Which one is a suitable Readout-Function for directed Edge features, given node representations $\mathbf{x}$

- $\mathbf{x}_i^T \mathbf{x}_j$
- $\mathbf{x}_i^T \mathbf{W} \mathbf{x}_j$
- $\mathbf{x}_i^T \mathbf{W}^T \mathbf{W} \mathbf{x}_j$
- $(\mathbf{x}_i + \mathbf{x}_j)^T \mathbf{W} (\mathbf{x}_j + \mathbf{x}_i)$
- something else I don't remember

Why would one Neuralize something like K-means

- Makes it possible to train using backprop
- Explain it using Gradient propagation based methods

Which one of these is invariant to SO

- $\mathbf{x}_i + \mathbf{x}_j$
- $\tanh(\mathbf{x}_i + \mathbf{x}_j)$
- $|\mathbf{x}_i + \mathbf{x}_j|$
- $(\mathbf{x}_i + \mathbf{x}_j)^2$
- ??

How does BYOL work

- it pushes negatives away and pulls positives together
- it uses a large amount of negative examples in batch
- it learns from a temporal moving average of a copy of itself
- ???

Which Reinforcement Learning Function is this
$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1} - V(S_t))]$

- Monte Carlo
- Temporal Difference Learning
- Bellman Equation

Why would we use Optimize then Discretize

- always gives better gradients
- is memory efficient
- we do not need numerical integration
- ???

SimCLR loss

- Derive $\frac{\delta}{\delta z_i} \sum_{i \neq k} exp(\frac{sim(z_i, z_k)}{\tau})$
  for $sim : z_i^T z_k$
- Show Full simCLR loss
- Explain simCLR loss components and τ's role
- explain Triplett loss components
  $L = max(0, |z_i - z_k|^2 - |z_i - z_j|^2 + m)$
- with z_i being the anchor, z_k being a positive and z_j being a negative

Density Estimation

- Full ELBO Derivation (Like in Homework sheet)
- Then some weird shit with derive $p(x)$ given some bayesian functions full on Integration, probability, stuff no idea here

Programming

1. Invariant Message passing networks

- Write a function to compute distances between points given as matrix $\mathbf{X} \in \mathbb{R}^F$
- Write Message passing Function such that the messages are passed with a matrix $M \in \mathbb{R}^{K \times K \times F}$
- Write Update Function
- Compute 2 Layers of GNN

1. Modified Attention

- Attention approximated by using only a subset of sequence length (with dimension r)
- use torch*orthogonal* to compute modified $\mathbf{Q}', \mathbf{K}'$
- where $\mathbf{Q}', \mathbf{K}'$ was something like

$$\phi(x_i)_j = \exp\left(w_j^\top x_i - \frac{\|x_i\|^2}{2}\right)$$

- write function to compute modified attention layer, taking care that it's runtime is in $\mathcal{O}(\mathbf{L})$ for sequence length $\mathbf{L}$
- Modified Attention was something like this I believe
- $$V_{\text{out}} = \text{diag}\left(\phi(Q)\phi(K)^T\mathbf{1}_L\right)^{-1}\phi(Q)\left(\phi(K)^T V\right)$$