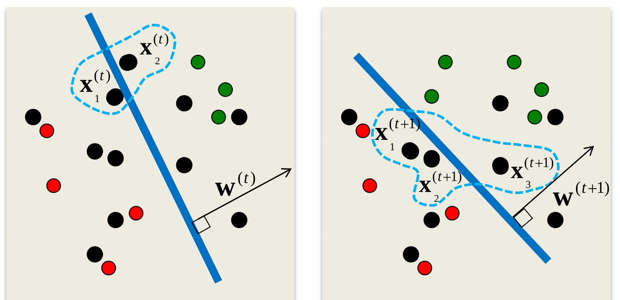
Hashing Hyperplane Queries to Near Points with Applications to Large-Scale Active Learning

Prateek Jain[1], Sudheendra Vijayanarasimhan[2], and Kristen Grauman[2]
[1]Microsoft Research Lab, Bangalore, INDIA, [2]University of Texas, Austin, TX, USA

Motivation

Goal: For large-scale active learning, want to repeatedly query annotators to label the most uncertain examples in a massive pool of unlabeled data $\mathcal{U}$.



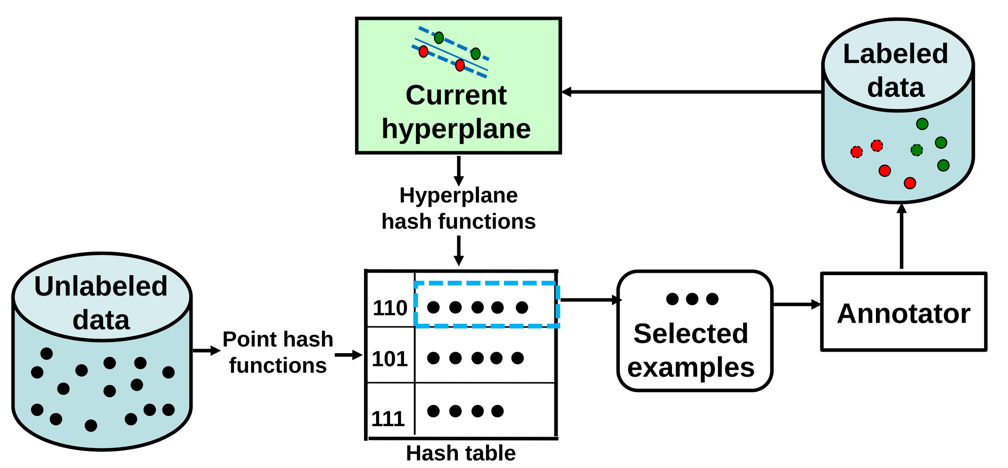
Margin-based selection criterion for SVMs [Tong & Koller, 2000] selects points nearest to current decision boundary:

$$\mathbf{x}^* = \underset{\mathbf{x}_i \in \mathcal{U}}{\operatorname{argmin}} |\mathbf{w}^T \mathbf{x}_i|$$

Problem: With massive unlabeled pool, cannot afford exhaustive linear scan.

Main Idea: Sub-linear Time Active Selection

Idea: We define two hash function families that are locality-sensitive for the *nearest neighbor to a hyperplane query* search problem. The two variants offer trade-offs in error bounds versus computational cost.



- Offline: Hash unlabeled data into table.
- Online: Hash current classifier as "query" to directly retrieve next examples for labeling.

Main contributions:

- Novel hash functions to map query hyperplane to near points in sub-linear time.
- Bounds for locality-sensitivity of hash families for perpendicular vectors.
- Large-scale pool-based active learning results for documents and images, with up to one million unlabeled points.

Background: Locality-Sensitive Hashing (LSH)

Let $d(\cdot,\cdot)$ be a distance function over items from a set S, and for any item $p \in S$, let $B(p, r)$ denote the set of examples from S within radius r from p.

Definition 1. LSH functions [Gionis, Indyk, & Motwani, 1999]

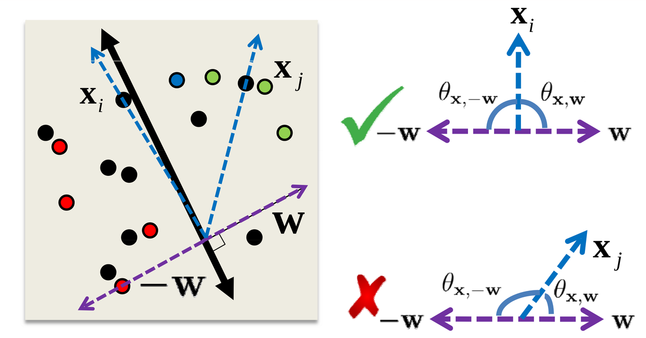
Let $h_{\mathcal{H}}$ denote a random choice of a hash function from the family $\mathcal{H}$. The family $\mathcal{H}$ is called $(r, r(1+\epsilon), p_1, p_2)$−sensitive for $d(\cdot,\cdot)$ when, for any $q, p \in S$,

- if $p \in B(q, r)$ then $\Pr[h_{\mathcal{H}}(q) = h_{\mathcal{H}}(p)] \geq p_1$,
- if $p \notin B(q, r(1+\epsilon))$ then $\Pr[h_{\mathcal{H}}(q) = h_{\mathcal{H}}(p)] \leq p_2$.

- Compute k-bit hash keys for each point p_i: $\left[h_{\mathcal{H}}^{(1)}(p_i), h_{\mathcal{H}}^{(2)}(p_i), \ldots, h_{\mathcal{H}}^{(k)}(p_i)\right]$.
- Given a query q, search over examples in the l buckets to which q hashes.
  - Use $l = N^\rho$ hash tables for N points, where $\rho = \frac{\log p_1}{\log p_2} \leq \frac{1}{1+\epsilon}$.
  - A $(1+\epsilon)$-approximate solution is retrieved in time $O(N^{\frac{1}{(1+\epsilon)}})$.

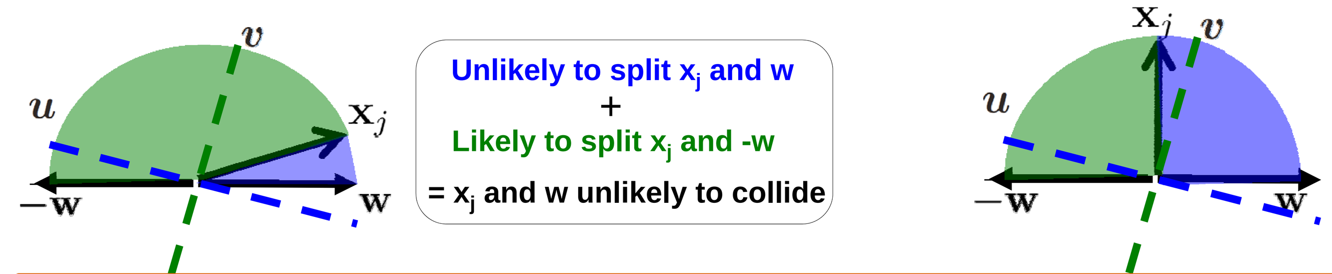
First Solution: Hyperplane Hash

Intuition: To retrieve those points for which $|\mathbf{w}^T \mathbf{x}|$ is small, we want collisions to be probable for vectors *perpendicular* to hyperplane normal (assuming normalized data).



For $\mathbf{u} \sim \mathcal{N}(0, I)$, $\Pr[\text{sign}(\mathbf{u}^T\mathbf{w}) \neq \text{sign}(\mathbf{u}^T\mathbf{x})] = \frac{1}{\pi}\theta_{\mathbf{w},\mathbf{x}}$ [Goemans & Williamson, 1995].

Our idea: Generate two independent random vectors $\mathbf{u}$ and $\mathbf{v}$: one to capture angle between $\mathbf{w}$ and $\mathbf{x}$, and one to capture angle between $-\mathbf{w}$ and $\mathbf{x}$.



Definition 2. Hyperplane Hash (H-Hash) Functions

We define H-Hash function family $\mathcal{H}$ as:

$$h_{\mathcal{H}}(\mathbf{z}) = \begin{cases} h_{\mathbf{u},\mathbf{v}}(\mathbf{z}, \mathbf{z}), & \text{if } \mathbf{z} \text{ is a database point vector,} \\ h_{\mathbf{u},\mathbf{v}}(\mathbf{z}, -\mathbf{z}), & \text{if } \mathbf{z} \text{ is a query hyperplane vector.} \end{cases}$$

where $h_{\mathbf{u},\mathbf{v}}(\mathbf{a}, \mathbf{b}) = [\text{sign}(\mathbf{u}^T\mathbf{a}), \text{sign}(\mathbf{v}^T\mathbf{b})]$, is a two-bit hash, and $\mathbf{u}, \mathbf{v} \sim \mathcal{N}(0, I)$.

- Probability of collision between $\mathbf{w}$ and $\mathbf{x}$ is given by

$$\Pr[h_{\mathcal{H}}(\mathbf{w}) = h_{\mathcal{H}}(\mathbf{x})] = \frac{\theta_{\mathbf{x},\mathbf{w}}}{\pi}\left(1 - \frac{\theta_{\mathbf{x},\mathbf{w}}}{\pi}\right) = \frac{1}{4} - \frac{1}{\pi^2}\left(\theta_{\mathbf{x},\mathbf{w}} - \frac{\pi}{2}\right)^2$$

and we have

$$p_1 = \frac{1}{4} - \frac{r}{\pi^2}, \quad p_2 = \frac{1}{4} - \frac{r(1+\epsilon)}{\pi^2}$$

- Hence, can return a point for which $(\theta_{\mathbf{x},\mathbf{w}} - \frac{\pi}{2})^2 \leq r$ in **sub-linear time** $O(N^\rho)$.

$$\rho = \frac{1 - \log(1 - \frac{4r}{\pi^2})}{1 + \frac{\epsilon}{1 + \frac{\pi^2}{4r}\log 4}} < 1$$

Second Solution: Embedded Hyperplane Hash

Intuition: Design Euclidean embedding after which minimizing distance is equivalent to minimizing $|\mathbf{w}^T\mathbf{x}|$, making existing approx. NN methods applicable.

Definition 3. Embedded Hyperplane Hash (EH-Hash) Functions

We define EH-Hash function family $\mathcal{E}$ as:

$$h_{\mathcal{E}}(\mathbf{z}) = \begin{cases} h_{\mathbf{u}}(V(\mathbf{z})), & \text{if } \mathbf{z} \text{ is a database point vector,} \\ h_{\mathbf{u}}(-V(\mathbf{z})), & \text{if } \mathbf{z} \text{ is a query hyperplane vector,} \end{cases}$$

where $V(\mathbf{a}) = vec(\mathbf{a}\mathbf{a}^T) = [a_1^2, a_1a_2, \ldots, a_1a_d, a_2^2, a_2a_3, \ldots, a_d^2]$ gives the embedding, and $h_{\mathbf{u}}(\mathbf{b}) = \text{sign}(\mathbf{u}^T\mathbf{b})$, with $\mathbf{u} \in \Re^{d^2}$ sampled from $\mathcal{N}(0, I)$.

- Embedding inspired by [Basri et al., 2009]; we give LSH bounds for $(\theta_{\mathbf{x},\mathbf{w}} - \pi/2)^2$.

Embedded Hyperplane Hash (continued)

- Since $||V(\mathbf{x}) - (-V(\mathbf{w}))||^2 = 2 + 2(\mathbf{x}^T\mathbf{w})^2$, distance between embeddings of $\mathbf{x}$ and $\mathbf{w}$ proportional to desired distance, so standard LSH function $h_{\mathbf{u}}(\cdot)$ applicable.
- Probability of collision between $\mathbf{w}$ and $\mathbf{x}$ is given by

$$\Pr[h_{\mathcal{E}}(\mathbf{w}) = h_{\mathcal{E}}(\mathbf{x})] = \cos^{-1}\left(\cos^2(\theta_{\mathbf{x},\mathbf{w}})\right)/\pi$$

and we have $p_1 = \frac{1}{\pi}\cos^{-1}\sin^2(\sqrt{r})$.

- Hence, sub-linear time search with about twice the p_1 guaranteed by H-Hash.
- **Issue:** $V(\mathbf{a})$ is d^2-dimensional, higher hashing overhead.
- **Solution:** Compute $h_{\mathbf{u}}(V(\mathbf{a}))$ approximately using randomized sampling:

Lemma 4. Sampling to Approximate Inner Product

Let $\mathbf{v} \in \mathbb{R}^d$, define $p_i = v_i^2/\|\mathbf{v}\|^2$. Construct $\tilde{\mathbf{v}} \in \mathbb{R}^d$ such that the i-th element is v_i with probability p_i and is 0 otherwise. Select t such elements using sampling with replacement. Then, for any $\mathbf{y} \in \mathbb{R}^d$, $\epsilon > 0$, $c \geq 1$, $t \geq \frac{c}{\epsilon^2}$,

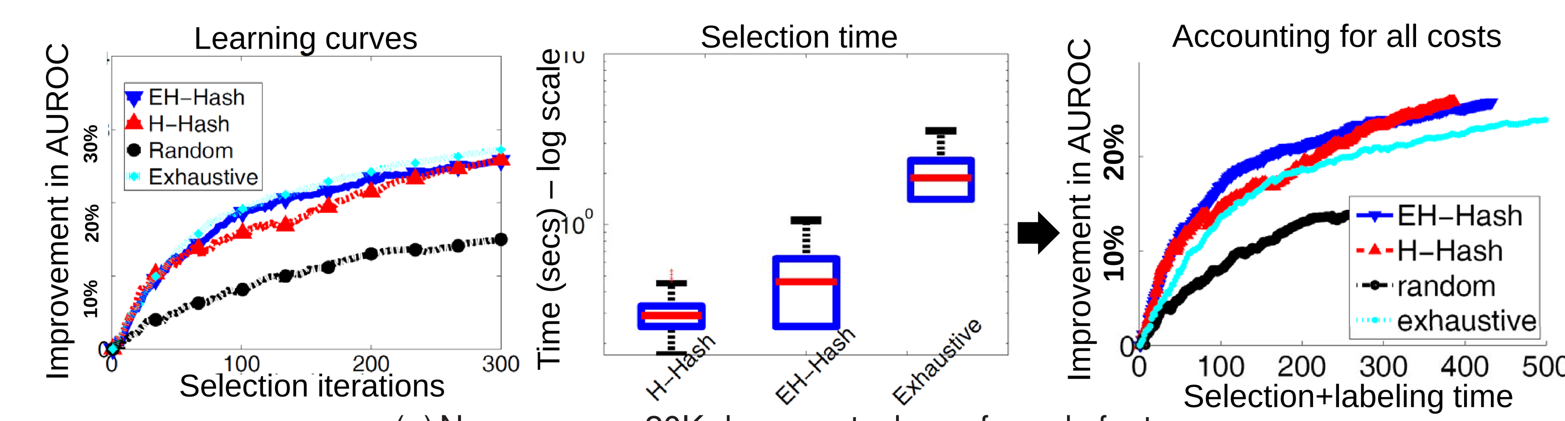
$$\Pr[|\tilde{\mathbf{v}}^T\mathbf{y} - \mathbf{v}^T\mathbf{y}| \leq \epsilon'\|\mathbf{v}\|^2\|\mathbf{y}\|^2] > 1 - 1/c$$

Trade-off: H-Hash has faster pre-processing, but EH-Hash has stronger bounds.

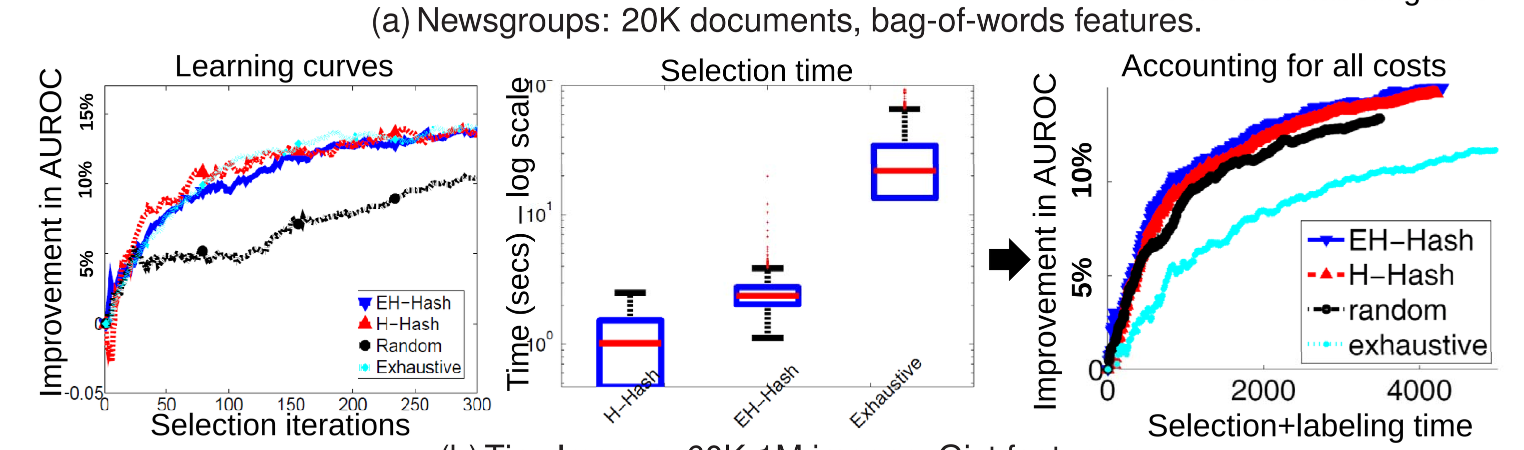
	Accuracy	Hashing insertion time
H-Hash:	$p_1 = \frac{1}{4} - \frac{r}{\pi^2}$	$\propto d$
EH-Hash:	$p_1 \geq 2\left(\frac{1}{4} - \frac{r}{\pi^2}\right)$	$\propto d^2$ (d with sampling)

Experimental Results

- **Goal:** Show that proposed algorithms can select examples nearly as well as the exhaustive approach, but with substantially greater efficiency.



(a) Newsgroups: 20K documents, bag-of-words features.



(b) Tiny Images: 60K-1M images, Gist features.

- Accounting for both selection and labeling time, our approach performs better than either random selection *or* exhaustive active selection.
- Trade-offs confirmed in practice: H-Hash faster, EH-Hash more accurate.
- In future work, we plan to explore extensions for non-linear kernels.