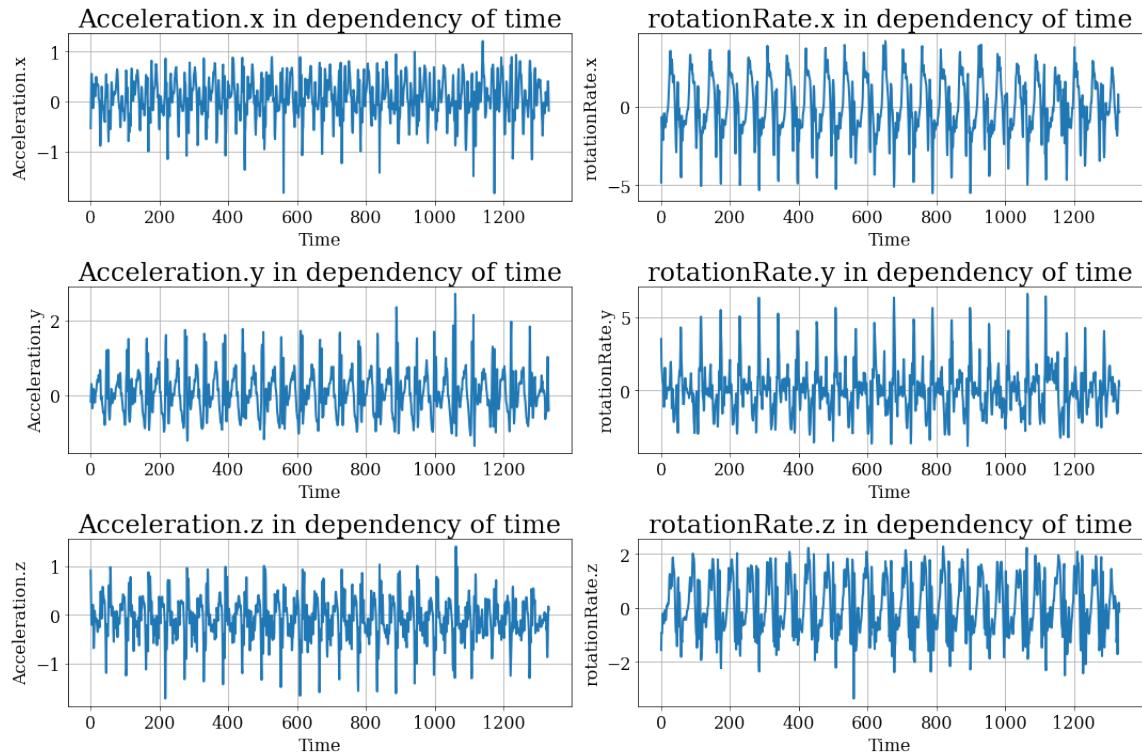


Рис. 4.3. Визуализация данных

В качестве тензора **X** я взял фазовые матрицы для x, y, z компонент ускорения, зафиксированных при ходьбе человека. По ним, с помощью метода *HOLPS*, я предсказывал тензор **Y** — фазовые матрицы для x, y, z компонент показаний гироскопа:

	rotationRate.x	rotationRate.y	rotationRate.z
userAcceleration.x	-0.1397	-0.2036	-0.1060
userAcceleration.y	0.2666	0.1080	-0.0729
userAcceleration.z	-0.0281	0.3082	0.1436

Таблица 4.1. Корреляция данных

Как видно из 4.1 линейная связь если и есть, то очень слабая.



Рис. 4.4. Графики наиболее коррелированных компонент

Однако из графиков, «проглядывается» четкая зависимость компонент друг от друга. Вопрос: увидит ли это модель?

4.4.1. Эксперимент 1

Начнем с простого эксперимента: По тензору \mathbf{X} — фазовые матрицы для всех компонент ускорения предсказываем игрек компоненту показаний гироскопа:

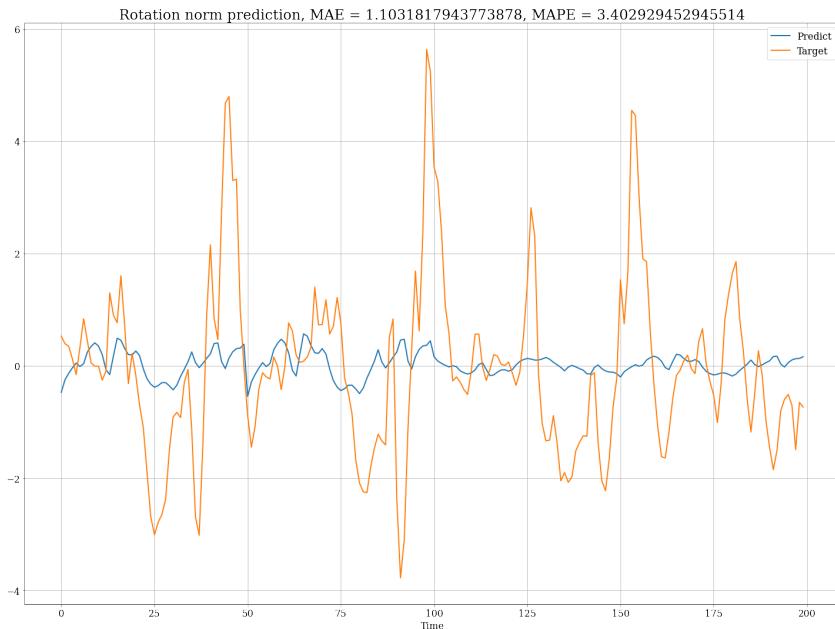


Рис. 4.5. Предсказание со «случайными гиперпараметрами»

Далее я запустил *python* пакет для умного перебора гиперпараметров под называнием *optuna* и получил следующий результат:

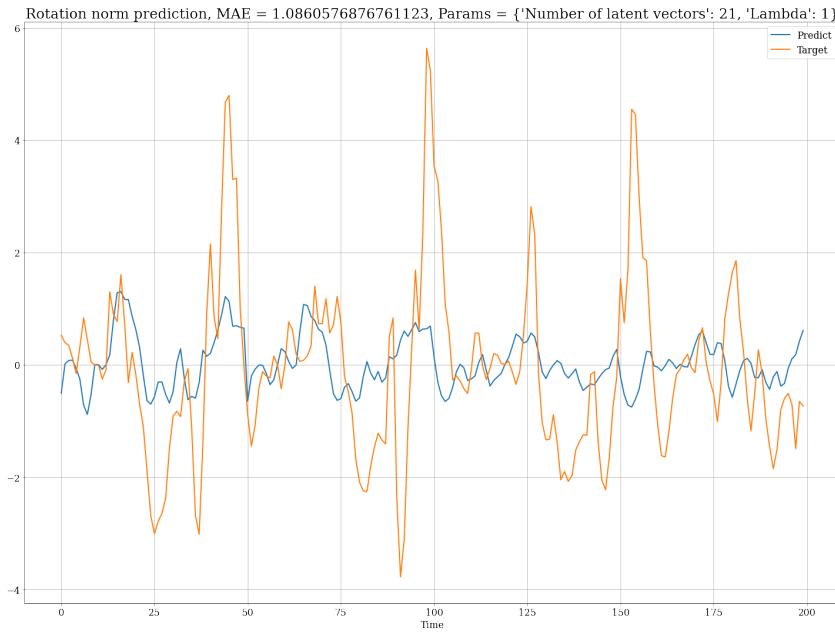


Рис. 4.6. Предсказание с оптимальными гиперпараметрами

Более подробные результаты по работе пакета, а так же оцененная важность гиперпараметров можно посмотреть в самом журнале эксперимента.

4.4.2. Эксперимент 2

По тензору \mathbf{X} — фазовые матрицы для всех компонент ускорения предсказываем все компоненты показаний гироскопа:



Рис. 4.7. Предсказание со «случайными гиперпараметрами»

Далее я снова запустил *python* пакет *optuna* и получил следующий результат:



Рис. 4.8. Предсказание с оптимальными гиперпараметрами

Более подробные результаты по работе пакета, а так же оцененная важность гиперпараметров можно посмотреть в самом журнале эксперимента.

4.5. Выводы

На мой взгляд результаты регрессии получились довольно слабыми. Это связано прежде всего с нелинейной связью данных. Дальнейшая работа может быть проведена в области написания более нового и оптимального *HOPLS* алгоритма, а так же можно попробовать применить обычный *PLS* и сравнить полученные результаты.

Глава 5 Solodyankin

5.1. Введение

В задачах машинного обучения зачастую используются сложные алгоритмы, которые требуют большого количества ресурсов. Встает вопрос об оптимизации и снижении сложности, за счет незначительного снижения точности. В данной работе рассматривается вариант снижения размерности входных данных, удаляя не несущие дополнительной информации признаки. В работе используется датасет Cartoon, который приводится к трехмерному тензору. Необходимо исследовать способы получения мало ранговых приближений данного тензора.

5.2. Постановка задачи

Формально задача ставится следующим образом. Имеется исходная последовательность кадров, она представляется в виде трехмерного тензора X . Рассматривается его приближение X^* и ошибка аппроксимации $\varepsilon = \frac{\|X - X^*\|_F}{\|X\|_F}$. Исследуется поведение ошибки при увеличении ранга аппроксимации тензора (мультиранг k_1, k_2, k_3 для разложения Такера в мультиранг k_1, k_2, k_3 для разложения Такера). Ранг подбираем следующим образом - останавливаемся, когда на последних шагах дисперсия ошибки становится меньше ϵ . Достаточная точность приближения выбирается следующим образом: мы прекращаем увеличивать ранг, когда дисперсия ошибки перестает значимо изменяться между шагами алгоритма.

5.3. Модель

В работе рассматривались следующие модели: Tucker Decomposition в двух вариантах HOSVD и HOOI, Tensor Train Decomposition. Реализации алгоритмов были взяты из библиотеки *hottbox*.

5.3.1. Tucker Decomposition

Tensor Train Decomposition

Разложение Такера заключается в представлении тензора $\underline{X} \in \mathbb{R}^{I \times J \times K}$ в виде тензора-ядра, заполненного элементами \underline{G} и набора фактор-матриц $\mathbf{A} \in \mathbb{R}^{I \times Q}, \mathbf{B} \in \mathbb{R}^{J \times R}$ и $\mathbf{C} \in \mathbb{R}^{K \times P}$ Иначе говоря, тензор \underline{X} представляется в форме

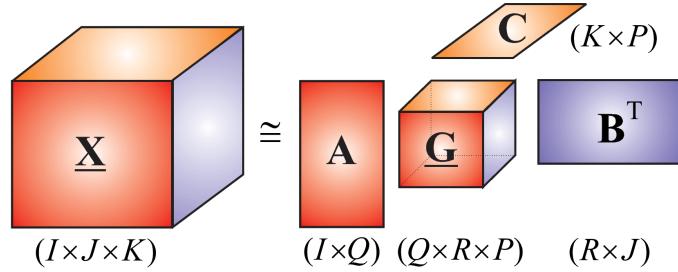


Рис. 5.1. Tucker Decomposition

Такера следующим образом:

$$\begin{aligned}\underline{\mathbf{X}} &\simeq \sum_{q=1}^Q \sum_{r=1}^R \sum_{p=1}^P g_{qrp} \mathbf{a}_q \circ \mathbf{b}_r \circ \mathbf{c}_p \\ &= \underline{\mathbf{G}} \times {}_1\mathbf{A} \times {}_2\mathbf{B} \times {}_3\mathbf{C} \\ &= [\mathbf{G}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\end{aligned}$$

В пакете hottbox разложение Такера реализовано в двух видах: *HOSVD* и *HOOI*.

5.3.2. HOSVD

Higher Order Singular Value Decomposition (*HOSVD*) - частный случай разложения Таккера, в котором все матрицы факторов ограничены ортогональностью. Они вычисляются как усеченная версия сингулярных матриц всех возможных поворотов тензора.

$$\begin{aligned}\mathbf{X}_{(1)} &= \mathbf{U}_1 \boldsymbol{\Sigma}_1 \mathbf{V}_1^T \quad \rightarrow \quad \mathbf{A} = \mathbf{U}_1 [1 : R_1] \\ \mathbf{X}_{(2)} &= \mathbf{U}_2 \boldsymbol{\Sigma}_2 \mathbf{V}_2^T \quad \rightarrow \quad \mathbf{B} = \mathbf{U}_2 [1 : R_2] \\ \mathbf{X}_{(3)} &= \mathbf{U}_3 \boldsymbol{\Sigma}_3 \mathbf{V}_3^T \quad \rightarrow \quad \mathbf{C} = \mathbf{U}_3 [1 : R_3]\end{aligned}$$

Причем, мы рассматриваем трехмерный тензор $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$ как следующее представление (после применения разложения Такера):

$$\underline{\mathbf{X}} = \underline{\mathbf{G}} \times {}_1\mathbf{A} \times {}_2\mathbf{B} \times {}_3\mathbf{C}$$

Для тензора общего порядка - кортеж N -tuple (R_1, \dots, R_N) называется мультилинейным рангом и обеспечивает гибкость при сжатии и аппроксимации исходного тензора. После получения матриц коэффициентов, основной тензор вычисляется как

$$\underline{\mathbf{G}} = \mathbf{X} \times {}_1\mathbf{A}^T \times {}_2\mathbf{B}^T \times {}_3\mathbf{C}^T$$

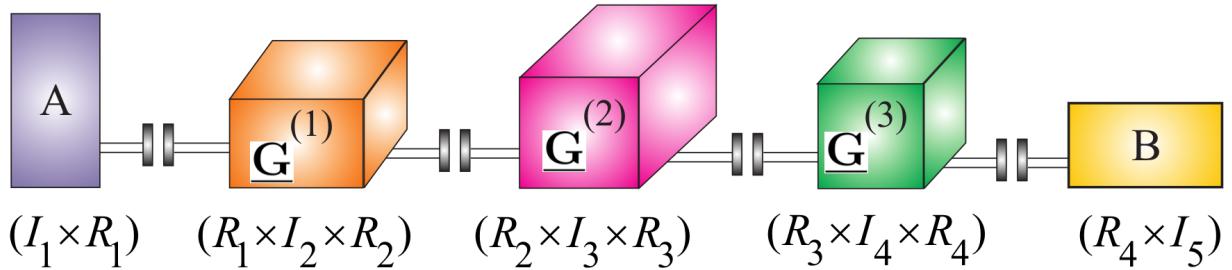


Рис. 5.2. Tensor Train Decomposition

5.3.3. Tensor Train Decomposition

Tensor train decomposition представляет данный тензор в виде мало связанных тензоров меньшего порядка и фактор-матриц. Другими словами, алгоритм представляет тензор N -го порядка $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ следующим образом

$$\begin{aligned}\underline{\mathbf{X}} &= \left[\mathbf{A}, \underline{\mathbf{G}}^{(1)} \underline{\mathbf{G}}^{(2)}, \dots, \underline{\mathbf{G}}^{(N-1)}, \mathbf{B} \right] , \\ \underline{\mathbf{X}} &= \mathbf{A} \times_2^1 \underline{\mathbf{G}}^{(1)} \times_3^1 \underline{\mathbf{G}}^{(2)} \times_3^1 \dots \times_3^1 \underline{\mathbf{G}}^{(N-1)} \times_3^1 \mathbf{B}\end{aligned}$$

5.4. Вычислительный эксперимент и анализ ошибки

Ссылка на код работы

Глава 6

Vladimirov

6.1. Введение

Цель лабораторной работы заключается в применении метода HOSVD для снижения размерности кадров gif-изображения и использовании метода ССМ для обнаружения связи между картинками и звуком. Эксперимент проводится на 15-кадровом черно-белом мультфильме с идущей уткой и сгенерированным звуковым рядом.

Ссылка на код: [тык](#)

6.2. Постановка задачи

Введём обозначения:

$\underline{X} \in R^{N \times I \times J}$ - временной ряд кадров: число кадров · размеры изображения

$Y \in R^N$ - временной ряд звуков

Наша цель заключается в определении наличия связи $\underline{X} \rightarrow Y$. Строгое математическое правило для этого будет предъявлено ниже.

6.2.1. HOSVD

Мы можем расписать \underline{X} как:

$$\underline{X} \cong \sum_{t=1}^N \sum_{i=1}^{R_i} \sum_{j=1}^{R_j} \sigma_{tij} (u_t^{(1)} \circ u_i^{(2)} \circ u_j^{(3)}),$$

где \circ — это внешнее произведение и

$$U^{(1)} = [u_1^{(1)}, \dots, u_N^{(1)}], U^{(2)} = [u_1^{(2)}, \dots, u_{R_i}^{(2)}], U^{(3)} = [u_1^{(3)}, \dots, u_{R_j}^{(3)}].$$

Или в обозначениях Такера:

$$\begin{aligned} \underline{X} &\cong \underline{G} \times_1 U^{(1)} \times_2 U^{(2)} \times_3 U^{(3)}, \\ [G]_{tij} &= \sigma_{tij} \text{ — core-тензор}, \\ \underline{G} &\in R^{N \times R_i \times R_j}, \end{aligned}$$

где \times_i — операция матрично-тензорного умножения.

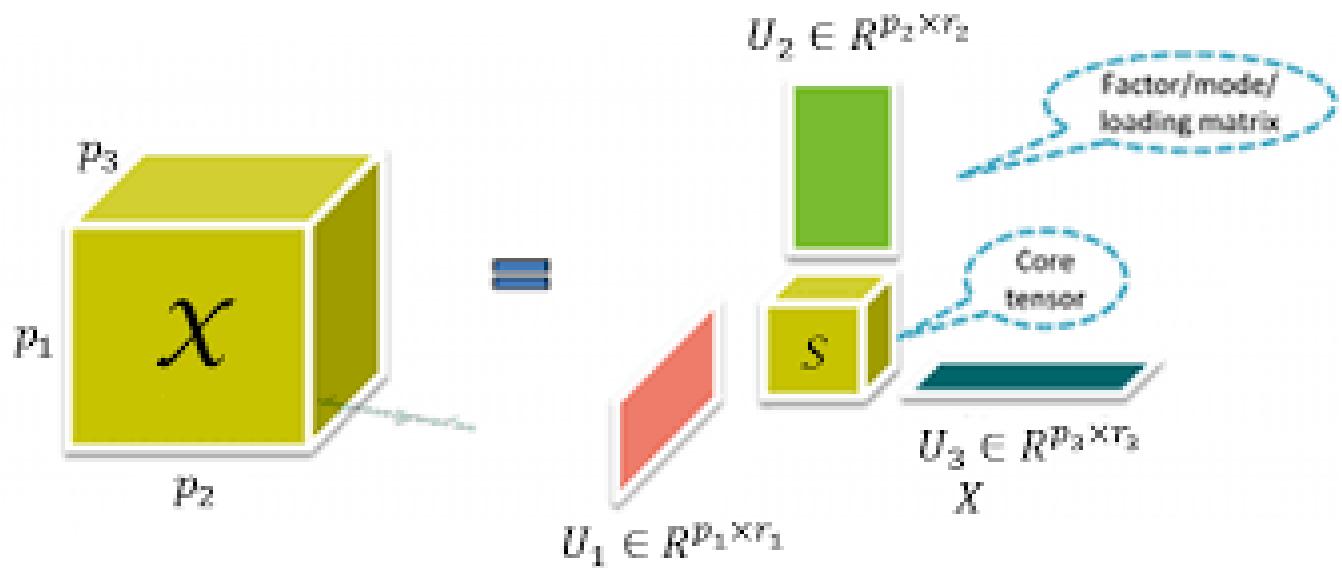


Рис. 6.1. Иллюстрация метода HOPLS

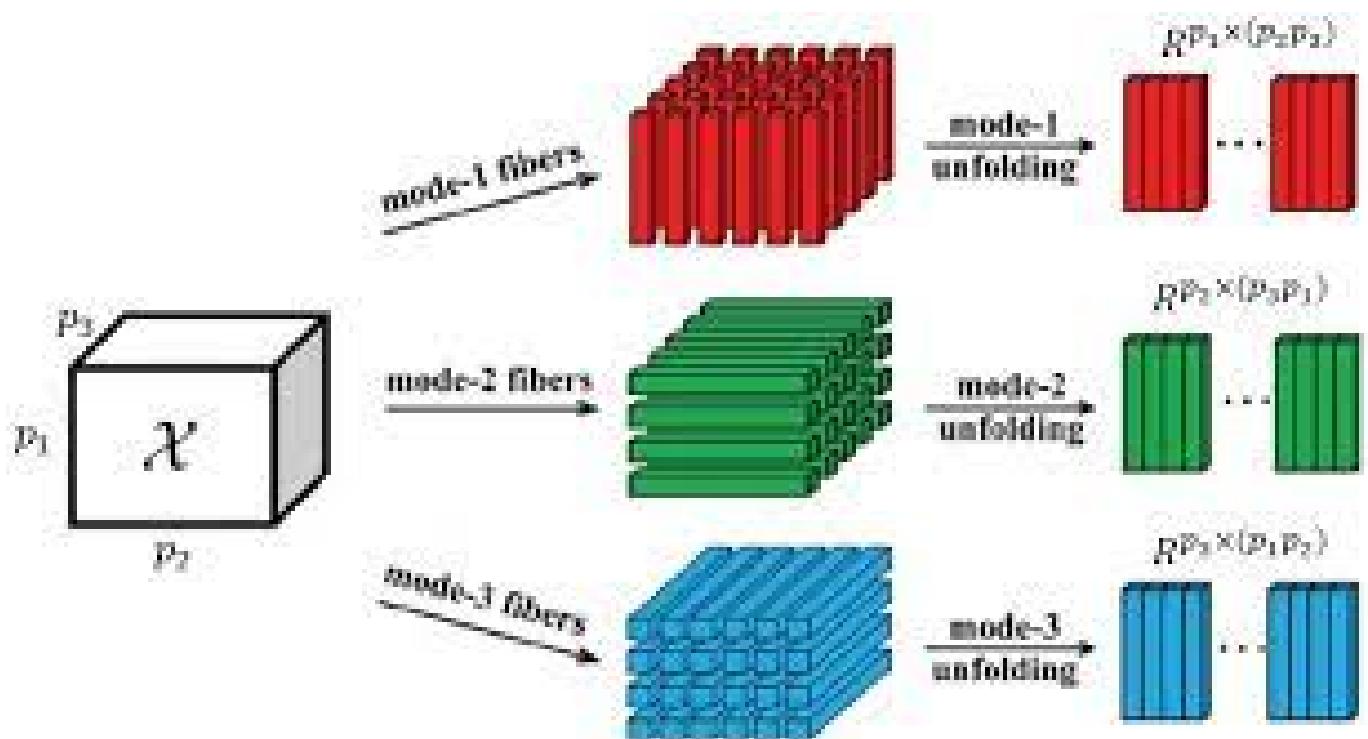


Рис. 6.2. Матризация тензора X , использующая в операция матрично-тензорного умножения \times_l

6.2.2. CCM

Из аппроксимированного тензора \underline{X} и Y составим их многомерные траекторные матрицы, считая период движения равным T :

$$\mathbf{H}_X \in \mathbb{R}^{(N-T+1) \times T \times I \times J},$$

$$\mathbf{H}_Y \in \mathbb{R}^{(N-T+1) \times T},$$

Определим отображение из траекторного пр-ва звукового сигнала $\mathbb{H}_Y \in R^T$ в траекторное пространство кадров $\mathbb{H}_X \in R^{T \times I \times J}$ следующим образом:

$$\varphi : \mathbf{y}_0 \mapsto \widehat{\mathbf{x}}_0 = \sum_{i=1}^K w_i \mathbf{x}_i, \quad w_i = \frac{u_i}{\sum_{j=1}^K u_j}, \quad u_i = \exp(-\|\mathbf{y}_0 - \mathbf{y}_i\|).$$

В работе будет рассмотрено два определения наличия связи между временными рядами:

1. Липшицевость отображения φ :

$$\rho_{\mathbb{H}_X}(\varphi(\mathbf{y}_i), \varphi(\mathbf{y}_j)) \leq C \rho_{\mathbb{H}_Y}(\mathbf{y}_i, \mathbf{y}_j) \quad \mathbf{y}_i, \mathbf{y}_j \in \mathbb{H}_Y.$$

2. Высокая корреляция между $\widehat{\mathbf{x}}_0$ и \mathbf{x}_0

6.3. Вычислительный эксперимент

Было взято gif-изображение размером 128×128 и состоящее из 15 кадров. Далее продублировали сигнал 5 раз, чтобы у нас был набор кадров из 5 периодов.

После этого применяется HOSVD с core-тензорами разных размерностей $75 \times \text{hid-size} \times \text{hid-size}$ и восстанавливается аппроксимированный тензор. На рисунке 6.3 изображены исходное и восстановленные изображения. Из аппроксимированного тензора составляется траекторная матрица картинок. В качестве звукового сигнала Y используется синусоида с периодом в 15 шагов с добавленным нормальным шумом:

$$Y = \left\{ \sin \left(i * \frac{4\pi}{15} \right) + \xi_i \right\}_{i=1}^{75}, \text{ где } \xi_i \sim \mathcal{N}(0, 0.1^2)$$

К траекторным матрицам $\mathbf{H}_X, \mathbf{H}_Y$, полученным из аппроксимированного тензора \underline{X} и Y , применяется метод CCM.

Для каждого $i \in \{47, \dots, 61 = (75 - 15 + 1)\}$ вычисляется $\varphi(\mathbf{H}_Y[i])$. Для проверки на липшицевость в пр-вах \mathbb{H}_X и \mathbb{H}_Y использовались следующие метрики:

$$\rho_{\mathbb{H}_Y}(y_1, y_2) = \|y_1 - y_2\|_2, \quad \rho_{\mathbb{H}_X}(x_1, x_2) = \|x_1 - x_2\|_\infty.$$



Рис. 6.3. Результат применения HOSVD с hid-size, равным (а) 4; (б) 8; (в) 12; (д) 16; (е) 20; (ж) 32; (з) 64; (и) 128;

Установлено, что при $C = 100$ отображение φ является липшицевым (рисунки 6.5, 6.6).

Более того, корреляция между предсказаниями и исходными элементами траекторного пр-ва кадров превосходит 0.74 (рисунок 6.4).

Таким образом, можно заключить, что искомая причинная связь между временными рядами присутствует.

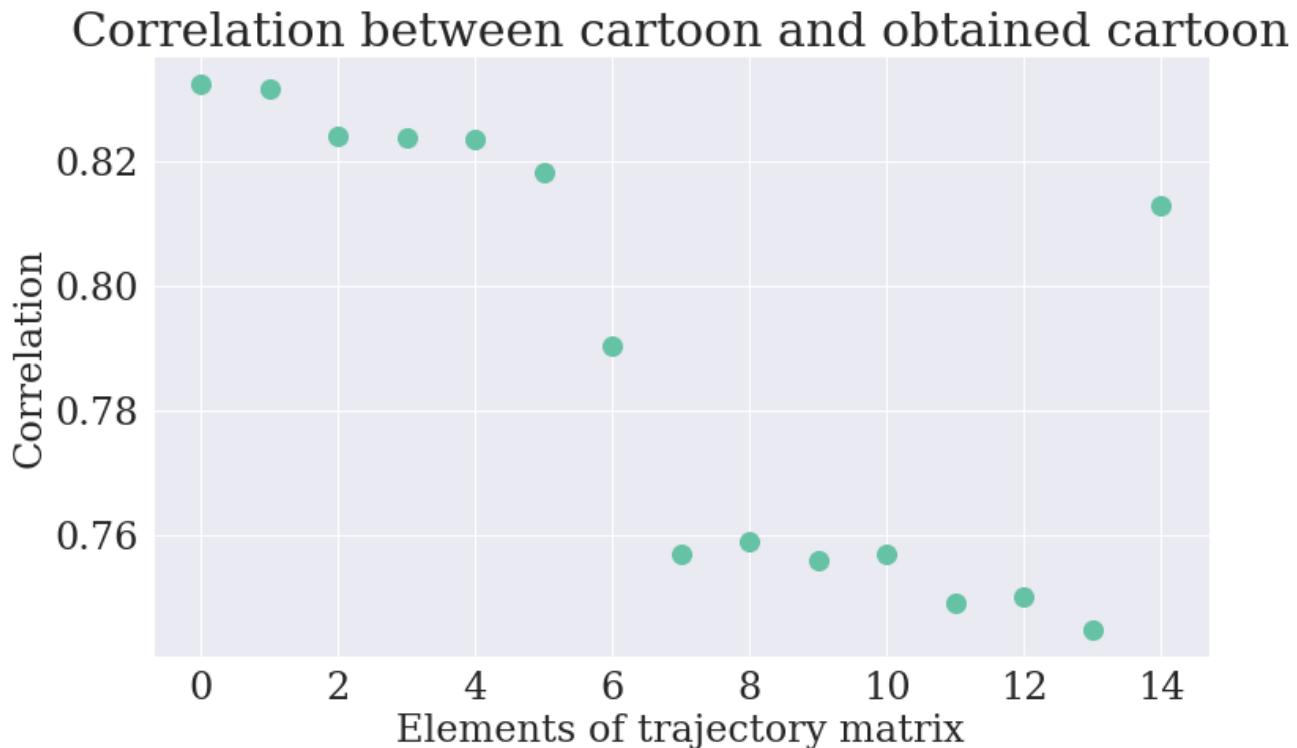


Рис. 6.4. Корреляция между элементами траекторной матрицы кадров и предсказаниями, полученными с помощью метода ближайших соседей (функции φ)

6.4. Литература

NULL

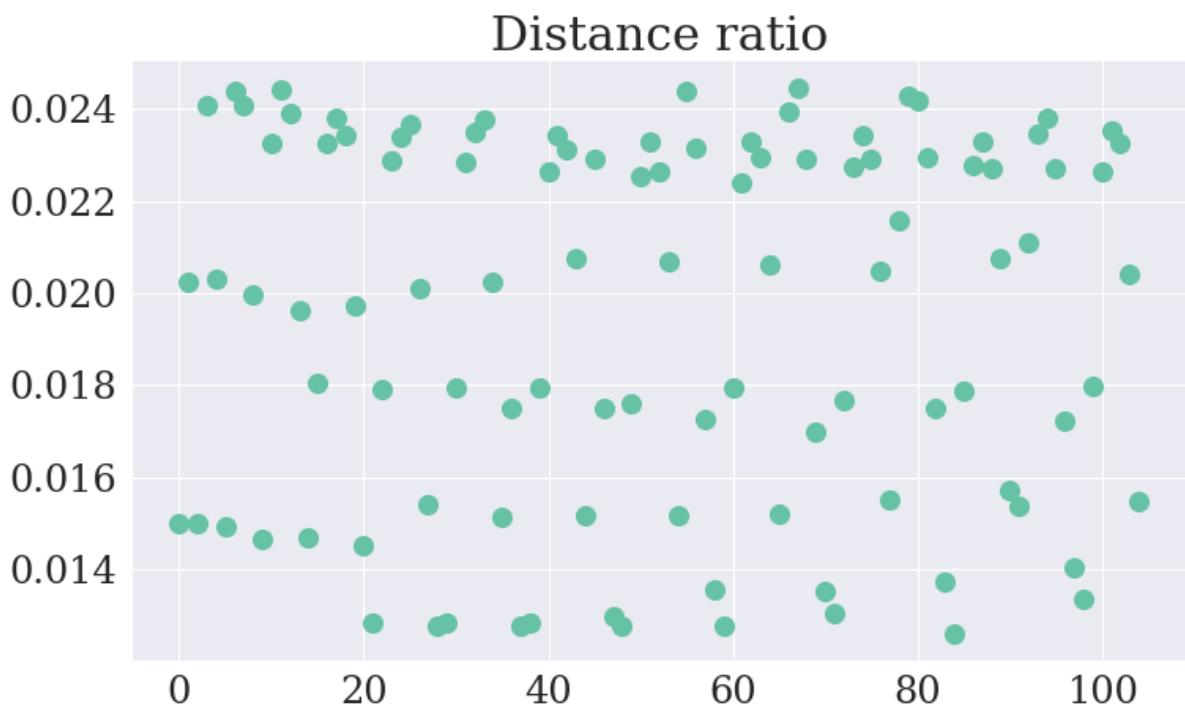


Рис. 6.5. Отношения расстояний (точечный график)

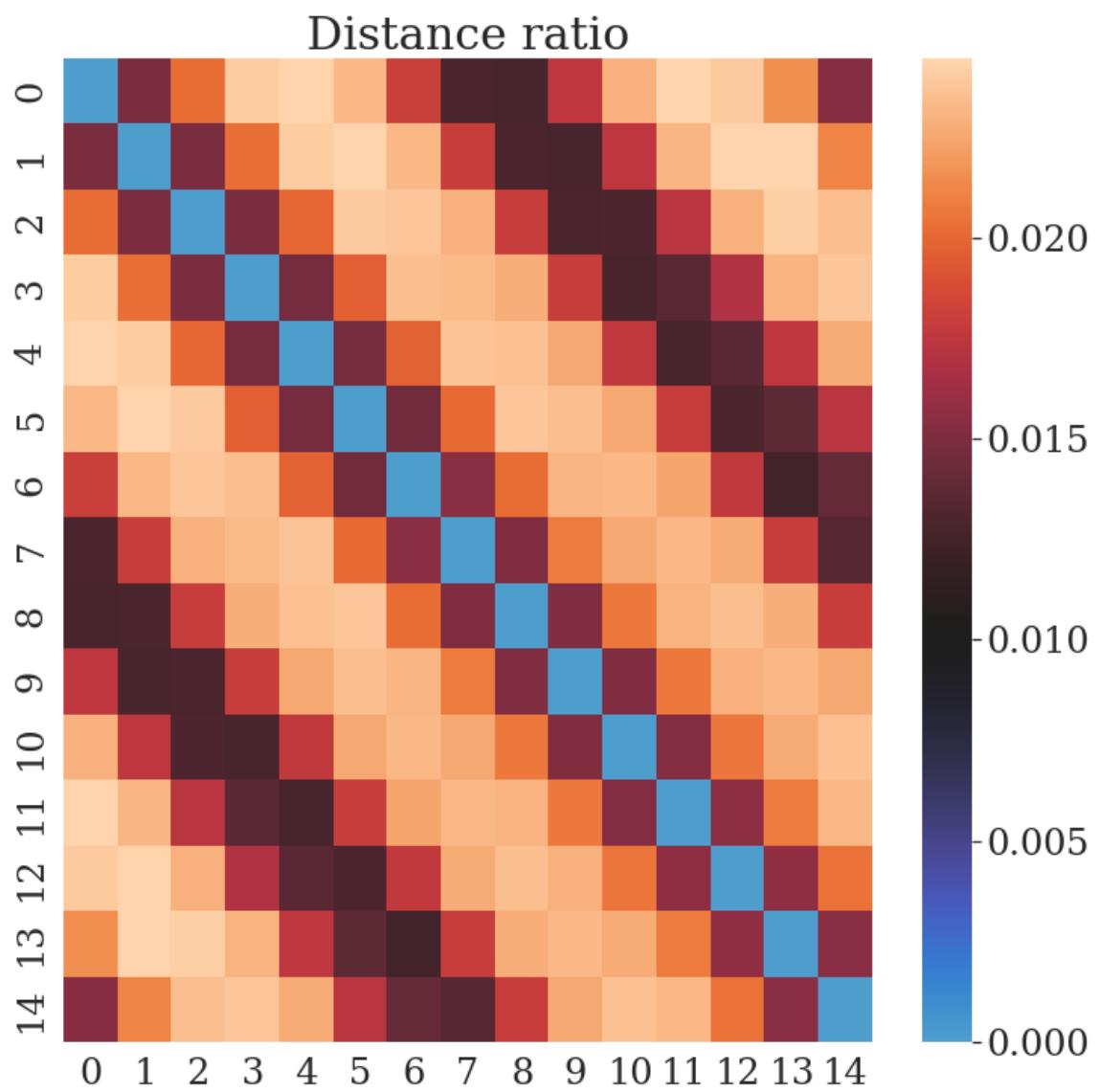


Рис. 6.6. Отношения расстояний (таблица)