A Probabilistic Language Model for Hand Drawings

Abdullah Akce
Department of Computer Science
University of Illinois at Urbana-Champaign
aakce2@illinois.edu

Timothy Bretl
Department of Aerospace Engineering
University of Illinois at Urbana-Champaign
tbretl@illinois.edu

Abstract

Probabilistic language models are critical to applications in natural language processing that include speech recognition, optical character recognition, and interfaces for text entry. In this paper, we present a systematic way to learn a similar type of probabilistic language model for hand drawings from a database of existing artwork by representing each stroke as a sequence of symbols. First, we propose a language in which the symbols are circular arcs with length fixed by a scale parameter and with curvature chosen from a fixed low-cardinality set. Then, we apply an algorithm based on dynamic programming to represent each stroke of the drawing as a sequence of symbols from our alphabet. Finally, we learn the probabilistic language model by constructing a Markov model. We compute the entropy of our language in a test set as measured by the expected number of bits required for each symbol. Our language model might be applied in future work to create a drawing interface for noisy and low-bandwidth input devices, for example an electroencephalograph (EEG) that admits one binary command per second. The results indicate that by leveraging our language model, the performance of such an interface would be enhanced by about 20 percent.

1 Introduction

We are motivated by the recent development of interfaces for text entry that allow users to type at a high rate even with a limited, low-bandwidth, and possibly noisy input device. Of particular interest to us are brain-computer interfaces for text entry. One successful example is "Dasher" [9], which has been used with an electroencephalograph (EEG) to type 1-2 words per minute [4]. These interfaces work well because text has two key properties that admit an efficient implementation. First, it is made up of symbols from a finite alphabet that allows users to alphabetize text. Second, it admits a probabilistic language model that allows the interface to perform inference.

In our own work, we are interested in exploring other types of data entry that might be enabled by using an appropriate ordered symbolic language. *In particular, one of our current goals is to enable users with limited input devices to draw pictures*. It is clear that the performance of such an interface is proportional to the entropy of its language, i.e., the expected number of bits required to represent the data with respect to a language model. For instance, the entropy of English text containing 26 letters and the space character with respect to a uniform model is $\log(27) = 4.76$ bits per character. It can be reduced to 4.03 by using a unigram model, to 2.8 by using a bigram model, and to about 2.0 by using prediction by partial matching [7, 2].

In this paper, we propose a probabilistic language model for hand drawings and perform an analysis of its entropy. Our language, which we describe in Section 2, assumes that drawings are composed of smooth planar curves that we call *strokes*. Instead of characters, the symbols in our alphabet are circular arcs with length fixed by a scale parameter and with curvature chosen from a fixed low-cardinality set. In Section 3, we give an algorithm based on dynamic programming to approximate strokes by sequences of these symbols. In Section 4, we describe how we construct a probabilistic model from the resulting sequences. Finally, in Section 5, we present the performance of our language model by computing its entropy in two different sets of drawings.

2 A language for smooth planar curves

Any smooth two-dimensional curve $\gamma\colon [0, L] \to \mathbb{R}$ of arbitrary length L can be described by $x' = \cos\theta$, $y' = \sin\theta$, $\theta' = -\kappa$, where s is the arc-

This research was supported by awards NSF-CNS-0931871 and NSF-CMMI-0956362-EAGER.

1051-4651/10 $26.00 © 2010 IEEE
DOI 10.1109/ICPR.2010.35

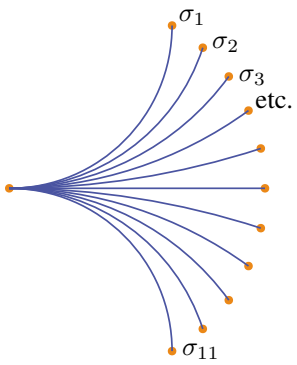

Figure 1. An example alphabet used in our language of smooth curves.

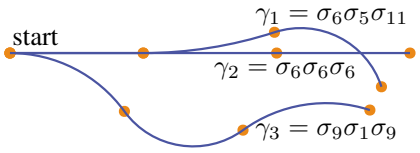

Figure 2. Three smooth curves composed from the symbols of Figure 1.

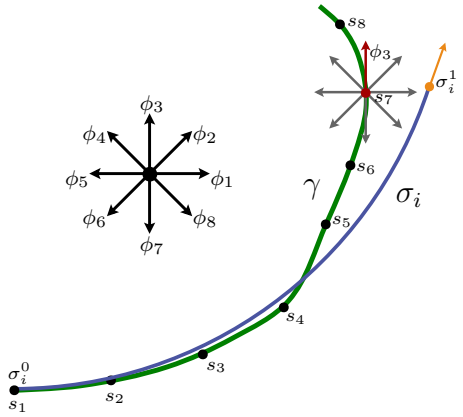

Figure 3. Projection to a point and orientation on the curve γ.

length, $\theta(s)$ is the angle of the tangent to the curve at $(x(s), y(s))$, and $\kappa(s)$ is the curvature. The curve is straight when $\kappa = 0$, turns left when $\kappa < 0$, and turns right when $\kappa > 0$. We can think of curvature as the input to these ordinary differential equations. Given $\kappa(s)$ and the initial conditions $x(0)$, $y(0)$, and $\theta(0)$, we can integrate to find $x(s)$, $y(s)$, and $\theta(s)$.

Our language models a subset of curves for which κ is piecewise constant on intervals of length d. We further restrict these curves by choosing each $\kappa_i \in \{c_1, \ldots, c_m\}$ from a finite set. We associate a symbol σ_i with the arc generated by κ_i. We define an alphabet $\Sigma = (\sigma_1, \ldots, \sigma_m)$ so that curves can be described concisely as strings of symbols from Σ. The language we propose is most closely related to the idea of a *chain code* in the literature [5]. For a chain code, the alphabet contains eight cardinal directions that describe how to move from one pixel to the next along the contour of a shape. Our own language model for hand drawings (Section 2) uses curvatures rather than cardinal directions as symbols, and is intended to generate smooth curves rather than polygonal chains of pixels.

The sequential structure of our language allows a user to "spell" curves, one symbol at a time, just like he or she would spell text. Our language also admits an intuitive lexicographic ordering. For two arcs $\gamma_i, \gamma_j \in \Sigma$, we say that $\gamma_i < \gamma_j$ if and only if γ_i turns left at the first point at which it differs from γ_j. This ordering allows a user to "alphabetize" curves just like he or she would alphabetize strings of text. It also allows the user to search for curves with a binary input, just like he or she would search for words in a dictionary by turning pages forward or back. We employed this property in a brain-machine interface to allow a human pilot to remotely teleoperate an unmanned aircraft by specifying the desired path using only binary input [1].

3 From strokes to words

Given a stroke of a drawing represented as a two-dimensional curve $\gamma\colon [0, L] \to \mathbb{R}$ parameterized by arc-length s, our objective is to find the sequence of symbols constituting the word $w \in \Sigma^*$ that best approximates γ with respect to a dissimilarity cost. This objective is quite different than approximating a digital curve with circular arcs as in [8, 6]. First, we require the curvature of each arc to come from a finite alphabet. Second, we enforce a smoothness constraint and require that the consecutive arcs share the same tangent angle. Third, endpoints of arcs do not generally lie on the curve because we require all arcs to have the same length.

We compute the dissimilarity between the curve γ and a word w by first projecting the endpoints of the symbols in w to points and orientations on γ. Let $S = \{s_1, s_2, \ldots, s_M\}$ be a set of arc-lengths identifying the locations of the projection points, and $\Phi = \{\phi_1, \phi_2, \ldots, \phi_T\}$ be a set of quantized orientations. $proj(q) \in S \times \Phi$ maps a configuration $q = (x, y, \theta)$ to the arc-length of its nearest projection point and to the orientation $\phi \in \Phi$ closest to θ. For ex. in Figure 3, the endpoint σ_i^1 of a symbol σ_i is mapped to (s_7, ϕ_3).

The dissimilarity cost is then given by

$$E(\gamma, w = \sigma_1 \cdots \sigma_N) = \sum_{i=1}^{N} e(\sigma_i, \gamma_{\sigma_i}),$$

where γ_{σ_i} is the portion of the curve γ from $proj(\sigma_i^0)$ to $proj(\sigma_i^1)$ and $e(\gamma_1, \gamma_2)$ is the sum-squared distance between two curves γ_1 and γ_2. Our algorithm attempts to minimize this cost using dynamic programming. Let $TC(s, \phi)$ be the total cost of the best approximation to γ from the start to a projection point at $s \in S$ with quantized orientation $\phi \in \Phi$. We assume that the following recursive relation holds:

$$TC(s, \phi) \approx \min_{\substack{s' \in S, \phi' \in S \\ s' < s}} \left\{ TC(s', \phi') + SC(s', \phi', s, \phi) \right\}$$

where the step cost $SC(s', \phi', s, \phi)$ is the minimum cost of approximating the portion of γ from (s', ϕ') to (s, ϕ) by appending a single circular arc $\sigma \in \Sigma$ to the best approximating sequence for (s', ϕ'). More precisely, the step cost is given by

$$\min_{\substack{\sigma \in \Sigma \\ \sigma^0 = q^1(s', \phi')}} \begin{cases} e(\sigma, \gamma_\sigma) & \text{if } proj(\sigma^1) = (s, \phi) \\ \inf & \text{otherwise} \end{cases}$$

where $q^1(s', \phi')$ is the endpoint of the best approximating sequence for (s', ϕ'). The complexity of this algorithm when implemented iteratively is $O(|\Sigma||S||\Phi|)$.

4 Learning the language model

Our language allows us to specify a stroke of a drawing by a word $w \in \Sigma^*$ and an initial configuration $q = (x, y, \theta)$. We introduce a special symbol σ_f to mark the end of each word. Given a sequence of strokes obtained from a drawing, we represent it by a sequence $W \in (\Sigma \cup \{\sigma_f\})^*$ discarding the initial configurations of each stroke. We can think of this sequence as being generated by a Markov process. From training data, we can compute a Markov model that assigns a conditional probability to each symbol. A k-th order Markov model takes into account only the immediate k previous symbols. When $k = 0$, this is called a unigram model and when $k = 1$, it is called a bigram model. We also consider a variable order model in which the order is adjusted based on the available statistics. In particular, we use prediction by partial matching (PPM) [3] which is an effective method for compressing text.

We evaluate the performance of a language model by computing its cross-entropy on a test set. Given a language model P_m and a sequence W representing a drawing, the cross-entropy is $\frac{1}{|W|} \sum_{i=1}^{|W|} -\log P_m(W_i)$ which can be interpreted as the expected number of bits required to represent a symbol of W.

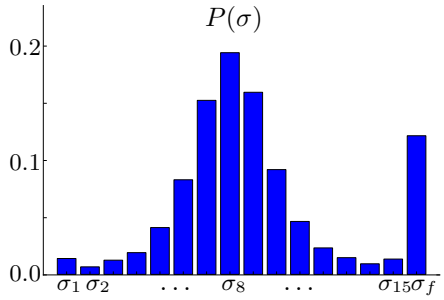

Figure 5. The unigram model obtained from the "smurfs" dataset.

5 Results

We experimented with two datasets: "smurfs" and "incredibles", each containing 15 drawing images. The strokes in each image were obtained by tracing the contours of the drawing using a graphics tablet. On average, there were about 66 strokes in a "smurfs" drawing, and about 94 strokes in an "incredibles" drawing. We used an alphabet consisting of 15 circular arcs with evenly spaced curvatures to approximate the strokes of the drawings. The length of each arc was set to one eightieth of the width plus height of the drawing image. The results of applying our dynamic programming based algorithm on two of the drawings are shown in Figure 4.

We obtained three language models: unigram, bigram and 5-order PPM from a set of images reserved for training and evaluated the performance of these models in the test set by doing 5-folds cross validation. The performance results as measured by cross-entropy are shown in Table 1. The unigram model obtained from "smurfs" dataset is shown in Figure 5 (the unigram model for "incredibles" was similar). The unigram model suggests that the prior distribution of arcs are symmetric around the zero curvature corresponding to a line segment. This is interesting because it supports the common assumption about curvature that is being made by most previous work on stroke extraction. The performance of the bigram model was only slightly better than the unigram model and the performance of the higher-order PPM model was worse. That is in contrast to the language models for text where a bigram model is about 30 percent better than a unigram model and a higher-order PPM model can provide up to 30 percent improvement over a bigram model. Our results indicate that by using a bigram model, the performance of an interface for drawing can be enhanced by about 20 percent over an interface not using any language model.

original in our language original in our language

Figure 4. Two drawings and their representation in our language.

Dataset	Language model	Entropy
smurfs	none (uniform)	4
smurfs	unigram	3.36
smurfs	bigram	3.30
smurfs	5-order PPM	3.60
incredibles	none (uniform)	4
incredibles	unigram	3.23
incredibles	bigram	3.12
incredibles	5-order PPM	3.45

Table 1. The entropy of hand drawings.

6 Conclusion

One of our current goals is to enable users with input devices that are limited, low-bandwidth and possibly noisy to draw pictures just as easily as they can write text. Toward this end in this paper, we presented a systematic way to learn a probabilistic language model for hand drawings. The language we chose consisted of symbols that represented circular arcs. We presented a dynamic programming based algorithm to approximate each stroke as a sequence of symbols. We computed the expected number of bits to represent a symbol with respect to n-gram models learned from two different image sets. In these datasets, higher-order models gave only a small or no improvement over a unigram model. It was interesting to see that even with a low-order model computed from a small set of data, we were able to get about 20 percent improvement. In future work, it would be interesting to see how the model performance changes for different and larger datasets. We also want to characterize the influence of the symbol scale and the alphabet size on the results. Applications of our language model to writer identification and hand gesture recognition as well as to the task level control of mobile vehicles such as wheel chairs are also of interest.

References

[1] A. Akce, M. Johnson, and T. Bretl. Remote teleoperation of an unmanned aircraft with a brain-machine interface: Theory and preliminary results. In *Robot. and Autom. Proceedings. IEEE Int. Conf. on*, May 2010.

[2] R. Begleiter, R. El-Yaniv, and G. Yona. On prediction using variable order markov models. *Journal of Artificial Intelligence Research (JAIR)*, 22:385–421, 2004.

[3] J. Cleary and I. Witten. Data compression using adaptive coding and partial string matching. *Communications, IEEE Transactions on*, 32:396–402, 1984.

[4] E. Felton, N. L. Lewis, S. A. Wills, R. G. Radwin, and J. C. Williams. Neural signal based control of the Dasher writing system. *IEEE EMBS Conf. Neural Engr.*, pages 366–370, 2007.

[5] H. Freeman. On the encoding of arbitrary geometric configurations. *Electronic Computers, IRE Trans. on*, 10:260–268, 1961.

[6] J. H. Horng and J. T. Li. A dynamic programming approach for fitting digital planar curves with line segments and circular arcs. *Pattern Recognition Letters*, 22:183–197, 2001.

[7] C. Manning and H. Schütze. *Foundations of statistical natural language processing*. MIT Press, 2002.

[8] S. C. Pei and J. H. Horng. Optimum approximation of digital planar curves using circular arcs. *Pattern Recognition*, 29:383–388, 1996.

[9] D. J. Ward, A. F. Blackwell, and D. J. C. MacKay. Dasher: A gesture-driven data entry interface for mobile computing. *Human-Computer Interaction*, 17:199–228, 2002.