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

version 0.01

# 1 Постановка задачи

Дан набор данных $X \in \mathbb{R}^{N \times D}$, состоящий из векторов $\{x_1, \ldots, x_N\}$ размерности $D$. Предполагается, что существует некоторое представление данных размерности $d$ (где $d < D$), которое вложено в исходное представление. Методы снижения размерности осуществляют преобразование выборки, получая новое представление $Y \in \mathbb{R}^{N \times d}$, как можно больше при этом сохраняя геометрию исходных данных.

# 2 Обзор существующих алгоритмов

## 2.1 PCA

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 $d$ ортогональных осей, вдоль которых данные имеют наибольшую дисперсию и осуществляет проекцию данных на них. Нахождение таких осей тесно связано с нахождением сингулярного разложения матрицы:

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

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

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

$$Y = X \cdot V_d, \qquad V_d \in \mathbb{R}^{D \times d}$$

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

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

## 2.2 Isomap

Данный метод снижения размерности базируется на построении попарных геодезических расстояний. По сути, геодезическое расстояние — расстояние, получаемое

между точками посредством суммирования расстояний между ближайшими соседями. Соответственно, единственным параметром метода является количество соседей $K$ для каждой точки. Алгоритм можно описать следующим образом:

1. Инициализировать матрицу $D$ попарных расстояний евклидовыми расстояниями между точками $||x_i - x_j||$
2. Если точка $i$ не является одним из $K$ ближайших соседей точки $j$, положить $D_{ij} = \infty$
3. Для каждой пары $i, j$ и для каждого $k = 1, \ldots, N$ заменить $D_{ij} = min(D_{ij}, D_{ik} + D_{kj})$ (Алгоритм Флойда-Уоршелла)
4. Решить задачу минимизации $\sum_{i,j}(D_{ij} - ||y_i - y_j||)^2 \to \min_{Y \in \mathbb{R}^{N \times d}}$

Отметим, что задача минимизации на последнем шаге является выпуклой и имеет глобальный оптимум.

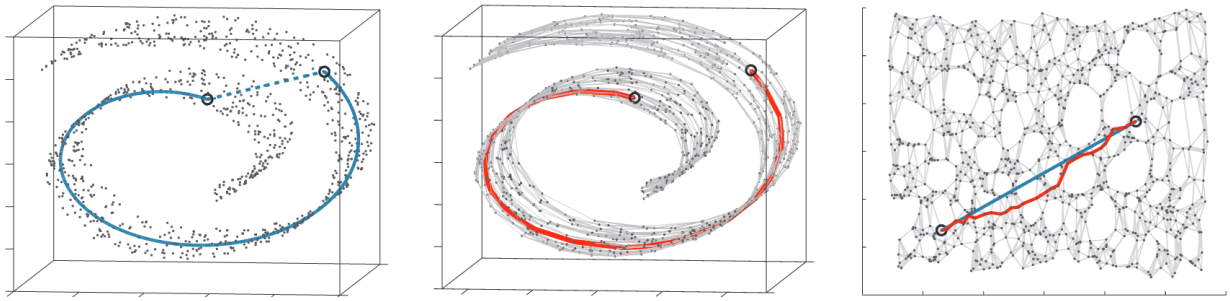

Рис. 1: Демонстрация работы Isomap на наборе трёхмерных данных "Swiss Roll". На первой картинке пунктиром изображено евклидово расстояние между двумя случайными точками, а сплошной линией — расстояние "по многообразию". По центру изображены расстояния между этими точками после шага 3. На последней картинке получено двумерное представление исходных данных

Несмотря на то, что алгоритм справляется отлично с нахождением структуры в искусственных наборах данных (как в примере выше), на практике он обычно не применим из-за высокой вычислительной сложности шага 3.

## 2.3 LLE

Locally Linear Embedding представляет точку как линейную комбинацию её ближайших соседей, получая таким образом матрицу весов $W$, где $W_{ij}$ — вклад $j$-го соседа в восстановление $i$-ой точки. Затем данные веса используются для построения низкоразмерного представления, сохраняющего соседство между точками.

Алгоритм сводится к двум задачам оптимизации:

$$||X - W \cdot X||_F^2 \to \min_{W \in \mathbb{R}^{N \times N}}$$

При следующих ограничениях:

- $W_{ij} = 0$, если $x_j$ не является одним из ближайших соседей $x_i$
- $\sum_j W_{ij} = 1$ для любой строки матрицы $W$

Последнее ограничение позволяет добиться инвариантности к сдвигам, поворотам и масштабированию точки и её соседей. Решив данную задачу, мы можем находить низкоразмерное представление, сохраняющее полученные веса:

$$||Y - W \cdot Y||^2 \to \min_{Y \in \mathbb{R}^{N \times d}}.$$

Отметим, что матрица $W$ является сильно разреженной, поэтому вычислительная сложность данного метода составляет лишь $O(pN^2)$, где $p$ — доля ненулевых элементов $W$. Однако, как следует из названия и описания, метод направлен на сохранение локальных особенностей данных, не всегда находя близкую к реальной глобальную геометрию данных.

## 2.4 Autoencoder

Автокодировщиком называется нейронная сеть, целевой вектор которой полагается равным её входу. То есть это модель, выучивающая приближение для тождественной функции. Накладывая различные ограничения, мы можем обнаружить структурные закономерности в данных. К примеру, такими ограничениями могут быть ограничение на число нейронов в скрытых слоях нейронной сети или ограничение разреженности на нейроны скрытого слоя.

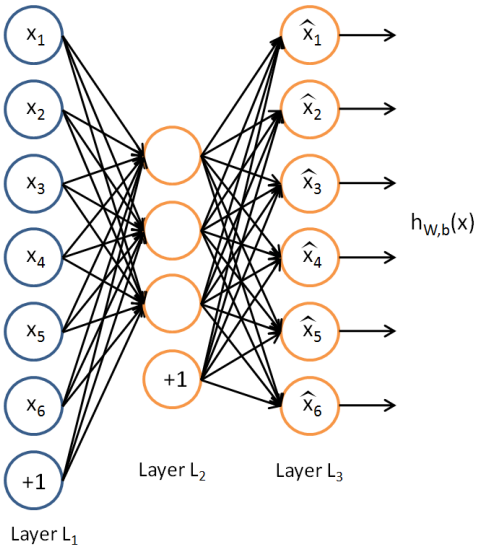


Рис. 2: Пример архитектуры автокодировщика

## 2.5 t-SNE

t-distributed Stochastic Neighborhood Embedding является методом снижения размерности, основным предназначением которого является визуализация данных (иначе говоря, предполагается $d = 2$ или $d = 3$).

Первым шагом метода является преобразование евклидовых расстояний между точками в условные вероятности $p_{j|i}$ того, что $x_i$ выберет $x_j$ в качестве своего соседа, если бы соседи выбирались с вероятностями, пропорциональными плотности вероятности нормального распределения с центром в $x_i$:

$$p_{j|i} = \frac{\exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-||x_i - x_k||^2/2\sigma_i^2)}.$$

Так как нам интересна только попарная схожесть, $p_{i|i}$ полагается равным нулю. Способ выбора $\sigma_i^2$ обговорим позже. Далее мы хотели бы максимизировать схожесть между условными вероятностями $p_{j|i}$ в исходном пространстве и условными вероятностями $q_{j|i}$ в пространстве новой размерности, способ вычисления которых будет дан ниже. Естественным для этого кажется минимизировать дивергенцию Кульбака-Лейблера (обратим внимание, что вероятности $p_{j|i}$ образуют распределение для каждого $i$).

В прародителе данного метода, SNE, для подсчёта условных вероятностей в новом пространстве также использовались гауссианы:

$$q_{j|i} = \frac{\exp(-||x_i - x_j||^2)}{\sum_{k \neq i} \exp(-||x_i - x_k||^2)},$$

однако авторы модификации, t-SNE, отмечают, что при таком способе подсчёта схожестей возникает проблема со скоплением точек в одном месте (crowding problem в оригинальной статье). К примеру, если точки равномерно распределены вокруг точки $i$ в десятимерном пространстве, и мы пытаемся моделировать расстояние от $i$ до других точек в двумерном пространстве, то для аккуратного отображения небольших расстояний в двумерном пространстве необходимо поместить все точки, расположенные на умеренном расстоянии, достаточно далеко от точки $i$. Но при использовании нормального распределения, основная вероятностная масса которого расположена рядом с центром, в низкоразмерном представлении все точки скучиваются, нивелируя зазоры между кластерами данных. Поэтому в t-SNE было предложено использовать распределение Стьюдента с одной степенью свободы (распределение Коши) для моделирования схожести между низкоразмерными представлениями точек:

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l} (1 + ||y_k - y_l||^2)^{-1}}.$$

Ещё одним отличием модификации от оригинального метода является переход к совместным вероятностям, добиваясь таким образом симметрии схожести между точками: для этого при подсчёте $q_{ij}$ нормировка происходит по всем точкам, а $p_{ij}$ полагается равным $\frac{p_{j|i} + p_{i|j}}{2}$.

Представление $Y$ находится градиентными методами, минимизируя дивергенцию Кульбака-Лейблера между полученными распределениями:

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}},$$

частная производная по $y_i$ которой имеет простой вид:

$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + ||y_i - y_j||^2)^{-1}.$$

Получаемое представление более терпимо к большим расстояниям между точками из-за тяжёлых хвостов распределения Коши, благодаря чему похожие точки оказываются рядом, а различные — далеко.

До сих пор мы не обговорили способ нахождения параметров $\sigma_i^2$. Так как плотность точек может варьироваться в разных участках пространства, кажется нелогичным брать единую дисперсию для каждой из гауссиан. Метод находит бинарным поиском значения $\sigma_i$, для которых распределение $P_i$ имеет перплексию, заданную пользователем:

$$Perp(P_i) = 2^{H(P_i)}, \qquad H(P_i) = -\sum_j p_{j|i} \log p_{j|i}.$$

Резюмируя, получаем следующий алгоритм:

**Data:** Набор данных $X$, перплексия $Perp$,
количество итераций $T$, длина шага $\eta$, моментум $\alpha(t)$.
**Result:** Низкоразмерное представление данных $Y^{(T)}$.
**begin**
  Посчитать близости $p_{j|i}$, имеющие заданную перплексию $Perp$;
  Положить $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2}$;
  Инициализировать $Y^{(0)} = \{y_1, \ldots, y_N\}$, сэмплируя из $\mathcal{N}(0, 10^{-4}I)$;
  **for** $t = 1$ *to* $T$ **do**
    Посчитать $q_{ij}$;
    Посчитать $\frac{\partial C}{\partial Y}$;
    Положить $Y^{(t)} = Y^{(t-1)} + \eta \frac{\partial C}{\partial Y} + \alpha(t)(Y^{(t-1)} - Y^{(t-2)})$;
  **end**
**end**

**Algorithm 1:** t-distributed Stochastic Neighborhood Embedding