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.

20 Deep Generative Models

- We define the Boltzmann machine over a d-dimensional binary random vector $\mathbf{x} \in \{0,1\}^d$. It is an energy-based model, meaning we define the joint PD using an energy function: $P(\boldsymbol{x}) = \frac{\exp(-E(\boldsymbol{x}))}{Z}$, where $E(\boldsymbol{x})$ is the energy function, and Z is the partition function that ensures that $\sum_{\boldsymbol{x}} P(x) = 1$. The energy function and the Boltzmann machine is given by $E(x) = -\boldsymbol{x}^\top \boldsymbol{U}\boldsymbol{x} - \boldsymbol{b}^\top \boldsymbol{x}$, where $\boldsymbol{U}$ is the "weight" matrix of the model parameters and $\boldsymbol{b}$ is the vector of bias parameters.
- One interesting property of Boltzmann machines when trained with learning rules based on maximum likelihood is that the update for a particular weight connecting two units depends only on the statistics of those two units, collected under different distributions: $P_{model}(\boldsymbol{v})$ and $\hat{P}_{data}(\boldsymbol{v})P_{model}(\boldsymbol{h}|\boldsymbol{v})$.
- Restricted Boltzmann Machines (RBM) are undirected probabilistic graphical models containing a layer of observable variables and a single layer of latent variables. They can be stacked to form deeper models.
- Deep Belief Networks (DBN) are generative models with several layers of latent variables. The latent variables are typically binary, while the visible units may be binary or real. There are no intralayer connections. A DBN with l hidden layers contains l weight matrices: $\boldsymbol{W}^{(1)}, \dots, \boldsymbol{W}^{(l)}$. It also contains $l+1$ bias vectors $\boldsymbol{b}^{(0)}, \dots, \boldsymbol{b}^{(l)}$, with $\boldsymbol{b}^{(0)}$ providing the biases for the visible layer. It has undirected connections in the deepest layer and directed connections pointing downward between all other pairs of consecutive layers.
- Inference in a DBN is intractable because of the explaining away effect within each directed layer and the interaction between the two hidden layers that have undirected connections.
- A Deep Boltzmann Machine (DBM) is a generative model. It is an entirely undirected model, it has several layers of latent variables, and within each layer, each of the variables are mutually independent, conditioned on the variables in the neighboring layers.
  It is an energy-based model, meaning that the joint PD over the model variables is parametrized by an energy function E.
- Extremely high-dimensional inputs place great strain on the computation, memory and statistical requirements of ML models. Replacing matrix multiplication by discrete convolution with a small kernel is the standard way of solving these problems for inputs that have translation invariant spatial or temporal structure.
- Deep convolutional networks require a pooling operation so that the spatial size of each successive layer decreases.
- In the structured output scenario, we wish to train a model that can map from some input $\boldsymbol{x}$ to some output $\boldsymbol{y}$, and the different entries of $\boldsymbol{y}$ are related to each other and must obey some constraints.
- For sequence modeling, the model must estimate a PD over a sequence of variables, $p(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(\tau)})$. Conditional Boltzmann machines can represent factors of the form $p(\mathbf{x}^{(t)}|\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(\tau)})$ in order to accomplish this task.
- Traditional NN implement a deterministic transformation of some input variables $\boldsymbol{x}$. When developing generative models, we often wish to extend NN to implement stochastic transformations of $\boldsymbol{x}$. One straightforward way to do this is to augment the NN with extra inputs $\boldsymbol{z}$ that are sampled from some simple PD. The NN can then continue to perform deterministic computation internally, but the function $f(\boldsymbol{x}, \boldsymbol{z})$ will appear stochastic to an observer who does not have access to $\boldsymbol{z}$. Provided that f is continuous and differentiable, we can then compute the gradients necessary for training using back-propagation as usual.

- Sigmoid belief networks are a simple directed graphical model with a specific conditional PD. In general, we can think of it as having a vector of binary states $\boldsymbol{s}$, with each element of the state influenced by its ancestors: $p(s_i) = \sigma(\sum_{j<i} W_{j,i} s_j + b_i)$.
  Its most common structure is one that is divided into many layers, with ancestral sampling proceeding through a series of many hidden layers and then ultimately generating the visible layer.
- A differentiable generator network transforms samples of latent variables $\mathbf{z}$ to samples $\mathbf{x}$ or to distributions over samples $\mathbf{x}$ using a differentiable function $g(\boldsymbol{z}; \boldsymbol{\theta}^{(g)}$, which is typically represented by a NN. This model class includes variational autoencoders, GANs, and techniques that train generator networks in isolation.
- Generator networks are essentially just parametrized computational procedures for generating samples, where the architecture provides the family of possible distributions to sample from and the parameters select a distribution from within that family.
- Generative adversarial networks are based on a game theoretic scenario in which the generator network must compete against an adversary. The generator network directly produces samples $\boldsymbol{x} = g(\boldsymbol{z}; \boldsymbol{\theta}^{(g)}$. Its adversary, the discriminator network, attempts to distinguish between samples drawn from the training data and samples drawn from the generator. The discriminator emits a probability value given by $d(\boldsymbol{x}; \boldsymbol{\theta}^{(d)}$, indicating the probability that $\boldsymbol{x}$ is a real training example rather than a fake sample drawn from the model.
  The learning process requires neither approximate inference nor approximation of a partition function gradient.
- Generative moment matching networks are a form of generative model based on differentiable generator networks. They are trained with a technique called moment matching. The basic idea behind moment matching is to train the generator in such a way that many of the statistics of samples generated by the model are as similar as possible to those of the statistics of the examples in the training set. In this context, a moment is an expectation of different powers of a random variable.
- Auto-regressive networks are directed probabilistic models with no latent random variables. The conditional PD in these models are represented by NN.
- Linear auto-regressive networks: the simplest form of auto-regressive network has no hidden units and no sharing of parameters or features. Each $P(x_i|x_{i-1}, \ldots, x_1$ is parametrized as a linear model. If the variables are continuous, this model is merely another way to formulate a multivariate Gaussian distribution.
- Contractive autoencoders are designed to recover an estimate of the tangent plane of the data manifold. This means that repeated encoding and decoding with injected noise will induce a random walk along the surface of the manifold.
- Generalized denoising autoencoders are specified by a denoising distribution for sampling an estimate of the clean input given the corrupted input. The Markov chain that generates from the estimated distribution consists of:
  1. Starting from the previous step $\boldsymbol{x}$, inject corruption noise, sampling $\tilde{\boldsymbol{x}}$ from $C(\tilde{\boldsymbol{x}}|\boldsymbol{x})$.
  2. Encode $\tilde{\boldsymbol{x}}$ into $\boldsymbol{h} = f(\tilde{\boldsymbol{x}})$.
  3. Decode $\boldsymbol{h}$ to obtain the parameters $\boldsymbol{\omega} = g(\boldsymbol{h})$ of $p(\mathbf{x}|\boldsymbol{\omega} = g(\boldsymbol{h})) = p(\mathbf{x}|\tilde{\boldsymbol{x}})$.
  4. Sample the next state $\boldsymbol{x}$ from $p(\mathbf{x}|\boldsymbol{\omega} = g(\boldsymbol{h})) = p(\mathbf{x}|\tilde{\boldsymbol{x}})$.
- The walk-back training procedure accelerates the convergence of generative training of denoising autoencoders. Instead of performing a one-step encode-decode reconstruction, this procedure consists of alternative multiple stochastic encode-decode steps, initialized at a training example, and penalizing the last probabilistic reconstruction (or all the reconstructions along the way).
- Generative stochastic networks (GSN) are generalizations of denoising autoencoders that include latent variables $\mathbf{h}$ in the generative Markov chain, in addition to visible variables (usually denoted $\mathbf{x}$). A GSN is parametrized by two conditional PD that specify one step of the Markov chain:
  1. $p(\mathbf{x}^{(k)}|\mathbf{h}^{(k)})$ tells how to generate the next visible variable given the current latent state.
  2. $p(\mathbf{h}^{(k)}|\mathbf{h}^{(k-1)}, \mathbf{x}^{(k-1)})$ tells how to update the latent state variable, given the previous latent state and visible variable.
- Training generative models with hidden units is a powerful way to make models understand the world represented in the given training data. By learning a model $p_{model}(\boldsymbol{x})$ and a representation $p_{model}(\boldsymbol{h}|\boldsymbol{x})$, a generative model can provide answers to many inference problems about the relationships between input variables in $\boldsymbol{x}$ and can offer many different ways of representing $\boldsymbol{x}$ by taking expectations of $\boldsymbol{h}$ at different layers of the hierarchy.

References

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