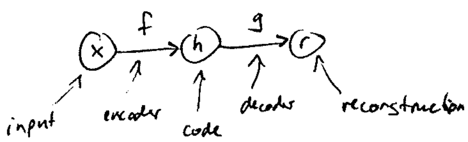
- An autoencoder is a neural network trained to attempt to copy its input to its output
- Internally it has hidden layer h that describes a <u>code</u> or <u>latent space</u> used to represent the input



Tries to set $g(f(x)) = x$ but are designed to be unable to learn to copy perfectly.

→ restricted to prevent exact copying

→ model forced to prioritize which aspects of the input should be copied, so often learns <u>most useful</u> properties of data

Not just deterministic functions f, g, but also stochastic mappings

$$P_{encoder}(h|x) \quad \text{and} \quad P_{decoder}(x|h)$$

<u>undercomplete autoencoders</u>

constrain h to have smaller dimension than x, to force to have preserved most salient properties.

→ reminiscent of data compression and very much a form of approximation theory.

Learning process is minimizing loss function

$$L(x, g(f(x)))$$

where L is loss function penalizing $g(f(x))$ for lack of fidelity with x

when $g(\cdot)$ fixed to be linear and L is mean-squared error, an undercomplete autoencoder recovers principal components analysis (PCA), the Karhunen-Loeve Transform.

why?

when allowing nonlinear (f,g), get generalization of PCA.

One can also regularize, e.g. with sparsity autoencoder

$$L(x, g(f(x))) + \Omega(h) \quad \text{where } \Omega(\cdot) \text{ is sparsity penalty on latent space } h$$

in addition to data fidelity term. Hopefully this forces learning of relevant things.

Denoising autoencoders

minimize $L(x, g(f(\hat{x})))$ instead of $L(x, g(f(x)))$

where $\hat{x}$ is noisy version of x.

DAE have to undo the noise corruption rather than simply copying, so again try to get most informative elements. [connection to universal denoising DUDE?]

Contractive autoencoders

different kind of regularization, to force a function that doesn't change much when x changes a little

$$L(x, g(f(x))) + \Omega(h, x) \quad \text{where } \Omega(h, x) = \lambda \sum_i \| \nabla_x h_i \|^2$$

or $\Omega(h) = \lambda \left\| \dfrac{\partial f(x)}{\partial x} \right\|_F^2$ where penalty is squared Frobenius norm (sum of squared elements)

of Jacobian matrix of partial derivatives associated with encoder function.

AE used for dimensionality reduction, etc. but what about generation?

$$x \xrightarrow{p_{encoder}(h|x)} h \xrightarrow{p_{decoder}(x|h)} r$$

Stochastic encoder/decoder.

Note that $p_{encoder}$, $p_{decoder}$ don't have to correspond to same valid joint distribution $p_{model}(x,h)$ in fact usually don't.

Back up to graphical models before getting to variational autoencoders (VAEs)

Consider conditional model $p_\theta(y|x)$ that approximates underlying conditional distribution, $p^*(y|x)$, a distribution over variables y conditioned on input x.

want to learn $p_\theta(y|x) \approx p^*(y|x)$.

→ graphical models have an interesting calculus, e.g. Forney-style factor graphs ⇔ block diagrams

Parameterizing conditional distributions with neural networks

- differentiable feedforward neural networks are a flexible, computationally-scalable kind of function approximator (universal approximation theorem)

Learning models with neural networks with many hidden layers is deep learning.

Notably NNs can be used to approximate pdfs and pmfs.

→ probabilistic models based on neural networks are computationally scalable since they allow stochastic gradient-based optimization and scaling to large models, large datasets

→ think of them as an operator $NN(\cdot)$

for example, neural net can parameterize categorical distribution $p_\theta(y|x)$ over a class label y, conditioned on image x as

$$p = NN(x)$$
$$p_\theta(y|x) = \text{categorical}(y; p)$$

Consider directed graphical models that have latent variables

→ latent variables are part of model but not observed directly and not part of dataset

denote by z

Joint distribution $p_\theta(x, z)$ considers observed variables and latent variables z.

A deep latent variable model (DLVM) denotes a latent variable model $p_\theta(x,z)$ whose distributional properties are parameterized by neural networks

Also conditional $p_\theta(x,z|y)$

Even when each factor (prior or conditional distribution) is relatively simple, marginal $p_\theta(x)$ can be very complex

A main difficulty of maximum likelihood learning in DLVMs is that marginal probability of data under model is typically intractable since $p_\theta(x) = \int p_\theta(x,z)dz$ may not have analytical solution or efficient estimator.

Due to intractability, we cannot differentiate w.r.t. its parameters and optimize, as we can with fully observed models. [Main difficulty is posterior $p_\theta(z|x)$]

There are approximate inference techniques but often computationally intense.

The framework of VAE provides a computationally efficient way of optimizing DLVMs jointly with corresponding inference model using stochastic gradient descent (SGD)

To turn DLVM's intractable learning problem into tractable problem, introduce a parametric inference model $q_\phi(z|x)$ which is an encoder (recognition model)

ϕ are parameters of inference model, called variational parameters

optimize variational parameters ϕ such that $q_\phi(z|x) \approx p_\theta(z|x)$.