Classificação Hierárquica de Caracteres Matemáticos

Breno Flesch Franco[1], Ricardo Alexandre Bastos[2] *
Orientadora: Profa. Dra. Nina Sumiko Tomita Hirata
Instituto de Matemática e Estatística
Universidade de São Paulo

9 de Fevereiro de 2010

*bflesch@linux.ime.usp.br, rabastos44@gmail.com

Conteúdo

1 Introdução

A produção de textos científicos e artigos que utilizam fórmulas e expressões matemáticas em textos eletrônicos sempre foi uma atividade difícil, dada a falta de boas ferramentas para a criação de fórmulas em programas para edição de texto. Com a criação do *TeX* [1] e do *LaTeX* essa atividade se tornou mais fácil, mas a linguagem *TeX* é grande e complexa. O objetivo da ferramenta Express-Math é permitir a escrita natural de expressões matemáticas, usando reconhecimento por computador tanto para os caracteres quanto para a estrutura da fórmula.

1.1 Reconhecimento de Expressões Matemáticas

Para converter uma fórmula manuscrita para sua representação em *LaTeX* devemos seguir os seguintes passos:

Captura dos caracteres é a leitura das informações da entrada, ou seja, a captura do traçado do caractere e dos pontos que o compõem.

Reconhecimento dos caracteres é a conversão dos conjuntos de pontos que representam um caractere para a sua codificação eletrônica em LaTeX.

Estruturação da fórmula é a montagem da fórmula, isto é, qual caractere é subscrito a outro, quais os caracteres sob a raiz e qual o papel de alguns caracteres por exemplo símbolo Σ representa um somatório ou a letra grega sigma maiúscula.

Esse trabalho se concentra nas duas primeiras etapas, as de captura e reconhecimento dos caracteres manuscritos.

1.2 Reconhecimento de Caracteres

Reconhecimento de caracteres é a conversão de caracteres manuscritos para alguma codificação eletrônica. Essa atividade tem como objetivo facilitar o trabalho com caracteres nos meios eletrônicos, ao permitir que pessoas utilizem fluidamente a habilidade de escrita que possuem, sem ter de fazê-las aprender meios específicos de inserção dos caracteres, possibilitando maior produtividade na produção de textos eletrônicos.

Em particular, este trabalho versa sobre o reconhecimento online de escrita. Nesse tipo de sistema, o usuário escreve em um dispositivo de entrada diretamente acoplado ao sistema, contrastando com o reconhecimento offline, o qual ocorre quando se realiza o reconhecimento de imagens de um texto previamente escrito. O reconhecimento online, portanto, permite o uso de informações relacionadas ao tempo, aumentando o espectro de dados disponível sobre os caracteres escritos.

Várias técnicas computacionais para a classificação de caractere baseiam-se em características - medidas numéricas extraídas a partir dos resultados da captura da escrita do usuário.

O problema de reconhecimento de caracteres é bastante estudado, tanto com conjuntos de caracteres latinos quanto com linguagens ideogramáticas, e tem ganhado força recentemente com a proliferação de aparelhos de consumo com interface sensível a toque. O reconhecimento de expressões matemáticas, no entanto, é consideravelmente menos estudado e mais complexo do que o reconhecimento de escrita monolingue.

A complexidade do reconhecimento de caracteres matemáticos se deve, em sua maioria, a dois pontos principais: o grande número de símbolos que devem ser reconhecidos, o que aumenta a complexidade do problema, e a falta de restrições de contexto dos caracteres, o que impede que técnicas comuns de inferência possam ser aplicadas. Quando todo caractere pode, potencialmente, ser de qualquer uma das classes disponíveis para o usuário, não podemos ignorar nenhuma classe de caracteres durante o reconhecimento.

1.3 Tópicos abordados

Nos seções posteriores iremos detalhar os seguintes temas:

Caracteristicas nesse capitulo detalha quais caracteristicas estamos usando na implementação e como são coletadas

Conceitos Básicos onde são detalhadas as técnicas e os termos usados em Reconhecimento de caracteres.

BHC a sessão onde são descritos detalhes o algoritmo do BHC

Implementação Essa sessão contém breves descrições das principais classes no desenvolvimento do projeto.

Teste e Resultados obtidos são detalhados nessa sessão.

2 Características

Antes de definirmos características, detalhamos o formato de representação dos caracteres.

Os dados coletados de cada símbolo são conjuntos de pontos no Z^3. As duas primeiras dimensões são as coordenadas espaciais do ponto. A terceira coordenada indica o momento de sua captura.

O tamanho do conjunto de pontos que define um caractere não é fixo; nem sequer é limitado por um número. Para poder padronizar esses dados, utilizamos a abordagem de reconhecimento por vetor de características (*feature vector*) para tratar esse conjunto de pontos de maneira a resumir as informações obtidas para obter um formato com o qual se possa trabalhar.

As características escolhidas na implementação atual estão detalhadas abaixo.

2.1 Densidade de Pontos

Uma característica simples, mas muito utilizada no reconhecimento de caracteres. Consiste em sobrepor uma grade com certa quantidade de divisões, regulada por um parâmetro g, chamado de *granularidade*. Na implementação corrente, é utilizada uma grade de $g \times g$ espaços com $g = 4$. O retângulo circunscrito ao caractere é então dividido em g^2 regiões retangulares de mesmo tamanho, e o número de pontos dentro de cada região é contado.

Essa técnica gera um vetor de tamanho g^2 que indica as regiões com maior ou menor concentração de traços; é interessante citar que com uma granularidade suficientemente alta, temos um ponto em cada região, recriando assim o caractere original. Entretanto, esta característica permite a diferenciação de caracteres com uma medida-resumo facilmente configurável.

2.2 Direções principais

Dado o conjunto de pontos do caractere, calculamos o vetor direção entre um ponto e seu sucessor cronológico, montando um novo vetor que contém as direções extraídas do caractere, ordenadas pela coordenada temporal. Após calculado o vetor de direções, nós o dividimos em k setores e calculamos a média das direções dentro de cada setor. Desta forma, capturamos as k principais direções do caractere. Na implementação corrente, $k = 3$.

2.3 Contagem de Laços

O número de laços em um caractere é uma informação resumo importante: a presença ou ausência de um laço permite a diferenciação de caracteres por seres humanos de maneira rápida, mesmo quando a caligrafia do escritor não é facilmente reconhecida. Para detectar um laço basta encontrar uma posição onde ocorra uma sobreposição de pontos de um mesmo traço. A consistência em traçar um caractere com laço ou não é bastante grande para um mesmo escritor.

2.4 Contagem de Quinas

Da mesma forma, a contagem de quinas - pontos onde há a formação de ângulos agudos no traço do caractere - é uma medida-resumo interessante: há carac-

teres que não exibirão quinas em hipótese alguma: aqueles que são traçados com linhas retas ou curvas suaves. Isso permite que sejam classificados mais facilmente, ao resumir uma propriedade geral do traço do caractere em uma única característica numérica.

3 Conceitos Básicos

Os principais conceitos necessários para a compreensão do reconhecedor implementado estão detalhados nesta seção, bem como algumas técnicas utilizadas. Outras técnicas e abordagens são comentadas para contextualização.

3.1 Problemas de Classificação

Um problema de classificação é, dado um conjunto de classes $\Omega = \{\omega_1, \omega_2, \ldots \omega_c\}$ e uma amostra coletada, encontrar a qual classe a amostra pertence.
Os algoritmos de classificação, também chamados de classificadores, utilizam um conjunto de amostras pré-classificadas, o conjunto de treinamento, para extrair informações sobre as classes, e a partir dessas informações tentam inferir a qual classe pertence a amostra a ser classificada.

3.2 Classificadores Binários

Classificadores binários são aqueles que cumprem a tarefa de classificar um elemento entre duas categorias, apoiados em exemplos de elementos previamente classificados.

Existem várias implementações de classificadores binários: redes neurais [2], classificador de Bayes, Máquinas de Vetores de Suporte (SVMs, do Inglês Support Vector Machines) [3], entre outras técnicas.

3.2.1 Redes Neurais

Uma rede neural é composta por neurônios conectados uns aos outros, onde cada neurônio representa um hiperplano diferente, e a rede representa um a superfície de decisão cujo formato depende da função de ativação de cada neurônio. O algoritmo funciona ajustando os pesos das funções de cada neurônio, alterando o hiperplano por ele representado, e tentando minimizar o erro dentro do conjunto de amostras de treinamento. [3]

3.2.2 Máquinas de Vetores de Suporte

Máquinas de Vetores de Suporte mapeiam os vetores de entrada em um espaço de características com dimensão alta, através de alguma função não linear escolhida a priori. Nesse novo espaço, uma superfície de decisão é construída com vetores que definem um hiperplano nesse novo espaço.

3.2.3 Classificador de Bayes

Seja ω uma classe e x uma amostra a ser classificada, então, apoiados no teorema de Bayes:

$$p(\omega|x) = \frac{p(\omega)p(x|\omega)}{p(x)},$$

onde definimos $p(\omega)$ como a probabilidade da classe - a frequência relativa da classe, definida pelo problema, $p(x|\omega)$ como a probabilidade de uma amostra pertencer a determinada classe - isto é, uma medida de verossimilhança da amostra x como pertencente à classe ω. Tal distribuição de probabilidade da classe pode ser encontrada usando estimadores de seus parâmetros e por estimação de máxima verossimilhança a partir de um conjunto de amostras, e $p(x)$ é a probabilidade da amostra - a frequência relativa dessa amostra em relação a todas as amostras possíveis.

É razoável considerarmos que a classe a qual x pertence é aquela ω que maximiza a probabilidade a posteriori $p(\omega|x)$. Logo, podemos utilizar um classificador binário baseado nessa probabilidade:

$$C = p(\omega_1|x) - p(\omega_2|x),$$

Se C for positivo, então x pertence à classe ω_1 e pertence a ω_2 caso contrário.

3.3 Classificação Multiclasses

Um classificador multiclasses é semelhante a um classificador binário, com a diferença que, em vez de classificar entre "pertencente à classe" e "não pertencente à classe", o classificador realiza a diferenciação entre várias classes.

Utilizar classificadores binários para a classificação multiclasses é uma abordagem comum, podendo ser feita através de adaptações nos algoritmos binários ou da decomposição do problema multiclasse em vários problemas binários.

O desenvolvimento de um algoritmo multiclasses a partir de um algoritmo binário nem sempre é possível ou fácil de se implementar. A divisão do problema em subproblemas binários é mais comum e pode ser feita de diversas maneiras [4].

Uma forma de fazê-lo é através de classificação hierárquica, equivalente a uma árvore, cuja raiz representa o conjunto de todas as classes, e o conjunto de filhos de cada nó compõe uma partição do conjunto de classes desse nó. É fácil ver que é possível usar classificadores binários para dividir um conjunto de classes em dois subconjuntos menores, recorrendo-se a esse mesmo método até o problema ser reduzido a uma série de problemas de classificação binária. Foi essa a escolha feita neste trabalho.

3.4 Análise de discriminante linear

Discriminante lineares são métodos utilizados pela áreas de Estatística e de Aprendizado Computacional para encontrar uma combinação linear das características que maximize a separação entre as distribuições das classes.

O discriminante escolhido para a implementação foi o Discriminante de Fisher, pela simplicidade de sua ideia. O discriminante utiliza duas matrizes de relações entre as classes, as matrizes *entre-classes* $B_{\omega,\omega'}$ e matriz intra classes $W_{\omega,\omega'}$, definidas abaxo.

$$B_{\omega,\omega'} = (\mu_\omega - \mu_\Omega)(\mu_\omega - \mu_\Omega)^t + (\mu'_\omega - \mu_\Omega)(\mu'_\omega - \mu_\Omega)^t$$

onde μ_ω é a média da classe ω e μ_Ω é a média de todas as amostras

$$W_{\omega,\omega'} = \Sigma_\omega + \Sigma_{\omega'}$$

A separação da classes em uma direção ψ é dada por:

$$S = \frac{\psi B \psi^t}{\psi W \psi^t}$$

Para utilizarmos o discriminante de Fisher temos que adaptar o discriminante para trabalhar com conjuntos de classes além de levar em conta as associações entre classes e metaclasses. Para isso, temos que calcular a probabilidade, média e a matriz de covariancia das metaclasses, estamos supondo que toda metaclasse tem distribuição normal. Seja Ω uma metaclasse e ω_i $i = 1, 2, ..n$ todas pertencentes a Ω e a^Ω_ω, então a probabilidade de Ω é dada por :

$$P(\Omega) = \sum_{w_i} p(w_i) a^\Omega_{w_i}, i = 1, 2, ...3$$

a média deΩ é dada por

$$\mu_\Omega = \sum_{w_i} P(w_i|\Omega)\mu_{w_i}, i = 1, 2, ...3$$

a matriz de covariancia pode ser calculada como abaixo

$$\Sigma^\Omega = \sum_{w_i} \frac{P(w_i|\Omega)}{N_w} [\sum_{x \in \chi_w} (x - \mu_\Omega)(x - \mu_\Omega)^t]$$

4 Classificador binário hierárquico

O BHC [6] utiliza a abordagem de divisão-e-conquista para tratar o problema de classificação entre C classes como $C - 1$ problemas de classificação entre duas metaclasses. Dessa forma, permite o aprendizado em módulos (cada um dos problemas entre metaclasses corresponde a um módulo de aprendizado, possibilitando o uso de uma solução que mais se adeque ao problema local) e apresenta uma boa razão entre taxa de acerto e tempo despendido em treinamento e classificação.

4.1 Arquitetura

O classificados é definido como uma árvore binária onde:

- a raiz representa o conjunto de todas as classes, Ω, ou seja, Ω_0
- cada nó interno n contém uma metaclasse (um conjunto de classes $\Omega_n \subset \Omega$), um extrator de características ψ_n e um classificador ϕ_n, possuindo dois nós filhos ($2n$ e $2n+1$, cujas metaclasses compõem uma partição de Ω_n),
- as folhas representam as classes $\omega_i (i = 1, 2, \cdots, n)$.

Para obter o classificador proposto são necessárias pelo menos duas etapas. A construção da árvore de metaclasses e a preparação dos classificadores de cada nó.

Para cada nó temos que encontrar quais características diferenciam as duas metaclasses filhas e criar novos extratores para essas características.

Para a construção da hierarquia de características que otimize a classificação, é necessário um conhecimento profundo da distribuição dos símbolos. Encontrar a melhor hierarquia é um problema díficil. Utilizamos uma abordagem gulosa, na qual um discriminante linear é usado para obter um espaço onde se pode medir a separação imposta às metaclasses por um conjunto de características, e selecionar o conjunto de características que melhor as separam.

Com a hierarquia montada e os extratores de características definidos, basta escolhermos alguma técnica de classificação e treiná-la para classificar entre os filhos de cada um dos $C - 1$ nós internos.

4.2 Classificador

A implementação do BHC pode ser feita em três etapas: o extrator de características, a construção da árvore taxonômica e o classificador interno de cada nó.

4.2.1 Extrator de características

Na escolha das características que serão utilizadas em cada nó e na construção da hierarquia de classes que otimiza a classificação, é necessário conhecimento da distribuição das características através dos símbolos. Encontrar tal hierarquia é um problema bastante difícil. Para simplificá-lo, é usado um discriminante linear, que permite calcular rapidamente a distância entre duas metaclasses, dado um conjunto de características. O discriminante linear escolhido foi o discriminante de Fisher, adaptado para trabalhar com conjuntos de classes, além de

levar em conta as associações entre classes e metaclasses.

4.2.2 Discriminante de Fisher

O discriminante de Fisher depende da probabilidade, média e matriz de covariância das metaclasses.

Supondo que toda metaclasse tem distribuição normal, seja Ω uma metaclasse, para todas as classes ω_i $i = 1, 2, ..n$ pertencentes a Ω, e considerando $a^{\Omega}_{\omega_i}$ como a associação entre a classe $omega_i$ e a metaclasse $Omega$, temos que a probabilidade de Ω é dada por :

$$P(\Omega) = \sum_{w_i} p(w_i) a^{\Omega}_{w_i}, i = 1, 2, ...3$$

a média de Ω é dada por

$$\mu_{\Omega} = \sum_{w_i} P(w_i|\Omega)\mu_{w_i}, i = 1, 2, ...3$$

a matriz de covariância de de Ω é calculada como

$$\Sigma^{\Omega} = \sum_{w_i} \frac{P(w_i|\Omega)}{N_w} [\sum_{x \in \chi_w} (x - \mu_{\Omega})(x - \mu_{\Omega})^t]$$

4.2.3 buildTree

Dada a estrutura de árvore do BHC, a maneira mais simples de construí-la é através de recursão. O método *buildTree* recebe um conjunto de classes *mc*, na posição t da árvore. Se esse conjunto contém mais que uma classe, nós o particionamos, adicionamos ambas as suas partições α e β na árvore, nas posições $2t+1$ e $2t+2$, respectivamente, e recorremos. Caso contrario o algoritmo para.

```
1  private void buildTree(MetaClass mc, int t) {
2
3          if (mc.getClasses().size() > 1) {
4              partitionNode(mc, alpha, beta);
5
6              tree.put(2 * t + 1, alpha);
7              tree.put(2 * t + 2, beta);
8
9              buildTree(alpha, 2 * t + 1);
10             buildTree(beta, 2 * t + 2);
11         }
12     }
```

O núcleo do BHC, portanto, é o método *partitionNode*, que recebe como parâmetro uma metaclasse e devolve sua partição. Dessa forma conseguimos encontrar uma hierarquia de classes de maneira automatizada.

4.2.4 partitionNode

O *partitionNode* recebe um conjunto de classes, uma metaclasse Ω_t, e a separa em duas metaclasses Ω_{2t+1} e Ω_{2t+2}.

O método é composto por seis etapas: inicialização das associações, busca do espaço de características que mais separa as metaclasses, computar a verossimilhança, atualização das associações, checagem de convergência e checagem da entropia.

Para explicar melhor essas etapas precisamos definir *associação*. Seja Ω um conjunto de classes e $\omega \in \Omega$, então a associação a^{ω}_{Ω} é definida como $P(\Omega|\omega)$

1. **Iniciando as associações** Uma classe tem sua associação fixada em 1 em relação a uma metaclasse Ω_{2t+1}, enquanto as outras são associadas igualmente entre as duas metaclasses e iniciamos a temperarura T com 1. Essa associação determinística e não simétrica só pode ser feita porque o objetivo do algoritmo é a divisão em duas metaclasses e não em 3 ou mais metaclasses.

2. **Encontrar o espaço de características** $\mathcal{F}(\Omega_{2t+1}, \Omega_{2t+2})$ que mais separa as metaclasses. Esse espaço deve ser criado para as associações atuais das metaclasses, de modo que ao se interromper as iterações, o espaço armazenado na árvore é o que melhor divide as metaclasses disjuntas. Para gerar $\mathcal{F}(\Omega_{2t+1}, \Omega_{2t+2})$ e stamos utilizando um o discriminante de Fisher para essa etapa.

3. **Computar a verossimilhança** usamos a função abaixo para calcular a log-verossimilhança média da classe dada uma metaclasse.

$$\mathcal{L}(\omega|\Omega_\rho) = \frac{1}{N_\omega} \sum_{x \in \chi_\omega} log(p(\psi(x|A)|\Omega_\rho)), \rho \in \{2t+1, 2t+2\}, \forall \omega \in \Omega,$$

   A verossimilhança pode ser interpretada como o custo da associação $a^{\omega}_{\Omega_\rho}$. E $\psi((x|A)|\Omega_\rho)$ pode ser modelado como uma distribuição de probabilidade qualquer. No caso, estamos usando uma Gaussiana.

4. **Atualizar as associações** é na verdade minizar a função objetiva,

$$\mathcal{J}(\omega) = \sum_{\rho \in \Omega_2 t+1, \Omega_2 t+2} a^{\omega}_{\Omega_\rho} \mathcal{L}(\omega|\Omega_\rho) - T(\mathcal{H}(\omega) - \mathcal{H}_T),$$

   onde T é a temperatura da iteração atual e $\mathcal{H}(\omega)$ é a entropia da classe ω, definida na etapa 6. O segundo termo é usado para forçar que a entropia diminua junto com a temperatura. Isso evita que a precisão do algoritmo piore durante as iterações.

   Para miniminizar a função $\mathcal{J}(\omega)$, sabemos que é suficiente maximizar a função abaixo:

$$a^{\omega}_{\Omega_\rho} = \frac{exp(\mathcal{L}(\omega|\Omega_\rho/T)}{\sum_{\rho \in \{2t+1, 2t+2\}} exp(\mathcal{L}(\omega|\Omega_\rho/T}, \forall \rho \in \{2t+1, 2t+2\},$$

fazemos isso para toda classe ω e teremos as associações atualizadas.

5. **Checar a convergência** da função objetivo $\mathcal{J}(\omega)$, se o incremento em $\sum_{\omega_\Omega} \mathcal{J}(\omega)$ for maior que um limite definido pelo usuário (na nossa implementação 10%), repetimos as etapas 1 a 4, até alcançarmos a convergência desejada.

6. **Medição da entropia** A entropia é definida como a medida de divisão da associação da classe entre os subconjuntos de classes. E é calculada por:

$$\mathcal{H}(\omega) = -\sum_{\rho \in \{2t+1, 2t+2\}} a^{\omega}_{\Omega_\rho} log(a^{\omega}_{\Omega_\rho})$$

uma entropia alta indica que a classe está associada igualmente a duas metaclasses. O objetivo do algoritmo é minimizar essa entropia. Se a somatória das entropias de todas as classes está acima do limite estipulado (na implementação, 0,01), diminuímos a temperatura T e executamos os passos de 1 a 5 novamente.

Ao atingirmos a entropia desejada, as classes estão particionadas entre as duas metaclasses com associações próximas a 1 ou a 0. Desta maneira, podemos dividir o subconjunto inicial Ω_t de Ω nos dois subconjuntos Ω_{2t+1} e Ω_{2t+2}.

4.3 Classificação

Para a classificação, desejamos saber $P(\omega|x)$, onde x é uma amostra que queremos classificar, no caso algum caractere escrito a mão. Utilizamos o classificador de Bayes, definido abaixo:

$$\phi_t(x) = p(\Omega_t|x) = \frac{p(\Omega_t)p(x|\Omega_t)}{p(x)},$$

onde $p(\Omega_t) = \frac{|\Omega_t|}{|\Omega|}$, e $p(x|\Omega_t)$ possui uma função densidade de probabilidade definida (no caso, uma Gaussiana). Para $p(x)$ supomos uma distribuição uniforme para as amostras. Mas, antes de definirmos $P(\omega|x)$, temos que classificar x nos diferentes subproblemas da arvore hierárquica, para isso, devemos projetar as características de x no espaço de características ψ_t, ou seja, classificamos $\psi_t(x)$. Usando a função abaixo,

$$\mathcal{C}(x,t) = \phi_{2t+1}(\psi(x))p(\Omega_{2t+1}) - \phi_{2t+2}(\psi(x))p(\Omega_{2t+2})$$

se $\mathcal{C}(x,t) > 0$, classificamos x como Ω_{2t+1} e se $\mathcal{C}(x,t) < 0$ classificamos x como Ω_{2t+2} e repetimos o processo para a metaclasse na qual x foi classificado até ser classificado entre folhas.

4.4 Linguagem e Bibliotecas

A implementação do projeto foi feita em Java, com o auxilio das bibliotecas:

Jama , *http://math.nist.gov/javanumerics/jama/* biblioteca para a manipulação de matrizes, com métodos para criar decomposições de matrizes.

Swt , *http://www.eclipse.org/swt/* para a criação e alteração da interface

Dom4J , *http://sourceforge.net/projects/dom4j/* para manipulação de arquivos *XML*, utilizados para o armazenamento dos caracteres.

5 Implementação

Esta seção contém detalhes da implementação do classificador de caracteres e das estruturas do Math-Picasso alteradas e utilizadas. Além disso, as novas classes para a criação do classificador.

5.1 Estruturas do Math-Picasso

As classes aqui descritas são apenas aquelas as utilizadas pelo classificador implementado, como o Projeto Math-Picasso contem diversas classes e interfaces que não foram alteradas ou usadas no Express-Math.

5.1.1 Symbol

A classe *Symbol* é responsável por armazenar a representação gráfica de um caractere escrito pelo usuário utilizando as estruturas:

- Uma lista de objetos *Stroke* que representam os vários traços do caractere. O objeto *Stroke* é contem a lista de objetos *StrokePoint* e os limites do caractere, isto é, as coordenada do retângulo que circunscreve o traço.
- A classe *StrokePoint* representa um ponto (x, y, t) onde x e y são as coordenadas e t o momento de captura desse ponto.
- *BoundingBox* é a classe que guarda os limites do retângulo que circunscreve um traço ou um caractere.

5.1.2 Character

Essa classe tem como responsabilidade permitir a criação das representações em LaTeX dos diferentes caracteres aceitos. Um símbolo associado a uma instância da classe *Character* é a representação de um símbolo classificado.

5.1.3 CharSymbol

A classe abstrata *CharSymbol* em um objeto da classe *Symbol* para armazenar o caractere capturado, a *BoundingBox* desse caractere e um objeto da classe *Character* com a classificação do caractere.

É responsável por gerar novos exemplos com pertubações a partir do simbolo atual.

Também contém as informações necessárias para as informações para a estruturação da fórmula.

5.1.4 Recognizer

A interface utilizada para acoplar novos reconhecedores tem cinco métodos *recognize*, *recognizeAll*, *train*, *addAndTrain*, *addCharSymbol*.

5.2 Classes do Reconhecedor

5.2.1 Classe

A classe *Classe* o conjunto de treinamento de uma determinada classe. É responsável pelo cálculo da média das características e da matriz de covariância da classe.

O cálculo da média da matriz de covariância são armazenados para evitar que sejam recalculadas.

5.2.2 MetaClass

Objetos da classe *MetaClass* representam conjuntos de classes, também conhecidos como metaclasses. Esta classe é responsável pelo cálculo da média das características e da matriz de covariância da união dos conjuntos de treinamento de cada classe pertencente a metaclasse, e a associação entre cada classe e a metaclasse.

Também é responsável por armazenar o discriminante de Fisher que melhor separa as classes suas metaclasses derivadas da separação. As metaclasses filhas não são armazenadas, pois estão definidas na árvore de classificação.

5.2.3 Gaussian

A classe *Gaussian* recebe o vetor de médias, μ_ω, e a matriz de covariância, Σ_ω. É responsável por calcular a probabilidade de uma caractere em uma distribuição normal de média μ_ω e de variância Σ_ω

5.2.4 FisherDiscriminant

Recebe como parâmetros de criação duas metaclasses e o número de características. E calcula a projeção que melhor separa as duas metaclasses com as suas associações atuais, conforme descrito na sessão 3.4.

5.2.5 UnclassifiedCharSymbol

A classe UnclassifiedCharSymbol foi criada para fins de integração com o trabalho de estruturação de expressões matemáticas desenvolvido em paralelo pelos alunos Ricardo Sider e Bruno Yoiti Ozahata. É subclasse de CharSymbol, e responsável pela representação de um símbolo ainda não classificado.

5.2.6 Extractor

Uma classe estática responsável pelo cálculo de características de um símbolo. As características coletadas são as definidas na seção 2. Quando são requisitadas as características de um CharSymbol este delega a geração dessas características para a classe *Extractor*.

5.2.7 Bhc

A classe principal do Reconhecedor de Caracteres, implementa a interface *Recognizer* . Armazena cada uma das metaclasses na árvore de classificação e é responsável por fazer a classificação dos caracteres.

Como é responsável pela construção da árvore de classificação, contém as funções descritas na sessão 4.2.3.

5.3 Alterações no Math-Picasso

Mudanças foram feitas na interface do Math-Picasso para que se tornasse mais simples e intuitiva:

- A eliminação da aba de treinamento.
- A inclusão de um botão para a classificação manual dos caracteres.
- A criação de perfis de usuário.

6 Testes e Resultados

O classificador binário hierárquico foi testado, primeiramente, com um conjunto de testes padrão, que consiste de 55 símbolos distintos coletados duas vezes, com 10 escritores diferentes.

Esse conjunto, obtido por Cristiano Garcia para seu reconhecedor, feito para o projeto ExpressMath, [5] foi escolhido com a intenção de permitir uma comparação justa entre as implementações do BHC e os resultados obtidos no trabalho anterior.

O teste inicial (de número 3 na tabela abaixo) obteve 40% de acerto com esse conjunto, separando 10 amostras aleatoriamente para serem usadas como teste contra os 10 exemplos de treinamento. O classificador implementado por Garcia conseguia resultados de até 81% de acerto ao classificar esses caracteres [5], usando distorções dos originais como conjunto de treinamento.

Como as técnicas de treinamento são bastante diferentes, é difícil fazer uma comparação plenamente adequada, mas a abordagem escolhida no trabalho anterior possibilita o treinamento com pouquíssimas amostras. Já o treinamento realizado neste trabalho é mais simples, e requer uma maior quantidade de amostras para obter um desempenho adequado.

Teste	No. de Classes	No. De Escritores	Amostras de treinamento	Amostras de teste	Taxa de Acer
1	8	1 / mouse	18/classe	10 a 30/classe	63%
2	15	2/ tablet	18/classe	5/classe	54%
3	37	10/ tablet	10/classe	10/classe	40%
4	34	1 / tablet	8/classe	12/classe	74%
5	50	1 / tablet	8/classe	42/classe	77%
6	70	1/ tablet	8/classe	42/classe	75%

Tabela 1: Testes executados com a técnica de validação cruzada

Além desse conjunto de testes, foram realizados outros, relacionados na tabela a seguir:

Os testes foram realizados com validação cruzada por amostragem aleatória, na qual o conjunto de treinamento para cada classe é escolhido aleatoriamente, o classificador é treinado com esse conjunto e a taxa de acerto do classificador é medida. O processo é repetido diversas vezes, e é considerada para efeito de desempenho a taxa de acerto média do classificador. Para os testes de 1 a 3, foram usadas dez repetições; para os testes 4 e 6, trinta repetições, e para o teste 5, cem repetições.

Os resultados obtidos com essas amostras indicam que o BHC tem bom desempenho na classificação dos diversos símbolos, mesmo quando o número de classes é elevado, com um conjunto de dados consistente e uma amostragem de tamanho razoável, mas tem dificuldade em acertar a classificação nos três primeiros testes, nos quais a variância das características é maior (no teste número 1, foi usado um mouse, menos adequado para a escrita do que a tablet, e nos outros dois testes, a quantidade) e com mais de um escritor acabam piorando o desempenho do classificador, como ocorre nos três primeiros casos. Isso ocorre porque é mais difícil de manter baixa a variância das características nessas ocasiões.

6.1 Aumento do número de classes

Conforme o número de classes ($|\Omega|$) aumenta, cresce também a probabilidade de erro de classificação: cada classe cobre apenas $\frac{1}{|\Omega|}$ do espaço, supondo uma distribuição uniforme das classes. Considerando que, para haver erro de classificação, basta que as características extraídas indiquem uma maior associação com alguma das outras classes, essa ocorrência é mais frequente quanto maior for o número de classes disponível.
Entretanto, o BHC responde bem à alteração do número de classes, como pode ser observado nos resultados dos testes, feitos com validação cruzada baseada em separação aleatória entre conjunto de testes e conjunto de treinamento.

Quanto ao conjunto completo de classes necessárias para a escrita de expressões matemáticas, não temos como afirmar que o desempenho do BHC ainda se mantém satisfatório, mas os resultados indicam um queda muito pequena na

taxa de acerto em relação ao aumento do número de classes.

6.2 Presença de diferentes escritores

Pessoas diferentes escrevem de maneira diferente, e conforme as peculiaridades de cada escritor são expressas nas características extraídas, conjuntos de dados consistindo em símbolos caligrafados por diversos escritores diferentes tendem a ter uma variância muito maior nas características, o que resulta em uma classificação menos precisa. Entretanto, classificadores que atuem bem mesmo em conjuntos de dados pouco uniformes são altamente desjáveis.
Não há dados suficientes para concluir que o desempenho do BHC com diversos escritores é pior ou melhor do que outros reconhecedores, mas os dados indicam que há uma queda acentuada de desempenho conforme a consistência da escrita diminui, como já era esperado.

7 Conclusão

O reconhecimento de caracteres é uma ferramenta com grande utilidade, mas há falta de projetos que visem a produção de textos científicos e há bastante espaço para ferramentas úteis para o gênero. Dentro do escopo do projeto Express-Math, tentamos propor uma ferramenta que seja capaz de suprir essa necessidade.

O reconhecimento de escrita, foco deste trabalho, é um grande desafio. Mesmo tendo diversos trabalhos sobre o tema, poucos se concentram no problema de reconhecimento matemático e utilizam técnicas de inferência possíveis dentro da estrutura de um texto.

O BHC visa reduzir a complexidade do problema usando uma abordagem hierárquica. Para esse fim, utiliza diversas técnicas de aprendizado de máquina para cada uma de suas etapas e demonstra uma grande capacidade para a adaptação do algoritmo central. Mesmo não alcançando os resultados desejados, ainda há diferentes técnicas de abordagem hierárquica que podem ser usadas para criar um novo reconhecedor. Talvez a utilização de classificadores que não utilizem informações da árvore gerada produzirão resultados melhores, mas ao mesmo tempo seria necessária uma outra etapa de treinamento depois da geração da árvore de características.

Um dos problemas encontrados foi a necessidade de um grande número de amostras para um treinamento que alcance o resultado desejado. Mas isso é esperado de um problema de classificação com grande quantidade de classes.

Além disso, a construção automatizada da hierarquia obscurece o conhecimento de qual característica tem maior influência na divisão de classes, o que é uma característica ruim, especialmente se a taxonomia das classes é uma in-

formação desejada. Outro problema é que a estrutura não possui treinamento incremental deixando o sistema menos intuitivo para o treinamento durante sua utilização.

Os teste mostraram que o BHC é um classificador com bons resultados para problemas com um número de classes grande, com um único escritor. O tempo de treinamento do algoritmo é rápido, comparado a outras abordagens, e por causa de sua estrutura de árvore pode ser paralelizado de maneira simples.

Referências

[1] N. H. F. Beebe. 25 years of tex and metafont: Looking back and looking forward. *TUGboat*, 2004.

[2] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford : Clarendon, 1996.

[3] V. V. Corina Cortes. Support-vector networks. *Machine Learning*, 1995.

[4] A. C. L. e Andre C. P. L. F. de Carvalho. Estratégias para a combinação de classificadores bináriosem soluções multiclasses. *Revista de Informática Teórica e Aplicada, Vol. 15, No 2*, 15(2):65–86, 2008.

[5] C. P. Garcia. Uma abordagem hierárquica para o reconhecimento de caracteres em expressões matemáticas manuscritas. 2008.

[6] S. Kumar. *Modular learning through output space decomposition*. PhD thesis, 2000. Supervisor-Ghosh, Joydeep and Supervisor-Crawford, Melba M.