1 The tradeoff between flexibility and generalization for discrete codes

1.1 Approximating code flexibility

We consider a code for a discrete set of stimuli to be flexible if arbitrary binary partitions of that set can be read out with a linear decoder. When a code has a mixture of linear and nonlinear stimulus representations, some partitions are orthogonal to the linear structure in the representation and can be implemented only if the nonlinear components of the representation are strong enough – one such partition is the parity or XOR partition. Thus, to approximate code flexibility, we will focus on this case. It allows us to ignore any contribution to the representation from the linear code and focus only on the nonlinear code.

In the nonlinear code for $n^K = N_s$ stimuli, all of the stimuli are $\sqrt{P_N}$ from the origin in representation space and $\sqrt{2P_N}$ from each other. In this case, the vector corresponding to the optimal hyperplane for a linear decoder that implements an arbitrary partition of such stimuli has a constant magnitude c in the direction of all stimuli – and the magnitude is positive for stimuli in one category and negative for stimuli in the other category. Using this understanding, we can calculate the performance of the linear decoder where r is the decoding vector, x is particular stimulus representation in the positive category, and σ^2 is the variance of normally distributed output noise for the neurons in the code:

$$\begin{aligned}
E_f &= P(r\dot{x} > 0) \\
&= P(\mathcal{N}(\sqrt{P_N}, N_s\sigma^2) > 0) \\
&= Q\left(-\frac{\sqrt{P_N}}{\sqrt{N_s\sigma^2}}\right) \\
&= Q\left(-\frac{\sqrt{P_N}}{n^{K/2}\sigma}\right)
\end{aligned}$$

where Q is the cumulative distribution function of the standard normal distribution.

1.2 Approximating code generalization

We consider a code to have good generalization performance when a linear decoder aligned with some combination of code features that is learned on one part of the stimulus space provides good performance on another part of the stimulus space. In a simple case with two stimulus features that each take on two values, this means that a linear decoder that discriminates the value of the second feature (0 or 1) learned for a fixed value of the first feature (say, 0) will generalize with minimal loss of performance to other values of the first feature (in this case, 1). This notion of generalization performance is referred to as cross-condition generalization performance (CCGP).

We set out to approximate CCGP for two sets of two stimulus representations each. Here, we consider a linear code that is distorted by a nonlinear code. Thus, we can consider distances in the purely linear code and distances in the purely nonlinear code separately.

1.2.1 Preliminaries

First, we find how the distance between adjacent stimuli in the linear code depends on the number of features K and the number of values that each feature takes on n along with the linear power of the code (P_L),

$$d_L = \sqrt{\frac{12P_L}{K(n^2-1)}}$$

This distance can be used as a scaling factor that allows translation between distance in stimulus space and distance in representation space – that is, if two stimuli have distance l in stimulus space, then they have distance ld_L in representation space. However, this assumes that each feature is encoded with the same fidelity, which may not always be true.

Second, we find that the nonlinear distance is,

$$d_N = \sqrt{2P_N}$$

Further, we can also observe that each representation in the linear code undergoes a distortion of magnitude P_N in a random direction.

Third, we remind ourselves that the dot product of two random unit vectors $u_1 \cdot u_2$ in a D-dimensional space follows the distribution

$$u_1 \cdot u_2 \sim \mathcal{N}(0, 1/D)$$

for large D.

1.2.2 Main derivation

We use d_{LL} to denote the distance in the linear code between the two stimulus representations used to learn the classification and d_{LG} to denote the distance between the two stimulus representations that are generalized to. This distance is along the same unit vector f_1. We use d_{LA} to denote the distance between the two pairs of stimuli along the axis that they are to be generalized over, which we denote as the unit vector f_2. Each of the four stimuli also undergoes a distortion of magnitude $\sqrt{P_N}$ due to the nonlinear code, we denote the direction of these four distortions as the unit vectors n_i for $i \in [1, 2, 3, 4]$. They are chosen such that $n_i \cdot n_j = 0$ for $i \neq j$ – however, $n_i \cdot f_j$ is not constrained to be zero. From above, we know that $n_i \cdot f_j \sim \mathcal{N}(0, 1/D)$ where D is the full dimensionality of the space (i.e., the number of neurons in the code). Additionally, for convenience, we also use n_{ij} for any number of indices to refer to the following

$$n_{ij} = \frac{n_i + n_j}{\sqrt{2}}$$

and similarly for more indices, so that the end vector is a unit vector.

For simplicity, we assume that $d_{LL} = d_{LG}$, but this can be relaxed later.

First, we find the center points between our two pairs of stimuli in the full code with reference to the "bottom left" stimulus, s_2. In particular, $s_1 = f_1 d_{LL} + d_N n_{12}$, $s_2 = 0$, $s_3 = d_{LL} f_1 + d_{LA} f_2 + d_N n_{23}$, and $s_4 = d_{LA} f_2 + d_N n_{24}$. Thus,

$$\hat{s}_{12} = \frac{1}{2}\left(d_{LL} f_1 + d_N n_{12}\right)$$
$$\hat{s}_{34} = \frac{1}{2}\left(d_{LL} f_1 + 2 d_{LL} f_2 + d_N n_{23} + d_N n_{24}\right)$$

Next, we find the axes of the disorted space in the full code. The vector between s_1 and s_2 is the distorted version of f_1, and is given by

$$f_{1N} = \frac{1}{c}\left(d_{LL} + d_n n_{12}\right)$$

where c is a random variable representing the distance between the two stimulus representations in the full code,

$$\begin{aligned} c &= \sqrt{d_{LL}^2 + d_N^2 + 2 d_{LL} d_N f_1 \cdot n_{12}} \\ &= \sqrt{d_{LL}^2 + d_N^2 + 2 d_{LL} d_N \mathcal{N}(0, 1/D)} \end{aligned}$$

Now, we find the optimal f_2 in the disorted code, which is also the category boundary that would be learned by a decoder trained with both stimulus representation pairs. This is given by,

$$\begin{aligned} f_{2N}^{\text{opt}} &= \hat{s}_{34} - \hat{s}_{12} \\ &= \frac{1}{a}\left(d_{LA} f_2 + \frac{\sqrt{2}}{2} d_N n_{1234}\right) \end{aligned}$$

where a is a random variable given by

$$a = \sqrt{d_{LA}^2 + \frac{1}{2} d_N^2 \sqrt{2} d_N d_{LL} \mathcal{N}(0, 1/D)}$$

which is the distance between the centers of the two sets of points.

The decoding hyperplane used by a decoder trained on only the first set of stimulus representations is given by the unit vector orthogonal to it, which is simply f_{1N}. If $f_{1N} \cdot f_{2N} = 0$, then the decoding hyperplane includes f_{2N}^{opt} and generalization performance will be as good as possible. Thus, we can approximate CCGP by studying the dot product of these two vectors. The dot product can be written as,

$$\begin{aligned} b &= f_{2N}^{\text{opt}} \cdot f_{1N} \\ &= \frac{1}{ac}\sqrt{\frac{3}{2}} d_N d_{LA} \mathcal{N}(0, 1/D) \end{aligned}$$

Geometrically, $b\frac{c}{2}$ is the distance along f_{2N}^{opt} that s_1 and s_2 are from the center point $\hat{s}_{12}$. Here, we have two sides of a right triangle, the hypotenuse has length $c/2$ and the other with length $b\frac{c}{2}$. We use these to find the angle between the learned hyperplane and the optimal hyperplane,

$$\begin{aligned}\theta &= \frac{\pi}{2} - \arccos\left(\frac{2b}{c}\right)\\ &= \arcsin\left(\frac{2b}{c}\right)\end{aligned}$$

In most cases, the larger θ, the worse generalization performance will be.

We can also directly approximate CCGP. To do this, we need to find the position of s_3 and s_4 along the decoding vector defined by f_{1N}, and then we evaluate whether that magnitude is greater or smaller than the threshold $c/2$. So, to find this distance relative to the threshold, we need d such that

$$\begin{aligned}d_3 &= f_{1N} \cdot s_3 - \frac{c}{2}\\ &= \frac{1}{c}\left(d_{LL}f_1 + d_N n_{12}\right)\left(d_{LL}f_1 + d_{LA}f_2 + d_N n_{23}\right) - \frac{c}{2}\end{aligned}$$

First, we focus on the first term and drop c for now,

$$\begin{aligned}t_1 &= \left(d_{LL}f_1 + d_N n_{12}\right)\left(d_{LL}f_1 + d_{LA}f_2 + d_N n_{23}\right)\\ &= d_{LL}^2 + d_{LL}d_N f_1 n_{23} + d_{LL}d_N f_1 n_{12} + d_{LA}d_N f_2 n_{12} + d_N^2 n_{23}n_{12}\\ &= d_{LL}^2 + \frac{1}{2}d_N^2 + \frac{d_{LL}d_N}{\sqrt{2}}\left(f_1n_1 + f_1n_2 + f_1n_2 + f_1n_3\right) + d_{LA}d_N n_{12}f_2\end{aligned}$$

Next, we bring back the full expression and multiply everything by c,

$$\begin{aligned}cd_3 &= d_{LL}^2 + \frac{1}{2}d_N^2 + \frac{d_{LL}d_N}{\sqrt{2}}\left(f_1n_1 + f_1n_2 + f_1n_2 + f_1n_3\right) + d_{LA}d_N n_{12}f_2 - \frac{c^2}{2}\\ &= d_{LL}^2 + \frac{1}{2}d_N^2 + \frac{d_{LL}d_N}{\sqrt{2}}\left(f_1n_1 + f_1n_2 + f_1n_2 + f_1n_3\right) + d_{LA}d_N n_{12}f_2\\ &\quad - \frac{1}{2}d_{LL}^2 - \frac{1}{2}d_N^2 - \frac{d_{LL}d_N}{\sqrt{2}}\left(f_1n_1 + f_1n_2\right)\\ &= \frac{1}{2}d_{LL} + d_{LL}d_N f_1 n_{23} + d_{LA}d_N f_2 n_{12}\\ d &= \frac{\frac{1}{2}d_{LL} + d_{LL}d_N f_1 n_{23} + d_{LA}d_N f_2 n_{12}}{\sqrt{d_{LL}^2 + d_N^2 + 2d_{LL}d_N f_1 n_{12}}}\end{aligned}$$

$$a_{3,4} = a \pm \frac{\sqrt{2}}{2a} d_N \left(\frac{1}{2} d_{LA} \mathcal{N}(0, 1/D) + d_{LL} \mathcal{N}(0, 1/D) \right)$$

where the second term captures the distortion relative to the mean distance between the two stimuli. Then, we can find the distance of s_i for $i \in [3, 4]$ from the learned decoding hyperplane by finding

$$d_3 = f_{1N} s_3 - \frac{c}{2}$$