List of topics to cover

With section titles and brief explantions.

Yiftach Kolb

Berlin, September 20, 2022

Abstract

punkt. punkt.

Declaration

punkt. punkt.

Acknowledgement

punkt.[3] punkt.

List of Tables

List of Figures

Contents

Chapter 1

Introduction

punkt. punkt.

Chapter 2

Autoencoders, VAEs, and Variance inference

2.1 Some basic definitions and concepts

Definition 2.1. A *data matrix* is simply a real valued matrix $X \in \mathbb{R}^{N \times n}$ which represent a set of N n-dimensional data points. The N rows are also called *observations* and the n columns are *variables*. For example X may represent N cells over n genes in the case of single-cells RNAseq data.

Definition 2.2. A *feed forward neural network* is simply a differentiable map $\phi : \mathbb{R}^n \to \mathbb{R}^m$. Normally ϕ is a sequence of composition of some more simple functions which we call *layers*. A layer is composed of a single affine function, followed by 0 or more dimension preserving functions such as normalization functions or activation functions. An activation function is a real values non-linear function which is applied element-wise over the input later. For example the sigmoid function anb the ReLU (rectified linear unit) are probably the most well-known activation functions.

Usually together with a neural network comes an associated differentiable function which is the *loss function* $\mathcal{L} : \mathbb{R}^m \to \mathbb{R}$.

The basic functions which comprise ϕ are parametrized. For example each affine function has the form $f : x \to a\dot{x} + b$. a and b are its trainable parameters, whereas the input is treated as fixed (the data we are trying to explain). The set of parameters of ϕ (coming from all the functions comprising it) is also referred to as its *weights*. We can therefore treat ϕ as a parametrized function ϕ_w where w represents the set of trainable weights.

Trining the neural network means finding the weights that minimize the loss function applied on the training set X, in other words minimizing $\min_w(\mathcal{L}(\phi_w(X))$. During a training step the network is applied on a batch X_k (a subset of size $k \leq N$ of X, X_k is therefore a $k \times n$ data matrix). Then the the loss function is applied at the output and a gradient (with relation to the weights) is taken. This gradient is used for the wieght update rule, which varies depending on the specific training algorithm (for example SGD or Adam).

2.2 Autoencoders

An *Autoencoder* (AE) is a neural network $\phi = D \circ E$ with a "bottleneck" layer which approximates the identity function on the training input. We call the part which maps the input into the bottleneck the *encoder* and the part which maps the bottleneck into the input-spce the *decoder*.

Them loss function for the AE is usually $\mathcal{L}(X;\phi) = \|X - \phi(X)\| = \|X - D \circ E(X)\|$.

In fact, for centered data X (every variable has 0 sample mean), the first $k \leq R(X)$ principle components P are the soloution for $P = \text{argmin}_{W^TW=I_k}(\|X - XWW^T\|_F^2)$, whereas a linear autoencoder solves $E, D = \text{argmin}_{E,D}(\|X - XED\|_F^2)$, and it can be shown than it must hold that $E = D^\dagger$ (the encoder must be the Moore-Penrose inverse of the decoder).

A linear autoencoder (an AE where ϕ is linear) is therefore almost equivalent to PCA [5], in that in the optimum, a bottleneck space of dimension k is spanned by the first k principle components of the input X. In general, an AE can be seen a PCA-like, but non-linear method for dimensionality resuction.

2.3 Variational Inference

Here we briefly explain the idea behind variational inference and introduce the ELBO which is the loss function we'll use throughout this text. For more details see Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*. Vol. 4. 4. Springer, 2006.

We have a set of observation $X = \{x_1, \dots, x_N\}$ which we try to explain by a probabilitic model. X is called the data or the observation (or something similar). We assume that the x_i's are i.i.d with some distribution function $p(x)$ and therefore for the entire dataset it holds that $p(X) = \prod x_i$.

We have some reason to believe that behind the scences there is some hidden (latent) shot caller $Z = \{z_1 \dots z_N\}$ that determines the fate of X. In other words we think that X is conditioned on Z and and we can speak of the joint distribution $p(X, Z) = p(X|Z)p(Z)$ and all said distributions factor over the individual samples multiplicatively, i.e. $p(X|Z) = \prod p(z_i|x_i)$.

Suppose that we have a fully Bayesian model. In this case there are no parameters because the parameters are themselves stochastic variables with some suitable priors. We can therefore pack all the latent variables and stochastic parameters into one latent "meta variable" $Z = (z_1, z_2, \dots)$, each z_i is some multidimensional r.v and possibly composed of several simpler r.vs (for example a categotical and a normal r.vs). We similarly pack all the observed variables into one meta variable X. Together we have a distribution $p(X, Z)$ and the working assumption is that it is easy to factorize $p(X, Z) = p(X|Z)p(Z)$, however $p(Z|X)$ is intractable and $p(X)$ is unknown.

We are being Bayesian here so we consider $X = (x_1, x_2, \dots)$ to really be constant, a set of observation and we want to best explain $p(X)$ by finding as high as possible lower boud (or rather to $\log p(X)$, the log evidence). A second goal is to approximate the intractable

$p(Z|X)$ by some simpler distribution $q(Z)$ taken from some family of distributions.

The following equation finds a lower bound (elbo) for the log evidence. (using Jensen's inequality)

$$\begin{aligned}\log p(X) &= \log \int p(X,Z)dZ = \log \int \frac{p(X,Z)}{q(Z)}q(Z)dZ \\ &= \log \int \frac{p(X,Z)}{q(Z)}dq(Z) \geq \int \log \frac{p(X,Z)}{q(Z)}dq(Z) \triangleq \mathcal{L}(q)\end{aligned} \tag{2.1}$$

In equation 2.1 we found a lower bound $\mathcal{L}(q)$ for the log evidence $\log p(X)$, which we call "evidence lower bound" or elbo for short. Whatever distribution q we put in elbo will not be greater than the real log evidence so we are looking for the q which maximizes it.

Now we show that maximizing actually obtains the log evidence and it is equivalent to minimizing $KL(q(Z)\|p(Z|X)$:

$$\begin{aligned}\mathcal{L}(q) &\triangleq \int \log \frac{p(X,Z)}{q(Z)}dq(Z) = \int \log \frac{p(Z|X)p(X)}{q(Z)}dq(Z) \\ &= \int \log p(X)dq(Z) - \int \log \frac{q(Z)}{p(Z|X)}dq(Z) = \log p(X) - KL(q(Z)\|p(Z|X)\end{aligned} \tag{2.2}$$

We can rewrite equation 2.2 as:

$$\log p(X) = \mathcal{L}(q) - KL(q(Z)\|p(Z|X)) \tag{2.3}$$

Equation 2.3 shows the the elbo minus the kl-divergence are constant and equal the log evidence. Therefore minimizing the kl-divergence (which is always non-negative) simultaneously maximizes the elbo and vicer-versa.

2.4 Adding parameters

Our models will not be fully Bayesian, but rather parametrized. In this case let θ represent the set of parameters for p, and ϕ the parameters for q. Meaning we are dealing with a family of distributions $p_\theta(x,z)$ and another family $q_\phi(z)$.

For any θ and any ϕ, the equations from the previous chapter hold also in the parametrize form, i.e $\log p_\theta(x) = \mathcal{L}(q_\phi) - KL(q_\phi(Z)\|p_\theta(Z|X)$.

We assume that we can only approach the "real" distribution using θ from below $\log p(X) \geq \log p_\theta(X)$. So together with equation 2.1 we have

$$(\forall\theta,\phi)\log p(X) \geq \log p_\theta(X) \geq \mathcal{L}(q_\phi) = \int \frac{p_\theta(Z|X)}{q_\phi(Z)}dq_\phi(Z) \tag{2.4}$$

So from equation 2.3 we again see that by finding the parameters ϕ, θ that maximize the elbo we approach the real log evidence as much as we can within the limits of the parametrized family of distributions we use.

2.4.1 Using NN for the parametrization

In this text we deal with variational autoencoders (VAE). A VAE is a neural network which is used to define and optimize the parameters ϕ and θ.

Specifically the encoder part of the network is a non-linear map $f_\theta(Z)$ which is used to define $P_\theta(X|Z)$. For example, we can assume that P_θ is a family of multivariate Gaussians and in this case $f_\theta(Z) = (\mu(Z), \Sigma(Z))$. Meaning the encoder maps Z to the location vector and covariance matrice. The parameter θ in this case are the weights of the encoder neural network. In parametrize the prior $p(Z)$ however in this case its parameter is not a function of X. In practice there is no reason to do this for most VAEs and we choose some simple fixed prior distribution for $p(Z)$.

The decoder network is similarly defined as a non-linear function $g_\phi(X)$ which maps X into the parameters defining the family $q_\phi(Z)$. Here too ϕ represent the weights of the decoder.

Remark 2.1. In many papers about VAEs (including this text) the decoder is described as $q_\phi(X|Z)$. This is an abuse of notation, because there is neither join distribution $q_\phi(X, Z)$ nor marginal $q_\phi(X)$ to speak of. However $q_\phi(Z)$ is meant to approximate the intractable $p(Z|X)$ and that's presumably the reason for the abused notation.

2.5 Graphical representation

It is both convenient as well as informative to include a graphical description of our probabilitic models by way of plate diagrams.

Please note that we drop the ϕ, θ subscript but they are still there in reality.

In a plate diagram nodes represent random variables and arrows represent dependency. Figure 2.1 is a plate diagram of the VAE model with slight adaptation. We use doted arrows to represent the arrows of the inference model, and regular arrows for the generative model. Regular (triangular) arrowhead represents real probabilitic dependency whereas rounded arrows are reminding us that this is not a real probabilitic dependency (recall 2.1) which maybe we can call 'parametric dependency'.

Plate represents packing of N i.i.ds since we have N observations $X = (x_i)_1^N$ and correspondingly N latent variables $Z = (z_i)$.

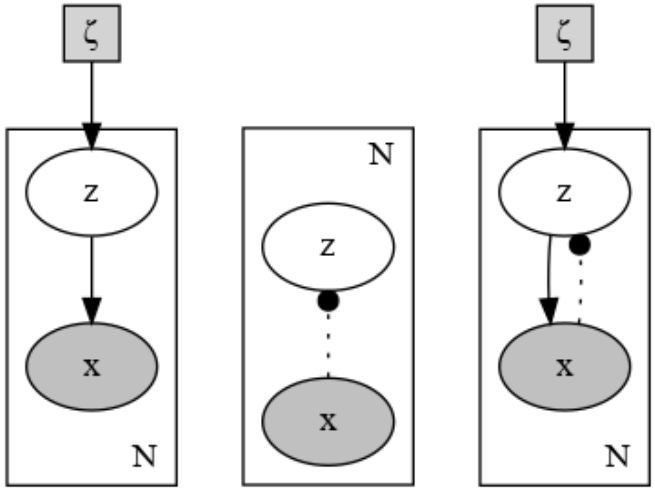
Shaded node represent known values (either observation of prior).

The squared ζ node represent some fixed parameters which describes the prior distribution of $P(z)$. Usually it is not shown in the papers about VAE but we just wanted to remind the reader that it can be parametrize in general.

The generative model therefore factors as: $p(x, z) = p(x|z)p(z|\zeta) = p(x|z)p(z)$

The inference model in this case is just $q(z)$ but we might denote is as $q(z|x)$ because it tries to approximate $p(z|x)$.

Note that the graphical model has no assumption about the specific types of distributions involved (Gaussian, Dirichlet or whateve ...) and that is left for the actual



(a) generative model ($p(x, z)$) (b) inference model ($q(z|x)$) (c) the combined graphical model

Figure 2.1: VAE graphical model

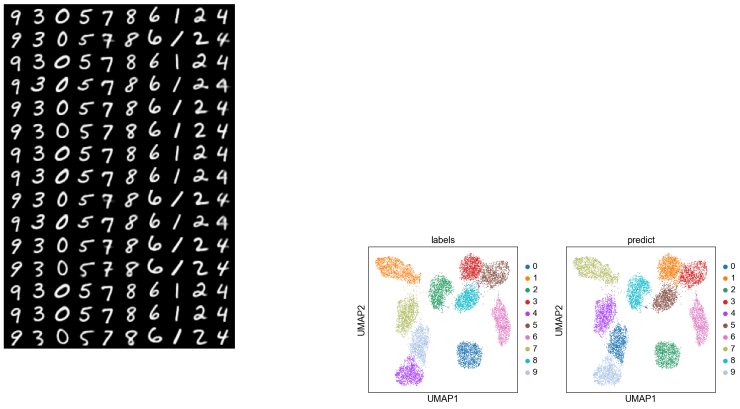
implementation.

In the case of a "vanilla" VAE, We chose the prior to be diagonal standard Gaussian $p(z) \sim N(; 0, 1)$. And $p(z|x)$ is then chosed as diagonal Gaussian (whose means and variances we find by training the network).

Chapter 3

Gaussian mixture model VAEs

Theoretical background and with some examples from publications and my own tests.



(a) (b)

Figure 3.1: a figure

Chapter 4

Datasets which were tested and results

some words about (sc)RNAseq and published papers where AE and VAE models have been applied. What we were hoping to achieve and compare with.

Chapter 5

Discussion, some remarks and conclusions

punkt. punkt. punkt.

Bibliography

[1] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*. Vol. 4. 4. Springer, 2006.

[2] Xifeng Guo et al. "Improved deep embedded clustering with local structure preservation." In: *Ijcai*. 2017, pp. 1753–1759.

[3] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[4] Mohammad Lotfollahi, F Alexander Wolf, and Fabian J Theis. "Generative modeling and latent space arithmetics predict single-cell perturbation response across cell types, studies and species". In: *bioRxiv* (2018), p. 478503.

[5] Elad Plaut. "From principal subspaces to principal components with linear autoencoders". In: *arXiv preprint arXiv:1804.10253* (2018).

[6] Denis Serre. "Matrices: Theory & Applications Additional exercises". In: *L'Ecole Normale Supérieure de Lyon* (2001).