

Scalable Data Science
Prof. Anirban Dasgupta
Department of Computer Science and Engineering
Indian Institute of Technology, Gandhinagar

Lecture – 13b
Data Dependent LSH

Welcome to the course on Scalable Data Science. I am Anirban from IIT, Gandhinagar and we are still talking about locality sensitive hashing. So, today's topic is Data Dependent Locality Sensitive Hashing.

(Refer Slide Time: 00:27)

Locality Sensitive Hashing

Given input data, radius r , approx factor c and confident δ

Output: if there is any point at distance $\leq r$ then w.p. $1 - \delta$ return one at distance $\leq cr$ $c \sim 1 + \epsilon$

Algo: Choose (k, L) .

do L times

iid hash functions : $\{h_{i1} \dots h_{ik}\}$

Create hash table H_i by putting each x in bucket

$$H_i(x) = (h_{i1}(x), \dots, h_{ik}(x))$$

Store non-empty buckets in normal hash table

Picture courtesy Slaney et al.

IIT Gandhinagar
Indian Institute of Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

Again just to refresh a memory a offer locality sensitive hashing was that we were looking at this particular framing of the nearest neighbor question. That we are given a set of data points x_1 to x_n . We are also given a query point q and a radius r .

So, what we have to return is the following that if there is any point that is a distance L minus r then you your algorithm should return one that is a distance $c r$. So, think of c as a small constant. So, c is a small constant. So, let us say $1 + \epsilon$. And your algorithm is a randomized one. So, so you are allowed to fail with probability δ , which means that with probability $1 - \delta$ you must satisfy this guarantee.

So, the parameters of the of the locality sensitive hashing are k and L and if you remember the if you remember the algorithm, it was we chose for each hash table h_i , we

chose k iid hash functions, $h_{i,1}$ to $h_{i,k}$ and then we created and then we hashed the point x using $h_{i,1}$ to $h_{i,k}$ that gave us a k tuple and that k tuple is the bucket id for the hash table $h_{i,k}$.

So, note. So, before we even proceed something. So, before we then proceed something should strike you that the choice of the hash functions has nothing to do with the data right. It has to do with the distance metric that you promised to use right, but it does not really sort of take into account the data itself this is completely oblivious to the data is, this a good thing right.

(Refer Slide Time: 02:11)

The slide is titled "Issues" and contains three bullet points. To the right of the text is a diagram showing a 2D space partitioned by several blue lines representing hash functions. Red dots representing data points are scattered across the space, with some falling into the partitions. The bottom of the slide features logos for IIT Gandhinagar and NPTEL Online Certification Courses.

Issues

- Parameters k, L need to be tuned for each domain
- Random directions are meant to create a random partitioning of the dataset
- While useful to guard against "worst case datasets", we do not exploit the dataset structure

IIT Gandhinagar
Indian Institute of Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

So, the what it does is that, the basic intuition of the lsh brings is that we are creating a random partitioning of the data set, that if that lies as follows right. Then each hash bucket h_i can be thought of as building a random partition that we have already talked about before right that, we are sort of partitioning and the points in each of this partition go to one of the buckets in one of the hash tables. Each hash table is a separate random partitioning of the data also.

So, why is this useful? So, this is really useful to guard against worst case data sets, right. And the guarantee that the way to interpret that the guarantee the guarantee that lsh brings is to say that what; however, you design the data set right we can get sub linear query time some into the rhos query time by doing by running this algorithm right with constant probability $1 - \delta$.

So, even for adversarially design data sets we can get sub linear query time; however, this is not to say that this is the, that if you know something about the data set, if your data set is not adversely is not adversarially chosen this is not necessarily the best algorithm to run right. I mean potentially I could try to exploit some property of the data set in order to create this random partitioning right and this could possibly act and this could possibly perform better than the than the partitioning that lsh creates.

(Refer Slide Time: 03:50)

The slide is titled "Hashing as binary codes" and contains the following content:

- Assume points are in Euclidean space
- How can we get binary vectors so that Hamming distance approximates Euclidean distance

The slide includes several diagrams:

- A set of points in a 2D space with several red arrows representing random directions.
- A 3D ellipsoid representing a high-dimensional space with multiple red arrows indicating random directions.
- A diagram showing a point being projected onto a line, with a vertical arrow labeled $h_i(a)$ and a horizontal arrow labeled a_i .
- A diagram showing two binary vectors: $\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$ with arrows pointing to the right.

Logos for IIT Gandhinagar and NPTEL ONLINE CERTIFICATION COURSES are visible at the bottom.

Right for instance assume here assume for instance that the points lie in Euclidean space right.

So, let us say that the points are all lie in this particular ellipsoid right in this high dimensional ellipsoid. So, what that means, is that if you choose random directions and try to partition the data right you might not be very successful for most of the directions right. If the directions if the random partitions if the random directions lie along I mean lie along orthogonal to this to this ellipsoid more or less orthogonal to this ellipsoid right , then in those directions right. If you try to project along this vector for instance from this vector right.

Now on this vector the points will be bunched up right would be very bunched up. So, then random partitioning does not really I mean then doing this bucketing along this direction does not really work that well it is not I mean because all distances I mean

distance from here and here right distance in these 2 points right gets bunched up in a very I mean points that are really far apart potentially come to come very close right.

So, basically it sort of a it does not it what it means is that practically what would happen is that for is hash table that is created by projecting along these directions right. Only a small number of hash buckets will contain all the data. So, this is not really nice ok. So, ideally if I knew that my data has this particular structure right. Then I should exploit this and say that I should be choosing the partitioning along I should be choosing the projection vectors along these directions. Not along vectors that are orthogonal to the subspace of the data more or less orthogonal to the subspace of the data right.

So, how do we exploit this? So, how do we formalize it right? So, so one way that this has been studied is to say that is to look at lsh as a coding problem. In sense that what are we doing what we doing is that we are taking this points in Euclidean space right and we are assigning let us say binary codes to it right. So, we could as well imagine the as well imagine the lsh indices to be assigning bits right. I mean in the case of the Euclidean lsh $h_i q_i$ was a was a index, but if we were taking same hash for instance then $h_i q_i$ is something that is a binary value 0 or 1 or a plus 1 or minus 1 equivalently right.

So, let us imagine that that we are assigning binary codes right. And what we want is that nearby points should get nearby codes, what is nearby mean. So, the notion of nearby that is the, that is sort of valid in binary is that of hamming distance. So, just to recall what hamming distance is that suppose we have 2 binary values 0 1 0 1 0 0 1 0 and 1 1 0 1 1 0. Then the hamming distance between them is the number of positions right that they differ on for instance here and here right.

So, the hamming distance is 3 between these 2 between these 2 binary vectors.

So, what we want is that points that are closer let us say these 2 points should have hamming (Refer Time: 07:28) should have vectors that are close in hamming distance and points that are far apart for instance these 2 right, should have bit vectors that are far apart in hamming distance. And if I have this formulation right why not use the property of the input data set because the input data set is already with us right. So, then the question is that can I try to find out such binary codes by solving an optimization problem right.

(Refer Slide Time: 08:00)

Properties of a binary code

- Should be easily computable
- Should preserve distances approximately
- Should have small number of bits
 - the bits should be independent and unbiased

$k \rightarrow 2^k =$

IIT Gandhinagar
Indian Institute of
Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

So, we write down the optimization problem, let us see what are the properties of such binary code that we want. Number one is fairly obvious that the binary code should be easily computable. Right, computing it, computing the binary code should not take I mean should not be very computationally expensive. Secondly, it should preserve distances approximately, that is the entire sort of property of LSH right.

So, the locality sensitive hashing preserves locality that in it sort of I mean we know that, we know that the probability of collision right which means that the probability of having the same smaller hamming distance right is proportional to the distance between is proportional to the distance between the points. If the distance is small property of collision is high, if the distance is large property of collision is low and lastly it should also have a small number of bits because the number of bits directly translates into the number of buckets right for instance, if I have k bits that says that we need 2^k buckets more or less right.

So, the number of bits is should be more or less small what it further also implies is that the bits need to be independent of each other, right for instance if you take if you take one I mean if you take I mean if you take projection directions, that are this and this right, let us say let us say v_1 and v_2 right. So and you could imagine the projections of points along v_1 is really sort of close to it is projection on v_2 . So, v_1 and v_2

information theoretically, they capture the same information about the distance between 2 points a very similar information right.

So, we do not want such vectors v_1 and v_2 we want them to be independent or uncorrelated to with each other as much as possible right and also it makes sense that we want the bits to be biased unbiased which means that for about 50 percent of the points there should be every bit for about 50 percent of the points it should be plus 1 and about 50 percent of the points it should be minus 1. So, plus 1 or minus 1 or 0 I will keep referring to both of them.

(Refer Slide Time: 10:13)

Optimization

- W_{ij} = similarity between i and j
- Say $W_{ij} = \exp\left(-\frac{|x_i - x_j|^2}{s}\right)$
- y_i = codeword for point $i \rightarrow 0, 1, \dots, 10$
- $|y_i - y_j|^2$ also equals $\text{Hamming}(i, j)$

Handwritten notes:
 $x_i \in \mathbb{R}^d$
 $\|x_i - x_j\|_2$
 $x_i \approx O(s)$
 $x_j \approx O(s) \rightarrow cs \exp(-c)$
 (Diagram: A vector x_i in a space with radius r)

IIT Gandhinagar
 Indian Institute of Technology Gandhinagar
 NPTEL ONLINE CERTIFICATION COURSES

So, in order to do this, let us try to create an optimization problem right. And let us deal with a specific value of similarity. So, this is called the Gaussian kernel. That supposing W_{ij} supposing x_i are points in \mathbb{R}^d ok. So, of course, you could define a similarity by taking let us say the dot product between them, but here is a nicer notion of similarity, which says that W_{ij} is defined as the first you take the distance between x_i and x_j the L_2 distance right. And then you define the similarity as \exp of minus $|x_i - x_j|^2$ whole square by s .

So, this s is kind of like that radius right at which you are interested in sort of considering similar points. What it means is that if x_i and x_j if the distance between x_i and x_j right is more or less of the order of s right then this W_{ij} gets a high value reasonably high value right. If s if the distance between x_i and x_j is much much bigger

than let us say ω s, then W_{ij} gets quickly gets to be small right. Because this is the let us say the I mean if this is let us say some c times s the W_{ij} gets to be exp of minus c right which means that it decreases exponentially ok.

So, this is called the Gaussian kernel and this, and let us take this notion of similarity between points. Let us say the w_i is the code word for point i right which means that that I mean y_i is a code word for point i which means that y_i is a bit vector let us say of length k . So, k is going to be fixed beforehand. So, then it is not very hard to see that $\|y_i - y_j\|_2^2$ also equals hamming distance between i and j ok.

(Refer Slide Time: 12:15)

Learning codes

- Average hamming distance = $\sum_{ij} W_{ij} |y_i - y_j|^2$
- $y_i \in \{-1, +1\}^k$
- Each bit should be unbiased: $\sum_i y_i = 0_k$
- Bits should be uncorrelated $\sum_i y_i y_i^t = I$

The diagram shows a matrix with rows labeled $y_1, y_2, y_3, \dots, y_n$ and columns labeled $1, 2, 3, \dots, k$. The entries are $+1$ and -1 . The first row is $+1, -1, +1$. The second row is $-1, +1, +1$. The third row is $-1, +1, -1$. Below the matrix, there are three zeros $0, 0, 0$ under the first three columns, representing the sum of each column.

So, the average hamming distance between the points is now is the summation the $ij W_{ij}$ times $\|y_i - y_j\|_2^2$ ok. So, and each y_i is a bit vector of length k and I say that we will either denote them as 0 once or it is for the optimization purposes it is easier to denote them as -1 and 1 . So, that is what we will go with and each bit should be unbiased right. What it means is that let us say let us say k equal to 3 . So, so y_1 which is the bit vector for the item 1 is $+1, -1, +1$ plus y_2 should be $-1, +1, +1$ plus y_3 plus $-1, +1, -1$ and so on and y_n .

So now if you add up the y_i s, for every coordinate these should sum up to 0 . These should sum up to 0 for every coordinate k equal to $1, 2, 3$. This is 0_k ok. This is what this equation means the summation $\sum_i y_i$ is 0 right. So, this is 0 is really of length k a vector of length k here. Now we also want the bits to be uncorrelated right, which means that what

we want is that the y_i transpose y_i the covariance matrix, you look at the covariance matrix of the bits right and that is identity ok.

(Refer Slide Time: 13:50)

Learning codes

- Average hamming distance = $\sum_{ij} W_{ij} |y_i - y_j|^2$
- $y_i \in \{-1, +1\}^k$
- Each bit should be unbiased: $\sum_i y_i = 0$ ←
- Bits should be uncorrelated $\sum_i y_i y_i^t = I$ ←

$k=1$ $k=2$

IIT Gandhinagar
 Indian Institute of
 Technology Gandhinagar

NPTEL ONLINE
 CERTIFICATION COURSES

So, and under these constraints. So, so why did this come about? This came about because we do not want let us say that the i th bit the k equal to 1 and the and the and the second bit let us say that i th and the j th the first and second to be copies of one and other right. So, we want the dot product of these to be 0 and that is why the identity comes into the picture.

So, under these constraints that the individual coordinates are independent of each other and each coordinate is unbiased right I want to minimize this average hamming distance.

(Refer Slide Time: 14:30)

The slide is titled "Casting as optimization problem" and is attributed to [Waiss et al.]. It contains the following text:

- Can we solve : minimize $\sum_{ij} W_{ij} |y_i - y_j|^2$
- subject to
 - $y_i \in \{-1, +1\}^k$
 - $\sum_i y_i = 0$
 - $\sum_i y_i y_i^t = I$

At the bottom of the slide, there are logos for IIT Gandhinagar (Indian Institute of Technology Gandhinagar) and NPTEL ONLINE CERTIFICATION COURSES.

So, whatever. So, why do I want to minimize it? I want to minimize it right because if you look at this quantity right I want the W_{ij} s are fixed right. So now, if a particular W_{ij} is high which means that the similarity is high, I want y_i to be close to y_j right. If the similarity of i and j is high, then I want y_i and y_j to be more or less similar to each other the hamming distance will be small right and if the W_{ij} is small then I do not care so much whether y_i and y_j are similar to each other if the points are far not very similar to each other right.

So, and these are the constraints that we have that we have written down the before right. So, can we solve this problem? So now, we have written this down as an explicit optimization problem, can you solve this right. Unfortunately, no although it might seem like that this starts looking like a convex programming problem, but this particular constraint is a combinatorial constraint right which means that this problem becomes very hard to solve, this can be formalized in this following way that.

(Refer Slide Time: 15:51)

Hardness

- Unfortunately, no!, even for single bit
- Graph partitioning problem: For graph G partition $V(G)$ into two sets A and B such that $|A| = |B|$ and

minimize $\sum_{i \in A, j \in B} W_{ij}$

$V(G) = A \cup B$

Handwritten annotations:
 $k=1$
NP-hard
 $P=NP$

IIT Gandhinagar
Indian Institute of Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

This problem is even for k equal to 1 right this is an NP hard problem right.

So, if you if you do not remember what are NP hard problems, NP hard problems are problems that you can guess and verify the solution in polynomial time, but without resolving the question P equal to NP , you are not expected to get a algorithm for this for an NP-hard problem right. That sort of getting a polynomial time algorithm for this would be equivalent to solving the I mean would be equivalent to solving the (Refer Time: 16:33) NP question. So, and the corresponding NP hard problem that we resolved that we that we use to show the to show that our problem is also NP hard is this problem of called graph partitioning, which is a problem in which we have we have we are partitioning the set V of G into 2 sets A and B right.

So, intuitively because there is one bit here, this is the this is the and then that particular bit is going to determine which elements fall in A and which elements fall in B . So, what we will and what we will do is what this graph partitioning problem says is that we have 2 sets A and B and we have we have the values can we try to partition the set of vertices into 2 sets A and B right such that we minimize the edges crossing the cut ok.

So, normally if I did not have this constraint, then this is an easy problem, but under this constraint that the size of the 2 sets have to equal right this becomes an NP hard problem called balanced partitioning. Ok now what I mean if you want to show that if you want to

show that our problem is also NP hard right you have to reduce this graph partitioning problem to our problem right.

And while we are not going to go into the explicit reduction the intuition is that, you can model this graph partitioning problem as the optimization problem that we just saw right and the bit y_i , because there is only one such bit right. Now is going to tell us whether a bucket whether a vertex should be put whether a vertex or a point should be put in the set i or in the in the set A or in the set B right and this optimization problem and this minimization problem is exactly our objective function exactly (Refer Time: 18:29) our objective function within a factor exactly (Refer Time: 18:31) objective function.

So, then this in some sense is exactly the problem that we are solving for k equal to 1. And we and thus we do not hope to solve even polynomial algorithm for this.

(Refer Slide Time: 18:43)

Spectral Relaxation

- $Y = n \times k$ code matrix
- Diagonal D , $D_{ii} = \sum_j W_{ij}$
- minimize $\sum_{ij} W_{ij} |y_i - y_j|^2 = \text{trace}(Y^t(D - W)Y)$
- $Y^t \cdot \mathbf{1} = 0$
- $Y^t Y = I$

Drop the constraint that Y are in $\{-1, +1\}$

So, what do we do in such setting? There is a standard trick which is to say that we relax the problem. So, what is relaxing mean? Relaxing basically means dropping some constraints right and the most obnoxious that we had was the one that said that the y_i have to be in minus 1 all of them have to be minus 1 or plus 1 that each coordinate has to be minus 1 and 1 right.

So, I haven't introduced this notation before. So, I am calling y the code matrix. So, y is the matrix where every row y_i right is really the code for or the binary code for the point

i or the for the data set for the for the data point i right. And so, y_i is of dimension n by k . n by k it is much easier to write if it this way also here I have defined the diagonal matrix D where D_{ii} is the is a summation $\sum_j W_{ij}$.

So, the matrix D minus W right is also known as the Laplacian matrix, but that is we do not need to go into that right now. So, then I mean with these new definitions we can we can write down the our objective function right as the trace of this matrix $y^T (D - W) y$. So, if you do not remember the or the definition of a trace the trace of a matrix is really nothing but the sum of the diagonal elements right.

So, if you it takes a little bit of algebra to see that if you calculate $y^T (D - W) y$ times y you get exactly this objective function and the other 2 constraints that we had before boil down to saying that $y^T \mathbf{1} = 0$ right and $y^T y = I$ because I mean the that that is exactly the matrix form of the 2 constraints that we had before and we are dropping the constraint that y is are in minus $\mathbf{1}$.

So now the y s can contain real numbers ok.

(Refer Slide Time: 20:47)

Spectral codes $L = D - W$
 $\sum_j y_j = 0$

- The above problem is solved by $Y =$ smallest k eigenvectors of $D - W$
 - After dropping the one with value 0
- To get codes,
 - We could threshold eigenvectors, but then hard to extend it for query

Handwritten notes on the slide include a matrix:

$$\begin{pmatrix} 18 & -1 \\ -2 & -1 \\ 17 & +1 \\ -19 & +1 \end{pmatrix} y \cdot \mathbf{1} = 0$$
 and

$$\sum_j y_j = 0$$

IIT Gandhinagar
 Indian Institute of Technology Gandhinagar
 NPTEL ONLINE CERTIFICATION COURSES

So now if we drop the constraint, this problem becomes surprisingly similar to what we to something that we have seen before. And it is the eigenvector problem right. The above problem that we that is this particular problem is solved by taking y to be the smallest k eigenvectors of D minus W right except one with the value 0.

So, if you know anything about Laplacian you know that if you know if I take the matrix D minus W right this is a positive semi definite matrix right. So, all it is eigenvalues are greater than equal to 0 it does have an eigenvalue at 0 which is given by the all ones eigenvector right. And then and then everything else is non negative right. In fact, strictly bigger than 0 if under some connect under some simple constraints ok.

So, what we are going to do is that we are going to drop the eigenvector that is all ones right because that does not have any meaning and then we are going to take the next smallest k eigenvectors of D minus W right the next k , k of them and this will be my y ok. So, ideally we could take this, but remember we needed finally, we needed we needed plus 1s and minus 1s right.

Now each of these eigenvectors now because this is positive semi definite matrix each of these eigenvectors again are orthogonal to each other. And therefore, they are orthogonal to the all ones matrix to the all one sector, which means that each of these eigenvectors satisfy if y is an eigenvector then if y_i is the i th eigenvector then y_i satisfies y_i^T one equal to 0 which means that summation y_i equal to 0 summation y_{ij} . Summation over all j is equal to 0 right and this starts looking like the this starts looking like the constraint that we had before ok.

So, what we could do is that we could we could take this eigenvectors and we could threshold them right we could threshold them at 0, that any coordinate if an eigenvector comes out to be let us say 0.8, minus 0.2, 0.7 minus 0.9 and so on and so, forth then this is converted to 1, this is converted to 0 this is converted to 1 this is converted to 0 and so on and so, forth or 1 and minus 1 let me go with one and minus 1s here 1 and minus 1 plus 1 minus 1 and so on and so, forth. So, this is my quantized y_i we could always do this, but this is a slight problem right that what do we do when I get the query point right. Because the eigenvectors depend on the entire data set once the query point comes that you have to recompute this entire thing right because the query point was not there in the data set when we compute to that I eigenvector. So, the Eigen vectors change, but I do not want to do this because that takes up a lot of query time ok.

So, then the paper by this and this and this was a this entire formulation was by here way said all. So, what they said is it that they went about it in a very clever way we said that. Let us imagine that the data points are that, we have a really samples from some

distribution and the query is also a sample from that distribution ok. And what I am really looking for is by using the eigenvectors that I calculate from the from the from the query from the training set or from the from the query points or from the data points are given to me, can I create eigenvectors or Eigen functions of the underlying distribution right, that these data points have been drawn from ok. So, how do we do this?

(Refer Slide Time: 24:37)

The slide is titled "Eigenvectors" and contains the following text:

- Assume that the data is coming from some distribution in R^d
 - But estimating this distribution is hard also
 - We could try to interpolate the eigenvectors to query points, under above assumptions, but is computationally expensive (Nystrom extension)
- Assume data distribution is product of uniform distributions
 - Use PCA to find the axes

Hand-drawn in red on the slide is a 2D coordinate system with axes labeled a_1 and a_2 . A rectangular region is drawn, representing a uniform distribution in 2D. The region is filled with red scribbles, and the axes are also drawn with red lines.

At the bottom of the slide, there are logos for IIT Gandhinagar (Indian Institute of Technology Gandhinagar) and NPTEL ONLINE CERTIFICATION COURSES.

So, you could do this is very hard if it is a general distribution. If I do not assume anything about what the distribution of the data points is this? This is a hard this is not a trivial question. In fact, I mean some of you might have heard about it, but we could try interpolating we could try a method of interpolation of the eigenvectors. So, the query points and this is known as Nystrom extension right, but this is also computationally expensive so. In fact, it is as computationally expensive as doing a as doing a naive search itself.

So, we do not want to do this. So, what we instead go about is to say that we make a very simple assumption about what the data distribution is. We see that the data distribution is really just a product of uniform distributions that is what we say is that let us say we are drawing a data distribution in 2 D, we say that in 2 D if the data is 2 D it really comes from product of 2 types of distributions it is it lies between some I mean, it is a uniform distribution between some A and B on one of the axis and a uniform distribution A 1 and

B 1 on this axis and an uniform distribution between a 2 and b 2 on this on this other axis.

So, it is uniform in this rectangle A and B right. And I do not know what this rectangle is, but what we gone assume is that the data is drawn uniformly from such a distribution that is each coordinate is independent right. And then it is there is a particular interval in which it lies independent right. And so, I am going to try to use the eigenvectors of the data points to figure out what the eigenvectors of this underlying or Eigen functions really of the of this underlying distribution r ok.

So, one other sort of twist that, we do is to say that fine, maybe these axes are not known maybe this axes are not the input axes right. What we say more accurately is to is that.

(Refer Slide Time: 26:43)

The slide is titled "Eigenvectors" and contains the following text:

- Assume that the data is coming from some distribution in R^d
 - But estimating this distribution is hard also
 - We could try to interpolate the eigenvectors to query points, under above assumptions, but is computationally expensive (Nystrom extension)
- Assume data distribution is product of uniform distributions
 - Use PCA to find the axes

Below the text, there is a hand-drawn diagram in red ink showing an ellipse with two intersecting lines representing its principal axes. The slide footer includes the IIT Gandhinagar logo and the text "NPTEL ONLINE CERTIFICATION COURSES".

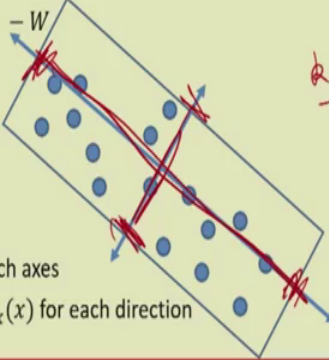
That suppose the data lies is something like this right then let me try to find out some axes right, that first best fit best fit the data and in this axes along this axes, I assume that the data is uniformly distributed.

So, let me draw you a picture first of all let. I will draw you a picture. So, for instance if the data lies something like this is something like this right.

(Refer Slide Time: 27:09)

Algorithm

Create top k PCA of $D - W$



Gives us top k axes
Find the $[a_i, b_i]$ for each axes
and create $\phi_1(x) \dots \phi_k(x)$ for each direction

$k=2$

16

Then what we do right is to first find the axes the main axes of this data the 2 main axes of this data which means the 2 eigenvectors of the of the principle directions right. And then we assume that the data is uniformly distributed along this axes and along this axes and why is that useful? That is useful because going back.


(Refer Slide Time: 27:33)

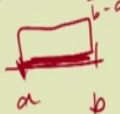
Eigenfunctions

- Take limit of eigenvectors as $n \rightarrow \infty$, and consider the “normalized” similarity matrix (Laplacian)
- Analytical form of Eigenfunctions exists for certain distributions (uniform, $\frac{1}{b-a}$, Gaussian)
- For uniform

$$\Phi_k(x) = \sin\left(\frac{\pi}{2} + \frac{k\pi}{b-a}x\right)$$

$$\lambda_k = 1 - e^{-\frac{2^2}{b-a} \left(\frac{k\pi}{b-a}\right)^2}$$





[Image from Waiss et al]
- Constant time calculation for any new point

17

That as I was mentioning that if we do have the uniform distribution right, in a particular in the in r 1 right we can actually as n tends to infinity we can actually calculate the Eigen functions, which are the limit of the eigenvectors as n tends to infinity for a

particular distribution see. Imagine that you are you have the uniform distribution in a to b and you keep on sampling you keep on sampling these keep on sampling points from here right.

And then you look at the corresponding eigenvector and you let that and you and you consider n tends to infinity. So, that becomes a function right instead of I instead of a vector a function for every point. And if the data distribution is uniform we can actually write down the form of this function. And we takes a very simple form actually right for instance if the data was uniform in a to b right.

So, the height of the data is 1 over b minus a if the data is uniform is drawn from the uniform distribution from a to b , then the Eigen function for this the k th Eigen function for this is nothing but this particular expression $\sin \pi$ by $2 k \pi$ b minus a times x ok. So, notice that it only depends on this interval b minus a on the size of this interval b minus a nothing else and the k th eigenvector value is this quantity ok.

So, so here in this plot I have drawn the 2 D, I mean here, here I was assuming that the points are drawn from uniform in 1 D if the points are drawn in uniform from 2 D the Eigen functions would look like this is kind of a heat map ok. So, so what we will do is then as follows, as I as I mentioned that the input data is given and the target dimensional it is given first we find out the principle the k principle axes of this its. So, these are the top k axes. So, here k equal to 2 ok.

So, and then we find out the a_i b_i for each axis right which are these 2 limits. And these let say in these 2 limits in this direction. And these 2 limits in this direction right and the way you could find this out is let us say I find by finding on the 5 percentile and the 95 percentile of the data in each of the dimensions right.

So, this gives me and now we are assuming that the data along this dimension is uniformly distributed in this region a_i b_i right. So now, along each of the directions we create ϕ_1 ϕ_2 to ϕ_k x ϕ_1 x to ϕ_k x right this gives me k eigenvectors in each direction right and the corresponding eigenvalues λ_1 to λ_k .

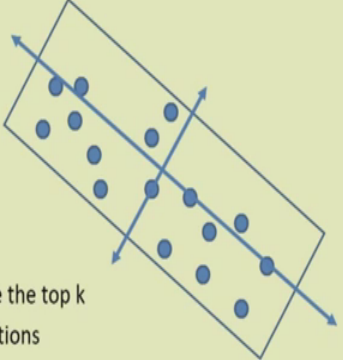
(Refer Slide Time: 30:07)


Algorithm


Create top k PCA of $D - W$

Gives us top k axes
Find the $[a_i, b_i]$ for each axes
and create $\phi_{i1}(x) \dots \phi_{ik}(x)$
and $\lambda_{i1} \dots \lambda_{ik}$ for each direction

Total dk eigenvalues \rightarrow sort and take the top k
eigenvalues and corresponding functions



 IIT Gandhinagar
Indian Institute of
Technology Gandhinagar

 NPTEL ONLINE
CERTIFICATION COURSES

17

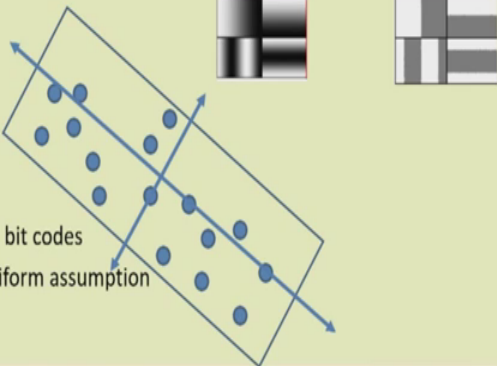
So, that is a total of dk eigenvalues. So, we sort them and take the top k eigenvalues and the corresponding eigenvectors. So, from dk we go down to k ok. So now, I have and now I have a bunch of a bunch of directions and some Eigen functions have chosen in each of them right.

So now, when the query point comes we are going to project the query point on exactly those axes we I mean and we are going to take the Eigen functions that have been chosen and we are going to just plug in the value of the query point the. So, the x of the query point right remember here, here this ϕ_k this ϕ_k function depends on x . So, we plug in this value and then we quantize it and then we get the bit value of the query point for this direction the k th bit value of the query point.


(Refer Slide Time: 31:07)


Algorithm

Threshold chosen
Eigenfunctions



Empirical observation: bit codes
seem robust to the uniform assumption

 IIT Gandhinagar
Indian Institute of
Technology Gandhinagar

 NPTEL ONLINE
CERTIFICATION COURSES


18

So, the algorithm in it is essence is very simple right. It just involves taking principle components and then and then doing a very simple calculation.


(Refer Slide Time: 31:13)


Results

- Shown to have better properties than naïve LSH on large datasets



[Image from Weiss et al]

 IIT Gandhinagar
Indian Institute of
Technology Gandhinagar

 NPTEL ONLINE
CERTIFICATION COURSES

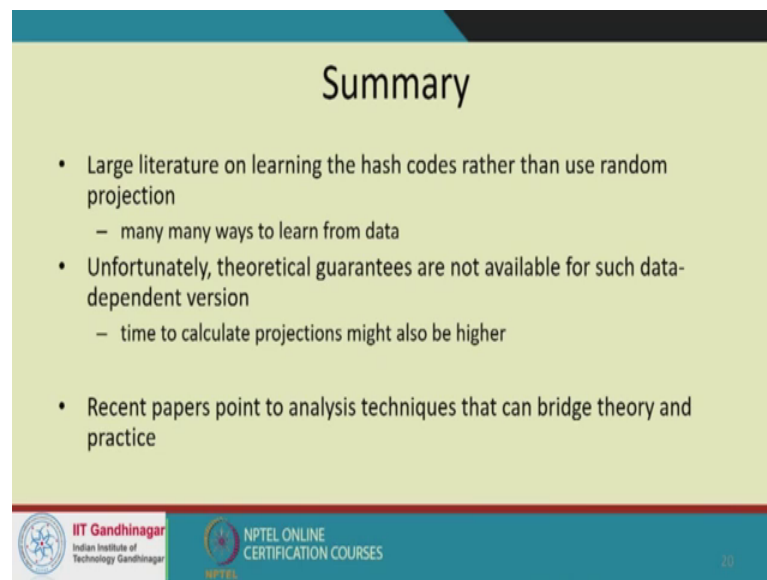
19

It turns out that this is surprisingly effective. So, here is I mean here is another plot taken from the paper, in which they actually take a very a pretty large data set called a label me data set and they compare naïve Lsh. So, this is the naïve Lsh algorithm this is a spectral hashing algorithm with only 10 bits, this is yet another machine learning algorithm and we are plotting the number of bits versus the versus in some sense the what fraction that

if I take a hamming distance of less hamming distance of 2 right what are the good neighbors that, that captured right within that actually fall within a hamming distance of 2.


In some sense we recall at to recall at to and they and they and it shows that the recalled for spectral hashing is much much higher than that of lsh they also show a bunch of anecdotal examples.


(Refer Slide Time: 32:05)



Summary

- Large literature on learning the hash codes rather than use random projection
 - many many ways to learn from data
- Unfortunately, theoretical guarantees are not available for such data-dependent version
 - time to calculate projections might also be higher
- Recent papers point to analysis techniques that can bridge theory and practice

 IIT Gandhinagar
Indian Institute of Technology Gandhinagar

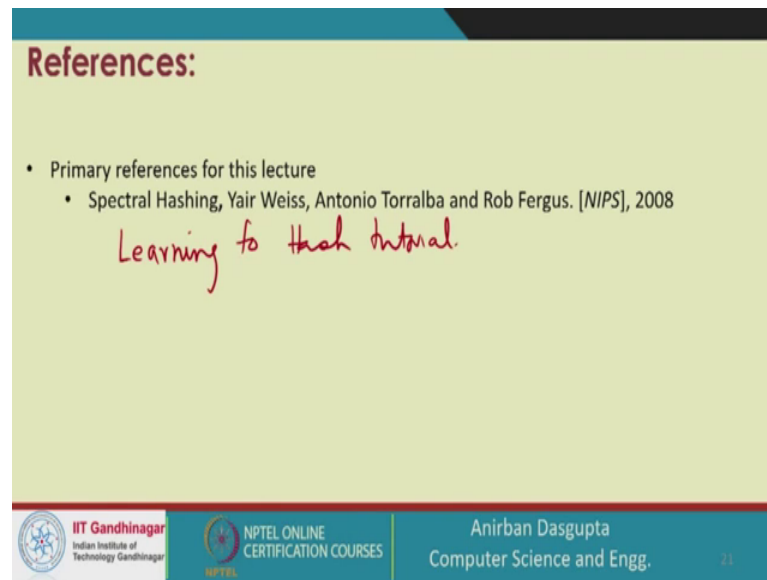
 NPTEL ONLINE
CERTIFICATION COURSES

20

And there since there has been a large literature on learning the hash codes rather than using random projections and the many different slightly variations variations of the ways this has been formulated. Unfortunately, most for most of them theoretical guarantees are not available for such data dependent versions of the lsh right.

I mean there are some recent papers that point to analysis techniques that bridge theory with practice and this is an interesting research direction. Another issue is that the time to calculate projections is also higher right because it takes a lot more time to calculate pca than to actually do any projections, but this is something that we will also look at in the next in the next few lectures of the course, that how can we calculate pca in a much more efficient manner.

(Refer Slide Time: 32:50)



References:

- Primary references for this lecture
 - Spectral Hashing, Yair Weiss, Antonio Torralba and Rob Fergus. [NIPS], 2008

Learning to Hash Tutorial.

IIT Gandhinagar
Indian Institute of Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

Anirban Dasgupta
Computer Science and Engg.

23

Finally, the primary reference for this lecture is spectral hashing is the spectral hashing paper by Weiss Torralba and Fergus. There is also a very nice set of lecture notes I mean lecture notes comes slides on learning to hash, if you search for learning to hash tutorial and we will also put this up on our course home page the very nice set of pretty extensive survey a hand.

Thank you.