Journal of Machine Learning Research 1 (2021) 1-48 Submitted 12/9; Published 12/9

Introduction to Quantum Machine Learning

Theodore Dyer TDYER4@JHU.EDU
Johns Hopkins University, Whiting School of Engineering
Quantum Computing (EN 605.728)
David Zaret - Fall 2021

Editor: Theodore Dyer

Abstract

This paper details an introduction into the topic of quantum machine learning. Given that to many audiences this is a very advanced subject, I will not presuppose any knowledge of machine learning or quantum topics. Thus, included is a brief introduction to quantum machine learning on a broad scope, followed by an explanation of the necessary topics to understand in baseline machine learning and baseline quantum computing. Finally we will then explore quantum machine learning, including its limitations, power, and potential future applications.

Keywords: Quantum Computing, Machine Learning, Neural Networks, Data, Cross-validation, Evaluation Metrics, Learning Algorithms, Supervised Learning, Unsupervised Learning, Regression, Clustering, Processing, Qubits, Quantum Scaling, Quantum Advantage, Hybrid Quantum-Classical Models

1. Quantum Machine Learning Overview

Quantum Machine Learning - although this phrase may sound intimidating or confusing at first, through learning about the following foundational concepts I hope you will come to understand that it is in fact relatively straightforward, with extremely interesting potential at the crossroads of two exciting fields within the computing space.

1.1 What is Quantum Machine Learning?

In short, quantum machine learning refers to the application of quantum computing technology to the field of machine learning. If this explanation is only leaving you more confused - the following few sections of this paper should provide you with enough information on the two major building blocks of quantum machine learning (classical machine learning, and quantum computing) to explore in more detail quantum machine learning itself.

1.2 Why Quantum Machine Learning?

Machine learning, a field that is generally only considered in the realm of classical computing, has a few characteristics as a subject that make it an ideal candidate to be optimized by quantum systems. One big limitation with machine learning in the modern era is the sheer size of problems we want to solve. Many companies in the past decade have pivoted their main profit generating business component to be data itself, and the amount of data

being collected in every facet of life is staggering.

This is where quantum computing comes in - with this surplus of data, there is massive potential for machine learning to identify meaningful and actionable trends. With some large scale simulations and model training, required computational power is simply not feasible with classical computing. Being able to leverage clever quantum solutions to break down computational complexity could be an extremely lucrative way to work on problems not even conceivable without quantum innovation.

Aside from this potential solution to some scaling complexity issues with classical systems, quantum systems are potentially even able to achieve higher prediction accuracy and generalization power when provided the same problem as a classical system.

2. Machine Learning

At this point no matter what your background is in education or professional life, machine learning is a term that you have likely heard in conversation. Additionally, machine learning is a very common buzzword thrown around in the tech world today - however many people have a strong misconception of what this term actually refers to.

Contrary to what some people believe, machine learning as a concept is not what is going to bring us "Skynet" from the Terminator series - and in fact in most cases the term machine learning is just a fancy layer of abstraction on top of simple statistics. That being said, there are a number of foundational machine learning concepts that will be important for you to understand when considering quantum machine learning at a theoretical level. Let's now take a look at some inner workings of classical machine learning.

2.1 Machine Learning Objectives

At the most basic and high level, the goal of machine learning is to extract patterns and meaning from data in order to make predictions about any future, unseen data that you may encounter. Machine learning and the underlying math has been around for a very long time, thus the recent surge of interest in machine learning has been sparked by a few key points. First, we now live in an era where data is everywhere, and where there is data there are patterns to find within this data to generate business advantages, thus there is machine learning to find these patterns. Second, until recently the computational power required to train some machine learning models on large scale datasets has just not been feasible - but we are no longer faced with this problem especially with the broad acceptance of cloud computing as a commonplace tool.

Now you may be asking yourself, what are some of the common types of questions you might want to answer with machine learning? And that itself is a hard question to fully answer considering the obscurities with different model types that exist within machine learning, but the following are a few.

2.2 Machine Learning Paradigms

2.2.1 Supervised Learning

Linear Regression - Given data of houses in a neighborhood (number of bedrooms, number of bathrooms, square footage, year of construction, location, etc) - what should I price my

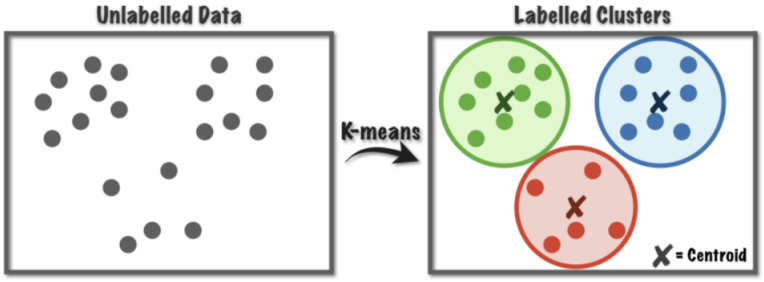
house at when listing it on the market? This type of question is referred to as linear regression, because given any number of features we are attempting to predict a continuous value. (in this case a dollar value in a range).

Logistic Regression - Given healthcare data of elderly patients (age, gender, height, weight, prior conditions, relatives, etc). Predict whether or not a patient will be at risk of contracting a particular disease. This type of problem is very common in many industries - where the general format is taking any number of input features, and attempting to predict a boolean result.

Supervised learning, illustrated by the examples above in linear/logistic regression - refers to a machine learning problem where within your input data, you are given the "solution" or target value of what a specific data entry should output: for example: (2 bedroom, 3 bathroom, 1500sqft, 2003, TARGET = 500,000 USD). So generally the goal with supervised learning is to follow the very human-intuitive notion of learning by example. Put into simple English, if you see what price a specific house and model should have, within a supervised learning problem you remember that price for the future when you see similar houses.

2.2.2 Unsupervised Learning

K-means Clustering - Suppose you're working with a database of plants. Each plant with features such as (petal width, petal length, sepal width, etc), but none of the plants have labels or 'target' variables that describe what the type of plant is. It is intuitive to think that despite this there will be underlying groupings within the data of plants that are the same species. Each of these entries corresponding to the same species will likely have similar values for each of the above attributes. These types of problems are the motivation for the k-means clustering algorithm, which is used to generate groupings among an unlabeled data set.



Unsupervised learning is just the opposite of supervised, generally in supervised learning you are learning from example by observing a data entry's label or target variable and associating each of its other features with that label. Unsupervised learning on the other hand does not have this ability to associate information with labels, so the alternative is to associate information with semi-arbitrary groupings selected by the model.

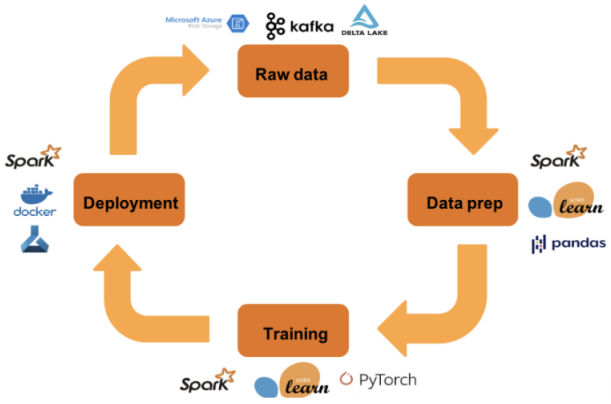
Now there are even more specific types of models than this, for example k-nearest neighbors which is a very popular tool doesn't even use a parametric definition of the underlying model. For now this will be enough to begin to understand the basics of quantum

machine learning which we will get into later on.

Next, it is important to learn about the machine learning "pipeline" or in other words, how we can go from data and a question, to a tangible solution and prediction engine.

2.3 Machine Learning Data Pipeline

In implementation there are many different choices you can make for a particular machine learning solution, but the general process will look like the following regardless of model selection or learning paradigm.



2.3.1 Sourcing Data

Given that machine learning entirely revolves around data, it makes sense that the first step in implementing a machine learning solution would be to obtain our data. This step can either be extremely easy or extremely difficult based on the problem and scale of operation. In some cases this step will be a simple query to a database with a programming language such as SQL, alternatively this step could entail web scraping (The Beautifulsoup Python library is a good example of this), various hacking techniques, or pulling information from APIs. Putting this into the frame of machine learning in industry: large companies that have the resources to do so will often hire people specifically for this role under the title of data engineer.

2.3.2 Data Prep

Often whenever data is received it is in a format that is not acceptable to plug directly into a machine learning model. For example: many models cannot handle string data, missing data, or certain ranges of values. Thus, this step generally includes either dropping data entries with missing fields, removing outliers, and standardizing values (in many cases a value between 0-1 is preferable over a value in an arbitrarily large continuous range). This step in the pipeline can be quite difficult depending on where the data is pulled from, generally values from an internal company database can be reliable, but especially when sourcing data from the internet or unknown sources it is extremely important to be thorough with cleaning and preparing data. This step is also generally when Exploratory Data Analysis (EDA) will be performed in order to identify potential trends within data and generate questions - which we then solve in the next steps with machine learning solutions.

2.3.3 Training

Now for the core of the machine learning process, the training of a model. This step can vary dramatically based on the underlying model selection as you can likely imagine, but the basic process is generally always the same. Assuming we have a cleaned and prepped data set, the next step is to split this data into two partitions - about 80 percent of the data will be declared the 'training set' and 20 percent of the data will be declared the 'test set'. Following this, we scan through every entry in our training set and in many cases the actual computational process performed for each entry is tweaking weights or scalars held by an underlying object. Sparing details here as for each model this process is vastly different - if any particular model discussed in this paper piques your interest I would highly recommend further research on that particular model's training process as I personally find this step to be extremely interesting.

Additionally at the end of the training cycle, it is common to take our trained model and practice making predictions with the remaining 20 percent of the data originally partitioned as our 'test' set, in order to determine how effective and accurate our predictions are. If we are not satisfied with our results we can tweak parameters or perhaps switch to a different model. Once we have achieved satisfactory results, the next step in the machine learning cycle is to deploy our solution to make real world predictions.

2.3.4 Model Deployment

Now that we have a model the real value from machine learning can be leveraged, namely making predictions on unseen data. The step of deployment will again depend on context. This could take the form of a prediction engine deployed in a script on a website, or an internal company tool. An example of a very simple deployed model can be found at the following link. *https://theodoredyer.github.io/brainjs/brain.html*. In the linked example the model learns dynamically, with each user input it adjusts internal neural network weights.

2.4 Processing Paradigms

Now that you have a basic understanding of the types of problems we can solve with machine learning and the process we must follow to begin to solve these problems, let's now look at where exactly machine learning shines and what its underlying computational patterns look like.

Any problem able to be solved by a computer involves two things - data, and an instruction or set of instructions. Given that there are two components here, there are 4 resultant paradigms of problem types - machine learning leverages the power of SIMD processing explained below.

2.4.1 SISD (Single Instruction Single Data)

This pattern includes most basic computer programs, or more specifically computer programs that do not employ multi-threading or parallel processing. Most simple programs follow a linear instruction stream executing line by line in a standard fashion on a clearly defined discrete data stream.

2.4.2 SIMD (Single Instruction Multiple Data)

These types of problems involve applying a single instruction to a large pool of data. A very common example of this exists in graphics processing: if in a virtual environment a light source is added, every single pixel in that environment must be modified by the light in some way. Thus the same instruction (modifying light values of pixels) must be performed on an extremely vast array of data points or pixels. This type of problem prompted the need for GPUs, which specialize in exactly this domain (SIMD), and this paradigm is where machine learning also resides. In training a machine learning model, we have this exact same scenario. We know what operation we need to perform on a point of data, and we need to employ this operation to an entire data set. For those wondering, this is why machine learning leverages GPUs effectively.

2.4.3 MISD (Multiple Instruction Single Data)

This paradigm applies to problems that include what many people may know as parallel processing or multithreading. In some cases there may exist some shared resource or data stream (for example a server) that many different entities need to interact with at the same time, each with their own stream of instructions or queries.

2.4.4 MIMD (Multiple Instruction Multiple Data)

These systems are those with multiple processors allowing for parallel processing, but not limited to just one information stream, examples of this can be harder to find in common applications, so for now this is all that is necessary to understand.

So with all of these paradigms in mind, we are able to understand where machine learning finds its niche of computational power, namely in the SIMD space. Because there have been such vast advancements in GPU technology, the efficiency and practicality of machine learning has also risen as a result. Looking forward this is exactly where my main concern with quantum machine learning lies, essentially what has allowed machine learning to rise to prevalence is the existence of compute resources and with comparable quantum systems not even scratching the surface of physical qubit availability it may be quite some time until quantum machine learning sees use.

2.5 Machine Learning's Limitations

Everything discussed so far has highlighted the positive aspects of machine learning, we're now able to solve problems previously thought to be impossible and creative new uses of the technology are cropping up every single day. However, machine learning is not without its drawbacks, and the vast majority of these additionally apply to quantum machine learning.

2.5.1 Problem Setup

There are many cases in which people simply try to force machine learning solutions into domains that do not call for it. Even when machine learning can be applied, it is commonplace for people to assume the most complicated machine learning models are the best regardless of problem type. This poses a problem as for example oftentimes linear regres-

sion can be much better suited to a problem that some decide to solve with a much more complicated model.

Additionally, there are simply some real world phenomena that cannot be modeled by data despite how hard we try, "A neural network does not understand Newton's second law, or that density cannot be negative — there are no physical constraints." (*The Limitations of Machine Learning*, Stewart). One key component of a successful machine learning solution is a human being with expert subject knowledge working alongside developers to ensure certain parameters and restrictions are set in place. For example: one might end up in a situation where a model predicts a value that simply does not have a real world counterpart (negative areas, negative distances, etc).

At the end of the day machine learning solutions are only as good as their setup, and properly outlining the problem domain and carefully monitoring model weights and predictions can be just as important as defining the correct problem in the first place.

2.5.2 Interpretation

Machine learning in general is a concept that exists at the edge of many other subjects - namely to understand and be able to apply machine learning you must first understand statistics, mathematics, and computer science to a fairly high extent. Additionally, because machine learning as a concept has been around for such a long time, there has been a large amount of innovation and advancement made in the theory behind some different types of machine learning models.

These two facts lead us to an un-ignorable flaw with machine learning when it comes to finding practical business applications. Sometimes specific machine learning solutions can simply be too complicated to explain to people who do not have similar backgrounds to the developers. For example, why would a business higher-up without a heavy tech background decide to invest money and resources into a problem solution that they don't even understand? How can you pitch implementing a feedforward neural network utilizing backpropagation to a client who has not taken a math course in the last 30 years? The answer is that sometimes the right solution is just to not utilize advanced machine learning in the place of something that is more easily explainable to a client. This is said with the caveat that in general many people in the tech world are gaining general familiarity with machine learning.

2.5.3 Compute Power

We just got done exploring how machine learning has been able to leverage compute power to find more practical applications today, so why am I now listing this as a flaw? Moving forward, the production of data will certainly now slow down, "Consider that 90 percent of the world's data has been produced in just the last two years. This explosion of information is known as "Big Data," and it is completely transforming the world around us," (*Big Data and What it Means*, Bradshaw).

This rapid increase in data production, if it follows a similar trend, will far outpace the trend described in Moore's law in terms of compute power. Inevitably we will reach a point where we are faced with problems that again are not feasible to be solved with classical

machine learning, requiring a more sophisticated solution which will hopefully be found in quantum machine learning.

3. Quantum Computing

You may be wondering, after all of the prior mentions of classical computing and classical machine learning, why distinguish classical? What exactly is the alternative I have been distinguishing from? The alternative is quantum computing and quantum machine learning. Given quantum computing itself is a very complicated field to begin to understand, I will spare many details in this description and keep it to a minimum needed to understand the following quantum machine learning concepts.

3.1 Qubits vs. Bits

Classical computers utilize at their lowest level what is referred to as 'bits', an entity that is able to exist in one of two states, either 0 or 1, and is physically represented by an electrical signal of high or low. With current developments in classical computer hardware we are able to store trillions of bits in memory systems. Additionally, the computational power of classical computers has generally followed what is referred to as Moore's law - stating that the number of transistors on an integrated circuit doubles about every two years (since 1965). Although this is slowing down in modern days, this is to say that we have reached vast improvements in computational power, and there now exist very few problems that can be solved with brute force by adding more bits at a problem. New advancements in being able to solve problems will likely only be made with different systems, ie: quantum systems.

Quantum systems do not use bits represented by simple electrical signals on wires at all, and in their place they instead utilize 'qubits'. At the core of the system, qubits serve the same purpose as bits in classical systems as they are the baseline entity able to store pieces of information. The key difference between a bit and a qubit is that a qubit is able to exist in a 'superposition' of 0/1 and is essentially able to represent both 0 and 1 at the same time.

3.2 Quantum Scaling

One of the main draws to implementations of quantum computing, and the motivation for further development of quantum machines is the potential for quantum systems' computational power scaling. There exist already today some problems that are simply not feasible to compute using classical machines, problems for which computational complexity is increased by some exponential or other factor. For example, "It would take a classical computer around 300 trillion years to break a RSA-2048 bit encryption key. That's why we all feel that we are "safe" from these attacks. But it does illustrate that the foundation of all of our cryptography is not guaranteed to be secure, it is only known to be really, really hard to solve," (*Breaking RSA Encryption*, Baumhof). The reason that this problem is so difficult to solve with a classical computer is due to the fact that it requires exponential computational complexity, specifically in the realm of (2 raised to the power of 2048) operations.

Where quantum computing can be extremely powerful is in situations exactly like this. With utilization of Shor's algorithm, this problem of cracking RSA encryption can be broken down into a polynomial computational complexity problem. This means that not only is this problem feasible to be solved in our lifetime, but once sufficient quantum machines have been built - able to be solved almost instantly.

One might imagine that in the problem domain of machine learning, which will continue to grow in requiring more computational power as simulations and model training begin to consume more data, the ability to reduce complexity of training could become extremely desirable if a quantum solution were able to be found.

3.3 Physical Quantum Computers

Because at the core of quantum computing is the qubit instead of a classical bit, one might imagine that physical quantum systems must be quite different from classical computers and that is indeed correct. Physically representing a bit is quite simple given that it can only exist in one of two states - a simple electrical signal will suffice, this is certainly not the case with qubits.

It turns out that qubits in practice can be implemented many different ways, such as trapped ions, neutral atoms, or superconducting qubits (those used by IBM which is currently the leader in developing quantum systems). The fact that quantum systems require a completely different foundation in physical implementation leads to one problem being that we cannot re-use any technology previously built for classical machines to accelerate the development of quantum machines. Thus for the foreseeable future there will be a race between parties to build bigger and better quantum machines.

3.4 Quantum Limitations

Now as previously stated quantum machines in their physical implementation are still very much in the early stages of development. This means that even though for example we know that RSA-2048 bit encryption can be cracked with a quantum computer quite easily, and the process for doing so has already been figured out, we simply do not have a quantum computer large enough for this to be possible yet.

Another complication with the physical implementation of quantum machines are the conditions in which they must operate. Qubits can be quite finicky, even measuring the state of a qubit disturbs the state of a qubit, so how can we possibly maintain the state of a physical implementation of one? Of course there are many ways this problem could be approached, but in practice so far it has been addressed by keeping them in extremely cold environments. Even with precautions in place, quantum computers have high error rates generally requiring the use of a partition of qubits to monitor errors. For these reasons I personally find it extremely unlikely that anyone in our lifetime will experience the availability of a personal quantum computer.

Additionally, the ability to break an exponential complexity scaling problem into one of polynomial complexity is quite an impressive feat, however, it is not going to apply to all problems. Given that the application of quantum algorithms is still a heavily researched subject it is simply to impossible to model certain problems in a context for which a quantum computer will be able to achieve super-polynomial speedup. For some exam-

ples check out the following link to get a sense for which this speedup is indeed possible: *https://quantumalgorithmzoo.org/*

4. Quantum Machine Learning

Earlier I stated that quantum machine learning is the application of quantum computing to the field of machine learning. Hopefully now this is beginning to make more sense and the pairing of these two subjects together seems intuitive.

4.1 Motivation for Quantum Machine Learning

As we now know, machine learning is a subject with very high potential when applied to the right data set and to the right type of problem, as is quantum computing. There are two main areas where quantum applications have the potential to improve machine learning solutions.

First, looking at the sheer scale of some classical machine learning problems, for example Google's "... baseline model for JFT was a deep convolutional neural network that had been trained for about six months using asynchronous stochastic gradient descent on a large number of cores. This training used two types of parallelism," (*Distilling the Knowledge in a Neural Network*, Hinton, Vinyals, Dean). Six months is an extremely long time to spend training a model before it even sees practical use, although with the prior stipulations about quantum computing not always being able to provide speedup, the potential it could have when paired with specific types of models could be extremely valuable, considering problems of this scale to indeed exist.

Second, when framing a problem in terms of a quantum solution, many preconceived notions about how problems should behave go out the window. Quantum systems allow us to interpret generalizations and patterns within a data set which are simply not possible to detect by classical systems. This potentially leads us to be able to make better predictions across the board and achieve higher prediction accuracy.

4.2 Quantum Advantage

Summarizing the above: the base goal of implementing quantum machine learning is to answer the question of whether or not we can achieve quantum advantage by doing so. Simply put, "The idea of quantum advantage is typically phrased in terms of computational advantages. That is, given some task with well defined inputs and outputs, can a quantum computer achieve a more accurate result than a classical machine in a comparable runtime?" (*Quantum Machine Learning and the Power of Data*, McClean, Huang). I would expand on this to say that quantum advantage also includes the ability of a quantum computer to achieve similar accuracy to a classical system while at the same time utilizing much less run-time.

4.3 Quantum Machine Learning Models

As there are many different ways to represent even a single qubit in the context of quantum computing, you will not be surprised to find out that there are many ways to represent and utilize a quantum machine learning model. The following discussion will be based on

the structure of a model introduced by Google's TensorFlow Quantum (A public quantum machine learning library), which is at the forefront of quantum machine learning research and development.

At the most basic level, a quantum machine learning model is a machine learning model that is able to represent data and generalize information from data using quantum mechanics as its basis. One approach to this is to utilize quantum data in combination with hybrid quantum-classical models.

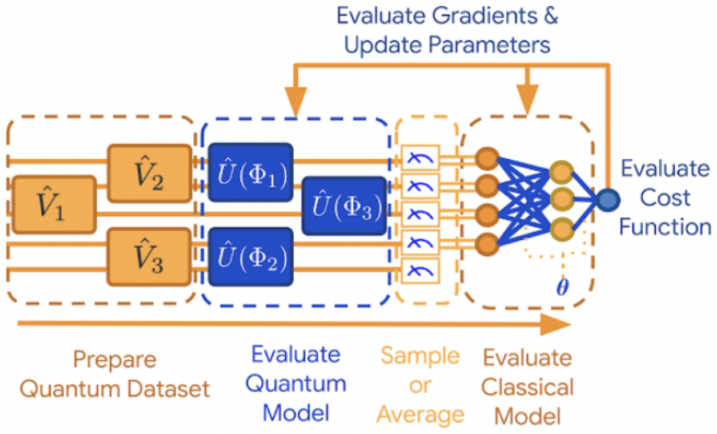
4.3.1 Quantum Data

Quantum data is simply data which possesses the quantum qualities of superposition and entanglement, such data can be created, or simulated on a quantum processor. Quantum data can tend to be quite noisy given the properties it is associated with, but that is not necessarily a bad thing, as it is said, "applying quantum machine learning to noisy entangled quantum data can maximize extraction of useful classical information" (*Announcing TensorFlow Quantum*, Ho, Mohseni).

4.3.2 Hybrid Quantum-Classical Models

Because of limitations with modern quantum processors being quite limited, much of what has been done in regards to present day quantum machine learning research has been done utilizing whatever quantum processing resources are available in tandem with classical processors. Specifically in regards to TensorFlow Quantum's platform - machine learning in general is already a concept in which parallelism is extremely lucrative for training and utilizing models (especially across different types of processors - namely GPUs and CPUs for machine learning), thus they already have framework in place to implement such quantum-classical parallelism.

With these concepts in mind we can continue with how TensorFlow Quantum is able to represent larger quantum machine learning models, and what their pipeline looks like.



A high-level abstract overview of the computational steps involved in the end-to-end pipeline for inference and training of a hybrid quantum-classical discriminative model for quantum data

4.4 TensorFlow Quantum Neural Network Example

The following steps refer to the above figure.

4.4.1 Data Prep

Data prep for a quantum neural network is different that what you are likely thinking of for traditional machine learning, although it does serve essentially the same purpose. In this step, quantum data is initialized as a group of tensors, which can be thought of as n-dimensional arrays of numbers, with each data tensor represented by a quantum circuit written in Cirq (quantum programming language). Tensors are then executed by TensorFlow to generate a quantum dataset.

4.4.2 Evaluating the Quantum Network Model

This step echoes closely what is done in a classical system in terms of assembling a structure to extract information from classical data, however, in the quantum context these models are dealing with quantum data which may be entangled, and must extract information from qubits into a classical representation for the rest of the pipeline.

4.4.3 Sampling

Taking a measurement on a qubit or piece of quantum data will result in classical information, which follows the process of taking a sample from a random variable in the context of classical mathematics.

4.4.4 Classical Post-Processing

The remainder of the pipeline follows the traditional/classic, neural network format of data processing, avoiding specifics for now.

4.5 When to use Quantum Machine Learning

As previously stated, a limitation of both machine learning and quantum computing is that the problem domain needs to be just right for both of these fields to coincide and both provide value. A way we can determine if both of these criteria have been met, developed by Google quantum machine learning researchers, is to utilize quantum feature mapping / kernel methods to translate data sets into classical and quantum counterparts. Before we can delve into the specifics of this process let's first explore the foundational concepts addressed therein.

4.5.1 Feature Mapping

Features of a data set or data entry simply refer to the 'columns' or attributes of that data point that are not what we are trying to predict: for example when looking at housing data, the features might include (number of bedrooms, number of bathrooms, square footage, etc) and the target feature would then be price. Tieng this back to the concept of feature mapping, the objective of which is to take features of data entries and translate them into a format that is able to be represented on a graph, or visually in general.

4.5.2 Kernel Functions

Kernel functions at their core are simply a way to take and transform data into intermediate values that better suit a given problem. For example: if an input existed in the form of a non-linear pattern, we can translate this data into a linear form based on some set of rules defined in a kernel function. The uses of these can vary quite widely, but the main benefit is to be able to translate data into arbitrary bases or representations to meet requirements or intermediate needs.

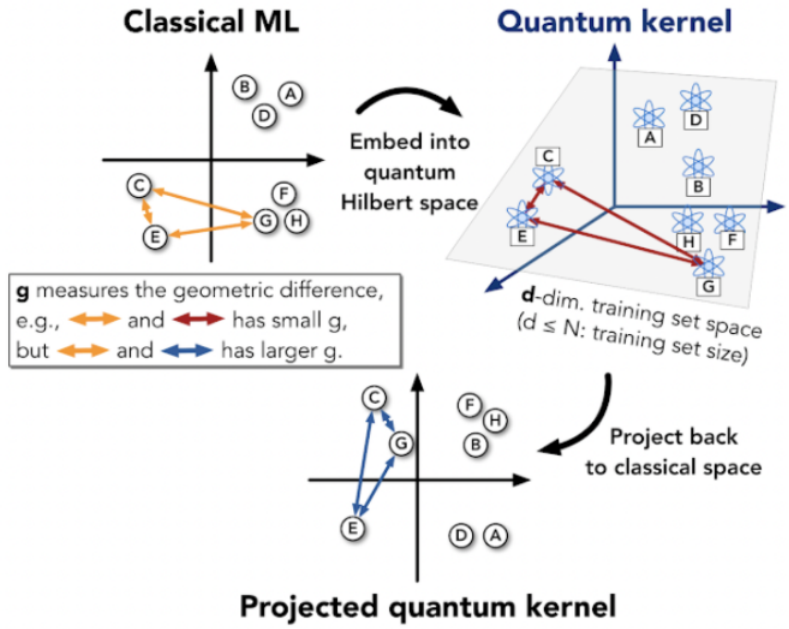
4.5.3 Testing for Quantum Advantage

With all of this in mind, we can boil down the question of "when should we use quantum machine learning" to the analysis of when quantum advantage is present. We're now going to analyze a method created by the aforementioned Google quantum machine learning researchers behind detecting this advantage - which they refer to as the 'Geometric Test for Quantum Learning Advantage'.

The first step of this process is to create a "quantum embedding" of our dataset, which is essentially a representation of our feature space (range of values a dataset can hold) acceptable to our quantum machine, along with an evaluation of this data corresponding to a quantum kernel method. This evaluation, and the use of quantum kernels in general, is simply the application of any classical kernel method to change the base of a feature space, but evaluated on the quantum embedding of the data as opposed to the baseline classical representation of our data.

Now with this quantum embedding, some arbitrary kernel function, and dataset in question, the test seeks to compare against a simple classical embedding over a range of possible target values or output predictions. The test utilizes a geometric constant 'g' to define the distance between the feature-mapped results of the quantum and classical embeddings of the data, and based on the value of this constant, determine if a quantum solution is worthwhile for the given problem.

The first experiments of this test revealed that in general, quantum kernels, or kernels applied to quantum acceptable mappings of input data, valued memorization over actual generalization gained from data. Thus, these methods perform worse than expected on test sets - a solution found for this involved after applying the quantum kernel to then translate back into a classical data space (the new classical representation now referred to as the projected quantum kernel) - illustrated by the following figure.



Now one might be asking at this point: why even utilize a quantum kernel in the first place if we're simply going to translate back into a classical representation at the end either way? The answer to this is in the fact that even when projecting back into the classical data space researchers found that they were able to preserve some aspects of quantum geometry not able to be simulated classically, hence the projected quantum kernel was by no means the same as the original classical model as illustrated in the above feature mapping.

In conclusion, the utilization of the projected quantum kernel when applied to the right problem domain is said to lead to, "improved ability to describe non-linear functions of the existing embedding, a reduction in the resources needed to process the kernel from quadratic to linear with the number of data points, and the ability to generalize better at larger sizes," (*Quantum Machine Learning and the Power of Data*, McClean, Huang).

This is exactly what we were looking for: in the case of some data sets and problems, we are able to observe both pre-discussed portions of quantum advantage. The researchers behind this test do carry on to state that this is generally not the case for most problems, the type of data available entirely shapes what solution is going to be effective, but they have concluded that in some cases with proper resources available the introduction of this projected quantum kernel does have positive performance implications and could be a worthwhile tool when qubits become a less scarce resource.

5. Conclusion

Quantum machine learning is currently a heavily researched subject that for the reasons outlined above, has extremely high upside potential. With both the ability to reduce computational complexity requirements and increase accuracy of predictions - the future of quantum machine learning has promise.

Unfortunately in modern times, it remains at just potential and has yet to see real heavy

business application - which I imagine will be the case for quite some time. Given that it is just not possible for nearly anyone to access quantum compute resources, I imagine that the first examples of quantum machine learning that see use in industry will come as a result of Google / IBM renting out their quantum machines in order to train models for others. Given the potential of this technology, I would not be surprised if in 5 years from now we will be answering questions with quantum machine learning that we never thought possible.

References

[1] *Quantum Machine Learning and the Power of Data*
Jarrod McClean, Hsin-Yuan Huang (June 2021), Google AI Blog.
https://ai.googleblog.com/2021/06/quantum-machine-learning-and-power-of.html

[2] *Power of Data in Quantum Machine Learning*
Hsin-Yuan Huang, Michael Broughton, Masoud Mohsent, Ryan Babbush, Sergio Boixo, Hartmut Neven, Jarrod McClean (May 2021), Nature Communications
https://www.nature.com/articles/s41467-021-22539-9

[3] *Exploring Quantum Neural Networks*
Jarrod McClean, Hartmut Neven (December 2018), Google AI Blog
https://ai.googleblog.com/2018/12/exploring-quantum-neural-networks.html

[4] *Announcing TensorFlow Quantum*
Alan Ho, Masoud Mohseni (March 2020), Google AI Blog
https://ai.googleblog.com/2020/03/announcing-tensorflow-quantum-open.html

[5] *The Limitations of Machine Learning*
Matthew Stewart (July 2019), Towards Data Science
https://towardsdatascience.com/the-limitations-of-machine-learning-a00e0c3040c6

[6] *Big Data and What it Means*
Leslie Bradshaw, U.S. Chamber of Commerce Foundation
https://www.uschamberfoundation.org/bhq/big-data-and-what-it-means: :text=In%20the%2021st%20century%2C%20digital,transforming%20the%20world%20around%20us

[7] *Breaking RSA Encryption - an Update*
Andreas Baumhof (June 2019), Quintessence Labs
https://www.quintessencelabs.com/blog/breaking-rsa-encryption-update-state-art/: :text=It%20would%20take%20a%20classical,RSA%2D2048%20bit%20encryption%20key

[8] *Distilling the Knowledge in a Neural Network*
Geoffrey Hinton, Oriol Vinyals, Jeff Dean (March 2015), University of Toronto Dept. Computer Science
http://www.cs.toronto.edu/ hinton/absps/distillation.pdf