Online Gaussian Estimation with Long-term Memory

CMPS 290C Project Report

Sai Xiao

June 13, 2013

1 Introduction

We study the long-term memory algorithm for online Gaussian density estimation. We pay much emphasis on the scenario where the best expert is changing over time. In this scenario, the comparator is not a fixed best expert. When the experts only switch among a small set of experts, the comparator is called sparse partition model [KAW12].

In the setting of finite experts, [BW02] develop the mixing post posterior (MPP) algorithm with several mixing schemes. The main feature of this algorithm is the capability of long-term memory. The "posterior" weight of each expert mixes with its own past posterior in each update so that its weight can be recovered quickly after it becomes best expert again. [KAW12] interprets the MPP algorithm in a Bayesian way by introducing the asleep/awake configuration for the expert specialists. We give a brief review of MPP algorithm as follows,

FOR $t = 1$ TO T DO

- Make prediction $\hat{y}_t$ by $p(\hat{y}_t) = w_t \cdot y_t$
- Loss update: $w_{t,i}^m = \frac{w_{t,i} e^{L_{t,i}}}{normalization}$
- Mixing update: $w_{t+1} = (1-\alpha) w_{t,i}^m + \sum_{q=0}^{t-1} \beta_t(q) w_q^m, \quad \sum_{q=0}^{t-1} \beta_t(q) = \alpha$

The mixing strategy is shown in Figure 1. In the context of finite experts, the posterior of mean of Gaussian is actually a discrete distribution with point masses at the value of experts. In the problem

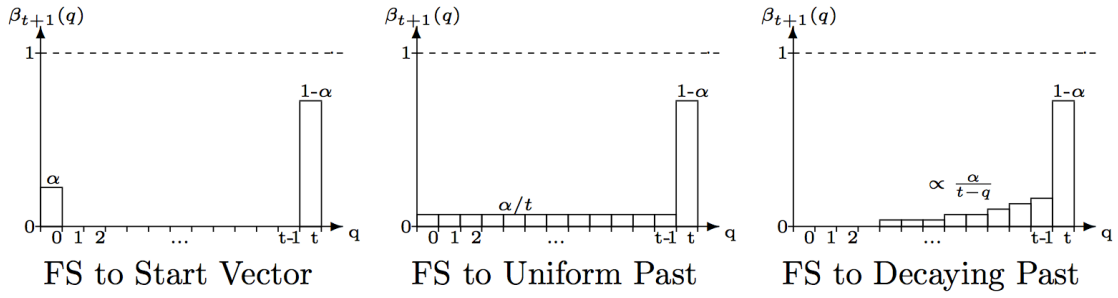

Figure 1: Mixing strategy

of learning the mixture of Gaussian from on-line data, all the grid points are assigned as experts (the mean of Gaussian) by assuming the standard deviation is known and fixed. The drawback of this algorithm is that in the high-dimensional density estimation, the number of grid points grows as an exponential function of the number of dimensions.

My main task in the project is to generalize the MPP algorithm to the setting of "infinite" experts. Instead of preassigning fixed values for a finite number of experts, we will give a distribution for the mean of Gaussian μ. We propose the algorithm to estimate the posterior of μ by Bayesian updates. As a result, the time complexity grows as a polynomial function of size of dimensions. We also study and improve its performance under different mixing strategies in the following sections.

2 Algorithm

We mimic the MPP idea to update the distribution of μ by multiplying its data likelihood and mix with its past posterior distributions. We have the simple "infinite expert" version of MPP algorithm to estimate μ as follows,

Let $p_0(\mu) \sim N(\mu_0, \sigma_0^2)$, a conjugate prior for mean of Gaussian.
FOR $t = 1$ TO T DO

- Make prediction $\hat{y}_t$ by $p(\hat{y}_t|y_{1:(t-1)}) \propto \int p(\hat{y}_t|\mu) p_{t-1}(\mu|y_{1:(t-1)}) d\mu$
- Get Loss update from multiplying the likelihood and obtain the intermediate posterior distribution: $\tilde{p}_t(\mu|y_{1:t}) \propto p_{t-1}(\mu|y_{1:(t-1)}) p(y_t|\mu)$
- Mixing update: the final posterior $p_t(\mu|y_{1:t})$ is a linear combination of $\tilde{p}_t(\mu|y_{1:t})$ and past intermediate posteriors $\tilde{p}_q(\mu|y_{1:q})$, $q = 0, \cdots, t-1$.

$$p_t(\mu|y_{1:t}) = (1-\alpha)\tilde{p}_t(\mu|1:t) + \sum_{q=0}^{t-1} \beta_t(q)\tilde{p}_q(\mu|y_{1:q}), \quad \sum_{q=0}^{t-1} \beta_t(q) = \alpha$$

In the loss update, we claim that if $p_{t-1}(\mu|y_{1:(t-1)})$ is a mixture of Gaussian distribution and $p(y_t|\mu)$ is a Gaussian distribution. then $\tilde{p}_t(\mu|y_{1:t})$ is also a mixture of Gaussian distribution and both the kernel and weights in the mixture are changed.

Proof. Without loss of generality, assume that $p_{t-1}(\mu|y_{1:(t-1)})$ is a mixture of two univariate Gaussians such that $p_{t-1}(\mu|y_{1:(t-1)}) = wN(\mu|a, \tau_1^2) + (1-w)N(\mu|b, \tau_2^2)$, where $0 < w < 1$.

$$\begin{aligned}
\tilde{p}_t(\mu|y_{1:t}) &= \left[wN(\mu|a, \tau_1^2) + (1-w)N(\mu|b, \tau_2^2)\right] N(y_t|\mu, \sigma^2) \\
&= \frac{wq_1h_1 + (1-w)q_2h_2}{wq_1 + (1-w)q_2} \\
&= \frac{wq_1}{wq_1 + (1-w)q_2}h_1 + \frac{(1-w)q_2}{wq_1 + (1-w)q_2}h_2
\end{aligned}$$

q_1 and q_2 are normalizing constants and h_1 and h_2 are the new mixture components.

$$\begin{aligned}
q_1 &= \int N(\mu|a, \tau_1^2)N(y_t|\mu, \sigma^2)du = N(y_t|a, \tau_1^2 + \sigma^2), \\
q_2 &= \int N(\mu|b, \tau_2^2)N(y_t|\mu, \sigma^2)du = N(y_t|b, \tau_2^2 + \sigma^2). \\
h_1 &\propto N(\mu|a, \tau_1^2)N(y_t|\mu, \sigma^2) = N(\mu|\frac{a\sigma^2 + \tau_1^2 y_t}{\sigma^2 + \tau_1^2}, \frac{\sigma^2\tau_1^2}{\sigma^2 + \tau_1^2}), \\
h_2 &\propto N(\mu|b, \tau_2^2)N(y_t|\mu, \sigma^2) = N(\mu|\frac{b\sigma^2 + \tau_2^2 y_t}{\sigma^2 + \tau_2^2}, \frac{\sigma^2\tau_2^2}{\sigma^2 + \tau_2^2})
\end{aligned}$$

□

From the claim, we can conclude that in loss update step, a new set of Gaussian mixtures are created. In the mixing update, because every $\tilde{p}_i(\mu|1:i),\ i=1,\cdots,t$ is a mixture of Gaussians, $p_t(\mu|y_{1:t})$ is certainly a mixture of Gaussians. Each $\tilde{p}_i(\mu|1:i),\ i=1,\cdots,t$ does not share any same mixture components. So, we conclude that the more past posterior are incorporated in mixing update for $p_{t-1}(\mu|y_{1:(t-1)})$, the more number of new Gaussian mixtures will be created.

3 The complexity of the algorithm

Let n be the number of dimensions. The time complexity in calculating each mixture component is at most $O(n^3)$ because of the matrix multiplication, so this algorithm works in high dimensional Gaussian density estimation. We will focus on the number of distinct Gaussian mixtures because we find it significantly influences the time and space complexity of the algorithm. For Static Experts ($\alpha = 0$) without share, the posterior of μ is always a univariate Gaussian distribution. For fixed share (FS) to start vector, the number of mixture components is $t+1$ in trial t. For $p_t(\mu|y_{1:t})$, it has the form of $\sum_{i=1}^{t} w_i p(\mu|y_{i:t}) + (1-\sum_{i=1}^{t} w_i)p_0(\mu)$. So the final posterior of μ only contains posterior of μ given a subset of combination of the data.

For FS to past Uniform and decaying past, the number of mixture components is 2^t in trial t. We elaborate it by the plot as below. In each line, we show which subset of data decides the mixture component in $\tilde{p}_t(\mu|y_{1:t})$ and the numbers in the plot are the indices of the data. The colored index indicates the new data in trial t and the black indices indicate the subset of data associated with the mixture components in $p_{t-1}(\mu|y_{1:(t-1)})$. For example, in trial $t=1$, $\tilde{p}_1(\mu|y_1)$ only has one component. In trial $t=2$, $\tilde{p}_2(\mu|y_{1:2})$ has two mixture component $p(\mu|y_2)$ and $p(\mu|y_1,y_2)$. The final posterior $p_2(\mu|y_{1:2})$ contains four mixture components prior $p_0(\mu)$, $p(\mu|y_1)$, $p(\mu|y_2)$ and $p(\mu|y_1,y_2)$. In the trial $t=3$, the intermediate posterior $\tilde{p}_3(\mu|y_{1:3})$ updates $p_2(\mu|y_{1:2})$ with the new data y_3 and creates four new mixture components $p(\mu|y_3)$, $p(\mu|y_1,y_3)$, $p(\mu|y_2,y_3)$ and $p(\mu|y_1,y_2,y_3)$. The final posterior $p_3(\mu|y_{1:3})$ contains eight mixture components because it contains all past posterior distributions and μ's prior. In the same token, in trial $t=4$, eight new mixture components are created after loss update and the final posterior includes the prior and all fifteen components generated through trial 1 to trial 4. So in every trial, the number of mixture components doubles. The total number of mixture components is 2^t in trial t.

$\tilde{p}_t(\mu \mid y_{1:t})$ in the trial t								
$t=1:$	1							
$t=2:$	2	12						
$t=3:$	3	13	23	123				
$t=4:$	4	14	24	124	34	134	234	1234

In this case, the posterior of μ contains its posterior given all combinations of the data from $y_1, \cdots, y_t$. From this point of view, this explains why the FS to past Uniform and decaying past is most powerful algorithm in terms of the long-term memory. However, it is not an efficient algorithm because the number of mixture components grows exponentially with the trial t.

Empirical results We do this empirical results on a simulated data set to compare the performance of FS to start vector and FS to past Uniform. Since the FS to past Uniform is quite inefficient, we do this experiment in a small data set of only 20 data. The data set include two clusters $N((2,2),0.1I_2)$ and $N((-2,-2),0.1I_2)$. The sequence of data constitute five data alternatively generated from cluster 1, 2, 1, 2. The loss is negative log predictive density of the new data given past data $-\log p(y_t|y_{1:(t-1)})$. In Figure 2, it shows that FS to past Uniform outperforms FS to start vector. Both algorithms have bumps at the beginning of each section. However, during the section, FS to past Uniform has much lower loss rate than FS to start vector. Through our experimentation, we found that the performance is significantly effected by the prior $p_0(\mu)$ because the size of data is very small in each section. In this experiment, we intentionally set the prior's mean at $(0,0)$ which is in the center of data.

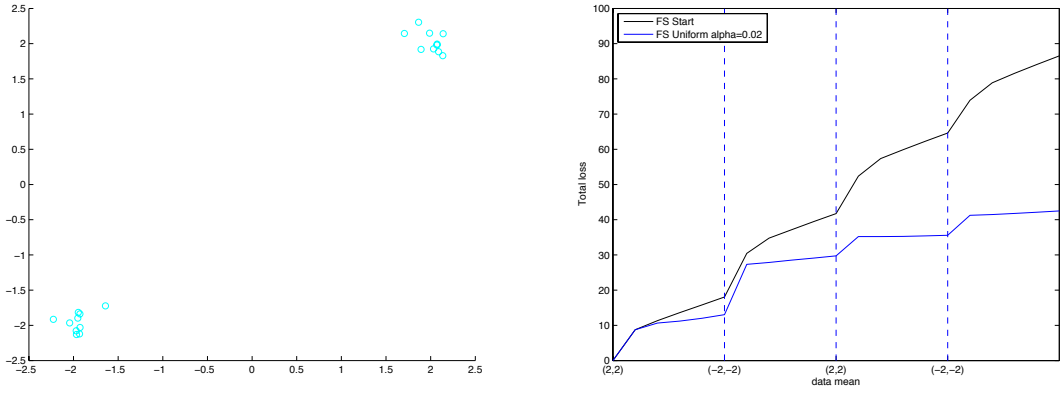

Figure 2: Simulated 20 data.

4 Trick of reducing the number of mixture components

It is awkward that the FS to past Uniform strategy performs better but is unrealistic in practice because of its exponential time complexity. In this section, we try to reduce the number of mixture components by not sacrificing much performance. As is analyzed in section 3, the posterior distribution $p_t(\mu|y_{1:t})$ contains 2^t mixture components depending on all combinations of $y_1, \cdots, y_t$. The main idea is that we will only store the useful posteriors among the 2^t posteriors. We propose one method and it has time and space complexity of $O(t)$.

We name this method "FS past Uniform Reduce": in each trial t, we only keep two mixture components in $\tilde{p}_t(\mu|y_{1:t})$. One is the posterior $p(\mu|y_t)$. Beside, we choose the posterior which gives highest predictive density for the new data and do loss update. For example, in trial $t = 3$, we will compare $p(y_3|y_1), p(y_3|y_2)$ and $p(y_3|y_1, y_2)$. For example, if $p(y_3|y_1, y_2)$ has the highest value, we will only keep $p(\mu|y_1, y_2, y_3)$. So $\tilde{p}_3(\mu|y_{1:3})$ is a mixture of $p(\mu|y_3)$ and $p(\mu|y_1, y_2, y_3)$ with normalized weights.

We firstly apply the trick to the previous simulated data set, We can see that performance worsens a little bit but still beats FS to start vector. In trial 20, FS to past Uniform Reduce and FS to past Uniform even have very similar total loss.

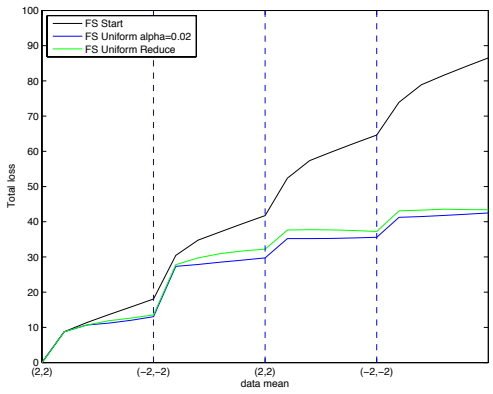

Figure 3: Simulated 20 data. Compare with FS to start vector, FS to past Uniform and FS to past Uniform after applying the trick.

Then we run an experiment with 1000 simulated data. We run 100 trials in each section and data in each section shift among 6 clusters which are centered around (2,2) , (0,0), (-1,-1), (-2, 1), (2,-1), (0, 2). The shifting order is 1, 2, 1, 3, 1, 4, 1, 5. In Figure 4, it shows the FS to past Uniform Reduce performs much better than FS to start vector. The MATLAB program of FS to

past Uniform Reduce is attached with this report.

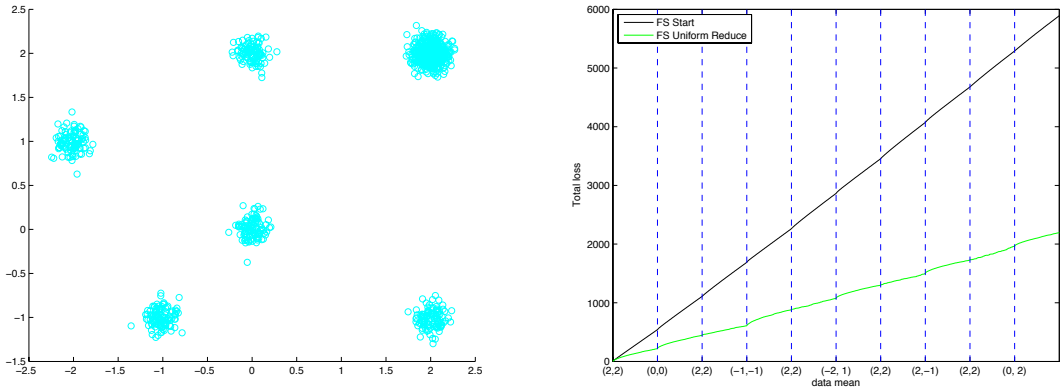

Figure 4: Simulated 1000 data.

References

[BW02] Olivier Bousquet and Manfred K. Warmuth. Tracking a small set of experts by mixing past posteriors. Journal of Machine Learning Research 3 (2002) 363-396.

[KAW12] Wouter M. Koolen, Dmitry Adamskiy, Manfred K. Warmuth: Putting Bayes to sleep. NIPS 2012: 135-143.