A Support Vector Approach to the Acoustic-to-Articulatory Mapping

Asterios Toutios and Konstantinos Margaritis

Parallel and Distributed Processing Laboratory, Department of Applied Informatics,
University of Macedonia, Thessaloniki, Greece

{toutios, kmarg}@uom.gr

Abstract

We report work on mapping the acoustic speech signal, parametrized using Mel Frequency Cepstral Analysis, onto electromagnetic articulography trajectories from the MOCHA database. We employ the machine learning technique of Support Vector Regression, contrasting previous works that applied Neural Networks to the same task. Our results are comparable to those older attempts, even though, due to training time considerations, we use a much smaller training set, derived by means of clustering the acoustic data.

1. Introduction

The acoustic-to-articulatory mapping [1, 2], also termed acoustic-to-articulatory inversion of speech, is a special speech processing related problem that has attracted the attention of several researchers for many years now. It refers to the estimation of articulatory (speech production related) information using solely the speech signal as an input source. A succesful solution could find numerous applications, such as helping individuals with speech and hearing disorders by providing visual feedback, very low bit-rate speech coding and the possibility of improved automatic speech recognition.

In the past, the articulatory features used in such a context were mostly inferred by the corresponding acoustic data using vocal-tract models, synthesis models, or linguisting rules. But recent technologies have made it possible to record actual articulator movements in parallel with speech acoustics in a minimally invasive way. This "real" human data is arguably preferable to older techniques, where additional complications may be imposed by intrinsic flaws of the models themselves.

One of the forementioned technologies is the Electromagnetic Misdagittal Articulography (EMMA) or Electromagnetic Articulography (EMA). Roughly speaking, for the acquisition of EMA data, sensor coils are attached to the human subject, on specific places on the lips, the teeth, the jaw, and the soft palate (velum). Then the human subject wears a special helmet which produces an alternating magnetic field that records the position of the coils at end points of small fixed-size time intervals. The outcomes are trajectories that illustrate the movement of the coils. Usually, there are two trajectories for each coil, one for the movement in the front-back direction of the head, and one for the movement in the top-bottom direction.

In this paper we follow Richmond's work [1], who proposed a quite succesful mapping of the speech signal onto EMA data, using Neural Networks (Multilayer Perceprtons and Mixture Density Networks). We study an alternative –Machine Learning– approach using Support Vector Regression, a more recent and very promising method. We use the same dataset as Richmond (though we finally arrive at a significantly smaller training set), namely the fsew0 speaker data from the MOCHA dabase.

2. The MOCHA Database

The MOCHA (Multi-Channel Articulatory) [3] database is evolving in a purpose built studio at the Edinburgh Speech Production Facility at Queen Margaret University College.

During speech, four data streams are recorded concurrently straight to a computer: the acoustic waveform, sampled at 16kHz with 16 bit precision, together with laryngograph, electropalatograph and electromagnetic articulograph data. The articulatory channels include EMA sensors directly attached to the upper and lower lips, lower incisor (jaw), tongue tip (5-10mm from the tip), tongue blade (approximately 2-3cm posterior to the tongue tip sensor), tongue dorsum (approximately 2-3cm posterior from the tongue blade sensor), and soft palate. Two channels for every sensor are recorded at 500Hz: the positioning on the x-axis (front-back direction) and on the y-axis (top-bottom direction).

The speakers are recorded reading a set of 460 British TIMIT sentences. These short sentences are designed to provide phonetically diverse material and capture with good coverage the connected speech processes in English. All waveforms are labelled at the phonemic level.

The final release of the MOCHA database will feature up to 40 speakers with a variety of regional accents. At the time of writing this paper two speakers are available, one male with a Northern English accent and one female with a Southern English accent. For the experiments described herein, the acoustic waveform and EMA data, as well as the phonemic labels for the female speaker, fsew0, are used.

3. Support Vector Regression

The ϵ-SVR algorithm [4] is a generalization of the better known Support Vector Classification algorithm [5] to the regression case. Given n training vectors $\mathbf{x_i}$ and a vector $y \in R^n$ such that $y_i \in R$, we want to find an estimate for the function $y = f(\mathbf{x})$ which is optimal from a Structural Risk Minimization viewpoint. According to ϵ-SVR, this estimate is:

$$f(\mathbf{x}) = \sum_{i=1}^{n} (a_i^* - a_i) k(\mathbf{x_i}, \mathbf{x}) + b, \tag{1}$$

where b is a bias term and $k(\mathbf{x_i x_j})$ is a special function called the *kernel*. The coefficients a_i and a_i^* are the solution of the quadratic problem

$$\text{maximize}$$
$$W(\mathbf{a},\mathbf{a}^*) = -\epsilon \sum_{i=1}^{n}(a_i^* + a_i) + \sum_{i=1}^{n}(a_i^* - a_i)y_i$$
$$-\frac{1}{2}\sum_{i,j=1}^{n}(a_i^* - a_i)(a_j^* - a_j)k(\mathbf{x_i}\mathbf{x_j}) \tag{2}$$
$$\text{subject to}$$
$$0 \le a_i, a_i^* \le C, i = 1, \ldots, n,$$
$$\sum_{i=1}^{n}(a_i^* - a_i) = 0,$$

where $C > 0$ and $\epsilon \geq 0$ are parameters chosen by the user. The "penalty parameter" C may be as high as infinity, while usual values for ϵ are 0.1 or 0.01.

The kernel function serves to convert the data into a higher-dimensional space in order to account for non-linearities in the estimate function. A commonly used kernel is the Radial Basis Function (RBF) kernel:

$$k(\mathbf{x},\mathbf{y}) = \exp(-\gamma \parallel \mathbf{x} - \mathbf{y} \parallel^2), \tag{3}$$

where the γ parameter is selected by the user.

4. Data Processing

The MOCHA database includes 460 utterances of the fsew0 speaker. In order to render these data into input-output pairs suitable for function estimation, we process them as follows.

First, based on the label files we omit silent parts from the beginning and end of the utterances. During silent stretches the articulators can possibly take any configuration, something that could pose serious difficulties to our task.

Next, we perform a standard Mel Frequency Spectral Analysis [6] on the acoustic signal with the VOICEBOX Toolkit [7], using a window of 16ms (256 points) with a shift of 5ms. We use 30 filterbanks and calculate the first 13 Mel Frequency Cepstral Coefficients. Then, we normalize them in order have zero mean and unity standard deviation.

In order to account for the dynamic properties of the speech signal and cope with the temporal extent of our problem, we just use the commonplace in the speech processing field *spatial metaphor for time*. That is, we construct input vectors spanning over a large number of acoustic frames. Based on some previous small-scale experiments of ours, we construct input vectors consisting of the MFCCs of 17 frames: the frame in question, plus the 8 previous ones, plus the 8 next ones.

The steps taken to process the EMA data are similar to those described by Richmond. First, the EMA data are resampled to match the frameshift of the acoustic coefficients (5ms). At the same time, they are smoothed, using a moving average window of 40ms so that recording noise is eliminated (after all, it is known that EMA trajectories vary relatively slowly with time).

The mean values of the EMA trajectories calculated for every utterance vary considerably during the recording process. There are two kinds of variation: rapid changes, due to the phonemic content of each utterance, and slowly moving trends, mainly due to the fact that the subject's articulation adapts in certain ways during the recording session. It is beneficial to remove from the EMA data the second type of variation, while keeping the first. Thus, we calculate the means, low-pass filter them, and subtract those filtered means from the EMA data. (See Figure 1 for an explanation).

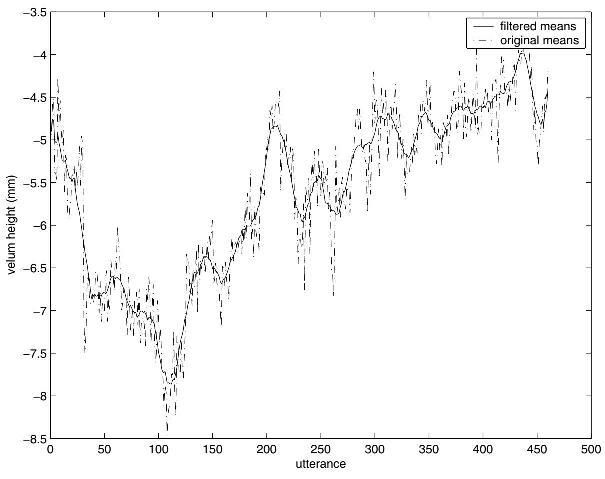

Figure 1: Mean values of the "velum height" (v_y) channel across the utterances in the recording session. The dashed line shows the real means and the solid line their filtered version which is actually used for normalization.

Thus, we end up with training examples, each consisting of a 221-dimensional (17×13) real-valued vector as input and a 14-dimensional real-valued vector as output. We split our data into two big halves: the even-numbered utterances constitute what we call an "extended training set", and the odd-numbered ones an "extended test set". Each one contains more than 100.000 examples.

5. Training Set Selection

Support Vector Regression training is a relatively slow process. Training time increases with the cube of the amount of training data. Using our whole "extended training set" would be far too much time consuming. Therefore, we would like a reduced training set.

On the other hand, we need this reduced training set to be somehow "representative" of the whole corpus. If we were to take, say, 10 random utterances and use for training we would possibly lose important information.

In order to choose representative examples we employ clustering in our input–MFCC space. We use a rather simple clustering algorithm, namely K-means [8]. We ask for 5000 individual clusters and the algorithm returns 4587. Then, we find the examples with the minimum distance from the centers of these clusters. These examples constitute the training set which we actually use.

6. SVR Training and Results

The ϵ-SVR algorithm, as described, works for only one output. It does not work "as is" for multiple outputs. Thus, we have to split our problem into 14 distinct (prefferably independent or at least uncorrelated) function estimation problems, considering each time a different trajectory as output.

Just before SVR training we perform some further preprocessing steps on our output data, mainly to remove lower-order statistical properties. The first step is to *center* the data so that the mean value of every channel is zero. This may seem redundant, considering the mean normalization previously described, but it is desirable that the means *across the training set* are zero.

The second step is to *whiten* the data [8] so that its covariance matrix become an identity matrix. This is accomplished by

$$Y_{whitened} = \sqrt{E\{YY^T\}}^{\,-1} Y \tag{4}$$

where Y is the data table (where every example is row) and the square root of a matrix is defined so that $M = \sqrt{M}^T \sqrt{M}$.

The third step is to scale the data by four times their standard deviation, so that they roughly lie in the interval $(-1, 1)$, something crucial for SVR training.

In order to train our 14 function estimators, we use the RBF kernel with $\gamma = 0.0045$ and select $C = 1$, $\epsilon = 0.1$, based on heuristics found in [9], employing the LibSVM software [10] for our experiments. We, finally, virtually "combine" the 14 estimators into one "system".

After testing our estimators we invert the processes of scaling, whitening and centering. Finally, we smooth the output trajectories using again a moving average window of 40ms

For evaluating the performance of our system we use two measures. The first one is the RMS error which is an indication of the overall "distance" between two trajectories. It is calculated as:

$$E_{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(o_i - y_i)^2} \tag{5}$$

where N is the number of input-output vector pairs, in the test set, o_i is the estimated value for the articulator channel output, and y_i is the real value.

The second measure is the correlation score, which is an indication of similarity of shape and synchrony of two trajectories. It is calculated by dividing their covariance by the product of their variances:

$$r = \frac{\sum_i (o_i - \bar{o})(y_i - \bar{y})}{\sqrt{\sum_i (o_i - \bar{o})^2 \sum_i (y_i - \bar{y})^2}} \tag{6}$$

where $\bar{o}$ and $\bar{y}$ are the mean channel value for the estimated and real articulator position respectively. .

For testing our system we use 10 utterances spanning our whole "extended test set". Our overall results for this test set are presented in Table 1. Figure 2 shows the real and estimated trajectories for a single utterance. For the results in Table 2 we split the examples of our test set according to which phoneme they correspond to, and present the performance of our system of estimators on some of these small–phoneme test sets.

7. Conclusion

We applied Support Vector Regression to the task of mapping the acoustic speech signal onto EMA trajectories. Our results were comparable to those found in the literature, even though we used for training a rather small subset of the data available to us. This subset was selected by clustering in the input space.

Analysis of the results by phoneme, reveals that the system of function estimators we end up with behaves quite differently for different phonemes. Some EMA channels are estimated better for specific phonemes, while others are not. It may be the case that some phonemes provide for a stronger "link" between the Mel Frequency Cepstral Coefficients and particular EMA trajectories.

EMA Channel	**RMS Error (mm)**	**Correlation**
lower incisor x	1.003	0.511
lower incisor y	1.076	0.826
upper lip x	0.979	0.583
upper lip y	1.278	0.597
lower lip x	1.358	0.540
lower lip y	2.309	0.828
tongue tip x	2.464	0.812
tongue tip y	2.589	0.828
tongue body x	2.323	0.796
tongue body y	2.230	0.816
tongue dorsum x	2.184	0.760
tongue dorsum y	2.588	0.668
velum x	0.431	0.723
velum y	0.376	0.753

Table 1: Overall performance of the system of estimators.

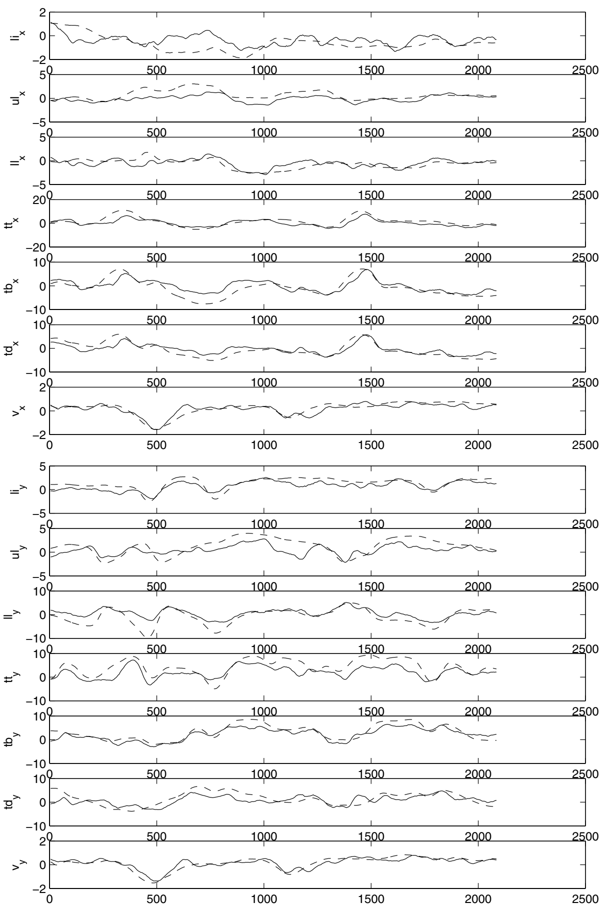

Figure 2: Real (dashed lines) and estimated (solid lines) articulatory trajectories of fsew0 uttering the phrase "Clear pronunciation is appreciated.", plotted against time in milliseconds. From top to bottom: lower incisor (jaw), upper lip, lower lip, tongue tip, tongue dorsum, tongue blade and velum. First are shown the projections of the articulator's movement on the x axis, followed by those on the y axis.

	li$_x$	li$_y$	ul$_x$	ul$_y$	ll$_x$	ll$_y$	tt$_x$	tt$_y$	tb$_x$	tb$_y$	td$_x$	td$_y$	v$_x$	v$_y$
h	0.416	0.351	0.441	0.341	-0.031	0.926	0.619	0.473	0.863	0.714	0.644	0.219	0.161	0.168
ii	0.446	0.822	0.537	0.821	0.725	0.845	0.787	0.752	0.782	0.644	0.430	0.472	0.204	0.365
i	0.513	0.796	0.193	0.522	0.448	0.817	0.541	0.630	0.643	0.694	0.653	0.655	0.628	0.690
l	0.641	0.775	0.540	0.072	0.683	0.864	0.700	0.651	0.547	0.566	0.511	0.470	-0.019	0.324
r	-0.146	0.587	0.708	0.202	0.047	0.766	0.576	0.403	0.078	0.543	0.068	0.313	0.494	0.638
p	0.190	0.307	0.371	0.534	0.177	0.831	0.790	0.673	0.803	0.820	0.745	0.748	0.550	0.450
oo	0.530	0.260	-0.335	0.041	0.261	0.014	0.910	0.951	0.509	0.550	0.223	0.398	-0.066	0.497
s	-0.211	0.392	0.368	0.444	0.195	0.361	0.358	0.279	0.011	0.567	0.217	0.729	0.381	0.478
k	0.252	0.683	0.500	0.330	0.605	0.581	0.764	0.629	0.710	0.581	0.664	0.144	0.672	0.592
m	0.530	0.479	0.486	0.029	0.589	0.708	0.767	0.679	0.848	0.919	0.862	0.905	0.459	0.317
f	-0.502	0.604	0.352	0.257	-0.315	0.492	0.690	0.667	0.540	0.538	0.573	0.598	0.848	0.584
d	0.445	0.265	0.818	0.013	0.588	0.730	0.476	0.154	0.010	0.204	0.034	0.712	0.909	0.905
g	0.297	0.411	0.773	0.204	0.654	0.918	0.681	0.954	0.739	0.617	0.733	0.047	0.083	-0.043
z	0.016	0.280	0.322	0.094	0.151	0.448	0.191	0.401	0.368	0.642	0.378	0.468	0.200	0.393
e	0.596	0.430	0.582	0.706	0.744	0.725	0.846	0.699	0.823	0.925	0.820	0.834	0.312	-0.038
sh	0.130	0.654	0.513	0.530	0.451	0.748	0.509	0.033	0.284	0.780	0.287	0.586	0.513	0.418
t	0.374	0.758	0.480	0.461	0.268	0.690	0.421	0.609	0.542	0.781	0.531	0.696	0.477	0.591
y	-0.013	0.076	-0.426	0.038	-0.212	0.550	0.454	0.420	-0.096	-0.384	0.097	-0.257	0.810	0.873
o	0.769	0.827	0.679	0.716	0.793	0.811	0.846	0.937	0.824	0.926	0.711	0.640	0.491	0.729
n	-0.058	0.573	0.339	-0.121	0.197	0.729	0.782	0.473	0.699	0.599	0.606	0.279	0.713	0.675
a	0.192	0.657	0.791	0.629	0.797	0.706	0.892	0.527	0.580	0.857	0.545	0.899	0.088	0.516
v	0.787	0.786	0.403	0.594	0.317	0.201	0.836	0.909	0.955	0.464	0.943	0.436	0.914	0.314
uh	0.808	0.882	0.798	0.378	0.092	0.715	0.771	0.952	0.816	0.695	0.732	-0.112	0.775	0.676
ou	0.776	0.933	0.034	0.798	0.548	0.439	0.225	0.715	-0.245	0.356	-0.385	0.910	0.922	0.935
iy	0.586	0.631	0.548	0.229	0.052	0.919	0.902	0.836	0.840	0.832	0.644	0.856	0.869	0.977
ei	0.581	0.356	0.775	-0.061	0.955	-0.188	0.617	0.640	0.405	0.965	0.396	0.772	0.696	0.891
th	-0.063	0.978	0.714	0.873	-0.529	0.534	0.097	0.159	0.423	0.844	0.059	0.909	0.579	0.700
b	0.748	0.850	0.314	0.222	0.316	0.411	0.880	0.714	0.917	0.867	0.956	0.602	0.624	0.825
@	0.178	0.717	0.540	0.646	0.212	0.709	0.911	0.841	0.883	0.776	0.824	0.661	0.580	0.774

Table 2: Correlation scores between real and estimated articulatory trajectories for specific (out of 44) phonemes in the test set.

Regarding future work directions, it is probable that using more training data would improve performance. Training time is always an issue, but recent findings in the machine learning field, such as Cross-Training [11], seem quite promising in the direction of speeding up things.

Finally, the temporal dimension of the problem might be taken more rigorously into account. There are promising proposals in that area as well, such as the HMM–SVM method [12].

8. Aknowledgement

The first author is supported by the EPEAEK 2–Heracletus Dissertation Scholarship, code 237-88724-2.

9. References

[1] Korin Richmond. *Estimating Articulatory Parameters from the Speech Signal*. PhD thesis, The Center for Speech Technology Research, Edinburgh, 2002.

[2] Miguel Á. Carreira-Perpiñán. *Continuous Latent Variable Models for Dimensionality Reduction and Sequential Data Reconstruction*. PhD thesis, University of Sheffield, UK, February 2001.

[3] Alan A. Wrench and William J. Hardcastle. A multichannel articulatory database and its application for automatic speech recognition. In *5th Seminar on Speech Production: Models and Data*, pages 305–308, Kloster Seeon, Bavaria, 2000.

[4] Alex Smola and Bernhard Schölkhopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, August 2004.

[5] Vladimir Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[6] Steven B. Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. In Alexander Waibel and Kai-Fu Lee, editors, *Readings in speech recognition*, pages 65–74. Morgan Kaufmann Publishers Inc., 1990.

[7] Mike Brooks. The VOICEBOX toolkit. Software available at http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html.

[8] Heikki Hyötyniemi. Multivariate regression – techniques and tools. Technical report, Control Engineering Laboratory, Helsinki University of Technology, July 2001.

[9] Jason Weston, Arthur Gretton, and Andre Elisseeff. SVM practical session (how to get good results without cheating). Machine Learning Summer School 2003, Tuebingen, Germany.

[10] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[11] Gökhan Bakir, Léon Bottou, and Jason Weston. Breaking SVM complexity with cross training. In *18th Annual Conference on Neural Information Processing Systems, NIPS-2004*, 2004.

[12] Yasemin Altun, Ioannis Tsochantaridis, and Thomas Hofmann. Hidden markov support vector machines. In *20th International Conference on Machine Learning ICML-2004*, Washington DC, 2003.