Machine Learning

CSCI 5622 Fall 2020

Prof. Claire Monteleoni

Today

- Intro. to Unsupervised Learning
  - Intro. to Clustering

- Intro. to Interactive Learning
  - Intro. to Active Learning

With credit to S. Dasgupta

Unsupervised learning

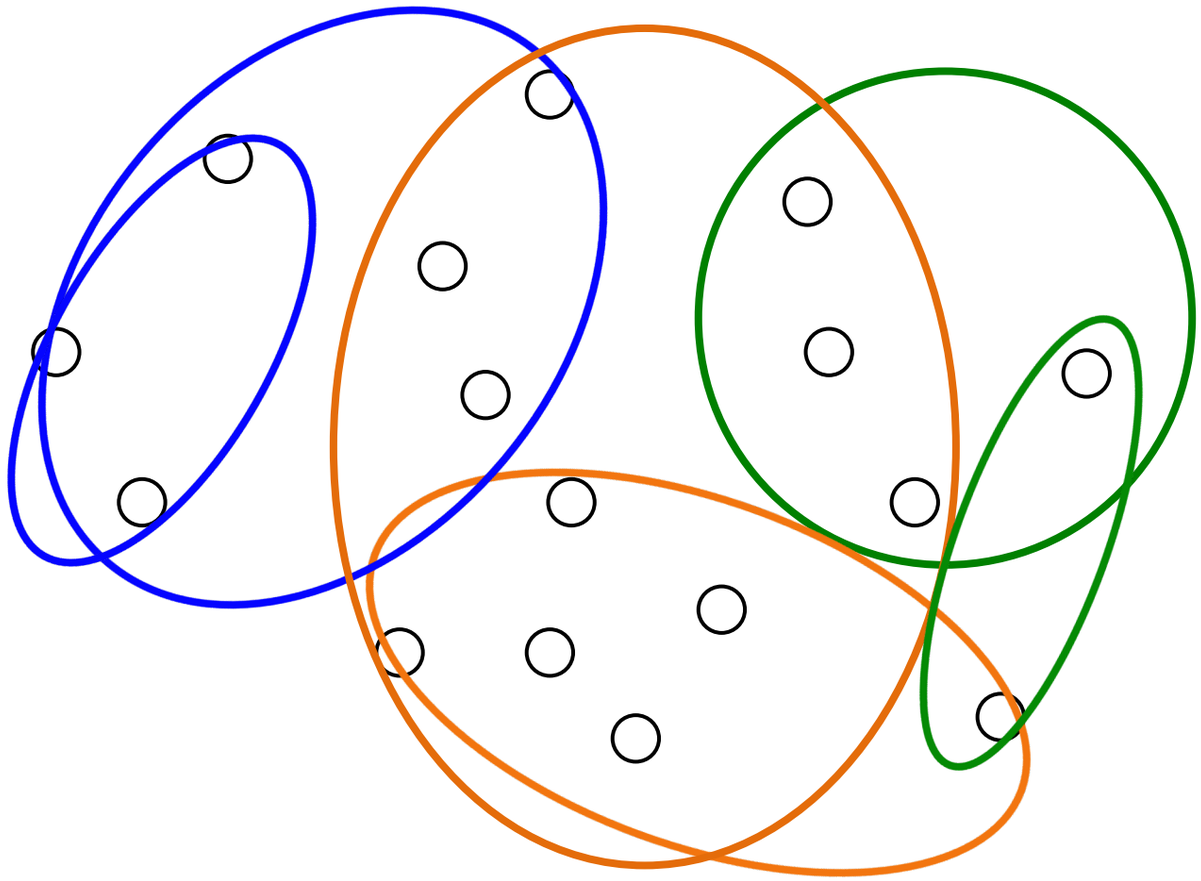
There are 4 Unsupervised Learning tasks:

1. Clustering
2. Embedding
3. Learning a generative model
4. Completion (estimating missing values), Approximation

Learning from raw data

What can be done without any labels?

Unsupervised learning



E.g. Clustering

Intro to clustering

Why use clustering?

- Preprocessing in a supervised learning pipeline
  - Quantization or discretization of a continuous data space
  - Summarization of big data
  - Cluster data points and then apply supervised learning separately to each cluster
  - Cluster the features, as a form of embedding

- Exploratory data analysis
  - Discover “groups” in the data (e.g. communities in social networks)
  - Discover hierarchical or nested structure

Clustering is a huge field

- Probabilistic methods
  - Mixture models
    - E.g. EM, and line of results depending on variance, etc.
  - Topic models
- Spectral clustering methods
- Clustering in metric spaces
- Hierarchical clustering
- Among others, …

k-means clustering objective

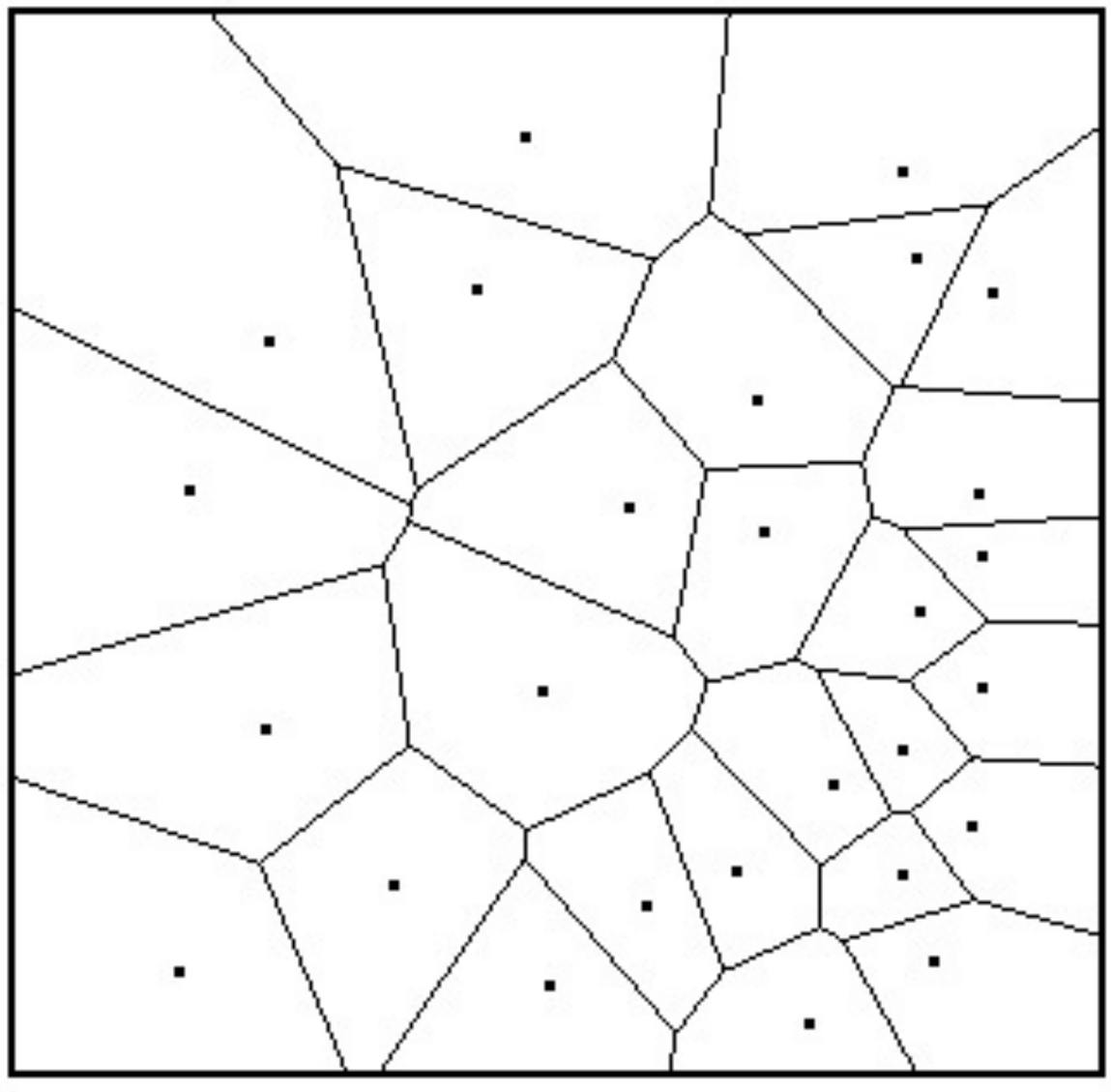
Clustering algorithms can be hard to evaluate without prior information or assumptions on the data.

With *no* assumptions on the data, one evaluation technique is w.r.t. some objective function.

A widely-cited and studied objective is the k-means clustering objective: Given set, $X \subset R^d$, choose $C \subset R^d$, $|C| = k$, to minimize:

$$\phi_C = \sum_{x \in X} \min_{c \in C} \|x - c\|^2$$

Voronoi regions



k-means clustering objective

A widely-cited and studied objective is the k-means clustering objective: Given set, $X \subset R^d$, choose $C \subset R^d$, $|C| = k$, to minimize:

$$\phi_C = \sum_{x \in X} \min_{c \in C} \|x - c\|^2$$

Unfortunately this in an NP-hard optimization problem, even when d=2!

Even the algorithm called “the k-means clustering algorithm” (a.k.a. Lloyd’s algorithm) does not have an approximation guarantee!

*without assumptions on the data

Algorithm: k-means++:

Choose first center c_1 uniformly at random from X,
and let $C = \{c_1\}$.
Repeat (k-1) times:
Choose next center $c_i = x' \in X$ with prob.

$$\frac{D(x', C)^2}{\sum_{x \in \mathcal{X}} D(x, C)^2}$$

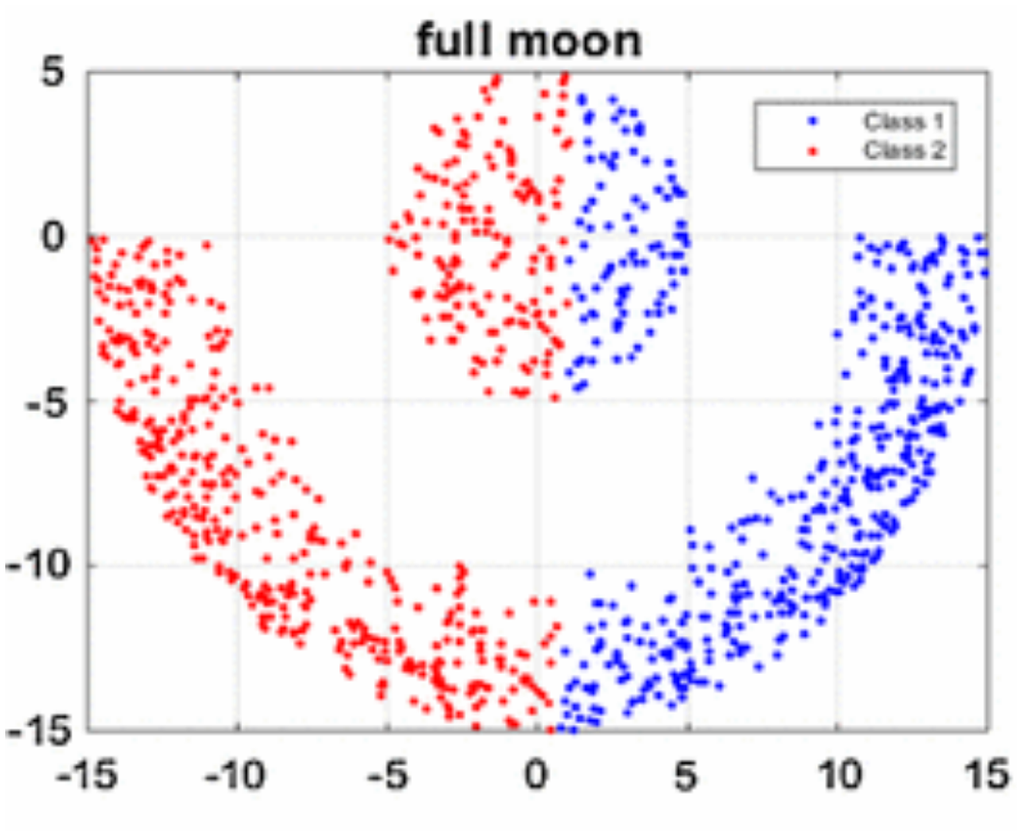
$C \leftarrow C \cup \{c_i\}$

where:

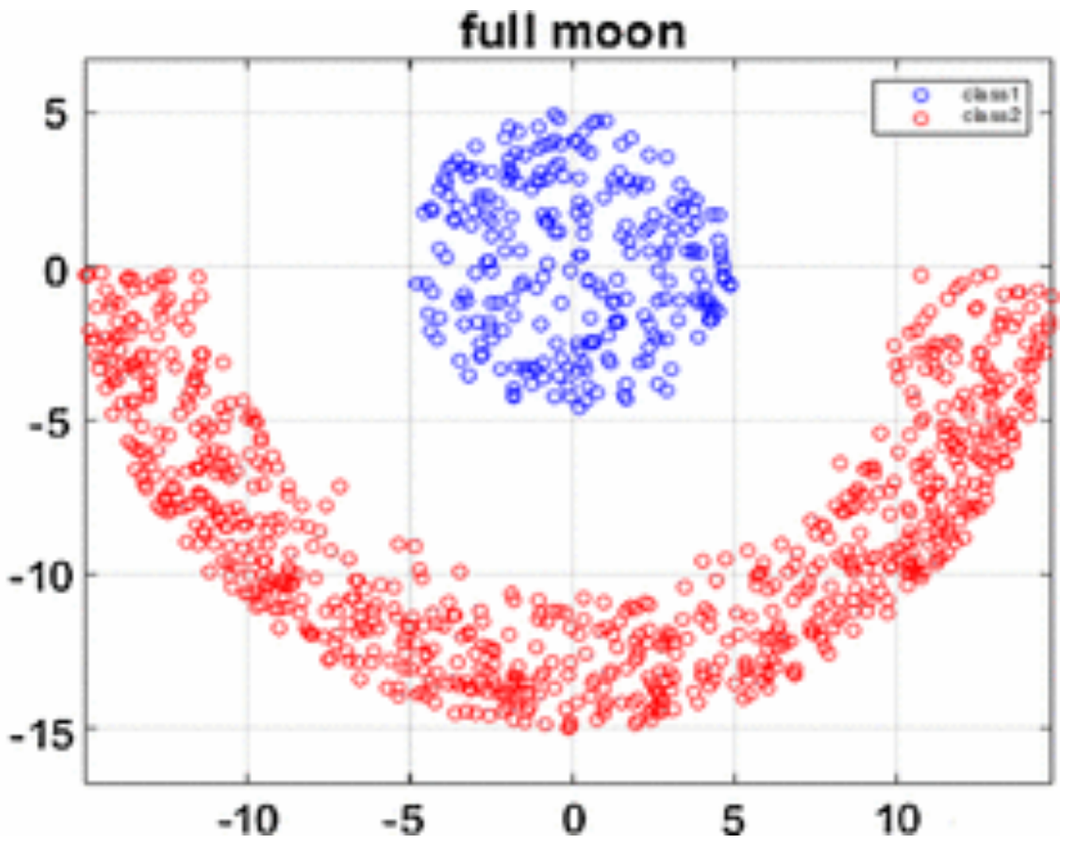
$$D(x, C) = \min_{c \in C} \|x - c\|$$

Theorem (Arthur & Vassilvitskii ’07):
Returns an O(log *k*)- approximation to the *k*-means objective,
in expectation.

Kernel clustering



(Vanilla) *k*-means



Kernel *k*-means

Image credit: M. Lincoln

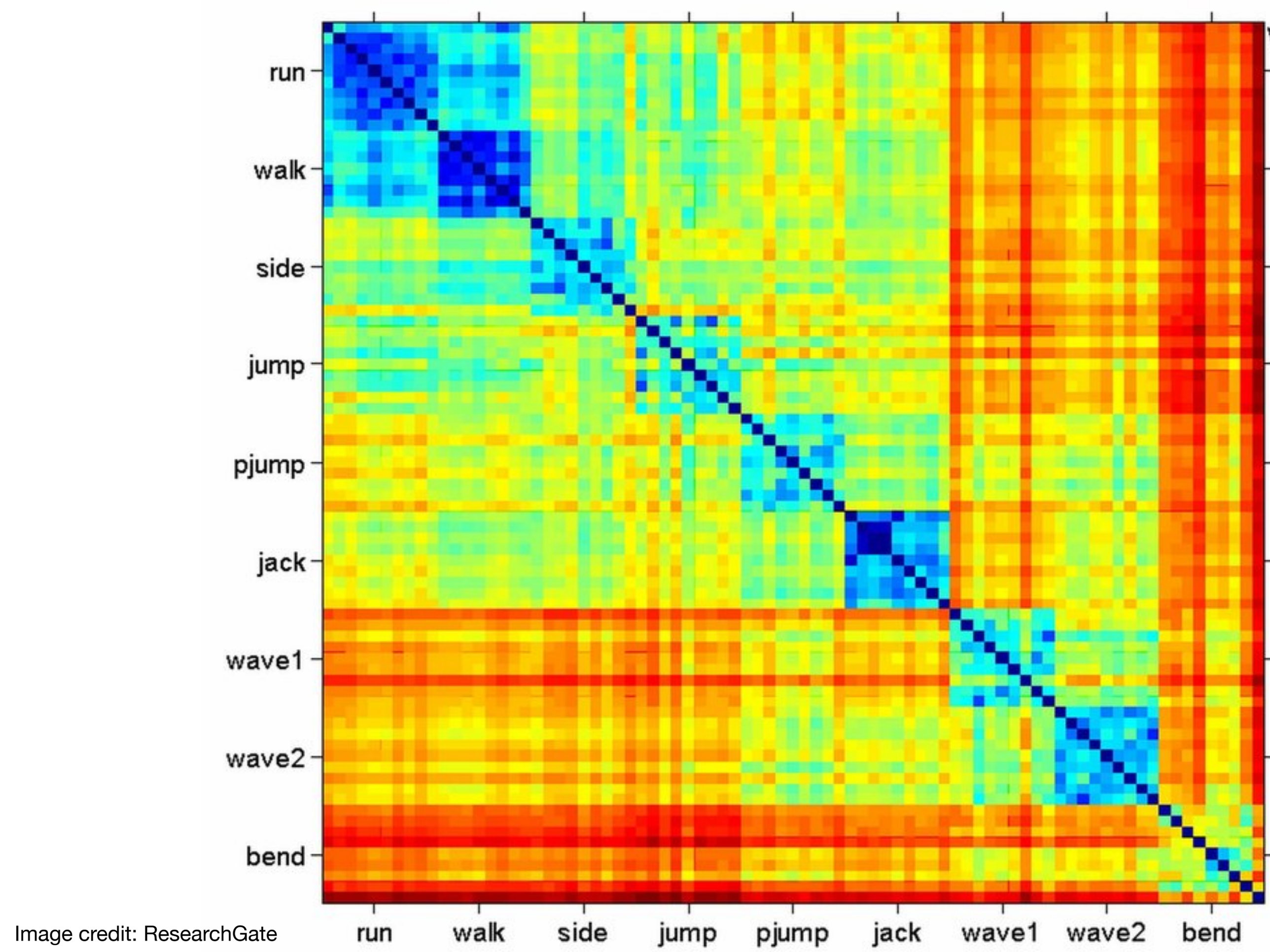


Image credit: ResearchGate

Spectral Clustering

- Algorithm template:
  - Compute Affinity matrix, A
    - Compute Graph Laplacian, L, from A
  - Perform *k*-PCA embedding on L
  - Perform clustering on embedded data
  - This gives you a partition of the points, return this as the clustering

Spectral clustering

[Ng, Jordan, Weiss, '02]

Input: data set $S = \{s_1, s_2 \ldots, s_n\} \in \mathbb{R}^d$,

number of clusters *k*, kernel function $\kappa : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$

Compute:

Affinity matrix: $A \in \mathbb{R}^{n \times n}$ s.t. $A_{ij} = \delta[i \neq j] \kappa(s_i, s_j)$

$D \in \mathbb{R}^{n \times n}$ s.t. $D_{ij} = \delta[i = j] \sum_{j=1}^{n} A_{ij}$

Graph Laplacian: $L = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$

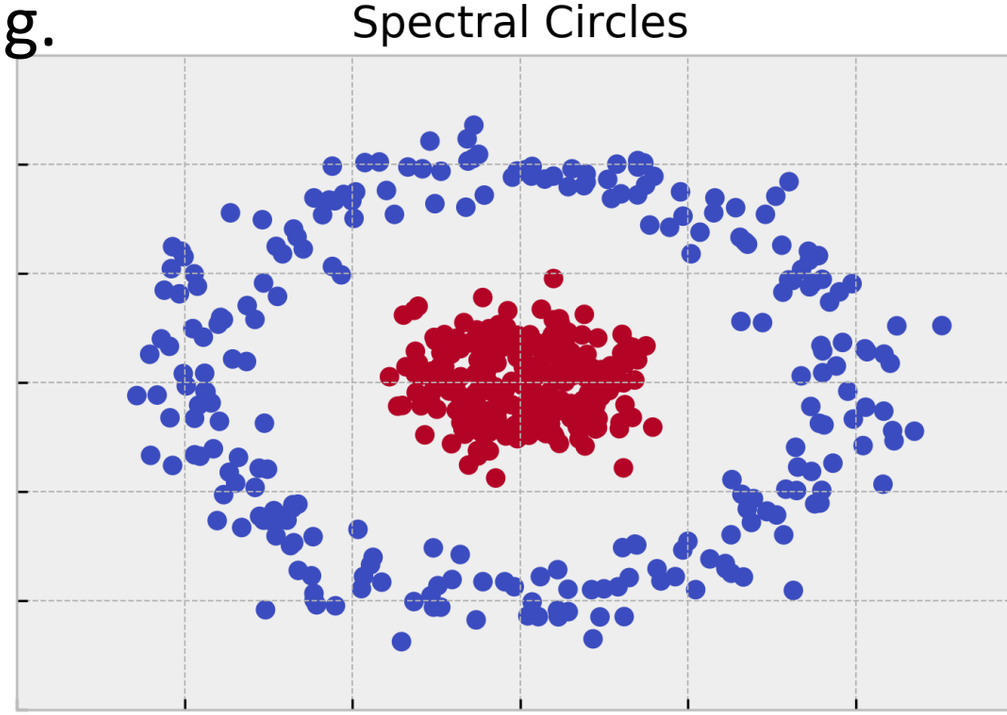
Top *k* eigenvectors of *L* := $X \in \mathbb{R}^{n \times k}$

Normalize rows of X; Cluster rows of X; Return clusters, {C_1, …, C_k}

Cluster assignment of data point *i* is cluster assignment of row X_i.

Spectral Clustering

- Pros
  - Any data can be represented as a graph
  - Gracefully handles large d, via kernel function.
  - Can find interesting cluster structure, e.g.



- Con
  - Spectral decomposition of n x n matrix:

    → Running time and memory are Polynomial in n.

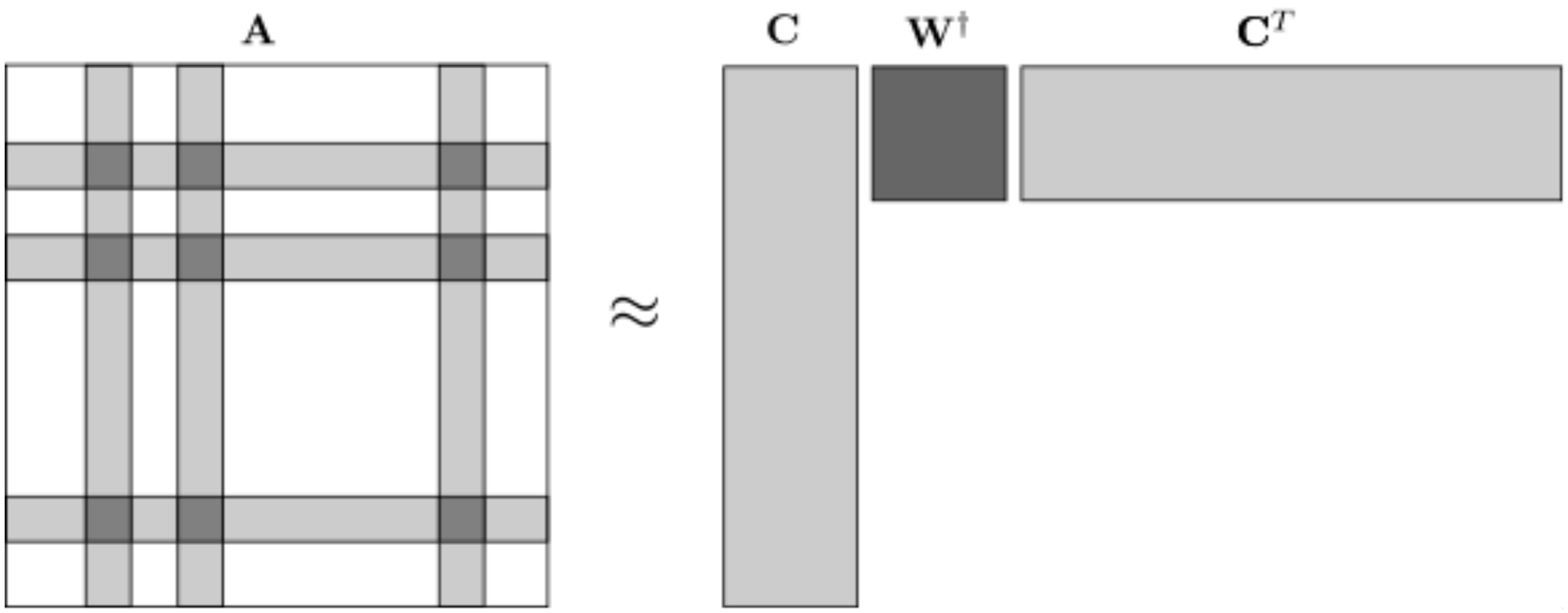
Scaling up spectral clustering to big data

- Spectral clustering exhibits performance advantages in a variety of applications
  - Graph data
  - You can represent any data with affinity matrix, given a similarity function (e.g. a Kernel).

- However maintaining and computing on the affinity matrix becomes computationally prohibitive as *n* grows

- How can we create light-weight approximations to the affinity matrix?

Approximating the affinity matrix



Credit: Gittens ‘13.

SPSD Sketching Model. Let $\mathbf{A}$ be an $n \times n$ positive semi-definite matrix, and let $\mathbf{S}$ be a matrix of size $n \times \ell$, where $\ell \ll n$. Take $\mathbf{C} = \mathbf{AS}$ and $\mathbf{W} = \mathbf{S}^T\mathbf{AS}$. Then $\mathbf{CW}^\dagger\mathbf{C}^T$ is a low-rank approximation to $\mathbf{A}$ with rank at most ℓ.

[Gittens & Mahoney ‘13]

Clustering in metric spaces

Definition 1. A metric space $(\mathcal{X}, \rho)$ consists of a set $\mathcal{X}$ and a distance function $\rho : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that satisfies the three properties of a metric:

1. Reflexivity: $\rho(x, y) \geq 0$ with equality iff $x = y$
2. Symmetry: $\rho(x, y) = \rho(y, x)$
3. Triangle inequality: $\rho(x, z) \leq \rho(x, y) + \rho(y, z)$

k-center clustering objective

k-CENTER CLUSTERING

Input: Finite set $S \subset \mathcal{X}$; integer k.

Output: $T \subset \mathcal{X}$ with $|T| = k$.

Goal: Minimize $\text{cost}(T) = \max_{x \in S} \rho(x, T)$.

Algorithm: Farthest-first retrieval

pick any $z \in S$ and set $T = \{z\}$
while $|T| < k$:
    $z = \arg\max_{x \in S} \rho(x, T)$
    $T = T \cup \{z\}$

Hierarchical clustering

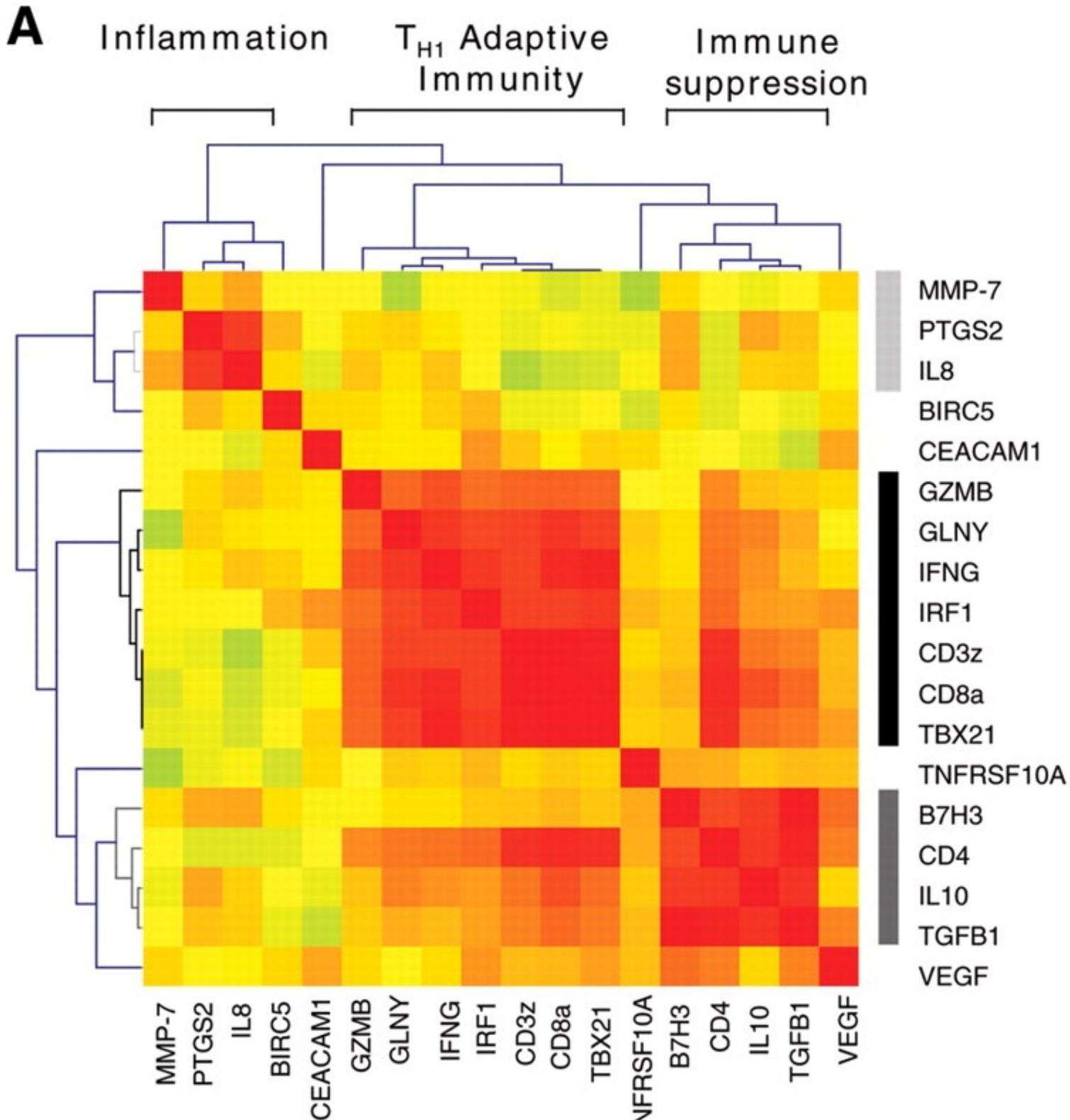
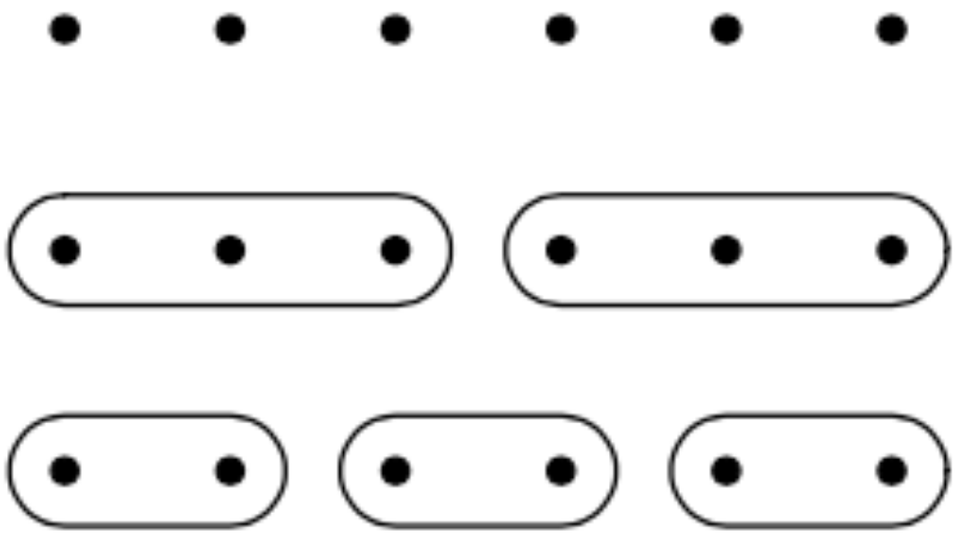


Figure credit: Stack Exchange

Hierarchical clustering

It is not always possible to construct a hierarchical clustering whose induced k-clustering is optimal for all k.

E.g.



So instead, the goal is to provide an algorithms that approximate the optimum, for all k.

Figure credit: S. Dasgupta

Hierarchical Agglomerative Clustering (HAC)

Input: a set of *n* points in some metric space $(\mathcal{X}, \rho)$

Algorithm:

Initialize *n* clusters, each a single data point

While the total number of clusters is > 1:

    Merge the two "closest" clusters into one cluster

Semi-supervised learning

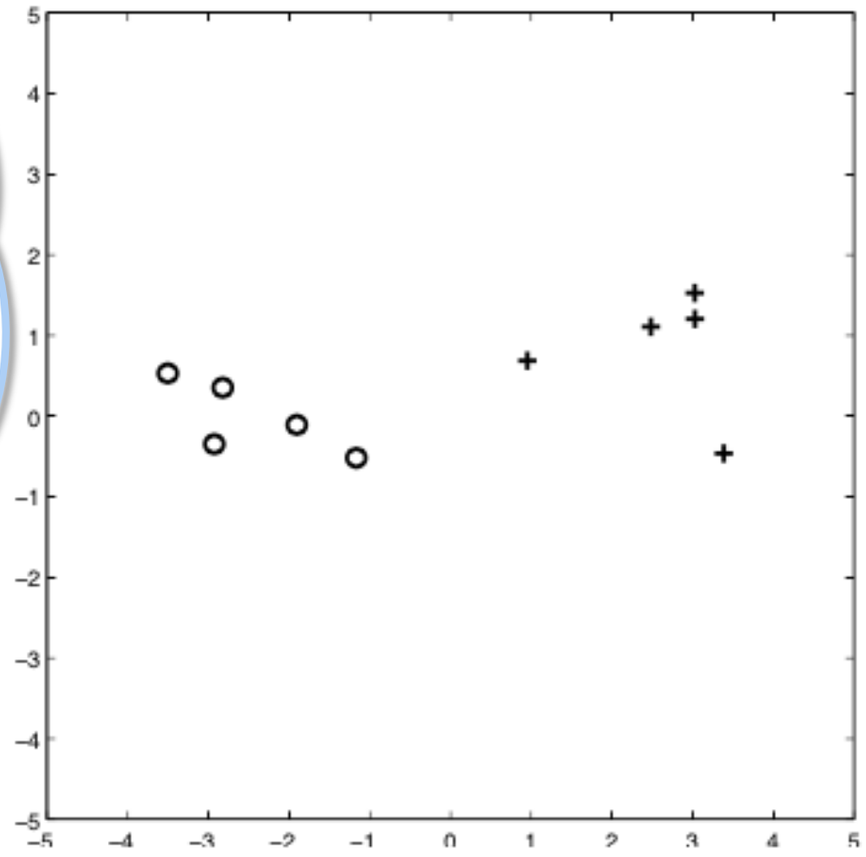
- Semi-supervised learning is an umbrella term for problems in which some of the data is labeled and some of the data is unlabeled.

- Typically this is determined in advance, and algorithms do not get to interact with the label-generating process.

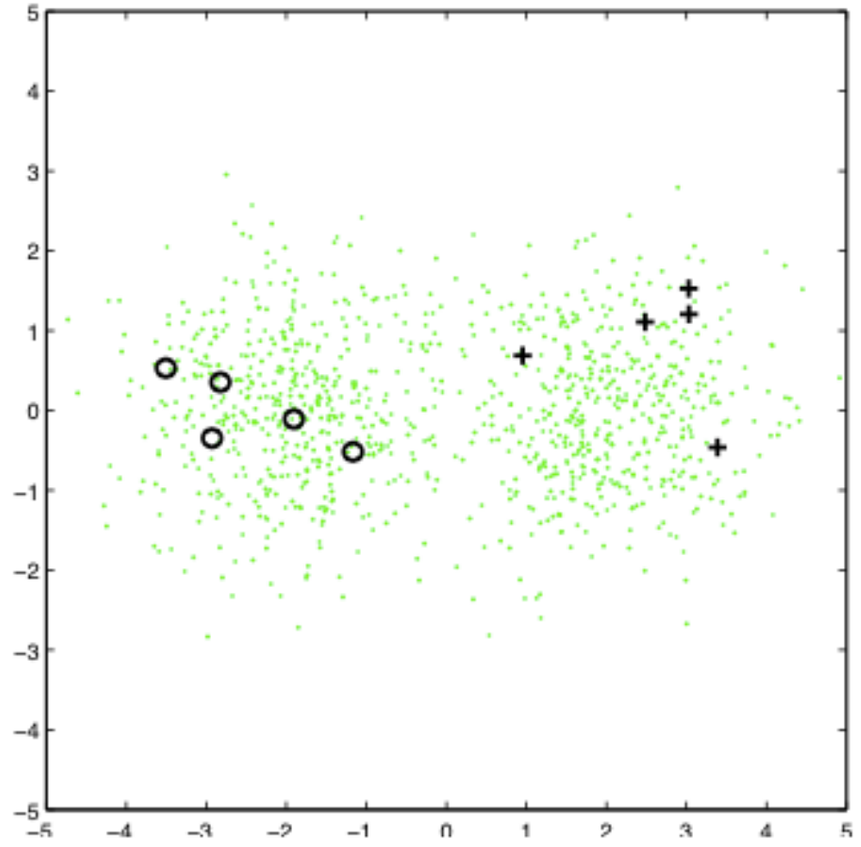
- Semi-supervised learning can be studied in the standard setting, in which we need to learn a classifier/regressor to generalize to future unseen data.

- OR, the goal could be transduction: only need to output labels on the unlabeled points in the input set.

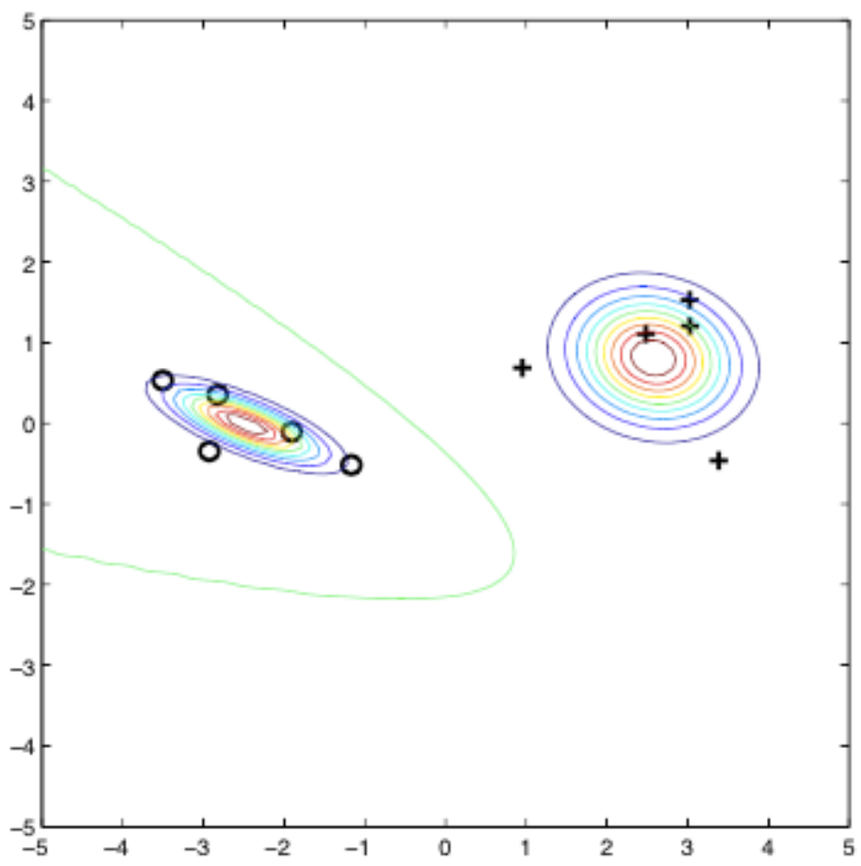
How can unlabeled data help?



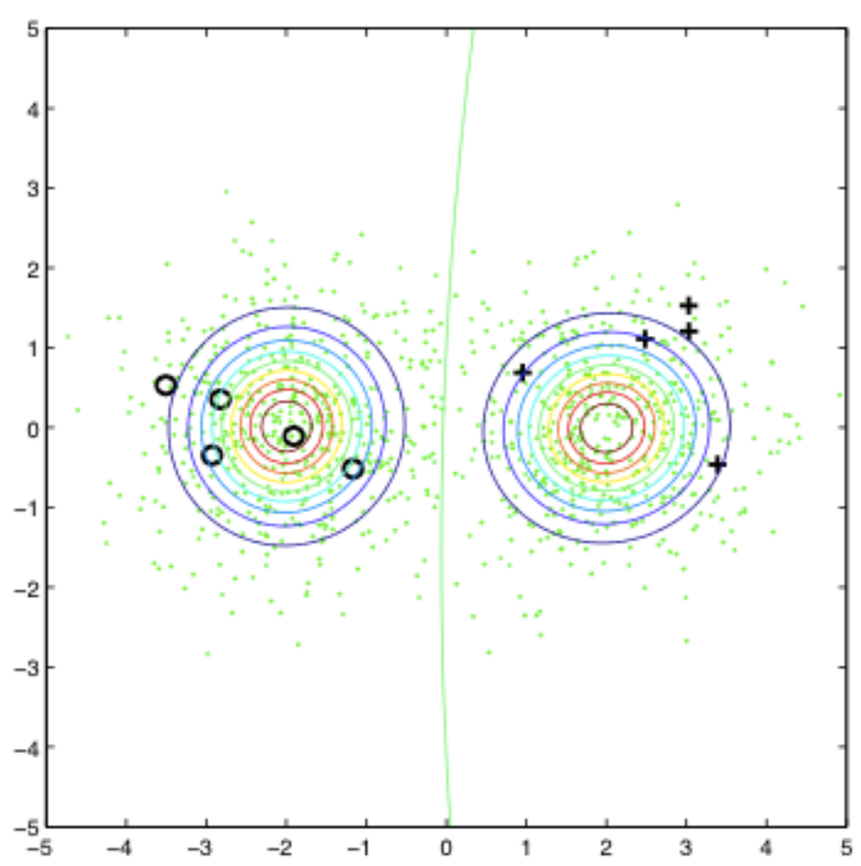
(a) labeled data



(b) labeled and unlabeled data (small dots)



(c) model learned from labeled data



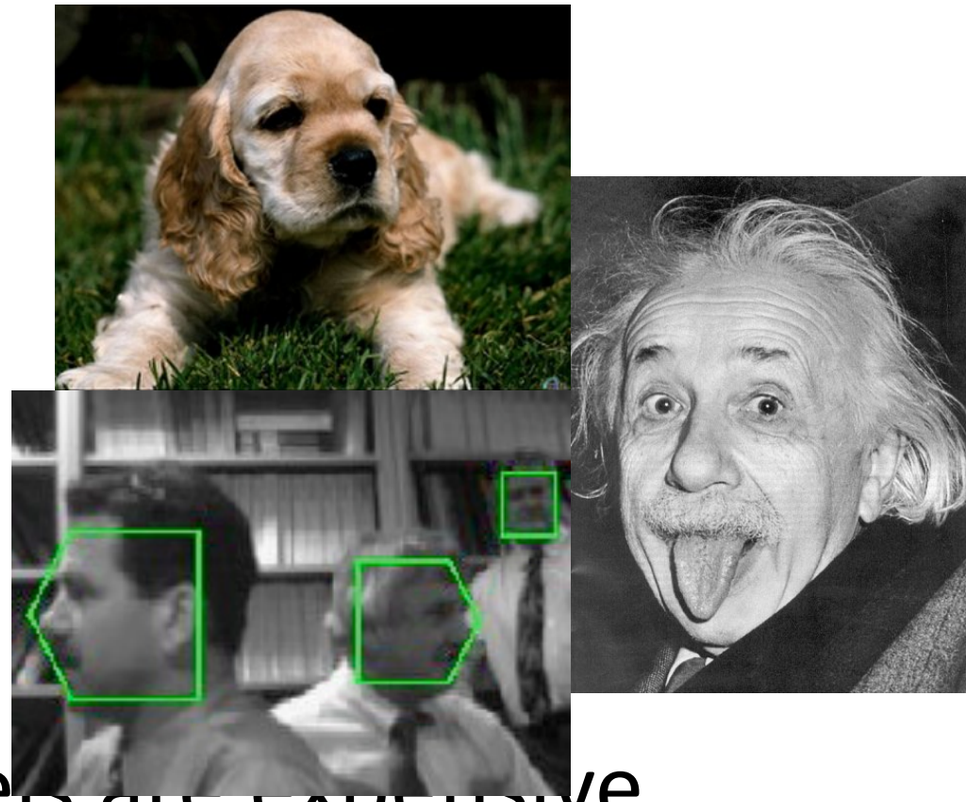
(d) model learned from labeled and unlabeled data

Credit: X. Zhu ‘08

Active Learning

Many data-rich applications:

- Image/document classification
- Object detection/classification in video
- Speech recognition
- Analysis of sensor data



Unlabeled data is abundant, but labels are expensive.

Active Learning model: learner can pay for labels.

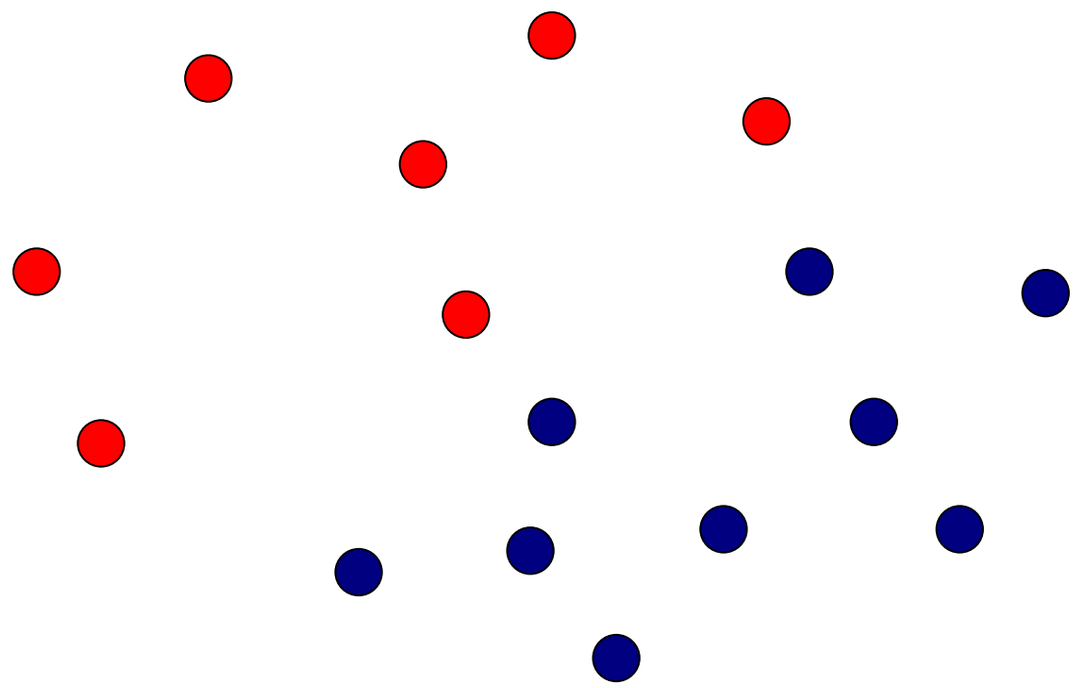
Allows for intelligent choices of which examples to label.

Goal: given stream (or pool) of unlabeled data, use fewer labels to learn (to a fixed accuracy) than via supervised learning.

General field: Interactive Learning: learner interacts with teacher

Supervised learning

Given access to labeled data (drawn iid from an unknown underlying distribution P), want to learn a classifier chosen from hypothesis class H, with misclassification rate < ε.



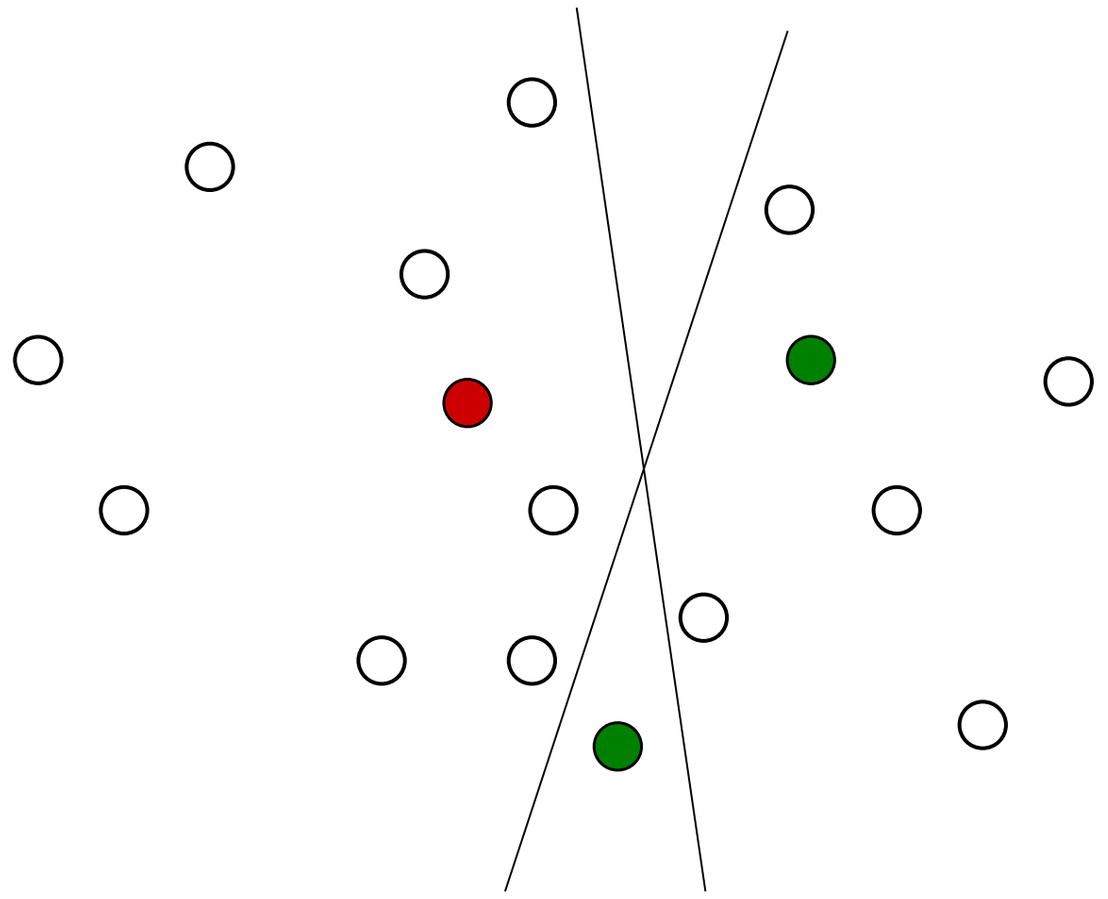
Sample complexity characterized by d = VC dimension of H.

If data is *separable*, need roughly d/ε labeled samples (PAC sample complexity).

Slide credit: Sanjoy Dasgupta

Active Learning

Given unlabeled data, choose which labels to buy, to attain a good classifier, at a low cost (in labels).



Label-complexity: What is the minimum number of labels needed to achieve the target error rate?

Active learning variants

There are several models of active learning:

- Query learning (a.k.a. Membership queries)
- Pool-based AL
- Active model selection
- Experiment design

Various evaluation frameworks:

- Regret minimization
- Minimize label-complexity to reach fixed error rate
- Label-efficiency (fixed label budget)

Membership queries

Early model of active learning in theory work [Angluin 1992]

X = space of possible inputs, e.g. R^n

H = class of hypotheses

Target concept h^* in H to be identified *exactly*.

You can ask for the label of any point in X: *no unlabeled data*.

$H_0 = H$

For t = 1,2,…

pick a point x in X and query its label $h^*(x)$

let H_t = all hypotheses in H_{t-1} consistent with $(x, h^*(x))$

What is the minimum number of "membership queries" needed to reduce H to just $\{h^*\}$?

Slide credit: S. Dasgupta

Membership queries: problem

Many results in this framework, even for complicated hypothesis classes.

Problem: informative synthetic queries can be hard to label!

[Baum and Lang, 1991] tried fitting a neural net to handwritten characters.

Synthetic instances created were incomprehensible to humans!

[Lewis and Gale, 1992] tried training text classifiers.

"an artificial text created by a learning algorithm is unlikely to be a legitimate natural language expression, and probably would be uninterpretable by a human teacher."

Slide credit: S. Dasgupta

Pool-based active learning

Framework due to [Cohn, Atlas, Ladner, et al. NIPS '89]

Assume a fixed probability distribution, D over X x Y, X some input space, $Y = \{+1, -1\}$.

Given: stream (or pool) of unlabeled examples, x, drawn i.i.d. from marginal distribution, D_X over X.

Learner may request labels on examples in the stream.

Oracle access to labels, y in $\{+1, -1\}$ from conditional at x, $D_{Y \mid x}$

Constant cost per label.

The error rate of any classifier v is w.r.t. distribution D:

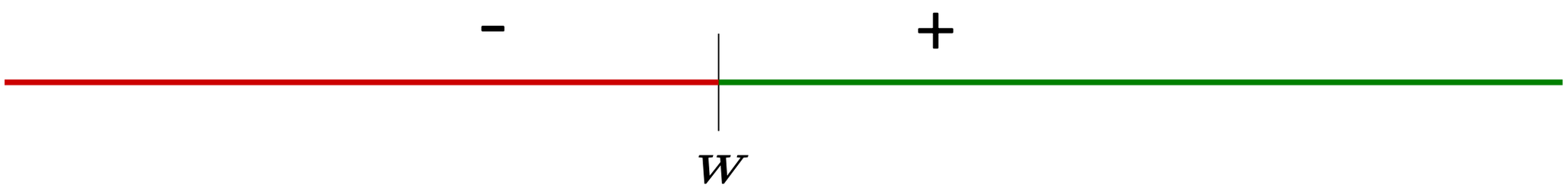
$$err(h) = P_{(x, y)\sim D}[v(x) \neq y]$$

Goal: minimize number of *labels* to learn the concept (w.h.p.) to a fixed final error rate, ε, on input distribution.

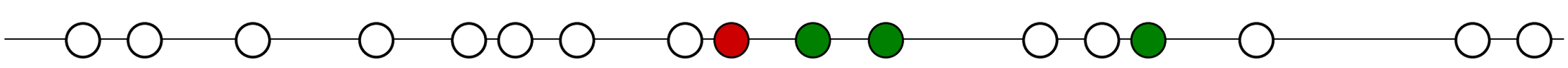
Can active learning really help?

[Cohn, Atlas & Ladner ‘94; Dasgupta ‘04]:
Threshold functions on the real line: $h_w(x) = \text{sign}(x - w)$, $H = \{h_w: w \text{ in } R\}$

Supervised learning: need $\Omega(1/\varepsilon)$ examples to reach error rate $< \varepsilon$.



Active learning: given $1/\varepsilon$ unlabeled points,



Binary search – need just $\log(1/\varepsilon)$ labels, from which the rest can be inferred! Exponential improvement in sample complexity.

→ However, many negative results, e.g. [Dasgupta ‘04], [Kääriäinen ‘06].

More general hypothesis classes

For a general hypothesis class with VC dimension d, is a "generalized binary search" possible?

Random choice of queries d/ε labels

Perfect binary search d log 1/ε labels

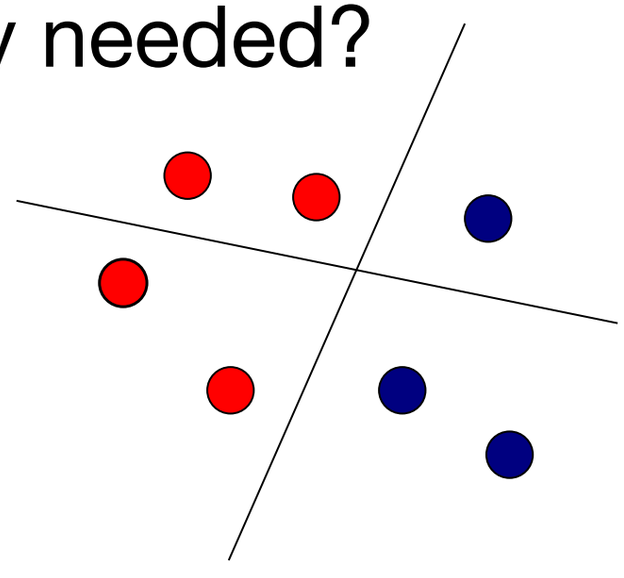
Where in this large range does the label complexity of active learning lie?

We've already handled linear separators in 1-d…

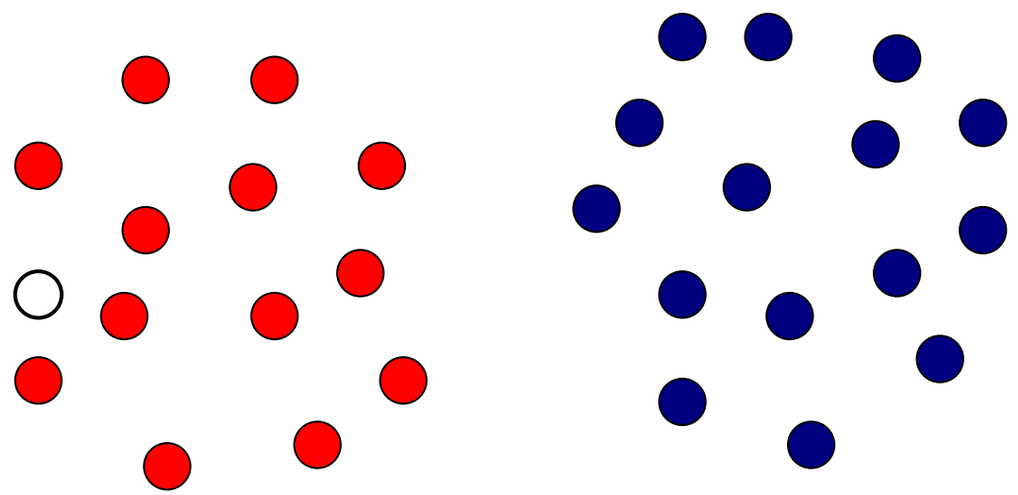
Slide credit: S. Dasgupta

When is a label needed?

Is a label query needed?



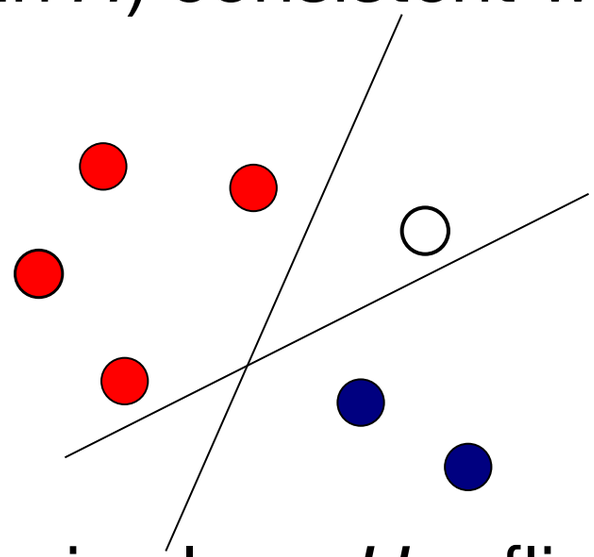
- Linearly separable case: NO
- There may not be a perfect linear separator: YES



- Either case: NO

Selective sampling algorithm

Region of uncertainty [CAL ‘94]: subset of data space for which there exist hypotheses (in *H*) consistent with all previous data, that disagree.



Example: hypothesis class, *H* = {linear separators}. Separable assumption.

Algorithm: Selective sampling [Cohn, Atlas & Ladner ’94]:

For each point in the stream, if point falls in region of uncertainty, request label.

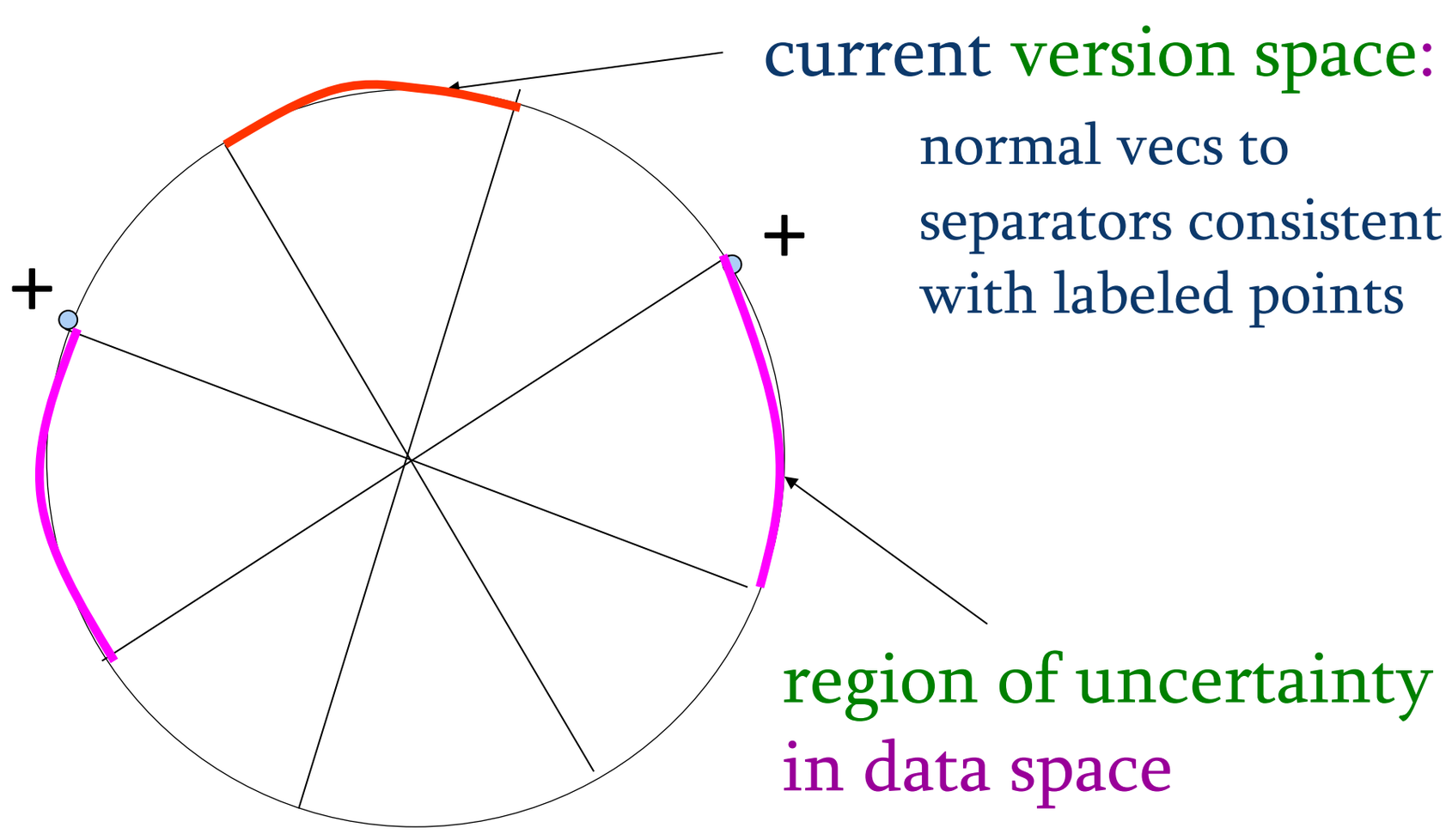
Region of uncertainty

Current version space: portion of H consistent with labels so far.

"Region of uncertainty" = part of data space about which there is still some uncertainty (ie. disagreement within version space)

Suppose data lies on circle in R^2; hypotheses are linear separators.

(spaces X, H superimposed)



Slide credit: S. Dasgupta

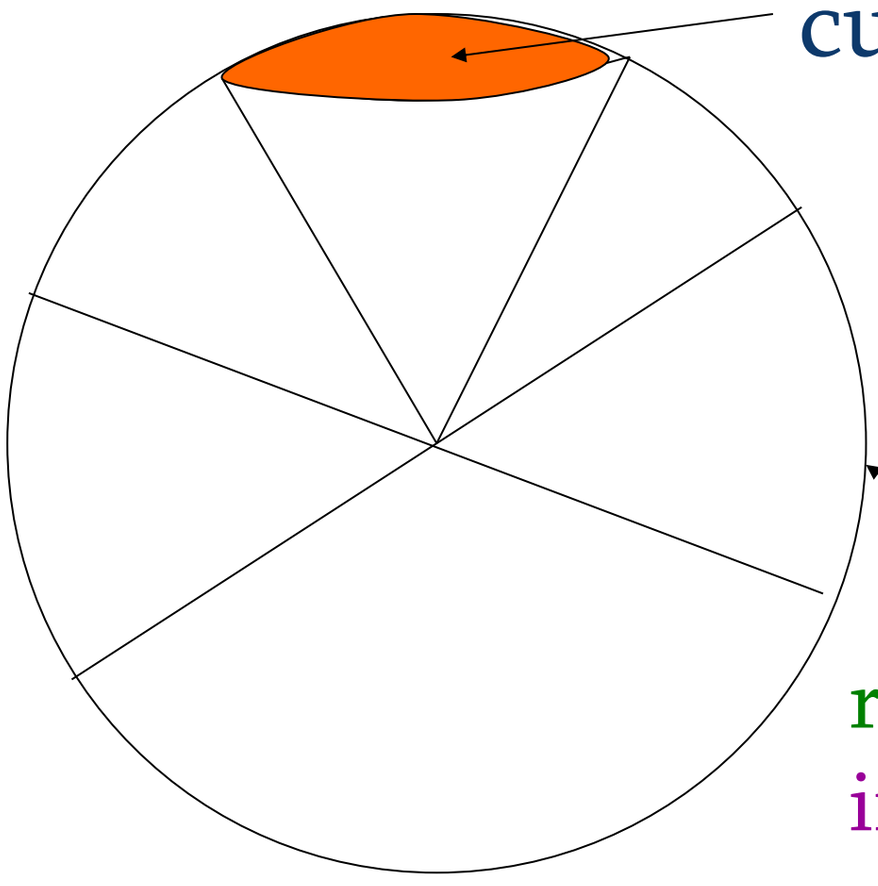
Region of uncertainty

Selective Sampling Algorithm [CAL+ 89]:

Of the unlabeled points which lie in the region of uncertainty, pick one at random and query its label.

Data and hypothesis spaces, superimposed:

(both are the surface of the unit sphere in R^d)



current version space:

normal vecs to separators consistent with labeled points

region of uncertainty in data space

Slide credit: S. Dasgupta

Region of uncertainty

Number of labels needed depends on H and also on P.

Special case: H = {linear separators in R^d}, P = uniform distribution over unit sphere.

Theorem [Dasgupta, Hsu, & Monteleoni, NIPS '07]: $\tilde{O}(d \log 1/\varepsilon)$ labels suffice to reach a hypothesis with error rate < ε.

In contrast: supervised learning: $\Theta(d/\varepsilon)$ labels (PAC complexity).

Slide credit: S. Dasgupta

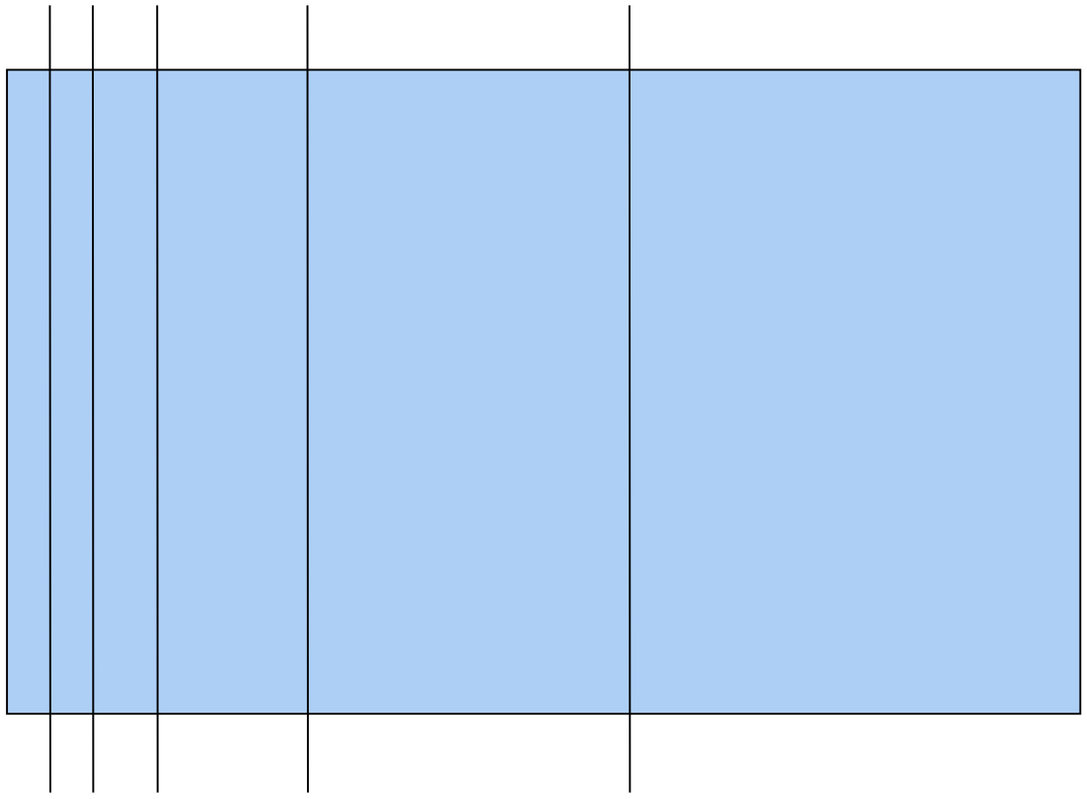
Query-by-committee

First idea: Try to rapidly reduce volume of version space?

Problem: doesn't take data distribution into account.

To keep things simple, say d(h,h') ≈ Euclidean distance in this picture.

H:



Error is likely to remain large!

Slide credit: S. Dasgupta

Query-by-committee

[Seung, Opper, Sompolinsky, 1992; Freund, Seung, Shamir, Tishby 1997]

First idea: Try to rapidly reduce volume of version space?

Problem: doesn't take data distribution into account.

H:

Which pair of hypotheses is closest? Depends on data distribution P.

Distance measure on H: $d(h,h') = P[h(x) \neq h'(x)]$

Slide credit: S. Dasgupta

Query-by-committee

Elegant scheme which decreases volume in a manner which is sensitive to the data distribution.

Bayesian setting: given a prior π on H

$H_1 = H$

For t = 1, 2,

Receive an unlabeled point x_t drawn from P

Choose two hypotheses h,h' randomly (i.i.d.) from (π, H_t)

If $h(x_t) \neq h'(x_t)$: ask for x_t's label

H_{t+1}= all hypotheses in H_t consistent with x_t and label

Else $H_{t+1} = H_t$

Slide credit: S. Dasgupta

Query-by-committee

For t = 1, 2, …

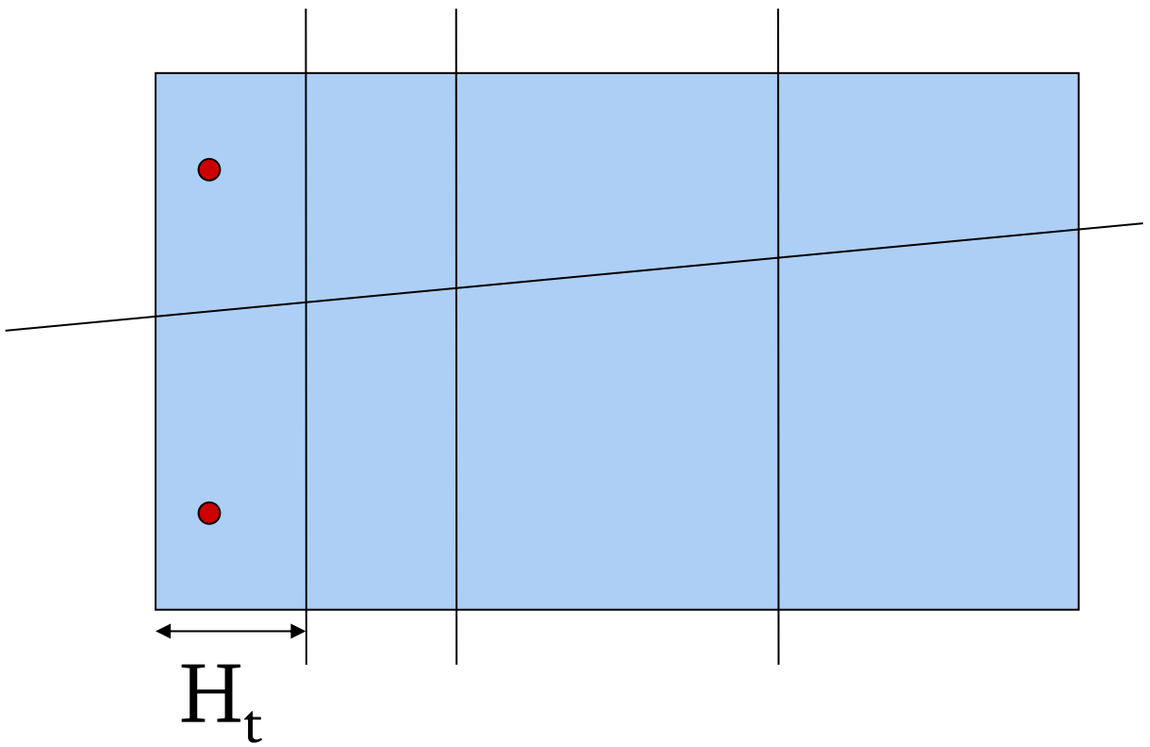
receive an unlabeled point x_t drawn from P

choose two hypotheses h,h' randomly from (π, H_t)

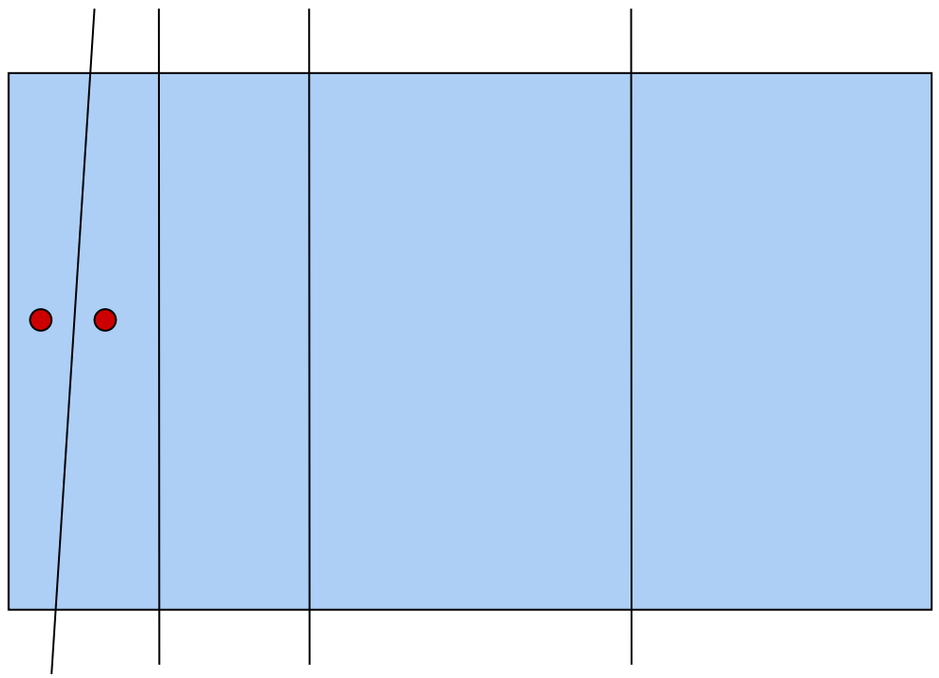
if $h(x_t) \neq h'(x_t)$: ask for x_t's label

set H_{t+1}

Observation: the probability of getting pair (h,h') that leads to a label query is proportional to $\pi(h)\ \pi(h')\ d(h,h')$.



vs.



Slide credit: S. Dasgupta

Query-by-committee

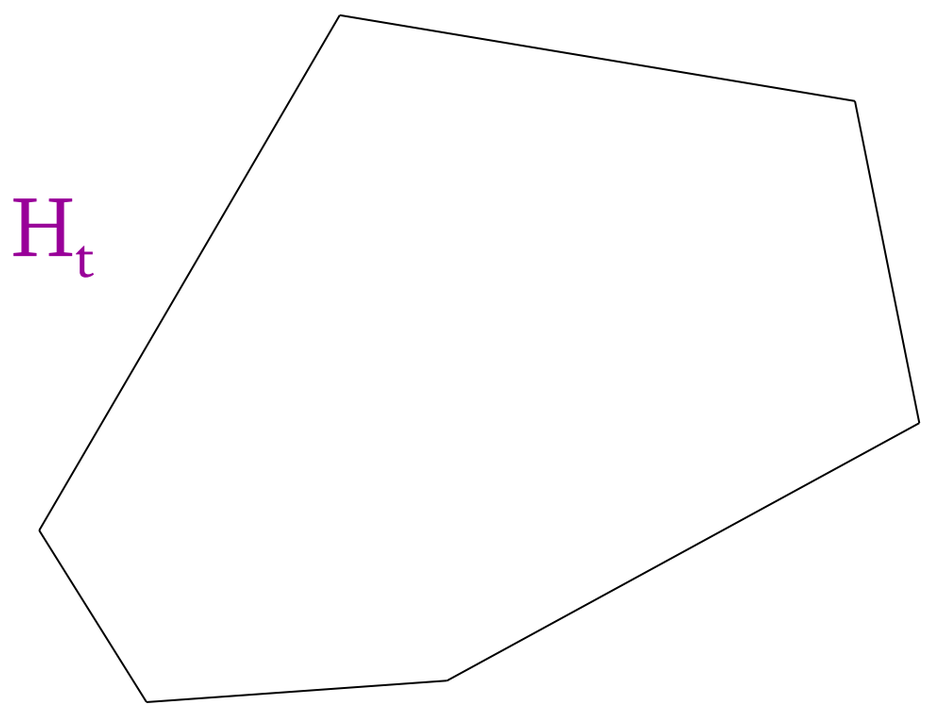
Label bound, Theorem [FSST97] :

For H = {linear separators in R^d}, P = uniform distribution, then

$\tilde{O}(d \log 1/\varepsilon)$ labels suffice to reach a hypothesis with error < ε.

Implementation: need to randomly pick h according to (π, H_t).

e.g. H = {linear separators in R^d}, π = uniform distribution:



How do you pick a random point from a convex body? (Difficult)

Random walk techniques, see a practical variant: [Gilad-Bachrach, Navot & Tishby NIPS '05]

Slide credit: S. Dasgupta

Bonus slides

Active Learning

For linear separators in high dimension, is a generalized binary search possible, allowing exponential label savings?

[Dasgupta, Kalai & Monteleoni, JMLR 2009 (COLT 2005)]: Online active learning with exponential error convergence.

Theorem. There exists an online active learning algorithm that converges to generalization error □□after Õ(d log 1/□) labels.

Corollary. The total errors (labeled and unlabeled) will be at most Õ(d log 1/□).

Active Learning

In general, is it possible to reduce active learning to supervised learning?

[Monteleoni, Open Problem, COLT 2006]: Goal: general, efficient active learning.

[Dasgupta, Hsu & Monteleoni, NIPS 2007]: General active learning via reduction to supervised learning.

Problem: efficient, general AL

[Monteleoni, Open Problem, COLT 2006]

Efficient algorithms for active learning under general input distributions, *D*.

→ Previous label complexity upper bounds for general distributions are based on *intractable* schemes!

Provide an algorithm such that w.h.p.:

1. After *L* label queries, algorithm's hypothesis *v* obeys:

   $P_{x \sim D_X}[v(x) \neq u(x)] < \varepsilon$.

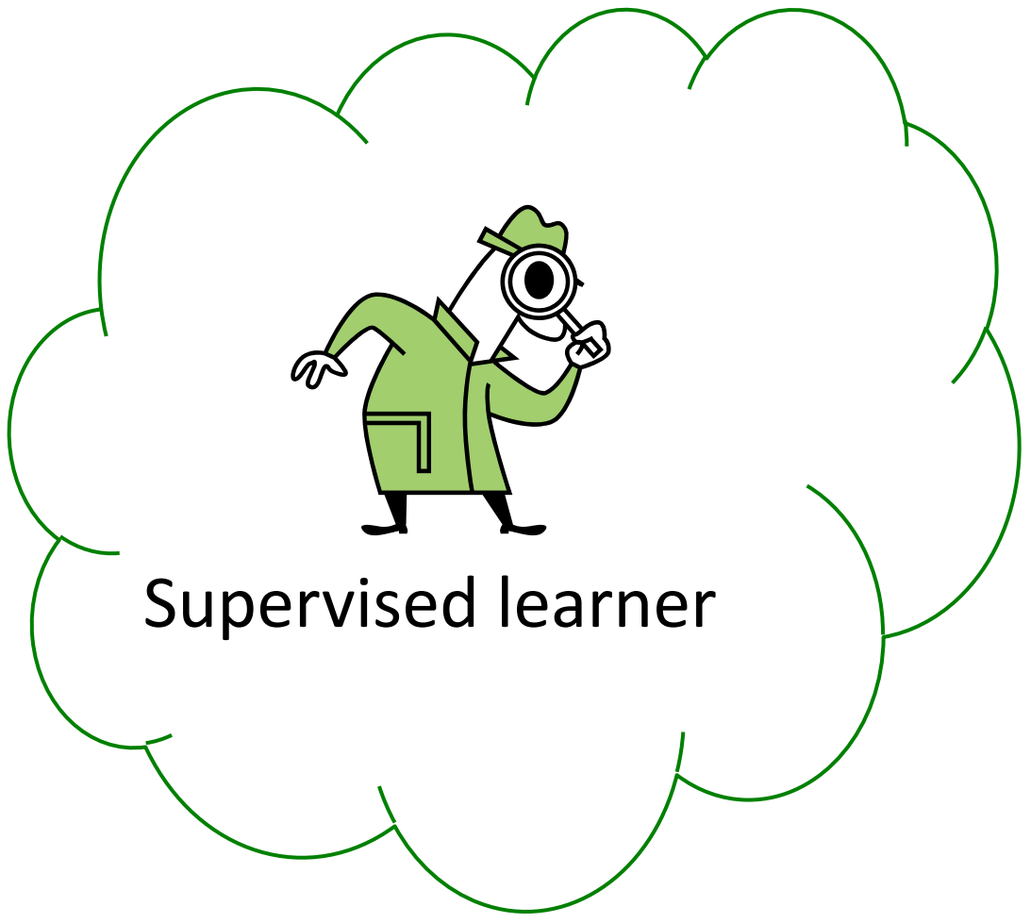
2. *L* is at most the supervised sample complexity, and for a general class of input distributions, *L* is significantly lower.
3. Running time is at most *poly*(d, $1/\varepsilon$).

→ Was open even for specific hypothesis classes, batch case!

General active learning via reduction

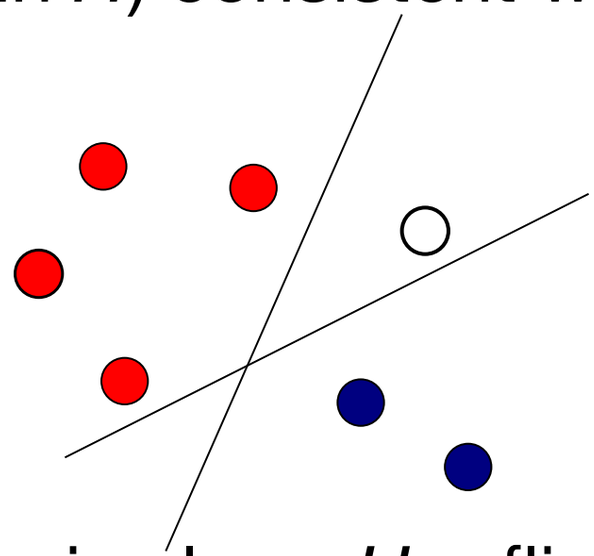
First reduction from active learning to supervised learning.

- Any data distribution (including arbitrary noise)
- Any hypothesis class



Selective sampling algorithm

Region of uncertainty [CAL '94]: subset of data space for which there exist hypotheses (in *H*) consistent with all previous data, that disagree.



Example: hypothesis class, *H* = {linear separators}. Separable assumption.

Algorithm: Selective sampling [Cohn, Atlas & Ladner '94] (orig. NIPS 1989):

For each point in the stream, if point falls in region of uncertainty, request label.

Easy to represent the region of uncertainty for certain, separable problems. BUT, in this work we address:

- What if data is not separable?
- General hypothesis classes?

} → Reduction!

General active learning via reduction

[Dasgupta, Hsu & Monteleoni, NIPS 2007]

First positive step towards answering [M, Open Problem, COLT 2006]:
general active learning: arbitrary input distribution and hypothesis class.

Technique: reduce to supervised learning. Call a supervised learner twice to determine whether a current unlabeled point is "uncertain." Only request labels on uncertain points.

Performance guarantees:

Upper bounds on label complexity:

- Never worse than supervised (PAC) sample complexity.
- Exponential savings for families of distributions/problems.

Consistency: algorithm's error converges to optimal.

Efficiency: running time is at most (up to polynomial factors) that of supervised learning algorithm for the problem.

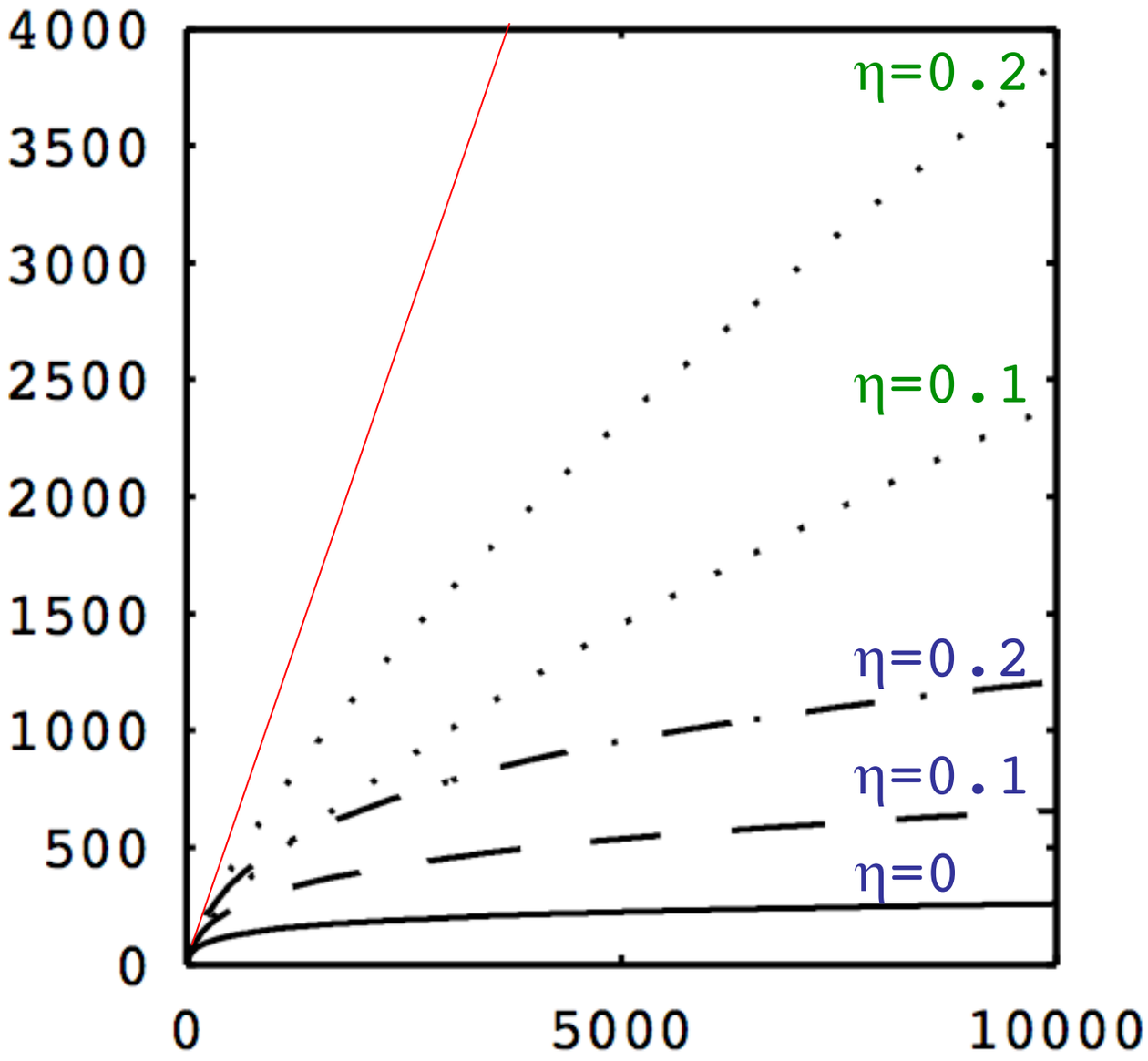
Active learning via reduction: experiments

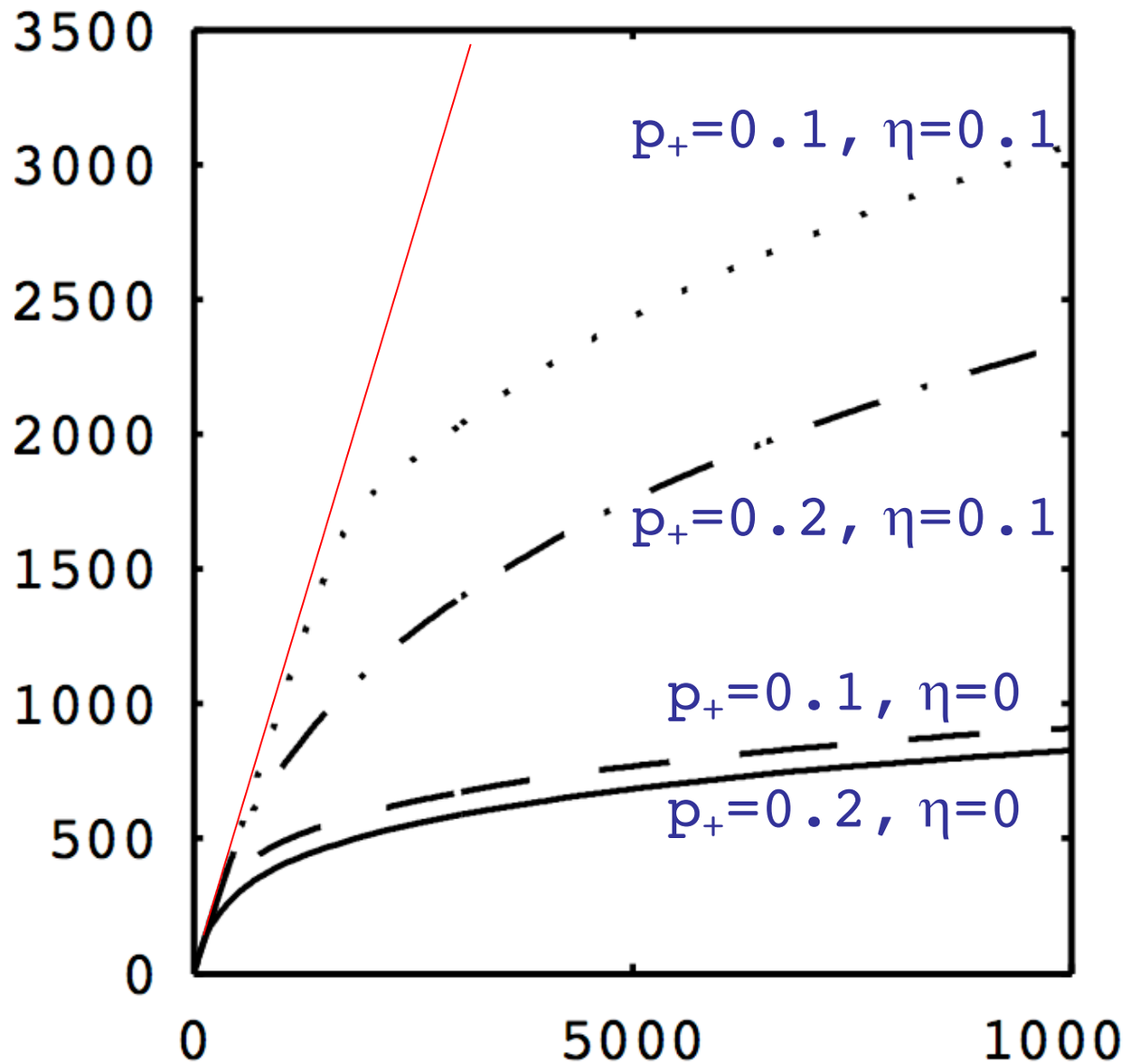
Hypothesis classes in R^1:

Thresholds: $h^*(x) = \text{sign}(x - 0.5)$

Intervals: $h^*(x) = I(x \text{ in } [\text{low}, \text{high}])$

$p_+ = P_{x \sim D_X}[h^*(x) = +1]$





Number of label queries versus points received in stream.

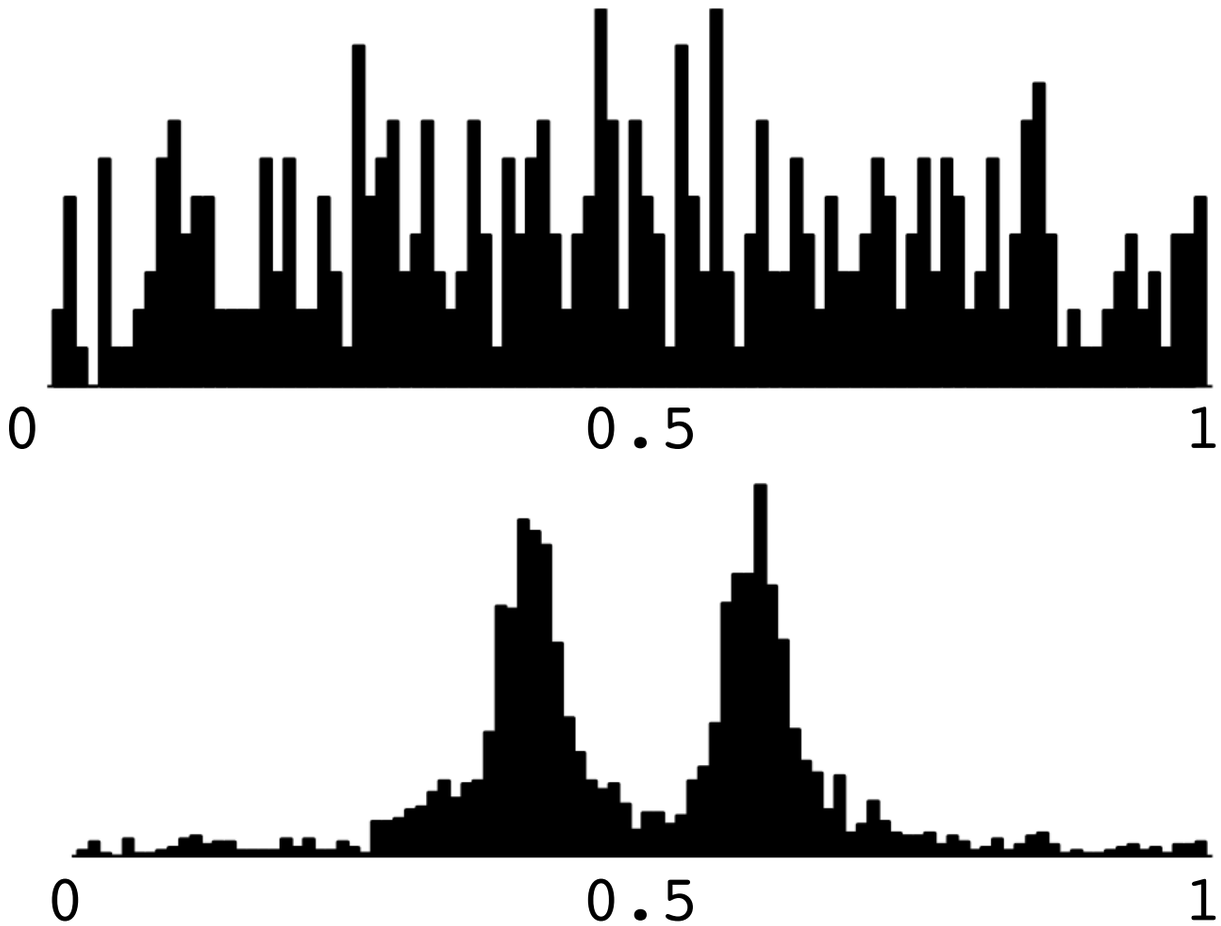
Red: supervised learning. Blue: random misclassification, Green: Tsybakov boundary noise

Experiments

Interval in R^1:

$h^*(x) = I(X \text{ in } [0.4, 0.6])$

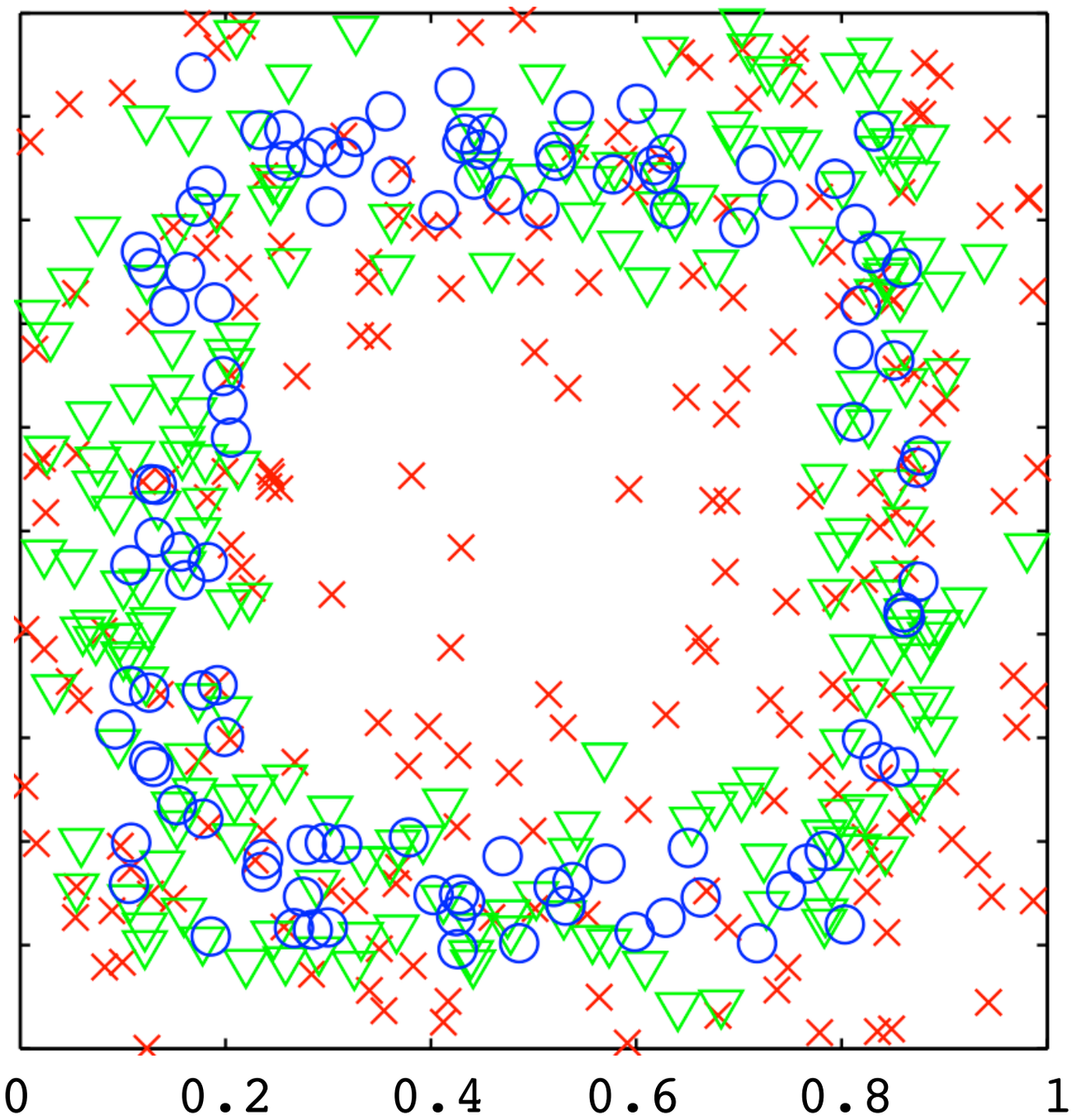
Label queries: 1-400:



All label queries (1-2141).

Interval in R^2 (Axis-parallel boxes):

$h^*(x) = I(X \text{ in } [0.15, 0.85]^2)$



Temporal breakdown of label request locations. Queries: 1-200, 201-400, 401-509.

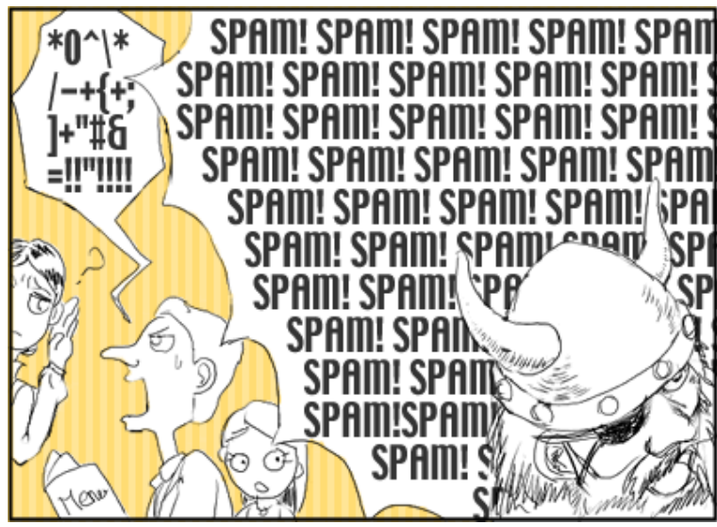
Online active learning: motivations

Data-rich applications:

Spam filtering, e.g. [Sculley CEAS 2007]

Image/webpage relevance filtering

Object detection in video



Resource-constrained applications:

Interactive learning on sensors, mobile robots.

Human-interactive learning on small devices:

e.g. OCR on handhelds used by doctors.

Active learning scenario:

User writes characters.

Device occasionally queries a label (user must type into keypad). Human users likely prefer fewer interruptions (label queries).

Online active learning: OCR application

[M & Kääriäinen CVPR workshop ‘07]

We apply online active learning to OCR due to its potential efficacy for OCR on small devices:

Scenario: user writes characters.

Device occasionally queries a label (user must type into keypad).

Human users likely prefer fewer interruptions (label queries).

OCR data highly non-uniform: test [DKM’05/’09] algorithm when relax distributional and separability assumptions.

Algorithms and evaluation

[Cesa-Bianchi,Gentile & Zaniboni ‘06] algorithm (parameter b):

Filtering rule: flip a coin w.p. $b/(b + |x \cdot v_t|)$

Update rule: standard Perceptron.

Relative bounds on error w.r.t. best linear classifier (regret, non i.i.d.).

Fraction of labels queried depends on b.

Experiments with all 6 combinations of:

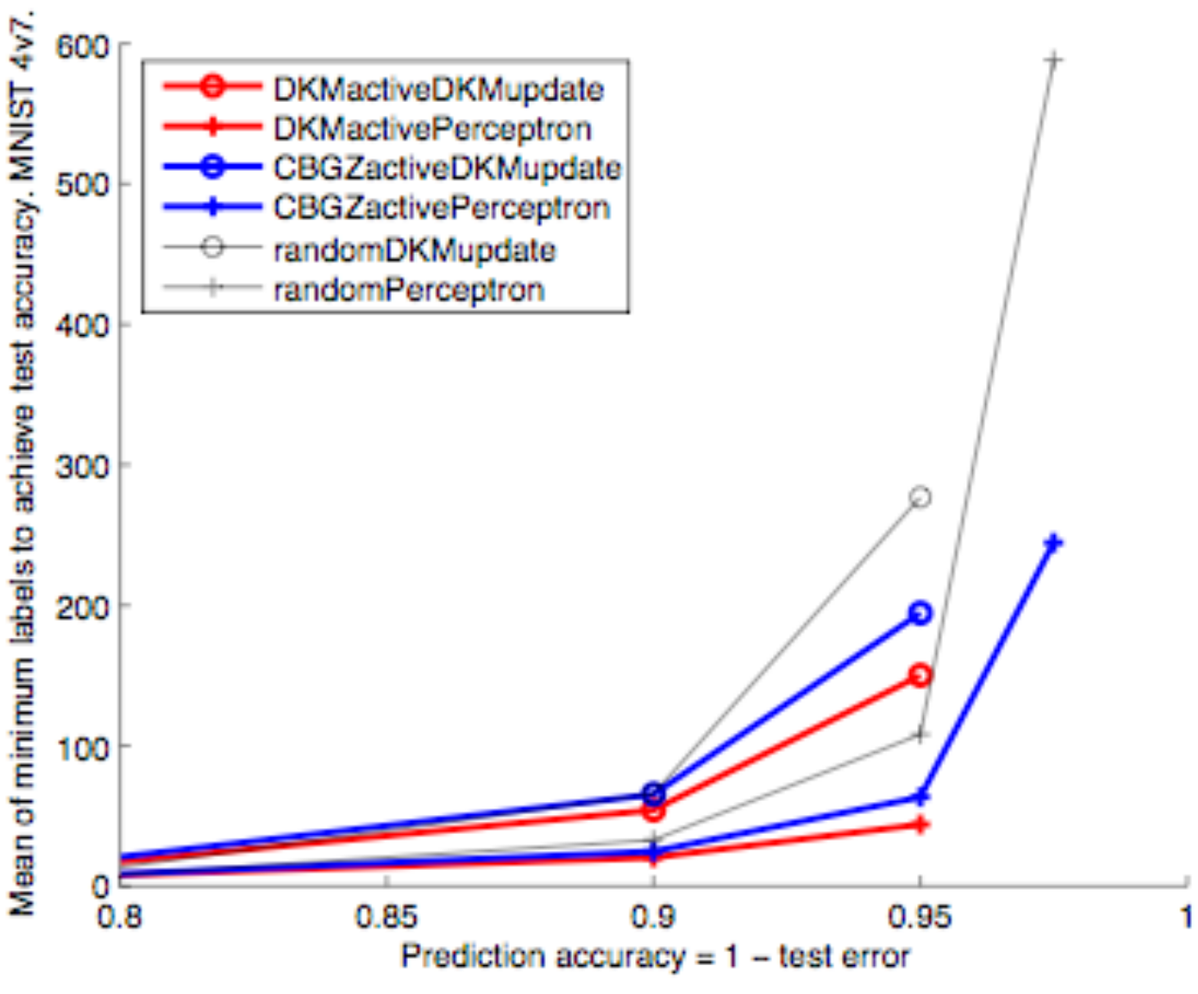
Update rule in {Perceptron, DKM modified Perceptron}

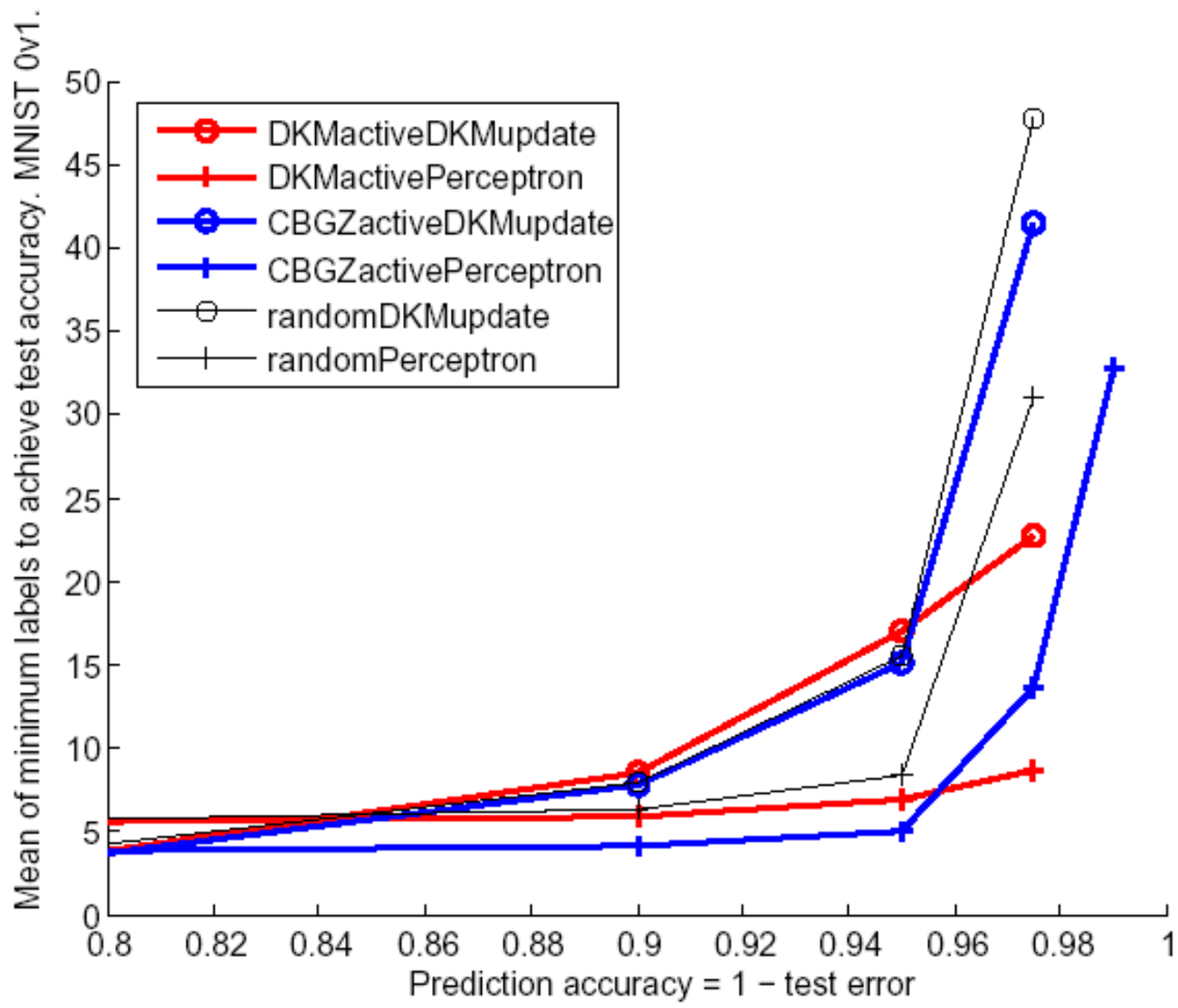
Active learning logic in {DKM, C-BGZ, random}

MNIST (d=784) and USPS (d=256) OCR data.

7 problems, with approx 10,000 examples each.

5 random restarts of 10-fold cross-validation.





Active learning always quite outperformed random sampling:

Random sampling perc. used 1.26–6.08x as many labels as active.

Factor was at least 2 for more than half of the problems.

DKMperceptron performed best overall, followed by DKM.

Superior supervised learning sub-algorithm: Perceptron.

Superior active learning rule: DKM.

Online active learning framework

Pool-based framework of [Cohn,Atlas&Ladner'89]

Assume a fixed probability distribution, D over X x Y, X some input space, $Y = \{\pm 1\}$.

Given: stream (or pool) of unlabeled examples, x, drawn i.i.d. from marginal distribution, D_X over X.

Learner may request labels on examples in the stream.

Oracle access to labels, y in $\{\pm 1\}$ from conditional at x, $D_{Y \mid x}$

Constant cost per label.

The error rate of any classifier v is measured on distribution D:

$$err(v) = P_{(x, y)\sim D}[v(x) \neq y]$$

Goal: minimize number of *labels* to learn the concept (whp) to a fixed final error rate, ε, on input distribution.

Must respect online constraints on time and memory.

Measures of complexity

PAC sample complexity:

Supervised setting: number of (labeled) examples, sampled iid from *D*, to reach error rate ε.

Mistake-complexity:

Supervised setting: number of mistakes to reach error rate ε.

Label-complexity:

Active setting: number of label queries to reach error rate ε.

Error complexity:

Total prediction errors made on (labeled and/or unlabeled) examples, before reaching error rate ε.

Supervised setting: equal to mistake-complexity.

Active setting: mistakes are a subset of total errors on which learner queries a label.