Performing Variational Inference on CIFAR 10 Dataset

Model Selection. We decided to apply a Multivariate Gaussian Mixture Model to the data and perform variational inference and observe the convergence of the objective ELBO as we train on the CIFAR 10 training data batches.

Data Pre-Processing. To perform variational inference on the CIFAR 10 image data, we must first decide on how to represent the 32x32 images as feature vectors. Each image sample was provided as a numpy array of size 3072, each containing 1024 R, G, and B values in row-major order. We initially decided to keep the data in their original representation as vectors of dimension 3072, but we found the inference time taking much longer than we anticipated to produce any interpretable results. Hence, we performed dimensionality reduction through principal component analysis (PCA) to constrain the number of features in our dataset to 100. Reducing the size of the input vectors from 3072 to 100 speeds up our stochastic variational inference algorithm significantly.

Mixture Model Representation. The model we selected to fit the observed samples is a Multivariate Gaussian Mixture Model as represented in the graphical model below:

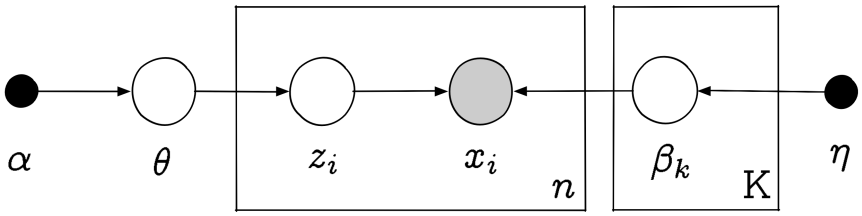

Figure 1. Graphical representation of a Gaussian mixture.

The number of classes K is 10. β_k represents the mixture components with variance η that form each cluster. The total number of samples n is 50K excluding the held-out dataset. θ represents the mixture proportions. Given that CIFAR 10 contained 6K samples per class in a 60K sample dataset, θ was set to a constant $1/K$ for each mixture component. z_i is a latent categorical variable representing the cluster assignments for each data point x_i.

Generative Model and Guide Implementation. We used the probabilistic programming framework `pyro` to build a variational distribution, also known as the guide, that would approximate the posterior distribution using stochastic variational inference (SVI). When implementing our generative model, we used a `MultivariateNormal` distribution to sample the mixture components β_k where the covariance matrix was diagonal since we assume independence between features. We then used a `Categorical` distribution to model the latent cluster assignments z_i and sample the observed samples x_i from the distribution of its randomly assigned cluster.

Upon building a generative model for the SVI algorithm, we used Automatic Guide Generation to provide learned parameters for a variational distribution that approximates our posterior distribution. The `AutoDiagonalNormal` is based on a normal distribution with a diagonal covariance matrix, which is suitable for our purposes. A potential extension of this study would be to use a custom

guide such as the mean field family instead of the automatic guides and the performance differences between the two options.

ELBO Convergence using SVI. After defining a generative model, a guide, and the loss (ELBO), we could use SVI on the image data. We used the Adam optimizer and set the learning rate set to 0.1 and iterated each step on the full 50K observed samples as shown below:

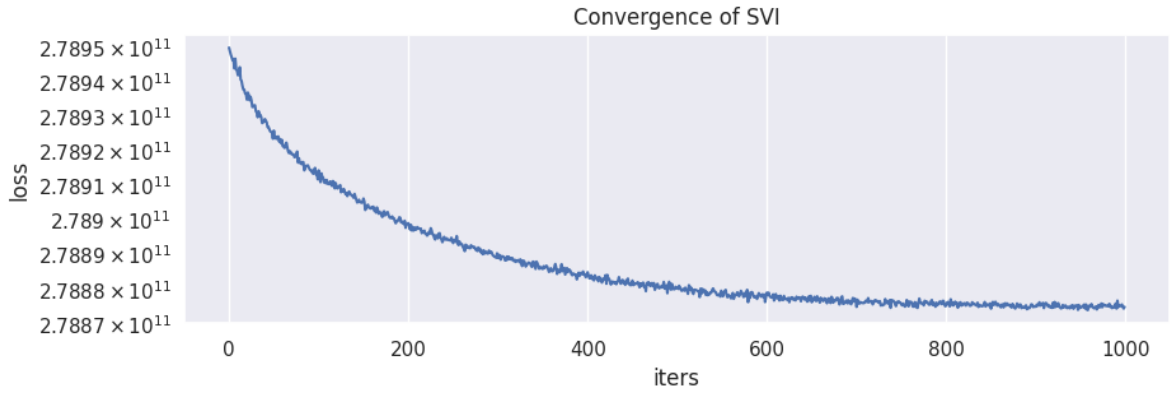

Figure 2. Stochastic variational inference convergence using 50K samples from five training batches over 1000 iterations.

The SVI convergence plot demonstrates convergence at around 800-1000 iterations. The loss represented in the figure above is the negative ELBO, which demonstrates that the inference algorithm is successful in maximizing the ELBO. We then used these learned parameters from SVI to perform inference on the held-out dataset to observe ELBO convergence for comparison.

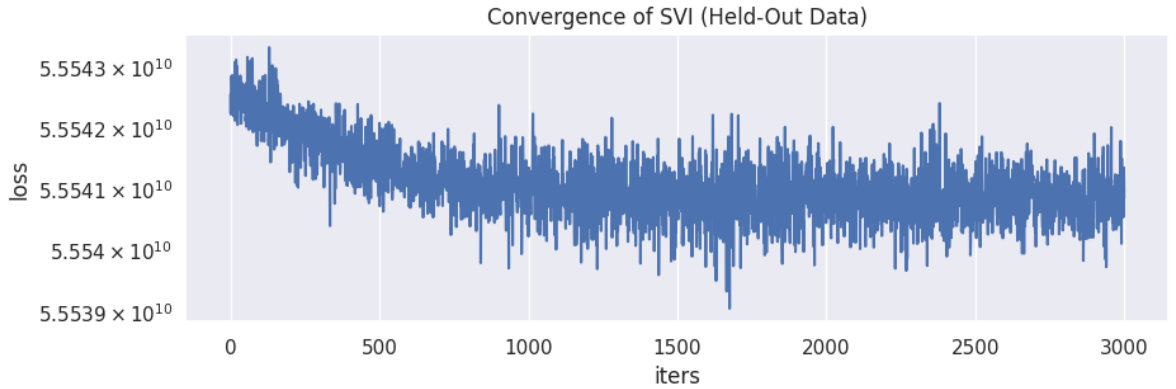

Figure 3. Stochastic variational inference convergence using 10K samples from held-out dataset over 3000 iterations.

Compared to the training on the previous dataset, we observe far more noisy convergence shown in the plot above. Despite the noisier plot, we definitely do observe a general pattern of minimizing loss across the 3000 iterations. Perhaps, this is due to the SVI procedure using the parameters that were learned in the previous procedure using the training data which indicates smaller room for

further optimization. Perhaps tuning hyperparameters such as the learning rate would contribute to improvements in SVI.

Interpretation of learned parameters for the mixture components. The parameters that we have learned from SVI were the mixture components β_k and their variance η. We decided to plot the values that we extracted from a single mixture component. The values correspond to a hundred feature weights, as we reduced the images to 100 dimensions from 3072 using PCA.

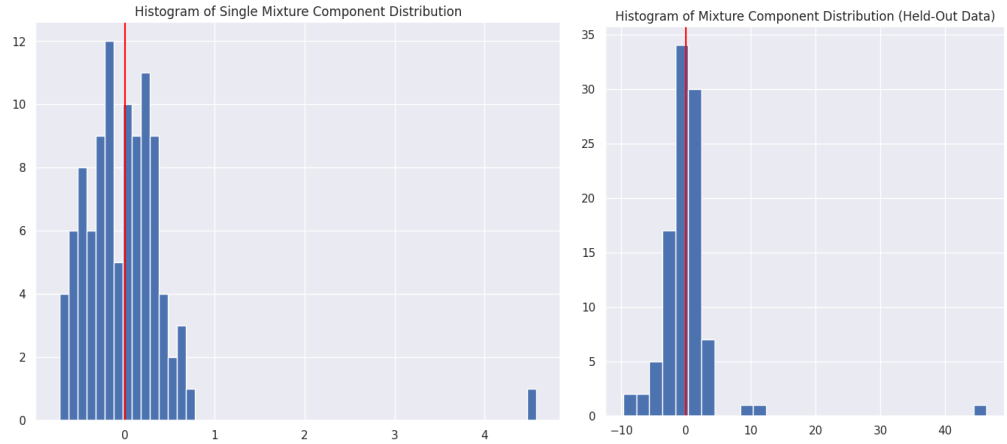

Figure 4. Histograms represent distributions of values assigned to each dimension of the beta component means.

The leftmost figure represents the distribution from the 50K training samples. The rightmost figure represents the distribution from held-out data (10K samples). There seem to be a few outliers whose counts are in the 1-2 range. These patterns are consistent across both the learned parameters from SVI on the 50K samples and the parameters from SVI on the held-out data. However, for the most part, we can observe that the features are fairly normally distributed around zero. It is interesting given that we observed a similar normal distribution when observing the weights assigned to features when performing text classification.