**MIT 9.014: Quantitative Methods and Computational Models** **Fall 2017**

# Notes for Problem Set 5

*Daniel Zysman*

## Introduction

In this notes, we will review:

- the equations needed to solve PSET 5
- how to implement them in MATLAB
- how to use `fminsearch` to find optimal parameters for our model.

## The Model

In the experiment described in Jazayeri and Shadlen (2010), the subject is shown a stimulus of duration $t_s$, and makes a noisy measurement of this duration $t_m$. This measurement is combined with prior information to make an estimate $t_e$ of the stimulus duration. Finally, the subject is asked to reproduce $t_s$ based on this internal estimate. The reproduction is given by $t_p$. The sequence of steps is illustrated below.

$$t_s \rightarrow t_m \rightarrow t_e \rightarrow t_p$$

For simplicity, we are going to omit the production stage, thus our model will be concerned with this sequence:

$$t_s \rightarrow t_m \rightarrow t_e$$

In reality we do not have access to $t_e$ as this is merely an internal representation of the subject. However, this values have been given to us as they come from a synthetic experiment, where data was generated via numerical simulation. In the next section we go over the components of the model.

### Prior distribution of $t_s$

The $t_s$'s come from a **discrete uniform distribution**. We will call this prior distribution $p(t_s)$. To simplify derivations, we will model the discrete uniform prior as continuous. For each prior condition (short and long), we specify the domain of the sample intervals between $t_s^{\min}$ and $t_s^{\max}$ on the basis of the minimum and maximum values used in the experiments. In this way, $p(t_s)$ is given by:

$$p(t_s) = \begin{cases} \frac{1}{t_s^{\max} - t_s^{\min}} & \text{for } t_s^{\min} \leq t_s \leq t_s^{\max} \\ 0 & \text{otherwise} \end{cases}$$

## Likelihood

The noise distribution associated with the measurement stage determines the distribution of $t_m$ for a given $t_s$. This is captured by the conditional probability $p(t_m|t_s)$. We model this conditional probability as a Gaussian distribution with mean $t_s$ and a fixed standard deviation $\sigma_m$:

$$p(t_m|t_s;\sigma_m) \quad = \quad \frac{1}{\sqrt{2\pi}\sigma_m} e^{-\frac{(t_m-t_s)^2}{2\sigma_m^2}}$$

From the perspective of the observer who makes an measurement $t_m$ but does not known $t_s$, this relationship becomes a function of $t_s$ that is known as the likelihood function $\mathcal{L}(t_s)$. From the perspective of the modeler, this relationship becomes a function of the only unknown, which is the only parameter in our model so far $\sigma_m$. This latter version is also a likelihood function $\mathcal{L}(\sigma_m)$. In short:

$$\mathcal{L}(\sigma_m) \equiv \mathcal{L}(t_s) \equiv p(t_m|t_s) = \frac{1}{\sqrt{2\pi}\sigma_m} e^{-\frac{(t_m-t_s)^2}{2\sigma_m^2}}$$

## Posterior

By means of the Bayes' rule the posterior distribution is given by

$$p(t_s|t_m) = \frac{p(t_m|t_s)p(t_s)}{p(t_m)}$$

Furthermore, $p(t_m)$ can be obtained by marginalizing the joint distribution: $p(t_m) = \int p(t_m, t_s)dt_s$. Next, we apply Bayes' rule one more time to write joint distribution as: $p(t_m, t_s) = p(t_m|t_s)p(t_s)$. Thus, the marginal distribution of $p(t_m)$ is:

$$p(t_m) = \int_{t_s^{\min}}^{t_s^{\max}} p(t_m|t_s)p(t_s)dt_s$$

Using this formulation the posterior distribution becomes:

$$p(t_s|t_m) = \frac{p(t_m|t_s)p(t_s)}{\int_{t_s^{\min}}^{t_s^{\max}} p(t_m|t_s)p(t_s)dt_s}$$

Since the prior distribution is uniform and therefore has a constant value for all $t_s$'s it can be pulled out of the integral sign. In this way the posterior distribution simplifies to:

$$p(t_s|t_m) = \frac{p(t_m|t_s)}{\int_{t_s^{\min}}^{t_s^{\max}} p(t_m|t_s)dt_s}$$

## Bayesian Least Squares (BLS)

So far we have a full distribution (the posterior) but we would like to get a point estimate, $t_e$, for the shown $t_s$, that is optimal under some specified criterion. One way to produce this estimate, is to combine the posterior distribution with a cost function. The optimal $t_e$ will be given by the value that minimizes the specifed cost. The Bayesian Least Squares (BLS) cost function is defined as $l(t_e, t_s) = (t_e - t_s)^2$. Under BLS, the optimal $t_e$ is prescribed by:

$$t_e \;=\; f_{BLS}(t_m) = \arg\min_{t_e} \int_{t_s^{\min}}^{t_s^{\max}} p(t_s|t_m)(t_e - t_s)^2 dt_s$$

At this point I want to emphasize three important things:

1. $t_e$ is a function of $t_m$ as we are minimizing with respect of $t_e$ and integrating over $t_s$.
2. $t_m$ is given by the likelihood function and in this case depends on only one parameter: $\sigma_m$.
3. $f_{BLS}(t_m)$ is a *deterministic* function. For a given $t_m$ and $\sigma_m$ it will always return the same $t_e$.

### BLS returns the mean of the posterior

In the following I want to show you that the $t_e$ returned under the BLS cost function is the expected value of the posterior distribution. To do so, let's work on the integral by recalling that $(t_e - t_s)^2 = t_e^2 + t_s^2 - 2t_e t_s$. Then:

$$\begin{aligned}\int_{t_s^{\min}}^{t_s^{\max}} p(t_s|t_m)(t_e - t_s)^2 dt_s &= \int_{t_s^{\min}}^{t_s^{\max}} p(t_s|t_m)(t_e^2 + t_s^2 - 2t_e t_s) dt_s \\ &= \int t_e^2 p(t_s|t_m) dt_s + \int t_s^2 p(t_s|t_m) dt_s \\ &- 2\int t_e t_s p(t_s|t_m) dt_s\end{aligned}$$

Since we are integrating over $t_s$, $t_e$ is a constant under integration and can be pulled out of the integral. Thus:

$$\begin{aligned}\int_{t_s^{\min}}^{t_s^{\max}} p(t_s|t_m)(t_e - t_s)^2 dt_s &= t_e^2 \int p(t_s|t_m) dt_s + \int t_s^2 p(t_s|t_m) dt_s \\ &- 2t_e \int t_s p(t_s|t_m) dt_s\end{aligned}$$

Let's work on each term of the RHS. The first integral is equal to one, as we are integrating a conditional probability over all possible values ($t_s$). The second integral turns out to be equal to $\mathbb{E}(p(t_s^2|tm))$ [1]. The third integral is just $\mathbb{E}(p(t_s|t_m))$, which is the mean of the posterior distribution. After these steps we reach to:

[1]by means of the law of the unconscious statistician.

$$\int_{t_s^{\min}}^{t_s^{\max}} p(t_s|t_m)(t_e - t_s)^2 dt_s \quad = \quad t_e^2 + \mathbb{E}(p(t_s^2|tm)) - 2t_e\mathbb{E}(p(t_s|t_m))$$

The objective of the Bayesian Least Squares estimator is to find the $t_e$ that minimizes the above expression. To find $t_e$, we simply take the partial derivative of the RHS with respect to $t_e$ and set it to 0. Then, we solve for $t_e$.

Let's compute the derivative and solve for $t_e$:

$$\begin{aligned} \frac{\partial}{\partial t_e} &= 2t_e - 2\mathbb{E}(p(t_s|t_m)) = 0 \\ t_e &= \mathbb{E}(p(t_s|t_m)) \\ f_{BLS}(t_m) &= \mathbb{E}(p(t_s|t_m)) \end{aligned}$$

This last step shows that the BLS estimate is the expected value of the posterior distribution.

### A way to compute BLS estimates

The expected value of the posterior is given by:

$$\mathbb{E}(p(t_s|t_m)) = \int_{t_s^{\min}}^{t_s^{\max}} t_s p(t_s|t_m) dt_s$$

In our setup the posterior is only accessible to us as a function of the likelihood:

$$p(t_s|t_m) = \frac{p(t_m|t_s)}{\int_{t_s^{\min}}^{t_s^{\max}} p(t_m|t_s) dt_s}$$

To compute the expected value, we integrate (apply the definition of expected value) on both sides of the equality :

$$\mathbb{E}(p(t_s|t_m)) = \int_{t_s^{\min}}^{t_s^{\max}} t_s p(t_s|t_m) dt_s = \int_{t_s^{\min}}^{t_s^{\max}} t_s \frac{p(t_m|t_s)}{\int_{t_s^{\min}}^{t_s^{\max}} p(t_m|t_s) dt_s} dt_s$$

Notice that the term $\int p(t_m|t_s)dt_s$ is a constant under the outer integral, as we have already integrated $t_s$ out. Thus, the BLS estimate is given by:

$$f_{BLS}(t_m) = \mathbb{E}(p(t_s|t_m)) = \frac{\int_{t_s^{\min}}^{t_s^{\max}} t_s p(t_m|t_s) dt_s}{\int_{t_s^{\min}}^{t_s^{\max}} p(t_m|t_s) dt_s}$$

## $t_m$'s are needed to fit $\sigma_m$

The model described in the previous section depends on a single parameter, $\sigma_m$. However, $\sigma_m$ enters to the model via the likelihood function. To compute the likelihood and estimate

$\sigma_m$, we need the $t_m$'s for each trial, which we don't have. So, how, can we solve this puzzle then?

Let's pretend for a second that $\sigma_m$ is initialized at a given value $\sigma_m^\dagger$ (like when we start an fminsearch). With this initial value for sigma an a set of putative $t_m$ values that we will call $t_m^\dagger$ we can compute $f_{BLS}$ for each value in $t_m^\dagger$. This will give us a look-up table[2]:

| $t_m^\dagger$ | $f_{BLS}(t_m^\dagger, \sigma_m^\dagger)$ |
|---|---|
| 1 | 2 |
| 4 | 5 |
| 7 | 8 |

Let's recall that $f_{BLS}$ is an estimate of $t_e$ for a given trial. For simplicity, let's call the estimate $\hat{t}_e$. Furthermore, the $t_e$ for each trial are supplied to us. Let's add to the table the value of $t_e$ for a specific trial, so that we can compare the estimates and actual values hand by hand:

| $t_m^\dagger$ | $\hat{t}_e$ | $t_e$ |
|---|---|---|
| 1 | 2 | 4 |
| 4 | 5 | 4 |
| 7 | 8 | 4 |

We want to find the $\hat{t}_e$ that is closest to the actual $t_e$. The criterion is to look at $|(\hat{t}_e - t_e)|$ (i.e. the absolute value of the difference) and pick the minimum. Let's add this to our table:

| $t_m^\dagger$ | $\hat{t}_e$ | $t_e$ | $\lvert(\hat{t}_e - t_e)\rvert$ |
|---|---|---|---|
| 1 | 2 | 4 | 2 |
| 4 | 5 | 4 | 1 |
| 7 | 8 | 4 | 4 |

In this case, the minimum is attained on the second row. However, we are interested in the value of $t_m^\dagger$ that produced the $\hat{t}_e$ closest to $t_e$. In essence, our criterion to find $t_m$ is

$$t_m = \arg\min_{t_m^\dagger} |\hat{t}_e - t_e|$$

This procedure works, because the $f_{BLS}$ is an invertible function[3]. That is, there is a one-to-one correspondence between $t_m$ and $t_e$, as illustrated in the figure below. So, now we have a procedure to obtain the

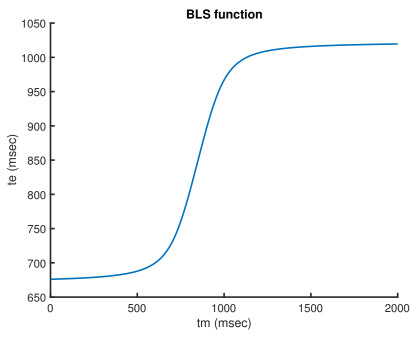
## Computing integrals in MATLAB

To obtain the $t_m$ for each trial we use the $f_{BLS}$ as described in the previous section. Let's recall the formula for this estimate:

$$f_{BLS}(t_m) = \mathbb{E}(p(t_s|t_m)) = \frac{\int_{t_s^{\min}}^{t_s^{\max}} t_s p(t_m|t_s) dt_s}{\int_{t_s^{\min}}^{t_s^{\max}} p(t_m|t_s) dt_s}$$

[2]this is fake data to illustrate the principle

[3]Keep in mind that there is no analytical formula for the inverse and thus we need to construct a look-up table as a proxy for the inverse



**BLS function**

This function requires the computation of two integrals numerically. We will go over how to implement this in MATLAB. First, the integrals require computing $p(t_m|t_s)$. So, the first step is to write a MATLAB function that computes $p(t_m|t_s)$ Open the file **Ptmts.m** and complete it to compute the likelihood. The file is reproduced here for clarity.

```
function p = Ptmts(ts, tm, sigma)
% Computes Likelihood function
%
% Inputs:
% ts: ts values, column vector
% tm: tm values, column vecotr
% sigma: standard deviation of likelihood
%
%Output:
% p: likelihood value, column vector:
% p returns these values:
% $\frac{1}{\sqrt{2\pi}\sigma_m} e^{-\frac{(t_m - t_s)^2}{2\sigma_m^2}}$


end
```

Now that we have the function we want to integrate, we can tackle integration in MATLAB. Let's say we want to compute the integral of:

$$\int_0^\pi \sin(x)dx$$

In MATLAB we compute this integral with command **integral**, which has the following syntax:

```
integral(@(x)sin(x), 0, pi)
```

The first inupt argument is the function we want to integrate with a handle (@ sign) to the variable we are integrating over. As in the **fminsearch** case the function is evaluated at different values of $x$ in this case. The last two input arguments are the lower and upper limits for integration.

In some instances we want to compute an integral of an array valued function. For example we might want to compute in a single shot the integral of $\sin(x), \sin(2x), \sin(3x)$. The `integral` function allows this type of input, the syntax is:

```
integral(@(x)sin((1:3)*x), 0, pi, 'ArrayValued', true)
```

In this case the output will be a 3 by 1 vector with the integral of $\sin(x), \sin(2x), \sin(3x)$ in each component. This feature will allow us to compute $t_e$ via $f_{BLS}$ for a vector of $t_m$ values without calling a for loop.

With this information complete the `BLS_est.m` file, such that it computes the BLS estimator. The file is reproduced here for clarity:

```
function te = BLS_est( ts, tm, sigma)
% Computes te by means of the BLS. Which is equivalent to the mean of the
% posterior distribution.

% inputs
% ts: column vector
% tm: column vector
% sigma standard deviation of likelihood
%
% Output:
% te: column vector


end
```

Finally complete the function `inv_BLS_est.m` that gets $t_m$ for each trial by empirically inverting the BLS. This function should call `BLS.est.m` and apply the look-up table approach we have previously discussed. Again, the function is reproduced here.

```
function tm = inv_BLS_est(ts,tm0, te, sigma)
% Finds the tm that generates the te closest to the observed value. It does
% so by empirically inverting the BLS function
%
%  inputs
% ts: column vector
% tm0: column vector
% sigma standard deviation of likelihood
% te: column vector
%
% Output:
% tm: column vector

end
```

## Running fminsearch to estimate $\sigma_m$

With the functions we wrote in the preceding section we are ready to estimate $\sigma_m$ from the supplied data. For this, we will rely one more time on a Maximum Likelihood Estimate (MLE). For this purpose, we will use `fminsearch` as in PSET 4.

Below is a copy of `MainScript.m`, which loads the data, generates a vector of $t_m^dagger$ to look over when inverting the BLS function., and the calls `fminsearch`. Notice again: we are searching for a single paramter.

```
% Load data
load('short_prior.mat');
ts_short = ts;
te_short = te;

clearvars te ts;
% Set range of tm's to invert BLS estimate
tm0 = (0:0.5:2000)';

% Options for fminsearch
opts = optimset('fminsearch');
opts.Display = 'final';

% Initialize sigma
sigma_init = 200;

sigma_short = fminsearch(@(s) NegLogLike(ts_short, tm0, te_short, s), ...
    sigma_init, opts);
```

At each iteration, `fminsearch` calls function `NegLogLike` that executes these steps:

1. Computes $t_m$ by inverting the BLS for the current $\sigma_m$ estimate.
2. Computes the likelihood
3. It returns the negative log likelihood.

This function is provided to you and is reproduced to you.

```
function NeglogLike = NegLogLike(  ts, tm0, te, sigma)
% compute neg log likelihood
%
% Inputs
% ts: ts per trial; Column Vector
% tm0: range of tm to invert BLS; Column vector
% te: observed te; column vector
%sigma: standard deviation of likelihood
% Output:
%
% Negative Log Likelihood ; scalar

% Find best tm from inverting BLS
tm = inv_BLS_est(ts, tm0, te, sigma);
% Compute Likelihood
p = Ptmts(ts, tm, sigma);
% Take negative log
NeglogLike = -sum(log(p));
end
```

## Appendix: Useful functions for statistics on grouped data

You will find functions `findgroups` and `splitapply` handy to compute mean and standard deviations for grouped data (in this case for different values of $t_s$). To plot error bars, the function `errorbar` will do the job.