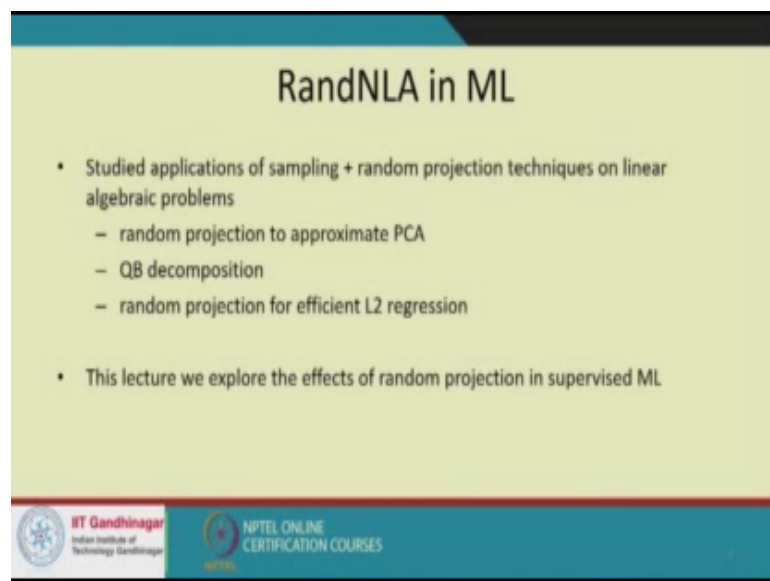


Scalable Data Science
Prof. Anirban Dasgupta
Department of Computer Science and Engineering
Indian Institute of Technology, Gandhinagar

Lecture – 16 c
Feature Hashing

Welcome to the course on Scalable Data Science, my name is Anirban and I am from IIT Gandhinagar. So, today's lecture is going to be on Feature Hashing.

(Refer Slide Time: 00:27)



RandNLA in ML

- Studied applications of sampling + random projection techniques on linear algebraic problems
 - random projection to approximate PCA
 - QB decomposition
 - random projection for efficient L2 regression
- This lecture we explore the effects of random projection in supervised ML

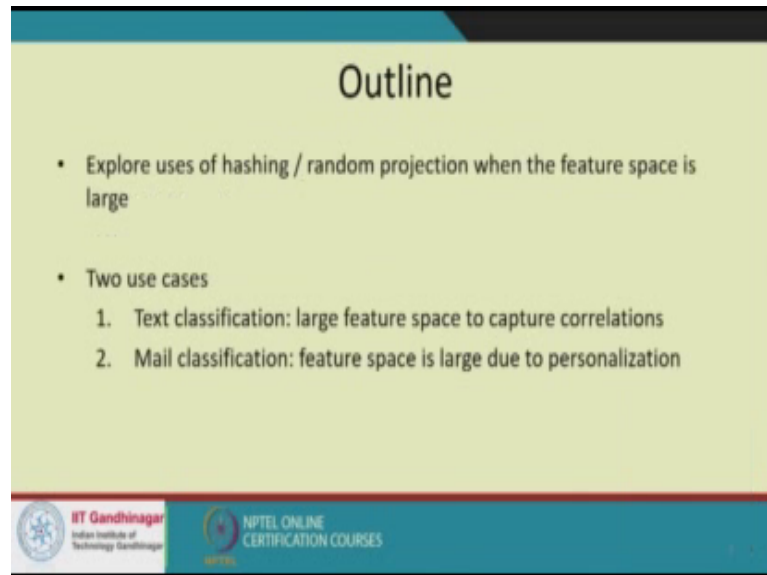
IIT Gandhinagar
Indian Institute of Technology Gandhinagar NPTEL ONLINE
CERTIFICATION COURSES

So, until now you have studied applications of both sampling and random projections to a bunch of optimization problems relating to linear algebra right. We have we first saw applications of random projections to this low rank approximation, here of at least two different forms, to matrix multiplication, to L 2 regression. And, then we and in the last lecture you also saw how we can turn some of these how we can also imply sampling to solve some of these problems right using the leverage score sampling techniques.

So, all of a lot of these one big thing that is missing, while a lot of these is relevant in a unsupervised machine learning right, the supervised is a slightly different ballgame ok. And they in this lecture well explore how random projection and related techniques can also be useful in supervised machine learning right. The basic techniques and the intuition is going to be the same right, the problem is going to come from that of

supervised machine learning. Therefore, and what we will discuss at the end is what a new kind of guarantees can be should we hope to have in this case.

(Refer Slide Time: 01:43)

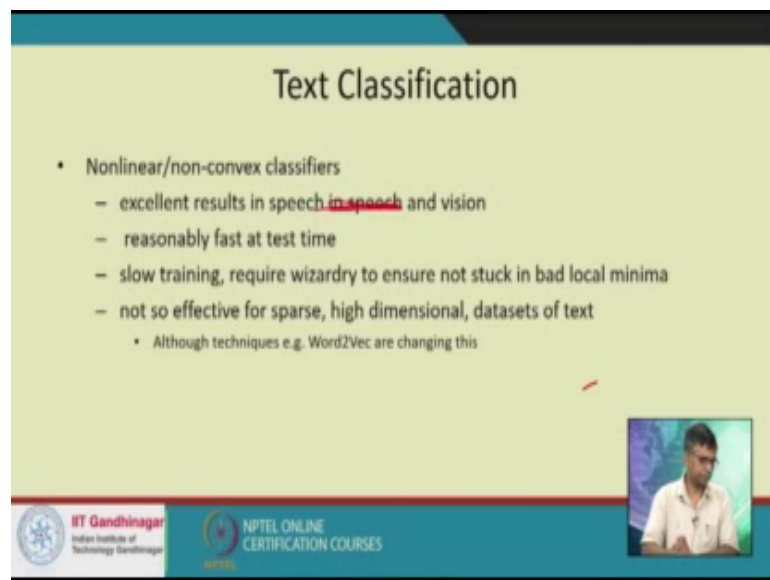


So, let me give the outline first, there are going to be two different use cases both of them are relating to text. In one use case, what we'll consider is that there is going to be a very large feature space right and we will point out why having a large feature space is actually useful. Essentially it is going to be to capture very long range correlations in the in my document in a specific document. And, we will have to come up with some technique to control this the size of this feature space right and I mean the large, having a large feature space is actually useful for classification. However, I mean in a is actually useful in improving classification accuracy, however, it is computationally very inefficient.

Therefore, what we will do is that first we will pose the problem as having a large feature space, and then we will use random projection to cut down the feature space while preserving the benefits, some of the benefits that we had brought in because we introduced the large feature space right. So, in this way we will sort of I mean do both right; a similar story is going to come out in mails file classification. Here what we are going to see is that we have to introduce a large feature space in order I mean as a cheap way of trying to give personalized classifiers to each individual user.

And again this is going to blow up the entire dental problem size by a lot right and therefore, we are going to use random projection related technique to cut it down right. And what we will see in that in both these in both these settings are very specific type of random projection is going to be more useful to us.

(Refer Slide Time: 03:31)



The slide is titled "Text Classification" and contains the following text:

- Nonlinear/non-convex classifiers
 - excellent results in speech in speech and vision
 - reasonably fast at test time
 - slow training, require wizardry to ensure not stuck in bad local minima
 - not so effective for sparse, high dimensional, datasets of text
 - Although techniques e.g. Word2Vec are changing this

The slide also features logos for IIT Gandhinagar and NPTEL ONLINE CERTIFICATION COURSES, and a small video inset of a speaker in the bottom right corner.

So, let me outline the two use cases first. So, in the text classification setting right let us think about what kind of classifiers work best right; while nowadays you see a lot of this non-linear non convex classifiers I mean the deep neural nets that give excellent results, in speech and vision right. There is a typo out here and these are good because they have very high accuracy, they are also reasonably fast at test time. However, during training time they can be terribly slow right and often require a lot of thinking a lot of domain expertise in order to ensure that you do not get stuck in bad local minima right there are lot of heuristics, that you need to know about.

Furthermore it is not clear that a lot of these I mean these non-convex classifiers are very useful when the data is sparse and high dimensional right i. e. when the data comes from text right. Although there are techniques I like what to vet that are changing this that are creating dense representations of word doc of sort of very sparse documents and words and then and thereby using it. However, it is still not clear that they are that they can beat the I mean very practical systems that are robust right, that are I mean that are formed by much simpler means ok.

(Refer Slide Time: 04:58)

In practice: count based features [John Langford]

- In practice, lot of features are based on "normalized counts" of tokens
 - easy to train, at least on single machine
 - fast at test time: calculate some statistics and then predict
- Works great for settings e.g. text classification
- A common practice, in order to capture higher level correlations is to take various combinations of n-grams and skip-grams
 - Captures high level correlations, still efficient to calculate

IFT Gandhinagar
Indian Institute of
Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

So, in practice what is what is still very useful are what I call count based features right. And these slides are actually from a talk by John Langford ok. So, in a sense what I mean is that I mean if you remember the normalized or rather the back of words model of a document right in which we essentially represent a document as a set of its tokens along with the counts, that are associated with these tokens. And, these counts could represent some normalized frequency in the document frequency things like this.

We could come up with the normalization with the term count in very fancy ways right, we could take this pack of words model and we could augment it quite a bit right. So, the benefit of the simple bag of words model is that, it is really easy to train at least on single machine ok. And paralysation is a different ball game altogether it is also clear that its really fast at this time right because, all you need to do is essentially compute some statistics from this from the query document and then use that for prediction right.

And it turns out that this still works surprisingly well for text classification right, which is which still forms a large part of the machine learning and practical machine learning tasks that we do right. And a common practice I mean if you want to augment the naive bag of if you think that a bag of words model itself is it is knife to start with, a common practice to augment it is to take various combinations of n grams and skip grams right. And the intuition is that, that if I take a I mean if I take bigrams trigrams etcetera and scale grams well be able to capture high level correlations. But still they are fairly

efficient to calculate and therefore, we are capturing the high level correlations, while being while retaining the efficiency of feature construction ok.

(Refer Slide Time: 06:53)

Example

"the rain in Spain falls mainly on the plain"

2-grams:
the rain, rain in, in Spain, Spain falls, falls mainly, mainly on, on the, the plain.

1-skip-2-grams: ✓
the in, rain Spain, in falls, Spain mainly, falls on, mainly the, on plain.

→ Can also look at this as a kernel representation
→ Feature space become very high dimensional, how to deal with it?

IFT Gandhinagar
Indian Institute of Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

Let me give an example of bigrams and n grams. So, for instance here is the sentence that we have the rain in Spain falls mainly on the plain ok. So, the two grams in this and this in so, of course, the unigrams in this in this particular document, would be would be the individual words the rain in Spain falls mainly in the plain right. The 2 grams would be the rain raining in Spain Spain falls etcetera. And what you can easily see is that by looking at the this sort of unigrams you are getting a better idea of what the document is about.

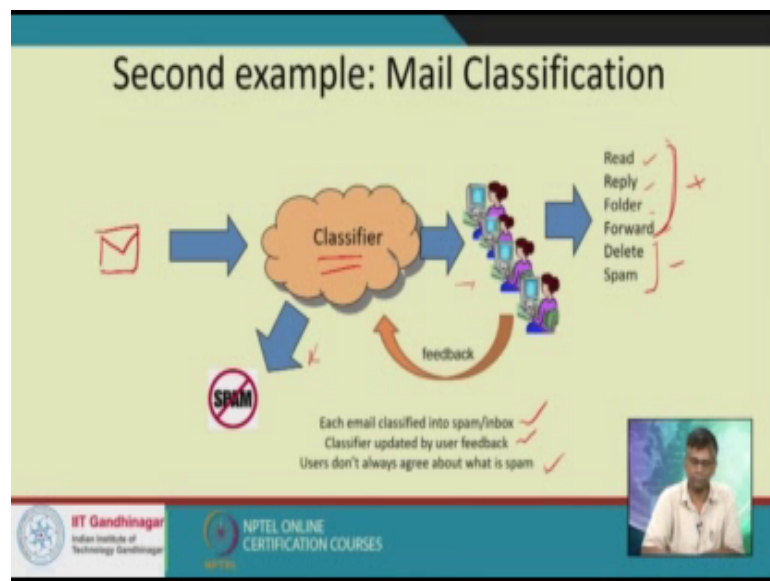
So, that the skip grams that we that I just mentioned has various has different variants to it right. Here is one particular variant there is known as a 1 skip 2 gram right, that is you take bigrams right where between 2 bigrams that you take you just skip one of the bigrams. For instance you take the and in right similarly you take rain and Spain right, you take in and false and so on and so forth right. And the point is that is that if you take a very a number of such variations of this 1 skip 2 grams to skip 3 grams and so on and so forth you captured you essentially capture this collection this.

So, first of all these set of features are trivial to automatically construct you are not sitting down and thinking about special feature engineering and secondly, once you throw in all these features, they actually capture a lot of semantics of the document right.

An interesting way of looking at this is that I mean if I sort of up in take a representation of the document in this bloated up feature space, and then calculate the similarity of this you can also think about it at some kind of kernel representation of the document right.

Essentially instead of instead of simple features we are taking combinations two combinations, three combinations, four combinations of these features. The only problem that I see right now is that all these are good, but the feature space is becoming very high dimensional. For instance if you take a few variants of the subsets of this unigrams trigrams and then and then 1 skip 2 grams and so on and so forth then the number of then the number of possible features becomes let us say if it is a 3 gram it becomes the this the cube of the size of the image dictionary which is a lot. So, how should we deal with this ok?

(Refer Slide Time: 09:14)

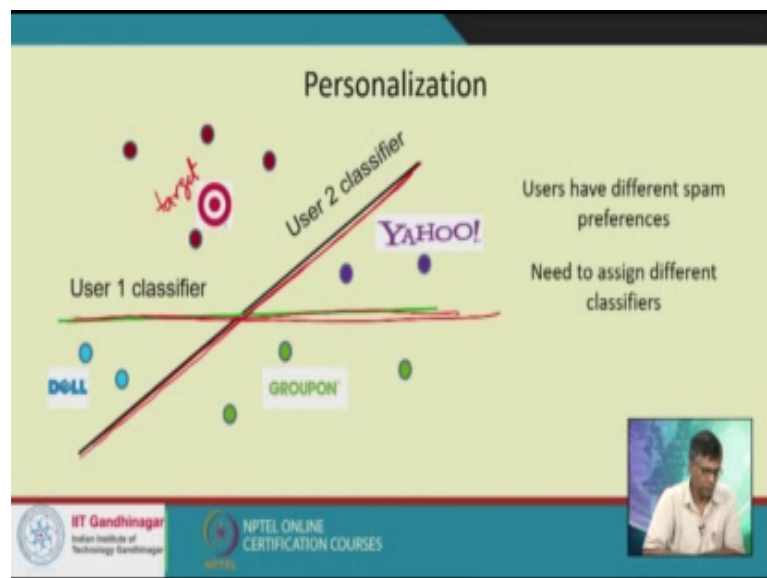


So, for the second example let us look at the problem of mail classification right and here is the problem. That is suppose that suppose a piece of mail comes to us right and now we have to decide, whether to put it in your inner in a particular users inbox or in the spam box ok. Now, there are different and I mean in order to do this all the standard online email systems. So, think of our webmail system have their own classifiers that they have trained right and these classifiers they I mean based their decision they put it in the they decide to put it in the inbox or in the spam box and now the classifier also gets

feedback from the users right. So, the user can possibly read, reply, forward, delete, forward, and then I mean or she could delete or mark the user a spam right.

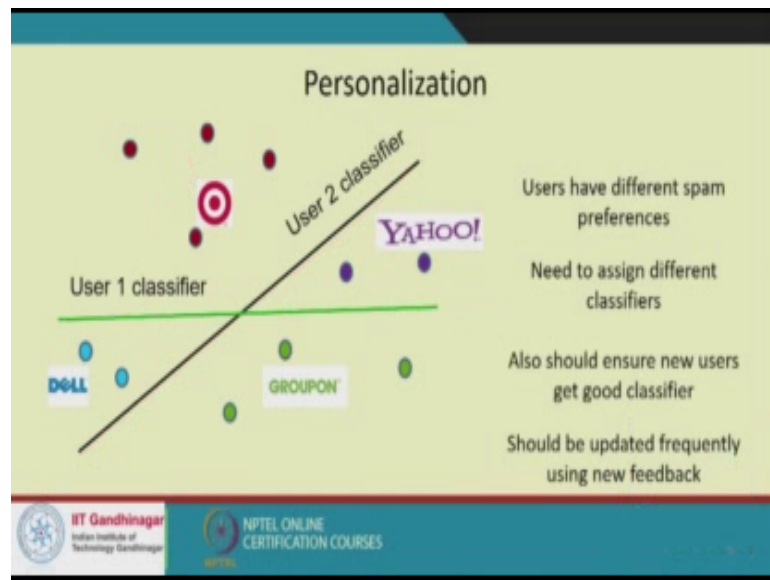
So, it is possible that I take these as the positive feedback and this as a negative feedback on the email right. And based on this feedback I can retrain my classifier ok. So, the things to keep in mind is that each email needs to be classified in the spam box or inbox and therefore, this has to be done in basically real time. A classifier has to be updated by user feedback, which means that it has to be updated frequently because, the nature of spam keeps on changing and users do not always agree about what is spam right.

(Refer Slide Time: 10:32)



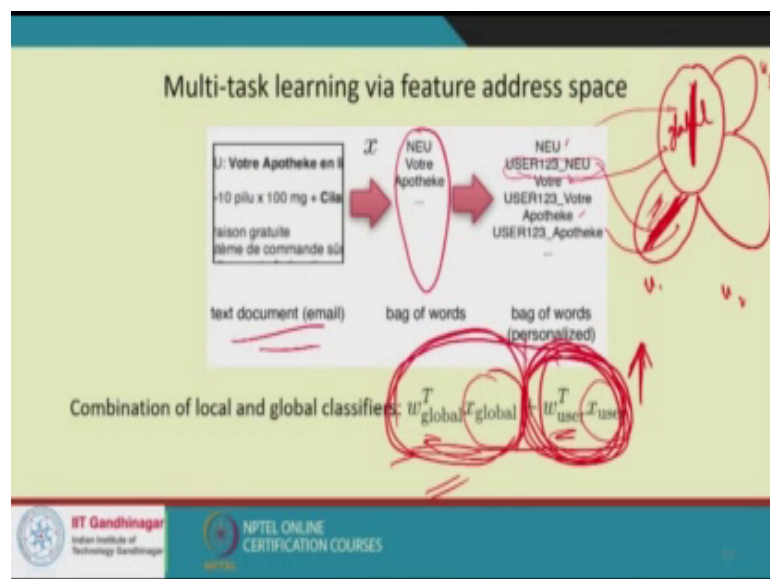
So, in machine learning terms, this is in order to satisfy those requirements this is what we have to do right. So, suppose we are presenting representing each email as a point in some high dimensional space. So, these are the emails, this points are the emails ok. And now and every users classified representing as a vector in this high dimensional plane right users 1 classifier users 2 classifier. So, first I say that users have different spam preferences. So, what it means is that the classifier for each user is slightly different right. So, the user 1 can potentially consider the emails from Groupon and Yahoo to be spam, but maybe user 2 considers this company, this is a company called target and Yahoo to be spam, but not Groupon ok.

(Refer Slide Time: 11:22)



So, we need to assign different classifiers to each user, which means that we need to store a different hyper plane for each user. We also need to ensure at the same time we also need to ensure that new users get a good classifier right. Because new users, for new users we do not have that many feedback from the user. But we still need to guarantee that we get a reasonably good classifier for the user right. Furthermore we want to update this classifier currently using new feedback ok. So, how can we achieve all these? So, here is one particular very interesting way of doing all this.

(Refer Slide Time: 11:51)



What we do is that we say that let us blow up the feature space right and how do we blow up the feature space? Let us take the email right and we do sort of standard bag of words or unigram or bigram decomposition and we get this and we get this unigrams or bigrams from the email right. Now in order to sort of one way now what we do is that in order to assign we assign every user their own feature address space right. What it means is that every I mean the global set of features, this is this global set of features and then there is a set of features that belong to user 1 then there is a set of features that belong to user to belong 2 user 3 and so on and so forth.

So, this is global set of features right and how do we implement this? We implement this it is simply, we take all the tokens right and then we take all the tokens by themselves we these belong to the global feature space right. We just prefix the name the token by the name of the user right and these then belong to the specific to the user specific feature space. So, notice that only the emails that u 1 gets right I mean that belong to u 1 will have the I mean will have tokens that are specific to u 1 right.

The emails that the same email if u 2 gets it would not have it would not have this particular token right because it is prefixed by the user 2's it is going to be prefix by that user 2's name not by user 1's name ok. And now what well do? What we say that that let us learn a classifier on this entire feature space right. It is not very hard for you to see that what this corresponds to is that is that for a for every user we are learning a classifier of the following form. That a classifier which has a component in the global in the I mean a set of coefficients for the global feature space and a classifier which has set of coefficients for that for the user feature space right.

So, therefore, for a particular email for a test email right this I mean for the classifier does is as follows, that it takes it I mean let me call this w_{global} this is w_{global} and the set of coefficients here to be w_{user} right. So, therefore, for a test email, we again do sort of representation of the email in the global space and in the user space and then we take the corresponding dot products. And therefore, what it turns down to is that we sort of and we could apply our threshold on this and therefore, what it says is that we are combining some a global classifier, with a user specific classifier and we are giving use and we are giving a particular user the combination of these two classifiers right I mean it is entirely possible that.

So, why is this good? This is good because maybe it is a considered particularly user who has just come into the system. So, for that particular user the I mean the w transpose user is 0 right because I have not yet trained user specific classifier, because the user has not given any feedback right therefore, there would not be any token on the on the on the on the user specific part ok. So, but then, but for this user we still have the global component right and we are still and because that is trained by everybody's feedback right.

That is a reasonably good classifier it is not clear it is not very tailored to that users needs, but it is fairly its fairly good ok. So, therefore, if I start of this new user with his global classifier, she has a reasonably good experience there are and they are all on onwards this particular user can start providing feedback if she wants to and therefore, the value of this the importance of this part of the of the classification technique increases. And, so there might come a time if the user provides enough feedback that this part of this part of the of the of the this part of the classification, really has more importance than this part which is the global part. And then the user will start getting them will start sort of acquiring the benefits of personalized classifier ok. So, this is a very natural way of going from a from a global initialization to a personalized classifier for more involved universe ok.

(Refer Slide Time: 15:59)

Multi-task learning via feature address space

$w_{global}^T x_{global} + w_{user}^T x_{user}$

N_{user}
 D
 $current: ND$

Dimension of resulting space blows up

IIT Gandhinagar
 Indian Institute of Technology Gandhinagar

NPTEL ONLINE
 CERTIFICATION COURSES

What is the bad thing about it right? And obvious bad thing about it is that the size of the dimension the resulting dimension space has blown up a lot, because if you if imagine that if I have n users right if D is the if D was the size of the of my initial dictionary my current set of dimensions is N times D right. Because I am multiplying the I am multiplying the user by the I am multiplying the number of possible size of the possible dictionary by the number of users and this is potentially a huge right.

Because while the while the emails are sparse the vectors w and sparse. So, therefore, keeping all these vectors the w in this and I mean maintaining this vector w in this particular space is fairly expensive right. So, maintaining even a single w which is what you have to maintain in this in this space of size ND is very very expensive it is impossible.

(Refer Slide Time: 16:57)

The slide is titled "Multi-task learning via feature address space". It shows a matrix with columns for a global dictionary and user-specific dictionaries. The global dictionary has features like "V. Votre Apotheke en l" and "NEU Votre Apotheke". The user-specific dictionary for "USER123_NEU" has features like "NEU USER123_NEU" and "NEU". A blue box lists requirements for a compression method: "Want a method to 'compress' these features into available memory", "should be linear (makes updates easy)", "should have guarantees", and "should maintain sparsity as much as possible (personalized)". At the bottom, it shows the equation for a combination of local and global classifiers: $w_{\text{global}}^T x_{\text{global}} + w_{\text{user}}^T x_{\text{user}}$. Logos for IIT Gandhinagar and NPTEL ONLINE CERTIFICATION COURSES are at the bottom.

So, what can we do? So, if we want to compress this features into a into memory right and the way to compress. And what we want is that this compression should be linear why is this linear? It makes updating easy right because if you remember that we want to update the classifier constantly right and one of the ways of doing the update is by doing some kind of a gradient descent.

For instance a sarcastic gradient descent that that when you get a new email or a batch of emails with the user feedbacks on them you just do you just do a linear update on the I mean you just do gradient update on the classifier ok. Ideally it should have guarantees

right and should maintain sparsity as much as possible right that I do not want I mean the emails themselves are; obviously, sparse data I and I do not want to densify it by too much ok.

(Refer Slide Time: 17:43)

The slide is titled "Hashing" and contains the following text and diagram:

- Text: "Create a hash-table that hashes features into buckets"
- Text: "Are there principled ways of thinking about this?"
- Equation: $h(D) \rightarrow (d)$
- Equation: $a_1 + a_k + a_i$
- Diagram: A large array of size D with elements a_1, a_k, a_i is mapped to a smaller array of size d . The mapping function is $h(D) \rightarrow (d)$.

So, so how can we do this right? So, here is an obvious way that suppose we create a hash table that hashes features into buckets right. So, what I mean is as follows; that suppose I have and this is my original set of features right, which is which essentially very large right, the dimension is very large, but it is sparse. So, maybe I have some values out here C plus 1 plus 2 and then plus 3 and so on and so forth.

So, this dimension is really large let me call it capital D . So, what I could easily do is that is that hash it into a small number of dimensions small d right. So, so all that I need is that I need a hash function that goes from capital D to small d right and I can put it in here I can put it in here right. So, basically then a bunch of features can collide. So, a bunch of features might go to the same position in the smaller in this in this in the smaller representation, and what I will just do is basically add up put in half a counter out here that is the addition of these counts.

So, suppose these three these three features go into the same bucket in this using this hash function, then the value out here will be the let me call this a 1 let me call this a k and let me call this a i . So, the value out here is a 1 plus a k plus a i right that is the value that you put in here this could be one way of doing it right. The question is that is there a

principle better simple way of thinking about this right sure is this enough, should I be doing something else right and the what about the guarantees you because we were promising some bounds with the I mean we were promising some algorithms that comes with some guarantees right.

(Refer Slide Time: 19:29)

Hashing

Subject: projection

$s \sim \text{Bern}(\pm 1, \frac{1}{2})$

Create a hash-table that hashes features into buckets

Are there principled ways of thinking about this?

- This is essentially SparseJL, with only one hash being used!
- We should take guidance from principles of designing it, as well as draw from theoretical guarantees
- In particular, should use a sign hash to make this unbiased

IT Gandhinagar
Indian Institute of Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

Now if you have paid attention in this course yeah you should it should not be very hard for you to realize that this is essentially Sparse Johnson linear transformation right. We were transforming from one particular vector to a smaller set to a smaller set right by using a hash function right, except that here we are using only a single hash function right instead of using multiple hash functions. So, I could think of it as a hash function, as a hash transformation or I could think of it as a random projection that preserves sparsity ok. So, then in that case we should really take guidance from the principles of designing a random projection.

As well as and if you do this well be able to draw upon the theoretical guarantees random projection comes with right which we have seen is really strong. And in particular one of the lessons that you should draw is that it is not enough to say supposing this is again a 1 a k a i and let us say all of these have gone to the have gone to the same bucket. In particular it is not enough to say to put a 1 plus a k plus a i in the bucket right because remember what SparseJL was doing SparseJL was adding a random sign to each of them was multiplying each of them by a random sign. So, I should do s 1 times a 1 you pick

keep this as it is. So, instead I should do s_1 times a 1 plus s_k times a k plus s_i times a i , where each of the s_i is a binomial is a Bernoulli plus minus one with probability half.

So, why is this important intuitively? Intuitively this is important because it helps the counts out here be unbiased estimators of the counts of the original set of counts ok. So, so this is one of the and once we do this, we get exactly this Sparse Johnson linear transformation with the I mean with the sparsity 1 . And in fact, we have seen this variant this is what we called as a subspace projection right and we saw that, it does preserve some very nice properties for the subspace reservation ok.

(Refer Slide Time: 21:38)

The slide is titled "Using in practice" and contains the following bulleted list:

- Theoretical guarantees assume a k -wise independent hash function
- Feature hashing now a standard api in many machine learning packages
- Use a standard, fast, hash function
 - Vowpal Wabbit uses MurmurHash3
- Standard procedure:
 - Hash input data, learning classifier over hashed space
 - When testing, hash test point first and then apply classifier
 - Collision is fine, that's the point
 - Beyond this, use domain knowledge if we want to ensure specific "super important" features or groups of features

Handwritten notes in red ink include "roughly" in the top right corner and circles around "MurmurHash3" and "Collision is fine, that's the point". A small video inset shows a man speaking.

So, how do we use it practice ok? So, the algorithm seems fairly clear right in terms of the theory. So, what do we do in practice? See the theoretical guarantees, we assumed all independent hash functions or at least some key wise independent hash functions for some value of k ; typically k is like $\log \frac{1}{\delta}$ right by δ right.

So, in practice we do not really create k wise independent hash functions by ourselves. What we do is that we typically use standard fast hash functions. There are deeper reasons for doing this as to as to why this should work. So, people have thought about it and there are some theoretical guarantees, but in, but basically the idea is that there I mean given the given that the practical data is not entirely adversarial chosen, you do not really have to guarantee complete randomness perfect randomness in your in your hash function.

It is enough to choose a practical fast hash function for instance there is a particular software called Vowpal Wabbit right that actually implements this I mean gives you a very fast classifier using this feature hashing, and it uses and it uses something known as a murmur hash 3. And in fact, you can use a lot of the a lot of APIs standard API I mean machine learning, packages official have feature hashing have a as a standard API including ones that including packages that run on scalable systems.

Like Hadoop and spark ok. So, then given I mean that ideally if you I mean if you are using one of these machine learning packages the suggestion is that you should use definitely use that feature hashing API that they have provided. If you are using implementing your own classifier, you should use one of the standard fast hash functions like murmur hash right and this is what you typically do right.

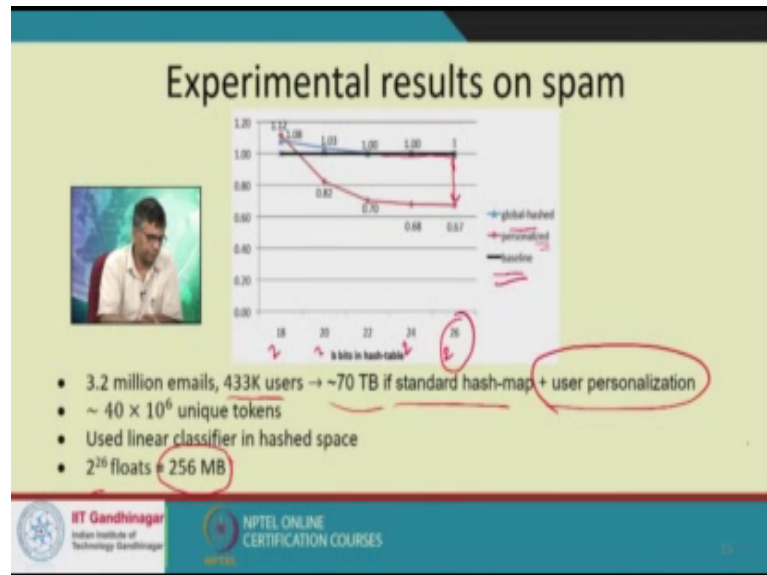
That you create your let us say you are doing text classification, you create your I mean you find out a representation of your document in terms of the engrams bigrams etcetera, then you hash your input data and then you learn your classifier over this hashed space ok. So and when testing we first hashed the test point using hash functions that we have used. So, it is very important that you that you use the same hash functions that you use it, do not change the random seed and then you apply the classifier collision will happen of course, collision will happen, but that is exactly the point.

That we are using these collisions we; I mean using these collisions to compress the set of features into a much smaller much smaller dimension, while preserving a lot of the geometry of the original set of data points right. And the kind of geometry that they preserve is what we have seen for Johnson linens transformation right. That it preserves moral is approximately the pair wise distance of the data points and therefore, if two classes are separable it reserves the susceptibility right.

So, this is the standard way of implementing; it beyond this you should use a domain knowledge for instance if you know that there are super important features that you do not want to collide with anybody or there are groups of features, such that you do not want this set of features to collide with this set of features, but maybe collisions among this set of features is fine right then you should design your hash function in an you should basically partition your hash the range of your hash functions in an appropriate

way right. And all and for all that you need to think carefully about the problem setting ok.

(Refer Slide Time: 24:57)



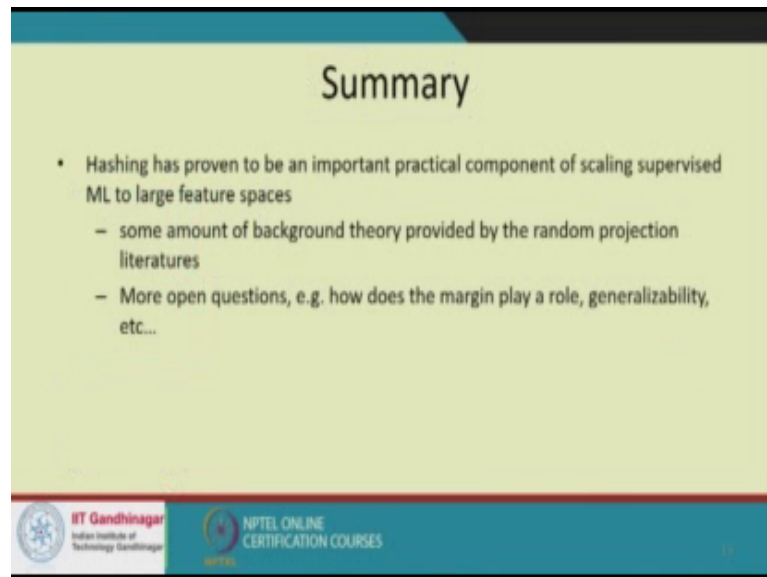
So, before we end I will show you a bunch of experimental results on the for spam classification from a particularly from a paper that we had. This is this is a result that took 3.2 million year emails from yahoos data and about 400 key users right. So, if we used the user personalization technique that I had mentioned as well as the stand a standard hash map it would have taken us 70 terabytes.

Just to store this right because there happens to be around 40 times 10 to the 6 unique tokens in this in this data right what we did was that, we sort of used these hash functions in which in which we controlled in which we control that the size of the size of the size of the of the hash space right. And then we say that the hash space is only going to contain 2 to the 26 floats and therefore, the maximum size is 256 MB right and on the x axis, you see the number of bits in the hash table and therefore, the size of the hash table is 2 to the power of this right.

So, this is 256 MB instead of the 70 terabytes right and on the y axis you we have plotted the relative classification error; that means that the black one is the baseline classifier; where there is a single classifier that is assigned to everybody right. If we assign I mean if we hash happened to hash the classifier that performance is; obviously, a little worst right, but it soon boils down to the performance of the standard baseline classifier. If you

assign everybody their own persons classifier we get a huge boost more or less like 30 percent I mean 30 percent decrease in the error rate ok. So, this is actually something that is very practical.

(Refer Slide Time: 26:49)



Summary

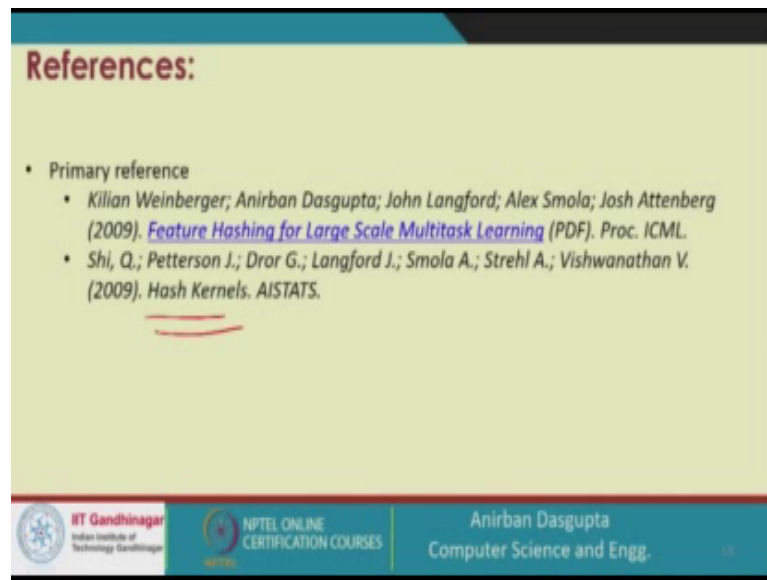
- Hashing has proven to be an important practical component of scaling supervised ML to large feature spaces
 - some amount of background theory provided by the random projection literatures
 - More open questions, e.g. how does the margin play a role, generalizability, etc...

IIT Gandhinagar Indian Institute of Technology Gandhinagar NPTEL ONLINE CERTIFICATION COURSES

So, just to summarize hashing has proven to be an important practical component in order to scale supervised machine learning the large feature spaces some amount of background theory for this is provided by the random projection literature right. That this gives us the other random projection literature gives us intuition, that what were really doing is preserving the structure the pair wise geometrical structure of the original space. And therefore, the and therefore, it is useful in class in the supervised setting because the classes remain separable if they were originally separable.

However, it is actually a lot of open questions for instance, if the if the margin is large how does that play a role in deciding the size of the target hash function right what about generalizability what kind of generalizability bounds can we can we actually prove right. Can I get do I really need the entire the entire strength of the random projection or can we sort of do with much smaller number of dimensions. So, these are all questions does this generalize to non-linear in the non-linear setting for instance. And these are all questions that people are pursuing for research ok.

(Refer Slide Time: 28:03)



References:

- Primary reference
 - Kilian Weinberger; Anirban Dasgupta; John Langford; Alex Smola; Josh Attenberg (2009). *Feature Hashing for Large Scale Multitask Learning* (PDF). Proc. ICML.
 - Shi, Q.; Petterson J.; Dror G.; Langford J.; Smola A.; Strehl A.; Vishwanathan V. (2009). *Hash Kernels*. AISTATS.

IT Gandhinagar
Indian Institute of Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

Anirban Dasgupta
Computer Science and Engg.

So, the primary reference for this are is a paper of us that is called Feature Hashing for Large Scale Multitask Learning as well as the original feature paper that introduced Hash Kernels by Shi Petterson, Dror and Smola and Strehl and Vishwanathan that came out AISTATS and then finally.

Thank you.