# 快速核Logistic回归算法

## 摘 要

逻辑回归作为一个经典的分类算法，其局限性在于只能作用于线性可分的数据集，而对于线性不可分的数据集，则可以通过内核方法将数据映射到更高纬的空间，希望在这个更高维空间中变得更容易分离或结构化。但是，随着数据的不断增大，核方法的使用越来越受限制，在大规模数据的情况下，存储和计算核矩阵的开销都非常的大。因此，本文为了解决核矩阵开销问题，通过低秩近似核矩阵加速了核Logistic回归（Kernel logistic regression, KLR）算法的求解过程。该方法还结合了序列最小最优化算法（Sequential minimal optimization, SMO），不仅加速了核矩阵的求解，还通过低秩去掉了数据中的冗余信息提高了分类精确度。

## 一、引言

逻辑回归(Logistic Regression, LR)算法是统计分析、机器学习和数据挖掘领域的一个经典分类算法，主要针对二分类问题提出，用于估计事件发生的可能性，输出等价于模型预测某个数据点属于正类的概率估计。它是一种线性的分类算法，具有求解速度快、预测结果可解释性强的特点，它为探索性解释数据提供了一个有用的概率模型。但是，一般的LR只能用于线性分类，对于非线性的特征数据分类就十分的困难。Logistic回归经过核化扩充之后的分类器称之为核Logistic回归（Kernel logistic regression, KLR）, 这样就变成了可用于非线性特征数据分类的分类器[1]。通过将原数据的样本映射到一个高维或无穷维的特征空间，使得样本在这个特征空间内线性可分[2, 3]。虽然此过程的映射规则通常是不可知的，但是可以用核函数(Kernel function)去代替特征空间上的内积运算，因此并不需要知道此特征空间里特征向量的具体表达形式。[4]

核方法[5]在近年来机器学习领域十分流行，尤其是基于统计学习理论的支持向量机[6]的成功，由此产生了多种基于核函数的方法[7]，并且被广泛的应用于模式识别、文本分类、信号处理等多个领域[8]。相比于支持向量机，KLR的目标函数不是风险最小函数，而是最大似然值。因此KLR的解会产生类别分类的后验概率值，同时KLR也是一个凸优化问题[9]，局部最优解一定是全局最优解。而对于一个凸优化问题，可以运用梯度法或牛顿迭代法进行求解[10]，但是该两种方法在每次迭代过程中都需要对一个的核矩阵进行求逆等运算，其中表示样本个数，通常的数量达到几千个时，计算时间代价就变得非常的大，甚至难以接受，其时间复杂度为。在支持向量机的凸二次规划问题求解中，Platt等人提出了序列最小最优化算法（Sequential minimal optimization, SMO）[11]，基于此算法思想的启发，Keerthi等人又在2005年给出了一种用于计算KLR的快速对偶算法，该算法不需要将整个核矩阵带入迭代步骤里进行计算，每一次迭代只优化序列里的两个值，避免了对核矩阵进行求逆等操作，因此迭代的计算代价十分的小[12]。

随着互联网技术的不断发展，数据规模的不断扩大，核方法的应用越来越受限制。其中一个关键的问题是核矩阵通常都是稠密矩阵，其存储和计算代价都非常的高，存储稠密矩阵需要的空间，而计算这样的矩阵则需要的代价，这里和分别代表了样本的个数和维度[13]。最常用的解决方法是用有限的内存来计算得到一个近似的核矩阵[14, 15]，这种方法不仅解决了内存问题，也加快了核矩阵的计算[16, 17]。目前已有多种核近似方法[18]被提出，其中Nystrom方法的应用最为广泛，也是之后很多求解近似矩阵算法的基础[19, 20]。Nystrom方法的主要思想是通过降低秩来得到原始矩阵的近似矩阵，相当于从中随机取行列，则,其中是原始矩阵K中的块矩阵，而。本文将KLR的快速对偶算法和近似求解核矩阵的Nystrom方法相结合，提出了3种不同的基于近似核矩阵的快速求解KLR的算法。

本文贡献：

1. 联合近似求解核矩阵方法提出了3种快速求解KLR的算法。
2. 探索了不同数据规模下，以上3种算法的求解效率问题。
3. 给出了KLR算法在大规模数据下的求解建议。

## 二、相关工作

### 2.1 核Logistic回归算法

对于Logistic回归,其假设有数据集,输入向量为,类标签y为二值函数。由于二分类结果是0或者1，这与数学的阶跃函数很类似，但是阶跃函数在x=0的位置会发生突变，这个突变在数学上很难处理。所以一般使用sigmoid函数来拟合：

上式表示y=1的预测函数为。在这里，假设y服从伯努利分布，取值为0和1，那么得到下面两个式子：

对于上面的两个表达式，可以将其合并为以下表达式：

假定给的样本是独立的，便可以构造出似然函数，然后使用极大似然估计(Maximum Likelihood, ML)的思想来求解参数。但是，为了满足最小化风险理论，可以将MLE的思想转化为最小化风险理论，最大化似然函数其实就等价于最小化负的似然函数。使用MLE推导的过程如下:

直接对上面的式子求导会不方便，为了便于计算，对似然函数取对数：

=

损失函数便可以通过最小化负的似然函数得到，即：

根据表达定理：，此处的为在特征空间的投影[4]。

于是将特征投影到高维空间有：

=

=

这里的是核矩阵的第i行向量。令，

引入正则项

所以，得到KLR的似然函数：

最终得到KLR的损失函数：

### 2.2 核Logistic回归的梯度下降法求解

对于上式损失函数，需要求解的变量只有，于是对其求一阶导得到梯度：

所以，迭代公式则为： ，其中为步长，为惩罚项。

|  |
| --- |
| **算法1：梯度下降算法** |
| **输入**：D， ,参数，  **输出**：  **while not converged do**   1. 通过数据集D生成核矩阵**K** 2. 计算梯度更新方向： 3. 更新 ：   **return** |

### 2.3 核Logistic回归的牛顿迭代法求解

对于上式损失函数的求解，也可以采用牛顿迭代法进行求解[3]。基于上式的一阶导，再对该式求二阶导，则有：

其中，

所以，

因此，

这里的***V***是一个对角矩阵，其元素是

于是，迭代公式则为：

|  |
| --- |
| **算法2：牛顿迭代算法** |
| **输入**：D， ,参数，  **输出**：  **while** not converged **do**   1. 通过数据集D生成核矩阵**K**   更新 ：  **return** |

### 2.4 核Logistic回归的SMO求解

如果KLR中，类标签y是1或-1，即,则

将其合并为以下表达式：

同样，使用MLE作估计，得到似然函数：

对上式取负对数便能得到另一种形式的损失函数：

再根据表达定理：，并引入正则项

得到KLR的另一种损失函数形式：

这里的**，**C表示惩罚项。

对于上式的损失函数，其本身是一个凸二次规划问题[21]，如果直接求解是比较繁琐的，于是可以使用拉格朗日乘子法得到其对偶问题，然后在对偶形式上进行求解[22]。

首先定义

原问题:

引入广义拉格朗日函数（Generalized Lagrange function）[23]:

此处的表示拉格朗日乘子，。令对的偏导为零：

于是有， ,

将上面求导得到的式子带入到拉格朗日函数中，并令 ,因为是的函数，所以令：

根据的形式，于是可以对上式求导得到：

由于，因此 ，此处是单调、可微的，而的反函数的定义域是(0,1)。通过检查二阶导数的非负性很容易验证如果g是一个凸函数，那么G也是一个凸函数，综合前面的式子，可以得到以下函数：

再利用Wolfe 对偶理论并使原问题最大化来得到其对偶形式[24]：

带入 , , 并去掉负号得到：

考虑对偶问题的优化条件，为了确定阈值参数b和算法的停止条件，对上式再次引入拉格朗日乘子则有：

这里的C为惩罚项，是拉格朗日乘子。

又有:

定义：

因此，对偶问题的最优条件可以改写为：

再令： , 其中

, 其中

此处， 都是关于的函数，通常有

于是便可以得到该对偶问题的终止条件：

在对向量更新的时候只要还存在一对满足时，就可以认为还存在不满足最优解条件的，就会持续更新下去，直到达到终止条件。基于坐标下降的思路，当向量中的处满足并保持等式约束的时候，要让函数减小，于是有以下定义：

, ,

这里， ,可以使用牛顿迭代法来求解:

最后， ，

而对于终止条件，在实际操作中通常不会达到绝对的最优精确度。因此，可以使用近似的最优条件：, 这里的是精度参数。

|  |
| --- |
| **算法3：SMO算法** |
| **输入**：D， ,参数，  **输出**：,b  **while not converged do**   1. 通过数据集D生成核矩阵**K** 2. 初始化，选择 3. 更新和 4. 令,   **return** ,b |

### 2.5 核矩阵的近似估计

近年来核方法使用非常的广泛，而核矩阵的计算代价则通常需要O(n2)，这里的n表示样本的数量。随着数据规模的不断扩大，样本数量也不断的增加，这样的计算代价越来越难以接受。而如果使用Nystrom方法去近似核矩阵K，则只需要花费O(m2n)的代价。这里m通常远小于n，而最终得到的近似矩阵是低秩的，这样不仅保留了原矩阵的主要特征，还降低了核矩阵的存储空间和计算时间的复杂度[19]。

#### 2.5.1 Nystrom方法近似特征函数

当输入样本映射到高纬特征空间，为了计算特征空间里的内积,假定在原空间中用协方差核去代替：

这里表示核矩阵的特征值，表示算子的特征函数，于是有：

表示原空间输入向量的概率密度，而特征函数是正交的，令：

从中取独立同分布的样本然后用均值去代替上的积分：

再由正交的特征函数得到约束条件 ,因此，就有：

这里是的核矩阵，列是正交的，是元素为的对角矩阵，其中，于是可以得到相关近似：

综合以上式子，最终得到Nystrom方法对第i个特征函数的近似：

其中，是矩阵中的第i列。上式也可以看成是在特征空间中将一个新的输入投影到了第i个特征向量上[25]。

#### 2.5.2 Nystrom方法近似核矩阵

使用特征分解可以将核矩阵**K**分解成如下形式：

其中，是一个正交矩阵，是对角线元素为降序排列的对角矩阵。如果，取矩阵中的前p列构建，且。于是，就可以用来近似核矩阵。引入是为了避免产生奇异矩阵，增加稳定性。而是很小的一个正数[26]。

如果能进行特征分解的话，上述的近似方法能够为核方法的计算节约很大的开销。但是矩阵特征分解的时间复杂度通常是O(n3)，所以通常的做法是在计算核矩阵K的时候，只需要先计算K的前p个特征值和特征向量，当时，时间复杂度就有明显的降低。此时再通过上面描述的Nystrom方法来近似得到所有样本点的特征方程，而**K**的低秩近似矩阵就可以写成：

此处的分别表示用Nystrom方法得到的全部样本矩阵的特征值和特征向量。带入上面的近似式子中，则有：

这里的是矩阵的第i个特征向量，而是核矩阵**K**中的块矩阵。

假定对称半正定核矩阵 , 从**K**中随机选取列组成矩阵**C，**如以下形式：

其中。如果对进行SVD分解得到，这里是正交矩阵，而是块矩阵的奇异值按降序排列的对角矩阵。如果有，那么原核矩阵**K**的低秩近似矩阵则为：

这里, 是矩阵的第i列。

## 三、核Logistic回归快速求解算法

为了减少KLR算法中核矩阵的计算时间，可以利用Nystrom方法去近似求解**，**即：

然后将近似的带入前面的算法中，从而节省了整体的计算开销。

### SMO联合近似核矩阵求解算法

先从整体训练样本中选取m个行向量，计算其特征值和特征向量，然后带入上式去近似核矩阵K，最后结合算法3,则有：

其中，

|  |
| --- |
| **算法4：SMO联合近似核矩阵求解算法** |
| **输入**：D， ,参数，，Rank  **输出**：**,** b  **while not converged do**   1. 通过Rank值去选取m个向量求解近似核矩阵 2. 初始化，选择 3. 更新和 4. 令,   **return**  **,** b |

### 梯度下降联合近似核矩阵求解算法

先从整体训练样本中选取m个行向量，计算其特征值和特征向量，然后带入上式去近似核矩阵K，最后结合算法1,则有：

其迭代公式则为：，此处为步长，为惩罚项。

|  |
| --- |
| **算法5：梯度下降联合近似核矩阵求解算法** |
| **输入**：D， ,参数，，，Rank  **输出**：  **while not converged do**   1. 通过Rank值去选取m个向量求解近似核矩阵 2. 计算梯度更新方向： 3. 更新 ：   **return** |

### 3.3 牛顿迭代联合近似核矩阵求解算法

先从整体训练样本中选取m个行向量，计算其特征值和特征向量，然后带入上式去近似核矩阵K，最后结合算法2,则有：

带入，因此

这里的***V***是一个对角矩阵，其元素是

于是，迭代公式则为：

|  |
| --- |
| **算法6：牛顿迭代联合近似核矩阵求解算法** |
| **输入**：D， ,参数，，Rank  **输出**：  **while not converged do**   1. 通过Rank值选取m个行向量求解近似核矩阵   更新 ：  **return** |

## 四、收敛性分析

### 4.1 SMO联合近似核矩阵求解算法收敛性分析

在证明收敛性之前，先给出以下说明：

首先让,因为是连续的，所以意味着，也是有边界的。另外，对于任意的序列,当时就会接近A的边界点，而，这里m是变量的个数。由于SMO是基于坐标下降的算法，所以每一次的迭代都是十分明确的。

引理1：当的时候：

proof:

首先，将在附近进行有限二阶泰勒级数展开：

这里，介于t和之间，但是不等于t或，对求二阶导，则有：

又因为有 ，所以，我们就能得到这个边界：。

再根据式2，令t=0，于是有：

因此，引理1得证。

定理1：

1. 在A中至少有一个极值点。
2. A中的里的每一个极值点都是SMO算法的解。

proof:根据引理1，由于，在B中有极值点，且中的每一个极值点也都属于B，又由于，所以的每一个极值点也都属于A。

由于算法在每一次迭代的时候f都在减小，并且f是下边界，所以是一个收敛的序列。由引理1，可以立即得到收敛至0。

令为收敛子序列，表示在A中收敛了的极值点。对于任意的，让,,i的坐标在第r步迭代优化的时候选择。当时，由，得到：

由于index是有限的，在无限的s中，至少存在一对,其中，,为了便于书写，重新定义这个子序列：

和是关于的连续函数，以此可以得到：

这里，有：

当收敛至0的时候，有=0，，再由式1可得，。

所以，只要当的时候，就是SMO所求的解序列，而是收敛了的极值点。

### 4.2 梯度下降和牛顿迭代联合近似核矩阵算法收敛性分析

对于梯度下降算法，每一次的迭代更新都是以负梯度作为搜索方法前进一个步长，因此最终一定是线性收敛的[27]。当迭代次数t趋于无穷时，代价函数收敛到最优解，收敛速率为。

牛顿迭代法则是采用Newton步径为搜索方向，而Newton步径也是x处采用Hessian矩阵定义的二次范数。因此，一般情况下Newton方法收敛很快，在附近二次收敛，一旦进入二次收敛阶段，只需要再经过少量的迭代就可以产生具有很高精度的解。其具体收敛证明可以参考The theory of Newton's method（Gal\_antai）[28]和Convex Optimization(Stephen Boyd)[9]。

## 五、实验

### 5.1 实验设置

为了比较上述6种算法在求解KLR上的计算时间花费和分类精确度的差异，我们选取了10个不同的来自UCI上的公共数据集，其中banana数据集的样本特征是非线性的。表1列出了各种数据集的详细信息：

表1 数据集信息

|  |  |  |  |
| --- | --- | --- | --- |
| 数据集名 | 特征数 | 训练样本数 | 测试样本数 |
| 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 |

表2给出了实验的参数设置。在以下的所有实验中，我们主要采用的是高斯核

,收敛条件参数。对于，我们将其初始值取为，在SMO算法中，必须要让,所以将训练集中class1和class2的数量统计为和,最终被初始化为和 。除此之外，也可以将全部初始化为0 。

表2 实验参数

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Method | SMO | 加速SMO | | 梯度下降 | 加速梯度下降 | | | 牛顿迭代 | 加速牛顿迭代 | | |
| 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 |

### 5.2 实验结果

表3 计算开销

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Method  Dataset | SMO | 加速SMO | 梯度下降 | 加速梯度下降 | 牛顿迭代 | 加速牛顿迭代 |
| 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 | - | - |

表3列举出了上述6种算法在不同数据集上达到同等收敛条件时的计算开销(in seconds)。当计算代价超过5000 seconds时，统一用“-”代替。

表4 分类精确度

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 数据集 | SMO | 加速SMO | 梯度下降 | 加速梯度下降 | 牛顿迭代 | 加速牛顿迭代 | 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 | 0.9034 | - |
| 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 |

表4列举了8种算法在不同数据集上的分类精确度。对于不能用某方法进行分类或者计算时间花销超过5000 seconds的数据集，其精确度用“-”代替。

## 六、总结

对于KLR的求解，采用快速对偶SMO算法在不同规模的数据集上表现都十分的优秀，通过联合核矩阵的近似求解方法，能够在此基础上发挥更好的效果，不仅在求解速度上有很大的提升，精确度也超越了梯度下降法、牛顿迭代法以及SVM和LR等常用算法。对于部分本身就是线性可分的数据集，加了核函数不一定能提高其分类的精确度，反而会增加其训练时间，但是对于例如banana这种非线性可分的数据集就是很必要的了。

在某些例如ionosphere数据集上，采用Nystrom方法去近似求解核矩阵的全局开销反而会变大，这是因为在样本较少的情况下为了达到与原核矩阵求解时同样的精度误差，会导致迭代的次数增加，虽然单次迭代的时间花销有所下降，但是整体的时间花销反而增加了。由于近似核矩阵的秩是需要Rank参数去控制的，如果Rank值设置得不合理，也会导致迭代次数的增加。

而梯度下降和牛顿迭代这两种算法在迭代的过程中整个核矩阵都参与了运算，其中包括了对核矩阵进行求逆等操作，当使用近似核矩阵去代替的时候，计算的开销有明显的下降，但是精准度也会下降。例如cancer和image数据集，在使用近似核矩阵的时候，对近似核矩阵再次进行求逆和求Hessian矩阵会损失掉大部分的相关信息，所以对该两种方法采用核矩阵近似加速求解的时候是会以牺牲分类精确度为代价的。同时，由于近似核矩阵是原核矩阵的低秩近似，所以减少了冗余的无关分类信息，对于大多数公共数据集都提高分类的精确度。

## 七、参考文献

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.