Design of a Variational Bayesian system for joint unsupervised learning of a phonetic lexicon and phonological rules

Emily Kellison-Linn, Tim O'Donnell, Elias Stengel-Eskin

1 Introduction

The task of listening to spoken language and dividing it up into a finite set of sound units is one of the central tasks of speech perception. All languages make use of a restricted set of speech sounds, which are combined in a particular order to form words and sentences. Human infants must identify these sounds and learn how to produce them, given only speech audio as input. Therefore, this is in essence an unsupervised learning task. Designing a computational model capable of performing this task offers the potential for learning more about human speech perception, as well as the possibility of learning phonological information about low-resource languages.

Previous work has explored the computational problem of learning a set of phones from unlabeled speech audio. Lee and Glass (2012) and Lee et al. (2013) develop a nonparametric Bayesian system for learning a set of phone-like units from an underlying acoustic signal, by modeling the speech system as an HMM which emits audio samples according to a multivariate Gaussian mixture model. In particular, Ondel et al. (2016) extend this line of research by implementing a similar model and learning the parameters via variational Bayesian inference, which converges more quickly and can be parallelized.

These systems show success at learning a set of phones, but represent speech merely as a sequence of sounds and have have no concept of structured phonological processes. Conversely, some systems have been trained to learn phonological rules, but they operate on strings of phoneme symbols rather than on the underlying speech audio. In short, no existing system learns a phoneme inventory and phonology together.

We propose a system that, given a high level phonemic transcription and corresponding acoustic signal, jointly learns a model of phonological variation and a set of lower-level phonemic units corresponding more closely to the acoustic input. In this report, we present the specification of the generative model, and derive the update equations necessary to implement learning via variational Bayesian inference.

2 Unsupervised learning of phones from acoustic input

Ondel et al. (2016) present a system for learning a set of phone-like units from an acoustic signal, using Variational Bayesian inference techniques. A sequence of phones is modeled as a series of draws from a Dirichlet-distributed multinomial distribution over all phones in the language. Each individual phone is represented as a set of parameters for an n-dimensional Gaussian mixture model, from which individual audio samples of that phone are drawn.

In the generative model, a multinomial distribution over phones is drawn from a Dirichlet distribution. For each phone, a mean and variance is drawn from a Normal-Gamma distribution for each Gaussian component in the mixture model. Then, a phone sequence is generated by repeated draws from the phone distribution, and for each phone, a sequence of beginning, middle, and end states is drawn based on a fixed set of HMM parameters. Finally, for each state in the state sequence, an audio sample is drawn from the corresponding Gaussian mixture model.

Based on this specification, the values over which inference is performed are the following:

- The natural parameters of the Dirichlet distribution over phone frequencies
- The natural parameters of the Dirichlet distribution over Gaussian component choices, for each phone
- The parameters of the Normal-Gamma distribution from which phone means and variances are drawn

Our system extends the Ondel system by learning an editable correspondence between an underlying phone sequence and its realization in speech.

3 Learning phonology

The Ondel system and other similar models are effective at capturing a set of phones, but do not incorporate any concept of phonology. This is a simpler problem than the one faced by human infants, who must simultaneously learn a set of phones and the circumstances under which the realization of these phones differs. We introduce a basic phonological model which can be jointly learned along with the phones themselves.

Broadly, phonology can be thought of as the system of variation between underlying and surface forms of lexical items. Kaplan et al. (1994) show that any phonology based on ordered rules is in the space of finite-space transducers. A particular subset of this finite-state transducer space is that of Levenshtein transducers, which can perform insert, delete, and substitute operations on the input string.

Therefore, we propose to model phonology as a Levenshtein-based noisy channel operation: Underlying phonological forms pass through a 'noisy channel' Levenshtein transducer, which can edit the underlying form by inserting phones, substituting one phone for another, or deleting phones from the sequence. The resulting output is the surface form of the lexical item.

We choose this formulation of phonology and perform a search over this class of transducers – calculating a probability distribution over all possible underlying forms which could have produced a given surface form, given probabilities for all edit operations.

4 Description of the generative model

Summary:

1. Underlying phones given as input
2. Distributions over edit operations drawn from Dirichlet distributions
3. Distributions over Gaussian mixture components drawn from Dirichlet distributions
4. A mean and variance for each Gaussian component in the mixture model is drawn from Normal-Gamma distributions
5. For each phone in the underlying phone sequence, sample an edit operation, producing surface-level phone string
6. From this point the generative process is identical to that in the Ondel model:
    (a) For each phone, a sequence of beginning, middle, and end states is drawn based on a fixed set of HMM parameters
    (b) For each state in the state sequence, an audio sample is drawn from the corresponding Gaussian mixture model.

4.1 Model hyperparameters

We define a set of top-level PLUs U_T, a set of bottom-level PLUs U_B, a number of HMM states per bottom PLU n_h, a number of Gaussian components per HMM state n_g, and a dimensionality for the Gaussian distributions n_d.

4.2 Model parameters

1. Draw distributions E over edit operations conditioned on the next top level PLU q for each top PLU from a Dirichlet distribution with parameters $\boldsymbol{\alpha_q}$:
$$E_q \sim D(\boldsymbol{\alpha_q})\ \forall q \in U_T$$
2. Draw distributions C over Gaussian component selection for each HMM state s for each bottom-level PLU from a Dirichlet distribution with parameters $\boldsymbol{\alpha_s}$:
$$C_s \sim D(\boldsymbol{\alpha_s})\ \forall s \in all\_HMM\_states$$
3. Draw n_d-dimensional Gaussian distributions with mean μ and covariance matrix Σ for each Gaussian component c for each HMM state for each bottom-level PLU from a Normal-Gamma distribution:
$$\mu_c, \Sigma_c \sim NormalGamma(\mu'_c, \lambda_c, \alpha_c, \beta_c)$$

4.3 Generative process

1. Start with a given sequence of of top-level PLUs, $a_1, a_2, ...a_{N_{top}}$, where $a_i \in U_T \forall i$.
2. For i in range $1...N_{top}$:
   (a) Sample an edit operation e from E_{a_i}.
   (b) If $e = insert\_bottom(r)$ for some $r \in U_B$, append r to the list of bottom PLUs.
   (c) if $e = insert\_top(a_i)$, set $i = i + 1$.
   (d) If $e = substitute(a_i, r)$ for some $r \in U_B$, append r to the list of bottom PLUs and set $i = i + 1$.

   The result is a sequence of bottom-level PLUs $b_1, ...b_{N_{bot}}$ where $b_i \in U_B \forall i$.
3. For each bottom PLU b_i in the bottom-level sequence:
   (a) Sample an HMM state sequence $s_1, ...s_{N_{states}}$through b_i with all initial probability on the first state $s = 1$ and transition matrix M, where $m_{x,y} = P(s_{t+1} = y | s_t = x)$ and $|M| = (n_h, n_h)$:
$$M_{x,x} = 0.5$$
$$M_{x,x+1} = 0.5$$
4. For each HMM state s in the HMM state sequence:
   (a) Sample a Gaussian component c from C_s
   (b) Sample an n_d-dimensional vector from the Gaussian distribution with mean μ_c and covariance matrix Σ_c.

5 Variational Bayesian updates

(Derivation of VB update equations for each of the parameters)

6 Detailed description of the chart/parsing-as-deduction system

We accumulate the required counts by conceiving of the model as a system which can transition from state to state based on rules.

6.1 Levenshtein accumulation

We use a method of calculating a distribution over all possible underlying phone sequences that could have generated a given surface phone sequence.

6.2 Conception as a parsing-as-deduction system

6.2.1 Item format

Each item has the following entries. In general, numerical indices are denoted by lowercase letters and other entries are denoted by uppercase letters.

1. frame index $(i, j, ...)$
2. PLU-internal HMM state $(s \in \{start, mid, end\})$
3. PLU bottom type $(A, B, ...)$
4. PLU bottom index $(a, b, ...\ )$
5. edit operation type $(E \in \{NONE, IB, IT, SUB\})$
6. PLU top index $(m, n, ...)$
7. the probability of the item $(P$ or $P')$

We also assume the existence of the following functions:

- $TOP(m)$ returns the type of the PLU at position M in the top-level PLU sequence.
- $p_{hmm}(s_1 \to s_2)$ returns the probability of transitioning from PLU-internal HMM state s_1 to s_2. This is always 0.5 under the current implementation.

- $p_{op}(E)$ returns the probability of the given operation type ($\{IB, IT, SUB\}$).
- $p_{ib}(A)$ returns the probability of the insert bottom operation for PLU A, given that $E = IB$.
- $p_{it}(M)$ returns the probability of the insert top operation for PLU M, given that $E = IT$.
- $p_{sub}(M, A)$ returns the probability of the substitute operation that substitutes PLU A for PLU M, given that $E = SUB$.
- $lh(A, s, i)$ returns the likelihood of state s of PLU A at frame i (based on the audio input).

6.2.2 Moves in Levenshtein matrix (PLU transitions)

Insert Bottom

$$\frac{[i, end, A, a, E, m, P]}{\begin{array}{c}[i+1, start, B, a+1, IB, m,\\ P' = P \cdot p_{hmm}(end \to start) \cdot p_{op}(IB) \cdot p_{ib}(B) \cdot lh(B, start, i+1)]\end{array}}$$

Insert Top

$$\frac{[i, end, A, a, E, m, P]}{\begin{array}{c}[i, end, A, a, IT, m+1,\\ P' = P \cdot p_{op}(IT) \cdot p_{it}(TOP(m+1))]\end{array}}$$

Substitute

$$\frac{[i, end, A, a, E, m, P]}{\begin{array}{c}[i+1, start, B, a+1, SUB, m+1,\\ P' = P \cdot p_{hmm}(end \to start) \cdot p_{op}(SUB) \cdot p_{sub}(TOP(m+1), B) \cdot lh(B, start, i+1)]\end{array}}$$

6.2.3 PLU-internal transitions

HMM-state-internal transition

$$\frac{[i, s, A, a, E \in \{NONE, IB, SUB\}, m, P]}{\begin{array}{c}[i+1, s, A, a, NONE, m,\\ P' = P \cdot p_{hmm}(s \to s) \cdot lh(A, s, i+1)]\end{array}}$$

PLU-internal HMM state transition

$$\frac{[i, s \in \{start, mid\}, A, a, E \in \{NONE, IB, SUB\}, m, P]}{\begin{array}{c}[i+1, s+1, A, a, NONE, m,\\ P' = P \cdot p_{hmm}(s \to s+1) \cdot lh(A, s+1, i+1)]\end{array}}$$

6.2.4 Start items

The start items are as follows:

- { $[\mathbf{i} = \mathbf{0}, start, A, \mathbf{a} = \mathbf{0}, IB, \mathbf{m} = -\mathbf{1}, P = p_{op}(IB) \cdot p_{ib}(A) \cdot lh(A, start, 0)]$ },
  $\forall A \in$ {bottom-level PLUs}
- { $[\mathbf{i} = -\mathbf{1}, end, A, \mathbf{a} = -\mathbf{1}, IT, \mathbf{m} = \mathbf{0}, P = p_{op}(IT) \cdot p_{it}(TOP(0))]$ }
- { $[\mathbf{i} = \mathbf{0}, start, A, \mathbf{a} = \mathbf{0}, SUB, \mathbf{m} = \mathbf{0}, P = p_{op}(SUB) \cdot p_{sub}(TOP(0), A) \cdot lh(A, start, 0)]$ },
  $\forall A \in$ {bottom-level PLUs}

6.2.5 Completion rules

The parse is complete when any item of the following is reached, where x is the number of frames in the audio input and y is the number of PLUs in the top-level sequence.

- { $[\mathbf{i} = \mathbf{n}, end, A, a, E, \mathbf{m} = \mathbf{y}, P]$ }, $\forall A \in$ {bottom-level PLUs}, $1 \le a \le max\_bottom$

6.2.6 Iteration order

Iterating through the items in a correct order, such that all items from which item x is reachable are completed before item x is entered, is nontrivial.

As a starting point for thinking about ordering, below is a list of all items from which item $x = [i, s, A, a, E, m, P]$ is reachable, and thus must be completed before x is entered, given certain conditions on the parameters.

Item	Conditions	Transition type
$[i-1, end, B, a-1, E', m, P']$	if $E = IB, s = start$	Insert-bottom operation
$[i, end, A, a, E', m-1, P']$	if $E = IT, s = end$	Insert-top operation
$[i-1, end, B, a-1, E', m-1, P']$	if $E = SUB$,$s = start$	Substitute operation
$[i-1, s, A, a, NONE, m, P']$	if $E \in \{NONE, IB, SUB\}$	PLU-internal HMM self-transi
$[i-1, s-1, A, a, NONE, m, P']$	if $E \in \{NONE, IB, SUB\}$,$s \in \{mid, end\}$	PLU-internal HMM transition

7 Optimizations

To limit the search space, we introduce some limitations on allowed states. The surface and underlying phone sequences are allowed to differ only by a constant factor of string position; this is necessary to prevent an infinite number of delete operations. In addition, the audio frame sequence is allowed to differ from the bottom string only by a constant factor of string position. This latter requirement is not strictly necessary in order for the model to be well defined; however, it is a practical necessity in order to limit the search space to a reasonable size.

7.1 Time complexity analysis

The unpruned chart, under the 3-state-HMM implementation, has the following dimensionality.

Dimension	Length	Typical value for a 15-second utterance
Number of frames	i	1500
HMM state	3 (constant)	3
Number of PLU bottom types	A	50
Number of PLU bottom indices	a	130
Number of edit operation types	4 (constant)	4
Number of PLU top indices	m	120

Full chart contains $n = O(i \times 3 \times A \times a \times 4 \times m) = O(iAam)$ items. However:

- The number of PLU top indices has an approximately linear relationship with the number of frames (and is upper bounded by it). We can express this as $m = O(i)$.
- Likewise, the number of PLU bottom indices also has an approximately linear relationship with i, so $a = O(i)$.
- Therefore, $n = O(Ai^3)$. In other words, the number of chart items grows as the cube of the number of frames and linearly as the number of PLU bottom types.

Full chart for a typical 15-second utterance contains approximately 1.4×10^{10} items.

8 Implementation

To implement the model, we use Ondel's implementation of variational Gaussian parameter updates given a distribution over bottom level strings, and the implementation of variational Dirichlet parameter updates. However, we replace Ondel's HMM implementation with the chart parsing enumeration over edit operations described above.

9 Conclusion