Modularity and Code Length in Community Detection

Social Networks (SOC6110, Spring 2023)

Barum Park (b.park@cornell.edu)

Modularity

Modularity: Two Definitions

You'll find two equivalent definitions of the modularity in the literature:

1. One that looks like this (Newman and Grivan 2004):

$$Q = \sum_{k=1}^{K} (e_{kk} - e_k^2)$$

where

- K is the number of *modules*,
- e_{kk} is the *proportion* of edges connecting nodes within module k,
- e_k is the *marginal* proportion of edges that have at least one end node in module k.

Modularity: Two Definitions

You'll find two equivalent definitions of the modularity in the literature:

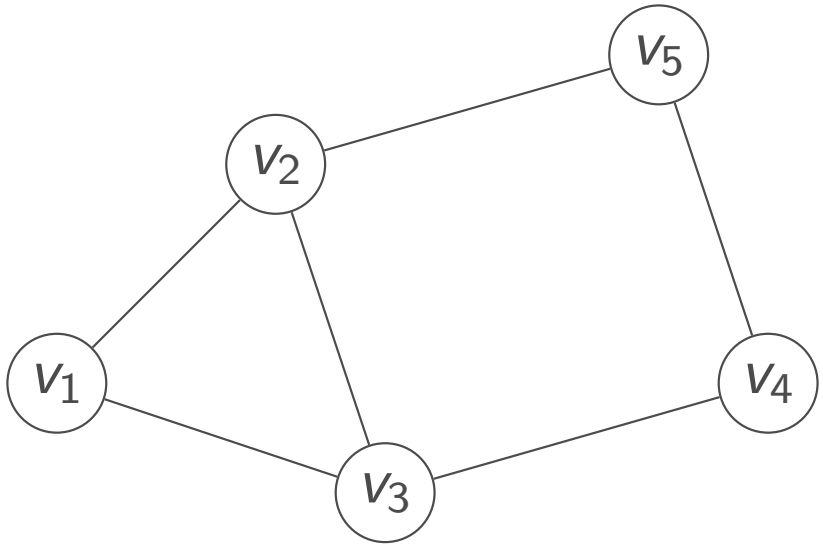
1. The other definition looks like this (Blondel et al. 2008):

$$Q = \frac{1}{2m} \sum_{i=1}^{n} \sum_{j=1}^{n} \left(a_{ij} - \frac{d_i d_j}{2m} \right) \delta(c_i, c_j)$$

where

- m is the number of edges,
- a_{ij} is the (i, j)th element of the adjacency matrix,
- d_i is the degree of node i,
- c_i is an indicator of the module to which node i belongs
- δ is the Kronecker delta function.

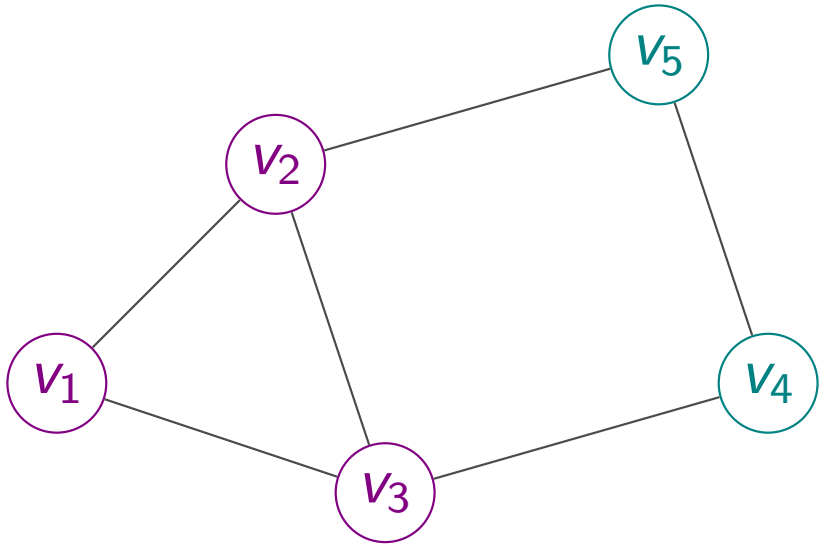
Consider the following graph



with adjacency matrix

$$\mathbf{A} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Suppose we cluster this graph in to the following partition:



with adjacency matrix

$$\mathbf{A} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

From

$$\mathbf{A} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

We can create a new matrix that contains the *within- and between-module* sum of ties

$$\mathbf{B} = \begin{array}{c} \\ C_1 \\ C_2 \end{array} \begin{array}{c} \begin{array}{cc} C_1 & C_2 \end{array} \\ \begin{bmatrix} 6 & 2 \\ 2 & 2 \end{bmatrix} \end{array}$$

Large numbers in the diagonals of this matrix mean that most ties connect nodes of the same module (strong clustering)

$$\mathbf{B} = \begin{array}{c} \\ C_1 \\ C_2 \end{array} \begin{array}{c} \begin{array}{cc} C_1 & C_2 \end{array} \\ \begin{bmatrix} 6 & 2 \\ 2 & 2 \end{bmatrix} \end{array}$$

Problem I: the numbers in the diagonal depend on the density of the network.

So, we divide each element of $\mathbf{B}$ by $2m$, where $m = 6$ is the number of edges in the network (why $2m$ and not m?).

$$\mathbf{B} = \begin{array}{c} \\ C_1 \\ C_2 \end{array} \begin{array}{c} \begin{array}{cc} C_1 & C_2 \end{array} \\ \begin{bmatrix} 6 & 2 \\ 2 & 2 \end{bmatrix} \end{array}$$

Problem I: the numbers in the diagonal depend on the density of the network.

So, we divide each element of $\mathbf{B}$ by $2m$, where $m = 6$ is the number of edges in the network (why $2m$ and not m?).

$$\mathbf{E} = \begin{array}{c} \\ C_1 \\ C_2 \end{array} \begin{array}{c} \begin{array}{cc} C_1 & C_2 \end{array} \\ \begin{bmatrix} \frac{6}{12} & \frac{2}{12} \\ \frac{2}{12} & \frac{2}{12} \end{bmatrix} \end{array}$$

The closer the sum over the diagonals to 1, the stronger would be the clustering.

$$\mathbf{E} = \begin{array}{c} \\ C_1 \\ C_2 \end{array} \begin{array}{c} \begin{array}{cc} C_1 & C_2 \end{array} \\ \begin{bmatrix} \frac{6}{12} & \frac{2}{12} \\ \frac{2}{12} & \frac{2}{12} \end{bmatrix} \end{array}$$

Problem II: we could always throw *all* nodes into one module and get a diagonal sum of 1.

The matrix **E** would have only one cell which *is* the diagonal (and equal to 1).

$$\mathbf{E} = \begin{array}{c} \\ C_1 \\ C_2 \end{array} \begin{array}{c} \begin{array}{cc} C_1 & C_2 \end{array} \\ \begin{bmatrix} \frac{6}{12} & \frac{2}{12} \\ \frac{2}{12} & \frac{2}{12} \end{bmatrix} \end{array}$$

Problem II: we could always throw *all* nodes into one module and get a diagonal sum of 1.

The matrix **E** would have only one cell which *is* the diagonal (and equal to 1).

So, we need a *null model* to which we can to compare the clustering strength (i.e., switch to relative strength).

$$\mathbf{E} = \begin{array}{c} \\ C_1 \\ C_2 \end{array} \begin{array}{c} \begin{array}{cc} C_1 & C_2 \end{array} \\ \begin{bmatrix} \frac{6}{12} & \frac{2}{12} \\ \frac{2}{12} & \frac{2}{12} \end{bmatrix} \end{array} \begin{array}{c} \\ \frac{2}{3} \\ \frac{1}{3} \end{array}$$

Notice that the sum of the kth row of $\mathbf{E}$ gives us the proportion of edges that have at least one end-point in cluster k.

Denote this proportion as e_k.

If we would place the edges in the network at random, under the constraint that the resulting module sizes remain fixed, the *expected proportion* of ties in the (k, l)th cell of $\mathbf{E}$ would be $e_k \times e_l$.

So the *observed* and *expected* between-module tie-matrix looks like:

$$\mathbf{E} = \begin{array}{c} \\ C_1 \\ C_2 \end{array} \begin{array}{c} \begin{array}{cc} C_1 & C_2 \end{array} \\ \begin{bmatrix} \frac{6}{12} & \frac{2}{12} \\ \frac{2}{12} & \frac{2}{12} \end{bmatrix} \end{array} \begin{array}{c} \\ \frac{2}{3} \\ \frac{1}{3} \end{array}$$

$$\mathbb{E}[\mathbf{E}] = \begin{array}{c} \\ C_1 \\ C_2 \end{array} \begin{array}{c} \begin{array}{cc} C_1 & C_2 \end{array} \\ \begin{bmatrix} \frac{4}{9} & \frac{2}{9} \\ \frac{2}{9} & \frac{1}{9} \end{bmatrix} \end{array} \begin{array}{c} \\ \frac{2}{3} \\ \frac{1}{3} \end{array}$$

So the *observed* and *expected* between-module tie-matrix looks like:

$$\mathbf{E} = \begin{array}{c} \\ C_1 \\ C_2 \end{array} \begin{array}{c} \begin{array}{cc} C_1 & C_2 \end{array} \\ \begin{bmatrix} \frac{6}{12} & \frac{2}{12} \\ \frac{2}{12} & \frac{2}{12} \end{bmatrix} \end{array} \begin{array}{c} \\ \frac{2}{3} \\ \frac{1}{3} \end{array} \qquad \mathbb{E}[\mathbf{E}] = \begin{array}{c} \\ C_1 \\ C_2 \end{array} \begin{array}{c} \begin{array}{cc} C_1 & C_2 \end{array} \\ \begin{bmatrix} \frac{4}{9} & \frac{2}{9} \\ \frac{2}{9} & \frac{1}{9} \end{bmatrix} \end{array} \begin{array}{c} \\ \frac{2}{3} \\ \frac{1}{3} \end{array}$$

The modularity is simply the sum of the diagonal elements of $\mathbf{E} - \mathbb{E}[\mathbf{E}]$, i.e.,

$$Q = \text{Trace}(\mathbf{E} - \mathbb{E}[\mathbf{E}]) = \sum_{k=1}^{K} (e_{kk} - e_k^2),$$

where e_{kk} is the kth diagonal element and e_k is the kth row-sum of the matrix $\mathbf{E}$.

Modularity Maximization

Newman and Grivan (2004) use a divisive algorithm to find a partition that maximizes Q

Blondel et al. (2008) use a multi-level agglomerative algorithm to find partitions that maximize Q

Traag et al. (2019) show that the algorithm by Blondel is flawed and suggest an efficient alternative

Implications

1. The modularity shows how "good" a clustering is, where "good" means better than a scenario where
   - ▷ ties are placed at random between the nodes of the network
   - ▷ given the constraint that the module sizes remain fixed

2. As the modularity is a sum of the difference of proportions, $-1 \leq Q \leq 1$.

Considerations

1. Suppose we have a network with two clusters of equal size, where all ties are within each module and no tie connects nodes from different modules. What would be the modularity?

Considerations

1. Suppose we have a network with two clusters of equal size, where all ties are within each module and no tie connects nodes from different modules. What would be the modularity?

   It would be $Q = 0.5$.

   - ▷ Half of the edges will be within-modules in the null model (i.e., by chance)
   - ▷ The upper bound of $Q = 1$ is reached only in networks with infinite clusters.

Considerations

1. Suppose we have a network with two clusters of equal size, where all ties are within each module and no tie connects nodes from different modules. What would be the modularity?

2. It is designed for *undirected graphs* (see Kim et al., 2010)

Considerations

1. Suppose we have a network with two clusters of equal size, where all ties are within each module and no tie connects nodes from different modules. What would be the modularity?

2. It is designed for *undirected graphs* (see Kim et al., 2010)

   There were suggestions to generalize it for directed graphs, some of them are more successful than others.

Considerations

1. Suppose we have a network with two clusters of equal size, where all ties are within each module and no tie connects nodes from different modules. What would be the modularity?

2. It is designed for *undirected graphs* (see Kim et al., 2010)

3. Resolution limit

Equivalence to Second Formula

The second expression of the modularity is given as

$$Q = \frac{1}{2m} \sum_{i=1}^{n} \sum_{j=1}^{n} \left(a_{ij} - \frac{d_i d_j}{2m} \right) \delta(c_i, c_j)$$

It's quite simple to show that this is the same as the expression from the previous slides.

Recall that we had

$$Q = \sum_{k=1}^{K} (e_{kk} - e_k^2) = \sum_{k} e_{kk} - \sum_{k} e_k^2.$$

We will focus of each of the term separately.

First term: $\sum_k e_{kk}$

Let $I_{ik} = 1$ if $c_i = k$ and zero otherwise, where $c_i = k$ means that node i belongs to module k.

Recall that e_{kk} is simply the proportion of ties that are in module k. So, we can write

$$e_{kk} = \frac{1}{2m} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} I_{ik} I_{jk}$$

and summing over all k, we get

$$\sum_{k=1}^{K} e_{kk} = \frac{1}{2m} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} \delta(c_i, c_j).$$

Second term: $\sum_k e_k^2$

We note that

$$e_k^2 = \left(\frac{1}{2m} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} I_{ik} \right)^2 = \left(\frac{1}{2m} \sum_{i=1}^{n} d_i I_{ik} \right)^2$$
$$= \frac{1}{4m^2} \sum_{i=1}^{n} \sum_{j=1}^{n} d_i d_j I_{ik} I_{jk}$$

Summing this over all k gives, therefore,

$$\sum_k e_k^2 = \frac{1}{2m} \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\frac{d_i d_j}{2m} \right) \delta(c_i, c_j).$$

Putting the terms together

Putting these two expressions together, we obtain

$$\begin{aligned} Q &= \sum_{k=1}^{K} e_{kk} - \sum_{k} e_k^2 \\ &= \frac{1}{2m} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} \delta(c_i, c_j) - \frac{1}{2m} \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\frac{d_i d_j}{2m} \right) \delta(c_i, c_j) \\ &= \frac{1}{2m} \sum_{i=1}^{n} \sum_{j=1}^{n} \left(a_{ij} - \frac{d_i d_j}{2m} \right) \delta(c_i, c_j) \end{aligned}$$

as desired.

Code Length and the Map Equation

The Minimum Description Length (MDL) Principle

Informally stated, the MDL principle states

> *Any regularity in the data can be compressed with the help of a model. The shorter our description of the data (with the help of the model) the better our model.*

The Minimum Description Length (MDL) Principle

Informally stated, the MDL principle states

> *Any regularity in the data can be compressed with the help of a model. The shorter our description of the data (with the help of the model) the better our model.*

So, if

1. there are patterns in the data; and
2. we have a "good" model to describe the pattern

we'll be able to describe our data with a shorter code, once we use the model

And a model that leads to a shorter description length in total is the better model

The Minimum Description Length (MDL) Principle

To be a bit more formal, let D be the data and M our model, and let $L(x)$ be the total description length of x.

The Minimum Description Length (MDL) Principle

To be a bit more formal, let D be the data and M our model, and let $L(x)$ be the total description length of x.

When we encode the data *with* the help of our model, then

$$L_M(D) = \underbrace{L(D \mid M)}_{\text{DL of data given model}} + \underbrace{L(M)}_{\text{DL of model}} .$$

The Minimum Description Length (MDL) Principle

To be a bit more formal, let D be the data and M our model, and let $L(x)$ be the total description length of x.

When we encode the data *with* the help of our model, then

$$L_M(D) = \underbrace{L(D \mid M)}_{\text{DL of data given model}} + \underbrace{L(M)}_{\text{DL of model}} .$$

So, $L_M(D)$ will be shorter when our model compresses the data a lot ($L(D \mid M)$ is small) and is parsimonious ($L(M)$ is small).

Description Length?

But what exactly *is* the description length of the data or model?

Description Length?

But what exactly *is* the description length of the data or model?

Of course, the description length depends on the code we use.

Description Length?

But what exactly *is* the description length of the data or model?

Of course, the description length depends on the code we use. And we don't want to deal with specific codes in this course (!)

Description Length?

But what exactly *is* the description length of the data or model?

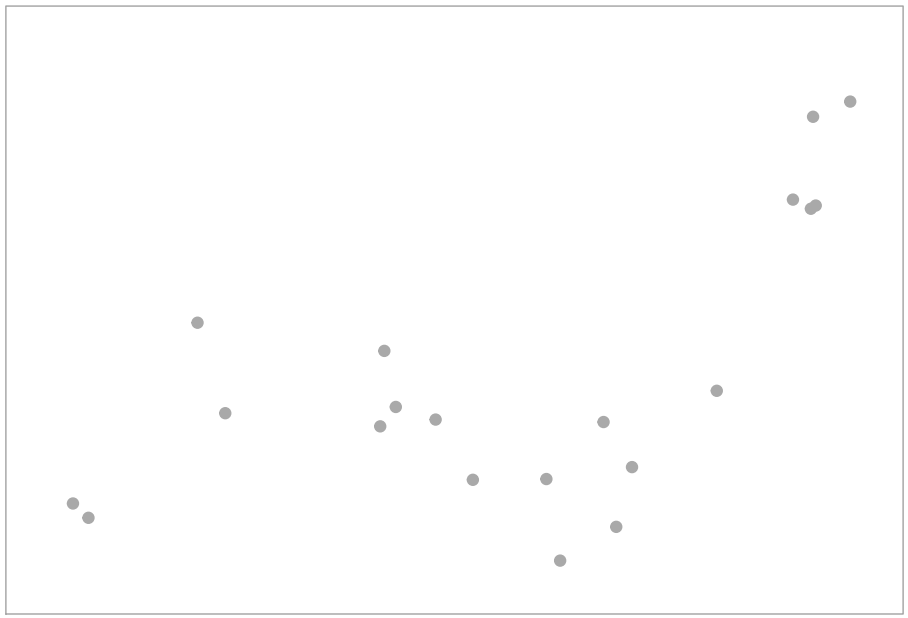
Of course, the description length depends on the code we use. And we don't want to deal with specific codes in this course (!)

Here, we just note that the description length of

1. a random source increases with its entropy (read uncertainty/variability)
2. a model increases with the (effective) number of parameters

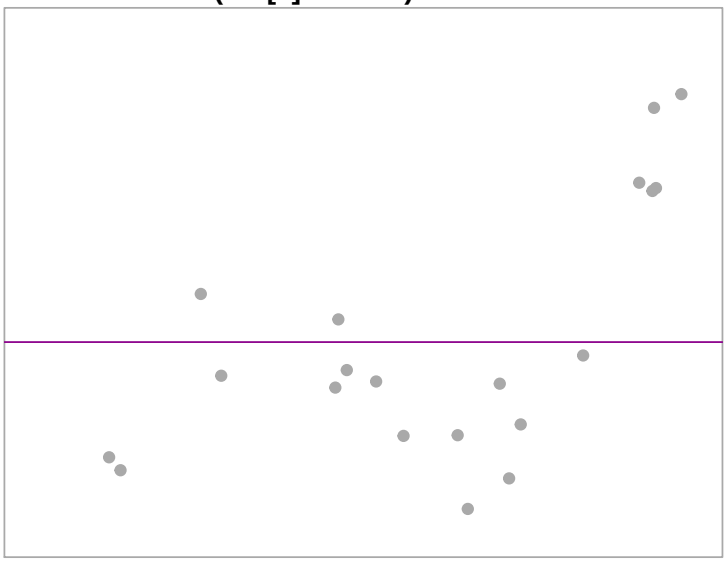
Example: Regression Problem

Suppose we have the following dataset and we want to describe how the y-values depend on x-values.

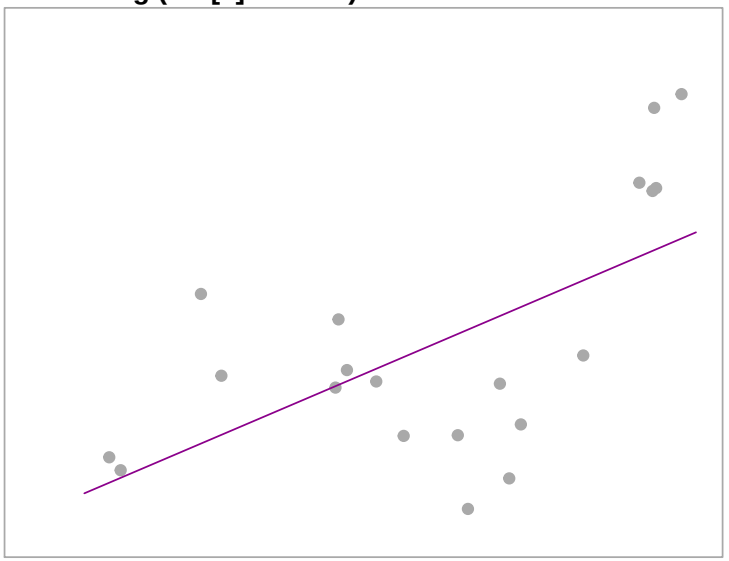


We'll try to capture the pattern in the data using a polynomial regression model.

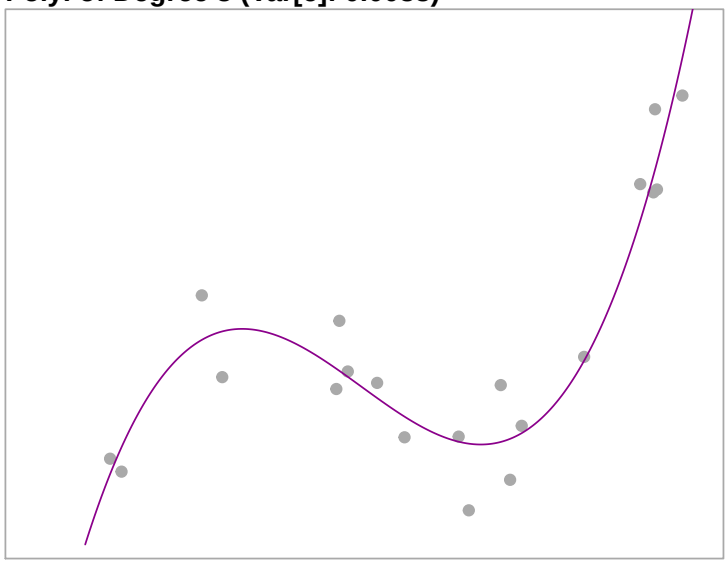
Mean Outcome (Var[e]: 0.0389)



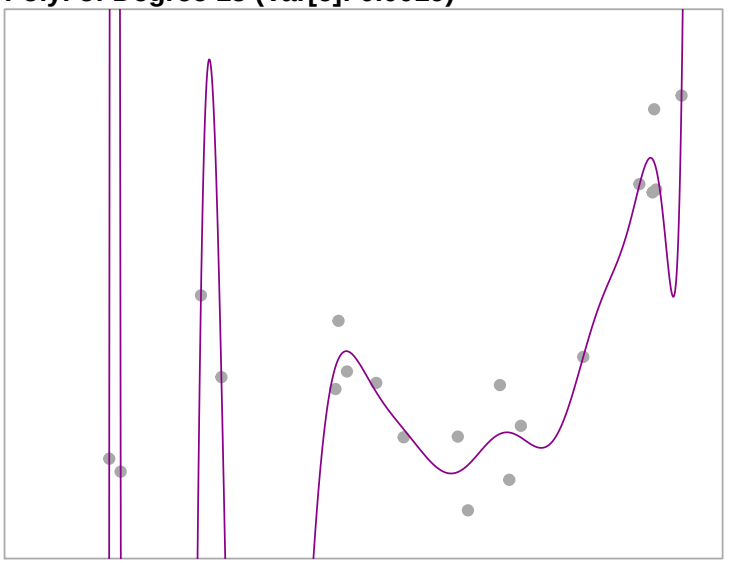
Linear Reg (Var[e]: 0.0241)



Poly. of Degree 3 (Var[e]: 0.0033)



Poly. of Degree 15 (Var[e]: 0.0019)



$L(D \mid M)$: The description length of the data *decreases* monotonically as our model becomes more complex (i.e, residual variance decreases).

$L(D \mid M)$: The description length of the data *decreases* monotonically as our model becomes more complex (i.e, residual variance decreases).

$L(M)$: The description length of the model *increases* with more complexity

$L(D \mid M)$: The description length of the data *decreases* monotonically as our model becomes more complex (i.e, residual variance decreases).

$L(M)$: The description length of the model *increases* with more complexity

$L(D \mid M) + L(M)$: The description length of the data (encoded with the help of our model) will first increase and, thereafter, decrease again.

$L(D \mid M)$: The description length of the data *decreases* monotonically as our model becomes more complex (i.e, residual variance decreases).

$L(M)$: The description length of the model *increases* with more complexity

$L(D \mid M) + L(M)$: The description length of the data (encoded with the help of our model) will first increase and, thereafter, decrease again.

The MDL principle says that we should choose the model M^* for which $L(D \mid M) + L(M)$ is minimized.

The principle helps us find the "sweet spot" between underfitting and overfitting.

The Map Equation

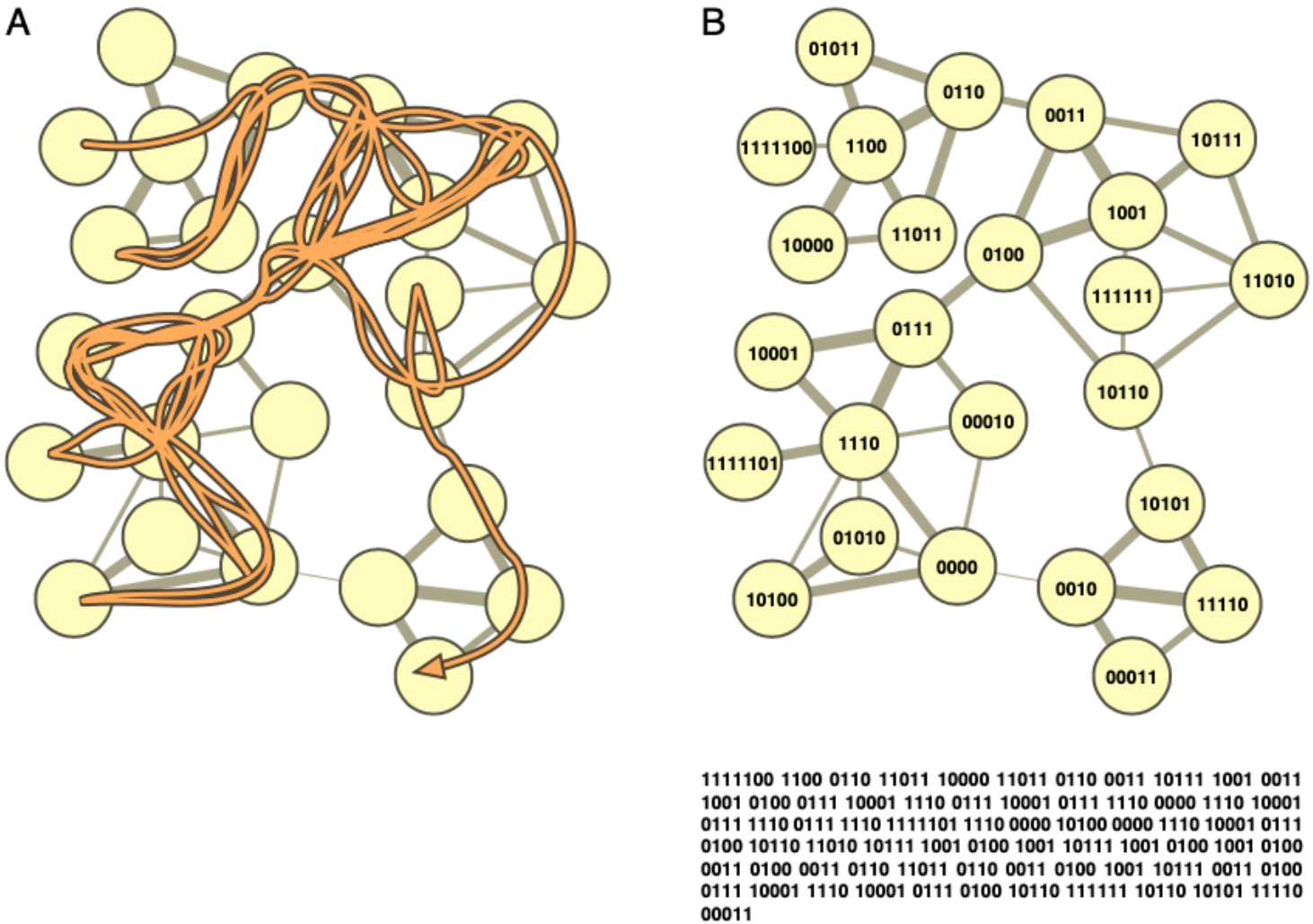
Consider describing an infinite-length random walk on a graph.

We need to describe where the random walker is at time t and, thus, need a code for each node in the graph.

Under reasonable conditions, the visiting probabilities will converge to a unique limiting distribution $\boldsymbol{\pi}_\infty$

The most efficient way to describe the trajectory of the random walker is by

1. assigning short codes to nodes with high visiting probabilities (elements of $\boldsymbol{\pi}_\infty$ that are close to 1)
2. long codes to nodes with low visiting probabilities



Source: Rosvall & Bergstrom 2008 PNAS

Here is the crucial observation for community detection:

Just as we can use a regression model to capture the pattern in data and, thereby, reduce the description length of the data, *we can exploit the clustering pattern in the network to reduce the description length of the random walk*.

Here is the crucial observation for community detection:

Just as we can use a regression model to capture the pattern in data and, thereby, reduce the description length of the data, *we can exploit the clustering pattern in the network to reduce the description length of the random walk*.

If the network is sufficiently clustered, we can shorten the descriptioin length by

1. assigning unique codes to *modules*
2. and repeatedly using the *same short codes* within each module

As long as we are able to communicate when the random walker moves from one module to another, there is no ambiguity in the code.

Here is the crucial observation for community detection:

Just as we can use a regression model to capture the pattern in data and, thereby, reduce the description length of the data, *we can exploit the clustering pattern in the network to reduce the description length of the random walk*.

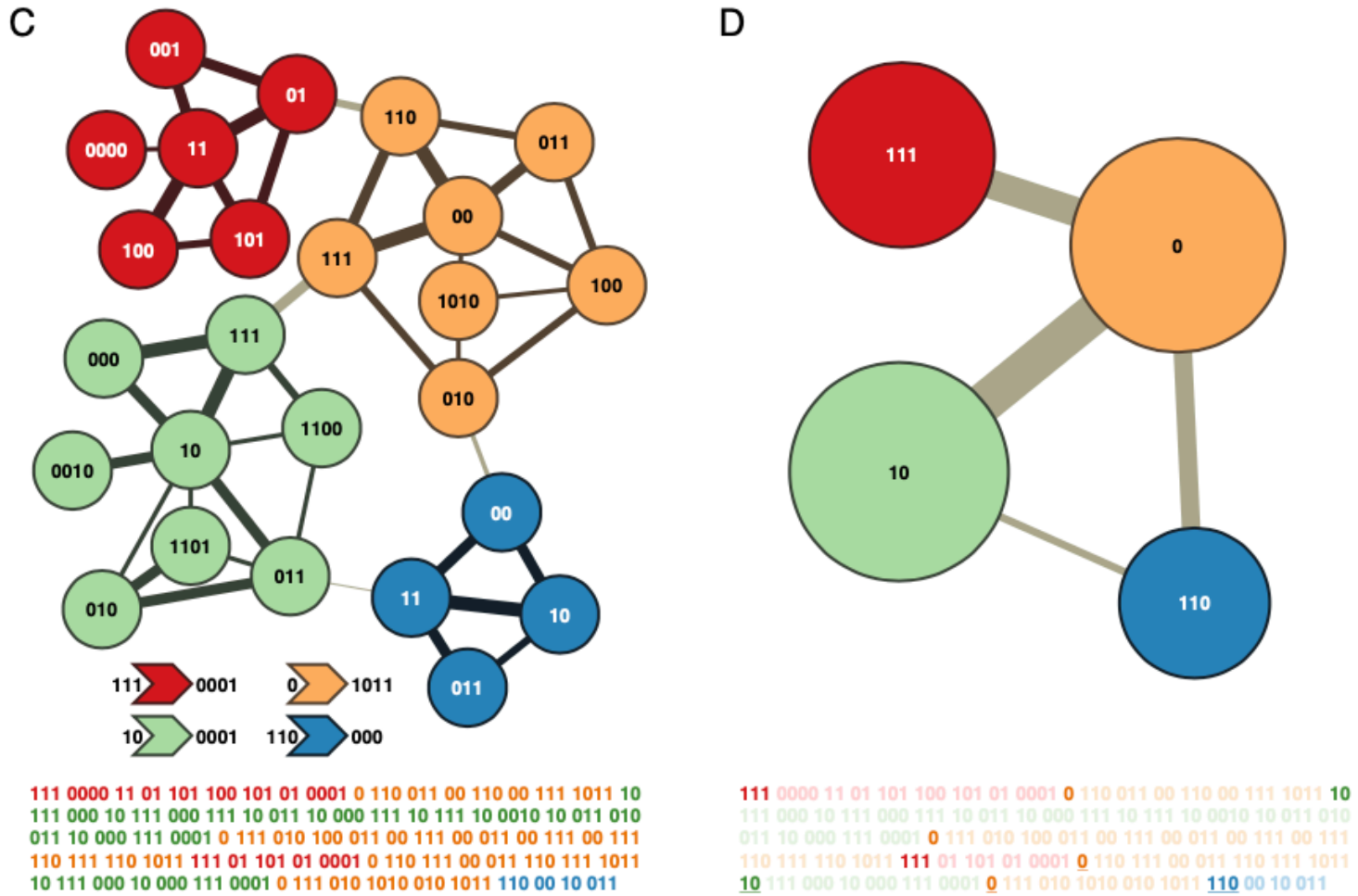
If the network is sufficiently clustered, we can shorten the descriptioin length by

1. assigning unique codes to *modules*
2. and repeatedly using the *same short codes* within each module

As long as we are able to communicate when the random walker moves from one module to another, there is no ambiguity in the code.

It's the same principle as the polynomial regression model example.

$$L(D) = L(D \mid M) + L(M).$$



Source: Rosvall & Bergstrom 2008 PNAS

The Map Equation

We don't need to assign "real" codes to the node. Instead, we rely on Shannon's (1948) Source Coding Theorems to obtain the *map equation*:

$$
\begin{aligned}
L(D) &= L(D \mid M) + L(M) \\
&= \sum_{i=1}^{m} p^{i}_{\circlearrowright} H(\mathcal{P}^{i}) + q_{\curvearrowright} H(\mathcal{Q}),
\end{aligned}
$$

(see Rosvall et al. 2009. "The map equation," Eur. Phys. J. Special Topics 178 for details.)

The Map Equation

We don't need to assign "real" codes to the node. Instead, we rely on Shannon's (1948) Source Coding Theorems to obtain the *map equation*:

$$
\begin{aligned}
L(D) &= L(D \mid M) + L(M) \\
&= \sum_{i=1}^{m} p^i_{\circlearrowright} H(\mathcal{P}^i) + q_{\curvearrowright} H(\mathcal{Q}),
\end{aligned}
$$

(see Rosvall et al. 2009. "The map equation," Eur. Phys. J. Special Topics 178 for details.)

Following the MDL principle, the partition of the node set $\mathcal{P} = \{\mathcal{P}_1, ..., \mathcal{P}_m\}$ that minimizes the expression above is chosen as the best partition.

Considerations

1. MDL is a "general principle" to find good models. Accordingly, the Map Equation can be easily generalized to find more complex structures
   - ▷ Second-order Markov dyamics (Rosvall et al., 2014, Nature Communications)
   - ▷ Finding multi-level community structures (Rosvall and Bergstrom, 2011, Plos One)
   - ▷ Finding overlapping community structures (Esquivel and Rosvall, 2011, Physical Review X)
2. The Map Equation naturally incorporates the *direction of ties* in networks
3. If you are curious about MDL, read Grünwald (2007)