Abstract—In this paper, we investigate the wellposedness of the kernel adaline. The kernel adaline finds the linear coefficients in a radial basis function network using deterministic gradient descent. We will show that the gradient descent provides an inherent regularization as long as the training is properly early-stopped. This understanding provides an alternative and possibly simpler way to obtain regularized solutions comparing with the cross-validation approach in regularization networks.

I. INTRODUCTION

The kernel adaline [1] is derived in a possibly infinite dimensional space with finite training data, and its well-posedness study is crucial for its real world applications. The concept of well-posedness was proposed by Hadamard [2] and is closely related to the concepts such as over-learning, stability, and generalization. Regularization as a remedy for ill-posedness became widely known due to the work of Tikhonov [3], and also has a very principled Bayesian learning interpretation [4]. In least-squares problems, Tikhonov regularization is essentially a trade-off between fitting the training data and reducing solution norms. The solution with smaller norm possesses more stability and guarantees better generalization performance in statistical sense [5].

We utilize the singular value decomposition along with the gradient descent iteration to show that the kernel adaline is well-posed and therefore does not need explicit regularization as long as the training is properly early stopped. This understanding will potentially simplify the implementation of regularization networks and bypass the tedious cross-validation for selecting the suitable regularization parameter.

The organization for the rest of the paper is as follows. A review of the kernel adaline is included in section II. In section III, the well-posedness analysis of the kernel adaline is conducted. Then a numerical simulation is provided to support our theory in section III and finally we conclude in section IV.

II. KERNEL ADALINE

Suppose the goal is to approximate a continuous input-output mapping $f: U \rightarrow \mathbb{R}$ based on a set of training data $\{(u_1, d(1)), \ldots, (u_N, d(N))\}$ of $(u_i, d(i)) \in U \times Y$ where $Y$ is a compact subset of $R$ and $U$ is assumed as a compact subspace of $\mathbb{R}^L$. We use parentheses to denote the time-dependency of a scalar quantity and subscripts to denote the time-dependency of a vector or matrix quantity, e.g., if $d$ is a scalar then $d(i)$ refers to its value at time $i$. On the other hand, if $u$ is a vector, then $u_i$ denotes its value at time $i$.

A kernel [6] is a continuous, symmetric, positive-definite function $\kappa: U \times U \rightarrow \mathbb{R}$. The commonly used kernels include the Gaussian kernel (1) and the polynomial kernel (2) among many others:

$$\kappa(u_i, u_j) = \exp(-a \| u_i - u_j \|^2) \quad (1)$$

$$\kappa(u_i, u_j) = (u_i^T u_j + 1)^r \quad (2)$$

By the Mercer’s theorem [6], for any kernel there exists a mapping $\varphi$ such that

$$\kappa(u_i, u_j) = \varphi(u_i)^T \varphi(u_j), \quad \text{for } \forall u_i, u_j \in U \quad (3)$$

We utilize this theorem to transform the input $u_i$ into the feature space $F$ as $\varphi(u_i)$ (sometimes we use $\varphi_i$ for simplicity) and interpret (3) as the usual dot product. The linear span of all the transformed data forms a reproducing kernel Hilbert space (RKHS) [6]. Define the data matrices

$$\Phi_N = [\varphi(u_1), \varphi(u_2), \ldots, \varphi(u_N)]$$

$$d_N = [d(1), d(2), \ldots, d(N)]^T$$

Then the kernel adaline solves the following conventional least-squares problem:

$$\min_{\Omega} J_{\text{emp}}(\Omega) = \| d_N - \Phi_N^T \Omega \|^2 \quad (4)$$

The dimensionality of the feature space can be very high and even infinite when the Gaussian kernel is used. The ill-posedness of this problem arises from the fact that the number of the unknown variable (the dimensionality of the weight vector $\Omega$) is far more than the number of training data $N$.

The gradient of the cost function is

$$\nabla J_{\text{emp}} = -2\Phi_N(d_N - \Phi_N^T \Omega) \quad (5)$$

Therefore, the gradient descent is
\[ \Omega_i = \Omega_{i-1} - \eta \nabla J_{\text{reg}} \]
\[ = \Omega_{i-1} + \eta \Phi_N(d_N - \Phi_N^T \Omega_{i-1})/N \]  \hspace{1cm} (6)

where \( \Omega_i \) denotes the estimate (at iteration \( i \)) of the weight vector in the RKHS. \( \eta \) is the step size which absorbs the factor 2 and \( N \). The purpose of introducing \( N \) here will be clear later.

Claim 1: The weight is a linear combination of the transformed data, i.e.
\[ \Omega_i = \sum_{j=1}^{N} a_{i-1}(j)\varphi(u_j) \]  \hspace{1cm} (7)
where \( a_{i-1}(j) \) denotes the \( j \)-th coefficient of the network at iteration \( i-1 \).

Proof: The initial value \( \Omega_0 = 0 \), so \( a_0(j) = 0 \) for all \( j \). The claim is valid for \( i = 1 \).
Then we use mathematical induction. Suppose (7) is true. And
\[ \Omega_i = \sum_{j=1}^{N} a_{i-1}(j)\varphi(u_j) + \sum_{j=1}^{N} \eta e_i(j)\varphi(u_j) / N \]  \hspace{1cm} (9)
So the update equation for the weight is equivalent to the update for the linear coefficients
\[ a_i(j) = a_{i-1}(j) + \eta e_i(j) / N \]  \hspace{1cm} (10)
which is exactly the kernel adaline [1].

III. WELLPOSEDNESS ANALYSIS OF THE KERNEL ADALINE

A. Tikhonov Regularization

In least-squares (LS) the multiple linear regression model is hypothesized as
\[ d(i) = (\Omega)^T \varphi_i + \nu(i) \]  \hspace{1cm} (11)

From now on, \( \varphi_i \) is used for \( \varphi(u_i) \) unless the role of \( u_i \) is emphasized and \( \nu(i) \) is the modeling uncertainty. Introduce the correlation matrix \( R_p = \Phi_N \Phi_N^T / N \), the cross correlation vector \( p_\nu = \Phi_N d_N / N \) and the Gram matrix \( G_p = \Phi_N^T \Phi_N = [\kappa(u_i, u_j)]_{N \times N} \). Let the singular value decomposition (SVD) of \( \Phi_N \) is
\[ \Phi_N = P \begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix} Q^T \]  \hspace{1cm} (12)
where \( P, Q \) are orthogonal matrices and \( S = \text{diag}(s_1, s_2, \ldots, s_r) \) assuming the rank of \( \Phi_N \) is \( r \). \( s_n \) are the singular values of \( \Phi_N \) and assumed \( s_1 \geq s_2 \geq \ldots \geq s_r > 0 \).

Then we have
\[ R_p = P \begin{bmatrix} S^2 / N & 0 \\ 0 & 0 \end{bmatrix} P^T \]  \hspace{1cm} (13)
\[ G_p = Q \begin{bmatrix} S^2 & 0 \\ 0 & 0 \end{bmatrix} Q^T \]  \hspace{1cm} (14)
The well known pseudo-inverse solution to (4) is give by [7]
\[ \Omega_{\text{pl}} = (\Phi_N^T)^+ d_N = P \begin{bmatrix} S^{-1} & 0 \\ 0 & 0 \end{bmatrix} Q^T d_N \]  \hspace{1cm} (15)
where \((\cdot)^+\) denotes the general pseudo-inverse.

However, the LS problem can be ill-posed due to the nature of the problem, small data size, or severe noise. The Tikhonov regularization [3] is widely used to address this issue. A regularization term is introduced in the LS cost function which penalizes the solution norm
\[ \min_{\Omega} J_{\text{reg}}(\Omega) = \|d_N - \Phi_N^T \Omega\|^2 + \lambda \|\Omega\|^2. \]  \hspace{1cm} (16)
The solution is given by [3]
\[ \Omega_{\text{tr}} = (\Phi_N \Phi_N^T + \lambda I)^{-1} \Phi_N d_N. \]  \hspace{1cm} (17)

More insights can be obtained through the SVD,
\[ \Omega_{\text{tr}} = P \text{diag}(s_1 / (s_1^2 + \lambda), \ldots, s_r / (s_r^2 + \lambda)) Q^T d_N. \]  \hspace{1cm} (18)
Comparing (18) with the pseudo-inverse solution
\[ \Omega_{\text{pl}} = P \text{diag}(s_1^{-1}, \ldots, s_r^{-1}, 0, \ldots, 0) Q^T d_N, \]  \hspace{1cm} (19)
we see that the Tikhonov regularization modifies the singular values through the following regularization function (reg-function):
\[ H_{\text{reg}}(x) = x^2 / (x^2 + \lambda). \]  \hspace{1cm} (20)
Notice that \( H_{tr}(s_r)s_r^{-1} \to s_r^{-1} \) when \( s_r \) large and \( H_{tr}(s_r)s_r^{-1} \to 0 \) when \( s_r \) small. However, the pseudo-inverse solution becomes problematic when \( s_r \to 0 \). In this sense, the Tikhonov regularization smoothly filters out the singular components that are small (relative to \( \lambda \)). It is also clear that filtering out the small singular value is important for us to get a small norm solution. With this understanding, the so-called truncated pseudo-inverse regularization is nothing but the following hard cut-off reg-function

\[
H_{rc}(x) = \begin{cases} 
1 & \text{if } x > a \\
0 & \text{if } x \leq a 
\end{cases}
\]  

(21)

where \( a \) is the cut-off threshold. If \( s_m > a \geq s_{m+1} \), the solution becomes

\[
\Omega_{rc} = P \text{diag}(s_1^{-1}, \ldots, s_m^{-1}, 0, \ldots, 0)Q^Td_N
\]

(22)

This method is equivalent to applying principal component analysis (PCA) to the data and using the first \( m \) principal components to represent the original data. Under reasonable signal-noise-ratio (SNR), the small singular value components are purely associated with the noise. Discarding these spurious features can effectively prevent over-learning. A similar idea can be found in [8] and [9].

**B. Regularization through Gradient Descent**

In the section, we will show that the gradient descent iteration provides an inherent regularization similar to the Tikhonov regularization. First rewrite (6) as

\[
\Omega = \Omega_{l-1} + \eta \Phi_N(d_N - \Phi_N^T \Omega_{l-1})/N
\]

\caption{Claim 2: If \(|1 - \eta s_m^2 / N| < 1 \) for all non-zero \( s_m \), the kernel adaline converges to the pseudo-inverse solution (19).

Proof: If \( s_m \neq 0 \) and \(|1 - \eta s_m^2 / N| < 1 \), as \( i \to \infty \), (30) yields

\[
\lim_{i \to \infty} b_i(m) = Q_{(m)}^Td_N/s_m
\]

(31)

On the other hand, if \( s_m = 0 \), \( b_i(m) = b_i(m) = 0 \) provided that the initial value \( \Omega_0 = 0 \). To conclude, the kernel adaline converges to

\[
\lim_{i \to \infty} b_i = \text{diag}(s_1^{-1}, \ldots, s_m^{-1}, 0, \ldots, 0)Q^Td_N
\]

(32)

The proof is complete by using (27).

This result is not surprising. Notice that \( s_m^2 / N \) is the eigenvalue of the correlation matrix which is asymptotically
independent of \( N \), so introducing a factor of \( N \) make the convergence condition on \( \eta \) independent of \( N \).

However, the interesting observation is if proper early-stop is used in the training, the solution of the kernel adaline is self-regularized. For example, we start from \( \Omega_0 = 0 \) and the training stops after \( n \) steps. The solution is

\[
\Omega_{n,k} = \text{Pdiag}([1-(1-\eta s_1^2 / N)^n]s_1^{-1}, \\
\ldots,[1-(1-\eta s_N^2 / N)^n]s_N^{-1},0,...,0)Q^T d_N
\]

Therefore, the reg-function for kernel-adaline is

\[
H_{n,k}(x) = 1-(1-\eta x^2 / N)^n
\]

Claim 3: \( \lim_{x \to 0} H_{n,k}(x) x^{-1} = 0 \).

Proof:

\[
\lim_{x \to 0} \frac{1-(1-\eta x^2 / N)^n}{x} = \lim_{x \to 0} n(1-\eta x^2 / N)^{n-1}(2\eta x / N)
\]

by L’Hospital rule.

The significance of this result is that we can replace the tedious cross validation for the regularization parameter by the simple early-stop technique. A comparison of three regularization methods is illustrated in Figure 1. In the reg-function of Tikhonov regularization, the regularization parameter is chosen as 1. For the reg-function of the kernel adaline, \( \eta = 0.1 \), \( N = 500 \), and \( n = 600 \). For reg-function (21), \( a = 0.5 \).

### IV. Simulations

The simulation is based on the short-term prediction of the Mackey-Glass chaotic time series with parameter \( \tau = 30 \) and the sampling period 6s. We compare the performance of the kernel adaline and the regularization network with cross-validation. The time embedding is 8 for both methods. A segment of 1000 samples is used as the training data and another 100 as the test data. All the data is corrupted by Gaussian noise with zero mean and 0.01 variance. A Gaussian kernel with \( a = 1 \) is chosen for both the kernel adaline and the regularization network. In regularization network, every input point is used as the center and the training is done in batch mode. Figure 2 is the learning curves for the kernel adaline with a learning rate of 0.001. As we can see, the kernel adaline converges very quickly and the mean square error for the training data set keeps decreasing as training go on whereas the mean square error for the test data set increases after some point. This point is actually the early-stop point. In Figure 2, the minimal mean square error of the test data occurs at iteration 161 and the minimal value is 0.0205. The cross-validation result for the regularization network is present in Figure 3. 20 regularization parameters in the interval [0.01, 10] are tested. The best performance on the test set occurs when the regularization parameter equal 0.1179 and minimal mean square error of the test set is 0.0205, the same as the result from the kernel adaline.
V. CONCLUSIONS

In this paper, we theoretically prove that the kernel adaline trained with early stop possesses a self-regularization through gradient descent. This understanding not only has the theoretical significance for the kernel adaline but also provides an alternative way of obtaining regularized solutions besides the tedious cross-validation. The computation complexity of the cross-validation is the number of the regularization parameters searched multiplies the complexity of inverting the Gram matrix of the regularization network, which is $O(N^3)$. The complexity of the kernel adaline is the number of iterations multiplies the complexity of updating the coefficients which requires $O(N^2)$. As we see, when $N$ is very large, the kernel adaline scales better than the regularization network.

REFERENCES


