Notes on Distributed Distributional (Population) Code

Kevin W Li

March 12, 2018

This is my own notes on how to use DDPC to do computation.

Contents

1 Probablistic Population Code (PPC)

1.1 Encoding and decoding

In the popular probablistic population code, each neurons response is thought to encode one dimensional quantity x(e.g. orientation, frequency). If a population of neurons share the same tuning curves shape but evenly spaced preferred stimuli, then the rate is $r(x) = r(x - x_i)$, then the responses of this population to a single input is similar to this tuning curve, which is a "discrete convolution" between $r(x)$ and delta function $\delta(x - x^*)$. The observed rates will then be a noisy version of $r_i(x)$. The probablity of observing n_i is

$$P(n_i|x) = e^{-r_i(x)} \frac{[r_i(x)]^{n_i}}{n_i!} \tag{1}$$

Usually, tuning curves are assumed to be Gaussian

$$r_i(x) \propto \exp\left[-\frac{(x - x_i)^2}{2\sigma^2}\right] \tag{2}$$

Decoding using Bayes rule, we have

$$\begin{aligned} P(x|\mathbf{n}) &\propto \prod_i e^{-r_i(x)} \frac{[r_i(x)]^{n_i}}{n_i!} \\ \log P(x|\mathbf{n}) &= -\sum_i r_i(x) + \sum_i n_i \log r_i(x) + K \end{aligned} \tag{3}$$

It is common to make the assumption that the tuning functions $r_i(x)$ tile the space densely enough so that $\sum_i r_i(x)$ is roughly as constant

Therefore, equation (3) becomes

$$P(x|\mathbf{n}) \propto \exp\left(\sum n_i \log r_i(x)\right) \tag{4}$$

Assuming Gaussian tuning curve,

$$\begin{aligned} \log P(x|\mathbf{n}) &= \sum_i n_i \log[r_i(x)] + K \\ &= -\frac{1}{2\sigma^2} \sum_i n_i (x - x_i)^2 + K' \\ &= -\frac{1}{2} \frac{\sum_i n_i}{\sigma^2} \left(x - \frac{\sum_i n_i x_i}{\sum_i n_i}\right)^2 + K'' \\ x|\mathbf{n} &\sim \mathcal{N}\left(\frac{\sum_i n_i x_i}{\sum_i n_i}, \frac{\sigma^2}{\sum_i n_i}\right) \end{aligned} \tag{5}$$

Because the tuning curves are Gaussian, the posterior over stimulus location is a Gaussian distribution. A few observations and corresponding problems[3]:

1. *Multiplicity*: MAP of the stimulus is the mean which is the population average and is a single value (though this is consistent with assumption that we are encoding a single dimension as seen in (1)): **cannot decode into multimodal stimulus**

2. *Uncertainty*: The variance is strictly less than the width of the tuning curve: c**annot represent extrinsic stimulus uncertainty greater than** σ

1.2 Computation

Computations using PPC have been explored in [1]. In particular, consider the cue combination is a task in which a population of neurons integrate information from two other populations. Let the three population activities be $\mathbf{n_1}$,$\mathbf{n_2}$and $\mathbf{n_3}$. We would like the thrid population to encode the same posterior density jointly encoded by the first two:

$$p(x|\mathbf{n}_3) = p(x|\mathbf{n}_1, \mathbf{n}_2) \tag{6}$$

Assume that the prior $p(x)$ is flat and the first two populations activities are independent given a stimulus, then

$$p(x|\mathbf{n}_1, \mathbf{n}_2) \propto p(\mathbf{n}_1, \mathbf{n}_2|x) = p(\mathbf{n}_1|x)p(\mathbf{n}_2|x) \propto p(x|\mathbf{n}_1)p(x|\mathbf{n}_2) \tag{7}$$

If the three populations have the same tuning curves $f_{1i}(x) = f_{2i}(x) = f_{3i}(x) = f_i(x)$, then using equation (4) we have

$$p(x|\mathbf{n}_1, \mathbf{n}_2) \propto \exp\left(\sum_i (n_{1i} + n_{2i}) \log[f_i(x)]\right) \tag{8}$$

To satisfy equation (6), it can be concluded that the $\mathbf{n}_3 = \mathbf{n}_2 + \mathbf{n}_1$. Even though the this result is obtained by explicitly combining the the first two population activites, this result also satisfy (4) which can be obtained by running Bayes rules on population 3 directly. The critical part is constructing the **likelihood function** $p(\mathbf{n}|s)$ and **operations on activities** such that this operation is optimal in the sense that it is equivalent to operations in **posterior spaces** $p(s|\mathbf{n})$.

1.3 Exponential family

More generally, PPC is in fact a coding scheme where the likelihood (encoding) function is in the exponential family with linear sufficient statistics. Here, the sufficient statistics is the response vector $\mathbf{n}$, and the natural parameters is a (tuning) function of the stimulus $\mathbf{h}(x)$. Decoding corresponds to finding the MAP of the natural parameters in terms of x[1].

Consider the same cue combination task above. If the individual or joint likelihood function of the first two populations is in the exponential family with linear sufficient statistics, and we **relax the assumption** that the tuning functions are identical in the three populations

$$p(\mathbf{n}_1, \mathbf{n}_2|x) = \frac{1}{Z_{12}(x)} \eta(\mathbf{n}_1, \mathbf{n}_2) \exp(\mathbf{h}_1^\intercal(x)\mathbf{n}_1 + \mathbf{h}_2^\intercal(x)\mathbf{n}_2) \tag{9}$$

Again, assume a flat prior over x, the posterior over is

$$p(x|\mathbf{n}_1, \mathbf{n}_2) \propto \frac{1}{Z_{12}(x)} \exp(\mathbf{h}_1^\intercal(x)\mathbf{n}_1 + \mathbf{h}_2^\intercal(x)\mathbf{n}_2) \tag{10}$$

If we further assume that the tuning curves share a common basis $\mathbf{h}_1(s) = \mathbf{A}_1\mathbf{b}(s)$ and $\mathbf{h}_1(s) = \mathbf{A}_2\mathbf{b}(s)$, it can be shown [1] that the optimal response of the third population is

$$\mathbf{n}_3 = \mathbf{A}_1^\intercal\mathbf{n}_1 + \mathbf{A}_2^\intercal\mathbf{n}_2 \tag{11}$$

a linear weighted sum of the first two population responses

2 Distributional Population Code (DPC)

Due to the limitation of PPC, a second encoding scheme is proposed in [3]. The authors considered a general stimulus distribution $m(x)$ which is not necessary a delta function. Instead, it is a distribution of stimulus that over a potentially infinite dimensional space (the continuous domain of orientation or location). In this paper, a probability distribution of $m(s)$ is considered, and this is generalised to any distribution over s in a further work [2]

2.1 Encoding and decoding

The stimulus is not represented as a univariate random variable s; instead, a distribution $p(s)$ is used to indicate the presence of the stimulus along multiple (if $p(s)$ is a vector) or infinite (if $p(s)$ is a density) dimension of the stimuls. Here, dimensionality is defined over the space of stimlus. For example, it could be the direction of a sound source $[0, 2\pi)$.The value of $p(s)$ then tells that how likely is this stimlus present at orientation s. If the stimulus is the transparent motion along two directions, $p(s) = 0.5[\delta(s - s_1) + \delta(s - s_2)]$. In addition, extrinsic undertainty or noise can also be represented into this stimulus distribution as spread over those delta functions. However, it also introducds a problem because multiplicity is conflated with undertainty, as the spread of this distribution could be caused by either or both of the two features.

For a given stimulus *probability* distribution $m(s)$, the mean rate of a neuron is a inner product (convolution) between stimulus distribution $m(s)$ and some basis function $f_i(x)$ (could be tuning curve)

$$r_i = \int f_i(s)m(s)ds \tag{12}$$

Thus, each neuron code some aspect of the stimulus distribution. Decoding is a bit complicated because now we would like to find a distribution of $m(s)$ which is a potentially infinite dimensional quantity. To regularize, a prior over the values of $m(s)$ needs to be specified. The prior could be smoothness or max-entropy.

Decoding in this case works as before, albeit that the posterior is now $p(m|\mathbf{n})$. The domain of the stimulus is usually continuous, so this distribution is over functions of the stimulus. In practise, in order to fascilitate computation, $m(s)$ is discretised over stimulus domain into a vector $\mathbf{m}$ with each element being the value of $m(s)$ at some values (grid) of$s$, and $P(\mathbf{m})$ becomes a multivariate distribtuion. As an example, consider the transparent motion situation, $\mathbf{m}$ could a vector representing the probability of motion at each of the 360 degrees. **It is important to realise that the quantity we care about is $m(s)$ or m NOT s as the latter as become an argument of the function or index of the vector**.

The actual posterior follows the same form as 3 with the addition of a prior over $\mathbf{m}$ and a constant α that specifies the strength of this prior. The rate $r_i(x)$ is also replace by discretised version of the definition in 12. In addition, the term $\sum_i \mathbf{f_i^\intercal m}$ is still a constant due to translation invariance. So the log posterior is

$$\log P(\mathbf{m}|\mathbf{n}) = \alpha \log P(\mathbf{m}) + \sum_i n_i \log[\mathbf{f}_i^\intercal \mathbf{m}] - K \tag{13}$$

The prior $\log P(\mathbf{m})$ can be a smoothness $\sum_j (m_j - m_{j+1})^2$ or entropy $\sum_j m_j \log(m_j)$. Decoding falls into a special case of DDPC which is discussed below.

3 Distributed Distributional Probablistic Coding

3.1 Encoding

The major draw back of the DPC scheme is that it an only represent *either* multiplicity *or* undertainty. The interpretation of the probability distribution over s is more powerful than PPC but not rich enough to distinguish these two features of a stimulus. In [2], Sahani and Dayan introduced three modifications to DPC:

1. The distribution of stimulus $m(s)$ does not need to be a probability distribution. It can be any distribution/function and is itself random with density $p(m)$

2. Condition on one stimulus distribution m, the mean rate of a neuron $\psi_i(m)$ is an inner product of $m(s)$ and a function $f_i(s)$ passed through a nonlinearity $\sigma_i(\cdot)$,
3. The marginal mean rate of a neuron is then averaged over $p(m)$

In mathematical form, this means that the mean firing rate of a neuron is

$$r_i = \langle \psi_i(m) \rangle_{p(m)} = \left\langle \sigma_i \left(\int f_i(s) m(s) ds \right) \right\rangle_{p(m)} \tag{14}$$

where, $f(s)$ and $m(s)$ represents same quantities as in DPC. This way, multiplicity and uncertainty are separated into two components. $m(s)$ indicates the **intensity** or **magnitude** of stimulus at different points in the stimulus domain, $p(m)$ represents the **uncertainty** over many possible intensity configurations over function space. Again, in computations, m is usually discretised over stimulus domain to form a vector $\mathbf{m}$. Each neuron also as its own nonlinearity σ_i. There is evidence for multiple threshold found in the auditory system.

[1]

3.2 Why is DDPC powerful

Before decoding, let's try to see why this encoding scheme is so powerful. Each neuron not only has its basis tuning function $f_i(s)$ (possibly with different preferred centre), but also a nonlinearity σ_i that has a different threshold (usually ReLU, sigmoid or step). The neuron population hence tiles different regions in the stimulus domain and is sensitive to different levels of stimulus intensity. Cruicially, the expectation over m is taken after the nonlinearity. This ordering creates a overcomplete representation of the stimulus and enables DDPC to distinguish multiplicity and uncertainty.

For example, suppose we are trying to distinguish two types of stimulus:

- *Stimulus A*: A certain stimulus where the only possible distribution is $m_A(s) = 0.5\delta(s - s_1) + 0.5\delta(s - s_2)$
- *Stimulus B*: An uncertain stimulus distribution that takes on $m_{B1}(s) = \delta(s - s_1)$ or $m_{B2}(s) = \delta(s - s_2)$ with equal probability

If we average over uncertainty distribution $p(m)$, these two types of stimulus are exactly the same. If we extend the DPC scheme naively in which we encode each stimulus density $m(s)$ linearly and then take the average response over $p(m)$, it would be impossible to recover the difference between these two. So it should come naturally that there must be some sort of nonlinearity before averaging across stimulus distribution; otherwise, this average could equivalently happen before encoding $m(s)$.

But why do we need different nonlinearity for each neuron? It can be seen that the mean rate is only sensitive to stimulus intensity under each particular stimulus configuration, but not to the average strength. If a neuron has a step-threshold at 0.8, the it will have zero mean rate under the first case, but not the second. The discriminant threshold will be lower if the average strength at each s is lower (e.g. same two stimuli but with half the intensities). Therefore, **different thresholds are used to distinguish uncertainty and multiplicity at all average strength.**

3.3 Decoding

Decoding is slightly more complicated because now the likelihood is the activities given a *distribution* over m instead of a single m. Writing the distribution over m as $q(m)$, equation 3 should be written as below

[1] This formulation is in fact very similar to mean embedding in RKHS. Namely, using RKHS terminology, the firing rate of a neuron could be written as

$$r_i = \mathbb{E}_{p(s)} \langle f_i, \phi(s) \rangle_{\mathcal{H}} = \mathbb{E}_{p(s)} \langle f_i, \phi(s) \rangle_{\mathcal{H}} = \langle f_i, \mathbb{E}_{p(s)} \phi(s) \rangle_{\mathcal{H}} = \langle f_i, \mu(x) \rangle_{\mathcal{H}}$$

where $\phi(x)$ is the feature space representation of s in the RKHS $\mathcal{H}$, and f is a function in $\mathcal{H}$. The difference here with DDPC is that the mapping is linear and the stochasticity comes from x itself instead of the embedding function ϕ.

$$\begin{aligned}\log p(\mathbf{n}|q[m]) &= \sum_i \left[-r_i + n_i \log r_i\right] \\ &= \sum_i \left[-\int q(m)\psi_i(m)dm + n_i \log \int q(m)\psi_i(m)dm\right] \end{aligned} \tag{15}$$

Unlike in previous coding schemes, we cannot ignore the first term in DDPC because now translation invariance is violated by neurons having non-unifom nonlinearities. This is in contrast with PPC and DPC where the rate r is linear in the stimulus distribution. To uniquely identify the MAP, a prior over $p(q[m])$. One choice is to make the log prior proportional to the entropy of q.

$$\log p(q[m]) = \alpha H[q] + K(\alpha) = \alpha \int q(m) \log q(m) dm + K(\alpha) \tag{16}$$

For small α, this prior will select among many possible decoded distribution that will give the same likelihood, choosing the one with greatest uncertainty. Combining the two, together with the constraint that $q(m)$ has to be normalized, we obtain the log posterior $\mathcal{L}[q(m)] = \log p(q[m]|\mathbf{n})$

$$\mathcal{L}[q(m)] = \sum_i \left[-\int q(m)\psi_i(m)dm + n_i \log \int q(m)\psi_i(m)dm\right] + \alpha \int q(m) \log q(m) dm - \lambda \left[\int q(m)dm - 1\right] \tag{17}$$

To find the fixed point solution, one needs to take the variational derivative with respect to function $q(m)$ (assume this density is always positive)

$$\begin{aligned}\frac{\delta \mathcal{L}}{\delta q(m)} &= \sum_i \left[-\psi_i(m) + \frac{n_i}{r_i}\psi_i(m)\right] + \alpha(\log q(m) + 1) - \lambda \\ q(m) &\propto \exp\left[\frac{1}{\alpha}\sum_i \left(\frac{n_i}{r_i} - 1\right)\psi_i(m)\right] \end{aligned} \tag{18}$$

Since r_i depends on $q(m)$, this method is only iterative. The original authors used a different approach where the derivative is take w.r.t. $\log q(m)$[2]to explicitly impose positivity. To fascilitate computation, one has to discretize the stimulus domain s into a sufficiently dense grid (i.e.$0°, \pm 10°, \pm 20°, \dots$) so that m is replace with a vector $\mathbf{m}$. The distribution $q(\mathbf{m})$, which gives the probability density of $\mathbf{m}$ taking on any particular strength at the chosen grid points, can either be a histogram or a parameterized density (e.g. Gaussian Mixture).

3.4 Computation (on-going)

So far, for each of the three coding methods, the main purpose has been to decode from an encoding scheme. A more interesting question is to ask how, if neurons indeed respond to stimulus the way we think, various computations should be performed. In particular, let's focus on the stimulus domain discretised version when $\mathbf{m}$ is a vector instead of a function. One of the main computations is taking expectation of a function over stimulus with respect to the density of this stimulus

$$\mathbb{E}_{p(\mathbf{m})}\left[F(\mathbf{m}|\boldsymbol{\theta})\right] \tag{19}$$

where $F(\mathbf{m}|\boldsymbol{\theta})$ can be a function parameterised by $\boldsymbol{\theta}$ that has a vector argument $\mathbf{m}$. This can be any of the following commonly seen objects

- joint distribution $p(\mathbf{m})p(x|\mathbf{m})$
- the log joint probability $\log p(X = x, \mathbf{m}|\boldsymbol{\theta})$ where X is observed to be x and $\mathbf{m}$ is the latent variable

- State-action value function $Q(s, \mathbf{m}_s)$ where s is the state of an agent and $\mathbf{m}_s$ is a set of stochastic policy vector at state s

One insight from the decoding section is that, if we assume that the decoder is performing max-entropy posterior estimation, 18 suggests that the resulting distribution is in the exponential family with sufficient statistics $\psi_i(m)$. If the population activity is already the expected value of this quantity, they are then in fact representing the exp-fam distribution through mean parameters.

This suggests an approach to compute the quantity in 19. If $F(\mathbf{m}|\boldsymbol{\theta})$ can be approximated by a linear combination of the sufficient statistics, then this expectation is simply a linear readout of the neuron activities.

$$\mathbb{E}_{p(\mathbf{m})}\left[F(\mathbf{m}|\boldsymbol{\theta})\right] \approx \mathbb{E}_{p(\mathbf{m})} \sum_i \left[\alpha_i(\theta)\psi_i(\mathbf{m})\right] = \sum_i \left[\alpha_i(\theta)\mathbb{E}_{p(\mathbf{m})}\psi_i(\mathbf{m})\right] = \sum_i \left[\alpha_i(\theta) r_i(\mathbf{m})\right]$$

References

[1] J Beck, W J Ma, P E Latham, and A Pouget. Probabilistic population codes and the exponential family of distributions. In Trevor Drew Paul Cisek and John F Kalaska, editors, *Computational Neuroscience: Theoretical Insights into Brain Function*, volume 165 of *Progress in Brain Research*, pages 509–519. Elsevier, 2007.

[2] Maneesh Sahani and Peter Dayan. Doubly distributional population codes: Simultaneous representation of uncertainty and multiplicity. *Neural Comput.*, 15(10):2255–2279, October 2003.

[3] Richard S. Zemel, Peter Dayan, and Alexandre Pouget. Probabilistic Interpretation of Population Codes. *Neural Computation*, 10(2):403–430, 1998.