Structured Support Vector Machines Advanced Machine Learning HW2 Columbia University

CS W7442
Devin Jones
dj2374

Monday, March 23, 2015

Abstract

The goal of the assingment is to label music by the Beatles where each song is represented as a sequence of Chroma frames over time. Chroma frames are 12-dimensional continuous features between 0,1 that compactly represent sound in a short time-window.

Various conditional learning methods could be applied to this problem. In this paper, Weller, Ellis, and Jebara show that structured support vector machines outperformed then state of the art methods, Hidden Markov Models, for the 2008 LabROSA Supervised Chord Recognition System.

Method

Structured SVM is a conditional learning method that generalizes large-margin SVM to handle multiclass and structured prediciton. The primal form of SVM [Vapnik] is

$$\min_{w,b,\xi\geq 0} \frac{1}{2}\|w\|^2$$

$$s.t. \forall i \in 1...n : y_i(w^T \cdot x_i - b) \geq 1$$

The primal form of the 1-slack formulation of SVMstruct from [JoFinYu08] is

$$\min_{w,\xi\geq 0} \frac{1}{2}\|w\|^2$$

$$s.t. \forall y'_1..y'_n \in Y : \frac{1}{n}\sum_{i=1}^{n}[w^T \cdot \phi(x, y_i) - w^T \cdot \phi(x, y'_i)] \geq \frac{1}{n}\sum_{i=1}^{n}[\Delta(y_i, y'_i)] - \xi$$

The 1-slack formulation is derived from the n-slack formulation, which is not included here, and is more efficient than the n-slack because constaints are only evaluated if they are violated.

Beyond the obvious difference in constaints between SVM and SVMstruct, in the latter, the feature vectors is generated in a clever way. A joint feature map $\phi(x, y)$ is used to map the combination of x and y into a sparse, wide vector. The resulting vector is DK dimensional, where D is the dimenion of x and K is the number of classes of y. This feature map replaces the x vector when trainingn the SVM. All values of $\phi(x, y)$ where x is not are set to zero.

Additionally the delta function can be formulated to suit the problem. That is, if we can quantify a distance between ys, we can add this information to the model to improve performance. For the chord data encoded into Chroma features, a simple [0,1] loss was used. Perhaps a music theory expert could suggest methods to quantify distances between chords, potentially under a prior of the key which the music piece is composed.

Also instead of using the support vectors to classify the observation outright as in SVM, a predition of y is found by the following evaluation:

$$y = argmax_{y'} w^T \cdot \phi(x, y')$$

This should be intuitive based on the joint feature map; training a model on the sparse feature map produces lower values in w where there is an absence of the original x values.

While experimenting with the suggested Structured SVM packages, SVMstruct and SVMhmm, two methods were tested to evaluate $argmax_{y'}$. The first was was a greedy algorithm, and the second was Viterbi decoding which is implemented in the SVMhmm package.

Data

The data consisted of 180 Beatles songs encoded into beat-synchronous Chroma features. These features are 12 dimensional vectors that range in value from [0,1] which are constructed to estimate the intensity of each semitone, regardless of octave. Each frame represents a beat in the music and has been manually annotated with a chord between 0 and 24 representing all possible chords in music. The number of frames in each song range from 77 to 1806.

The data used for the assignment can be obtained here.

Method

Various feature constructors and contraint settings were tested to find the best model for this data. Feature complexity ranged from a single, linear Chroma frame used to predict the corresponding chord label, ranging to three frames before and three frames after used to predict a chord, as well as a quadratic cross terms were introduced into the model to emulate a non-linear relationship over the Chroma features within and across frames. The most complex feature inlcuded 3 frames before, 3 frames after, and quadratic cross terms.

Using the notation from the paper, this feature would be labeled Q+3-3, and yields a dimensionality of 3654 for the x pattern, and after transformation via ϕ across 25 classes of y, is 91350. This feature model took approximately 6 hours to train over 10 songs. In entirety, the SVMhmm module took approximately 24 hours to train. The SVMstruct module required approximately 12 hours for training. In both training runs, the high dimensional features produced by quadrataic cross terms utilized approximately 90% of the total training time.

A random subset of 10 songs were selected from the data, and model was trained over a random 30% subset of data from each of the 10 songs for every feature. The hyper parameter C was tuned via k-fold cross validation between .01 and 100, and the best C was used to retrain over the same 30%. 3 folds were used in cross validaiton. The Hamming accuracy was used to evaluate said trained model on an unseen random 10% subset of data from the same song. SVMstruct's Matlab API enabled a parallel implementation of cross-validation.

Results

The table below shows averages and standard deviations of accuracies over 10 random songs trained with SVMhmm. Results are sorted by average accuracy.

When comparing these results to the paper, two things are immediately evident. The first is that the accuracies are consistently higher than the results of the paper, and the second is the standard deviations of the the feature models are also consistently higher compared to the paper.

The data used for the paper was found to have some indexing errors which could be the reason for the large imporovement in accuracy. The data used in this analysis had been corrected. Or, perhaps the range of C values were better suited for this model.

I initially hypothesized that the reason for the discrepancy in score variance was due to the method of tuning C and aggregating accuracy used in the paper: for each song, accuracies were computed two times on disjoint, unseen 5% samples from the song, and then averaged. The resulting averaged accuracy was then rolled into a 10 song average to evaluate the model, thus reducing the percieved variability of the scores.

Comparatively, I used k-vold cross validation. In k-fold cross validation, the training data is split into k subsets. For each k, the data is trained on the data less the kth subset, and then tested on the kth subset. This occurs k times, and the accuracy is then averaged over the k training procedures. k-fold cross validation was used to tune the hyper parameter C, and the best C from these runs was used to test accuracy on the 10% holdout set.

Given the noticeable difference in methods, I tested the latter method in the final hours before the assignment deadline, and found this method to increase the variances over the scores. Results from the paper's cross validation and aggregation method are included in the 2nd chart below. Regardless, the accuracies these results are so great that they would indeed outperform the state of the art methods with a t-test, accounting for score variance.

Finally, a third major difference between these results and the paper is that previous frames seem to provide additional predictive value to the model in both CV methods. Previous frames are used in the top 50% of the feature models. This contradicts the findings of the paper - the paper found that including future frames in the model improved performance.

The reason for the discrepancy could be due to the indexing error in the data used in the paper.

SVM HMM: Viterbi Decoding. K-fold CV

Feature	AvgScore	ScoreStddev
hmm_two_before_after	0.7927808	0.1028879
hmm_two_before	0.7884908	0.1014290
hmm_three_before	0.7784181	0.1029668
hmm_two_before_after_quad	0.7735088	0.0781303
hmm_three_before_after	0.7576602	0.1028808
hmm_one_before_after_quad	0.7303012	0.1364928
hmm_one_before	0.7055383	0.1149105
hmm_one_before_after	0.6996760	0.0951485
hmm_two_after_quad	0.6604183	0.1317440
hmm_linear_chroma_quad	0.6530214	0.1057658
hmm_three_after	0.6511218	0.0936476
hmm_three_after_quad	0.6387043	0.1523665
hmm_one_after_quad	0.6137359	0.1119173
hmm_two_after	0.6123229	0.1180711
hmm_one_after	0.6063641	0.1308405
hmm_linear_chroma	0.5925964	0.1562431

SVM HMM: Viterbi Decoding. Different CV

Feature	AvgScore	ScoreStddev
hmm_cv_two_before_after	0.7844828	0.1007224
hmm_cv_three_before	0.7811159	0.0997704
hmm_cv_three_before_after_quad	0.7625272	0.1285803
hmm_cv_two_before	0.7597403	0.1266030
hmm_cv_two_before_after_quad	0.7505376	0.0902101
hmm_cv_three_before_after	0.7451404	0.1405073

Feature	AvgScore	ScoreStddev
hmm_cv_one_before_after_quad	0.6952790	0.1219450
hmm_cv_two_after_quad	0.6845494	0.1150894
hmm_cv_one_before_after	0.6774194	0.1091404
hmm_cv_linear_chroma_quad	0.6616702	0.1185338
hmm_cv_one_before	0.6609442	0.1186834
hmm_cv_three_after	0.6220302	0.1126481
hmm_cv_two_after	0.6115880	0.1422242
hmm_cv_one_after	0.5995717	0.1175138
hmm_cv_one_after_quad	0.5948276	0.1401279
hmm_cv_three_after_quad	0.5862069	0.1678978
hmm_cv_linear_chroma	0.5438972	0.1634116

SVM Struct: Greedy Prediction

The Greedy Prediction method was tested extensively because it was the function refrenced in the lecture notes for prediction. For slack constraints, the hyperparameter C was tuned in cross validation, while in margin constraint models, e was tuned. Accuracies are lower in comparision to Viterbi Decoding, likley because that method has the ability to produce a richer decoding in producing ϕ, the combination of x and y used for prediction.

Feature	AvgScore	ScoreStddev
slack_three_before_after_quad	0.6631407	0.1316411
slack_three_before_after	0.6296897	0.1097605
slack_two_before_after_quad	0.6080898	0.1582449
slack_one_before_after	0.5760101	0.1671713
margin_three_before	0.5743484	0.1151140
slack_two_before_after	0.5643893	0.1287585
slack_two_before	0.5586952	0.2077504
slack_three_before	0.5555480	0.1113978
margin_three_before_after_quad	0.5420169	0.1437399
margin_two_before_after_quad	0.5412431	0.1218176
margin_three_before_after	0.5259532	0.1245796

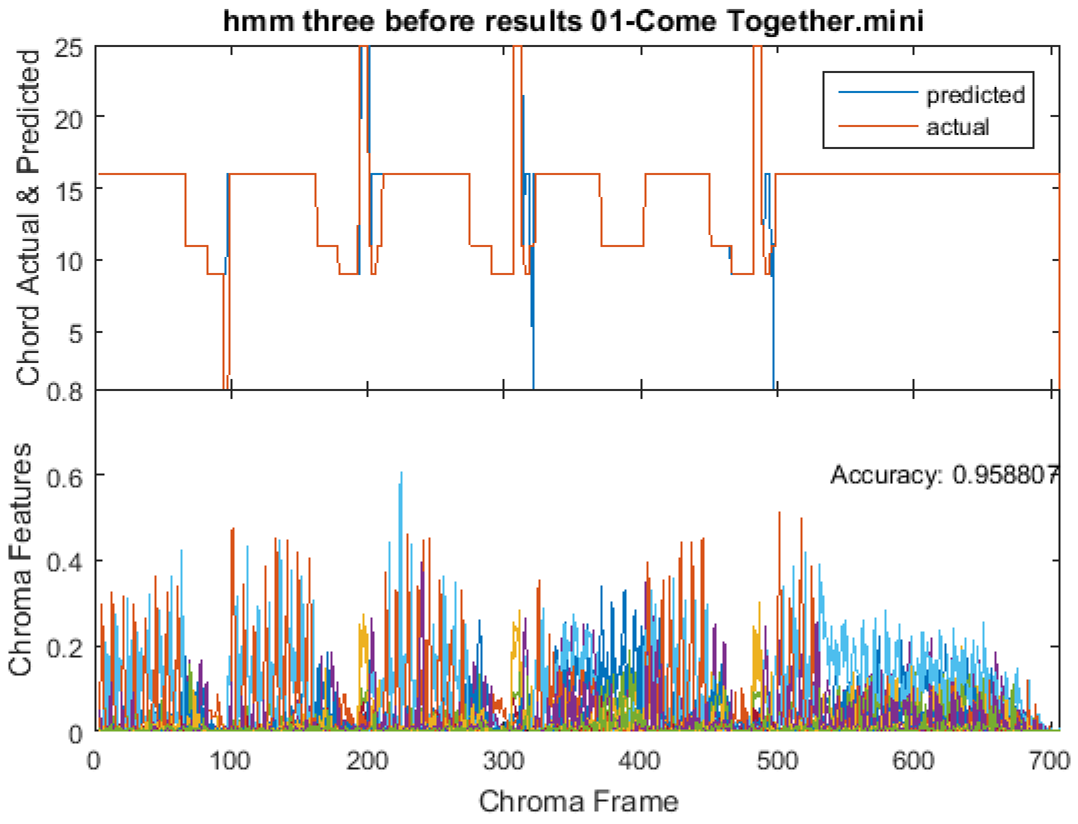
Feature	AvgScore	ScoreStddev
slack_one_before_after_quad	0.5256327	0.1217673
margin_one_before	0.5237705	0.1161583
slack_one_before	0.5237415	0.0903884
slack_one_after_quad	0.5230147	0.1505399
margin_two_before	0.5183957	0.1367383
slack_linear_chroma_quad	0.5173711	0.1121178
margin_linear_chroma	0.5166508	0.1461103
margin_one_before_after_quad	0.5154065	0.0786209
margin_linear_chroma_quad	0.5085828	0.0914193
margin_two_before_after	0.5055648	0.1133898
margin_three_after	0.4975824	0.1407363
slack_two_after	0.4974452	0.1521602
margin_one_before_after	0.4941127	0.1039655
slack_three_after_quad	0.4937974	0.1703892
slack_three_after	0.4900998	0.1914167
margin_one_after	0.4786855	0.0971992
margin_two_after_quad	0.4768997	0.1431857
slack_one_after	0.4759160	0.1966023
slack_linear_chroma	0.4720706	0.2315329
slack_two_after_quad	0.4709395	0.1571295
margin_one_after_quad	0.4670508	0.1063359
margin_three_after_quad	0.4407489	0.1285063
margin_two_after	0.4397058	0.1553045

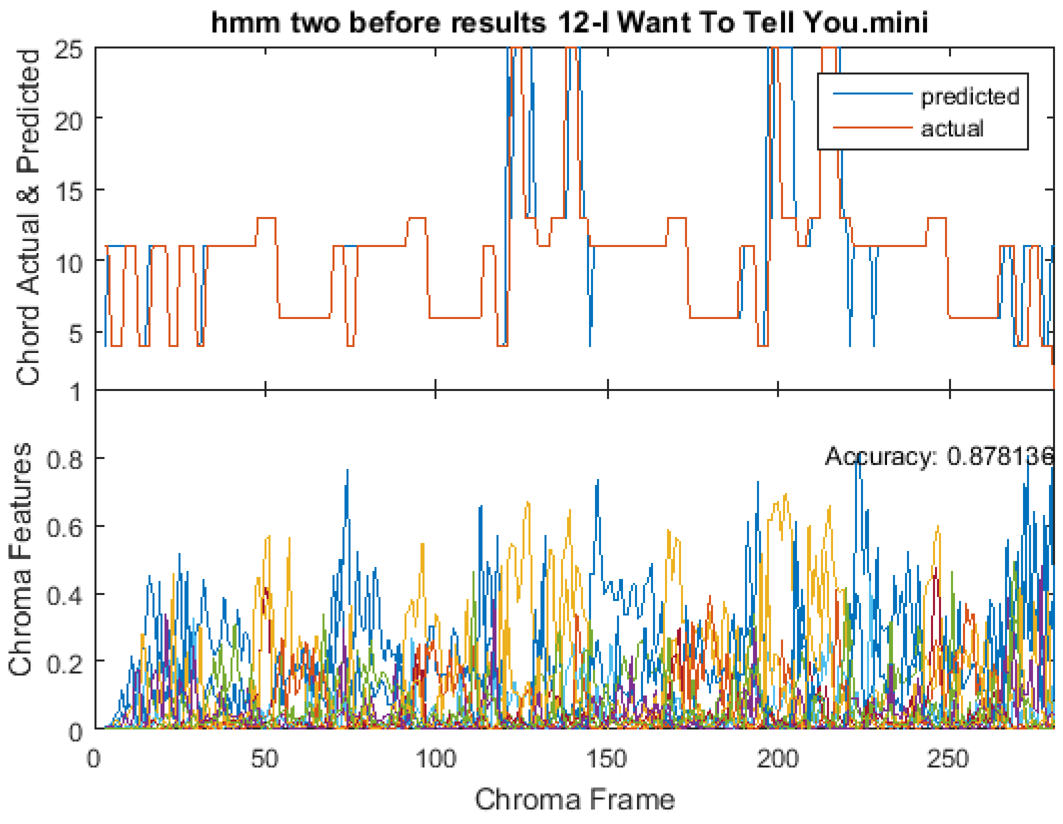
Plots

SVM HMM Plots

Below are visualizations for the best scoring songs for the 2 best performing models from the SVMhmm k-fold cross validatoin runs.

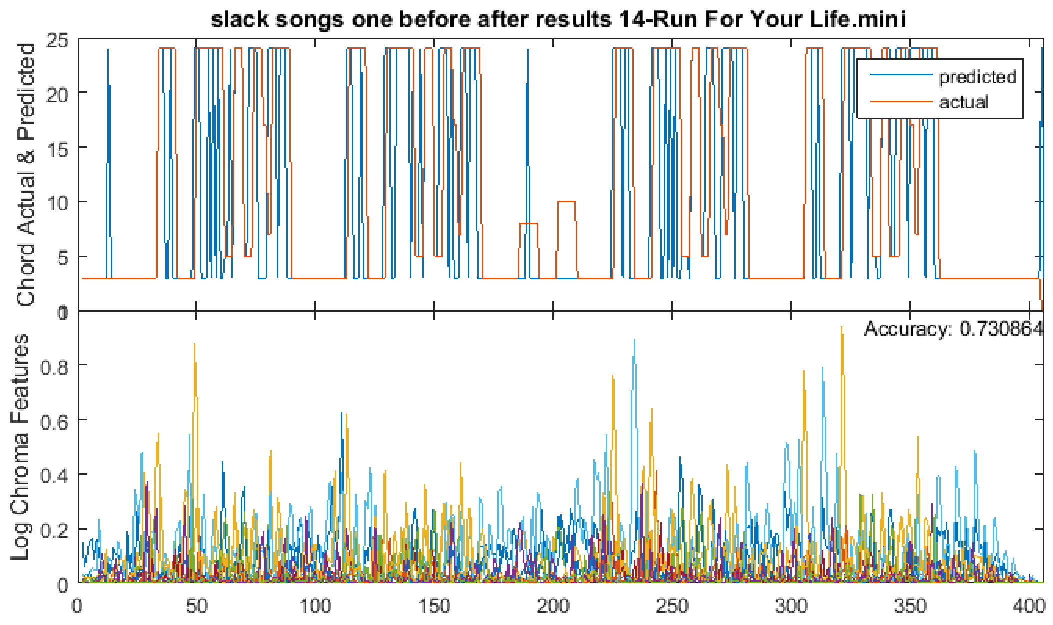
I've been told that "Come Together" is one of the simplest chord progressions in all of modern rock and roll, so I am not surprised that this song scored the highest. This is evident in the low variability in the true value of the chords plot. Additionally, the Chroma features look visually consistent across the whole song. Perhaps the audio recording was cleaner for this song.

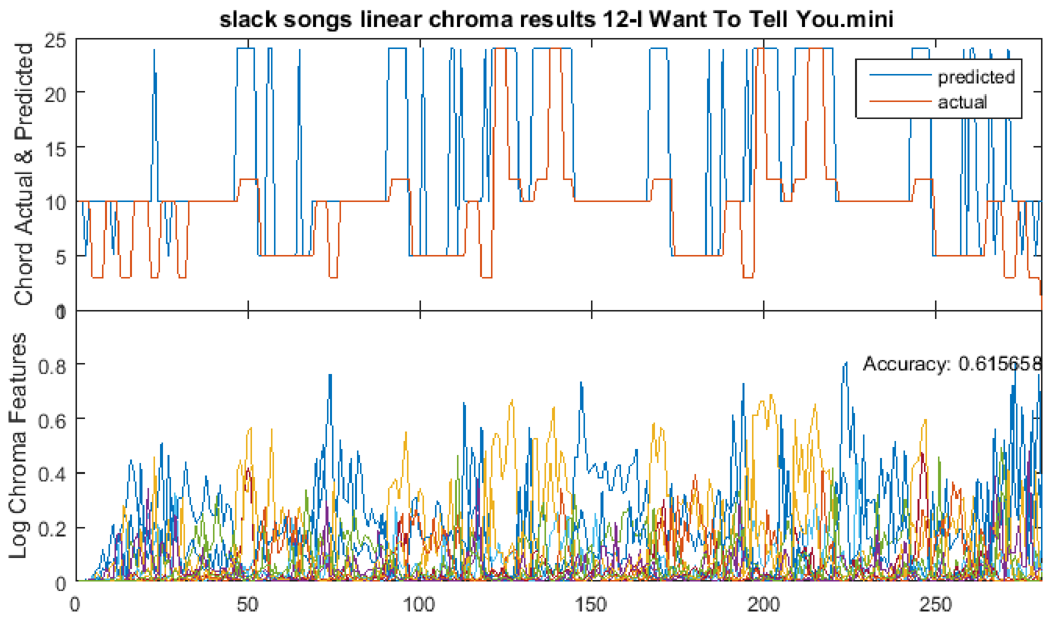




SVM Struct Plots

As mentioned earlier, performance was significantly lower for the SVMstruct runs. Below are the top performing models and songs for the greedy prediction algorithm.





Conclusion and Future Work

SVMhmm is a state of the art model for the chord prediction problem, out performing other methods (HMM) even if we normalize by standard deviations.

Further improvements in software could be made to create a Matlab API for the SVMhmm module to enable faster development and more of Matlab resources such as parellelization. Additionally, the quadratic cross terms are used in lieu of non-linear kernels due to constraints in the C implementation of SVMstruct. Nonlinear kernel implmentation would enable rapid testing and devlopment of nonlinear features that best suit this problem without increasing the dimensionality of the x vector by a polynomial factor, thus drastically reducing the training time for non-linear data mappings.

Perhaps variability of scores could be reduced by using more than 30% training data.

Finally, further research could be done in music theory to quantify the diference between chords, perhaps under a prior belief that a song is composed in a specific key.