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 $O(kn(\dim))$, 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 $\forall i, \forall x \in C_i, \forall y \notin C_i$,

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

   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 $\gamma \leq 0.84$. (Euclidean distance) In general, it is NP-hard for $\gamma \leq 1$.*

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.)

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.

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

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

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

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 $X \to \mathbb{R}$,

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

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.

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.