Подавление шума на изображениях с использованием алгоритма K-SVD

Никита Дроздов
ВМК МГУ

9 января 2024 г.

Содержание

1. Разреженное кодирование
2. Orthogonal Matching Pursuit
3. K-SVD
4. Шумоподавление с помощью K-SVD

Постановка задачи разреженного кодирования

Пусть $D \in \mathbb{R}^{m \times K}$ – *словарь*, $y \in \mathbb{R}^m$ – *сигнал*.

Цель – найти разреженный вектор $\alpha \in \mathbb{R}^K$, т.ч. $||y - D\alpha||$ мала.

Можно рассматривать одну из двух постановок задачи разреженного кодирования:

$$\underset{\alpha}{\operatorname{argmin}} ||\alpha||_0 \quad subject\ \ to \quad ||y - D\alpha|| \leq \epsilon, \tag{1}$$

$$\underset{\alpha}{\operatorname{argmin}} ||y - D\alpha|| \quad subject\ \ to \quad ||\alpha||_0 \leq T. \tag{2}$$

Здесь $|| \cdot ||_0$ – #ненулевых компонент вектора (l_0 - "норма").

Для N входных сигналов – y заменяем на матрицу $Y \in \mathbb{R}^{m \times N}$, α на $A \in \mathbb{R}^{K \times N}$.

На практике словарь часто бывает переопределен ($m < K$).

Sparse dictionary learning (SDL)

В постановках 1 и 2 можно оптимизировать по α и D сразу (задача *sparse dictionary learning*).
Классические алгоритмы SDL итеративно повторяют два шага:

1. *Разреженное кодирование* сигналов с текущим словарем D;
2. *Обновление элементов словаря* по некоторому алгоритму.

Для задания такого алгоритма нужно задать алгоритмы для обеих шагов.

Sparse dictionary learning (SDL)

В постановках 1 и 2 можно оптимизировать по α и D сразу (задача *sparse dictionary learning*).
Классические алгоритмы SDL итеративно повторяют два шага:

1. *Разреженное кодирование* сигналов с текущим словарем D;
2. *Обновление элементов словаря* по некоторому алгоритму.

Для задания такого алгоритма нужно задать алгоритмы для обеих шагов.

Рассмотрим подробно этап разреженного кодирования.

Содержание

1. Разреженное кодирование
2. Orthogonal Matching Pursuit
3. K-SVD
4. Шумоподавление с помощью K-SVD

Matching Pursuit

Построим *жадный* алгоритм решения задачи разреженного кодирования (1 или 2).

Пусть $f \in E = (\mathbb{R}^m, \langle\cdot\rangle)$, $D = \{d_i\}_{i=1}^K$ – словарь, причем $||d_i|| = 1 \, \forall i$.
Ортопроекция $f := R^0 f$ на произвольный $d_{\gamma_0} \in D$:

$$f = \langle f, d_{\gamma_0} \rangle d_{\gamma_0} + R^1 f$$

В силу ортогональности: $||f||^2 = |\langle f, d_{\gamma_0} \rangle|^2 + ||R^1 f||^2 \Rightarrow$ для локальной минимизации $||R^1 f||$ нужно выбрать $d_{\gamma_0} = \text{argmax}_{d \in D} |\langle f, d \rangle|$.

Применим данную процедуру к $R^1 f$ и следующим $p-2$ невязкам:

$$f = \sum_{n=0}^{p-1} \langle R^n f, d_{\gamma_n} \rangle d_{\gamma_n} + R^p f. \tag{3}$$

Matching Pursuit

Algorithm 1 Matching Pursuit

Require: dictionary $D \in \mathbb{R}^{m \times K}$ with elements d_i, signal $f \in \mathbb{R}^m$, stop condition
Ensure: sparse coefficient vector $\alpha \in \mathbb{R}^K$

$R^0 \leftarrow f$
$n \leftarrow 0$
$\alpha \leftarrow \mathbf{0} \in \mathbb{R}^K$
while not stop condition **do**
  $d_{\gamma_n} \leftarrow \text{argmax}_{d_k \in D} |\langle R^n, d_k \rangle|$
  $\alpha_{\gamma_n} \leftarrow \langle R^n, d_{\gamma_n} \rangle$
  $R^{n+1} \leftarrow R^n - \alpha_{\gamma_n} d_{\gamma_n}$
  $n \leftarrow n + 1$
end while

Matching Pursuit – *жадный* алгоритм решения задачи разреженного кодирования.

В силу (3) $||R^n||$ монотонно убывает $\Rightarrow$ локальная сходимость гарантированна.

Orthogonal Matching Pursuit (OMP)

Algorithm 1 Orthogonal Matching Pursuit

Require: dictionary $D \in \mathbb{R}^{m \times K}$ with elements d_i, signal $f \in \mathbb{R}^m$, stop condition
Ensure: sparse coefficient vector $\alpha \in \mathbb{R}^K$

$R^0 \leftarrow f$
$n \leftarrow 0$
$\alpha \leftarrow \mathbf{0} \in \mathbb{R}^K$
while not stop condition **do**
  $d_{\gamma_n} \leftarrow \underset{d_k}{\operatorname{argmax}} |\langle R^n, d_k \rangle|$
  $D_n \leftarrow [d_{\gamma_0}, \ldots, d_{\gamma_n}] \in \mathbb{R}^{m \times (n+1)}$
  $\alpha[\gamma_0, \gamma_1, \ldots, \gamma_n] \leftarrow \underset{a \in \mathbb{R}^{n+1}}{\operatorname{argmin}} ||f - D_n a||_2$
  $R^{n+1} \leftarrow f - D\alpha$
  $n \leftarrow n + 1$
end while

OMP – модификация MP с дополнительным обновлением *всех* коэффицентов.

Это более затратно с точки зрения вычислений, но дает лучший результат.

Содержание

1. Разреженное кодирование
2. Orthogonal Matching Pursuit
3. K-SVD
4. Шумоподавление с помощью K-SVD

K-SVD: введение

K-SVD – обобщение метода кластеризации K-Means на задачу sparse dictionary learning.

Входные данные: матрица сигналов $Y \in \mathbb{R}^{m \times N}$, начальный словарь $D_{init} \in \mathbb{R}^{m \times K}$.
Выход: матрица коэффицентов $A \in \mathbb{R}^{K \times N}$, оптимальный словарь D_{final}.
Общий алгоритм:

1. Разреженное кодирование с помощью любого метода (мы используем OMP);
2. Обновление словаря *поэлементно* путем последовательных вычислений SVD;
3. Повторять п.1-2 нужное число раз.

K-SVD: введение

K-SVD – обобщение метода кластеризации K-Means на задачу sparse dictionary learning.

Входные данные: матрица сигналов $Y \in \mathbb{R}^{m \times N}$, начальный словарь $D_{init} \in \mathbb{R}^{m \times K}$.
Выход: матрица коэффицентов $A \in \mathbb{R}^{K \times N}$, оптимальный словарь D_{final}.
Общий алгоритм:

1. Разреженное кодирование с помощью любого метода (мы используем OMP);
2. Обновление словаря *поэлементно* путем последовательных вычислений SVD;
3. Повторять п.1-2 нужное число раз.

П.1. рассмотрели ранее, рассмотрим подробнее п.2.

K-SVD: обновление словаря

Пусть A и D фиксированы. Рассмотрим элемент словаря d_k и строку матрицы A α_T^k.
Тогда

$$||Y - DA||_F^2 = \left|\left| Y - \sum_{j=1}^{K} d_j \alpha_T^j \right|\right|_F^2 = \left|\left| \left(Y - \sum_{j \neq k} d_j \alpha_T^j \right) - d_k \alpha_T^k \right|\right|_F^2 = ||E_k - d_k \alpha_T^k||_F^2. \quad (4)$$

Хотим минимизировать (4) по d_k и α_T^k.

K-SVD: обновление словаря

Пусть A и D фиксированы. Рассмотрим элемент словаря d_k и строку матрицы A α_T^k.
Тогда

$$||Y - DA||_F^2 = \left|\left| Y - \sum_{j=1}^{K} d_j \alpha_T^j \right|\right|_F^2 = \left|\left| \left(Y - \sum_{j \neq k} d_j \alpha_T^j \right) - d_k \alpha_T^k \right|\right|_F^2 = ||E_k - d_k \alpha_T^k||_F^2. \quad (4)$$

Хотим минимизировать (4) по d_k и α_T^k.
Делать это с помощью прямого вычисления SVD **нельзя!**

K-SVD: обновление словаря

Пусть A и D фиксированы. Рассмотрим элемент словаря d_k и строку матрицы A α_T^k.
Тогда

$$||Y - DA||_F^2 = \left|\left| Y - \sum_{j=1}^{K} d_j \alpha_T^j \right|\right|_F^2 = \left|\left| \left(Y - \sum_{j \neq k} d_j \alpha_T^j \right) - d_k \alpha_T^k \right|\right|_F^2 = ||E_k - d_k \alpha_T^k||_F^2. \quad (4)$$

Хотим минимизировать (4) по d_k и α_T^k.
Делать это с помощью прямого вычисления SVD **нельзя!** В этом случае мы не учитываем ограничение на разреженность коэффицентов.

K-SVD: обновление словаря

Пусть A и D фиксированы. Рассмотрим элемент словаря d_k и строку матрицы A α_T^k.
Тогда

$$||Y - DA||_F^2 = \left|\left| Y - \sum_{j=1}^{K} d_j \alpha_T^j \right|\right|_F^2 = \left|\left| \left(Y - \sum_{j \neq k} d_j \alpha_T^j \right) - d_k \alpha_T^k \right|\right|_F^2 = ||E_k - d_k \alpha_T^k||_F^2. \quad (4)$$

Хотим минимизировать (4) по d_k и α_T^k.
Делать это с помощью прямого вычисления SVD **нельзя!** В этом случае мы не учитываем ограничение на разреженность коэффицентов.
Идея: обновлять только ненулевые элементы α_T^k.

K-SVD: обновление словаря

Пусть A и D фиксированы. Рассмотрим элемент словаря d_k и строку матрицы A α_T^k.
Тогда

$$||Y - DA||_F^2 = \left|\left| Y - \sum_{j=1}^{K} d_j \alpha_T^j \right|\right|_F^2 = \left|\left| \left(Y - \sum_{j \neq k} d_j \alpha_T^j \right) - d_k \alpha_T^k \right|\right|_F^2 = ||E_k - d_k \alpha_T^k||_F^2. \qquad (4)$$

Хотим минимизировать (4) по d_k и α_T^k.

$\boldsymbol{I}_k := \operatorname{supp} \alpha_T^k = \{i : \alpha_T^k(i) \neq 0\}$.

$\widetilde{E_k}$, $\widetilde{\alpha}_T^k$ – составлены из столбцов соотв. матриц с индексами из $\boldsymbol{I}_k$.

Пара $\{d_k, \widetilde{\alpha}_T^k\}$, доставляющая минимум выражению $||\widetilde{E_k} - d_k \widetilde{\alpha}_T^k||_F^2$, находится с помощью SVD.

K-SVD: полный алгоритм

Algorithm 2 K-SVD

Require: initial dictionary $D_{init} \in \mathbb{R}^{m \times K}$ with elements d_i, signals $Y \in \mathbb{R}^{m \times N}$, stop condition for OMP, maximum iterations n

Ensure: sparse coefficient matrix $A \in \mathbb{R}^{K \times N}$, final dictionary D_{final}

$iter \leftarrow 0$

$D \leftarrow D_{init}$

while $iter < n$ **do**

  $A \leftarrow OMP(D, Y, stopcond)$

  $k \leftarrow 1$

  while $k \leq K$ **do**

    $D[k] \leftarrow \mathbf{0}$

    $E_k \leftarrow Y - DA$

    $\boldsymbol{I}_k \leftarrow \text{supp}\, \alpha_T^k$

    $\widetilde{E_k}, \widetilde{\alpha_T^k} \leftarrow E_k$ and α_T^k restricted by $\boldsymbol{I}_k$

    $\{d_k, \widetilde{\alpha_T^k}\} \leftarrow \text{argmin}_{d, \alpha_T} ||\widetilde{E_k} - d\alpha_T||_2^2$

    $D[k] \leftarrow d_k$

    $\alpha_T^k \leftarrow \widetilde{\alpha_T^k}$ zero-padded according to $\boldsymbol{I}_k$

    $k \leftarrow k + 1$

  end while

  $iter \leftarrow iter + 1$

end while

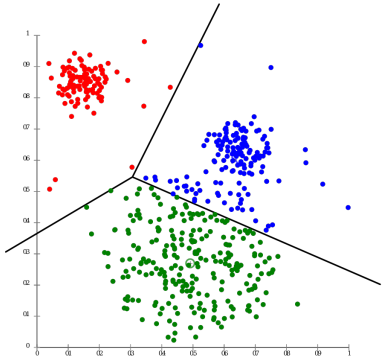
$D_{final} \leftarrow D$

K-SVD и K-Means

K-Means:

1. Каждый элемент кодируется ближайшим центроидом, т.о. образуются кластера;
2. Центроиды пересчитываются как среднее по своему кластеру;
3. Повторять п.1-2 до сходимости.

K-Means – частный случай K-SVD (*покажите!*)



Содержание

1. Разреженное кодирование
2. Orthogonal Matching Pursuit
3. K-SVD
4. Шумоподавление с помощью K-SVD

Постановка задачи

Пусть $\boldsymbol{x}_0 \in \mathbb{R}^N$ – ч/б изображение, $\boldsymbol{w} \in \mathbb{R}^N$ – аддитивный гауссовский шум, т.е. $\boldsymbol{w} \sim N(0, \sigma^2 I)$.

Задача – по зашумленному изображению $\boldsymbol{y} = \boldsymbol{x}_0 + \boldsymbol{w}$ восстановить изображение $\widehat{\boldsymbol{x}}_0$, близкое к $\boldsymbol{x}_0$ по метрике *Peak Signal-to-Noise Ratio (PSNR):*

$$PSNR(\boldsymbol{x}_0, \widehat{\boldsymbol{x}}_0) = 20 \cdot \log_{10}\left(\frac{\sqrt{N} \cdot 255}{||\boldsymbol{x}_0 - \widehat{\boldsymbol{x}}_0||_2 + \varepsilon}\right)$$

*255 – максимум диапазона интенсивностей 8-битных изображений.

Алгоритм шумоподавления

Рассматриваем все фрагменты изображения $\boldsymbol{y}$ размера $\sqrt{n} \times \sqrt{n}$.
$R_{ij}\boldsymbol{y}$ – фрагмент изображения $\boldsymbol{y}$ с координатой левого верхнего пикселя (i, j), $R_{ij} \in \mathbb{R}^{n \times N}$.
Алгоритм:

1. Получаем множество фрагментов зашумленного изображения $\{R_{ij}\boldsymbol{y}\} \rightarrow$ пропускаем через K-SVD $\rightarrow$ получаем множество $\{\boldsymbol{D}\alpha_{ij}\}$ очищенных от шума фрагментов;
2. Восстанавливаем изображение $\widehat{\boldsymbol{x}}_{\boldsymbol{\lambda}}$, решая задачу минимизации:

$$\widehat{\boldsymbol{x}}_{\boldsymbol{\lambda}} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \lambda ||\boldsymbol{y} - \boldsymbol{x}||_2^2 + \sum_{i,\,j} ||\boldsymbol{D}\alpha_{ij} - R_{ij}\boldsymbol{x}||_2^2. \tag{5}$$

Аналитическое решение (5): $\widehat{\boldsymbol{x}}_{\boldsymbol{\lambda}} = \left(\lambda I + \sum\limits_{i,\,j} R_{ij}^T R_{ij}\right)^{-1} \left(\lambda \boldsymbol{y} + \sum\limits_{i,\,j} R_{ij}^T \boldsymbol{D}\alpha_{ij}\right).$

Алгоритм шумоподавления

Аналитическое решение (5): $\widehat{\boldsymbol{x}}_{\boldsymbol{\lambda}} = \left(\lambda I + \sum\limits_{i,j} R_{ij}^T R_{ij}\right)^{-1} \left(\lambda \boldsymbol{y} + \sum\limits_{i,j} R_{ij}^T \boldsymbol{D}\alpha_{ij}\right).$

Формула выглядит сложно, но на самом деле *все очень просто!*

- При $\lambda = 0$ изображение $\widehat{\boldsymbol{x}}_0$ получается вставкой всех фрагментов из $\{\boldsymbol{D}\alpha_{ij}\}$ в соответствующие места с усреднением пикселей в пересекающихся областях;
- При $\lambda > 0$ каждый пиксель на $\widehat{\boldsymbol{x}}_{\boldsymbol{\lambda}}$ – взвешенная сумма пикселей с $\widehat{\boldsymbol{x}}_0$ и $\boldsymbol{y}$; больше λ – больше вклад зашумленного изображения.

λ надо подбирать аккуратно, особенно при сильном шуме!
Можно выбрать $\lambda = C/\sigma$ (в одной из статей $C = 30$).

Примеры тестовых изображений

(a) Lenna, 512x512

(b) Peppers, 512x512

(c) Beacon, 512x384

Сравнение K-SVD с NLM и TV

$\sigma/PSNR$	K-SVD	NLM	TV
5/34.15	37.58	36.31	34.96
10/28.13	33.81	32.46	32.01
15/24.61	31.77	30.65	29.80
20/22.11	30.29	29.50	28.48
25/20.17	29.08	28.53	27.26
50/14.15	24.93	24.30	23.92

PSNR в первой колонке – между $\boldsymbol{x}_0$ и $\boldsymbol{y}$, в остальных – между $\boldsymbol{x}_0$ и $\widehat{\boldsymbol{x}}_0$. Все значения усреднены по всему тестовому набору (8 изображений).

Сравнили K-SVD с алгоритмами:

- *Total Variation (TV)* – минимизация регуляризирующего функционала со специальным стабилизатором;
- *Non-Local Means (NLM)* – усреднение по достаточно большой окрестности каждого пикселя с весовой функцией.

Сравнение K-SVD с NLM и TV

$\sigma/PSNR$	K-SVD	NLM	TV
5/34.15	37.58	36.31	34.96
10/28.13	33.81	32.46	32.01
15/24.61	31.77	30.65	29.80
20/22.11	30.29	29.50	28.48
25/20.17	29.08	28.53	27.26
50/14.15	24.93	24.30	23.92

PSNR в первой колонке – между $\boldsymbol{x}_0$ и $\boldsymbol{y}$, в остальных – между $\boldsymbol{x}_0$ и $\widehat{\boldsymbol{x}}_0$. Все значения усреднены по всему тестовому набору (8 изображений).

Сравнили K-SVD с алгоритмами:

- *Total Variation (TV)* – минимизация регуляризирующего функционала со специальным стабилизатором;
- *Non-Local Means (NLM)* – усреднение по достаточно большой окрестности каждого пикселя с весовой функцией.

В экспериментах K-SVD показал себя лучше, чем два классических алгоритма шумоподавления. Но у K-SVD есть свои недостатки.

Главный недостаток K-SVD

K-SVD *очень медленный.*

В моих экспериментах:

- NLM работает менее секунды, наивный TV – 10 сек.;
- K-SVD — *0.5 - 4 минуты.*

В эпоху глубоких нейросетей это слишком долго!

Однако алгоритм все еще можно использовать offline.

Примеры работы K-SVD, NLM и TV

(d) K-SVD, PSNR 30.71

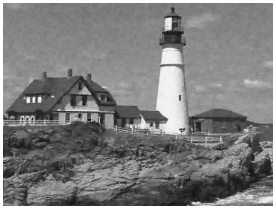
(e) NLM, PSNR 30.33

(f) TV, PSNR 29.55

(g) Reference

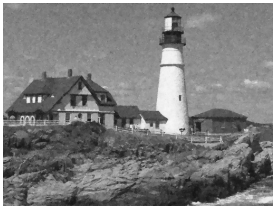
(h) Noisy, PSNR 20.24

Примеры работы K-SVD, NLM и TV



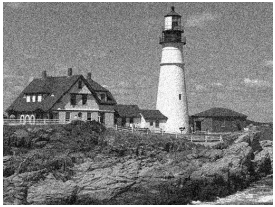
(i) K-SVD, PSNR 27.16

(j) NLM, PSNR 26.42



(k) TV, PSNR 26.21

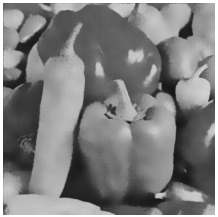
(l) Reference



(m) Noisy, PSNR 20.29

Примеры работы K-SVD, NLM и TV

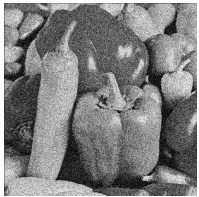
(n) K-SVD, PSNR 26.73



(o) NLM, PSNR 26.05

(p) TV, PSNR 25.88

(q) Reference



(r) Noisy, PSNR 14.72

Результат работы K-SVD на всем тестовом наборе

$\sigma/PSNR$	fingerprint	houses	injun girl	beacon	boats	barbara	lenna	peppers
5/34.15	36.60	35.94	38.47	37.18	39.18	38.15	37.57	37.57
10/28.13	32.38	31.57	34.76	32.48	35.52	34.42	34.72	34.67
15/24.61	30.08	29.37	32.71	29.99	33.37	32.37	33.07	33.18
20/22.11	28.49	27.87	31.26	28.43	31.78	30.75	31.81	31.95
25/20.17	27.32	26.61	30.07	27.20	30.50	29.38	30.71	30.82
50/14.15	23.71	22.30	25.47	23.41	26.16	24.94	26.76	26.70

Результаты работы K-SVD на тестовом наборе изображений. Значения PSNR для каждого изображения усреднены по 5 запускам алгоритма с различными реализациями шума. В первой колонке – стандартное отклонение шума и PSNR между y и x_0.

Заключение

Выводы:

- K-SVD хорошо справляется с подавлением шума, однако на данный момент существуют более эффективные алгоритмы (в основном нейросетевые);
- K-SVD проигрывает многим классическим алгоритмам в вычислительной сложности;
- Тем не менее, алгоритм имеет большое теоретическое значение и может использоваться в offline-задачах.

Материалы:

- Диск со статьями: https://clck.ru/36n9Sj
- Репозиторий с кодом: https://clck.ru/36n9tG

Спасибо за внимание!