COMS 6998-4 Fall 2017 Presenter: Kiran Vodrahalli
October 16, 2017 Scribe: Benjamin Kuykendall

Interactive Clustering

1 Introduction

The *clustering problem* on a data set is to partition it into clusters that reflect some similarity between the elements of a cluster. There are many different ways in which objects can be similar and dissimilar, so it is not always clear what the correct clustering is. But this ambiguity does not mean that clustering is impossible. Take a data set of newspaper articles including articles on {Nobel Prize, Columbia Lions Football, GRE Scores, US Soccer}. Then clearly we could make two clusters: one cluster Academics = {Nobel Prize, GRE Scores} and another Sports = {Columbia Lions Football, US Soccer}. But what model of input best models this sense of topical clustering?

Perhaps the most studied type of clustering is the *generative model* where we assume that the data set comes from a distribution. For example, each cluster may be drawn from a Gaussian as in [Das99]. Regardless of successes in that model, we will make no distribution assumption and treat this problem in the worst case over possible clusterings, i.e. clustering without assumptions.

So instead, we will use an *active* query model to learn the target clusters. To make our model realistic, we will stick to types of queries that seem easy for a human or algorithmic teacher to answer. This probably excludes the equivalence queries we studied last week. Instead, we will look at a new type of oracle based on the idea that a user will "know [a correct or incorrect cluster] when they see it". We will first introduce the *split/merge* model introduced by Balcam and Blum and then present their algorithms along with the later work of Awaasthi and Zadeh [BB08, AZ10].

2 Formal Setting

Take a universe $\mathcal{S} = \{x_1, x_2, \ldots, x_m\}$.

C is k-clustering if it partitions $\mathcal{S}$ into k sets;
i.e. $C = \{c_1, c_2, \ldots, c_k\}$ such that the c_i are disjoint and union to $\mathcal{C}$.

A hypothesis class $\mathcal{C}$ is the subsets of $\mathcal{S}$ considered valid clusters;
then $\{c_1, c_2, \ldots c_k\}$ belongs to $\mathcal{C}$ if each $c_i \in \mathcal{C}$.

H is a candidate clustering such that each $h_i \in H$ is a subset of $\mathcal{S}$;
this is the easiest problem. We discuss restrictions to the candidates later.

The split/merge oracle $\mathcal{O}$ for the target hypothesis C takes a candidate $H = \{h_1, h_2, \ldots, h_\ell\}$ and returns one of three types of outputs:

Correct if $H = C$

Split(h_i) if h_i contains elements from multiple clusters, i.e.

$$\exists x_1, x_2 \in h_i,\ \exists c_j, c_k \in C^* \text{ such that } x_1 \in c_j \text{ and } x_2 \in c_k.$$

Merge(h_i, h_j) if all elements of h_i, h_j belong to the same cluster, i.e.

$$\exists c_k \in C^* \text{ such that } h_i \cup h_j \subset c_k.$$

We say a deterministic learner learns a hypothesis class $\mathcal{C}$ with query complexity $q(k)$ if for every k-clustering C belonging to $\mathcal{C}$, the learner outputs C after at most q uses of the oracle.

2.1 Trivial algorithm

To get a sense of this setting and an easy upper bound, consider the following algorithm.

Algorithm 1 trivial algorithm for finite clustering

```
H_0 ← {{x} : x ∈ S}                          ▷ begin with all singletons
for t = 0, 1, ... do
    R ← O(H_t)
    if R = Merge(h_i, h_j) then
        H_{t+1} ← (H_t ∖ {h_i, h_j}) ∪ {h_i ∪ h_j}
    else R = Correct then                    ▷ never Split as proven below
        return H_t
    end if
end for
```

Theorem 1. *Algorithm 1 learns any class $\mathcal{C}$ over $\mathcal{S} = \{x_1, x_2, \ldots, x_m\}$ in time $m - k$.*

Proof. Claim the oracle never returns a Split answer. The clusters in H_t are one of two types

$\{x_i\}$: clearly a singleton can not be split.

$h_i \cup h_j$ such that we received Merge(h_i, h_j): then $h_i \cup h_j \subset c_j$ for some $c_k \in C$, and because the hypothesis clusters are disjoint any other $c_\ell \in C$ gives $c_\ell \cap (h_i \cup h_j) = \varnothing$.

By this claim, we know that each round we successfully merge, giving $|H_{t+1}| = |H_t| - 1 = nm - t$. So $|H_{(m-k)}| = k$, and we know the $h \in H_{(m-k)}$ are a partition of $\mathcal{S}$, and each $h \subseteq c$ for some $c \in C$. So we conclude $H_{(m-k)} = C$ and that the query complexity is $m - k$. □

This algorithm gives us a way to calibrate better approaches. If $O(n)$ is trivial, then maybe we can expect an efficient algorithm to achieve something like

$$q(k) = \text{poly}(\log m, k, \log |\mathcal{C}|).$$

2.2 Limiting Hypotheses

The intermediate hypotheses of the trivial algorithm are strange in a number of ways. This suggests that perhaps we have defined too strong of an oracle. Consider two ways to restrict oracle inputs that could make the problem more difficult.

In the trivial algorithm above, the singleton cluster hypothesis H_0 seems 'unnatural' in that it contains hypothesis clusters that may not be contained in $\mathcal{C}$. One way to make the problem harder is to only allow queries where $H \subset \mathcal{C}$. Call this the natural query setting.

Another potential problem is the size of the H_i. There are up to m hypotheses, which is potentially much larger than k. If we restrict to queries where $|H| < f(k)$, we get a weaker model we will call $f(k)$-restricted.

We will explore restricted oracles after presenting some specific problems in the basic setting, and seeing how they can be solved using potentially large sets of unnatural queries.

3 Specific Hypothesis Classes

3.1 Incremental Learners

The learners we build in this section will follow a similar pattern. An *incremental learner* maintains a single hypothesis and greedily updates it after each round of feedback. This category includes Algorithm 1 above. More generally, this template describes incremental learners for the split/merge feedback interactive clustering problem.

Algorithm 2 template for incremental learner

Parameters: H_0 an initial hypothesis, S and M splitting and merging functions
for $t = 0, 1, \ldots$ **do**
  $R \leftarrow \mathcal{O}(H_t)$
  if $R = \mathsf{Split}(h)$ **then**
    $H_{t+1} \leftarrow (H_t \setminus \{h\}) \cup S(h)$
  else if $R = \mathsf{Merge}(h_0, h_1)$ **then**
    $H_{t+1} \leftarrow (H_t \setminus \{h_0,\ h_1\}) \cup M(h_0, h_1)$
  else $R = \mathsf{Correct}$ **then**
    return H_t
  end if
end for

Where a splitting function is $S : h \mapsto \{a_0, a_1, \ldots, a_r\}$ such that $a_i \cap a_j = \varnothing$ and $\bigcup_i a_i = h$. And a merging function is $M : (h_0, h_1) \mapsto a$ such that $a \supseteq h_0 \cup h_1$.

In Algorithm 1, we had no splitting function, and our merging function was just a union. But even there we had an important property that helped the analysis of the proof. Say a merge function M is a good merge function when the oracle can never output both $\mathsf{Merge}(h_i, h_j)$

and $\mathsf{Split}(M(h_i, h_j))$. This means that you can think of an incremental learner as performing all of its splits, and then performing the merges. In practice, this means we only have to bound the number of splits. Then, since after s splits of arity r there are $(r-1)s + |H_0|$ clusters in H_s, we require $(r-1)s + |H_0| - k$ merges to get the final k-cluster solution. This bound will be used in both algorithms below.

3.2 Intervals on the Line

Let the universe consist of discrete points in $[0,1]$ ordered wlog $\mathcal{S} = \{x_1, x_2, \ldots, x_m\}$ with the hypothesis class of closed intervals. As closed intervals are subsets of $[0,1]$, not of $\mathcal{S}$, we have to be a little more precise. So let $\mathcal{C}$ be the projections of closed intervals onto the data set, namely $\mathcal{S} \cap [a,b]$ for $a, b \in [0,1]$, equating intervals with the same projection.

We have an incremental algorithm for this problem from [BB08]. Roughly, we begin with the whole interval. Then to split an interval, we cut it into two at the median of the contained x. To merge two intervals, take the smallest interval containing both. We give a more formal description in the theorem below.

Theorem 2. *The incremental learner*

$$H_0 = \{[0,1]\}$$

$$S([a,b]) = \big\{[a, m_-], [m_+, c]\big\} \text{ where } m \text{ is the median of } [a,b] \cap \mathcal{S} \text{ and } m_-,\ m_+ \in \mathcal{S} \text{ are the largest element to the left and smallest to the right of } m.$$

$$M([a,b],\ [c,d]) = [\min\{a,c\},\ \max\{b,d\}]$$

learns the class of intervals in $O(k \log m)$ queries.

Figure 1: The result of a Split in the incremental inverval learning algorithm.

Proof. First we need to show that M is good. This follows because if for some $[\alpha, \beta] \in C^*$ both $[a,b] \subset [\alpha, \beta]$ and $[c,d] \subset [\alpha, \beta]$, then any point in between is in $[\alpha, \beta]$ as well.

Next consider the boundary points of the target hypothesis, namely $a_1, \ldots, a_{k-1}$. We can split an interval $[x,y]$ only when some $x < a_i < y$. So we can do no more splits when the a_i are contained in none of the hypothesis intervals.

Let $\text{size}(a_i)$ be $|I(a_i) \cap \mathcal{S}|$ where $I(a_i)$ is the interval containing a_i, or 0 if there is no such interval. At the beginning of the algorithm, each $\text{size}(a_i) = m$. After a split, at least one of the $\text{size}(a_i)$ decreases to $\lceil \text{size}(a_i)/2 \rceil$. Further, when splitting $I(a_i)$ where $\text{size}(a_i) = 2$, the new size will be 0, because splitting an interval with two elements yields two singletons. Thus

each boundary point takes $\log m$ relevant splits before it has size 0. As each split is relevant to at least one boundary, there are at most $(k-1)\log m$ splits total possible.

After the splits, there are $(k-1)\log m + 1$ clusters, giving by the argument in Section 3.1 that there are at most $2(k-1)\log m + 1 - k$ queries total. □

Note that this algorithm is "natural" in the sense that each hypothesis is a valid clustering in the hypothesis class. However, the size of H might grow to m, so it is not valid in that limitted model.

3.3 Boxes in Euclidean Space

Taking $\mathcal{S}$ to be points in $[0,1]^d$ instead, we get a similar problem. In [AZ10] this algorithm is extended to give the following query complexity.

Theorem 3. *There is a $O((kd\log m)^d)$-query learner for closed boxes in $[0,1]^d$.*

We will show how the algorithm extends to $d = 2$, higher dimensions are identical. This algorithm is a bit different from the one dimensional case as the split operation is global: if effects each cluster in the hypothesis. But the analysis is much the same.

Theorem 4. *Take the following learner:*

> $H_0 = \{[0,1]\}$
>
> $M(h_0, h_1)$ *is the smallest rectangle containing both h_0 and h_1.*
>
> $S([x_0,x_1]\times[y_0,y_1])$*: project $\mathcal{S}\cap[0,1]\times[y_0,y_1]$ onto the x-axis find the median, and if any cluster in H_t that crosses the line, split it in two. Do the same for the y-axis.*

learns the class of boxes in $[0,1]^d$ in $O((k\log m)^2)$ queries.

Proof (sketch; pretending points never fall on the median). Follows much like Theorem 2. Again merged rectangles are never split. But boundaries are now the lines between target clusters. If a rectangle is split, that means it contains either a horizontal or vertical boundary line. So the size of that line (which is the sum of the number of points in each rectangle crossing that line) is decreased by $\lceil 1/2 \rceil$. There are $2k$ lines, so a total of $2k\log m$ splits may be used. Now, s splits induces an $s\times s$ grid on the data, giving s^2 total clusters. So after $(2k\log m)^2 - k$ merges, we are back down to k clusters, giving a total of $(2k\log m)^2 + 2k\log m - k$ queries. □

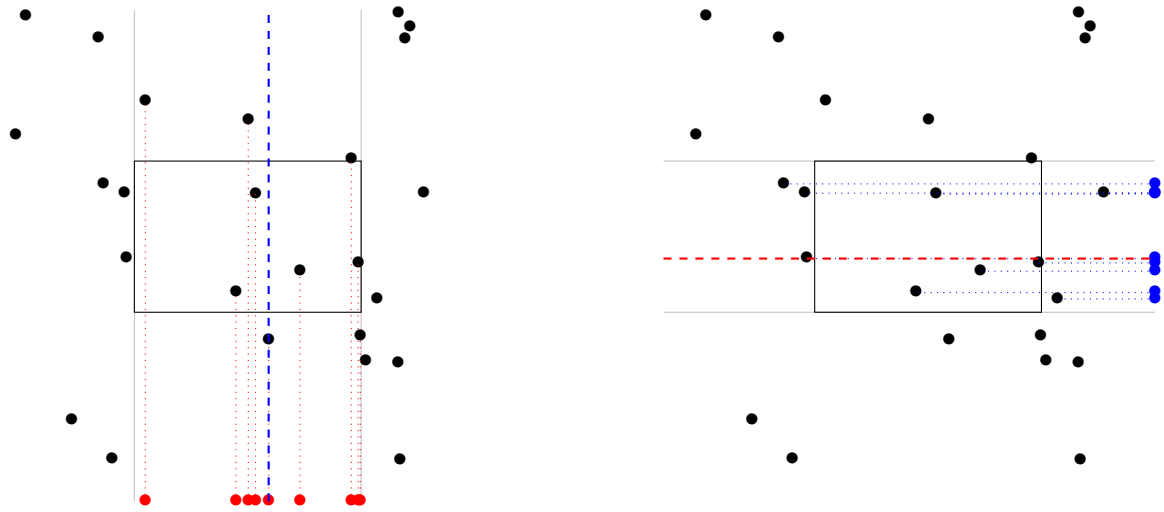

Figure 2: The x and y medians found by Split in the rectangle learning algorithm.

4 General Algorithm

Though the original [BB08] paper proposed a efficient generic algorithm, we will study the more efficient and simpler algorithm from [AZ10]. Roughly, the original attempted to cut the version space by a factor of $1 - 1/k^2$ each oracle class; the new manages $1/2$.

We will use some new notation in the algorithm. Let $\mathcal{C}^{VS}$ be all clusterings belonging to $\mathcal{C}$. Say a single cluster c is consistent with a clustering $\{c_1, \ldots, c_n\}$ if $c \subseteq c_i$ for some i. Let $\text{ccs}(c)$ denote the subset of $\mathcal{C}^{VS}$ which is consistent with c.

Algorithm 3 generic clustering algorithm

$V_0 \leftarrow \mathcal{C}^{VS}$
repeat for $t = 0, 1, \ldots$
  $L_i \leftarrow \varnothing$ for each i
  repeat for $i = 1, 2, \ldots$
    $L_i \leftarrow$ the largest $r \subseteq \mathcal{S} \setminus \bigcup_{n<i} L_n$ such that $|\text{ccs}(r) \cap V_t| \geq |V_t|/2$
  until the L_i partition $\mathcal{S}$
  $R \leftarrow \mathcal{O}(L)$
  if $R = \mathsf{Split}(h)$ **then**
    $V_{t+1} \leftarrow V_t \setminus \text{ccs}(h)$
  else if $R = \mathsf{Merge}(h_0, h_1)$ **then**
    $V_{t+1} \leftarrow V_t \cap \text{ccs}(h_0 \cup h_1)$
  end if
until $|V| = 1$
return the single $v \in V$

Theorem 5. *Algorithm 3 learns any class $\mathcal{C}$ in $O(\log|\mathcal{C}^{VS}|)$ queries.*

Proof. First we should show the construction of L is always feasible; this follows because singletons are always consistent with any hypothesis. Thus there is always some choice of L_i.

Next we claim that C^* is never removed from V; this follows from definitions. If $\mathsf{Split}(h)$ is returned, then we know C^* is inconsistent with h; if $\mathsf{Merge}(h_0, h_1)$ is returned, then we know C^* is consistent with $h_0 \cup h_1$. So what remains to show is that $|V_{t+1}| \leq |V|/2$.

For $\mathsf{Split}(L_i)$: by construction we have that $|\mathrm{ccs}(L_i) \cap V_t| \geq |V_t|/2$.

For $\mathsf{Merge}(L_i, L_j)$: assume towards contradiction $|\mathrm{ccs}(L_i \cup L_j) \cap V_t| \geq |V_t|/2$. Wlog $i < j$. But then we would have picked $L_i \cup L_j$ in round i because it is larger than L_i and it is consistent with enough of V. This contradicts the fact that L_i was picked; thus conclude $|\mathrm{ccs}(L_i \cup L_j) \cap V_t| < |V_t|/2$ so we remove more than half of V. □

We have a few useful bounds on $|\mathcal{C}^{VS}|$. To start, $\mathcal{C}^{VS} \subseteq \mathcal{C}^k$, so the algorithm in fact runs in $O(k \log|\mathcal{C}|)$ queries. Further, if d is the VC-dimension of $\mathcal{C}$, we know there are at most m^d ways to split $\mathcal{S}$ using $\mathcal{C}$ (the Sauer-Shelah lemma [Sau72, She72]), so we have $|\mathcal{C}^{VS}| \leq m^{kd}$ and $O(kd \log m)$ queries.

Note that this algorithm is not feasible to implement: like many of these generic algorithms, the version space is simply too large to keep track of. In terms of our limited models, the clusters are unnatural, and there are too many of them. There is in fact a lower bound that precludes efficient generic $\mathrm{poly}(k, \log m)$-restricted algorithms [BB08].

5 Discussion

In active learning, we want a query model that balances utility to the learner with how easy it is for a teacher to answer. How do the split/merge queries look by these metrics? The algorithms above surely suggest they are useful. But they might not be easy to answer.

> $\mathsf{Merge}(h_0, h_1)$ would be hard to check if cluster membership is hard to verify; the teacher has to check that every single $x \in h_0 \cup h_1$ is contained in some c.
>
> $\mathsf{Split}(h)$ would be hard to check if cluster non-membership is hard to verify; the teacher has to check that every single $x \in h$ does not belong to the same c.

It seems the teacher in the worst case must spend in the worst case time mT, where T is the time to verify or refute cluster membership, to answer a query.

Another possible remedy to the problem of hard queries is that the teacher is human, and has some very effective heuristics or probabilistic techniques to check membership. But in the case of human input, we would probably want a degree of error tolerance or noise.

The "noise" model considered by [AZ10] set some η and allowed $\mathsf{Merge}(h_0, h_1)$ responses when an η-fraction of $h_0 \cup h_1$ falls in some c. This modification certainly makes the task of verifying a merge easier, and as shown in the paper, still allows some effective algorithms. However, it does not address the possibility of actual errors in the response.

Another interesting modification of the model is to only allow hypotheses in $\mathcal{C}^{VS}$. This is certainly harder than either modification made in the paper. We could also strengthen the oracle considerably by allowing overlapping clusters in the hypothesis. To propose a new model in the interactive clustering setting, we should have efficient algorithms and well-motivated queries; perhaps some other modification of the [BB08] setting still makes sense by this measure.

References

[AZ10] Pranjal Awasthi and Reza Bosagh Zadeh. Supervised clustering. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems*, volume 1 of *NIPS'10*, pages 91–99, 2010.

[BB08] Maria-Florina Balcan and Avrim Blum. Clustering with interactive feedback. In *Proceedings of the 19th International Conference on Algorithmic Learning Theory*, ALT '08, pages 316–328, 2008.

[Das99] Sanjoy Dasgupta. Learning mixtures of gaussians. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, FOCS '99, page 634, 1999.

[Sau72] N Sauer. On the density of families of sets. *Journal of Combinatorial Theory, Series A*, 13(1):145–147, 1972.

[She72] Saharon Shelah. A combinatorial problem; stability and order for models and theories in infinitary languages. *Pacific J. Math.*, 41(1):247–261, 1972.