11-695: AI Engineering
VAE Implementation

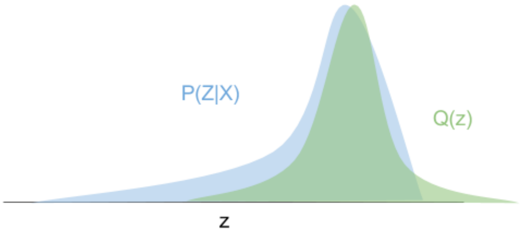
Spring 2019

Outline

1. Variational AutoEncoder: Review
2. Training with Reparametization
3. Tuning VAE (Active Research Topic)

Variational Parameters

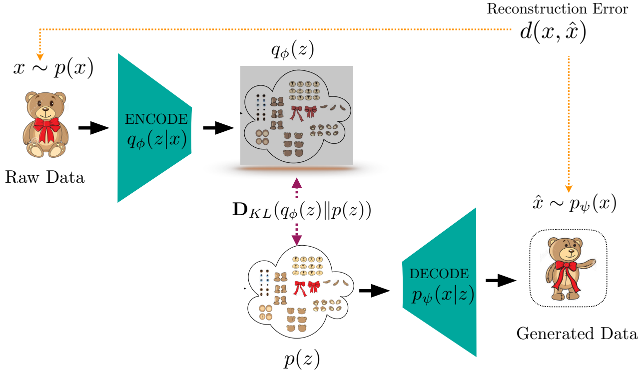
Carnegie Mellon



- So far, we use a variational distribution $q(z)$ to approximate $p(z|x)$
- Not yet mentioned: each variational distribution as a set of variational parameter ϕ
- And as usual, we aproximate ϕ by a neural network (MLP).

Image credit: Eric Jang

AE + Variational Approximation



- Still AutoEncoder, with KL term for variational vs. true distribution → Variational AutoEncoder

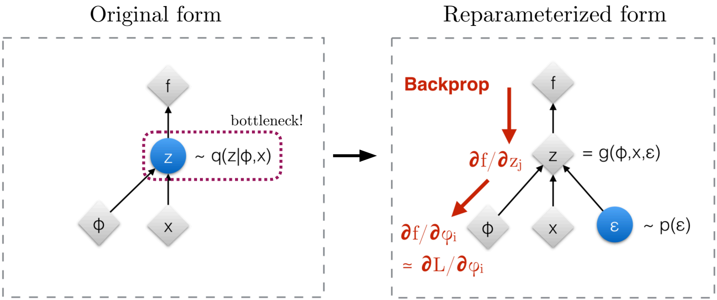
Image credit: Eric Jang

Outline

Carnegie Mellon

1. Variational AutoEncoder: Review
2. Training with Reparametization
3. Tuning VAE (Active Research Topic)

But, not yet fully differentiable



- Sampling $z \sim q_\phi(x) = q(x, \phi)$ is not continuous
- Trick: push the sampling to the input level of the graph
  - $\epsilon \sim p(\epsilon) = \mathbf{N}(\mathbf{0}, \mathbf{I})$
  - $z = f_\phi(x, \epsilon)$, a neural network now!

Image credit: Max Welling

VAE Encoder: Convolution

Carnegie Mellon

encoder.py

```python
encoder_net = tf.keras.Sequential([
    conv(base_depth, 5, 1),
    conv(base_depth, 5, 2),
    conv(2 * base_depth, 5, 1),
    conv(2 * base_depth, 5, 2),
    conv(4 * latent_size, 7, padding="VALID"),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(2 * latent_size, activation=None),
])

def encoder(images):
  images = 2 * tf.cast(images, dtype=tf.float32) - 1
  net = encoder_net(images)
  return tfd.MultivariateNormalDiag(
      loc=net[..., :latent_size],  # mean
      scale_diag=tf.nn.softplus(net[..., latent_size:] + # variance
                                  _softplus_inverse(1.0)),
      name="code")
```

- New Module: ▸ Tensorflow Probability ▸ tfp.distributions.MultivariateNormalDiag

VAE Encoder: Convolution

Carnegie Mellon

- Here: combine μ and σ in the same CNN
- Other option not using TFD:
  - $\epsilon \sim p(\epsilon) = \mathbf{N}(\mathbf{0}, \mathbf{I})$
  - Then $f(\phi, \epsilon) = \mu + \sigma * \epsilon$
- For numerical stability, usually we model $log(\sigma)$ instead of σ
- Also for *sigma* we use $f_{\text{softplus}}(\sigma) = log(e^{\sigma} + 1)$. Why?

VAE Decoder: De-Convolution

decoder.py

```
decoder_net = tf.keras.Sequential([
    deconv(2 * base_depth, 7, padding="VALID"),
    deconv(2 * base_depth, 5),
    deconv(2 * base_depth, 5, 2),
    deconv(base_depth, 5),
    deconv(base_depth, 5, 2),
    deconv(base_depth, 5),
    conv(output_shape[-1], 5, activation=None),
])

def decoder(codes):
  original_shape = tf.shape(input=codes)
  # Collapse the sample and batch dimension and convert to rank-4 tensor for
  # use with a convolutional decoder network.
  codes = tf.reshape(codes, (-1, 1, 1, latent_size))
  logits = decoder_net(codes)
  logits = tf.reshape(
      logits, shape=tf.concat([original_shape[:-1], output_shape], axis=0))
  return tfd.Independent(tfd.Bernoulli(logits=logits),
                         reinterpreted_batch_ndims=len(output_shape),
                         name="image")
```

- New Module: tfp.distributions.Independent

VAE Decoder: De-Convolution

- Recall: we need to use `deconv` (`tf.keras.layers.Conv2DTranspose`)
- Here we apply Bernoulli distribution to decoder's output
- Why? (Hint: Think of Binomial or binary cases)

VAE Prior: Mixture of Gaussians

Carnegie Mellon

prior.py

```
if mixture_components == 1:
  # See the module docstring for why we don’t learn the parameters here.
  return tfd.MultivariateNormalDiag(
      loc=tf.zeros([latent_size]),
      scale_identity_multiplier=1.0)

loc = tf.compat.v1.get_variable(
    name="loc", shape=[mixture_components, latent_size])
raw_scale_diag = tf.compat.v1.get_variable(
    name="raw_scale_diag", shape=[mixture_components, latent_size])
mixture_logits = tf.compat.v1.get_variable(
    name="mixture_logits", shape=[mixture_components])

return tfd.MixtureSameFamily(
    components_distribution=tfd.MultivariateNormalDiag(
        loc=loc,
        scale_diag=tf.nn.softplus(raw_scale_diag)),
    mixture_distribution=tfd.Categorical(logits=mixture_logits),
    name="prior")
```

- New Module: tfp.distributions.MixtureSameFamily

VAE Prior: Mixture of Gaussians

- Remember in Bayesian Statistics what is the role of prior?
- Here: mixture of Gaussian

Image credit: Angus Turner

VAE Model

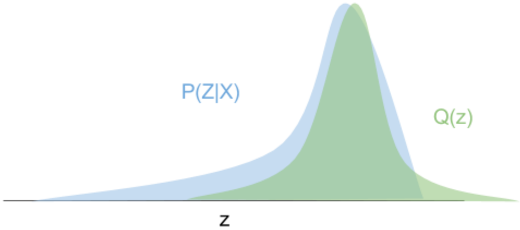
model.py

```python
approx_posterior = encoder(features)
approx_posterior_sample = approx_posterior.sample(params["n_samples"])
decoder_likelihood = decoder(approx_posterior_sample)

# ‘distortion‘ is just the negative log likelihood.
distortion = -decoder_likelihood.log_prob(features)
avg_distortion = tf.reduce_mean(input_tensor=distortion)

rate = tfd.kl_divergence(approx_posterior, latent_prior)
avg_rate = tf.reduce_mean(input_tensor=rate) # KL term

# total loss
elbo_local = -(rate + distortion)
elbo = tf.reduce_mean(input_tensor=elbo_local)
loss = -elbo

# train_op
optimizer = tf.compat.v1.train.AdamOptimizer(learning_rate)
train_op = optimizer.minimize(loss, global_step=global_step)
```

- KL Divergence distance: tfp.distributions.kl_divergence

VAE Model

Carnegie Mellon



- Objective is to maximize ELBO $= log\, p(x) - \mathbf{D}_{KL}(q(z) \| p(z|x))$
  - $log\, p(x)$ is log likelihood
  - The other is KL term calculated easily using TFD
  - Note: it might be tricky to calculate this term manually

Image credit: Eric Jag

VAE Results

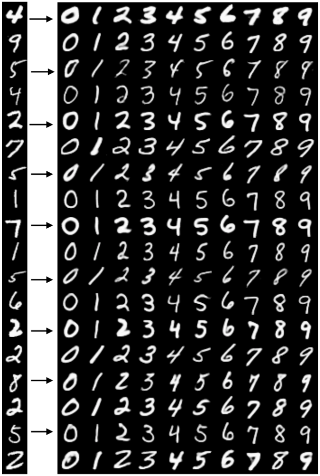


Image credit: Max Welling

VAE Results

Carnegie Mellon

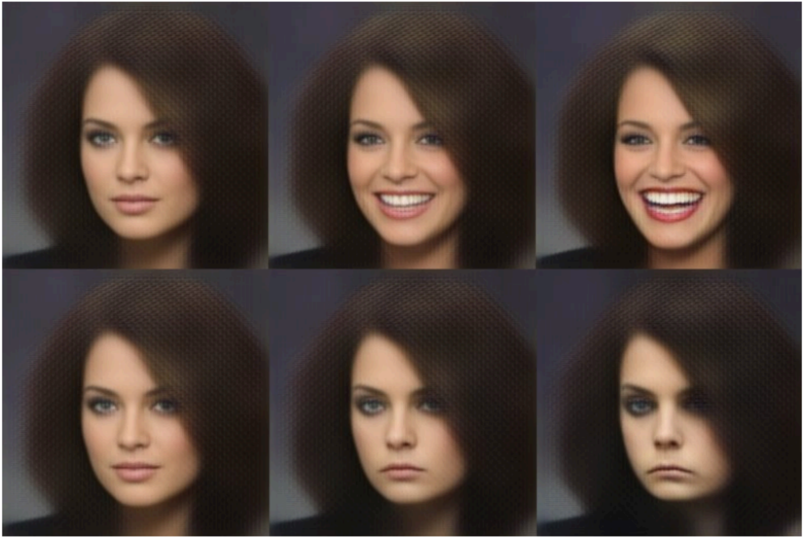
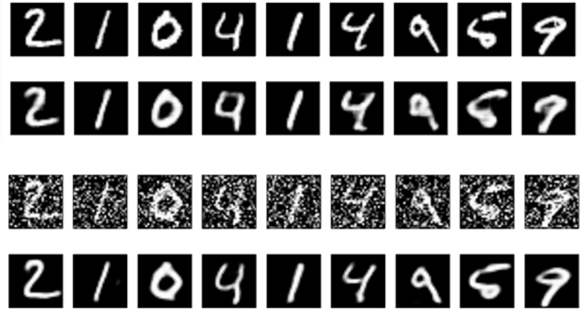


Image credit: Diederik P. Kingma

Play More with Code



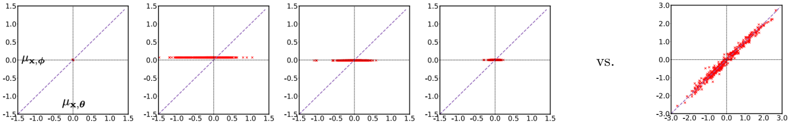
- ‣ Demo (D. P. Kingma)
- ‣ VAE with Keras
- ‣ VAE with TF
- ‣ VAE with TF2
- ‣ VAE with TF Probability

Outline

CarnegieMellon

1. Variational AutoEncoder: Review
2. Training with Reparametization
3. Tuning VAE (Active Research Topic)

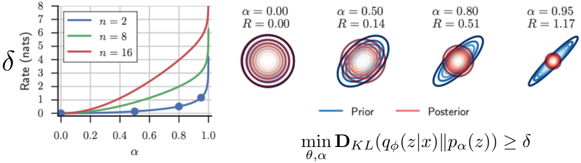
Posterior Collapse



- In VAE, the probability $q(z|x)$ is called posterior
- Posterior $q(z|x)$ can collapse to prior $p(z)$ for all x: $\mathbf{D}_{KL} \approx 0$
- Also means the model ignores the latent features, producing outputs not resembling inputs.
- Review: maximize ELBO $= log\, p(x) - \mathbf{D}_{KL}(q(z) \| p(z|x))$
  - Right from the start: $q(z)$ are far away from each other $p(z|x)$
  - So will be collapsed $\rightarrow$ (almost) ignore $\mathbf{D}_{KL}$, learn nothing from it.

Image credit: Junxian He *et al.* 2019

Posterior Collapse - Solutions



$$\min_{\theta,\alpha} \mathbf{D}_{KL}(q_\phi(z|x) \| p_\alpha(z)) \geq \delta$$

- Anneal weight for $\mathbf{D}_{KL}$ (Sam Bowman et al. 2015)
- Learn (dynamic) prior and discretize latent space (Aaron van den Oord et al. 2017)
- Use autoregressive flow to learn prior and decoding (Xi Chen et al 2017)
- Carefully init variational params and refine (Yoon Kim et al 2018)
- Agressively update encoder more (Junxian He et al. 2019)
- δ-VAE: Lower bound $\mathbf{D}_{KL}$ (Ali Razavi et al. 2019)

Image credit: Ali Razavi *et al.* 2019