Spectral Learning

27 Oct., 2015

Spectral Learning

Fairly broad term.

Any Algorithm that analyzes spectrum(eigenvalues) of operators

Two broad categories:

- **Dimensionality reduction**: Use the decomposition to get a low rank representation of observables. Eg. Spectral Clustering, CCA etc.

- **PGMs with Latent variables**: Use method of moments to estimate parameters and other quantities like marginal probabilities. Eg. HMM, L-PCFG etc.

Singular Value Decomposition

For some $A \in \mathbb{R}^{m \times n}$ with rank k $(k \leq min\{m, n\})$:

$$A = \sum_{i=1}^{k} \Sigma_i U_i V_i^T = \mathbf{U \Sigma V^T}$$

where Σ is a $k \times k$ diagonal matrix, $U_i \in \mathbb{R}^{m \times k}$, $V_i \in \mathbb{R}^{n \times k}$

and $U_1, ..., U_k$ as well as $V_1, ..., V_k$ are orthonormal.

A lot of research on efficient implementations, treat as a black box for this lecture.

Spectral Learning

Fairly broad term.

Any Algorithm that analyzes spectrum(eigenvalues) of operators

Two broad categories:

- **Dimensionality reduction**: Use the decomposition to get a low rank representation of observables. Eg. Spectral Clustering, CCA etc.

- **PGMs with Latent variables**: Use method of moments to estimate parameters and other quantities like marginal probabilities. Eg. HMM, L-PCFG etc.

Spectral Clustering

- Graph based data clustering approach.
- Works on the knowledge of connectivity and local neighborhood similarity
- Similarity graphs model local neighborhood relations between points.
- Uses low rank approximation of these similarity matrices to reduce dimensions and cluster.

Laplacian

Fairly general term but commonly defined as $L = D - W$

where W is the weight matrix and D ix the degree matrix.

$D = diag_{(d_i = \sum_j w_{ij})}(d_1, ..., d_n)$ for n points

Properties of Laplacian:

- L is symmetric and Positive semi definite
- The smallest eigenvalue of L is 0, corresponding to eigenvector of all 1s (multiplicity = #connected components)

$$f^T L f = \frac{1}{2} \sum_{i,j} w_{i,j} (f_i - f_j)^2$$

Min-cut problem?

- Consider the problem of finding two cluster in connected graph.
- **Goal:** Partition the graph into A,B such that weight of the edges connecting A to B is minimum <=> Min-Cut problem
- $cut(A, B) = \sum_{i \in A, j \in B} w_{ij}$
- Easy, but yields terrible solutions, sensitive to outliers.

Normalized Min-Cut?

- **Goal:** Partition the graph into A,B such that weight of the edges connecting A to B is minimum <=> Min-Cut problem and sizes of A and B are similar.
- $Ncut(A,B) = cut(A,B)(\frac{1}{vol(A)} + \frac{1}{vol(B)})$

  where $vol(A) = \sum_i d_i$
- NP Hard to solve. Spectral Clustering is a relaxation of this.

Normalized Min-Cut

$$\text{Let } f \in \mathbb{R}^n \text{ with } f_i = \frac{1}{vol(A)} \text{if } i \in A, \text{ else } \frac{-1}{vol(B)}$$

$$f^T L f = \sum_{i \in A, j \in B} w_{ij} (\frac{1}{vol(A)} + \frac{1}{vol(B)})^2$$

$$f^T D f = \sum_j d_j f_j^2 = \frac{1}{vol(A)} + \frac{1}{vol(B)}$$

$$\text{Therefore, } \min_f Ncut(A, B) = \min_f \frac{f^T L f}{f^T D f}$$

Relaxation

- Relaxation on f vector!

$$\text{Therefore, } \min_f Ncut(A,B) = \min_f \frac{f^T L f}{f^T D f}$$

$$\text{such that } f^T D 1 = 0$$

- Solution: Second eigenvector of a general eigenvalue problem: $$Lf = \lambda D f$$

- Threshold entries of f at 0 for cluster assignments.

- For k clusters?

Spectral Learning

Fairly broad term.

Any Algorithm that analyzes spectrum(eigenvalues) of operators

Two broad categories:

- **Dimensionality reduction**: Use the decomposition to get a low rank representation of observables. Eg. Spectral Clustering, CCA etc.

- **PGMs with Latent variables**: Use method of moments to estimate parameters and other quantities like marginal probabilities. Eg. HMM, L-PCFG etc.

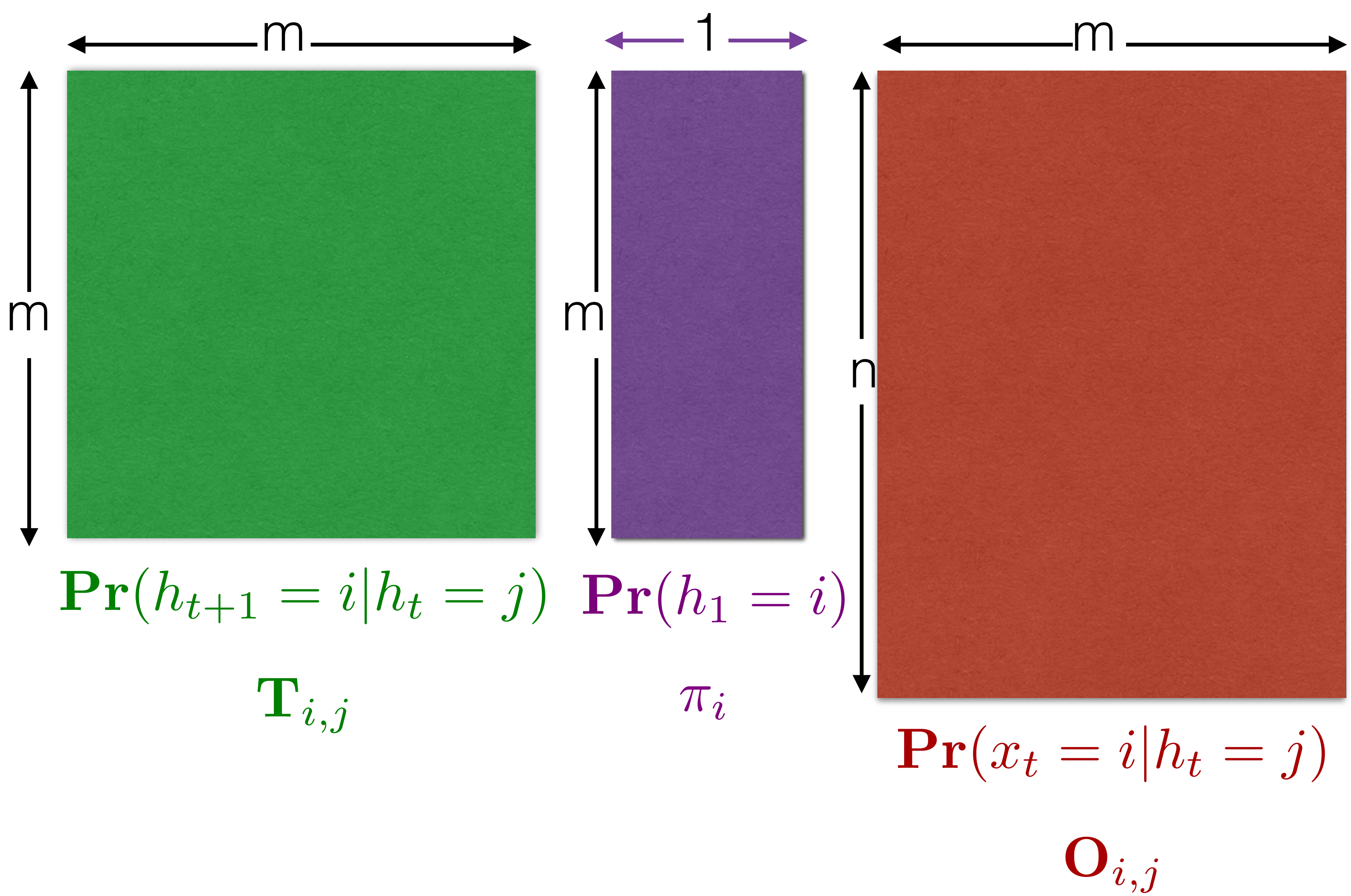
Method of moments

- Characterize the probability distributions by their moments.
- Take moments upto some high order.
- Match the moments to the empirical estimates of the moments and get k equations.
- If k > = #parameters, solve for the parameters.

Method of moments vs MLE

- Simple to implement.
- Yields consistent estimators
- MLE can be intractable or slow, but MoM can generally be quickly computed.
- MLE tends to find better estimates and more often unbiased
- The estimates given by the method of moments can fall outside of the parameter space.
- Method of moments may not capture parameters related to all the sufficient statistics

HMM Parameters



Example

$$\mathbf{Pr}(\,We\ learn\ about\ learning, h_1, h_2, h_3, h_4) =$$

$$\pi(h_1) \times \mathbf{O}(We, h_1) \times \mathbf{T}(h_2, h_1) \times \mathbf{O}(learn, h_2) \times$$

$$\mathbf{T}(h_3, h_2) \times \mathbf{O}(about, h_3) \times \mathbf{T}(h_4, h_3) \times \mathbf{O}(learning, h_4)$$

What about marginal probability?

$$\mathbf{Pr}(\,We\ learn\ about\ learning) =$$

$$\sum_{h_1, h_2, h_3, h_4} \mathbf{Pr}(\,We\ learn\ about\ learning, h_1, h_2, h_3, h_4)$$

Forward Algorithm

$$\alpha_h^0 = \pi(h) \qquad \alpha_h^1 = \sum_{h'} \mathbf{T}(h,h') \times \mathbf{O}(We,h') \times \alpha_{h'}^0$$

$$\alpha_h^2 = \sum_{h'} \mathbf{T}(h,h') \times \mathbf{O}(learn,h') \times \alpha_{h'}^1$$

$$\alpha_h^3 = \sum_{h'} \mathbf{T}(h,h') \times \mathbf{O}(about,h') \times \alpha_{h'}^2$$

$$\alpha_h^4 = \sum_{h'} \mathbf{T}(h,h') \times \mathbf{O}(learning,h') \times \alpha_{h'}^3$$

$$\mathbf{Pr}(We\ learn\ about\ learning) = \sum_{h'} \alpha_{h'}^4$$

- A little bit different from the convention? Looks more like:

$$\alpha^t \propto \mathbf{Pr}(h_{t+1}, x_{1..t})$$

Matricized forward algorithm

- Summation by dot product enables the following Matrix form:

$$\mathbf{A_x} = \mathbf{T}diag(\vec{O_x})$$

$$\vec{\alpha}^1 = \mathbf{T}diag(\vec{O_{We}})\vec{\pi} = \mathbf{A}_{We}\vec{\pi}$$

$$\vec{\alpha}^2 = \mathbf{A}_{learn}\mathbf{A}_{We}\vec{\pi}$$

$$\vec{\alpha}^4 = \mathbf{A}_{learning}\mathbf{A}_{about}\mathbf{A}_{learn}\mathbf{A}_{We}\vec{\pi}$$

$$\sum_{h'} \alpha^4_{h'} = \vec{\mathbf{1}}^T\vec{\alpha}^4 = \vec{\mathbf{1}}^T\mathbf{A}_{learning}\mathbf{A}_{about}\mathbf{A}_{learn}\mathbf{A}_{We}\vec{\pi}$$

Forward algorithm completely matricized!

But still, how do we use only the observables/ moments ??

Moments for HMM

- Carefully defined moments upto 3rd order sufficient to show correctness and consistency!
- First Order: $[\hat{P_1}]_i = \hat{\mathbf{Pr}}(X_1 = i)$
- Second Order: $[\hat{P_{21}}]_{(i,j)} = \hat{\mathbf{Pr}}(X_2 = i, X_1 = j)$
- Third Order: $[\hat{P_{3x1}}]_{(i,j)} = \hat{\mathbf{Pr}}(X_3 = i, X_2 = x, X_1 = j) \forall x \in \Sigma$
- Also: $[\hat{P_{31}}]_{(i,j)} = [\sum_x \hat{P_{3x1}}]_{(i,j)} = \hat{\mathbf{Pr}}(X_3 = i, X_1 = j)$

Observable Operator

- Need a quantity: $\mathbf{B_x} = G\mathbf{A_x}G^{-1}$ *for an invertible* G

- such that $\mathbf{B_x} = \mathbf{f}(P_1, P_{21}, P_{3x1})$ $G = \mathbf{f}(P_1, P_{21}, P_{3x1})$

- Recall :

$$\mathbf{Pr}(We\ learn\ about\ learning) = \vec{\mathbf{1}}^T\mathbf{A}_{learning}\mathbf{A}_{about}\mathbf{A}_{learn}\mathbf{A}_{We}\vec{\pi}$$

- Observe:

$$\mathbf{B}_{learning}\mathbf{B}_{about}\mathbf{B}_{learn}\mathbf{B}_{We} = G\mathbf{A}_{learning}\mathbf{A}_{about}\mathbf{A}_{learn}\mathbf{A}_{We}G^{-1}$$

- With: $b^{\infty} = \vec{1}^T G$ $b_1 = G\pi$

$$\mathbf{Pr}(We\ learn\ about\ learning) = b^{\infty}\mathbf{B}_{learning}\mathbf{B}_{about}\mathbf{B}_{learn}\mathbf{B}_{We}b_1$$

Observable operator

- With $U, s, V = SVD(P_{21}, m)$

$$b_1 = U^T P_1 = U^T O \vec{\pi}$$

$$b^\infty = (P_{21}^T U)^+ P_1 = \vec{\mathbf{1}}_m^T (U^T O)^{-1}$$

$$B_x = (U^T P_{3x1})(U^T P_{21})^+ = (U^T O) A_x (U^T O)^{-1}$$

- For conditional probabilities:

$$b_t = b_t(x_{1:t-1}) = \frac{B_{x_{t-1}:1} b_1}{b^{\infty T} B_{x_{t-1}:1} b_1}$$

$$Pr(x_t | X_{1:t-1}) = b^{\infty T} B_{x_t} b_t$$

Sanity Check

$$\begin{aligned} P_1^T &= \vec{1}_m^T T diag(\pi) O^T \\ &= \vec{1}_m^T (U^T O)^{-1} (U^T O) T diag(\pi) O^T \\ &= \vec{1}_m^T (U^T O)^{-1} U^T P_{21} \end{aligned}$$

$$b^\infty = P_1^T (U^T P_{21})^+$$

Similarly,

$$P_{3x1} = O A_x T diag(\pi) O^T = O A_x (U^T O)^{-1} U^T P_{21}$$

Can easily use this to verify for B

Recovering parameters

- Can be unstable, but reliable when stable

$$P_{31} = OTTdiag(\pi)O^T$$

Therefore,

$$U^T P_{3x1} = U^T OTO_x Tdiag(\pi)O^T$$

$$= (U^T OT)O_x(U^T OT)^{-1}(U^T P_{31})$$

Can get O from this. Other parameters can be similarly obtained once O is known.

Assumptions

- First three words include the whole vocabulary with infinite data. Although, in practice, we use all the data assuming stationarity.

- Invertibility assumption: The algorithm will break if G is not invertible

- Rank assumption: All the parameters have rank m. The algorithm will be unstable/incorrect with violation of this assumption.

PAC learning

- Estimate the concentration bounds of the observed quantities using McDiarmid's inequality.
- All the bounds for eigenvalues are found.
- Use above two bounds, triangle inequality, and Matrix perturbation theory to estimate bounds over transformed quantities like B.
- Propagation error in the model is bounded using Holder's inequality.
- Using all the results from the previous steps, bound over joint probability is found.