Boltzmann Machine and Restricted Boltzmann Machine

- Boltzmann Machine general settings: Binary variables formed undirected graph, with Energy-Based Distribution, we got hidden/latent variables and visible/observed variables for different usage.
- Restricted Boltzmann Machine
  - Restricted: No inter-connection between inter-layers. which means that the units within one layer (hidden or visible) are conditionally independent.
  - Formal Representation:

$$P(v,h) = \frac{1}{Z}\exp(-E(v,h)),\quad E(v,h) = -b^T v - c^T h - v^T W h$$

  - Conditional Distributions:

$$P(h|v) = P(v,h)/P(v) = \frac{1}{P(v)}\cdot\frac{1}{Z}\exp\{b^T v + c^T h + v^T W h\} = \frac{1}{Z}\exp(c^T h + v^T W h)$$

For binary cases:

$$P(h_j=1|v) = \delta(c_j + v^T W_{:,j})$$

- Training RBM

we can learn explicit form of $P(h|v)$, and if we want to know $P(v)$, Sampling techniques can be used.

Deep Belief Networks

- General Structure

Top layer (Deepest Hidden Layers) is an RBM, and others (from hidden to visible) are directed.

- Generating Samples

First we sample first top layers (RBM) and then sampling data using ancestral sampling.

- Training phase: First training an RBM to capture distribution of visible variables

then adding hidden layers by training more RBMs for capture top hidden layers' distribution. Note that for training, we don't specify top layers must following a directed graph style.

- We always initialize a nets using weights of a trained deep belief network

Deep Boltzmann Machines

- Stacking many RBMs and for visible units, the value could be real or binary, for hidden layers, the value is binary. RBM means no connection within layer. This structure can be interpreted as two separate layers with conditionally independent. (odd number and even number independent)
- Interesting Properties

  No intra layer interaction makes it possible to use fix-point equations to maximize variational lower bound and find true optimal mean field expectations.
- DBM mean field inference

  Because here $P(h_1, h_2 | v) \neq P(h_1|v)\, P(h_2|v)$, we need to use variational inference method, to define a Q to approximate $P(H|V)$

  As other algorithms of learning manifold of training data, we can use mean-field variational inference to approximate $P(H|V)$ and consequently learning parameters W, to build up the whole DBM.
- Learning parameters of DBM

  we may confront partition function when we doing parameter updating, thus training a DBM involves variational inference of $\tilde{P}(h|v)$ and sampling methods.
- Layer-wise Pretraining

  Layer-wise pretraining reveals most power of DBM, thus layer-wise pretraining acts as a greedy method, starting from two-layer RBM and greedy training by

adding layers.

- Jointly Training Deep Boltzmann Machines
  - Centered deep Boltzmann Machine: Reparametrize the model by $\hat{v} = v - \mu$, creating a well conditioned Hessian.
  - MP-DBM: Change the target to obtain a true value of remaining units. adding this mechanism is equivelant to training end-to-end in deep learning context, but gradient direction and value is not a probalistic explanation.
  - Dropout used in MP-DBM: Deletion and Catent.

Boltzmann Machines for real data

Gray scale image can be used for modeling scalar of pixel values.

- Gaussian-Bernoulli RBMs.

  Most intuitively, $P(v|h)$ in RBM, we define v as a continuous variable then hidden units related to visible ones form a Gaussian Distribution together.

  Corresponding Energy function: we eliminate cross-term.

- Undirected Model of Conditional Covariance.

  When we considering Gaussian RBM for real values, we often care more about mean than variance, which means that we may lose information interacted with pixels (natural images thus not ok for this)

  - Mean and Covariance RBM

    Part of the model for mean and part of the model for covariance. unlike VAE, here we use energy function and explicit calculating inference.

  - Mean-product of Student t-distributions
  - Spike and slab Restricted Boltzmann Machines

    mean is given by $(h \odot s) W^T$ (h: binary, s: continuous, thus they together)

act as gating of different densities to decide mean and co-variance

Convolutional Boltzmann Machines

Solving image's input by using convolutions; main difficulties here is not matrix multiplication, but how we deal with pooling.

1. max-pooling: How we calculate pooling? we need compute k×k values.
2. Probalistic max pooling: Indicating which state is most possible by a R^{n+1} vector but fixed size, we cannot adjust pooling dynamically.
3. Padding: May cause fail on hidden units.

Boltzmann Machines for Structured or Sequential Outputs

RNN-RBM: Using an RNN for outputting weights by layers in RBM. and training means back-propagation through time.

Other Boltzmann Machines

Combining with some ideas of supervised learning.

Back-propagation through Random Operations

Idea: For generating samples from model, we need to add random factor on input.

Reparametric trick: $y \sim N(u, \delta^2)$, $y = u + \delta z$, $z \sim N(0,1)$

The z is independent from model, we can learn u and δ, using back propagation.

- Back-propagation through Discrete Stochastic Operations.

If the input is discrete, we can use $\mathbb{E}_y[J(y)]$ to estimate loss of different y. (Monte Carlo Methods)

REINFORCE Algorithm:

Estimating $\mathbb{E}_{p(y)}\left[\frac{\partial J(y)}{\partial w}\right]$ using monte carlo methods, also we can add $b^*(w)$ as bias in case unstable updating gradients.

Directed Generative Nets

- Sigmoid Belief Networks

  A vector of binary state s, with each element of the state influenced by its ancestors:

  $$P(s_i) = \delta\left(\sum_{j<i} W_{j,i} s_j + b_i\right)$$

  A sigmoid belief network have universal estimation property.

- Differentiable Generator Networks
  - Generator: A neural network which can map a random variable into a generative sample, $P(X|Z)$ can represent this operation.
  - Two ways of generating samples (by modeling different kind of distribution)
    - Modeling marginal distribution $P(X)$, describing all the density pattern of training data.
    - Modeling conditional distribution $P(X|Z)$, where we control generative samples by modifying latent variable Z.
    - For generative modeling's representative ability, it can reach the ability of supervised learning as long as we choose appropriate Z and $P(X|Z)$.
- Variational Auto Encoder
  - Compared with normal auto encoder:

    Auto encoder: Capture Z, which indicates manifold of X, $P(Z|X)$.

    VAE: Capture $q_\phi(Z|X)$, variational inference of Z, have generation ability.
  - Compared with Generative Adversial Network

    GAN: Generative model combining with a discrimator

    VAE: Generative model combining with a variational inference network
  - Analysis of generating blurry images: Maximum likelihood and MSE for maximizing the ELBO.

- Generative Adversial Networks
  - A possible problem of training GAN: Generative model and discriminative model may dominate the loss in turn.
  - In real usage, we sometimes change target to maximize the classification loss.
  - GAN can capture features/patterns which do not exist in training data.
- Generative Moment Matching Networks

  Assume that training data and samples generated by model are as similar as possible, the metrics of similarity is moment: $\mathbb{E}_x \prod_i x_i^{n_i}$
- Convolutional Generative Networks

  Transpose of convolutional operation $\tilde{x} = W^T \cdot w \cdot x$, involves design a up-pooling method to ensure information flow back.
- Auto-Regressive Networks

Constraining dependencies
- Auto-Regressive Networks: All the variables in sigmoid network are visible, we inference $P(x_i|x_j)$ by designing structure and calculating probability.
- Linear Auto-Regressive Networks: only stands for $P(x_i|x_j)\ \forall j<i$.

using graphical model
- Neural Auto-Regressive Networks

parameter sharing
- Can deal with discrete values by setting units corresponding to different dimensions of data.
  - Using hidden units for gathering information groups of x.
- Neural Auto-Regressive Networks Estimator: Sharing weights from W_{ij}.

Drawing Samples from Autoencoders

- Markov chain Associated with any denoising autoencoder.

$x \to C(\tilde{x}|x) \to h = f(\tilde{x}) \to w = g(h)$ Construct a markov chain iteratively sampling encoding and decoding.

$x = p(x|w)$

- Clamping and Conditional sampling: $p(X_t|X_0)$
- Well-Back Training Procedures

Generative Stochastic Network

$p(X^{(k)}|h^{(k)}) \rightarrow P(h^{(k)}|X^{(k-1)}, h^{(k-1)})$, in one step of Markov Chain. engaging $h^{(k-1)}$ and $X^{(k-1)}$ together to get $h^{(k)}$.

- Discriminative GSN: $P(y|X)$ for supervised learning

Other Sampling methods: Recovering diffusion process of a structured probalistic model.

Evaluating Generative models and conclusion

- Human cannot find some potential mistakes of generative model, though data seems to be reasonable.
- use model $p(h|X)$, indicating the mani-fold of data the motif of generative modelling