Probabilistic Population Codes

Richard Turner

These are personal notes I made on probabilistic population codes. Please send any errors to turner@gatsby.ucl.ac.uk

1 An introduction to inference and computation with population codes

As theoretical neuroscientists, there are several key questions concerning populations of neurons that we would like to answer. These include:

1. What do populations of neurons encode?
2. How is it encoded: the medium (rates, times, correlations) and the method (medium = f[stimulus])?
3. How can we (does the brain) decode the neurons?
4. How do populations support non-linear computations over the information they represent?
5. How are multiple aspects of the world represented in a single population?
6. What are the computational advantages of one scheme over another?

Currently there are two main working hypotheses that purport to answer the first of these questions: what do neural populations represent? The first (standard model) claims that populations encode the value of a stimulus. Whilst the second, more recent perspective, claims they encode a probability distribution over the possible values of a stimulus.

The standard model can be caricatured in the following manner: Firstly we specify an encoding rule from stimulus x to neural rate r_i. This will be a probabilistic mapping $P(r_i|x)$ due to neuronal noise. To decode[1] we form $P(x|\mathbf{r})$ and typically take a point estimate of the stimulus, one popular choice for which is the MAP estimate: $\hat{x} = \arg\max_x P(x|\mathbf{r})$. In summary then, the standard model typically only considers a single source of uncertainty (arising from noisy neural activities) and only decodes a point estimate from the posterior.

An example of this approach might be as follows: Let each neuron in the population have a bell shaped tuning curve:

$$\langle r_i \rangle = \exp\left[-\frac{1}{2\sigma^2}\left(x-\mu_i\right)^2\right] \quad (1)$$

[1] We should bare in mind that it is not necessary for the brain to decode itself. However, the process does indicate the power of a candidate code and what type of information it can convey.

The neural responses are Poisson with a mean set by the above:

$$P(\mathbf{n}|x) = \prod_i \frac{1}{n_i!}(T\langle r_i\rangle)^{n_i}\exp\left[-T\langle r_i\rangle\right] \quad (2)$$

The posterior distribution when we have flat priors (see later for a derivation) is:

$$P(x|\mathbf{n}) = \mathrm{Norm}\left(\frac{\sum_i \mu_i n_i}{\sum_i n_i}, \frac{\sigma^2}{\sum_i n_i}\right) \quad (3)$$

The MAP estimate for the stimulus is given by a weighted average $\hat{x} = \frac{\sum_i \mu_i n_i}{\sum_i n_i}$. Notice that our uncertainty in the stimulus decreases as we increase the number of neurons. In the limit $I \to \infty$ the posterior becomes a delta function.

The second approach, which claims neurons encode probability distributions, generalises the first in two ways. Firstly it notes that the physical environment can be noisy too, and furthermore problems are often ill posed and therefore need to be solved probabilistically using prior information. These two types of uncertainty are quite different from that caused by neuronal noise (for example, the later reduces as we increase the number of neurons, whereas the former should not), and ideally they too should be encoded (you *believe* the person sitting next to you in the darkened room is your partner, but you'd like to be *sure* before you kiss them).

The encoding rule (the mapping from stimuli to rates) now has two stages. In the first the stimulus or latent variable, x (say the orientation of a bar) causes a particular response in sensory receptors $\mathbf{y}$ (say the photoreceptors). The mapping $x \to \mathbf{y}$ might be probabilistic $P(\mathbf{y}|x)$ due to the physics of the environment. The goal of downstream neurons is to encode (some approximation[2] to) the recognition distribution $q(x|\mathbf{y})$. This can be probabilistic even if $p(\mathbf{y}|x)$ is not due to the possible ill posedness of the problem. Next we have to specify some method for encoding (the approximation to) the recognition distribution in neural rates. As with the equivalent step in the standard model, this will be probabilistic due to neuronal noise: $q(x|\mathbf{y}) \to \mathbf{r}$ and accordingly the encoding model has to specify $p[\mathbf{r}|q(x|\mathbf{y})]$.

Of course, having gone into all this trouble to encode more than just a point estimate of the stimulus in the neural rates $\mathbf{r}$, it is a waste to decode just one point estimate of the stimulus. The second generalisation of the standard model is to keep around the full posterior distribution: $P[q(x|y)|r]$, from which we can read off the most likely $\hat{q}(x|y) = \arg\max_q P[q(x|y)|r]$, say, or carry out other computations.

1.1 Evidence that distributions are encoded

The probabilistic view of population codes has become more pervasive as suggestive experimental evidence has accumulated. Two salient examples of which are: 1) Integration of cues from different sensory modalities with different uncertainties is carried

[2]There are lots of interrelated reasons why this might be an approximation: 1. There are lots of latent variables in the world and you just want to keep around a summary of them all (e.g. random dot kinematograms), 2. recognition and learning are made more tractable by the approximations, 3. The brain has limited hardware and limited representational power...

out in a Bayes optimal way. 2) Moving gratings appear to move faster when contrast is dialled up (see Fig. 1.). This is consistent with the uncertainty story as we will now explain: Imagine the prior distribution over speeds is broad and centred on zero (slow speeds are more common in natural scenes). The likelihood function, which is centred on a non-zero speed, narrows as the contrast goes up. For low contrasts the prior dominates and the perceived speed is slower, as contrast increases the likelihood has more of an effect (and eventually dominates). Therefore the perceived speed grows.

1.2 Format of these notes

These notes exclusively review papers that use the probabilistic framework. In particular, the work on distributional population codes (DPCs) proposes a rule for encoding probability distributions (recognition distributions) i.e. it specifies a probabilistic mapping: $q(x|\mathbf{y}) \rightarrow \mathbf{r}$, and shows how to decode the resulting population vector. The work on doubly distributional population codes shows how to generalise the framework to code distributions over multiple latent variables $q(\mathbf{x}|\mathbf{y})$ into a population, and to decode without confounding multiplicity and uncertainty. The final piece of work is interested in implementing simple computations upon the probability distributions. The approach is to specify a simple network implementation and derive the distributions for which this implementation carries out the desired computation optimally.

2 Distributional Population Codes (Zemel, Dayan, Pouget, 1998)

Briefly, the approach of the paper is as follows: Firstly the authors propose a new encoding rule that enables neurons to represent potentially complicated distributions. They then decode, to assess the power of the candidate code.

2.1 Encoding Rule

We restrict ourselves to the case where the distribution we encode into I neurons is over a single scalar quantity, x.

$$\langle r_i \rangle = \int q(x|y,\theta) f_i(x) dx \tag{4}$$

$$P(n_i|\langle r_i \rangle) = \text{Poisson}(T\langle r_i \rangle) \tag{5}$$

$$= \frac{(T\langle r_i \rangle)^{n_i}}{n_i!} \exp(-T\langle r_i \rangle) \tag{6}$$

This is an example of an *expected value code* (the average neural firing rates are given by expectations of non-linear functions over the recognition distribution[3]). Such

[3]Of course the brain has to learn this mapping and one suggestion is that it uses Hebbian learning. In such a scheme a "teacher signal" derived from some other information (e.g. another sensory system)

codes are a generalisation of the typical encoding rule: if the posterior distribution is a delta function: $q(x|y,\theta) = \delta(x - x_0)$ then $\langle r_i \rangle = f_i(x_0)$ and therefore $f_i(x_0)$ have the interpretation as the tuning functions of neurons found if we probe with unambiguous stimuli. For this reason a typical form of $f_i(x)$ is a bell shaped function of the stimulus.

2.2 Decoding Rule

To assess the power of a candidate code it is useful to try and decode it. To decode the above, we need to form the posterior distribution over recognition distributions $q(x|y,\theta)$. The vector of counts is denoted: $\mathbf{n} = \{n_i\}_{i=1}^I$

$$P[q(x|y,\theta)|\mathbf{n}] = \frac{1}{Z} P[\{n_i\}_{i=1}^I | q(x|y,\theta)] P[q(x|y,\theta)] \tag{7}$$

This distribution over distributions is complex, and we have to simplify matters to proceed. One hope is that it might be strongly peaked around $q(x|y,\theta)$ (which will be the case if the neuronal noise is low) in which case it would be enlightening to find its mode. Differentiating the log posterior (including a Lagrange multiplier to ensure normalisation) we have:

$$L = K_1 + \log P[\mathbf{n}|q(x|y,\theta)] + \log P[q(x|y,\theta)] + \lambda \left[\int q(x|y,\theta) dx - 1 \right] \tag{8}$$

where

$$\log P[\mathbf{n}|q(x|y,\theta)] = \sum_i [n_i \log(T\langle r_i \rangle) - T\langle r_i \rangle - \log(n_i!)] \tag{9}$$

The functional dependence on $q(x|y,\theta)$ only enters through the $\langle r_i \rangle$ so we have:

$$\begin{aligned} L = K_2 + \sum_i \left[n_i \log \left(\int dx f_i(x) q(x|y,\theta) \right) - T \int dx f_i(x) q(x|y,\theta) \right] \\ + \log P[q(x|y,\theta)] + \lambda \left[\int q(x|y,\theta) dx - 1 \right] \end{aligned} \tag{10}$$

Finally we can specify a prior to proceed. One choice is favour those distributions with more uncertainty. Such a prior, depending on the strength, will select the distribution with the most uncertainty from a set with the same likelihoods[4]:

tells the neuron at what rate it should be firing for current input $\mathbf{y}$. If the teacher was derived from a sample from the recognition distribution $\langle \tilde{r} \rangle = f(x_n)$ where $x_n \sum q(x_n|\mathbf{y})$ then Hebbian learning would cause the average of the outputs to match the average of the teaching signal, after learning.

[4] Another choice for the prior, might be one favouring smooth distributions

$$P[q(x|y,\theta)] = \frac{1}{Z(\alpha)} \exp(\alpha H[q(x|y,\theta)]) \quad (11)$$

Thus we have:

$$\begin{aligned} L &= K_3 + \sum_i \left[n_i \log \left(\int dx f_i(x) q(x|y,\theta) \right) - T \int dx f_i(x) q(x|y,\theta) \right] \\ &\quad -\alpha \int dx q(x|y,\theta) \log q(x|y,\theta) + \lambda \left[\int q(x|y,\theta) dx - 1 \right] \end{aligned} \quad (12)$$

Which can be differentiated:

$$\frac{\delta L}{\delta \log q(x|y,\theta)} = q(x|y,\theta) \left[\sum_i \left(\frac{n_i}{\langle r_i \rangle} f_i(x) - T f_i(x) \right) - \alpha [\log q(x|y,\theta) + 1] + \lambda \right] \quad (13)$$

From which we get the fixed point updates:

$$q(x|y,\theta) = \frac{q(x|y,\theta)}{T \sum_i f_i(x)} \left[\sum_i \frac{n_i}{\langle r_i \rangle} f_i(x) - \alpha [\log q(x|y,\theta) + 1] + \lambda \right] \quad (14)$$

The normalisation λ has to be recalculated after each iteration.

In reality, this method has to be implemented using a discrete histogram approximation to $q(x|y)$. For high dimensional distributions $q(\mathbf{x}|\mathbf{y},\theta)$ this method is infeasible as the number of entries in the histogram scales exponentially with dimension. In which case we can choose a parametric approximation to $q(\mathbf{x}|\mathbf{y})$ with a number of parameters that is independent of the dimensionality of the encoded distribution e.g. a mixture of isotropic Gaussians.

2.3 An Example (Zemel and Dayan, 1999)

When two patterns slide across one another the perception is of two surfaces sliding across each other in different directions. A neuro-physiological finding is that the response of a cell is the average of its response to the individual components, where the response to an individual component is Gaussian plus some base line activity.

In the terminology of the above formalism we have:

$$q(\theta|y) = \frac{1}{2} [\delta(\theta - \theta_1) + \delta(\theta - \theta_2)] \quad (15)$$

$$f_i(\theta) = b_i + a_i \exp\left(-\frac{1}{\sigma^2} (\theta - \theta_i)^2 \right) \quad (16)$$

b_i, a_i and σ are set to match the individual responses. Choosing a prior $P(q(\theta))$ which favours smooth distributions the DPC framework then matches the experimental results (see Fig. 2):

1. for $\Delta\theta \geq 30$ the cell population response r_i is bimodal, but for $\Delta\theta \leq 30$ it is unimodal

2. for $\Delta\theta \geq 10$ the decoded distribution $q(\theta|y)$ is bimodal, but for $\Delta\theta \leq 10$ it is unimodal (matching the psychological threshold)

3. by judiciously choosing the motions, it is possible to increase the number of component patterns above two, whilst maintaining an identical population response. Naturally, the decoded distribution remains identical and just has two modes. Psychophysically, the multi-component patterns are perceived as having only two components.

2.4 Doubly distributional population codes (Sahani and Dayan, 2003)

As we will now explain through a number of examples, DPCs can conflate multiplicity (multiple latent variables) with uncertainty (in one latent variable). They can code one or the other, but not both. DDPCs address this issue.

2.5 Multiplicity and uncertainty through concrete examples

To contrast multiplicity and uncertainty let's return to our example with two fields of moving dots that are sliding over the top of one another. In the DPC work we claimed that neurons are encoding this by representing a distribution over angles: $P(\theta) = \frac{1}{2}[\delta(\theta-\theta_1)+\delta(\theta-\theta_2)]$. This is an example of *multiplicity* as there are two latent variables present. Imagine now that the dots are perturbed around their smooth motion by some noise, this would introduce *uncertainty* too. In the spirit of the above framework we could represent this by a mixture of two Gaussians, with the width of the Gaussian indicating our uncertainty in the direction of movement: $P(\theta) = \frac{1}{2}[\mathrm{Norm}(\theta_1, \sigma_\theta^2) + \mathrm{Norm}(\theta_2, \sigma_\theta^2)]$. Finally, imagine that the random perturbation is anisotropic - perhaps it is greatest at $\pm 5^\circ$ to the direction of motion. Using the representation consistently this uncertainty in the direction of motion looks like multiplicity: $P(\theta) = \frac{1}{2}[\frac{1}{2}\left(\mathrm{Norm}(\theta_1+\delta, \sigma_\theta^2) + \mathrm{Norm}(\theta_1-\delta, \sigma_\theta^2)\right) + \frac{1}{2}\left(\mathrm{Norm}(\theta_2+\delta, \sigma_\theta^2) + \mathrm{Norm}(\theta_2-\delta, \sigma_\theta^2)\right)]$

Notice that there are two steps here: What you actually observe is a whole bunch of dots moving in different directions. It seems unlikely that neurons encode the full posterior distribution over numbers of dots and directions (and positions too)[5]: $P(N, \theta_{1:N}|y)$ and so we need to propose an alternative, flexible but simpler form. The representation above is certainly simpler and essentially it amounts to encoding the probability distribution over the direction of motion when a dot is picked at random. However, it is not suitable as it conflates multiplicity and uncertainty. Choosing a sensible reduced representation is the first step of DPC. The second step is to specify how the rates of neurons encode this representation.

Here's a less contrived auditory example: inference of source location from interaural phase difference Φ is an ill posed problem. Many different locations corresponding

[5] Bayesian ideal observers fail to match psychophysical results on such tasks for this reason (Lu and Yuille, 2005)

to path differences that result in an indistinguishable phase difference of $\Phi + 2\pi n$. Inter-aural phase differences lead to a mixture of deltas over location. Under the DPC representation this looks like multiplicity i.e. more than one source is present. Additionally, uncertainty might exist as to the presence of a source. DPC does not have sufficiently rich representational power to code unambiguously for these situations.

2.6 A representation for multiplicity and uncertainty

Sahani and Dayan extend the DPC framework by introducing a distribution over multiplicity functions, which themselves are functions (distributions) over latent variables (hence the term doubly distributional). Again this is best illustrated by example (see Fig. 3 panels b. and c.). Going back to the auditory example, let's imagine two situations. In the first there are two sources (multiplicity), one located at $\hat{\theta}_1$ and another at $\hat{\theta}_2$. In the second example there is only one source but we are not sure whether it is located at $\hat{\theta}_1$ or $\hat{\theta}_2$ (uncertainty). The first example amounts to a single 'hypothesis' that is a joint distribution $P(\theta_1, \theta_2) = \delta(\theta_1 - \hat{\theta}_1)\delta(\theta_2 - \hat{\theta}_2)$. The second to two mutually exclusive, equi-probable hypotheses: $P(\theta) = \delta(\hat{\theta}_1 - \theta)$ or $P(\theta) = \delta(\hat{\theta}_2 - \theta)$.

The two situations involve probability distributions over spaces with different dimension and one of the contributions of DDPCs is to provide a flexible enough representation to encode both objects.

The trick is to assign each hypothesis a multiplicity function. This is best thought of (albeit loosely) as "what you'd end up seeing, if one of the hypotheses was true". The ms are functions, they can be probability distributions, but they need not be (and therefore the term "doubly distributional" can be confusing).

For the first case above if we had a "direction detector" we would expect it to read: $m(\theta) = \frac{1}{2}\delta(\theta - \hat{\theta}_1) + \frac{1}{2}\delta(\theta - \hat{\theta}_2)$, with $P(m) = 1$ as there is only one hypothesis. For the second if the first hypothesis is true our detector would read $m(\theta|H_1) = \delta(\theta_1 - \theta)$ and for the second $m(\theta|H_2) = \delta(\theta_2 - \theta)$. These hypotheses are equally probable, so $P(m(\theta|H_i)) = \frac{1}{2}$.

This representation is more flexible than the DPC preventing the conflation of multiplicity and uncertainty (see Fig. 3). Furthermore, the ms need not be normalised, so we can represent the hypothesis that no sources are present by giving some probability to $m(\theta) = 0$ (as our detector would not read anything under this hypothesis).

Finally, to make things really concrete, imagine encoding another of our motivating examples (the noisy version of case 1 above): $P(\theta_1, \theta_2) = \text{Norm}(\hat{\theta}, \sigma^2 I)$ i.e. there are definitely two sources, but we don't know exactly where. Under the DDPC setup a single Gaussian distribution should be regarded as an infinite number of hypotheses, and for each hypothesis an angle detector would show a single delta function. In the more complex case here, where we have two sources with Gaussian uncertainty, each hypothesis must correspond to multiplicity function which is a pair of deltas. Just as with the single Gaussian, the uncertainty is captured by $P(m)$ which weights each possible hypothesis in such a way that the Gaussian uncertainty is represented.

2.7 Encoding the new representation into a doubly distributional code

In a DDPC the representation is encoded in a similar manner to a DPC. The intensity function of a Poisson process being given by:

$$\langle r_i \rangle = \left\langle f_i \left(\int dx g_i(x) m(x) \right) \right\rangle_{p(m)} \tag{17}$$

where $g_i(x)$ is a linear response function, and $f_i(.)$ is a static non-linearity. When there is no multiplicity $m(x) = \delta(x - x_0)$, this reduces to the DPC. Additionally, when there is no uncertainty in the function $m(x)$ we recover the standard encoding model. Fig 3. illustrates the result of such an encoding.

2.8 Decoding doubly distributional codes

To decode we form $P\left(q\left[m(x)\right]\right)$, that is a distribution $P(.)$ over distributions $q(.)$ over distributions $m(.)$. Despite sounding complicated, we can find the MAP value of q in exactly the same way as for DPC (after replacing $m(x)$ by a suitable discrete approximation). This leads to an almost identical set of fixed-point updates. The difference being that the distribution q is over a vector, rather than a scalar:

$$q(\mathbf{m}) = \frac{q(\mathbf{m})}{T \sum_i f_i(\mathbf{m})} \left[\sum_i \frac{n_i}{\langle r_i \rangle} f_i(\mathbf{m}) - \alpha[\log q(\mathbf{m}) + 1] + \lambda \right] \tag{18}$$

3 Bayesian inference with probabilistic population codes (Pouget, Latham, Beck, and Ma; 2006)

3.1 Prologue

In the (D)DPC we talked about the following mapping: $x \rightarrow y \rightarrow Q(x|y) \rightarrow r$. The first mapping is probabilistic due to noise in the physics of the environment and the ill posed nature of the problem. The last mapping is probabilistic due to neuronal noise. From the perspective of an experimenter this is just a stochastic mapping from latent variable to neural rates: $x \rightarrow r$ that can be characterised by the distribution $P(r|x)$. This subsumes both noise in the outside world in $P(y|x)$ and neural noise $P[r|q(x|y)]$) and therefore such codes are termed *implicit*. One way to think about this is that DPCs specify an (infinite) mixture model over neural rates. $P(y|x)$ are the mixing proportions and $P[r|q(x|y)]$ are the mixture components. The important contribution of DPC is that now information about the whole of the distribution $q(x|y)$ is injected into $P(r|x)$ (and not just the peak). In the DPC paper, $P[q(x|y)|r]$ is decoded, which partitions our neural uncertainty $P[.]$ and physical uncertainty $Q(x|y)$. Of course, we could use a more direct approach and specify $P(x|r)$ directly. This is the formalism of the paper. Care is taken to ensure uncertainty information (the width of $q(x|\mathbf{x})$) is encoded and therefore the formalism defines an *implicit* encoding of $q(x|\mathbf{x})$.

3.2 Bayesian inference

One core idea from this paper[6] is that knowing 1) the computation that a region of the brain is responsible for, and 2) the biophysical constraints on building networks, together should give you some leverage toward understanding the neural representation of a stimulus and its uncertainty. Towards this end the authors of this paper would like to solve the following problem: You give them a computation you'd like to perform using probability distributions and a proposed (simple) network implementation of that computation, and they'll give you back an encoding rule $P(\mathbf{r}|x)$ (for which the network implementation you proposed does the computation, optimally).

Of course, they don't solve this problem in it's full generality, but they consider a simple example where the computation is multiplication of two distributions: $P(x|\mathbf{r}_1, \mathbf{r}_2) = P(x|\mathbf{r}_1)P(\mathbf{x}|\mathbf{r}_2)$, the proposed implementation is addition of two population's firing rates: $\mathbf{r}_3 = \mathbf{r}_1 + \mathbf{r}_2$ and the family of distributions for which this is optimal is the exponential family.

Other than its simplicity, why this is an interesting example to pick? Well, one of the simplest tasks the brain might be interested in is combining information about a latent variable from two different sensory modalities (haptic and visual, say) in order to estimate the latent variable. That is computing $P(x|\mathbf{r}_1, \mathbf{r}_2)$ from $P(x|\mathbf{r}_1)$ and $P(x|\mathbf{r}_2)$. If the noise in the two estimates is independent then, $P(x|\mathbf{r}_1, \mathbf{r}_2) \propto P(x|\mathbf{r}_1)P(x|\mathbf{r}_2)$.

To implement this computation in a network we have to wire the two populations representing $P(x|\mathbf{r}_1)$ and $P(x|\mathbf{r}_r)$ to a new population representing $P(x|\mathbf{r}_1, \mathbf{r}_2)$. This would be easy to implement if we just had to add the rates of the neurons. Processing by higher levels might also be easier if the representation of the stimulus didn't change. Mathematically:

$$P(x|\mathbf{r}_1, \mathbf{r}_2) \propto P(x|\mathbf{r}_1)P(x|\mathbf{r}_2) \propto P(x|\mathbf{r}_1 + \mathbf{r}_2) \tag{19}$$

Likelihoods which satisfy this property are those belonging to the exponential family:

$$P(\mathbf{r}|x) \quad = \quad \Phi(\mathbf{r})\Psi(\mathbf{h}(x)) \exp\left[\mathbf{h}(x)^T\mathbf{r}\right] \tag{20}$$

In other words, the interaction between the data and the parameters is log-linear. A complementary (subset of this) idea is that neurons code for log-probabilities.

3.2.1 An Example

Let's go through a concrete example:

Let the tuning curves of the neurons $i = 1 : I$ in the two populations $j = 1, 2$ be bell-shaped with the same width:

[6] In the following I'm speaking in more grandiose terms than they could get away with. The more cautious take on their work is "if the noise in spike trains has a specific form, then adding two population vectors carries out multiplication of probability distributions optimally.

$$\langle r_{i,j} \rangle = g_j \exp\left[-\frac{1}{2\sigma^2}(x - \mu_{i,j})^2\right] \quad (21)$$

Both the gains g_j and the latents x depend on the stimulus. For example x might be the position of an object in an image and the gain related to the contrast of the image. Intuitively, we expect the (inverse of the) gain to control our certainty in the latent variable x (equivalently, the width of posterior distribution over x).

The neural responses are Poisson with a mean set by the above:

$$P(\mathbf{n}_j|x, g_j) = \prod_i \frac{1}{n_{i,j}!}(T\langle r_{i,j}\rangle)^{n_{i,j}} \exp\left[-T\langle r_{i,j}\rangle\right] \quad (22)$$

The posterior distribution of everything we don't know given everything we know is:

$$P(x, g_1, g_2|\mathbf{n}_1, \mathbf{n}_2) = \frac{1}{Z_1}P(x)\prod_j P(g_j)\prod_i \frac{1}{n_{i,j}!}(T\langle r_{i,j}\rangle)^{n_{i,j}} \exp\left[-T\langle r_{i,j}\rangle\right] \quad (23)$$

Assuming a flat prior over the latent variable $P(x)$ we have:

$$P(x, g_1, g_2|\mathbf{n}_1, \mathbf{n}_2) = \frac{1}{Z_2}\prod_j P(g_j)\exp\left[\sum_i n_{i,j}\log(T\langle r_{i,j}\rangle) - T\langle r_{i,j}\rangle\right] \quad (24)$$

$$= \frac{1}{Z_3}\prod_j P(g_j)\exp\left[\sum_i n_{i,j}\left(\log g_j - \frac{1}{2\sigma^2}(x - \mu_i)^2\right)\right] \quad (25)$$

Where we have absorbed terms that don't depend on x, g_1 or g_2 into the normalising constant, and used the fact that the tuning curves are uniformly and densely distributed to note $\sum_i n_{i,j}\langle r_{i,j}\rangle$ is independent of the stimulus. Integrating out the gains is simple as the integrand is of the form $i(x, g) = f_1(g)f_2(x)$. The integral goes into the normalising constant leaving:

$$P(x|\mathbf{n}_1, \mathbf{n}_2) = \frac{1}{Z_4}\exp\left[\sum_i\sum_j n_{i,j}\frac{1}{2\sigma^2}(x - \mu_i)^2\right] \quad (26)$$

$$= \frac{1}{Z_5}\exp\left[s^2\frac{1}{2\sigma^2}\sum_i\sum_j n_{i,j} + \frac{s}{\sigma^2}\sum_i\sum_j n_{i,j}\mu_i\right] \quad (27)$$

$$= \text{Norm}\left(\frac{\sum_i \mu_i \sum_j n_{i,j}}{\sum_i\sum_j n_{i,j}}, \frac{\sigma^2}{\sum_i\sum_j n_{i,j}}\right) \quad (28)$$

Due to the Poisson noise, posterior has the same functional form as the tuning curves: it is Gaussian. The mean is given by a weighted average of the receptive field

centres and the variance is proportional to the width of the tuning curves divided by the sum of the counts.

The counts n_j are proportional to the gain g_j. Therefore the posterior variance is proportional to $(\sum_j g_j)^{-1}$. This makes sense as precisions add and the individual populations imply posteriors with precisions proportional to g_j.

3.2.2 Relating tuning functions to natural parameters and neural 'noise'

By definition the tuning curves are the average response of a neuron to a stimulus $\{x, g\}$:

$$f_i(x, g) = \int d\mathbf{r} P(\mathbf{r}|x) r_i \tag{29}$$

Where, to recap, we are averaging over:

$$P(\mathbf{r}|x) = \frac{1}{Z} \exp\left(\sum_i h_i(x) r_i\right) \tag{30}$$

$$Z = \int d\mathbf{r} \exp\left(\sum_i h_i(x) r_i\right) \tag{31}$$

Differentiating eq. 29, we have:

$$\frac{df_i(x, g)}{dx} = \int d\mathbf{r} \frac{dP(\mathbf{r}|x)}{dx} \tag{32}$$

This can be rewritten using a trick:

$$\frac{d \log P(\mathbf{r}|x)}{dx} = \frac{1}{P(\mathbf{r}|x)} \frac{dP(\mathbf{r}|x)}{dx} \tag{33}$$

$$= \sum_j \frac{dh_j(x)}{dx} [r_j - \langle r_j \rangle] \tag{34}$$

Substituting this relation in, yields:

$$\frac{df_i(x, g)}{dx} = \int d\mathbf{r} \frac{d \log P(\mathbf{r}|x)}{dx} P(\mathbf{r}|x) \tag{35}$$

$$= \int d\mathbf{r} r_i \sum_j \frac{dh_j(x)}{dx} [r_j - \langle r_j \rangle] P(\mathbf{r}|x) \tag{36}$$

$$= \sum_j \sigma_{i,j} \frac{dh_j(x)}{dx} \tag{37}$$

So the derivative of the natural parameter vector with respect to the stimulus is equal to the inverse covariance of the rates multiplied by the derivative of the tuning functions:

$$\mathbf{h}' = \Sigma^{-1}\mathbf{f}' \tag{38}$$

$\mathbf{h}'$ has to be independent of the gain if linear addition of the population vectors is optimal, but we know $\mathbf{f}'$ is proportional to the gain g. This means that the covariance of the rates must be proportional to the gain. The mean rates are also proportional to the gain - so the theory predicts a fano factor which is independent of T. Roughly speaking, this is observed over a reasonable range. However, little is known about the correlations between neurons and specifically whether correlations between them scale with the gain.

3.3 Relaxing these conditions

We can handle priors easily by setting up a population for which: $P(x) = P(x|\mathbf{r}_3)$. Base-line activity in a population essentially encodes the prior.

The tuning curves can be different for the two populations so long as they are linearly related (more generally, if they are basis sets $\mathbf{h}_j(x) = A_j\mathbf{b}$). Then we must combine the two populations linearly: $\mathbf{r}_3 = A_1\mathbf{r}_1 + A_2\mathbf{r}_2$ where the coefficients A_i depend on the tuning curve strengths.

A nice feature of the setup is that integration of information through time can be achieved by repeatedly adding rates. Saturation can be prevented by divisive normalisation. In the next section we look at such an example.

3.4 An example

Imagine that the latent variable we are coding for is correlated through time, and we want to continually update the representation of the variable. How might we do this? If everything is linear and Gaussian, this amounts to implementing the Kalman filter:

$$\begin{aligned}
p(x_t|\mathbf{r}_{1:t}) &= \int dx_{t-1} p(x_t, x_{t-1}|\mathbf{r}_{1:t}) && (39)\\
&= \frac{1}{p(\mathbf{r}_t)} \int dx_{t-1} p(x_t, x_{t-1}, \mathbf{r}_t|\mathbf{r}_{1:t-1}) && (40)\\
&= \frac{1}{Z} \int dx_{t-1} p(x_t|x_{t-1}) p(\mathbf{r}_t|x_t) p(x_{t-1}|\mathbf{r}_{1:t-1}) && (41)\\
&= \frac{1}{Z} p(\mathbf{r}_t|x_t)\, \mathrm{Norm}(\lambda\langle x_{t-1}\rangle, \lambda^2\sigma_{t-1}^2 + \sigma^2) && (42)
\end{aligned}$$

In words, to form the posterior we take the distribution from the previous time step $P(x_{t-1}|\mathbf{r}_{1:t-1}) = \mathrm{Norm}(\langle x_{t-1}\rangle, \sigma_{t-1}^2)$ drift it towards zero (by $1 - \lambda\langle x_{t-1}\rangle$) and diffuse it. In terms of the neural representation, this amounts to shifting the population vector

to drift, and reducing the gain by divisive normalisation to diffuse. We then combine this with a population representing $p(\mathbf{r}_t|x_t)$ by adding the rates in the two populations.

3.5 References

Lu Y, Yuille A. (2005) Ideal Observers for Detecting Motion: Correspondence Noise. Advances in Neural Information Processing Systems

Ma W, Beck J, Latham P, and Pouget A. (2006) Bayesian inference with probabilistic population codes, to be published (nature neuroscience?)

Pouget A, Dayan P, Zemel R. (2003) Inference and Computation with population codes Annu. Rev. Neurosci. 26:381-410

Sahani M, Peter D. (2003) Doubly Distributional Population Codes: Simultaneous Representation of Uncertainty and Multiplicity. Neural Computation, Vol. 15, Issue 10

Zemel R and Dayan P. (1999) Distributional Population Codes and Multiple Motion Models. Advances in Neural Information Processing Systems 11, MIT Press

Zemel R, Dayan P, Pouget A. (1998) Probabilistic interpretation of population code. Neural Comput. 10:403-30

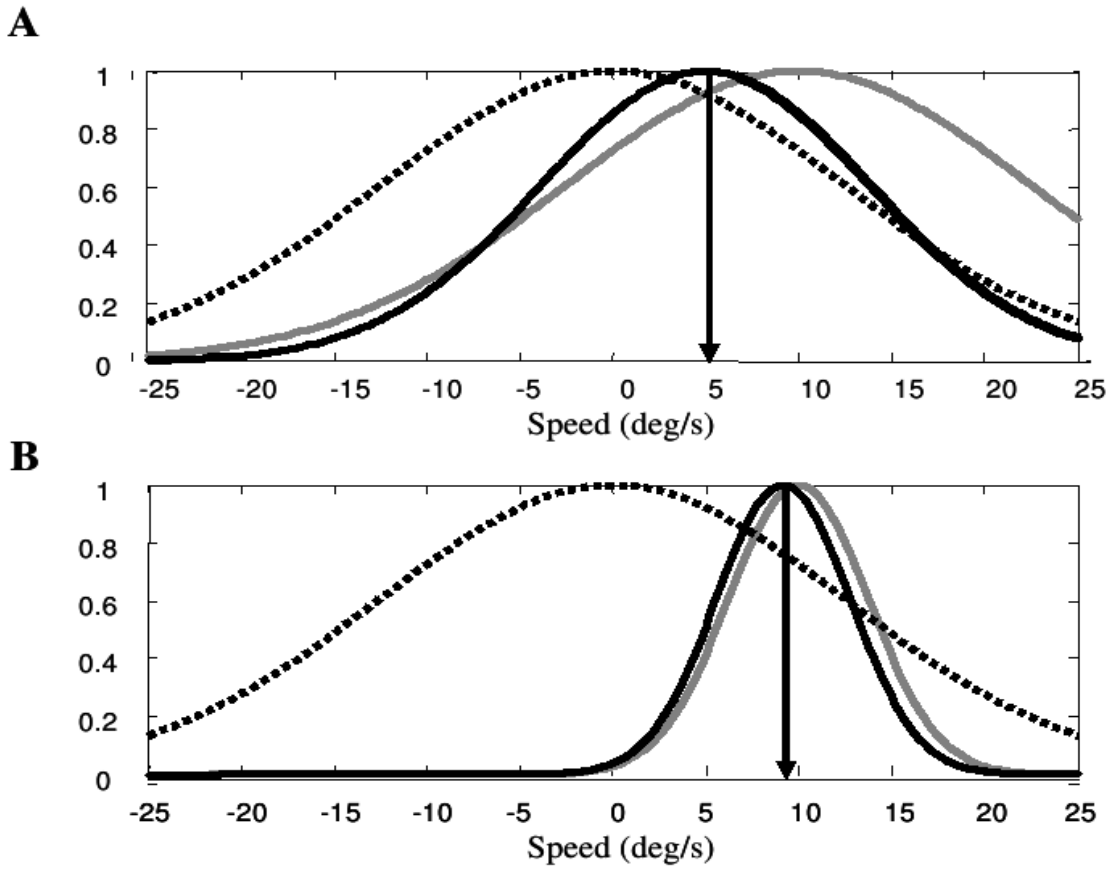

Figure 1: Dotted line = prior, grey line = likelihood, black line = posterior. Top: Low contrast, the prior is as strong as the likelihood and the perceived speed is low. Bottom: High contrast, the likelihood dominates the prior and the perceived speed is high. (Notice the argument is a little more subtle than you might think as the centre point of the likelihood changes on each trial.)

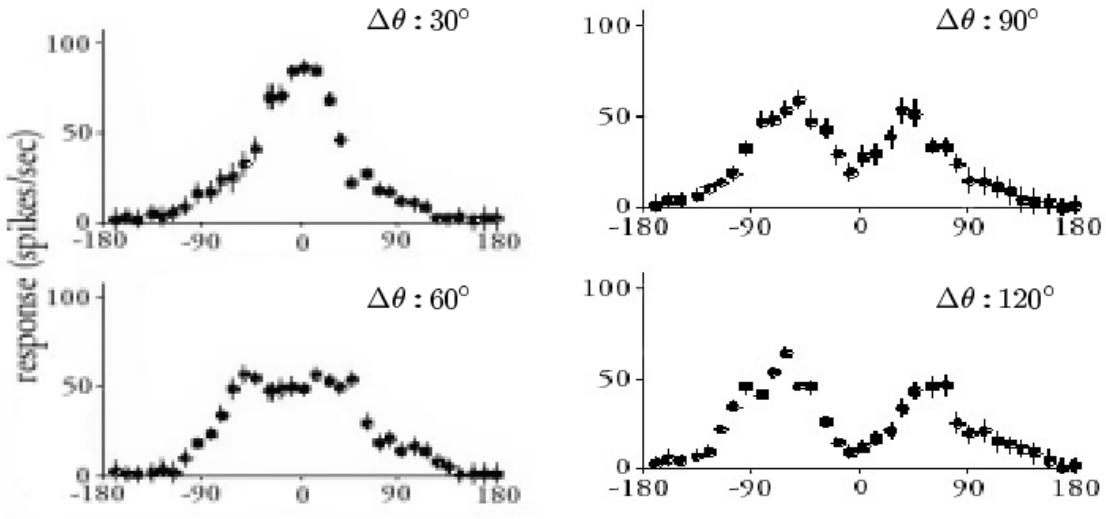

Figure 2: Left column: the population responses for patterns of dots moving in directions separated by $\Delta\theta$. Right: the decoded MAP distribution over directions. Notice how the population vector can be unimodal, whilst the decoded distribution is bimodal. At a separation of $\Delta = 10^\circ$ the decoded distribution tends to become unimodal and this corresponds to the point where chance is reached psychophysically.

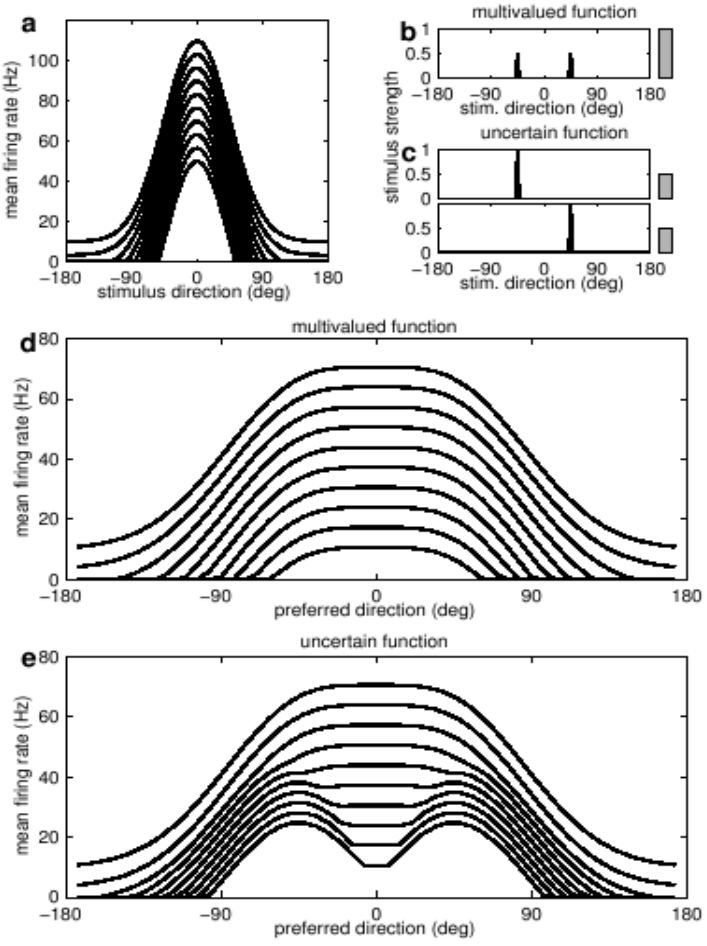

Figure 3: Schematics illustrating encoding in a DDPC. Panel a: The tuning curves of the neurons in the population sensitive to 0° in response to certain, single valued functions: bell shapes with different thresholds. Panel b: $m(x)$ for multivalued certain stimuli. Panel c: $m(x)$ for single valued, uncertain stimuli. Panel d: mean firing rates of the population for the multivalued stimulus. Panel e: mean firing rates for the uncertain stimulus.