The impact of stochasticity on learned representations in a neural network

Motivation: Lower-dimensional representations learned by a neural network depend on model parameters, which respond to several sources of stochasticity:

- Initial weights
- Distribution of training data
- Stochastic gradient descent

For example, varying the random seed in PyTorch yields significant differences in model performance[1].

Problem: How is stochasticity during network training reflected in the latent representation learned by a neural network? We will investigate this using a bottlenecked neural network (BNN), where the latent vector is defined by the activations of the bottleneck layer.

- BNN is trained to classify MNIST dataset
- **Hypothesis:** Latent distribution will be different for different sets of initial weights

Methods:

- Construct BNN from scratch (numpy) to remove all sources of stochasticity[2]
    - Verify no stochasticity in network by checking neuron weights after re-training
- Train multiple BNNs with bottleneck dimension $k \in 2$ to convergence
    - Introduce stochasticity in initial weights by exploring 10^3 random seeds (which are used to set initial weights of some or all nodes) → yielding 10^3 models which are otherwise identical
- Investigate effects of stochasticity:
    - Qualitatively compare 2D latent distributions by plotting scatterplot of latent vectors (color-coded by output class) for several models:
        - With the same accuracy score
        - With different accuracy scores (i.e. compare between lowest-performing and highest-performing models)
    - Across models, get distributions of:
        - Validation accuracy
        - Average pairwise distance between cluster centroids in latent space
        - Distances between centroids of specific clusters (e.g. how far away is Cluster 3 from Cluster 8)
    - Entire matrix of pairwise cluster distances can be computed from each latent distribution, and Frobenius similarity can be computed between two distance matrices[3]
        - Do two models with similar initial weights generate latent distributions that are more similar (smaller Frobenius distance) than two models with very different initial weights?

Potential extensions

- Repeat above with different bottleneck dimension (e.g. $k = 1$) or different source of stochasticity (e.g. noise in training images)
- What happens in the latent space when a neural network overfits on training data? Train models past convergence and compare resulting latent distribution with that of models trained to convergence

References

1. torch.manual_seed(3407) is all you need
2. https://towardsdatascience.com/mnist-handwritten-digits-classification-from-scratch-using-python-numpy-b08e401c4dab
3. https://math.stackexchange.com/questions/507742/distance-similarity-between-two-matrices/508388