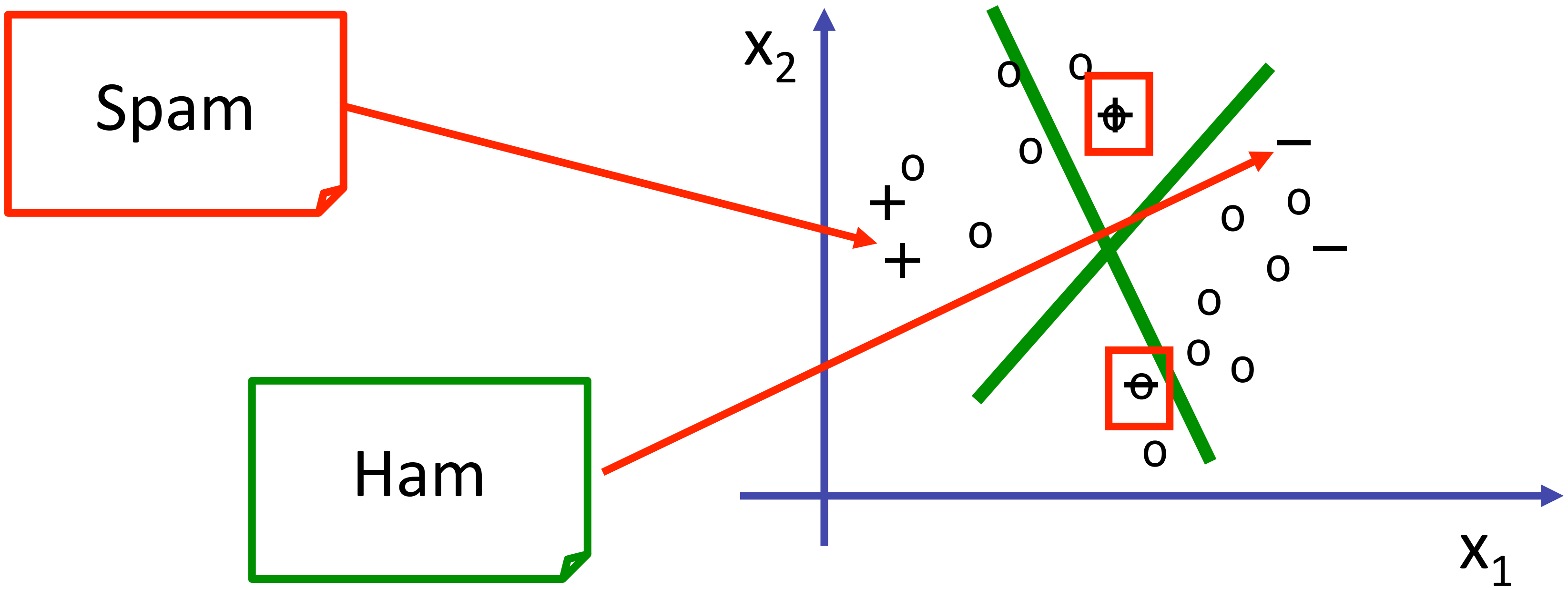
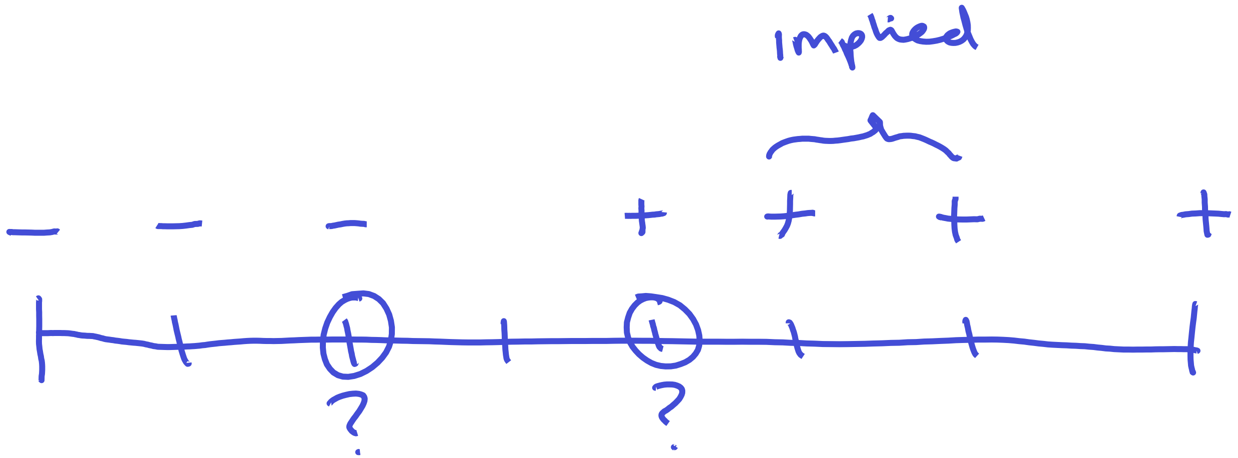
Active learning



- Labels are expensive (need to ask expert)
- **Want to minimize the number of labels**

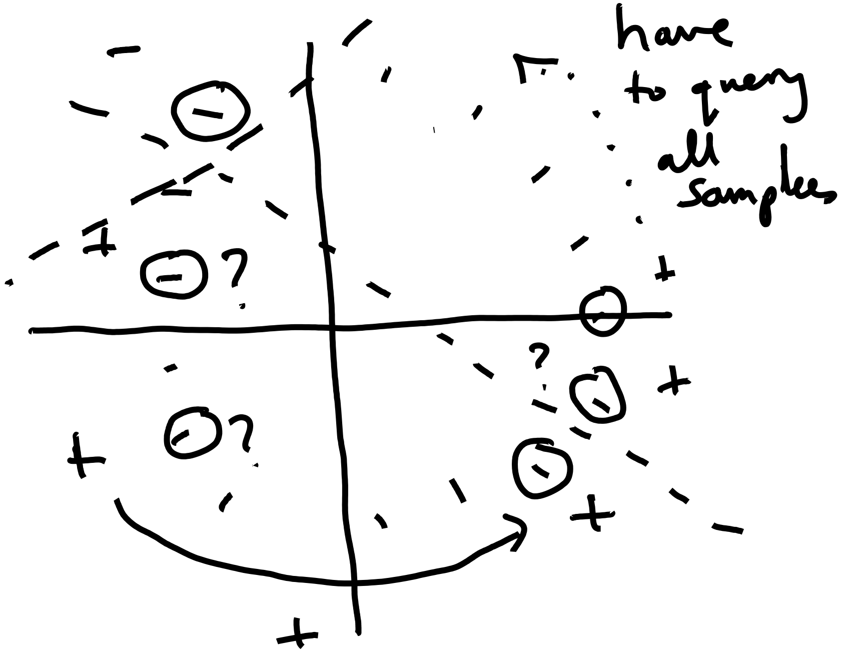
Why should active learning help?

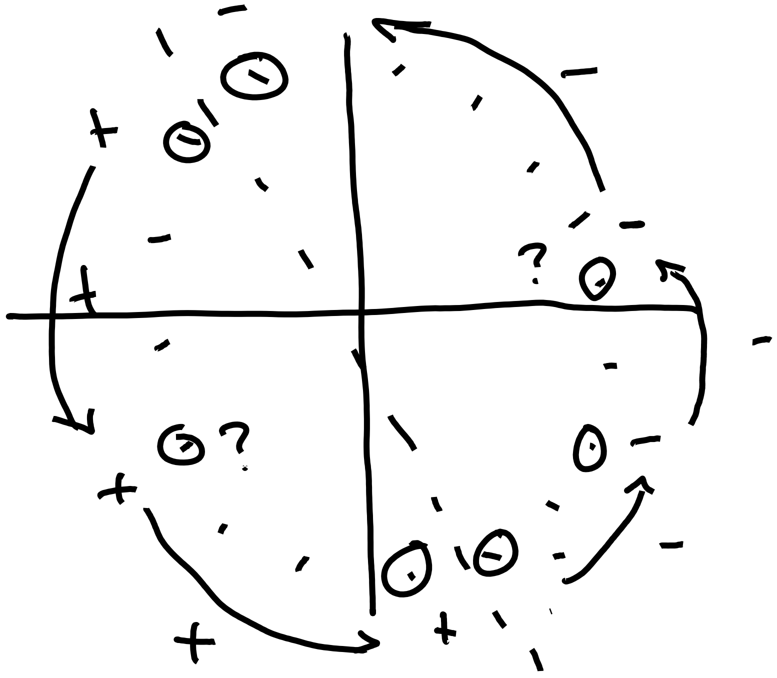
- **Example**: Learning linear separators in 1D
- For now, assume data is noise free



Can learn all labels in $O(\log_2 n)$ queries

Does active learning always help?





Can do binary search

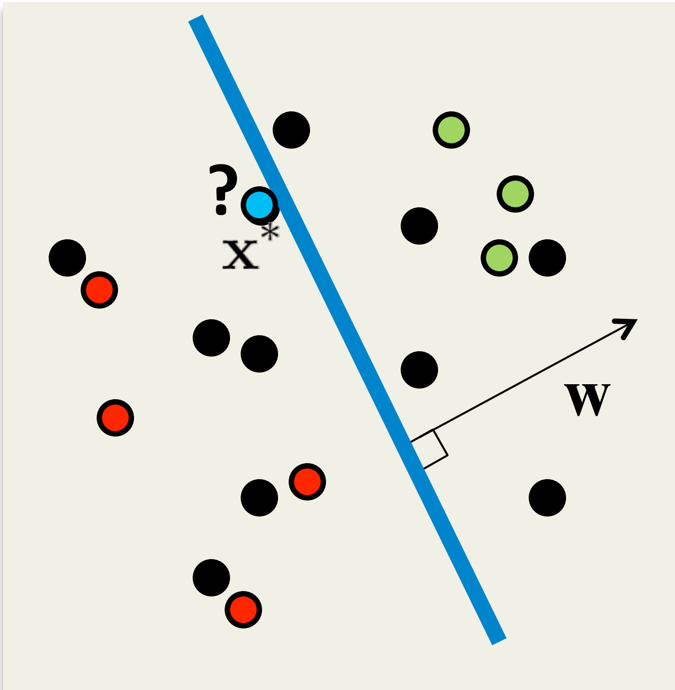
Pool-based active learning

- Pool-based active learning
  - Obtain large pool of unlabeled data
  - Selectively request a few labels, until we can infer all remaining labels
- Resulting classifier "as good" as that obtained from complete labeled set
- Reduction in labels
  - In some cases, exponential reduction possible!
  - In other cases, may need to request almost all labels
- How should we request labels??

Uncertainty sampling

- Given pool of *n* unlabeled examples
- Repeat until we can infer all remaining labels:
  - Assign each unlabeled data an "uncertainty score"
  - Greedily pick the most uncertain example and request label
- One of the most popular heuristics!

Uncertainty sampling in SVMs

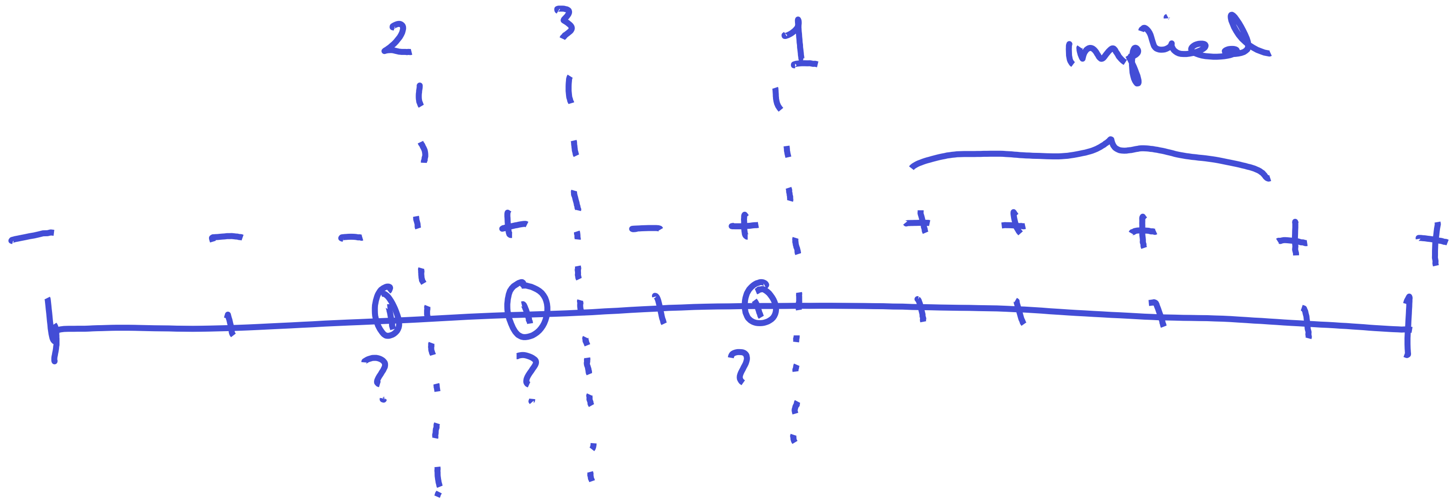


Select point nearest to hyperplane decision boundary for labeling

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

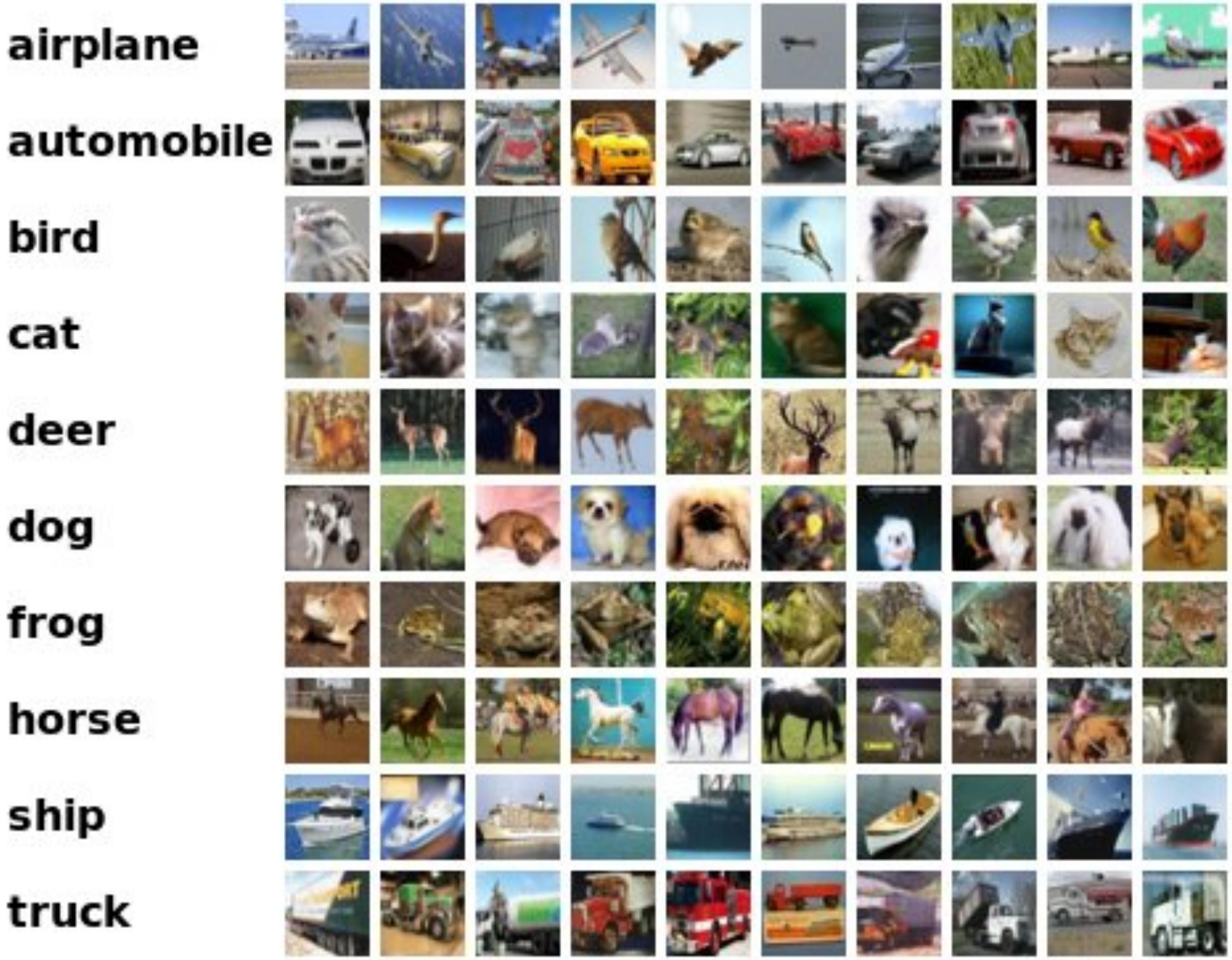
[Tong & Koller, 2000; Schohn & Cohn, 2000; Campbell et al. 2000]

Example: linear classifiers in 1D



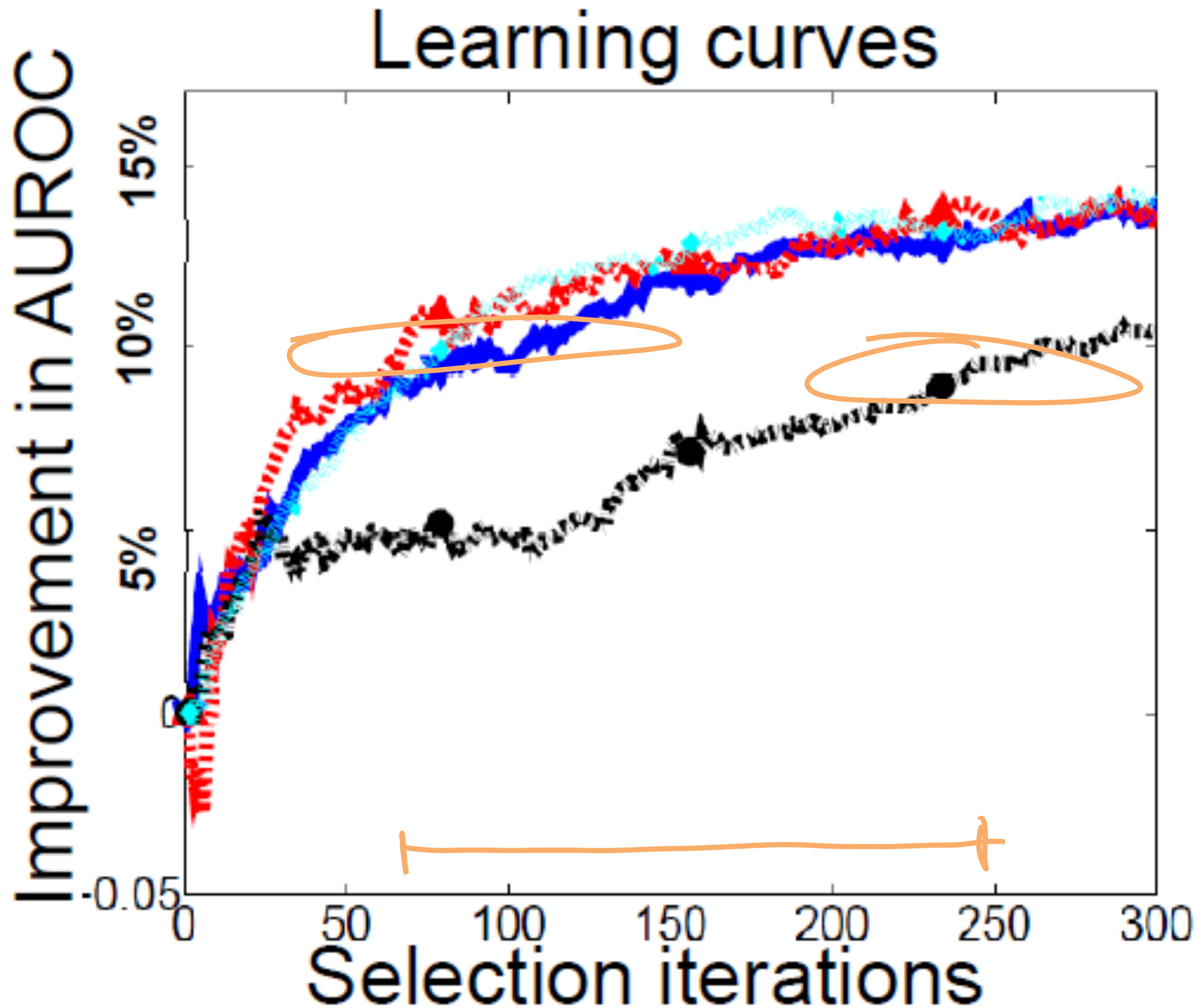
Real data example

[Grauman et al]



Active learning results

[Grauman et al]



Uncertainty sampling in large data

- For i = 1:max_labels
  - For j = 1:n
    - Calculate uncertainty U(j) score of example j
  - Pick most uncertain example
  - Retrain SVM

- Complexity to pick m labels? m << n

for each label

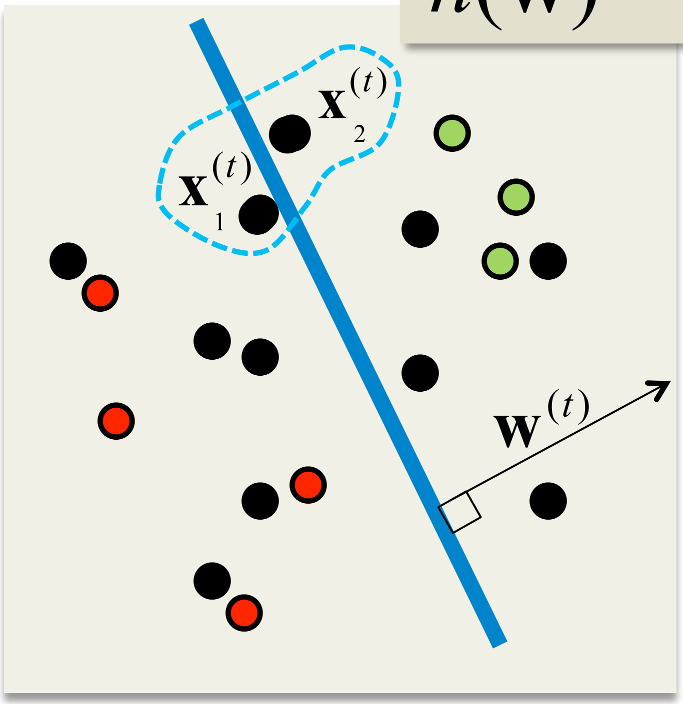
- $|w^T x_i|$ for i=1:n ← expensive

- train SVM ← cheap

Sub-linear time active learning

Goal: Map hyperplane query directly to its nearest points.

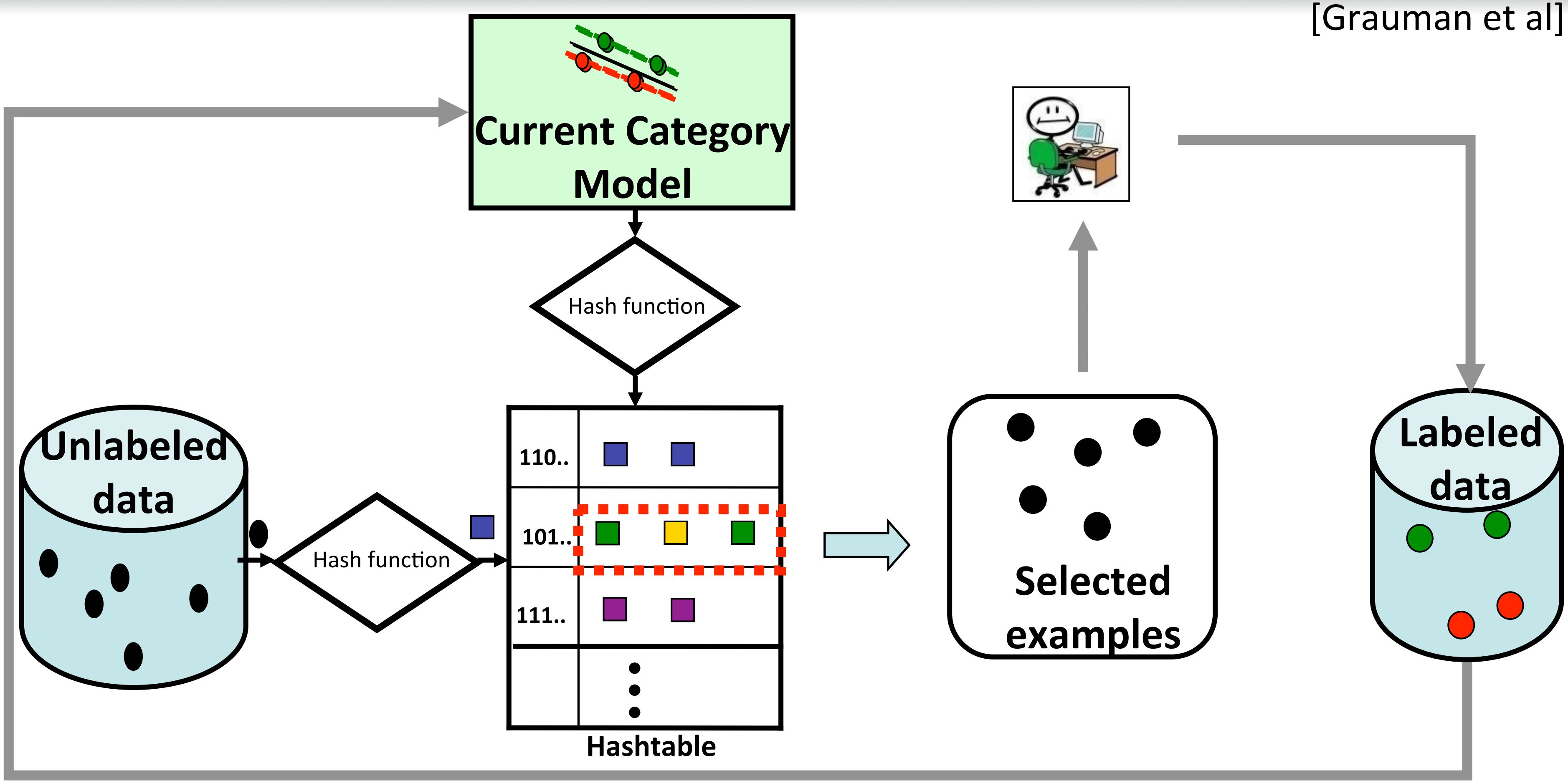
$$h(\mathbf{w}) \rightarrow \{\mathbf{x}_1, \ldots \mathbf{x}_k\}$$



[Jain, Vijayanarasimhan & Grauman, NIPS 2010]

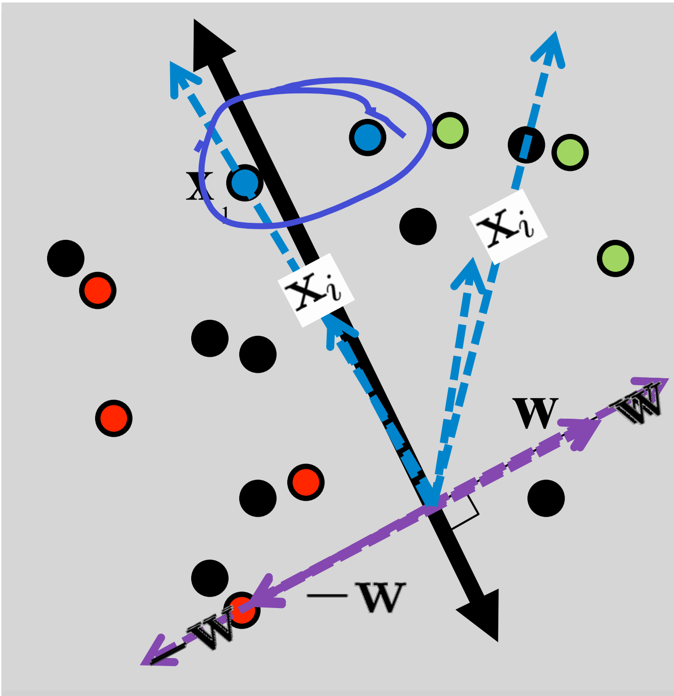
Sub-linear time active selection

[Grauman et al]

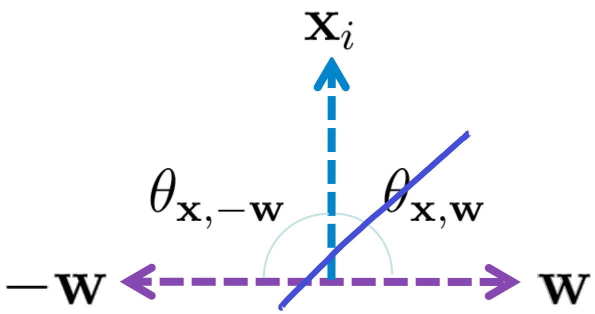


Hashing a hyperplane query

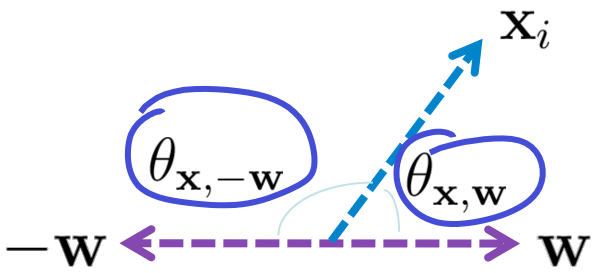
To retrieve those points for which $|\mathbf{w}^T \mathbf{x}_i|$ small, want probable collision for **perpendicular** vectors:



Assuming normalized data.



Most likely to collide

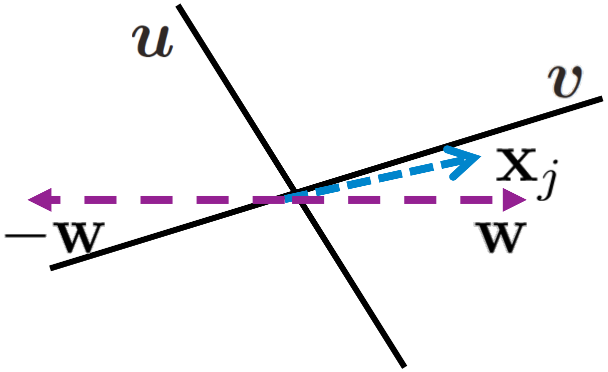


Unlikely to collide

[Jain, Vijayanarasimhan & Grauman, NIPS 2010]

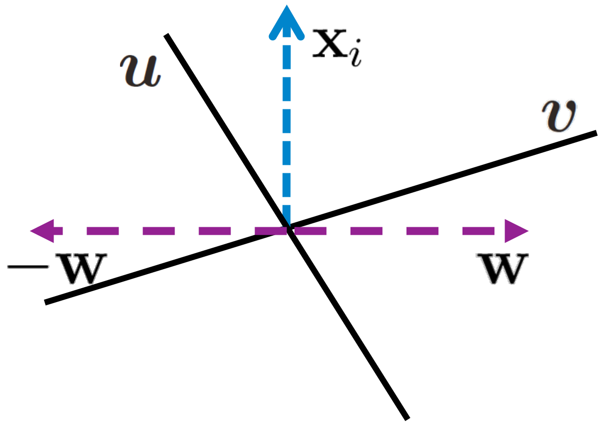
Hashing a hyperplane query

[Grauman et al]



Less likely to split + Highly likely to split

= Unlikely to collide



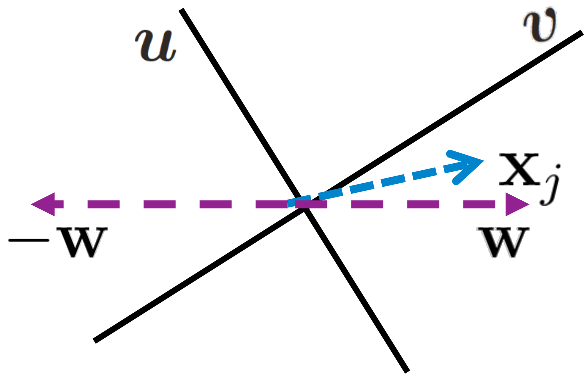
Less likely to split + Less likely to split

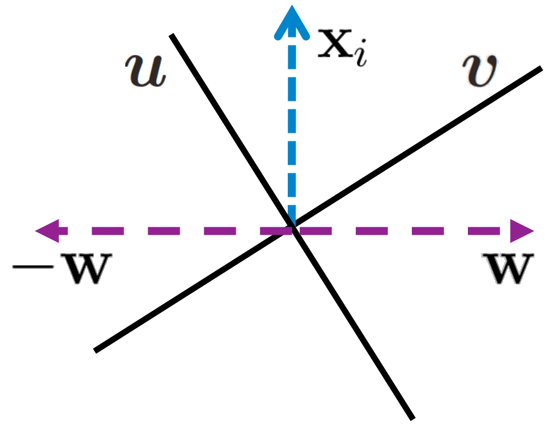
= More likely to collide

- Use two random vectors, two-bit hash key
  - one to constrain the angle with w
  - one to constrain the angle with -w

Hashing a hyperplane query

[Grauman et al]





Less likely to split + Highly likely to split

= Unlikely to collide

Less likely to split + Less likely to split

= More likely to collide

- Use two random vectors, two-bit hash key
  - one to constrain the angle with w
  - one to constrain the angle with -w

Cosine distance LSH

Hashing a hyperplane query

[Grauman et al]

Resulting asymmetric two-bit hash:

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

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

Hashing a hyperplane query

[Grauman et al]

Resulting asymmetric two-bit hash:

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

Define hash family:

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

Can calculate LSH collision probability

Proof in course book

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

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

$\theta \to 0$, $P \to 0$

$\theta \to \frac{\pi}{2}$, $P \to \frac{1}{4}$

Boosting!

[Jain, Vijayanarasimhan & Grauman, NIPS 2010].

Data flow: Hashing a hyperplane query

[Grauman et al]

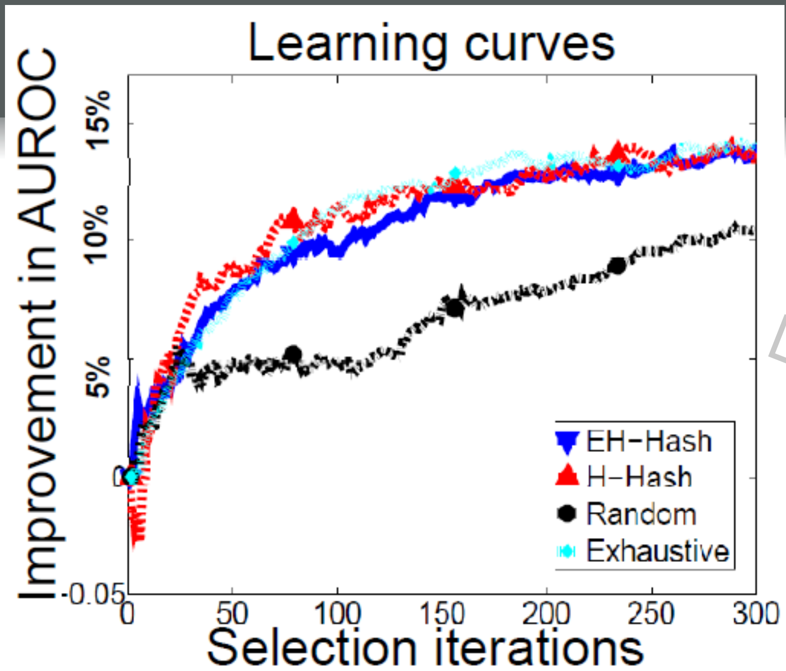
- Hash all unlabeled data into table

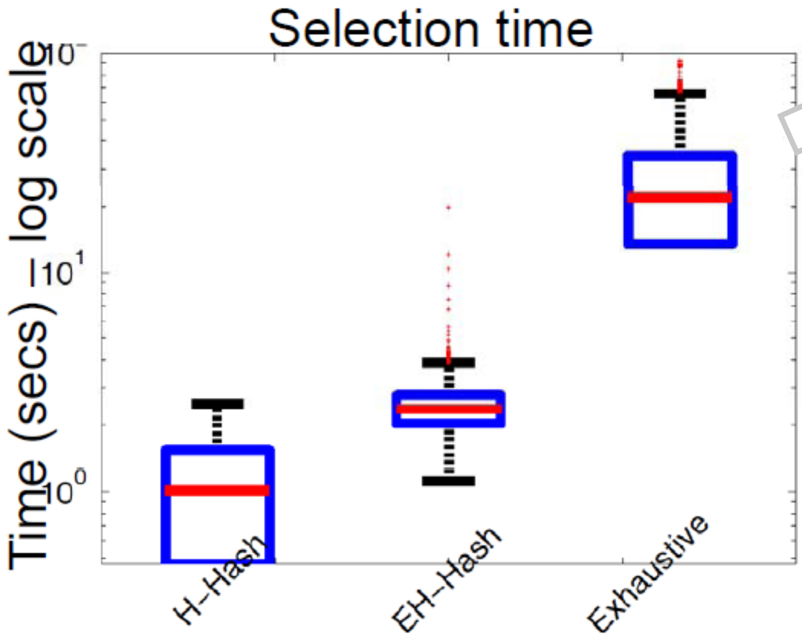
Only need to compute $u^T x_i$'s and $v^T x_i$'s once

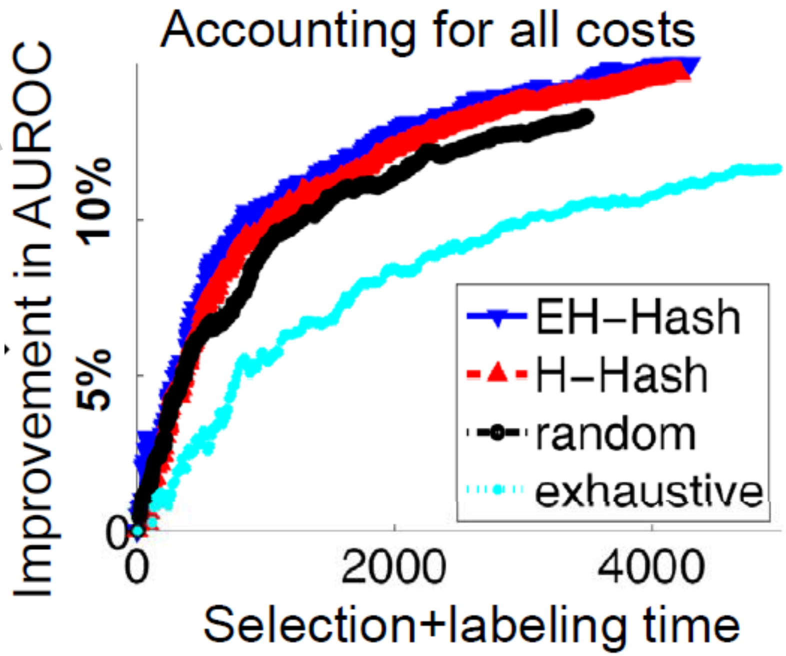
- Active selection loop:
  - Hash current hyperplane as query
  - Retrieve unlabeled data points with which it collides
  - Request labels for them
  - Update hyperplane

Results: Hashing a hyperplane query

[Grauman et al]







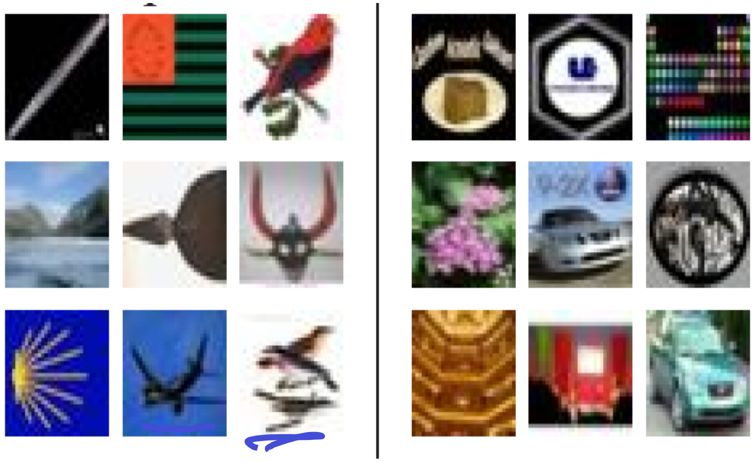
By minimizing **both** selection and labeling time, provide the best accuracy per unit time.

Tiny Images Dataset / CIFAR

Results: Hashing a hyperplane query

[Grauman et al]

Learning "airplane" | Learning "automobile"



we pick images nearer the true boundary

Selected for labeling in first 9 iterations

Efficient active selection with pool of 1 Million unlabeled examples and 1000s of categories.

Summary so far:

- Uncertainty sampling: Simple heuristic for active learning
- For SVMs:
  - pick points closest to decision boundary
  - Can select efficiently using LSH
- Can get significant gains in labeling cost, even for large data sets.

- Now:
  - Theory of active learning
  - Criteria beyond uncertainty sampling

Issues with uncertainty sampling

uncertain ≠ informative!

Defining “informativeness”

- Need to capture how much “information” we gain about the true classifier for each label

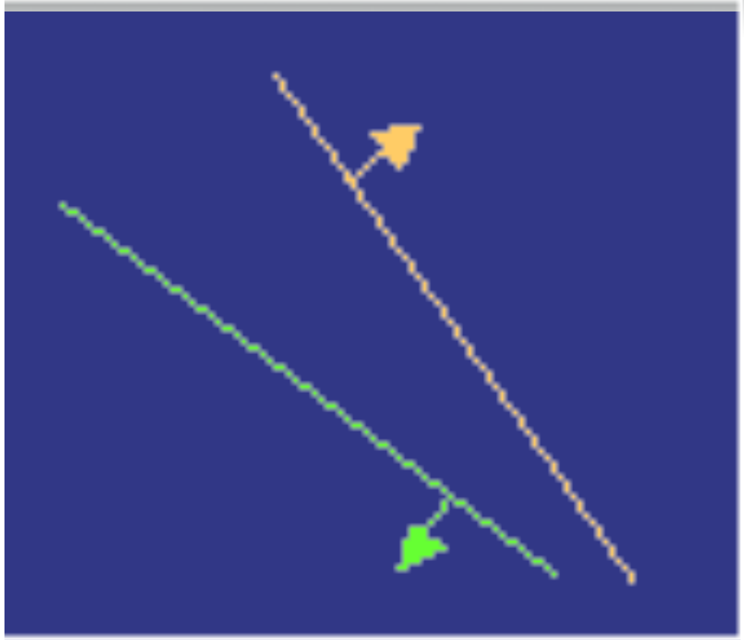
- Version space:
  set of all classifiers consistent with the data

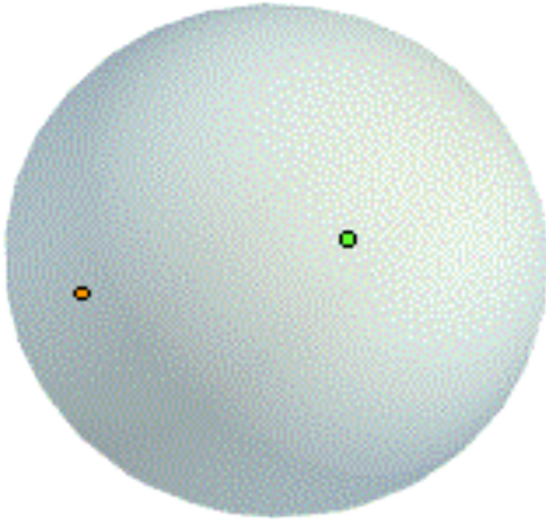
$$\mathcal{V}(D) = \{\mathbf{w} : \forall(\mathbf{x}, y) \in D \;\; \text{sign}(\mathbf{w}^T\mathbf{x}) = y\}$$

- **Idea**:
  would like to shrink version space as quickly as possible

Version space for SVM

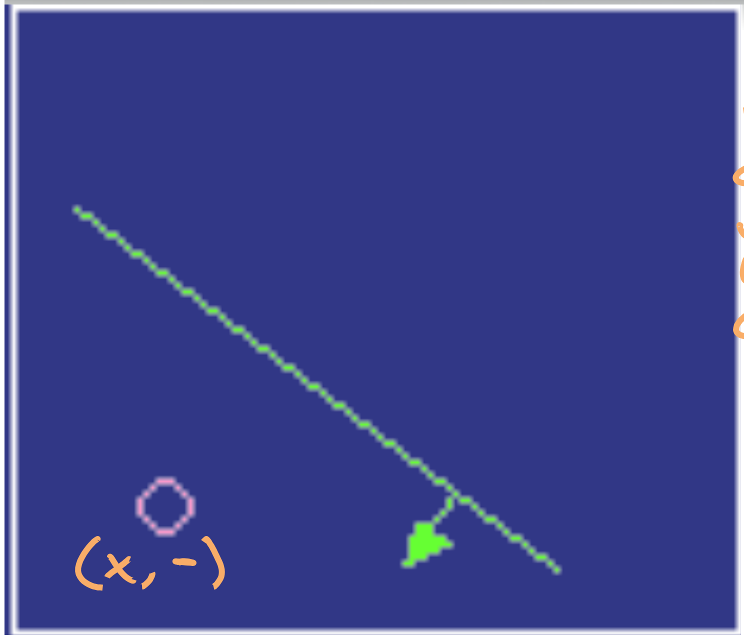
[Tong & Koller]





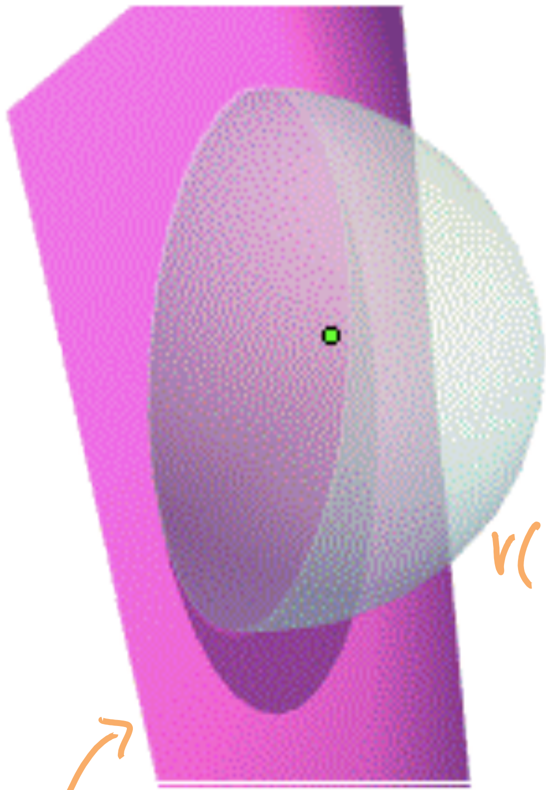
Version space for SVM

[Tong & Koller]



feature space

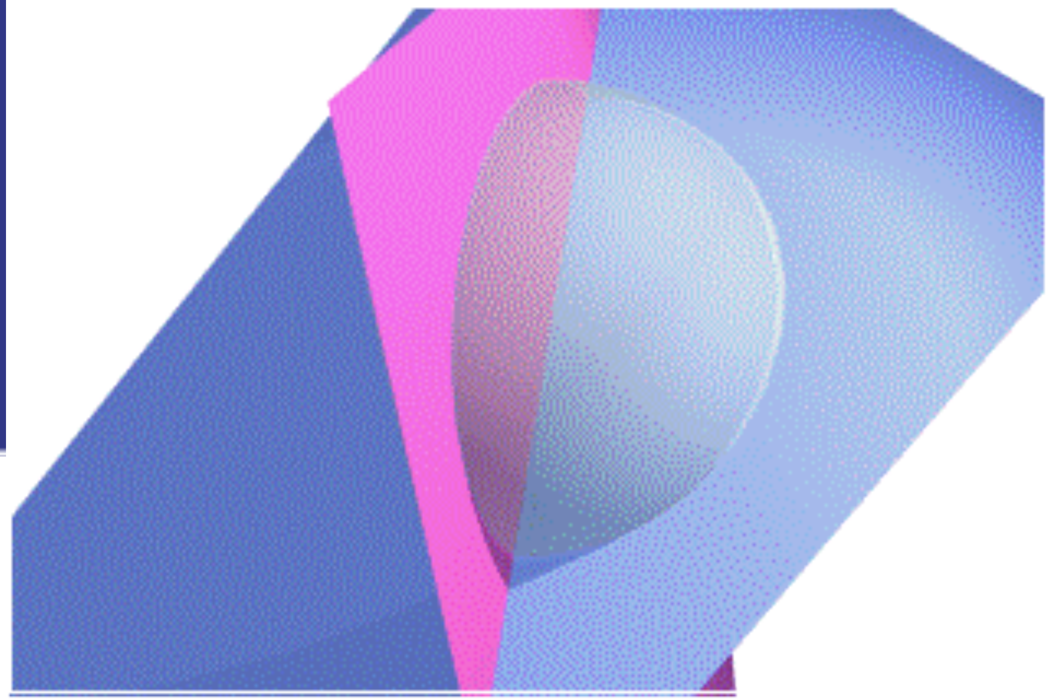
The other classifier is no longer consistent



points (x, -) in F-space are hyperplanes in V-space

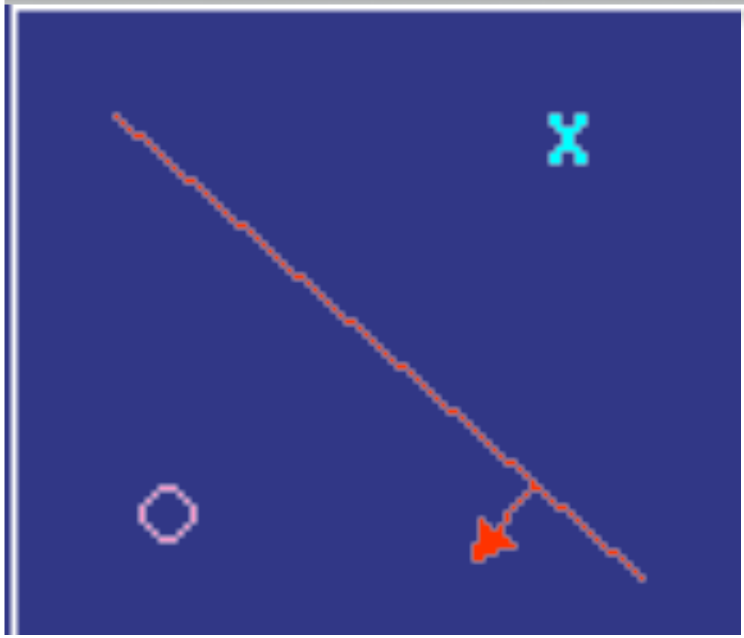
Version space for SVM

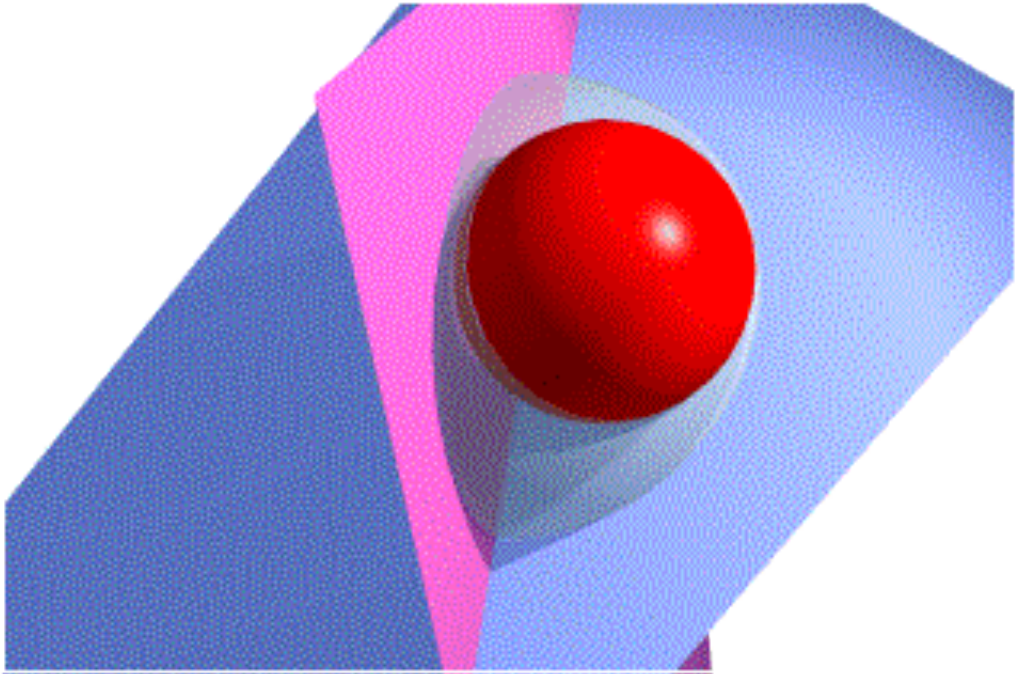
[Tong & Koller]



Version space for SVM

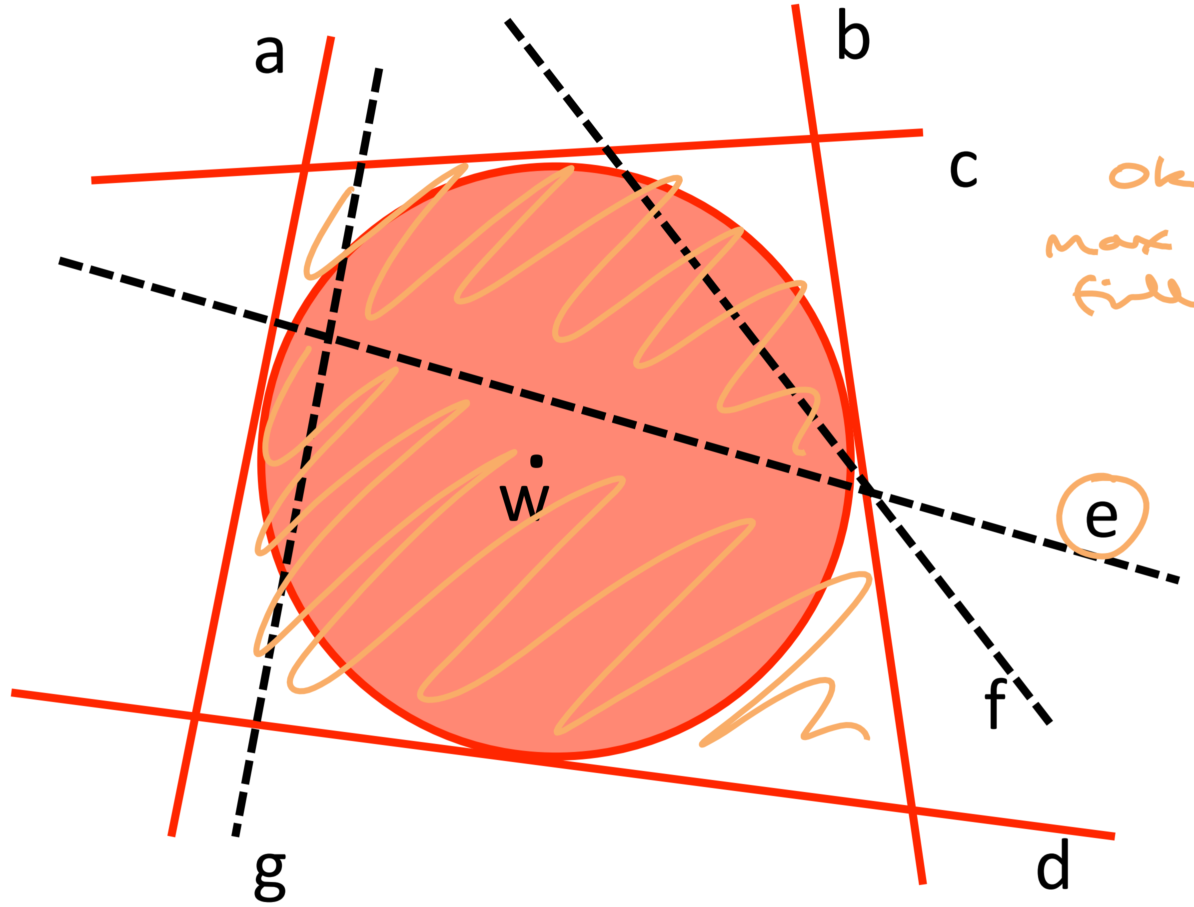
[Tong & Koller]





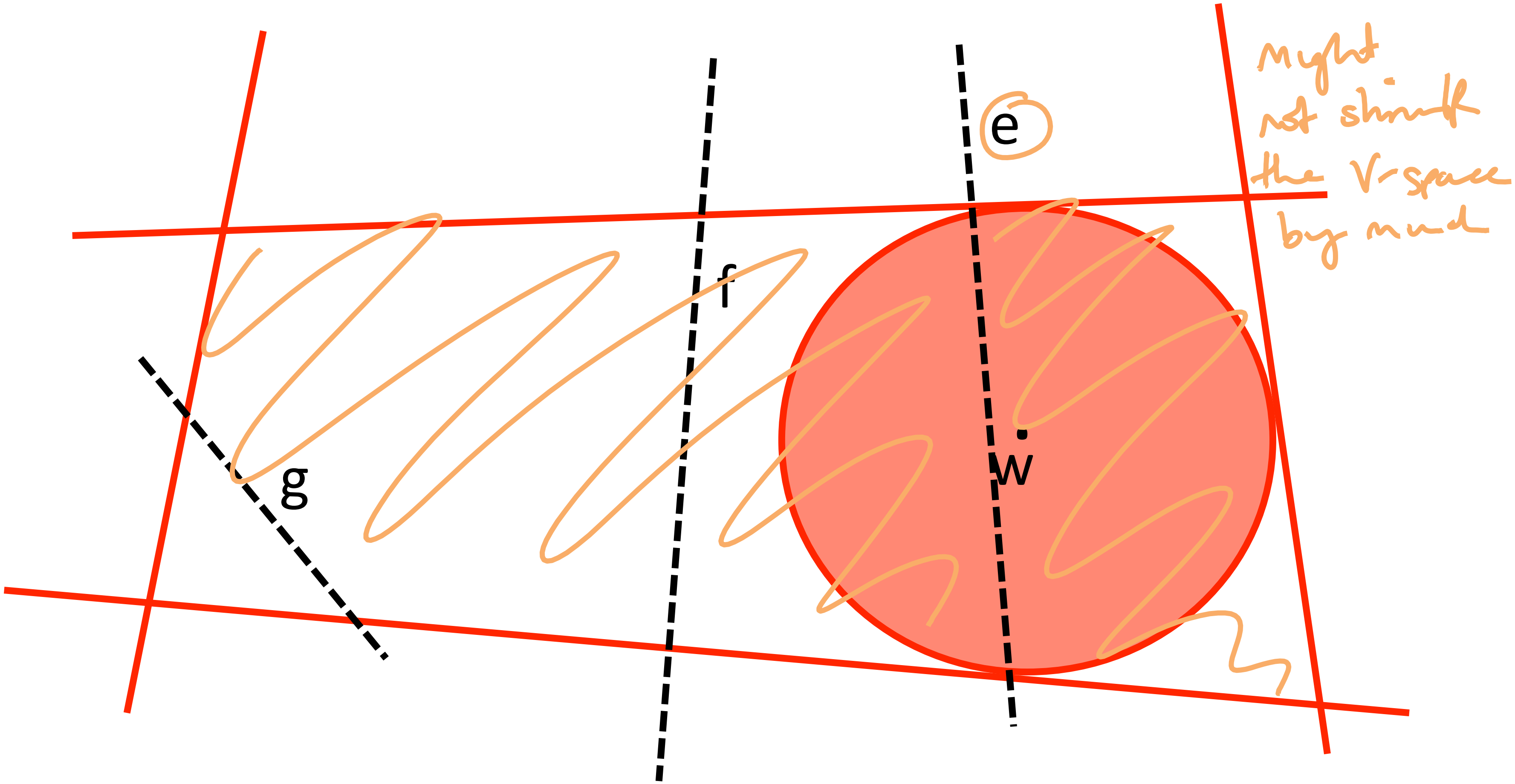
radius of sphere = margin of classifier
in V-space in F-space

Understanding uncertainty sampling



- Uncertainty sampling picks data point closest to current solution

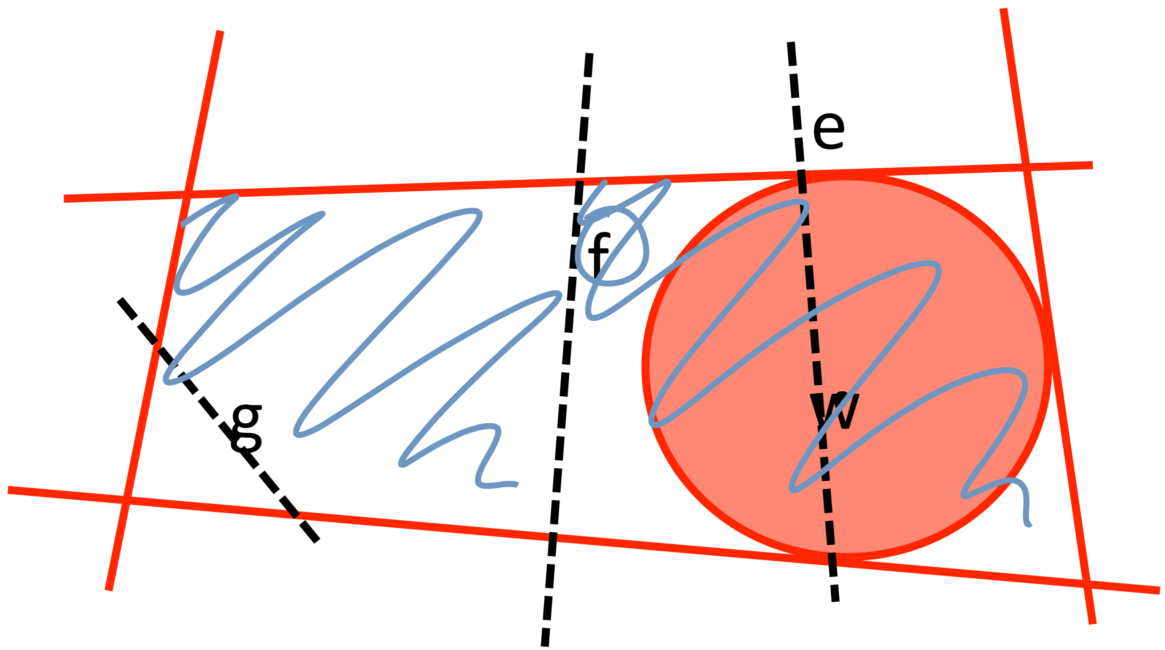
Approximation for sample selection



- Uncertainty sampling picks data point closest to current solution

Version space reduction

- *Ideally*: Wish to select example that splits the version space as equally as possible
- In general, halving may not be possible
  ➔ find "balanced" split
- How do we quantify how "balanced" a split is?



Relevant version space

- Version space for data set $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_k, y_k)\}$

$$\mathcal{V}(D) = \{\mathbf{w} : \forall(\mathbf{x}, y) \in D \ \ \text{sign}(\mathbf{w}^T\mathbf{x}) = y\}$$

Un countable

- Suppose we're also given an unlabeled pool

$$U = \{\mathbf{x}'_1, \ldots, \mathbf{x}'_n\}$$

- Relevant version space:
  Labelings of pool *consistent with the data*

$$\widehat{\mathcal{V}}(D; U) = \{h : U \to \{+1, -1\} : \exists w \in \mathcal{V}(D) \forall \mathbf{x} \in U \ \ \text{sign}(\mathbf{w}^T\mathbf{x}) = h(y)\}$$

Generalized binary search

- Start with D = {}
- While
  - For each unlabeled example x in U compute

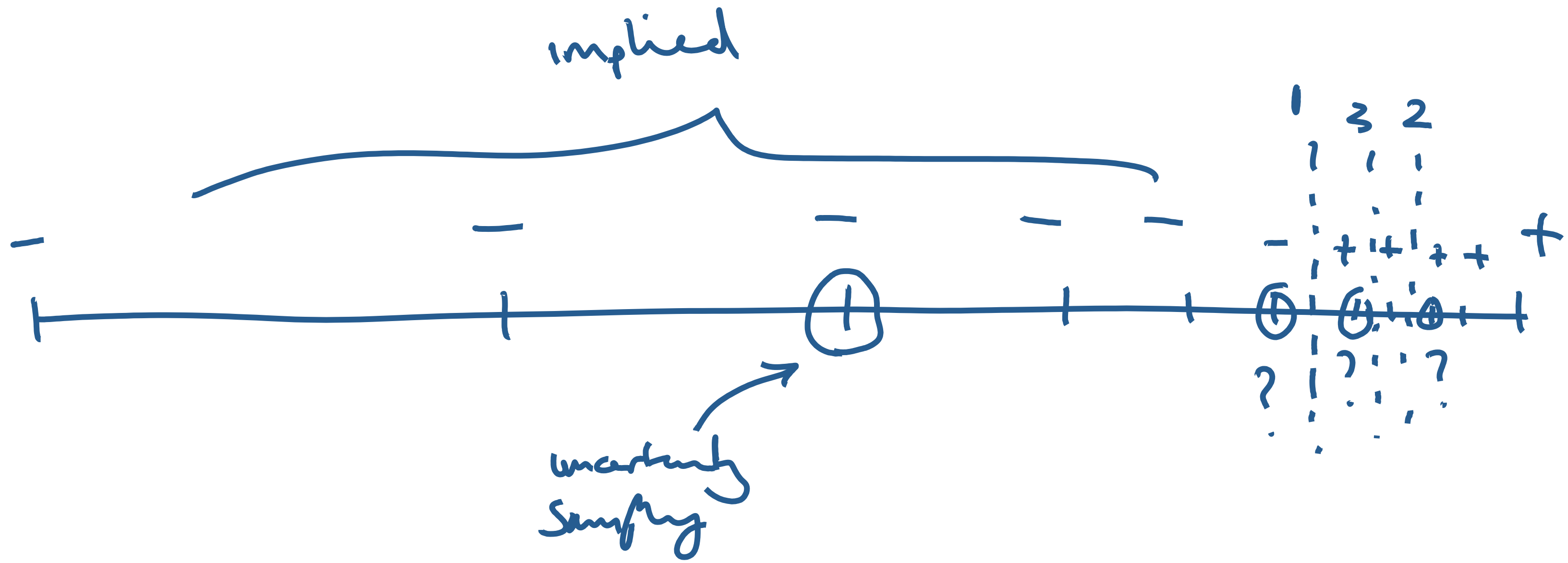
    ← hallucinate +ve

    ← hallucinate −ve

  - Pick example x where is largest, request label and add to D

Can prove that GBS requires only
more labels than any other active learning strategy,
both on average and in worst-case

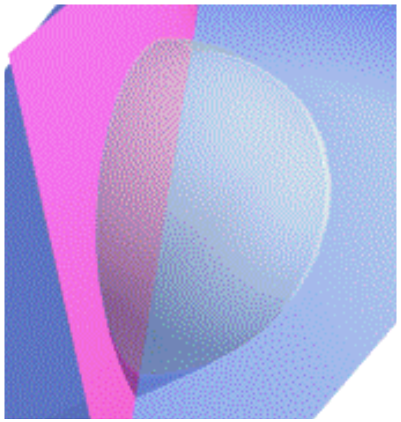
GBS for linear separators in 1D



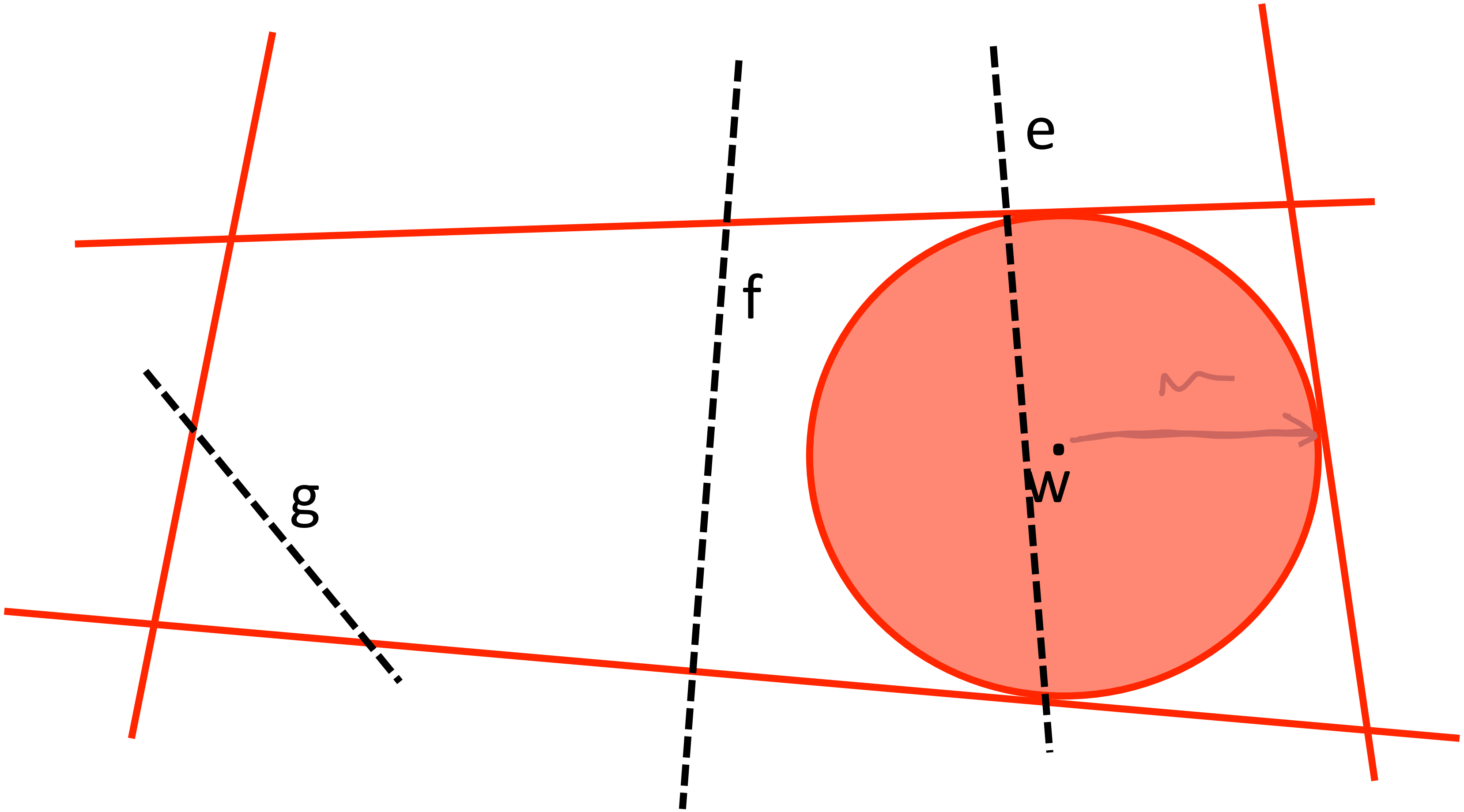
Version space reduction

- *Ideally*: Wish to select example that splits the version space as equally as possible
- In general, halving may not be possible
  ➔ find "balanced" split
  - Generalized binary search
  - Competitive with optimal active learning scheme (in the case of no noise) [c.f., Dasgupta '04]

- Size of the (relevant) version space difficult to calculate
- Need approximation!

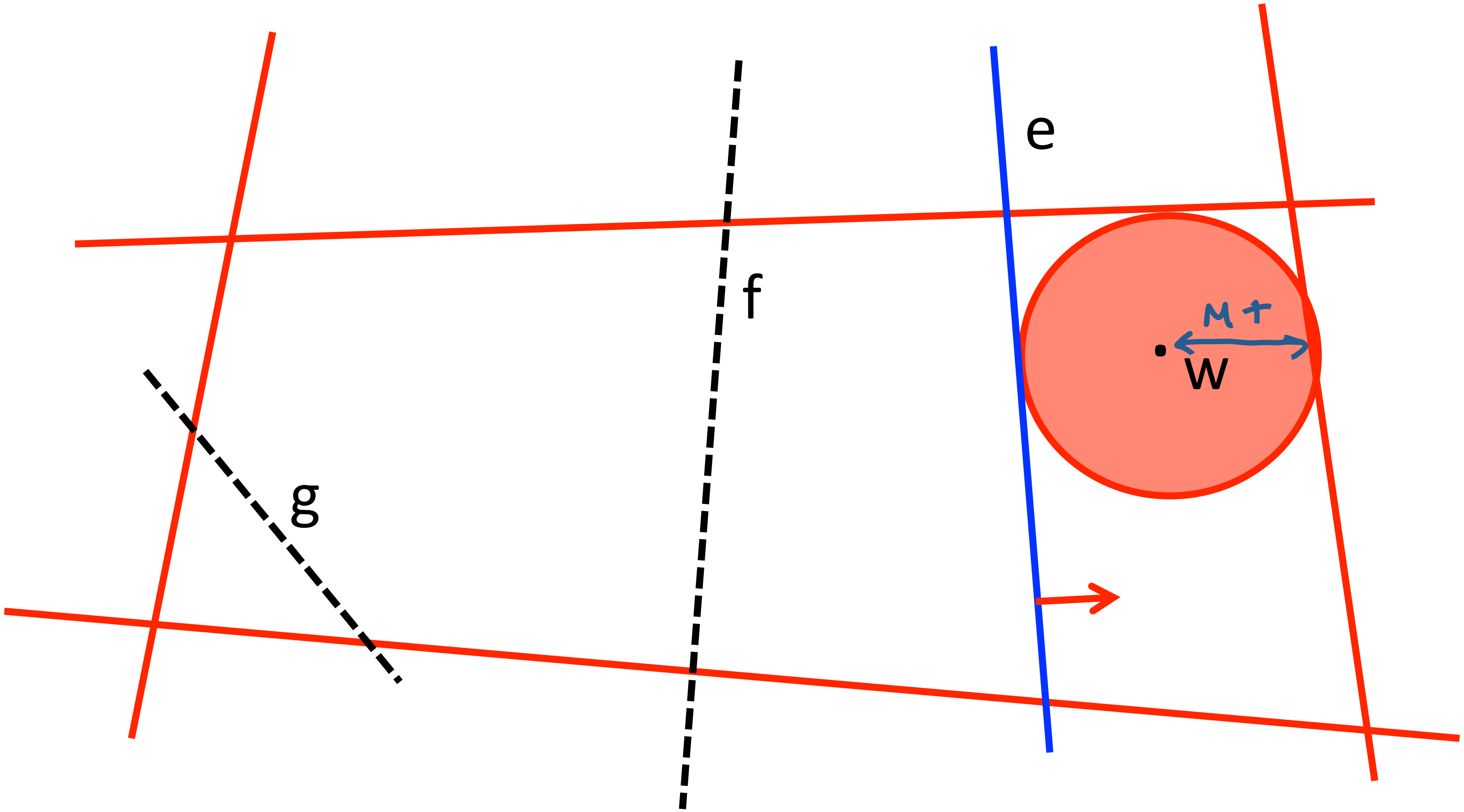


Approximation for sample selection

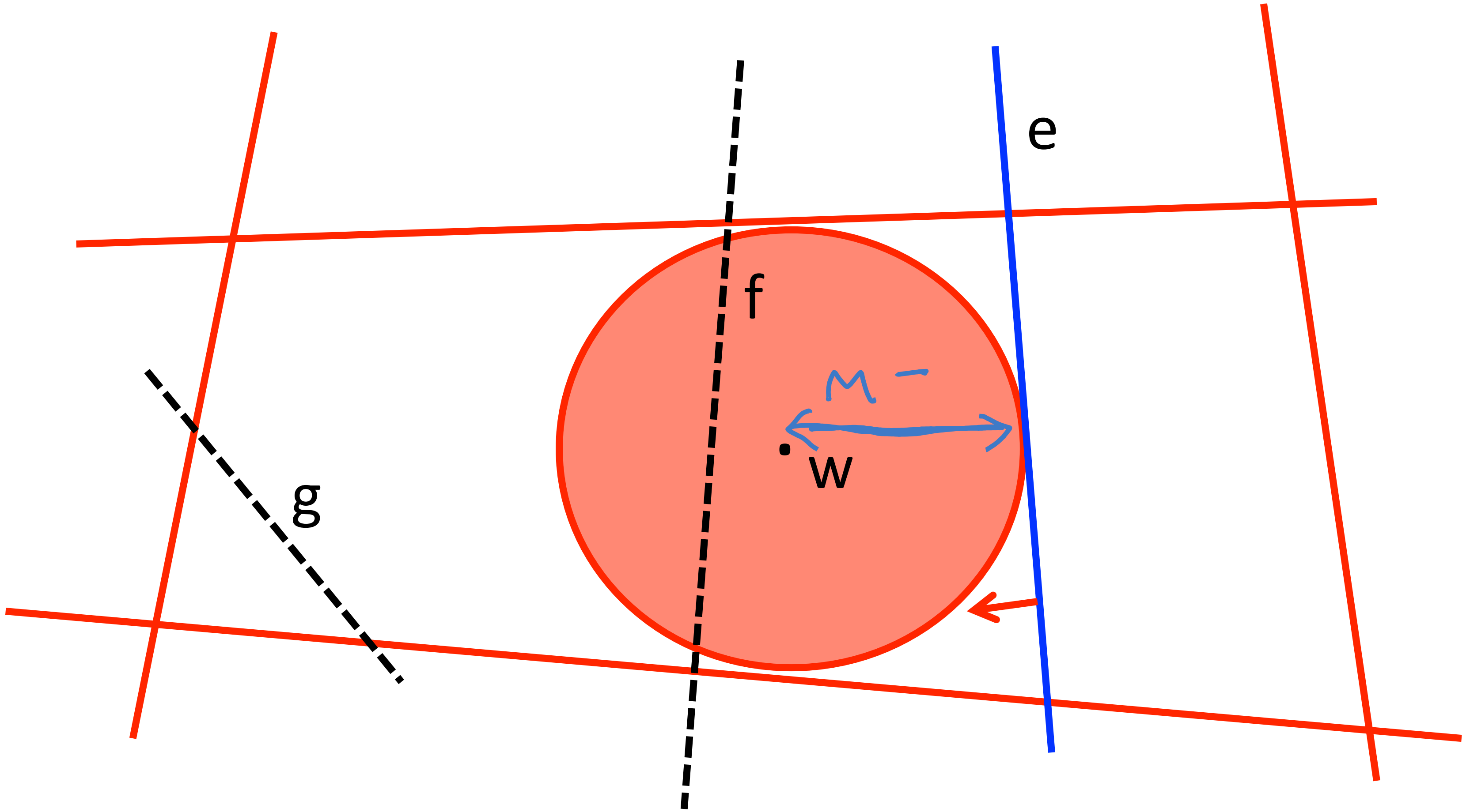


- Uncertainty sampling picks data point closest to current solution

Approximation for sample selection



Approximation for sample selection

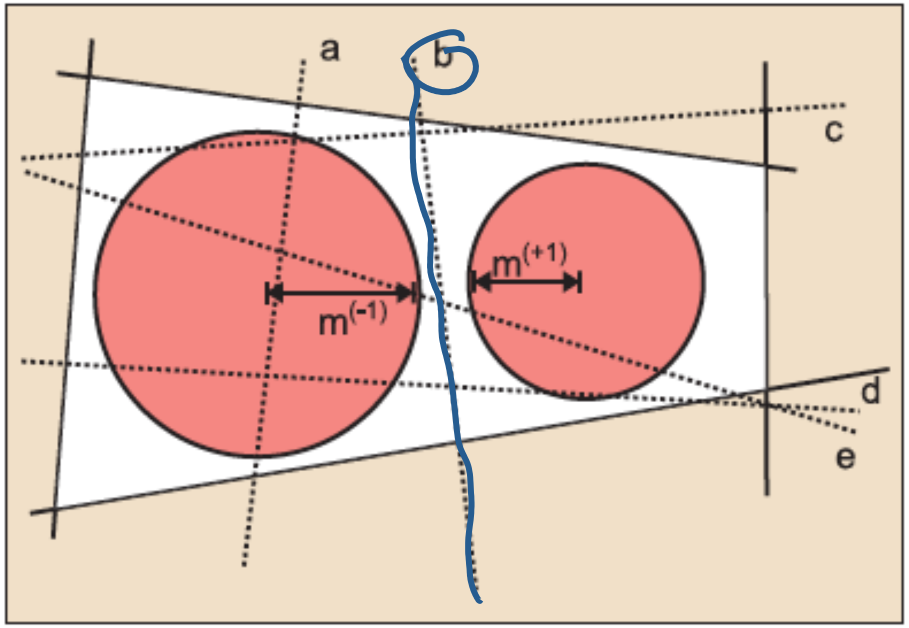


- Suggests looking at the margins of the resulting SVMs

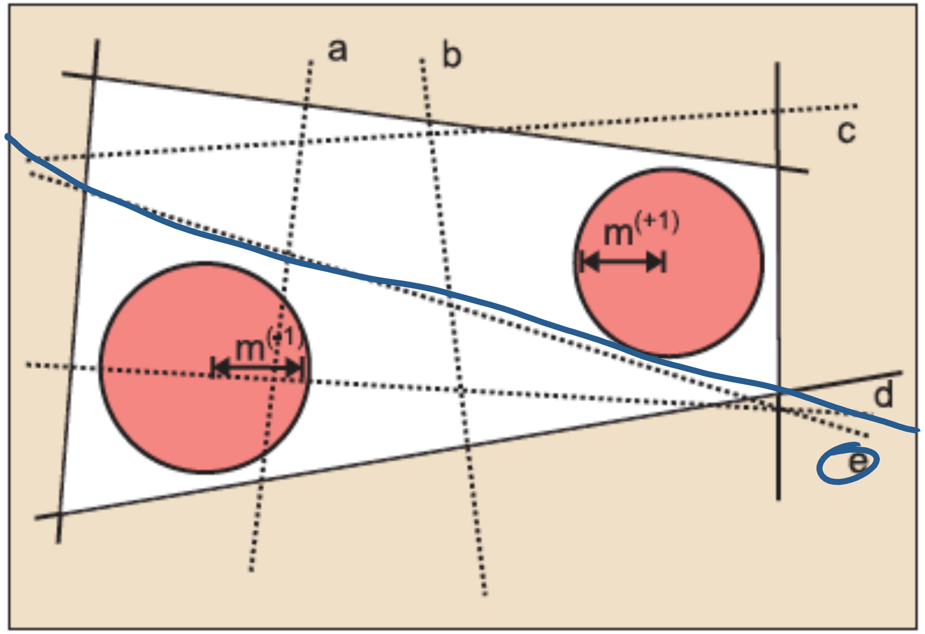
Achieving "balanced" splits

- *Key idea*: look at how labels affect resulting classifier
- Suppose we're considering data point *i*
- For each possible label {+,-} calculate resulting SVMs, with margins m^+, m^-
- Define informativeness score of *i* depending on how "balanced" the resulting margins are
  - Max-min margin: $\min(m^+, m^-)$
  - Ratio margin: $\min\left(\frac{m^+}{m^-}, \frac{m^-}{m^+}\right)$

Selecting “balanced” splits



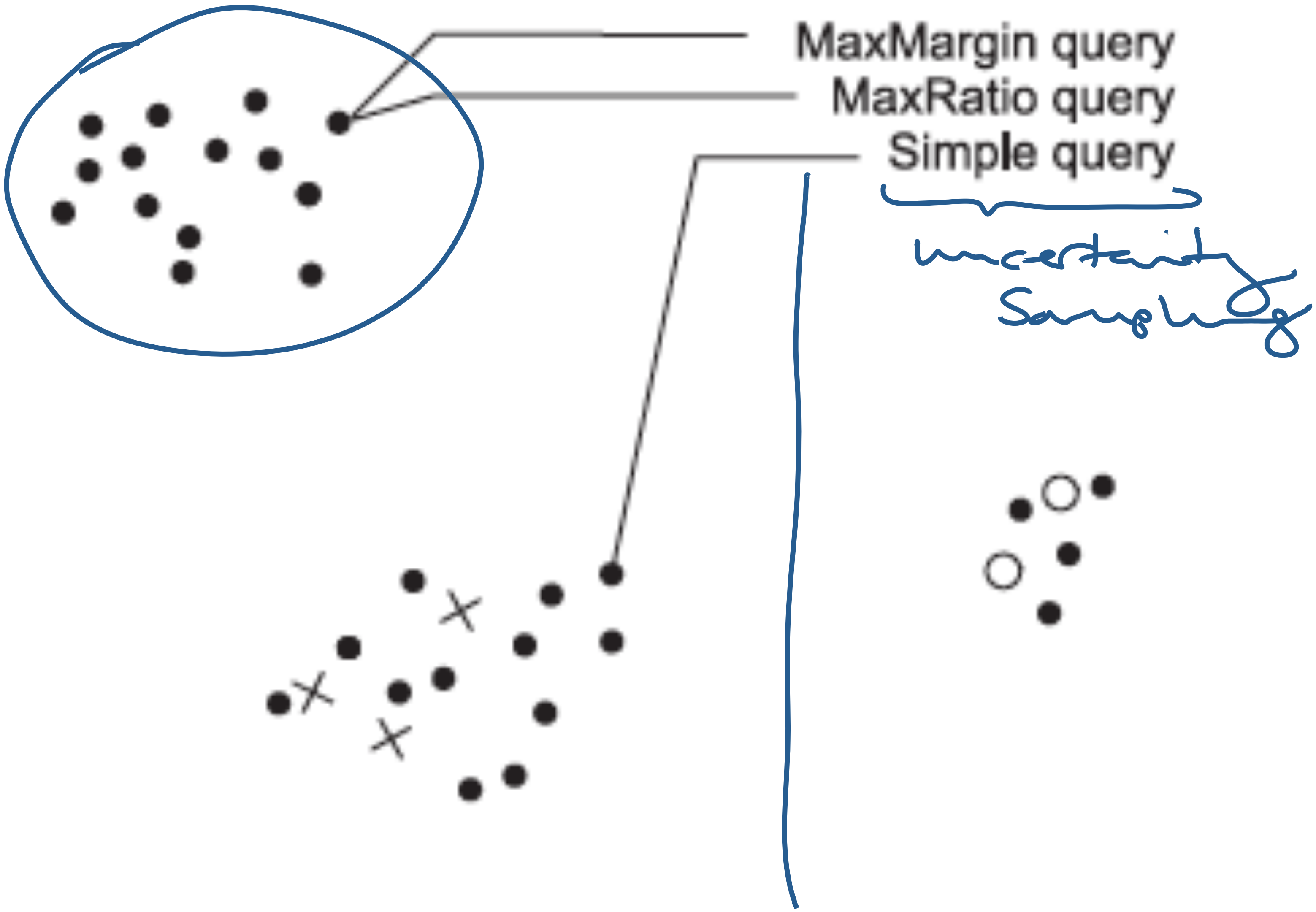
Max-min margin



Ratio margin

Selection

[Tong & Koller]

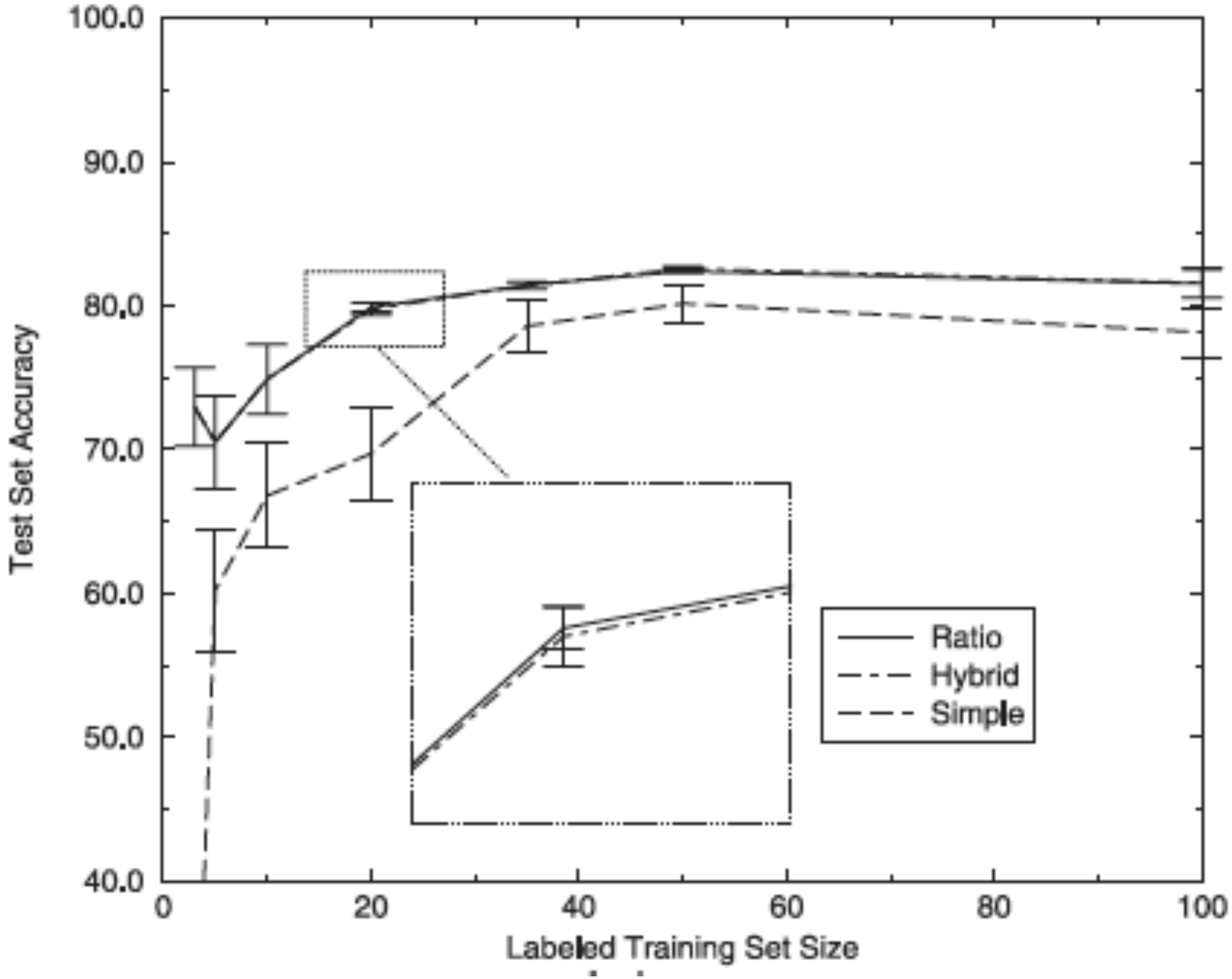


Computational challenges

- Max-min margin and ratio margin more expensive
  - Need to train an SVM for each data point, for each label!!

- Practical tricks:
  - Only score and pick from small random subsample of data
  - Only use “fancy” criterion for the first 10 examples, then switch to uncertainty sampling
  - Occasionally pick points uniformly at random

Results (text classification)



Dealing with noise

- So far, we have assumed that labels are exact
- In practice, there is always noise. How should we deal with it?
- Practice:
  - Can use same algorithms (simply use SVM with slack variables)
- Theory:
  - Analysis **much** harder
  - Modified version of generalized binary search still works if noise is i.i.d. [Novak, NIPS '09]
  - If noise is correlated need new criterion [Golovin, Krause, Ray, NIPS '10]

What you need to know

- Pool-based active learning
- Different selection strategies
  - Uncertainty sampling: Efficient, but can fail
  - Informative sampling: Expensive, but can effectively reduce version space
- Computational tricks
  - Locality sensitive hashing to speed up uncertainty sampling
  - Hybrid selection criteria