Functional genomics = field that attempts to use the vast data produced by genomic projects (e.g. genome sequencing projects) to describe gene (and protein) functions and interactions.

Focuses on dynamic aspects, e.g. transcription, translation, and protein–protein interactions, as opposed to static aspects of the genome such as DNA sequence or structures.
Functional genomics + Data mining

= field that attempts to computationally discover patterns in large data sets

Adapted from Wikipedia
We’re going to first learn about clustering algorithms & classifiers.

Clustering = task of grouping a set of objects in such a way that objects in the same group (a cluster) are more similar (in some sense) to each other than to those in other groups (clusters).

Adapted from Wikipedia
We’re going to first learn about clustering algorithms & classifiers

Classification = task of categorizing a new observation, on the basis of a training set of data with observations (or instances) whose categories are known

Let’s motivate this with an important historical example:

Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling

Ash A. Alizadeh,1,2 Michael B. Eisen,3,4, R. Eric Davis,5 Chi Ma,6 Izidore S. Lossos,6 Andreas Rosenwald,7 Jennifer C. Boldrick,8 Najeeb Sabat,9 Truc Tran,10 Xin Tu,11 John L. Powell,12 Liming Yang,13 Gerald E. Mandel,14 Troy Moore,15 James Hudson Jr,16 Lisheng La17, David B. Lewis18, Robert Tibshirani19, Gavin Sherlock,20 Wing C. Chan21, Timothy C. Greene22, Dennis S. Weisenburger,23 James O. Armitage24, Roger Warneke25, Ronald Levy25, Wyndham Wilson25, Michael R. Grever24, John C. Byrd25, David Botstein4, Patrick O. Brown16,18 & Louis M. Staudt6

Nature 2000
“Diffuse large B-cell lymphoma (DLBCL), the most common subtype of non-Hodgkin's lymphoma ... is one disease in which attempts to define subgroups on the basis of morphology have largely failed...”

“DLBCL ... is clinically heterogeneous: 40% of patients respond well to current therapy and have prolonged survival, whereas the remainder succumb to the disease.

We proposed that this variability in natural history reflects unrecognized molecular heterogeneity in the tumours.”

---

**Blast from the past: Profiling mRNA expression with DNA microarrays**

DNA molecules are attached to a solid substrate, then... ...probed with a labeled (usually fluorescent) DNA sequence
Blast from the past: Profiling mRNA expression with DNA microarrays

(FYI, we would generally now just sequence the cDNA)

Note that some arrays are 1-color, some are 2. Why?
DNA microarrays are a great example of the “arc” of a technology over time

DNA microarrays

RNA sequencing

Worldwide Google trends, 2004-present

Back to diffuse large B-cell lymphoma...

96 patient biopsies
(normal and malignant lymphocyte samples)

Extract mRNA from each sample

Perform DNA microarray experiment on each to measure mRNA abundances (~1.8 million total gene expression measurements)

Cluster samples by their expression patterns

Nature 2000
Hierarchical clustering of the gene expression data

Genes can be found whose expression is specific to germinal centre B cells, and different across DLBCL’s
We can break up the DLBCL’s according the germinal B-cell specific gene expression:

What good is this? These molecular phenotypes predict clinical survival.

Kaplan-Meier plot of patient survival
What good is this? These molecular phenotypes predict clinical survival.

Gene expression, and other molecular measurements, provide far deeper phenotypes for cells, tissues, and organisms than traditional measurements.

These sorts of observations have now motivated tons of work using these approaches to diagnose specific forms of disease, as well as to discover functions of genes and many other applications.
So, how does clustering work?

First, let’s think about the data, e.g. as for gene expression. From one sample, using DNA microarrays or RNA-seq, we get:

Expression level of gene 1
Expression level of gene 2
Expression level of gene 3
.
.
.
Expression level of gene $i$
.
.
Expression level of gene $N$

i.e., a vector of $N$ numbers

For yeast, $N \sim 6,000$
For human, $N \sim 22,000$

So, how does clustering work?

Every additional sample adds another column, giving us a matrix of data:

$M$ samples

Gene 1, sample 1 ... ... Gene 1, sample $j$ ... ... Gene 1, sample $M$
Gene 2, sample 1 ... ... Gene 2, sample $j$ ... ... Gene 2, sample $M$
Gene 3, sample 1 ... ... Gene 3, sample $j$ ... ... Gene 3, sample $M$
.
.
.
Gene $i$, sample 1 ... ... Gene $i$, sample $j$ ... ... Gene $i$, sample $M$
.
.
.
Gene $N$, sample 1 ... ... Gene $N$, sample $j$ ... ... Gene $N$, sample $M$

i.e., a matrix of $N \times M$ numbers

For yeast, $N \sim 6,000$
For human, $N \sim 22,000$
So, how does clustering work?

Every gene has a **feature vector** of \( M \) numbers associated with it.

Similarly, every sample has a **feature vector** of \( N \) numbers associated with it.
So, how does clustering work?

The first clustering method we’ll learn about simply groups the objects (samples or genes) in a hierarchy by the similarity of their feature vectors.

A hierarchical clustering algorithm

Start with each object in its own cluster

Until there is only one cluster left, repeat:
- Among the current clusters, find the two most similar clusters
- Merge those two clusters into one

We can choose our measure of similarity and how we merge the clusters
**Hierarchical clustering**

*Conceptually*

Data points on an X-Y plane

---

**We’ll need to measure the similarity between feature vectors. Here are a few (of many) common distance measures used in clustering.**

<table>
<thead>
<tr>
<th>Names</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean distance</td>
<td>$|a - b|_2 = \sqrt{\sum_i (a_i - b_i)^2}$</td>
</tr>
<tr>
<td>Manhattan distance</td>
<td>$|a - b|_1 = \sum_i</td>
</tr>
<tr>
<td>cosine similarity</td>
<td>$\frac{a \cdot b}{|a| |b|}$</td>
</tr>
</tbody>
</table>
Back to the B cell lymphoma example

Hierarchical clustering

Similarity measure = **Pearson correlation coefficient** between gene expression vectors

Similarity between clusters = average similarity between individual elements of each cluster (also called average linkage clustering)

K-means clustering is a common alternative clustering approach

*mainly because it’s easy and can be quite fast!*

The basic algorithm:
1. Pick a number \((k)\) of cluster centers
2. Assign each gene to its nearest cluster center
3. Move each cluster center to the mean of its assigned genes
4. Repeat steps 2 & 3 until convergence

*See the K-means example posted on the web site*
A 2-dimensional example

A 2-dimensional example: hierarchical
A 2-dimensional example: $k$-means

Decision boundaries
Some features of K-means clustering

• Depending on how you seed the clusters, it may be stochastic. You may not get the same answer every time you run it.
• Every data point ends up in exactly 1 cluster (so-called hard clustering)
• Not necessarily obvious how to choose $k$
• Great example of something we’ve seen already: Expectation-Maximization (E-M) algorithms

EM algorithms alternate between assigning data to models (here, assigning points to clusters) and updating the models (calculating new centroids)

Let’s think about this aspect for a minute. Why is this good or bad? How could we change it?
**k-means**

The basic algorithm:
1. Pick a number \((k)\) of cluster centers
2. Assign each gene to its nearest cluster center
3. Move each cluster center to the mean of its assigned genes
4. Repeat steps 2 & 3 until convergence

**Fuzzy k-means**

The basic algorithm:
1. Choose \(k\). Randomly assign cluster centers.
2. Fractionally assign each gene to each cluster: 
   \[ \text{occupancy} \ (g_i, m_j) = e^{-||g_i - m_j||^2} / \sum_j e^{-||g_i - m_j||^2} \]
   Note: \(||x||\) is just shorthand for the length of the vector \(x\).
   \(g_i = \text{gene } i\)
   \(m_j = \text{centroid of cluster } j\)
3. For each cluster, calculate weighted mean of genes to update cluster centroid
4. Repeat steps 2 & 3 until convergence
Remove genes correlated >0.7 to the identified centroids.

Iterating fuzzy k-means.
A fun clustering strategy that builds on these ideas: Self-organizing maps (SOMs)

- Combination of clustering & visualization
- Invented by Teuvo Kohonen, also called Kohonen maps

Iterating fuzzy \( k \)-means
A fun clustering strategy that builds on these ideas: Self-organizing maps (SOMs)

SOMs have:
- your data (points in some high-dimensional space)
- a grid of nodes, each node also linked to a point someplace in data space

1. First, SOM nodes are arbitrarily positioned in data space. Then:
2. Choose a training data point. Find the node closest to that point.
3. Move its position closer to the training data point.
4. Move its grid neighbors closer too, to a lesser extent.
Repeat 2-4. After many iterations, the grid approximates the data distribution.

--

Here's an example using colors. Each color has an RGB vector. Take a bunch of random colors and organize them into a map of similar colors:

Each SOM node lives in RGB space →

Here's the SOM →

Here's the input color data →
Iteratively test new colors, update the map using some rule

\[ m_i(t + 1) = m_i(t) + \alpha(t)[x_i(t) - m_i(t)] \quad \text{for each} \quad i \in N_i(t), \]

Updated node vector

Starting node vector

Difference from data vector

The weight and node neighborhoods shrink with time (iterations)

Over time, the map self-organizes to show clusters of like colors.

A SOM of U.S. Congress voting patterns

Red = yes votes
Blue = no votes
Exploratory Analysis of CIA Factbook Data Using Kohonen Self-Organizing Maps

Botswana

USA
SOM of Wikipedia (from Wikipedia, naturally)
(data = wiki article word frequency vectors)

SOMs can accommodate unusual data distributions

One-dimensional SOM

Data points
Finally, **t-SNE** is a nice way to visualize data in 2 or 3D = *t-distributed stochastic neighbor embedding*

**t-SNE** tries to reproduce high-D *data neighborhoods* in a 2D or 3D picture by:

1. Defining a probability distribution over pairs of high-D objects such that “similar” objects have a high probability of being picked, whilst “dissimilar“ objects have an extremely small probability of being picked

2. Defining a similar probability distribution over the points in the low-D map

3. Minimizing the Kullback–Leibler divergence between the two distributions by varying the locations of the points in the low-D map, i.e.

$$\sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

- probability $i$ and $j$ are close in high-D space
- probability $i$ and $j$ are close in low-D space

Sum over all pairs of points

---

**Separating cells into cell types by t-SNE**
- healthy human bone marrow, stained with 13 markers and measured by mass cytometry, visualized with viSNE

---

You can compute your own t-SNE embeddings using the online tools at:
http://projector.tensorflow.org/

There are also some great examples at:
http://distill.pub/2016/misread-tsne/

There are only a couple of parameters you can tweak, mainly perplexity, which effectively captures the number of neighbors (often 5 to 50)