# Topic 2: Tensor network machine learning

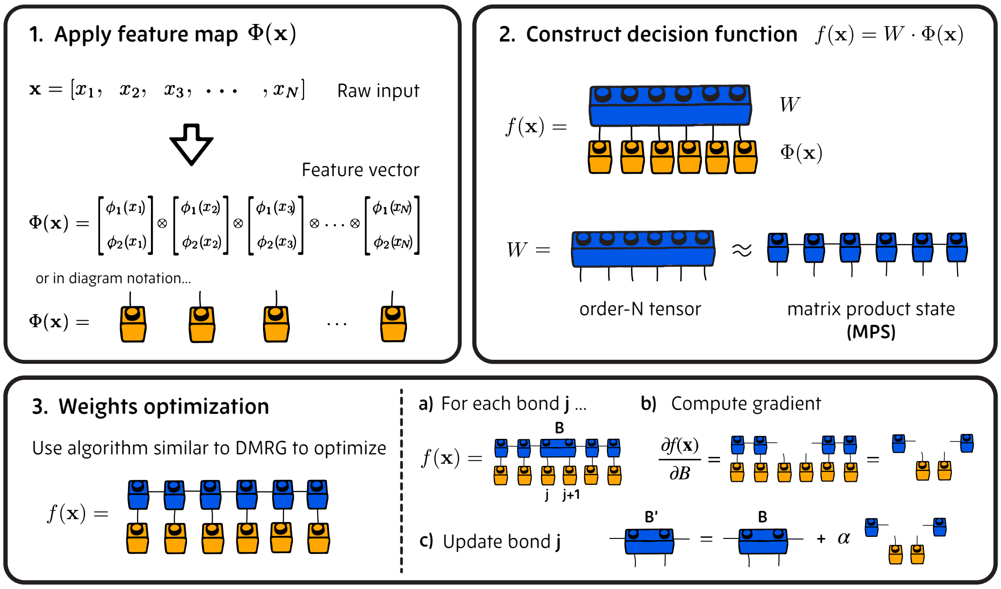
## What to expect

This topic is an opportunity to familiarize yourself with the application of tensor networks for solving machine learning problems. You will focus on supervised learning, specifically on the dataset classification problem. A basic knowledge of machine learning and tensor networks is enough to approach the tasks. Knowledge of Python is required.

## Overview

Tensor network algorithms extend beyond quantum many-body physics and find applications in various machine learning tasks. You can use tensor networks for directly solving the machine learning task, as datasets classification[3], or for compressing the size of a traditional machine learning model[4]. When used to solve a machine learning task, they are better than traditional machine learning models in a specific domain: *explainability*. Tensor networks do not display non-linearities, and are thus much easier to understand.

In this project, you will analyze the performances of a matrix product state (MPS) ansatz to classify the MNIST dataset. You will use the MPS class in `quantum_tea_leaves`, and specifically, employ the optimization algorithm from Ref. [3]. The idea of the algorithm is sketched below. A feature map $\Phi(x)$ is applied to the input data. Then, the decision function is constructed and the weights are optimized via a sweeping algorithm similar to DMRG.



1. Apply feature map $\Phi(\mathbf{x})$
$\mathbf{x} = [x_1, x_2, x_3, \dots, x_N]$
Raw input
Feature vector
$\Phi(\mathbf{x}) = \begin{bmatrix}\phi_1(x_1)\\ \phi_2(x_1)\end{bmatrix} \otimes \begin{bmatrix}\phi_1(x_2)\\ \phi_2(x_2)\end{bmatrix} \otimes \begin{bmatrix}\phi_1(x_3)\\ \phi_2(x_3)\end{bmatrix} \otimes \dots \otimes \begin{bmatrix}\phi_1(x_N)\\ \phi_2(x_N)\end{bmatrix}$
or in diagram notation...
$\Phi(\mathbf{x}) =$
...
2. Construct decision function $f(\mathbf{x}) = W \cdot \Phi(\mathbf{x})$
$f(\mathbf{x}) =$
$W$
$\Phi(\mathbf{x})$
$W =$
$\approx$
order-N tensor
matrix product state (MPS)
3. Weights optimization
Use algorithm similar to DMRG to optimize
$f(\mathbf{x}) =$
a) For each bond j ...
$f(\mathbf{x}) =$
B
j
j+1
b) Compute gradient
$\frac{\partial f(\mathbf{x})}{\partial B} =$
=
c) Update bond j
B'
=
B
+ $\alpha$

## Tasks

- Successfully classify the digits 3 and 8 of the MNIST dataset using the MPS classifier. Encode the dataset mapping each pixel in a qubit with state:

$$|q\rangle = \sqrt{1-p_i}\,|0\rangle + p_i\,|1\rangle\,, \tag{1}$$

  where $|p_i|^2$ is the (normalized) intensity of the pixel. We call this method MPS1. What are the performances?

- Successfully classify the digits 3 and 8 of the MNIST dataset using the MPS classifier. Encode the dataset in a statevector form, such as:

$$|\psi\rangle = \sum_i p_i\,|i\rangle\,, \tag{2}$$

  where $|p_i|^2$ is the (normalized) intensity of the pixel. We call this method MPS2. What are the performances?

- By analyzing the entanglement entropy of the MPS classifier, are you able to understand which are the important characteristics of your system? Which of the two encodings is better for explainabilities? Does it become easier if you instead try to classify 0 and 1? How does the entanglement entropy and the accuracy vary with the bond dimension?

- **Optional:** Solve the same problem with your favorite neural network. Following [4] compress the weights of the neural network. Which method is better memory-wise? MPS1, MPS2, NN, or MPS-compressed NN?

[3] E. Stoudenmire and D. J. Schwab, Advances in neural information processing systems **29** (2016).

[4] Y. Qing et al., arXiv preprint arXiv:2305.06058 (2023).