Variational autoencoder: the foundations

- Kingma and Welling, *Autoencoding variational Bayes*, ICLR 2014

A Bayesian’s favourite model, of a biased coin

Z~Beta(α, β) for fixed parameters α, β

X~Bin(1, Z)

The Bayesian likes to ask:
given observations \(x_1, x_2, ..., x_N\) drawn from distribution \(X\),
what is the posterior distribution of \(Z\)?

We can fit this model using maximum likelihood estimation.

\[
\log \text{lik} = \max_\theta \frac{1}{N} \sum_{i=1}^{N} \log \Pr_X(x_i|\theta)
\]

\[
\Pr_X(x_i|\theta) \approx \int \Pr_X(x_i|Z) \Pr_Z(Z) \, dZ \approx \frac{1}{M} \sum_{m=1}^{M} \Pr_X(x_i|Z(m), \theta)
\]

\[
\log \text{lik} (\theta) = \frac{1}{N} \sum_{i=1}^{N} \log \Pr_X(x_i|\theta)
\]

\[
\log \text{lik} (\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \log \left( \frac{1}{M} \sum_{j=1}^{M} \Pr_X(x_i|Z_j = z_i, \theta) \right)
\]

\[
\log \text{lik} (\theta) \geq \frac{1}{N} \sum_{i=1}^{N} \frac{1}{M} \sum_{j=1}^{M} \log \Pr_X(x_i|Z = z_i, \theta)
\]

This is a nice simple function,
and we can maximize the \(\log \text{lik} (\theta)\) lower bound using gradient ascent.
Training a generative model

TRAINING GOAL
find \( \theta \) to maximize the approximate lower bound

\[
\log \text{liklb}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{J} \sum_{j=1}^{J} \log \text{Pr}_X(x_i | Z = z_j, \theta)
\]

TRAINING METHOD
For each datapoint \( x_i \) (or each batch):
- Generate one or more random \( z \) samples from \( \text{Pr}_Z \)
- Compute the loss function, \( L(\theta) = -\log \text{Pr}_X(x_i | Z = z, \theta) \)
  as well as its gradient \( dL/d\theta \)
- Update \( \theta \) to reduce the loss function

EVALUATION
evaluate the model by computing \( \log \text{liklb}(\theta) \) on a holdout set

Digression: Monte Carlo versus Importance Sampling

PROBLEM STATEMENT
Given a random distribution \( Z \) and a function \( h \), we want to approximate \( \mathbb{E}_Z h(Z) \)

We want:

\[
\text{Pr}_X(x) = \mathbb{E}_Z \left[ \frac{\text{Pr}_X(x | Z, \theta)}{h(z)} \right]
\]

MONTE CARLO APPROXIMATION
Sample \( z_1, ..., z_j \) from \( Z \). Then \( \mathbb{E}_Z h(Z) \approx \frac{1}{J} \sum_{j=1}^{J} h(z_j) \)

IMPORTANCE SAMPLING APPROXIMATION
Choose a distribution \( \tilde{Z} \). Sample \( z_1, ..., z_j \) from \( \tilde{Z} \). Then \( \mathbb{E}_Z h(Z) \approx \frac{1}{J} \sum_{j=1}^{J} h(z_j) \frac{\text{Pr}_Z(z_j)}{\text{Pr}_Z(z_j)} \)

This approximation is valid for any distribution \( \tilde{Z} \).
It works best (i.e. is good for small \( J \)) if \( \tilde{Z} \) is biased in favor of values where \( h(z) \) is large.

In our case, the approximation is perfect when \( Z \) is the Bayesian posterior, \( Z \sim (Z | X = x) \)

Summary

Suppose we want to train a latent-variable generative model, i.e. use maximum likelihood estimation to find the \( \theta \) that maximizes the likelihood of the dataset.

In toy examples, we can write down a formula for the likelihood \( \text{Pr}_X(x) \).
For interesting neural networks this is intractable, so we have to approximate.

The best approximation comes from building an autoencoder, where the encoder has the job of approximating the Bayesian posterior \( \text{Pr}_Z(z | X = x) \).
The loss function is derived using probability theory, based on the noise model.