Graph autoencoders

Convolutional GNNs and attention GNNs are both discriminative models, meaning they're designed to tell apart different data points, like figuring out if a picture is of a cat or a dog.

Generative models are like trying to understand the whole picture, instead of just focusing on the edges. Discriminative models, on the other hand, are more about spotting the differences between things. For instance, a generative model would pick up on how to create images of cats and dogs, not just the specific features that make them different, like a cat's pointed ears or a spaniel's long ears.

The aim here is to extend generative architectures to act on graph-structured data, leading to graph autoencoders (GAEs) and variational graph autoencoders (VGAEs).

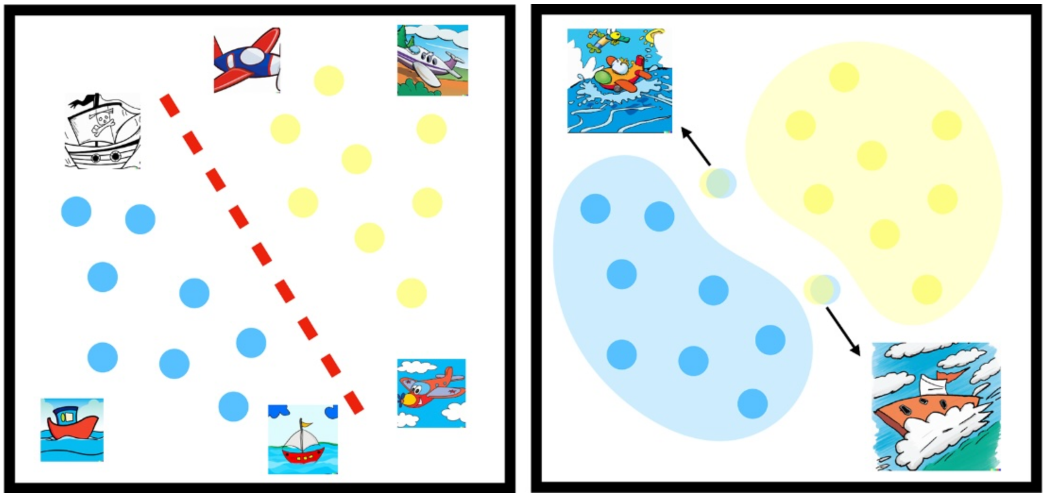
Discriminative --> We want our models to return the probability of some target, Y, given an instance of data, X. We can write this as P(Y|X)

$$\max_{w,b} \prod_{n=1}^{N} P(y_n \mid x_n)$$

Generative --> they model the joint probability between data and targets, P(X,Y)

$$\max_{\theta} \prod_{n=1}^{N} P(x_n, y_n)$$

Both models output a probability P(Y|X). The difference is what they learn to get there.



Generative models can be a great way to handle data labeling when it’s a bit pricey, but creating datasets is a breeze.

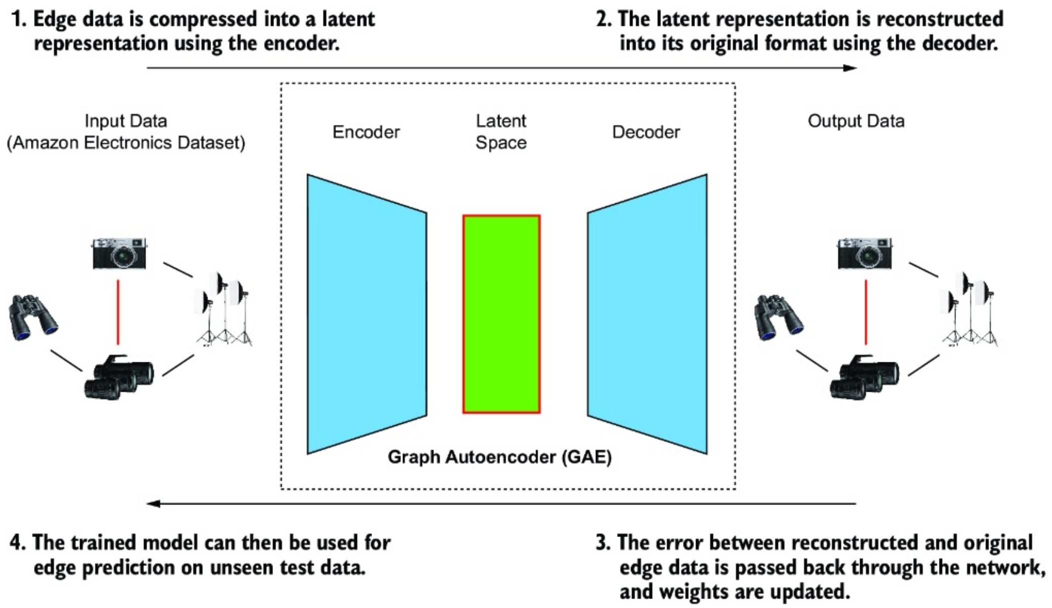
Supports data augmentation.

Also, discriminative models are frequently used after generative models. This is because generative models usually learn without labels, using a “self-supervised” approach. They figure out how to compress or encode complex, high-dimensional data into simpler forms.

Graph autoencoders for link prediction

```
1.Define the model:
Create both an encoder and decoder.
Use the encoder to create a latent space to sample from.
2.Define the training and testing loop by including a loss
suitable for constructing a generative model.
3.Prepare the data as a graph, with edge lists and node features.
4.Train the model, passing the edge data to compute the loss.
5.Test the model using the test dataset.
```

The GAE model is similar to a typical autoencoder. The only difference is that each individual layer of our network is a GNN, such as a GCN or GraphSAGE.



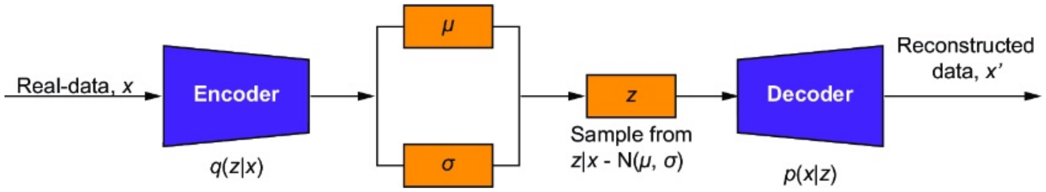
1. Edge data is compressed into a latent representation using the encoder.

2. The latent representation is reconstructed into its original format using the decoder.

3. The error between reconstructed and original edge data is passed back through the network, and weights are updated.

4. The trained model can then be used for edge prediction on unseen test data.

Building a variational graph autoencoder



Reference

Graph Neural Networks in Action, K. Broadwater, 2025