Quantum Machine Learning Seminars

Week 7: Quantum Neural Networks

Dr. Hassan Alshal

Outline

1. Kernel Methods
2. Neural Networks

Kernel Methods

Support Vector Machine

- SVM is a binary classifier that finds a maximum-margin separating hyperplane in feature space, written as $w \cdot x + b = 0$, with labels $y_i \in \{\pm 1\}$ for training points $x_i \in \mathbb{R}^d$.
- The goal is to maximize the margin by solving $\arg\min_{w,b} \frac{|w|^2}{2}$ subject to $y_i(w \cdot x_i + b) \geq 1$ for all i.
- The Lagrangian introduces multipliers $\alpha_i \geq 0$ as $\mathcal{L}(w, b, \alpha) = \frac{1}{2}|w|^2 - \sum_{i=1}^{N} \alpha_i \big(y_i(w \cdot x_i + b) - 1\big)$.
- Stationarity gives the key structural facts $w = \sum_{i=1}^{N} \alpha_i y_i x_i$ and $\sum_{i=1}^{N} \alpha_i y_i = 0$, meaning the solution is a weighted combination of training points and only $\alpha_i > 0$ points act as support vectors.

Quantum Classification

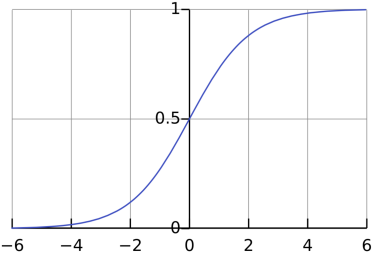
Support Vector Machine

- Eliminating w, b yields the dual objective $\mathcal{L}_d(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j)$ with constraints $\alpha_i \geq 0$ and $\sum_i \alpha_i y_i = 0$.
- The classifier evaluates new inputs via a kernel expansion $y(x) = \operatorname{sgn}\left(\sum_{i=1}^{N} \alpha_i \kappa(x_i, x) + b \right)$, where in the linear case $\boxed{\kappa(x_i, x) = x_i \cdot x}$.
- The "kernel trick" generalizes to nonlinearly separable data by embedding with $\phi : \mathbb{R}^d \to \mathbb{R}^{d'}$ and defining $\kappa(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$, so the algorithm uses inner products in feature space without explicitly constructing $\phi(x)$.
- The central quantum object is the "kernel matrix" K with entries $K_{ij} = \kappa(x_i, x_j)$; and the goal is to estimate it.

Neural Networks

Logistic Regression

- The dataset contains examples/trials $\{(x_i, y_i)\}_{i=1}^N$, $y_i \in \{\pm 1\}$, and each x_i has features $\{x_{im}\}_{m=1}^M$.
- Each feature is assigned a weight w_m, and the model forms a linear score $z_i = \sum_{m=1}^M w_m x_{im} + b_0$, where weights measure the importance of each feature.
- For preactivation z_i, the prediction $\hat{y}_i = \text{sign}(z_i) \lessgtr 0$, i.e., $\hat{y}_i \in \{\pm 1\} \Rightarrow$ activation $\varphi(z_i) \lessgtr 1/2$.



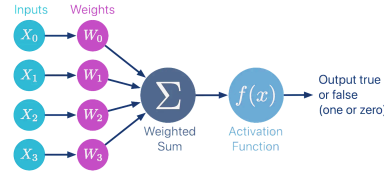
- Conventionally, classification $\{\pm 1\}$ is based on:

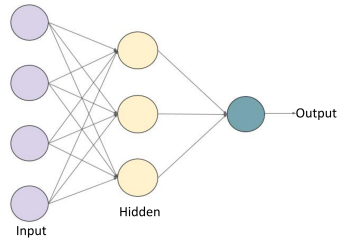
$$\boxed{\mathbb{P}(y = 1 \mid x_i) = \varphi(z_i) = \frac{1}{1 + e^{-z_i}} \Rightarrow \ln\left[\frac{\mathbb{P}(y_i = 1)}{1 - \mathbb{P}(y_i = 1)}\right] = z_i}.$$

Neural Networks

Perceptron & Feedforward Neural Networks

- For single perceptron, weights are updated by $w_j^{(t+1)} = w_j^{(t)} + \eta[y_i - \hat{y}_i]x_{ij}$.
- With one hidden layer, the model takes the form $f(x, W_1, W_2) = \varphi_2(W_2 h + b_2)$ where $h = \varphi_1(W_1 x + b_1)$. Then generalize by combining many perceptrons into feedforward neural networks.
- Training a feedforward neural network is posed as minimizing a loss such as $\mathcal{L}(W_1, W_2) = \sum_{i=1}^{N} \left| f\left(x_i, W_1, W_2\right) - t_i \right|^2$, which is solved by backpropagation with gradient-descent updates $w_{ij}^{(t+1)} = w_{ij}^{(t)} - \eta \dfrac{\partial \mathcal{L}(W_1, W_2)}{\partial w_{ij}}$.

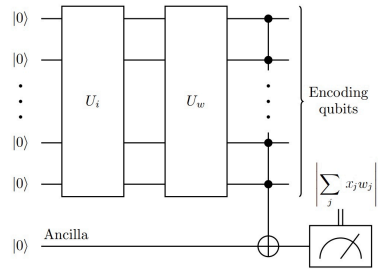




Neural Networks

Quantum Perceptron

- The classical data are $\mathbf{x}, \mathbf{w} \in \mathbb{R}^d$ with components $x_i, w_i \in \{-1, 1\}$, stored in an n-qubit register.
- The x, w vectors are amplitude encoded as $|\psi_{\mathbf{x}}\rangle = \frac{1}{\sqrt{d}} \sum_{i=0}^{d-1} x_i |i\rangle$ and $|\psi_{\mathbf{w}}\rangle = \frac{1}{\sqrt{d}} \sum_{i=0}^{d-1} w_i |i\rangle$.
- Start with $|0 \cdots 0\rangle$ then prepare the input state through a unitary $U_{\mathbf{x}}$ such that $|\psi_{\mathbf{x}}\rangle = U_{\mathbf{x}} |0 \cdots 0\rangle$.
- A second unitary $U_{\mathbf{w}}$ is chosen so that it maps the weight state to the last computational basis state, namely $U_{\mathbf{w}} |\psi_{\mathbf{w}}\rangle = |1 \cdots 1\rangle = |d-1\rangle$.
- Now register state becomes $|\phi_{\mathbf{w},\mathbf{x}}\rangle = U_{\mathbf{w}} U_{\mathbf{x}} |0 \cdots 0\rangle = U_{\mathbf{w}} |\psi_{\mathbf{x}}\rangle = \sum_{i=0}^{d-1} c_i |i\rangle$.

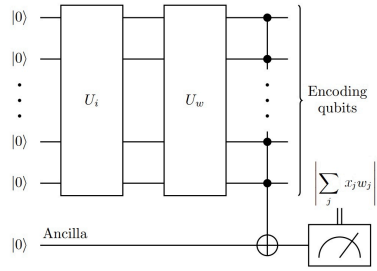


Tacchino et al, arXiv:1811.02266

Neural Networks

Quantum Perceptron

- The amplitude of the basis state $|d-1\rangle$ equals the normalized classical overlap, since $\langle\psi_{\mathbf{w}}|\psi_{\mathbf{x}}\rangle = \frac{1}{d}\,\mathbf{w}\cdot\mathbf{x} = \langle d-1|U_{\mathbf{w}}|\psi_{\mathbf{x}}\rangle = c_{d-1}$.
- An ancilla qubit is then prepared in $|0\rangle_a$, and apply $(n+1)$-qubit `CNOT` so ancilla flips only when the encoding register is in the state $|d-1\rangle$.
- Measuring the ancilla gives outcome 1 with probability $\mathbb{P}_a(1) = |c_{d-1}|^2 = |\mathbf{w}\cdot\mathbf{x}|^2/d^2$ as the perceptron outcome.
- This nonlinear activation of the quantum perceptron $|0\rangle_a \to |1\rangle_a$ comes with cost $\mathcal{O}(d)$.

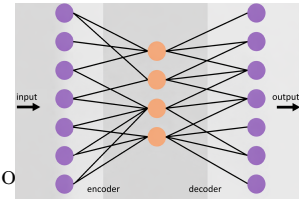


Tacchino et al, arXiv:1811.02266

Neural Networks

Classical Autoencoder

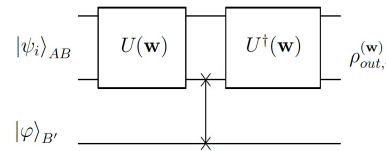
- The encoder compresses the input into a lower-dimensional latent layer as $z = f_w(x)$, where $z \in \mathbb{R}^m$, $x \in \mathbb{R}^n$ with $m < n$.
- The latent layer, or bottleneck, retains only the most important features of the data, i.e., it does NOT memorize every component directly.
- The decoder reconstructs the original input from the latent code as $\hat{x} = g_\phi(z)$, expanding the compressed input back into the original feature space, where $\phi \equiv \phi(w, b)$.
- The network is trained by minimizing a reconstruction loss such as $\mathcal{L}(x, \hat{x}) = \|x - \hat{x}\|^2$. And w and ϕ are updated concurrently.



Neural Networks

Quantum Autoencoder

- Start with $|\psi_i\rangle_{AB}|\varphi\rangle_{B'}$, where A stores the later compressed qubits and B stores trash qubits.
- The encoder $U(\mathbf{w})$ acts on $A+B$ as $U(\mathbf{w})|\psi_i\rangle_{AB} = |\psi_i^c\rangle_A|\varphi\rangle_B$.
- A SWAP gate is then applied between B and B' so B contains the reference state $|\varphi\rangle$, while B' has trash state.
- The decoder $U^\dagger(\mathbf{w})$ still acts on $A+B$: $|\psi_i^c\rangle_A|\varphi\rangle_B|\varphi\rangle_{B'} \to |\psi_i\rangle_{AB}|\varphi\rangle_{B'}$.
- The parameters $\mathbf{w}$ are trained by minimizing $L(\mathbf{w}) = -\sum_i \mathcal{F}(|\varphi\rangle_{B'}, \rho^{(\mathbf{w})}_{B',i})$, where $\mathcal{F} = {}_{B'}\langle\varphi|\rho^{(\mathbf{w})}_{B',i}|\varphi\rangle_{B'}$.



Thank You!