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.hajim.rochester.edu/ece/sites/gmateos/`

April 10, 2023

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
- Often some quantity associated with each of the elements is of interest
- Quantities may be influenced by the interactions among elements
  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 $\mathbb{R}_+$

Nearest-neighbor prediction

- 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

Static network process prediction

Predict X_i, given observations of the adjacency matrix $\mathbf{A} = \mathbf{a}$ and of all attributes $\mathbf{X}^{(-i)} = \mathbf{x}^{(-i)}$ but X_i.

- Idea: exploit the network graph structure in $\mathbf{A}$ for prediction
- For binary $X_i \in \{0, 1\}$, say, simple nearest-neighbor method predicts

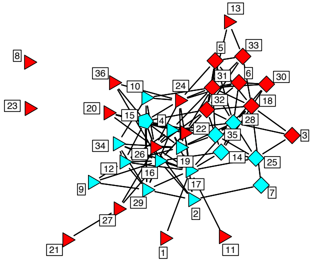
$$\hat{X}_i = \mathbb{I}\left\{\frac{\sum_{j \in \mathcal{N}_i} x_j}{|\mathcal{N}_i|} > \tau\right\}$$

$\Rightarrow$ Average of the observed process in the neighborhood of i

$\Rightarrow$ 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
  - $\Rightarrow$ Type of practice, i.e., litigation (red) and corporate (cyan)
- Suspect lawyers collaborate more with peers in same legal practice
  - $\Rightarrow$ Knowledge of collaboration useful in predicting type of practice

Example: predicting law practice (cont.)

- Q: In predicting practice X_i, how useful is the value of one neighbor?

  $\Rightarrow$ Breakdown of 115 edges based on practice of incident lawyers

	Litigation	**Corporate**
Litigation	29	43
Corporate	43	43

- Looking at the rows in this table
  - Litigation lawyers collaborators are 40% litigation, 60% corporate
  - Collaborations of corporate lawyers are evenly split

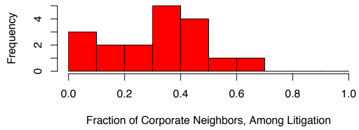
  $\Rightarrow$ Suggests using a single neighbor has little predictive power

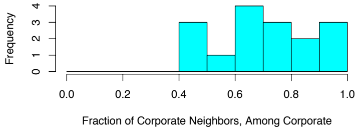
- But 60% (29+43=72) of edges join lawyers with common practice

  $\Rightarrow$ 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 $X_i = 1$ for corporate





- Nearest-neighbor prediction rule

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

$\Rightarrow$ Infers correctly 13 of the 16 corporate lawyers (i.e., 81%)

$\Rightarrow$ Infers correctly 16 of the 18 litigation lawyers (i.e., 89%)

$\Rightarrow$ Overall error rate is just under 15%

Modeling static network processes

- Nearest-neighbor methods may seem rather informal and simple
  - $\Rightarrow$ 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 $\mathbf{X} := \{X_i\}_{i \in V}$ given an observed graph $\mathbf{A} = \mathbf{a}$
  - $\Rightarrow$ Markov random field (MRF) models
  - $\Rightarrow$ Kernel-regression models using graph kernels

Markov random fields

Nearest-neighbor prediction

Markov random fields

Kernel regression on graphs

Case study: Predicting protein function

Markov random field models

- Consider a graph $G(V, E)$ with given adjacency matrix $\mathbf{A}$

  $\Rightarrow$ Collection of discrete RVs $\mathbf{X} = [X_1, \ldots, X_{N_v}]^\top$ defined on V

- **Def:** process $\mathbf{X}$ is a Markov random field (MRF) on G if

$$\mathrm{P}\left(X_i = x_i \,\middle|\, \mathbf{X}^{(-i)} = \mathbf{x}^{(-i)}\right) = \mathrm{P}\left(X_i = x_i \,\middle|\, \mathbf{X}_{\mathcal{N}_i} = \mathbf{x}_{\mathcal{N}_i}\right), \;\; i \in V$$

  - X_i conditionally independent of other X_k, given neighbors values
  - 'Spatial' Markov property, generalizing Markov chains in time
  - G defines neighborhoods $\mathcal{N}_i$, hence dependencies

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

  $\Rightarrow$ MRFs used extensively in spatial statistics and image analysis

- Definition requires a technical condition $\mathrm{P}(\mathbf{X} = \mathbf{x}) > 0$, for all $\mathbf{x}$

MRFs and Gibbs random fields

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

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

$\Rightarrow$ Energy function $U(\cdot)$, partition function $\kappa=\sum_{\mathbf{x}}\exp\{U(\mathbf{x})\}$

$\Rightarrow$ Equivalence follows from the Hammersley-Clifford theorem

- Energy function decomposable over the maximal cliques in G

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

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

- Can show $\mathrm{P}\left(X_i \,\middle|\, \mathbf{X}^{(-i)}\right)$ depends only on cliques involving vertex i

Example: auto-logistic MRFs

- May specify MRFs through choice of clique potentials $U_c(\cdot)$
- Ex: Class of auto models are defined through the constraints:
  - (i) Only cliques $c \in \mathcal{C}$ of size one and two have $U_c \neq 0$
  - (ii) Probabilities $\mathrm{P}\left(X_i \,\middle|\, \mathbf{X}_{\mathcal{N}_i}\right)$ have an exponential family form
- For binary RVs $X_i \in \{0, 1\}$, the energy function takes the form

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

- Resulting MRF is known as auto-logistic model, because

$$\mathrm{P}\left(X_i = 1 \,\middle|\, \mathbf{X}_{\mathcal{N}_i} = \mathbf{x}_{\mathcal{N}_i}\right) = \frac{\exp\{\alpha_i + \sum_{j \in \mathcal{N}_i} \beta_{ij} x_j\}}{1 + \exp\{\alpha_i + \sum_{j \in \mathcal{N}_i} \beta_{ij} x_j\}}$$

$\Rightarrow$ Logistic regression of x_i on its neighboring x_j's

$\Rightarrow$ Ising model a special case, when G is a regular lattice

Homogeneity assumptions

- Typical to assume that parameters α_i and β_{ij} are homogeneous
- Ex: Specifying $\alpha_i = \alpha$ and $\beta_{ij} = \beta$ yields conditional log-odds

$$\log\left[\frac{\mathrm{P}\left(X_i = 1 \,\middle|\, \mathbf{X}_{\mathcal{N}_i} = \mathbf{x}_{\mathcal{N}_i}\right)}{\mathrm{P}\left(X_i = 0 \,\middle|\, \mathbf{X}_{\mathcal{N}_i} = \mathbf{x}_{\mathcal{N}_i}\right)}\right] = \alpha + \beta \sum_{j \in \mathcal{N}_i} x_j$$

$\Rightarrow$ Linear 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

$$\log\left[\frac{\mathrm{P}\left(X_i = 1 \,\middle|\, \mathbf{X}_{\mathcal{N}_i} = \mathbf{x}_{\mathcal{N}_i}\right)}{\mathrm{P}\left(X_i = 0 \,\middle|\, \mathbf{X}_{\mathcal{N}_i} = \mathbf{x}_{\mathcal{N}_i}\right)}\right] = \alpha + \beta_1 \sum_{j \in \mathcal{N}_i} x_j + \beta_2 \sum_{j \in \mathcal{N}_i} (1 - x_j)$$

$\Rightarrow$ 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
  - $\Rightarrow$ Gaussian distribution common for analytical tractability
- Ex: auto-Gaussian model specifies Gaussian $X_i \,\big|\, \mathbf{X}_{\mathcal{N}_i} = \mathbf{x}_{\mathcal{N}_i}$, with

$$\mathbb{E}\left[X_i \,\big|\, \mathbf{X}_{\mathcal{N}_i} = \mathbf{x}_{\mathcal{N}_i}\right] = \alpha_i + \sum_{j \in \mathcal{N}_i} \beta_{ij}(x_j - \alpha_j)$$

$$\text{var}\left[X_i \,\big|\, \mathbf{X}_{\mathcal{N}_i} = \mathbf{x}_{\mathcal{N}_i}\right] = \sigma^2$$

  - $\Rightarrow$ Values X_i modeled as weighted combinations of i's neighbors
- Let $\boldsymbol{\mu} = [\alpha_1, \ldots, \alpha_{N_v}]^\top$ and $\boldsymbol{\Sigma} = \sigma^2(\mathbf{I} - \mathbf{B})^{-1}$, where $\mathbf{B} = [\beta_{ij}]$
  - $\Rightarrow$ Under $\beta_{ii} = 0$ and $\beta_{ij} = \beta_{ji} \rightarrow \mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
- Homogeneity assumptions can be imposed, simplifying expressions
  - $\Rightarrow$ Further set $\alpha_i = \alpha$ and $\beta_{ij} = \beta \rightarrow \mathbf{X} \sim \mathcal{N}(\alpha\mathbf{1}, \sigma^2(\mathbf{I} - \beta\mathbf{A})^{-1})$

Inference and prediction for MRFs

- In studying process $\mathbf{X} = \{X_i\}_{i \in V}$ of interest to predict some or all of $\mathbf{X}$
- MRF models we have seen for this purpose are of the form

$$\mathrm{P}_\theta(\mathbf{X} = \mathbf{x}) = \left(\frac{1}{\kappa(\boldsymbol{\theta})}\right) \exp\{U(\mathbf{x}; \boldsymbol{\theta})\}$$

$\Rightarrow$ Parameter $\boldsymbol{\theta}$ low-dimensional, e.g., $\boldsymbol{\theta} = [\alpha, \beta]$ in auto-models

- Predictions can be generated based on the distribution $\mathrm{P}_\theta(\cdot)$

  $\Rightarrow$ Knowledge of $\boldsymbol{\theta}$ is necessary, and typically $\boldsymbol{\theta}$ is unknown

- Unlike nearest-neighbors prediction, MRFs requires inference of θ first

Inference for MRFs

- Estimation of $\boldsymbol{\theta}$ most naturally approached via maximum-likelihood
- Even though the log-likelihood function takes a simple form

$$\ell(\theta) = \log \mathrm{P}_\theta(\mathbf{X} = \mathbf{x}) = U(\mathbf{x};\boldsymbol{\theta}) - \log \kappa(\boldsymbol{\theta})$$

$\Rightarrow$ Computing $\kappa(\boldsymbol{\theta}) = \sum_{\mathbf{x}} \exp\{U(\mathbf{x};\boldsymbol{\theta})\}$ often intractable

- Popular alternative is maximum pseudo-likelihood, i.e., maximize

$$\sum_{i \in V} \log \mathrm{P}_\theta \left(X_i = x_i \,\middle|\, \mathbf{X}^{(-i)} = \mathbf{x}^{(-i)} \right)$$

$\Rightarrow$ Ignores dependencies beyond the neighborhood of each X_i

$\Rightarrow$ Probabilities depend on clique potentials U_c, not on $\kappa(\boldsymbol{\theta})$

Gibbs sampler

- Given a value of $\boldsymbol{\theta}$, consider predicting some or all of $\mathbf{X}$ from $\mathrm{P}_\theta(\cdot)$

  $\Rightarrow$ Computing $\mathrm{P}_\theta(\cdot)$ hard, can draw from it using a Gibbs sampler

- Gibbs sampler exploits $\mathrm{P}_\theta\left(X_i \,\middle|\, \mathbf{X}^{(-i)} = \mathbf{x}^{(-i)}\right)$ in simple closed form
  - New value $\mathbf{X}_{(k)}$ obtained from $\mathbf{X}_{(k-1)} = \mathbf{x}_{(k-1)}$ by drawing

$$\begin{array}{ccc} X_{1,(k)} & \text{from} & \mathrm{P}_\theta\left(X_1 \,\middle|\, \mathbf{X}^{(-1)} = \mathbf{x}^{(-1)}_{(k-1)}\right) \\ & \vdots & \\ X_{N_v,(k)} & \text{from} & \mathrm{P}_\theta\left(X_{N_v} \,\middle|\, \mathbf{X}^{(-N_v)} = \mathbf{x}^{(-N_v)}_{(k-1)}\right) \end{array}$$

  $\Rightarrow$ Generated sequence $\mathbf{X}_{(1)}, \mathbf{X}_{(2)}, \ldots$ forms a Markov chain

- Under appropriate conditions, stationary distribution equals $\mathrm{P}_\theta(\cdot)$

Prediction with MRFs

- Given large sample from $\mathrm{P}_\theta(\cdot)$, predict $\mathbf{X}$ using empirical distributions
  Ex: for binary $\mathbf{X}$ use empirical marginal frequencies to predict X_i, i.e.,

$$\hat{X}_i = \mathbb{I}\left\{\frac{1}{n}\sum_{k=m+1}^{m+n} X_{i,(k)} > 0.5\right\} \text{ for large } m, n$$

- Suppose we observe some elements $\mathbf{X}^{obs} = \mathbf{x}^{obs}$, and wish to predict $\mathbf{X}^{miss}$
  $\Rightarrow$ Draw from the relevant $\mathrm{P}_\theta\left(\mathbf{X}^{miss} \,\middle|\, \mathbf{X}^{obs} = \mathbf{x}^{obs}\right)$ as

$$X_{i,(k)} \quad \text{from} \quad \mathrm{P}_\theta\left(X_i \,\middle|\, \mathbf{X}^{obs} = \mathbf{x}^{obs}, \mathbf{X}^{(-i),miss} = \mathbf{x}^{(-i),miss}_{(k-1)}\right)$$

  $\Rightarrow$ Prediction from empirical distributions analogous
- Prior inference of $\boldsymbol{\theta}$ based on limited data $\mathbf{X}^{obs} = \mathbf{x}^{obs}$ 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?
  A1: Yes! A regression-based approach on G is in order
- Methods such as LS regression relate data in Euclidean space
- Q2: Can these methods be tuned to accommodate graph-indexed data?
  A2: Yes! Kernel methods consisting of:
  1) Generalized predictor variables (i.e., encoded using a kernel)
  2) 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 $\mathbf{X} = \{X_i\}_{i \in V}$ a vertex attribute process
  - $\Rightarrow$ Suppose we observe $X_i = x_i$ for $i \in V^{obs} \subset V$, with $n = |V^{obs}|$

Regression on graphs

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 a called a kernel if for each $m = 1, \ldots, N_v$ and subset of vertices $\{i_1, \ldots, i_m\} \subseteq V$, matrix

  $\mathbf{K}^{(m)} = [K(i_j, i_{j'})] \in \mathbb{R}^{m \times m}$ is symmetric and positive semi-definite

- Think of kernels as functions that produce similarity matrices
  - $\Rightarrow$ Kernel regression builds predictors from such similarities
  - $\Rightarrow$ Need to also decide on the space $\mathcal{H}$ where to search for $\hat{h}$

Reproducing-kernel Hilbert spaces

- Since V is finite, represent functions h on V as vectors $\mathbf{h} \in \mathbb{R}^{N_v}$
  - $\Rightarrow$ Form $\mathbf{K}^{(N_v)} \in \mathbb{R}^{N_v \times N_v}$ by evaluating K in all pairs $(i, j) \in V^{(2)}$
  - $\Rightarrow$ Suppose $\mathbf{K}^{(N_v)}$ admits an eigendecomposition

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

Kernel regression

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

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

- $\mathcal{H}_K$ is the reproducing-kernel Hilbert space induced by K
  - $\Rightarrow$ Members $\mathbf{h} \in \mathcal{H}_K$ are linear combinations of eigenvectors of $\mathbf{K}^{(N_v)}$
  - $\Rightarrow$ Constrained to finite norm $\|\mathbf{h}\|_{\mathcal{H}} = \|\mathbf{\Phi}\boldsymbol{\beta}\|_{\mathcal{H}} := \boldsymbol{\beta}^\top \mathbf{\Delta}^{-1} \boldsymbol{\beta} < \infty$

Penalized regression in RKHS

- Choose appropriate $\hat{\mathbf{h}} \in \mathcal{H}_K$ using penalized kernel regression
- Q: Appropriate? Data fidelity and small norm (i.e., low complexity)

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

  - Convex loss $C(\cdot, \cdot)$ encourages goodness of fit to $\mathbf{x}^{obs}$
  - The term $\|\mathbf{h}\|_{\mathcal{H}} = \boldsymbol{\beta}^\top \mathbf{\Delta}^{-1} \boldsymbol{\beta}$ penalizes excessive complexity
  - Tuning parameter λ trades off data fidelity and complexity

- Generalized ridge-regression with columns of $\mathbf{\Phi}$ as predictors

$\Rightarrow$ Eigenvectors with small eigenvalues penalized more harshly

Representer theorem

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

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

- Just need to evaluate K in $V^{obs} \times V^{obs}$ to form $\mathbf{K}^{(n)}$
  - $\Rightarrow$ Complexity scales with the number of observations n, not N_v
- Because $\hat{\mathbf{h}} = \mathbf{K}^{(N_v,n)}\hat{\boldsymbol{\alpha}}$, 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

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

$\Rightarrow$ Particular method known as kernel ridge regression. Intuition?

- Define $\boldsymbol{\theta} := (\mathbf{K}^{(n)})^{1/2}\boldsymbol{\alpha}$ and $\mathbf{M} := (\mathbf{K}^{(n)})^{1/2}$. An equivalent problem is

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

- Standard ridge regression with solution $\hat{\boldsymbol{\theta}} = (\mathbf{M}^\top \mathbf{M} + \lambda \mathbf{I})^{-1} \mathbf{M}^\top \mathbf{x}^{obs}$

$\Rightarrow$ The kernel regression function is $\hat{\mathbf{h}} = \mathbf{K}^{(N_v, n)} (\mathbf{K}^{(n)})^{-1/2} \hat{\boldsymbol{\theta}}$

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

$$C(x, a) = \ln\left(1 + e^{-xa}\right)$$

$\Rightarrow$ Corresponds to the negative log-likelihood of a Bernoulli RV

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

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

$\Rightarrow$ No closed-form solution for $\hat{\boldsymbol{\alpha}}$, need iterative algorithms

- Given $\hat{\mathbf{h}} = \mathbf{K}^{(N_v, n)} \hat{\boldsymbol{\alpha}}$, prediction of X_i for $i \in V^{miss}$ based on

$$\hat{\mathrm{P}}\left(X_i = 1 \,\middle|\, \mathbf{X}^{obs} = \mathbf{x}^{obs}\right) = \frac{e^{\hat{h}_i}}{1 + e^{\hat{h}_i}}$$

Designing kernels on graphs

- In designing a kernel K on a graph G, desired properties are:
  - P1) $\mathbf{K}^{(N_v)}$ is symmetric and positive semi-definite
  - P2) K captures suspected similarity among vertices in V
- Presumption: proximity of vertices in G already indicative of similarity

  $\Rightarrow$ Most kernels proposed are related to the topology of G
- Ex: the Laplacian kernel is $\mathbf{K}^{(N_v)} := \mathbf{L}^{\dagger}$, where † denotes pseudo-inverse

  $\Rightarrow$ Penalty term $\|\mathbf{h}\|_{\mathcal{H}} = \boldsymbol{\beta}^{\top}\boldsymbol{\Delta}^{-1}\boldsymbol{\beta}$ takes the form

$$
\begin{aligned}
\boldsymbol{\beta}^{\top}\boldsymbol{\Delta}^{-1}\boldsymbol{\beta} &= \boldsymbol{\beta}^{\top}\boldsymbol{\Phi}^{\top}\boldsymbol{\Phi}\boldsymbol{\Delta}^{-1}\boldsymbol{\Phi}^{\top}\boldsymbol{\Phi}\boldsymbol{\beta} \\
&= \mathbf{h}^{\top}\mathbf{K}^{\dagger}\mathbf{h} = \mathbf{h}^{\top}\mathbf{L}\mathbf{h} \\
&= \sum_{(i,j)\in E} (h_i - h_j)^2
\end{aligned}
$$

- Kernel regression seeks smooth $\hat{\mathbf{h}}$ with respect to the topoology of G

Diffusion kernels

- Laplacian kernel $\mathbf{K} = \mathbf{L}^{\dagger}$ encodes similarity among vertices through $\mathbf{A}$

  $\Rightarrow$ Can encode similarity through paths, powers of $\mathbf{A}$ and $\mathbf{L}$

- Popular choice incorporating all powers of $\mathbf{L}$ is the diffusion kernel

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

  - Decay factor $0 < \zeta < 1$ controls similarity assigned to longer paths
  - Defined in terms of the matrix exponential $e^{-\zeta \mathbf{L}}$

- Treating $\mathbf{K}$ as a function of ζ yields the differential equation

$$\frac{\partial \mathbf{K}}{\partial \zeta} = -\mathbf{L}\mathbf{K}$$

  $\Rightarrow$ Parallels the heat equation in physics, motivating its name

Regularized Laplacian kernels

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

$$\mathbf{K} = \sum_{i=1}^{N_v} r^{-1}(\gamma_i)\phi_i\phi_i^\top$$

$\Rightarrow$ $\mathbf{K}$ is the inverse of the regularized Laplacian $r(\mathbf{L}) := \mathbf{\Phi} r(\mathbf{\Gamma})\mathbf{\Phi}^\top$

- Regularization function $r(\cdot) \geq 0$ is increasing, including:

  Ex: Identity function $r(\gamma) = \gamma$

  Ex: Exponential function $r(\gamma) = \exp(\zeta\gamma)$

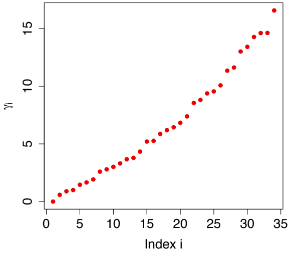
  Ex: Linear inverse function $r(\gamma) = (1 - \frac{\gamma}{\gamma_{\max}})^{-1}$

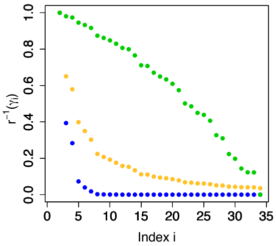
- All $\mathbf{K}$ have identical eigenvectors, just vary the eigenvalues $r^{-1}(\gamma_i)$

$\Rightarrow$ Same predictors in the kernel regression, different penalty

Example: kernels in the lawyer collaboration graph

- Network of lawyer collaboration, connected component with $N_v = 34$





  - Left figure shows eigenvalues $\gamma_1, \ldots, \gamma_{34}$ of $\mathbf{L}$, recall $\gamma_1 = 0$
  - Right figure shows values of $r^{-1}(\gamma_i)$, for $i = 2, \ldots, 34$

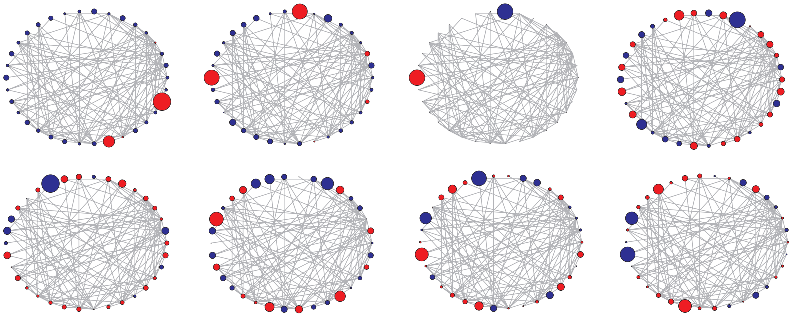
- Regularizers: identity, exponential, and linear inverse functions

  $\Rightarrow$ First two damp most eigenvalues, only few ϕ_i affect $\mathbf{K}$

  $\Rightarrow$ Small decay in the last, all ϕ_i play a substantial role in $\mathbf{K}$

Visual representation of eigenvectors

- Visual representation of 8 'smallest' eigenvectors ϕ_i, $i = 2, \ldots, 9$
  - Vertex size proportional to the component in ϕ_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

  $\Rightarrow$ Understanding their function is critical in biology and medicine
- But $\sim 70\%$ of genes code for proteins with unknown function

  $\Rightarrow$ Prediction of protein function a task of great importance
- Methodologies explored so far:

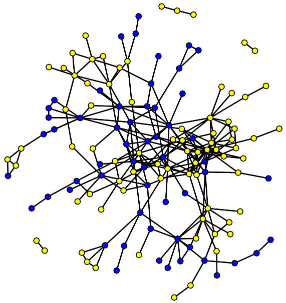
  (i) Traditional experiment-intensive approaches

  (ii) Methods based on sequence-similarity, protein structure

  (iii) Network-based methods
- Networks of protein-protein interactions natural in the latter

Protein-protein interaction network

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



- Predict functional annotation intracellular signaling cascade (ICSC)

  $\Rightarrow$ Signal transduction, how cells react to the environment

- Let $\mathbf{X} = \{X_i\}_{i \in V}$ 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 τ

Method 2: MRF with predictors counting nodes with and without ICSC

- Parameters $(\alpha, \beta_1, \beta_2)$ estimated via maximum pseudo-likelihood
- Drew 1,000 samples of vertex annotations using a Gibbs sampler
- Predictions based on empirical estimates of $\mathrm{P}\left(X_i = 1 \,\middle|\, \mathbf{X}^{obs} = \mathbf{x}^{obs}\right)$

Method 3: kernel logistic regression (KLR) with $\mathbf{K} = \mathbf{L}^{\dagger}$ and $\lambda = 0.01$

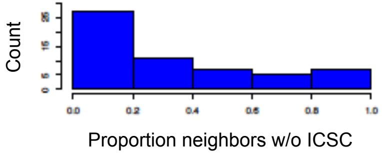
- In all cases predictions generated using 10-fold cross validation

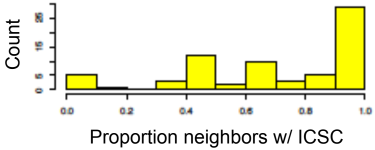
  $\Rightarrow$ 90% of the labels used to train the prediction methods

  $\Rightarrow$ Remaining 10% used to test obtained predictors

Nearest-neighbor prediction

- Empirical proportions of neighbors with and without ICSC





$\Rightarrow$ Classes less-well separated than for the lawyer data

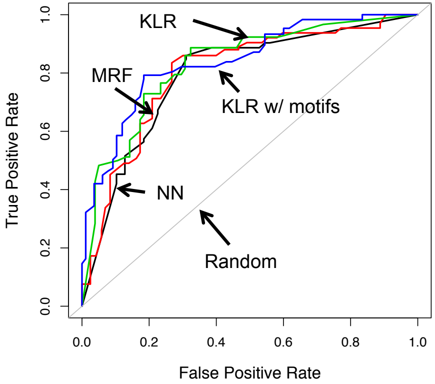
- Recall nearest-neighbor prediction rule for $\tau = 0.5$ is

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

$\Rightarrow$ Yields a decent missclasification rate of roughly 23%

Receiver operating characteristic

- ROC 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

Closing remarks

- Not surprising that all three methods performed similarly
  - $\Rightarrow$ NN and MRF use same statistics $\sum_{j \in \mathcal{N}_i} x_j$ and $\sum_{j \in \mathcal{N}_i} (1 - x_j)$
  - $\Rightarrow$ NN equivalent to a form of graph partitioning [Blum-Chawla'01]
  - $\Rightarrow$ $\mathbf{L}$ key to many graph partitioning algorithms
- Simple NN prediction comparable to sophisticated classification methods
  - $\Rightarrow$ 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 $\mathbf{M} \in \{0, 1\}^{134 \times 154}$, construct kernel $\bar{\mathbf{K}} = \mathbf{M}\mathbf{M}^\top$

  $\Rightarrow$ Improvement in performance with the combined kernel

$$\mathbf{K} = 0.5 \times \mathbf{L}^\dagger + 0.5 \times \mathbf{M}\mathbf{M}^\top$$

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