# SEMI-SUPERVISED LEARNING

Matt Stokes

November 3, 2011

## Topics

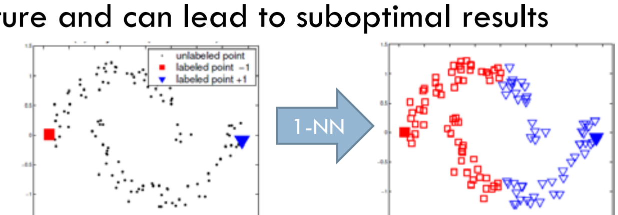
- Background
- Label Propagation
  - Definitions
  - Transition matrix (random walk) method
  - Harmonic solution
  - Graph Laplacian method
- Kernel Methods
  - Smoothness
  - Kernel alignment

# Types of Learning

- Unsupervised
  - Class labels are unknown
  - No feedback/error signal
  - Essentially density estimation
- Supervised
  - Given labeled training examples
  - Can evaluate performance directly
  - Learn mapping of $X$ to $Y$
- Semi-supervised
  - Only some samples are labeled
  - Saves time/cost of labeling large datasets

# Assumptions

- Data exist in some kind of clusters
- Local assumption
  - Points near one another likely to have the same label
- Global assumption
  - Points on the same structure (i.e. manifold) likely to have the same label
- Simple clustering methods (k-NN) rely only on local structure and can lead to suboptimal results



# Label Propagation

## Problem setup (Zhu, 2002)

- Data $(x_1, y_1)\ldots(x_N, y_N)$ consist of:
  - $L$ labeled samples $(x_1, y_1)\ldots(x_L, y_L)$
  - $U$ unlabelled samples $(x_{L+1}, y_{L+1})\ldots(x_{L+U}, y_{L+U})$ where class labels $\{y_{L+1}\ldots y_{L+U}\}$ are unknown
  - Usually, $L<<U$
  - Number of classes (C) is known
- Create a fully connected graph with samples as nodes, connection weights proportional to sample proximity

$$w_{ij} = \exp\left(-\frac{d_{ij}^2}{\sigma^2}\right) = \exp\left(-\frac{\sum_{d=1}^{D}(x_i^d - x_j^d)^2}{\sigma^2}\right)$$

- Asymmetric transition matrix $T$ has dimensions $N$ x $N$

$$T_{ij} = P(j \to i) = \frac{w_{ij}}{\sum_{k=1}^{N} w_{kj}}$$

## Label Propagation

- Node labels represented as a distribution over classes in label matrix $Y$ ($N$ rows, $C$ columns)
- Begin with arbitrary assignment of class distributions to unlabeled points, known class to labeled points
- Repeat:
  - 1. Propagate $Y \leftarrow TY$
    - Labels spread information along local structure
  - 2. Row normalize $Y$
    - Keep proper distribution over classes
  - 3. Clamp labeled data to original value
    - Keep originally labeled points

$$Y_{ic} = \delta(y_i, c)$$

## Convergence

- Represent as row-normalized block matrices: $Y = \begin{bmatrix} Y_L \\ Y_U \end{bmatrix} \leftarrow \begin{bmatrix} \overline{T}_{ll} & \overline{T}_{lu} \\ \overline{T}_{ul} & \overline{T}_{uu} \end{bmatrix} \begin{bmatrix} Y_L \\ Y_U \end{bmatrix}$
- Iterative update for $Y_U$
  - $Y_L$ is clamped at original values

$$Y_U \leftarrow \overline{T}_{ul} Y_L + \overline{T}_{uu} Y_U$$

- Result of iteration: $Y_U = \left[ \sum_{i=1}^{n} \overline{T}_{uu}^{(i-1)} \right] \overline{T}_{ul} Y_L + \lim_{n \to \infty} \overline{T}_{uu}^{\,n} Y^0$
- Because $T$ row-normalized and $T_{uu}$ is a submatrix, we have: $\lim_{n \to \infty} \overline{T}_{uu}^{\,n} Y^0 = 0$
- Converges regardless of initial $Y^0$: $Y_U = \left( I - \overline{T}_{uu} \right)^{-1} \overline{T}_{ul} Y_L$

## Class Assignment

- How should we assign classes to unlabeled points?
- Could choose most likely class
  - ML method does not explicitly control class proportions
- Suppose we want labels to fit a known or estimated distribution over classes
  - Normalize class mass – scale columns of $Y_U$ to fit class distribution and then pick ML class
    - Does not guarantee strict label proportions
  - Perform label bidding – each entry $Y_U(i,c)$ is a "bid" of sample $i$ for class c
    - Handle bids from largest to smallest
    - Bid is taken if class c is not full, otherwise it is discarded

## Parameterization

- Single parameter σ controls spread of labels
  - For $\sigma \to 0$, classification of unlabeled points dominated by nearest labeled point
  - For $\sigma \to \infty$, class probabilities just become class frequencies (no information from label proximity)
- Build minimum spanning tree, longest edges first
  - Set $\sigma = d^*/3$, where $d^*$ is the first edge connecting subgraphs containing differently labeled points
- Can minimize entropy of class labels
  - Leads to confident classifications
  - However, minimum entropy at $\sigma=0$

## Optimizing σ

- Add uniform transition component ($\mathbf{U}_{ij}$=1/N) to *T*

$$\widetilde{T} = \varepsilon \mathbf{U} + (1-\varepsilon)T$$

- For small σ, uniform component dominates
  - Minimum entropy no longer at σ=0
- Use $\sigma_1 \ldots \sigma_N$ to scale each dimension independently
- Perform gradient descent with respect to σ's in order to minimize entropy

$$\frac{\partial H}{\partial \sigma_d} = \sum_{i=L+1}^{L+U} \sum_{c=1}^{C} \frac{\partial H}{\partial Y_{ic}} \frac{\partial Y_{ic}}{\partial \sigma_d}$$

## What is going on?

- Transition matrix *T* holds probabilities of moving from one node to another
- Very similar to Markov random walker
  - However, insensitive to timescale of the walk
  - Constant source labels leads to equilibrium as iterations increase
- Mean field approximation interpretation for pairwise Markov random field *F*
  - Label propagation finds most likely labels for the approximate mean field solution of *F*
  - Not just most likely state (MinCut)
  - Can split clusters equidistant from labeled points

## Harmonic Functions (Zhu, 2003)

- Now define class labeling *f* in terms of a Gaussian over continuous space, instead of random field over discrete label set
- Distribution on *f* is a Gaussian field

$$p_\beta(f) = \frac{e^{-\beta E(f)}}{Z_\beta}$$
$$Z_\beta = \int_{f|_{L=f_l}} \exp(-\beta E(f))df$$

- Useful for multi-label problems (NP-hard for discrete random fields)
  - ML configuration is now unique, attainable by matrix methods, and characterized by harmonic functions

## Harmonic Energy

- "Energy" of solution labeling *f* is defined as:

$$E(f) = \frac{1}{2}\sum_{i,j} w_{ij}(f(i) - f(j))^2$$

  - Nearby points should have similar labels
- Solution which minimizes *E(f)* is harmonic
  - $\Delta f=0$ for unlabeled points, where $\Delta=D-W$ (combinatorial Laplacian)
  - $\Delta f=f_l$ for labeled points
  - Value of *f* at an unlabeled point is the average of *f* at neighboring points

$$f(j) = \frac{1}{d_j}\sum_{i\sim j} w_{ij} f(i), \text{for } j = L+1,\ldots,L+U$$
$$f = D^{-1}Wf$$

## Harmonic Solution

- As before, split problem into:

$$f = \begin{bmatrix} f_l \\ f_u \end{bmatrix} \qquad W = \begin{bmatrix} W_{ll} & W_{lu} \\ W_{ul} & W_{uu} \end{bmatrix} \qquad P = D^{-1}W$$

- Solve using $\Delta f=0$, $f|_L = f_l$ :

$$f_u = (D_{uu} - W_{uu})^{-1} W_{ul} f_l = (I - P_{uu})^{-1} P_{ul} f_l$$

- Can be viewed as heat kernel classification, but independent of time parameter

## Other interpretations

- Consider random walker on data graph with given transition probabilities starting from unlabeled node *i*
  - *f(i)* is the probability that the first labeled node encountered is of class 1
  - Solution is an equilibrium state, not depending on time *t*
- Can also be viewed as electrical network
  - Class 1 labels connected to source, class 0 labels to ground
  - Weights represent conductance
  - $f_u$ is the resulting voltage on an unlabeled node
  - Minimizes energy dissipation in the network

# Reformulation (Zhou)

- Explicitly model self-reinforcement of labeled nodes
  - No clamping of values
  - Original labels stored in *Y*
  - Distribution of labels now stored in *F(t)*
- Information spreads symmetrically
- *S* is the normalized graph Laplacian
  - Identical to spectral clustering
  - Similar to transition matrix
- Note that $(I-\alpha S)^{-1}$ is a diffusion kernel

$$w_{ij} = \exp\left(-\frac{d_{ij}^2}{\sigma^2}\right) for\, i \neq j$$

$$w_{ii} = 0$$

$$D = \begin{bmatrix} \sum_{j=1}^{N} w_{1j} & 0 & 0 & 0 \\ 0 & \sum_{j=1}^{N} w_{2j} & 0 & 0 \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & \dots & \sum_{j=1}^{N} w_{Nj} \end{bmatrix}$$

$$S = D^{-1/2} W D^{-1/2}$$

$$F(t+1) = \alpha S F(t) + (1-\alpha) Y$$

$$F^* = \lim_{t \to \infty} F(t) = (1-\alpha)(I-\alpha S)^{-1} Y$$

# Regularization

- Define cost function Q associated with assignment of class labels F

$$Q(F) = \underbrace{\frac{1}{2} \sum_{i,j=1}^{N} W_{ij} \left\| \frac{F_i}{\sqrt{D_{ii}}} - \frac{F_j}{\sqrt{D_{jj}}} \right\|^2}_{Smoothness} + \underbrace{\mu \sum_{i=1}^{N} \left\| F_i - Y_i \right\|^2}_{Fitting}$$

- Smoothness constraint ensures classification does not change much between nearby points
- Fitting constraint ensures classification does not deviatedmuch from initial assignment
- *F** optimizes solution to the regularized framework

$$\left.\frac{\partial Q}{\partial F}\right|_{F=F^*} = F^* - SF^* + \mu(F^* - Y) = 0$$

$$F^* - \frac{1}{1+\mu} SF^* - \frac{\mu}{1+\mu} Y = 0$$

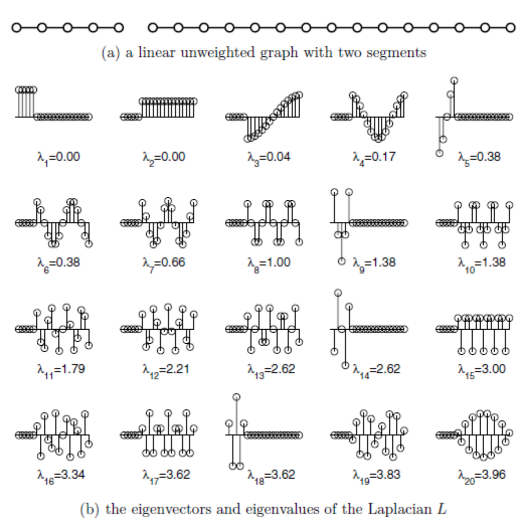
$$\alpha = \frac{1}{1+\mu}$$

$$F^* = (1-\alpha)(I-\alpha S)^{-1} Y$$

# Kernel Methods

## Review

- Graph Laplacian has eigenvectors $\varphi_1 \ldots \varphi_N$, eigenvalues $\lambda_1 \ldots \lambda_N \geq 0$
- Smallest eigenvalues correspond to "smoothest" eigenvectors
- These eigenvectors most useful for classification




(a) a linear unweighted graph with two segments

(b) the eigenvectors and eigenvalues of the Laplacian $L$

## Kernels by Spectral Transform

- Semi-supervised learning creates a smooth function over unlabeled points
  - Generally, smooth if *f(i)≈f(j)* for pairs with large $W_{ij}$

$$f^T L f = \frac{1}{2}\sum_{i,j=1}^{N} W_{ij}(f(i)-f(j))^2 = \sum_{i=1}^{N}\alpha_i^2\lambda_i$$

  - Different weightings (i.e. spectral transforms) of Laplacian eigenvalues leads to different smoothness measures
- We want a kernel *K* that respects smoothness
  - Define using eigenvectors of Laplacian (φ) and eigenvalues of *K* (μ)

$$K = \sum_{i=1}^{N}\mu_i\phi_i\phi_i^T$$

  - Can also define in terms of a spectral transform of Laplacian eigenvalues

$$K = \sum_{i=1}^{N} r(\lambda_i)\phi_i\phi_i^T$$

## Types of Transforms

- $r(\lambda_i)$ is a non-negative and decreasing transform

| | |
|---|---|
| Regularized Laplacian | $r(\lambda) = \frac{1}{\lambda+\varepsilon}$ |
| Diffusion Kernel | $r(\lambda) = \exp\left(-\frac{\sigma^2}{2}\lambda\right)$ |
| 1-step Random Walk | $r(\lambda) = (\alpha-\lambda), \alpha \geq 2$ |
| p-step Random Walk | $r(\lambda) = (\alpha-\lambda)^p, \alpha \geq 2$ |
| Inverse Cosine | $r(\lambda) = \cos(\lambda\pi/4)$ |
| Step Function | $r(\lambda) = 1 \text{ if } \lambda \leq \lambda_{cut}$ |

- Reverses order of eigenvalues, so smooth eigenvectors have larger eigenvalues in *K*
- Is there an optimal transform?

# Kernel Alignment

- Assess fitness of a kernel to training labels
- Empirical kernel alignment compares kernel matrix $K_{tr}$ for training data to target matrix $T$ for training data
  - $T_{ij}=1$ if $y_i=y_j$, otherwise $T_{ij}=-1$

$$\hat{A}(K_{tr},T)=\frac{\langle K_{tr},T\rangle_F}{\sqrt{\langle K_{tr},K_{tr}\rangle_F\langle T,T\rangle_F}}$$

$$\langle M,N\rangle_F = Tr(MN)$$

Frobenius Product

- Alignment measure computes cosine between $K_{tr}$ and $T$
- Find the optimal spectral transformation $r(\lambda_i)$ using the kernel alignment notion

# QCQP

- Kernel alignment between $K_{tr}$ and $T$ is a convex function of kernel eigenvalues $\mu_i$
  - No assumption on parametric form of transform $r(\lambda_i)$
- Need $K$ to be positive semi-definite
  - Restrict eigenvalues of $K$ to be ≥0
- Leads to computationally efficient Quadratically Constrained Quadratic Program
  - Minimize convex quadratic function over smaller feasible region
  - Both objective function and constraints are quadratic
  - Complexity comparable to linear programs

## Constraints

- We would like to keep decreasing order on spectral transformation
  - Smooth functions are preferred – bigger eigenvalues for smoother eigenvectors
- Constant eigenvectors act as a bias term in the graph kernel
  - $\lambda_1=0$, corresponding eigenvector $\varphi_i$ is constant
  - Need not constrain bias terms

$$
\begin{array}{ll}
\max_K & \hat{A}(K_{tr}, T) \\
\text{subject to} & \langle K_{tr}, T \rangle_F \le 1 \\
 & K = \sum_{i=1}^{N} \mu_i K_i \\
 & K_i = \phi_i \phi_i^{\mathrm{T}} \\
 & \mu_i \ge 0 \\
 & \mu_i \ge \mu_{i+1}, i = 1 \ldots n-1, \phi_i \text{ not constant}
\end{array}
$$

## Summary

- Unsupervised learning involves spreading information from labeled nodes to unlabeled nodes
- Multiple formulations with different interpretations
  - Clamped version equivalent to Markov random walk
  - Harmonic solution equivalent to electrical network
  - Unclamped version equivalent to diffusion kernel
- Kernel methods use optimally smoothing spectral transforms of the data
  - Align kernel to labeled training data for optimal performance