Gaussian Processes for Orientation Preference Maps

Dominik Straub

August 16, 2018

This documents includes the notes I took while trying to reproduce Macke et al. [2011].

Notational remarks: N is the number of trials, d the number of map components and $n = n_x n_y$ the number of pixels. Vectors are boldfaced, matrices are uppercase letters and scalars are lowercase letters, unless noted otherwise.

1 Synthetic OPM

A synthetic ("ground truth") orientation preference map (OPM) is created as follows: The real part a(x) and the imaginary part b(x) are created as standard Gaussian white noise, which is then convoluted with a difference of Gaussians (DOG) filter (equations are analogous for b(x)):

$$\tilde{a}(x) = \mathcal{N}(0, 1) \tag{1}$$

$$a(x) = \tilde{a}(x) * \mathcal{N}(x \mid 0, \sigma^2) - \tilde{a}(x) * \mathcal{N}(x \mid 0, k\sigma^2) \tag{2}$$

The orientation preference map is then

$$m(x) = a(x) + ib(x) = A(x)\ \exp\left[2i\,\theta\,(x)\right], \tag{3}$$

where $A(x)$ is the absolute value of $m(x)$ and $2\theta(x)$ is the argument of $m(x)$.

Pinwheels of an OPM can be detected as zero-crossings of the real and imaginary parts of the map.

2 Response model

We now model the response to a stimulus with contrast c_i and orientation θ_i. The stimulus can be parametrized as $\mathbf{v}_i = (v_{1i}, v_{2i}, v_{3i})^T = (c_i \cos(2\theta_i), c_i \sin(2\theta_i), \sqrt{0.5})^T$, where $v_{1i} + iv_{2i} + v_{3i} = c_i * exp(2i\theta_i) + \sqrt{0.5}$.

The response at pixel x is a linear function of the stimulus parameters:

$$r_i(x) = \mathbf{v}_i^T \mathbf{m}(x) + \epsilon_i(x), \tag{4}$$

where $\mathbf{m}(x) = (a(x), b(x), c(x))^T$ (the orientation preference map components plus a constant). $\epsilon_i(x)$ models the noise at pixel x on trial i. Applying $\cos(\alpha - \beta) = \cos(\alpha)\cos(\beta) + \sin(\alpha)\sin(\beta)$, we obtain:

$$r_i(x) = A(x)\, c_i\ \cos(2(\psi(x) - \theta_i)) + c(x) + \epsilon_i(x), \tag{5}$$

which can be intuitively understood as tuning curves with preferred orientations $\psi(x)$.

2.1 Alternative notations

Equation (4) can also be written in matrix notation:

$$\mathbf{r}_i = M\mathbf{v}_i + \epsilon_i \tag{6}$$
$$= V_i^T\mathbf{m} + \epsilon_i, \tag{7}$$

where $V_i = \mathbf{v}_i \otimes \mathbb{I}_n$ is nd x n. The latter notation will be convenient when we define priors over maps, it includes all the map components in a vector.

The response model can also be formalized in a probabilistic fashion:

$$p(\mathbf{r}_i \,|\, \mathbf{m}) = \mathcal{N}\left(V_i^T\mathbf{m}, \Sigma_\epsilon\right), \tag{8}$$

where Σ_e is the noise covariance matrix. If the noise is independent across pixels and map components, Σ_e is diagonal.

2.2 Stimulus covariance matrix

No matter whether we use vector averaging (i.e. Maximum Likelihood Inference, Section 3) or Bayesian Inference (Section **??**), we will need the stimulus covariance matrix

$$\sum_{i=1}^{N} \mathbf{v}_i\mathbf{v}_i = V^TV, \tag{9}$$

where V is a matrix, whose i^{th} row is $\mathbf{v}_i^T$. V^TV is d x d and consists of the sums of outer products of the stimulus vectors:

$$V^TV = \sum_{i=1}^{N} \begin{pmatrix} \cos(2\theta_i)^2 & \cos(2\theta_i)\sin(2\theta_i) & \sqrt{1/2}(\cos(2\theta_i) \\ \cos(2\theta_i)\sin(2\theta_i) & \sin(2\theta_i)^2 & \sqrt{1/2}(\sin(2\theta_i) \\ \sqrt{1/2}(\cos(2\theta_i) & \sqrt{1/2}(\sin(2\theta_i) & 1/2 \end{pmatrix} \tag{10}$$

If we design our stimuli, s.t. $\theta_i = \frac{i\pi}{N}$, the sums reduce to:

$$\sum_{i=1}^{N} \cos{(2\theta_i)}^2 = \sum_{i=1}^{N} \frac{1}{2} + \frac{1}{2}\cos(2\theta_i) = \frac{N}{2} + \frac{1}{2}\sum_{i=1}^{N} \cos\left(\frac{2\theta_i\pi}{N}\right) \tag{11}$$

$$= \frac{N}{2} + \frac{1}{2}\left[\sum_{i=1}^{N/2} \cos\left(\frac{2\theta_i\pi}{N}\right) + \sum_{i=1}^{N/2} \cos\left(\pi + \frac{2\theta_i\pi}{N}\right)\right] = \frac{N}{2}. \tag{12}$$

The same holds for $\sum_{i=1}^{N} \sin(2\theta_i)^2 = N/2$, sin has the same periodicity as cos. The off-diagonal entries $\sum_{i=1}^{N} \sqrt{1/2}(\cos(2\theta_i)$ and $\sum_{i=1}^{N} \sqrt{1/2}(\sin(2\theta_i)$ sum up to 0 in the same way.

It remains

$$\sum_{i=1}^{N} \cos(2\theta_i)\sin(2\theta_i) = \sum_{i=1}^{N} \frac{1}{2}\sin\left(\frac{4i\pi}{N}\right) \tag{13}$$

$$= \sum_{i=1}^{N/2} \frac{1}{2}\sin\left(\frac{4i\pi}{N}\right) + \sum_{i=1}^{N/2} \frac{1}{2}\sin\left(2\pi + \frac{4i\pi}{N}\right) = \sum_{i=1}^{N/2} \sin\left(\frac{4i\pi}{N}\right) \tag{14}$$

$$= \sum_{i=1}^{N/4} \sin\left(\frac{4i\pi}{N}\right) + \sum_{i=1}^{N/4} \sin\left(\pi + \frac{4i\pi}{N}\right) = 0. \tag{15}$$

We then obtain a stimulus covariance matrix of the form

$$V^T V = \begin{pmatrix} N/2 & 0 & 0 \\ 0 & N/2 & 0 \\ 0 & 0 & N/2 \end{pmatrix} = \frac{N}{2}\mathbb{I}_d. \tag{16}$$

Thus, if we design stimuli of at least 8 different orientations evenly spaced between 0 and π, we get a stimulus matrix proportional to the identity.

3 Maximum Likelihood Inference

Assume that we have observed neural activities for a set of N stimuli:

$$\text{responses } r_1(x)...r_N(x) \tag{17}$$
$$\text{orientations } \theta_1...\theta_N \tag{18}$$
$$\text{contrasts } c_1...c_N \tag{19}$$

The usual procedure is called vector averaging and can be understood as the result of the maximum likelihood solution of the regression problem posed by our response model

$$r_i(x) = \mathbf{v}_i^T \mathbf{m}(x) + \epsilon_i(x) \tag{20}$$

Solving for the map components, we obtain

$$\begin{pmatrix} a(x) \\ b(x) \\ c(x) \end{pmatrix} = \left(\sum_i \mathbf{v}_i \mathbf{v}_i^T \right)^{-1} \sum_i \mathbf{v}_i r_i(x) \tag{21}$$
$$= \left(V^T V\right)^{-1} V \mathbf{r}(x), \tag{22}$$

where V is a d x N matrix, whose i^{th} row is $\mathbf{v}_i^T$. This notation represents a solution for each pixel individually. If we adopt matrix notation, we can write the solution for all pixels as

$$\begin{pmatrix} \mathbf{a}^T \\ \mathbf{b}^T \\ \mathbf{c}^T \end{pmatrix} = \left(V^T V\right)^{-1} V R, \tag{23}$$

where R is N x n and $\mathbf{a}, \mathbf{b}$ and $\mathbf{c}$ are n x 1.

4 Bayesian Inference

The prior distribution over orientation preference maps can be specified as a Gaussian process with mean function $\mu_{prior}(x,k)$ (which is the expected value at the pixel x for the k^{th} map component) and covariance function $K_{prior}(x,x',k,l) = Cov_{prior}(m_k(x), m_l(x'))$. We assume that all orientations are equally likely a priori: $\mu_{prior}(x,k) = 0$.

4.1 Prior covariance

The covariance function encodes a priori beliefs about how orientation preferences are correlated between different pixels and map components. We assume that the map components are independent: $Cov_{prior}(m_k(x), m_l(x')) = 0 \; \forall \, x, x'$.

We assume that the real and imaginary part of OPMs are generated as described in Section 1, i.e. Gaussian white noise $\tilde{a}(x) = \mathcal{N}(0,1)$ is convoluted with a DOG filter

$$f(x) = \sum_{k=1}^{2} \frac{\alpha_k}{2\pi\sigma_k^2} \exp\left\{ \frac{||x||^2}{2\sigma_k^2} \right\}. \tag{24}$$

This filter can be parametrized with three parameters (this is the parametrization I use in my code):

- σ, the standard deviation of the positive Gaussian (σ_1)
- k, the scaling factor for the standard deviation of the negative Gaussian w.r.t. that of the positive Gaussian ($k = \frac{\sigma_2}{\sigma_1}$)
- α, an overall scaling factor ($\alpha_1 = -\alpha_2$)

One map component is then

$$\begin{aligned}
a(x) = (\tilde{(a)} * f)(x) &= \int_{-\infty}^{\infty} \tilde{a}(x) f(x-u)\, du && (25)\\
&= \int_{-\infty}^{\infty} \tilde{a}(x) \left(\sum_{k=1}^{2} \frac{\alpha_k}{2\pi\sigma_k^2} \exp\left\{ \frac{||x-u||^2}{2\sigma_k^2} \right\} \right) du && (26)\\
&= \sum_{k=1}^{2} \alpha_k \int_{-\infty}^{\infty} \tilde{a}(x) G_k(x-u)\, du && (27)\\
&= \sum_{k=1}^{2} \alpha_k \, \mathbb{E}\left[G_k(x-u) \right], && (28)
\end{aligned}$$

where $G_k(x) = \frac{1}{2\pi\sigma_k^2} \exp\left\{ \frac{||x||^2}{2\sigma_k^2} \right\}$.
The covariance function is then

$$\begin{aligned}
Cov_{prior}(a(x), a(x')) &= \mathbb{E}[a(x)a(x')] - \mathbb{E}[a(x)]\mathbb{E}[a(x')] && (29)\\
&= \mathbb{E}[a(x)a(x')] && (30)\\
&= \sum_{a,b=1}^{2} \alpha_a \alpha_b \, \mathbb{E}\left[\mathbb{E}\left[G_a(x-u) \right] \mathbb{E}\left[G_b(x-u) \right] \right] && (31)\\
&= \sum_{a,b=1}^{2} \alpha_a \alpha_b \, \mathbb{E}[G_a(x-u) G_b(x-u)] && (32)\\
&= \sum_{a,b=1}^{2} \frac{\alpha_a \alpha_b}{2\pi(\sigma_a^2 + \sigma_b^2)} \exp\left\{ \frac{||x-x'||^2}{2(\sigma_a^2 + \sigma_b^2)} \right\} && (33)
\end{aligned}$$

This function is the kernel, which is used to obtain the covariance matrix K for one map component. Since the map components are independent, the full prior covariance matrix can be written as $K_m = \mathbb{I}_d \otimes K$.

4.1.1 Fit prior parameters to empirical map

A straightforward way to estimate the hyperparameters, i.e. the parameters σ and α of the covariance function, is to match them to the radial component of the autocorrelation of the emprirical map. Assuming we have obtained an empirical estimate of the real and imaginary components of the map (as in Section 3), we compute the autocorrelation of each component, which each is a matrix C (index ommited for readability) of dimensionality $(2n_x - 1, 2n_y - 1)$, where

$$C(i,j) = \frac{1}{n} \sum_{k=0}^{n_x-1} \sum_{l=0}^{n_y-1} M(k,l) M(k-i, l-j), \qquad (34)$$

where M is a map component reshaped as a n_x by n_y matrix. The indices of the matrix C are $-(n_x - 1) \leq i \leq n_x - 1$ and $-(n_y - 1) \leq j \leq n_y - 1$.

In order to obtain the radial component of this autocorrelation, a rotational average is performed, which means that all entries in this matrix, for which $k = \sqrt{i^2 + j^2}$ is the same, are averaged in order to obtain the average autocorrelation at distance k. Alternatively, one can bin the autocorrelations and average the bins.

The covariance function's hyperparameters are then optimized using least squares in order to match this radial component.

4.2 Posterior inference

The posterior covariance is

$$\Sigma_{\text{post}}^{-1} = K_m^{-1} + \sum_i \mathbf{v}_i \mathbf{v}_i^T \Sigma_\epsilon^{-1} \tag{35}$$

and the posterior mean is

$$\mu_{post} = \Sigma_{\text{post}}^{-1} \left(\mathbb{I}_d \otimes \Sigma_\epsilon^{-1} \right) \sum_i \mathbf{v}_i \otimes \mathbf{r}_i. \tag{36}$$

4.3 Noise model

Our response model assumes the data to be Gaussian distributed with mean $V_i^T \mathbf{m}$ and covariance matrix Σ_ϵ (see Section 2. Posterior inference as in Section 4.2 can be performed assuming that the noise covariance is known. This is not the case with real imaging data, so we have to estimate Σ_ϵ from the data. As imaging data generally have a very low signal-to-noise ratio, it is reasonable to start with an initial estimate that assumes the whole signal is noise, i.e. the covariance of the data. This is carried out for each stimulus condition separately and then averaged across stimulus conditions:

$$\Sigma_{\text{init}} = \frac{1}{N_c} \sum_{j=1}^{N_c} \Sigma_j, \tag{37}$$

where N_c is the number of conditions and Σ_i is covariance matrix in stimulus condition j. This estimate is then used to iteratively estimate the noise covariance by repeating the following steps:

1. Fit the posterior mean according to equation (36).
2. Update the noise covariance using a factor analysis model of the residuals $\mathbf{z}_i = \mathbf{r}_i - V_i^T \mu_{\text{post}}$.

4.3.1 Factor Analysis

In a factor analysis model, the data $\mathbf{z}_i$ are assumed to be a linear combination of q-dimensional latent variables $\mathbf{h}_i$ with additional Gaussian noise:

$$p(\mathbf{z}_i | \mathbf{h}_i) = \mathcal{N}(W\mathbf{h}_i + \mu, \Psi), \tag{38}$$

where Ψ is diagonal. The marginal distribution is then also Gaussian

$$p(\mathbf{z}_i) = \mathcal{N}(\mu, D + WW^T). \tag{39}$$

Thus, in a factor analysis model, the observed variables have a covariance matrix that can be understood as a sum of a diagonal matrix (modeling noise variances) and a low-rank matrix of rank q (modeling noise correlations).

4.4 Low-rank approximation

The prior covariance matrix K of a single map component is of shape n x n. The full covariance matrix K_m is then nd x nd. For large OPMs, we do not want to store or even invert this matrix. For our choice of covariance function (DOG), the spectrum of eigenvalues of K quickly drops off. Thus, we can approximate K by a low-rank matrix product $K = GG^T$, where G is n x q, q « n. This can be achieved by Incomplete Cholesky Decomposition (Bach and Jordan [2002], Section 4.2). The Incomplete Cholesky Decomposition can be performed without having to store the full covariance matrix K in memory. Its time complexity is $O(q^2 n)$ instead of $O(n^2)$.

This low-rank approximation can then be used to make posterior inference more tractable. Starting from the formula for the posterior covariance above and using the fact that $K_m = \mathbb{I}_d \otimes K$ and $\sum_i \mathbf{v}_i \mathbf{v}_i^T = \beta^{-1} \mathbb{I}_d$ with $\beta = {}^2\!/_N$, we obtain

$$\Sigma_{post}^{-1} = (\mathbb{I}_d \otimes K)^{-1} + \beta^{-1}\mathbb{I}_d \otimes \Sigma_\epsilon^- 1 \tag{40}$$

$$= \mathbb{I}_d \otimes (K^{-1} + \beta^{-1}\Sigma_\epsilon^- 1) \tag{41}$$

$$\implies \Sigma_{post} = \mathbb{I}_d \otimes (K^{-1} + \beta^{-1}\Sigma_\epsilon^- 1)^{-1} \tag{42}$$

Thus, inference for different map components is decoupled and the joint covariance need not be stored in memory (or be inverted).

We can now make use of the low-rank approximation of K by applying the Woodbury matrix identity

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1} \tag{43}$$

first with $U = \beta^{-1}, C = \Sigma_\epsilon^1, A = K^{-1}, V = \mathbb{I}_n$, which yields

$$(K^{-1} + \beta^{-1}\Sigma_\epsilon^- 1)^{-1} = K - \beta^{-1}K(\Sigma_\epsilon + \beta^{-1}K)^{-1}\mathbb{I}_n K \tag{44}$$

$$= K - \beta^{-1}K(\Sigma_\epsilon + \beta^{-1}GG^T)^{-1}K. \tag{45}$$

Applying the Woodbury identity a second time, this time with $A = \Sigma_\epsilon, U = G, C = \beta^{-1}\mathbb{I}_q, V = G^T$, we get

$$\Sigma_{post} = \mathbb{I}_d \otimes (K - \beta^{-1}K(\Sigma_\epsilon^{-1} - \Sigma_\epsilon^{-1}G(\beta^{-1}\mathbb{I}_q + G^T\Sigma_\epsilon^{-1}G)^{-1}G^T\Sigma_\epsilon^{-1})K) \tag{46}$$

References

Francis R. Bach and Michael I. Jordan. Kernel Independent Component Analysis. *Journal of Machine Learning Research*, 3:1–48, 2002. ISSN 0003-6951. doi: 10.1162/153244303768966085. URL `http://jmlr.org/papers/volume3/bach02a/bach02a.pdf`.

Jakob H. Macke, Sebastian Gerwinn, Leonard E. White, Matthias Kaschube, and Matthias Bethge. Gaussian process methods for estimating cortical maps. *NeuroImage*, 56(2):570–581, may 2011. ISSN 10538119. doi: 10.1016/j.neuroimage.2010.04.272. URL `http://linkinghub.elsevier.com/retrieve/pii/S1053811910007007`.