Deep Probabilistic Methods with PyTorch

A tutorial...

Chris Ormandy

April 25, 2018

Outline

LONGSHOT SYSTEMS

Outline

LONGSHOT SYSTEMS

Who Am I?

- Chris Ormandy
- Senior ML Researcher @ Longshot Systems
- We develop high throughput, low latency trading software for sports betting markets.
- We are a mix of developers and quants/researchers.
- Explicitly hiring for Senior Python developers.
- Keen to meet developers, quants, ML engineers and ML Researchers who feel their skillset could complement our use case.

Why PyTorch?

LONGSHOT SYSTEMS

- I have used both Tensorflow and PyTorch extensively.
- Both have Pro's and Con's.
- For me, PyTorch is heavily overlapping with numpy - I can leverage my experience with doing things in numpy.
- Conceptually easier - in tensorflow you have to care about computational graphs.
- I find it easier to use and faster to write.
- PyTorch is very new. All code in this presentation runs on bleeding edge, master source installed from github.
- Half the functionality I rely on doesn't exist in version 0.3 - so also consider this a preview of upcoming version 0.4!

Recap

Notation

- x is a 1d data point, $\mathbf{x}$ is a multi dimensional data point (a vector), $\mathbf{X}$ is a matrix of data, where rows are samples and columns are features. Formatting conventions follow for other variables too.
- y is a target or class label (depending on setting).
- w denotes a weight.

Resources

LONGSHOT SYSTEMS

http://github.com/chrisorm/pydata-2018

Prerequisites & Goals

Knowledge is a brick wall that you raise line by line forever

Prerequisites

- Probability Theory
- Basic Bayesian terminology
- Broad ML knowledge - Neural nets, regression etc.

Goals

- Recap some key ideas
- Cover VI, Discriminators, VAEs GANs
- Lots to cover, will probably require more work on your part!

Outline

Motivation

Doing Bayesian Inference!

- This talk is about doing approximate Bayesian Inference!
- E.g. The coin toss problem:
- If I toss a coin 20 times, and get 5 heads and 15 tails, what does that mean about the fairness of the coin?
- I am looking for a posterior distribution of the coin's bias - given I have seen some data, what is the probability it is biased?
- Most real world problems are not so simple, and we need to use approximations to answer this!
- Also called Variational Inference.

Motivation

What kind of problems is this useful for?

- In short - where we want to apply probabilistic reasoning.
- Uncertainty estimates for classifiers and neural nets.
- If we want true 'predictive' distributions that account for training data.
- If we are looking to understand empirical or heuristic methods.

Bayes Theorem

Simple Concepts

$$P(A|B) = \frac{P(B \mid A)\,\mathrm{P(A)}}{\mathrm{P(B)}}$$

Prior
Likelihood
Model Evidence
Posterior

Recap

Simple Concepts

- A likelihood function for a model with parameters, θ, is $P(y_i|\theta)$
- A prior gives our 'prior belief' about the distribution of a variable. Normally over the parameters, so can be written $P(\theta)$.
- A Posterior is our updated belief about the distribution over the variable in question, given the data we have now seen $P(\theta|y_i)$.
- Related via Bayes Theorem.

Outline

LONGSHOT SYSTEMS

What is a Divergence?

LONGSHOT SYSTEMS

Divegence

- Essentially a measure of the divergence of one distribution from another.
- At a high level, same idea as a distance measure, but distance obey:
  - $d(x,y) \geq 0$
  - $d(x,y) = d(y,x)$
  - $d(x,z) \leq d(x,y) + d(y,z)$
- Divergence measures may not have all of these.
- E.g. KL divergence not symmetric

KL Divergence

LONGSHOT SYSTEMS

Understanding the KLD

- Defined as: $KL(q||p) = E_q[log\frac{q(x)}{p(x)}]$
- Not symmetric; $KL(P \,||\, Q) \neq KL(Q \,||\, P)$.
- Different formulations induce different behaviours when learning Q.

KL Divergence

Example

Jupyter Notebook 1 - Understanding the KLD

Outline

Why Care about the posterior?

Predictive Distribution

$$P(y^* \mid \mathbf{x}^*) = \int P(y^* \mid \mathbf{w}, \mathbf{x}^*)\, P(\mathbf{w} \mid \mathbf{y}, \mathbf{X})\, d\mathbf{w}$$

Good for: **Out of Sample prediction and uncertainty estimates**

Model Evidence

$$P(\mathbf{y} \mid \mathbf{X}) = \frac{P(\mathbf{w},\mathbf{y}|\mathbf{X})}{P(\mathbf{w} \mid \mathbf{y}, \mathbf{X})}$$

Good for: **Hyper-Parameter Selection**

Why Approximate?

Why do we need to use approximations?

Bayes theorem tells us:

$$P(w \mid \mathbf{y}, \mathbf{X}) = \frac{P(w,\mathbf{y}|\mathbf{X})}{P(\mathbf{y}|\mathbf{X})}$$

$$P(\mathbf{y} \mid \mathbf{X}) = \int P(\mathbf{y}, w \mid \mathbf{X}) dw$$

- But computing this integral generally is hard!
- For some models (like a Normal joint), we know the analytical answer.
- For most, there is no easy analytical integral.
- Computing this numerically only possible for very simple cases - definitely not scalable.

Why Approximate?

LONGSHOT SYSTEMS

Example

$\int \sigma(\mathcal{N}(y \mid \mathbf{Xw}, \beta^{-1}\mathbf{I})) \cdot \mathcal{N}(\mathbf{w} \mid \mathbf{0}, \mathbf{I}) d\mathbf{w} \;\; == \; ???$

- Possible to compute with numerical integration for a given $\mathbf{x}$.
- Curse of dimensionality kicks in pretty quickly.

Outline

Conjugate Example

LONGSHOT SYSTEMS

Bayesian Linear Regression

$P(\mathbf{w}) \sim \mathcal{N}(\mathbf{w}|0, \alpha^{-1}\mathbf{I})$
$P(\mathbf{y}|\mathbf{X}, \mathbf{w}) \sim \mathcal{N}(\mathbf{y}|\mathbf{X}\mathbf{w}, \beta^{-1}\mathbf{I})$
We can make use of conjugancy, and as we know that the posterior must also be Normal, so:
$P(\mathbf{w}|\mathbf{y}, \mathbf{X}) \sim \mathcal{N}(\mathbf{w}|\mu, \Sigma)$
$\mu = \beta\Sigma\mathbf{X}^T\mathbf{y}$
$\Sigma = (\alpha\mathbf{I} + \beta\mathbf{X}^T\mathbf{X})^{-1}$
See PRML, Bishop for more detail if you aren't familiar with this result.

Notebook

Example

Jupyter Notebook 2 - Bayesian Linear Regression

Conjugate Example

Bayesian Linear Regression Cont.

- This works because a linear transformation of a normally distributed R.V. is also normally distributed.
- Normal likelihood with a Normal Mean is still Normal.
- But Normal likelihood can be wrong for many reasons:
  - Usually justified by the Central Limit Theorem and lots of hand waving - not all 'error' is the sum of lots of independent sources of error!
  - Depending on the underlying process it may be wrong to assume symmetric error. For example, if we predict 'fair' house prices, observed variables are unlikely to be symmetric about fair - people like money!
  - Likelihood mean may not be linear function of prior.

Outline

Approximate Bayesian Logistic Regression

Historically, when we could no longer rely on conjugacy, we had to fall back to analytical approximations. The Laplace approximation is argueably the most widely used example.

Example

Jupyter Notebook 3 - Laplace Approximation

Using the ELBO

Approximate posterior with a fixed model

$$KL(q(z)||p(z \mid y)) = E_q[log\ q(z) - log\ p(z \mid y)]$$

$$= E_q[log\ q(z) - log\ \tfrac{p(z,y)}{p(y)}]$$
$$= E_q[log\ q(z) - (log\ p(z,y) - log\ p(y))]$$

$$KL(q(z)||p(z \mid y)) = E_q[log\ q(z) - log\ p(z,y)] + log\ p(y))$$

$$log\ p(y) = E_q[log\ p(z,y) - log\ q(z)] + KL(q(z)||p(z \mid y))$$

$$log\ p(y) \geq \mathbf{E_q[log\ p(z,y) - log\ q(z)]}$$

Approximate Bayesian Logistic Regression

Example

Jupyter Notebook 4 - Approximate Posterior with the ELBO

Bayesian NN

LONGSHOT SYSTEMS

ELBO as a loss function

- **Regression**: $log\ p(y \mid \mathbf{X}, w) = log\ \mathcal{N}(y \mid f(\mathbf{X}, W), \sigma^2)$
- **Classification**: $log\ p(y \mid \mathbf{X}, w) =$ $y\ log\ \sigma(f(\mathbf{X}, W)) + (1-y)\ log(1-\sigma(f(\mathbf{X}, W)))$
- Where $f(\mathbf{X}, W)$ is some arbitrary function of the input data and weights. In this case it is our neural network architecture!
- $log\ p(w) = \mathcal{N}(w \mid 0, 1)$
- $log\ q(w) = \mathcal{N}(w \mid \mu, \Sigma)$
- $q(w)$ doesn't have to be normal - it can be anything we choose.

Bayesian NN

LONGSHOT SYSTEMS

Example

Jupyter Notebook 5 - Bayesian Neural Network

Outline

Learning the model too...

LONGSHOT SYSTEMS

Deep Learning to learn complex distributions

- Previously we used the ELBO to learn the parameters ϕ for an approximation, q_ϕ, to the fixed posterior, p.
- No reason why p must be fixed.

Fixed Model:

$$max_\phi \; E_q[log \; p(y \mid z)p(z) - log \; q_\phi(z)]$$

Learnt Model:

$$max_{\phi,\theta} \; E_q[log \; p_\theta(y \mid z)p(z) - log \; q_\phi(z)]$$

Simple to do in PyTorch - we can automatically maximise the ELBO over two sets of parameters instead of one.

VAE

LONGSHOT SYSTEMS

VAEs

- Make Approximate posterior conditional on input y. $q_\phi(z \mid y)$.
- P and q are Normals, parameterised by neural networks.
- Each network produces the mean and variance of a Normal distribution.
- $p(y \mid z) \sim \mathcal{N}(f(z)[0], f(z)[1])$
- $q(z \mid y) \sim \mathcal{N}(g(y)[0], g(y)[1])$
- Backprop over the weights of both f and g, with ELBO as the loss function.
- The weights of f and g are considered the parameters of p and q respectively.

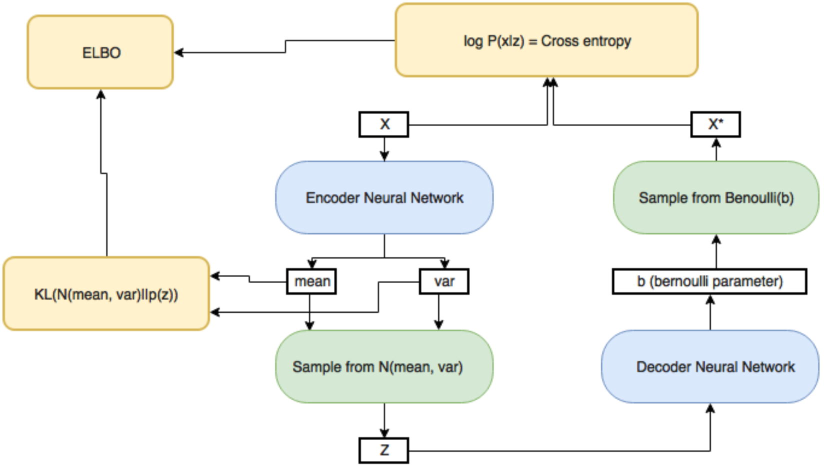
VAE

LONGSHOT SYSTEMS

Why Variational "Autoencoder"?

- Usually, the approximate posterior is $q_\phi(z)$.
- For VAEs it is $q_\phi(z \mid x)$.
- Sampling from $q(z \mid x)$ and then sampling from $p(x \mid z)$ is like an auto-encoder.
- $log\ p(x \mid z)$ is then cross entropy between original and the 'reconstruction' (sample from $p(x \mid z)$).

VAE


ELBO
log P(x|z) = Cross entropy
X
X*
Encoder Neural Network
Sample from Benoulli(b)
mean
var
KL(N(mean, var)||p(z))
b (bernoulli parameter)
Sample from N(mean, var)
Decoder Neural Network
Z

VAEs

Example

Jupyter Notebook 6 - Variational Autoencoder

Outline

Discriminators

LONGSHOT SYSTEMS

Probabilistic Interpretation of Discriminators

- Consider a dataset of real samples, and we add to this an equal number of generated 'fake' samples. We can add a label variable, y, that is 1 if the sample is real and 0 if it is fake.
- $P(y=1 \mid x) = \frac{P(x|y=1)P(y=1)}{P(x|y=0)P(y=0)+P(x|y=1)P(y=1)}$
- $P(y=1) = P(y=0) = 0.5$
- $P(y=1 \mid x) = \frac{P(x|y=1)}{P(x|y=0)+P(x|y=1)}$
- $P(y=1 \mid x) = \frac{1}{1+\frac{P(x|y=0)}{P(x|y=1)}} = \frac{1}{1+exp(ln\ \frac{P(x|y=0)}{P(x|y=1)})}$

Discriminators

LONGSHOT SYSTEMS

Probabilistic Interpretation of Discriminators

- $P(y = 1 \mid x) = \frac{1}{1+\frac{P(x|y=0)}{P(x|y=1)}} = \frac{1}{1+exp(ln \frac{P(x|y=0)}{P(x|y=1)})}$
- If we train a neural network with a sigmoid on the output, the form is: $\frac{1}{1+exp(-f(x))}$
- So we can see that training a discriminator relates to learning probability ratios.

Discriminators

Example

Jupyter Notebook 7 - Learning Distributions with Discriminators

Outline

Introduction to GANs

GANs

- Uses an adversarial approach - uses two neural networks, a generator and a discriminator.
- Generator tries to learn to generate 'fake' samples to fool the discriminator.
- Discriminator tries to predict if a sample is real or fake.
- $min_G max_D E_{P_d}[log\ D(x)] + E_{P_z}[log\ 1 - D(G(z))]$
- As shown previously, discriminators approximate ratios of distributions - in this case between the 'real' distribution and our generators distributions.

GANs

Example

Jupyter Notebook 8 - Generative Adversarial Networks

Some issues

LONGSHOT SYSTEMS

Mode Dropping

From the GAN paper, if we assume optimality of the GAN's discriminator:

$$E_{P_d}[log\ D(x)] + E_{P_g}[log\ 1 - D(G(z))]$$
$$= E_{P_d}[log\ \frac{P_d(x)}{P_d(x)+P_g(x)}] + E_{P_g}[log\ \frac{P_g(x)}{P_d(x)+P_g(x)}]$$
$$= KL(P_d||P_d(x) + P_g(x)) + KL(P_g||P_d(x) + P_g(x))$$

This is a symmetric divergence - so it shouldn't exhibit mode seeking or mode covering behaviour.

Mode Dropping

Mode Dropping

- The previous result that the loss function for the generator is symmetric makes a strong assumption - optimality of the discriminator.
- In practice, this is not even kind of close to being true!
- Normally we train GANs by iterating between a step for the generator and a step for the discriminator. However, the above result expects we should train the discriminator to optimality, then take a single step with the generator.
- It's possible to derive a result that doesn't rely on optimality.
- We will see this in the final section.

GANs

LONGSHOT SYSTEMS

Example

Jupyter Notebook 9 - Mode Dropping in Generative Adversarial Networks

Outline

Unifying Theory

Joining GANs and VAEs

- Not as hard to do as you might think.
- Does take a bit of work though, and notation change can be confusing!
- This last section will be a brief, high level summary of recent work.
- If you want more detail, references at the end!

GANs

LONGSHOT SYSTEMS

GANs

- Introduce auxilarly 'fake' variable, y. y=0 for real, y=1 for generated data.
- Discriminator is now given by $q(y|\mathbf{x})$. So $q(y = 0|\mathbf{x}) = D(x)$
- Our full set of training data is a mixture of real and fake examples.
- $P_g(x)$ is our generator's (implicit) distribution.
- $p(x \mid y = 0) = P_{data}(x)$, $p(x \mid y = 1) = P_g(x)$.

GANs

LONGSHOT SYSTEMS

$$p(x \mid y = 0) = P_{data}(x)$$
$$p(x \mid y = 1) = P_g(x).$$

$$E_{P_d}[log\ D(x)] + E_{P_g}[log\ 1 - D(x)]$$
$$= E_{P_d}[log\ q(y = 0 \mid x)] + E_{P_g}[log\ q(y = 1 \mid x)]$$
$$= E_{P_\theta(x|y)P(y)}[log\ q_\phi(y \mid x)]$$

Won't derive here, but the gradient of this with respect to θ:

$$\bigtriangledown_\theta E_{P_\theta(x|y)P(y)}[log\ q_\phi(y \mid x)]$$
$$\propto \bigtriangledown_\theta[KL(P_{g_\theta} \mid\mid q_\phi(x \mid y = 1)) - JSD(P_d \mid\mid P_{g_\theta})]$$

GANs and VAEs

Comparing two deep generative models

- **GANs Optimise:** $KL(P \, || \, Q)$
- This causes mode seeking behaviour in the Generator - hence why we see mode dropping in GANs.
- **VAEs Optimise:** $KL(Q \, || \, P)$
- This causes mass covering behaviour in the 'true' joint distribution - one of the reasons we tend to see VAEs producing lower quality samples than GANs.

Outline

LONGSHOT SYSTEMS

Summary

- We have seen how starting at conjugate models, we can make gradual relaxations to end up with VI, and we can make further generalizations to end up with VAEs, GANs and everything inbetween.
- VAEs and GANs not really different camps - both theoretically quite similar.
- If we want to understand these complex methods, simpler examples and simpler versions can help.

The End

Thanks for listening. Questions?

Outline

LONGSHOT SYSTEMS

References and further reading

Bayesian Linear and Logistic Regression

- Pattern Recognition and Machine Learning, Chris Bishop
- Bayesian Reasoning and Machine Learning, David Barber

ELBO

- Harvard AM 207 - A good introduction to the ELBO.
- Auto-encoding Variational Bayes
- M. Shakir's blog post on reparameterisation methods

VAEs

- Auto-encoding Variational Bayes
- Durk Kingma's Thesis (direct link)

References and Further Reading

Discriminators

- Ferenc Huszár's Series of posts on implicit models
- M. Shakir's blog post on Density Ratio trick
- Adversarial Variational Bayes

GANs

- Original Paper
- Unrolled GANs (mode dropping example)

References and Further Reading

Unifying Theory and other

- On Unifying Deep Generative Models
- Log derivative trick
- Pyro Documentation on Stochastic Variational Inference
- My Blog!