Literature Review

**Restricted Boltzmann Machine and Deep Belief Network: Tutorial and Survey**

* **Required Background/History:**
  + Probabilistic Graphical Models and Markov Random Field (2009)
    - A mixture of graph and probability theory which can represent a complex distribution in a possibly high-dimensional space on a graph
    - Random variables are represented by nodes/vertices
    - Probability between two interacting variables form edges
    - Two types of PGM:
      * Markov network (Markov Random Field): undirected edges
        + BM and RBM have undirected links
      * Bayesian network: directed edges
  + Gibbs Sampling (1984)
    - Use *d* conditional distributions to draw samples from a *d*-dimensional multivariate distribution P(X)
      * Assumes the conditional distributions of every dimension are simple to draw samples from when conditioned on the rest of coordinates
    - Algorithm is as follows
      * Start from random *d*-dimensional vector in range of data
      * Sample 1st dimension of 1st sample from the distribution of the 1st dimension conditioned on the other dimensions and continue for all dimensions
        + *j*-th dimension is
        + Do this for all dimensions until all dimensions of 1st sample are drawn
      * Repeat this process for dimensions of the 2nd sample and then perform this iteratively for all samples
    - BM and RBM use this for generating visible and hidden samples
  + Boltzmann (1868) / Gibbs (1902) Distribution
    - Probability that a given system is in a specific state based on the energy of that state and the temperature of the system:
      * , where is the canonical partition function used for normalization so that the probabilities sum to one
        + β ≔ , where kB is the Boltzmann constant and T is the temperature in Kelvin

This shows that absolute zero temperatures are extremely rare because as T -> 0, β -> ∞ so the probability of that state -> 0

* + - Free energy is the amount of internal energy in a system available to perform work. Internal energy is the total energy that a system contains.
      * Free energy: F ≔
      * Internal energy: U ≔ = =
    - Entropy is the measure of the disorder/randomness of a system. As stated by the 2nd law of thermodynamics, entropy of an isolated system always increases with time.
      * Entropy: H ≔ = = = using the above equations
  + [Ising Model](http://micro.stanford.edu/~caiwei/me334/Chap12_Ising_Model_v04.pdf) (1925):
    - Simplest explanation of ferromagnetism
      * particles have spin +1 or -1 where each interacts with its nearest neighbors in a chain (1D), lattice (2D), closed chain, or torus
      * Uses the Boltzmann distribution with energy given by:
        + E ≔ H = , where Jij is a coupling parameter dependent on the model, and the summation is over particles which interact with each other.
        + If for all i,j Jij ≥ 0 or Jij < 0, ferromagnetic or anti-ferromagnetic, respectively

If Jij can be both + and -, called spin glass

* + - * + Systems want lowest energy, so

Ferromagnetic: spins want to be parallel

Anti-ferromagnetic: spins want to be anti-parallel

* + - BM and RBM are Ising models with coupling parameters as weights learned using maximum likelihood estimation (MLE)
      * Therefore, we can say they are energy-based learning methods
  + [Hopfield Network](https://pubmed.ncbi.nlm.nih.gov/6953413/) (1982):
    - Uses Ising model for a neural network
      * Network of neurons with binary states {-1, 1}
      * Weights between neuron i and neuron j denoted wij
        + Hebbian Learning Rule (“Neurons that fire together, wire together. Neurons that fire out of sync, fail to link.”)

wij ≔

Very weak and ungeneralizable

* + - * Outputs of neurons determined for input if summed weighted inputs pass threshold θ:
        + xi ≔
      * Ising model uses energy from above, which is used in Boltzmann distribution
    - BM and RBM are Hopfield networks whose weights are learned through MLE instead of Hebbian learning
* **Restricted Boltzmann Machine:**
  + Structure
    - Boltzmann Machine is a generative model and PGM named after the distribution used in this model
      * Visible layer **v** = [v1, …, vd] ∈
        + Layer we can see such as data
      * Hidden layer **h** = [h­1, …, hd] ∈
        + Layer of latent variables

Meaningful features

Embeddings of visible data

* + - * Meaningful connection between the two despite possible different dimensions (d ≠ p)
    - PGM has connection links between **v** and **h**, bias for each element of **v** and **h**, as well as links between the elements in **v** and the elements in **h**
      * **W** = [wij] ∈
        + wij is link between vi and hj
        + Symmetric, i.e., wij = wji
      * **L** = [lij] ∈
        + lij is link between vi and vj
        + No from node to itself, so lii = 0 ∀i
      * **J** = [jij] ∈
        + jij is link between hi and hj
        + No from node to itself, so jii = 0 ∀i
      * **b** = [b1, …, bd] ∈
        + bi is the bias link for vi
      * **c** = [c1, …, cp] ∈
        + ci is the bias link for hi
    - Restricted Boltzmann Machine is a specific form of the BM
      * Restricts links to be **L = J = 0**
        + no links between the elements of **v** and no links between the elements of **h**
      * Yields energy from Ising model as:
        + E(v, h) ≔ based on interactions between linked units

Want to reduce this energy as per natural systems

Training the BM or RBM does so

* + - * Leads to joint Boltzmann distribution:
        + P =

Z is the partition function

Z ≔

* + Conditional Distributions
    - If given visible variables, hidden variables are conditionally independent and vice versa if given hidden variables
      * Only true for RBM, not BM, because no links within each layer
      * Proof involves Bayes rule and simplifications that show the joint distribution is a product of every distribution, but obtain:
        + P(hj, **v**) =
        + P(**h**, vi) =
  + Sampling variables
    - Gibbs Sampling
      * Generates hidden and visible units following algorithm:
        + Input the visible dataset **v**
        + Get initialization or do random initialization of v
        + Iteratively sample until burn-out convergence:

**for** *j from 1 to p* **do**

**for** *i from* 1 *to d* **do**

* + - * Generates observation and hidden units in both training and evaluation phases of RBM
      * Once trained, an RBM model can meaningfully represent the *d*-dimensional observation through generation of any amount of *p*-dimensional hidden variables by Gibbs sampling
      * RBM can also represent the *p*-dimensional hidden variables through generation of other *d*-dimensional observations
        + Above two show that BM and RBM are generative models
  + Training RBM by Maximum Likelihood Estimation
    - To generate hidden and visible units, need to learn weights of links **W**, **b**, and **c**
    - Let the *i*th visible data instance in a dataset of *n* visible vectors be denoted **v**i
      * Can denote the *j*th dimension of **v**i by **v***i,j*, **v***i* = [**v***i,*1, …, **v***i,d*]T
    - Log-likehood for visible data:
      * **=** 
        + Simplifies to:
    - Use MLE to find the θ ≔ {***W***,***b***,***c***} by derivative of log-likelihood:
      * + is the definition of expectation
        + Not possible to get a closed-form solution by setting derivative equal to 0, so learn by gradient ascent for MLE
        + Derivate w.r.t each parameter:

Define

* + - * + Again, no closed form solutions exist after setting these derivatives equal to zero, so we must find solutions iteratively using gradient descent with the above gradients.
        + is the conditional expectation based on data ***v****i*
        + is the joint expectation only about the RBM model
  + Contrastive Divergence
    - Exact computation of MLE difficult because summations over all possible values for both hidden and visible units
      * Need to approximate it instead. Use *contrastive divergence*!
        + Improves efficiency and reduces estimation variance
    - With Gibbs sampling that starts from an observation ***v***i, obtain a point
    - Compute expectation using only the one point
      * If wrong belief, we don’t want to generate observations by RBM using that belief
        + Called *negative sampling*

Trains iteratively but less ambitiously each iteration

By learning which outputs are wrong, the model learns how to generate correct observations

* + - Let be the sampled ***h*** to
      * Used to approximate joint expectation by Monte-Carlo evaluated at and for the *i*-th observation
    - Needs very few iterations of Gibbs sampling
    - The approximation for joint expectation leads to new gradient equations:
    - Gradients go to zero as Gibbs sampling approximations get closer to data/hidden variables
    - *INSERT ALGORITHM 2 PICTURE*
* **Deep Belief Network:**
  + Stacking RBM Models
    - RBM training can train neural networks, resulting in good weight initialization for training networks
      * Before ReLu and dropout methods were developed, good initialization of weights was required for backpropagation lest one runs into the problem of vanishing gradients
    - Neural networks have several layers
      * # of layers = *𝓁*
        + 1st layer gets input data
      * # of neurons in *𝓁*-th layer =
        + by convention
      * Consider every 2 successive layers as one RBM starting from the 1st pair of layers
        + Training data used as visible variable in the 1st pair of layers

Train weights and bias of this layer w/ algorithm 2

* + - * + Use Gibbs sampling from algorithm 1 to generate *n* -dimensional hidden variables

These hidden variables will be the visible variables for the second RBM

* + - * + Repeat this process until all layer pairs are trained

Greedy approach leads to good initial weights and biases for entire neural network

* + - * Use backpropagation to fine-tune weights and biases
    - Proposed network above referred to as the *Deep Belief Network* because it is pre-trained by RBM training (belief propagation), which avoids gradient descent due to good weight and bias initialization, allowing the network to be as deep as one wants.
    - DBN easily seen as a stack of RBM models
    - Training is unsupervised since RBM training is unsupervised
    - Fine-tuning can be supervised or unsupervised based on loss function
    - *SHOW ALGORITHM 3 STEPS*
  + Improvements over RBM and DBN
    - Convolutional DBN
    - Recurrent RBM to handle temporal information of data
    - Other energy-based models: Helmholtz machine for example
    - Deep Boltzmann Machines also exist with slight differences to DBN
      * Potential for document processing or face modeling
* Conclusion
  + History of underlying mechanisms for BM, RBM, and DBN
  + After background, covered structure of RBM, sample generation, and training
  + Lastly, DBN explained in detail