**Detecção de rasuras em símbolos com
aprendizado Bayesiano de programas (BPL)**

Raphael Davis de Oliveira Costa

DISSERTAÇÃO APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO
PARA
OBTENÇÃO DO TÍTULO
DE
MESTRE EM CIÊNCIA DA COMPUTAÇÃO

Programa: Ciência da Computação
Orientador: Prof. Dr. Marcel Parolin Jackowski

São Paulo, maio de 2019

Detecção de rasuras em símbolos com aprendizado Bayesiano de programas (BPL)

Esta versão da dissertação contém as correções e alterações sugeridas pela Comissão Julgadora durante a defesa da versão original do trabalho, realizada em 20/05/2019. Uma cópia da versão original está disponível no Instituto de Matemática e Estatística da Universidade de São Paulo.

Comissão Julgadora:

- Prof. Dr. Marcel Parolin Jackowski (orientador) - IME-USP
- Prof. Dr. Fábio A. M. Cappabianco - UNIFESP
- Prof. Dr. Helton Hideraldo Biscaro - EACH-USP

Agradecimentos

Agradeço primeiramente ao professor Marcel, que através de sua atenção e orientação em todos os nossos encontros permitiram que eu me desenvolvesse como pesquisador e pessoa diante essa difícil e gratificante fase de minha vida. Agradeço a todos os funcionários do IME que me proporcionaram uma bela infra estrutura de trabalho para que eu pudesse realizar essa pesquisa sem qualquer agravante que por ventura pudessem interromper essa minha realização, as novas amizades conquistadas no instituto durante meu período de formação na pós como mestrando, que me apoiaram e me ajudaram com alguns momentos de reflexão e discussão de gostos pessoais próximos, também ao meu irmão e cunhada que permitiram momentos de descontração e tiveram paciência com o meu período de pós graduando aqui na USP.

Um agradecimento especial aos meus pais que de certa forma me ajudaram financeiramente para com parte das despesas e tiveram paciência com esse tempo decorrido onde fiquei sem dar a devida atenção presencialmente, e que certamente sem esse apoio eu não teria o esforço e dedicação necessários para realizar este sonho.

Resumo

COSTA, R. **Detecção de rasuras em símbolos com aprendizado Bayesiano de programas (BPL)**. 2019. 78 f. Dissertação (Mestrado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2019.

Avanços significativos no reconhecimento de manuscritos rasurados e na recuperação de textos degradados tem sido obtidos através do uso de técnicas de aprendizado de máquina [RD16, KMRW14, CA17]. No entanto, o grande número de exemplos necessários na etapa de treinamento pode comprometer o uso prático de tais métodos. Este trabalho descreve um modelo de aprendizado para a classificação de dígitos rasurados denominado de “aprendizado por um tiro”, pois que permite uma caracterização mais próxima ao de um ser humano [Nei14]. O termo “aprendizado por um-tiro” especifica que o aprendizado de um novo conceito é obtido pelo reconhecimento dos principais traços característicos de um objeto, palavra ou símbolo dado um conhecimento a priori relativamente pequeno sobre um novo objeto ainda não identificado. Deste modo, este pré conceito sobre sua caracterização admite a construção de classificadores que realizem a predição a partir de uma imagem de teste como entrada com um conjunto reduzido de imagens de treinamento.

Este projeto cria um método que, embasado em [LST15], busca classificar imagens rasuradas de manuscritos a partir de um conjunto reduzido de imagens de treinamento sem rasura. Todo o pré-ajuste calculado através dessas imagens são separados por amostras chamadas de “programas”, ou seja, uma composição do conhecimento a priori de todos os traços e sub-traços existentes na imagem, componentes estas que atribuem variabilidades locais e globais que poderão ser reconhecidas estatisticamente por Inferência Bayesiana. Destacamos aqui que o modelo consegue predizer dígitos e símbolos independentemente das rasuras pré-dispostas no conjunto de imagens de testes, onde este conhecimento a priori é inexistente dentro do conjunto de imagens de treinamento.

Apresentamos em nossos resultados uma análise que, dado o ajuste inicial aplicado pelo aprendizado Bayesiano de Programas (BPL), nos permitiu utilizar características locais (traços) retiradas de cada símbolo para mensurar o custo referente entre todos os traços de uma imagem de teste e os traços em cada classe representado como imagem de treinamento. Para a obtenção do custo, utilizamos da métrica de deformação dinâmica aplicado em séries temporais (DTW) otimizadas por [SB16]. Finalmente, realizamos a predição para cada grupo de execuções considerando três conjuntos de dados: no primeiro com o grupo de imagens de teste rasuradas, no qual se obteve uma média global de acertos de 76%; no segundo conjunto com imagens de testes sem rasura (com diferença nos traços entre teste e treinamento), obtivemos uma média de 90% para com os acertos; e o terceiro conjunto usando o Omniglot. Tais reajustes adquiridos com poucos exemplos vislumbram futuras aplicações mais complexas em manuscritos degenerados ainda não digitalizados.

Palavras-chave: BPL, DTW, manuscritos, ciência cognitiva, rasuras, aprendizado de máquina.

Abstract

COSTA, R. **Detection of erasures in symbols with Bayesian Program Learning**. 2019. 78 f. Dissertation (Master Degree) - Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2019.

Significant advances in the recognition of erased handwritten texts and in the recovery of degraded texts have been obtained through the use of machine learning techniques [RD16, KMRW14, CA17]. However, the large number of examples required in the training stage can compromise the practical use of such methods. This paper describes a learning model for the classification of erased digits known as “one-shot learning”, which employs a characterization closest to that of a human being [Nei14]. The term “one-shot learning” specifies that the learning of a new concept is obtained by the perception of main characteristic traits of an object, word, or symbol given a prior relatively small knowledge about a new object not yet identified. Thus, it allows for the construction of classifiers that perform the prediction using a reduced set of training images.

This work describes a method that, based on [LST15], seeks to classify images containing handwritten digits with erasures given a set of training images without any erasures. Every calculated fitting using these images are called “programs”, i.e., a composition of prior features of all strokes and sub-strokes in the image that can be statistically recognized by Bayesian Inference. This model, known as Bayesian Program Learning (BPL), is then able to predict digits and symbols regardless of the erasures present in the set of test images, where this prior knowledge is non-existent within the set of training images. Given an applied fitting, the local features of strokes taken from each symbol were used to measure the relative cost between all strokes between a test image and the class represented by a training image. To obtain this cost, a Dynamic Time Warping (DTW) algorithm applied to 2D time series was employed [SB16].

This method achieved an average accuracy of 76% for recognizing erased digits, and 90% for recognizing digits with no erasures (in which the tests present a deformed shape compared to trained image). While the overall accuracy is still low compared to traditional deep learning techniques, this method requires fewer training datasets, and with proper characterization of stroke features it may aid in the recovery on degenerated handwritten text.

Keywords: BPL, DTW, handwritten, cognitive science, erasure, machine learning.

Sumário

Lista de Abreviaturas	ix
Lista de Símbolos	xi
Lista de Figuras	xiii
Lista de Tabelas	xvii
1 Introdução	1
1.1 Objetivos	3
2 Trabalhos Relacionados	5
3 Conceitos	9
3.1 Inferência Bayesiana	9
3.1.1 Probabilidades	10
3.1.2 Probabilidade Condicional	10
3.1.3 Probabilidade conjunta	11
3.1.4 Probabilidades Marginais	11
3.1.5 Teorema de Bayes	12
3.2 A Abordagem Cognitiva	13
3.2.1 Desenvolvimento Cognitivo através de Modelos Bayesianos	14
3.2.2 Relação probabilística do princípio de tamanho da amostra	16
3.2.3 Modelo Bayesiano Hierárquico em restrições indutivas	17
3.2.4 Qualidade de ajuste para as restrições indutivas	19
3.3 Representação de Caracteres com aprendizado Bayesiano de Programas	22
3.3.1 Relações entre os traços	22
3.4 BPL - Token de Caracteres	23
3.4.1 Mapeamento da trajetória	24
3.4.2 Parâmetros para a Imagem resultante	25
3.5 Caminhos aleatórios (em grafos)	25
3.6 O traço de um caractere	27
3.6.1 B-Splines	28
4 Metodologia	31
4.1 <i>Dataset</i>	32

4.2	Ajuste de programas Motores	33
4.2.1	Esqueleto da Imagem	34
4.2.2	Gerando amostras aleatórias	34
4.2.3	Busca por traços e sub-traços	35
4.2.4	Otimização e refinamento	37
4.2.5	Estimativa da Variância Local	37
4.3	Identificação de rasuras e reajuste das imagens	37
4.3.1	Extração das características locais com DTW	38
4.3.2	Junção das características locais e globais	39
5	Resultados	43
6	Conclusões	51
A	Código Fonte	53
A.1	Geração do contêiner das imagens (Ajuste dos programas motores)	54
A.2	Predição das imagens de teste sobre as classes de imagem de treinamento (reajuste) .	54
A.3	Comparação e soma das características de maior semelhança entre os programas motores de teste e treinamento	56
A.4	Função para o cálculo de pontuação preditiva	56
B	Algoritmo DTW (<i>Dynamic Time Warping</i>)	59
	Referências Bibliográficas	63

Lista de Abreviaturas

BPL	Aprendizado Bayesiano de Programas (<i>Bayesian Program Learning</i>)
DL	Aprendizado Profundo (<i>Deep Learning</i>)
DTW	Deformação dinâmica no tempo (<i>Dynamic Time Warping</i>)
GPU	Unidade de Processamento Gráfico (<i>Graphic Processing Unit</i>)
NN	Redes Neurais (<i>Neural Network</i>)
SVM	Máquina de Vetores de Suporte (<i>Support Vector Machine</i>)
OCR	Reconhecimento óptico de caracteres (<i>Optical Character Recognition</i>)
MCMC	Métodos de Monte Carlo via Cadeias de Markov (<i>Markov Chain Monte Carlo</i>)
MNIST	Dados Modificados do Instituto Nacional de Padrões e Tecnologia
SVHN	Conjunto de Imagens de números de casas (<i>Street View House Number</i>)
HBM	Modelo Bayesiano Hierárquico (<i>Hierarchical Bayesian Model</i>)
MDL	Descrição de comprimento mínimo (<i>Minimum description length</i>)
CAPTCHA	Teste de Turing público autorizado para diferenciação entre computadores e humanos

Lista de Símbolos

ξ	Conjunto de tipos de Relações.
ψ	conjunto de características de nível-tipo.
θ	conjunto de características de nível-token.
τ	coordenadas de uma Relação Longa ξ_i .
δ	variação entre as amostras de nível-token ou nível-tipo.
θ_A	ângulo entre dois vetores no cruzamento de um passeio aleatório.
λ	parâmetro para o controle de ângulos da distribuição de trajetória em passeios aleatórios.
γ	parâmetro de “força” para a escolha de novo vértice de borda em um passeio aleatório.
k	quantidade total de traços
S	variável descritiva dos traços
R	variável descritiva de relações entre os traços
z_{ij}	índice de um sub-traço
x_{ij}	vetor com as coordenadas de um sub-traço
y_{ij}	caracterização de escala de um sub-traço
$L_i^{(m)}$	posição inicial de um traço
$T_i^{(m)}$	sentido de trajetória do traço.
$A^{(m)}$	variável de transformada afim da imagem resultante.
σ	ruído para expansão e suavização dos pixels da imagem resultante.
ϵ	ruído que realiza a troca de pixels vizinhos de uma imagem.
$I^{(m)}$	Imagem resultante.
w_i	função objetiva que pontua cada programa motor (amostra) de uma imagem.
c	número total de classes dentro de um conjunto de dados de treinamento.
V	vértices de um grafo.
E	arestas de um grafo.
u	pixels que delimitam uma aresta do esqueleto (grafo) de uma imagem.
K	comprimento de um grafo.
$\vec{a}$	vetor anterior a um vértice no grafo.
$\vec{p}$	vetor posterior a um vértice no grafo.
T	vetor de nós de uma curva.
p	pontos de controle de uma curva.
d	grau do polinômio para se construir uma curva.
b	função de base para gerar uma curva.

Lista de Figuras

1.1	Como transformamos a percepção cognitiva de uma criança em modelos computacionais?	2
2.1	Estrutura de duas sub-redes para uma classificação binária com nó de predição logística p . (representação do artigo [KZS15]).	7
3.1	Público estimado do cinema.	10
3.2	(a) Exemplo de uma nova amostra (cinza) sendo identificada no espaço de hipóteses. Representação gráfica da amostragem dos dados para cada possível hipótese existente no meio considerando o cenário do cinema. (b) Algumas hipóteses geradas aleatoriamente dentro do espaço H . Isso demonstra como nosso cenário de problemas podem se ajustar a infinitas tendências dentro da visão Bayesiana.	15
3.3	Visão sistemática da relação probabilística entre o conjunto de dados e o universo de hipóteses. (a) Os retângulos são as representações hierárquicas do ponto (imagem do cão). Hipóteses de maior probabilidade tendem a ser mais escuras. (b) Com mais imagens de cães (pontos) apresentadas, hipóteses mais restritivas terão maior destaque.	16
3.4	Aprendizado super generalizado de informação. a) Não podemos identificar qual dos dois pontos (2 ou 3) estão mais próximos de 1. b) Com o crescimento da amostra em uma região, conseguimos mensurar a restrição de inferência formada pelo espalhamento desses dados (retângulos). Os retângulos são orientados ao longo do eixo y (maior largura (w) e menor comprimento (l)). Após avaliar essa super-generalização e as restrições dentro do grupo de hipóteses para cada exemplo de dados, podemos afirmar que os pontos 1 e 2 possuem características próximas.	18
3.5	Distribuição de bolinhas coloridas em um espaço amostral. Para as bolsas b_1 e b_2 foram retirados todas as bolinhas de dentro, realizando a super generalização dentro do espaço de hipóteses. Em b_n retiramos apenas uma esfera e aplicamos o aprendizado de alto nível a partir das experiências a priori. Θ é a distribuição de cores para cada bolsa e α e β são as probabilidades a priori decorrentes da frequência de cores uniformes (tendência) e totais de cores no geral. Qual será a frequência de cores (α) em b_n e quantas cores temos nesse espaço de amostras (β)?	18
3.6	Representação da qualidade de ajuste sobre um espaço de hipóteses. a) Hipótese únicas e simples sobre todo o conjunto de dados. Em c) os dados possuem um ajuste bem próximo, só que com a distribuição mais complexa; enquanto em b) temos uma distribuição mais equilibrada. Uma análise Bayesiana adequada sobre esses busca o equilíbrio entre a complexidade da teoria e o ajuste dos dados, como ocorre em b.	20

3.7	Representação do equilíbrio entre qualidade de ajuste com simplicidade entre os subgrupos de dados. Para o dataset a), temos uma visão mais simplificada pela escoasses de dados no meio; ao contrário de c), que possui um amontoado de dados maior em cada subgrupo considerado de b).	21
3.8	Exemplo de construção de um dígito rasurado no primeiro nível do BPL.	23
3.9	Exemplo de todas as possíveis relações entre os traços.	23
3.10	Exemplo: duas trajetórias a partir dos pontos iniciais de cada traço - $L_1^{(m)}$ e $L_2^{(m)}$.	24
3.11	Exemplo de criação de caracteres pelos dois níveis do BPL (ψ e θ): partindo das primitivas utilizadas por cada um até as suas respectivas imagens digitais. (representação do artigo [LST15]).	25
3.12	Um passeio aleatório em um grafo (em amarelo) sendo realizado a partir do vértice superior da cruz. Pontos em vermelho são os vértices do grafo, traço em verde são arestas do grafo. a), b) e c) representam três possíveis passeios a partir de um mesmo vértice.	26
3.13	Exemplo de uma b-spline de grau 2. Os pontos de controle são definidos em vermelho; os nós em laranja são os pontos existentes na curva C gerada por um polinômio.	28
3.14	Exemplos de b-splines criadas a partir de 5 pontos de controle (em vermelho). Note a característica de variabilidade de encolhimento da curva entre os pontos iniciais e finais do polinômio.	29
4.1	Fluxograma do modelo de classificação de aprendizado em uma execução.	31
4.2	Exemplos de algumas imagens de testes. (a) imagem com rasura simples. (b) inclinada. (c) em onda. (d) cruz. (e) múltipla. (f) zig-zag.	33
4.3	Exemplo de um esqueleto de imagem representado no Matlab.	34
4.4	Conjunto de amostras aleatórias geradas a partir de uma imagem de entrada. O retângulo vermelho representa as 2K melhores amostras em ordem decrescente (do melhor para o pior) pontuadas pelo processo de busca por traços e sub-traços.	36
4.5	Os 5 melhores programas motores de um dígito rasurado, pontuados através da equação E4 e selecionados dentro do grupo total de amostras aleatórias (figura 4.4).	36
4.6	5 programas motores de um dígito rasurado após o processo de otimização e refinamento.	37
4.7	Comparação entre traços de uma figura sem rasura com uma rasurada. Abaixo dos dois números são selecionados os traços que melhor se ajustem à imagem de treinamento. Os pontilhados representam o melhor caminho entre dois pontos das duas curvas, ou seja, o menor custo na diagonal da matriz (DTW_{final}).	39
4.8	Exemplo de imagem $I^{(T)}$ sendo reajustada por um novo dígito $I^{(c)}$.	40
5.1	Imagens utilizadas na acurácia da tabela 5.1. a) imagens de teste e b) imagens de treinamento.	43
5.2	Imagens utilizadas na acurácia da tabela 5.2. a) imagens de teste e b) imagens de treinamento.	44
5.3	Imagens utilizadas na acurácia da tabela 5.3. Novo conjunto de imagens para o treinamento para as imagens de teste da figura 5.2.	45

5.4	Algumas poucas imagens utilizadas na acurácia da tabela 5.4 (mais especificamente entre as execuções 1 e 5). a) imagens de teste e b) imagens de treinamento. As imagens em a) contemplam todos os tipos de rasuras utilizados no conjunto de dados completo.	46
5.5	Conjunto de predições separados por tipos de rasura.	47
5.6	Exemplo de computação do algoritmo DTW em um dos caracteres do Omniglot . . .	48
5.7	Exemplo de comparação entre amostras de dois dígitos de mesma classe. Note que na segunda coluna dos números 2 (a) e 9 (b) o modelo encontrou dificuldade na realização do reajuste.	48

Lista de Tabelas

5.1	Acurácia para identificação de dígitos com rasura . 10 classes de números variando de 0 a 9. Em cada execução, o valor 1 representa o acerto para o par de imagens e 0 o erro.	43
5.2	Acurácia para identificação de dígitos sem rasura . 10 classes de números variando de 0 a 9. Em cada execução, o valor 1 representa acertos e 0 os erros. A execução 5 possui as mesmas imagens em teste e treinamento como meio de calibração do modelo.	44
5.3	Acurácia para identificação de dígitos sem rasura . 10 classes de números variando de 0 a 9. Em cada execução, o valor 1 representa acertos e 0 os erros. Este conjunto de execuções foi realizado com o dataset de imagens sem rasura, mas com um novo conjunto de imagens de treinamento estabelecidos pela figura 5.3.	45
5.4	Acurácia para identificação de caracteres do Omniglot[LST15] com rasura	46

Capítulo 1

Introdução

Humanos possuem uma forte habilidade no reconhecimento de padrões. Em particular, quando nós somos apresentados a certos estímulos, temos a capacidade de entender rapidamente novos conceitos e logo reconhecer variações desses conceitos em percepções futuras. Esta abordagem se aplica mais amplamente a classes de objetos também não reconhecidas por um ser humano. Se mostramos uma imagem de uma espécie de animal específica a uma pessoa, ela não só pode usar esta imagem para identificar outros do mesmo tipo, mas como poderá fazer suposições a respeito das características gerais daquela imagem. Um dos desejos a respeito desta aplicação dentro de visão computacional e aprendizado de máquina é que, através de um conjunto de imagens com categorias distintas seja possível rotular a classe de um novo objeto desconhecido sem a necessidade de um número extensivo de exemplos.

Recentes avanços relacionados ao desenvolvimento de paradigmas em Inteligência Artificial permitem uma maior aproximação de modelos estatísticos ao reconhecimento de padrões e treinamento de dados em larga escala. O aprendizado de máquina foi usado com sucesso para alcançar o estado da arte em uma variedade de aplicações como: filtro de pesquisas na web, detecção de spam, geração de legendas, separação de textos e reconhecimento de fala e de imagem. Dentre as técnicas mais pesquisadas, o *Deep Learning* (DP) é um dos paradigmas mais famosos para a obtenção de características locais na classificação utilizando múltiplas camadas de convolução [LBH15]. Podemos considerar que o DP é uma evolução das redes neurais (NN), que utilizam de seus nós (neurônios) para classificar uma grande quantidade de dados e obter um resultado aceitável. Outros autores como [RD16] demonstram que a estrutura de aprendizado semi-supervisionado também visa a criação de algoritmos otimizados que consigam rotular dados de larga escala em pouco tempo (ou seja, um método de menor complexidade e mais eficientes) para a classificação. Neste trabalho, a principal motivação dos autores foi a construção de um método que otimize o tempo de escalonamento da informação em grafos que possuam um número elevado de nós e arestas, resultando em uma menor utilização de memória a cada iteração e identificação dos rótulos. Em [KMRW14] são criadas duas etapas do modelo: uma que captura as características mais latentes em um grupo de dados de imagem com poucos rótulos a serem classificados (neste caso usam o conjunto de dados de imagem modificados do Instituto Nacional de Padrões e Tecnologia (MNIST) e o conjunto de Imagens de números de casas (SVHN) e a outra que utiliza dos rótulos de saída definidos na primeira etapa para discriminar um modelo generativo semi-supervisionado que serão utilizados em um classificador com uma grande quantidade de amostras sem rótulos. A maior semelhança dentre esses métodos citados, é que para obtermos um resultado válido em sua classificação, necessitamos de uma quan-

tidade elevada de exemplos, e que ao contrário de métodos tradicionais, boa parte dos dados de treinamento carregados no classificador não necessitarão de rótulos pré-armazenados. O conceito de aprendizado semi-supervisionado trouxe uma melhoria significativa para os modelos tradicionais, pois proporciona uma generalização mais efetiva sobre os dados que convencionalmente possuam poucos rótulos (e são mais fáceis de se obter), o que é uma característica notável por exemplo na Inferência Bayesiana.

Apesar do avanço pertinente apresentado por técnicas de aprendizado semi-supervisionado, a demanda por dados necessários para que o classificador consiga atingir um grau de aceitação confiável é bem alto. Neste quesito, a ciência cognitiva busca justamente reduzir essa necessidade do armazenamento excessivo da informação, compensando por capturar uma gama de características vitais para a descrição do dado, o que reflete na maneira com que nós formulamos hipóteses por um ínfimo conjunto de informações. Um algoritmo típico de reconhecimento de caracteres manuscritos é o LeNet5 [LBBH98], que para o MNIST, são necessários e estão disponíveis 60 mil imagens para treinamento e 10 mil para a realização de testes. Repare que em tese, o dataset MNIST detém padrões de tracejados simplificados dentre os 10 números possíveis, mas computacionalmente a construção do conjunto de hipóteses necessários para a acurácia se torna inviável pelos métodos tradicionais considerando algumas poucas imagens.

Diante desses problemas comumente encontrados nesses modelos, tivemos um particular interesse em considerar metodologias que estão fortemente ligadas a classificação com poucos exemplos, também conhecida como classificação com um tiro. As Crianças apresentam uma aptidão nata em detectar simbologias e linguagens com o aprendizado por um tiro [STG04]. Isto se manifesta durante o desenvolvimento motor e de habilidades cognitivas na infância (figura 1.1). Por exemplo, as crianças assimilam linguagens em um curto espaço de tempo, onde que para cada palavra nova ela não necessite de uma reafirmação contínua dos pais. Além disso, elas podem generalizar a partir de construções linguísticas simples, desenvolvendo regras sofisticadas sobre novas categorias de palavras utilizando poucos exemplos [Fla99].

Figura 1.1: Como transformamos a percepção cognitiva de uma criança em modelos computacionais?

Para fins de otimização de processamento de dados em massa, e compartilhando de novos métodos que se ajustem a um resultado rápido de classificação, utilizamos como base deste trabalho o conceito de aprendizado por indução probabilístico Bayesiano. O *framework* sintetiza exatamente o que o aprendizado com um tiro apresenta: capaz de estimar muitas classes de conceitos visuais com apenas um exemplo e generalizando formas que são principalmente indistinguíveis das pessoas. O aprendizado Bayesiano de programas (BPL) considera três pontos chaves encontrados na ciência cognitiva: a composicionalidade do elemento, a causalidade e aprendizado por experiência.

Como resultado, ricos conceitos podem ser construídos “de forma composta” através de primitivas mais simples. Sua semântica probabilística lida com o ruído e generalizações das características de uma nova informação de forma processual que (ao contrário de outros modelos probabilísticos) naturalmente captura a estrutura abstrata de “causalidade” de processos do mundo real que produzem exemplos de uma categoria. O aprendizado procede na construção de programas que melhor explicam a observação dentro de um critério Bayesiano, e o modelo “aprende por experiência” desenvolvendo prioridades hierárquicas que mais se aproximam da estrutura de ajuste cognitivo que nós apresentamos inconscientemente. Essas prioridades representam um viés induzido que abstrai as principais regularidades e dimensões de variação em ambos os tipos de conceitos e em todas as instâncias (ou tokens) de um dado domínio. Em suma, o BPL pode construir novos programas reutilizando as partes existentes, capturando as propriedades causais e compostas dos processos generativos do mundo real, operando em múltipla escala.

Considerando a abordagem do aprendizado Bayesiano de programas (BPL) apresentado por [LST15], podemos identificar, em partes, todas as possíveis primitivas na observação de um manuscrito, criando programas motores que generalizam um conjunto de características necessários da imagem. Os programas resultantes cobrem os principais conceitos análogos ao processo de reconhecimento visual e de memorização de símbolos identificados em nós humanos. Não seria difícil também imaginar que pessoas possam identificar símbolos ou imagens variadas com um certo grau de deterioração ou rasura a partir do conhecimento prévio assimilado na infância. As primitivas que dizem respeito as principais formas dos objetos aprendidos ficam armazenados em nossa memória de longo prazo.

1.1 Objetivos

Este trabalho apresenta um método de reconhecimento de padrões que identifique rasuras pela abordagem teórica do aprendizado cognitivo traduzido para o sistema computacional através da generalização indutiva Bayesiana. Esta metodologia traz avanços comparáveis ao estado da arte no que diz respeito a publicações relevantes na área de *Deep Learning* para classificação de dados, pois ele considera um novo ramo estrutural base ainda não visto fora da academia; valorizando modelos estatísticos já conhecidos, mas pautando a curva de aprendizado mais próxima da realidade, que no caso desta linha de pesquisa é a aplicabilidade da ciência cognitiva em computadores. Além disso, podemos visualizar melhorias consideráveis à técnicas que já identificam padrões e criam estruturas similares utilizando, por exemplo, redes neurais siamesas ou até mesmo o próprio aprendizado Bayesiano de programas (BPL). Tendo em mente que para uma pessoa, dependendo do grau de complexidade da rasura, a identificação da classe da palavra ou símbolo se torna trivial diante do conhecimento prévio já identificado anteriormente; nosso método procura além de identificar as classes numéricas nos exemplos de testes, também separar quais traços desses programas não são pertencentes a estrutura original da imagem.

Na ciência forense por exemplo, a detecção de rasuras e a limpeza desses textos é algo crucial para o mapeamento da informação. Para isso, na literatura existem casos pesquisados como em [CA17] que identificam alguns padrões de rasuras aplicados no cotidiano. No modelo, um classificador de aprendizado supervisionado é utilizado como intermédio no reconhecimento das classes de imagens que possuem ou não a rasura. Em seguida é realizado um pós-processamento nas imagens

identificadas como rasuradas, rotulando o tipo da rasura em específico a serem eliminadas por um método de limpeza em [Li11]. Realizamos também uma analogia entre a estrutura do modelo de detecção de rasuras pelo BPL, com os resultados apresentados em [CA17].

De modo geral esta pesquisa descreve os seguintes tópicos: inicialmente serão apresentados os trabalhos relacionados mais importantes como comparativo ao nosso modelo de classificação, que utilizem de métodos semelhantes ou até mesmo de premissas teóricas que se aproximem a nossa linha de pesquisa (Capítulo 2). Nos Conceitos (Capítulo 3) são apresentadas as principais abordagens para a aplicabilidade da Inferência Bayesiana e da ciência cognitiva de forma a dissertar a respeito de como a obtenção de características prévias de um símbolo ou dígito podem ser lapidados em um modelo de classificação. Na Metodologia (Capítulo 4) é feita a apresentação técnica do classificador que utiliza de base o aprendizado Bayesiano de Programas (BPL). Nos Resultados (Capítulo 5) realizamos uma bateria de execuções avaliando a qualidade da acurácia obtida a partir de alguns conjuntos de dados com poucos exemplos. Nas Conclusões (Capítulo 6) fechamos a pesquisa debatendo sobre os resultados obtidos, determinando a aplicabilidade deste classificador e avaliando possíveis trabalhos futuros que poderão ser realizados com base ao que está sendo apresentado até então.

Capítulo 2

Trabalhos Relacionados

Em geral, trabalhos que considerem o aprendizado por um tiro com incrementos de técnicas para se detectar rasuras ou falhas em imagens ainda é um campo pouco explorado, e que recebeu uma atenção limitada pela comunidade de pesquisadores relacionado ao aprendizado de máquina. No entanto, existem algumas linhas-chaves que incorporam temas precedidos a esta pesquisa na área de detecção de rasuras e ciência cognitiva.

Executado em uma base de Reconhecimento óptico de caracteres (OCR), o modelo [CA17] busca distinguir textos que possuam ou não rasuras. Eles apresentam uma combinação de (a) classificação de padrões e (b) método de identificação baseado em grafo de caminhos. Em (a), as características são baseadas em duas classes (normal e rasura). Um classificador SVM é utilizado para detectar tamanhos e tipos moderados de componentes rasurados. Em (b), o esqueleto do componente de texto é considerado como um grafo, e um traço de rasura é identificado usando uma restrição de menor caminho do grafo. Algumas propriedades adicionais foram incluídas na identificação de rasuras mais complexas como: zig-zag e onda. Seu propósito geral foi apresentar uma abordagem de identificação de rasuras pré-determinadas junto a um algoritmo de pós processamento que remove os pixels associados a esse problema. Com 500 páginas de documentos manuscritos, o SVM obteve uma acurácia de 91,56% para Inglês (e 91,06% Bengali) em (a). Para (b), a acurácia de identificação de rasuras e exclusão foram: 89,65% (89,31%) e 91,16% (89,29%), respectivamente.

Nosso trabalho usa a base teórica e de aplicação apresentados por [LST15]. Um modelo computacional que captura as habilidades de aprendizado humano (ciência cognitiva) para uma extensa classe de conceitos visuais: imagens de manuscritos de um conjunto de alfabetos. O modelo representa conceitos do aprendizado por um tiro em ajustes que melhor descrevem exemplos observáveis sobre um critério Bayesiano. Com essa descrição abstrata de cada imagem, o *framework* consegue compor os ideais de composicionalidade, causalidade e aprendizado por experiência descritos pela ciência cognitiva na forma de um programa motor. O BPL define um modelo generativo que relaciona novos tipos de conceitos por combinação de partes e sub-partes em novas formas simbólicas. Sua pesquisa desenvolve dois módulos do BPL: (a) criação de outros K exemplos de imagens a partir de uma imagem de entrada e (b) classificação de imagens dado um conjunto de execuções ajustados no pré-processamento.

Em [NPH06], campo Aleatório de Markov (MRF) foi utilizado para sub-categorizar problemas em segmentação de imagem de documentos manuscritos. Neste processo, os autores empregaram vários métodos de otimizações como o ICM (*Iterated Conditional Modes*), HCF (*Highest Confidence First*) e programação dinâmica em 2D. Além da rotulagem de atributos de layout como corpo de

texto, margens, cabeçalho/rodapé, números de páginas, etc. Os autores tentaram marcar as zonas de deterioração e rasuras de textos existentes nos documentos. Para testar seu método, eles escolheram alguns manuscritos históricos do escritor francês Gustave Flaubert, comparando as faixas globais de rótulos mapeados da imagem que identificam um grupo de textos rasurados. A acurácia para cada rasura encontrada em um rótulo mapeado não é quantificado ou está presente de forma direta neste trabalho, tão pouco os autores demonstraram uma certa preocupação em identificar os tipos de rasuras presentes em cada palavra com um método de remoção assim como em [CA17].

Em [BNJ09], foi proposto uma abordagem para restauração de imagens de documentos degradados com o devido ao tempo, usando um modelo contextual probabilístico. Diferente de outras abordagens tradicionais apresentadas pelos autores, o seu modelo é independente de características globais do documento (tipo de fonte, estilo, script, etc). Assim como em [NPH06], utilizaram do MRF para mapear e restaurar em longos caminhos pela imagem: cortes, erosões, junções de letras e partes rasuradas. O modelo também pode integrar restauração de documentos e ampliação de resolução em um único *framework*, assim gerando imagens de alta qualidade a partir de imagens rasuradas. Os resultados experimentais mostram uma melhoria significativa em várias fontes coletadas (incluindo revistas e livros), e demonstra compreensivelmente a robustez e adaptatividade da abordagem.

Em [SFH17] os autores propõem uma estrutura em redes neurais que consiga classificar e gerar novos dígitos a partir do conjunto de dados MNIST, mesmo que estes passem por pequenas deformações em sua orientação na imagem. A arquitetura desta rede também chamada de *ConvNET* é definida metodologicamente em três fases: onde a primeira é um conjunto de convoluções simples (8 convoluções com kernel 9x9) da imagem de entrada, a segunda fase é constituída por grupos de camadas escondidas que trabalham em paralelo (também chamados de cápsulas). Tais cápsulas processam os pesos que resultam em vetores $(v_0, \dots, v_j)$ que especificam o fator característico de cada tipo de transformação que está sendo disposto pela imagem durante o treinamento (e.g escala, translação, rotação). Sendo mais específico, cada uma dessas cápsulas retornam vetores que definem a orientação do conjunto de imagens de acordo com sua matriz de transformações. Por fim a terceira camada da rede retorna os pesos diante todas as classes existentes do conjunto de dados (que no caso do MNIST são 10 classes de números entre 0 e 9). Em suas execuções obtiveram resultados promissores onde, de acordo com os autores, se comparam a redes convolucionais mais complexas que dependam de um maior número de camadas e computações para o mesmo feito: no *dataset* MNIST obtiveram o melhor resultado de 0.25% na margem de erro para as 10 mil imagens utilizadas na fase de testes, e como adicional, fizeram a execução dos testes também para uma versão modificada dos dados chamada MultiMNIST (dataset MNIST com sobreposição de até dois números por imagem tanto para a fase de testes quanto para o treinamento) onde obtiveram o melhor resultado de 5.2% na margem de erro também para 10 mil imagens modificadas na fase de testes. Tanto o conjunto MNIST quanto o MultiMNIST utilizaram de 60 mil imagens na fase do treinamento.

Em [FF⁺03], é apresentado um método que utiliza do conceito sobre aprendizado com um tiro (conceito este existente como base deste trabalho para a identificação de caracteres rasurados). Através de uma abordagem Bayesiana, extraem o conhecimento necessário de características obtidas de um conjunto de imagens de treinamento. O processo de aprendizado das imagens decorre de forma não-supervisionada, onde o framework registra das imagens de treinamento um número fixo de regiões (círculos) que definem as “aparências”, ou conjunto de pontos dentro do raio de cada

círculo. Para estimar a distribuição desse conjunto de características obtidos em cada imagem de treinamento, formularam este problema de aprendizado como uma maximização variável da expectativa Bayesiana (VBEM), aplicado a um modelo de mistura gaussiana multidimensional. Foram utilizados quatro classes de imagens nos ajustes: gatos, carros, faces de pessoas e aviões. Cada *dataset* foi dividido aleatoriamente em dois conjuntos disjuntos de mesmo tamanho. Os autores não deixam claro o número total de imagens de treinamento usados em seu framework, mas separaram 50 imagens para o conjunto de testes. Em cada categoria, eles variaram o conjunto de imagens de treinamento entre 1 e 6, repetindo os experimentos em cada imagem de testes por 10 vezes, trocando as imagens utilizadas no conjunto de treinamento a cada uma dessas interações (assim obtendo uma maior robustez na estimativa). Na seção de resultados encontram uma acurácia com taxas de erros entre 8% e 22% utilizando o algoritmo de aprendizado Bayesiano por um tiro (resultado este extraído de 5 execuções aleatórias considerando as 4 classes de imagens).

No artigo [KZS15], os autores exploram um método para aprendizado com redes neurais siamesas introduzidas em [BGL⁺94]: tendo em princípio uma única estrutura com duas sub-redes; cada uma com seu conjunto simétrico de camadas escondidas que medem o peso associativo de cada pixel atrelado às duas imagens de entrada (X_1 e X_2), o que remete a analogia de aprendizado 1 a 1 entre uma imagem de teste e uma de treinamento apresentado pelo modelo Bayesiano descrito nos trabalhos relacionados anteriores. Uma vez que a rede foi configurada, os autores garantem que para qualquer classe pré-identificada, serão extraídas características suficientes para inferir uma acurácia em cima de uma imagem de teste como entrada. Usando uma arquitetura de convolução, obtiveram fortes resultados que excedem alguns outros modelos de *deep learning* tradicionalmente reconhecidos ([B⁺09], [KSH12], [SZ14], [Sri13]). Observe na Figura 2.1 como é a estrutura base da rede apresentada neste trabalho: os dois dados de entrada passam pelo mesmo conjunto de camadas dentre as duas sub-redes. A camada de distância ($\{d_1 \dots d_{N_2}\}$) define o valor de similaridade entre os dois dados de entrada.

Figura 2.1: Estrutura de duas sub-redes para uma classificação binária com nó de predição logística p . (representação do artigo [KZS15]).

Os autores aproveitam do mesmo *dataset* criado por [LST15] (Omniglot), conservando os 40 alfabetos criados como imagens de treinamento e os 10 para avaliação (testes). O número de letras em cada alfabeto varia consideravelmente de 15 para 40 tipos diferentes (cada uma dessas letras

foi produzida por 20 pessoas diferentes). Para os experimentos na rede sem convolução, utilizaram 3 conjuntos de imagens para treinamento de tamanhos: 30k, 90k e 150k. Considerando os três conjuntos de camadas (2, 3 e 4) da rede apresentados, temos como acurácia média em imagens de teste¹ sem distorções: 69,86% (30k), 73,55% (90k), 75,26% (150k). E com distorções afim 8x: 72,86% (30k), 76,83% (90k), 79,87% (150k). Para os resultados experimentais da rede siamesa convolucional: com um total de 6 camadas de convoluções, apresentaram a seguinte acurácia em cada conjunto de testes sem distorções: 90,61% (30k), 91,54% (90k) e 91,63% (150k). E com distorções afim 8x: 91,90% (30k), 93,15% (90k) e 93,42% (150k).

Através deste conjunto de análises a cerca dos principais pontos expostos neste capítulo de trabalhos relacionados, podemos certamente destacar que nenhum destes contemplam de metodologias no qual nos apoiamos inicialmente e buscamos atingir como nosso objetivo: a classificação de um manuscrito de dígito ou símbolo independente da rasura existente - no qual a fase de reajuste o classificador carrega características primárias identificadas apenas em um conjunto pequeno de imagens sem rasura. Alguns desses artigos se aproximam mais de nossa proposta, tais como os que utilizam da estrutura conceitual de Inferência Bayesiana para a classificação 1 a 1 de imagens de testes com treinamento; enquanto outros possuem propostas mais simplificadas quanto ao número limitado de classes utilizadas pelo modelo e a quantidade elevada de amostras de treinamento necessárias para a obtenção de uma boa acurácia.

¹Para o conjunto de imagens de validação criaram mais 10.000 imagens de 10 alfabetos existentes no Omniglot. Foram reservados os 10 últimos alfabetos do Omniglot também com 10.000 para testes.

Capítulo 3

Conceitos

Nesta seção serão detalhados os conceitos básicos de probabilidade e Inferência Bayesiana; formulações estas importantes para as consequentes definições a respeito da conversão da estrutura do pensamento cognitivo em parâmetros reconhecíveis por um computador. Em seguida serão abordadas as técnicas de base utilizados nessa pesquisa para a construção de um caractere através do Aprendizado Bayesiano de Programas (BPL), tal arcabouço foi necessário para a extração de características de dígitos ou símbolos a partir de uma imagem digital (mais detalhes a respeito do pré-ajuste de uma imagem no Capítulo 4).

3.1 Inferência Bayesiana

Esta seção traz toda base teórica utilizada no mapeamento do cenário de aprendizado cognitivo transcritos diretamente pelo teorema Bayesiano aplicado à tarefa de classificação.

A inferência Bayesiana é reconhecida hoje no cenário de pesquisas e aplicações probabilísticas graças a Thomas Bayes, criador do teorema de Bayes. Como o aprendizado cognitivo tem como principal característica o conhecimento a priori de um certo símbolo ou objeto, o teorema de Bayes se encaixa como uma “luva” em termos estatísticos por compartilhar dessa definição. O teorema mostra como alterar as probabilidades a priori tendo em vista novas evidências para obter probabilidades a posteriori.

Quando aplicado, as probabilidades envolvidas no teorema de Bayes podem ter diferentes interpretações de probabilidade. Com a interpretação bayesiana de probabilidade, o teorema expressa como a probabilidade de um evento (ou o grau de crença da ocorrência desse evento) deve ser alterada após considerar evidências sobre a ocorrência deste evento.

Podemos utilizar um exemplo simples para a demonstração do teorema de Bayes: Imagine que você vá assistir a um filme no cinema e uma pessoa logo a frente derruba o ticket de entrada. Ao pegá-lo do chão você pretende entregar a essa pessoa, mas nota que ela tem cabelo comprido. Como você chamaria sua atenção? Dependendo de seu grupo social rotineiro, você teria algumas hipóteses a priori para prever se a pessoa que derrubou o ticket seria um homem ou uma mulher. Vamos supor que existem o total de 100 pessoas na fila de entrada para o cinema. 50 dessas pessoas são mulheres e as outras 50 homens. Dentre todas essas mulheres, 25 possuem cabelo curto e 25 cabelo comprido. Do lado dos homens, 48 possuem cabelo curto e 2 possuem cabelo comprido.

Para deixar mais claro como esses dados podem ser utilizados com inferência Bayesiana, precisamos separar alguns conceitos de probabilidade que montem todo o cenário apropriado do teorema.

Os quatro conceitos são: probabilidades, probabilidades condicionais, probabilidades conjuntas e probabilidades marginais.

Figura 3.1: *Público estimado do cinema.*

3.1.1 Probabilidades

A probabilidade de um evento X acontecer é o número de maneiras que este pode ocorrer dividida pelo número total da ocorrência de X . Em nosso exemplo sobre o cinema ficaria da seguinte forma:

$$P(X) = X / \text{total}(X)$$

$$\begin{aligned} P(\text{mulheres}) &= \text{mulheres} / \text{pessoas} \\ &= 50 / 100 = 0,5 \end{aligned}$$

$$\begin{aligned} P(\text{homens}) &= \text{homens} / \text{pessoas} \\ &= 50 / 100 = 0,5 \end{aligned}$$

3.1.2 Probabilidade Condicional

A probabilidade condicional nos dá as chances de um dado evento A ocorrer sabendo que um evento B já aconteceu. Podemos usar o exemplo da seguinte pergunta: “se eu sei que a pessoa é uma mulher, qual a probabilidade dela possuir cabelos longos?”. A condicional é calculada da mesma forma que as probabilidades diretas, mas elas focam apenas no subconjunto de toda a quantidade da população - aqueles que atendem a uma certa condição. Neste caso, $P(\text{cabelos longos} | \text{mulheres})$

(a condicional que alguém tenha cabelos longos, dado que ela seja uma mulher), é o número de mulheres com cabelos longos dividido pelo total de mulheres. E o mesmo vale para o homens.

Probabilidade da pessoa ter cabelos longos sendo uma mulher:

$$\begin{aligned} P(\text{cabelosLongos}|\text{mulher}) &= \text{mulheresCabelosLongos} / \text{mulheres} \\ &= 25/50 = 0,5 \end{aligned}$$

Probabilidade da pessoa ter cabelos longos sendo um homem:

$$\begin{aligned} P(\text{cabelosLongos}|\text{homem}) &= \text{homensCabelosLongos} / \text{homens} \\ &= 2/50 = 0,04 \end{aligned}$$

3.1.3 Probabilidade conjunta

Dado ao menos duas variáveis aleatórias X e Y definidas em um espaço de probabilidades, a probabilidade conjunta é a distribuição que definem as chances de ocorrer X e Y dentro da margem espacial do universo de hipóteses onde essas duas variáveis são especificadas. Em nosso exemplo, uma pergunta que utiliza dessa junção de probabilidades seria: “Qual a probabilidade de alguém ser uma mulher de cabelos curtos?”. Encontrar essa afirmativa nos leva a dois passos já especificados: primeiro precisamos definir a probabilidade de mulheres no todo $P(\text{mulheres})$, e logo precisamos identificar a probabilidade condicional na qual uma pessoa possui cabelos curtos, dado que ela seja uma mulher $P(\text{cabelosCurtos}|\text{mulher})$. Combinando as duas probabilidades por uma multiplicação, temos assim sua conjunção. Esta abordagem é análoga na identificação de homens com cabelos longos por exemplo. Observe abaixo como ficaria a aplicação de conjunção para homens e mulheres:

Probabilidade da pessoa ser uma mulher e também ter cabelos curtos:

$$\begin{aligned} P(\text{mulheres com cabelos curtos}) &= P(\text{mulher}) * P(\text{cabelosCurtos}|\text{mulher}) \\ &= 0,5 * 0,5 = 0,25 \end{aligned}$$

Probabilidade da pessoa ser um homem e também ter cabelos longos:

$$\begin{aligned} P(\text{homens com cabelos longos}) &= P(\text{homem}) * P(\text{cabelosLongos}|\text{homem}) \\ &= 0,5 * 0,04 = 0,02 \end{aligned}$$

Diferente da probabilidade condicional, a conjunção não apresenta problemas quanto a ordem de precedência na operação, ou seja, a multiplicação entre as probabilidades $P(A \text{ and } B)$ será análogo a $P(B \text{ and } A)$.

3.1.4 Probabilidades Marginais

A distribuição marginal dentro de um subconjunto de coleções de variáveis aleatórias, é a distribuição de probabilidade das variáveis contidas nesse subconjunto. Ela oferece as probabilidades

de vários valores das variáveis no subconjunto sem referenciar aos valores das outras variáveis. Em nosso exemplo podemos considerar a seguinte pergunta: “Qual a probabilidade de alguém ter cabelos longos?”. Para encontrarmos isso, utilizamos dessa distribuição somando as probabilidades conjuntas de uma mulher possuir cabelos longos ($P(\text{mulheres com cabelos longos})$) com homens que possuem cabelos longos ($P(\text{homens com cabelos longos})$).

$$\begin{aligned} P(\text{cabelos longos}) &= P(\text{mulheres com cabelos longos}) + P(\text{homens com cabelos longos}) \\ &= 0,25 + 0,02 = 0,27 \end{aligned}$$

3.1.5 Teorema de Bayes

Com o conjunto de probabilidades discriminadas de nosso exemplo, podemos inferir finalmente como se dá a construção do Teorema de Bayes. Um resultado a posteriori que poderia nos interessar seria a partir da seguinte pergunta: “Se sabemos que existe alguém de cabelos longos, qual é a probabilidade dessa pessoa ser um homem?”. Este questionamento remete a uma probabilidade condicional $P(\text{homem}|\text{cabelosLongos})$. Uma possível inversa dessa condição seria aparentemente: $P(\text{cabelosLongos}|\text{homem})$. Infelizmente uma probabilidade condicional não pode ser inversível, o que nos leva a utilizar dessa propriedade através da probabilidade conjunta:

$$\begin{aligned} P(\text{homem com cabelos longos}) &= P(\text{cabelosLongos}) * P(\text{homem}|\text{cabelosLongos}) \\ P(\text{cabelos longos e homem}) &= P(\text{homem}) * P(\text{cabelosLongos}|\text{homem}) \end{aligned}$$

Sabendo que: $P(\text{homem com cabelos longos}) = P(\text{cabelos longos e homem})$

$$P(\text{cabelosLongos}) * P(\text{homem}|\text{cabelosLongos}) = P(\text{homem}) * P(\text{cabelosLongos}|\text{homem})$$

Isolando a nossa condição desejada:

$$P(\text{homem}|\text{cabelosLongos}) = \frac{P(\text{homem}) * P(\text{cabelosLongos}|\text{homem})}{P(\text{cabelosLongos})}$$

Substituindo os termos de exemplo indicados na expressão por variáveis genéricas X e Y, nós temos o teorema de Bayes:

$$P(X|Y) = \frac{P(Y|X) * P(X)}{P(Y)}$$

De acordo com [TP15], em resumo, a distribuição a *posteriori* incorpora, por via do Teorema de Bayes, toda a informação disponível sobre o parâmetro: informação inicial + informação da experiência ou da amostra.

Voltando em nosso exemplo, podemos resolver o dilema a respeito da pessoa que deixou o ticket cair usando o teorema:

$$\begin{aligned}
P(\text{homem}|\text{cabelosLongos}) &= \frac{P(\text{cabelosLongos}|\text{homem}) * P(\text{homem})}{P(\text{cabelosLongos})} \\
&= \frac{P(\text{cabelosLongos}|\text{homem}) * P(\text{homem})}{P(\text{mulheres com cabelos Longos}) + P(\text{homens com cabelos Longos})} \\
&= \frac{0,04 * 0,5}{0,25 + 0,02} = 0,07
\end{aligned}$$

Concluimos em nosso exemplo que existe uma chance de 7% da pessoa que você chamar a atenção a respeito do ingresso ser um homem. O Teorema de Bayes capturou nossa intuição a respeito do universo de hipóteses apresentado no início do exemplo. Esse universo inicial é o nosso conjunto incorporado de um conhecimento pré-estabelecido, demonstrando que o número de mulheres no ambiente era muito maior que o número de homens com cabelos longos.

É claro que esta inferência pode ser alterada quando passamos a considerar um conjunto diferente de dados ou até mesmo uma outra visão de contexto, como por exemplo, chamando a atenção de uma pessoa com cabelos longos próxima ao banheiro masculino.

3.2 A Abordagem Cognitiva

Se considerarmos de forma exclusiva os sentidos e meios mais comuns que as pessoas utilizam para capturar e processar qualquer informação contida em um conjunto de afirmações ou objetos disponíveis no ambiente, (comparando com a abordagem de reconhecimento de padrões e aprendizado mais pesquisados) certamente os olhos e o cérebro humano seriam os principais mecanismos de entrada e processamento a serem convergidos de forma teórica e de aprendizado inato pré-estabelecido desde a nossa infância. Dentro da visão de vários pesquisadores psicólogos e até filósofos, tudo o que aprendemos varia muito do meio em que vivemos. Em [Nei14] debatem que toda a informação que sabemos sobre a realidade tem sido medida, não só pelos órgãos que representam os sentidos, mas também por sistemas complexos que interpretam e reintroduzem a informação sensorial. Esse estímulo de saída que produzimos também é chamado de “comportamento”. Para cada conhecimento adquirido do meio, armazenamos uma informação a priori desse meio, e reproduzimos através de uma informação sensorial ou comportamental. Basicamente, o termo “cognição” se refere a todos os processos pelos quais a informação sensorial de entrada é transformada, reduzida, elaborada, armazenada, recuperada e utilizada.

De acordo com [Nei14], toda a estrutura psicológica que nos permite avaliar sistemas e realizar classificações de forma cognitiva se compara aos programas de computador. Lembrando que um programa é uma série de instruções para lidar com símbolos/problemas. Podemos inferir resultados a priori como em nossa mente, e também vincular características similares da captura e processamento de informação assim como realizados por nossos sistema sensorial. Embora um programa de computador possui uma analogia muito fraca na forma com que nossa mente processa a informação e infere um possível resultado plausível, ele ainda consegue seguir boa parte do conceito básico psicológico de como nós tomamos certas decisões. Ao contrário de nós, os programas não carregam em sua bagagem alguns termos racionais ou até sentimentais existentes em nosso processamento neurológico. Podemos notar que várias decisões que tomamos a respeito de uma classificação ou identificação, é em grande parte embasada pelo conhecimento hipotético que temos daquele sistema

e, dependendo da estrutura a ser enfatizada, usamos parte de nosso emocional para descrever. Nesses casos, por mais evoluído o conjunto de programas de um modelo em inteligência artificial, ainda não se consegue atingir tal patamar que reflete ao conjunto de inferências e hipóteses que as pessoas podem tomar com um simples olhar.

A analogia do pensamento cognitivo com a programação gera uma vantagem interessante para o aspecto da inferência humana que será explorada de forma mais detalhada nas sub-sessões abaixo. Essa vantagem é a aplicação de aspectos técnicos existentes no campo da estatística que possam simular o espaço de hipóteses com escalas altíssimas que nosso cérebro pode identificar. Detalharemos então como a Inferência Bayesiana se encaixa nos moldes de inferência e na realização de teorias mais simples, como as identificadas no desenvolvimento cognitivo de crianças (até 2 anos de vida), refletidas em um modelo computacional passível de aplicação.

3.2.1 Desenvolvimento Cognitivo através de Modelos Bayesianos

Na sessão 3.1 apresentamos alguns conceitos básicos a respeito da Inferência Bayesiana, resultando no famoso Teorema de Bayes, assim como um exemplo que reflete sua aplicação em nosso cotidiano. Agora, precisamos entender como o modelo que visa a resolução em problemas Bayesianos pode ser aplicado dentro da estrutura biológica e do raciocínio humano, mais especificamente, como o aprendizado cognitivo de forte presença em nossa infância pode ser interpretado por esse método estatístico.

Umas das questões centrais do desenvolvimento cognitivo, é como nós podemos aprender com uma quantidade limitada de evidências. Ao aprender sobre relações causais, raciocínio sobre certas categorias de objetos ou suas propriedades, aprendizado de uma nova linguagem, ou a construção de teorias indutivas, as crianças fazem essa inferência de forma rotineira que vão além dos dados que observam. [PTGX11] considera três problemas gerais de indução que os “alunos” enfrentam, cada um baseados em desafios específicos de desenvolvimento:

1. Generalização indutiva de exemplos, com foco em aprendizado de rótulos para categoria de objetos.
2. Aquisição de restrições indutivas, refinando e modelando o conhecimento a priori de uma experiência, com foco em aprendendo a aprender categorias.
3. *Frameworks* de aprendizado indutivo, construindo ou selecionando espaços de hipóteses apropriadas para generalização indutiva, com aplicações em adquirir teorias intuitivas da mente e inferir estrutura hierárquica em uma linguagem.

Tais propriedades são perfeitamente representadas pela aplicação da Inferência Bayesiana, e para isso vamos tomar como base o exemplo aplicado na sessão 3.1. Pensando a longo prazo, podemos notar que a estrutura do público existente no cinema daquele exemplo pode mudar drasticamente. O espaço de hipóteses H desse cenário pode conter infinitas previsões possíveis para os dois públicos.

Para facilitar a compreensão do espaço de amostras que o modelo de Inferência pode enxergar, [PTGX11] avalia como um grupo de informações pode categoricamente se modificar dependendo de seu posicionamento no espaço amostral, independente do tipo de processo que gerou esse espaço de amostras.

Figura 3.2: (a) Exemplo de uma nova amostra (cinza) sendo identificada no espaço de hipóteses. Representação gráfica da amostragem dos dados para cada possível hipótese existente no meio considerando o cenário do cinema. (b) Algumas hipóteses geradas aleatoriamente dentro do espaço H . Isso demonstra como nosso cenário de problemas podem se ajustar a infinitas tendências dentro da visão Bayesiana.

Na figura 3.2 consideramos que os pontos¹ são gerados independentemente do evento realizado, e pela inferência é possível cobrir um novo ponto (cor mais clara) gerado em uma outra posição no espaço. Os retângulos são a representação de cada espaço de hipótese amostral sobre mulheres e homens de cabelos curtos ou longos do exemplo. No plano (a) os retângulos com um tamanho mais expressivo no comprimento (w) determinam as probabilidades a priori de pessoas do sexo masculino (sólido) e feminino (tracejado). Os retângulos com uma maior largura (l), determinam as pessoas que possuem cabelos curtos (pontilhado) ou compridos (tracejado maior). O ponto isolado no campo de hipóteses masculino, seria um possível homem que não possui características proeminentes dentro das duas outras hipóteses que pertençam a esse conjunto.

Nota-se que o espaço de hipóteses é definido de acordo com a natureza do problema de aprendizagem apresentado, e assim provê algumas informações a *priori* do cenário: os já mencionados comprimentos e largura de cada espaço amostral, bem como seu posicionamento em relação aos eixos x e y do plano (considerando o ponto inferior a esquerda de cada retângulo). Essas variáveis também são conhecidas como sendo os pontos de restrição de nosso problema. Sabendo que o espaço de hipóteses possui uma grandeza elevada de representações, essas restrições podem limiar dentre o valor 0 a um número extremamente alto como 10^9 (também contextualizado em uma distribuição de probabilidade com uma longa calda).

O Modelo Bayesiano possui a vantagem de poder ser definido em qualquer *framework* generativo bem especificado. A inferência se ajusta ao espaço generativo de qualquer classe de problema expresso no plano de H [PTGX11]. A representação da flexibilidade do modelo Bayesiano nos permite avançar em algumas dicotomias tradicionais ligadas ao desenvolvimento cognitivo: o conhecimento estruturado em contrapartida do aprendizado probabilístico, ou o conhecimento estruturado inato versus o aprendizado de conhecimento não estruturado. Existem várias pesquisas recentes citadas por [PTGX11] que relacionam exemplos de modelos Bayesianos que abraçam a representação de linguagem ou de composição em domínios que variam de indução causal a aprendizagem gramatical e a aquisição da teoria.

¹Pense no cenário do exemplo do cinema, onde todos os pontos são divisões a respeito de homens com cabelos curtos ou longos, e de mulheres com cabelos curtos ou longos. Isso se aplica a qualquer tipo de categoria a se aprender: palavras, eventos, classe de objetos, etc.

3.2.2 Relação probabilística do princípio de tamanho da amostra

Para ilustrar de forma mais concreta como uma análise básica Bayesiana de generalização indutiva é aplicada ao desenvolvimento cognitivo, vamos imaginar um caso de estudo clássico na nomeação de objetos ou imagens apresentados a uma criança. Uma palavra simples como “cão” pode se referir a um número infinito de hipóteses existentes dentro de sua sub-categoria (raças) ou super-categorias (animais, mamíferos, quadrúpedes, etc).

Com poucas evidências de imagens de cães apresentados a uma criança, é possível mapear um conjunto geral de características para outras espécies de cães, mesmo que elas não tenham ainda sido apresentadas. Sugere-se em [PTGX11] que recém nascidos são equipados com um forte conhecimento a priori sobre quais tipos de significados de palavras são naturais, com um certo nível de restrições considerados em todas as hipóteses existentes sobre tal palavra. Uma criança criará uma classificação simplista na taxonomia do cão apresentado, não relacionando com a raça do animal por exemplo. Intuitivamente, quanto menor o número de exemplos apresentados para um vínculo de aprendizado, mais níveis hierárquicos de hipóteses estarão alinhados a inferência (relação probabilística fraca). Pra um número maior de dados apresentados, hipóteses mais restritivas serão consideradas.

Na figura 3.3 retratamos essa abstração de cães dentro de uma hierarquia de animais. O retângulo mais externo cobre todas as hipóteses de animais existentes no conjunto. Com duas amostras de cães apresentadas em (a), podemos notar uma ressonância próxima a hipótese esperada daquela classe. Os retângulos menores correspondem as hipóteses subordinadas as maiores.

Figura 3.3: Visão sistemática da relação probabilística entre o conjunto de dados e o universo de hipóteses. (a) Os retângulos são as representações hierárquicas do ponto (imagem do cão). Hipóteses de maior probabilidade tendem a ser mais escuras. (b) Com mais imagens de cães (pontos) apresentadas, hipóteses mais restritivas terão maior destaque.

Podemos usar este exemplo para entender como a ressonância do espaço Bayesiano captura a noção de coincidência suspeita dentre os grupos hierárquicos de uma inferência. Esta habilidade de aproximação do cenário de hipóteses está diretamente vinculado a relação probabilística $p(d|h)$, a probabilidade do dado observado d assumindo que a hipótese h é verdadeira. Em geral, mais hipóteses restritivas correspondem a menores regiões no espaço de dados, recebendo mais probabilidades por uma certa parte do dado. Note que a relação a posteriori da inferência exige um equilíbrio entre quantos dados devem ser representados no meio para que o conjunto necessário de hipóteses ajustados sejam suficientes na inferência desejada. A quantidade de exemplos de dados mostrados a um “aluno” não refletem necessariamente na qualidade da inferência, e sim ao nível de abstração necessário de acordo com o cenário do problema.

Mais formalmente, se assumirmos que os dados são amostrados uniformemente ao acaso em todas as formas de distribuição possível dentro do espaço, então a probabilidade para cada dado

único d ligado ao conjunto h será inversamente proporcional ao tamanho de regiões de h detectados - também chamado de “tamanho de h ” [PTGX11]. A ressonância por hipóteses mais consistentes dentro de um conjunto longo de hipóteses é exponencialmente reajustável com o acréscimo de dados dentro daquele meio, e geralmente as hipóteses mais restritivas são fortemente favoráveis na maioria dos cenários probabilísticos. Essa definição de equilíbrio é muitas vezes chamado de *princípio de tamanho*.

O princípio de tamanho captura a noção de uma coincidência suspeita: ao aumentarmos o número de dados no meio, as hipóteses selecionadas podem realizar previsões mais específicas - com um maior poder de explicação - comparadas ao menor conjunto de dados, com previsões mais sutis. Isto provê uma solução natural para o problema de “evidência não negativa”². À medida que o tamanho do conjunto de dados tende ao infinito, o “aluno” (classificador) com visão Bayesiana rejeita hipóteses maiores ou mais gerais em favor das mais precisas. A probabilidade no aprendizado Bayesiano pode ser visto como um princípio de medida quantitativa dos pesos da evidência explícita negativa - o que explica como e quando uma sobre-generalização deve ocorrer.

3.2.3 Modelo Bayesiano Hierárquico em restrições indutivas

Uma das implicações de problemas clássicos na indução é a necessidade de uma generalização sobre generalização, ou restrições indutivas. O núcleo do problema é definir como um conjunto de dados finito pode ser caracterizado dentro do espaço, e dentro dessa premissa vamos instigar como essa restrição permite com que o aprendizado ocorra. Como já mencionado nas seções anteriores, as crianças possuem uma evidência indutiva com um alto nível de processamento a priori de uma nova informação. [PTGX11] Dizem que as crianças preferem vincular palavras a objetos por inteiro do que suas partes separadas. O mesmo vale para a identificação de palavras compostas ou sentenças gramaticais. Dado que este comportamento é ligado pela visão de separação categórica de formas de objeto desde o nascimento, é sensato considerar que essas restrições no meio de hipóteses estão mais ligada ao raciocínio inato do que ao aprendizado.

A super-generalização é ligada ao processo contínuo de padrões identificados dentro de um problema, delimitados na estrutura de uma hipótese disposta no meio. Um possível exemplo a se ilustrar é na identificação de bolinhas coloridas dentro de um pote [KPT07]: considere que no primeiro pote, a pessoa retire todas as bolinhas, sendo que todas elas são de cor preta. No segundo pote ele repara que todas as bolinhas retiradas são brancas. No terceiro, ao começar a retirar as bolinhas vermelhas, ele pode inferir que todas as bolinhas desse pote serão da mesma cor antes mesmo de verificar todas. Essa previsão pode ser considerada como uma indução por experiência, e com isso sua cognição restringiu sua previsão. Nós podemos ilustrar de forma abstrata como a indução restritiva de um conjunto de dados se encaixa com uma super-generalização (veja a Figura 3.4).

Considerando a abordagem cognitiva sobre esta abstração, o “aluno” (classificador) poderá dividir a estrutura de aprendizado em duas etapas: o aprendizado em primeiro ordem que analisa diretamente o formato de distribuição dos dados, quanto as características mais comuns dos retângulos representados na figura; e o aprendizado em segunda ordem (sobre-hipóteses) que se referem aos parâmetros que definem o comprimento, largura e sua orientação ao longo dos eixos x , y do

²Em [Mar93] o problema de evidência não negativa se refere ao aprendizado de uma linguagem/símbolo sem uma evidência negativa, ou seja, aprender a respeito de palavras que formam uma sentença gramatical.

Figura 3.4: *Aprendizado super generalizado de informação. a) Não podemos identificar qual dos dois pontos (2 ou 3) estão mais próximos de 1. b) Com o crescimento da amostra em uma região, conseguimos mensurar a restrição de inferência formada pelo espalhamento desses dados (retângulos). Os retângulos são orientados ao longo do eixo y (maior largura (w) e menor comprimento (l)). Após avaliar essa super-generalização e as restrições dentro do grupo de hipóteses para cada exemplo de dados, podemos afirmar que os pontos 1 e 2 possuem características próximas.*

plano. Esse tipo de interpretação nos ajuda a aprender melhor sobre teorias que ainda não nos foram apresentados, e ao final chegar a uma conclusão a respeito do comportamento homogêneo das hipóteses existentes em cada conjunto de dados de b): onde o ponto 1 está mais próximo do ponto 2, mesmo o ponto 2 e 3 sendo equidistantes de 1.

Um modelo Bayesiano que ajuda a abstrair um universo de hipóteses em problemáticas menores é o modelo Bayesiano Hierárquico (HBM) [KPT07]. O HBM constrói uma generalização de alto nível sobre essas hipóteses. Diferente de um modelo não hierárquico, neste o modelador especifica hiper parâmetros que permitem o modelo aprender o intervalo de parâmetros por si mesmo - a abordagem hierárquica mostra como o conhecimento pode ser simultaneamente adquirido em múltiplos níveis de abstração (imagine que em cada retângulo da figura 3.4 existem infinitas possibilidades de comprimento e largura para os variados conjuntos amostrais). Logo, o modelo hierárquico pode ao invés disso aprender o intervalo típico entre o comprimento e largura de cada hipótese (ou seja, que o comprimento tende a ser maior que a largura, ou vice versa).

Figura 3.5: *Distribuição de bolinhas coloridas em um espaço amostral. Para as bolsas b_1 e b_2 foram retirados todas as bolinhas de dentro, realizando a super generalização dentro do espaço de hipóteses. Em b_n retiramos apenas uma esfera e aplicamos o aprendizado de alto nível a partir das experiências a priori. Θ é a distribuição de cores para cada bolsa e α e β são as probabilidades a priori decorrentes da frequência de cores uniformes (tendência) e totais de cores no geral. Qual será a frequência de cores (α) em b_n e quantas cores temos nesse espaço de amostras (β)?*

Alguns detalhes matemáticos mais complexos ligados a este modelo são apresentados por [KPT07], assim nós mostraremos o conceito básico envolvendo o HBM com o exemplo anterior sobre potes

de bolinhas coloridas - Figura 3.5. Imaginemos que em cada pote b temos a distribuição de cores: chamamos essa distribuição de Θ^b . Ao mesmo tempo, para cada restrição de hipóteses representado pelos potes, temos a frequência de cores uniformes de cada pote (chamado de α) e os tipos de cores que existem no experimento no geral (chamado de β). Aqui, α e β são os hiper parâmetros para cada valor de Θ^b , já que a probabilidade de qualquer pote em particular dependerá das suposições de alto nível dos potes: se por instância assimilarmos que dentro de alguns potes terão bolinhas uniformemente coloridas, então as chances de β possuir diferentes cores resulta em baixa probabilidade comparados a α . Este exemplo é um ponto factual que podemos propagar em problemas de maior nível de abstração, como o utilizado neste trabalho para a separação dos traços de um símbolo manuscrito. O equilíbrio entre as probabilidades a priori em α e β determinam o grau de confiança de uma bolinha branca ser retirada de um possível pote b_n , ou seja, a probabilidade a posteriori $P(\alpha, \beta|y)$ tal que y é a nova amostra retirada de b_n .

O HBM demonstra ser uma valiosa ferramenta para explorar níveis de maior abstração na classificação de um objeto.

3.2.4 Qualidade de ajuste para as restrições indutivas

Vimos que o modelo Hierárquico Bayesiano explica as origens do viés indutivo e as restrições pela abordagem a priori em resposta à experiência do dado observado pelo modelador, usando múltiplas configurações ou contextos. Mas a aquisição de conhecimento abstrato frequentemente demonstram ser mais discretos ou qualitativos - mais do que construir um espaço de hipóteses apropriado, ou selecionar um espaço em que os dados são muito bem generalizados. Em [PTGX11] é dito que precisamos considerar a visão “teórica” do desenvolvimento cognitivo para o cenário. O conhecimento das crianças sobre o mundo é organizado por induções teóricas com uma estrutura e funções análogas as teorias científicas. A teoria serve como um *framework* abstrato que guia a generalização indutiva para os mais concretos níveis de conhecimento, gerando um espaço de hipóteses. Teorias intuitivas foram postuladas para fundamentar a categorização em um grupo de hipóteses mais se represente fatores existentes do mundo real (indução causal, reincidência biológica, reincidências físicas e interação social).

Se torna natural a mudança constante de teorias dentro de um modelo Bayesiano, para que esse ajuste se encaixe melhor a ideia do modelador e do problema apresentado. Em meio a esta visão, o desenvolvimento cognitivo reconhece que uma atual teoria de um domínio pode ser inadequada, e sua revisão constante em favor a uma nova teoria pode realizar distinções qualitativas conceituais que não foram apresentados por uma teoria anterior. Isso gera um desafio constante: o compartilhamento de parcimônia, ou simplicidade de uma teoria com o quão bom poderá ser o seu ajuste em um conjunto de dados. Devemos entender como criar um equilíbrio no grau de qualidade de ajuste dentro de um conjunto de dados representado no meio, para isso podemos usar como base a visão do modelo de aprendizado computacional apresentado em redes neurais ou redes profundas: um polinômio de grande ordem vai realizar ajustes mais precisos sobre os dados apesar de ser mais complexo computacionalmente [AMMIL12]. Então, quanto mais expressiva e detalhada for a teoria ou gramática, dado o seu grau de liberdade para com a informação, melhor será o ajuste do resultado aparente em relação a esta teoria. Mas também teremos o preço de balancear o quão detalhado queremos que nosso universo de dados sejam ajustados com essa visão teórica sobre as hipóteses - resultando em um maior número de variáveis a serem consideradas.

A qualidade de ajuste sobre uma hipótese h é capturada pela probabilidade a posteriori da Regra de Bayes, ou $P(d|h)$, enquanto a probabilidade a priori $P(h)$ reflete outras fontes de graus de aprendizado. A priori pode deter vários formatos dentro do espaço, mas em geral, uma preferência por hipóteses mais simples ou parcimoniosas vão emergir naturalmente sem ter sido projetado propositalmente. Esta preferência deriva das premissas generativas que rodeiam um *framework* Bayesiano, no qual as hipóteses são por si mesmas geradas em um processo estocástico que produz um espaço de hipóteses candidatas e $P(h)$ reflete a probabilidade da criação de h em torno deste processo. Consideremos as três hipóteses representada na Figura 3.6:

Figura 3.6: Representação da qualidade de ajuste sobre um espaço de hipóteses. a) Hipótese únicas e simples sobre todo o conjunto de dados. Em c) os dados possuem um ajuste bem próximo, só que com a distribuição mais complexa; enquanto em b) temos uma distribuição mais equilibrada. Uma análise Bayesiana adequada sobre esses busca o equilíbrio entre a complexidade da teoria e o ajuste dos dados, como ocorre em b.

Comparado aos exemplos anteriores, nós temos um conjunto de hipóteses mais expressivos, separando o grau de complexidade da divisão dos dados. Vamos destacar aqui novamente que: existem comprimento (l), largura (w), coordenadas x e y para cada retângulo representado na figura. No exemplo de uma hipótese mais simples, temos a demonstração de um cenário teórico com ajuste de cálculo simples, pois identificamos apenas uma vez os parâmetros l , w , x e y . Mas para os demais exemplos, aumentamos sua complexidade com o crescimento do ajuste, resultando em 60 variáveis com valores distintos para o conjunto em c). Em [PTGX11] destacam a estranheza da intuição como uma das características do pensamento cognitivo: intuitivamente, quanto mais complicado for a definição de uma teoria, menores serão as chances da construção válida de uma informação a posteriori; quanto maior as escolhas necessárias dentro de um espaço amostral, maiores são as chances de que a teoria inferida em princípio não esteja se restringindo adequadamente aos dados, o que resulta em uma informação a posteriori incorreta. Mais formalmente especificado: sabendo que a probabilidade a priori de uma hipótese é o produto das probabilidades de todas as escolhas necessárias para gerá-la - e a probabilidade de realizar umas dessas escolhas em particular deve ser baixa, então uma hipótese especificada com mais escolhas (variáveis) será melhor ajustável à informações a priori menos restritivas.

Esta mesma lógica geral sobre uma subdivisão de retângulos pode se aplicar a espaços de hipóteses mais interessantes no campo do aprendizado cognitivo: por instância, gramáticas mais complexas incorporam mais regras e não-terminais (e portanto mais escolhas se envolvem em especificar cada uma delas além de valores relacionados a cada coordenada no espaço), e mais complexas teorias causais envolvem um maior grau de especificação sobre a forma em que esses relacionamentos causais se comportam. Essa escolha de alto nível entre as distribuições da teoria no campo de hipóteses pode incorporar versões mais discretas e qualitativas do HBM apresentados na subseção 3.2.3.

Em [PTR06] demonstram como a hierarquia em uma construção gramatical ajudam crianças em fase de desenvolvimento a aprender poucas sentenças em poucas horas de forma parcimoniosa, o que corrobora com um detalhe importante para a **Navalha de Ockham** na qualidade de ajuste em restrições indutivas: para gramáticas menores, existirão poucos não terminais e menos regras - ou seja, poucos pontos de escolha. A partir destas condições, temos uma chave poderosa para a escolha de teorias ou conjunto hierárquico mais eficazes de acordo com o crescimento dos dados observáveis. No aprendizado de uma linguagem, uma solução tradicional para o problema de restrições de generalização na identificação de evidência negativa é o princípio de Subconjuntos [MCM13]: deve-se escolher a gramática mais específica e consistente com os dados observáveis. Na figura 3.6 a preferência pelo conjunto de hipóteses B ao invés de A se torna mais compreensível quando levamos em conta a premissa do princípio de subconjuntos, mas para a escolha de B ao invés de C devemos levar em conta o que visualizamos a respeito da complexidade das variáveis a priori detectadas para C.

Veja pela figura 3.7 como se comporta a versão Bayesiana da Navalha de Ockham: note que o compartilhamento entre a simplicidade e a qualidade de ajuste são modulados naturalmente de acordo com o amontoado de dados em cada conjunto de hipóteses.

Figura 3.7: Representação do equilíbrio entre qualidade de ajuste com simplicidade entre os subgrupos de dados. Para o dataset a), temos uma visão mais simplificada pela escoasses de dados no meio; ao contrário de c), que possui um amontoado de dados maior em cada subgrupo considerado de b).

Quanto mais se aumenta o número de dados observáveis no meio, maior é sua complexidade quanto ao conjunto de hipóteses a se escolher. Isto não vai ser verdadeiro para todos os casos, mas é um modelo a se considerar à medida em que os dados crescem. Mas ainda assim, a preferência em moldar o seu conjunto de hipóteses respeitando os fatores de qualidade e simplicidade, faz com que o modelo fique mais próximo a um processo de generalização conhecido como o princípio de Descrição de comprimento mínimo (MDL) [Ris78]. Em outras palavras, se o dado é verdadeiramente gerado por um processo correspondente ao número de hipóteses representados em cada exemplo da figura acima, então existirá uma regularidade em termos de distância para o aglomerado de dados alocados em cada uma dessas hipóteses, resultando em uma maior probabilidade a posteriori nos grupos a, b ou c.

Para as próximas Seções iremos apresenta um modelo que consegue generalizar o desenvolvimento cognitivo e modelos Bayesianos de forma a extrair a melhor caracterização possível de um caractere, que neste caso parte do princípio de que todo caractere, levando em conta a abordagem cognitiva, é gerado a partir de primitivas expressas como traços.

3.3 Representação de Caracteres com aprendizado Bayesiano de Programas

O aprendizado Bayesiano de programas (BPL) trabalha o nível de abstração de características de um caractere desde a fase de obtenção da teoria associada ao “traçar” todas as formas deste símbolo até a fase de combinação de todas as sub-partes resultando em uma imagem digital. A estrutura é dividida em dois níveis: Tipos de Caracteres e Tokens de caracteres.

Para a geração de uma nova amostra de um símbolo ou palavra, segue a seguinte definição da estrutura representada por [LST15]: um tipo de caractere $\psi = \{k, S, R\}$ é uma tupla de k traços $S = \{S_1, \dots, S_k\}$ e relações espaciais $R = \{R_1, \dots, R_k\}$ entre os traços. Esta distribuição pode ser representada da seguinte forma:

$$P(\psi) = P(k) \prod_{i=1}^k P(S_i) P(R_i | S_1, \dots, S_{i-1}) \quad (\text{E1})$$

O pressionar e soltar de uma caneta sobre o papel define um traço. Para cada traço, existem sub-componentes que se chamam sub-traços ou primitivas $S_i = \{s_{i1}, \dots, s_{in_i}\}$. Cada uma delas seriam possíveis pausas entre os traços feitos pela caneta. Limitamos o número de sub-traços de acordo com uma frequência empírica $P(n_i | k)$ ao número total de traços k .

Cada sub-traço é criado como uma *b-spline* cúbica uniforme que pode ser decomposta em três variáveis $s_{ij} = \{z_{ij}, x_{ij}, y_{ij}\}$ com distribuição conjunta $P(S_i) = P(z_i) \prod_{j=1}^{n_i} P(x_{ij} | z_{ij}) P(y_{ij} | z_{ij})$.

A classe discreta $z_{ij} \in \mathbb{N}$ é o índice relacionado as primitivas que mapeiam cada sub-traço, onde a distribuição $P(z_i) = P(z_{i1}) \prod_{j=2}^{n_i} P(z_{ij} | z_{i(j-1)})$ é uma cadeia de Markov de primeira ordem que geram bigramas com ângulos retos, estruturas repetitivas e características comuns.

A variável $x_{ij} \in \mathbb{R}^{10}$ é representada por 5 pontos de controle das primitivas pela imagem 3.11. Eles são amostras criadas por uma Gaussiana $P(x_{ij} | z_{ij}) = N(\mu_{z_{ij}}, \Sigma_{z_{ij}})$, existentes em um espaço abstrato ainda não definidos em um frame da imagem.

A variável y_{ij} define a característica de escala nível-tipo relativo ao frame da imagem como uma amostra de $P(y_{ij} | z_{ij}) = \text{Gamma}(\alpha_{z_{ij}}, \beta_{z_{ij}})$; uma vez que essa distribuição mapeia a pontuação de acordo com o ângulo de uma primitiva rotulada por z_{ij} . As primitivas (sub-traços) são sub-partes pré-determinadas de um conjunto total de partes finais (traços). Na figura 3.8 é apresentado o exemplo da construção de uma tupla para o primeiro nível da estrutura geral do BPL.

3.3.1 Relações entre os traços

De acordo com [LST15], a relação espacial R_i define a conexão de um traço a outro $\{S_1, \dots, S_{i-1}\}$ ao longo de uma imagem. As relações possuem uma estrutura que se divide em quatro tipos: $\xi \in \{\text{Independente}, \text{Inicio}, \text{Fim}, \text{Longa}\}$, com probabilidades θ_R . Cada ξ_i possui diferentes sub-variáveis e dimensionalidades registradas de acordo com o tipo de símbolo a ser mapeado (vide Figura 3.11). As especificações básicas das relações são apresentadas a seguir:

- Relação Independente $R_i = \{\xi_i = \text{Independente}, J_i, L_i\}$: onde a posição de um traço S_i não depende de outro traço definido anteriormente. A variável $J_i \in \mathbb{N}$ mapeia a primeira estrutura de z_{ij} , ou seja, uma relação independente está ligada ao fator do nível tipo ψ indexada em

Figura 3.8: Exemplo de construção de um dígito rasurado no primeiro nível do BPL.

cada primitiva z_{ij} com probabilidade $P(J_{z_{ij}})$. A variável $L_i \in \mathbb{R}^2$ define a posição inicial (eixo x,y) de um fator independente J_i .

- Relação Inicial, $R_i = \{\xi_i = \text{Inicio}, u_i\}$: onde o traço S_i começa no início do traço anterior $u_i \in \{S_1, \dots, S_{i-1}\}$.
- Relação Final, $R_i = \{\xi_i = \text{Final}, u_i\}$: onde o traço S_i começa no fim do traço anterior $u_i \in \{S_1, \dots, S_{i-1}\}$.
- Relação Longa, $R_i = \{\xi_i = \text{Longa}, u_i, v_i, \tau_i\}$: onde o traço S_i começa ao longo do traço anterior $u_i \in \{S_1, \dots, S_{i-1}\}$ se tornando um sub-traço $v_i \in \{s_1, \dots, s_{u_i}\}$ com coordenadas de curva do tipo de nível $\tau_i \in \mathbb{R}$.

Na figura 3.9 apresentamos uma ilustração de todas as relações possíveis entre os traços de símbolo criado na primeira e segunda fase do BPL.

Figura 3.9: Exemplo de todas as possíveis relações entre os traços.

3.4 BPL - Token de Caracteres

Um caractere no nível de token é a finalização ou junção das curvas de tipo, que geram a distribuição de uma nova representação simbólica da imagem ajustada. As variáveis de token são definidas como $\theta^{(m)} = \{L^{(m)}, x^{(m)}, y^{(m)}, R^{(m)}, A^{(m)}, \sigma_b^{(m)}, \epsilon^{(m)}\}$, com distribuição probabilística definida por [LST15]:

$$P(\theta^{(m)}|\psi) = P(L^{(m)}|\theta_{L^{(m)}}^{(m)}, \psi) \prod_i P(R_i^{(m)}|R_i) P(y_i^{(m)}|y_i) P(x_i^{(m)}|x_i) P(A^{(m)}, \sigma_b^{(m)}, \epsilon^{(m)}) \quad (\text{E2})$$

Explicamos aqui que a distribuição E2 faz a junção de probabilidades dos pontos das relações, pontos de controle para cada tipo de relação e escalas existentes no nível tipo ψ da criação de um símbolo.

3.4.1 Mapeamento da trajetória

De acordo com [LST15], uma trajetória $T_i^{(m)}$ representada no nível de um token, é uma sequência de pontos no plano da imagem que representa um caminho de um traço completo. Cada trajetória $T_i^{(m)} = f(L_i^{(m)}, x_i^{(m)}, y_i^{(m)})$ é uma função determinística com um ponto inicial $L_i^{(m)} \in \mathbb{R}^2$, pontos de controle $x_i^{(m)} \in \mathbb{R}^{10}$ e um valor de escala a nível de token $y_i^{(m)} \in \mathbb{R}$. Os pontos de controle e escalas são versões ruidosas de suas contrapartes representadas no nível de tipo, gerando uma relação $P(x_{ij}^{(m)}|x_{ij}) = N(x_{ij}, \sigma_x^1 I)$ para pontos de controle e $P(y_{ij}^{(m)}|y_{ij}) \propto N(y_{ij}, \sigma_y^2)$ para escala. Pela proporcionalidade gerada, o valor de escala é truncada a um valor abaixo de 0.

A construção de uma trajetória $T_i^{(m)}$ é definida pela junção das curvas que mapeiam tanto a escala quanto os pontos de controle $y_i^{(m)} x_i^{(m)} \in \mathbb{R}^{10}$ a partir de um ponto inicial do plano de imagem $L_i^{(m)}$. Para cada final de uma sub-spline, a próxima será iniciada a partir do final da anterior, até que todas as sub-partes sejam alcançadas.

As relações no nível de token são exatamente iguais ao mapeamento do nível de tipo, exceto pela relação longa que permite uma variabilidade anexa à spline usando uma gaussiana $P(\tau_i^{(m)}|\tau_i) \propto N(\tau_i, \sigma_\tau^2)$ truncado ao fim de uma curva. Dado a trajetória inicial do traço anterior, a posição inicial $L_i^{(m)}$ é uma amostragem de $P(L_i^{(m)}|R_i^{(m)}, T_1^{(m)}, \dots, T_{i-1}^{(m)}) = N(g(R_i^{(m)}, T_1^{(m)}, \dots, T_{i-1}^{(m)}), \Sigma_L)$. A função $g(\cdot)$ localiza o traço com uma posição apropriada dependendo da relação. Para relações *independentes* temos $g(\cdot) = L_i$, onde L_i é a localização global do nível-tipo. Para relações *Fin* (e análogo a relação *Início*), $g(\cdot) = \text{final}(T_{u_i}^{(m)})$, onde $\text{final}(\cdot)$ seleciona o último elemento em uma sequência da trajetória de pontos. Para uma relação *longa*, $g(\cdot) = \text{spline-eval}(T_{u_i}^{(m)}, v_i, \tau_i)$ é a avaliação da posição τ_i ao longo da curva que define o v_i -ésimo sub-traço que construa a trajetória do traço $T_{u_i}^{(m)}$. Na figura 3.10 temos o exemplo de duas trajetórias especificadas (uma para cada traço) para um número três rasurado.

Figura 3.10: Exemplo: duas trajetórias a partir dos pontos iniciais de cada traço - $L_1^{(m)}$ e $L_2^{(m)}$.

3.4.2 Parâmetros para a Imagem resultante

Uma transformação de uma imagem resultante $A^{(m)}$ é um conjunto de $P(A^{(m)}) = N([1, 1, 0, 0], \Sigma_A)$ de acordo com [LST15], onde os primeiros dois elementos representam o controle de escala global da imagem resultante, e os outros valores o controle de translação do centro de massa de $T^{(m)}$.

Basicamente, o modelo de mapeamento dos traços como imagem, cria linhas finas em tons de cinza pelo percurso, onde elas são expandidas por dupla convolução com filtro $b[\frac{a}{12}, \frac{a}{6}, \frac{a}{12}, \frac{a}{6}, 1 - a, \frac{a}{6}, \frac{a}{12}, \frac{a}{6}, \frac{a}{12}]$ e valores de limiarização maiores que 1 ($a = 0.5$ e $b = 6$).

Esta imagem em tons de cinza recebe então uma perturbação por dois tipos de processos ruidosos, fazendo o gradiente mais robusto durante a otimização e encorajando uma solução parcial durante a classificação. O primeiro processo cria uma espécie de borrão ao modelo de renderização do caractere. O borrão é realizado por uma convolução com filtro gaussiano de desvio padrão $\sigma_b^{(m)}$. O desvio padrão que gera esse ruído é um valor aleatório, amostrado de uma distribuição uniforme em um alcance especificado ($\sigma^{(m)} \text{ uniform}(0.5, 16)$), permitindo ao modelo uma melhor adaptação fiel a uma imagem. O segundo ruído altera a coloração de pixels mais próximos com probabilidade estocástica $\epsilon^{(m)}$, mantendo a uniformidade do traço esperado da imagem. A variação do valor de ruído para esta variável é aleatória ($\epsilon^{(m)} \text{ uniform}(.0001, 0.5)$), adaptativa ao conjunto durante a inferência. Os pixels em escala de cinza então parametrizam uma imagem de resolução 105x105, completando o modelo de probabilidade desse símbolo resultante: $P(I_{(m)}|\theta_{(m)}) = P(I^{(m)}|T^{(m)}, A^{(m)}, \sigma_b^{(m)}, \epsilon^{(m)})$. Na figura 3.11 são representados os passos de construção das seções 3.3 e 3.4.

Figura 3.11: Exemplo de criação de caracteres pelos dois níveis do BPL (ψ e θ): partindo das primitivas utilizadas por cada um até as suas respectivas imagens digitais. (representação do artigo [LST15]).

3.5 Caminhos aleatórios (em grafos)

Um grafo $G(V, E)$ é definido por um conjunto de ramificações ou caminhos que possuem vértices $V = \{v_1, v_2, \dots, v_n\}$ e um subconjunto de pares não ordenados de V , chamados arestas $E = \{e_1, e_2, \dots, e_m\}$. Para esta aplicação o grafo não é direcionado, ou seja, para cada vértice existente no grafo, o caminho a ser escolhido não depende de uma direção pré-determinada na con-

figuração geral da estrutura desse grafo sobre a imagem correspondente. Logo, consideramos que o grafo a ser utilizado para o pré-processamento se restringe aos possíveis vértices e arestas existentes em um esqueleto extraídos de uma imagem (mais detalhes em 4.2.1).

Um passeio finito de comprimento k em um grafo é uma sequência alternante de vértices e arestas $v_1, e_1, v_2, e_2, \dots, v_{k-1}, e_{k-1}, v_k$ que sempre começa e termina em um vértice. Nosso grafo é definido como sendo um conjunto de pixels de uma imagem. Logo, para cada aresta e_i existem pixels $u_1, u_2, \dots, u_s$ que o compõem. Então um caminho entre dois vértices em um esqueleto de uma imagem é definido como:

$$v_i, u_{i+1}, u_{i+2}, \dots, u_s, v_{i+s} | e_i = \{u_{i+1}, u_{i+2}, \dots, u_s\}$$

Um vértice de borda ou terminal é definido como sendo o vértice v_i de grau 1, ou seja, ele terá ligação apenas com uma aresta e_i . Veja na Figura 3.12 o exemplo de um grafo extraído de uma imagem, e três possíveis passeios realizados a partir de um mesmo vértice:

Figura 3.12: Um passeio aleatório em um grafo (em amarelo) sendo realizado a partir do vértice superior da cruz. Pontos em vermelho são os vértices do grafo, traço em verde são arestas do grafo. a), b) e c) representam três possíveis passeios a partir de um mesmo vértice.

De forma geral, um passeio aleatório no grafo da imagem é um caminho construído de acordo com as seguintes propriedades:

- Há um ponto de partida ($v_1 = 0$);
- A distância de um vértice até o outro é $s = |u|$;
- A diferença de um vértice no caminho para o próximo é escolhido aleatoriamente, e nenhuma direção é mais provável do que a outra.

Para passos determinantes de continuidade onde, o vértice atual v_i se encontra em um cruzamento com mais de uma aresta: $e_1, e_2, \dots, e_m$ (tal qual o vértice possui minimamente grau 2), expandimos o campo de “visão” do vértice v_i por uma projeção circular de raio $r = 5$. Esta projeção vai detectar a partir do vértice: pelo menos 2 e no máximo 5 pixels anteriores de $e_{i-1} | e_{i-1} = \{u_{i-1}, u_{i-2}, \dots, u_x\}$ para $2 \leq x \leq r$. Definição esta análoga ao conjunto de pixels da aresta e_i posterior ao vértice v_i . Observe que na figura 3.12 a delimitação desses pixels anteriores e posteriores ao vértice são relacionados respectivamente pelas cores azul e rosa no percurso de um novo passeio.

Vamos considerar que cada conjunto de pixels posteriores e anteriores ao vértice v_i serão vetores: logo, para determinar um possível vetor precisamos calcular a norma para os dois vetores da seguinte forma:

$$|\vec{a}| = \sqrt{(x_{u_{i-x}} - x_{u_{i-1}})^2 + (y_{u_{i-x}} - y_{u_{i-1}})^2}$$

$$|\vec{p}| = \sqrt{(x_{u_{i+x}} - x_{u_i})^2 + (y_{u_{i+x}} - y_{u_i})^2},$$

onde $|\vec{a}|$ é o comprimento do vetor anterior ao vértice v_i e $|\vec{p}|$ o comprimento do posterior. Para as novas coordenadas (x,y) de $\vec{a}$ e $\vec{p}$ em relação a origem (0,0) do plano cartesiano, temos:

$$\begin{aligned}\vec{a} &= (x_{u_{i-x}} - x_{u_{i-1}}; y_{u_{i-x}} - y_{u_{i-1}}) \\ \vec{p} &= (x_{u_{i+x}} - x_{u_i}; y_{u_{i+x}} - y_{u_i}),\end{aligned}$$

Ao final, o ângulo θ_A entre esses dois vetores é obtido através do produto escalar:

$$\theta_A = \arccos \frac{\vec{a} \cdot \vec{p}}{|\vec{a}| |\vec{p}|},$$

Nas sub Seções metodológicas 4.2.1 e 4.2.2 são descritos como toda a estrutura de passeio aleatório em grafos e o fator de escolha por ângulo θ_A se comportam na fase de ajuste inicial de uma imagem digital para a detecção das características de acordo com a base de parâmetros determinada pelo BPL.

3.6 O traço de um caractere

Ao final da execução da função de caminhos aleatórios, é realizado uma suavização em cada passeio capturado, gerando curvas que mapeiam os traços e sub-traços do caracteres caracterizados pelo BPL. De acordo com [MS15], as splines são uma boa forma de representar as curvas suavizadas com funções polinomiais.

Para descrever uma curva, nós precisamos abordar alguns fatos a respeito de suas propriedades. Ao contrário de algumas formas geométricas como o círculo que possui uma propriedade de descrição única (seu valor de raio e posição do centro), existem variados tipos de curvas com graus de liberdade e tracejado diferentes uma da outra. Isto porque necessitamos de um conjunto maior de atributos que consigam descrever variadas formas individuais de curvas. Algumas propriedades são atribuídas a uma região local da curva, enquanto outras propriedades requerem um conhecimento total da mesma.

As propriedades locais são importantes ferramentas pois descrevem as características de uma curva mais facilmente comparadas as propriedades globais. Essas propriedades podem ser descritas como:

- continuidade;
- posição em um local específico da curva;
- direção em um local específico da curva;
- curvatura (separado por graus de derivadas - aumento de suavidade).

O fator de continuidade está ligado diretamente a linearidade entre duas ou mais peças na geração de uma curva dentro de um determinado tempo. Em termos de análise, podemos verificar se a derivativa na primeira parte de uma curva corresponde com a sua parte consecutiva, ou seja, não existe uma mudança brusca na linearidade dentre as n peças de uma curva C^n . Esta visão também ajuda a definir as primitivas associadas (partes de um traço completo) de um texto ou

símbolo completo. Dependendo da restrição utilizada em uma função polinomial que mapeia a curva, podemos perder a continuidade na criação suave de uma curva sobre os pontos de controle. As chamadas B-Splines provêm a liberdade necessária para a manipulação de curvas independentes entre si, alinhadas aos pontos de controle distribuídos em toda sua trajetória; uma característica intrínseca existente em manuscritos de dígitos ou símbolos.

3.6.1 B-Splines

B-Splines aproximam um conjunto de n pontos com uma curva feita de funções polinomiais de grau d , resultando em $C^{(d-1)}$, ao fato em que C é a caracterização desta curvatura com grau $(d-1)$ em relação aos pontos de controle p . O modelo permite que as curvas sejam geradas a partir de qualquer grau de continuidade desejado. Veja na figura 3.13 o exemplo de uma b-spline de grau 2:

Figura 3.13: Exemplo de uma b-spline de grau 2. Os pontos de controle são definidos em vermelho; os nós em laranja são os pontos existentes na curva C gerada por um polinômio.

Uma B-Spline também é mencionada como a generalização das curvas de Bezier. Vamos considerar um vetor de nós definido como:

$$T = [t_0, t_1, \dots, t_m],$$

onde T é uma sequência de pontos com intervalo $t_i \in [0, 1]$, e com pontos de controle que variam de $p_0, \dots, p_n$. Assim temos a função de base:

$$b_{i,0}(t) = \begin{cases} 1 & \text{para } t_i \leq t < t_{i+1} \\ 0 & \text{caso contrário.} \end{cases}$$

$$b_{i,d}(t) = \frac{t - t_i}{t_{i+d} - t_i} b_{i,d-1} + \frac{t_{i+d+1} - t}{t_{i+d+1} - t_{i+1}} b_{i+1,d-1}$$

De acordo com [MS15] cada B-Spline pode ser criada a partir de um conjunto de $d+1$ polinômios de grau d . Uma curva será basicamente a combinação linear de um conjunto de outras B-Splines, que nos remete a notação paramétrica da curva:

$$f(t) = \sum_{i=1}^n p_i b_i(t), \quad (\text{EC1})$$

onde p_i são os pontos de controle (coeficientes) e b_i são as funções de bases. A chave para que a função EC1 funcione adequadamente é através da escolha apropriada de b_i . A variável t obviamente são os pontos de controle utilizados como domínio desta função paramétrica. Um conjunto de B-splines podem ser definidos por um número de coeficientes n e um valor de parâmetro k . O valor de k é um mais o grau de polinômios usados para montar a B-spline ($k = d + 1$).

As B-splines contemplam de vários métodos gerais para a criação de funções descrita por um número útil de propriedades. Uma curva com n pontos criados com uma b-spline e valor de parâmetro k :

- são $C^{(k-2)}$ contínuos;
- são feitos de polinomiais de grau $k-1$;
- possui um controle local - qualquer região na curva depende apenas de k pontos de controle;
- é delimitado pela distribuição convexa dos pontos de controle;
- possuem invariância afim - os pontos que definem a curva não serão mutáveis em relação as transformações de escala, rotação ou translação;
- exibe a propriedade da variação de encolhimento representado na figura 3.14.

Figura 3.14: Exemplos de b-splines criadas a partir de 5 pontos de controle (em vermelho). Note a característica de variabilidade de encolhimento da curva entre os pontos iniciais e finais do polinômio.

Uma curva criada usando b-spline não necessariamente interpola seus pontos de controle. Todos os traços suavizados a partir do conjunto de caminhos retirados do grafo inicial serão computados por uma b-spline cúbica. Uma b-spline de terceiro grau é definida por 4 partes de funções polinomiais ($k = 3 + 1$). Em EC2 temos a expressão dos polinômios para a construção de uma possível b-spline cúbica:

$$f(t) = \frac{1}{6}(-t^3 + 3t^2 - 3t + 1)p_i + \frac{1}{6}(3t^3 - 6t^2 + 4)p_{i+1} + \frac{1}{6}(-3t^3 + 3t^2 + 3t + 1)p_{i+2} + \frac{1}{6}t^3 p_{i+3} \quad (\text{EC2})$$

Considere o vetor de nós $t = [i, i + 1, i + 2, \dots, i + m]$ para a geração de uma curva suavizada; nós podemos escrever a função ao longo da curva entre os pontos $i + 3$ e $i + 4$ como uma função de parâmetro $t_i \in [0, 1]$ sobre a influência de curvatura dos 4 pontos de controle $p = [i, i + 1, i + 2, i + 3]$. O resultado expresso separadamente das partes da curva entre os 4 intervalos de t é apresentado com mais detalhes por [MS15].

Capítulo 4

Metodologia

Neste capítulo é apresentada a metodologia de nosso modelo com embasamento em aprendizado cognitivo e aplicável através de inferência Bayesiana, utilizando as definições conceituais expressas no capítulo 3. Partindo do princípio de que para se classificar os dados de imagens rasuradas em cima de um conjunto de imagens sem rasura se torna necessário a inversão de construção de características até então determinadas pelo BPL, logo: dado uma imagem digital pré-definida, criaremos programas Bayesianos que caracterizarão todos os traços e sub-traços daquele dígito ou caractere, ao passo de que suas variáveis de nível tipo e nível token serão indispensáveis para a fase de classificação.

Figura 4.1: Fluxograma do modelo de classificação de aprendizado em uma execução.

Na figura 4.1 apresentamos o fluxograma sobre o funcionamento de toda a estrutura do modelo de aprendizagem cognitivo e identificação de rasuras a partir do momento em que eu carrego uma nova imagem de testes no classificador.

Na Seção 4.1 mostramos com detalhes todo o banco de imagens utilizados neste projeto. Em 4.2 são detalhadas as estruturas necessárias do conjunto de ajustes feitos pelo BPL para as imagens. Na seção 4.3 apresentamos nossa estrutura adicional que realiza a predição de números rasurados (imagens de teste) comparado ao conjunto de símbolos numéricos sem rasura (treinamento). Mais alguns detalhes a respeito da função de predição são apresentados no Apêndice da pesquisa.

Neste projeto pretendemos comparar nossos resultados com trabalhos que sigam a mesma linha (apresentados no Capítulo 2), levando em maior consideração o trabalho de [CA17], onde montam uma estrutura de predição de textos rasurados utilizando máquina de vetores de suporte (SVM). Em seus resultados, um grupo de voluntários gerou 500 (250 em inglês e 250 em bengali) novos exemplos de páginas manuscritas. Um total de 1935 palavras foram identificadas para os documentos em inglês e 1432 palavras em bengali. Usaram 100 páginas de forma aleatória (50 para cada língua) com conjunto de imagens de treinamento do classificador SVM. Comparando ao modelo de ajuste cognitivo, esse é o principal processo descrito pelo fluxograma que este modelo vai propor uma visão de alto nível para a possibilidade da detecção de rasuras em palavras, sem a necessidade de uma forma de ajuste tradicional que necessite de um poder de processamento massivo (reflexo do alto número de imagens de treinamento necessárias) se compararmos com a visão de reconhecimento rápido desse tipo de problema por qualquer pessoa.

4.1 *Dataset*

Um banco de imagens foi criado voluntariamente pelo autor, onde tal decisão permitiu avaliar o grau de liberdade do classificador quanto aos tipos de traços e padrões independentes feitos por cada pessoa; consistindo em um total de 200 imagens de dígitos numéricos com 10 classes (números de 0 a 9) assim como no MNIST. Cada imagem possui resolução de 105x105 pixels no formato *.png*. Deste total, 100 imagens foram separadas para um novo ajuste considerando a melhoria do modelo a ser apresentado, ou seja, tanto as imagens de teste quanto as de treinamento não possuem rasura.

No segundo conjunto de 100 imagens: temos 50 imagens com rasura, destinada para o ajuste de testes, e 50 sem rasuras, para o ajuste de treinamento. A predição em cima do novo modelo de verificação de características entre os dois tipos, consiste do reajuste das características das imagens de treinamento sobre as imagens de teste.

As rasuras a serem identificadas sobre as 50 imagens de treinamento criadas, seguirão os modelos de controle¹ típico apresentado em [CA17] como: traço simples, traço múltiplo, inclinado, cruz, zig-zag e onda. A pesquisa não se prende a um formato específico de imagens ou sistema de padronização de imagens, como o Reconhecimento óptico de caracteres (OCR). Qualquer pessoa poderá gerar seus próprios exemplos. Lembrando que quanto maior a resolução das imagens, mais lento será a execução do modelo na fase de ajuste inicial: tanto na identificação de curvas quanto no mapeamento de todos os caminhos possíveis que simulam cada traço do manuscrito. O pré-processamento é melhor aplicado em imagens que contenham uma escala pequena de cores (preto e branco por exemplo).

¹para um novo *dataset*, poderá ser criado novos tipos de rasuras sobre os dígitos. Através das amostras reajustadas, o modelo não se prende ao tipo de rasura que ele deve identificar.

Figura 4.2: Exemplos de algumas imagens de testes. (a) imagem com rasura simples. (b) inclinada. (c) em onda. (d) cruz. (e) múltipla. (f) zig-zag.

4.2 Ajuste de programas Motores

Programas motores definem as estruturas necessárias do conjunto de ajustes expressos pelas seções 3.3 e 3.4. O princípio de entendimento de um programa motor é definido como sendo o melhor conjunto de amostras que caracterizam uma imagem digital, seja ela uma imagem de testes ou treinamento, explorando o largo espaço combinatório para identificar os diferentes números de partes do nível tipo, sub-traços e traços, e todas as relações possíveis (sub-seção 3.3.1) entre eles. Em [LST15] é citada várias abordagens aplicadas pela academia que trazem um certo resultado na extração desse tipo de característica, mas que infelizmente pecam em nível e otimização para a criação desses contêineres.

O número de motores a serem criados depende do valor da variável K , que neste projeto usamos $K = 5$ amostras necessárias para cada ajuste de primitivas em cima de um símbolo (tanto para testes quanto para treinamento). Assim, a estrutura de programas motores (amostras) são identificados como $\psi^{[1]}, \theta^{(m)[1]}, \dots, \psi^{[K]}, \theta^{(m)[K]}$ (mais promissores candidatos escolhidos pelo algoritmo). Uma distribuição discreta de aproximação é necessária para identificar os pesos associados a cada amostra de pontuação marginal daquela curva. A distribuição é expressa por:

$$P(\psi, \theta^{(m)} | I^{(m)}) \approx \sum_{i=1}^K w_i \delta(\theta^{(m)} - \theta^{(m)[i]}) \delta(\psi - \psi^{[i]}) \quad (\text{E3})$$

onde cada peso w_i é proporcional a pontuação da amostra, marginalizando sobre as variáveis de superfície de nível tipo (sec. 3.3) x e posições de anexo τ , com restrição $\sum_i w_i = 1$,

$$w_i \propto \tilde{w}_i = P(\psi_{x,r}^{[i]}, \theta^{(m)[i]}, I^{(m)}) \quad (\text{E4})$$

Ao invés de se usar apenas um ponto para estimar cada amostra, a aproximação pode ter uma melhoria incorporando uma variância local sobre o modelo. As variáveis de nível-token (sec. 3.4) permitem uma pequena variabilidade desde que elas estejam bem próximas a estrutura da imagem de entrada. Esta variância é estimada com uma melhoria sobre a distribuição discreta do nível-tipo da equação E3. Um algoritmo utilizado para esta tarefa seria *Metropolis Hastings* [CG95], um poderoso método de cadeias de Markov para simular distribuições multi-variadas. Este algoritmo produz

N amostras para cada motor (amostra de análise) $\theta^{(m)[i]}$, denotado por $\psi^{[i1]}, \dots, \psi^{[iN]}$, gerando a seguinte melhoria de aproximação:

$$P(\psi, \theta^{(m)} | I^{(m)}) \approx Q(\psi, \theta^{(m)}, I^{(m)}) = \sum_{i=1}^K w_i \delta(\theta^{(m)} - \theta^{(m)[i]}) \frac{1}{N} \sum_{j=1}^N \delta(\psi - \psi^{[ij]}) \quad (\text{E5})$$

Dado uma aproximação posterior de uma imagem em particular, o modelo pode avaliar a pontuação preditiva de uma nova imagem através de um reajuste das variáveis de nível do token.

4.2.1 Esqueleto da Imagem

Em [LST15] é aplicado um algoritmo para a extração do esqueleto da imagem bem parecido com a metodologia usada em [CA17]. Esta técnica consiste em encontrar as regiões de pixels centralizados, formando um grafo de caminhos, onde cada vértice desse grafo são os pixels existentes nos extremos da imagem e nos cruzamentos. As arestas desse grafo seriam os pixels que formam uma fina linha entre dois vértices. Este grafo de pontos poderá fornecer pontos de controle nos cruzamentos que serão cruciais para a aplicação do BPL na criação das amostras ajustadas (motores).

Figura 4.3: Exemplo de um esqueleto de imagem representado no Matlab.

Infelizmente esse método identifica alguns candidatos a vértices desnecessários para a geração dos motores (pontos próximos identificados dentro do círculo vermelho do grafo (cru) da figura 4.3). Muitas dessas inconsistências foram corrigidas removendo alguns desses ramos espúrios com um critério de detecção circular [LH90] em uma região dessa imagem. Este algoritmo coloca um círculo fixo em cada ponto identificado no grafo. Para mais de um ponto que for identificado dentro dessa distância específica em um círculo, será realizado um *merge* em todos esses pixels (como é representado pelo grafo limpo).

4.2.2 Gerando amostras aleatórias

Uma amostra candidata é gerada por um caminho aleatório sobre o esqueleto do caractere alvo, visitando nós até que cada aresta (caminho) tenha sido percorrido ao menos uma vez. Desde que o espaço de amostras cresça exponencialmente em números de bordas, os caminhos aleatórios são

necessários para explorar o máximo das partes do espaço global de um caractere. Assim, [LST15] utilizaram de um caminho de probabilidade estocástico A que minimize o ângulo local da trajetória do traço em torno de um ponto decisivo:

$$P(A) \propto \exp(-\lambda\theta_A), \quad (\text{E6})$$

onde θ_A é o ângulo associado a possível melhor escolha em um cruzamento que existam duas arestas ou mais a serem escolhidas pelo vértice atual e λ é um parâmetro que aumenta o peso de decisão do vértice em relação a cada “esquina” identificada no mesmo caminho. Além da distribuição E6 ser utilizada como principal fator de decisão na construção de um caminho, existem mais dois ângulos de custo fixo para duas situações em especial:

- 45° para o custo do ângulo caso o passeio chegue em um vértice de borda (simulação do processo de “levantar” uma caneta);
- 90° para o custo do ângulo caso o passeio esteja sendo feito em caminhos já identificados anteriormente.

Ao final de um passeio, além do acréscimo do custo de 45 graus para a escolha de um novo vértice, o modelo escolherá o próximo seguindo uma proporção $\frac{1}{b^\gamma}$, onde b é o número das novos (não visitados) vértices de bordas do nó e γ uma variável que determina o quão “forte” será a escolha de um novo nó b . O caminho aleatório irá processar isso repetidas vezes até gerar uma quantidade de candidatos para as amostras. Como métrica padrão, foi fixado uma quantidade máxima de 150 amostras, ou até atingir 100 únicos traços, compartilhados ao longo de todos os conjuntos de amostras. Diferentes valores de λ e γ são mapeados antes de começar cada caminho aleatório, produzindo ambos baixa e alta entropia nos caminhos candidatos.

Este pré-processamento possui uma falha quando aplicado a uma imagem de maior complexidade como o manuscrito de textos completos, palavras, ou que contenha várias interrupções de um caminho a outro no grafo. Quanto maior esse grafo, e quanto mais ramos existirem dentro dele, maior o tempo de execução para identificar esses modelos. Apesar de existirem as variáveis que limitem a quantidade de amostras ou traços, esse caminho leva um longo percurso de busca até identificar uma dessas características e confirmar na contagem. Esse foi um dos motivos de mantermos a utilização de imagens simples no ajuste inicial. Na figura 4.4 temos um conjunto de amostrar aleatórias capturadas após a realização dos passeios sobre o esqueleto de um dígito rasurado. A pontuação acima de cada dígito é descrita com maior detalhe na Subseção 4.2.3.

4.2.3 Busca por traços e sub-traços

Antes que uma amostra possa ser pontuada (eq. E4), os traços precisam ser subdivididos em sub-traços. Para isso, os traços em cada caminho é suavizado para seguir a estrutura característica do traço original da imagem, só que de forma corrigida em relação ao grafo limpo encontrado através da técnica explicada em 4.2.1. Os traços suavizados são então divididos em sub-traços com a execução de um algoritmo guloso de busca (identificando “pausas” durante os caminhos que mapearão esses sub-traços). Para pontuar a decomposição, os sub-traços são classificados com primitivas z_i e pontuado pelo modelo generativo de traços:

Figura 4.4: Conjunto de amostras aleatórias geradas a partir de uma imagem de entrada. O retângulo vermelho representa as 2K melhores amostras em ordem decrescente (do melhor para o pior) pontuadas pelo processo de busca por traços e sub-traços.

$$P(x_i^{(m)}, y_i^{(m)}, z_i) = P(z_i) \prod_{j=1}^{n_i} P(y_{ij}^{(m)} | y_{ij}) P(y_{ij} | z_{ij}) \int P(x_{ij}^{(m)} | x_{ij}) P(x_{ij} | z_{ij}) dx_{ij}, \quad (E7)$$

onde y_i é um valor ajustado próximo a $y_i^{(m)}$. Depois que o processo é executado em cada trajetória do traço, cada candidato a programa motor com variável ψ e $\theta^{(m)}$ é completamente especificado e tem uma boa precisão da característica geral da imagem. A ordem do traço é otimizada para os melhores 2K programas motores (amostras ajustadas). Então finalmente a pontuação a priori $P(\theta^{(m)} | \psi) P(\psi)$ é usada para selecionar os K melhores candidatos para o próximo processo de otimização e refinamento. A figura 4.5 é um exemplo dos 5 melhores programas motores selecionados após o conjunto de permutações entre os traços das 10 amostras pontuadas pela distribuição E7.

Figura 4.5: Os 5 melhores programas motores de um dígito rasurado, pontuados através da equação E4 e selecionados dentro do grupo total de amostras aleatórias (figura 4.4).

4.2.4 Otimização e refinamento

Após a correção das variáveis discretas, o conjunto de variáveis contínuas (incluindo $L^{(m)}$, $\tau^{(m)}$, $x^{(m)}$, $y^{(m)}$, $\epsilon^{(m)}$, $\sigma_b^{(m)}$) é otimizado para ajustar os pixels da imagem com um “active-set” do Matlab (algoritmo de otimização interno), usando o modelo generativo de pontuação da equação E4. Existem duas simplificações para reduzir o número de variáveis: a transformada afim $A^{(m)}$ é desabilitada e as relações R_i não são especificadas e re-otimizadas durante cada avaliação da função objetiva (Eq. E4). Depois que a otimização encontrar um máximo local, a melhor junção de configurações de direções dos traços e ordem do traço é escolhido usando enumeração exaustiva para caracteres com cinco traços ou menos, enquanto é considerado subconjuntos aleatórios para caracteres mais complexos. Finalmente, a melhor relação de pontuação é escolhida, e uma busca gulosa prossegue dividindo traços e unindo traços já processados até que os pontos não sejam mais alterados.

4.2.5 Estimativa da Variância Local

Neste passo, o contêiner dos K-ésimos melhores programas motores (amostras) $\psi^{[1]}, \theta^{(m)[1]}, \dots, \psi^{[K]}, \theta^{(m)[K]}$ passaram pelo processo de refinamento pela permutação dos traços especificados na Subseção 4.2.4. Agora cada amostra passará por uma execução individual para gerar novas amostras que se deformarão ao longo de uma cadeia de Markov (MCMC) [LST15]; o que permitirá estimar a variância local em torno do nível-tipo de uma amostragem de probabilidade $P(\psi|\theta_{(m)[i]})$. Com o algoritmo de *Metropolis Hastings* [CG95] e uma simples proposta Gaussiana serão definidas as novas conjunturas de x , escalas y , posições globais L , e anexos τ . As ids z dos sub-traços (primitivas) serão atualizados com amostragens de *Gibbs*. Cada cadeia será executada por 200 iterações sobre essas variáveis e então serão sub-agrupadas gerando o resultado de $N = 10$ amostras uniformemente espaçadas e a serem utilizadas no cálculo da distribuição $Q(\cdot)$ (equação E5). Veja na figura 4.6 a pontuação final dos 5 programas motores após a fase de otimização e refinamento. Amostras estas que serão utilizadas na fase de reajuste para com o conjunto de imagens de treinamento.

Figura 4.6: 5 programas motores de um dígito rasurado após o processo de otimização e refinamento.

4.3 Identificação de rasuras e reajuste das imagens

No cálculo base destinado a predição das imagens, usaremos toda a estrutura de programas motores apresentados na seção 4.2, separando as características locais e globais a respeito das

amostras a serem comparadas. Nesta função, vamos analisar o custo associado as características locais de cada traço de treinamento para um respectivo traço de teste, e também uma pontuação que avalia as características globais da mesma amostra de treinamento e testes pré-ajustados.

Para as distribuições globais devemos pegar os pontos associados as características de nível-token definidas em 3.4 como: pontuação associada a transformação global da imagem $P(A^{(m)})$, distribuição de pontos de ruído $P(\epsilon^{(m)})$, distribuição do borrão de imagem $P(\sigma_b^{(m)})$, distribuição de probabilidade da imagem (sub. 3.4.2).

Para as distribuições locais usaremos de uma estratégia comumente trabalhada e reconhecida para comparar duas séries temporais: As deformações dinâmicas no tempo (DTW). Escolhemos essa abordagem, pois ela consegue mensurar pixel a pixel de duas formas de traços distintos independentemente de seu comprimento. Utilizamos o material teórico de [SB16], pois ele traz recentes melhorias do algoritmo do DTW quanto a sua otimização em tempo de execução. Na sub-seção 4.3.1, iremos explicar sucintamente o encaixe desse algoritmo para o nosso objetivo final: a apresentação das distribuições que caracterizam as pontuações locais e globais de todos os programas motores de um caractere. No apêndice B apresentamos com mais detalhes o algoritmo e a computação realizada para capturar o custo mínimo entre duas amostras.

4.3.1 Extração das características locais com DTW

Em séries temporais comumente utilizam do algoritmo de DTW para mensurar a distância entre duas curvas no espaço. Em nosso caso, uma série temporal não é exatamente o que caracteriza uma curva no plano, pois toda série temporal é definida por um vetor de n pontos pelo tempo t . Por se tratar de um conjunto de pontos em uma dimensão, precisamos ajustar o modelo para que ele consiga capturar as características associadas em duas dimensões. Com isso [SYHJ⁺17] apresenta uma generalização do DTW que permite compararmos essas séries em N dimensões.

Para duas curvas $C_1 = (x_n, y_n)$ e $C_2 = (x_m, y_m)$, calculamos a função que retorna a matriz de pesos cruzando cada dimensão: $DTW_1(x_n, x_m)$ e $DTW_2(y_n, y_m)$. Ao final, somamos cada dimensão capturando a lista de caminhos ótimos com a matriz $DTW_{final} = DTW_1 + DTW_2$. DTW_{final} é a matriz que contém o melhor (e também o menor custo na diagonal) caminho entre as curvas de duas dimensões C_1 e C_2 .

Na figura 4.7 apresentamos o custo associado entre os traços de uma manuscrito do número 6 (treinamento) com um 6 rasurado (teste). Este ajuste é realizado para cada amostra dos programas motores adquiridos no ajuste inicial. Assim, para cada amostra calculamos a média do menor custo dos melhores traços selecionados em cada comparação, que por sua vez, será nossa variável de característica local. Na equação E9 mostramos o somatório dos menores custos pela diagonal da matriz DTW_{final} :

$$path = [p_1, p_2, p_3, \dots, p_l] \mid n \leq l \leq n + m$$

$$minCost_k = \sum_{i=1}^l path_i \quad (E8)$$

onde $path$ é o vetor de custos tendendo pela diagonal da matriz, l o comprimento desse vetor e k é o k -ésimo traço de treinamento a ser comparado.

Figura 4.7: Comparação entre traços de uma figura sem rasura com uma rasurada. Abaixo dos dois números são selecionados os traços que melhor se ajustem à imagem de treinamento. Os pontilhados representam o melhor caminho entre dois pontos das duas curvas, ou seja, o menor custo na diagonal da matriz (DTW_{final}).

4.3.2 Junção das características locais e globais

A descrição do custo local mencionada anteriormente é representado da seguinte forma:

$$P(S_1, \dots, S_k) = \frac{1}{k} \sum_{i=1}^k \min Cost^k \quad (E9)$$

onde k é o número total das melhores combinações de traços do teste e o treinamento. E essa probabilidade destaca a média dos menores custos entre os traços, excluindo os possíveis traços que seriam identificados como rasura entre as duas amostras.

Para a variável global, subtraímos as características da imagem de teste com as de treinamento, deste modo identificaremos que quanto mais perto de 0 a pontuação da distribuição chegar, mais próximos serão suas características relacionada a uma imagem de treinamento no reajuste.

$$P(I_g^{(T)} | I_g^{(c)}) = P(I^{(T)} | A^{(T)}, \sigma_b^{(T)}, \epsilon^{(T)}) - P(I^{(c)} | A^{(c)}, \sigma_b^{(c)}, \epsilon^{(c)}) \quad (E10)$$

A equação E10 apresenta uma avaliação próxima ao cálculo de custos das variáveis locais, onde o valor de similaridade tenderá a 0 quando os traços forem semelhantes.

Após a captura das características locais e globais, se dá como necessário a separação dos traços de menor similaridade entre as duas imagens. Como já determinamos o custo local $\min Cost$, nós também teremos a referência desses traços admitindo apenas os que serão de maior relevância, ignorando a possível rasura. Nossa nova tupla terá a estrutura $\psi_{new} = \{k, S_k = \text{vecMax}_k, R\}$, onde

k é o total de traços selecionados dentre os mais próximos, $vecMax = [\psi_1^{(T)}, \dots, \psi_k^{(T)}]$ e R as relações que já estavam previamente ajustadas entre esses traços.

Tendo em vista o valor resultante da pontuação global e local entre teste e treinamento, agora precisamos definir o custo de similaridade existente nesse reajuste. Para isto, somamos a pontuação de característica global com o custo mínimo médio encontrado dentre todos os traços, resultando em:

$$similarity = P(I_g^{(T)} | I_g^{(c)}) + P(S_1, \dots, S_k). \quad (E11)$$

ψ_{new} será a componente do novo programa motor criado para o conjunto de predições, como representado graficamente nas amostras da terceira coluna da figura 4.8. Caso o número de traços nas imagens de treinamento seja maior que nas de teste, um programa motor vazio será criado como resultado dessa computação.

Figura 4.8: Exemplo de imagem $I^{(T)}$ sendo reajustada por um novo dígito $I^{(c)}$.

Para o reajuste das imagens, seguimos o modelo de [LST15], incrementando agora os resultados da equação E11. Sabendo o valor de similaridade associado a cada ajuste de treinamento e teste, iremos complementar o modelo Bayesiano para o aprendizado cognitivo já existente [PTGX11], onde a avaliação probabilística para esse aprendizado de poucos exemplos integra: $I^{(T)}$ como sendo a imagem de teste; e $I^{(c)}$ o conjunto de novas imagens de classes $\{c | c = 0, \dots, 9\}$. A classificação segue a regra de aproximação:

$$\log P(I^{(T)}|I^{(c)}) \approx \log \sum_{i=1}^K w_i \max_{\theta^{(T)}} P(I^{(T)}|\theta^{(T)}) \frac{1}{N} \sum_{j=1}^N P(\theta^{(T)}|\psi^{[ij]}), \quad (\text{E12})$$

$$\underset{c}{\operatorname{argmax}} \log P(I^{(T)}|I^{(c)}).$$

Intuitivamente, a aproximação aplica o algoritmo de busca BPL em cima de cada uma das $K = 5$ amostras de ajustes das imagens de treinamento $I^{(c)}$, utilizando das N -ésimas cadeias de *Markov* estimadas para identificar a variabilidade local de cada uma de suas amostras (vide SubSeção 4.2.5); para então com o procedimentos de otimização de gradiente reajustar suas variabilidades de nível token $\theta^{(T)}$ sobre as amostras ajustadas das imagens de teste $I^{(T)}$. O que explica a construção da aproximação de pontuação de reajuste apresentado em E12: onde o peso w_i assim como a média calculada dentre todas as 10 variabilidades locais identificadas por amostra seguem da distribuição E5.

Desta forma fizemos o balanceamento na escolha da classe adicionando o modelo a E11, que além do reajuste dos traços das amostras de treinamento $I^{(c)}$ sobre as de teste $I^{(T)}$, fará o mapeamento dos possíveis traços identificados como rasura e classificará um para um independente desse possível obstáculo:

$$\underset{c}{\operatorname{argmax}} \log P(I^{(T)}|I^{(c)}) + \log(\textit{similarity}). \quad (\text{E13})$$

Um exemplo de reajuste pode ser visualizado na imagem 4.8. A primeira e a segunda coluna demonstram a aproximação do reajuste das amostras de treinamento (mais a esquerda) sobre as imagens com rasura (coluna do meio). A coluna mais a direita exibe a imagem de teste alvo sem os possíveis traços que exibiam a rasura².

Para mais detalhes relacionados ao código fonte dessa Seção, consulte nosso apêndice A.

²Neste modelo não aplicamos um pós processamento como em [CA17], que remove os pixels indesejados das imagens de teste. Apenas mapeamos os candidatos a traços rasurados, atualizando um novo programa motor (já demonstrado na seção).

Capítulo 5

Resultados

Pela estrutura dos dados explicados na seção 4.1, foram separadas 100 imagens para obter um conjunto aceitável de resultados. Cada execução possui 20 imagens (10 para treinamento e 10 para testes). As imagens de testes são as únicas que possuem algum tipo de rasura. A acurácia é obtida conforme o resultado nos reajustes, aplicando a equação E13 em cada uma das imagens de testes. Para cada execução calcula-se o percentual de acertos pela quantidade total de classes. Nos resultados onde o valor da classe é igual ao rótulo especificado pela imagem de teste (neste caso os rótulos estão configurados no próprio nome da imagem, ex: *run1_test1.png*), o script vai plotar apenas uma análise a respeito da classe escolhida. Caso contrário, será plotado uma análise ao resultado encontrado pelo modelo Bayesiano junto à análise da classe correta.

Figura 5.1: Imagens utilizadas na acurácia da tabela 5.1. a) imagens de teste e b) imagens de treinamento.

execução	classes										acurácia
	0	1	2	3	4	5	6	7	8	9	
1	0	1	1	0	1	0	1	1	0	1	60%
2	0	1	1	1	0	1	1	1	1	0	70%
3	0	0	1	1	1	1	1	1	1	1	80%
4	1	1	1	1	1	1	0	1	0	1	80%
5	1	1	1	1	1	1	1	0	1	1	90%

Tabela 5.1: Acurácia para identificação de dígitos **com rasura**. 10 classes de números variando de 0 a 9. Em cada execução, o valor **1** representa o acerto para o par de imagens e **0** o erro.

Na tabela 5.1 são demonstrados os resultados independentes entre as 5 execuções para um conjunto de 10 classes (números de 0 a 9). Na figura 5.1 estão distribuídas todas as imagens utilizadas em cada execução desta tabela: considere que em cada linha, tanto para o conjunto de a) e b), temos a separação que compara cada classe de uma linha de a) com todas as classes de uma linha de b). Se

um número rasurado conseguir se identificar com a classe associada como correta em sua consecutiva linha de dígitos em b), receberá um acerto (1), caso contrário será um erro (0).

Também foi realizado o reajuste com outro conjunto de 5 execuções, ainda com o dataset de classes numéricas. Mas diferente do anterior, não foi utilizado rasuras nas imagens de teste. Foram feitos embaralhamentos entre as amostras para verificar se a proposta original do BPL apresentada por [LST15] obteve algum nível de melhoria com a adição da detecção de traços de menor termo característico entre dois ajustes. Após o reajuste, o modelo apresentou bons resultados para algumas execuções. Na tabela 5.2, notamos que entre as execuções de 1 a 3 a acurácia em cada foi satisfatória. Para a execução 5 em negrito, realizamos uma combinação proposital de números idênticos entre os ajustes de teste e treinamento como uma espécie de calibração do método.

Dentre todas essas 5 execuções, a quarta foi a que apresentou um pior resultado mesmo aplicando a melhoria. Com estes resultados de menor percentual, foi realizado uma avaliação das possíveis limitações a cerca deste método junto com um novo conjunto de dados de treinamento utilizados a seguir.

Figura 5.2: Imagens utilizadas na acurácia da tabela 5.2. a) imagens de teste e b) imagens de treinamento.

execução	classes										acurácia
	0	1	2	3	4	5	6	7	8	9	
1	0	1	1	0	1	1	0	1	0	0	50%
2	1	0	0	1	0	1	1	1	1	1	70%
3	1	1	1	1	1	1	0	1	1	1	90%
4	0	0	0	0	0	1	1	1	1	0	40%
5	1	1	1	1	1	1	1	1	1	1	100%

Tabela 5.2: Acurácia para identificação de dígitos *sem rasura*. 10 classes de números variando de 0 a 9. Em cada execução, o valor **1** representa acertos e **0** os erros. A execução 5 possui as mesmas imagens em teste e treinamento como meio de calibração do modelo.

Uma estrutura em particular dos dígitos construídos em nossa pesquisa, e até mesmo por conjuntos como o MNIST, é a sua construção por diferentes formas de tracejado. Notamos que para esses dígitos que apresentaram mudanças aparentes entre si na fase do reajuste, resultaram em uma menor acurácia mesmo com a nossa melhoria para identificação de rasuras. Veja que na figura 5.2 os números 1 e 7 por exemplo são bem similares quanto a distribuição dos traços, o que dificulta na predição do modelo para a escolha dos traços em comum que definem o dígito. Assim, criamos mais um conjunto de imagens de treinamento para a realização de mais 5 execuções, aproveitando o conjunto de imagens de testes sem rasura do conjunto de dados anterior. Na figura 5.3 construímos um novo conjunto de imagens de treinamento que serão reajustadas com as imagens de teste do dataset sem rasura, mas para este realizamos de forma que a morfologia dos dígitos não se mostras-

sem tão distantes em relação aos dígitos das imagens de teste (tais alterações se limitam a pequenas transformações de escala no centro de massa de cada dígito, rotação e translação).

Figura 5.3: Imagens utilizadas na acurácia da tabela 5.3. Novo conjunto de imagens para o treinamento para as imagens de teste da figura 5.2.

execução	classes										acurácia
	0	1	2	3	4	5	6	7	8	9	
1	1	0	1	1	1	1	1	1	1	0	80%
2	1	1	0	1	1	1	1	1	1	1	90%
3	1	1	1	1	1	1	0	1	1	1	90%
4	1	1	1	1	1	1	1	1	1	1	100%
5	1	0	1	1	1	1	1	1	1	1	90%

Tabela 5.3: Acurácia para identificação de dígitos *sem rasura*. 10 classes de números variando de 0 a 9. Em cada execução, o valor **1** representa acertos e **0** os erros. Este conjunto de execuções foi realizado com o dataset de imagens sem rasura, mas com um novo conjunto de imagens de treinamento estabelecidos pela figura 5.3.

Na tabela 5.3 obtivemos resultados promissores quanto ao reajuste, o que demonstra que para um alto grau de modificação em cima de um caractere, ou até mesmo caracteres ou dígitos que podem ser construídos de formas distintas, podem ocasionar em uma queda considerável para a acurácia. Essas limitações serão pontuadas mais a fundo nas considerações finais de nossos resultados, assim como iremos identificar as dificuldades do modelo a cerca da complexidade em cada tipo de rasura com o próximo conjunto de dados.

Para verificar o nível de melhoria do modelo, foram realizados uma bateria de testes também no dataset original apresentado por [LST15]. Em seu projeto disponibilizado no *github*, eles criam um conjunto de 20 execuções, onde em cada execução temos uma quantidade de 20 símbolos para imagens de testes e mais 20 símbolos (com pequenas deformações no traço) para imagens de treinamento. A proposta original da predição pelo autor era justamente averiguar o potencial do BPL quanto a pequenas deformidades de um mesmo manuscrito realizado por diferentes pessoas. Com isto, todas as imagens pertencentes ao conjunto de testes foram alteradas, aplicando os mesmos tipos de rasuras expressos no conjunto de exemplos anteriores (rasura simples, inclinada, onda, cruz, múltipla, e zig-zag). Feito tais modificações em cima das imagens de teste, são então criados novos programas motores de acordo com a metodologia 4.

Diferente dos símbolos numéricos, em cada execução¹ deste dataset o autor considera um conjunto diferente de letras de um determinado alfabeto. Abaixo segue uma tabela comparativa com

¹A complexidade em cada execução deve ser analisada de forma isolada, ou seja, deve ser considerado que para cada execução seu espectro de acerto se limita ao conjunto total de classes (no caso o total de 20).

as predições para com esse conjunto de dados que totalizam em 800 imagens (testes e treinamento).

Execuções	tipo rasura	Acurácia
1	simples, inclinada	70%
2	múltipla	65%
3	zig-zag	40%
4	cruzada	40%
5	onda	40%
6	simples	60%
7	inclinada	55%
8	múltipla	30%
9	zig-zag	40%
10	cruzada	20%
11	onda	55%
12	simples	40%
13	inclinada	45%
14	onda	25%
15	cruzada	30%
16	múltipla	35%
17	zig-zag	35%
18	simples	45%
19	inclinada	50%
20	onda	45%

Tabela 5.4: Acurácia para identificação de caracteres do Omniglot [LST15] com rasura.

Figura 5.4: Algumas poucas imagens utilizadas na acurácia da tabela 5.4 (mais especificamente entre as execuções 1 e 5). a) imagens de teste e b) imagens de treinamento. As imagens em a) contemplam todos os tipos de rasuras utilizados no conjunto de dados completo.

Para vários casos apresentados na tabela 5.4, note que o modelo apresentou certa dificuldade em prever várias das imagens de testes do Omniglot [LST15]. Assim como nos conjuntos de dados numéricos anteriores, em cada linha em a) e b) da figura 5.4 se separam o conjunto de imagens utilizados em cada execução (mais especificamente entre as execuções 1 e 5). Dada esta descrição, o gráfico de barras da figura 5.5 traça uma importante descrição que servirá como complemento justificativo de análise geral desses resultados. Veja que para rasuras do tipo simples e inclinada,

os valores foram promissores se comparado ao conjunto de execuções que contemplam as do tipo zig-zag e cruzada.

Figura 5.5: Conjunto de predições separados por tipos de rasura.

Sabendo a respeito da melhoria que obtivemos na acurácia da tabela 5.3 em relação a tabela 5.2 e o conjunto de acertos avaliados pela complexidade de cada rasura no gráfico da figura 5.5, construímos abaixo o raciocínio que define as características limitantes da estrutura do BPL, que são válidas independentemente se o caractere possui rasura ou diferenças em sua construção morfológica:

- O primeiro é relacionado a quantidade de quebras de traços existentes nas amostras de testes rasuradas: quanto maior o número de quebras, mais complexo será o reajuste entre essas amostras pela disposição da imagem de treinamento sobre o teste. Repare na imagem 5.6, onde realizamos o teste da captura dos custos em uma das amostras da execução 10 (tabela 5.4), que o símbolo rasurado (teste) possui quebras cruciais resultantes da disposição da rasura em cruz que não permitem um bom reajuste diante da imagem de amostra referência (treinamento).
- O segundo problema é sobre a disposição morfológica dos traços existentes nos dígitos ou símbolos: mesmo para as execuções dentre o conjunto de imagens que não possuem rasura (exemplo de imagens de teste e treinamento da figura 5.2), a deformidade de sua construção morfológica será um fator importante que definirá a qualidade da acurácia, ou seja, ao compararmos dois dígitos ou símbolos de mesma classe e com “tracejados” diferentes, teremos uma queda no reajuste de uma para outra devido a essa diferença característica. Veja na figura 5.7 alguns exemplos na fase de reajuste da classificação que destacam justamente essa diferença na comparação direta entre dois dígitos.

Os resultados das 100 imagens com rasuras observadas na tabela 5.1, e as outras 100 com imagens sem rasura da tabela 5.3 foram relevantes na maioria das execuções, considerando a quantidade reduzida de exemplos necessários para a nossa predição e a disposição 1 a 1 (*one-shot learning*) entre cada par de amostras. Comparando com modelos tradicionais como o apresentado por [CA17], que apesar de uma acurácia promissora de 80%, seu classificador exige uma quantidade elevada de exemplos e também só separa para com duas classes (rasurado/sem rasura), ao contrário da técnica explorada neste trabalho que considera uma identificação de rasuras mais abrangente, não

Figura 5.6: Exemplo de computação do algoritmo DTW em um dos caracteres do Omniglot

Figura 5.7: Exemplo de comparação entre amostras de dois dígitos de mesma classe. Note que na segunda coluna dos números 2 (a) e 9 (b) o modelo encontrou dificuldade na realização do reajuste.

se limitando aos tipos de traços e identificações locais que o algoritmo pode rotular como sendo um possível problema da imagem. Quanto a bateria de testes com o dataset utilizado originalmente nas predições de [LST15], avalia-se os resultados como sendo medianos, já que foram expostos as limitações que permitiram que os resultados não atingissem um alto grau de aceitação. Este último serviu como um teste de estresse do modelo que permitiu separar os principais pontos que desencadeiam essa baixa acurácia tanto para imagens rasuradas (que é o objetivo principal) quanto para a até então proposta original de [LST15], na classificação de símbolos semelhantes.

Este projeto não busca descartar qualquer outro tipo de metodologia já aplicada na literatura, mas sim enfatizar que a técnica representada em cima do aprendizado cognitivo traz alguns avanços consideráveis ao que temos hoje publicamente disponível como: a identificação de símbolos ou caracteres independentemente da rasura pré-identificada, o que futuramente pode se escalonar para sistemas de reconhecimento e recuperações de imagens degeneradas, sejam esses livros ou artefatos históricos que por ventura poderão ser armazenados digitalmente. O modelo também permite que outros trabalhos futuros o utilizem como embasamento para estruturas mais complexas de pré e pós processamento na área de visão computacional; algoritmos que aumentem a segurança (ou que

consigam burlar) para com o acesso em certas páginas que apresentem um conjunto de caracteres gerados automaticamente para uma pré-identificação - mais conhecido como CAPTCHA (teste de Turing público completamente automatizado para diferenciação entre computadores e humanos).

Capítulo 6

Conclusões

Os resultados obtidos através do conjunto de dados com rasura foram expressivos, comparando com técnicas já pesquisadas e correlatas a esta dissertação que utilizem de um número extenso de dados (vide capítulo 2), nos remetendo a uma média global de 76% nos acertos (tabela 5.1). Para o segundo conjunto de dados com testes sem rasura (tabela 5.2), após a avaliação e utilização de um novo conjunto de dados de teste com pequenas deformidades morfológicas, obtivemos uma média global de 90% (tabela 5.3).

Pela bateria de execuções aplicadas em cima do conjunto de dados Omniglot, nota-se que o BPL ainda apresenta uma limitação inerente à todos os traços capturados por técnicas aplicadas no nível de ajuste dos programas motores. Podemos considerar que a Inferência Bayesiana se mostrou como uma possível saída encontrada atualmente na conversão do aprendizado cognitivo em ajustes reconhecidos por um computador; mas através do que foi apresentado, percebemos que algumas estruturas decorrentes desses pré-ajustes precisam sempre ser analisadas e moldadas de acordo com o conjunto de dados utilizados como entrada, como por exemplo a comparação pixel a pixel entre a paridade de traços das imagens de teste rasuradas com as de treinamento sem rasura deste trabalho. Ao aumentarmos o campo de complexidade de um problema, e. g. a adição de uma rasura em um texto, nosso método se comportou bem para as que não cobrissem muito os manuscritos, o que é interessante ao analisarmos que nós humanos não conseguimos identificar um caractere ou texto dependendo do grau de sua obstrução. Mas para alguns destes, ainda existe esse impedimento no ajuste fino para os moldes de representação passíveis da realidade que ainda se prendem nas estruturas bases computacionais que até então utilizamos. Uma sugestão cabível para um trabalho futuro, seria justamente investigar maneiras que facilitassem o rearranjo de hipóteses no processo de ajuste inicial nas imagens de teste ou treinamento, que não se prendam tanto a quantidade ou morfologias necessárias para todas as comparações. Não sabemos se essa visão de ajuste inicial está tão atrelada ao requisito mínimo para se computar uma pontuação utilizando do Teorema de Bayes, mas a melhoria precisa se concentrar em simplificar a maneira como nós representamos o conhecimento a priori de um objeto, ou para o nosso foco da pesquisa, uma imagem.

Outro fator construtivo deste projeto era fazer com que a técnica já apresentada aumentasse seu leque de estruturas existentes na ciência cognitiva[Nei14], onde notamos ser aplicado inconscientemente no dia-a-dia das crianças em desenvolvimento (foco para a etapa deste processo de aprendizado). Tendo em voga nosso embasamento bibliográfico, identificamos que poucos artigos retratam técnicas para a abstração característica de alto nível em um caractere com aprendizado por ‘um tiro’ ([LST15], [KZS15], [FF⁺03]); unindo este arcabouço apresentado sobre aprendizado

Bayesiano e ciência cognitiva, junto ao conjunto de resultados que obtivemos aplicando nosso método, podemos afirmar que: a contribuição desta pesquisa está no mérito da construção de um classificador que consiga aprender e predizer a classe de um dígito ou símbolo rasurado, sabendo que a informação a priori da rasura daquele caractere é inexistente, ou seja, o conjunto de conhecimento a respeito daquele caractere rasurado deve ser obtido através do aprendizado inicial que o computador possui de um conjunto limitado de caracteres sem rasura (ao fato que nosso modelo não se prende ao tipo de rasura existente na imagem a ser classificada). A complexidade atrelada a rasura vai pesar diretamente na qualidade da acurácia, limitações estas também existentes em nós humanos quando tentamos identificar textos deteriorados e avaliadas diretamente em nossos resultados.

Apêndice A

Código Fonte

Neste apêndice apresentamos a visão geral de algumas funções utilizadas na pesquisa. Dentre elas estão a explicação da função de predição e dos arquivos proeminentes no carregamento das imagens em forma de contêineres (programas motores). Todo o projeto possui como base a pesquisa de [LST15] com atualizações relacionado a detecção inata de símbolos numéricos.

O desenvolvimento e análise do código fonte foram realizados no MATLAB R2015a. Abaixo destacamos o hardware utilizado para a geração dos ajustes e classificação no modelo:

- CPU: Intel Core i7 - 4810MQ (2.8GHZ) - 8 núcleos
- GPU: Nvidia Geforce GTX 860M (2gb vram)
- Memória: 8gb ram

A.1 Geração do contêiner das imagens (Ajuste dos programas motores)

Algoritmo 1 Carregamento e pré-processamento das imagens de teste e treinamento

```

1:  $K \leftarrow 5$                                 ▷ números de amostras para uma imagem
2:  $verbose \leftarrow true$                         ▷ parâmetro para ilustrar o processo de pré-ajuste
3:  $include\_mcmc \leftarrow true$                     ▷ estimar variabilidade local?
4:  $fast\_mode \leftarrow true$                       ▷ parâmetro para ajuste fino de cada traço
5:  $runs \leftarrow 5$                                 ▷ número total de execuções
6:  $tests \leftarrow 10$                             ▷ número total de imagens de testes
7:  $trains \leftarrow 10$                             ▷ total de imagens de treinamento
8:  $files \leftarrow dir('modelSS\_fits/* .png')$       ▷ caminho do diretório das imagens
9:  $n \leftarrow length(files)$                       ▷ total de arquivos encontrados
10: for  $i = 1 \dots runs$  do
11:   for  $j = 1 \dots (tests + trains)$  do
12:     if  $j \leq tests$  then
13:        $st \leftarrow makestr('modelSS\_fits/run', i - 1, '_test', j - 1, '.png')$ 
14:     else
15:        $st \leftarrow makestr('modelSS\_fits/run', i - 1, '_train', j - 1 - tests, '.png')$ 
16:     end if
17:      $img \leftarrow imcomplement(im2bw(imread(st)))$ 
18:      $G \leftarrow fit\_motorprograms(img, K, verbose, include\_mcmc, fast\_mode)$ 
19:     if  $j < (tests + 1)$  then
20:        $st \leftarrow makestr('modelSS\_fits/run', i, '_test', j, '_G')$ 
21:     else
22:        $st \leftarrow makestr('modelSS\_fits/run', i, '_train', j - tests, '_G')$ 
23:     end if
24:      $save((st'.mat'), G)$                         ▷ salvar contêiner de programas motores
25:   end for
26: end for

```

Função que carrega todas as imagens identificadas pelo diretório *modelSS_fits* para a construção dos contêineres. Note que dentro do loop duplo é feita uma configuração para o rastreo das imagens de teste e de treinamento. Os dois tipos de imagem serão configurados pela mesma função que realiza o pré-ajuste (*fit_motorprograms*).

A.2 Predição das imagens de teste sobre as classes de imagem de treinamento (reajuste)

Abaixo segue o código fonte relacionado a função de reajuste e predição para a criação de uma nova execução dentre 10 classes de dígitos numéricos (números de 0 a 9). A estrutura de carregamento das imagens pré-ajustadas de teste e treinamento são feitos até a linha 10. O parâmetro *use_precomputed* separa a condição permitindo a execução rápida do reajuste dos resultados já calculados em uma primeira instância. Ao final o código imprime o conjunto de amostras de testes comparados com o treinamento assim como na figura 4.8 da Seção Metodológica. Na possibilidade de uma acurácia errada, a função irá imprimir uma tabela com a paridade de amostras incorretas e outra para as que seriam da classe correta.

Algoritmo 2 Predição do modelo (reajuste)

```

1: procedure PREDICT_EXAMPLE(itest, run, classes, use_precomputed, verbose)
2:   lib ← loadlib                                ▷ carregamento de pré-definições
3:   load(fullfile('modelSS_fits', itest), 'G')      ▷  $I^{(T)}$ 
4:   img_test ← G                                ▷ carregamento dos programas motores da imagem de teste
5:   pairGroup ← struct
6:   pair ← struct
7:   predict_score ← nan(classes, 1)                ▷ inicializando matriz de pontos da predição
8:   for c = 1...classes do
9:     fn_fit_train ← fullfile('model_fits', run, c)  ▷  $I^{(c)}$ 
10:    load(fn_fit_train, 'G')
11:    if use_precomputed then
12:      fn_refit_test ← fullfile('modelSS_refits', run, c, itest)
13:      load(fn_refit_test, 'pair')
14:      preComp(pairGroup, pair)                    ▷ pré-inicialização da estrutura já reajustada
15:    else
16:      K ← length(G.models)                        ▷ comprimento do tamanho de amostras K
17:      Mbest ← cell(K, 1)
18:      likeness ← nan(K, 1)
19:      modStroke ← cell(K, 1)
20:      fit_score ← nan(K, 1)
21:      prior_score ← G.scores
22:      for i = 1...K do
23:        [scoreL, modStrokei] ← scoreLikelihood(img_test.modelsi, G.modelsi, lib)
24:        likeness(i) ← scoreL
25:        [Mbesti, fscore] ← FitNewExemplar(img_test.img, G.samples_typei, lib)
26:        fit_score(i) ← fscore
27:      end for
28:      pairGroup(c).likeness ← likeness
29:      pairGroup(c).modStroke ← modStroke
30:      pairGroup(c).Mbest ← Mbest
31:      pairGroup(c).fit_score ← fit_score
32:      pairGroup(c).prior_score ← prior_score
33:      clear(pair)
34:      pair ← pairGroup(c)
35:      file ← makestr('modelSS_refits', run, c, itest)
36:      save([file, '.mat'], pair)
37:      print('File %s saved Sucessfully.', file)
38:    end if
39:    pairGroup(c).img_train ← G.img
40:    pairGroup(c).models ← G.models
41:    predict_score(c) ← log_post_pred(pairGroup(c))  ▷  $\log P(I^{(T)} | P(I^{(c)}))$ 
42:    print('Logposterior predictive score : %f', predict_score(c))
43:  end for
44:  [M, I] ← max(predict_score)
45:  print('Predicted class in run %d : %d Max Score : %d', run, I - 1, M)
46:  showRefitResult(pairGroup, img_test, I, verbose)  ▷ visualizar resultado obtido
47:  if I ≠ itest then
48:    I ← itest
49:    showRefitResult(pairGroup, img_test, I, verbose)  ▷ visualiza resultado esperado
50:  end if
51: end procedure
    
```

A.3 Comparação e soma das características de maior semelhança entre os programas motores de teste e treinamento

Função destinada a comparação entre os contêineres de teste e treinamento: linhas 2 e 3 captam os scores para o conjunto de programas motores gerados em cada um dos ajustes. A função *sumfield* (linhas 4 e 5) realiza uma soma recursiva para as características globais dos ajustes em teste e treinamento. Logo, é feito o cálculo dos custos locais através da função *compare_strokes* (linha 8 e algoritmo 4). Validamos através de uma condicional, se imagens de teste possuem mais traços em relação a treinamento. Para positivo geramos um programa motor atualizado com as novas informações e descartando os possíveis traços considerados como rasuras.

Veja maiores detalhes nos códigos 3 e 4:

Algoritmo 3 Pontuação de similaridade

```

1: procedure SCORELIKELIHOOD(sTest, sTrain, lib)
    ▷ pegando pontuações parciais para cada programa motor
2:   [scoreTrain, ouTrain] ← scoreMP(sTrain, lib)
3:   [scoreTest, ouTest] ← scoreMP(sTest, lib)
    ▷ variáveis de características globais
4:   fTest ← sumfields(ouTest, false)
    ▷  $P(I^{(T)}|A^{(T)}, \sigma_b^{(T)}, \epsilon^{(T)})$ 
5:   fTrain ← sumfields(ouTrain, false)
    ▷  $P(I^{(c)}|A^{(c)}, \sigma_b^{(c)}, \epsilon^{(c)})$ 
6:   fGlobal ← fTest − fTrain
    ▷  $P(I_g^{(T)}|I_g^{(c)})$ 
7:   maxS ← max(sTest.ns, sTrain.ns)
8:   pScore ← compare_strokes(sTest.motor, sTrain.motor)
    ▷ custo min. local
9:   motorT ← MotorProgram(maxS)
    ▷ inicializando programa motor
    ▷ setando parâmetros básicos para os programas motores de teste
10:  motorT.I ← sTest.I
11:  motorT.parameters ← sTest.parameters
12:  motorT.epsilon ← sTest.epsilon
13:  motorT.blur_sigma ← sTest.blur_sigma
14:  if sTest.ns ≥ sTrain.ns then
15:    for i = 1...sTrain.ns do
    ▷ mapeando todos os traços válidos
16:      motorT.S{pScore(i, 1)} ← sTest.S{pScore(i, 1)}
17:    end for
18:  else
19:    for i = 1...sTest.ns do
20:      motorT.Si ← sTest.Si
21:    end for
22:  end if
23:  return modStroke ← motorT
    ▷ atualizando programas motores
24:  fLocal ← mean(pScore(:, 2))
    ▷  $P(S_1, \dots, S_k)$ 
25:  return likeness ← |fGlobal + fLocal|
    ▷  $likeness = P(I_g^{(T)}|I_g^{(c)}) + P(S_1, \dots, S_k)$ 
26: end procedure
  
```

A.4 Função para o cálculo de pontuação preditiva

Função que define o valor de hipótese a posteriori entre a pontuação geral dentre todos os ajustes das imagens de teste com as imagens de treinamento: $P(I^{(T)}|I^{(c)})$. A equação E12 é aplicada a partir

Algoritmo 4 Comparar pares de traços (teste e treinamento) para extrair características locais

```

1: procedure COMPARE_STROKES(testMotor, trainMotor)
2:    $n_{test} \leftarrow \text{length}(\text{testMotor})$  ▷ comprimento dos motores de teste
3:    $n_{train} \leftarrow \text{length}(\text{trainMotor})$  ▷ comprimento dos motores de treino
4:    $pScore \leftarrow \text{zeros}(n_{train}, 2)$  ▷ matriz de melhores custos
5:    $bPath \leftarrow \text{cell}(n_{train}, 2)$  ▷ célula para melhores caminhos
6:   for  $i = 1 \dots n_{train}$  do
7:      $stkScore \leftarrow \text{zeros}(1, n_{test})$  ▷ inicializando vetor de pontuação parcial dos traços
8:      $costDir \leftarrow \text{zeros}(1, 2)$  ▷ array para custo de direção entre dois traços
9:      $pCostDir \leftarrow \text{zeros}(n_{test}, 1)$ 
10:     $pPath \leftarrow \text{cell}(n_{test}, 1)$ 
11:     $STrain \leftarrow \text{concatSubStrokes}(\text{trainMotor}_i)$  ▷ concat. sub-traços se houver
12:     $STinv \leftarrow \text{flipud}(STrain)$ 
13:    for  $j = 1 \dots n_{test}$  do
14:       $STest \leftarrow \text{concatSubStrokes}(\text{testMotor}_j)$ 
15:      for  $ns = 1 \dots \min[\text{length}(STest), \text{length}(STrain)]$  do
16:         $costDir(1) \leftarrow costDir(1) + d(STest_{ns}, Strain_{ns})$ 
17:         $costDir(2) \leftarrow costDir(2) + d(STest_{ns}, STinv_{ns})$ 
18:      end for
19:      if  $costDir(1) < costDir(2)$  then ▷ inverte traços se direções opostas
20:         $dtw \leftarrow \text{dtwDistance2D}(STest, STrain)$ 
21:         $pCostDir(j) \leftarrow 0$ 
22:      else
23:         $dtw \leftarrow \text{dtwDistance2D}(STest, STinv)$ 
24:         $pCostDir(j) \leftarrow 1$ 
25:      end if
26:       $s \leftarrow \text{length}(dtw(:, 1))$ 
27:       $t \leftarrow \text{length}(dtw(1, :))$ 
28:       $tp \leftarrow 1$ 
29:       $cost(end + 1) \leftarrow dtw(s, t)$  ▷ procura pelos melhores caminhos e custos usando DTW
30:       $path_{tp} \leftarrow [s; t]$ 
31:      while  $s > 1$  and  $t > 1$  do
32:         $aux \leftarrow [dtw(s - 1, t); dtw(s, t - 1); dtw(s - 1, t - 1)]$ 
33:         $idx \leftarrow [s - 1t; st - 1; s - 1t - 1]$ 
34:         $[v, id] \leftarrow \min(aux)$ 
35:         $s \leftarrow idx(id, 1)$ 
36:         $t \leftarrow idx(id, 2)$ 
37:         $tp \leftarrow tp + 1$ 
38:         $cost(end + 1) \leftarrow v$ 
39:         $path_{tp} \leftarrow [s; t]$ 
40:      end while
41:       $pPath_j \leftarrow path$ 
42:       $stkScore(j) \leftarrow \text{sum}(cost)$ 
43:    end for
44:     $[minS, idBS] \leftarrow \min(stkScore)$ 
45:     $pScore_i \leftarrow [idBS; minS]$ 
46:     $bPath_i \leftarrow [pPath(idBS); pCostDir(idBS)]$ 
47:  end for
48:  return  $bPath$  ▷ Todos os caminhos ótimos entre os traços teste e treino
49:  return  $pScore$  ▷ Todos os custos mínimos:  $minCost_k = \sum_{i=1}^l path_i$ 
50: end procedure

```

da linha 41 do algoritmo 2.

Algoritmo 5 Log a posteriori da pontuação preditiva - $\log P(I^{(T)}|I^{(c)})$

```

1: procedure LOG-POST-PRED(pair)
2:   likeness  $\leftarrow$  pair.likeness                                 $\triangleright$  pontuação de similaridade
3:   logfit  $\leftarrow$  pair.fit_score                                $\triangleright$  pontuação de ajuste inicial
4:   return prior_scores  $\leftarrow$  pair.prior_score                 $\triangleright$  pontuação de reajuste
5:   logwt  $\leftarrow$  prior_scores - logsumexp(prior_scores(:))       $\triangleright$  normalização dos pesos
6:   logwt2  $\leftarrow$  likeness - logsumexp(likeness(:))
7:   logv  $\leftarrow$  logfit(:) + logwt + logwt2                     $\triangleright$  combinação dos pesos com o termo de ajuste
8:   return logscore  $\leftarrow$  logsumexp(logv)  $\triangleright$  log a posteriori considerando o fator de similaridade
9: end procedure

```

Apêndice B

Algoritmo DTW (*Dynamic Time Warping*)

Neste apêndice explicaremos o passo a passo do funcionamento do algoritmo de deformação dinâmica entre duas séries temporais, e ao final do capítulo será exposto o código-fonte desta computação para curvas em 2D. Vamos iniciar considerando duas séries temporais como exemplo: $S_1 = [1, 2, 8, 4, 6, 7]$ e $S_2 = [1, 3, 6, 5, 8, 10, 11, 16, 18, 20]$ | $|S_1| = m$ e $|S_2| = n$. Cada valor existente dentro desses dois vetores serão os índices bases que determinam o comprimento da linha e da coluna da matriz de pesos. A matriz de pesos abaixo segue a estrutura de inicialização otimizada apresentado por [SB16]:

$$DTW = \begin{matrix} & \begin{matrix} 1 & 3 & 6 & 5 & 8 & 10 & 11 & 16 & 18 & 20 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 8 \\ 4 \\ 6 \\ 7 \end{matrix} & \begin{pmatrix} 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \infty & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \infty & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \infty & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \infty & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{matrix}$$

Propriedades da inicialização:

$$DTW(i, j) = \begin{cases} \infty, & \text{se } i = 0 \text{ ou } j = 0 \\ 0, & \text{se } i = j = 0 \end{cases} \quad (\text{AB1})$$

Logo, começamos a percorrer a matriz a partir da posição (2,2) (número 0 em verde) definindo o custo para cada posição. Para isso temos de verificar qual o menor custo dentre as três posições anteriores mais próximas ao número atual. As posições verificadas anteriormente a posição atual são, junto com a definição de seu custo segue a distribuição de pesos:

$$DTW(i, j) = d(S_i^1, S_j^2) + \min \begin{cases} DTW(i-1, j) \\ DTW(i, j-1) \\ DTW(i-1, j-1) \end{cases} \quad (\text{AB2})$$

onde d é uma função que calcula a distância euclidiana entre os dois pontos de S_1 e S_2 .

Depois de definido a inicialização da matriz e a estrutura de computação ao longo de suas posições, iremos percorrer mostrando o passo a passo de cada iteração, sabendo que a posição mínima desta matriz é com índices $i = 2$ e $j = 2$:

1. $DTW(2, 2) = |2 - 3| + \min[\infty, \infty, 0] = 1$
2. $DTW(2, 3) = |2 - 6| + \min[\infty, 1, \infty] = 5$
3. $DTW(2, 4) = |2 - 5| + \min[\infty, 5, \infty] = 8$
4. $DTW(2, 5) = |2 - 8| + \min[\infty, 8, \infty] = 14$
5. $DTW(2, 6) = |2 - 10| + \min[\infty, 14, \infty] = 22$
6. $DTW(2, 7) = |2 - 11| + \min[\infty, 22, \infty] = 31$
7. $DTW(2, 8) = |2 - 16| + \min[\infty, 31, \infty] = 45$
8. $DTW(2, 9) = |2 - 18| + \min[\infty, 45, \infty] = 61$
9. $DTW(2, 10) = |2 - 20| + \min[\infty, 61, \infty] = 79$
10. $DTW(3, 2) = |8 - 3| + \min[1, \infty, \infty] = 6$
11. $DTW(3, 3) = |8 - 6| + \min[5, 6, 1] = 3$
12. $DTW(3, 4) = |8 - 5| + \min[8, 3, 5] = 6$
13. $DTW(3, 5) = |8 - 8| + \min[14, 6, 8] = 6$
14. $DTW(3, 6) = |8 - 10| + \min[22, 6, 14] = 8$
15. $DTW(3, 7) = |8 - 11| + \min[31, 8, 22] = 11$
16. $DTW(3, 8) = |8 - 16| + \min[45, 11, 31] = 19$
17. $DTW(3, 9) = |8 - 18| + \min[61, 19, 45] = 29$
18. $DTW(3, 10) = |8 - 20| + \min[79, 29, 61] = 41$
19. $DTW(4, 2) = |4 - 3| + \min[6, \infty, \infty] = 7$
20. $DTW(4, 3) = |4 - 6| + \min[3, 7, 6] = 5$
21. $DTW(4, 4) = |4 - 5| + \min[6, 5, 3] = 4$
22. $DTW(4, 5) = |4 - 8| + \min[6, 4, 6] = 8$
23. $DTW(4, 6) = |4 - 10| + \min[8, 8, 6] = 12$
24. $DTW(4, 7) = |4 - 11| + \min[11, 12, 8] = 15$
25. $DTW(4, 8) = |4 - 16| + \min[19, 15, 11] = 23$
26. $DTW(4, 9) = |4 - 18| + \min[29, 23, 19] = 33$

27. $DTW(4, 10) = |4 - 20| + \min[41, 33, 29] = 45$
28. $DTW(5, 2) = |6 - 3| + \min[7, \infty, \infty] = 10$
29. $DTW(5, 3) = |6 - 6| + \min[5, 10, 7] = 5$
30. $DTW(5, 4) = |6 - 5| + \min[4, 5, 5] = 5$
31. $DTW(5, 5) = |6 - 8| + \min[8, 5, 4] = 6$
32. $DTW(5, 6) = |6 - 10| + \min[12, 6, 8] = 10$
33. $DTW(5, 7) = |6 - 11| + \min[15, 10, 12] = 15$
34. $DTW(5, 8) = |6 - 16| + \min[23, 15, 15] = 25$
35. $DTW(5, 9) = |6 - 18| + \min[33, 25, 23] = 35$
36. $DTW(5, 10) = |6 - 20| + \min[45, 35, 33] = 47$
37. $DTW(6, 2) = |7 - 3| + \min[10, \infty, \infty] = 14$
38. $DTW(6, 3) = |7 - 6| + \min[5, 14, 10] = 6$
39. $DTW(6, 4) = |7 - 5| + \min[5, 6, 5] = 7$
40. $DTW(6, 5) = |7 - 8| + \min[6, 7, 5] = 6$
41. $DTW(6, 6) = |7 - 10| + \min[10, 6, 6] = 9$
42. $DTW(6, 7) = |7 - 11| + \min[15, 9, 10] = 13$
43. $DTW(6, 8) = |7 - 16| + \min[25, 13, 15] = 22$
44. $DTW(6, 9) = |7 - 18| + \min[35, 22, 25] = 33$
45. $DTW(6, 10) = |7 - 20| + \min[47, 33, 35] = 46$

Nossa matriz resultante será:

$$DTW = \begin{matrix} & \begin{matrix} 1 & 3 & 6 & 5 & 8 & 10 & 11 & 16 & 18 & 20 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 8 \\ 4 \\ 6 \\ 7 \end{matrix} & \begin{pmatrix} 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 1 & 5 & 8 & 14 & 22 & 31 & 45 & 61 & 79 \\ \infty & 6 & 3 & 6 & 6 & 8 & 11 & 19 & 29 & 41 \\ \infty & 7 & 5 & 4 & 8 & 12 & 15 & 23 & 33 & 45 \\ \infty & 10 & 5 & 5 & 6 & 10 & 15 & 25 & 35 & 47 \\ \infty & 14 & 6 & 7 & 6 & 9 & 13 & 22 & 33 & 46 \end{pmatrix} \end{matrix}$$

Após esta computação, note que a partir da última posição desta matriz, podemos realizar uma busca gulosa, percorrendo pelos pares de índices que contenham os menores custos possíveis nessa matriz, ou seja, decompondo os menores valores que respeitem a distribuição AB2. Desta forma, nosso percurso trará o caminho ótimo entre as ligações dos pontos das séries S_1 a S_2 (destacado em vermelho):

$$DTW = \begin{matrix} & \begin{matrix} 1 & 3 & 6 & 5 & 8 & 10 & 11 & 16 & 18 & 20 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 8 \\ 4 \\ 6 \\ 7 \end{matrix} & \begin{pmatrix} \textcolor{red}{0} & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \textcolor{red}{1} & 5 & 8 & 14 & 22 & 31 & 45 & 61 & 79 \\ \infty & 6 & \textcolor{red}{3} & 6 & 6 & 8 & 11 & 19 & 29 & 41 \\ \infty & 7 & 5 & \textcolor{red}{4} & 8 & 12 & 15 & 23 & 33 & 45 \\ \infty & 10 & 5 & \textcolor{red}{5} & 6 & 10 & 15 & 25 & 35 & 47 \\ \infty & 14 & 6 & 7 & \textcolor{red}{6} & \textcolor{red}{9} & \textcolor{red}{13} & \textcolor{red}{22} & \textcolor{red}{33} & \textcolor{red}{46} \end{pmatrix} \end{matrix}$$

O somatório do vetor com os menores custos $path = [46, 33, 22, 13, 9, 6, 5, 4, 3, 1, 0]$ será o custo mínimo ótimo entre as respectivas séries, onde:

$$path = [p_1, p_2, p_3, \dots, p_l] \mid n \leq l \leq n + m, \quad p = (i, j)$$

$$minCost = \sum_1^l path_i$$

lembrando que n é o comprimento do vetor S_2 e m é o comprimento do vetor S_1 .

Para curvas em duas dimensões, realizamos todo o processo de computação em uma dimensão, ou seja, atualizamos os valores do DTW para o eixo X e outra para o eixo Y. Ao final somamos essas duas matrizes, tendo como resultado ambos o melhor caminho possível que as liga e o menor custo associado entre elas [SYHJ⁺17], como já foi apresentado em 4.3.1. Código Fonte:

Algoritmo 6 DTW-2D

```

1: procedure DTWDISTANCE2D( $c1, c2$ )
2:    $xDTW \leftarrow length(0, 0)$  ▷ matriz de zeros para o eixo X
3:    $yDTW \leftarrow length(0, 0)$  ▷ matriz de zeros para o eixo Y
4:   for  $i=1 \dots c1$  do ▷ Inicialização da matriz para o eixo X
5:      $xDTW(i, 1) \leftarrow \infty$ 
6:      $yDTW(i, 1) \leftarrow \infty$ 
7:   end for
8:   for  $i=1 \dots c2$  do ▷ Inicialização da matriz para o eixo Y
9:      $xDTW(1, i) \leftarrow \infty$ 
10:     $yDTW(1, i) \leftarrow \infty$ 
11:  end for
12:   $xDTW(1, 1) \leftarrow 0$ 
13:   $yDTW(1, 1) \leftarrow 0$ 
14:  for  $i=2 \dots c1$  do
15:    for  $j=2 \dots c2$  do
16:       $costX \leftarrow |(c1(i, 1) - c2(j, 1))|$  ▷ distância entre os pontos para o eixo X
17:       $costY \leftarrow |(c1(i, 2) - c2(j, 2))|$  ▷ distância entre os pontos para o eixo Y
18:       $auxX \leftarrow \{xDTW(i-1, j); xDTW(i, j-1); xDTW(i-1, j-1)\}$ 
19:       $auxY \leftarrow \{yDTW(i-1, j); yDTW(i, j-1); yDTW(i-1, j-1)\}$ 
20:       $xDTW(i, j) \leftarrow costX + \min(auxX)$ 
21:       $yDTW(i, j) \leftarrow costY + \min(auxY)$ 
22:    end for
23:  return  $DTW \leftarrow xDTW + yDTW$  ▷ matriz de custo mínimo em 2D
24: end procedure

```

Referências Bibliográficas

- [AMMIL12] Yaser S Abu-Mostafa, Malik Magdon-Ismail e Hsuan-Tien Lin. *Learning from data*, volume 4. AMLBook New York, NY, USA:, 2012. [19](#)
- [B⁺09] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009. [7](#)
- [BGL⁺94] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger e Roopak Shah. Signature verification using a “siamese” time delay neural network. Em *Advances in Neural Information Processing Systems*, páginas 737–744, 1994. [7](#)
- [BNJ09] Jyotirmoy Banerjee, Anoop M Namboodiri e CV Jawahar. Contextual restoration of severely degraded document images. Em *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, páginas 517–524. IEEE, 2009. [6](#)
- [CA17] Bidyut B Chaudhuri e Chandranath Adak. An approach for detecting and cleaning of struck-out handwritten text. *Pattern Recognition*, 61:282–294, 2017. [iii](#), [v](#), [3](#), [4](#), [5](#), [6](#), [32](#), [34](#), [41](#), [47](#)
- [CG95] Siddhartha Chib e Edward Greenberg. Understanding the metropolis-hastings algorithm. *The american statistician*, 49(4):327–335, 1995. [33](#), [37](#)
- [FF⁺03] Li Fe-Fei et al. A bayesian approach to unsupervised one-shot learning of object categories. Em *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, páginas 1134–1141. IEEE, 2003. [6](#), [51](#)
- [Fla99] John H Flavell. Cognitive development: Children’s knowledge about the mind. *Annual review of psychology*, 50(1):21–45, 1999. [2](#)
- [KMRW14] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende e Max Welling. Semi-supervised learning with deep generative models. Em *Advances in Neural Information Processing Systems*, páginas 3581–3589, 2014. [iii](#), [v](#), [1](#)
- [KPT07] Charles Kemp, Amy Perfors e Joshua B Tenenbaum. Learning overhypotheses with hierarchical bayesian models. *Developmental science*, 10(3):307–321, 2007. [17](#), [18](#)
- [KSH12] Alex Krizhevsky, Ilya Sutskever e Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. Em *Advances in neural information processing systems*, páginas 1097–1105, 2012. [7](#)

- [KZS15] Gregory Koch, Richard Zemel e Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. Em *ICML Deep Learning Workshop*, volume 2, 2015. [xiii](#), [7](#), [51](#)
- [LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio e Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. [2](#)
- [LBH15] Yann LeCun, Yoshua Bengio e Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. [1](#)
- [LH90] Chia-Wei Liao e Jun S Huang. Stroke segmentation by bernstein-bezier curve fitting. *Pattern Recognition*, 23(5):475–484, 1990. [34](#)
- [Li11] Xin Li. Image recovery via hybrid sparse representations: A deterministic annealing approach. *IEEE Journal of Selected Topics in Signal Processing*, 5(5):953–962, 2011. [4](#)
- [LST15] Brenden M Lake, Ruslan Salakhutdinov e Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. [iii](#), [v](#), [xiv](#), [xvii](#), [3](#), [5](#), [7](#), [22](#), [23](#), [24](#), [25](#), [33](#), [34](#), [35](#), [37](#), [40](#), [44](#), [45](#), [46](#), [48](#), [51](#), [53](#)
- [Mar93] Gary F Marcus. Negative evidence in language acquisition. *Cognition*, 46(1):53–85, 1993. [17](#)
- [MCM13] Ryszard S Michalski, Jaime G Carbonell e Tom M Mitchell. *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013. [21](#)
- [MS15] Steve Marschner e Peter Shirley. *Fundamentals of computer graphics*. CRC Press, 2015. [27](#), [28](#), [30](#)
- [Nei14] Ulric Neisser. *Cognitive psychology: Classic edition*. Psychology Press, 2014. [iii](#), [v](#), [13](#), [51](#)
- [NPH06] Stéphane Nicolas, Thierry Paquet e Laurent Heutte. Markov random field models to extract the layout of complex handwritten documents. Em *Tenth International Workshop on Frontiers in Handwriting Recognition*. Suvisoft, 2006. [5](#), [6](#)
- [PTGX11] Amy Perfors, Joshua B Tenenbaum, Thomas L Griffiths e Fei Xu. A tutorial introduction to bayesian models of cognitive development. *Cognition*, 120(3):302–321, 2011. [14](#), [15](#), [16](#), [17](#), [19](#), [20](#), [40](#)
- [PTR06] Amy Perfors, Joshua Tenenbaum e Terry Regier. Poverty of the stimulus? a rational approach. Em *Annual Conference of the Cognitive Science Society*. Cognitive Science Society, 2006. [21](#)
- [RD16] Sujith Ravi e Qiming Diao. Large scale distributed semi-supervised learning using streaming approximation. Em *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016. [iii](#), [v](#), [1](#)

- [Ris78] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978. [21](#)
- [SB16] Diego F Silva e Gustavo EAPA Batista. Speeding up all-pairwise dynamic time warping matrix calculation. Em *Proceedings of the 2016 SIAM International Conference on Data Mining*, páginas 837–845. SIAM, 2016. [iii](#), [v](#), [38](#), [59](#)
- [SFH17] Sara Sabour, Nicholas Frosst e Geoffrey E Hinton. Dynamic routing between capsules. Em *Advances in neural information processing systems*, páginas 3856–3866, 2017. [6](#)
- [Sri13] Nitish Srivastava. Improving neural networks with dropout. *University of Toronto*, 182:566, 2013. [7](#)
- [STG04] David M Sobel, Joshua B Tenenbaum e Alison Gopnik. Children’s causal inferences from indirect evidence: Backwards blocking and bayesian reasoning in preschoolers. *Cognitive science*, 28(3):303–333, 2004. [2](#)
- [SYHJ⁺17] Mohammad Shokoohi-Yekta, Bing Hu, Hongxia Jin, Jun Wang e Eamonn Keogh. Generalizing dtw to the multi-dimensional case requires an adaptive approach. *Data mining and knowledge discovery*, 31(1):1–31, 2017. [38](#), [62](#)
- [SZ14] Karen Simonyan e Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. [7](#)
- [TP15] M. Antónia Amaral Turkman e Carlos Daniel Paulino. *Estatística Bayesiana Computacional - uma introdução*. Sociedade Portuguesa de Estatística - SPE, 2015. [12](#)