Deep Learning

AlejandroMllo

This document serves as a very brief summary of the topics covered in each chapter of the book Deep Learning [1].

Disclaimer: This document is completely extracted from [1], the author does not attribute any ownership over the material.

14 Autoencoders

- NN trained to attempt to copy its input to its output. They are designed to be unable to learn to copy perfectly, so it prioritizes to learn useful properties of the data.
- Internally, it has a hidden layer $\boldsymbol{h}$ that describes a code used to represent the input. The network may be viewed as consisting of two parts: an encoder function $\boldsymbol{h} = f(\boldsymbol{x})$ and a decoder that produces a reconstruction $\boldsymbol{r} = g(\boldsymbol{h})$.
- Modern autoencoders have generalized to stochastic mappings $p_{encoder}(\boldsymbol{h}|\boldsymbol{x})$ and $p_{decoder}(\boldsymbol{x}|\boldsymbol{h})$. This means that both the encoder and the decoder are not simple functions but instead involve some noise injection.
- One way to obtain useful features from the autoencoder is to constraint $\boldsymbol{h}$ to have a smaller dimension than $\boldsymbol{x}$. This autoencoders are known as undercomplete.
- The learning process is described simply as minimizing a loss function $L(\boldsymbol{x}, g(f(\boldsymbol{x})))$.
- The autoencoder also fails to learn anything useful if the hidden code is allowed to have dimension greater than (overcomplete) or equal to the input.
- Regularized autoencoders let you train any architecture of autoencoder successfully, choosing the code dimension and the capacity of the encoder and decoder based on the complexity of the distribution to be modeled. Its loss function encourages the model to have other properties: sparsity of the representation, smallness of the derivative of the representation, and robustness to noise or to missing inputs.
- Nearly any generative model with latent variables and equipped with an inference procedure may be viewed as a particular form of autoencoder.
- A sparse autoencoder is an autoencoder whose training criterion involves a sparsity penalty $\Omega(h)$ on the code layer $\boldsymbol{h}$, in addition to the reconstruction error: $L(\boldsymbol{x}, g(f(\boldsymbol{x}))) + \Omega(h)$. They are typically used to learn features for another task, such as classification.
- Training an autoencoder is a way of approximately training a generative model.
- A denoising autoencoder (DAE) minimizes $L(\boldsymbol{x}, g(f(\tilde{\boldsymbol{x}})))$, where $\tilde{\boldsymbol{x}}$ is a copy of $\boldsymbol{x}$ that has been corrupted by some form of noise. DAEs must therefore undo this corruption rather than simply copying their input.
  - The training procedure introduces a corruption process $C(\tilde{\mathtt{x}}|\mathtt{x})$. The autoencoder then learns a reconstruction distribution $p_{reconstruct}(\tilde{\mathtt{x}}|\mathtt{x})$ estimated from training pairs $(\tilde{\boldsymbol{x}}|\boldsymbol{x})$ as follows:
    1. Sample a training example $\boldsymbol{x}$ from the training data.
    2. Sample a corrupted version $\tilde{\boldsymbol{x}}$ from $C(\tilde{\mathtt{x}}|\mathtt{x} = \boldsymbol{x})$.
    3. Use $(\tilde{\boldsymbol{x}}|\boldsymbol{x})$ as a training example for estimating the autoencoder reconstruction distribution $p_{reconstruct}(\boldsymbol{x}|\tilde{\boldsymbol{x}}) = p_{decoder}(\boldsymbol{x}|\boldsymbol{h})$ with $\boldsymbol{h}$ the output of encoder $f(\tilde{\boldsymbol{x}})$ and $p_{decoder}$ typically defined by a decoder $g(\boldsymbol{h})$.
- Another strategy for regularizing an autoencoder is to use a penalty Ω, as in sparse autoencoders, but with a different form of Ω: $\Omega(\boldsymbol{h}, \boldsymbol{x}) = \lambda \sum_i ||\nabla_{\boldsymbol{x}} h_i||^2$. This forces the model to learn a function that does not change much when $\boldsymbol{x}$ changes slightly, and to learn features that capture information about the training distribution.
  This are known as contractive autoencoders (CAE).
- Using deep encoders and decoders offers many advantages. Depth can exponentially reduce the computational cost of representing some functions and the amount of training data needed.

- Autoencoders are just feedforward networks.
- A general strategy for designing the output units and the loss function of a feedforward network is to define an output distribution $p(\boldsymbol{y}|\boldsymbol{x})$ and minimize the negative log-likelihood $-\log p(\boldsymbol{y}|\boldsymbol{x})$.
- In an autoencoder, $\boldsymbol{x}$ is now the target as well as the input.
- Score matching provides a consistent estimator of probability distributions based on encouraging the model to have the same score as the data distribution at every training point $\boldsymbol{x}$. In this context, it is a particular gradient field: $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})$.
- Like many other ML algorithms, autoencoders exploit the idea that data concentrates around a low-dimensional manifold or a small set of such manifolds. Autoencoders take this idea further and aim to learn the structure of the manifold.
- By making the reconstruction function sensitive to perturbations of the input around the data points, we cause the autoencoder to recover the manifold structure.
- Nonparametric manifold learning procedures build a nearest neighbor graph in which nodes represent training examples.
- The contractive autoencoder introduces an explicit regularizer on the code $\boldsymbol{h} = f(\boldsymbol{x})$, encouraging the derivatives of f to be as small as possible:

$$\Omega(\boldsymbol{h}) = \lambda \left\| \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\|_F^2 .$$

- Predictive sparse decomposition (PSD) is a model that is a hybrid of sparse coding and parametric autoencoders. The model consists of an encoder $f(\boldsymbol{x})$ and a decoder $g(\boldsymbol{h})$ that are both parametric. During training, $\boldsymbol{h}$ is controlled by minimizing: $\|\boldsymbol{x} - g(\boldsymbol{h})\|^2 + \lambda \mid \boldsymbol{h} \mid_1 + \gamma \|\boldsymbol{h} - f(\boldsymbol{x})\|^2$.
  PSD is an example of learned approximate inference.
- Autoencoders have been successfully applied to dimensionality reduction and information retrieval tasks.

References

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.