

Case Study 1: Estimating Click Probabilities

Tackling an Unknown Number of Features with Sketching

Machine Learning for Big Data
CSE547/STAT548, University of Washington

Sham Kakade

April 6, 2017

©Sham Kakade 2017

1

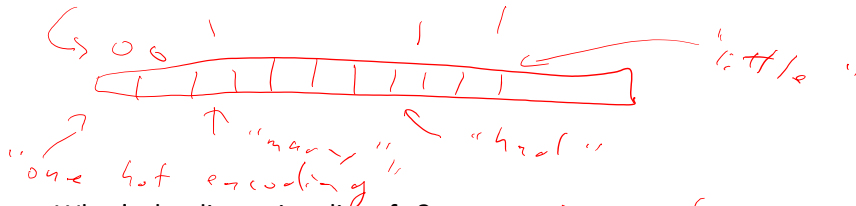
Announcements:

- HW1 due next week
 - updated TA office hours
 - Project Proposals due tomo:
 - ‘big data’ questions v.s. ‘real data’ questions
 - Today:
 - Review: bloom filter
 - Sketching counts; Hash kernels
- MW 3:30-4:30*
CSE 406
+ tomo (posted soon)
- must be novel
(state overlap to
research /
o free projects)

©Kakade 2017

Problem 2: Unknown Number of Features

- For example, bag-of-words features for text data:
 - “Mary had a little lamb, little lamb...”



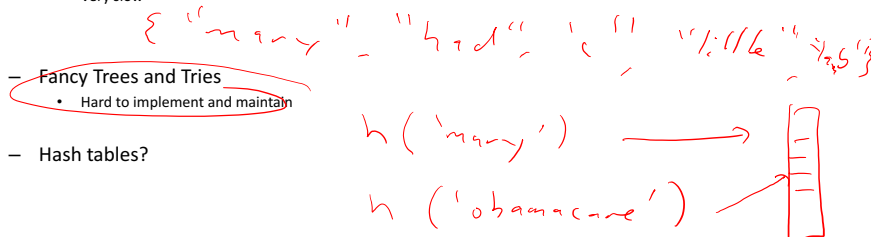
- What’s the dimensionality of \mathbf{x} ? ← size of vocab.
- What if we see new word that was not in our vocabulary?
 - Obamacare
 - Theoretically, just keep going in your learning, and initialize $\mathbf{w}_{\text{Obamacare}} = 0$
 - In practice, need to re-allocate memory, fix indices,... A big problem for Big Data

©Sham Kakade 2017

3

What Next?

- Hashing & Sketching!
 - Addresses both dimensionality issues and new features in one approach!
- Let’s start with a much simpler problem: Is a string in our vocabulary?
 - Membership query
- How do we keep track?
 - Explicit list of strings
 - Very slow
 - Fancy Trees and Tries
 - Hard to implement and maintain
 - Hash tables?

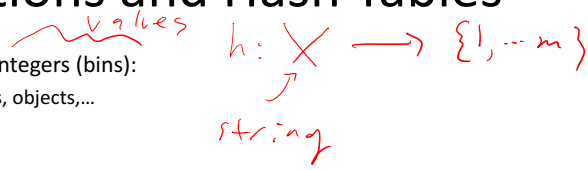


©Sham Kakade 2017

4

Hash Functions and Hash Tables

- Hash functions map **keys** to integers (bins):
 - Keys can be integers, strings, objects,...



- Simple example: **mod**

$h(i) = (a \cdot i + b) \% m$

a = 7, b = 11, m = 32

i = 4, h(i) = 39 % 32 = 7

- Random choice of (a, b) (usually primes)
- If inputs are uniform, bins are uniformly used
- From two results can recover (a, b) , so not pairwise independent -> Typically use fancier hash functions
- Hash table:
 - Store list of objects in each bin
 - Exact, but storage still linear in size of object ids, which can be very long
 - E.g., hashing very long strings, entire documents

©Sham Kakade 2017

5

Hash Bit-Vector Table-Based Membership Query

- Approximate queries with one-sided error: Accept false positives only
 - If we say no, element is not in set
 - If we say yes, element is very to be likely in set

- Given hash function, keep binary bit vector v of length m :



- Query $Q(i)$: Element i in set?

$V(h(i)) = 0 \Rightarrow Q(i) = 0$

- Collisions: $V(h(i)) = 1 \Rightarrow Q(i)$ probably yes

example 'head' & 'obamacare' collide

- Guarantee: One-sided errors, but may make many mistakes
 - How can we improve probability of correct answer?

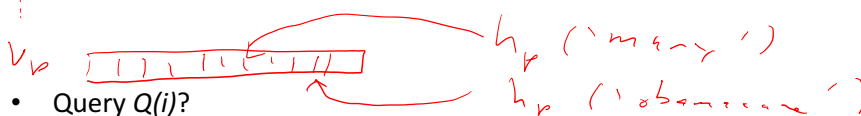
©Sham Kakade 2017

6

Bloom Filter: Multiple Hash Tables

- Single hash table \rightarrow Many false positives
- Multiple hash tables with independent hash functions

– Apply $h_1(i), \dots, h_p(i)$, set all bits to 1



- Query $Q(i)$?

if $\forall j, V_j(h_j(i)) = 1 \rightarrow$ "Yes"
else no!
(very likely yes)

- Significantly decrease probability of false positives

©Sham Kakade 2017

7

Analysis of Bloom Filter

- Want to keep track of n elements with false positive probability of $\delta > 0$... how large m & p ?

- Simple analysis yields:

$$m = \frac{n \log_2 \frac{1}{\delta}}{\ln 2} \approx 1.5n \log_2 \frac{1}{\delta}$$

$$p = \log_2 \frac{1}{\delta}$$

$$m = O(n)$$

$$p = O(\log \frac{1}{\delta})$$

$$\Rightarrow \text{Pr}(\text{false positive}) \leq \delta$$

©Sham Kakade 2017

8

Case Study 1: Estimating Click Probabilities

sketching
counts

Tackling an Unknown Number of Features with Sketching

Machine Learning for Big Data
CSE547/STAT548, University of Washington

Sham Kakade

April 6, 2017

©Sham Kakade 2017

9

Sketching Counts

- Bloom Filter is super cool, but not what we need...
 - We don't just care about whether a feature existed before, but to keep track of counts of occurrences of features! (assuming x_i integer)

- Recall the LR update:

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta_t \left\{ -\lambda w_i^{(t)} + x_i^{(t)} [y^{(t)} - P(Y = 1 | \mathbf{x}^{(t)}, \mathbf{w}^{(t)})] \right\}$$

- Must keep track of (weighted) counts of each feature:
 - E.g., with sparse data, for each non-zero dimension i in $\mathbf{x}^{(t)}$:

without reg., the update
is sparse
(only touch w_i if $x_i \neq 0$).

- Can we generalize the Bloom Filter?

©Sham Kakade 2017

10

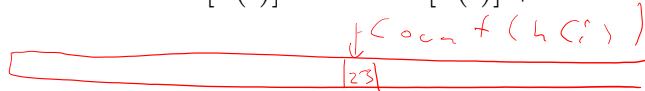
Count-Min Sketch: single vector

$$a_i = \# \text{ times seen } i$$

- Simpler problem: Count how many times you see each string
- Single hash function:
 - Keep *Count* vector of length m
 - every time see string i :

$$Q(i) = \text{count}(h(i)) \geq a_i$$

$$\text{Count}[h(i)] \leftarrow \text{Count}[h(i)] + 1$$



- Again, collisions could be a problem:
 - a_i is the count of element i :

$$\text{count}(j) = \sum_{i: h(i)=j} a_i$$

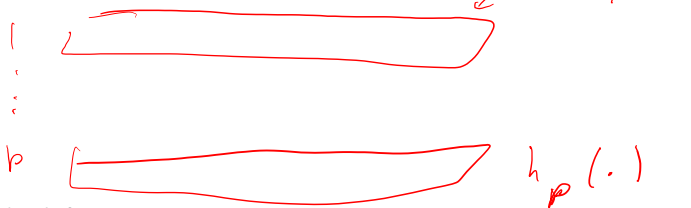
©Sham Kakade 2017

11

Count-Min Sketch: general case

p hash tables

- Keep p by m Count matrix



- p hash functions:
 - Just like in Bloom Filter, decrease errors with multiple hashes
 - Every time see string i :

$$\forall j \in \{1, \dots, p\} : \text{Count}[j, h_j(i)] \leftarrow \text{Count}[j, h_j(i)] + 1$$



©Sham Kakade 2017

12

Querying the Count-Min Sketch

$$\forall j \in \{1, \dots, p\} : \text{Count}[j, h_j(i)] \leftarrow \text{Count}[j, h_j(i)] + 1$$

- Query $Q(i)$?

- What is in $\text{Count}[j, k]$?

- Thus: $\sum_{i: h_j(i)=k} a_i \geq a_i$

- Return: $Q(i) = \min_j \text{count}(j, h_j(i)) \geq a_i$

©Sham Kakade 2017

13

Analysis of Count-Min Sketch

$$\hat{a}_i = \min_j \text{Count}[j, h(i)] \geq a_i$$

- Set:

$$m = \left\lceil \frac{e}{\epsilon} \right\rceil \quad p = \left\lceil \ln \frac{1}{\delta} \right\rceil$$

let
h be
the
maximal
of
counts

- Then, after seeing n elements:

$$\hat{a}_i \leq a_i + \epsilon n$$

- With probability at least $1-\delta$

©Sham Kakade 2017

14

Proof of Count-Min for Point Query with Positive Counts: Part 1 – Expected Bound

- $I_{i,j,k}$ = indicator that i & k collide on hash j :

$$\text{assume } \sum a_i \leq n$$

- Bounding expected value:

$$E[I_{i,j,k}] = \Pr(h_j(i) = h_j(k)) = \frac{1}{m}$$

- $X_{i,j}$ = total colliding mass on estimate of count of i in hash j :

$$X_{i,j} = \sum_{k \neq i} I_{i,j,k} a_k$$

- Bounding colliding mass:

$$E[X_{i,j}] \leq \frac{n}{m} = O(n \epsilon)$$

- Thus, estimate from each hash function is close in expectation

©Sham Kakade 2017

15

Proof of Count-Min for Point Query with Positive Counts: Part 2 – High Probability Bounds

- What we know: $\text{Count}[j, h_j(i)] = a_i + X_{i,j}$ $E[X_{i,j}] \leq \frac{\epsilon}{e} n$

- Markov inequality: For z_1, \dots, z_k positive iid random variables

$$P(\forall z_i : z_i > \alpha E[z_i]) < \alpha^{-k}$$

- Applying to the Count-Min sketch:

apply
markov
inequality

©Sham Kakade 2017

16

But updates may be positive or negative

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta_t \left\{ -\lambda w_i^{(t)} + x_i^{(t)} [y^{(t)} - P(Y = 1 | \mathbf{x}^{(t)}, \mathbf{w}^{(t)})] \right\}$$

- Count-Min sketch for positive & negative case
 - a_i no longer necessarily positive
- Update the same: Observe change Δ_i to element i :

$$\forall j \in \{1, \dots, p\} : \text{Count}[j, h_j(i)] \leftarrow \text{Count}[j, h_j(i)] + \Delta_i$$

- Each $\text{Count}[j, h_j(i)]$ no longer an upper bound on a_i
- How do we make a prediction?

$$Q(i) = \underset{j}{\text{median}} \left(\text{count}[j, h_j(i)] \right)$$

- Bound: $|\hat{a}_i - a_i| \leq 3\epsilon \|\mathbf{a}\|_1$
 - With probability at least $1 - \delta^{1/4}$, where $\|\mathbf{a}\|_1 = \sum_i |a_i|$

©Sham Kakade 2017

17

Finally, Sketching for LR

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta_t \left\{ -\lambda w_i^{(t)} + x_i^{(t)} [y^{(t)} - P(Y = 1 | \mathbf{x}^{(t)}, \mathbf{w}^{(t)})] \right\}$$

- Never need to know size of vocabulary!
- At every iteration, update Count-Min matrix:

- Making a prediction:

- Scales to huge problems, great practical implications...

©Sham Kakade 2017

18

Hash Kernels

- Count-Min sketch not designed for negative updates
- Biased estimates of dot products

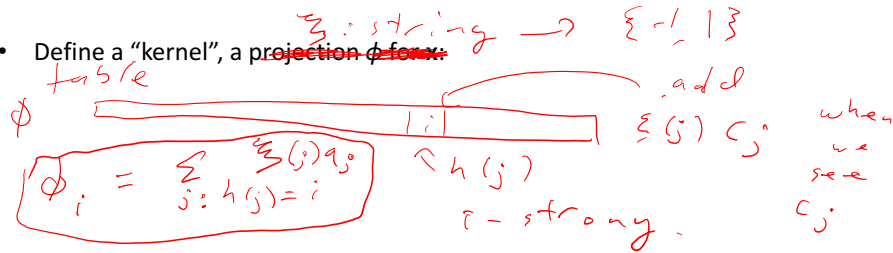
$$x \cdot x' \neq E[\phi(x) \cdot \phi(x')]$$

- Hash Kernels:** Very simple, but powerful idea to remove bias
- Pick 2 hash functions:

- h : Just like in Count-Min hashing $h: \text{string} \rightarrow \{1, \dots, m\}$

- ξ : Sign hash function
 - Removes the bias found in Count-Min hashing (see homework)

- Define a "kernel", a projection ϕ for a table



©Sham Kakade 2017

19

Hash Kernels Preserve Dot Products

$$\phi_i(\mathbf{x}) = \sum_{j: h(j)=i} \xi(j) x_j$$

- Hash kernels provide unbiased estimate of dot-products!

$$E[\phi(x) \cdot \phi(y)] = x \cdot y$$

- Variance decreases as $O(1/m)$

- Choosing m ? For $\epsilon > 0$, if

$$m = O\left(\frac{\log \frac{N}{\delta}}{\epsilon^2}\right)$$

- Under certain conditions...
- Then, with probability at least $1 - \delta$:

$$(1 - \epsilon) \|\mathbf{x} - \mathbf{x}'\|_2^2 \leq \|\phi(\mathbf{x}) - \phi(\mathbf{x}')\|_2^2 \leq (1 + \epsilon) \|\mathbf{x} - \mathbf{x}'\|_2^2$$

same ideas for random projections ($\rightarrow L$)

©Sham Kakade 2017

20

Learning With Hash Kernels

- Given hash kernel of dimension m , specified by h and ξ
 - Learn m dimensional weight vector
- Observe data point \mathbf{x}
 - Dimension does not need to be specified a priori!
- Compute $\phi(\mathbf{x})$:
 - Initialize $\phi(\mathbf{x})$
 - For non-zero entries j of \mathbf{x}_j :

$$P_r(Y=1 | \phi(\mathbf{x}), \mathbf{w}^t) = \frac{\phi(\mathbf{x}) \cdot \mathbf{w}^t}{1 + e^{\phi(\mathbf{x}) \cdot \mathbf{w}^t}}$$

- Use normal update as if observation were $\phi(\mathbf{x})$, e.g., for LR using SGD:

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta_t \left\{ -\lambda w_i^{(t)} + \phi_i(\mathbf{x}^{(t)}) [y^{(t)} - P(Y = 1 | \phi(\mathbf{x}^{(t)}), \mathbf{w}^{(t)})] \right\}$$

©Sham Kakade 2017

21

Interesting Application of Hash Kernels: Multi-Task Learning

- Personalized click estimation for many users:

- One global click prediction vector \mathbf{w} :

$$\mathbf{z}(x_u) = (\underbrace{x_1, \dots, x_d}_{\text{global}}, \dots, \underbrace{x_1, \dots, x_d}_{\text{user } u}, \dots)$$

- But...
- A click prediction vector \mathbf{w}_u per user u :

$$\mathbf{w} = (\mathbf{w}_{\text{common}}, \dots, \mathbf{w}_u, \dots)$$

- Multi-task learning: Simultaneously solve multiple learning related problems:

- Use information from one learning problem to inform the others

$$\mathbf{z}(x_u) \cdot \mathbf{w} = \mathbf{w} \cdot \mathbf{x} + \mathbf{w}_u \cdot \mathbf{x}$$

- In our simple example, learn both a global \mathbf{w} and one \mathbf{w}_u per user:

- Prediction for user u :

- If we know little about user u :

- After a lot of data from user u :

©Sham Kakade 2017

22

Problems with Simple Multi-Task Learning

- Dealing with new user is annoying, just like dealing with new words in vocabulary
- Dimensionality of joint parameter space is HUGE, e.g. personalized email spam classification from Weinberger et al.:
 - 3.2M emails
 - 40M unique tokens in vocabulary
 - 430K users
 - 16T parameters needed for personalized classification!

©Sham Kakade 2017

23

Hash Kernels for Multi-Task Learning

- Simple, pretty solution with hash kernels:
 - Very multi-task learning as (sparse) learning problem with (huge) joint data point z for point x and user u :

- Estimating click probability as desired: $\sim \sigma(\eta z)$

$$\phi(z) \cdot w$$

- Address huge dimensionality, new words, and new users using hash kernels:

©Sham Kakade 2017

24

Simple Trick for Forming Projection $\phi(\mathbf{x}, u)$

- Observe data point \mathbf{x} for user u
 - Dimension does not need to be specified a priori and user can be new!
- Compute $\phi(\mathbf{x}, u)$:
 - Initialize $\phi(\mathbf{x}, u)$
 - For non-zero entries j of \mathbf{x} :
 - E.g., $j = \text{'Obamacare'}$
 - Need two contributions to ϕ :
 - Global contribution
 - Personalized Contribution
 - Simply:
- Learn as usual using $\phi(\mathbf{x}, u)$ instead of $\phi(\mathbf{x})$ in update function

©Sham Kakade 2017

25

Results from Weinberger et al. on Spam Classification: Effect of m

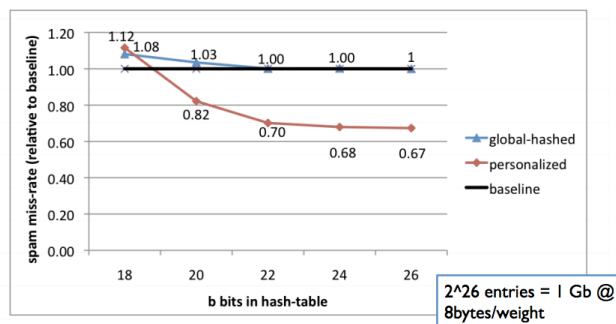


Figure 2. The decrease of uncaught spam over the baseline classifier averaged over all users. The classification threshold was chosen to keep the not-spam misclassification fixed at 1%. The hashed global classifier (*global-hashed*) converges relatively soon, showing that the distortion error ϵ_d vanishes. The personalized classifier results in an average improvement of up to 30%.

©Sham Kakade 2017

26

Results from Weinberger et al. on Spam Classification: Multi-Task Effect

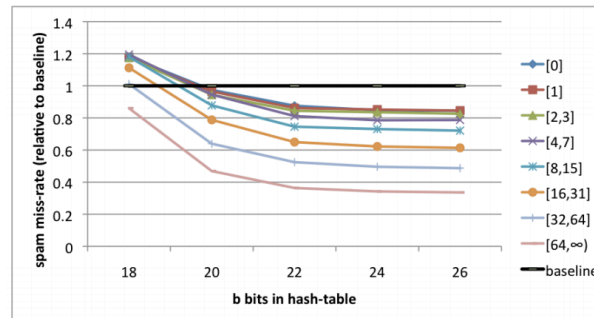


Figure 3. Results for users clustered by training emails. For example, the bucket [8, 15] consists of all users with eight to fifteen training emails. Although users in buckets with large amounts of training data do benefit more from the personalized classifier (up to 65% reduction in spam), even users that did not contribute to the training corpus at all obtain almost 20% spam-reduction.

©Sham Kakade 2017

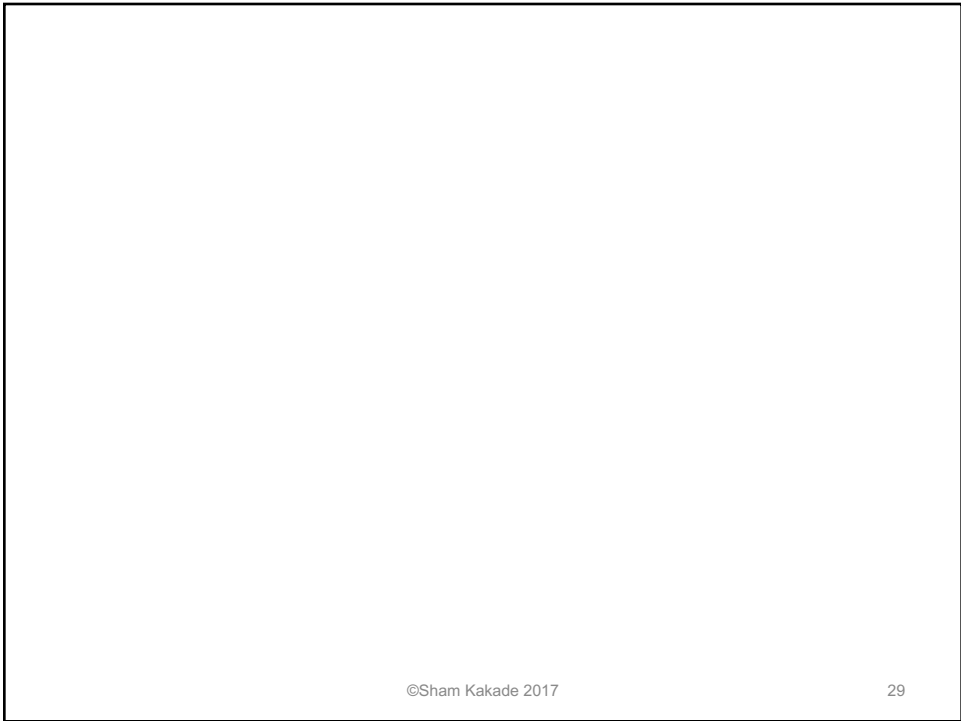
27

What you need to know

- Hash functions
- Bloom filter
 - Test membership with some false positives, but very small number of bits per element
- Count-Min sketch
 - Positive counts: upper bound with nice rates of convergence
 - General case
- Application to logistic regression
- Hash kernels:
 - Sparse representation for feature vectors
 - Very simple, use two hash function (Can use one hash function...take least significant bit to define ξ)
 - Quickly generate projection $\varphi(\mathbf{x})$
 - Learn in projected space
- Multi-task learning:
 - Solve many related learning problems simultaneously
 - Very easy to implement with hash kernels
 - Significantly improve accuracy in some problems (if there is enough data from individual users)

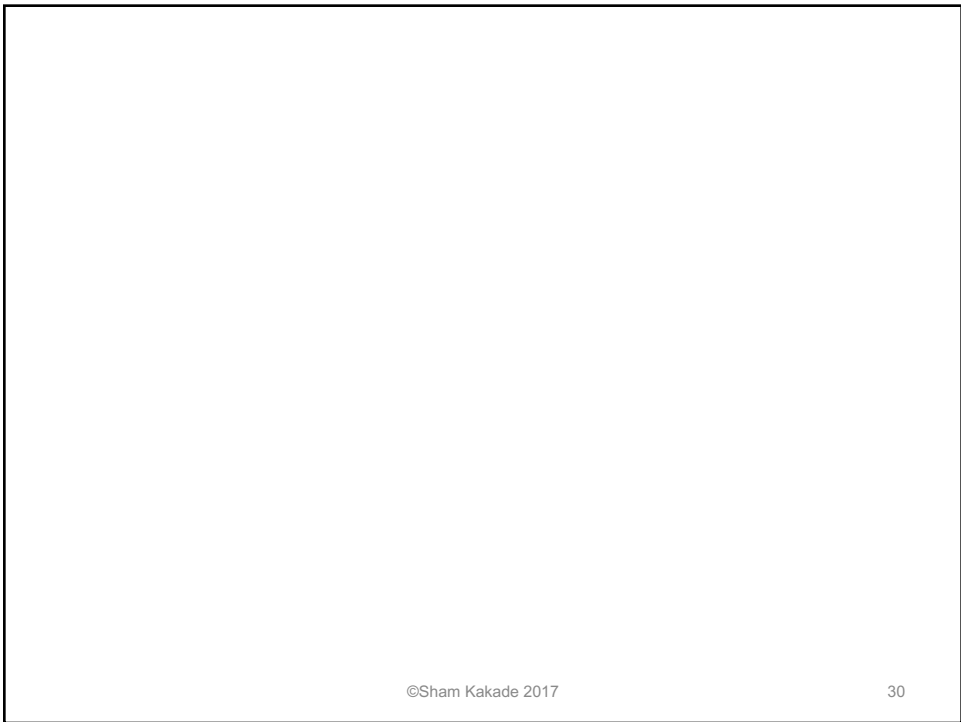
©Sham Kakade 2017

28



©Sham Kakade 2017

29



©Sham Kakade 2017

30