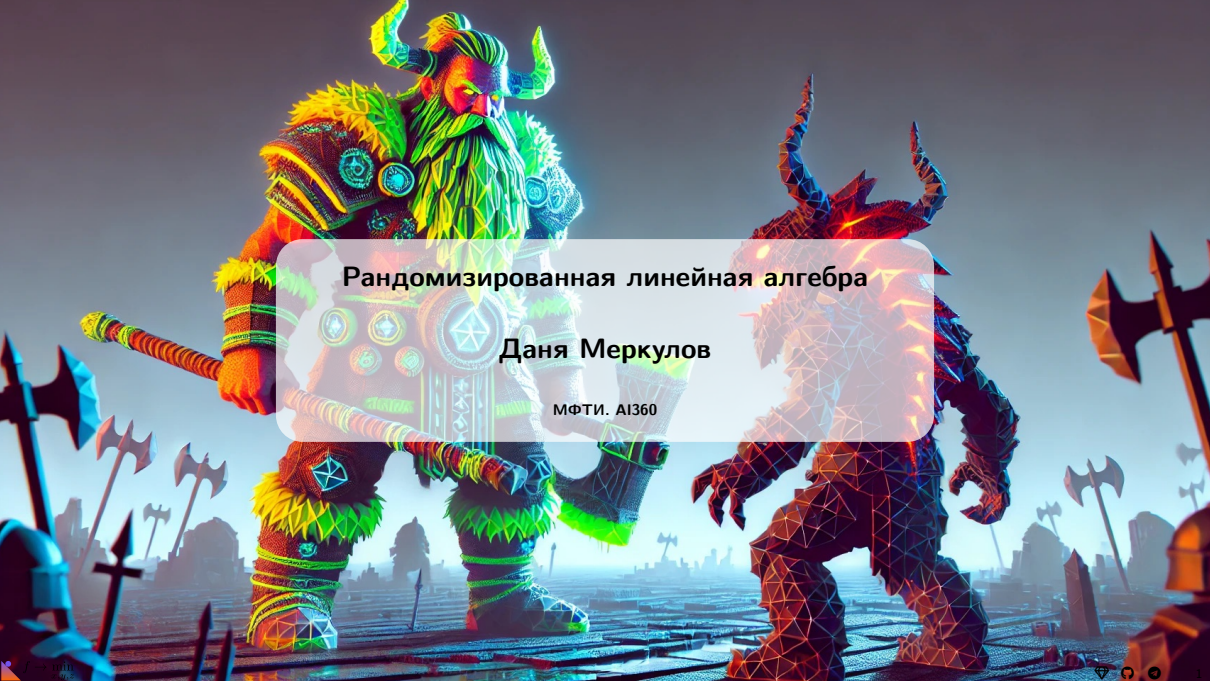

Рандомизированная линейная алгебра

Даня Меркулов

МФТИ. AI360

Рандомизированная линейная алгебра

Проверка матричного равенства: Алгоритм Фрейвалдса

Пусть нам даны три матрицы $A, B, C \in \mathbb{R}^{n \times n}$, и мы хотим проверить, равенство $AB = C$. Наивный способ - просто перемножить A и B за $\mathcal{O}(n^3)$ операций и сравнить с C.

Проверка матричного равенства: Алгоритм Фрейвалдса

Пусть нам даны три матрицы $A, B, C \in \mathbb{R}^{n \times n}$, и мы хотим проверить, равенство $AB = C$. Наивный способ - просто перемножить A и B за $\mathcal{O}(n^3)$ операций и сравнить с C.

Алгоритм Фрейвалдса показывает, что это можно сделать гораздо быстрее - за $\mathcal{O}(kn^2)$ операций при вероятности ошибки не более $1/2^k$.

Проверка матричного равенства: Алгоритм Фрейвалдса

Пусть нам даны три матрицы $A, B, C \in \mathbb{R}^{n \times n}$, и мы хотим проверить, равенство $AB = C$. Наивный способ - просто перемножить A и B за $\mathcal{O}(n^3)$ операций и сравнить с C.

Алгоритм Фрейвалдса показывает, что это можно сделать гораздо быстрее - за $\mathcal{O}(kn^2)$ операций при вероятности ошибки не более $1/2^k$.

Идея алгоритма:

1. Сгенерировать случайный вектор $r \in \mathbb{R}^n$.

Проверка матричного равенства: Алгоритм Фрейвалдса

Пусть нам даны три матрицы $A, B, C \in \mathbb{R}^{n \times n}$, и мы хотим проверить, равенство $AB = C$. Наивный способ - просто перемножить A и B за $\mathcal{O}(n^3)$ операций и сравнить с C.

Алгоритм Фрейвалдса показывает, что это можно сделать гораздо быстрее - за $\mathcal{O}(kn^2)$ операций при вероятности ошибки не более $1/2^k$.

Идея алгоритма:

1. Сгенерировать случайный вектор $r \in \mathbb{R}^n$.
2. Вычислить произведения Br и Cr.

Проверка матричного равенства: Алгоритм Фрейвалдса

Пусть нам даны три матрицы $A, B, C \in \mathbb{R}^{n \times n}$, и мы хотим проверить, равенство $AB = C$. Наивный способ - просто перемножить A и B за $\mathcal{O}(n^3)$ операций и сравнить с C.

Алгоритм Фрейвалдса показывает, что это можно сделать гораздо быстрее - за $\mathcal{O}(kn^2)$ операций при вероятности ошибки не более $1/2^k$.

Идея алгоритма:

1. Сгенерировать случайный вектор $r \in \mathbb{R}^n$.
2. Вычислить произведения Br и Cr.
3. Вычислить $A(Br)$ и сравнить полученный вектор с Cr.

Проверка матричного равенства: Алгоритм Фрейвалдса

Пусть нам даны три матрицы $A, B, C \in \mathbb{R}^{n \times n}$, и мы хотим проверить, равенство $AB = C$. Наивный способ - просто перемножить A и B за $\mathcal{O}(n^3)$ операций и сравнить с C.

Алгоритм Фрейвалдса показывает, что это можно сделать гораздо быстрее - за $\mathcal{O}(kn^2)$ операций при вероятности ошибки не более $1/2^k$.

Идея алгоритма:

1. Сгенерировать случайный вектор $r \in \mathbb{R}^n$.
2. Вычислить произведения Br и Cr.
3. Вычислить $A(Br)$ и сравнить полученный вектор с Cr.
4. Если $A(Br) \neq Cr$, то точно $AB \neq C$.

Проверка матричного равенства: Алгоритм Фрейвалдса

Пусть нам даны три матрицы $A, B, C \in \mathbb{R}^{n \times n}$, и мы хотим проверить, равенство $AB = C$. Наивный способ - просто перемножить A и B за $\mathcal{O}(n^3)$ операций и сравнить с C.

Алгоритм Фрейвалдса показывает, что это можно сделать гораздо быстрее - за $\mathcal{O}(kn^2)$ операций при вероятности ошибки не более $1/2^k$.

Идея алгоритма:

1. Сгенерировать случайный вектор $r \in \mathbb{R}^n$.
2. Вычислить произведения Br и Cr.
3. Вычислить $A(Br)$ и сравнить полученный вектор с Cr.
4. Если $A(Br) \neq Cr$, то точно $AB \neq C$.
5. Если $A(Br) = Cr$, то с вероятностью **не менее** $1 - 1/2$ мы угадали правильно. Для снижения вероятности ошибки до $1/2^k$ повторяем процедуру k раз с разными случайными векторами.

Проверка матричного равенства: Алгоритм Фрейвалдса

Пусть нам даны три матрицы $A, B, C \in \mathbb{R}^{n \times n}$, и мы хотим проверить, равенство $AB = C$. Наивный способ - просто перемножить A и B за $\mathcal{O}(n^3)$ операций и сравнить с C.

Алгоритм Фрейвалдса показывает, что это можно сделать гораздо быстрее - за $\mathcal{O}(kn^2)$ операций при вероятности ошибки не более $1/2^k$.

Идея алгоритма:

1. Сгенерировать случайный вектор $r \in \mathbb{R}^n$.
2. Вычислить произведения Br и Cr.
3. Вычислить $A(Br)$ и сравнить полученный вектор с Cr.
4. Если $A(Br) \neq Cr$, то точно $AB \neq C$.
5. Если $A(Br) = Cr$, то с вероятностью **не менее** $1 - 1/2$ мы угадали правильно. Для снижения вероятности ошибки до $1/2^k$ повторяем процедуру k раз с разными случайными векторами.

Проверка матричного равенства: Алгоритм Фрейвалдса

Пусть нам даны три матрицы $A, B, C \in \mathbb{R}^{n \times n}$, и мы хотим проверить, равенство $AB = C$. Наивный способ - просто перемножить A и B за $\mathcal{O}(n^3)$ операций и сравнить с C.

Алгоритм Фрейвалдса показывает, что это можно сделать гораздо быстрее - за $\mathcal{O}(kn^2)$ операций при вероятности ошибки не более $1/2^k$.

Идея алгоритма:

1. Сгенерировать случайный вектор $r \in \mathbb{R}^n$.
2. Вычислить произведения Br и Cr.
3. Вычислить $A(Br)$ и сравнить полученный вектор с Cr.
4. Если $A(Br) \neq Cr$, то точно $AB \neq C$.
5. Если $A(Br) = Cr$, то с вероятностью **не менее** $1 - 1/2$ мы угадали правильно. Для снижения вероятности ошибки до $1/2^k$ повторяем процедуру k раз с разными случайными векторами.

Сложность каждого шага: $\mathcal{O}(n^2)$ (домножение на вектор), поэтому общее время $\mathcal{O}(kn^2)$.

Рандомизированное решение линейной системы

Рассмотрим СЛАУ

$$Ax = b,$$

где $A \in \mathbb{R}^{m \times n}$. Метод Качмарца (Kaczmarz) обновляет приближение x_k путём выборки случайной строки i (обычно с вероятностью пропорциональной $\|a_i\|^2$):

$$x_{k+1} = x_k - \frac{a_i^T x_k - b_i}{\|a_i\|^2} a_i.$$

Рандомизированное решение линейной системы

Рассмотрим СЛАУ

$$Ax = b,$$

где $A \in \mathbb{R}^{m \times n}$. Метод Качмарца (Kaczmarz) обновляет приближение x_k путём выборки случайной строки i (обычно с вероятностью пропорциональной $\|a_i\|^2$):

$$x_{k+1} = x_k - \frac{a_i^T x_k - b_i}{\|a_i\|^2} a_i.$$

- Если система совместна, метод сходится к точному решению.

Рандомизированное решение линейной системы

Рассмотрим СЛАУ

$$Ax = b,$$

где $A \in \mathbb{R}^{m \times n}$. Метод Качмарца (Kaczmarz) обновляет приближение x_k путём выборки случайной строки i (обычно с вероятностью пропорциональной $\|a_i\|^2$):

$$x_{k+1} = x_k - \frac{a_i^T x_k - b_i}{\|a_i\|^2} a_i.$$

- Если система совместна, метод сходится к точному решению.
- Если система переопределена или шумна, то можно показать сходимость к решению наилучшего приближения.

Рандомизированное решение линейной системы

Рассмотрим СЛАУ

$$Ax = b,$$

где $A \in \mathbb{R}^{m \times n}$. Метод Качмарца (Kaczmarz) обновляет приближение x_k путём выборки случайной строки i (обычно с вероятностью пропорциональной $\|a_i\|^2$):

$$x_{k+1} = x_k - \frac{a_i^T x_k - b_i}{\|a_i\|^2} a_i.$$

- Если система совместна, метод сходится к точному решению.
- Если система переопределена или шумна, то можно показать сходимость к решению наилучшего приближения.
- С точки зрения SGD это шаг стохастического градиента для задачи минимизации $\|Ax - b\|^2$.

Рандомизированное решение линейной системы

Рассмотрим СЛАУ

$$Ax = b,$$

где $A \in \mathbb{R}^{m \times n}$. Метод Качмарца (Kaczmarz) обновляет приближение x_k путём выборки случайной строки i (обычно с вероятностью пропорциональной $\|a_i\|^2$):

$$x_{k+1} = x_k - \frac{a_i^T x_k - b_i}{\|a_i\|^2} a_i.$$

- Если система совместна, метод сходится к точному решению.
- Если система переопределена или шумна, то можно показать сходимость к решению наилучшего приближения.
- С точки зрения SGD это шаг стохастического градиента для задачи минимизации $\|Ax - b\|^2$.

Рандомизированное решение линейной системы

Рассмотрим СЛАУ

$$Ax = b,$$

где $A \in \mathbb{R}^{m \times n}$. Метод Качмарца (Kaczmarz) обновляет приближение x_k путём выборки случайной строки i (обычно с вероятностью пропорциональной $\|a_i\|^2$):

$$x_{k+1} = x_k - \frac{a_i^T x_k - b_i}{\|a_i\|^2} a_i.$$

- Если система совместна, метод сходится к точному решению.
- Если система переопределена или шумна, то можно показать сходимость к решению наилучшего приближения.
- С точки зрения SGD это шаг стохастического градиента для задачи минимизации $\|Ax - b\|^2$.

Скорость сходимости:

$$\mathbb{E}[\|x_{k+1} - x^*\|^2] \leq \left(1 - \frac{1}{\kappa_F^2(A)}\right) \mathbb{E}[\|x_k - x^*\|^2],$$

где $\kappa_F(A) = \frac{\|A\|_F}{\sigma_{\min}(A)}$.

Рандомизированное матричное умножение

Хотим приблизить произведение AB, где $A \in \mathbb{R}^{m \times p}$, $B \in \mathbb{R}^{p \times n}$. Стоимость точного умножения: $\mathcal{O}(mnp)$.
Рандомизированный подход даёт приближение:

$$AB \approx \sum_{t=1}^{k} \frac{1}{k\, p_{i_t}} A^{(i_t)} B_{(i_t)},$$

Рандомизированное матричное умножение

Хотим приблизить произведение AB, где $A \in \mathbb{R}^{m \times p}$, $B \in \mathbb{R}^{p \times n}$. Стоимость точного умножения: $\mathcal{O}(mnp)$. Рандомизированный подход даёт приближение:

$$AB \approx \sum_{t=1}^{k} \frac{1}{k\, p_{i_t}} A^{(i_t)} B_{(i_t)},$$

где $A^{(i_t)}$ - i_t-й столбец A, а $B_{(i_t)}$ - i_t-я строка B, а p_{i_t} - вероятность выбора i_t-го столбца/строки. Обычно p_i пропорциональны $\|A^{(i)}\|\|B_{(i)}\|$ (или другой норме).

Рандомизированное матричное умножение

Хотим приблизить произведение AB, где $A \in \mathbb{R}^{m \times p}$, $B \in \mathbb{R}^{p \times n}$. Стоимость точного умножения: $\mathcal{O}(mnp)$. Рандомизированный подход даёт приближение:

$$AB \approx \sum_{t=1}^{k} \frac{1}{k\, p_{i_t}} A^{(i_t)}\, B_{(i_t)},$$

где $A^{(i_t)}$ - i_t-й столбец A, а $B_{(i_t)}$ - i_t-я строка B, а p_{i_t} - вероятность выбора i_t-го столбца/строки. Обычно p_i пропорциональны $\|A^{(i)}\|\|B_{(i)}\|$ (или другой норме).

Идея:

1. По нормам столбцов A (и строк B) выбираем k столбцов-строк.

Рандомизированное матричное умножение

Хотим приблизить произведение AB, где $A \in \mathbb{R}^{m \times p}$, $B \in \mathbb{R}^{p \times n}$. Стоимость точного умножения: $\mathcal{O}(mnp)$. Рандомизированный подход даёт приближение:

$$AB \approx \sum_{t=1}^{k} \frac{1}{k\, p_{i_t}} A^{(i_t)} B_{(i_t)},$$

где $A^{(i_t)}$ - i_t-й столбец A, а $B_{(i_t)}$ - i_t-я строка B, а p_{i_t} - вероятность выбора i_t-го столбца/строки. Обычно p_i пропорциональны $\|A^{(i)}\|\|B_{(i)}\|$ (или другой норме).

Идея:

1. По нормам столбцов A (и строк B) выбираем k столбцов-строк.
2. Усреднённая сумма полученных ранга-k матриц приближает всё произведение.

Рандомизированное матричное умножение

Хотим приблизить произведение AB, где $A \in \mathbb{R}^{m \times p}$, $B \in \mathbb{R}^{p \times n}$. Стоимость точного умножения: $\mathcal{O}(mnp)$. Рандомизированный подход даёт приближение:

$$AB \approx \sum_{t=1}^{k} \frac{1}{k\, p_{i_t}} A^{(i_t)} B_{(i_t)},$$

где $A^{(i_t)}$ - i_t-й столбец A, а $B_{(i_t)}$ - i_t-я строка B, а p_{i_t} - вероятность выбора i_t-го столбца/строки. Обычно p_i пропорциональны $\|A^{(i)}\|\|B_{(i)}\|$ (или другой норме).

Идея:

1. По нормам столбцов A (и строк B) выбираем k столбцов-строк.
2. Усреднённая сумма полученных ранга-k матриц приближает всё произведение.

Рандомизированное матричное умножение

Хотим приблизить произведение AB, где $A \in \mathbb{R}^{m \times p}$, $B \in \mathbb{R}^{p \times n}$. Стоимость точного умножения: $\mathcal{O}(mnp)$. Рандомизированный подход даёт приближение:

$$AB \approx \sum_{t=1}^{k} \frac{1}{k\, p_{i_t}} A^{(i_t)}\, B_{(i_t)},$$

где $A^{(i_t)}$ - i_t-й столбец A, а $B_{(i_t)}$ - i_t-я строка B, а p_{i_t} - вероятность выбора i_t-го столбца/строки. Обычно p_i пропорциональны $\|A^{(i)}\|\|B_{(i)}\|$ (или другой норме).

Идея:

1. По нормам столбцов A (и строк B) выбираем k столбцов-строк.
2. Усреднённая сумма полученных ранга-k матриц приближает всё произведение.

Сложность: $\mathcal{O}(mnk)$, если $k \ll p$, экономим по сравнению с $\mathcal{O}(mnp)$.

Оценка следа матрицы методом Хатчинсона [1]

Пусть $X \in \mathbb{R}^{d \times d}$ и $v \in \mathbb{R}^d$ - случайный вектор такой, что $\mathbb{E}[vv^T] = I$. Тогда,

$$\mathrm{Tr}(X) = \mathbb{E}[v^T X v] = \frac{1}{V} \sum_{i=1}^{V} v_i^T X v_i$$

Интересно, что для оценки следа матрицы иногда можно не знать самой матрицы. Например, в современных моделях машинного обучения можно вычислять произведение гессиана функции на произвольный вектор, не вычисляя сам гессиан с помощью автоматического дифференцирования. Пример

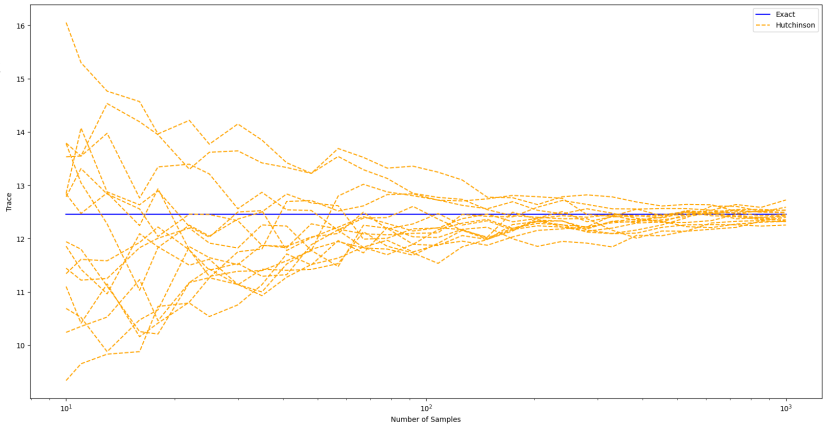

Рис. 1: Source

[1] A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines - M.F. Hutchinson, 1990

Оценка следа матрицы методом Жирара

Метод Жирара:
- Предшественник метода Хатчинсона, где вектор w берём из $\mathcal{N}(0, I)$ (гауссовский).
- Дисперсия получается немного больше, чем у Хатчинсона (у хатчинсона минимальная дисперсия).

Оценка следа матрицы методом Жирара

Метод Жирара:
- Предшественник метода Хатчинсона, где вектор w берём из $\mathcal{N}(0, I)$ (гауссовский).
- Дисперсия получается немного больше, чем у Хатчинсона (у хатчинсона минимальная дисперсия).

Intrinsic dimension (intdim):
Для симметричной положительно определённой матрицы A вводится понятие

$$\mathrm{intdim}(A) = \frac{\mathrm{Tr}(A)}{\|A\|_F}.$$

Оценка следа матрицы методом Жирара

Метод Жирара:
- Предшественник метода Хатчинсона, где вектор w берём из $\mathcal{N}(0, I)$ (гауссовский).
- Дисперсия получается немного больше, чем у Хатчинсона (у хатчинсона минимальная дисперсия).

Intrinsic dimension (intdim):
Для симметричной положительно определённой матрицы A вводится понятие

$$\mathrm{intdim}(A) = \frac{\mathrm{Tr}(A)}{\|A\|_F}.$$

- Минимальное значение равно 1.

Оценка следа матрицы методом Жирара

Метод Жирара:
- Предшественник метода Хатчинсона, где вектор w берём из $\mathcal{N}(0, I)$ (гауссовский).
- Дисперсия получается немного больше, чем у Хатчинсона (у хатчинсона минимальная дисперсия).

Intrinsic dimension (intdim):
Для симметричной положительно определённой матрицы A вводится понятие

$$\mathrm{intdim}(A) = \frac{\mathrm{Tr}(A)}{\|A\|_F}.$$

- Минимальное значение равно 1.
- Максимальное - при всех сингулярных (или собственных) значениях равных, тогда $\mathrm{intdim}(A)$ может достичь $\sqrt{\mathrm{rank}(A)}$.

Оценка следа матрицы методом Жирара

Метод Жирара:
- Предшественник метода Хатчинсона, где вектор w берём из $\mathcal{N}(0, I)$ (гауссовский).
- Дисперсия получается немного больше, чем у Хатчинсона (у хатчинсона минимальная дисперсия).

Intrinsic dimension (intdim):
Для симметричной положительно определённой матрицы A вводится понятие

$$\mathrm{intdim}(A) = \frac{\mathrm{Tr}(A)}{\|A\|_F}.$$

- Минимальное значение равно 1.
- Максимальное - при всех сингулярных (или собственных) значениях равных, тогда $\mathrm{intdim}(A)$ может достичь $\sqrt{\mathrm{rank}(A)}$.

Оценка следа матрицы методом Жирара

Метод Жирара:
- Предшественник метода Хатчинсона, где вектор w берём из $\mathcal{N}(0, I)$ (гауссовский).
- Дисперсия получается немного больше, чем у Хатчинсона (у хатчинсона минимальная дисперсия).

Intrinsic dimension (intdim):
Для симметричной положительно определённой матрицы A вводится понятие

$$\mathrm{intdim}(A) = \frac{\mathrm{Tr}(A)}{\|A\|_F}.$$

- Минимальное значение равно 1.
- Максимальное - при всех сингулярных (или собственных) значениях равных, тогда $\mathrm{intdim}(A)$ может достичь $\sqrt{\mathrm{rank}(A)}$.

С помощью этой величины можно оценивать вероятность больших отклонений случайной оценки следа.

Рандомизированный SVD

Напомним, что SVD матрицы $A \in \mathbb{R}^{m \times n}$:

$$A = U \Sigma V^T.$$

Для больших m и n полное вычисление SVD занимает $\mathcal{O}(\min\{mn^2, m^2 n\})$.

Рандомизированный подход (Halko et al., 2011):

1. Выбираем $G \in \mathbb{R}^{n \times (k+p)}$ со случайными элементами.

Параметр p - **oversampling**, чтобы уменьшить ошибку.

Рандомизированный SVD

Напомним, что SVD матрицы $A \in \mathbb{R}^{m \times n}$:

$$A = U \Sigma V^T.$$

Для больших m и n полное вычисление SVD занимает $\mathcal{O}(\min\{mn^2, m^2 n\})$.

Рандомизированный подход (Halko et al., 2011):

1. Выбираем $G \in \mathbb{R}^{n \times (k+p)}$ со случайными элементами.
2. Считаем $Y = A \cdot G$ и делаем QR-разложение $Y = QR$.

Параметр p - **oversampling**, чтобы уменьшить ошибку.

Рандомизированный SVD

Напомним, что SVD матрицы $A \in \mathbb{R}^{m \times n}$:

$$A = U \Sigma V^T.$$

Для больших m и n полное вычисление SVD занимает $\mathcal{O}(\min\{mn^2, m^2 n\})$.

Рандомизированный подход (Halko et al., 2011):

1. Выбираем $G \in \mathbb{R}^{n \times (k+p)}$ со случайными элементами.
2. Считаем $Y = A \cdot G$ и делаем QR-разложение $Y = QR$.
3. Утверждается, что $QQ^T A \approx A$ при хорошей выборке и $k + p$ надёжно покрывают ведущие сингулярные компоненты.

Параметр p - **oversampling**, чтобы уменьшить ошибку.

Рандомизированный SVD

Напомним, что SVD матрицы $A \in \mathbb{R}^{m \times n}$:

$$A = U \Sigma V^T.$$

Для больших m и n полное вычисление SVD занимает $\mathcal{O}(\min\{mn^2, m^2 n\})$.

Рандомизированный подход (Halko et al., 2011):

1. Выбираем $G \in \mathbb{R}^{n \times (k+p)}$ со случайными элементами.
2. Считаем $Y = A \cdot G$ и делаем QR-разложение $Y = QR$.
3. Утверждается, что $QQ^T A \approx A$ при хорошей выборке и $k + p$ надёжно покрывают ведущие сингулярные компоненты.
4. Строим $B = Q^T A$, у которого размер $(k + p) \times n$.

Параметр p - **oversampling**, чтобы уменьшить ошибку.

Рандомизированный SVD

Напомним, что SVD матрицы $A \in \mathbb{R}^{m \times n}$:

$$A = U \Sigma V^T.$$

Для больших m и n полное вычисление SVD занимает $\mathcal{O}(\min\{mn^2, m^2 n\})$.

Рандомизированный подход (Halko et al., 2011):

1. Выбираем $G \in \mathbb{R}^{n \times (k+p)}$ со случайными элементами.
2. Считаем $Y = A \cdot G$ и делаем QR-разложение $Y = QR$.
3. Утверждается, что $QQ^T A \approx A$ при хорошей выборке и $k + p$ надёжно покрывают ведущие сингулярные компоненты.
4. Строим $B = Q^T A$, у которого размер $(k + p) \times n$.
5. Вычисляем точное SVD для $B = \hat{U} \hat{\Sigma} \hat{V}^T$.

Параметр p - **oversampling**, чтобы уменьшить ошибку.

Рандомизированный SVD

Напомним, что SVD матрицы $A \in \mathbb{R}^{m \times n}$:

$$A = U \Sigma V^T.$$

Для больших m и n полное вычисление SVD занимает $\mathcal{O}(\min\{mn^2, m^2 n\})$.

Рандомизированный подход (Halko et al., 2011):

1. Выбираем $G \in \mathbb{R}^{n \times (k+p)}$ со случайными элементами.
2. Считаем $Y = A \cdot G$ и делаем QR-разложение $Y = QR$.
3. Утверждается, что $QQ^T A \approx A$ при хорошей выборке и $k + p$ надёжно покрывают ведущие сингулярные компоненты.
4. Строим $B = Q^T A$, у которого размер $(k + p) \times n$.
5. Вычисляем точное SVD для $B = \hat{U} \hat{\Sigma} \hat{V}^T$.
6. Тогда $U = Q\hat{U}$, и получаем искомое разложение (приближённое).

Параметр p - **oversampling**, чтобы уменьшить ошибку.