New Directions in Learning Theory

Avrim Blum
Carnegie Mellon University

[ACM-SIAM Symposium on Discrete Algorithms 2015]

Machine Learning

Machine Learning is concerned with:

- Making useful, accurate generalizations or predictions from data.
- Improving performance at a range of tasks from experience, observations, and feedback.

Typical ML problems:

Given database of images, classified as  
male or female, learn a rule to classify new images.

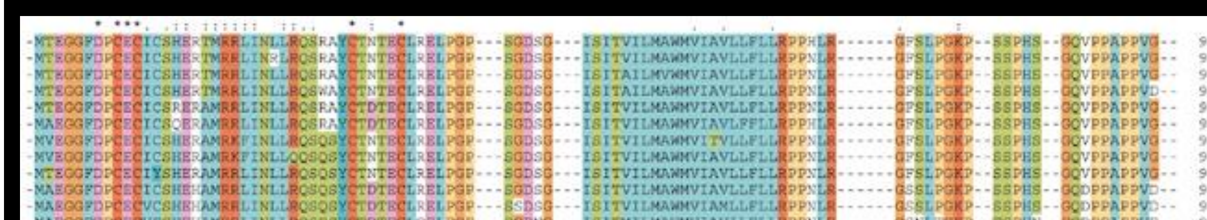
Machine Learning

Machine Learning is concerned with:

- Making useful, accurate generalizations or predictions from data.
- Improving performance at a range of tasks from experience, observations, and feedback.

Typical ML problems:

Given database of protein seqs, labeled by function, learn rule to predict functions of new proteins.



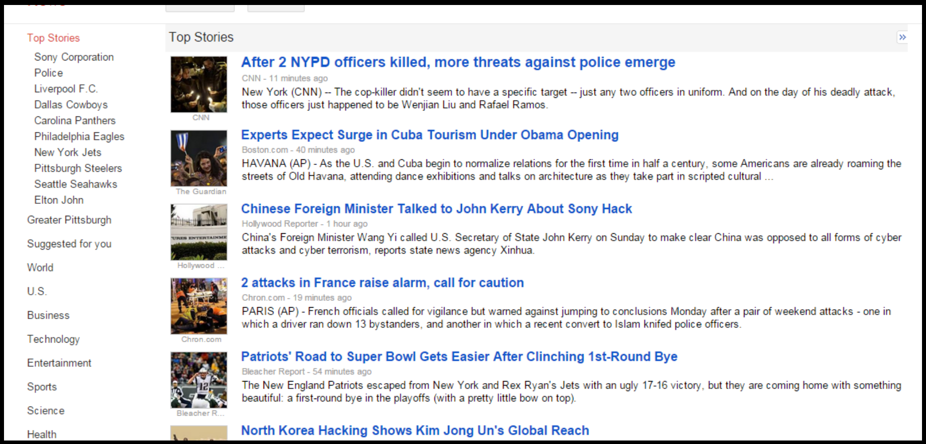
Machine Learning

Machine Learning is concerned with:

- Making useful, accurate generalizations or predictions from data.
- Improving performance at a range of tasks from experience, observations, and feedback.

Typical ML problems:

Given lots of news articles, learn about entities in the world.



Machine Learning

Machine Learning is concerned with:

- Making useful, accurate generalizations or predictions from data.
- Improving performance at a range of tasks from experience, observations, and feedback.

Typical ML problems:

Develop truly useful electronic assistant through personalization plus experience from others.

Machine Learning Theory

Some tasks we'd like to use ML to solve are a good fit to classic learning theory models, others less so.

Today's talk: 3 directions:

1. Distributed machine learning
2. Multi-task, lightly-supervised learning.
3. Lifelong learning and Autoencoding.

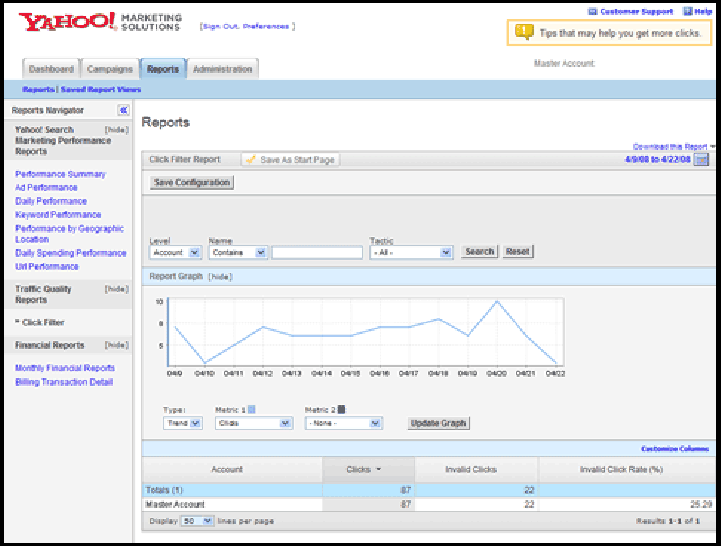
Distributed Learning

Many ML problems today involve massive amounts of data distributed across multiple locations.

Distributed Learning

Many ML problems today involve massive amounts of data distributed across multiple locations.

Click data



Distributed Learning

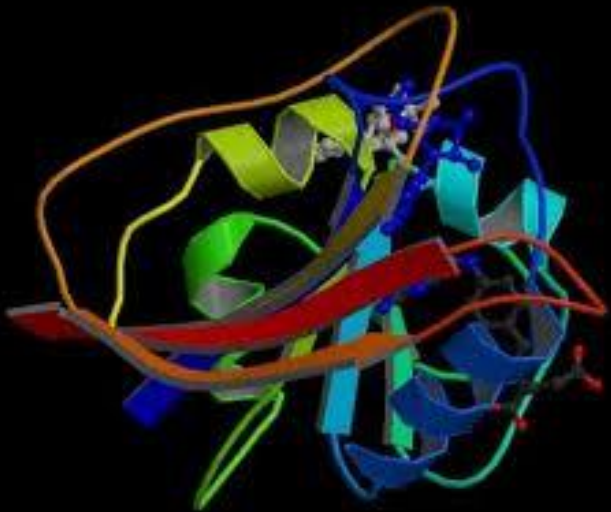
Many ML problems today involve massive amounts of data distributed across multiple locations.

Customer data

Distributed Learning

Many ML problems today involve massive amounts of data distributed across multiple locations.

Scientific data



Distributed Learning

Many ML problems today involve massive amounts of data distributed across multiple locations.

Each has only a piece of the overall data pie

Distributed Learning

Many ML problems today involve massive amounts of data distributed across multiple locations.

In order to learn over the whole thing, holders will need to communicate.

Distributed Learning

Many ML problems today involve massive amounts of data distributed across multiple locations.

Classic ML question: how much data is needed to learn a given type of function well?

Distributed Learning

Many ML problems today involve massive amounts of data distributed across multiple locations.

These settings bring up a new question: how much communication?

Distributed Learning

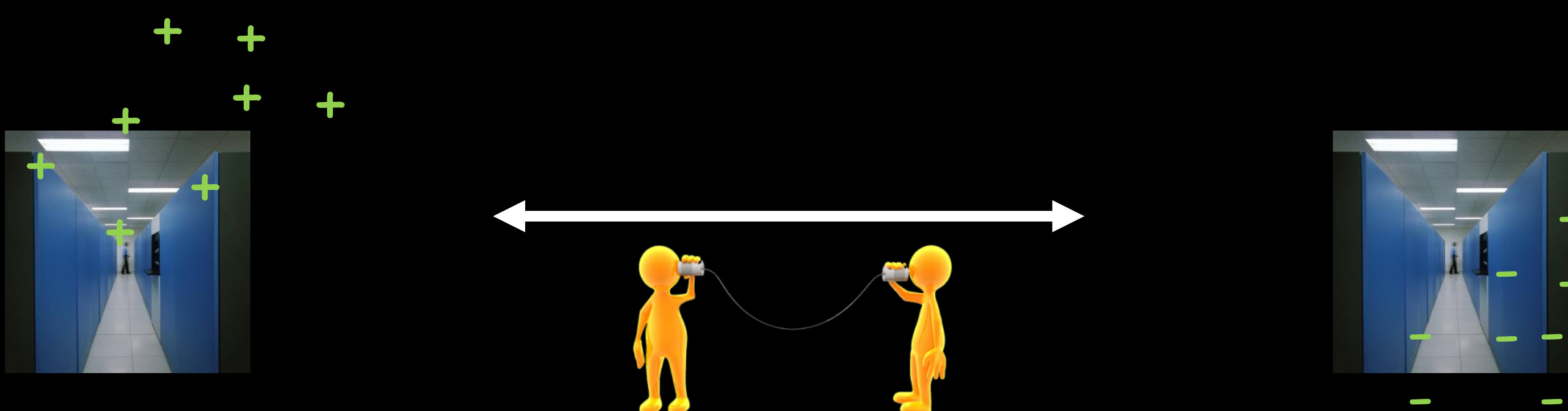
Two natural high-level scenarios:

1. Each location has data from same distribution.
   - So each could in principle learn on its own.
   - But want to use limited communication to speed up – ideally to centralized learning rate.
   - Very nice work of [Dekel, Giliad-Bachrach, Shamir, Xiao],…

Distributed Learning

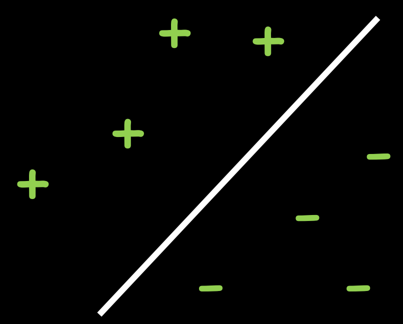
Two natural high-level scenarios:

2. Data is arbitrarily partitioned.
   - E.g., one location with positives, one with negatives.
   - Learning without communication is impossible.
   - This will be our focus here.
   - Based on [Balcan-B-Fine-Mansour]. See also [Daume-Phillips-Saha-Venkatasubramanian].



A model

- Goal is to learn unknown function $f \in C$ given labeled data from some prob. distribution D.
- However, D is arbitrarily partitioned among k entities (players) 1,2,…,k. [k=2 is interesting]



A model

- Goal is to learn unknown function $f \in C$ given labeled data from some prob. distribution D.
- However, D is arbitrarily partitioned among k entities (players) 1,2,…,k. [k=2 is interesting]
- Players can sample $(x,f(x))$ from their own D_i.

$$D = (D_1 + D_2 + \ldots + D_k)/k$$

1

2

…

…

k

A model

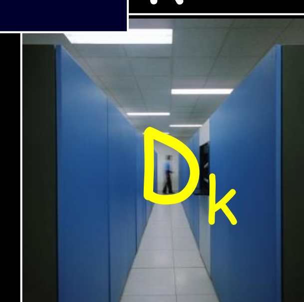
- Goal is to learn unknown function $f \in C$ given labeled data from some prob. distribution D.
- However, D is arbitrarily partitioned among k entities (players) 1,2,…,k. [k=2 is interesting]
- Playe … own D_i.

Goal: learn good rule over combined D, using as little communication as possible.

1

…

k



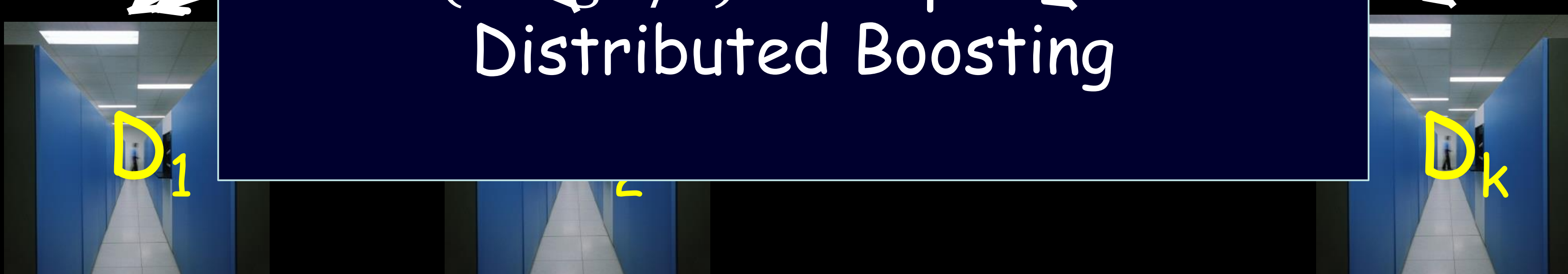
A Simple Baseline

We know we can learn any class of VC-dim d to error ϵ from $m = O(d/\epsilon \log 1/\epsilon)$ examples.

- Each player sends 1/k fraction to player 1.
- Player 1 finds rule h that whp has error $\leq \epsilon$ with respect to D. Sends h to others.
- Total: 1 round, $O(d/\epsilon \log 1/\epsilon)$ examples sent.

Can we do better in general? Yes.

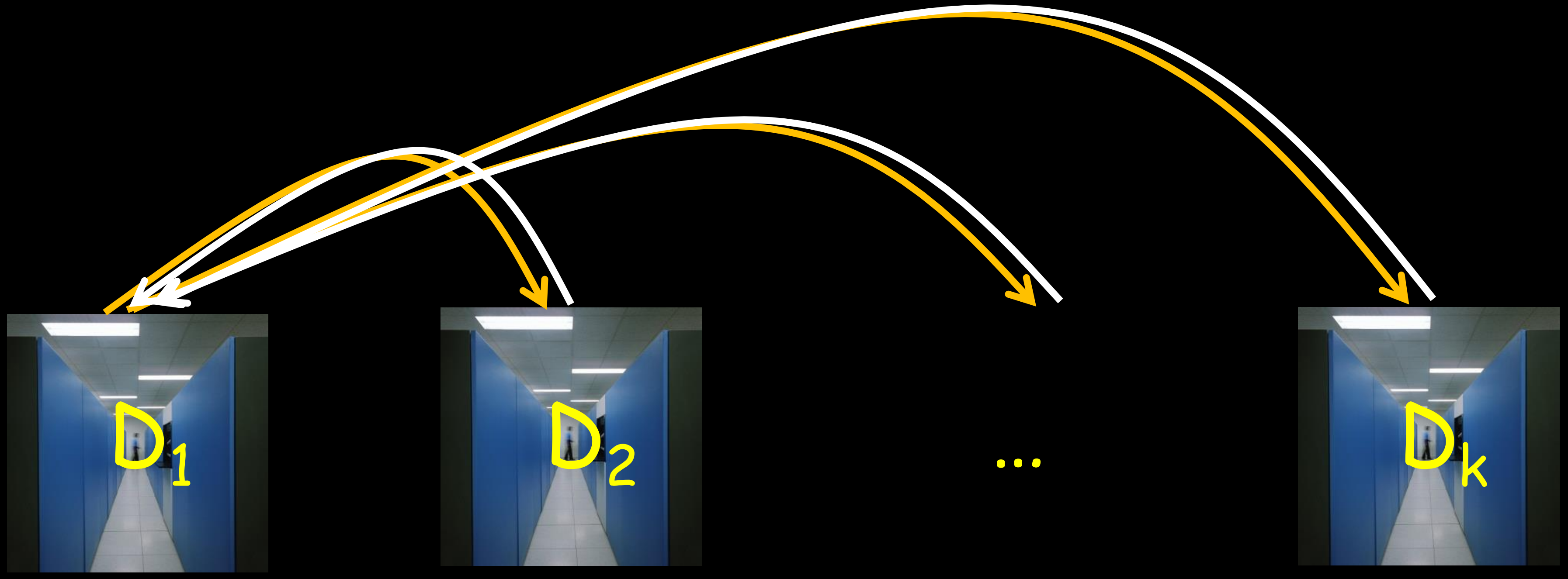
$O(d \log 1/\epsilon)$ examples with Distributed Boosting



Distributed Boosting

Idea:

- Run baseline #1 for $\epsilon = \frac{1}{4}$. [everyone sends a small amount of data to player 1, enough to learn to error $\frac{1}{4}$]
- Get initial rule h_1, send to others.



Distributed Boosting

Idea:

- Players then reweight their D_i to focus on regions h_1 did poorly.
- Repeat

- Distributed implementation of Adaboost Algorithm.
- Some additional low-order communication needed too (players send current performance level to #1, so can request more data from players where h doing badly).
- Key point: each round uses only O(d) samples and lowers error multiplicatively.
- Total $O(d \log 1/\epsilon)$ examples + $O(k \log d)$ extra bits.

…

Can we do better for specific classes of functions?

Yes.

Here are two classes with interesting open problems.

...

Parity functions

Examples $x \in \{0,1\}^d$. $f(x) = x \cdot v_f \bmod 2$, for unknown v_f.

- Interesting for k=2.
- Classic communication LB for determining if two subspaces intersect.
- Implies $\Omega(d^2)$ bits LB to output good v.
- What if allow rules that "look different"?

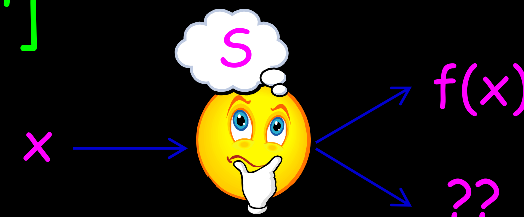
Parity functions

Examples $x \in \{0,1\}^d$. $f(x) = x \cdot v_f \bmod 2$, for unknown v_f.

- Parity has interesting property that:

(a) Can be "properly" PAC-learned. [Given dataset S of size $O(d/\epsilon \log 1/\epsilon)$, just solve the linear system]

(b) Can be "non-properly" learned in reliable-useful model of Rivest-Sloan'88. [if x in subspace spanned by S, predict accordingly, else say "??"]



Parity functions

Examples $x \in \{0,1\}^d$. $f(x) = x \cdot v_f \bmod 2$, for unknown v_f.

- Algorithm:
  - Each player i properly PAC-learns over D_i to get parity function g_i. Also improperly R-U learns to get rule h_i. Sends g_i to other player.
  - Uses rule: "if h_i predicts, use it; else use g_{3-i}."

Can one extend to k=3 players?

h_1

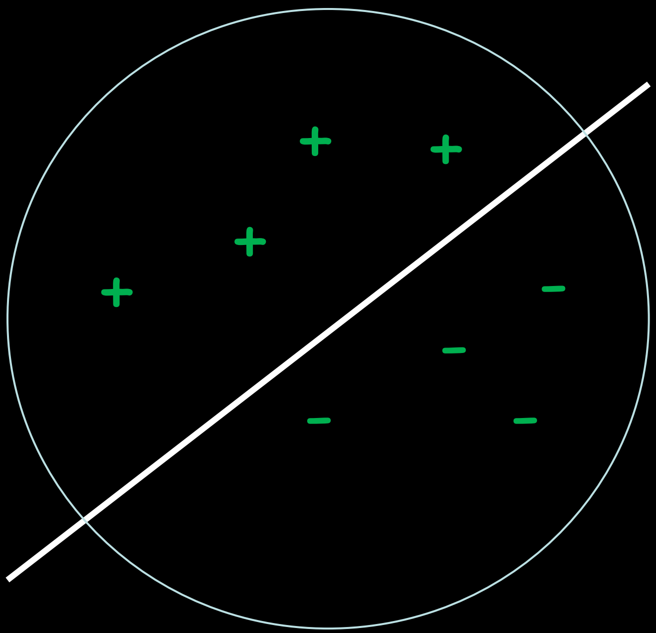
h_2

Linear Separators

Linear separators thru origin. (can assume pts on sphere)

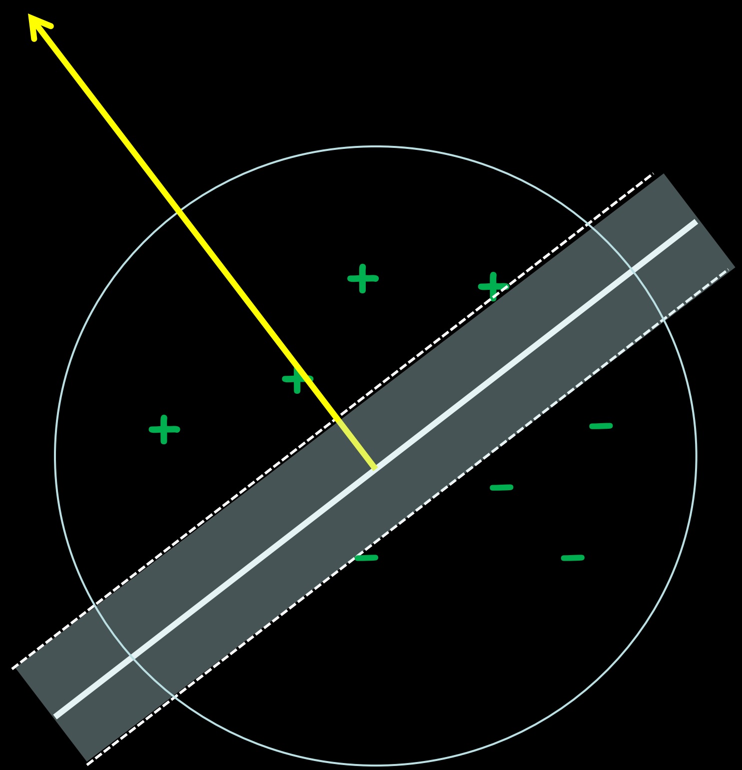
- Say we have a near-uniform prob. distrib. D over S^d.
- VC-bound, margin bound, Perceptron mistake-bound all give O(d) examples needed to learn, so O(d) examples of communication using baseline (for constant k, ϵ).

Can one do better?



Linear Separators

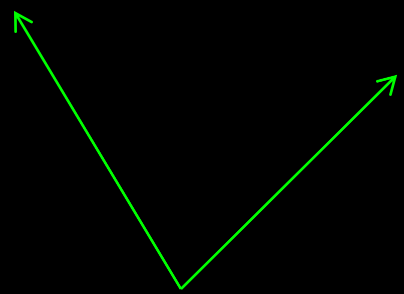
Idea: Use margin-version of Perceptron alg [update until $f(x)(w \cdot x) \geq 1$ for all x] and run round-robin.



Linear Separators

Idea: Use margin-version of Perceptron alg [update until $f(x)(w \cdot x) \geq 1$ for all x] and run round-robin.

- So long as examples x_i of player i and x_j of player j are reasonably orthogonal, updates of player j don't mess with data of player i.
  - Few updates $\Rightarrow$ no damage.
  - Many updates $\Rightarrow$ lots of progress!



Linear Separators

Idea: Use margin-version of Perceptron alg [update until $f(x)(w \cdot x) \geq 1$ for all x] and run round-robin.

- If overall distrib. D is near uniform [density bounded by c·unif], then total communication (for constant k, ϵ) is $O((d \log d)^{1/2})$ vectors rather than $O(d)$.

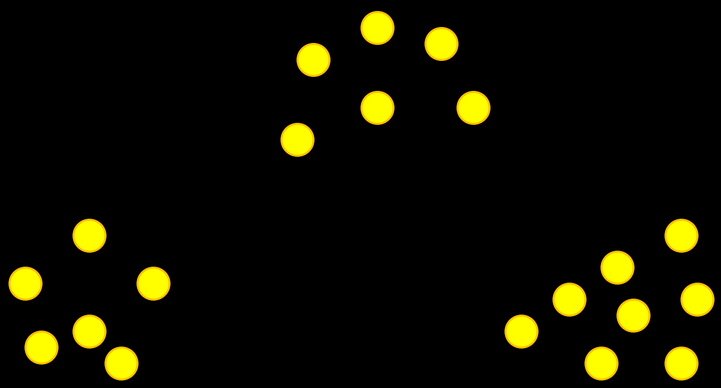
Get similar savings for general distributions?

Under just the assumption that exists a linear separator of margin γ, can you beat $O(1/\gamma^2)$ vectors of angular precision γ?

Clustering and Core-sets

Another very natural task to perform on distributed data is clustering.

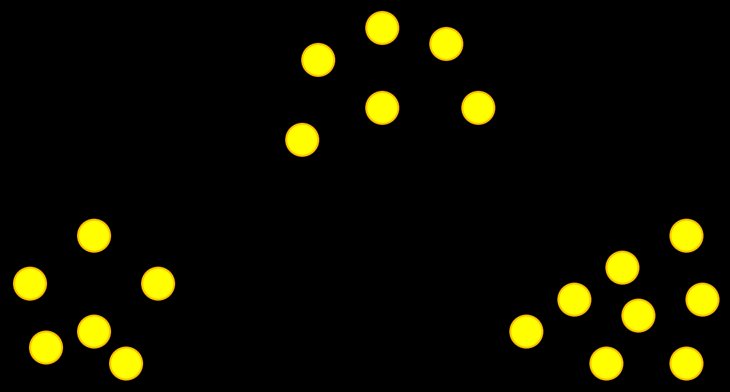
- Data is arbitrarily partitioned among players. (<u>not</u> necessarily related to the clusters)
- Want to solve k-median or k-means problem over overall data. (now using k for # clusters)



Clustering and Core-sets

[Balcan-Ehrlich-Liang] building on [Feldman-Langberg]

- Key to algorithm is a distributed procedure for constructing a small core-set.
  - Weighted set of points S s.t. cost of any proposed k centers on S is within $1 \pm \epsilon$ of cost on entire dataset P.



Clustering and Core-sets

[Balcan-Ehrlich-Liang] building on [Feldman-Langberg]

- Each player i computes $O(1)$ k-me[di]an approx B_i for its own points P_i, sends $cost(P_i, B_i)$ to others.
- Each player i samples using $\Pr(p) \propto cost(p, B_i)$ for # times proportional to overall cost.
- Show an appropriate weighting of samples and B_i is a core-set of the overall dataset.

Overall size $\tilde{O}\left(\frac{kd}{\epsilon^2}\right)$ for k-median, $\tilde{O}\left(\frac{kd}{\epsilon^4}\right)$ for k-means

Use some interactive strategy like in distributed boosting to reduce dependence on ϵ?

Direction 2: multi-task, lightly supervised learning

Growing number of scenarios where we'd like to learn

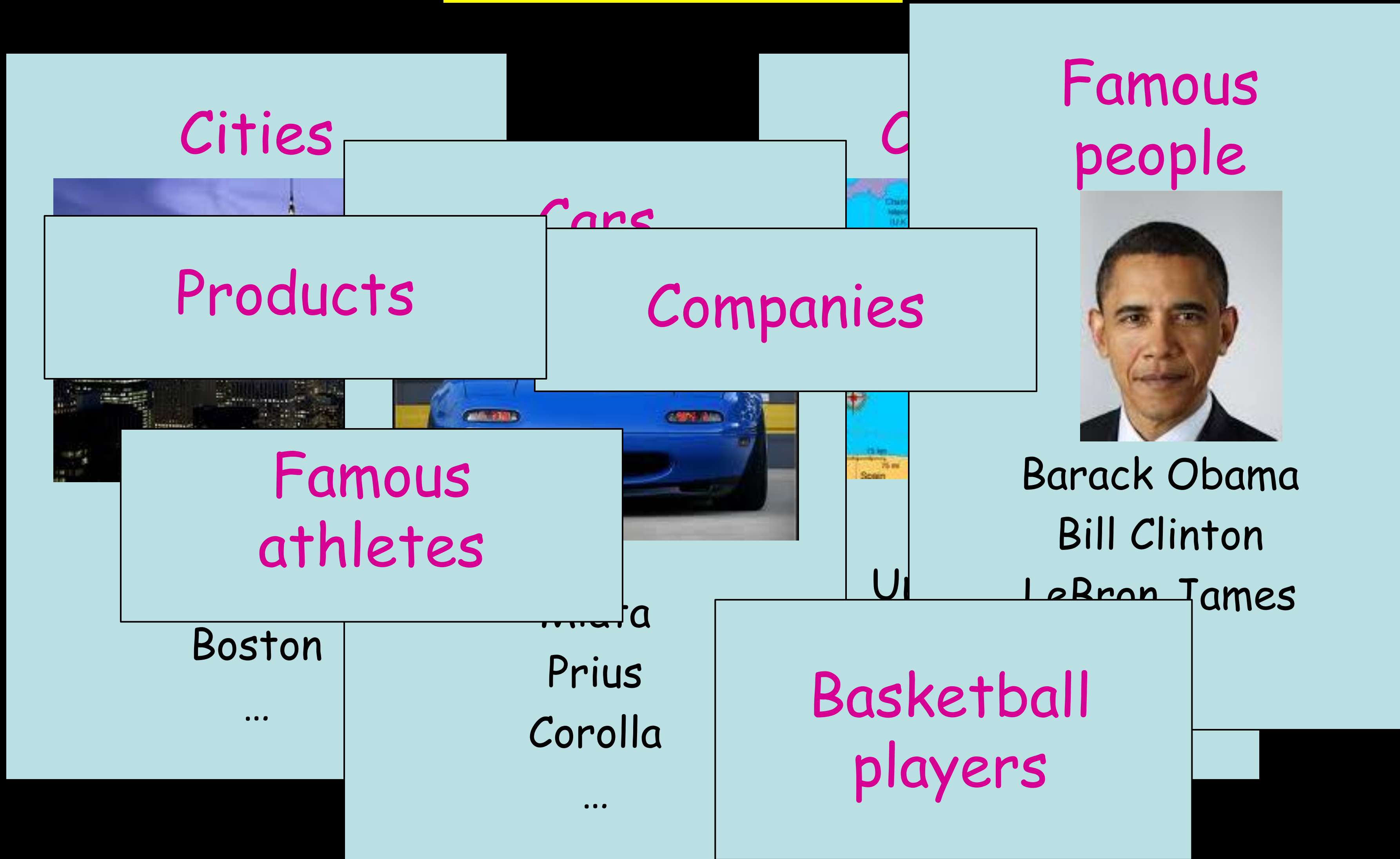
many related tasks

from few labeled examples of each

by leveraging how the tasks are related

E.g., NELL system [Mitchell; Carlson et al] learns multiple related categories by processing news articles on the web

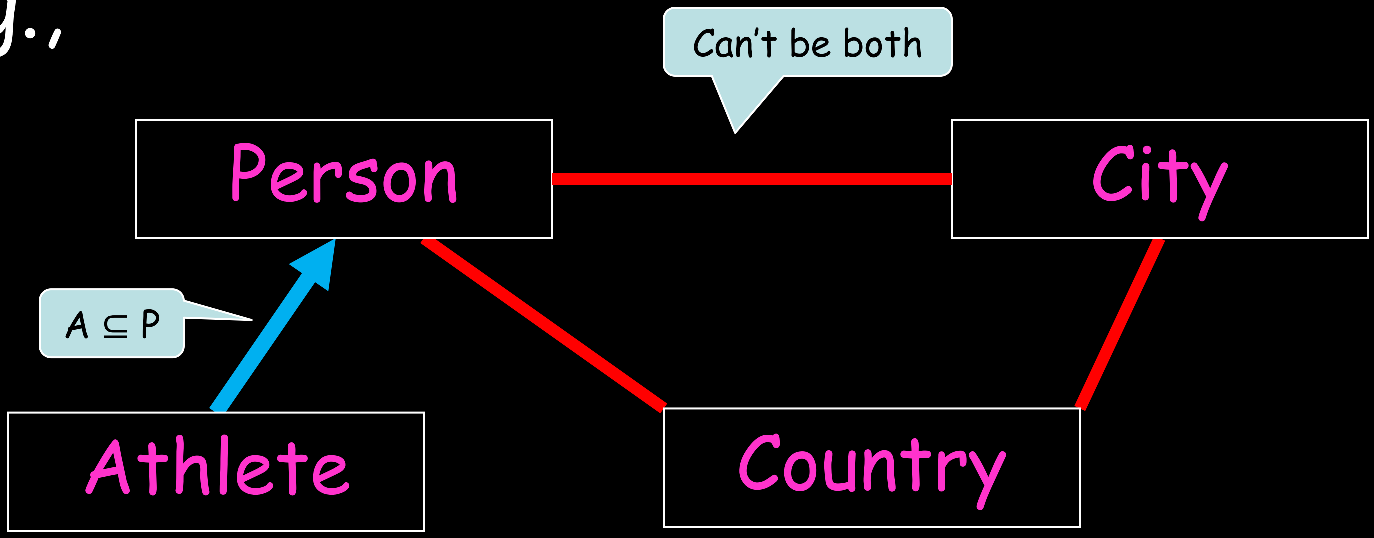
http://rtw.ml.cmu.edu



One thing to work with

Given knowledge of relation among concepts, i.e., an ontology.

E.g.,



Suggests the following idea used in the NELL system

Run separate algorithms for each category:

- Starting from small labeled sample, use patterns found on web to generalize.

…<X> symphony orchestra…

…I was born in <X>…

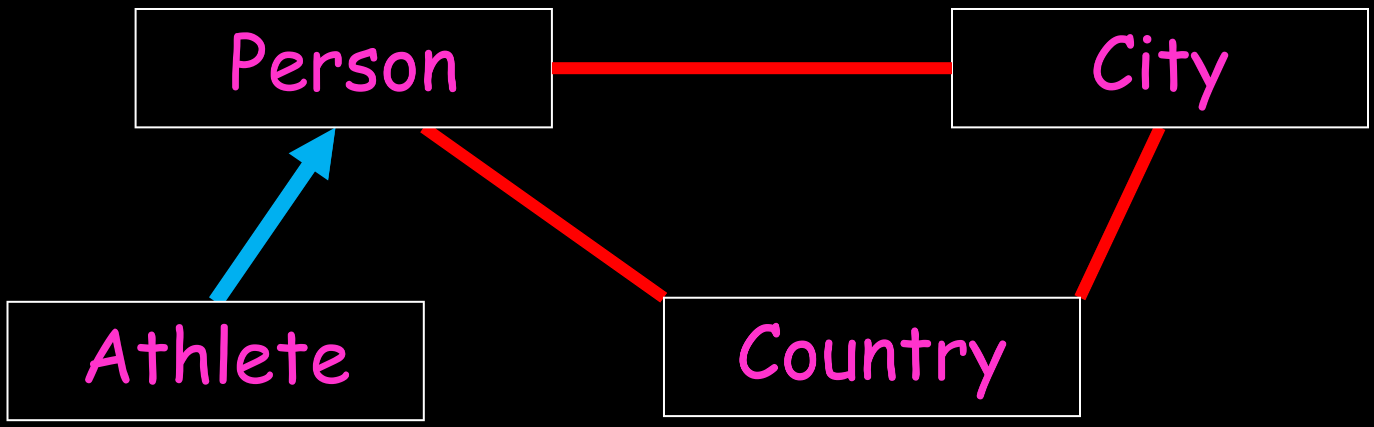
- Use presence of other learning algorithms & ontology to prevent over-generalization.

Can we give a theoretical analysis?

[Balcan-B-Mansour ICML13]

Setup

- L categories.
- Ontology $R \subseteq \{0,1\}^L$ specifies which L-tuples of labelings are legal. (Given to us)
- Focus on those described by a graph of NAND and SUBSET relations.



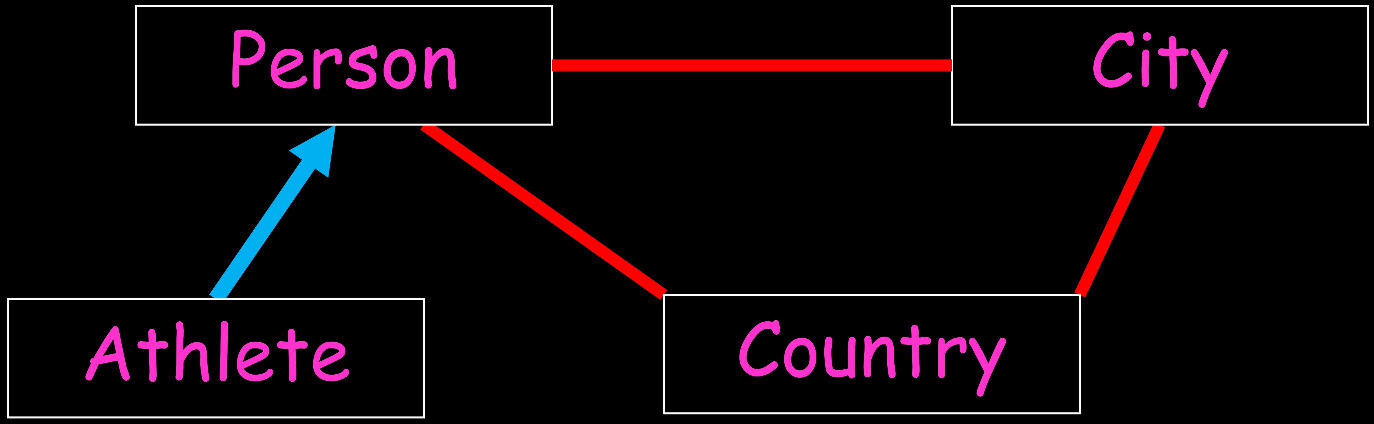
Setup

Use a multi-view framework for examples:

- Example x is L-tuple $(x_1, x_2, \ldots, x_L)$. [x_i is a vector and is the view for category i]

Think of space X_i as plausibly-useful phrases for determining membership in category i.

- c_i^* is target classifier for category i.
- Correct labeling is $(c_1^*(x_1), c_2^*(x_2), \ldots, c_L^*(x_L))$.



Setup

Use a multi-view framework for examples:

- Example x is L-tuple $(x_1, x_2, \dots, x_L)$. [x_i is a vector and is the view for category i]

Think of space X_i as plausibly-useful phrases for determining membership in category i.

- c_i^* is target classifier for category i.
- Correct labeling is $(c_1^*(x_1), c_2^*(x_2), \dots, c_L^*(x_L))$.

Algorithm's goal:

Find $h_1, h_2, \dots, h_L$ s.t. $\Pr_{x \sim D}(\exists i : h_i(x_i) \neq c_i^*(x_i))$ is low.

Unlabeled error rate

Given $h = (h_1, h_2, \ldots, h_L)$, define:

$$err_{unl}(h) = \Pr_{x \sim D}\Big((h_1(x_1), h_2(x_2), \ldots, h_L(x_L)) \notin R\Big)$$

Clearly, $err_{unl}(h) \leq err(h)$.

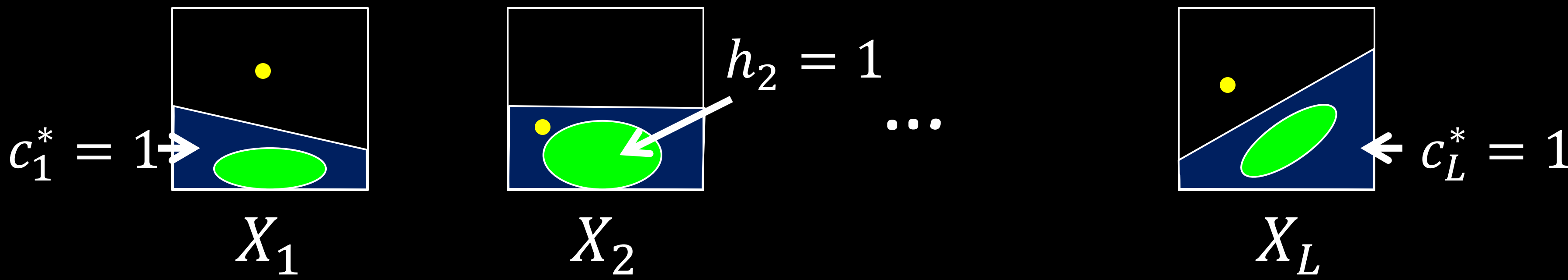
Prediction violates ontology $\Rightarrow$ Prediction differs from target

If we could argue some form of the other direction, then in principle could optimize over just unlabeled data to get low error.

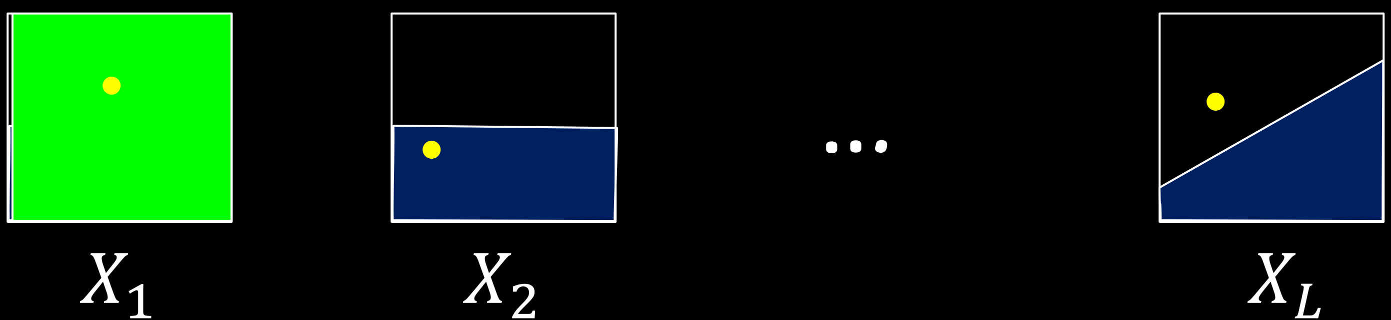
Some complications

Let's consider ontology of complete NAND graph.

Any rule $h = (h_1, h_2, \ldots, h_L)$ of this form has $err_{unl}(h) = 0$.



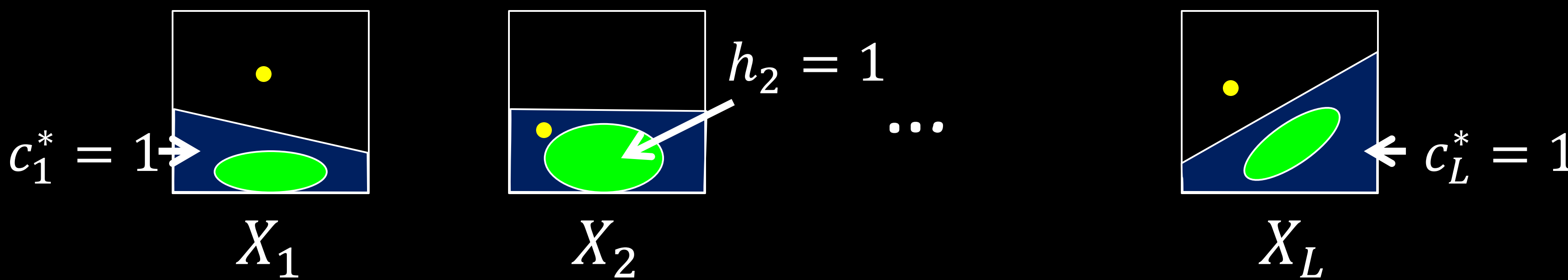
Or of this form:



(one h_i always positive, the rest always negative)

But here is something you can say

If we assume $\Pr_{D}[c_i^*(x_i) = 1] \in [\alpha, 1-\alpha]$.



And we maximize aggressiveness

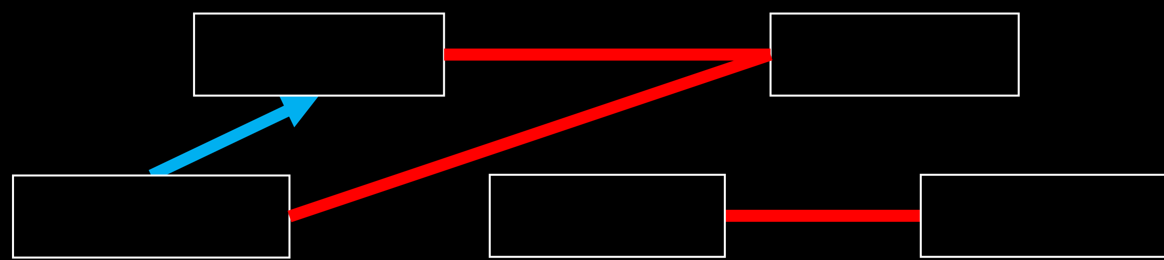
$$\boxed{\sum_i \Pr_D(h_i(x_i) = 1)}$$

subject to low $err_{unl}(h)$ and $Pr(h_i(x_i) = 1) \in [\alpha, 1-\alpha]$ for all i,

can show achieves low $err(h)$ under a fairly interesting set of conditions.

More specifically

1. Assume each category has at least 1 NAND incident edge.



2. For each edge, all 3 non-disallowed options appear with prob $\geq \alpha'$. [e.g., person-noncity, nonperson-city, nonperson-noncity]

3. For any categories i, j, rules h_i, h_j, labels l_i, l_i', l_j, l_j',

$$\boxed{\begin{aligned}&\Pr\left(h_i(x_i) = l_i' \middle| c_i^*(x_i) = l_i, h_j(x_j) = l_j', c_j^*(x_j) = l_j\right) \\ &\quad \geq \lambda \cdot \Pr(h_i(x_i) = l_i' | c_i^*(x_i) = l_i)\end{aligned}}$$

for some $\lambda > 0$.

More specifically

Then to achieve $err(h) \leq \epsilon$, it suffices to choose most aggressive h subject to

$$\Pr(h_i(x_i) = 1) \in [\alpha, 1 - \alpha] \quad \text{and} \quad err_{unl}(h) \leq \frac{\alpha\alpha'\lambda^2\epsilon}{4L}.$$

"aggressive" $=$ maximizing $\sum_i \Pr_D(h_i(x_i) = 1)$

Application to stylized version of NELL-type algorithm

Can we use this to analyze iterative greedy "region-growing" algorithms like in NELL?

Application to stylized version of NELL-type algorithm

- Assume have algs $A_1, A_2, \ldots, A_L$ for each category.
- From a few labeled pts get $h_i^0 \subseteq c_i^*$ s.t. $\Pr(h_i^*) \geq \alpha$.

- Given $h_i \subseteq c_i^*$, alg A_i can produce k proposals for adding $\geq \alpha$ probability mass to h_i.
- Each is either good or at least α-bad.

Analysis $\Rightarrow$ can use unlabeled data + ontology to identify good extension.

Open questions/directions: ontology-based learning

- Weaken assumptions needed on stylized iterative alg (e.g., allow extensions that are only "slightly bad").
- Allow ontology to be imperfect, allow more dependence - perhaps getting smooth tradeoff with labeled data requirements.
- Extend to learning of relations, more complex info about objects (in addition to category memberships).

Part 3: Lifelong Learning and Autoencoding

[Balcan-B-Vempala]

- What if you have a series of learning problems that share some commonalities?
  - Want to learn these commonalities as you progress through life in order to learn faster/better.
- What if you have a series of images and want to adaptively learn a good representation / autoencoder for these images?
- For today, just give one clean result.

Sparse Boolean Autoencoding

Say you have a set S of images represented as n-bit vectors.

   ...

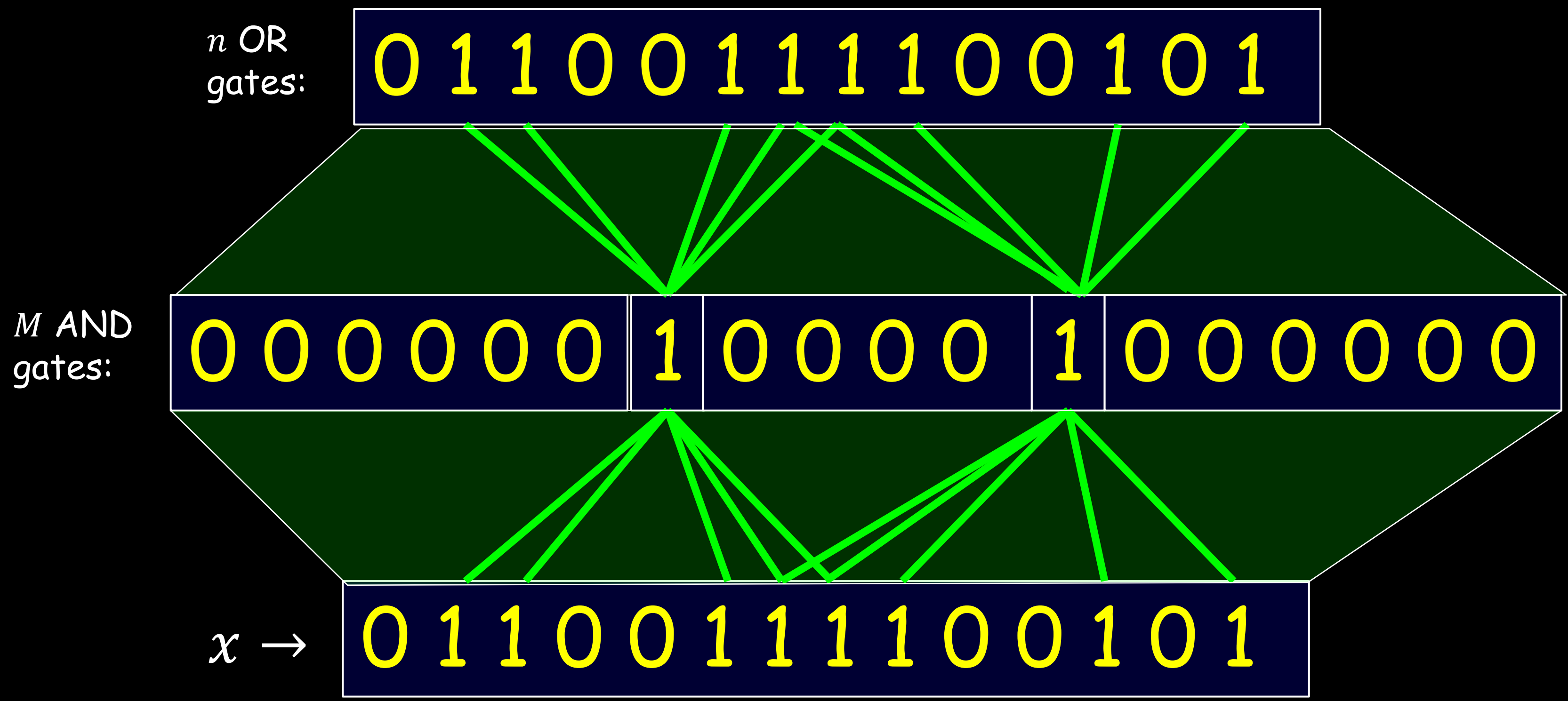
You want to find a "better", sparse representation.

- Goal is to find M meta-features $m_1, m_2, \ldots$ (also n-bit vectors) s.t. each given image can be reconstructed by superimposing (taking bitwise-OR) some k of them.
  - In fact, each $x \in S$ is reconstructed by taking bitwise-OR of all $m_j \preccurlyeq x$, and $|\{m_j \preccurlyeq x\}| \leq k$.

$m_j[i] \leq x[i]$ for all i.

Sparse Boolean Autoencoding

Equivalently, want a 2-layer AND-OR network, with M nodes in the middle level, s.t. each $x \in S$ is represented k-sparsely.



Trivial with $M = |S|$ (let's assume all $x \in S$ in middle slice) or $k = n$. Interesting is $k \ll n,\ M \ll |S|$.

Sparse Boolean Autoencoding

Equivalently, want a 2-layer AND-OR network, with M nodes in the middle level, s.t. each $x \in S$ is represented k-sparsely.

- Unfortunately even the case $k = M$ is NP-hard.
  "Set-basis problem".

- We'll make a "c-anchor set" assumption:
  - Assume exists M meta-features $m_1, m_2, \dots, m_M$ s.t. each $x \in S$ has a subset R_x, $|R_x| = k$, s.t. x is the bitwise-OR of the $m_j \in R_x$.
  - For each m_j, exists $y_j \preccurlyeq m_j$ of Hamming weight at most c, s.t. for all $x \in S$, if $y_j \preccurlyeq x$ then $m_j \in R_x$.

- Here, y_j is an "anchor set" for m_j: it identifies m_j at least for the images in S. An x that contains all bits in y_j has m_j in its relevant set.

Sparse Boolean Autoencoding

Now, under this condition, can get an efficient log-factor approximation.

- In time poly(n^c) can find a set of $O(M\log(n|S|))$ meta-features that satisfy the c-anchor-set assumption at sparsity-level $O(k\log(n|S|))$.

- Idea: first create candidate meta-features $\tilde{m}_y$ for each y of Hamming weight $\leq c$.
- Then set up LP to select. Variables $0 \leq Z_y \leq 1$ for each y and constraints:
  - For all x, i: $\sum_{y: e_i \preccurlyeq \tilde{m}_y \preccurlyeq x} Z_y \geq 1$ (each x is fractionally covered)
  - For all x: $\sum_{y: \tilde{m}_y \preccurlyeq x} Z_y \leq k$ (but not by more than k)
- And then round.

Extensions/Other Results

- *Online* Boolean autoencoding.
  - Examples are arriving online. Goal is to minimize number of "mistakes" where current set of meta-features is not sufficient to represent the new example.
- Learning related LTFs.
  - Want to learn a series of related LTFs.
  - Learn a good representation as we go.
  - Issue: haven't learned previous targets perfectly, so can make it harder to extract commonalities.
- Lots of nice open problems
  - E.g., for autoencoding, can assumptions be weakened? Approximation / mistake bounds be improved?

Conclusions

A lot of interesting new directions in practical ML that are in need to theoretical models and understanding, as well as fast algorithms.

This was a selection from my own perspective.

ML is an exciting field for theory, and there are places for all types of work (modeling, fast algorithms, structural results, connections,...)