Квантизация больших языковых моделей в переопределённом базисе

Гладков Андрей

30 декабря 2024

Содержание

Идея

Языковые модели содержат большое количество линейных слоёв, которые можно квантизовать для уменьшения их размера и ускорения инференса.
Предположим, что возможно разложение:

$$x \approx u + Qv, \quad x, u, v \in \mathbb{R}^n, Q \in \mathbb{R}^{n \times n} - \text{ортогональная, } \|u\|_\infty \text{ и } \|v\|_\infty \text{ малы.}$$

Тогда есть надежда, что распределение значений этих векторов будет хорошо квантизоваться. Значения u, v можно кластеризовать и заменить каждое из них на среднее по кластеру. Чем более компактными получаются кластеры, тем ниже ошибка аппроксимации.

Цель

- Реализовать квантизацию Кашина.
- Провести эксперимент на случайных матрицах.
- Произвести эксперимент на реальной языковой модели.

Жадный алгоритм для нахождения разложения $x \approx u + Qv$

Algorithm 1 Vector Decomposition Kashin Algorithm

Input: Vector $x \in \mathbb{R}^n$, Orthogonal matrix Q, Tolerance $\varepsilon > 0$

Output: Vectors $u, \hat{v} \in \mathbb{R}^n$ such that $x \approx u + \hat{v} = u + Qv$, and both u and v have small infinity norm.

Initialize $u \leftarrow 0^n, \hat{v} \leftarrow 0^n$

Define projection $\pi_x(y) \coloneqq \dfrac{x^\top y}{\|y\|_2^2} \cdot y$

while $\|x - u - \hat{v}\| \geq \varepsilon$ **do**
  if $\|x\|_1 > \|Q^T x\|_1$ **then**
    $\pi \leftarrow \pi_x(\text{Sign}(x))$
    $u \leftarrow u + \pi$
  else
    $\pi \leftarrow \pi_x(Q\text{Sign}(Q^T x))$
    $\hat{v} \leftarrow \hat{v} + \pi$
  end if
  $x \leftarrow x - \pi$
end while

Return: $x, u, \hat{v}$

Матричная версия для разложения $X \approx U + Q_1 V Q_2^T$

Algorithm 2 Matrix Decomposition Kashin Algorithm

Input: Matrix $X \in \mathbb{R}^{m \times n}$, Orthogonal matrices Q_1, Q_2, Tolerance $\varepsilon > 0$

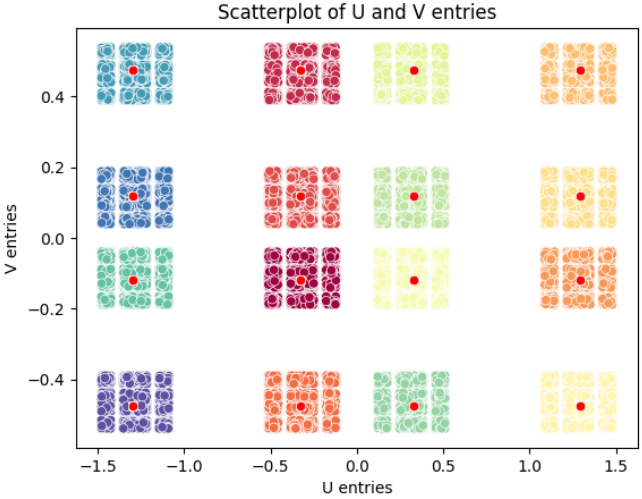
Output: Matrices $U, \hat{V} \in \mathbb{R}^{m \times n}$, such that $X \approx U + \hat{V} = U + Q_1 V Q_2^T$ and both $\mathrm{Vec}(U)$ and $\mathrm{Vec}(V)$ have small infinity norm.

Initialize $U \leftarrow 0^{m \times n}, \hat{V} \leftarrow 0^{m \times n}$

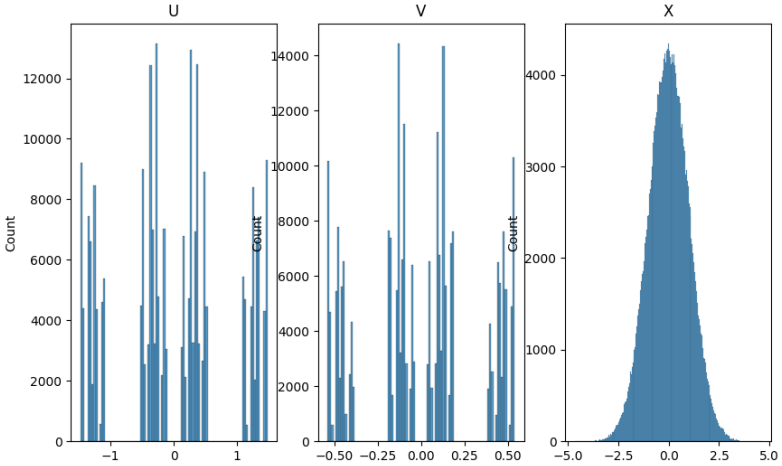
Define projection $\pi_X(Y) := \dfrac{\mathrm{Vec}(X)^\top \mathrm{Vec}(Y)}{\|\mathrm{Vec}(Y)\|_2^2} \cdot Y$

while $\|X - U - \hat{V}\| \geq \varepsilon$ **do**
  $Y \leftarrow Q_1^T X Q_2$
  if $\|\mathrm{Vec}(X)\|_1 > \|\mathrm{Vec}(Y)\|_1$ **then**
    $\pi \leftarrow \pi_X(\mathrm{Sign}(X))$
    $U \leftarrow U + \pi$
  else
    $\pi \leftarrow \pi_X(Q_1 \mathrm{Sign}(Y) Q_2^\top)$
    $\hat{V} \leftarrow \hat{V} + \pi$
  end if
  $X \leftarrow X - \pi$
end while
Return: $X, U, \hat{V}$

Результаты для разложения случайной матрицы $X_{500\times500}$



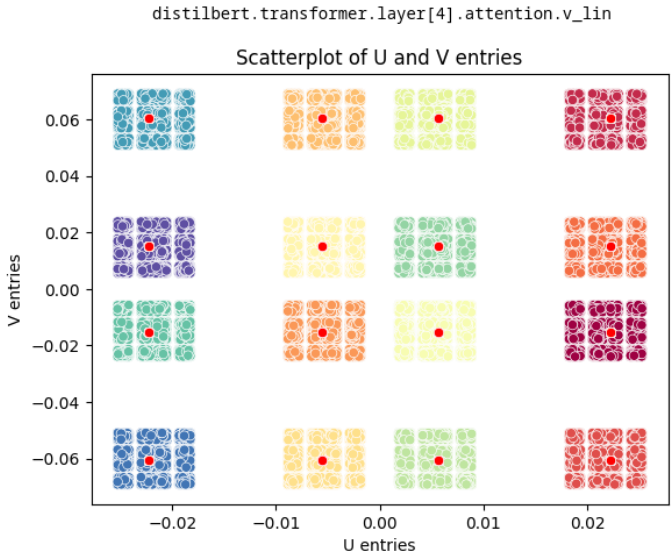
Результаты для разложения случайной матрицы 500×500



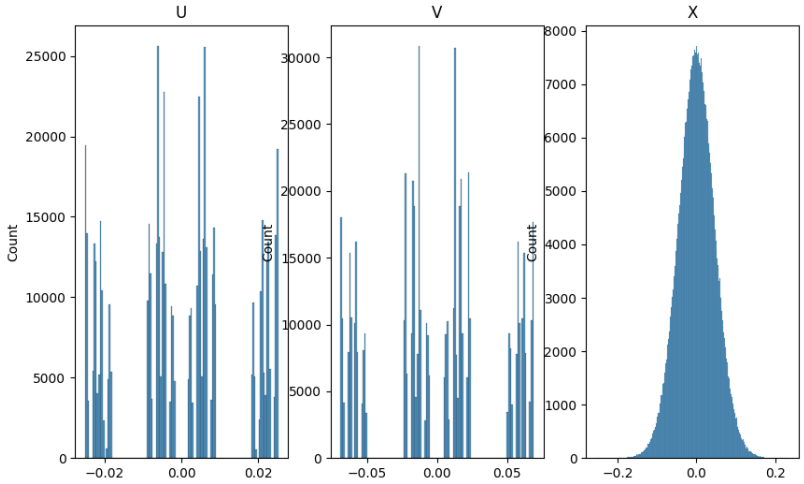
Особенности решения

- Метод кластеризации: *KMeans*, 4 кластера.
- Базис: из случайных матриц, генерируемых из QR-разложения матрицы из нормального распределёния.
- $\varepsilon = 10^{-4}$.
- .
- Pytorch, numpy, sklearn.

Результаты для разложения случайной матрицы $X_{500\times500}$



Результаты для разложения случайной матрицы 500×500



Проблемы

- Метод не всегда сходится за фиксированное число итераций. *В таком случае не квантизую слой.*
- Некоторые матрицы при разложении не группируются по кластерам. *Можно увеличивать количество кластеров.*

В обоих случаях можно пробовать брать другие ортогональные матрицы для базиса.

Качество на downstream задаче

- Модель: DistillBERT.
- Датасет: IMDB Movie Reviews Dataset.
- 10000 - train, 2500 - test. Датасет сбалансирован по меткам 0 / 1.

Метрика / модель	**DistilBERT**	**QDistilBERT**
Accuracy	92.32%	92.22%
f1 score	92.38%	92.20%
ROC-AUC	92.31%	92.20%
test loss	0.385	0.361

Разложение случайных векторов

- Метод квантизации Кашина как для векторов, так и для матриц, сходится.
- Удалось повторить эксперименты на случайных матрицах с похожими графиками распределения значений матриц.
- Проведена квантизация Distilbert для решения задачи классификации сантимента. Квантизованная модель показывает результаты не хуже исходной.

Спасибо за внимание!