

Tensor network machine learning

Tensor Network Hackathon, Topic 2

Mentor: Alice Pagano

Algorithm implemented in QTEA

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...



2. Construct decision function $f(\mathbf{x}) = W \cdot \Phi(\mathbf{x})$

$$f(\mathbf{x}) = \begin{array}{c} \text{blue bar with 6 legs} \\ \downarrow \\ \text{6 yellow blocks with 2 legs each} \end{array} \quad \begin{array}{l} W \\ \Phi(\mathbf{x}) \end{array}$$

$$W = \begin{array}{c} \text{blue bar with 6 legs} \\ \downarrow \\ \text{6 vertical lines} \end{array} \approx \begin{array}{c} \text{6 blue blocks with 2 legs each} \\ \downarrow \\ \text{6 vertical lines} \end{array} \quad \begin{array}{l} \text{order-N tensor} \\ \text{matrix product state (MPS)} \end{array}$$

3. Weights optimization

Use algorithm similar to DMRG to optimize

$$f(\mathbf{x}) = \begin{array}{c} \text{6 blue blocks with 2 legs each} \\ \downarrow \\ \text{6 yellow blocks with 2 legs each} \end{array}$$

a) For each bond $j \dots$

$$f(\mathbf{x}) = \begin{array}{c} \text{blue bar B with 6 legs} \\ \downarrow \\ \text{6 yellow blocks with 2 legs each} \end{array}$$

b) Compute gradient

$$\frac{\partial f(\mathbf{x})}{\partial B} = \begin{array}{c} \text{blue bar with 6 legs} \\ \downarrow \\ \text{6 yellow blocks with 2 legs each} \end{array} = \begin{array}{c} \text{blue bar with 6 legs} \\ \downarrow \\ \text{6 yellow blocks with 2 legs each} \end{array}$$

c) Update bond j

$$\begin{array}{c} \text{blue bar B' with 6 legs} \\ \downarrow \\ \text{6 yellow blocks with 2 legs each} \end{array} = \begin{array}{c} \text{blue bar B with 6 legs} \\ \downarrow \\ \text{6 yellow blocks with 2 legs each} \end{array} + \alpha \begin{array}{c} \text{blue bar with 6 legs} \\ \downarrow \\ \text{6 yellow blocks with 2 legs each} \end{array}$$

Tasks 1 and 2

Classify the digits 3 and 8 of the MNIST dataset using the MPS classifier.

● Encoding 1: $|q\rangle = \sqrt{1 - p_i} |0\rangle + p_i |1\rangle$

● Encoding 2: $|\psi\rangle = \sum_i p_i |i\rangle$

Compare the performances of the two different encodings!

Steps for tasks 1 and 2

1. Load MNIST dataset

2. Build classifier, i.e. convert dataset into a list of MPS

```
svd, loss = tn_classifier.ml_optimize_mps(X_train_mps,  
                                           y_train,  
                                           batch_size=batch_size,  
                                           learning_rate=learning_rate,  
                                           num_sweeps=num_sweeps,  
                                           n_jobs=1,  
                                           verbose=True)
```

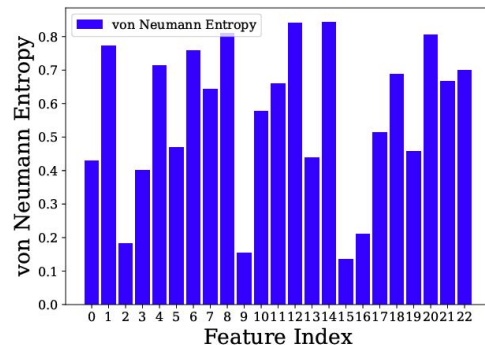
3. Optimize MPS

4. Get accuracy

```
y_train_pred = tn_classifier.ml_predict(X_train_mps, n_jobs=1)  
y_test_pred = tn_classifier.ml_predict(X_test_mps, n_jobs=1)
```

Task 3: analyze entanglement

- Analyze entanglement entropy of the bipartitions
- Are you able to understand the important characteristics of your system?
Which of the two encodings is better for explainability?
- How does the entanglement entropy and the accuracy vary with the bond dimension?



[optional] Task 4: weight compression with MPS

- Solve the same problem with your favorite neural network
- Following <https://arxiv.org/pdf/2305.06058> compress the weights of the neural network
- Which method is better memory-wise? MPS1, MPS2, NN, or MPS-compressed NN?