PCA + HMM

Frances Ding, Charles Liu, Mark Goldstein

April 2017

1 Principal Component Analysis

1.1 Motivation

So far, in many of the supervised learning problems we've seen, we have tried to find rich features that increase the expressivity of our model. In practice this often involves using basis functions to transform our input into a higher dimensional space (eg. given data x, using x and x^2 as features, or using features learned by a neural network).

However, sometimes we actually want to reduce the dimensionality of our data for a number of reasons: fewer features lend themselves to easier interpretation (we might want to know why our model output a certain diagnosis, and only some of the thousands of details in patient records will be relevant); models with fewer features are easier to handle computationally; and our data might be arbitrarily high-dimensional because of noise, and we would like to access the lower-dimensional *signal* from the data. One method for dimensionality reduction through **linear projections** of the original data is PCA. When reducing the dimensionality of our data from m to d, PCA can be interpreted as minimizing the reconstruction loss of projecting data onto a d basis vectors, or as maximizing the variance in data that can be explained by d basis vectors.

1.2 Finding the lower dimensional representation

To perform PCA, we first calculate the normalized **feature covariance** matrix:

$$\mathbf{S} = (\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top) = \mathbf{X}^\top \mathbf{X}$$

We then decide how many dimensions d out of the original m that we want to keep in the final representation (for visualizations, often this will be $d = 2$ or $d = 3$). We then find the d largest eigenvalues of $\mathbf{S}$. The eigenvectors $(\mathbf{u}_1, \ldots, \mathbf{u}_d)$ corresponding to these eigenvalues will be our lower-dimensional basis. Thus, we reduce the dimensionality of a data point $\mathbf{x}$ by projecting it onto this basis - we combine the eigenvectors into the $d \times d$ matrix $\mathbf{U}$, and compute $\mathbf{Ux}$.

2 Hidden Markov Models

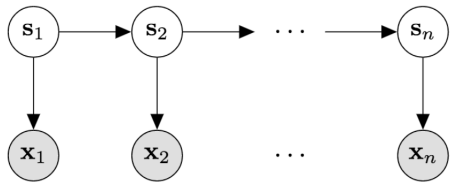
Last week in lecture, we covered Hidden Markov Models (HMMs). This model is useful for inferring a sequence of uknown states from a corresponding sequence of observed evidence (we call these observations, some sources call these "emissions"). Consider the following two examples:

1) Speech Recognition: Your N data points $\{\mathbf{x}^i\}_{i=1}^N$ are recordings of human speech, each segmented into n time steps. Given a sequence of audio information, you would like to infer the true sequence of discrete words that speaker i intended to communicate.

2) Facial Expression Recognition / Affective Computing: Each data entry is a sequence of photographs of a single person's face over a short period of time. Using these photo sequences as your observations, you would like to infer a sequence of associated underlying emotions/affects/moods that the human was experiencing.

Practical Note: notice that in the first example, we also have the problem of effectively segmenting the observation into time steps, while the second example's observations are already discrete over time.

2.1 HMM Graphical Model



Consider a sequence of one-hot encoded states $\mathbf{s}_1$,...,$\mathbf{s}_n$ where $\mathbf{s}_t \in \{S_k\}_{k=1}^c$ and a corresponding sequence of observations ($\mathbf{x}_1$,...,$\mathbf{x}_n$) where $\mathbf{x}_t \in \{O_j\}_{j=1}^m$. Each state can be one of c possible states, and each observation can be one of m possible observations. Note that N is the number of data points (each of which is a sequence), where n is the length of a sequence (assume all sequences are the same length). HMMs are characterized by, and help us reason about, the following joint distribution:

$$\begin{aligned} p(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n, \mathbf{x}_1, \ldots, \mathbf{x}_n) = \\ p(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n) p(\mathbf{x}_1, \ldots, \mathbf{x}_n \mid \boldsymbol{s}_1, \ldots, \boldsymbol{s}_n) = \\ p(\boldsymbol{s}_1) \prod_{t=1}^{n-1} p(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t) \prod_{t=1}^{n} p(\mathbf{x}_t \mid \boldsymbol{s}_t) \end{aligned}$$

Parameters:

- $\boldsymbol{\theta}$: distribution over initial states, $c \times 1$
- $\mathbf{T}$: $c \times c$ transition matrix such that t_{kj} is the probability of transitioning from S_k to S_j
- $\{\boldsymbol{\pi}\}_{k=1}^c$: state conditional observation probabilities such that $p(\mathbf{x}_t = O_j | \mathbf{s}_t = S_k; \{\boldsymbol{\pi}\}) = \pi_{kj}$. Each $\boldsymbol{\pi}_k$ is $m \times 1$.

Our goals are twofold. First, we need to estimate the parameters from the data. We will do this with a special variant of EM. This corresponds to training our model. Then, with our trained HMM, we are able to perform several inference tasks on our data: these include smoothing, prediction, and more (see below).

2.2 Properties of HMMs

- future is independent of past given present (Markov Property):

  $p(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_1, \ldots \boldsymbol{s}_t, \mathbf{x}_1, \ldots, \mathbf{x}_t) = p(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t)$

- observations only depend on present:

  $p(\mathbf{x}_t \mid \boldsymbol{s}_1, \ldots, \boldsymbol{s}_t, \mathbf{x}_1, \ldots, \mathbf{x}_{t-1}) = p(\mathbf{x}_t \mid \boldsymbol{s}_t)$

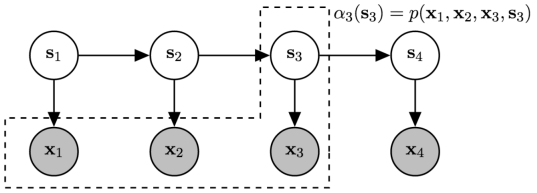
2.3 EM for HMMs: finding parameters with the Baum-Welch Algorithm

The Training Problem: given data points $\{\mathbf{x}^i\}_{i=1}^N$ defined by sequences $(x_1^i, \ldots, x_n^i)$ of length n represented as row vectors, we must infer the parameters $\{\mathbf{T}, \boldsymbol{\theta}, \{\boldsymbol{\pi}_k\}\}$. Had we been given the true states, we could easily compute joint probability $p(\mathbf{x}^i, \mathbf{s}^i)$ and write the complete-data log likelihood, and maximize with respect to the parameters. Instead, we must proceed by estimating state distributions and parameters iteratively.

2.3.1 Forward-Backward

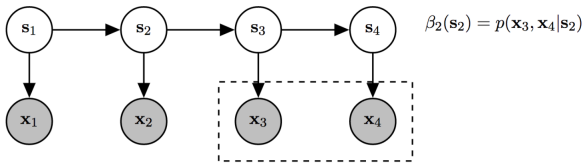
The HMM model is characterized by the joint distribution $p(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n, \mathbf{x}_1, \ldots, \mathbf{x}_n)$, which means that many of our training and inference tasks are an issue of marginalizing to obtain conditionals. Thus, naive algorithms can be expensive (lots of nested summations over states). EM for HMMs using the efficient Forward-Backward inference algorithm is called the Baum-Welch algorithm. Following the algorithm, we define the recurrence relations $\alpha_t(\boldsymbol{s}_t)$ and $\beta_t(\boldsymbol{s}_t)$ in the E step:

- $\alpha_t(\mathbf{s}_t)$ represents the joint probability of observations $1, \ldots, t$ and state t. α_t can be defined in terms of α_{t-1}. Move **forwards** through the sequence to calculate the α's
- $\beta_t(\mathbf{s}_t)$ represents the joint probability of observations $t+1, \ldots, n$ conditioned on state t. β_t can be defined in terms of β_{t+1}. Move **backwards** through the sequence to calculate the β's.



$$\forall \mathbf{s}_3 : \quad \alpha_3(\mathbf{s}_3) = p(\mathbf{x}_3 \mid \mathbf{s}_3) \sum_{\mathbf{s}_2} p(\mathbf{s}_3 \mid \mathbf{s}_2) \alpha_2(\mathbf{s}_2)$$

(a) alpha



$$\forall \mathbf{s}_2 : \quad \beta_2(\mathbf{s}_2) = \sum_{s_3} p(\mathbf{s}_3 \mid \mathbf{s}_2) p(\mathbf{x}_3 \mid \mathbf{s}_3) \beta_3(\mathbf{s}_3)$$

(b) beta

Remember: the probabilities used in the alpha/beta definitions come from the parameters that we consider fixed in the E step!

$$\forall \mathbf{s}_t : \quad \alpha_t(\mathbf{s}_t) = \begin{cases} p(\mathbf{x}_t \mid \mathbf{s}_t) \sum_{\mathbf{s}_{t-1}} p(\mathbf{s}_t \mid \mathbf{s}_{t-1}) \alpha_{t-1}(\mathbf{s}_{t-1}) & \text{if } 1 < t \leq n \\ p(\mathbf{x}_1 \mid \mathbf{s}_1) p(\mathbf{s}_1) & \text{o.w.} \end{cases}$$

$$\forall \mathbf{s}_t : \quad \beta_t(\mathbf{s}_t) = \begin{cases} \sum_{\mathbf{s}_{t+1}} p(\mathbf{s}_{t+1} \mid \mathbf{s}_t) p(\mathbf{x}_{t+1} \mid \mathbf{s}_{t+1}) \beta_{t+1}(\mathbf{s}_{t+1}) & \text{if } 1 \leq t < n \\ 1 & \text{o.w.} \end{cases}$$

2.3.2 Inference Patterns with $\boldsymbol{\alpha}, \boldsymbol{\beta}$

The following patterns are useful for inference with a trained HMM, but also in the E step during training (basically, any time we are considering the parameters fixed):

- $\alpha_t(\mathbf{s}_t)\beta_t(\mathbf{s}_t) = p(\mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{s}_t) \propto p(\mathbf{s}_t | \mathbf{x}_1, \ldots, \mathbf{x}_n)$
- joint of observations: $p(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \sum_{\mathbf{s}_t} \alpha_t(\mathbf{s}_t)\beta_t(\mathbf{s}_t)$ (for any t)
- smoothing: $p(\mathbf{s}_t \,|\, \mathbf{x}_1, \ldots, \mathbf{x}_n) \propto p(\mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{s}_t) = \alpha_t(\mathbf{s}_t)\beta_t(\mathbf{s}_t)$
- prediction: $p(\mathbf{x}_{n+1} \,|\, \mathbf{x}_1, \ldots, \mathbf{x}_n) \propto \sum_{\mathbf{s}_n, \mathbf{s}_{n+1}} \alpha_n(\mathbf{s}_n) p(\mathbf{s}_{n+1} \,|\, \mathbf{s}_n) p(\mathbf{x}_{n+1} \,|\, \mathbf{s}_{n+1}) \beta_n(\mathbf{s}_n)$
- transition: $p(\mathbf{s}_t, \mathbf{s}_{t+1} \,|\, \mathbf{x}_1, \ldots, \mathbf{x}_n) \propto \alpha_t(\mathbf{s}_t) p(\mathbf{s}_{t+1} \,|\, \mathbf{s}_t) p(\mathbf{x}_{t+1} \,|\, \mathbf{s}_{t+1}) \beta_{t+1}(\mathbf{s}_{t+1})$

2.3.3 E step

For sequence $\mathbf{x}^i$ and a fixed set of parameters $\mathbf{w} = \{\mathbf{T}, \boldsymbol{\theta}, \{\boldsymbol{\pi}_k\}\}$, we estimate the state distribution for $\mathbf{s}_1^i, \ldots, \mathbf{s}_n^i$ given $\mathbf{x}^i$. Let the $c \times 1$ vector $\mathbf{q}_t^i = (q_{t1}^i, \ldots, q_{tc}^i)$ represent $\mathbf{x}^i$'s distribution over states for time t under the current parameters. Let $\mathbf{Q}_{t,t+1}^i$ be the $c \times c$ matrix of transition probabilities under the current parameters.

- α's and β's are defined in terms of fixed parameters.
- $\mathbf{q}$'s defined in terms of α's and β's
- Calculate $q_{tk}^i = p(\mathbf{s}_t^i = S_k | \mathbf{x}^i; \mathbf{w})$ for all t and k (use smoothing eq. just above)
- Calculate $q_{t,t+1,k,\ell}^i = p(\mathbf{s}_t^i = S_k, \mathbf{s}_{t+1}^i = S_\ell | \mathbf{x}^i; \mathbf{w})$ (use transition eq. just above)
- Compute the following $\hat{N}$'s in terms of $\mathbf{q}$'s:

$$\hat{N}_{1k} = \sum_{i=1}^{N} q_{1k}^i \text{ (first period)} \quad \text{and more generally} \quad \hat{N}_k = \sum_{i=1}^{N} \sum_{t=1}^{n} q_{tk}^i \text{ (all periods)}$$

$$\hat{N}_{-nk} = \sum_{i=1}^{N} \sum_{t=1}^{n-1} q_{tk}^i \text{ (without last period)}$$

$$\hat{N}_{k\ell} = \sum_{i=1}^{N} \sum_{t=1}^{n-1} q_{t,t+1,k,\ell}^i \text{ (transitions)}$$

$$\hat{N}_{kj} = \sum_{i=1}^{N} \sum_{t=1}^{n} q_{tk}^i x_{tj}^i \text{ (observations)}$$

2.3.4 M step

Update parameters to maximize the expected complete-data log likelihood $\mathbb{E}_{\mathbf{S}}[\ln p(\mathbf{x}, \mathbf{S}; \mathbf{w})]$. Using the $\hat{N}$'s, we update the parameters in the following way:

$$\hat{\theta}_k = \frac{\hat{N}_{1k}}{N} \quad \hat{\pi}_{kj} = \frac{\hat{N}_{kj}}{\hat{N}_k} \quad \hat{t}_{k\ell} = \frac{\hat{N}_{k\ell}}{\hat{N}_{-nk}}$$

1. **PCA**

You are given the following data set:

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad \mathbf{x}_3 = \begin{bmatrix} -2 \\ -1 \end{bmatrix}$$

You would like to use PCA to find a 1-dimensional representation of the data.

(a) Plot the data set.

(b) Compute the feature covariance matrix $\mathbf{S}$.

(c) You find that $\mathbf{S}$ has eigenvector $[-1\ 1]^\top$ with eigenvalue 3 and eigenvector $[1\ 1]^\top$ with eigenvalue 9. What is the (normalized) basis vector $\mathbf{u}_1$ of your 1-dimensional representation? Add the basis vector $\mathbf{u}_1$ to your plot.

(d) Compute the coefficients z_1, z_2, z_3. Add the lower-dimensional representations $z_1\mathbf{u}_1, z_2\mathbf{u}_1, z_3\mathbf{u}_1$ to your plot. Based on your plot, what is the relationship between $z_i\mathbf{u}_1$ and $\mathbf{x}_i$ with respect to the new basis?

(e) Based on your plot, what would happen if you chose the unused eigenvector to be your basis vector?

2. **When to Use HMM's (Source: CMU)**

For each of the following scenarios, is it appropriate to use a Hidden Markov Model? Why or why not? What would the observed data be in each case, and what would the hidden states capture?

(a) Stock market price data
(b) Recommendations on a database of movie reviews
(c) Daily precipitation data in Boston
(d) Optical character recognition for identifying words

3. **Parameter Estimation in Supervised HMMs**

You are trying to predict the weather using an HMM model. The hidden states of the HMM are the weather of the day, which may be sunny or rainy, and the observable states are the color of the clouds, which can be white or gray. You have data on the weather and clouds from one sequence of four days (note: you have observed the hidden states too!):

Day	Weather	Clouds
1	Sunny	White
2	Rainy	Gray
3	Rainy	Gray
4	Sunny	Gray

(a) Draw a graphical model representing the HMM.

(b) Write out the values of N, n, c and of the one-hot vectors $\mathbf{s}_1^1, \ldots, \mathbf{s}_4^1, \mathbf{x}_1^1, \ldots, \mathbf{x}_4^1$.

(c) Estimate and interpret the values of the parameters $\boldsymbol{\theta}, \mathbf{T}, \{\boldsymbol{\pi}_k\}_{k=1}^c$ using the MLE estimators for the supervised HMM:

$$\hat{\theta}_k = \frac{N_{1k}}{N}, \quad \hat{t}_{kl} = \frac{N_{kl}}{N_{-nk}}, \quad \hat{\pi}_{kj} = \frac{N_{kj}}{N_k}$$

$$N_k = \sum_{i=1}^{N} \sum_{t=1}^{n} s_{tk}^i, \quad N_{1k} = \sum_{i=1}^{N} s_{1,k}^i, \quad N_{-nk} = \sum_{i=1}^{N} \sum_{t=1}^{n-1} s_{tk}^i$$

$$N_{kl} = \sum_{i=1}^{N} \sum_{t=1}^{n-1} s_{t,k}^i s_{t+1,l}^i, \quad N_{kj} = \sum_{i=1}^{N} \sum_{t=1}^{n} s_{tk}^i x_{tj}^i$$

4. **EM for HMMs**

You are trying modeling a toy's state using an HMM. At each time step, the toy can be active (state 1) or inactive (state 2), but you can only observe the color of the indicator light, which can be red (observation state 1) or green (observation state 2). You have collected data from one sequence:

Time	Light
1	Green
2	Red
3	Green

You initialize your EM with $\boldsymbol{\theta} = [\frac{1}{2}\ \frac{1}{2}]^\top, \mathbf{T} = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} \end{bmatrix}, \boldsymbol{\pi}_1 = [\frac{1}{4}\ \frac{3}{4}]^\top, \boldsymbol{\pi}_2 = [\frac{3}{4}\ \frac{1}{4}]^\top$.

(a) Compute $\alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2, \beta_3$ for the forward-backward algorithm using the initial parameter values.

(b) How is $\mathbf{q}_t^1$ defined? Compute the values of $\mathbf{q}_1^1, \mathbf{q}_2^2$ using the α and β values.

(c) How is $\mathbf{Q}_{t,t+1}^1$ defined? Compute the value of $\mathbf{Q}_{1,2}^1$ using the α and β values.

During EM, at one point you obtain the following values after the E step:

$$\mathbf{q}_1^1 = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} \end{bmatrix}^\top, \quad \mathbf{q}_2^1 = \begin{bmatrix} \frac{1}{3} & \frac{2}{3} \end{bmatrix}^\top, \quad \mathbf{q}_3^1 = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} \end{bmatrix}^\top, \quad \mathbf{Q}_{1,2}^1 = \begin{bmatrix} \frac{1}{6} & \frac{1}{2} \\ \frac{1}{6} & \frac{1}{6} \end{bmatrix}, \quad \mathbf{Q}_{2,3}^1 = \begin{bmatrix} \frac{1}{6} & \frac{1}{6} \\ \frac{1}{2} & \frac{1}{6} \end{bmatrix}$$

(d) Use the above values to compute $\hat{N}_k, \hat{N}_{kl}, \hat{N}_{kj}$.

(e) Complete the M step by updating the parameters $\boldsymbol{\theta}, \mathbf{T}, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2$.