Active learning: Beyond the classics

Christopher Tosh

Columbia University

TRIPODS Bootcamp

Last time: Active learning for general hypothesis classes

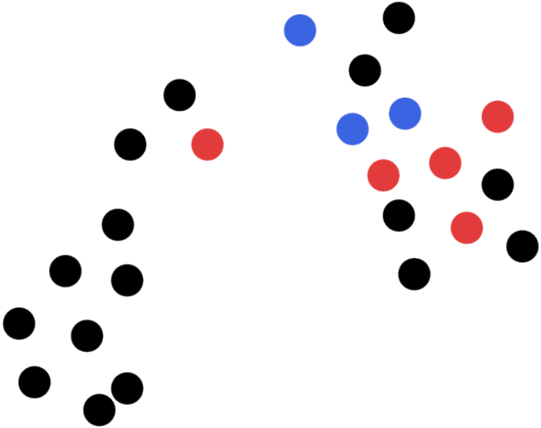
	Separable data	General (nonseparable) data
Aggressive	QBC [FSST97] Splitting index [D05] GBS [D04, N09]	
Mellow	CAL [CAL94]	A^2 algorithm [BBL06, H07] Reduction to supervised [DHM07] Importance weighted [BDL09] Confidence rated prediction [ZC14]

Today: Beyond classical active learning

- Nonparametric active learning
- Interactive clustering

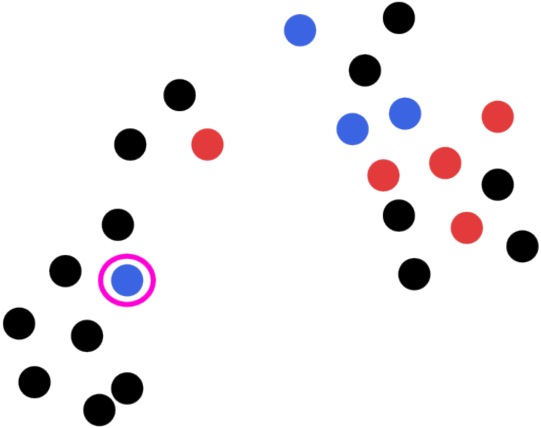
What's wrong with active learning (so far)?

- Don't always know right hypothesis class a priori.
- Labeled dataset from active learning is highly biased.



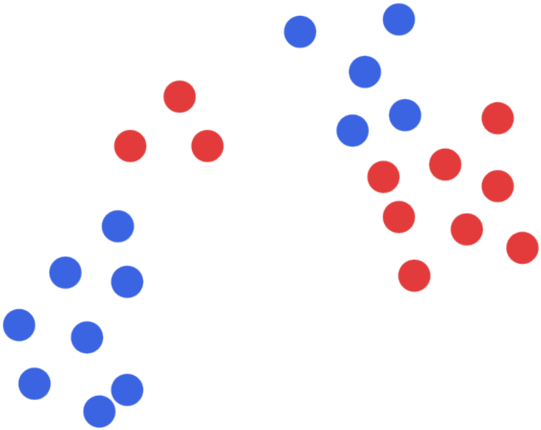
What's wrong with active learning (so far)?

- Don't always know right hypothesis class a priori.
- Labeled dataset from active learning is highly biased.



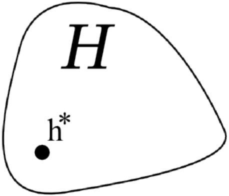
What's wrong with active learning (so far)?

- Don't always know right hypothesis class a priori.
- Labeled dataset from active learning is highly biased.



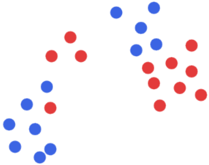
Nonparametric active learning

Parametric



- Given: fixed hypothesis class and unlabeled data
- Can query data points for their labels
- Goal: find low error hypothesis from this class

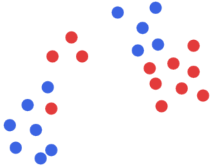
Nonparametric



- Given: unlabeled data
- Can query data points for their labels
- Goal: infer the labels of all data points with low error

Nonparametric active learning

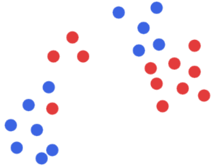
Nonparametric



- Given: unlabeled data
- Can query data points for their labels
- Goal: infer the labels of all data points with low error

Nonparametric active learning

Nonparametric

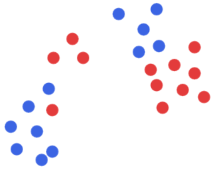


- Given: unlabeled data
- Can query data points for their labels
- Goal: infer the labels of all data points with low error

Issue: 2^n possible labelings!

Nonparametric active learning

Nonparametric



- Given: unlabeled data
- Can query data points for their labels
- Goal: infer the labels of all data points with low error

Issue: 2^n possible labelings!

Solution: some labelings are more likely than others

Preferences in labelings

How are some labelings given preference over others?

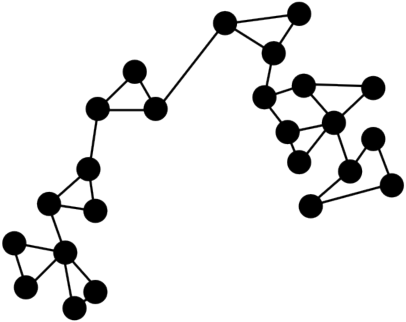
- Graph-based methods
- Cluster-based methods

Preferences in labelings

How are some labelings given preference over others?

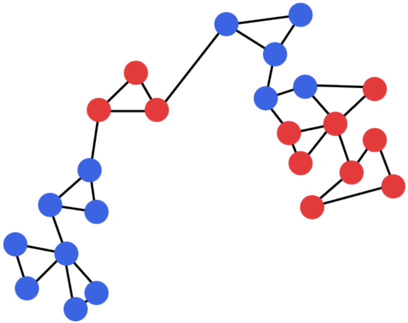
- Graph-based methods
- Cluster-based methods

Graph-based methods



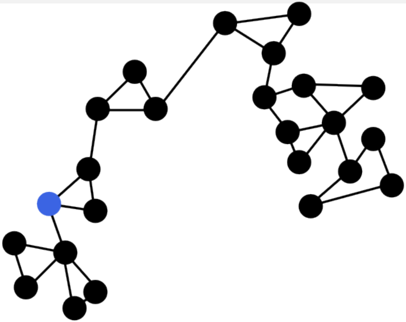
- Given (or construct): a similarity graph $G = (V, E)$
- Assumption: Vertices that share an edge are more likely to have same label

Graph-based methods



- Given (or construct): a similarity graph $G = (V, E)$
- Assumption: Vertices that share an edge are more likely to have same label

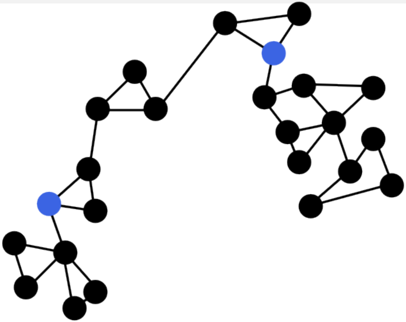
S^2: Active learning strategy



While budget not exhausted:

- Randomly sample until there are differently labeled points on same connected component
- Repeat until all connected components have single label:
  - Remove edges between differently labeled points
  - Find shortest path between differently labeled points and query midpoint.

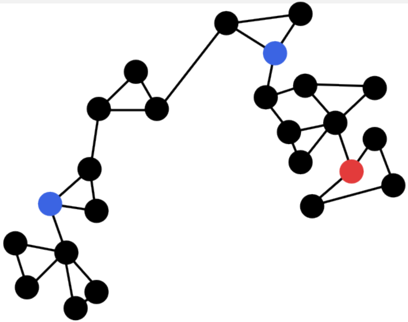
S^2: Active learning strategy



While budget not exhausted:

- Randomly sample until there are differently labeled points on same connected component
- Repeat until all connected components have single label:
  - Remove edges between differently labeled points
  - Find shortest path between differently labeled points and query midpoint.

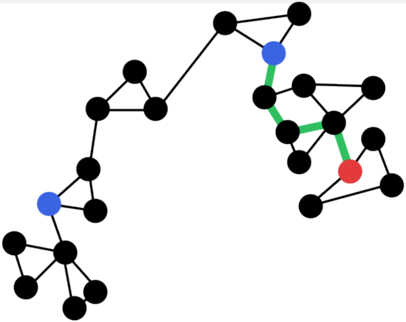
S^2: Active learning strategy



While budget not exhausted:

- Randomly sample until there are differently labeled points on same connected component
- Repeat until all connected components have single label:
  - Remove edges between differently labeled points
  - Find shortest path between differently labeled points and query midpoint.

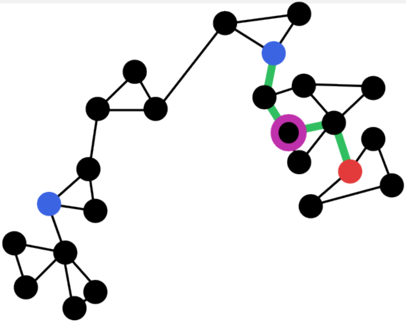
S^2: Active learning strategy



While budget not exhausted:

- Randomly sample until there are differently labeled points on same connected component
- Repeat until all connected components have single label:
  - Remove edges between differently labeled points
  - Find shortest path between differently labeled points and query midpoint.

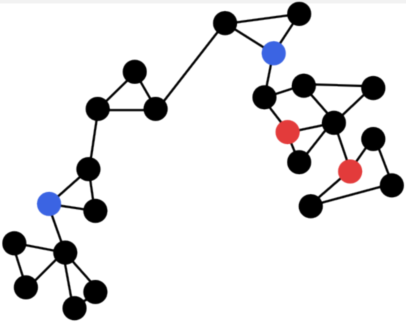
S^2: Active learning strategy



While budget not exhausted:

- Randomly sample until there are differently labeled points on same connected component
- Repeat until all connected components have single label:
  - Remove edges between differently labeled points
  - Find shortest path between differently labeled points and query midpoint.

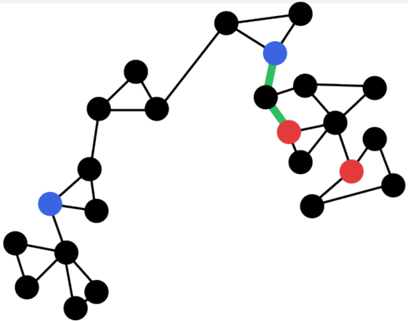
S^2: Active learning strategy



While budget not exhausted:

- Randomly sample until there are differently labeled points on same connected component
- Repeat until all connected components have single label:
  - Remove edges between differently labeled points
  - Find shortest path between differently labeled points and query midpoint.

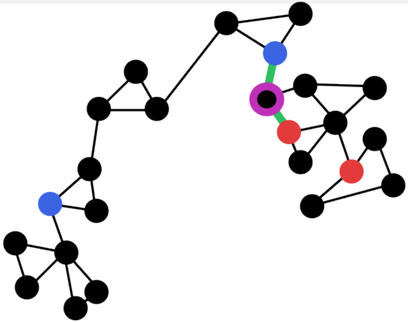
S^2: Active learning strategy



While budget not exhausted:

- Randomly sample until there are differently labeled points on same connected component
- Repeat until all connected components have single label:
  - Remove edges between differently labeled points
  - Find shortest path between differently labeled points and query midpoint.

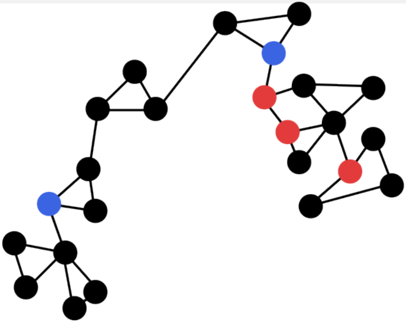
S^2: Active learning strategy



While budget not exhausted:

- Randomly sample until there are differently labeled points on same connected component
- Repeat until all connected components have single label:
  - Remove edges between differently labeled points
  - Find shortest path between differently labeled points and query midpoint.

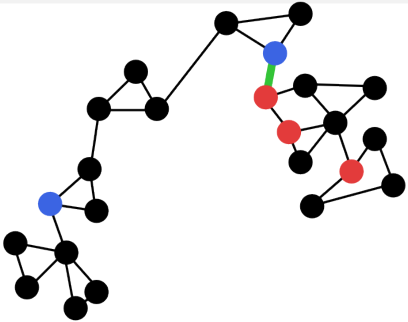
S^2: Active learning strategy



While budget not exhausted:

- Randomly sample until there are differently labeled points on same connected component
- Repeat until all connected components have single label:
  - Remove edges between differently labeled points
  - Find shortest path between differently labeled points and query midpoint.

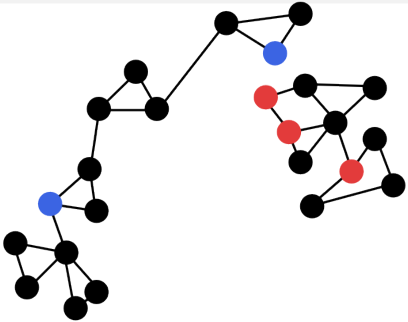
S^2: Active learning strategy



While budget not exhausted:

- Randomly sample until there are differently labeled points on same connected component
- Repeat until all connected components have single label:
  - Remove edges between differently labeled points
  - Find shortest path between differently labeled points and query midpoint.

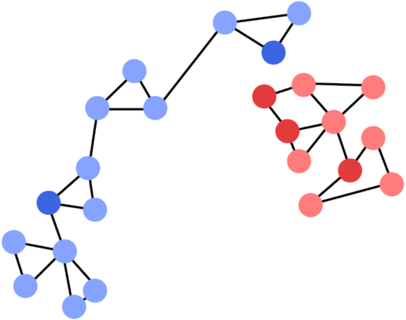
S^2: Active learning strategy



While budget not exhausted:

- Randomly sample until there are differently labeled points on same connected component
- Repeat until all connected components have single label:
  - Remove edges between differently labeled points
  - Find shortest path between differently labeled points and query midpoint.

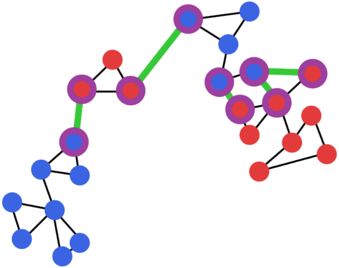
S^2: Active learning strategy



When budget is exhausted

- Give each connected component the majority label.

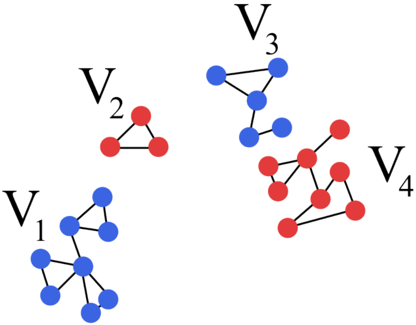
S^2: Label complexity



Relevant quantities:

- Cutset: $C = \{(u, v) \in E \,:\, h^*(u) = +, h^*(v) = -1\}$
- Cutset boundary: $\partial C = \bigcup_{(u,v) \in C} \{u, v\}$
- Balanced-ness: $\beta = \min \frac{|V_i|}{|V|}$ for connected components $V_1, \ldots, V_m$
- Clustered-ness: $\kappa =$ 'how tightly connected cutset edges are'

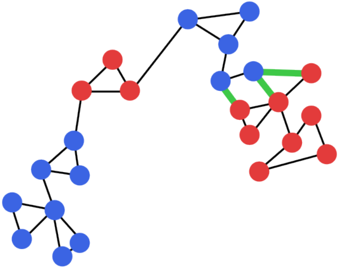
S^2: Label complexity



Relevant quantities:

- Cutset: $C = \{(u,v) \in E \,:\, h^*(u) = +, h^*(v) = -1\}$
- Cutset boundary: $\partial C = \bigcup_{(u,v)\in C} \{u,v\}$
- Balanced-ness: $\beta = \min \frac{|V_i|}{|V|}$ for connected components $V_1, \ldots, V_m$
- Clustered-ness: $\kappa =$ 'how tightly connected cutset edges are'

S^2: Label complexity



Relevant quantities:

- Cutset: $C = \{(u,v) \in E \,:\, h^*(u) = +, h^*(v) = -1\}$
- Cutset boundary: $\partial C = \bigcup_{(u,v)\in C} \{u,v\}$
- Balanced-ness: $\beta = \min \frac{|V_i|}{|V|}$ for connected components $V_1, \ldots, V_m$
- Clustered-ness: $\kappa =$ 'how tightly connected cutset edges are'

S^2: Label complexity

Relevant quantities:

- Cutset: $C = \{(u, v) \in E : h^*(u) = +, h^*(v) = -1\}$
- Cutset boundary: $\partial C = \bigcup_{(u,v)\in C} \{u, v\}$
- Balanced-ness: $\beta = \min \frac{|V_i|}{|V|}$ for connected components $V_1, \ldots, V_m$
- Clustered-ness: $\kappa =$ 'how tightly connected cutset edges are'

Theorem (Dasarathy et al. 2015)

With probability $1 - \delta$*, can recover all labels after*

$$\frac{1}{\beta} \log\left(\frac{m}{\delta}\right) + m \log \frac{n}{\kappa} + |\partial C|(1 + \log \kappa)$$

queries

S^2: Label complexity

Theorem (Dasarathy et al. 2015)

With probability $1-\delta$, can recover all labels after

$$\frac{1}{\beta}\log\left(\frac{m}{\delta}\right) + m\log\frac{n}{\kappa} + |\partial C|(1+\log\kappa)$$

queries

- Random sampling phase
- Binary search phase

S^2: Proof idea

Can handle random sampling phase and binary search phase separately.

S^2: Proof idea

Can handle random sampling phase and binary search phase separately.

Random sampling phase: $R = \#$ of random labels requested.

$$R \;\leq\; \# \text{ of random labels needed to find a point in each } V_i =: k$$

S^2: Proof idea

Can handle random sampling phase and binary search phase separately.

Random sampling phase: $R = \#$ of random labels requested.

$$R \ \leq \ \# \text{ of random labels needed to find a point in each } V_i =: k$$

How big do we need k to be?

$$\begin{aligned}
\Pr(\text{there is some } V_i \text{ with no labels}) &\leq \sum_{i=1}^{m} \Pr(V_i \text{ doesn't get sampled}) \\
&\leq \sum_{i=1}^{m} \left(1 - \frac{|V_i|}{|V|}\right)^k \\
&\leq \sum_{i=1}^{m} (1-\beta)^k \ \leq \ me^{-\beta k}
\end{aligned}$$

Taking $k = \frac{1}{\beta} \log \frac{m}{\delta}$ makes this hold with probability $1 - \delta$.*

S^2: Proof idea

Can handle random sampling phase and binary search phase separately.

Binary search phase: $B = \#$ of binary search labels requested.

Simple analysis:

S^2: Proof idea

Can handle random sampling phase and binary search phase separately.

Binary search phase: $B = \#$ of binary search labels requested.

Simple analysis: Given that we have a labeled point in each component,

$$
\begin{aligned}
B &\leq \sum_{e \in C} \# \text{ of queries needed to find endpoints of } e \\
&\leq \sum_{e \in C} \log\left(\text{longest length of a shortest path containing } e\right) \\
&\leq |C| \log n
\end{aligned}
$$

S^2: Proof idea

Can handle random sampling phase and binary search phase separately.

Binary search phase: $B = \#$ of binary search labels requested.

Simple analysis: Given that we have a labeled point in each component,

$$
\begin{aligned}
B &\le \sum_{e \in C} \# \text{ of queries needed to find endpoints of } e \\
&\le \sum_{e \in C} \log \left(\text{longest length of a shortest path containing } e\right) \\
&\le |C| \log n
\end{aligned}
$$

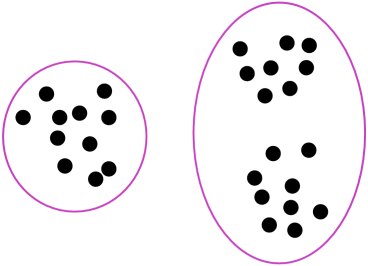
More complicated analysis: take advantage of ‘clustered-ness’ of cut-edges.

Preferences in labelings

How are some labelings given preference over others?

- Graph-based methods
- Cluster-based methods

Clustering-based methods

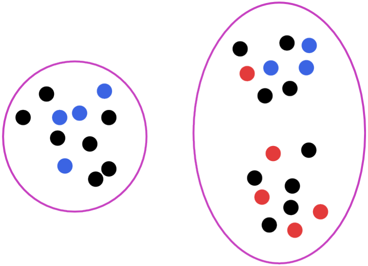


For rounds $t = 1, 2, \ldots, T$:

- Maintain a clustering $\mathcal{C}_t$
- Query some data points
- Possibly split some clusters to obtain a new clustering $\mathcal{C}_{t+1}$

At the end, each point gets majority label of its cluster in $\mathcal{C}_T$.

Clustering-based methods

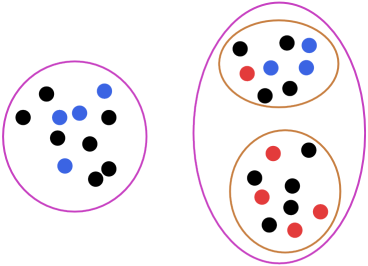


For rounds $t = 1, 2, \ldots, T$:

- Maintain a clustering $\mathcal{C}_t$
- Query some data points
- Possibly split some clusters to obtain a new clustering $\mathcal{C}_{t+1}$

At the end, each point gets majority label of its cluster in $\mathcal{C}_T$.

Clustering-based methods

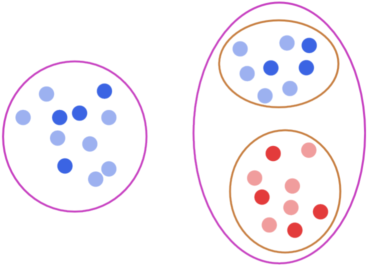


For rounds $t = 1, 2, \ldots, T$:

- Maintain a clustering $\mathcal{C}_t$
- Query some data points
- Possibly split some clusters to obtain a new clustering $\mathcal{C}_{t+1}$

At the end, each point gets majority label of its cluster in $\mathcal{C}_T$.

Clustering-based methods

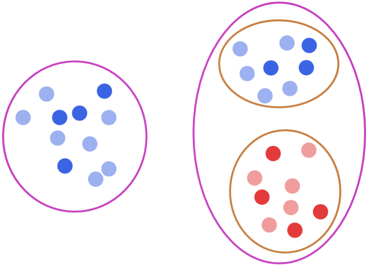


For rounds $t = 1, 2, \ldots, T$:

- Maintain a clustering $\mathcal{C}_t$
- Query some data points
- Possibly split some clusters to obtain a new clustering $\mathcal{C}_{t+1}$

At the end, each point gets majority label of its cluster in $\mathcal{C}_T$.

Clustering-based methods



For rounds $t = 1, 2, \ldots, T$:

- Maintain a clustering $\mathcal{C}_t$
- Query some data points
- Possibly split some clusters to obtain a new clustering $\mathcal{C}_{t+1}$

At the end, each point gets majority label of its cluster in $\mathcal{C}_T$.

Question: When does this strategy work?

Clustering-based methods: Rules

Question: When does this strategy work?

- Rule 1: At each round t, query is a uniform random draw from some chosen cluster $C \in \mathcal{C}_t$.
- Rule 2: At two rounds $t' > t$, the clustering $\mathcal{C}_{t'}$ is a refinement of $\mathcal{C}_t$:

  for all $C' \in \mathcal{C}_{t'}$ there exists a $C \in \mathcal{C}_t$ such that $C' \subseteq C$

- Rule 3: When a cluster is split, the manner of split cannot depend on the labels seen so far.

Clustering-based methods: Rules

Question: When does this strategy work?

- Rule 1: At each round t, query is a uniform random draw from some chosen cluster $C \in \mathcal{C}_t$.
- Rule 2: At two rounds $t' > t$, the clustering $\mathcal{C}_{t'}$ is a refinement of $\mathcal{C}_t$:

  for all $C' \in \mathcal{C}_{t'}$ there exists a $C \in \mathcal{C}_t$ such that $C' \subseteq C$

- Rule 3: When a cluster is split, the manner of split cannot depend on the labels seen so far.

Rules 2 + 3 $\Longrightarrow$ might as well start with a hierarchical clustering

Clustering-based methods: Algorithms

Start with hierarchical clustering T, $\ell = 0$, let $\mathcal{C} = \{\text{root node}\}$

While there are unlabeled points:

- For each cluster $C \in C$:
  - Request labels for $n(\ell)$ random points
  - If all labels in C are the same:
    - Assign this label to rest of points in C
    - Remove C from $\mathcal{C}$
  - Otherwise if there are also unlabeled points in C:
    - Replace C its children in T

Clustering-based methods: Guarantees

Theorem (Urner et al. 2013)

With probability $1-\delta$, the above procedure gets all but an ϵ-fraction of the points correct using $n(\ell) = \frac{1}{\epsilon}(2\ell \ln 2 + \ln(1/\delta))$.

Only need to consider case where we propagated labels, but an ϵ-fraction of those were incorrect.

Given $n(\ell)$ random labels, the probability of this happening in a particular node is

$$\begin{aligned}\Pr\left(\text{bad event in cell at level } \ell\right) &= \leq (1-\epsilon)^{n(\ell)} \\ &\leq e^{-\epsilon n(\ell)}\end{aligned}$$

Clustering-based methods: Guarantees

Theorem (Urner et al. 2013)

With probability $1-\delta$, the above procedure gets all but an ϵ-fraction of the points correct using $n(\ell) \; = \; \frac{1}{\epsilon}(2\ell \ln 2 + \ln(1/\delta))$.

Summing over all levels in the tree and all nodes in each level,

$$\begin{aligned}
\Pr\left(\text{any of these bad events happen}\right) &\leq \sum_{\ell=1}^{\infty} \sum_{C \in T:\text{level } \ell} e^{-\epsilon n(\ell)} \\
&\leq \sum_{\ell=1}^{\infty} 2^{\ell} \cdot e^{-\epsilon n(\ell)} \\
&= \sum_{\ell=1}^{\infty} 2^{-\ell} \delta \; = \; \delta
\end{aligned}$$

Clustering-based methods: Label complexity

How many labels does this procedure need? Depends on the data:

- How much are the clusters shrinking as we move down the tree?
- How often do labels of x, x' differ when $d(x, x')$ is small?

A partial list of interactive “unsupervised” learning cases

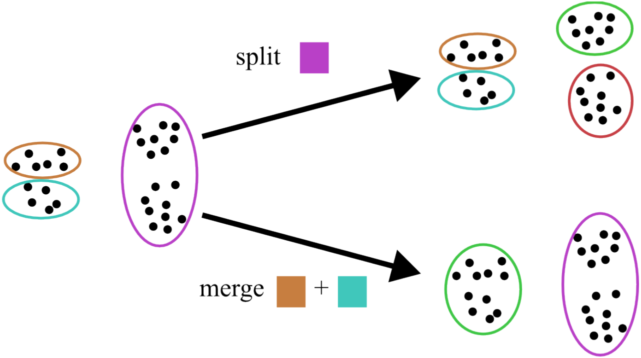
Learning task	Feedback type
Flat clustering	Split-and-merge requests Must-link/Cannot-link constraints
Hierarchical clustering	Triplet constraints
Embedding	Ordinal comparisons
Interactive topic modeling	Word-constraints

A partial list of interactive “unsupervised” learning cases

Learning task	Feedback type
Flat clustering	Split-and-merge requests Must-link/Cannot-link constraints
Hierarchical clustering	Triplet constraints
Embedding	Ordinal comparisons
Interactive topic modeling	Word-constraints

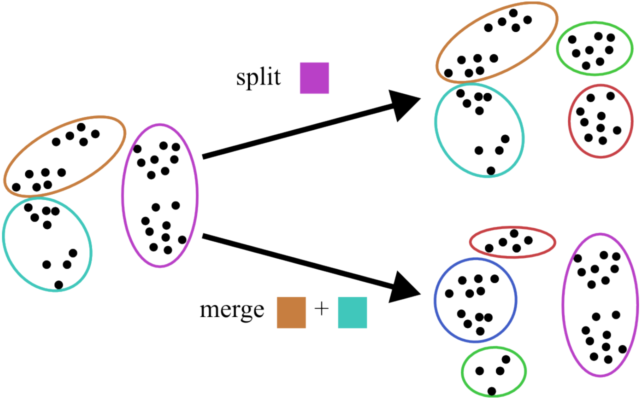
Split-and-merge feedback

Ideal:



Split-and-merge feedback

Actual:



Split-and-merge feedback

Assumptions:

- There is some ground truth clustering $\mathcal{C} = \{C_1, \ldots, C_k\}$
- A user requests to *split* a cluster C only if C contains points from more than one target cluster
- A user requests to *merge* two clusters C and C' only if there exists a cluster C_i such that

$$\min\{|C \cap C_i|/|C|, |C' \cap C_i|/|C'|\} \;\geq\; \eta$$

Split-and-merge algorithms

Given: an initial clustering $\widehat{\mathcal{C}}$ and a hierarchical clustering T s.t. a pruning of T corresponds to target $\mathcal{C}$. Every cluster is initially 'impure.'

Split-and-merge algorithms

Given: an initial clustering $\widehat{\mathcal{C}}$ and a hierarchical clustering T s.t. a pruning of T corresponds to target $\mathcal{C}$. Every cluster is initially 'impure.'

`Split(`C`)`:

- Search T to find shallowest node N at which the points in C are split into two clusters N_1 and N_2.
- Replace C with $C \cap N_1$ and $C \cap N_2$, and mark both as 'impure.'

`Merge(`C_1, C_2`)`:

- If C_1 is 'pure' then $\eta_1 = 1$ else $\eta_1 = \eta$. Similarly for C_2, η_2.
- Search T to find deepest node N at which

$$|N \cap C_1|/|C_1| \geq \eta_1 \text{ and } , |N \cap C_2|/|C_2| \geq \eta_2$$

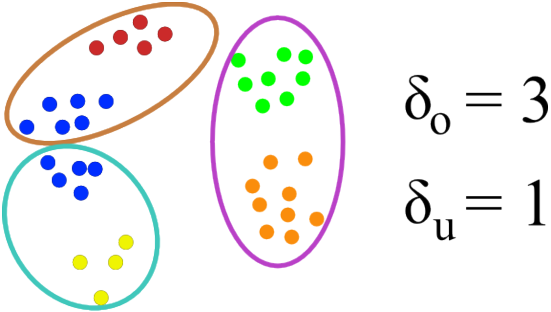
- Replace C_1 with $C_1 \setminus N$, C_2 with $C_2 \setminus N$ and create new 'pure' cluster $N \cap (C_1 \cup C_2)$.

Clustering errors

Let $\mathcal{C}^*$ be target clustering and $\mathcal{C}$ be arbitrary clustering.

$$\delta_o(\mathcal{C}) = \sum_{C_i \in \mathcal{C}} |\{C_j^* \in \mathcal{C}^* : C_i \cap C_j^* \neq \emptyset\}| - |\mathcal{C}|$$

$$\delta_u(\mathcal{C}) = \sum_{C_j^* \in \mathcal{C}^*} |\{C_i \in \mathcal{C} : C_i \cap C_j^* \neq \emptyset\}| - |\mathcal{C}^*|$$



Split bounds: sketch

Lemma

Say the initial clustering is $\mathcal{C}$ and the target clustering is $\mathcal{C}^*$. Then

$$\#\ \textit{of split requests} \ \leq \ \delta_o(\mathcal{C})$$

Observation 1: `Merge` does not increase δ_o.

Observation2: Whenever `Split(`C`)` is called to create nodes C_1 and C_2, we have by laminarity of T with $\mathcal{C}^*$

$$C_j^* \cap C_1 = C_j^* \cap C \qquad \textbf{or} \qquad C_j^* \cap C_2 = C_j^* \cap C$$

for all $C_j^* \in \mathcal{C}^*$. Thus $k = k_1 + k_2$ for

$$\begin{aligned} k &= |\{C_j^* \in \mathcal{C}^* \ : \ C \cap C_j^* \neq \emptyset\}| \\ k_1 &= |\{C_j^* \in \mathcal{C}^* \ : \ C_1 \cap C_j^* \neq \emptyset\}| \\ k_2 &= |\{C_j^* \in \mathcal{C}^* \ : \ C_2 \cap C_j^* \neq \emptyset\}| \end{aligned}$$

Split bounds: sketch

Lemma

Say the initial clustering is $\mathcal{C}$ and the target clustering is $\mathcal{C}^*$. Then

$$\#\textit{ of split requests} \ \leq \ \delta_o(\mathcal{C})$$

Thus $k = k_1 + k_2$ for

$$\begin{aligned} k &= |\{C_j^* \in \mathcal{C}^* \ : \ C \cap C_j^* \neq \emptyset\}| \\ k_1 &= |\{C_j^* \in \mathcal{C}^* \ : \ C_1 \cap C_j^* \neq \emptyset\}| \\ k_2 &= |\{C_j^* \in \mathcal{C}^* \ : \ C_2 \cap C_j^* \neq \emptyset\}| \end{aligned}$$

Then after `Split(`C`)`, we have

$$\begin{aligned} \delta_o((\mathcal{C} \setminus \{C\}) \cup \{C_1, C_2\}) &= \delta_o(\mathcal{C}) - (k-1) + (k_1 - 1) + (k_2 - 1) \\ &= \delta_o(\mathcal{C}) - 1 \end{aligned}$$

Merge bounds: sketch

Lemma

Say the initial clustering is $\mathcal{C}$ and the target clustering is $\mathcal{C}^*$. Then

$$\#\text{ of merge requests } \leq 2(\delta_u(\mathcal{C}) + |\mathcal{C}^*|) \log_{1/(1-\eta)} n$$

Each merge is either:

- Pure: both clusters are marked 'pure.' Creates a single pure cluster.
- Impure: one of the clusters is marked 'impure.' Creates at least one pure cluster.

Let $P = \{C_i \cap C_j^* : C_i \text{ is 'impure' and } C_i \cap C_j^* \neq \emptyset\}$.

An impure merge reduces at least one of the elements of P by an η fraction.

$\#$ of times set S can be reduced by an η fraction is $\leq \log_{1/(1-\eta)} |S|$

$|P| \leq \sum_{C_j^* \in \mathcal{C}^*} |\{C_i \in \mathcal{C} : C_i \cap C_j^* \neq \emptyset\}| = \delta_u(\mathcal{C}) + |\mathcal{C}^*|$

Merge bounds: sketch

Lemma

Say the initial clustering is $\mathcal{C}$ and the target clustering is $\mathcal{C}^*$. Then

$$\#\ \textit{of merge requests} \ \leq\ 2(\delta_u(\mathcal{C}) + |\mathcal{C}^*|) \log_{1/(1-\eta)} n$$

Each merge is either:

- Pure: both clusters are marked 'pure.' Creates a single pure cluster.
- Impure: one of the clusters is marked 'impure.' Creates at least one pure cluster.

So # of impure merges $\leq (\delta_u(\mathcal{C}) + |\mathcal{C}^*|) log_{1/(1-\eta)} n$

And # of pure merges $\leq$ # of pure clusters $\leq$ # of impure merges

Split-and-merge bounds

Combining the lemmas, we have

$$\text{total \# of interactions} \;\leq\; \delta_o(\mathcal{C}) + 2(\delta_u(\mathcal{C}) + |\mathcal{C}^*|)\log_{1/(1-\eta)} n.$$

Often much less than specifying a clustering directly.

Active research directions

- Rates for ‘aggressive’ nonparametric active learning
- Interaction for other types of structures