Exposing Parameters of a Trained Dynamic Model for Interactive Music Creation

Dan Morris
Microsoft Research
Redmond, WA
dan@microsoft.com

Ian Simon
University of Washington
Seattle, WA
iansimon@cs.washington.edu

Sumit Basu
Microsoft Research
Redmond, WA
sumitb@microsoft.com

Abstract

As machine learning (ML) systems emerge in end-user applications, learning algorithms and classifiers will need to be robust to an increasingly unpredictable operating environment. In many cases, the parameters governing a learning system cannot be optimized for every user scenario, nor can users typically manipulate parameters defined in the space and terminology of ML.

Conventional approaches to user-oriented ML systems have typically hidden this complexity from users by automating parameter adjustment. We propose a new paradigm, in which model and algorithm parameters are exposed directly to end-users with intuitive labels, suitable for applications where parameters cannot be automatically optimized or where there is additional motivation – such as creative flexibility – to expose, rather than fix or automatically adapt, learning parameters.

In our CHI 2008 paper, we introduced and evaluated MySong, a system that uses a Hidden Markov Model to generate chords to accompany a vocal melody. The present paper formally describes the learning underlying MySong and discusses the mechanisms by which MySong's learning parameters are exposed to users, as a case study in making ML systems user-configurable. We discuss the generalizability of this approach, and propose that intuitively exposing ML parameters is a key challenge for the ML and human-computer-interaction communities.

1. Introduction and Related Work

Machine learning (ML) systems have long been used by the scientific community for data analysis and scientific inference, but recently have begun to achieve widespread success in consumer applications as well. Recommender systems (Linden et al. 2003) and e-mail "spam" filtering applications (McGregor 2007) exemplify the recent commercial success of ML systems.

In the research community, ML systems have been deployed for a variety of end-user applications, generally with a level of automation that hides the complexity of the underlying learning system from users. A popular approach has been to allow users to provide labeled examples for a supervised learning system that infers rules and applies them to subsequent data that require classification. In this way, systems develop user-specific models, but users need to interact with the system only by labeling examples. This general approach has been applied to image search (Nguyen et al. 2007, Fogarty et al. 2008), user task identification (Shen et al. 2006), determination of group scheduling preferences (Brzozowski et al. 2006), prediction of interruptibility (Fogarty et al. 2004, Horvitz et al. 2004), and e-mail classification (Kiritchenko and Matwin 2001, Dredze et al. 2006). Supervised machine learning with explicit training by users has been successfully used in commercial applications as well, particularly in speech recognition (Baker 1975) and handwriting recognition (Plamondon and Srihari 2000).

Some user-oriented ML systems have demanded even less of users, running entirely without user invention either through unsupervised learning or through supervised learning with labeling via implicit data collection. This approach has been applied to classification of user activities (Kushmerick and Lau, 2005) and prediction of a user's presence and availability for meetings or interruptions (Horvitz et al., 2002).

An additional body of work has made ML algorithms and models accessible to a wider community, but has targeted application developers and designers, rather than end-users. The Weka library (Witten and Frank, 2005), for example, has made ML techniques available to a wide variety of domains and applications by exposing numerous classifiers through a consistent and simple programming interface. Fogarty et al. (2007) similarly present the "SUBTLE" toolkit, a programmatic interface for sensor data analysis, including online adaptation and automatic feature extraction, to allow programmers and application designers to make inferences on sensor data with minimal experience in ML or signal processing. Fails and Olsen (2003) present a system ("Crayons") for training an image classifier, intended for application designers who want to build and export classifiers without programmatically processing a training set. Ware et al. (2002) allow direct manipulation of decision tree thresholds, but this process is centered around the action of building the classifier itself, and thus is targeted at developers, not end-users.

Virtually no systems, to our knowledge, have directly exposed the parameters of a machine learning system to end-users. Tóth (2003), Ware et al. (2002), and Hartmann et al. (2007) allow direct manipulation of machine learning parameters (genetic algorithm evolution parameters, decision tree thresholds, and classifier distance thresholds, respectively), but all are centered around the action of building the classifier itself, and thus are targeted at

application designers, not end-users.

The primary reason that end-users have traditionally been shielded from ML parameter adjustment or model details is that few users have the expertise or even the vocabulary required to meaningfully interact with a system at this level. However, recent field study results (Tullio et al. 2007, Stumpf et al. 2007) suggest that users can in fact build meaningful mental models of machine learning systems, suggesting that exposing more details of such systems may be a promising approach.

We postulate several reasons why, despite the barriers in intuitive interface development, exposing ML concepts to end-users will benefit certain scenarios:

1) As ML systems are increasingly deployed "in the wild", it will become increasingly difficult to select parameters that apply to all environments. End-user adjustment allows tailoring to a specific scenario.

2) Allowing direct manipulation of an underlying model promotes user exploration and exploratory usage of applications that would be stifled with fixed parameters or automatic parameter adjustment.

3) This approach will also lend itself directly to the application-designer scenarios discussed above, where a user with *some* expertise in machine learning or willingness to perform deeper interactions is building or editing a learning model that will be exported and applied to an end-user system.

4) In certain scenarios, such as the one presented in this paper, there is intrinsic value to providing more degrees of freedom in a user's interaction with a model. In this case, these additional degrees of freedom lead directly to increased creative expressiveness.

In our CHI 2008 paper (Simon et al. 2008), we introduced MySong, a system that uses a Hidden Markov Model to automatically generate chords to accompany a vocal melody. We have demonstrated that this is a powerful tool for allowing non-musicians to create accompanied music. However, allowing the user to explore the space of possible accompaniments – without requiring knowledge of either music theory or machine learning – further enhances the creative expression available with this tool and renders it useful for novices and songwriters alike.

In this paper, we make the following three contributions:

1) We formally describe MySong's algorithm for automatically generating accompaniments.

2) We discuss the novel mechanisms by which MySong's machine learning parameters are exposed to users, as a case study in the increasingly-important space of making AI and ML systems user-configurable.

3) We discuss the generalization of this approach to other domains, and we conclude by calling on the ML and HCI communities to develop additional intuitive mechanisms for exposing details of machine learning systems to end-users.

We conclude this section with a brief description of other work in the computer music domain that ties intuitive labels or parameters to machine learning systems. For a more comprehensive overview of work specifically related to the MySong system, see (Simon et al. 2008).

Legaspi et al. (2007) used explicit labels to learn a model mapping musical features to affect and emotion, for the purpose of generating new music that inspires a user-specific emotional response. Given training data and a desired affective descriptive, this system builds a first-order logic that generates song components and uses a genetic algorithm to build chords from fundamental musical units. Turnbull et al. (2008) and Torres et al. (2007) solve the related problem of conducting a search based on intuitive labels; this system learns a Gaussian mixture model that maps local audio features to intuitive labels using an explicitly-labeled data set, and allows users to search a music database using intuitive labels.

To our knowledge, no previous systems exist that allow users to directly manipulate components or parameters of a learning system for a creativity-support application.

2. System Description

2.1 MySong Overview

The MySong system allows a user with no knowledge of chords or music theory to quickly create and explore chord-based accompaniments for a vocal melody. Primary goals of the system are to enable non-musically-trained users to participate in the craft of songwriting and to provide songwriters and musicians with an interactive, exploratory scratchpad for songs.

The basic interaction paradigm requires a user to simply click on a "record" button, sing a melody while listening to a drum pattern that fixes the timing of musical beats, and click on a "stop" button to end the recording. MySong then immediately generates a set of chords to accompany the recorded vocal melody and uses a commercial system to render those chords as an audio accompaniment in a user-selected style. The core of this system is a Hidden Markov Model that represents a chord sequence and is used to generate chords to accompany a vocal melody; in this section we will describe the architecture and training of that model. In Section 3, we will discuss several mechanisms by which users can intuitively interact with that model after recording a melody.

2.2 Model and Assumptions

We model a vocal melody as a sequence of notes in which each element corresponds to a specific "pitch class": a frequency corresponding to one of the standard 12 tones in the chromatic musical scale (e.g. C, C#, D, etc.). In our training phase, this sequence will be derived from published musical scores (Section 2.3). In our decoding phase, this sequence is derived by sampling and discretizing a user's vocal melody (Section 2.4).

In popular musical genres, a musical accompaniment is typically represented for performance purposes as a sequence of chords (often referred to as a "lead sheet"). We thus build our model around this representation, and assume that this is adequate for producing appropriate accompaniments. This assumption is appropriate for a wide variety of popular musical genres (rock, pop, country, R&B, jazz, etc.), but would not be valid for non-chord-based music, including a significant fraction of classical pieces, heavy metal pieces, etc.

We make the following additional assumptions:

1) Each chord in a sequence lasts for exactly one measure. Here we use the term "measure" to refer to the smallest amount of time for which a chord will be played; in general this corresponds to a typical musical measure, but this is not necessary; hence this assumption can be made without loss of generality.
2) All notes can be reduced to a single octave without losing information that is meaningful to chord selection; therefore, there are only 12 possible pitch classes.
3) A sufficient statistic (for the purpose of chordal accompaniment) for the notes in a measure is the fraction of the measure during which each pitch class (C, C#, D, etc.) is heard.
4) The musical key (the prior distribution of chords and melody notes) does not change within a melody.
5) Possible chords are chosen from a finite dictionary available during model training.
6) The location of measure boundaries in a vocal melody is known during both training and interactive use of our model. In the training phase, measure boundaries are delineated in training data; during interactive use, measure boundaries are determined by the timing of the drum beat along with which the user sings (Section 2.1).

2.2.1 Notes-only model

Given these assumptions, one simple model is to choose the chord for each measure considering *only* the notes that appear in that measure. So, for each measure, the chord is chosen to maximize *P*(notes | chord). We will begin our discussion with this simple model, and build our Hidden Markov Model on this description.

We sample the sequence of notes observed in a measure of melody at regular intervals (these intervals are arbitrarily short and do not correspond to musical note durations), and our model's observations are the notes occurring at each interval *i*. Because we assume that the precise temporal ordering of notes is not relevant to the selection of a chord within a measure (assumption (3) above), this sampling is equivalent to building a "pitch histogram": a 12-dimensional vector x in which each element corresponds to the amount of time spent in the corresponding pitch class.

For a given chord type c, we refer to the vector of (a priori) *expected* pitch class frequencies (i.e., as estimated from training data) for a measure containing that chord as μ_c, and the element of μ_c corresponding to a specific pitch class p as μ_{c_p}.

We thus model the probability of the pitch histogram x occurring in a measure with a chord c as:

$$P(x|c) = \prod_{i=1}^{T} P(n_i|\mu_c)$$

Here n_i is the note appearing in timeslice i and T is the number of timeslices in a measure. Looking at this in the log space, noticing that the probability of seeing note n at some timeslice given our model is precisely μ_n, and then letting T go to infinity (infinitely short timeslices), we have:

$$\log P(x|c) =$$

$$\sum_{i=1}^{T} \log \mu_{c_i} \propto \sum_{k=1}^{12} t_k \log \mu_{c_k} \propto \sum_{k=1}^{12} x_k \log u_{c_k}$$

Here k is a pitch class and t_k is the amount of time spent in the pitch class k. In short, we can compute the relative probability of an observation vector x for a chord c by taking the dot-product of that observation vector with the log of the vector of expected pitch class frequencies u_{c_k}.

2.2.2 Chord transitions

A simple extension to the notes-only model is to also model the likelihood of transitions among chords, i.e. to incorporate the fact that for a chord c_t in a chord sequence (chord type c occurring at measure t), the distribution of probabilities over possible subsequent chords c_{t+1} is highly non-uniform. In other words, in popular music, certain chord transitions are much more likely than others, an important basic principle in music theory.

We represent the probability of a chord c_{t+1} occurring in the measure following chord c_t as $P(c_{t+1} \mid c_t)$. These probabilities can be stored in an *m*-by-*m* table, where *m* is the total number of chords in our dictionary.

2.2.3 Hidden Markov Model

MySong uses as its core representation a Hidden Markov Model in which each measure corresponds to a single node whose (initially unknown) state represents the chord selected to be played during that measure. The observation at each node is the melody fragment sung during that measure, treated as in Section 2.2.1. Transition probabilities among states are estimated from training data (Section 2.3) and are stored as in Section 2.2.2.

2.3 Training

We train the Hidden Markov Model described in Section 2.2.3 using a database of 300 "lead sheets": published musical scores containing melodies along with chord sequences aligned to those melodies. Without loss of generality, all of these lead sheets are transposed to a single musical key before training (this will not limit us to working with melodies in this key; we discuss our handling

of arbitrary-key melodies in Section 2.6).

Transition probabilities (Section 2.2.2) are computed by counting the chord transitions occurring from each chord type to each chord type in the database and normalizing these counts. "Beginning of song" and "end of song" are included in the transition matrix, but will not be part of the dictionary from which node states are assigned.

2.4 Pitch-tracking

In order to provide the observation vector x at each measure, a pitch-tracking step analyzes each measure of vocal audio, computes the fundamental frequency over 40ms windows at 10ms intervals, discretizes those frequencies into 12 pitch classes, and builds the vector x as a histogram of samples observed in each pitch class.

We note that this is *not* equivalent to the problem of transcribing a musical melody from a singer's voice; because our model assumes that a pitch histogram is a sufficient description of a measure for purposes of harmonization, we do not build a detailed rhythmic transcription of the melody. This approach allows MySong to be robust to small errors in both a user's pitch accuracy and our own frequency estimation.

We compute the fundamental frequency using a variant on the method of (Boersma, 1993), but – as we assume that octave information is not relevant to harmonization – we do not perform the dynamic programming step described in that work, which primarily serves to eliminate octave errors. We briefly summarize our pitch-tracking procedure here; constants are selected empirically and have been robust to a wide range of vocal audio:

1) Extract a single measure m of audio data, sampled at 22kHz and normalized to the range (-1.0, 1.0).
2) Extract 40ms windows w of audio data at 10ms intervals (100Hz), and center each window w around zero-amplitude by subtracting its mean.
3) Discard any window w whose root-mean-squared (RMS) amplitude is less than 0.01 or whose RMS amplitude is less than 0.05 times the RMS amplitude of the measure; these heuristics indicate near-silence.
4) Compute the power spectrum (the magnitude of the zero-padded FFT) of the window w, then compute the autocorrelation a of the window w by taking the IFFT of its power spectrum.
5) Normalize the autocorrelation a by the mean-squared amplitude (energy) of the window w.
6) Within the range of a corresponding to the frequency range [75Hz, 300Hz] find the minimum and maximum normalized autocorrelation values $a_{\min}$ and $a_{\max}$.
7) Discard any window w for which $a_{\max} < 0.4$ or $a_{\min} > 0.05$. These heuristics indicate weak autocorrelation peaks and, consequently, non-pitched voice.
8) For each qualifying window w, compute the frequency $f_{\max}$ corresponding to the peak $a_{\max}$. This corresponds to the estimated fundamental frequency at w.
9) Compute the continuous (non-discretized) pitch class p corresponding to $f_{\max}$ as $p = 12\log_2(f_{max}/f_c)$, where f_c is the frequency of a member of a known musical pitch class (we choose the note C5=523.2Hz).
10) For all windows w in the measure m, compute the offset between p and the nearest known musical pitch class, and compute the mean p_{offset} of all such offsets.
11) Add p_{offset} to the value p for each window; this shifts the computed pitch sequence to optimally align with the standard chromatic scale (i.e., this minimizes the mean-squared difference between the computed frequencies and standard musical notes subject to the constraint that relative pitches must be preserved).
12) Round each value p to the nearest integer p_{int} and compute the final pitch class as $p_{\text{int}} \bmod 12$; the histogram of these integer pitch classes over the entire measure m is the observation vector x for this measure.

2.5 Decoding

Given the vector x output by the pitch-tracking system for each measure in a recorded melody, MySong chooses the sequence of chords that maximizes the likelihood of this sequence of vectors. I.e., our harmonization model selects chords that maximize the following objective function over the sequence *chords* for the sequence of observation vectors *melody*:

$$L = \log P(chords) + \log P(melody|chords)$$

…where:

$$\log P(chords) = \log P(c_1|start) + \sum_{i=2}^{n} \log P(c_i|c_{i-1}) + \log P(end|c_n)$$

…and:

$$\log P(melody|chords) = \sum_{i=1}^{n} \log P(x_i|c_i)$$

Here n is the total number of recorded measures (known a priori since the user sang along with a drum beat that defined measure timing), c_k is the chord assigned to measure k, $P(c_1|start)$ is the (known) probability that a song begins with c_1, $P(end|c_n)$ is the probability that a (known) probability that a song ends with c_n, and x_k is the observation vector corresponding to measure k.

We use a single parameter $0 \le \alpha \le 1$ to weight the importance of observations versus transitions (the interpretation of this parameter is discussed in more detail in Section 3.2). The objective function then becomes:

$$L = (1-\alpha)\log P(chords) + \alpha \log P(melody|chords)$$

For the Hidden Markov Model defined in Section 2.2.3, the Viterbi algorithm chooses the sequence of chords $c_1 \ldots c_n$ that maximizes this total likelihood. This is the set of chords we present to the user for a new melody.

2.6 Key determination

As we discuss above, we transpose all the songs in our training database into a single musical key (expected

distribution of notes and chords) before training our model. It is musically reasonable to assume that the transition matrices in each key are identical other than a simple circular shift of the transition matrix, so we maximize the effectiveness of our training database by considering all 12 keys to be equivalent for the purposes of harmonization.

However, when harmonizing a new melody, we do not know the key in which the user sang. We therefore consider multiple transpositions T_k applied to the vocal melody and all candidate chord sequences, where $0 \leq k < 12$. $T_k(chords)$ transposes each chord by k half-steps, and $T_k(melody)$ transposes the new melody (the observation vectors x) by k half-steps. The objective function then becomes:

$$L = \log P\big(T_k(chords)\big) + \log P\big(T_k(melody)|T_k(chords)\big)$$

We now optimize over chords and k by computing the optimal chord sequence for each k, and then choosing the key k (and corresponding sequence) with the highest likelihood. Empirically, this method nearly always selects the same key that a musician would manually assign to a new melody. For example, for the 26 melodies used in a recent evaluation of our system (Simon et al. 2008), this approach selected the correct key for all 26 melodies.

3. Exposing Learning Parameters

Using the model we presented in Section 2, a user could record a vocal melody and have a mathematically-optimal sequence of chords assigned to that melody. In many cases this would provide satisfactory results, but we would like the user to be able to subsequently manipulate the selected sequence of chords for two reasons:

1) A major goal of our system is to provide a *creativity support tool*; if users cannot modify the output of our core algorithm, we are not enabling a *creative* process.
2) The mathematically-optimal sequence of chords for any model may not always be the subjectively-optimal sequence of chords, and subjective preference may vary among users. Therefore it is important to allow a user to modify the selected chord sequence, treating our automatically-selected chords as a starting point.

However, our target audience is unfamiliar with musical notation, so asking them to directly manipulate the chord sequence would undermine a major goal of our system. Furthermore, even a musically-trained user derives the most benefit from this system when leveraging the underlying optimization framework to rapidly explore chord patterns. Therefore we would like the manipulation stage that takes place after the original optimization to also be enhanced by the underlying learning mechanisms.

However, our target audience is also unfamiliar with concepts and notation from machine learning, and could not reasonably be asked to manipulate "observation weights", "transition matrices", etc. Therefore we now turn our attention to the mechanisms by which MySong exposes components of the underlying learning system via interactions that are *intuitive* to users. Each of these mechanisms will modify our objective function in some way; we note that the computational efficiency of the Viterbi procedure allows all of these mechanisms to be manipulated in real-time.

3.1 The "Happy Factor"

In practice, a single transition matrix is insufficient to capture the variation among chord progressions. Orthogonal to the classification of songs into musical keys, songs can typically be assigned to a "mode", which indicates a particular distribution of chords within the distribution of notes representing the key, and an associated emotional character. The two most common modes in popular music are the "major" and "minor" modes. We therefore divide our training database into major- and minor-mode songs before performing the training procedure described in Section 2.3 and compute separate transition probabilities, henceforth called P_{maj} and P_{min}, for each sub-database.

We perform this modal division of our training database automatically, using an iterative procedure. To begin, we initialize the transition matrices P_{maj} and P_{min} using a series of simple heuristics (Appendix A).

After initialization, we alternate between the following two steps in a k-means-like manner until the sets of songs classified as major and minor do not change:

1) For each song in the database, estimate its likelihood using both the major and minor transition models, and assign the song to whichever yields higher likelihood. Likelihood is computed as in Section 2.5.
2) Re-compute the major and minor transition matrices separately using the set of songs assigned to each model, by counting all transitions and normalizing.

When this procedure is complete, we have two transition matrices P_{maj} and P_{min} available during the decoding stage. We use a parameter $0 \leq \beta \leq 1$ (called the *happy factor*) to weight the relative contribution of these two transition models. The transition probabilities used in our objective function now look like:

$$\log P\,(c_i|c_{i-1}) = \beta \log P_{\text{maj}}\,(c_i|c_{i-1}) + (1-\beta)\log P_{\text{min}}\,(c_i|c_{i-1})$$

The reader will likely find this form of mixing unusual, as we are *linearly* mixing transition matrices in the *log* domain. In the non-log domain, mixing two transition matrices with β and $1-\beta$ yields a valid transition matrix:

$$P(c_i|c_{i-1}) = \beta P_{maj}\,(c_i|c_{i-1}) + (1-\beta)P_{min}\,(c_i|c_{i-1})$$

However, in the log domain, we are effectively taking the product of the two transition matrices raised to complementary powers, which will not (without normalization) result in a valid transition matrix:

$$P(c_i|c_{i-1}) = P_{maj}\,(c_i|c_{i-1})^{\beta} \cdot P_{min}\,(c_i|c_{i-1})^{(1-\beta)}$$

Empirically, this method produced chord combinations that were perceived to be better than those achieved via linear mixing; the reasons behind this are subtle. Since the two matrices are sufficiently different from each other (i.e., very major and very minor), the weighted average of the two results in a musically non-sensical middle ground. To see why this happens, imagine that in the major-key transition matrix the C chord always transitions to a G chord, whereas in the minor-key transition matrix the C chord always transitions to an A-minor chord. Linearly mixing these results in the transition probability being split between G and A-minor according to β. For medium values of β, both are almost equally likely, which in practice is not the case in songs that mix major and minor components.

In fact, while many songs have both major- and minor-mode components, major-mode songs tend to express this combination by reinforcing the minor-mode components that are already typical to major-mode songs (and vice versa), as opposed to mixing in all possible minor transitions/chords. This is precisely what our log-space mixing does: when transition matrices are multiplied, common components are reinforced, while disparate components are reduced in probability.

We highlight this point as an example of a non-traditional engineering/learning decision, guided by usability and musicality, at the expense of "correctness" of the underlying system. We argue that such tradeoffs of usability and intuitiveness for "correctness" may be increasingly appropriate in user-centric ML systems.

The value β is directly exposed to the user as a slider on MySong's graphical user interface, and is labeled as the "Happy Factor". Users do not need to understand the actual implementation of this factor as a transition-matrix blending weight, nor do non-musically-trained users need to understand the nature of major and minor modes to effectively use this slider. In the usability study we present in (Simon et al. 2008), non-musically-trained users were able to create subjectively-appropriate accompaniments for melodies. During this study, *all* 13 participants made extensive and successful use of this slider.

We also note that at values of 0 and 1 for β, we are selecting the learned transition matrices p_{min} or p_{maj},.

3.2 The "Jazz Factor"

The objective function presented in Section 2.5 *summed* the likelihood of observations and transitions, implicitly assuming that these contribute equally to the subjective quality of a chord sequence. In practice, this is not always the case.

We use a single parameter $0 \le \alpha \le 1$, called the *jazz factor*, to weight the importance of observations versus transitions. The objective function then becomes:

$$L = (1-\alpha)\log P(chords) + \alpha \log P(melody|chords)$$

Setting $\alpha = 1$ causes MySong to ignore the transition matrices entirely, leading to surprising and unfamiliar chord progressions that optimally match the recorded melody. Setting $\alpha = 0$ causes MySong to ignore the observed melody entirely, leading to familiar chord progressions; even this extreme setting can be useful, for example, when building an accompaniment for non-pitched vocals (such as rap) or experimenting with instrumental chord patterns.

We expose this observation-weighting factor as another slider on MySong's graphical user interface, labeled as the "Jazz Factor". We do not claim that this in any way represents a bias toward accompaniments in the jazz genre, but pilot testing suggested that this was a fairly intuitive name for this parameter.

We highlight that even though the precise behavior of this slider is difficult to explain without introducing machine learning terminology, novice users were able to effectively use this slider to explore a wide space of accompaniments, all of which were musically reasonable (as they each optimized our objective function for a particular value of α). This underscores another key point of this work: for certain applications, providing reasonably intuitive handles to algorithmic parameters, with appropriate constraints that prevent non-intuitive behavior, can allow users to explore a parameter space and therefore make effective use of the underlying system while introducing human knowledge and subjective preference.

3.3 "Top Chords" List

When working with a chord progression or accompaniment, musicians often experiment with replacing individual chords with other chords that are expected to be appropriate at that point in the song and comparing the subjective impact of each. The model underlying MySong allows us to provide this same experimentation process to non-musically-trained users who would have no intuitive metric for possibly-appropriate chord substitutions.

In MySong's graphical user interface, right-clicking on a chord brings up a list of the top five chords that MySong "recommends" for this measure. For the i^{th} measure, we compute this listed by sorting all possible chords according to the following quantity:

$$L_i = \alpha \log P(x_i|c_i) + (1-\alpha)\log P(c_i|c_{i-1}) + (1-\alpha)\log P(c_{i+1}|c_i)$$

These are simply the terms in the global objective function which are dependent on the chord in the i^{th} measure. This metric takes into account both melody and chord context, but again does not require a user to understand the underlying model.

3.4 Chord-locking

Often a user encounters a chord that is subjectively pleasing when paired with the melody at a particular measure. The user might like to continue varying other parameters (e.g. α and β) while maintaining this chord, and

guaranteeing that transitions in and out of this measure will still follow the appropriate "rules" defined by the trained transition model.

MySong thus allows a user to "lock" a particular chord via the graphical user interface. When a chord is "locked", further manipulation of α and β won't change this chord, but – importantly – the selection of adjacent chords will reflect the locked chord. Locking the chord at the i^{th} measure to chord C has the following effect on our objective function:

$$P(c_i|c_{i-1}) = \begin{cases} 1 & \text{if } c_i = C \\ 0 & \text{if } c_i \neq C \end{cases}$$

3.5 Ignore melody

For several reasons, it is sometimes useful to disregard the vocal melody for a measure. For example, it may be the case that a measure was performed inaccurately, that a measure contains non-musical speech, or that pitch-tracking errors yield an observation vector that does not correlate well with the user's intended pitch. MySong thus allows a user to ignore a particular measure of audio via the graphical user interface.

Ignoring the vocal input for the i^{th} measure can be efficiently accomplished by simply ignoring this measure's $\log P(x_i|c_i)$ term, leaving it out of the global optimization. In practice this is implemented by locally setting the observation weight α to 0.

4. Results

For brevity, we will not repeat the results of the usability studies that are the focus of (Simon et al. 2008). We summarize our two studies as follows:

1) A listening study showed that MySong produces subjectively-appropriate accompaniments. This is a validation of the model and objective function described in Section 2.

2) A usability study showed that MySong's interaction techniques provided are intuitive to novice, non-musically-trained users. This is a validation of the interaction techniques described in Section 3, as well as the overall paradigm of vocally-driven automatic accompaniment.

As further verification of the intuitiveness of the mechanisms by which we have exposed MySong's learning parameters to users, we provide several quotes from participants in our usability study that provide qualitative support for the quantitative results of our usability study. We consider qualitative, subjective validation to be central to our core argument that, in certain situations, it *is* appropriate to expose learning parameters directly to an end-user.

When asked the free-response question "What things were most helpful about this system?", the following responses were provided by study participants:

- "The 'happy/sad' and 'jazzy' options are useful for learning." (P3)
- "The ability to change the 'mood' of the song just by moving the slider." (P5)
- "The ranges that were available with the happy & jazz factor." (P8)
- "Easy to understand concepts (happy, jazzy)" (P9)
- "Sliders are easy." (P10)
- "[Sliders are] easy, not complicated. No need to think." (P11)

5. Discussion: Interactive Decoding

We have presented MySong as a case study in "interactive decoding": bringing the user into the ML-driven decoding process by exposing system parameters in intuitive terms. We highlight again that this differs from previous work in user systems, which typically minimize the degree to which learning parameters are exposed to users, and from previous work in ML toolkits, which typically expose a large set of parameters to developers and engineers.

We propose that this new "middle ground" could have similar success in a number of other areas. A number of creativity-oriented systems, for example, could provide similar assistance to untrained users while providing sufficient degrees of freedom to allow a deep creative process. This approach could be applied not just to musical accompaniment, but to painting, computational photography, painting, graphic design, writing, etc.

Outside of the creativity space, any system that uses machine learning to process complex data could also benefit from this approach to allow the user some level of interactivity. Image- and video-editing applications, for example, increasingly provide facilities for face detection and processing, keyframe and thumbnail selection, object removal, automatic cropping, etc. In almost all of these cases, the underlying system has a series of parameters that may typically be hidden from users to minimize complexity, but in all of these cases, as with MySong, users' overall productivity may in fact benefit from an ability to rapidly explore the space of relevant parameters.

We do not argue that all end-user systems that incorporate some form of machine learning should expose core parameters to users; inevitably, this process does add interface complexity and the potential for users to develop an inaccurate intuition for how a system behaves. We propose that this approach will successful in spaces where target users have sufficient motivation to explore a parameter space and build an intuition for how parameters behave. Creativity support is an excellent example of such a space, where users derive a direct benefit in expressiveness by exploring the parameter space and building a mental model of a system. Scientific data visualization is another domain in which users have a vested interest in exploring a large space, so we expect a similar benefit if this approach is applied to classification or regression systems for scientific data.

We further propose that for systems that benefit from

direct user interaction, it may be necessary not only to make implementation decisions with "intuitiveness" in mind (Section 3.1), but also to use the intuitiveness of the parameter space as a metric for evaluating competing learning approaches. The HMM used in MySong, for example, comes with some limitations in terms of generality relative to a more general or higher-order model, but in terms of user experience, the ability to expose relevant parameters in meaningful terms and the fact that those parameters have predictable effects on the system argue heavily in favor of the HMM as an appropriate model for this system. This approach is in some ways in contrast with more traditional approaches for selecting learning systems, which focus on metrics such as classification accuracy and computational performance.

Appendix A: Matrix initialization heuristics

To initialize the major- and minor-mode transition matrices before iterative refinement, we assign a high probability p to all transitions leading to the I, IV, and V chords in the major-mode transition matrix, and the same probability to the vii, III, and ii chords in the minor key transition matrix. We assign the same value to the "song start → I" transition (for P_{maj}) and the "vii → song end" transition (for P_{min}). All other transitions are assigned a low probability ε. We note that these are simply coarse approximations of basic observations from music theory; in practice, this procedure is robust to different values for p and ε, and to other variants on these assumptions that conform to basic definitions of "major" and "minor".

References

Baker, J.K. The DRAGON System - An Overview. IEEE Trans on Acoustics, Speech and Sig Proc, February 1975.

Boersma, P. Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound. Proc Inst Phonetic Sci, v17, 1993.

Brzozowski, M., Carattini, K., Klemmer, S. R., Mihelich, P., Hu, J., Ng, A. Y. groupTime: preference based group scheduling. CHI 2006.

Dredze, M., Lau, T., Kushmerick, N. Automatically classifying emails into activities. IUI 2006.

Fails, J. A. and Olsen, D. R. Interactive machine learning. IUI 2003.

Fogarty, J. and Hudson, S. E. Toolkit support for developing and deploying sensor-based statistical models of human situations. CHI 2007.

Fogarty, J., Hudson, S. E., Lai, J. Examining the robustness of sensor-based statistical models of human interruptibility. CHI 2004.

Fogarty, J., Tan, D., Kapoor, A., Winder, S. CueFlik: Interactive Concept Learning in Image Search. CHI 2008.

Hartmann, B., Abdulla, L., Mittal, M., Klemmer, S. R. Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition. CHI 2007.

Horvitz, E., Koch, P., Apacible, J. BusyBody: Creating and Fielding Personalized Models of the Cost of Interruption. CSCW 2004.

Horvitz, E., Koch, P., Kadie, C., Jacobs, A. Coordinate: Probabilistic Forecasting of Presence and Availability. Proc Conf on Uncertainty and AI, 2002.

Kiritchenko, S. Matwin, S. Email classification with co-training. Proc Conf Centre for Adv Stud on Collaborative Research, 2001.

Kushmerick, N. and Lau, T. Automated email activity management: an unsupervised learning approach. IUI 2005.

Legaspi, R., Hashimoto, Y., Moriyama, K., Kurihara, S., Numao, M. Music compositional intelligence with an affective flavor. IUI 2007.

Linden, G., Smith, B., York, J. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. IEEE Internet Computing 7, 1 (Jan. 2003), p76-80.

McGregor, C. Controlling spam with SpamAssassin. Linux J. 153, Jan 2007.

Nguyen, G., Worring, M., Smeulders, A. Interactive Search by Direct Manipulation of Dissimilarity Space. IEEE Trans Multimedia, 9(7), Nov 2007.

Plamondon, R and Srihari, S. On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey. IEEE Trans Pattern Analysis & Mach Intelligence 22(1), 2000.

Shen, J., Li, L., Dietterich, T. G., Herlocker, J. L. A hybrid learning system for recognizing user tasks from desktop activities and email messages. IUI 2006.

Simon, I., Morris, D., and Basu, S. MySong: Automatic Accompaniment Generation for Vocal Melodies. CHI 2008.

Stumpf, S., Rajaram, V., Li, L., Burnett, M., Dietterich, T., Sullivan, E., Drummond, R., Herlocker, J. Toward harnessing user feedback for machine learning. IUI 2007.

Torres, D., Turnbull, D., Barrington, L., Lanckriet, G. Identifying Words that are Musically Meaningful. Proc Intl Symp on Music Information Retrieval (ISMIR), 2007.

Tóth, Z. A graphical user interface for evolutionary algorithms. Acta Cybern. 16, 2 (Jan. 2003), 337-365.

Tullio, J., Dey, A. K., Chalecki, J., Fogarty, J. How it works: a field study of non-technical users interacting with an intelligent system. CHI 2007.

Turnbull, D., Barrington, L., Torres, D., Lanckriet, G. Semantic Annotation and Retrieval of Music and Sound Effects. IEEE Trans on Audio, Speech, and Language Processing, February 2008 (in press).

Ware, M., Frank, E., Holmes, G., Hall, M., Witten, I. 2002. Interactive machine learning: letting users build classifiers. Intl J Hum.-Compu. Stud. 56, 3 (Mar. 2002), 281-292.

Witten, I. and Frank, E. Data Mining: Practical machine learning tools and techniques, 2nd ed, Morgan Kaufmann, CA, USA, 2005.