SANTA CLARA UNIVERSITY

Quantum Machine Learning Seminars

Week 8: Quantum Neural Networks

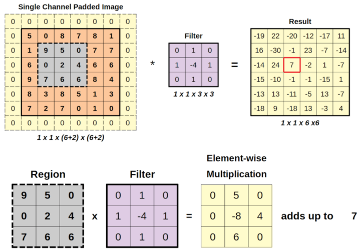
Dr. Hassan Alshal

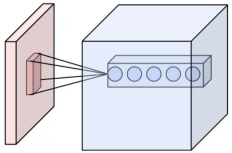
Outline

1. Neural Networks
2. Boltzmann Machine

Convolutional Neural Networks

- A CNN is organized as an alternating sequence of convolutional and pooling layers. Each stage produces an intermediate array, aka a feature map, that is a transformed version of the previous one.
- Each new feature-map entry is computed from a local neighborhood of the previous layer through a weighted linear combination:
$x_{ij}^{(\ell)} = \sum_{a,b=1}^{W} w_{a,b}\, x_{i+a,j+b}^{(\ell-1)}.$
- The coefficients $w_{a,b}$ form a kernel of size $W \times W$ to be applied across the image so the network extracts repeated local patterns rather than learning a separate rule pixel by pixel.



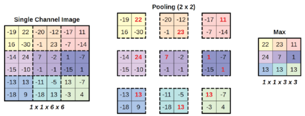


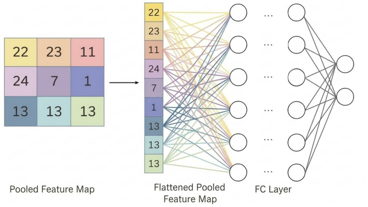
https://en.wikipedia.org/wiki/Convolutional_neural_network

Neural Networks

Convolutional Neural Networks

- A pooling layer reduces the spatial dimension of the feature map. It can be implemented by dividing the input into small patches and replacing each patch by its maximum value, thereby discarding less dominant entries.
- After sufficient convolution & pooling steps, the feature map becomes small enough that a fully connected layer can act on all remaining effective degrees of freedom and produce the final output. The learned objects are the kernel weights and the final classifier map, while the number of layers & the window size W are fixed through the training.

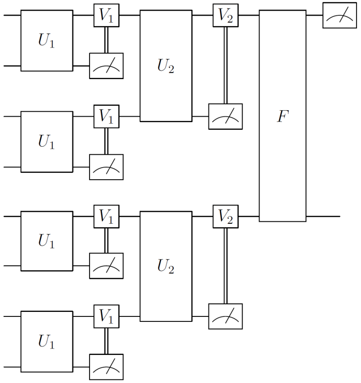




https://en.wikipedia.org/wiki/Convolutional_neural_network

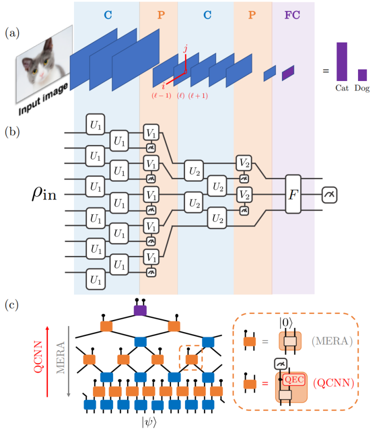
Quantum Convolutional Neural Networks

- In QCNN the classical input image is replaced by an unknown quantum state ρ_{in}, and convolutional layers are local few-qubit unitaries U_i. Then pooling layer measures a subset of qubits to conditionally applies nearby unitaries V_j, and a final FC layer applies a unitary F to the remaining qubits.
- The training objective is expressed through MSE. For labeled training data $\{(|\psi^\alpha\rangle, y_\alpha)\}_{\alpha=1}^{M}$, the loss is $\text{MSE} = \frac{1}{2M}\sum_{\alpha=1}^{M}\left(y_\alpha - f_{U_i,V_j,F}(|\psi^\alpha\rangle)\right)^2$, where $f_{U_i,V_j,F}(|\psi^\alpha\rangle)$ is the expected QCNN output on the input state $|\psi^\alpha\rangle$.



Quantum Convolutional Neural Networks

- MERA is a tensor-network/state-preparation framework built from hierarchical layers of unitaries, efficiently represents many-body states/correlations.
- QCNN has essentially the same circuit skeleton as a MERA, but run in the reverse direction; therefore, if a state $|\psi\rangle$ has a MERA representation, QCNN recognizes it deterministically.
- Pooling layers measurements act as error syndromes, and conditional V_j gates act as recovery operations, allowing QCNN to encode/decode error correction methods, with $\mathcal{O}(\log N)$ of trainable parameters.

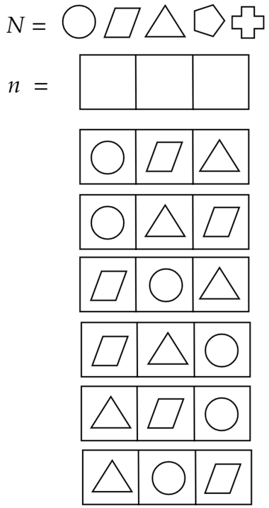


Cong et al., arXiv: 1810.03787

Boltzmann Machine

Entropy

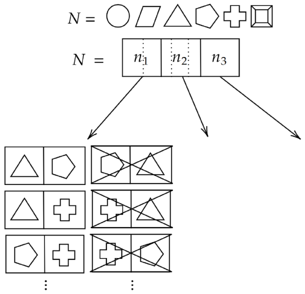
- Entropy is a representation of the number of indistinguishable ways that the particles can be distributed over the system energies. These arrangements are called a microstates. The higher the number of microstates, the higher the entropy.
- The number of possible distinguishable distributions is given by
$$^{N}P_n = \frac{N!}{(N-n)!} = N \times (N-1) \times \cdots \times (N-n+1).$$
- And for indistinguishable distributions we have
$$^{N}C_n = \frac{^{N}P_n}{n!} = \frac{N!}{(N-n)!} = \frac{N!}{(N-n) \times !n!}.$$



Boltzmann Machine

Entropy

- If a box can be filled with n_1 objects, then the indistinguishable distributions are $^{N}C_{n_1} = \frac{N!}{(N-n_1)! \times n_1!}$
- the next box can be filled with n_2 objects, i.e., the indistinguishable distributions are $^{N-n_1}C_{n_2} = \frac{N-n_1!}{(N-n_1-n_2)! \times n_2!}$



Boltzmann Machine

Entropy

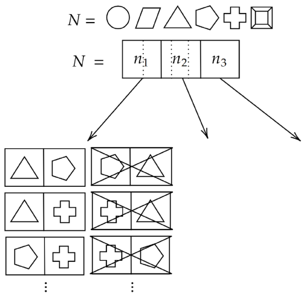
- Generalize the previous remarks to get

$$
\begin{aligned}
W &= {}^{N}C_{n_1} \times {}^{N-n_1}C_{n_2} \times {}^{N-n_1-n_2}C_{n_3} \times \cdots \\
&= \frac{N!}{(N-n_1)!n_1!} \times \frac{(N-n_1)!}{(N-n_1-n_2)!n_2!} \times \frac{(N-n_1-n_2)!}{(N-n_1-n_2-n_3)!n_3!} \times \cdots \\
&= \frac{N!}{n_1!n_2!n_3!\cdots} = \frac{N!}{\prod_i n_i!},
\end{aligned}
$$

  where $\sum_i n_i = n_1 + n_2 + n_3 + \cdots = N$.

- Boltzmann equation $\boxed{S_B = k_B \ln(W)}$.
- Why the ln function?

  Because entropy needs to be an extensive property.



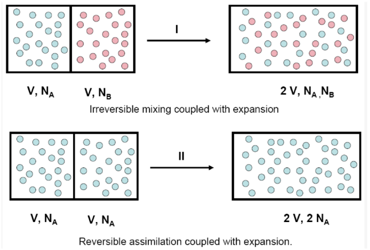
Boltzmann Machine

Entropy

- Gibbs entropy:

$$S_G = \frac{S_B}{N} = \frac{k_B}{N}\ln\left(\frac{N!}{\prod_i n_i!}\right)$$
$$= \frac{k_B}{N}\left[\ln(N!) - \sum_i \ln(n_i!)\right]$$
$$= \frac{k_B}{N}\left[N\ln(N) - N - \sum_i \left(n_i \ln\left(\frac{n_i N}{N}\right) - n_i\right)\right]$$
$$= -k_B \sum_i \frac{n_i}{N}\ln\left(\frac{n_i}{N}\right)$$

$$W \sim V^N \Rightarrow S_B = Nk_B\ln(V) + \cdots$$
$$\Delta S = 2Nk_B\ln 2$$


I
V, N_A
V, N_B
2 V, N_A, N_B
Irreversible mixing coupled with expansion
II
V, N_A
V, N_A
2 V, 2 N_A
Reversible assimilation coupled with expansion.

A. Ben-Naim, Entropy 2007, 9(3), 132-136

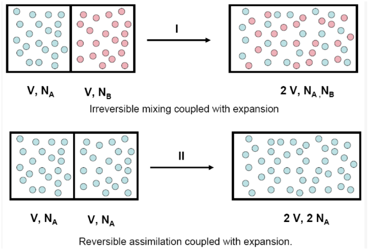
- If $p_i = \frac{n_i}{N}$, then $\boxed{S_G = -k_B \sum_i p_i \ln(p_i)}$.

Boltzmann Machine

Entropy

- $S_G = -k_B \sum_i p_i \ln(p_i)$.
- $U = \sum_i p_i \epsilon_i$.
- As $F = U - TS$ and $dQ = TdS = dU + PdV$, then rearrange and maximize $dS/dp_i = 0$ such that: $p_i = e^{(\alpha_V - k_B T - \epsilon_i)/(k_B T)}$.
- $\sum_i p_i = e^{(\alpha_V/k_B T - 1)} \sum_i e^{-\epsilon_i/k_B T} = 1$.
- $Z = e^{(1-\alpha_V/k_B T)} = \sum_i e^{-\epsilon_i/(k_B T}$.
- $\boxed{p_i = \dfrac{e^{-\epsilon_i/k_B T}}{Z}}$ & $\boxed{F = -k_B T \ln(Z)}$.

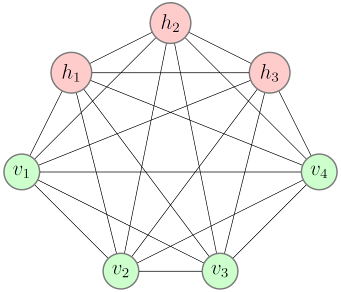
$$W \sim V^N \Rightarrow S_B = N k_B \ln(V) + \cdots$$
$$\Delta S = 2 N k_B \ln 2$$


I
V, N_A
V, N_B
2 V, N_A, N_B
Irreversible mixing coupled with expansion
II
V, N_A
V, N_A
2 V, 2 N_A
Reversible assimilation coupled with expansion.

A. Ben-Naim, Entropy 2007, 9(3), 132-136

Entropy

- Boltzmann machine is a stochastic energy-based neural network with binary units $s_i \in \{\pm 1\}$.
- $E(s) = -\sum_i b_i s_i - \sum_{i<j} w_{ij} s_i s_j$ with p_i and Z as defined before.
- $p(s_i = 1) = \sigma\left(b_i + \sum_j w_{ij} s_j\right)$, where $\sigma(x) = \dfrac{1}{1+e^{-x}}$.
- Extremize $\Delta w_{ij} \propto \langle s_i s_j \rangle_{\text{data}} - \langle s_i s_j \rangle_{\text{model}}$(How?)
- $\mathcal{L}(p^d, p) = -\sum\limits_i p_i^d \ln p_i \Rightarrow \Delta w = -\eta \frac{\partial \mathcal{L}}{\partial w}$.
- $p(v) = \text{tr}(\Lambda_v \rho) \qquad \text{with} \qquad \Lambda_v := |v\rangle\langle v| \otimes \mathbb{I}_h$.
- $\mathcal{L} = -\sum_v p^d(v) \ln \left[\dfrac{\text{tr}\left(\Lambda_v e^{-E}\right)}{\text{tr}(e^{-E})} \right]$.



Thank You!