

# Discriminating Speech and Non-Speech with Regularized Least Squares

Ryan Rifkin

Honda Research Institute USA, Inc.  
One Cambridge Center Suite 401  
Cambridge MA 02142  
rifkin@honda-ri.com

Nima Mesgarani

Neural Systems Laboratory  
1103 A. V. Williams Building  
University of Maryland  
College Park, MD 20742  
mnima@isr.umd.edu

## Abstract

We consider the task of discriminating speech and non-speech in noisy environments. Previously, Mesgarani et. al [1] achieved state-of-the-art performance using a cortical representation of sound in conjunction with a feature reduction algorithm and a non-linear support vector machine classifier. In the present work, we show that we can achieve the same or better accuracy by using a linear regularized least squares classifier directly on the high-dimensional cortical representation; the new system is substantially simpler conceptually and computationally. We select the regularization constant automatically, yielding a parameter-free learning system. Intriguingly, we find that optimal classifiers for noisy data can be trained on clean data using heavy regularization.

**Index Terms:** speech detection, discriminative methods, regularization.

## 1. Introduction

The task of categorizing sound is important in many applications, including speech recognition, audio retrieval and robotics. Motivated by our interest in humanoid robots, we consider the task of discriminating speech from non-speech. The task is difficult for several reasons: the robot’s environment is frequently noisy and reverberant, the sound source may be far from the robot, and the robot contains many motors and fans.

A key decision in any auditory application is the representation of sound. Recently, there has been substantial activity and progress in understanding how sound is represented in mammalian cortex. We consider a model by Shamma and colleagues, that is inspired by neurophysiological knowledge of the early and central stages of the mammalian auditory system [2]. In this model, each short (8 ms) time-slice is represented as a three-dimensional tensor in frequency, rate and scale space. The model generally uses 128 frequencies, 12 rates and 5 scales, so each time-slice is 7,680 dimensional.

This model was used by Mesgarani et al. to discriminate speech from non-speech [1]. In their application, the discrimination was done in two stages: dimensionality reduction was performed using a higher-order singular value decomposition (HOSVD) [3], an analog of the SVD that respects the tensor nature of the cortical representation, and classification was done with a Gaussian support vector machine. The resulting system compared favorably to other well-regarded systems [4, 5]. The system had several tunable parameters: the number of components to keep was determined separately for the scale, rate, and frequency subspaces,

and the bandwidth of the SVM Gaussian kernel was chosen.

In the present work, we show that optimal performance can be obtained by using a regularized linear classifier *directly* in the cortical space. We achieve the same accuracy as HOSVD/SVM, but our new method is conceptually and computationally much simpler, and has no tunable parameters — there is a single regularization parameter  $\lambda$ , but this parameter is found automatically. Because the algorithm is so simple, we are able to test a very large range of regularization parameters, and make an interesting observation about clean and noisy sound.

In Section 2, we briefly describe the cortical representation. Section 3 presents the regularized least squares algorithm, and shows how to effectively determine the regularization constant  $\lambda$  using leave-one-out cross-validation; we believe these ideas are not widely disseminated in the acoustical machine learning community. Section 4 presents experimental validation of the RLS approach. Section 5 compares our algorithm to the HOSVD/SVM from a computational standpoint. We discuss implications in Section 6.

## 2. Cortical Representation of Sound

Spectrotemporal modulation features are extracted from an auditory model inspired by psychoacoustical and neurophysiological findings in the early and central stages of the auditory pathway. The early stage converts the sound waveform into an *auditory spectrogram* — a time-frequency distribution along a tonotopic (logarithmic frequency) axis. The second (cortical) stage performs a two dimensional wavelet transform of the auditory spectrogram, providing an estimate of its spectral and temporal modulation content. It is computationally implemented by a bank of two-dimensional (spectrotemporal) filters that are selective to different spectrotemporal modulation parameters ranging from slow to fast *rates* temporally (in Hertz), and from narrow to broad *scales* spectrally (in Cycles/Octave). The spectrotemporal impulse responses (or “receptive fields”) of these filters are centered at different frequencies along the tonotopic axis. The basic mathematical formulation of the model can be summarized as:

$$\begin{aligned} y_{cochlea}(t, f) &= s(t) * h_{cochlea}(t, f) \\ y_{an}(t, f) &= g_{cochlea}(\partial_t y_{cochlea}(t, f)) * \mu_{haircell} \\ y(t, f) &= \max(\partial_f y_{an}(t, f), 0) * \mu_{midbrain} \\ r_{\pm}(t, f; \omega, \Omega) &= \\ & y(t, f) *_{tf} [STRF_{\pm}(t, f; \omega, \Omega)] \end{aligned}$$

where  $s(t)$  is the sound,  $y_{cochlea}(t, f)$  is the cochlear filter output,  $y_{an}(t, f)$  are auditory nerve patterns,  $y(t, f)$  is the auditory spectrogram and  $r_{\pm}(t, f; \omega, \Omega)$  is the cortical representation. The sign of  $r$  specifies the direction of spectrotemporal modulation:  $-$  for downward,  $+$  for upward patterns. The modulation representation of sound is a 4-dimensional function of time ( $t$ ), frequency ( $f$ ), rate ( $\omega$ ) and scale ( $\Omega$ ). One can think of each point in the spectrogram as having a 2-dimensional rate-scale representation,  $r(t_c, f_c, \omega, \Omega)$  which indicates the modulation strength at all  $\omega$ 's and  $\Omega$ 's for that channel and instant. If we average the time dimension on a given duration of sound, we obtain the average rate-scale-frequency representation in the given time window.

The cortical representation is summarized in Figure 1. For more details of the representation see [2].

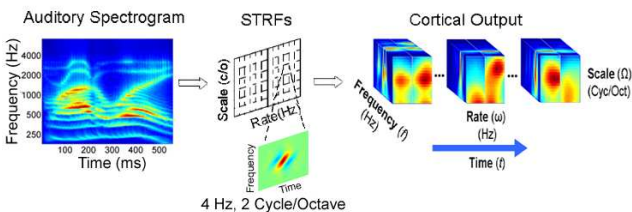


Figure 1: The cortical representation of sound.

### 3. Regularized Least Squares

Given data points  $(x_1, y_1), \dots, (x_n, y_n)$ , where  $\mathbf{x}_i \in \mathcal{R}^d$  and  $y_i \in \{0, 1\}$ , and a kernel function  $k : x \times x \rightarrow \mathcal{R}$ , the regularized least squares (RLS) algorithm solves the optimization problem: [6, 7]:

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

$\lambda$  is a regularization constant controlling the tradeoff between fitting the training set accurately and finding a function with small norm, and  $K$  is the kernel matrix defined by  $K_{ij} = k(x_i, x_j)$ . Linear RLS arises when  $k(x_i, x_j) = x_i^t \cdot x_j$  is used as the kernel function. In this case, the solution  $c$  to the optimization problem defines a classifying hyperplane  $w$  via

$$w = \sum_{i=1}^n c_i x_i.$$

Differentiation and simple algebra shows [7] that we can find the coefficients  $c$  by solving the linear system

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

The constant  $\lambda$  must be determined somehow. Defining the  $i$ th leave-one-out (LOO) error to be the error made on the  $i$ th training point when it is removed from the data set, a classifier is built on the remaining  $n-1$  data points and tested on  $x_i$ , the RLS algorithm has the remarkable property [8, 7] that the vector of LOO errors is given by

$$\frac{c}{\text{diag}(K + \lambda I)^{-1}}.$$

One of the most computationally useful properties of RLS is that finding both  $c$  and the LOO vector for a large number of  $\lambda$ s requires almost no additional work compared to that required for a

single  $\lambda$ , if we utilize the eigendecomposition of the kernel matrix  $K$ :

$$K = V\Lambda V^T,$$

where  $\Lambda$  is diagonal and  $VV^T = I$ . Once we have obtained this eigendecomposition (in  $O(n^3)$  time), we can find  $c$ , the LOO errors and  $w$  for a given value of  $\lambda$  in  $O(n^2)$  time by exploiting the relation

$$\begin{aligned} (K + \lambda I)^{-1} &= (V\Lambda V^T + \lambda I)^{-1} \\ &= (V(\Lambda + \lambda I)V^T)^{-1} \\ &= V(\Lambda + \lambda I)^{-1}V^T. \end{aligned}$$

The diagonal matrix  $(\Lambda + \lambda I)$  is trivially invertible in  $O(n)$  time, and it is not difficult to show that we can compute  $(K + \lambda I)^{-1}y$  or the diagonal of  $(K + \lambda I)^{-1}$  in  $O(n^2)$  time (computing all of  $(K + \lambda I)^{-1}$  explicitly would require  $O(n^3)$  time, but this is never needed.)

The RLS algorithm makes it feasible to examine the behavior of the system over a wide range of values of the regularization parameter  $\lambda$ . In this work, we make use of 400 different values of  $\lambda$ , logarithmically spaced between  $10^{-12}$  and  $10^{12}$ . In practice, using all 400  $\lambda$ 's increases the running time by less than a factor of three compared to using a single  $\lambda$ .

There exist specialized methods to perform linear RLS in  $O(nd + d^2)$  memory and  $O(nd^2)$  time, and these methods are invaluable when  $n \gg d$ . Here,  $d > n$ , so it is more effective to use the standard nonlinear kernel machinery described here.

### 4. Experiments

We study the performance of RLS with an experiment. We mostly follow the experimental setup of [1]. In that work, the authors compared their system, which used a higher-order singular value decomposition (HOSVD) for dimensionality reduction followed by a support vector machine with a Gaussian kernel, to other systems, and found that the HOSVD/SVM system achieved state-of-the-art performance compared to well-regarded multifeature [4] and voicing-energy [5] based methods.

The data set consists of speech and non-speech examples; the non-speech examples are drawn from several databases. For each sample, the *average* cortical representation over one second from the center of the sound file is used to represent the sound as a point in a 7,680 dimensional space. We considered both clean data, and data mixed with white noise at varying SNRs. For further details of the experimental setup and the databases, see [1].

We use the same training and testing sets as [1]. However, the sound files in [1] were recorded at differing frequencies, while we downsampled all sounds a priori to 8 kHz (the lowest recording frequency used). Additionally, in [1], the same piece of white noise was mixed repeatedly with different sound files. We use a unique white noise sample for each sound sample (we use the same white noise sample for each sound sample at differing SNRs). The numbers here are therefore not directly comparable to those in [1] (in informal experiments, reusing the same white noise across all sounds yielded approximately 50% fewer errors).

In addition to the RLS algorithm described here, we reran the algorithms from [1] (the HOSVD/SVM, as well as [1]'s implementations of the multifeature and voicing-energy methods) on the revised data. Table 1 summarizes the results. All the classifiers used produce real-valued predictions, and by varying a threshold, we

Algorithm	SNR (dB)				
	-4	-8	-12	-16	-20
Multifeature, Noisy Trained	.13	.25	.40	.46	.48
Voicing-Energy, Noisy Trained	.12	.16	.26	.47	.53
HOSVD/SVM, Clean Trained	.04	.10	.30	.49	.53
HOSVD/SVM, Noisy Trained	.03	.03	.08	.17	.33
RLS, Clean Trained, LOO- $\lambda$	.02	.05	.16	.38	.48
RLS, Clean Trained, Best- $\lambda$	.03	.05	.08	.16	.37
RLS, Noisy Trained, LOO- $\lambda$	.02	.03	.08	.19	.34
RLS, Noisy Trained, Best- $\lambda$	.02	.04	.08	.15	.35

Table 1: Summary of Results. All numbers are equal error rates. “Best  $\lambda$ ” is for comparative purposes; as it assumes an ability to select the right  $\lambda$  for each task.

can produce an ROC (or DET) curve [9]. For easy comparability of a range of algorithms across a range of SNRs, we report *equal error rates* (EERs), the point on the ROC (or DET) curve where the rate of errors on each class is equal.<sup>1</sup> We note that the testing set contains only 651 points, and small jitters in the predictions can have moderately large effects on the EERs, so we cannot put too much stock in small differences in EERs. That said, we make the following observations:

- The multifeature and voicing energy systems degrade very badly in the noisier conditions, relative to the cortical based systems, even when they were trained and tested on noisy data.
- When trained on noisy data, the HOSVD/SVM and RLS systems both yield very good (and similar) performance.
- The noisy RLS LOO- $\lambda$  and noisy RLS best- $\lambda$  (best  $\lambda$  is the performance we would achieve if we could magically pick the optimal  $\lambda$  for each SNR) are quite close over most SNRs tested, indicating that the LOO procedure for picking  $\lambda$  is working well.
- The LOO- $\lambda$  classifier trained on clean data is stronger than the HOSVD/SVM system trained on clean data, but not nearly as strong as the noisy trained systems.
- On the other hand, the best- $\lambda$  classifier trained on the clean data performs as well as the noisy trained systems.

This last point is quite intriguing. Figure 2 shows the EER of the clean trained classifiers across the range of  $\lambda$ s, for varying SNRs. As the SNR decreases, the appropriate value of  $\lambda$  increases, and at very low SNRs, the best  $\lambda$  is several orders of magnitude (note the logarithmic  $x$ -axis) larger than the best  $\lambda$  for clean data. Our ability to easily sweep  $\lambda$  as described in Section 3 is a key enabling technique for making observations of this sort.

## 5. Algorithmic Comparison of HOSVD/SVM and RLS

The RLS system is substantially faster than the HOSVD/SVM system at both training and testing time. On a 3.4 GHz Pentium IV system, the HOSVD takes 37 seconds (assuming fixed HOSVD parameters) and the SVM takes 157 seconds (including time to ten-fold cross-validate the Gaussian kernel parameter). On the other

<sup>1</sup>In informal examination, the ROC curves rarely crossed (if one classifier gave a substantially lower EER than another, it had better performance at all operating points), so we feel justified in using EERs for comparison.

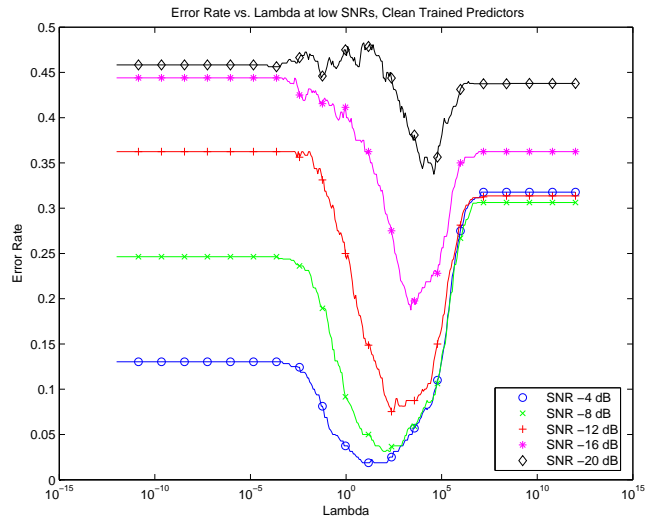


Figure 2: EER of clean trained predictors across a range of  $\lambda$ 's, at different SNRs.

hand, on a slower 3.0 GHz Pentium IV system, the RLS system requires 31 seconds to form the eigendecomposition of the kernel matrix, and about .15 seconds to train and compute the leave-one-out error for a single  $\lambda$ : we compute 400 classifiers for a wide range of  $\lambda$  in 92 seconds.

At testing time, the HOSVD projects the 128 by 5 by 12 dimensional tensor down to a 140 dimensional vector, and the distance between this vector and the 81 (140 dimensional) support vectors generated by the SVM algorithm. In contrast, the RLS method simply converts the input tensor to a vector and takes a dot product, and is approximately 10 times faster. For larger datasets or more complicated tasks, the advantage of the linear RLS will be even greater — the number of support vectors for the nonlinear SVM will grow linearly in the number of data points (assuming we cannot totally separate the data, we will have a fraction of the data corresponding to the Bayes error rate as support vectors), while the linear RLS will always output a single hyperplane.

The HOSVD/SVM system requires choosing the number of basis vectors in three different dimensions for the HOSVD (this was done once for [1], and we reused those values in experiments for this paper), a regularization parameter  $C$  for the SVM (this was always set to 1), and a Gaussian width  $\sigma$  for the SVM kernel (this is optimized by 10-fold cross-validation for each noise condition). On the other hand, the RLS system requires only a single regularization parameter  $\lambda$ , and this is chosen automatically by LOO cross-validation as described in Section 3, so the system is essentially parameter-free.

## 6. Discussion

We have shown how a simple linear classifier can be used to obtain state-of-the-art performance on a speech detection task. We achieve the same accuracy as the HOSVD/SVM system, but our system is much simpler both conceptually and computationally.

We showed that we can use clean training data to build accurate classifiers for very noisy sound signals. We find this quite intriguing, compared to the situation in speech recognition, where it is frequently necessary to directly train on noisy data to achieve

noise robustness. Compared to the optimal classifiers for clean sounds, the optimal clean-trained classifiers for noisy sounds had much larger values of  $\lambda$ , and were therefore *smoother* functions. Indeed, when we visually inspected the clean-trained classifiers with large  $\lambda$  (another advantage of using linear models is that we can plot them in exactly the same way we plot the cortical representations themselves), we find that they look like “average” speech signals, and are representing the “coarse structure” of the cortical representation of speech. On the other hand, the classifiers with smaller  $\lambda$  (which are optimal for clean speech) do not have an immediately visible interpretation, and are classifying the distinction between speech and non-speech at a much finer scale.

We should not be too surprised that a linear classifier performs as well as a nonlinear classifier. All kernel learning algorithms, of which SVM and RLS are examples, construct functions in a linear *feature* space. When nonlinear kernels such as the Gaussian are used, the feature space is implicit, and high or infinite dimensional. However, the cortical representation itself is *already* an explicit nonlinear “lifting” of the auditory spectrogram into a high-dimensional space: a linear function on the cortical representation is a nonlinear function on the spectrogram. More generally, we believe that explicitly choosing a nonlinear (but computationally tractable) feature space, and explicitly working in that feature space, is frequently an excellent way to achieve state-of-the-art tradeoffs between accuracy and computational demands.

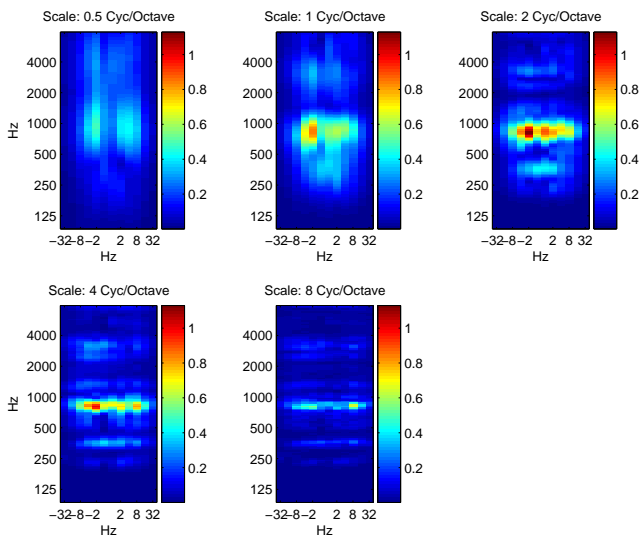


Figure 3: A woman says “...encourage ’em to ex...”: borderline speech.

Although these results are promising, there are many directions to explore. Figure 4 shows an example that is consistently classified as speech by our systems. To human ears, the sound is clearly not speech. Nevertheless, the visual representation seems indistinguishable from some speech signals (compare Figure 3 and 4). It seems plausible that by averaging the cortical representation over a full second, we have thrown away valuable *dynamic* information; future work should attempt to incorporate it, although this will lead to much more computationally challenging problems.

The detector seems good enough to be incorporated onto a humanoid robot, which usually operates at an SNR of around 10 dB (although SNR is itself a vague notion for nonstationary sounds,

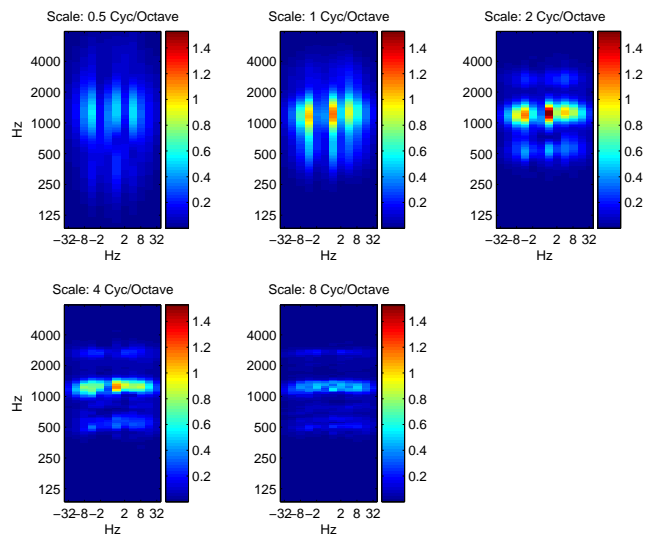


Figure 4: A dog barks: borderline non-speech.

and the noise the robot is exposed to is certainly *not* white). The current implementation of the cortical model is written in Matlab, and is several times slower than realtime, but we believe a realtime implementation is possible. We intend to incorporate the system onto our robot in the near future.

## 7. References

- [1] Mesgarani, Slaney, and Shamma, “Discrimination of speech from non-speech based on multiscale spectro-temporal modulations,” *IEEE Transactions on Speech and Audio Processing*, pp. 920–930, 2006.
- [2] Chi, Ru, and Shamma, “Multiresolution spectrotemporal analysis of complex sounds,” *Journal of the Acoustical Society of America*, pp. 887–906, 2005.
- [3] De Lathauwer, De Moor, and Vandewalle, “A multilinear singular value decomposition,” *SIAM Journal of Matrix Analysis and Applications*, pp. 1253–1278, 2000.
- [4] Scheirer and Slaney, “Construction and evaluation of a robust multifeature speech/music discriminator,” in *ICASSP*, 1997.
- [5] Kingsbury, Saon, Mangu, Padmanabhan, and Sarikaya, “Robust speech recognition in noisy environments: the 2001 IBM SPINE evaluation system,” in *ICASSP*, 2002.
- [6] Wahba, *Spline Models for Observational Data*, Society for Industrial & Applied Mathematics, 1990.
- [7] Rifkin, *Everything Old Is New Again: A Fresh Look at Historical Approaches to Machine Learning*, Ph.D. thesis, Massachusetts Institute of Technology, 2002.
- [8] Green and Silverman, *Nonparametric Regression and Generalized Linear Models*, Chapman & Hall, 1994.
- [9] Martin, Doddington, Kamm, Ordowski, and Przybocki, “The DET curve in assessment of detection task performance,” in *Proc. Eurospeech ’97*, Rhodes, Greece, 1997, pp. 1895–1898.