EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2007)
M. van de Panne, E. Saund (Editors)

Addressing Class Distribution Issues of the Drawing vs Writing Classification in an Ink Stroke Sequence

Xin Wang and Manoj Biswas and Sashi Raghupathy

Ink Parsing Team, Tablet PC, Microsoft Corp.,
One Microsoft Way, Redmond, WA 98052, USA

Abstract

Complicated by temporal correlations among the strokes and varying distributions of the underlying classes, the drawing/writing classification of ink strokes in a digital ink file poses interesting challenges.

In this paper, we present our efforts in addressing some of the issues. First, we describe how we adjust the outputs of the neural network to a priori probabilities of new observations to produce more accurate estimates of the posterior probabilities. Second, we describe how to adapt the parameters of the HMM to new data sets. Albeit the fact that the emission probabilities of the HMM are computed indirectly from the outputs of the neural network, our modified Baum-Welch algorithm still finds the correct estimates for the HMM's parameters.

We also present experimental results of our new algorithms on 6 real–world data sets. The results show that our methods increase the F–Measures of both the drawing and the writing classes on the more "drawing–intensive" data sets which have stronger temporal correlations. But they do not perform well on the more "writing–intensive" data sets.

1. Introduction

A digital ink file is a sequence of ink strokes. Classifying the ink strokes into either drawing or writing class is one of the fundamental problems of the analysis of free–form ink files. At first look, this is a classical binary classification problem, and should be easy to solve, but in practice, complicated by strong temporal correlations among the ink strokes and problems with the underlying class distriutions, the problem poses interesting challenges.

To solve the drawing/writing classification prolem, Bishop et al [BSH04] proposed a method that first uses a neural network (NN) to produce posterior probabilities $p(d_i|x_i)$ for each stroke x_i. Then the probabilities are used as the emission probabilities to an HMM to exploit the temporal correlations among successive ink strokes. In the last step, the method uses the Viterbi Algorithm to find the most probable sequence of stroke labels for the current ink sequence.

This NN+HMM method requires that the priors of the two classes and the transition probabilities from one stroke to the next be fixed and known aforehand. But in practice, as shown in this paper, the priors of the two classes often vary among different types of ink file. Even for the same type of file, the distribution could vary from one file to another. The same is true for the transition probabilities.

This paper presents our efforts in addressing these issues with the underlying class distributions. Following the example of Saerens et al [SLD02], we describe how to adjust the outputs of the neural network to a new file to produce more accurate estimates of the posterior probabilities, $p(d_i|x_i)$. Then we present an EM algorithm to compute the prior and transition probabilities parameters of the HMM. For our problem, the emission probabilities are indirectly given by the neural network and only partially known. We prove that we can modify the Baum-Welch algorithm to use the outputs of a neural network, and the EM algorithm still produces the correct estimates of the parameters for the HMM. But the modification could introduce a numerical problem in some extreme cases. We describe a simple "trick" that can be used to avoid the numerical overflow problem and to make sure the EM algorithm still converges to the same estimates of the parameters.

In section 6, we present experimental results on 6 real–world data sets of ink files. Each data set consists of files

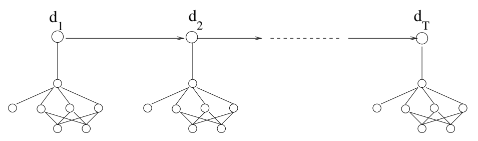


Figure 1: *The outputs of the neural network used indirectly as observation probabilities to the HMM.*

from a unique file type, with its own priors and transition probabilities as shown in Table 3. The results show that the NN+HMM method presented by [BSH04] increases the precision at the cost of seriously reducing the recall of the minority class. In our problem, the drawing class is the minority class (see Table 3). In all except one case, the NN+HMM decreases the F–Measures of both the majority and the minority class.

In comparison, our methods increase the F–Measures of both the drawing and the writing classes on the more "drawing–intensive" data sets which have stronger temporal correlations and relatively lower recall of the drawing class. But they do not perform well on the more "writing–intensive" data sets which have weaker temporal correlations.

2. Related Work

For the drawing/writing classification of ink strokes in a digital ink file, Bishop, et al [BSH04] proposed a method that uses the outputs of a multi-layer peceptron [Bis95] as the observations of an HMM to find the most probable sequence of stroke labels. First, they trained the neural network with the cross–entropy function:

$$E = -\sum_{n=1}^{N}\Big(d_n \ln y_n + (1-d_n)\ln(1-y_n)\Big) \tag{1}$$

as the objective function. In Eq 1, y_n is the output of the neural net for the nth training example, $n = 1\ldots N$. $d_n = 1$ when the nth training example is a drawing stroke; otherwise $d_n = 0$.

As pointed out by Richard et al [MR91], a multi-layer peceptron produces probabilistic predictions when trained in this manner—the outputs of the neural net, y_n, is an estimate of the posterior probability of $p(d_n|x_n)$.

The advantage of training the neural network to output probabilities is that they can later be used in a probabilistic framework to make classifications that involve more global context. For example, in the drawing/writing stroke classification problem, there is often strong temporal correlation among successive strokes. Bishop et al proposed to exploit this sequential relationship with a Hidden Markov Model(HMM) [Rab90]. They represent the entire sequence of strokes with the following HMM:

$$p(x_1,x_2,\ldots,x_T,d_1,d_2,\ldots,d_T) = p(d_1)\prod_{t=1}^{T-1} p(d_{t+1}|d_t)\prod_{t=1}^{T} p(x_t|d_t),$$

where T denotes the length of the ink sequence.

With the outputs of the NN, $p(d_t|x_t)$, we can compute the emission probabilities of the HMM, $p(x_t|d_t)$, indirectly with the Bayes' theorem. The likelihood of the entire sequence can then be computed as

$$p(x_1,x_2,\ldots,x_T,d_1,d_2,\ldots,d_T) = p(d_1)\prod_{t=1}^{T-1} p(d_{t+1}|d_t)\prod_{t=1}^{T} \frac{p(d_t|x_t)p(x_t)}{p(d_t)}. \tag{2}$$

If we can estimate the priors $p(d_t)$ and the transition probabilities $p(d_{t+1}|d_t)$ of the HMM from the training data, then we can find the most probable sequence of stroke labels by solving the following optimization problem with the Viterbi Algorithm:

$$\arg\max_{d_1,\ldots,d_n} p(d_1)\prod_{t=1}^{T-1} p(d_{t+1}|d_t)\prod_{t=1}^{T} \frac{y_t}{p(d_t)}.$$

The term $p(x_t)$ introduced by the Bayes' theorem relates only to the observation sequence, and can be omitted.

3. Practical Issues of Varying Class Distribution

Like many real world applications [CJK04], the drawing/writing classification problem is also plagued by issues related to the underlying class distribution.

The first issue is related to the data imbalance problem [CJK04, PF01, JS02]. In our case, for most digital ink files, the underlying class distribution is often strongly biased towards writing. For example, writing strokes are usually smaller and more cursive, and drawing strokes are usually bigger and less curvy. So, intuitively, a digital ink file can have more writing strokes than drawing strokes even if there appears to be an equal amount of each in the file. For some of our data sets, the prior of the minority class can be as small as 0.012 or 0.02 (see Table 3).

As reported by Japkowicz et al [JS02], multi-layer perceptrons also suffer from the class imbalance problem. Bishop et al [BSH04] proposed to address the problem by scaling the cross–entropy objective function with the estimated priors of the two classes:

$$\tilde{E} = -\sum_{n=1}^{N}\left(\frac{1}{\pi_d} d_n \ln y_n + \frac{1}{\pi_w}(1-d_n)\ln(1-y_n)\right), \tag{3}$$

where π_d is the prior of the drawing class, and π_w is the prior of the writing class.

The scaling in Eq 3 is equivalent to training with a balanced data set. For prediction, we can adjust the outputs of the neural network through the Bayes' theorem to get the correct posterior probabilities:

$$\tilde{y}_n = \frac{\pi_d y_n}{\pi_d y_n + \pi_w(1-y_n)}$$

The above method to address the problem of imbalanced underlying class distribution only works when the priors π_d and π_w are the same for the training set and the files we see at run time. Unfortunately, for the drawing/writing classification problem, this assumption is not always true. The class distribution of strokes in a digital ink file could be quite different from that of another file or the training set. For example, a writing–intensive file such as a formal letter could contain only a few or probably even no drawing strokes, but overwhelmingly more writing strokes. And similarly, a file of a doodling drawn by a small kid could have no writing strokes at all. It all depends on the particular type of the file (see Section 6).

As observed by Saerens et al [SLD02], under these circumstances, the estimates of the posterior probabilities are not valid for the new observed data, because they are generated by the neural network trained on a different distribution. And they may lead to sub–optimal classification results. As for the NN+HMM method, the problem might be worse, because these posteriori probabilities are also used indirectly as observation probabilities of the HMM. It is obvious that we need to find a way to adapt the outputs of the NN to get more accurate estimates of the posteriors $p(d_n|x_n)$.

As shown in Table 3, for the drawing/writing classification problem, the transition probabilities among successive strokes also change from one file set to another. Thus, the assumptions of the NN+HMM algorithm—the transition probabilities and priors are fixed aforehand and can be transfered from the training set to the testing set—both fail to hold. So, in addition to adjusting the outputs of the neural network, we also need to find the "correct" priors and transition probabilities for the HMM.

In Section 4, we describe how to adjust the outputs of the neural network with an EM algorithm [SLD02]. In Section 5, we show how to adapt the parameters of the HMM to newly observed data with an EM algorithm albeit that the emission probabilities are derived indirectly from the outputs of a deterministic model.

4. Adjusting Outputs of the Neural Network

Saeren et al [SLD02] proved that under certain assumptions the outputs of a neural network could be adjusted to new a priori probabilities with EM. The first assumption is that the within–class conditional probabilities, $p(x_t|d_t)$, do not change in the new data set. They remain the same at the training time and at prediction. The second assumption is that the N samples in the training data set $\{x_1, x_2, ..., x_N\}$ are drawn independently according to $p(x|d=i)$ given $d=i, i=1\ldots S$, S is the number of classes of the neural network and the number of hidden states for the HMM.

Let $\hat{p}_{trn}(d=i)$ be the priors of the training set and $\hat{p}_{trn}(d=i|x)$ be the posterior probability predicted by the classifier, we have

$$\hat{p}_{trn}(x|d=i) = \frac{\hat{p}_{trn}(d=i|x)\hat{p}_{trn}(x)}{\hat{p}_{trn}(d=i)}. \tag{4}$$

Analogously, let $\hat{p}_a(d=i)$ be the new a priori probabilities, $\hat{p}_a(d=i|x)$ be the adjusted a posteriori probabilities, and $\hat{p}_a(x)$ be the new prior of observations, we have

$$\hat{p}_a(x|d=i) = \frac{\hat{p}_a(d=i|x)\hat{p}_a(x)}{\hat{p}_a(d=i)}. \tag{5}$$

According to the first assumption,

$$\hat{p}_{trn}(x|d=i) \Longleftrightarrow \hat{p}_a(x|d=i).$$

With Eq 4, and Eq 5, we have

$$\hat{p}_a(d=i|x) = \frac{\frac{\hat{p}_a(d=i)}{\hat{p}_{trn}(d=i)}\hat{p}_{trn}(d=i|x)}{\sum_{j=1}^{S}\frac{\hat{p}_a(d=j)}{\hat{p}_{trn}(d=j)}\hat{p}_{trn}(d=j|x)}.$$

As pointed out by Saeren et al [SLD02], with the above equation, we can estimate $\hat{p}_a(d=i)$ iteratively with the EM algorithm in Table 1 (we denote the algorithm as NN*). In our case, by using HMM to model the entire ink sequence, we have already assumed that $p(x_t|d_t)$ does not change, and that given d_t, x_1, x_2, ..., and x_T are independent of each other. So we can apply the technique to adjust the outputs of the neural network to get more accurate predictions of $p(d_t|x_t)$.

Table 1: *EM algorithm for computing new a priori probabilities for the outputs of an NN—NN**

```
Initialize: $i = 1\ldots S, n = 1\ldots N$
    $\hat{p}^{(0)}(d=i) = \hat{p}_{trn}(d=i)$
E–Step: $\hat{p}^{(k)}(d=i|x_n) = \frac{\frac{\hat{p}^{(k)}(d=i)}{\hat{p}_{trn}(d=i)}\hat{p}_{trn}(d=i|x_n)}{\sum_{j=1}^{S}\frac{\hat{p}^{(k)}(d=j)}{\hat{p}_{trn}(d=j)}\hat{p}_{trn}(d=j|x_n)}$,
M–Step: $\hat{p}^{(k+1)}(d=i) = \frac{1}{N}\sum_{n=1}^{N}\hat{p}^{(k)}(d=i|x_n)$
```

5. Adjusting the Parameters of HMM

EM algorithms are not only useful for adjusting the outputs of the NN, they can also be applied to find the "true" parameters of the HMM [Bil98, MK97] to new observed data.

In our case, the observation probabilities of the HMM are obtained indirectly from a deterministic model, so we can't use the classical Baum–Welch algorithm to find the parameters of the HMM. Nevertheless, as shown below, we can formulate the problem in such a way that we can still use the EM algorithm to solve it. In the E–step of the EM algorithm, we use a modified version of the Forward–Backward algorithm to compute the necessary quantities, and in M–step, we use these quantities to find a closed form solution that maximizes $Q(\lambda, \lambda')$.

5.1. Formulation of the EM algorithm

Here is how we formulate the problem to solve it with the EM algorithm. Let $x_1, x_2, \ldots, x_T$ be the observed sequence of strokes (the incomplete data). Let $d_1, d_2, \ldots, d_T$ be the labels of the strokes—the hidden states of the HMM (the unknown data). Then the likelihood function for the complete data $x_1, x_2, \dots, x_T$, $d_1, d_2, \dots, d_T$ can be defined as in Eq 2. With Eq 2, the log likelihood of the complete data becomes

$$\begin{aligned}&\log p(x_1, x_2, \dots, x_T, d_1, d_2, \dots, d_T)\\ &= \log p(d_1) + \sum_{t=1}^{T-1} \log p(d_{t+1}|d_t) + \sum_{t=1}^{T} p(d_t|x_t)\\ &\quad + \sum_{t=1}^{T} \log p(x_t) - \sum_{t=1}^{T} \log p(d_t)\\ &= \sum_{t=1}^{T-1} \log p(d_{t+1}|d_t) + \sum_{t=1}^{T} p(d_t|x_t) + \sum_{t=1}^{T} \log p(x_t) - \sum_{t=2}^{T} \log p(d_t).\end{aligned} \tag{6}$$

Note that in Eq 6 the term $\sum_{t=1}^{T} \log p(x_t)$ relates only to the observations, and thus is a constant term. With the term $\sum_{t=1}^{T} \log p(d_t|x_t)$ computed from the outputs of the neural network, we can compute the maximum likelihood estimates of the transition probabilities $\{p(d=j|d=i)\}$, and priors $\{p(d=i)\}$ with EM, as stated in Theorem 1.

Theorem 1 Let $\{x_1, x_2, \dots, x_T\}$ be a given sequence of ink strokes. Let $\{p(d_t|x_t); t = 1 \dots T\}$ be the posterior probabilities output by the neural network. Let the log likelihood of the complete data be defined as in Eq 6. Let S be the number of all possible hidden states. Let $p(d=j|d=i)$ be denoted by $a_{i,j}$, and $p(d=i)$ be denoted by π_i. Let λ be the collection of parameters $\left(a_{i,j}, \pi_i\right)$, and λ' be the current estimates of the parameters. We can use an EM algorithm to estimate new values of $a_{i,j}$ and π_i. In the M–Step of the algorithm, we update $a_{i,j}$ and π_i as

$$a_{i,j} = \frac{\sum_{t=1}^{T-1} p(x_1, x_2, \dots, x_T, d_t = i, d_{t+1} = j|\lambda')}{\sum_{j=1}^{S} \sum_{t=1}^{T-1} p(x_1, x_2, \dots, x_T, d_t = i, d_{t+1} = j|\lambda')}$$

and

$$\pi_i = \frac{\sum_{t=1}^{T} p(x_1, x_2, \dots, x_T, d_t = i|\lambda')}{\sum_{j=1}^{S} \sum_{t=1}^{T} p(x_1, x_2, \dots, x_T, d_t = j|\lambda')}.$$

(Limited by space, proof omitted.)

Since we compute the observation probabilities $p(x_t|d_t)$ indirectly from $p(d_t|x_t)$ through Bayes' theorem, and we do not have the probabilities $p(x_t)$, we can't compute the quantities

$$\alpha_i(t) = p(x_1, x_2, \dots, x_t, d_t = i|\lambda')$$

and

$$\beta_i(t) = p(x_{t+1}, \dots, x_T|d_t = i, \lambda').$$

directly as in the classical Forward–Backward algorithm. But as entailed by Theorem 1, these quantities are only used in a "fractional" form. So we can modify the Forward–Backward algorithm as shown in Eq 7 and Eq 8 to compute the quantities $\hat{\alpha}_i(t)$ and $\hat{\beta}_i(t)$ and then use them to find the new estimates for $a_{i,j}$ and π_i.

Modified Forward algorithm:

$$\hat{\alpha}_i(1) = \tilde{y}_{1,i}, \quad \hat{\alpha}_j(t+1) = \sum_{i=1}^{S} \hat{\alpha}_i(t)\, a_{i,j}\, \frac{\tilde{y}_{t+1,j}}{\pi_j}. \tag{7}$$

Modified Backward algorithm:

$$\hat{\beta}_i(T) = 1, \quad \hat{\beta}_i(t) = \sum_{j=1}^{S} a_{i,j}\, \frac{\tilde{y}_{t+1,j}}{\pi_j} \hat{\beta}_j(t+1). \tag{8}$$

In fact, the quantities $\hat{\alpha}_i(t)$ and $\hat{\beta}_i(t)$ computed by the above modified Forward– Backward are not unrelated to $\alpha_i(t)$ and $\beta_i(t)$. By simple induction, we can establish the relationships among them and the probability of the observation sequence, $p(x_1, x_2, \dots, x_T|\lambda)$, as shown in Theorem 2.

Theorem 2 Let $\alpha_i(t) = p(x_1, x_2, \dots, x_t, d_t = i|\lambda')$ and $\beta_i(t) = p(x_{t+1}, \dots, x_T|d_t = i, \lambda')$. Let $\hat{\alpha}_i(t)$ and $\hat{\beta}_i(t)$ be defined by the modified Forward–Backward algorithm in Eq 7 and Eq 8, we have

$$\begin{aligned}\alpha_i(t) &= \hat{\alpha}_i(t) \prod_{\tau=1}^{t} p(x_\tau)\\ p(x_1, x_2, \dots, x_T|\lambda) &= \sum_{i=1}^{S} \hat{\alpha}_i(T) \prod_{\tau=1}^{T} p(x_\tau)\\ \beta_i(t) &= \hat{\beta}_i(t) \prod_{\tau=t+1}^{T} p(x_\tau), 1 \le t < T\\ p(x_1, x_2, \dots, x_T|\lambda) &= \sum_{i=1}^{S} \hat{\beta}_i(1) \tilde{y}_{1,i} \prod_{\tau=1}^{T} p(x_\tau)\end{aligned}$$

In the Baum-Welch algorithm, we compute $\gamma_i(t)$ and $\xi_{i,j}(t)$ from the values $\alpha_i(t)$ and $\beta_i(t)$ and then use them to to estimate the new values for the parameters $a_{i,j}$ and π_i of the HMM. $\gamma_i(t)$ is usually defined as

$$\gamma_i(t) = \frac{p(x_1, x_2, \dots, x_T, d_t = i|\lambda)}{p(x_1, x_2, \dots, x_T|\lambda)} \tag{9}$$

Given the Markov Property of HMM, with d_t known, $x_1, x_2, \dots, x_t$ and $x_{t+1}, \dots, x_T$ are independent of each other. Thus, in the Baum-Welch algorithm, $\gamma_i(t)$ is computed as:

$$\gamma_i(t) = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{S} \alpha_j(t)\beta_j(t)}.$$

Although we can't compute $\alpha_i(t)$ and $\beta_i(t)$ directly, with the relationships established in Theorem 2, we can easily prove the following corollary.

Corollary 3 Let $\hat{\alpha}_i(t)$ and $\hat{\beta}_i(t)$ be defined by the modified Forward–Backward algorithm in Eq 7 and Eq 8. $\gamma_i(t)$ as defined in Eq 9 can be computed as:

$$\gamma_i(t) = \frac{\hat{\alpha}_i(t)\hat{\beta}_i(t)}{\sum_{j=1}^{S} \hat{\alpha}_j(t)\hat{\beta}_j(t)}.$$

Similarly, let $\xi_{i,j}(t)$ be defined as the probability of $p(d_t = i, d_{t+1} = j|x_1, x_2, \dots, x_T, \lambda)$. In the Baum-Welch algorithm, $\xi_{i,j}(t)$ are computed as

$$\xi_{i,j}(t) = \frac{\alpha_i(t) a_{i,j} b_j(x_{t+1}) \beta_j(t+1)}{\sum_{i=1}^{S} \sum_{j=1}^{S} \alpha_i(t) a_{i,j} b_j(x_{t+1}) \beta_j(t+1)}, \tag{10}$$

whereas $b_j(x_{t+1})$ are the parameters for the observation probabilities $p(x_{t+1}|d_{t+1} = j)$. In our case, we do not need the parameters $b_j(x_{t+1})$ because our emission probabilities are computed indirectly from the outputs of the neural network.

Analogously, with the relationships established in Theorem 2, we can still compute $\xi_{i,j}(t)$ in a similar manner with $\hat{\alpha}_i(t)$ and $\hat{\beta}_i(t)$.

Corollary 4 Given the definition of $\xi_{i,j}(t)$ as in Eq 10 and $\hat{\alpha}_i(t)$ and $\hat{\beta}_i(t)$ as defined by the iterative processes in Eq 7 and Eq 8, $\xi_{i,j}(t)$ can be computed as

$$\xi_{i,j}(t) = \frac{\hat{\alpha}_i(t) a_{i,j} \frac{\tilde{y}_{t+1,j}}{\pi_j} \hat{\beta}_j(t+1)}{\sum_{i=1}^{S} \sum_{j=1}^{S} \hat{\alpha}_i(t) a_{i,j} \frac{\tilde{y}_{t+1,j}}{\pi_j} \hat{\beta}_j(t+1)}.$$

With Theorem 1, Corollary 3, and Corollary 4, we can still use similar equations to estimate the new values of the parameters $a_{i,j}$ and π_i.

Theorem 5 With $\gamma_i(t)$ and $\xi_{i,j}(t)$ computed as in Corollary 3 and Corollary 4, in the M–step of the EM algorithm, the new values of the transition probabilities for the HMM can be computed as

$$a_{i,j} = \frac{\sum_{t=1}^{T-1} \xi_{i,j}(t)}{\sum_{j=1}^{S} \sum_{t=1}^{T-1} \xi_{i,j}(t)}$$

and the new values of the priors for the HMM can be computed as:

$$\pi_i = \frac{\sum_{t=2}^{T} \gamma_i(t)}{\sum_{j=1}^{S} \sum_{t=2}^{T} \gamma_j(t)}$$

5.2. Numerical Overflow in the Modified Forward–Backward Algorithm

In practice, the modified Forward–Backward algorithm in Eq 7 and Eq 8 can have potential numerical overflow problems. In the Forward–Backward pass of the Baum–Welch Algorithm, $\alpha_i(t)$ is the probability of observing the partial sequence $x_1, x_2, \ldots, x_t$ and ending in the state i; $\beta_i(t)$ is the probability of observing the partial sequence $x_{t+1}, x_{t+2}, \ldots, x_T$, given the HMM is in state i at time t. Thus, the $\alpha_i(t)$ and $\beta_i(t)$ quantities generated at each step by Forward–Backward algorithm are always within the range of $[0,1]$.

But the $\hat{\alpha}_i(t)$ and $\hat{\beta}_i(t)$ values computed by our modified Forward–Backward algorithm are not probabilities, and from Theorem 2, we know that $\alpha_i(t) = \hat{\alpha}_i(t) \prod_{\tau=1}^{t} p(x_t)$. Since $\prod_{\tau=1}^{t} p(x_t)$ can be very small, $\hat{\alpha}_i(t)$ can become much larger than $\alpha_i(t)$. In fact, it could even cause a numeric overflow problem in some extreme cases.

This numerical overflow problem of the modified Forward–Backward algorithm can be avoided by normalizing $\hat{\alpha}_i(t)$ and $\hat{\beta}_i(t)$. A natural choice for the normalization factors is $\prod_{\tau=1}^{t} p(x_\tau)$ for $\hat{\alpha}_i(t)$, and $\prod_{\tau=t+1}^{T} p(x_\tau)$ for $\hat{\beta}_i(t)$. But we do not know the quantities $p(x_1), p(x_2), \ldots, p(x_T)$. We have to improvise and find other suitable normalization factors.

In fact, as stated in Theorem 6, no matter what values of μ_t and ν_t, we use to normalize $\hat{\alpha}_i(t)$ and $\hat{\beta}_i(t)$ at step t of the forward and backward passes, our estimates for $\xi_{i,j}(t)$ and $\gamma_i(t)$ remain the same. This means that the formulae to estimate the parameters $(a_{i,j}, \pi_i)$ of the HMM do not change, and the EM algorithm still converges to the same local optimal values for $(a_{i,j}, \pi_i)$.

Theorem 6 Let $\{\mu_1, \mu_2, \ldots, \mu_T\}$ be the normalization sequence for the $\hat{\alpha}$ sequence, and let $\{\nu_1, \nu_2, \ldots, \nu_T\}$ be the normalization sequence for the $\hat{\beta}$ sequence. Let $\alpha'(t) = \mu_t \hat{\alpha}_i(t)$, and $\beta'(t) = \nu_t \hat{\beta}_i(t)$. We have

$$\xi_{i,j}(t) = \frac{\alpha'_i(t) a_{i,j} \frac{\tilde{y}_{t+1,j}}{\pi_j} \beta'_j(t+1)}{\sum_{i=1}^{S} \sum_{j=1}^{S} \alpha'_i(t) a_{i,j} \frac{\tilde{y}_{t+1,j}}{\pi_j} \beta'_j(t+1)},$$

$$\gamma_i(t) = \frac{\alpha'_i(t) \beta'_i(t)}{\sum_{j=1}^{S} \alpha'_j(t) \beta'_j(t)}.$$

In summary, with the normalization of $\hat{\alpha}_i(t)$ and $\hat{\beta}_i(t)$, the EM algorithm (denoted by HMM*) to estimate the parameters $(a_{i,j}, \pi_i)$ can be summarized as in Table 2.

Table 2: *EM with Modified Forward–Backward Algorithm (HMM*)*

Initialize: $a_{i,j}^{(0)} = a_{i,j}^{trn}$, $\pi_i^{(0)} = \pi_i^{trn}$

E–Step: Forward Pass:

$$\alpha'_i(1) = \tilde{y}_{1,i},\ \alpha'_j(t) = \frac{\sum_{i=1}^{S} \alpha'_i(t) a_{i,j}^{(k)} \frac{\tilde{y}_{t+1,j}}{\pi_j^{(k)}}}{\sum_{j=1}^{S} \sum_{i=1}^{S} \alpha'_i(t) a_{i,j}^{(k)} \frac{\tilde{y}_{t+1,j}}{\pi_j^{(k)}}}$$

Backward Pass:

$$\beta'_i(T) = 1,\ \beta'_i(t) = \frac{\sum_{j=1}^{S} a_{i,j}^{(k)} \frac{\tilde{y}_{t+1,j}}{\pi_j^{(k)}} \beta'_j(t+1)}{\sum_{i=1}^{S} \sum_{j=1}^{S} a_{i,j}^{(k)} \frac{\tilde{y}_{t+1,j}}{\pi_j^{(k)}} \beta'_j(t+1)}$$

$$\xi_{i,j}(t) = \frac{\alpha'_i(t) a_{i,j}^{(k)} \frac{\tilde{y}_{t+1,j}}{\pi_j^{(k)}} \beta'_j(t+1)}{\sum_{i=1}^{S} \sum_{j=1}^{S} \alpha'_i(t) a_{i,j}^{(k)} \frac{\tilde{y}_{t+1,j}}{\pi_j^{(k)}} \beta'_j(t+1)},\ \gamma_i(t) = \frac{\alpha'_i(t) \beta'_i(t)}{\sum_{j=1}^{S} \alpha'_j(t) \beta'_j(t)}$$

M–Step:

$$a_{i,j}^{(k+1)} = \frac{\sum_{t=1}^{T-1} \xi_{i,j}(t)}{\sum_{j=1}^{S} \sum_{t=1}^{T-1} \xi_{i,j}(t)},\ \pi_i^{(k+1)} = \frac{\sum_{t=2}^{T} \gamma_i(t)}{\sum_{j=1}^{S} \sum_{t=2}^{T} \gamma_j(t)}$$

6. Evaluation

To evaluate the algorithms NN* and HMM*, we use 6 sets of real–world digital ink files, manuanlly selected and labeled to guarantee that each set correspond to a unique file type [Rag05].

The following list gives a brief description of each data set:

1. Free–Form Drawings: files that contain mostly drawing strokes and relatively few or no writing strokes.
2. Flow–Charts: in these files, writing strokes are usually grouped into small blocks. These writing blocks are often enclosed by drawing strokes forming containers, and/or connected by drawings strokes of callouts or connectors [QSM05].
3. Maps: these files are collected to illustrate how people would draw a map for traveling or driving instructions. They usually contain a decent amount of drawing strokes with strong temporal correlations, and also a signficant amount of writing strokes embedded in the drawings.
4. Tables: files that consist of writing strokes separated by a few long and easy–to–recognize drawing strokes.

Table 4: *Metrics Definition*

$Recall_D$	$\frac{\#\ \text{Drawing strokes classified correctly}}{\#\ \text{strokes labeled as Drawing}}$
FPR	$\frac{\#\ \text{Writing strokes classified as Drawing}}{\#\ \text{strokes labeled as Writing}}$
C	$\frac{\#\ \text{strokes labeled as Writing}}{\#\ \text{strokes labeled as Drawing}}$
$Precision_D$	$\frac{Recall_D}{Recall_D + C \cdot FPR}$
F_D	$\frac{2 * \#\ \text{Drawing strokes classified correctly}}{\#\ \text{strokes labeled as Drawing} + \#\ \text{strokes classified as Drawing}}$
F_W	$\frac{2 * \#\ \text{Writing strokes classified correctly}}{\#\ \text{strokes labeled as Writing} + \#\ \text{strokes classified as Writing}}$

5. Highly Annotated Text files: files that contain mostly writing strokes, and a few drawing strokes that annotate the writings.
6. Lightly Annotated Text files: files that contain mostly writing strokes, and few drawing strokes that annotate the writings.

Table 3 shows the priors and the transition probabilities of the data sets. From the table, we can see that the Free–form Drawings data set has the strongest temporal correlations between successive strokes. The probability of transiting from a writing stroke to a drawing stroke is only 0.074738, while the probability of transiting from a drawing stroke to a writing stroke is also very small, 0.092399. Another interesting trend to notice is that as the prior probability of a stroke being drawing drops, the temporal correlation among the successive strokes also gets weaker. In this section, we describe the design and results of a series of experiments to test the following hypotheses:

- NN vs NN+HMM: By using HMM to exploit the temporal correlation between successive strokes, does the NN+HMM algorithm improve the classification performance of the neural network?
- NN vs NN*: By using EM to adjust the posterior probabilities output by the neural network, does NN* improve on the performance of the neural network?
- NN+HMM vs NN*+HMM: By using the more accurate posterior probabilities produced by NN*, does the HMM find better assignment of the labels?
- NN*+HMM vs NN*+HMM*: By adjusting the priors and transition probabilities to the data set (the current sequence or the current file), does NN*+HMM* find better label assignments than NN*+HMM?

In this section, we use the following four metrics (Table 4) to evaluate these algorithms: $\mathcal{F}_D$, $\mathcal{F}_W$, $Precision_D$ and $Recall_D$. $\mathcal{F}_D$ is the F–measure [Fla03] with drawing as the positive class, $\mathcal{F}_W$ is the F–measure with writing as the positive class, and $Precision_D$ [Fla03] and $Recall_D$ [Fla03] are the precision and recall of the drawing class.

NN vs NN+HMM: The figure (NN vs NN+HMM) presents the experimental results of the algorithms NN vs NN+HMM. We can see that by using HMM to arbitrate the outputs of a neural network in the context of the entire sequence, NN+HMM does improve the precision of the minority class. The improvement is even more significant for the Highly Annotated Text and Lightly Annotated Text data sets, where the priors of the two classes is even more skewed.

But the improvement in the precision comes at the expense of the recall of the minority class. In fact, in 4 out of the 6 data sets, $\mathcal{F}_D$ drops by 0.018 to 0.049, while $\mathcal{F}_W$ also decreases.

NN vs NN* Figure NN vs NN* shows the experimental results of the algorithms NN vs NN*. First, we can see by using EM to adjust the output posterior probabilities, NN* increases both $\mathcal{F}_D$ and $\mathcal{F}_W$ on all data sets except the Tables data set. The second interesting fact to notice is that the precision of drawing drops but the recall of drawing increases on the more "drawing–intensive" data sets, but the trend reverses on the more "writing–intensive" data sets. The more "drawing–intensive" data sets are the Free–form Drawings, Flow Charts, and the Maps data sets. The more "writing–intensive" data sets are the Highly Annotated Text and the Lightly Annotated Text sets.

The third interesting fact to notice is that with EM, NN* even out–performs NNs on the Free–form Drawing data set, which the neural network was trained on. This might look weird, considering the fact that the empirical priors we used to train the neural network are the "correct" priors of the data set we test it on. In fact, the differences are caused by the way we test the NN* and the HMM* algorithms—we take each file (a.k.a a sequence of ink strokes) as the "observation" set for adjusting the priors for the neural network, and the parameters of the HMM.

NN+HMM vs NN*+HMM The figure compares the experimental results of the two algorithms. Similar to the results of the experiments for NN vs NN*, on the more "drawing–intensive" classes, NN*+HMM increases F-measures on both the drawing and writing classes. The increase in the $\mathcal{F}_D$ varies from 0.009 (on the Maps data set) to 0.11 (on the Free–form Drawings data set). On these data sets, the recall of drawing also increases dramatically (0.10 for Free–form Drawings, 0.13 for Flow Charts, 0.07 for Maps, and 0.03 for Tables), while the precision of drawing drops slightly, around 0.005 to 0.017.

On the two "writing–intensive" data sets, again, the trend reverses. The precision of the drawing class increases even further, while $Recall_D$, $\mathcal{F}_D$ and $\mathcal{F}_W$ all decrease. **NN*+HMM vs NN*+HMM*** Similar to the proceeding two experiments, on the more "drawing–intensive" data sets (in this case, the Free-form Drawings, Flow Charts and Maps data sets), NN*+HMM* further increases the $\mathcal{F}_D$, $\mathcal{F}_W$ and $Recall_D$, but decreases $Precision_D$ slightly.

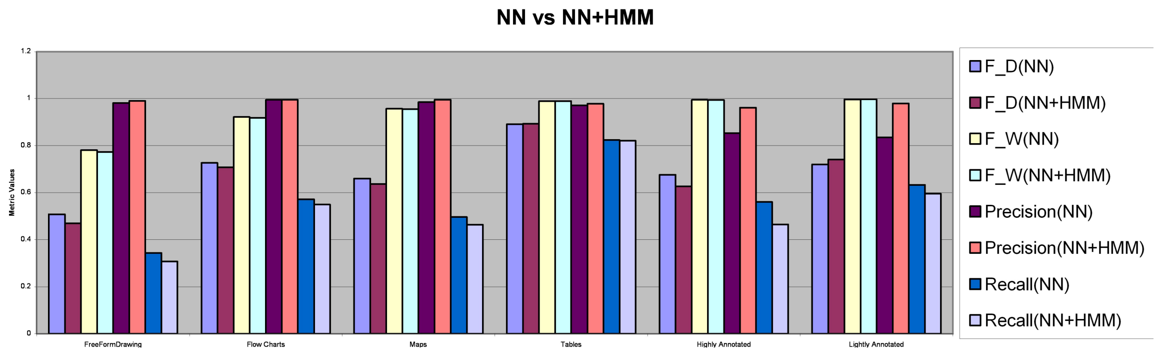
On the other three data sets, all four metrics decrease after adapting the priors and the transition probabilities of the HMM to the current ink file.

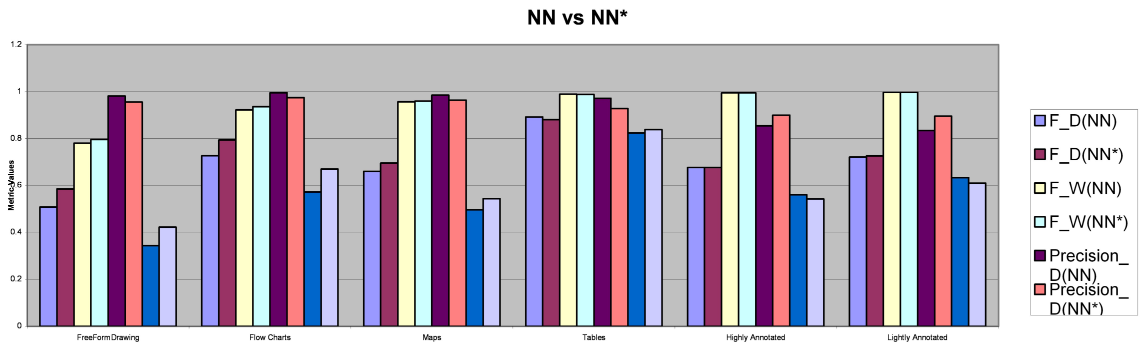
7. Discussion

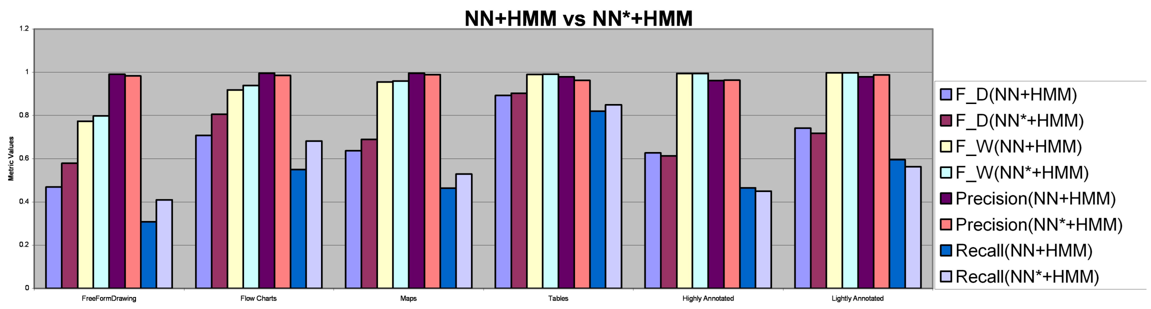
From the experimental results in Section 6, we found that the six data sets can be grouped into two clusters, the more "drawing–intensive" sets that include the Free–form Drawings, the Flow Charts, and the Maps data sets, and the more

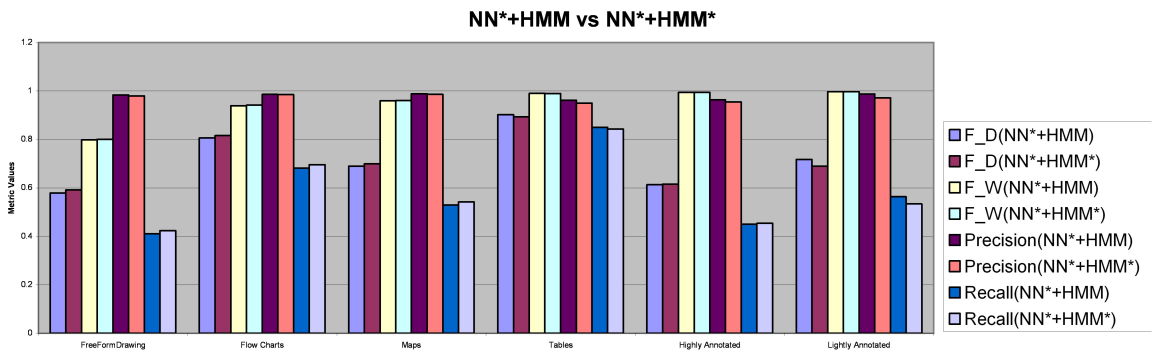
Table 3: *Priors and Transition Probabilities of Data Sets*

Distribution	Free–form Drawing	Flow Charts	Maps	Tables	Highly Annotated	Lightly Annotation
$p(d = D)$	0.457708	0.283563	0.298731	0.094095	0.020122	0.0123021
$p(d_{t+1} = W \mid d_t = D)$	0.092399	0.20133	0.150312	0.241339	0.494001	0.404412
$p(d_{t+1} = D \mid d_t = W)$	0.074738	0.079360	0.061211	0.024737	0.010405	0.005435
# of files	491	339	507	141	500	518
Avg Length	150.09	188.18	222.30	133.77	339.44	339.08









"writing–intensive" sets that include the Highly Annotated Text and the Lightly Annotated Text data sets. The sixth data set, the Tables set, is kind of an outlier. It sometimes behaves more like a "drawing–intensive" data set, and sometimes behaves more like a "writing–intensive" data set. Both the EM algorithm for adjusting the outputs of the neural network, and the EM algorithm for adjusting the parameters of the HMM work reasonably well on the "drawing–intensive" data sets, but not as well on the "writing–intensive" data sets.

The problem could be caused by the HMM's strong dependency on the Markov Property. For the more "writing–intensive" data sets, the temporal correlation from a drawing stroke to another drawing stroke is weaker (drawing strokes behave more like outliers for these scenarios), and our HMM based algorithm is no longer a good candidate for these situations.

8. Conclusion

In this paper, we present our efforts on addressing the issues of varying underlying class distribution of drawing/writing classification in an ink file.

First, we use an EM algorithm to adjust the outputs of the neural network to the priors of the newly observed data to get more accurate estimates of the posterior probabilities. Second, we use another EM algorithm to adapt the parameters of the HMM to new data sets.

From the experimental results of Section 6, on the more "drawing–intensive" data sets, both the EM algorithm for adjusting the outputs of the neural network and the EM algorithm for adapting the parameters of the HMM to the current observed data set increase the F–Measures of both the drawing and the writing classes and the recall of the drawing class, while only slightly decreasing the precision of the drawing class. The algorithms do not perform well on the more "writing–intensive" classes, where the prior of the drawing strokes are much smaller, and the temporal correlations between successive strokes are also weaker.

References

[Bil98] Bilmes J. A.: *A Gentle Tutorial on the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models*. Tech. Rep. TR–97–021, International Computer Science Institute, Berkeley CA, 1998.

[Bis95] Bishop C. M.: *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[BSH04] Bishop C. M., Svensen M., Hinton G. E.: Distinguishing text from graphics in on-line handwritten ink. pp. 142–147.

[CJK04] Chawla N. V., Japkowicz N., Kotcz A.: Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explorations 6*, 1 (2004), 1–6.

[Fla03] Flach P. A.: The geometry of roc space: Understanding machine learning metrics through roc isometrics. In *ICML* (2003), pp. 194–201.

[JS02] Japkowicz N., Stephen S.: The class imbalance problem: A systematic study. *Intell. Data Anal. 6*, 5 (2002), 429–449.

[MK97] McLachlan G. J., Krishnan T.: *The EM Algorithm and Extensions*. John Wiley & Sons, 1997.

[MR91] M.D.Richard, R.P.Lippmann: Neural network classifiers estimate bayesian a posteriori probabilities. *Neural Computation 3* (1991), 461–483.

[PF01] Provost F., Fawcett T.: Robust classification for imprecise environments. *Mach. Learn. 42*, 3 (2001), 203–231.

[QSM05] Qi Y., Szummer M., Minka T. P.: Diagram structure recognition by bayesian conditional random fields. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 191–196.

[Rab90] Rabiner L. R.: A tutorial on hidden markov models and selected applications in speech recognition. 267–296.

[Rag05] Raghupathy S.: *Ink Document and Writing Region Styles*. Tech. rep., TabletPC, Microsoft Corp., Redmond WA, 2005.

[SLD02] Saerens M., Latinne P., Decaestecker C.: Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. *Neural Computation 14*, 1 (2002), 21–41.