Machine Learning Foundations **Fall semester, 2012**

Lecture 7: December 2, 2012

Lecturer: Yishay Mansour

Scribe: Ariel Jarovsky, Evgeny Gorokhovsky, Eyal Altshuler [1]

7.1 Nearest Neighbors Algorithm

7.1.1 Overview

Consider a Sample Space $X = \langle x_i, b_i \rangle$ of points and their classifications ($b \in \{0, 1\}$). Given a new point x, we wish to find its nearest point x_i, and predict x's classification to be b_i. An implicit assumption here is that close points have the same classification.
For example: we wish to learn a continuous function $f : \mathbb{R} \to \mathbb{R}$. Let $X = \langle (x_i, y_i), b_i \rangle$ be a set of points in the plane. The classification b_i indicates whether (x_i, y_i) is over the function or below. Given a new point x, returning the classification of the nearest point seen will be a good approximation for the classification of x.

7.1.2 Various Methods of NN

- **NN**
  Given a new point x, we wish to find it's nearest point and return it's classification.

- **K-NN**
  Given a new point x, we wish to find its k nearest points. Lets call this group S_k :
  $S_k = \{$ k nearest points $\}$. We predict that the probability that X will be classified as '1', is equal to the relation between the number of points from S_k that are classified as '1', and k. That is, $\frac{l}{k}$ where $l = |\{\langle x_i, b_i \rangle \, | b_i = 1 \ and \ x_i \in S_k\}|$

- **Assigning Weights According to Distances**
  We shall sort all points in the sample Scope according to their distance from x (x_1 is the closest). Each point x_i will be assigned with a weight function $w_i = \frac{i}{H(k)}$, where

$$H(k) = \sum_{i=1}^{k} \frac{1}{i} \sim \ln k \, .$$

[1] Based on scribe written by Aya Aner, Mar. 11th 1997

We estimate the probability of x to be '1' as

$$\sum_{i=1}^{m} w_i \cdot b_i \, .$$

7.1.3 Models of the Nearest Neighbor Algorithm

Why do we expect the above methods to work? Because we believe, intuitively, that close points classify similarly. We shall show a simple theoretical model, in which we will prove this intuition.
There are 3 equivalent ways for representing the model that generates the points:

1. We assume a distribution $\mathcal{D}$ on all points x. For every x there is a probability function $p(x)$ by which we classify x, that is,

   $$p(x) = \mathbf{Pr}[c^*(x) = 1]$$

   We first choose x according to $\mathcal{D}$ and then we choose b according to $p(x)$. Although this is the most intuitive model, there is a problem with this method since we need $p(x)$ to be measurable.

2. Let $\mathcal{D}_0$ be the distribution of the negative points, and let $\mathcal{D}_1$ be the distribution of the positive points. Let $\alpha \in [0, 1]$. First, we sample b ($b \in \{0, 1\}$) with bias α, i.e $\mathbf{Pr}[b = 1] = \alpha$. Then,

   - If $b = 1$: choose x from $\mathcal{D}_1$.
   - If $b = 0$: choose x from $\mathcal{D}_0$.

   Finally, return $\langle x, b \rangle$. Thus,

   $$p(x) = \frac{\alpha \cdot \mathcal{D}_1(x)}{\alpha \cdot \mathcal{D}_1(x) + (1 - \alpha) \cdot \mathcal{D}_0(x)} \, .$$

   Here $\mathcal{D}(x) = \alpha \cdot \mathcal{D}_0(x) + (1 - \alpha) \cdot \mathcal{D}_1(x)$

3. We assume a distribution $\mathcal{D}$ on the space $X \times \{0, 1\}$. We will choose from $\mathcal{D}$ a point of the form $\langle x, b \rangle$. In this case,

   $$p(x) = \frac{\mathcal{D}(\langle x, 1 \rangle)}{\mathcal{D}(\langle x, 0 \rangle) + \mathcal{D}(\langle x, 1 \rangle)} \, .$$

Notes

- Each of the three models is a legitimate model (Though the first model, which is the most intuitive one, requires restrictions related to Measure Theory).
- In each model we assume that the points are chosen independent from the distribution.
- Up to now, $\mathcal{C}^*$ was allways our target function. There was also just one distribution $\mathcal{D}$, and now we assume two different distributions: one for the positive points and one for the negative ones.

The Optimal Rule

Assume we wish to minimize the number of errors (in the 0-1 Loss Model). Given a probability function $p(x)$, let us define

$$h(x) = \begin{cases} 1 & p(x) \geq \frac{1}{2} \\ 0 & p(x) < \frac{1}{2} \end{cases}$$

We prove that $h(x)$ is the optimal rule. Assume $p(x) > \frac{1}{2}$. Then for a given x,

$$E[Loss(h, x)] = 1 - p(x)$$

Let $g(x)$ be an arbitrary rule, so that $\mathbf{Pr}[g(x) = 1] = \beta > 0$. (That is, $g(x)$ is a randomized rule) . Calculating the expected loss for g,

$$E[Loss(g, x)] = p(x) \cdot (1 - \beta) + (1 - p(x)) \cdot \beta$$

It is easy to see that the outcome is between $p(x)$ and $1 - p(x)$. Since we assumed $p(x) > \frac{1}{2}$, we obtain $\beta \cdot p(x) + (1 - \beta) \cdot (1 - p(x)) > 1 - p(x)$. Thus, we conclude that the average loss for g is greater than the average loss for p : $E[Loss(g, x)] > E[Loss(h, x)]$.

A similar proof can be shown for $p(x) < \frac{1}{2}$, which makes $p(x)$ the best rule.

Bayes Risk

We define 'Bayes Risk' as the 'expected loss' when 'everything is known', that is, when we use the optimal rule h. Let the loss of h at x $r(x) = \min(p(x), 1 - p(x))$. Define:

$$R^* = E[Loss(h, x)] = \int_x r(x)\mathcal{D}(x)dx \ ,$$

to be the expected loss.

Now, we wish to find upper and lower bounds for the error of the NN rule. Let us define $P = E_x[Loss(NN, x)]$ as the value of the error of the nearest neighbor of x. Evidently, $R^* \leq P$ (since R^* is the optimal loss). We will prove that $P \leq 2R \cdot (1 - R^*)$.
First let us examine the implication of these bounds :

1. If $R^* \approx 0$ then $P \approx 0$.
   (If $R^* = \epsilon$ then $\epsilon \leq P \leq 2\epsilon \cdot (1-\epsilon) \leq 2\epsilon$. Taking $\epsilon \to 0$ we receive $P \to 0$)

2. If $R^* \approx \frac{1}{2}$ then $P \approx \frac{1}{2}$. (If $R^* = \frac{1-\epsilon}{2}$ then $P \leq 2 \cdot \frac{1-\epsilon}{2} \cdot \frac{1+\epsilon}{2} = \frac{1-\epsilon^2}{2}$)

We shall show our outcome at the **limit** of a large sample.

In order to derive our bound, let us consider a simple case:

Simple Case

We wish to classify a new point x. Let us assume that there is exactly one point, $\langle x_i, b_i \rangle$ in the sample space, such that $x_i = x$. Naturally, this is the nearest point, so the NN Algorithm returns b_i. What is the probability of an error, that is , $\mathbf{Pr}[b \neq b_i]$?
Let us observe that when we "generated" b_i, we chose its value to be '1' with probability of $p(x)$, and with probability $1 - p(x)$ a value of '0'. We will "generate" b in the same manner. It is easy to see that

$$\mathbf{Pr}[b = b_i] = p(x) \cdot p(x) + (1 - p(x)) \cdot (1 - p(x))$$

Therefore,

$$\mathbf{Pr}[b \neq b_i] = p(x) \cdot (1 - p(x)) + (1 - p(x)) \cdot p(x) = 2p(x) \cdot (1 - p(x)) = 2r(x) \cdot (1 - r(x))$$

The last equality results from the definition of $r(x) = \min(p(x), 1 - p(x))$.

Now let us sum this over all the points in the sample space. The expected error is :

$$\int_x 2p(x) \cdot (1 - p(x)) \cdot \mathcal{D}(x)dx =$$
$$2\int_x r(x) \cdot (1 - r(x)) \cdot \mathcal{D}(x)dx =$$
$$2\int_x r(x) \cdot \mathcal{D}(x)dx - 2\int_x r(x)^2 \cdot \mathcal{D}(x)dx \leq 2R^* - 2(R^*)^2 =$$
$$2R^* \cdot (1 - R^*)$$

(The inequality holds since $E[y]^2 \leq E[y^2]$)
Therefore, we proved the bound in this special case. We Will now extend the bound to the general case.

General Case

We like to extend the proof to the general case. For this purpose we will need to prove :

1. The nearest neighbor of x converges to the point x (when the sample size, m, goes to infinity):
$$E_x E_{x_1,\ldots,x_m}[min_i \|x_i - x\|] \xrightarrow[m\to\infty]{} 0$$

2. The classification of the neighborhood of x is close to that of $p(x)$: x will be positive with probability approximately $p(x)$, and negative with probability $1 - p(x)$, for large enough m.

For simplification we assume the following :

1. The density function $\mathcal{D}(x)$ is non-zero over the whole sample space (i.e., there are no holes in $\mathcal{D}$ and there are no mass points).

2. The function $p(x)$ is continuous (if $\mathcal{D}_0$ and $\mathcal{D}_1$ are continuous, then $p(x)$ continuous).

(Our claim can also be proved with much weaker assumptions)
Given $x_1, x_2, \ldots, x_m$, let $NN_1^m(x)$ be the nearest neighbor of x from $x_1, x_2, \ldots, x_m$:

$$NN_1^m(x) = \arg\min_i \|x_i - x\|$$

Theorem 7.1 *For every given x, $NN_1^m(x) \xrightarrow[m\to\infty]{} x$ with probability 1.*

Proof: For every ϵ let us build a sphere $B_\epsilon(x)$: The center is x and the radius is ϵ. Given that the density function $\mathcal{D}(x)$ is never zero, $\mathbf{Pr}[x_i \in B_\epsilon(x)] = q > 0$ (q is the 'positive weight' of the sphere, with respect to the distribution).

Therefore, if we sample m points, the probability that all of them are **not** in $B_\epsilon(x)$ is $(1-q)^m$. Since q is a constant and $m \to \infty$, this probability converges to zero. This is true for every ϵ, therefore $NN_1^m(x)$ converges to x with probability 1.
We can also write it in a quantitative analoguous way:

$$\forall \epsilon, \delta > 0.\ \exists m_{\mathcal{D}}(\epsilon, \delta).\ \mathbf{Pr}[NN_1^m \notin B_\epsilon(x)] \leq \delta$$

The main issue is that the sample size depends on $\mathcal{D}$. ■

Theorem 7.2 *Let $z = NN_1^m(x)$. Then $r(z) \longrightarrow r(x)$ when $m \to \infty$ with probability 1.*

Proof: From Theorem 7.1 we know that $z \longrightarrow x$ (when $m \to \infty$) with probability 1. Since p is continuous, $p(z) \longrightarrow p(x)$.

■

Note that the continuity of p is a sufficient condition but not a necessary one.

Conclusion: Let us define $e^m = \{$ The event in which the NN Algorithm made a mistake with a sample space of m points $\}$. Then $\mathbf{Pr}[e^m] = \int_x Pr[e^m|x]\mathcal{D}(x)dx$.
Since $r(z) \longrightarrow r(x)$ when $m \to \infty$ we can reduce it to the simple case. Replacing x_i with z we get :

$$\mathbf{Pr}[e^m|x] \xrightarrow[m\to\infty]{} 2r(x) \cdot (1 - r(x))$$

We are interested in the error of NN. Assuming that the sample size goes to infinity, we have:

$$\lim_{m\to\infty} \int_x \mathbf{Pr}[e^m|x]\mathcal{D}(x)dx = \int_x \lim_{m\to\infty} \mathbf{Pr}[e^m|x] \cdot \mathcal{D}(x)dx \leq 2R^* \cdot (1 - R^*)$$

Note that we can interchange the limit with the integral in the first equality since this is a sequence of events that goes to infinity.

What happens when $p(x)$ is not continuous ? In that case, Theorem 7.2 will not apply. If for every ϵ there is a function $q(x)$ which approximates $p(x)$ with an error of ϵ, and $q(x)$ is continuous, we can implement all our assumptions on $q(x)$ and proceed from there. (Here we will also have to require some conditions from Measure Theory).

7.1.4 More Complex Algorithms

K-NN :

K-NN takes the k nearest neighbors. Let us define NN_k^m :
$NN_k^m = \{$*the k* x_i *neighbors which are closest to* $x\}$.

We will estimate $\hat{p}(x) = \frac{l}{k}$ where l is the number of points which are classified as '1' in NN_k^m . Note that for $k = m$ this estimation is clearly bad (it is α, the probability for a positive sample). Thus, our conditions are :

- $k \longrightarrow \infty$ - the dependency on the set grows up as m grows up.
- $\frac{k}{m} \longrightarrow 0$ - k will grow slower than m.

For example: $k = \log m, or\ \sqrt{m}$.
Like before, we shall prove :

1. The k'th nearest neighbor of x converges to x (i.e. $\forall z \in NN_k^m(x)$, $z \to x$).
2. For each neighbor $z \in NN_k^m(x)$, $p(z) \longrightarrow p(x)$.

Proof :

1. **Convergence to** x
   The Simple Case:
   Let us assume that all the k nearest points are identical to x. This case is similar to flipping a coin k times, with probability $p(x)$ of the coin to be '1'. Since $k \to \infty$, we can use one of the rules of the Concentration Bounds, e.g. Chernoff:

$$\mathbf{Pr}\left[|\frac{l}{k} - p(x)| \geq \lambda\right] \leq e^{-\lambda^2 k} \xrightarrow[k\to\infty]{} 0$$

   Thus, the average converges to the true expected value: $\frac{l}{k} \longrightarrow p(x)$.

   The General Case:
   Like before, we shall build a sphere $B_\epsilon(\vec{x})$. Now we wish that k points will coincide in $B_\epsilon(\vec{x})$. Defining $Pr[x_i \in B_\epsilon(\vec{x})] = q$, we expect about $q \cdot m$ points to be in $B_\epsilon(\vec{x})$. We would like to bound the probability that less then k points are in $B_\epsilon(\vec{x})$. Let Avg be the number of points in $B_\epsilon(\vec{x})$, divided by m. Let z be the number of points in $B_\epsilon(\vec{x})$

$$\mathbf{Pr}\left[\textit{At most } k-1 \textit{ points in } B_\epsilon(\vec{x})\right] = \mathbf{Pr}\left[|z-q| \geq q - \frac{k-1}{m}\right] \leq 2e^{-(q-\frac{k}{m})^2 \cdot m}$$

   Since $\frac{k}{m} \longrightarrow 0$ when $m \to \infty$, we have for each q : $e^{-(q-\frac{k}{m})^2 m} \longrightarrow 0$, and we conclude that the k nearest points are at a distance of at most ϵ from x, with probability '1', as m goes to infinity.

2. **Convergence to** $p(x)$ This can be concluded directly from the continuity of $p(x)$. The fraction of positive points, l, out of the nearest k, converges to $p(x)$. i.e., $\frac{l}{k} \longrightarrow p(x)$. The proof uses concentration bound, $\mathbf{Pr}\left[|\frac{l}{k} - p(x)| \geq \lambda\right] \leq e^{-\lambda^2 k} \xrightarrow[k\to\infty]{} 0$, where we assume that for any $z \in p(x)$, $p(z) = p(x)$, by contiouity.

7.1.5 General Method

Let w_i be the weight of the i's neighbor of x. We can similarly show that

$$\sum_{i=1}^{m} w_i b_i \longrightarrow p(x)$$

if :

1. $\sum_{i=1}^{m} w_i = 1$
2. $w_1 \to 0$

3. For every constant α :

$$\sum_{i=\alpha m}^{m} w_i \longrightarrow_{m\to\infty} 0$$

We shall call each set of the above described weights 'consistent'.
Remarks:
Condition 3 assures us that only points which are close to x will influence the weighted sum. Condition 2 assures us that there will not be a dominant point.

Drawbacks of the General Method

1. *Rate of Convergence*

   If we take a sample of a fixed size, we will not be able to compute a bound on the error. For example, let us look at a 2-dimensional space (Figure (7.1)). Let the segment $y = 0$

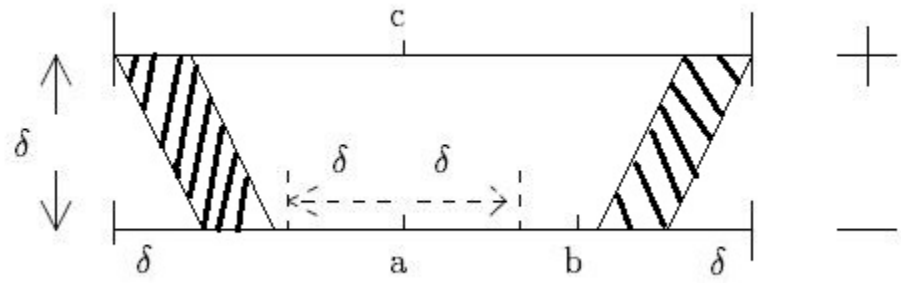

Figure 7.1: Example of a Slow Rate

be classified '-' and $y = \delta$ be classified '+'. The probability distribution $\mathcal{D}_0$ is uniform over $x \in [0, 1]$ and $y = 0$, and $\mathcal{D}_1$ is uniform over $x \in [0, 1]$ and $y = \delta$. The parameter is $\alpha = \frac{1}{2}$. If we disregard the segment of size 2δ containing a, then the probability of an error (when sampling the point c) is equivalent to the probability of a non-error (when sampling the point b). Thus, we shall have to sample $O(\frac{1}{\delta})$ points in order that a will have a neighbor at a distance less than δ . The probability for an error is

$$\frac{1 - \mathbf{Pr}\left[S_1^m \in [x - \delta, x + \delta]\right]}{2}$$

Considering our initial assumptions, we have

$$\mathbf{Pr}\left[S_1^m \in [x-\delta, x+\delta]\right] = 1-(1-2\delta)^m \leq 2\delta m\,,$$

so the error is at least

$$\frac{1-2\delta m}{2} = \frac{1}{2} - \delta m\,.$$

2. *High Dimensions*

In low dimension it is very natural that points are close to each other. We show that in R^n all points are distant . For example, let us choose random points $x_1, x_2, \ldots, x_m \in \{0,1\}^n$. For each x_i, the probability that x and x_i agree on more than $\frac{n}{2}+\lambda$ coordinates is

$$\mathbf{Pr}\left[|\frac{d(x_i,x)}{n} - \frac{1}{2}| \geq \lambda\right] \leq e^{-\lambda^2 n}\,,$$

where $d(x_i, x)$ is the Hamming distance, i.e., the number of coordinates that x_i and x disagree on. The probability this holds for some x_i is

$$\mathbf{Pr}\left[\exists i|\frac{d(x_i,x)}{n} - \frac{1}{2}| \geq \lambda\right] \leq \sum_i \mathbf{Pr}\left[|d(x_i,x) - \frac{n}{2}| \geq \lambda\right] \leq m \cdot e^{-\lambda^2 n}$$

Thus, if $\lambda >> \sqrt{\frac{\log m}{n}}$, then the probability that x and x_i agree on more than $\frac{n}{2}+\lambda$ coordinates is very small. Therefore, we expect the closest neighbor to agree on $\frac{n}{2} \pm O(\sqrt{n \log m})$, which means that if $m << 2^n$ (Sample space not big enough) , a point chosen randomly will be very distant from all the other points.
To conclude - this method is efficient mostly for lower dimensions, or when there is significant structure in high dimensions.

3. *Determining the Distance Function (Metrics)*

We want that our way of computing the distances between the points will **not** be effected by the different scales (grams vs. kilograms, minutes vs. seconds).
There are two ways to normalize the distances:

(a) We will compute the $\min_i$ and $\max_i$ for each attribute. If the initial value at the i'th coordinate is v_i, then the value we use is

$$\frac{v_i}{\max_i - \min_i}$$

This method is consistent with the assumption of a uniform prior distribution over the i'th attribute.

(b) Assuming normal distribution, we will compute the average of the values of the ith coordinate, $\mu_i = E[x]$, and the variation $\sigma_i^2 = E[x^2] - E^2[x]$. The value we use is

$$\frac{v_i - \mu_i}{\sigma_i}$$

This is compatible with assuming a prior distribution. A second problem is determining the metric. Each of the following is acceptable :

(a) $L_\infty(x, y) \Rightarrow \max_i(|x_i - y_i|)$
(b) $L_1(x, y) \Rightarrow \sum_{i=1}^m |x_i - y_i|$
(c) $L_2(x, y) \Rightarrow \sqrt{\sum_{i=1}^m |x_i - yi|^2}$

7.1.6 Nearest Neighbour Algorithm - Locality Sensitive Hashing

The trivial algorithm for K-NN search is linear in search time - just go over all points and compute the distance. We would like to generate a sub-linear algorithm for this problem. To do that, we are going to ask the inverted question: given x and R find $y \in S$ s.t $\|x - y\| \leq R$ or return that there is no such R

$$\forall y \in S \, \|x - y\| \geq R$$

in fact, we will use an approximation, and return "there is no such R" if for some constant c

$$\forall y \in S \, \|x - y\| \geq cR$$

*D*efinition. A Locality Sensitive Hashing hash function family, is a set of functions H s.t for any $p, q \in R^d$,

1. If $|p - q| < R$ then $\Pr_H [h(p) = h(q)] \geq p_1$
2. If $|p - q| \geq cR$ then $\Pr_H [h(p) = h(q)] \leq p_2$

for some probabilities $p_1 > p_2$

Example

Say $x \in \{0, 1\}^n$ and we take hash function that return only one of the bits: $H = \{h_i\}$ for $h_i(x_1, \ldots, x_d) = x_i$. The probability of two values mapping to the same value is

$$\Pr_H [h(x) = h(y)] = 1 - \frac{1 - d_h(x, y)}{d}$$

where d_h is the hamming distance. If $cR \geq R + 1$ we will have $p_1 > p_2$ as required.

7.1.7 Using LSH function to support neighbour queries

We will use H to create a data structure that supports the required query using two steps.

Step 1: Amplification

To use H in to solve our problem, we will need to amplify the probability difference. The amplification will use functions

$$g(x) = \langle h_i(x), \ldots, h_k(x) \rangle$$

where h_i are randomly selected from H. Let us examing the probabilities for g.

1. If $|p-q| < R$ then $\Pr_H[g(p) = g(q)] \geq p_1^k$
2. If $|p-q| \geq cR$ then $\Pr_H[g(p) = g(q)] \leq p_2^k$

We will set k s.t $p_2^k = \frac{1}{n}$, which implies that

$$p_1^k = p_1^{\frac{lnn}{\ln 1/p_2}} = n^{-\frac{ln\frac{1}{p_1}}{ln\frac{1}{p_2}}}$$

We define $\rho = -\ln\frac{1}{k} / \ln\frac{1}{p_2}$ so $\rho = \frac{\ln\frac{1}{p_1}}{\ln\frac{1}{p_2}} < 1$.

Step 2: Combination

The amplification step produced a hash function that has a larger distance between $p_1$and p_2. However, it also lowered the probability of a match when $|p-q| < R$. We will want to increase this probability. This is done by picking several functions $g_1, \ldots, g_L$ and requiring at least one of them to match. We know that

$$\Pr_H[g_i(p) \neq g_i(q) \mid \|p-q\| < R] \leq 1 - n^{-\rho}$$

for any g_i. The probability that there is not match for any of the functions is $(1-n^{-\rho})^L$. For any probability δ if we take $L = n^\rho \ln\frac{1}{\delta}$ we get a failure probability on matching points of

$$\left(1-n^{-\rho}\right)^L = \left(1-n^{-\rho}\right)^{n^\rho \ln\frac{1}{\delta}} \approx e^{-\ln\frac{1}{\delta}} = \delta$$

For "far" points, the probability to hit is $\frac{1}{n}$ so the expected of times we will find a single "far" point in the tables is:

$$\frac{L}{n} = n^\rho \ln\frac{1}{\delta}$$

where we sum over all far points.

7.1.8 Algorithm description

We are given a hash family H of functions with the required properties and a sample. We want to pre-process the sample in a way that allows us to efficiently find points with distance less than cR from the input.

Pre-processing

Pick L functions $g_1, \ldots, g_L$, where $g(x) = \langle h_i(x), \ldots, h_k(x) \rangle$ with h_i randomly selected from H. For each sample, calculate $g_i(x)$ and enter the result into a regular hash table.

Search

Given an input q, we calculate $g_i(q)$ for $i = 1, \ldots, k$ and search the result in the ith hash table. Depending on the requirements we could return the first p s.t. $g_i(p) = g_i(q)$ or all $P = \{p | \exists i \ g_i(p) = g_i(q)\}$. If we want the nearest neighbor we calculate the distance between q and every point in P.

Space complexity

We have L hash tables, each storing the results of hash functions for each of the n samples and size $O(n)$. Overall the space used is $O(nL) = O(n^{1+\rho})$

Search time complexity

Assuming the hash table lookup time is constant, search will need to make at most $O(L)$ queries to lookup tables. The samples retrieved will be checked to verify that they are close to the input. The expected number of "far" points found is at most $n \cdot \frac{L}{n} = L$ so we will spend $O(L)$ time comparing distances and rejecting "far" samples. The time for processing "close" samples depends on the number of neighbors we are looking for. For k neighbors the lookup time will be $O(kL)$.

7.1.9 Extension for bounded values

We presented an example for H that works on discrete input. Now assume $X = \{0, \ldots, s-1\}^d$ for some finite s and we will want to use L_1 as our distance metric. The solution will use unary encoding - i.e. d blocks of s with each value a represented by a consecutive 1s followed by $s - a$ 0s. For example with $s = 8$, the value $x = \langle 5, 7 \rangle$will be represented as

$$x = 1111100011111110$$

The Hamming distance of two values in this representation is the same as the L_1distance in the original representation so we can reduce this problem to the original LSH presented above. Problems with real values can be reduced to $\{0,\ldots,s-1\}^d$ with quantization.

7.1.10 Extension for real numbers with small radius

In this example we take $x=[0,1]^d$ and we assume $R \ll 1$. The solution is to pick at random uniformly $s_1,\ldots,s_d \in [0,1]$ and use

$$h(x) = \langle sign(x_1 - s_1), sign(x_2 - s_2), \ldots, sign(x_d - s_d)\rangle$$

This hash function meets the criteria since

$$\Pr[sign(x-s) \neq sign(y-s)] = |x-y|$$

for $x,y \in [0,1]$ so

$$\Pr[h(x) = h(y)] = \prod_{i=1}^{d} (1 - |x_i - y_i|)$$

If we have small R we have

$$R = \sum_{i=1}^{d} |x_i - y_i| \ll 1$$

and

$$\Pr[h(x) = h(y)] = \prod_{i=1}^{d} (1 - |x_i - y_i|) \approx 1 - \sum_{i=1}^{d} |x_i - y_i|$$

so we get a separation between p_1 and p_2 given a big enough constant c.