Inverse Kinematics via Gaussian Mixture Modeling

Aykut C. Satici
July 10, 2021

1 Problem Statement

We want to devise a general procedure that solves the inverse kinematics problem for redundant manipulators. Concretely, I will work with the planar three-link articulated robot. We are only interested in the position of its end-effector. Therefore, the forward kinematics map goes from the joint angles $\theta = (\theta_1, \theta_2, \theta_3)$ to the end-effector position $\mathrm{x} = (x, y)$. The problem is to find the inverse (one-to-many) correspondence, i.e., the joint angles θ inducing a given end-effector position x.

2 Theory

The solution will be to utilize a Gaussian Mixture Model (GMM) to represent the inverse kinematics map. My presentation will closely follow three sources: Ghahramani (1993); McLachlan and Krishnan (2007); Xu, Chen, Lau, and Ren (2017).

We assume that the data $\Xi = \{\xi_1, \ldots, \xi_N\}$ were generated independently by a mixture density:

$$P(\xi_i) = \sum_{j=1}^{M} \underbrace{P(\omega_j)}_{\pi_j} P(\xi_i \mid \omega_j; \psi_j), \tag{1}$$

where $\{\pi_j\}_j^M$ are the mixture proportions, each component of the mixture is denoted ω_j and parametrized by ψ_j. The mixture proportions are to be picked such that

$$\sum_{j=1}^{M} \pi_j = 1.$$

The full set of parameters of this model are π_j's and ψ_j's. We will write $\Psi = (\pi_j, \psi_j)_j$. When we restrict these probability distributions to be Gaussian, we will further have $\psi_j = (\mu_j, \Sigma_j)$, where μ_j is the mean and Σ_j the covariance of the j^{th} Gaussian. The log of the likelihood of the parameters given the data set is

$$\ell(\Psi \mid \Xi) = \log \prod_{i=1}^{N} \sum_{j=1}^{M} P(\xi_i \mid \omega_j; \psi_j) P(\omega_j) = \sum_{i=1}^{N} \log \sum_{j=1}^{M} P(\xi_i \mid \omega_j; \psi_j) P(\omega_j). \tag{2}$$

We seek to find the parameter vector Ψ that maximizes $\ell(\Psi \mid \Xi)$. However, this function is not easily maximized numerically because it involves the log of a sum. Intuitively, it is not easily maximized because for each data point, there is a "credit-assignment" problem, i.e., it is not clear which component of the mixture generated the data point and thus which parameters to adjust to fit that data point.

The expectation maximization (EM) algorithm applied to mixtures is an iterative method for overcoming this credit-assignment problem. The intuition behind it is that if one had access to a "hidden" random variable z that indicated which data point was generated by which component, then the maximization problem would decouple into a set of simple maximizations. Mathematically, given $\mathcal{Z} = \{z_1, \ldots, z_N\}$ a "complete-data" log likelihood function could be written,

$$\ell_c(\Psi \mid \Xi, \mathcal{Z}) = \sum_{i=1}^{N} \sum_{j=1}^{M} z_{ij} \log P(\xi_i \mid z_i; \Psi) P(z_i; \Psi), \tag{3}$$

such that it does not involve a log of a summation.

As proven in Dempster, Laird, and Rubin (1977), $\ell(\Psi \mid \Xi)$ can be maximized by iterating the following two steps,

$$\begin{aligned} \text{E-step:} \quad & Q(\Psi \mid \Psi_k) = \mathbb{E}\left[\ell_c(\Psi \mid \Xi, \mathcal{Z}) \mid \Xi, \Psi_k\right] \\ \text{M-step:} \quad & \Psi_{k+1} = \arg\max_{\Psi} Q(\Psi \mid \Psi_k). \end{aligned} \tag{4}$$

The E (Expectation) step computes the expected complete data log likelihood and the M (Maximization) step finds the parameters that maximize this likelihood.

2.1 Mixture of Gaussians

Let me now specialize to the case where the probability functions above are Gaussians. For this model, the E-step simplifies to computing $h_{ij} = \mathbb{E}\left[z_{ij} \mid \xi_i, \psi_k\right]$, the probability that Gaussian j, as defined by the mean $\hat{\mu}_j$ and covariance matrix $\hat{\Sigma}_j$ estimated at time step k, generated data point i:

$$h_{ij}^{(k)} = \frac{\left|\hat{\Sigma}_j^{(k)}\right|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}\left(\xi_i - \hat{\mu}_j^{(k)}\right)^{\top}\left(\hat{\Sigma}_j^{(k)}\right)^{-1}\left(\xi_i - \hat{\mu}_j^{(k)}\right)\right\}}{\sum_{l=1}^{M}\left|\hat{\Sigma}_j^{(k)}\right|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}\left(\xi_i - \hat{\mu}_l^{(k)}\right)^{\top}\left(\hat{\Sigma}_l^{(k)}\right)^{-1}\left(\xi_i - \hat{\mu}_l^{(k)}\right)\right\}}. \tag{5}$$

The M-step then involves re-estimating the means and covariances of the Gaussians along with the mixing proportions using the data set weighted by the h_{ij}:

$$\begin{aligned} \hat{\mu}_j^{(k+1)} &= \frac{\sum_{i=1}^{N} h_{ij}^{(k)} \xi_i}{\sum_{i=1}^{N} h_{ij}^{(k)}}, \\ \hat{\Sigma}_j^{(k+1)} &= \frac{\sum_{i=1}^{N} h_{ij}^{(k)}\left(\xi_i - \hat{\mu}_j^{(k+1)}\right)\left(\xi_i - \hat{\mu}_j^{(k+1)}\right)^{\top}}{\sum_{i=1}^{N} h_{ij}^{(k)}}, \\ \pi_j^{(k+1)} &= \frac{1}{N}\sum_{i=1}^{N} h_{ij}^{(k)}. \end{aligned} \tag{6}$$

2.2 Supervised Learning

When viewed as supervised learning, each vector ξ_i in the training set is composed of an "input" subvector x_i and a "target" or output subvector θ_i. Applying the learning algorithm, we obtain

an estimate of the density of the data in this input/output space. This estimate can be used to approximate a function in the following way:

Given the input vector x_i, we extract all the relevant information from the joint probability density function (pdf) $P(\mathrm{x}, \theta)$ by conditionalizing to $P(\theta \mid \mathrm{x})$. For a single Gaussian this conditional density is normal, and by linearity, since $P(\mathrm{x}, \theta)$ is a mixture of Gaussians, so is $P(\theta \mid \mathrm{x})$. Let me elaborate on how to compute this posterior (conditional) density for Gaussian mixture models. First, we partition the mean vector $\hat{\mu}_j$ and the covariance matrix $\hat{\Sigma}_j$.

$$\hat{\mu}_j = \begin{bmatrix} \hat{\mu}_{\mathrm{x},j} & \hat{\mu}_{\theta,j} \end{bmatrix}, \qquad \hat{\Sigma}_j = \begin{bmatrix} \Sigma_{\mathrm{xx},j} & \Sigma_{\mathrm{x}\theta,j} \\ \Sigma_{\theta\mathrm{x},j} & \Sigma_{\theta\theta,j} \end{bmatrix}. \tag{7}$$

Given each component ω_j and output x, the conditionial probability distributioin of θ can be obtained as:

$$\begin{aligned} P(\theta \mid \mathrm{x}; \omega_j) &= \mathcal{N}\left(\theta \mid \tilde{\mu}_{\theta,j}, \tilde{\Sigma}_{\theta\theta}\right), \\ \tilde{\mu}_{\theta,j} &= \mu_{\theta,j} + \Sigma_{\theta\mathrm{x},j}\Sigma_{\mathrm{xx},j}^{-1}\left(\mathrm{x} - \mu_{\mathrm{x},j}\right), \\ \tilde{\Sigma}_{\theta\theta,j} &= \Sigma_{\theta\theta,j} - \Sigma_{\theta\mathrm{x},j}\Sigma_{\mathrm{xx},j}^{-1}\Sigma_{\mathrm{x}\theta,j}. \end{aligned} \tag{8}$$

Finally, we marginalize out the component ω_j to find the conditional probability of θ given x:

$$\begin{aligned} P(\theta \mid \mathrm{x}) &= \sum_{j=1}^{M} \beta_j \mathcal{N}\left(\theta \mid \tilde{\mu}_{\theta,j}, \tilde{\Sigma}_{\theta\theta,j}\right), \\ \beta_j &= \frac{\pi_j P(\mathrm{x}_j)}{\sum_{j=1}^{M} \pi_j P(\mathrm{x}_j)} = \frac{\pi_j \mathcal{N}\left(\mathrm{x} \mid \mu_{\mathrm{x},j}, \Sigma_{\mathrm{xx},j}\right)}{\sum_{j=1}^{M} \pi_j \mathcal{N}\left(\mathrm{x} \mid \mu_{\mathrm{x},j}, \Sigma_{\mathrm{xx},j}\right)}. \end{aligned} \tag{9}$$

Here, notice that, to compute β_j, we compute the probability of observing the data point x under the Gaussian distribution whose mean is $\mu_{\mathrm{x},j}$ and whose covariance is $\Sigma_{\mathrm{xx},j}$, for each j.

In principle, this conditional density is the final output of the density estimator. That is, given a particular input, the network returns the complete conditional density of the output. However, for the purposes of comparison to function approximation methods and since many applications require a single estimate of the output, I will outline two possible ways to obtain such an estimate $\hat{\theta}_i$ of $\theta_i = f(x_i)$.

1. Stochastic Sampling (STOCH) samples according to the distribution $\hat{\theta}(\mathrm{x}_i) \sim P(\theta \mid \mathrm{x}_i)$:

$$\hat{\theta} = \sum_{j=1}^{M} \beta_j \tilde{\mu}_{\theta,j}, \qquad \hat{\Sigma}_{\theta\theta} = \sum_{j=1}^{M} \beta_j^2 \tilde{\Sigma}_{\theta\theta,j}.$$

2. Single component least-squares estimation (SLSE) takes $\hat{\theta}(\mathrm{x}_i) = \mathbb{E}\left(\theta \mid \mathrm{x}_i, \omega_j\right)$, where $j = \arg\max_k \mathcal{N}\left(\mathrm{x} \mid \mu_{\mathrm{x},k}, \Sigma_{\mathrm{xx}_k}\right)$. That is, for a given input, SLSE picks the Gaussian with the highest posterior, and for that Gaussian approximates the output with the least-squares estimator given by that Gaussian alone.

2.3 Initialization of Expectation Maximization

The EM algorithm is iterative, which means, in particular, that we need to start it off with an initial guess for the parameters. Clearly, the better the initial guess, the faster/easier the convergence will be. The initial guesses I used were as follows:

1. For π_j, I used a uniform distribution: $\pi_j^{(0)} = \frac{1}{M}$ for all $j = 1, \ldots, M$.
2. For each j, I started the covariance matrix as a multiple of the identity matrix: $\Sigma_j^{(0)} = \alpha_j I$.
3. I used the k-means clustering method to generate initial values for $\mu_j^{(0)}$. This method aims to partition the N observations into $M \leq N$ sets so as to minimize the within-cluster sum-of-squares cost Wikipedia (2021).

2.4 Summary of the Algorithm

Let me provide a brief run-down of the method.

1. Randomly select the data points $\{\theta_1, \ldots, \theta_N\}$ and generate the corresponding $\{\mathrm{x}_1, \ldots, \mathrm{x}_N\}$.
2. Collect the full data in one vector: $\xi_i = \begin{bmatrix}\mathrm{x}_i, \theta_i\end{bmatrix}$ and $\xi = \{\xi_i\}_{i=1}^N$.
3. Model the data to be generated independently by a Gaussian mixture density (1) and initialize the parameters of the model following Section 2.3.
4. Employ EM to find $P(\xi) = P(\mathrm{x}, \theta)$ using equations (5) and (6).
5. Marginalize the sum of the Gaussians to find $P(\theta \mid \mathrm{x})$ using equation (7)–(9).
6. Given an end-effector position x, estimate the joint angles using STOCH or SLSE.

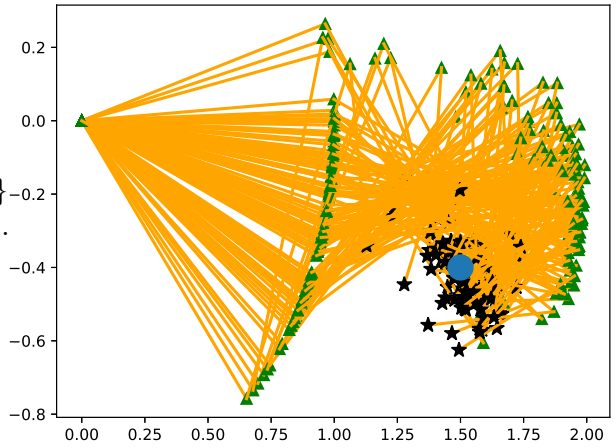

Figure 1: A visualization of the algorithm run with $N = 2001$, $M = 101$ over the range $-\pi \leq \theta_1, \theta_2, \theta_3 \leq \pi$. The desired end-effector position, $\mathrm{x} = \begin{bmatrix}1.5 & -0.4\end{bmatrix}$, is depicted by the big blue circle. The learned posterior distribution $P(\theta \mid \mathrm{x})$ is used to sample 100 solutions, which yield the configurations shown in orange. The actual location of the end-effector the orange configurations yield are depicted by the black stars.

3 Numerical Example

A Julia implementation of this algorithm may be found in Satici (2021). The implementation allows the user to specify how many data points to randomly generate and how many components to use in the Gaussian mixture model. Once these are specified, the model can be trained by executing the `execute_em!` function.

The trained model with $N = 2001$ sample points and $M = 101$ components over the range $-\pi \leq \theta_1, \theta_2, \theta_3 \leq \pi$ may be tested by running the `test_training` function, which generates 200 extra points and returns the average error incurred by the output of the algorithm. A visualization of the performance of the algorithm is depicted in Figure 1. Table 1 compares how the performance of the algoritm changes as the number of components M is varied. The posterior $P(\theta \mid \mathrm{x})$ is initialized with a uniform covariance matrix with a mean coming from the initial k-means algorithm.

Components	41	51	61	71	81	91	101	111	121
Mean ℓ_2 error	0.2368	0.2104	0.1613	0.1795	0.1685	0.1926	0.1392	0.1792	0.1584

Table 1: The performance of the algorithm for different number of Gaussian components for $N = 2001$ data points, tested over 200 fresh samples.

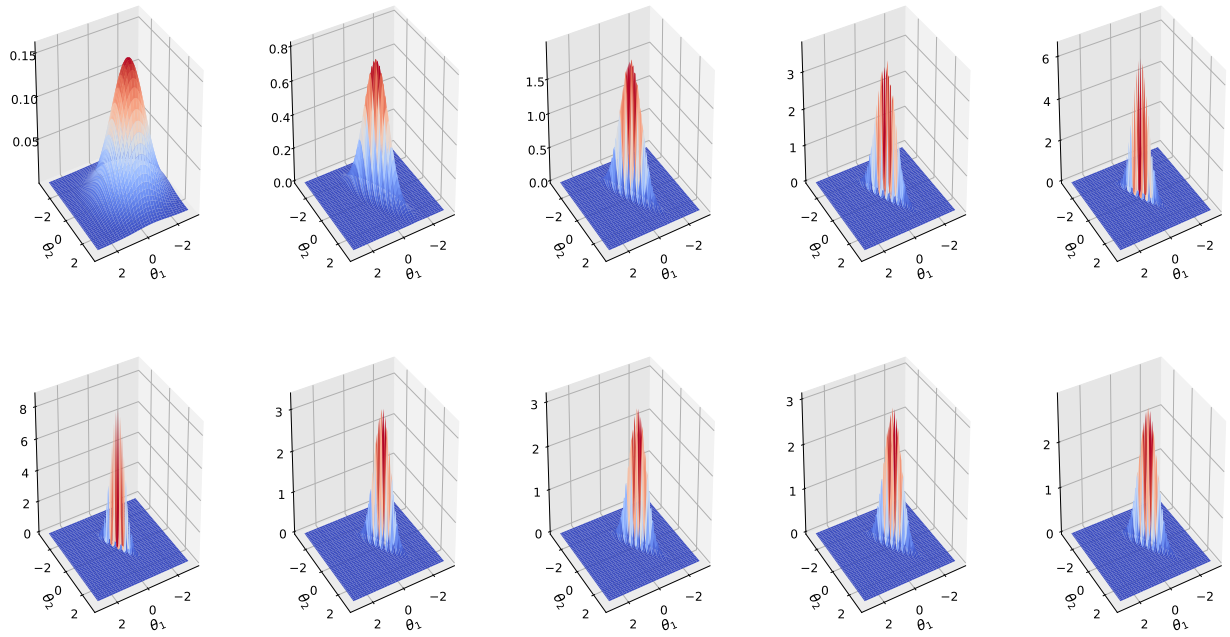

Figure 2: (Progression: left-to-right then top-to-bottom) The evolution of the posterior distribution, $P(\theta \mid \mathrm{x})$, as the EM algorithm iterates. The end-effector location is $\mathrm{x} = \begin{bmatrix} 1.5 & -0.4 \end{bmatrix}$ as in Figure 1. The posterior distribution $P(\theta \mid \mathrm{x})$ is marginalized over θ_3 for visualization.

As the EM algorithm proceeds, our belief on $P(\theta \mid \mathrm{x})$ is updated. The evolution of our belief is depicted in Figure 2.

References

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, *39*(1), 1–22.

Ghahramani, Z. (1993). Solving inverse problems using an em approach to density estimation. In *Proceedings of the 1993 connectionist models summer school.*

McLachlan, G. J., & Krishnan, T. (2007). *The em algorithm and extensions* (Vol. 382). John Wiley & Sons.

Satici, A. C. (2021). *Inverse kinematics via gaussian mixture modeling.* https://github.com/Symplectomorphism/gmm-ik. ([Online; accessed 09-July-2021])

Wikipedia. (2021). *K-means clustering — Wikipedia, the free encyclopedia.* http://en.wikipedia.org/w/index.php?title=K-means%20clustering&oldid=1029789661. ([Online; accessed 08-July-2021])

Xu, W., Chen, J., Lau, H. Y., & Ren, H. (2017). Data-driven methods towards learning the highly nonlinear inverse kinematics of tendon-driven surgical manipulators. *The International Journal of Medical Robotics and Computer Assisted Surgery*, *13*(3), e1774.