Importance Sampling via Locality Sensitive Hashing.

Rice University

Anshumali Shrivastava

anshumali@rice.edu

7^{th} March 2019

Motivating Problem: Stochastic Gradient Descent

$$\theta^* = \arg\min_\theta F(\theta) = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} f(x_i, \theta) \tag{1}$$

Standard GD

$$\theta_t = \theta_{t-1} - \eta^t \frac{1}{N} \sum_{i=1}^{N} \nabla f(x_j, \theta_{t-1}) \tag{2}$$

SGD, pick a random x_i, and

$$\theta_t = \theta_{t-1} - \eta^t \nabla f(x_j, \theta_{t-1}) \tag{3}$$

SGD Preferred over GD in Large-Scale Optimization.

- Slow Convergence per epoch.
- Faster Epoch, O(N) times and hence overall faster convergence.

Better SGD?

Why SGD Works? (It is Unbiased Estimator)

$$\mathbb{E}(\nabla f(x_j, \theta_{t-1})) = \frac{1}{N}\sum_{i=1}^{N} \nabla f(x_i, \theta_{t-1}). \quad (4)$$

Are there better estimators? YES!!

- Pick x_i, with probability proportional to w_i
- Optimal Variance (Alain et. al. 2015): $w_i = ||\nabla f(x_i, \theta_{t-1})||_2$
- Many works on other Importance Weights (e.g. works by Rachel Ward)

The Chicken-and-Egg Loop

- Maintaining w_i, requires $O(N)$ work.
- For Least Squares, $w_i = ||\nabla f(x_i, \theta_t)||_2 = \left|2(\theta_t \cdot x_i - y_i)||x_i||_2\right|$, changes in every iteration.

Better SGD?

Why SGD Works? (It is Unbiased Estimator)

$$\mathbb{E}(\nabla f(x_j, \theta_{t-1})) = \frac{1}{N}\sum_{i=1}^{N} \nabla f(x_i, \theta_{t-1}). \quad (4)$$

Are there better estimators? YES!!

- Pick x_i, with probability proportional to w_i
- Optimal Variance (Alain et. al. 2015): $w_i = ||\nabla f(x_i, \theta_{t-1})||_2$
- Many works on other Importance Weights (e.g. works by Rachel Ward)

The Chicken-and-Egg Loop

- Maintaining w_i, requires $O(N)$ work.
- For Least Squares, $w_i = ||\nabla f(x_i, \theta_t)||_2 = \left|2(\theta_t \cdot x_i - y_i)||x_i||_2\right|$, changes in every iteration.

Can we Break this Chicken-and-Egg Loop? Can we get adaptive sampling in constant time O(1) per Iterations, similar to cost of

Detour: Probabilistic Hashing

Probabilistic Fingerprinting (Hashing)

Hashing: Function **(Randomized)** h that maps a given data object (say $x \in \mathbb{R}^D$) to an integer key $h : \mathbb{R}^D \mapsto \{0, 1, 2, ..., N\}$. $h(x)$ serves as a discrete fingerprint.

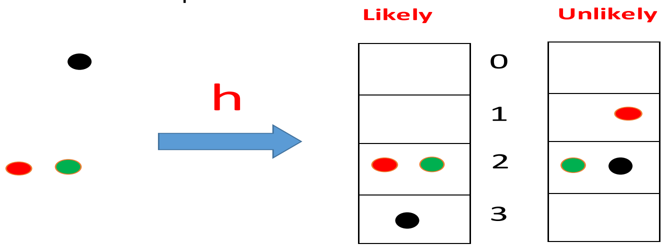
Probabilistic Fingerprinting (Hashing)

Hashing: Function **(Randomized)** h that maps a given data object (say $x \in \mathbb{R}^D$) to an integer key $h : \mathbb{R}^D \mapsto \{0, 1, 2, ..., N\}$. $h(x)$ serves as a discrete fingerprint.

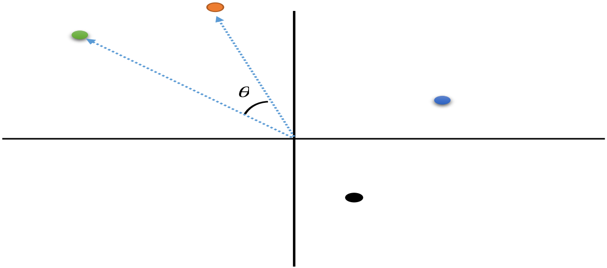
Locality Sensitive Property:

- if ~~$x = y$~~ Sim(x,y) is high then ~~$h(x) = h(y)$~~ $Pr(h(x) = h(y))$ is high.
- if ~~$x \neq y$~~ Sim(x,y) is low then ~~$h(x) \neq h(y)$~~ $Pr(h(x) = h(y))$ is low.

Similar points are more likely to have the same hash value (hash collision) compared to dissimilar points.



Popular Hashing Scheme 1: SimHash (SRP)

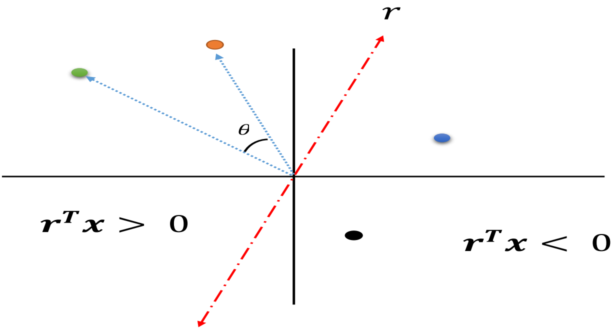


$$h_r(x) = \begin{cases} 1 & \text{if } r^T x \geq 0 \\ 0 & \text{otherwise} \end{cases} \qquad r \in \mathbb{R}^D \sim N(0, \mathcal{I})$$

$$Pr_r(h_r(x) = h_r(y)) = 1 - \frac{1}{\pi}\cos^{-1}(\theta),$$ monotonic in θ **(Cosine Similarity)**

A classical result from Goemans-Williamson (95)

Popular Hashing Scheme 1: SimHash (SRP)



$$h_r(x) = \begin{cases} 1 & \text{if } r^T x \geq 0 \\ 0 & \text{otherwise} \end{cases} \qquad r \in \mathbb{R}^D \sim N(0, \mathcal{I})$$

$$Pr_r(h_r(x) = h_r(y)) = 1 - \frac{1}{\pi}\cos^{-1}(\theta),$$ monotonic in θ **(Cosine Similarity)**

A classical result from Goemans-Williamson (95)

Some Popular Measures that are Hashable

Many Popular Measures.

- Jaccard Similarity (MinHash)
- Cosine Similarity (Simhash and also MinHash if Data is Binary)
- Euclidian Distance
- Earth Mover Distance, etc.

Recently, Un-normalized Inner Products[1]

1. With bounded norm assumption.
2. Allowing Asymmetry.

[1] SL [NIPS 14 (Best Paper), UAI 15, WWW 15], APRS [PODS 16].

Sub-linear Near-Neighbor Search

Given a query $q \in \mathbb{R}^D$ and a giant collection $\mathcal{C}$ of N vectors in $\mathbb{R}^D$, search for $p \in \mathcal{C}$ s.t.,

$$p = \arg\max_{x \in \mathcal{C}} \quad sim(q, x)$$

- *sim* is the similarity, like Cosine Similarity, Resemblance, etc.
- Worst case $O(N)$ for any query. N is huge.
- Querying is a very frequent operation.

Sub-linear Near-Neighbor Search

Given a query $q \in \mathbb{R}^D$ and a giant collection $\mathcal{C}$ of N vectors in $\mathbb{R}^D$, search for $p \in \mathcal{C}$ s.t.,

$$p = \arg\max_{x \in \mathcal{C}} \quad sim(q, x)$$

- *sim* is the similarity, like Cosine Similarity, Resemblance, etc.
- Worst case $O(N)$ for any query. N is huge.
- Querying is a very frequent operation.

Our goal is to find sub-linear query time algorithm.

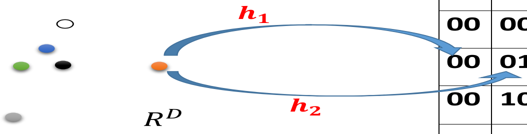
1. Approximate (or Inexact) answer suffices.
2. We are allowed to pre-process $\mathcal{C}$ once. (offline costly step)

Probabilities Hash Tables

Given: $Pr_h\left[h(x) = h(y)\right] = f(sim(x, y))$, f is monotonic.

Probabilities Hash Tables

Given: $Pr_h\left[h(x) = h(y)\right] = f(sim(x, y))$, f is monotonic.

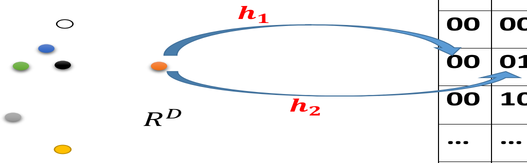


$h_1, h_2 : R^D \rightarrow \{0, 1, 2, 3\}$

h_1	h_2	Buckets (pointers only)
00	00	
00	01	●
00	10	
...	...	
11	11	

Probabilities Hash Tables

Given: $Pr_h\left[h(x) = h(y)\right] = f(sim(x, y))$, f is monotonic.

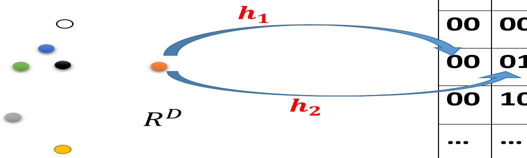


$h_1, h_2 : R^D \rightarrow \{0, 1, 2, 3\}$

h_1	h_2	Buckets (pointers only)
00	00	...
00	01	...
00	10	Empty
...	...	...
11	11	...

Probabilities Hash Tables

Given: $Pr_h\left[h(x) = h(y)\right] = f(sim(x, y))$, f is monotonic.



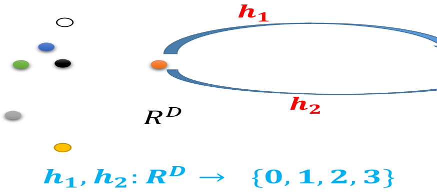
$h_1, h_2 : R^D \rightarrow \{0, 1, 2, 3\}$

h_1	h_2	Buckets (pointers only)
00	00	...
00	01	...
00	10	Empty
...	...	...
11	11	...

- Given query q, if $h_1(q) = 11$ **and** $h_2(q) = 01$, then probe bucket with index **1101**. It is a good bucket !!

Probabilities Hash Tables

Given: $Pr_h\left[h(x) = h(y)\right] = f(sim(x, y))$, f is monotonic.



h_1	h_2	Buckets (pointers only)
00	00	…
00	01	…
00	10	Empty
…	…	…
11	11	…

- Given query q, if $h_1(q) = 11$ **and** $h_2(q) = 01$, then probe bucket with index **1101**. It is a good bucket !!
- (Locality Sensitive) $h_i(q) = h_i(x)$ noisy indicator of high similarity.
- Doing better than random !!

The Classical LSH Algorithm

Table 1

h_1^1	…	h_K^1	**Buckets**
00	…	00	…
00	…	01	…
00	…	10	**Empty**
…	…	…	…
11	…	11	…

- We use K concatenation.

The Classical LSH Algorithm

Table 1

h_1^1	...	h_K^1	Buckets
00	...	00	● ● ...
00	...	01	● ○ ...
00	...	10	Empty
...	...	...	...
11	...	11	...

...

Table L

h_1^L	...	h_K^L	Buckets
00	...	00	● ● ...
00	...	01	● ● ...
00	...	10	○ ● ●
...	...	...	...
11	...	11	Empty

- We use K concatenation.
- Repeat the process L times. (L Independent Hash Tables)

The Classical LSH Algorithm

Table 1

h_1^1	...	h_K^1	Buckets
00	...	00	...
00	...	01	...
00	...	10	Empty
...	...	...	...
11	...	11	...

...

Table L

h_1^L	...	h_K^L	Buckets
00	...	00	...
00	...	01	...
00	...	10	
...	...	...	...
11	...	11	Empty

- We use K concatenation.
- Repeat the process L times. (L Independent Hash Tables)
- **Querying :** Probe one bucket from each of L tables. Report union.

The Classical LSH Algorithm

Table 1

h_1^1	...	h_K^1	Buckets
00	...	00	...
00	...	01	...
00	...	10	Empty
...	...	...	...
11	...	11	...

...

Table L

h_1^L	...	h_K^L	Buckets
00	...	00	...
00	...	01	...
00	...	10	
...	...	...	...
11	...	11	Empty

- We use K concatenation.
- Repeat the process L times. (L Independent Hash Tables)
- **Querying :** Probe one bucket from each of L tables. Report union.

1. Two knobs K and L to control.

Success of LSH

Similarity Search or Related (Reduce n)

- Similarity Search or related.
- Plenty of Applications.

[2] Li et. al. NIPS 2011

Success of LSH

Similarity Search or Related (Reduce n)

- Similarity Search or related.
- Plenty of Applications.

Similarity Estimation and Embedding (Reduce dimensionality d)

- Basically JL (Johnson-Lindenstrauss) or Random Projections does most of the job!!
- Similarity Estimation. (Usually not optimal in Fisher Information Sense)
- Non-Linear SVMs in Learning Linear Time [2].

Result: Won 2012 ACM Paris Kanellakis Theory and Practice Award.

[2] Li et. al. NIPS 2011

Success of LSH

Similarity Search or Related (Reduce n)

- Similarity Search or related.
- Plenty of Applications.

Similarity Estimation and Embedding (Reduce dimensionality d)

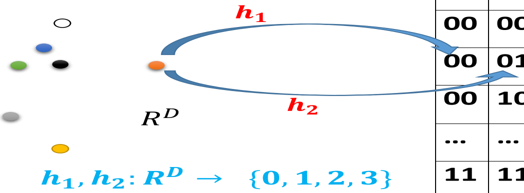
- Basically JL (Johnson-Lindenstrauss) or Random Projections does most of the job!!
- Similarity Estimation. (Usually not optimal in Fisher Information Sense)
- Non-Linear SVMs in Learning Linear Time [2].

Result: Won 2012 ACM Paris Kanellakis Theory and Practice Award.

Are there other Fundamental Problems?

[2] Li et. al. NIPS 2011

A Step Back



h_1	h_2	Buckets (pointers only)
00	00	...
00	01	...
00	10	Empty
...	...	...
11	11	...

Is LSH really a search algorithm?

- Given the query x, LSH samples θ_y from the dataset, with probability exactly $p_y = 1 - (1 - p(x, \theta_y)^K)^L$.
- LSH is considered a black box for near-neighbor search. It is not!!
- Adaptive Sampling is being converted into an algorithm for high similarity search.

New View: Hashing is an Efficient Adaptive Sampling in Disguise.

Partition Function in Log-Linear Models

$$P(y|x,\theta) = \frac{e^{\theta_y \cdot x}}{Z_\theta}$$

- θ_y is the weight vector
- x is the (current context) feature vector (word2vec).
- $Z_\theta = \sum_{y \in Y} e^{\theta_y \cdot x}$ is the partition function

Issues:

- Z_θ is expensive. $|Y|$ is huge. (billion word2vec)
- Change in context x requires to recompute Z_θ.

Question: Can we reduce the amortized cost of estimating Z_θ?

Importance Sampling (IS)

Summation by expectation: But sampling $y_i \propto e^{\theta_y \cdot x}$ is equally harder.

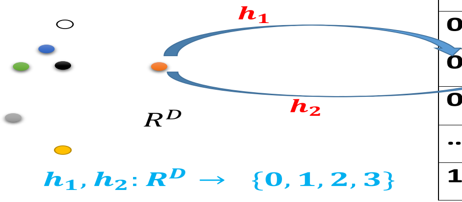
Importance Sampling

- Given a normalized proposal distribution $g(y)$ where $\sum_y g(y) = 1$.
- We have an unbiased estimator $\mathbb{E}\left[\frac{f(y)}{g(y)}\right] = \sum_y g(y)\frac{f(y)}{g(y)} = \sum_y f(y) = Z_\theta$
- Draw N samples $y_i \sim g(y)$ for $i = 1 \dots N$. we can estimate $Z_\theta = \frac{1}{N} sum_{i=1}^{N} \frac{f(y_i)}{g(y_i)}$.

Yet Another Chicken and Egg Loop:

- Does not really work if $g(y)$ is not close to $f(y)$.
- Getting $g(y)$ which is efficient and close to $f(y)$ is not known.
- No efficient choice in literature. Random sampling or other heuristics.

Detour: LSH as Samplers

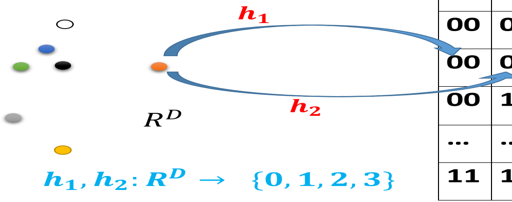


h_1	h_2	Buckets (pointers only)
00	00	● ● ● ...
00	01	● ● ○ ...
00	10	Empty
...	...	...
11	11	● ...

(K, L) parameterized LSH algorithm is an efficient sampling:

- Given the query x, LSH samples θ_y from the dataset, with probability exactly $p_y = 1 - (1 - p(x, \theta_y)^K)^L$.
- LSH is considered a black box for near-neighbor search. It is not.

Detour: LSH as Samplers



h_1	h_2	Buckets (pointers only)
00	00	● ● ● ...
00	01	● ● ○ ...
00	10	Empty
...	...	...
11	11	● ...

(K, L) **parameterized LSH algorithm is an efficient sampling:**

- Given the query x, LSH samples θ_y from the dataset, with probability exactly $p_y = 1 - (1 - p(x, \theta_y)^K)^L$.
- LSH is considered a black box for near-neighbor search. It is not.

Unnormalized Importance Sampling:

- It is not normalized $\sum_y p_y \neq 1$
- Samples are correlated.

It turns out, we can still make them work!

Beyond IS: The Unbiased LSH Based Estimator

Procedure:

- For context x, report all the retrieved y_is from the (K, L) parameterized LSH Algorithm. (just one NN query)
- Report $\hat{Z}_\theta = \sum_i \frac{e^{\theta_{y_i} \cdot x}}{1-(1-p(x,\theta_{y_i})^K)^L}$

Properties:

- $E[\hat{Z}_\theta] = Z_\theta$ (Unbiased)
- $$Var[\hat{Z}_\theta] = \sum_i \frac{f(y_i)^2}{p_i} - \sum_{i=1}^{N} f(y_i)^2 + \sum_{i \neq j} \frac{f(y_i)f(y_j)}{p_i p_j} \mathrm{Cov}(\mathbf{1}_{[y_i \in S]} \cdot \mathbf{1}_{[y_j \in S]})$$
- Correlations are mostly negative (favorable) with LSH.

MIPS Hashing is Ideal for Log-Linear Models

Theorem

For any two states y_1 and y_2:

$$P(y_1|x;\theta) \geq P(y_2|x;\theta) \iff p_1 \geq p_2$$

where

$$p_i = 1 - (1 - p(\theta_{y_i} \cdot x)^K)^L$$

$$P(y|x,\theta) \propto e^{\theta_y \cdot x}$$

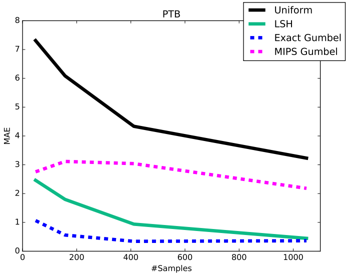
Corollary

The modes of both the sample and the target distributions are identical.

.

Efficient as well as similar to target (Adaptive).

How does it works? (PTB and Text8 Datasets)



Running Time:

Samples	Uniform	LSH	Exact Gumbel	MIPS Gumbel
50	0.13	0.23	531.37	260.75
400	0.92	1.66	3,962.25	1,946.22
1500	3.41	6.14	1,4686.73	7,253.44
5000	9.69	17.40	42,034.58	20,668.61

Final Perplexity of Language Models

Standard	LSH	Uniform	Exact Gumbel	MIPS Gumbel
91.8	98.8	524.3	91.9	Diverged
140.7	162.7	1347.5	152.9	

Back to Adaptive SGD

Why SGD Works? (It is Unbiased Estimator)

$$\mathbb{E}(\nabla f(x_j, \theta_{t-1})) = \frac{1}{N}\sum_{i=1}^{N} \nabla f(x_i, \theta_{t-1}). \quad (5)$$

Are there better estimators? YES!!

- Pick x_i, with probability proportional to w_i
- Optimal Variance (Alain et. al. 2015): $w_i = ||\nabla f(x_i, \theta_{t-1})||_2$
- Many works on Other Importance Weights

Optimal Variance w_i

- $w_i = ||\nabla f(x_i, \theta_{t-1})||_2 = 2\Big|\langle \theta_t, -1 \rangle \cdot \langle x_i||x_i||, y_i||x_i|| \rangle\Big|$
- Large Inner Product, θ_t changes, x_i's remains fixed :)
- We wont sample exactly in proportion to w_i, but with some w_i', which is monotonic in w_i.

The Complete Picture

One time Cost

- Preprocess $< x_i||x_i||, y_i||x_i|| >$ into Inner Product Hash Tables. (Data Reading Cost)

Per Iteration

- Query hash tables with $< \theta_{t-1}, -1 >$ for sample x_i. (1-2 Hash Lookups)
- Estimate Gradient as $\frac{\nabla f(x_i, \theta_{t-1})}{N \times SamplingProbability}$
- Can show: Unbiased and better variance than SGD.

The Complete Picture

One time Cost

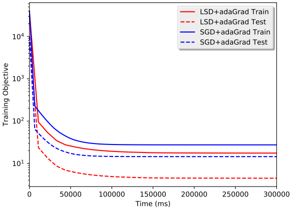
- Preprocess $< x_i||x_i||, y_i||x_i|| >$ into Inner Product Hash Tables. (Data Reading Cost)

Per Iteration

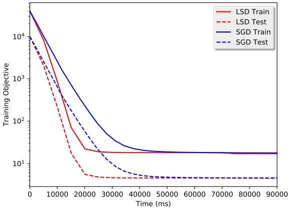
- Query hash tables with $< \theta_{t-1}, -1 >$ for sample x_i. (1-2 Hash Lookups)
- Estimate Gradient as $\frac{\nabla f(x_i, \theta_{t-1})}{N \times SamplingProbability}$
- Can show: Unbiased and better variance than SGD.

Per iterations cost is 1.5 times that of SGD, but superior variance.

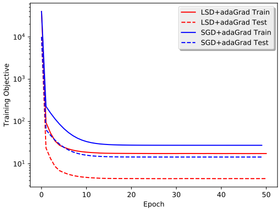
How it works?



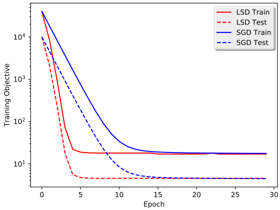
(a) Ada Time



(b) Plain Time



(c) Ada Epoch



(d) Plain Epoch

Conclusion

Hashing can change the equation!!