Object recommendation (motivation for collaborative filtering)

- An important real-world/commercial problem is that of matching consumers to products
- Make connections using user feedback about subsets of products e.g. Netflix movie ratings, Amazon ratings of products, Yelp, YouTube etc.
- Recommendation systems use this info to make suggestions to customers using user provided information.

Content, collaborative and location-based collaborative filtering methods

Content filtering:

- use 'known' information about products and users to make recommendations
- creates profile for each user or product to characterise:-
  - Product - movie info, price info, product descriptions
  - Users - demographic information, questionnaire info
- Example - Pandora using "Music Genome Project" - expert scores, user music preference information; recommendations on pairing these (i.e. not using listening behaviour)
- uses external information gathering - expensive

Collaborative filtering:-

A different strategy for object recommendation:-
only use past user inputs/behaviour to make recommendations. Ignore 'a priori' user or object information.
Analyses relationships between users and interdependencies among objects to identify new user-object associations
Supposedly 'domain-free': in that subjectivity considerations about interpreting data is backgrounded (but re-emerges through mathematical formalisms)
Accuracy vs cold-start problem in both filtering methods.
Two collaborative filtering methods - neighbourhood based and location/latent factor models.

1) Neighbourhood based — item based (evaluate user pref. for items based on ratings of 'neighbouring' items by same user. Product neighbours are products with similar ratings by same user)
   - user-based

identifies like-minded users who can complement each other's ratings; define similarity score between me and other users based on how much overlapping ratings agree; using scores, let others "vote" on your recommend.)
not mutually exclusive; content filtering can enhance collaborative filtering performance.

Location-based/latent factor models: explain ratings by characterising items AND users from (20-100) factors inferred from ratings patterns

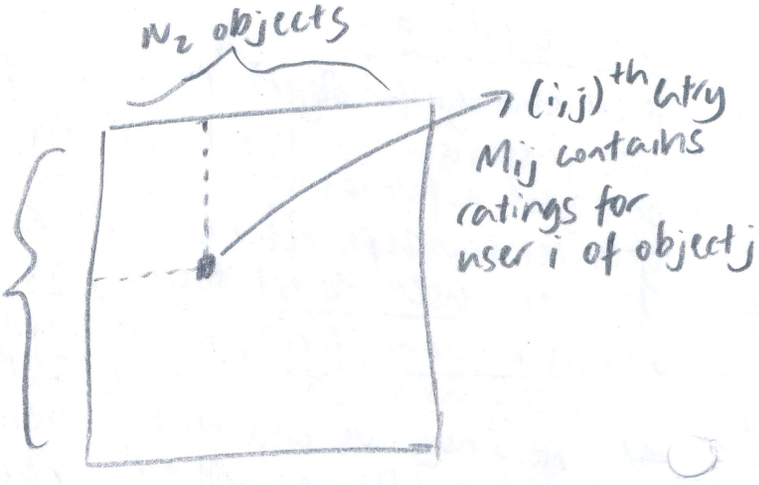
- Interpretable factors in movie context include highbrow, arthouse; or uninterpretable
- Each factor measures how much the user likes movies that score highly on that factor.
- (III) Location-based/latent factor models embed users and objects into points in $\mathbb{R}^d$.
- Recommendations are based on the proximity of user/object in latent embedding space (inferred (estimated) from data). Estimate embeddings via matrix factorisation

Matrix factorisation (MF) techniques

- most successful (according to Koren, Bell, Volinsky 2009) latent factor models are based on matrix factorisation (MF)
- ~~How~~ MF characterises items and users by vectors of factors inferred from item ratings patterns. High correspondence between item and user factors leads to a recommendation.
- Explicit vs implicit feedback
- MF gives a way to estimate user and object locations (by writing them as vectors in a latent space)

Technique:

- Form a ratings matrix M
- will have many missing values (sparse)
  i.e. $M_{ij} = \text{NaN}$ for many (i,j)
- Goal: Fill in these missing values (using appropriate statistical techniques)
- missing values because a single user is not a ratings machine.

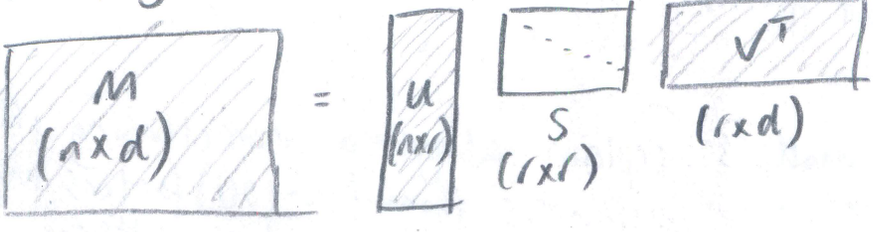


- Planned solution to goal: 1) Have a prediction of every missing rating for user i
  2) Recommend highly rated objects amongst predicted

(III): MF: Estimate a low-rank factorisation using observed data, ignoring unobserved data.

SVD:

- One method for factorising M is singular value decomposition (analogous to eigen-value/eigenvector decomposition but for non-~~invertible~~ square matrices)



Singular value decomposition:

- Every matrix M can be written as:

$$M = USV^T \quad \text{with} \quad U^TU = I, \quad V^TV = I$$

$S = \text{diag}(S_{ii})$ with $S_{ii} \geq 0$

- U - orthonormal columns
- V - orthonormal columns

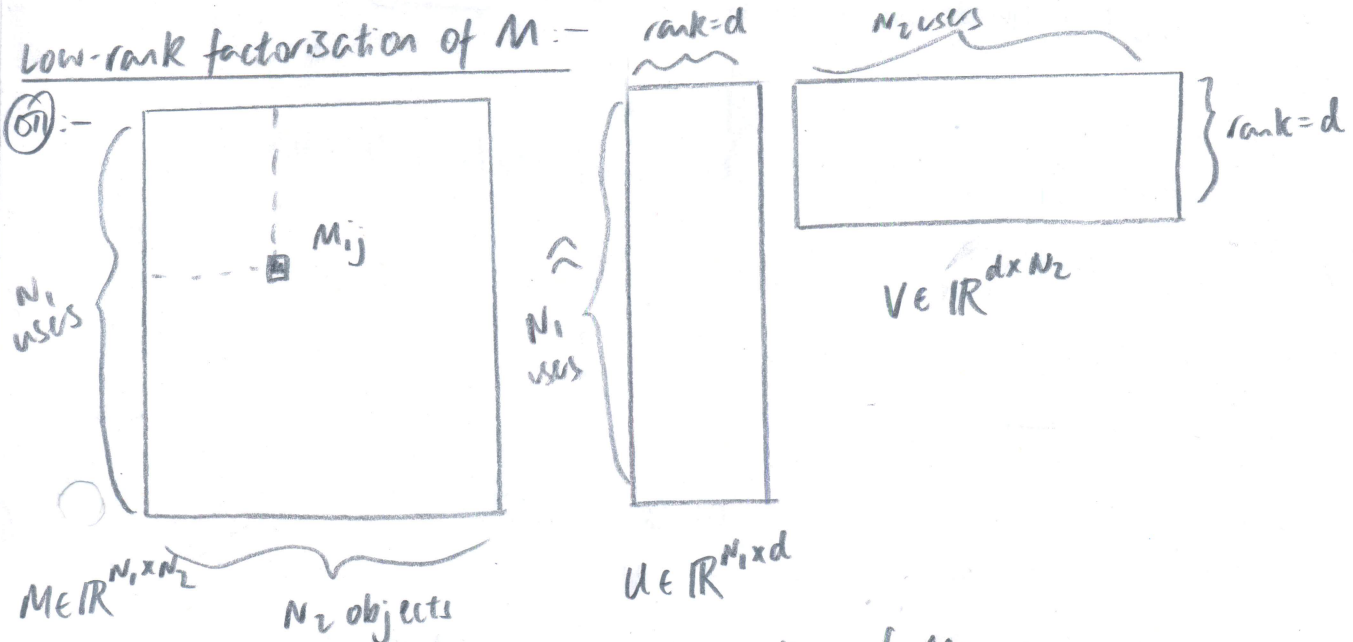
$M \in \mathbb{R}^{n \times d}$, $U \in \mathbb{R}^{n \times r}$, $S \in \mathbb{R}^{r \times r}$, $V^T \in \mathbb{R}^{r \times d}$

- Note that for orthonormal square matrices; $X^TX = I = XX^T$ $X \in \mathbb{R}^{n \times n}$; this does not hold for U and V (i.e. $UU^T \neq I$ and $V^T \neq I$) as they are not square.
- Note that r (which features in each of the decomposition matrices) ~~features~~ is the rank of M.

$$r = \text{rank}(M)$$

- Recall rank(M) is the size of the largest subset of columns that are linearly independent (constitute a linearly independent set)
- Recall if rank(X) = $\min(n, n)$ ($X \in \mathbb{R}^{n \times n}$) then it is full-rank.
- Recall r (rank of M) means it has fewer 'degrees of freedom'

Low-rank factorisation of M:-



- criteria on modelling low-rank factorisation of M:-
1. Account for M being sparse; many $M_{ij} =$ NaN
2. low-rank $\rightarrow$ $d \ll \min_{\text{vector}} \{N_1, N_2\}$ (e.g. $d \approx 10$)
3. Estimate a location $u_i \in \mathbb{R}^d$ for user i and $v_j \in \mathbb{R}^d$ for object j
- d corresponds to no. of factors (indication of latent things which are 'really considered' in a person's ratings)

Low-rank factorisation of M (semi-formal)

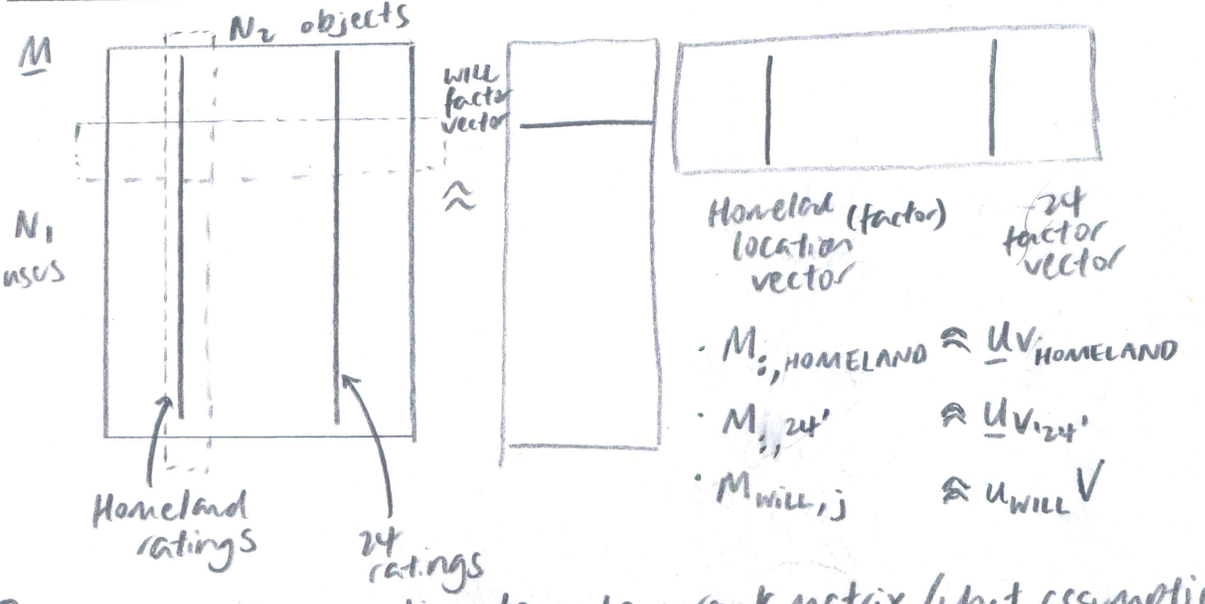
- An MF model models map both users and items to a joint latent factor space of dimensionality d; here determined by low-rank properties (rank(M))
- user-item interactions are modelled as inner-products within that space.

- Each user i is associated with a vector $u_i \in \mathbb{R}^d$ (indicating user-location)
- Each object j is associated with a vector $v_j \in \mathbb{R}^d$ (indicating object-location)
- For a given user i, the elements of u_i measure the extent to which the item possesses those factors, positive or negative.
- For a given object j, the elements of v_j measure the extent of interest the user has in items that are high in the corresponding factors, positive or negative.
- The dot product $u_i^T v_j$ captures the interaction between user i and object j, the user's overall interest in the item's characteristics (j (rating))
- This approximates the user's (i) ratings of objects (j), leading to the estimate

$M_{ij} \approx u_i^T v_j$ or equivalently, $\hat{M}_{ij} \triangleq u_i^T v_j$ (OTI)

- The challenge consists in computing the mapping of each user i and object j to factor vectors $u_i, v_j \in \mathbb{R}^d$, estimated with above.
- conventional SVD (for latent semantic factors in information retrieval) impossible factorisation of M that is rendered difficult due to sparsity; in fact SVD is undefined.
- Carelessly addressing missing values $\Rightarrow$ overfitting
- Missing data imputation :- computationally expensive, data distortion

Low-rank matrix factorisation - intuition :-



(Q) Why should we estimate a low-rank matrix / what assumptions are we relying on

- Imagine 2 movies Homeland, 24 (similar)
- In the ratings matrix M, as it is sparse, a subset of N_1 users will have rated Homeland and another subset rated 24. (with ratings in the row of each column vector $M_{:,HOMELAND}$ and $M_{:,'24'}$ in ratings matrix M)

and one user rated one but not the other)

- As movies are similar (how you want to interpret the factors i.e. spy thriller, movie post-2000 is up to you); the ratings across the columns $M_{:,\text{HOMELAND}}$ and $M_{:,\text{'24'}}$ will be highly correlated
- low-rank factorisations enforce this correlation (via restrictions on all objects and users to be in a (smaller) latent embedding space $\mathbb{R}^d$)
- Amounts to a restriction on type of matrices U and V.
- low-rank also means that the N_1-dimensional columns do not 'fill up' $\mathbb{R}^{N_1}$ (formally the span of the column vectors in ratings matrix is not $\mathbb{R}^{N_1}$ which is the situation under full-rank; the low-rank nature means there are fewer linearly independent column vectors; together spanning a region of $\mathbb{R}^{N_1}$ that is significantly smaller i.e. $\mathbb{R}^d$! (our latent embedding space)
- Recall the span of a set of vectors $\{x_1, \ldots, x_n\}$ is the set of all vectors that can be expressed as a linear combination of $\{x_1, \ldots, x_n\}$ i.e.

$$\text{span}(\{x_1, \ldots, x_n\}) = \left\{ v : v = \sum_{i=1}^{n} \alpha_i x_i \;\; \alpha_i \in \mathbb{R} \right\}$$

If $\{x_1, \ldots, x_n\}$ constitutes a linearly independent set, then $\text{span}\{x_1, \ldots, x_n\} \in \mathbb{R}^n$

As >95% values missing, low-rank restriction/high-correlation factorisation allows us to 'borrow' information across movies to learn locations ~~predict~~ and eventually predict ratings and vice versa

This reliance on correlation is the basis for filling in data.

All correlations, all movies ⟶ predictions

Assuming same arguments can be made for rows of M (i.e. user oriented?)

Probabilistic matrix factorisation (PMF) - setup, inference procedure discussion

One model for learning a low-rank factorisation in missing data problem.

Setup:

Define set Ω containing the pairs (i,j) that are observed :-

$$\Omega = \{(i,j) : M_{ij} \text{ is measured } (\neq \text{NaN})\}$$

$(i,j) \in \Omega$ if user i rated object j

Define:

Ω_{u_i} - index set of objects rated by user i

Ω_{v_j} - index set of users who rated object j

(holes moving ways)

Generative model:

- For N_1 users and N_2 objects, assume the locations/factor vectors of users and objects in $\mathbb{R}^d$ are generated thus:-

User locations :- $u_i \sim N(0, \lambda^{-1}I)$ $i = 1, \dots, N_1$

Object " :- $v_j \sim N(0, \lambda^{-1}I)$ $j = 1, \dots, N_2$

} 'priors' zero-mean Gaussian with isotropic covariance

- Given these locations, the data distribution :-

$$M_{ij} \sim N(u_i^T v_j, \sigma^2) \quad \text{for each } (i,j) \in \Omega$$

Comments

- As M_{ij} is a rating, Gaussian is ~~wrong~~ wrong → only applies to continuous rvs
- Assumption still used as implementable and "data is ordinal which Gaussians can capture"

Inference procedure discussion - EM?

Q Should we use EM algorithm to estimate us and vs? As many missing values in M.

- Recall that having defined a generative, probabilistic model (distribution family) on data, we now attempt to infer its parameters (of model) using data.

Let M_o be observed, M_m be missing in ratings matrix M; then

$$p(M_o|U,V) = \int p(M_o, M_m|U,V)\, dM_m$$

- EM is merely a tool for maximising $p(M_o|U,V)$ over U and V
- Required when following criteria not:-

1) $p(M_o|U,V)$ difficult to maximise 2) $p(M_o, M_m|U,V)$ easy to work with

3) posterior $p(M_m|M_o, U, V)$ is known

A If $p(M_o|U,V)$ does not present issues for inferences (ie we can derive closed form updates for ML/MAP) → NO

- Similar conclusion in MAP scenario → maximising $p(M_o, U, V)$
- Test maximisation $p(M_o, U, V)$ over U, V → write joint log-likelihood and take derivatives wrt u_i and v_j

Joint likelihood:- $p(M_o, U, V) = \underbrace{\prod_{(i,j)\in\Omega} p(M_{ij}|u_i, v_j)}_{(i)} \times \underbrace{\left[\prod_{i=1}^{N_1} p(u_i)\right]}_{(ii)} \times \underbrace{\left[\prod_{j=1}^{N_2} p(v_j)\right]}_{(iii)}$

PMF - MAP solution, co-ordinate ascent, output

log-joint likelihood and MAP

$$u_{MAP}, V_{MAP} = \underset{U,V}{\operatorname{argmax}} \ln p(M_0, U, V)$$

$$= \underset{U,V}{\operatorname{argmax}} \ln \left\{ \prod_{(i,j)\in\Omega} p(M_{ij}|u_i, v_j) \times \left[\prod_{i=1}^{N_1} p(u_i)\right] \times \left[\prod_{i=1}^{N_2} p(v_j)\right] \right\}$$

$$= \underset{U,V}{\operatorname{argmax}} \sum_{(i,j)\in\Omega} \ln p(M_{ij}|u_i, v_j) + \sum_{i=1}^{N_1} \ln p(u_i) + \sum_{j=1}^{N_2} \ln p(v_j)$$

- Setting MAP objective function as $\mathcal{L}$, we want to maximise :-

$$\mathcal{L} = -\sum_{(i,j)\in\Omega} \frac{1}{2\sigma^2} \|M_{ij} - u_i^T v_j\|_2^2 - \sum_{i=1}^{N_1} \frac{\lambda}{2} \|u_i\|_2^2 - \sum_{j=1}^{N_2} \frac{\lambda}{2} \|v_j\|_2^2$$

(*) Derivation in comments

- To update each u_i and v_j, take gradients of $\mathcal{L}$ and set to zero :-

$$\nabla_{u_i} \mathcal{L} = \sum_{j\in\Omega_{u_i}} \frac{1}{\sigma^2} (M_{ij} - u_i^T v_j) v_j - \lambda u_i = 0$$

- Note symmetry (wlog of v_j and u_i)

(*) Derivations

$$\nabla_{v_j} \mathcal{L} = \sum_{i\in\Omega_{v_j}} \frac{1}{\sigma^2} (M_{ij} - v_j^T u_i) u_i - \lambda v_j = 0$$

- Solve for each u_i and v_j individually (EM not required)

(*)

$$u_i = \left(\lambda\sigma^2 I + \sum_{j\in\Omega_{u_i}} v_j v_j^T\right)^{-1} \left(\sum_{j\in\Omega_{u_i}} M_{ij} v_j\right)$$

$$v_j = \left(\lambda\sigma^2 I + \sum_{i\in\Omega_{v_j}} u_i u_i^T\right)^{-1} \left(\sum_{i\in\Omega_{v_j}} M_{ij} u_i\right)$$

- Not possible to solve for u_i and v_j simultaneously to find MAP solution
- As with K-means and GMM, use co-ordinate ascent.

comments/derivations:

- Write joint likelihood $p(M_0, U, V)$ as product over all observed ratings that we have in our dataset of the rating user i gives to object j (ie $(i,j)\in\Omega$) given user location (u_i) and object location (v_j) of conditional independent individual likelihoods $p(M_i|u_i, v_j)$ multiplied by priors on locations, $p(u_i)$ and $p(v_j)$.
- Again, using iid from Gaussian for (i) (ii) (iii) :-

$$p(M_0, U, V) = p(M_0|U, V)\, p(U)\, p(V) \quad \text{via probability rules}$$

$$= \prod_{(i,j)\in\Omega} p(M_{ij}|u_i, v_j) \prod_{i=1}^{N_1} p(u_i) \prod_{j=1}^{N_2} p(v_j) \quad \text{via iid}$$

(2) Derivation

$$U_{MAP}, V_{MAP} = \underset{U,V}{\operatorname{argmax}} \sum_{(i,j)\in\Omega} \ln p(M_{ij}|u_i, v_j) + \sum_{i=1}^{N_1} \ln p(u_i) + \sum_{j=1}^{N_2} \ln p(v_j) = \underset{U,V}{\operatorname{argmax}}\ \mathcal{L}$$

$$\mathcal{L} = \sum_{(i,j)\in\Omega} \ln\left\{\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}\left(M_{ij} - u_i^T v_j\right)^2\right)\right\} + \sum_{i=1}^{N_1} \ln\left\{\frac{1}{\sqrt{2\pi|\Sigma|}^{1/2}} \exp\left(-\frac{1}{2} u_i^T \Sigma^{-1} u_i\right)\right\}$$

$$+ \sum_{j=1}^{N_2} \ln\left\{\frac{1}{\sqrt{2\pi|\Sigma|}^{1/2}} \exp\left(-\frac{1}{2} v_j^T \Sigma^{-1} v_j\right)\right\}$$

$$= \sum_{(i,j)\in\Omega} -\frac{1}{2}\ln\left(2\pi\sigma^2\right) + \sum_{(i,j)\in\Omega} -\frac{1}{2\sigma^2}\left(M_{ij} - u_i^T v_j\right)^2 + \sum_{i=1}^{N_1} -\frac{1}{2}\ln\left(2\pi|\Sigma|\right) + \sum_{j=1}^{N_1} -\frac{1}{2} u_i^T \Sigma^{-1} u_i$$

$$+ \sum_{j=1}^{N_2} -\frac{1}{2}\ln\left(2\pi|\Sigma|\right) + \sum_{j=1}^{N_2} -\frac{1}{2} v_j \Sigma^{-1} v_j$$

- Setting $\Sigma = \lambda^{-1} I$, and noting that $|\Sigma| = \lambda^{-d}$ as $\det\left(\operatorname{diag}(\lambda_i)\right) = \prod_i \lambda_i$; $\left(\lambda^{-1} I\right)^{-1} = \lambda I$

$$\mathcal{L} = \sum_{(i,j)\in\Omega} -\frac{1}{2\sigma^2}\left\|M_{ij} - u_i^T v_j\right\|_2^2 + \sum_{i=1}^{N_1} -\frac{1}{2} u_i^T \left(\lambda^{-1} I\right)^{-1} u_i + \sum_{j=1}^{N_2} -\frac{1}{2} v_j^T \left(\lambda^{-1} I\right)^{-1} v_j$$

$$-\frac{1}{2}|\Omega|\ln\left(2\pi\sigma^2\right) + \sum_{i=1}^{N_1} -\frac{1}{2}\ln\left(2\pi\left|\lambda^{-1} I\right|\right) + \sum_{j=1}^{N_2} -\frac{1}{2}\ln\left(2\pi\left|\lambda^{-1} I\right|\right)$$

$$\mathcal{L} = -\sum_{(i,j)\in\Omega} \frac{1}{2\sigma^2}\left\|M_{ij} - u_i^T v_j\right\|_2^2 - \frac{\lambda}{2}\sum_{i=1}^{N_1}\|u_i\|_2^2 - \frac{\lambda}{2}\sum_{i=1}^{N_2}\|v_j\|_2^2 \quad \text{(QED)}$$

$$-\frac{1}{2}|\Omega|\ln\left(2\pi\sigma^2\right) - \left(\frac{N_1+N_2}{2}\right)\ln\left(2\pi\right) + d\left(\frac{N_1+N_2}{2}\right)\ln\left(\lambda\right)$$

constant wrt u_i, v_j ; $|\Omega|$ denotes cardinality of Ω

Algorithm: - MAP inference co-ordinate ascent algorithm

Input: An incomplete ratings matrix M, as indexed by set Ω. Rank d (dimensionality of latent space we want to embed users, objects into.

Output: N_1 user locations $u_i \in \mathbb{R}^d$ and N_2 object locations $v_j \in \mathbb{R}^d$

Initialise each v_j (e.g. by random generation) e.g. generate $v_j \sim N(0, \lambda^{-1}I)$

For each iteration, do

- For $i = 1, \dots, N_1$ update user location

$$u_i = \left(\lambda\sigma^2 I + \sum_{j \in \Omega_{u_i}} v_j v_j^T\right)^{-1} \left(\sum_{j \in \Omega_{u_i}} M_{ij} v_j\right)$$

- For $j = 1, \dots, N_2$ update object location

$$v_j = \left(\lambda\sigma^2 I + \sum_{i \in \Omega_{v_j}} u_i u_i^T\right)^{-1} \left(\sum_{i \in \Omega_{v_j}} M_{ij} u_i\right)$$

Predict that user i rates object j as $u_i^T v_j$ i.e. $\hat{M}_{ij} = u_i^T v_j$ rounded to closest rating option (for ratings $M_{ij} = 0$)

- Assess convergence through log of joint likelihood i.e. $\ln p(M_\Omega, U, V)$; evaluate after each iteration

Output

- E.g. with latent embedding $\mathbb{R}^2$ (2 factor vectors)
- users and objects each lie in $\mathbb{R}^2$
- Latent space embedding and proximities meaningful and interpretable though similarity (relations between $v_{HOMELAND}, v_{24}$) (correlation) in patterns
- E.g. 2 'apparently' similar series - Homeland and 24 → similar ratings from users
- PMF factorises sparse matrix M into a product of two $(N_1 \times d)$; $(d \times N_2)$ matrices

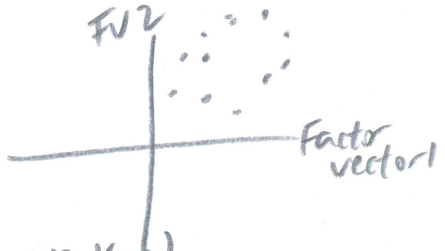
(i) If Homeland and 24 have similar ratings patterns, $M_{:,24}$ and $M_{:,HOMELAND}$ should be close in high-dimensional $\mathbb{R}^{N_1}$ space (i.e. low Euclidean norm).

As $M_{:,HOMELAND} \approx U v_{HOMELAND}$ and $M_{:,24} \approx U v_{24}$, then proximity of $M_{:,24}$ and $M_{:,HOMELAND}$ should imply proximity of $v_{HOMELAND}$ and v_{24} (note matrix U is same in both approximations)

(ii) correlation/similarity of $M_{:,HOMELAND}, M_{:,24}$ ⇒ proximity of $v_{HOMELAND}$ and v_{24} in $\mathbb{R}^d$
in $\mathbb{R}^{N_1}$

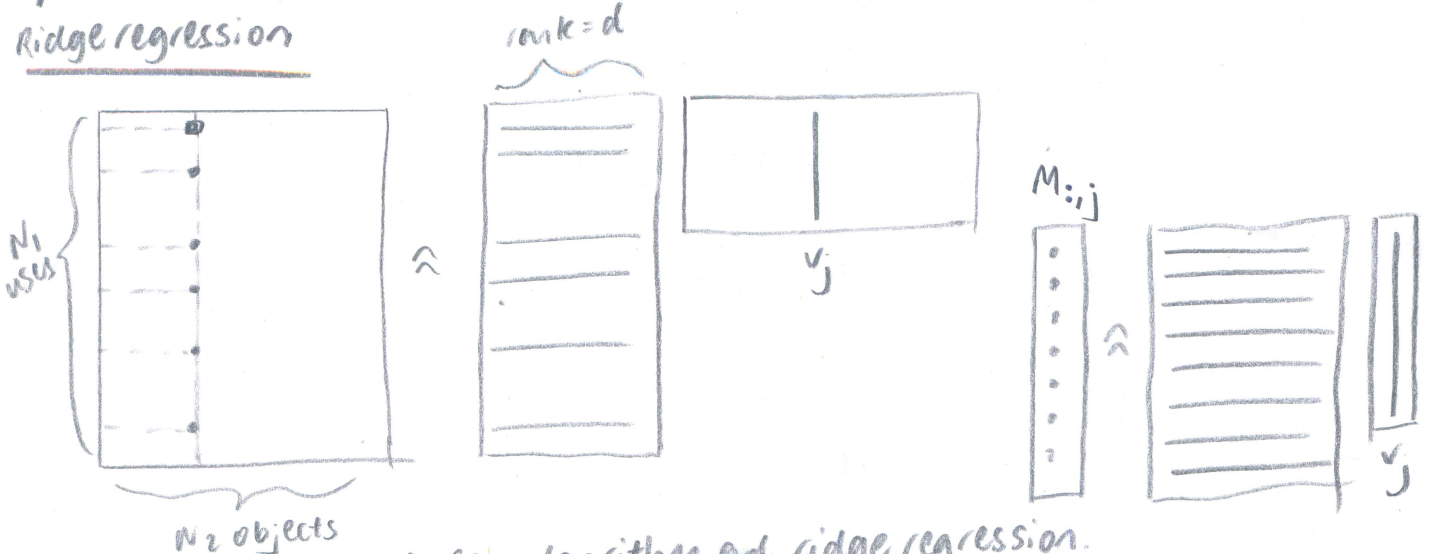
(iii) Errors due to Gaussian likelihood, corresponding to default noisy variation in two vectors $v_{HOMELAND}$ and v_{24}

(iv) Same arguments made about pairs of movies v_j and $v_{j'}$; and also users u_i and $u_{i'}$. (see earlier diagram)



MF and analogies between supervised learning methods - ridge regression and least squares

Ridge regression



- Close relationship between algorithm and ridge regression.
- Updating location/factor vector for object j (v_j)
- Minimise sum of squared error $\sum_{(i,j)\in\Omega} \frac{1}{\sigma^2}(M_{ij} - u_i^T v_j)^2$ with penalty $\lambda \|v_j\|^2$
- Ridge regression for v_j:-

$$v_j = \left(\lambda\sigma^2 I + \sum_{(i\in\Omega v_j)} u_i u_i^T\right)^{-1} \left(\sum_{(i\in\Omega v_j)} M_{ij} u_i\right)$$

- Model is a set of $N_1 + N_2$ coupled ridge regression problems
- For any particular user i that rated object j, we use user's location (u_i) as the covariate vector for that rating M_{ij}. M_{ij} is analogue of the output we want to predict for that particular rating.
- In our case, we do not have the covariates and we want to ~~estimate~~ estimate them from the distribution family defining our model.

Least squares

- Removing Gaussian priors on u_i and v_j; update for v_j is then:-

$$v_j = \left(\sum_{i\in\Omega v_j} u_i u_i^T\right)^{-1} \left(\sum_{i\in\Omega v_j} M_{ij} u_i\right)$$

(remove term that arises from regularisation $\lambda\sigma^2 I$)

- Sequence of updates for each column and row corresponds to maximising likelihood of $P(M_O|U,V)$ (conditional likelihood of observed values M_O on U and V) instead of maximising joint likelihood $p(M_O, U, V)$
- Without priors on U and V, require every single user has rated at least ~~least~~ d objects and every object has been rated more than d times.
- The above criteria is comes from the requirement of invertibility on $u_i u_i^T$ and in the case of updating u_i, $v_j v_j^T$.
- Bayesian approach is necessary for invertibility on every user (and object)

derivation (ii)

$$\mathcal{L} = -\sum_{(i,j)\in\Omega} \frac{1}{2\sigma^2}\|M_{ij} - u_i^T v_j\|_2^2 - \sum_{i=1}^{N_1}\frac{\lambda}{2}\|u_i\|_2^2 - \sum_{j=1}^{N_2}\frac{\lambda}{2}\|v_j\|_2^2 + C$$

$$\nabla_{u_i}\mathcal{L} = -\sum_{(j\in\Omega_{u_i})} \frac{1}{2\sigma^2}\nabla_{u_i}(M_{ij} - u_i^T v_j)^T(M_{ij} - u_i^T v_j) - \sum_{i=1}^{N_1}\frac{\lambda}{2}\nabla_{u_i} u_i^T u_i - \underbrace{\sum_{j=1}^{N_2}\nabla_{u_i} v_j^T v_j}_{=0} = 0$$

$$= -\sum_{(j\in\Omega_{u_i})} \frac{1}{2\sigma^2}\nabla_{u_i}\left(M_{ij}^2 - 2M_{ij}u_i^T v_j + v_j^T u_i u_i^T v_j\right) - \frac{\lambda}{2}(2u_i) = 0$$

$$= -\sum_{(j\in\Omega_{u_i})} \frac{1}{2\sigma^2}\left(-2M_{ij}v_j + 2v_j v_j^T u_i\right) - \lambda u_i = 0 \qquad (ii)$$

$$= \sum_{j\in\Omega_{u_i}}\frac{1}{\sigma^2}\left(M_{ij}v_j - v_j v_j^T u_i\right) - \lambda u_i = \sum_{j\in\Omega_{u_i}}\frac{1}{\sigma^2}\left(M_{ij} - u_i^T v_j\right)v_j - \lambda u_i = 0 \quad (i)$$

(oh) as required

Rearranging for u_i :-

$$\lambda u_i + \sum_{j\in\Omega_{u_i}}\frac{1}{\sigma^2}v_j v_j^T u_i = \sum_{j\in\Omega_{u_i}}\frac{1}{\sigma^2}M_{ij}v_j \qquad (\times\sigma^2)$$

$$\left(\lambda\sigma^2 I + \sum_{j\in\Omega_{u_i}} v_j v_j^T\right)u_i = \sum_{j\in\Omega_{u_i}} M_{ij}v_j$$

(u_i update expression)

$$\Rightarrow u_i = \left(\lambda\sigma^2 I + \sum_{j\in\Omega_{u_i}} v_j v_j^T\right)^{-1}\left(\sum_{j\in\Omega_{u_i}} M_{ij}v_j\right)$$

(oh) as required

Without loss of generality cd via symmetry of u_i and v_j, we can replace u_i with v_j :-

$$v_j = \left(\lambda\sigma^2 I + \sum_{i\in\Omega_{v_j}} u_i u_i^T\right)^{-1}\left(\sum_{i\in\Omega_{v_j}} M_{ij}u_i\right)$$

(v_j update expression)

(oh) as required.

- Note on $\nabla_{u_i}(v_j^T u_i u_i^T v_j)$:-

$$\nabla_{u_i}(v_j^T u_i u_i^T v_j) = v_j v_j^T u_i = v_j u_i^T v_j = v_j(u_i^T v_j)$$

i.e. note that dot product is commutative

$$u_i^T v_j = v_j^T u_i$$

- Significant time was spent clearing some confusion :- (i) and (ii)
- set $u_i = \underline{u}$ $\quad v_j = \underline{v}$
- $\nabla_u (v^T u u^T v)$ is a gradient $\nabla_u f \in \mathbb{R}^d$ that is defined on a scalar fn
- $f: \mathbb{R}^d \to \mathbb{R}$ where $f(\underline{v}) = v^T u u^T v = \|u^T v\|_2^2$
- Let $\underline{u} = \begin{bmatrix} u_1 \\ \vdots \\ u_d \end{bmatrix}$ and $v = \begin{bmatrix} v_1 \\ \vdots \\ v_d \end{bmatrix}$ and for illustrative purposes, set $d = 2$

- $v^T u u^T v = \begin{bmatrix} v_1 & v_2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \begin{bmatrix} u_1 & u_2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = (u_1 v_1 + u_2 v_2)^2 = u_1^2 v_1^2 + 2u_1 v_1 u_2 v_2 + u_2^2 v_2^2$

- $\nabla_u (v^T u u^T v) = \begin{bmatrix} \frac{\partial f}{\partial u_1} \\ \frac{\partial f}{\partial u_d} \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial u_1} \\ \frac{\partial f}{\partial u_2} \end{bmatrix} = \begin{bmatrix} 2u_1 v_1^2 + 2u_2 v_1 v_2 \\ 2u_2 v_2^2 + 2u_1 v_1 v_2 \end{bmatrix} = 2 \begin{bmatrix} v_1 (u_1 v_1 + u_2 v_2) \\ v_2 (u_1 v_1 + u_2 v_2) \end{bmatrix}$ $\quad d=2$

$$= 2 \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} (u_1 v_1 + u_2 v_2) = 2 (u_1 v_1 + u_2 v_2) \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

$$= 2v(u^T v) = 2(u^T v)v \quad \text{as } u^T v \text{ is scalar}$$

$$= 2v(v^T u) = 2(v^T u)v \quad \text{as } u^T v = v^T u \text{ (dot product commutativity)}$$

- $\nabla_u (v^T u u^T v) = 2vv^T u$, or any combination that is of same dimensionality $\mathbb{R}^d$

Factoring out v_j and concerns about right multiplying or left multiplying :-

Topic modelling - motivation, overview

- with given text data we want to organise/visualise/summarise/search/predict/understand documents
- Topic models allow us to discover themes, annotate documents, organise etc.
- useful in information retrieval, recommendation, prediction
- Example of a document containing themes
- (see diagram!) - toy example of topic modelling
- A probabilistic topic model :- (visualise the document(s))

i) Topics - Estimates distributions on words called topics shared by documents;
- Represented in diagram as a list of words that ~~are~~ are associated with a probability. Lists are ordered and topics are global/corpus-wide.
- set of probability distributions on a fixed vocabulary
- Each probability distribution assigns all/almost all probability to a subset of words that 'belong together' and express a theme

ii) Topic proportions - estimates a distribution on topics for each document.
- Each document contains variable proportions of a number of topics

iii) Topic assignment - Assigns every word in a document to a topic

Latent Dirichlet Allocation (LDA) is the most famous example of probabilistic topic modelling. (i), (ii), (iii)

None of the ~~for~~ above, is known and must be 'learned'

Each document is represented as a bag of words (more later)

Have to define :-

1) Model
- Inference techniques will not be covered.

2) 'Learning/estimation algorithm'

Latent Dirichlet Allocation (LDA) - model setup, Dirichlet distribution properties, output

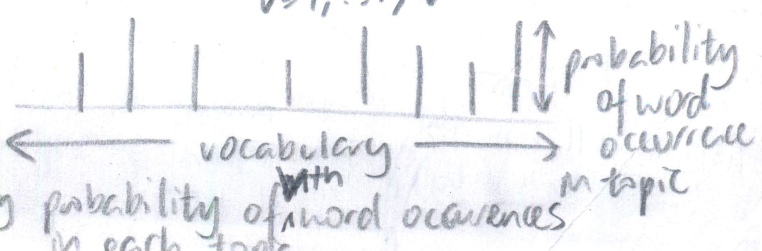
There are two key elements to LDA :-

1) A collection of (probability) distributions on words (topics)
2) A distribution on topics for each document (topic proportions)

(see diagram) :- (it's a key visualisation anchor)

Each topic is represented by a probability distribution on a global, corpus-wide fixed set of vocabulary words (words indexed by i i.e. ~~$i = 1, \ldots, N$~~ $v = 1, \ldots, V$)

For the k^{th} topic out of K topics :- $\beta_K =$



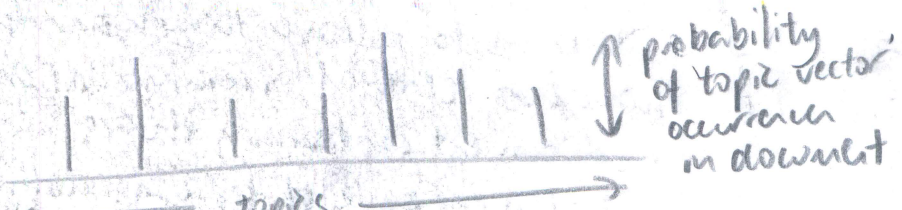
k^{th} topic vector β_K

β_K contains V elements $[\beta_1, \ldots, \beta_V]_K$ indicating probability of word occurences in each topic

2) Each document is represented by a document-specific distribution on topics or by a 'signature' topic proportion

For d^{th} document out of D documents :- (D documents indexed by $d = 1, \ldots, D$)

- d^{th} topic distribution vector $\underline{\theta}_d =$
- $\underline{\theta}_d$ contains K elements
- $\underline{\theta}_d = [\theta_1, \ldots, \theta_K]_d$ indicating probability of K topic occurrences in each document



- Formally :-
- The generative process for LDA is :-

1. Generate each topic, which is a distribution on words :-

$$\beta_K \sim \text{Dirichlet}_{(V)}(\gamma) \quad k = 1, \ldots, K$$

2. For each document, generate a distribution on topics :-

$$\theta_d \sim \text{Dirichlet}_{(K)}(\alpha) \quad d = 1, \ldots, D$$

3. For the n^{th} word in the d^{th} document :-
   a. Allocate the word to a topic $z_{dn} \sim \text{Discrete}(\theta_d)$
   b. Generate the word from the selected topic $x_{dn} \sim \text{Discrete}(\beta_{z_{dn}})$

comments (on 1) (*) hypothesise
- This is a Bayesian model → generative process (in the form of a distribution family) then derive an inference procedure which conversely estimates parameters' under that distribution family / calculates posterior

(1)
- K different topics underlying dataset ⇒ K different probability distributions on same set of words (in vocabulary)
- Each of these K topic-specific word distributions $\underline{\beta}_K$ allocates probability mass on the coherent subset of words in vocabulary; each probability mass value represented by the elements of $\underline{\beta}_K = (\beta_1, \ldots, \beta_V)$
- Prior model assumes each topic is generated ~~independently~~ iid as a Dirichlet distribution with parameter (γ), and is placed on each of K topics
- Topic-specific word distributions are drawn from this distribution then fixed.

(2)
- For each individual d^{th} document, we need to decide for that document how it will use the topics available to it / topic proportions by generating a ~~distribution~~ (probability) distribution on topics within that document

Each of the D document-specific topic distributions θ_d allocate probability mass on the K different topics, each probability mass value represented by the K elements of $\underline{\theta}_d = (\theta_1, \ldots, \theta_K)$

- Prior model assumes each topic proportion within the document is generated iid as a Dirichlet distribution with parameter α, and is placed on each of D documents.

(3) Generating words in the document (topic assignment) –

- For the d^{th} document and the n^{th} generated word, we have to decide
  a) which topic that word comes from i.e. generate a topic indicator c_{dn} from a discrete distribution using the distribution on topics for the d^{th} document θ_d. c_{dn} selects one of the K topics available to it, with probability encoded in distribution vector $\underline{\theta}_d$ in d^{th} document
  b) once topic is chosen, we have the data-level: n^{th} word is generated from a discrete distribution using the topic selected by topic indicator c_{dn}.

- $c_{dn} \in \{1, \ldots, K\}$ that selects the index of the topic to use to generate the data; selects correct index for that word in the document

(III) We do not know c_{dn} (topic indicators), $\underline{\theta}_d$ (distributions on topics / topic proportions) and ~~topics~~ β_K (distributions on words / topics), only have data x_{dn}.

(IV) Bag of words model – word order is removed and only absolute counts are modelled in this process, generated document will not be readable, appropriate for thematic modelling but not for readable language. sequential dependencies ~~are~~ in word order removed.

Properties of Dirichlet distribution

- Dirichlet distribution is a continuous distribution on discrete (probability) vectors. (eg. β_K)
- These probability vectors must satisfy i) non-negativity i.e. each ~~(today?)~~ element of β_K must be greater than or equal to 0 i.e. $\beta_{Ki} \geq 0$ ✓

  ii) $\sum_{i=1}^{V} \beta_{Ki} = 1$

- Suitable when we need to put a probability distribution on discrete distri. e.g. a prior on a K or V dimensional discrete probability distribution.

· Formally

· Let $\underline{\beta_K}$ be a probability vector and $\underline{\gamma}$ a positive parameter vector:-

$$p(\underline{\beta_K}|\underline{\gamma}) = \frac{\Gamma\left(\sum_{v=1}^{V}\gamma_v\right)}{\prod_{v=1}^{V}\Gamma(\gamma_v)} \prod_{v=1}^{V} \beta_{K,v}^{\gamma_v - 1}$$

defines a Dirichlet distribution with $V \geq 2$ (no. of categories)

γ_v - concentration parameters

② How does distribution vary with parameter γ

② what assumptions are different values of γ enforcing?

· Indexes on β_K and γ_v are shared ⇒ V dimensional probability distribution in $\underline{\beta}$ and $\underline{\gamma}$; ⇒ dimensionality of Dirichlet distribution and topic-specific word distr. β_K is V

· Assume all values in parameter vector γ are constant and the same.

· shown in the examples:-

· Examples of β_K generated from this distribution for a constant value of γ and $V=10$.

· $\underline{\beta_K}$ is a V-dimensional probability vector $\beta_K = \begin{bmatrix} \beta_{K,1} \\ \vdots \\ \beta_{K,V} \end{bmatrix}$ $\gamma = \begin{bmatrix} \gamma_1 \\ \vdots \\ \gamma_V \end{bmatrix}$

$$\cdot\ p(\beta_K|\gamma) = \frac{\Gamma\left(\sum_{v}^{V}\gamma_v\right)}{\prod_{v=1}^{V}\Gamma(\gamma_v)} \prod_{v=1}^{V} \beta_{K,v}^{\gamma_v - 1} = \frac{\Gamma(\gamma_1 + \ldots + \gamma_V)}{\Gamma(\gamma_1)\cdot\ldots\cdot\Gamma(\gamma_V)} \cdot \beta_{K,1}^{\gamma_1 - 1} \cdot \ldots \cdot \beta_{K,V}^{\gamma_V - 1}$$

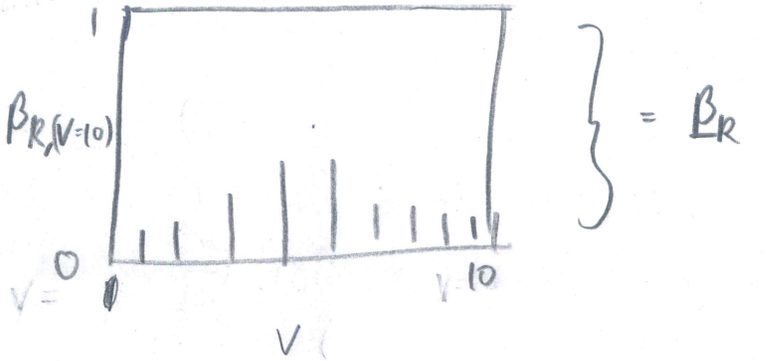
· Set $V=10$ and $\gamma_1 = \gamma_2 = \ldots = \gamma_V = 1, 10, 100, 0.01$

· (confusingly!); 10 different random vectors are drawn from above distribution i.e. $K=10$ (topics) so we have $\underline{\beta_1}, \underline{\beta_2}, \underline{\beta_3}, \ldots, \underline{\beta_{10}}$ with each

$$\beta_1 = \begin{bmatrix} \beta_{1,1} \\ \vdots \\ \beta_{1,V} \end{bmatrix} \quad \beta_2 = \begin{bmatrix} \beta_{2,1} \\ \vdots \\ \beta_{2,V} \end{bmatrix} \quad \ldots \quad \beta_{K=10} = \begin{bmatrix} \beta_{10,1} \\ \vdots \\ \beta_{10,V} \end{bmatrix}$$

· Each diagram:-

$\gamma = 1$



$= \underline{\beta_K}$

Note

$\beta_{K,v} \geq 0 \quad \forall v, K$

$\sum_{v}^{V} \beta_{K,v} = 1 \quad \forall K$

- Each diagram in the lecture represents the draw of a random vector β_K (i.e. topic specific word-distribution) / discrete probability vector from the continuous Dirichlet distribution.
- As $K=10$, 10 discrete probability distributions are generated.

(OTT) As $\gamma \to \infty$, each element $\beta_{K,v}$ of the topic-specific word distribution vector β_K becomes uniform for all K topics

- As $\gamma \to 0$, probability mass is concentrated on a subset of the dimensions V (i.e. words). A little like sparsity among elements of β_K for all $K=10$ topics.

(OTT) In practice $\gamma < 1$ to reflect the a priori belief that each topic should put probability mass on a small subset of the total words available to it (V) in the vocabulary.

- E.g. sports topic puts most of probability mass on sports-related words (small subset of vocabulary)

(OTT) prior distribution on the topic proportions also Dirichlet, parametrised by $\alpha_{(K)}$ less than $1 \Rightarrow$ a priori belief that each document uses a subset of ~~do~~ topics available to it.
- E.g. 30 different topics i.e. topic-specific word distributions / vectors β_K, then perhaps set prior $\alpha_{(K)}$ so that on average only see around 5 topics per document.
- Inference algorithm $\longrightarrow$ variational inference methods. (not covered)

Output

- See diagram (10 topics from a set of topics on millions of docs over 2 year period in NYT)
- Vocabulary consists of 8000 words; so $\beta_K \in \mathbb{R}^{8000}$ only $\beta_{K,1} \dots \beta_{K,10}$ shown for each of the 10 β_Ks where β_K is sorted so that most probable are shown first.
- Each list represents one β_K with 10 elements shown; ~~each β~~

(OTT) each β_K selects a coherent theme, and crucially, this theme is not encoded within the model.

(OTT) To summarise, LDA outputs:-

1) A set of (K) topic-specific word distributions β_K represented as discrete probability vectors. (topics)
2) (Not shown) A set of (D) document-specific topic distributions θ_d represented as probability vectors (topic proportions)

Relation between LDA and NF via magnalisation

(Q): For a particular document, what is $p(x_{dn}=i \mid \underline{\beta}, \underline{\theta}_d)$?

- For "what is probability n^{th} word in d^{th} document is equal to index i in the vocabulary list, given only set of topics and given topic proportion for d^{th} document?"
- Topic indicator c_{dn} of which topic a word comes from not present.
- statistically asking for a marginal distribution on x_{dn} with c integrated out:-

$$p(x_{dn}=i \mid \underline{\beta}, \underline{\theta}_d) = \sum_{k=1}^{K} p(x_{dn}=i, c_{dn}=k \mid \underline{\beta}, \underline{\theta}_d)$$

$$= \sum_{k=1}^{K} \underbrace{p(x_{dn}=i \mid \underline{\beta}, c_{dn}=k)}_{=\beta_{ki}} \underbrace{p(c_{dn}=k \mid \underline{\theta}_d)}_{=\theta_{dk}}$$

- Recall $\underline{\beta}_k = \begin{bmatrix} \beta_{k,1} \\ \vdots \\ \beta_{k,V} \end{bmatrix}$ $\underline{\theta}_d = \begin{bmatrix} \theta_{d,1} \\ \vdots \\ \theta_{d,K} \end{bmatrix}$

- Let $\underline{\beta} = [\underline{\beta}_1, \dots, \underline{\beta}_K]$ and $\underline{\theta} = [\underline{\theta}_1, \dots, \underline{\theta}_D]$

- $\underline{\beta} \in \mathbb{R}^{(V \times K)}$ and $\underline{\theta} \in \mathbb{R}^{(K \times D)}$

$$\underline{\beta}\underline{\theta} = \begin{bmatrix} \beta_{1,1} & \cdots & \beta_{K,1} \\ \vdots & \ddots & \\ \beta_{1,V} & \cdots & \beta_{K,V} \end{bmatrix} \begin{bmatrix} \theta_{1,1} & \cdots & \theta_{D,1} \\ \vdots & \ddots & \vdots \\ \theta_{1,K} & \cdots & \theta_{D,K} \end{bmatrix}$$

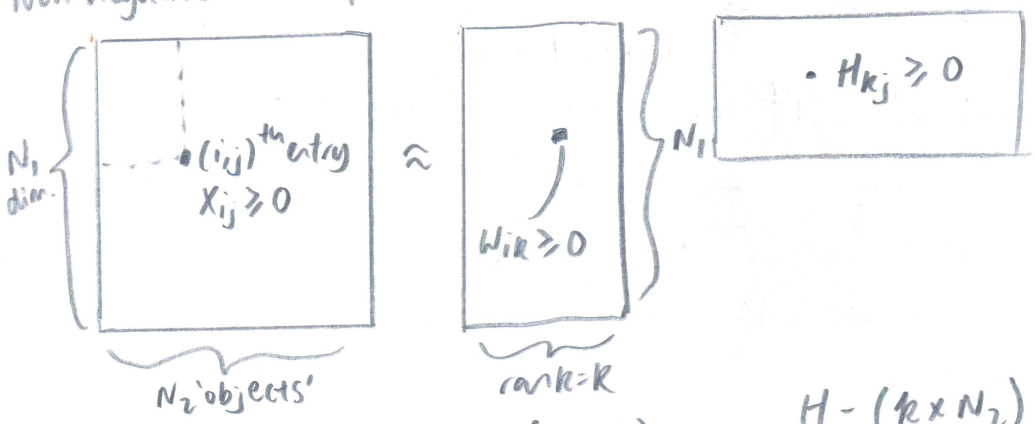
(!!) :- $p(x_{dn}=i \mid \underline{\beta}, \underline{\theta}) = (\underline{\beta}\underline{\theta})_{id}$

- Pobabilities are entries of matrix formed by product of 2 matrices with non-negative entries.

Non-negative matrix factorisation technique and 2 NMF objectives

- LDA is can be interpreted as an instance of nonnegative metrix factorisation; is probabilistic and inference techniques not taught.
- Non-negative matrix factorisation technique vs NMF algorithm
- NMF Bears similarities to PMI, but notation and interpretation different to avoid confusion.

Non-negative matrix factorisation



$X - (N_1 \times N_2)$ $W - (N_1 \times k)$ $H - (k \times N_2)$

- Data X - non-negative entries; none missing, many $X_{ij} = 0$ $X_{ij} \geq 0$
- Estimated factorisation → W and H non-negative entries $W_{ik}, H_{kj} \geq 0$
- $X_{ij} \approx \sum_{k=1}^{K} W_{ik} H_{kj}$ (not written in vector notation) (OTT)
- Interpret in terms of columns of W and H

(Q) - Data modelling problems constituting X?

- Text data: X_{ij} - word-term frequencies X_{ij} - no. of times word i appears in document j
  X - columns represent a histogram on word frequencies in a document j
  rows represent a histogram of word i occurrences across documents
- Image data: Face identification data sets
  Vectorise $N \times M$ face image into a column of X
- Discrete grouped data: Quantise continuous sets of K means (i.e. compress info into a prototype based description defined by cluster medoids)
  X_{ij} - count on times group j uses cluster i or cluster i appears in group j
  E.g i indexing features and j indexing songs
  (Q) ← $(d \times n)$ spectral information matrix (quantise)
  d-frequency content, n-time slice

2 NMF objective functions

- NMF minimises one of two objective functions over W and H:-

NMF #1: squared error objective $\|X - WH\|_2^2 = \sum_i \sum_j (X_{ij} - (WH)_{ij})^2$

NMF #2: Divergence objective $D(X\|WH) = -\sum_i \sum_j \left[X_{ij} \ln (WH)_{ij} - (WH)_{ij} \right]$

- subject to W_{ik} and $H_{kj} \geq 0 \ \forall k,i,j$ (non-negativity constraint)
- NMF → fast simple algorithm via multiplicative updates

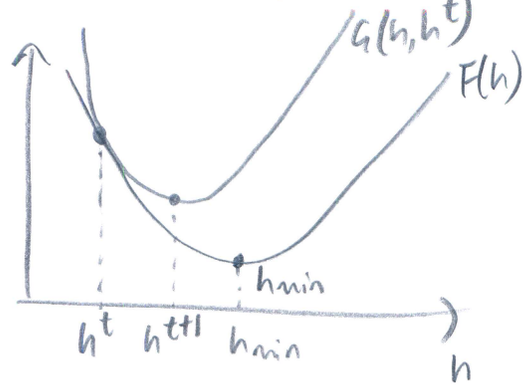
NMF local minima proof of convergence, multiplicative updates

- Based entirely on "Algorithms for non-negative matrix factorisation" - Lee & Seung (2001)

- We want to find h such that $\min_h F(h)$
- We require :-

1) A sequence of parameter values $h^1, \ldots, h^t$ such that as $t \to \infty$ $F(h^t) \to k$ (monotonically decreasing)
   i.e. $F(h^1) \geq F(h^2) \geq F(h^3) \geq \ldots \geq \ldots$

2.) Sequence converges to a local minimum of F



- Following algorithms fulfil these requirements.
  minimise via an auxiliary function G
  → multiplicative algorithm for W and H
- Auxiliary function at iteration t is equal to $F(h^t)$, is convex and bounded below by $F(h)$
- similarities to EM algorithm :- $G(h,h^t)$ - $\mathcal{L}$; $G(h,h^t)$ - $F(h)$ - KL ; $F(h)$ - $p(x|\theta)$
- NMF update algorithm, visualisation, maximum likelihood interp. of penalties for :-
- NMF #1 - squared error multiplicative update $\|X-WH\|_2^2$

optimisation problem :-

$$\min_{W,H} \sum_{ij} \left(X_{ij} - (WH)_{ij}\right)^2 \quad \text{st} \quad W_{ik} \geq 0 \quad H_{kj} \geq 0$$ (all values in W,H non-negative)

Algorithm:

- Randomly initialise H and W with nonnegative values
- Iterate the following, first for all values in H, then all in W :-

$$H_{kj} \leftarrow H_{kj} \left(\frac{(W^T X)_{kj}}{(W^T W H)_{kj}} \right)$$

$$W_{ik} \leftarrow W_{ik} \left(\frac{(X H^T)_{ik}}{(W H H^T)_{ik}} \right)$$

until change in $\|X-WH\|_2^2$ is 'small'

comment:

- To update $(k,j)^{th}$ and (i,k^{th}) element of H and W ; take old value of element and multiply by a "gain" (a function of old values of W and H and the data X) of W and H
- Proof : iterative updates lead to monotonic decrease of objective $\|X-WH\|_2^2$
- In practice, entire matrices can be updated simultaneously

· using :- via independence

$$P(X|W,H) = \prod_{i,j} P(X_{ij}|W,H) = \prod_{i,j} Po(X_{ij}|(WH)_{ij}) \leftarrow \text{maximise log-likelihood of this term}$$

$$\ln\left(\prod_{i,j} Po(X_{ij}|(WH)_{ij})\right) = \sum_{i,j} \ln\left(Po(X_{ij}|(WH)_{ij})\right)$$

$$= \sum_{i,j} \ln\left(\frac{(WH)_{ij}^{X_{ij}}}{(X_{ij})!} e^{-(WH)_{ij}}\right)$$

$$= \sum_{i,j} X_{ij} \ln\left[(WH)_{ij}\right] - (WH)_{ij} - \ln\left(X_{ij}!\right)$$

$$= \sum_{i,j}\left[X_{ij} \ln (WH)_{ij} - (WH)_{ij}\right] + C$$

$$\Rightarrow \underset{W,H}{\text{argmax}} \ln\left(\prod_{i,j} Po\left(X_{ij}|(WH)_{ij}\right)\right) = \underset{W,H}{\text{argmax}} \left(-D(X\|WH)\right) = \underset{W,H}{\text{argmin}} \left(D(X\|WH)\right)$$

· Recall for $Po(X|\lambda)$; $\lambda = E(X)$

· matrix factorisation under divergence $D(W\|XH)$; $E(X_{ij}) = \lambda = (WH)_{ij}$

(i) NMF is estimating expected value for data matrix $E(X)$ under generative assumption that likelihood is from the Poisson distribution family, parametrised by $(i,j)^{th}$ element of the product of non-negative matrix factors $(WH)_{ij}$.

· Intuitively, Poisson distribution family $Po(\cdot)$ makes sense for count data; no limit on integer count interval, but low probability of high counts is captured.

(ii) Divergence penalty has intuitive explanation.

Relation between LDA and NMF via divergence penalty :-

· NMF has an application in topic modelling ; divergence penalty closely related to LDA :-

1) Form term frequency matrix X - (X_{ij} = # times word i appears in doc j)

2) Run NMF to learn/estimate W and H using $D(X\|WH)$ penalty

3) After 2) complete ; for $k = 1, \ldots, K$

i) Set $a_k = \sum_i W_{ik}$ (a_k is sum of elements in k^{th} column of W)

ii) Divide W_{ik} (k^{th} column of W) by a_k for all i

iii) Multiply H_{kj} (k^{th} row of H) by a_k

} scaled data

visualisation + interp.

- see diagram
- * and / refer to element-wise multiplication/division ⊛
- Probabilistically; squared error penalty ⇒ Gaussian dist'n.

$$X_{ij} \sim N(\Sigma_k W_{ik} H_{kj}, \sigma^2) \quad \text{st. non-negativity constraints}$$

- Analogous to maximum likelihood on a model where elements of X, X_{ij} are Gaussian with mean $\Sigma_k W_{ik} H_{kj}$ and variance σ^2.
- As $X_{ij} \geq 0$ (and often not continuous); this is an incorrect modelling assumption that is excused by way of pragmatic functionality.

NMF#2 - KL divergence objective multiplicative update $D(X||WH)$

optimisation problem:-

$$\min_{W,H} \Sigma_{ij} \left[X_{ij} \ln \frac{1}{(WH)_{ij}} + (WH)_{ij} \right] \quad \text{st} \quad W_{ik} \geq 0 \quad H_{kj} \geq 0$$

Algorithm:-

- Randomly initialise W and H with non-negative values
- Iterate the following, first for all values in H, then all values in W:-

$$H_{kj} \leftarrow H_{kj} \left(\frac{\Sigma_i W_{ik} X_{ij} / (WH)_{ij}}{\Sigma_i W_{ik}} \right)$$

- similar intuition regarding gain
- notice symmetry of $\Sigma_i W_{ik}$ and $\Sigma_j H_{kj}$

$$W_{ik} \leftarrow W_{ik} \left(\frac{\Sigma_j H_{kj} X_{ij} / (WH)_{ij}}{\Sigma_j H_{kj}} \right)$$

until change in $D(X||WH)$ is "small"

visualisation + ML interp.

- update $H \Rightarrow$ update $\left(\frac{X}{WH}\right)$
- understand normalisation over rows/columns
- (Q) what is the equivalent probabilistic model (explicit) that leads to divergence penalty?

(A) negative divergence penalty is maximum likelihood for W and H

if we model the data as independent Poisson variables:-

$$X_{ij} \overset{iid}{\sim} \text{Pois}((WH)_{ij}) \qquad \text{Poisson}(x|\lambda) = \frac{\lambda^x}{x!} e^{-\lambda} \qquad x \in \{0, 1, 2, \ldots, \}$$

- discrete distribution on non-negative count-data integers

- Does not change matrix multiplication WH
- From perspective of divergence minimisation and maximum likelihood estimation, both penalty terms do not penalise specific values of W and H, but the resulting values in the matrix product $\underline{W}\underline{H}$!

(con) Interpretation:- The k^{th} column of W can be interpreted as the k^{th} topic (i.e. a distribution on words) and the j^{th} column of H can be interpreted as how much document j uses each topic (document-specific distribution on topics).

↳ Algorithm estimates $\underline{H}^*$ and $\underline{W}^*$ i.e. H^*_{kj} and W^*_{ik} $\forall (i,j,k)$ such that $D(X||WH)$ is minimised.

set $K = K$ topics i.e. k as index letter on K topics

- $\underline{W}^* \in \mathbb{R}^{N_1 \times K} = \begin{bmatrix} w^*_{1,1} & \cdots & w^*_{1,K} \\ \vdots & \ddots & \vdots \\ w^*_{N_1,1} & \cdots & w^*_{N_1,K} \end{bmatrix}$ $\quad \underline{H}^* \in \mathbb{R}^{K \times N_2} = \begin{bmatrix} h^*_{1,1} & \cdots & h^*_{1,N_2} \\ \vdots & \ddots & \\ h^*_{K,1} & \cdots & h^*_{K,N_2} \end{bmatrix}$

- For k^{th} column in W^*, divide by $a_k = \sum_{i=1}^{N_1} w^*_{i,k}$:-

i.e. $\underline{w}_k = \begin{bmatrix} w^*_{1,k}/a_k \\ \vdots \\ w^*_{N_1,k}/a_k \end{bmatrix}$ column $\quad (\underline{w}_k)_i = \left(w^*_{i,k} \Big/ \sum_{i=1}^{N_1} w_{i,k}\right) = w^*_{i,k} \alpha_k$

so each i^{th} element of $\underline{w}_k \in \mathbb{R}^{N_1}$ represents the probability of word i occurring within the k^{th} topic for a topic k

- Each vector $\underline{w}_k$ then represents a probability distribution on words, and is a topic
- For k^{th} row of H^*, multiply by $a_k = \sum_{i=1}^{N_1} w^*_{i,k}$
- The j^{th} column of H^* after multiplication :-

$$\underline{h}_j = \begin{bmatrix} h^*_{1,j} \cdot a_1 \\ h^*_{2,j} \cdot a_2 \\ \vdots \\ h^*_{K,j} \cdot a_K \end{bmatrix}$$

so each k^{th} element of $\underline{h}_j \in \mathbb{R}^K$ represents the amount document j uses the k^{th} topic

$$(\underline{h}_j)_k = \left(h^*_{k,j} \Big/ \sum_{i=1}^{N_1} w^*_{i,k}\right) \cdot h^*_{k,j} a_k$$

- Each column vector $\underline{h}_j$ represents a distribution on topics for a document j.
- Application of K-means, SVD and NMF to face modelling data
- K-means, SVD and NMF applied to facial modelling data.
- see diagram face

- $X = \begin{matrix} \text{pixel} \\ \text{val.} \end{matrix} \begin{bmatrix} \downarrow \\ 101 \\ 100 \\ 0 \\ \vdots \\ 255 \end{bmatrix}$ Each face is represented as a column of pixel values (non-neg.)

i.e.
- Image database of (pixel value x face)

- $X \approx WH$ $\quad X_{ij} \approx \sum_{R=1}^{K} W_{ik} H_{kj}$
  - K columns of W are basis images
  - Columns of H is an encoding; with one-to-one correspondence with a face in X
  - An encoding consists of the coefficients by which a face is represented by a linear combination of basis images.

K-means/vector quantisation (VQ) — prototypical (centroid defined)

- Every column of X (i.e. face) is represented by a single basis image (column of W) with each column of H many i.e. one element = 1 and rest equal to zero; with that unity element falling on the clustering assignment.
- Note that many native of columns of H ⇒ hard clustering

Singular value decomposition (SVD)

- Every column of X (i.e. face) is represented by an eigenface single basis image (column of W) and a non-spase matrix of encodings H.
- PCA ⇒ columns of W are orthonormal and rows of H orthogonal
- A face is a linear combination of all basis images/eigenfaces
- visual interpretation difficult due to relaxed non-negativity of values in W and H together with orthogonality ⇒ complex additive/subtractive cancellations little intuitive meaning to an eigenface.

NMF — interpretable

Every column of X (i.e. face) is represented by parts-based localised features as single basis image (column of W) and sparse matrix H

NMF ⇒ non-negativity of W and H

A face is a linear combination of parts/localised features

Non-negativity constraints ⇒ parts based learning

Abstract NMF framework is powerful

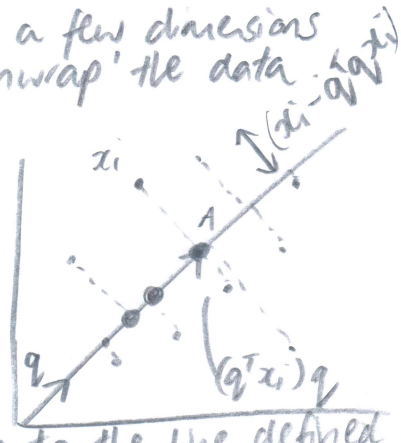
- topic modelling:- Each column of X (doc) represented as linear combination of topics

- Dimensionality reduction - overview, motivation
- Given data $x_1, \dots, x_n$ with $x_i \in \mathbb{R}^d$, data is high-dimensional, but 'info' does not use full d dimensions.
- Represent no. 3 with three numbers corresponding to 3 degrees of freedom, two shifts, one rotation; or, write as a 2-dimensional vector (covering 2 shifts) and one rotation.
- Principal components analysis (PCA) is a way of automatically mapping high-dim data x_i into a new, low-dimensional coordinate system
  - captures most of the 'information' in the data into a few dimensions
  - extensions allow us to handle missing data, and 'unwrap' the data

Principal components analysis (PCA)

Toy example - reducing $x_i \in \mathbb{R}^2$ to $\mathbb{R}^1$

- $x_i \in \mathbb{R}^2$, objective is to map $x_i \in \mathbb{R}^2 \to (q^T x_i) \in \mathbb{R}^1$ (approj)
- How to approximate data $\{x_i\}$ using a unit-length vector q?
- q, a unit length vector $\Rightarrow q^T q = 1$ (orthogonally)



- Large dots + length $q^T x_i$ to the axis after projecting x onto the line defined by the unit vector q.
- The vector $(q^T x_i)q$ takes q and stretches it to the large dot (think scalar multiple of a unit vector, with scalar $(q^T x_i)$.
- How to select q? Minimise square approximation error

$$q = \underset{q}{\operatorname{argmin}} \sum_{i=1}^{n} \| x_i - qq^T x_i \|_{(2)}^2 \quad \text{subject to } q^T q = 1$$

$$= \underset{q}{\operatorname{argmin}} \sum_{i=1}^{n} (x_i - qq^T x_i)^T (x_i - qq^T x_i)$$

[...] difficulties here

$$\underset{q}{\operatorname{argmin}} \sum_{i=1}^{n} \left[x_i^T x - x_i^T qq^T x_i - x_i^T qq^T x_i - x_i^T q(q^T q) q^T x_i \right]$$

$(ABC)^T = C^T B^T A^T$

Recall $(x_i^T q) = (q^T x_i)$ and $q^T q = 1$

$$= \underset{q}{\operatorname{argmin}} \sum_{i=1}^{n} \left[x_i^T x_i - 2 x_i^T qq^T x_i + x_i^T qq^T x_i \right]$$

$$= \underset{q}{\operatorname{argmin}} \sum_{i=1}^{n} \left[x_i^T x_i - x_i^T qq^T x_i \right]$$

$$= \underset{q}{\operatorname{argmin}} \sum_{i=1}^{n} \left[x_i^T x_i - q^T x_i x_i^T q \right]$$

$$= \underset{q}{\operatorname{argmin}} \sum_{i=1}^{n} x_i^T x_i - q^T \left(\sum_{i=1}^{n} x_i x_i^T \right) q$$

$$= \underset{q}{\operatorname{argmax}}\ q^T (XX^T) q \quad \text{s.t. } q^T q = 1$$

in this case $d = 2$ - and this time switched features and data index

- As q is not indexed by i (*)
- As $\sum_{i=1}^{n} x_i x^T = XX^T$ - see end for clarification
- where $X \in \mathbb{R}^{d \times n}$ with $X = \begin{pmatrix} | & & | \\ x_1 & \dots & x_n \\ | & & | \end{pmatrix}$
- each x_i stacked as column of X
- $XX^T \in \mathbb{R}^{d \times d}$ (symmetric)
- As q is not inside $x_i^T x_i \Rightarrow$ drop

- This is an eigendecomposition problem:- as (XX^T) is symmetric

with q as the first eigenvector of XX^T

and λ as the fist eigenvalue and $\lambda = q^T(XX^T)q\big|_{q_{max}}$ i.e. λ is the value of $q^T(XX^T)q$ evaluated at q_{max}

for the corresponding problem: $(XX^T)q = \lambda q$

General form

- General form of PCA considers K eigenvectors where K is arbitary (q_K)
- we want to approximate $x_i \in \mathbb{R}^D$ with a linear combination of K different vectors, each of unit length, and orthogonal to one another

- $q = \underset{q}{\operatorname{argmin}} \sum_{i=1}^{n} \|x_i - \sum_{k=1}^{K} (x_i^T q_k) q_k\|^2$ s.t. $q_k^T q_{k'} = \begin{cases} 1 \text{ if } k=k' \\ 0 \text{ if } k \neq k' \end{cases}$

i) Remember $x^T y$ is scalar and is also commutative

ii) Bear in mind dimensionality

iii) $A^T = A$ if scalar

($\sum_{k=1}^{K} (x_i^T q_k) q_k$ approximates x)

$$= \underset{q}{\operatorname{argmin}} \sum_{i=1}^{n} \left[x_i^T x_i - 2x_i^T \left(\sum_{k=1}^{K} (x_i^T q_k) q_k \right) + \left(\sum_{k=1}^{K} q_k^T (x_i^T q_k)^T \right) \left(\sum_{k=1}^{K} (x_i^T q_k) q_k \right) \right]$$

$$= \underset{q}{\operatorname{argmin}} \sum_{i=1}^{n} \left[x_i^T x_i - 2x_i^T \left(\sum_{k=1}^{K} (x_i^T q_k) q_k \right) + \sum_{k=1}^{K} \sum_{k'=1}^{K} q_k^T (x_i^T q_k)^T (x_i^T q_{k'}) q_{k'} \right]$$

$$= \underset{q}{\operatorname{argmin}} \sum_{i=1}^{n} \left[x_i^T x_i - 2x_i^T \left(\sum_{k=1}^{K} (x_i^T q_k) q_k \right) + \sum_{k=1}^{K} \sum_{k'=1}^{K} (x_i^T q_k)^T (x_i^T q_{k'}) q_k^T q_{k'} \right]$$

- As $q_k^T q_{k'} = 1$ when $k=k'$ and $q_k^T q_{k'} = 0$ when $k \neq k'$, this reduces to:-

$$= \underset{q}{\operatorname{argmin}} \sum_{i=1}^{n} \left[x_i^T x_i - 2x_i^T \left(\sum_{k=1}^{K} (x_i^T q_k) q_k \right) + \sum_{k=1}^{K} (x_i^T q_k)^T (x_i^T q_k) \right]$$

$$= \underset{q}{\operatorname{argmin}} \sum_{i=1}^{n} \left[x_i^T x_i - 2x_i^T \left(\sum_{k=1}^{K} (x_i^T q_k) q_k \right) + x_i^T \left(\sum_{k=1}^{K} (x_i^T q_k) q_k \right) \right]$$

$$= \underset{q}{\operatorname{argmin}} \sum_{i=1}^{n} \left[x_i^T x_i - x_i^T \left(\sum_{k=1}^{K} (x_i^T q_k) q_k \right) \right]$$

$$= \underset{q}{\operatorname{argmin}} \sum_{i=1}^{n} \left[x_i^T x_i - \sum_{k=1}^{K} (x_i^T q_k)(x_i^T q_k) \right]$$

$$= \underset{q}{\operatorname{argmin}} \sum_{i=1}^{n} \left[x_i x_i^T - \sum_{k=1}^{K} (q_k^T x_i x_i^T q_k) \right] = \underset{q}{\operatorname{argmin}} \sum_{i=1}^{n} x_i x_i^T - \sum_{k=1}^{K} q_k^T \left(\sum_{i=1}^{n} x_i x_i^T \right) q_k$$

$$= \underset{q}{\operatorname{argmin}} \sum_{i=1}^{n} x_i^T x_i - \sum_{k=1}^{K} q_k^T (XX^T) q_k = \underset{q}{\operatorname{argmax}} \sum_{k=1}^{K} q_k^T (XX^T) q_k \text{ s.t. } q_k^T q_{k'} = \begin{cases} 1 \text{ if } k=k' \\ 0 \text{ if } k \neq k' \end{cases}$$

PCA Application - digit recognition

- Data is a 16x16 image of handwritten 3s $\Rightarrow x_i \in \mathbb{R}^{256}$
- stack each x_i to form a matrix $X \in \mathbb{R}^{256 \times (\text{no. of images})}$ after pre-processing
- Form data-covariance matrix $(XX^T) \in \mathbb{R}^{256 \times 256}$
- Shown are 4 eigenvectors for $\lambda_1, \lambda_2, \lambda_3, \lambda_4$, each $q \in \mathbb{R}^{256}$
- Each eigenvector captures decreasing amounts of 'information' not captured by previous eigenvectors
- Each eigenvector still abides by basis constraints i.e. $q_k^T q_k = 1$ and $q_k^T q_{k'} = 0$
- Bottom panel:- reconstruction of an original image using $(M-1)$ eigenvectors plus mean and approximation:-

$$x \approx \text{mean} + \sum_{k=1}^{M-1} (x^T q_k) q_k \quad (*)$$

- Mean is pre-processed by subtracting mean from each x_i to get $(x_i - \bar{x})$
- Reconstruction: using $\lambda_1, \ldots,$ and $q_1, \ldots$; take dot product of means subtracted data x_i (centered data) with each of k eigenvectors, multiply by corresponding eigenvector; add sum up to get approximation
- Each $M = 1, \ldots, 250$ is an $(M-1)$ dimensional representation of original image

Probabilistic PCA

Motivation

- using PCA and SVD perspective, we derive a probabilistic model and use EM algorithm to learn parameters (by defining a generative model and a point estimate from EM)
- Allows:- 1) Handle missing data
  2) Learn additional parameters e.g. noise
  3) Provide a modular framework
  4) Specify distributions to characterise uncertainty in prediction

SVD perspective

$X = USV^T$, $U^TU = I$, $V^TV = I$ (via orthonormality)

where $X \in \mathbb{R}^{d \times n}$ $U \in \mathbb{R}^{d \times d}$ $S \in \mathbb{R}^{d \times n}$ $V^T \in \mathbb{R}^{n \times n}$, assuming that $n \gg d$ (i.e. more data points than dimensions)

AND also recall switch of feature-sample notation

with $S_{ii} \geq 0$ i.e. diagonal S with non-negative entries

$$\Rightarrow XX^T = USV^TVS^TU^T = US^2U^T$$

$$\Rightarrow (XX^T)U = US^2 \quad (\text{SVD})$$

with U - matrix of eigenvectors and S^2 - diagonal eigenvalue matrix

- The vectors $q_1, \dots, q_K$ i.e. each eigenvector can be stacked into a matrix Q such that:

$$Q = \begin{bmatrix} | & | & & | \\ q_1 & q_2 & \dots & q_K \\ | & | & & | \end{bmatrix}$$ with each $q_K \in \mathbb{R}^d$ and hence $Q \in \mathbb{R}^{d \times K}$

- vectors in q give a K-dimensional subspace with which to represent the data

$$x_{proj} = \begin{bmatrix} q_1^T x \\ \vdots \\ q_K^T x \end{bmatrix} \qquad x \approx \sum_{k=1}^{K} (q_k^T x) q_k = Q x_{proj} \qquad x_{proj} \in \mathbb{R}^K \text{ and } Q x_{proj} \in \mathbb{R}^d \quad (\star)$$

- call in-built functions to find the K largest eigenvectors and eigenvalues

($\star$) see eigenvalues and optimisation (CS229 lin alg.)

Eigenvalues, vectors, SVD

- An equivalent eigendecomposition problem, we have to find (λ, q) such that

$$(XX^T)q = \lambda q$$

- As (XX^T) is PSD there are are $r \leq \min\{d, n\}$ pairs with

$$\downarrow \quad \lambda_1 \geq \lambda_2 \geq \lambda_3 \dots \geq \lambda_r > 0 \qquad q_k^T q_k = 1 \text{ and } q_k^T q_{k'} = 0$$

- for a matrix $A \in \mathbb{R}^{m \times n}$ $\text{rank}(A) \leq \min(m, n)$ (linear independent col.)
- $\text{rank}(A)$ = no. of non-zero eigenvalues

- Justification for (XX^T) i.e. data covariance matrix being PSD: Singular value decom.

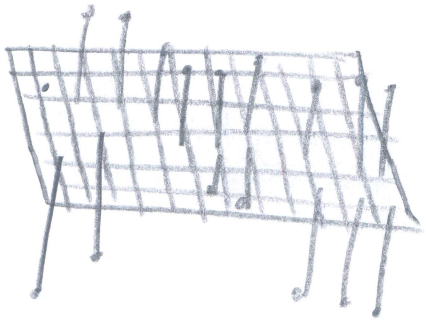
X has SVD, $X = USV^T$, we have that :-

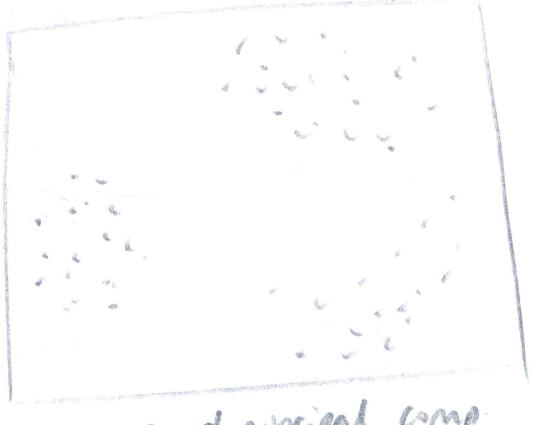
$X \in \mathbb{R}^{d \times n}$, $U \in \mathbb{R}^{d \times d}$ (orthogonal)
$V^T \in \mathbb{R}^{n \times n}$ $S \in \mathbb{R}^{d \times n}$

$$(XX^T) = USV^T VS^T U^T$$
$$= US^2U^T \quad \text{i.e. think}$$

$\Rightarrow Q = U$ and $\lambda_i = (S^2)_{ii} \geq 0$

- Pre-processing is usually done as PCA is sensitive to scaling, via subtracting off the mean of each dimension of x centres the data so they have origin at 0 and ensures principal components pass through origin.
- Projecting from $\mathbb{R}^3$ to $\mathbb{R}^2$:-





- most information (structure in data) can be captured in $\mathbb{R}^2$
- Left is data in $\mathbb{R}^3$ with a hyperplane defined by q_1 and q_2
- Right is new co-ordinates for data $x_i \rightarrow x_{proj} = \begin{bmatrix} x_i^T q_1 \\ x_i^T q_2 \end{bmatrix}$
- Intuition: Shift and rotate data to throw away dimensions, align info along dimensions we are going to keep.

setup

- with SVD, $X = USV^T$
- Approximate $X \approx WZ$ then:- $X \in \mathbb{R}^{d \times n}$

$W \in \mathbb{R}^{d \times K}$ - known as a factor loadings matrix/dictionary; similar to eigenvectors with no constraint of orthonormality and no unitlength

$Z \in \mathbb{R}^{K \times n}$ - i^{th} column of Z is called $z_i \in \mathbb{R}^K$, a low-dimensional representation of x_i

(*)

- Generative process:-

$$x_i \sim N(Wz_i, \sigma^2 I) \quad \text{and} \quad z_i \sim N(0, I)$$

where we don't know W or any of z_i

(NB): Crucial difference of probabilistic and non-probabilistic PCA are orthogonality and unit length constraints

Maximum-likelihood/expectation-maximisation estimation

- Our initial goal is to learn/estimate W and z_i
- Find maximum likelihood solution of W with wide marginal distribution i.e. with z_i integrated out then do a point estimate that maximises marginal likelihood with respect to W.

$$W_{ML} = \underset{W}{\operatorname{argmax}} \ln p(x_1, \dots, x_n | W) = \underset{W}{\operatorname{argmax}} \sum_{i=1}^{n} \ln p(x_i | W)$$

via iid of data x_i

(?) (*) (?)

- Intractable as $p(x_i | W) = N(x_i | 0, \sigma^2 I + WW^T)$

$$N(x_i | 0, \sigma^2 I + WW^T) = \frac{1}{(2\pi)^{d/2} |\sigma^2 I + WW^T|^{1/2}} e^{-\frac{1}{2} x^T (\sigma^2 I + WW^T)^{-1} x}$$

(?)

(?) - Not consistent with Bishop, Tipping & Bishop

and cannot find gradient wrt W

- Reintroduce $z_1, \dots, z_n$ as latent/auxiliary variables

EM-setup

- marginal log-likelihood expressed using EM as:-

$$\sum_{i=1}^{n} \ln p(x_i | W) = \sum_{i=1}^{n} \ln \int p(x_i, z_i | W) \, dz_i =$$

- see LIS on how this is derived

$$\sum_{i=1}^{n} \int q(z_i) \ln \frac{p(x_i, z_i | W)}{q(z_i)} \, dz_i + \sum_{i=1}^{n} \int q(z_i) \ln \frac{q(z_i)}{p(z_i | x_i, W)} \, dz_i$$

$\mathcal{L}$ KL

- E-M:
  1) set $q(z_i) = p(z_i | x_i, W)$ for each i to make $KL = 0$, calculate $\mathcal{L}$
  2) Maximise $\mathcal{L}$ with respect to W

which:
1) Requires calculation of $p(z_i|x_i, W)$
2) Requires tractable maximisation of $\mathcal{L}$ (in this case update W using simple eq.)

EM Algorithm for PPCA

Given: Data $x_{1:n}$, $x_i \in \mathbb{R}^d$ and model $x_i \sim N(Wz_i, \sigma^2 I)$ $z_i \sim N(0, I)$, $z \in \mathbb{R}^K$
Output: Point estimate of W and a conditional posterior $p(z_i|x_i, W)$ on each z_i (that maximises marginal likelihood)

E-step: set each $q(z_i) = p(z_i|x_i, W) = N(z_i|\mu_i, \Sigma_i)$ where

$$\Sigma_i = (I + W^T W/\sigma^2)^{-1}, \quad \mu_i = \Sigma_i W^T x_i/\sigma^2$$

M-step: update W by maximising $\mathcal{L}$ from E-step

$$W = \underbrace{\left[\sum_{i=1}^{n} x_i \mu_i^T\right]}_{D \times K}\underbrace{\left[\sigma^2 I + \sum_{i=1}^{n}(\mu_i \mu_i^T + \Sigma_i)\right]^{-1}}_{K \times K}$$

Iterate E and M steps until increase in $\sum_{i=1}^{n} \ln p(x_i|W)$ is small (marginal likelihood)

Comments

- conditional posterior $p(z_i|x_i, W)$ captures our posterior belief about what a low-dimensional embedding should be; so not only a low-dimensional embedding for each x_i in the $q(z_i)$, but a distribution
- Probabilistic framework gives a way to estimate K and σ^2 also
- Even though $\sum_{i=1}^{n} \ln p(x_i|W)$ is intractable, can still evaluate

PPCA Applications - image processing, denoising, missing data

- For each 8×8 patch, vectorise so that $x_i \in \mathbb{R}^{64}$ and stack into columns of matrix X such that $X \in \mathbb{R}^{64 \times 262,144}$
- use a factor model (e.g. PPCA)
- Approximate $x_i \approx W\mu_i$ where μ_i is the posterior mean of z_i (i.e ith column of Z)
- $X \approx WZ \approx W\mu_i$
- Recall that we have a point estimate of W and a conditional posterior of z_i, and a mean for z_i which is μ_i
- Reconstruct image by replacing x_i with $W\mu_i$ (and averaging)
- Noise from denoised patches has been captured in the noise variance parameter

$x_i \sim N(Wz_i, \sigma^2 I)$ ← noise variance 'soaks up' noise
↖ assuming denoised mean

- σ^2 has been estimated

(5)(4)(a)
- go over eqn

MBSing data

- 480×220×3 image (RGB)
- shows capability of image processing techniques

remember matrices as transformations

Kernel PCA

Setup

- Algorithms which make use of dot products can be generalised with a non-linear kernel; and applied to PCA.
- seems counter-intuitive to implicitly map to a higher dimensional space and then carry out dimensionality reduction.
- recall with PCA, we find eigenvectors of matrix $XX^T = \sum_{i=1}^{n} x_i x_i^T$
- Define $\phi(x)$ as a feature mapping $\phi: \mathbb{R}^d \to \mathbb{R}^D$ such that $d \ll D$
- we want to solve the eigendecomposition:

$$\left[\sum_{i=1}^{n} \phi(x_i)\phi(x_i)^T\right] q_k = \lambda_k q_k$$

without working in higher dim. space i.e. explicitly using $\phi(\cdot)$ and q

- Reorganising:-

$$\sum_{i=1}^{n} \phi(x_i) \underbrace{\left(\phi(x_i)^T q_k\right)/\lambda_k}_{=a_{ki}} = q_k$$

- setting $a_{ki} = \phi(x_i)^T q_k / \lambda_k$
- Dimensionality: $x_i \in \mathbb{R}^d$ → $\begin{pmatrix} x_1 \\ \vdots \\ x_d \end{pmatrix}$
  $\phi(x_i) \in \mathbb{R}^D$ → $\begin{pmatrix} \phi(x_i)_1 \\ \vdots \\ \phi(x_i)_D \end{pmatrix}$
  $a_{ki} \in \mathbb{R}$ $\underline{a}_k \in \mathbb{R}^n$

$$\Rightarrow q_k = \sum_{i=1}^{n} a_{ki}\phi(x_i) \quad \text{for some vector } \underline{a}_k \in \mathbb{R}^n$$

- (on) Key tactical move is to estimate $\underline{a}_k$ instead of q_k (eigenvector)

$$\Rightarrow \left(\sum_{i=1}^{n} \phi(x_i)\phi(x_i)^T\right)\left(\sum_{j=1}^{n} a_{kj}\phi(x_j)\right)/\lambda_k = \sum_{i=1}^{n} a_{ki}\phi(x_i)$$

- $K(\cdot,\cdot): \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$

$$\Rightarrow \left(\sum_{i=1}^{n} \phi(x_i)\phi(x_i)^T\right)\left(\sum_{j=1}^{n} a_{kj}\phi(x_j)\right) = \lambda_k \sum_{i=1}^{n} a_{ki}\phi(x_i)$$

↓ ⊛ justify

$$\Rightarrow \sum_{i=1}^{n} \phi(x_i) \sum_{j=1}^{n} a_{kj} \underbrace{\phi(x_i)^T\phi(x_j)}_{K(x_i,x_j)} = \lambda_k \sum_{i=1}^{n} a_{ki}\phi(x_i)$$

$$\Rightarrow \sum_{i=1}^{n} \phi(x_i) \sum_{j=1}^{n} a_{kj} K(x_i,x_j) = \lambda_k \sum_{i=1}^{n} a_{ki}\phi(x_i)$$

- set $\Phi = \begin{bmatrix} | & & | \\ \phi(x_1) & \cdots & \phi(x_n) \\ | & & | \end{bmatrix} \in \mathbb{R}^{D \times n}$

$$\Rightarrow \Phi^T\left(\sum_{i=1}^{n} \phi(x_i) \sum_{j=1}^{n} a_{kj} K(x_i,x_j)\right) = \Phi^T\left(\lambda_k \sum_{i=1}^{n} a_{ki}\phi(x_i)\right)$$

and $\Phi^T = \begin{bmatrix} -\phi(x_1)^T- \\ -\phi(x_n)^T- \end{bmatrix} \in \mathbb{R}^{n \times d}$ ⊛ justify

$$\Rightarrow \phi(x_l)^T\left(\sum_{i=1}^{n} \phi(x_i) \sum_{j=1}^{n} a_{kj} K(x_i,x_j)\right) = \phi(x_l)^T\left(\lambda_k \sum_{i=1}^{n} a_{ki}\phi(x_i)\right)$$

i.e. multiply by $\phi(x_l)^T$ for $l = 1, \ldots, n$

i.e. $\forall l \in [1, n]$

$$\Rightarrow \sum_{i=1}^{n} \phi(x_l)^T \phi(x_i) \sum_{j=1}^{n} a_{Rj} K(x_i, x_j) = \lambda_R \sum_{i=1}^{n} a_{Ri} \phi(x_l)^T \phi(x_i)$$

$$\Rightarrow \sum_{i=1}^{n} K(x_l, x_i) \sum_{j=1}^{n} a_{Rj} K(x_i, x_j) = \lambda_R \sum_{i=1}^{n} a_{Ri} K(x_l, x_i)$$

(2) justify

$$\Rightarrow \underline{K}^2 \underline{a}_R = \lambda_R \underline{K} \underline{a}_R$$

$$\Rightarrow \boxed{\underline{K}\underline{a}_R = \lambda_R \underline{a}_R} \quad (OTI)$$

where K is the kernel matrix constructed on the data and $K \in \mathbb{R}^{n \times n}$

Comments

- K is PSD as it is a matrix of dot-products
- Therefore LHS and RHS share a solution for $(\lambda_R, \underline{a}_R)$ (?)
- Recall PSD symm. matrix $A \Rightarrow x^T A x \geq 0$ for all non-zero vectors $x \in \mathbb{R}^n$
- (OTI) perform PCA on kernel matrix K rather than data matrix XX^T.

Algorithm - Kernel PCA

Given: Data $x_1, \ldots, x_n$ $x_i \in \mathbb{R}^d$ and a kernel function $K(x_i, x_j)$

Construct: The kernel matrix on the data e.g. $K_{ij} = b \exp\left\{-\frac{\|x_i - x_j\|^2}{c}\right\}$

Solve: The eigendecomposition

$$\underline{K}\underline{a}_R = \lambda \underline{a}_R$$

for the first $r \ll n$ eigenvector/eigenvalue pairs $(\lambda_1, \underline{a}_1), \ldots, (\lambda_r, \underline{a}_r)$

Output: A new co-ordinate system for x_i by implicitly mapping $\phi(x_i)$ and then projecting $q_R^T \phi(x_i)$

$$x_i \xrightarrow{\text{proj}} \begin{bmatrix} \lambda_1 a_{1i} \\ \vdots \\ \lambda a_{ri} \end{bmatrix}$$

where a_{Ri} is the i^{th} dimension of k^{th} eigenvector $\underline{a}_R$

New Data

- How do we handle new data x_0? Previously we use eigenvectors q_R and project $x_0^T q_R$ but $\underline{a}_R$ is different here.
- Recall $q_R = \sum_{i=1}^{n} a_{Ri} \phi(x_i)$; kernel trick used to avoid defining $\phi(x_i)$

- similar to regular PCA, after mapping x_0, project onto eigenvectors

$$x_0 \xrightarrow{\text{PROJ}} \begin{bmatrix} \phi(x_0)^T q_1 \\ \vdots \\ \phi(x_0)^T q_l \end{bmatrix}$$

(9) (4) (?) (but now about reconstruction using (M-1) eigenvects)

- substitute for $q_k \Rightarrow \phi(x_0)^T q_k = \sum_{i=1}^{n} a_{ki}\, \phi(x_0)^T \phi(x_i) = \sum_{i=1}^{n} a_{ki}\, K(x_0, x_i)$

Example

- separation of three rings of data points ⇒ project to higher dimensional space; perform PCA there
- Take each data point on LH panel, calculate kernel matrix K using a Gaussian Kernel; then projected it down to $d=2$.
- Projected from $\mathbb{R}^2$ to $\mathbb{R}^2$ though an infinite-dimensional intermediate mapping
- related to spectral clustering (i) and manifold learning (ii) ✓ non-linear dimensionality reduction

(i) selecting different space to cluster data
(ii) using Euclidean distance in different spaces

L20

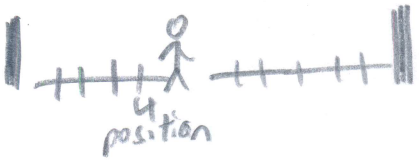
Sequential data - motivation

- Probabilistic settings involving iid i.e. data x_i are independt given a model parameter are often made for convenience
- Poor assumption when there are temporal dependencies
- E.g. modelling rainfall time series, currency exchange rates, speech audio acoustic features
- In these cases, distribution on the next value depends on previous values.
- Sequential information can be modelled using a discete first-order Markov chain.

Discete First-order Markov chains

Toy Example

- Alley drunk with 3 actions at each time step (l, s, r) with probability (p_l, p_s, p_r)
- unless next to wall $\Rightarrow$ 2 actions (s, w) with probability (p_s^w, p_w^w)
- Modelling this 'random walk' assuming that the distribution on the next location only depends on the current location.
- Assume distance from left wall $d(t)$ is discrete rather than continuous r.v.



- And that there are only a finite number of positions the drunk can be in.
- Assume a latent variable s which gives position as a function of time of random walker.
- Discretise the state space by indexing position

First-order Markov property:

For position i away from wall: $s_{t+1} | \{s_t = i\} = \begin{cases} i+1 & \text{wp } p_r \\ i & \text{wp } p_s \\ i-1 & \text{wp } p_l \end{cases}$

- second-order and higher nth order Markov chains are computationally expensive/difficult to implement.

Transition Matrix

- In giving 6 (or more generally n) positions called states, we have

$$M \in \mathbb{R}^{6\times 6} \text{ (or more generally } \mathbb{R}^{n\times n}) = \begin{bmatrix} p_s^w & p_w^w & 0 & 0 & 0 & 0 \\ p_l & p_s & p_r & 0 & 0 & 0 \\ 0 & p_l & p_s & p_r & 0 & 0 \\ 0 & 0 & p_l & p_s & p_r & 0 \\ 0 & 0 & 0 & p_l & p_s & p_w^r \\ 0 & 0 & 0 & 0 & p_w^w & p_s \end{bmatrix}$$

- Each element M_{ij} is the probability that the next position is j given current postion
- ...
- In this case, no skipping steps

Semi-formal definition

- Let $s \in \{1, \dots, S\}$ i.e. an index of a finite number of potential states we can be in
- A sequence $(s_1, \dots, s_t)$ is a first-order Markov chain if

$$p(s_1, \dots, s_t) \overset{(a)}{=} p(s_1) \prod_{u=2}^{t} p(s_u | s_1, \dots, s_{u-1}) \overset{(b)}{=} p(s_1) \prod_{u=2}^{t} p(s_u | s_{u-1})$$

(a) Property a holds via the chain rule of probability

chain rule for PDFs $f(x_1, \dots, x_n) = f(x_n | x_1, \dots, x_{n-1}) f(x_1, x_2, \dots, x_{n-1})$

$= f(x_n | x_1, \dots, x_{n-1}) f(x_{n-1} | x_1, x_2, \dots, x_{n-2}) f(x_1, \dots, x_{n-2})$

$= \dots = f(x_1) \prod_{i=2}^{n} f(x_i | x_1, \dots, x_{i-1})$

Expanding: $p(s_1, \dots, s_t) = p(s_1) p(s_2|s_1) \cdot p(s_3|s_1, s_2) \dots p(s_t | s_1, \dots, s_{t-1})$

(b) Results from Markov property assumption

Expanding $p(s_1, \dots, s_t) = p(s_1) \cdot p(s_2|s_1) \cdot p(s_3|s_2) \dots p(s_t|s_{t-1})$

- Difference from iid:

$$p(s_1, \dots, s_t) = \begin{cases} p(s_1) \prod_{u=2}^{t} p(s_u | s_{u-1}) & \text{Markov (distribution at a given time point is conditionally independent on value of sequence at previous time point)} \\ \prod_{u=1}^{t} p(s_u) & \text{iid} \end{cases}$$

Transition Matrix (general)

- encoding these conditional probabilities / general probability distribution into a matrix
- $M_{ij} = p(s_t = j | s_{t-1} = i)$ for S possible states, $M \in \mathbb{R}^{S \times S}$ (OTT)
- M is a transition matrix with (i,j^{th}) element of M_{ij} representing probability of transitioning to state j conditional on being (previous) state i.
- $M \in \mathbb{R}^{S \times S}$ and each row is a probability distribution so each row sums to unity and is non-negative i.e. for particular i, $\sum_{j=1}^{S} M_{ij} = 1$ and $M_{ij} \geq 0$
- Given a starting state so we generate a sequence $(s_1, \dots, s_t)$ by sampling (OTT)

$$s_t | s_{t-1} \sim \text{Discrete}(M_{s_{t-1}, :})$$

and by modelling the starting state with its own separate distribution.

- In above case, s_{t-1} indexes the row (prob. distribution) of matrix M, and samples along elements of that row.
- Assume that the transition matrix is time-homogeneous i.e. transition matrix M and each element does not vary with time-step and so k^{th} step transition probabilities can be computed as M^k

Maximum-likelihood estimation

- usual assumptions / methodology: Assume we know all model parameters and these generate data. In reality, we have data (sequence of states) and infer model parameters i.e. transition matrix M
- (OTT) $M_{ML} = \underset{M}{\text{argmax}}\, p(s_1, \dots, s_t | M) = \sum_{u=1}^{t-1} \sum_{i,j}^{S} \mathbb{1}(s_u = i, s_{u+1} = j) \ln M_{ij}$
- As each row of M is a probability distribution, we can show that:

$$M_{ML}(i,j) = \frac{\sum_{u=1}^{t-1} \mathbb{1}(s_u = i, s_{u+1} = j)}{\sum_{u=1}^{t-1} \mathbb{1}(s_u = i)}$$

(OTT)

Comments

- Given an observed sequence $(s_1, \dots, s_t)$, each with realised values, our goal is to find an estimate of the parameters/transition matrix probabilities.
- Under maximum likelihood estimation, each estimated transition probability i.e. each element of $\hat{M}_{ij}$ is set so as to maximise the log-joint likelihood of that observed sequence given M.
- For clarity, we can expand the log joint likelihood of the observed sequence:-

$$\ln p(s_1, \dots, s_t | M) = \ln\left[p(s_1)p(s_2|s_1)p(s_3|s_2)\dots p(s_t|s_{t-1})\right]$$

$$= \ln p(s_1) + \ln p(s_2|s_1) + \dots + \ln p(s_t|s_{t-1})$$

- Essentially, we are maximising over the sum of every transition's log-likelihood.
- Perhaps clear to think of an observed sequence as fixing the state random variable, so observed sequence is $(s_1 = \bar{s}_1, s_2 = \bar{s}_2, \dots, s_t = \bar{s}_t)$ i.e. $(\bar{s}_1, \bar{s}_2, \dots, \bar{s}_t)$
- As significant conceptual issues with compact notation, we expand the RHS for clarity from the outside

$$\underset{M}{\text{argmax}} \sum_{u=1}^{t-1} \sum_{i,j}^{S} \mathbb{1}(s_u = i, s_{u+1} = j) \ln M_{ij} = \underset{M}{\text{argmax}} \sum_{u=1}^{t-1} \left(\sum_{i,j}^{S} \mathbb{1}(s_u = i, s_{u+1} = j) \ln M_{ij} \right)$$

$$= \underset{M}{\text{argmax}} \Bigg(\underbrace{\sum_{i,j}^{S} \mathbb{1}(s_1 = i, s_2 = j) \ln M_{ij}}_{u=1 \text{ i.e. transition from } t=1 \to t=2} + \left(\sum_{i,j}^{S} \mathbb{1}(s_2 = i, s_3 = j) \ln M_{ij} \right) + \dots + \left(\sum_{i,j}^{S} \mathbb{1}(s_{t-1} = i, s_t = j) \ln M_{ij} \right) \Bigg)$$

($u = t$ i.e. transition from penultimate to final t)

- At each, e.g. $u = 1$:

$$\sum_{i,j}^{S} \mathbb{1}(s_1 = i, s_2 = j) \ln M_{ij} = \sum_{i}^{S} \sum_{j=1}^{S} \mathbb{1}(s_1 = i, s_2 = j) \ln M_{ij}$$

$$= \sum_{i=1}^{S} \mathbb{1}(s_1 = i, s_2 = 1) \ln M_{i1} + \mathbb{1}(s_1 = i, s_2 = 2) \ln M_{i2} + \dots + \mathbb{1}(s_1 = i, s_2 = S) \ln M_{iS}$$

$$= \Big(\mathbb{1}(s_1 = 1, s_2 = 1) \ln M_{11} + \mathbb{1}(s_1 = 1, s_2 = 2) \ln M_{12} + \dots + \mathbb{1}(s_1 = 1, s_2 = S) \ln M_{1S} \Big)$$

$$+ \Big(\mathbb{1}(s_1 = 2, s_2 = 1) \ln M_{21} + \dots + \mathbb{1}(s_1 = 2, s_2 = S) \ln M_{2S} \Big)$$

$$\vdots + \dots$$

$$+ \Big(\mathbb{1}(s_1 = S, s_2 = 1) \ln M_{S1} + \dots + \mathbb{1}(s_1 = S, s_2 = S) \ln M_{SS} \Big)$$

And similarly,

At $u=(t-1)$

$$\sum_{i,j}^{S} \mathbb{1}(s_{t-1}=i, s_t=j) \ln M_{ij} = \sum_{i=1}^{S}\sum_{j=1}^{S} \mathbb{1}(s_{t-1}=i, s_t=j)$$

$$= \sum_{i=1}^{S} \left(\mathbb{1}(s_{t-1}=i, s_t=1) \ln M_{i1} + \dots + \mathbb{1}(s_{t-1}=i, s_t=S) \ln M_{iS} \right)$$

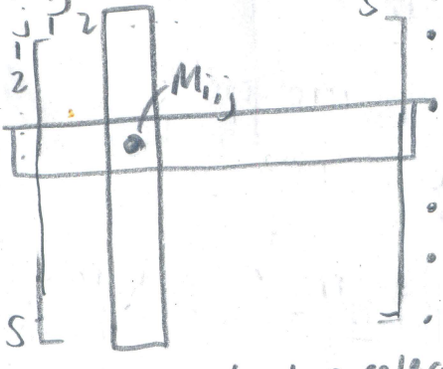
$$= \left(\mathbb{1}(s_{t-1}=1, s_t=1) \ln M_{11} + \dots + \mathbb{1}(s_{t-1}=1, s_t=S) \ln M_{1S} \right)$$

$$+ \dots$$

$$+ \left(\mathbb{1}(s_{t-1}=S, s_t=1) \ln M_{S1} + \dots + \mathbb{1}(s_{t-1}=S, s_t=S) \ln M_{SS} \right)$$

- Recall t indexes time, s is a state variable and i,j index states (realisations)
- Essentially, $\underset{M}{\text{argmax}} \sum_{u=1}^{t-1} \sum_{i,j}^{S} \mathbb{1}(s_u=i, s_{u+1}=j) \ln M_{ij}$ is doing the following:-
- For an observed sequence $(\bar{s}_1, \dots, \bar{s}_t)$, the outer summation, indexed by u, is the sum of over all transitions.
- And similarly, the inner summation, indexed by i and j, is the sum over all possible realisations of state over a time interval indexed by u (for a particular u). i.e. at two different time points.
- The indicators select the correct event, given we have observed the sequence $(\bar{s}_1, \dots, \bar{s}_t)$ and ~~not~~ all of them except one at each u index will be zero.
- The $\ln M_{ij}$ selects correct log-probability

- $\underline{M}$ =



- $M_{ij} = p(s_t=j \mid s_{t-1}=i)$
- $M_{i:} = p(s_t=j \mid s_{t-1})$ - each row of M is a probability distribution
- For an observed sequence $(\bar{s}_1, \bar{s}_2, \dots, \bar{s}_t)$
- u indexes a particular pair $(s_1=\bar{s}_1, s_2=\bar{s}_2)$

- Summation over indicators selects appropriate log probability defined by observed sequence at each u.
- Summation over all transitions
- $\hat{M}_{ML}(i,j)$ is an estimate i.e. a function of the data/sequence based on a particular realisation
- It is derived by setting up a constrained optimisation problem with the following constraints :- $M_{ij} \geq 0 \ \forall i,j$ and $\sum_{j=1}^{S} M_{ij} = 1$ (these impose restrictions on parameters i.e. M_{ij} and reduce degrees of freedom.)

from i:

$$\hat{M}_{ML}(i,j) = \frac{\sum_{u=1}^{t-1} \mathbb{1}(s_u = i, s_{u+1} = j)}{\sum_{u}^{t-1} \mathbb{1}(s_u = i)} = \frac{\#\text{ transitions from } i \to j}{\#\text{ total trans. from } i}$$

E.g. Model probability of rain tomorrow given rain today with over all time periods.

$$\frac{\#\{r \to r\}}{\#\{r\}}$$

Discrete first-order Markov chains - ~~stat~~ state and stationary distributions

Q Can we say at beginning i.e. at $t=0$ at what state we will be at step $t+1$

A At time step t, we have a probability distribution on which state we are in, $p(s_t = i)$.

Distribution on s_{t+1} is:

$$p(s_{t+1} = j) = \sum_{i=1}^{S} p(s_{t+1} = j, s_t = i) = \sum_{i=1}^{S} p(s_{t+1} = j \mid s_t = i)\, p(s_t = i) \quad (A)$$

Representing $p(s_t = i)$ with the row vector w_t (state distribution); then

$$\underbrace{p(s_{t+1} = j)}_{w_{t+1}(j)} = \sum_{i=1}^{S} \underbrace{p(s_{t+1} = j \mid s_t = i)}_{M_{ij}}\, \underbrace{p(s_t = i)}_{w_t(i)} \quad (B)$$

$$\Rightarrow w_{t+1}(j) = \sum_{i=1}^{S} M_{ij}\, w_t(i)$$

→ "Given a current state distribution w_t, distribution on next state is $w_t M$"

$$\Rightarrow \boxed{w_{t+1} = w_t M} \quad (on)$$

- w_1 is indicator if starting state known

$$\Rightarrow \boxed{w_{t+1} = w_1 M^t} \quad (on)$$

(via a recursive opening of above)

- vector indices changes as confusing (was formally indexed time previously)

arises from $p(X = x) = \sum_y p(X = x, Y = y)$ i.e. marginal of X (integral) is sum of joint probabilities over all outcomes of Y

$$\sum_y p(X = x, Y = y) = \sum_y p(X = x \mid Y = y) \cdot p(Y = y)$$ i.e. joint probability is product of conditional and prior on y

Probability weighted calculation summed over all states

Stationary distribution

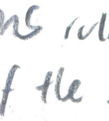
What happens when we project an infinite no. of steps outwards?

(i.e. what is long-run behaviour of this system?)

and are there any interesting statistical properties

Let $w_\infty = \lim_{t\to\infty} w_t$; then w_∞ is the stationary distribution of a Markov chain

If the following holds then w_∞ is the same vector $\forall w_0$ (convergence to same regardless of initial condition)

1. We can reach any state starting from any other state (irreducibility)
2. The sequence does not loop between states in a predefined pattern (aperiodic)
   i.e. no deterministic convergent transitions ruling out eg. 

This implies that the following property of the stationary distribution vector holds:

$$w_\infty = w_\infty M \quad \text{since } w_t \text{ is converging and } w_{t+1} = w_t M \quad (OTI)$$

This property can be restated in terms of eigenvalues and eigenvectors, i.e. the first eigenvector of M^T:

$$w_\infty = w_\infty M \Rightarrow I \times w_\infty^T = M^T w_\infty^T \Rightarrow M^T q_1 = \lambda_1 q_1$$

$$\Rightarrow \lambda_1 = 1 \text{ and } w_\infty = \frac{q_1}{\sum_{i=1}^{S} q_1(i)}$$

due to probability distribution constraints $M_{ij} \geq 0$ $\sum_{j=1}^{S} M_{i,j} = 1$; we have that first eigenvalue (greatest) is 1, with corresponding eigenvector.
by definition, the eigenvector has an L_2 norm of 1
we can get a stationary distribution with L_1 norm of 1. (?)

$$\|w_\infty\|_2^2 = 1 \Rightarrow \|w_\infty\|_1 = 1$$

we recover the stationary distribution, w_∞, by taking the first eigenvector of M^T and normalising it.
computationally more efficient than carrying out multiple matrix multiplications $w_\infty = w_0 M^t$ to use this method.

cation - ranking algorithm (Rasley)

stationary distribution of a Markov chain to rank objects
a - pairwise comparisons between objects
e entities from best to worst
truct random walk matrix on objects, ranking given by stationary distri.
kov chain is artificial modelling construct that allows meaningful interp.
tionary distribution, even if sequential aspect in data not used.

team is a Markov chain state

- Encourage transitions from losing to winning teams.
- Predicting state in future → better team is a more probable state
- Transitions only occur between teams that play each other
- Team A beats Team B ⇒ high P_{BA} and low P_{AB}
- Strength of the transition linked to the game score
- An interesting example of how these intuitions are codified into a model :-

Rankings -

- Initialise $\hat{M}$ as a matrix of zeros
- For a particular game, let j_1 be index of Team A, j_2 be index of Team B

update rules :-

$$\hat{M}_{j_1 j_1} \leftarrow \hat{M}_{j_1 j_1} + \mathbb{1}\{\text{Team A wins}\} + \frac{\text{points}_{j_1}}{\text{points}_{j_1} + \text{points}_{j_2}}$$

$$\hat{M}_{j_2 j_2} \leftarrow \hat{M}_{j_2 j_2} + \mathbb{1}\{\text{Team B wins}\} + \frac{\text{points}_{j_2}}{\text{points}_{j_1} + \text{points}_{j_2}}$$

$$\hat{M}_{j_1 j_2} \leftarrow \hat{M}_{j_1 j_2} + \mathbb{1}\{\text{Team B wins}\} + \frac{\text{points}_{j_2}}{\text{points}_{j_1} + \text{points}_{j_2}}$$

$$\hat{M}_{j_2 j_1} \leftarrow \hat{M}_{j_2 j_1} + \mathbb{1}\{\text{Team A wins}\} + \frac{\text{points}_{j_1}}{\text{points}_{j_1} + \text{points}_{j_2}}$$

(i) old value
(ii) indicator encoding result info
(iii) encodes fraction of points scored (quantitative)

- After processing games, let M be the matrix formed by normalising rows of $\hat{M}$ so they sum to 1

Example : 2014-2015 Basketball season

- An expert opinion poll of rankings averaged over 25 experts (ground truth)
- The Markov chain ranking outputs SCORE = stationary distribution w_{SS}

Intuitions :

- more flow to the good teams (have higher transition probability to those states)
- Not enough to win lots of games, but winning lots of games against other teams that have won lots of games
- Markov chain → find v_{SS} by finding q_1 and normalising ; then ranking accord. to probability
- Markov chain is purely data driven and statistical and arguably still recreates features of expert opinion poll.

Application - classification algorithm

Semi-supervised learning

- Imagine data with few labels
- use inherent structure in dataset to 'infer' classifications on unlabelled data
- use a Markov chain (one sol.)

$x_i \in \mathbb{R}^2$

Random walk classifier

- Define a classifier, where starting from a data point x_i,
  - Random walker moves from point to point
  - transition between nearby points have higher probability
  - Transition to a labelled point terminates the walk
  - Label of point x_i is the label of terminal point
- These intuitions can be encoded using the following possible random walk matrix

1. Let unnormalised transition matrix be

$$\hat{M}_{ij} = \exp\left\{-\frac{\|x_i - x_j\|^2}{b}\right\} = K \quad \text{(i.e. Gaussian proximity kernel matrix)}$$

2. Normalise rows of $\hat{M}$ to get M
3. If x_i has label y_i, redefine $M_{ii} = 1$ (i.e. terminate)

i.e. construct a transition matrix between initial point and all other points including itself

n x_i data points $\Rightarrow M$ and $\hat{M} \in \mathbb{R}^{n \times n}$ (vector); each point is analogous to a state

Markov chain properties - absorbing states, absorbing states distribution

We semi-supervised classification problem can be solved through an equivalent absorbing state Markov problem

Imagine we have S states. If $p(s_t = i | s_{t-1} = i) = 1$ then the i^{th} state is an absorbing state and we can never leave it

$$p(s_t = i | s_{t-1} = i) = 1 \Rightarrow i^{th} \text{ state is absorbing}$$

k absorbing states

Given an initial state $s_0 = j$ and a set of absorbing states $\{i_1, \dots, i_k\}$ what is the probability a Markov chain terminates at a particular absorbing state $p(s_t = i_k | s_0 = j)$ given that $p(s_t = i_k | s_{t-1} = i_k) = 1 \ \forall k$ and a transition matrix

We want to find a k-dimensional probability distribution

If we can calculate $p(s_t = i_k | s_0 = j) \ \forall k$ i.e. a distribution, then our semi-superv. classific problem

Framed in terms of semi-supervised classification, the answer will give a probability on the label of x_j.

Start a random walk at j, keep track of the distribution on states

w_0 is a vector of 0s with 1 in entry j as $s_0 = j$

For a transition matrix M, $w_{t+1} = w_t M$

We want $w_\infty = w_0 M^\infty$ (more accurately, $w_\infty = \lim_{t \to \infty} w_0 M^t$) or $w_\infty^{(j)} = w_0^{(j)} M^\infty$

Crucially, $w_\infty \neq w_\infty M$ means irreducibility does not hold, due to presence of absorbing states, and hence w_∞ is not the same regardless of starting state.

• Grouping the absorbing states and break the transition matrix M into quadrants:-

$$\underline{M} = \begin{matrix} (S-R) \\ R \end{matrix}\begin{Bmatrix} \underline{A} & \underline{B} \\ \underline{O} & \underline{I} \end{Bmatrix} \begin{matrix} \} (S-R) \\ \} R \end{matrix}$$

$\underline{A} \in \mathbb{R}^{(S-R)\times(S-R)}$ $\quad \underline{B} \in \mathbb{R}^{(S-R)\times R}$ $\quad$ and R - no. of absorbing states

$\underline{O} \in \mathbb{R}^{R\times(S-R)}$ $\quad I \in \mathbb{R}^{R\times R}$ $\quad S$ - no. of states

• $\underline{A}$ represents ~~prob~~ transition probabilities from non-absorbing → non absorbing states
• $\underline{B}$ represents transition probabilities from non-absorbing → absorbing states
• $\underline{O}$ represents transition probabilities from ~~non~~ absorbing → non absorbing (= 0 via definition of absorbing state ~~i.e. pfst~~
• $\underline{I}$ represents transition probabilities from absorbing → absorbing (= 1)
• Bottom half represents self-transitions of absorbing states.

Recall: $w_{t+1} = w_t M = w_{t-1} M^2 = \dots = w_0 M^{t+1}$

• We need to discern regularities in M^t:

evaluating at $t=2$: $M^2 = \begin{bmatrix} A & B \\ 0 & I \end{bmatrix}\begin{bmatrix} A & B \\ 0 & I \end{bmatrix} = \begin{bmatrix} A^2 & AB+B \\ 0 & I \end{bmatrix}$

evaluating at $t=3$: $M^3 = \begin{bmatrix} A & B \\ 0 & I \end{bmatrix}\begin{bmatrix} A^2 & AB+B \\ 0 & I \end{bmatrix} = \begin{bmatrix} A^3 & A^2B+AB+B \\ 0 & I \end{bmatrix}$

• Essentially, top-right quandrant can be represented as a sum of geometric series in matrices with first term I and common ratio A.

Absorbing state distribution

• matrix version of geometric series:-

$$M^t = \begin{bmatrix} A^t & \left(\sum_{n=0}^{t-1} A^n\right) B \\ 0 & I \end{bmatrix}$$

• It is stated without proof that:-

$$\text{ie } A^\infty = \lim_{t\to\infty} A^t = \underline{O}$$

and that $\sum_{n=0}^{\infty} A^n = (I-A)^{-1}$

$$\Rightarrow M^\infty = \lim_{t\to\infty} M^t = \begin{bmatrix} \underline{O} & (I-A)^{-1}B \\ \underline{O} & I \end{bmatrix}$$

• After an infinite # of steps $w_\infty = w_0 M^\infty = w_0 \begin{bmatrix} \underline{O} & (I-A)^{-1}B \\ \underline{O} & I \end{bmatrix}$

• Non-zero dimension of w_0 picks out a row of $(I-A)^{-1}B$.

- Matrix-vector product $w_0 \begin{bmatrix} \underline{0} & (\underline{I}-\underline{A})^{-1}\underline{B} \\ \underline{0} & \underline{I} \end{bmatrix}$

- w_0 is a (row?) vector of 0s except on starting state w_{0j} corresponding to $s_0 = j$
- w_0 selects a row of $(I-A)^{-1}B$ (the quadrant for non-absorbing → absorbing states)
(non-zero dimension)

- semi-supervised classification ⇒
  - Probability that a random walk started at x_j (corresponding to $s_0 = j$) terminates at the i^{th} absorbing state is given by: $[(I-A)^{-1}B]_{ji}$

Classification example

- see colour diagram
- Gaussian kernel normalised on rows; red and blue colour indicates distribution on terminal state for each starting point.
- Dependent on kernel width tuning i.e. (hyperparameter settings)
- Purple region - 50% of getting red or blue classification respectively
- In practice, manifolds and contours not known.

Hidden Markov models - motivation and intuition

Motivation

- Markov models can model sequential data
- We assumed :-
  - Each observation was the sequence of states
- To motivate Hidden Markov models (HMMs), we may treat each state as :-
  - defining a distribution on observations (i.e. corresponding to a probability distribution on an observable part of the sequence)

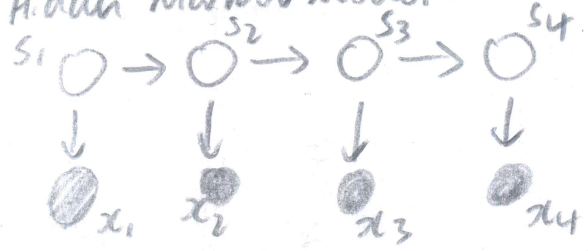
Representation of HMM

- A hidden Markov model treats a sequence of data differently
- Assume a hidden/unobserved/latent sequence of states (STT)
- An observation is drawn from a distribution associated with its state
- Our observations are realisations of a random variable whose conditional probability distribution is indexed/picked out by state

- Graphical models representation

Markov model

Hidden Markov model :-



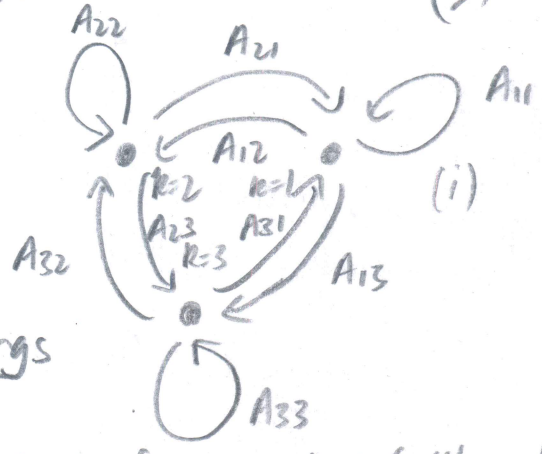
- Coloured - observed
- Blank - unobserved

(state-space model)

- Each obs. x_i conditioned on state of latent variable s_i (I)

Markov models - state representation

Imagine 3 possible states in $\mathbb{R}^2$
The data is a sequence of the positions
Since there are 3 unique positions, we
will give an index instead of co-ordinates
So in Markov models, we do not have embeddings
of states in a space, just state index.



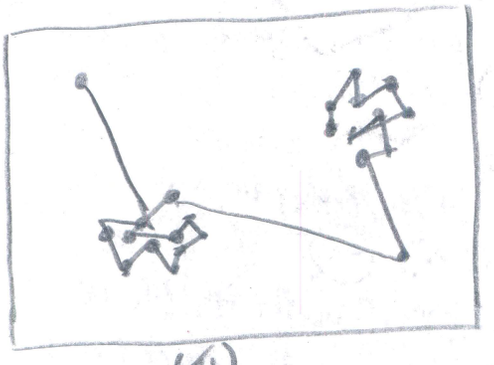
e.g. sequence $\{1,2,1,3,2,\ldots\}$ would map to a sequence of 2D vectors (although
no illustrative need.
A is a 3×3 transition matrix i.e. $A \in \mathbb{R}^{3\times3}$ and A_{ij} is probability of transitioning
from state i to state j.

Hidden Markov models - state representation
HMM - think of states as being embedded in a latent space (STT)
Imagine the same three states in Markov models, but each time the
state vectors have (their co-ordinates) been perturbed
The state sequence is still a set of indexes $\{1,2,1,3,2,\ldots\}$ of positions in $\mathbb{R}^2$.
However, if μ_1 is the position of state 1, we observe :-

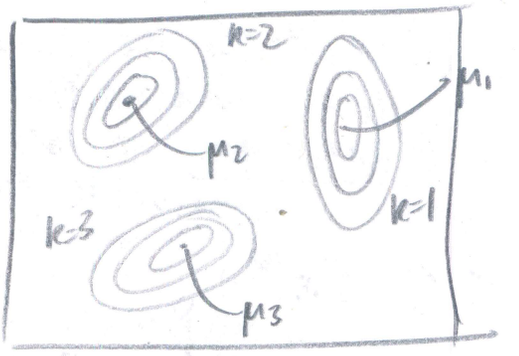
$$x_i = \mu_1 + \epsilon_i \quad \text{if } s_i = 1 \qquad \text{(STT)}$$

observed unobserved

- We have a similar state transition matrix $A \in \mathbb{R}^{3\times 3}$
- However, observed data is a sequence $\{x_1, x_2, x_3, \ldots\}$ where each $x_i \in \mathbb{R}^2$ is a random perturbation of the state it's assigned to $\{\mu_1, \mu_2, \mu_3\}$ (III)
- We do not directly observe the state, but its location (with) affle random perturbation.



(III)



(II)

(I): Markov state transition distribution for an unobserved state sequence

(II): state dependant distributions used to generate observations

(III): Data sequence

- In particular, ordering matters, in contrast to Gaussian mixture models (GMMs)
- Each point is realisation of a random variable according to a state dependant (conditional) probability distribution.

Hidden Markov models (HMMs)

Definition

- A hidden Markov model consists of:-
  - An SxS Markov transition matrix A for transitioning between S states
    i.e. $A \in \mathbb{R}^{S\times S}$
  - An initial state distribution π for selecting the first state
  - A state-dependant emission distribution $\underbrace{P(x_i|s_i=k) = P(x_i|\theta_{s_i})}_{(1)}$

Generative process

- Model generates a sequence $\{x_1, x_2, x_3, \ldots,\}$ by:-

1) Sampling the first state $s_1 \sim \text{Discrete}(\pi)$ and $x_1 \sim p(x|\theta_{s_1})$

2) Sampling the Markov chain of states, $s_i|\{s_{i-1}=k'\} \sim \text{Discete}(A_{k'},:)$
   and then the observation $x_i|s_i \sim p(x|\theta_{s_i})$

- Hidden Markov models are discete in state
- It is the choice of whether to model the data (observed) as continuous/discete that yields the continuous/discete HMMs
- Continuous HMM: $p(x|\theta_s)$ is a continuous distribution, often Gaussian
- Discete HMM: $p(x|\theta_s)$ is a discete distribution, θ_s, a vector of probabilities on discete set of observed values x_i

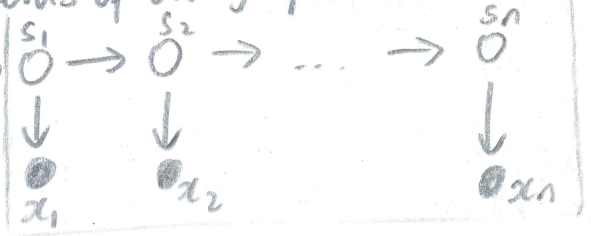
- We focus on discete case
- Let B be a matrix, where $B_{s,:} = \theta_s$ (from above)

(i): density function - represents probability distribution on observed x_i given at time i, state is R

- $B \in \mathbb{R}^{S \times V}$ and $x_i \in \{1, \dots, V\}$ i.e. observed data can take on values

$B = \begin{matrix} S \\ \{ \end{matrix} \begin{bmatrix} \cdot B_{i,j} \end{bmatrix}$ (V columns)

- B_{ij} = probability observation i comes from state j
- In terms of our graphical model representation



→ : given by A

↓ : given by B

O_{s_1} : given by π

Example - dishonest casino

- Discete hidden Markov model
- Two dice, one fair, one unfair (can be thought of as 2 r.vs, not drawn iid)
- At each roll, we keep current dice, or switch to the other
- We do not observe which dice is rolled, only the observe sequence of numbers rolled
- The first state is the fair dice
- The second state is the unfair dice
- Hidden state transition is selecting the dice

Transition matrix:

$$A = \begin{matrix} \\ s_1 \\ s_2 \end{matrix} \begin{matrix} \begin{matrix} s_1 & s_2 \end{matrix} \\ \begin{bmatrix} 0.95 & 0.05 \\ 0.10 & 0.90 \end{bmatrix} \end{matrix}$$

Initial

$\pi = \left[\frac{1}{2}, \frac{1}{2}\right]$

Emission matrix

$$B = \begin{matrix} \\ s_1 = \text{fair} \\ s_2 = \text{unfair} \end{matrix} \begin{matrix} \begin{matrix} v_1=1 & v_2 & v_3 & v_4 & v_5 & v_6=6 \end{matrix} \\ \begin{bmatrix} \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{10} & \frac{1}{10} & \frac{1}{10} & \frac{1}{10} & \frac{1}{10} & \frac{1}{2} \end{bmatrix} \end{matrix}$$

- Estimation problems for HMMs
- Recall we define a generative process, have observational data, and then infer parameters of generative process

State estimation

- Given an HMM $\{\pi, A, B\}$ and observation sequence $\{x_1, \dots, x_T\}$ Estimate state probability for x_i using forward-backward algorithm

$$p(s_i = R \mid x_1, \dots, x_T, \pi, A, B) \text{ or } p(\vec{s} \mid \underline{x}_1, \dots, \underline{x}_T, \underline{\pi}, A, B) \text{ (i)}$$

State sequence

- Given an HMM $\{\pi, A, B\}$ and observation sequence $\{x_1, \dots, x_T\}$ Estimate most probable state sequence using the Viterbi algorithm

$$s_1, \dots, s_T = \underset{\vec{s}}{\text{argmax}}\, p(s_1, \dots, s_T \mid x_1, \dots, x_T, \pi, A, B) \quad \text{(ii)}$$

Estimate parameters

- Given an observation sequence $(x_1, \dots, x_T)$, and no. of states S so $s \in \{1, \dots, S\}$
- Estimate HMM parameters π, A, B via maximum likelihood

$$\hat{\pi}_{ML}, \hat{A}_{ML}, \hat{B}_{ML} = \underset{\pi, A, B}{\text{argmax}}\, p(x_1, \dots, x_T \mid \pi, A, B) \quad \text{(iii)}$$

(i) conditional posterior probability of being in state k at time i, given entire observation sequence, cd HMM generative process

(ii): most probable state sequence $\{s_1, \ldots, s_T\}$, argmax of conditional posterior of entire sequence of states s_i, given data realisation sequence, HMM parametes - point estimate via MAP solution

(iii): At time i, which state were we in? Not known, as assumed latent, so only a distribution of states that we could have been in. forward-backward outputs is a conditional posterior probability (distribution) for all ~~st~~ time points i and states k.

Example - dishonest casino

state estimation and state sequence for dishonest casino (via forward-back cd Viterbi.)
π, A, B were first estimated to calculate these (not shown)
300 rolls (not shown)
shaded ⇒ loaded dice - unsheded ⇒ ~~to~~ unloaded dice (representing unobserved
rue state of the model)
iven the sequence of 300 rolls, the forward-backward algorithm will output
sequence of conditional posterior probabilities of a given state; ~~so~~ i.e.
arginal conditional posterior probabilities $p(s_i = \text{loaded} \mid x_{1:T}, \pi, A, B)$
iven the sequence of 300 rolls, the Viterbi algorithm estimates the most probable
te sequence
t as trivial as rounding marginal conditional posterior probabilities due to
ertial dependence
ost probable state sequence takes a more global, rather than local view
erbi algorithm is not just explaining data, but explaining a likely state sequence,
ransition of state sequences between states.
P estimate not ideal → (Q) why?

nating HMM parametes

MP
ete HMM case
observed sequence $\{x_1, \ldots, x_T\}$ be represented as $\vec{x}$; hidden states $\{s_1, \ldots, s_T\}$ as $\vec{s}$
atent sequence is not visible, expand marginal likelihood of observed as
likelihood of observed and latent, summed over all state realisations.

mise:-

$$p(\vec{x} \mid \pi, A, B) = \sum_{s_1=1}^{S} \cdots \sum_{s_T=1}^{S} p(\vec{x}, s_1, \ldots, s_T \mid \pi, A, B)$$

- AS $p(x) = \sum_y p(x, y)$

T summations, for each time index ~~i~~, each summation over S state realisations

$$= \sum_{s_1=1}^{S} \cdots \sum_{s_T=1}^{S} \left(\prod_{i=1}^{T} p(x_i \mid s_i, B)\, p(s_i \mid s_{i-1}, \pi, A) \right)$$

(i) (ii)

- via Markov property

(Q) Derivations

(i) Summing over all observations of log probability of observing data at time point i, given that we are in state k at time point i.
- Indicator $\mathbb{1}(s_i = k)$ selects correct probability
- As we focus on log joint-likelihood over observed $\vec{x}$ and latent $\vec{s}$, we have:-

$$\ln p(\vec{x}, \vec{s} \mid \pi, A, B) = \ln \left(\prod_{i=1}^{T} p(x_i \mid s_i, B) p(s_i \mid s_{i-1}, \pi, A) \right)$$

$$= \sum_{i=1}^{T} \ln p(x_i \mid s_i, B) + \sum_{i=1}^{T} \ln p(s_i \mid s_{i-1}, \pi, A)$$

$$= \sum_{i=1}^{T} \ln p(x_i \mid s_i, B) + \ln p(s_1) + \sum_{i=2}^{T} p(s_i \mid s_{i-1}, \pi, A)$$

observations (A) — initial state (B) — Markov chain (C)

(ii) indicator $\mathbb{1}(s_1 = k)$ selects correct log probability of π_k (initial)

(iii) remaining portion of sequence: At state ($i = 2$) to end (T), we have indicator $\mathbb{1}(s_{i-1} = j, s_i = k)$ of being in state j at $(i-1)$ and transitioning to state k at time (i) times log probability of that transition $\ln A_{j,k}$, summed over all state transitions to select correct one.

- Notice that (i), (ii), (iii) are just formalised ways of writing (A), (B), (C) in our context - don't be put off by summations and indicators!
- Results (i.e. update rules for $\hat{\pi}, \hat{A}, \hat{B}$) are taken from Rabiner (1989)

Preliminary observations

1. $\mathbb{E}_q(\mathbb{1}(x \in A)) = q(x \in A)$ i.e. expectation of an indicator of an event $x \in A$ is equal to probability of that event according to the distribution

$$\mathbb{E}_q[\ln p(\vec{x}, \vec{s} \mid \pi, A, B)] = \mathbb{E}_q \left(\sum_{i=1}^{T} \sum_{k=1}^{S} \mathbb{1}(s_i = k) \ln B_{k, x_i} + \sum_{k=1}^{S} \mathbb{1}(s_1 = k) \ln \pi_k + \sum_{t=1}^{T} \sum_{j=1}^{S} \sum_{k=1}^{S} \mathbb{1}(s_{i-1} = j, s_i = k) \ln A_{j,k} \right)$$

$$= \sum_{i=1}^{T} \sum_{k=1}^{S} \mathbb{E}_q(\mathbb{1}(s_i = k)) \ln B_{k, x_i} + \sum_{k=1}^{S} \mathbb{E}_q(\mathbb{1}(s_1 = k)) \ln \pi_k + \sum_{t=1}^{T} \sum_{j=1}^{S} \sum_{k=1}^{S} \mathbb{E}_q(\mathbb{1}(s_{i-1} = j, s_i = k)) \cdot \ln A_{j,k}$$

$\mathbb{E}_q(\mathbb{1}(s_i = k)) = p(s_i = k \mid \vec{x}, \pi, A, B)$

$\mathbb{E}_q(\mathbb{1}(s_1 = k)) = p(s_1 = k \mid \vec{x}, \pi, A, B)$

$\mathbb{E}_q(\mathbb{1}(s_{i-1} = j, s_i = k)) = p(s_{i-1} = j, s_i = k \mid \vec{x}, \pi, A, B)$

as $\mathbb{E}_q(\mathbb{1}(x \in A)) = q(x \in A)$

(i): $p(x_i|s_i;B) = B_{s_i,x_i}$, s_i indexes the distribution, x_i indexes the observation
- conditional distribution of observed data x_i given hidden state s_i and emission probability matrix B.

(ii) $p(s_i|s_{i-1},\pi,A) = A_{s_{i-1},s_i}$ (or π_{s_1}) since $\{s_1,\dots,s_T\}$ is a Markov chain
- probability of state transition from state at $(t-1)$ to t

Log-likelihood:

- Maximising $p(\vec{x}|\pi,A,B)$ is tricky due to log-sum form:-

$$\ln p(\vec{x}|\pi,A,B) = \ln \underbrace{\sum_{s_1=1}^{S} \cdots \sum_{s_T=1}^{S}}_{T \text{ times}} \prod_{i=1}^{T} p(x_i|s_i,B)\, p(s_i|s_{i-1},\pi,A)$$

latent

(on) - If we know, can observe, or estimate the state sequence $\vec{s} = \{s_1, s_2, \dots, s_T\}$, problem would become easier i.e. treat latent sequence $\vec{s}$ as missing data, rewrite log-marginal likelihood of observed sequence $\vec{x}$ as joint likelihood over observed $\vec{x}$ and latent sequence $\vec{s}$.

(n) - Recall we can calculate conditional posterior of entire state sequence $\vec{s}$, or important parts of it, given HMM $\{\pi,A,B\}$ and observation sequence $\vec{x}$ via the forward-backward algorithm i.e. $p(\vec{s}|\vec{x},\pi,A,B)$, which is not shown here, but see Rabiner (1989) on how this is calculated. (q)

We can use EM algorithm in context of above two points.

EM Algorithm for HMM - high level overview

Strategically, the EM-algorithm for HMMs, involves using the forward-backward algorithm to calculate $q(\vec{s}) = p(\vec{s}|\vec{x},\pi,A,B)$ as an additional intermediary step during the E-step.

- Remember that in this course, letter/s puts parameters on RHS of conditional $\not\Rightarrow$ necessarily that it means conditioning on parameter (check!), merely means parametrised by $p(\vec{x}|\vec{s};A,B)$

E-step: using $q(\vec{s}) = p(\vec{s}|\vec{x},\pi,A,B)$, calculate:

$$\mathcal{L}(\vec{x},\pi,A,B) = \mathbb{E}_q\left[\ln p(\vec{x},\vec{s}|\pi,A,B)\right]$$

M-step: Maximise $\mathcal{L}$ with respect to π, A, B, ~~this~~
tricky, as we need to take expectation using $q(\vec{s})$ of :-

$$\ln p(\vec{x},\vec{s}|\pi,A,B) = \sum_{i=1}^{T}\sum_{k=1}^{S} \underbrace{\mathbb{1}(s_i=k)\ln B_{k,x_i}}_{\text{observations (i)}} + \sum_{k=1}^{S} \underbrace{\mathbb{1}(s_1=k)\ln \pi_k}_{\text{initial state (ii)}}$$

$$+ \sum_{i=2}^{T}\sum_{j=1}^{S}\sum_{k=1}^{S} \underbrace{\mathbb{1}(s_{i-1}=j, s_i=k)\ln A_{j,k}}_{\text{Markov chain (iii)}}$$

(log of joint-likelihood over observed $\vec{x}$ and latent $\vec{s}$)

E-step:

- The forward backward algorithm estimates the conditional posterior of the entire state sequence i.e. $q(\vec{s}) = p(\vec{s}|\vec{x}, \pi, A, B)$ and outputs:-

$\gamma_i(k)$ - posterior probability that $s_i = k$ where $\gamma_i \in \mathbb{R}^S$

$\xi_i(j,k)$ - posterior probability that $s_{i-1} = j$ and $s_i = k$ $\xi_i \in \mathbb{R}^{S \times S}$

with all elements of γ_i and $\xi_i(j,k)$ summing to 1 and being non-negative, and where both vary over i.

- Given these values, the E-step is:-

with $q(\vec{s}) = p(\vec{s}|\vec{x}, \pi, A, B)$ i.e setting prior over latent variable sequence $q(\vec{s})$ as the conditional posterior of the entire state sequence (or important parts of it), which is calculated by the forward-backward algorithm.

- Hence

$$\mathcal{L}(\vec{x}, \pi, A, B) = E_q(\ln p(\vec{x}, \vec{s}|\pi, A, B))$$

$$= \sum_{R=1}^{S} \gamma_1(k) \ln \pi_R + \sum_{i=2}^{T} \sum_{j=1}^{S} \sum_{R=1}^{S} \xi_i(j,k) \ln A_{j,R} + \sum_{i=1}^{T} \sum_{k=1}^{S} \gamma_i(k) \ln B_{k,x_i}$$

initial state — Markov chain — observations

- We can now maximise $\mathcal{L}$ with respect to π, A, B by taking derivatives and with constraints arising from using probability distributions

M-step:

- Updates for HMM parameters, after calculating γ_i and ξ_i in E-step

$$\pi_k \sim \frac{\gamma_1(k)}{\sum_j \gamma_1(j)} \qquad A_{j,R} = \frac{\sum_{i=1}^{T} \xi_i(j,R)}{\sum_{i=1}^{T} \sum_{\ell=1}^{S} \xi_i(j,\ell)} \qquad B_{k,v} = \frac{\sum_{i=1}^{T} \gamma_i(k) \mathbb{1}\{x_i = v\}}{\sum_{i=1}^{T} \gamma_i(k)} \quad (\dagger\dagger)$$

- Updates understood as follows:-

- $A_{j,R}$ is the expected fraction of transitions $j \to k$ when we start at j
  - numerator - expected count of transitions $j \to k$
  - denominator - expected total no of transitions from j
- $B_{k,v}$ is the expected fraction of data coming from state k and equal to v
  - numerator - expected no. of observations $= v$ from state k
  - denominator - expected total no. of observations from state k

- π - similar interpretation to A
- Denominators provide normalisation constants so rows of π, A, B sum to unity

(††) This is maximum-likelihood solution for an HMM with one sequence

Generalisation to N sequences

- Usually, we'll have multiple sequences that are modelled by an HMM.
- Updates for HMM parameters with N sequences are:-

$$\pi_R = \frac{\sum_{n=1}^{N} \gamma_1^n(k)}{\sum_{n=1}^{N} \sum_j \gamma_1^n(j)} \qquad A_{j,k} = \frac{\sum_{n=1}^{N} \sum_{i=2}^{T_n} \xi_i^n(j,k)}{\sum_{n=1}^{N} \sum_{i=2}^{T_n} \sum_{\ell=1}^{K} \xi_i^n(j,\ell)} \qquad B_{k,v} = \frac{\sum_{n=1}^{N} \sum_{i=1}^{T_n} \gamma_i^n(k) \mathbb{1}\{x_i = v\}}{\sum_{n=1}^{N} \sum_{i=1}^{T_n} \gamma_i^n(k)}$$

Modifications:-

- Each sequence can be of different, arbitrary length T_n
- Each sequence has its own set of γ and ξ values, indexed by n
- We sum over the sequences, with interpretation the same.

Speech recognition application

- HMMs are a fundamental method for speech recognition

Problem

- Given speech in the form of an audio signal, determine the words spoken

Method

- HMMs model transitions between phonemes
- Words are broken into small sound units (phonemes). The states in the HMM are intended to represent phonemes
- Incoming sound signal is transformed into a sequence of vectors via feature extraction. Each vector x_i is indexed by a time step i.
- The sequence $\vec{x} = x_{1:T}$ of feature vectors is the data used to estimate HMM parameters

Phoneme models

- A phoneme is the defined as the smallest unit of sound in a language that distinguishes between distinct meanings e.g. zero - z ih r ow
- Around 50 phonemes in English
- Each state defines a distribution on the sound associated with the phoneme
- HMM estimates a phoneme for each state ($x_i = \mu_i + \epsilon_i$)
- Assume voice is making the same sound, regardless of durations each phoneme is pronounced or accounting for natural variation.

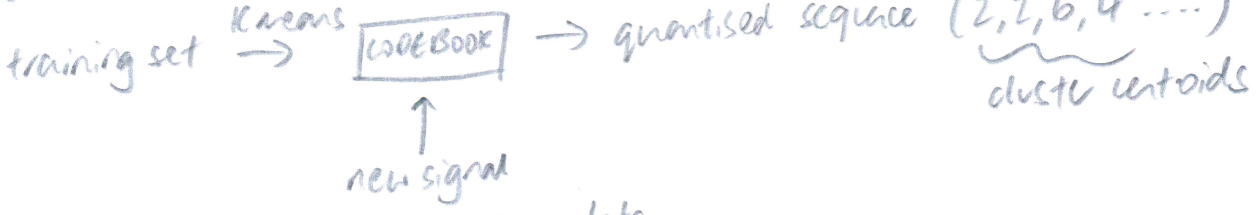
Pre-processing speech

- A speech signal is measured as amplitude over time
- The signal is typically transformed into features via a Fourier transform, breaking down the frequency content of the signal in a sliding-time window.
- At different points in time, we expose an 50ms window, examine audio signal and extract feature i.e. frequency content
- MFCCs i.e. mel-frequency cepstrum used to reduce this to a sequence of 40-d vectors

- Pre-processing ⇒ sequence of vectors capturing frequency content

Data quantisation

- We could work with extracted features and learn a Gaussian distribution for each state i.e. a Gaussian HMM where each state has a D dimensional MVG associated.
- To transition to a discrete HMM, we can perform vector quantisation using a codebook learned by K-means.



- K-means often used for discretising data
- For a sequence of D-dimensional feature vectors, K-means creates a codebook where each vector marks off a space in $\mathbb{R}^D$.
- Quantise each column by assigning each column to nearest centroid, report sequence of centroids rather than sequence of D-dimensional continuous vectors

Application - simple speech recognition

- These models and problems can become more complex
- Imagine a simple automated phone conversation using question/answer format

Training Data: Quantised feature sequences of words (i.e. many instances of a person saying 'yes', 'no', 'travel' etc.)

Learn: An HMM for each word using all training sequences of that word

- Have an HMM representation of all possible words that could occur

Automated system asks question, it hears yes, extracts features from audio, quantises the sequence, so what is word that corresponds to this quantised sequence?

- Therefore we calculate the probability of that sequence given the HMM for a particular word.
- If we want to query whether a word spoken corresponds to w, we take the HMM for word w, calculate probability of sequence given HMM for word w (via forward backward algorithm integrating out state sequence)
- We get probability of that sequence given HMM for that particular word
- We do this for every word and assign the label for this sequence to be the most probable word.
- So we pick the HMM that most likely generated the sequence we've observed, and that HMM will correspond to a word and we declare that the word spoken was particular word

Predict. Let w index the word. Predict the word of a new sequence using

$$w_{new} = \arg\max_w p(\vec{x}_{new} | \pi_w, A_w, B_w)$$ (requires forward-backward?) (?)

- This is a Bayes classifier with a uniform prior on the word.
- Could improve by learning frequencies of different words and having a prior probability on each of the words.
- More complicated than previous Bayes classifier.
- Here, each word is a class, and we have to learn a class-specific model and class-conditional model (as opposed to simple MVG distribution over a vector)

Alternative approaches in context of Bayes classifier might:-

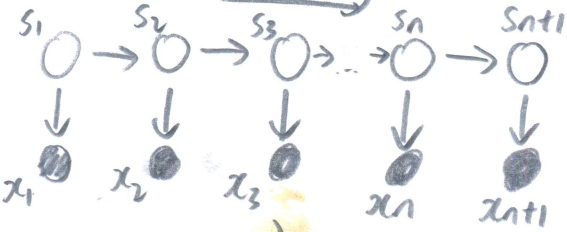
- Learning a class-conditional discrete HMM
- Could try Gaussian mixture model, which does not take into account temporal information - treats original vectors as though no time-dependence and learned word/class specific Gaussian mixture model.
- Anticipate HMM to do better ~~due to sequential~~ choice of class conditional density/model due to use of sequential information.

Markov models

The sequence $(s_1, s_2, s_3, \ldots, s_T)$ has the Markov property if for all t:

$$p(s_t | s_{t-1}, \ldots, s_1) = p(s_t | s_{t-1})$$

- Markov models assumed a finite state space and also discrete, meaning we can define an indexing such that $s \in \{1, \ldots, S\}$



Hidden Markov models

- HMMs modify this by assuming the sequence of states is an unobserved, latent process.
- An observation x_t is associated with each s_t where $x_t | s_t \sim p(x | \theta_{s_t})$.

The latent states follow a first-order Markov process

So the state sequence $(s_1, \ldots, s_T)$ follows a Markov chain, but is unobserved.

observations are drawn from a state-dependant distribution

At each time point, we observe a value x_n, drawn from a probability distribution with parameter dependent on state.

The probability distribution over observations are the from the same distribution family, but the parameters are state-dependent.

In both our treatments of the Markov model and HMM, whilst thee may have been choices made regarding continuity/discreteness of the data (representation), we assumed a discrete state space

For the Markov chain, our states corresponded to positions in $\mathbb{R}^d$ (vectors)

For the (continuous-variable) HMM, we modelled our observations x_i as perturbations of the latent state of the Markov chain i.e. $x_i = s_i + \epsilon_i$

However, the unobserved state sequence $s_{1:T}$ is still a discrete-state Markov chain.

Discrete state Markov models allow for the state to model a data point x_i. (Markov classifier), an object (object ranking), a link destination (internet search engines)

Discrete-state HMMs allow for simplification of complex data; so sequences of discrete and continuous data may be modelled as coming from a number of distributions.

Continuous-state (hidden) Markov models.

Questions:- are they mathematically convenient/tractable?; what can we gain?

Continuous-state Markov models extend the state space to a continuous domain instead of $s \in \{1, \ldots, S\}$; it can take any value in $\mathbb{R}^d$

States reside in a continuous domain; (and so infinitely many states)

Simplest example is the Brownian motion/random walk process:-

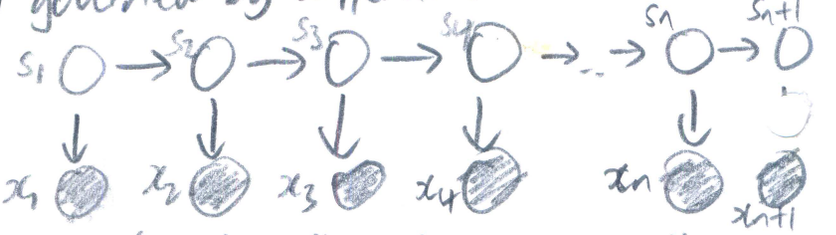
$s_t = s_{t-1} + \epsilon_t \quad \epsilon_t \sim N(0, \sigma I)$ i.e. iid Gaussian noise

- This is a continuous state random walk model, satisfying the Markov property, and where each successive state is a perturbed version of the previous state.
Linear Gaussian Markov Model (Kalman filter)
- Most basic continuous-state HMM is known as the linear Gaussian Markov model, or the Kalman filter.
- Specified by:

$$s_t = C s_{t-1} + \varepsilon_{t-1} \qquad s_t \in \mathbb{R}^p \qquad \varepsilon_t \overset{iid}{\sim} N(0, Q) \qquad (11)$$

$$x_t = D s_t + \varepsilon_t \qquad x_t \in \mathbb{R}^d \qquad \& \; \varepsilon_t \sim N(0, V)$$

- where s_t is a continuous-state latent Markov process $\forall t$
x_t is a continuous-valued observation $\forall t$
ε_t is process noise and ε_t is measurement noise
- There is a continuous functional form for the state and data processes.
- Noise in process and in measurement governed by different covariance matrices Q and V respectively
- Graphical models representation:



- Whilst the graphical models representation is the same as the HMM (discete state), which will be discussed later, the difference is that :-
- Both s_t and x_t are from continuous distributions.
- Applications of the Kalman filter in tracking moving objects, automatic control systems, and in economics and finance.

Application - tracking

- We get very noisy measurements of an object's position in time $x_t \in \mathbb{R}^2$
- The time varying state vector $s = [pos_1, vel_1, acc_1, pos_2, vel_2, acc_2]^T$
- Motivated by the underlying physics, we can write explicitly :-

$$s_{t+1} = \underbrace{\begin{bmatrix} 1 & \Delta t & \frac{1}{2}(\Delta t)^2 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & e^{-\alpha \Delta t} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & \frac{1}{2}(\Delta t)^2 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & e^{-\alpha \Delta t} \end{bmatrix}}_{\equiv C} s_t + \varepsilon_t \quad \text{and} \quad x_{t+1} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}}_{\equiv D} s_{t+1} + \varepsilon_{t+1}$$

- Each dimension in space corresponds to three states
- 2-dimensional locations $\Rightarrow$ 6 dimensional latent state vector
- First state evolution equation would specify 1st dim position in time (t+1) as a function of previous (time t) position, velocity, acceleration in 1st dimen.
- The matrix D is only selecting position along 1st and 2nd dimension of the previous state vector
- s_t not only approximates where the target is, but where it's going

- We will use an inference algorithm to predict:-
  i) location
  ii) underlying state (capturing position, velocity, acceleration as fns of time)
- See diagram - (Q)
- Line - actual trajectory of the object as function of time.
- Black dots - measurements of location at time points; ordering not visible, noisy
- Blue line - velocity vector in dimension 1 and 2 (estimate of direction and speed object is moving)
- Red dot - 1st or 4th dimension of state vector (i.e. pos_1, pos_2)
- We (estimate?) location, direction and speed of movement of object as a function of time

Estimation problem - Kalman filter

- We are given a sequence of observations $(x_1 = \bar{x}_1, x_2 = \bar{x}_2, \dots, x_t)$ with each $x_t \in \mathbb{R}^d$
- Goal: learn a state sequence $(s_1, s_2, s_3, \dots, s_t)$
- All distributions are Gaussian:-

$$p(s_{t+1} = s | s_t) = N(C s_t, Q) \quad \text{and} \quad p(x_t = x | s_t) = N(D s_t, V)$$

- With C and D given by physics in previous example (given by design)
- Q and V have to be estimated.
- With discrete-state HMM, we wanted to estimate parameters π, A, B, corresponding to the initial-state distribution, state transition matrix, and state-dependant distri. on discrete data.
- In the LGMM, each state transition is to a brand new state, so each s_t has its own unique probability distribution.
- In the LGMM, each state is unique amongst a continuous space and so the distribution on x_t is different for each $t \Rightarrow$ unique parameters for every value of x.
- (OR) we can learn 2 posterior distributions:

(OR) 1) Kalman filtering: $p(s_t | x_1, \dots, x_t)$: posterior distribution over current state given past observations.

2) Kalman smoothing: $p(s_t | x_1, \dots, x_T)$: posterior distribution on each latent state in the sequence given all including future observations

$x_1, \dots, x_t, \dots x_T$

Kalman filter - setup

Goal: Estimate the sequence of ~~dts~~ time-evolving posterior distributions of the hidden state, $p(s_t | x_1, \dots, x_t)$, given a sequence of observed data and ~~the model~~ $(x_1, \dots, x_t)$ and the model

$$s_{t+1} | s_t \sim N(C s_t, Q) \quad \text{and} \quad x_t | s_t \sim N(D s_t, V)$$

- This is the linear Kalman filtering problem, often used for tracking

Setup

using Bayes rule:-

$$p(s_t|x_1,\dots,x_t) \propto p(x_t|s_t)\,p(s_t|x_1,\dots,x_{t-1})$$

(?) why not $p(x_1,\dots,x_t|s_t)\cdot p(s_t)$

and represent the prior $p(s_t|x_1,\dots,x_{t-1})$ as a marginal distribution

$$p(s_t|x_1,\dots,x_{t-1}) = \int p(s_t, s_{t-1}|x_{1:t-1})\,ds_{t-1}$$

(marginalising over joint condit. distribution on s_t and s_{t-1})

$$= \int p(s_t|s_{t-1}, x_{1:t-1})\,p(s_{t-1}|x_{1:t-1})\,ds_{t-1} \quad \text{(ii)}$$

$$= \int p(s_t|s_{t-1})\,p(s_{t-1}|x_{1:t-1})\,ds_{t-1} \quad \text{(iii)}$$

(ii) - decomposing joint conditional distri $p(s_t, s_{t-1}|x_{1:t-1})$ into a product of conditional distribution on s_t, given s_{t-1} and sequence $x_{1:t-1}$ AND a posterior distribution on state s_{t-1} (i.e. at previous time point), given observation sequence $x_{1:t-1}$ - similar to online learning for Bayes rule; where the posterior at the previous time point forms the prior at the next time point

(iii) $p(s_t|s_{t-1}, x_{1:t-1})$ simplifies to $p(s_t|s_{t-1})$ because given state at $(t-1)$, s_{t-1}, state at time t, s_t is conditionally independent of the sequence up to $(t-1)$, $x_{1:t-1}$, so we can drop $x_{1:t-1}$ from the conditioning.

- Putting together our decomposition:-

$$p(s_t|x_1,\dots x_t) \propto \underbrace{p(x_t|s_t)}_{N(Cs_t, V)} \int \underbrace{p(s_t|s_{t-1})}_{N(Cs_{t-1}, Q)}\, \underbrace{p(s_{t-1}|x_1,\dots,x_{t-1})}_{?}\, ds_{t-1}$$

Comments

- posterior distribution at time t, given entire sequence of observations up to time t $p(s_t|x_1,\dots,x_t)$ is proportional to the likelihood of the observation at time t, given the state at time t, $p(x_t|s_t)$ multiplied by the prior, $p(s_t|x_1,\dots,x_{t-1})$
- The prior $p(s_t|x_1,\dots,x_{t-1})$ has been written as the marginal of the transition distribution from state $(t-1)$ to state (t), $p(s_t|s_{t-1})$ multiplied by posterior of previous state at the last time point $p(s_{t-1}|x_1,\dots,x_{t-1})$ which we still do not know.
- LHS - contains conditional posterior on s_t; RHS contains conditional posterior on s_{t-1}
- Hints at some kind of recursivity; if we can solve for posterior at previous time we can solve for posterior at current time.

- Integral on RHS in closed form gives suggestions on distribution candidates for tractability
- Require product of likelihood and prior to lead to a known posterior (outer) i.e. conjugacy between $p(x_t|s_t)$ and $\int p(s_t|s_{t-1})\,p(s_t|x_1,\dots,x_{t-1})\,ds_{t-1}$
- Makes future calculations of s_{t+1} easy/tractable
- surprisingly (!) - hypothesise $p(s_{t-1}|x_{1:t-1})$ as Gaussian i.e. $N(\mu,\Sigma)$

Step #1 - Calculate marginal for prior

- Hypothesise temporarily that unknown distribution is Gaussian:

$$p(s_t|x_1,\dots,x_t) \propto \underbrace{p(x_t|s_t)}_{N(Ds_t,V)} \int \underbrace{p(s_t|s_{t-1})}_{N(Cs_{t-1},Q)}\,\underbrace{p(s_{t-1}|x_1,\dots,x_{t-1})}_{N(\mu,\Sigma)\text{ by hypothesis}}\,ds_{t-1}$$

(*) Bayes th. + cond/marg Gaussian results ↗

(i) (Q) how?

- Property of Gaussians is that marginals are still Gaussian:-

$$\int N(s_t|Cs_{t-1},Q)\,N(s_{t-1}|\mu,\Sigma)\,ds_{t-1} = N(s_t|C\mu,Q+C\Sigma C^T)$$

- Involves multiplying out and integrating out with respect to ~~t-1~~ s_{t-1}.
- We know C and Q by design and μ and Σ (by hypothesis)
- Key point is that our hypothesis has yielded a calculable closed-form solution for the prior (a marginalisation)
- Integrating out s_{t-1} $\Rightarrow$ new mean of $C\mu$ and covariance $(Q+C\Sigma C^T)$

Step#2 - Calculate the posterior

- This step hinges on whether likelihood and prior (obtained via hypothesis and marginalisation) can be normalised and calculated.
- Substituting our marginalised distribution for prior:-

$$\underbrace{p(s_t|x_1,\dots,x_t)}_{\text{posterior}} \propto \underbrace{N(x_t|Ds_t,V)}_{\text{likelihood}}\,\underbrace{N(s_t|C\mu,Q+C\Sigma C^T)}_{\text{prior}}$$

- Posterior $p(s_t|x_1,\dots,x_t)$ is then just Gaussian [given our hypothesis!]

$$p(s_t|x_1,\dots,x_t) = N(s_t|\mu',\Sigma')$$

$$\Sigma' = \left[(Q+C\Sigma C^T)^{-1} + D^TV^{-1}D\right]^{-1}$$

$$\mu' = \Sigma'\left(D^TV^{-1}x_t + (Q+C\Sigma C^T)^{-1}C\mu\right)$$

- Note that these new parameters are all functions of what we already know!

Gaussian assumption

- we have closed form solution for conditional posterior $p(s_t|x_1,\dots,x_t)$
- An assumption of a Gaussian on the prior has yielded a final posterior which is Gaussian and has new mean and covariance.
- Essentially, our observations have led to a hypothesis that links posterior distributions between time t and $(t-1)$; similar to the kind of assumptions one makes in inductive proofs in mathematics.
- This yields a recursive chain that can be opened up to the start (initial prior).
- (Ctd): we only need to define a Gaussian prior on the first state to make the chain e.g.

$$p(s_0) \sim N(0, I)$$ or any other variant of some distribution family

- with only this step; all future calculations are in closed form.

Predictive distribution

- we now know how to update entire sequence of posterior distributions $p(s_t|x_{\dots},x_t)$
- How about predicting x_{t+1}?
- Form a marginal predictive distribution on our next observation :-

$$p(x_{t+1}|x_1,\dots,x_t) = \int p(x_{t+1}, s_{t+1}|x_{1:t})\, ds_{t+1}$$

marginal predictive distribution

$$= \int p(x_{t+1}|s_{t+1}, x_{1:t})\, p(s_{t+1}|x_{1:t})\, ds_{t+1}$$

(As knowing state s_{t+1}, observation x_{t+1} is independent of all other data $x_{1:t}$)

$$= \int p(x_{t+1}|s_{t+1})\, p(s_{t+1}|x_{1:t})\, ds_{t+1}$$

$$= \int p(x_{t+1}|s_{t+1})\, p(s_{t+1}|x_1,\dots,x_t)\, ds_{t+1}$$

$$= \int p(x_{t+1}|s_{t+1}) \int p(s_{t+1}, s_t|x_{1:t})\, ds_t\, ds_{t+1}$$

(similar arg for why $p(s_{t+1}|s_t, x_{1:t}) = p(s_{t+1}|s_t)$)

$$= \int \underbrace{p(x_{t+1}|s_{t+1})}_{= N(x_{t+1}|Cs_{t+1}, V)} \int \underbrace{p(s_{t+1}|s_t)}_{N(s_{t+1}|s_t, Q)}\, \underbrace{p(s_t|x_{1:t})}_{N(s_t|\mu', \Sigma')}\, ds_t\, ds_{t+1}$$

- (Ctd) Integrating out s_t and s_{t+1}
- Mean and covariance of predictive distribution not specified here
- Marginal predictive distribution is still Gaussian (with integral performed twice)

Kalman filter-algorithm

- Kalman filtering algorithm can be run in real time

Set initial state distribution $p(s_0) = N(0, I)$

- Prior to observing each $x_t \in \mathbb{R}^d$, predict via predictive distribution :-

$$x_t \sim N(\mu_t^x, \Sigma_t^x) \quad \text{i.e. } p(x_t | x_1, \dots, x_{t-1})$$

- After observing each new $x_t \in \mathbb{R}^d$, update

$$p(s_t | x_1, \dots, x_t) = N(\mu_t^s, \Sigma_t^s) \quad \text{i.e. conditional posterior on underlying state, given data up to time } t$$

Tracking example

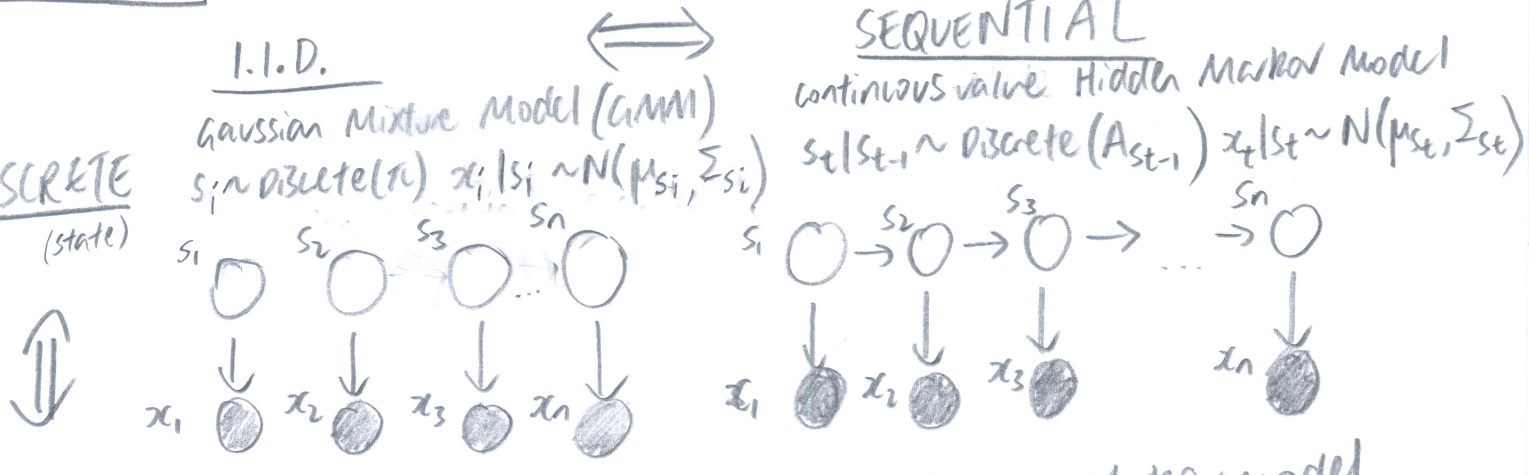
- see diagram (2)
- Green - true trajectory; blue - observed trajectory; red - state distribution
- Intuitions: . Prior distribution adds Q to covariance

$$p(s_t | x_1, \dots, x_{t-1}) = N(s_t | C\mu, Q + C\Sigma C^T)$$

- This adds drift/variance to latent states so that it does not converge to a point estimate as t increases.
- Allows state s_t to drift away from s_{t-1}
- Prior constrains s_t (state), but x_t pulls state in direction so that s_t is pulled away by x_t, so it can model a little better.
- ~~Prior~~ Posterior distribution $p(s_t | x_1, \dots, x_t)$, x_t 'pulls' distribution away (?)

Comparison of models using graphical models representation

- Formulation of models via graphical models representations can reveal deep relationships (between HMMs and LDSs) even if they were developed independently



(A bit messy)

Summary of generative processes:- (cluste)

i) GMM - At time t, we pick a state $s_t = S$, and therefore the relevant Gaussian out of S possible Gaussians, with probability given from a discrete distribution $\underline{\pi}$, which is a vector of mixing weights corresponding to a probability distribution over our S Gaussians.
- Given the state/cluste s_t, which indexes a Gaussian $N(\mu_{s_t}, \Sigma_{s_t})$, then the observation x_t given s_t, and given the component assigned at time t, is generated from that Gaussian with mean μ_{s_t} and covariance Σ_{s_t}.

i) continuous variable HMM
- At time t, the state s_t is generated from a discrete distribution where we select the probability distribution indexed by the previous time point (s_{t-1}) from the relevant row of $\underline{A}$ i.e. $A_{s_{t-1},:}$ (transition matrix)
- Given that state is $s_t = s$, we generate our observation $x_t | s_t$ from a Gaussian, using a mean and covariance indexed by s_t i.e μ_{s_t}, Σ_{s_t}.

Given s_t in both models, we generate the observations from a Gaussian with mean and covariance indexed by s_t (=s) i.e. μ_{s_t} and Σ_{s_t}

However for GMM, we are selecting clusters/states from a discrete distribution using same probability vector/mixing weight vector $\underline{\pi}$.

For continuous variable HMM, we are also using a discrete distribution, but that distribution is selected according to a transition matrix A. For S different states, there will be S possible different probability distributions we could choose, indexed by the previous state s_{t-1}. The previous state s_{t-1} therefore indexes which row of $\underline{A}$ we choose.

However - HMM in this case is homogeneous ⇒ conditional distributions governing latent variables/states share the same parameters (transition matrix A) and all of the emission distributions share the same parameters θ (even though indexing may select from these) (Q) ?

Probabilistic PCA (PPCA)

continuous state
- At time t, we have a vector s_t which is generated from an MVG with mean 0 and covariance Q
- Given s_t, we generate observation $x_t | s_t$ from an MVG with mean Ds_t and covariance V

Linear Gaussian Markov Model

Take PPCA and build a sequential model over latent state evolution
At time t, state s_t given state s_{t-1} is a MVG with mean $C s_{t-1}$ and same covariance Q
Latent state at previous time point informs current latent state s_t

- Given this latent state s_t; the observation x_t has exactly the same distribution as with probabilistic PCA, with mean $\mu = Ds_{t-1}$ and $\Sigma = V$
- In a similarity with the discete-state models (GMM and continuous variable HMM); the data generating distribution governing observations x_t given the latent state s_t is the same
- And the similarity (in difference) is ~~that~~ the way these data generating distributions $x_t|s_t$ are selected are different.
- For discete-state models, GMMs are ~~one~~ hose iid distribution with mixing weight π over latent Gaussians, whilst continuous HMMs have sequential distributions on how ~~Gaussian~~ latent Gaussians are selected.
- For continuous state models; PPCA has latent states iid from a continuous Gaussian distribution (not referred to as states), whilst with LGMM we have a Markov sequence generating these latent states.

Extensions

- Extended Kalman filter:
  - Non linear Kalman filters use nonlinear functions of the state; replacing a nonlinear function with a linear approximation
  - EKF approximates $h(s_t) \approx h(z) + \nabla h(z)(s_t - z)$

$$s_{t+1}|s_t \sim N(Ds_t, Q) \qquad x_t|s_t \sim N(h(s_t), V)$$

- continuous time:
  - Time between observations vary. Let Δ_t be time between observation x_t and x_{t+1} then model:

$$s_{t+1}|s_t \sim N(s_t, \Delta_t Q) \qquad x_t|s_t \sim N(Ds_t, V)$$

Adding controls

- In dynamic models, add a control to the latent state which we can specify using a value u_t whose values we can choose

$$s_{t+1}|s_t = N(Cs_t + Gu_t, Q) \qquad x_t|s_t \sim N(Ds_t, V)$$