Our previous forays into unsupervised learning have been concerned with finding structure in unlabelled data (e.g. dimensional reduction, t-SNE, clustering).

Now we discuss another important flavor, generative modelling, which is concerned with learning the underlying probability distribution that describes some observed data set.

Generative Modelling

Consider the data set of observations $\vec{X} = \{\vec{x}_n\}_{n=1}^{N}$ where $\vec{x}_n \in \mathbb{R}^D, \mathbb{Z}_2^D, \ldots$

(could also be complex if our data set consists of coefficients of quantum wavefunctions). There exists an underlying probability distribution $P(\vec{x})$ that describes the data (i.e it is drawn from it) that we want to discover.

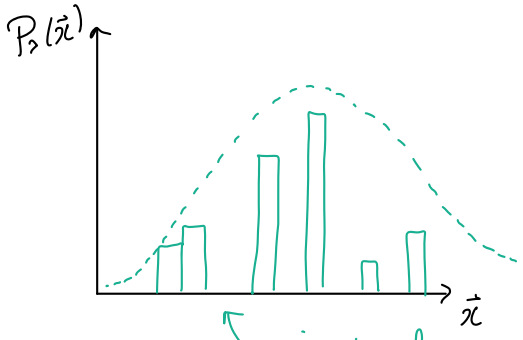
Goal: Introduce a parameterization (model) of the distribution $P_{\vec{\lambda}}(\vec{x})$ that depends on some parameters $\vec{\lambda}$ & find the parameters such that $P_{\vec{\lambda}}(\vec{x}) \simeq P(\vec{x})$.

$P_{\vec{\lambda}}(\vec{x})$: model dist. $P(\vec{x})$: data dist.

Naively with enough observations:

$$P_{\vec{\lambda}}(\vec{x}) = P(\vec{x}) = \frac{1}{N}\sum_{n=1}^{N} \delta_{\vec{x}_n, \vec{x}}$$

However unless N is much larger than the total number of possible observations, this will fail to generalize.



missing from $\vec{x}$, histogram predicts zero probability.

We want to learn a $P_\lambda(\vec{x})$ that:

i) Generalizes well

ii) Can be sampled to generate observations not seen in the data set.

Idea: Use a type of parameterization, energy based model that can be trained to minimize the Kullback-Liebler (KL) divergence

Energy Based Models

Let us restrict our set of observations to binary data, i.e. $\vec{x}_n \in \mathbb{Z}_2^D$. E-B models can then be thought of in terms of Ising Models that are a graphical representation of the probability distribution.

Quick history:

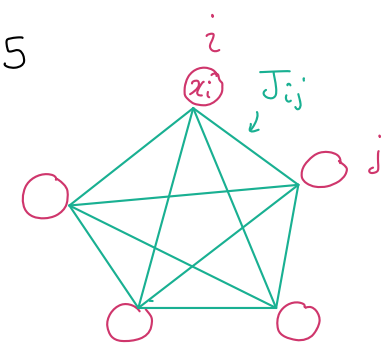
1982 Hopfield Network

couplings

$$E(\vec{x}) = -\sum_i a_i x_i - \frac{1}{2}\sum_{i,j} x_i J_{ij} x_j \qquad i, j \qquad x_i = \{0,1\}$$

configuration — magnetic field — spins

$D = 5$



Here our parameters are $\lambda = \{\vec{a}, J\}$

We take the probability as a Boltzman Distribution

$$P_\lambda(\vec{x}) = \frac{1}{Z} e^{-E(\vec{x})} \quad \text{with} \quad Z = \sum_{\vec{x}} e^{-E(\vec{x})} \qquad (k_B T = 1)$$

↑ sum over all 2^D configurations

i.e. $$\sum_{\vec{x}} \equiv \sum_{x_1=0}^{1} \cdot \sum_{x_2=0}^{1} \cdots \sum_{x_D=0}^{1}$$

$$= \prod_{i=1}^{D} \sum_{x_i=0}^{1}$$

Unfortunately the "all-to-all" coupling J_{ij} makes these models impossible to train. However there is a trick:

i) write $J_{ij} = \sum_k W_{ik} W_{kj}$ (always possible via SVD)

ii) Introduce auxilliary (latent) variables h_j ← hidden & perform a Hubbard-Stratonovich transformation, assuming they are continuous & normally distributed, this yields

$$P_\lambda(\vec{x}) = \frac{\int d\vec{h}\, e^{-E(\vec{x},\vec{h})}}{Z}$$

No connections within a layer!

$$E(\vec{x},\vec{h}) = -\sum_i a_i x_i + \frac{1}{2}\sum_j h_j^2 - \sum_{ij} x_i W_{ij} h_j$$

& define a new energy based model

$$p_\lambda(\vec{x},\vec{h}) \equiv \frac{e^{-E(\vec{x},\vec{h})}}{Z'} \quad ; \quad Z' = \sum_{\vec{x},\vec{h}} e^{-E(\vec{x},\vec{h})}$$

Restricted Boltzmann Machines (RBMs)

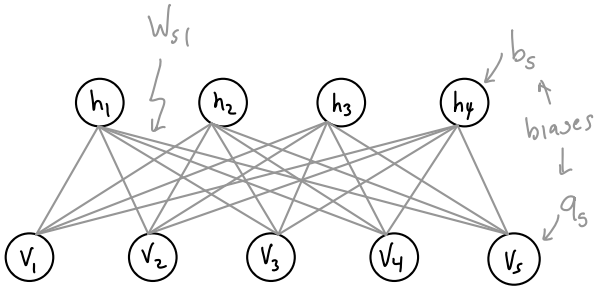
Introduced by Hinton in 1986, these represent a trainable class of energy-based models with an energy:

$$E(\vec{v},\vec{h}) = -\sum_{i=1}^{n_v} a_i v_i - \sum_{j=1}^{n_h} b_j h_j - \sum_{i,j} v_i W_{ij} h_j$$

$v_i \in \{0,1\}$

$h_j \in \{0,1\}$

visible units — hidden units — symmetric weight matrix



hidden layer $n_h = 4$

visible layer $n_v = 5$

* No connections inside a layer! This is what makes the model trainable!

As above the joint probability distribution has a Boltzmann form:

$$P_\lambda(\vec{v},\vec{h}) = \frac{e^{-E(\vec{h},\vec{v})}}{Z_\lambda} \quad ; \quad Z_\lambda = \sum_{\vec{h},\vec{v}} e^{-E(\vec{h},\vec{v})}$$

The quantity we are interested in is the marginal distribution

$$P_\lambda(\vec{v}) = \sum_{\vec{h}} P_\lambda(\vec{v},\vec{h})$$

by Bayes theorem this can be written in terms of a conditional distribution

$$P_\lambda(\vec{v}\,|\,\vec{h}) = \frac{P_\lambda(\vec{v},\vec{h})}{P_\lambda(\vec{h})}$$

($P_\lambda(\vec{v}\,|\,\vec{h})$: conditional; $P_\lambda(\vec{v},\vec{h})$: joint; $P_\lambda(\vec{h})$: marginal)

similarly: $P_\lambda(\vec{h}\,|\,\vec{v}) = \frac{P_\lambda(\vec{v},\vec{h})}{P_\lambda(\vec{v})}$

Before discussing how to train, let's dig deeper into the properties of our graphical model:

$$Z_\lambda P_\lambda(\vec{v}) = \sum_{\vec{h}} e^{-E(\vec{h},\vec{v})}$$

$$= \sum_{\vec{h}} e^{\sum_i a_i v_i + \sum_j b_j h_j + \sum_{ij} v_i w_{ij} h_j}$$

$$= e^{\sum_i a_i v_i} \left(\prod_{j=1}^{n_h} \sum_{h_j=0}^{1} \right) e^{\sum_j h_j \left(b_j + \sum_i v_i w_{ij} \right)}$$

($z_j = b_j + \sum_i v_i w_{ij}$)

$$= e^{\sum_i a_i v_i} \left(\prod_{j=1}^{n_h} \sum_{h_j=0}^{1} e^{h_j z_j} \right)$$

(only 2 values)

used $e^{\sum_j \alpha_j} = \prod_j e^{\alpha_j}$

$$= e^{\sum_i a_i v_i} \prod_{j=1}^{n_h} \left(1 + e^{z_j} \right)$$

* only possible because of restricted form!

Now we can examine the conditional distribution:

$$P_\lambda(\vec{h}\,|\,\vec{v}) = \frac{P_\lambda(\vec{v},\vec{h})}{P_\lambda(\vec{v})} = \frac{e^{-E(\vec{v},\vec{h})}}{Z_\lambda P_\lambda(\vec{v})} = \frac{e^{\sum_i a_i v_i + \sum_j b_j h_j + \sum_{ij} v_i w_{ij} h_j}}{e^{\sum_i a_i v_i} \prod_j \left(1 + e^{z_j} \right)}$$

($z_j = b_j + \sum_i v_i w_{ij}$)

$$= \frac{\prod_j e^{\left(b_j + \sum_i v_i w_{ij} \right) h_j}}{\prod_j \left(1 + e^{z_j} \right)} = \prod_j \frac{e^{z_j h_j}}{1 + e^{z_j}}$$

conditional distribution factorizes!

$$\Rightarrow \quad P_\lambda(\vec{h}|\vec{v}) = \prod_j P_\lambda(h_j|\vec{v}) \qquad \Rightarrow \quad h_j \text{ are iid!}$$

Now, recall there are only two possible values, $h_j \in \{0,1\}$

$$\therefore \quad P_\lambda(h_j=1|\vec{v}) = \frac{e^{z_j}}{1+e^{z_j}} = \frac{1}{1+e^{-z_j}} = \sigma(z_j)$$

(our familiar sigmoid!)

(normalized.)

$$\& \quad P_\lambda(h_j=0|\vec{v}) = \frac{1}{1+e^{z_j}} = 1-\sigma(z_j)$$

∴ we now have a mechanism to sample the hidden units based on a fixed visible configuration, i.e. Calculate $P_\lambda(h_j=1|\vec{v})$ & compare with a uniformly distributed $r \in U_{[0,1]}$

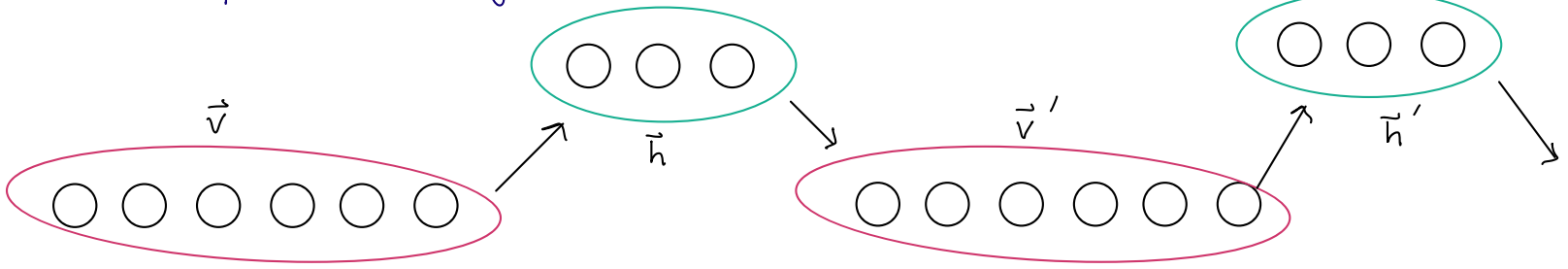
$h_j = r < P_\lambda$.

Also, looking at the form of the energy, we can see that making the exchange $a_i \to b_j$ & $v_i \to h_j$ (since $w_{ij} = w_{ji}$)

$$P_\lambda(v_i=1|\vec{h}) = \sigma(z_i') \quad \text{where} \quad z_i' = a_i + \sum_j w_{ij} h_j$$

$$P_\lambda(v_i=0|\vec{h}) = 1-\sigma(z_i')$$

This parameterization gives us the ability to sample both the visible & hidden layers with Markov Chain Monte Carlo:



After the MCMC has equilibrated, this gives us the ability to sample $P_\lambda(\vec{v})$ using Block Gibbs Sampling.

$$P_\lambda(\vec{h}\leftarrow\vec{v}) = P_\lambda(\vec{h}|\vec{v}) = \frac{P_\lambda(\vec{v},\vec{h})}{P_\lambda(\vec{v})}$$

$$P_\lambda(\vec{v}\leftarrow\vec{h}) = P_\lambda(\vec{v}|\vec{h}) = \frac{P_\lambda(\vec{v},\vec{h})}{P_\lambda(\vec{h})}$$

$$\frac{P_\lambda(\vec{h}\leftarrow\vec{v})}{P_\lambda(\vec{v}\leftarrow\vec{h})} = \frac{P_\lambda(\vec{v})}{P_\lambda(\vec{h})}$$

Satisfies detailed balance & stationary distributions are desired marginal ones!

Training & the KL Divergence

Now we can return to the problem of choosing parameters $\lambda = \{\vec{a}, \vec{b}, \overleftrightarrow{w}\}$ such that we can reproduce the data distribution $P(\vec{v})$ ← $\vec{v}$ is used instead of $\vec{x}$. using our training samples $\{\vec{v}_n\}_{n=1}^N$.

↑ no λ here!

We want to minimize a cost function given by the Kullback-Liebler divergence:

Trace over all configs.

$$D_{KL}(P||P_\lambda) = \sum_{\vec{v}} P(\vec{v}) \log \frac{P(\vec{v})}{P_\lambda(\vec{v})}$$

$D_{KL} \geq 0$

$D_{KL} = 0 \iff P = P_\lambda$

$$= \underbrace{\sum_{\vec{v}} P(\vec{v}) \log P(\vec{v})}_{\text{-entropy of } P \equiv -H} - \underbrace{\sum_{\vec{v}} P(\vec{v}) \log P_\lambda(\vec{v})}_{\text{depends on parameters "binary cross-entropy"}}$$

Unlike when training our DNN we can't just evaluate $P(\vec{v})$ (this is unknown) but we can approximate the trace via

sampling over our training data: $V \equiv \{\vec{v}_n\}_{n=1}^{N}$

$$D_{KL}(P \| P_\lambda) \simeq -H_V - \frac{1}{N}\sum_{\vec{v}\in V} \log P_\lambda(\vec{v})$$

only this part depends on parameters λ

$-H_V$: entropy of observations

$\frac{1}{N}\sum_{\vec{v}\in V} \log P_\lambda(\vec{v})$: log-likelihood from MLE for classification

$\therefore$ Take our cost function:

$$C = -\frac{1}{N}\sum_{\vec{v}\in V} \log P_\lambda(\vec{v})$$

$$= -\langle \log P_\lambda(\vec{v}) \rangle_{data}$$

Updating Parameters

With access to C we can update all model parameters $\lambda = \{\vec{a}, \vec{b}, \overleftrightarrow{w}\}$ via algorithms like gradient descent. We need to take gradients wrt parameters: $\lambda \leftarrow \lambda - \eta \nabla_\lambda C$

Ultimately we will perform averages over mini-batches & thus let us consider gradients evaluated at a fixed visible layer configuration $\vec{v}$ (i.e. a data point).

Recall: $P_\lambda(\vec{v}) = \sum_{\vec{h}} P_\lambda(\vec{v},\vec{h})$

so

$$\frac{\partial \log P_\lambda(\vec{v})}{\partial \lambda} = \frac{\partial}{\partial \lambda}\left[\log \sum_{\vec{h}} P_\lambda(\vec{v},\vec{h})\right]$$

$$= \frac{\partial}{\partial \lambda}\left[\log \sum_{\vec{h}} \frac{e^{-E(\vec{v},\vec{h})}}{Z_\lambda}\right]$$

$$\Rightarrow \frac{\partial}{\partial \lambda} \log P_\lambda(\vec{v}) = \frac{\partial}{\partial \lambda}\left[\log \sum_{\vec{h}} e^{-E(\vec{v},\vec{h})}\right] - \frac{\partial}{\partial \lambda}\left[\log \sum_{\vec{h},\vec{v}} e^{-E(\vec{v},\vec{h})}\right]$$

$$= \frac{-1}{\sum_{\vec{h}} e^{-E(\vec{v},\vec{h})}} \sum_{\vec{h}} e^{-E(\vec{v},\vec{h})} \frac{\partial E}{\partial \lambda} + \frac{1}{\sum_{\vec{h},\vec{v}} e^{-E(\vec{v},\vec{h})}} \sum_{\vec{h},\vec{v}} e^{-E(\vec{v},\vec{h})} \frac{\partial E}{\partial \lambda}$$

Now recall our expression for the conditional probabilities:

$$P_\lambda(\vec{h}\,|\,\vec{v}) = \frac{P_\lambda(\vec{v},\vec{h})}{P_\lambda(\vec{v})} = \frac{\frac{1}{Z_\lambda} e^{-E(\vec{v},\vec{h})}}{\frac{1}{Z_\lambda}\sum_{\vec{h}} e^{-E(\vec{v},\vec{h})}} = \frac{e^{-E(\vec{v},\vec{h})}}{\sum_{\vec{h}} e^{-E(\vec{v},\vec{h})}}$$

$$\Rightarrow \boxed{\frac{\partial}{\partial \lambda} \log P_\lambda(\vec{v}) = -\sum_{\vec{h}} P_\lambda(\vec{h}\,|\,\vec{v}) \frac{\partial E}{\partial \lambda} + \sum_{\vec{h},\vec{v}} P_\lambda(\vec{v},\vec{h}) \frac{\partial E}{\partial \lambda}}$$

& we can easily identify:

$$\boxed{\frac{\partial E}{\partial a_i} = -v_i} \quad ; \quad \boxed{\frac{\partial E}{\partial b_i} = -h_i} \quad ; \quad \boxed{\frac{\partial E}{\partial w_{ij}} = -v_i h_j}$$

as well as partially factorizing the last term using the probability product rule:

$$P_\lambda(\vec{v},\vec{h}) = P_\lambda(\vec{h}\,|\,\vec{v}) P_\lambda(\vec{v})$$

So putting everything together:

$$\frac{\partial C}{\partial \lambda} = -\sum_{\vec{v}} P(\vec{v}) \frac{\partial}{\partial \lambda} \log P_\lambda(\vec{v})$$

($P(\vec{v})$: data dist.)

$$\frac{\partial C}{\partial \lambda} = \sum_{\vec{v}}' \left[\sum_{\vec{h}} \frac{\partial E}{\partial \lambda} P_\lambda(\vec{h}|\vec{v}) \right] P(\vec{v}) - \sum_{\vec{v}}' \left[\sum_{\vec{h}} \frac{\partial E}{\partial \lambda} P_\lambda(\vec{h}|\vec{v}) \right] P_\lambda(\vec{v})$$

Easy! Draw a single training sample, generate $\{\vec{h}\}$ from $\{\vec{v}\}$ using Block Gibbs, average over all samples $\vec{v}$.

Hard! Need to average over the correct stationary distribution $P_\lambda(\vec{v})$ belonging to the RBM.

$$= \left\langle \frac{\partial E}{\partial \lambda} \right\rangle_{data} - \left\langle \frac{\partial E}{\partial \lambda} \right\rangle_{model}$$

The second part is difficult as we would need to run a long MCMC, very computationally expensive! Solution (due to Hinton) Contrastive Divergence (C.D.) → ignore converging the MC & take the average of $\frac{\partial E}{\partial \lambda}$ w.r.t. configurations after k steps, usually $k=1$.

In practice we use stochastic gradient descent & update parameters based on mini-batches.

Let's code it up!