Bayesian Deep Learning
Going full Bayesian

- Bayes = Probabilities
- Bayes Theorem

\[ p(H|E) = \frac{p(E|H)p(H)}{p(E)} \]

Hypothesis = Model
Evidence = data
Going full Bayesian

• Start with a prior on the model parameters $p(\theta)$

• Choose a statistical model $p(x|\theta)$

• Use data to refine my prior, i.e., compute the posterior

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)}$$

No dependence on parameters
Going full Bayesian

- Start with a prior on the model parameters $p(\theta)$
- Choose a statistical model $p(x|\theta)$
- Use data to refine my prior, i.e., compute the posterior

$$p(\theta|x) = p(x|\theta)p(\theta)$$
• 1. Learning: Computing the posterior
  – Finding a point estimate (MAP) \( \Rightarrow \) what we have been doing so far!
    \[ p(\theta|x) = p(x|\theta)p(\theta) \]
  – Finding a probability distribution of \( \theta \) This lecture
    \[ p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)} \]
What have we learned so far?

• **Advantages** of Deep Learning models
  – Very expressive models
  – Good for tasks such as classification, regression, sequence prediction
  – Modular structure, efficient training, many tools
  – Scales well with large amounts of data

• But we have also **disadvantages**…
  – "Black-box" feeling
  – We cannot judge how "confident" the model is about a decision
Modeling uncertainty

• Wish list:
  – We want to know what our models know and what they do not know
Modeling uncertainty

• Example: I have built a dog breed classifier

What answer will my NN give?

Bulldog
German shepherd
Chihuaha
Modeling uncertainty

- Example: I have built a dog breed classifier

Bulldog

German shepherd

Chihuahua

I would rather get as an answer that my model is not certain about the type of dog breed.
Modeling uncertainty

• Wish list:
  – We want to know what our models know and what they do not know

• Why do we care?
  – Decision making
  – Learning from limited, noisy, and missing data
  – Insights on why a model failed
Modeling uncertainty

- Finding the posterior
  - Finding a point estimate (MAP) → what we have been doing so far!
  - Finding a probability distribution of $\theta$

Image: https://medium.com/@joeDiHare/deep-bayesian-neural-networks-952763a9537
Modeling uncertainty

• We can sample many times from the distribution and see how this affects our model’s predictions
• If predictions are consistent = model is confident

Image: https://medium.com/@joeDiHare/deep-bayesian-neural-networks-952763a9537
Modeling uncertainty

How do we get the posterior?

• Compute the posterior over the weights

\[ p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)} \]

• Probability of observing our data under all possible model parameters

\[ p(\theta|x) = \frac{p(x|\theta)p(\theta)}{\int_{\theta} p(x|\theta)p(\theta) d\theta} \]
How do we get the posterior?

• How do we compute this?

\[ p(\theta|x) = \frac{p(x|\theta)p(\theta)}{\int_\theta p(x|\theta)p(\theta)d\theta} \]

• Denominator = we cannot compute all possible combinations

• Two ways to compute the approximation of the posterior:
  - Markov Chain Monte Carlo
  - Variational Inference
How do we get the posterior?

• Markov Chain Monte Carlo (MCMC)
  – A chain of samples
    \[ \theta_t \rightarrow \theta_{t+1} \rightarrow \theta_{t+2} \ldots \]
    that converge to \( p(\theta|x) \)

• Variational Inference
  – Find an approximation \( q(\theta) \) that.
    \[ \arg \min \ KL(q(\theta)||p(\theta|x)) \]
Dropout for Bayesian Inference
Recall: Dropout

• Disable a random set of neurons (typically 50%)
Recall: Dropout

- Using half the network = half capacity

Furry
Has two eyes
Has a tail
Has paws
Has two ears

Redundant representations
Recall: Dropout

- Using half the network = half capacity
  - Redundant representations
  - Base your scores on more features

- Consider it as model ensemble
Recall: Dropout

- Two models in one

(b) After applying dropout.
MC dropout

- Variational Inference
  - Find an approximation $q(\theta)$ that
    $$\arg\min KL(q(\theta)||p(\theta|x))$$

- Dropout training
  - The variational distribution is from a Bernoulli distribution (where the states are “on” and “off”)

MC dropout

• 1. Train a model with dropout before every weight layer
• 2. Apply dropout at test time
  – Sampling is done in a Monte Carlo fashion, hence the name Monte Carlo dropout

MC dropout

- Sampling is done in a Monte Carlo fashion, e.g.,

\[ p(y = c|x) \approx \frac{1}{T} \sum_{t=1}^{T} \text{Softmax}(f_{\hat{\theta}_t}(x)) \]

where \( \hat{\theta}_t \sim q(\theta) \)

and \( q(\theta) \) is the dropout distribution


Prof. Leal-Taixé and Prof. Niessner
Measure your model’s uncertainty

Another look
Let us take another look

- We know it is intractable, we approximate it

\[ p(\theta|x) = \frac{p(x|\theta)p(\theta)}{\int_{\theta} p(x|\theta)p(\theta)d\theta} \]

- The denominator expresses how my data is generated

\[ p(x) = \int_{\theta} p(x|\theta)p(\theta)d\theta \]
Let us take another look

- We assume that the data is generated by some random process, involving an unobserved continuous random (latent) variable $z$

- Generation process: $p_\theta(x) = \int_z p_\theta(x|z)p_\theta(z)dz$

- Posterior: $p_\theta(z|x) = \frac{p_\theta(x|z)p_\theta(z)}{\int_z p_\theta(x|z)p_\theta(z)dz}$
Let us take another look

• Variational Inference
  – Find an approximation. $q_\phi(z|x)$

• My approximation is parameterized by a model $\phi$
Variational Autoencoders
Recall: Autoencoders

- Encode the input into a representation (bottleneck) and reconstruct it with the decoder
Variational Autoencoder

$q_\phi(z|x)$
Encoder

$p_\theta(\tilde{x}|z)$
Decoder

Conv

Transpose Conv

$x$

$\phi$

$z$

$\tilde{x}$
Variational Autoencoder

- Latent space is now a distribution
- Specifically it is a Gaussian

\[ q_\phi(z|x) \]

Encoder
Variational Autoencoder

- Latent space is now a distribution
- Specifically it is a Gaussian

\[ q_\phi(z|x) \]

Encoder

\[ \mu_{z|x} \quad \text{Mean} \]

\[ \Sigma_{z|x} \quad \text{Diagonal covariance} \]

\[ z|x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x}) \]
Variational Autoencoder

- Latent space is now a distribution
- Specifically it is a Gaussian

\[ q_\phi(z|x) \]

Encoder

\[ z|x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x}) \]

Mean

Diagonal covariance

Prof. Leal-Taixé and Prof. Niessner
Variational Autoencoder

• Back to our Bayesian view, our generation process was:

\[ p_\theta(x) = \int_z p_\theta(x|z)p_\theta(z)dz \]

• Which is the denominator of the posterior:

\[ p_\theta(z|x) = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(x)} \]

I want to optimize
Variational Autoencoder

• Loss function for a data point $x_i$

$$\log(p_\theta(x_i)) = E_{z \sim q_\phi(z|x_i)}[\log(p_\theta(x_i))]$$

I draw samples of the latent variable $z$ from my encoder
Variational Autoencoder

- Loss function for a data point $x_i$

\[
\log(p_\theta(x_i)) = \mathbb{E}_{z \sim q_\phi(z|x_i)}[\log(p_\theta(x_i))] \\
= \mathbb{E}_{z \sim q_\phi(z|x_i)} \left[ \log \frac{p_\theta(x_i|z)p_\theta(z)}{p_\theta(z|x_i)} \right]
\]

Bayes Rule

Posterior
Variational Autoencoder

- Loss function for a data point $x_i$

$$
\log(p_\theta(x_i)) = E_{z \sim q_\phi(z|x_i)}[\log(p_\theta(x_i))]
$$

$$
= E_{z \sim q_\phi(z|x_i)} \left[ \log \frac{p_\theta(x_i|z)p_\theta(z)}{p_\theta(z|x_i)} \right]
$$

$$
= E_z \left[ \log \frac{p_\theta(x_i|z)p_\theta(z)}{p_\theta(z|x_i)} \frac{q_\phi(z|x_i)}{q_\phi(z|x_i)} \right]
$$

Just a constant
Variational Autoencoder

- Loss function for a data point $x_i$

$$\log(p_\theta(x_i)) = E_z \left[ \log \frac{p_\theta(x_i \mid z)p_\theta(z)}{q_\phi(z \mid x_i)} \right]$$

$$= E_z \left[ \log p_\theta(x_i \mid z) \right] - E_z \left[ \log \frac{q_\phi(z \mid x_i)}{p_\theta(z)} \right] + E_z \left[ \log \frac{q_\phi(z \mid x_i)}{p_\theta(z \mid x_i)} \right]$$
Variational Autoencoder

• Loss function for a data point $x_i$

$$
= E_z [\log p_\theta(x_i | z)] - E_z \left[ \log \frac{q_\phi(z | x_i)}{p_\theta(z)} \right] + E_z \left[ \log \frac{q_\phi(z | x_i)}{p_\theta(z | x_i)} \right]
$$

$$
= E_z [\log p_\theta(x_i | z)] - KL(q_\phi(z | x_i) || p_\theta(z)) + KL(q_\phi(z | x_i) || p_\theta(z | x_i))
$$

Kullback-Leibler Divergences
Variational Autoencoder

• Loss function for a data point \( x_i \)

\[
E_z \left[ \log p_\theta(x_i | z) \right] - KL( q_\phi(z | x_i) \| p_\theta(z) ) + KL( q_\phi(z | x_i) \| p_\theta(z | x_i) )
\]

- **Reconstruction loss**
- Measures how good my latent distribution is with respect to my prior
- I still cannot express the shape of the distribution. But I know \( \geq 0 \)
Variational Autoencoder

- Loss function for a data point $x_i$

\[
E_z [\log p_\theta(x_i | z)] - KL(q_\phi(z | x_i) || p_\theta(z)) + KL(q_\phi(z | x_i) || p_\theta(z | x_i)) \geq 0
\]

Loss function (lower bound)

\[
\mathcal{L}(x_i, \phi, \theta) \geq \log(p(x_i))
\]
Variational Autoencoder

- Loss function for a data point $x_i$

\[ E_z[\log p_\theta(x_i|z)] - KL(q_\phi(z|x_i)||p_\theta(z)) + KL(q_\phi(z|x_i)||p_\theta(z|x_i)) \geq 0 \]

Loss function (lower bound)

\[ \mathcal{L}(x_i, \phi, \theta) \]

- Optimize $\phi^*, \theta^* = \arg \max \sum_{i=1}^{N} \mathcal{L}(x_i, \phi, \theta)$
Variational Autoencoder

- Training

\[ E_z [\log p_\theta(x_i|z)] - KL(q_\phi(z|x_i)||p_\theta(z)) + KL(q_\phi(z|x_i)||p_\theta(z|x_i)) \]

Make posterior distribution close to prior (close to unit Gaussian distribution)
Variational Autoencoder

- Training

\[ E_z [\log p_\theta(x_i | z)] - KL(q_\phi(z | x_i) || p_\theta(z)) + KL(q_\phi(z | x_i) || p_\theta(z | x_i)) \]
Variational Autoencoder

- Training

\[ E_z [\log p_\theta(x_i | z)] - KL(q_\phi(z | x_i) \| p_\theta(z)) + KL(q_\phi(z | x_i) \| p_\theta(z | x_i)) \]

Encoder

\[ \mu_{z|x}, \Sigma_{z|x} \]

Sample

\[ z|x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x}) \]
Variational Autoencoder

• Training

\[ E_z [\log p_\theta(x_i | z)] - KL(q_\phi(z | x_i) \| p_\theta(z)) + KL(q_\phi(z | x_i) \| p_\theta(z | x_i)) \]

Encoder

Decoder

\[ x \rightarrow \phi \rightarrow \mu_{z|x}, \Sigma_{z|x} \rightarrow \mathcal{N}(\mu_{z|x}, \Sigma_{z|x}) \rightarrow z \rightarrow \theta \rightarrow \tilde{x} \]
Variational Autoencoder

- Training

\[
E_z \left[ \log p_\theta(x_i | z) \right] - KL(q_\phi(z | x_i) || p_\theta(z)) + KL(q_\phi(z | x_i) || p_\theta(z | x_i))
\]

Decoder

\[x | z \sim \mathcal{N}(\mu_{x|z}, \Sigma_{x|z}) \]

\[\tilde{x}\]

Output is also parameterized
Variational Autoencoder

- Training

\[ E_z [\log p_\theta(x_i | z)] - KL(q_\phi(z | x_i) || p_\theta(z)) + KL(q_\phi(z | x_i) || p_\theta(z | x_i)) \]

Maximize the likelihood of reconstructing the input
Variational Autoencoder

- For more details and mathematical derivation
- Reparameterization trick that allows us to backprop
How about generating data?

- Training as seen before

Prof. Leal-Taixé and Prof. Niessner

http://kvfrans.com/variational-autoencoders-explained/
How about generating data?

- After training, generate random samples

Sample from the distribution (e.g., unit Gaussian)
Generating data

Each element of $z$ encodes a different feature
Generating data

Degree of smile

Head pose

Prof. Leal-Taixé and Prof. Niessner
Autoencoder vs VAE

Autoencoder

Variational Autoencoder

Ground Truth

https://github.com/kvfrans/variational-autoencoder
Autoencoder Overview

• Autoencoders (AE)
  – Reconstruct input
  – Unsupervised learning
  – Latent space features are useful

• Variational Autoencoders (VAE)
  – Probability distribution in latent space (e.g., Gaussian)
  – Sample from model to generate output
Autoencoder Overview

• Autoencoders (AE)
  – Reconstruct input
  – Unsupervised learning
  – Latent space features are useful

• Variational Autoencoders (VAE)
  – Probability distribution in latent space (e.g., Gaussian)
  – Interpretable latent space (head pose, smile)
  – Sample from model to generate output
Generative models
Taxonomy of generative models

Generative models

- Explicit density
  - Tractable density
    - Fully Visible Belief Nets
      - NADE
      - MADE
      - PixelRNN/CNN
    - Change of variables models (nonlinear ICA)
  - Approximate density
    - Variational
      - Variational Autoencoder
  - Implicit density
    - Markov Chain
      - GSN
    - Markov Chain
      - Boltzmann Machine

Figure from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017
Taxonomy of generative models

\[ p_\theta(x) = \int_z p_\theta(x|z)p_\theta(z)dz \]

Generative models

Explicit density

Tractable density
- Fully Visible Belief Nets
  - NADE
  - MADE
  - PixelRNN/CNN
- Change of variables models (nonlinear ICA)

Implicit density

Approximate density

Variational

Variational Autoencoder

Markov Chain

GSN

Direct GAN

Figure from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017
Define a more tractable density function

Taxonomy of generative models

- Explicit density
  - Fully Visible Belief Nets
    - NADE
    - MADE
    - PixelRNN/CNN
  - Change of variables models (nonlinear ICA)

- Implicit density
  - Approximate density
  - Variational Autoencoder
  - Boltzmann Machine

- Markov Chain
  - GSN

- Direct GAN

Figure from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017
I do not care about the shape, I just want to sample!
Next lectures

• Next Monday 10th, more on Generative models

• 3\textsuperscript{rd} round of presentations this Friday $\rightarrow$ you will receive feedback about the presentations

• Keep working on the projects!
Other references

• Conditional Variational Autoencoders:
  – Xinchen Yan, Jimei Yang, Kihyuk Sohn, Honglak Lee, Attribute2Image: Conditional Image Generation from Visual Attributes, ECCV, 2016 –
Other references

• Interesting read:
  – Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, Ole Winther, Autoencoding beyond pixels using a learned similarity metric, ICML, 2016
  – Aditya Deshpande, Jiajun Lu, Mao-Chuang Yeh, David Forsyth, Learning Diverse Image Colorization, arXiv, 2016