# COMP 652: ASSIGNMENT 3

CARLOS G. OLIVER (260425853)

## 1. Q1: PCA

Dimensionality reduction using PCA on this dataset appears to have been quite effective. We can see that it is likely that the true dimensionality is much lower than the original number of features of 250. I performed PCA (implemented in scikit-learn) and report averaged values over 5 folds (yielding 80-20 train-test split) of the data. Looking at the mean variance explained by the top eigenvalues **Fig. 1** we see that the vast majority (over 90%) of variance in the data can be explained by the first few eigenvectors. Consequently, we achieve very low reconstruction errors **Fig. 2** when projecting on the largest eigenvectors. We achieve a mean squared reconstruction error under 0.05 by projecting on the top 5 eigenvalues. We can conclude that it is safe to perform further analysis on a reduced representation of the original data.

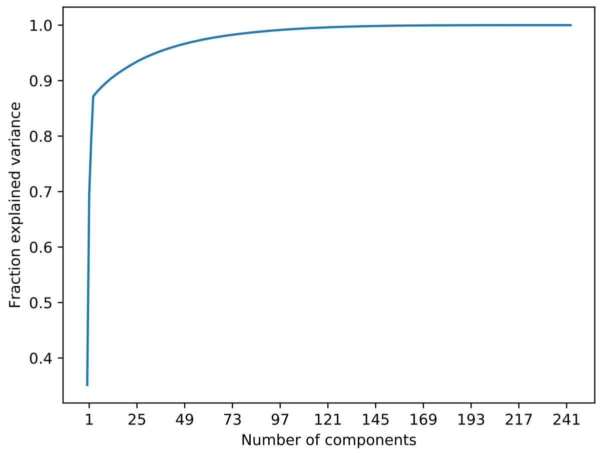
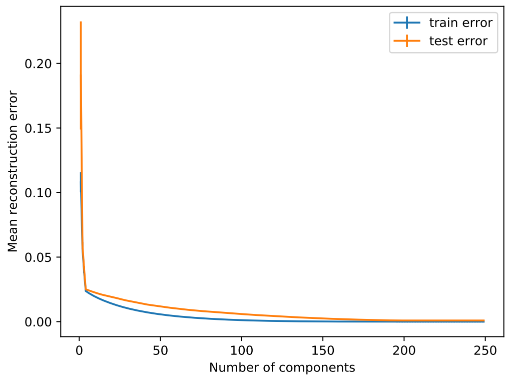
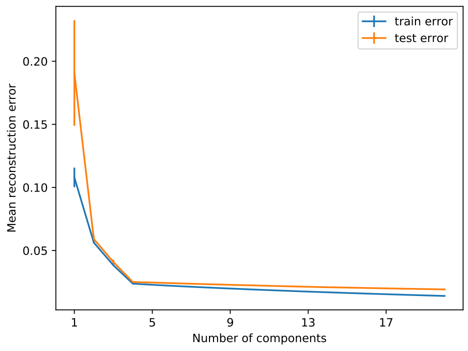


FIGURE 1. Fraction of variance explained from top $n$ eigenvectors.

*Date*: April 14, 2017.



(A)



(B)

Figure 2. Reconstruction error when projecting on top $n$ eigenvectors. Figure on the right is a truncated version of the full plot on the left.

## 2. Q2: Spectral methods for weighted automata

2.1. **(a).** We use a property of Hankel matrices which is: if $\text{rank}(H_f) = n \Rightarrow$ there exists a weighted finite automaton $\mathcal{A}$ with $n$ states such that $g = g_{\mathcal{A}}$ for some function $g : \Sigma^* \to \mathbb{R}$ where $\Sigma^*$ is the set of all strings that can be generated from alphabet $\Sigma$. In this example, the function $g$ counts the number of 1s in a string generated from $\Sigma = \{0, 1\}$ the Hankel matrix $H_g$ would take the form:

$$H_g = \begin{array}{c|ccccc} & \lambda & 0 & 1 & 11 & \dots \\ \hline \lambda & 0 & 0 & 1 & 2 & \dots \\ 0 & 0 & 0 & 1 & 2 & \\ 1 & 1 & 1 & 2 & 3 & \\ 11 & 2 & 2 & 3 & 4 & \\ 111 & 3 & 3 & 4 & 5 & \\ 110 & 2 & 2 & 3 & 4 & \\ \vdots & \vdots & & & & \ddots \end{array}$$

The above matrix is arranged so all possible strings composed only of 1s, $Q = \{1\}^N \quad \forall N \in \mathbb{Z}^+$ precede all other strings $(\Sigma^* - Q)$ with the exception of $\{0\}^1$ and $\lambda$. If we order prefixes and suffixes in S in increasing $N$ we can easily see that any row $m$ in this sub block $H_g^1$ of $H_g$ is a linear combination in the form $H_g^1(m, \cdot) = 2H_g^1(m, \cdot) - H_g^1(m-1, \cdot)$. The remaining entries in $H_g$ are formed by prefixes and suffixes containing 0s which do not contribute to the evaluation of $g$ and therefore can also be obtained from rows or columns in $H^1$. Therefore the rank of $H_g$ is 2 where only the $\lambda$ or $\{0\}^1$ and the $\{1\}^1$ contribute to the row rank. The same can be shown for the column rank.

2.2. **(b).** If $f$ is a probability distribution over $\Sigma^*$ then we have $\sum_{s\in\Sigma^*} f(s) = 1$ gives a probability for every string in $\Sigma^*$. Then $f_{sub}(w)$ is the probability that the string $w$ occurs in any word generated by $\Sigma^*$ since $u$ and $v$ represent all possible prefixes and suffixes to the word $w$.

2.3. **(c).** Let $\Sigma_{w\in\Sigma^*} A_w \equiv S$.

$$\begin{aligned} S = (I - A_0 - A_1)^{-1} = (I - A_0 - A_1)S &= I \\ (I - A_0 - A_1)SS^{-1} &= S^{-1} \\ (I - A_0 - A_1)^{-1} &= S \end{aligned} \tag{1}$$

Now we use this identity to compute the sum of the function $f$ over all words as $\Sigma_{w\in\Sigma^*} f(w)$. We can express the function over a string as

$$f(w) = \alpha_0^T A w \alpha_\infty \tag{2}$$

So for all the words we have

$$\begin{aligned} \sum_{w\in\Sigma^*} f(w) &= \sum_{w\in\Sigma^*} \alpha_0^T A_w \alpha_\infty \\ &= \alpha_0^T \Big( \sum_{w\in\Sigma^*} A_w \Big) \alpha_\infty \\ &= \alpha_0^T (I - A_0 - A_1)^{-1} \alpha_\infty \end{aligned} \tag{3}$$

We can use this property to compute $f_{subsrting}(w)$ using an automaton by recognizing that the contribution of transition matrices from $u$ and $v$ to the sum can be reduced to the same identity as above as they sum over all of $\Sigma^*$.

$$\begin{aligned} f_{substring}(w) &= \sum_{u\in\Sigma^*, v\in\Sigma^*} f(uwv) \\ &= \sum_{u\in\Sigma^*, v\in\Sigma^*} \alpha_0^T A_u A_w A_v \alpha_\infty \\ &= \alpha_0^T (I - A_0 - A_1)^{-1} A_w (I - A_0 - A_1)^{-1} \alpha_\infty \end{aligned} \tag{4}$$

This yields a new set of initial and final vectors:

$$\begin{aligned} \hat{\alpha_0}^T &= \alpha_0^T (I - A_0 - A_1)^{-1} \\ \hat{\alpha}_\infty &= (I - A_0 - A_1)^{-1} \alpha_\infty \end{aligned} \tag{5}$$

2.4. **(d).** Given a sample $s_1, s_2, s_3, ..., s_n \in S$ of words from the distribution on $f$ we seek to learn $f_{substring}$ which takes all possible substrings over the strings in $\Sigma^*$. We can use the equation derived in part (c) to learn $f_{substring}$ from samples of $f$ to yield an automaton $\hat{A}$.

- Compute empirical Hankel matrix $\hat{H}_{substring}$ by counting the number of occurrences of $w$ in $S$ and insuring sampling is done in an independent fashion.
- Estimate prefix and suffix sub-blocks $\tilde{h}_{u,\lambda}, \tilde{h}_{\lambda,v}$ and transition matrix $H_\sigma$ from finite sub-blocks of $\hat{H}$
- Perform singular value decomposition of $\hat{H}$ and use factorization to solve for $\hat{\alpha}_0, \hat{\alpha}_\infty, \hat{A}_0, \hat{A}_\sigma$

To retrieve estimates of $f$ we can simply use the relation derived in (c) and solve for the parameters of $A$ from $\hat{A}$.

$$\begin{aligned} \alpha_0^T &= \hat{\alpha}_0^T(I - A_0 - A_1) \\ \alpha_\infty &= (I - A_0 - A_1)\hat{\alpha}_\infty \end{aligned} \tag{6}$$

We can again use spectral methods to recover an automaton for $f$. So we have a two way relationship between $f$ and $f_{substring}$ (proven more formally in [2]) which has the benefit of adapting automata for more specific modelling and prediction tasks. However, obtaining an estimate of $f$ from $f_{substring}$ is more computationally expensive than learning $f$ directly.

## 3. Q3: Method of moments and multiview model

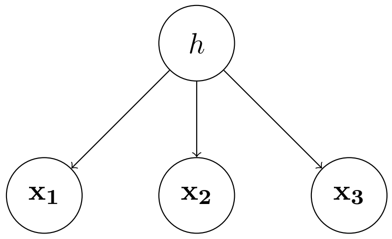


Figure 3. Graphical representation of conditional independence relation between random variables $\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}$ and $h$.

Since the $\mathbf{x_t}$ are conditionally independent given $h$ we can decompose the conditional expectation of the cross moments.

$$\begin{aligned} \mathbb{E}[\mathbf{x_1} \otimes \mathbf{x_2} | h = j] &= \mathbb{E}[\mathbf{x_1}|h = j] \otimes \mathbb{E}[\mathbf{x_2}|h = j] && \text{by conditional independence} \\ &= \mu_{t=1,j} \otimes \mu_{t=2,j} \in \mathbb{R}^{d_1 \times d_2} && \text{by question statement} \\ &= \sum_{i=1}^{k} w_i \mu_{1,i} \otimes \mu_{2,i} \end{aligned} \tag{7}$$

$$\mathbb{E}[\mathbf{x_1} \otimes \mathbf{x_2} \otimes \mathbf{x_3} | h = j] = \sum_{i=1}^{k} w_i \mu_{1,i} \otimes \mu_{2,i} \otimes \mu_{3,i} \in \mathbb{R}^{d_1 \times d_2 \times d_3} \tag{8}$$

Because the tensor products of the $\mathbf{x_t}$ give rise to non-symmetric tensors, we cannot apply the standard tensor method of moments as eigenvalue decomposition requires symmetric matrices [1]. Other techniques can be employed but are beyond the scope of this question.

## 4. Q4: Coupled Hidden Markov Models

4.1. **(a).** Parametrization of CHMM:

- Initial probabilities:

$$\begin{aligned} P(s_0 = s) \quad & \forall s \in S \\ P(u_0 = u) \quad & \forall u \in U \end{aligned} \tag{9}$$

- Transition Probabilities

$$\begin{aligned} P(s_{t+1} = s' | s_{t-1} = s, u_{t-1} = u) \quad & \forall s', s, u \in S \cup U \\ P(u_{t+1} = u' | u_{t-1} = u, s_{t-1} = s) \quad & \forall u', u, s \in S \cup U \end{aligned} \tag{10}$$

- Emission Probabilities

$$\begin{aligned} P(y_t = y | s_t = s) \quad & \forall s \in S, y \in Y \\ P(z_t = y | u_t = u) \quad & \forall u \in U, z \in Z \end{aligned} \tag{11}$$

4.2. **(b).** We want to compute $P((y_0, z_0), (y_1, z_1), ...(y_T, z_T))$. We can obtain this probability by marginalizing over the states in the joint over the full model.

$$\begin{aligned} & P((y_0, z_0), \dots, (y_T, z_T)) \\ &= \sum_{s_0,\dots,s_T,u_0,\dots u_T} P((y_0, z_0), \dots (y_T, z_T), s_0, \dots, s_T, u_0, \dots, u_T) \\ &= \sum_{s_0,\dots,s_T,u_0,\dots u_T} P(s_0)P(u_0) \prod_{t=1}^{T} P(s_t | s_{t-1}, u_{t-1}) \prod_{t=1}^{T} P(u_t | s_{t-1}, u_{t-1}) \prod_{t=0}^{T} P(y_t | s_T) \prod_{t=0}^{T} P(z_t | u_t) \\ &= \sum_{s_T} P(y_T | s_T) \sum_{u_T} P(z_T | u_T) \sum_{s_0,\dots,s_{T-1},u_0,\dots u_{T-1}} P(s_T | s_{T-1}) P(u_T | u_{t-1}) P(s_0) P(u_0) \\ & \quad \prod_{t=1}^{T-1} P(s_t | s_{t-1}, u_{t-1}) \prod_{t=1}^{T-1} P(u_t | s_{t-1}, u_{t-1}) \end{aligned} \tag{12}$$

We obtain the second step simply by factoring the graph into its independent components according to the graphical model. By rearranging terms we see that the full joint distribution can be written as a product of the distribution at the final time $T$ and the joint over all times $T-1$ which suggests a dynamic programming algorithm which we can write in the following form:

(13)
$$P((y_0,z_0),...(y_T,z_T)) = \sum_{s_T,u_T} P((y_0,z_0),(y_1,z_1),...(y_T,z_T),u_T,s_T)$$
$$= \sum_{s_T,u_T} P(z_T|u_T)P(y_T|s_T) \sum_{S_{T-1}} P(s_T|s_{T-1})P(u_T|u_{T-1})P((y_0,z_0),...(y_T,z_T),s_{T-1},u$$

We define a matrix $A_t(s,u)$ of dimensions $|S|$ and $|U|$ that will store the probability of the observation sequence until time $t$ and of state $s_t$ and $u_t$. In the uncoupled version of HMM we have a vector instead of a matrix. The entries of this matrix have the following form:

$$A_t(s,u) = P((y_0,z_0),...(y_t,z_t),s_t = s,u_t = u) \tag{14}$$

At $t=0$ we have our base case which is a function of the initial probabilities $s_0$ and $u_0$:

(15)
$$A_0(s,u) = P(y_0,z_0,u_0 = u,s_0 = s) = P(s_0 = s)p(u_0 = u)P(y_0|s_0 = s)P(z_0|s_0 = s) \qquad \forall s,u \in S,U$$

For all subsequent time steps $t=1,...t=T$ we can use the recursive expression derived in (12):

$$A_t(s,u) = P(y_t|s_t)P(z_t|u_t) \sum_{s_{t-1},u_{t-1}} P(s_t|s_{t-1},u_{t-1})P(u_t|s_{t-1},u_{t-1})A_{t-1}(s,t) \tag{16}$$

Rewriting this in more compact form:

$$P(y_0,z_0),...(y_t,z_t)) = \sum_{s_T,u_T} A_T(s,u) \tag{17}$$

4.3. **(c).** This task can be accomplished with a modified version of the Viterbi algorithm which is a dynamic programming technique.

We wish to compute the set of hidden states with the maximum probability given the observed sequence:

$$\underset{s_0u_0,s_1u_1,...,s_Tu_T}{\text{argmax}} P(s_0u_0,s_1u_1,...,s_Tu_T|(y_0,z_0),...(y_t,z_t)) \tag{18}$$

Using simple rules of probability we can rewrite the conditional probability:

$$\underset{s_0u_0,s_1u_1,...,s_Tu_T}{\text{argmax}} \frac{P(s_0u_0,s_1u_1,...,s_Tu_T,(y_0,z_0),...(y_t,z_t))}{P((y_0,z_0),...(y_t,z_t))} \tag{19}$$

We note that the denominator is just the probability of the observation sequence and does not depend on the hidden states so we can omit it from further calculations.

Now we can adapt the Viterbi algorithm (as described on Wikipedia) as follows:

To simplify notation we let:

$$\begin{gathered} P(u_t = u' | u_{t-1} = u) \equiv \mathbb{U}_{u,u'} \\ P(s_t = s' | s_{t-1} = s) \equiv \mathbb{S}_{s,s'} \end{gathered} \tag{20}$$

Now we consider the observed sequence $y_0 z_0, y_1 z_1, ..., y_t, z_t$. Let the probability of the *most likely* state sequence $s_0 u_0, s_1 u_1, ... s_i u_j$ ending at states $i \in S$ and $j \in U$ be stored in the 3D matrix $V$ where

$$V_{0,i,j} = P(y_0|i)P(z_0|j)P(s_0)P(u_0) \tag{21}$$

Is the sequence at $t = 0$ so we just have the emission probabilities multiplied by the initial state probabilities. For subsequent time steps we have the same as above, instead now we include transition probability factors instead of initial state probabilities, as well as the probability of the sequence of states which led to time $t$ which is just $V_{t-1,s,u}$:

$$V_{t,i,j} = \underset{s \in S, u \in U}{\operatorname{argmax}} P(y_t|i)P(z_t|j)\mathbb{S}_{s,i}\mathbb{U}_{u,j}V_{t-1,s,u} \tag{22}$$

This will result in a dynamic programming table storing the probability of all possible paths through the CHMM. Instead of a 2D table as in the standard Viterbi algorithm we will have a 3D table of dimensions $|T| \times |S| \times |U|$. To retrieve the most likely sequence of hidden states, we store back pointers at every $V_t$ that lead to the state that maximized $V_{t-1}$.

4.4. **(d).** If chains are coupled every $k$ time steps, then for time steps that are multiples of $k$ we have the same form as the CHMM and for those that are not multiples of $k$ we have two independent chains. This would allow us to decompose the probabilities for time steps in parts (b) and (c) into those that are multiples of $k$ i.e. $\{t \in T_k \quad \text{if} \quad t \mod k = 0\}$ and thus remain as before, and those that are not multiples of $k$, i.e. $t \notin \pi_k$ and there is no dependence between chains. This would result in a new parametrization of the transition probabilities which can be easily plugged back into both algorithms.

For all $t \in T_k$:

$$\begin{gathered} P(s_{t+1} = s' | s_{t-1} = s, u_{t-1} = u) \quad \forall s', s, u \in S \cup U \\ P(u_{t+1} = u' | u_{t-1} = u, s_{t-1} = s) \quad \forall u', u, s \in S \cup U \end{gathered} \tag{23}$$

For all $t \notin T_k$:

$$\begin{aligned} &P(s_{t+1} = s' | s_{t-1} = s) \quad \forall s', s \in S \\ &P(u_{t+1} = u' | u_{t-1} = u) \quad \forall u', u \in U \end{aligned} \tag{24}$$

4.5. **(e).** We want to find the $k$ and the set of model parameters $\mathcal{A}$ that maximizes the probability of the observed sequences, given the set of possible states. i.e.:

$$\underset{k,\mathcal{A}}{\operatorname{argmax}} P((y_0, z_0), ...(y_t, z_t) | k, \mathcal{A}) \tag{25}$$

To do this we can use the Baum-Welch algorithm for Expectation Maximization over all possible values of $k$ and pick the one which maximizes the likelihood of the data. At each value of $k$ we pick the parameters that maximize likelihood given $k$ and at the end we pick the $k$ that produces the most likely model. It is important to note that because this is an EM-based algorithm, the parametrization is not guaranteed to converge to a global optimum.

```
input  : $(y_0, z_0), ...(y_t, z_t), S, U, \mathcal{A}_0$
output: $\operatorname{argmax}_{k,\mathcal{A}} P((y_0, z_0), ...(y_t, z_t) | k, \mathcal{A})$
$bestProb \leftarrow 0$
$bestK \leftarrow 0$
for $k$ in $K$ do
    $\mathcal{A}' \leftarrow$ BaumWelch$((y_0, z_0), ...(y_t, z_t), S, U, \mathcal{A}_0, k)$
    $P((y_0, z_0), ...(y_t, z_t) | \mathcal{A}, k) \leftarrow$ ForwardAlg$((y_0, z_0), ...(y_t, z_t), S, U, \mathcal{A}', k)$
    $P(k, \mathcal{A}') = P((y_0, z_0), ...(y_t, z_t) | \mathcal{A}, k) P(A, k)$
    if $P(k, \mathcal{A}') > bestProb$ then
        $bestK \leftarrow k$
    end
end
return $bestK$
```

## 5. Notes

Collaboration with Navin Mordani and Jeremy Georges-Filtreau.

## References

[1] Animashree Anandkumar, Rong Ge, Daniel J Hsu, Sham M Kakade, and Matus Telgarsky. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15(1):2773–2832, 2014.

[2] Borja Balle, Xavier Carreras, Franco M Luque, and Ariadna Quattoni. Spectral learning of weighted automata. *Machine Learning*, 96(1-2):33–63, 2014.