Lecture 14
Clustering Possibilities: Impossibilities
Intro to Gaussian Mixture Models

Outline:
- Implications of Kleiner’s impossibility theorem
  - What’s “wrong” with allowing partitions that refine each other?
  - How should we think about the clustering problem, given the impossibility result?

Sidebar: Proving a subpart of the impossibility theorem.
- Reason: The kind of reasoning that can help in A2:
  - A souped-up version of
  - Creating a dataset that forces a clustering algorithm to output a desired partition
    - See previous lecture’s notes for proof (or the actual paper)
- (Gaussian) mixture model
  - How does having prior information or assumptions change how we think about clustering?

Pg 2 of handout: Three properties of clustering functions we discussed last time.

Ex: \( X = \{1, 2, 3, 4\} \) partitions of \( X \): \( \{\text{“clustering”} = \text{“partition”}\)
Recall: clustering functions take as input a distance function \( d \) and output a partition of \( X \).

Review the three properties on the handout:

1. **Richness**: there's no clustering/partitioning of \( X \) that can't be achieved by \( f \) by getting the right \( d \).

   - or,
   - for any partition you might want a priori (w/o looking at the data), there's some distance function \( d \) that causes \( f \) to output that target.

   so, an "expressiveness" property

2. **Scale-invariance**:

   - the distance function output shouldn't depend on the units (miles vs. mm) that the distances represent.

   - or,
   - the output doesn't change if you scale all the distances by the same amount.

   so, a "stability" property

3. **Consistency**:

   - if you change the \( d \) suppose you have a \( d \) that gives up

   - suppose \( f \) produces a partition \( P \) when given \( d \).

   If we alter \( d \) so that the clusters of \( P \) get even more "packed together" and the clusters of \( P \) grow even further apart from each other,

   so, a "stability" property.
thm 3.1 + thm 3.2 = (scale-invariance + consistency) \leq "pseudo-richness" \leq richness.

pseudo-rich: possible output sets are the antichains for X.

sets of partitions where no partition refines another.

implications:

Cluster q:

Can we have a \( f \) whose range is an anti-chain \( A \) where \( |A| > 1 \) and \( x \in \text{range}(f) \)?

All points in one big cluster

class went 70/80/20 on this.

No. Every other partition of X refines X.

\( \{1,2,3\}, \{4,5\} \) refines \( \{1,2,3,4,5\} \).

\( \{1,2,3\}, \{4,5\} \) refines \( \{1,2,3,4,5\} \).

[skip, but for those reading these notes:

You also couldn't have \( |A| > 1 \) and the partition \( P \) where every point is in its own cluster,
because \( P \) singletons refines any other partition

You can have \( A \subseteq \) set of all \( \mathbf{g} \) k-partitionings
(2 partitions of X that both consist of k clusters
can't possibly refine each other).

... and hey, that coincides with
the fixed-k-means! fixed-k
single-link algs that are so famous

Why is refinement such a problem for scale-invariance and consistency?

Some examples for intuition.

\( \square \): a partition
\( \bigcirc \): another partition, refining \( \square \).

Whether all-in-one or each-in-one seems to be a judgment about whether the distances are "big" or "little" in an absolute sense.

\( \square \) is a partition
\( \bigcirc \) another partition, refining \( \square \).

Whether to pick or \( \bigcirc \) seems to depend on whether the left-hand subcluster distances cause "subclusters" to grow unlike.
So, should we just give up on clustering?

Don't think so; it's just too useful a task - it tells you about the structure of your data.

It's just: you either have to:

- give up on some of the properties
- formulate your own properties
- break the formulation

... including, perhaps what the theorem says we need is more information than just a distance function. (an example: what to pre-determine k.)

After all, when we are clustering, is our goal really:

*just put points that are close together in the same group*?

or is it:

*try to determine the underlying structure of the data*.

It seems like a more fulfilling thing to do.

perhaps it has become time to consider a big question:

Where does this data come from?
When does our data come from? (the birds, bees talk)

What kind of generative story or account can we make up that's reasonable for how clusters underly the data?

Ex: generative story: **Parentage**

- **pick breed - nature selects**
- we have tree species: oaks, maples, and apple trees.
- Mother Nature picks a species
- picks a tree of that species, to be a parent
- plants a seed around the location of that parent.

Ex: generative story: **Gaussian sources**

- Assume: we have species, oaks, maples, apple trees.
- Explain distribution of height, width, leaf lengths, leaf widths.
- M.N. picks a species, grows a tree of that species, chooses the height, width values as from appropriate Gaussian for that species - i.e. each species has a template (with variance)

**Table with mixture models**

<table>
<thead>
<tr>
<th>Gaussian observed data</th>
<th>w/o- identifying labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>clustering? recover the underlying parameters of the &quot;true&quot; model?</td>
<td></td>
</tr>
</tbody>
</table>

- in this case, to figure out which trees belong to the same species?

- don't see the species!?
But there's a problem with thinking of our quest as to recover 'the' source model, and that is: it's an impossible problem.

Ex: top figure of pg 4:

assume we're dealing with one-dimensional data.

black dots = data, all on one line.

and we tell you that the data was generated by two Gaussians, so one cluster should be the points generated by one of the Gaussians, and the other cluster should be the points generated by the other Gaussian.

So, one possibility is that we have a Gaussian centered under the left bump of the green curve, and a Gaussian centered right, this would be a plausible way the black dots got generated, it does look like there's a rightmost clump of points and a leftmost clump.

But, it really could be that the true source had one Gaussian a mile to the left, and one Gaussian a mile to the right, so both totally off the screen.

We cannot say that that's not the source model, it could be that that was the source model, and we just got a really unlikely draw of the data.

So, we can't say that we can select the true source.

However, if we have to pick some choice of how to decide what model to settle on in order to create output, then one idea is to use the maximum-likelihood principle. (See on handout)

Assume fixed model class, where elements are indexed by parameter setting θ, (a vector)

Each possible source

Ex: Gaussian mixture model w/ 2 Gaussians w/ fixed variances, one chosen w/ fixed prob 2/3, one w/ fixed prob 1/3

⇒ possible θ's are the: the means of the two Gaussians.

Ex: green curve: θ = (−2, 3) → left mean

resulting densities shown on handout.
another possible choice would be our "miles-away" alternative

\[ \Theta = (-\infty, \infty) \]

\[ \rightarrow \text{ mean of 1st Gaussian} \rightarrow \text{ mean of second Gaussian}. \]

for given data \( X_{\text{givn}} \), pick model \( \Theta \) that maximizes \( P(X_{\text{givn}} | \Theta) \)

\[ \hat{\Theta} = \arg \max_{\Theta} P(X_{\text{givn}} | \Theta) \]

thus, since the green-curve model thinks \( X_{\text{givn}} \) is pretty likely,

whereas the \( X_{\text{givn}} \) is very unlikely according to the "miles-away" model,

MLP says to prefer the green-curve model over the miles-away model.

How to find the \( \hat{\Theta} \)? "Derivative":

let \( X_{\text{givn}} = (x_1, x_2, \ldots) \) for notation sake

let \( \Theta = (\Theta_1, \Theta_2, \ldots, \Theta[l], \ldots) \)

"Derivative":

\[ \frac{\partial}{\partial \Theta[l]} P(X_{\text{givn}} | \Theta) \quad \text{and set to } 0, \text{ solve for } \Theta[l]. \]

but a trick:

- almost always a good idea to use the log of the likelihood instead

(and since log is monotonic increasing in its argument,

make no diff in finding the argmax)

Handout: the log-likelihood function for our two-Gaussian model, \( \frac{2}{3} \) vs. \( \frac{1}{3} \) prob:

\[ \Theta = (\mu_1, \mu_2) \rightarrow \mu, \text{ is shown as } x \text{ axis, } \mu_2 \text{ shown as } y \text{ axis.} \]

\[ p(x | \Theta) = \frac{1}{3} e^{-\frac{1}{2}(x-\mu_1)^2} + \frac{2}{3} e^{-\frac{1}{2}(x-\mu_2)^2} \]

prob of choosing 1st Gaussian

prob of \( x \) if generated by 1st Gaussian

\[ \log P(x | \mu_1, \mu_2) \text{ is the surface plot.} \]
in orange: the application of an iterative method for trying to do gradient ascent to find a local max of the log-likelihood.

Then are two trajectories, corresponding to two different starting points.

\[ \mu_a, \mu_b \] are the two places this search converged.

\[ \mu_a, \mu_b \]

- on the contour plot below the surface.

(in our notation, \( \mu_a \) would be written \( \Theta_a \), and \( \mu_b \) as \( \Theta_b \)).

- on the figure above, the distribution of densities for \( \Theta_a \); \( \Theta_b \) are shown:

\[ \Theta_a \text{ has } \mu_1 = 2, \mu_2 \geq 2 \]

\[ \Theta_b \text{ has } \mu_1 = -1, \mu_2 \geq 2 \]

Note: the iterative method mentioned above and whose trajectories are depicted in the handout does not make use of latent variables. It instead uses the hideous equation whose existence is mentioned in the next lecture.