Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2007-025
CBCL-268

May 1, 2007

# Notes on Regularized Least Squares

Ryan M. Rifkin and Ross A. Lippert

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

# Notes on Regularized Least-Squares

**Ryan M. Rifkin**
MIT Center for Biological and Computational Learning
rif@mit.edu

**Ross A. Lippert**
D. E. Shaw Research
ross.lippert@deshaw.com

## Abstract

This is a collection of information about regularized least squares (RLS). The facts here are not "new results", but we have not seen them usefully collected together before. A key goal of this work is to demonstrate that with RLS, we get certain things "for free": if we can solve a single supervised RLS problem, we can search for a good regularization parameter $\lambda$ at essentially no additional cost.

The discussion in this paper applies to "dense" regularized least squares, where we work with matrix factorizations of the data or kernel matrix. It is also possible to work with iterative methods such as conjugate gradient, and this is frequently the method of choice for large data sets in high dimensions with very few nonzero dimensions per point, such as text classifciation tasks. The results discussed here do not apply to iterative methods, which have different design tradeoffs.

We present the results in greater detail than strictly necessary, erring on the side of showing our work. We hope that this will be useful to people trying to learn more about linear algebra manipulations in the machine learning context.

## 1 Notation and Basics

We are given data points $S = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$. We define $S^i$ to be the data set with the $i$th point removed: $S^i = \{(X_1, Y_1), \ldots, (X_{i-1}, Y_{i-1}), (X_{i+1}, Y_{i+1}), \ldots, (X_n, Y_n)\}$. We let $X$ simultaneously refer to the set $\{X_1, \ldots, X_n\}$ and to the $n$ by $d$ matrix whose $i$th *row* is $X_i^t$.

We assume a positive semidefinite *kernel function* $k$, which generalizes the notion of dot product in a Reproducing Kernel Hilbert Space (RKHS) [1, 6]. Commonly used kernels include:

$$\begin{aligned} \text{linear:} \quad & k(X_i, X_j) = X_i^t X_j \\ \text{polynomial:} \quad & k(X_i, X_j) = (X_i^t X_j + 1)^d \\ \text{gaussian:} \quad & k(X_i, X_j) = \exp\left(\frac{||X_i - X_j||^2}{\sigma^2}\right) \end{aligned}$$

The polynomial order $d$ or gaussian bandwidth $\sigma$ must be specified by the user. We define the kernel matrix $K$ to satisfy $K_{ij} = k(X_i, X_j)$. Abusing notation slightly, we allow the kernel function $k$ to take multiple data points and produce a matrix of results: $k(X, X) = K$, and, given an arbitrary point $X_*$, $k(X, X_*)$ is a column vector whose $i$th entry is $k(X_i, X_*)$.

We note that the linear kernel (also called the dot-product kernel) has special computational properties, which we discuss in Section 5.

Given a square matrix $M$, $\text{diag}_m(M)$ denotes the diagonal matrix satisfying $\text{diag}_m(M)_{ii} = M_{ii}$, and $\text{diag}_v(M)$ the column vector satisfying $\text{diag}_v(M)_i = M_{ii}$. We abuse notation somewhat by assuming that fractional division is done elementwise, so a vector divided by a vector yields another vector.

We use the standard convention that $I$ represents an identity matrix of the appropriate size (we occasionally index the size of the matrix for clarity, e.g. $I_n$), and $e_i$ represents a column vector with a 1 in the $i$th position and zeros elsewhere.

## 2 RLS Basics

RLS arises as a Tikhonov minimization problem [2] with a square loss:

$$\min_{f \in \mathcal{H}} \frac{1}{2} \sum_{i=1}^{n} (f(X_i) - Y_i)^2 + \frac{\lambda}{2} ||f||_K^2. \tag{1}$$

The representer theorem [7, 5] guarantees that the solution to (1) can be written as

$$f = \sum_{i=1}^{n} c_i k(X_i, \cdot),$$

for some $c \in \mathbb{R}^n$. Using basic properties of RKHS [1, 6], we can therefore rewrite (1) as

$$\min_{c \in \mathbb{R}^n} \frac{1}{2} ||Y - Kc||_2^2 + \frac{\lambda}{2} c^t K c.$$

Setting the derivative with respect to $c$ to zero, we see that $c$ must satisfy

$$(K + \lambda I)c = Y.$$

We note that $c$ exists and is unique: $K$ is positive semidefinite, so $K + \lambda I$ is positive definite (for $\lambda > 0$). We define $G(\lambda) = K + \lambda I$. Frequently, $\lambda$ will be clear from context, and we just write $G$.

The predictions at the training points will be given by

$$f(X) = Kc = K(K + \lambda I)^{-1} Y = K G^{-1} Y,$$

and the prediction at a new test point $X_*$ is:

$$\begin{aligned} f(X_*) &= \sum c_i k(X_i, X_*) \\ &= k(X, X_*)^t c \\ &= Y^t G^{-1} k(X, X_*). \end{aligned}$$

Note that we are *not* suggesting forming $G^{-1}$ explicitly and manipulating it; we discuss computation below.

## 3 Leave-one-out computations

In general, we need some mechanism for finding a "good" value of the regularization parameter $\lambda$. We are usually interested in finding a function that does well on future examples. Since the distribution over examples is unknown, this is not that easy. One method is to use a "holdout" set: we try a bunch of values of $\lambda$, and for each value, we compute the performance the holdout (also called the development) set, and choose the $\lambda$ that does best on it. If we have enormous quantities of data, a holdout set is a good option. If we have relatively few data points, we might like to use leave-one-out values instead.

The $i$th leave-one-out *value* is $f_{S^i}(X_i)$, the value of the RLS function trained on $S^i$ applied at $X_i$. The $i$th leave-one-out *error* is $Y_i - f_{S^i}(X_i)$. We define $LOOV$ and $LOOE$ to be the vectors of leave-one-out values and errors over the training set. $||LOOE||_2^2$ is considered a good empirical proxy for the error on future points, and we often want to choose parameters by minimizing this quantity.[1] In this section, we develop tools for working with $LOOE$ and $LOOV$ for RLS. We will see that for RLS, working with $LOOE$ and

[1] It is possible to overfit by doing this, but in practice it is often quite effective. Note that combining minimizing $||LOOE||$ with feature selection on the *entire* dataset can overfit disastrously — in general, if we are minimizing $||LOOE||$, we must be able to "form" $f_{S^i}$ without looking at $X_i$ at all.

*LOOV* are computationally effective. Note that this material supersedes Section 3.3 of [4], which contains several typos.

We consider two different approaches (really two different views of the same approach) for finding leave-one-out values. The first approach can be seen in many references, such as [4] or [3]. The second approach, which demonstrates that leave-one-out RLS computations can be treated as augmented linear systems, is (to our knowledge) an original development by the authors.

### 3.1 The Direct Approach

Imagine we knew $f_{S^i}$. Define the vector $Y^i$ via

$$Y^i_j = \begin{cases} Y_j & j \neq i \\ f_{S^i}(X_i) & j = i \end{cases}$$

If we solve (1) but replace $Y$ with $Y^i$, the optimal solution will be $f_{S^i}$. This is easy to see — if we solve a smaller problem by discarding the $i$th example entirely, we will get $f_{S^i}$, and adding the example $(X_i, f_{S^i}(X_i))$ adds nothing to the cost. More formally, for any $f \in \mathcal{H}$,

$$\begin{aligned} \frac{1}{2}\sum_{j=1}^{n}(Y^i_j - f(X_i))^2 + \frac{\lambda}{2}||f||^2_K &\geq \frac{1}{2}\sum_{j\neq i}(Y^i_j - f(X_i))^2 + \frac{\lambda}{2}||f||^2_K \\ &\geq \frac{1}{2}\sum_{j\neq i}(Y^i_j - f_{S^i}(X_i))^2 + \frac{\lambda}{2}||f_{S^i}||^2_K \\ &= \frac{1}{2}\sum_{j=1}^{n}(Y^i_j - f_{S^i}(X_i))^2 + \frac{\lambda}{2}||f_{S^i}||^2_K. \end{aligned}$$

We note in passing that this result is much more general than what we've just written: we did not use the fact that we are optimizing an RKHS norm, or the specific form of the square loss — we used only that the square loss allows us to "construct" a $Y$ value with loss zero.

The above derivation makes it clear that for RLS, we can define the expansion coefficients for $f_{S^i}$ via $c^i = G^{-1}Y^i$, and therefore

$$f_{S^i}(X_i) = (KG^{-1}Y^i)_i.$$

Note that this is all theoretical so far, because in order to form $Y^i$, we need to know $f_{S^i}(X_i)$. However, assuming we have solved RLS, and therefore computed $f_S(X) = KG^{-1}Y$, we can derive a nice expression for $f_{S^i}(X_i)$:

$$\begin{aligned} f_{S^i}(X_i) - f_S(X_i) &= \sum_j (KG^{-1})_{ij}(Y^i_j - Y_j) \\ &= (KG^{-1})_{ii}(f_{S^i}(X_i) - Y_i), \end{aligned}$$

which leads immediately to

$$\begin{aligned} f_{S^i}(X_i) &= \frac{f_S(X_i) - (KG^{-1})_{ii}Y_i}{1 - (KG^{-1})_{ii}} \\ &= \frac{(KG^{-1}Y)_i - (KG^{-1})_{ii}Y_i}{1 - (KG^{-1})_{ii}}. \end{aligned}$$

Therefore,

$$\begin{aligned} \mathit{LOOV} &= \frac{KG^{-1}Y - \mathrm{diag}_m(KG^{-1})Y}{\mathrm{diag}_v(I - KG^{-1})}, \\ \mathit{LOOE} &= Y - \mathit{LOOV} \\ &= Y + \frac{\mathrm{diag}_m(KG^{-1})Y - KG^{-1}Y}{\mathrm{diag}_v(I - KG^{-1})} \end{aligned}$$

$$
\begin{aligned}
&= \frac{\mathrm{diag}_m(I - KG^{-1})Y}{\mathrm{diag}_v(I - KG^{-1})} + \frac{\mathrm{diag}_m(KG^{-1})Y - KG^{-1}Y}{\mathrm{diag}_v(I - KG^{-1})} \\
&= \frac{Y - KG^{-1}Y}{\mathrm{diag}_v(I - KG^{-1})}. \qquad (2)
\end{aligned}
$$

We can simplify our expressions in a way that leads to better computational and numerical properties by noting that

$$
\begin{aligned}
KG^{-1} &= Q\Lambda Q^t Q(\Lambda + \lambda I)^{-1} Q^t \\
&= Q\Lambda(\Lambda + \lambda I)^{-1} Q^t \\
&= Q(\Lambda + \lambda I - \lambda I)(\Lambda + \lambda I)^{-1} Q^t \\
&= I - \lambda G^{-1}.
\end{aligned}
$$

Substituting into our expression for $LOOE$ yields

$$
\begin{aligned}
LOOE &= \frac{Y - KG^{-1}Y}{\mathrm{diag}_v(I - KG^{-1})} \\
&= \frac{Y - (I - \lambda G^{-1})Y}{\mathrm{diag}_v(I - (I - \lambda G^{-1}))} \\
&= \frac{\lambda G^{-1} Y}{\mathrm{diag}_v(\lambda G^{-1})} \\
&= \frac{G^{-1} Y}{\mathrm{diag}_v(G^{-1})} \\
&= \frac{c}{\mathrm{diag}_v(G^{-1})}.
\end{aligned}
$$

### 3.2 The Augmented Linear System Approach

Suppose we consider the augmented optimization problem

$$
\min_{c \in \mathbb{R}^n, \beta \in \mathbb{R}} \frac{1}{2} ||Y - Kc - \beta e_i||_2^2 + \frac{\lambda}{2} c^t K c.
$$

While it is not immediately obvious, we will now proceed to show that solving this $n+1$ variable optimization problem yields the $i$th LOO error.

Taking the derivative of the augmented problem with respect to $c$ and $\beta$ shows that the solution is given by the solution of the augmented linear system

$$
\begin{bmatrix} K + \lambda I & e_i \\ e_i^t K & 1 \end{bmatrix} \begin{bmatrix} c \\ \beta \end{bmatrix} = \begin{bmatrix} Y \\ e_i^t Y \end{bmatrix}.
$$

We begin with an intuitive argument. From inspection of either the original augmented problem of the linear system, it is obvious that $\beta$ will take the value $Y - (Kc)_i = Y_i - e_i^t K c$. Using this fact, the top row of our augmented linear system can be rewritten as

$$
\begin{aligned}
(K + \lambda I)c + e_i(Y_i - e_i^t K c) &= Y \\
\left((I - e_i e_i^t)K + \lambda I\right) c &= Y - e_i Y_i.
\end{aligned}
$$

Consider the $i$th equation in the above system. The $i$th row of $(I - e_i e_i^t)K$ is zero, and the $i$th element on the right hand side is also zero. Therefore $\lambda c_i = 0$, and we conclude that $c_i = 0$. It is now clear that the remaining $c$'s must be the RLS solution over the other $n - 1$ points, and that $\beta$ is a "readout" of the $i$th RLS error.

We can also easily derive the actual value of the LOO error. We take the top equation in the linear system and multiply it (on the left) by $-e_i^t K(K + \lambda I)^{-1}$, yieding

$$
-e_i^t K c - e_i^t K (K + \lambda I)^{-1} e_i \beta = e_i^t K (K + \lambda I)^{-1} Y.
$$

We add this to the bottom equation, obtaining

$$\begin{aligned}\left(1 - e_i^t K(K+\lambda I)^{-1} e_i\right)\beta &= Y_i - e_i^t K(K+\lambda I)^{-1} Y \\ \beta &= \frac{Y_i - e_i^t K(K+\lambda I)^{-1} Y}{1 - e_i^t K(K+\lambda I)^{-1}},\end{aligned}$$

which is the result we derived in Equation 2.

## 4 Searching for $\lambda$ is free

For a single value of $\lambda$, we can solve RLS by solving the linear system (2). In Matlab, we would do this with the 'slash' operator, rather than explicitly forming the inverse. However, usually we do not know a good value of $\lambda$ in advance. We will therefore make use of the eigendecomposition $K = Q\Lambda Q^t$, where $\Lambda$ is diagonal with $\Lambda_{ii} \geq 0$ and $QQ^t = I$. We now see that

$$\begin{aligned}G &= K + \lambda I \\ &= Q\Lambda Q^t + \lambda I \\ &= Q(\Lambda + \lambda I)Q^t,\end{aligned}$$

which of course implies $G^{-1} = Q(\Lambda + \lambda I)^{-1}Q^t$.

It takes $O(n^3)$ time to compute the eigendecomposition.[2] Asymptotically, this is no slower than solving a single linear system, although the constant is worse by maybe a factor of 4. Once we have this eigendecomposition, we can find $c$ for a given $\lambda$ in $O(n^2)$ time by solving

$$c(\lambda) = Q(\Lambda + \lambda I)^{-1}Q^t Y,$$

noting that $(\Lambda + \lambda I)$ is diagonal, so multiplying the vector $Q^tY$ by its inverse is a linear time operation. Once we have $c$, we can also get the predictions $Kc$ in $O(n^2)$ time.

Furthermore, we can compute a single entry of $G(\lambda)^{-1}$ in $O(n)$ time:

$$\begin{aligned}G^{-1}_{ij} &= (Q(\Lambda + \lambda I)^{-1}Q^t)_{ij} \\ &= \sum_{k=1}^{n} \frac{Q_{ik}Q_{jk}}{\Lambda_{kk} + \lambda},\end{aligned}$$

and therefore we can compute $\mathrm{diag}(G^{-1})$, and compute $LOOE$, in $O(n^2)$ time.

In conclusion, it takes $O(n^3)$ to solve a single RLS problem for $c$. If we compute an eigendecomposition of $K$, we can compute $c(\lambda)$ and $||LOOE(\lambda)||$ over a quite fine grid of $\lambda$, at basically no additional cost.

## 5 The Linear Case

We now discuss the special case of a linear kernel $k(X_i, X_j) = X_i \cdot X_j^t$. We assume throughout this section that we are in $\mathbb{R}^d$, with $d < n$. (If $d > n$, we are better off ignoring the linearity, and working with the $n$ by $n$ kernel matrix).

It is easy to see that with a linear kernel, the function we are learning is linear as well:

$$\begin{aligned}f(x) &= c^t k(X, x) \\ &= c^t X x \\ &= w^t x,\end{aligned}$$

[2] In practice, to machine precision. In theory, finding an exact eigendecomposition is as hard as exactly finding the roots of an arbitrary polynomial.

where we define the hyperplane $w$ to be $X^t c$. We can classify new points in $O(d)$ time, using $w$, rather than having to compute a weighted sum of $n$ kernel products (which will usually cost $O(nd)$ time).

In the nonlinear case, there is basically only one technique available: to work with the eigendecomposition of $K$. In the linear case, we have two options. The first is to work with the singular value decomposition of $X$ and to compute leave-one-out values. The second is to work with the eigendecomposition of $X^t X$ and to use a hold-out set to validate $\lambda$.

### 5.1 The SVD Approach

Suppose that we only have a moderate amount of data ($n$ is not too large), and we want to compute leave-one-out values to validate $\lambda$. Then we can work with the SVD of $X$.

The economy-size SVD of $X$ can be written as $X = USV^t$, with $U \in \mathbb{R}^{n \times d}$, $S \in \mathbb{R}^{d \times d}$, $V \in \mathbb{R}^{d \times d}$, $U^t U = V^t V = VV^t = I_d$, and $S$ diagonal and positive definite. (Note that $UU^t \neq I_n$). In Section 3 we derived an expression for the leave-one-out error in terms of $c$ and $G^{-1}$. In the linear case, we can use the SVD to express these quantities directly in terms of $X$, rather than $K$:

$$
\begin{aligned}
K &= XX^t = (USV^t)(VSU^t) = US^2U^t \\
K + \lambda I &= US^2U^t + \lambda I_n \\
&= \begin{bmatrix} U & U_\perp \end{bmatrix} \begin{bmatrix} S^2 + \lambda I_d & \\ & \lambda I_{n-d} \end{bmatrix} \begin{bmatrix} U^t \\ U_\perp^t \end{bmatrix} \\
&= U(S^2 + \lambda I_d)U^t + \lambda U_\perp U_\perp^t \\
&= U(S^2 + \lambda I_d)U^t + \lambda(I_n - UU^t) \\
(K + \lambda I)^{-1} &= (US^2U^t + \lambda I_n)^{-1} \\
&= \left( \begin{bmatrix} U & U_\perp \end{bmatrix} \begin{bmatrix} S^2 + \lambda I_d & \\ & \lambda I_{n-d} \end{bmatrix} \begin{bmatrix} U^t \\ U_\perp^t \end{bmatrix} \right)^{-1} \\
&= \begin{bmatrix} U & U_\perp \end{bmatrix} \begin{bmatrix} S^2 + \lambda I_d & \\ & \lambda I_{n-d} \end{bmatrix}^{-1} \begin{bmatrix} U^t \\ U_\perp^t \end{bmatrix} \\
&= U(S^2 + \lambda I)^{-1}U^t + \lambda^{-1} U_\perp U_\perp^t \\
&= U(S^2 + \lambda I)^{-1}U^t + \lambda^{-1}(I - UU^t) \\
&= U\left[(S^2 + \lambda I)^{-1} - \lambda^{-1} I\right] U^t + \lambda^{-1} I \\
c &= (K + \lambda I)^{-1} Y \\
&= U\left[(S^2 + \lambda I)^{-1} - \lambda^{-1} I\right] U^t Y + \lambda^{-1} Y \\
G_{ij}^{-1} &= \sum_{k=1}^{d} U_{ik} U_{jk} [(S_{kk} + \lambda)^{-1} - \lambda^{-1}] + [i = j]\lambda^{-1} \\
G_{ii}^{-1} &= \sum_{k=1}^{d} U_{ik}^2 [(S_{kk} + \lambda)^{-1} - \lambda^{-1}] + \lambda^{-1} \\
LOOE &= \frac{c}{\mathrm{diag}_v(G^{-1})} \\
&= \frac{U\left[(S^2 + \lambda I)^{-1} - \lambda^{-1} I\right] U^t Y + \lambda^{-1} Y}{\mathrm{diag}_v(U\left[(S^2 + \lambda I)^{-1} - \lambda^{-1} I\right] U^t + \lambda^{-1} I)}
\end{aligned}
$$

### 5.2 The SVD Approach, Direct Derivation

We can obtain the same results via an alternate derivation, which we include for pedagogical purposes. Suppose we phrase the RLS problem directly in terms of $w$, rather than $c$:

$$\min_{w \in \mathbb{R}^d} L(w) = \frac{1}{2}||Y - Xw||_2^2 + \frac{\lambda}{2}||w||_2^2.$$

Taking the derivative with respect to $w$,

$$\frac{\partial L}{\partial w} = X^t X w - X^t Y + \lambda w,$$

and setting to zero implies

$$\begin{aligned} w &= (X^t X + \lambda I)^{-1} X^t Y \\ &= V(S^2 + \lambda I)^{-1} V^t X^t Y \\ (&= V(S^2 + \lambda I)^{-1} V^t (VSU^t) Y) \\ (&= VS(S^2 + \lambda I)^{-1} U^t Y) \end{aligned}$$

Define $w_{\setminus i}$ to be the hyperplane obtained when we remove the $i$th training point and train on the remaining $n-1$ points (the linear analog of $f_{S^i}$). Defining $Y^i$ as before,

$$\begin{aligned} \min_{w \in R^n} \frac{1}{2} \sum_j (Y_j^i - w^t x_j)^2 + \lambda ||w||_2^2 &\geq \min_{w \in R^n} \frac{1}{2} \sum_{j \neq i} (Y_j^i - w^t x_j)^2 + \lambda ||w||_2^2 \\ &= \frac{1}{2} \sum_{j \neq i} (Y_j^i - w_{\setminus i}^t x_j)^2 + \lambda ||w_{\setminus i}||_2^2 \\ &= \frac{1}{2} \sum_{j} (Y_j^i - w_{\setminus i}^t x_j)^2 + \lambda ||w_{\setminus i}||_2^2, \end{aligned}$$

so $w_{\setminus i}$ "solves" RLS with $Y$ replaced with $Y^i$, and

$$w_{\setminus i} = (X^t X + \lambda I)^{-1} X^t Y^i.$$

Precisely analogously to the nonlinear case,

$$\begin{aligned} (w_{\setminus i} - w)^t x_i &= \left( (X^t X + \lambda I)^{-1} X^t (Y^i - Y) \right)^t x_i \\ &= (Y^i - Y)^t X (X^t X + \lambda I)^{-1} x_i \\ &= (w_{\setminus i}^t x_i - y_i) x_i^t (X^t X + \lambda I)^{-1} x_i, \end{aligned}$$

which implies

$$w_{\setminus i}^t x_i = \frac{w^t x_i - y_i x_i^t (X^t X + \lambda I)^{-1} x_i}{1 - x_i^t (X^t X + \lambda I)^{-1} x_i}.$$

Therefore,

$$\begin{aligned} LOOE &= Y - \frac{Xw - \mathrm{diag}_m(X(X^t X + \lambda I)^{-1} X^t) Y}{\mathrm{diag}_v(I - X(X^t X + \lambda I)^{-1} X^t)} \\ &= Y - \frac{X(X^t X + \lambda I)^{-1} X^t Y - \mathrm{diag}_m(X(X^t X + \lambda I)^{-1} X^t) Y}{\mathrm{diag}_v(I - X(X^t X + \lambda I)^{-1} X^t)} \\ &= \frac{Y - X(X^t X + \lambda I)^{-1} X^t Y}{\mathrm{diag}_v(I - X(X^t X + \lambda I)^{-1} X^t)} \end{aligned}$$

Noting that

$$\begin{aligned} X(X^tX + \lambda I)^{-1}X^t &= USV^t(VS^2V^t + \lambda I)^{-1}VSU^t \\ &= US^2(S^2 + \lambda I)^{-1}U^t \\ &= U[S^2 + \lambda I - \lambda I](S^2 + \lambda I)^{-1}U^t \\ &= UU^t - \lambda U(S^2 + \lambda I)^{-1}U^t \\ I - X(X^tX + \lambda I)^{-1}X^t &= I - \lambda U(\lambda^{-1}I)U^t + \lambda U(S^2 + \lambda I)^{-1}U^t \\ &= \lambda\left(\lambda^{-1}I + U\left[(S^2 + \lambda I)^{-1} - \lambda^{-1}I\right]U^t\right), \end{aligned}$$

we can derive the same expression for LOOE as we did in the previous section.

### 5.3 The covariance matrix approach

If we require only $w$ and not the LOO values, we need only the $V$ and $S$ pieces of the SVD. These can be computed from the eigendecomposition of the (nonstandarized) covariance matrix $X^tX$, which is only $d$ by $d$. Furthermore, computation of this matrix is fairly easily parallelizable. For very large datasets in moderate numbers of dimensions, this is probably the method of choice.

## 6 Conclusions

The interplay of the RLS algorithm with simple matrix factorization techniques yields extremely efficient algorithms. For both linear and nonlinear RLS, in the same (asymptotic) time it takes to solve a single problem, we can find a good $\lambda$ using LOO cross-validation. For linear RLS, we have an additional option of using a validation set, which can be even faster in practice.

The primary disadvantage of nonlinear RLS is the need to work with the entire kernel matrix, which is $n$ by $n$. There have been many attempts to approximate the entire function using an expansion over a subset of the data points. We leave the extension of the methods contained here to this case for future work.

## References


[1] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68:337–404, 1950.

[2] Theodoros Evgeniou, Massimiliano Pontil, and Tomaso Poggio. Regularization networks and support vector machines. *Advanced In Computational Mathematics*, 13(1):1–50, 2000.

[3] P. J. Green and B. W. Silverman. *Nonparametric Regression and Generalized Linear Models*. Number 58 in Monographs on Statistics and Applied Probability. Chapman & Hall, 1994.

[4] Ryan M. Rifkin. *Everything Old Is New Again: A Fresh Look at Historical Approaches to Machine Learning*. PhD thesis, Massachusetts Institute of Technology, 2002.

[5] Bernhard Schölkopf, Ralf Herbrich, and Alex J. Smola. A generalized representer theorem. In *14th Annual Conference on Computational Learning Theory*, pages 416–426, 2001.

[6] Bernhard Schölkopf and Alex Smola. *Learning with Kernels*. MIT Press, 2002.

[7] Grace Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial & Applied Mathematics, 1990.