Word Normalization for On-Line Handwritten Word Recognition

Yoshua Bengio
Dept. Informatique et Recherche
Opérationelle, Université de Montréal
Montreal, Qc H3C-3J7, Canada
bengioy@iro.umontreal.ca

Yann Le Cun
AT&T Bell Laboratories
Holmdel, NJ 07733
U.S.A.
yann@research.att.com

Abstract

We introduce a new approach to normalizing words written with an electronic stylus that applies to all styles of handwriting (upper case, lower case, printed, cursive, or mixed). A geometrical model of the word spatial structure is fitted to the pen trajectory using the EM algorithm. The fitting process maximizes the likelihood of the trajectory given the model and a set a priors on its parameters. The method was evaluated and integrated to a recognition system that combines neural networks and hidden Markov models.

1 Introduction

Natural handwriting can be a mixture of different "styles", lower case printed, upper case, cursive, and punctuation. In order to improve the success of pen-based computers, we would like a recognizer that reliably handles such handwriting, but its implementation faces major technical challenges [9]. It has been long known that, although characters taken in isolation can be very ambiguous, considerable information is available from the context of the whole word. Many recognition systems operate uniquely on the *shape* of each candidate character, and delay the use of contextual information such as the position relative to the baseline or the height, until the word-level postprocessor. However, some recent recognition methods, most notably Neural Networks, have been shown to yield better results when word-level normalization is performed instead of individual character normalization. This is particularly true when scores from the recognizer are used in an integrated segmentation/recognition system. It is even more the case with so-called "Space Displacement Neural Networks" (SDNN) that take an entire unsegmented word as input [7].

Many recognition methods, particularly the ones that use pictorial representations of the characters, require that the size, position, and orientation of objects be approximately constant, at least within a single class. Word-level normalization can achieve this by identifying the location of the baseline of a word, and the approximate height (or, *word core height*) of lower case characters without ascenders or descenders such as "o". We propose a new word normalization scheme, based on fitting a geometric model of the word structure. Our model comprises four "flexible" lines representing respectively the ascender line, the core line (i.e. the line that joins the tops of small lower-case characters), the baseline and the descender line (see Figure 1). Once the model has been fitted to the data, the pen trajectories can be normalized (scaled, rotated, translated) according to the expectations of a character or word recognizer.

This important preprocessing step has been integrated in a word recognition system described in a companion paper [3]. This word recognition system for pen-based devices is based on four main modules: the word normalization preprocessor described in this paper; a module that produces an "annotated image" from the normalized pen trajectory; a replicated convolutional neural network that spots and recognizes characters; and a Hidden Markov Model (HMM) that interprets the network output by taking word-level constraints into account. The network and the HMM are *jointly* trained to minimize an error measure defined at the word level.

The word normalization preprocessor has been evaluated by comparing the performance of the above system with a character-level normalization. Writer-independent tests performed on a database of lower case words showed large reductions in error rates when using the word normalization preprocessor.

2 Word Model

Several authors have advocated the use geometrical models for locating, recognizing, or performing measurements on objects in images. A widely used paradigm is the so called "elastic matching" method. The main idea is to come up with an "energy" function composed of two terms. The first term represents the quality of the fit between the model and the data, while the second term measures the internal "stress" in the model, that is, how much the model must be distorted to fit the data. Various search techniques can be used to minimize the energy function (gradient descent, simulated annealing, combinatorial techniques), but formulating the problem in a probabilistic framework allows us to use the fast and powerful Expectation-Maximization algorithm (EM) [2]. In the probabilistic framework, the model is viewed as *generating* the data through a stochastic process. The energy function is interpreted as the negative logarithm of the likelihood that the data be generated by the model.

Our geometrical model of the word structure is based on four lines that must respectively go through the ascenders (top of tall characters, like "E" or "l"), the top of the word core (top of lower case characters such as "o"), the baseline points, and the descenders (bottom of characters like "p" and "q"). The fitting energy reflects how close the bottom two lines are to the local minima of the vertical displacement, and how close the top two lines are to the local maxima of the vertical displacement. The internal "stress" term of the energy can be understood with a mechanical analogy. Each line behaves like a flexible rod with a given stiffness. The rods are linked with each other with "springs" that have given spring constants and rest lengths. The rods are constrained to be parallel, and are all kept nearly horizontal with another "spring". The energy of a spring is proportional to the square of the difference between its length and its rest length. In the equivalent probabilistic framework, the exponential of the negative of the above energy corresponds to a prior probability distribution on the parameters of the model (i.e. the probability distribution of the parameters in the absence of data). Because of the quadratic form of the energy, these distributions are Gaussians whose means are the rest lengths of the springs, and whose variances are the inverse of the spring constants.

Formally, the lines are parameterized as second degree polynomials of the following form:

$$\mathbf{y} = f_k(\mathbf{x}) = \kappa(\mathbf{x} - x_0)^2 + s(\mathbf{x} - x_0) + y_{0k} \qquad (1)$$

where κ controls curvature, s is the slant, and (x_0,y_0) is a translation vector. The parameters κ, s, and x_0 are shared among all four curves, whereas each curve has its own vertical translation parameter y_{0k}. The reference abscissa x_0 is computed as the horizontal center of mass of the set of extrema points. The parameters of the model (i.e., of the four lines) are collectively referred to as the vector θ.

The input to this preprocessor is the pen trajectory, i.e. a sequence of spatio-temporal coordinates $\{(x, y, t)\}$ produced when the pen touches the tablet. After simple preprocessing (resampling, smoothing), the local minima and maxima of the vertical position are extracted. These points are used as the *observations* that the model will fit.

Let U be the set of local maxima (upper extremes), and L be the set of local minima (lower extremes). The likelihood that a particular maximum point (resp. minimum point) (x_i, y_i) be generated by the model with parameter vector θ is computed from the square distances between that point and the top two curves (resp. bottom two curves). Now, since each point can be assigned to one of two lines, there is an ambiguity in choosing what line to compute the distance from. The probabilistic framework provides us with an elegant solution to this problem by modelling the likelihood of a maximum point (resp minimum point) as a *mixture of two Gaussian distributions* whose means are the ordinates of the top two lines (resp. bottom two lines) at the corresponding abscissa. In addition, the points are assigned a small probability to be generated by a diffuse "background model" $\mathcal{B}$ that is meant to account for outliers.

More formally, the likelihood for a minimum point is:

$$\begin{aligned} P(x_i, y_i \mid p_i \in L; \theta) = \quad & w_0 N_0 e^{S_0(y_i - f_0(x_i))^2} \\ & + w_1 N_1 e^{S_1(y_i - f_1(x_i))^2} \\ & + w_4 P(x_i, y_i | \mathcal{B}) \qquad (2) \end{aligned}$$

and the likelihood for a maximum point is

$$\begin{aligned} P(x_i, y_i \mid p_i \in U; \theta) = \quad & w_2 N_2 e^{S_2(y_i - f_2(x_i))^2} \\ & + w_3 N_3 e^{S_3(y_i - f_3(x_i))^2} \\ & + w_4 P(x_i, y_i | \mathcal{B}) \qquad (3) \end{aligned}$$

where the N_k are the appropriate normalization constants for the Gaussians; $f_k(x_i)$ is the ordinate of line k at abscissa x_i as given by equation (1). The inverse variances of the Gaussian distribution S_0, S_1, S_2, S_3 control the "strength" of the force that pulls the points and the lines together. The w_k are mixture parameters which control the prior probability of assigning a point to one of the four lines or to the background model $\mathcal{B}$.

The background model prevents points that are very far from all four lines (outliers) to have too much of an influence on the result of the fit. We chose the background model to be a uniform distribution over the range of accessible coordinates (i.e. $P(x_i, y_i|\mathcal{B})$ is a constant P_b). The first four mixing coefficients w_0, w_1, w_2, w_3 were pre-computed from the frequency of association of extrema to curves measured on a large corpus of words.

The parameters to be estimated for the fit, collectively denoted by θ, are those defined in (1), i.e., global curvature, slant, and baseline position, and relative vertical position of the three other lines with respect to the baseline.

Let $X \stackrel{\text{def}}{=} U \cup L$ be the set of observed extrema points. Given a point $p_i = (x_i, y_i, t_i)$ from X, we write

$$P(p_i \mid \theta) = \begin{cases} P(x_i, y_i \mid p_i \in L; \theta) & \text{if } p_i \in L \\ P(x_i, y_i \mid p_i \in U; \theta) & \text{if } p_i \in U \end{cases} \quad (4)$$

3 Model Fitting with the EM Algorithm

Given an input word, our goal is to find the parameter vector θ that is the most likely according to our probabilistic model. In other words we want to find the vector θ^* that maximizes

$$\theta^* \quad = \operatorname*{argmax}_{\theta} P(\theta \mid X) \quad (5)$$

Since X is fixed while we allow θ to vary, this can be shown to be equivalent to finding the maximum of the joint probability of θ and X:

$$\theta^* = \operatorname*{argmax}_{\theta} P(\theta, X) = \operatorname*{argmax}_{\theta} P(X \mid \theta) P(\theta) \quad (6)$$

where $P(\theta)$ is an a priori probability distribution on the parameters, the negative logarithm of which is the "internal stress" term of the energy. Appropriate choice for these prior probabilities is crucial to the success of the method. As stated above, they incorporate the internal geometrical constraints of the model. For example, we know a priori that words are mostly straight and horizontal. We also know that the lines should be approximately equally spaced and should not be on top of each other. In addition, we may also have a good a priori estimate of the baseline position or of the core height. All of these can be modelled as simple independent Gaussian prior distributions on the parameters in θ (quadratic, spring-like energy terms).

$$P(\theta) = N \prod_j e^{K_j(\theta_j - \Theta_j)^2} \quad (7)$$

where Θ_j is the correct a priori value for θ_j (i.e., the rest length of the springs in the mechanical metaphor); K_j is the inverse variance of the Gaussian controlling the strength of the prior (i.e., the spring constant); and N is an appropriate normalization constant.

The energy function to be minimized is the negative log of the joint probability:

$$C(\theta) \quad = -log P(X \mid \theta) - log P(\theta) \quad (8)$$

As expected, the first term in the right hand side measures the quality of fit, while the second term measures the internal stress. According to our model, the first term is given by a sum over all extrema points of the logarithm of a mixture of Gaussians, while the second term is simply a sum of quadratic terms.

Because of the mixture, no simple, direct method exists to find the minimum of this function. In fact, the function is likely to be non-convex and have local minima. Although minimizing this function can be done with standard techniques such as gradient descent or conjugate gradient, the probabilistic formulation allows us to use the more efficient EM algorithm [2].

The idea of EM in this context is to express C as a function $C(\theta, Z)$ of θ and a of set of "hidden" random variables Z whose distribution depends on θ. Let $\bar{C}(\theta)$ represent the expected value of $C(\theta, Z)$ over Z. The hidden variables are chosen in such a way that, if their distribution is kept constant, the minimization of $\bar{C}(\theta)$ reduces to a trivial (or simple) problem. Here, the hidden variables are discrete variables that determine the assignment of an extremum point to one of the elements of the mixture: for example, $z_i = 2$ means that point i is accounted for by line 2, $z_i = 4$ means that it is accounted for by the background model. As stated above, the distribution of Z really depend on θ, but if it is held constant, the expected value of $\bar{C}(\theta)$ is quadratic with respect to θ. The basic mechanism of EM is, at iteration k, to compute the distribution of Z given the current parameter vector $\theta^{(k)}$ (Expectation step), then to find the new parameter vector $\theta^{(k+1)}$ which minimizes the expected value $\bar{C}(\theta)$, while keeping the distribution on Z constant (Maximization step), and to iterate until convergence. Since the expected value $\bar{C}(\theta)$ is quadratic, the second step involves solving a simple linear system.

In our experiments, convergence was always very quick. At most four iterations were required to obtain a good estimate of the parameters.

4 Preliminary Preprocessing

To help fitting the model to the observations, preliminary preprocessing was performed to remove gross variabilities from the extrema points. The whole sequence of operations performed to extract the normalization information from the trajectory is sketched by the following:

1. The pen trajectory is dehooked, smoothed, and rotated to an approximately horizontal orientation using simple projection techniques.

2. An approximate estimate of the word scale is computed.

3. Maxima and minima of vertical displacement are extracted.

4. The priors on the model parameters are either set from an application dependent directive or set according to the results of steps 1 and 2.

5. The model parameters are estimated with four iterations or less of the EM algorithm described in the previous section.

6. These parameters can be used to normalize the original pen trajectory with respect to the word model.

5 Experiments on the Word Recognizer

The handwriting recognition system described in [3] is based on a spatial representation, convolutional neural networks, and a post-processor that integrates lexical and grammatical constraints. After the word normalization preprocessing, the input trajectory is transformed into a representation scheme called AMAP, which is a low-resolution image in which each picture element contains information about the local properties of the trajectory (e.g. orientation, curvature). This representation is particularly well suited for use in combination with Multi-Layer Convolutional Neural Networks (MCLNN) [6, 4]. These networks are

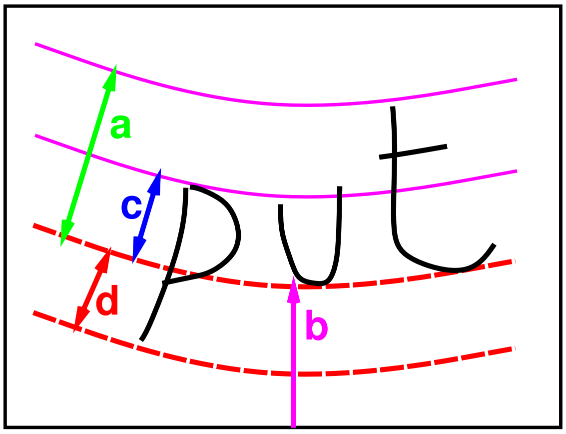

Figure 1: Word Normalization Model: *y-maxima* are fit to the ascenders and core curves, whereas *y-minima* are fit to the descenders and baseline curves. There are 6 parameters: **a** (ascenders curve height relative to baseline), **b** (baseline absolute vertical position), **c** (core line position), **d** (descenders curve position), κ (curvature), **s** (angle).

feedforward and their architecture is tailored to minimizing sensitivity to translations, rotations or distortions of the input image. They are trained with a variation of the Back-Propagation algorithm [8, 5]. The neural network gives scores associated to characters when the network has an input field, called segment, that covers a connected subset of the whole word input. A segmentation is a sequence of such segments that covers the whole word input. Hidden Markov Models and dynamic programming are used to model constraints at the word level and search for the best segmentation.

The experiments described here concern the recognition of lower case words independently of the writer. The tests were performed on a database of 881 words. We evaluated the improvements brought by the word normalization to a word recognition system that is based on optimally combining a large number of character candidates in order to form a word hypothesis. With the word recognition system described in a companion paper [3], and before doing any word-level training, we obtained without word normalization (using character-level normalization) 7.3% and 3.5% word and character errors (adding insertions, deletions and substitutions) when the search was constrained within a 25461-word dictionary. When using the word normalization preprocessing instead of a character level normalization, error rates dropped to 4.6% and 2.0% for word and character errors respectively, i.e., a relative drop of 37% and 43% in word and character error respectively.

Further improvements were obtained by training at the word level (as in [1]) both the neural network recognizer and the post-processor. With the 25461-word dictionary, errors dropped to 3.2% and 1.4% word and character errors respectively.

6 Conclusion

We have demonstrated a new approach to handwritten word normalization that fits a geometrical model of word structure to the pen trajectory with the EM algorithm. Tests of this new algorithm with a neural network based word recognizer yielded large reductions in error rates.

References

[1] Y. Bengio, R. De Mori, G. Flammia, and R. Kompe. Global optimization of a neural network-hidden Markov model hybrid. *IEEE Transactions on Neural Networks*, 3(2):252–259, 1992.

[2] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society B*, 39:1–38, 1977.

[3] Y. Le Cun and Y. Bengio. Word-level training of a handritten word recognizer based on convolutional neural networks. In IEEE, editor, *ICPR'94*, Jerusalem 1994, 1994.

[4] Y. Le Cun, Y. Matan, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, and H.S. Baird. Handwritten zip code recognition with multilayer networks. In IAPR, editor, *International Conference on Pattern Recognition*, Atlantic City, 1990. IEEE.

[5] Y. LeCun. Learning processes in an asymmetric threshold network. In E. Bienenstock, F. Fogelman-Soulié, and G. Weisbuch, editors, *Disordered Systems and Biological Organization*, pages 233–240. Springer-Verlag, Berlin, Les Houches 1985, 1986.

[6] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.

[7] O. Matan, C.J.C. Burges, Y. LeCun, and J.S. Denker. Multi-digit recognition using a space displacement neural network. In J.E. Moody, S.J. Hanson, and R.P. Lipmann, editors, *Advances in Neural Information Processing Systems 4*, pages 488–495, San Mateo CA, 1992. Morgan Kaufmann.

[8] D.E. Rumelhart, J.L. McClelland, and the PDP Research Group. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1. MIT Press, Cambridge, 1986.

[9] C. Tappert, C. Suen, and T. Wakahara. The state of the art in on-line handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(12), 1990.