Deep Density Estimation
10716, Spring 2020
Pradeep Ravikumar

1 Introduction

Consider the density estimation problem where we wish to estimate the density p of some distribution P, and where we are given samples $\{X_i\}_{i=1}^n$ drawn iid from that distribution. Suppose we wish to do parametric density estimation: we then start with a parametric class of densities $\{p_\theta\}_{\theta \in \Theta}$, and then estimate the density $p_{\widehat{\theta}}$, for some $\widehat{\theta} \in \Theta$, with the best fit to the data $\{X_i\}_{i=1}^n$. There are two technical facets to this: (a) how to specify a parametric family of densities, and (b) how to determine goodness of fit of any member of the family to data.

Let us consider the first technical component. A popular class of univariate parametric families is the class of exponential family distributions, which include popular distributions such as the univariate Gaussian, Bernoulli, Poisson, and exponential distributions, among others. These have also been extended to the multivariate case [Yang et al., 2015], which moreover have some deeper connections to probabilistic graphical models. These parametric classes could be enriched further via mixtures of such exponential family distributions, but nonetheless, these might not always fit data such as images, with low-level features, very well.

Now instead of such "classical" parametric families, suppose we have a very expressive class of parametric functions $\{f_\theta\}$ (e.g. deep neural networks) that can approximate very complex functions very well. How do we use these for density estimation? One caveat to directly using these as a class of parametric densities is the contraint that the densities be non-negative, and integrate to one. One approach to enforce that is by parameterizing the logistic transform $\eta(x)$ instead, so that $p_\theta(x) = \frac{\exp(\eta_\theta(x))}{\int_{x \in \mathcal{X}} \exp(\eta_\theta(x)) dx}$ is non-negative and is normalizable by construction, with no further constraints on $\eta(x)$ (other than for identifiability such as that $\int_{x \in \mathcal{X}} \eta(x) = 0$, or $\eta(x_0) = 0$, for some $x_0 \in \mathcal{X}$). The main caveat with this approach is the normalization constant, which involves a multi-dimensional integral. Thus, the MLE estimate of the parameters would yield:

$$\inf_\theta \left\{ -\frac{1}{n} \sum_{i=1}^n \eta_\theta(x_i) + \log \int_{x \in \mathcal{X}} \exp(\eta_\theta(x)) dx \right\},$$

which is in general intractable due to the multi-dimensional integral. Approaches to address this range from approximations to the normalization factor, to surrogate likelihoods. Jeon and Lin [2006] for instance (in the context of more general non-parametric density estimation)

suggest the following estimator:

$$\inf_{\theta} \left\{ \frac{1}{n} \sum_{i=1}^{n} \exp(-\eta_\theta(x_i)) + \int_{x \in \mathcal{X}} \eta_\theta(x) \rho(x) dx \right\},$$

where $\rho(x)$ is any simpler known density with the same support as the true density $p(x)$. As they show, the M-estimator above is consistent, and moreover is much more tractable than the MLE. But overall, such surrogate likelihood approaches coupled with the logistic transform seem less popular, since their theoretical properties are less well-understood, and perhaps empirically they have not performed as well.

Over the last decade there have been a slew of alternative approaches that sidestep the logistic transform route, with its normalization difficulties, altogether and instead specify the random vector X as a *transformation* of some other latent variables, with some known distribution, and since these transformations in general can be relatively unconstrained, we sidestep issues of normalizability. These transformations typically involve deep neural network based parametric functions, and hence are loosely called deep density estimators.

2 Variational Auto-Encoders

A very natural approach [Kingma and Welling, 2013] along these lines is to have:

$$\begin{aligned} Z &\sim N(0, I_d) \\ X|Z = z &\sim N(\mu_\theta(z), \sigma^2_\theta(z) I), \end{aligned}$$

or alternatively:

$$X = \mu_\theta(Z) + \sigma_\theta(Z)\, W$$

where $Z, W \sim N(0, I)$. Thus, X has a well-defined density even when the mean and variance functions $\{\mu_\theta(z), \sigma_\theta(z)\}_{\theta \in \Theta}$ are relatively unconstrained, with no normalization terms. For instance, a highly expressive parametric family of choice for these mean and variance functions are deep neural networks. The density of X is then given as:

$$p(X; \theta) = \int_{z \in \mathbb{R}^d} p_N(x; \mu_\theta(z), \sigma^2_\theta(z) I)\, p_N(z; 0, I) dz,$$

where $p_N(\mu, \Sigma)$ is the multivariate Gaussian density. It can be seen that the density does not have an explicit tractable form. To fit the parameters θ to the data, we could maximize the likelihood of the observed data:

$$\max_{\theta} \frac{1}{n} \sum_{i=1}^{n} \log p(x_i; \theta),$$

but this is typically difficult due to the integral over the latent Gaussians. Even if we were interested in Bayesian inference, the posterior $P(Z|X)$ is intractable as well. At this stage, it is not clear if we have gained anything over the logistic transform: substituting one high-dimensional integral for another.

As before, we could optimize a surrogate likelihood instead. In so-called variational inference, we compute parameterized lower bounds of the likelihood and optimize this lower bound instead. Thus, if $p_\theta(X) \geq g_{\theta;\gamma}(X)$, for $\gamma \in \Gamma$, then we solve for:

$$\max_{\theta \in \Theta, \gamma \in \Gamma} \frac{1}{n} \sum_{i=1}^{n} \log g_{\theta;\gamma}(x_i).$$

With the above latent variable model, we have the following classical variational bound, also called the Evidence Lower Bound or ELBO:

$$\begin{aligned}
\log p_\theta(x) &= \log \int_z p_\theta(x, z) dz \\
&= \log \int_z q_\phi(z|x) p_\theta(x|z) p(z) / q_\phi(z|x) \\
&\geq \int_z q_\phi(z|x) \log p(z)/q_\phi(z|x) + \int_z q_\phi(z|x) \log p_\theta(x|z) \\
&= L(\theta, \phi, x) := -\mathrm{KL}(q_\phi(z|x) \| p(z)) + \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)],
\end{aligned}$$

so that instead of the maximizing the empirical expectation of $\log p_\theta(x)$, we maximize the empirical expectation of the lower bound $L(\theta, \phi, x)$ instead. We can moreover show that:

$$\log p_\theta(x) - L(\theta, \phi, x) = \mathrm{KL}(q_\phi(z|x) || p_\theta(z|x)),$$

so that the ELBO bound gets tighter as the variational approximation $q_\phi(z|x)$ gets closer to the intractable true posterior $p_\theta(z|x)$.

A natural flexible parameterization is simply:

$$q_\phi(z|x) = N(\mu_\phi(x), \sigma_\phi^2(x) I),$$

where again the mean and variance functions $\mu_\phi(x), \sigma_\phi(x)$, can again be parameterized by flexible families such as deep neural networks. Note that when taking an expectation $\mathbb{E}_{q_\phi(z|x)}[f(z)]$, we can "reparameterize" $z = h(x, w) := \mu_\phi(x) + \sigma_\phi(x) w$ in terms of $w \sim N(0, I)$, so that

$$\mathbb{E}_{q_\phi(z|x)}[f(z)] = \mathbb{E}_{w \sim N(0,I)}[f(h(w, x))],$$

which can be approximated via Monte Carlo samples of w, and no explicit density calculations of $q_\phi(z|x)$. This is called the "reparameterization trick".

The above approach is also called the Variational Auto-encoder, since it is reminiscent of auto-encoders that were used to learn compact representations of the input x. As an instance,

suppose we wish to get a representation $z \in \mathbb{R}^d$ of the input $x \in \mathbb{R}^p$ for some $d < p$, via the following "encoder" model:

$$z = g(b + Wx),$$

which is then coupled with a "decoder" model:

$$\widehat{x} = g(c + Vz),$$

for some point-wise non-linearity $g(\cdot)$, and some vectors b, c, and matrices W, V. We could learn these parameters by minimizing the reconstruction error:

$$\inf_{\theta} \sum_i \|\widehat{x}_i - x_i\|.$$

Here the "encoder" transformation from x to z, as well as a "decoder" transformation from z to x are both deterministic, and hence this does not specify a density model for x. With the variational autoencoder, both these transformations are stochastic, and moreover, there was an explicit distribution imposed on the latent representations z, which thus specified a distribution over the inputs x.

While in the original variational auto-encoder, $q_\phi(z|x)$ was set to be a Gaussian with parameterized mean and variance, one could also use other flexible parameterizations, including the invertible neural networks or normalizing flows to be discussed in the next section.

One could also use a stacked set of latent Gaussians as:

$$\begin{aligned} z_L &\sim N(0, I) \\ z_l|z_{l+1} &\sim N(\mu_l(z_{l+1}), \sigma_l^2(z_{l+1})I) \\ x|z_1 &\sim N(\mu_0(z_1), \sigma_0^2(z_1)I) \end{aligned}$$

in what are called Deep Latent Gaussian Models [Rezende et al., 2014], though these seem less popular, perhaps due to the added complexity.

3 Normalizing Flows

Suppose as before, we have a latent representation $Z \sim N(0, I)$. But now, suppose we have a deterministic transformation from Z to X as:

$$X = g_\theta(Z),$$

for some flexible parametric function g_θ. Suppose g_θ is invertible (which is a big if). Then by the change of variables formula:

$$p_{X;\theta}(x) = p_Z(g_\theta^{-1}(x))|\text{det}Jg^{-1}(x)|,$$

where $[Jh(x)]_{ij} = \partial h_i(x)/\partial x_j$, so that the density has a nice closed form expression. Thus, given samples $\{x_i\}_{i=1}^n$, we could thus directly solve for the MLE:

$$\inf_\theta \sum_{i=1}^n -\log p_{X;\theta}(x_i).$$

Note that these can be stacked, so that we could obtain a stacked transformation $Z_K = g_K \circ \ldots \circ g_1(Z_0)$, which in turn will have the log-density:

$$\log p_K(z_K) = \log p_0(z_0) - \sum_{k=1}^K \log |\det Jg_k(z_k)|.$$

The path formed by the random variables Z_k is called a "flow," and the path formed by the distributions P_k is called a "normalizing flow" [Rezende and Mohamed, 2015].

Note that by the so-called reparameterization trick introduced earlier $E_{p_K}[h(X)] = E_{p_0}[h(g_K \circ \ldots \circ g_1(z_0))]$ which does not involve Jacobian calculations.

3.1 Invertible Maps

The key caveats with normalizing flows are two-fold: (a) the transformation g_θ has to be invertible, and (b) the density involves the Jacobian of the transformation, which could be expensive for general invertible maps.

Some simple classes of invertible transformations (which as noted above can be stacked to get "deep" flow tranforms) include:

$$g(z) = z + uh(w^T z + b),$$

which are invertible for specific settings of (h, u, w) e.g. $h = \tanh(\cdot)$ and $w^T u >= -1$ [Rezende and Mohamed, 2015].

An alternative approach, called NICE [Dinh et al., 2014], is to split $X = (X_1, X_2)$ as well as $Z = (Z_1, Z_2)$ into two blocks of variables with the blocked transform:

$$\begin{aligned} X_1 &= Z_1 \\ X_2 &= Z_2 + m(Z_1), \end{aligned}$$

for an arbitrary, potentially non-invertible function $m(\cdot)$. It can be seen that the transformation from Z to X is trivially invertible:

$$\begin{aligned} Z_1 &= X_1 \\ Z_2 &= X_2 - m(X_1). \end{aligned}$$

Moreover, the Jacobian of the transformation is triangular, so that its determinant is simply the product of diagonal entries, and hence easy to compute. A related triangular Jacobian transformation [Dinh et al., 2016] is given by:

$$X_1 = Z_1$$
$$X_2 = Z_2 \odot \exp(m_1(Z_1)) + m_2(Z_1),$$

which can again be trivially inverted for arbitrary $m_1(\cdot), m_2(\cdot)$, via:

$$Z_1 = X_1$$
$$Z_2 = (X_2 - m_2(X_1)) \odot \exp(-m_1(X_1)).$$

4 Autoregressive Flows

The simple triangular Jacobian examples had an implicit auto-regressive character: we could specify the joint distribution via the marginal distribution of a subset of variables X_1, and the conditional distribution of the remaining subset X_2 conditioned on X_1. Autoregressive flows generalize this to allow for more general auto-regressive transformations.

In a so-called Masked Autoregressive Flow (MAF) [Papamakarios et al., 2017], this is given as:

$$X_i = \mu_i + Z_i \exp(\alpha_i),$$

where $Z_i \sim N(0, 1)$, and $\mu_i = g_{\mu_i}(X_{<i})$, and $\alpha_i = g_{\sigma_i}(X_{<i})$, so that X is a transformation of the standard Gaussian vector Z, and where the transformation is specified in an auto-regressive manner. It can be seen that the inverse is easily computed:

$$Z_i = (X_i - \mu_i) \exp(-\alpha_i),$$

so that Z can be recovered given X, and that moreover the determinant of the Jacobian of the transformation $X = g(Z)$ is easily computed as $|\det Jg^{-1}(x)| = \exp(-\sum_i \alpha_i)$. MAF can transform X to Z in one (parallelized) iteration, but requires p iterations to transform Z to X.

A variant of MAF is Inverse Autoregressive Flow (IAF) [Kingma et al., 2016], where we have:

$$X_i = \mu_i + Z_i \exp(\alpha_i),$$

where $\mu_i = g_{\mu_i}(Z_{<i})$, and $\alpha_i = g_{\alpha_i}(Z_{<i})$. Its inverse is again given as:

$$Z_i = (X_i - \mu_i) \exp(-\alpha_i),$$

but note that in this case, IAF can transform Z to X in one vectorized iteration, but requires p iterations to transform X to Z. Thus the slight difference in choices between IAF and MAF

can result in vastly different computational times for specific tasks. Note that transform X to Z is required for calculating the density of a point X, while transforming Z to X is required to generate new samples.

Stacking such auto-regressive transformations $X = g_K \circ \ldots \circ g_1(Z)$ is called an "autoregressive flow", as a special instance of normalizing flows.

4.1 General Auto-regressive Distributions

Classically, auto-regressive models were used to directly parameterize joint distributions (rather than simply transformations) via parameterizing conditional distributions specified by the standard chain rule

$$p_\theta(x) = \prod_{i=1}^{p} p_\theta(x_i | x_{<i}).$$

A classical approach to model $p_\theta(x_i|x_{<i})$ is to make a Markov assumption that $p_\theta(x_i|x_{<i}) = p_\theta(x_i|x_{i-1}, \ldots, x_{i-k})$ so that the conditional distribution of X_i conditioned on all previous variables only depends on the k most recent variables before X_i. Another approach is to use sequence model based recurrences, such as:

$$\begin{aligned} h_i &= f_{\theta_h}(h_{i-1}) \\ x_i &= f_{\theta_x}(x_{i-1}, h_i), \end{aligned}$$

for some parametric functions $f_{\theta;h}(\cdot)$, and $f_{\theta;x}(\cdot, \cdot)$. Such recurrence based sequence models, such as recurrent neural networks (RNNs), are by now the parametric models of choice for sequence based data where the sequence order is very naturally specified, for instance, via time. But they are far less popular when there is no such natural sequence order, in large part because the performance of such models is very sensitive to such ordering. This can be seen even in the two variable case in the following example where an auto-regressive model with the ordering (x_1, x_2) is able to model the data, but an auto-regressive model with the ordering (x_2, x_1) is not able to. [Figure 1 from Masked Autoregressive flow for density estimation].

One approach to address this to use different orderings, and use an ensemble or mixture of the resulting distributions. Another approach is to use different orderings in each layer of an "autoregressive flow" $X = g_K \circ \ldots \circ g_1(Z)$, where we use a different ordering for each auto-regressive transformation g_i, for $i = 1, \ldots, K$. This provides another rationale for the use of auto-regressive flows, rather than sequence based auto-regressive recurrence models, in addition to other benefits of normalizing flows, such as the ease of computing the density, and sampling.

5 Generative Adversarial Networks (GANs)

Suppose we have a parametric transformation $X = g_\theta(Z)$ of some base distribution z. Provided the transformation g_θ is invertible as with normalizing flows, we can compute the density $p_\theta(x)$ of the random vector x, and consequently use MLE:

$$\widehat{\theta} \in \arg\inf_\theta \frac{1}{n}\sum_{i=1}^{n} \log p_\theta(x_i),$$

to estimate the parameters $\widehat{\theta}$ given samples $\{x_i\}_{i=1}^n$. There are two caveats here. The first is that this is not feasible when g_θ is not invertible, which would be the case for instance, for most modern architectures of deep neural networks. The second caveat is more subtle, and is due to the very nature of the MLE as minimizing the empirical variant of the KL divergence between the true data distribution P and the implicit distribution P_θ over X with density p_θ:

$$\inf_\theta \mathrm{KL}(P, P_\theta).$$

Note that $KL(P,Q) = \int p(x) \log p(x)/q(x) dx$, so that this would be large if there are P-likely regions where $q(x)$ is small and $p(x)$ is large: which encourages $q(x)$ to have support in the P-likely regions of the input space. But this does not ensure that $q(x)$ be small where $p(x)$ is small: such a property would ne required to ensure that samples from Q be P-realistic (i.e. do not have small density with respect to true data distribution P). How do we encourage the latter property? By simply minimizing $KL(Q,P) = \int q(x) \log q(x)/p(x) dx$, which would be large if there are Q-likely regions where $p(x)$ is small and $q(x)$ is large. A caveat with $KL(Q_\theta, P)$ on the other hand is practical: it is not decomposable, so that it is not clear how to optimize this given just samples $\{x_i\}_{i=1}^n$ from P. Combining both these asymmetric KL divergences yields the Jensen-Shannon divergence:

$$\mathrm{JSD}(P,Q) = \mathrm{KL}\left(P\,, \frac{P+Q}{2}\right) + \mathrm{KL}\left(Q\,, \frac{P+Q}{2}\right),$$

which has the additional advantage of being symmetric in its arguments. This loss again is not decomposable, so that it is not clear how to optimize this given just samples $\{x_i\}_{i=1}^n$ from P. In a seminal paper, Goodfellow et al. [2014] showed that one can indeed minimize the Jensen-Shannon divergence given samples by considering a variational form using "generators" and "discriminators".

Suppose $D : \mathcal{X} \mapsto [0,1]$ be a classifier (ideally probabilistic, but more generally with an output of classifier scores between 0 and 1). Given the parameterized density q_θ, and the density of the true data distribution p, consider the following variational form:

$$V(p, q_\theta, D) = \mathbb{E}_{x\sim p}[\log D(x)] + \mathbb{E}_{x\sim q_\theta}[\log(1 - D(x))].$$

Goodfellow et al. [2014] then showed the following useful result:

$$\max_D V(p, q_\theta, D) = -\log(4) + 2\mathrm{JSD}(p, q_\theta),$$

so that

$$\arg\min_{\theta}\max_{D} V(p, q_\theta, D) = \arg\min_{\theta} \text{JSD}(p, q_\theta).$$

The interesting facet of $V(p, q_\theta, D)$ is that it is decomposable, so that it can be approximated well via samples (from both p as well as q_θ), thus facilitating learning the parameters of the density q_θ by minimizing the Jensen-Shannon divergence itself with respect to the true data distribution.

The variational objective $V(p, q_\theta, D)$ can also be motivated as specifying a min-max adversarial game between the "generative" density q_θ, and a discriminator D that aims to discriminate between samples from Q_θ and P, while the generator Q_θ aims to fool the discriminator D. Specifically, consider the following classification task, where $Y = 1$ indicates the true data distribution and $Y = 0$ indicates Q_θ, so that $X|(Y = 1) \sim P$, and $X|(Y = 0) \sim Q_\theta$. The expected cross-entropy loss of a probabilistic discriminator $D : \mathcal{X} \mapsto [0, 1]$ is then given by

$$\begin{aligned}
&\mathbb{E}\left[Y \log D(X) + (1 - Y) \log(1 - D(X))\right] \\
&= \mathbb{E}[\log D(X)|Y = 1]P(Y = 1) + \mathbb{E}[\log(1 - D(X))|Y = 0]P(Y = 0) \\
&= 0.5 * \mathbb{E}_{X\sim P}[\log D(X)] + 0.5 * \mathbb{E}_{X\sim Q_\theta}[\log(1 - D(X))] \\
&= 0.5 * V(p, q_\theta, D)
\end{aligned}$$

using $P(Y = 1) = P(Y = 0) = 1/2$.

Thus, the variational objective $V(p, q_\theta, D)$ is simply twice the expected cross entropy loss of the discriminator $D(\cdot)$ in the classification task of discriminating between the true distribution P and the generative model Q_θ.

6 Destructive Distribution Learning

So far we have considered a largely "constructive" learning approach where we learn a transformation $g_\theta(Z)$ of a random vector Z with known simple distribution (such as independent Gaussian) and fit the parameters so that the transformed distribution $P_{g_\theta(Z)}$ is as close to the true data distribution P_X as possible, for instance by solving for the following (population) objective:

$$\inf_{\theta} KL(P_X, P_{g_\theta(Z)}).$$

An alternative approach is to consider a "destructive" learning approach where we learn a transformation $h_\theta(X)$ of the data random vector X, and fit the parameters so that the transformed distribution $P_{h_\theta(X)}$ is as close to a random vector Z with known simple distribution (such as independent Gaussian). Such a transformation is called destructive learning since we aim to "destroy" the structure in X, reducing it to say an independent Gaussian distribution.

But while (imperfectly) transforming Z to X seems useful from a density estimation perspective, why would we want to (imperfectly) transform X to Z? There are two reasons to do so. The first is that of representation learning, as we discuss further below. The second is that we could also use it as density estimation procedure.

6.1 Representation Learning

Tranforming X to Z with known or simple distribution could be cast as "encoding" the data X into a representation that is "simple". A variant of this is Independent Component Analysis (ICA), where we (typically) only assume that Z is independent. This is not however sufficient to make the transformation identifiable. For instance, if Z_1 and Z_2 are independent, then so are component-wise transformations $f_1(Z_1)$ and $f_2(Z_2)$. Even if we restrict the distribution of the independent vector Z, the indeterminacy remains. Suppose we have a transformation $h : \mathcal{X} \mapsto [0,1]^d$ that maps X to a uniform random vector $Z \in [0,1]^d$. Then, given any measure-preserving automorphism $g : [0,1]^d \mapsto [0,1]^d$, it is clear that $g \circ h$ will be another solution to the ICA problem of transforming X to a uniform random vector.

So, such a destructive mapping, if it exists, is not unique. But what about existence of such a mapping? One can show this via the following constructive mapping.

Suppose we set $Z_1 \sim \text{Unif}[0,1]$. And for $j = 2, \ldots, d$, denote $F_j(x; z_1, ..., z_j) = P(X \leq x | Z_1 = z_1, \ldots, Z_j = z_j)$ as the conditional CDF of X conditioned on j uniform random variables $\{Z_\ell\}_{\ell=1}^j$. Then set $Z_{j+1} = F_j(X; z_1, \ldots, z_j)$. It can be seen that $(Z_1, ..., Z_d) \sim \text{Unif}[0,1]^d$. This is simply the multivariate extension of the classical univariate CDF transformation result that $F_V(V)$ Unif$[0,1]$, where $V \in \mathbb{R}$ is some real-valued random variable, and F_V is its CDF. Thus, stitching these conditional CDF transformations together, we get the mapping: $Z = h(X)$. The main caveat with this constructive mapping is that such conditional CDFs are difficult to estimate for multivariate data.

6.2 Density Estimation

The other reason we might want to learn an imperfect mapping $h_\theta(\cdot)$ from X to Z is that in the limit where we are able to truly convert X to Z with known distribution, then we can recover the density of X by the change of variable formula applied to the transformation $h_\theta^{-1}(Z)$, so that

$$p_{h_\theta^{-1}(Z)}(x) = p_Z(h_\theta(x))|\text{det}Jh_\theta(x)|,$$

as with normalizing flows. Since h_θ is an imperfect transformation of X to Z, similarly, h_θ^{-1} will be an imperfect transformation from Z to X, which is the case with constructive approaches such as normalizing flows as well. A more critical concern with the destructive learning approach is computational/practical.

Consider the objective:

$$\begin{aligned}&\inf_\theta \mathrm{KL}(P_{h_\theta(X)}, P_Z)\\ =&\inf_\theta \int_z p_{h_\theta(X)}(z) \log p_{h_\theta(X)}(z)/p_Z(z) dz.\end{aligned}$$

It can be seen that it is not clear how to optimize this objective given just samples $\{x_i\}_{i=1}^n$ drawn from P_X, since without access to the true density p_X, we might not be able to evaluate the transformed density $p_{h_\theta(X)}(z)$ (note that in the case of normalizing flows, we had access to the base density p_Z, and so could evaluate the density of transformations of this base density). But the following simple identity essentially notes reduces it to constructive learning:

Theorem 1 (Destructive-Constructive Identity)

$$KL(P_{h_\theta(X)}, P_Z) = KL(P_X, P_{h_\theta^{-1}(Z)}).$$

This theorem has (re-)appeared in many recent generative model papers; see for instance [Ballé et al., 2015, Papamakarios et al., 2017]. The proof just follows from some applications of the change of variables formula:

$$\begin{aligned}\mathrm{KL}(P_{h_\theta(X)}, P_Z) &= \int_z p_{h_\theta(X)}(z) \left(\log p_{h_\theta(X)}(z) - \log p_Z(z)\right) dz\\ &= \int_x p_X(x) \left|\det \frac{\partial x}{\partial z}\right| \left(\log p_X(x) \left|\det \frac{\partial x}{\partial z}\right| - \log p_Z(h_\theta(x))\right) \left|\det \frac{\partial z}{\partial x}\right| dx\\ &= \int_x p_X(x) \left(\log p_X(x) - \log p_Z(h_\theta(x)) \left|\det \frac{\partial z}{\partial x}\right|\right) dx\\ &= \int_x p_X(x) \left(\log p_X(x) - \log p_{h_\theta^{-1}(Z)}(x)\right) dx\\ &= \mathrm{KL}(P_X, P_{h_\theta^{-1}(Z)}),\end{aligned}$$

where $\frac{\partial x}{\partial z} = Jh_\theta^{-1}(h_\theta(x))$, and $\frac{\partial z}{\partial x} = Jh_\theta(x)$, and where we used the property of Jacobians that $\frac{\partial x}{\partial z} = (\frac{\partial z}{\partial x})^{-1}$, and the property of determinants that $\det(A^{-1}) = 1/\det(A)$.

Thus, destructive learning is equivalent to constructive learning with invertible transformations, so that it is not clear why we should not simply use constructive learning approaches if we care about density estimation. One methodological advantage could be that we could use insights from other fields to obtain invertible "destructive" transformations. For instance [Ballé et al., 2015] suggest the following "divisive normal" transformation from X to Z:

$$\begin{aligned}U &= H\,X\\ Z_i &= \frac{U_i}{(\beta_i + \sum_{j=1}^d \gamma_{ij} |U_j|^{\alpha_{ij}})^{\epsilon_i}},\end{aligned}$$

for some parameter matrices $H, \alpha, \gamma \in \mathbb{R}^{d\times d}$, and parameter vectors $\beta, \epsilon \in \mathbb{R}^d$ which they motivate from neuroscience considerations.

There are also particular classes of transformations where the solution of the destructive learning problem is easier. Consider the class of transformations of the following form:

$$\mathcal{H} = \{h : \mathcal{X} \subseteq \mathbb{R}^d \mapsto \mathbb{R}^d \,|\, h(x) = (\Psi_1(A_1 x), \ldots, \Psi_d(A_d x)\},$$

where $A \in \mathbb{R}^{d\times d}$, and A_j is the j-th row of A, for $j \in [d]$, and $\{\Psi_j\}_{j=1}^d$ are pointwise invertible univariate transformations. We will denote $\Psi(u) = (\Psi_1(u_1), \ldots, \Psi_d(u_d))$, so that the transformations $h = \Psi(Ax)$ consist of a linear transformation, followed by coordinatewise transformations.

We then wish to solve for:

$$\inf_{\Psi, A} \mathrm{KL}(P_{h_{\Psi,A}(X)}, P_Z).$$

The solution (Ψ^*, A^*) to this can be characterized simply:

$$A^* = \arg\inf_A I(AX), \Psi_j^* \qquad\qquad = \Phi^{-1} F_{A_j^* X}(A_j^* X),$$

where $I(\cdot)$ is the mutual information of a random vector, and $\Phi(\cdot)$ is the standard Gaussian CDF. Minimizing $I(AX)$ over matrices A is essentially the linear ICA problem, which aims to find a linear transformation of X that reduces dependence among the transformed variables as much as possible. While Ψ_j^* is simply a univariate Gaussianization transform, which is a composition of the univariate CDF of the linear transformed variable, and an inverse of the standard Gaussian CDF. Both of these have practical if approximate implementations.

To see why the solution has such a nice closed form, let us first define the marginal KL divergence: marginal-$\mathrm{KL}(P_U, P_V) := \sum_{j=1}^d \mathrm{KL}(P_{U_j}, P_{V_j})$, as the sum of the KL divergences between the corresponding d marginal distributions. From some algebraic calculations, we can then write

$$\mathrm{KL}(P_{\Psi\circ AX}, P_Z) = \text{marginal-KL}(P_{\Psi\circ AX}, P_Z) + I(\Psi \circ AX).$$

But for invertible Ψ, $I(\Psi \circ AX) = I(AX)$, so that Ψ can be obtained by minimizing just the first term, which yields that it is the pointwise Gaussianization of the optimal linear transformation of X. Given this optimal Ψ^*, the first term becomes zero, so that the objective then reduces to the second term which is precises the linear ICA objective $I(AX)$. One can also use the decomposition above to suggest a linear ICA algorithm. Note that if we restrict A to be orthogonal, we then have that:

$$\mathrm{KL}(P_X, P_Z) = \mathrm{KL}(P_{AX}, P_Z) = \text{marginal-KL}(P_{AX}, P_Z) + I(AX).$$

Since the LHS does not depend on A, we then get that:

$$\inf_A I(AX) = \sup_A \text{marginal-KL}(P_{AX}, P_Z),$$

so that we aim to find a linear transformation A that makes the coordinates of AX be as non-Gaussian as possible. The overall optimal solution then seems very intuitive: A aims to find the directions under which the projection of X is most non-Gaussian. Ψ then marginally Gaussianizes these transformed variables.

Both the destructive transforms above might not seem that flexible. But one advantage of destructive transformations such as the above is that we can iterate over these, destroying a bit of X in every iteration. So, starting with $X_1 = X$, in iteration $t = 1, \ldots$, we find $h_t = \arg\inf_{h \in \mathcal{H}} \mathrm{KL}(P_{h(X_t)}, P_Z)$, and then destroy X_t as $X_{t+1} = h_t(X_t)$. A consistency property that would be good to have is that $\mathrm{KL}(P_{X_t}, P_Z) \to 0$. This was shown to indeed be the case with the Gaussianization transformation above [Chen and Gopinath, 2001].

One could also view the iterates above as greedily learnt stacked destructors: $X_{t+1} = h_t \circ \ldots \circ h_1(X)$. Given the equivalence in the beginning of the section, Inouye and Ravikumar [2018] thus suggested the following general algorithm for destructive learning:

$$g_t = \arg\inf_{g \in \mathcal{G}} \mathrm{KL}(P_{X_t}, P_{g(Z)}),$$

for some simple class of invertible functions $\mathcal{G}$, and then use $X_{t+1} = g_t^{-1}(X_t)$. This thus generalizes the Gaussianization transforms above to a much larger class of generative models, that simply fit a generative model over the current data iterate X_t, and then use this generative model to extract a destructive transform to further transform the data and iterate. Note that such a destructive iterative algorithm is much more computationally feasible than a constructive iterative algorithm that would aim to solve:

$$g_t = \arg\inf_{g \in \mathcal{G}} \mathrm{KL}(P_X, P_{g(Z_t)}),$$

where $Z_t = g_{t-1} \circ \ldots \circ g_1(Z)$, where the main computational concern is the computation of the densities of Z_t.

Inouye and Ravikumar [2018] also show that even if we simply solve for simple or shallow density estimation via:

$$Q_t = \arg\inf_{Q \in \mathcal{Q}} \mathrm{KL}(P_{X_t}, P_Q),$$

one can in most cases extract a destructive transform $h(\cdot)$ from P_Q such that $P_Q = P_{h(Z)}$. This further increases the ease of each destructive iteration: one performs shallow density estimation using their method of choice, extract the corresponding destructive transform, and use this to further transform the data, and iterate. So by a series of shallow density estimation procedures, we are able to fit a "deep" density destructively.

One simple approach to extract the invertible destructor $h(\cdot)$, for a given Q (so that $P_{h(Z)} \equiv P_Q$) is to use the conditional univariate CDF transformations discussed earlier. But in most cases such destructive transforms can be obtained even more simply. See [Inouye and Ravikumar, 2018] for examples with many common used shallow densities.

References

Eunho Yang, Pradeep Ravikumar, Genevera I Allen, and Zhandong Liu. Graphical models via univariate exponential family distributions. *The Journal of Machine Learning Research*, 16(1):3813–3847, 2015.

Yongho Jeon and Yi Lin. An effective method for high-dimensional log-density anova estimation, with application to nonparametric graphical model building. *Statistica Sinica*, pages 353–374, 2006.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and variational inference in deep latent gaussian models. In *International Conference on Machine Learning*, volume 2, 2014.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

Johannes Ballé, Valero Laparra, and Eero P Simoncelli. Density modeling of images using a generalized normalization transformation. *arXiv preprint arXiv:1511.06281*, 2015.

Scott Saobing Chen and Ramesh A Gopinath. Gaussianization. In *Advances in neural information processing systems*, pages 423–429, 2001.

David Inouye and Pradeep Ravikumar. Deep density destructors. In *International Conference on Machine Learning*, pages 2167–2175, 2018.