Variational Autoencoders - Notes

Anna-Lena Popkes

April 18, 2017

1 Variational Inference

- Variational Bayesian methods are a family of techniques for approximating intractable integrals arising in Bayesian inference and machine learning
- They are typically used in complex statistical models consisting of observed variables (data) as well as unknown parameters and latent variables
- Variational Bayesian methods are primarily used for two purposes:
    1. To provide an approximation to the posterior probability of the unobserved variables, in order to do statistical inference over these variables
    2. To derive a lower bound for the marginal likelihood (evidence) of the observed data (i.e. the marginal probability of the data given the model, with marginalization performed over unobserved variables).
- We want to estimate the posterior distribution $p(z|x)$. However, for many models the exact posterior cannot be computed
- In variational inference, the posterior distribution over a set of unobserved variables $z_1, ..., z_n$ given some data x is approximated by a *variational distribution* $q(z)$: $p(z|x) \approx q(z)$
- The distribution $q(z)$ is restricted to belong to a family of distributions of simpler form than $p(z|x)$
- $q(z)$ should be as similar to the true posterior $p(z|x)$ as possible
- This is done by finding a setting of the parameters that makes q as close to the true posterior as possible
- The dissimilarity between the two distributions is typically measured using the Kullback-Leipler divergence $D_{KL}(q(z)||p(z|x))$
- The KL divergence can't be minimized exactly. But we can minimize a function that is equal to it up to a constant. This is the *evidence lower bound* (ELBO)

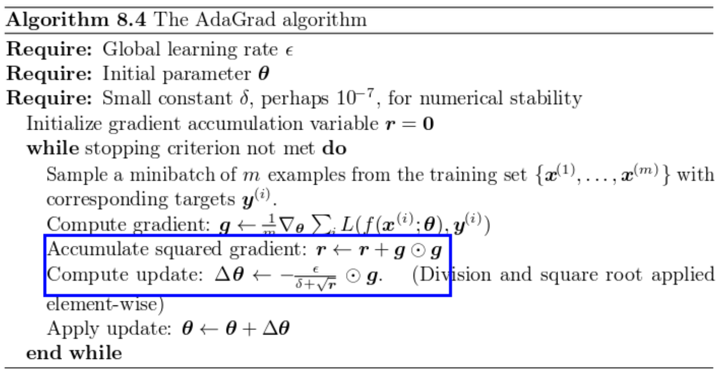
- We reformulate the KL divergence as follows:
  $D_{KL}(q(z)||p(z|x)) = \mathbb{E}_q\Big[\log\frac{q(z)}{p(z|x)}\Big]$
  $= -\mathbb{E}_q\big[\log p(z,x)\big] + \mathbb{E}_q\big[\log q(z)\big] + \log p(x)$
- This is the negative ELBO plus the log marginal probability of x
- Maximizing the ELBO is equivalent to minimizing the KL divergence

2 Optimization algorithms

This section introduces the major optimization algorithms that use adaptive learning rates for the parameters in a model. All figures come from source 2.

2.1 AdaGrad

- Individually adapts the learning rates of all model parameters by scaling them inversely proportional to the square root of the sum of all their historical squared values (see figure 1)
- Therefore, parameters with large partial derivatives have a higher decrease in their learning rate
- Parameters with small partial derivatives have a smaller decrease in their learning rate



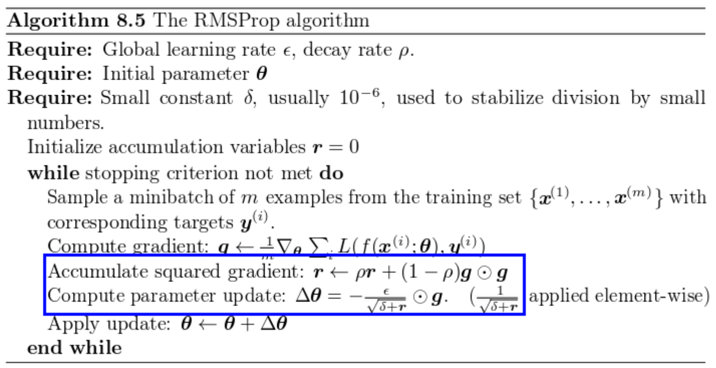
```
Algorithm 8.4 The AdaGrad algorithm
Require: Global learning rate $\epsilon$
Require: Initial parameter $\boldsymbol{\theta}$
Require: Small constant $\delta$, perhaps $10^{-7}$, for numerical stability
  Initialize gradient accumulation variable $\boldsymbol{r} = \boldsymbol{0}$
  while stopping criterion not met do
    Sample a minibatch of $m$ examples from the training set $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ with
    corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    Accumulate squared gradient: $\boldsymbol{r} \leftarrow \boldsymbol{r} + \boldsymbol{g} \odot \boldsymbol{g}$
    Compute update: $\Delta\boldsymbol{\theta} \leftarrow -\frac{\epsilon}{\delta+\sqrt{\boldsymbol{r}}} \odot \boldsymbol{g}$.   (Division and square root applied
    element-wise)
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
  end while
```

Figure 1: The AdaGrad algorithm

For the training of deep networks it has been shown that accumulating squared gradients from the beginning of training can decrease learning rates too much and too quickly.

2.2 RMSProp

Modifies AdaGrad and changes the gradient accumulation into an exponentially weighted moving average. This enables RMSProp to discard history from the extreme past. It can be used with or without Nesterov momentum. Figure 2 displays the algorithm without Nesterov momentum.



```
Algorithm 8.5 The RMSProp algorithm
Require: Global learning rate $\epsilon$, decay rate $\rho$.
Require: Initial parameter $\boldsymbol{\theta}$
Require: Small constant $\delta$, usually $10^{-6}$, used to stabilize division by small
  numbers.
  Initialize accumulation variables $\boldsymbol{r} = 0$
  while stopping criterion not met do
    Sample a minibatch of $m$ examples from the training set $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ with
    corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    Accumulate squared gradient: $\boldsymbol{r} \leftarrow \rho \boldsymbol{r} + (1-\rho) \boldsymbol{g} \odot \boldsymbol{g}$
    Compute parameter update: $\Delta\boldsymbol{\theta} = -\frac{\epsilon}{\sqrt{\delta + \boldsymbol{r}}} \odot \boldsymbol{g}$.   ($\frac{1}{\sqrt{\delta + \boldsymbol{r}}}$ applied element-wise)
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
  end while
```

Figure 2: The RMS algorithm

RMSProp has been shown to be both effective and practical for training deep networks.

2.3 Adam

2.4 Sources

1. `http://sebastianruder.com/optimizing-gradient-descent/index.html#adagrad`
2. `http://www.deeplearningbook.org/contents/optimization.html#pf23`

3 Variational Autoencoders

3.1 Basic Problem Setup

- We are given a dataset of observations $x_1, ..., x_n$
- And a set of latent variables $z_1, ..., z_m$ with $z_i \sim p(z)$
- In variational autoencoders (VAEs) $p(z)$ is specified as a standard Normal distribution with mean zero and variance one, so $p(z) = \mathcal{N}(0, 1)$
- The goal is to

3.2 Structure

- A VAE consists of two parts
  1. A probabilistic encoder $q_\phi(z|x)$ approximating the true but intractable posterior distribution $p(z|x)$
  2. A generative decoder $p_\theta(x|z)$ which does not rely on any particular input x
- The encoder is known as the *inference* or *recognition network*
- The decoder is known as the *generative network*
- Both the encoder and decoder are implemented as neural networks with tunable parameters θ and ϕ
- For example, for $q_\phi(z|x)$ the neural network with parameters θ receives the input x and produces μ, Σ as an output, e.g. $q_\theta(z|x) = \mathcal{N}(z|\mu(x,\theta), \Sigma(x,\theta))$. From this distribution we can draw samples of z, i.e. $z \sim q_\phi(z|x)$.
- $p_\theta(x|z)$ is implemented in the same fashion
- Different to the classical autoencoder, we do not have a deterministic z but we draw a random sample of z from $q_\phi(z|x)$, i.e. z is stochastic

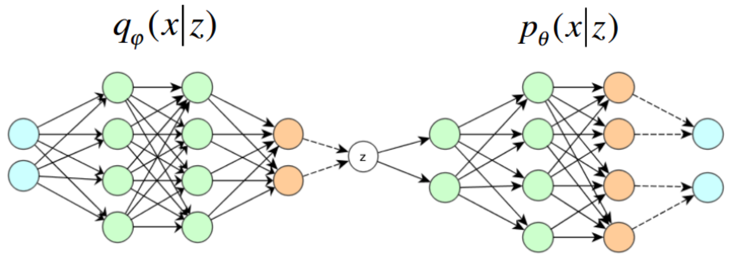


Figure 3: Encoding and decoding part of a VAE

3.3 Training

- Training a VAE or, more generally, a generative model, is the same as learning the joint distribution $p(x,z)$ over the data and latent variables
- We can write the joint probability distribution as $p(x,z) = p(x|z)p(z)$
- In inference, the goal is to infer good values of the latent variables given the observed data. So we want to calculate $p(z|x) = \frac{p(x,z)}{p(x)} = \frac{p(x|z)p(z)}{p(x)}$
- $p(x)$ is called the evidence and can be calculated by marginalizing out the latent variables: $\int p(x|z)p(z)dz$

- We cannot compute this integral. Therefore, we approximate $p(z|x)$ with a family of distributions $q_\phi(z|x)$
- The variational parameter ϕ indexes the family of distributions. If q were Gaussian, ϕ would be the mean and variace of the latent variables for each datapoint
- Our goal is to find the variational parameters ϕ that minimize the KL divergence between $q_\phi(z|x)$ and $p(z|x)$
- So we want to compute arg $\min_q$ $\mathrm{KL}(q_\phi(z|x)||p(z|x))$
- As mentioned before, we cannot minimize KL directly. Therefore, we instead maximize the ELBO
- The ELBO for a single datapoint x_i is given by: $ELBO(\phi,\theta) = \mathbb{E}_{q_\phi(z|x_i)}\big[\log p(x_i|z)\big] - D_{KL}(q_\phi(z|x_i)||p(z))$
- The ELBO rewards a good reconstruction (first term). The second term represents a regularizer that prevents col1=111xubuntuxubuntu xubuse ubuntu onuse ubutnlapsing to the deterministic autoencoder
- The parameters ϕ, θ are learned by maximizing this quantity using gradient descent and backpropagation

3.4 Reparametrization Trick

3.5 Applications

- Analysis of brain MRI images:
  `http://www.cs.unc.edu/~eunbyung/papers/manifold_variational.pdf`
- Adversarial autoencoders:
  `https://arxiv.org/pdf/1511.05644.pdf`
- Sentence generation using an RNN-based VAE:
  `https://arxiv.org/pdf/1511.06349.pdf?TB_iframe=true&width=921.6&height=921.6`
- Predict the trajectory of pixels in scenes:
  `https://arxiv.org/pdf/1606.07873.pdf`
- Anomaly detection using a VAE+ESN:
  `http://ieeexplore.ieee.org/document/7727309/?reload=true`
- Anomaly detection:
  `http://dm.snu.ac.kr/static/docs/TR/SNUDM-TR-2015-03.pdf`
- Adversarial autoencoder:
  `https://blog.paperspace.com/adversarial-autoencoders-with-pytorch/`

3.6 Sources

- https://www.cs.princeton.edu/courses/archive/fall11/cos597C/lectures/variational-inference-i.pdf)
- https://home.zhaw.ch/~dueo/bbs/files/vae.pdf
- https://arxiv.org/pdf/1601.00670.pdf
- https://www.youtube.com/watch?v=lG1AIqIEv9I&index=3&list=PLR6O_WZHBlOE6xgFU725tenJszVDDHZyH
- https://jaan.io/what-is-variational-autoencoder-vae-tutorial/
- http://stats.stackexchange.com/questions/199605/how-does-the-reparameterization-trick-
- http://dpkingma.com/wordpress/wp-content/uploads/2015/12/talk_nips_workshop_2015.pdf
- https://github.com/oduerr/dl_tutorial/blob/master/tensorflow/vae/vae_demo.ipynb
- https://jmetzen.github.io/2015-11-27/vae.html
- http://int8.io/variational-autoencoder-in-tensorflow/