COS 597G: Toward Theoretical Understanding of Deep Learning **Fall 2018**

Lecture 10: 18 October 2018

Lecturer: Sanjeev Arora *Scribe: Nataly Brukhim, Arushi Gupta, Edgar Minasyan*

10.1 Unsupervised Learning

There are two broad categories of learning we will be talking about in these notes, namely supervised learning and unsupervised learning. Supervised learning is learning with labels and unsupervised learning is learning without labels. What could be the possible goals of unsupervised learning?

Well, a possible goal could be to learn some hidden structure of the data. An example would be Principal Component Analysis (PCA) concerned with finding the most important directions in the data. In the context of the previous lecture, A_i is like a covariance matrix of the data and one is picking out the largest singular vectors, which is similar to picking the most important directions. Many other examples of structure learning can include probabilistic models as well such as topic models.

Another basic goal of unsupervised learning could be to learn the distribution of the data, otherwise called density estimation. A classic example of density estimation is the crab example: one collects different features of a population of crabs on an island such as the weight and length presuming that the distributions of these features should follow the normal distribution – eventually finding out about a mixture of Gaussians in one of the features which brings about essential evolutionary implications about the crab population on that island. In general, in density estimation the hypothesis is that the distribution of data is $p(h, x)$ where x is the data and h are some hidden variables, often called latent variables.

A third goal of unsupervised learning is learning good features, a meaningful representation, that are useful in downstream tasks. A good representation of the data, i.e. the learned features, would enable more efficient learning of the downstream tasks. This approach allows one to learn from fewer examples in a semi-supervised fashion.

10.2 Density estimation

Given some i.i.d. samples $S = \{x^{(1)}, x^{(2)}, ...\}$ from an unknown distribution, we have a parametric model $p_\theta(x)$ which is the probability of observing x given the parameters of the model, θ. So whats the best θ? Well the usual way to pick θ is according to the maximum likelihood principle.

$$\max_\theta \prod_{x^{(i)} \in S} p_\theta(x^{(i)}) \tag{10.1}$$

Because $\log$ is monotone, this can be rewritten as

$$\max_\theta \sum_{x^{(i)} \in S} \log p_\theta(x^{(i)}) \tag{10.2}$$

One question to ask about this, however, is how stable this estimate of θ is to noise. As an aside, most methods for learning densities also come with a candidate for $p(h|x)$. Assuming you can sample from this, you have a representation of h given x, which is perhaps why people conflate all these terms.

10.3 Variational methods (calculus of variations)

We will want to maintain some estimate $q(h|x)$ of $p(h|x)$. One useful fact is that:

$$\log p(x) \geq \mathbb{E}_{q(h|x)}[\log(p(x,h))] + H[q(h|x)], \quad \forall q(h|x) \tag{10.3}$$

where H is the Shannon Entropy.

We would like to prove this bound on $\log p(x)$ and resort to maximizing the lower bound given in (10.3), referred to as the evidence lower bound (ELBO). Towards this end we will introduce the Kullback Leibler divergence (KL) between two distributions given by

$$KL[q(h|x)\,||\,p(h|x)] = \mathbb{E}_{q(h|x)}\left[\log \frac{q(h|x)}{p(h|x)}\right] \tag{10.4}$$

Moreover, $p(x)p(h|x) = p(x,h)$ is true by Bayes Rule. Then we can see that

$$\begin{aligned} KL[q(h|x)|p(h|x)] &= \mathbb{E}_{q(h|x)}[\log \frac{q(h|x)}{p(x,h)} \cdot p(x)] && (10.5)\\ &= \underbrace{\mathbb{E}_{q(h|x)}[\log(q(h|x))]}_{-H(q(h|x))} - \mathbb{E}_{q(h|x)}[\log(p(x,h))] + \mathbb{E}_{q(h|x)}[\log p(x)] && (10.6) \end{aligned}$$

But we know that the KL divergence is always nonnegative, so we get:

$$\mathbb{E}_{q(h|x)}[\log(p(x))] - \mathbb{E}_{q(h|x)}[\log(p(x,h))] - H(q(h|x)) \geq 0 \tag{10.7}$$

which is the same as ELBO (10.3) since $\log(p(x))$ is constant over $q(h|x)$, hence is equal to its expectation.

10.4 Autoencoders

An autoencoder learns to map data from an input x into a representation z, from which the model could recover x' a value that is close to the original input. In other words, it is trying to learn an approximation to the identity function, so as to output x' that is similar to x. The identity function seems trivial to be trying to learn; but by placing constraints on the model, such as by limiting the representation z to be of low dimensionality, we can discover interesting structures about the data. In the simplest example, assume k vectors $u_1, ..., u_k \in \mathbb{R}^n$, where $k \ll n$, and $x = \sum_i \alpha_i u_i + \sigma$, where σ is Gaussian noise. By applying rank-k PCA, one could recover values in $span(u_1, ..., u_k)$.

10.4.1 Sparse autoencoder

Our argument above relied on the size of the encoded representation of the data to be small. But even when this is not the case, (i.e., $k > n$), we can still learn meaningful structure, by imposing other constraints on the network. In particular, we could impose sparsity constraints on the mapped value. Essentially, if $x = \sum_i \alpha_i u_i + \sigma$ as above, we could enforce $\boldsymbol{\alpha}$ to be r-sparse, i.e., allowing only r non-zero values. Examples of ”sparse coding” include [4, 2].

10.4.2 Topic models

Topic models are used to learn the abstract "topics" that occur in a collection of documents, and uncover hidden semantic structures. Assume the documents are given in a bag-of-words representation. As defined above, we have $u_1, ..., u_k \in \mathbb{R}^n$ vectors, with $k < n$. We enforce that $\sum_i \alpha_i = 1$, such that α_i is the coefficient of the i-th topic. For example, a news article might be represented as a mixture of 0.7 of the topic *politics* and 0.3 of the topic *economy*. The goal in topic modeling is, when given a large enough collection of documents, to discover the underlying set of topics used to generate them, both efficiently and accurately. A practical algorithm for topic modeling with provable guarantees is given by [1].

10.5 Variational Autoencoder (VAE)

In the context of deep learning, Variational Autoencoders (VAEs) [3] are one of the earliest models that have demonstrated promising qualitative performance in learning complicated distributions. As its name suggests two core classical ideas rest behind the design of VAEs: autoencoders – the original data $x \in \mathbb{R}^n$ is mapped into a high-level descriptor $z \in \mathbb{R}^d$ on a low dimensional (hopefully) meaningful manifold; variational inference – the objective to maximize is a lower bound on log-likelihood instead of the log-likelihood itself.

Recall that in density estimation we are given a data sample $x_1, \ldots, x_m$ and a parametric model $p_\theta(x)$, and our goal is to maximize the log-likelihood of the data: $\max_\theta \sum_{i=1}^m \log p_\theta(x_i)$. As a variational method, VAEs use the evidence lower bound (ELBO) as a training objective instead. For any distributions p on (x, z) and q on $z|x$, ELBO is derived from the fact that $KL(q(z|x) \,||\, p(z|x)) \geq 0$

$$\log p(x) \geq \mathbb{E}_{q(z|x)}[\log p(x, z)] - \mathbb{E}_{q(z|x)}[\log q(z|x)] = ELBO \tag{10.8}$$

where equality holds if and only if $q(z|x) \equiv p(z|x)$. In the VAE setting, the distribution $q(z|x)$ acts as the encoder, mapping a given data point x to a distribution of high-level descriptors, while $p(x, z) = p(z)p(x|z)$ acts as the decoder, reconstructing a distribution on data x given a random seed $z \sim p(z)$. Deep learning comes in play for VAEs when constructing the aforementioned encoder q and decoder p. In particular,

$$q(z|x) = \mathcal{N}(z; \mu_x, \sigma_x^2 I_d), \quad \mu_x, \sigma_x = E_\phi(x) \tag{10.9}$$

$$p(x|z) = \mathcal{N}(x; \mu_z, \sigma_z^2 I_n), \quad \mu_z, \sigma_z = D_\theta(z), \quad p(z) = \mathcal{N}(z; 0, I_d) \tag{10.10}$$

where E_ϕ and D_θ are the encoder and decoder neural networks parameterized by ϕ and θ respectively, μ_x, μ_z are vectors of corresponding dimensions, and σ_x, σ_z are (nonnegative) scalars. The particular choice of Gaussians is not a necessity in itself for the model and can be replaced with any other relevant distribution. However, Gaussians provide, as is often the case, computational ease and intuitive backing. The intuitive argument behind the use of Gaussian distributions is that under mild regularity conditions every distribution can be approximated (in distribution) by a mixture of Gaussians. This follows from the fact that by approximating the CDF of a distribution by step functions one obtains an approximation in distribution by a mixture of constants, i.e. mixture of Gaussians with ≈ 0 variance. The computational ease, on the other hand, is more clearly seen in the training process of VAEs.

10.5.1 Training VAEs

As previously mentioned, the training of variational autoencoders involves maximizing the RHS of (10.8), the ELBO, over the parameters ϕ, θ under the model described by (10.9), (10.10). Given that the parametric model is based on two neural networks E_ϕ, D_θ, the objective optimization is done via gradient-based methods. Since the objective involves expectation over $q(z|x)$, computing an exact estimate of it, and consequently its gradient, is intractable so we resort to (unbiased) gradient estimators and eventually use a stochastic gradient-based optimization method (e.g. SGD).

In this section, use the notation $\mu_\phi(x), \sigma_\phi(x) = E_\phi(x)$ and $\mu_\theta(z), \sigma_\theta(z) = D_\theta(z)$ to emphasize the dependence on the parameters ϕ, θ. Given training data $x_1, \dots, x_m \in \mathbb{R}^n$, consider an arbitrary data point $x_i, i \in [m]$ and pass it through the encoder neural network E_ϕ to obtain $\mu_\phi(x_i), \sigma_\phi(x_i)$. Next, sample s points $z_{i1}, \dots, z_{is}$, where s is the batch size, from the distribution $q(z|x = x_i) = \mathcal{N}(z; \mu_\phi(x_i), \sigma_\phi(x_i)^2 I_d)$ via the reparameterization trick [3] by sampling $\epsilon_1, \dots, \epsilon_s \sim \mathcal{N}(0, I_d)$ from the standard Gaussian and using the transformation $z_{ij} = \mu_\phi(x_i) + \sigma_\phi(x_i) \cdot \epsilon_j$. The reason behind the reparameterization trick is that the gradient w.r.t. parameter ϕ of an unbiased estimate of expectation over a general distribution q_ϕ is not necessarily an unbiased estimate of the gradient of expectation. This is the case, however, when the distribution q_ϕ can separate the parameter ϕ from the randomness in the distribution, i.e. it's a deterministic transformation that depends on ϕ of a parameter-less distribution. With the s i.i.d. samples from $q(z|x = x_i)$ we obtain an unbiased estimate of the objective ELBO

$$\sum_{j=1}^{s} \log p(x_i, z_{ij}) - \sum_{j=1}^{s} \log q(z_{ij}|x_i) = \sum_{j=1}^{s} [\log p(x_i|z_{ij}) + \log p(z_{ij}) - \log q(z_{ij}|x_i)] \tag{10.11}$$

Here the batch size s indicates the fundamental tradeoff between computational efficiency and accuracy in estimation. Since each of the terms in the sum in (10.11) is a Gaussian distribution, we can write the ELBO estimate explicitly in terms of the parameter-dependent $\mu_\phi(x_i), \sigma_\phi(x_i), \mu_\theta(z_{ij}), \sigma_\theta(z_{ij})$ (while skipping some constants). A single term for $j \in [s]$ is given by

$$-\frac{1}{2}\left[\frac{||x_i - \mu_\theta(z_{ij})||^2}{\sigma_\theta(z_{ij})^2} + n \log \sigma_\theta(z_{ij})^2 + ||z_{ij}||^2 - \frac{||z_{ij} - \mu_\phi(x_i)||^2}{\sigma_\phi(x_i)^2} - d \log \sigma_\phi(x_i)^2\right] \tag{10.12}$$

Notice that (10.12) is differentiable with respect to all the components $\mu_\phi(x_i), \sigma_\phi(x_i), \mu_\theta(z_{ij}), \sigma_\theta(z_{ij})$ while each of these components, being an output of a neural network with parameters ϕ or θ, is differentiable with respect to the parameters ϕ or θ. Thus, the tractable gradient of the batch sum (10.11) w.r.t. ϕ (or θ) is, *due to the reparameterization trick*, an unbiased estimate of $\nabla_\phi ELBO$ (or $\nabla_\theta ELBO$) which can be used in any stochastic gradient-based optimization algorithm to maximize the objective ELBO and train the VAE.

10.6 Main open question

Main open question for this lecture is to design methods with provable guarantees for the learning problems discussed here. Can we show that VAEs correctly (and efficiently) learn simple families of probability distributions?

There were notable successes in analysis of method of moments for learning probability distributions, as mentioned above. Variational methods rely upon gradient descent, which seems harder to analyse as of now.

References

[1] Sanjeev Arora, Rong Ge, Yonatan Halpern, David Mimno, Ankur Moitra, David Sontag, Yichen Wu, and Michael Zhu. A practical algorithm for topic modeling with provable guarantees. *ICML*, 2013.

[2] Sanjeev Arora, Rong Ge, Tengyu Ma, and Ankur Moitra. Simple, efficient, and neural algorithms for sparse coding. *PMLR*, 2015.

[3] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *ICLR*, 2014.

[4] Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325, 1997.