Machine Learning: Social Values, Data Efficiency, and Beyond Prediction

Thesis Proposal

Travis Dick

Dec. 11, 2018

Thanks to my Thesis Committee!

Nina Balcan

Yishay Mansour

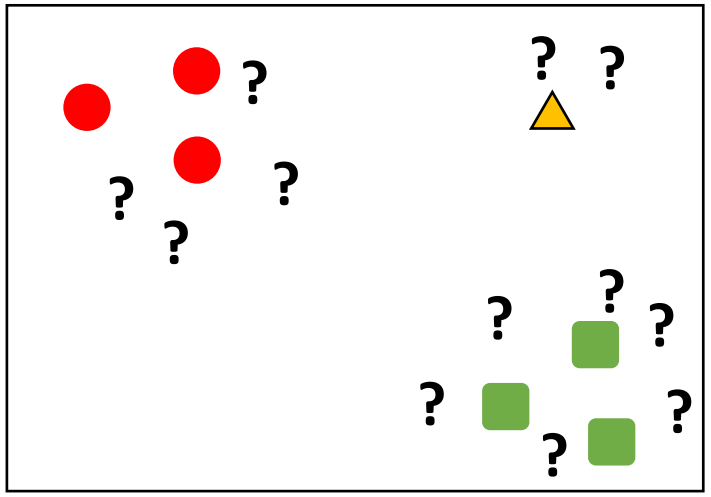
Tom Mitchell

Ariel Procaccia

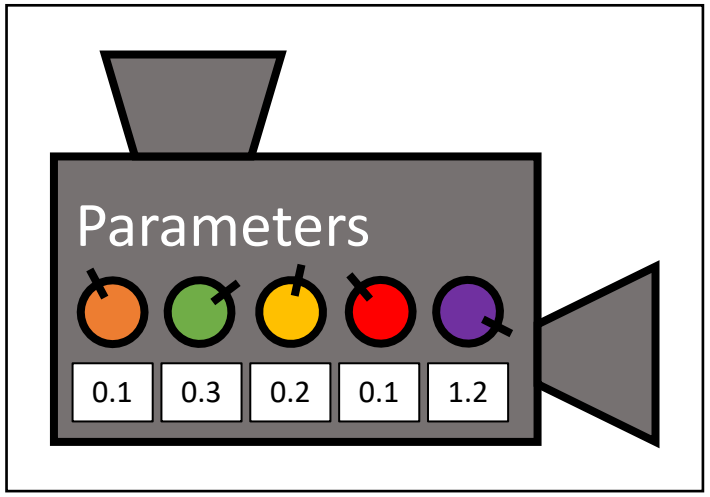
The Goal of this Thesis

Extend the theory and practice of machine learning to accommodate new requirements emerging from its use in real-world contexts.

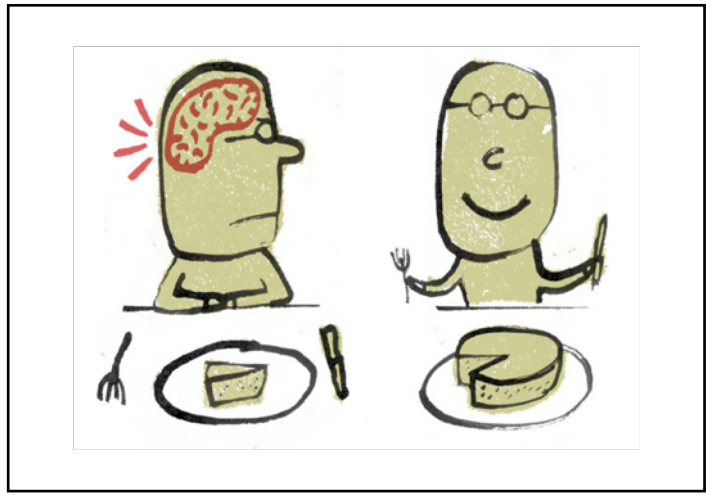
Data Efficiency



Beyond Prediction

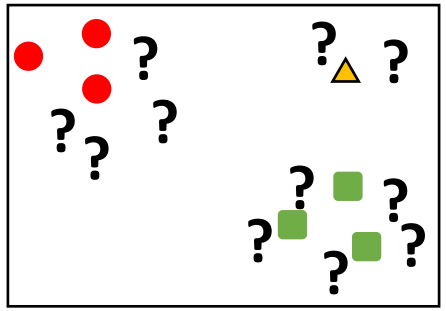


Social Values



The Goal of this Thesis

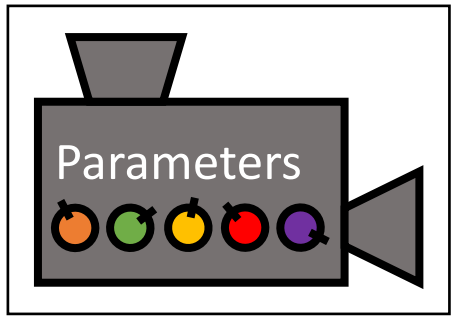
Data Efficiency



- There is disparity in the cost and availability of different types of data.
- Need to make the best use of cheap and abundant data.
- We focus on using unlabeled data in multi-class classification.

[BDM AAAI 2017]

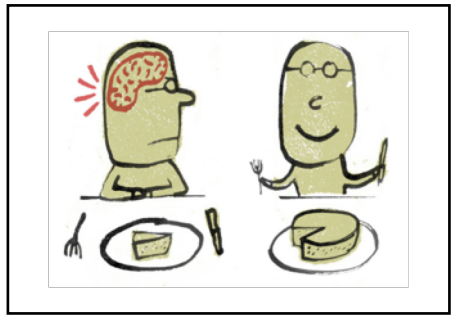
Beyond Prediction



- Many learners learn to make predictions (e.g., medical diagnosis).
- Important settings where learner's output is not a prediction rule.
- We'll look at algorithm configuration / parameter tuning.

[BDV FOCS 2018, BDV ICML 2018, BDW NeurIPS 2018]

Social Values



- Learning systems now regularly interact with us.
- Learning from personal data, making predictions about our behavior.
- We want these systems to uphold our social values (e.g. privacy, fairness)

[BDLMZ ICML 2017, Ongoing work with Nina, Ariel, Ritesh]

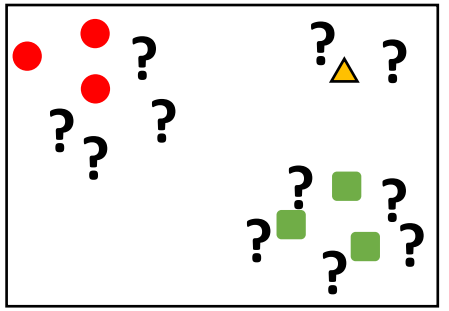
Outline

Three projects, each addressing at least one of the new requirements.

1. Label-efficient Learning in Multi-Class Problems

   Joint with Nina Balcan and Yishay Mansour

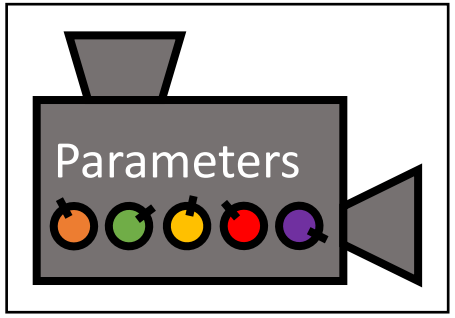
Data Efficiency



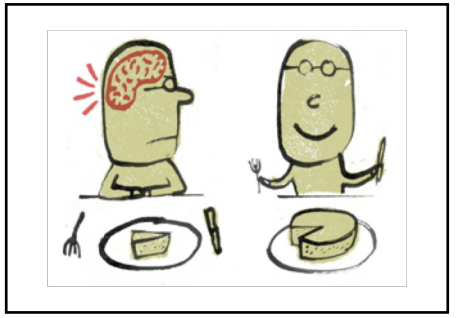
2. Online and Private Optimization of Piecewise Lipschitz Functions

   Joint with Nina Balcan and Ellen Vitercik

Beyond Prediction



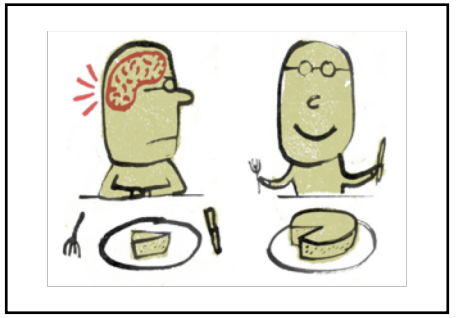
Social Values



3. Envy-free Classification

   Joint with Nina Balcan, Ariel Procaccia, Ritesh Noothigattu

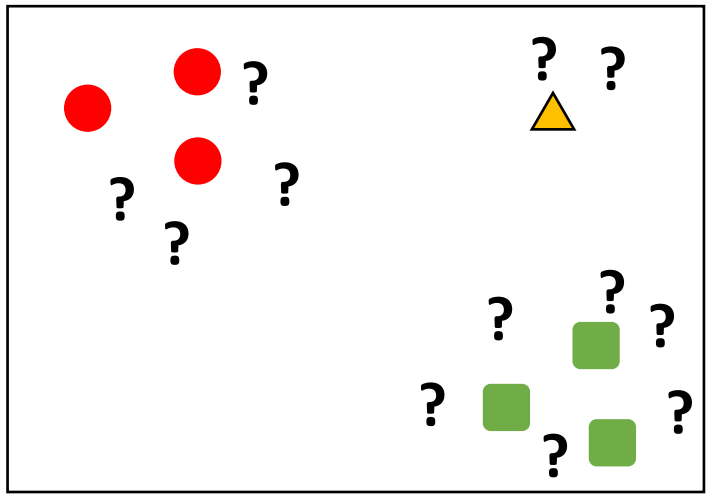
Social Values



Part 1:
Label-efficient Multi-class Learning

Joint work with Nina Balcan and Yishay Mansour [AAAI 2017]

Data Efficiency



Label-efficient Multi-class Learning

- Many modern learning problems take the form of multi-class prediction.

  E.g., Classifying images of animals

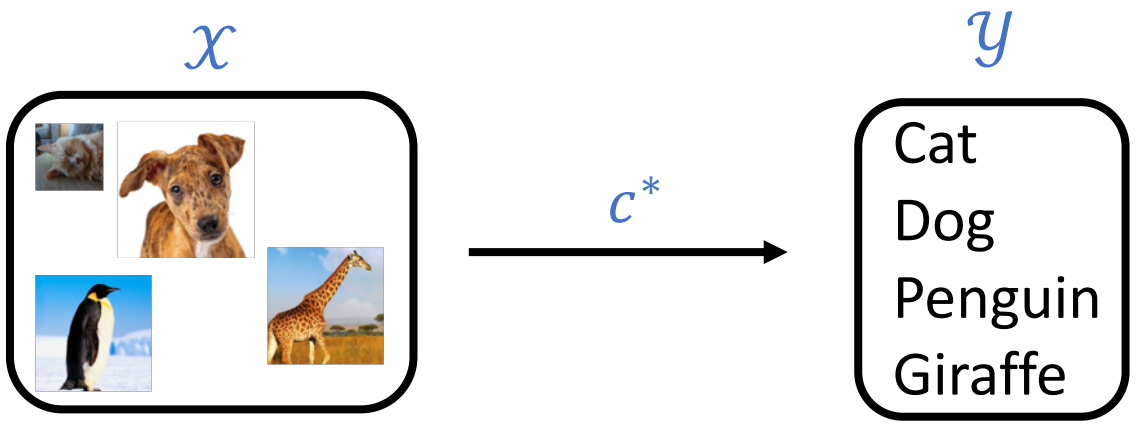
cat dog penguin girrafe

- Unlabeled data is cheap, labeled data is expensive.
- For example:
  - Can scrape the internet for images of animals.
  - Identifying animals in those labels is more expensive.

- More generally: we care about predicting labels because they are not readily available!
- **When can learning systems provably benefit from unlabeled data?**

Problem Setting

- Instance space $\mathcal{X}$ and label space $\mathcal{Y}$.
- Target concept $c^*: \mathcal{X} \to \mathcal{Y}$.
- Unknown distribution $\mathcal{P}$ over $\mathcal{X}$.



- Learner observes a large iid sample $x_1, \ldots, x_n \sim \mathcal{P}$.
- Learns about the target labels for a small subset.
  - (Semi-supervised) The target concept is queried on a random subset of examples.
  - (Active) The learner interactively chooses which examples to query.

What can we learn from unlabeled data?

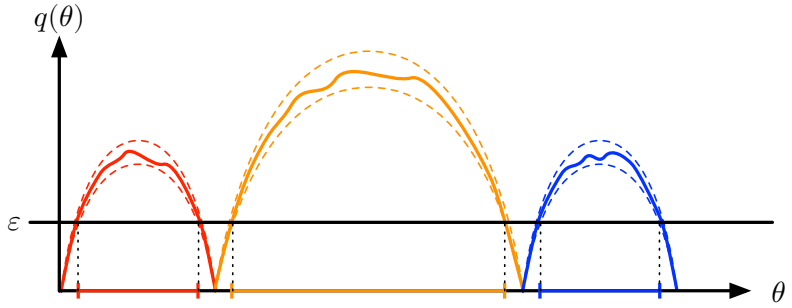
- An unlabeled sample tells us about $\mathcal{P}$.
- Need a connection between $\mathcal{P}$ and c^*!
- E.g., "c^* doesn't cut high-density regions"

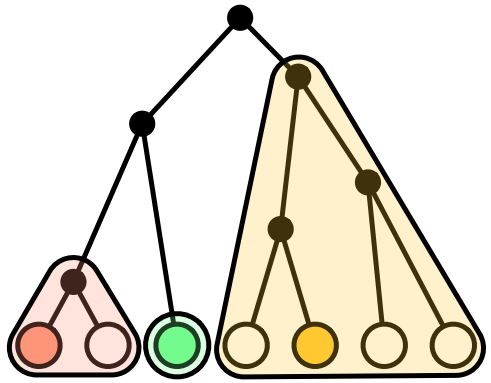
Our Approach

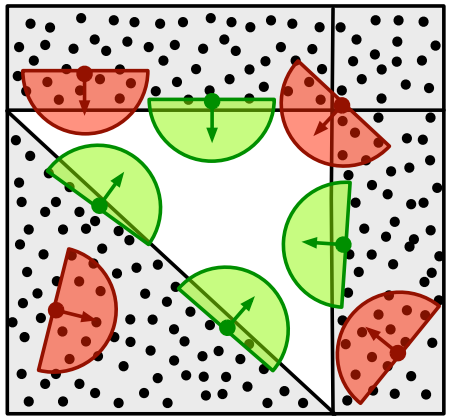
- Don't make explicit assumption on relationship between $\mathcal{P}$ and c^*.
  - Hard to verify.
  - Might not cleanly capture the real structure of the problem.

- Assume a specific **supervised** algorithm would succeed at learning from **fully labeled data.**
- Use that algorithm's **implicit** assumptions to design and analyze label-efficient learners.
- We focus on analyzing supervised **output codes**.

- Our label-efficient learners partition the unlabeled sample into label-homogeneous groups.
- Use a small number of labels to identify the label of each group.

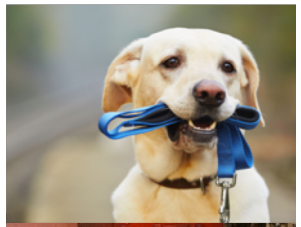
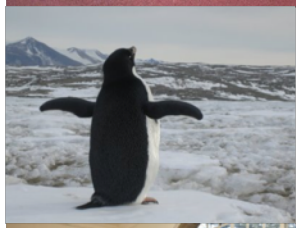






Output Codes

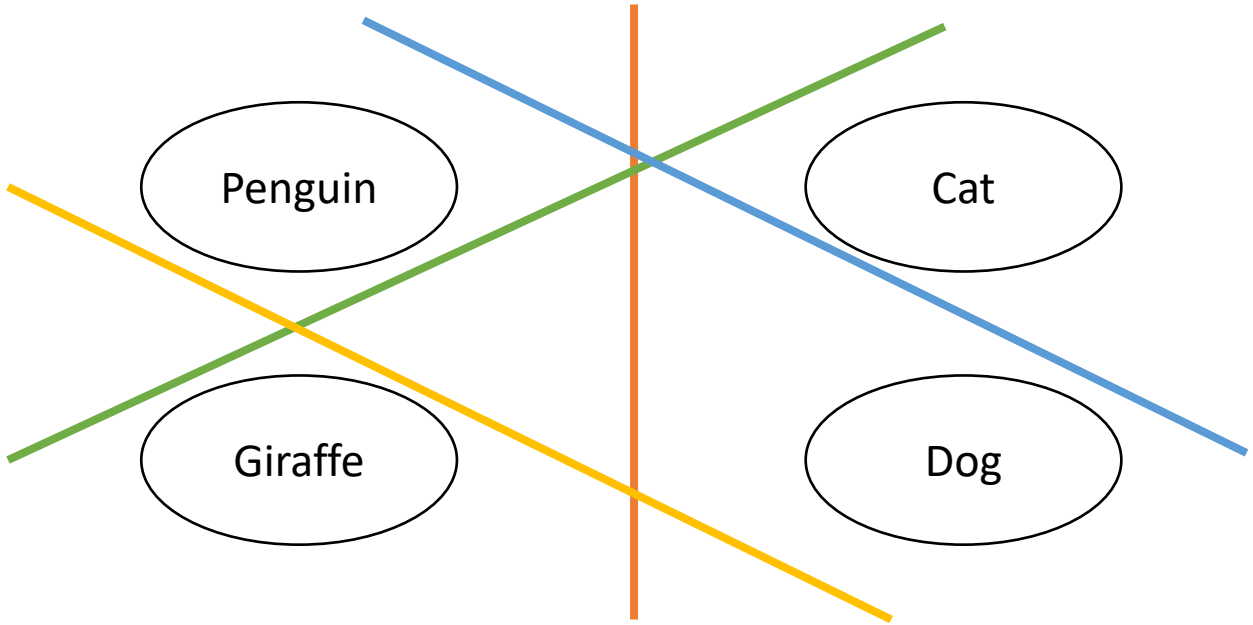
- Output codes are a general reduction from multi-class to binary classification.
- Design m binary partitions of the classes.
- Think of each partition as a *semantic feature*.

	Pet?	**Fur?**	**Long Neck?**	**Multiple lives?**
	yes	yes	no	no
	yes	yes	no	yes
	no	no	no	no
	no	yes	yes	no

Output Codes: Training and Prediction

- learn a binary classifier for each semantic feature.
- Result is $\hat{h}: \mathcal{X} \to \{\pm 1\}^m$ that predicts semantic features.

	Pet?	Fur?	Long neck?	Multiple lives?
cat	+1	+1	-1	+1
dog	+1	+1	-1	-1
penguin	-1	-1	-1	-1
giraffe	-1	+1	+1	-1



Prediction:

$$\hat{h}(\quad) = (-1, -1, -1, -1)$$

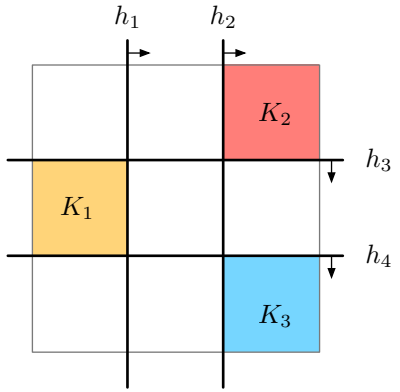
Pet? Fur? Long neck? Multiple lives?

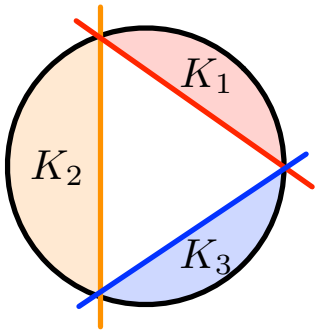
Flavor of our Results

- Assume that there exists a (nearly) consistent output code classifier with linear separators.
- Where the code matrix C has a bit of additional structure:
  - E.g. Large Hamming distance between rows.

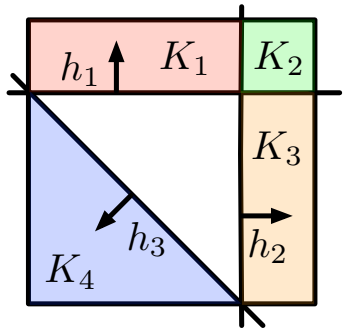
- Then we show that very label-efficient algorithms succeed at learning.
- Often only need one labeled example per class!

- Three cases:
  - Error correcting: large Hamming distance.





  - One-vs-all: C is the identity matrix



  - Boundary Features Condition

Error Correcting Codes

Dist ≥ 3

$$\begin{bmatrix} -1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 \\ +1 & +1 & +1 & +1 \end{bmatrix}$$

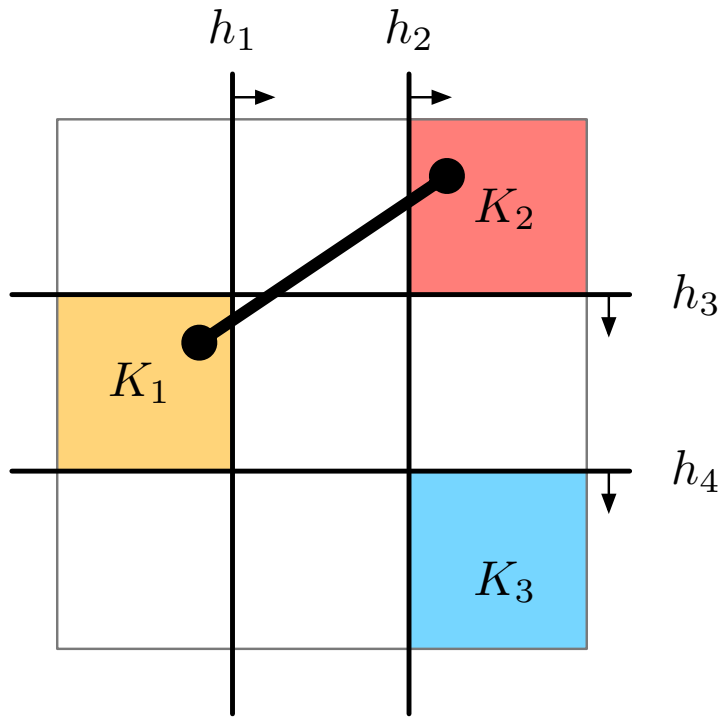
- Suppose that $\mathcal{X} \subset \mathbb{R}^d$ and codewords differ on at least $d + 1$ bits
- The consistent output code predicts every feature correctly (can relax).

Claim: If the learned linear separators are in general position, then there is a margin of size $g > 0$ between all pairs of classes.

Idea:

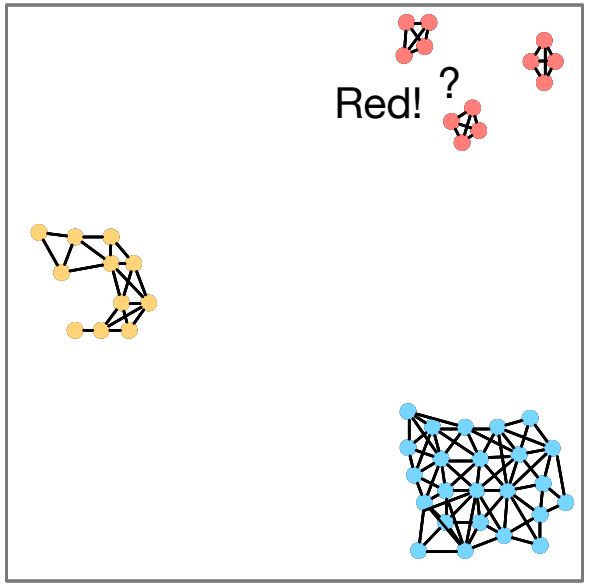
- The linear separators partition $\mathcal{X}$ into cells.
- Each cell has a codeword.
- The code matrix maps classes into those cells.
- The number of linear separators crossed by a line segment between two regions is exactly the number of bits their codewords disagree on.

- Must cross $d + 1$ linear separators to get from one class to another. But at most d can intersect at any point.

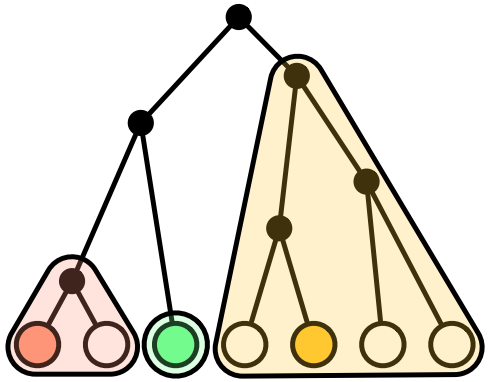


Clustering-based Algorithm

1. Run single linkage with connection radius $r_c < g$.
2. Use a small amount of labeled data to determine the label of each cluster.
3. To classify a new example, assign it to the same class as the nearest "sufficiently large cluster".



- Need conditions on $\mathcal{P}$ to guarantee we wont have too many large clusters.
  - $\mathcal{P}$ has a density p with "thick level sets".

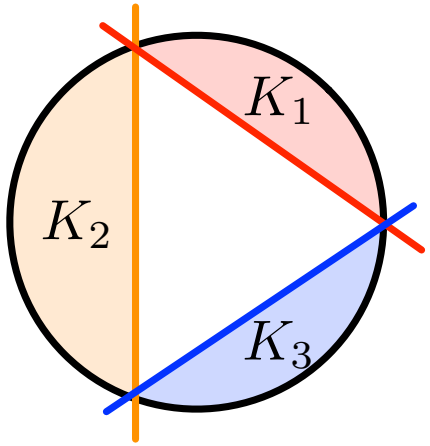


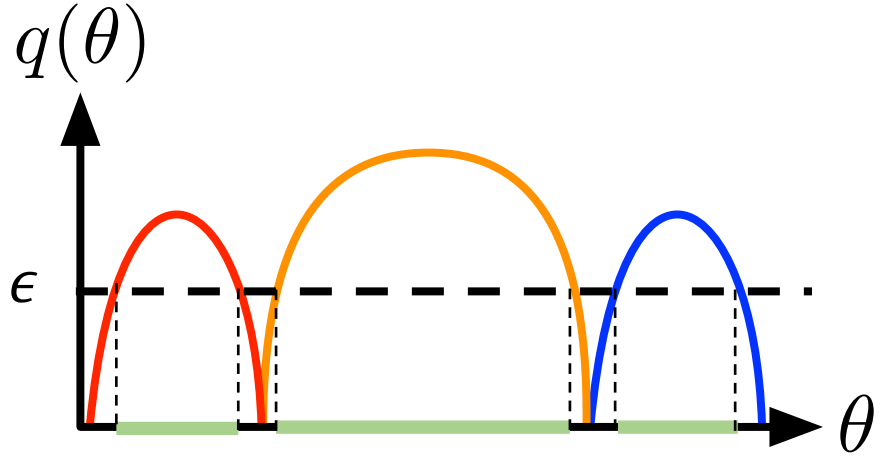
- Then number of required labels = natural number of clusters.

If we have an estimate of the number of clusters (instead of g), can run active algorithm.

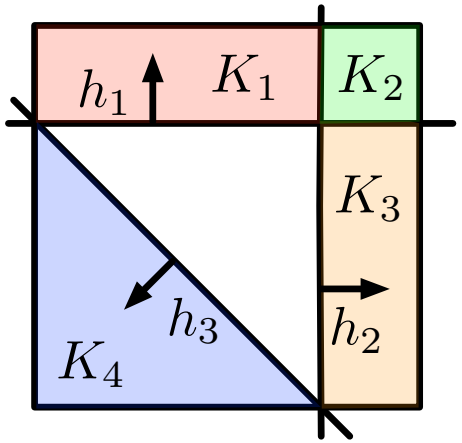
More results

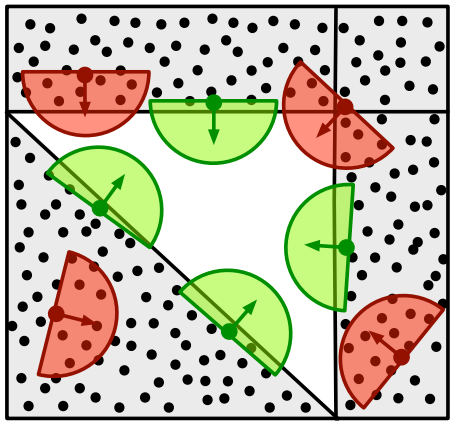
- Our other results significantly relax the large Hamming distance requirement.
- One-vs-all (Hamming distance = 2)





- Also some cases when Hamming distance is 1 (Clustering fails here!)

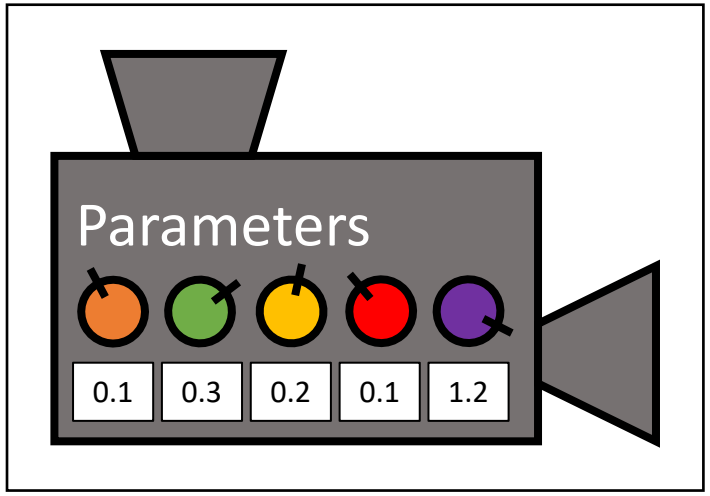




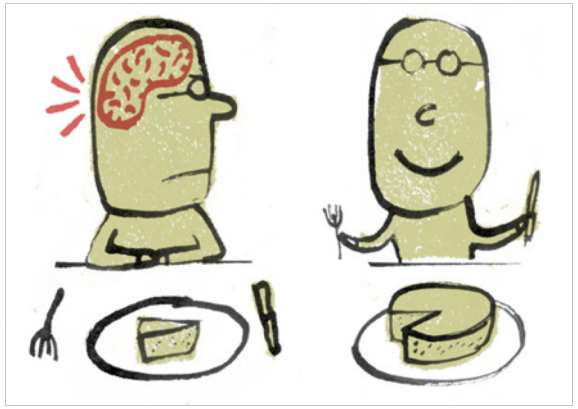
Part 2:
Piecewise Lipschitz Optimization

Joint work with Nina Balcan and Ellen Vitercik [FOCS 2018]

Beyond Prediction

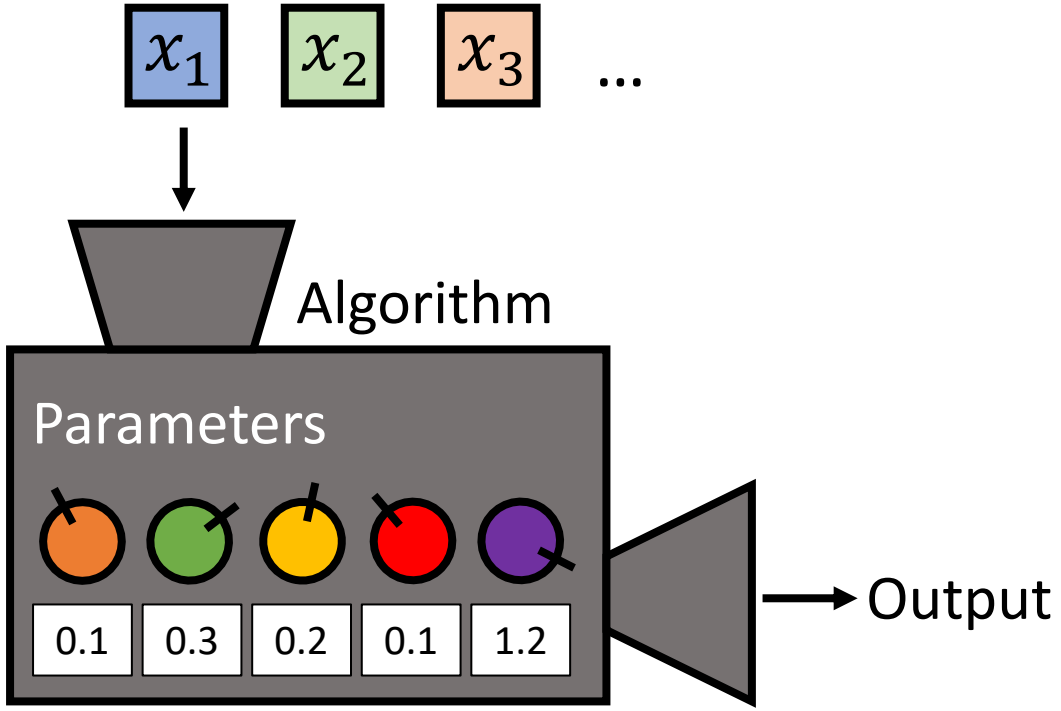


Social Values



Data-Driven Algorithm Configuration

Problem instances from specific application.



Goal:

- Automatically find the best parameters for a specific application domain.
- Algorithm is run repeatedly, historic instances are training data.
- Want provable guarantees for online and private settings.

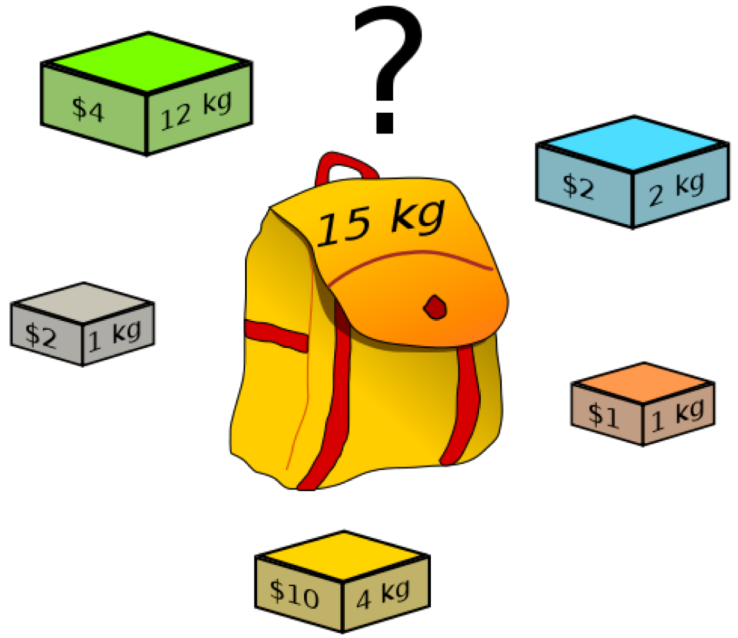
Example: Greedy Knapsack Algorithm

[Gupta & Roughgarden '16]

Problem Instance:

- Given n items
- item i has value v_i and size s_i
- a knapsack with capacity K

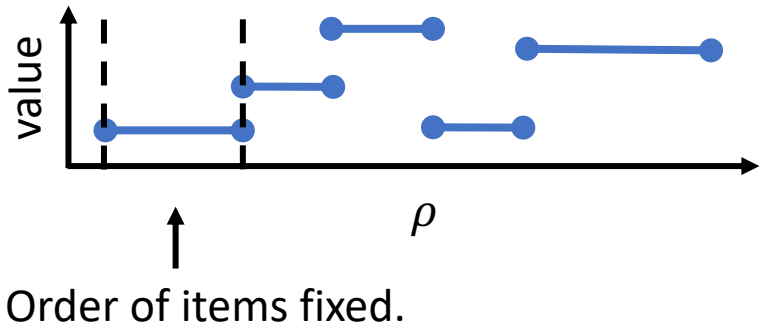
Find the most valuable subset of items that fits.



Algorithm: (Parameter $\rho \geq 0$)

Add items in decreasing order of $\text{score}_\rho(i) = v_i / s_i^\rho$.

Observation: For one instance, total value is piecewise constant in ρ.



Online Piecewise Lipschitz Optimization

More generally: utility is a piecewise Lipschitz function of parameters.

Learning protocol:

For each round $t = 1, \dots, T$:

1. Learner chooses point $\rho_t \in C \subset R^d$.
2. Adversary chooses piecewise L-Lipschitz function $u_t : C \to R$.
3. Learner gets reward $u_t(\rho_t)$
4. **Full information:** Learner observes entire function u_t
5. **Bandit information:** Learner only observes the scalar $u_t(\rho_t)$

Goal: Minimize regret $= \max_{\rho \in C} \sum_{t=1}^{T} u_t(\rho) - \sum_{t=1}^{T} u_t(\rho_t)$.

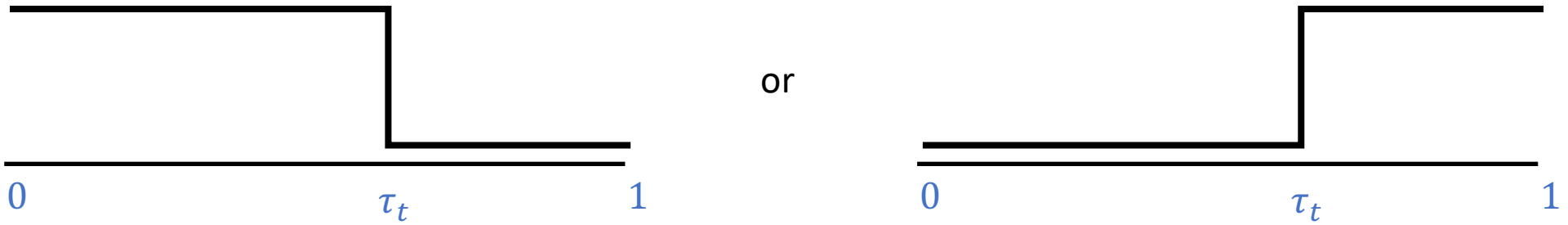
Meaningful Learning: Regret sublinear in T → optimal average per round utility

- *[Gupta & Roughgarden '16] have online algorithms for Max-Weight Independent Set with smoothed adversaries.*
- *[Cohen-Addad & Kanade '17] consider 1-dim. piecewise constant functions with smoothed adversaries.*

A Mean Adversary

Fact: There exists an adversary choosing piecewise constant functions from $[0,1]$ to $[0,1]$ such that **every** full information online algorithm has **linear regret**.

At round t, adversary chooses a threshold τ_t and flips a coin to choose either



Every learner has expected utility of ½ per round → expected total utility $T/2$.

Let $G_t = \{\rho \in [0,1] : u_s(\rho) = 1 \text{ for all } 1 \leq s \leq t\}$

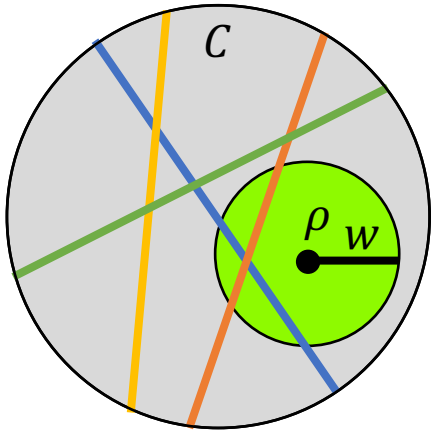
Set τ_t to be midpoint of G_{t-1} → $\max_T U_T(\rho) = T$.

Regret = $T/2$.

Dispersion

The mean adversary concentrated discontinuities near ρ^*. Even very near points had low utility!

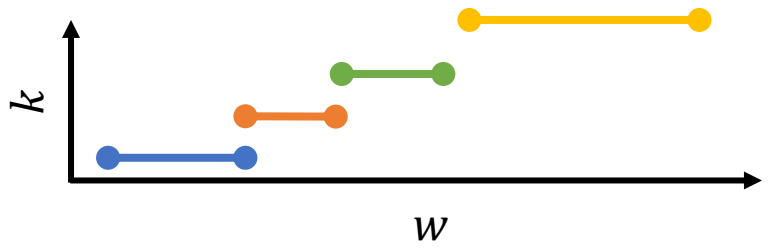
Def: Functions $u_1(\cdot), \ldots, u_T(\cdot)$ are $(\boldsymbol{w}, \boldsymbol{k})$-***dispersed at point*** $\boldsymbol{\rho}$ if the ℓ_2-ball $B(\rho, w)$ contains discontinuities for at most k of $u_1 \ldots, u_T$.



Each colored line is a discontinuity of one function.

Ball of radius w about ρ contains 2 discontinuities.
→ $(w, 2)$-dispersed.

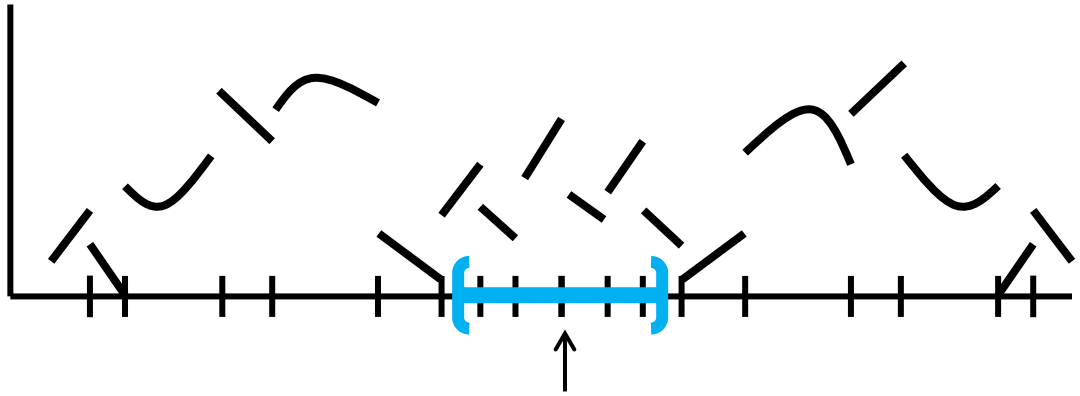
Functions will satisfy a range of dispersion parameters:



The Sum of Disperse Functions

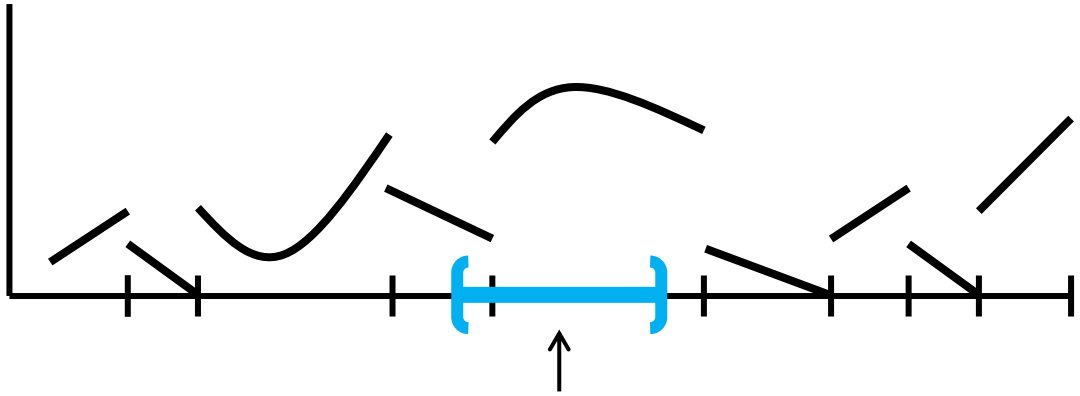
Let $u_1, \ldots, u_T$ be PWL functions and plot their sum $\sum_t u_t$

Not disperse



Many boundaries within interval

Disperse



Few boundaries within interval

Full Information Regret Bounds

We analyze the classic Exponentially Weighted Forecaster *[Cesa-Bianchi & Lugosi '06]*

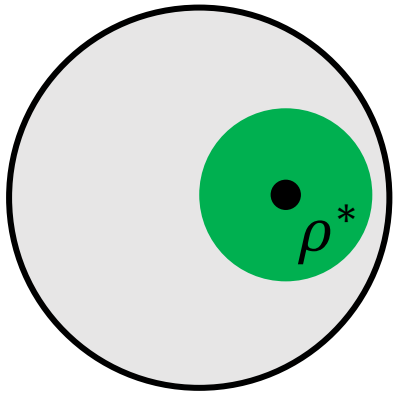
Algorithm: (Parameter $\lambda > 0$)
At round t, sample ρ_t from $p_t(\rho) \propto \exp(\lambda \sum_{s=1}^{t-1} u_s(\rho))$.

Theorem: If $u_1, \dots, u_T : C \to [0,1]$ are piecewise L-Lipschitz and (w, k)-dispersed at ρ^*,

EWF has regret $O\left(\sqrt{Td \log \frac{1}{w}} + TLw + k\right)$.

When is this a good bound? For $w = 1/(L\sqrt{T})$ and $k = \tilde{O}(\sqrt{T})$ regret is $\tilde{O}(\sqrt{Td})$.

(We essentially achieve these values in all of our applications)

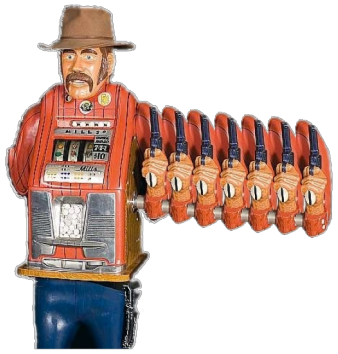


Intuition:

- The ball $B(\rho^*, w)$ has utility at least $OPT - TLw - k$.
- EWF can compete with $B(\rho^*, w)$ when its volume is non-negligible.

Note: Don't need to know (w, k) to run algorithm.

More Results from Dispersion



Bandit online optimization of piecewise Lipschitz functions:

- Learner only observes the scalar $u_t(\rho_t)$ each round.
- Dispersion with $w \approx 1/T^{1/(d+2)}$ and $k \approx T^{\frac{d+1}{d+2}}$ implies regret $\tilde{O}\left(T^{\frac{d+1}{d+2}}\left(\sqrt{d3^d}+L\right)\right)$.

Differentially Private Batch Optimization

- Given $u_1, \ldots, u_T$ up-front, estimate maximizer of $\frac{1}{T}\sum_t u_t$.
- Satisfy (ϵ, δ)-differential privacy (w.r.t. changing any one function).
- Exponential mechanism has suboptimality $\tilde{O}\left(\frac{1}{T\epsilon} d \log\frac{1}{w} + Lw + \frac{k}{T}\right)$.
- Matching lower bounds.

Verify Dispersion for Algorithm Configuration

- Greedy algorithms for knapsack and max-weight independent set.
- SDP rounding schemes for integer quadratic programs.

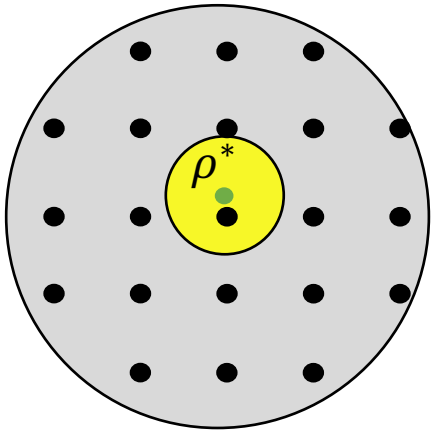
Future Work

Tidying Up and Easier Tools

- Current definition is a bit complicated with many "gotchas".
- Working on getting crisp and intuitive definitions and better tools for verifying dispersion in practice.
- Plan to submit to COLT 2019.

Better (Semi)Bandit Algorithms

- Current algorithm discretizes $\mathcal{C}$ using a w-net.
  - Exponential running time in the dimension.
  - Does not adapt to optimal (w, k)-parameters.
- Exploit partial feedback (often observe an entire piece of u_t)



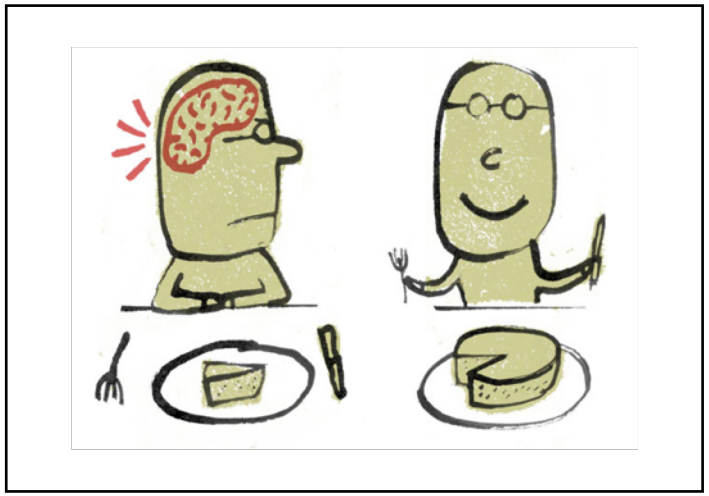
More Algorithm Configuration

- Derive dispersion for new algorithms (e.g., Initialization of Lloyd's method)
- Empirically evaluate performance of tuning algorithms.

Part 3:
Envy-free Classification

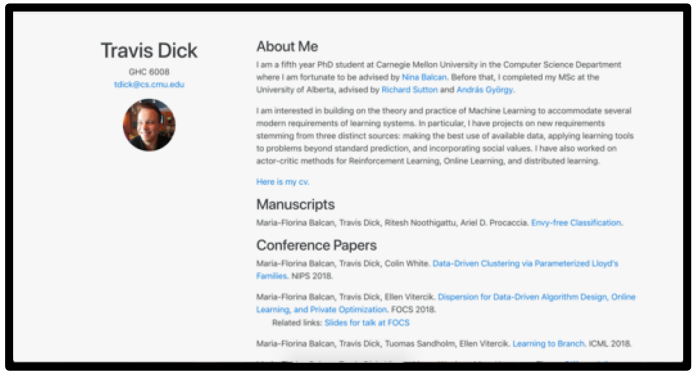
Joint work with Nina Balcan, Ariel Procaccia, and Ritesh Noothigattu [Ongoing]

Social Values



Problem Setting

- We want to use ML to learn to assign outcomes to individuals.
- For example, deciding which advertisements to show in order to maximize revenue.



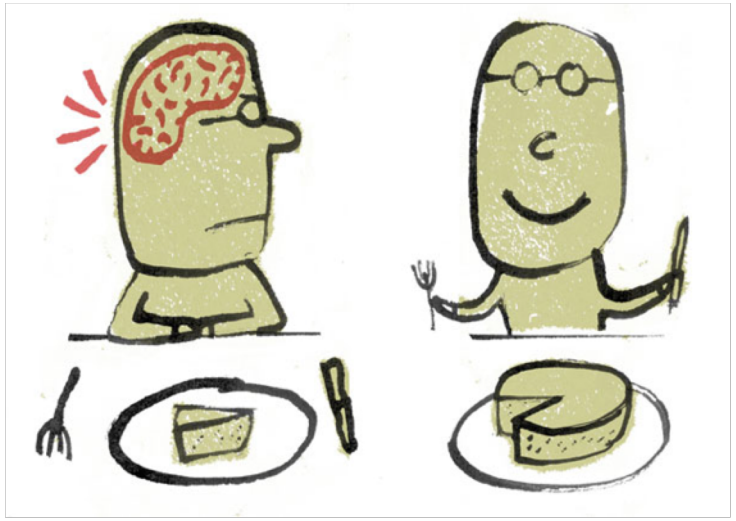
- Assignment rule is a classifier $h: \mathcal{X} \to \mathcal{Y}$.
- Two objectives:
  1. Minimize expected loss (e.g., maximize revenue with advertisements)
  2. Treat people fairly (what does this mean?)

- We import the notion of **envy freeness** from fair division.
  - Suitable when there are many outcomes and heterogeneous preferences.
  - Provides fairness guarantees to individuals, rather than groups.

Envy-freeness

Envy-freeness is the gold standard of fairness for fair division.

E.g.



cake cutting

chore assignment

Idea: Assignment is envy free if no one prefers the assignment of anyone else over their own.

Envy-free Classifiers

($\Delta(\mathcal{Y})$ is the set of distributions on $\mathcal{Y}$)

- Given a utility function $u: \mathcal{X} \times \mathcal{Y} \to [0,1]$ measuring individual preferences.
- A classifier $h: \mathcal{X} \to \Delta(\mathcal{Y})$ is (α, β)-envy free with respect to u and $\mathcal{P}$ if:

$$\Pr_{x,x' \sim \mathcal{P}}\big(u\big(x, h(x)\big) < u\big(x, h(x')\big) - \beta\big) \leq \alpha\,.$$

- $\alpha = \beta = 0$: every individual likes their assignment as much as anyone else's.
- $\alpha > 0$: A small number of pairs of individuals are allowed to be envious.
- $\beta > 0$: A small amount of envy is tolerated.

Example: Two individuals and two outcomes

Utilities:

		
	1	0
	0	1

- $(0,0)$-EF requires $h($  $,$  $) \geq h($$,$$)$ and $h($$,$  $) \geq h($  $,$$)$.

Randomized Classifiers

- Envy-freeness is a very strong requirement on deterministic classifiers:

Lem: Every *deterministic* EF map is of the form $h(x) = \text{argmax}_{y \in Y'} u(x, y)$ where Y' is a subset of the classes Y (breaking ties arbitrarily).

- We consider randomized classifiers (that assign distributions to individuals) since they have a lot more freedom.
- Better tradeoffs between fairness and accuracy.
  - Can construct examples where best randomized EF classifier has arbitrary multiplicative improvement over best deterministic EF classifier.

Main Question: Can we learn EF classifiers?

- If a classifier is EF on a sample of individuals, does this imply that it will remain EF on the underlying distribution?

- Note: every fairness notion in machine learning should generalize, otherwise we have no way of finding fair classifiers from data.

- Following classic learning theory results, we assume the learner chooses an assignment rule from some class $\mathcal{H} = \{h: \mathcal{X} \to \Delta(\mathcal{Y})\}$ of *low complexity*.

- Main challenge: no known complexity measure applicable for distribution-valued outputs.

  - VC-dimension works for binary outputs.
  - Pseudo-dimension and Rademacher complexity work for a single real value output.
  - Natarajan dimension works for multi-class outputs.

Warmup Generalization Result

Lem: Let $\mathcal{H} = \{h: X \to \Delta(Y)\}$ be *finite*. If $(x_1, x_1'), \ldots, (x_n, x_n')$ is an i.i.d. sample of pairs from P^2 of size $n = O\left(\frac{\log(|H|/\delta)}{\gamma^2}\right)$, then w.p. $\geq 1 - \delta$ every h in $\mathcal{H}$ that is (α, β)-EF on the sample is $(\alpha + \gamma, \beta)$-EF on $\mathcal{P}$.

Idea:

- Use Hoeffding's inequality to get convergence for an appropriate "fairness loss" for each classifier.
- Union bound over the finite set.

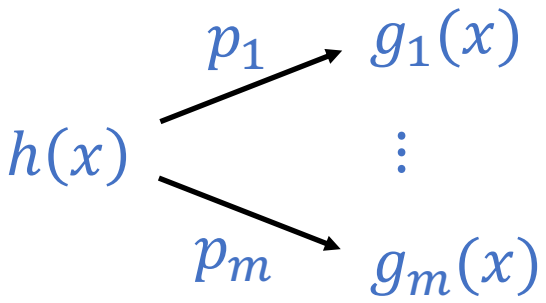
- For infinite $|\mathcal{H}|$, should expect to replace $\log|\mathcal{H}|$ by some complexity measure.

Random Mixtures of Deterministic Classifiers

- No existing natural notion of complexity for distribution-valued outputs…
- **Idea:** Many randomized classifiers can be expressed as mixtures of deterministic classifiers.

Def: A class $H = \{h: X \to \Delta(Y)\}$ is an $\boldsymbol{m}$**-mixture** of a deterministic class $G = \{g: X \to Y\}$ if for each $h \in H$ there exists m functions $g_1, \dots, g_m \in G$ and mixing weights $p_1, \dots, p_m \in \Delta_m$ such that $\Pr(h(x) = y) = \sum_i p_i 1\{g_i(x) = y\}$.

h randomly picks g_i with probability p_i and outputs its classification of x.



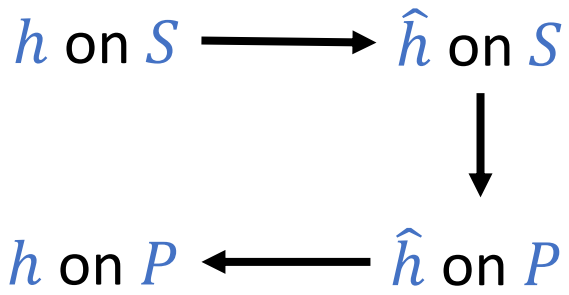
If G has low Natarajan dimension, and H is an m-mixture of G, then we should expect envy-freeness to generalize for H.

Envy-Freeness Generalizes for Random Mixtures

Thm: Let $H = \{h: X \to \Delta(Y)\}$ be an m-mixture of $G = \{g: X \to Y\}$ and suppose the Natarajan dimension of G is d. If $S = \{(x_1, x_1'), \ldots, (x_n, x_n')\}$ is an iid sample of $n = \tilde{O}\left(\frac{m^2 d}{\gamma^2}\right)$ pairs from P, then w.h.p. every $h \in H$ that is (α, β)-EF on S will be $(\alpha + 7\gamma, \beta + 4\gamma)$-EF on P.

Idea:

- Approximate H by a finite class $\widehat{H}$ (the size will depend on m and d).
- For each h, let $\hat{h}$ be its approximation in $\widehat{H}$.
- The classifiers h and $\hat{h}$ behave similarly on P and sufficiently large samples.



- If h is EF on S then $\hat{h}$ is too.
- $\hat{h}$ belongs to a finite class, so $\hat{h}$ is also EF on P.
- If $\hat{h}$ is EF on P, then h is too.

EF parameters degrade along each edge.

Future Work

Improved Dependence on m:

- Current generalization bound is $\tilde{O}\left(\frac{dm^2}{\gamma^2}\right)$, but it should be $\tilde{O}\left(\frac{d}{\gamma^2}\right)$.

ERM Algorithms:

- Our current results show that envy-freeness generalizes from a sample.
- Need efficient algorithms that find envy-free classifiers on a sample.

- If we just want assignments for the individuals in the sample, then the problem is a linear program.
- Unfortunately, we show that generalization for any extension of a sample assignment like this requires exponentially large samples.
- Goal: Find a mixture of deterministic classifiers that has low loss and is envy-free.

New Fairness Notions:

- Explore more borrowed notions of fairness from fair division (e.g., for public goods)
- Particularly interested in problems where we learn to rank items (e.g., search results)

Timeline

- Now – End of February:
  - Focus on dispersion and algorithm configuration.
  - Experiments finished by end of January.
  - Plan to submit paper to COLT 2019 (Feb. 1 Deadline)

- March – April:
  - Focus on social values.
  - Algorithms and better bounds.
  - Exploring new notions of fairness.

- April - May:
  - Wrap up, writing, defense.

Thanks!