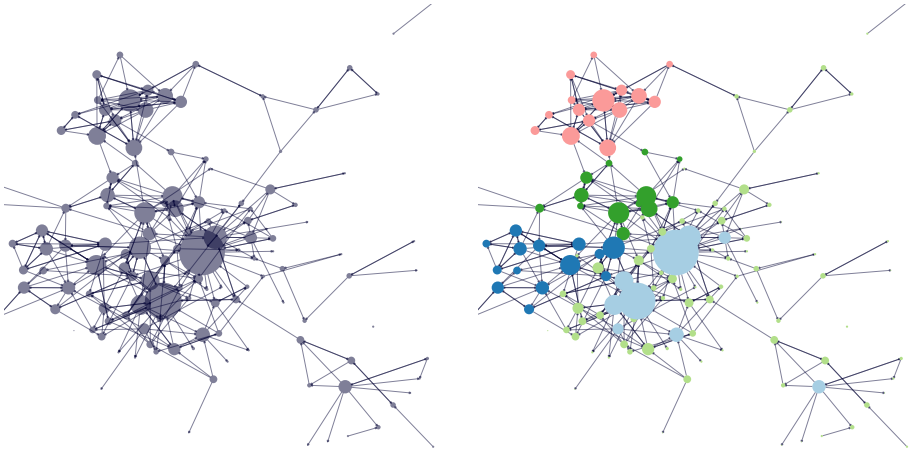
Chapter 4

Random graphs and stochastic block models

4.1 Introduction

So far, we only dealt with tabular data in the form $x_1, \dots, x_N$ with each x_i being a vector living in $\mathbb{R}^D$ and considered as the observed realisation of a random variable X_i following some multivariate distribution. We used DAGs in order to model the factorization properties of such distribution and/or of the entire model likelihood. The key point is that in the graphs we considered the nodes were (associated with) random variables and edges were fixed and modelled causality relations between the observed and/or latent nodes. With a slight abuse of notation we can say that the observed data lied on the nodes of the graph. In this chapter, instead we consider graphs whose N nodes are fixed (not random) and whose edges are random. So the data we observe lies on the edges of the graph and, in more detail, inside its adjacency matrix $A \in \{0,1\}^{N \times N}$, which we introduced in Chapter 2. In this context, we use the words "graph" and "network" as synonyms and in the simplest case of binary networks, A_{ij} is a Bernoulli random variable whose probability of success we try to infer (in some sense). Random graphs are used to model networks of interactions in several fields, including but not limited to economy, biology, physics and social sciences. An example of social network can be seen in Figure 4.1a. It is the famous Enron communication network, built from all e-mail exchanges between 149 employees of the company, that failed bankruptcy in December 2001. The original dataset is available at



(a) Enron communication network. (b) Nodes clustering.

Figure 4.1: E-mail exchanges between the Enron employees between September, 3rd, 2001 and January, 28th, 2002.

`http://www.cs.cmu.edu/~./enron/` and cover the time horizon 1999-2002. The time window considered to build the graph in Figure 4.1a spans from September, 3rd, 2001 to January, 28th, 2002. The nodes of the graph represent the employees. A link connects two nodes whether they had a direct e-mail exchange during the period. When observing a single graph like the one in Figure 4.1a, it might be difficult to capture relevant information or uncover hidden patterns. One (among many other) solution to address such issues is community detection, which is part of a larger discipline known as *social network analysis* (SNA, Tabassum et al., 2018). A **community** is a group of densely connected nodes, having few interactions outside the group. Community detection can be (and generally it is) done by discriminative approaches who do *not* need to see and edge as the realisation of random variable. However, partitioning the network's nodes into communities is an instance of **nodes clustering** (see Figure 4.1b), but not the only one. One on the main advantages of the model based approach that we introduce in this chapter, is that it is capable to detect several structures (e.g. hubs and stars) beyond communities. So nodes clustering is the main focus of this chapter. We now move to the mathematical details of one of the most famous generative models allowing one to perform nodes clustering in networks.

4.2 Stochastic block model(s)

In order to fix the ideas, let us assume that we work with undirected graphs, with no self loops. Everything we state is this chapter can easily be generalized to directed graphs. Since A_{ij} is a Bernoulli random variable, the easiest way to model a random graph would be to say that $A_{ij} \sim \text{Ber}(\pi)$ independently for all $i \in \{1, \dots, N\}$ and $j \in \{i+1, \dots, N\}$. Such a model exists, it is know as Erdös-Rényi and assumes that the observed adjacency matrix A is the realisation of $N(N-1)/2$ i.i.d. random variables following a Bernoulli distribution of parameter π. However, this model was shown to be by far too simplistic to represent real networks (Daudin et al., 2008). In particular the degree of the i-th node $d_i := \sum_{j \neq i} A_{ij}$ follows a binomial distribution of parameters $(N-1, \pi)$ (why?) which in large and sparse networks is well approximated (why?) by a Poisson distribution of parameter $(N-1)\pi$. Poisson distribution does *not* fit the degrees distribution in real networks, which is more likely a scale-free distribution[1].

Stochastic block model (SBM) is an attempt to overcome the limitations of Erdös-Rényi by means of mixtures. In more detail, assume that the nodes of the observed network are partitioned into K groups and a *latent* vector $\mathbf{z} := (z_1, \dots, z_N)$ exists, labelling the nodes' membership to one cluster. As in the previous chapter, with a slight abuse of notation z_i can denote either the integer $k \leq K$ corresponding to the cluster of the i-th node, or a binary vector having a one in k-th position and zero elsewhere (hard clustering). SBM assumes that z_i is the realisation of a random variable Z_i following a categorical distribution of parameter $\alpha = (\alpha_1, \dots, \alpha_Q)$:

$$\mathbb{P}(Z_i = k | \alpha) = \alpha_k,$$

for all $k \leq K$, independently for all i. The corresponding probability mass function is

$$p(\mathbf{z}|\alpha) = \prod_{i=1}^{N} \alpha_{z_i} = \prod_{i=1}^{N} \prod_{k=1}^{K} \alpha_k^{z_{ik}}. \tag{4.1}$$

A more relevant assumption in SBM is that the probability of and edge between two nodes only depends on the clusters they are in

$$\mathbb{P}(A_{ij} = 1 | \mathbf{Z}, \Pi) = \pi_{Z_i Z_j}, \tag{4.2}$$

[1] https://en.wikipedia.org/wiki/Scale-free_network.

where Π denotes a matrix $K \times K$ whose entry π_{kl} is the probability of one interaction between any node in cluster k and any node in cluster l. The other fundamental assumption in SBM is that all the entries of A are *conditionally* independent given $\mathbf{Z}$. As such the conditional likelihood of A given $\mathbf{Z} = \mathbf{z}$ is

$$\begin{aligned} p(A|\mathbf{z}, \Pi) &= \prod_{i=1}^{N} \prod_{j>i}^{N} \pi_{z_i z_j}^{A_{ij}} (1 - \pi_{z_i z_j})^{1 - A_{ij}} \\ &= \prod_{i=1}^{N} \prod_{j>i}^{N} \prod_{k=1}^{K} \prod_{l=1}^{K} \left(\pi_{kl}^{A_{ij}} (1 - \pi_{kl})^{1 - A_{ij}} \right)^{z_{ik} z_{jl}} . \end{aligned} \tag{4.3}$$

Combining Eqs. (4.1) and (4.3) and taking the logarithm we obtain the log-likelihood of the complete data

$$\begin{aligned} \log p(A, \mathbf{z}|\Pi, \alpha) = \frac{1}{2} \sum_{j \neq i}^{N} \sum_{k,l}^{K} (z_{ik} z_{jl} A_{jl} \log \pi_{kl} + z_{ik} z_{jl} (1 - A_{ij}) \log(1 - \pi_{kl})) \\ + \sum_{i=1}^{N} \sum_{k=1}^{K} z_{ik} \log \alpha_k \end{aligned} \tag{4.4}$$

where we adopted the shorthand notations $\sum_{j \neq i}^{N} := \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N}$ and $\sum_{k,l}^{K} := \sum_{k=1}^{K} \sum_{l=1}^{K}$ and used the symmetry of A in order to sum over all the out-of-diagonal terms of A and divide it by two. Based on the above log-likelihood, the complete data ML estimates are (**exercise**):

$$\begin{aligned} \hat{\pi}_{kl} &:= \frac{\sum_{j \neq i}^{N} z_{ik} z_{jl} A_{ij}}{\sum_{j \neq i}^{N} z_{ik} z_{jl}} \\ \hat{\alpha}_k &= \frac{1}{N} \sum_{i=1}^{N} z_{ik} \end{aligned} \tag{4.5}$$

Note, however, that in practice those estimates are worthless since $\mathbf{z}$ is not observed.

In Figure 4.2 we see the graphical model corresponding to the SBM described so far at the level of the nodes pair (i, j). It is important to underline that this graphical representation is *incomplete*: the graphical

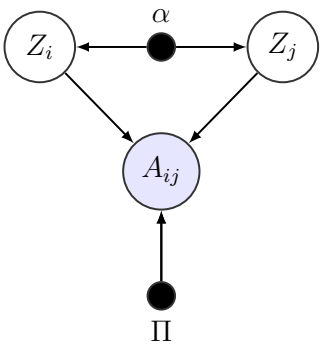


Figure 4.2: Graphical representation of SBM at the (i, j) nodes pair level.

model corresponding to the whole generative model, involving N nodes is difficult to represent due to the cross dependencies occurring in all directions. In more detail one should plot all the $N(N-1)/2$ observed random variables A_{ij}, all the N latent random variables $Z_1, \ldots, Z_N$ and each A_{ij} would receive two arrows issued from Z_i and Z_i. Parameters and relative links should finally be added. In contrast with all the generative models we saw in previous chapters, no synthetic representation of this exists.

A final remark before moving to the inference of the model parameters in SBM. The generative model detailed so far can be (and was!) generalized in several directions. For instance, one may deal with weighted adjacency matrices whose entries might be counts, continuous data or more complex entities (e.g. text). For counting data, for instance, we can replace Eq. (4.2) with

$$\mathbb{P}(A_{ij} = k|\mathbf{Z}, \Lambda) = \frac{\lambda_{Z_i Z_j}^k}{k!} \exp(-\lambda_{Z_i Z_j}) \mathbf{1}_{\mathbb{N}}(k)$$

with $\mathbf{1}_I(\cdot)$ being the indicator function on the set I. Similarly, the Gaussian distribution can be used for continuous data and so on. Other extensions aimed to extend SBM to a *soft* clustering scenario in which clusters overlaps and many other extensions exist. To conclude, stochastic block models should be thought of as an entire family of mixture models for random graphs and not as a single generative model.

Exercise. Show that in the binary SBM introduced in this section, a node's degree is (approximately) distributed according to a mixture of Poisson distributions.

4.3 Variational inference

We are in a situation similar to the one seen in the previous chapter, with GMMs. Due to the presence of hidden variables $\mathbf{Z}$, the quantity we would like to work with (Eq (4.4)) is useless. Instead, the log-likelihood of the observed data

$$\log p(A|\Pi, \alpha) = \log \sum_{\mathbf{z}} p(A, \mathbf{z}|\Pi, \alpha), \tag{4.6}$$

that we have, is not tractable analytically. Similarly to what we did with GMMs, we can rely on a variational decomposition of the observed log-likelihood, which now reads

$$\log p(A|\Pi, \alpha) = \mathbb{E}_{\mathbf{Z}\sim q}\left[\log \frac{p(A, \mathbf{Z}|\Pi, \alpha)}{q(Z)}\right] + KL(q||p_{\mathbf{Z}}), \tag{4.7}$$

where $p_{\mathbf{z}}$ denotes the posterior distribution $p(\mathbf{z}|A, \Pi, \alpha)$ of the cluster labels given the data and the model parameters. The main difference with respect to the GMMs is that here, due to the cross dependences mentioned in the previous section, the posterior probability is not tractable. In order to overcome this issue, several solutions have been proposed in the literature. Many of them rely on Bayesian techniques, allowing one to sample from the posterior distribution. Here we present a standard solution known as pure variational EM algorithm consisting into setting $q(\cdot)$ to a tractable family of distributions (recall that the above decomposition holds for any distribution $q(\cdot)$ sharing the support with the prior distribution on $\mathbf{Z}$) and choosing the representative of this family who maximizes the lower bound

$$\mathcal{L}(q, \Pi, \alpha) := \mathbb{E}_{\mathbf{Z}\sim q}\left[\log \frac{p(A, \mathbf{Z}|\Pi, \alpha)}{q(\mathbf{Z})}\right].$$

Once more, since $q(\cdot)$ does not appear on the left hand side of Eq. (3.9), maximizing the lower bound with respect to $q(\cdot)$ is equivalent to minimize the KL divergence between $q(\cdot)$ and the posterior $p_{\mathbf{z}}$.

A tractable family can be (and very often it is) chosen relying on the *mean field approximation*:

$$q(\mathbf{z}) = \prod_{i=1}^{N} \tau_{iz_i} = \prod_{i=1}^{N}\prod_{k=1}^{K} \tau_{ik}^{z_{ik}} \tag{4.8}$$

with $\tau_{i1}, \ldots, \tau_{iK}$ being unknown probabilities such that $\sum_{k=1}^{K} \tau_{ik} = 1$ for all nodes i. We are now ready to describe the variational EM algorithm.

E-step. Via Eqs. (4.4),(4.8) and the definition of $\mathcal{L}$ we (almost) immediately have

$$\begin{aligned}\mathcal{L}(\tau, \Pi, \alpha) = \frac{1}{2} \sum_{j \neq i}^{N} \sum_{k,l}^{K} (\tau_{ik}\tau_{jl} A_{jl} \log \pi_{kl} + \tau_{ik}\tau_{jl}(1 - A_{ij}) \log(1 - \pi_{kl})) \\ + \sum_{i=1}^{N} \sum_{k=1}^{K} \tau_{ik} \log \frac{\alpha_k}{\tau_{ik}}.\end{aligned} \tag{4.9}$$

Take some time to notice the similarities and differences with Eq. (4.4). Now, whereas in the previous chapter τ_{ik} was defined as the posterior distribution of Z_i given the data, here it is an unknown quantity (since the posterior is not tractable) that must be chosen in order to maximize the above lower bound. This can be done by taking the partial derivative with respect to τ_{ik} of $\mathcal{L}(q, \Pi, \alpha) + \sum_{i=1}^{N} \left(\lambda_i \left(\sum_{k=1}^{K} \tau_{ik} - 1 \right) \right)$, where $\lambda_1, \ldots, \lambda_N$ are Lagrange multipliers accounting for the simplex constraints over the rows of τ. That derivative is

$$\log \alpha_k + \sum_{j \neq i}^{N} \sum_{l=1}^{K} \tau_{jl} \left[A_{ij} \log \pi_{kl} + (1 - A_{ij}) \log(1 - \pi_{kl}) \right] - \log \tau_{ik} + 1 + \lambda_i$$

with $\exp(1 + \lambda_i)$ being the normalizing constant. After some calculations one obtains

$$\hat{\tau}_{ik} \propto \alpha_k \exp \left(\sum_{j \neq i} \sum_{l} \tau_{jl} \left[A_{ik} \log \pi_{kl} + (1 - A_{ij}) \log(1 - \pi_{kl}) \right] \right), \tag{4.10}$$

which is the optimal update of τ_{ik}. It is a fixed point equation that can be computed independently for all i.

M-step. After replacing $\hat{\tau}_{ik}$ (for all i and k) in Eq. (4.9) the maximization with respect to the model parameters α and Π is straightforward and follows what you did in order to find Eq.(4.5), thus leading to

$$\begin{aligned}\hat{\pi}_{kl} &:= \frac{\sum_{j \neq i}^{N} \tau_{ik}\tau_{jl} A_{ij}}{\sum_{j \neq i}^{N} \tau_{ik}\tau_{jl}}, \\ \hat{\alpha}_k &= \frac{1}{N} \sum_{i=1}^{N} \tau_{ik}.\end{aligned} \tag{4.11}$$

As we saw in the EM algorithm for the mixture of Gaussian distributions, the variational M updates of the model parameters are the same as the MLE estimates from the complete-data log-likelihood except for τ replacing $\mathbf{z}$.

Once more, the main difference with respect to what we saw for GMMs is that here the KL divergence of the right-hand side of Eq. (4.7) never vanishes since $q(\cdot)$ only is an approximate posterior. Therefore, iterating the (variational) E-M steps described above certainly leads to an increase of the lower bound $\mathcal{L}(q, \Pi, \alpha)$ until a stationary point is reached. However, it is much more difficult to asses the convergence properties of the variational EM estimators toward the ML estimates from the observed data log-likelihood. A pseudo-code summarizing the VEM algorihtm for SBM is reported in Algorithm 3. Just a word of caution to stress that in both GMMs and SBMs

Algorithm 3 Pseudocode: VEM - SBM

function FIT(A,K)
  $\hat{\tau} \leftarrow$ INIT(A,K, type) ▷ type is "K-means" or "spectral clutering"
  First updates of $(\hat{\Pi}, \hat{\alpha})$ via Eq (4.11)
  while $\mathcal{L}(q, \hat{\Pi}, \hat{\alpha})$ increases **do**
    Update $\hat{\tau}$ via Eq. (4.10)
    Compute the new $\mathcal{L}(q, \hat{\Pi}, \hat{\alpha})$ ▷ E-step
    Update $(\hat{\Pi}, \hat{\alpha})$ via Eq. (4.11) ▷ M-step
  end while
  return $(\hat{\tau}, \hat{\alpha}, \hat{\Pi})$
end function

the objective functions we seek to maximize are *not* concave, so multiple stationary points corresponding to local maxima can be and indeed are reached by the (V)EM algorithm. In the hope to reach a global maximum, and knowing that (V)EM algorithms are very sensitive to the initialization, it is a good practice to initialize $\hat{\tau}$ in a "clever" way, for instance based on other clustering techniques such as k-means, spectral clustering (see Appendix 4.A) etc.

4.4 Selecting the number of clusters

The whole discussion that we had in Section 3.4, regarding the need to maximize the posterior probability of K (and possibly $\mathbf{Z}$) given the data in order to select K remains valid here. However, due to the "softer" relation between the lower bound and the observed data log-likelihood in SBM with respect to GMMs, BIC in not usually used in order to select K in SBMs. Instead, ICL remains a good alternative and in this case it looks like

$$ICL_K := \log p(A, \hat{\mathbf{z}}|\hat{\Pi}, \hat{\alpha}, K) - \frac{K^2}{2} \log\left(\frac{N(N-1)}{2}\right) - \frac{(K-1)}{2} \log N.$$

In the first term on the right hand side of the equality, Π and α denote the final value of the model parameters after the VEM converged. and $\mathbf{z}$ is the MAP estimates of the cluster memberships based on the final value of $\hat{\tau}$, namely $\hat{z}_i := \arg\max_{k \leq K} \tau_{ik}$. The second term on the right hand side of the equality is a BIC-like penalty related with the the log-likelihood term $\log p(A|\hat{\mathbf{z}}, \hat{\Pi}, K)$, with K^2 being the number of free parameters inside Π and the argument of the logarithm being the number of observations in A (symmetric, no self loops). Finally the third term accounts for the log-likelihood $\log p(\mathbf{z}|\hat{\alpha}, K)$, with $K-1$ being the number of free parameters in α and N the number of observations.

The above criterion can be independently computed for different values of K, and the value of K leading to the highest values of ICL_K is finally retained.

4.A Graph Laplacian and spectral clustering

In Algorithm 3 we mentioned that $\hat{\tau}$ can be initialized via K-means or spectral clustering. The former would consist into optimizing the K-means loss function in Eq. (3.13) with $x_1, \dots, x_N$ being replaced by $A_1, \dots, A_N$, namely the *rows* of the adjacency matrix. Then, one sets the i-th row of $\hat{\tau}$, namely $\hat{\tau}_i$ equal to the optimal $r_i \in \{0,1\}^K$ for all i to get the initialisation. However, an elegant alternative consists into fitting K-means to a "filtered" version of A, after some noise being removed and this is precisely what spectral clustering does. First we need to consider the *Graph Laplacian* defined by the following matrix

$$L := D - A, \tag{4.12}$$

where D is a diagonal matrix having the nodes degrees $d_1, \dots, d_N$ on the main diagonal (i.e. $D_{ii} = d_i = \sum_{j=1}^N A_{ij}$). As far as we deal with undirected graphs, L is symmetric (as well as A) and admits the following spectral decomposition

$$L = Q \Lambda Q^T$$

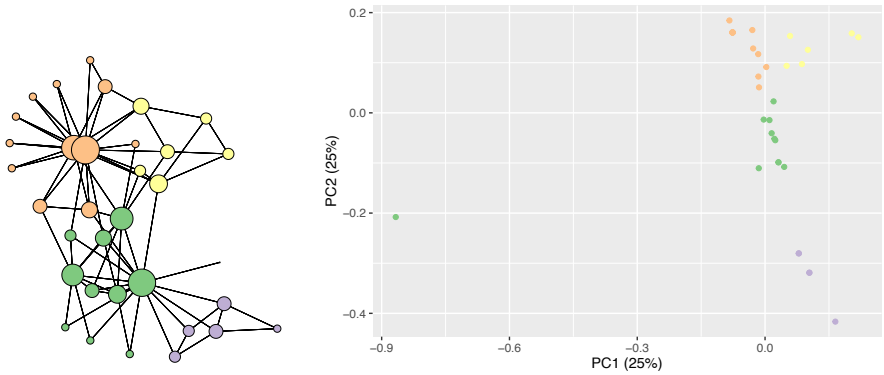
with Λ being a diagonal matrix with the real eigenvalues on the main diagonal and Q is the orthogonal matrix of the eigenvectors. Now, for all non-null $y \in \mathbb{R}^N$, one has

$$\begin{aligned} y^T L y = y^T D y - y^T A y &= \frac{1}{2}\left(y^T D y - 2 y^T A y + y^T D y\right) \\ &= \frac{1}{2}\left(\sum_{i=1}^N d_i y_i^2 - 2\sum_{i,j=1}^N A_{ij} y_i y_j + \sum_{j=1}^N d_j y_j^2\right) \\ &= \frac{1}{2}\sum_{i,j=1}^N A_{ij}(y_i - y_j)^2 \geq 0, \end{aligned}$$

where the last equality comes from the definition of d_i and the inequality follows from $A_{ij} \geq 0$. Therefore L is symmetric and positive semi-definite and its eigenvalues are non-negative. Thus, by definition of L one immediately sees that the smallest eigenvalue is zero and the corresponding eigenvector $(1/N, \dots, 1/N)$. Indeed it can be shown that the multiplicity of the eigenvector 0 is equal to the number of connected components in the graph.

Now the spectral clustering algorithm works as follows: assuming that the eigenvalues on the main diagonal of Λ are in the increasing order, for a given

number of clusters K, the first K columns of Q corresponding to *strictly* positive eigenvalues are collected in a matrix $U \in \mathbb{R}^{N\times K}$ and the K-means loss function in Eq (3.13) is optimized with $x_1, \ldots, x_N$ being replaced by the rows of U: $U_1, \ldots, U_N$. Those row vectors are known as *positional embeddings* and they are representative of the topological structure of the graph as it can be seen in Figure 4.3. On the left hand side we see the Zachary's



(a) Communities in a graph.

(b) Positional embeddings.

Figure 4.3: (a) Zachary's karate club network: nodes belonging to the same community (modularity maximization) have the same color. (b) Positional embeddings (PCA) corresponding to the first $K = 4$ eigenvectors of the graph Laplacian. The colour scheme is "imported" from Figure (a).

karate club network[2] with different colors corresponding to the communities found via modularity maximization[3]. The same colours are applied to the positional embeddings on the right hand side, where principal component analysis was use to visualize the embeddings originally in dimension $K = 4$. No clustering/classification was done on the embeddings. Still, as it can be seen, points in the same community are nearby in the latent space.

A final remark. The binary adjacency matrix we considered here defines a notion of affinity (or similarity) between the nodes of a graph. However, spectral clustering has much wider application then social network analysis. Indeed, *weighted* adjacency matrices can be used to introduced pairwise similarity scores in a more general dataset. For instance, consider a data set

[2] https://en.wikipedia.org/wiki/Zachary%27s_karate_club.
[3] https://en.wikipedia.org/wiki/Modularity_(networks)

$x_1, \ldots, x_N$, with $x_i \in \mathbb{R}^D$ and and a matrix W such that

$$W_{ij} = \exp\left(-\frac{||x_i - x_j||_2^2}{\gamma}\right),$$

where $\gamma > 0$ is a user defined parameter. Well, W is a symmetric positive-definite matrix whose graph Laplacian can be used to (spectrally) cluster the data points $x_1, \ldots, x_N$ in same way as described before. In order to learn more about spectral clustering, the reader is referred to the excellent tutorial Von Luxburg (2007).