Quantum Implementation of Quadratic Programming for Feature Selection

Amir F. Shibru, Elizabet Yonas, Hope Alemayehu,
Nahome Garefo, Peniel Yohannes

April 2025

Contents

Introduction

In machine learning and data science, the importance of a clean dataset is invaluable. The performance of our algorithm or our model depends heavily on the type of dataset we are dealing with, and in some cases- like medical and financial matters- we encounter datasets that have exceedingly high dimensions(variables), that usually leads to high computational cost or overfitting. there are different methods we can use to solve these kinds of problem there are, the most common ones are:

- Increasing the power of our model.
- Decreasing the dimension of the dataset.

Here, we mainly focus on the second method, specifically feature selection- It is a process of choosing a subset of relevant features (variables) from a dataset to improve model performance, reduce complexity, and enhance interpretability by removing irrelevant or redundant features. Three main benefits can be drawn from successful feature selection method: first, a substantial gain in computational efficiency (especially important for any application that requires classifier execution in real time); second, scientific discovery by determining which features are most correlated with the class labels (which may in turn reveal unknown relationships among features); and, third, reduction of the risk of overfitting if too few training instances are available (a serious problem particularly in situations with high dimensionalities relative to training set sizes).

Many methods have been suggested to solve the variable selection problem. The first is the filter method. It is the most common method for selecting features. This method relies on the characteristic relation between the features themselves instead of using our model performance accuracy as a parameter to select subsets of features. The second is the Wrapper method; this method greedily selects a subset of features that yields highest performance accuracy for our models. The third method is the embedded method; this method incorporates feature selection as a part of the model training process, which means that the model learns the relation between features in addition to the relation to target value.

Filter methods are often preferable to other selection methods because of their usability with alternative classifiers, their computational speed, and their simplicity. However, filter algorithms often score variables separately from each other, so they do not achieve the goal of finding combinations of variables that give the best classification performance. It has been shown that simply combining good variables does not necessarily lead to good classification accuracy. Therefore, a common improvement direction for filter algorithms is to consider the dependencies among the variables. In this direction approaches

based on mutual information, in particular Maximal Dependency (MaxDep) and minimal-Redundancy-Maximum-Relevance (mRMR), have been significant advances.

The central idea of the MaxDep approach is to find a subset of features which jointly have the largest dependency on the target class. However, it is often infeasible to compute the joint density functions of all features and of all features with the class. One approach to making the MaxDep approach practical is the selection of maximum relevance features (MaxRel). This approach selects those features that have the highest relevance (mutual information) to the target class. The main limitation of MaxRel is that it does not account for redundancy between features. The mRMR criterion is another version of MaxDep that chooses a subset of features with both minimum redundancy (approximated as the mean value of the mutual information between each pair of variables in the subset) and maximum relevance (estimated as the mean value of the mutual information between each feature and the target class). Given the prohibitive cost of considering all possible subsets of features, the mRMR algorithm selects features greedily, minimizing their redundancy with features chosen in previous steps and maximizing their relevance to the class.

Even though the mRMR method has some advantages it has its own drawback: the computational cost for high-dimensional data, greedy selection method of features. A new proposed method that we will be discussing in this paper is a method that tries to resolve those drawbacks, QPFS (Quadratic programming for Feature Selection). It proposes a novel formulation of the feature selection task based on quadratic programming[2].

To experiment on QPFS algorithm over a quantum computing domain an online cloud platform, called Dirac-3, was used. Dirac-3 is the latest addition to QCi's Entropy Quantum Computing (EQC) product line, a unique quantum-hardware approach for tackling hard polynomial optimization problems. Dirac-3 uses non-binary qudits as its unit of quantum information (cf. binary qubits), where each quantum state is represented by d2 dimensions. Dirac-3 can solve problems with variables beyond binary (0,1), including non-negative integers and sum-constrained continuous numbers. For further information on qudits, please read through the Qudit Primer to better understand the benefits of high-dimensional programming. To delve deeper into the underlying physics of EQC technology, refer to our paper: An Open Quantum System for Discrete Optimization.[1]

Quadratic Programming for Feature Selection

In Quadratic Programming we define a quadratic objective function that represents our optimization problem. Here in our case let's define the output of our function to be redundancy(R).

$$R(w_1, w_2, ...w_m) = \frac{1}{2} w^T Q w - F^T w \qquad Eq.2.1$$

Where w is M-dimensional vector, $Q \in \mathbb{R}^{MxM}$ is asymmetric positive semi-definite matrix, and F is a vector in $\mathbb{R}^M$ with non-neagtive entries. Our function has two terms: a quadratic term and linear term, the former quantifies the relation between each feature, by assigning high mutual information when two features are highly correlated. The latter captures the independent contribution of each feature to the output. This is important because not all redundancy is due to interaction between features; some features may be redundant simply because they are individually less informative, thus representing relevance in our function. In addition our function has linear constraints It needs to follow, and based on these constraints and quadratic functions we perform our optimization.

Suppose a classifier learning problem involving N training samples and M variables. A quadratic programming problem is to minimize a multivariate quadratic function subject to linear constraint as follows:

$$\min_x \left(\frac{1}{2} w^T Q w - F^T w \right) \qquad Eq.2.2$$

Above, w is an M-dimensional vector, $Q \in \mathbb{R}^{MxM}$ is a symmetric positive semi-definite matrix, and F is a vector in $\mathbb{R}^M$ with non-negative entries. Applied to the feature selection task, the QPFS method assumes that Q represents the similarity among variables (redundancy), F measures how correlated each feature is with the target class(relevance) and the optimal value for w is used for feature ranking by interpreting the components of w as the weight (or importance) of each feature. Thus, features with higher weights are better variables to use for subsequent classifier training. Since w_i represents the weight of each variable, it is reasonable to enforce the following constraints:

$$w_i \geq 0$$
$$\sum_{i=1}^{M} w_i = 1 \qquad Eq.2.3$$

for all $i = 1, ..., M$

In order to provide some flexibility to the model, let introduce a scalar parameter $\alpha \in [0, 1]$ which, depending on the learning problem, over weights the linear and the quadratic terms as follows,

$$\min_x \left(\frac{1}{2} (1 - \alpha) w^T Q w - F^T w \right) \qquad Eq.2.4$$

$$w_i \geq 0$$

$$\sum_{i=1}^{M} w_i = 1$$

for all $i = 1, ..., M$

If $\alpha = 1$, only relevance is considered; the quadratic programming problem becomes linear and equivalent to classical univariate filter methods in which features are ranked according to thier relevance with the class; among these approaches, criteria based in Pearson correlation or mutual information (MaxRel). On the country, if $\alpha = 0$ only independence between features is considered and features with higher weights are those having the lowest similarity coefficients with the rest of features.

Every dataset has its best choice of α to extract the minimum number of features for classfication purposes. The best methdology for determining an appropriate value for α would be to use a validation subset[2]. However, that approach requires evaluating the accuracy of the underlying classifier for each point in a grid of values for α in which case, QPFS would become a wrapper feature selection method instead of a filter method because it would depend on the classifier accuracy to determine the proper value of α. Unfortunately, the evaluation of a classifier for each α value makes the QPFS extremely costly high dimensional datasets, making necessary an heuristic approximation for α providing competitive classification results and less computational load. A reasonable choice of α must balance the linear and quadratic terms of Equation 2.4 in order to ensure that both redundancy and relevance are taking into account. If features are only slightly redundant, i.e. they have low correlation with each other, then the linear term in Equation 2.4 is dominant: $f \gg q$. making α smaall reduces this dominance. On the other hand, if the feature have a high level of redundency relative to relevance ($q \gg f$), then the qudratic term in Equation 2.4 can dominate th elinear one. In this case, overweighting the linear term ($\alpha closeto 1$) makes the objective function to be baalnced. Thus, estimating the mean value q of the elements of the matrix Q and the mean value f of the elements of the vector F as [2]

$$q = \frac{1}{M^2} \sum_{i=1}^{M} \sum_{j=1}^{M} q_{ij} \qquad Eq.2.5$$

$$f = \frac{1}{M} \sum_{i=1}^{M} f_i$$

The relevance and the redundecy terms in Equation 2.4 are balanced when

$(1-\alpha)q = \alpha f$ Then, a reasonable initial estimate of α is

$$\alpha = \frac{q}{q+f}$$

Regarding that the quadratic programming problem given by the equation 2.4 is a convex if the matrix Q is positive semi-definite and strictly convex if matrix Q is positive definite. Hence, one immediate advantage of QPFS formulation is that it is sufficentlly general to permit the use of any symmetric similarity measure as long as Q verifeis these conditions in Q. In this paper we use mutual information as a similarity measures because it can capture non-linear relationships within features and with target class. The general formula to calculate mutual information is given below for a discrete datasets:

$$Q(x_i; y_j) = \sum_{i=1}^{M}\sum_{j=1}^{M} p(x_i, y_j) \log \frac{p(x_i, y_j)}{p(x_i)p(y_j)} \qquad Eq.2.6$$

In the above equation $p(x_i, y_j)$ represent a discrete joint probability distribution and $p(x_i)$ represent a discrete marginal probability distribution. on the other hand for a continuous datasets we need to approximate their discrete probability distribution. In this paper we used KSG(Kraskov-St¨ogbauer-Grassberger) method of estimating mutual information. The KSG2 estimator for mutual information is given by:

$$Q(X;Y) = \psi(k) + \frac{1}{N}\sum_{i=1}^{N} [\psi(n_x(i)+1) + \psi(n_y(i)+1)] - \psi(N) \qquad Eq.2.7$$

where:

- N is the number of data points,
- k is the number of nearest neighbors,
- $n_x(i)$ and $n_y(i)$ are the number of neighbors within the marginal neighborhoods for X and Y,
- $\psi(\cdot)$ is the digamma function.

The quadratic programming formulation of the feature selection problem is elegant and provides insight but the formulation by itself does not significantly reduce the computational complexity of well established feature selection methods like mRMR. Nevertheless, in high-dimensional domains, it is likely that the feature space is redundant. If so, the symmetric matrix Q is singular and Equation 3.4 can then be simplified and solved in a space of dimension less than

M, thus reducing the computational cost.

Given the Eigen-decomposition $Q = U \wedge U^T$ in decreasing order of eigenvalues, Equation 3.4 is equivalent to

$$\min_x \left(\frac{1}{2}(1-\alpha) w^T U \wedge U^T w - F^T w \right) \qquad Eq.2.8$$

If the rank of Q is $k \ll M$, then the diagonalization $Q = U \wedge U^T$can be written as $Q = \overline{U} \overline{\wedge} \overline{U}^T$, where $\overline{\wedge}$ is a diagonal matrix consisting of the highest k eigenvalues of Q in decreasing order and $\overline{U}$ is Mxk matrix consisting of the first k eigenvectors of Q. Then, Equation 2.8 can be rewritten as

$$\min_x \left(\frac{1}{2}(1-\alpha) w^T \overline{U} \overline{\wedge} \overline{U}^T w - F^T w \right) \qquad Eq.2.9$$

Let $z = \overline{U}^T w$ be a vector in $\mathbb{R}^k$. The optimization problem is reduced to minimizing a derived quadratic function in k-dimensional space:

$$\min_x \left(\frac{1}{2}(1-\alpha) z^T \overline{\wedge} z - F^T \overline{U} z \right) \qquad Eq.2.10$$

under $M + 1$ constraints:

$$\overline{U} z \geq 0$$

$$\sum_{i=1}^{M} \sum_{j=1}^{k} \overline{u}_{ij} z_j = 1 \qquad Eq.2.11$$

And the original vector w can be approximated as $w \approx \overline{U} z$.

Nonetheless, this scenario is not practical, because the above method assumes Q is singular when we stipulate the rank $k \ll M$, but Q is seldom precisely singular for real world datasets. However, Q can normally be reasonably approximated by a low-rank matrix formed from its $\overline{k}$ eigenvectors whose eigenvalues are greater than a fixed threshold $\sigma > 0$. more precisely, let $Q = \overline{U D U}^T$ to be approximation of Q, where $D \in \mathbb{R}^{MxM}$ is a diagonal matrix consisting of the $\overline{k}$ highest eigenvalues of Q and the rest of diagonal entries are zero. Then, the approximate quadratic programming problem is formulated as

$$\min_x \left(\frac{1}{2}(1-\alpha) w^T U D U^T w - F^T w \right) \qquad Eq.2.12$$

Equivalently,

$$\min_x \left(\frac{1}{2}(1-\alpha) z^T \overline{D} z - F^T \overline{U} z \right) \qquad Eq.2.13$$

where $z = \overline{U}^T w \in \mathbb{R}^{\overline{k}}, \overline{D} \in \mathbb{R}^{\overline{k}x\overline{k}}$ is a diagonal matrix with the nonzero eigenvalues of D and $\overline{U} \in \mathbb{R}^{Mx\overline{k}}$ are the first $\overline{k}$ eigenvectors of U. The $M+1$ constaints of the optimization problem are now defined as

$$\overline{U}z \geq 0$$

$$\sum_{i=1}^{M}\sum_{j=1}^{k} \overline{u}_{ij} z_j = 1 \qquad Eq.2.14$$

QPFS with Nystrom Approximation

Although choosing k highest eigenvalues for approximation-Equation 2.13 reduces the quadratic programming problem to a lower dimensional space, the bottleneck of QPFS falls now on the diagonalization of the $Q \in \mathbb{R}^{MxM}$ matrix, whose cubic cost, $O(M^3)$, compromises the QPFS scalability. However, taking advantage of the redundancy that typically makes the matrix Q almost singular, it is possible to speed up the diagonalization through the Nystrom approximation[1][3]. When the feature space is highly redundant, the rank of Q is much smaller than M and the Nystrom method can approximate eigenvalues and eigenvectors of Q by solving a smaller eigenproblem using only a subset of rows and columns of Q. Suppose that $k < M$ is the rank of Q which is represented as

$$Q = \begin{pmatrix} A & B \\ B^T & E \end{pmatrix} \qquad Eq.2.15$$

where $A \in \mathbb{R}^{kxk}, B \in \mathbb{R}^{kx(M-k)}, E \in \mathbb{R}^{(M-k)x(M-k)}$, and the rows of [A B] are independent, Then, the eigenvalues and eigenvectors of Q can be calcualated exactly from the submatrix [A B] and the diagonalization of A. let $S = A + A^{-\frac{1}{2}} B B^T A^{-\frac{1}{2}}$ and its diagonalization $S = R\overline{\sum}R^T$ then, the highest k eigenvalues of Q are given by $\overline{\wedge} = \overline{\sum}$ and its associated eigenvectors $\overline{U}$ are calculated as,

$$\overline{U} = \begin{pmatrix} A \\ B^T \end{pmatrix} A^{-\frac{1}{2}} R \overline{\sum}^{-\frac{1}{2}} \qquad Eq.2.16$$

The application of the Nystrom approximation method entails some practical issues:

- A prior knowledge of the rank k of Q is, in general, unfeasible and it is necessary to estimate the number of subsamples r to be used in the Nystrom approximation.
- The r roows of [A B] should be, ideally, linearly independent. If the rank of Q is greater than r or the rows of [A B] are not linearly independent, an approximation of thediagonalization of Q is obtained whose error can be quantified, in general, as $||E - B^T A^{-1} B||$

Although the Nystrom approximation is not error-free, if the redundancy of the feature space is large enough, then good approximations can be achieved, as shown later on. When QPFS+Nystrom is used, not only the diagonalization of the redundancy matrix can be accelerated but also the calculation of the entries of such matrix. As explained above, the Nystrom approximation just needs to know the submatrix [A B] whose calculation requires kxM operations in contrast with the calculation of the whole Q matrix consisting of MxM entries (in fact, only $\frac{M^2}{2}$ entries should be computed due to the symmetry in Q)[2]. In this case, it would be inefficient to compute all the entries of Q only to estimate the value of the parameter so, a slightly different rule for setting the value of is suggested. The Nystrom approximation Q of the original matrix Q is defined as,

$$\overline{Q} = (\overline{q}_{ij}) \begin{pmatrix} A & B \\ B^T & B^T A^{-1} B \end{pmatrix} \qquad Eq.2.17$$

QPFS objective function minization with Dirac-3

After finding the relevance vector and the redundancy matrix, they were passed to the dirac-3 implementation. The dirac-3 can be used either as an Integer solver or a continuous one. For this experiment, as the weights need to be handled in a continuous manner, the continuous solver property was specified with its 'job-type' argument. With this inputs, the program returned a solution that contained the weights for each feature set, with the redundant ones being assigned zero or small value while those with high relevance and less redundancy getting high values. The constraint of QPFS, i.e, the sum of all weights being 1.0 was specified in the code and the result also found to be constarined under this restriction.

Dataset description

The selection of datasets for this experiment aimed at balancing reasonable yet examinable dimensions of features. A dataset with numerical(both continuous and discrete) feature types was used to produce an objective function that's defined under the constraints of Quadratic Programming. For the first experiment 'Taiwanese bankruptcy prediction' was used as it was suitable for the aimed objective function.The Taiwanese Bankruptcy Prediction dataset contains financial indicators of companies collected from the Taiwan Economic Journal between 1999 and 2009. The dataset aims to predict company bankruptcy based on financial metrics, following the business regulations of the Taiwan Stock Exchange. It was donated on June 27, 2020 and is commonly used for financial risk assessment and corporate bankruptcy prediction. It contains 95 features with 6,819 samples. This dataset was a perfect fit for the experiment as the limit of 100 features was a factor.

Alternative Datasets

For the experiment , different datasets were collected and preprocessed from UCL machine learning repository and Sklearn datasets. These candidate datasets vary both in dimensionality and the type of information they convey. Due to the computing capacity limit these datasets were not used for the experiment, but they can be used to check the effectiveness and efficiency of QPFS under different scenarios and more computing power. The alternative datasets that were under consideration for this experiments were

	Dataset	Dimension	source
	Gene expression		
1	cancer RNA-Seq	801 x 20,531	UCI
2	Isolet	7,797x617	UCI
3	Ifw people	13,233 x 5,828	Sklearn.datasets
4	Musk version 2	6,598x166	UCI
5	Olivetti faces	400x4,096	Sklearn.datasets

Conclusion

In general, different implementations were performed on the classical and QCI's Dirac-3 machine for the QPFS algorithm. These experiments have given insights on the efficiency and working principles of Dirac-3. The main difference observed from the result is that the classical implementation produced weights for each feature with rank based manner. Dirac-3 on the other hand assigned null/0 values for the redundant and less relevant features. As for the most of dataset set analysis part we only want to be concerned about the relevant features, this result can be taken as significant in avoiding the need for determining the amount of features needed.

Code Availability

The code for the classical and quantum implementation is found in the following repository:
https://github.com/Hope-Alemayehu/QPFS-using-Dirac3/tree/main

References

[1] QCI, "Dirac-3 Developer Beginner Guide," *Introduction to Dirac-3*. [Online]. Available: https://quantumcomputinginc.com/learn/module/introduction-to-dirac-3/dirac-3-developer-beginner-guide. [Accessed: Apr. 7, 2025].

[2] Rodriguez-Lujan, R. Huerta, C. Elkan, and C. Cruz, , *"Quadratic programming feature selection,"*, Mach. Learn. Res.,vol. 11, pp. 1491–1516, 2010.

[3] P. L. Williams and Y. Li, "Estimation of mutual information: A survey," in *Proc. 4th Int. Conf. Rough Sets Knowl. Technol.*, 2009, pp. 389–396, doi: 10.1007/978-3-642-02962-2_49.

[4] L. Novelli, , *"Kraskov (KSG) estimator of mutual information,"* , Wolfram Demonstrations Project, 2018. [Online]. Available: https://demonstrations.wolfram.com/KraskovKSGEstimatorOfMutualInformation/. [Accessed: Month Day, Year].