Stochastic Approximate Nearest Neighbor in High Dimensions

Alex Jones, Neeraj Kumar, Isaac Mackey

February 13, 2017

Abstract

We pose the approximate nearest neighbor problem in high dimensions with an auxillary stochastic requirement. This problem serves as a crucial primitive for the most likely neighbor problem in high dimensions and as a important problem on its own. We analyze the difficulty of this problem, and the efficacy of proposed solutions through theoretical and emprical methods.

1 Introduction

1.1 Problem Definition

Consider a database of *stochastic sites* $P = \{(p, u) \ : \ p \in \mathbb{R}^d, u \in [0, 1]\}$. These entries represent a location in high dimensional Euclidean space associated with an auxillary probability value meant to represent existence, risk, or other stochastic quantity of interest. A stochastic approximate neighbor query is a quadruplet (r, c, q, u), $r, c \in \mathbb{R}, q \in \mathbb{R}^d, u \in [0, 1]$. A data structure which supports S-ANN queries will return a stochastic site (p, v) such that $d(p, q) \leq cr$ and $v \geq u$ with a probability at least f for some fixed f. Intuitively, this output is a "nearby" point which also satisfies our auxillary requirement.

2 Solution Approaches

For an arbitrary query and database, we have no hope to improve upon the running time, preprocessing time or space of existing ANN, since queries of the form $(r, c, q, 0)$ are identical to ANN queries. Our theoretical goal then is to assume limited information about the database and/or queries and improve preprocessing time, space or running time of an appropriate data structure.

At our disposal is an ANN algorithm which has the following properties for arbitrary n sized databases with dimension d

- Preprocessing Time - $O(d^a \cdot n^b)$
- Space - $O(dn + n^c)$
- Query Time - $O(d^e \cdot n^h)$
- Probability of Success (at least) - $f_{success}$

for suitable constants a, b, c, e, h). For the recently developed LSH algorithm of [**?**, **?**, Andoni - Optimal Data-dependent] we have in Theorem 2.3 that with probability $f_{success} = n^{-\rho - o_c(1)}$, the constants fall out as $a = d = 1$, $c = 1 + o_c(1)$, $b = 1 + o_c(1)$ and $h = o_c(1)$.

2.1 Unique Probabilities

For simplicity, consider when P contains k unique values of u over all stochastic sites, namely $u_1, ..., u_k$, and let the number of entries in P with existence value u_i be n_i. As a naive approach, we can treat each equivalence class $P_i = \{(p, u) : u = u_i\}$ as a separate database.

Known Query Distribution A natural first question is the following: what is the expected number of entries of our database that are valid with respect to the auxillary requirement for a query drawn from $\mathcal{Q}$. Assume that all queries have identical values of r, c and are drawn from the joint distribution $\mathcal{Q}$, whose marginal distribution of $u \in [0, 1]$ is given by $\mathcal{U}$. Imbibe the random variable U with the marginal distribution of $\mathcal{U}$.

We say that a stochastic site (p, u) *satisfies the auxillary condition with respect to a query* (r, c, q, u') when $u \geq u'$ Our question posed in the previous paragraph is then

$$\mathbb{E}_{\mathcal{Q}}\big[|\{s = (p, u) \ : \ s \text{ satisfies the query auxillary condition } \}|\big] \tag{1}$$

where our randomness is taken over independent samples of $\mathcal{Q}$. Decomposing this, we get

$$\begin{aligned}
&\mathbb{E}_{\mathcal{Q}}\big[|\{s = (p, u) \ : \ s \text{ satisfies the query auxillary condition } \}|\big] && (2)\\
&= \mathbb{P}\big\{(p, u) \in P, u \geq u', (r, c, q, u') \leftarrow \mathcal{Q}\big\} && (3)\\
&= \sum_{i=1}^{k} n_i \cdot F_U(u_i) && (4)
\end{aligned}$$

where F_U is the distribution function of U. If the fraction of points with auxillary value u is given by $f_x(u)$, then we can rewrite equation 4 as $\int_0^1 f_x(u) F_U(u) du$. This is maximized when the mass of u in the dataset lies above all the mass of U, and our queries become ANN queries. This is minimized when all of $f_x(u)$ is exhausted before any mass of U is used, and our queries become empty. Hence, it is interesting to consider performance of a black box type algorithm in between the two extremes.