Quanteninformatik 2

02.04.2023

basierend auf einer Vorlesung von Prof. Dr. Dr. h.c. Frank Leymann und Dr. Johanna Barzen, IAAS, Universität Stuttgart

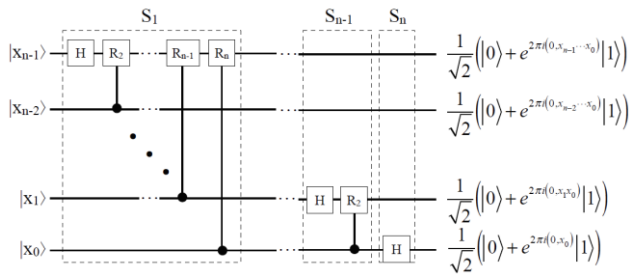
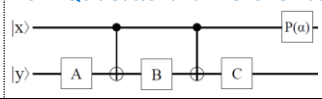
Dichteoperator

Gem. Zustand	$\rho = \sum_i p_i \lvert\Psi_i\rangle\langle\Psi_i\rvert$ mit $\sum_i p_i = 1$	Gem. Zustand ≠ Superpos.:	$\rho = \frac{1}{2}\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} \Rightarrow$ Gemisch $\lvert0\rangle, \lvert1\rangle$, $\frac{1}{2}\begin{pmatrix}1 & 1\\ 1 & 1\end{pmatrix} \Rightarrow \frac{1}{\sqrt{2}}\lvert0\rangle + \frac{1}{\sqrt{2}}\lvert1\rangle$
Eigenschaften	$\mathrm{Tr}(\rho) = 1,\ \rho$ positiv $(\forall\lambda_i > 0)$	Reiner Zustand: $\mathrm{Tr}(\rho^2) = 1$	$\mathrm{Tr}(\rho^2)$... Reinheit; Messung: $\langle\hat{O}\rangle_\rho = \sum_i p_i \langle\varphi_i\rvert\hat{O}\lvert\varphi_i\rangle = \mathrm{Tr}(\hat{O}\rho)$
	Dichteoperator enthält alle Informationen, die man durch Messung des Gemischs erhalten kann Verschiedene Gemische können denselben Dichteoperator haben (z.B. Gemisch $\lvert0\rangle, \lvert1\rangle$, Gemisch $\lvert+\rangle, \lvert-\rangle$)		
	ρ_{ii} ... Populationen (Wahrscheinlichkeit des Auftretens des Zustands entsprechend der Spalte i), ρ_{ij} ... Kohärenzen (Korrelation der Zustände in Zeile i und Spalte j)		
Partielle Spur	Zurückführen eines Gemischs in einem „großen System" auf ein „kleineres" System. Einige Komponenten des „größeren" Systems werden *ausgespurt*. Die Partielle Spur ist ungefähr die Inverse zum Tensorprodukt. (Bei Produktzuständen genau): $\rho_{AB} \overset{\text{def}}{=} \rho_A \otimes \rho_B \Rightarrow \mathrm{Tr}_B(\rho_{AB}) = \sum_j (\mathbb{1}_A \langle j\rvert_B)\rho_{AB}(\mathbb{1}_A \lvert j\rangle_B) = \sum_j (\mathbb{1}_A\rho_A\mathbb{1}_A)(\langle j\rvert_B\rho_B\lvert j\rangle_B) = \rho_A\,\mathrm{Tr}(\rho_B) = \rho_A\mathbb{1}_B \Rightarrow \boxed{\mathrm{Tr}_B(\rho_{AB}) = \rho_A}$		
	Das „kleinere" System ist durch den *reduzierten Dichteoperator* gegeben, der durch die partielle Spur berechnet wird.		
	Partielle Spuren reiner Zustände können gemischt sein. Z.B. Bell-Zustand $\lvert\varphi^+\rangle = \frac{1}{\sqrt{2}}\lvert0\rangle_A\lvert0\rangle_B + \frac{1}{\sqrt{2}}\lvert1\rangle_A\lvert1\rangle_B \Rightarrow$ $\mathrm{Tr}_B(\rho_{AB}) = \mathrm{Tr}_B(\lvert\varphi^+\rangle\langle\varphi^+\rvert) = \frac{1}{2}\mathbb{1}_A, \Rightarrow \mathrm{Tr}\left(\left(\frac{1}{2}\mathbb{1}\right)^2\right) < 1 \Rightarrow$ gemischt! Fehler entstehen durch Verschränkung mit Umgebung.		
Lokale Messg	$\langle\hat{O}\otimes\mathbb{1}_B\rangle_{\rho_{AB}} = \mathrm{Tr}(\hat{O}\rho_A)$ Mit ρ_A kann man die Erwartungswerte aller lokaler Messungen auf A berechnen.		
Bereinigung	Sei ρ_A der verrauschte Zustand. Bereinigung: reiner Zustand $\lvert\Psi\rangle_{RA}$: $\mathrm{Tr}_R(\lvert\Psi\rangle\langle\Psi\rvert_{RA}) = \rho_A$ (nicht eindeutig)		

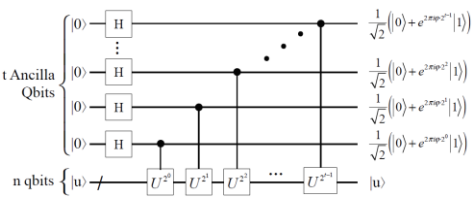
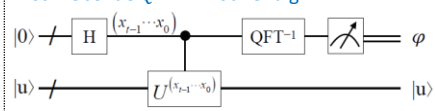
Entropie

Shannon:	$S(X) = H(p_1, \dots, p_n) = -\sum_x p_x \log_2(p_x)$	Von Neumann:	$S(\rho) = -\mathrm{Tr}(\rho\log_2(\rho)) = -\sum_x \lambda_x \log_2(\lambda_x) \geq 0$ | $S(\rho) = 0 \Leftrightarrow \rho$ rein
Messung:	$S(\rho') \geq S(\rho)$ ... Entropie nimmt durch Messung zu. $S(\rho') = S(\rho) \Leftrightarrow \rho' = \rho$	weiters:	$S(\rho\otimes\sigma) = S(\rho) + S(\sigma)$
$S(\rho) \leq H(X)$	N Qbits $\{\lvert\Psi_1\rangle \dots \lvert\Psi_n\rangle\}$ orthogonal $\Rightarrow S(\rho) = H(X)$	N Qbits $\{\lvert\Psi_1\rangle \dots \lvert\Psi_n\rangle\}$ nicht orthogonal $\Rightarrow S(\rho) < H(X)$	

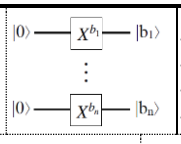
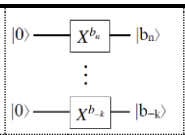
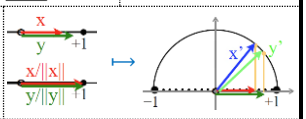
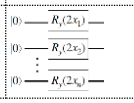
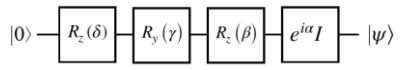
Quantum Fourier Transform QFT

QFT	$QFT_N = \frac{1}{\sqrt{N}}\begin{pmatrix}1 & 1 & 1 & \cdots & 1\\ 1 & \omega_N & \omega_N^2 & \cdots & \omega_N^{N-1}\\ 1 & \omega_N^2 & \omega_N^4 & \cdots & \omega_N^{2(N-1)}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 1 & \omega_N^{N-1} & \omega_N^{2(N-1)} & \cdots & \omega_N^{(N-1)(N-1)}\end{pmatrix}$ mit $\omega_N \overset{\text{def}}{=} e^{2\pi i/N}$	Sei $\{\lvert0\rangle, \lvert1\rangle, \dots, \lvert N-1\rangle\}$ ONB; dann ist $QFT_N = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1}\omega_N^{jk}\lvert k\rangle = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{2\pi ijk/N}\lvert k\rangle$ $QFT_2 = \begin{pmatrix}1 & 1\\ 1 & \omega_2\end{pmatrix} = \begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} = \hat{H}$ (Hadamard)
Binärdarstellg	$j \in \mathbb{N} \Rightarrow j = (x_{n-1} \dots x_1 x_0)$ mit $x_i \in \{0,1\} \Rightarrow j = \sum_{i=0}^{n-1} x_i 2^i$ $l: 0 \leq l \leq 1 \Rightarrow l = (0, y_{m-1} \dots y_1 y_0)$ mit $y_i \in \{0,1\} \Rightarrow l = \sum_{i=0}^{m} y_i 2^{-i}$	dann ist $(x_{n-1} \dots x_1 x_0 y_{m-1} \dots y_1 y_0)$ eine positive rationale Zahl
QFT Produkt-darstellung	$QFT_N(\lvert x_{n-1} \dots x_0\rangle) = \frac{1}{2^{n/2}}\left(\lvert0\rangle + e^{2\pi i(0,x_0)}\lvert1\rangle\right) \otimes \left(\lvert0\rangle + e^{2\pi i(0,x_1x_0)}\lvert1\rangle\right) \otimes \dots \otimes \left(\lvert0\rangle + e^{2\pi i(0,x_{n-1}\dots x_0)}\lvert1\rangle\right)$	
Schaltkreis		$\hat{R}_k = \begin{pmatrix}1 & 0\\ 0 & e^{2\pi i/2^k}\end{pmatrix}$ Swaps an den Ausgängen noch notwendig! QFT benötigt $O(n^2)$ 1-Qbit und CNOT Gatter FFT benötigt im Vergleich $O(n \cdot n^2)$ Schritte Exponentieller Speedup! Sei $\hat{U} \in \mathbb{C}^{2\times2}$ unitär. Dann lässt sich $\hat{U}^c$ durch vier 1-Qbit Gatter und zwei CNOT darstellen 

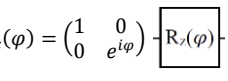
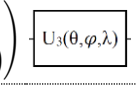
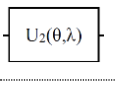
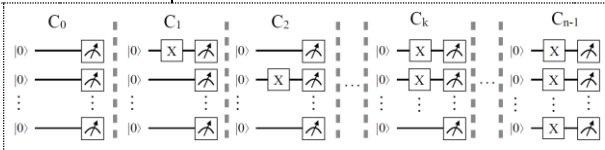
Quantum Phase Estimation QPE

Eigenwerte	$\hat{U}$ unitär $\Rightarrow \forall\lambda_i: \lambda_i = e^{2\pi i\varphi} \Rightarrow$ Bestimmung der Phase φ is Bestimmung des Eigenwertes λ_i	
Schaltkreis m. t Ancilla Nachkommastellen Eigenvektor $\lvert u\rangle$ muss bekannt sein!		Anschließende QFT^{-1} notwendig:  Komplexität $O(t^2) = O(n^2)$. Exponentieller Speedup! Um bei nicht-rationalen Phasen q Stellen mit $p = 1 - \varepsilon$ Wahrscheinlichkeit zu bestimmen, braucht man $t = q + \left\lceil\log\left(2 + \frac{1}{2\varepsilon}\right)\right\rceil$ Ancilla Qbits

State Preparation

Basis Encoding	$x \in \mathbb{N} \Rightarrow$ $x = \sum_{i=0}^{n-1} b_i 2^i \Rightarrow$ $x \mapsto \lvert b_{n-1}..b_0\rangle \Rightarrow$ $x = \hat{X}^{b_n} \otimes ... \otimes \hat{X}^{b_0} \lvert 0 ... 0\rangle$		$x \in \mathbb{R} \Rightarrow$ $x = \sum_{i=0}^{n-1} b_i 2^i + \sum_{i=1}^{k} b_{-i} 2^{-i} \Rightarrow$ $x \mapsto \lvert b_{n-1}..b_0 b_{-1} ... b_{-k}\rangle$ $x = \hat{X}^{b_n} \otimes ... \otimes \hat{X}^{b_{-k}} \lvert 0 ... 0\rangle$		Vektor: $x = \begin{pmatrix} 0.7 \\ 0.1 \end{pmatrix}$ $x = \begin{pmatrix} 1011 \\ 1001 \end{pmatrix}$ $x \mapsto \lvert 1011\ 1001\rangle$
Amplitude encoding	Required Qbits: $\lceil \log_2(N) \rceil$ Required Gates: 4^N	$\vec{x} \in \mathbb{R}^N \backslash \{0\} \Rightarrow$ $\vec{x} \mapsto \sum_i \frac{x_i}{\lVert \vec{x} \rVert} \lvert i\rangle$	$A \in \mathbb{R}^{n \times m} \backslash \{0\} \Rightarrow$ $A \mapsto \sum_{ij} \frac{a_{ij}}{\lVert A \rVert} \lvert i\rangle \lvert j\rangle$	Problem: Distanzen zw. Vektoren gehen mit Normierung verloren. Zusätzliche Dimension!	
(Tensor) Product Encoding (angle encod.)	$\vec{x} \in \mathbb{R}^n, \forall i: \lvert x_i \rvert \leq 1$, req: n Qbits,n gates $x \mapsto \begin{pmatrix} \cos(x_0) \\ \sin(x_0) \end{pmatrix} \otimes ... \otimes \begin{pmatrix} \cos(x_{n-1}) \\ \sin(x_{n-1}) \end{pmatrix}$		$\hat{R}_y(2x) = \begin{pmatrix} \cos(x) & -\sin(x) \\ \sin(x) & \cos(x) \end{pmatrix} \Rightarrow$ $\hat{R}_y(2x) \lvert 0\rangle = \cos(x) \lvert 0\rangle + \sin(x) \lvert 1\rangle$		
Präp. beliebiger Zustand	$\lvert \Psi\rangle = \cos(\vartheta) \lvert 0\rangle + e^{i\varphi} \sin(\vartheta) \lvert 1\rangle = \hat{U} \lvert 0\rangle$ mit $\hat{U} = \begin{pmatrix} \cos(\vartheta) & -e^{-i\varphi} \sin(\vartheta) \\ e^{i\varphi} \sin(\vartheta) & \cos(\vartheta) \end{pmatrix} = e^{i\alpha} \hat{R}_z(\beta) \hat{R}_y(\gamma) \hat{R}_z(\delta)$				

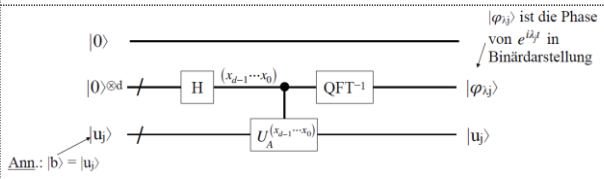
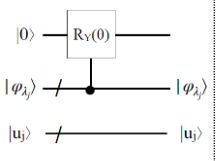
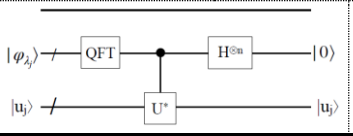
Transpilation und Readout-Fehler

Tiefe:	Anzahl Schichten	Breite:	Anzahl manipulierter Qbits.	Transpilation: Übersetzen in Hardware-Gates. Topologie! Fidelity!
Rotationen:	$\hat{R}_x(\varphi) = \begin{pmatrix} \cos\left(\frac{\varphi}{2}\right) & -i \sin\left(\frac{\varphi}{2}\right) \\ -i \sin\left(\frac{\varphi}{2}\right) & \cos\left(\frac{\varphi}{2}\right) \end{pmatrix}$ 	$\hat{R}_y(\varphi) = \begin{pmatrix} \cos\left(\frac{\varphi}{2}\right) & -\sin\left(\frac{\varphi}{2}\right) \\ \sin\left(\frac{\varphi}{2}\right) & \cos\left(\frac{\varphi}{2}\right) \end{pmatrix}$ 	$\hat{R}_z(\varphi) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{pmatrix}$ 	
	$\hat{U}_3(\vartheta, \varphi, \lambda) = \begin{pmatrix} \cos\left(\frac{\vartheta}{2}\right) & -e^{i\lambda} \sin\left(\frac{\vartheta}{2}\right) \\ e^{i\varphi} \sin\left(\frac{\vartheta}{2}\right) & e^{i(\lambda+\varphi)} \cos\left(\frac{\vartheta}{2}\right) \end{pmatrix}$ 	$\hat{U}_2(\varphi, \lambda) = \hat{U}_3\left(\frac{\pi}{2}, \varphi, \lambda\right) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -e^{i\lambda} \\ e^{i\varphi} & e^{i(\lambda+\varphi)} \end{pmatrix}$ 		
Unfolding	$\vec{t} = C\vec{m}$ mit C ... Kalibrierungsmatrix, $\vec{m}$ ...measured values, $\vec{t}$ ... true values $C_{ij} = \text{prob}(\text{measured } j \lvert \text{true } i) \Rightarrow$ C_i m mal ausführen, es wird k mal j gemessen $\Rightarrow C_{ij} = k/m$				

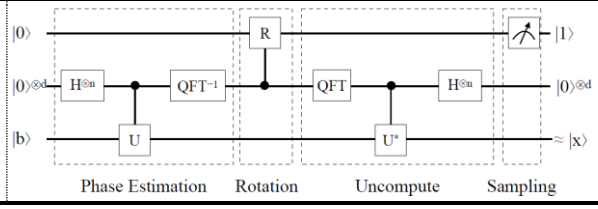
Simulation eines Hamilton Operators

Matrix-Exponential	$X \in \mathbb{C}^{n \times n} \Rightarrow e^X = \sum_{k=0}^{\infty} \frac{1}{k!} X^k$, $e^{\mathbb{0}} = \mathbb{1}$, $e^{aX} e^{bX} = e^{(a+b)X}$, $e^X e^{-X} = \mathbb{1}$, $(e^X)^{-1} = e^{-X}$, $(e^X)^\dagger = e^{X^\dagger}$, $e^{YXY^{-1}} = Ye^X Y^{-1}$ $\det(e^X) = e^{\text{tr}(X)}$, $e^{\text{diag}(\lambda_1,...,\lambda_n)} = \text{diag}(e^{\lambda_1}, ..., e^{\lambda_n})$, X hermitesch $\Rightarrow e^{iX}$ unitär, $e^{X+Y} = e^X e^Y e^{-[X,Y]/2}$, $e^X e^Y = e^Y e^X e^{[X,Y]}$
Lie-Trotter	A, B hermitesch, $t \geq 0 \Rightarrow e^{it(A+B)} = \lim_{N \to \infty} \left(e^{\frac{itA}{N}} e^{\frac{itB}{N}} \right)^N$
Trotter-Entw.	sei $H = H_1 + \cdots + H_M$ und H ...k-lokal $\Rightarrow e^{-iHt} \approx (e^{-iH_1 t/L} ... e^{-iH_M t/L})^L$ (wenn t nicht klein ist, dann L entspr. groß wählen)

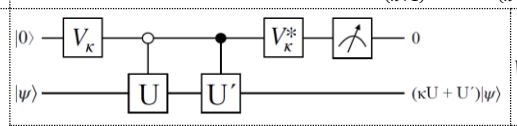
HHL (Lösen linearer Gleichungssysteme)

Lineares Gleichungssystem	$A\vec{x} = \vec{b}$ mit $A \in \mathbb{C}^{n \times n}$ hermitesch	Sei $A \in \mathbb{C}^{n \times m}$ nicht hermitesch, dann ist $\begin{pmatrix} \mathbb{0} & A \\ A^\dagger & \mathbb{0} \end{pmatrix}$ hermitesch und quadratisch. Lösung von $\begin{pmatrix} \mathbb{0} & A \\ A^\dagger & \mathbb{0} \end{pmatrix} \begin{pmatrix} \vec{0} \\ \vec{x} \end{pmatrix} = \begin{pmatrix} \vec{b} \\ \vec{0} \end{pmatrix}$ ist die Lösung von $A\vec{x} = \vec{b}$	
HHL Schritt 1	Spektralzerlegung $A = \sum_i \lambda_i \lvert u_i\rangle\langle u_i \rvert$ mit $\{\lvert u_i\rangle\}$ ONB. Vorläufige Annahme: $\lvert b\rangle = \lvert u_j\rangle$. Weil A hermitesch $\Rightarrow U = e^{iA}$ unitär. $\sigma(A) = \{\lambda_k\}$, EV $\{\lvert u_k\rangle\} \Rightarrow \sigma(U) = \{e^{i\lambda_k}\}$, EV $\{\lvert u_k\rangle\}$ (A und U haben dieselben Eigenvektoren)		
HHL Schritt 2 QPE		Anwenden der Phase Estimation auf $\lvert 0\rangle^{\otimes d} \otimes \lvert b\rangle$ Annahme: $\lvert b\rangle = \lvert u_i\rangle$ Anfangszustand: $\lvert \Psi_0\rangle = \lvert 0\rangle \otimes \lvert 0\rangle^{\otimes d} \otimes \lvert b\rangle$ Nächster Zustand: $\lvert \Psi_1\rangle = \lvert 0\rangle \otimes \lvert \varphi_{\lambda_j}\rangle \otimes \lvert b\rangle$	
HHL Schritt 3		$\hat{R}_y$-Rotation des ersten Acilla-Qbits mit $\vartheta = \arccos\left(\frac{c}{\varphi_{\lambda_j}}\right)$ C .. Hyperparameter $\lvert \Psi_2\rangle = \left(\sqrt{1 - \frac{c^2}{\varphi_{\lambda_j}^2}} \lvert 0\rangle + \frac{c}{\varphi_{\lambda_j}} \lvert 1\rangle \right) \otimes \lvert \varphi_{\lambda_j}\rangle \otimes \lvert b\rangle$	
HHL Schritt 4		Annahme $\lvert b\rangle = \lvert u_i\rangle$ wird fallen gelassen. $\lvert b\rangle = \sum_i \beta_i \lvert u_i\rangle \Rightarrow$ $\lvert \Psi_3\rangle = \sum_i \beta_i \left(\sqrt{1 - \frac{c^2}{\varphi_{\lambda_j}^2}} \lvert 0\rangle + \frac{c}{\varphi_{\lambda_j}} \lvert 1\rangle \right) \otimes \lvert \varphi_{\lambda_j}\rangle \otimes \lvert b\rangle$	
HHL Schritt 5: QPE^{-1}		$\lvert \Psi_4\rangle = \sum_i \beta_i \left(\sqrt{1 - \frac{c^2}{\varphi_{\lambda_j}^2}} \lvert 0\rangle + \frac{c}{\varphi_{\lambda_j}} \lvert 1\rangle \right) \otimes \lvert 0\rangle^{\otimes d} \otimes \lvert b\rangle$	
HHL Schritt 6			$\sigma(A) = \{\lambda_k\} \Leftrightarrow \sigma(A^{-1}) = \left\{\frac{1}{\lambda_k}\right\} \Rightarrow$ $\hat{A}^{-1} \lvert b\rangle = \sum_i \frac{\beta_i}{\lambda_i} \lvert u_i\rangle \Rightarrow$ Messung des 1 Registers, nur $\lvert 1\rangle$ berücks.

HHL (Zusammenfassung)

HHL Schaltkreis gesamt		Achtung: $\|x\rangle$ muss noch gemessen werden! Exponentiell schneller in n (Größe der Matrix) Linear langsamer in s und κ (s -dünn und Kondition κ) Exponentiell langsamer in Genauigkeit ε Liefert nur den Quantenzustand $\|x\rangle$, nicht $\vec{x}$ selber $\Rightarrow$ - nutzbar als Zwischenergebnis, - oder aufwändige Messung notwendig

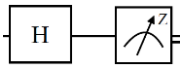
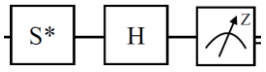
Linearkombinationen von Unitären Abbildungen

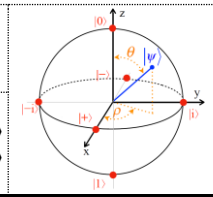
Hilfssatz	Seien $\hat{U},\hat{U}' \in \mathbb{C}^{2^n\times 2^n}$ unitäre Operationen, und $\Delta \overset{\text{def}}{=} \left\|\hat{U}-\hat{U}'\right\|_\sigma$ die Spektralnorm. Dann $\exists$ Algorithmus, der für $\forall\,|\Psi\rangle$ und $\forall\,\kappa \geq 0$ mit Wahrscheinlichkeit $p_E = 1-\frac{\kappa\Delta^2}{(\kappa+1)^2} \geq 1-\frac{4\kappa}{(\kappa+1)^2}$ das Ergebnis von $(\kappa\hat{U}+\hat{U}')\|\Psi\rangle$ berechnet.
Schaltkreis	 $\hat{V}_\kappa = \begin{pmatrix}\sqrt{\frac{\kappa}{\kappa+1}} & -\frac{1}{\sqrt{\kappa+1}}\\ \frac{1}{\sqrt{\kappa+1}} & \sqrt{\frac{\kappa}{\kappa+1}}\end{pmatrix}$ $\hat{V}_\kappa^\dagger = \begin{pmatrix}\sqrt{\frac{\kappa}{\kappa+1}} & \frac{1}{\sqrt{\kappa+1}}\\ -\frac{1}{\sqrt{\kappa+1}} & \sqrt{\frac{\kappa}{\kappa+1}}\end{pmatrix}$
	$\|0\rangle\|\Psi\rangle \xrightarrow{\hat{V}_\kappa\otimes\mathbb{1}} \left(\sqrt{\frac{\kappa}{\kappa+1}}\|0\rangle+\frac{1}{\sqrt{\kappa+1}}\|1\rangle\right)\|\Psi\rangle \xrightarrow{c_{[0]}\hat{U}} \sqrt{\frac{\kappa}{\kappa+1}}\|0\rangle\hat{U}\|\Psi\rangle+\frac{1}{\sqrt{\kappa+1}}\|1\rangle\|\Psi\rangle \xrightarrow{c\hat{U}'} \sqrt{\frac{\kappa}{\kappa+1}}\|0\rangle\hat{U}\|\Psi\rangle+\frac{1}{\sqrt{\kappa+1}}\|1\rangle\hat{U}'\|\Psi\rangle \xrightarrow{\hat{V}_\kappa^\dagger\otimes\mathbb{1}}$ $\sqrt{\frac{\kappa}{\kappa+1}}\left(\sqrt{\frac{\kappa}{\kappa+1}}\|0\rangle-\frac{1}{\sqrt{\kappa+1}}\|1\rangle\right)\hat{U}\|\Psi\rangle+\frac{1}{\sqrt{\kappa+1}}\left(\frac{1}{\sqrt{\kappa+1}}\|0\rangle+\sqrt{\frac{\kappa}{\kappa+1}}\|1\rangle\right)\hat{U}'\|\Psi\rangle = \boxed{\|0\rangle\left(\frac{\kappa}{\kappa+1}\hat{U}+\frac{1}{\kappa+1}\hat{U}'\right)\|\Psi\rangle+\|1\rangle\frac{\sqrt{\kappa}}{\kappa+1}(\hat{U}'-\hat{U})\|\Psi\rangle}$ Messen der Ancilla-Qbits mit Ergebnis $\|0\rangle$: Der Algorithmus liefert $\boxed{\|\Psi\rangle \mapsto \frac{1}{\kappa+1}(\kappa\hat{U}+\hat{U})}$ wobei $p_E = p(\|0\rangle) \geq 1-\frac{4\kappa}{(\kappa+1)^2}$

Skalarprodukt

$\mathrm{Re}(\langle x\|y\rangle)$	$\|\varphi_{\text{in}}\rangle = \frac{1}{\sqrt{2}}(\|0\rangle\|x\rangle+\|1\rangle\|y\rangle) \Rightarrow \|\varphi_{\text{out}}\rangle \overset{\text{def}}{=} \hat{H}\otimes\mathbb{1}\|\varphi\rangle = \frac{1}{\sqrt{2}}\left(\hat{H}\|0\rangle\|x\rangle+\hat{H}\|1\rangle\|y\rangle\right) = \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}(\|0\rangle+\|1\rangle)\|x\rangle+\frac{1}{\sqrt{2}}(\|0\rangle-\|1\rangle)\|y\rangle\right) \Rightarrow$ $\|\varphi_{\text{out}}\rangle = \frac{1}{2}\|0\rangle(\|x\rangle+\|y\rangle)+\frac{1}{2}\|1\rangle(\|x\rangle-\|y\rangle)$ Wahrscheinlichkeit $\|0\rangle$ zu messen: $p_{\text{re}} = \frac{1}{4}(\langle x\|+\langle y\|)(\|x\rangle+\|y\rangle) \Rightarrow$ $p_{\text{re}} = \frac{1}{4}(\langle x\|x\rangle+\langle y\|y\rangle+\langle x\|y\rangle+\langle y\|x\rangle) = \frac{1}{4}(2+\langle x\|y\rangle+\langle x\|y\rangle^*) = \frac{1}{4}(2+2\,\mathrm{Re}(\langle x\|y\rangle)) \Rightarrow \boxed{p_{\text{re}} = \frac{1}{2}(1+\mathrm{Re}(\langle x\|y\rangle))}$
$\mathrm{Im}(\langle x\|y\rangle)$	$\|\varphi_{\text{in}}\rangle = \frac{1}{\sqrt{2}}(\|0\rangle\|x\rangle+\|1\rangle\|y\rangle) \Rightarrow \|\varphi_{\text{out}}\rangle \overset{\text{def}}{=} (\hat{H}\otimes\mathbb{1})(\widehat{SZ}\otimes\mathbb{1})\|\varphi\rangle = \hat{H}\otimes\mathbb{1}\frac{1}{\sqrt{2}}(\|0\rangle\|x\rangle-i\|1\rangle\|y\rangle) = \frac{1}{\sqrt{2}}\left(\hat{H}\|0\rangle\|x\rangle-i\hat{H}\|1\rangle\|y\rangle\right) \Rightarrow$ $\|\varphi_{\text{out}}\rangle = \frac{1}{2}\|0\rangle(\|x\rangle-i\|y\rangle)+\frac{1}{2}\|1\rangle(\|x\rangle+i\|y\rangle)$ Wahrscheinlichkeit $\|0\rangle$ zu messen: $p_{\text{im}} = \frac{1}{4}(\langle x\|+i\langle y\|)(\|x\rangle-i\|y\rangle) \Rightarrow$ $p_{\text{im}} = \frac{1}{4}(\langle x\|x\rangle+\langle y\|y\rangle-i\langle x\|y\rangle+i\langle y\|x\rangle) = \frac{1}{4}(2-i\langle x\|y\rangle+i\langle x\|y\rangle^*) = \frac{1}{4}(2-2\,\mathrm{Im}(\langle x\|y\rangle)) \Rightarrow \boxed{p_{\text{im}} = \frac{1}{2}(1-\mathrm{Im}(\langle x\|y\rangle))}$

Pauli-Matrizen

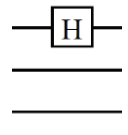
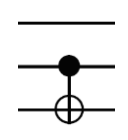
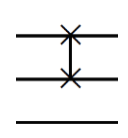
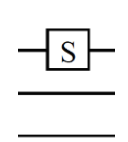
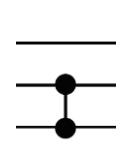
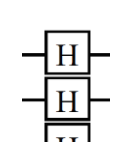
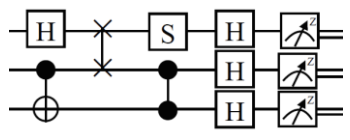
Pauli-Matrizen	$\hat{X} = \begin{pmatrix}0&1\\1&0\end{pmatrix}$ $\hat{Y} = \begin{pmatrix}0&-i\\i&0\end{pmatrix}$ $\hat{Z} = \begin{pmatrix}1&0\\0&-1\end{pmatrix}$ $\hat{I} = \begin{pmatrix}1&0\\0&1\end{pmatrix}$ hermitsch, unitär		Regeln	$\hat{X}\hat{X} = \hat{Y}\hat{Y} = \hat{Z}\hat{Z} = \hat{I}$ (involutorisch) $\hat{X}\hat{Y} = -\hat{Y}\hat{X}, \hat{X}\hat{Z} = -\hat{Z}\hat{X}, \hat{Y}\hat{Z} = -\hat{Z}\hat{Y}, \hat{Y} = i\hat{X}\hat{Z}$ $\sigma(.) = \{-1,1\}$ $\det(.) = -1$ $\mathrm{tr}(.) = 0$
$\{\hat{I},\hat{X},\hat{Y},\hat{Z}\}$ lin. unabhäng.	PM spannen $\mathbb{C}^{2\times2}$ auf: $\mathrm{L}\{\hat{I},\hat{X},\hat{Y},\hat{Z}\}=(\mathbb{C}^{2\times2},+,\cdot)$	$\hat{M} = \begin{pmatrix}m_{11}&m_{12}\\m_{21}&m_{22}\end{pmatrix} = a\hat{I}+b\hat{X}+c\hat{Y}+d\hat{Z} = \begin{pmatrix}a+d&b-ic\\b+ic&a-d\end{pmatrix}$	$a=\frac{1}{2}(m_{11}+m_{22})$ $c=\frac{i}{2}(m_{21}-m_{12})$ $b=\frac{1}{2}(m_{12}+m_{21})$ $b=\frac{1}{2}(m_{11}-m_{22})$	
hermitesche Matrizen	Sei $\hat{M}$ hermitesch $(\hat{M}=\hat{M}^\dagger) \Rightarrow$ $\hat{M}$ ist reelle LK von $\{\hat{I},\hat{X},\hat{Y},\hat{Z}\}$	Frobenius Skalarprod.	$\langle A,B\rangle_F \overset{\text{def}}{=} \sum_{i=1}^n\sum_{j=1}^n a_{ij}^*b_{ij}$ $\langle A,B\rangle_F = \mathrm{tr}(A^\dagger B) = \mathrm{tr}(AB^\dagger)$	mit $\langle\cdot,\cdot\rangle_F$ ist $(\mathbb{C}^{n\times n},+,\cdot)$ ein Hilbertraum mit $\langle\cdot,\cdot\rangle_F$ ist $\{\hat{I},\hat{X},\hat{Y},\hat{Z}\}$ eine OGB
Messen mit $\hat{X}$	$\hat{X}\|+\rangle = \|+\rangle$ $\hat{X}\|-\rangle = -\|-\rangle$ X-Basis: $\{\|+\rangle,\|-\rangle\}$	$\|+\rangle = \frac{1}{\sqrt{2}}(\|0\rangle+\|1\rangle)$ $\|-\rangle = \frac{1}{\sqrt{2}}(\|0\rangle-\|1\rangle)$		Messen von $\|0\rangle \mathrel{\widehat{=}} \|+\rangle$ Messen von $\|1\rangle \mathrel{\widehat{=}} \|-\rangle$
Messen mit $\hat{Y}$	$\hat{Y}\|+i\rangle = \|+i\rangle$ $\hat{Y}\|-i\rangle = -\|-i\rangle$ Y-Basis: $\{\|+i\rangle,\|-i\rangle\}$	$\|+i\rangle = \frac{1}{\sqrt{2}}(\|0\rangle+i\|1\rangle)$ $\|-i\rangle = \frac{1}{\sqrt{2}}(\|0\rangle-i\|1\rangle)$		Messen von $\|0\rangle \mathrel{\widehat{=}} \|+i\rangle$ Messen von $\|1\rangle \mathrel{\widehat{=}} \|-i\rangle$



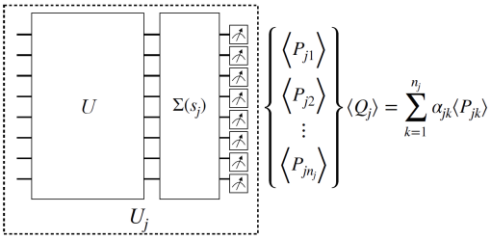
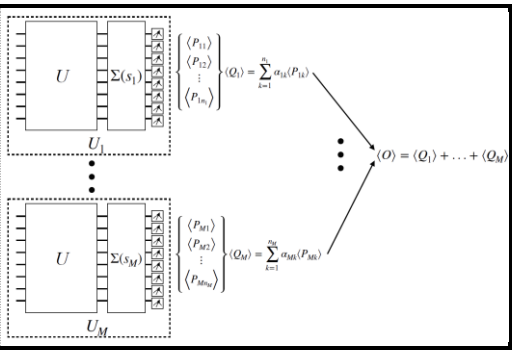
Pauli-Strings

Pauli-Strings	$\hat{P} = \hat{A}_1\otimes\ldots\otimes\hat{A}_n$ mit $\hat{A}_i \in \{\hat{I},\hat{X},\hat{Y},\hat{Z}\}$, $\hat{P}$ hermitesch, unitär — Menge aller $\hat{P}$: $\mathfrak{P}_n$ ist eine Basis von $(\mathbb{C}^{2^n\times2^n},+,\cdot)$
Basis von $\mathcal{H}$	Sei $\hat{M} \in \mathbb{C}^{2^n\times2^n}$ und $\hat{M}$ hermitesch $(\hat{M} \in \mathcal{H}) \Rightarrow \hat{M}$ ist reelle Linearkombination von Pauli-Strings
Zerlegung v. $\hat{O}$	Für $\hat{O} \in \mathcal{H} \Rightarrow \hat{O} = \frac{1}{2^n}\sum_{P\in\mathfrak{P}_n}\langle\hat{P},\hat{O}\rangle_F\hat{P} = \frac{1}{2^n}\sum_{P\in\mathfrak{P}_n}\mathrm{tr}(\hat{P}\hat{O})\,\hat{P}$, oder einfacher: $\hat{O} = \sum_k\alpha_k\hat{P}_k$ I.A. benötigt man $O(4^n)$ P-Strings
Erwartgs.wert	$\langle\hat{O}\rangle = \sum_i\alpha_k\langle\hat{P}_i\rangle$ Praktisch: partitionieren von $\{\hat{P}_i\}$ in gleichzeitig messbare (kommutierende) Pauli-Strings. Kommutierende Observable haben gemeinsame Eigenräume. Gleichzeitiges Messen = Projektion auf diese Eigenräume.
Kommutativität (QWK, AK)	Zwei Pauli-Strings $\hat{A} = \hat{A}_1\otimes\ldots\otimes\hat{A}_n$ und $\hat{B} = \hat{B}_1\otimes\ldots\otimes\hat{B}_n$ sind allg. kommutativ (AK), wenn $[\hat{A},\hat{B}] = 0$ Qbit-weise Kommutativität: (QWK): $[\hat{A}_i,\hat{B}_i] = 0$ für $1 \leq i \leq n$. QWK$\Rightarrow$ AK aber nicht zwingend umgekehrt! Pauli-Strings sind AK, genau dann wenn sie an einer geraden Zahl von Stellen nicht QWK sind.
Minimal kommutierende Partitionen	AK-Graph: $G = (\{\hat{P}_i\},E)$ mit $\{\hat{P}_i,\hat{P}_j\} \in E \Leftrightarrow [\hat{P}_i,\hat{P}_j] = 0$. Clique: Eine Teilmenge $C \subseteq \{\hat{P}_i\}$ heißt Clique $\Leftrightarrow$ Je zwei verschiedene Knoten von C sind durch Kanten aus E verbunden. AK-Partition S (Clique im AK-Graphen): Eine Partition S der Pauli-Strings $\{\hat{P}_1,\ldots,\hat{P}_m\}$ heißt AK-Partition, wenn die Pauli-Strings jedes Elements von S paarweise AK sind. Minimale AK-Partition: Bestimmung ist NP-schwer.

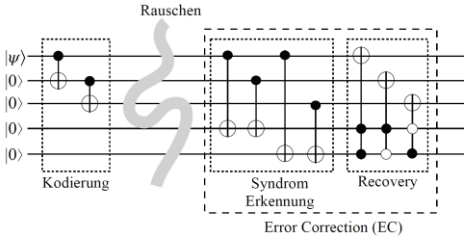
Stabilisator-Matrix

Zweck	Bestimmung eines Quantenschaltkreises zur Messung der Observablen, die gleichwertig ist zu den Observablen aller Elemente einer Clique im AK-Graphen.	
Erstellung	Enthält die Clique T m Pauli-Strings, die auf n Qbits wirken, dann hat die Stabilisator-Matrix $\widehat{M}(T)$ $2n$ Zeilen und m Spalten. Die obere Hälfte ist die „Z-matrix“, die untere Hälfte die „X-Matrix“. $T = \{\hat{P}_1, \dots, \hat{P}_m\} \subseteq \mathfrak{P}_n \Rightarrow \widehat{M}(T) \in \{0,1\}^{2x \times m}$. Sei $\hat{P}_i = \hat{A}_1 \otimes \dots \otimes \hat{A}_n \in T$. Dann: $\hat{A}_j = \hat{I} \Rightarrow S_{ji} = 0 \wedge S_{j+n,i} = 0$, $\hat{A}_j = \hat{Z} \Rightarrow S_{ji} = 1 \wedge S_{j+n,i} = 0$, $\hat{A}_j = \hat{X} \Rightarrow S_{ji} = 0 \wedge S_{j+n,i} = 1$, $\hat{A}_j = \hat{Y} \Rightarrow S_{ji} = 1 \wedge S_{j+n,i} = 1$	Beispiel: $\{IYX, ZZZ, XIX\}$ Z-Matrix X-Matrix $\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix}$
Regel A	$\hat{H}$-Transformation von $\widehat{M}(T)$: $\hat{H}$ auf Qbit i anwenden, vertauscht Zeilen i und $n+i$ Also: „Die i-te Zeile der Z-matrix und der X-Matrix wird vertauscht“	$\{IYX, ZZZ, XIX\}$ $\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix}$  $\{IYX, XZZ, ZIX\}$ $\begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix}$
Regel B	$\widehat{CX}$-Transformation von $\widehat{M}(T) = t_{ij}$: $\widehat{CX}_{\text{control,target}} \Rightarrow$ Für alle Spalten j von $\widehat{M}(T)$: $t_{\text{target,j}} \oplus t_{\text{control,j}} \mapsto t_{\text{control,j}}$ $t_{\text{target}+n,\text{j}} \oplus t_{\text{control}+n,\text{j}} \mapsto t_{\text{target}+n,\text{j}}$ Also: Control- und Target-Qbit modulo 2 addieren. Das Ergebnis in der Z-Matrix beim Control-Qbit eintragen, in der X-Matrix beim Target-Qbit.	$\{IYX, XZZ, ZIX\}$ $\begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix}$  $\{IYI, XIZ, ZIX\}$ $\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
Regel C	SWAP-Transformation von $\widehat{M}(T)$: SWAP_{ij} vertauscht Zeile i und Zeile j in Z- und X-Matrix	$\{IYI, XIZ, ZIX\}$ $\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$  $\{YII, IXZ, IZX\}$ $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
Regel D	$\hat{S}$-Transformation von $\widehat{M}(T)$: $\hat{S}$ auf Qbit i setzt das (i,i) Diagonalelement der Z-Matrix auf 0	$\{YII, IXZ, IZX\}$ $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$  $\{XII, IXZ, IZX\}$ $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
Regel E	CZ-Transformation von $\widehat{M}(T)$: CZ_{ij} setzt die (i,j) und (j,i)-Elemente der Z-Matrix auf 0	$\{XII, IXZ, IZX\}$ $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$  $\{XII, IXI, IIX\}$ $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
Regel F	$\hat{H}^{\otimes n}$-Transformation von $\widehat{M}(T)$: Anwenden von $\hat{H}^{\otimes n}$ vertauscht Z- und X-Matrix	$\{XII, IXI, IIX\}$ $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$  $\{ZII, IZI, IIZ\}$ $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$
Zusammen-setzen	Messen des oberen Qbits: Messung von IYX Messen des mittleren Qbits: Messung von ZZZ Messen des unteren Qbits: Messung von XIX	$\{IYX, ZZZ, XIX\}$ $\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix}$  $\{ZII, IZI, IIZ\}$ $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$
Gesamter Algorithmus	(1) Anwenden von Regel A, bis die Z-Matrix maximalen Rang $\min(m,n)$ hat. (2) Anwenden von Regel B und C, bis die X-Matrix die Einheitsmatrix ist. (3) Für jedes (i,i)-Diagonalelement der Z-Matrix, welches gleich 1 ist, Regel D anwenden. (4) Für jedes $(i+1,i)$-Element unter der Hauptdiagonale der Z-Matrix, welches gleich 1 ist, Regel E anwenden. (5) Anwenden von Regel F (6) Messung hinzufügen.	

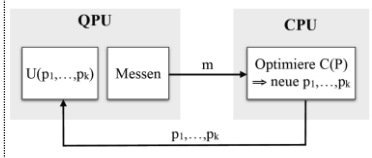
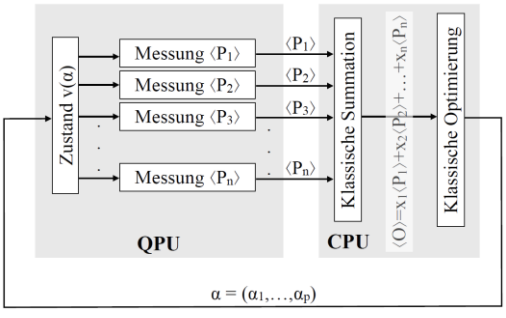
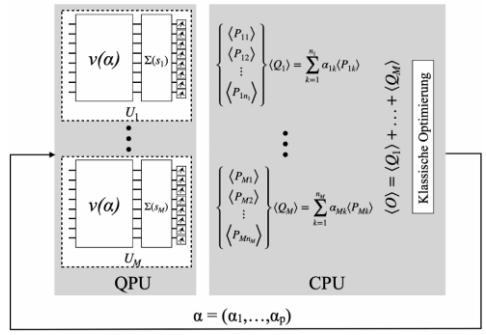
Messen von Cliquen und der gesamten Observable

Messen der Clique:		Gesamt	

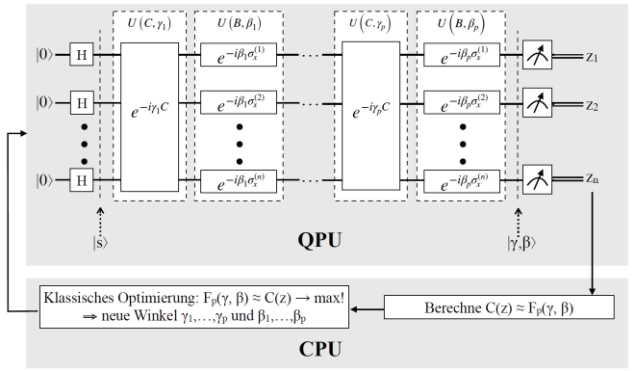
Quanten-Rauschen

Dekohärenz	Übergang eines reinen Zustands in einen gemischten Zustand durch Wechselwirkung mit der Umgebung über die Zeit		
Relax. time	T_1 … Übergang in orthogonalen Zustand	Dephasing time:	Kleine Störung, zufällige Änderung der Phase
Error-Correction	No-Cloning: Kopieren eines Quantenzustandes nicht möglich Aber: Error-Correction möglich, z.B. 9 physische Qbits codieren ein logisches QBit $\|0\rangle \mapsto \frac{(\|000\rangle+\|111\rangle)\otimes(\|000\rangle+\|111\rangle)\otimes(\|000\rangle+\|111\rangle)}{2\sqrt{2}}$ $\|1\rangle \mapsto \frac{(\|000\rangle-\|111\rangle)\otimes(\|000\rangle-\|111\rangle)\otimes(\|000\rangle-\|111\rangle)}{2\sqrt{2}}$		
Gate Infidel.	Keine exakten Rotationen auf Bloch-Kugel. Der akkumulierte Fehler wächst linear mit der Länge der Rechnung. $wd \ll \frac{1}{\varepsilon}$		
Threshold Th.	Für jede vorgegebene Genauigkeit gibt es eine Implementierung mit fehlertoleranten Gattern, falls Fehlerrate klein genug.		

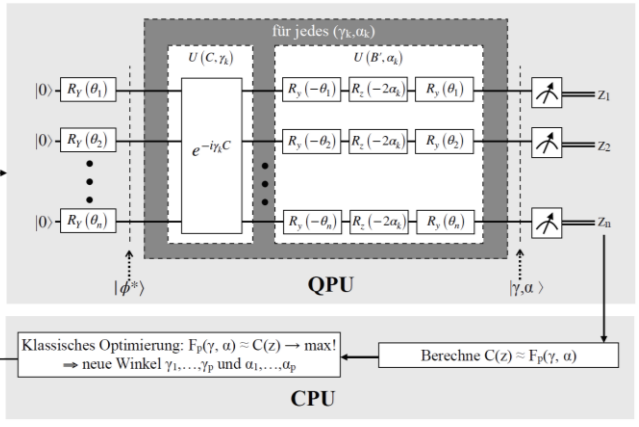
Hybride und Variationelle Algorithmen (VQA)

Hybrid	Aufgeteilt auf QPU und CPU	VQA:	 Ansatz U sollte geringe Tiefe haben, so dass $T_U < T_D$. Messung sollte effizient sein, so dass $T_U + T_M < T_D$ Gut für NISQ (Noisy Intermediate-Scale Quantum Computers) Kostenfunktion $C(\vec{P})$ codiert Lösung des Problems Ansatz $\hat{U}(\vec{P})$: Unitärer Operator, schätzt C ab Optimierung auf CPU mit $\vec{P}_{\text{opt}} = \underset{P}{\text{argmax}}\, C(\vec{P})$
Rayleigh-Ritz	$\lambda_{\min} = \min(\sigma(\hat{O})) \leq \langle\Psi\|\hat{O}\|\Psi\rangle = \langle\hat{O}\rangle_{\|\Psi\rangle}$ (Ann: $\|\|\Psi\rangle\| = 1$)		
Variational Quantum Eigensolver (VQE)	Wähle Testvektoren $\vec{v}(\vec{\alpha})$, so dass $\langle\hat{O}\rangle_{\|v(\vec{\alpha})\rangle} = \underbrace{\sum_i \sum_j \alpha_{ij} \langle P_{ij}\rangle_{\|v(\vec{\alpha})\rangle}}_{\text{Pauli-String Zerlegung}}$ minimal wird mittels klassischer Optimierung von $\vec{\alpha}$ auf der CPU. Dann ist $\langle\hat{O}\rangle_{\|v(\vec{\alpha})\rangle} \approx \lambda_{\min}$ und $\frac{\vec{v}(\vec{\alpha})}{\|\vec{v}(\vec{\alpha})\|}$ die Näherung des zugehörigen Eigenvektors.		
VQE Schaltkreis, vereinfacht:		Tat-sächlich:	
Anwendung: Schnitte als Basisvektoren	Sei $G = (K, E)$ ein Graph, bestehend aus Knoten K und Kanten E, mit $\text{card}(K) = n$ (Anzahl der Knoten). Sei $\{S, T\}$ ein Schnitt von K. $(x_1, \dots, x_n) \in \{0,1\}^n$ ist definiert als: $x_i = 1 \Leftrightarrow i \in S$ und $x_i = 0 \Leftrightarrow i \in T$. Jeder Schnitt $\{S, T\}$ wird so abgebildet auf $x_{ST} \in \{0,1\}^n$, was wiederum in der Berechnungsbasis $\{\|x_1, \dots, x_n\rangle\}$ ausgedrückt werden kann. Sei g die Anzahl Kanten zwischen Knoten aus der gleichen Menge (in S oder T) und u die Anzahl der Kanten zwischen Knoten unterschiedlicher Mengen ($S \leftrightarrow T$). Für die entsprechende Partition $x \in \{0,1\}^n$ ist $w_x \overset{\text{def}}{=} \frac{1}{2}(g - u)$.			
Max Cut als Eigenwert-problem	Definiere Diagonalmatrix $M = \text{diag}(m_{ii})$ mit $m_{ii} = w_i$. M ist hermitesch, da diagonal. Sei $\lambda_{\min} = \min(\sigma(M))$. Dann gilt: S, T ist ein maximaler Schnitt von $G \Leftrightarrow \|x_{ST}\rangle$ ist Eigenvektor von $\lambda_{\min}$ $\Rightarrow$ Max Cut lässt sich mit VQE lösen!		

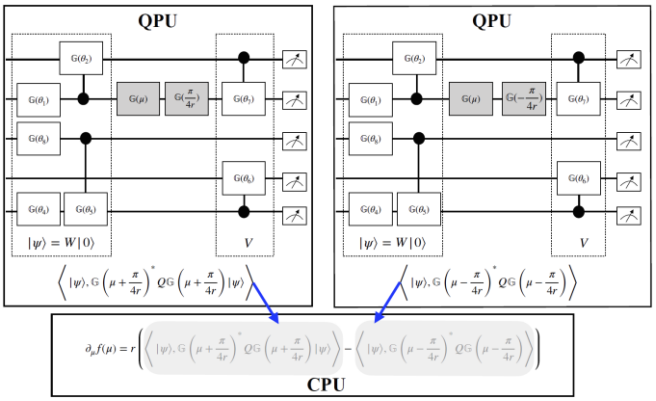
Quantum Approximate Optimization Algorithm (QAOA)

QUBO, Klauseln	Sei $\{S,T\}$ ein Schnitt von K. $(x_1,\dots,x_n)\in\{0,1\}^n$ ist definiert als: $x_i=1\Leftrightarrow i\in S$ und $x_i=0\Leftrightarrow i\in T$. Dann wird ein maximaler Schnitt definiert durch folgende (zu maximierende) Kostenfunktion: $\mathrm{C}(x)=\frac{1}{2}\sum_{ij}\mathrm{C}_{ij}(x)$ mit der Klausel $\mathrm{C}_{ij}(x)=x_i(1-x_j)+x_j(1-x_i)$. QUBO: Quadratic Unconstrained Binary Optimization
Verallgemein.	Allgemeine Klausel $\mathrm{C}_\alpha(z):\mathbb{B}^n\to\mathbb{B}$ mit Kostenfunktion $\mathrm{C}(z)=\sum_\alpha \mathrm{C}_\alpha(z)\Longrightarrow$ Kombinatorisches Optimierungsproblem
Kostenoperator	$\hat{\mathrm{U}}(C,\gamma)=e^{-i\gamma\,\mathrm{C}(z)}\Longrightarrow$ Dreht die Phase umso mehr, je mehr Klauseln in der Kostenfunktion erfüllt sind. Unitär.
Mixer $\hat{\mathrm{U}}(\hat{B},\beta)$	$\hat{\mathrm{B}}=\sum_{j=1}^{n}\sigma_x^{(j)}$ mit $\sigma_x^{(j)}\overset{\text{def}}{=}I\otimes\dots I\otimes\underbrace{\sigma_x}_{j}\otimes\dots\otimes I$. Hermitesch. Kovertiert Phasendrehungen in Amplituden. Mixer (unitär): $\hat{\mathrm{U}}(\hat{B},\beta)=e^{-i\beta\hat{B}}$ mit $\beta\in[0,\pi]$. Weil $\left[\sigma_x^{(i)},\sigma_x^{(j)}\right]=0$ ist Trotter-Entwicklung möglich: $\boxed{\hat{\mathrm{U}}(\hat{B},\beta)=\prod_{j=1}^{n}e^{-i\beta\sigma_x^{(j)}}}$ Weil $e^{-i\beta\sigma_x^{(j)}}=\hat{R}_x(2\beta)$, rotiert $\hat{\mathrm{U}}(\hat{B},\beta)$ jedes Qbit im Register um 2β.
Quantum Approximate Optimization Algorithm QAOA	 wähle Hyperparameter $p\in\mathbb{N},p\geq 1$, wähle $\vec{\gamma}=(\gamma_1,\dots,\gamma_p)\in[0,2\pi]^p$ wähle $\vec{\beta}=(\beta_1,\dots,\beta_p)\in[0,\pi]^p$ erzeuge $\lvert s\rangle=\hat{H}^{\otimes n}\lvert 0\rangle^{\otimes n}=\frac{1}{\sqrt{2^n}}\sum_{z=0}^{2^n-1}\lvert z\rangle$ $\lvert\vec{\gamma},\vec{\beta}\rangle=\prod_{i=1}^{p}\hat{\mathrm{U}}(\hat{B},\beta)\,\hat{\mathrm{U}}(C,\gamma)\,\lvert s\rangle$ mit $\lvert\vec{\gamma},\vec{\beta}\rangle$ messe $z\in\mathbb{B}^n$ berechne $\mathrm{F}_p(z)\approx\max(C)$ für „große“ p wenn nötig: berechne neue Winkel $\vec{\gamma},\vec{\beta}$ mittels klassischer Optimierung, um $\mathrm{F}_p(z)\approx\mathrm{C}(z)$ zu maximieren
Dichte α	$\alpha=\frac{K}{N}$ mit K … Anzahl Klauseln, N … Anzahl Variablen. Für eine gegebene Tiefe p gibt es ein α_p, so dass für Probleme mit Dichte $\alpha>\alpha_p$ keine Lösung mit QAOA in akzeptabler Zeit gefunden wird. $\Longrightarrow$ Einsatzbereich von QAOA problemabhängig!
Vorteile	Jeder Quantenalgorithmus kann mit QAOA approximiert werden (auch auf NISQ-Maschinen!)

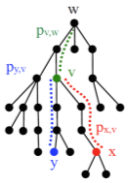
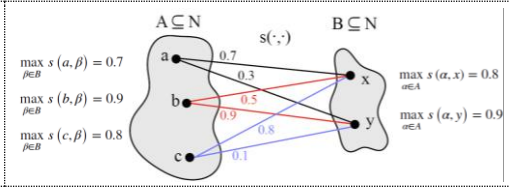
Warm Start – QAOA (WS-QAOA)

Güte	Sei $\mathcal{P}$ ein NP-Problem, und P eine Instanz von $\mathcal{P}$ mit Problemgröße N. Sei $\mathrm{OPT}(P)$ der Wert der optimalen Lösung. Sei ALG ein polynomialer Approximationsalgorithmus für $\mathcal{P}$, und $\mathrm{ALG}(P)$ die von ALG berechnete Lösung. Dann gilt für die Güte (den Approximationsfaktor) C: $\mathrm{ALG}(P)\geq\mathrm{OPT}(P)\,/C$ für Maximierungsprobleme, bzw $\mathrm{ALG}(P)\leq C\,\mathrm{OPT}(P)$ für Minimierungsprobleme
UGC	Unique Games Conjecture (UGC):Jede Kante eines Graphen hat der mögliche Paare von Farben für die Knoten vorgegeben. “Einzigartiges Spiel“: Man muss für jede Kante ein Paar wählen, sodass der Graph konsistent gefärbt wird. Vermutung: Die Lösung ist NP-schwer. Unter der Voraussetzung, dass UGC wahr ist, lassen sich die besten Approximationsfaktoren vieler Probleme bestimmen. $\Longrightarrow$ Die Güte C bestimmt ein Intervall rund um die optimale Lösung $\mathrm{OPT}(P)$, die mit einem klassischen polynomialer Approximationsalgorithmus $\mathrm{ALG}(P)$ nicht erreicht werden kann.
Verschränkg.	Wenn Verschränkung ausgenutzt wird, ist UGC falsch! Man kann mit $\mathrm{ALG}(P)$ näher an die optimale Lösung herankommen.
Warm Start QAOA	 wähle Hyperparameter $\varepsilon\in\mathbb{R},0\leq\varepsilon\leq\frac{1}{2}$ wähle $\vec{\gamma}=(\gamma_1,\dots,\gamma_p)\in[0,2\pi]^p$ wähle $\vec{\alpha}=(\alpha_1,\dots,\alpha_p)\in[0,\pi]^p$ starte m. Näherung $\lvert s\rangle=\otimes_{i=1}^{n}\hat{R}_y(\theta_i)\,\lvert 0\rangle^{\otimes n}$ wobei $\theta_i=\begin{cases}2\arcsin\sqrt{c_i^*} & \text{für } c_i^*\in[\varepsilon,1-\varepsilon]\\ 2\arcsin\sqrt{\varepsilon} & \text{für } c_i^*\leq\varepsilon\\ 2\arcsin\sqrt{1-\varepsilon} & \text{für } c_i^*\geq 1-\varepsilon\end{cases}$ und $c_i^*\in[0,1]$ (stetige Relaxation) Modifizierter Mixer: $\hat{\mathrm{U}}(\hat{B}',\alpha_i)=\prod_{j=1}^{n}e^{-i\alpha_i\beta_j}=$ $\prod_{j=1}^{n}\hat{R}_y(\theta_j)\,\hat{R}_z(-2\alpha_i)\,\hat{R}_y(-\theta_j)$ mit $B'=\sum_{i=1}^{n}B_i$, $B_i\overset{\text{def}}{=}I\otimes\dots I\otimes\underbrace{\beta_i}_{i}\otimes\dots\otimes I$ und $\beta_i=\begin{pmatrix}2c_i^*-1 & -2\sqrt{c_i^*(1-c_i^*)}\\ -2\sqrt{c_i^*(1-c_i^*)} & 1-2c_i^*\end{pmatrix}$
Karge Hochebenen	Ist die Kostenfunktion C global, kann es immer zu „kargen Hochebenen“ kommen, egal ob U tief oder flach ist. Nur wenn U flach ist mit $O(\log(n))$ und die Kostenfunktion C lokal ist, gibt es keine kargen Hochebenen.

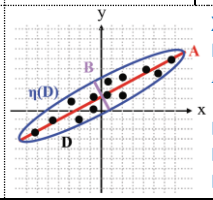
Berechnung von Gradienten mittels Quantenschaltkreisen

Annahme	$\hat{U}(\vec{\theta})$ ist eine Komposition von (kontrollierten) Ein-Parameter-Gattern $\mathbb{G}$ und $\mu \in \{\theta_1, \dots, \theta_n\}$ ist der Parameter eines einzelnen Gatters $\mathbb{G}(\mu)$, dann lässt sich $\hat{U}(\vec{\theta})$ zerlegen in $\hat{U}(\vec{\theta}) = \hat{V}\,\mathbb{G}(\mu)\,\hat{W}$
Berechnung des Gradienten	 Erweitere $\hat{U}(\vec{\theta}) = \hat{V}\,\mathbb{G}(\mu)\,\hat{W}$ zu $\hat{U}_1(\vec{\theta}) = \hat{V}\,\mathbb{G}(\mu)\,\mathbb{G}\left(\frac{\pi}{4r}\right)\hat{W}$ und $\hat{U}_2(\vec{\theta}) = \hat{V}\,\mathbb{G}(\mu)\,\mathbb{G}\left(\frac{\pi}{4r}\right)\hat{W}$ Ausführen von $\hat{U}_1(\vec{\theta})$ und $\hat{U}_2(\vec{\theta})$ Subtraktion der Eigenwerte auf der CPU Multiplikation mit r Damit hat man $\partial_\mu f(\mu)$ für ein $\mu \in \{\theta_1, \dots, \theta_n\}$ Wiederholung für jedes $\mu \in \{\theta_1, \dots, \theta_n\}$ → liefert $\nabla f(\mu)$

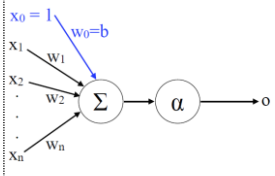
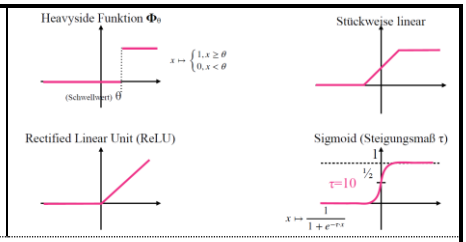
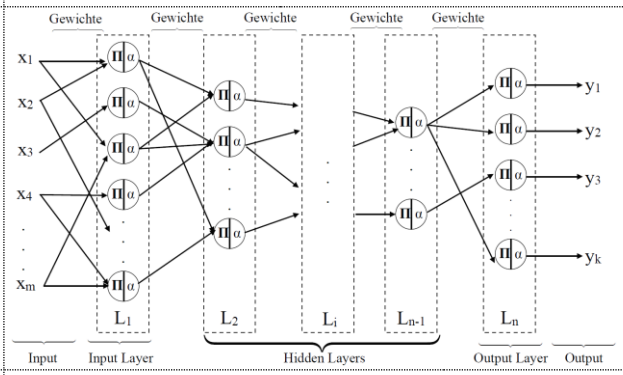
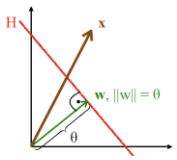
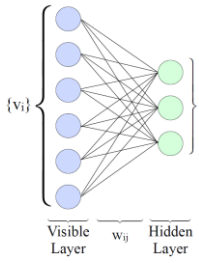
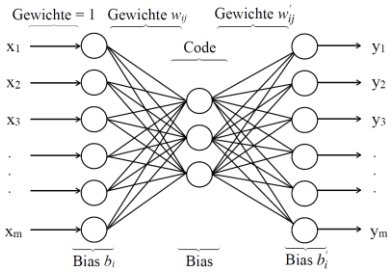
Kategoriale Daten

Nominale Dat.	Endlicher Wertevorrat, ohne natürliche Ordnung (z.B. Farbe: {rot, grün, blau}). Nur Vergleich auf Gleichheit möglich.
Ordinale Daten	Endlicher Wertevorrat, mit natürlicher Ordnung (z.B. Kleidergröße: {S, M, L, XL}. Auch größer/kleiner-Vergleich möglich.
Abb num Daten	z.B. rot ↦ 1, grün ↦ 2, blau ↦ 3. Problem: Farbe rot<grün macht keinen Sinn (Kleidergröße S↦1 < M↦2 dagegen schon).
1-aus-n-Code	z.B. rot ↦ $(1,0,0)$, grün ↦ $(0,1,0)$, blau ↦ $(0,0,1)$. Problem: Dimension des Featureraums sehr groß, Vektoren äquidistant
Einbettung	Einbettung φ ist Abb. Eines hoch-dim. Boolschen Featureraums $\mathbb{B}^n$ in den niedrigerdimensionalen Vektorraum $\mathbb{R}^d$, $d \ll n$
Ähnlichkeits-maß $s(i,i)$	Sei M eine endlich Menge, dann ist $s: M \times M \to \mathbb{R}$ ein Ähnlichkeitsmaß, wenn $\forall i,j \in M$: (i) $s(i,j) = s(j,i)$ (symmetrisch) (ii) $s(i,i) \geq s(i,j)$ (maximal ähnlich) (iii) $s(i,j) > 0$ (positiv) (iv) $s(i,i) = 1$ (selbst-identisch) Def. macnhmal ohne (iii) und (iv)
Distanzmaß $t(i,j)$	Auch: Unähnlichkeit. Erfüllt i.A. nicht die Dreiecksungleichung. Sei M eine endlich Menge, dann ist $t: M \times M \to \mathbb{R}$ ein Distanzmaß, wenn $\forall i,j \in M$: (i) $t(i,j) > 0$ (positiv) (ii) $t(i,j) = 0 \Leftrightarrow i = j$ (definit) (iii) $t(i,j) = t(j,i)$ (symmetrisch)
Metrik	Zusätzlich zu den Regeln für das Distanzmaß: (iv) $t(i,j) \leq t(j,k) + t(k,j)$ (Dreiecksungleichung)
Ähnl→Distanz	$t(i,j) = \sqrt{s(i,i) + s(j,j) - 2\,s(i,j)}$ | Distanzmaß→Ähnlichkeitsmaß | $s(i,j) = \frac{1}{1+t(i,j)}$
MDS	Multi-Dimensionale Skalierung. Abbildung $\varphi(M,t) \to (\mathbb{R}^n, d)$, sodass $d_{ij} = d(\varphi(i), \varphi(j)) \approx t(i,j) = t_{ij}$ Spezialfall: $\varphi: M \subseteq \mathbb{R}^m \to \mathbb{R}^n$ mit $n \ll m$ … Feature Reduction
Verlustfunk-tion (Stress)	Menge der Bilder $X = \varphi(M)$ heißt Konfiguration. Finden der optimalen Konfiguration durch Minimieren einer Verlustfunkt. Raw Stress: $\sigma_r(X,a) = \sum_{i<j}(d_{ij} - at_{ij})^2$ Norm. Stress: $\sigma_n(X,a) = \sum_{i<j}\frac{(d_{ij}-at_{ij})^2}{a^2 t_{ij}^2}$ Kruskal Stress: $\sigma_n(X,a) = \sum_{i<j}\frac{(d_{ij}-at_{ij})^2}{d_{ij}^2}$
Wu-Palmer 	Ähnlichkeitsmaß für kategoriale Daten, die durch Taxonomie strukturiert sind, d.h. MDS anwendbar $\omega(x,y) = \frac{2\,L(p_{v,w})}{L(p_{x,v})+L(p_{y,v})+2\,L(p_{v,w})} = \frac{2\,L(p_{v,w})}{L(p_{x,w})+L(p_{y,w})}$ mit $p_{a,b}$ … Pfadlänge von Knoten a zu Knoten b w …Wurzelknoten v… Nächster gemeinsamer Vorgänger von x, y
Ähnlichkeit mengen-werter Eigenschaften	 $\sigma(A,B) = \frac{1}{2}\left(\frac{1}{\text{card}(A)}\sum_{a\in A}\max_{b\in B} s(a,b) + \frac{1}{\text{card}(A)}\sum_{b\in B}\max_{a\in A} s(a,b)\right)$ Beispiel: $=\frac{1}{2}\left(\frac{1}{3}(0.7+0.9+0.8) + \frac{1}{2}(0.8+0.9)\right) \approx 0.825$
Tupel	Vergleich von Tupeln mengenwertiger Eigenschaften: $\mu(\mathcal{A},\mathcal{B}) = \frac{1}{m}\sum_{i=1}^{m}\sigma_i(A_i,B_i)$

Dimensionsreduktion

Zufallsvariable	Ist keine Variable, sondern eine Abbildung, die jedem Ereignis eines Zufallsexperiments eine relle Zahl zuordnet $X: \Omega \to \mathbb{R}$		
beobachteter Mittelwert	N Versuche, jedes der n möglichen Ereignisse $\omega_i \in \Omega$ wird dabei h_i mal beobachtet. Mittelwert $\mathrm{m}(X) = \frac{1}{N}\sum_{i=1}^{N} X(\omega_i)\, h_i = \sum_{i=1}^{N} X(\omega_i)\frac{h_i}{N} \approx \sum_{i=1}^{N} X(\omega_i)\, P(\omega_i)$ weil $\frac{h_i}{N} \approx P(\omega_i)$	Erwartgs-wert E(X)	$E(X) = \sum_{\omega\in\Omega} X(\omega)\, P(\omega)$ $\mathrm{m}(X) \xrightarrow{N\to\infty} E(X)$
Varianz 1D	$\mathrm{var}(X) = \mathrm{E}((X - \mathrm{E}(X))^2) = \mathrm{E}(X^2) - \mathrm{E}(X)^2 \approx \mathrm{m}((X - \mathrm{m}(X))^2) = \frac{1}{N}\sum_{i=1}^{N}(x_i - m)^2$	Std.Abw. (Streuung)	$\sigma = \sqrt{\mathrm{var}(X)}$
Varianz 2D	$\sigma_x^2 \approx \frac{1}{N}\sum_{i=1}^{N}(x_i - m_x)^2,\ \sigma_y^2 \approx \frac{1}{N}\sum_{i=1}^{N}(y_i - m_y)^2$	Kovarianz	$\mathrm{cov}(G,H) \approx \frac{1}{N}\sum_{i=1}^{N}(g_i - m_G)(h_i - m_H)$ mit $G = \{g_i\}$, $H = \{h_i\}$
Kovarianz Matrix 2D	$\Sigma(D) = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{yx} & \sigma_y^2 \end{pmatrix}$	Kovarianz Matrix allg.	$\Sigma(D) = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_n^2 \end{pmatrix}$ symmetrisch $\Rightarrow$ alle $\lambda_i \in \mathbb{R}$ pos. semidefinit diagonalisierbar
PCA Principal Component Analysis		$\Sigma(D)$ beschreibt einen Ellipsoiden (eine ellipsoide Quadrik), wobei die Eigenvektoren von $\Sigma(D)$ die Hauptachsenrichtungen bestimmen, und die zugehörigen Eigenwerte die Länge der Halbachsen. D liegt im Inneren des durch $\Sigma(D)$ beschrieben Ellipsoiden. Die längsten Halbachsen bestimmen die Hauptkomponenten (Principal Components) Projektion der Punkte aus D auf die Hyperebene der Hauptkomponenten („Feature Extraction“) Beispiel links: Die Information von D ist im Wesentlichen in der Projektion auf A enthalten.	

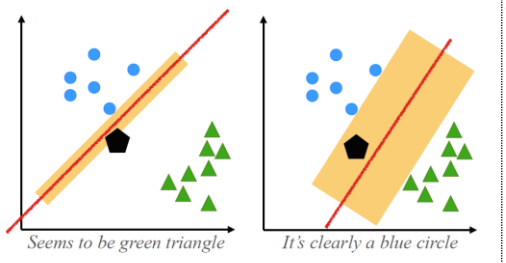
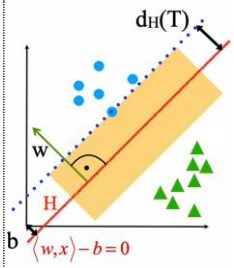
Neuronale Netze

Neuron		$\sum_{i=1}^{n} w_i x_i$ … Propagierungsfunktion alternativ: $\prod_{i=1}^{n} w_i x_i$ oder $\max_i(w_i x_i)$ $w_0 = b$ … Schwellwert (Bias) $\alpha: \mathbb{R} \to \mathbb{R}$ … Aktivierungsfunktion Beispiele: Siehe rechts - monoton steigend - stückweise C^1 (stetig diff.bar) - Nichlinearität	
Neuronales Netz		Als Funktion: $\vec{y} = \mathrm{F}(\vec{x}; \{w_{ij}\}; \{b_r\})$ Lernen: iteratives Anpassen der Gewichte $\{w_{ij}\}$ und der Schwellwerte $\{b_r\}$, sodass $\mathrm{F}(\vec{x}_i) = \vec{Y}_i$ bestmöglich alle $\vec{y}_i$ approximiert. Loss Function: $\mathrm{L}(w,b) = \sum_i (\vec{y}_i - \vec{Y}_i)^2$ Lösung mit Back-Propagation, Gradienteverfahren, Belder-Mead, …	
Perzeptron	Ein Neuron $\mathrm{v}(\vec{x}) = \Theta_\theta \sum_{i=1}^{n} w_i x_i$ mit der Heavyside-Funktion Θ_θ als Aktivierungsfunktion heißt Perzeptron. $\mathrm{v}(\vec{x}) = 1 \Leftrightarrow \vec{x}$ liegt „rechts“ von der durch $\vec{w}$ definierten Hyperebene H (siehe rechts) Geeignet für Klassifikation lin. seperabler Daten		Sei $T = \{(\vec{t}_1, d_1), \ldots, (\vec{t}_n, d_n)\}$ … Trainingsdaten mit $d_j \in \{0,1\}$ wähle für $\vec{w}(0)$ zufällige (kleine) Werte berechne $\mathrm{v}(\vec{x})$, berechne neues $w_i(t+1) = w_i(t) + r(d_j - y_i(t))x_{ij}$ für zufälliges j wiederholen, bis $\mathrm{L}(w) < \varepsilon$
RBM Restricted Boltzmann Maschine	Neuronales Netz mit 2 Layern. Kanten nur zwischen Knoten verschiedener Layer. Keine Verbindungen innerhalb der Layer („restricted“). Jeder Knoten des Input-layers ist mit allen Knoten Des „hidden“-Layers verbunden. Geeignet für Klassifikation oder Feature Selection. Neuronen des RBM nehmen nur boolsche Werte an.		Zwei gestapelte RBM ergeben einen Auto-encoder. Input-Layer mit Hidden Layer: Encoder. Hidden Layer mit Output Layer: Decoder. 

No Free Lunch Theorem

No free lunch	Seien A, B endliche Mengen, und B linear geordnet („es gibt ein größtes Element in B"). Sei F_{AB} die Menge aller Abbildungen $f: A \to B$. Für ein beliebiges $f: A \to B$ wird das Maximum $\text{argmax}(f)$ gesucht. Sei H eine beliebige Suchheuristik zur bestimmung von $\text{argmax}(f)$ für eine Funktion f aus F_{AB}, und $r(H)$ die durchschnittliche Laufzeit von H über alle Funktionen f aus F_{AB}. Dann ist $r(H)$ unabhängig von $H, d.h. \forall H_1, H_2: r(H_1) = r(H_2)$ „Kein Algorithmus, der beliebige (Black-Box) Optimierungsprobleme löst, ist besser als irgend ein anderer solcher Algorithmus."
Supervised learning	Sei $S = \{(x_1, y_1), \dots, (x_n, y_n)\} \subseteq \mathcal{X} \times \mathcal{Y}$ die Menge der Trainingsdaten. S wurde erzeigt von einer unbekannten Funktion $f: \mathcal{X} \to \mathcal{Y}$. Aufgabe: Die unbekannte Funktion f ist aus den Trainingsdaten zu bestimmen. Gesucht ist die Hypothese $h(x_i) = y_i$ für $1 \le i \le n$. Die Erwartung ist, dass dann $f = h$ gilt. Ein Lernalgorithmus $\mathcal{A}$ berechnet so eine Hypothese.
Risiko	Die Verlustfunktion $L: \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ misst, wie weit $h(x_i)$ durchschnittlich vom den tatächlichen y_i abweicht. Je kleiner, der durchschnittliche Verlust (Risiko, out-of-sample-error, Generalisierungsfehler) $L(y, h(x))$, desto besser ist h.
In-sample-error	Durchschnittlicher Verlust auf den Trainingsdaten (in-sample-error) $\hat{R}(h) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, h(x_i))$. Für endliche $\mathcal{Y}$ gibt es immer ein h, das $\hat{R}$ minimiert
Classical No-Free-Lunch Theorem in Supervised Learning	Seien S die Trainingsdaten, sei $\lvert\mathcal{X}\rvert < \infty, \lvert\mathcal{Y}\rvert < \infty$ und $\lvert\mathcal{X}\rvert > \lvert S\rvert$. Für $f \in F_{\mathcal{X}\mathcal{Y}}$ definiere das Risiko $R_f(h) = \sum_{x \in \mathcal{X}} \mathrm{X}(x)\, \mathbb{P}(h(x) \neq f(x))$. Dann ist für jeden Lernalgorithmus $\mathcal{A}$ das durchschnittliche Risiko, gemittelt über alle $f \in F_{\mathcal{X}\mathcal{Y}}$ $\mathbb{E}_f\left(\mathbb{E}_S(R_f(h_S))\right) \ge \left(1 - \frac{1}{\lvert\mathcal{Y}\rvert}\right)\left(1 - \frac{\lvert S\rvert}{\lvert\mathcal{X}\rvert}\right)$ Interpretation: Wenn keine Traningsdaten ($\lvert S\rvert = 0$) $\Longrightarrow$ Risiko $1 - 1/\lvert\mathcal{Y}\rvert$ („bloßes Raten") $\lvert\mathcal{Y}\rvert \to \infty \Longrightarrow$ Risiko wird groß, weil es immer mehr Funktionen gibt, die man raten muss Festes $\lvert\mathcal{Y}\rvert$, viele Testdaten $\lvert S\rvert \to \lvert\mathcal{Y}\rvert \Longrightarrow$ Durchschnittliches Risiko wird beliebig klein
Quantum No-Free-Lunch Theorem	Durchschnittliches Risiko gemittelt über alle unitären Transformationen U und Traininsgdaten S $\mathbb{E}_U\left(\mathbb{E}_S\left(R_U\left(\hat{V}_S\right)\right)\right) \ge 1 - \frac{r^2 n^2 + d + 1}{d(d+1)}$ r … Schmidt-Rang der Traingsdaten, n… Anzahl Trainingsdatensätze, d …Dim. Hilbertraum Interpretation: - Sind die Traininsdaten unverschränkt, verschwindet das Risiko nur für $n = d$ und $d \to \infty$. Man benötigt dann wegen $d = 2^m$ exponentiell viele Trainingsdaten. - Sind aber die Traningsdaten maximal verschränkt ($r = d$), dann geht das Risiko für $d \to \infty$ bereits für $n = 1$ gegen Null! Ein einziger, maximal verschränkter Trainingsdatensatz hoher Dimension genügt, um eine unitäre Transformation mit geringem Risio zu lernen.

Support Vector Machines

Ziel	 Wir suchen nach einer Hyperebene H, die zwei Klassen von Datenpunkten „am besten" trennt. Positionierung der Hyperebene, so das sie einen „maximalen Rand" ohne Datenpunkte hat.  Training Data $T = \{(\vec{x}_1, y_1), \dots, (\vec{x}_n, y_n)\}, \vec{x}_i \in \mathbb{R}^g, y_i \in \{-1, 1\}$. Hyperebene: $\{x: \langle \vec{x}, \vec{w}\rangle - b = 0\}$. Klassifizierung der Datenpunkte mit $\text{sign}(\langle x, w\rangle - b)$. Problem: Finde Hyperebene, so dass $\mathrm{d}_H(T)$ maximiert wird, wobei $\mathrm{d}_H(T) = \min\{\mathrm{d}_H(\vec{x}_i)\}$
SVM Optimierung	$\left.\begin{matrix} y_i = 1 \implies d_H(\vec{x}_i) \ge d_H(T) \\ y_i = -1 \implies -d_H(\vec{x}_i) \ge d_H(T) \end{matrix}\right\} \Longrightarrow y_i\, d_H(\vec{x}_i) \ge d_H(T) \Longrightarrow y_i(\langle \vec{x}_i, \vec{w}\rangle - b) \ge d_H(T) \mid : d_H(T) \Longrightarrow y_i\left(\langle \vec{x}_i, \frac{\vec{w}}{d_H(T)}\rangle - \frac{b}{d_H(T)}\right) \ge 1$ Redefinition: $\vec{w} \to \frac{\vec{w}}{d_H(T)}$ und $b \to \frac{b}{d_H(T)} \Longrightarrow \boxed{y_i(\langle \vec{x}_i, \vec{w}\rangle - b) \ge 1}$ Finde Maximum von $\mathrm{d}_H(T) = \frac{1}{\lVert\vec{w}\rVert} \Longrightarrow$ Finde Minumum von $\lVert\vec{w}\rVert$ (bzw $\frac{1}{2}\sum_{j=1}^{g} w_j^2$) unter der Nebenbed. $y_i(\langle \vec{x}_i, \vec{w}\rangle - b) \ge 1$
Finde Extremwert mit NB	Sei f die zu maximierende Funktion, $g = c$ die Nebenbedingung und P ds lokale Maximum von f unter der Nebenbedingung $g = c$. Dann sind die Tangenten von f und g an der Stelle P parallel $\Longrightarrow \vec{\nabla} f(P) \parallel \vec{\nabla} g(P) \Longrightarrow \vec{\nabla} f(P) = \lambda \vec{\nabla} g(P)$ Die lokale Extrema unter Nebenbedingung zu finden ist äquivalent dazu, die lokalen Extrema der Lagrange-Funktion $\mathrm{L}(x, y, \lambda) = f(x, y) + \lambda(g(x, y) - c)$ zu finden $\Longrightarrow \vec{\nabla}_{xy\lambda} \mathrm{L}(x, y, \lambda) = 0$ Verallgemeinerung: $\mathrm{L}(\vec{x}, \lambda) = f(\vec{x}) + \sum_{i=1}^{m} \lambda_i\, g_i(\vec{x}) \Longrightarrow \vec{\nabla}_{x_1, \dots, x_n, \lambda_1 \dots \lambda_n} \mathrm{L}(x_1, \dots, x_n, \lambda_1 \dots \lambda_n) = 0$
Geränderte Hesse-Matrix	$\bar{H} = \begin{pmatrix} 0 & \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial^2 L}{\partial x^2} & \frac{\partial^2 L}{\partial x \partial y} \\ \frac{\partial g}{\partial y} & \frac{\partial^2 L}{\partial y \partial x} & \frac{\partial^2 L}{\partial y^2} \end{pmatrix}$ $\det(\bar{H}(x_0, y_0)) > 0 \Longrightarrow$ lokales Maximum (unter NB) $\det(\bar{H}(x_0, y_0)) < 0 \Longrightarrow$ lokales Minimum (unter NB) $\det(\bar{H}(x_0, y_0)) = 0 \Longrightarrow$ nicht entscheidbar
SVM Lagrange	$\mathrm{L}(\vec{w}, b, \vec{\alpha}) = f(\vec{w}) - \sum_{i=1}^{N} \alpha_i\, g_i(\vec{w}, b)$ mit $f(\vec{w}) = \frac{1}{2}\sum_{j=1}^{g} w_j^2$ und NB $g_i(\vec{w}, b) = y_i(\langle \vec{x}_i, \vec{w}\rangle - b) - 1$
Äquivalent	Finde Maximum von $f(\vec{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \langle x_i, x_j\rangle$ unter der Nebenbedingung $\sum_{j=1}^{g} \alpha_i y_i = 0$

Ableitungsfreie Optimierungsmethoden

Values on a Simplex

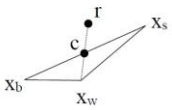
Compute indices $w, b, x \in \{1,\dots,n+1\}$ such that(*):

$x_w := \text{argmin}\{f(x) \mid x \in \Delta\}$ → worst point

$x_s := \text{argmin}\{f(x) \mid x \in \Delta, s \neq w\}$ → second worst point

$x_b := \text{argmax}\{f(x) \mid x \in \Delta, b \neq s, b \neq w\}$ → best point

Compute the *centroid* of the n best points: $c = \frac{1}{n}\sum_{i \neq w} x_i$ (i.e. centroid of the points leaving out worst point x_w)

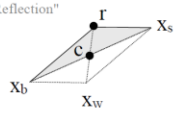


Compute the point r by reflecting x_w at the centroid c (*reflection point*):

$r = c + \alpha(c - x_w)$; α is a hyper-parameter — typically α = 1

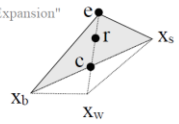
Iteration Steps

"Reflection"



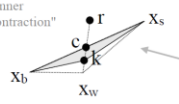
- $f(x_b) \geq f(r) \geq f(x_s)$: Exchange x_w by r in Δ

"Expansion"

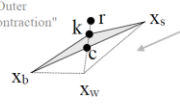


- $f(r) > f(x_b)$: Compute the expanded point $e = c + \beta(c - x_w)$ with β > α (typically, β = 2);
  - if $f(e) > f(r)$, then exchange x_w by e in Δ;
    otherwise: exchange x_w by r in Δ

"Inner contraction"

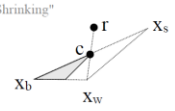


"Outer contraction"



- $f(x_s) > f(r)$: Compute the contracted point k as follows
  - $f(r) > f(x_w)$: $k = c + \gamma(x_w - c)$, $0 < \gamma < \alpha$ (typically γ = 1/2)
  - $f(r) \leq f(x_w)$: $k = c + \gamma(c - x_w)$
- $f(k) > f(x_w)$: Exchange x_w by k in Δ;
  otherwise: see next

"Shrinking"



- $f(x_s) > f(r)$: Compute the contracted point k …
  - $f(k) \leq f(x_w)$: Shrink simplex Δ by 1/2 towards x_b, i.e. exchange x_i in Δ by $x_i \mapsto \frac{1}{2}(x_b + x_i)$ for i ≠ b

Methode

Initialize simplex Δ

Set termination condition ε_f, ε_Δ

repeat

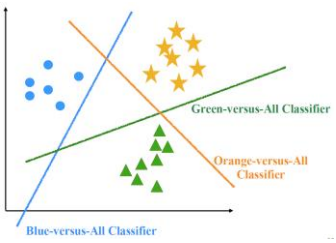
Compute x_b, x_s, x_w, r

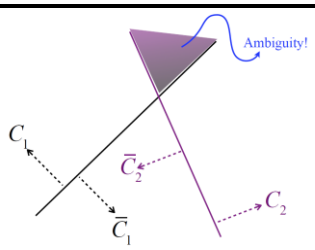
Exchange x_w in Δ (by reflection, expansion, contraction, shrinking

until $f(x_b) - f(x_w) < \varepsilon_f$ or $\max_{x_i \neq x_b} \| x_i - x_b \| < \varepsilon_\Delta$

Multiklassen-Probleme

one-versus-all





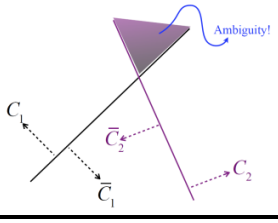
Paarweise

Erzeuge binären Klassifizierer T_{ij} für jedes Klassenpaar C_i, C_j

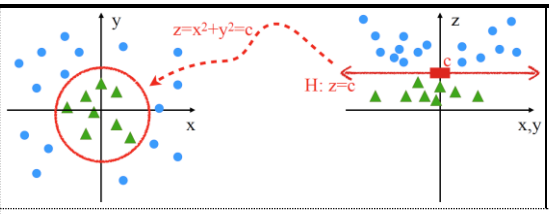
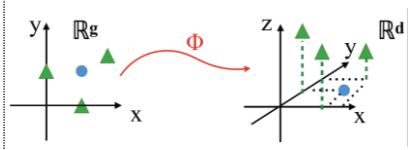
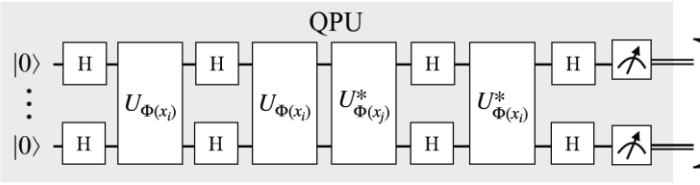
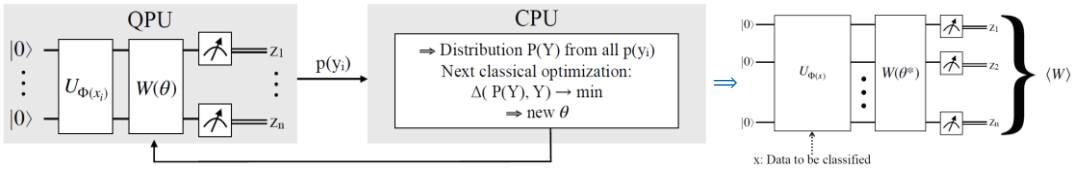
Definiere einen Klassifizierer für C_j als $T_j(\vec{x}) = \sum_{i=1;i\neq j}^{M} T_{ij}(\vec{x})$

Der Gesamt-Klassifizierer ist dann $T(\vec{x}) = \max_{1 \leq j \leq M} T_j(\vec{x})$

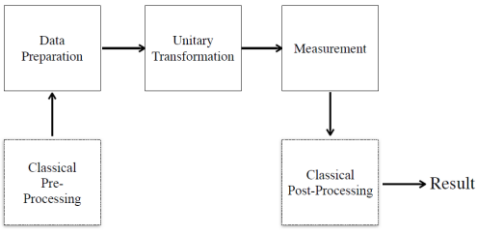
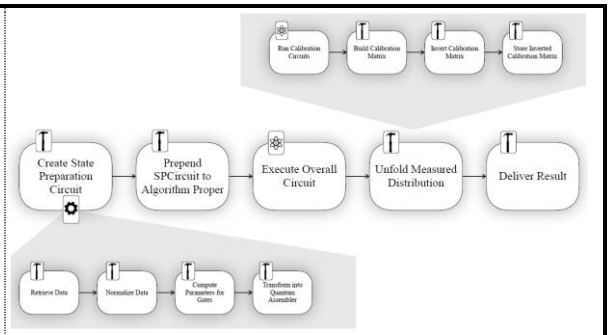
Auch hier git es Ambiguitäten!



Nicht-Lineare Separierung

Übergang in höherdim. Feature-Raum z.B. $z = x^2+y^2$		Allg. Trick: Projektion in höherdim. Featureraum $\Phi: \mathbb{R}^g \rightarrow \mathbb{R}^d$ (Feature-map)	
Trick Kernel-Funktion	Wir müssen daher maximieren: $f(\vec{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \langle \Phi(\vec{x}_i), \Phi(\vec{x}_j) \rangle$ unter Nebenbedingung $\sum_{j=1}^{g} \alpha_i y_i = 0$ Wir brauchen Φ nicht zu kennen, wenn wir eine Kernel-Funktion $K(\vec{x}_i, \vec{x}_j) = \langle \Phi(\vec{x}_i), \Phi(\vec{x}_j) \rangle$ finden! K: $\mathbb{R}^g \times \mathbb{R}^g \rightarrow \mathbb{R}$ Während der Feature-Raum extrem hochdimensional ist. Ist der kernel in einem vergleichsweise niedrigdimensionalen Raum berechnet werden. Insbesondere müssen wir den Feature-Raum und die Feature-Map Φ gar nicht kennen! Das Skalaprodukt wird durch den entsprechenden Krenel ersetzt („Kernelization“)		
Quantum Kernel Estimation		Leite mit diesem Schaltkreis für alle Paare $\{(\vec{x}_i, y_i)\}$ die Koeffizienten $K(\vec{x}_i, y_i)$ der Kernel-Matrix $K = \left(K(\vec{x}_i, y_j)\right)_{ij}$ her. Damit ist das Optimierungsproblem gelöst.	
Alternativ: Klassifizierer auf der QPU berechnen, hybrider Algorithmus			

Hybride Quantum/Klassische Applikationen

Struktur und Workflow Packaging in QAA (Quantum Application Archive)		
Middleware	**Provisioning Engine** mit **Topology Modeler** für „Topology Orchestration“ **Workflow Engine** mit **Workflow Modeler** für “Workflow Orchestration”