S&DS 365 / 665

Intermediate Machine Learning

Variational Inference and VAEs

March 9

Yale

Reminders

- Assignment 2 due today at midnight
- Practice midterm posted tomorrow
- Multiple review sessions

For Today

- Variational inference: The ELBO
- Derivations (partial) and examples
- Variational autoencoders (VAEs)

Inverting generative models

Template for generative model:

1. Choose Z
2. Given z, generate (sample) X

We often want to invert this:

1. Given x
2. What is Z that generated it?

Inverting models

Bayesian setup:

1. Choose θ
2. Given θ, generate (sample) X

Posterior inference:

1. Given x
2. What is θ that generated it?

Approximate inference

If we have a random vector $Z \sim p(Z \mid x)$, we might want to compute the following:

- marginal probabilities $\mathbb{P}(Z_i = z \mid x)$
- marginal means $\mathbb{E}(Z_i = z \mid x)$
- most probable assignments $z^\star = \arg\max_z \mathbb{P}(\{Z_i = z_i\} \mid x)$
- maximum marginals $z_i^\star = \arg\max_{z_i} \mathbb{P}(Z_i = z_i \mid x)$
- joint probability $\mathbb{P}(Z \mid x)$
- joint mean $\mathbb{E}(Z \mid x)$

Each of these quantities is intractable to calculate exactly, in general.

Variational methods

- Gibbs sampling is *stochastic* approximation
- Variational methods iteratively refine *deterministic* approximations
- Variational and Markov chain approximations originated in physics

Example 1: Interacting particles

We have a graph with edges E and vertices V. Each node i has a random variable Z_i that can be "up" (Z_i) or "down" ($Z_i = 0$)

$$\mathbb{P}_\beta(z_1, \ldots, z_n) \propto \exp\left(\sum_{s \in V} \beta_s z_s + \sum_{(s,t) \in E} \beta_{st} z_s \, z_t\right)$$

This is called an "Ising model" and is central to statistical physics.

Example 1: Interacting particles

We have a graph with edges E and vertices V. Each node i has a random variable Z_i that can be “up” (Z_i) or “down” ($Z_i = 0$)

$$\mathbb{P}_\beta(z_1, \ldots, z_n) \propto \exp\left(\sum_{s \in V} \beta_s z_s + \sum_{(s,t) \in E} \beta_{st} z_s\, z_t\right)$$

E are the set of edges, V are the vertices. Imagine the Z_i are votes of politicians, and the edges encode the social network of party affiliations

Stochastic approximation

Gibbs sampler

Iterate until converged:

1. Choose vertex $s \in V$ at random
2. Sample z_s holding others fixed

$$\theta_s = \text{sigmoid}\left(\beta_s + \sum_{t \in N(s)} \beta_{st} z_t\right)$$
$$Z_s \mid \theta_s \sim \text{Bernoulli}(\theta_s)$$

sigmoid$(u) = 1/(1 + e^{-u})$

Determimnistic approximation

Mean field variational algorithm

Iterate until converged:

1. Choose vertex $s \in V$ at random
2. Update mean μ_s holding others fixed

$$\mu_s = \text{sigmoid}\left(\beta_s + \sum_{t \in N(s)} \beta_{st}\, \mu_t\right)$$

sigmoid$(u) = 1/(1 + e^{-u})$

Determimnistic vs. stochastic approximation

- The z_s variables are random
- The μ_s variables are deterministic
- The Gibbs sampler convergence is in distribution
- The mean field convergence is numerical
- The Gibbs sampler approximates the full distribution
- The mean field algorithm approximates the mean of each node

Think of how to interpret this with Z_s the vote of politician s

Example 2: A finite mixture model

Fix two distributions F_0 and F_1, with densities $f_0(x)$ and $f_1(x)$, and form the mixture model

$$\theta \sim \text{Beta}(\alpha, \beta)$$
$$X \mid \theta \sim \theta F_1 + (1-\theta) F_0.$$

The likelihood for data $x_1, \ldots, x_n$ is

$$p(x_{1:n}) \;=\; \int_0^1 \text{Beta}(\theta \mid \alpha, \beta) \prod_{i=1}^{n} \left(\theta f_1(x_i) + (1-\theta) f_0(x_i)\right) \, d\theta.$$

Our goal is to approximate the posterior $p(\theta \mid x_{1:n})$

Stochastic approximation

Gibbs sampler

1. Sample $Z_i \mid \theta, x_{1:n}$ for $i = 1, \ldots, n$
2. Sample $\theta \mid z_{1:n}, x_{1:n}$

The first step is carried out by sampling

$$Z_i = \begin{cases} 1 & \text{with probability} \propto \theta f_1(x_i) \\ 0 & \text{with probability} \propto (1-\theta) f_0(x_i) \end{cases}$$

Stochastic approximation

Gibbs sampler

1. Sample $Z_i \mid \theta, x_{1:n}$ for $i = 1, \ldots, n$
2. Sample $\theta \mid z_{1:n}, x_{1:n}$

The second step is carried out by sampling

$$\theta \sim \text{Beta}\left(\sum_{i=1}^{n} z_i + \alpha, n - \sum_{i=1}^{n} z_i + \beta\right).$$

Posterior over θ is approximated as *mixture* of Beta distributions; number of components is $n + 1$

Variational inference: Strategy

- We'd like to compute $p(\theta, z \mid x)$, but it's too complicated.
- Strategy: Approximate as $q(\theta, z)$ that has a "nice" form
- q is a function of variational parameters, optimized for each x.
- Maximize a lower bound on $p(x)$.

Variational inference: The ELBO

The ELBO is the following lower bound on $\log p(x)$:

$$
\begin{aligned}
\log\ p(x) &= \int \sum_z q(z,\theta) \log p(x)\, d\theta \\
&= \sum_z \int q(z,\theta) \log \left(\frac{p(x,z,\theta)\, q(z,\theta)}{p(z,\theta \,|\, x)\, q(z,\theta)} \right)\ d\theta \\
&= \sum_z \int q(z,\theta) \log \left(\frac{p(x,z,\theta)}{q(z,\theta)} \right) d\theta + \sum_z \int q(z,\theta) \log \left(\frac{q(z,\theta)}{p(z,\theta \,|\, x)} \right) d\theta \\
&\geq \sum_z \int q(z,\theta) \log \left(\frac{p(x,z,\theta)}{q(z,\theta)} \right) d\theta \\
&= H(q) + \mathbb{E}_q\big(\log p(x,z,\theta)\big)
\end{aligned}
$$

We maximize this over the parameters of q

Variational inference: The ELBO

The inequality above uses concavity of the logarithm:

$$\log\left(\sum_\alpha w_\alpha x_\alpha\right) \geq \sum_\alpha w_\alpha \log x_\alpha$$

So, if $q_\alpha \geq 0$ and $p_\alpha \geq 0$ sum (or integrate) to one, then

$$0 = \log\left(\sum_\alpha p_\alpha\right) = \log\left(\sum_\alpha q_\alpha \frac{p_\alpha}{q_\alpha}\right) \geq \sum_\alpha q_\alpha \log\left(\frac{p_\alpha}{q_\alpha}\right)$$

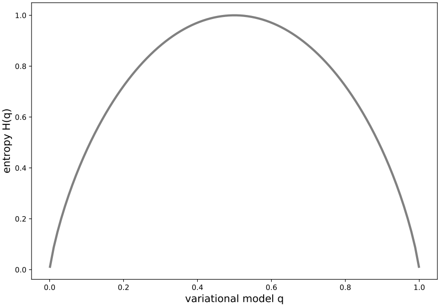
Therefore

$$\sum_\alpha q_\alpha \log\left(\frac{q_\alpha}{p_\alpha}\right) \geq 0$$

Variational inference: The ELBO

The ELBO is $H(q) + \mathbb{E}_q(\log p)$

The entropy term $H(q)$ encourages q to be spread out:



The cross-entropy $\mathbb{E}_q \log p$ tries to match q to p

Example 2: A finite mixture model

Fix two distributions F_0 and F_1, with densities $f_0(x)$ and $f_1(x)$, and form the mixture model

$$\theta \sim \text{Beta}(\alpha, \beta)$$
$$X \mid \theta \sim \theta F_1 + (1-\theta) F_0.$$

The likelihood for data $x_1, \ldots, x_n$ is

$$p(x_{1:n}) \; = \; \int_0^1 \text{Beta}(\theta \mid \alpha, \beta) \prod_{i=1}^{n} \left(\theta f_1(x_i) + (1-\theta) f_0(x_i)\right) \, d\theta.$$

Our goal is to approximate the posterior $p(\theta \mid x_{1:n})$

Variational approximation

Our variational approximation is

$$q(z,\theta) = q(\theta \mid \gamma_1, \gamma_2) \prod_{i=1}^{n} q_i^{z_i}(1 - q_i)^{(1-z_i)}$$

where $q(\theta \mid \gamma_1, \gamma_2)$ is a Beta(γ_1, γ_2) distribution, and $0 \leq q_i \leq 1$ are n free parameters.

Need to maximize ELBO $H(q) + \mathbb{E}_q \log p$

Let's sketch part of the calculation

Variational approximation

First, we have

$$\log p(x, \theta, z) = \log p(\theta \mid \alpha, \beta) + \sum_{i=1}^{n} \left\{ \log\left(\theta^{z_i} f_1(x_i)\right) + \log\left(\theta^{1-z_i} f_0(x_i)\right) \right\}$$

Next we use identities such as

$$\mathbb{E}_q \log \theta = \psi(\gamma_1) - \psi(\gamma_1 + \gamma_2)$$

for the digamma function $\psi(\cdot)$.

After some calculus and algebra  , we end up with the following algorithm

Variational algorithm for mixture

Variational inference

Iterate the following steps for variational parameters $q_{1:n}$ and (γ_1, γ_2):

1. Holding q_i fixed, set $\gamma = (\gamma_1, \gamma_2)$ to

$$\gamma_1 = \alpha + \sum_{i=1}^{n} q_i \qquad \gamma_2 = \beta + n - \sum_{i=1}^{n} q_i$$

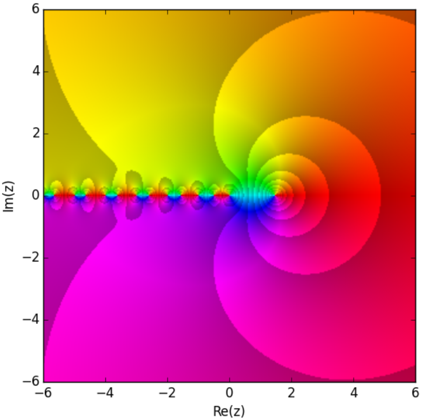
2. Holding γ_1 and γ_2 fixed, set q_i to

$$q_i = \frac{f_1(x_i) \exp \psi(\gamma_1)}{f_1(x_i) \exp \psi(\gamma_1) + f_0(x_i) \exp \psi(\gamma_2)}$$

After convergence, approximate posterior distribution over θ is

$$\widehat{p}(\theta \mid x_{1:n}) = \text{Beta}(\theta \mid \gamma_1, \gamma_2)$$

Digamma function



$\psi(x)$ is the *digamma function*

`https://en.wikipedia.org/wiki/Digamma_function`

Deterministic approximation

- Convergence is numerical, not stochastic
- Posterior is approximated as a *single* Beta
- Very similar algorithm is used for topic models distribution

Example 3: More general mixtures

$$\theta \sim \text{Dirichlet}(\alpha_1, \ldots, \alpha_k)$$

$$X \mid \theta \sim \theta_1 F_1 + \cdots + \theta_k F_k$$

The likelihood for single data point x is

$$p(x) \;=\; \int \text{Dirichlet}(\theta \mid \alpha_1, \ldots, \alpha_k) \left(\sum_{j=1}^{k} \theta_j f_j(x) \right) \, d\theta.$$

When distributions F_j are learned, this is a “topic model.” Variational inference is one of the most useful ways of training topic models

Variational autoencoders

- Variational autoencoders are generative models that are trained using variational inference
- The "decoder" is a neural net that generates from a latent variable
- The "encoder" approximates the posterior distribution with another neural network trained using variational inference

Variational autoencoders

Start with a generative model

$$z \sim N(0, I_K)$$
$$x \mid z = G(z)$$

$G(z)$ is the *generator network* or *decoder*

For example, use a 2-layer network

$$G(z) = A_2 \mathrm{ReLU}(A_1 z + b_1) + b_2$$

Posterior inference

How do we train the generative network?

$$
\begin{aligned}
Z &\sim N(0, I_k) \\
X \mid z &= G(z)
\end{aligned}
$$

We want the posterior distribution $p(z \mid x)$

But this is intractable, because $G(\cdot)$ is nonlinear

Approach: Use variational inference

Using variational inference

For variational inference we take

$$q(z \mid x) = N(\mu(x), \sigma^2(x) I_k)$$

where now $\mu_j(x)$ and $\sigma_j^2(x)$ are the *variational parameters* for $j = 1 \ldots, k$.

Using neural networks

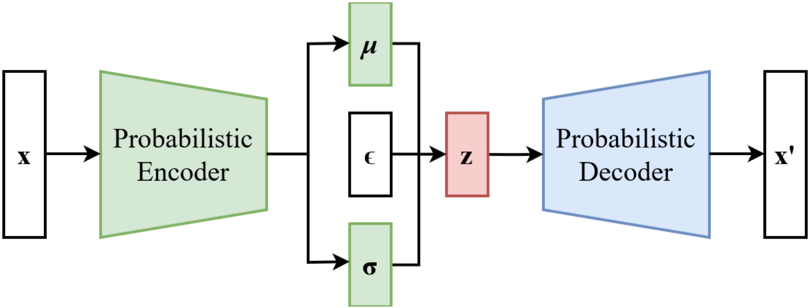
Rather than estimate $\mu(x)$ for each x, we build an encoder neural network that outputs the mean and variance.

For example:

$$\mu(x) = B_2 \, \mathrm{ReLU}(B_1 x + d_1) + d_2$$

and similarly for $\log \sigma^2(x)$.

VAE architecture



Training the neural networks

Decoder network: $z \mapsto x$, weights A, b

Encoder network: $x \mapsto \mu(x),\ \log \sigma^2(x)$, weights B, d

Train both networks simultaneously, to maximize the ELBO

- Decoder trained with $\mathbb{E}_q \log p(x, Z_s)$ over weights A, b
- Encoder trained with $H(q) + \mathbb{E}_q \log p(x, Z_s)$ over weights B, d

---

The entropy of a multivariate Gaussian with covariance Σ is $\frac{1}{2} \log |\Sigma| + \text{constant}$

The entropy term here is $\log \sigma^2(x)$, which is taken to be an output of the encoder network.

Using neural networks

Now, approximate $\mathbb{E}_q(\log p(x, Z))$ by sampling (weak law of large numbers)

$$\mathbb{E}_q(\log p(x, Z)) \approx \frac{1}{N} \sum_{s=1}^{N} \log p(x, Z_s)$$

Problem: The parameters of the recognition network have disappeared!

Solution: Reparameterize the samples by $Z_i = \mu(x) + \sigma(x)\epsilon_i$ where $\epsilon_i \sim N(0, I_k)$.

This is called "the reparameterization trick." D. Kingma and M. Welling, "Autoencoding Variational Bayes," `https://arxiv.org/abs/1312.6114`.

Simple example

Suppose $x \mid z \sim N(G(z), I)$ where generator network is

$$G(z) = \mathrm{ReLU}(Az + b).$$

Then $-\log p(x \mid z)$ is

$$\frac{1}{2}\|x - \mathrm{ReLU}(Az + b)\|^2$$

Simple example (continued)

Next, suppose the approximate posterior is

$$q(z \mid x) = N(\mu(x), \sigma^2(x) I_k)$$

where encoder network is $\mu(x) = \mathrm{ReLU}(Bx + d)$. Then

$$-\mathbb{E}_q \log p(x \mid Z) = \frac{1}{N} \sum_{s=1}^{N} \frac{1}{2} \|x - \mathrm{ReLU}(AZ_s + b)\|^2$$

$$\stackrel{d}{=} \frac{1}{N} \sum_{s=1}^{N} \frac{1}{2} \|x - \mathrm{ReLU}(A\,(\mathrm{ReLU}(Bx + d) + \epsilon_s \sigma(x)) + b)\|^2$$

Overall training objective

The (negative) ELBO objective is then

$$\sum_{i=1}^{n} \left\{ \frac{1}{N} \sum_{s=1}^{N} \frac{1}{2} \|x_i - \mathrm{ReLU}(A\left(\mathrm{ReLU}(Bx_i + d) + \epsilon_s \sigma(x_i)\right) + b)\|^2 - \log \sigma^2(x_i) \right\}$$

- The entropy term serves to “spread out” the Gaussian in the variational approximation
- This objective is optimized by alternating stochastic gradient steps on the weights in the decoder and encoder networks

Demo

Variational Autoencoder on MNIST data

This is a Tensorflow implementation of Variational Autoencoder (VAE) on MNIST data, based on *Auto-Encoding Variational Bayes* (Kingma and Welling 2014).

It uses probabilistic encoders and decoders realized by Multilayer Perceptrons (MLP) with a single hidden layer. The VAE was trained incrementally with mini-batches using partial fit.

The MNIST dataset, distributed by Yann Lecun's THE MNIST DATABASE of handwritten digits website, consists of pair "handwritten digit image" and "label". The image is a gray scale image with 28 x 28 pixels. Pixel values range from 0 (black) to 255 (white), scaled in the [0, 1] interval. The label is the actual digit, ranging from 0 to 9, the image represents.

The original notebook was composed by Jan Hendrik Metzen. Modifications were made by Sunnie Kim on May 24th, 2018.

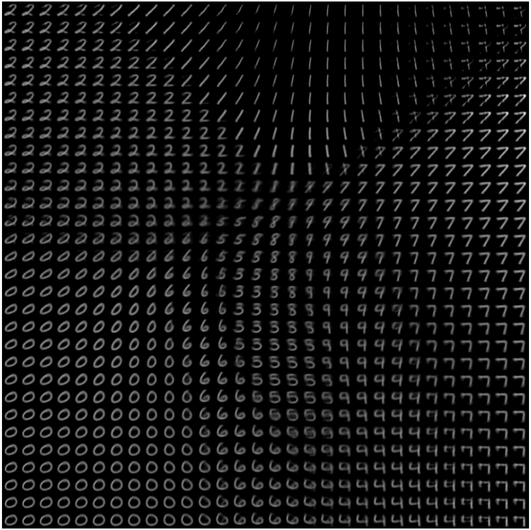
In [1]:
```
import numpy as np
import tensorflow as tf

import matplotlib.pyplot as plt
%matplotlib inline

np.random.seed(0)
tf.set_random_seed(0)
```

Load MNIST data in a format suited for Tensorflow

The 'input_data' script is available at: https://raw.githubusercontent.com/tensorflow/tensorflow/master/tensorflow/examples/tutorials/mnist/input_data.py

In [2]:
```
from tensorflow.examples.tutorials.mnist import input_data
tf.logging.set_verbosity(tf.logging.ERROR)
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
n_samples = mnist.train.num_examples
```

Visualizing a 2-dim latent space



Summary

- Gibbs sampling makes stochastic approximations
- Variational methods make deterministic approximations
- General recipe: Maximize ELBO over variational parameters
- VAEs: Variational mean is output of a second neural network
- Gives a powerful approach to generative modeling