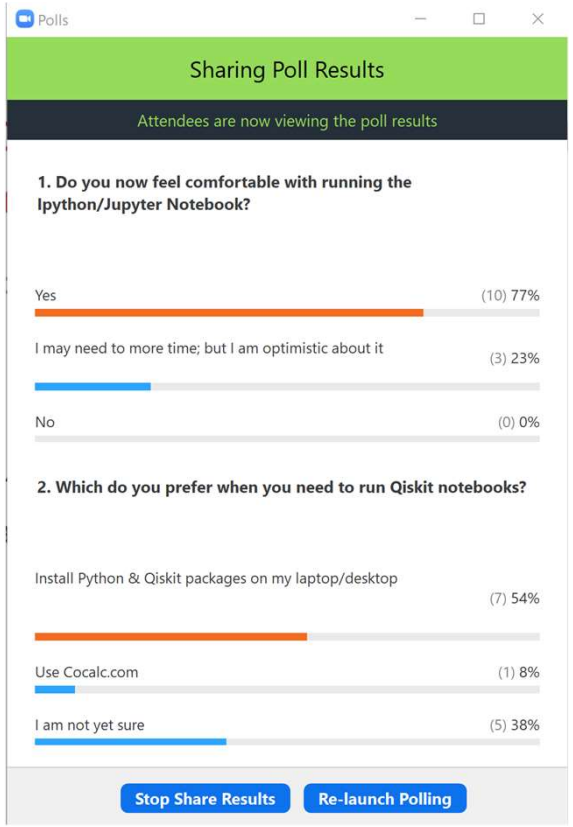
PHY682 Special Topics in Solid-State Physics: Quantum Information Science

Lecture time: 2:40-4:00PM Monday & Wednesday

Today 9/23:

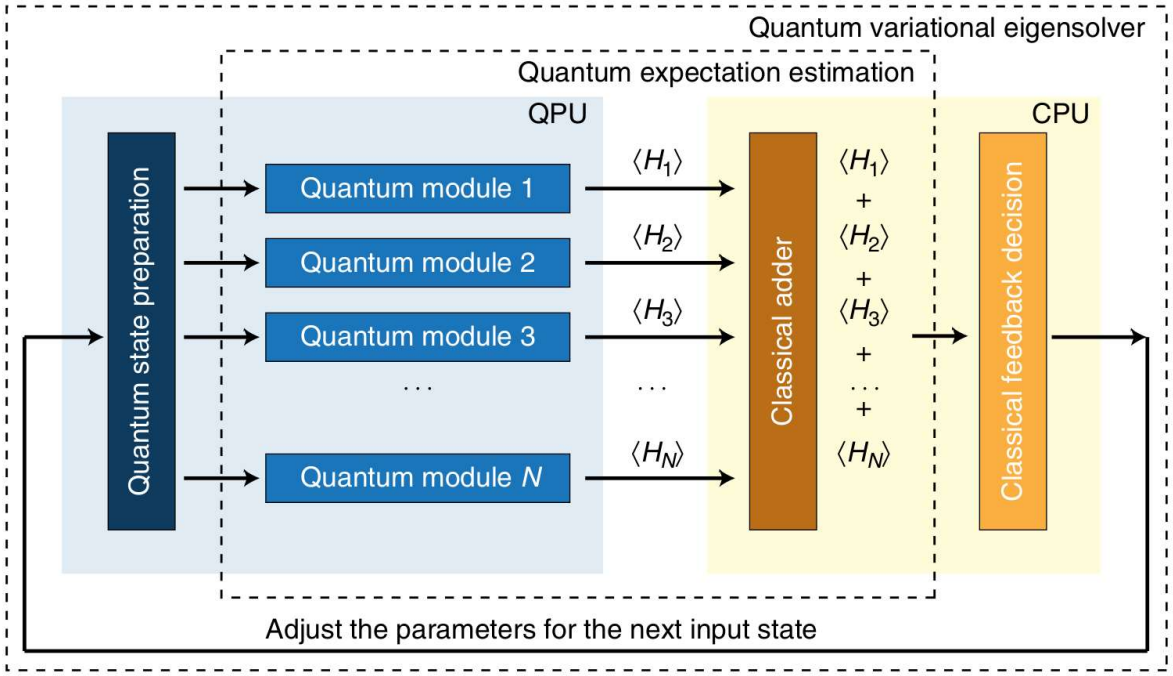
More on Week 5's topics: VQE, QAOA, Hybrid Q-Classical Neural Network, Application to Molecules



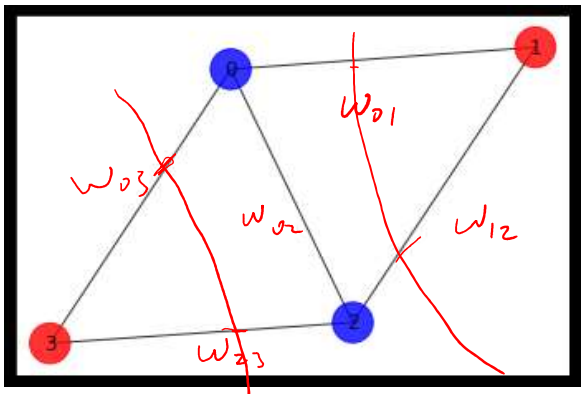
Summary of VQE

[Peruzzo et al. Nat. Comm. 5, 4213 (2014)]

$$H = \sum_{i=1}^{N} c_i H_i \qquad \text{e.g. } H = -B_x\sigma_x - B_z\sigma_z$$



MaxCut



1. Map MaxCut problem to Ising Hamiltonian

$$\tilde{C}(\mathbf{x}) = \sum_{i,j} w_{ij} x_i (1 - x_j).$$

$x_i = 0, 1$

- x: binary 0/1 → x = (1-z)/2, with z=1 and -1

[Generalize to allow local terms]

turn binary into +1/-1 ⇒ σ_z

Q. operator

$$C(\mathbf{Z}) = \sum_{i,j} \frac{w_{ij}}{4}(1-Z_i)(1+Z_j) + \sum_i \frac{w_i}{2}(1-Z_i) = -\frac{1}{2}\left(\sum_{i<j} w_{ij} Z_i Z_j + \sum_i w_i Z_i\right) + \text{const}$$

$$\text{Minimize the Hamiltonian}: H = \sum_i w_i Z_i + \sum_{i<j} w_{ij} Z_i Z_j.$$

2. Approximate universal quantum computing for optimization (use variational wavefunction to minimize Hamiltonian)

$$|\psi(\theta)\rangle = [U_{\text{single}}(\boldsymbol{\theta}) U_{\text{entangler}}]^m |+\rangle$$

$+ \cdots +$

$$C(\boldsymbol{\theta}) = \langle \psi(\boldsymbol{\theta}) | H | \psi(\boldsymbol{\theta}) \rangle$$

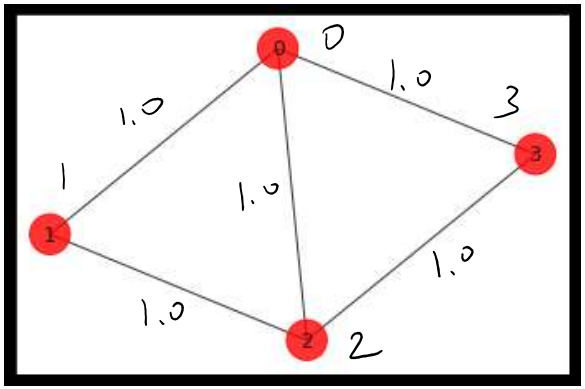
evaluated by QC.

entgl

Do Notebook

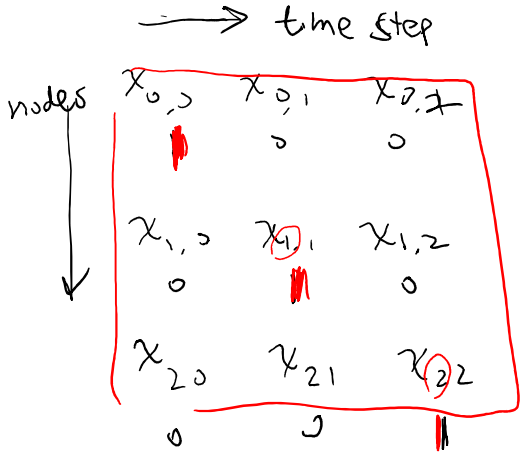
The Graph (used in the notebook demo)

```
# Generating a graph of 4 nodes

n=4 # Number of nodes in graph
G=nx.Graph()
G.add_nodes_from(np.arange(0,n,1))
elist=[(0,1,1.0),(0,2,1.0),(0,3,1.0),(1,2,1.0),(2,3,1.0)]
# tuple is (i,j,weight) where (i,j) is the edge
G.add_weighted_edges_from(elist)

colors = ['r' for node in G.nodes()]
pos = nx.spring_layout(G)
default_axes = plt.axes(frameon=True)
nx.draw_networkx(G, node_color=colors, node_size=600,
         alpha=.8, ax=default_axes, pos=pos)
```



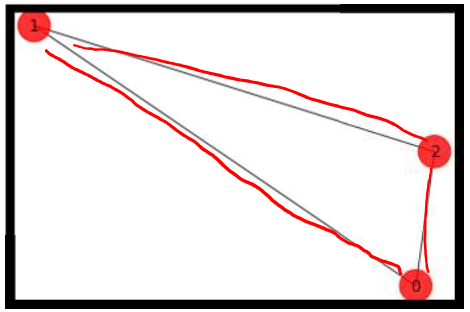
Traveling Salesman Problem

Find the shortest Hamiltonian cycle in a graph $G = (V, E)$ with $n = |V|$ nodes and distances, w_{ij} (distance from vertex i to vertex j). A Hamiltonian cycle is described by n^2 variables $x_{i,p}$, where i represents the node and p represents its order in a prospective cycle. The decision variable takes the value 1 if the solution occurs at node i at time order p. We require that every node can only appear once in the cycle, and for each time a node has to occur. This amounts to the two constraints (here and in the following, whenever not specified, the summands run over 0,1,...,n-1)



$$\sum_i x_{i,p} = 1 \quad \forall p$$

$$\sum_p x_{i,p} = 1 \quad \forall i.$$

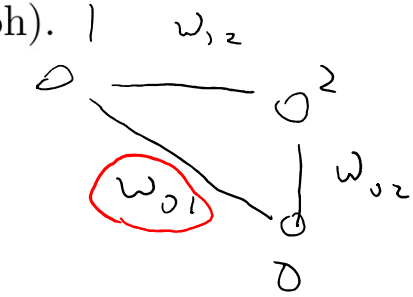


Do Notebook

Traveling Salesman Problem

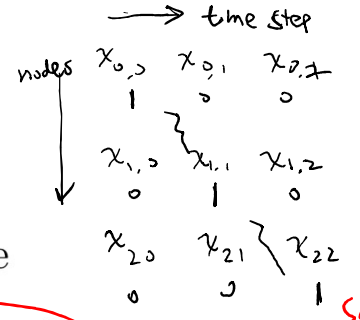
For nodes in our prospective ordering, if $x_{i,p}$ and $x_{j,p+1}$ are both 1, then there should be an energy penalty if $(i,j) \notin E$ (not connected in the graph). The form of this penalty is

$$\sum_{i,j \notin E} \sum_{p} x_{i,p} x_{j,p+1} > 0,$$



where it is assumed the boundary condition of the Hamiltonian cycles $(p = n) \equiv (p = 0)$. However, here it will be assumed a fully connected graph and not include this term. The distance that needs to be minimized is

$$C(\mathbf{x}) = \sum_{i,j} w_{ij} \sum_{p} x_{i,p} x_{j,p+1}$$



Putting everything in a single objective function to be minimized, we have

$$C(\mathbf{x}) = \sum_{i,j} w_{ij} \sum_{p} x_{i,p} x_{j,p+1} + A \sum_{p} \left(1 - \sum_{i} x_{i,p}\right)^2 + A \sum_{i} \left(1 - \sum_{p} x_{i,p}\right)^2 .$$

Qiskit implementation

https://quantum-computing.ibm.com/jupyter/tutorial/

(1) MaxCut and (2) Traveling Salesman Problem (TSP)

https://quantum-computing.ibm.com/jupyter/tutorial/advanced/aqua/optimization/max_cut_and_tsp.ipynb

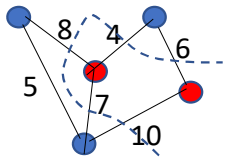
Note the newest notebook works on qiskit 0.20.x but NOT on qiskit 0.19.x
➔ I had to modify codes for them to work on the latter

Do Notebook

Quantum approximate optimization algorithm (QAOA)

[Farhi et al. arXiv:1411.4028]

- The goal is to optimize a classical function, in particular, combinatorial problems such as (weighted) MAXCUT and MAX-3SAT: $C_{\max} = \max_{\vec{x}} C(\vec{x})$



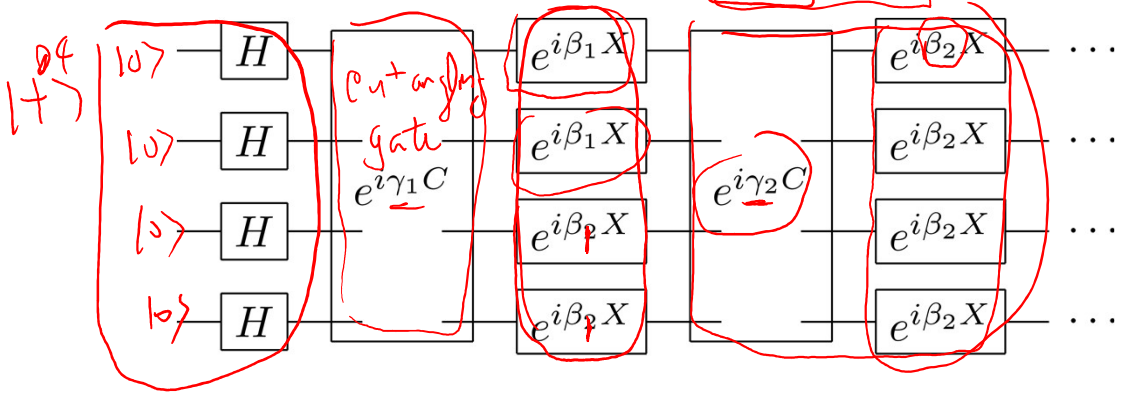
$$(x_1 \vee \bar{x}_2 \vee \bar{x}_4) \wedge (x_2 \vee x_3 \vee x_5) \vee (x_1 \vee x_5 \vee \bar{x}_6)$$

- It uses a specific *variational form (p layers)*:

$$|\psi_p(\vec{\gamma}, \vec{\beta})\rangle = e^{i\beta_p H_x} e^{i\gamma_p C} \cdots e^{i\beta_1 H_x} e^{i\gamma_1 C} |+\rangle^{\otimes n}$$

$$C = \sum_{\vec{z}} c_{\vec{z}} |\vec{z}\rangle\langle\vec{z}|$$

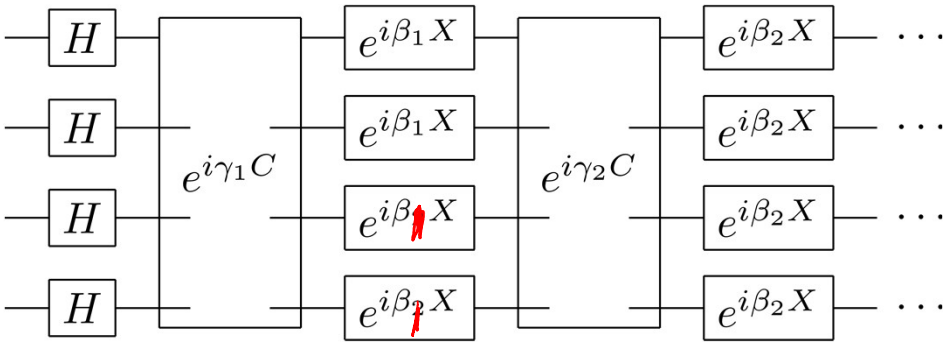
$$H_x = \sum_{j=1}^{n} \sigma_x^j$$



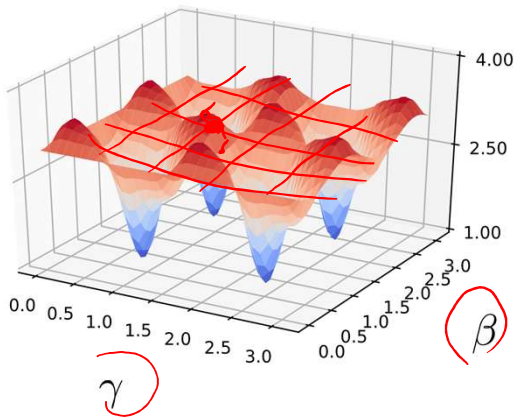
$$\max_{\vec{\gamma},\vec{\beta}} \sum_{\vec{z}} c_{\vec{z}} |\langle\vec{z}|\psi(\vec{\gamma},\vec{\beta})\rangle|^2$$

➔ Use a simple grid search

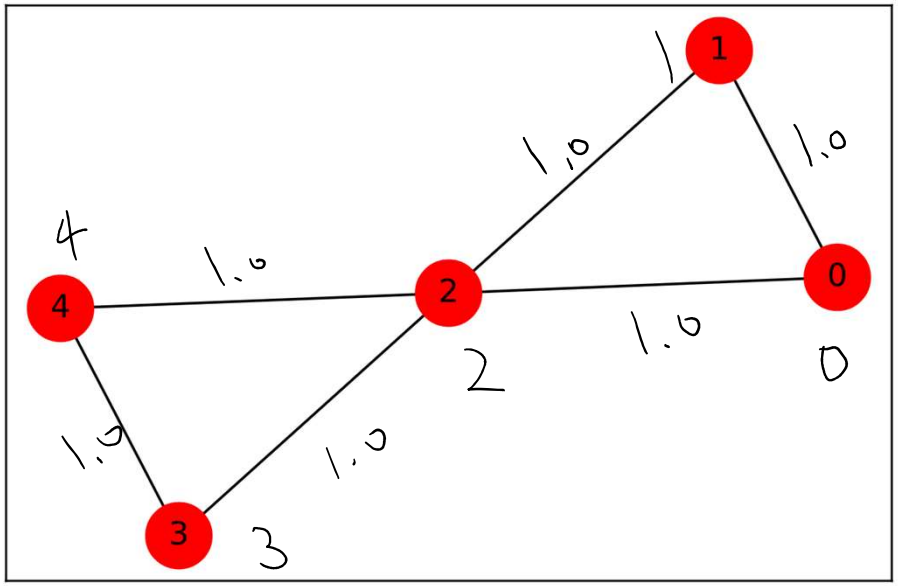
QAOA: parameter optimization



- Optimal parameters β's and γ's are sometimes found using grid search (evaluating all grid points and identify the optimal ones)
- Optimal parameters β's and γ's can also be found iteratively using classical optimization (such as gradient descent, Nelder-Mead, etc.)



Qiskit implementation



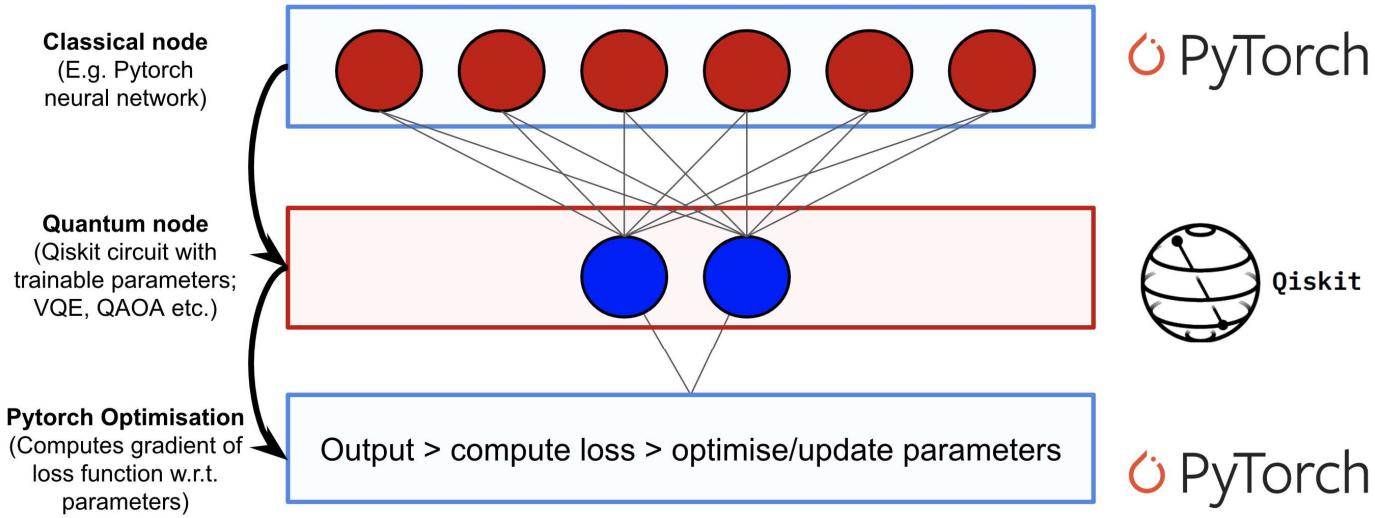
Do Notebook

Hybrid classical-quantum neural network*

- Classical Machine Learning has become an important tool
- Quantum Machine Learning has been a recent research hot topic

➔ Hybrid classical-quantum approach is already implementable

Use ‘Pytorch’ (useful machine learning package) and ‘Qiskit’

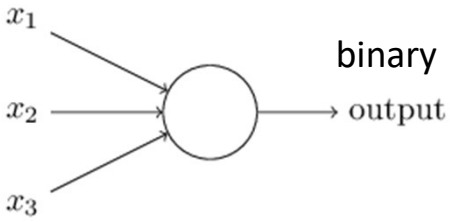


https://qiskit.org/textbook/ch-machine-learning/machine-learning-qiskit-pytorch.html

Quick review of classical neural network

- Perceptron:

binary

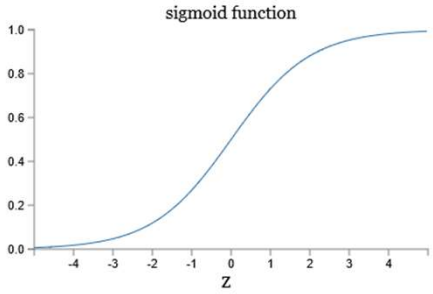


binary

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \equiv -b \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \equiv -b \end{cases}$$

Sigmoid function
(like Fermi-Dirac distribution)

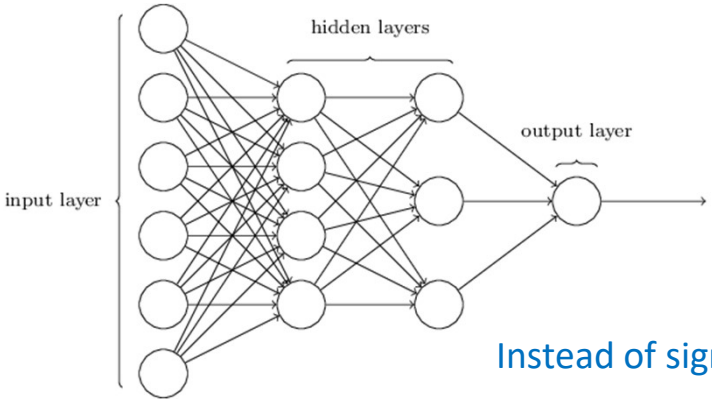
$$\sigma(z) \equiv \frac{1}{1+e^{-z}}$$



Sigmoid neuron:
➔ output is (not binary)

$$\sigma(z = x \cdot w + b)$$

- FCN (fully-connected network)



Supervised learning: some training data to optimize some cost function to obtain the optimal weights w and biases b

Instead of sigmoid function, can also use the recliniear function: $f_{\text{RL}}(z) \equiv \max(0, z)$

❑ Use gradient descent for optimization

$$\Delta C(v) = \nabla C \cdot \Delta v, \quad \text{choose } \Delta v = -\eta \nabla C \text{ to update } v \; : v \to v' = v - \eta \nabla C.$$

C is cost function, eta is the "learning rate", v can be weights w or biases b

$$w_k \quad \to \quad w_k' = w_k - \eta \frac{\partial C}{\partial w_k}$$

$$b_l \quad \to \quad b_l' = b_l - \eta \frac{\partial C}{\partial b_l}.$$

❑ *Stochastic* gradient descent

Ideas: a training epoch uses n training sets of data too large

➔ break down to smaller mini-batches, each with m sets

$$w_k \quad \to \quad w_k' = w_k - \frac{\eta}{m} \sum_j^m \frac{\partial C_{X_j}}{\partial w_k}$$

$$b_l \quad \to \quad b_l' = b_l - \frac{\eta}{m} \sum_j^m \frac{\partial C_{X_j}}{\partial b_l},$$

Learning rate eta η cannot be too small nor to big!

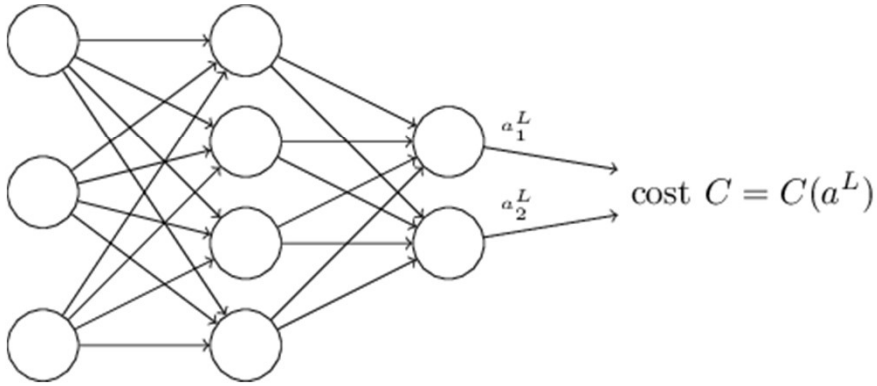
➔ Re-use same training data but randomly re-shuffle it for the next training epoch

❑ Cost functions (assuming input x & y)

$$C = \frac{1}{2n}\sum_x \|y(x) - a^L(x)\|^2,\ a^L(x) \text{ is the vector of activation output after } L \text{ layers}$$

Using cross-entropy as cost function

$$C = -\frac{1}{n}\sum_x \sum_j \left[y_j \ln a_j^L + (1-y_j)\ln(1-a_j^L)\right]$$

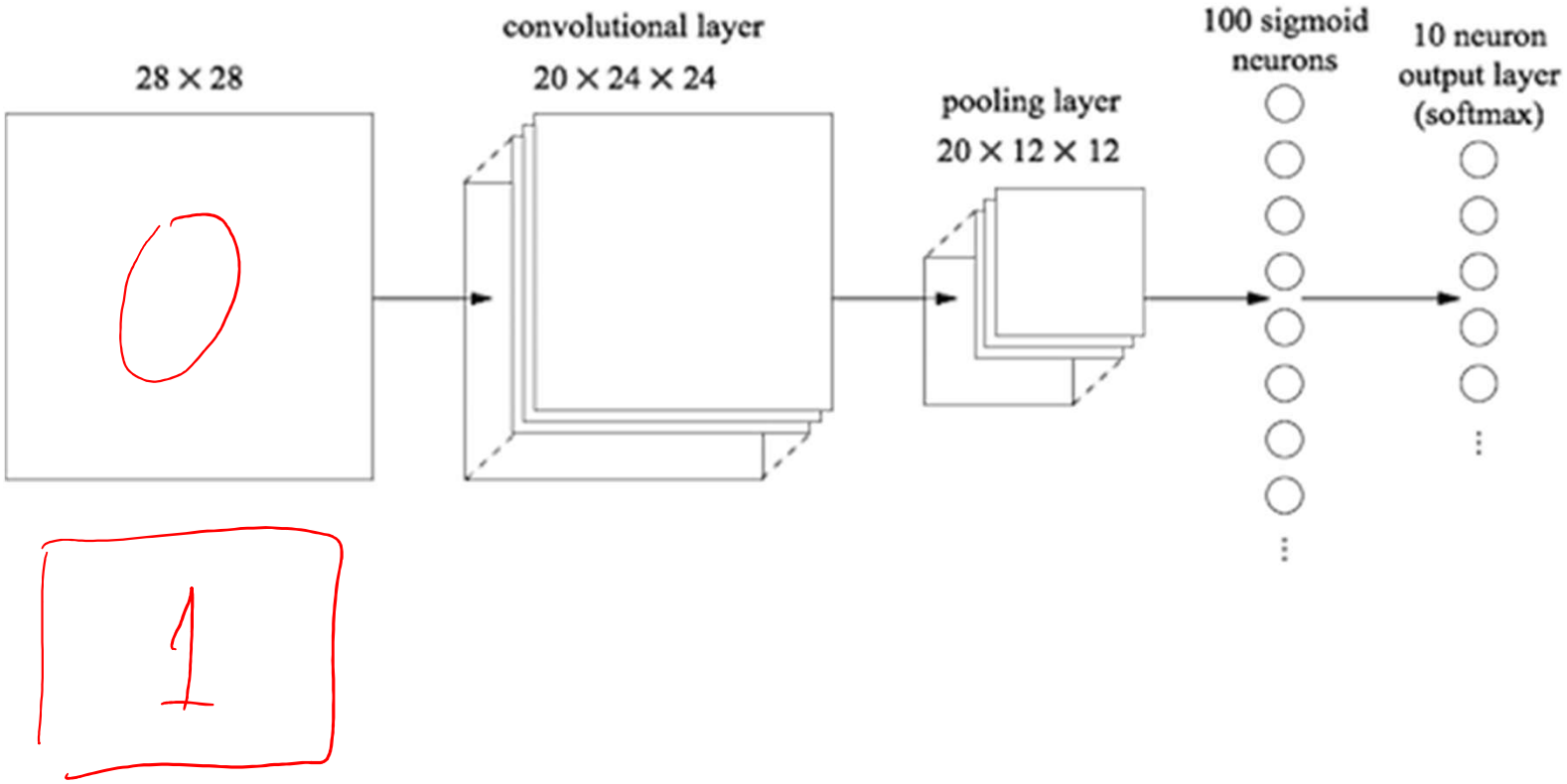


(i) The greater the error, the faster the neurons learn
(ii) Prevent learning slowdown

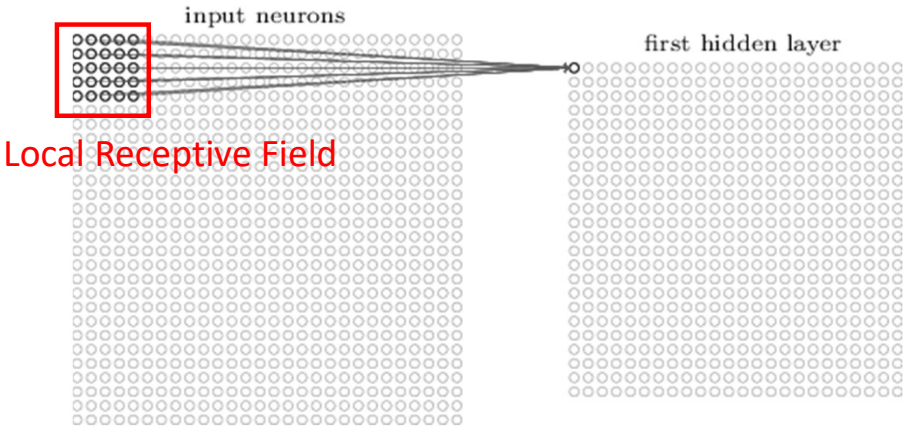
❑ Softmax layer of neurons (cf Boltzmann weights)

$$a_j^L = \frac{e^{z_j^L}}{\sum_k e^{z_k^L}}, \quad z_j^L \equiv \sum_k w_{jk}^L a_k^{L-1} + b_j^L,$$

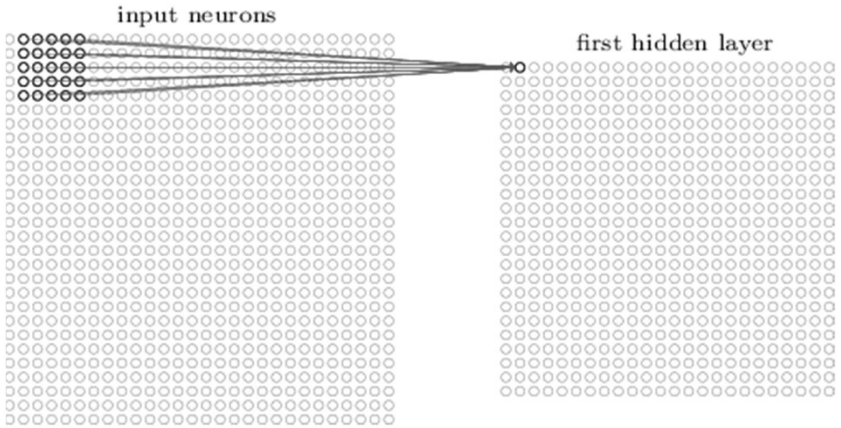
Convolutional Neural Network (CNN)



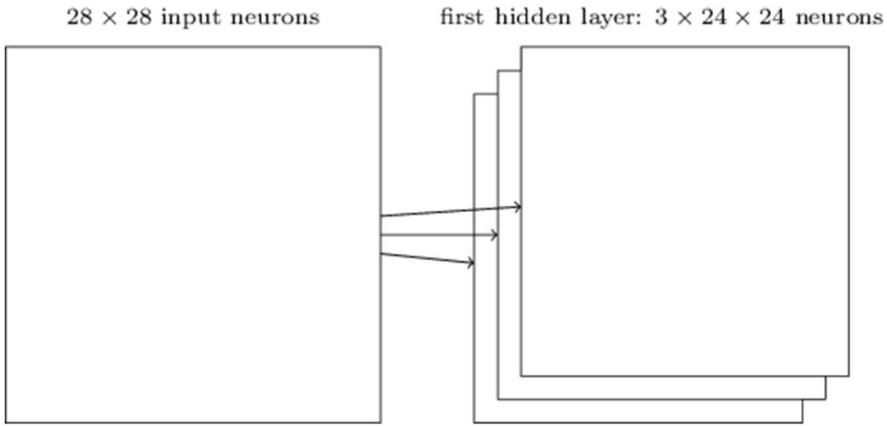
❑ Convolutional Neural Network (CNN)



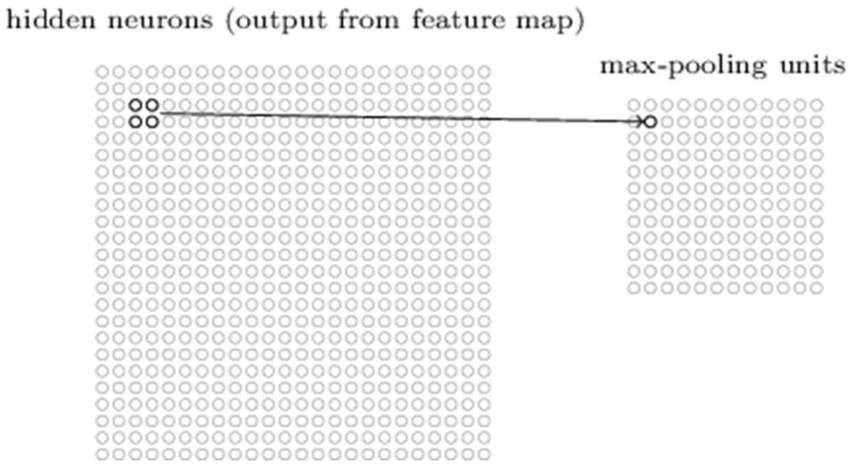
Need not move by one ➔ move by "stride"



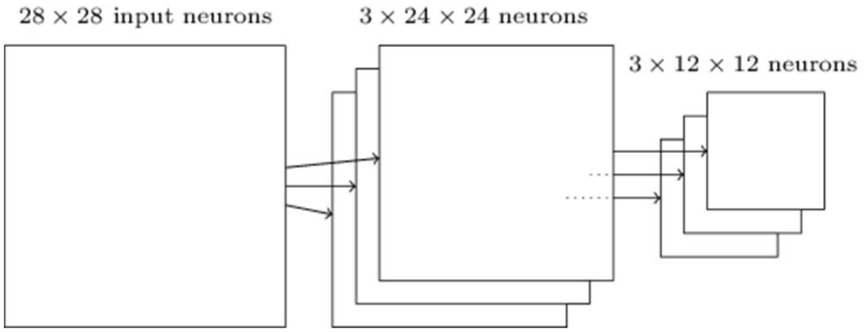
Using 3 different Local Receptive Fields ➔ 3 feature maps/3 kernels



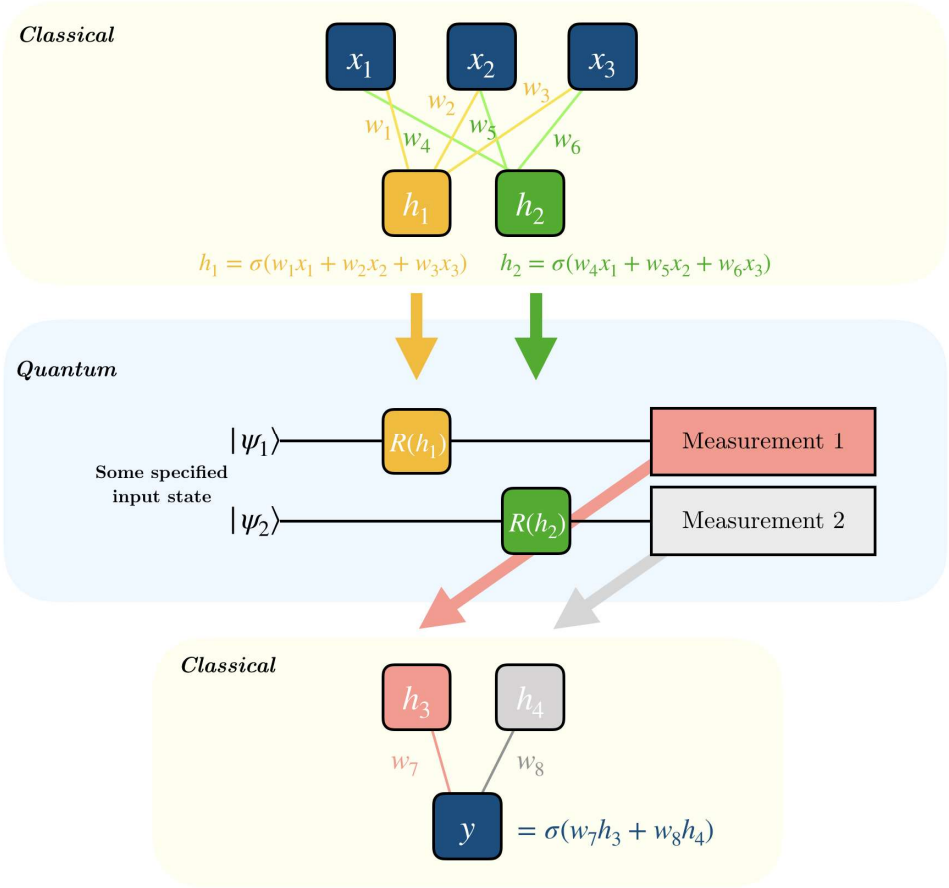
Pooling layer



Alternative to max-pooling: *L2 pooling*



Hybrid classical-quantum neural network*

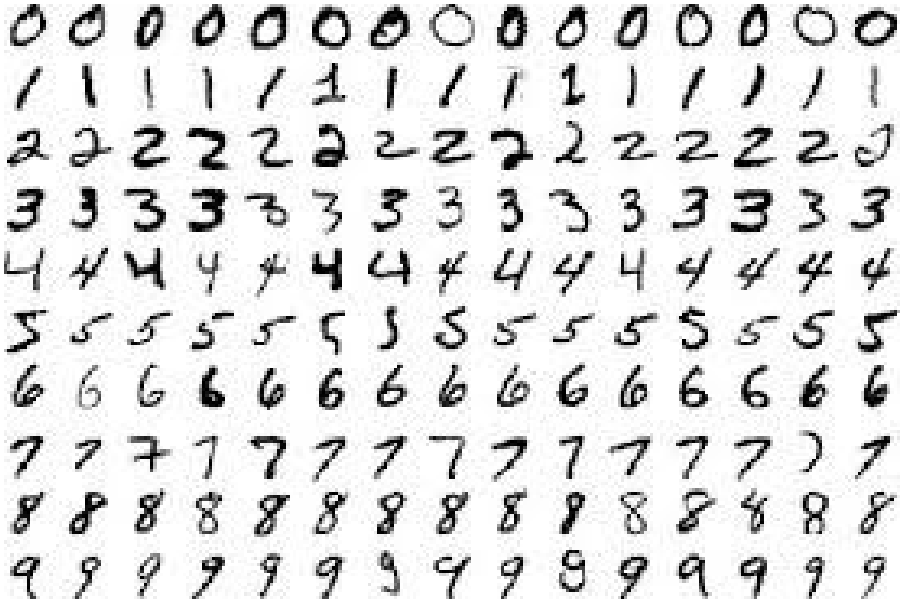


https://qiskit.org/textbook/ch-machine-learning/machine-learning-qiskit-pytorch.html

Hybrid classical-quantum neural network*

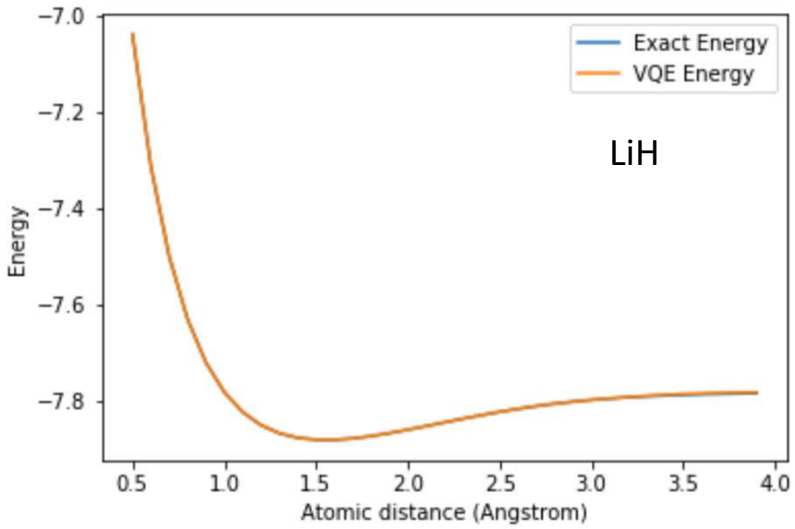
Do Notebook

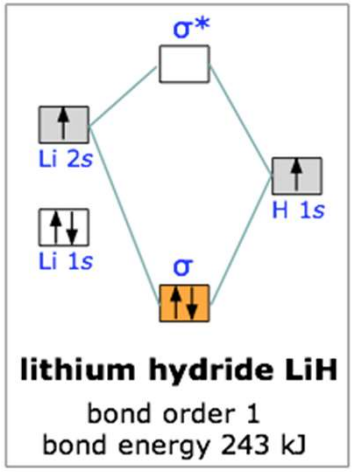
➔ Will use MNIST data set and try to distinguish 0 from 1



Other applications*

- Simulating molecular energy with VQE





https://chem230.fandom.com/wiki/Trend_1.7

- Background: orbitals, Coulomb interaction, Hamiltonian in second quantized form