# Using low-rank approximations to speed up Kernel logistic regression algorithm

## Abstract

Logistic regression as a classic classification algorithm, has limitations in that it can only be applied to linearly separable datasets. For linearly indivisible datasets, we can map data to higher-dimensional space through kernel trick. It becomes easier to separate or structure in this higher dimensional space. However, with the increasing scale of data, the use of kernel trick is more and more restricted. In the case of large-scale data, the cost of storage and computing kernel matrix is very large. To solve the problem of kernel matrix overhead, we using the low-rank approximate kernel matrix to speed up the kernel Logistic regression (KLR), and a framework for fast kernel Logistic regression algorithm is proposed. This method combines Sequential Minimal Optimization (SMO), Gradient descent algorithm and Newton iterative algorithm. It not only speeds up the kernel matrix but also improves the classification accuracy by removing redundant information from the data with low-rank.

## Introduction

Logistic Regression Algorithm (LR) is a classic classification algorithm in the field of statistical analysis, machine learning and data mining. It is mainly proposed for the two-class problem and is used to estimate the possibility of event occurrence; the output is equivalent to the model predicting a probability estimate that a data belongs to a positive class. It is a linear classification algorithm with the characteristics of fast solving speed and strong interpretability of prediction results. It provides a useful probability model for exploratory interpretation of data. However, general LR can only be used for linear classification, so it is quite difficult to classify nonlinear feature data. The Logistic regression is called kernel Logistic regression (KLR) after the kernel expansion, it becomes a classifier that can be used for nonlinear feature data classification[1]. According to mapping the sample of the original data to a feature space of high or infinite dimensions, so that the sample is linearly separable within this feature space[2, 3]. Although the mapping rules of this process are usually agnostic, the kernel function can be used to replace the inner product operations on the feature space. Therefore, it is not necessary to know the specific expression of the feature vector in this feature space[4].

Kernel trick[5] is very popular in the field of machine learning in recent years, especially the success of support vector machine based on statistical learning theory[6], resulting in a variety of kernel-based tricks[7], and is widely used in pattern recognition, text classification, signal processing and other fields[8]. Compared to the support vector machine, the KLR's objective function is not the risk minimum function but the maximum likelihood value. Therefore, the KLR will produce the posterior probability value of the class classification, and KLR is also a convex optimization problem[9], the local optimal solution must be the global optimal solution. For a convex optimization problem, the gradient method or Newton iteration method can be used to solve the problem[10], but the two methods need to perform an inverse calculation on an n×n kernel matrix in each iteration, Where n represents the number of samples, usually when the number of n reaches several thousand, the computational time cost becomes very high, even unacceptable, and its time complexity is . In the solution of convex quadratic programming problems with support vector machines, Platt et al proposed a sequence minimum optimization algorithm(SMO)[11], Based on the inspiration of this algorithm, Keerthi et al. gave a fast dual algorithm for calculating KLR in 2005, The algorithm does not need to bring the entire kernel matrix into the iterative step for calculation. Each iteration only optimizes the two values in the sequence, avoiding the calculation of inverting the kernel matrix, etc., so the calculation cost of the iteration is very small[12].

With the continuous development of Internet technology, the scale of data continues to expand, and the application of kernel trick is increasingly restricted. One of the key problems is that kernel matrices are usually dense matrices with very high storage and computational costs, the storage of dense matrices requires space, while the calculation of such matrices requires the cost of , Where n and m represent the number and dimensions of the sample, respectively[13]. The most common solution is to use a finite memory to calculate an approximate kernel matrix[14, 15]. This method not only solves the memory problem, but also speeds up the calculation of the kernel matrix[16, 17]. A variety of kernel matrix approximation methods have been proposed[18], among them, the Nystrom method is the most widely used, and it is also the basis for many approximate kernel matrix algorithms[19, 20]. The main idea of the Nystrom method is to obtain the approximate matrix of the original matrix K by reducing the rank, equivalent to randomly taking m rows and m columns from K, then where is the n × m block matrix in the original matrix K, but . In this paper, KLR's fast dual algorithm and Nystrom method for solving kernel matrix are combined, and three different algorithms based on approximate kernel matrix for solving KLR are proposed.

**Contribution:**

1. Three algorithms for solving KLR quickly are proposed in combination with the approximate kernel matrix method.
2. Explored the efficiency of solving the above three algorithms under different data scales.
3. Proposed the recommendation of KLR algorithm under large-scale data.

## Related Work

### 2.1 Kernel Logistic regression algorithm

For the logistic regression, suppose there is a dataset , Input vector is , class label y is a binary function. Since the result of the two classification is 0 or 1, this is like the mathematical step function, but the step function will mutate at the position of x = 0. This mutation is mathematically difficult to handle. So generally, use the sigmoid function to fit:

The above formula represents the prediction function of y=1 is , we suppose y obeys the Bernoulli distribution and takes values of 0 and 1, Then there are the following two formulas:

For the two expressions above, we can combine them into the following expression:

If the given samples are independent, we can construct a likelihood function and then use the idea of Maximum Likelihood (ML) to solve the parameters. However, in order to satisfy the theory of minimizing risk, the idea of MLE can be transformed into a minimization of risk theory, and maximizing the likelihood function is equivalent to minimizing the negative likelihood function:

It is inconvenient to directly refer to the above formula. For the convenience of calculation, the logarithm of the likelihood function:

=

using the fact that can be written as: , where is the projection of in the feature space[4].

So projecting features into high-dimensional spaces is:

=

=

where is the ith row vector of the kernel matrix .

Define:

introduce the , there is:

Therefore, we obtain the likelihood function of KLR:

Finally, the loss function of KLR is obtained:

### 2.2 Gradient descent method for KLR

For the upper loss function, the only variable that needs to be solved is α, Then obtain a gradient by seeking the first derivative:

Therefore, the iterative formula is where β is the step size and λ is the penalty term.

|  |
| --- |
| **algorithm 1：Gradient descent method for KLR** |
| **Input**：D， , parameter: ，  **Output**：  **while not converged do**   1. Generating a kernel matrix K from dataset D 2. Calculate the gradient update direction： 3. update ：   **return** |

### 2.3 Newton iteration method for KLR

For the solution of the upper loss function, it can also be solved by Newton iteration method[3]. Based on the first-order derivative of the above formula, and then the second-order derivative of the equation, there are:

where，

so，

and,

where ***V*** is a diagonal matrix whose elements are

Thus, the iteration formula is:

|  |
| --- |
| **algorithm 2：Newton iteration method for KLR** |
| **Input**：D， , parameter:，  **Output**：  **while** not converged **do**   1. Generating a kernel matrix K from dataset D   update ：  **return** |

### 2.4 SMO method for KLR

If in the KLR, the class label y is 1 or -1, so, we have:

Then merge them into the following expression:

Similarly, using MLE as an estimate, the likelihood function is obtained:

A negative loss function can be obtained by taking negative logarithm on the upper form:

According to the fact theorem[4]: and introduce , another form of loss function for KLR is obtained:

where , C is a penalty item.

For the loss function of the above formula, it is itself a convex quadratic programming problem[21], The direct solution is cumbersome, we can use the Lagrangian multiplier method to get its dual problem and then solve it on the dual form[22].

Define:

Primal problem:

Introducing generalized Lagrange's function[23]:

where represents the Lagrangian multiplier, 。

then the optimality conditions are given by:

thus, ,

Bring the above derived formula into the Lagrangian function, let , Because is a function of ,we can define:

According to the form of, then the above formula can be derived:

because of , , where is monotonous and differentiable, but the domain of the inverse function of is (0,1). It is easy to verify by checking the non-negative nature of the second derivative. If g is a convex function, then G is also a convex function. Combining the previous formula, we can obtain the following function:

Then we use Wolfe duality theory and maximize the primal problem to obtain its dual form[24]:

introduce to the , we get:

To determine the threshold parameter b and the stopping condition of the algorithm, optimization conditions for considering dual problems, Introducing the Lagrange multiplier to the above formula:

where C is the penalty term and β is the Lagrange multiplier.

Then,

Define:

Therefore, the optimal condition for the dual problem can be rewritten as:

Define：

,

,

are all functions of α , usually have

Thus, the termination condition of the dual problem can be obtained:

When there is a pair of to update the vector, it satisfies the , We can assume that there exists an alpha which does not satisfy the optimal solution condition, and it will continue to update until the termination condition is reached. Thought based on coordinate descent, when in the vector satisfies and keeps the equality constraint , to reduce the function, so there are the following definitions:

, ,

where , and we can use the Newton iteration method to solve:

thus, ，

In general, termination conditions do not usually achieve absolute optimal accuracy in practice. So, we can use approximate optimal conditions:,where is a positive tolerance parameter.

|  |
| --- |
| **algorithm 3：SMO method for KLR** |
| **Input**：D， , parameter:，  **Output**：,b  **while not converged do**   1. Generating a kernel matrix K from dataset D 2. initialization，select 3. update and 4. ,   **return** ,b |

### 2.5 Approximate estimation of kernel matrix

In recent years, kernel trick has been widely used, but the computational cost of the kernel matrix usually requires O(n2), where n is the number of samples. As the size of the data continues to expand, the number of samples continues to increase, and the cost of such calculations is increasingly unacceptable. If we use the Nystrom method to approximate the kernel matrix **K**, we only need to spend O(m2n), where m is usually much smaller than n. The resulting approximate matrix is low-rank, which not only preserves the main features of the original matrix, but also reduces the storage space and computational complexity of the kernel matrix[19].

#### 2.5.1 Nystrom method approximation eigenfunction

When the input sample is mapped to the high dimension feature space, In order to calculate the inner product in the feature space, it is assumed that the covariance kernel is substituted in the original space:

Where represents the eigenvalue of the kernel matrix, represents the feature function of the operator, then there is:

represents the probability density of the original space input vector x, and the feature function is orthogonal.

Define:

Take an independent and identically distributed sample from and replace the integral on with the mean:

Then the constraint condition is obtained by orthogonal feature function. So,

Where is a kernel matrix of , , Where the columns are orthogonal, is the diagonal matrix of the element , and , then we can get the relevant approximation:

Finally, the approximation of the i-th eigenfunction by the Nystrom method is obtained:

Where , is the i-th column in the matrix. The above formula can also be seen as projecting a new input **x** onto the i-th feature vector in the feature space[25].

#### 2.5.2 Nystrom method approximate kernel matrix

Using eigen decomposition, the kernel matrix **K** can be decomposed into the following form:

Where is an orthogonal matrix, is a diagonal matrix in which the diagonal elements are in descending order of . If , take the first p column in the matrix to construct , and . Then can be used to approximate the kernel matrix . is introduced to avoid the generation of singular matrices and increase stability, and σ is a small positive number[26].

If the feature decomposition can be performed, the above approximation method can save a lot of overhead for the calculation of the kernel trick. But the time complexity of matrix feature decomposition is usually O(n3), The usual practice is to calculate the first p eigenvalues and eigenvectors of **K** when calculating the kernel matrix **K**. when , The time complexity is significantly reduced. At this point, the feature equation of all sample points is approximated by the Nystrom method described above. The low rank approximation matrix of **K** can be written as follows:

Where and respectively represent the eigenvalues and eigenvectors of the matrix of all samples obtained by the Nystrom method.

In the approximate formula introduced above, there are:

Where is the i-th eigenvector of the matrix. is an block matrix in the kernel matrix **K**.

Assume a symmetric semi-positive definite kernel matrix, randomly select columns from **K** to form matrix **C**：

Where , If SVD is decomposed on , is obtained, the is an orthogonal matrix, is the diagonal matrix in which the singular values of the block matrix are arranged in descending order. If there is , then the low rank approximation matrix of the previous kernel matrix **K** is:

Where , is the i-th column of the matrix.

## 3. Kernel logistic regression fast algorithm

In order to reduce the computation time of the kernel matrix in the KLR algorithm, we can use the Nystrom method to approximate :

The approximate is then combine with the previous algorithm, saving overall computational overhead. Since the approximate kernel matrix is solved by the Nystrom method, the algorithm that combines the Nystrom method and the SMO method to solve the KLR is called the NSMO-KLR algorithm. The algorithm that combines the Nystrom method and the gradient descent method to solve KLR is called NGD-KLR algorithm. Similarly, the algorithm that combines the Nystrom method with the Newton iterative method to solve KLR is called NNI-KLR algorithm.

### SMO method combined with the approximate kernel matrix

For the NSMO-KLR algorithm, first we select m row vectors from the overall training samples, calculate their eigenvalues and eigenvectors, then bring them into the above equation to approximate the kernel matrix **K**, and finally combine algorithm 3, we can get:

Where

|  |
| --- |
| **algorithm 4：NSMO-KLR** |
| **Input**：D， , parameter:，，Rank  **Output**：**,** b  **while not converged do**   1. Selecting m vectors from the Rank value to calculate the approximate kernel matrix 2. initialization，select 3. Update and 4. ,   **return**  **,** b |

### Gradient descent combined with the approximate kernel matrix

For the NGD-KLR algorithm, first we select m row vectors from the overall training samples, calculate their eigenvalues and eigenvectors, and then bring them into the above equation to approximate the kernel matrix K. Finally, in combination with algorithm 1, there are:

The iteration formula is：

Where β is the step size and λ is the penalty term.

|  |
| --- |
| **algorithm 5：NGD-KLR** |
| **Input**：D， , parameter:，，，Rank  **Output**：  **while not converged do**   1. Selecting m vectors from the Rank value to calculate the approximate kernel matrix 2. Calculate the gradient update direction： 3. Update ：   **return** |

### Newton iteration combined with the approximate kernel matrix

For the NNI-KLR algorithm, Firstly, m row vectors are selected from the overall training samples, and their eigenvalues and eigenvectors are calculated, and then the above equation is added to approximate the kernel matrix **K**. Finally, in combination with algorithm 2, there are:

So, we can get:

Where ***V*** is a diagonal matrix whose elements are

Thus, the iteration formula is：

|  |
| --- |
| **algorithm 6：NNI-KLR** |
| **Input**：D， , parameter:，，Rank  **Output**：  **while not converged do**   1. Selecting m vectors from the Rank value to calculate the approximate kernel matrix   Update ：  **return** |

## Convergence analysis

### 4.1 Convergence analysis of the NSMO-KLR algorithm

Before proving convergence, give the following explanation:

Firstly let , since f is continuous, it means that B⊂A, B is also bounded. In addition, for any sequence, when it will be close to the boundary point of A, where and m is the number of variables . Since SMO is an algorithm based on coordinate descent, each iteration is very clear.

Lemma 1: When :

Proof：

A finite second-order Taylor series expansion will be performed near in :

Where, between t and , not equal to t or . Calculate the second derivative of :

Because there is , We can get this border:.

According to equation (2), order t=0, so,

Therefore, this proves Lemma 1.

Theorem 1：

1. has at least one limit point in A.
2. Each limits point of in A is the solution of the SMO algorithm.

Proof：

According to Lemma 1， ，because of ， has limit points in B and each limit point in also belongs to B , , So every limit point of also belongs to **A**. Since the algorithm is decreasing at each iteration, f is the lower boundary, so is a convergent sequence. By Lemma 1, we can immediately get converge to 0.

Define is the convergence subsequence, denotes the limit point that converges in A. For any , let , , the coordinates of i are selected during the iterative optimization of the r-th step. when , , we can get:

Since the index is finite, there is at least one pair in infinite s, where , redefine this subsequence:

and are continuous functions on , this can be obtained:

Where,

When converges to 0, there are =0，, It can be obtained by Equation 1:.

Thus, when , the is the solution sequence of SMO, and is the convergent limit point.

### 4.2 Convergence analysis of the NGD-KLR algorithm and NNI-KLR algorithm

For the gradient descent algorithm, each iteration update is a step forward with a negative gradient as the search method[27]. Therefore, in the end, it must be linearly convergent. When the number of iterations t tends to infinity, the cost function converges to the optimal solution , the convergence rate is .

Newton's iterative rule is to use the Newton step as the search direction, The Newton step is also the quadratic norm defined by the Hessian matrix at **x**. In general, the Newton method converges very quickly, Subsequent convergence around , once entering the quadratic convergence phase, only a small number of iterations can be used to produce a solution with very high precision. Its specific convergence proof can refer to The theory of Newton's method（Gal\_antai）[28] and Convex Optimization(Stephen Boyd)[9].

## Computational Complexity

The overall complexity of the KLR algorithm is mainly dominated by the maximum likelihood risk function and the kernel matrix, and its time complexity is O(n3)[29].The complexity of the kernel matrix operation is O(n2),the NSMO-KLR, NGD-KLR and NNI-KLR algorithm according to the Nystrom method selects the m rows/columns to approximate the kernel matrix. So the complexity of the kernel matrix becomes O(m2n)[19], where n is the number of samples, and m is the number of subsamples selected, usually m<<n. The SMO algorithm is mainly solved for the quadratic programming(QP) optimization problem, and its time complexity is O(n). Assuming the model stops within I iteration, the NSMO-KLR algorithm overall computational complexity is O(Im2n).

## Experiment

### 6.1 Experimental Setup

In order to compare the difference between the calculation time cost and the classification accuracy of the above six algorithms in solving KLR, We selected 10 different public datasets from UCI, where the sample features of the banana dataset are non-linear.

Table1 Properties of datasets

|  |  |  |  |
| --- | --- | --- | --- |
| Dataset | Number of features | Number of training examples | Number of test examples |
| ionosphere | 34 | 245 | 105 |
| Australian | 14 | 483 | 207 |
| diabetes | 8 | 537 | 230 |
| banknote | 4 | 960 | 411 |
| titanic | 3 | 1540 | 660 |
| cancer | 9 | 1939 | 831 |
| waveform | 40 | 2341 | 1003 |
| banana | 2 | 3710 | 1590 |
| mushroom | 22 | 3950 | 1693 |
| image | 18 | 3465 | 8085 |

Table 1 lists the details of the various datasets.

Table2 Experimental Parameters

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Method | SMO | NSMO-KLR | | GD | NGD-KLR | | | NI | NNI-KLR | | |
| 参数  Dataset |  |  |  |  |  |  |  |  |  |  |  |
| ionosphere | 2 | 2 | 15 | 3 | 2 | 0.2 | 25 | 0.05 | 15 | 0.05 | 25 |
| Australian | 2.8 | 3 | 55 | 5 | 8 | 2 | 55 | 0.2 | 3 | 0.5 | 100 |
| diabetes | 3.8 | 1.8 | 35 | 6 | 10 | 0.5 | 55 | 5 | 3 | 5 | 55 |
| banknote | 2 | 1.5 | 15 | 8 | 10 | 0.1 | 25 | 0.2 | 3 | 10 | 100 |
| titanic | 0.2 | 0.2 | 15 | 3 | 2.5 | 0.01 | 15 | 0.05 | 15 | 0.05 | 25 |
| cancer | 0.2 | 0.2 | 15 | 3 | 2.3 | 0.5 | 25 | 0.05 | 25 | 0.5 | 25 |
| waveform | 10 | 10 | 15 | 10 | 40 | 0.5 | 15 | 0.05 | 25 | 0.5 | 25 |
| banana | 0.2 | 0.2 | 80 | 1.9 | 1.9 | 0.5 | 55 | 0.5 | 2 | 0.5 | 55 |
| mushroom | 2 | 1.8 | 100 | 10 | 10 | 0.5 | 100 | 0.5 | 3 | 5 | 100 |
| image | 0.2 | 0.5 | 200 | 3 | 4.5 | 0.1 | 35 | 0.5 | 2 | 0.5 | 55 |

Table 2 gives the experimental parameter settings. In all of the following experiments, we mainly used Gaussian kernel . Convergence condition parameter . The , we take its initial value as , In the SMO algorithm, we must let . The number of class1 and class2 in the training set is and . Finally, is initialized to and . in addition to this, can be initialized to 0.

### 6.2 Experimental Results

Table3 Computational Costs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Method  Dataset | SMO | NSMO-KLR | GD | NGD-KLR | NI | NNI-KLR |
| ionosphere | **1.2** | 3.5 | 14.3 | 27.3 | 20.1 | 4.7 |
| Australian | 4.9 | **3.1** | 22.2 | 18.3 | 30.5 | 16.2 |
| diabetes | 5.9 | **3.4** | 18.1 | 18.3 | 18.1 | 6.4 |
| banknote | 16.3 | **5.9** | 41.7 | 25.9 | 231.9 | 103.1 |
| titanic | 61.3 | **30.8** | 76.9 | 45.9 | 2264.2 | 502.7 |
| cancer | 101 | **75.8** | 155.1 | 109 | 1875.9 | 896.4 |
| waveform | 113 | **19.4** | 220.3 | 106.8 | 610.7 | 263.5 |
| banana | 325 | **63.4** | 439.5 | 376 | 4907.2 | - |
| mushroom | 348.5 | 124.7 | 599.6 | 293 | 170.5 | **43.3** |
| image | 287.7 | **255.4** | 2167 | 1548.9 | - | - |

Table 3 lists the computational costs (in seconds) for the above 6 algorithms to achieve equal convergence conditions on different datasets. When the calculation cost exceeds 5000 seconds, the "-" is used instead.

Table4 Classification Accuracy

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Dataset | SMO | NSMO-KLR | GD | NGD-KLR | NI | NNI-KLR | SVM | LR |
| ionosphere | **0.9622** | **0.9622** | 0.9339 | 0.9339 | 0.9339 | 0.9057 | 0.9433 | 0.8801 |
| Australian | 0.8599 | 0.8647 | 0.8599 | **0.8696** | 0.8406 | 0.8454 | 0.8599 | 0.8550 |
| diabetes | **0.7965** | 0.7922 | 0.7835 | 0.7835 | 0.7835 | 0.7878 | 0.6797 | 0.7835 |
| banknote | 0.9976 | **1.0** | 0.9976 | 0.9976 | **1.0** | 0.9951 | **1.0** | 0.9879 |
| titanic | 0.7927 | 0.7943 | 0.7912 | **0.8167** | 0.7837 | 0.7791 | 0.7836 | 0.7791 |
| cancer | 0.9747 | **0.9819** | 0.8351 | 0.7942 | 0.9278 | 0.8122 | 0.9747 | 0.7701 |
| waveform | 0.9143 | 0.9193 | 0.9123 | 0.9143 | 0.9173 | 0.9163 | 0.9213 | **0.9277** |
| banana | 0.9132 | **0.9138** | 0.8943 | 0.8949 | 0.9006 | 0.9006 | - | - |
| mushroom | 0.9994 | 0.9982 | 0.9947 | 0.9929 | 0.9947 | 0.9548 | **1.0** | 0.9728 |
| image | 0.9856 | **0.9873** | 0.9431 | 0.8883 | - | - | 0.9612 | 0.8355 |

Table 4 lists the classification accuracy of the eight algorithms on different datasets. For datasets that cannot be classified by a method or that calculate time spent over 5000 seconds, the accuracy is replaced by "-".

Table5 Accuracy Improvement

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dataset | SMO VS SVM and LR | NSMO-KLR VS  SMO | NGD-KLR VS  GD | NNI-KLR VS  NI |
| ionosphere | **0.0189** | 0 | 0 | -0.0282 |
| Australian | 0 | **0.0048** | **0.0097** | **0.0048** |
| diabetes | **0.013** | -0.0043 | 0 | **0.0043** |
| banknote | -0.0024 | **0.0024** | 0 | -0.0049 |
| titanic | **0.0091** | **0.0016** | **0.0255** | -0.0046 |
| cancer | 0 | **0.0072** | -0.0409 | -0.1156 |
| waveform | -0.0134 | **0.005** | **0.002** | -0.001 |
| banana | **0.0098** | **0.0006** | **0.0006** | 0 |
| mushroom | -0.0006 | -0.0012 | -0.0018 | -0.0399 |
| image | **0.0244** | **0.0017** | -0.0548 | - |

Table 5 shows the changes in accuracy of SMO compared to SVM and LR, Also the NSMO-KLR and NNI-KLR algorithms compared to the Accuracy changes of the SMO and NI methods, respectively.

## Conclusion

For KLR, the fast-dual SMO algorithm performs very well on datasets of different scales. By combining the approximate solution of kernel matrix, we can achieve better results on this basis, not only does it have a large improvement in the speed of the solution, but the accuracy also surpasses the gradient descent method, Newton iteration method and common algorithms such as SVM and LR. For the part itself is a linearly separable dataset, using kernel function may not improve the accuracy of its classification, but rather increases its training time. For a non-linearly separable dataset such as banana, it is necessary.

For some datasets such as ionosphere, Using the Nystrom method to approximate the kernel matrix the global cost will become larger. This is because the same accuracy error as in the case of solving the original kernel matrix in the fewer samples leads to an increase in the number of iterations. Although the time spent on a single iteration has decreased, the overall time spent has increased. In addition, since the rank of the approximate kernel matrix is required to be controlled by the Rank parameter. If the Rank value is set unreasonably, it will also increase the number of iterations.

The two algorithms of gradient descent and Newton iteration involve the operation of the entire kernel matrix in the iteration process, including the operation of inversion of the kernel matrix. When an approximate kernel matrix is used instead, the computational costs are significantly reduced, but the accuracy is also reduced. Such as cancer and image datasets, when using the approximate kernel matrix, re-inverting the approximate kernel matrix and finding the Hessian matrix will lose most of the relevant information. Therefore, using the kernel matrix approximation to accelerate the solution of the two methods will reduce the classification accuracy. At the same time, since the approximate kernel matrix is a low rank approximation of the previous kernel matrix, redundant irrelevant classification information is reduced, and the classification accuracy is improved for most common datasets.

## References

1. Che, N., J.J.A.i.D.A. Wojtusiak, and Classification, *Extreme logistic regression.* 2016. **10**(1): p. 27-52.

2. Jaakkola, T. *Probabilistic kernel regression models*. in *Proc. 1999 Conference on AI and Statistics*. 1999.

3. Zhu, J., T.J.J.o.C. Hastie, and G. Statistics, *Kernel Logistic Regression and the Import Vector Machine.* 2005. **14**(1): p. 185-205.

4. Smola and J. Alexander, *Learning with kernels*. 2002: MIT Press. 2165 - 2176.

5. Kang, Z., C. Peng, and Q.J.N. Cheng, *Kernel-driven Similarity Learning.* 2017.

6. Vapnik, V.N.J.T., *The Nature of Statistical Learning Theory.* 1997. **38**(4): p. 409-409.

7. Zhou, J., et al., *Incorporating prior knowledge and multi-kernel into linear programming support vector regression.* 2015. **19**(7): p. 2047-2061.

8. Lin, S. and J.J.I.T.o.C. Zeng, *Fast Learning With Polynomial Kernels.* 2018. **PP**(99).

9. Boyd, Vandenberghe, and F.J.I.T.o.A. Control, *Convex Optimization.* 2006. **51**(11): p. 1859-1859.

10. Maalouf, M., T.B. Trafalis, and I.J.K.-B.S. Adrianto, *Kernel logistic regression using truncated Newton method.* 2014. **71**(4): p. 339-344.

11. Platt and C. John. *Fast training of support vector machines using sequential minimal optimization*. 2000.

12. Keerthi, S.S., et al., *A Fast Dual Algorithm for Kernel Logistic Regression.* 2005. **61**(1-3): p. 151-165.

13. Mangasarian, O.L. and D.R.J.M.L. Musicant, *Large Scale Kernel Regression via Linear Programming.* 2002. **46**(1-3): p. 255-269.

14. Ubaru, S., Y. Saad, and A.K.J.N.C. Seghouane, *Fast Estimation of Approximate Matrix Ranks Using Spectral Densities.* 2017. **29**(5): p. 1317-1351.

15. Stražar, M. and T. Curk, *Learning the kernel matrix via predictive low-rank approximations.* 2016.

16. Bach, F.R. and M.I. Jordan. *Predictive low-rank decomposition for kernel methods*. in *International Conference on Machine Learning*. 2005.

17. Fine, S. and K.J.J.o.M.L.R. Scheinberg, *Efficient svm training using low-rank kernel representations.* 2001. **2**(2): p. 243-264.

18. Cortes, C., M. Mohri, and A.J.J.o.M.L.R. Rostamizadeh, *Algorithms for Learning Kernels Based on Centered Alignment.* 2012. **13**(2): p. 795-828.

19. Williams, C.K.I. and M. Seeger. *Using the Nyström method to speed up kernel machines*. in *International Conference on Neural Information Processing Systems*. 2000.

20. Gittens, A. and M.W.J.E.A. Mahoney, *Revisiting the Nystrom Method for Improved Large-Scale Machine Learning.* 2013: p. 567-575.

21. Williams, C.K.I., D.J.P.A. Barber, and M.I.I.T. on, *Bayesian Classification With Gaussian Processes.* 1998. **20**(12): p. 1342-1351.

22. Cowell, R., Z.J.S.f.A.I. Ghahramani, and Statistics, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics.* 2005.

23. Cauwenberghs, G. *Kernel machine learning: a systems perspective*. in *The IEEE International Symposium on Circuits and Systems, 2001. Tutorial Guide: ISCAS*. 2001.

24. Wolfe, P.J.Q.a.m., *A duality theorem for nonlinear programming.* 1961: p. 239-244.

25. Baker, C.T.H.J.M.o.C., *The numerical treatment of integral equations.* 1977. **15**(76): p. 323-337.

26. Neal, R.M. *Regression and Classification Using Gaussian Process Priors*. in *Bayesian Statistics*. 2008.

27. Gower, R.M., P.J.S.J.o.M.A. Richtárik, and Applications, *Randomized Iterative Methods for Linear Systems.* 2015. **36**(4).

28. Galántai, A.J.J.o.C. and A. Mathematics, *The theory of Newton's method.* 2000. **124**(1): p. 25-44.

29. Elbashir, M.K. and J.J.I.A.J.I.T. Wang, *Kernel Logistic Regression Algorithm for Large-Scale Data Classification.* 2015. **12**(5): p. 465-472.