TOPIC 2

Spectral and singular value decompositions

5 Graph partitioning

Course: Math 535 (http://www.math.wisc.edu/~roch/mmids/) - Mathematical Methods in Data Science (MMiDS)
Author: Sebastien Roch (http://www.math.wisc.edu/~roch/), Department of Mathematics, University of Wisconsin-Madison
Updated: Sep 21, 2020

In this section, we use the spectral properties of the Laplacian of a graph to identify cuts.

5.1 Extremal characterization of eigenvalues

We first generalize the extremal characterization used in the previous section.

Definition (Rayleigh Quotient): Let $A \in \mathbb{R}^{d\times d}$ be a symmetric matrix. The Rayleigh quotient is defined as

$$\mathcal{R}_A(\mathbf{u}) = \frac{\langle \mathbf{u}, A\mathbf{u}\rangle}{\langle \mathbf{u}, \mathbf{u}\rangle}$$

which is defined for any $\mathbf{u} \neq \mathbf{0}$ in $\mathbb{R}^d$. ◁

Exercise: Let $A \in \mathbb{R}^{d\times d}$ be a symmetric matrix. Let $\mathbf{v}$ be a (not necessarily unit) eigenvector of A with eigenvalue λ. Show that $\mathcal{R}_A(\mathbf{v}) = \lambda$. ◁

Exercise: Let $\mathcal{U}, \mathcal{V} \subseteq \mathbb{R}^d$ be subspaces such that $\dim(\mathcal{U}) + \dim(\mathcal{V}) > d$. Show there exists a non-zero vector in the intersection $\mathcal{U} \cap \mathcal{V}$. [Hint: Take the union of a basis for $\mathcal{U}$ and a basis for $\mathcal{V}$, then use linear dependence.] ◁

Theorem (Courant-Fischer): Let $A \in \mathbb{R}^{d\times d}$ be a symmetric matrix with spectral decomposition $A = \sum_{i=1}^{d} \lambda_i \mathbf{v}_i \mathbf{v}_i^T$ where $\lambda_1 \geq \cdots \geq \lambda_d$. For each $k = 1, \ldots, d$, define the subspace

$$\mathcal{V}_k = \text{span}(\mathbf{v}_1, \ldots, \mathbf{v}_k) \quad \text{and} \quad \mathcal{W}_{d-k+1} = \text{span}(\mathbf{v}_k, \ldots, \mathbf{v}_d).$$

Then, for all $k = 1, \ldots, d$,

$$\lambda_k = \min_{\mathbf{u}\in\mathcal{V}_k} \mathcal{R}_A(\mathbf{u}) = \max_{\mathbf{u}\in\mathcal{W}_{d-k+1}} \mathcal{R}_A(\mathbf{u}).$$

Furthermore we have the following min-max formulas, which do not depend on the choice of spectral decomposition, for all $k = 1, \ldots, d$

$$\lambda_k = \max_{\dim(\mathcal{V})=k} \min_{\mathbf{u}\in\mathcal{V}} \mathcal{R}_A(\mathbf{u}) = \min_{\dim(\mathcal{W})=d-k+1} \max_{\mathbf{u}\in\mathcal{W}} \mathcal{R}_A(\mathbf{u}).$$

Proof idea: For the local formula, we expand a vector in $\mathcal{V}_k$ into the basis $\mathbf{v}_1, \ldots, \mathbf{v}_k$ and use the fact that $\mathcal{R}_A(\mathbf{v}_i) = \lambda_i$ and that eigenvalues are in non-increasing order. The global formulas then follow from a dimension argument.

Proof: We first prove the local formula, that is, the one involving a specific decomposition.

Local formulas: Since $\mathbf{v}_1, \ldots, \mathbf{v}_k$ form an orthonormal basis of $\mathcal{V}_k$, any nonzero vector $\mathbf{u} \in \mathcal{V}_k$ can be written as $\mathbf{u} = \sum_{i=1}^{k} \langle \mathbf{u}, \mathbf{v}_i \rangle \mathbf{v}_i$ and it follows that

$$\langle \mathbf{u}, \mathbf{u} \rangle = \sum_{i=1}^{k} \langle \mathbf{u}, \mathbf{v}_i \rangle^2$$

$$\langle \mathbf{u}, A\mathbf{u} \rangle = \left\langle \mathbf{u}, \sum_{i=1}^{k} \langle \mathbf{u}, \mathbf{v}_i \rangle \lambda_i \mathbf{v}_i \right\rangle = \sum_{i=1}^{k} \lambda_i \langle \mathbf{u}, \mathbf{v}_i \rangle^2$$

and, thus,

$$\mathcal{R}_A(\mathbf{u}) = \frac{\langle \mathbf{u}, A\mathbf{u} \rangle}{\langle \mathbf{u}, \mathbf{u} \rangle} = \frac{\sum_{i=1}^{k} \lambda_i \langle \mathbf{u}, \mathbf{v}_i \rangle^2}{\sum_{i=1}^{k} \langle \mathbf{u}, \mathbf{v}_i \rangle^2} \geq \lambda_k \frac{\sum_{i=1}^{k} \langle \mathbf{u}, \mathbf{v}_i \rangle^2}{\sum_{i=1}^{k} \langle \mathbf{u}, \mathbf{v}_i \rangle^2} = \lambda_k$$

where we used $\lambda_1 \geq \cdots \geq \lambda_k$ and the fact that $\langle \mathbf{u}, \mathbf{v}_i \rangle^2 \geq 0$. Moreover $\mathcal{R}_A(\mathbf{v}_k) = \lambda_k$. So we have established

$$\lambda_k = \min_{\mathbf{u}\in\mathcal{V}_k} \mathcal{R}_A(\mathbf{u}).$$

The expression in terms of $\mathcal{W}_{d-k+1}$ is proved similarly.

Global formulas: Since $\mathcal{V}_k$ has dimension k, it follows from the local formula that

$$\lambda_k = \min_{\mathbf{u} \in \mathcal{V}_k} \mathcal{R}_A(\mathbf{u}) \leq \max_{\dim(\mathcal{V})=k} \min_{\mathbf{u} \in \mathcal{V}} \mathcal{R}_A(\mathbf{u}).$$

Let $\mathcal{V}$ be any subspace with dimension k. Because $\mathcal{W}_{d-k+1}$ has dimension $d - k + 1$, we have that $\dim(\mathcal{V}) + \dim(\mathcal{W}_{d-k+1}) > d$ and there must be non-zero vector $\mathbf{u}_0$ in the intersection $\mathcal{V} \cap \mathcal{W}_{d-k+1}$ by the exercise above. We then have by the other local formula that

$$\lambda_k = \max_{\mathbf{u} \in \mathcal{W}_{d-k+1}} \mathcal{R}_A(\mathbf{u}) \geq \mathcal{R}_A(\mathbf{u}_0) \geq \min_{\mathbf{u} \in \mathcal{V}} \mathcal{R}_A(\mathbf{u}).$$

Since this inequality holds for any subspace of dimensiona k, we have

$$\lambda_k \geq \max_{\dim(\mathcal{V})=k} \min_{\mathbf{u} \in \mathcal{V}} \mathcal{R}_A(\mathbf{u}).$$

Combining with inequality in the other direction above gives the claim. The other global formula is proved similarly. □

5.2 Returning to the Laplacian

We record one important special case of Courant-Fischer for the Laplacian.

Corollary (Extremal Characterization of μ_2): Let $G = (V, E)$ be a graph with $n = |V|$ vertices. Assume the Laplacian L of G has spectral decomposition $L = \sum_{i=1}^{n} \mu_i \mathbf{y}_i \mathbf{y}_i^T$ with $0 = \mu_1 \leq \mu_2 \leq \cdots \leq \mu_n$ and $\mathbf{y}_1 = \frac{1}{\sqrt{n}}(1, \ldots, 1)^T$. Then

$$\mu_2 = \min \left\{ \frac{\sum_{\{u,v\} \in E} (x_u - x_v)^2}{\sum_{u=1}^{n} x_u^2} \; : \; \mathbf{x} = (x_1, \ldots, x_n)^T \neq \mathbf{0}, \sum_{u=1}^{n} x_u = 0 \right\}.$$

Proof: By *Courant-Fischer*,

$$\mu_2 = \min_{\mathbf{x} \in \mathcal{W}_{d-1}} \mathcal{R}_L(\mathbf{x}).$$

Since $\mathbf{y}_1$ is constant and $\mathcal{W}_{d-1}$ the subspace orthogonal to it, this is equivalent to restrictring the minimization to those non-zero $\mathbf{x}$'s such that

$$0 = \langle \mathbf{x}, \mathbf{y}_1 \rangle = \frac{1}{\sqrt{n}} \sum_{u=1}^{m} x_u.$$

Moreover, by the *Laplacian Quadratic Form Lemma*,

$$\langle \mathbf{x}, L\mathbf{x} \rangle = \sum_{\{u,v\} \in E} (x_u - x_v)^2$$

so the Rayleigh quotient is

$$\mathcal{R}_L(\mathbf{x}) = \frac{\langle \mathbf{x}, L\mathbf{x} \rangle}{\langle \mathbf{x}, \mathbf{x} \rangle} = \frac{\sum_{\{u,v\} \in E} (x_u - x_v)^2}{\sum_{u=1}^{n} x_u^2}.$$

That proves the claim. □

One application of this extremal characterization is the graph drawing heuristic we described previously. Consider the entries of the second Laplacian eigenvector $\mathbf{y}_2$ normalized to have unit norm. The entries are centered around 0 by the condition $\sum_{u=1}^{n} x_u = 0$. Because it minimizes the quantity

$$\frac{\sum_{\{u,v\} \in E} (x_u - x_v)^2}{\sum_{u=1}^{n} x_u^2}$$

over all centered unit vectors, $\mathbf{y}_2$ tends to assign similar coordinates to adjacent vertices. A similar reasoning applies to the third Laplacian eigenvector, which in addition is orthogonal to the second one.

NUMERICAL CORNER This is perhaps easiest to see on a path graph.

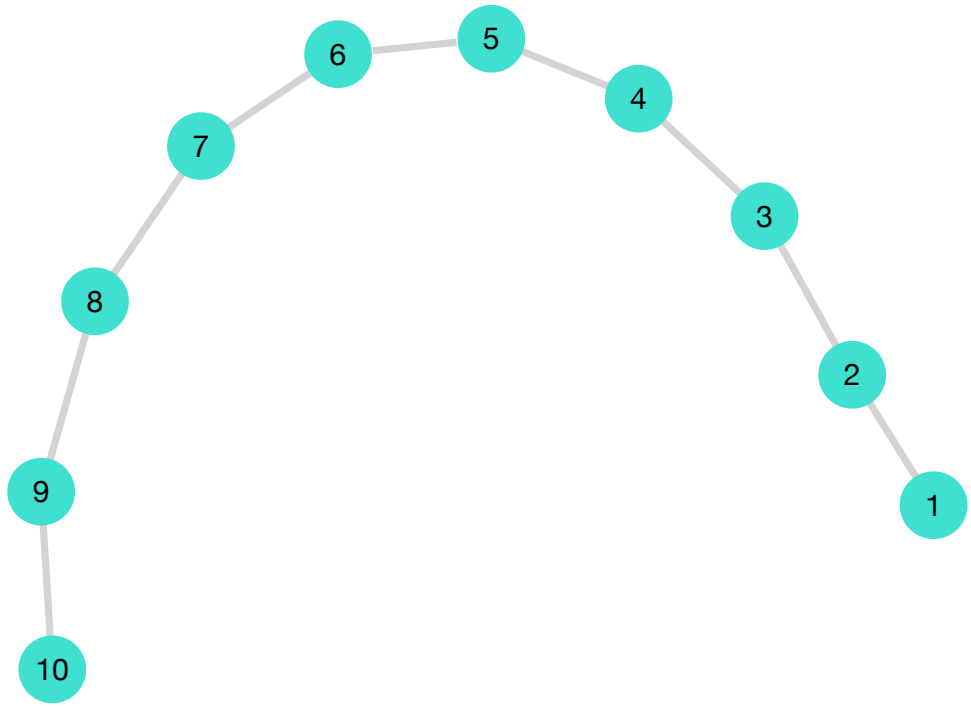
In [1]:
```
# Julia version: 1.5.1
using LightGraphs, GraphPlot, LinearAlgebra, Plots
```

In [2]:
```
n = 10
g = path_graph(n)
```

Out[2]:
```
{10, 9} undirected simple Int64 graph
```

In [3]:
```
gplot(g, nodelabel=1:n)
```

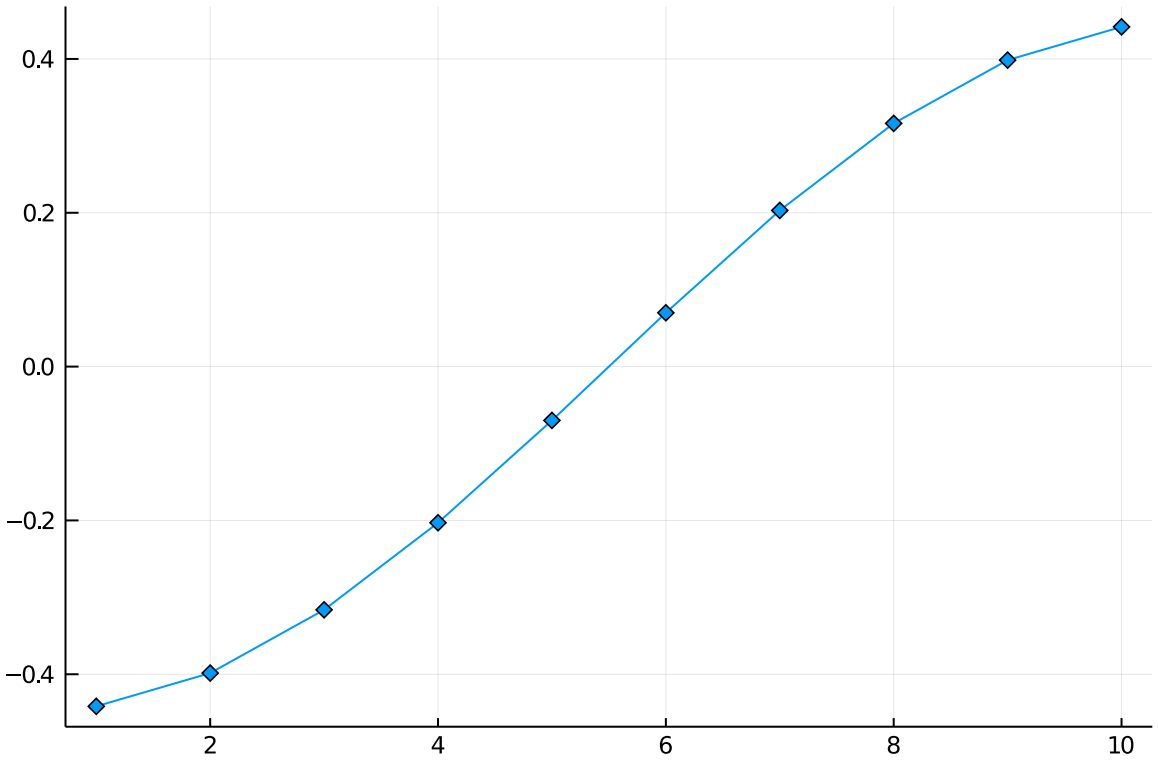
Out[3]:



We plot the second Laplacian eigenvector.

In [4]:
```
L = Array(laplacian_matrix(g))
F = eigen(L)
v = F.vectors
plot(v[:,2], legend=false, marker=:diamond)
```

Out[4]:



5.3 How to cut a graph

Let $G = (V, E)$ be a graph. Imagine that we are interested in finding a good cut. That is, roughly speaking, we seek to divide it into two disjoint subsets of vertices to achieve two goals simultaneously:

1. the two sets have relatively few edges between them
2. neither set is too small.

We will show that the Laplacian eigenvectors provide useful information in order to perform this kind of graph cutting. First we formulate the problem formally.

5.3.1 Cut ratio

One way to make the graph cutting more precise is to consider the following combinatorial quantity.

Definition (Isoperimetric Number): Let $G = (V, E)$ be a graph. A cut is a bipartition (S, S^c) of the vertices of G, where S and S^c are non-empty subsets of V. For short, we refer to $\emptyset \neq S \subset V$ as a cut. The size of the cut, $|\partial S|$, is the number of edges from S to S^c, where the edge boundary is defined as

$$\partial S = \{\{i, j\} \in E : i \in S, j \in S^c\}.$$

The cut ratio of S is

$$\phi(S) = \frac{|\partial S|}{\min\{|S|, |S^c|\}}$$

and the isoperimetric number (or Cheeger constant (https://en.wikipedia.org/wiki/Spectral_graph_theory#Cheeger_constant)) of G is the minimum cut ratio over all of its cuts, that is,

$$\phi_G = \min_{\emptyset \neq S \subset V} \phi(S).$$

◁

In words: the cut ratio is attempting to minimize the number of edges across a cut, while penalizing cuts with a small number of vertices. These correspond to the goals above and we will use this criterion to assess the quality of graph cuts.

Exercise: Show that the isoperimetric number of G is equal to

$$\phi_G = \min\left\{\frac{|\partial S|}{|S|} \;:\; S \subseteq V, 0 < |S| \leq \frac{1}{2}|V|\right\}.$$

◁

5.3.2 Cheeger inequality

A key result of spectral graph theory establishes a quantitative relation between the isoperimetric number and the second smallest Laplacian eigenvalue.

Theorem (Cheeger Inequality): Let $G = (V, E)$ be a graph with $n = |V|$ vertices and maximum degree $\bar{\delta}$. Let $0 = \mu_1 \leq \mu_2 \leq \cdots \leq \mu_n$ be its Laplacian eigenvalues. Then

$$\frac{\phi_G^2}{2\bar{\delta}} \leq \mu_2 \leq 2\phi_G.$$

We only prove the easy direction, which shows explicitly how the connection between μ_2 and ϕ_G comes about.

Proof idea: We find an appropriate test vector to plug into the extremal characterization of μ_2 and link it to ϕ_G.

Proof: Recall that, from the *Extremal Characterization of* μ_2, we have

$$\mu_2 = \min \left\{ \frac{\sum_{\{u,v\}\in E}(x_u - x_v)^2}{\sum_{u=1}^n x_u^2} \; : \; \mathbf{x} = (x_1, \ldots, x_n)^T \neq \mathbf{0}, \sum_{u=1}^n x_u = 0 \right\}.$$

Constructing a good test vector: We construct an $\mathbf{x}$ that provides a good upper bound. Let $\emptyset \neq S \subset V$ be a proper, nonempty subset of V such that $0 < |S| \leq \frac{1}{2}|V|$. Consider the vector $\mathbf{x} \in \mathbb{R}^n$ with entries

$$x_i = \begin{cases} |S| & \text{if } i \in S^c \\ -|S^c| & \text{if } i \in S. \end{cases}$$

This choice ensures that

$$\sum_{i=1}^n x_i = \sum_{i\in S^c} |S| + \sum_{i\in S}(-|S^c|) = |S^c||S| - |S||S^c| = 0.$$

and

$$\begin{aligned} \sum_{i=1}^n x_i^2 &= \sum_{i\in S^c} |S|^2 + \sum_{i\in S}(-|S^c|)^2 \\ &= |S^c||S|^2 + |S||S^c|^2 \\ &= |S^c||S|(|S| + |S^c|) \\ &= n|S^c||S|. \end{aligned}$$

So

$$\sum_{\{i,j\}\in E} (x_i - x_j)^2 = \sum_{\{i,j\}\in E} (|S| + |S^c|)^2 = n^2 |\partial S|$$

where we used that, for each edge $\{i, j\} \in E$, one endvertex is in S and one endvertex is in S^c. Hence $x_i - x_j$ is either $|S| + |S^c|$ or $-|S| - |S^c|$.

Using the definition of the isoperimetric number: So for this choice of $\mathbf{x}$ we have

$$\mu_2 \leq \frac{\sum_{\{i,j\}\in E}(x_i - x_j)^2}{\sum_{i=1}^n x_i^2} = \frac{n^2|\partial S|}{n|S^c||S|} = \frac{|\partial S|}{(|S^c|/n)|S|} \leq 2\frac{|\partial S|}{|S|}$$

where we used that $|S^c| \geq n/2$. This inequality holds for any S such that $0 < |S| \leq \frac{1}{2}|V|$. In particular, it holds for the S producing the smallest value. Hence, by the definition of the isoperimetric number (see the exercise above), we get

$$\mu_2 \leq 2\phi_G$$

as claimed. □

5.3.3 A graph cutting algorithm

We only proved the easy direction of the Cheeger Inequality. It is however useful to sketch the other direction, as it contains an important algorithmic idea.

An algorithm: Let $\mathbf{y}_2 \in \mathbb{R}^n$ be the unit-norm eigenvector of L associated to μ_2. There is one entry of $\mathbf{y}_2 = (y_{2,1}, \ldots, y_{2,n})^T$ for each vertex of G. We use these entries to embed the graph G in $\mathbb{R}$: vertex i is mapped to $y_{2,i}$. Now order the entries $y_{2,\pi(1)}, \ldots, y_{2,\pi(n)}$, where π is a permutation (https://en.wikipedia.org/wiki/Permutation), that is, a re-ordering of $1, \ldots, n$. Specifically, $\pi(1)$ is the vertex corresponding to the smallest entry of $\mathbf{y}_2$, $\pi(2)$ is the second smallest, and so on. We consider only cuts of the form

$$S_k = \{\pi(1), \ldots, \pi(k)\}$$

and we choose the $k \leq n/2$ that minimizes the cut ratio

$$\phi(S_k) = \frac{|\partial S_k|}{|S_k|}.$$

What can be proved rigorously is that there exists some k such that

$$\sum_{\{u,v\} \in E} (y_{2,u} - y_{2,v})^2 \geq \frac{\phi(S_k)^2}{2\bar{\delta}}$$

which implies the Cheeger Inequality.

See, for example, [Kel (https://ocw.mit.edu/courses/mathematics/18-409-topics-in-theoretical-computer-science-an-algorithmists-toolkit-fall-2009/index.htm), Lecture 3, Section 4.2].

The above provides a heuristic to find a good cut with provable guarantees. We implement it next. In contrast, the problem of finding a cut which mimizes the cut ratio is known to be NP-hard (https://en.wikipedia.org/wiki/NP-hardness), that is, computationally intractable.

NUMERICAL CORNER We implement the graph cutting algorithm above.

We now implement our heuristic in Julia. We first write an auxiliary function that takes as input an adjacency matrix, and ordering of the vertices and a value k. It returns the cut ratio for the first k vertices in the order.

In [5]:
```
function cut_ratio(A, order, k, n)
    edge_boundary = 0 # initialize size of edge boundary
    for i=1:k # for all vertices before cut
        for j=k+1:n # for all vertices after cut
            edge_boundary += A[order[i],order[j]] # add one if {i,j} in
 E
        end
    end
    denominator = min(k, n-k)
    return edge_boundary/denominator
end
```

Out[5]:
```
cut_ratio (generic function with 1 method)
```

Using the `cut_ratio` function, we first compute the Laplacian, find the second eigenvector and corresponding order of vertices, and then compute the cut ratio for every k. Finally we output the cut (both S_k and S_k^c) corresponding to the minimum, as a tuple of arrays.

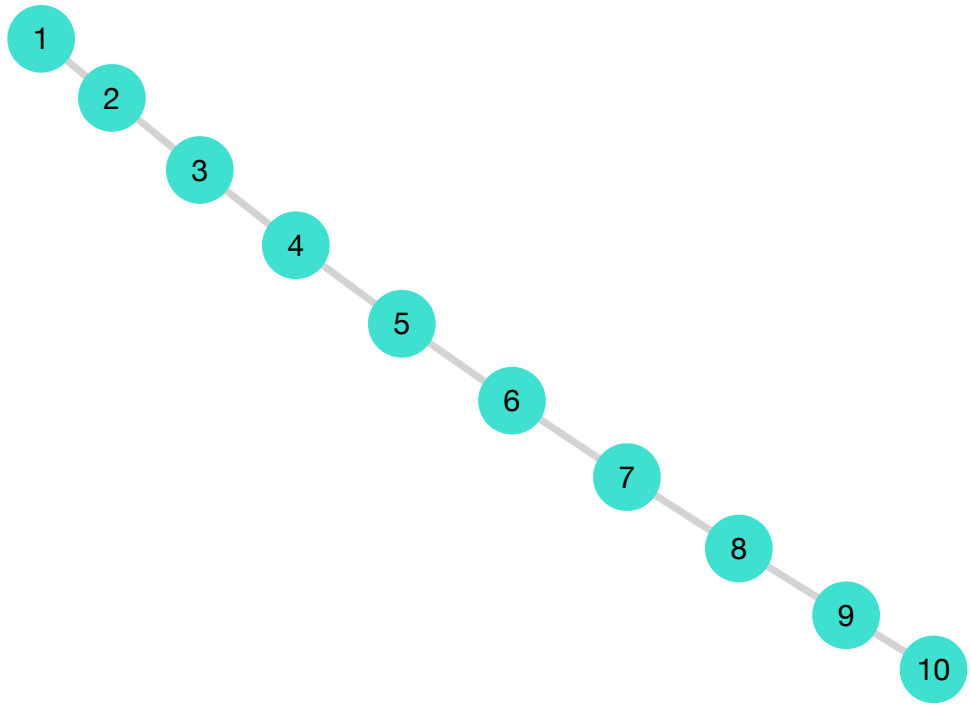
In [6]:
```
function spectral_cut2(A)
    n = size(A,1) # number of vertices
    L = Diagonal(reshape(sum(A,dims=2), n)) - A # construct Laplacian
    v = eigvecs(L) # eigenvectors of L
    order = sortperm(v[:,2]) # index of entries in increasing order
    phi = zeros(Float64, n-1) # initialize cut ratios
    for k=1:n-1
        phi[k] = cut_ratio(A, order, k, n)
    end
    (phimin, imin) = findmin(phi) # find best cut ratio
    return order[1:imin], order[imin+1:end]
end
```

Out[6]:
```
spectral_cut2 (generic function with 1 method)
```

We will illustrate this on the path graph.

In [7]:
```
n = 10
g = path_graph(n)
gplot(g, nodelabel=1:n)
```

Out[7]:



We first construct the Laplacian matrix.

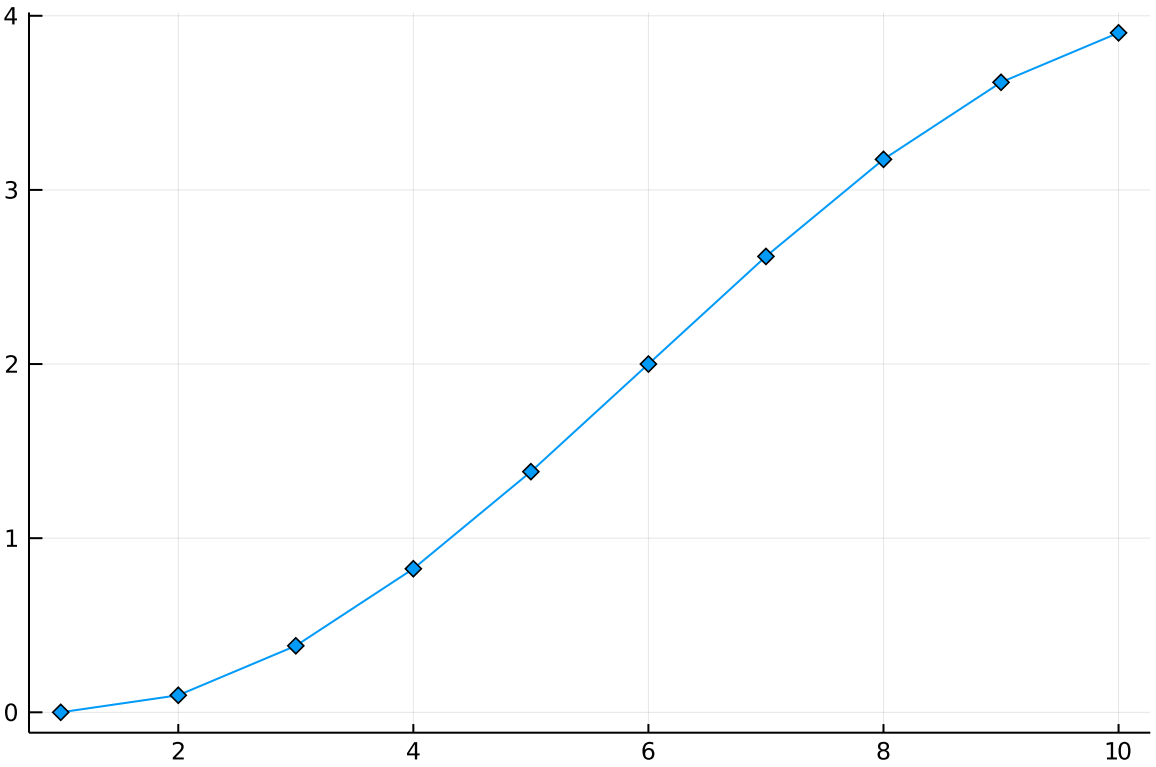
In [8]:
```
A = Array(adjacency_matrix(g))
degrees = reshape(sum(A,dims=2), n)
D = Diagonal(degrees)
L = D - A
```

Out[8]:
```
10×10 Array{Int64,2}:
  1  -1   0   0   0   0   0   0   0   0
 -1   2  -1   0   0   0   0   0   0   0
  0  -1   2  -1   0   0   0   0   0   0
  0   0  -1   2  -1   0   0   0   0   0
  0   0   0  -1   2  -1   0   0   0   0
  0   0   0   0  -1   2  -1   0   0   0
  0   0   0   0   0  -1   2  -1   0   0
  0   0   0   0   0   0  -1   2  -1   0
  0   0   0   0   0   0   0  -1   2  -1
  0   0   0   0   0   0   0   0  -1   1
```

We compute the spectral decomposition of L and plot the eigenvalues and the entries of the second eigenvector.

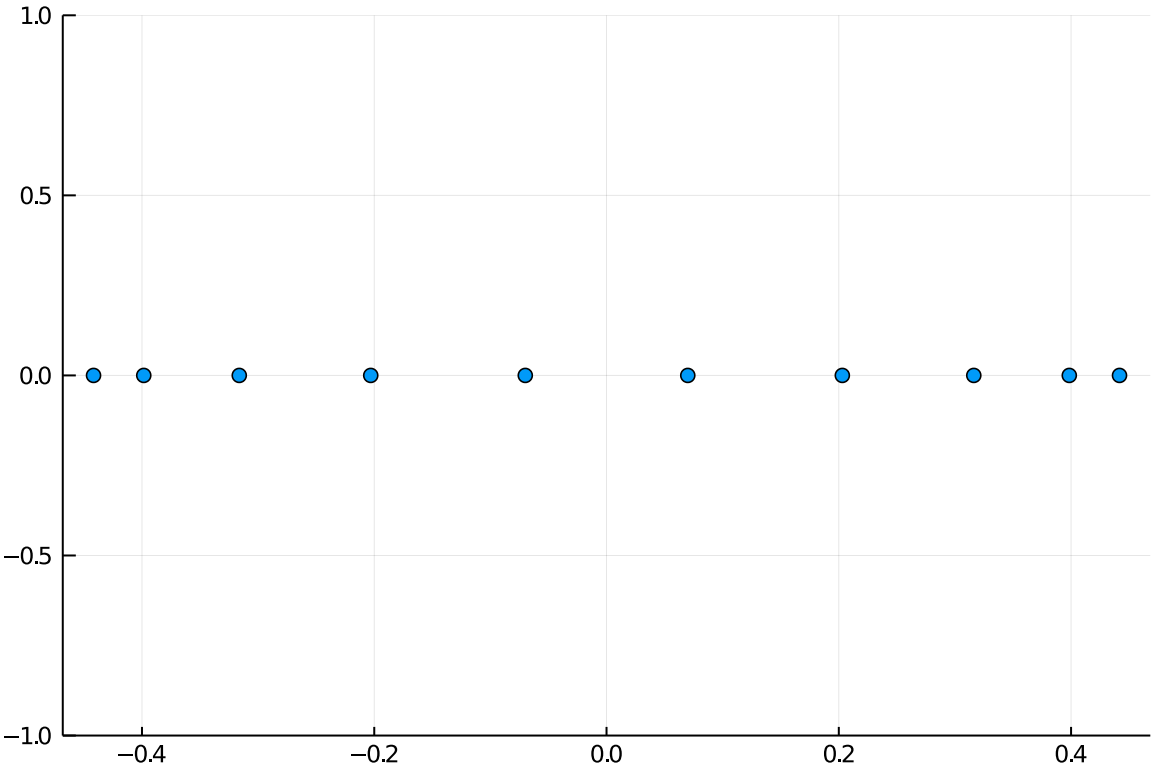
In [9]:
```
F = eigen(L)
lambda = F.values
plot(lambda, legend=false, marker=:diamond)
```

Out[9]:



In [10]:
```
v = F.vectors
scatter(v[:,2], zeros(n), legend=false, ylims=(-1,1))
```

Out[10]:



Finally, we apply our spectral-based cutting algorihtm.

In [11]:
```
(s, sc) = spectral_cut2(A)
```

Out[11]:
```
([1, 2, 3, 4, 5], [6, 7, 8, 9, 10])
```

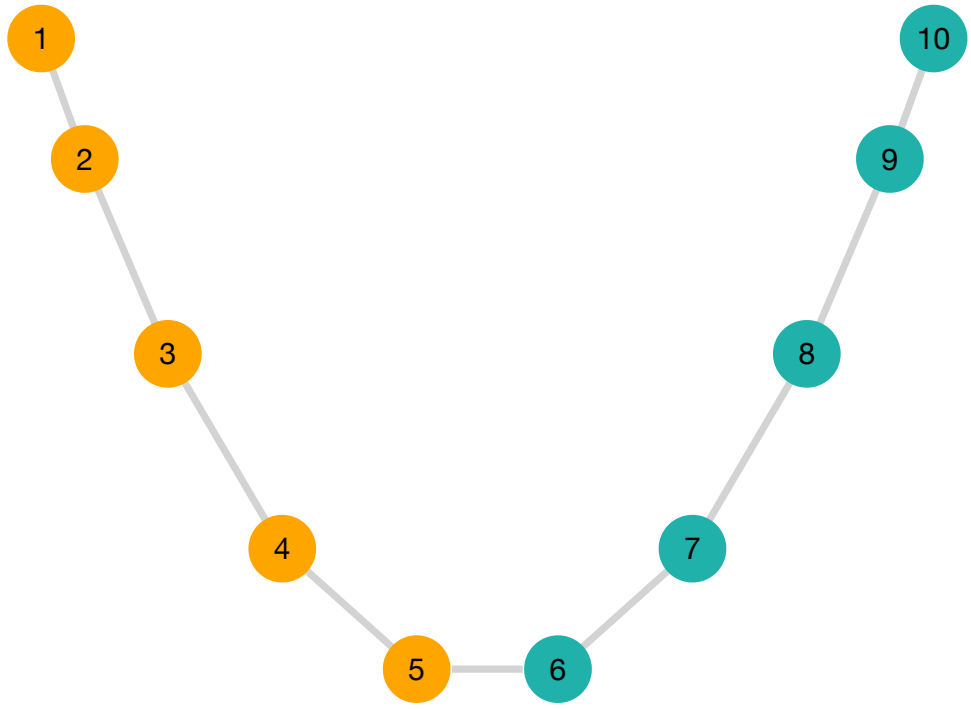
To help with vizualizing the output, we write a function coloring the vertices according to which side of the cut they are on. We use the option `nodefillc` (https://github.com/JuliaGraphs/GraphPlot.jl#control-the-node-color) of `gplot`.

In [12]:
```
function viz_cut(g, s; layout=spectral_layout)
    n = nv(g)
    assign = ones(Int64, n)
    assign[s] .= 2
    colors = [colorant"lightseagreen", colorant"orange"]
    nodecolor = colors[assign]
    gplot(g, nodelabel=1:n, nodefillc=nodecolor, layout=layout)
end
```

Out[12]:
```
viz_cut (generic function with 1 method)
```

In [13]:
```
viz_cut(g, s)
```

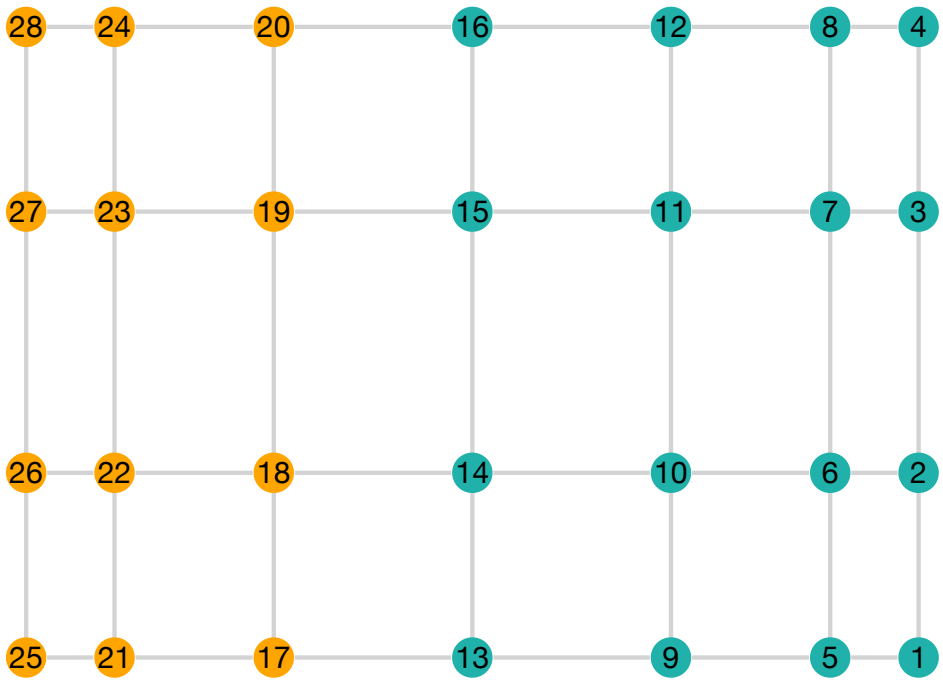
Out[13]:



Let's try it on the grid graph. Can you guess what the cut will be?

In [14]:
```
g = LightGraphs.SimpleGraphs.grid([4, 7])
A = Array(adjacency_matrix(g))
(s, sc) = spectral_cut2(A)
viz_cut(g, s)
```

Out[14]:



5.4 Weighted Laplacian

The concepts we have introduced can also be extended to weighted graphs, that is, graphs with weights on the edges. These weights might be a measure of the strength of the connection for instance. In this section, we briefly describe this extension, which is the basis for a [discrete calculus (https://en.wikipedia.org/wiki/Calculus_on_finite_weighted_graphs)](https://en.wikipedia.org/wiki/Calculus_on_finite_weighted_graphs).

Definition (Weighted Graph): A weighted graph is a triple $G = (V, E, w)$ where (V, E) is a graph and $w : E \to \mathbb{R}$ is a function that assigns real weights to the edges. For ease of notation, we write $w_e = w_{ij} = w(i, j)$ for the weight of edge $e = \{i, j\}$. ◁

As we did for graphs, we denote the vertices $\{1, \ldots, n\}$ and the edges $\{e_1, \ldots, e_m\}$, where $n = |V|$ and $m = |E|$. Properties of graphs can be generalized naturally. For instance, one defines the degree of a vertex i as

$$\delta(i) = \sum_{j:\{i,j\}\in E} w_{ij}.$$

Definition (Weighted Adjacency Matrix): Let $G = (V, E, w)$ be a weighted graph with $n = |V|$ vertices. The weight matrix A_w of G is the $n \times n$ symmetric matrix defined as

$$(A_w)_{ij} = \begin{cases} w_{ij} & \text{if } \{i, j\} \in E \\ 0 & \text{o.w.} \end{cases}$$

◁

The Laplacian can then be defined as follows.

Definition (Weighted Laplacian): Let $G = (V, E, w)$ be a weighted graph with $n = |V|$ vertices and weight matrix A_w. Let $D_w = \operatorname{diag}(\delta(1), \ldots, \delta(n))$ be the weighted degree matrix. The weighted Laplacian matrix associated to G is defined as $L_w = D_w - A_w$. ◁

It can be shown that the Laplacian quadratic form satisfies

$$\langle \mathbf{x}, L_w \mathbf{x} \rangle = \sum_{\{i,j\}\in E} w_{ij}(x_i - x_j)^2$$

for $\mathbf{x} = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$.

Exercise: Prove the formula above for the Laplacian quadratic form. [Hint: For an orientation $G^\sigma = (V, E^\sigma)$ of G (that is, give an arbitrary direction to each edge to turn it into a digraph), consider the matrix $B^\sigma \in \mathbb{R}^{n\times m}$ where the column corresponding to arc (i, j) has $-\sqrt{w_{ij}}$ in row i and $\sqrt{w_{ij}}$ in row j, and every other entry is 0.] ◁

As a PSD matrix (Exercise: Why?), the weighted Laplacian has an orthonormal basis of eigenvectors with nonnegative eigenvalues that satisfy the extremal characterization we derived above. In particular, if we denote the eigenvalues $0 = \mu_1 \leq \mu_2 \leq \cdots \leq \mu_n$, it follows that

$$\mu_2 = \min \left\{ \frac{\sum_{\{u,v\} \in E} w_{uv}(x_u - x_v)^2}{\sum_{u=1}^n x_u^2} \ : \ \mathbf{x} = (x_1, \ldots, x_n)^T \neq \mathbf{0}, \sum_{u=1}^n x_u = 0 \right\}.$$

If we generalize the cut ratio as

$$\phi(S) = \frac{\sum_{i \in S, j \in S^c} w_{ij}}{|S|}$$

for $0 < |S| \leq |V|/2$ and let

$$\phi_G = \min \{\phi(S) \ : \ S \subseteq V, 0 < |S| \leq |V|/2\}$$

it can be shown that

$$\mu_2 \leq 2\phi_G$$

as in the unweighted case.

Exercise: Prove the inequality $\mu_2 \leq 2\phi_G$. [Hint: Try the test vector $\mathbf{x} \in \mathbb{R}^n$ with entries

$$x_i = \begin{cases} \sqrt{\frac{|S|}{|S^c|}} & \text{if } i \in S^c \\ -\sqrt{\frac{|S^c|}{|S|}} & \text{if } i \in S. \end{cases}$$

and follow the proof for the unweighted case.] ⊲

See this tutorial (https://arxiv.org/abs/0711.0189) for more on graph cutting and, in particular, for other notions of cut ratio.