**11.3**

**Mixture Models as Probabilities**

A "mixture" is just a Gaussian probability and a weight.

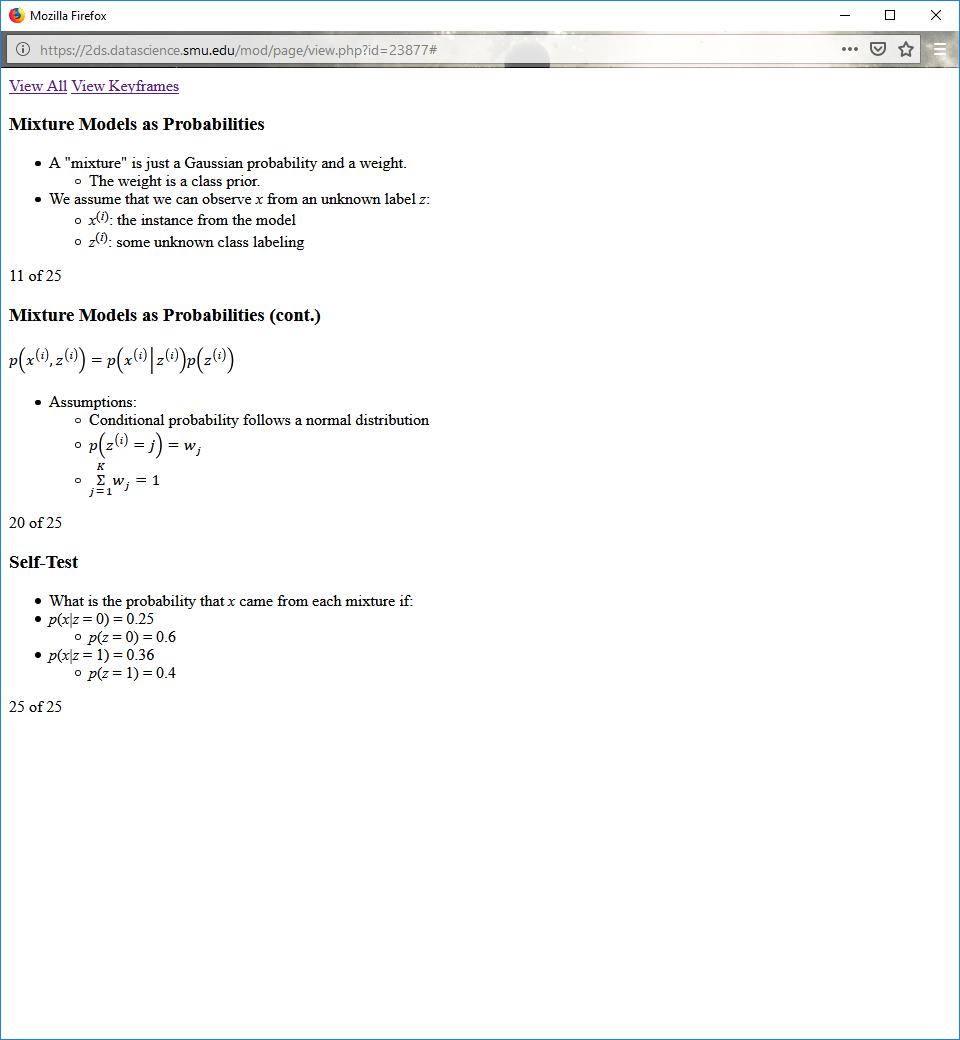
The weight is a class prior.

We assume that we can observe *x* from an unknown label *z*:

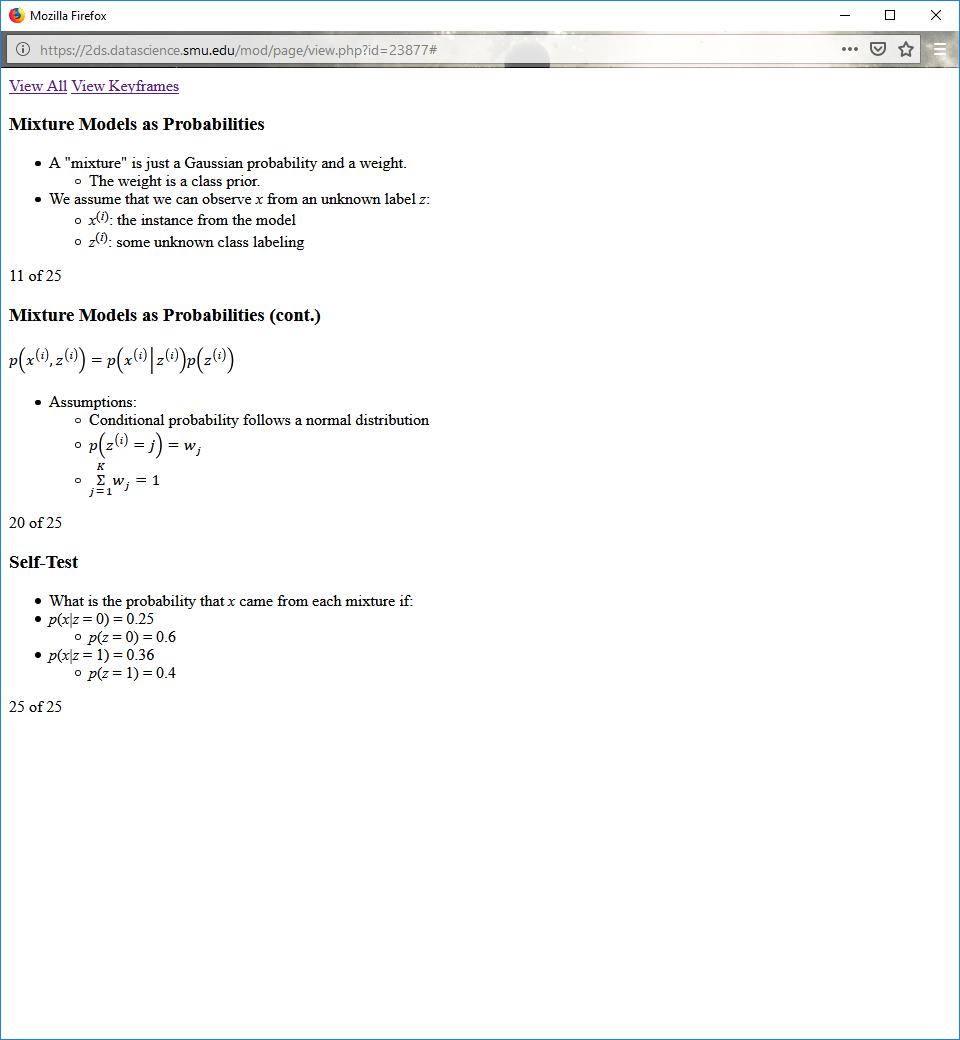
*x*(*i*): the instance from the model

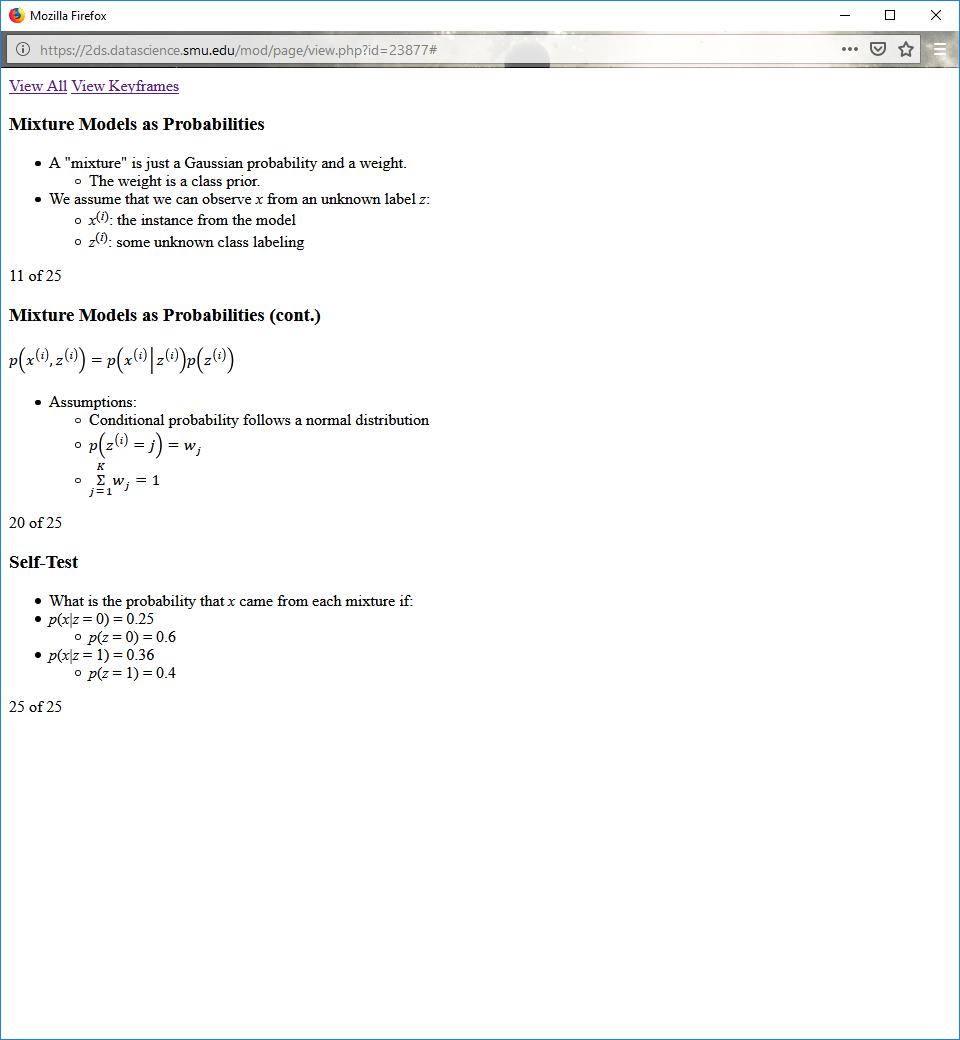
*z*(*i*): some unknown class labeling

**Mixture Models as Probabilities (cont.)**



Assumptions:

Conditional probability follows a normal distribution (this: )



**Self-Test**

What is the probability that *x* came from each mixture if:

*p*(*x*|*z* = 0) = 0.25

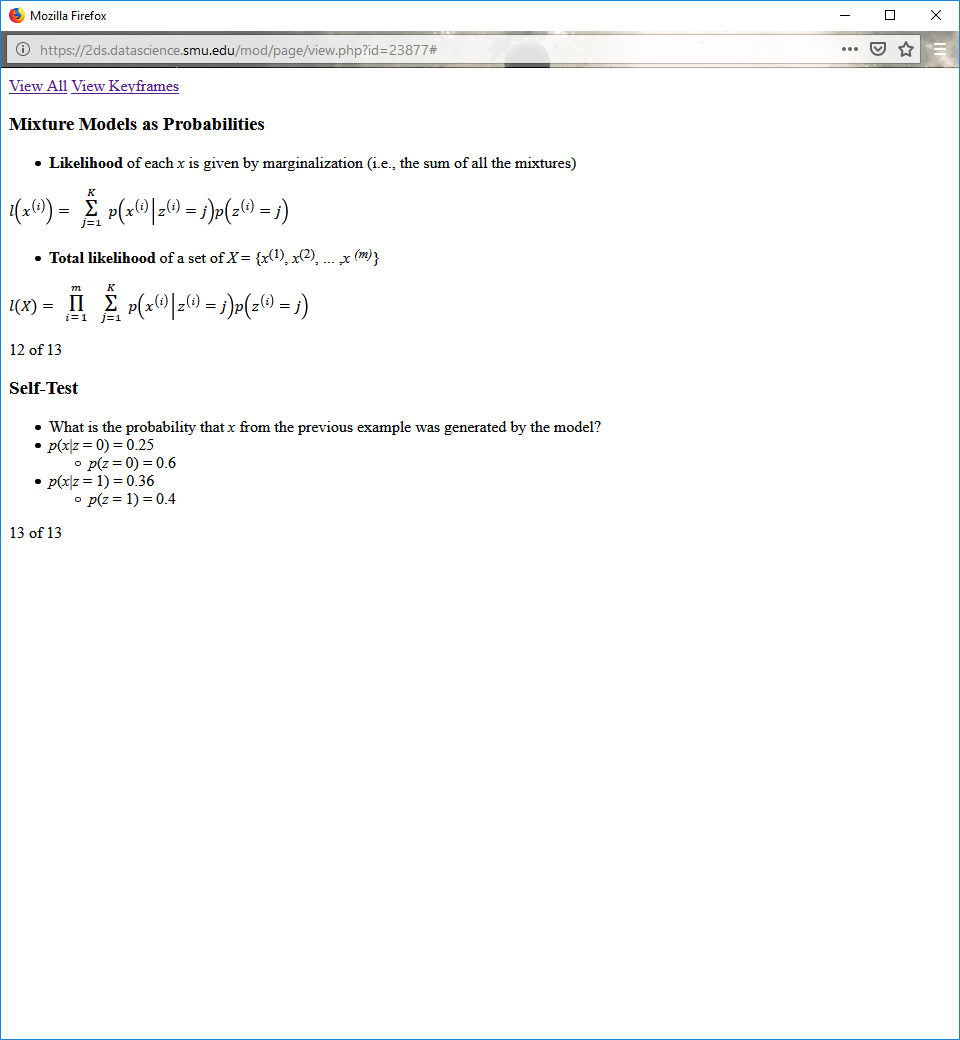
*p*(*z* = 0) = 0.6

*p*(*x*|*z* = 1) = 0.36

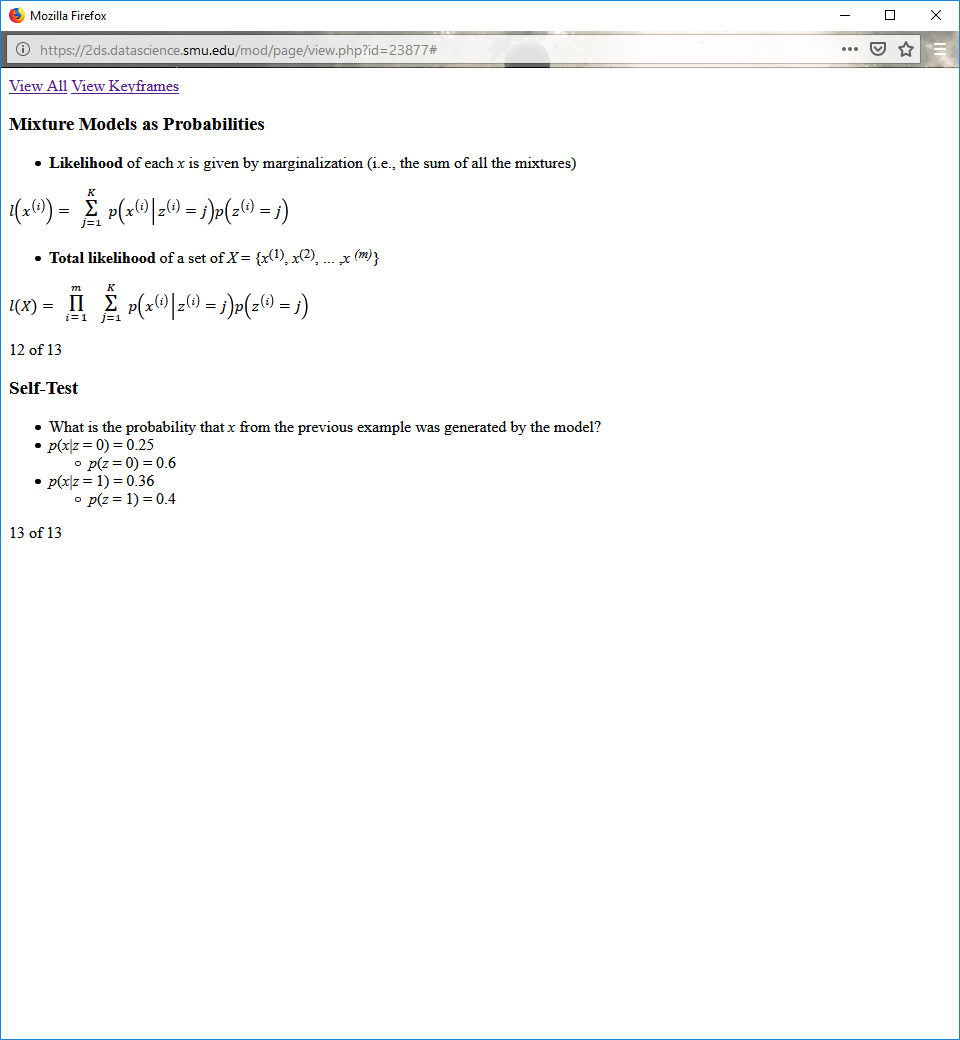
*p*(*z* = 1) = 0.4

**Mixture Models as Probabilities**

**Likelihood** of each *x* is given by marginalization (i.e., the sum of all the mixtures)



Total likelihood of a set of *X* = {*x*(1), *x*(2), ... ,*x* *(m)*}



**Self-Test**

What is the probability that *x* from the previous example was generated by the model?

*p*(*x*|*z* = 0) = 0.25

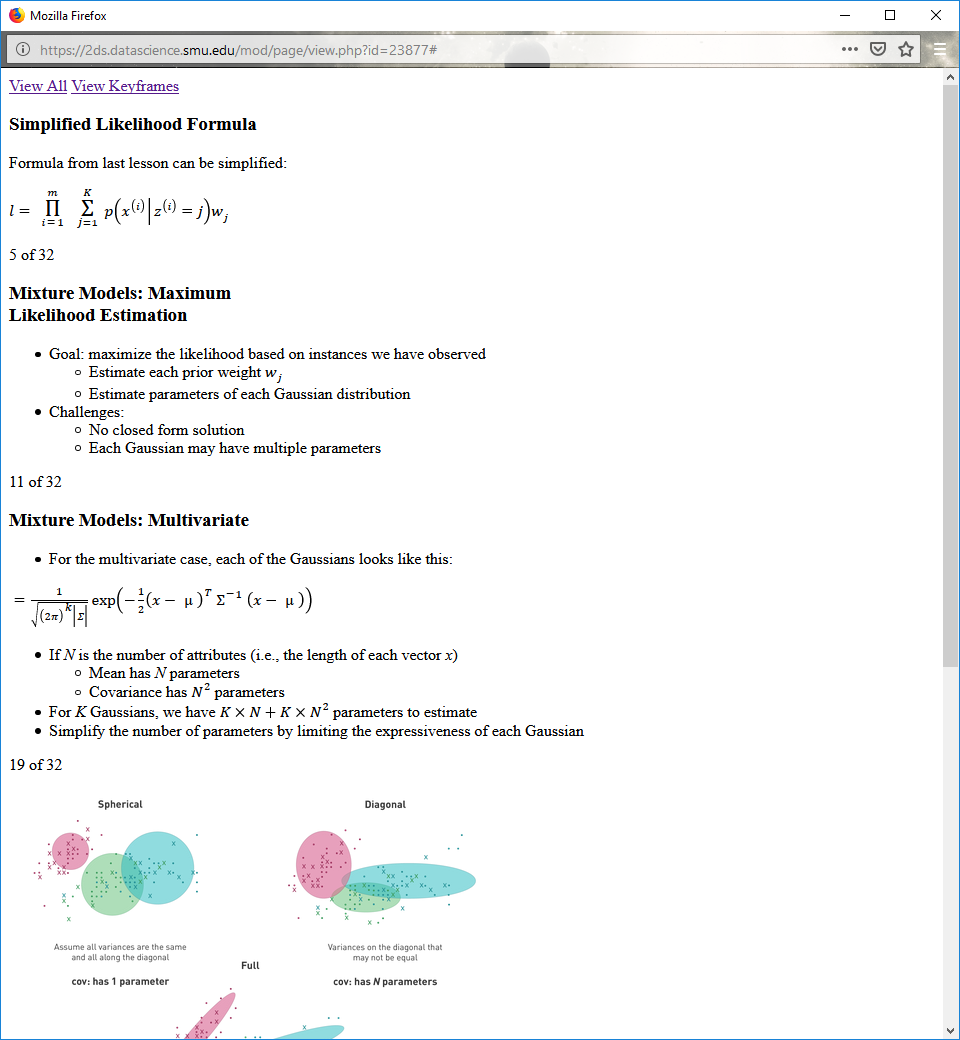
*p*(*z* = 0) = 0.6

*p*(*x*|*z* = 1) = 0.36

*p*(*z* = 1) = 0.4

**Simplified Likelihood Formula**

Formula from last lesson can be simplified:



**Mixture Models: Maximum Likelihood Estimation**

Goal: maximize the likelihood based on instances we have observed

Estimate each prior weight

Estimate parameters of each Gaussian distribution

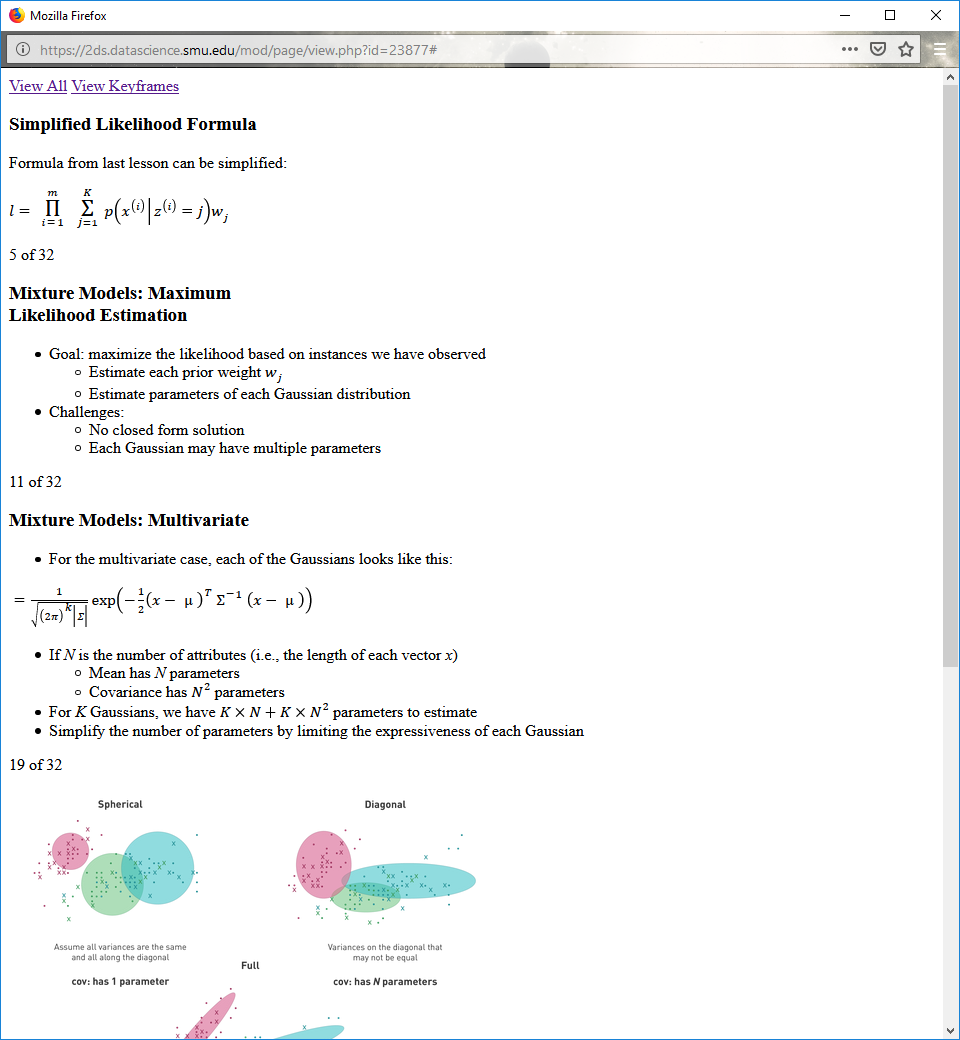
Challenges:

No closed form solution

Each Gaussian may have multiple parameters

**Mixture Models: Multivariate**

For the multivariate case, each of the Gaussians looks like this:

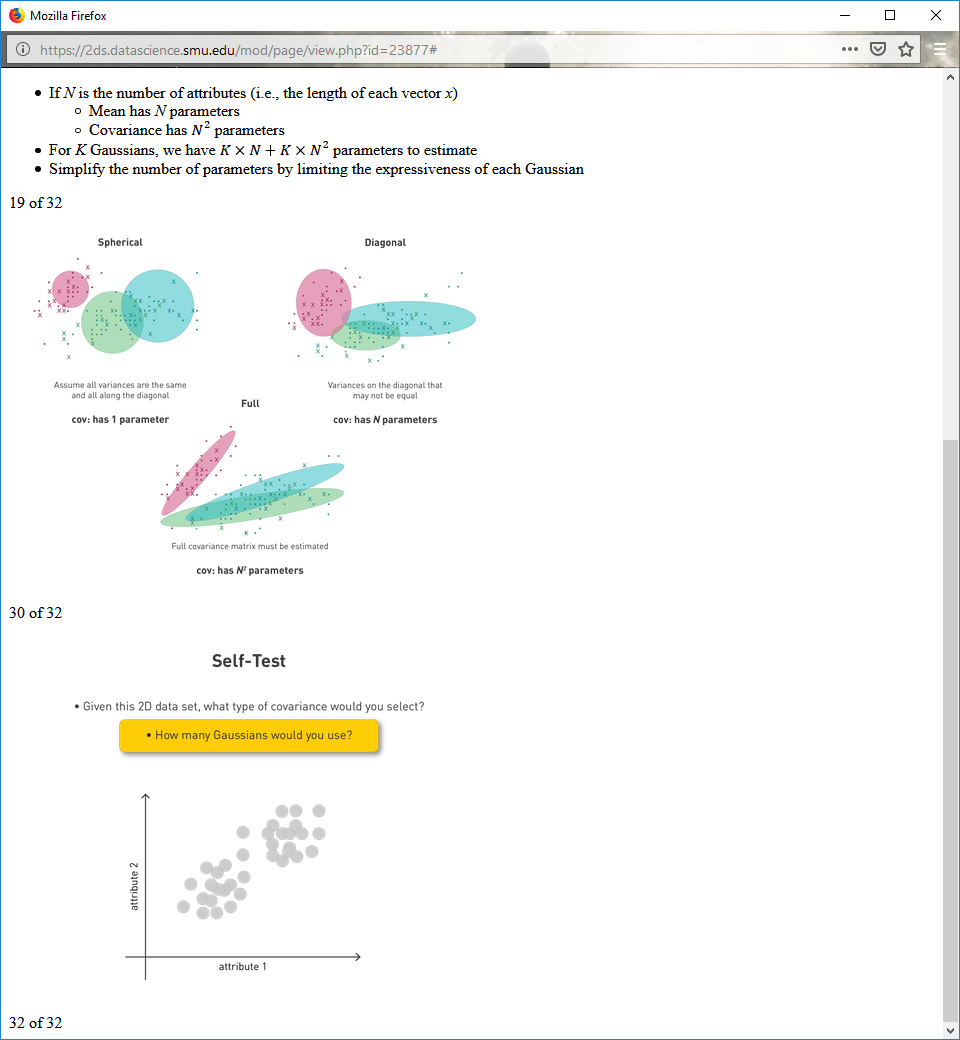
  
If *N* is the number of attributes (i.e., the length of each vector *x*)

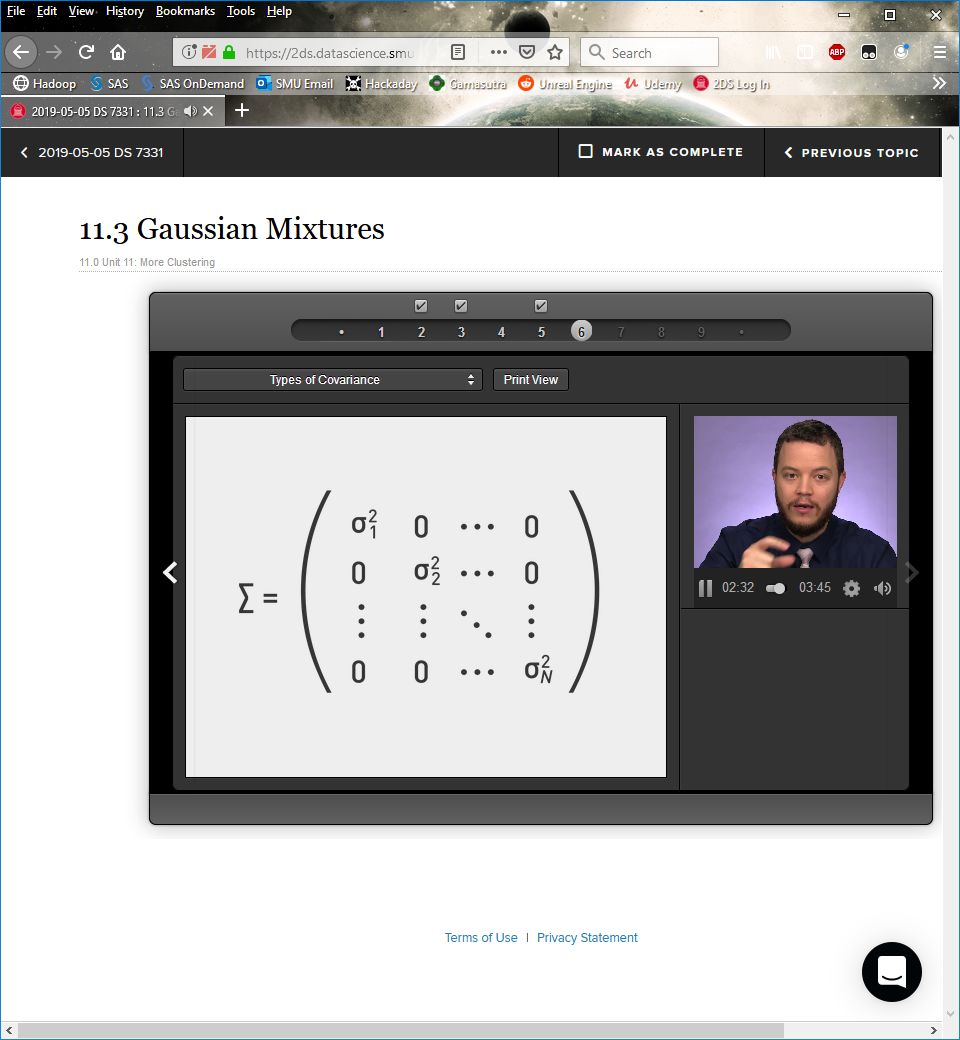
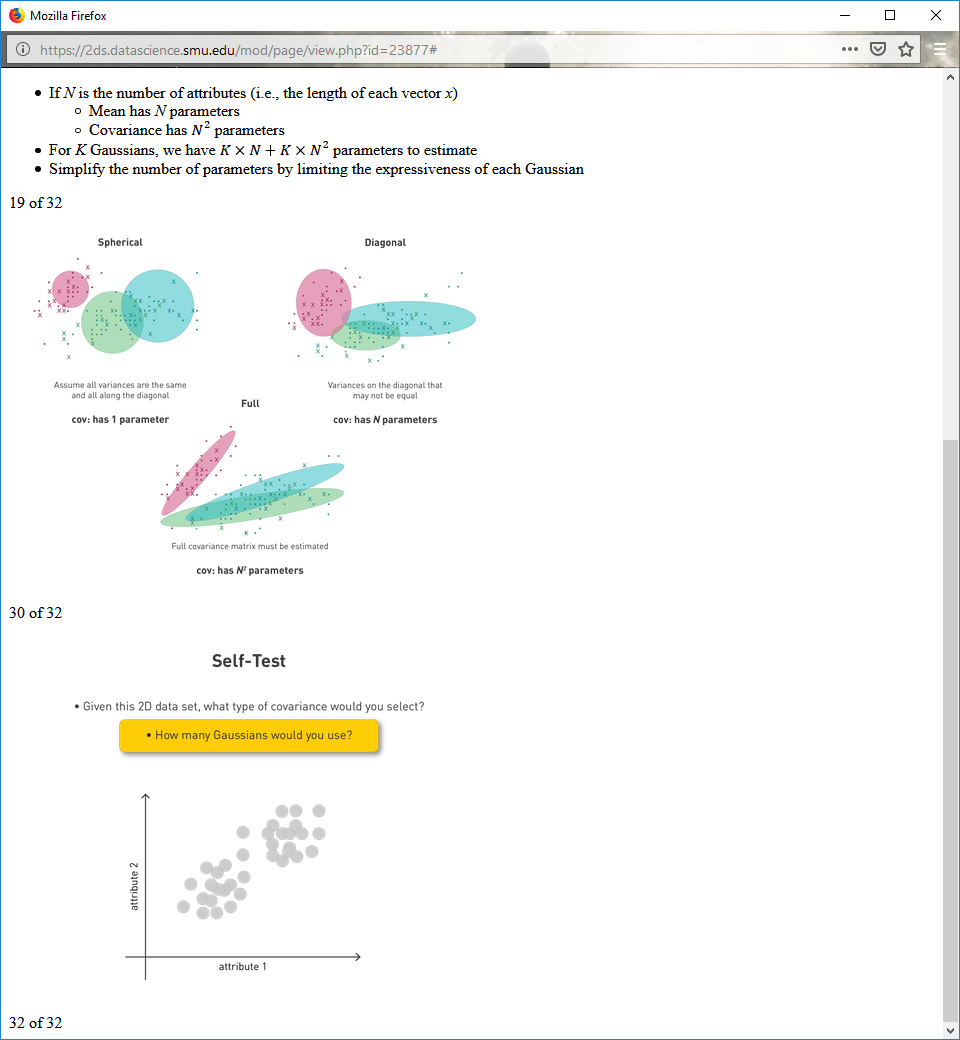
Mean has *N* parameters

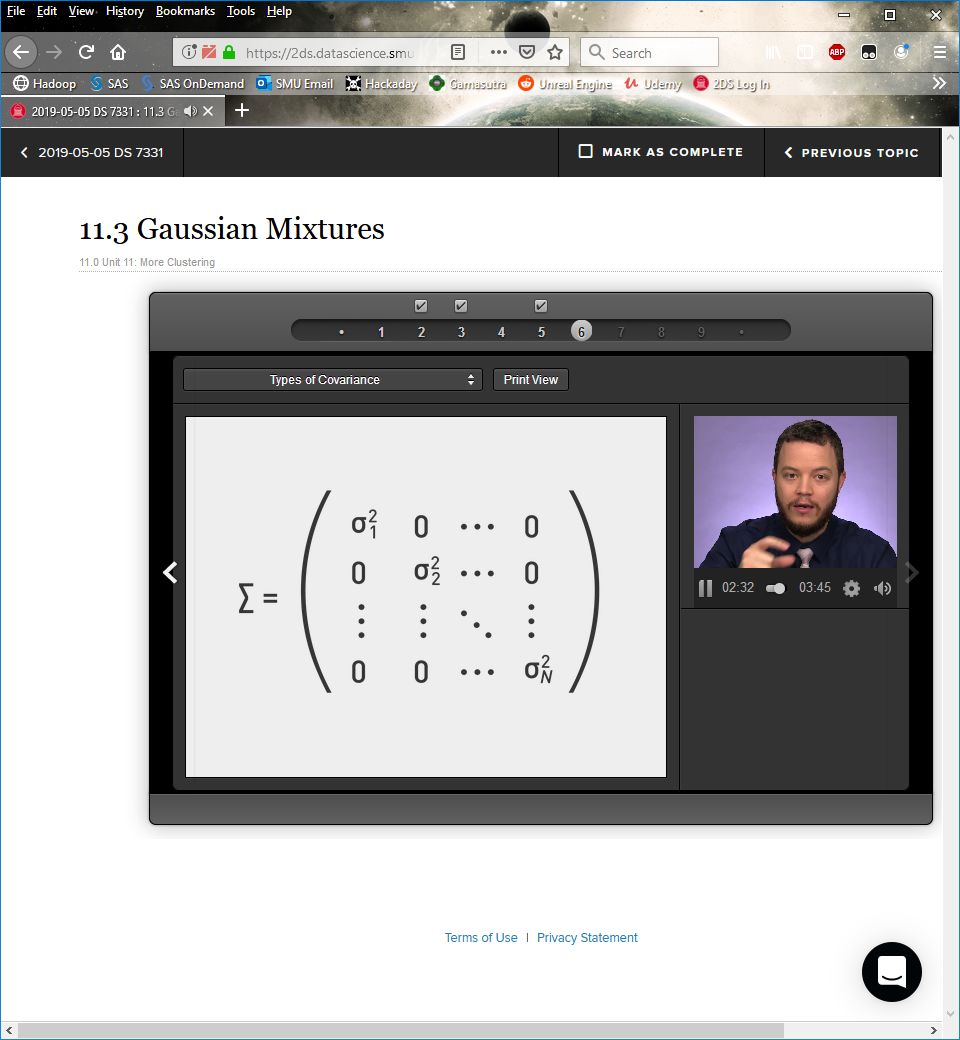
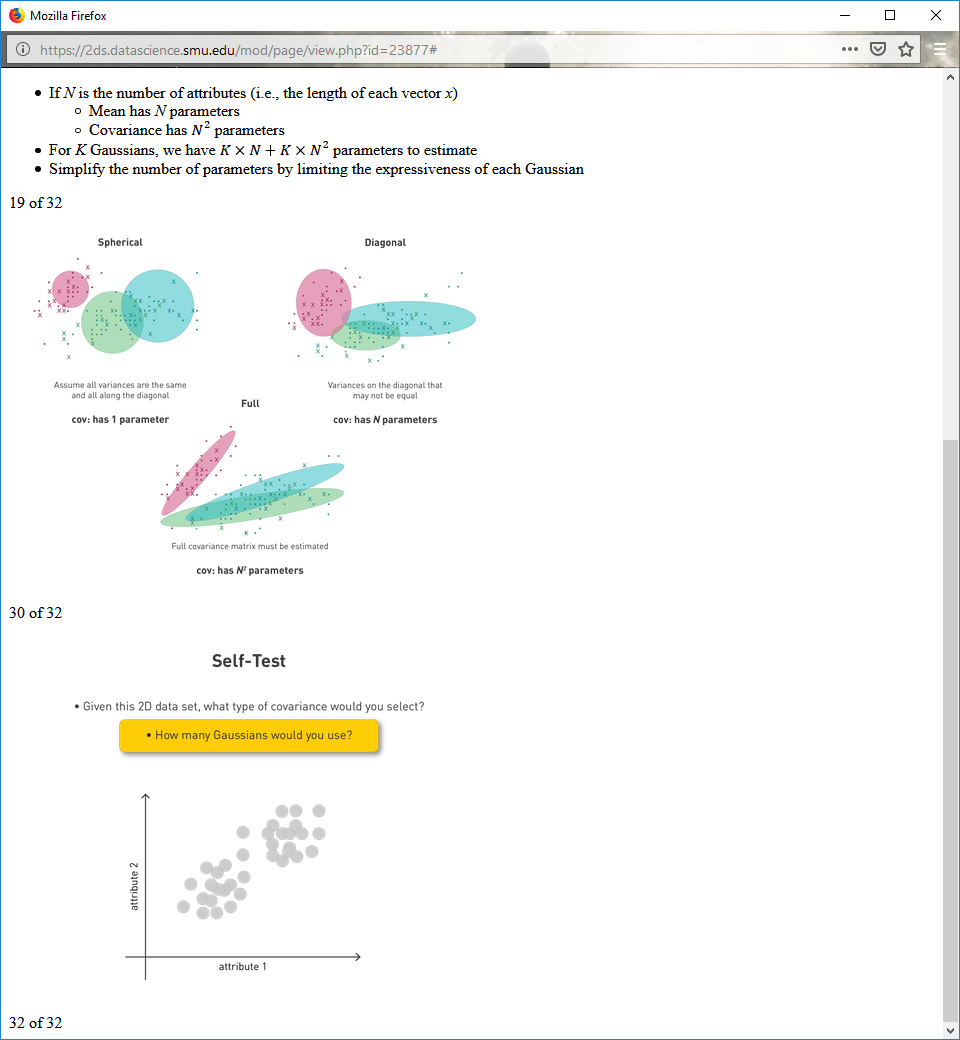
Covariance has parameters

For *K* Gaussians, we have K×N+K× parameters to estimate

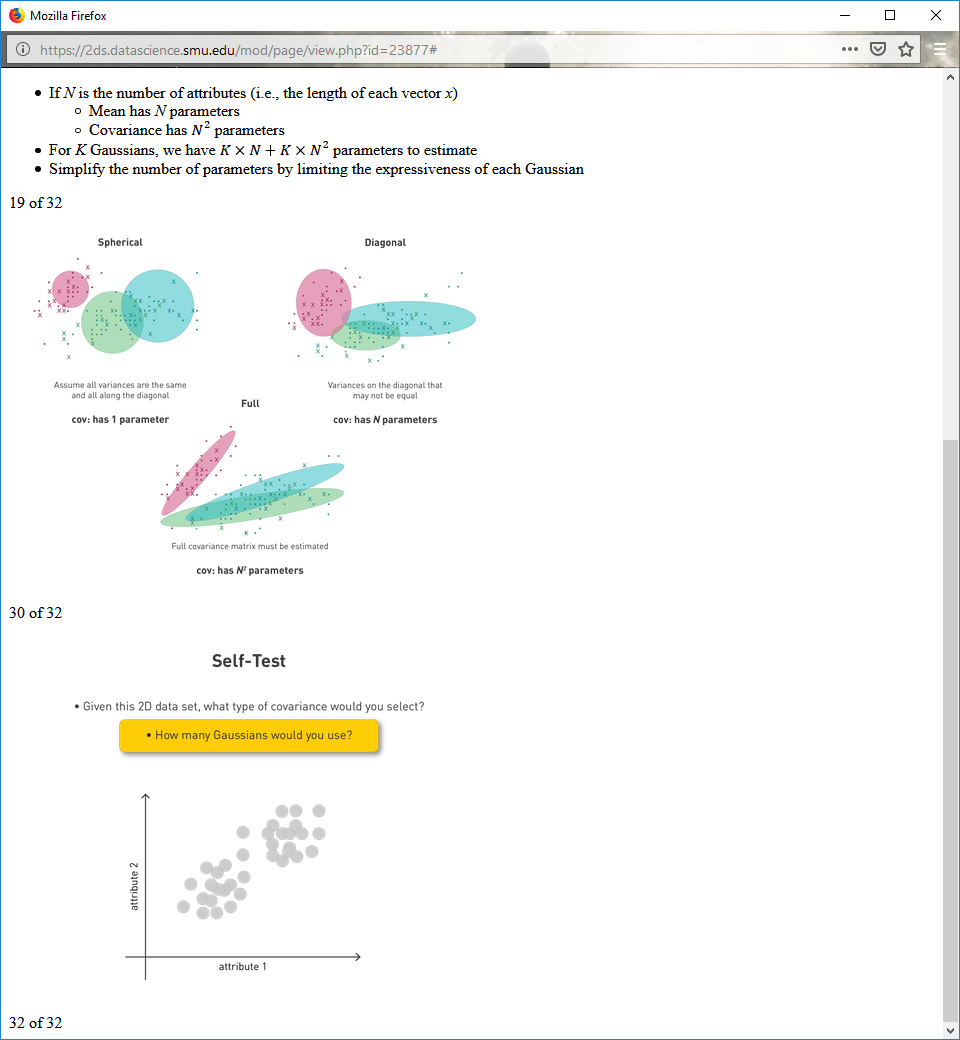
Simplify the number of parameters by limiting the expressiveness of each Gaussian







**Self Test**



**Expectation Maximization**

Iterative solution for updating the model with the maximum likelihood

**Init** centers of each Gaussian mixture

i.e., with K-means

**Repeat**

**Expectation:** assign each point to a cluster, calculate probability that point belongs to the cluster

**Maximization**: given the new probabilities, find the new mu, sigmas, and weights for each cluster that maximize expected probability

**Until** parameters do not change

**Disadvantages of Gaussian Mixtures**

Cannot scale up to handle large dimensional data sets (recommended to be 20 or less)

Require more time for covariance and determinant calculations

Can only handle globular clusters

Cannot easily capture small clusters or line-shaped clusters

May be distorted by spurious singularities

Limit the variance in order to exclude spurious singularities

**Advantages of Gaussian Mixtures**

More general than many other clustering types

Robust to density difference

More likely to create clusters that can easily be identified from their parameters

Many processes are the result of a Gaussian process.

**11.4**

**Graph-Based Clustering**

Hierarchical agglomerative clustering

Start with a proximity matrix

Make connections to view a MAX (complete-link) or MIN (single-link)

Decrease size of the proximity matrix at each iteration to connect components

**The Proximity Matrix**

Graph-based structure

Weight of every two-node connection is determined by the proximity matrix

Initially fully connected

Every node connects to every other node

This graph can be made more sparse

**Graph-Based Clustering: Sparsification**

Sparsification can eliminate more than 99% of the entries in a proximity matrix

Only about 1% of the information is useful for cluster analysis

Reduces the time required to cluster the data

Increases the size of problems that can be tackled

**Graph-Based Clustering: Implementation**

Measure the similarity of points

Nearest neighbors

Correlation

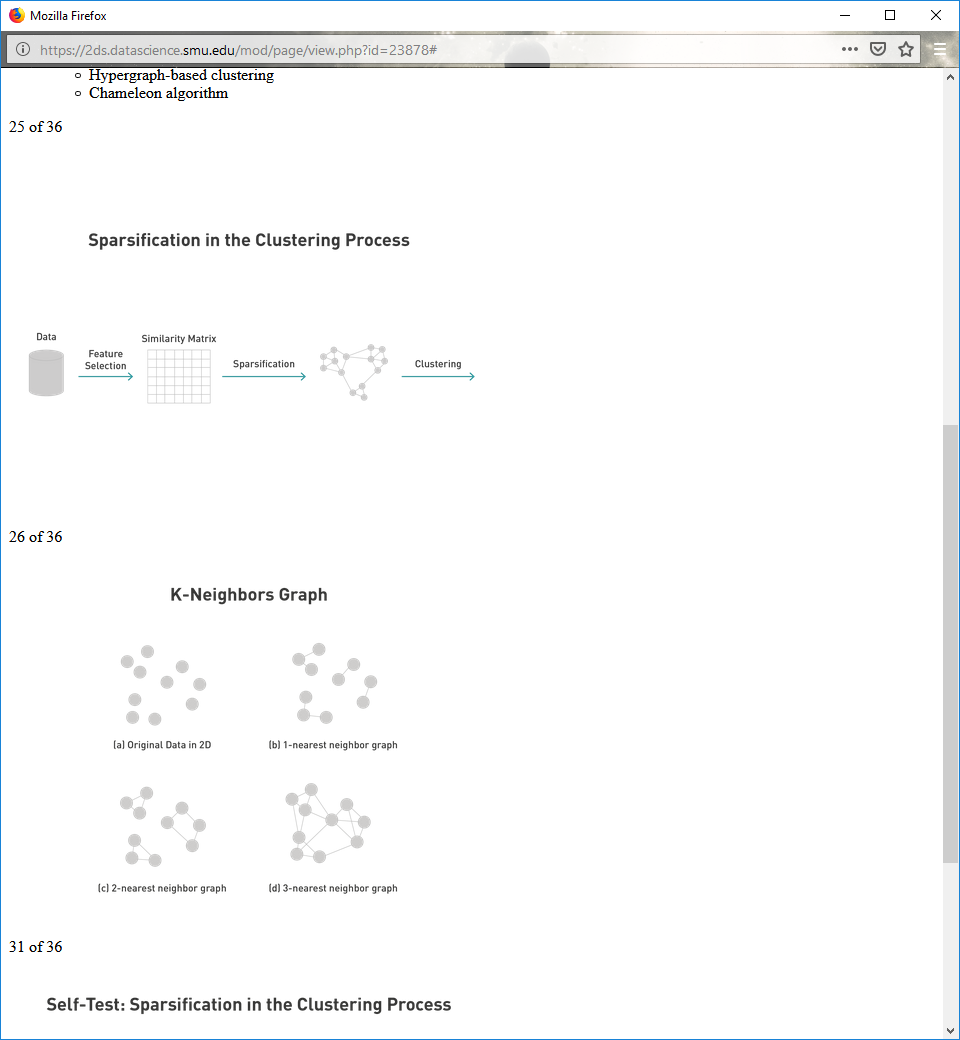
This reduces the impact of noise and outliers and sharpens the distinction between clusters

Sparsification facilitates the use of graph partitioning algorithms

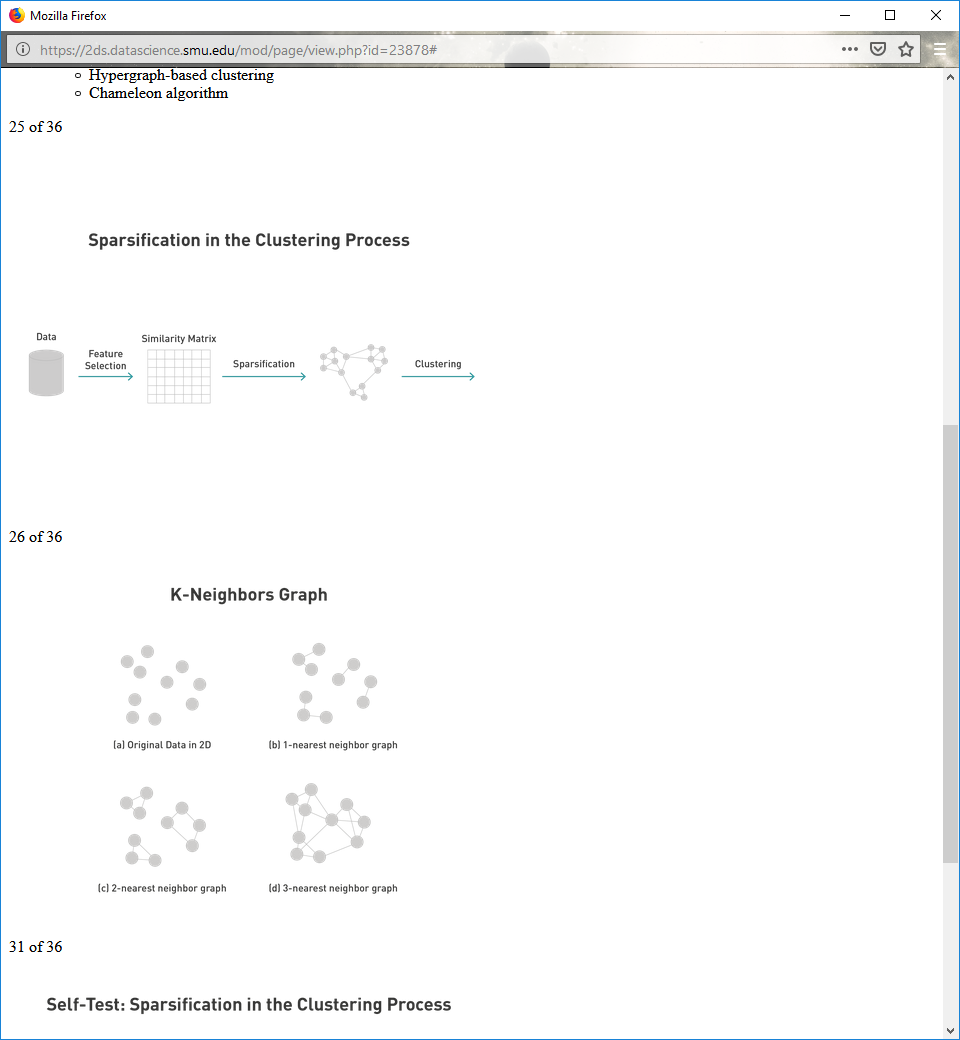
Hypergraph-based clustering

Chameleon algorithm

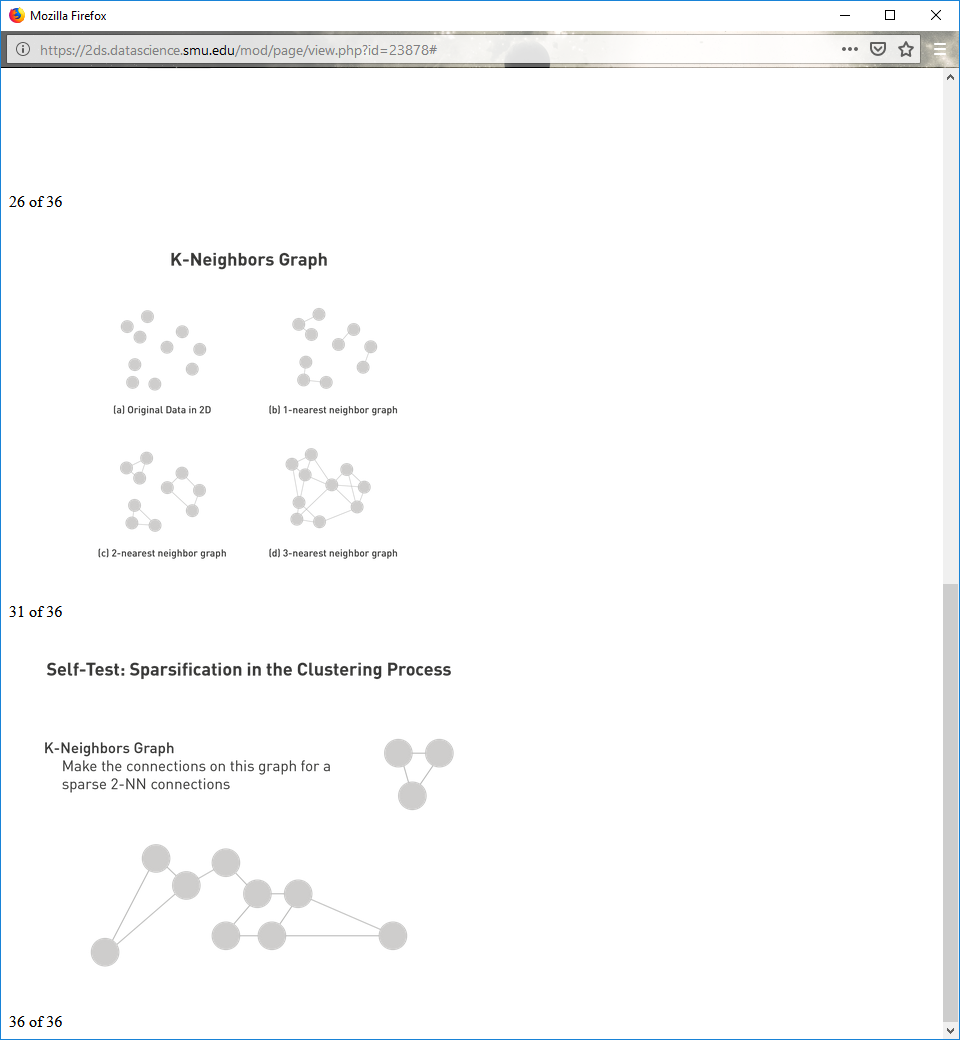
**Sparsification in the Clustering Process**



**K-Neighbors Graph**



**Self-Test: Sparsification in the Clustering Process**



**Limitations of Current Merging Schemes**

Hierarchical agglomerative clustering

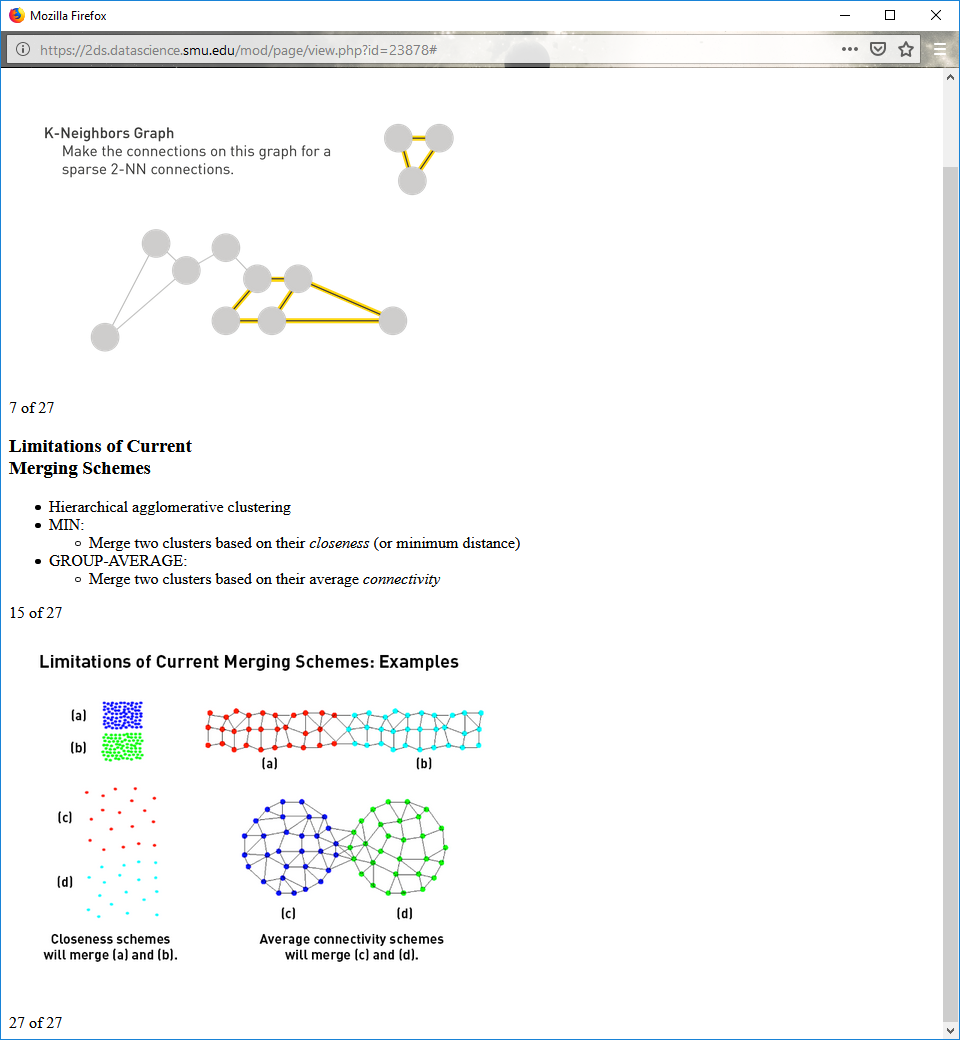
MIN:

Merge two clusters based on their *closeness* (or minimum distance)

GROUP-AVERAGE:

Merge two clusters based on their average *connectivity*

**Limitations of Current Merging Schemes: Examples**



**Chameleon: Clustering Using Dynamic Modeling**

Adapt to the characteristics of the data set in order to find the natural clusters

Use a dynamic model to measure the similarity between clusters

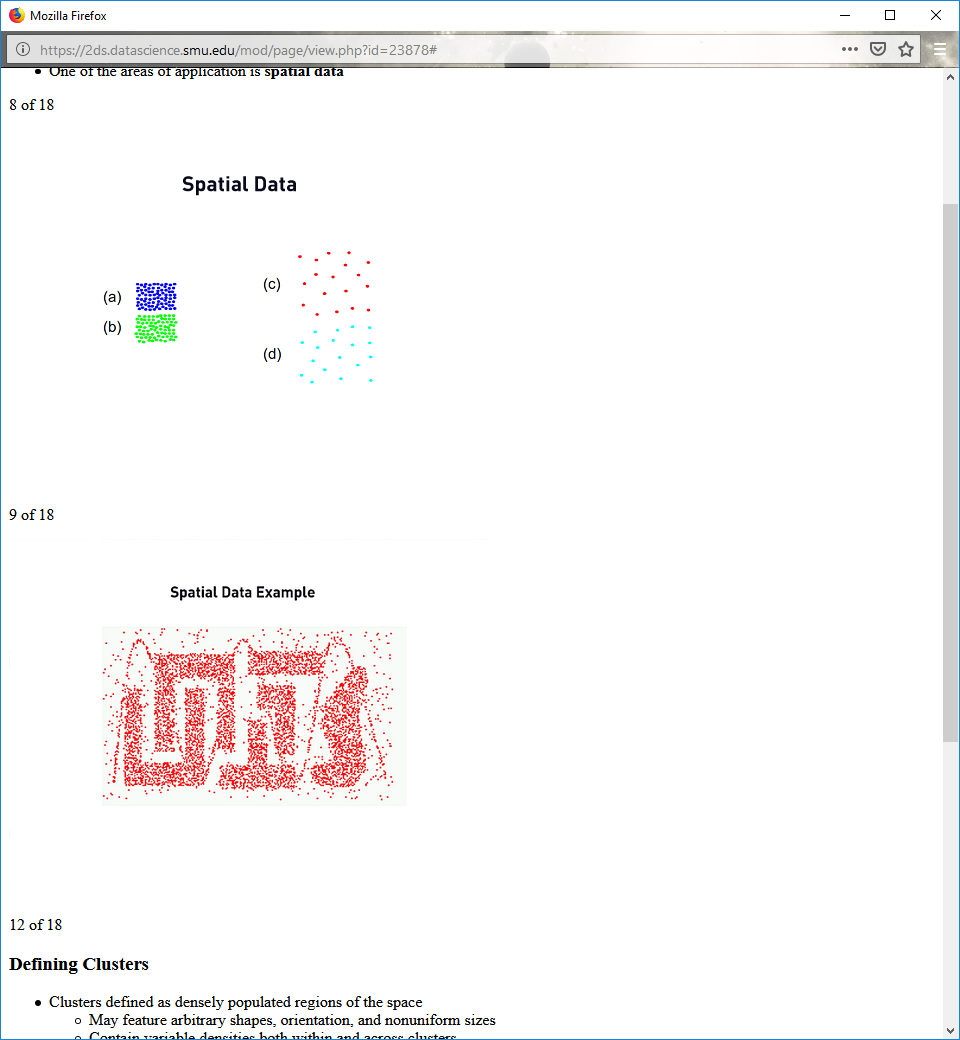
Key properties: relative closeness and relative interconnectivity

Two clusters are combined if the resulting cluster shares certain properties with the constituent clusters

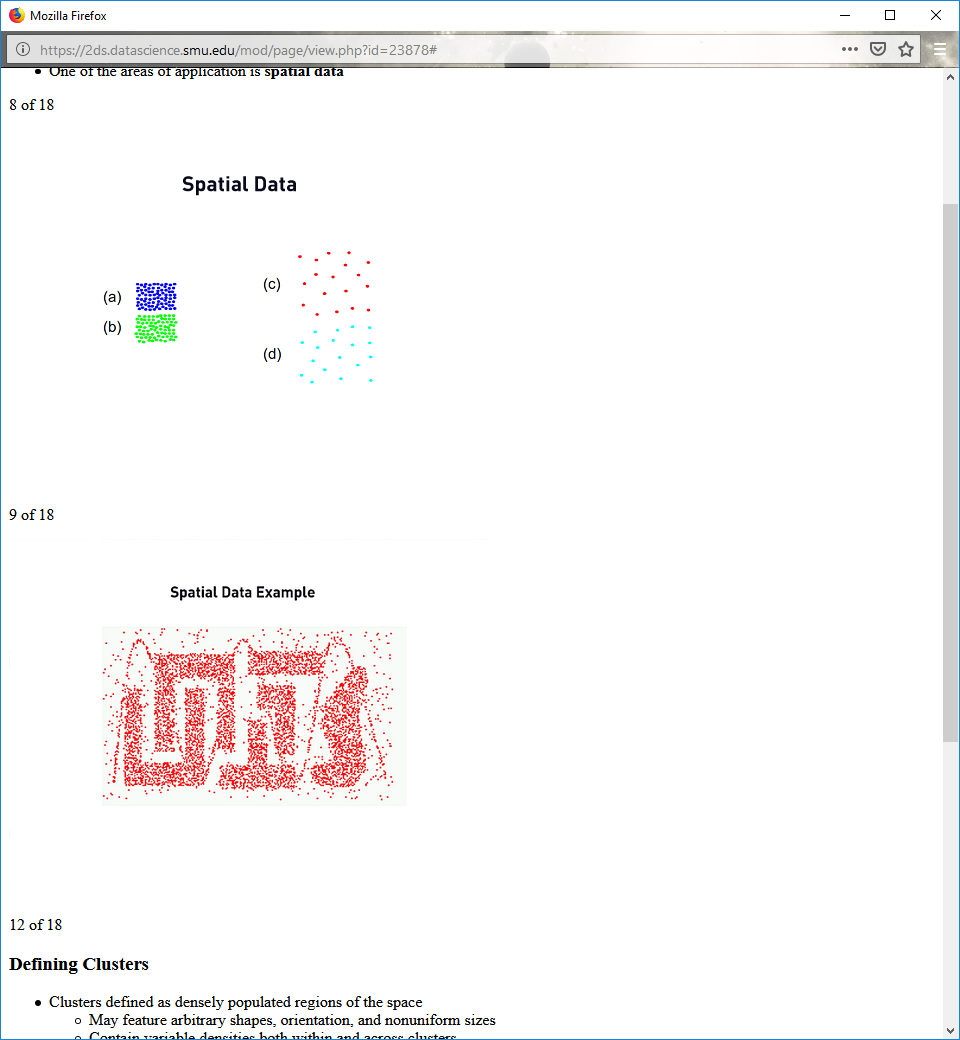
Merging scheme preserves *self-similarity*

One of the areas of application is **spatial data**

**Spatial Data**



**Spatial Data Example**



**Defining Clusters**

Clusters defined as densely populated regions of the space

May feature arbitrary shapes, orientation, and nonuniform sizes

Contain variable densities both within and across clusters

May contain special artifacts (streaks) and noise

**Chameleon: First Steps**

1. Make it sparse.

Represent the data by a sparse graph.

Construct the K-nearest-neighbor (KNN) graph.

2. Overcluster.

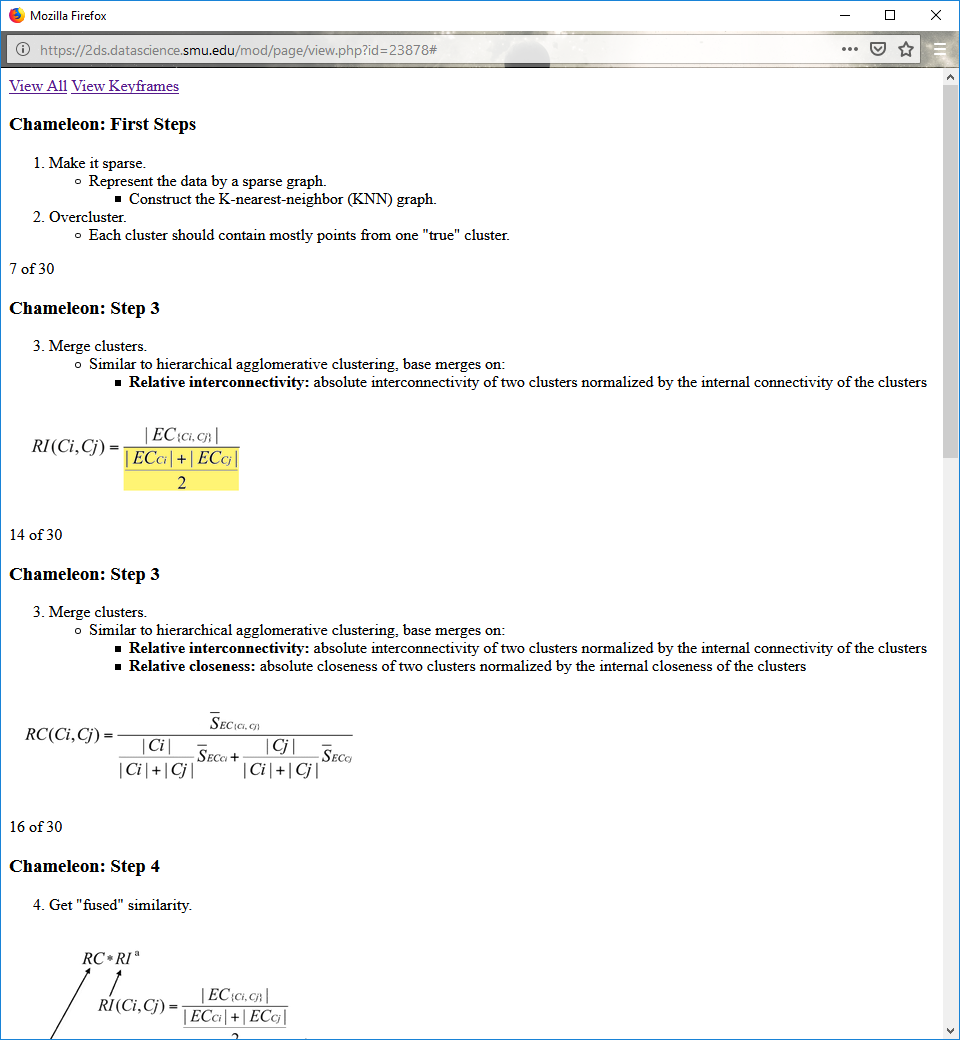
Each cluster should contain mostly points from one "true" cluster.

**Chameleon: Step 3**

3. Merge clusters.

Similar to hierarchical agglomerative clustering, base merges on:

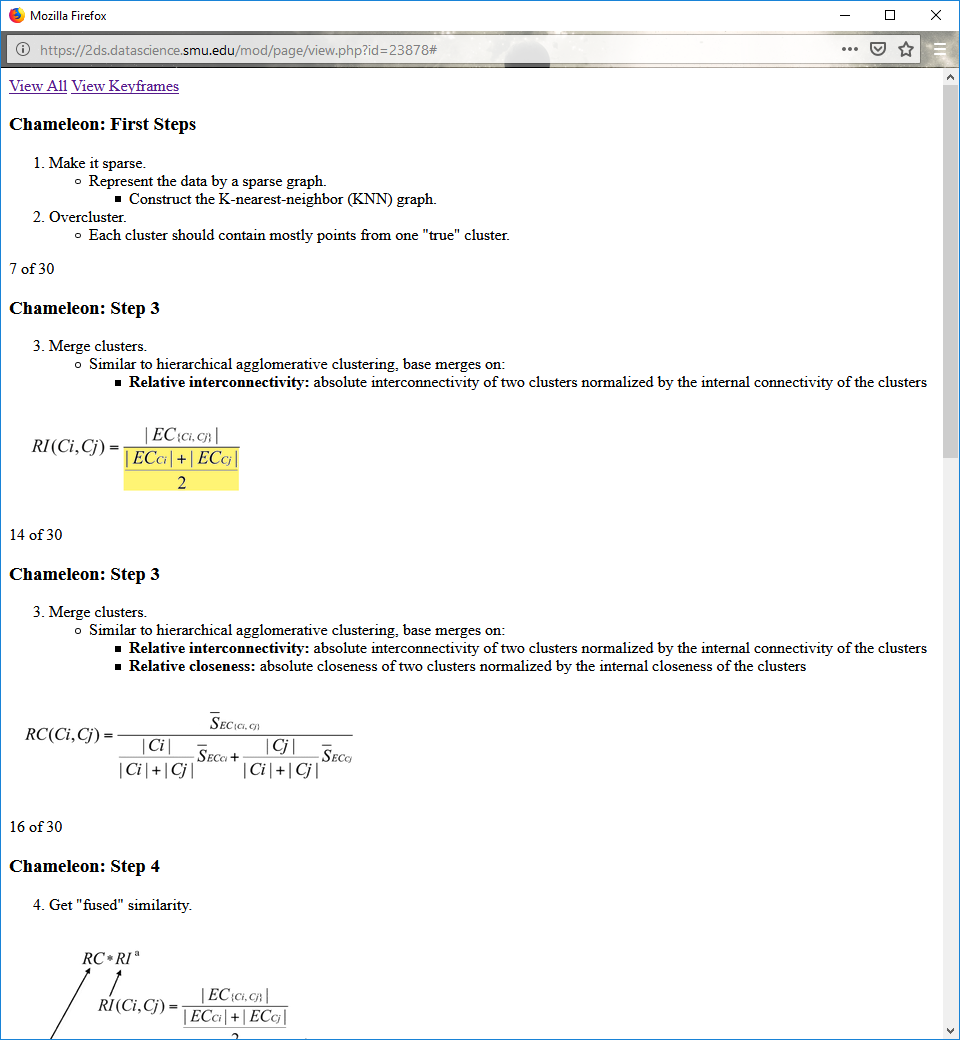
**Relative interconnectivity**: absolute interconnectivity of two clusters normalized by the internal connectivity of the clusters



Numerator: if we combined the two clusters, what is the number of edges inside that cluster (this is also called the volume of a cluster)

Denominator: The volume of the two clusters before they were merged divided by 2 (2 is the normalization constant)

**Relative closeness**: absolute closeness of two clusters normalized by the internal closeness of the clusters

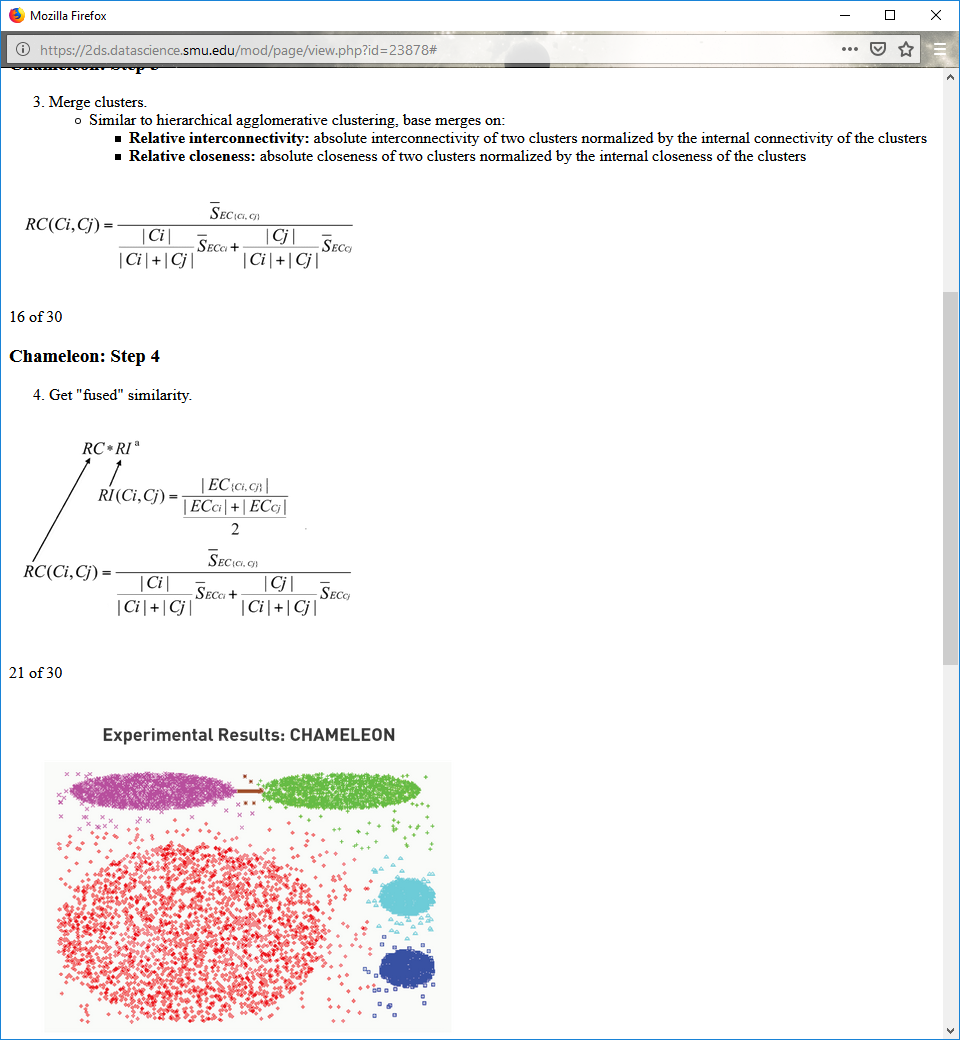


Numerator: Average weight of the edges after merging

Average weights before we clustered for each

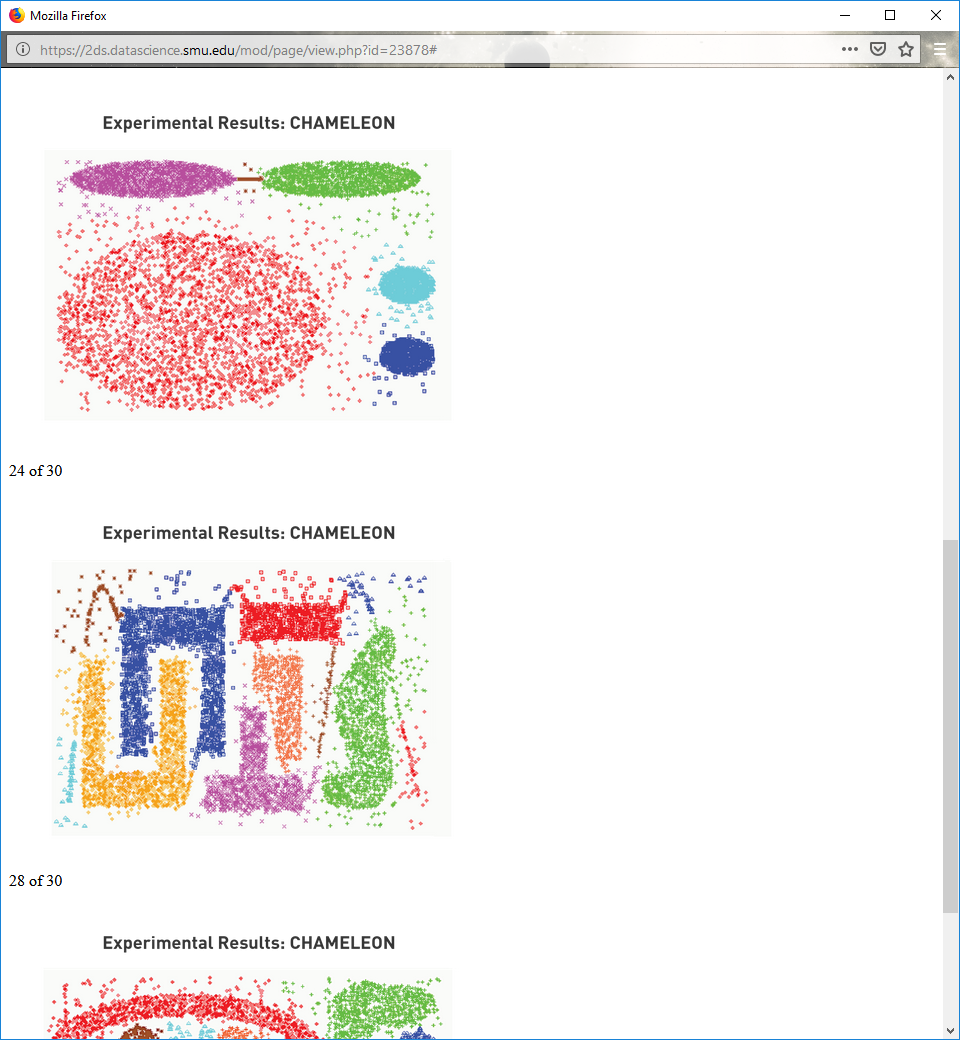
**Chameleon: Step 4**

4. Get “fused” similarity.

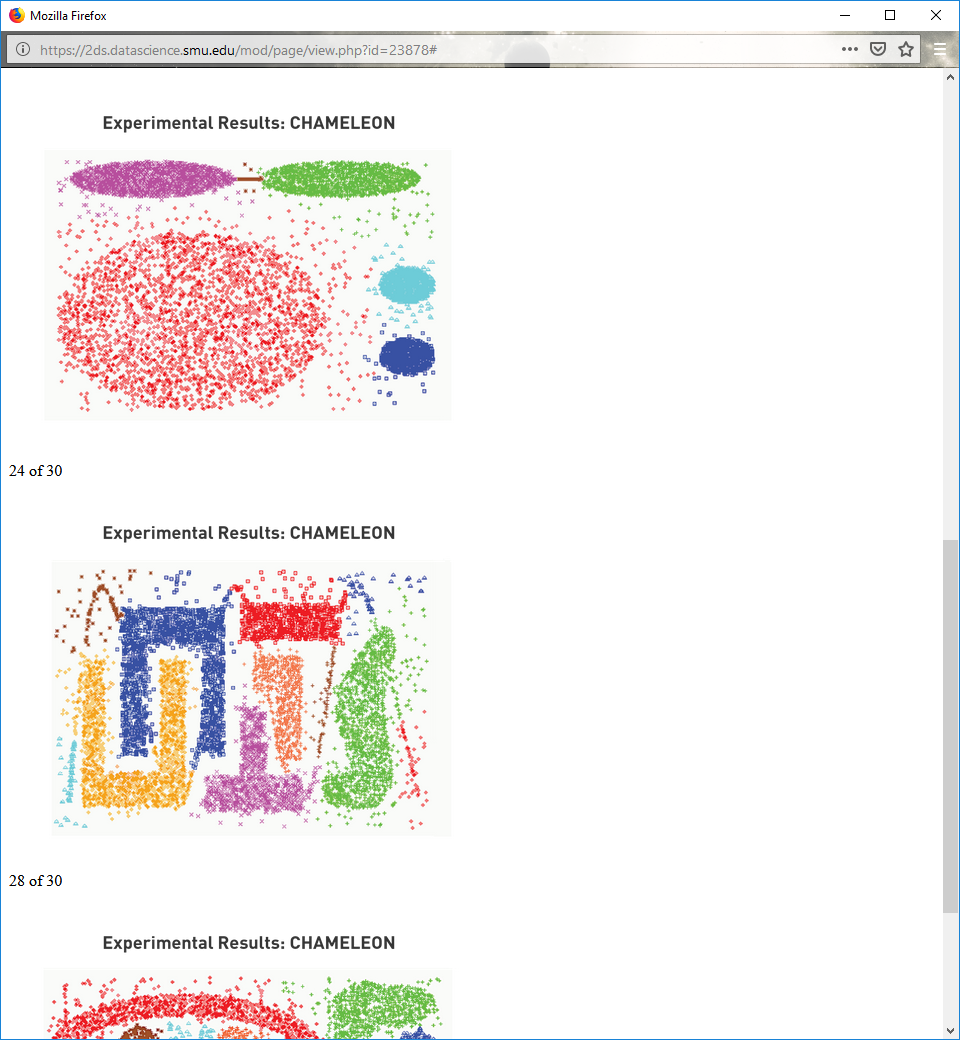


‘a’ is telling how much we value one vs the other

**Experimental Results: CHAMELEON**



**Experimental Results: CHAMELEON**



**Experimental Results: CHAMELEON**



**11.5**

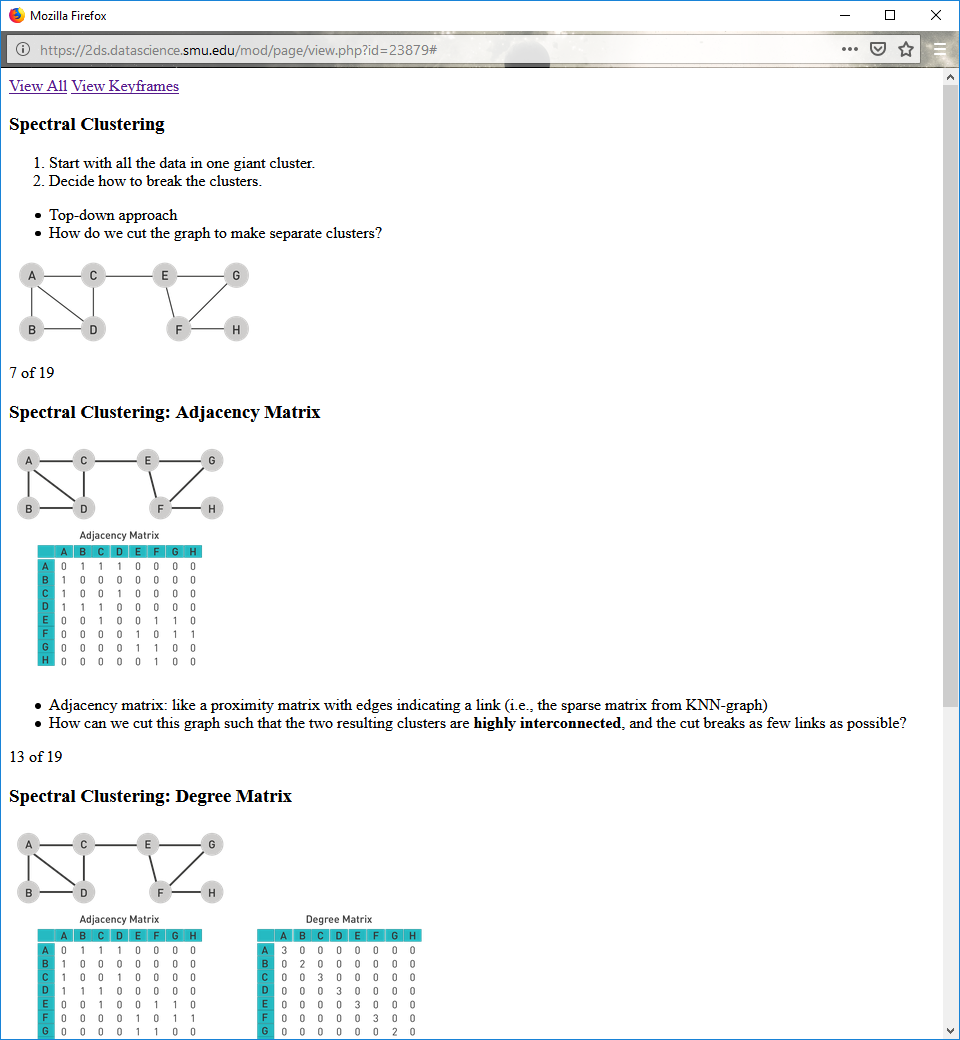
**Spectral Clustering**

1. Start with all the data in one giant cluster.

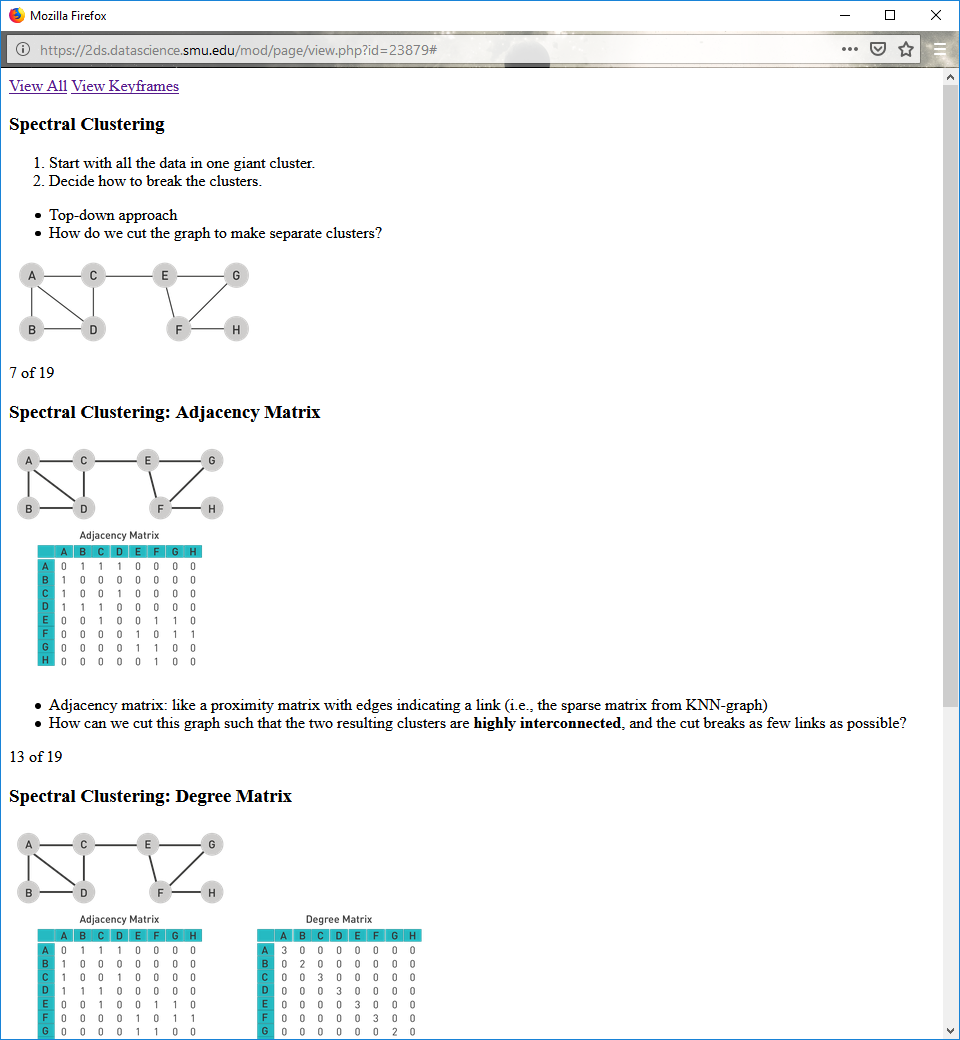
2. Decide how to break the clusters.

Top-down approach

How do we cut the graph to make separate clusters?



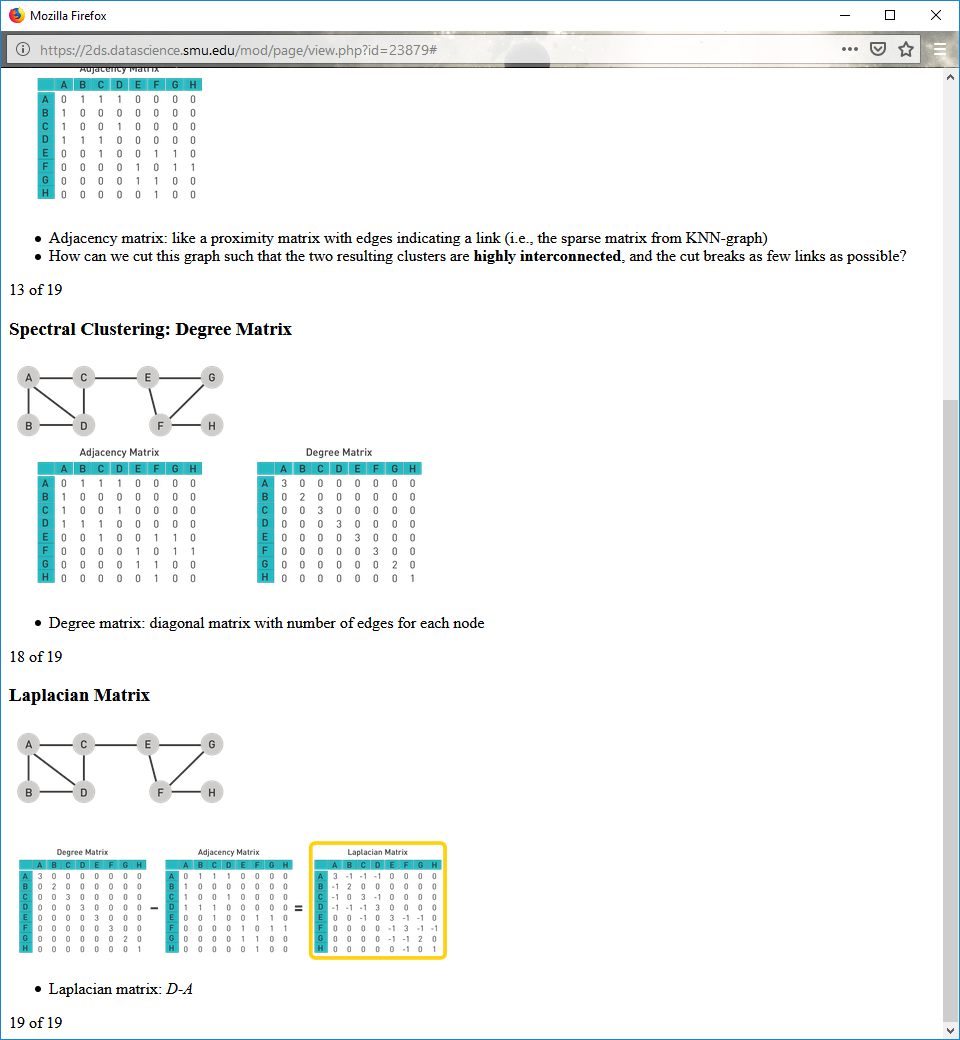
**Spectral Clustering: Adjacency Matrix**



**Adjacency matrix:** like a proximity matrix with edges indicating a link (i.e., the sparse matrix from KNN-graph)

How can we cut this graph such that the two resulting clusters are **highly interconnected**, and the cut breaks as few links as possible?

**Spectral Clustering: Degree Matrix**

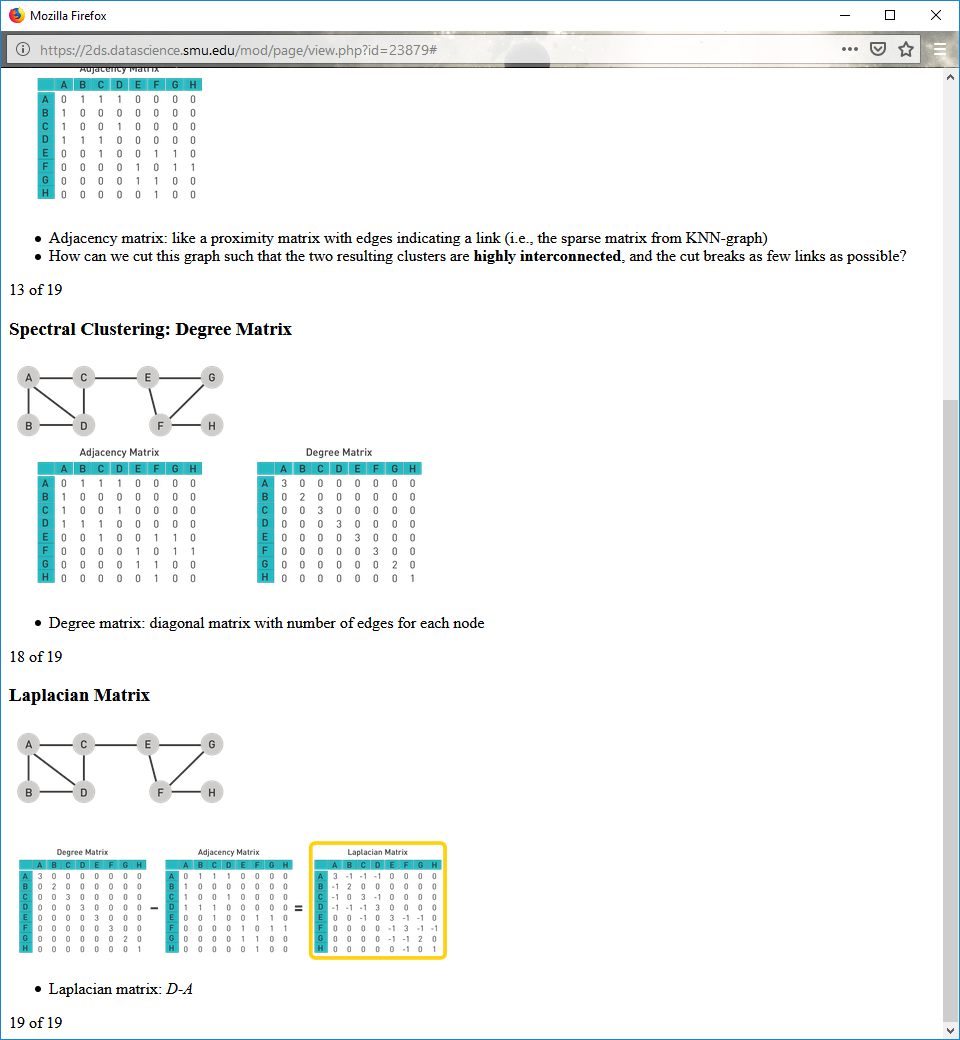


Degree matrix: diagonal matrix with number of edges for each node

Adjacency matrix is where the water is flowing to

Degree matrix is telling you the source of the water going out

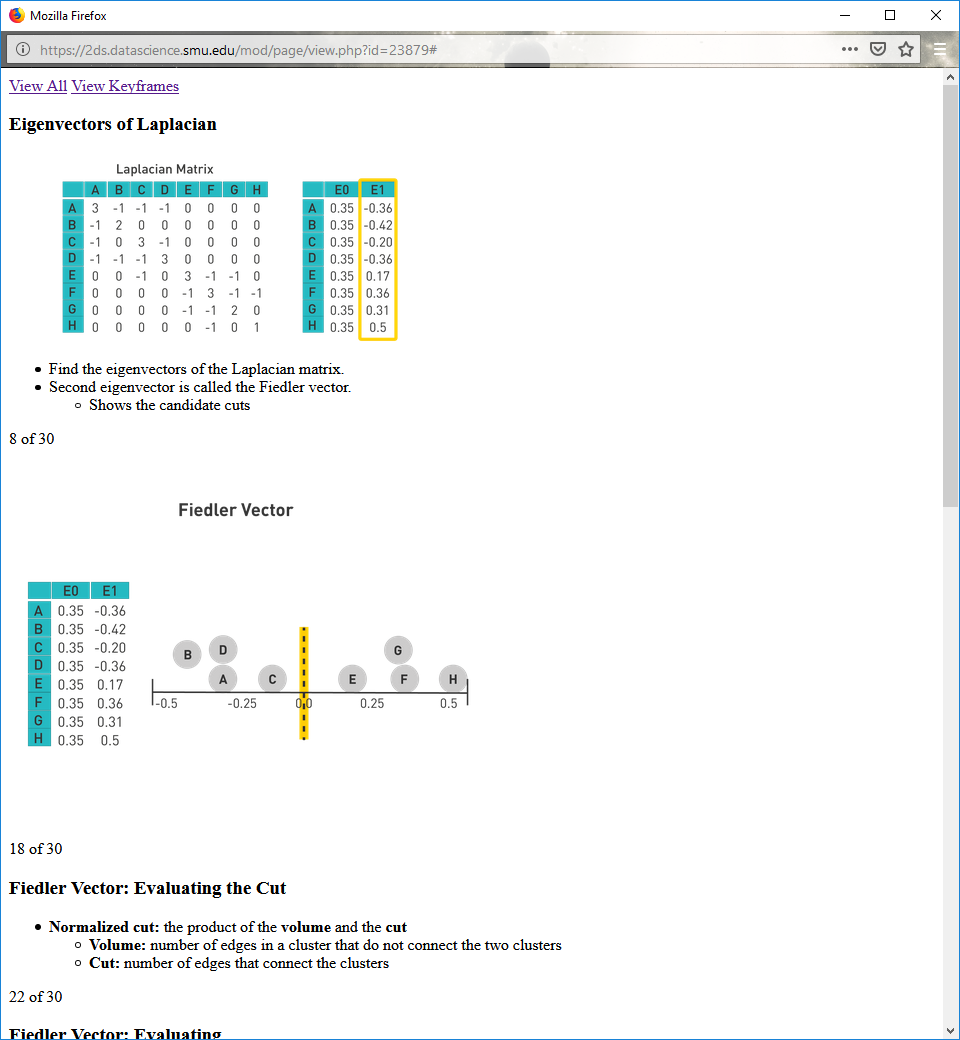
**Laplacian Matrix**



Laplacian matrix: D-A

Laplacian has some eigen vector properties

**Eigenvectors of Laplacian**

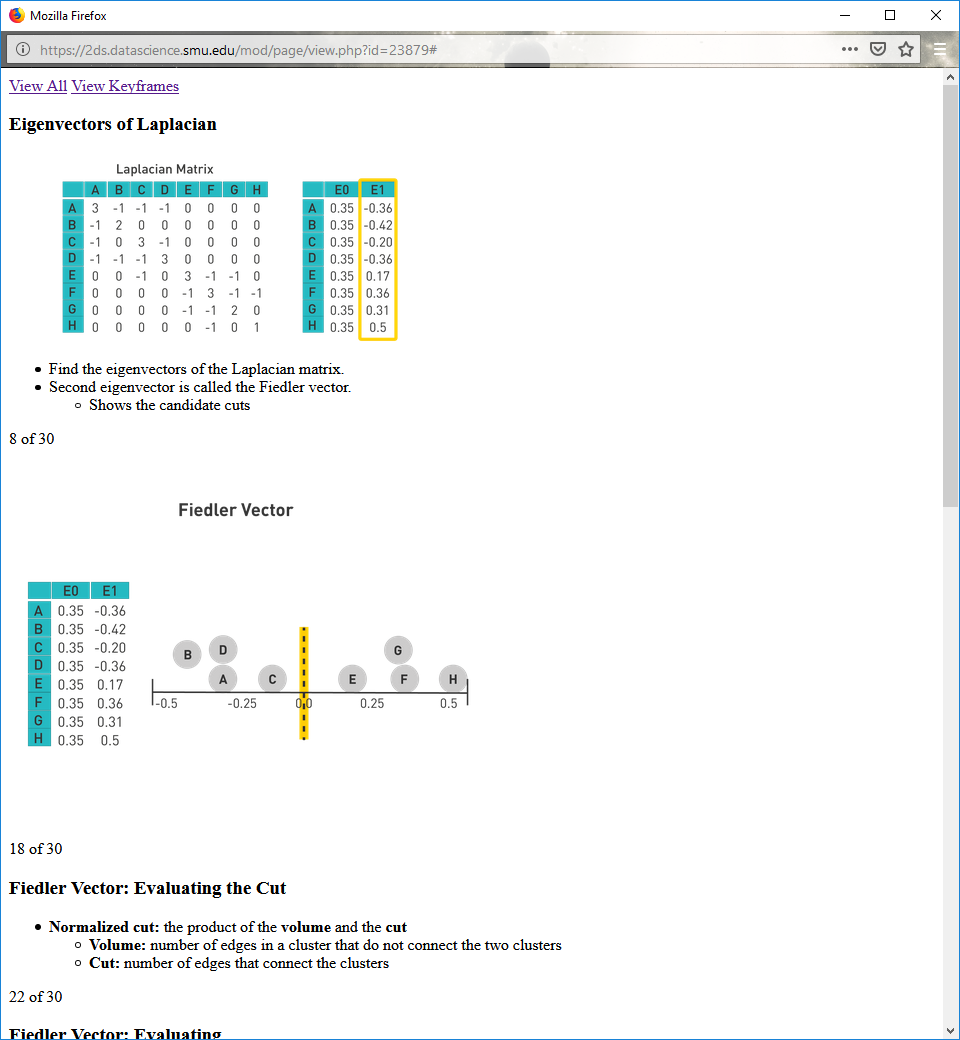


Find the eigenvectors of the Laplacian matrix. (sometimes called spectral vectors)

Second eigenvector is called the Fiedler vector.

Shows the candidate cuts

**Fiedler Vector**



You do not always cut on zero

**Fiedler Vector: Evaluating the Cut**

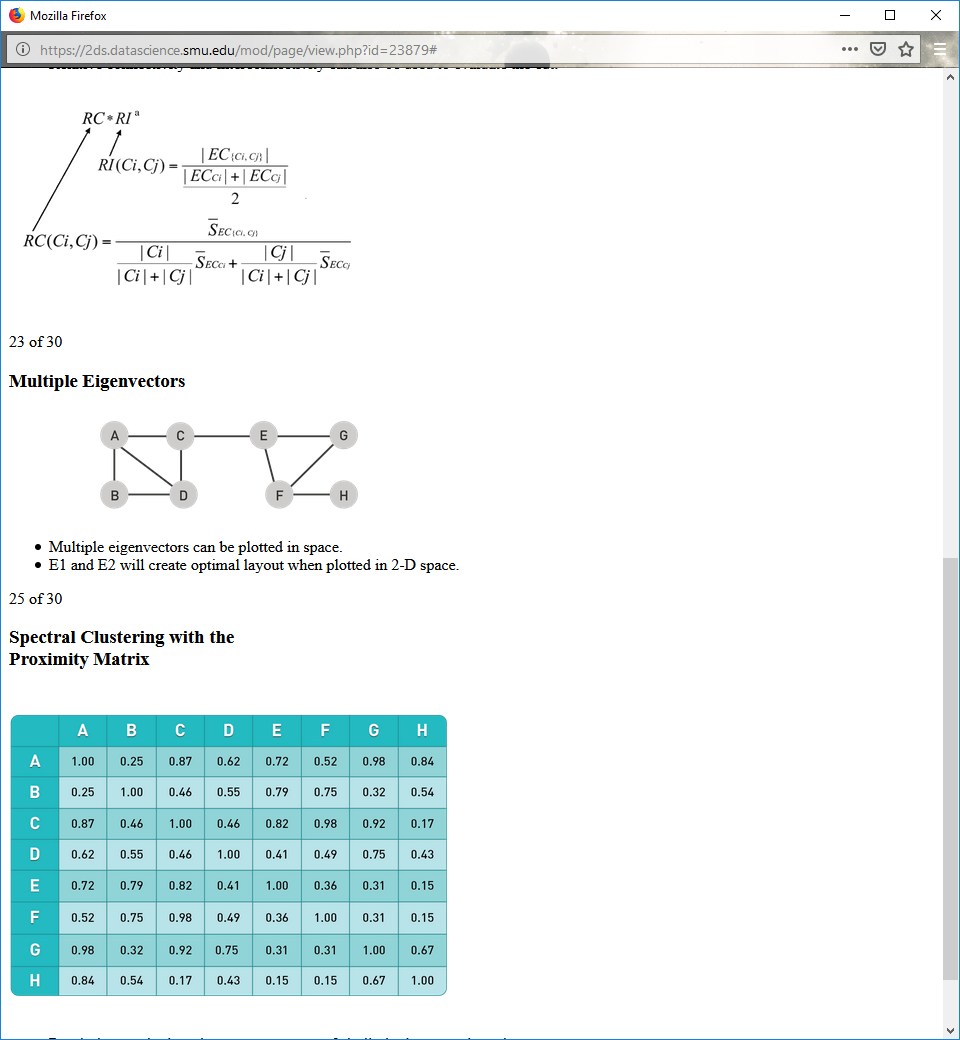
**Normalized cut**: the product of the **volume** and the **cut**

**Volume**: number of edges in a cluster that do not connect the two clusters

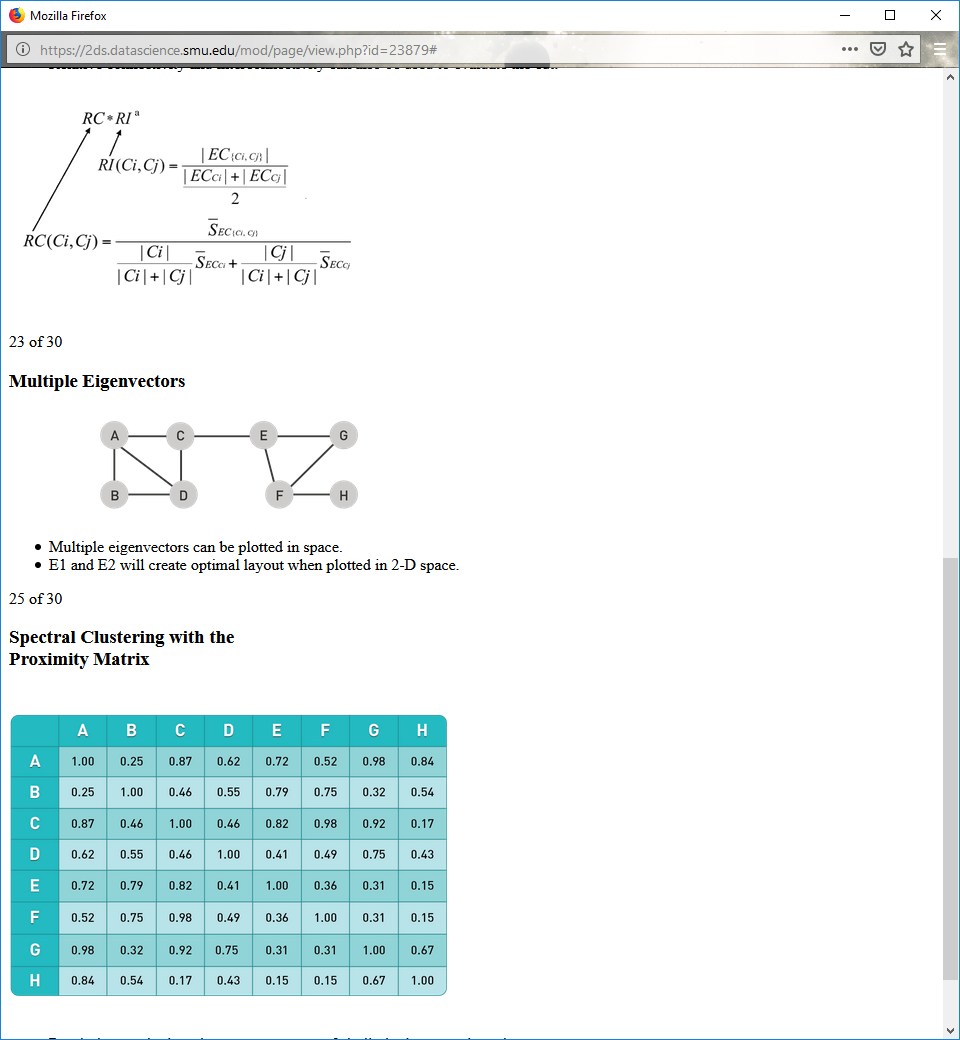
**Cut**: number of edges that connect the clusters

**Fiedler Vector: Evaluating the Cut (cont.)**

Relative connectivity and interconnectivity can also be used to evaluate the cut.

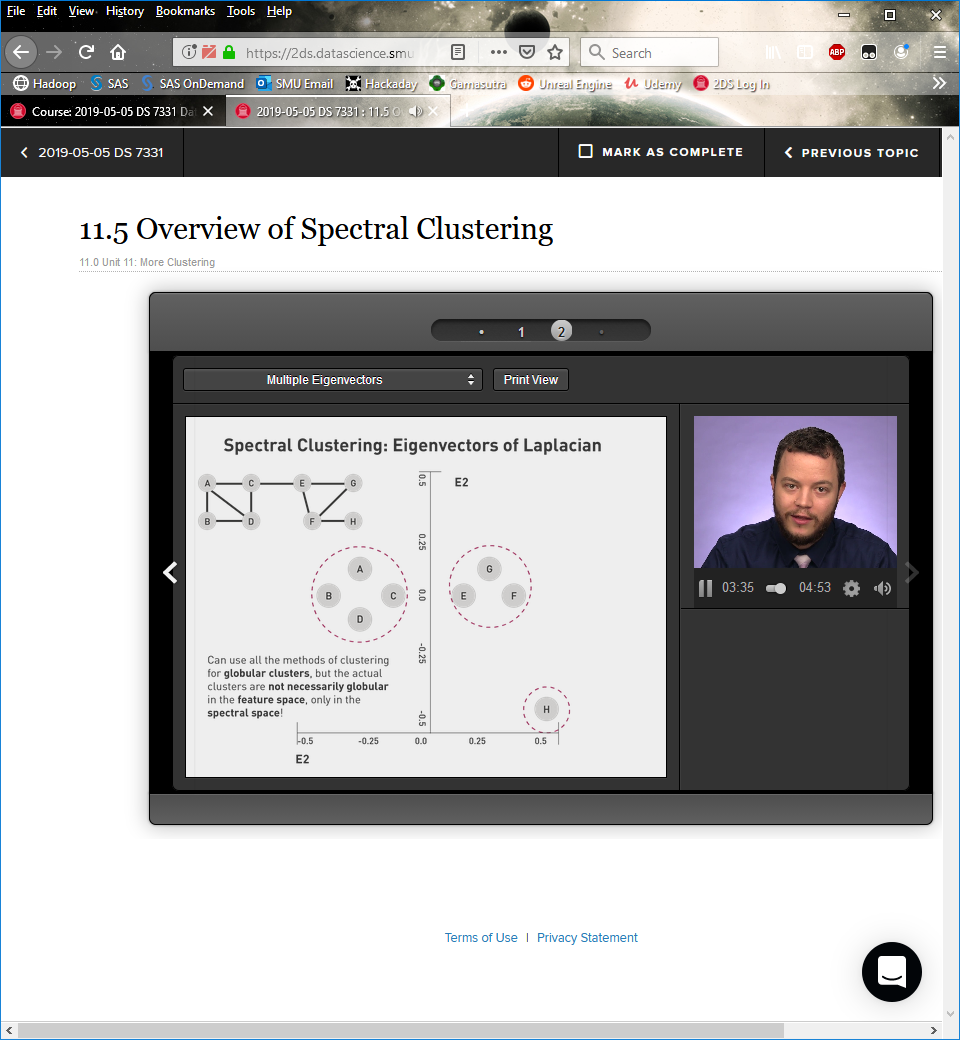


**Multiple Eigenvectors**

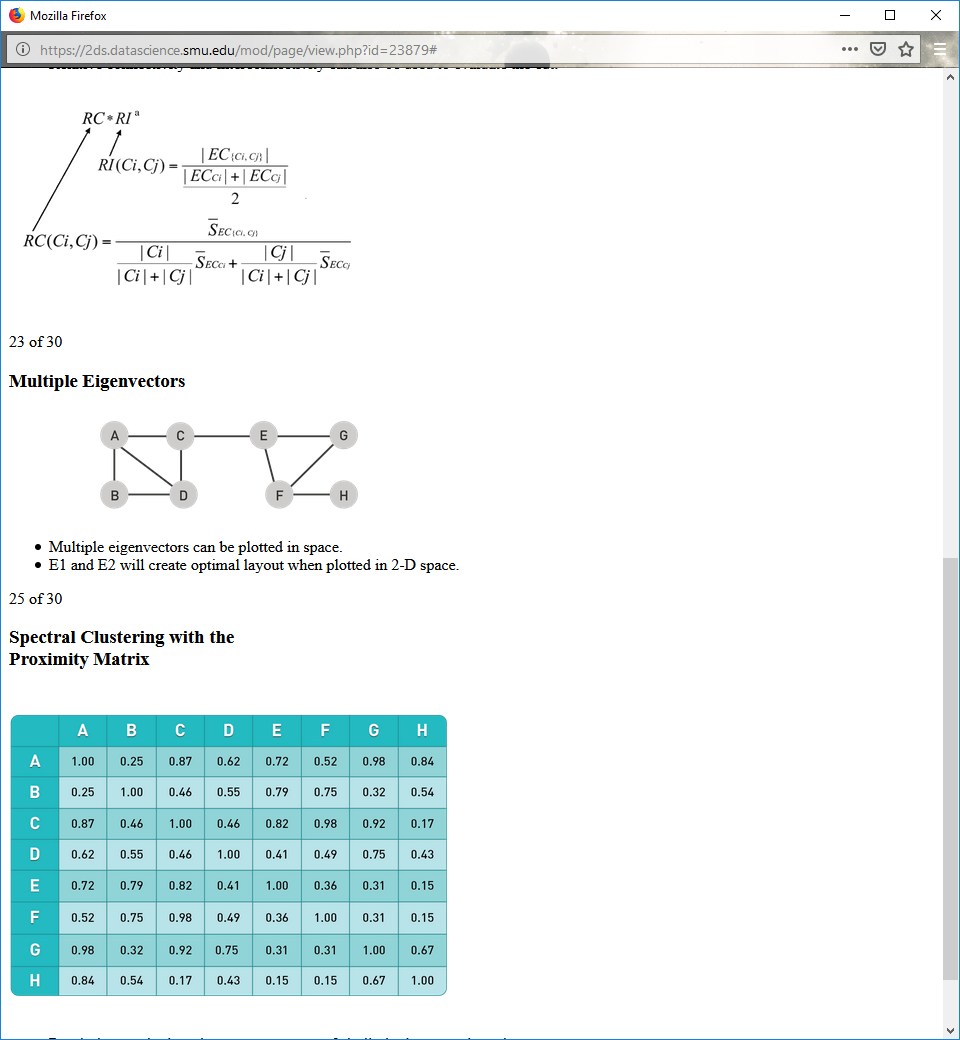


Multiple eigenvectors can be plotted in space.

E1 and E2 will create optimal layout when plotted in 2-D space.



**Spectral Clustering with the Proximity Matrix**



Proximity matrix: based on some measure of similarity between the points

Proximity matrix − degree matrix = Laplacian

Calculate eigenvectors in order to evaluate cuts.