Prediction for Processes on Network Graphs

Gonzalo Mateos
Dept. of ECE and Goergen Institute for Data Science
University of Rochester
gmateosb@ece.rochester.edu
http://www.ece.rochester.edu/~gmateosb/

April 14, 2016
Nearest neighbors

Nearest-neighbor prediction

Markov random fields

Kernel regression on graphs

Case study: Predicting protein function
**Processes on network graphs**

- **Motivation**: study complex systems of elements and their interactions
  - So far studied network graphs as representations of these systems
Motivation: study complex systems of elements and their interactions
   So far studied network graphs as representations of these systems

Often some quantity associated with each of the elements is of interest
⇒ Quantities may be influenced by the interactions among elements

Example

1) Behaviors and beliefs influenced by social interactions
2) Functional roles of proteins influenced by their sequence similarity
3) Spread of epidemics influenced by proximity of individuals
Processes on network graphs

- **Motivation:** study complex systems of elements and their interactions
  - So far studied network graphs as representations of these systems

- Often some quantity associated with each of the elements is of interest
  \[ \Rightarrow \text{Quantities may be influenced by the interactions among elements} \]

**Example**

1) Behaviors and beliefs influenced by social interactions
2) Functional roles of proteins influenced by their sequence similarity
3) Spread of epidemics influenced by proximity of individuals

- Can think of these quantities as random processes defined on graphs
  - Static \( \{X_i\}_{i \in V} \) and dynamic processes \( \{X_i(t)\}_{i \in V} \) for \( t \in \mathbb{N} \) or \( \mathbb{R}^+ \)
Consider prediction of a static process $\mathbf{X} := \{X_i\}_{i \in V}$ on a graph.

Process may be truly static, or a snapshot of a dynamic process.
Nearest-neighbor prediction

- Consider prediction of a static process $\mathbf{X} := \{X_i\}_{i \in \mathcal{V}}$ on a graph
  - Process may be truly static, or a snapshot of a dynamic process

- Idea: exploit the network graph structure for prediction
  - Consider a random graph $G(\mathcal{V}, E)$, observe adjacency matrix $\mathbf{Y} = \mathbf{y}$

- Given observations of all attributes $\mathbf{X}^{(-i)} = \mathbf{x}^{(-i)}$ but $X_i$. Predict $X_i$?
Consider prediction of a static process $\mathbf{X} := \{X_i\}_{i \in V}$ on a graph
- Process may be truly static, or a snapshot of a dynamic process

**Idea:** exploit the network graph structure for prediction
- Consider a random graph $G(V, E)$, observe adjacency matrix $\mathbf{Y} = y$
- Given observations of all attributes $\mathbf{X}^{(-i)} = \mathbf{x}^{(-i)}$ but $X_i$. Predict $X_i$?
- For binary $X_i \in \{0, 1\}$, say, simple nearest-neighbor method predicts

$$
\hat{X}_i = \mathbb{I}\left\{ \frac{\sum_{j \in \mathcal{N}_i} X_j}{\mathcal{N}_i} > \tau \right\}
$$

⇒ Average of the observed process in the neighborhood of $i$
⇒ Called ‘guilt-by-association’ or graph-smoothing method
Example: predicting law practice

- Network $G^{obs}$ of working relationships among lawyers [Lazega’01]
  - Nodes are $N_v = 36$ partners, edges indicate partners worked together

Data includes various node-level attributes $\{X_i\}_{i \in V}$ including:
  - Type of practice, i.e., litigation (red) and corporate (cyan)

- Suspect lawyers collaborate more with peers in same legal practice
  - Knowledge of collaboration useful in predicting type of practice
In predicting practice $X_i$, how useful is the value of one neighbor?

⇒ Breakdown of 115 edges based on practice of incident lawyers

<table>
<thead>
<tr>
<th></th>
<th>Litigation</th>
<th>Corporate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Litigation</td>
<td>29</td>
<td>43</td>
</tr>
<tr>
<td>Corporate</td>
<td>43</td>
<td>43</td>
</tr>
</tbody>
</table>

Looking at the rows in this table:

- Litigation lawyers collaborators are 40% litigation, 60% corporate
- Collaborations of corporate lawyers are evenly split

⇒ Suggests using a single neighbor has little predictive power
In predicting practice $X_i$, how useful is the value of one neighbor?  
⇒ Breakdown of 115 edges based on practice of incident lawyers

<table>
<thead>
<tr>
<th></th>
<th>Litigation</th>
<th>Corporate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Litigation</td>
<td>29</td>
<td>43</td>
</tr>
<tr>
<td>Corporate</td>
<td>43</td>
<td>43</td>
</tr>
</tbody>
</table>

Looking at the rows in this table
- Litigation lawyers collaborators are 40% litigation, 60% corporate
- Collaborations of corporate lawyers are evenly split
⇒ Suggests using a single neighbor has little predictive power

But 60% (29+43=72) of edges join lawyers with common practice
⇒ Suggests on aggregate knowledge of collaboration informative
Example: predicting law practice (cont.)

- Incorporate information of all collaborators as in nearest-neighbors
  - Let $X_i = 0$ if lawyer $i$ practices litigation, and $X_1 = 1$ for corporate

![Histograms of the fraction of corporate collaborators among lawyers in the network of](image)

Nearest-neighbor prediction rule

$$\hat{X}_i = \mathbb{I} \left\{ \frac{\sum_{j \in \mathcal{N}_i} X_j}{\mathcal{N}_i} > 0.5 \right\}$$

⇒ Infers correctly 13 of the 16 corporate lawyers (i.e., 81%)
⇒ Infers correctly 16 of the 18 litigation lawyers (i.e., 89%)
⇒ Overall error rate is just under 15%
Nearest-neighbor methods may seem rather informal and simple
⇒ But competitive with more formal, model-based approaches

Still, model-based methods have certain potential advantages:
  a) Probabilistically rigorous predictive statements;
  b) Formal inference for model parameters; and
  c) Natural mechanisms for handling missing data
Nearest-neighbor methods may seem rather informal and simple

⇒ But competitive with more formal, model-based approaches

Still, model-based methods have certain potential advantages:

a) Probabilistically rigorous predictive statements;

b) Formal inference for model parameters; and

c) Natural mechanisms for handling missing data

Model the process $\mathbf{X} := \{X_i\}_{i \in \mathcal{V}}$ given an observed graph $\mathbf{Y} = \mathbf{y}$

Two commonly used classes of paradigms:

⇒ Markov random field (MRF) models

⇒ Kernel-regression models using graph kernels
Markov random fields

Nearest-neighbor prediction

Markov random fields

Kernel regression on graphs

Case study: Predicting protein function
Consider a graph $G(V, E)$ with given adjacency matrix $A$

Let $X = [X_1, \ldots, X_{N_v}]^\top$ be a collection of discrete RVs defined on $V$

Process $X$ is a Markov random field (MRF) on $G$ if

$$
P \left( X_i = x_i \mid X^{(-i)} = x^{(-i)} \right) = P \left( X_i = x_i \mid X_{\mathcal{N}_i} = x_{\mathcal{N}_i} \right), \quad i \in V$$

$X_i$ conditionally independent of all other $X_k$, given neighbors values

A ‘spatial’ Markov property, generalizing Markov chains in time

Graph $G$ defines neighborhoods $\mathcal{N}_i$, hence dependencies

Roots in statistical mechanics, Ising model of ferromagnetism [Ising ’25]

MRFs used extensively in spatial statistics and image analysis

Definition requires a technical condition $P(X = x) > 0$, for all $x$
MRFs and Gibbs random fields

- MRFs equivalent to Gibbs random fields $\mathbf{X}$, having joint distribution

$$P(\mathbf{X} = \mathbf{x}) = \left( \frac{1}{\kappa} \right) \exp\{U(\mathbf{x})\}$$

- Energy function $U(\cdot)$, and partition function $\kappa = \sum_{\mathbf{x}} \exp\{U(\mathbf{x})\}$
- Equivalence follows from the Hammersley-Clifford theorem

- Energy function decomposable over the maximal cliques in $G$

$$U(\mathbf{x}) = \sum_{c \in \mathcal{C}} U_c(\mathbf{x})$$

- Defined clique potentials $U_c(\cdot)$ and set $\mathcal{C}$ of maximal cliques in $G$

- Can show $P(\mathbf{X}_i | \mathbf{X}^{(-i)})$ depends only on cliques involving vertex $i$
Example: auto-logistic MRFs

- May specify MRFs through choice of clique potentials $U_c(\cdot)$

- **Ex:** The class of auto models are defined through the constraints:
  
  (i) Only cliques $c \in C$ of size one and two have $U_c \neq 0$

  (ii) Probabilities $P(X_i \mid X_{\mathcal{N}_i})$ have an exponential family form

- For binary RVs $X_i \in \{0, 1\}$, the energy function takes the form

  $$U(x) = \sum_{i \in V} \alpha_i x_i + \sum_{(i, j) \in E} \beta_{ij} x_i x_j$$

- The resulting MRF is known as auto-logistic model, because

  $$P(X_i = 1 \mid X_{\mathcal{N}_i} = x_{\mathcal{N}_i}) = \frac{\exp\{\alpha_i + \sum_{j \in \mathcal{N}_i} \beta_{ij} x_j\}}{1 + \exp\{\alpha_i + \sum_{j \in \mathcal{N}_i} \beta_{ij} x_j\}}$$

  - Suggests logistic regression of $x_i$ on its neighboring $x_j$’s

  - Ising model a special case, with $G$ defined to be a regular lattice
Homogeneity assumptions

- Typical to assume that parameters $\alpha_i$ and $\beta_{ij}$ are homogeneous

- **Ex:** Specifying $\alpha_i = \alpha$ and $\beta_{ij} = \beta$ yields conditional log-odds

$$\log \left[ \frac{P(X_i = 1 | X_{\mathcal{N}_i} = x_{\mathcal{N}_i})}{P(X_i = 0 | X_{\mathcal{N}_i} = x_{\mathcal{N}_i})} \right] = \alpha + \beta \sum_{j \in \mathcal{N}_i} x_j$$

- Linear dependency in the number of neighbors $j$ of $i$ with $X_j = 1$

- **Ex:** Specifying $\alpha_i = \alpha + |\mathcal{N}_i|\beta_2$ and $\beta_{ij} = \beta_1 - \beta_2$ yields

$$\log \left[ \frac{P(X_i = 1 | X_{\mathcal{N}_i} = x_{\mathcal{N}_i})}{P(X_i = 0 | X_{\mathcal{N}_i} = x_{\mathcal{N}_i})} \right] = \alpha + \beta_1 \sum_{j \in \mathcal{N}_i} x_j + \beta_2 \sum_{j \in \mathcal{N}_i} (1 - x_j)$$

- Linear also in the number of neighbors $j$ of $i$ with $X_j = 0$
MRFs for continuous random variables

- **MRFs with continuous RVs:** replace PMFs/sums with pdfs/integrals
- Often use Gaussian distributions due to their analytical tractability
- **Ex:** the auto-Gaussian model specifies Gaussian $X_i \mid X_{\mathcal{N}_i} = x_{\mathcal{N}_i}$, with

  \[
  \mathbb{E} [X_i \mid X_{\mathcal{N}_i} = x_{\mathcal{N}_i}] = \alpha_i + \sum_{j \in \mathcal{N}_i} \beta_{ij} (x_j - \alpha_j)
  \]

  \[
  \text{var} [X_i \mid X_{\mathcal{N}_i} = x_{\mathcal{N}_i}] = \sigma^2
  \]

- Values $X_i$ modeled as weighted combinations of their neighbors
- Let $\bm{\mu} = [\alpha_1, \ldots, \alpha_{N_v}]^T$ and $\bm{\Sigma} = \sigma^2(\bm{I} - \bm{B})^{-1}$, where $\bm{B} = [\beta_{ij}]$
- Under $\beta_{ii} = 0$ and $\beta_{ij} = \beta_{ji}$, the joint distribution is $\bm{X} \sim \mathcal{N}(\bm{\mu}, \bm{\Sigma})$
- Homogeneity assumptions can be imposed, simplifying expressions
In studying process $X = \{X_i\}_{i \in V}$ of interest to predict some or all of $X$

MRF models we have seen for this purpose can be expressed in the form

$$P_\theta(X = x) = \left(\frac{1}{\kappa(\theta)}\right) \exp\{-U(x; \theta)\}$$

Parameter $\theta$ is typically low-dimensional, e.g., $\theta = [\alpha, \beta]$ in auto-models

Predictions can be generated based on the distribution $P_\theta(\cdot)$

Knowledge of $\theta$ is necessary, and typically $\theta$ is unknown

Unlike nearest-neighbors prediction, MRFs requires inference of $\theta$ first
Estimation of $\theta$ most naturally approached via maximum-likelihood

Even though the log-likelihood function takes a simple form

$$\ell(\theta) = \log P_{\theta}(X = x) = U(x; \theta) - \log \kappa(\theta)$$

Evaluation of $\kappa(\theta) = \sum_x \exp\{U(x; \theta)\}$ ends up being intractable

A popular alternative is maximum pseudo-likelihood, i.e., maximize

$$\sum_{i \in V} \log P_{\theta} \left( X_i = x_i \mid X^{(-i)} = x^{(-i)} \right)$$

Ignores dependencies beyond the neighborhood of each $X_i$

Probabilities depend on specified clique potentials $U_c$, not on $\kappa(\theta)$
Gibbs sampler

- Given a value of $\theta$, consider predicting some or all of $X$ from $P_{\theta}(\cdot)$
- Hard to evaluate $P_{\theta}(\cdot)$, but can simulate from it using a Gibbs sampler
- Gibbs sampler exploits $P_{\theta}(X_i \mid X^{(-i)} = x^{(-i)})$ in simple closed form
- New value $X_{(k)}$ obtained from $X_{(k-1)} = x_{(k-1)}$ by drawing

\[
X_{1,(k)} \text{ from } P_{\theta} \left( X_1 \mid X^{(-1)} = x^{(-1)}_{(k-1)} \right)
\]
\[
\vdots
\]
\[
X_{N_v,(k)} \text{ from } P_{\theta} \left( X_{N_v} \mid X^{(-N_v)} = x^{(-N_v)}_{(k-1)} \right)
\]

- The generated sequence $X_{(1)}, X_{(2)}, \ldots$ clearly forms a Markov chain
- Under appropriate conditions, stationary distribution is equal to $P_{\theta}(\cdot)$
Prediction with MRFs

- Given a big sample from $P_\theta(\cdot)$, can predict $X$ using empirical distributions
- **Ex:** for binary $X$ use empirical marginal frequencies to predict $X_i$, i.e.,
  \[
  \hat{X}_i = \mathbb{I}\left\{ \frac{1}{n} \sum_{k=m+1}^{n} X_{i,(k)} > 0.5 \right\} \text{ for large } m, n
  \]

- Suppose we observe some elements $X^{obs} = x^{obs}$, and wish to predict $X^{miss}$
- Can generate draws from the relevant $P_\theta(X^{miss} \mid X^{obs} = x^{obs})$ as
  \[
  X_{i,(k)} \text{ from } P_\theta(X_i \mid X^{obs} = x^{obs}, X^{(-i),miss} = x^{(-i),miss}_{(k-1)})
  \]
- Prediction from empirical distributions may be carried out analogously
- Prior inference of $\theta$ based on limited data $X^{obs} = x^{obs}$ non-trivial
Kernel-based regression

Nearest-neighbor prediction

Markov random fields

Kernel regression on graphs

Case study: Predicting protein function
Kernel methods

- MRFs specify precise dependency structures in $\mathbf{X}$, given the graph $G$
- Q1: Can we just learn a function relating the vertices to their attributes?
- A regression-based approach on the graph $G$ seems appealing
- Standard methods such as LS regression relate data in Euclidean space
- Q2: Can these methods be tuned to accommodate graph-indexed data?
- Yes! Kernel methods found useful to this end, and consist of:
  - Generalized predictor variables (i.e., encoded using a kernel)
  - Regression of a response to these predictors using ridge regression
- Key innovation here is the construction of graph kernels
Kernel regression on graphs

- Let $G(V, E)$ be a graph and $X = \{X_i\}_{i \in V}$ a vertex attribute process.
- Suppose we observe $X_i = x_i$ for $i \in V^{obs} \subset V$, with $n = |V^{obs}|$.
- Goal: learn $\hat{h} : V \mapsto \mathbb{R}$ describing how attributes vary across vertices.

Graph-indexed data not Euclidean $\Rightarrow$ kernel regression methods.

**Def:** A function $K : V \times V \mapsto \mathbb{R}$ is called a kernel if for each $m = 1, \ldots, N_v$ and subset of vertices $\{i_1, \ldots, i_m\} \in V$, matrix

$$K^{(m)} = [K(i_j, i_{j'})] \in \mathbb{R}^{m \times m}$$

is symmetric and positive semi-definite.

- Can think of kernels as functions that produce similarity matrices.
- Kernel regression builds predictors from such similarities (more soon).
- Need to also decide on the space $\mathcal{H}$ where to search for $\hat{h}$. 
Since $V$ is finite, can represent functions $h$ on $V$ as vectors $h \in \mathbb{R}^{N_v}$

Form $K^{(N_v)} \in \mathbb{R}^{N_v \times N_v}$ by evaluating $K$ in all pairs $(i, j) \in V^{(2)}$

Suppose $K^{(N_v)}$ admits an eigendecomposition of the form

$$K^{(N_v)} = \Phi^\top \Delta \Phi$$

Given kernel $K$ and data $x^{obs}$, kernel regression seeks $\hat{h}$ from the class

$$\mathcal{H}_K = \{ h \in \mathbb{R}^{N_v} : h = \Phi \beta \text{ and } \beta^\top \Delta^{-1} \beta < \infty \}$$

$\mathcal{H}_K$ is the reproducing-kernel Hilbert space (RKHS) induced by $K$

Elements $h$ in $\mathcal{H}_K$ are linear combinations of eigenvectors of $K^{(N_v)}$

They are also constrained to have finite norm, in the sense

$$\|h\|_\mathcal{H} = \|\Phi \beta\|_\mathcal{H} := \beta^\top \Delta^{-1} \beta < \infty$$
Penalized regression in RKHS

- Choose appropriate $\hat{h} \in \mathcal{H}_K$ using penalized kernel regression
- Appropriate? Fits the data well and small norm (i.e., low complexity)

$$\hat{h} = \Phi \hat{\beta}, \quad \text{where} \quad \hat{\beta} = \arg \min_{\beta} \left[ \sum_{i \in V^{obs}} C(x_i, [\Phi \beta]_i) + \lambda \beta^\top \Delta^{-1} \beta \right]$$

- Convex loss $C(\cdot, \cdot)$ encourages goodness of fit to $x^{obs}$
- The term $\|h\|_H$ penalizes excessive complexity
- Tuning parameter $\lambda$ trades off data fidelity and complexity
- Generalized ridge-regression with columns of $\Phi$ as predictors
- Eigenvectors with small eigenvalues penalized more harshly
Representer theorem

- Need to compute the entire $\Phi$ to find the regression function $\hat{h}$
- Complex to evaluate $K$ for all vertex pairs $V^{(2)}$, and find eigenvectors
- Consider instead evaluating $K$ in $V \times V^{obs}$, yielding $K^{(N_v,n)} \in \mathbb{R}^{N_v \times n}$
- The Representer theorem asserts that $\hat{h}$ equivalently given by

$$\hat{h} = K^{(N_v,n)} \hat{\alpha}, \text{ where } \hat{\alpha} = \arg \min_{\alpha} \left[ \sum_{i \in V^{obs}} C(x_i, [K^{(n)}\alpha]_i) + \lambda \alpha^\top K^{(n)} \alpha \right]$$

- Just need to evaluate $K$ in $V^{obs} \times V^{obs}$ to form $K^{(n)}$
- Complexity scales with the number of observations $n$, not $N_v$
- By virtue of $\hat{h} = K^{(N_v,n)} \hat{\alpha}$, can predict value in $i \in V^{miss}$ via

$$\hat{h}_i = \sum_{j \in V^{obs}} \hat{\alpha}_j K(i,j)$$
Example: Kernel ridge regression

- Let the $X_i$ be continuous and the loss quadratic, i.e., $C(x, a) = (x - a)^2$
- The optimization problem defining $\hat{\alpha}$ thus specializes to

\[
\min_{\alpha} \left[ \|x^{obs} - K^{(n)}\alpha\|^2_2 + \lambda \alpha^\top K^{(n)}\alpha \right]
\]

- This particular method is known as kernel ridge regression. Intuition?
- Define $\theta := (K^{(n)})^{1/2}\alpha$ and $M := (K^{(n)})^{1/2}$. An equivalent problem is

\[
\min_{\theta} \left[ \|x^{obs} - M\theta\|^2_2 + \lambda \theta^\top \theta \right]
\]

- Standard ridge regression with solution $\hat{\theta} = (M^\top M + \lambda I)^{-1}M^\top x^{obs}$
  \[ \Rightarrow \text{The kernel regression function is } \hat{h} = K^{(N_v,n)}(K^{(n)})^{-1/2}\hat{\theta} \]
Example: Kernel logistic regression

- Let binary $X_i \in \{-1, 1\}$ indicate class membership, for two classes
- A natural choice in this context is the logistic loss, given by
  \[
  C(x, a) = \ln \left( 1 + e^{-xa} \right)
  \]
- Corresponds to the negative log-likelihood of a Bernoulli RV

Kernel logistic regression selects $\hat{\alpha}$ via the optimization problem

\[
\min_{\alpha} \left[ \sum_{i \in V^{obs}} \ln \left( 1 + e^{-x_i[K^{(n)}\alpha]_i} \right) + \lambda \alpha \top K^{(n)} \alpha \right]
\]

- No closed-form solution for $\hat{\alpha}$, need iterative algorithms
- Given $\hat{h} = K^{(N_v, n)} \hat{\alpha}$, prediction of $X_i$ for $i \in V^{miss}$ based on
  \[
  \hat{P} (X_i = 1 \mid X^{obs} = x^{obs}) = \frac{e^{\hat{h}_i}}{1 + e^{\hat{h}_i}}
  \]
In designing a kernel $K$ on a graph $G$, desired properties are:

- $K^{(N_v)}$ is symmetric and positive semi-definite
- $K$ captures suspected similarity among vertices in $V$

**Presumption:** proximity of vertices in $G$ already indicative of similarity

Thus most kernels proposed are related to the topology of $G$

**Ex:** the Laplacian kernel is $K^{(N_v)} := L^\dagger$, where $^\dagger$ denotes pseudo-inverse

The penalty term $\|h\|_\mathcal{H} = \beta^T \Delta^{-1} \beta$ takes the form

\[
\beta^T \Delta^{-1} \beta = \beta^T \Phi^T \Phi \Delta^{-1} \Phi^T \Phi \beta
\]

\[
= h^T K^\dagger h = h^T L h
\]

\[
= \sum_{(i,j) \in E} (h_i - h_j)^2
\]

**Kernel regression seeks smooth $\hat{h}$ with respect to the topology of $G$**
Diffusion kernels

- Laplacian kernel $K = L^\dagger$ encodes similarity among vertices through $A$
- Can encode similarity through path structures, powers of $A$ and $L$
- Popular choice incorporating all powers of $L$ is the diffusion kernel

$$K = e^{-\zeta L} := \sum_{m=0}^{\infty} \frac{(-\zeta)^m}{m!} L^m$$

- Decay factor $0 < \zeta < 1$ controls similarity assigned to longer paths
- Defined in terms of the matrix exponential $e^{-\zeta L}$
- Treating $K$ as a function of $\zeta$ yields the differential equation

$$\frac{\partial K}{\partial \zeta} = -LK$$
- Parallels the heat equation in physics, motivating its name
Regularized Laplacian kernels

- Let \( \mathbf{L} = \Phi \Gamma \Phi^\top \), with \( \Gamma = [\gamma_1, \ldots, \gamma_{N_v}]^\top \) and \( \Phi = [\phi_1, \ldots, \phi_{N_v}] \)
- Laplacian and diffusion kernels within class of regularization kernels

\[
\mathbf{K} = \sum_{i=1}^{N_v} r^{-1}(\gamma_i) \phi \phi^\top
\]

- \( \mathbf{K} \) is the inverse of the regularized Laplacian \( r(\mathbf{L}) := \Phi r(\mathbf{L}) \Phi^\top \)
- The regularization function \( r(\cdot) \geq 0 \) is increasing, including:
  - Ex: Identity function \( r(\gamma) = \gamma \)
  - Ex: Exponential function \( r(\gamma) = \exp(\zeta \gamma) \)
  - Ex: Linear inverse function \( r(\gamma) = (1 - \frac{\gamma}{\gamma_{\text{max}}})^{-1} \)

- All \( \mathbf{K} \) have identical eigenvectors, just vary the eigenvalues \( r^{-1}(\gamma_i) \)
- Same predictors in the kernel regression, difference in penalty term
Example: kernels in the lawyer collaboration graph

- Network of lawyer collaboration, connected component with $N_v = 34$
- Left figure shows the eigenvalues $\gamma_1, \ldots, \gamma_{34}$ of $L$, recall $\gamma_1 = 0$

![Eigenvalues and Regularized Eigenvalues](image)

- Right figure shows values of $r^{-1}(\gamma_i)$, for $i = 1, \ldots, 34$
- Regularizers: identity, exponential, and linear inverse functions
  - First two damp most eigenvalues, only few $\phi_i$ affect $K$
  - Small decay in the last, all $\phi_i$ play a substantial role in $K$
Visual representation of eigenvectors

- Visual representation of 8 ‘smallest’ eigenvectors $\phi_i, i = 2, \ldots, 9$
- Vertex size proportional to the component in $\phi_i$, color indicates sign

- Early eigenvectors have entries relatively more uniform in size and color
  $\Rightarrow$ Eigenvectors become less ‘smooth’ with increasing eigenvalue
Case study

Nearest-neighbor prediction

Markov random fields

Kernel regression on graphs

Case study: Predicting protein function
Predicting protein function

- Proteins integral to complex biochemical processes within organisms
- Understanding their function is critical in biology and medicine
- But ~ 70% of genes code for proteins with unknown function
  ⇒ Prediction of protein function a task of great importance

- Methodologies that have been explored so far:
  (i) Traditional experiment-intensive approaches
  (ii) Methods based on sequence-similarity, protein structure
  (iii) Network-based methods
  ⇒ Networks of protein-protein interactions are natural here
Protein-protein interaction network

- Baker’s yeast data, formally known as *Saccharomyces cerevisiae*
- **Graph:** 134 vertices (proteins) and 241 edges (protein interactions)

![Network of interactions among proteins known to be responsible for cell communication in yeast. Yellow vertices denote proteins that are known to be involved in intracellular signaling cascades, a specific form of communication in the cell. The remaining proteins are indicated in blue.]

- Predict functional annotation **intracellular signaling cascade (ICSC)**
- Deals with signal transduction, how cells react to the environment
- Let $X = \{X_i\}_{i \in V}$ denote the vertex process of the annotation ICSC
  - $X_i = 1$ if protein $i$ annotated ICSC (yellow), $X_i = 0$ otherwise (blue)
Methods to predict protein function

- **Method 1:** nearest-neighbor (NN) prediction with varying threshold $\tau$
- **Method 2:** MRF with predictors counting nodes with and without ICSC
- Parameters $(\alpha, \beta_1, \beta_2)$ estimated via maximum pseudo-likelihood
- Generated 1,000 samples of vertex annotations using a Gibbs sampler
- Predictions based on empirical estimates of $P\left(X_i = 1 \mid X^{obs} = x^{obs}\right)$
- **Method 3:** kernel logistic regression (KLR) with $K = L^\dagger$ and $\lambda = 0.01$
- In all cases predictions generated using 10-fold cross validation
  - $\Rightarrow$ 90% of the labels used to train the prediction methods
  - $\Rightarrow$ Remaining 10% used to test obtained predictors
Nearest-neighbor prediction

- Empirical proportions of neighbors with and without ICSC

- Classes less-well separated than for the lawyer-collaboration data

- Recall nearest-neighbor prediction rule for $\tau = 0.5$ is

$$\hat{x}_i = \mathbb{I}\left\{\frac{\sum_{j \in \mathcal{N}_i} x_j}{\mathcal{N}_i} > 0.5\right\}$$

- Yields a decent misclassification rate of roughly 23%
Receiver operating characteristic

Receiver operating characteristic curves depict predictive performance.

All methods performed comparably. Area under the curve values:
NN - 0.80, MRF - 0.82, KLR - 0.83, KLR w/motifs - 0.85
Not surprising that all three methods performed similarly
\( \Rightarrow \) NN and MRF use the same statistics \( \sum_{j \in N_i} x_j \) and \( \sum_{j \in N_i} (1 - x_j) \)
\( \Rightarrow \) NN equivalent to a form of graph partitioning [Blum-Chawla’01]
\( \Rightarrow \) \( L \) at the heart of many graph partitioning algorithms

Simple NN prediction comparable to sophisticated classification methods

MRF and kernels flexible to incorporate information beyond \( G \)

Ex: certain DNA sequence motifs useful for function prediction

114 out of 134 proteins associated with one or more of 154 motifs

Encode associations in \( M \in \{0, 1\}^{134 \times 154} \), construct kernel \( \tilde{K} = MM^T \)

Improvement in performance with the combined kernel

\[ K = 0.5 \times L^\dagger + 0.5 \times MM^T \]
Glossary

- Graph-indexed process
- Static process
- Dynamic process
- Nearest-neighbor prediction
- Model-based prediction
- Markov random fields
- Ising model
- Gibbs random fields
- Partition function
- Clique potentials
- Auto models
- Pseudo-likelihood

- Gibbs sampler
- Kernel function
- Kernel regression
- Representer theorem
- Kernel logistic regression
- Graph kernels
- Diffusion kernel
- Regularized Laplacian
- Protein function
- ROC curve
- Area under the curve
- Combined kernels