Laboratório VISGRAF
Instituto de Matemática Pura e Aplicada

Reconhecimento de Digitos com HMM

Anderson Mayrink da Cunha
Luiz Velho (orientador)

Technical Report TR-03-04 Relatório Técnico

August - 2003 - Agosto

Reconhecimento de Dígitos com HMM

Anderson Mayrink da Cunha

Luiz Velho (orientador)

Resumo

Nesse texto vamos apresentar um sistema simples de reconhecimento de escrita de dígitos isolados com HMM (Hidden Markov Models). Apresentamos também uma breve teoria de cadeias de Markov e HMM, assim como algoritmos para HMM. Por fim mostramos a nossa implementação e alguns resultados.

1 Introdução

O Reconhecimento de Escrita é uma área que tem recebido muito interesse acadêmico e que tem importantes aplicações comerciais.

Hidden Markov Models (HMM) tem sido aplicado em reconhecimento de voz desde o final da década de 70 e da metade da década de 80 aos dias atuais é o principal método para reconhecimento de voz. Em [1], [2], [3] e [4] temos textos sobre reconhecimento de voz. Inspirado no sucesso de HMM na modelagem da percepção humana da voz, muitas aplicações utilizam HMM, principalmente na modelagem de sinais unidimensionais. HMM também tem atraído interesse em reconhecimento de escrita ([6], [7], [8], [9]).

O sinal de entrada para o reconhecimento de escrita pode ser uma imagem bidimensional de uma escrita ou uma sequência de pontos no plano, obtidos com o mouse ou com a mesa digitalizadora (tablet). Nos ocuparemos aqui somente do segundo caso, em que o sinal de entrada do reconhecimento de escrita é unidimensional (sequência de pontos no plano).

Para simplificar a tarefa de reconhecimento de escrita vamos nos restringir ao reconhecimento dos dígitos de 0 a 9.

Uma dificuldade em reconhecimento de escrita é quando os símbolos são conectados, de forma que devemos segmentar os símbolos ou reconhecer palavras inteiras (ou partes conectadas das palavras). Vamos fugir dessa dificuldade assumindo que os símbolos são escritos em um único traço, de modo que sabemos exatamente onde começa e termina cada dígito.

Nesse texto vamos descrever um sistema bastante simples de reconhecimento de escrita basedo em HMM. O texto é organizado da seguinte forma:

Na próxima seção apresentamos como obter num modelo probabilistico qual é o dígito mais provável dado um sinal de entrada. Na seção 3 descrevemos o processamento de escrita usado.

Nas seções 4 e 5 descrevemos Cadeias de Markov e HMM, assim como os algoritmos usados para o reconhecimento de dígitos.

Na seção 6 descrevemos a implementação. Na seção 7 temos alguns resultados e na seção 8 apresentamos as conclusões desse trabalho.

Figura 1: Pontos de Entrada do Sistema de Reconhecimento de Dígitos.

2 Modelo Probabilístico

O sinal de entrada do sistema de reconhecimento de dígitos isolados é um conjunto de pontos no plano como vemos na figura 1. Como os dígitos são escritos em um único traço, sabemos exatamente onde começa e termina cada dígito. Os valores das coordenadas x e y de cada ponto não são os valores mais relevantes para representar a escrita. O ângulo é um aspecto relevante (feature) comumente usado num sinal de escrita [6]. Na seção de Processamento de Escrita explicamos com mais detalhes como obter features num sinal de escrita. Por ora, denotamos o sinal de entrada de um dígito como a observação $O = O_1, O_2, \ldots, O_n$, que é uma sequência de valores O_i já quantizados.

O problema de reconhecimento de escrita consiste em descobrir de qual dígito (entre 0 e 9), a observação O é mais parecida.

Um modo de resolver esse problema é usar um modelo estocástico em que possamos:

1. Treinar o modelo, isto é, obter parâmetros específicos λ_i para cada dígito i. O treinamento deve ser feito a partir de dados reais. Por exemplo, o usuário (ou vários usuários) escrevem várias vezes o dígito i e o algoritmo de treinamento se encarrega de obter parâmetros específicos λ_i para o modelo do dígito i.

2. Calcular o valor de $P(O|\lambda_i)$, que é a probabilidade de ocorrência da observação O (sinal de entrada que queremos reconhecer) dado um modelo de parâmetros λ_i. Podemos decidir de qual dígito o dado de entrada O é mais parecido calculando o valor $P(O|\lambda_i)$ para todos símbolos i. O símbolo i^* escolhido é aquele com maior valor de $P(O|\lambda_{i^*})$. Isto é:

$$i^* = \underset{i}{argmax} P(O|\lambda_i)$$

O modelo probabilistico usado é o HMM. Na seção HMM descrevemos esse modelo, assim como os algoritmos para o cáculo de $P(O|\lambda_i)$ e para o treinamento de um HMM.

Na figura 2 temos um diagrama esquemático do reconhecimento de escrita. O sinal de entrada (fig. 1) é processado, obtendo-se O. Calculamos $P(O|\lambda_i)$ para todos os i e selecionamos aquele com valor máximo.

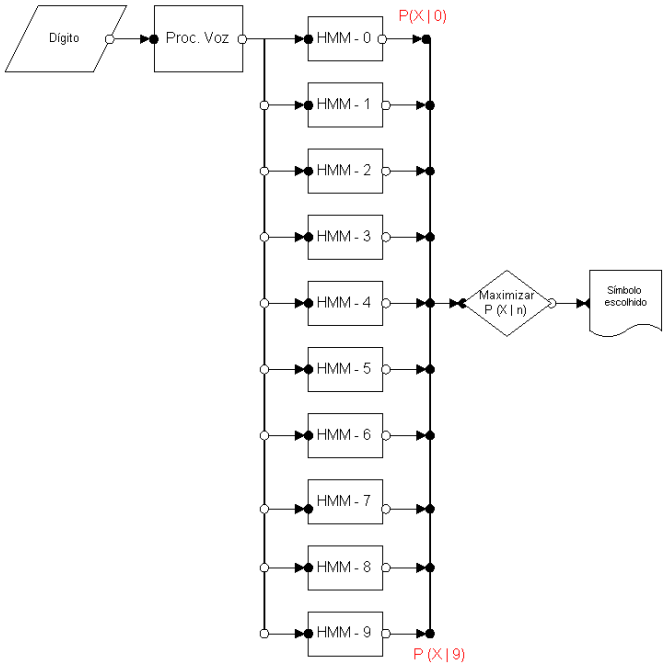

Figura 2: Sistema de Reconhecimento de Dígitos

3 Processamento de Escrita

O processamento da escrita é parte fundamental para uma boa performance de um sistema de reconhecimento de escrita. Como assumimos que os símbolos são escritos em um único traço, como na figura 1, não é necessário nenhum tipo de segmentação da escrita (no caso de mais de um símbolo escrito com um único traço) e também não é necessário verificar quais traços pertencem a um único símbolo (no caso de um único símbolo ser escrito em mais de um traço).

Um tipo de pré-processamento comum em escrita é aplicar alguma técnica de suavização para eliminar ruído ou algum problema de discretização. Também é comum parametrizar a curva por comprimento de arco, obtendo pontos equidistantes para a representação da curva. A obtenção de pontos equidistantes da curva é útil pois se utilizamos somente os pontos obtidos da interface, o reconhecimento fica altamente dependente da velocidade de obtenção dos pontos (em um computador lento (ou velocidade alta de escrita) temos poucos pontos na representação da curva e num computador rápido (ou velocidade baixa na escrita) a quantidade de pontos é muito maior). No nosso sistema simples de processamento de escrita não foram implementados nenhum dos pré-processamentos acima citados.

A obtenção das características relevantes (features) de um sinal é um fator importante na performance do reconhecimento de escrita. Uma feature comumente usada em escrita é o ângulo, que é invariante por translação e escalamento (invariância por rotação não é conveniente em reconhecimento de escrita pois não poderíamos distinguir entre os dígitos 6 e 9, por exemplo). Outras features podem ser usadas, como a curvatura ou mesmo um vetor de features para cada

instante de tempo, como é comum em reconhecimento de voz ([3], [4]).

Optamos pela implementação mais simples possível: usamos somente os pontos obtidos (sem nenhum pré-processamento ou reparametrização) pela interface do sistema (iup [10]) e a única feature usada é o ângulo entre pontos consecutivos. Se desejamos trabalhar com um modelo de saída discreto, devemos efetuar uma quantização vetorial para obter um conjunto de ângulos discretos. Um algoritmo de quantização vetorial comum em reconhecimento de voz e escrita é o k-means [3], [5] (e suas variantes, por exemplo, o algoritmo LBG). Optamos por uma quantização vetorial mais simples possível: simplesmente dividimos o espaço de ângulos em 64 partes iguais, obtendo um codebook de tamanho 64.

4 Cadeias de Markov

Em 1907, Markov definiu e investigou propriedades que são hoje conhecidas como cadeias de Markov. A principal característica dos processos de Markov é que o modo que toda a história passada afeta o futuro está completamente resumida no valor atual do processo. Modelos de Markov tem aplicações nos mais diversos ramos. Cadeias de Markov discretas são a base de HMM e vamos apresentá-la brevemente nesta seção.

Considere um sistema cuja evolução seja descrita por um processo estocástico $\{X(n) = X_n, n = 1, 2, \dots\}$, consistindo de uma família de variáveis aleatórias. O valor s_n assumido pela variável aleatória X_n é chamado de *estado* do sistema no tempo discreto n. O conjunto de todos os valores que as variáveis aleatórias podem assumir é chamado de espaço de estados do sistema. Se a estrutura do processo estocástico é tal que a distribuição de probabilidade condicional de X_n depende somente do valor de X_{n-1} e é independente de todos os valores anteriores, dizemos que o processo é uma cadeia de Markov. Mais precisamente:

Definição 1 *Uma sequência de variáveis aleatórias $X_1, X_2, \dots$ é uma cadeia de Markov em tempo discreto se para todo tempo n ($n = 1, 2, \dots$) e para todo o espaço de estados do sistema temos que:*

$$P[X_n = s_n \mid X_1 = s_1, X_2 = s_2, \dots, X_{n-1} = s_{n-1}] = P[X_n = s_n \mid X_{n-1} = s_{n-1}]$$

Podemos pensar na cadeia de Markov como um modelo gerador consistindo de estados ligados entre si por transições possíveis. Em cada unidade de tempo n um estado particular é visitado e o modelo coloca na saída o símbolo associado àquele estado.

Se a probabilidade condicional $P[X_n = s_j \mid X_{n-1} = s_i]$ for independente do tempo n, chamamos a cadeia de Markov de homogênea. Todas as cadeias de Markov a partir daqui são homogêneas. Se $P_{ij} = P[X_n = s_j \mid X_{n-1} = s_i]$ é independente de n, podemos organizar os P_{ij} numa matriz de transição de probabilidades.

Uma cadeia de Markov fica totalmente determinada pela matriz $A = \{a_{ij} = P[X_n = s_j \mid X_{n-1} = s_i]\}$ e pelo vetor π de probabilidades iniciais. Denotamos os parâmetros da cadeia de Markov por $\theta = \{A, \pi\}$.

Exemplo: Na figura 3 temos um exemplo simples de modelagem do clima com cadeias de Markov. O cálculo de $P(X|\theta)$ e a obtenção dos parâmetros θ do modelo a partir do treinamento é muito simples.

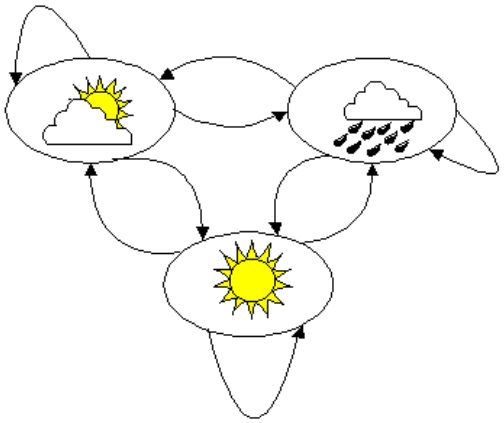

Figura 3: Cadeia de Markov para modelagem do clima

Vamos supor, por exemplo, que o sinal observado (sequência diária do clima) é:

$$X = S, N, C, C, N, S, S, N$$

onde S =sol, C =chuva e N =nublado. Como o estado futuro só depende do estado atual, temos que:

$$P(X|\theta) = \pi_S A_{SN} A_{NC} A_{CC} A_{CN} A_{NS} A_{SS} A_{SN}$$

Dado a sequência de observação X acima, podemos estimar os parâmetros da cadeia de Markov $\theta = \{A, \pi\}$ com as fórmulas:

$$\begin{cases} a_{ij} = P[X_n = s_j \mid X_{n-1} = s_i] = \frac{\text{n}^0 \text{ de transições de } s_i \text{ para } s_j}{\text{n}^0 \text{ de vezes no estado } s_i} \\ \pi_i = 1, \text{ se } X_1 = s_i \text{ e } \pi_i = 0, \text{ caso contrário.} \end{cases}$$

Daí temos que a seguinte estimativa para o modelo:

$$\begin{cases} A_{SS} = \frac{1}{3}, & A_{SC} = \frac{0}{3}, & A_{SN} = \frac{2}{3} \\ A_{CS} = \frac{0}{2}, & A_{CC} = \frac{1}{2}, & A_{CN} = \frac{1}{2} \\ A_{NS} = \frac{1}{2}, & A_{NC} = \frac{1}{2}, & A_{NN} = \frac{0}{2} \\ \pi_S = 1, & \pi_C = 0, & \pi_N = 0 \end{cases}$$

Note que como o tamanho da sequência X é curto, a estimativa não é confiável.

5 Hidden Markov Models (HMM)

Como já foi comentado, para modelar as características naturais de um sinal de voz ou de escrita à mão, a abordagem tradicional na área de reconhecimento de voz (ou escrita) tem sido utilizar HMMs, que descrevemos agora. Um Modelo Oculto de Markov (Hidden Markov Model - HMM) é uma cadeia de Markov em que cada estado pode emitir diferentes símbolos. Não conhecemos o

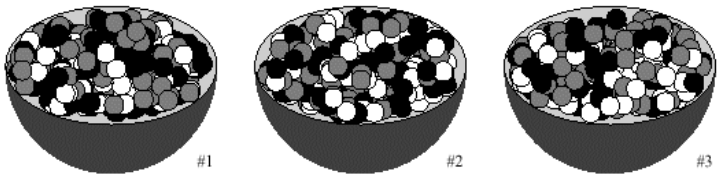

Figura 4: Urnas com bolas pretas, brancas e cinzas

Figura 5: Bolas obtidas

estado atual do modelo, mas apenas o sinal emitido. Um exemplo de HMM, obtido de [5] (assim como as figuras 4, 5 e 6), está descrito a seguir: Considere urnas com bolas pretas, brancas e cinzas como na figura 4. Vamos supor que o processo de retirar as bolas nas urnas é modelado por uma cadeia de Markov. Temos conhecimento das bolas selecionadas (como na figura 5), mas não de que urna cada uma dessas bolas foram retiradas. Esse processo é modelado por um HMM cujos parâmetros são as probabilidades iniciais e de transição da cadeia de Markov e ainda a probabilidade de retirarmos bolas pretas, brancas ou cinzas em cada um dos estados. Representamos graficamente esse HMM de 3 estados (urnas) e 3 símbolos de saídas (tipos de bolas em cada urna) na figura 6.

Vamos tratar somente de HMMs com número de estados e de unidades de tempo (tamanho da observação) discretos e finitos. O sinal O emitido em dada estado pode assumir os valores definidos pelo alfabeto, que também assumimos como sendo finito.

Um outro exemplo de HMM é uma partícula que para cada $t = 1, 2, \ldots, T$ muda para um local (ou estado) entre os N possíveis. Em cada um dos estados, a partícula emite um sinal (de um alfabeto de tamanho M). Temos que o estado futuro da partícula depende somente do estado atual, e não dos estados anteriores ou do tempo t. Não conhecemos os estados da partícula, mas somente os sinais emitidos por ela. Abaixo temos algumas notações.

- N é o número de estados do modelo, ou locais onde a partícula pode ir. Para simplificar a notação denominamos o conjunto de estados $\mathcal{S} = \{1, 2, \ldots, N\}$.
- M é o número total de símbolos distintos, o tamanho do alfabeto de sinais que a partícula

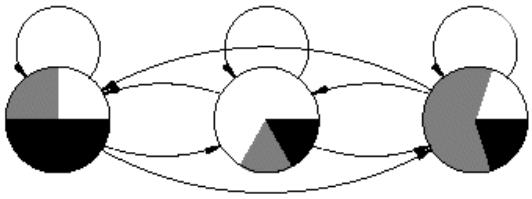

Figura 6: Representação do HMM com 3 estados e 3 símbolos.

emite. $V = \{v_1, v_2, \ldots, v_M\}$ é o alfabeto.

- T é a quantidade de unidades de tempo (tamanho da observação).
- $Q = \{q_1, q_2, \ldots, q_T\}$ onde q_t é o estado do modelo no tempo t.
- $O = \{O_1, O_2, \ldots, O_T\}$ onde O_t é símbolo observado no tempo t.
- $\pi = \{\pi_i\}, i = 1, \ldots, N.$ onde $\pi_i = P[q_1 = i]$ é a probabilidade de i ser o estado inicial do experimento.
- $A = \{a_{ij}\}$ é uma matriz $N \times N$, onde $a_{ij} = P[q_{t+1} = j \mid q_t = i]$ é a probabilidade da partícula ir do estado i para o estado j. Os a_{ij}'s são independentes do tempo t.
- $B = \{b_i(k)\}$ é uma matriz $N \times M$,onde $b_i(k)$ é a probabilidade do símbolo v_k ser observado no estado i.
- $\lambda = (A, B, \pi)$ é a notação compacta para um HMM.

Nas subseções seguintes apresentamos algoritmos para resolver dois dos principais problemas de HMM na maioria das aplicações que são: (1) Calcular $P[O \mid \lambda]$, a probabilidade da ocorrência da observação O dado o modelo λ e (2) Treinamento do HMM, isto é, a obtenção dos parâmetros do modelo $\lambda = (A, B, \pi)$ a partir de dados de treinamento O. Calculamos esses parâmetros do HMM com $\lambda^* = \arg\max_{\lambda} P[O \mid \lambda]$.

Não apresentamos todos os algoritmos necessários para o reconhecimento de escrita (por exemplo o algorimo backward para o cálculo de $P[O \mid \lambda]$, e o algoritmo de Baum-Welch usado no treinamento). Para uma visão mais completa desses algoritmos, consultar [2], [3] ou [5].

5.1 Cálculo de $P[O \mid \lambda]$

- *Dado um modelo $\lambda = (A, B, \pi)$ como calcular $P[O \mid \lambda]$, a probabilidade da ocorrência da observação $O_1, O_2, \ldots, O_T$?, isto é, $P[O \mid \lambda] =$?*

O modo mais imediato de se calcular $P[O \mid \lambda]$ é achar $P[O \mid Q, \lambda]$ para um estado fixado $Q = q_1, q_2, \ldots, q_T$ e somar as probabilidades sobre todos os estados possíveis, isto é:

$$P[O \mid \lambda] = \sum_Q P[O, Q \mid \lambda] = \sum_Q P[O \mid Q, \lambda] P[Q, \lambda]$$

onde $P[O \mid Q, \lambda] = b_{q_1}(O_1) b_{q_2}(O_2) \cdots b_{q_T}(O_T)$ e $P[Q, \lambda] = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \cdots a_{q_{T-1} q_T}$.

Para cada estado temos $2T - 1$ multiplicações para o cálculo de $P[O \mid Q, \lambda] P[Q, \lambda]$ e temos N^T estados. Daí temos ordem de $2TN^T$ multiplicações para o cálculo de $P[O \mid \lambda]$. Isso é inviável computacionalmente pois, por exemplo, um modelo com 10 estados e 100 instantes de tempo (observações), isto é, $N = 10, T = 100$, temos ordem de 10^{102} multiplicações. Para executar esse cálculo mais rapidamente usamos o procedimento descrito a seguir.

5.1.1 Algoritmo Forward

A variável forward $\alpha_t(i)$ é a probabilidade da observação da sequência parcial $O_1, O_2, \ldots, O_t$ e que no tempo t tenhamos o estado i.

$$\alpha_t(i) = P[O_1, O_2, \ldots, O_t, q_t = i \mid \lambda]$$

$\alpha_t(i)$ pode ser calculada recursivamente da seguinte forma:

$$\begin{cases} \alpha_1(i) = \pi_i b_i(O_1) & i = 1, \ldots, N \\ \alpha_{t+1}(j) = b_j(O_{t+1}) \left(\sum_{i=1}^{N} \alpha_t(i)\, a_{ij} \right) & t = 1, 2, \ldots, T-1 \quad , \quad j = 1, \ldots, N \end{cases}$$

Temos que

$$P[O \mid \lambda] = \sum_{i=1}^{N} \alpha_T(i)$$

Esse método de cálculo de $P[O \mid \lambda]$ envolve ordem de N^2T multiplicações.

A dedução da fórmula $\alpha_{t+1}(j) = b_j(O_{t+1}) \left(\sum_{i=1}^{N} \alpha_t(i)\, a_{ij} \right)$ deve ser feita com o devido cuidado pois $P[AB] = P[A]P[B]$ só se A e B são independentes.

$\alpha_{t+1}(j) = P[O_1, \ldots, O_t, O_{t+1}, q_{t+1} = j \mid \lambda] =$

$\sum_{i=1}^{N} P[O_1, \ldots, O_t, O_{t+1}, q_{t+1} = j \mid q_t = i, \lambda] P[q_t = i \mid \lambda] =$

$\sum_{i=1}^{N} P[O_1, O_2, \ldots, O_t \mid q_t = i, \lambda] P[q_t = i \mid \lambda] P[O_{t+1}, q_{t+1} = j \mid q_t = i, \lambda] =$

$\sum_{i=1}^{N} P[O_1, \ldots, O_t, q_t = i \mid \lambda] P[O_{t+1} \mid q_{t+1} = j, q_t = i, \lambda] P[q_{t+1} = j \mid q_t = i, \lambda] = b_j(O_{t+1}) \left(\sum_{i=1}^{N} \alpha_t(i)\, a_{ij} \right)$

5.1.2 Algoritmo Forward Escalado

O cálculo da variável forward com o algoritmo descrito na subseção acima envolve problema de precisão numérica (underflow) pois se, por exemplo, temos um alfabeto de tamanho 100 e 100 observações ($T = 100$), α_{t+1} é da ordem $T = 100$ vezes menor que α_t. Daí temos que $P[O \mid \lambda]$ é da ordem de 10^{-200}.

Para resolver este problema temos que escalar esse algoritmo. Esse escalamento consiste basicamente em cada passo do algoritmo criar uma variável auxiliar que indica o valor de $\sum_i \alpha_t(i)$.

Descrevemos a seguir o algoritmo forward escalado.

1. Inicialização:

$\widetilde{\alpha}_1(i) = \pi_i b_i(O_1), i = 1 \cdots N$

$c_1 = 1 / \left(\sum_{i=1}^{N} \widetilde{\alpha}_1(i) \right)$

$\widehat{\alpha}_1(i) = c_1 \widetilde{\alpha}_1(i), i = 1 \cdots N$ (escalado)

2. Indução

$$\widetilde{\alpha}_{t+1}(j) = b_j(O_{t+1}) \left(\sum_{i=1}^{N} \widehat{\alpha}_t(i)\, a_{ij} \right), t = 1, \ldots, T-1 \quad , \quad j = 1, \ldots, N$$
$$c_{t+1} = 1/\left(\sum_{i=1}^{N} \widetilde{\alpha}_{t+1}(i) \right), t = 1, \ldots, T-1$$
$$\widehat{\alpha}_{t+1}(j) = c_{t+1} \widetilde{\alpha}_{t+1}(i), t = 1, \ldots, T-1 \quad , \quad j = 1, \ldots, N \quad \text{(escalado)}$$

3. Finalização

$$P[O \mid \lambda] = 1/\left(\prod_{t=1}^{T} c_t \right)$$

isso é válido pois $\widehat{\alpha}_t(i) = \left(\prod_{\tau=1}^{t} c_\tau \right) \alpha_t(i)$ e $\sum_{i=1}^{N} \widehat{\alpha}_t(i) = 1$ logo $P[O \mid \lambda] = \sum_{i=1}^{N} \alpha_T(i) = \sum_{i=1}^{N} \widehat{\alpha}_T(i) / \left(\prod_{t=1}^{T} c_t \right) = \frac{1}{\prod_{t=1}^{T} c_t} \sum_{i=1}^{N} \widehat{\alpha}_T(i) = 1/\left(\prod_{t=1}^{T} c_t \right)$

Como $P[O \mid \lambda]$ pode ser muito pequeno, calculamos o seu logaritmo:

$$\log P[O \mid \lambda] = -\sum_{t=1}^{T} \log c_t$$

5.2 Treinamento de HMM

- *Quais os parâmetros $\lambda = (A, B, \pi)$ de um HMM para que $P[O \mid \lambda]$ seja maximizado?, isto é,* $\arg\max_{\lambda} P[O \mid \lambda] = ?$

O treinamento de um modelo estatístico não é, em geral, um problema simples e a existência de um algoritmo eficiente para esse problema é condição fundamental para a aplicabilidade desse modelo estatístico. Esse é o caso de HMM, pois existe o eficiente algoritmo de Baum-Welch, que é um caso particular do algoritmo EM (Expectation-Maximization).

O treinamento de HMM é um problema de otimização contínua de várias variáveis (os parâmetros $\lambda = (A, B, \pi)$) e pode ser resolvido aplicando métodos de otimização baseados no gradiente. Uma opção melhor é o algoritmo de Baum-Welch (ou EM) que consiste dos seguintes passos:

1. Inicialização: Obter parâmetros $\lambda = (A, B, \pi)$ iniciais do modelo. Podemos fazer isso de modo aleatório ou usar algum tipo de estimativa para a inicialização do HMM.

2. Passo **E** (Expectation): Calcular as esperanças matemáticas das variáveis ocultas, dado λ. No caso de HMM, o número de vezes nos estados, transição dos estados e outras variáveis indicadas no passo **M**.

3. Passo **M** (Maximization): Calcular novos parâmetros $\hat{\lambda}$ para que $P[O \mid \hat{\lambda}]$ seja maximizado, assumindo os valores das variáveis ocultas no passo **E**. Se já conhecemos os valores dos

estados ocultos (obtidos no passo **E**), o cálculo dos parâmetros λ do modelo para que $P[O \mid \lambda]$ seja máximo é simples (análogo para cadeias de Markov):

$$\begin{cases} a_{ij} = \frac{\text{número de transições do estado } i \text{ para o estado } j}{\text{número de vezes no estado } i} \\ b_j(k) = \frac{\text{número de vezes no estado } j \text{ com o símbolo } v_k}{\text{número de vezes no estado } j} \\ \pi_i = \text{número de vezes no estado } i \text{ no tempo } t = 1 \end{cases}$$

4. Iteração: Se $P[O \mid \hat{\lambda}] - P[O \mid \lambda]$ é maior que um valor fixo, ir para o passo **E** (com os novos parâmetros $\hat{\lambda}$). Se não, terminar a execução e retornar os novos parâmetros $\hat{\lambda}$.

Pode-se provar que $P[O \mid \hat{\lambda}] \geq P[O \mid \lambda]$ e que a sequência de modelos $\hat{\lambda}_i$ obtidos com o algoritmo EM converge para λ^*, um máximo local de $\arg\max_{\lambda} P[O \mid \lambda]$. O algoritmo de Baum-Welch converge localmente (mas não globalmente) para um máximo. No entanto, Baum-Welch é rápido e geralmente obtemos um bom máximo local.

Para maiores detalhes e a dedução das fórmulas explícitas para os parâmetros do HMM, assim como o Baum-Welch escalado e para observações contínuas (normalmente usado em reconhecimento de voz), consulte [2], [3] ou [5].

6 Implementação

O programa de reconhecimento dos dígitos de 0 a 9 baseado em HMM foi implementado em C e compilado com o Visual C++ sob ambiente Windows. Usamos o sistema de interface iup [10]. Obtemos do iup uma sequência de pontos correspondentes ao movimento da caneta (ou mouse pressionado) sobre a interface. Os dígitos devem ser escritos em um e somente um traço ou curva $C = \{p_1, p_2, \ldots, p_n\}$, onde $p_i = (x_i, y_i)$ é um ponto do plano. O dado de entrada da implementação é uma lista de curvas $L = \{C_1, C_2, \ldots, C_n\}$ como vemos na figura 7, onde os pontos obtidos do sistema de interface estão assinalados (em azul). Note que para os dígitos escritos na figura 7, o programa reconheceu erroneamente somente o dígito 5.

Não fazemos nenhum tipo de suavização ou processamento nessa lista. Só usamos o ângulo (com discretização em 64 partes iguais) entre pontos consecutivos como observação ou dado de entrada para o HMM. Essa observação é representada por $O = O_1, O_2, \ldots, O_{n-1}$ onde O_i é o ângulo entre pontos consecutivos de cada curva. Há dois modos de operação em um HMM: reconhecimento ou treinamento.

Para o reconhecimento em um sistema de HMM é necessário calcular $P(O|\lambda_i)$, que é a probabilidade de ocorrência da observação O (sinal de entrada que queremos reconhecer) num HMM de parâmetros λ_i. Esse HMM de parâmetros λ_i é obtido no treinamento como representativo do dígito i. Calculamos $P(O|\lambda_i)$ com o algorimo forward escalado descrito na seção de HMM. Podemos decidir de qual dígito o dado de entrada O é mais parecido calculando $P(O|\lambda_i)$ para todos os dígitos i. O dígito i^* escolhido é aquele com maior valor de $P(O|\lambda_{i^*})$. Isto é:

$$i^* = \arg\max_i P(O|\lambda_i)$$

Para o treinamento é necessário informar ao sistema qual o dígito que escrevemos. O treinamento de HMM é feito com o algoritmo de Baum-Welch, que nos fornece os parâmetros λ_i^* ótimos

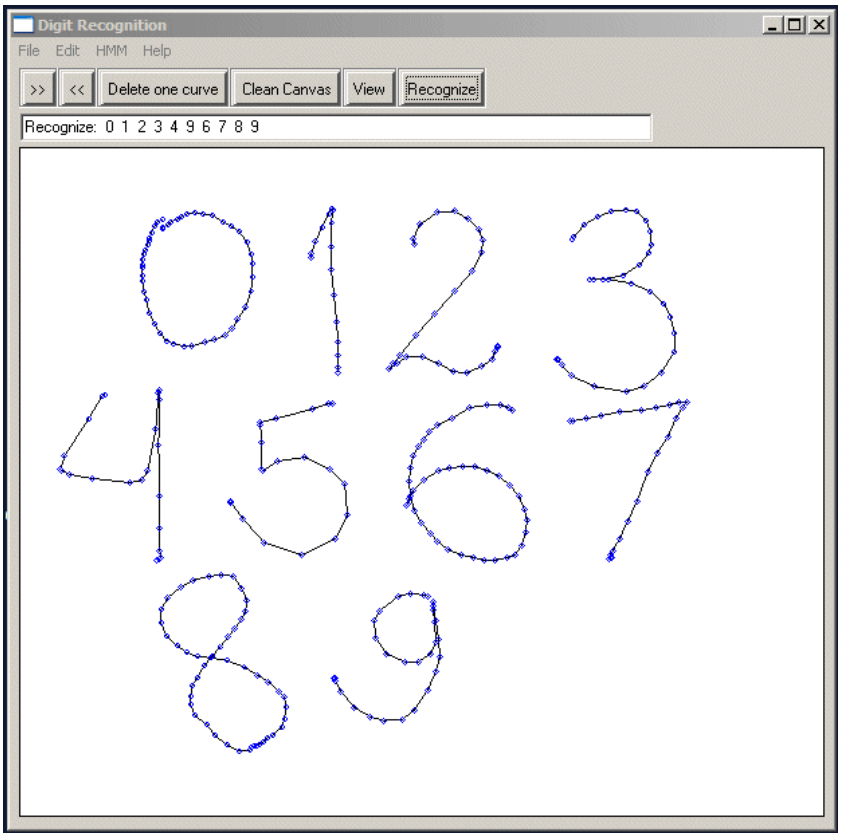

Figura 7: Programa de Reconhecimento de Escrita

para o modelo do dígito i:

$$\lambda_i^* = \arg\max_{\lambda} P(L_i|\lambda)$$

onde L_i é a lista de curvas que informamos serem do dígito i. Cada curva é representada por $O = O_1, O_2, \ldots, O_{n-1}$ onde O_i é o ângulo entre pontos consecutivos.

A inicialização do algoritmo é feita de modo aleatório. Como o algoritmo de Baum-Welch só garante maximização local, fazemos 5 tentativas para selecionar aquela que nos dá maior valor. É necessário fornecer ao algoritmo de Baum-Welch o número de estados do modelo. Para todos os dígitos, arbitramos um modelo de 5 estados.

7 Resultados

Foi feito um teste da implementação usando 4 conjuntos de dados, que denotamos por $A1$, $A2$, B e C. Cada um desses conjuntos contém amostras dos dígitos de 0 a 9. Os conjuntos $A1$ e $A2$ foram escritos por um mesmo usuário A. Já os conjuntos B e C foram escritos por duas outras pessoas.

O conjunto $A1$ foi usado como dado de treinamento. Nas tabelas abaixo indicamos a quantidade de cada dígito no conjunto de dados, assim como a performance (% de acertos) total e

para cada um dos dígitos nos conjuntos de dados.

Símbolo	0	1	2	3	4	5	6	7	8	9	Total
Quantidade	30	30	30	30	30	30	30	30	30	30	288
% de acertos	76.7	96.7	100	100	100	100	93.3	96.7	96.7	100	96

Tabela 1: Resultados para o conjunto de dados $A1$.

Símbolo	0	1	2	3	4	5	6	7	8	9	Total
Quantidade	9	11	10	8	6	9	11	9	6	6	85
% de acertos	55.6	54.6	50	62.5	100	55.6	81.8	77.8	100	83.3	69.4

Tabela 2: Resultados para o conjunto de dados $A2$.

Símbolo	0	1	2	3	4	5	6	7	8	9	Total
Quantidade	47	81	27	30	25	25	30	27	20	17	329
% de acertos	38.3	70.4	48.1	60	64	96	0	33.3	5	100	52.6

Tabela 3: Resultados para o conjunto de dados B.

Símbolo	0	1	2	3	4	5	6	7	8	9	Total
Quantidade	18	16	22	23	22	22	27	17	20	22	209
% de acertos	0	0	63.6	82.6	81.8	100	70.4	0	60	86.4	58.9

Tabela 4: Resultados para o conjunto de dados C.

8 Conclusões

Neste trabalho descrevemos a teoria de reconhecimento de escrita com HMM, que é um método estatístico simples e ao mesmo tempo poderoso, principalmente para modelagem de sinais unidimensionais complexos.

Como já foi dito, a implementação é muito simples. Os maiores problemas dela são:

1. Não tratar de escrita conectada, o que limita o uso em aplicações práticas.

2. Não fazer nenhum tipo de pré-processamento na escrita. Os pontos obtidos na interface não são um bom descritor para a curva (note que não foi usado a distância entre os pontos). Sem processamento algum, o tamanho do dígito, velocidade de escrita ou velocidade do computador alteram a disposição dos pontos obtidos para uma mesma curva.

3. Usamos HMM de 5 estados para cada dígito. Talvez mais estados captem melhor a variabilidade do sinal de escrita, principalmente para curvas mais complexas. Não foi feito nenhum teste para determinar um número ideal de estados para cada dígito.

Figura 8: HMM de estrutura left-right de cinco estados.

4. É permitido qualquer matriz de transição de estados para os HMMs usados na modelagem dos dígitos. A topologia usada para o HMM é geral, com todas as conexões possiveis entre os estados. É comum, tanto em reconhecimento de voz como de escrita, usar um HMM do tipo left-right, cuja matriz de transição de estados tem o elemento $a_{ij} = 0$, se $j > i + 2$ ou $j < i$, como na figura 8.

5. Só usamos o ângulo como feature para o HMM. Poderíamos usar combinação de features ou mesmo várias features (um vetor de features para cada ponto), como em reconhecimento de voz.

6. O sistema só foi treinado com o conjunto $A1$, que consiste de 30 amostras de cada dígito, como vemos na tabela 1 da seção Resultados. Com um conjunto maior de treinamento, provavelmente obteríamos melhor performance no reconhecimento.

Mesmo com todas as simplificações na implementação, os resultados obtidos não são desanimadores, como podemos ver na seção Resultados.

Na tabela 1 (da seção Resultados) vemos que o sistema atinge 96% de acertos para o conjunto $A1$. Uma quantidade alta de acertos era esperado pois o conjunto $A1$ foi usado como dado de treinamento para o HMM. No conjunto $A1$ somente o dígito 0 não atinge um alto percentual de acertos (76.7%). Já para os dígitos 2, 3, 4, 5 e 9 temos 100% de acertos.

Temos 69.4% de total de acertos (tabela 2) para o conjunto $A2$, que foi escrito pela mesma pessoa que escreveu $A1$, mas que não foi usado como dado de treinamento. Nos conjuntos B e C (que não foram escritos pelo usuário A) temos uma quantidade total de acertos de 52.6% e 58.9%, respectivamente. Note que no conjunto de dados $A2$ temos um valor mímino de acertos para o dígito 2 de 50%. Já para os conjuntos B e C temos uma quantidade de acertos de 0% para o dígito 6 (do conjunto B) e para os dígitos $0, 1, 7$ (conjunto C). Isso pode indicar que os usuários B e C escreveram esses dígitos de forma diferente da escrita pelo usuário A, que foi a base para o treinamento.

Um possível trabalho futuro é eliminar algumas das simplificações dessa implementação e analisar a provável melhora de performance no reconhecimento de escrita.

Referências

[1] Anderson Mayrink da Cunha, *Reconhecimento de Voz.* Tech. Report, IMPA, 2003.

[2] Frederick Jelinek, *Statistical Methods for Speech Recognition.* MIT Press, 1997.

[3] Lawrence Rabiner and Biing-Hwang Juang, *Fundamentals of Speech Recognition,* Prentice-Hall, 1993.

[4] Claudio Becchetti and Lucio Prima Ricotti, *Speech Recognition,* John Wiley&Sons, 1999.

[5] M. C. Nechyba, *Spring 2000 Lecture Notes - EEL6935: Machine Learning in Robotics II*, 2000, http://www.mil.ufl.edu/~nechyba/readings_s00.html

[6] Jianying Hu, Michael Brown and William Turin, *"HMM Based On-Line Handwriting Recognition"*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 18, n. 10, October, 1996.

[7] Jianying Hu, Sok Gek Lim and Michael Brown, *"HMM Based Writer Independent On-Line Handwriting Character and Word Recognition"*, 1998, http://citeseer.nj.nec.com/hu98hmm.html

[8] Homayoon S. M. Beigi, Krishna Nathan and Jayashree Subrahmonia, *"On-Line Unconstrained Handwriting Recognition Based on Probabilistic Techniques"*, 1995, http://citeseer.nj.nec.com/209795.html

[9] G. Rigoll, A. Kosmala and D. Willett, *"A New Hybrid Approach to Large Vocabulary Cursive Handwriting Recognition"*, Int. Conference on Pattern Recognition (ICPR), 1998, http://citeseer.nj.nec.com/rigoll98new.html

[10] Sistema de Interface iup, http://www.tecgraf.puc-rio.br/iup/