Ответ 2

2. Сравните математические основы и практические последствия применения скалярного квантования (Scalar Quantization, SQ) и продуктового квантования (Product Quantization, PQ) в Qdrant. Как эти методы влияют на точность поиска, использование памяти и вычислительную эффективность? Опишите процесс квантования и деквантования для SQ, а также схему кодирования и асимметричного вычисления расстояний для PQ.

На основе предоставленного учебного материала, сравнение скалярного (Scalar Quantization, SQ) и продуктового (Product Quantization, PQ) квантования в Qdrant, их математических основ и практических последствий выглядит следующим образом.

Сравнение Скалярного и Продуктового Квантования в Qdrant

Квантование в Qdrant — это метод, направленный на сокращение точности представления векторов с целью уменьшения занимаемой памяти и ускорения вычислений.

1. Скалярное квантование (Scalar Quantization, SQ)

Математические основы и процесс квантования/де-квантования

Скалярное квантование преобразует 32-битные числа с плавающей запятой (float32) в 8-битные целые числа (int8). Процесс основан на линейном отображении диапазона значений.

- **Математическая деривация:** Пусть дан вектор $\vec{v} = (v_1, \ldots, v_d) \in \mathbb{R}^d$.

  1. **Определение диапазона:** Для обеспечения устойчивости к выбросам в наборе данных определяются значения, задающие рабочий диапазон. В Qdrant для этого используются 2-й и 98-й перцентили, обозначаемые как p_2 и p_{98} соответственно.

  2. **Вычисление шага квантования:** Для b-битного квантования (в Qdrant обычно $b = 8$, что соответствует $2^8 = 256$ уровням) вычисляется шаг квантования Δ.

  $$\Delta = \frac{p_{98} - p_2}{2^b - 1}$$

     - **Пошаговая деривация:**
       - $p_{98} - p_2$: Вычисляется размах значений между 98-м и 2-м перцентилями, что отсекает экстремальные выбросы.
       - $2^b - 1$: Определяется количество интервалов, на которые будет разделен диапазон. Для $b = 8$ это 255 интервалов.
       - Деление первого на второе дает размер одного шага (интервала) квантования.

  3. **Процесс квантования:** Каждая компонента v_i исходного вектора преобразуется в целочисленное значение v'_i.

  $$v'_i = \text{round}\left(\frac{v_i - p_2}{\Delta}\right)$$

     - **Пошаговая деривация:**
       - $v_i - p_2$: Значение компоненты сдвигается так, чтобы нижняя граница диапазона (p_2) соответствовала нулю.
       - $\frac{\cdot}{\Delta}$: Полученное значение делится на шаг квантования, чтобы определить, в какой из $2^b - 1$ интервалов оно попадает.
       - $\text{round}(\cdot)$: Результат округляется до ближайшего целого числа, которое и является квантованным представлением компоненты.

  4. **Процесс де-квантования:** Для восстановления приближенного исходного значения $\hat{v}_i$ из квантованного v'_i

выполняется обратное преобразование.

$$\hat{v}_i = v'_i \cdot \Delta + p_2$$

- **Пошаговая деривация:**
  - $v'_i \cdot \Delta$: Целочисленный индекс умножается на шаг квантования для восстановления его приблизительного положения в исходном диапазоне.
  - $\cdots + p_2$: К результату прибавляется значение нижней границы диапазона (p_2) для возврата к исходной шкале значений.

Влияние на производительность и ресурсы

- **Использование памяти:** SQ снижает потребление RAM в 4 раза, так как каждый float32 (4 байта) заменяется на int8 (1 байт).
- **Вычислительная эффективность:** Qdrant может вычислять метрики расстояния непосредственно на квантованных векторах, используя быстрые целочисленные SIMD-инструкции (например, AVX). Это значительно ускоряет вычисления, что является критически важным для производительности.
- **Точность поиска:** Вносимая ошибка квантования считается приемлемой для большинства практических задач, включая RAG (Retrieval-Augmented Generation).

---

2. Продуктовое квантование (Product Quantization, PQ)

Математические основы и схема кодирования

Продуктовое квантование обеспечивает еще более сильное сжатие данных, что особенно полезно для хранения векторов на диске.

- **Математическая деривация и схема кодирования:**

  1. **Разбиение вектора:** Исходный вектор $\vec{v} \in \mathbb{R}^d$ разбивается на m непересекающихся суб-векторов $\vec{v}_1, \ldots, \vec{v}_m$, каждый из которых имеет размерность d/m.

  2. **Создание кодовых книг:** Для каждого из m подпространств независимо создается своя кодовая книга (codebook) C_j. Каждая книга C_j состоит из k векторов-центроид (обычно $k = 256$), которые находятся с помощью алгоритма кластеризации k-means на наборе суб-векторов из соответствующего подпространства.

  3. **Кодирование:** Каждый суб-вектор $\vec{v}_j$ исходного вектора заменяется индексом i_j ближайшего к нему центроида из соответствующей кодовой книги C_j. Таким образом, весь вектор $\vec{v}$ представляется в виде набора из m целочисленных индексов $(i_1, \ldots, i_m)$.

Асимметричное вычисление расстояний (Asymmetric Distance Computation)

Для вычисления расстояния между вектором запроса q (который не сжат) и сжатым вектором v используется высокоэффективный асимметричный метод.

- **Схема вычисления:**
  1. Вектор запроса q также разбивается на m суб-векторов $(q_1, \ldots, q_m)$, аналогично векторам в базе.
  2. Для каждого подпространства j предварительно вычисляется таблица расстояний между суб-вектором запроса q_j и всеми k центроидами из кодовой книги C_j.
  3. Итоговое расстояние между q и v аппроксимируется как сумма предвычисленных значений из этих таблиц, соответствующих индексам $(i_1, \ldots, i_m)$, которыми закодирован вектор v.

Этот метод описан в материале как "чрезвычайно быстрый".

Влияние на производительность и ресурсы

- **Использование памяти:** PQ обеспечивает "еще более сильное сжатие" по сравнению с SQ. Qdrant использует этот метод для хранения векторов на диске, что позволяет работать с наборами данных, значительно превышающими объем доступной RAM.
- **Вычислительная эффективность:** Асимметричное вычисление расстояний является чрезвычайно быстрым, так как основная часть вычислений сводится к извлечению предвычисленных значений из таблиц и их суммированию.
- **Точность поиска:** Расстояние является аппроксимацией, что подразумевает более высокую потерю точности по сравнению со скалярным квантованием.

Сравнительный итог

| Характеристика | Скалярное квантование (SQ) | Продуктовое квантование (PQ) | | :--- | :--- | :--- | | **Математическая основа** | Линейное преобразование каждой компоненты вектора в целое число (int8). | Разбиение вектора на суб-векторы и кодирование каждого суб-вектора индексом ближайшего центроида из кодовой книги. | | **Использование памяти** | Снижение в 4 раза (float32 -> int8). | Значительно более сильное сжатие, позволяющее хранить данные на диске и работать с наборами, превышающими RAM. | | **Вычислительная эффективность** | Ускорение вычислений расстояний за счет использования быстрых целочисленных SIMD-инструкций. | Чрезвычайно быстрое асимметричное вычисление расстояний с использованием предвычисленных таблиц. | | **Точность поиска** | Вносимая ошибка приемлема для большинства задач (например, RAG). | Аппроксимация расстояния, что подразумевает потенциально большую потерю точности. | | **Основной сценарий в Qdrant** | Ускорение вычислений и экономия RAM для векторов, находящихся в оперативной памяти. | Сжатие векторов для хранения на диске, работа с очень большими наборами данных. |