Contents

1 Introduction (Sanjoy Dasgupta)

I'll talk about 5 areas which need foundational work.

1. Teaching

2. Explanations and interpretations

3. Unsupervised++

4. Imitation

5. Semantic communication

These areas all have one feature in common: Cooperation between agents of different types, that don't know each other's insides (ex. machine and human).

1.1 Teaching

There is a spectrum of types of examples: adversarial (in online learning), random (in statistical learning), benign (in teaching). Does sample complexity dramatically improve? People have converged upon a particular model that is influential but also broken.

What is the minimum set of labeled examples needed to uniquely identify the target concept? It's a kind of description dimension. This is in relation to a specific concept class C.

Definition 1.1: Let C be a concept class and $h \in C$ be a target. $TD(h, C)$ is the smallest set of labeled instances for which h is the only consistent concept in C.

We can define $TD(C) = \max_h TD(h, C)$, or $\mathbb{E}_h TD(h, C)$.
This is geared towards finite concept classes.
This is broken because of the following problems.

1. It assumes the teacher knows the representation of the learner and the learner's concept class. Examples are tuned to the concept class.
2. The problem of selecting the teaching set can be NP-hard.
3. The predictions it gives for ideal teaching sets are ridiculous, ex. cat that looks like dog and dog that looks like cat. In practice people select examples that are far apart (the canonical cat/dog).
4. This only works for the realizable case.

What to do? What are better teaching models?

1. Who is teaching who? Human/machine teaching human/machine?
   (a) Human-human: education/cognitive science/childhood development
   (b) Human-machine: machine learning
   (c) Machine-human: intelligent tutoring
   (d) Machine-machine:
   cf. cotraining, two machines bootstrapping each other.
   GAN's teach each other to generate/discriminate (with opposite goals).
2. Avoid assuming the teacher knows the learner's represntation and concept class.
3. Interactivity: The machine could ask questions.

   Any semi-realistic model would be interactive.
4. Come up with models that gives far apart examples (Jerry Zhu).

   Ex. Let's say the learner does nearest neighbor. Suppose it is noisy nearest-neighbor. That pulls you apart.

5. Curriculum learning and self-paced learning strategies. Hierarchical learning. Simple things are learned first; then you add things.

6. Other kinds of tasks besides classification, ex. generative.

7. Restrict to an interesting domain like language.

The teacher does not have unbounded computation, but knows the concept and has a storehouse of examples gleaned from experience.

1.2 Explanations and interpretations

Ex. more than just saying you like a movie, say that you like a specific actor.

Ex. in computer vision In addition to giving a label (ex. zebra, antelope), give one-word explanations (stripes, antlers). Learn classifiers for these intermediate features as well. This taps into a potentially infinite latent space.

When feature space is high dimensional, this helps.

1. Models of explanation-based learning.

   What are the benefits of explanation-based learning.

2. Interpretable classifiers (transparency in ML).

   Output a hypothesis that scientists can understand.

   Accompany predictions with explanations. "Your loan was rejected because..."

   Decision trees used to do this automatically until people realized random forests do better.

   Use explanations to generate interpretable classifiers?

   Ex. sparse classifiers: give the support of which features you used.

1.3 Unsupervised learning++

Ex. topic modeling. Some are good, some are sliced/diced in various ways, some are garbage. Running the Gibbs sampler for longer doesn't solve the problem. This is ripe for interactive feedback of some type.

Sometimes you just literally need feedback; we want to quantify how much feedback is needed.

What type of interaction is useful algorithmically and for the human?

It could be relationships between data points, constraints based on features, etc. A practitioner would choose the algorithm and the form of interaction.

(Q: how to avoid tricks such as: To transmit finite automaton, grammar, write down grammar or automaton. Make an arbitrary convention of how to translate examples into a grammar.)

1. Improving unsupervised learning with interaction

(a) Modes of interaction

(b) How much interaction?

Normally use Euclidean distance. What you want to use if people's subjective similarity scores?

2. Generalization theory for unsupervised learning

"Unsupervised learning++=Supervised learning–": Unsupervised learning is talked about a lot as lossy compression. Here, you don't have exactly the label you want, but have something that's associated with what you want. This is unsupervised++ because one can imagine this built on top of unsupervised learning algorithms.

The only results I'm aware of are nonstatistical results: ex. for clustering points, query $n \ln n$ distances rather than $\binom{n}{2}$. There should not be any n here at all, just ε and the distribution.

1.4 Imitation learning (and teaching)

One or two decades we'll be telling our domestic robots "this is how we like to make our coffee."

Imitation learning seems a tractable case of reinforcement learning. Imitation implicitly assumes a sequence of actions.

It's not enough to explain why we did this; we have to explain things we don't want to do? Littman, NIPS had a formalization.

1.5 Semantic communication

1. One paper was by Juba, Sudan, Goldreich. Ex. You don't know the language. What protocol can you execute? The answer is disappointing: try everything.

2. Percy Liang: A computer is in charge of blocks. You want to move the blocks to a specific configuration by telling the computer what to do.

   Throw in 2 constraints: compositionality of language, pragmatics (different utterances probably mean different things).

   (Pragmatics is like dropout: it helps but is not crucial. There's other things going on, which NLP takes for granted but would be interesting for theorists: ex. loglinear model.)

1.6 Other topics

1. Language learning and generation

2. Crowdsourcing. Designing proper crowdsourcing experiments. How do we learn from weak teachers (Amazon Turkers) that make errors?

   See work by Nihar Shar, Kevin Jameson (Next, with Robert Nowak).

   `http://nextml.org/`

2 Two Paradigms of Semi-Supervised Active Clustering with Sample and Computational Complexity Bounds (Shai Ben-David, University of Waterloo)

I want to cover three things.

1. Introduction/preaching: what should clustering theory do?
2. Semi-supervised active clustering (joint work with Hassan Ashtiani and Shrinu Kushagra) [AKB16] `https://arxiv.org/abs/1606.02404`
3. Learning representations from clustering (Hassan) [AB15] `https://arxiv.org/abs/1506.05900`

2.1 Introduction

Different clustering algorithms yield very different outcomes. Contrast this with other computational tasks, like classification, for which different algorithms roughly give the same results.

Thus the choice of a clustering algorithm is very important. They have different objectives which cannot be satisfied simultaneously. Things we want may include:

1. Similar points are clustered together
2. Dissimilar points are in different clusters. (Already these may conflict—ex. a long chain of points)
3. Balanced cluster sizes
4. Stable under perturbation

Where in this simplex of properties do we want?

- Single linkage only cares about similar points.
- Max linkage (postpone merging far points) only cares about dissimilar points.
- k-means cares about balancing cluster sizes.

Different applications have different priorities. One application is record de-duplication: cluster together records that are the same. Here we would want max linkage: we want dissimilar records not to be identified. For viral spread, we want to group similar points together.

There is no universally optimal clustering. There is a need for domain-specific biases.

The first question is what algorithm to use; there is not enough research in this direction. It's not clear what tool is good for that. Most people use k-means without thinking. "Because everyone else is using this algorithm."

We can address the problem in two ways:

- I give similarity of pairs. Find the right clustering algorithm from that.
- Here is the tool I'm using, but I will play with the similarity.

We focus on the second approach. There are ways of asking for input from the user that is more intuitive than asking them to choose the objective—ex. ask them how to cluster a small sample of points.

2.2 Semi-supervised active clustering

See NIPS2016.

Here is the setup.

- We have unknown clustering $C_1, \ldots, C_K$ of (X, d) where d is known.
- The algorithm can interact with an oracle that knows $C_1, \ldots, C_K$.
- The type of interaction is queries of the form are x_1, x_2 in the same cluster or different clusters.

The algorithm computes, then decides which points to ask about next, etc.

If the user already knows, why do you need the clustering algorithm? There are so many data points; you don't want to ask n^2 queries. You need the algorithm to overcome the enormity of the data. There are specific applications for which this model is relevant, ex. data de-duplication. Here the number of clusters is large compared to the number of records.

The number of queries needed is logarithmic in the dataset.

I show the result that under some assumptions on the target clustering, there exists an algorithm that requires $O(k \ln n)$ queries and finds the target clustering in linear time $O(kn(\text{dim}))$, where dim is the dimension.

Without the queries the task is NP-hard. After log queries, the task collapses to linear time!

We assume that answers are noise-free.

We don't need to know k in advance. Hardness results kick in for k a function of n.

Trivially, kn queries always suffice. Find one representative in every cluster. For every new point, ask whether it's in the same cluster with every representative.

Here are the assumptions.

1. Center-based clustering. Every point belongs to the nearest center (Voronoi cell).
2. μ_i, the center of C_i, is $\mathbb{E}_{x \sim C_i} f(x)$, for some known f. This definitely holds for k-means.
3. Niceness (clusterability) assumption: γ-margin.

Definition 2.1: $C_1, \ldots, C_k$ satisfies the γ-margin condition if $\forall i, \forall x \in C_i, \forall y \notin C_i$,

$$d(x, \mu_i)(1 + \gamma) \le d(y, \beta_i).$$

Around every cluster there is an empty margin.

What do we know about k-means with such an assumption?

Theorem 2.2 (Hardness result)**.** *k-means is NP-hard under γ-margin condition as long as $\gamma \le 0.84$. (Euclidean distance) In general, it is NP-hard for $\gamma \le 1$.*

Here $k \sim n^{\varepsilon}$. The reduction is from set cover.

Although these conditions look strong, it is still hard to do k-means under these assumptions.

Positive result: For $\gamma > 2$, k-means without queries is feasible. Use single linkage and use dynamic programming to search over all prunings.

[1]

Usually when people find good clustering algorithms, it's with strong assumptions. The task becomes hard before the condition becomes natural.

The algorithm with queries:

1. Ask enough queries to get "many" (N) points in one cluster. Ask Nk queries.

   (Once you have representatives, you can compare, and we can ask which cluster you belong to.)

2. Pick a cluster with enough points and estimate its center. (This estimate is good by Chernoff.) The γ tells me how many points I need in my cluster to be able to tell whether points are in that cluster.

3. Binary search to find the cluster radius. Ask whether a point of some distance away is in the cluster. Need $\ln n$ queries.

4. Delete points in this cluster and repeat.

2.3 Learning representations from clustering

This is a different type of interaction. Hand the user a small subset $S \subseteq X$ and ask to get back their desired clustering of S.

What can I learn/generalize from this?

Based on this $S_1, \ldots, S_k$, how can we pick a suitable clustering tool for X? Note that maybe not all clusters are represented. (Otherwise, we can think of it as a classification problem.)

[1] AWigderson: Could you find a clustering that uses $k \ln n$ centers greedily? Probably.
Find too many clusters and use queries to prune?

The idea is that what we want to learn is the metric. Instead of picking between different clustering algorithms, fix the clustering algorithm (regularized k-means) and learn the metric.

We fix a family of embeddings of X into some $\mathbb{R}^d$ (a family of kernels over X), F.

Now we can phrase the problem more precisely: the algorithmic task is to find $f \in F$ such that $A(f(X))|_S$ is close to $S_1, \ldots, S_k$. This is a well-defined problem; we can talk about sample and computational complexity.

This is not an ideal solution because it could be that the number of clusters could be much larger. The user can also give clusterings that are inconsistent when given a small vs. large set.

For sample complexity analysis of ERM algorithms in the model, we need a notion of distance between clusterings, and a notion of complexity of F.

The distance is

$$D((C_1, \ldots, C_k), (C'_1, \ldots, C'_k)) = \min_{\pi \in S_k} \frac{1}{|X|} \sum_{i=1}^{k} |C_i \triangle C'_{\pi(i)}|.$$

Now we can phrase as a PAC problem. What is the sample size $m_F^{UC}(\varepsilon, \delta)$ to get within ε with probability $1 - \delta$? This depends on the generalized pseudodimension of F. The pseudodimension is defined as follows. For F is a family of functions $X \to \mathbb{R}$,

$$p\dim(F) = \max\left\{n : \exists x_1, \ldots, x_n, b_1, \ldots, b_n, \forall \sigma \in \{0,1\}^n, \exists f \in F, \forall i \le n, \mathbb{1}[f(x_i) \ge b_i] = \sigma_i\right\}.$$

It is the largest set which we can pseudo-shatter.

We have to generalize pseudodimension to vector-valued functions. The generalize is the maximum pseudodimension of all the projections.

In some common families of kernels, we can calculate the pseudodimension.

We don't have an efficient algorithm because to find the ERM we need to solve regularized k-means. How to use users' information to learn clustering on the whole dataset is interesting and this is only a partial answer.

There's little work on sample complexity of metric learning—how many pairs you need to get information about.

Would some condition on clustering make it easier? Fat-shattering dimension, etc.

3 Robust learning and inference (Yishay Mansour, TAU)

We consider the case that some of the attributes may be adversarially corrupted or missing. We limit the adversarial corruption to a finite set of modification rules, and we model it as a zero-sum game between an adversary, who selects a modification rule, and a predictor, who wants to accurately predict the state of nature. We consider a learning setting where the predictor receives a set of uncorrupted inputs and their classification. The predictor needs to select a hypothesis, from a known set of hypotheses, and is latter tested on inputs which the adversary might corrupt. We show how to utilize an ERM oracle to derive a near optimal predictor strategy, namely, picking a hypothesis that minimizes the error on the corrupted test inputs. We will also briefly mention the results for the inference model. In the inference setting the predictor has access to the joint uncorrupted distribution, and needs to build a predictor for adversarially corrupted inputs.

What do we mean by robustness? Different fields use it to refer to different things.

- Robust statistics: be immune to outliers.
- Robust optimization: be immune to small parameter perturbation (in some metric)
- Noise models in computational learning theory: Use a model of how data is generated, ex. flip labels with probability 0.1. But the algorithms can break if we flip with probability ≤ 0.1. Everything is calibrated to the specific noise level, 0.1.

  Ex. nearest neighbor
- Our model: "things are not what they seem."

Ex. Spam detection. Build many filters, and combine them into a spam filter. Unfortunately, the bad guys can adapt.

This is really a game between spammers and detectors. Spammers adjust content to the detectors. They can learn to fool a few detectors. Our goal is to classify spam correctly even if spammers adjust their messages.

Ex. Robust network failure detection.

- What happens when the detectors fail? We get from the detector something that looks reasonable but is incorrect.
- How to model failures? There are two ways, Bayesian vs. worst-case.

  A Bayesian needs to model the probability of failure and conditioning on the failure, what the distribution of outputs is.

  I'll consider worst-case.

The goal is to perform a good failure detection, in the sense of overcoming a k-point failure, under adversarial behavior.

I'll consider both missing and corrupted attributes.

1. Why do we have missing/corrupted data? Can we avoid it by requiring clean data?

   We used to think of ML as having correct inputs (we are choosing the inputs), and giving outputs.

   Applied ML doesn't have the methodology for choosing inputs.

2. Can we assume that it is iid? No, missing or corrupted data might depend on attribute value.

3. Do we have to clean the data? No, we should directly predict!

   Two paradigms: if you have missing data, fill in the missing data. If you clean the data you can use standard methods. There are multiple ways to clean, and cost could be high.

   I think the right thing to do is go directly to the next step, prediction.

Story: finance company wants to do data mining/machine learning. They find that 5% of the people were born Nov. 11, 1911. It's the only key you can press 6 strokes and get out of this field!

Ex. "Giants are more likely to be bilingual" because they entered their height in centimeters when asked for inches...

Consider some joint distribution D generating observable attributes x and labels y. An adversary corrupts x to produce z. We limit by the set of modification rules $\rho_i(y, x)$. The adversary can select the modification depending on x.

Notation:

- state of nature $y \in \{0, 1\}$.
- signals $x \in X$
- distribution $D(y, x)$
- observed signals $z \in Z$.
- Modification rules $\rho_i(y, x) = z$, computed in polytime, m modification rules.

  For example, $\rho_i(y, x)$ flips signal i, $\rho_o(y, x)$ flips the odd signals.

The goal is given z, predict y.

The predictor, given z, predicts y, and in this way defines a policy $\pi(z)$.

The adversary selects ρ_i either

- statically: before x is selected.
- adaptively: after x is selected.

Model as a zero-sum game. Fixing policy π and modification rule ρ_i,

$$\text{error}(\pi, \rho_i) = \mathbb{E}_{y,x}[\mathbb{P}[\pi(\rho_i(y, x)) \neq y]].$$

The optimal min-max error is

$$\text{error}^* = \min_{\pi} \max_{\rho_i} \text{error}(\pi, \rho_i).$$

Being optimal doesn't mean you're doing well: ex. the the adversary erases all inputs, and the predictor can't do anything. This means error* is very high.

The setting: Given a distribution D, known and computable, with set of possible corruptions:

For every D, there is an algorithm that given observable z, compute a prediction for y with probability of error error$^* + \varepsilon$, in time poly(n, m) in the static case, poly$(n)\exp(m)$ (?) in the adaptive case.

We assume that in the training set examples are uncorrupted, in the test set examples are corrupted by adversary. Given hypothesis class H, given oracle for ERM in H, the goal is to select a mixture of hypotheses from H that minimizes the error.

Theorem 3.1. *1. Given a sample S, we can efficiently compute a mixture which is ε-optimal on S.*

2. For $|S| \geq \frac{\ln |H|/\delta}{\varepsilon^4}$, with probability $1-\delta$, get error(h) − obsError$(h) \leq \varepsilon$.

References:

- Globerson and Roweis, nightmare at test time: robust learning by feature deletion
- Teo, Globerson, Roweis, Smola. Convex learning with invariances.

Where does it make a difference?

- Learning homogeneous hyperplanes $\text{sign}(wx^T)$. Regular learning is $\min_w \mathbb{P}[(wx^T)y < 0]$. Large margin is $\min_w \mathbb{P}[(wx^T)y < \lambda]$. Corruption is setting one x_i to 0.

  Assume uniform distribution.

  A static adversary would zero the coordinate having largest weight. Success probability becomes $\min_w \mathbb{P}[(wx^T)y < \max_i |w_i|]$

  An adaptive adversary zeros out the coordinate having largest weight with correct sign. For $\text{sign}(w_i x_i) = y$, get $\min_w \mathbb{P}[(wx^T)y < \max_i (x_i w_i)]$

Given ERM for class H, we'd like to learn a mixture of hypotheses $\Delta(H)$. Suppose that the number of corrupted inputs for x is $\leq m$. The error is

$$L(h) = \mathop{\mathbb{E}}_x [\max_{z \in \rho(x)} \ell(h(z), f(x))].$$

For any uncorrupted, consider the matrix M_x where $M_x(z,h) = \mathbb{1}(h(z) \neq y)$. (Here $z \in \rho(x), y = f(x)$.) The D selects the matrix.

The learner chooses $Q \in \Delta(H)$,

$$h_Q(z) = \sum_{h \in H} Q(h)h(z),$$

convex combination over h's. The learner doesn't know x, observes corrupted z, and generates a prediction. The adversary chooses $P_x \in \Delta(\rho(x))$, knows x, and generates $z \in \rho(x)$. They both play a mixed strategy. The goal of the learner is

$$Q^* = \text{argmin}_Q \mathop{\mathbb{E}}_x [\max_{P_x} P_x^T M_x Q] \tag{1}$$

$$\text{error}^* = \min_Q \max_P \mathop{\mathbb{E}}_x [P_x^T M_x Q] \tag{2}$$

$$= \max_P \min_Q \mathop{\mathbb{E}}_x [P_x^T M_x Q]. \tag{3}$$

Note this doesn't take advantage of a mild adversary.

Think of $\min_Q \mathbb{E}_x[P_x^T M_x Q]$ as an ERM over the fixed distribution.

We would like the algorithm to choose few $h_i \in H$ to get good generalization.

Algorithm is based on regret minimization. Use a variant of exponential weights (Cesa-Bianchi-Mansour-Stoltz 2007). Maintain weights for $(z, (x, y))$ for each $(x, y) \in S$ and $z \in \rho(x)$. Output defines both Q and P. Expand sample by all possible corruptions.

Initialize weights $w_q(z, (x, y)) = 1$. Update weights as follows: given $h_t(\cdot)$, if $h_t(z) \neq y$, then $w_{t+1}(z, (x, y)) = (1 + \eta)w_t(z, (x, y))$, else there is no change.

Normalize not for everything, but per x. Let P_t be the normalized w_t per x.

$$P_t(z, (x, y)) = \frac{w_t(z, (x, y))}{\sum_{z' \in \rho(x)} w_t(z', (x, y))}.$$

Given P_t use the ERM oracle to select h_t using $D^{P_t}(z, y)$:

$$\sum_{x: f(x) = y, z \in \rho(x)} P^t_x(z) D(x).$$

Use distribution to get next hypothesis to plug in.

Proof. The loss for $(z, (x, y))$ is

$$l_t(z, (x, y)) = l(h_t(z) \neq y) = M_x(z, h_t).$$

The cumulative loss of $(z, (x, y))$ is

$$L_T(z, (x, y)) = \sum_{t=1}^{T} l_t(z, (x, y)).$$

The algorithm loss is $L_T^{lin} = \sum_{t=1}^{T} P_t \cdot l_t$. Compare against the benchmark

$$L^* = \sum_{(x,y) \in S} \max_{z \in \rho(x)} L_T(z, (x, y)).$$

I finish learning, these are my hypothesis, what would you do? For each x choose the worst corruption.

The regret bound is

$$L^* - 2\sqrt{L^* |S| \ln m} \leq L_T^{lin}.$$

The strategies are $P^* = \frac{1}{T} \sum_t P_t$, $Q^* = \frac{1}{T} \sum_t h_t$.

Theorem 3.2. *Fix uncorrupted S, $T \geq \frac{4|S| \ln m}{\varepsilon^2}$. P^* is ε-optimal for adversary and Q^* is ε-optimal for the learner.*

The proof is like for exponential weights.

$$W^t_{(x,y)} = \sum_{z \in \rho(x,y)} w_t(z, (x, y)) \geq (1 + \eta)^{L^*(x,y)} \tag{4}$$

$$L^*(x, y) = \max_{z \in \rho(x)} L_T(z, (x, y)) \tag{5}$$

$$W^t = \prod_{(x,y) \in S} W^t_{(x,y)} \qquad \geq (1 + \eta)^{L^*} \tag{6}$$

$$L^* - \eta L^* - |S| \ln m / \eta \leq L_T^{lin}. \tag{7}$$

Multiplying the weights is the non-standard part. Take logs and do linear approximation.

Expected error given P, Q,

$$\begin{aligned} R(P,Q) &= \sum_{x,y}\sum_{z}\sum_{h} P(z,(x,y))Q(h)I(h(z)\neq y) && (8)\\ \max_P \min_Q R(P,Q) &\geq \min_Q R(P^*,Q) && (9)\\ &\geq \frac{L_T^{lin}}{T} && (10)\\ &\geq \frac{L^*}{T} - \frac{2\sqrt{L^*|S|\ln m}}{T} && (11)\\ &\geq \max_P R(P,Q^*) - \frac{2\sqrt{L^*|S|\ln m}}{T} && (12)\\ &\geq \min_Q \max_P R(P,Q) - \frac{2\sqrt{L^*|S|\ln m}}{T}. && (13) \end{aligned}$$

Using P^*, I'm getting as much as I can hope to get up to an additive factor. Q^* is ε-optimal for one side and P^* is ε-optimal for the other side.

To get generalization:

1. Let H^N be the class of averages of at most N $h_i \in H$. The first step is to limit N. It is enough to average $N_0 = O\left(\frac{1}{\varepsilon^2}\ln\left(\frac{1}{\delta}\right)\right)$. Use regular Chernoff bound.

2. Consider $h = \frac{1}{N}\sum_{i=1}^{N} h_i(z) \in H^N$. Bound the true and sample error by ε,

$$\mathop{\mathbb{E}}_{(x,y)}[\max_{z\in\rho(x)} \ell(h(z),y)].$$

   This requires $|S| \geq \frac{1}{\varepsilon^4}\ln\left(\frac{|H|}{\delta}\right)$.

   The max in the $\mathbb{E}$ makes things more complicated. We bounded sample by $O\left(\frac{1}{\varepsilon^2}\right)$ and N_0 by $O\left(\frac{1}{\varepsilon^2}\right)$ so get $O\left(\frac{1}{\varepsilon^4}\right)$.

The final theorem: given hypothesis class H and ERM oracle for H, there is algorithm that with probability $1-\delta$ computes ε-optimal learning hypothesis in time poly$(\left(\frac{1}{\varepsilon}\right), \left(\frac{1}{\delta}\right), \ln|H|)$.

Q: if adversary has finitely many choices, we can't model "choose error up to ε"—infinitely many choices?

A: First difficulty is showing the zero-sum game is well-defined.

This is far from robust statistics because I assume all inputs can be corrupted.

How to model robust statistics in this model: One modification rule is not modifying, the other is corrupt. Ex. put 98% on "don't modify".

4 Towards a Rigorous Approach to Approximate Computations in Huge-Scale Markovian Decision Processes (Csaba Szepesvari)

In this talk, without assuming any background beyond familiarity with basic probability and linear algebra, I will describe exciting new results on the computation of near-optimal policies in huge-scale Markovian decision processes (MDPs) via the so-called approximate linear programming methodology. In a nutshell, the new results provide meaningful upper bounds on the quality of the computed policies as a function of the linear basis functions chosen by the algorithm designer, while avoiding unrealistic assumptions which have effectively become the norm in prior results. As opposed to previous work that relied on worst-case reasoning over randomly sampled constraints, in this work the analysis is done using an operator-theoretic approach. I will finish by discussing the remaining main open problems, connections to alternative approaches, as well as potential extensions. Markov decision processes lie at the hear of many reinforcement learning algorithms and have found numerous applications across many engineering and scientific problems.

Based on joint work with Chandrashekar Lakshminarayanan and Shalabh Bhatnagar

For RL, one way to do things is: Let's learn a good model, run the planner, get a good policy. Does this work—do we have good planners?

Let

- S be state space
- A be action space
- P be transition probabilities (for every state and action, a distribution over next states, $P = (p_{sa} \in \Delta_n S), (s, a) \in S \times A)$.
- g be rewards, $g = (g_{sa} \in [0, 1])$
- $\alpha \in (0, 1)$ discount factor.

Classically, people think of S, A as finite. This is wrong: the number of states and actions in any reasonable space is exponential. You can't run algorithms even linear in the number of states and actions. We want sublinear scaling.

We want to maximize expected total reward

$$\mathbb{E}\left[\sum_{t=1}^{\infty} \alpha^t g_{S_t A_t}\right]$$

where $S_0, A_0, S_1, A_1, \ldots$ is the sequence of states and actions. [2]

Here we don't worry about exploration; we assume we can access the parameters.

[2] How strong do we believe in discounting?
We don't believe in it, we use it because it's convenient.
If you can do things with the discounted factor, you can probably do it without; this is the entry problem.

A policy is a function $u : S \to A$: look at the history and decide what action to take. The value of the policy is the expected total reward under the policy

$$J_u(s) = \mathbb{E}_\mu \left[\sum_{t=0}^{\infty} \alpha^t g_{S_t,A_t} | s_0 = s \right] \tag{14}$$

$$\sup_u J_u(s) = J^*(s). \tag{15}$$

The optimal policy is the u^* such that $J_{u^*}(s) = J^*(s)$.

There always exists a policy that doesn't look at the past history, and just acts based on the current state, and is deterministic.

Theorem 4.1. *There $u^* : S \to A$ such that $J_{u^*} = J^*$.*

Assume you have an oracle MDP that given input s, a, outputs g_{sa}, p_{sa}. Assume also that given $J : S \to \mathbb{R}$, the oracle MDP gives $p_{sa}^T J$. (If you have sampling, you can estimate this. There's not that much difference between being able to sample and having this.)

In reality, maybe the regression/classification function is very complex. Instead, we ask: can we compete with a good function?

Make a hypothesis space for the optimal value function. Consider a feature function $\varphi : S \to \mathbb{R}^d$, and look for

$$J^*(s) \approx \varphi^T(s) r^*, \quad r^* \in \mathbb{R}^d.$$

[3]

Ex. Consider $S = [0,1]^n$, $1 < |A| < \infty$.

Assume p_{sa}, g_{sa} is L-Lipschitz in L_1.

1. Chow-Tsitsiklis, negative result: To get $J_u \geq J^* - \varepsilon$, in the worst case we need $\left(\frac{1}{\varepsilon}\right)^{\frac{n}{L}}$ queries.

2. J. Rust: "Randomization, breaking the curse of dimensionality."

   Polynomial-time computation up to ε in expectation. Rust's model is different.

   What's missing? Computational complexity depends on the largest value in the density function, typically exponentially large. Can I rejection sample by sampling uniformly? [4]

I want to compute policies!

This line of work goes back to the 70's.

Schweitzer-Sneider: mix linear programming with linear function approximation.

You can find an optimal policy by solving a LP: take $c \in (0, \infty)^s$ and solve

$$\min_{J \geq TJ} c^T J$$

where

$$(TJ)(s) = \min_a (g_{sa} + \alpha p_{sa}^T).$$

[3] Debate: Assume something about model vs. assume something about solution (what we do here).

[4] Think of this as saying: MDP is a noisy system with no sinks, where you have no strong control. "If you have no control you can't be too stupid."

We know the optimal function is a fixed point $TJ^* = J^*$.

Define T_u by taking the action suggested by u: for $J : S \to \mathbb{R}$.

$$(T_u J)(s) = g_{su(s)} + \alpha p_{su(s)}^T J.$$

Let u be such that $T_u J = TJ$. The algorithm finds

$$\|J_u - J^*\|_\infty \leq \frac{\|J - J^*\|_\infty}{1-\alpha}.$$

This is a multiplicative blowup of the error, a weak reassurance.

Let's focus on calculating some approximation of J^*.

We add an additional constraint

$$\min_{J \geq TJ, J = \Phi r} c^T J$$

where $r \in \mathbb{R}^k$, $k \ll |S|$.

We would like: if Φ is chosen well, solution of LP will not be far from best approximation of J^*.

The hypothesis space may be crappy, we learn relative to the hypothesis space.

Let $T_a J(s) = g_{sa} + \alpha p_{sa}^T$. (Always take action a.)

Is this even feasible? If $\mathbb{1} \in \operatorname{span} \Phi$, this is feasible. We can prove something stronger. Given $J \in \operatorname{span}(\Phi)$, if there exists $J' \in \operatorname{span}(\Phi)$, $J' \geq TJ'$ such that $\|J - J'\|_\infty \leq \frac{(1+\alpha)\|J-J'\|_\infty}{1-\alpha}$, then there exists $r \in \mathbb{R}^k$, $J = \Phi r$, $\mathbb{1} \in \operatorname{span}(\Phi)$, $\exists r_0 \in \mathbb{R}^k$, $\mathbb{1} = \Phi r_0$, then for $\lambda \in \mathbb{R}$, $\lambda \geq 0$,

$$J' = \Phi(r + \lambda r_0) \geq \max_a T_a(r + \lambda r_0) \tag{16}$$

$$T_a(r + \lambda r_0)(s) = g_{sa} + \alpha p_{sa}^T \phi(r + \lambda r_0) = (g_{sa} + \alpha p_{sa}^T \Phi r) + \alpha\lambda. \tag{17}$$

Find the smallest value of λ that satisfies this: how much you have to travel to meet the feasibility constraint.

$$\Phi r + \lambda 1 \geq T\Phi r + \alpha\lambda 1. \tag{18}$$

Is it tractable to minimize the objective? We group

$$c^T \Phi r = \left(\sum c(s)\varphi(s)\right)^T r.$$

One popular choice for φ is where every feature depends on a few state space variables. The constraint is $J(s) \geq g_{sa} + \alpha p_{sa}^T$.

Algorithm takes as input Φ, c, consoluts the oracle, has a way of calculating $c^T\Phi$, and outputs $\widehat{r}$.

Can we evaluate whether constraints are met?

$$(\Phi r)(s) \geq g_{sa} + \alpha(p_{sa}^T \Phi) r.$$

There are as many as $S \times A$, too many.

D, Van Roy: We can't solve LP, just sample. LP: If you know the k constraints, you can just choose them. Error scales polynomially in the right quantities. Blowup depends on misalignment between sampling measure and probability measure induced by the optimal policy.

If I'm in a huge state space, this mismatch will likely be huge.

Project/linearly combine constraints: RALP

$$\min_{W_a^T J \ge W_a^T T_a J, T_a J = \Phi r} c^T J.$$

Think of this as keeping just a few constraints. How does this error propagate to the result?

Theorem 4.2. *Let $\widehat{J}$ be the solution to the RALP, $c \in \Delta_1(S)$. Then*

$$\left\| J^* - \widehat{J} \right\|_{1,C} \le \frac{C}{1-\alpha} \left(\inf_{r \in \mathbb{R}^d} \|J^* - \Phi r\|_\infty + \left\| \Gamma J^* - \widehat{\Gamma} J^* \right\|_\infty \right)$$

where

$$\Gamma J(s) = \min\left\{(\Phi r)(s) : \Phi r \ge T\Phi r, r \in \mathcal{N}\right\} \tag{19}$$

$$(\widehat{\Gamma} J)(s) = \min\left\{(\Phi r)(s) : W_a^T \Phi r \ge W_a^T T_a J, r \in \mathcal{N}\right\} \tag{20}$$

$$(\Gamma J^*)(s) = \min\left\{(\Phi r)(s) : \Phi r \ge J^*, r \in \mathcal{N}\right\} \tag{21}$$

$$(\widehat{\Gamma} J^*)(s) = \min\left\{(\Phi r)(s) : W^T \Phi r \ge W^T J^*\right\}. \tag{22}$$

(Once you drop some constraints, you can be unbounded, so we need $\mathcal{N}$.)

When you think about dropping the constraints, there are two things that are important.

Keep sufficiently many states, so for any new state you encounter, if you express as linear combination of features, coefficients should be small. This is like a subset selection problem: remaining ones can be expressed with linear combinations with small coefficients.

5 Sequential information maximization (Hamed Hassani)

Abstract: Optimal information gathering using uncertain observations is a central challenge in many disciplines of machine learning such as Bayesian experimental design, automated diagnosis, active learning, and decision making. A widely used method is to perform sequential observation selection, where the choice of the next observation depends on the history seen thus far. Despite the importance and widespread use in applications, little is known about the theoretical properties of sequential observation selection policies in the presence of noise. In particular, a long-open direction has been to analyse the persistent-noise setting that is arguably more relevant in practical applications. In this talk, we will present a new framework to capture the role of noise, and explain how it leads to the first rigorous analysis of the famous information-gain policy. We will then consider more general information gathering settings and provide new efficient algorithms, with provable guarantees, to deal with noisy observations.

Joint work with Yuxin Chen, Andreas Krause, and Amin Karbasi

5.1 General setup (Bayesian)

Let $Y \in \{y_1, \ldots, y_n\}$ the set of hypotheses, $X_1, \ldots, X_m$ the set of tests. The joint distribution is $P_{Y,X_1,\ldots,X_m}$.

Ex. sensors, recommendations, medical diagnoses (diseases, medical tests). A doctor wants to find out what diseases the patient has using tests. The first approach is the offline approach, select a susbet of tests and test. Based on all the results, decide what disease the patient case. In the adaptive setting, select one test; based on the result, choose the next test, and so on. Choose based on history. (Assume the tests don't affect the subject.)

Start with a uncertain prior. By performing a series of tests, we want to reduce uncertainty, have the posterior concentrated on one hypothesis.

Entropy measures uncertainty

$$H(Y) = -\sum_i p_{y_i} \ln p_{y_i}.$$

A uniform distribution has high entropy, a concentrated distribution has low entropy.

5.2 Information-gain heuristic

The information gain heuristic is a sequential greedy policy: at each step have a posterior $Y|$history with entropy $H(Y|\text{history})$. For each test X_i, the information that X_i provides is

$$H(Y|\text{history}) - H(Y|\text{history}, X_i) =: I(X_i; Y|\text{history}).$$

It is the difference in entropy before observing X_i and after observing X_i. The greedy policy is to pick the most informative test at each step.

The information-gain policy has been used since the 50's but there was no rigorous analysis.

The model is the conditionally independent Bayes model: conditioned on Y, the outcome of the tests $X_1, \ldots, X_m$ are independent. We know distributions $p_j(X_j = x_j|Y = y_i)$. People have analyzed this under the deterministic case and non-persistent noise case. (Mistakes are not systematic; you can make another query that's independent.)

One simple practical example if outcome of tests are noisy versions of the true outcome which is a deterministc function of Y.

A policy π is a decision tree; the next test is a function of the observations so far. We consider policies of length k. Initially there is some entropy $H(Y)$. At the end the entropy on average is $H(Y|\pi)$. We aim to maximize information gained by the policy,

$$I(\pi; Y) = H(Y) - H(Y|\pi).$$

We want

$$I_{\mathrm{OPT}[k]} = \max_{\pi \in \Pi_k} I(\pi; Y)$$

the optimal amount of information after k steps. We compare with

$$I_{IG[l]} = I(\pi_{IG[l]}; Y)$$

the gain of the information-gain heuristic after l steps. How large should l be so that we can approximate $I_{\mathrm{OPT}[k]}$?

Theorem 5.1. *To gain* $(1-\alpha)I_{OPT[k]}$ *it is sufficient to run IG for* $O(k \cdot \frac{\ln n \ln\left(\frac{1}{\alpha}\right)}{S})$.

Note in the offline noiseless setting this is a submodular optimization problem which is NP-hard.

Here S is the separability parameter,

$$S_i = \left(\min_{y,y':p_i(\cdot|y)\neq p_i(\cdot|y')} |p_i(\cdot|y) - p_i(\cdot|y')|_{TV}\right)^2, \tag{23}$$

$$S = \min_i S_i, \tag{24}$$

how well we can distinguish between any 2 hypotheses.

5.3 Analysis on conditionally independent model (proofs, insight)

Let G_l be the greedy algorithm after l steps. The gain is $I(G_l;Y)$. We want to compare with the optimal policy π_k^* which achieves $I(\pi_k^*;Y)$. We show

$$I(G_l;Y) \geq (1-\exp(-cl/k))I(\pi_k^*;Y)$$

where $c = \frac{S}{\ln n}$.

To simplify the proof, assume each test has the following structure: There is a deterministic outcome D_i, and noise is added to this outcome—the bit is flipped with probability ε. Call the noise part N_i. Here $S_i = (1-2\varepsilon)^2$ (BSC). The observed outcome is X_i.

The proof has 2 main steps.

1. Relate 1-step greedy to OPT.

   Let $i^* = \text{argmin}_{i\in[m]} I(X_i;Y|\text{history})$.

   Lemma 5.2.

   $$\max_{i\in[m]} I(X_i;Y|\textit{history}) \geq c\frac{\min_{\pi\in\Pi_k} I(\pi_k;Y|\textit{history})}{k}.$$

   LHS is the local gain; RHS is the global gain.

   Proof sketch of lemma: Relate the noiseless to noisy setting.

   (a) Consider the expected mass reduction by performing D_l, which is 1 with probability $1-p$. It is $2p(1-p)$.

   Pinsker's inequality gives $I(X_l;Y) \geq S_l p_l(1-p_l)$.

   (b) $I(\pi, N := \{N_i\}_{i\in M};Y) \geq I(\pi;Y)$.

   (c) (Longest step) $I(\pi,N;Y) = k\ln n \underbrace{\max_{i\in[m]} s_i p_i(1-p_i)}_{\leq I(x_i;y)}$.

2. The amount of information that is gained at the ith level of greedy is

$$\begin{aligned}
I(G_i;Y) - I(G_{i-1};Y) = I(G_i;Y|G_{i-1}) &= \mathbb{E}[I(X_{i^*};Y)|G_{i-1} = g_{i-1}] \tag{25}\\
&\geq \frac{C}{k}\mathbb{E}[I(\pi_k^*;Y|G_{i-1} = g_{i-1})] \tag{26}\\
&= \frac{C}{k}I(\pi_k^*;Y|G_{i-1}) \tag{27}\\
&= \frac{C}{k}(H(Y|G_{i-1}) - H(Y|G_{i-1},\pi_k^*)) \tag{28}\\
&\geq \frac{C}{k}(H(Y|G_{i-1}) - H(Y|\pi_k^*)) \tag{29}\\
&\geq \frac{C}{k}(H(Y) - H(Y|\pi_k^*) - (H(Y) - H(Y|G_{i-1}))) \tag{30}\\
I(G_i;Y) - I(G_{i-1};Y) = I(G_i;Y|G_{i-1}) &\geq \frac{C}{k}[I(\pi_k^*;Y) - I(G_{i-1};Y)] \tag{31}\\
\delta_i : &= I(\pi_k^*;Y) - I(G_i;Y) \tag{32}\\
\delta_i &\leq \left(1 - \frac{C}{k}\right)\delta_{i-1} \tag{33}\\
\implies \delta_l &\leq e^{-Cl/k}\delta_0. \tag{34}
\end{aligned}$$

5.4 Learning categories: new algorithms

Information gain heuristic could be arbitrarily bad. One such bad case is learning categories. We give new algorithms in this case.

Is the structural parameter S necessary? Yes. S can go to 0, making the bound arbitrarily bad.

Ex. find what category the hypothesis belongs to: type of disease, genre of movie. We want to learn Z depending on Y, which is much less complex than Y.

IG would choose the most informative test about Z. But S can become 0. Suppose hypothesis space has 4 hypotheses with $\frac{1}{4}$ probability. They are split into 2 groups, $Z =$green or red. Consider a test which cuts space into 2 parts, but doesn't inform you about Z. Then $I(X;Y) = 1$ but $I(X;Z) = 0$. So IG may fail arbitrarily badly compared to OPT.

We want to distinguish hypotheses in different categories. Transform hypothesis space, whose elements U are pairs of hypotheses y, y' with probability $p(y)p(y')$. We don't care about distinguishing y, y' in the same category. Merge those into one element. Learning Z is equivalent to learning U, $H(Z) = 0 \iff H(U) = 0$.

Pairwise information gain policy: At each step, choose the test that minimizes the uncertainty of U.

Ex. MovieLens-100k dataset. Objective: find category contains user's preference. Tests: given 2 movies, user picks one more similar to their preference.

References

[AB15] Hassan Ashtiani and Shai Ben-David. “Representation learning for clustering: A statistical framework”. In: *arXiv preprint arXiv:1506.05900* (2015).

[AKB16] Hassan Ashtiani, Shrinu Kushagra, and Shai Ben-David. “Clustering with Same-Cluster Queries”. In: *Advances In Neural Information Processing Systems*. 2016, pp. 3216–3224.