

Объяснение методологических обоснований LSH

Введение

Locality-Sensitive Hashing (LSH) представляет собой вероятностный метод приближённого поиска ближайших соседей (Approximate Nearest Neighbor, ANN) в высокоразмерных пространствах. В отличие от классических хеш-функций, минимизирующих коллизии, LSH целенаправленно максимизирует вероятность коллизии для семантически близких объектов. Ключевая идея заключается в проекции данных в пространство меньшей размерности с сохранением свойств локальности: близкие объекты с высокой вероятностью попадают в один и тот же "bucket".

Современные исследования предлагают инновационные подходы к оптимизации LSH, которые существенно расширяют возможности традиционного LSH, решая ключевые проблемы вычислительной сложности и эффективности поиска.

Теоретические основы LSH

Формальное определение и базовые принципы

Формально, семейство хеш-функций \mathcal{H} называется (r, cr, p_1, p_2) -чувствительным для метрики d , если для любых двух точек p и q выполняются условия:

1. Если $d(p, q) \leq r$, то $\Pr[h(p) = h(q)] \geq p_1$,
2. Если $d(p, q) \geq cr$, то $\Pr[h(p) = h(q)] \leq p_2$,

где $c > 1$, а $p_1 > p_2$ [1].

Для усиления вероятностных свойств используются конструкции AND и OR:

- **AND-construction:** Комбинация k независимых хеш-функций усиливает строгость попадания в bucket ($p' = p_1^k$)
- **OR-construction:** Использование L независимых хеш-таблиц увеличивает вероятность попадания хотя бы в один bucket ($p' = 1 - (1 - p_1)^L$)

На практике теоретическая вероятность коллизии $P_{h_k^e}(r)$ часто интерпретируется как ожидаемая полнота (recall rate). Хотя $P_{h_k^e}(r)$ является математическим ожиданием, эмпирические исследования показывают, что для большинства реальных datasets фактическая

полнота оказывается близка к теоретическому ожиданию при условии корректного выбора параметров k и ℓ и достаточного размера данных [1]. Это объясняется тем, что дисперсия полноты оказывается крайне мала благодаря агрегации множества независимых хеш-функций и действию закона больших чисел.

Вероятностные гарантии и регулирование ошибки

LSH предоставляет строгие вероятностные гарантии для качества поиска. Для найденного кандидата x можно оценить вероятность того, что он является истинным s -приближенным соседом:

$$P[\text{success}] \geq 1 - (1 - p_1^k)^L$$

Вероятность ложноположительного срабатывания (далекая точка попала в бакет) не превышает:

$$P[\text{false positive}] \leq p_2^k \cdot L$$

Параметры LSH позволяют явно регулировать точность и вероятность ошибки:

- **Ширина бакета (w):** Увеличивая w , снижаем вероятность коллизии для всех точек, но увеличиваем точность
- **Число sampled dimensions (m):** Увеличение m снижает дисперсию оценки расстояния, улучшая точность
- **Число хеш-функций (k):** Увеличение k снижает p_1 и p_2 , делая фильтрацию жестче
- **Число хеш-таблиц (L):** Увеличение L повышает вероятность нахождения истинного соседа, но увеличивает память и время

Поддержка различных метрик расстояния

LSH поддерживает различные метрики расстояния при условии выбора соответствующих LSH-семейств:

Метрики и соответствующие LSH-семейства

- **Евклидово расстояние:** Используется r -стабильное распределение (распределение Гаусса). Вероятность коллизии вычисляется как:

$$p(r) = \int_0^w \frac{1}{r} \exp\left(-\frac{t^2}{2r^2}\right) \left(1 - \frac{t}{w}\right) dt$$

- **Расстояние Хэмминга:** Используется LSH-семейство на основе битовых операций. Вероятность коллизии определяется долей совпадающих бит:

$$p(r) = 1 - \frac{r}{d}$$

где d — размерность пространства.

- **Косинусное расстояние:** Применяются гиперплоскостные LSH-функции ($sign(\langle r, x \rangle)$). Вероятность коллизии зависит от угла между векторами:

$$p(r) = 1 - \frac{\arccos(\cosine_sim)}{\pi}$$

Сравнительная характеристика метрик

Метрика	LSH-семейство	Вероятность коллизии	Оптимальные сценарии
Евклидово	р-стабильное распределение	$p(r) = \int_0^w \frac{1}{r} \exp\left(-\frac{t^2}{2r^2}\right) \left(1 - \frac{t}{w}\right) dt$	Общие данные, изображения, векторы
Расстояние Хэмминга	Битовые хэши	$p(r) = 1 - \frac{r}{d}$	Тексты, категориальные данные, хэши
Косинусное	Гиперплоскостное	$p(r) = 1 - \frac{\arccos(\cosine_sim)}{\pi}$	Тексты, рекомендации, NLP

Таблица 1: Характеристики LSH для разных метрик расстояния

1 Методологические улучшения

1.1 Ускорение вычислений: FastLSH

Классические реализации LSH для евклидова пространства (E2LSH) требуют вычисления скалярных произведений полномерных векторов, что приводит к сложности $O(n)$ на одну хеш-функцию. Метод FastLSH сочетает случайное проецирование и случайную выборку признаков (random sampling). Вместо полного n -мерного вектора для каждого хеша случайно выбирается подмножество из m координат ($m \ll n$), к которому применяется стандартная LSH-функция. Это сокращает вычислительную сложность до $O(m)$.

Формально, модифицированная хеш-функция приобретает вид:

$$h_{\tilde{a}, \tilde{b}}(v) = \left\lfloor \frac{\tilde{a}^T S(v) + \tilde{b}}{\tilde{w}} \right\rfloor$$

где $S(\cdot)$ — оператор случайной выборки m координат из исходного вектора v , $\tilde{a} \in \mathbb{R}^m$ — случайный вектор проекции, $\tilde{b} \sim U[0, \tilde{w}]$ — случайное смещение.

Теоретически показано, что при $m \rightarrow \infty$ распределение расстояний в sampled-пространстве сходится к исходному. При конечных m расхождения моментов распределения незначительны, что сохраняет LSH-свойства. Эксперименты подтверждают, что даже при $m = 30$ (при

исходной размерности в сотни/тысячи) качество поиска практически не ухудшается, а скорость возрастает в десятки раз.

1.2 Оптимизация поиска: roLSH

Другое направление оптимизации — сокращение дорогостоящих обращений к данным (I/O operations). Метод roLSH предлагает две стратегии:

roLSH-samp: На этапе индексации выполняется сэмплирование тестовых запросов для определения статистики оптимальных радиусов поиска. Это позволяет начинать поиск не с минимального радиуса, а с предварительно оцененного значения.

roLSH-NN: NN используется для предсказания оптимального радиуса поиска для конкретного запроса на основе его хешей. Это позволяет для каждого запроса стартовать с радиуса, близкого к оптимальному, минимизируя последующие расширения.

Оба метода значительно сокращают количество обращений к диску — основной проблемы в вычислениях.

1.3 Multi-Probe LSH

Multi-Probe LSH расширяет идею roLSH путем многозондового поиска в соседних корзинах. Алгоритм не ограничивается одной корзиной, а исследует несколько соседних корзин, что позволяет значительно сократить количество необходимых хэш-таблиц (до 5-8 раз) без потери точности.

1.4 Entropy-Based LSH

Entropy-Based LSH динамически адаптирует параметры хэширования на основе энтропии распределения данных. Это позволяет улучшить точность поиска в условиях неоднородного распределения данных.

2 Сравнительная характеристика современных алгоритмов LSH

3 Теоретические основы современных улучшений

3.1 Вероятностные гарантии FastLSH

Вероятность коллизии для FastLSH выражается как:

$$p(s, \sigma) = \int_0^{\tilde{w}} f_{|\tilde{s}X|}(t) \left(1 - \frac{t}{\tilde{w}}\right) dt$$

Алгоритм	Основное нововведение	Преимущества	Недостатки	Сценарии применения
FastLSH	Случайная выборка признаков	Снижение сложности до $O(m)$	Требует настройки m	Высокоразмерные данные
roLSH	Адаптивный радиус поиска	Сокращение I/O операций	Зависимость от качества сэмплирования	Большие datasets
Multi-Probe LSH	Многозондовый поиск	Сокращение хэш-таблиц	Усложнение реализации	Высокоточные приложения
Entropy-Based LSH	Адаптация к энтропии данных	Высокая точность	Вычислительная сложность	Неоднородные данные
C2LSH	Подсчет коллизий	Уменьшение хэш-функций	Ограниченная применимость	Общие случаи
QALSH	Учет особенностей запроса	Оптимизация под запрос	Зависимость от качества запроса	Рекомендательные системы
I-LSH	Оптимизация I/O операций	Эффективность в внешней памяти	Высокое время обработки	Внешняя память

Таблица 2: Сравнение современных алгоритмов LSH

где $s = \|v - u\|_2$ — евклидово расстояние между векторами, σ — стандартное отклонение распределения расстояний по компонентам, $f_{|\tilde{s}X|}(t)$ — плотность распределения модуля произведения $\tilde{s}X$.

3.2 Адаптивное расширение радиуса в roLSH

Вместо экспоненциального расширения (1, 2, 4, 8...), roLSH использует линейное или ступенчатое расширение на основе предсказанного радиуса:

$$R_{\text{new}} = \begin{cases} R_{\text{init}} + 2^x & 0 \leq x \leq \log_2 R_{\text{init}} \\ 2^x & x > \log_2 R_{\text{init}} \end{cases}$$

Это значительно сокращает количество случайных обращений к диску (random disk I/O), которые являются основным узким местом при работе с внешней памятью.

Ограничения и дальнейшие направления

Несмотря на эффективность, LSH имеет ряд ограничений:

- **Зависимость от параметров:** Производительность сильно зависит от выбора параметров (k , L , w и др.), которые часто подбираются эмпирически.

- **Чувствительность к данным:** Эффективность может снижаться на данных со специфическими распределениями или аномалиями.
- **Вычислительная сложность:** Хотя LSH сокращает общее число сравнений, стоимость построения индекса может быть высокой для очень больших datasets.

Заключение

Locality-Sensitive Hashing является мощным инструментом для приближённого поиска ближайших соседей в высокоразмерных пространствах. Его методологическая основа сочетает вероятностные гарантии с практической эффективностью. Теоретически обоснованная вероятность коллизии находит подтверждение в реальных приложениях благодаря эффекту больших данных и тщательному подбору параметров.

Современные улучшения LSH, такие как FastLSH и roLSH, делают ключевые ограничения классических схем, делая LSH ещё более производительными и практичным. Поддержка различных метрик расстояния и возможность явного регулирования точности делают LSH универсальным инструментом для решения задач поиска похожих объектов в больших массивах данных.

Список литературы

- [1] Lu, K., Wang, H., Xiao, Y., Song, H. Why locality sensitive hashing works: A practical perspective // Information Processing Letters. – 2018. – Т. 136. – С. 49-58.
- [2] Tan, Z., Wang, H., Xu, B., Luo, M., Du, M. Fast Locality Sensitive Hashing with Theoretical Guarantee // arXiv preprint arXiv:2309.15479v1. – 2020.
- [3] Jafari, O., Nagarkar, P., Montano, J. Improving Locality Sensitive Hashing by Efficiently Finding Projected Nearest Neighbors // arXiv preprint arXiv:2006.11284v1. – 2020.