Complement Coding

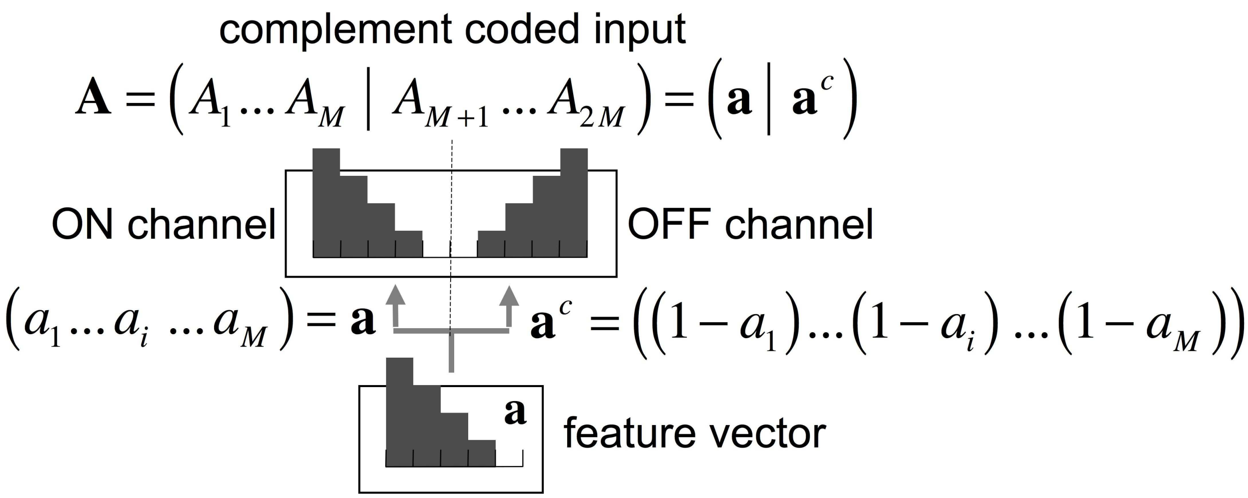

Figure 1. Complement coding transforms an M-dimensional feature vector **a** into a $2M$-dimensional system input vector **A**. A complement-coded system input represents both the degree to which a feature i is present (a_i) and the degree to which that feature is absent $(1-a_i)$.

Reference:

Carpenter, G. A., Grossberg, S., & Rosen, D.B. (1991)
Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system.
Neural Networks, 4, 759-771.
http://profusion.bu.edu/techlab/modules/mydownloads/singlefile.php?cid=26&lid=10

Complement coding: Learning both absent and present features

ART and ARTMAP employ a preprocessing step called *complement coding* (Figure 1), which models the nervous system’s ubiquitous use of the computational design known as *opponent processing* (Hurvich & Jameson, 1957). Balancing an entity against its opponent, as in agonist-antagonist muscle pairs, allows a system to act upon relative quantities, even as absolute magnitudes may vary unpredictably. In ART systems, complement coding (Carpenter, Grossberg, & Rosen, 1991) is analogous to retinal ON-cells and OFF-cells (Schiller, 1982). When the learning system is presented with a set of input features $\mathbf{a} \equiv (a_1 ... a_i ... a_M)$, complement coding doubles the number of input components, presenting to the network both the original feature vector and its complement.

Complement coding allows an ART system to encode within its critical feature patterns of memory features that are consistently *absent* on an equal basis with features that are consistently *present*. Features that are sometimes absent and sometimes present when a given category is learning become regarded as uninformative with respect to that category. Since its introduction, complement coding has been a standard element of ART and ARTMAP networks, where it plays multiple computational roles, including input normalization. However, this device is not particular to ART, and could, in principle, be used to preprocess the inputs to any type of system.

To implement complement coding, component activities a_i of a feature vector **a** are scaled so that $0 \le a_i \le 1$. For each feature *i*, the ON activity a_i determines the complementary OFF activity $(1 - a_i)$. Both a_i and $(1 - a_i)$ are represented in the 2*M*-dimensional system input vector $\mathbf{A} = (\mathbf{a} \mid \mathbf{a}^c)$ (Figure 1). Subsequent network computations then operate in this 2*M*-dimensional input space. In particular, learned weight vectors $\mathbf{w}_J$ are 2*M*-dimensional.

In Figure 1, the input vector [1 .75 .5 .25 0] is complement coded as [1 .75 .5 .25 0 *0 .25 .5 .75 1*].