Bayesian inference in diffusion tensor imaging

Course project in Advanced Probabilistic Machine Learning

Jens Sjölund JENS.SJOLUND@IT.UU.SE
Anton O'Nils ANTON.O-NILS@IT.UU.SE
Stina Brunzell STINA.BRUNZELL@IT.UU.SE
Lovisa Eriksson LOVISA.ERIKSSON@IT.UU.SE
Department of Information Technology
Uppsala University, Sweden

Abstract

In this course project, your goal is to infer local tissue properties from real-world diffusion MRI data. Given the same Bayesian model, each student implements a different inference technique. Together, you then compare and contrast the different techniques.

1. Introduction

Water molecules in tissue are constantly moving due to thermal energy. In free water, this motion is uniform in all directions (isotropic diffusion). In white matter, however, cell membranes and myelin sheaths restrict motion, causing water to diffuse more easily along axons than across them. Diffusion MRI measures this effect by applying magnetic field gradients in different directions and observing how the signal changes.

The challenge is to infer the local tissue structure from these measurements. One of the simplest and most widely used models is the *diffusion tensor model* (Basser et al., 1994). Although it was introduced more than 30 years ago, it remains popular because it is easy to fit, interpret, and visualize. In this model, diffusion within a voxel is described by a 3×3 symmetric positive-definite matrix—the *diffusion tensor*.

We model the measured diffusion signal at position $\mathbf{r}$ under an experimental setting $\mathbf{x} \in \mathbb{R}^3$ as:

$$S(\mathbf{r}, \mathbf{x}) = S_0(\mathbf{r}) \exp\big(-\mathbf{x}^T \mathbf{D}(\mathbf{r})\, \mathbf{x}\big)\,, \tag{1}$$

where S_0 is the nonnegative baseline signal without any diffusion weighting ($\mathbf{x} = 0$), $\mathbf{x}$ encodes the diffusion gradient direction and strength for a single measurement, and $\mathbf{D}$ is the symmetric diffusion tensor

$$\mathbf{D} = \begin{pmatrix} D_{xx} & D_{xy} & D_{xz} \\ D_{xy} & D_{yy} & D_{yz} \\ D_{xz} & D_{yz} & D_{zz} \end{pmatrix}. \tag{2}$$

Physically, $\mathbf{D}$ describes how diffusion occurs along and across different spatial directions, similar to a conductivity or stiffness matrix in other domains.

Let y denote a single noisy measurement. When the signal-to-noise ratio is high, it is reasonable to approximate the noise as Gaussian with known variance σ^2,

$$p(y \mid \mathbf{r}, \mathbf{x}) = \mathcal{N}\big(y; S(\mathbf{r}, \mathbf{x}), \sigma^2\big). \tag{3}$$

In practice, one performs the same N measurements at all positions in a 3D grid, resulting in a 4D tensor. However, for computational convenience, we only consider a single volume element (voxel) in this project.

Goal of the project. Given the observed variables $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ for a single voxel, our goal is to estimate two things. First, the baseline signal S_0, which is the signal intensity we would measure if no diffusion weighting were applied, and second, the six independent entries of the diffusion tensor $\boldsymbol{D}$, which describe how water molecules diffuse in different directions within that voxel.

Why these quantities matter

The baseline signal S_0 reflects the local tissue density and serves as a reference level for interpreting the diffusion-weighted measurements. The diffusion tensor $\boldsymbol{D}$ provides information about how water moves microscopically in the tissue, which in white matter is strongly influenced by the orientation and integrity of nerve fibers. From $\boldsymbol{D}$, one can compute derived quantities, such as fractional anisotropy or principal diffusion directions, which are routinely used in neuroscience and medicine to map structural connectivity in the brain or detect tissue damage. Importantly, both S_0 and $\boldsymbol{D}$ are *voxel-specific* – they are estimated separately for each small volume element of the brain. In a real scan there are tens of thousands of voxels, and combining their estimates provides a patient-specific map of white-matter microstructure. In this project, however, we focus on the much simpler setting of a single voxel to explore Bayesian inference techniques in depth.

2. Bayesian model

There are many ways to fit a diffusion tensor model to data. Several can be interpreted as special cases of Bayesian linear regression (Sjölund et al., 2018). Here, we take a different approach, following the Bayesian model of Behrens et al. (2003), for which you will compare different inference methods.

The model is based on the eigendecomposition of the diffusion tensor

$$\mathbf{D} = \mathbf{V}\,\mathbf{\Lambda}\,\mathbf{V}^T, \tag{4}$$

where $\mathbf{\Lambda}$ is a diagonal matrix with nonnegative eigenvalues $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \lambda_3)$, and $\mathbf{V} \in \mathrm{SO}(3)$ is a 3×3 rotation matrix (orthogonal with determinant $+1$) that specifies the principal diffusion directions. The matrix $\mathbf{V}$ thus encodes the orientation of the tensor's principal axes, while $\mathbf{\Lambda}$ determines the magnitude of diffusion along each axis.

Our latent (unknown) variables are

$$\boldsymbol{z} = \{S_0, \boldsymbol{\lambda}, \boldsymbol{V}\}, \tag{5}$$

and we place independent priors on each component:

$$p(\boldsymbol{z}) = p(S_0)\; p(\lambda_1)\; p(\lambda_2)\; p(\lambda_3)\; p(\boldsymbol{V}), \tag{6}$$

$$S_0 \sim \mathrm{Gamma}(\alpha_S, \theta_S), \tag{7}$$

$$\lambda_1, \lambda_2, \lambda_3 \sim \mathrm{Gamma}(\alpha_\lambda, \theta_\lambda), \tag{8}$$

$$\boldsymbol{V} \sim \mathrm{Uniform}\,(\mathrm{SO}(3))\,. \tag{9}$$

We use the *shape–scale* parameterization of the Gamma distribution consistently throughout the project, hence, a Gamma distributed random variable x has the probability density function

$$p(x \mid \alpha, \theta) = \frac{1}{\Gamma(\alpha)\theta^{\alpha}} x^{\alpha-1} e^{-x/\theta}, \tag{10}$$

where $\alpha > 0$ and $\theta > 0$ are the shape and scale parameters, respectively. Table 1 lists the hyperparameters used for the priors and likelihood.

Hyperparameter	Value
σ	29
α_S	2
θ_S	500
α_λ	4
θ_λ	$2.5 \cdot 10^{-4}$

Table 1: Hyperparameters of the prior and likelihood.

The uniform distribution on SO(3) reflects the assumption that there is no preferred diffusion direction a priori. To draw random rotations in practice, you can use the Scipy implementation[1].

Uniform prior on SO(3)

If we say $\boldsymbol{V} \sim \text{Uniform}(\text{SO}(3))$, we mean that all possible 3D orientations are equally likely. This is analogous to picking a random point on the surface of a sphere (uniform over $\mathbb{S}^2$), but in three dimensions we must also account for the possible rotations around any chosen axis. A uniform prior on SO(3) ensures that, over many draws, there is no bias toward any specific direction or orientation in space.

3. The Project

In this project, you will work in small groups to implement and compare Bayesian inference methods for the diffusion tensor model. The work is divided into four stages:

1. **Explore the data.** Make sure you understand how the variables in the provided code relate to the variables in the equations. Drawing a graphical model may help. Implement the likelihood and prior. In particular, write functions for:

   - Sampling random variables (`rvs`).
   - Evaluating the log of the probability density (`logpdf`).

   Work in log-probabilities to avoid numerical underflow. Check your implementation by verifying that empirical sample means agree with theoretical means.

2. **Implement inference methods.** Each group member chooses one of the four inference methods described below. All members should choose different methods. Your final report must include an explicit contribution statement describing who implemented each method.

3. **Compare the results.** Evaluate and discuss the methods:

   - Do the credible intervals reflect the true uncertainty?

1. `https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.transform.Rotation.random.html`

- What are the strengths and weaknesses of each method?
- How might they be improved?
- Consider scalability: in a real scan there may be $\sim 10^5$ voxels. Could each method handle this?

4. **Write the report.** Maximum 8 pages, excluding references and supplementary material. Use the provided LaTeX template, which is based on that for the NeurIPS conference. A suggested structure is:

   (a) Front page: title, names, abstract, contribution statement. The first submission should, however, be anonymous, i.e., without names and contribution statement.

   (b) Shared methodology and preprocessing (1 page).

   (c) One page for each method: design choices, results, figures/tables.

   (d) Comparison and discussion (2 pages).

   Submit the complete code needed to reproduce your results, together with the report.

Dataset

You will work with data from the Stanford HARDI dataset (Rokem et al., 2015), comprising diffusion-weighted MRI data from two healthy male participants, age 37 and 27, which the provided code automatically downloads for you. The experimental procedures were approved by the Stanford University Institutional Review Board and participants provided informed consent.[2]

Metrics

We encourage you to use the `plot_results` function in the provided code, which produces histograms for the baseline signal and some common scalar summaries defined as follows. The *mean diffusivity* (MD) is the average of the three eigenvalues $\lambda_1, \lambda_2, \lambda_3$ of $\boldsymbol{D}$,

$$\mathrm{MD} = \tfrac{1}{3}(\lambda_1 + \lambda_2 + \lambda_3),$$

and reflects the overall magnitude of diffusion independent of direction. The *fractional anisotropy* (FA) quantifies how strongly diffusion is directionally dependent. It is defined as

$$\mathrm{FA} = \sqrt{\tfrac{3}{2}}\,\frac{\sqrt{(\lambda_1 - \mathrm{MD})^2 + (\lambda_2 - \mathrm{MD})^2 + (\lambda_3 - \mathrm{MD})^2}}{\sqrt{\lambda_1^2 + \lambda_2^2 + \lambda_3^2}},$$

with values close to 0 indicating isotropic diffusion (same in all directions) and values close to 1 indicating highly anisotropic diffusion (one dominant direction). Finally, the *acute angle* φ is defined as the angle between the principal eigenvectors $\boldsymbol{v}_1$ and $\hat{\boldsymbol{v}}_1$, respectively, of a sample and a reference (taken to be the point estimate),

$$\varphi = \arccos\left(|\boldsymbol{v}_1 \cdot \hat{\boldsymbol{v}}_1|\right).$$

2. Additional details on the data are available at `https://purl.stanford.edu/ng782rw8378`.

Option 1: The Metropolis-Hastings algorithm

The standard Metropolis–Hastings algorithm often uses a diagonal Gaussian proposal in parameter space. Why is that a poor choice for this model? Instead, use a factorized proposal that samples (proposes) variables $\boldsymbol{z}'$ given the current variables $\boldsymbol{z}$ according to

$$q(\boldsymbol{z}' \mid \boldsymbol{z}) = q(S_0' \mid S_0)\, q(\boldsymbol{D}' \mid \boldsymbol{D}), \tag{11}$$
$$q(S_0' \mid S_0) = \text{Gamma}(\gamma^{-2}, \gamma^2 S_0), \tag{12}$$
$$q(\boldsymbol{D}' \mid \boldsymbol{D}) = \mathcal{W}(\boldsymbol{D}, \nu), \tag{13}$$

where $\mathcal{W}$ is the Wishart distribution with degrees of freedom ν, and the scale factor γ controls the spread of are S_0' given S_0. (Find the mean and variance of equation (12) to see how γ works.) Note that, per definition, the mean of the Wishart distribution in equation (13) is $\nu\boldsymbol{D}$ and not $\boldsymbol{D}$.

Implement the Metropolis-Hastings algorithm, noting that the proposal in equation (11) is *not* symmetric. Tune the ν and γ parameters to obtain an acceptance rate of about 50%. Make a trace plot and use it to determine the burn-in time. *Hint:* Initializing at the point estimate provided by `dipy.reconst.dti` reduces the burn-in time.

Option 2: Importance sampling

In importance sampling, we approximate expectations under the posterior by reweighting samples from a proposal. Since standard importance sampling is not a form of Markov Chain Monte Carlo, it does not need a state-dependent proposal and does not have a burn-in time. Nevertheless, the choice of the proposal is very important for its efficiency.

Implement importance sampling, using the same proposal distribution as described for the Metropolis-Hastings algorithm. Tune the hyperparameters by tracking the *effective sample size* (ESS), defined as

$$\hat{N}_{\text{ESS}} = \frac{1}{\sum_{i=1}^{N} w_i^2}, \tag{14}$$

where w_i are the normalized importance weights.

Even with well-tuned hyperparameters, the effective sample size may be rather small when using this proposal. Suggest at least one alternative proposal, provide motivation, and compare it with the original one. For instance, you could combine importance sampling with other inference techniques, such as those implemented by your fellow group members, or explore extensions such as sequential Monte Carlo (Naesseth et al., 2019).

Sequential Monte Carlo (SMC)

SMC constructs a sequence of intermediate distributions

$$\pi_0(x) \to \pi_1(x) \to \cdots \to \pi_T(x),$$

starting from something easy to sample (e.g., the prior) and ending at the target (e.g., the posterior). A set of particles is propagated through this sequence by:

1. Reweighting: adjusting particle weights to account for the change from π_t to π_{t+1}.
2. Resampling: duplicating high-weight particles and discarding low-weight ones.
3. Rejuvenation: applying a Markov kernel to add diversity.

SMC is especially useful when the target is difficult to sample from directly, and it can be viewed as a bridge between importance sampling and MCMC.

Option 3: Variational inference

In variational inference, we approximate the posterior with a family of tractable distributions and choose the member of this family that is closest to the true posterior (in the sense of Kullback–Leibler divergence). Here, we use the variational posterior

$$q(\boldsymbol{z}) = q(S_0)\, q(\boldsymbol{D}), \tag{15}$$
$$q(S_0) = \Gamma(\alpha, \beta), \tag{16}$$
$$q(\boldsymbol{D}) = \mathcal{W}(\boldsymbol{D}, \nu). \tag{17}$$

To ensure that the shape and scale parameters are positive, reparameterize them as $\alpha = e^{\theta_1}$ and $\beta = e^{\theta_2}$. For the Wishart distribution, ensure positive definiteness of the diffusion tensor by parameterizing it by its Cholesky factorization $\boldsymbol{D} = \boldsymbol{L}\boldsymbol{L}^\top$ where

$$\boldsymbol{L} = \begin{pmatrix} e^{\theta_3} & & \\ \theta_4 & e^{\theta^5} & \\ \theta_6 & \theta_7 & e^{\theta_8} \end{pmatrix}. \tag{18}$$

Finally, set $\nu = e^{\theta_9} + 2$ to ensure that the degrees of freedom ν are at least 2 (the number of dimensions minus one).

We fit the variational posterior by maximizing the *evidence lower bound* (ELBO) with respect to $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_9)^\top$ using a stochastic gradient method (Sjölund, 2023). To accelerate convergence, it may help to use an optimization method that uses momementum, such as Adam (Kingma and Ba, 2015). Estimate the gradient of the ELBO using the REINFORCE leave-one-out estimator (Shi et al., 2022):

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{q_{\boldsymbol{\theta}}}\left[f(\boldsymbol{z})\right] = \mathbb{E}_{q_{\boldsymbol{\theta}}}\left[f(\boldsymbol{z}) \nabla_{\boldsymbol{\theta}} \log q_{\boldsymbol{\theta}}(\boldsymbol{z})\right] \tag{19}$$
$$\approx \frac{1}{K} \sum_{k=1}^{K} \left(f(\boldsymbol{z}^k) - \frac{1}{K-1} \sum_{j \neq k} f(\boldsymbol{z}^{(j)}) \right) \nabla_{\boldsymbol{\theta}} \log q_{\boldsymbol{\theta}}(\boldsymbol{z}^{(k)}), \tag{20}$$

where $K \geq 2$ is the number of Monte Carlo samples and $f(\boldsymbol{z}) = \log p(\mathcal{D}, \boldsymbol{z}) - \log q_{\boldsymbol{\theta}}(\boldsymbol{z})$.

The provided code contains a `variational_posterior` class that has a method to compute the score $\nabla_{\boldsymbol{\theta}} \log q_{\boldsymbol{\theta}}(\boldsymbol{z})$ analytically. Alternatively, you can use finite differences or automatic differentiation.

Option 4: Laplace approximation

The Laplace approximation approximates the posterior by fitting a Gaussian distribution centered at its mode. To ensure positivity, we work in a transformed space using a similar parameterization as described for variational inference above, namely, $S_0 = e^{\theta_1}$ and $\boldsymbol{D} = \boldsymbol{L}\boldsymbol{L}^\top$ where $\boldsymbol{L}$ is defined as in equation (18). Note, however, that the fitting parameters $\boldsymbol{\theta} \in \mathbb{R}^7$ are *not* identical to those in variational inference.

In the transformed space of the fitting parameters, we fit a Gaussian $\mathcal{N}(\hat{\boldsymbol{\theta}}, \Sigma)$, where $\hat{\boldsymbol{\theta}}$ is the maximizer of the log-posterior and $\boldsymbol{\Sigma}$ is derived from the curvature of the log-posterior at that point as

$$\boldsymbol{\Sigma} = \left[-\nabla^2_{\boldsymbol{\theta}} \log p(\boldsymbol{z}(\boldsymbol{\theta}) \mid \mathcal{D}) \big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} \right]^{-1}. \tag{21}$$

In practice, you first need to find the mode $\hat{\boldsymbol{\theta}}$ via numerical optimization (e.g., using L-BFGS or Newton-CG), and then compute the Hessian at that point to determine $\boldsymbol{\Sigma}$. Compute the gradient and Hessian of the log-posterior using either an analytical expression, a finite-difference approximation, or automatic differentiation.

We recommend encapsulating the end result as a distribution object similar to `scipy.stats.multivariate_normal`, exposing methods such as `rvs` for sampling and `logpdf` for evaluating the log-density. This provides a standardized interface for other parts of the project that may require sampling from, or evaluating, the approximate posterior.

Contribution statement

The individual contributions to the creation of this project, according to the Contributor Roles Taxonomy (CRediT), are as follows. **Jens Sjölund**: conceptualization, data curation, investigation, methodology, project administration, software, writing – original draft preparation, writing – review & editing. **Anton O'Nils**: software, validation, writing – review & editing. **Stina Brunzell**: software, validation, writing – review & editing. **Lovisa Eriksson**: writing – review & editing.

References

Peter J Basser, James Mattiello, and Denis LeBihan. MR diffusion tensor spectroscopy and imaging. *Biophysical journal*, 66(1):259–267, 1994.

Timothy EJ Behrens, Mark W Woolrich, Mark Jenkinson, Heidi Johansen-Berg, Rita G Nunes, Stuart Clare, Paul M Matthews, J Michael Brady, and Stephen M Smith. Characterization and propagation of uncertainty in diffusion-weighted MR imaging. *Magnetic Resonance in Medicine*, 50(5):1077–1088, 2003.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.

Christian A Naesseth, Fredrik Lindsten, and Thomas B Schön. Elements of sequential Monte Carlo. *Foundations and Trends® in Machine Learning*, 12(3):307–392, 2019.

Ariel Rokem, Jason D Yeatman, Franco Pestilli, Kendrick N Kay, Aviv Mezer, Stefan Van Der Walt, and Brian A Wandell. Evaluating the accuracy of diffusion MRI models in white matter. *PloS one*, 10(4):e0123272, 2015.

Jiaxin Shi, Yuhao Zhou, Jessica Hwang, Michalis Titsias, and Lester Mackey. Gradient estimation with discrete Stein operators. *Advances in neural information processing systems*, 35:25829–25841, 2022.

Jens Sjölund. A tutorial on parametric variational inference. *arXiv preprint arXiv:2301.01236*, 2023.

Jens Sjölund, Anders Eklund, Evren Özarslan, Magnus Herberthson, Maria Bånkestad, and Hans Knutsson. Bayesian uncertainty quantification in linear models for diffusion MRI. *NeuroImage*, 175:272–285, 2018.