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 with a pair of stimuli used for training and a third stimulus that must be generalized to. 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),

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

Linear distance derivation: We approach this by computing the variance (i.e., the linear power P_L) of a uniformly sampled K-dimensional lattice with n points spaced at distance d_L along each dimension. Then, we invert the expression for the variance to find an expression for the distance between the points. First, we write the variance P_L as

$$\begin{aligned}
n^K P_L &= \sum_{i=0}^{n-1}\left[\left(i-\frac{n-1}{2}\right)^2 d_L^2 + \sum_{j=0}^{n-1}\left[\left(j-\frac{n-1}{2}\right)^2 d_L^2 + ...\right]\right] \\
&= \sum_{i}^{n-1}\sum_{j}^{n-1}...\sum_{k}^{n-1}\left(i-\frac{n-1}{2}\right)^2 d_L^2 + \left(j-\frac{n-1}{2}\right)^2 d_L^2 + ... + \left(k-\frac{n-1}{2}\right)^2 d_L^2 \\
&= Kn^{K-1}\sum_{i}^{n-1}\left(i-\frac{n-1}{2}\right)^2 d_L^2 \\
&= Kn^{K-1}d_L^2\sum_{i}^{n-1} i^2 - (n-1)\sum_{i}^{n-1} i + n\frac{n-1}{2}^2
\end{aligned}$$

and we can rewrite this with known expressions for the sum of integers and sum of squared integers up to a particular value,

$$\begin{aligned}
n^K P_L &= Kn^{K-1}d_L^2\left[\frac{(n-1)n(2n-1)}{6} - \frac{n(n-1)^2}{2} + \frac{n(n-1)^2}{4}\right] \\
&= Kn^K d_L^2\left[\frac{(n-1)(2n-1)}{6} - \frac{(n-1)^2}{4}\right] \\
&= Kn^K d_L^2\left[\frac{2n^2-3n+1}{6} - \frac{n^2-2n+1}{4}\right] \\
&= Kn^K d_L^2\left[\frac{4n^2-6n+2}{12} - \frac{3n^2-6n+3}{12}\right] \\
n^K P_L &= Kn^K d_L^2\frac{n^2-1}{12} \\
P_L &= Kd_L^2\frac{n^2-1}{12}
\end{aligned}$$

Now, we rewrite in terms of d_L,

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

which is the expression given above.

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. This distance is along the unit vector f_1. Further, we use d_{LG} to denote the distance along that unit vector f_1 of the third stimulus s_3 which is generalized to. We use d_{LA} to denote the distance between the pair of stimuli that is used for training and the third stimulus along the axis that they are to be generalized over, which we denote as the unit vector f_2. Each of the stimuli also undergoes a distortion of magnitude $\sqrt{P_N}$ due to the nonlinear code, we denote the direction of these distortions as the unit vectors n_i. 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.

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, $s_1 =$

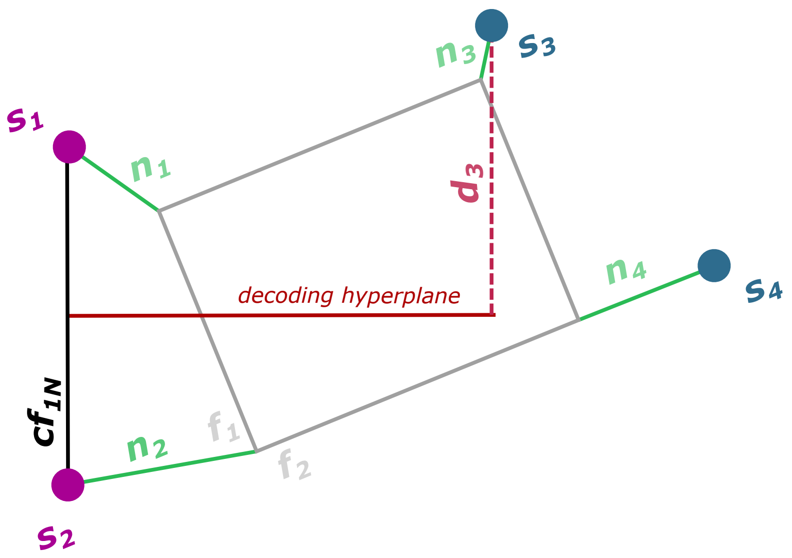

Figure 1: Illustration of the idea behind the CCGP approximation. The purple points are the representations used for training, the blue are those to be generalized to. The representations used for training define a hyperplane (red) that depends on both the linear (f_1) and nonlinear (n_1 and n_2) parts of the code. To approximate CCGP, we can simply take the dot product between the vector defined by this hyperplane and the positions of the other representations (blue). Thus, we find distance $d_3 = f_{1N} \cdot s_3 - \frac{c}{2}$.

$f_1 d_{LL} + d_N n_{12}$, $s_2 = 0$, and $s_3 = d_{LG} f_1 + d_{LA} f_2 + d_N n_{23}$. Thus,

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

Next, we find the vector pointing between the two representations used for learning (that is, s_1 and s_2), which is given by

$$\begin{aligned} f_{1N} &= \frac{1}{c} \left(s_1 - s_2 \right) \\ f_{1N} &= \frac{1}{c} \left(d_{LL} f_1 + d_n n_{12} \right) \end{aligned}$$

where c is a random variable that normalizes f_{1N} to be a unit vector, and corresponds to the distance between s_1 and s_2,

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

Now, using f_{1N}, c, along with our understanding of s_3 as a linear combination of linear and nonlinear codes, we can directly approximate CCGP. To do this, we need to find the position of s_3 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_3 such that

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

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

$$\begin{aligned} t_1 &= \left(d_{LL} f_1 + d_N n_{12}\right)\left(d_{LG} f_1 + d_{LA} f_2 + d_N n_{23}\right) \\ &= d_{LL} d_{LG} + d_{LL} d_N f_1 n_{23} + d_{LG} d_N f_1 n_{12} + d_{LA} d_N f_2 n_{12} + d_N^2 n_{23} n_{12} \\ &= d_{LL} d_{LG} + \frac{1}{2} d_N^2 + \frac{d_N}{\sqrt{2}}\left(d_{LG} f_1 n_1 + d_{LG} f_1 n_2 + d_{LL} f_1 n_2 + d_{LL} f_1 n_3\right) + d_{LA} d_N n_{12} f_2 \end{aligned}$$

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

$$\begin{aligned} c d_3 &= d_{LL} d_{LG} + \frac{1}{2} d_N^2 + \frac{d_N}{\sqrt{2}}\left(d_{LG} f_1 n_1 + d_{LG} f_1 n_2 + d_{LL} f_1 n_2 + d_{LL} f_1 n_3\right) + d_{LA} d_N n_{12} f_2 - \frac{c^2}{2} \\ &= d_{LL} d_{LG} + \frac{1}{2} d_N^2 + \frac{d_N}{\sqrt{2}}\left(d_{LG} f_1 n_1 + d_{LG} f_1 n_2 + d_{LL} f_1 n_2 + d_{LL} f_1 n_3\right) + d_{LA} d_N n_{12} f_2 \\ &\quad - \frac{1}{2} d_{LL}^2 - \frac{1}{2} d_N^2 - \frac{d_{LL} d_N}{\sqrt{2}}\left(f_1 n_1 + f_1 n_2\right) \\ &= d_{LG} d_{LL} - \frac{1}{2} d_{LL}^2 + (d_{LG} - d_{LL}) d_N f_1 n_{12} + d_{LL} d_N f_1 n_{23} + d_{LA} d_N f_2 n_{12} \\ d_3 &= \frac{d_{LG} d_{LL} - \frac{1}{2} d_{LL}^2 + (d_{LG} - d_{LL}) d_N f_1 n_{12} + d_{LL} d_N f_1 n_{23} + d_{LA} d_N f_2 n_{12}}{\sqrt{d_{LL}^2 + d_N^2 + 2 d_{LL} d_N f_1 n_{12}}} \\ d_3 &\sim \frac{d_{LG} d_{LL} - \frac{1}{2} d_{LL}^2 + (d_{LG} - d_{LL}) d_N \mathcal{N}(0, 1/D) + d_{LL} d_N \mathcal{N}(0, 1/D) + d_{LA} d_N \mathcal{N}(0, 1/D)}{\sqrt{d_{LL}^2 + d_N^2 + 2 d_{LL} d_N \mathcal{N}(0, 1/D)}} \end{aligned}$$

and this d_3 relates to the CCGP error rate in the following way,

$$P(\text{CCGP error}_1) = \mathbb{E}_{d_3}\, Q\left(-d_3/\sigma\right)$$

Thus, making d_3 as large as possible will minimize CCGP errors.

For D much larger than any of d_{LG}^2, d_{LA}^2, d_{LL}^2, we can further simplify by ignoring the random variables (which is also the case where the nonlinear and linear codes are perfectly orthogonal to each other),

$$P(\text{CCGP error}) \approx Q\left(-\frac{1}{\sigma}\frac{d_{LG}d_{LL} - \frac{1}{2}d_{LL}^2}{\sqrt{d_{LL}^2 + d_N^2}}\right)$$

Next, we can rewrite this in terms the number of features K, the number of values n taken on by each feature, the steps on the lattice separating the trained pair from each other i (i.e., $d_{LL} = id_L$) and the generalized stimulus from the reference stimulus j (i.e., $d_{LG} = jd_L$), as well as the linear P_L and nonlinear P_N power. So,

$$P(\text{CCGP error}) \approx Q\left(-\frac{12P_L}{K(n^2-1)\sigma}\frac{ij - \frac{1}{2}i^2}{\sqrt{i^2\frac{12P_L}{K(n^2-1)} + 2P_N}}\right)$$