Notes on matrix/spectral views of HMM's

probably has errors so look out

Brendan O'Connor

May 18, 2011

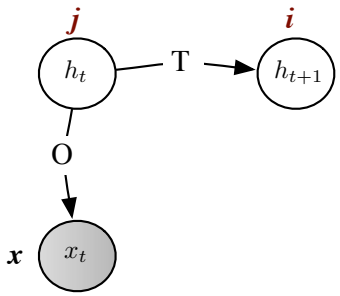
1 Definitions and lemmas

For the Hsu et al. setting, same notation as Siddiqi et al. Their convention: columns for conditioning variable. Lemma references are for Hsu, but have equivalents in Siddiqi. A little linear algebra review is at end of these notes.

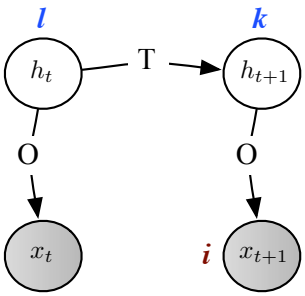
n observation types			
m hidden state classes; $m \leq n$			
$T \equiv [\ P(h_{t+1} = i \mid h_t = j)\ ]_{ij}$	$(m \times m)$		
$O \equiv [\ P(x_t = i \mid h_t = j)\ ]_{ij}$	$(n \times m)$		
$A_x \equiv [\ P(h_{t+1} = i \mid h_t = j)\ P(x_t = x \mid h_t = j)\ ]_{ij}$	$(m \times m)$		
$\equiv T\ \text{diag}(O_{1,x}, O_{2,x}\ ...\ O_{n,x}) = [\ T_{ij} O_{xj}\ ]_{ij}$			
$[P_1]_i \equiv P(x_1 = i)$	(n)	unigrams	
$[P_{2,1}]_{ij} \equiv P((x_2, x_1) = (i, j))$	$(n \times n)$	bigrams	
$[P_{3,x,1}]_{ij} \equiv P((x_3, x_2, x_1) = (i, x, j))$	$(n \times n)$	skip bigrams around x	
$P_1 = \vec{1}_m^\mathsf{T} T \text{diag}(\vec{\pi}) O^\mathsf{T} = \left[\sum_k^m \sum_l^m T_{kl} \vec{\pi}_l O_{il}\right]_i$		marg'ize hiddens	
$P_{2,1} = OT\text{diag}(\vec{\pi})O^\mathsf{T} = \left[\sum_k^m \sum_l^m O_{ik} T_{kl} \vec{\pi}_l O_{jl}\right]_{ij}$		marg'ize hiddens, see diagram	
$O = P_{2,1}(T\ \text{diag}(\vec{\pi})O^\mathsf{T})^+$			
$U \equiv$ left singular vectors of $P_{2,1}$	$(n \times m)$		
$\vec{b_1} \equiv U^\mathsf{T} P_1$	(m)		
$= (U^\mathsf{T} O)\vec{\pi}$		L3	
$\vec{b}_\infty \equiv (P_{2,1}^\mathsf{T} U)^+ P_1$	(m)		
$= 1_m (U^\mathsf{T} O)^{-1}$		L3	
$B_x \equiv (U^\mathsf{T} P_{3,x,1})(U^\mathsf{T} P_{2,1})^+$	$(m \times n)$		
$= (U^\mathsf{T} O) A_x (U^\mathsf{T} O)^{-1}$		L3	
$\vec{b}_t \equiv \frac{B_{x_{t-1:1}} b_1}{b_\infty^\mathsf{T} B_{x_{t-1:1}} b_1}$	(m)	observable repr. of state	
$\vec{b}_{t+1} = \frac{B_{x_t} b_t}{b_\infty^\mathsf{T} B_{x_{t-1:1}} b_t}$	(m)	L4	
$\vec{h}_t \equiv [\ P(h_t = i \vert x_{1:t-1})\ ]_i$	(m)	hidden state vector	
$\vec{b}_t = (U^\mathsf{T} O)\vec{h}_t$	(m)	L4: relationship to hidden state	
$U\vec{b}_t = O\vec{h}_t = [\ P(x_t = i \mid x_{1:t-1})\ ]_i$	(n)	R5	
$P(x_{1:t}) = \vec{b}_\infty^\mathsf{T} B_{x_{1:t}} \vec{b}_1$		L3	

2 Matrices as graph fragments

Here are graph fragments illustrating A_x and $P_{2,1}$, marking the random variable's values — i.e. the matrix indices.



$$[A_x]_{ij} = T_{ij} O_{xj}$$



$$\begin{aligned}
[P_{2,1}]_{ij} &= P((x_{t+1}, x_t) = (i, j)) \\
&= \sum_{k,l}^{m} P(x_{t+1:t} = (i, j), h_{t+1:t} = (k, l)) \\
&= \sum_{k,l}^{m} O_{ik} T_{kl} \pi_l [O^\mathsf{T}]_{lj}
\end{aligned}$$

3 A_x

A_x is called the "observation operator". We can think of it as the backward probability computation operator, and A_x^T is the forward probability computation operator. If you multiply by A_x^T, you compute the next marginal over hidden states accounting for the previous observation, summing out the possible previous states.[1] Using the transpose lets us restate Lemma 1 as right multiplications:

$$P(x_1, x_2, \ \ldots\ , x_t) = \vec{\pi} A_{x_1}^\mathsf{T} A_{x_2}^\mathsf{T} \ \ldots\ A_{x_t}^\mathsf{T} 1_m$$

Their Lemma 1 writes it as $\vec{1}_m A_{x_t} \ldots A_{x_1} \vec{\pi}$ (abbreviated $\vec{1}_m A_{x_{t:1}} \vec{\pi}$) so successive left-multiplication corresponds to the forward algorithm. Right-multiplication by A_x is the backward algorithm: compute previous timestep's marginal by summing out possible next timestep states, integrating against the previous timestep's conditional observation likelihood.

[1] Though, if I look it up I think the standard definition for forward probabilities uses O_{xi} not O_{xj}. But you can define either way to get a legitimate marginal probability of the observations.

4 $P_{2,1}$ and its SVD

Under the HMM, what's the bigram probability distribution? You can marginalize out the hidden variables like so:

$$[P_{2,1}]_{ij} = P((x_2, x_1) = (i, j)) = \sum_{k,l}^{m} O_{ik} T_{kl} \vec{\pi}_l O_{jl}$$

$P_{2,1}$ is rank m as noted in Lemma 2. Decompose as

$$\begin{array}{ccccc} P_{2,1} & = & U & \Lambda & V^\mathsf{T} \\ (n \times n) & & (n \times m) & (m \text{ diag}) & (m \times m) \end{array}$$

So a single bigram probability can be written more simply as

$$P((x_2, x_1) = (i, j)) = \sum_{z}^{m} \lambda_z U_{zi} V_{zj}$$

A possible interpretation: The right token i contributes a latent $U_{\cdot,i}$ vector, and the left token j contributes a latent $V_{\cdot,j}$ vector. The similarity (inner product) of the latent vectors tells you how compatible the words are — i.e. their bigram probability.

We can also view the HMM marginal bigram probability in this way, in which there are m^2 dimensions of possible compatibilities between i and j. But there isn't a single simple latent vector for each token; that's what the factorization of $P_{2,1} = OT\text{diag}(\vec{\pi})O^\mathsf{T}$ is for.

(What I was tempted to say was, you use one latent vector to project into a latent space, and the other to project down into the probabilities of the other token. But that's more like SVD on the conditional, not joint, bigram probabilities, and perhaps a little like the Saul and Pereira transition hidden class story instead of an HMM. Hm.)

Note the learning algorithm only uses U.

5 U, $U^\mathsf{T}O$ and HMM vs. observable representations

Hsu's Condition 2 says $U^\mathsf{T}O$ must be invertible. "In other words, U defines an m-dimensional subspace that preserves the state dynamics. A natural choice for U is the thin SVD of $P_{2,1}$."

$U^\mathsf{T}O$ is important. It's the projection operator between the observable and HMM representations of state.

$\vec{b}_t, B_x$ work in the observable representation, while $\vec{h}_t, A_x$ work in the HMM representation.

- $\vec{b}_t$ is the observable representation of state, while $\vec{h}_t$ is the HMM representation of state.

- B_x is the observation operator in the observable representation, while A_x is the observation operator in the HMM representation.

$\vec{b}_1 \equiv U^\mathsf{T} P_1 = [\sum_i^n U_{iz} P(x_t = i)]_z$ ... what is this? Expected value of projecting into the observable-repr-state z, expected over empirical unigram distribution? Then they prove also $\vec{b}_1 = (U^\mathsf{T} O)\vec{\pi}$ or more generally $\vec{b}_t = (U^\mathsf{T} O)\vec{h}_t$ which says how to transform between the HMM state and observation-representation state.

For the relationship between B_x and A_x, the Siddiqi et al. slides call $U^\mathsf{T} O$ the "similarity transform of the true HMM parameter A_x" because $B_x = (U^\mathsf{T} O) A_x (U^\mathsf{T} O)^{-1}$.

6 Reduced rank HMM

The twist in Siddiqi et al. is to say T is low rank k, so it decomposes as

$$\underset{(m \times m)}{T} = \underset{(m \times k)}{R} \underset{(k \times m)}{S}$$

This means their model is a two-layer model: from observation type space of n symbols, to the state transition space of m states, to their latent space of k dimensions. This is their Figure 1(B).

Because T is rank k, the generative process ensures that $P_{2,1}$ is only rank k as well. Therefore U is $(n \times k)$ in the Siddiqi et al. model.

7 Actual algorithm

In either the reduced rank or normal case, once you compute U, you compute $\vec{b}_1, \vec{b}_\infty, B_x$ and then recursively apply them to get $P(x_1..x_t)$ and/or $P(x_t|x_1..x_{t-1})$ as you like. I'm wondering if you can use the observable-represenations to back out the original HMM via a linear transform... but you don't know O so can't compute $U^\mathsf{T} O$ maybe. The ArXiV version of Siddiqi et al. mentions this at the very end, calling it the "positive identification problem." They mention a current technique for doing it is unstable but don't explain why.

8 Theoretical guarantees

This is the big attraction — this method estimates $P(x_1...x_t)$ with arbitrarily high accuracy with more data. EM does not have this guarantee. (Does MCMC? Tricky question. (1) Infinite computation time says MCMC gives correct posteriors for the data. Does that hold for these margnals too? (2) Spectral HMM doesn't need infinite or even high computation time. Count up the the n-gram matrices, do some SVD and a few inversions... very low-order polynomial.)

I'm more interested in whether it recovers the parameters of the HMM — T and O — correctly (or rather, correctly up to the linear $U^\mathsf{T}O$ transform) with more data. I would think this is more like the usual notion of statistical consistency, as opposed to probabilities of x. It's the more usual setting for unsupervised POS tagging and the like.

Note in their applications (state space tracking, video modeling), they're quite happy just to have a better $P(x_t|x_1..x_{t-1})$ model. The reduced rank method is a big win for them because there is a much lower dimensional manifold in their problems that they can exploit.

9 Relationship to Schuetze etc.

Turney and Pantel 2010 have a review of matrix factorization approaches to distributional similarity. The versions where rows are words and columns are contexts ... are a lot like the $P_{2,1}$ matrix. I think the early 90's Schuetze work may have been the first of this sort of thing.

10 Linear algrebra stuff

Sources: Wikipedia and Matrix Cookbook

10.1 SVD

(Most any) matrix can be written out as a singular value decomposition. If X is full rank and has $m < n$, it's

$$\underset{(n \times m)}{X} = \underset{(n \times m)}{U} \underset{(m \times m)}{\Lambda} \underset{(m \times m)}{V^\mathsf{T}}$$

Where $\Lambda = \text{diag}(\lambda_1 ... \lambda_m)$, the singular values.

If X is only rank $k < m$, that means you only need a k-dimensional latent space to perfectly reconstruct it. So the last $\lambda_{k+1} ... \lambda_m$ eigenvalues will be zero, and the $k + 1..m$ columns of U and V are irrelevant. So the SVD is just

$$\underset{(n \times m)}{X} = \underset{(n \times k)}{U} \underset{(k \times k)}{\Lambda} \underset{(k \times m)}{V^\mathsf{T}}$$

The reconstruction of an element X_{ij} uses a dot-product of column of U against a column of V — called the left and right singular vectors — scaled by Λ.

$$X_{ij} = \sum_{z=1..k} \lambda_z U_{zi} V_{zj}$$

Note that if all you want to do is come up with a factorization of X into two matrices, you can multiply the singular values into either U or V (or a bit of both). Scaling out the singular values in the SVD makes the columns of U and V be orthogonal and unit-norm.

10.2 Moore-Penrose pseudoinverse (A^+)

It's a generalization of the matrix inverse. For any matrix A,

- There are 0 or 1 vanilla inverses A^{-1}.
  def: $AA^{-1} = A^{-1}A = I$
- There are 1 or more generalized inverses A^-.
  def: $AA^-A = A$
- There is exactly 1 Moore-Penrose pseudoinverse A^+.
  def: $AA^+A = A$ and $A^+AA^+ = A^+$,
  and both A^+A and AA^+ are symmetric.

11 References

Hsu, Kakade, Zhang. A Spectral Algorithm for Learning Hidden Markov Models. COLT 2009.
Sajid M. Siddiqi, Byron Boots, Geoffrey J. Gordon. Reduced-Rank Hidden Markov Models. AISTATS 2010. Also see ArXiV version (has more proofs), and slides on Byron Boots' website.