Quantum Expectation-Maximization for Gaussian mixture models

Iordanis Kerenidis [1 2] **Alessandro Luongo** [3 1] **Anupam Prakash** [2]

Abstract

We define a quantum version of Expectation-Maximization (QEM), a fundamental tool in unsupervised machine learning, often used to solve maximum likelihood (ML) and Maximum a posteriori (MAP) estimation problems. We use QEM to fit a Gaussian mixture model, and show how to generalize it to fit mixture models with base distributions in the exponential family. Given quantum access to a dataset, our algorithm has convergence and precision guarantees similar to the classical algorithm, while the runtime is polylogarithmic in the number of elements in the training set and polynomial in other parameters, such as the dimension of the feature space and the number of components in the mixture. We discuss the performance of the algorithm on a dataset that is expected to be classified successfully by classical EM and provide guarantees for its runtime.

1. Introduction

Over the last few years, the effort to find real-world applications of quantum computers has greatly intensified. Along with chemistry, material sciences, finance, one of the fields where quantum computers are expected to be most beneficial is machine learning. Several different algorithms have been proposed for quantum machine learning (Wiebe et al., 2017; Harrow et al., 2009; Subaşı et al., 2019; Farhi & Neven, 2018; Montanaro, 2016; Biamonte et al., 2017; Chakrabarti et al., 2019), both for the supervised and unsupervised setting, and despite the lack of large-scale quantum computers and quantum memory devices, some quantum algorithms have been demonstrated in proof-of-principle experiments (Li et al., 2015; Otterbach et al., 2017; Jiang et al., 2019).

[*]Equal contribution [1]Institute de Recherche en Informatique Fondamental - Paris, France [2]QC Ware Corp. - Palo Alto, California [3]Atos Quantum Lab - Les Clayes sous Bois, France. Correspondence to: Alessandro Luongo <aluongo@irif.fr>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

Here, we look at Expectation-Maximization (EM), a fundamental algorithm in unsupervised learning, that can be used to fit different mixture models and give maximum likelihood estimates for *latent variable models*. Such generative models are one of the most promising approaches to unsupervised problems. The goal of a generative model is to learn a probability distribution that is most likely to have generated the data collected in a dataset set $V \in \mathbb{R}^{n\times d}$ of n vectors of d features. Fitting a model consists in learning the parameters of a probability distribution p in a certain parameterized family that best describes our vectors v_i. This formulation allows one to reduce a statistical problem into an optimization problem. For a given machine learning model γ, under the assumption that each point is independent and identically distributed the log-likelihood of a dataset V is defined as $\ell(\gamma; V) := \sum_{i=1}^{n} \log p(v_i|\gamma)$, where $p(v_i|\gamma)$ is the probability that a point v_i comes from model γ. For ML estimates we want to find the model $\gamma^*_{ML} := \arg\max_\gamma \ell(\gamma; V)$. Due to the non-convex landscape of the function, optimizing ℓ using gradient based techniques often do not perform well. MAP estimates can be seen as the Bayesian version of maximum likelihood estimation problems, and are defined as $\gamma^*_{MAP} := \arg\max_\gamma \sum_{i=1}^{n} \log p(v_i|\gamma) + \log p(\gamma)$. MAP estimates are often preferred over ML estimates, due to a reduced propensity to overfit.

The EM algorithm has been proposed in different works by different authors but has been formalized as we know it in 1977 (Dempster et al., 1977). For more details, we refer to (Lindsay, 1995; Bilmes et al., 1998). EM is an iterative algorithm which is guaranteed to converge to a (local) optimum of the likelihood, and it is widely used to solve the ML or the MAP estimation problems. This algorithm has a striking number of applications and has been successfully used for medical imaging (Balafar et al., 2010), image restoration (Lagendijk et al., 1990), problems in computational biology (Fan et al., 2010), and so on.

One of the most important applications of the EM algorithm is for fitting mixture models in machine learning (Murphy, 2012). Most of the mixture models use a base distribution that belongs to the exponential family: Poisson (Church & Gale, 1995), Binomial, Multinomial, log-normal (Dexter & Tanner, 1972), exponential (Ghitany et al., 1994), Dirichlet multinomial (Yin & Wang, 2014), and others. EM is also used to fit mixtures of experts, mixtures of the student T dis-

tribution (which does not belong to the exponential family, and can be fitted with EM using (Liu & Rubin, 1995)), factor analysis, probit regression, and learning Hidden Markov Models (Murphy, 2012). The estimator of the parameter computed by EM has many relevant properties. For instance, it is asymptotically efficient: no other estimator can achieve asymptotically smaller variance in function of the number of points (Moitra, 2018).

In this work, we introduce Quantum Expectation-Maximization (QEM), a quantum algorithm for fitting mixture models. We detail its usage in the context of Gaussian mixture models, and we extend the result to other distributions in the exponential family. It is straightforward to use the ML estimates to compute the MAP estimate of a mixture model. Our main result can be stated as:

Result (Quantum Expectation-Maximization)**.** *(see Theorem 4.9) Given quantum access to a GMM and a dataset $V \in \mathbb{R}^{n\times d}$ like in Definition 2, for parameters $\delta_\theta, \delta_\mu > 0$, Quantum Expectation-Maximization (QEM) fits a maximum likelihood (or a Maximum A Posteriori) estimate of a Gaussian mixture model with k components, in running time per iteration which is dominated by:*

$$\widetilde{O}\left(\frac{d^2 k^{4.5}\eta^3\kappa^2(V)\kappa^2(\Sigma)\mu(\Sigma)}{\delta_\mu^3}\right), \tag{1}$$

where Σ is a covariance matrix of a Gaussian distribution of one of the mixture components, η is a parameter of the dataset related to the maximum norm of the vectors, $\delta_\theta, \delta_\mu$ are error parameters in the QEM algorithm, $\mu(\Sigma)$ (which is always $\leq \sqrt{d}$) is a factor appearing in quantum linear algebra and κ is the condition number of a matrix.

To run the quantum algorithm we assume to have access to data structures that allow quantum access to the data and the GMM, as in Definition 2. The preprocessing time needed to create these data structures in linear in the size of the data. It is typical for quantum algorithms to depend on a set of parameters, and they can achieve speedups with respect to classical algorithms when these parameters are within a certain range. Importantly, the value of these parameters can be estimated (as we do in this work for the VoxForge dataset) and we expect it not to be too high on real datasets. Most of these parameters can be upper bounded, as described in the Experiment section. Remark that we have formal bounds for the condition number as $\kappa(V) \approx 1/\min(\{\theta_1, \cdots, \theta_k\} \cup \{d_{st}(\mathcal{N}(\mu_i, \Sigma_i), \mathcal{N}(\mu_j, \Sigma_j)) | i \neq j \in [k]\})$, where d_{st} is the statistical distance between two Gaussian distributions. (Kalai et al., 2012). Here we only kept the term in the running time that dominates for the range of parameters of interest, while in Theorem 4.9 we state explicitly the running times of each step of the algorithm. As in the classical case, QEM algorithm runs for a certain number of iterations until a stopping condition is met (defined by a parameter $\epsilon_\tau > 0$) which implies convergence to a (local) optimum. We remark that in the above result we performed tomography enough times to get an ℓ_2 guarantee in the approximation, nevertheless, a lesser guarantee may be enough, for example using ℓ_∞ tomography (see Supplementary Material, Theorem 1.11), which can potentially remove the term d^2 from the running time.

We discuss a first high-level comparison of the quantum algorithm with the standard classical algorithms. The runtime of a single iteration in the standard implementation of the EM algorithm is $O(knd^2)$ (Pedregosa et al., 2011; Murphy, 2012). The advantage of the quantum algorithm is an exponential improvement with respect to the number of elements in the training set, albeit with a worsening on other parameters. Note that we expect the number of iterations of the quantum algorithm to be similar to the number of iteration of the classical case, as the convergence rate is not expected to change, and this belief is corroborated by experimental evidence for the simpler case of univariate Gaussians of k-means (Kerenidis et al., 2019). In this work, we tested our algorithm on a non-trivial dataset for the problem of speaker recognition on the VoxForge dataset (Voxforge.org). The experiment aimed to gauge the range of the parameters that affects the runtime, and test if the error introduced in the quantum procedures still allow the models to be useful. From the experiments reported in Section 5, we believe that datasets where the number of samples is very large might be processed faster on a quantum computer. One should expect that some of the parameters of the quantum algorithm can be improved, especially the dependence on the condition numbers and the errors, which can extend the number of datasets where QEM can offer a computational advantage. We also haven't optimized the parameters that govern the runtime in order to have bigger speedups, but we believe that hyperparameter tuning techniques, or a simple grid search, can improve the quantum algorithm even further. ML may not always be the best way to estimate the parameters of a model. For instance, in high-dimensional spaces, ML estimates often tend to overfit. MAP estimates inject into a ML model some external information, perhaps from domain experts. This usually avoids overfitting by having a kind of regularization effect on the model. For more information on how to use QEM for a MAP estimate we refer to (Murphy, 2012) and to the Supplementary Material.

This work is organized as such. First, we review previous efforts in quantum algorithms for unsupervised classification and classical algorithms for GMM. Then we present the classical EM algorithm for GMM. Then, in Section 4 we describe QEM and its theoretical analysis, and we show how to use it to fit a GMM and other mixture models. We conclude by giving some experimental evidence on the expected runtime on real data. In the Supplementary Material the interested reader can find all the proofs of the Lemmas

that we used to state our main result, details on the experiments, initialization strategies of QEM, the rules for the MAP update of a GMM, and the description of the possible covariance matrices for GMM models.

1.1. Related work

Many classical algorithms for GMM exist. Already in 1895, Karl Pearson fitted manually a GMM of 2 features using the methods of moments (Pearson, 1895). Without resorting to heuristics - like the EM algorithm - other procedures with provable guarantees exist. For example, in (Dasgupta, 1999) they assume only one shared covariance matrix among mixtures, but they have a polynomial dependence on the number of elements in the training set, and in (Kannan et al., 2005) they developed the spectral projection technique. Formal learnability of Gaussian Mixtures has been studied (Kalai et al., 2012; Moitra & Valiant, 2010). While these important results provide provable algorithms for the case when the Gaussians are well-separated, we think that the heuristic-based approaches, like the classical and the quantum EM, should not be directly compared to this class of algorithms.

In the quantum setting, a number of algorithms have been proposed in the context of unsupervised learning (Aïmeur et al., 2013; Lloyd et al., 2013; Otterbach et al., 2017; Kerenidis et al., 2019). Recently, classical machine learning algorithms were obtained by "dequantizing" quantum machine learning algorithms (Tang, 2018a; Gilyén et al., 2018a; Tang, 2018b; Gilyén et al., 2018b; Chia et al., 2018). This class of algorithms is of high theoretical interest, as runtime is poly-logarithmic in the dimensions of the dataset. However, the high polynomial dependence on the Frobenius norm, the error, and the condition number, makes this class of algorithms still impractical for interesting datasets, as shown experimentally (Arrazola et al., 2019). We believe there can be a "dequantized" version of QEM, but it seems rather unlikely that this algorithm will be more efficient than QEM or the classical EM algorithm.

As the classical EM for GMM can be seen as a generalization of k-means, our work is a generalization of the q-means algorithm in (Kerenidis et al., 2019). Independently and simultaneously, Miyahara, Aihara, and Lechner also extended the q-means algorithm and applied it to fit a Gaussian mixture models (Miyahara et al., 2019). The main difference with this work is that the update step in (Miyahara et al., 2019) is performed using a hard-clustering approach (as in the k-means algorithm): for updating the centroids and the covariance matrices of a cluster j, only the data points for which cluster j is *nearest* are taken into account. In our work, as in the classical EM algorithm, we use the soft clustering approach: that is, for updating the centroid and the covariance matrices of cluster j, all the data points *weighted by their responsibility* (Defined in Eq. 2) for cluster j are taken into account. Both approaches have merits and can offer advantages (Kearns et al., 1998), albeit is more adherent to the original EM algorithm.

2. EM and Gaussian mixture models

GMM are probably the most used mixture model used to solve unsupervised classification problems. In unsupervised settings, we are given a training set of unlabeled vectors $v_1 \dots v_n \in \mathbb{R}^d$ which we represent as rows of a matrix $V \in \mathbb{R}^{n\times d}$. Let $y_i \in [k]$ one of the k possible labels for a point v_i. We posit that the joint probability distribution of the data $p(v_i, y_i) = p(v_i|y_i)p(y_i)$, is defined as follow: $y_i \sim$ Multinomial(θ) for $\theta \in \mathbb{R}^k$, and $p(v_i|y_i = j) \sim \mathcal{N}(\mu_j, \Sigma_j)$. The θ_j such that $\sum_j \theta_j = 1$ are called *mixing weights*, i.e. the probabilities that $y_i = j$, and $\mathcal{N}(\mu_j, \Sigma_j)$ is the Gaussian distribution centered in $\mu_j \in \mathbb{R}^d$ with covariance matrix $\Sigma_j \in \mathbb{R}^{d\times d}$ (Ng, 2012). We use the letter ϕ to represent our *base distribution*, which in this case is the probability density function of a multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$. Using this formulation, a GMM is expressed as: $p(v) = \sum_{j=1}^k \theta_j \phi(v; \mu_j, \Sigma_j)$. Fitting a GMM to a dataset reduces to finding an assignment for the parameters of the model $\gamma = (\theta, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = (\theta, \mu_1, \cdots, \mu_k, \Sigma_1, \cdots, \Sigma_k)$ that best maximize the log-likelihood for a given dataset. Note that the algorithm used to fit GMM can return a local minimum which might be different than γ^*: the model that represents the global optimum of the likelihood function. For a given γ, the probability for an observation v_i to be assigned to the component j is given by:

$$r_{ij} = \frac{\theta_j \phi(v_i; \mu_j, \Sigma_j)}{\sum_{l=1}^k \theta_l \phi(v_i; \mu_l, \Sigma_l).} \tag{2}$$

This value is called *responsibility*, and corresponds to the posterior probability of the sample i being assigned label j by the current model. As anticipated, to find the best parameters of our generative model, we maximize the log-likelihood of the data. Alas, it is seldom possible to solve maximum likelihood estimation analytically (i.e. by finding the zeroes of the derivatives of the log-like function) for mixture models like the GMM. To complicate things, the likelihood function for GMM is not convex, and thus we might find some local minima (Hastie et al., 2009).

EM is an iterative algorithm that solves numerically the optimization problems linked to ML and MAP estimations. The classical EM algorithm works as follows: in the expectation step all the responsibilities are calculated, and we estimate the missing variables y_i given the current guess of the parameters $(\theta, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ of the model. Then, in the maximization step, we use the estimate of the latent variables obtained in the expectation step to update the estimate of the parameters: $\gamma^{t+1} = (\theta^{t+1}, \boldsymbol{\mu}^{t+1}, \boldsymbol{\Sigma}^{t+1})$. While in the expectation step we calculate a lower bound on the likelihood, in the

maximization step we maximize it. Since at each iteration the likelihood can only increase, the algorithm is guaranteed to converge, albeit possibly to a local optimum (see (Hastie et al., 2009) for the proof). The stopping criterion is usually a threshold on the increment of the log-likelihood: if it changes less than a ϵ_τ between two iterations, then the algorithm stops. As the value of the log-likelihood depends on the amount of data points in the training sets, it is often preferable to adopt a scale-free stopping criterion, which does not depend on the number of samples. For instance, in scikit-learn (Pedregosa et al., 2011) the stopping criterion is given by a tolerance on the average increment of the log-probability, chosen to be 10^{-3}. More precisely, the stopping criterion in the quantum and classical algorithm is $|\mathbb{E}[\log p(v_i;\gamma^t)] - \mathbb{E}[\log p(v_i;\gamma^{t-1})]| < \epsilon_\tau$.

Dataset assumptions in GMM In the remainig of the paper we make an assumption on the dataset, namely that all elements of the mixture contribute proportionally to the total responsibility:

$$\frac{\sum_{i=1}^{n} r_{ij}}{\sum_{i=1}^{n} r_{il}} = \Theta(1) \quad \forall j,l \in [k] \tag{3}$$

This is equivalent to assuming that $\theta_j/\theta_l = \Theta(1) \quad \forall j,l \in [k]$. The algortihm we propose can of course be used even in cases where this assumption does not hold. In this case, the running time will include a factor as in Eq. 3 which for simplicity we have taken as constant in what follows. Note that classical algorithms would also find it difficult to fit the data in certain cases, for example when some of the clusters are very small. In fact, it is known (and not surprising) that if the statistical distance between the probability density function of two different Gaussian distributions is smaller than $1/2$, then we can not tell for a point v from which Gaussian distribution it belongs to, even if we knew the parameters (Moitra, 2018). Only for convenience in the analysis, we also assume the dataset as being normalized such that the shortest vector has norm 1 and define $\eta := max_i \|v_i\|^2$ to be the maximum norm squared of a vector in the dataset.

3. Preliminaries

Quantum computing provides a new way to encode information and perform algorithms. The basic carrier of quantum information is the *qubit*, which can be in a superposition of two states $|0\rangle$ and $|1\rangle$ at the same time. More formally, a quantum bit can be written as: $|x\rangle = \alpha|0\rangle + \beta|1\rangle$ and it corresponds to a unit vector in the Hilbert space $\mathcal{H}_2 = \text{span}\{|0\rangle, |1\rangle\}$ with $\alpha,\beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. An n-qubit state corresponds to a unit vector in $\mathcal{H}_n = \otimes_{i\in[n]}\mathcal{H}_2 \sim \mathbb{C}^{2^n}$, there $\otimes$ is the tensor product. Denoting by $\{|i\rangle\}$ the standard basis of $\mathcal{H}_n$, an n-qubit state can be written as: $|x\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle$ with $\alpha_i \in \mathbb{C}$ and $\sum_i |\alpha_i|^2 = 1$. Quantum states evolve through unitary matrices, which are norm-preserving, and thus can be used as suitable mathematical description of pure quantum evolutions $U|\psi\rangle \mapsto |\psi'\rangle$. A quantum state $|x\rangle$ can be measured, and the probability that a measurement on $|x\rangle$ gives outcome i is $|\alpha_i|^2$. The quantum state corresponding to a vector $v \in \mathbb{R}^m$ is defined as $|v\rangle = \frac{1}{\|v\|}\sum_{j\in[m]} v_j |j\rangle$. Note that to build $|v\rangle$ we need $\lceil \log m \rceil$ qubits. In the following, when we say with high probability we mean a value which is inverse polynomially close to 1. The value of $\mu(V)$ which often compares in the runtimes comes from the procedure we use in the proof. While its value for real dataset is discussed in the manuscript, for a theoretical analysis we refer to the Supplementary Material and (Kerenidis & Prakash, 2020).

The cluster centers, called centroids, at time t are stored in the matrix $C^t \in \mathbb{R}^{k\times d}$, such that the j^{th} row c_j^t for $j \in [k]$ represents the centroid of the cluster $\mathcal{C}_j^t$. The number of non-zero elements of the rows of V is $nnz(V)$. Let $\kappa(V)$ be the condition number of V: the ratio between the biggest and the smallest (non-zero) singular value. All the tools used in this work, like quantum algorithms for computing distances, linear algebraic operations, and further introduction to quantum notation is reported in the Supplementary Material. We recommend Nielsen and Chuang (Nielsen & Chuang, 2002) for an introduction to the subject.

4. Quantum EM for GMM

In this section, we present a quantum EM algorithm to fit a GMM. The algorithm can also be adapted fit other mixtures models where the probability distributions belong to the exponential family. As the GMM is both intuitive and one of the most widely used mixture models, our results are presented for the GMM case. As in the classical algorithm, we use some subroutines to compute the responsibilities and update our current guess of the parameters which resemble the E and the M step of the classical algorithm. While the classical algorithm has clearly two separate steps for Expectation and Maximization, the quantum algorithm uses a subroutine to compute the responsibilities inside the step that performs the maximization. During the quantum M step, the algorithm updates the model by creating quantum states corresponding to parameters γ^{t+1} and then recovering classical estimates for these parameters using quantum tomography or amplitude amplification. In order for the subroutines to be efficient, we build quantum access (as in Definition 2) to the current estimate of the model, and we update it at each maximization step.

An approximate version of GMM Here we define an approximate version of GMM, that we fit with QEM algorithm. The difference between this formalization and the original GMM is simple. Here we make explicit in the model the *ap-*

proximation error introduced during the training algorithm.

Definition 1 (Approximate GMM)**.** *Let* $\gamma^t = (\theta^t, \boldsymbol{\mu}^t, \boldsymbol{\Sigma}^t) = (\theta^t, \mu_1^t \cdots \mu_k^t, \Sigma_1^t \cdots \Sigma_k^t)$ *a model fitted by the standard EM algorithm from* γ^0 *an initial guess of the parameters, i.e.* γ^t *is the error-free model that standard EM would have returned after* t *iterations. Starting from the same choice of initial parameters* γ^0*, fitting a GMM with the QEM algorithm with* $\Delta = (\delta_\theta, \delta_\mu)$ *means returning a model* $\overline{\gamma}^t = (\overline{\theta}^t, \overline{\boldsymbol{\mu}}^t, \overline{\boldsymbol{\Sigma}}^t)$ *such that:*

- $\left\|\overline{\theta}^t - \theta^t\right\| < \delta_\theta$,
- $\left\|\overline{\mu_j}^t - \mu_j^t\right\| < \delta_\mu$ *for all* $j \in [k]$,
- $\left\|\overline{\Sigma_j}^t - \Sigma_j^t\right\| \leq \delta_\mu \sqrt{\eta}$ *for all* $j \in [k]$.

Quantum access to the mixture model Here we explain how to load into a quantum computer a GMM and a dataset represented by a matrix V. This is needed for a quantum computer to be able to work with a machine learning model. The definition of quantum access to other kind of models is analogous.

Definition 2 (Quantum access to a GMM)**.** *We say that we have quantum access to a GMM of a dataset* $V \in \mathbb{R}^{n \times d}$ *and model parameters* $\theta_j \in \mathbb{R}$, $\mu_j \in \mathbb{R}^d, \Sigma_j \in \mathbb{R}^{d \times d}$ *for all* $j \in [k]$ *if we can perform in time* $O(polylog(d))$ *the following mappings:*

- $|j\rangle |0\rangle \mapsto |j\rangle |\mu_j\rangle$,
- $|j\rangle |i\rangle |0\rangle \mapsto |j\rangle |i\rangle |\sigma_i^j\rangle$ *for* $i \in [d]$ *where* σ_i^j *is the* i*-th rows of* $\Sigma_j \in \mathbb{R}^{d \times d}$,
- $|i\rangle |0\rangle \mapsto |i\rangle |v_i\rangle$ *for all* $i \in [n]$,
- $|i\rangle |0\rangle |0\rangle \mapsto |i\rangle |vec[v_i v_i^T]\rangle = |i\rangle |v_i\rangle |v_i\rangle$ *for* $i \in [n]$,
- $|j\rangle |0\rangle \mapsto |j\rangle |\theta_j\rangle$.

For instance, one may use a QRAM data structure as in (Kerenidis & Prakash, 2017; 2020) so that quantum access to a matrix A can be reduced to being able to perform the unitary mapping $|i\rangle |j\rangle |0\rangle \to |i\rangle |j\rangle |a_{ij}\rangle$, in other words being able to map the indices (i, j) to the element of the matrix a_{ij}, which is the usual quantum oracle model.

4.1. QEM algorithm and analysis

We describe the different steps of QEM as Algorithm 1, and analyze its running time, by showing the runtime and the error for each of the steps of the algorithm. Quantum initialization strategies exists, and are described in the Supp. Material. In the statement of the following Lemmas, Σ is defined as one of the covariance matrices of the mixture. It appears in the runtime of the algorithms because is introduced during the proofs, which can be found in the Supp. Material.

Algorithm 1 QEM for GMM

Require: Quantum access to a GMM model, precision parameters $\delta_\theta, \delta_\mu$, and threshold ϵ_τ.

Ensure: A GMM $\overline{\gamma}^t$ that maximizes locally the likelihood $\ell(\gamma; V)$, up to tolerance ϵ_τ.

1: Use a heuristic (Supp. Material) to determine the initial guess $\gamma^0 = (\theta^0, \boldsymbol{\mu}^0, \boldsymbol{\Sigma}^0)$, and build quantum access as in Definition 2 those parameters.
2: Use Lemma 4.1 to estimate the log determinant of the matrices $\{\Sigma_j^0\}_{j=1}^k$.
3: t=0
4: **repeat**
5: **Step 1:** Get an estimate of θ^{t+1} using Lemma 4.4 such that $\left\|\overline{\theta}^{t+1} - \theta^{t+1}\right\| \leq \delta_\theta$.
6: **Step 2:** Get an estimate $\{\overline{\mu_j}^{t+1}\}_{j=1}^k$ by using Lemma 4.6 to estimate each $\left\|\mu_j^{t+1}\right\|$ and $|\mu_j^{t+1}\rangle$ such that $\left\|\mu_j^{t+1} - \overline{\mu_j}^{t+1}\right\| \leq \delta_\mu$.
7: **Step 3:** Get an estimate $\{\overline{\Sigma_j}^{t+1}\}_{j=1}^k$ by using Lemma 4.7 to estimate $\left\|\Sigma_j^{t+1}\right\|_F$ and $|\Sigma_j^{t+1}\rangle$ such that $\left\|\Sigma_j^{t+1} - \overline{\Sigma_j}^{t+1}\right\| \leq \delta_\mu \sqrt{\eta}$.
8: **Step 4:** Estimate $\mathbb{E}[\overline{p(v_i; \gamma^{t+1})}]$ up to error $\epsilon_\tau/2$ using Theorem 4.8.
9: **Step 5:** Build quantum access to γ^{t+1}, and use Lemma 4.1 to estimate the determinants $\{\log det(\Sigma_j^{t+1})\}_{j=0}^k$.
10: $t = t + 1$
11: **until**

$$|\mathbb{E}[\overline{p(v_i; \gamma^t)}] - \mathbb{E}[\overline{p(v_i; \gamma^{t-1})}]| < \epsilon_\tau$$

12: Return $\overline{\gamma}^t = (\overline{\theta}^t, \overline{\boldsymbol{\mu}}^t, \overline{\boldsymbol{\Sigma}}^t)$

4.1.1. Expectation

In this step of the algorithm we are just showing how to compute efficiently the responsibilities as a quantum state. First, we compute the responsibilities in a quantum register, and then we show how to put them as amplitudes of a quantum state. We start by a classical algorithm used to efficiently approximate the log-determinant of the covariance matrices of the data. At each iteration of QEM we need to approximate the log-determinant of the updated covariance matrices using Lemma 4.1. We will see from the error analysis that in order to get an estimate of the GMM, we need to call Lemma 4.1 with precision for which the runtime of Lemma 4.1 gets subsumed by the running time

of finding the updated covariance matrices through Lemma 4.7. Thus, we do not explicitly write the time to compute the log-determinant from now on in the running time of the algorithm and when we say that we update Σ we include an update on the estimate of $\log(\det(\Sigma))$ as well. The method can be extended to fit other kind of mixture models with base distribution in exponential family (Definition 1 Supplementary Material), by replacing the function to compute the posterior probabilities $p(v_i|j)$ (Lemma 4.3) by a different procedure for computing the responsibility. The technique is given in Lemma 1.3 and 1.4 of the Supplementary Material, it can be used to bound the error of Lemma 4.3 for distributions in the exponential family using the smoothness properties of the softmax function. The log-determinant in the Gaussian case generalizes to the cumulant generating function $A(\nu)$ for the exponential families.

Lemma 4.1 (Determinant approximation)**.** *There is an algorithm that, given a matrix Σ and a parameter $0 < \delta < 1$, outputs an estimate $\overline{\log(det(\Sigma))}$ such that $|\overline{\log(det(\Sigma))} - \log(det(\Sigma))| \leq \epsilon$ with probability $1-\delta$ in time:*

$$T_{Det,\epsilon} = \widetilde{O}\left(\epsilon^{-2}\kappa(\Sigma)\log(1/\delta)nnz(\Sigma)|\log(\det(\Sigma))|\right)$$

This Lemma follows from the previous results of (Boutsidis et al., 2017). Now we can state the Lemma used to compute the responsability: a quantum algorithm for evaluating the exponent of a Gaussian distribution.

Lemma 4.2 (Quantum Gaussian Evaluation)**.** *Suppose we have stored in the QRAM a matrix $V \in \mathbb{R}^{n\times d}$, the centroid $\mu \in \mathbb{R}^d$ and the covariance matrix $\Sigma \in \mathbb{R}^{d\times d}$ of a multivariate Gaussian distribution $\phi(v|\mu,\Sigma)$, as well as an estimate for $\log(\det(\Sigma))$. Then for $\epsilon_1 > 0$, there exists a quantum algorithm that with probability $1-\gamma$ performs the mapping,*

- *$U_{G,\epsilon_1} : |i\rangle|0\rangle \to |i\rangle|\overline{s_i}\rangle$ such that $|s_i - \overline{s_i}| < \epsilon_1$, where $s_i = -\frac{1}{2}((v_i-\mu)^T\Sigma^{-1}(v_i-\mu) + d\log 2\pi + \log(det(\Sigma)))$ is the exponent for the Gaussian probability density function.*

The running time is $T_{G,\epsilon_1} = O\left(\frac{\kappa^2(\Sigma)\mu(\Sigma)\log(1/\gamma)}{\epsilon_1}\eta\right)$.

Using controlled operations it is simple to extend the previous Theorem to work with multiple Gaussians distributions (μ_j, Σ_j). That is, we can control on a register $|j\rangle$ to do $|j\rangle|i\rangle|0\rangle \mapsto |j\rangle|i\rangle|\phi(v_i|\mu_j,\Sigma_j)\rangle$. In the next Lemma we will see how to obtain the responsibilities r_{ij} using the previous Theorem and standard quantum circuits for doing arithmetic, controlled rotations, and amplitude amplification.

Lemma 4.3 (Calculating responsibilities)**.** *Suppose we have quantum access to a GMM with parameters $\gamma^t = (\theta^t, \boldsymbol{\mu}^t, \boldsymbol{\Sigma}^t)$. There are quantum algorithms that can:*

1. *Perform the mapping $|i\rangle|j\rangle|0\rangle \mapsto |i\rangle|j\rangle|\overline{r_{ij}}\rangle$ such that $|\overline{r_{ij}} - r_{ij}| \leq \epsilon_1$ with high probability in time:*

$$T_{R_1,\epsilon_1} = \widetilde{O}(k^{1.5} \times T_{G,\epsilon_1})$$

2. *For a given $j \in [k]$, construct state $|\overline{R_j}\rangle$ such that $\left\||\overline{R_j}\rangle - \frac{1}{\sqrt{Z_j}}\sum_{i=0}^{n} r_{ij}|i\rangle\right\| < \epsilon_1$ where $Z_j = \sum_{i=0}^{n} r_{ij}^2$ with high probability in time:*

$$T_{R_2,\epsilon_1} = \widetilde{O}(k^2 \times T_{R_1,\epsilon_1})$$

4.1.2. MAXIMIZATION

Now we need to get an updated estimate for the parameters of our model. At each iteration of QEM we build a quantum state proportional to the updated parameters of the model, and then recover them. Once the new model has been obtained, we update the QRAM such that we get quantum access to the model γ^{t+1}. The possibility to estimate θ comes from a call to the unitary we built to compute the responsibilities, and amplitude amplification.

Lemma 4.4 (Computing θ^{t+1})**.** *We assume quantum access to a GMM with parameters γ^t and let $\delta_\theta > 0$ be a precision parameter. There exists an algorithm that estimates $\overline{\theta}^{t+1} \in \mathbb{R}^k$ such that $\left\|\overline{\theta}^{t+1} - \theta^{t+1}\right\| \leq \delta_\theta$ in time $T_\theta = O\left(k^{3.5}\eta^{1.5}\frac{\kappa^2(\Sigma)\mu(\Sigma)}{\delta_\theta^2}\right)$.*

We use quantum linear algebra to transform the uniform superposition of responsibilities of the j-th mixture into the new centroid of the j-th Gaussian. Let $R_j^t \in \mathbb{R}^n$ be the vector of responsibilities for a Gaussian j at iteration t. The following claim relates the vectors R_j^t to the centroids μ_j^{t+1} and its proof is straightforward.

Claim 4.5. *Let $R_j^t \in \mathbb{R}^n$ be the vector of responsibilities of the points for the Gaussian j at time t, i.e. $(R_j^t)_i = r_{ij}^t$. Then $\mu_j^{t+1} \leftarrow \frac{\sum_{i=1}^n r_{ij}^t v_i}{\sum_{i=1}^n r_{ij}^t} = \frac{V^T R_j^t}{n\theta_j}$.*

From this Claim, we derive the procedure to estimate the new centroids μ_j^{t+1}: we use Lemma 4.3 along with quantum access to the matrix V.

Lemma 4.6 (Computing $\boldsymbol{\mu}_j^{t+1}$)**.** *We assume we have quantum access to a GMM with parameters γ^t. For a precision parameter $\delta_\mu > 0$, there is a quantum algorithm that calculates $\{\overline{\mu_j}^{t+1}\}_{j=1}^k$ such that for all $j \in [k]$ $\left\|\overline{\mu_j}^{t+1} - \mu_j^{t+1}\right\| \leq \delta_\mu$ in time $T_\mu = \widetilde{O}\left(\frac{kd\eta\kappa(V)(\mu(V)+k^{3.5}\eta^{1.5}\kappa^2(\Sigma)\mu(\Sigma))}{\delta_\mu^3}\right)$*

From the ability to calculate responsibility and indexing the centroids, we derive the ability to reconstruct the covariance matrix of the Gaussians as well. Again, we use quantum linear algebra subroutines and tomography to recover an

approximation of each Σ_j. Recall that we have defined the matrix $V' \in \mathbb{R}^{n \times d^2}$ where the i-th row of V' is defined as $\text{vec}[v_i v_i^T]$. Note that the quantum states corresponding to the rows of V' can be prepared as $|i\rangle|0\rangle|0\rangle \to |i\rangle|v_i\rangle|v_i\rangle$, using twice the procedure for creating the rows of V.

Lemma 4.7 (Computing Σ_j^{t+1})**.** *Given quantum access to a GMM with parameters γ^t. We also have computed estimates $\overline{\mu_j}^{t+1}$ of all centroids such that $\left\|\overline{\mu_j}^{t+1} - \mu_j^{t+1}\right\| \leq \delta_\mu$ for precision parameter $\delta_\mu > 0$. Then, there exists a quantum algorithm that outputs estimates for the new covariance matrices $\{\overline{\Sigma}_j^{t+1}\}_{j=1}^k$ such that $\left\|\Sigma_j^{t+1} - \overline{\Sigma}_j^{t+1}\right\|_F \leq \delta_\mu\sqrt{\eta}$ with high probability, in time,*

$$T_\Sigma := \widetilde{O}\Big(\frac{kd^2\eta\kappa^2(V)(\mu(V') + \eta^2 k^{3.5}\kappa^2(\Sigma)\mu(\Sigma))}{\delta_\mu^3}\Big)$$

4.1.3. Quantum estimation of log-likelihood

Now we are going to state how to get an estimate of the log-likelihood using a quantum procedure and access to a GMM model. Because of the error analysis, in the quantum algorithm is more conveniente to estimate $\mathbb{E}[p(v_i;\gamma^t)] = \frac{1}{n}\sum_{i=1}^n p(v_i;\gamma)$. From this we can estimate an upper bound on the log-likelihood as $n\log\mathbb{E}[p(v_i)] = \sum_{i=1}^n \log\mathbb{E}[p(v_i)] \geq \sum_{i=1}^n \log p(v_i) = \ell(\gamma; V)$.

Lemma 4.8 (Quantum estimation of likelihood)**.** *We assume we have quantum access to a GMM with parameters γ^t. For $\epsilon_\tau > 0$, there exists a quantum algorithm that estimates $\mathbb{E}[p(v_i;\gamma^t)]$ with absolute error ϵ_τ in time*

$$T_\ell = \widetilde{O}\left(k^{1.5}\eta^{1.5}\frac{\kappa^2(\Sigma)\mu(\Sigma)}{\epsilon_\tau^2}\right)$$

4.1.4. QEM for GMM

Putting together all the previous Lemmas, we write the main result of the work.

Theorem 4.9 (QEM for GMM)**.** *We assume we have quantum access to a GMM with parameters γ^t. For parameters $\delta_\theta, \delta_\mu, \epsilon_\tau > 0$, the running time of one iteration of the Quantum Expectation-Maximization (QEM) algorithm is*

$$O(T_\theta + T_\mu + T_\Sigma + T_\ell),$$

for $T_\theta = \widetilde{O}\left(k^{3.5}\eta^{1.5}\frac{\kappa^2(\Sigma)\mu(\Sigma)}{\delta_\theta^2}\right)$,
$T_\mu = \widetilde{O}\left(\frac{kd\eta\kappa(V)(\mu(V)+k^{3.5}\eta^{1.5}\kappa^2(\Sigma)\mu(\Sigma))}{\delta_\mu^3}\right)$,
$T_\Sigma = \widetilde{O}\Big(\frac{kd^2\eta\kappa^2(V)(\mu(V')+\eta^2k^{3.5}\kappa^2(\Sigma)\mu(\Sigma))}{\delta_\mu^3}\Big)$ *and*
$T_\ell = \widetilde{O}\left(k^{1.5}\eta^{1.5}\frac{\kappa^2(\Sigma)\mu(\Sigma)}{\epsilon_\tau^2}\right)$.

For parameter that are expected to be predominant in the runtime, d and $\kappa(V)$, the dominant term is T_Σ.

The proof follows directly from the previous lemmas. Note that the cost of the whole algorithm is given by repeating the expectation and the maximization steps several times, until the threshold on the log-likelihood is reached. The expression of the runtime can be simplified by observing that the cost of performing tomography on the covariance matrices Σ_j dominates the runtime.

5. Experimental Results

In this section, we present the results of some experiments on a real dataset, on which we estimated the runtime of the quantum algorithm. We also bound the value of the parameters that governs the runtime, like $\kappa(\Sigma)$, $\kappa(V)$, $\mu(\Sigma)$, $\mu(V)$, δ_θ, and δ_μ, and we give heuristic for dealing with the condition number. We can put a threshold on the condition number of the matrices Σ_j, by discarding singular values which are smaller than a certain threshold. This might decrease the runtime of the algorithm without impacting its performances. This is indeed done often in classical machine learning models, since discarding the eigenvalues smaller than a certain threshold might even improve upon the metric under consideration (i.e. often the accuracy), by acting as a form of regularization (Murphy, 2012, Section 6.5). This is equivalent to limiting the eccentricity of the Gaussians. We can do similar considerations for putting a threshold on the condition number of the dataset $\kappa(V)$. Recall that the value of the condition number of the matrix V is approximately $1/\min(\{\theta_1, \cdots, \theta_k\} \cup \{d_{st}(\mathcal{N}(\mu_i, \Sigma_i), \mathcal{N}(\mu_j, \Sigma_j)) | i \neq j \in [k]\})$, where d_{st} is the statistical distance between two Gaussian distributions (Kalai et al., 2012) and θ_i are the mixing weights. We have some choice in picking the definition for μ: in previous experiments it has been found that choosing the maximum ℓ_1 norm of the rows of V lead to values of $\mu(V)$ around 10 for the MNIST dataset (Kerenidis & Luongo, 2020; Kerenidis et al., 2019). Because of the way μ is defined, its value will not increase significantly as we add vectors to the training set. In case the matrix V can be clustered with high-enough accuracy by distance-based algorithms like k-means, it has been showed that the Frobenius norm of the matrix is proportional to $\sqrt{k}$, that is, the rank of the matrix depends on the number of different classes contained in the data. Given that EM is just a more powerful extension of k-means, we can rely on similar observations too. Usually, the number of features d is much more than the number of components in the mixture, i.e. $d \gg k$, so we expect d^2 to dominate the other parameters, thus making T_Σ the leading term in the runtime. We expect this cost to be be mitigated by using ℓ_∞ form of tomography (Theorem 1.11 in Supplementary Material) but we defer further experiment for future research.

As we said, the quantum running time saves the factor that depends on the number of samples and introduces a number

of other parameters. Using our experimental results we can see that when the number of samples is large enough one can expect the quantum running time to be faster than the classical one. One may also save some more factors from the quantum running time with a more careful analysis.

To estimate the runtime of the algorithm, we need to gauge the value of the parameters δ_μ and δ_θ, such that they are small enough so that the likelihood is perturbed less than ϵ_τ, but large enough to have a fast algorithm. We have reasons to believe that on well-clusterable data, the value of these parameters will be large enough, such as not to impact dramatically the runtime. For instance, a quantum version of k-means algorithm has already been simulated on the MNIST dataset under similar assumptions (Kerenidis et al., 2019). The experiment concluded that, for datasets that are expected to be clustered nicely by this kind of clustering algorithms, the value of the parameters δ_μ did not decrease by increasing the number of samples nor the number of features. There, the value of δ_μ (which in their case was called just δ) has been kept between 0.2 and 0.5, while retaining a classification accuracy comparable to the classical k-means algorithm. We expect similar behavior in the GMM case, namely that for large datasets the impact on the runtime of the errors $(\delta_\mu, \delta_\theta)$ does not cancel out the exponential gain in the dependence on the number of samples, and we discuss more about this in the next paragraph. The value of the threshold of the likelihood ϵ_τ is usually (for instance in scikit-learn (Pedregosa et al., 2011)) chosen to be 10^{-3}. We will see that the value of η has always been 10 on average, with a maximum of 105 (as an outlier) in the experiments.

	MAP		ML	
	avg	max	avg	max
$\|\Sigma\|_2$	0.22	2.45	1.31	3.44
$\|\log\det(\Sigma)\|$	58.56	70.08	14.56	92.3
$\kappa^*(\Sigma)$	4.21	50	15.57	50
$\mu(\Sigma)$	3.82	4.35	2.54	3.67
$\mu(V)$	2.14	2.79	2.14	2.79
$\kappa(V)$	23.82	40.38	23.82	40.38

Table 1. We estimate some of the parameters of the VoxForge (Voxforge.org) dataset. Each model is the result of the best of 3 different initializations of the EM algorithm. The first and the second rows are the maximum singular values of all the covariance matrices, and the absolute value of the log-determinant. The row $\kappa^*(\Sigma)$ shows the condition number after the smallest singular values have been discarded. We report the value of $\mu(\Sigma)$ and $\mu(V)$ where the choice of μ is the maximum ℓ_1 norms of the rows of the matrices.

Experiments We analyzed a dataset which can be fitted well with the EM algorithm (Reynolds et al., 2000; Kumar; Voxforge.org). Specifically, we used QEM to do speaker recognition: the task of recognizing a speaker from a voice sample, having access to a training set of recorded voices of all the possible speakers. The training set consist in 5 speech utterances for 38 speakers (i.e. clips of a few seconds of voice speech). For each speaker, we extract the mel-frequency cepstral coefficients (MFCC) of the utterances (Reynolds et al., 2000), resulting in circa 5000 vectors of 40 dimensions. This represent the training set for each speaker. A speaker is then modeled with a mixture of 16 different diagonal Gaussians. The test set consists of other 5 or more unseen utterances for each of the same speakers. To label an utterance with a speaker, we compute the log-likelihood of the utterance for each trained model. The label consist in the speaker with highest log-likelihood. The experiment has been carried in form of classical simulation on a laptop computer. We repeated the experiment using a perturbed model, where we added some noise to the GMM at each iteration of the training, as in Definition 1. Then we measured the accuracy of the speaker recognition task. At last, we measured condition number, the absolute value of the log-determinant, and the value of $\mu(V)$ and $\mu(\Sigma)$. In this way we can test the stability and accuracy of the approximate GMM model introduced in Section 4, under the effect of noise. For values of $\delta_\theta = 0.038$, $\delta_\mu = 0.5$, we correctly classified 98.7% utterances. The baseline for ML estimate of the GMM is of 97.1%. We attribute the improved accuracy to the regularizing effect of the threshold and the noise, as the standard ML estimate is likely to overfit the data. Details of the experiment are reported in the Supplementary Material. We report the results of the measurement in Table 1.

6. Conclusions

Given the tremendous relevance of classical Expectation-Maximization algorithm, it is important to consider quantum versions of EM. Here we proposed a quantum Expectation-Maximization algorithm, and showed how to use it to fit a GMM in the ML estimation setting. We analyzed theoretically the runtime of QEM, and estimate it on real-world datasets, so to better understand cases where quantum computers can offer a computational advantage. While we discussed how to use QEM to fit other mixture models, its usage in the MAP settings is detailed in the Supp. Material.

The experiments suggest that the influence of the extra parameters in the quantum running time is moderate. This makes us believe that, when quantum computers will be available, our algorithm could be useful in analyzing large datasets. We believe further improvements of the algorithm can reduce even more the complexity with respect to these extra parameters. For instance, the ℓ_∞ tomography can potentially remove the dependence on d from the runtime. As we discussed above, we believe dequantization techniques can successfully be applied to this work as well, but we

don't expect the time complexity of the final algorithm to be competitive with the classical EM algorithm or with this result. We leave for future work the study of quantum algorithms for fitting GMM robust to adversarial attacks (Xu & Mareček, 2018; Pensia et al., 2019; Dasgupta, 1999), and quantum algorithms for computing the log-determinant of symmetric positive definite matrices, and further experiments.

7. Acknowledgement

This research was supported by QuantAlgo, Quantex, QuData, and ANR.

References

Aïmeur, E., Brassard, G., and Gambs, S. Quantum speed-up for unsupervised learning. *Machine Learning*, 90(2): 261–287, 2013.

Arrazola, J. M., Delgado, A., Bardhan, B. R., and Lloyd, S. Quantum-inspired algorithms in practice. *arXiv preprint arXiv:1905.10415*, 2019.

Balafar, M. A., Ramli, A. R., Saripan, M. I., and Mashohor, S. Review of brain mri image segmentation methods. *Artificial Intelligence Review*, 33(3):261–274, 2010.

Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., and Lloyd, S. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

Bilmes, J. A. et al. A gentle tutorial of the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. , 1998.

Boutsidis, C., Drineas, P., Kambadur, P., Kontopoulou, E.-M., and Zouzias, A. A randomized algorithm for approximating the log determinant of a symmetric positive definite matrix. *Linear Algebra and its Applications*, 533: 95–117, 2017.

Chakrabarti, S., Yiming, H., Li, T., Feizi, S., and Wu, X. Quantum wasserstein generative adversarial networks. In *Advances in Neural Information Processing Systems*, pp. 6778–6789, 2019.

Chia, N.-H., Lin, H.-H., and Wang, C. Quantum-inspired sublinear classical algorithms for solving low-rank linear systems. *arXiv preprint arXiv:1811.04852*, 2018.

Church, K. W. and Gale, W. A. Poisson mixtures. *Natural Language Engineering*, 1(2):163–190, 1995.

Dasgupta, S. Learning mixtures of gaussians. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pp. 634–644. IEEE, 1999.

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.

Dexter, A. and Tanner, D. Packing densities of mixtures of spheres with log-normal size distributions. *Nature physical science*, 238(80):31, 1972.

Fan, X., Yuan, Y., Liu, J. S., et al. The EM algorithm and the rise of computational biology. *Statistical Science*, 25 (4):476–491, 2010.

Farhi, E. and Neven, H. Classification with quantum neural networks on near term processors. *arXiv preprint arXiv:1802.06002*, 2018.

Ghitany, M., Maller, R. A., and Zhou, S. Exponential mixture models with long-term survivors and covariates. *Journal of multivariate Analysis*, 49(2):218–241, 1994.

Gilyén, A., Lloyd, S., and Tang, E. Quantum-inspired low-rank stochastic regression with logarithmic dependence on the dimension. *arXiv preprint arXiv:1811.04909*, 2018a.

Gilyén, A., Lloyd, S., and Tang, E. Quantum-inspired low-rank stochastic regression with logarithmic dependence on the dimension. *arXiv preprint arXiv:1811.04909*, 2018b.

Harrow, A. W., Hassidim, A., and Lloyd, S. Quantum Algorithm for Linear Systems of Equations. *Physical Review Letters*, 103(15):150502, 10 2009. ISSN 0031-9007. doi: 10.1103/PhysRevLett.103.150502. URL http://link.aps.org/doi/10.1103/PhysRevLett.103.150502.

Hastie, T., Tibshirani, R., and Friedman, J. *The Elements of Statistical Learning*, volume 1 of *Springer Series in Statistics*. Springer New York, New York, NY, 2009. ISBN 978-0-387-84857-0. doi: 10.1007/b94608. URL http://www.springerlink.com/index/10.1007/b94608.

Jiang, N., Pu, Y.-F., Chang, W., Li, C., Zhang, S., and Duan, L.-M. Experimental realization of 105-qubit random access quantum memory. *npj Quantum Information*, 5(1): 28, 2019.

Kalai, A. T., Moitra, A., and Valiant, G. Disentangling gaussians. *Communications of the ACM*, 55(2):113–120, 2012.

Kannan, R., Salmasian, H., and Vempala, S. The spectral method for general mixture models. In *International Conference on Computational Learning Theory*, pp. 444–457. Springer, 2005.

Kearns, M., Mansour, Y., and Ng, A. Y. An information-theoretic analysis of hard and soft assignment methods for clustering. In *Learning in graphical models*, pp. 495–520. Springer, 1998.

Kerenidis, I. and Luongo, A. Quantum classification of the MNIST dataset via Slow Feature Analysis. *Physical review letters A*, 101(6):062327, 2020.

Kerenidis, I. and Prakash, A. Quantum recommendation systems. In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

Kerenidis, I. and Prakash, A. Quantum gradient descent for linear systems and least squares. *Physical Review A*, 2020.

Kerenidis, I., Landman, J., Luongo, A., and Prakash, A. q-means: A quantum algorithm for unsupervised machine learning. In *Advances in Neural Information Processing Systems*, pp. 4136–4146, 2019.

Kumar, A. Speaker identification based on gaussian mixture models. https://github.com/abhijeet3922/Speaker-identification-using-GMMs. accessed 20/07/2019.

Lagendijk, R. L., Biemond, J., and Boekee, D. E. Identification and restoration of noisy blurred images using the expectation-maximization algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(7): 1180–1191, 1990.

Li, Z., Liu, X., Xu, N., and Du, J. Experimental realization of a quantum support vector machine. *Physical review letters*, 114(14):140504, 2015.

Lindsay, B. G. Mixture models: theory, geometry and applications. In *NSF-CBMS regional conference series in probability and statistics*, pp. i–163. JSTOR, 1995.

Liu, C. and Rubin, D. B. ML estimation of the t distribution using EM and its extensions, ECM and ECME. *Statistica Sinica*, pp. 19–39, 1995.

Lloyd, S., Mohseni, M., and Rebentrost, P. Quantum algorithms for supervised and unsupervised machine learning. *arXiv*, 1307.0411:1–11, 7 2013. URL http://arxiv.org/abs/1307.0411.

Miyahara, H., Aihara, K., and Lechner, W. Quantum expectation-maximization algorithm. *Personal Communication*, 2019.

Moitra, A. *Algorithmic aspects of machine learning*. Cambridge University Press, 2018.

Moitra, A. and Valiant, G. Settling the polynomial learnability of mixtures of gaussians. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pp. 93–102. IEEE, 2010.

Montanaro, A. Quantum algorithms: an overview. *npj Quantum Information*, 2(1):1–8, 2016.

Murphy, K. P. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Ng, A. Cs229 lecture notes - machine learning. Lecture notes CS229 Stanford, 2012.

Nielsen, M. A. and Chuang, I. Quantum computation and quantum information, 2002.

Otterbach, J., Manenti, R., Alidoust, N., Bestwick, A., Block, M., Bloom, B., Caldwell, S., Didier, N., Fried, E. S., Hong, S., et al. Unsupervised machine learning on a hybrid quantum computer. *arXiv preprint arXiv:1712.05771*, 2017.

Pearson, K. X. contributions to the mathematical theory of evolution.—ii. skew variation in homogeneous material. *Philosophical Transactions of the Royal Society of London.(A.)*, (186):343–414, 1895.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Pensia, A., Jog, V., and Loh, P.-L. Extracting robust and accurate features via a robust information bottleneck. *arXiv preprint arXiv:1910.06893*, 2019.

Reynolds, D. A., Quatieri, T. F., and Dunn, R. B. Speaker verification using adapted gaussian mixture models. *Digital signal processing*, 10(1-3):19–41, 2000.

Subaşı, Y., Somma, R. D., and Orsucci, D. Quantum algorithms for systems of linear equations inspired by adiabatic quantum computing. *Physical review letters*, 122 (6):060504, 2019.

Tang, E. Quantum-inspired classical algorithms for principal component analysis and supervised clustering. *arXiv preprint arXiv:1811.00414*, 2018a.

Tang, E. A quantum-inspired classical algorithm for recommendation systems. *arXiv preprint arXiv:1807.04271*, 2018b.

Voxforge.org. Free speech... recognition - voxforge.org. http://www.voxforge.org/. accessed 20/07/2019.

Wiebe, N., Shankar, R., and Kumar, S. Hardening Quantum Machine Learning Against Adversaries. 2017.

Xu, J. and Mareček, J. Parameter estimation in gaussian mixture models with malicious noise, without balanced mixing coefficients. In *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 446–453. IEEE, 2018.

Yin, J. and Wang, J. A dirichlet multinomial mixture model-based approach for short text clustering. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 233–242. ACM, 2014.