**Contrastive Representation Learning - InfoNCE loss**

**Introduction**

The goal of contrastive representation learning is to learn such an embedding space in which similar sample pairs stay close to each other while dissimilar ones are far apart. Contrastive learning can be applied to both supervised and unsupervised settings. The **InfoNCE loss** in CPC (Contrastive PredictiveCoding; [van den Oord, et al. 2018](https://arxiv.org/abs/1807.03748)), inspired by NCE, uses categorical cross-entropy loss to identify the positive sample amongst a set of unrelated noise samples.

**InfoNCE loss**

**Definition**

InfoNCE, short for Information Noise Contrastive Estimation, is a widely used loss function in contrastive learning frameworks, particularly in self-supervised learning tasks. Its primary purpose is to estimate the mutual information between two variables by distinguishing between positive and negative samples.

**Pytorch implementation of the InfoNCE loss (**[**git**](https://github.com/RElbers/info-nce-pytorch/blob/main/info_nce/__init__.py)**)**

For calculating *positive\_logit*, only the *positive\_key* element (row) corresponding to *query* element (row) is considered. This is done as sum of rows of the element-wise product of *positive\_key and query.*

For calculating *negative\_logits*, all the *negative\_keys* elements (rows) are considered. This is done as cross product of *negative\_keys* and *query*.

The *logits* after concatenation has shape *[batch\_size, batch\_size+1].*

The labels are zeros of size [*batch\_size]*. This means for each example (row) in *logits,* only the first element is positive and rest are negative.

**Changing *negative\_logits* of pytorch implementation to sum of element-wise multiplication**

For calculating *positive\_logit*, only the *positive\_key* element (row) corresponding to *query* element (row) is considered. This is done as sum of rows of the element-wise product of *positive\_key and query.*

For calculating *negative\_logits*, only the *negative\_keys* element (row) corresponding to *query* element (row) is considered. This is done as sum of rows of the element-wise product of *negative\_keys* and *query*.

The *logits* after concatenation has shape *[batch\_size, 2].*

The labels are zeros of size [*batch\_size]*. This means for each example (row) in *logits,* only the first element is positive and the second is negative.

**InfoNCE loss implemented in this** [**blog**](https://medium.com/@_prinsh_u/why-use-infonce-loss-in-self-supervised-learning-1318b98f001b)

For calculating *positive\_logit*, all the *positive\_key* elements are considered. This is done as cross product of *positive\_key and query.*

For calculating *negative\_logits*, all the *negative\_keys* elements are considered. This is done as cross product of *negative\_keys* and *query*.

The *logits* after concatenation has shape *[batch\_size, 2\*batch\_size].*

The labels are *arange* of size [*batch\_size]*. This means for the diagonal elements of *logits* are considered as positive and the rest as negative.

**2D plots of embeddings of dimension 2**

Consider as simple model (here, a simple fully connected neural network) which transforms the input into 2-dimensional embeddings. As we train this model, we can visualize these embeddings as 2D vectors and observe how they change over the epochs.