MACHINE LEARNING & DATA MINING

Recap

Entropy:

$$H(p) = -\mathop{\mathbb{E}}_{p} \log p$$

Cross entropy:

$$H(p, q) = -\mathop{\mathbb{E}}_{p} \log q$$

Kullback–Leibler divergence:

$$D_{KL}(p \,||\, q) = H(p, q) - H(p) = \mathop{\mathbb{E}}_{p} \left[\log \frac{p}{q} \right]$$

Model with hidden parameters

- Model:

$$p(v, h)$$

- Can be defined e.g. in terms on prior on the hidden variables + conditional for the observed ones:

$$p(v \mid h) \cdot p(h)$$

  - I.e. hidden state is 'the cause' of the observed state

- Learning with the maximum likelihood method:

$$\log p(v) \to \max$$

Learning with max likelihood

- Need to marginalize out the hidden variables:

$$\log p(v) = \log \frac{p(v)p(h\,|\,v)}{p(h\,|\,v)} = \log \frac{p(h,v)}{p(h\,|\,v)}$$

$$= \mathop{\mathbb{E}}_{h\sim p(h\,|\,v)} \log \frac{p(h,v)}{p(h\,|\,v)}$$

$$= \mathop{\mathbb{E}}_{h\sim p(h\,|\,v)} \log p(h,v) + H(p(h\,|\,v))$$

- This requires the knowledge of the posterior $p(h\,|\,v)$ – typically intractable

Approximate inference

- We can approximate the posterior $p(h \mid v)$ with some $q(h)$

$$L = \mathop{\mathbb{E}}_{h \sim q(h)} \log p(h, v) + H(q(h))$$

- Let's compare it with the original log-likelihood:

$$\begin{aligned} \log p(v) - L &= \log p(v) - \mathop{\mathbb{E}}_{h \sim q(h)} \log p(h, v) - H(q(h)) \\ &= \mathop{\mathbb{E}}_{h \sim q(h)} \left[\log p(v) - \log p(h, v) + \log q(h)\right] \\ &= \mathop{\mathbb{E}}_{h \sim q(h)} \left[- \log p(h \mid v) + \log q(h)\right] = D_{KL}(q(h) \,||\, p(h \mid v)) \end{aligned}$$

Approximate inference

- We've shown that:

$$L = \mathop{\mathbb{E}}_{h \sim q(h)} \log p(h, v) + H(q(h))$$

$$\log p(v) - L = D_{KL}(q(h) \,||\, p(h \,|\, v)) \geq 0$$

- This means that L is the lower bound for the true log-likelihood
  - Also called evidence lower bound (ELBO) or variational lower bound
- The better q approximates the posterior — the closer the bound is to the actual log-likelihood
- Instead of maximizing the likelihood we can maximize ELBO!

ELBO

- Alternative form:

$$\begin{aligned} L &= \mathop{\mathbb{E}}_{h \sim q(h)} \log p(h, v) + H(q(h)) \\ &= \mathop{\mathbb{E}}_{h \sim q(h)} [\log p(v \mid h) + \log p(h) - \log q(h)] \\ &= \underbrace{\mathop{\mathbb{E}}_{h \sim q(h)} \log p(v \mid h)}_{\text{Data term}} - \underbrace{D_{KL}(q(h) \,||\, p(h))}_{\text{Regularizer}} \end{aligned}$$

Approximate inference

- There might be different choices for q depending on the problem
- Some models can give the q distribution analytically. E.g. expectation maximization (EM) algorithm optimizes ELBO by repeating the following steps:
  - E-step: set q to equal the posterior precisely (as defined by the current non-optimal model parameters)
  - M-step: completely or partially maximize ELBO with respect to the model parameters (with fixed q)
- Alternatively we can specify the form of q such that it is easy to sample from and calculate KL divergence with the prior

Bayesian NN

- Key idea: treat the weights as hidden parameters
- Define q in some simple form (e.g. independent normal distributions for each of the weights)
  - simple to sample from
  - calculate KL-divergence analytically
- Define prior on the weights
  - different priors cause different properties of the network
  - e.g. log-uniform prior favors removing noisy weights [1]
- Optimize the ELBO

[1] D. Molchanov, et. al. Variational Dropout Sparsifies Deep Neural Networks, Published in ICML 2017, https://arxiv.org/abs/1701.05369

Bayesian NN

- Incorporates regularization naturally (using prior)
- Results in an ensemble of networks
  - Robust to errors due to data out of the training set domain
- Can estimate uncertainties
- On-line learning possible (using the learnt weights distribution as the new prior)

Back-propagating through random operations

$$L = \mathop{\mathbb{E}}_{h \sim q(h)} \log p(v \mid h) - D_{KL}(q(h) \,||\, p(h))$$

Optimizing ELBO requires calculating the gradients w.r.t. parameters of q from which we sample

Q: How can we achieve this?

Back-propagating through random operations

$$L = \mathop{\mathbb{E}}_{h \sim q(h)} \log p(v \,|\, h) - D_{KL}(q(h) \,||\, p(h))$$

Optimizing ELBO requires calculating the gradients
w.r.t. parameters of q from which we sample

Q: How can we achieve this?

A: Reparametrization trick — define the random sampling as sampling from fixed distribution + differentiable transformation

Reparametrization trick (example)

- Say, we want to sample $y \sim N(\mu, \sigma^2)$ and then optimize some $f(y)$ wrt to μ and σ
- We can sample $z \sim N(0, 1)$
- Then transform it with $y = \sigma z + \mu$

Variational Autoencoders (VAE)

- Generative model, learning data distribution $p(x)$ using a conditional on the hidden variables $p(x \mid h)$ and a prior $p(h)$
- Posterior approximation q with a neural network – 'encoder'
  - More precisely: q is the product of independent normal distributions
  - Encoder neural net inputs a data object
  - and outputs means and variances for these normal distributions
- Conditional $p(x \mid h)$ defined as another network – 'decoder'
- Prior as the product of independent normal distributions with 0 mean and unit variance
- Trained by maximizing the ELBO wrt parameters of encoder and decoder (simultaneously)

Variational Autoencoders (VAE)

- ELBO:

$$L = \mathop{\mathbb{E}}_{h \sim q(h)} \log p(v \,|\, h) - D_{KL}(q(h) \,||\, p(h))$$

- Typically decoder predicts means of the data vector
- assuming all variances are same (hyper-parameter)
- This reduces to an MSE loss term
- Variance hyper-parameter controls tradeoff between precision and diversity

- KL-divergence between two normal distributions can be calculated analytically

Limitations

- When applied to image generation, MSE loss typically results in blurry images

(a) (b) (c)

Image (b) — slightly altered image (a), image (c) — image (a) shifted by several pixels.
Under MSE metric, image (b) is much closer to (a), than (c) to (a).

- MSE loss doesn't reflect our perception of good vs bad image quality