Support Vector Machines & K-Nearest Neighbor

Universität Stuttgart

IAAS

Daniel Fink

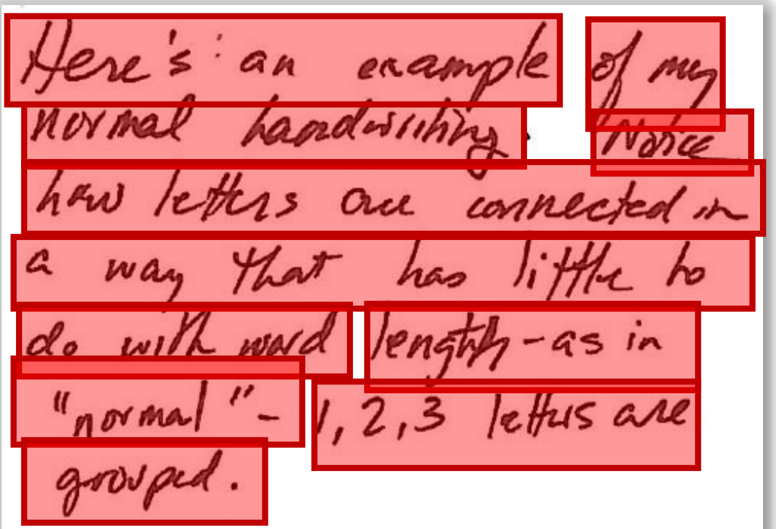
Quantum Machine Learning – Seminar

24. Januar 2020

Motivation

Gesichtserkennung
- → Face Detection
- → Face Recognition

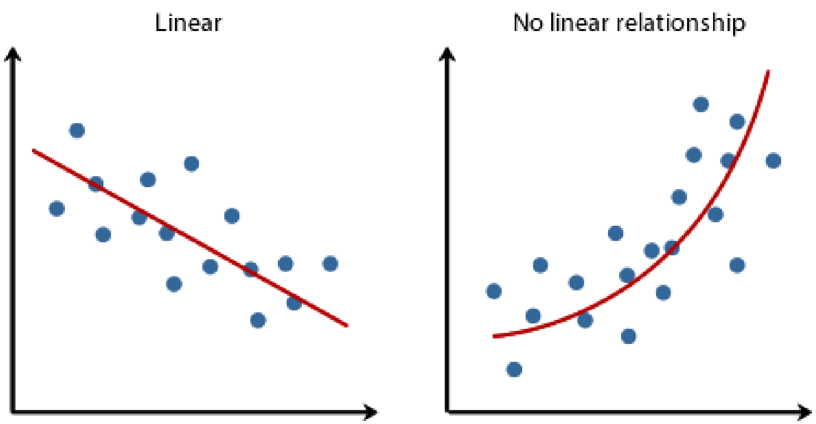
Motivation



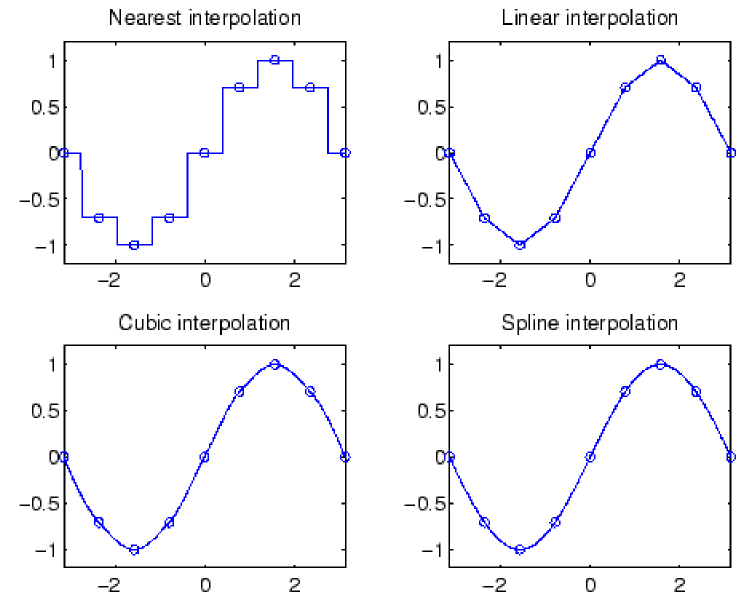
Here‘s an example of my normal handwriting. Notice how letters are connected in a way that has little to do with word length - as in “normal“ – 1, 2, 3 letters are grouped.

Handschrifterkennung

Motivation



Regression

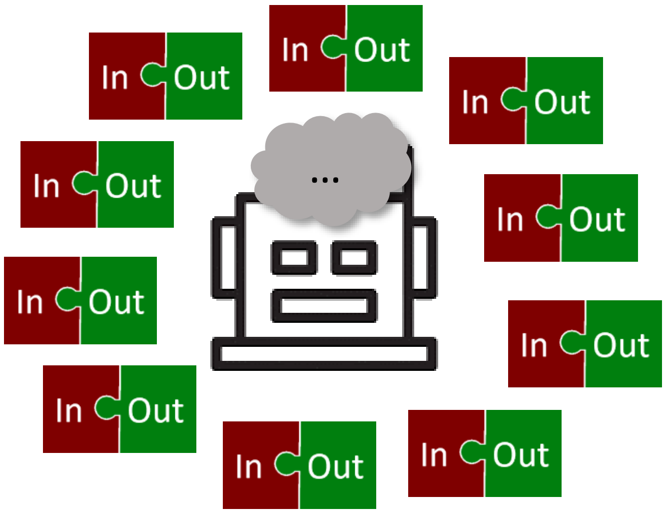


Interpolation

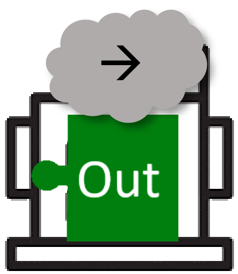
Motivation

Supervised Learning

Motivation


In Out
In Out
In Out
In Out
In Out
In Out
In Out
In Out
In Out
In Out
In Out
...


...


Out


Keine Katze

Übersicht

K-Nearest Neighbor	Theorie
	Auf Quanten-Computer
Support Vector Machines	Theorie
	Im Kontext von NISQ
	Für Big Data

K-Nearest Neighbor
- Theorie

K-Klassifizierungsproblem:

- Gegeben: Trainingsmenge $T = \{(\boldsymbol{v}^p, c^p)\}_{p=1,\dots,N}$

Feature Vector $\rightarrow$ $\boldsymbol{v}^p \in \mathbb{R}^n$ $c^p \in \{1, \dots, k\}$

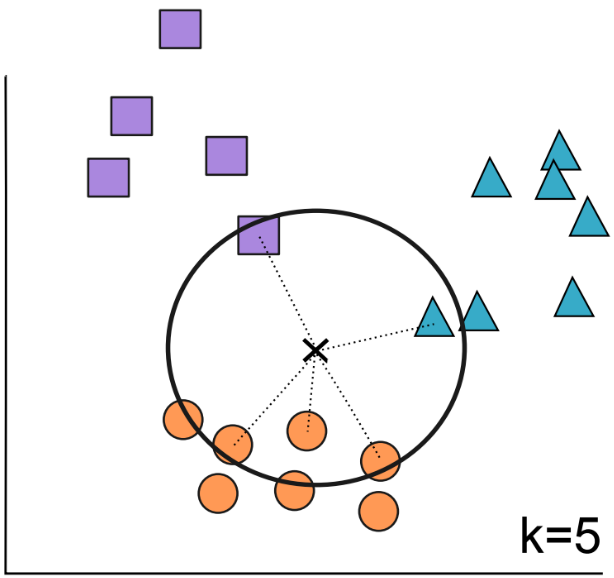
Zu klassifizierender Input-Vektor $\tilde{\boldsymbol{x}} \in \mathbb{R}^n$

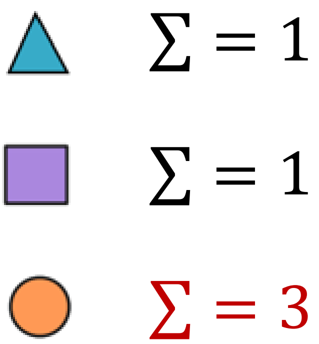
- Gesucht: Klassifizierung $\tilde{c} \in \{1, \dots k\}$

K-Nearest Neighbor

Idee:

- Wähle die k nächsten Nachbarn von $\tilde{\boldsymbol{x}}$
- Weise $\tilde{\boldsymbol{x}}$ die Klasse der Mehrheit der Nachbarn zu





K-Nearest Neighbor

Variationen:

- *K-Nearest Neighbor* mit *Distanz-Gewichtung*
  → Gewichte die Nachbarn mit $1/dist$

- *K-Nearest Centroids*
  → Berechne vorab die jeweiligen Zentren der Klassen
  → Wähle die Klasse mit dem nächsten Zentrum

K-Nearest Neighbor

Frage:

- Was bedeutet „am nächsten"?

→ Geeignetes Ähnlichkeits- bzw. Distanzmaß notwendig

Distanzmaß	$\boldsymbol{d(i,j)}$
Manhattan-Metrik (L_1)	$\sum_k \lvert x_{i,k} - x_{j,k} \rvert$
Euklidische Metrik (L_2)	$\sqrt{\sum_k \lvert x_{i,k} - x_{j,k} \rvert^2}$
Hamming-Metrik	$\lvert\{ k \in \{1, \dots n\} : x_{i,\mathrm{k}} \neq x_{j,\mathrm{k}} \}\rvert$
…	…

K-Nearest Neighbor
– Auf Quanten-Computer

K-Nearest Neighbor

K-Quanten-Klassifizierungsproblem:

k Klassen

- Gegeben: Trainingsmenge

$$\left\{\left|v_1^p \dots v_n^p, c^p\right\rangle\right\}_{p=1,\dots,N} \subset H_2^{\otimes n} \otimes H_k$$

Zu klassifizierender Input-Vektor $|\tilde{x}_1 \dots \tilde{x}_n\rangle \in H_2^{\otimes n}$

- Gesucht: Klassifizierung $|\tilde{c}\rangle \in H_k$

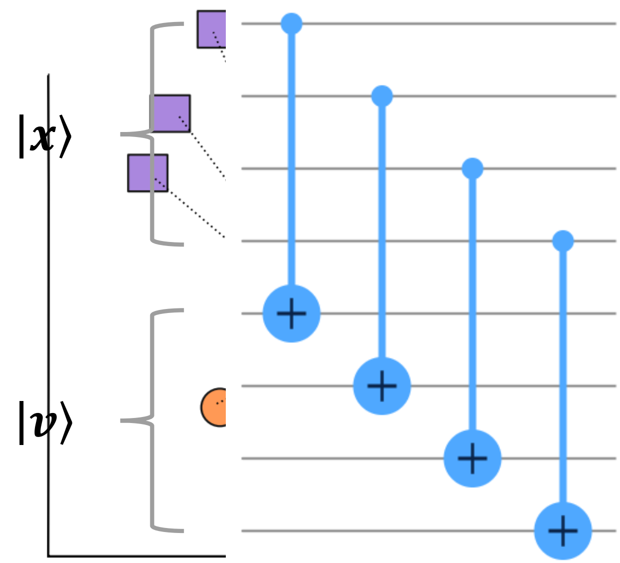
→ Kodierung der Daten als Bitstrings

→ Verwende Hamming-Distanz

K-Nearest Neighbor

Idee:

- K-Nearest Neighbor mit Distanzgewichtung
- Betrachte alle Trainingsdaten
- Berechne Hamming-Distanz mit CNOT



$$\boldsymbol{x} = 01101101$$
$$\boldsymbol{v} = 00101011$$

$$\Rightarrow \quad d(\boldsymbol{x}, \boldsymbol{v}) = 3$$

$$|\boldsymbol{x}\rangle = |01101101\rangle$$
$$|\boldsymbol{v}\rangle = |00101011\rangle$$

$$\xRightarrow{\text{CNOT}} \quad |\boldsymbol{x}'\rangle = |01101101\rangle$$
$$|\boldsymbol{v}'\rangle = |01000110\rangle \longrightarrow \sum = 3$$

K-Nearest Neighbor

Ablauf:

1) Erzeuge Superposition der Trainingsdaten

$$|T\rangle = \sum_p \left| \boldsymbol{v}_1^p \ldots \boldsymbol{v}_n^p, c^p \right\rangle$$

2) Superpositioniere mit Input-Zustand

$$|\psi_o\rangle = \sum_p \left| \tilde{\boldsymbol{x}}_1 \ldots \tilde{\boldsymbol{x}}_n; \boldsymbol{v}_1^p \ldots \boldsymbol{v}_n^p, c^p; 0 \right\rangle$$

Input

Training

Anchilla

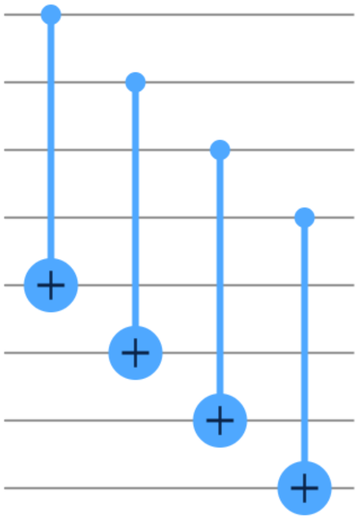
3) Führe Hadamard auf Anchilla aus

$$|\psi_1\rangle = \sum_p \left| \tilde{\boldsymbol{x}}_1 \ldots \tilde{\boldsymbol{x}}_n; \boldsymbol{v}_1^p \ldots \boldsymbol{v}_n^p, c^p \right\rangle \otimes (|0\rangle + |1\rangle)$$

K-Nearest Neighbor

Ablauf:

4) Berechne Hamming-Distanz mit CNOT



$$|\psi_2\rangle = \prod_k CNOT(\tilde{\boldsymbol{x}}_k, \boldsymbol{v}_k^p)\,|\psi_1\rangle$$

- Derzeit:

  Große Ähnlichkeit = Kleine Hamming-Distanz

- Für später sinnvoller:

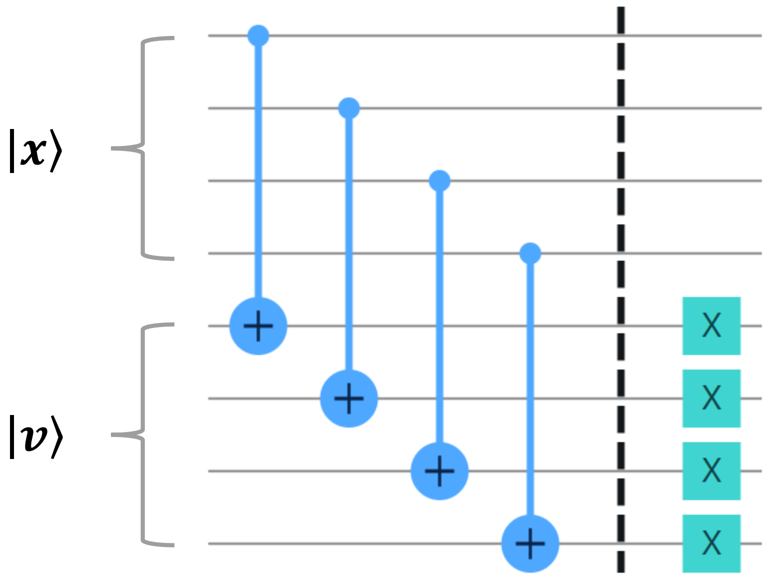
  Große Ähnlichkeit = Große inverse Hamming-Distanz

K-Nearest Neighbor

Ablauf:

4) Berechne inverse Hamming-Distanz mit CNOT

$$|\psi_2\rangle = \prod_k \, X(\tilde{\boldsymbol{x}}_k)\, CNOT(\tilde{\boldsymbol{x}}_k, \boldsymbol{v}_k^p)\, |\psi_1\rangle$$



K-Nearest Neighbor

Ablauf:

4) Berechne inverse Hamming-Distanz mit CNOT

$$|\psi_2\rangle = \prod_k X(\tilde{\boldsymbol{x}}_k)\, CNOT(\tilde{\boldsymbol{x}}_k, \boldsymbol{v}_k^p)\, |\psi_1\rangle$$
$$= \sum_p \left|\tilde{\boldsymbol{x}}_1 \dots \tilde{\boldsymbol{x}}_n; d_1^p \dots d_n^p, c^p\right\rangle \otimes (|0\rangle + |1\rangle)$$

5) Summiere inverse Hamming-Distanzen

$$U = e^{i\frac{\pi}{2n}H} \text{ mit } H = 1 \otimes \underbrace{\sum_k \left(\frac{\sigma_z+1}{2}\right)_k}_{\text{Summation}} \otimes 1 \otimes \sigma_z$$

Negativ, wenn Ancilla = 1

$$\left(\frac{\sigma_z+1}{2}\right)|0\rangle = \frac{1\cdot|0\rangle+|0\rangle}{2} = |0\rangle \qquad \left(\frac{\sigma_z+1}{2}\right)|1\rangle = \frac{-1\cdot|1\rangle+|1\rangle}{2} = 0$$

K-Nearest Neighbor

Ablauf:

5) Summiere inverse Hamming-Distanzen

$$|\psi_3\rangle = U\,|\psi_2\rangle$$

$$= \sum_p e^{i\frac{\pi}{2n}d_H(\tilde{x},v^p)}\left|\tilde{\boldsymbol{x}}_1 \dots \tilde{\boldsymbol{x}}_n; d_1^p \dots d_n^p, c^p; 0\right\rangle + e^{-i\frac{\pi}{2n}d_H(\tilde{x},v^p)}\left|\tilde{\boldsymbol{x}}_1 \dots \tilde{\boldsymbol{x}}_n; d_1^p \dots d_n^p, c^p; 1\right\rangle$$

6) Wende erneut Hadamard auf Anchilla an

$$|\psi_4\rangle = (1 \otimes 1 \otimes 1 \otimes H)\ |\psi_3\rangle$$

$$= \sum_p \cos\left[\frac{\pi}{2n} d_h(\tilde{\boldsymbol{x}}, \boldsymbol{v}^p)\right] |\tilde{\boldsymbol{x}}; \boldsymbol{d}^p, c^p; 0\rangle + i\ sin\left[\frac{\pi}{2n} d_h(\tilde{\boldsymbol{x}}, \boldsymbol{v}^p)\right] |\tilde{\boldsymbol{x}}; \boldsymbol{d}^p, c^p; 1\rangle$$

Mit Eulerscher Formel: $e^{ix} = \cos(x) + i\sin(x)$

K-Nearest Neighbor

Ablauf:

$\cos(0) = 1 \qquad \sin(0) = 0$

7) Messe Anchilla

$P(|a\rangle = |0\rangle) \propto \sum_p \cos^2\left[\frac{\pi}{2n} d_h(\tilde{\boldsymbol{x}}, \boldsymbol{v}^p)\right]$ → Input weit weg von Trainingsdaten

$P(|a\rangle = |1\rangle) \propto \sum_p \sin^2\left[\frac{\pi}{2n} d_h(\tilde{\boldsymbol{x}}, \boldsymbol{v}^p)\right]$ → Input nah dran an Trainingsdaten

$d_h(\tilde{\boldsymbol{x}}, \boldsymbol{v}^p)$
- Klein, wenn $\tilde{\boldsymbol{x}}$ und $\boldsymbol{v}^p$ unähnlich sind
- Groß, wenn $\tilde{\boldsymbol{x}}$ und $\boldsymbol{v}^p$ ähnlich sind

8) Verwende Threshold bis Anchilla = 1 gemessen wird

→ Messe $m \in \mathbb{N}$ mal
- $\forall m\colon\ Anchilla = 0$ → Abbruch!
- $\exists m\colon\ Anchilla = 1$ → Bestimme Klasse $\tilde{c}$

Ablauf:

8) Verwende Threshold bis Anchilla = 1 gemessen wird

$$|\psi_5\rangle = \sum_p i\ sin\left[\frac{\pi}{2n} d_h(\tilde{\boldsymbol{x}}, \boldsymbol{v}^p)\right] |\tilde{\boldsymbol{x}}; \boldsymbol{d}^p, c^p; 1\rangle$$

$$= \sum_c |c\rangle \otimes \sum_{l \in c} i\ sin\left[\frac{\pi}{2n} d_h(\tilde{\boldsymbol{x}}, \boldsymbol{v}^l)\right] |\tilde{\boldsymbol{x}}; \boldsymbol{d}^l; 1\rangle$$

9) Messe Klassen-Qbit

$$P(c) \propto \sum_{l \in c} sin^2\left[\frac{\pi}{2n} d_h(\tilde{\boldsymbol{x}}, \boldsymbol{v}^l)\right]$$

Summe läuft über die Trainingsdaten der Klasse c

Proportional zur inversen Hamming-Distanz

K-Nearest Neighbor

Fazit:

- Konstruktion einer beliebigen Superposition $\sum_p |\tilde{\boldsymbol{x}}_1 \dots \tilde{\boldsymbol{x}}_n; \boldsymbol{v}_1^p \dots \boldsymbol{v}_n^p, c^p; 0\rangle$ ist schwierig → QRAM
- Laufzeit $O(TNn)^4$
- Qbits $O(n)$
- Liefert mehr Informationen → WS für die Klassen

Support Vector Machine
- Theorie

Support Vector Machine

Binäres Klassifizierungsproblem:

- Gegeben: Trainingsmenge $T = \{(\boldsymbol{v}^p, c^p)\}_{p=1,\ldots,N}$

  $\boldsymbol{v}^p \in \mathbb{R}^n$ $\quad$ $c^p \in \{-1, +1\}$

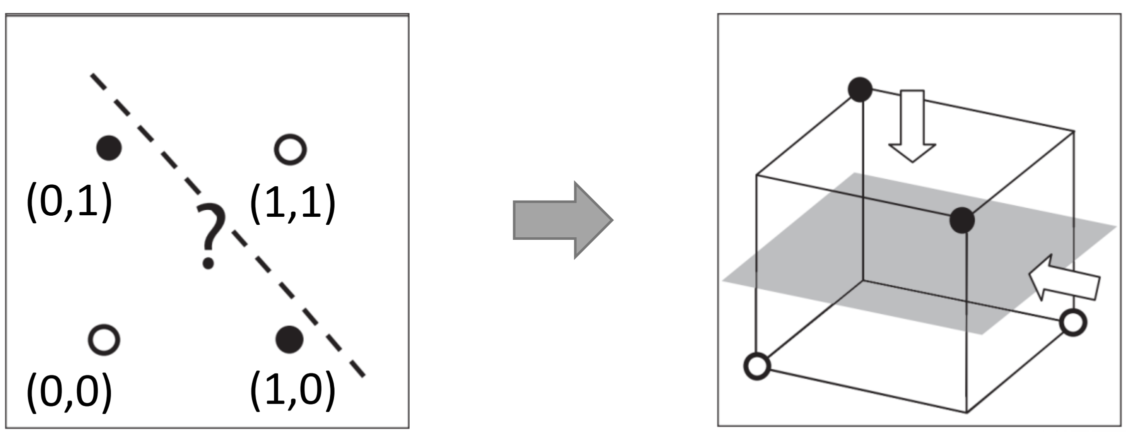
  Zu klassifizierender Input-Vektor $\tilde{\boldsymbol{x}} \in \mathbb{R}^n$

- Gesucht: Klassifizierung $\tilde{c} \in \{-1, +1\}$

Idee:

- Transformiere Daten in einen höherdimensionalen Raum
- Finde Hyperebene, welche die Datenpunkte trennt
- Prüfe auf welcher Seite $\tilde{\boldsymbol{x}}$ liegt und weise Klasse zu

- Beispiel XOR-Problem:

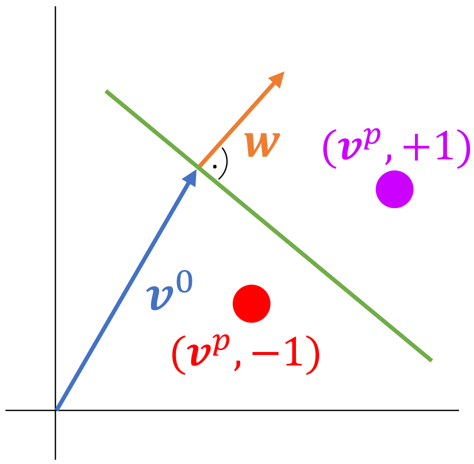


Geometrische Herleitung:

- Hyperebene als Punkt-Normalenform

$$\boldsymbol{w}^T(\boldsymbol{v} - \boldsymbol{v}^0) = 0$$

$$\boldsymbol{w}^T\boldsymbol{v} + b = 0 \text{ mit } b = -\boldsymbol{w}^T\boldsymbol{v}^0$$



- Trennung der Trainingsdaten durch

$$\boldsymbol{w}^T\boldsymbol{v}^p + \boldsymbol{b} \geq 0 \quad \text{für alle} \quad c^p = +1$$

$$\boldsymbol{w}^T\boldsymbol{v}^p + \boldsymbol{b} < 0 \quad \text{für alle} \quad c^p = -1$$

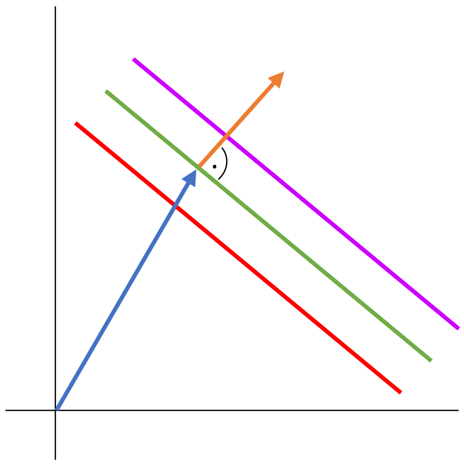
Support Vector Machine

Geometrische Herleitung:

- Führe Trennbereich ein

$$\boldsymbol{w}^T \boldsymbol{v}^p + \boldsymbol{b} \geq +1 \quad \text{für alle} \quad c^p = +1$$

$$\boldsymbol{w}^T \boldsymbol{v}^p + \boldsymbol{b} < -1 \quad \text{für alle} \quad c^p = -1$$



- Damit gilt

$$c^p(\boldsymbol{w}^T \boldsymbol{v}^p + \boldsymbol{b}) \geq +1$$

variabel

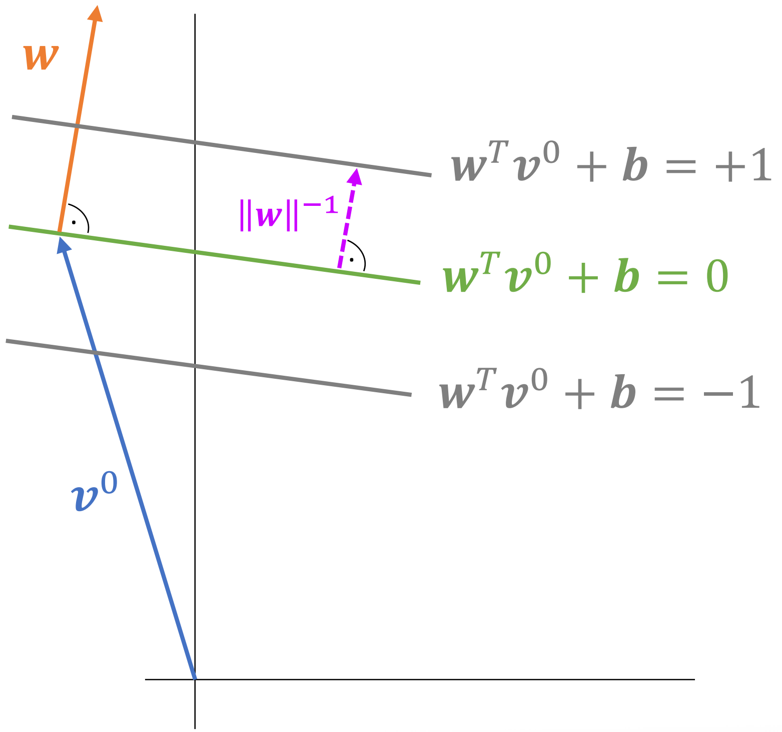
b hat keinen Einfluss auf die Breite

Bestimme $\boldsymbol{w}$ so, dass diese Gleichung erfüllt wird

Support Vector Machine

Geometrische Herleitung:

$$\boxed{c^p(\boldsymbol{w}^T\boldsymbol{v}^p - b) \geq +1}$$



Trennbereich soll möglichst groß werden

→ Minimiere Kostenfunktion

$$\Phi(w) = \frac{1}{2}\|w\|^2$$

Support Vector Machine

Bisher:

- Minimiere $\Phi(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|^2$ → Maximaler Trennbereich
- Erfülle $c^p(\boldsymbol{w}^T\boldsymbol{v}^p - b) \geq +1$ → Trennbarkeit

→ Quadratisches Optimierungsproblem

→ Verwende Lagrange-Multiplikatoren

Minimiere

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|w\|^2 - \sum_p \alpha_p \left(c^p(\boldsymbol{w}^T\boldsymbol{v}^p - b) - 1\right)$$

hinsichtlich $\boldsymbol{w}$ und b

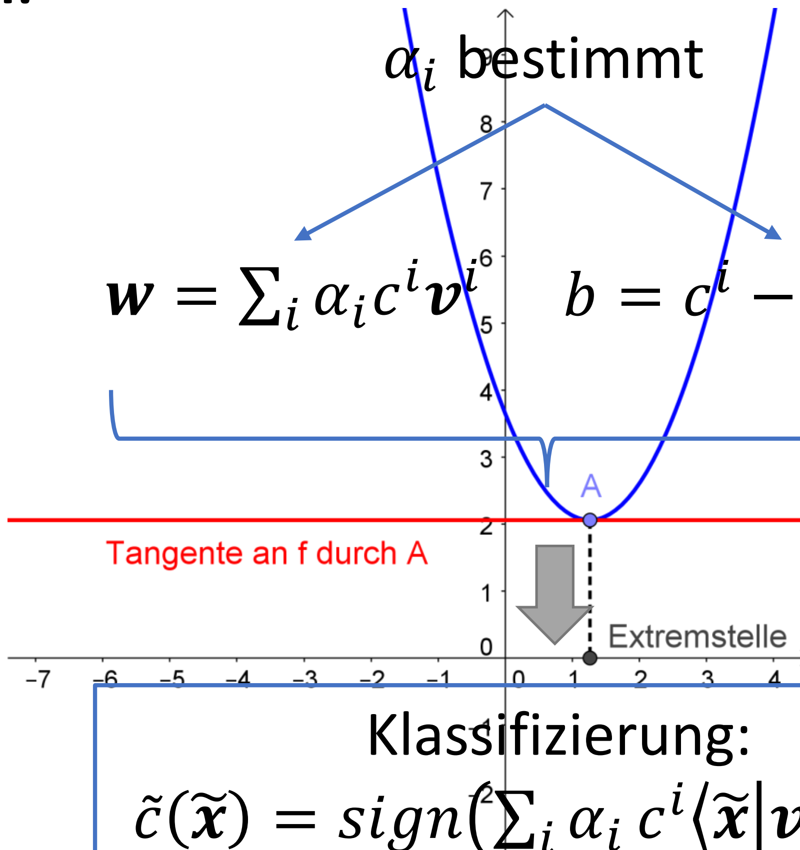
Support Vector Machine

Optimierungsproblem:

$$\left.\begin{aligned}\frac{\partial \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha})}{\partial \boldsymbol{w}} &= 0 \\[2em] \frac{\partial \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha})}{\partial b} &= 0\end{aligned}\right\}$$

α_i bestimmt

$$\underbrace{\boldsymbol{w} = \sum_i \alpha_i c^i \boldsymbol{v}^i \qquad b = c^i - \langle \boldsymbol{w} | \boldsymbol{v}^i \rangle}$$



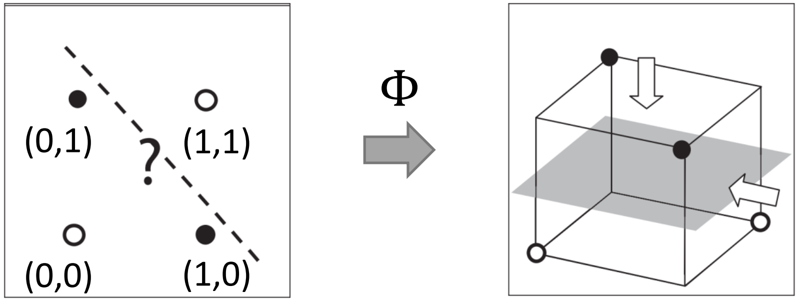
Klassifizierung:

$$\tilde{c}(\tilde{\boldsymbol{x}}) = sign\left(\sum_i \alpha_i\, c^i \langle \tilde{\boldsymbol{x}} | \boldsymbol{v}^j \rangle + b\right)$$

Support Vector Machine

Bisher:

- Daten wurden noch nicht in höherdimensionalen Raum transformiert



$$\Phi: \mathbb{R}^n \rightarrow H$$

nicht gut trennbar

mehr Freiheiten für Trennung

- Verwende Kernel-Trick

$$\langle \boldsymbol{x} | \boldsymbol{y} \rangle \quad \longrightarrow \quad \underbrace{\langle \Phi(\boldsymbol{x}) | \Phi(\boldsymbol{y}) \rangle}_{\text{schwer zu berechnen}} \quad = \quad \underbrace{K(\boldsymbol{x}, \boldsymbol{y})}_{\text{einfach zu berechnen}}$$

Support Vector Machine
- NISQ

Support Vector Machine

Vorher:

- Klassischer Algorithmus → Quanten Algorithmus

Jetzt:

- Quanten-Rechner → Quanten Klassifikator

Konkret:

- IBMQ mit 5 Qbits
- Keine Fehlerkorrektur
- Eingeschränkte Konnektivität
- Eingeschränkte Breite
- Eingeschränkte # Operatoren

Was kann man damit anstellen?

Support Vector Machine

Kodierung der Daten:

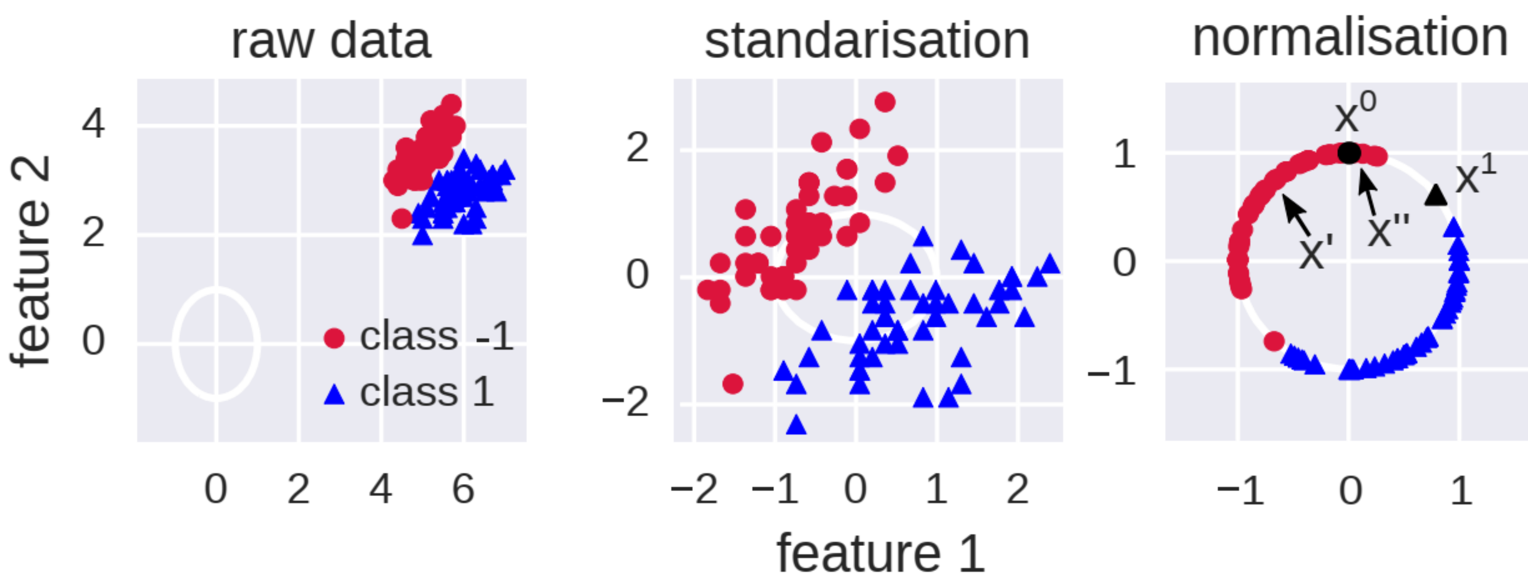
- $\boldsymbol{v} = \underbrace{(v_1, v_2, \ldots, v_n)}_{O(n)\ \text{Bits}} \in \mathbb{R}^n \rightarrow |\Psi_v\rangle = \underbrace{\sum_{i=0}^{n-1} v_i\, |i\rangle}_{n\ \text{Terme}} \in H_2^{\otimes d}$

→ Superposition: $n = 2^d \rightarrow \mathrm{d} = \log_2(\mathrm{n})$

→ Exponentielle „Komprimierung“ der Daten

Support Vector Machine

(Klassische) Aufbereitung der Daten:



Support Vector Machine

Ablauf:

1) Superpositioniere und verschränke Input und Trainingsdaten

$$|\Psi_o\rangle = \sum_p |p\rangle \left(|0\rangle|\Psi_{\tilde{x}}\rangle + |1\rangle|\Psi_{v^p}\rangle\right)|c^p\rangle$$

Kennzeichnung Anchilla Input/Training Klasse

2) Führe Hadamard auf Anchilla aus

$$|\Psi_1\rangle = \sum_p |p\rangle \left(|0\rangle|\Psi_{\tilde{x}+v^p}\rangle + |1\rangle|\Psi_{\tilde{x}-v^p}\rangle\right)|c^p\rangle$$

$$\left|\Psi_{\tilde{x}\pm v^p}\right\rangle = |\Psi_{\tilde{x}}\rangle \pm |\Psi_{v^p}\rangle$$

Support Vector Machine

Ablauf:

$$|\Psi_1\rangle = \sum_p |p\rangle \left(|0\rangle|\Psi_{\tilde{\boldsymbol{x}}+\boldsymbol{v}^p}\rangle + |1\rangle|\Psi_{\tilde{\boldsymbol{x}}-\boldsymbol{v}^p}\rangle\right)|c^p\rangle$$

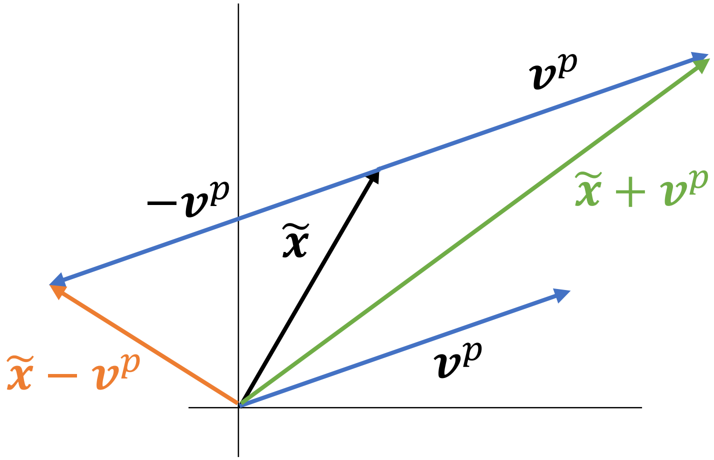
3) Messe Anchilla

$$|\Psi_{\tilde{\boldsymbol{x}}}\rangle + |\Psi_{\boldsymbol{v}^p}\rangle = \sum_{i=0}^{n-1} \tilde{\boldsymbol{x}}_i\,|i\rangle + \sum_{i=0}^{n-1} \boldsymbol{v}_i^p\,|i\rangle$$

$$= \sum_{i=0}^{n-1} (\tilde{\boldsymbol{x}}_i + \boldsymbol{v}_i^p)\,|i\rangle$$

→ $P(|a\rangle = |0\rangle) \propto \sum_p |\tilde{\boldsymbol{x}} + \boldsymbol{v}^p|^2$

→ Hohe WS wenn Input nah an Trainingsdaten



Support Vector Machine

Ablauf:

3) Resultierender Zustand

$$|\Psi_2\rangle = \sum_p \sum_{i=0}^{n-1} |p\rangle \underbrace{(\tilde{\boldsymbol{x}}_i + \boldsymbol{v}_i^p)}_{\text{Gewichtung von Klasse } c^p} |i\rangle\, |c^p\rangle$$

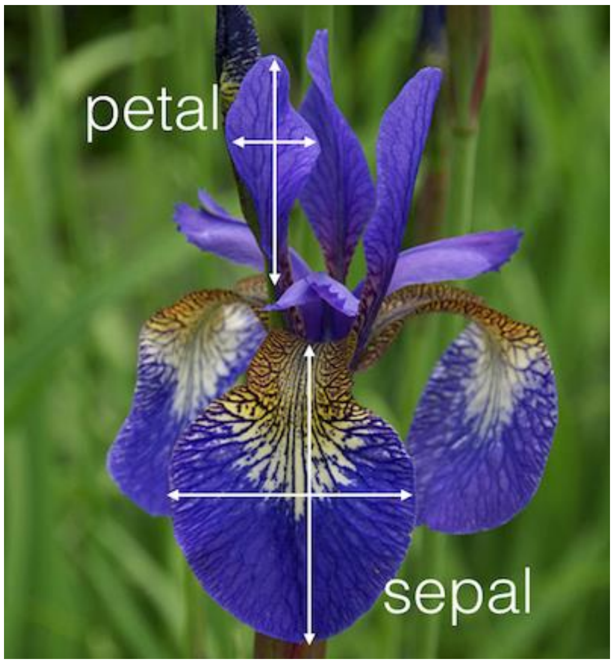
4) Messe Klasse

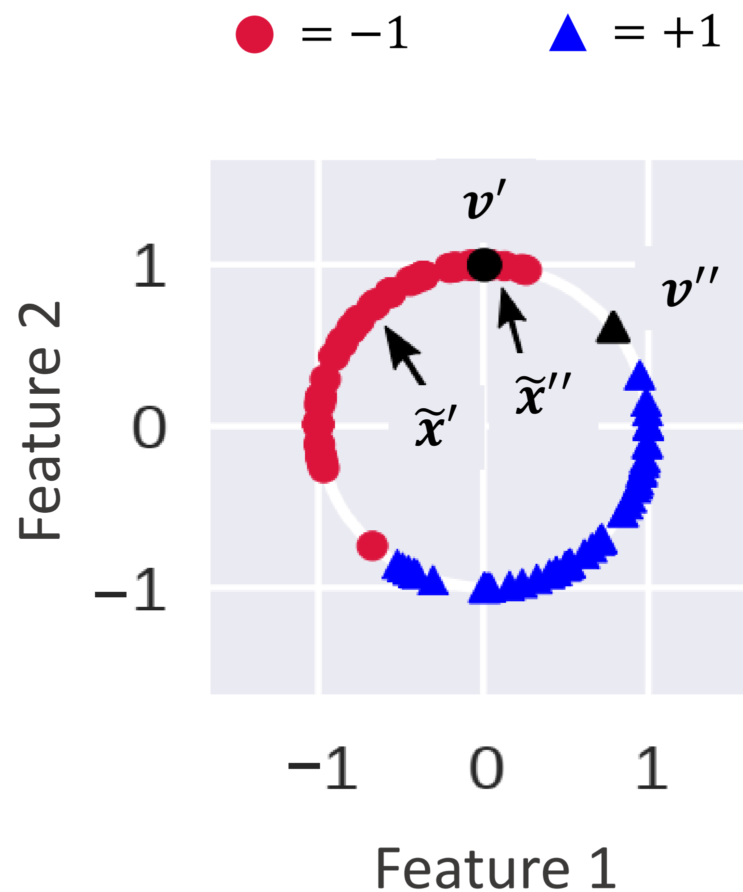
$$P(|c\rangle = |0\rangle) \propto \sum_{p:c^p=0} |\tilde{\boldsymbol{x}} + \boldsymbol{v}^p|^2 \Rightarrow \tilde{c} = -1$$

$$\boxed{\text{Klassifizierung:} \quad \tilde{c}(\tilde{\boldsymbol{x}}) = sign\Big(\sum_p c^p \underbrace{\Big[1 - \tfrac{1}{4N} |\tilde{\boldsymbol{x}} - \boldsymbol{v}^p|^2\Big]}_{\text{Kernel}}\Big)}$$

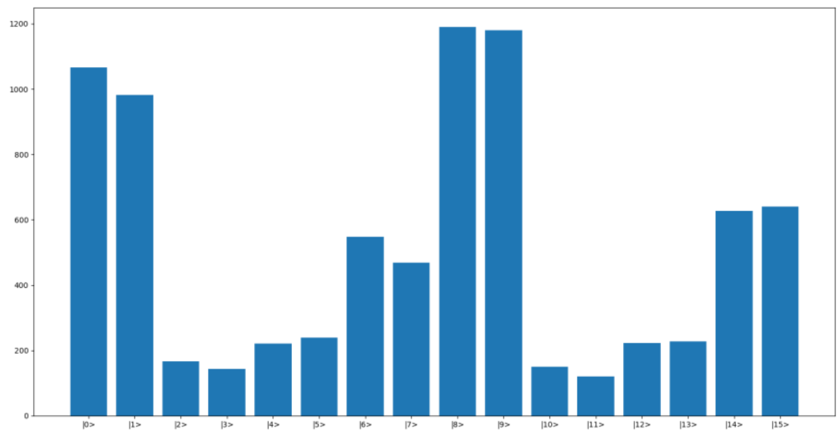
Support Vector Machine

Live Demo:



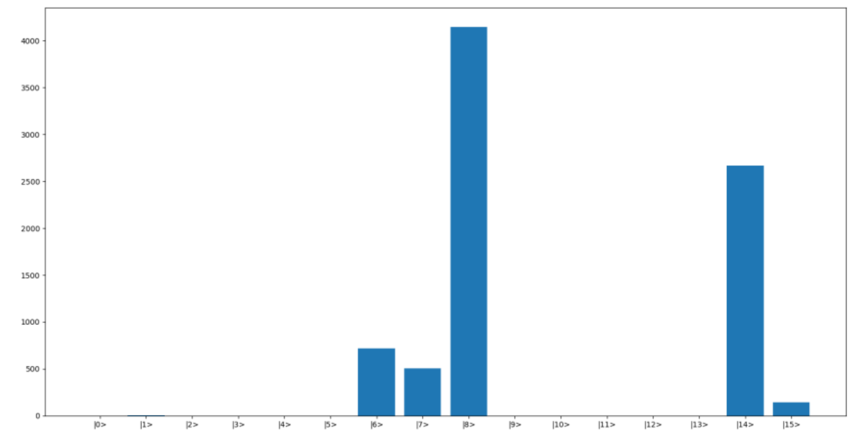


Live Demo:



IBMQ

$$P(\tilde{c} = -1) \approx 60\%$$

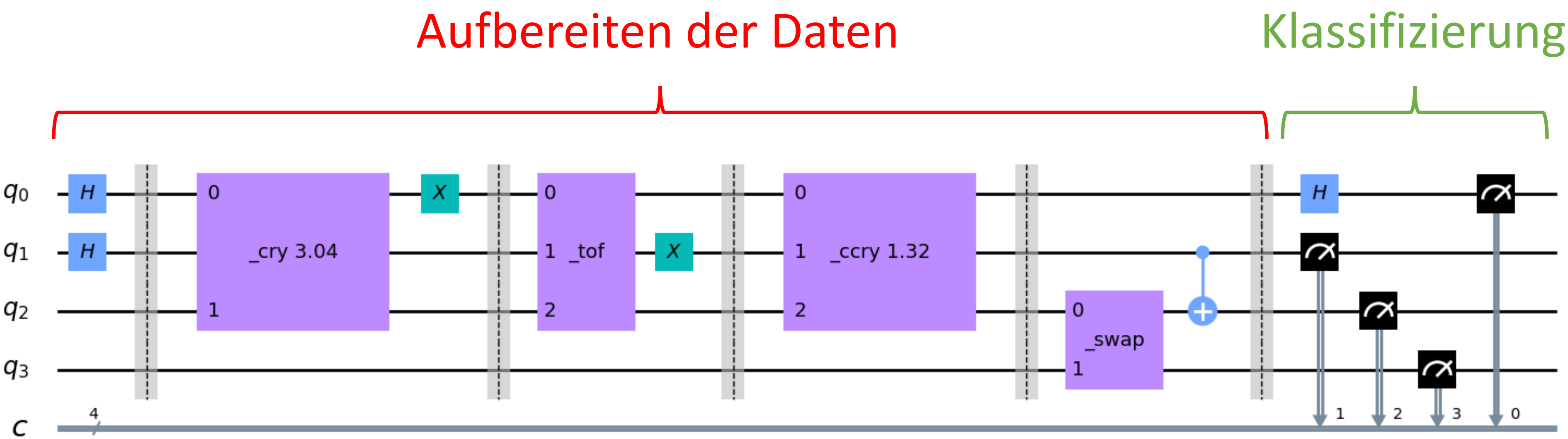


Simulator

$$P(\tilde{c} = -1) = 55\%$$

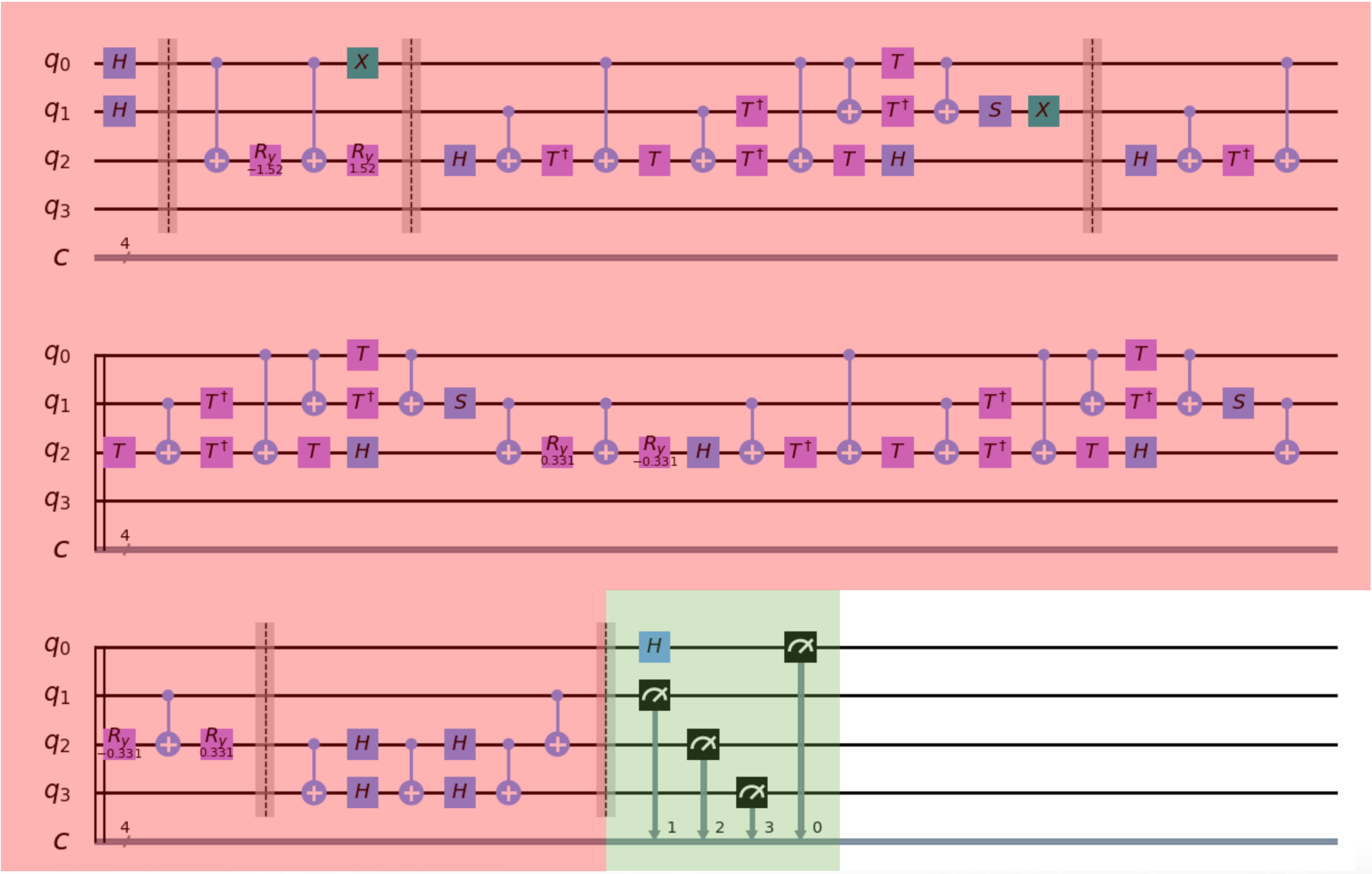
Support Vector Machine

Live Demo:



Kompakter Schaltkreis

Live Demo:



Support Vector Machine
- Big Data (Ausblick)

Quantum Random Access Machine:

- Zugriff auf Superpositionen in $O(\log\ Nn)$
- Benötigt $O(nN)$ Hardware-Resourcen

→ Q-SVM durchführbar in $O(\log\ Nn)$

SVM als „least squares" Problem:

- Führe andere Lagrange-Multiplikatoren ein

$$c^p(\boldsymbol{w}^T\boldsymbol{v}^p + b) \geq 1 \quad \rightarrow \quad (w^T v^p + b) = c^p - c^p\beta_p$$

$$\Rightarrow \quad \begin{bmatrix} 0 & \mathbf{1}^T \\ \mathbf{1} & \boldsymbol{K} + \gamma^{-1}\boldsymbol{I} \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ \boldsymbol{c} \end{bmatrix}$$

- Löse Gleichungssystem mit HHL
- In $O(\text{polylog } Nn)$ durchführbar

Support Vector Machine

Fazit:

- Größtes Bottleneck ist das Laden der Daten
- Durch Kernel-Trick lässt sich klassisch bereits in hohen Dimensionen arbeiten
- Datenkomprimierung ermöglicht aber Vorteil
- Bereits gute Ergebnisse auf NISQ-Rechnern

Danke für die Aufmerksamkeit

→ Fragen und Diskussion