We introduce the Cluster-aware Generative Model (CaGeM), an extension of a VAE, that improves the generative performance by being able to model the natural clustering in the higher feature representations through a discrete variable. The model can be trained fully unsupervised, but its performances can be further improved using labelled class information that helps in constructing well defined clusters.

Cluster-aware Generative Models

In VAEs, the addition of more stochastic layers is often accompanied with a built-in pruning effect so that the higher layers become inactive and therefore not exploited by the model (Sønderby et al., 2016). CaGeM provides the possibility of learning a representation in the higher stochastic layers that can model the natural clustering of the data. This results in a model that is able to disentangle some of the factors of variation in the data and that extracts a hierarchy of features that are beneficial for the generation phase.

We favour the flow of higher-level global information by extending the generative model of a VAE with a discrete variable representing the choice of different clusters in the data (Figure 1):

\[
p_q(x, z_1, z_2) = \sum_y p_q(y|z_1, z_2) p(z_1)p(z_2). \]

The corresponding variational approximation looks as follows:

\[
q_\phi(y, z_1, z_2|x) = q_\phi(z_1|x, y)q_\phi(y|z_1, x)q_\phi(z_2|x, y, z_1). \]

The discrete variable depends solely on the highest latent stochastic layer that needs therefore to stay active for the model to be able to represent clusters in the data. The ELBO in CaGeM is:

\[
\mathcal{F}(\phi, \theta) = \mathbb{E}_{q_\phi(z_1|x)} \left[ \sum_y q_\phi(y|z_1, x) \right] - \mathbb{E}_{q_\phi(z_2, y|x, z_1)} \left[ \log \frac{p_q(x, y|z_1, z_2)q_\phi(y|z_1, x)}{q_\phi(z_2, y|x, z_1)} \right].
\]

In some applications we may have class label information for some of the data points in the training set. In the following we will show that CaGeM provides a natural way to exploit additional labelled data to improve the performance of the generative model. This differs from traditional semi-supervised learning (Kingma et al., 2014, Maalæe et al., 2016), since the labelled data support the generative task. CaGeM contains two classifiers:

1. In the inference network we can compute the class probabilities given the data by integrating out the stochastic variables from:

\[
q_\phi(y|x) = \int q_\phi(y|z_1, x)dz_1 = \int q_\phi(y|z_1, x)q_\phi(z_1|x)dz_1
\]

2. In the generative model we have another set of class probabilities given the posterior distribution of the top latent stochastic variables:

\[
p_q(y|x) = \int p_q(y|z_2)q_\phi(z_2|x)dz_2 = \int p_q(y|z_2)\left(\int q_\phi(z_1|x, y, z_1)dz_1\right)dz_2.
\]

We evaluate the generative performance of CaGeM on the MNIST dataset. We can see from Table 1 that the more labelled samples we use, the better the generative performance will be. Even though the results are not directly comparable, since CaGeM exploits a small fraction supervised information, we find that by using only 100 labelled samples, CaGeM achieves state-of-the-art performance on permutation invariant MNIST.

It is also interesting to see that the fully unsupervised CaGeM-0 performs substantially better than directly comparable architectures, by embedding information on clusters in the higher latent stochastic layers.

When testing the performance on Omniglot we use the same algorithm as cluster inference. From Table 2 we see an improvement over other comparable VAE architectures (VAE, IWAE and LVAE), however, the performance is far from the once reported from the auto-regressive models. This indicates that the alphabet information is not as strong.

Conclusion

In this work we have shown how to perform semi-supervised generation with CaGeM. We showed that CaGeM improves the generative log-likelihood performance over similar deep generative approaches by creating clusters for the data in its higher latent representation using unlabelled information. CaGeM also provides a natural way to refine the clusters using additional labelled information to further improve its modelling power.

References


DVAE (Rolf, 2017)

Table 1: Test log-likelihood on permutation invariant MNIST for L number of stochastic layers, M importance weighted samples and K number of clusters. Directly comparable models are IWAE and LVAE.

We evaluate the generative performance of CaGeM on the MNIST dataset. We can see from Table 1 that the more labelled samples we use, the better the generative performance will be. Even though the results are not directly comparable, since CaGeM exploits a small fraction supervised information, we find that by using only 100 labelled samples, CaGeM achieves state-of-the-art performance on permutation invariant MNIST.

It is also interesting to see that the fully unsupervised CaGeM-0 performs substantially better than directly comparable architectures, by embedding information on clusters in the higher latent stochastic layers.

When testing the performance on Omniglot we use the same algorithm as cluster inference. From Table 2 we see an improvement over other comparable VAE architectures (VAE, IWAE and LVAE), however, the performance is far from the once reported from the auto-regressive models. This indicates that the alphabet information is not as strong.

Conclusion

In this work we have shown how to perform semi-supervised generation with CaGeM. We showed that CaGeM improves the generative log-likelihood performance over similar deep generative approaches by creating clusters for the data in its higher latent representation using unlabelled information. CaGeM also provides a natural way to refine the clusters using additional labelled information to further improve its modelling power.

References


