Hypothesis Space

• Continuous inputs
• Output is a binary tree with data points as leaves.

• Useful for explaining the training data.
• Not useful for making new predictions.
Direction of Clustering

Two basic algorithms for building the tree:

Agglomerative (bottom-up)

• Each point starts in its own cluster.
• Repeatedly merge the two most-similar clusters until only one remains.

Divisive (top-down)

• All points start in a single cluster.
• Repeatedly split the data into the two most self-similar subsets.
Agglomerative Clustering

• Each point starts in its own cluster.
• Repeatedly merge the two most-similar clusters until only one remains.

• How do we decide which clusters are most similar?
  • Need a measure of similarity between points.
  • Need a measure of similarity between clusters of points.
Similarity Measures

• Euclidean distance (2-norm)

• Other distance metrics
  • 1-norm, ∞-norm

• Cosine similarity
  • Cosine of the angle between two vectors.

• Edit distance
  The smallest number of mutations/deletions/insertions to transform between two words.
  • Good for genomes and text documents

\[ \text{aaabbb } \Rightarrow \text{aabab} \]
Requires 3 edits.
$p$-norm

\[ \|x\|_p \equiv \left( \sum_{i=1}^{d} |x_i|^p \right)^{\frac{1}{p}} \]

$p = 1$ Manhattan distance

$p = 2$ Euclidean distance

$p = \infty$ largest distance in any dimension
Cluster Similarity

If we’ve chosen a point-similarity measure, we still need to decide how to extend it to clusters.

- Distance between closest points in each cluster (single link).

- Distance between farthest points in each cluster (complete link).

- Distance between centroids (average link).
Which clusters should be merged next?

- Under single link?
- Under complete link?
- Under average link?

Use Euclidean distance.
Divisive Clustering

• All points start in a single cluster.
• Repeatedly split the data into the two most self-similar subsets.

How do we split the data into subsets?
• We need a subroutine for 2-clustering.
• Options include k-means and EM.
Strengths and weaknesses of hierarchical clustering

+ Creates easy-to-visualize output (dendrograms).
+ We can pick what level of the hierarchy to use after the fact.
+ It’s often robust to outliers.
- It’s extremely slow: the basic agglomerative clustering algorithm is $O(n^3)$; divisive is even worse.
- Each step is greedy, so the overall clustering may be far from optimal.
- Doesn’t generalize well to new points.
- Bad for online applications, because adding new points requires recomputing from the start.
Growing Neural Gas

A different approach to unsupervised learning:
• Adaptively learns a “map” of the data set.

Start with a two-node graph, then repeatedly pick a random data point and:
1. Pull nodes of the graph closer to the data point.
2. Occasionally add new nodes and edges in the places where we had to adjust the graph a lot.
3. Discard nodes and edges that haven’t been near any data points in a long time.
Growing Neural Gas Demo

https://www.youtube.com/watch?v=1zyDhQn6p4c
Growing Neural Gas Algorithm

Start with two random connected nodes, then repeat 1...9:

1. Pick a random data point.
2. Find the two closest nodes to the data point.
3. Increment the age of all edges from the closest node.
4. Add the squared distance to the error of the closest node.
5. Move the closest node and all of its neighbors toward the data point.
   • Move the closest node more than its neighbors.
6. Connect the two closest nodes or reset their edge age.
7. Remove old edges; if a node is isolated, delete it.
8. Every $\lambda$ iterations, add a new node.
   • Between the highest-error node and its highest-error neighbor
9. Decay all errors.
Adjusting nodes based on a data point
Adjusting nodes based on a data point

This node’s error increases

These edges’ ages increase

This edge’s age is set to zero

If age is too great, delete the edge.
Every $\lambda$ iterations, add a new node
Consider the GNG hypothesis space

What does the output of the GNG look like?

What unsupervised learning problem is growing neural gas solving?

- Is it clustering?
- Is it dimensionality reduction?
- Is it something else?