# Motion Segmentation using EM - a short tutorial

Yair Weiss
MIT E10-120, Cambridge, MA 02139, USA
yweiss@psyche.mit.edu

The expectation-maximization algorithm (EM) is used for many estimation problems in statistics. Here we give a short tutorial on how to program a segmentation algorithm using EM. Those interested in the theory or in more advanced versions of the algorithm should consult the references at the end.

Suppose we are given a set of datapoints that were generated by multiple processes, for example two lines. We need to estimate two things: (1) the parameters (slope and intercept) of the two lines and (2) the assignment of each datapoint to the process that generated it. The intuition behind EM is that each of these steps is easy assuming the other one is solved. That is, assuming we know the assignment of each datapoint, then we can estimate the parameters of each line by taking into consideration only those points assigned to it. Likewise, if we know the parameters of the lines we can assign each point to the line that fits it best.

This gives the basic structure of an EM algorithm:

- start with random parameter values for the two models.
- Iterate until parameter values converge:
  - E step: assign points to the model that fits it best.
  - M step: update the parameters of the models using only points assigned to it.

In fact both steps are slightly more complicated, due to the assignment being continuous rather than binary valued. The following sections go into more detail regarding the E step and the M step.

## 1 The Expectation (E) step

In the E step we compute for each datapoint two weights $w_1(i), w_2(i)$ (the soft assignment of the point to models 1 and 2 respectively[1]). Again, we assume that the parameters of the processes are known. Thus for each datapoint we can calculate two residuals $r_1(i), r_2(i)$ - the difference between the observation at point $i$ and the predictions of each model. In the case of line fitting the residual is simply given by:

$$r_1(i) = a_1 x_i + b_1 - y_i \tag{1}$$

[1]Throughout this tutorial we assume the number of models is known and is equal to two. A method for automatically estimating the number of models is presented in (Weiss and Adelson, 1996)

and similarly for $r_2(i)$. For example, suppose we have two line models: (1) $y = x + 3$ and (2) $y = 2x - 1$. Suppose the $i$th datapoint is $x = 1, y = 1.1$. Then the residual for line 1 is $r_1^2(i) = (1 + 3 - 1.1)^2 = 2.9^2$ and for line 2 we get $r_2^2(i) = (2 - 1 - 1.1^2) = 0.1^2$. Intuitively we would expect this datapoint to be assigned to model 2 because the residual is smaller. Indeed the formula for the weights is consistent with the intuition:

$$w_1(i) = \frac{e^{-r_1^2(i)/\sigma^2}}{e^{-r_1^2(i)/\sigma^2} + e^{-r_2^2(i)/\sigma^2}} \tag{2}$$

$$w_2(i) = \frac{e^{-r_2^2(i)/\sigma^2}}{e^{-r_1^2(i)/\sigma^2} + e^{-r_2^2(i)/\sigma^2}} \tag{3}$$

There are a few things to note about this formula. First, note that $w_1(i)$ and $w_2(i)$ sum to one. This is because these weights are actually probabilities - the formula is simply derived from Bayes rule. Second, note that there is a "free parameter" here $\sigma$. Roughly speaking, this parameter corresponds to the amount of residual expected in your data (e.g. the noise level). Finally, note that if $r_1^2(i)$ is much smaller than $r_2^2(i)$ then $w_1(i) = 1$ and $w_2(i) = 0$. For this reason equations 2-3 are sometimes known as the "softmin" equations. They are a way of generalizing the concept of a minimum of two numbers into a smooth function. More details on this can be found in the references.

So to summarize, the E step calculates two weights for every datapoint by first calculating residuals (using the parameters of each model) and then running the residuals through a softmin function as defined above.

# 2 The Maximization (M) step

In the M step we assume the weights are given, i.e. for each datapoint we know $w_1(i)$ and $w_2(i)$. To estimate the parameters of each process we just use weighted least squares. An example of least squares estimation is when we want to fit a line to datapoints. It is easy to show that the parameters of the line $a, b$ are a solution to the following equation:

$$\begin{pmatrix} \sum_i x_i^2 & \sum_i x_i \\ \sum_i x_i & \sum_i 1 \end{pmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum_i x_i y_i \\ \sum_i y_i \end{bmatrix}$$

Now in weighted least squares we are also given $w_i$ for each point and the equation simply become:

$$\begin{pmatrix} \sum_i w_i x_i^2 & \sum_i w_i x_i \\ \sum_i w_i x_i & \sum_i w_i 1 \end{pmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum_i w_i x_i y_i \\ \sum_i w_i y_i \end{bmatrix} \tag{4}$$

So in the M step we solve the above equation twice. First with $w_i = w_1(i)$ for the parameters of line 1 and then with $w_i = w_2(i)$ for the parameters of line 2. In general, in the M step we solve two weighted least squares - one for each model, with the weights given by the results of the E step.

# 3 So what about motion segmentation?

To apply the EM algorithm for motion segmentation we need to make the following decisions:

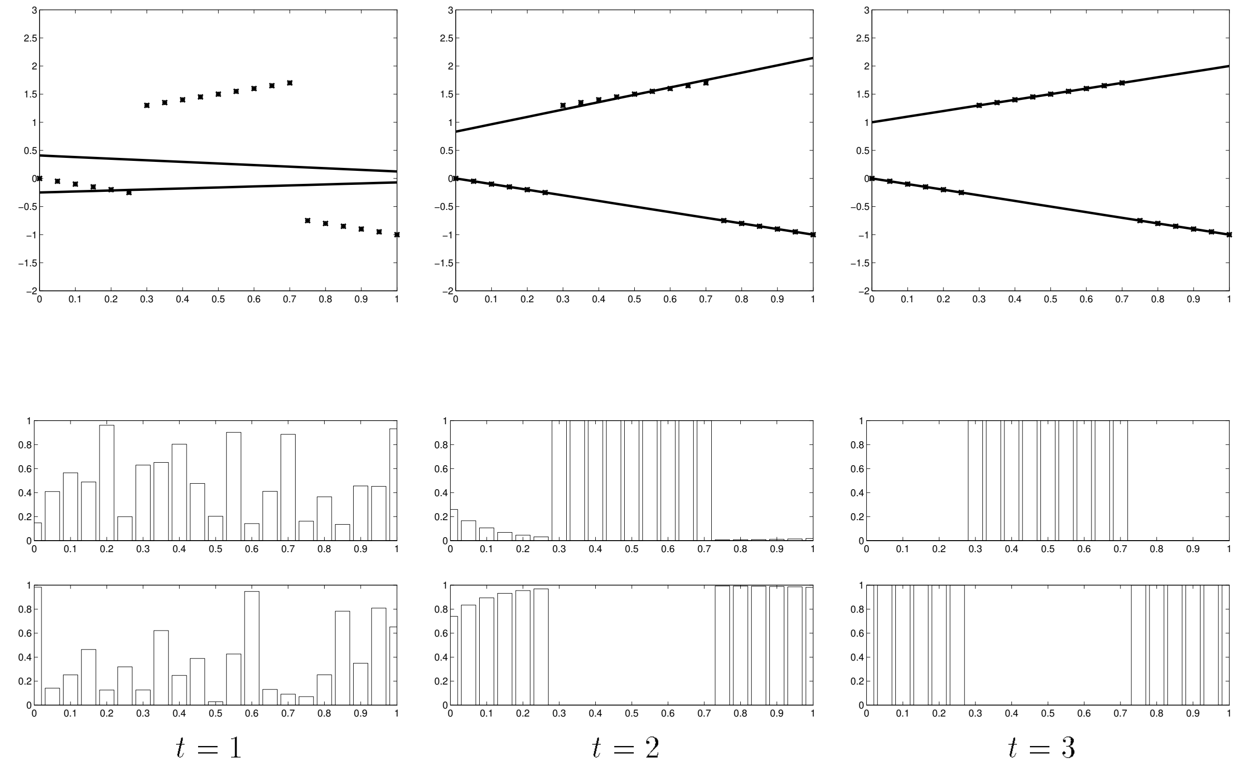


Figure 1: An example of the EM algorithm for fitting two lines to data. The algorithm starts at random conditions and converges in three iterations. The top panels show the line fits at every iteration and the bottom panels show the weights. (data was generated by setting $y = x+1$ for $|x-0.5| < 0.25$ and $y = -x$ otherwise. The parameter $\sigma^2 = 0.1$).

- What is the class of possible motion models we are considering? What are the parameters that need to be estimated for each model? (e.g. translational models or affine models)
- What is the nature of the data that the models need to fit? (e.g. raw pixel data or results of optical flow).

Here we will describe the simplest version of an EM based motion segmentation algorithm: the models will be assumed to be translational and the image data will be results of optical flow. More complex versions are described in the references.

We are given the results of optical flow $v_x(i,j)$ and $v_y(i,j)$. We assume this flow was generated by two global translation models. Each translation model is characterized by two numbers: $(u,v)$ which give the horizontal and vertical translation. The problem is to estimate the parameters $(u_1,v_1),(u_2,v_2)$ as well as assigning each pixel to the model that generated it.

The E step is entirely analogous to the one described for line fitting. We estimate two weights for every pixel $w_1(i,j)$ and $w_2(i,j)$. This is done by assuming $(u_1,v_1),(u_2,v_2)$ are known and calculating residuals:

$$r_1^2(i,j) = (u_1 - v_x(i,j))^2 + (v_1 - v_y(i,j))^2 \tag{5}$$

$$r_2^2(i,j) = (u_2 - v_x(i,j))^2 + (v_2 - v_y(i,j))^2 \tag{6}$$

The residuals are converted into weights by passing them through the softmin function:

$$w_1(i,j) = \frac{e^{-r_1^2(i,j)/\sigma^2}}{e^{-r_1^2(i,j)/\sigma^2} + e^{-r_2^2(i,j)/\sigma^2}} \tag{7}$$

$$w_2(i,j) = \frac{e^{-r_2^2(i,j)/\sigma^2}}{e^{-r_1^2(i,j)/\sigma^2} + e^{-r_2^2(i,j)/\sigma^2}} \tag{8}$$

The M step is also analogous to the one used for line fitting. In this case, the parameters $(u_1,v_1)$ satisfy:

$$\begin{pmatrix} \sum_{i,j} w_1(i,j) & 0 \\ 0 & \sum_{i,j} w_1(i,j) \end{pmatrix} \begin{bmatrix} u_1 \\ v_1 \end{bmatrix} = \begin{bmatrix} \sum_{i,j} w_1(i,j) v_x(i,j) \\ \sum_{i,j} w_1(i,j) v_y(i,j) \end{bmatrix} \tag{9}$$

And the equations for $(u_2,v_2)$ are the same with $w_1(i,j)$ replaced everywhere by $w_2(i,j)$.

To summarize, the motion segmentation starts by choosing an initial random value for the translation parameters, and iterating the E and M steps detailed above until the translation parameters converge.

In the simple case described here, since the EM algorithm is using the results of an optical flow algorithm, it makes sense to get the final segmentation by going back to the raw pixel data. That is, after the algorithm has converged and two translation models have been found, we let the models "compete" to explain the pixel data. This is done by:

- warping frame 2 to frame 1 with the translation of model 1. Subtracting the warped image from the true frame 1. This gives the prediction error of model 1.
- repeating the previous step with the translation of model 2 and obtaining the prediction error of model 2.
- assigning each pixel to the model whose prediction error is lowest at that pixel.

## 4 Conclusion

This tutorial has discussed how to use the EM algorithm for motion segmentation. Although we have given an informal exposition, it should be noted that the actual algorithm is derived from a rigorous statistical estimation framework and proofs about convergence and rate of convergence can be found in the literature. The combination of the algorithm's principled derivation and intuitively decoupled steps is probably responsible for its continued success in many estimation problems.

## References


Ayer, S. and Sawhney, H. S. (1995). Layered representation of motion video using robust maximum likelihood estimation of mixture models and mdl encoding. In *Proc. Int'l Conf. Comput. Vision*, pages 777–784.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. R. Statist. Soc. B*, 39:1–38.

Jepson, A. and Black, M. J. (1993). Mixture models for optical flow computation. In *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, pages 760–761, New York.

Weiss, Y. and Adelson, E. H. (1994). Perceptually organized EM: a framework for motion segmentation that combines information about form and motion. Technical Report 315, MIT Media Lab, Perceptual Computing Section.

Weiss, Y. and Adelson, E. H. (1996). A unified mixture framework for motion segmentation: incorporating spatial coherence and estimating the number of models. In *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, pages 321–326.