NN/VI/Autoencoder Reading Group Notes

Evan Ott
evan.ott@utexas.edu

Last updated: October 30, 2017

Outline

- Variational inference (James)
- Stochastic variational inference (Michael)
- NN and backpropagation (Yuguang)
- Introduction to TensorFlow (Mo and Evan) – actual hello world (like a single layer perceptron or a linear model) – leave out complicated things like dropout, etc.
- Classical autoencoders (Mauricio)
- Variational autoencoders (Jennifer)

We'll have a GitHub page / Google Doc for papers, examples, talks, code, etc.

That's at https://github.com/jestarling/DeepProbModels

2017-09-11: Variational Inference (James)

"Variational Bayes for Idiots"

Today is a basic introduction to variational inference.

Observed data X and hidden variables Z (could be parameters for the whole dataset – like means and variances of mixture components of a GMM – or the per-data-point hidden variable – like an indicator for each datapoint being in a cluster).

$$p(x, z) = p(x|z) \cdot p(z)$$

$$\text{Bayes: } p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

$$\text{Problem: } p(x) = \int_Z p(x, z)dz$$

$p(x)$ is challenging or impossible to compute.

Idea:

- posit some family of approximations, $\mathcal{Q}$
- Find the member of $\mathcal{Q}$ that is "closest" to $p(z|x)$ in KL-divergence (or other measure)

  Formally:

$$q^*(z) = \arg\min_{q(z)\in\mathcal{Q}} KL\left(q(z)\|p(z|x)\right) \tag{1}$$

  where

$$\begin{aligned} KL\left(q(z)\|p(z|x)\right) &= \mathbb{E}_q(z)\left[\log \frac{q(z)}{p(z|x)}\right] \\ &= \mathbb{E}_q\left[\log q(z)\right] - \mathbb{E}_q\left[\log p(z|x)\right] \end{aligned} \tag{2}$$

This is a calculus of variations problem (like the brachistochrome problem), where this will eventually lead to a vector optimization problem that's standing for a functional optimization problem.

This also will give us point estimates that are often better than running a Gibbs sampler, and be faster. KL will favor putting probability mass where $p(z|x)$ is large, so that the variance of the approximated distribution is often smaller than the variance of the true posterior. If we flipped the ratio in equation 2, we would instead be looking at Expectation Propagation.

Example: One-Hot Encoding

$$\begin{aligned}(x_i|\mu, c_i) &\sim N(c_i^\top \mu, 1)\\ \text{e.g., } c_i &= (0,0,1,0,0) \in \{0,1\}^k\\ \mu &\in \mathbb{R}^k\\ \mu_k &\sim N(0, \tau^2)\end{aligned}$$

Joint of data, parameters:

$$p(\mu, c, x) = p(\mu)\prod_{i=1}^{N} p(c_i)p(x_i|\mu, c_i)$$

Marginal or "Evidence": $p(x) = \int_{\mu,c} p(x, \mu, c)p(\mu, c)d\mu dc$

ELBO

ELBO: Evidence lower bound.

$\mathcal{Q}$: family of approximations.

Each $q(z) \in \mathcal{Q}$ is a candidate.

$\arg\min_{q(z)\in\mathcal{Q}} KL\left(q(z)\|p(z|x)\right)$ is not computable:

$$\mathbb{E}_q\left[\log q(z)\right] - \mathbb{E}_q\left[\log p(z|x)\right] = \mathbb{E}_q\left[\log q(z)\right] - \mathbb{E}_q\left[\log p(x, z) - \log p(x)\right]$$

But we can't compute $p(x)$, so we can't compute it, but the term is not dependent on the choice of z. So, we ignore that term, and want to maximize its negative (the ELBO):

$$\begin{aligned}ELBO(q) &= \mathbb{E}_q\left[\log p(x, z)\right] - \mathbb{E}_q\left[\log q(z)\right]\\ &= -KL\left(q(z)\|p(z|x)\right) + \log(p(x))\\ &\leq \log(p(x))\end{aligned}$$

So we can re-write this as:

$$\begin{aligned}ELBO(q) &= \mathbb{E}_q\left[\log p(x|z)\right] + \mathbb{E}_q\left[\log p(z)\right] - \mathbb{E}_q\left[\log q(z)\right]\\ &= \mathbb{E}_q\left[\log p(x|z)\right] - KL\left(q(z)\|p(z)\right)\end{aligned}$$

So we're making the likelihood as large as possible, and making the approximate distribution that penalizes moving away from the prior. In other words, be close to the data, and don't stray far from the prior.

Note

Carlos asked a question about not having to do the expectation of the log posterior with respect to q to get something more like a MAP estimate. But a reasonable compromise is doing a maximum marginal a posteriori estimate where we marginalize over the local variables in the model (the cluster assignments in a GLM), but keep the global parameters to find the MAP. That's a nice sort of Bayesian thing to do.

Mean-field family

This is a family of approximations where the correlation structure is completely independent.

$$\mathcal{Q} = \left\{ q(z) : q(z) = \prod_{j=1}^{M} q_j(z_j) \right\}$$

This is not a model of the data (there's no x in that). But we're going to connect it to the data through the ELBO. Now, we have options for each q_j, where we might take a nice parametric form. We might have Gaussians for location parameters or inverse gammas for scale parameters. But for some models where the complete conditionals are exponential, we can find optimal approximation to take for each q_j.

Coordinate ascent

Not totally different from Gibbs sampling, only we're doing optimization instead of probabilistic draws. We'll do one latent variable at a time, such as all the components one after another, then each mean in a GMM. Then, start over.

Consider the jth latent variable z_j with complete conditional $p(z_j|z_{-j}, x)$. With z_{-j} fixed, the optimal q_j (that makes ELBO as large as possible) is of the form:

$$q_j^*(z_j) \propto \exp\left\{\mathbb{E}_{-j}\left[\log p(z_j|z_{-j}, x)\right]\right\} \tag{3}$$

Or, can write in terms of the joint (the constants will go away). The expectation is with respect to the variational density over all the other parameters, i.e.,

$$z_{-j} \sim \prod_{l \neq j} q_l(z_l)$$

So, we set $q_j^*(z_j)$ for each j, cycle through, update, etc. until it converges.

2017-09-18: Variational Inference (James)

CAVI

Pick up from last time on Coordinate Ascent Variational Inference (CAVI).

Consider z_j. Complete conditional is $p(z_j|z_{-j}, x)$. Fact: with z_{-j} fixed, the optimal $q_j(z_j)$ is of the form

$$\log q_j^*(z_j) = \mathbb{E}_{-j} \log(p(z_j|z_{-j,x}))$$

where $\mathbb{E}_{-j}$ refers to taking the expectation over $q_{-j}(z_{-j}) = \prod_{l \neq j} q_l(z_l)$.

So to do CAVI, cycle through $j = 1, \ldots, m$, each time updating $q_j(z_j) = q_j^*(z_j)$ until no improvements are made to the ELBO.

Proof of q_j^*

$$ELBO(q) = \mathbb{E}_q[\log p(x, z)] - \mathbb{E}_q[\log q(z)]$$

We want to isolate $q_j(z_j)$:

$$\begin{aligned} ELBO(q_j) &= \mathbb{E}_{q_j}\left[\mathbb{E}_{q_{-j}}\left[\log p(x, z_j, z_{-j})\right]\right] - \mathbb{E}_{q_j}\left[\log q_j(z_j)\right] + \text{ constant} \\ &= \mathbb{E}_{q_j}\left[\log q_j^*(z_j)\right] - \mathbb{E}_{q_j}\left[\log q_j(z_j)\right] \\ &= -KL\left(q_j \| q_j^*\right) - \mathbb{E}_{q_j}\left[\log q_j(z_j)\right] \end{aligned}$$

Which guarantees that q_j^* is optimal, because we are minimizing the KL divergence between q_j and q_j^*.

GMM

$$
\begin{aligned}
x_i &\in \mathbb{R} \quad i = 1, \ldots, N \\
\mu_k &: k^{\text{th}} \text{ mean} \quad k = 1, \ldots, L \\
c_i &: \text{indicator vector (“one hot encoding”)} \\
x_i|\mu &\sim N(c_i^\top \mu, 1) \\
\mu_k &\sim N(0, \tau^2)
\end{aligned}
$$

For now, set τ as fixed, to focus on inference for μ and c.

Mean-field family:

$$
\begin{aligned}
q(z) &= \prod_j q_j(z_j) \\
&= \prod_{l=1}^{L} q_k(\mu_k) \cdot \prod_{i=1}^{N} q_i(c_i) \\
&= \prod_{l=1}^{L} N(\mu_k|m_k, s_k^2) \cdot \prod_{i=1}^{N} \text{Multinomial}(c_i|\phi_i)
\end{aligned}
$$

Now, go through the ELBO, not really different from having to derive a Gibbs sampler.

$$
\begin{aligned}
ELBO(q) &= \mathbb{E}_q \left[\log \left(\prod_{i=1}^{N} N(x_i|c_i^\top \mu, 1) \cdot p(\mu, c) \right) \right] - \mathbb{E}_q \left[\log \left(\prod_{l=1}^{L} q(\mu_k|m_k, s_k^2) \right) + \log \left(\prod_{i=1}^{N} q(c_i|\phi_i) \right) \right] \\
&= \sum_{k=1}^{L} \mathbb{E}_q \left[\log p(\mu_k); m_k, s_k^2 \right] + \sum_{i=1}^{N} \mathbb{E}_q \left[\log p(c_i); \phi_i \right] + \sum_{i=1}^{N} \mathbb{E}_q \left[\log p(x_i|c_i, \mu); \phi_i, m, s^2 \right] - \sum_{i=1}^{N} \mathbb{E}_q \left[\log q_i(c_i; \phi_i) \right] - \sum_{k=1}^{L} \mathbb{E}_q \Big[
\end{aligned}
$$

So the update for c_i is

$$
\begin{aligned}
q^*(c_i) &= \log p(c_i) + \mathbb{E}_{-c_i} \left[\log p(x_i|c_i, \mu); m, s^2 \right] \\
\log p(c_i) &= \log \left(\frac{1}{K} \right) = \text{ constant in } c_i \\
\mathbb{E} \left[\log p(x_i|c_i, \mu); m, s^2 \right] &= \mathbb{E} \left[-\frac{1}{2} \sum_{k=1}^{L} c_{ik} (x_i - \mu_k)^2; m, s^2 \right] \\
&= \mathbb{E} \left[-\frac{1}{2} \sum_{k=1}^{L} c_{ik} (x_i^2 - 2x_i\mu_k + \mu_k^2); m, s^2 \right] \\
&= \sum_{k=1}^{L} \left[\mathbb{E} \left(-\frac{1}{2} c_{ik} x_i^2 \right) + \mathbb{E} \left(c_{ik} \mu_k \right) x_i - \frac{1}{2} \mathbb{E}(c_{ik} \mu_k^2) \right] \\
&= \text{const} + \sum_{k=1}^{L} \left[c_{ik} x_i \mathbb{E}[\mu_k] - \frac{1}{2} c_{ik} \mathbb{E}[\mu_k^2] \right] \\
&= \sum_{k=1}^{L} c_{ik} \left[x_i \mathbb{E}[\mu_k] - \frac{1}{2} \mathbb{E}[\mu_k^2] \right] \\
\mathbb{E}[\mu_k] &= m_k \\
\mathbb{E}[\mu_k^2] &= s_k^2 + m_k^2 \\
q^*(c_i|\phi_i) &\propto \prod_{k=1}^{L} (e^{\psi_{ik}})^{c_{ik}} \\
\psi_{ik} &= x_i m_k - \frac{1}{2}(s_k^2 + m_k^2)
\end{aligned}
$$

The above was not edited and may contain errors or omissions.

In the GitHub repo, see the Variational Inference section.

2017-09-25: Stochastic Variational Inference (Michael)

Basic idea: instead of using the full dataset, use samples. In general, the posterior is hard. Want to minimize KL divergence, or maximize ELBO.

Today, working with GMM again. Make the mean-field approximation, and have it factor nicely. Then, get exponential family:

$$P(X|\theta) = h(X) \exp\left[\sum_i \eta_i(\theta) T_i(X) - A(\theta)\right]$$

Where $\eta(\theta)$ is the "natural parameter." If complete conditional and variational approximation forms are both in the same exponential family, everything is easy. Below, λ will be variational for β and ϕ_n will be variational for z, the indicators.

Typically:

$$\lambda^{(t)} = \lambda^{(t-1)} + \rho \nabla f(\lambda)$$

But instead, move with the "natural gradient" (rather than the gradient as-is). A motivating example is looking at pairs of normal distributions: $N(0, 10000)$ and $N(10, 10000)$ are obviously close together, whereas $N(0, 0.1)$ and $N(10, 0.1)$ are obviously far apart. The natural gradient moves in directions that are roughly constant length in "information space" rather than Euclidean space. It's not altogether different from comparing the gradient and the Newton direction.

The natural gradient is the inverse Fisher information matrix times the standard gradient (notation from http://andymiller.github.io/2016/10/02/natural_gradient_bbvi.html):

$$F_\lambda = \mathbb{E}_{q(\cdot;\lambda)} \left[\nabla_\lambda \ln q(x;\lambda) \left(\nabla_\lambda \ln q(x;\lambda)\right)^\intercal\right]$$
$$\tilde{\nabla}_\lambda = F_\lambda^{-1} \nabla_\lambda \mathcal{L}(\lambda)$$

Gradient: Pick a coordinate direction and maximize the difference between where you are and where you're going with a particular Euclidean step size.

Natural gradient: do the same thing, but with a particular KL-divergence step size.

However, the natural gradient for exponential family is much easier:

$$\hat{\nabla} f(\lambda) = \mathbb{E}_Q \left[\eta(\cdot)\right] - \lambda$$
$$\lambda^{(t)} = (1-\rho)\lambda^{(t-1)} + \rho \mathbb{E}_Q \left[\eta(\cdot)\right]$$

So in practice, we could run on individual data points, but instead minibatches work well. They're unbiased, but a subsample:

1. Select size M minibatch of data
2. For $n = 1 \cdots M$ set $\phi_n = \mathbb{E}_{Q(\lambda)} \left[\eta(\cdot)\right]$
3. For $n = 1 \cdots M$ $\hat{\lambda}_n = \mathbb{E}_{Q(\phi)} \left[\eta(\cdot)\right]$ with estimate of gradient which is the best guess for the parameter based on a single example.
4. Update $\lambda^{(t)} = (1-\rho_t)\lambda^{(t-1)} + \frac{\rho_t}{M} \sum_{n=1}^M \hat{\lambda}_n$

$$x_n \sim N(z_n^\top \mu, \sigma^2)$$
$$\mu \sim N(\mu_0)$$
$$z_n \sim \text{Multinomial}$$

So the variational approximation gives z as a Multinomial and μ as a Normal, both of which are in the exponential family.

For additional reference (because I could barely keep up):

miller2016natural
miller2016natural

hoffman2013stochastic
hoffman2013stochastic

"This is one of the most important papers in Bayesian inference in the last 10 years" –James.

Advice: do the algebra once for the GMM update, taking for granted the definition of the natural gradient for exponential families as something to do once.

2017-10-02: Neural Networks and Backpropagation (Yuguang)

Introduction

Proposed in 1943, based on rough outline of how the brain works.

Notation

- L total layers
- S_l number of perceptrons in lth layer
- w^l_{ji} weight from i in layer (l-1) to j in l
- z^l_i input of perceptron i in lth layer (including bias)
- σ activation function
- a^l_i activation of perceptron i in lth layer
- b^l_i bias of —
- C cost function: $C = \frac{1}{n}\sum_x C_x$ and $C = \frac{1}{2}\left\|y - a^L\right\|^2$
- $\odot$ Hadamard product

Forward step

Initialize weights with random values, this lets us pass forward, then later update

$$w^{l'}_{ji} = w^l_{ji} - \alpha\frac{\partial C}{\partial w^l_{ji}}$$

and similar for bias terms.

Define $\delta^l_i = \frac{\partial C}{\partial Z^l_i}$ where $Z^l_i = \sum_k w^l_{ik}a^{l-1}_k + b^l_i$ and $\delta^L = \nabla_a C \odot \sigma'(Z^L)$ and for $l < L$, $\delta^l = \left(\left[W^{L+1}\right]^\top \delta^{l+1}\right) \odot \sigma'(z^l)$

Then $\frac{\partial C}{\partial b^L_i} = \delta^L_i$ and $\frac{\partial C}{\partial w^L_{ji}} = a^{l-1}_i\delta^L_j$

Backward step

Do gradient update for last layer, then work backwards. Use a fancy application of the chain rule, using calculations we made on the forward step:

$$\delta_i^L = \frac{\partial C}{\partial Z_i^L} = \sum_k \frac{\partial C}{\partial a_k^L}\frac{\partial a_k^L}{\partial z_i^L}$$
$$a^L = \sigma(z^L) = \sigma(W^L a^{L-1} + b^L)$$

So that only a_i^L has a term with z_i^L. So we can simplify:

$$\delta_i^L = \frac{\partial C}{\partial Z_i^L} = \frac{\partial C}{\partial a_i^L}\frac{\partial a_i^L}{\partial z_i^L}$$
$$= \frac{\partial C}{\partial a_i^L}\sigma'(Z_i^L)$$

Do a similar chain rule for the hidden layers, bias terms. Eventually arrive at something that is just a function of things we computed on the forward pass.

2017-10-23: Classical autoencoders (Mauricio)

Mostly today, code on GitHub group.

Motivation

Data compression, denoising, hashing, feature extraction, transfer learning.

Description

Input and output are the same, just with a bottleneck layer (or set of layers). First part is encoder, second is decoder.

Similar in ideology to PCA, trying to find some "hyperplane" that explains data simply. PCA maximizes the variance of a linear combination of principal components. Solution comes from eigenvectors from estimated covariance matrix of X.

Instead, can show that PCA (where W has the eigenvectors) satisfies the reconstruction error problem:

$$\min_W \mathbb{E}\left((WW^\top X_i - X_i)^2\right) \Rightarrow \min_W \frac{1}{N}\sum_{i=1}^N \|WW' x_i - x_i\|^2$$

It's like having a single hidden layer where $W_1 = W$ and $W_2 = W'$, with no activation function (technically, identity).

Could do approximate PCA this way, but not necessarily orthogonal, only reasonable to use SGD on large data, but there are different PCA algorithms that would be efficient.

Instead, we allow any activation, any size of network, etc. Just have a bottleneck in the middle (that also gets a bias).

Idea from neuroscience (visual system): neurons fire for a specific pattern.

2017-10-30: Variational Autoencoders (Jennifer)

Data x, latent z, approximate $p(z|x)$ where $p(x) = \int p(x|z)p(z)dz$ is intractible.

Reminder, minimizing KL divergence is intractable, but maximizing ELBO will minimize it indirectly and tractably.

Going to use encoder $q_\phi(z|x)$, latent $z \sim N(\mu, \sigma I)$, decoder $p_\theta(x|z)$.

Use the reparameterization trick is $z = \mu + \sigma \odot \epsilon$ where $\epsilon \sim N(0, I)$. This lets you do backpropagation on μ and σ. Just do that draw once per training epoch / training. They are the output of the first neural network (the encoder) such that $(\mu, \sigma) = f(x|\phi)$. At the end, $\tilde{x} = f(z; \theta)$ with θ as the output parameters. Can always put a Bernoulli or Normal as the likelihood instead.

Going to use the -ELBO as the loss function we're optimizing to find ϕ and θ (the variational parameters), while noting the Bayesian (model) parameter z.