VAE.Py

MAIN METHODS

- init
  - ENCODERS
    - Z-encoder — posterior of Z (Enc) - sample from it
    - L-encoder — posterior of L-enc. - sample from it
  - DECODER (Decoder SCVI)

+

- INFERENCE
  - Encoders: Sample from Encoders
  - Decoder: Use these samples, building 3 decoders based on latent space
    - Linear Sequential Softmax NN
    - Linear for R
    - Linear for Dropout

+ dependent whether you have
  - gene w/ label (gene-label)
  - gene w/ batch
  - gene

  — takes gene r or linear comb. from px_r from the vae-object

→ Inference is called in FORWARD PASS.

FIND out: what is px_...?

PX is a type of a decoder → FC layers

Forward pass finds the KL divergence between the (pre-defined params variational N-dist) and the N-dist found by Inference, both GLOBALLY (z) and locally (l)

In addition, forward pass finds the REconstruction loss using input x (data) and parameters obtained from inference.

Forward pass returns: (Reconst_loss + Local_KL, Global_KL)

Abstract
High-level Class $\boxed{\text{Trainer}}$ that Unsupervised Trainer Inherits from:

↳ Mostly has training and perf related methods e.g. to compute metrics on the posterior distribution.

+ Methods to both CORRUPT and UNCORRUPT Posteriors - adding noise to the posterior

Differentiate between inference method inside VAE and INFERENCE module that has a file called Inference.py.

MODELS.MODULES – Children of nn.Module

$$\text{nn.Module} \xrightarrow{\text{inherits}} \text{VAE}$$

VAE → VAEC
VAE → VAEF
VAE → SCANVI

nn.Module → Decoder
nn.Module → DecoderSCVI
nn.Module → Encoder
nn.Module → FCLayers (fully connected layers for a neural network)
nn.Module → Classifier

$$\text{FCLayers} \xrightarrow{\text{used}} \text{Classifier}$$

Classifier – Classifier: composed of FC layers (arbitrary amount, by default using batch normalization). final layer in Softmax that can infer the output dimension.

MERJA: Need Corruption; don't need differential expression

TOTAL LOSS = binary cross entropy + KL-divergence

reconstruction Loss

CHRIS

encode: return relu-activated μ, σ

reparam: returns sample from posterior $q(z)$

FORWARD

encode $\xrightarrow{\mu, \sigma}$ reparam $\xrightarrow{z}$ return (decode(z), μ, Log(σ))

returns reconstr + variational dist params

+ Loss

FORWARD PASS

VAE

using encoder

z-encoder, L-encoder

get-latents → given (x, y), sample from $q(z)$

forward calls inference

from our VAE to PBMC

$\log var = \log(\sigma^2)$ ← Log var used because of numerical stability?

$\mathrm{Logvar.exp}() = \sigma^2$

$\log(\sigma^2) = \frac{1}{2} \log var$

Variational Lower bound: $\boxed{\mathcal{L} = E_q\left[\log \frac{P(X,Z)}{q(Z)}\right] = E_q\left[\log P(Z,X) - \log q(Z)\right]}$

$$D_{KL}\left(q(z)\,||\,P(Z|X)\right) = -\mathcal{L} + \log P(X) = -E_q\left[\log P(Z,X) - \log q(Z)\right] + \log P(X)$$

In our case:

$$-E_q\left[\log p(x|z)\,p(z) - \log q(z|x)\right] + \log P(X) =$$ ← DROP

$$E_q\left[\log\left[\mathcal{N}(f(z)[0], f(z)[1]) \cdot \mathcal{N}(0, I_2)\right]\right] + E_q\left[\log\left[\mathcal{N}(g(x)[0], g(x)[1])\right]\right] \pm$$

LOSS: → Bin.crentr(data, reconstruction) = bce

→ $KLD = \left[-\frac{1}{2} \cdot \sum\left(1 + \log var - \mu^2 - \sigma^2\right)\right] / \mathrm{norm}$ ← If data is binary, KLD has closed form

RETURN: bce + KLD

THEORY):

$$D_{KL}(\mathcal{N}_0 \| \mathcal{N}_1) = \frac{1}{2}\left[tr\left(\Sigma_1^{-1}\Sigma_0\right) + (\mu_1-\mu_0)^T \Sigma_1^{-1}(\mu_1-\mu_0) - k + Ln\left(\frac{\det \Sigma_1}{\det \Sigma_0}\right)\right]$$

In variational inference:

$$D_{KL}\left(\mathcal{N}((\mu_1,\dots,\mu_k)^T, diag(\sigma_1^2,\dots,\sigma_k^2)) \| \mathcal{N}(0,I)\right) = \frac{1}{2}\sum_{i=1}^{k}\left(\sigma_i^2 + \mu_i^2 - Ln(\sigma_i^2) - 1\right)$$

$$= \frac{1}{2}\sum\left(\sigma_i^2 + \mu_i^2 - \text{Logvar} - 1\right)$$

$= D_{KL}($

6 VIDEOS:

0: Introduction to SCVI as Dimensionality Reduction Problem + all about latents

1 Problem Setup and Mathematical backg.

2. PyTorch: nn.Linear

3. DEMO: Running the Network

ENCODER

DECODER

w/ new computer

4. Math Details:
- REPARAM TRICK
- Loss fun.
- ELBO

5. Comparison with SCVI ones

MAIN MODULE : INFERENCE

POSTERIOR ≡ functional DATASET = MODEL + DATASET + + DATA LOADER + + SAMPLER

IWIH

SO POSTERIOR Inference

↳ POSTERIOR Instance is used as an Attribute, it's like a data set, e.g.

↳ TRAIN_SET or TEST_SET

MAIN METHODS:

- ↳ LL and marginal_ll to compute likelihoods || Find Entropy of Batch Mixing
- ↳ getting the latent space
- ↳ Finding Differential_Expression Statistics and Score, also for ∀ cluster
- ↳ Batch Mixing Entropy

Additional: Different scores, TSNE, Batch Mixing 'Entropy', finding Entropy from indices