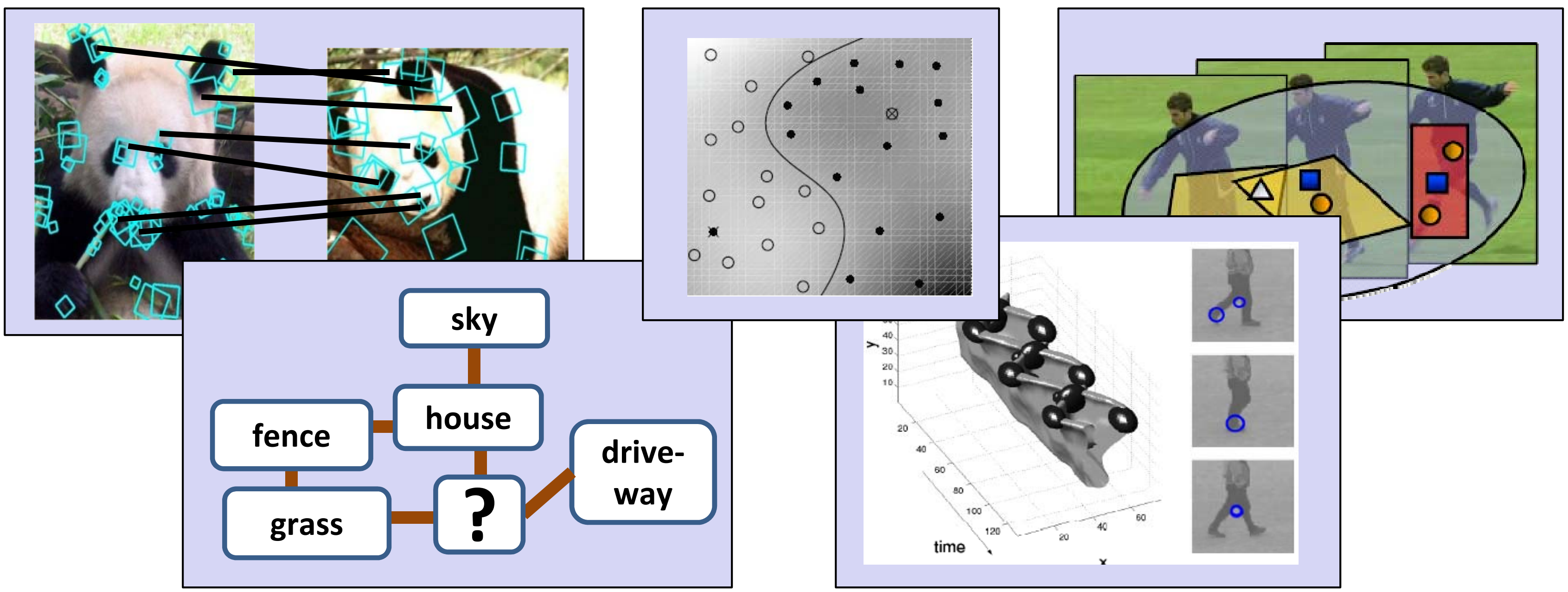
Retrieving Similar or Informative Instances on a Budget

Kristen Grauman
Dept. of Computer Science
University of Texas at Austin

Work with Sudheendra Vijayanarasimham,
Brian Kulis, and Prateek Jain

The visual search problem

- Massive search pools
  - ~20 hours of video/minute added to YouTube
  - ~5,000 new tagged photos/minute added to Flickr,…
- Complex representations and distances



Retrieval on a budget

Goal: Specify resources available →
algorithm focuses search accordingly.

Cost of search

Accuracy of retrieved results

Cost of annotation

Accuracy of learned models

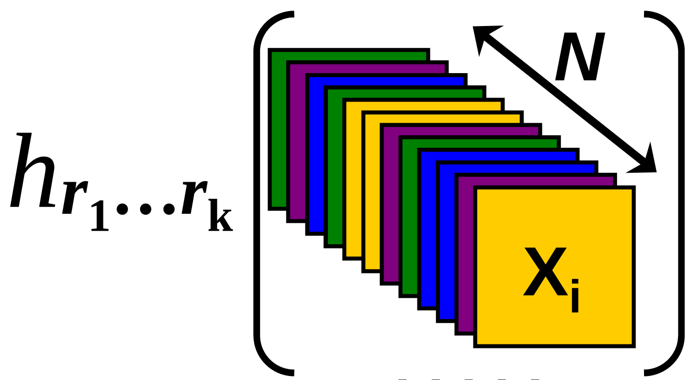
Retrieval on a budget

- Retrieving *similar* instances with a search time budget
  - Novel hash functions for learned metrics and arbitrary kernel functions

- Retrieving *informative* instances with a search time or annotation cost budget
  - Novel hash functions for hyperplane queries
  - Budgeted batch-mode active selection

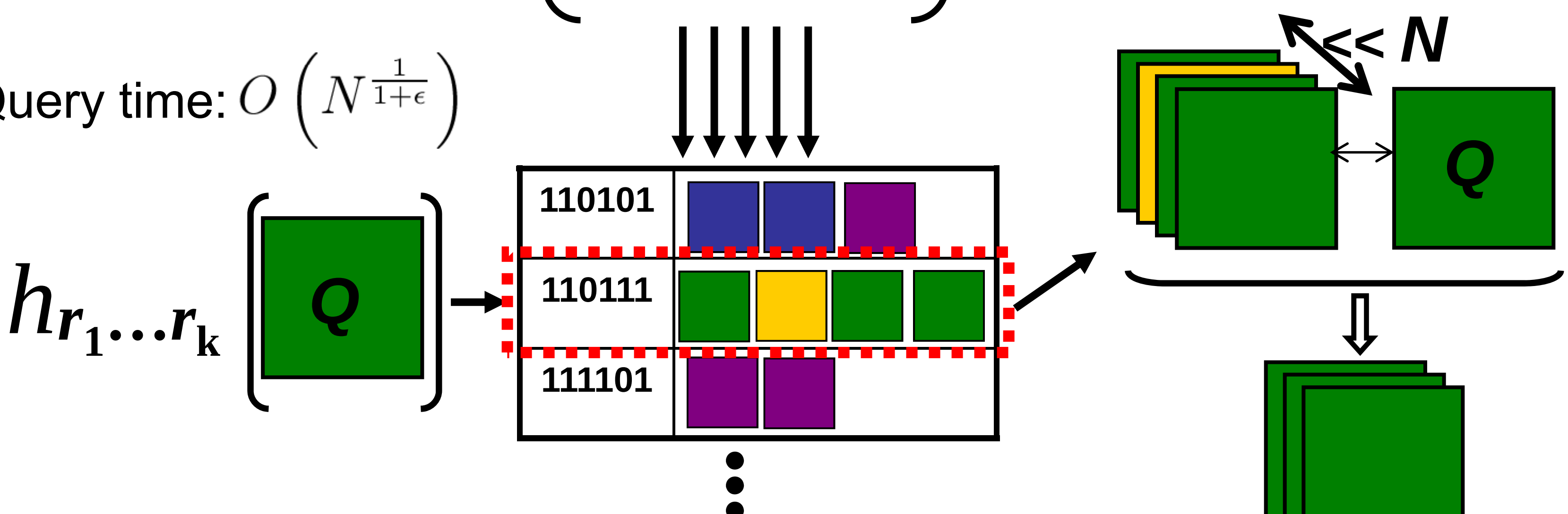
Locality Sensitive Hashing (LSH)

$$\Pr_{h \in \mathcal{F}} [h(x) = h(y)] = sim(x, y)$$



Guarantees approximate near neighbors in sub-linear time, *given appropriate hash functions*.

Query time: $O\left(N^{\frac{1}{1+\epsilon}}\right)$

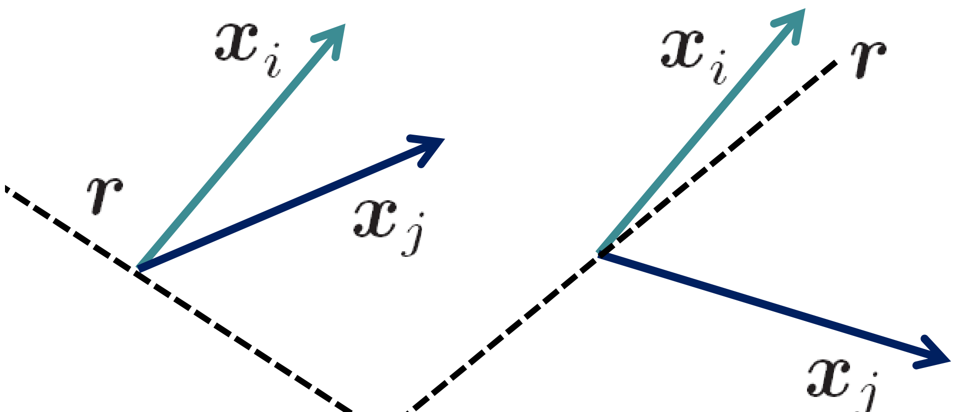


[Indyk and Motwani 1998, Charikar 2002]

LSH functions for dot products

The probability that a *random hyperplane* separates two unit vectors depends on the angle between them:

$$\Pr[\text{sign}(\boldsymbol{x}_i^T \boldsymbol{r}) = \text{sign}(\boldsymbol{x}_j^T \boldsymbol{r})] = 1 - \frac{1}{\pi}\cos^{-1}(\boldsymbol{x}_i^T \boldsymbol{x}_j)$$



High dot product: unlikely to split

Lower dot product: likely to split

Corresponding hash function:

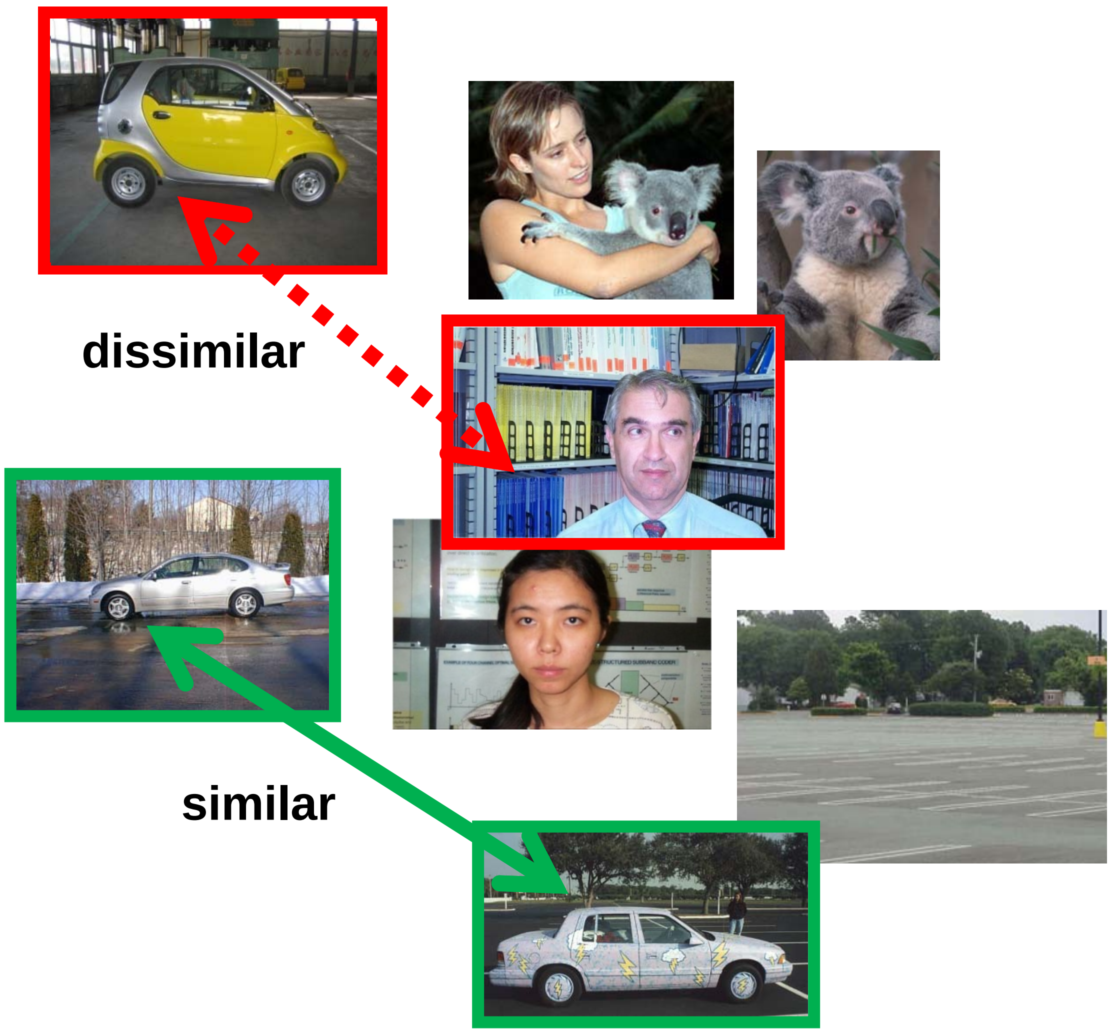
$$h_{\boldsymbol{r}}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{r}^T \boldsymbol{x} \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

$$r_i \sim \mathcal{N}(0, 1)$$

[Goemans and Williamson 1995, Charikar 2004]

Kristen Grauman, UT-Austin

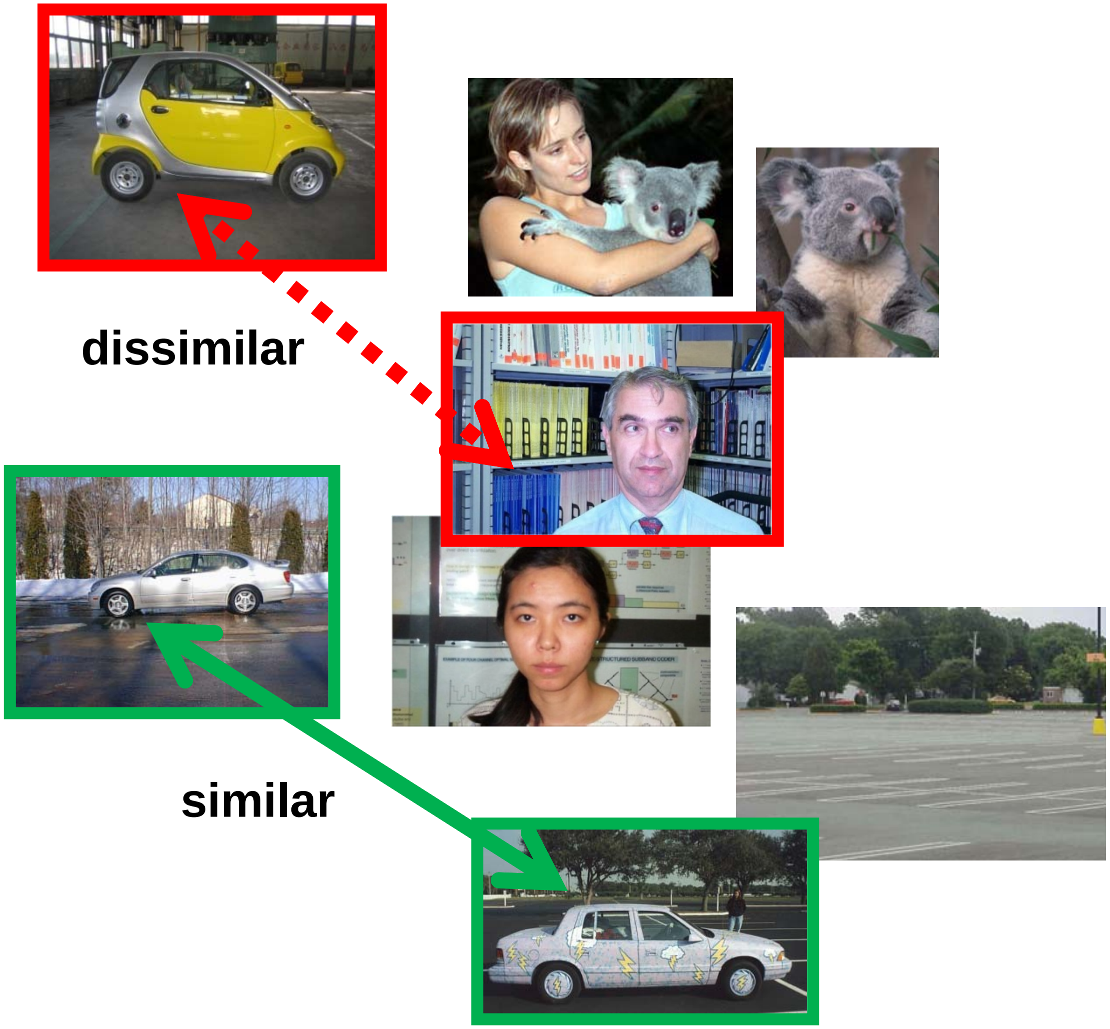
Learning how to compare images



- Exploit (dis)similarity constraints to construct more useful distance function
- Number of existing techniques for metric learning

[Weinberger et al. 2004, Hertz et al. 2004, Frome et al. 2007, Varma & Ray 2007, Kumar et al. 2007]

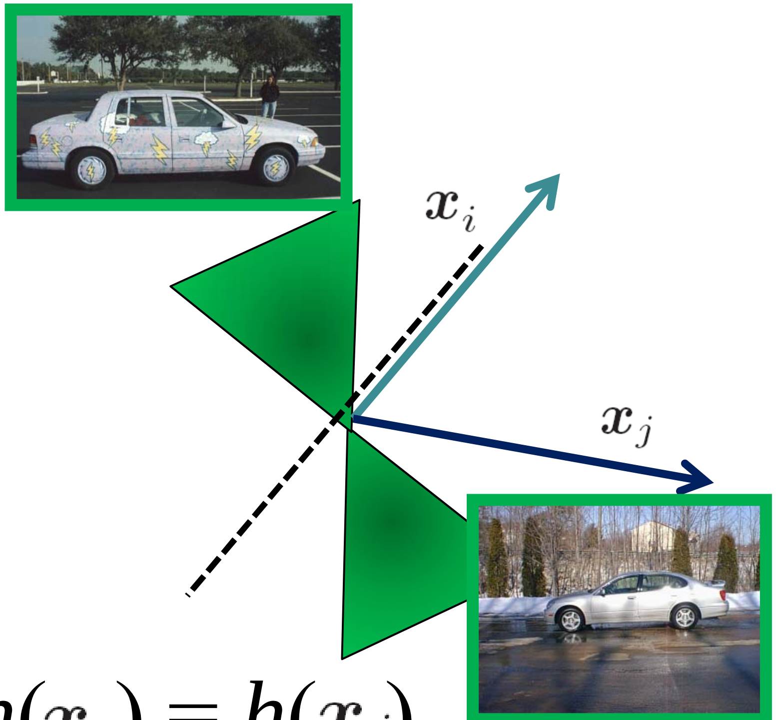
Learning how to compare images



- Exploit (dis)similarity constraints to construct more useful distance function
- Number of existing techniques for metric learning

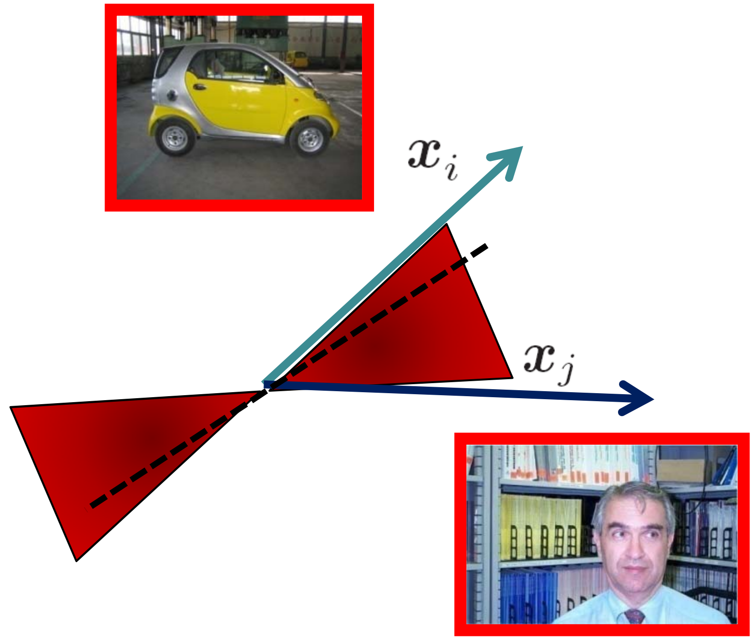
[Weinberger et al. 2004, Hertz et al. 2004, Frome et al. 2007, Varma & Ray 2007, Kumar et al. 2007]

Our idea: Semi-supervised hash functions



$$h(x_i) = h(x_j)$$

Less likely to split pairs like those with similarity constraint



$$h(x_i) \neq h(x_j)$$

More likely to split pairs like those with dissimilarity constraint

[Jain, Kulis, & Grauman, CVPR 2008]

Semi-supervised hash functions

- Given learned Mahalanobis metric, $A = G^T G$
- We generate parameterized hash functions for $s_A(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T A \boldsymbol{x}_j$:

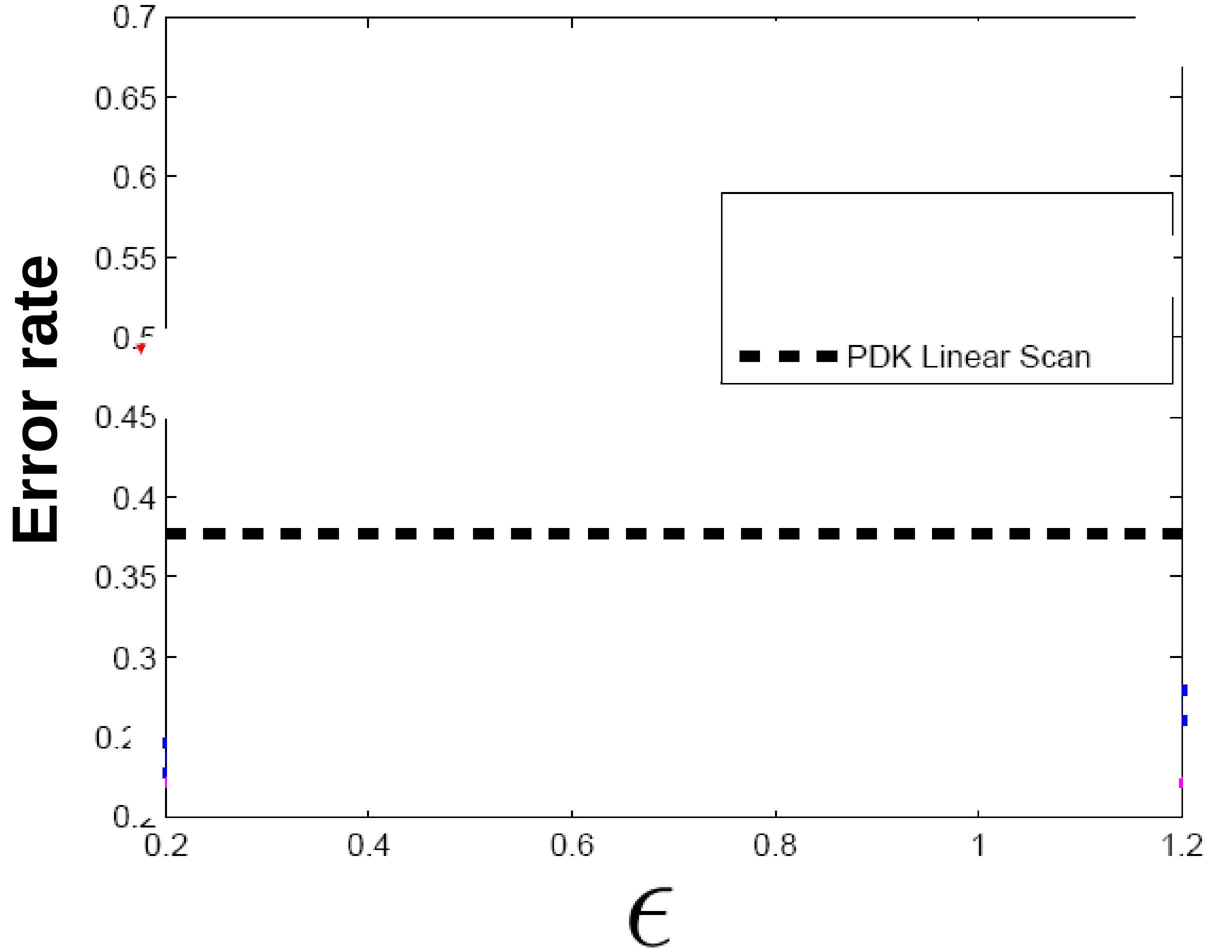
$$h_{\boldsymbol{r},A}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{r}^T G \boldsymbol{x} \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

Satisfies the locality-sensitivity condition:

$$\Pr\left[h_{\boldsymbol{r},A}(\boldsymbol{x}_i) = h_{\boldsymbol{r},A}(\boldsymbol{x}_j)\right] = 1 - \frac{1}{\pi}\cos^{-1}\left(\frac{\boldsymbol{x}_i^T A \boldsymbol{x}_j}{\sqrt{|G\boldsymbol{x}_i||G\boldsymbol{x}_j|}}\right)$$

[Jain, Kulis, & Grauman, CVPR 2008]

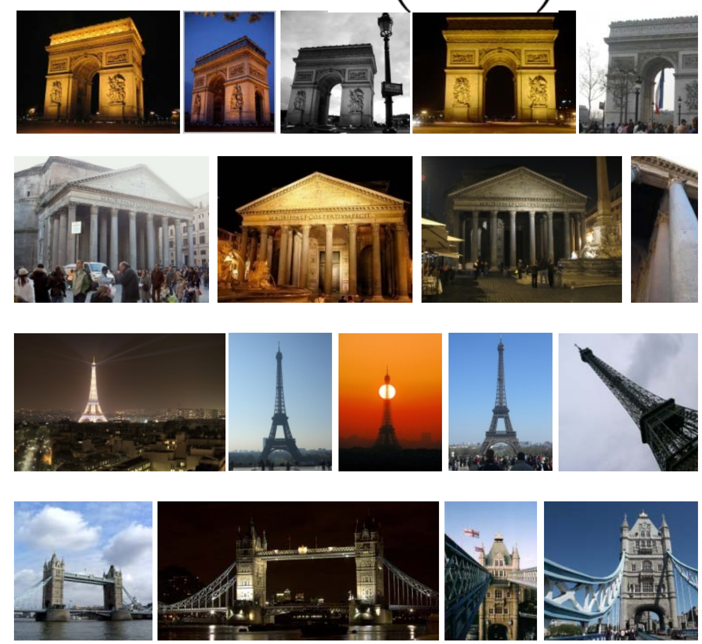
Semi-supervised hash functions



slower search ⟷ **faster search**

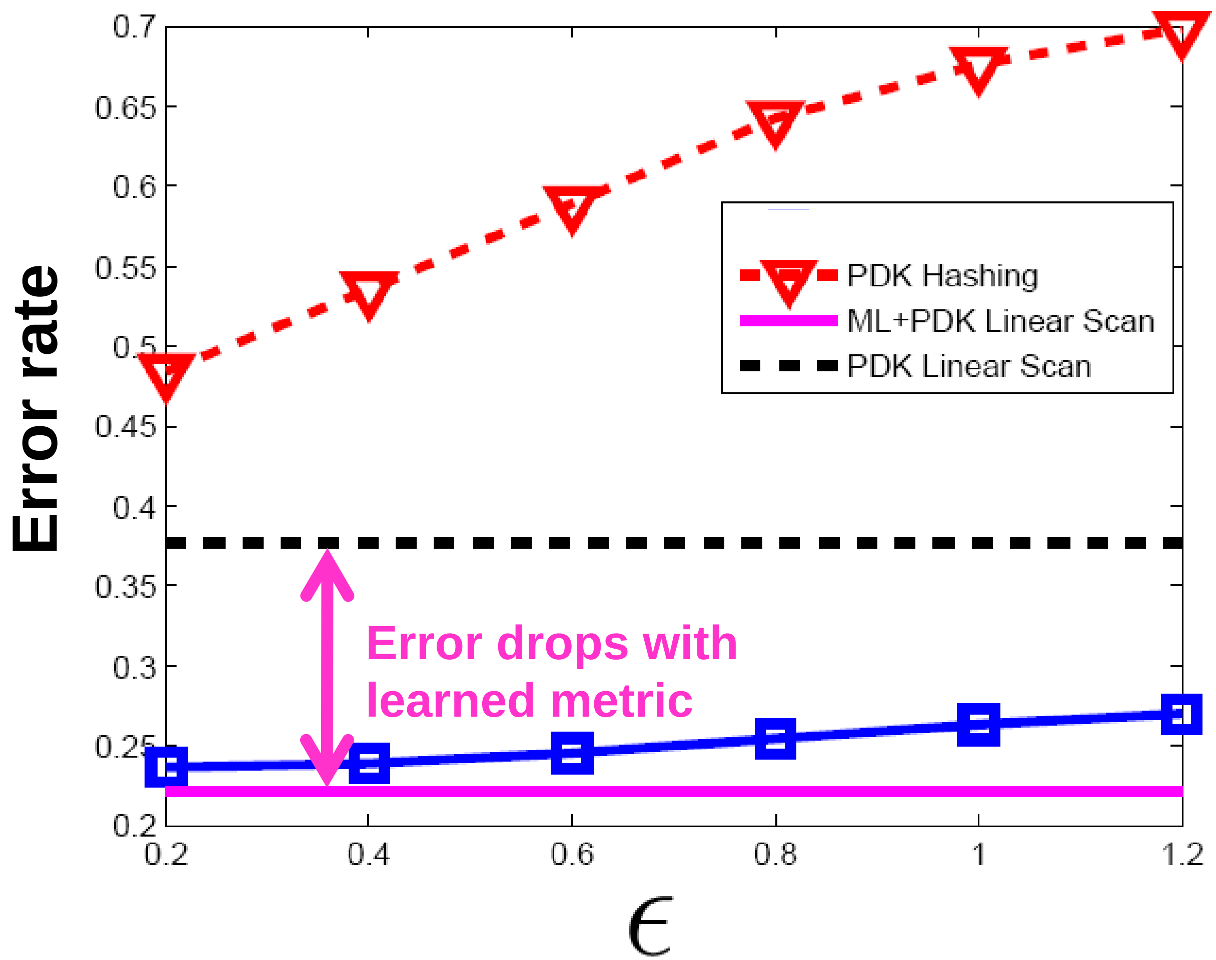
[Kulis, Jain, & Grauman, PAMI 2009]

Query time: $O\left(N^{\frac{1}{1+\epsilon}}\right)$



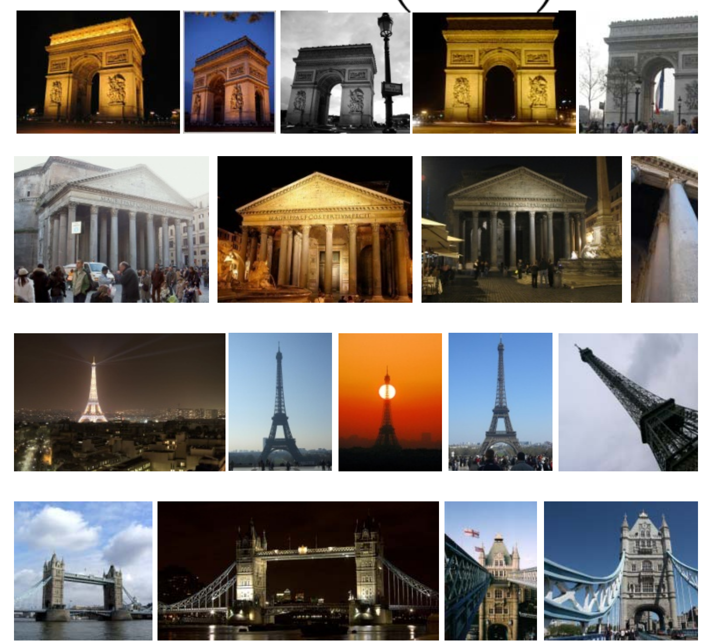
- Flickr dataset
- Categorize scene based on nearest exemplars
- Base metric: Ling & Soatto's Proximity Distribution Kernel (PDK)

Semi-supervised hash functions



slower search ⟷ **faster search**

Query time: $O\left(N^{\frac{1}{1+\epsilon}}\right)$



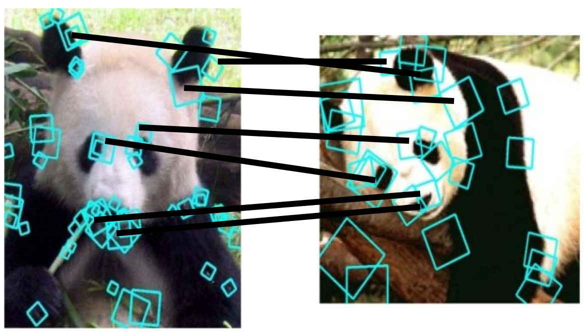
- Flickr dataset
- Categorize scene based on nearest exemplars
- Base metric: Ling & Soatto's Proximity Distribution Kernel (PDK)

[Kulis, Jain, & Grauman, PAMI 2009]

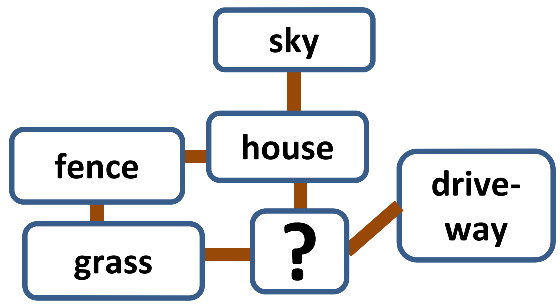
Searching with kernel functions

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$$

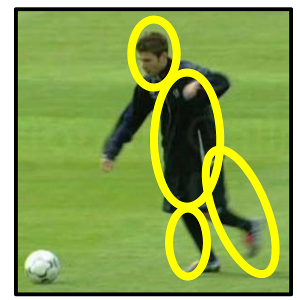
Kernels encompass many useful similarity measures, many for structured input data.



sets



graphs

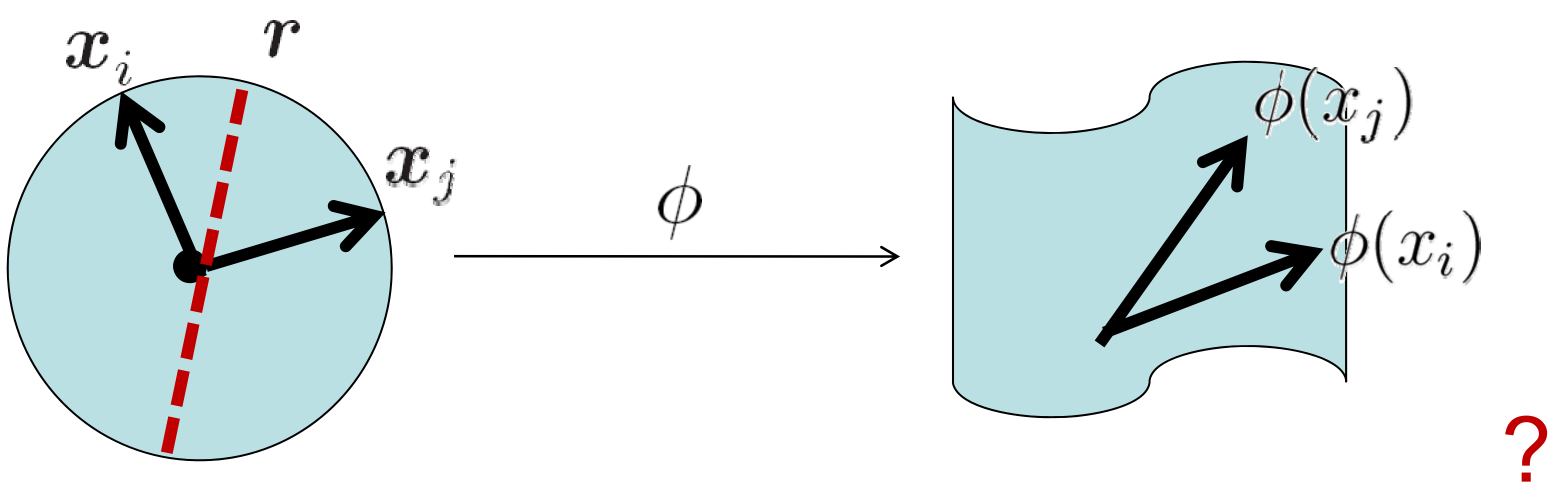


trees

How can we search efficiently with an *arbitrary* kernel function?

Hash functions for kernels?

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$$



$$h_{\boldsymbol{r}}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{r}^T\boldsymbol{x} \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

$$h_{\boldsymbol{r}}(\phi(\boldsymbol{x})) = \begin{cases} 1, & \text{if } \boldsymbol{r}^T\phi(\boldsymbol{x}) \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

$$r_i \sim \mathcal{N}(0, 1)$$

$$r_i \sim \mathcal{N}(0, 1)$$

Our idea: Kernelized LSH (KLSH)

Main idea:

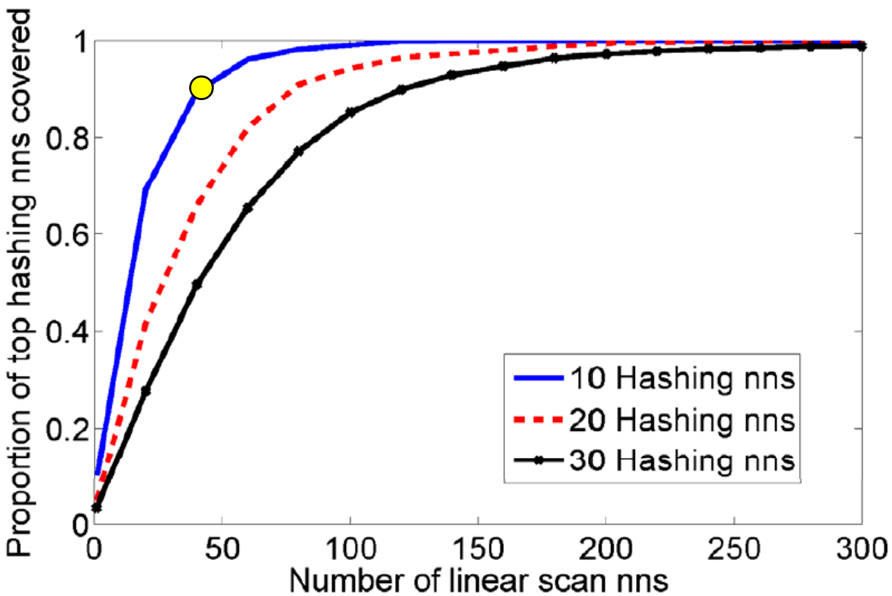
- Draw on Central Limit Theorem to (implicitly) generate random Gaussian hyperplanes in the kernel-induced feature space.
- Show that products with those hyperplanes require only kernel and sparse set of data objects.

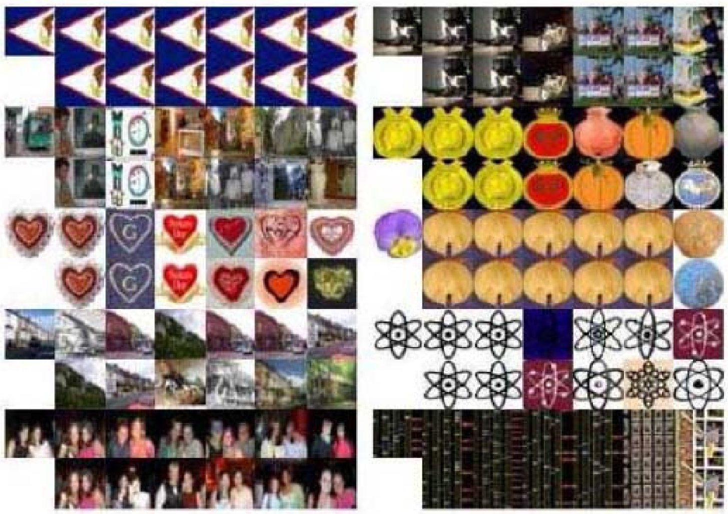
$$h_{\boldsymbol{r}}(\phi(\boldsymbol{x})) = \begin{cases} 1, & \text{if } \sum_i \boldsymbol{w}(i)\kappa(\boldsymbol{x}, \boldsymbol{x}_i) \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

[Kulis & Grauman, ICCV 2009] Kristen Grauman, UT-Austin

Result: Kernelized LSH (KLSH)

80 Million Tiny Images dataset





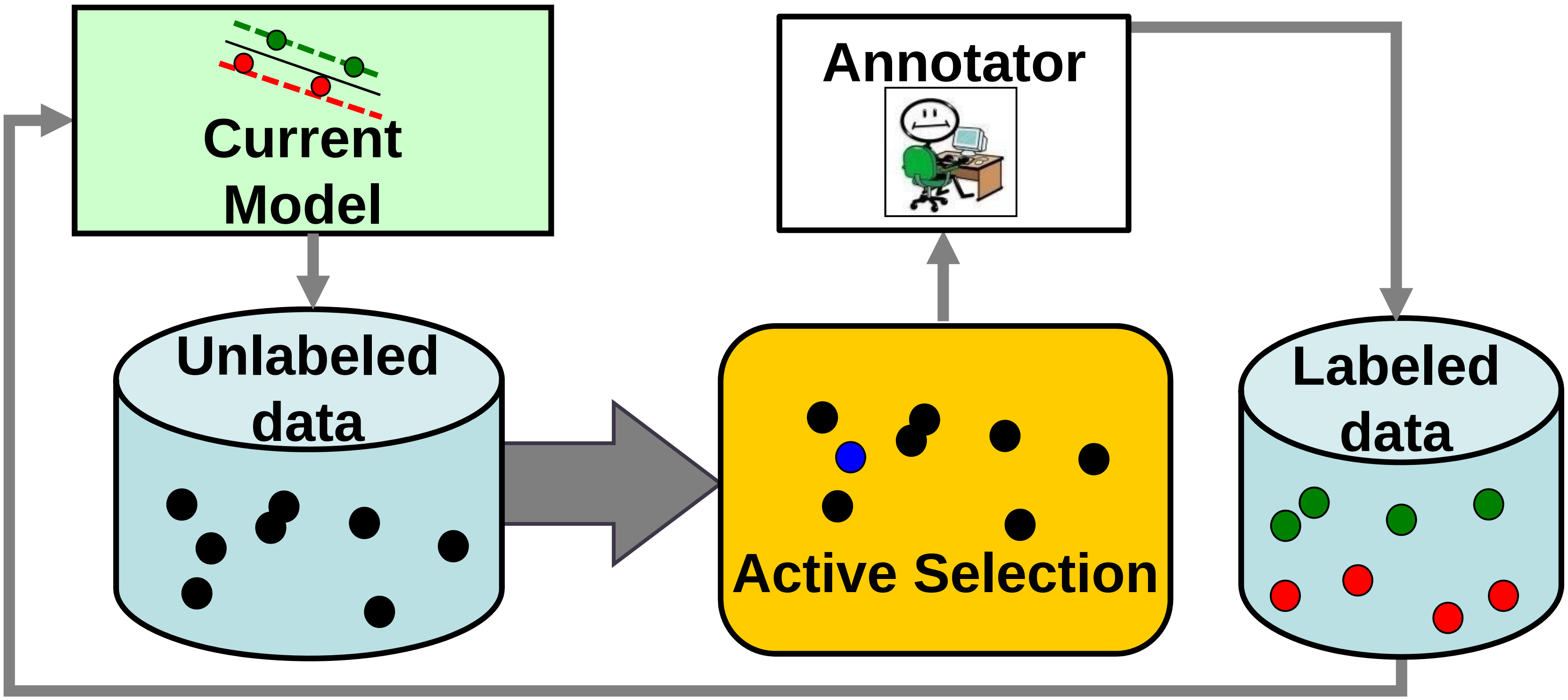
- Gist descriptor + Gaussian RBF kernel
- KLSH **searches less than 1%** of the database to find a query's approximate near neighbors.

Retrieval on a budget

- Retrieving *similar* instances with a search time budget
  - Novel hash functions for learned metrics and arbitrary kernel functions

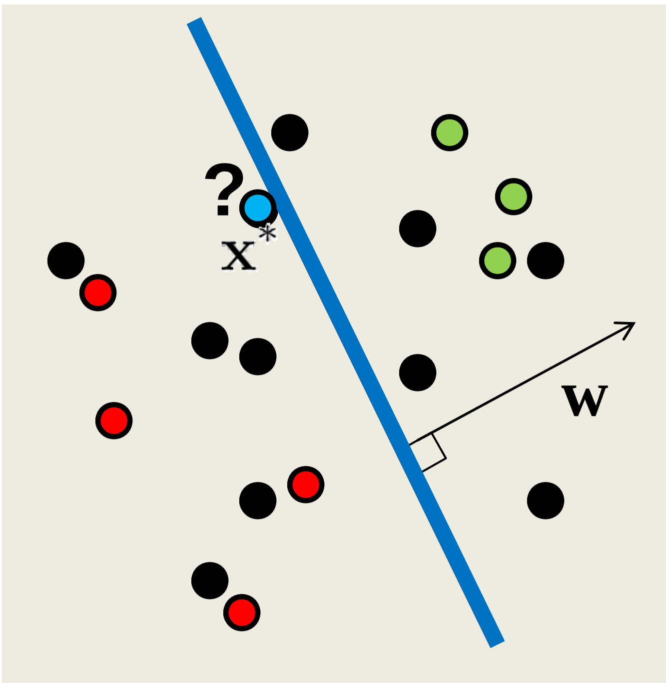
- Retrieving *informative* instances with a search time or annotation cost budget
  - Novel hash functions for hyperplane queries
  - Budgeted batch-mode active selection

Active selection: retrieving informative instances



We have demands on *both* search time and annotation resources.

SVM margin criterion for active selection



Select point nearest to hyperplane decision boundary for labeling.

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

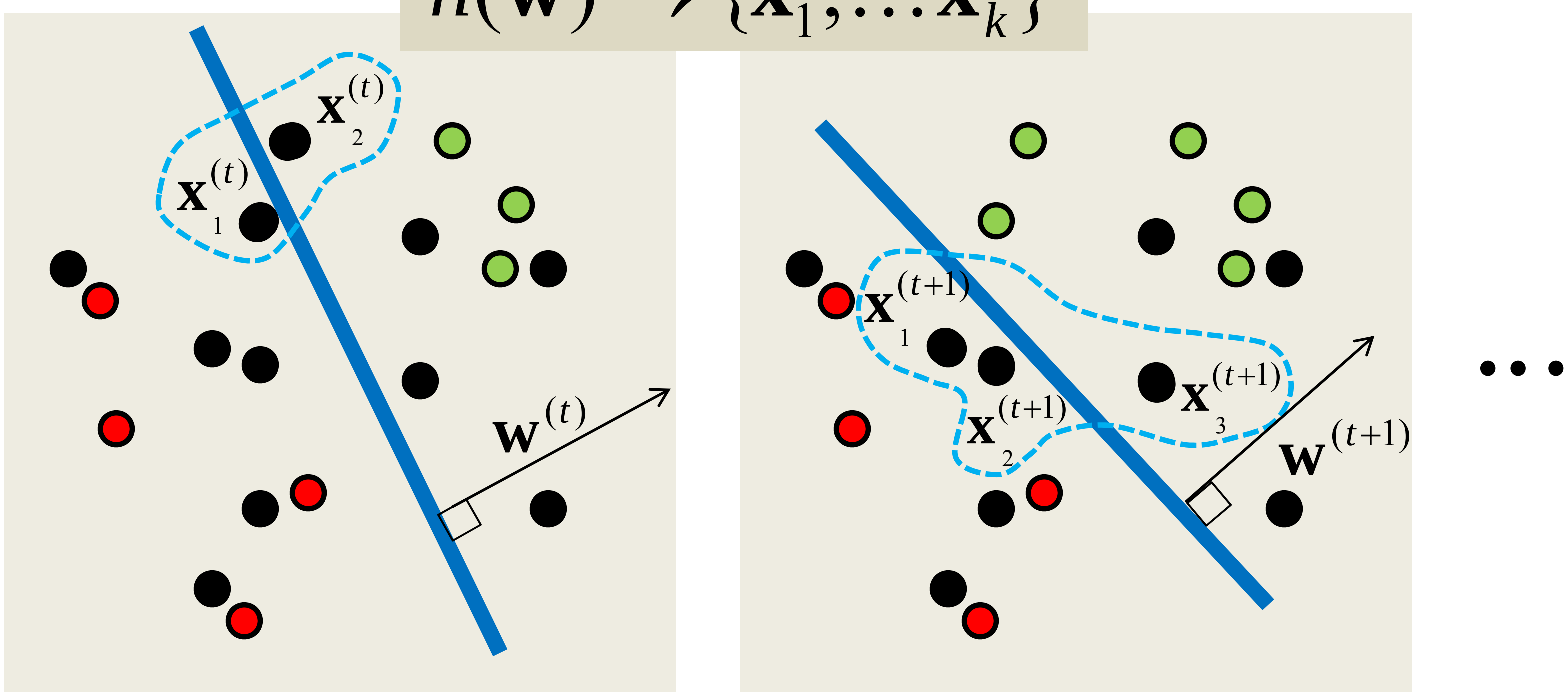
[Tong & Koller, 2000; Schohn & Cohn, 2000; Campbell et al. 2000]

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

Sub-linear time active selection

Goal: Map <u>hyperplane query</u> directly to its nearest points.

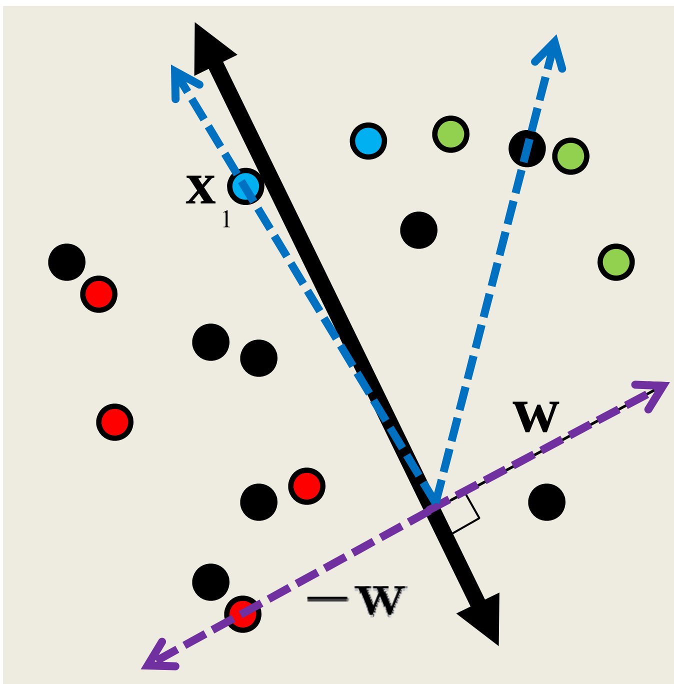
$$h(\mathbf{w}) \rightarrow \{\mathbf{x}_1, \ldots \mathbf{x}_k\}$$



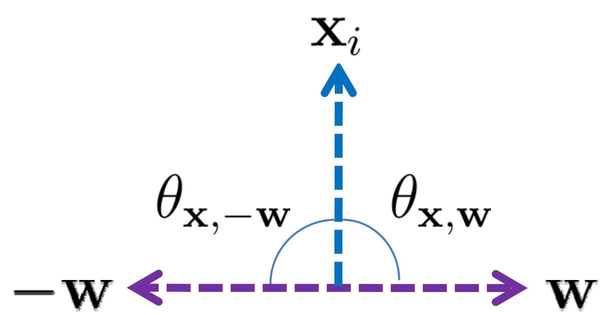
[Jain, Vijayanarasimhan & Grauman, NIPS 2010]

Hashing a hyperplane query

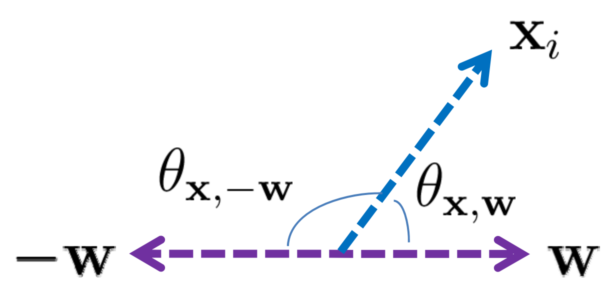
To retrieve those points for which $|\mathbf{w}^T \mathbf{x}_i|$ is small, want probable collision for perpendicular vectors:



Assuming normalized data.



Most likely to collide



Unlikely to collide

[Jain, Vijayanarasimhan & Grauman, NIPS 2010]

Hashing a hyperplane query

To achieve this, we define asymmetric two-bit hash:

Let: $h_{\boldsymbol{u},\boldsymbol{v}}(\boldsymbol{a},\boldsymbol{b}) = [h_{\boldsymbol{u}}(\boldsymbol{a}), h_{\boldsymbol{v}}(\boldsymbol{b})] = [\mathrm{sign}(\boldsymbol{u}^T\boldsymbol{a}), \mathrm{sign}(\boldsymbol{v}^T\boldsymbol{b})]$

$\boldsymbol{u}, \boldsymbol{v} \sim \mathcal{N}(0, I)$

Hashing a hyperplane query

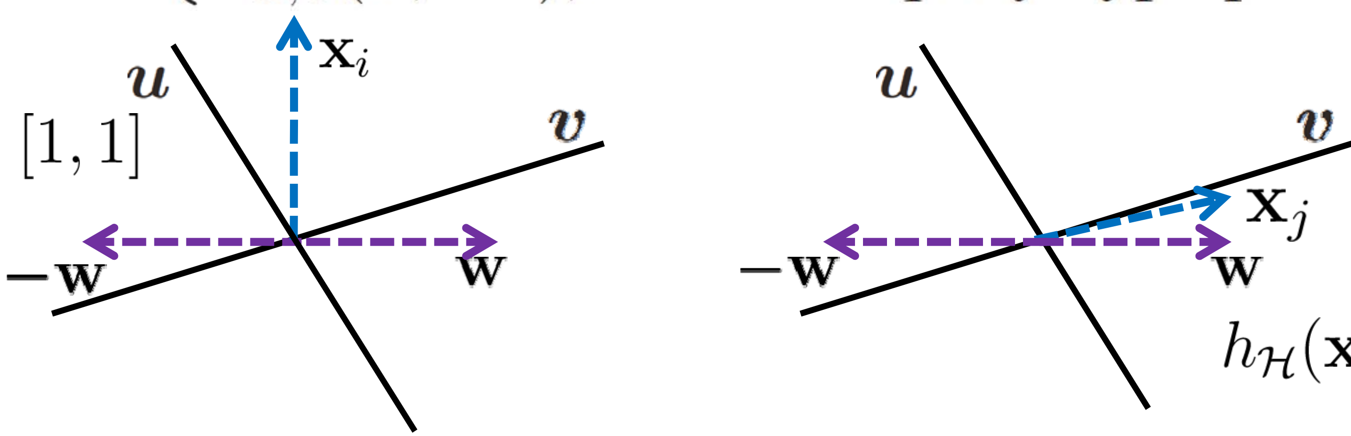
To achieve this, we define asymmetric two-bit hash:

Let: $h_{\boldsymbol{u},\boldsymbol{v}}(\boldsymbol{a},\boldsymbol{b}) = [h_{\boldsymbol{u}}(\boldsymbol{a}), h_{\boldsymbol{v}}(\boldsymbol{b})] = [\text{sign}(\boldsymbol{u}^T\boldsymbol{a}), \text{sign}(\boldsymbol{v}^T\boldsymbol{b})]$

Then define:

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

$h_{\mathcal{H}}(\mathbf{x}_i) = [1,1]$



$h_{\mathcal{H}}(\mathbf{x}_j) = [1,0]$

Likely to collide

Unlikely to collide

$h_{\mathcal{H}}(\mathbf{w}) = [1,1]$

Kristen Grauman, UT-Austin

Hashing a hyperplane query

To achieve this, we define asymmetric two-bit hash:

Let: $h_{\boldsymbol{u},\boldsymbol{v}}(\boldsymbol{a},\boldsymbol{b}) = [h_{\boldsymbol{u}}(\boldsymbol{a}), h_{\boldsymbol{v}}(\boldsymbol{b})] = [\text{sign}(\boldsymbol{u}^T\boldsymbol{a}), \text{sign}(\boldsymbol{v}^T\boldsymbol{b})]$

Then define:

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

We prove necessary LSH bounds, e.g.:

$$\Pr[h_{\mathcal{H}}(\boldsymbol{w}) = h_{\mathcal{H}}(\boldsymbol{x})] = \Pr[h_{\boldsymbol{u}}(\boldsymbol{w}) = h_{\boldsymbol{u}}(\boldsymbol{x})] \Pr[h_{\boldsymbol{v}}(-\boldsymbol{w}) = h_{\boldsymbol{v}}(\boldsymbol{x})]$$

$$= \frac{1}{4} - \frac{1}{\pi^2}\left(\theta_{\boldsymbol{x},\boldsymbol{w}} - \frac{\pi}{2}\right)^2$$

See *[Jain, Vijayanarasimhan & Grauman, NIPS 2010].*

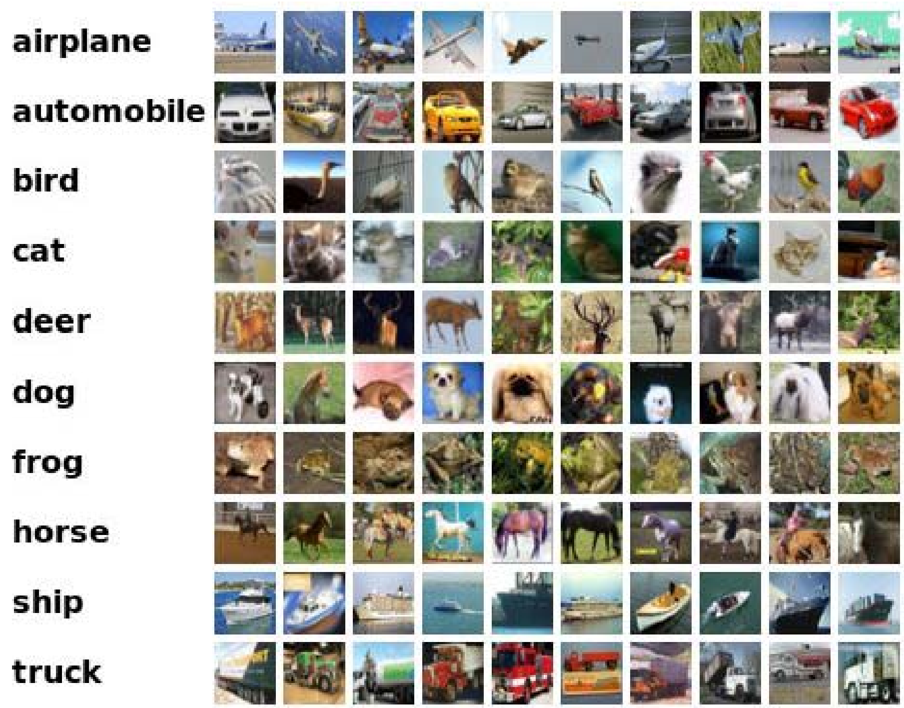
Data flow: Hashing a hyperplane query

- Hash all unlabeled data into table.
- Active selection loop:
  - Hash current hyperplane as query.
  - Retrieve unlabeled data points with which it collides.
  - Request labels for them.
  - Update hyperplane.

Kristen Grauman, UT-Austin

Results: Hashing a hyperplane query

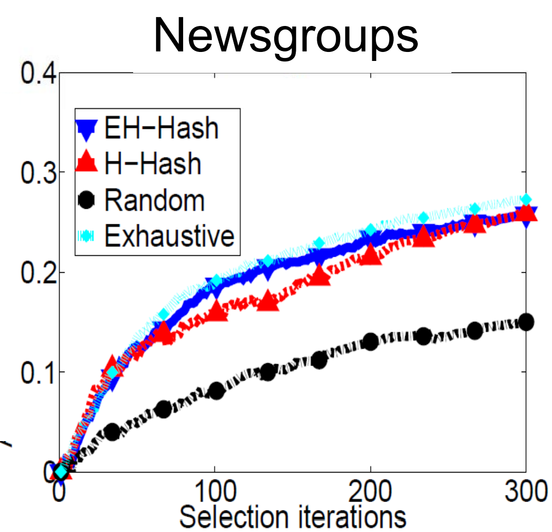
- **Tiny-1M**
  - 1 Million images from 1000s of categories
- **CIFAR-10**
  - 60,000 images in 10 categories
- **Newsgroups**
  - 20,000 documents in 20 categories

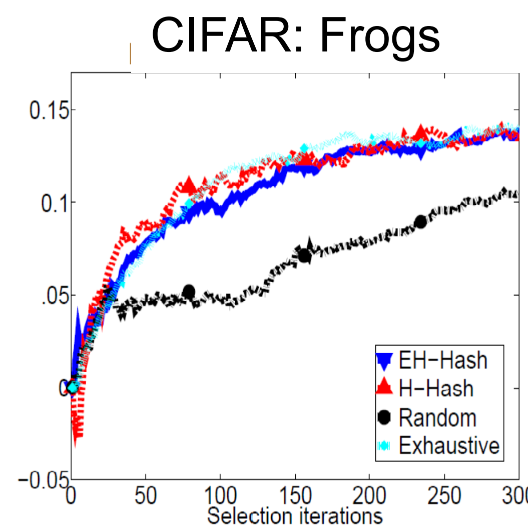


comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

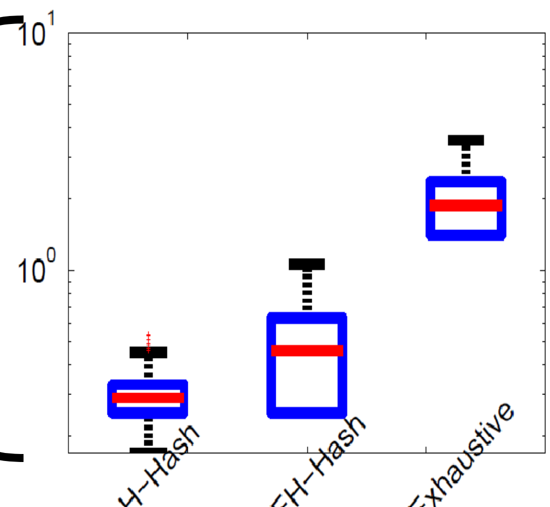
Results: Hashing a hyperplane query

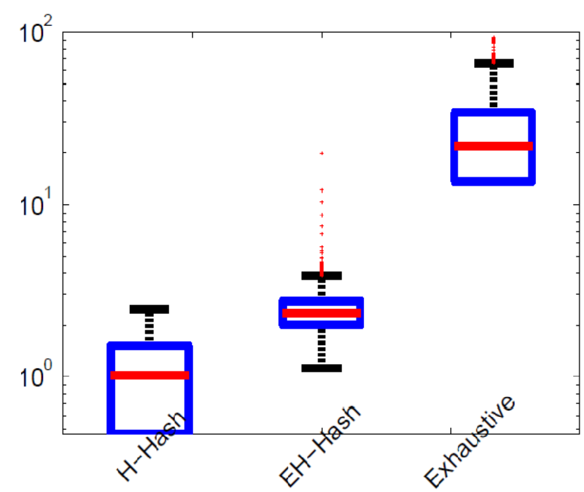
Accuracy improvements as more data labeled



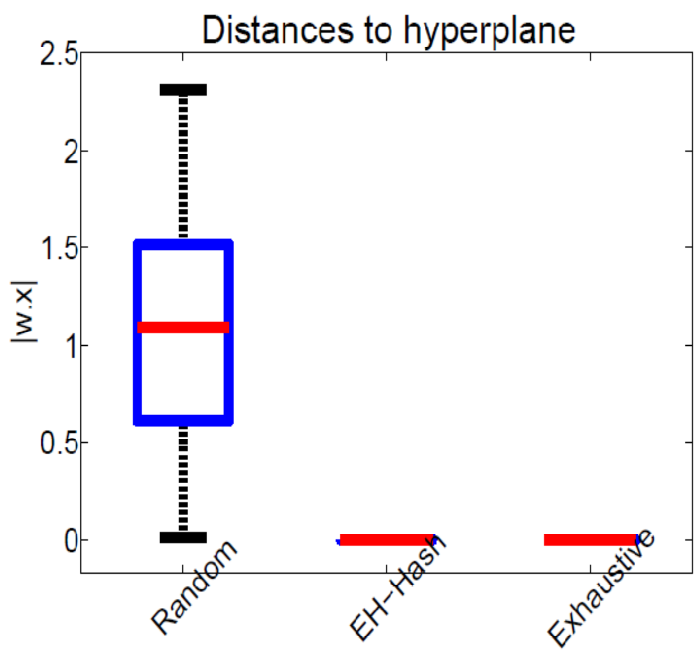


Time spent searching for selection (log scale)





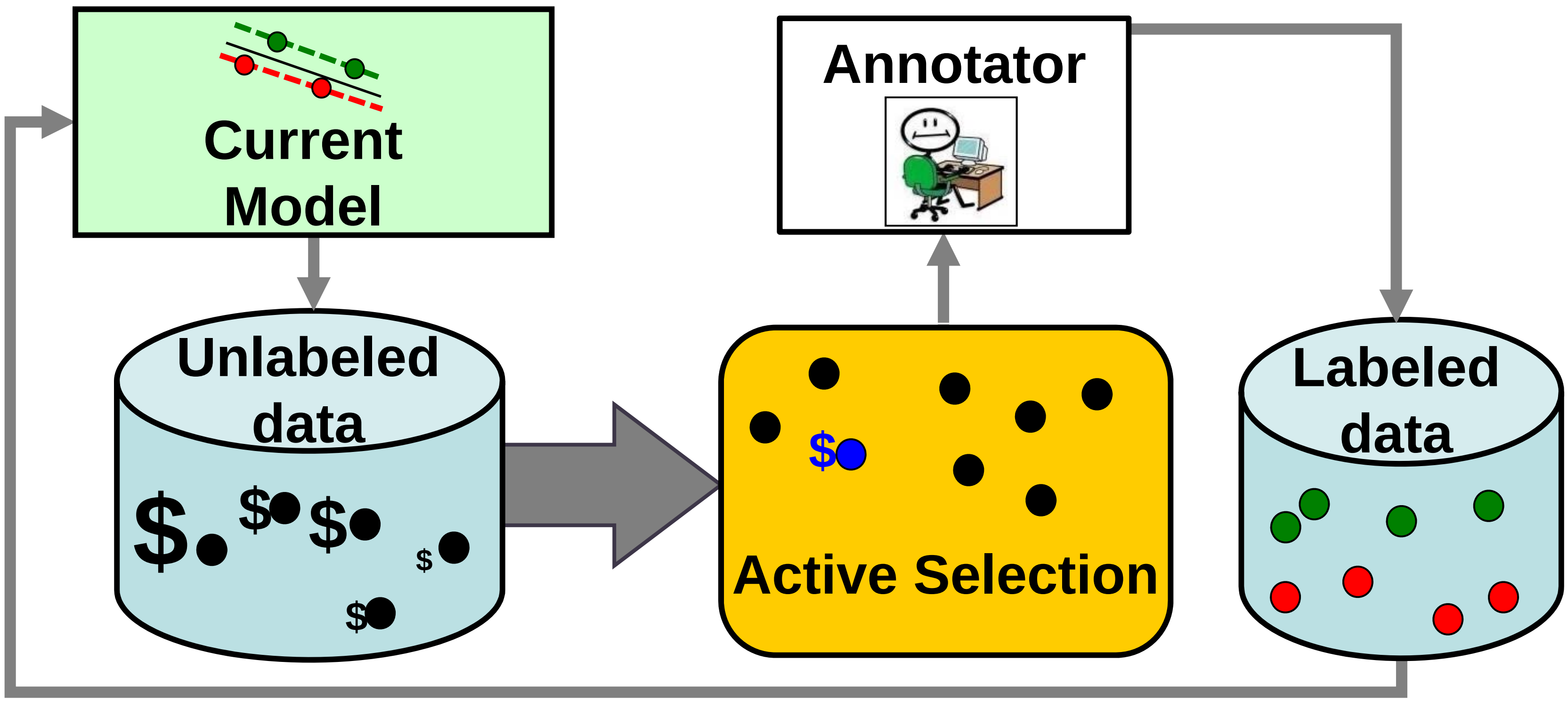
Results: Hashing a hyperplane query



Selected for labeling in first 10 iterations

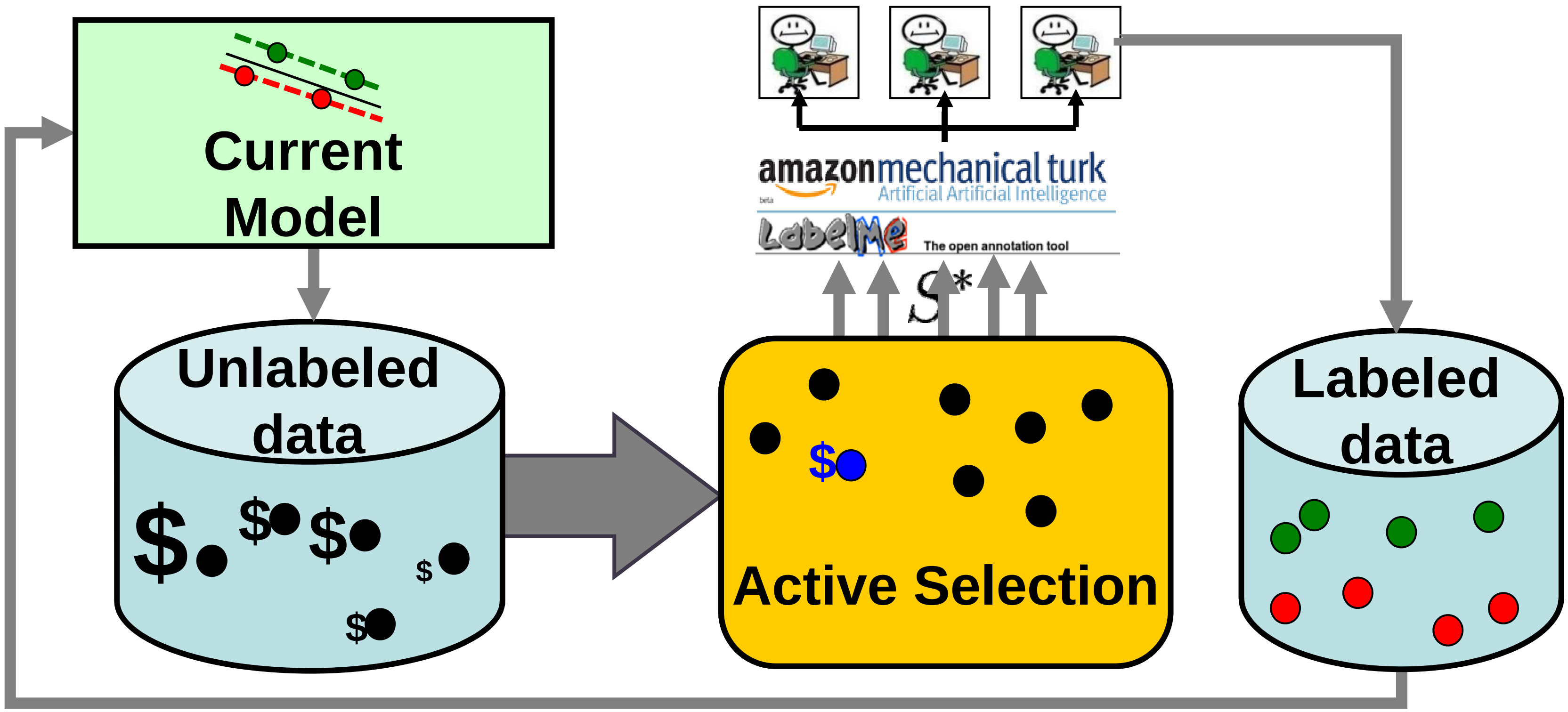
Efficient active selection with pool of
1 Million unlabeled examples!

Active selection: retrieving informative instances



How can we actively leverage many annotators at once?

Our idea:
Budgeted batch active selection



Select a *batch* of examples such that together they most improve classifier objective *and* meet the annotation *budget*.

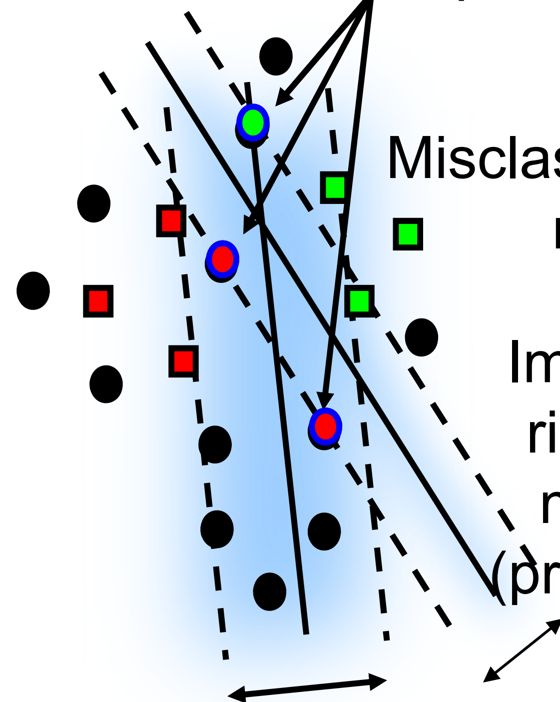
[Vijayanarasimhan et al. CVPR 2010]

Our idea:
Budgeted batch active selection

$$S^* = \text{argmax Pred.Gain}(S)$$

$$s.t. \sum_{x \in S} \text{LabelCost}(x) \leq \text{Budget}$$

S^*

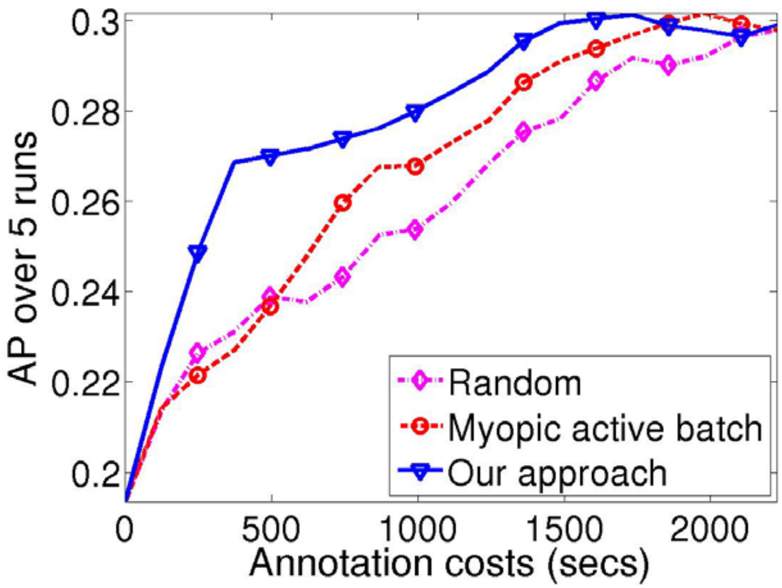


[Vijayanarasimhan et al. CVPR 2010]

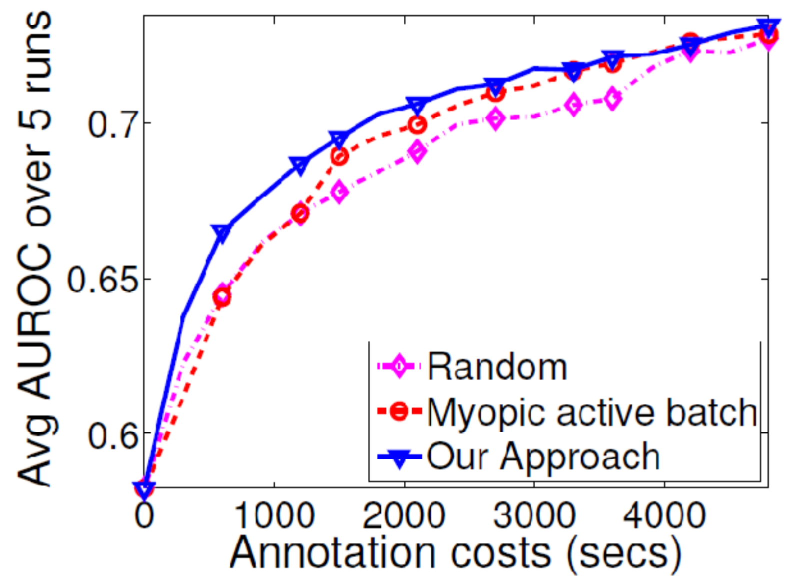
Results: Budgeted batch active selection

Annotation cost = video length, segmentation time.

Hollywood Activities: All 8 classes



SIVAL Object: All 25 classes

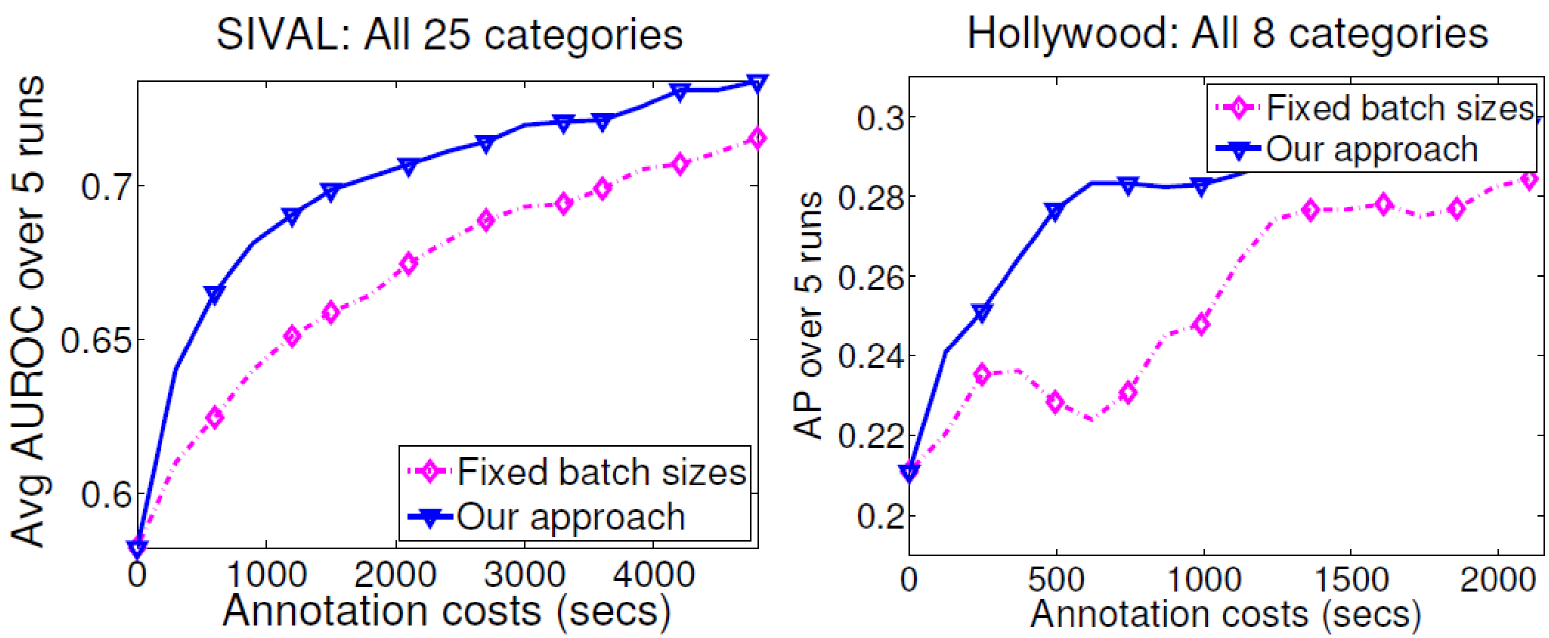


Example selection at a single batch iteration: Positive action class = **Stand up**

Random	Myopic active batch	Ours
 Answer phone	 Answer phone	 Sit down
	 Stand up	 Sit down
		 Sit down
		 Stand up

Results: Budgeted batch active selection

Optimizing a budgeted choice is crucial when candidate annotations vary in cost.



Comparison to state-of-the-art batch-mode active learning approach for choosing fixed-size batches *[Hoi et al. 2009]*.

Summary: Retrieval on a budget

- To perform well with limited resources, we need search and learning algorithms that
  - Offer guarantees on error ↔ search speed tradeoffs
  - Target human supervision to use it most wisely

- Algorithms presented provide
  - New families of locality-sensitive hash functions
  - Large-scale active learning strategies to select points in sub-linear time and/or in batches.