Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ciencias de la Computación

Aprendizaje Automático Bayesiano

Student State Estimation

Student: Mario W. Garrido F.
Professor: Pablo Guerrero P.

Introduction

The chief goal of the present model is to estimate the probability that a given student will select a given answer to a learning experience. To simplify modelling it is assumed that said experiences have a finite number of possible answers (e.g: Multiple choice questions), this allows an assumption to be established: The student's answers, for a given time t, follow a Multinomial(1, x_t) distribution. Therefore, the problem of estimating the student's internal state is reduced to the problem of estimating the probability vector x_t.

The dataset D starts empty and is subsequently filled with the following information:

- *Experiences {u}*: At time t an experience u_t is supplied to the student. The experience has a fixed number of possible answers.
- *Observations {y}*: Each y_t corresponds to the student's answer to experience u_t. Each sample is an n-dimensional vector, where n is the number of possible answers to the experience.
- *Estimations {x}*: A new estimation x_t is computed using the previous estimation x_{t-1} and the new observation y_t.

In order to perform the estimation a Bayesian Filter will be used. In particular, Sequential Bayesian Estimation via a Particle Filter. It is necessary to define the following concepts:

- *Process Model*: Given an estimation x_t, a prediction regarding the distribution of x_{t+1}. It is denoted $P(x_t|x_{t-1}, u_t)$.
- *Observation Model*: Given an estimation x_t, a prediction of the probability of the sample y_t. It is denoted $P(y_t|x_t)$.
- *Belief*: Current estimate of the distribution of x.

Model

- $P(x_t|x_{t-1}, u_t, w) = Dirichlet(\alpha)$
- $P(y_t|x_t) = Multinomial(1, x_t)$
- Initial Belief = $Dirichlet(j)$, where j is an n-dimensional vector with all entries equal to 1.

Algorithm

The present algorithm corresponds to Sequential Importance Resampling, which approximates the posterior density by a weighted empirical probability density function $P(x_t|y_{1:t}) \approx \sum_{i=1}^{N} w_t^{(i)} \delta(x_t - x_t^{(i)}), \forall i \in \mathbb{N}, i \leq N$ and subject to the restrictions $w_i \geq 0, \sum_{i=1}^{N} w_i = 1$, where x_t^i denotes the value of the i-th particle: possible state of the system at time t, w_t^i signifies weight of the i-th particle at time t: scalar proportional to expected probability of the system being in the state for a small neighbourhood of x_t^i and N is the total number of particles.

Initialization: N particles are sampled from the Initial Belief. Since j is the vector of ones this constitutes an uninformative prior: any output vector is equally likely. All sampled particles are given equal weight: $\frac{1}{N}$. Since the samples are taken from a Dirichlet distribution the sum of the elements of each sample will always be 1. The transition from $t-1$ to t can then be computed as described below.

1. For each i compute x_t^i by random sampling from the Process Model: $P(x_t|x_{t-1}^i)$.
2. For each i compute weight $w_t^i = P(y_t|x_t^i)$ using the observation y_t.
3. Normalize weights, so that approximation of posterior integrates to one.
4. For each i resample x_t^i from approximate posterior (the new Belief) and reset all weights to $\frac{1}{N}$. If sampling is truly random the result is that each particle is, on average, copied $n_i \approx w_t^i N$ times, which allows the assumption that the statistics of the posterior probability are, on average, maintained.

The expected value of the approximate posterior of the last step will be used, ultimately, as the proposed value for x_t.

Implementation

This section is intended to complement the inspection of the provided code, and is not intended for a standalone description of the implementation.

The implementation is built on top of the *PyBayes* library, which provides basic support for probability density function construction and some simple filters. The following probability distributions are included in the code:

- Multinomial
- Dirichlet
- ConditionalMultinomial: the conditional argument is the probability vector x_t.
- ConditionalDirichlet: the conditional argument is the vector α.

All of them implement the functions:

- *mean*: Returns the mean of the distribution.
- *sample*: Returns a sample from the distribution.
- *variance*: Returns the variance of the distribution.
- *eval*: Returns the value of the probability distribution function or probability mass function evaluated at the point, accordingly.
- *eval_log*: Returns the logarithm of eval.

The filter, *conditionalFilter*, takes as parameters: number of particles, Initial Belief, Process Model and the Observation Model. There is no mention of a *learning rate* because two strategies were adopted for the evolution of the Process Model, with respect to its conditional parameter.

The first approach considers computing the parameter $\alpha^i = a(w_{t-1}^i j + (j - w_{t-1}^i j)x_{t-1}^i)$. This parameter is used on the sampling step, to sample x_t^i, as the parameter of $Dirichlet(\alpha)$, the process model. Both approaches use a parameter a. This parameter is used to amplify the result so that the values in the resulting vector are meaningful.

The second approach considers computing the parameter $\alpha^i = a(wx_{t-1}^i + (1-w)ax')$, where w is a fixed parameter (which could be considered a learning rate) and x' corresponds to the current expected value of x. If

α is understood as the pseudocounts that generate a given multinomial probability vector then this formulation comes naturally.

Each Question implements its own filter, which means they are completely independent from other questions. As the student provides answers to a given question the estimate is updated. The Student implemented has 5 questions, with different degrees of difficulty (from the perspective of state estimation).

Particles refers to the number of particles used by the filters. *Iterations* refers to the number of times the student generates an answer, for dynamic questions the underlying state of the student changes in time. *Threads* specifies the number of times the experiment is repeated. To appreciate the results of a single instance it can be set to 1.

All results are plotted, with additional Mean Square Error information. The following curves are computed:

- Predictions: At each t an estimation for x_t is generated, the mean of the distribution is saved in predictions[t]. All curves below consider the value of the mean saved in predictions[t] as the estimation for t.
- Mean: At each t all estimations to that point are averaged, the resulting value corresponds to means[t].
- Weighted Mean: At each t the last 10 estimations, including the estimation at t, are averaged according to the weights = [0.02, 0.03, 0.05, 0.08, 0.08, 0.09, 0.10, 0.10, 0.20, 0.25], the resulting value corresponds to weighted_means[t].
- ML5: At each t the estimation most likely to have produced the last 5 samples, from the last 5 estimations including the estimation at t, is selected, the resulting value corresponds to ml5[t].
- ML10: At each t the estimation most likely to have produced the last 10 samples, from the last 10 estimations including the estimation at t, is selected, the resulting value corresponds to ml10[t].

If the option *plot* is set to 1 then all 5 curves will be plotted for each experiment, in addition to the plots generated for the averages.

It should be noted that:

- If the answers of the student given to each experiment were the same, the randomness in the sampling/resampling steps would still justify the repetition.
- Since all the answers are generated independently for each experiment the interpretation of the averaged values should take this into consideration.

Results

Both methods were tested with an a value of 25. The questions presented in this section are 4-dimensional: there are only 4 possible answers from where to choose. Question 1 corresponds to a static student, one which does not modify x_t throughout the whole experiment. Question 5 is a dynamic question, one which has a student with varying x_t. In all plots, the value of the true state of the student is plotted in blue. Each dimension of the 4-dimensional vector is plotted separatedly, for convenience, and each dimension is called a State, where State 1 corresponds to the first dimension, and so forth. All plots contain MSE information of the curves, with respect to the true curve, where the curve MSE corresponds to the average of the MSE of each separate curve (for each State), and the MSE of each State is indicated on the corresponding plot.

- *First approach*: Figures 1 to 4 correspond to plots of the predictions, in red, the weighted mean, in black, and the goal, which corresponds to the true state x_t of the student, in blue, regarding the first question. This question is static, in the sense that the student does not modify x_t. The weighted mean has lower MSE for all States, as it constitutes a smoother curve. The high variability of the predictions allow them to adjust quickly to changes, but that is missed on the static estimation. Figures 5 to 8 compare the weighted mean with the mean, for the same experiment. It is clear that the mean approximates the real curve much better, as can be evidenced from the lower MSE, but this is only due to the static nature of the example, in a dynamic environment the mean prediction would be meaningless.
  Figures 9 to 12 correspond to plots of the predictions and the weighted mean for question 5, which is dynamic. The change in the state of the student can be evidenced on the blue line. Once again, the weighted mean gives a better approximation than the predictions for all curves, even if it is an average over points with very changing underlying conditions (the change in x_t) they still manage to be responsive enough to keep on par with the predictions themselves. Figures 13-16 show the stark contrast with the mean of the static question. Here the mean is a much poorer approximation to the real curve, but it should be noted that on a signal with significant noise it could be the only curve which yields useful information.

- *Second approach*: Figures 17 to 20 correspond to plots of the predictions and the weighted mean for question 5 using a w of 0.8. While, according to the MSE, the results of the predictions are in par with the weighted mean of the previous method, it should be noted that a weighted average hurts the prediction (increasing the MSE). The variance of the predictions is much smaller, as should be expected by the method used to compute α which takes into account the mean of the current posterior approximation. This convergence to the mean makes the curve less responsive, as can be evidenced in Figure 20 where the curve lags behind the change in the value of x_t in contrast to Figure 12 where the curve rapidly adapts.
  Figures 21 to 24 correspond to plots of the predictions and the weighted mean for question 5 using a w of 0.85 and Figures 25 to 28 correspond to plots of the predictions and the weighted mean for question 5 using a w of 0.90. As w increases the system becomes increasingly more variable, as less emphasis is given to the pseudocount component added by the mean.

It should be noted that these results were computed using 1000 particles for each thread, which is a cheap approximation in terms of computing resources.

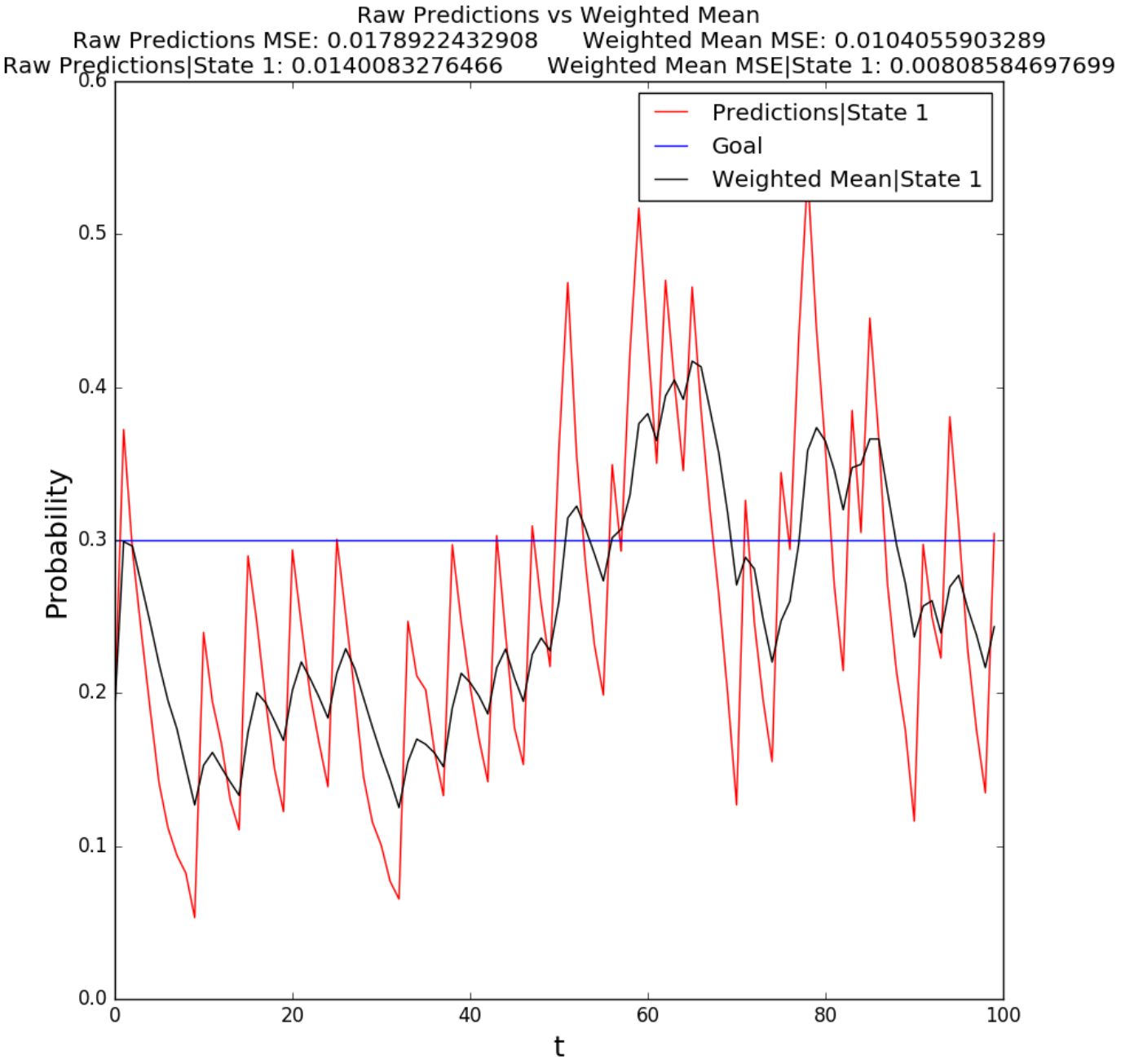

Figure 1: First Method - Question 1 - State 1
Predictions vs Weighted Mean

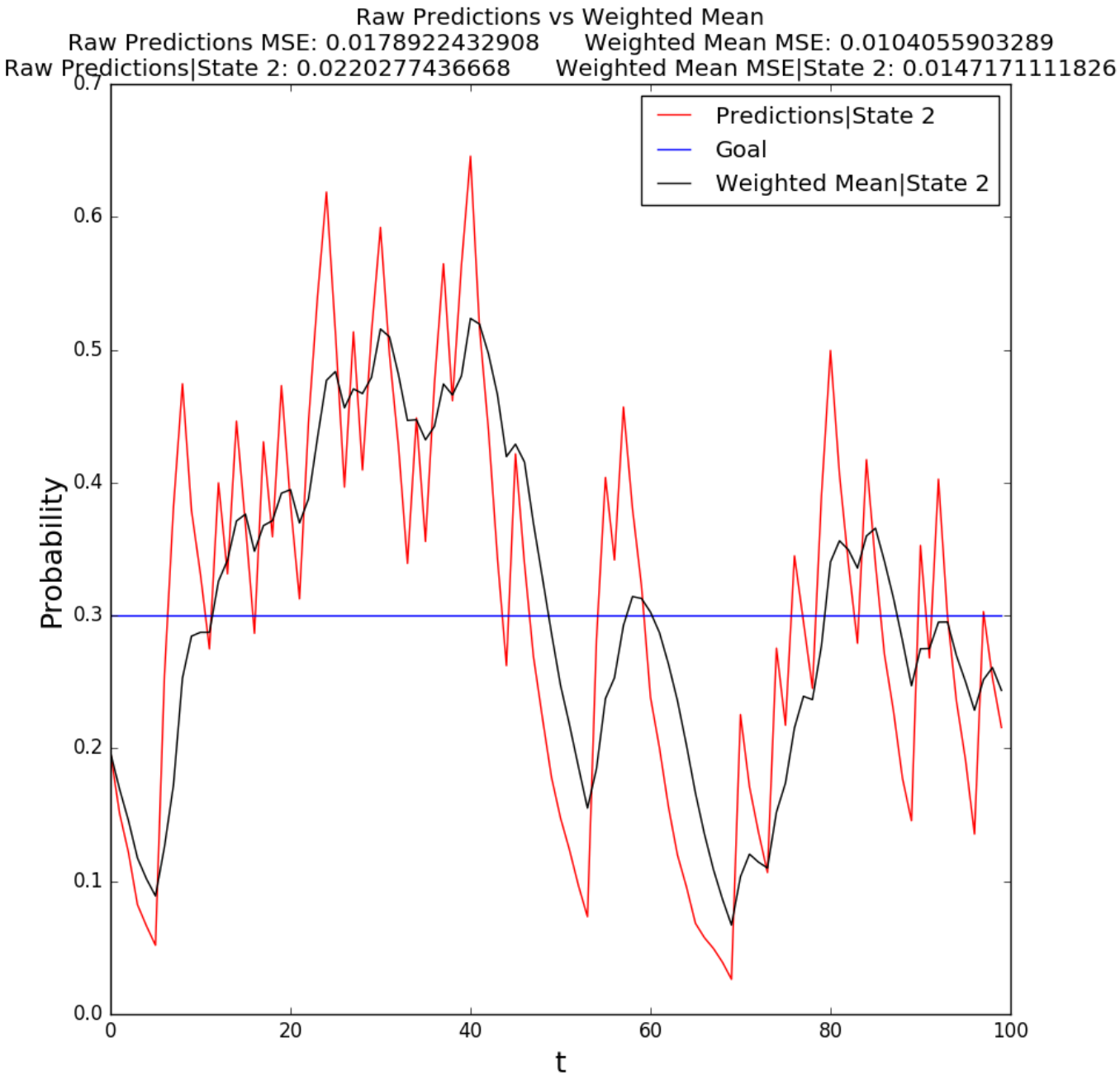

Figure 2: First Method - Question 1 - State 2
Predictions vs Weighted Mean

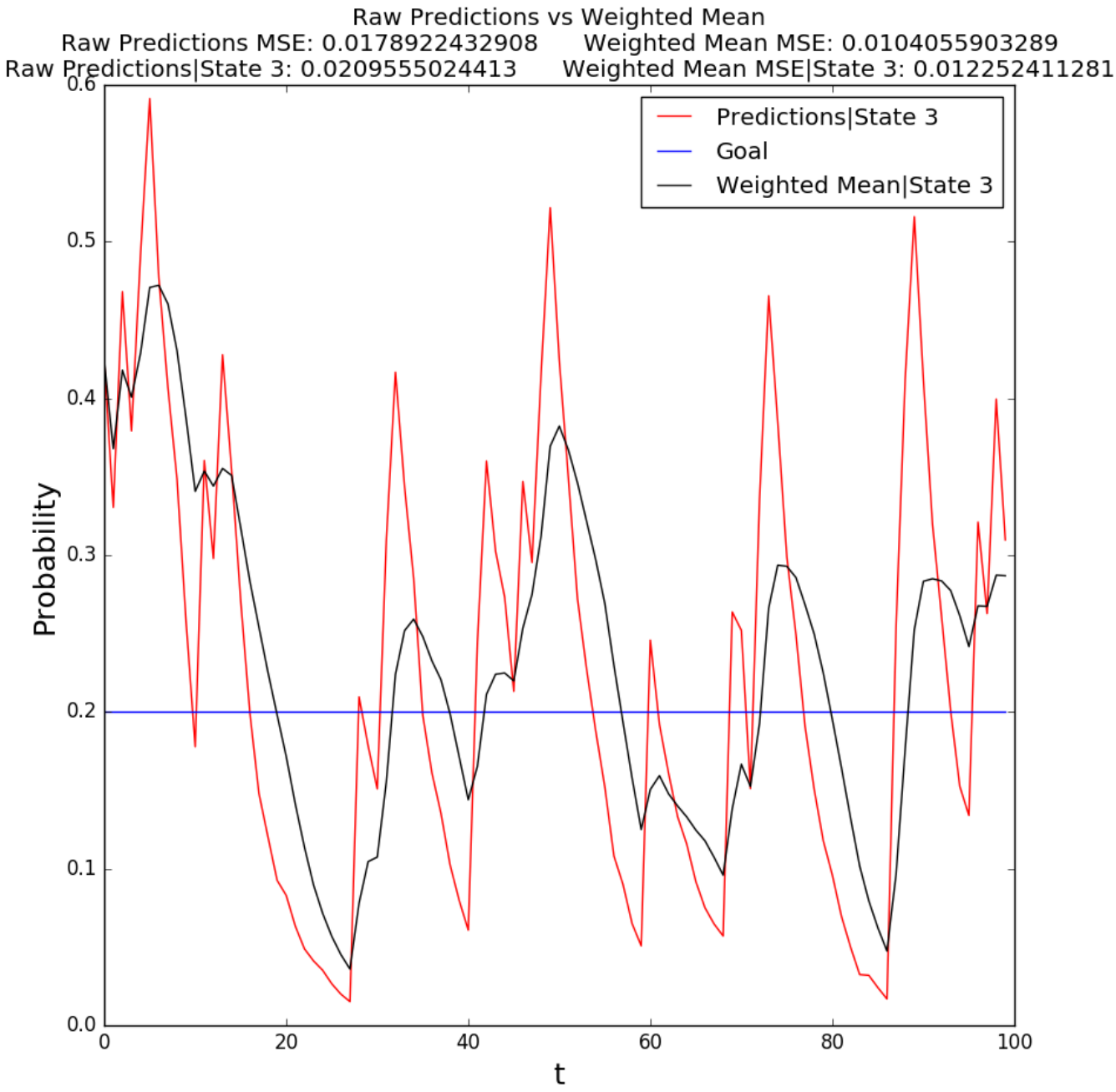

Figure 3: First Method - Question 1 - State 3
Predictions vs Weighted Mean

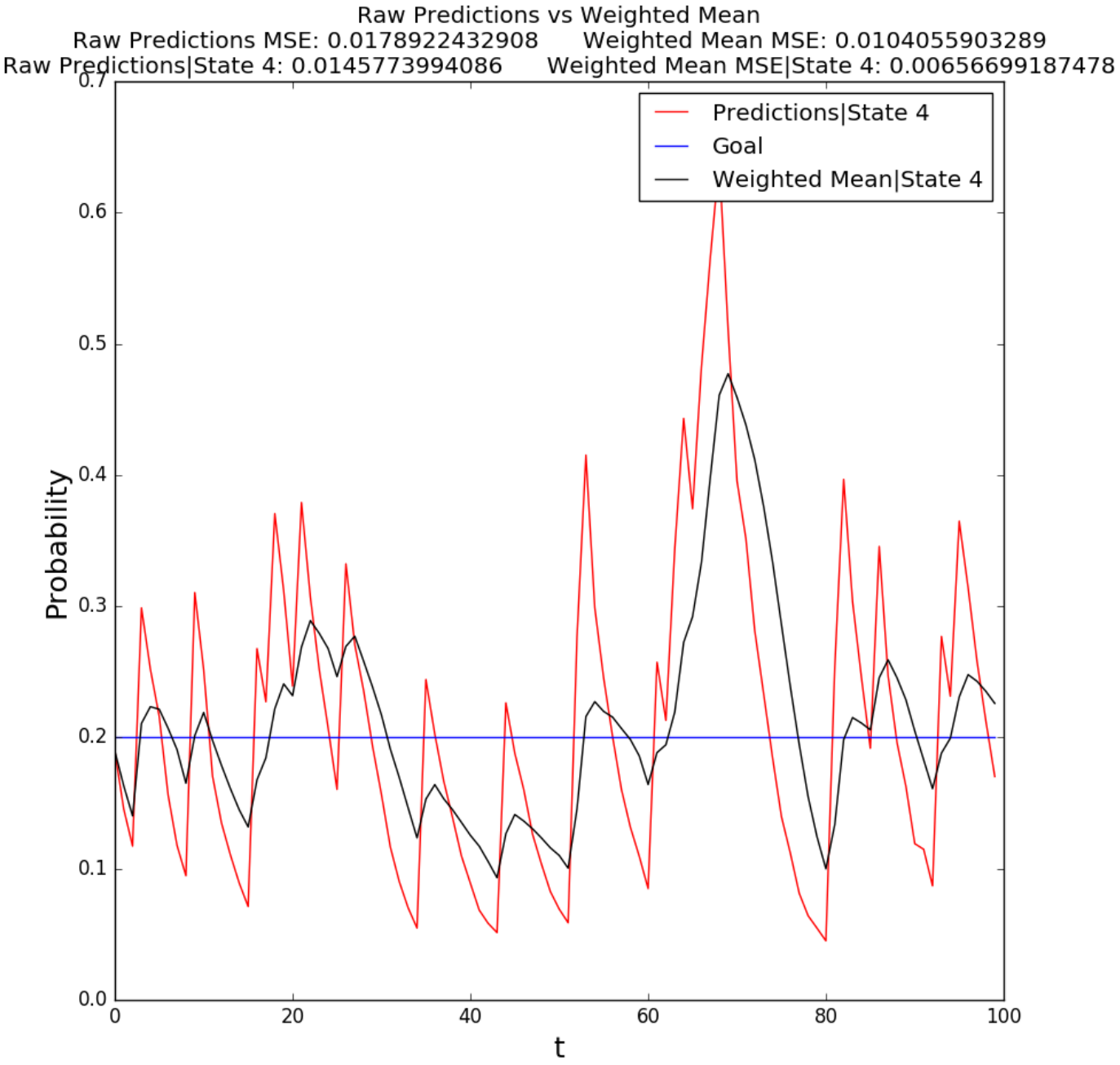

Figure 4: First Method - Question 1 - State 4
Predictions vs Weighted Mean

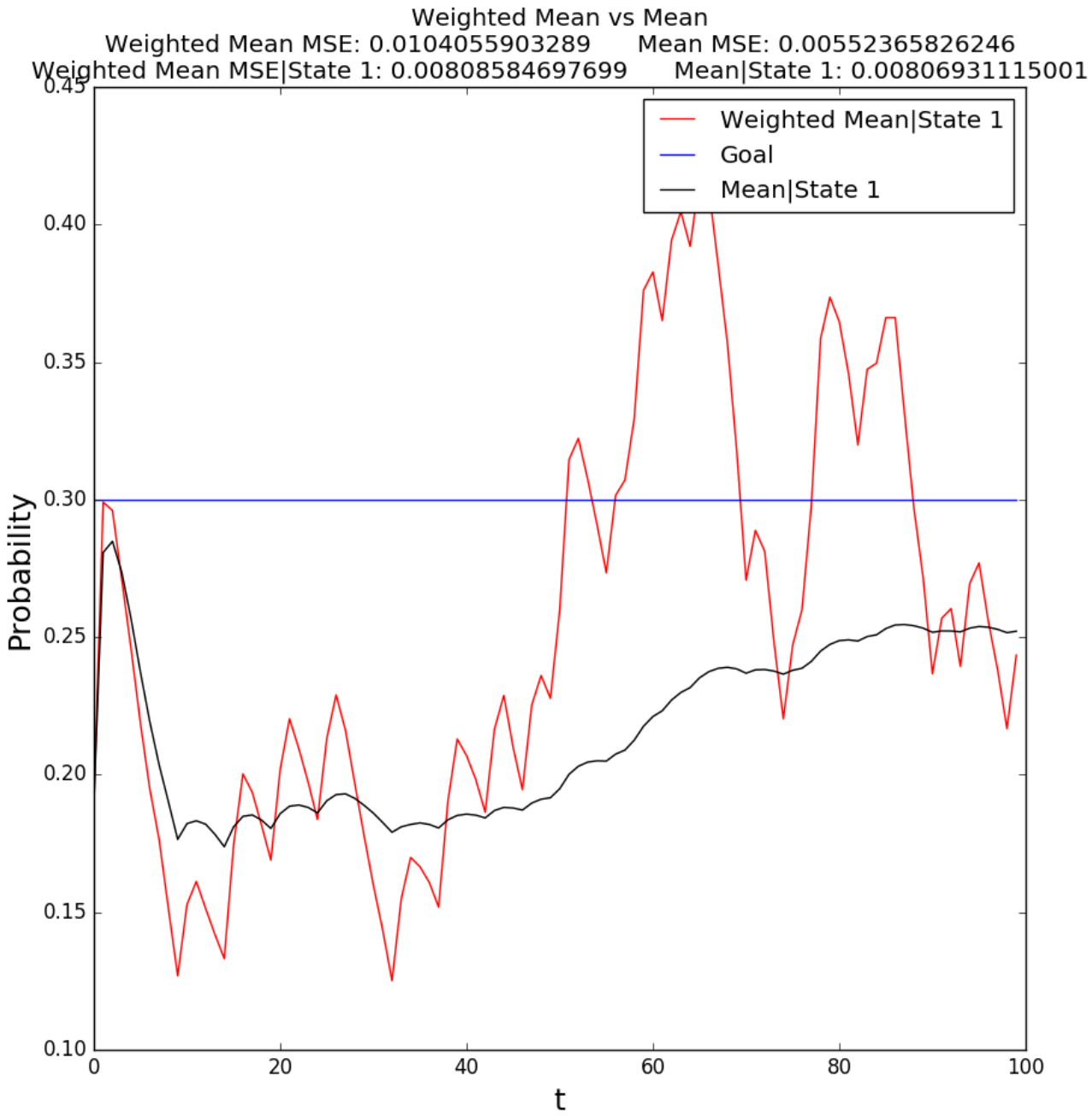

Figure 5: First Method - Question 1 - State 1
Weighted Mean vs Mean

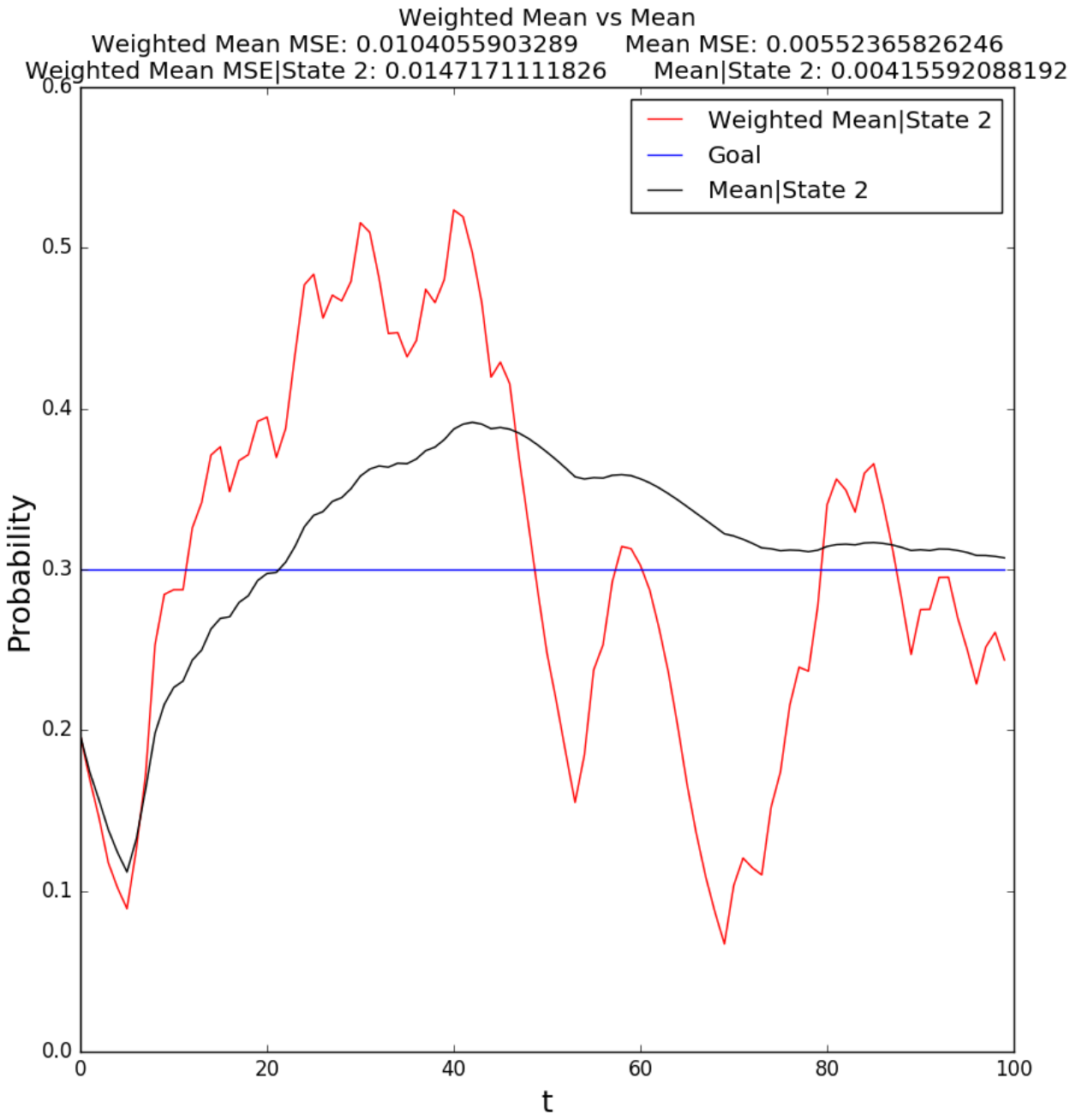

Figure 6: First Method - Question 1 - State 2
Weighted Mean vs Mean

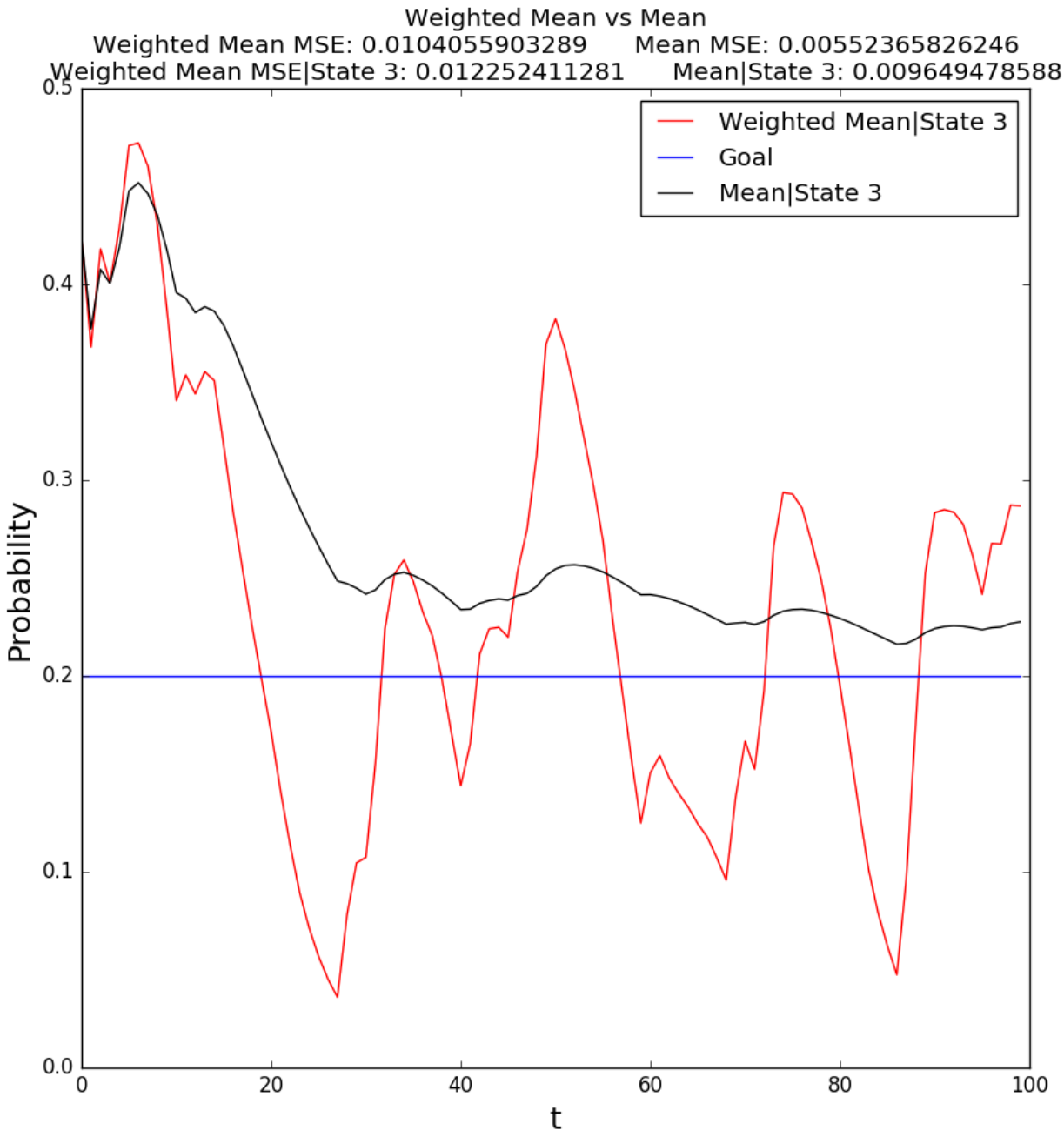

Figure 7: First Method - Question 1 - State 3
Weighted Mean vs Mean

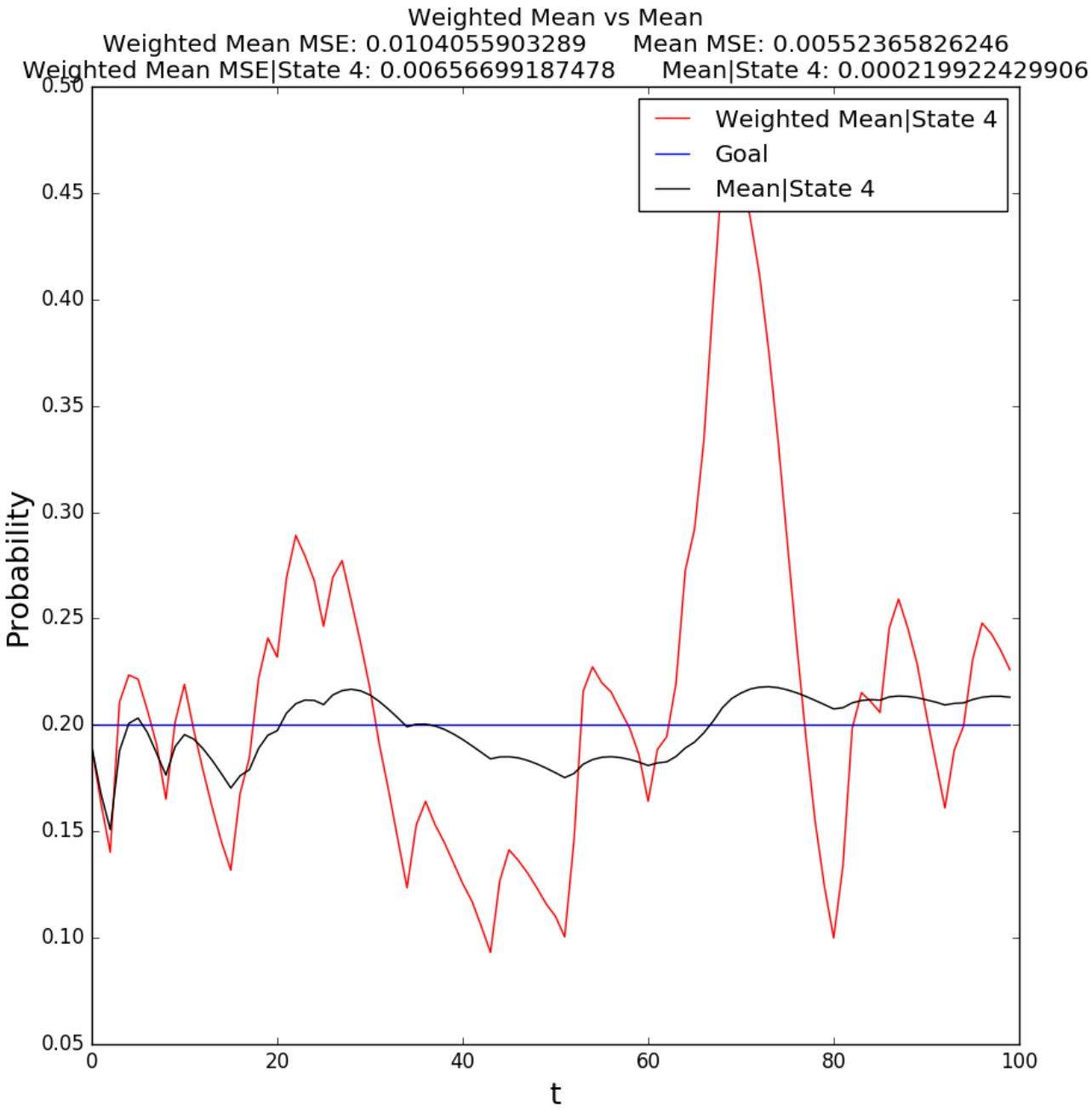

Figure 8: First Method - Question 1 - State 4
Weighted Mean vs Mean

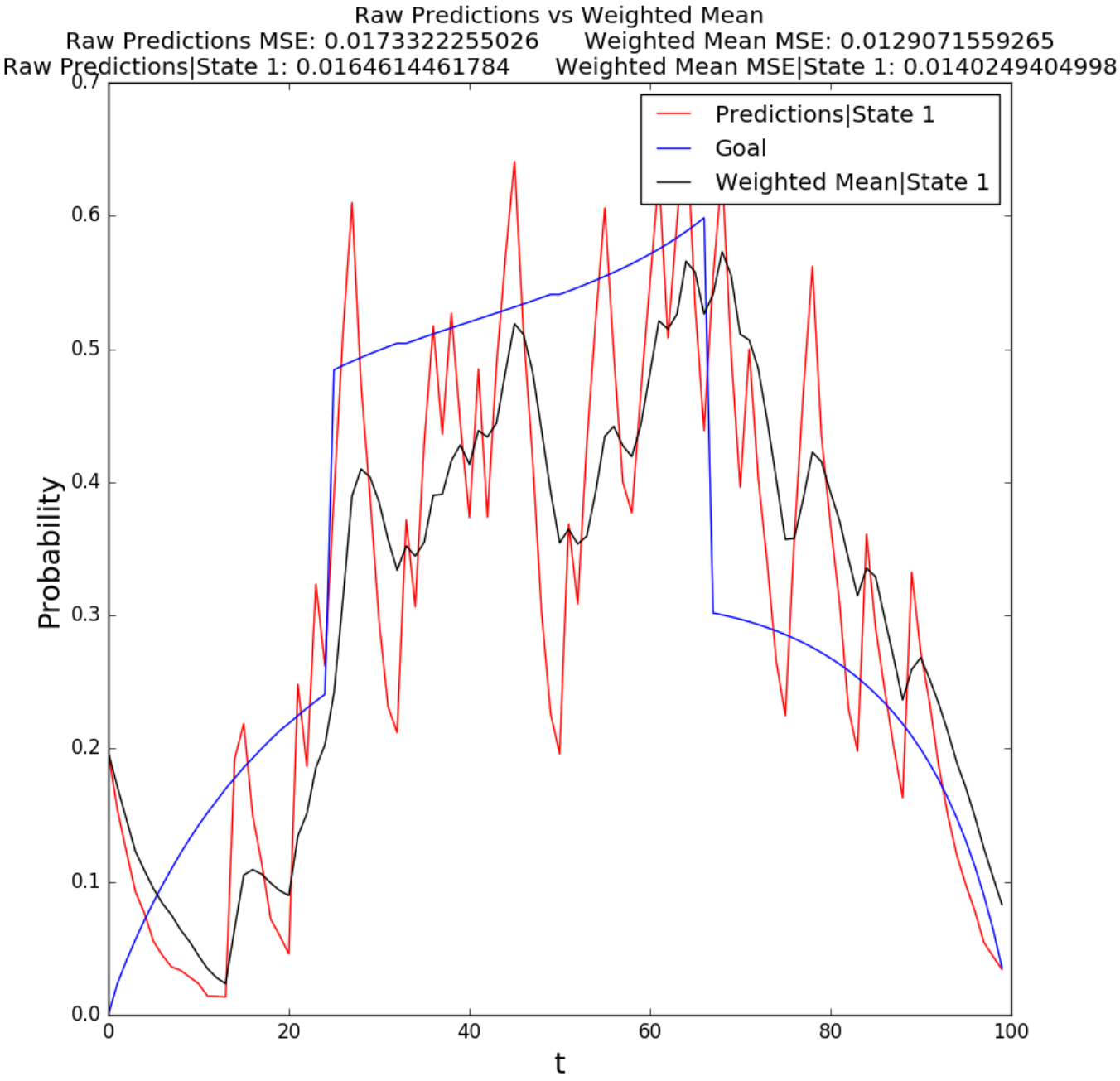

Figure 9: First Method - Question 5 - State 1
Predictions vs Weighted Mean

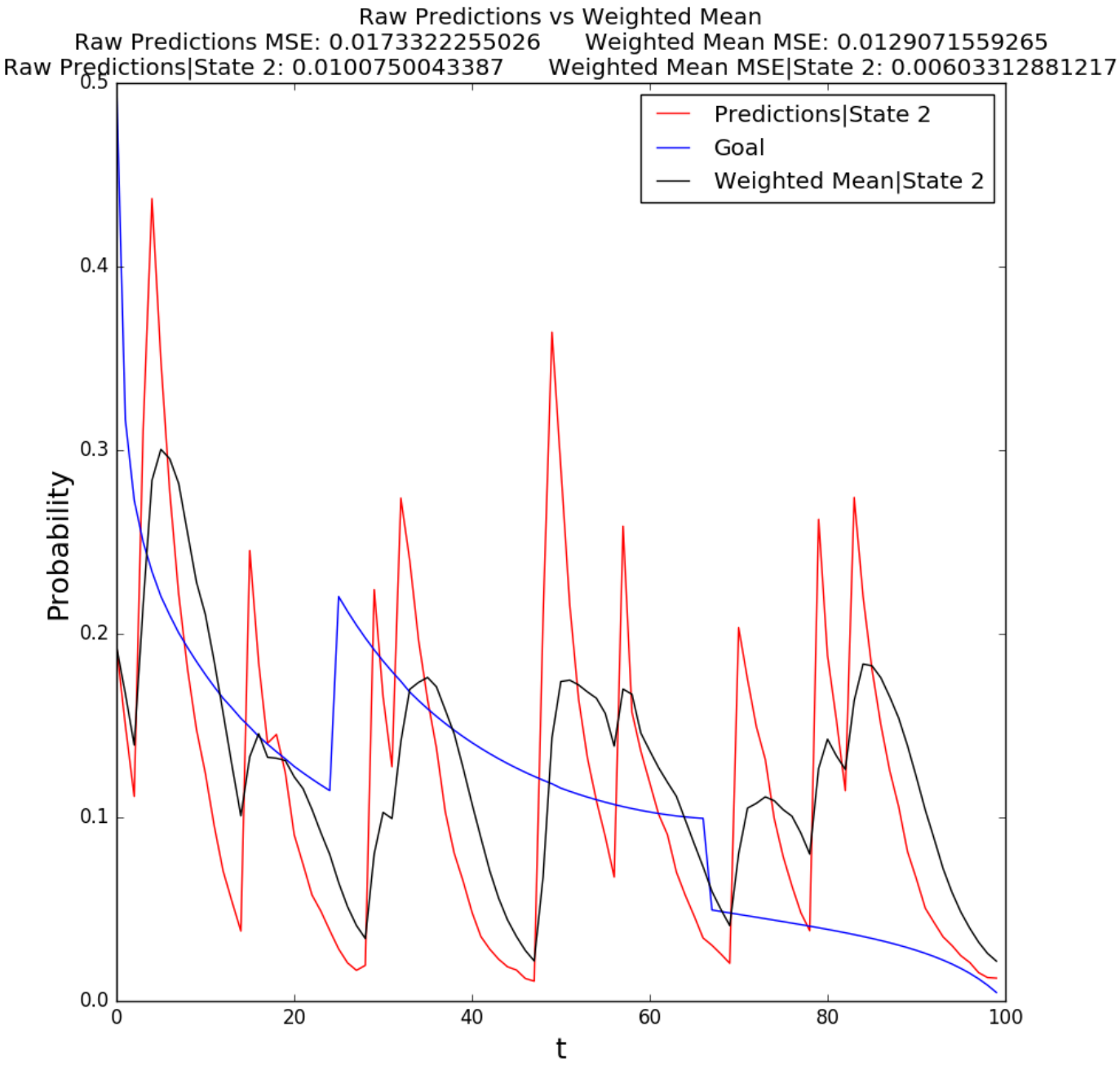

Figure 10: First Method - Question 5 - State 2
Predictions vs Weighted Mean

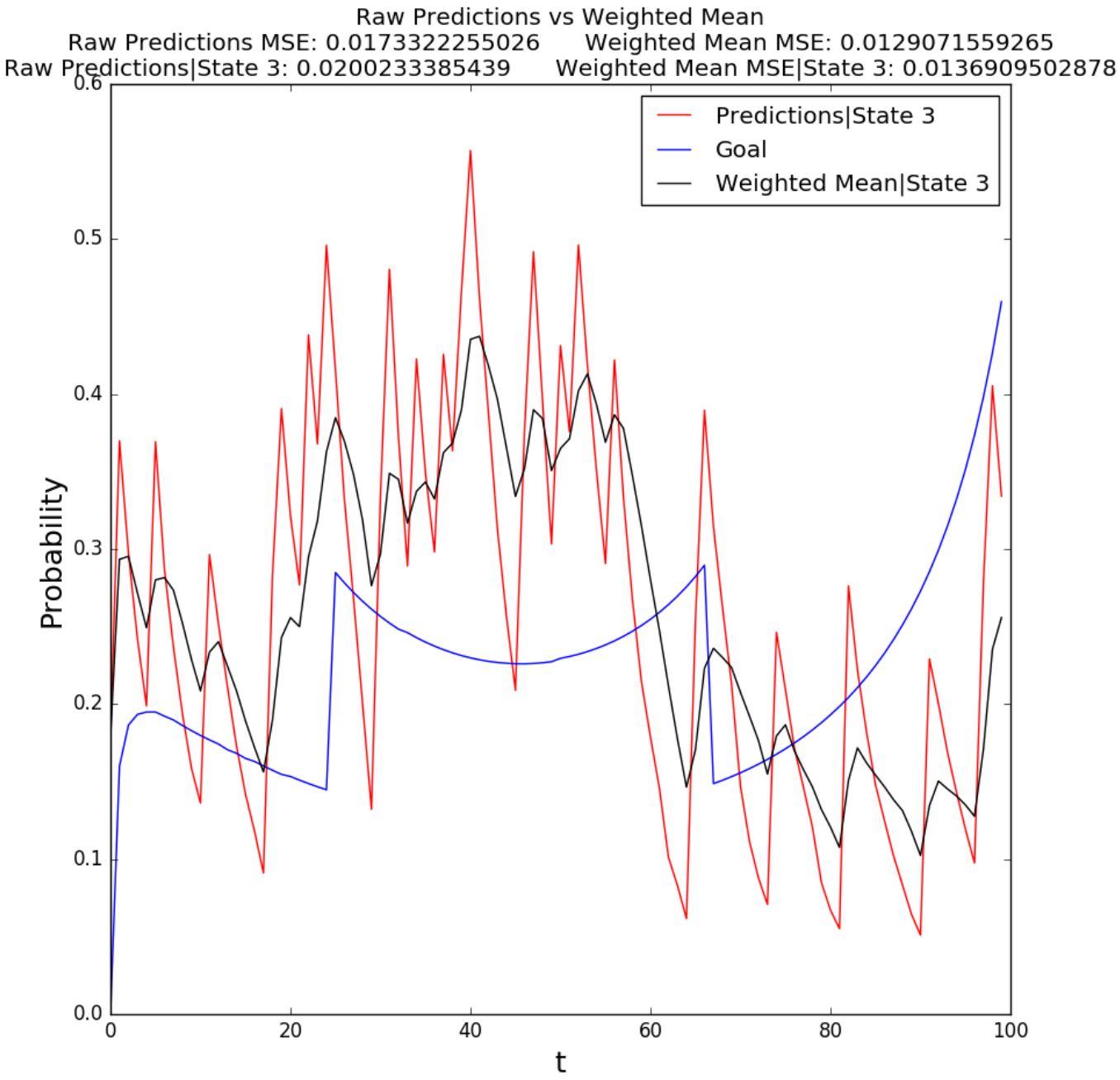

Figure 11: First Method - Question 5 - State 3
Predictions vs Weighted Mean

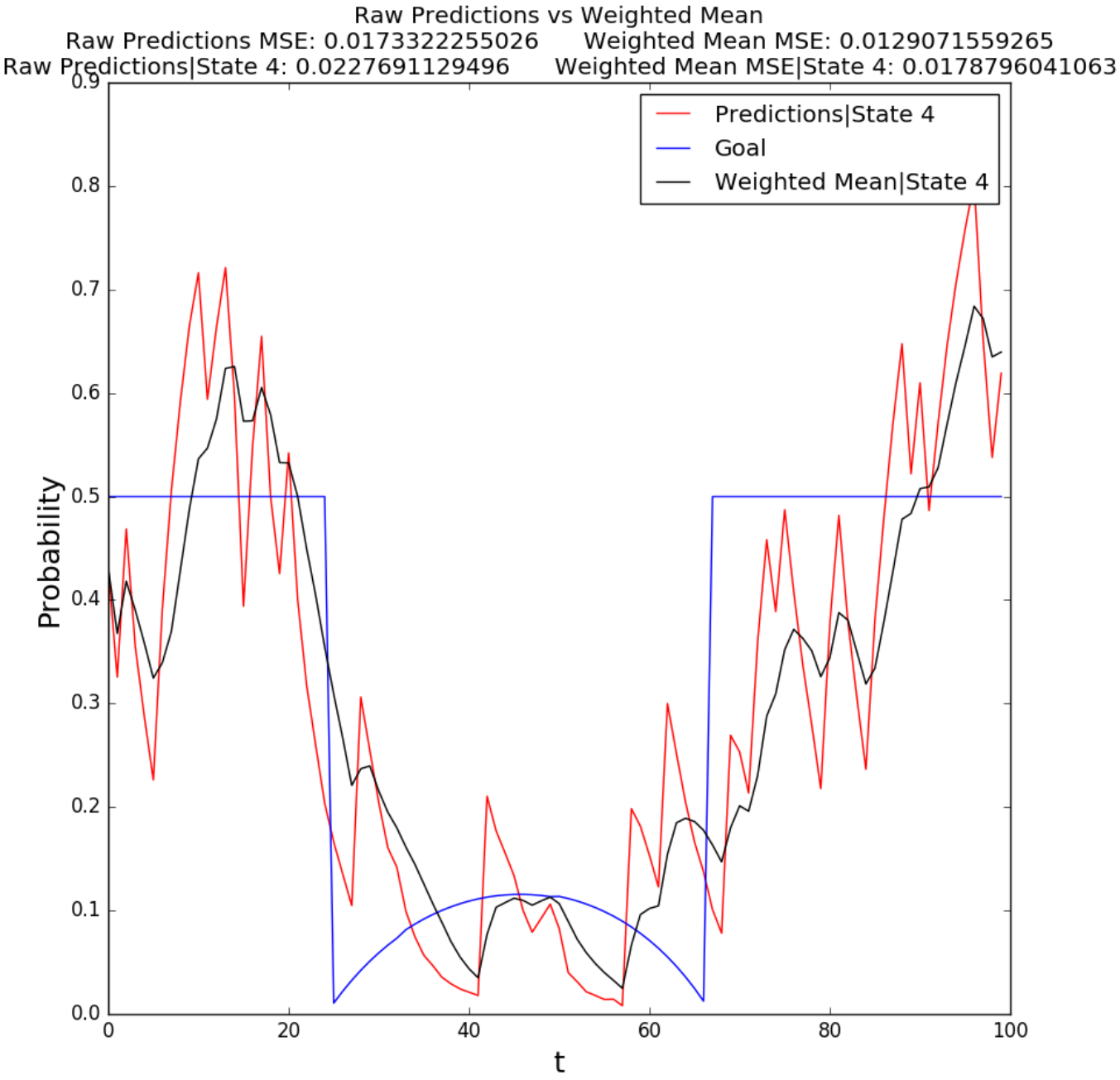

Figure 12: First Method - Question 5 - State 4
Predictions vs Weighted Mean

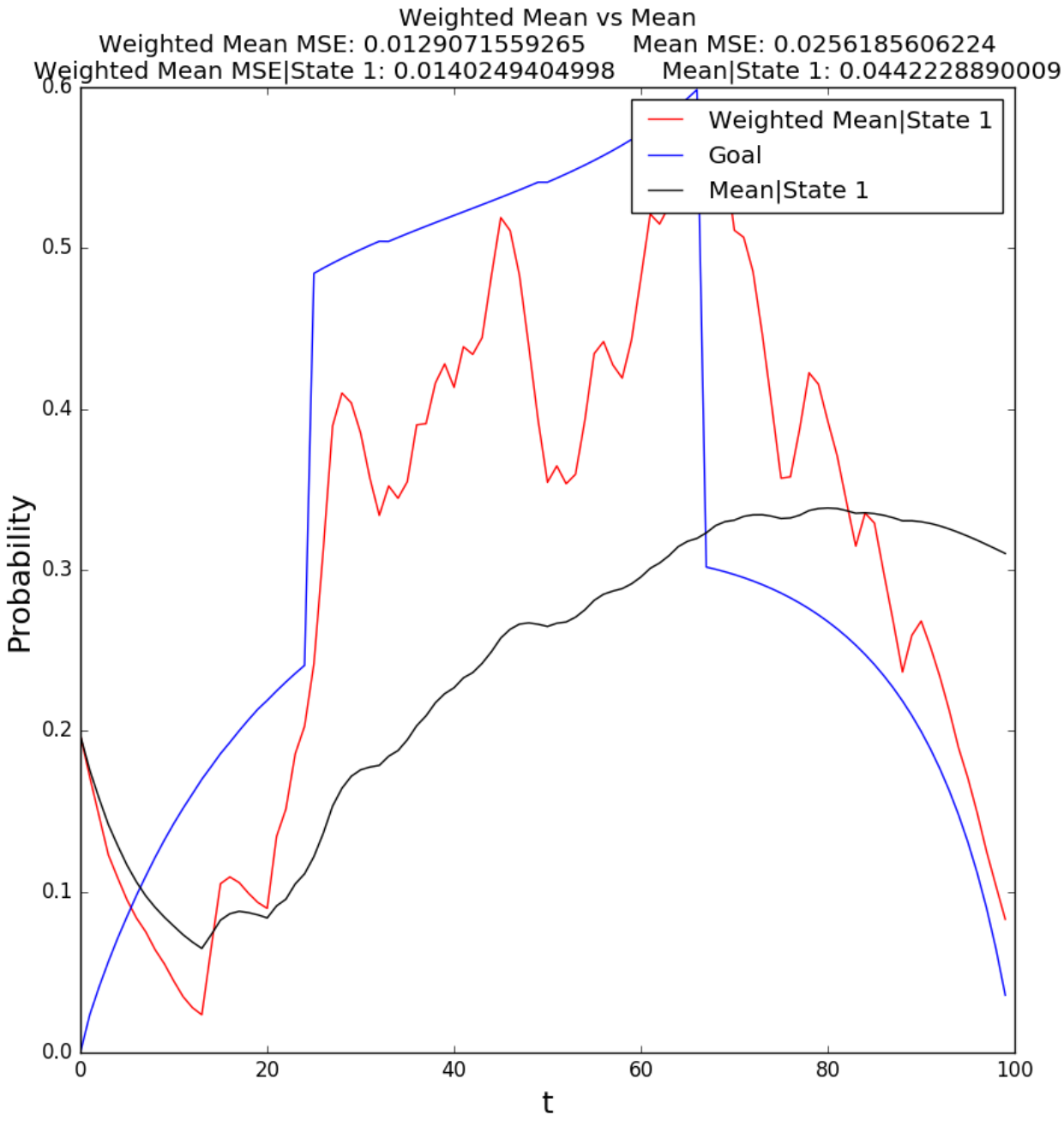

Figure 13: First Method - Question 5 - State 1
Weighted Mean vs Mean

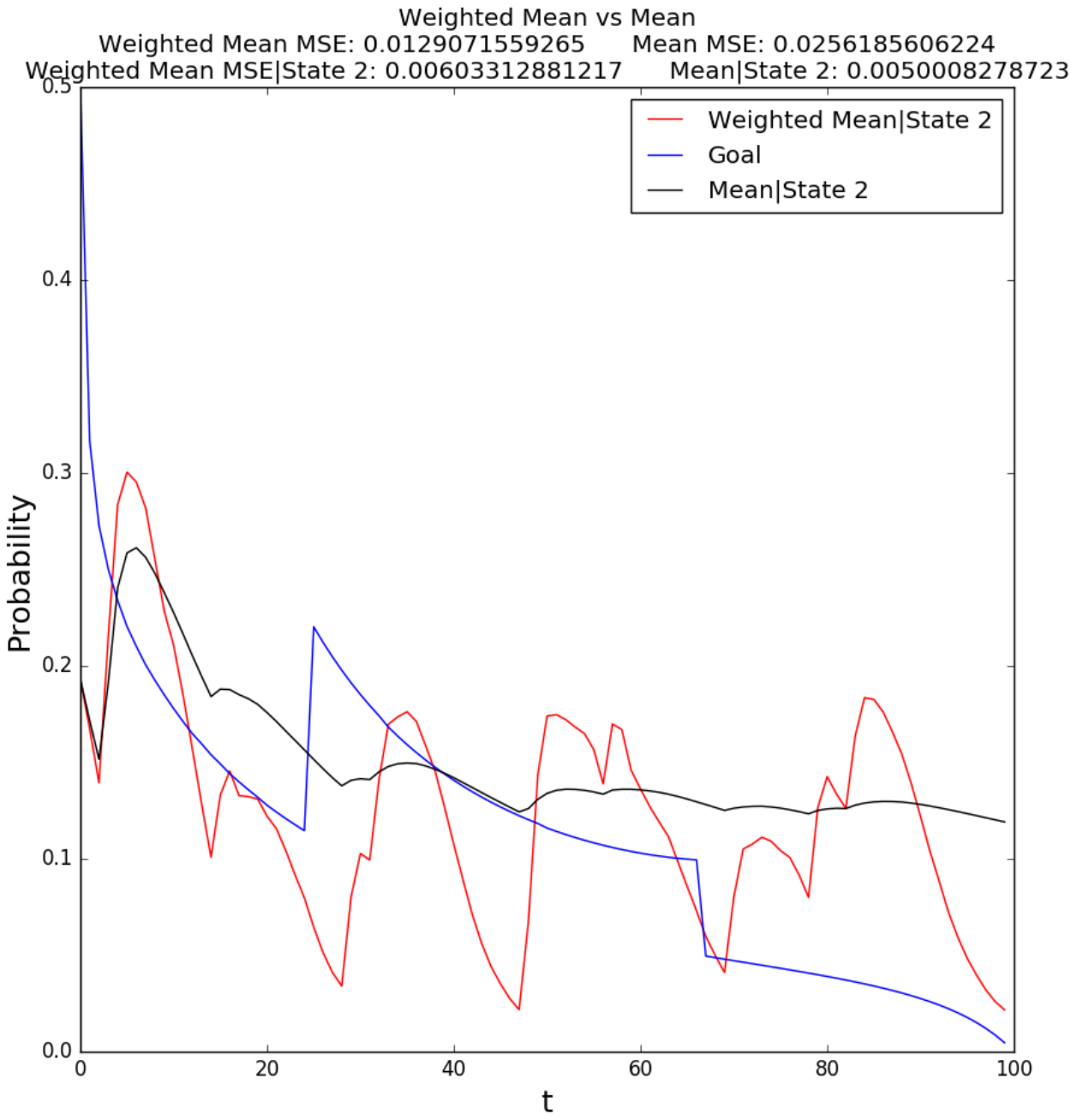

Figure 14: First Method - Question 5 - State 2
Weighted Mean vs Mean

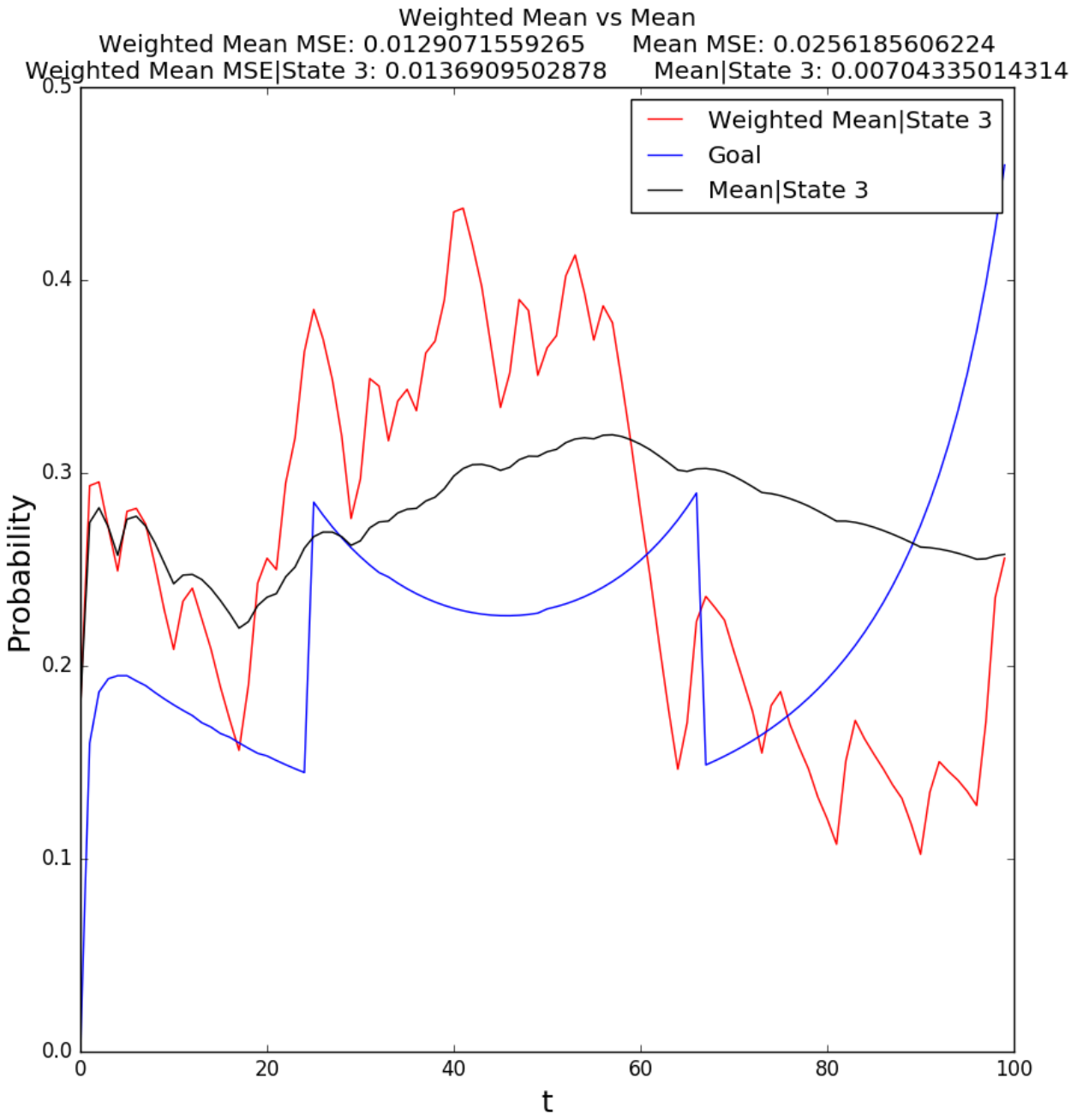

Figure 15: First Method - Question 5 - State 3
Weighted Mean vs Mean

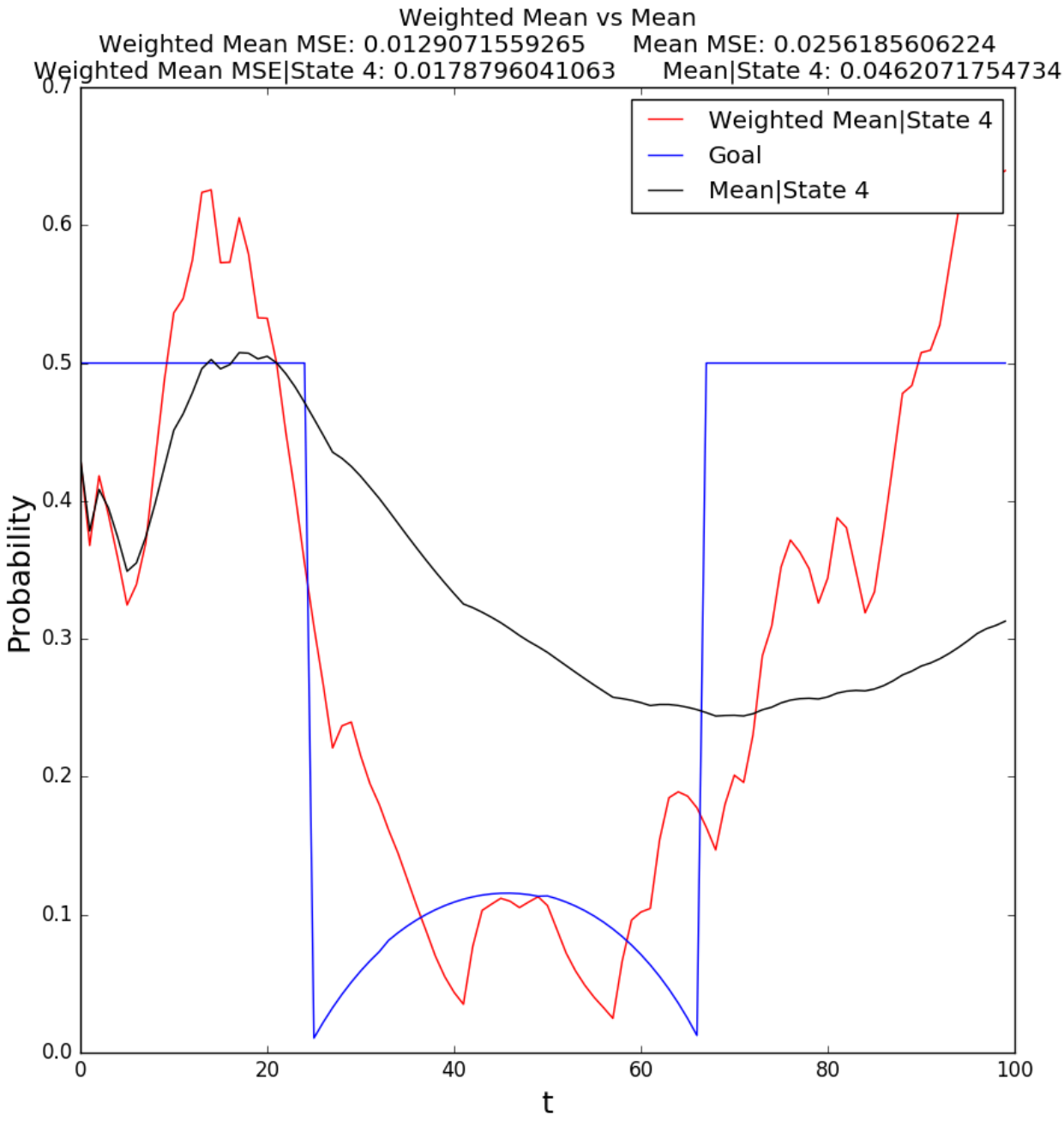

Figure 16: First Method - Question 5 - State 4
Weighted Mean vs Mean

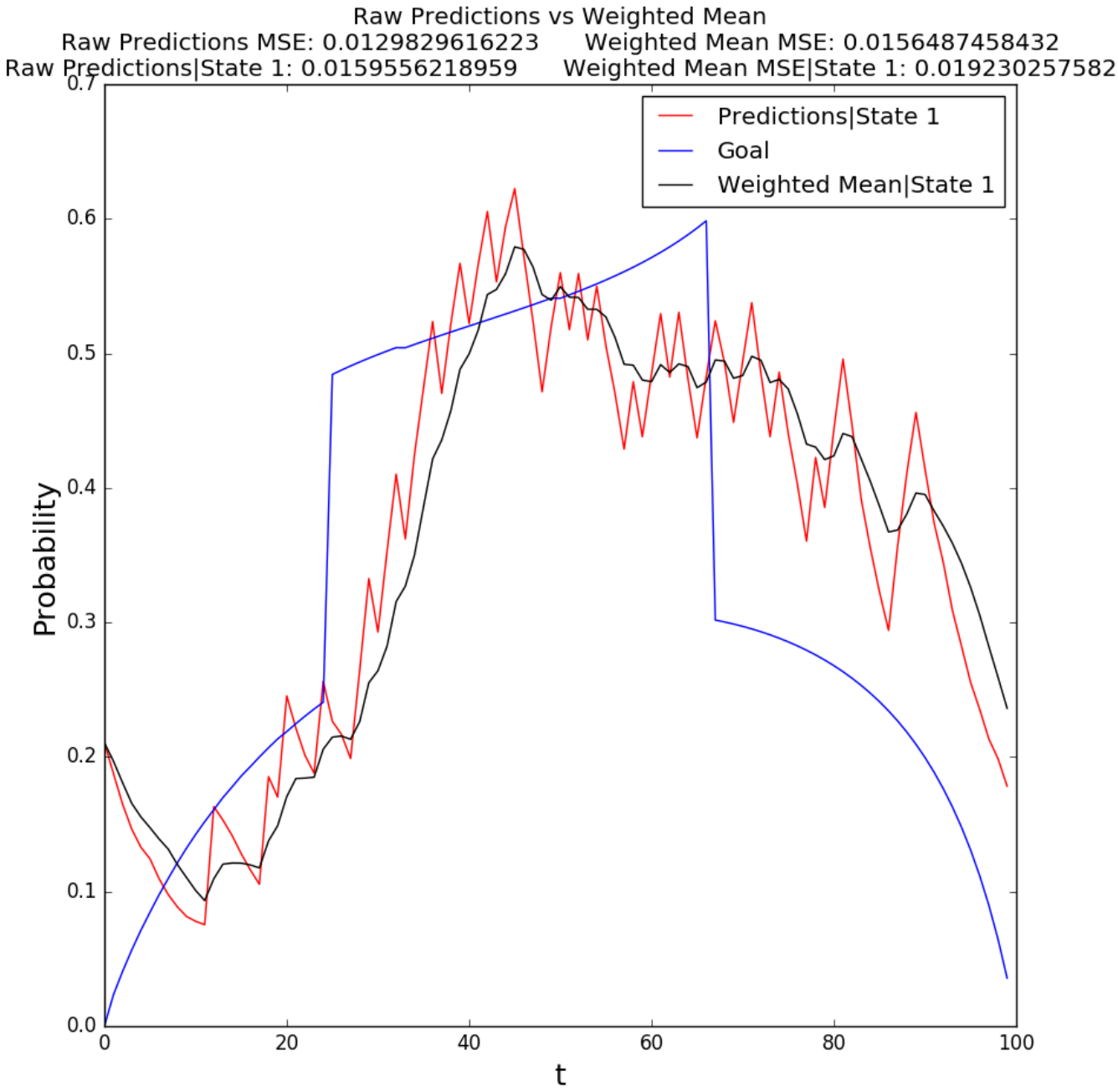

Figure 17: Second Method - Question 5 - State 1
Predictions vs Weighted Mean
w = 0.80

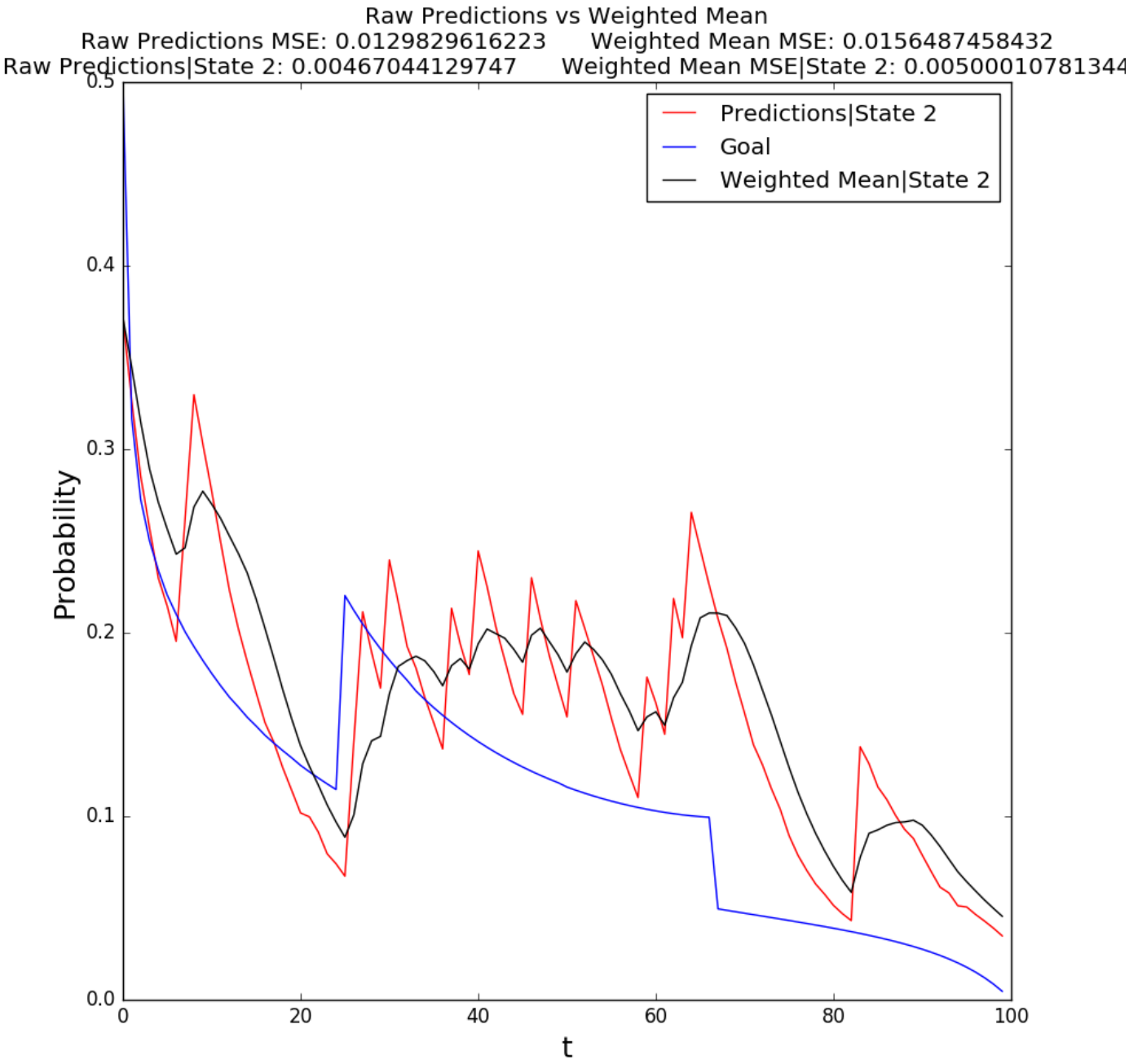

Figure 18: Second Method - Question 5 - State 2
Predictions vs Weighted Mean
w = 0.80

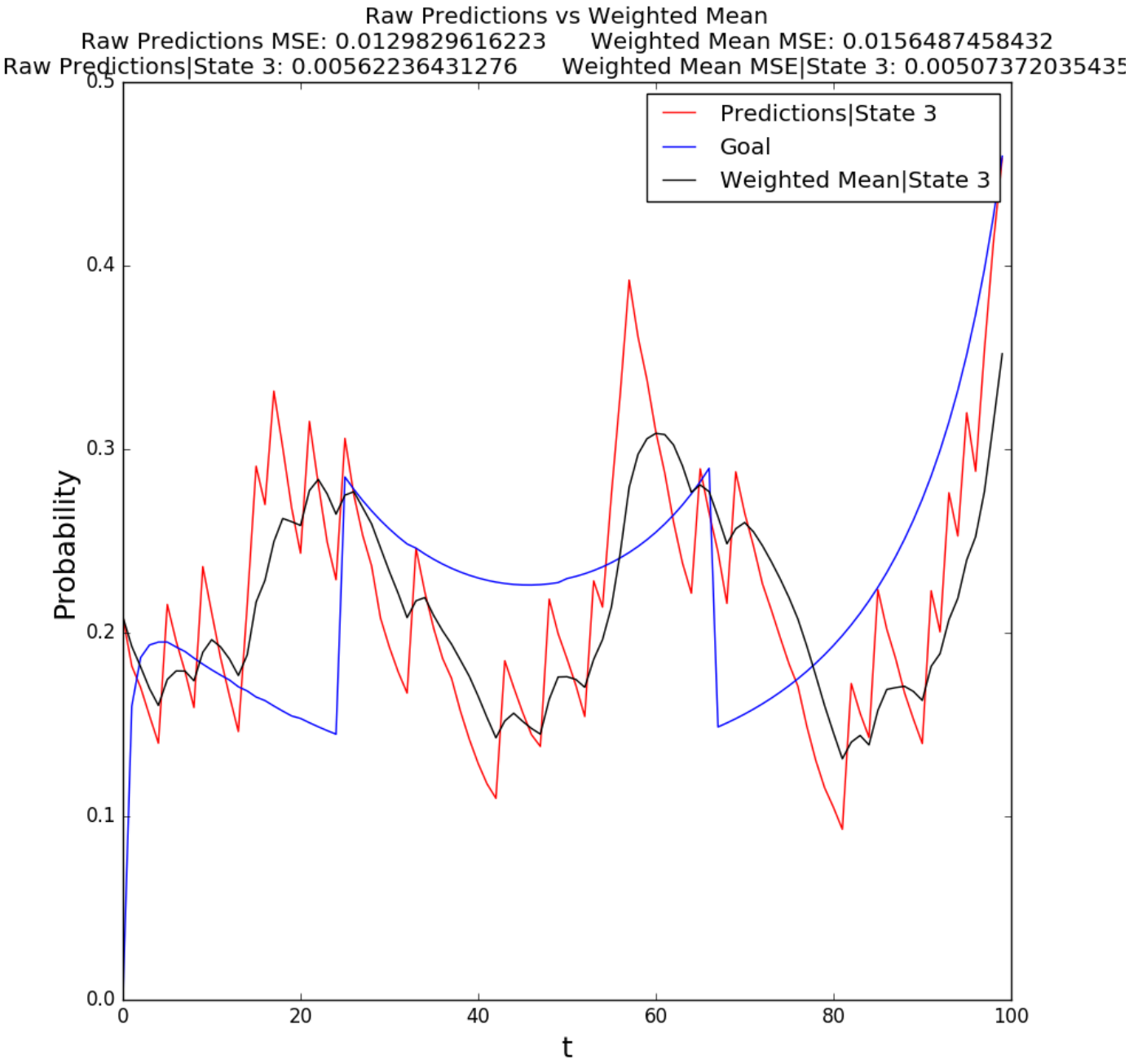

Figure 19: Second Method - Question 5 - State 3
Predictions vs Weighted Mean
w = 0.80

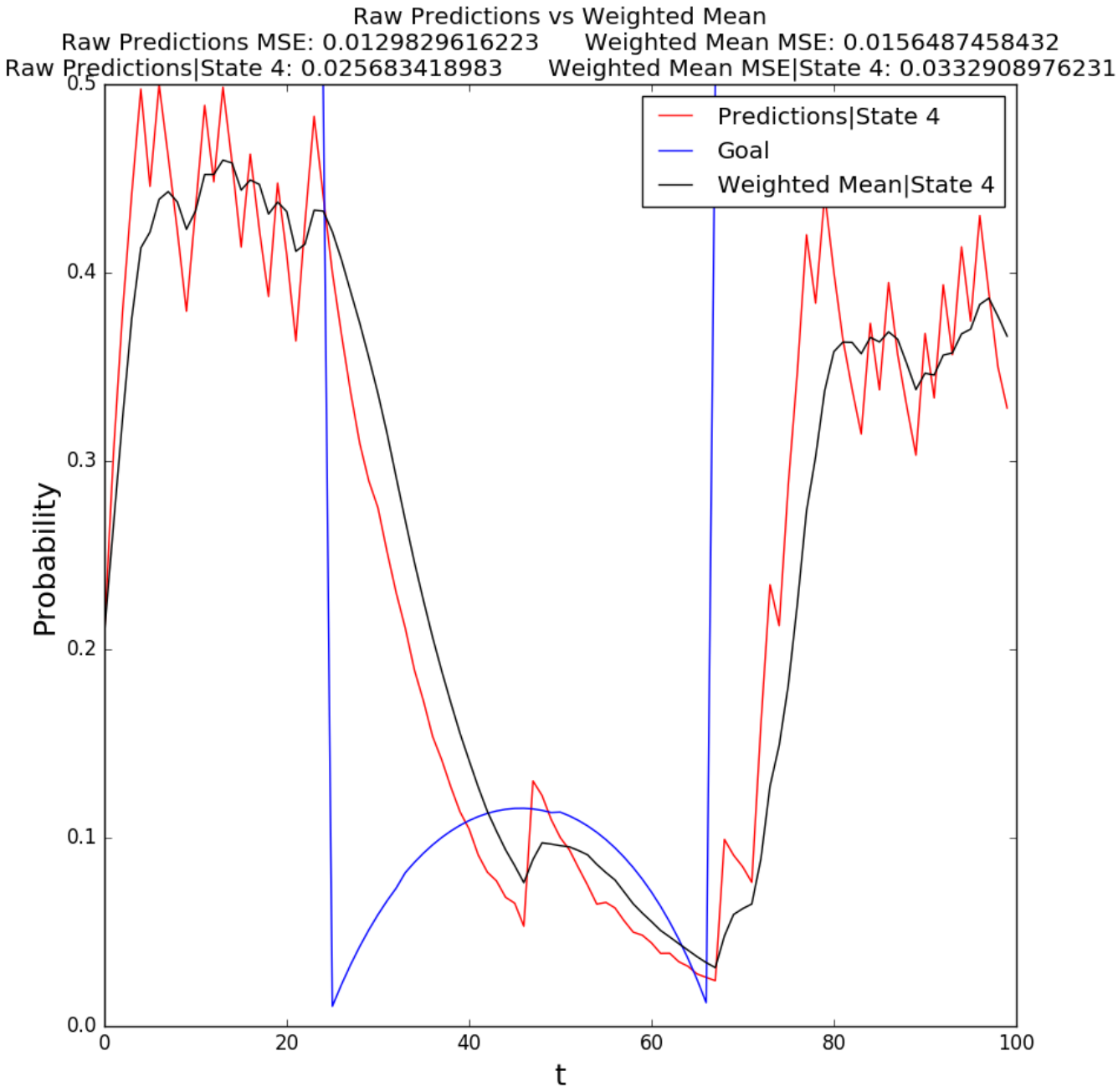

Figure 20: Second Method - Question 5 - State 4
Predictions vs Weighted Mean
w = 0.80

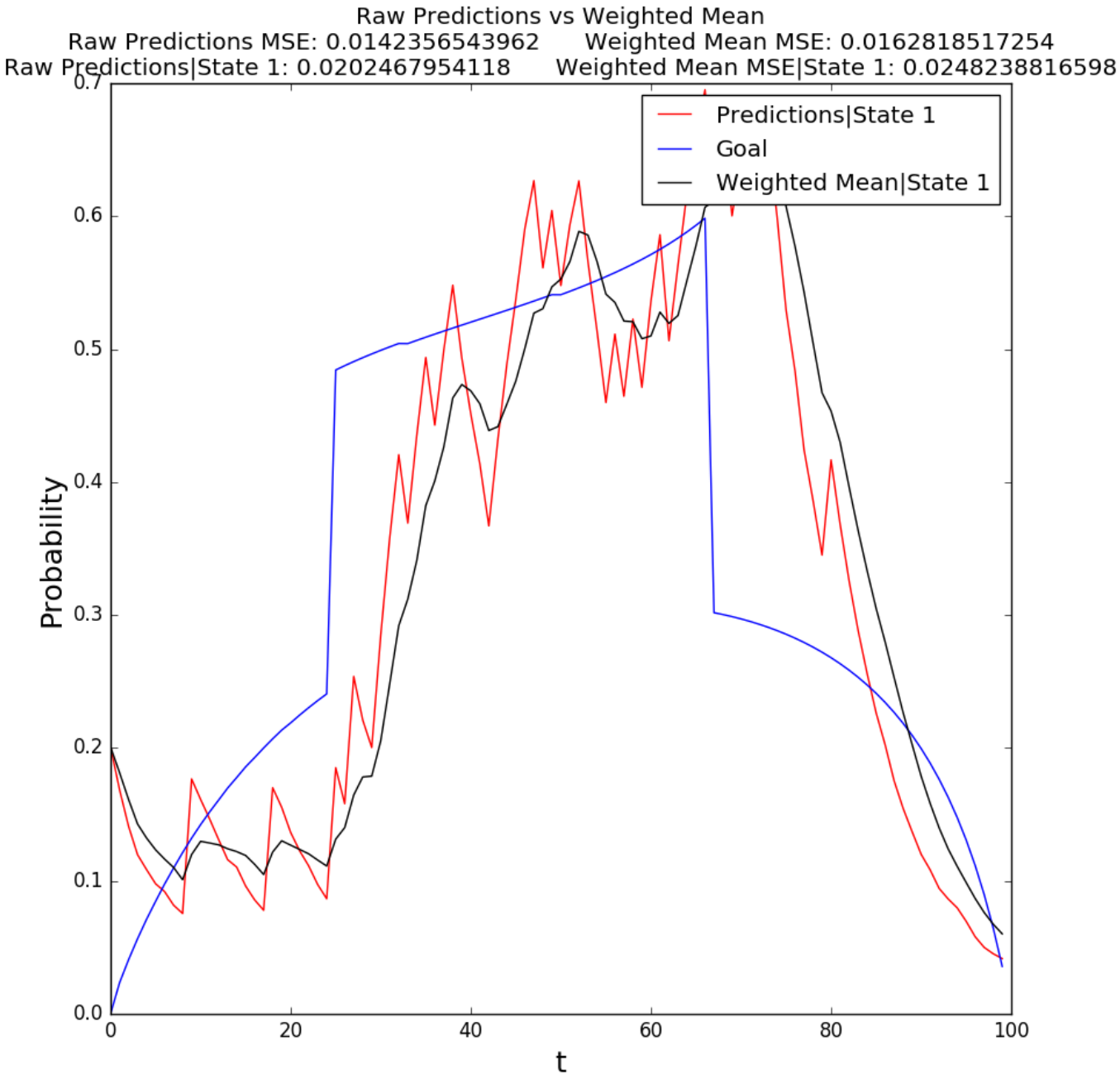

Figure 21: Second Method - Question 5 - State 1
Predictions vs Weighted Mean
w = 0.85

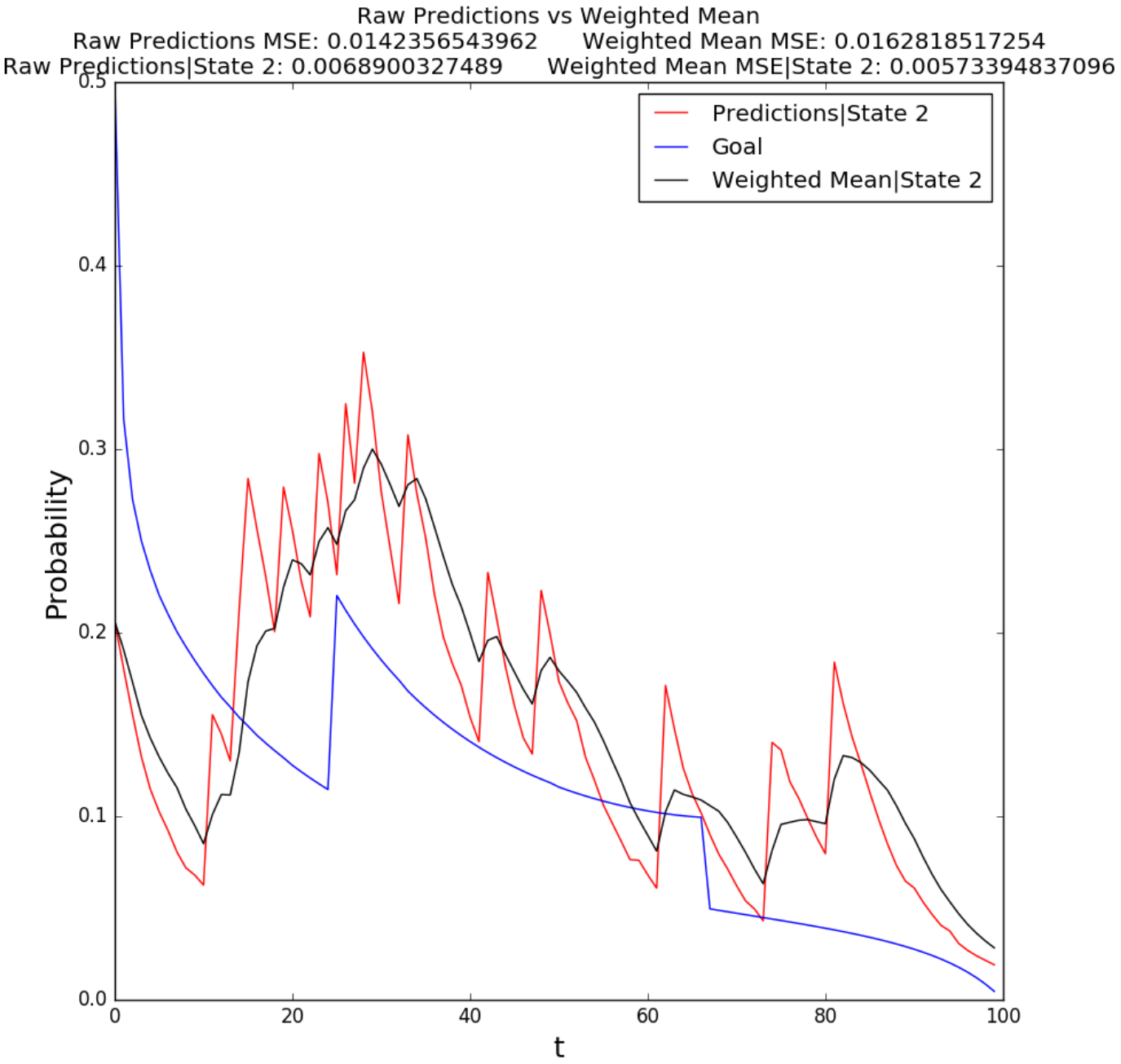

Figure 22: Second Method - Question 5 - State 2
Predictions vs Weighted Mean
w = 0.85

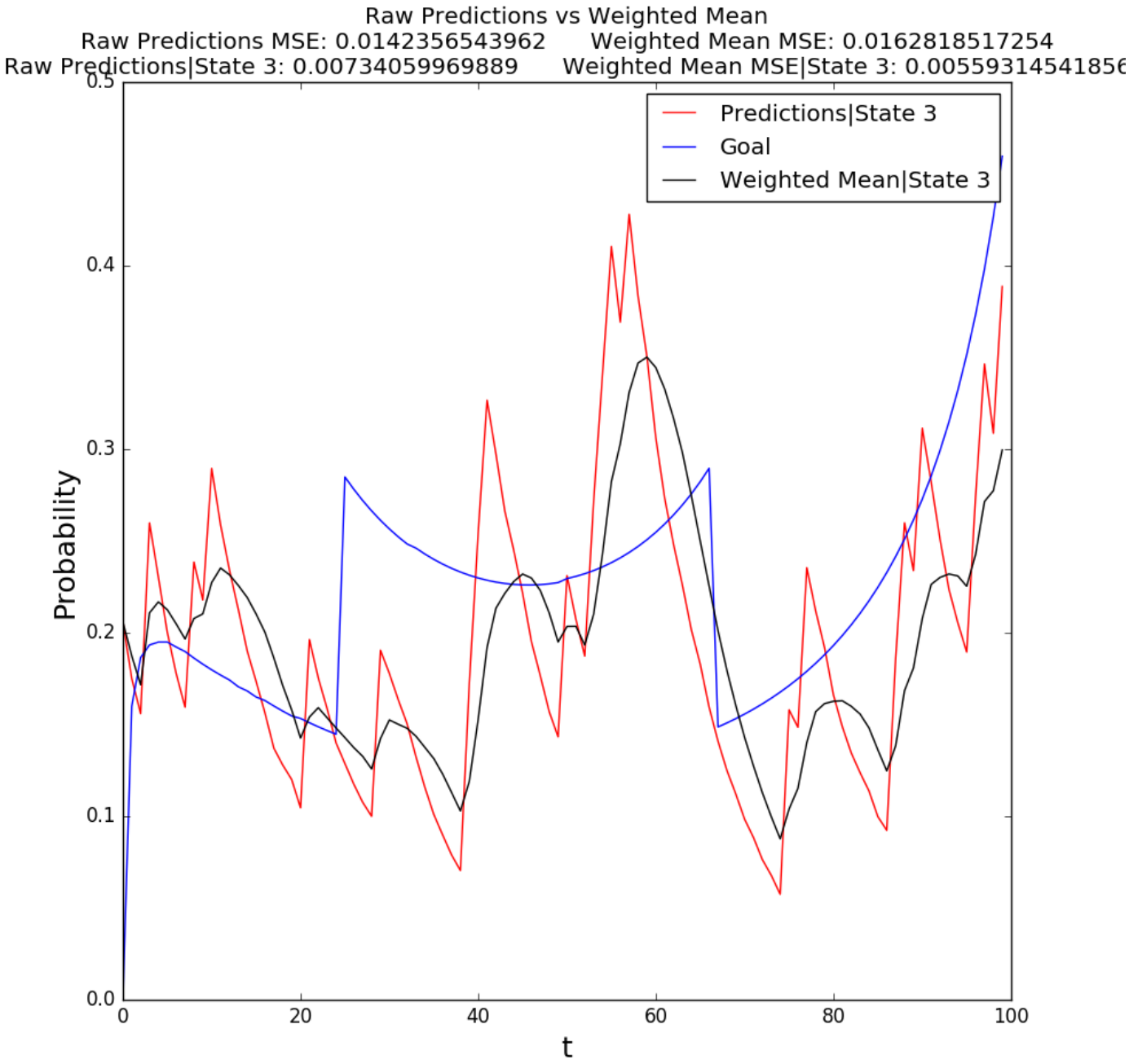

Figure 23: Second Method - Question 5 - State 3
Predictions vs Weighted Mean
w = 0.85

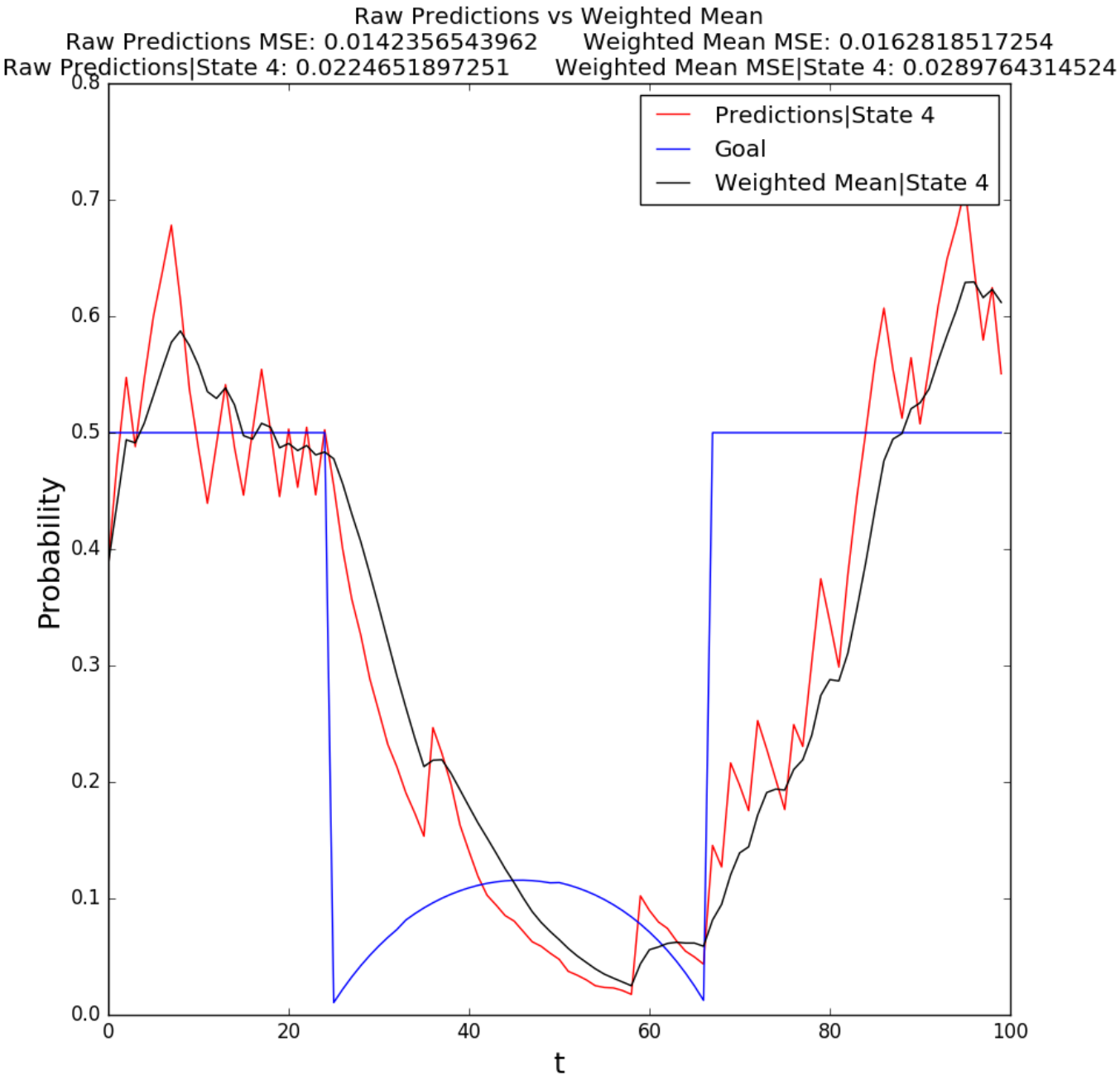

Figure 24: Second Method - Question 5 - State 4
Predictions vs Weighted Mean
w = 0.85

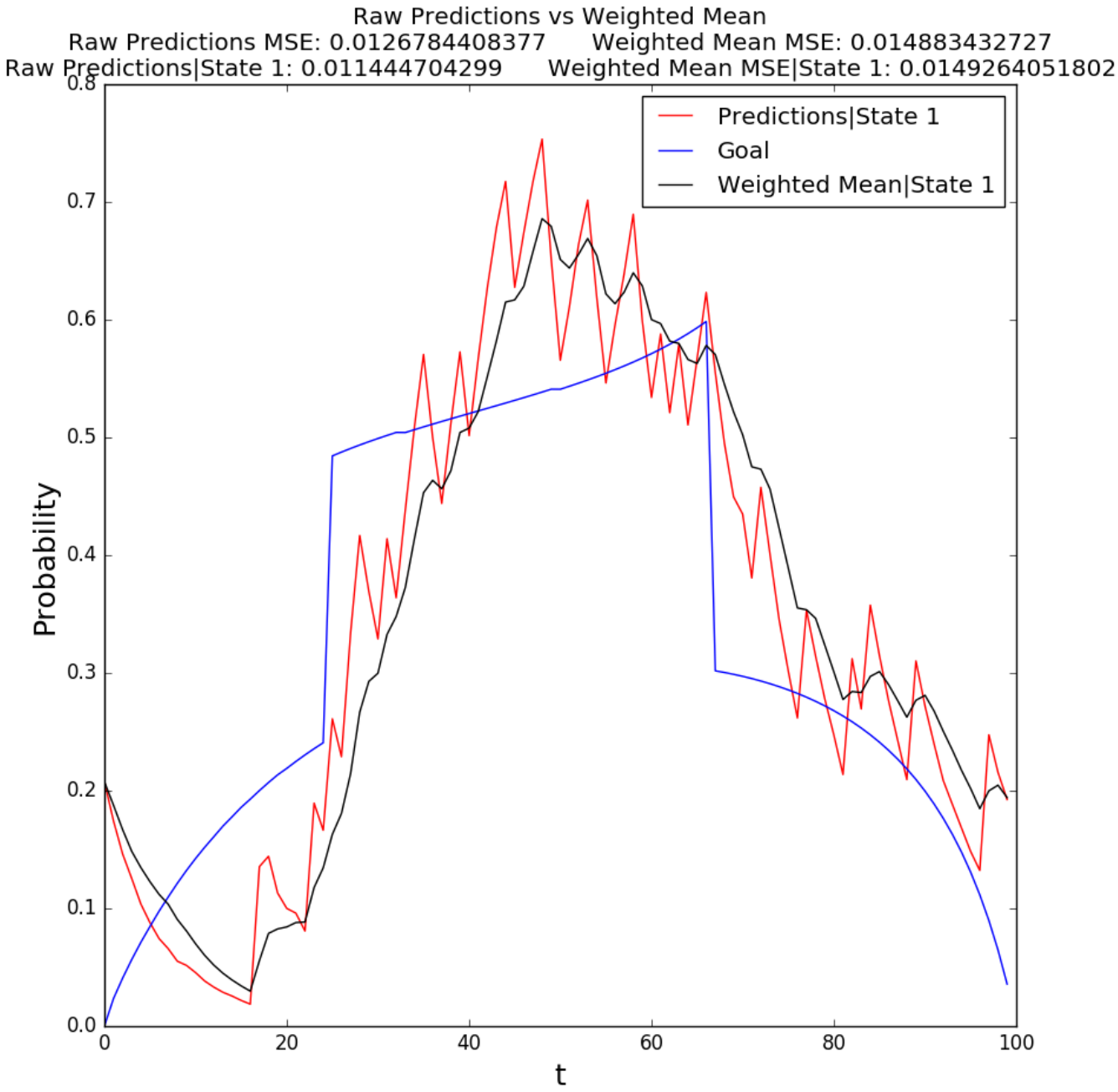

Figure 25: Second Method - Question 5 - State 1
Predictions vs Weighted Mean
w = 0.90

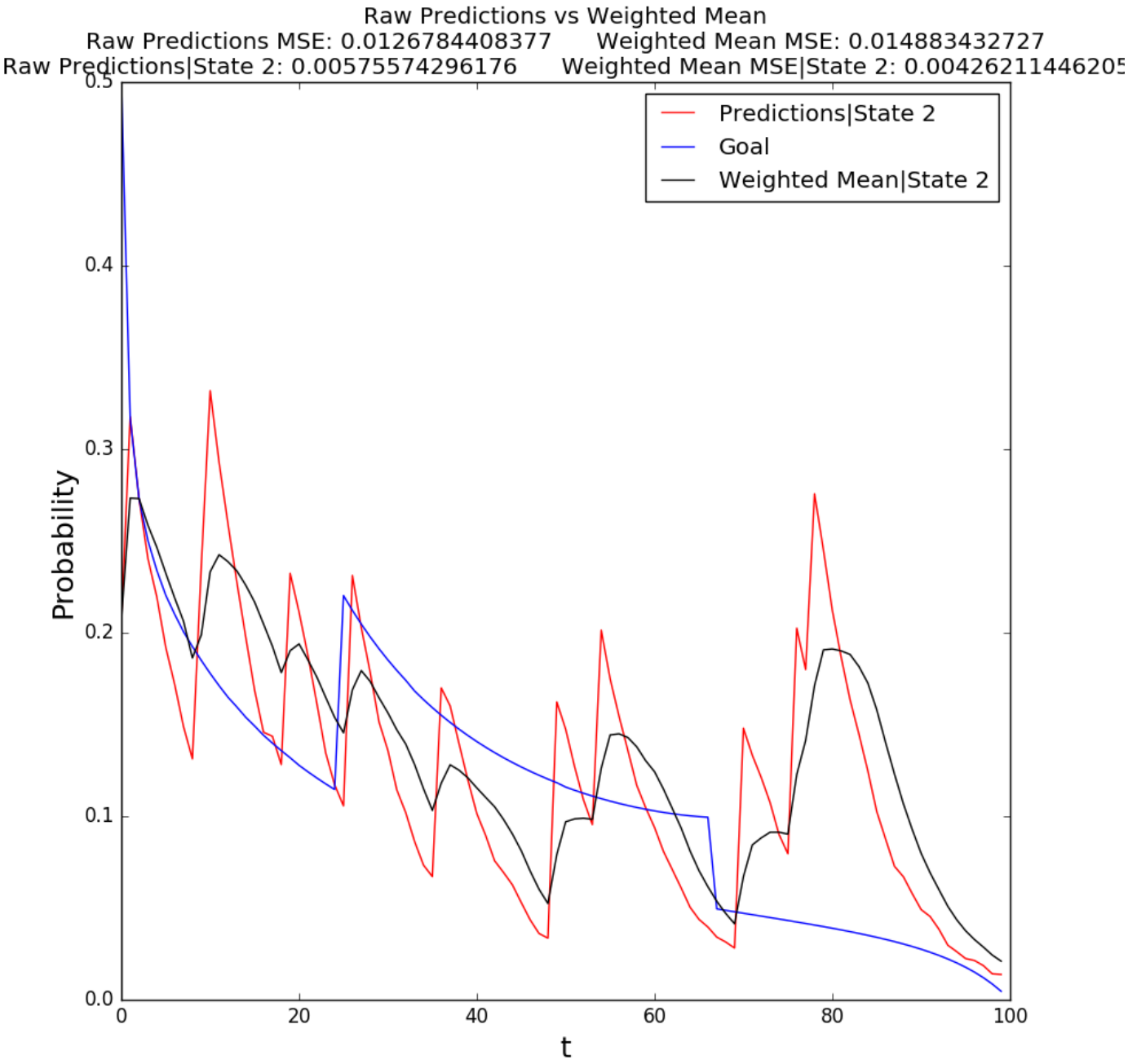

Figure 26: Second Method - Question 5 - State 2
Predictions vs Weighted Mean
w = 0.90

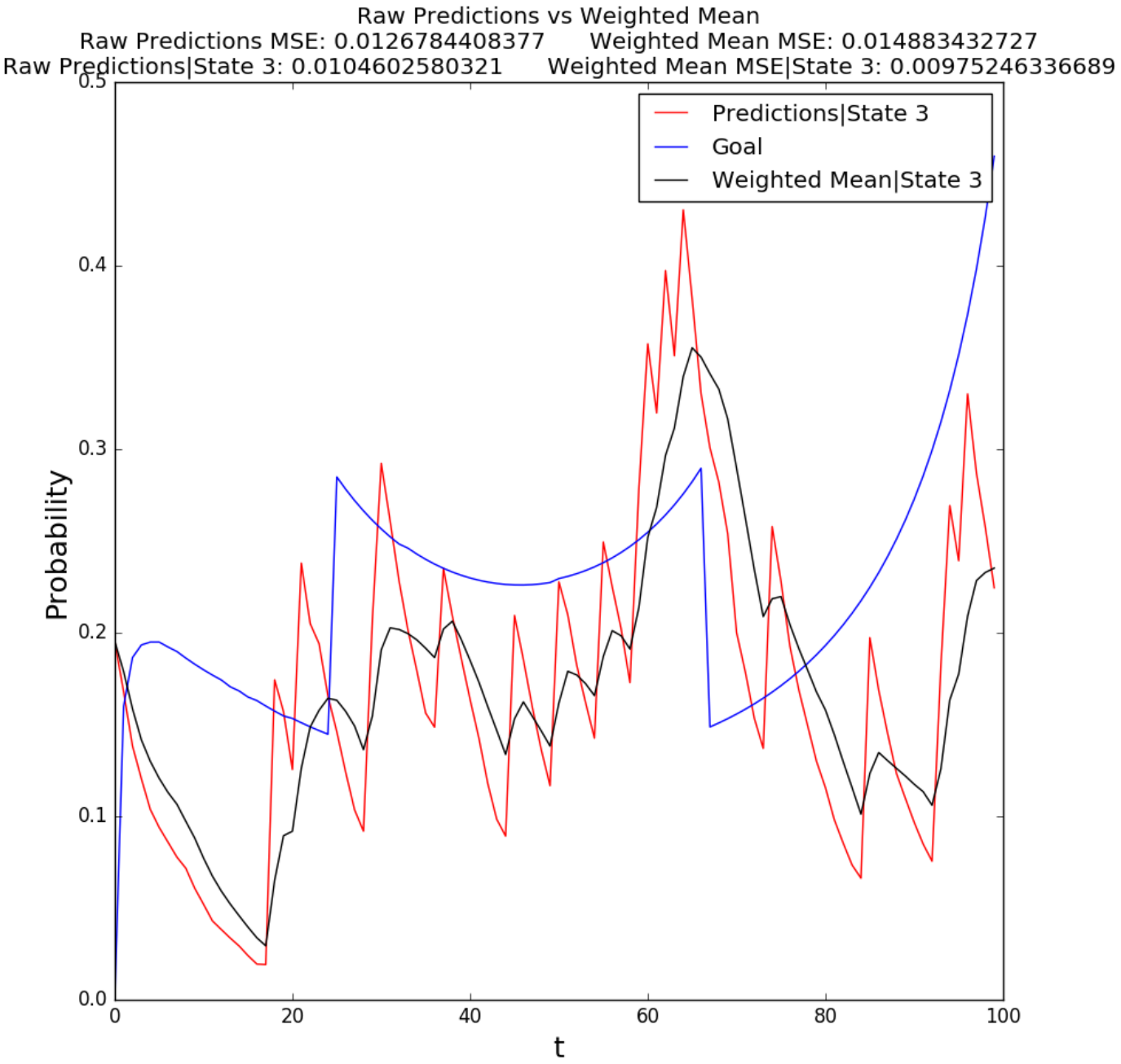

Figure 27: Second Method - Question 5 - State 3
Predictions vs Weighted Mean
w = 0.90

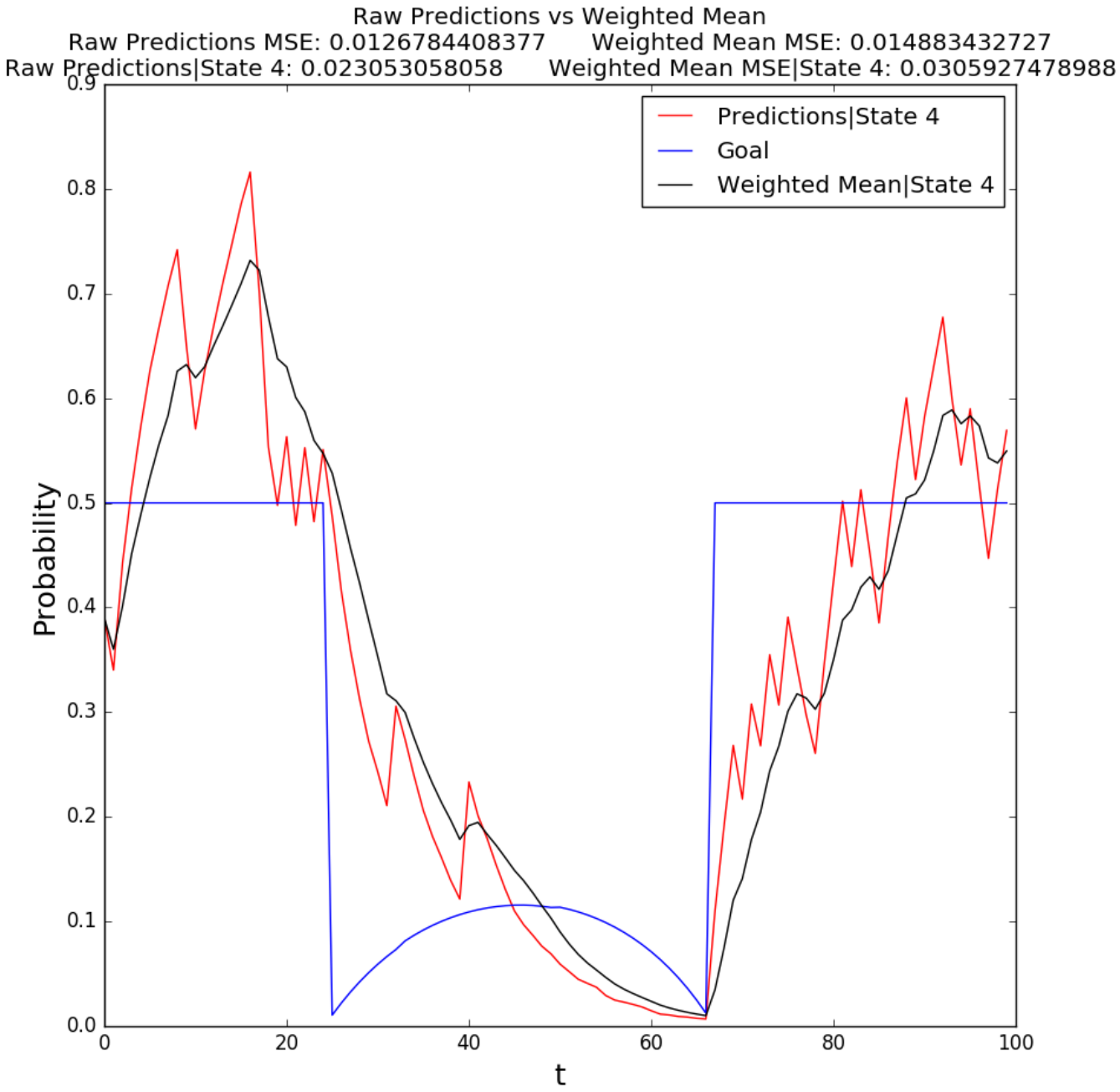

Figure 28: Second Method - Question 5 - State 4
Predictions vs Weighted Mean
w = 0.90

Conclusions and future work

Since all computations for every particle are carried out in a separate fashion, this algorithm could be easily improved upon by the use of multithreading. Minding the steps on which the weights are normalized, which could be a potential bottleneck. Alternative approaches to the computation of the α parameter should be sought, to make the model more responsive to changes (as in approach 1) but also not so prone to outliers (as approach 2). The use of integral methods could be pursued, in order to contrast the results of the particle filter. However, these methods would come in hand with a much higher expenditure of computational resources. It is also necessary to test the concept over real data, as the synthetic experiments only serve as a proof of concept for the viability of the method with the chosen probability distributions.