

# LSML #8: Методы понижения размерности

5 июня 2017 г.

# Содержание

- 1 Мотивация
- 2 Проблемы LSH и обучение хешированию
- 3 Word embeddings
- 4 Autoencoders
- 5 t-SNE

## Постановка задачи

Дано:

- Обучающая выборка  $X^\ell = \{x_1, \dots, x_\ell\}$
- Ответы  $y_1, \dots, y_\ell$
- Признаковые описания  $x_i = (x_{i1}, \dots, x_{iD})$

Задача:

- Найти новые признаковые описания  $z_i = (z_{i1}, \dots, z_{id})$
- $d \ll D$
- Новые признаки должны удовлетворять определенным условиям

$$a(x) = \langle w, x \rangle + w_0$$

$$a(x) = \text{sign}(\langle w, x \rangle + w_0)$$

- Часто используются на разреженных признаках (категориальные, bag-of-words)
- Быстрое вычисление прогноза
- Чтобы учитывать нелинейные зависимости, нужно добавлять признаки высоких порядков
- Альтернатива: перевести объекты в новое признаковое пространство с помощью нелинейных преобразований

$$a(x) = \sum_{n=1}^N \alpha_n b_n(x),$$

$b_n(x)$  — решающие деревья небольшой высоты (2 – 10).

- Один из самых сильных методов машинного обучения
- Плохо работает на разреженных признаках
- Преобразование разреженных признаков в небольшое число плотных может существенно повысить качество

$$a(x) = \arg \max_y \sum_{k=1}^K w(x_{(k)})[y_{(k)} = y]$$

Примеры применений:

- Поиск наиболее похожих изображений в базе
- Распознавание лиц
- Медицинская диагностика на основе близости генетических профилей
- Один из базовых алгоритмов при блендинге

Особенности при больших объемах данных:

- В пространствах большой размерности евклидова метрика теряет смысл (проклятие размерности)
- Сложность поиска ближайшего соседа примитивным способом:  $O(\ell d)$
- Есть методы быстрого поиска соседей: kd-tree, ball-tree и т.д., которые при малых  $d$  имеют сложность  $O(\log \ell)$
- При  $d \rightarrow \infty$  сложность этих методов становится линейной:  $O(\ell d)$

Выход — понижение размерности.

# Содержание

- 1 Мотивация
- 2 Проблемы LSH и обучение хешированию
- 3 Word embeddings
- 4 Autoencoders
- 5 t-SNE

## Проблемы LSH

Идея LSH: построить хэш-функцию, которая будет давать одинаковые значения на близких объектах, и разные — на далеких.

Проблемы:

- Хэш-функции выбираются случайно, без учета распределения данных
- Могут потребоваться композиции из сотен хэш-функций, чтобы получить хорошее качество
- Хэш-функции известны лишь для небольшого числа метрик

Идея: обучать хэш-функции

## Обучение хэшированию

Будем строить хэши  $h(x) = (h_1(x), \dots, h_n(x))$ , на основе которых будет определяться близость объектов:

$$\rho(x, z) \approx \rho(h(x), h(z)).$$

Требования к хэшам:

- Дисперсия обучающей выборки после хэширования должна быть как можно больше
- Каждая хэш-функция центрирована:

$$\sum_{i=1}^{\ell} h_k(x_i) = 0$$

- Хэши ортогональны:

$$\sum_{i=1}^{\ell} \sum_{j=1}^{\ell} h_k(x_i)h_m(x_j) = 0, \quad k \neq m$$

## Обучение хэшированию

Будем искать хэши в виде

$$h_k(x) = \text{sign}\langle w_k, x \rangle$$

Матрица хэшей:

$$H = (h_k(x_i))_{i,k} = \text{sign } XW$$

Задача оптимизации:

$$\left\{ \begin{array}{l} \text{tr } H^T H \rightarrow \max_W \\ h_k(x_i) \in \{-1, +1\}, \quad k = 1, \dots, n; i = 1, \dots, \ell \\ \sum_{i=1}^{\ell} h_k(x_i) = 0, \quad k = 1, \dots, n \\ \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} h_k(x_i) h_m(x_j) = 0, \quad k \neq m \end{array} \right.$$

Слишком сложно из-за бинарности хэшей.

## Обучение хэшированию

Упростим:

$$h_k(x) = \langle w_k, x \rangle$$

Задача оптимизации:

$$\begin{cases} \operatorname{tr} W^T X^T X W \rightarrow \max_W \\ W^T W = I \end{cases}$$

Похоже на PCA!

Решение:  $w_1, \dots, w_k$  — собственные векторы  $X^T X$ , соответствующие наибольшим собственным значениям.

## Spectral Hashing

В PCA Hashing мы искали хэши, исходя из максимизации дисперсии, хотя исходная задача — сохранение расстояний в пространстве хэшей.

Исправимся:

$$\left\{ \begin{array}{l} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} v_{ij} \|h(x_i) - h(x_j)\|^2 = H^T (D - V) H \rightarrow \min_H \\ h_k(x_i) \in \{-1, +1\}, \quad k = 1, \dots, n; i = 1, \dots, \ell \\ \sum_{i=1}^{\ell} h_k(x_i) = 0, \quad k = 1, \dots, n \\ \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} h_k(x_i) h_m(x_j) = 0, \quad k \neq m \end{array} \right.$$

$$v_{ij} = \exp(-\|x_i - x_j\|^2 / \sigma^2)$$

$$d_{ii} = \sum_{j=1}^{\ell} v_{ij}$$

Задача NP-полная.

## Spectral Hashing

Построим релаксацию:

$$\left\{ \begin{array}{l} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} H^T (D - V) H \rightarrow \min_H \\ \sum_{i=1}^{\ell} h_k(x_i) = 0, \quad k = 1, \dots, n \\ \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} h_k(x_i) h_m(x_j) = 0, \quad k \neq m \end{array} \right.$$

Решение: собственные векторы  $D - V$ , соответствующие наименьшим собственным значениям

А как вычислять хэши для новых объектов?

## Spectral Hashing

Пусть объекты имеют распределение  $p(x)$ .

$$\left\{ \begin{array}{l} \int \|h(x_1) - h(x_2)\|^2 W(x_1, x_2) p(x_1) p(x_2) dx_1 dx_2 \rightarrow \min_{h(x)} \\ \int h(x) p(x) dx = 0 \\ \int h(x) h^T(x) p(x) dx = I \end{array} \right.$$

Решения — собственные функции (дискретизованного) оператора Лапласа-Бельтрами

# Содержание

- 1 Мотивация
- 2 Проблемы LSH и обучение хешированию
- 3 Word embeddings
- 4 Autoencoders
- 5 t-SNE

## Векторные представления слов

Хотим каждое слово представить как вещественный вектор:

$$w \rightarrow \vec{w} \in \mathbb{R}^d$$

Какие требования?

- Размерность  $d$  должна быть не очень велика
- Похожие слова должны иметь близкие векторы
- Арифметические операции над векторами должны иметь смысл

## word2vec

- Будем обучать представления слов так, чтобы они хорошо предсказывали свой контекст
- Выборка состоит из текстов, каждый представляет собой последовательность слов  $w_1, \dots, w_i, \dots, w_n$
- Контекст слова  $w_i$ :  $c(w_i) = (w_{i-k}, \dots, w_{i-1}, w_{i+1}, \dots, w_{i+k})$

Функционал для каждого текста:

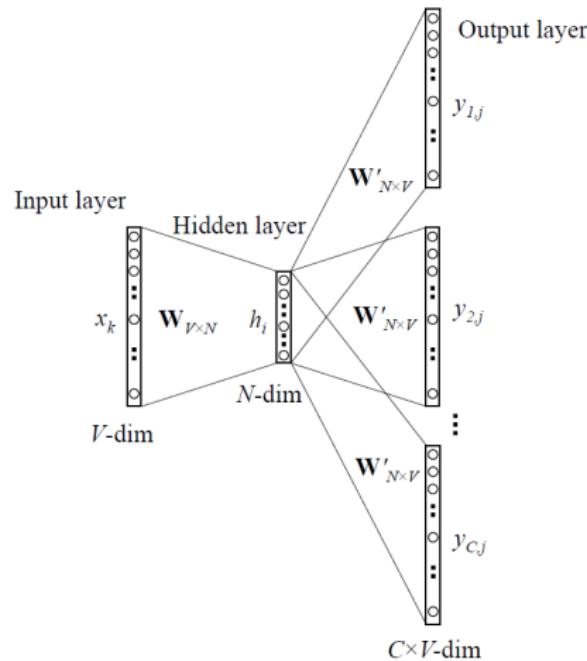
$$\sum_{i=1}^n \sum_{\substack{j=-k \\ j \neq 0}}^k \log p(w_{i+j} | w_i) \rightarrow \max,$$

где вероятность вычисляется через soft-max:

$$p(w_i | w_j) = \frac{\exp(\langle \vec{w}_i, \vec{w}_j \rangle)}{\sum_w \exp(\langle \vec{w}, \vec{w}_j \rangle)}$$

(сумма в знаменателе — по всем словам из словаря)

## Skip-gram модель:



## Свойства представлений

- Косинусное расстояние хорошо отражает схожесть слов по тематике (в зависимости от корпуса)
- $\vec{\text{king}} - \vec{\text{mān}} + \vec{\text{wōmān}} \approx \vec{\text{quēen}}$
- $\vec{\text{Moscow}} - \vec{\text{Russia}} + \vec{\text{England}} \approx \vec{\text{London}}$
- Перевод:  $\vec{\text{oñe}} - \vec{\text{uñō}} + \vec{\text{four}} \approx \vec{\text{quātro}}$
- Среднее представление по всем словам в тексте — хорошее признаковое описание

## word2vec как факторизация матрицы «слово-контекст»

В строках слова, в столбцах слова из контекста.

**Идея:** похожие слова имеют похожие векторы контекста.

	aardvark	computer	data	pinch	result	sugar	...
apricot	0	0	0	1	0	1	
pineapple	0	0	0	1	0	1	
digital	0	2	1	0	1	0	
information	0	1	6	0	4	0	

<https://levyomer.files.wordpress.com/2014/09/neural-word-embeddings-as-implicit-matrix-factorization.pdf>

## GloVe векторы

Будем факторизовать по-другому, минимизируя  $J$ :

$$J = \sum_{i,j} f(X_{ij})(w_i^T w_j + b_i + b_j - \log X_{ij})^2$$

$X_{ij}$  - частота встречаемости слова  $i$  в контексте со словом  $j$ .

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k ice)$	$1.9 \times 10^{-4}$	$6.6 \times 10^{-5}$	$3.0 \times 10^{-3}$	$1.7 \times 10^{-5}$
$P(k steam)$	$2.2 \times 10^{-5}$	$7.8 \times 10^{-4}$	$2.2 \times 10^{-3}$	$1.8 \times 10^{-5}$
$P(k ice)/P(k steam)$	8.9	$8.5 \times 10^{-2}$	1.36	0.96

<http://www-nlp.stanford.edu/pubs/glove.pdf>

# Содержание

- 1 Мотивация
- 2 Проблемы LSH и обучение хешированию
- 3 Word embeddings
- 4 Autoencoders
- 5 t-SNE

## Нейросети

Нейронная сеть принимает на вход вектор признаков и делает нелинейное преобразование на каждом слое:

$$v_1 = v_1(x) = \sigma(W_1 x)$$

$$v_2 = v_2(v_1) = \sigma(W_2 v_1)$$

...

$$v_n = v_n(v_{n-1}) = \sigma(W_n v_{n-1})$$

$v_n$  — вектор ответов (как правило, одно число в регрессии и бинарной классификации)

Идея:

- Сделать слои симметричными, центральный слой самый маленький по числу нейронов
- Последний слой по размеру совпадает с первым
- Требовать, чтобы выход как можно сильнее совпадал со входом
- Тогда внутренний слой будет давать сжатое описание объекта, по которому хорошо восстанавливаются исходные признаки

Пример архитектуры для MNIST:

784-1000-500-250-30-250-500-1000-784

## Автокодировщики

Экстремальный пример MNIST: 784-50-50-2-50-50-784

Онлайн демо: <http://cs.stanford.edu/people/karpathy/convnetjs/demo/autoencoder.html>

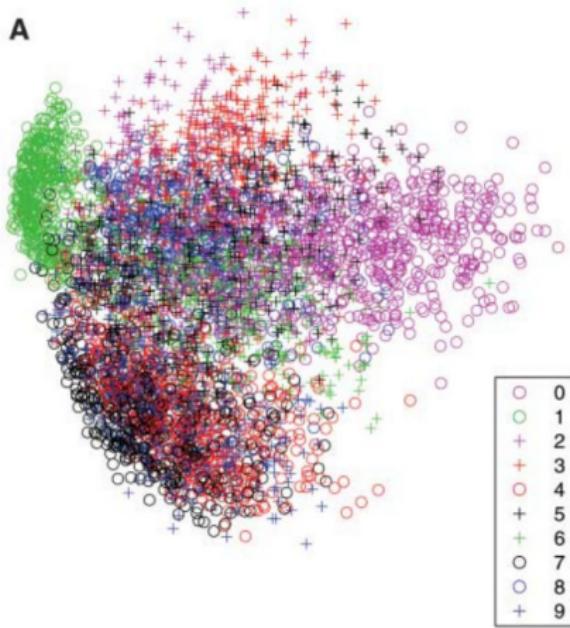


## Примеры использования

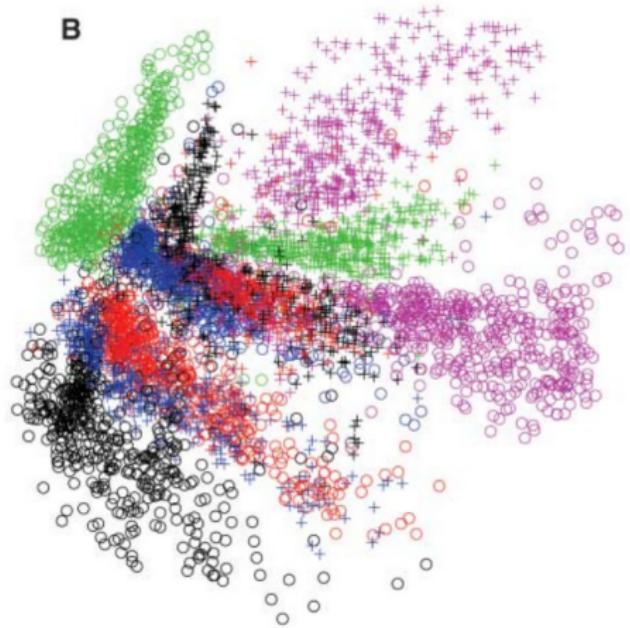
- Генерация новых признаков, нелинейно зависящих от исходных
- Поиск близких объектов по сжатым признакам (kNN)
- Визуализация (при размере внутреннего слоя 2–3)

## Примеры использования

A



B



MNIST. Слева PCA, справа автокодировщик

## Обучение автокодировщиков

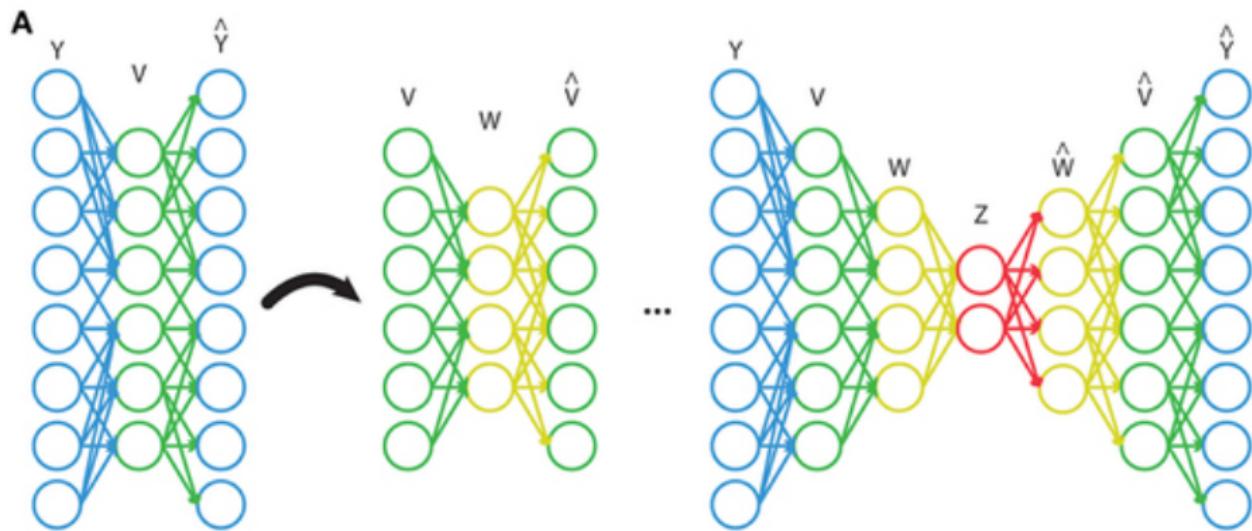
Проблема: обучение глубокой нейросети с помощью backpropagation затруднено.

Решение:

- Жадное обучение нейросети по слоям
- Обучение одного слоя — backpropagation или RBM
- Fine-tune всей сети в конце используя backpropagation

Автокодировщик — хороший способ инициализировать нейросеть для решения supervised-задачи.

# Обучение автокодировщиков



# Содержание

- 1 Мотивация
- 2 Проблемы LSH и обучение хешированию
- 3 Word embeddings
- 4 Autoencoders
- 5 t-SNE

## t-SNE

Популярный способ визуализации многомерных данных, содержащих сотни или тысячи признаков. <http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf>

Задача состоит в нахождении такого отображения исходных точек в двухмерное (обычно используется на практике, может быть большим), в котором исходно «близкие» точки будут «близки» и в новом пространстве признаков.

Как померять близость?

## t-SNE

Дано  $N$  объектов высокой размерности  $\mathbf{x}_1, \dots, \mathbf{x}_N$ .

t-SNE считает вероятности  $p_{ij}$ , которые пропорциональны похожести объектов  $\mathbf{x}_i$  и  $\mathbf{x}_j$  в исходном пространстве:

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2/2\sigma_i^2)}$$
$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

Значение  $\sigma_i$  будет разным для каждой точки. Оно выбирается так, чтобы точки в областях с большей плотностью имели меньшую  $\sigma_i$ .

Для этого используется оценка перплексии:

$$Perp(P_i) = 2^{-\sum_j p_{j|i} \log_2 p_{j|i}}$$

В чем ее смысл?

## t-SNE

- $Perp(P_i)$  - оценка эффективного количества «соседей» для точки  $\mathbf{x}_i$  (грубо)
- $Perp(P_i)$  - гиперпараметр метода (рекомендуется от 5 до 50)
- $\sigma_i$  определяется для каждого  $\mathbf{x}_i$  при помощи алгоритма бинарного поиска

Задача t-SNE состоит в нахождении такого отображения исходных точек в  $\mathbf{y}_1, \dots, \mathbf{y}_N$  меньшей размерности, при котором новые похожести  $q_{ij}$  между двумя точками  $\mathbf{y}_i$  и  $\mathbf{y}_j$  будут приближать похожести  $p_{ij}$ :

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k \neq i} (1 + \|\mathbf{y}_k - \mathbf{y}_i\|^2)^{-1}}$$

Как приближать  $q_{ij}$  к  $p_{ij}$ ?

## t-SNE

Положения точек  $y_i$  определяются в процессе минимизации дивергенции Кульбака-Лейблера:

$$KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

Минимизируется при помощи градиентного спуска:

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

**Проблемы:** медленный и не позволяет посчитать embedding для новой точки.

## t-SNE: примеры

Применим к задаче MNIST

<http://colah.github.io/posts/2014-10-Visualizing-MNIST>



A t-SNE plot of MNIST

## t-SNE: примеры

Применим к картинкам <https://github.com/genekogan/ofxTSNE>



## t-SNE: советы по использованию

Советы по использованию и интерактивные визуализации можно найти здесь: <http://distill.pub/2016/misread-tsne>

Быстрая реализация от Димы Ульянова:

<https://github.com/DmitryUlyanov/Multicore-TSNE>