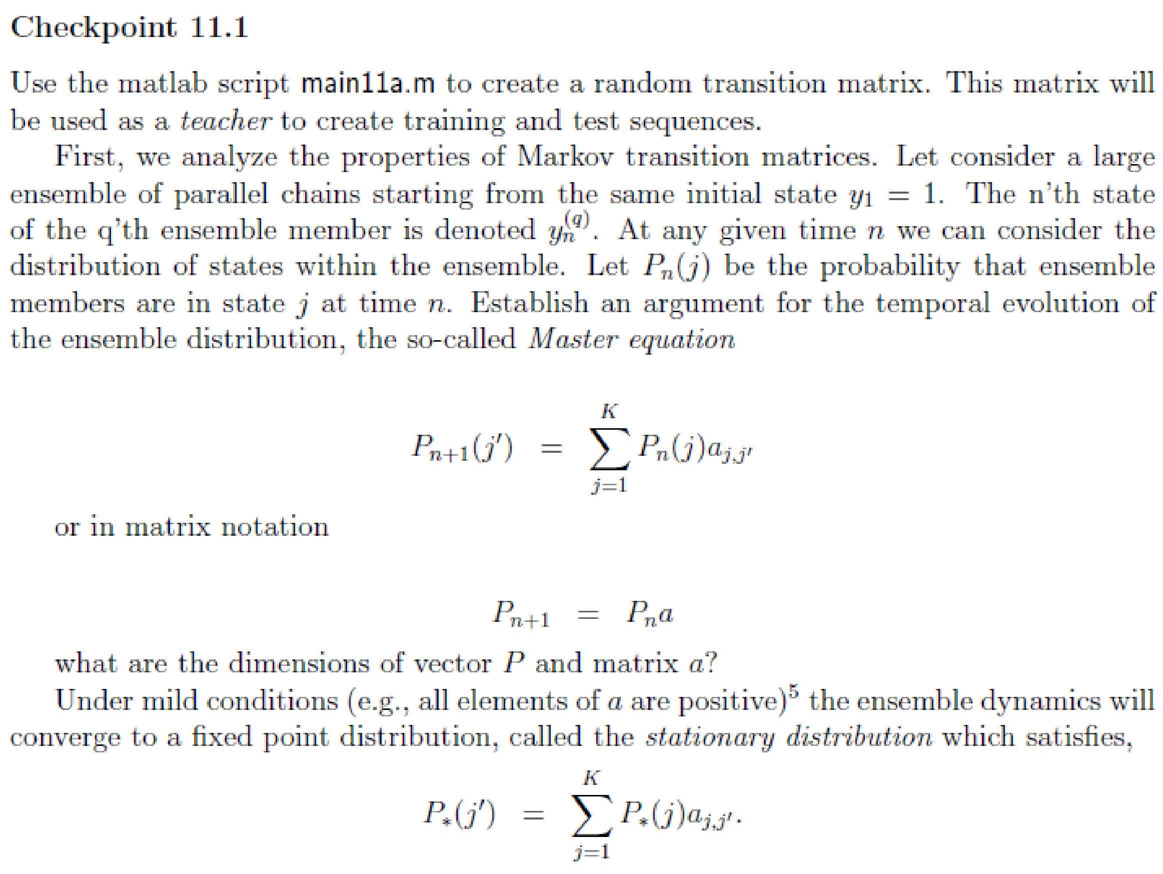


Checkpoint 11.1

Use the matlab script main11a.m to create a random transition matrix. This matrix will be used as a *teacher* to create training and test sequences.

First, we analyze the properties of Markov transition matrices. Let consider a large ensemble of parallel chains starting from the same initial state $y_1 = 1$. The n'th state of the q'th ensemble member is denoted $y_n^{(q)}$. At any given time n we can consider the distribution of states within the ensemble. Let $P_n(j)$ be the probability that ensemble members are in state j at time n. Establish an argument for the temporal evolution of the ensemble distribution, the so-called *Master equation*

$$P_{n+1}(j') = \sum_{j=1}^{K} P_n(j) a_{j,j'}$$

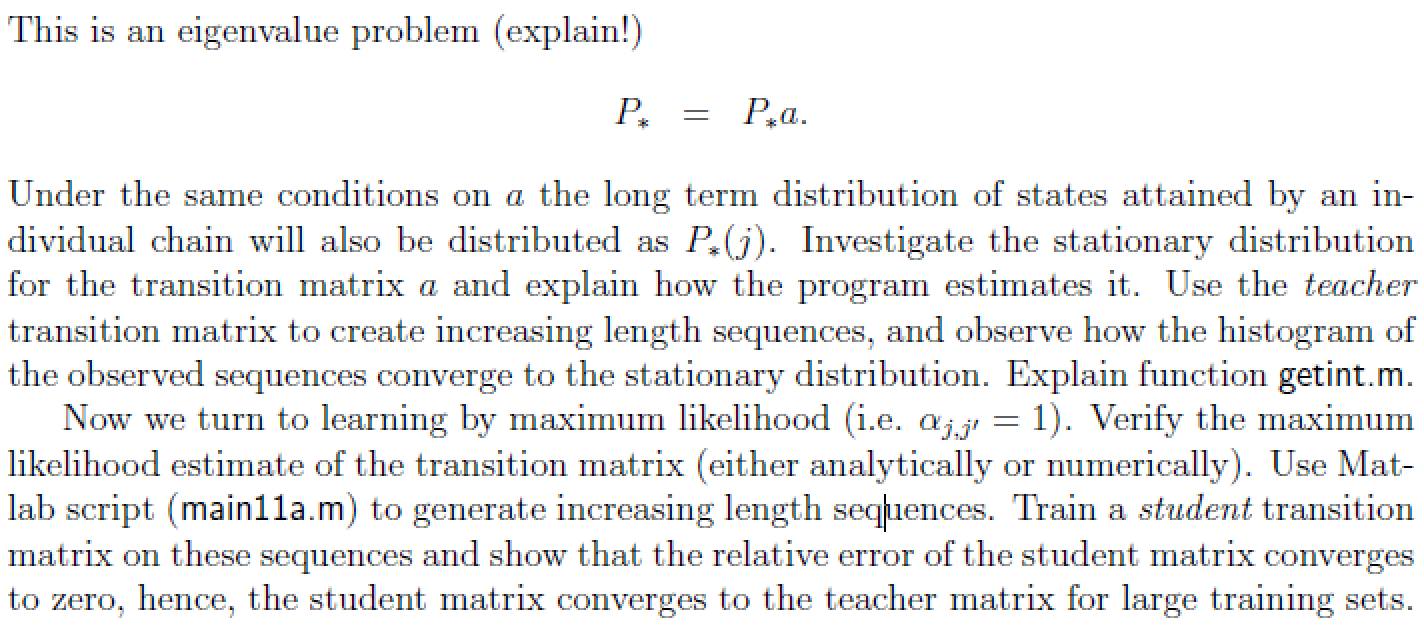
or in matrix notation

$$P_{n+1} = P_n a$$

what are the dimensions of vector P and matrix a?

Under mild conditions (e.g., all elements of a are positive)[5] the ensemble dynamics will converge to a fixed point distribution, called the *stationary distribution* which satisfies,

$$P_*(j') = \sum_{j=1}^{K} P_*(j) a_{j,j'}.$$



This is an eigenvalue problem (explain!)

$$P_* = P_* a.$$

Under the same conditions on a the long term distribution of states attained by an individual chain will also be distributed as $P_*(j)$. Investigate the stationary distribution for the transition matrix a and explain how the program estimates it. Use the *teacher* transition matrix to create increasing length sequences, and observe how the histogram of the observed sequences converge to the stationary distribution. Explain function getint.m.

Now we turn to learning by maximum likelihood (i.e. $\alpha_{j,j'} = 1$). Verify the maximum likelihood estimate of the transition matrix (either analytically or numerically). Use Matlab script (main11a.m) to generate increasing length sequences. Train a *student* transition matrix on these sequences and show that the relative error of the student matrix converges to zero, hence, the student matrix converges to the teacher matrix for large training sets.

The dimensions of the P and a matrixes are shown below:

$$\begin{bmatrix} P_{n-1}(0) \\ P_{n-1}(1) \\ \dots \\ P_{n-1}(K) \end{bmatrix} = \begin{bmatrix} P_n(0) & P_n(1) & \dots & P_n(K) \end{bmatrix} \begin{bmatrix} a_{0\,0'} & a_{0\,1'} & .. & a_{0\,K'} \\ a_{10'} & a_{11'} & .. & a_{1\,K'} \\ .. & .. & .. & .. \\ a_{K\,0'} & a_{K\,1'} & .. & a_{K\,K'} \end{bmatrix}$$

Thus, the dimensions of P are Kx1 or 1xK and the dimensions of a are K x K.

The equation $P_* a = P_*$ look quite similar to the equation of eigenvectors and eigenvalues $Mv = \lambda v$ with eigenvalues equal to 1. In fact, we can present it the same way by transposing both terms, which give us $a^T P_*^T = P_*^T$ which means that the transition matrix transpose eigenvector is equal to the transpose of the stationary matrix with eigenvalue equal to 1. When we obtain more than one eigenvector with eigenvalue equal to 1 that means that there is an associated stationary distribution and that the markov chain is reducible.

This is calculated by the program in an iterative fasion, using the power iteration method. First, we need to assume that $A^k q = \lambda^k q$ which is, in general, true for all the cases. In this contest, if we take $v^{(0)}$ as an approximation of an eigenvector of A, with $\|v^{(0)}\| = 1$, we can write this vector as a linear combination of the eigenvectors of A for certain weights c:

$$v^{(0)} = c_1 q_1 + \dots + c_n q_n,$$

and we will assume for now that $c_1 \neq 0$.

Now

$$Av^{(0)} = c_1 \lambda_1 q_1 + c_2 \lambda_2 q_2 + \dots + c_n \lambda_n q_n$$

and so

$$\begin{aligned} A^k v^{(0)} &= c_1 \lambda_1^k q_1 + c_2 \lambda_2^k q_2 + \dots + c_n \lambda_n^k q_n \\ &= \lambda_1^k \left(c_1 q_1 + c_2 \left(\frac{\lambda_2}{\lambda_1}\right)^k q_2 + \dots + c_n \left(\frac{\lambda_n}{\lambda_1}\right)^k q_n \right). \end{aligned}$$

Since the eigenvalues are assumed to be real, distinct, and ordered by decreasing magnitude, it follows that for all $i = 2, \dots, n$,

$$\lim_{k \to \infty} \left(\frac{\lambda_i}{\lambda_1}\right)^k = 0.$$

So, as k increases, $A^k v^{(0)}$ approaches $c_1 \lambda_1^k q_1$, and thus for large values of k,

$$q_1 \approx \frac{A^k v^{(0)}}{\|A^k v^{(0)}\|}.$$

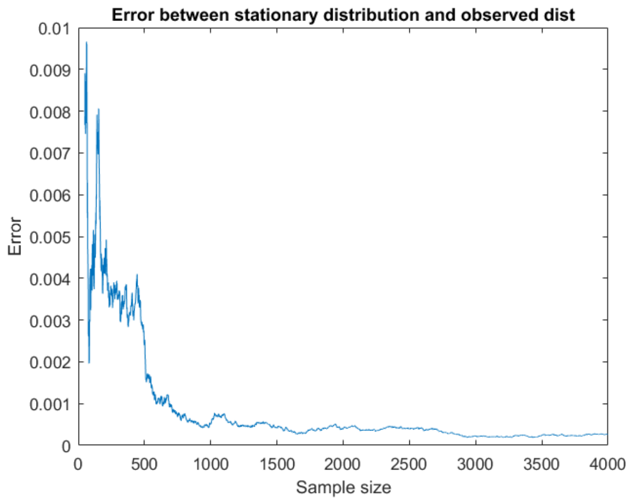
Thus, we can iterate to find the value of q1 (p in the code) by iterating with the normalized transition matrix and an initialized random vector p with norm equal to 1.

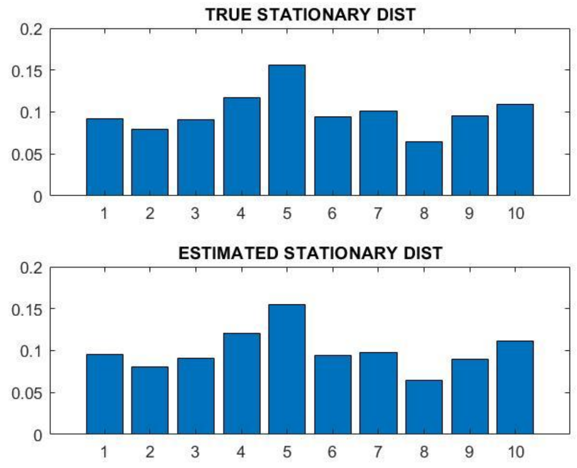
Once the value of the eigenvector p is calculated the next step is to calculate the teacher transition matrix. This is calculated by taking the cumulative sum of each of the raws in a_jj' normalized.

Afterwards we are going to use this cumulative matrix to estimate a large sequence of states. In order to do so, we are going to use the function getint, which algorithm can be explained as follows:

Firstly, we take a random number between 0 and 1.Then, given a row of the ac matrix (the cumulative probabilities to other states), we are going to select the estate number of the next value after the random number in the row. This will give us a path to continue with our sequence, in a fashion given by the probabilities of the teacher matrix.

In this iterative process we can see how the value of the observed sequences converges to the stationary distribution. In order to visualize this we are going to plot the error between the stationary distribution out of the eigenvector property and the distribution that we are creating.



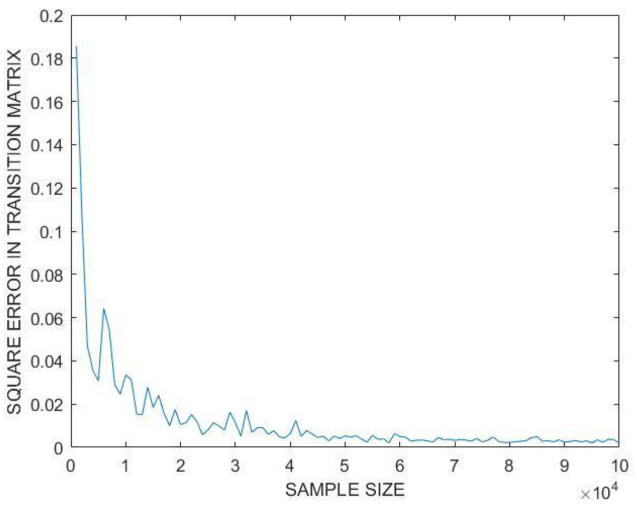


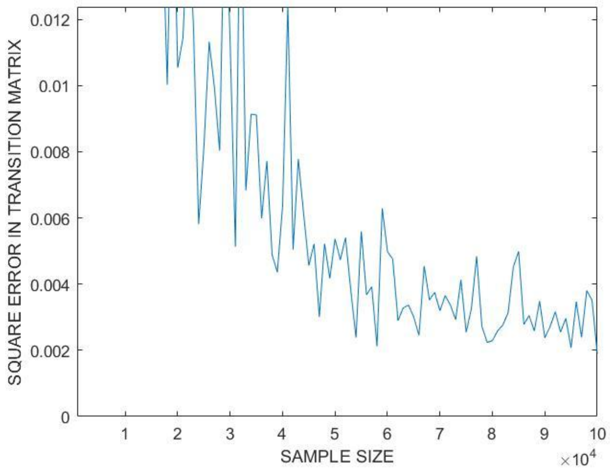
We see that even when the sample size is really small the error is still really small. However, we can consider that it converges at around sample size of 3000. We can also see the final result in a histogram form.

After obtaining the stationary distribution by the means of the properties of the markov transition models, we are now also able to calculate the transition matrix given a sequence and using maximum likelihood. i.e. using $\hat{a}_{j,j'} = \frac{n_{j,j'}}{\sum_{j''} n_{j,j''}}$.

This can be achieved numerically by taking the whole sequence and adding K times the next number of the sequence minus 1. This will give us a sequence of values between 1 and K x K, from which a histogram can be created and then casted into the predicteed a matrix. Finally, we will normalize this matrix so that the sum of all the rows is equal to 0.

Following this method, we can estimate the sequence when the number of samples is big enough. We can actually see the change in the error with the number of elements in our sequence in the following graph





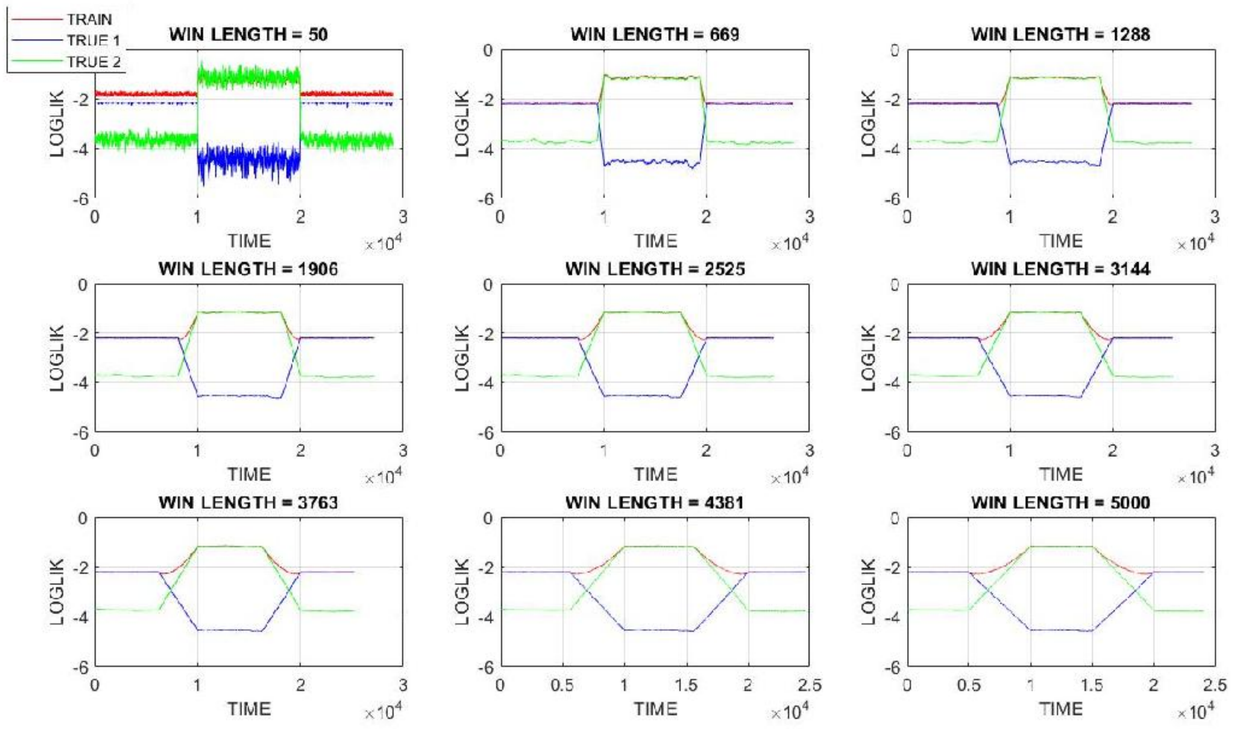
As we can see, the square error decreases with the number of samples, reaching the convergence point at around 5000, few samples more than to the previous case. However, this measurements cannot be really compared, as this last one is the error between a matrix and the previous case was the error of the stationary distribution.

Checkpoint 11.2

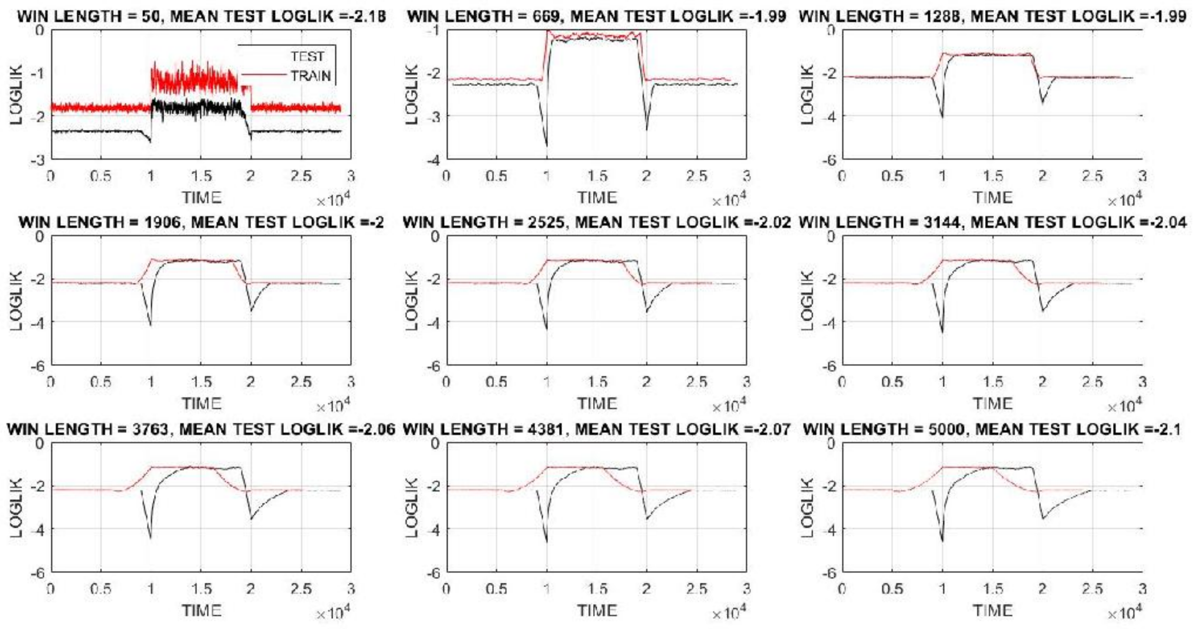
We design a simple ”switching” non-stationarity with two Markov models taking turns. Model 1 generates the states in the intervals $[1, N_1]$ and $[N_1 + N_2 + 1, 2N_1 + N_2]$, while Model 2 generates the states in the interval $[N_1 + 1, N_1 + N_2]$.

We estimate Markov models using the MAP estimator for overlapping windows using a set of different window sizes. Compare the likelihood of the transition matrix computed on the training data and the likelihoods of the two ”true” models.

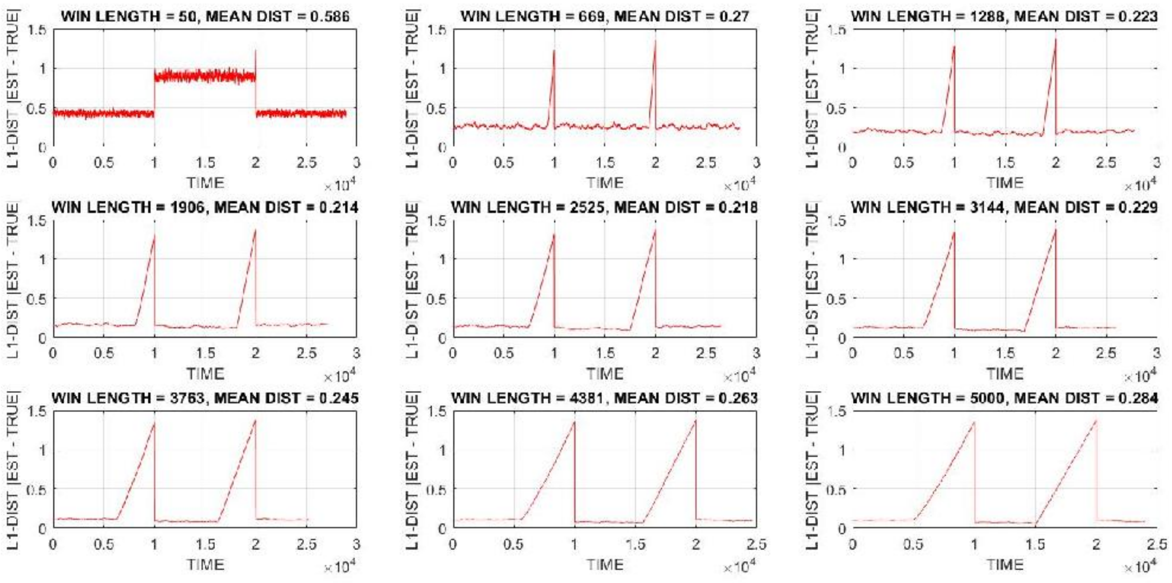
We evaluate the estimated Markov models by their L_1 distances to the true model and using the log-likelihood on test data. Discuss the impact of the window size and the prior α on the estimated Markov model and these generalization performance estimates. Which window size would you recommend?



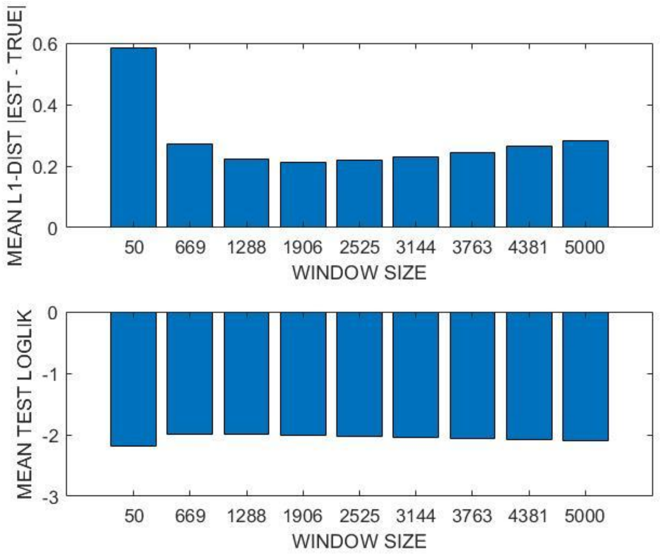
In this case we are plotting the resultant log-likelihood results for different window sizes and train and true values. First of all, it is important to explain how are these true values calculated. These values are estimated by estimating the loglikelihood between the current window and the transformation matrix one and two, respectively. In the results we see that when we increment the size of the window, the transitions between high/low likelihood periods are more smooth. This is caused because in these stages part of both models are taken into count in order to calculate the log-likelihood.



Now if we compare the log likelihood of the train and test sets we see how some patterns from the previous plot are also relevant in this case. For example, in both cases when we increment the size of the window we obtain a less noisy likelihood function. In this plot, it seems important to realize how the test log-likelihood suffers with each change of the sequence model. As we are predicting just based on the previous window, the results are bad after a change. The results are specially bad in the case when the window size is big, as we are predicting a larger subset.

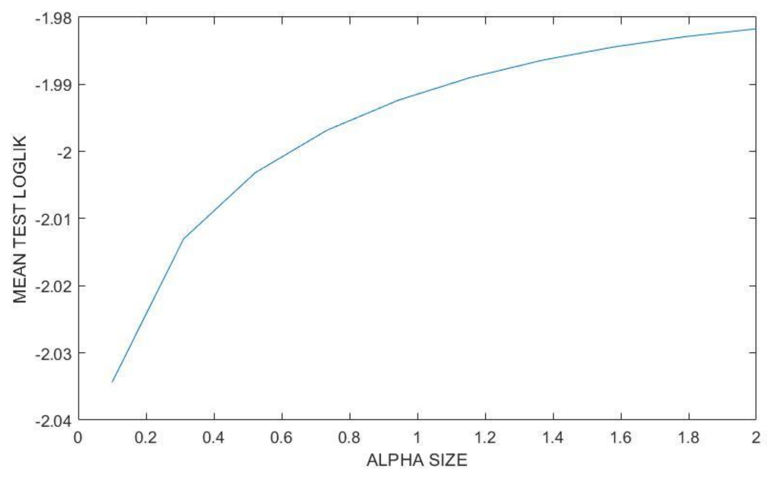


If we take a look at the L1 distance instead of the log likelihood we keep seing the same paterns. If we introduce a bigger window, the error grows as the window lays in the intersection of two sequences.



We can compare the different errors for the test set, an assess which is the best window size based on this results. Thus, we have selected a window size of 1288 as the most optimal, as this is the one that minimizes the L1 distance and gives more appropiate results in terms of the log-likeihood.

In order to asses the importance of alpha we will just sample different values of alpha (between 0.1 and 2) and we calculate the errors as we did with the windows size. This gave us some results



As we can see, the plot is nearly exponential, even though it is also important to take into count the scale, as it doesn't make a big difference in the results.