technische universität
dortmund

Fakultät Informatik

Sommersemester 2015

Seminararbeit

Support Vector Machine
und
Quantum Computing

Bearbeiter:	Yayla, Mikail
Matrikelnummer:	148962
Telefonnummer:	015779675480
Anschrift:	Bergheimer Steig 3, 45357 Essen
Email-Adresse:	mikail.yayla@tu-dortmund.de

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

SVM	Support Vector Machine
ML	Machine Learning
QC	Quantum Computing

1 Einleitung

Der Begriff „Quantum“ scheint bei vielen Menschen unangenehme Gefühle hervorzurufen. Es ist eine „geheimnisvolle“ Wissenschaft, die auch von Physikern wie z.B Seth Lloyd, der am MIT an Quantum Machine Learning forscht als „weird“ beschrieben wird. Sie scheint unserer Logik zu widersprechen. Dies macht es jedoch interessant. Es ist ein völlig neues Gebiet für Informatiker, die strenge Logik gewöhnt sind. Die grundlegende Idee des Quantum Computing und die Möglichkeiten, die es für die Machine Learning Algorithmen und insbesondere für die Support Vector Machine bieten könnte wird in dieser Arbeit diskutiert.

2 Einführung in Quantum Computing

2.1 Welle Teilchen Dualismus

Ein wichtiges Prinzip der Quantenphysik ist der Welle-Teilchen-Dualismus. Nach diesem Prinzip hat ein Objekt gleichzeitig Eigenschaften einer klassischen Welle (z.B wie im Wasser) und die von einem klassischen Teilchen (z.B eine Erbse). Ein Elektron ist ein kleines Teilchen, das sich um das Atom bewegt. Wenn man nicht „misst“, dann besitzt dieses Elektron die Eigenschaften einer Welle. Wenn man das Elektron durch einen Schlitz führen würde, dann würde es sich so ausbreiten wie eine Welle im Wasser. Das ist der erste Teil des Prinzips und es gilt jedoch nur, wenn keine Messungen durchgeführt werden. Bei einer Messung ist das Ziel, die Position des Elektrons zu bestimmen. Eine Messung kann folgendermaßen durchgeführt werden: Es wird eine „Elektronenkanone“ vorbereitet, die Elektronen auf eine Wand schießt. Zwischen Wand und Kanone wird ein Messgerät gestellt, das ein Signal gibt, wenn das Elektron durchfliegt. Durch diese Messung kann man den Ort des Elektrons bestimmen. Jetzt kommt der zweite Teil des Prinzips ins Spiel. Wenn man die Position des Elektrons durch eine solche Messung bestimmt, dann verliert das Elektron seine Welleneigenschaft und weist nur noch Teilcheneigenschaften auf. Es verhält sich danach wie ein festes Objekt und nicht mehr wie eine Welle. Ohne Messung würde auf der Wand ein Wellenmuster erkennbar sein. Das kann man sich wie Licht vorstellen. In der Mitte der Wand, da wo der vorderste Teil der Welle auf die Wand stößt, würde es heller sein als links und rechts von diesem Punkt. Wenn jedoch gemessen werden würde, dann würden Punkte auf der Wand erkennbar sein, denn das Elektron würde sich in diesem Fall wie ein festes Objekt verhalten. Dieses Zusammenspiel von Zuständen, sowie der Kollaps der Wellenfunktionen bei Messungen sind wichtige Konzepte im Quantum Computing.

2.2 Superposition

Superposition ist die Überlagerung von gleichartigen physikalischen Größen. Wenn man z.B. eine Kraft mit einer anderen Kraft addiert, dann erhält man als Ergebnis auch eine Kraft. Einige physikale Größen können als Vektoren modelliert werden. Wenn zwei Vektoren gleicher Dimension addiert werden, dann erhält man einen Vektor, der die gleiche Dimension hat, wie diese zwei Vektoren. Die Wellenfunktion eines Elektrons, die durch Vektoren modelliert wird befindet sich ständig in einer solchen Superposition

2.3 Verschränkung

Dies ist ebenfalls ein wichtiges Prinzip. In einfachen Worten kann man es so erklären: Der Zustand eines Systems kann nicht durch die Betrachtung seiner Einzelteile bestimmt werden. Also liegt Verschränkung vor, wenn der Zustand eines Systems von mehreren Teilchen sich nicht durch die Betrachtung unabhängiger Ein-Teilchen-Zustände bestimmen lässt, sondern nur durch einen gemeinsamen Zustand. Dieses Prinzip kann man besser verstehen, wenn man sich Qubits anschaut.

3 Das Quantum Bit - Qubit

Es gibt in unserem Universum Teilchen, die quantenmechanische Eigenschaften besitzen, wie z.B. das Elektron. Ein Elektron hat viele Eigenschaften, wie z.B. den Spin. Der Spin ist der Eigendrehimpuls von Teilchen. Wie auch den Drehimpuls in der klassischen Mechanik kann man den Spin als Vektor im zweidimensionalen Raum beschreiben. Dieser Vektor kann entweder nach „oben“ oder nach „unten“ zeigen. „Spin-Oben“ wird als Eins und „Spin-Unten“ wird als Null modelliert. Eins repräsentiert den Einheitsvektor $\begin{pmatrix}0 & 1\end{pmatrix}^T$. Dieser Vektor ist auch ein Basisvektor des zweidimensionalen Raumes. Null wird als $\begin{pmatrix}1 & 0\end{pmatrix}^T$ dargestellt, dieser ist orthogonal zu dem ersten Vektor und zusammen spannen sie den zweidimensionalen Raum auf. Durch die Linearkombination (oder Superposition) dieser beiden Vektoren können beliebige Vektoren im zweidimensionalen Raum erstellt werden. Die Schreibweise |x> heißt Dirac-Notation. Einfach erklärt sagt sie aus, dass x ein quantenmechanisches Objekt beschreibt.

$$0 \rightarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix} \equiv |0> \ , \ 1 \rightarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix} \equiv |1>$$

Der Vektor $(\alpha \quad \beta)^T \in \mathbb{C}^2$ ist bereits eine vollständige Modellierung eines Qubits. Es ist zu erkennen, dass mit Qubits im zweidimensionalen komplexen Raum gerechnet werden muss. $\mathbb{C}$ bietet die mathematischen Grundlagen für die Beschreibung von quantenmechanischen Objekten.

Schreibt man den Vektor $(\alpha \quad \beta)^T$ mithilfe der Basisvektoren, so erhält man folgende Notation:

$$\alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \alpha |0> + \beta |1>$$

Noch immer ist nicht geklärt, wie man mithilfe dieses Vektors rechnen oder Null bzw. Eins repräsentieren kann, die Grundlage von jedem Computer.

Nehmen wir ein quantenmechanisches Objekt, z.B ein Elektron, das zwei mögliche Zustände hat, |1> als „Spin-Oben“ und |0> als „Spin-Unten“. Es befindet sich in einem unbestimmten quantenmechanischen Zustand, das heißt, dass es unbestimmt ist, in welche Richtung der Spin zeigt.

Die Unbestimmtheit ist das Schlüsselfeature der Quantenmechanik. Man kann kann hier den Welle-Teilchen-Dualismus als Analogie wählen und zuerst die Eigenschaft „Ort“ betrachten, dies ist intuitiver. Es ist unbestimmt, an welcher Position sich das Elektron befindet. Es besitzt noch seine Wellenfunktion. Wenn nun die Position des Elektrons bestimmt wird, zerfällt die Welleneigenschaft und man bekommt Auskunft über den Ort.

Der Spin besitzt auch diese Unbestimmtheit. Wenn man die Eigenschaft „Spin“ misst, dann erhält man als Antwort „Oben“ oder „Unten“ und die Welleneigenschaft zerfällt. Eins oder Null wird also zurückgegeben. Bis zur Messung befindet sich das Qubit in einer Superposition von |0> und |1>, es ist weder |0> noch |1>. Es befindet sich in beiden Zuständen gleichzeitig und ist gleichzeitig in keinem der beiden Zustände |0> und |1>, denn der Zustand

ist unbestimmt. Es ist so lange in diesem unbestimmten Zustand bis eine Messung vorgenommen wird.

Nun kann man auch die Bedeutung von α und β besser verstehen. α und β sind nicht die Wahrscheinlichkeiten, sie sind die Amplituden der Wellenfunktion von den quantenmechanischen Eigenschaften der Elektronen und sind somit nur proportional zur Wahrscheinlichkeit. α^2 ist die Wahrscheinlichkeit für das Erhalten einer Null, oder „Spin-Unten“ wenn die quantenmechanische Eigenschaft Spin gemessen wird. β repräsentiert dabei die Eins. Dabei muss immer $|\alpha|^2 + |\beta|^2 = 1$ gelten.

4 Manipulation von Qubits

Wenn man in der Frage „Wie kann kann man den Zustand eines Qubits verändern?“ das Wort „Qubit“ durch „Vektor“ ersetzt, so ist die Frage leicht zu beantworten. Die Zustände von Qubits können durch Multiplikation mit Matrizen verändert werden.

$$\vec{A} \cdot \vec{x} = \vec{y}$$

Es werden dabei nur unitäre Matizen zugelassen, diese sind komplex und quadratisch, wobei die Zeilen und die Spalten orthogonal sind:

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Jetzt wird Matrix H genommen und an das Qubit |0> multipliziert.

$$H|0> = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Nach Auswertung und durch expandieren des Vektors bezüglich der Einheitsvektoren kann man die typische Schreibweise für Qubits wiedererkennen.

$$\frac{1}{\sqrt{2}}(|0> + |1>)$$

Hier wurde das Qubit |0> genommen und mit einer „zulässigen“ Matrix multipliziert. Betrachtet man das Ergebnis, so ist zu erkennen, dass α und β den gleichen Wert haben. Damit wurde also ein Qubit, das sich im Zustand |0> befand in einen unbestimmten Zustand überführt, wobei man beim Messen dieses Elektrons mit der Wahrscheinlichkeit ½ eine 0 und mit der gleichen Wahrscheinlichkeit eine 1 als Ergebnis bekommt.

5 Die Struktur von Quantum Computing Algorithmen

Ein einfacher Algorithmus, der jedoch zeigt, wie mit Qubits gerechnet werden kann:

1. $|x> \leftarrow |0>$
2. $|x> \leftarrow \mathrm{H}|0>$
3. $\mathit{Messen}\,|x>$

Nun kann die grundlegende Struktur von QC Algorithmen angeschaut werden. Jeder QC-Algorithmus initialisiert zuerst seine freien Speicherstellen, die Qubits. Hier wird das Qubit |x> mit |0> beschrieben. Dies würde in einem echten Quantencomputer durch das Anstoßen eines Elektrons funktionieren, dadurch würde es einen Drall bekommen und sich um seine eigene Achse drehen. Diesen Umstand kann man als Vektor |0>, der in die Richtung des Drehmoments zeigt und als „Spin-Unten“ beschreiben. Das Qubit |x> speichert jetzt |0>. In Zeile zwei des Algorithmus wird eine Matrix an |x> multipliziert, wodurch der „Drall“ oder „Spin“ des Qubits wie im vorherigen Abschnitt manipuliert wird. Vom Zustand „Spin-Unten“ oder |0> verändert sich das Qubit und wird in den unbestimmten Zustand überführt. Hier kann man erkennen, dass die Wahrscheinlichkeit für das Erhalten einer 0 (oder 1) ½ beträgt. Im dritten und letzten Schritt wird wie bei jedem QC-Algorithmus die Messung durchgeführt, wodurch die gerade erst durch die Matrixmultiplikation entstandene Superposition zerstört wird. Durch nur ein Qubit wurde ein echter Zufallsgenerator programmiert, der mit echtem Zufall arbeitet. Zufall ist eine Eigenschaft von quantenmechanischen Objekten, und erfordert keiner weiteren Arbeit als Messen.

6 N-Bit Quantenregister

Ein Bit reicht nicht aus, deswegen sind Quantenregister der nächste Schritt.

$$R = \sum_{i=0}^{2^n-1} \alpha_i |\mathrm{i}>, \ \sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1 \text{ wobei } R \in \mathbb{C}^{2^n}$$

Bei n-Bit Quantenregister hat man nicht nur ein Bit, sondern wie der Name schon sagt n Qubits.

Für $n=1$ erhält man nach Ausführen der Summe die Darstellung in Kapitel 3.
Für $n=2$ wird es etwas komplizierter:

$$R = \sum_{i=0}^{3} \alpha_i |\mathrm{i}> \ = \ \alpha_0|0>+\alpha_1|1>+\alpha_2|10>+\alpha_3|11>$$

Es ist zu beachten, dass die Zahlen in der Dirac-Notation im Binärsystem notiert sind. Zu erkennen ist, dass das Elektronenpaar (zwei nebeneinander) nun vier mögliche Zustände hat.

Die Zustände |0>, |1>, |10>, |11> haben jeweils die Wahrscheinlichkeit α_i. Das System befindet sich dann in einer Superposition von vier Größen, es ist in allen vier Zuständen gleichzeitig und nur nach dem Messen bekommt man mit der Wahrscheinlichkeit α_i^2 die Zahl i zurück.

Führt man die Formel also n mal aus, so erhält man 2^n Zustände, in denen sich das System gleichzeitig befindet, wenn noch nicht gemessen wird. Das heißt, dass man mit n Bits einen Vektor darstellen kann, der die Dimension 2^n hat. Die α_i könnten als Informationsspeicher genutzt werden. Zu bemerken ist, dass ein n-Qubit-System immer nur 2^n als die Anzahl der möglichen Zustände hat.

Rechnen kann man mit diesem Vektor auch ohne Probleme. Dies geschiet dadurch, dass man die quadratische Hadamard-Matrix der Größe 2^n zur Manipulation des Systems nutzt.

Diese Matrix wächst auch wie die Dimension des n-Qubit-Systems exponentiell, damit man diese miteinander multiplizieren kann. Die Hadamard Matrix zur Manipulation eines n-Qubit-Systems kann z.B nach James J. Sylvester konstuiert werden:

$$H_{2n} = \begin{pmatrix} H_n & H_n \\ H_n & -H_n \end{pmatrix}$$

7 Tensorprodukt

Was geschiet aber genau, wenn mehrere Qubits „nebeneinander“ stehen? Für $n = 2$ kann man das noch zeigen. Es wird das Tenorprodukt der beiden Qubit-Systeme berechnet.

$$\begin{pmatrix} a \\ b \end{pmatrix} \text{TP} \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} ac \\ ad \\ bc \\ bd \end{pmatrix} \in \mathbb{C}^4 \quad \begin{pmatrix} a \\ b \end{pmatrix} \in \mathbb{C}^2 \quad , \quad \begin{pmatrix} c \\ d \end{pmatrix} \in \mathbb{C}^2$$

Hierbei wird die erste Zeile des ersten Vektors mit dem zweiten Vektor multipliziert, danach wird die zweite Zeile mit dem rechten Vektor multipliziert, die beiden entstehenden Vektoren werden übereinander geschrieben. Dabei erhält man immer einen Vektor der im Raum $\mathbb{C}^{2^n}$ ist, wenn man das Tensorprodukt von zwei Vektoren der Dimension n berechnet. Die Dimension des n-Qubit-Systems ist dann 2^n .[1]

8 Vom Quantum Computing zur Support Vector Machine

Seth Lloyd, der am MIT unter anderem an Quantum Support Vector Machines arbeitet, leitete seinen Vortrag bei Google TechTalk folgendermaßen ein: „Quantum Mechanics is about linear manipulation of vectors in large vector spaces“. Im darauf folgenden Satz ersetzte er das Wort „Quantum Mechanics“ mit „Machine Learning“. In beiden Bereichen arbeitet man mit sehr großen Datenmengen, wobei die Vektoren eine hohe Dimensionszahl haben.

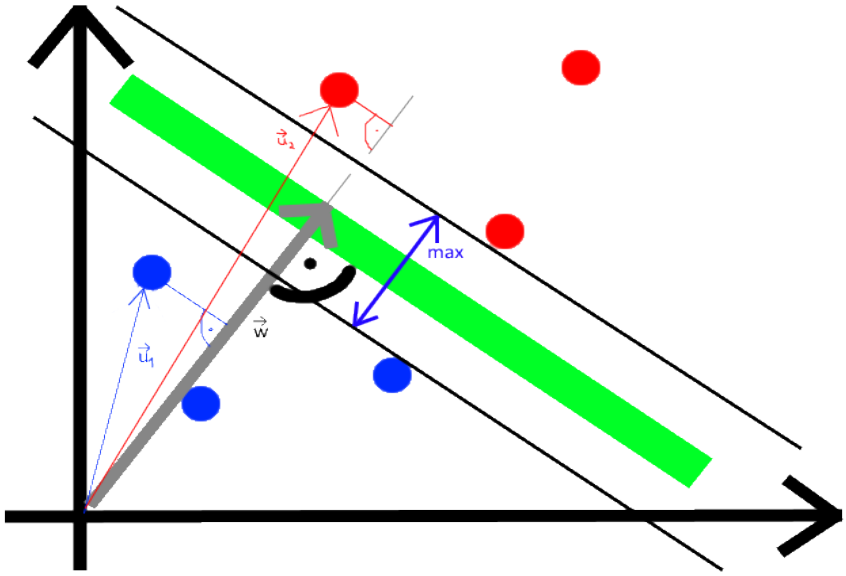
[1] Die Grundlagen des Quantum Computing von den Kapitel 1-7 orientieren sich am Buch Homeister (2013) S. 9-66

Dadurch, dass man im Quantum Computing einen Vektor der Dimension 2^n anhand von n Qubits darstellen kann und im Bereich Machine Learning große Datenmengen hat, könnte man sich diese Eigenschaft zu Nutze machen.

Im weiteren Abschnitt werden SVM erklärt und es wird gezeigt, wieso es Sinn macht, sich QC und ML zu verbinden.

Das Ziel von SVM ist es eine Hyperebene zu finden, die eine Punktmenge in zwei Bereiche unterteilt. Dabei soll die Ebene so gewählt werden, dass der Abstand von Ebene zu den beiden am nächsten liegenden Punkten maximal wird. Dazu wird die SVM mithilfe von Trainingsobjekten dazu gebracht, durch Beispiele zu lernen, ähnliche Datenmengen auf Basis der Lernerfahrung durch die Trainingsbeispiele zu klassifizieren.

Abbildung 1: Grundidee der SVM



Quelle: In Anlehnung an Patrick Winston Herbst 2010 Vorlesung MIT

Die Grundidee ist es, Vektoren wie $\vec{u}_1$ und $\vec{u}_2$ mit $\vec{w}$ zu multiplizieren und damit auf den Vektor $\vec{w}$, den Normalenvektor der Hyperebene, zu projezieren. Durch einen Vergleich des Skalarproduktes mit einer Konstante c kann bestimmt werden, ob der Punkt links oder rechts

von der Hyperebene liegt. Dies würde $\vec{w} \cdot \vec{u} \geqslant c$ entsprechen. Das c kann man subtrahieren und man erhält $\vec{w} \cdot \vec{u} + b \geqslant 0$. Das reicht noch nicht, denn man muss das b und das $\vec{w}$ bestimmen. Außerdem wird der zu maximierende Abstand zwischen Ebene und der Stützvektoren nicht berücksichtigt.

Eine Darstellung, die die „Straße“ um die Ebene berücksichtigt beitet folgende Modellierung:

$$\vec{w} \cdot \vec{x_+} + b \geqslant 1$$
$$\vec{w} \cdot \vec{x_-} + b \geqslant -1$$

Dadurch kann man unterscheiden, ob ein Punkt links oder rechts der Straße um die Hyperebene liegt. Diese Formeln kann man noch weiter vereinfachen, indem man die Variable y_i einführt, die folgende Eigenschaften besitzt:

y_i ist $+1$ für $\vec{x_+}$
y_i ist -1 für $\vec{x_-}$

Multipliziert man nun die beiden Gleichungen oben mit y_i , so ergeben sich:

$$y_i \cdot (\vec{w} \cdot \vec{x_+} + b) \geqslant 1$$
$$y_i \cdot (\vec{w} \cdot \vec{x_-} + b) \geqslant 1$$

Dabei sieht man, dass die beiden Ungleichungen gleich sind und man kann sie zu einer Formel zusammenfassen:

$$y_i \cdot (\vec{w} \cdot \vec{x}_i + b) - 1 \geqslant 0$$

Diese Formel klassifiziert Punkte unter Berücksichtigung der Breite des Bereichs um die Hyperebene.

Es fehlt noch die Bestimmung der Breite. Dies geschieht dadurch, dass man zwei Punkte aus der Punktmenge nimmt und sie als Stützvektoren, also die Punkte, die der Ebene am nächsten sind ausprobiert.

$$width = (\vec{x}_+ - \vec{x}_-)\frac{\vec{w}}{|\vec{w}|}$$

Die Breite kann mit einfachen mathematischen Mitteln berechnet werden. Es werden zwei Punkte aus der Menge der zu unterteilenden Punkte aus jeweils verschiedenen Seiten als Stützvektoren ausprobiert. Die beiden Punkte werden erst voneinander abgezogen, danach erfolgt eine Projektion auf den normierten Normalenvektor der Hyperebene.

Löst man die Formeln

$$y_i \cdot (\vec{w} \cdot \vec{x}_+ + b) \geq 1$$
$$y_i \cdot (\vec{w} \cdot \vec{x}_- + b) \geq 1$$

nach $\vec{x}_+$ und $\vec{x}_-$ auf und setzt diese in die Formel für die Breite ein, so erhält man:

$$width = \frac{2}{|\vec{w}|}$$

Diese Formel muss bezüglich $\vec{w}$ maximiert werden.

Um mathematisch besser mit dieser Formel arbeiten zu können, empfielt sich, folgende Umformung durchzuführen:
Statt

$$width = max(\frac{2}{|\vec{w}|})$$

kann auch

$$width = min(\frac{1}{2} \cdot |\vec{w}|^2)$$

berechnet werden.

Wenn man jetzt die Lagrangeformel anwendet, um den Ansatz für die Lösung des Optimierungsproblems aufzustellen, dann erhält man

$$L = \frac{1}{2} \cdot |\vec{w}|^2 - \frac{1}{2} \sum_i \alpha_i [y_i (\vec{w} \cdot \vec{x}_i + b) - 1] .$$

Es werden also die Nebenbedingungen, die in diesem Fall den auszuwertenden Trainingsbeispielen entsprechen, von der zu maximierenden Funktion abgezogen.

Nach den Operationen $\nabla L = 0$, also $\frac{\partial L}{\partial \vec{w}} = 0$ und $\frac{\partial L}{\partial b} = 0$ erhält man nach Einsetzen der Ergebnisse aus diesen Berechnungen in den Ausdruck

$$L = \frac{1}{2} \cdot |\vec{w}|^2 - \frac{1}{2} \sum_i \alpha_i [y_i (\vec{w} \cdot \vec{x}_i + b) - 1]$$

die umgeformte Variante

$$L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \vec{x}_i \vec{x}_j \, .$$

Hierbei kann man erkennen, dass das Lösen dieser Lagrangefunktion nur von Skalarprodukten der Trainingsbeispiele abhängt.[2]

Was passiert aber, wenn man keine Ebene finden kann ? Manchmal lassen sich Punktmengen in einem Raum mit einer festen Dimensionszahl einfach nicht durch eine Ebene trennen. Es wird dann versucht, eine Hyperebene für diese Punktmenge zu bestimmen, die nur in einem Raum existiert, der eine höhere Dimensionszahl besitzt. Diese Dimensionen können sehr hoch werden. Bei der Rücktransformation dieser Ebene in den Raum mit niedrigeren Dimensionen wird diese nichtlinear. Diese Dimensionen können sehr hoch werden.

Man wäre dann wieder bei „large vectors in large vector spaces“. Hier kann mithilfe den Prinzipien des Quantum Computing enorm Rechenzeit eingespart werden.

Fazit - Was kann QC für ML bieten?

- Mit nur n Bits können Vektoren der Dimension 2^n dargestellt werden
- Arbeiten mit Vektoren der Größe 2^n , z.B Skalarprodukt

[2]Vgl. Patrick Winston (2010) Vorlesung SHerbst

Dadurch kann ein exponentieller speed-up erreicht werden. Es können also Skalarprodukte mit Vektoren der Dimension 2^n auf n Bits berechnet werden.

Durch die Möglichkeiten, die QC bietet, kann man die aktuelle polynomielle Zeitschranke von $O(NM)$ für das Trainieren und für die Klassifikation für ML-Algorithmen auf $O(\log(NM))$ reduzieren.[3]

9 Schlusswort

QC ist noch in der Anfangsphase, aber es gibt schon kleine Erfolge. Wissenschaftler an der UC Santa Barbara haben es geschafft, mit einem Quantencomputer die Zahl 15 in ihre Primfaktoren zu zerlegen.[4] Das zeigt, dass es nicht unmöglich ist, einen solchen Computer zu bauen. Jedoch ist es sehr schwer, mit Qubits zu arbeiten, weil sie wegen ihrer Störanfälligkeit gegenüber äußeren Einflüssen sehr stark von der Außenwelt abgeschottet werden müssen.

Wenn jedoch in Zukunft die nötigen Techniken entwickelt werden, dann wird es sicherlich eine Revolution in der Informatik geben, denn den Logarithmus einer polynomiellen Zeit zu nehmen ist Bezogen auf Zeitschranken unschlagbar.

[3] Vgl. Lloyd u.a. (2014) S. 4
[4] Vgl. http://www.news.ucsb.edu/2012/013340/ucsb-researchers-demonstrate-153x5-about-half-time (10.10.2015)

Literaturverzeichnis

Adams, Allen (2013): Vorlesung Quantum Physics I, MIT

Homeister, Matthias (2013): Quantum Computing verstehen, 3. Aufl., Wiesbaden

Lloyd, Seth/Mohseni, Masoud/Rebentrost, Patrick (2013): Quantum algorithms for supervised and unsupervised machine learning, MIT, Google Research

Lloyd, Seth/Mohseni, Masoud/Rebentrost, Patrick (2014): Quantum support vector machine for big data classification, MIT, Google Research Venice

Winston, Patrick (2010): Vorlesung (Herbst), MIT Artificial Intelligence

.http://www.news.ucsb.edu/2012/013340/ucsb-researchers-demonstrate-153x5-about-half-time (10.10.2015)