ORGANIZAÇÃO E PROCESSAMENTO DE INFORMAÇÃO NO CÓRTEX A PARTIR DE AGRUPAMENTOS SUCESSIVOS DE PADRÕES

Luciano Dyballa

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Valmir Carneiro Barbosa

Rio de Janeiro
Julho de 2015

ORGANIZAÇÃO E PROCESSAMENTO DE INFORMAÇÃO NO CÓRTEX A PARTIR DE AGRUPAMENTOS SUCESSIVOS DE PADRÕES

Luciano Dyballa

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Valmir Carneiro Barbosa, Ph.D.

Prof. Felipe Maia Galvão França, Ph.D.

Prof. Luís Alfredo Vidal de Carvalho, D.Sc.

Prof.ª Roseli Suzi Wedemann, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
JULHO DE 2015

Dyballa, Luciano

Organização e processamento de informação no córtex a partir de agrupamentos sucessivos de padrões/Luciano Dyballa. – Rio de Janeiro: UFRJ/COPPE, 2015.

IX, 98 p. 29, 7cm.

Orientador: Valmir Carneiro Barbosa

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2015.

Referências Bibliográficas: p. 89 – 95.

1. redes corticais. 2. aprendizado de padrões. 3. agrupamento. I. Barbosa, Valmir Carneiro. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Para Luiza.

Agradecimentos

Agradeço à minha família, por todo o carinho e por acreditar em mim e me apoiar sempre que preciso, assim como aos meus amigos, indispensáveis. Vocês todos estão sempre em meus pensamentos.

Agradeço ao meu orientador Valmir, essencial para o desenvolvimento deste trabalho, não só pelo aprendizado incomensurável que me proporcionou, mas também pela grande amizade. Nossas reuniões semanais vão deixar muita saudade.

Agradeço à professora Celina Figueiredo, que tive o privilégio de conhecer logo na primeira semana de aulas e que me acompanhou ao longo de quase todos os períodos, sendo fundamental para meu desenvolvimento acadêmico.

Agradeço também a todos os colegas, professores e funcionários com os quais convivi nesses pouco mais de dois anos na COPPE, e à CAPES pelo auxílio financeiro.

E, sem me esquecer dos tempos de Escola de Química, faço aqui um agradecimento aos professores Daniel Barreto e Frederico Tavares, por me fazerem acreditar em meu potencial como pesquisador.

Por fim, agradeço ao meu avô Paulo, que despertou em mim o gosto pela matemática.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ORGANIZAÇÃO E PROCESSAMENTO DE INFORMAÇÃO NO CÓRTEX A PARTIR DE AGRUPAMENTOS SUCESSIVOS DE PADRÕES

Luciano Dyballa

Julho/2015

Orientador: Valmir Carneiro Barbosa

Programa: Engenharia de Sistemas e Computação

O presente trabalho parte de uma série de observações para, com base nelas, apresentar uma proposta de um panorama geral da organização do córtex dos mamíferos, bem como alguns dos possíveis mecanismos utilizados no processamento de informação sensorial realizado no córtex. Primeiramente, é feita a análise de um conjunto de dados de conectividade recentemente publicado para o córtex de macacos, revelando aspectos inéditos acerca da comunicação entre áreas corticais. Além disso, é feito um agrupamento dessa rede em comunidades de arestas que permite um melhor entendimento dos fluxos principais de informação ocorrendo no córtex. Em seguida, um modelo inspirado no processamento sensorial do cortéx é proposto para o aprendizado *online* de padrões a partir do agrupamento progressivo de símbolos presentes em sequências como texto e música simbólica. Esse modelo é aplicado à tarefa de codificação, dando origem a um compressor universal responsável pelas melhores taxas de compressão de texto já alcançadas até hoje. Uma segunda aplicação do modelo é apresentada que permite o agrupamento não-supervisionado de livros e músicas. Ambas as aplicações fazem uso de estruturas para organização dos padrões descobertos que sugerem maneiras com as quais a memória poderia ser armazenada e acessada no cérebro. Por fim, observamos que todos os nossos resultados relacionam-se, direta ou indiretamente, à tarefa de agrupamento, ou *clustering*—ao longo do texto, agrupamos áreas corticais, símbolos e padrões, com cada tipo de agrupamento fornecendo sugestões úteis a respeito do funcionamento do córtex—, o que parece apontar tal tarefa como um fenômeno central no processamento e organização corticais. Essa ideia é discutida ao final do trabalho e algumas conjecturas são feitas a respeito da própria evolução do córtex como tendo sido orientada pela tarefa de agrupamento.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

ORGANIZATION AND INFORMATION PROCESSING IN THE CORTEX BY MEANS OF SUCCESSIVE CLUSTERING OF PATTERNS

Luciano Dyballa

July/2015

Advisor: Valmir Carneiro Barbosa

Department: Systems Engineering and Computer Science

This work presents a series of observations and derives from them a proposal for a general outlook of the organization taking place in the mammalian cortex, as well as some of the possible mechanisms used in the cortical processing of sensory information. First, we present an analysis of a recently published connectivity dataset for the macaque cortex, revealing new aspects of interareal communication. We also show that a clustering of this cortical network's links into communities allows for a better understanding of the major flows of information taking place in the cortex. Next, a model inspired in the cortical sensory processing is proposed for the online learning of patterns by means of the progressive clustering of symbols from sequences such as text and symbolic music. This model is applied to the task of encoding, originating a universal compressor that achieves the highest rates of text compression ever reported. A second application for the model is its use in the unsupervised clustering of books and music. Both these applications make use of structures for the organization of the discovered patterns that suggest ways in which memory could actually be stored and accessed in the brain. Lastly, we observe that all our results are related, directly or indirectly, to the task of clustering—throughout the text, we perform the clustering of cortical areas, symbols and patterns, with each one of these bringing to the fore useful observations concerning cortical functioning—, which seems to indicate that this could be a central phenomenon in cortical processing and organization. This idea is discussed at the end, and some conjectures are made with respect to the evolution of the cortex as having been oriented by clustering.

Sumário

Capítulo 1

Introdução

O cérebro vem despertando a curiosidade humana há milênios como poucos outros assuntos o fizeram. Na antiguidade, o enigma maior era sobre que tipo de função um órgão tão peculiar poderia desempenhar. De alguns séculos para cá, quando tornou-se claro seu papel central no comando de todas as nossas atividades conscientes e insconscientes, passamos a nos perguntar: como é possível que uma massa cinzenta pesando pouco mais que um quilo seja responsável por raciocínio, reflexos e emoções tão complexos? Entretanto, essa aparente simplicidade desaparece assim que descemos à escala celular e nos damos conta da imensidão gerada por bilhões de neurônios conectados através de trilhões de sinapses. Como fazer sentido de tamanha complexidade?

Nos últimos anos, uma série de descobertas tem sido desencadeada a partir de inovações tecnológicas que permitem um acompanhamento da atividade cerebral com precisão e resolução inéditas [1]. A informação gerada por essas técnicas vem tornando possível não apenas diagnósticos médicos mais adequados, mas também um melhor entendimento do funcionamento do sistema nervoso, o que por sua vez inspira a descoberta de modelos teóricos para tentar desvendar os princípios básicos por trás de suas capacidades. Tal tarefa demanda não somente um entendimento fisiológico de o que levou e leva as conexões entre neurônios a se organizarem de uma forma ou de outra, mas também um entendimento do ponto de vista computacional, no sentido de como é possível que estruturas tão básicas consigam processar informação real e aprender com ela, modificando a si próprias para se tornarem mais úteis. Neste trabalho, vamos tentar seguir ambas as vertentes com o intuito de revelar alguns dos mecanismos que tornam isso possível.

Começamos mostrando, no Capítulo 2, que um conjunto de dados recentemente publicado possibilitou desvendar algumas regularidades no padrão global de comunicação entre as diversas áreas do córtex de macacos do gênero *Macaca*. O uso de técnicas para a divisão de redes complexas em módulos, ou comunidades, reforçou a ideia de que redes corticais são altamente modulares. É possível imaginar, conforme

será discutido, que cada uma dessas comunidades no córtex tenha se originado a partir de uma única área, em geral áreas de processamento sensorial primário. Portanto, entender que tipo de computação é realizada nessas áreas centrais seria fundamental para se pensar num modelo que dê conta das propriedades macroscópicas exibidas pela rede cortical completa.

Um processador de sinais de entrada, nos moldes do que se sabe sobre o córtex visual, deve, em princípio, ser capaz não apenas de reconhecer padrões, mas também de aprendê-los automaticamente, e à medida que os recebe. Cada sentido exige um aparato adequado às particularidades do sinal que é recebido (fótons na visão, ondas sonoras na audição, pressão no tato, composição química no olfato e paladar, por exemplo), mas em essência suas tarefas são as mesmas. Vamos propor, no Capítulo 3, um algoritmo baseado em um modelo distribuído de computação que cumpra esses requisitos e que seja versátil o suficiente para que, a partir de um mesmo princípio de funcionamento, consiga aprender padrões relevantes em quaisquer domínios de informação.

Nosso modelo vai sugerir que, para aprender conceitos ou padrões relevantes, naturalmente é formada uma hierarquia de processamento, e que padrões mais complexos vão exigir parâmetros distintos dos de padrões mais elementares. Isso é altamente compatível com o fato de que temos áreas distintas para níveis diferentes de processamento dentro de uma mesma função sensorial (vide córtices visual e auditivo, por exemplo), onde áreas diferentes implicam não só diferentes distribuições de neurônios em cada camada, mas até mesmo o tipo de neurônios empregados. Conforme será visto, o aprendizado de padrões é baseado em agrupamentos sucessivos em diversos níveis de complexidade, o que, por sua vez, acarretaria naturalmente no agrupamento da rede dos padrões gerados em comunidades.

O trabalho prossegue, nos capítulos 4 e 5, com aplicações do algoritmo de aprendizado de padrões. Veremos que uma pergunta natural que advém da disponibilidade desses padrões aprendidos é: como usá-los para montar descrições de alto nível das sequências de sinais de entrada que chegam ao nosso córtex? No Capítulo 4, mostramos as implicações de uma memória organizada em caminhos direcionados, e a utilizamos para criar um compressor universal de arquivos. No Capítulo 5, adotamos uma organização da memória em agrupamentos sucessivos que permite que identifiquemos livros e compositores agrupando, respectivamente, texto e música de maneira não-supervisionada.

Por fim, tentamos conciliar o que foi descoberto a partir da rede biológica do Capítulo 2 com o que foi aprendido nos Capítulos 3 a 5 para especular sobre um modelo global de organização e evolução do córtex dos mamíferos. Veremos que a ideia de agrupamento permeia todas as etapas, partindo da pré-existência de padrões no ambiente ao redor de um organismo até a deteção desses padrões em seu cérebro, o

que por sua vez faria com que o córtex se desenvolvesse modularmente, e que abstrações de alto nível também fossem assimiladas através do agrupamento dos padrões aprendidos (estes próprios também resultantes de agrupamento de símbolos recebidos sensorialmente). Essa organização seria diretamente responsável pelo agrupamento físico do córtex (conexões entre neurônios), formando áreas e, em um nível mais alto, dando origem às comunidades observadas nas redes corticais.

Capítulo 2

Conectividade e fluxo de informação em uma rede cortical

2.1 Introdução

Nos últimos anos, temos presenciado o uso cada vez maior de ferramentas da área de "redes complexas", ou *network science*, com a finalidade de buscar sentido para a estonteante complexidade do cérebro e de descobrir alguns dos mecanismos responsáveis por sua organização [2, 3]. A visão do sistema nervoso central como sendo formado por redes em diversas escalas foi amplamente adotada na neurociência, despertando o interesse de neurocientistas por métodos grafo-teóricos para caracterizar os padrões de conectividade estrutural e funcional entre regiões do cérebro [4]. Diversos laboratórios vêm sendo, com isso, estimulados a mapear as redes do sistema nervoso central de humanos [5], macacos [6], gatos [7] e camundongos [8], disponibilizando bancos de dados cada vez mais precisos e completos.

Redes estruturais do tipo "inter-áreas", ou de "mesoescala", representam as interações entre diferentes regiões do cérebro. Cada nó na rede representa uma área definida através de sua citoarquitetura e cada aresta corresponde a uma conexão entre duas áreas. Essa conexão, entretanto, pode representar diferentes tipos de relações entre as áreas: a depender do método de detecção utilizado, podemos ter dados de conectividade "estrutural", "funcional" ou "efetiva" [4]. No primeiro tipo, as arestas representam conexões anatômicas entre as áreas, promovidas por feixes de axônios, detectadas com a injeção de marcadores de neurônios. No segundo, temos as dependências estatísticas (não-direcionadas) inferidas através da correlação entre as atividades das diferentes áreas durante a execução de tarefas cognitivas, as quais podem ser obtidas, por exemplo, por ressonância magnética funcional do tipo *BOLD* (que mede os níveis de oxigenação das células de cada área) [9]. Já a conectividade efetiva é definida como a influência que uma população neuronal exerce sobre

outra sob um determinado modelo de dinâmicas causais, cujos parâmetros são em geral estimados a partir de dados obtidos por eletroencefalograma, magnetoencefalograma ou *BOLD* [9]. Cada um desses tipos de aresta dá origem a redes que nos fornecem informações distintas, porém complementares. Os dados de conectividade destes três tipos permitem que investiguemos propriedades dos processos de alto nível ocorrendo no córtex, tais como eficiência de comunicação [10], integração de informação [11, 12], organização modular [13], robustez perante lesões [14] e o efeito causado por doenças na conectividade cerebral [15].

Um estudo recente [16] usou uma técnica de rastreamento anatômico quantitativo para mapear a conectividade estrutural entre áreas do córtex de macacos do gênero *Macaca* com uma riqueza de detalhes sem precedentes. Em contraste com outros *datasets* amplamente estudados [17, 18], os novos dados incluem não apenas a direção, mas também o número de neurônios envolvidos em cada conexão, bem como a distribuição da origem laminar para muitas delas [19]. Arestas com esse tipo adicional de informação podem permitir que levemos em conta diferenças importantes entre cada conexão e que, com isso, obtenhamos um melhor entendimento da rede [20]. Esse *dataset* também é particularmente notável devido ao alto grau de confiança, já que todos os experimentos foram conduzidos pelo mesmo grupo e, portanto, estiveram sujeitos aos mesmos critérios e validações estatísticas [16].

Neste capítulo, iremos explorar esse território ainda consideravelmente desconhecido da rede cortical dos macacos. O conjunto de dados acima citado revelou uma rede muito mais densa do que previamente reportado, desafiando a visão tradicional de que redes corticais de mamíferos exibem uma arquitetura tipo *small-word* [21] e, consequentemente, desafiando cientistas a buscar novas perspectivas de análise e visualização desses dados. Por trás dessa maior densidade está a descoberta de um grande número de conexões até então não detectadas, das quais a maioria reside entre as mais fracas do córtex (isto é, formadas por poucas projeções axonais). O fato de termos uma presença tão grande de conexões fracas no córtex imediatamente suscita questões importantes, relativas, por exemplo, ao papel que elas possam desempenhar na atividade cortical e, mais fundamentalmente, à influência da força de conexões na comunicação cortical.[1]

[1]Muitas das observações apresentadas neste trabalho baseiam-se no pressuposto de que conexões físicas implicam na transmissão de sinal de fato entre as diversas áreas, e de que algumas relações funcionais podem ser inferidas a partir delas. Embora esse ponto de vista seja amplamente utilizado em estudos que utilizam dados de conectividade estrutural, muitas vezes ele pode ser enganoso, visto que uma conexão anatômica pode estar presente mas na prática não ser utilizada, e que áreas do córtex podem se relacionar por diversos outros meios que não apenas através de sinal elétrico transmitido por axônios.

2.2 Características dos dados utilizados e definições

Em [16], foi usado rastreamento anatômico de trato axonal com marcadores retrógrados para mapear as conexões entre áreas no córtex de macacos. Nesse tipo de procedimento, o marcador é injetado em uma dada área-alvo e subsequentemente é transportado por difusão através dos axônios que terminam naquela área, percorrendo o caminho de volta até os neurônios nos quais aqueles axônios se originam. A localização desses neurônios é comparada e correspondida a áreas corticais conhecidas. As áreas marcadas dessa forma são então incluídas nos dados como vizinhos de entrada (*in-neighbors*) da área injetada. Esse experimento, em particular, consistiu em injeções repetidas em 29 áreas corticais distribuídas pelos quatro lobos de um mesmo hemisfério do córtex de macacos do gênero *Macaca*, de um total de 91 áreas. Para mais detalhes acerca dos métodos, ver [22]; para mais detalhes sobre a demarcação das áreas do córtex utilizada, ver [23].

É importante observar que, embora só tenhamos 29 áreas injetadas, os dados incluem as conexões que chegam até essas áreas vindas de todas as 91 áreas. Como consequência, temos virtualmente todas as conexões existentes entre as 29 injetadas, o que nos fornece uma matriz de adjacências de tamanho 29×29. A rede resultante contém 536 arestas direcionadas. Podemos também considerar a matriz de adjacências incompleta de tamanho 91×29, que inclui todas as conexões detectadas nos experimentos.

A seguir, vamos apresentar a terminologia necessária para caracterizar cada projeção. O *dataset* completo e maiores informações podem ser obtidos em http://www.core-nets.org.

O número de neurônios (NN) de uma dada conexão da área B para a área A corresponde ao número de neurônios marcados em B após injeção de marcador em A (veja Figura 2.1. O valor de NN usado para essa conexão é a média geométrica dos valores para todos os espécimes estudados.

A quantidade usada em [16] como o peso de uma aresta de B para A é a fração de neurônios marcados (FLN, do inglês, *fraction of labeled neurons*) em uma área B relativa a todos os neurônios que foram marcados após injeção de marcador em A:

$$\mathrm{FLN}_{\mathrm{B}\to\mathrm{A}} = \frac{\mathrm{NN}_{\mathrm{B}\to\mathrm{A}}}{\sum_{\alpha} \mathrm{NN}_{\alpha\to\mathrm{A}}},$$

onde α é uma área qualquer dentre as 91 possíveis. O FLN é, portanto, uma versão normalizada do NN, e é útil porque ajuda a avaliar a contribuição relativa de cada conexão para a área que está recebendo, independentemente do volume ou densidade celular dessa área.

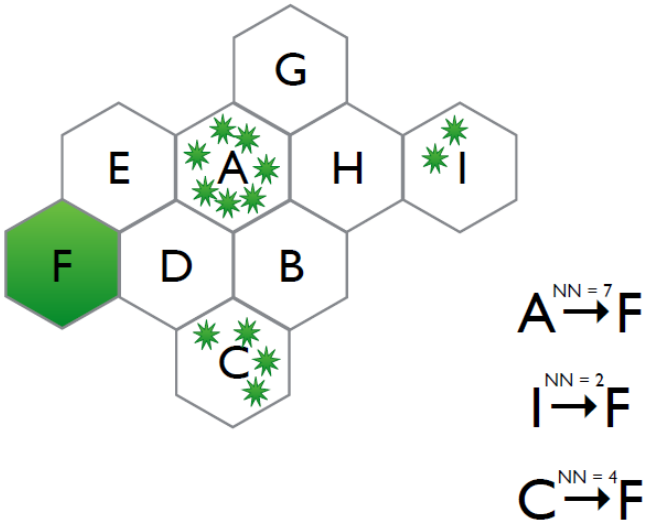


Figura 2.1: Representação esquemática de um córtex onde cada hexágono equivale a uma área cortical. Após injeção de marcador em F, cada neurônio marcado em outras áreas é ilustrado como uma estrela. Os valores de NN para as três conexões correspondentes são exemplificados à direita.

Os dados também incluem aproximações para as distâncias axonais (físicas) entre áreas. Vamos nos referir a esses valores como "comprimentos de conexão", reservando o termo "distância" para a medida grafo-teórica. As abreviações dos nomes das áreas seguem aquelas listadas por [22].

2.3 Hierarquia cortical

2.3.1 Distribuição de pesos por área

Devido à maneira como os pesos FLN foram definidos, os pesos de entrada para qualquer uma das 29 áreas injetadas somam 1 quando consideramos as conexões de entrada vindas de todas as 91 áreas. A Tabela 2.1 mostra a distribuição de FLN para as dez conexões de mais altos FLN em quatro áreas: V1, 2, 10 e 9/46v. Um aspecto em comum entre elas é que todas possuem poucas conexões com FLN relativamente alto e um grande número de conexões com FLN muito baixo (digamos, menor que 1%). Curiosamente, esse mesmo padrão é visto para todas as 29 áreas, o que significa que, por exemplo, não há nenhuma área tendo seu total de NN distribuído uniformemente entre seus vizinhos de entrada.

Mas, a despeito dessa similaridade, existem algumas diferenças acentuadas entre as distribuições dos pesos das diferentes áreas (ver Figura 2.2). Algumas, como V1, têm uma ou duas conexões de entrada com FLN muito alto, seguidas de diversas outras com muito baixo FLN, enquanto outras, como 7A, parecem ter uma variação muito menos acentuada de FLN entre seus vizinhos de entrada. O fato de V1 ser uma

Tabela 2.1: Dez maiores valores de FLN das conexões de entrada para as áreas V1, 2, 10 e 9/46v. A linha inferior contém as médias desses valores.

V1	**2**	**10**	**9/46v**
0.7477	0.3798	0.2271	0.3005
0.1223	0.2400	0.1994	0.0971
0.0594	0.1423	0.1204	0.0816
0.0240	0.1108	0.0904	0.0602
0.0074	0.0424	0.0708	0.0570
0.0067	0.0300	0.0374	0.0385
0.0056	0.0104	0.0368	0.0347
0.0044	0.0093	0.0307	0.0226
0.0036	0.0078	0.0269	0.0223
0.0031	0.0076	0.0265	0.0203
Média: 0.0984	Média: 0.0980	Média: 0.0866	Média: 0.0735

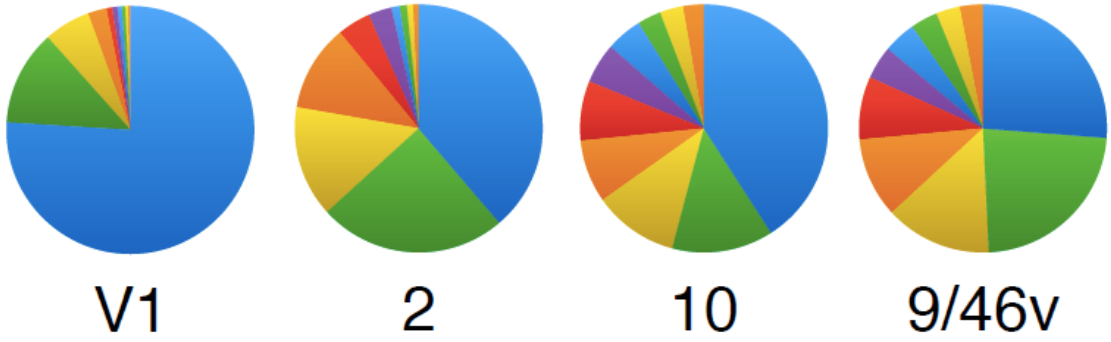


Figura 2.2: Distribuição dos dez maiores valores de FLN das conexões de entrada para as áreas V1, 2, 10 e 9/46v.

área inferior a 7A em termos de uma hierarquia de processamento de informação no córtex parece indicar que a distribuição dos pesos de cada área nos diz algo sobre sua posição hierárquica.

2.3.2 Distância da periferia sensorial

Para investigar se a distribuição de FLN de uma área pode nos dizer algo sobre a posição daquela área na hierarquia cortical, vamos definir a "distância da periferia sensorial", DPS, de uma dada área A como sendo a menor distância direcionada até ela partindo-se das áreas de entrada sensorial no córtex—V1 (córtex visual primário); 1, 2 e 3 (córtex primário somatossensorial); Gu (córtex primário gustativo); ENTO e PIRI (córtex primário olfativo); e Core (córtex primário auditivo). Valores de DPS, portanto, referem-se à matriz de adjacências 91×29.

A cada aresta designamos um comprimento igual ao inverso de seu peso FLN (o que, na prática, significa que quanto maior é o FLN de uma aresta menor a distância entre os dois nós que ela conecta). Logo, a distância direcionada de B para A é o

comprimento total do caminho direcionado de B para A cujo comprimento total seja mínimo. O valor de DPS para todas as áreas de entrada sensorial é, portanto, 0.

Como exemplo, o valor de DPS para V2 é 1.30, uma vez que essa área tem um caminho consistindo numa única aresta de comprimento $1/0.77 = 1.30$ partindo de V1 e não existe nenhum caminho mais curto partindo de nenhuma das demais áreas de entrada sensorial. Outro exemplo é o valor para a área 10, que é 44.09 porque o caminho mais curto até ela desde qualquer uma das áreas primárias sensoriais é Core $\rightarrow$ PBr $\rightarrow$ STPr $\rightarrow$ 10, de comprimento total 44.09. Note que essa é a distância mais curta mesmo havendo um caminho que usa apenas uma aresta de Core até 10, cujo comprimento é 4106.00 (devido ao FLN muito baixo dessa aresta, que é da ordem de 10^{-4}). Quanto maior o DPS de uma área, maior ela está na hierarquia cortical.

Outras métricas para distância hierárquica foram propostas [6, 24], levando-se em consideração as distribuições laminares na origem e terminação de cada conexão, que não estão disponíveis no conjunto de dados utilizado. Não obstante, a Tabela 2.2 mostra que nosso método ordena as 29 áreas aproximadamente de acordo com outras ordenações reportadas para o córtex de macacos [25, 26], indicando que temos uma medida razoável, ainda que aproximada.

2.3.3 DPS *vs.* FLN médio

Quando plotamos o DPS de cada área *versus* seu FLN médio (Figura 2.3(a)), descobrimos que as duas quantidades são fortemente correlacionadas negativamente (coeficiente de correlação de Pearson $r = -0.60$, p-valor $= 5 \times 10^{-4}$). Isso sugere que, olhando-se para a maneira como as forças das conexões são distribuídas entre os vizinhos de entrada de uma dada área, pode ser possível dizer se ela processa informação de nível alto ou baixo. Curiosamente, quando tomamos a média das dez conexões com mais alto FLN para cada área (Figura 2.3(b)), a linearidade é ainda maior ($r = -0.78$, p-valor $= 5 \times 10^{-7}$).

Note que, já que os valores de FLN somam 1, sua média é, na verdade, uma função somente do número de conexões que uma área possui. Quando consideramos apenas as dez conexões de maior peso, porém, a distribuição dos pesos assume um papel central na correlação.

2.4 Comunidades de arestas

A estrutura de comunidades, ou organização modular, do córtex dos mamíferos foi amplamente estudada [13]. Ela é supostamente responsável por promover segregação funcional através de um alto grau de interação entre áreas compartilhando papéis

Tabela 2.2: Valores de DPS para as 29 áreas injetadas.

Área	**DPS**
9/46v	60.33
7m	45.66
10	44.09
46d	39.83
7A	39.58
9/46d	38.18
8l	36.07
24c	34.47
8m	33.67
7B	32.22
8B	31.02
F7	24.40
STPc	17.75
STPr	17.33
STPi	15.36
TEpd	14.93
F2	14.11
PBr	12.18
5	10.94
F1	7.97
TEO	7.66
DP	6.72
MT	6.38
ProM	6.21
F5	4.92
V4	3.76
V2	1.30
V1	0.00
2	0.00

funcionais similares [27], formando módulos ou comunidades. Estes, por sua vez, facilitam uma integração global da rede através da comunicação entre os nós mais centrais (*hubs*) pertencentes às diferentes comunidades [12, 28].

A maioria dos estudos de redes do cérebro feitos até hoje usou técnicas tradicionais para detecção de comunidades de nós, que particionam a rede em módulos ou comunidades com altas densidades de arestas dentro de um mesmo módulo e, ao mesmo tempo, baixas densidades de arestas entre módulos. A grande desvantagem dessa abordagem é que as comunidades resultantes são disjuntas, ou seja, cada nó pertence a uma única comunidade. Resultados anteriores encontraram comunidades de nós altamente relacionadas às posições físicas de suas áreas constituintes [29], com cada comunidade correspondendo aproximadamente à região cortical onde suas áreas estão localizadas. No entanto, essas comunidades não revelam muito mais do

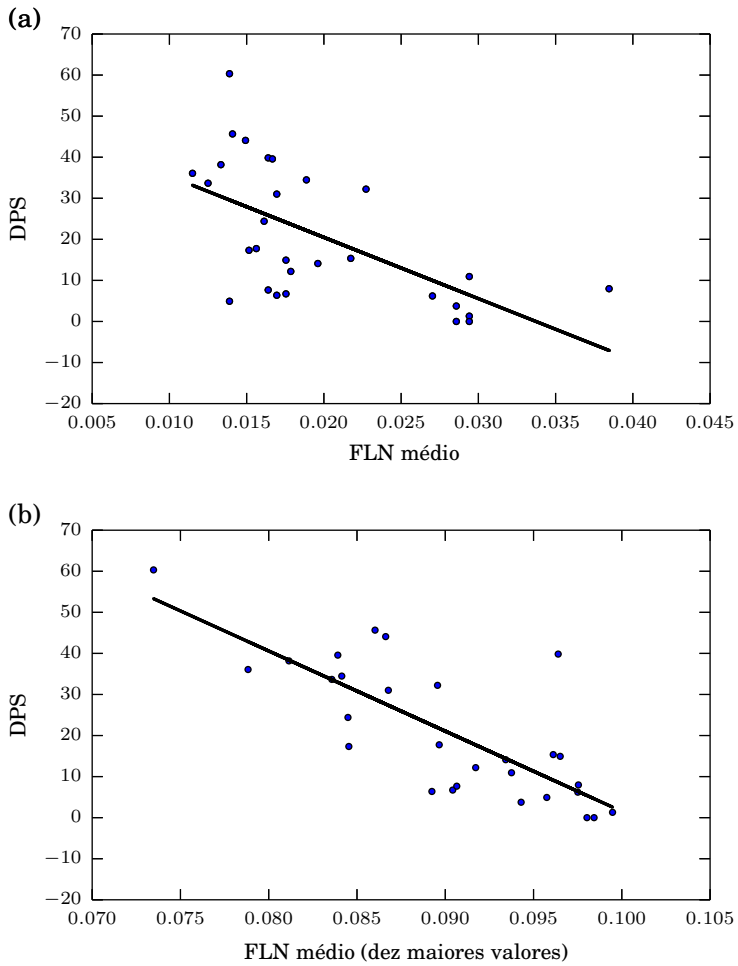


Figura 2.3: DPS *vs.* FLN médio de todas as conexões de entrada (a), e DPS *vs.* FLN médio levando-se em conta apenas os dez maiores valores de FLN (b). Cada ponto corresponde a uma das 29 áreas injetadas.

que o que é logicamente esperado, já que áreas fisicamente próximas costumam compartilhar funções similares. Além disso, uma vez que conexões mais curtas tendem a possuir valores mais altos de FLN [16], é provável que métodos para detecção de comunidades baseados em maximização de modularidade acabem por simplesmente agrupar áreas próximas umas das outras no córtex de macacos em uma mesma comunidade.

Na verdade, gostaríamos de uma divisão em comunidades que nos fornecesse informações úteis a respeito de similaridade funcional, independentemente das forças das conexões. Entretanto, os pesos não devem ser ignorados, já que não podemos ignorar o fato de que uma conexão fraca de longa distância é muito diferente de uma conexão curta e forte. Além disso, dado o caráter altamente integrador do córtex, parece natural imaginar que cada módulo não esteja isolado, e provavelmente deva possuir um ou mais nós responsáveis pela troca de informação com outros módulos com funções distintas. Para tanto, a detecção de comunidades de arestas apresentada

por [30] parece ser uma maneira natural de incorporar essa perspectiva em nossa análise, e mais: ao agrupar arestas ao invés de nós, esperamos capturar comunidades mais significativas que não apenas nos digam que nós são mais relacionados entre si, mas também que nos apontem a natureza dessas relações, conforme indicado pela direções das arestas em cada comunidade.

Para conseguir isso, utilizamos o método descrito por [30], que consiste em calcular uma medida de similaridade baseada em interseção de vizinhança para todos os pares de arestas possuindo um nó em comum. Um algoritmo de agrupamento hierárquico, normalmente usando *single-linkage*,[2] é subsequentemente aplicado para a construção de um dendrograma, cuja seção de máximo valor para uma heurística denominada "densidade de partição" é aquela que fornece as comunidades de melhor qualidade. Nesse estudo, fizemos uma pequena adaptação na fórmula de similaridade de modo a melhor incorporar pesos e direções e permitir conexões recíprocas (ver Apêndice A para detalhes).

A aplicação do agrupamento hierárquico, considerando-se todas as arestas dadas pela matriz 29×29, resulta, entretanto, em uma distribuição de comunidades de baixa qualidade, com um único módulo que inclui todos os nós. Os resultados mudam dramaticamente, porém, se filtrarmos as arestas mais fracas. Fizemos uma busca pelo valor de corte (*threshold*) que resultasse no máximo número de comunidades, e esse valor foi 0.0003616 (Figura 2.4). Isso significa que, ao usarmos somente arestas com FLN ≥ 0.0003616 (em outras palavras, descartando-se todas as arestas B $\rightarrow$ A cujo NN é menor que 0.03616% do NN total sendo projetado até A), descobrimos uma partição das arestas em 24 comunidades.

Antes de detalhar essa partição em 24 comunidades de arestas, queremos enfatizar que o filtro das arestas mais fracas é um recurso crucial para permitir a extração de um conjunto razoável dos dados utilizados. Vemos na Figura 2.4 que, se nenhuma conexão for excluída, nenhuma estrutura em comunidades será revelada; logo, algum tipo de filtro é necessário. Porém, que critério utilizar para a escolha de um valor de corte? Se olharmos com mais atenção os resultados do método utilizado, veremos que uma de suas características é que cada comunidade produzida, apesar de poder incluir os mesmos nós que outra, é única em termos das arestas que contém. Portanto, recorremos à intuição de que, se temos um número maior de comunidades, em princípio teremos mais informação. No nosso caso, isso seria particularmente sugestivo, já que existe um único valor de FLN para o qual o número de comunidades

[2]Neste agrupamento, partimos de uma divisão inicial em que temos uma comunidade para cada aresta individualmente. Em seguida, tomamos sucessivamente cada par de arestas em ordem decrescente de similaridade e fundimos as duas comunidades a que pertencem as duas arestas. Tal processo pode ser representado por um dendrograma cujas folhas são as comunidades iniciais (uma para cada aresta), as quais vão sendo fundidas em diferentes níveis de similaridade. O resultado final é uma árvore cuja raiz é a comunidade única obtida após todos os pares de arestas terem sido considerados.

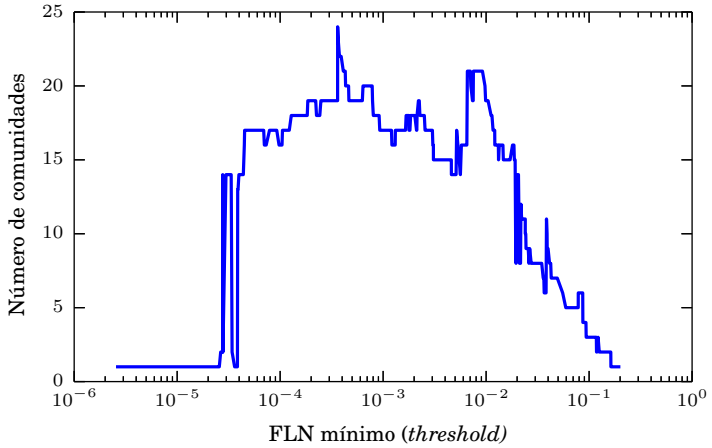


Figura 2.4: Número de comunidades de arestas encontradas para diferentes *thresholds* de FLN. Um máximo de 24 comunidades é encontrado para um *threshold* de 0.0003616.

obtidas é máximo (24). As comunidades encontrados são mostradas na Figura 2.5.

As primeiras quatro comunidades se destacam por serem altamente "clusterizadas", cada qual contendo quase todas as arestas possíveis entre seus nós. Juntas, elas cobrem todas as 29 áreas da rede sem qualquer repetição, exatamente como uma partição dos nós. Notavelmente, essas quatro comunidades se assemelham às que foram anteriormente reportadas usando-se detecção de comunidades de nós [29]—nos nossos resultados, temos uma comunidade a menos e a área 7m pertence a uma comunidade diferente. Esse fato não apenas serve como base para a validação de nossos resultados,[3] mas também indica que comunidades de arestas nos fornecem uma maior riqueza de informações, já que obtivemos aproximadamente os mesmos módulos obtidos para comunidades de nós e ainda outros além desses. Mais que isso, essas quatro comunidades nos dão uma maneira mais fácil de olhar para as demais 20, uma vez que estas últimas representam grupos de arestas responsáveis pelas interações que ocorrem entre as primeiras quatro.

Por exemplo, de acordo com a numeração adotada na Figura 2.5, a comunidade 7 nos diz que a área 8l é uma importante integradora da informação vinda das áreas da comunidade 2. As comunidades 5 e 6 indicam que a área 7A funciona como uma mediadora entre as áreas visuais da comunidade 2 e as áreas parietais na comunidade 3, já que atua tanto como integradora de informação visual (tal como a área 8l na comunidade 7, mas recebendo projeções de um conjunto de áreas um

[3] A robustez desses resultados também foi verificada usando-se uma variação da metodologia proposta por [31]. A variação foi feita de modo a ser compatível com redes direcionadas e com pesos, e utiliza a informação mútua normalizada [32] ao invés da variância de informação do método original. Os resultados indicaram que a estrutura de comunidades de uma versão aleatória da rede é muito mais afetada por perturbações do que a estrutura de comunidades encontradas para a rede real (dados não mostrados).

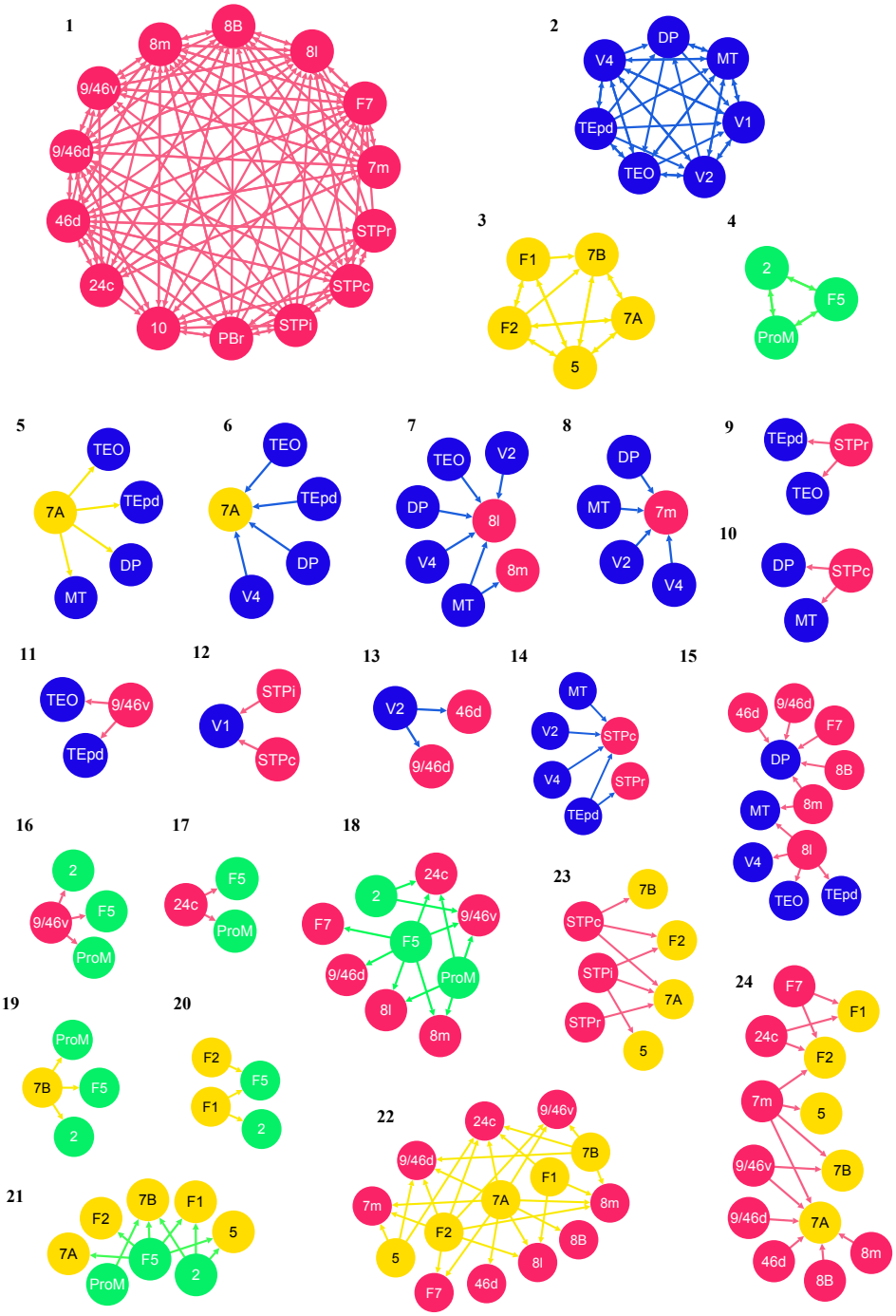


Figura 2.5: Comunidades de arestas encontradas usando-se o método descrito no Apêndice A. As cores utilizadas nos nós das comunidades 5–24 se referem às cores usadas nas primeiras quatro comunidades. As arestas estão coloridas de acordo com o nó em que elas se originam.

pouco diferente) quanto como disseminadora de informação que influencia as áreas visuais, provavelmente na forma de uma resposta tipo *feedback*, já que as áreas na comunidade 3 estão em um nível mais alto na hierarquia cortical (ver Tabela 2.2). O mesmo tipo de análise pode ser conduzido para todas as demais comunidades.

Usando as primeiras quatro comunidades (1–4) como referência, podemos resumir os principais fluxos de informação no córtex de macacos da seguinte forma: as comunidades 1 e 3 tanto enviam quanto recebem sinais das comunidades 2 e 4. As comunidades 1 e 3 também trocam sinal entre elas próprias, em ambas as direções. No entanto, não há qualquer comunidade representando interações entre as comunidades 2 e 4, o que contribui para uma visão desses dois subconjuntos de áreas como periféricos em um cenário global de processamento cortical (note também como os nós nessas duas comunidades possuem todos eles baixos valores de DPS (Tabela 2.2). Em contraste, as comunidades 1 e 3 parecem mediar a troca de informação e promover a integração através de todo o córtex. A posição elevada de suas áreas constituintes na Tabela 2.2 indica que elas o fazem por meio de processamento de informação de mais alto nível. Interessante é que esse cenário é bastante similar à estrutura de *bow-tie* sugerida por [21], cujo "núcleo central" corresponde quase que exatamente à união das comunidades 1 e 3.

2.5 Conexões recíprocas

Duas conexões são ditas recíprocas se elas envolvem o mesmo par de áreas mas possuem direções opostas. A maioria ($\sim 80\%$, totalizando 214 pares) das conexões entre áreas no *dataset* é recíproca [16], algo que tem sido consistentemente observado também em outros conjuntos de dados [7, 8, 18].

A ideia de fluxos de informação em contracorrente, ou de ciclos fechados de *feedback*, é considerada há muito tempo um fenômeno que naturalmente deve ocorrer em qualquer sistema que exiba grande capacidade de autorregulação [33]. Isso pode ocorrer por meio de ciclos direcionados na rede cortical, e o menor tipo de ciclo possível é aquele formado por conexões recíprocas, que permitem uma resposta de *feedback* extremamente rápida (se considerarmos o número de passos necessários para se percorrer o ciclo). Logo, seria de esperar que um sistema de alta capacidade autorregulatória e eficiente como o córtex dos mamíferos exiba o máximo possível de conectividade recíproca. Nós investigamos algumas das características pertencentes a essa classe de conexões para o córtex de macacos.

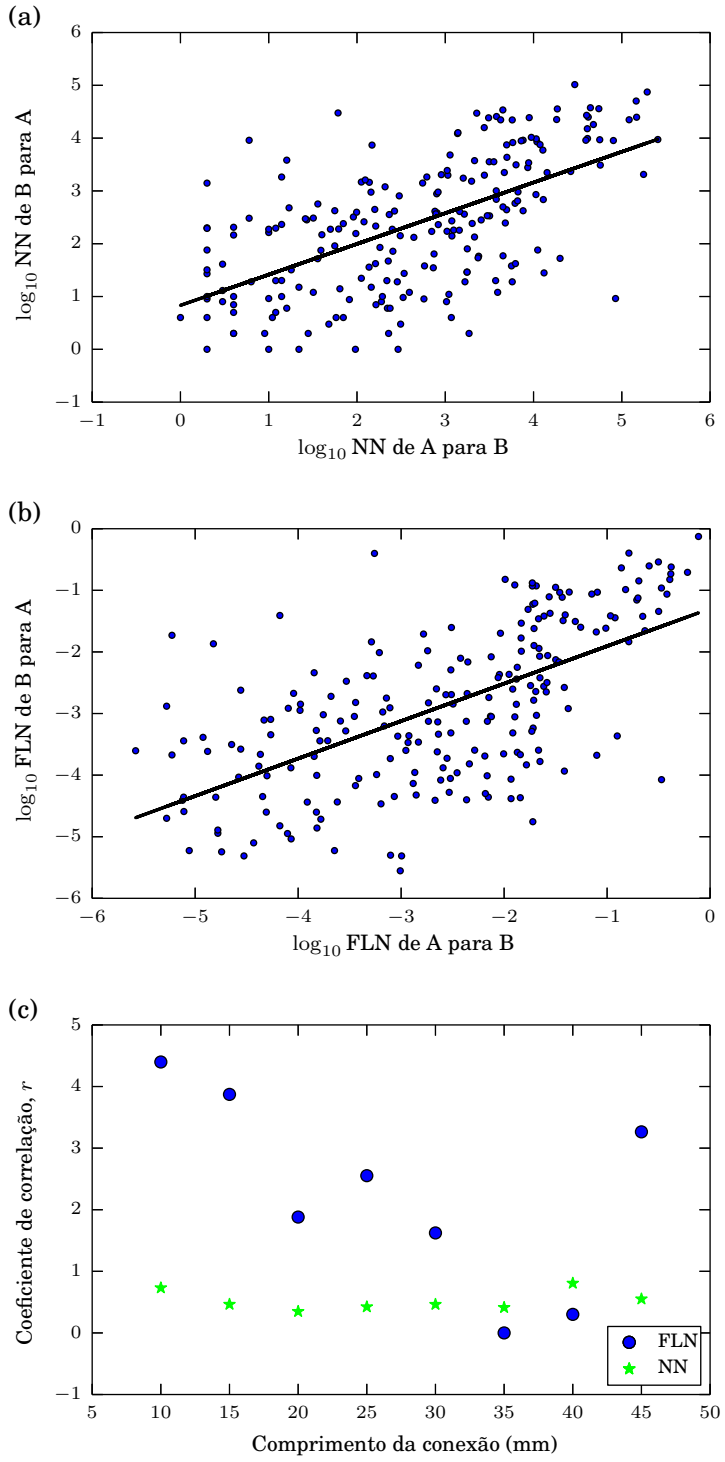


Figura 2.6: Correlação entre o $\log_{10}$ NN de duas conexões recíprocas (a) e entre seus $\log_{10}$ FLN (b). Nos painéis (a) e (b), cada ponto corresponde a um dos 214 pares de conexões recíprocas. Os coeficientes de correlação (r) são mostrados para oito intervalos de comprimentos de conexão (c), cada qual representado por seu valor mais à direita (um intervalo adicional, [45,50), compreende apenas dois pares de conexões recíprocas e foi por essa razão omitido). Os p-valores são todos da ordem de 10^{-2} a 10^{-6}, com exceção do intervalo [40,45), onde temos p-valores mais altos: 0.16 para $\log_{10}$ NN e 0.15 para $\log_{10}$ FLN.

2.5.1 NN e FLN

Descobrimos que as conexões recíprocas exibem uma forte correlação entre seus valores tanto de NN quanto de FLN, em cada direção. A Figura 2.6(a) mostra o resultado de plotarmos, para todos os 214 pares de conexões recíprocas, $\log_{10}$ NN em uma direção *vs.* $\log_{10}$ NN na outra direção. O mesmo gráfico é encontrado na Figura 2.6(b), mas usando-se FLN ao invés de NN. Para decidir qual das direções usar como abscissa nos gráficos, para cada par de áreas escolhemos a direção da área inferior para a superior na hierarquia cortical da Tabela 2.2. (Por exemplo: o par (V1,V2) possui conexões recíprocas e V1 é inferior na hierarquia (menor DPS), portanto a abscissa do par se refere a V1 $\rightarrow$ V2 e sua ordenada a V2 $\rightarrow$ V1.) Correlações lineares foram encontradas tanto para os logaritmos de NN ($r = 0.59$, p-valor $= 4 \times 10^{-21}$) quanto para os logaritmos de FLN ($r = 0.68$, p-valor $= 6 \times 10^{-30}$).

Também verificamos se essas correlações seriam válidas através de diferentes intervalos de comprimentos de conexão entre as áreas. A Figura 2.6(c) mostra que os coeficientes de correlação são altos tanto para conexões recíprocas longas quanto para conexões recíprocas curtas.

Note que o fato de os valores de NN em conexões recíprocas serem correlacionados não implica automaticamente que seus valores de FLN também sejam correlacionados. Uma vez que o FLN de uma dada conexão reflete a contribuição relativa daquela conexão para sua área-alvo, e que conexões recíprocas têm alvos diferentes, seria perfeitamente possível que duas conexões recíprocas com valores similares de NN contribuíssem de forma bastante diferente para suas áreas-alvo, e portanto acarretar valores bastante distintos de FLN.

Por causa da tendência geral de que os valores de NN têm de diminuir quanto maior é o comprimento da conexão entre as áreas [10], poderíamos supor que, uma vez que ambas as conexões de cada par recíproco têm aproximadamente o mesmo comprimento, seria de esperar que elas devessem ter seus valores de NN correlacionados. De fato, há uma correlação negativa entre $\log_{10}$ NN e comprimento de conexão ($r = -0.48$, p-valor $= 8 \times 10^{-95}$, gráfico não mostrado). Entretanto, se eliminarmos o efeito do comprimento de conexão computando a correlação parcial entre os valores de $\log_{10}$ NN em direções opostas, ainda assim uma correlação significativa é obtida ($r = 0.46$, p-valor $= 1 \times 10^{-12}$). Isso sugere que o comprimento de conexão, embora influencie até certo ponto os valores de NN, não é inteiramente determinante. O mesmo raciocínio é válido para os valores de FLN. Vamos retornar a esse ponto na Seção 2.6.

As Figuras 2.6(a),(b) não apenas mostram que existe uma relação linear entre os logaritmos de valores de NN e FLN em direções opostas, mas, o que é mais importante, é que suas retas de tendência sugerem que os próprios valores devem

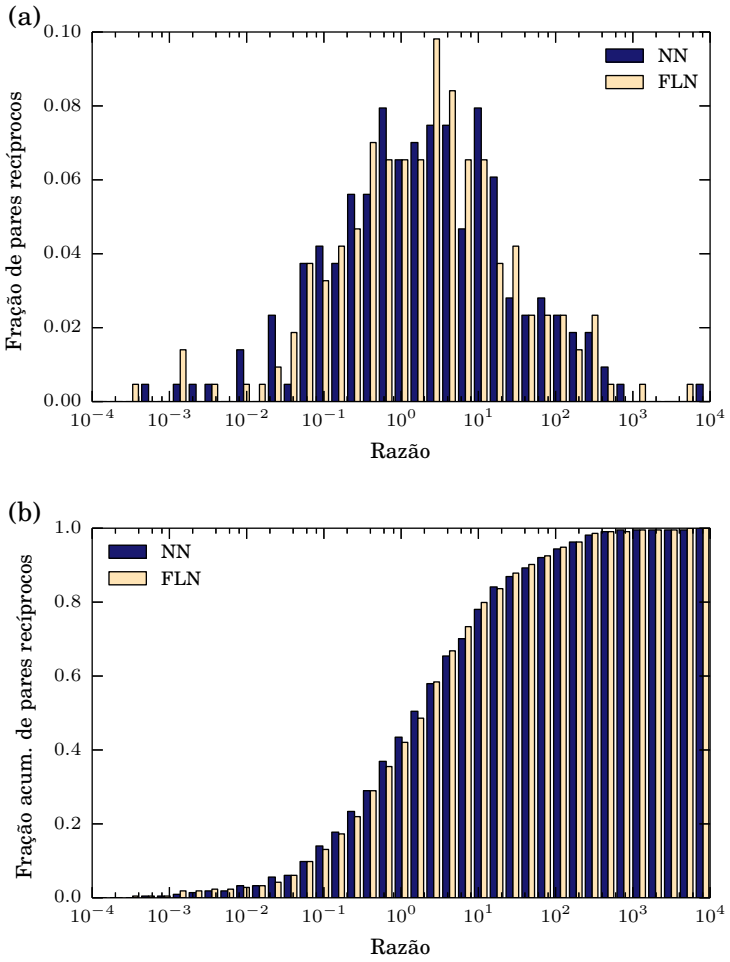


Figura 2.7: Distribuição (a) e distribuição acumulada (b) das razões de NN e de FLN para pares recíprocos de conexões. Cada razão é obtida dividindo-se o valor da conexão que vai da área inferior em direção à área superior na hierarquia da Tabela 2.2 pelo valor correspondente na direção oposta. Os dados usam *log-binning* de base 1.6.

ter ordens de magnitude similares. Para avaliar isso, computamos, para cada par de conexões recíprocas, a razão entre o valor de NN na direção da área inferior para a superior na hierarquia da Tabela 2.2 e o valor na direção oposta. O mesmo foi calculado para os valores de FLN.

As Figuras 2.7(a),(b) mostram a distribuição e a distribuição acumulada, respectivamente, das razões dos valores de NN e de FLN para os 214 pares recíprocos (as médias das razões de NN e FLN são, respectivamente, 69.3 e 44.7). Dois fatos importantes podem ser observados: o primeiro é que a maioria das conexões recíprocas têm valores de NN ou FLN com ordens de magnitude não muito diferentes em ambas as direções, diferindo apenas por um fator de menos de 100 (um tanto notável, se considerarmos que os valores de NN e FLN compreendem cinco ordens de magnitude). O segundo é que a distribuição de razões é deslocada para

a direita—conexões de áreas inferiores para superiores na hierarquia cortical geralmente empregam mais neurônios do que as conexões na direção oposta. De forma interessante, se descartarmos as conexões mais fracas usando o mesmo valor de corte da Seção 2.4, as razões médias de NN e FLN passam a ser muito mais próximas de 1 (8.1 e 6.6, respectivamente).

2.5.2 Conexões não-recíprocas

Os resultados expostos acima suscitam a questão de quais seriam as características das conexões que não possuem uma equivalente recíproca, isto é, que existem apenas de A para B mas não de B para A. Descobrimos que elas compreendem em sua maioria conexões com valores relativamente baixos de FLN. Isso é ilustrado na Figura 2.8(a), que compara conexões recíprocas e não-recíprocas com respeito às suas distribuições de FLN. Isso parece, a princípio, sugerir que talvez as conexões mais fracas não sejam muito relevantes, e por isso não requeiram reciprocidade. No entanto, uma fração considerável das conexões recíprocas constitui-se de conexões fracas (50% das conexões recíprocas estão entre as 25% mais fracas, considerando-se a matriz de adjacências 29×29), o que nos impede de fazer essa generalização.

Uma segunda justificativa para não-reciprocidade que poderíamos imaginar é que essas conexões talvez sejam em sua maioria de longa distância, o que implicaria um alto custo energético. Assim, a ausência de uma contraparte recíproca poderia representar uma economia de energia. Mas esse também não é o caso, rigorosamente, pois, como pode ser visto na Figura 2.8(b), existe uma porção considerável de conexões de curta distância não-recíprocas.

Outra possibilidade é que conexões não-recíprocas envolvem tão poucos neurônios que eles não puderam ser todos detectados nos experimentos conduzidos, o que também não parece muito provável dada a consistência dos resultados reportados por [16].

Uma outra justificativa para sua existência seria, talvez, a de executarem uma função fundamentalmente distinta, de forma a não requerer o tipo de troca de sinal em duas direções observado para as demais. De fato, já foi sugerido que algumas delas podem estar envolvidas em acesso direto tipo *top-down* à memória [34], e portanto a investigação de tipo aparentemente especial de conexão poderia se beneficiar de análises futuras que levem em consideração as identidades das áreas corticais envolvidas.

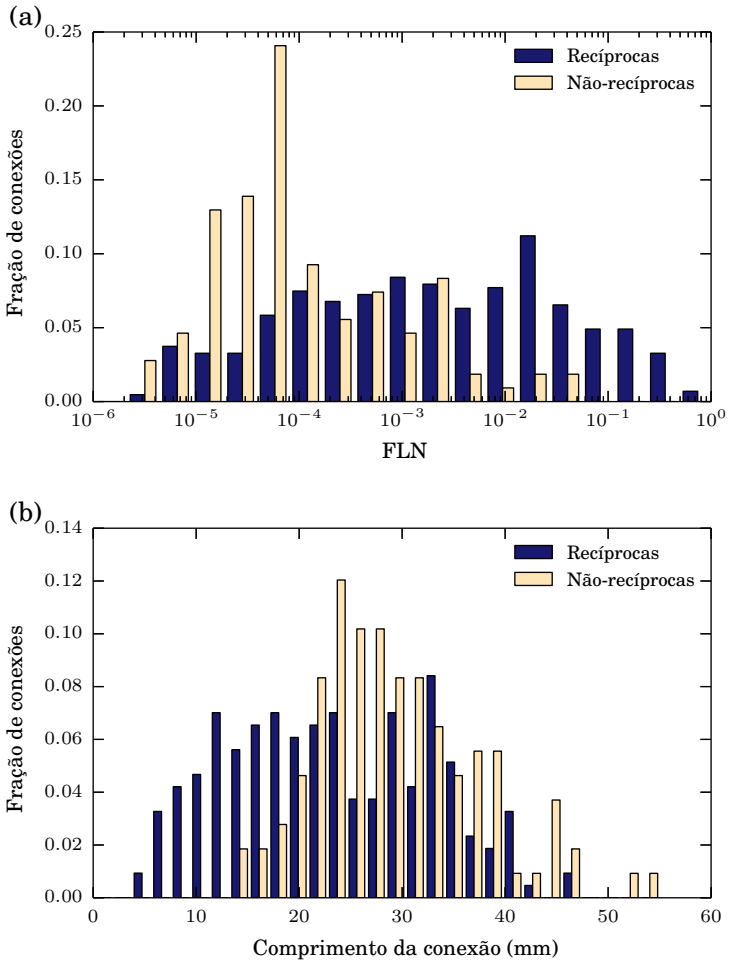


Figura 2.8: Comparação das conexões recíprocas e não-recíprocas com respeito à distribuição de valores de FLN (a) e de comprimentos de conexão (b). Os dados no painel (a) usam *log-binning* de base 2.

2.6 Discussão

2.6.1 Resumo dos resultados

Talvez um dos aspectos mais difíceis na análise dos dados de redes reais seja o que diz respeito à sua interpretação. Em geral é difícil avaliar a real relevância, sob o ponto de vista biológico, de muitas das propriedades grafo-teóricas tipicamente investigadas nesse tipo de estudo, tais como a centralidade de *betweenness* de um dado nó, ou a distribuição de motivos (*motifs*) da rede, por exemplo. Muitas dessas propriedades já foram exaustivamente descritas para redes do cérebro, entretanto muito pouco foi revelado sobre como o processamento de informação ocorre no cérebro que já não tenha sido sugerido por meio de outras abordagens.

No presente estudo, propusemos uma medida relacionada à hierarquia (DPS) para mostrar que parece haver uma forte relação entre a distribuição dos pesos das conexões de entrada em uma dada área e sua posição na hierarquia cortical. A

escolha do inverso do FLN de uma aresta como seu comprimento, embora um tanto arbitrária, garante que quanto mais uma área A é influenciada por outra área B (conforme refletido pelo FLN da aresta B $\rightarrow$ A), mais próximas elas devem estar numa hierarquia de processamento.

Além disso, a técnica de detecção de comunidades de arestas se mostrou valiosa, expondo não somente uma partição dos nós em comunidades análogas às tradicionais comunidades de nós, mas também um mapa geral dos principais trajetos da informação no córtex de macacos. Mesmo para uma rede tão pequena em termos do número de nós, sua alta densidade de arestas torna uma inspeção visual direta extremamente difícil, mesmo após filtrarmos as arestas de menor peso. A detecção de comunidades de arestas, portanto, se mostrou útil como um procedimento capaz de nos ajudar a dar sentido aos padrões intrincados presentes em uma rede tão densa.

Muitos nós, ou áreas, aparecem em diversas comunidades, indicando que cada área, a despeito de ser uma entidade individual com características citoarquitetônicas particulares, provavelmente tem a capacidade de atuar diferentemente e seletivamente de acordo com o vizinho com que estiver interagindo. Sob esse ponto de vista, ao invés de executar sempre um único tipo de computação, que na ausência de outras áreas seria inútil (tal como uma parte mecânica em um automóvel), cada área cortical aparenta ser uma unidade complexa de processamento contida em si mesma, com diversas formas de computação podendo ocorrer dependendo das fontes da informação recebida, e que, quando junto às demais áreas, torna possível a emergência de uma unidade integrada ainda maior, isto é, o córtex inteiro. Essa autonomia local no nível das áreas proporciona uma flexibilidade global no nível do córtex que explicaria, por exemplo, algumas das impressionantes capacidades de recuperação do sistema nervoso após algum trauma ou lesão.

Por fim, investigamos as características das conexões recíprocas, as quais apresentaram altas correlações entre os logaritmos de seus NN e FLN em cada direção. Ademais, a distribuição das razões de NN para cada par recíproco mostrou que conexões recíprocas empregam um número similar de neurônios em cada direção. Já a distribuição das razões de FLN revelou que conexões recíprocas tendem a exercer graus de influência similares em cada direção. É importante reforçar aqui a diferença que existe entre valores de NN e de FLN. Embora o FLN aparentemente seja uma maneira natural de traduzir dados de projeções em pesos de arestas, o próprio NN também representa uma quantidade importante, pois reflete a banda de comunicação usada para transmitir sinais entre duas áreas. Portanto, valores de FLN contam apenas parte da história por trás da comunicação no córtex.

Dito isso, devemos nos perguntar por que razões uma dada conexão de A para B deveria envolver um valor de NN similar ao de sua recíproca de B para A. Primeiro, note que, em princípio, não há nada que impeça uma área A de usar milhares de

neurônios para enviar sinais (digamos, informação visual de entrada) para B, e ao mesmo tempo receber uma resposta disparada por apenas alguns neurônios em B, por sua vez codificando, por exemplo, alguma mensagem genérica ou modulatória. Afinal, redes neurais são notadamente extremamente versáteis, permitindo toda sorte de arquiteturas (por exemplo, um único potencial disparado por um único neurônio pode, em teoria, ser propagado e levado a ativar um grupo enorme de outros neurônios).

No entanto, nossos resultados sugerem que não é isso que acontece na comunicação entre duas áreas entre as quais existem canais recíprocos de informação: uma mensagem que é recebida através de milhares de neurônios irá, da mesma forma, utilizar milhares de neurônios para ser respondida. Isso permite que seja dada uma resposta de alta "resolução". Utilizando analogia simplificada, é a diferença entre, após lermos um livro, poder dar nossa opinião sobre ele lançando mão de uma a cinco estrelas, ou escrever uma crítica detalhada, onde podem ser especificadas as partes exatas do livro que achamos boas ou ruins. Com esse tipo de resposta mais rica, uma área cortical pode atuar sobre diferentes seções da mensagem recebida, inibindo, modulando ou reforçando cada uma delas de acordo com as necessidades e os objetivos que o organismo possui naquele momento. Portanto, a reciprocidade ubíqua observada na comunicação entre pares de áreas e sua tendência a ser relativamente simétrica ajudam a reafirmar o papel de cada área cortical como uma unidade de processamento em grande parte independente, conforme discutido acima.

Também vimos na Figura 2.7 que conexões que saem de áreas inferiores rumo a superiores na hierarquia cortical tendem a apresentar valores de NN e FLN um pouco maiores do que suas contrapartes recíprocas. Isso indica que, embora as magnitudes de conexões recíprocas sejam similares, informação menos processada necessita de uma banda mais larga do que respostas processadas em mais altos níveis. Podemos imaginar informação bruta sendo conduzida de áreas inferiores para áreas superiores, sendo refinadas e mais eficientemente codificadas à medida que sobem na hierarquia. Isso resultaria em um *feedback* otimizado vindo de áreas superiores que, ainda assim, retém as características essenciais do sinal original, conforme é sugerido pelas magnitudes dos valores de NN e FLN em direções opostas.

2.6.2 Força *vs.* importância de conexões

É interessante observar que uma questão recorrente em nossas três análises principais foi o comportamento aparentemente contrastante das arestas fracas quando comparadas com as demais. Primeiro, vimos que a distribuição dos valores de FLN das arestas que chegam a uma dada área correlaciona-se mais fortemente com a posição

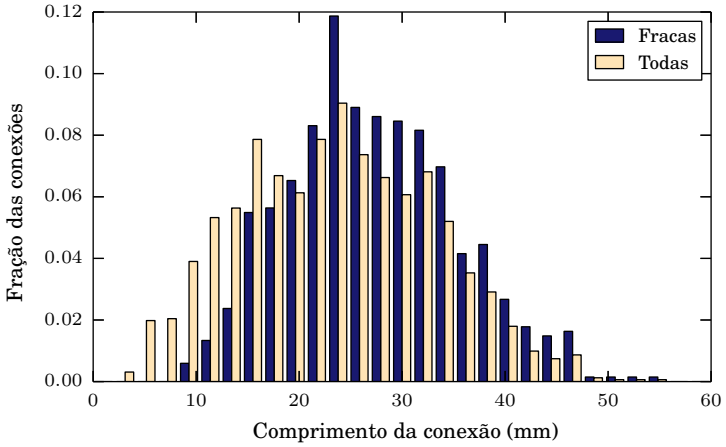


Figura 2.9: Distribuição dos comprimentos de conexão considerando-se todas as conexões e apenas as mais fracas (FLN < 0.0003616).

hierárquica daquela área quando descartamos as arestas mais fracas. Depois, vimos que a modularidade da rede parece ser muito mais bem estruturada quando ignoramos a presença dessas arestas. Finalmente, vimos ainda que não apenas as razões de NN e FLN entre conexões recíprocas ficam mais próximas de 1 quando ignoramos as arestas mais fracas, mas também que estas últimas constituem a maioria das conexões não-recíprocas. Isso nos leva a questionar a importância das conexões mais fracas para o funcionamento do córtex, já que elas parecem apenas prejudicar algumas análises úteis da rede, muito embora elas correspondam a quase metade das conexões presentes no *dataset* (se usarmos o valor de FLN igual a 0.0003616 como um limite aproximado entre conexões fracas e fortes).

Na literatura, essas conexões fracas já foram associadas a conexões longas, sugerindo que elas integrassem regiões corticais com funções distintas [21] e a promoção de sincronização através de todo o cérebro. Também já foi sugerido que elas seriam responsáveis por conectar áreas com vizinhanças dessemelhantes [29]. Tais conclusões parecem frágeis, porém, primeiro porque conexões fracas existem em todas as faixas de comprimento de conexão (Figura 2.9), e segundo porque elas estão presentes mesmo no interior de comunidades onde todas as áreas desempenham, supostamente, funções similares (conforme pode ser visto na Figura 2.5).

Em vista disso, poderíamos ficar tentados a desprezá-las como sendo secundárias às funções essenciais do cérebro. Se pensarmos em neurônios como meios de transferência de informação, faz sentido interpretar suas projeções axonais como canais de informação. Logo, parece razoável esperar que uma projeção envolvendo um número de neurônios ordens de magnitude maior que o número de neurônios em outra exerça uma influência maior sobre a atividade da área-alvo.

Seria possível ainda argumentar que a importância intrínseca de uma conexão

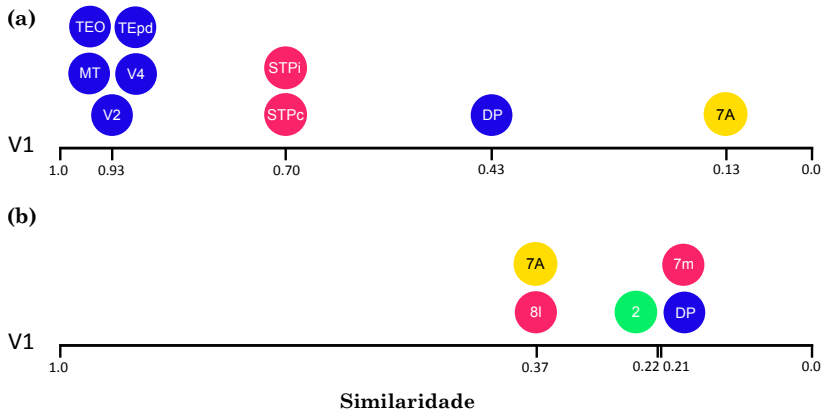


Figura 2.10: Outra maneira de se obter informações interessantes a partir dos resultados dados pelo método de [30] é inspecionando-se o dendrograma completo gerado, isto é, sem se limitar à seção de máxima densidade de partição. Um modo de fazer isso é focar em um único nó por vez e percorrer o dendrograma das folhas até a raiz, e observando, a cada nível de similaridade, quais outros nós aparecem nos *clusters* em que o nó escolhido está presente. Mostramos o resultado para a área V1, com o painel (a) referindo-se à rede sem as arestas mais fracas (FLN ≥ 0.0003616) e o painel (b) à rede contendo apenas as arestas mais fracas (FLN < 0.0003616).

entre áreas esteja no significado real do sinal sendo transmitido através dela e em sua contribuição para a sobrevivência do organismo. Mas, mesmo sob essa perspectiva, parece difícil ignorar o fato de que, por uma mera questão de proporção, uma conexão envolvendo mil vezes mais neurônios que outra terá o potencial de causar um efeito físico maior, por exemplo, ou de possuir um repertório mais variado de mensagens, e certamente de codificar muito mais informação. Também sabemos que boa parte do nosso aparato biológico deve sua estrutura a pressões evolutivas, logo é razoável admitir que conexões utilizando mais neurônios tenham tido um papel mais importante na adaptação do organismo através das eras do que uma conexão que é ordens de magnitude mais "fina".

Por outro lado, dado o custo energético considerável da comunicação no cérebro, parece improvável que um número tão grande de conexões não seja importante para o processamento cortical. Nossos estudos apresentaram novas observações acerca das características dessas conexões: elas são altamente distribuídas, não contribuem para a organização modular, e a maioria delas ($\sim 89\%$, considerando a matriz de adjacências 29×29) não possui contraparte recíproca, o que as diferencia ainda mais das conexões fortes. Se elas são de fato importantes para o processamento no cérebro, então provavelmente estão envolvidas em atividades bem diferentes.

Se examinarmos mais atentamente essas conexões olhando para as comunidades resultantes de uma rede contendo apenas elas (isto é, apenas as arestas com FLN $<$ 0.0003616), encontramos uma única comunidade compreendendo todos os nós no ponto de máxima densidade de partição. Além disso, averiguando-se o dendrograma

completo (veja a Figura 2.10), fica claro que as conexões fracas geram similaridades que agrupam áreas pertencentes às mais diversas regiões corticais, de acordo com o que foi sugerido por [29]. Essa tendência é notavelmente distinta daquela vista para as arestas escolhidas acima do *threshold* de FLN: naquele caso, as comunidades resultantes continham em sua maioria nós fisicamente próximos uns dos outros. É como se as arestas fracas formassem uma rede própria, representada por essa única comunidade, completamente distribuída através de todas as regiões do córtex e independentemente das funções individuais das áreas às quais elas são incidentes. O mais provável é que essas conexões estejam todas envolvidas em alguma função que permeia todo o córtex.

Portanto, afinal de contas talvez as conexões fracas tenham de fato um papel crucial no processamento cortical: elas poderiam facilitar sincronização e integração, ou comunicar algum tipo específico de informação, ser responsável pela formação de memórias de longo termo, ou até mesmo servir como vias alternativas a serem usadas em caso de lesão ou mau funcionamento no cérebro. Estudos futuros que foquem especificamente esse aspecto da comunicação cortical certamente nos levarão um passo adiante em direção a um melhor entendimento do cérebro dos mamíferos.

Capítulo 3

Aprendizado e detecção de padrões

3.1 Aprendizado de padrões no córtex

A capacidade inata dos mamíferos—e de grande parte dos animais—de encontrar e reconhecer padrões é indiscutível. Isso inclui desde a simples percepção de imagens e sons do ambiente até o reconhecimento de um fruto como estando maduro ou do ataque de um predador. Nós, humanos, levamos essa habilidade ao extremo ao sermos capazes de criar linguagens complexas para comunicação e usar analogias e extrapolações para resolver problemas de todo tipo. Talvez o mais impressionante seja que façamos isso de forma totalmente automática: começamos a falar simplesmente ouvindo as pessoas se comunicando ao nosso redor, e ninguém precisa de aulas para aprender a reconhecer o rosto dos pais, por exemplo. Nosso córtex tem uma capacidade inata de construir padrões baseado simplesmente no fluxo de informação que chega até ele através de nossos sentidos. De fato, o córtex dos mamíferos parece ser especificamente adaptado para permitir esse aprendizado de forma poderosa e eficiente. Por exemplo, é amplamente conhecido que nossa seletividade de padrões vai se tornando mais complexa à medida que o sinal de entrada passa por áreas sucessivas do córtex. No caso do sinal visual, certamente o mais estudado e mais bem compreendido dentre nossos sentidos, partimos de uma mera identificação de formas tipo *center-surround* na retina, passando pela identificação de arestas orientadas em V1, depois por formas geométricas em V4, para, em áreas mais superiores no lobo temporal, codificarmos conceitos semânticos como "casa", ou até mesmo um exemplo específico como "a casa onde fomos criados na infância". Tendemos a imaginar esse processo como uma aglomeração de características (*features*) cada vez mais complexas e incorporando cada vez mais informação, inclusive de outras naturezas, como cor, ou posicionamento relativo ao organismo. E, mais que isso,

o córtex realiza o feito impressionante de "abstrair", ou extrapolar, esses padrões permitindo reconhecê-los independentemente da presença de ruídos de diversos tipos (iluminação, tamanho, rotação, etc.).

3.2 Aprendizado de máquina e modelos distribuídos de aprendizado

Não é de hoje que o homem tenta imitar essa capacidade de aprendizado em programas de computador; na verdade a busca por uma chamada "inteligência artificial" se mistura com a própria história da ciência da computação. Desde os anos 60, e mesmo antes disso, com o estudo da cibernética (ver, por exemplo, [35] para um resumo histórico), vem rompendo barreiras sucessivas no caminho para esse objetivo cuja definição é até mesmo difícil de precisar. Um dos campos de maior sucesso nesse sentido tem sido o chamado "aprendizado de máquina", que engloba uma coleção de técnicas que permitem inferir regras ou funções a partir de um número geralmente grande de exemplos de treinamento. Entretanto, boa parte dos métodos se propõe a um aprendizado de alto nível, o que requer exemplos nos quais implicitamente esteja incluída uma parcela considerável do nosso próprio conhecimento. Já os métodos que aceitam exemplos "puros", isto é, representados de forma análoga àquela em que nosso sentidos recebem informação do ambiente (como imagens em forma de pixels ou sons em forma de frequências), funcionam de forma supervisionada (requerem um rótulo classificando cada exemplo). Logo, fica patente que, embora dando origem a métodos extremamente úteis e eficazes no aprendizado de tarefas das mais variadas, o aprendizado de máquina tradicional não nos conduz a um melhor entendimento do funcionamento do córtex.

Recentemente, técnicas para aprendizado automático (não-supervisionado) de *features*, em analogia ao que é feito no córtex, têm sido desenvolvidas e empregadas com aparente sucesso, em especial para a classificação de imagens. Um dos maiores motivadores para isso é a abundância de imagens, vídeos e textos disponíveis gratuitamente na internet: a ideia de utilizar esse gigantesco repositório de exemplos para treinar algoritmos sem a necessidade de supervisão é bastante atraente. Dessas técnicas, tem tido destaque a *unsupervised feature learning* (aprendizado não-supervisionado de atributos), que é a etapa preliminar do *deep learning* (que usa redes neurais artificiais "profundas", isto é, construídas com múltiplas camadas ocultas) [36]. Em linhas gerais, a etapa não-supervisionada consiste em treinar com o algoritmo de *backpropagation* uma rede neural de apenas uma camada oculta que forneça como saída vetores iguais aos que ela receber como entrada (em outras palavras, a rede aprende uma função identidade). A ideia por trás disso é que, embora

esse aprendizado pareça trivial, podemos impor restrições (por exemplo, usando menos neurônios na camada oculta do que na camada de entrada) que levam a rede a aprender uma representação comprimida dos exemplos de entrada. Essa abordagem é repetida diversas vezes, dando origem a cada uma das camadas ocultas que serão usadas na "rede profunda" para aprendizado supervisionado. Entretanto, essa técnica é extremamente custosa do ponto de vista computacional. Além disso, o aprendizado só pode ser feito após todos os exemplos de entrada terem sido utilizados (diversas vezes, cada um), e portanto difere essencialmente da forma como nosso cérebro processa informação sensorial.

Em paralelo à corrente do aprendizado de máquina, ao longo das últimas décadas diversos modelos chamados "conexionistas" foram propostos especificamente com o intuito de tentar explicar ou replicar nossa capacidade inata de aprendizado, acompanhando os estudos e descobertas da psicologia cognitiva ([37], [38], [39], e, mais recentemente, [40], [41], [42]). Assim como nas famosas redes neurais artificiais, eles se baseiam na utilização de unidades distribuídas de processamento conectadas entre si através de arestas com pesos, inspiradas livremente nas conexões entre neurônios presentes em nosso córtex. Embora em geral exemplos bastante simples sejam usados, o treinamento é não-supervisionado e atingem-se muitos resultados compatíveis com as capacidades humanas, até mesmo em domínios teoricamente mais complexos como linguagem natural (por exemplo, o aprendizado automático da formação de verbos no tempo pretérito [43]). Além disso, muitas vezes é possível explorar arquiteturas mais complexas (tais como o uso de diversas redes em paralelo com comunicação entre si) sem perder o controle do que cada unidade na rede representa, o que em geral não é o caso com redes neurais artificiais. Em resumo, utilizando-se um modelo simples porém focado em simular as capacidades humanas, busca-se entender melhor as origens da inteligência nos seres vivos. Isso em geral implica uma incapacidade de lidar com problemas práticos de larga escala; nesse sentido, tais modelos e as técnicas de aprendizado de máquina são complementares.

3.3 Um algoritmo para aprendizado de padrões

3.3.1 Requisitos

Conforme discutido acima, o aprendizado não-supervisionado de padrões é potencialmente uma das bases do processamento de informação no córtex dos mamíferos. Logo, é possível que um algoritmo que seja capaz de aprender padrões e que seja minimamente plausível do ponto de vista biológico possa auxiliar no entendimento das descobertas feitas no Capítulo 2.

Para criar um algoritmo que seja ao menos plausível em termos do processamento

real que ocorre no córtex de humanos e mamíferos em geral, gostaríamos que algumas características básicas fossem atendidas:

- genérico o suficiente para aprender padrões, qualquer que seja o domínio a que pertençam as entradas;
- *online*, isto é, o aprendizado precisa ocorrer de forma dinâmica, à medida em que o algoritmo recebe as entradas (ou em "tempo real");
- permitir uma implementação distribuída, ou seja, que utilize unidades de processamento tal que cada nova entrada lida provoque alterações apenas nas unidades relevantes, sem depender das demais; essa é uma das características mais marcantes do processamento no cérebro.

Para tanto, vamos seguir as diretrizes resumidas em [40] acerca da natureza das representações mentais:

1. o conhecimento é representado na forma de padrões de atividade distribuídos através de grandes conjuntos de unidades simples de processamento;
2. o processamento ocorre a partir de transformações dos padrões de atividade através de um grande número de conexões entre as unidades;
3. o aprendizado ocorre como resultado da confluência de mecanismos inatos porém genéricos (sem especificidade de domínio) com a experiência adquirida; isso se traduz na rede através de mudanças nas forças da conexões entre as unidades em resposta a sinais (ou entradas) externos.

Para pensar num algoritmo que atenda a esses itens, primeiro precisamos de um domínio no qual testá-lo. Muito embora os exemplos mais clássicos de aprendizado de padrões ou *features* na literatura sejam com imagens, este é um domínio que traz consigo dificuldades intrínsecas em função das duas dimensões envolvidas e da presença inerente de ruído. Além disso, há ainda um terceiro complicador mais fundamental que é o simples fato de que não sabemos dizer ao certo quais deveriam ser os padrões a serem descobertos; logo, não temos uma base sólida para comparação e avaliação dos resultados. Para simplificar nossos objetivos, seria interessante trabalhar com um domínio de apenas uma dimensão, onde não haja ruído (ou seja, os padrões são absolutos) e para o qual saibamos avaliar facilmente a pertinência dos padrões aprendidos.

O domínio escolhido para começo de trabalho, portanto, é o de texto, ou seja, uma sequência de símbolos, ou caracteres, organizados de tal forma a fazer sentido em uma dada linguagem. Este domínio apresenta a vantagem de que certos complicadores, tais como a necessidade de lidar com translação ou rotação existentes

no processamento de imagens, por exemplo, estão ausentes (cabe reiterar aqui que estamos tratando o texto como uma sequência de elementos abstratos pertencentes a um determinado alfabeto, e não como uma sequência de imagens de letras). Além disso, ao optarmos pela ausência de ruído (o qual, no contexto de textos, equivaleria à presença de algumas letras trocadas, ausentes ou em excesso), estamos simplificando enormemente o aprendizado de padrões.

Quanto à última necessidade, referente à avaliação dos padrões aprendidos, uma vez que sabemos que um texto é formado por frases, e essas por palavras, as quais por sua vez são formadas por sílabas e letras, esperamos que os padrões aprendidos sejam alguns desses elementos que apareçam com maior frequência. Intuitivamente, os padrões aprendidos deverão ser rapidamente reconhecidos por nós como elementos básicos para a construção do texto em questão. Se isso acontecer, será indicativo de que o algoritmo poderia funcionar também para outros domínios. (Em especial, se o algoritmo detectar as palavras mais frequentes como padrões, por exemplo, saberemos que estamos no caminho certo. Em outras palavras, queremos que o resultado seja consideravelmente diferente de um algoritmo simplório que simplesmente particione o texto em diversos pedaços, todos com o mesmo número de caracteres, os quais corresponderiam a palavras por pura sorte.) O uso de texto apresenta ainda a vantagem de poder ser alimentado ao algoritmo como um fluxo contínuo de informação, análogo ao que os animais recebem através de seus sentidos, o que é particularmente interessante uma vez que queremos simular o tipo de aprendizado tal como ocorre em organismos vivos.

3.3.2 Princípio de funcionamento

Uma estratégia ingênua seria estipular um tamanho fixo para nossos padrões e simplesmente percorrer o texto anotando todos os trechos (*substrings*) com esse tamanho e contando a frequência com que cada uma deles apareceu no texto todo. Isso poderia ser feito com uma única passada pelo texto. Entretanto, como escolher um tamanho "correto"? Usar os "espaços" como delimitadores de palavras não é uma boa ideia: muitos sistemas antigos de escrita, e até mesmo hoje em dia, não utilizam qualquer símbolo para separação de palavras, sem falar em linguagens que estejam em código e outros tipos de linguagens mais abstratas. O ideal seria que pudéssemos encontrar padrões de diversos tamanhos. Ora, para isso basta percorrer o texto sucessivamente, cada vez buscando por um tamanho diferente: 2, 3, 4, 5, 6... Anotamos as frequências de todos esses padrões e selecionamos aqueles mais frequentes. Mas agora, além de necessitarmos de múltiplas passadas no texto, indo contra nosso requisito de aprendizado *online* acima, estamos criando um problema adicional: para cada padrão frequente que encontrarmos de tamanho 5, digamos,

teremos encontrado também todos os sub-padrões de tamanhos 4, 3 e 2 que estão contidos nesse padrão de tamanho 5. Logo, teremos a tarefa adicional de separar, dentre esses padrões menores, aqueles que tenham frequência significativamente maior do que os padrões em que eles estejam contidos e descartar os demais. Há ainda um problema mais fundamental: como estipular as frequências relevantes para cada tamanho de padrão, se não conhecemos a linguagem de antemão?

Ao invés disso, vamos usar os princípios e requisitos enumerados acima para elaborar uma estratégia distribuída de aprendizado de padrões em uma rede composta por diversas unidades de processamento conectadas entre si. Imagine que os padrões serão representados por nós, e que partiremos de padrões bastante simples, mas que padrões mais complexos poderão ser formados a partir de combinações de outros mais simples. Estas seriam representadas através da formação de novos nós conectados por arestas aos nós mais simples. Ou seja, nós serão conectados uns aos outros para formar, progressivamente, padrões mais complexos dispostos em uma rede.

Suponha que tenhamos uma rede onde já existam de antemão alguns padrões, cada um deles representado por um nó. Toda vez que um desses padrões é encontrado no sinal de entrada (isto é, no texto sendo lido sequencialmente), seu nó correspondente é "ativado". Essa ativação contribui para fazê-lo sobreviver, pois gostaríamos que nós que não forem ativados suficientemente sejam removidos (como será explicado mais adiante). A rede segue sendo construída da seguinte maneira: para cada par de nós ativados consecutivamente, criamos um novo nó-filho o qual é conectado a seus dois nós-pais através de arestas. Este novo nó representará o segmento de texto formado pela concatenação dos padrões representados por seus nós-pais (ver Figura 3.1). Com isso, pouco a pouco o sinal de entrada vai desencadeando a formação de nós cada vez mais complexos e, dessa forma, o crescimento da rede pode ser encarado como sendo feito de baixo para cima (ou *bottom-up*). Mas, como saber se esses nós iniciais representam padrões realmente pertinentes? Gostaríamos de encontrar uma maneira automática de reter apenas os padrões mais frequentes e descartar os demais.

Considerando que um texto pode, em princípio (dado que conheçamos os padrões adequados), ser totalmente particionado em padrões de diversos tamanhos (por exemplo, em palavras ou conjuntos de palavras mais frequentes), então qualquer fragmento de texto obrigatoriamente deverá ser um padrão, ou um pedaço de um padrão (um "sub-padrão"), ou então deverá possuir algum padrão contido nele. Por exemplo: se num texto em português o fragmento "roupa" é considerado um padrão, então "oup" seria um pedaço desse padrão e "s roupas azu" seria um fragmento contendo tal padrão. Logo, podemos resumir nosso problema da seguinte forma: queremos uma maneira de adivinhar se um padrão representado por um nó

presente na rede está grande ou pequeno demais. Para isso, podemos ir lendo a entrada e, sempre que encontrarmos parte de um dos padrões presentes na rede (ou seja, um sub-padrão), vamos reconhecê-la como forte candidata a ser ela própria um padrão, pois pode ser que o padrão que tínhamos possuísse símbolos em excesso. (Por exemplo: se o texto está escrito em português e já possuímos um nó com o padrão "sempre que", mas nos deparamos com "mas sempre via", podemos reconhecer o sub-padrão "sempre"; isso seria um indicativo de que talvez "sempre" seja mais relevante que "sempre que".)

Por outro lado, pode ser também que um padrão que já possuamos seja apenas uma parte de outro padrão mais significativo. Logo, precisamos juntar nosso padrão atual com mais algum pedaço de texto adjacente a ele, o que é feito através do procedimento de formação de pares descrito anteriormente. Ao concatenar padrões adjacentes, temos como resultado três possibilidades: podemos estar dando origem a um padrão pertinente, o que resolveria nosso problema, ou então a um padrão grande demais, ou ainda a um que continue pequeno demais. Mas note que o segundo caso já foi resolvido acima (se um padrão é menor do que o necessário, acabaremos encontrando um sub-padrão relevante), e, no último caso, o novo nó seguirá formando pares até chegar ao primeiro ou segundo casos. Com isso, estamos nos aproveitando de padrões de "baixa qualidade" como guias para nos ajudar a encontrar os padrões realmente significativos.

E o que fazer com uma sequência de símbolos encontrada no texto que porventura não faça parte de nenhum dos padrões já presentes na rede? Vamos tratá-la como um novo padrão. Na realidade, isso inclusive nos revela como dar origem aos padrões iniciais na rede: se estipularmos um tamanho máximo de símbolos permitido para os padrões, podemos começar com a rede vazia e, após termos lido esse número de símbolos no sinal de entrada, tratamos esse trecho como sendo um novo padrão. A partir disso seguimos conforme explicado acima.

Vamos ilustrar com um exemplo: suponha que o texto comece com "`era-uma-vez...`" (em que trocamos os espaços por "`-`", por uma questão de clareza) e que nosso tamanho máximo de símbolos por padrão seja 5. A rede inicialmente não possui nós, portanto após ler os 5 primeiros símbolos, formamos o primeiro padrão com eles, "`era-u`", o qual é adicionado à rede na forma de um nó e em seguida ativado. Seguimos lendo o texto de onde paramos, e prosseguimos até encontrar um padrão ou sub-padrão presente na rede. O próximo símbolo lido é "`m`", que não é sub-padrão, seguido de "`a`", que é sub-padrão de "`era-u`". Acrescentamos e ativamos portanto um novo padrão "`m`" à rede e, uma vez que ativamos dois nós em sequência, formamos o par "`era-um`" (um novo nó conectado aos dois que deram origem a ele).

Seguimos lendo os símbolos de entrada enquanto houver correspondência entre

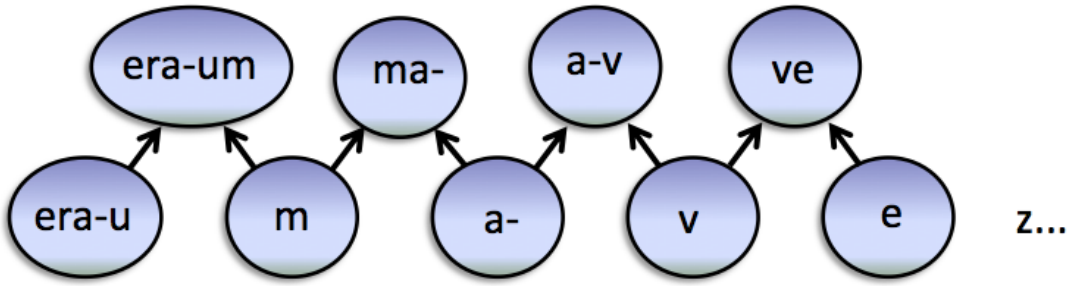


Figura 3.1: Exemplo de rede de padrões formada após processamento de uma entrada começando por "`era-uma-vez...`".

os símbolos chegando com algum sub-padrão da rede. O próximo símbolo é "`-`", que aumenta o sub-padrão que já estava reconhecido para "`a-`". O próximo símbolo lido, "`v`", não contribui para aumentar esse sub-padrão, portanto finalizamos a correspondência do sub-padrão "`a-`' e o acrescentamos à rede, ativando-o e formando um par com o anterior, o que dá origem ao nó "`a-v`". O próximo símbolo lido, "`e`", faz parte do padrão já conhecido "`era-u`"; seguimos, portanto, para o próximo, pois ele poderá ainda ampliar essa correspondência. (Repare como a ideia é sempre reconhecer o maior padrão ou sub-padrão possível—mesmo que já tenhamos feito a correspondência com um padrão completo, pode ser que haja um padrão maior no qual o primeiro está contido e que poderá ser reconhecido com os símbolos da entrada que ainda estão por vir. Além disso, os pares recém-formados não são usados para formar par com os outros nós ativados.) Porém, ao ler "`z`", vemos que isso não ocorreu; portanto, vamos criar o nó "`e`", que é ativado e forma o par "`ve`". E assim por diante, enquanto houver entrada sendo alimentada ao algoritmo. Veja na Figura 3.1 a rede resultante desse curto exemplo treinamento.

3.3.3 Controlando o número de nós da rede

Nosso algoritmo ainda não está completo. Queremos que padrões úteis sejam mantidos e padrões inúteis sejam descartados, para que nossa rede não tenha um número de nós além do mínimo necessário. Suponha que contemos o tempo de treinamento decorrido de acordo com o número de padrões lidos pelo algoritmo. Então, podemos designar um tempo de vida k para cada padrão P formado, após o qual o mesmo será destruído. Mas a cada vez que um padrão aparece seu tempo restante de vida, R, é prolongado por mais k unidades. Com isso, se um padrão P_i tem tempo de vida k_i, então após k_i passos do algoritmo (ou seja, após k_i padrões encontrados), o padrão P_i será destruído, a menos que ele apareça novamente dentro desse intervalo de tempo. Logo, quanto mais vezes ele for encontrado, mais longa será sua existência. Portanto, o parâmetro k reflete a frequência mínima com que um dado

padrão precisa aparecer para continuar existindo na rede (uma vez a cada k passos). Note também que, em qualquer instante, se dividirmos o tempo restante de vida R_i de cada padrão pelo seu tempo de vida k_i e tomarmos o teto desse resultado, obteremos o número de vezes que o padrão foi reconhecido dentro dos últimos k_i passos do algoritmo.

Mas como escolher valores adequados para k? Uma proposta fadada ao fracasso seria escolher um k único que servisse para todos os padrões; afinal, é altamente plausível que alguns padrões sejam menos frequentes do que outros, embora igualmente ou até mais importantes. Por exemplo: a sílaba "re" aparece muito mais frequentemente em um texto em português ou inglês do que qualquer palavra individual que a contenha, simplesmente porque existe mais de uma palavra contendo essa sílaba. Logo, um k adequado à frequência da sílaba "re" não servirá para nenhuma palavra real que a contenha, e como resultado nossa rede manteria apenas padrões simples demais. De fato, intuitivamente esperamos que padrões maiores apareçam com menor frequência que padrões menores, e portanto precisem de tempos de vida maiores.

Uma ideia que acomodaria esse fato seria designar valores maiores de k para padrões mais complexos, isto é, gerados a partir da formação de pares. Note que o número de símbolos que constituem um padrão não é um indicador preciso de sua frequência; afinal, dois padrões podem ser igualmente frequentes porém diferir drasticamente no número de símbolos que contêm (por exemplo, "is" e "were", no inglês, têm frequências parecidas porém o segundo tem o dobro de letras). Em contrapartida, o número de pares que precisaram ser formados para a descoberta de um dado padrão é um indicador muito mais confiável da frequência esperada com que um padrão deve aparecer.

Portanto, uma ideia razoável seria definir o k_i de um padrão P_i como sendo uma função da "altura" de P_i na rede, ou a menor distância dele até uma das "folhas" (nós sem pai). Mas não podemos perder de vista que todo novo padrão formado é essencialmente uma aposta: não sabemos se ele é de fato pertinente ou não. Logo, o problema com essa estratégia é que, se começarmos com padrões pequenos e formos aumentando o valor de k para cada nível que subimos na rede, então, a depender da regra que usarmos para aumento de k, todo padrão formado acima de um certo nível acabará por ter um k grande demais, e com isso estaremos "eternizando" todo padrão grande, indiscriminadamente. De qualquer forma, escolher uma função ideal para o valor de k não parece ser tarefa simples.

Ao invés disso, vamos partir da observação de que, se um determinado padrão é muito frequente, esperamos encontrá-lo mais cedo no texto do que um outro padrão que seja mais raro. É claro que um padrão, por mais raro que seja, pode perfeitamente aparecer logo no começo do texto. Mas, ainda assim, depois disso ele deverá

levar, em média, bastante tempo para aparecer de novo, já que é raro. Com isso em mente, podemos pensar em atribuir valores de k baseados em quanto tempo (número de passos do algoritmo) levamos para descobrir cada padrão.

Padrões descobertos rapidamente teriam valores de k pequenos, pois esperamos que eles apareçam frequentemente. Por outro lado, padrões que custaram a aparecer receberão maior tempo de vida, pois esperamos que sejam menos frequentes. Uma maneira simples e direta de se conseguir isso é atribuir, para cada padrão, um k igual ao número de passos decorridos quando de sua descoberta. Note ainda que, na eventualidade de um padrão ser aprendido "cedo demais", isto é, se um padrão pouco frequente porém importante é aprendido logo no início da leitura do texto, ele será extinto pouco tempo depois (devido a seu k baixo). Entretanto, se ele for de fato importante, ele fatalmente reaparecerá, e, dessa vez, com mais tempo havendo decorrido, ele finalmente receberá um valor de k condizente com sua frequência verdadeira. O outro extremo, que seria um padrão muito frequente que demore a aparecer no texto, não seria problema, pois se ele é importante não faremos mal em designar a ele um k grande. Porém, note que isso não é esperado, já que se um padrão é frequente então ele deverá fatalmente aparecer rapidamente.

Para completar, podemos estipular um valor máximo para k, que corresponderia a uma frequência mínima de ocorrência de um padrão para que ele seja considerado um padrão pertinente. Isso para que, independentemente do tamanho total do texto, tenhamos um tempo máximo aceitável de vida para um padrão após o qual ele deve aparecer de novo. Assim, evitamos atribuir valores de k exageradamente grandes para eventuais padrões aprendidos tardiamente no texto mas que sejam espúrios, isto é, meros acidentes—como uma palavra que é inventada pelo autor, ou até mesmo um erro de ortografia (embora tenhamos escolhido usar entradas sem ruído, como ter certeza de que eliminamos todo e qualquer ruído num texto longo?). Isso garante que esses "não-padrões", decorrido um número razoável de passos, acabem sendo excluídos.

Um detalhe adicional: para garantir que os padrões detectados logo no início do treinamento (quando o tempo decorrido ainda é extremamente pequeno) não sejam excluídos antes de terem tido a oportunidade de reaparecerem ou de servirem como base para outros padrões, definimos um valor mínimo $k_{\min}$ para k; ao mesmo tempo, garantimos um mínimo de tempo razoável para que ele tenha a chance de reaparecer. Assim, se $k_{\min} = 20$, um padrão criado, por exemplo, no tempo $t = 2$, ao invés de ser excluído dois passos depois, sobreviverá por pelo menos 20 passos, durante os quais ele terá a chance de formar pares ou de ter ele próprio (ou um sub-padrão seu) reconhecido.

Com isso, já temos o suficiente para implementar nosso algoritmo. Idealmente, queremos que ele seja minimamente eficiente do ponto de vista computacional de

modo a permitir o processamento de grandes corpos de texto em tempo viável.

3.3.4 Implementação

O exemplo dado acima mostrou que, a cada novo símbolo lido na entrada, precisamos verificar se o trecho sendo lido no momento faz parte de algum dentre todos os padrões presentes na rede. Essa é sem dúvida a operação mais custosa do algoritmo: se queremos eficiência, é essencial realizar essa busca de forma otimizada. Note que, se pudéssemos executar o algoritmo de maneira verdadeiramente distribuída, uma alternativa muito eficiente seria ignorar a exclusão de nós, mantendo todo o histórico da rede. Com isso, bastaria ativar os nós mais básicos (aqueles que não possuem nós-pais) presentes na entrada e ativar seus filhos. A partir daí, cada nó cujos dois pais tiverem sido ativados também é ativado, ativando seus filhos, e assim sucessivamente. Os padrões maximais reconhecidos seriam simplesmente os últimos nós ativados ao final desse processo, que teria complexidade $O(x)$, onde x é o número máximo de pares necessários para formar qualquer padrão da rede.[1]

Na prática, entretanto, precisamos de um procedimento que seja eficiente ao ser executado sequencialmente, compatível com a arquitetura de nossos computadores. Em essência, a tarefa que precisamos realizar é reconhecer trechos parciais (seja no começo, no fim ou no meio) de padrões previamente aprendidos. Existe uma estrutura de dados que cumpre justamente essa função: trata-se da árvore de sufixos [44], ou *suffix trie*[2], em que organizamos sequências de símbolos em uma estrutura em forma de árvore, com cada símbolo constituindo um nó, de forma análoga ao que é feito numa *trie* tradicional. Ela se propõe a armazenar diversas sequências, possibilitando a busca posterior de qualquer sufixo de uma delas. Para isso, ao inserirmos nela uma sequência de símbolos (ou "palavra"), adicionamos todos os seus sufixos acrescidos de um marcador de final de palavra (\$). Por exemplo, para adicionar "`azul`", seriam adicionados: "`a` → `z` → `u` → `l$`", "`z` → `u` → `l$`′′, "`u` → `l$`" e "`l$`", sempre usando o máximo de nós já presentes na árvore que puderem ser aproveitados. Uma simples modificação no algoritmo original permite que encontremos não apenas sufixos, mas qualquer trecho: basta retornar a busca independentemente de se ter chegado a um marcador de final de palavra. Com isso, passamos a achar tanto sufixos como qualquer segmento de uma palavra previamente adicionada à árvore, uma vez que qualquer sub-palavra pode ser vista como o prefixo

[1] Observe que a complexidade real, porém, em geral seria menor que isso, pois só levaremos x passos se o padrão sendo buscado tiver sido gerado símbolo a símbolo.

[2] Na literatura costuma-se fazer uma distinção entre "árvore" e "trie" no que diz respeito ao número de símbolos representados em cada nó: se cada nó guarda apenas um símbolo, temos uma *trie*; caso contrário, temos uma árvore (ou ainda uma "*trie* compacta"). Embora vamos utilizar apenas um símbolo para cada nó, ao longo do texto não faremos distinção entre uma denominação ou outra.

de algum sufixo da palavra principal (no exemplo que acabamos de dar, poderíamos encontrar "`zu`", ou "`u`", que são prefixos dos sufixos "`z` → `u` → `l$`" e "`u` → `l$`", respectivamente). Isso significa que, no nosso algoritmo, podemos adicionar todos os padrões aprendidos a uma árvore desse tipo e encontrar qualquer sub-padrão com facilidade.

De fato, uma implementação direta dessa estrutura permite adicionar ou remover um padrão em tempo $O(n^2)$ e encontrar um sub-padrão qualquer em tempo $O(n \log n)$, onde n é o tamanho do padrão e do sub-padrão, respectivamente (ambas as complexidades podem chegar a $O(n)$ com otimizações para casos específicos, conforme [45]). Note ainda que essas operações podem todas ser realizadas a qualquer momento do aprendizado. A principal desvantagem dessa estrutura é notadamente o espaço requerido, já que pretendemos guardar um número bastante grande de sequências. Outros algoritmos a princípio mais eficientes para esse tipo de tarefa, que dispensam o uso da árvore de sufixos, não são adequados para o nosso caso pois temos uma situação um tanto particular: um número muito grande padrões a serem buscados cujos tamanhos podem variar desde um símbolo até o tamanho máximo estipulado. Além disso, a necessidade de alterar nossa coleção de padrões à medida que vamos lendo a entrada elimina também diversos outros algoritmos que utilizam algum tipo de pré-processamento dos padrões antes de buscá-los.

Finalmente, podemos escrever nosso algoritmo. O pseudo-código para ele encontra-se no Algoritmo 3.1. Ele recebe como entradas a rede atual, bem com o texto sendo aprendido[3] e a posição atualmente sendo lida, retornado a nova posição de onde a leitura parou. O algoritmo é executado repetidamente, enquanto i for menor que o tamanho total do texto ou entrada.

Estamos considerando que o objeto Rede mantém, além dos nós representando cada padrão, a informação sobre o último padrão visto (isto é, o último nó ativado), bem como o $k_{\min}$ e tamanho máximo permitido por janela. Além disso, dentre as sub-rotinas utilizadas, a única que merece detalhamento é a *ENCONTRAR_PADRAO*: ela recebe uma janela e, partindo da primeira letra nessa janela, inicia uma busca na árvore de sufixos da rede (*suffix trie*). Se a primeira letra faz parte de um padrão já existente, ela continua a busca e retorna o maior padrão ou sub-padrão encontrado. Caso contrário, continua de letra em letra até encontrar a primeira letra que faça parte de um padrão já existente, digamos a n-ésima letra. Nesse caso ela retorna as $n-1$ primeiras letras na forma de um padrão inédito.

É possível executar agora esse algoritmo em um texto qualquer e observar os padrões aprendidos, mas o ideal seria termos algum critério para julgar a qualidade desses padrões. Lembre-se de que escolhemos trabalhar com texto justa-

[3]Embora estejamos assumindo que já possuímos a entrada completa a ser alimentada ao algoritmo, este é facilmente modificado para receber símbolo a símbolo, de forma *online*.

Algoritmo 3.1 Algoritmo para aprendizado de padrões

Input: Rede, texto, i
  Rede.t $\leftarrow$ Rede.t + 1 // *atualiza o tempo de treinamento*
  ATUALIZAR_NOS(Rede) // *verifica os nós a serem ser excluídos dado o tempo atual*
  janela $\leftarrow$ texto $[i : i +$ Rede.tamanho_max_janela$]$
  padrao $\leftarrow$ *ENCONTRAR_PADRÃO*(janela, Rede.suffix_trie)
  $i \leftarrow i +$ padrao.**size**
  if *INÉDITO?*(padrao) **then**
    $k \leftarrow$ Rede.$k_{\min}$
    ATIVAR_NO(Rede, padrao, k)// *ATIVAR_NO inclui a tarefa de criar o nó, caso ele não exista*
  else if *É_PADRAO?*(padrao) **then**
    // *nó já existe, então é ativado para atualizarmos seu tempo restante de vida*
    ATIVAR_NO(Rede, padrao)
  else
    // *não é um padrão inteiro, ou seja, está propriamente contido em um padrão já existente*
    $k \leftarrow$ Rede.t
    ATIVAR_NO(Rede, padrao, k)
  end if
  if Rede.ultimo_padrao_visto $\neq$ **null then**
    $k \leftarrow$ Rede.t
    FORMAR_PAR(Rede.ultimo_padrao_visto, padrao, k)
  end if
  Rede.ultimo_padrao_visto $\leftarrow$ padrao // *atualiza o último padrão visto*
  return i

mente pela facilidade de avaliação dos padrões detectados, já que temos as palavras como sendo conceitos que julgamos plausíveis como unidades básicas de formação de um texto em liguagem natural. Uma maneira direta de fazer isso é verificar a fração das palavras presentes no texto que foram aprendidas, ou seja, que foram apontadas como sendo um padrão. Usando essa abordagem, testamos nosso algoritmo com textos de domínio público disponíveis na página do Projeto Gutenberg (http://www.gutenberg.org).

3.4 Resultados

Os resultados para o aprendizado de padrões a partir da leitura dos livros "David Copperfield", de Charles Dickens, e "Os Maias", de Eça de Queirós, estão resumidos na Tabela 3.1. Uma simples verificação dos padrões encontrados com maior frequência nos mostra que, de fato, estamos reconhecendo muitas palavras. Observe que um complicador que poderia surgir é a presença de espaços entre as palavras: como decidir a que palavra pertence cada espaço? Pois, se um padrão usa os dois

espaços adjacentes a uma palavra, o padrão seguinte não poderá usá-los. De forma interessante, nosso algoritmo "escolhe" naturalmente designar apenas um espaço por palavra, na maior parte dos casos à direita. Uma outra solução seria considerar que os espaços não fazem parte dos padrões; mas, nesse caso, teria que existir um padrão para o símbolo de espaço, o que não faria muito sentido, já que os espaços não existem sozinhos. Entretanto, a porcentagem de padrões que correspondem a palavras contidas no texto é bastante baixo: após treinar com os 100 mil primeiros caracteres de "David Copperfield" e de "Os Maias", obtivemos aproximadamente 6% para ambos.[4] Parte disso é explicada pela existência de muitos padrões contendo duas palavras que costumam aparecer frequentemente juntas. De certa forma, até faz sentido considerarmos duplas como "Mr. Copperfield" ou "have to", por exemplo, como sendo de fato entidades próprias. Essa ideia é mais convincente se pensarmos na linguagem falada, onde não há um equivalente aos espaços entre cada palavra, apenas um fluxo contínuo de som por frase. A divisão formal entre algumas palavras é muitas vezes uma mera convenção histórica de como organizar e grafar os sons que emitimos ao usarmos uma dada língua.[5] Isso ilustra parte da dificuldade de se avaliar a qualidade dos padrões formados, mesmo quando utilizamos um domínio sobre o qual supostamente sabemos bastante.

Embora estejamos aprendendo muitas palavras, a quantidade de padrões aprendidos é muito maior que o número real de palavras distintas presentes no texto (apenas 6% do total de padrões correspondem a palavras exatas, tanto para "David Copperfield" quanto para "Os Maias"). Nossa rede deveria ser de alguma forma mais econômica. (Poderíamos nos indagar se isso não se resolveria com uma quantidade maior de treinamento. Porém, se ampliarmos o trecho usado no treinamento para os primeiros 500 mil caracteres de "David Copperfield" e "Os Maias", diminuímos esse valor para 5% e 4%, respectivamente, de um total de 69548 e 64274 padrões, respectivamente.)

Por outro lado, podemos calcular a porcentagem de palavras aprendidas, independentemente do número total de nós na rede. Para tal, uma ideia é percorrer o texto do início ao fim e contar o número de vezes que encontramos uma palavra para a qual temos um padrão equivalente. Dividindo-se esse número pelo total de palavras no texto, temos uma medida aproximada da fração de palavras aprendidas

[4]Para realizar esse cálculo, consideramos que um padrão P corresponde a uma palavra exata w se P é idêntico a w a menos de espaços à esquerda ou à direita, tomando o cuidado de não computar duas correspondências caso tenhamos dois padrões que difiram apenas com relação à posição do espaço (por exemplo, se tivermos dois padrões "`-urso`" e "`urso-`", contamos apenas um como correspondente à palavra "urso"). Dessa forma, evitamos que palavras que eventualmente tenham dado origem a mais de um padrão contribuam para aumentar excessivamente o percentual de correspondências.

[5]Conforme contado por [46], muitas preposições e verbos auxiliares originalmente representados por palavras próprias foram sendo, através dos séculos, aglutinados aos verbos principais dando origem a declinações.

Tabela 3.1: Dez padrões de maior peso presentes na rede em diferentes momentos (medidos em número de caracteres lidos) nos treinamentos feitos com "David Copperfield" e com "Os Maias". O peso de cada padrão para um dado momento é definido como o tempo restante de vida do seu nó na rede dividido por seu valor de k. Foram usados $k_{\min} = 20$ e tamanho de janela igual a 15 para ambos os treinamentos.

Caracteres lidos	David Copperfield	Os Maias
5000	to-, u, ng-, t-, and-, y-, b, as-, in-, ne	or-, l, s-, va-, ci, de-, e-, des, a-, pa
10000	to-, and-, in-, u, it-, do, su, b, but-, si	de-, carlos-, da-, re, va-, des, se, s-, ver, tra
25000	very-, said-miss-betse, ly-, mr-copperfield-, in-, said-my-mother-, one-, for-, but-, ma	que, lhe-, o-seu-, no-, ndo-, como-, vi, para-, res-, bu

que naturalmente dá um peso maior para as palavras mais frequentes (já que muitas palavras se repetem ao longo texto). Em outras palavras, estamos incorporando o fato de que gostaríamos que fossem aprendidos os padrões relativos às palavras mais frequentes no texto. Os valores encontrados para"David Copperfield" e "Os Maias" foram, respectivamente, 77% e 80%, o que parece ser um forte indicativo de que estamos no caminho certo.

Outra análise útil pode ser feita comparando-se a distribuição do tamanho das palavras no texto original com a do tamanho dos padrões criados. Vemos que há um excesso de padrões tanto de tamanhos grandes quanto de tamanhos pequenos (Figura 3.2(a)). Uma inspeção cuidadosa dos padrões que não correspondem a palavras exatas revela que muitos deles são formados por uma palavra acompanhada de parte de outra palavra, bem como muitos padrões equivalente a pedaços de palavras recebendo altos valores de k. Vamos lançar mão de duas ideias para tentar remediar esses problemas.

3.4.1 Aperfeiçoamentos

Podemos interpretar ambos os problemas citados acima como o resultado de valores de k inadequados: no primeiro caso, padrões complexos ("grandes") inevitavelmente são formados apenas após algum tempo considerável de processamento (necessário para a formação de todos os pares que vão dar origem ao padrão), logo recebem um k alto; no segundo, padrões pequenos que ainda não chegaram a incluir uma palavra real mas que aparecem tardiamente no texto também acabam com um k mais alto do que o necessário. É verdade que esses últimos rapidamente param de ser ativados (assim que o padrão correspondendo à palavra completa é descoberto), porém seu

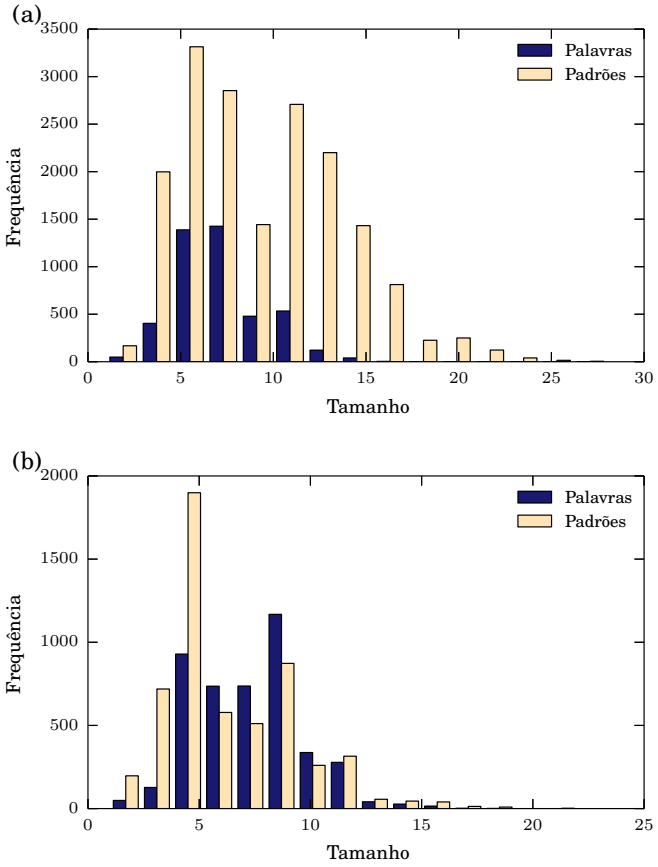


Figura 3.2: Distribuição do tamanho das palavras e do tamanho dos padrões criados tendo como base os primeiros 100 mil caracteres do livro "Os Maias" , antes (a) e depois (b) dos aperfeiçoamentos feitos no algoritmo.

alto k faz com que eles levem muito tempo a ser excluídos.

Com relação ao último problema, note que, após algum tempo de aprendizado, qualquer palavra inédita que surge no texto pode ser "montada" pelo algoritmo a partir de sub-padrões.[6] Nesse momento, se o tempo decorrido já for grande, esses sub-padrões receberão valores de k mais altos que o desejável. Um remédio para isso seria, de alguma forma, encontrar valores mais "adequados" para os sub-padrões. Ora, é fácil notar que uma sílaba só pode ter frequencia igual ou maior que qualquer palavra de que ela faz parte; logo, o mesmo vale para um sub-padrão. Assim, um valor lógico para o k de um sub-padrão seria o k do padrão de que ele faz parte, independente do tempo já decorrido. E, no caso de um sub-padrão ser parte de diversos padrões ao mesmo tempo, faz sentido que tomemos o k do menor desses padrões, o qual teoricamente seria o mais parecido com o próprio sub-padrão.

Na prática, encontramos um sub-padrão após percorrer nossa árvore de sufixos.

[6] A não ser quando surge algum símbolo inédito: nesse caso, ele é tratado como um padrão inédito, tal como é feito no começo do treinamento. O k escolhido para esse tipo de padrão é o $k_{\min}$, para que esse padrão "temporário" (já que ele imediatamente forma algum par que deve ser mais significativo que o símbolo sozinho) não seja retido por tempo demais na rede.

Logo, para descobrir o menor padrão completo de que ele faz parte, precisamos continuar percorrendo a árvore a partir do ponto em que a busca terminou até encontrarmos um sinal de final de palavra. No entanto, não temos como saber se encontramos um padrão verdadeiro ou apenas um sufixo. Para resolver esse impasse, acrescentamos um detalhe no procedimento de inserção na árvore: ao invés de anotarmos um marcador \$ ao final de cada sufixo, mantemos uma lista na qual anotamos o padrão completo do qual aquele sufixo faz parte.

No primeiro problema, estamos basicamente criando padrões mais complexos do que o desejado, o que é consequência inevitável de nossa estratégia de ir formando novos padrões a partir da concatenação de padrões adjacentes. Não temos como saber quando parar de fazer isso, logo, em algum momento, ultrapassamos o limite que separa um padrão significativo de um demasiadamente grande. Em resumo: queremos que os padrões sejam os maiores possíveis, mas não grandes demais. Mas como definir "grande demais"? Uma maneira de enxergar isso é: se um determinado padrão P possui sub-padrões sendo ativados mais frequentemente do que ele próprio, isso é um sinal de que P é maior do que deveria. Podemos pensar: mas e quanto às sílabas em uma palavra? Elas sempre irão aparecer mais no texto do que as palavras completas. Mas lembre-se de que um padrão só é ativado quando ele é detectado inteiramente; depois que já aprendemos a maioria das palavras relevantes para o texto sendo lido, não iremos detectar mais as sílabas: esses padrões mais simples ter-se-ão tornado obsoletos. Tomando proveito disso, adotamos a seguinte estratégia: toda vez que o algoritmo fizer uma correspondência com um sub-padrão p (ao invés de com um padrão inteiro), penalizamos o tempo de vida restante do padrão P no qual p está contido.

Como fazer essa penalização? De forma a economizar no número de parâmetros necessários para nosso algoritmo, escolhemos diminuir o tempo de vida restante de P em k unidades, onde k é a constante relativa ao nó que representa P. Mas e se p estiver contido em mais de um padrão? Nesse caso, que é de fato comum, penalizamos todos os padrões que contêm p. Isso pode ser descoberto em nossa árvore de sufixos: para isso basta continuar percorrendo-a do ponto em que busca havia terminado até encontrar o marcador de final de sufixo, que agora é uma lista com todos os padrões completos que contêm aquele sufixo. Porém, ao invés de parar no primeiro final de sufixo encontrado, fazemos uma busca em largura para encontrar todos os finais de sufixo alcançáveis a partir daquele sub-padrão. Dessa forma, encontramos facilmente todos os padrões que o contêm para que sejam penalizados.

Essas modificações estão presentes no Algoritmo 3.2. Os resultados obtidos podem ser comparados com os anteriores na Figura 3.2(b), onde observa-se uma distribuição muito mais parelha do número de padrões para cada faixa de tamanho com

Algoritmo 3.2 Algoritmo para aprendizado de padrões após aperfeiçoamentos

Input: Rede, texto, i
  Rede.t $\leftarrow$ Rede.t + 1
  ATUALIZAR_NOS(Rede)
  janela $\leftarrow$ texto$[i : i +$ Rede.tamanho_max_janela$]$
  padrao $\leftarrow$ *ENCONTRAR_PADRÃO*(janela, Rede.suffix_trie)
  $i \leftarrow i +$ padrao.**size**
  if *INÉDITO?*(padrao) **then**
    $k \leftarrow$ Rede.$k_{\min}$
    ATIVAR_NO(Rede, padrao, k)
  else if *É_PADRAO?*(padrao) **then**
    ATIVAR_NO(Rede, padrao)
  else
    padrao_inteiro $\leftarrow$ *ENCONTRAR_PADRAO_INTEIRO*(Rede.suffix_trie, padrao)
    // *acha o menor padrão inteiro que contenha* padrao
    $k \leftarrow$ padrao_inteiro.k // *usa o* k *de* padrao_inteiro
    ATIVAR_NO(Rede, padrao, k)
    padroes_maiores $\leftarrow$ *BUSCAR_PADROES_MAIORES*(Rede.suffix_trie, padrao)
    // *faz a busca em largura para achar todos os padrões contendo* padrao
    for all $P \in$ padroes_maiores **do**
      PENALIZAR(Rede, P) // *penaliza cada padrão maior*
    end for
  end if
  if Rede.ultimo_padrao_visto $\neq$ **null then**
    $k \leftarrow$ Rede.t
    FORMAR_PAR(Rede.ultimo_padrao_visto, padrao, k)
  end if
  Rede.ultimo_padrao_visto $\leftarrow$ padrao
  return i

relação ao número de palavras reais de mesmos tamanhos. O percentual de padrões na rede correspondendo a palavras exatas após um treinamento com os primeiros 500 mil caracteres subiu a 8% para ambos os livros, cada um dando origem a cerca de 22 mil nós (aproximadamente um terço das quantidades anteriores). Se olharmos agora para os novos valores da porcentagem de palavras no texto que correspondem a padrões exatos, passamos de 80% para 74% com "Os Maias" e de 77% para 81% com "David Copperfield". Ou seja, eles se mantiveram aproximadamente estáveis, o que significa que conseguimos manter a fração de palavras aprendidas porém utilizando para isso uma rede consideravelmente menor! Nosso algoritmo está, em teoria, pronto para ser aplicado a outros domínios.[7]

[7]Embora 8% possa ainda parecer um valor muito baixo, é importante ressaltar o fato de que nosso objetivo maior não é aprender palavras exatas, mas sim os elementos mais estatisticamente relevantes, que não necessariamente são as palavras. Ainda assim, o algoritmo está conseguindo aprender a maioria delas, então ele parece estar num bom caminho.

3.5 Aprendendo padrões musicais

Seguindo as escolhas feitas na Seção 3.3.1, um domínio sequencial que mantém as restrição de ausência de ruído é o de música em notação simbólica. Entretanto, antes de poder testar o aprendizado de padrões nesse domínio, precisamos estabelecer uma forma de representar notas musicais que permita gerar uma entrada adequada ao nosso algoritmo. Na notação musical ocidental tradicionalmente usada em partituras, cada nota tem uma altura dada por sua posição no pentagrama (nome da nota e sua oitava, ex. `ré3`, `dó1`) e uma duração que é dada por sua figura (para um compasso 4/4, uma semínima equivale a um tempo, uma colcheia equivale a meio tempo, etc.).

Uma opção seria designar um par de símbolos para cada nota, o primeiro representando a altura e o segundo sua duração. Entretanto, se queremos que a sequência de entrada se assemelhe o máximo possível ao tipo de sinal que o córtex auditivo recebe, em primeiro lugar notas mais longas devem produzir um sinal que dure mais tempo do que notas curtas. Além disso, nossa percepção musical é em grande parte independente da altura absoluta em que as notas estão: reconhecemos uma música independentemente do tom em que ela é tocada (isso é facilmente percebido ao reconhecermos um cantiga de roda cantada por um homem adulto ou por uma criança: embora cada um cante usando registros bem diferentes, não há qualquer dificuldade em reconhecer a melodia).[8] Gostaríamos, ainda, que a notação usada fosse capaz de permitir a representação de mais de uma nota soando simultaneamente, fato que ocorre em qualquer música onde há mais de uma voz ou instrumento, ou mesmo quando só há um instrumento mas ele pode emitir mais de uma nota simultaneamente (como o piano, ou o violão).

Uma notação que atende a esses requisitos é a usada em [47]. Nela, a cada altura corresponde uma lista de segmentos de reta, cada qual referente a uma nota emitida naquela altura, e de comprimento proporcional ao seu tempo de duração. Vamos usar uma versão adaptada dessa notação. Nela, utilizaremos uma matriz onde cada linha corresponde a uma altura e as colunas particionam o tempo total da música em partes iguais (cada duas colunas equivalem ao máximo divisor comum entre todas as durações presentes na música). As notas são indicadas nesta matriz com 1s, e as demais posições são preenchidas com 0s. Resta ainda um detalhe: se uma mesma nota for executada duas ou mais vezes seguidamente, como saberemos onde uma termina e a outra começa? Afinal, em nossa matriz veremos apenas uma sequência

[8] Mesmo as pessoas que possuem o chamado "ouvido absoluto", isto é, que ao ouvirem uma nota sabem dizer sua altura exata, não perdem essa capacidade de "relativizar" o tom em que a música é executada. O que essas pessoas ganham é uma habilidade extra, por assim dizer, que as permite, por exemplo, dispensar o uso de uma referência externa ou diapasão para cantar uma música em sua altura original ou para reconhecer quando um instrumento está desafinado.

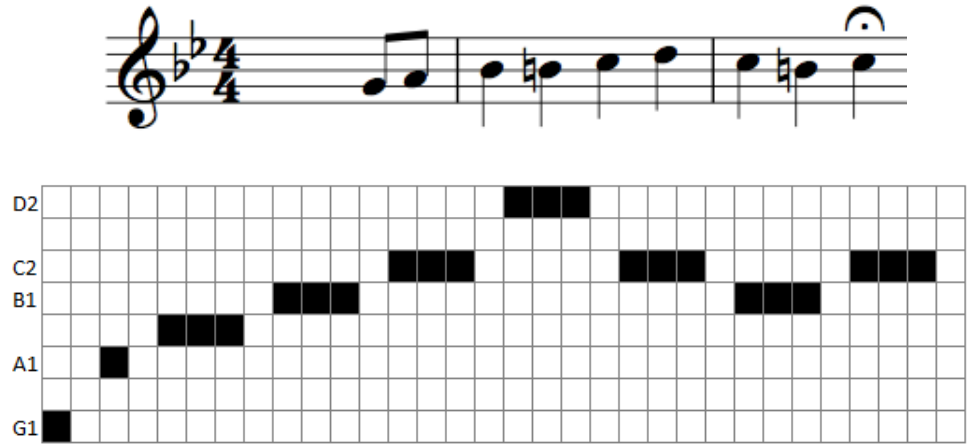


Figura 3.3: Primeiras notas da linha do soprano em uma cantata de Bach (acima) e sua notação correspondente em forma de matriz (abaixo). Observe que as colcheias, por serem as notas de menor duração na partitura em questão, utilizam duas colunas cada (uma preenchida e uma em branco, esta última marcando o fim da nota). Já as semínimas utilizam o dobro, ou seja, quatro colunas (três preenchidas e uma em branco). (Seguindo a notação que utiliza letras do alfabeto para cada nota, C corresponde a dó, D a ré, G a sol, A a lá e B a si.)

de 1s consecutivos em uma mesma linha. Para que essa distinção seja possível, a última coluna de cada nota é preenchida com um 0 (daí a necessidade de usarmos duas colunas para cada unidade de tempo).

Para ilustrar, digamos que cada coluna equivale a um quarto de tempo, e queremos escrever dois `dó3` seguidos, cada um com duração de um quarto de tempo. Cada nota ocupará duas colunas, a primeira com um 1 e a segunda com um 0: $[1, 0, 1, 0]$. Se ao invés disso tivéssemos apenas um `dó3` de duração meio tempo, teríamos $[1, 1, 1, 0]$. Note que não é possível confundir esses 0s com pausas, já que a menor pausa ocupará no mínimo duas colunas seguidas com 0s. A Figura 3.3 mostra um trecho de uma partitura de uma cantata de J. S. Bach e sua representação em forma de matriz.

Para que nosso algoritmo consiga lidar com esse novo tipo de entrada, algumas adaptações são necessárias. Primeiro, vamos nos limitar a um caso mais simples que é o das músicas ditas monofônicas, isto é, que só possuem uma nota soando a cada instante. Na matriz isso corresponde a termos nó máximo um 1 em cada coluna. E, ao invés de mantermos uma matriz contendo em sua maioria 0s, podemos guardar simplesmente uma lista das posições que na matriz forem iguais a 1 (coordenadas na forma (índice da linha, índice da coluna)). Nosso algoritmo recebe, uma a uma, essas posições e vai formando padrões. Entretanto, para podermos relativizar cada nota, toda vez que o algoritmo começa a tentar formar um padrão (ou seja, a cada janela processada) ele guarda os índices da primeira posição (x_0, y_0) recebida e a trata

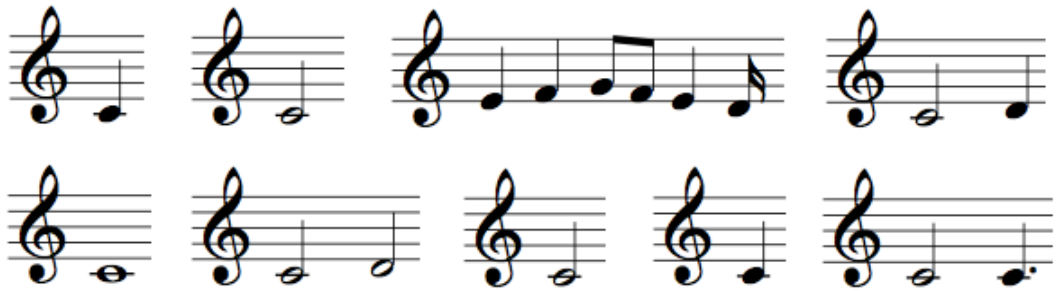


Figura 3.4: Padrões mais ativados após aprendizado de dez corais de Bach (BWV 250–259) (acima) e após aprendizado com os cinco primeiros madrigais do Quinto Livro de Monteverdi (abaixo). Para manter a restrição de monofonia, utilizou-se apenas a linha do soprano em ambos os treinamentos. O tom de dó maior usado nos trechos de partituras foi escolhido apenas para auxiliar na notação, uma vez que todos os padrões aprendidos são relativos.

como sendo $(0, 0)$. As posições seguintes $\{(x_1, y_1), (x_2, y_2), (x_3, y_3) \ldots\}$ passam a ser tomadas relativas a essa primeira: $\{(x_1 - x_0, y_1 - y_0), (x_2 - x_0, y_2 - y_0) \ldots\}$. Por exemplo: digamos que a primeira posição recebida é $(5, 23)$, que é tratada como $(0, 0)$. Se as posições seguintes forem $(6, 24)$ e $(4, 25)$, serão interpretadas como $(1, 1)$ e $(-1, 2)$, respectivamente.

A Figura 3.4 mostra os padrões de maior peso encontrados ao treinarmos o algoritmo com dez corais de J. S. Bach e com cinco madrigais de Claudio Monteverdi. No entanto, como saber, nesse caso, se nosso algoritmo foi bem sucedido? Uma possibilidade é ouvir cada um desses padrões e ver eles soam "representativos", tarefa extremamente subjetiva. Mas repare que o fato de que os padrões mais ativados no treinamento com Bach serem diferentes dos ativados com Monteverdi já é um sinal de que esses padrões sejam minimamente úteis, dado que cada compositor tem um estilo diferente (barroco *vs.* renascentista). Portanto, seria interessante, agora, buscar aplicações que utilizassem esses padrões descobertos para produzir resultados úteis. Só assim teremos uma forma concreta de avaliar a qualidade dos padrões aprendidos.

Capítulo 4

Compressão utilizando padrões organizados em caminhos direcionados

4.1 Compressão baseada em modelos probabilísticos

4.1.1 Usando padrões para prever sequências

Uma vez que temos as unidades básicas de formação de um dado domínio ou linguagem, uma possibilidade de uso para elas que surge naturalmente é a de completar sequências em tal linguagem onde haja presença de ruído na forma de símbolos faltantes. Por exemplo: se a linguagem for o inglês, é fácil prever que o símbolo desconhecido em "`whatev?r`" é a letra "`e`", ou que o próximo símbolo depois de "`wate...`" tem alta probabilidade de ser "`r`". Para tanto, bastaria procurar dentre os padrões aprendidos para a linguagem em questão aquele—ou aqueles, dependendo do caso—que se encaixariam dado o segmento de texto disponível. A maioria de nós conhece desde criança o jogo da forca, onde precisamos descobrir a palavra oculta adivinhando as letras que a constituem. Quanto maior nosso conhecimento sobre o vocabulário, mais rapidamente conseguimos acertar a palavra. Agora, se expandirmos esse raciocínio para uma frase, ou um trecho maior de um texto, a brincadeira torna-se mais interessante ainda, pois passamos a poder utilizar a informação das palavras anteriores para ajudar a inferir as letras e palavras que estão por vir.

A capacidade de comprimir uma sequência de símbolos está intimamente relacionada à facilidade com que ela pode ser prevista. Afinal, intuitivamente, quanto melhor podemos adivinhar o que vem a seguir, menor a quantidade de informação necessária para ajudar-nos a decidir o símbolo seguinte. Uma maneira bastante sim-

ples de usar essa ideia para comprimir texto é, usando a frequência observada de cada letra do alfabeto, criar um código binário para cada uma delas, designando os menores códigos para as letras mais frequentes. Isso é usado no código Morse, por exemplo, onde as letras mais frequentes do inglês, "`e`" e "`t`", são representadas pelos códigos mais simples ("·" e "−", respectivamente). É fácil perceber, portanto, que quanto melhor forem aprendidas as regularidades da linguagem em que um texto está escrito, melhor poderá ser, em princípio, a compressão atingida.

4.1.2 Modelos probabilísticos e codificação aritmética

A ideia de comprimir sequências prevendo-se o que virá a seguir não apenas é atraente como vem sendo usada com sucesso há muito anos [48]. Para tanto, precisamos de um modelo para a fonte que gerou a sequência a ser comprimida: quanto mais esse modelo for parecido com a fonte, melhores serão suas previsões. Mais do que adivinhar o próximo símbolo, para que um modelo possa ser usado num esquema de compressão ele precisa atribuir uma probabilidade para cada símbolo possível. Isto é, se lêssemos a sequência "`carr`", poderíamos dar probabilidades altas para "`o`", "`i`", "`a`" e "`e`", por exemplo, e baixas para as demais letras do alfabeto. De forma análoga ao que foi exemplificado com o código Morse, queremos traduzir essas probabilidades em um esquema de codificação eficiente, de modo que quanto maiores forem as probabilidades atribuídas aos símbolos presentes na sequência, menor será o número de bits necessário para transformá-la em um código binário.

A solução está na chamada "codificação aritmética" [49]. Nela, a modelagem probabilística é claramente separada da tarefa de codificação, o que, segundo [48], foi um dos principais avanços na teoria de compressão de dados. Temos um modelo que gera, para cada ponto da sequência sendo codificada, uma distribuição das probabilidades de cada símbolo do alfabeto ser encontrado ali. O codificador, por sua vez, munido dessas probabilidades e da sequência original, constrói uma sequência binária.[1] Esta pode ser lida por um decodificador munido do mesmo modelo usado na compressão para recuperar um a um os símbolos da sequência original.

A etapa de codificação baseia-se no seguinte princípio: observe que uma sequência binária pode ser interpretada como definindo um intervalo na reta real entre 0 e 1. Partimos do intervalo $[0, 1)$. Se o dividirmos ao meio, podemos codificar o intervalo $[0.0, 0.5)$ como `0` e o intervalo complementar, $[0.5, 1.0)$, como `1`. Podemos continuar dividindo esses intervalos ao meio, acrescentando `0` ou `1` à codificação quando tomarmos a primeira ou segunda metades, respectivamente. Assim, o intervalo $[0.00, 0.25)$ é codificado por `00`, e $[0.25, 0.50)$ por `01`. Da mesma forma, a

[1] Essa sequência é praticamente ótima, no sentido de que garantidamente difere no máximo em dois bits da quantidade total de informação de Shannon da sequência sendo codificada, dadas as probabilidades geradas pelo modelo [50].

sequência `10` codifica o intervalo $[0.50, 0.75)$ e `11` o intervalo $[0.75, 1.00)$. (Note que, na verdade, é possível escolher qualquer base numérica para o código; a binária é a escolha mais natural pois permite representação direta em computadores digitais.)

Ora, se temos um modelo probabilístico das letras em uma sequência $\{x_1, x_2, \ldots, x_n\}$ que use um alfabeto A, então podemos usá-lo para gerar cada probabilidade $P(x_i = a | \{x_1, x_2, \ldots, x_{i-1}\})$, onde $1 < i \leq n$ e $a \in A$, isto é, para cada posição na sequñcia temos a probabilidade de cada letra aparecer ali condicionada às letras que já foram lidas anteriormente. Cada probabilidade dessas define um intervalo contido entre 0 e 1. Por exemplo, se temos $A = \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}\}$ e as letras são i.i.d., então inicialmente "`a`", "`b`", "`c`" e "`d`" correspondem respectivamente aos intervalos $[0.00, 0.25)$, $[0.25, 0.50)$, $[0.50, 0.75)$ e $[0.75, 1.00)$. Na sequência "`bbc`", para encontrar o intervalo correspondente à probabilidade de encontrar o segundo "`b`", vamos dividir o intervalo $[0.25, 0.50)$ (definido pelo primeiro "`b`") em quatro partes iguais, cada qual correspondendo a uma das letras de A; com isso, o intervalo correspondente a "`bb`" é $[0.3125, 0.3750)$, ou seja, o segundo quarto do intervalo anterior, $[0.25, 0.50)$. Procedendo analogamente, o intervalo correspondente à probabilidade da sequência "`bbc`" será $[0.343750, 0.359375)$ (terceiro quarto do intervalo anterior). Codificar este último intervalo equivale a codificar a mensagem "`bbc`" inteira, e para isso basta encontrar o intervalo binário, conforme mostrado no parágrafo anterior, que esteja totalmente contido nesse intervalo.

A sequência binária resultante pode ser facilmente decodificada se estivermos munidos do mesmo modelo probabilístico; para isso, uma vez mais partimos do intervalo $[0, 1)$ e o dividimos de acordo com as probabilidades dadas pelo modelo para cada letra de A. Lemos um a um os dígitos da sequência binária até que eles especifiquem um intervalo totalmente contido em um dos intervalos relativos às letras de A, digamos a. Com isso descobrimos a primeira letra da mensagem original. Em seguida, dividimos o intervalo correspondente a a de acordo com as probabilidades dadas pelo modelo, condicionadas ao trecho já decodificado (a). E assim por diante.

Outro detalhe é que é necessário acrescentar um símbolo especial ao alfabeto representando o final da sequência, ou "fim de arquivo". Esse e outros pormenores da implementação são cuidadosamente descritos em [49] e em [50].

4.1.3 Em busca de um modelo probabilístico

Muitos trabalhos se dedicaram a buscar um algoritmo que conseguisse construir, a partir de uma amostra de texto, um modelo probabilístico para previsão de suas letras e subsequente compressão via codificação aritmética. Uma das estratégias de maior sucesso até hoje é a *Prediction by Partial Matching* (previsão por correspondências parciais), ou PPM, desenvolvida por [51]. A beleza desse método está no

fato de que as probabilidades vão sendo aprendidas e a codificação gerada à medida que a entrada é lida. Para descobri-las, utiliza-se um modelo de Markov onde a probabilidade de cada letra é condicionada a um certo número m de letras que a precedem. Logo, a ordem desse modelo de Markov é dada por m. O valor de m, entretanto, não é fixo: ele possui um máximo arbitrado mas vai sendo variado de acordo com a necessidade.

O método PPM engloba hoje em dia uma família de algoritmos, já que ao longo dos anos passou por diversas modificações e aperfeiçoamentos [52, 53]. Outra observação importante acerca desse método é que ele se propõe a codificar qualquer sequência de símbolos, e não apenas texto. Por exemplo, um conjunto da dados usado tradicionalmente para avaliação de um desses algoritmos baseados em codificação aritmética é a coleção Calgary de arquivos (*Calgary corpus*, em inglês), que inclui dois romances, mas também artigos científicos, um índice bibliográfico, relatórios numéricos, uma imagem em escala de cinza, códigos em linguagem de programação e até mesmo arquivos binários, entre outros [48].

Outras duas abordagens de sucesso na literatura são o *Block-sorting* [54] e o *Sequence Memoizer* [55]. O primeiro, ao contrário do PPM, precisa percorrer todo o texto e pré-processá-lo antes de realizar a compressão. Esse pré-processamento separa o texto em diversos blocos, no quais é usado um esquema de reordenação das letras de forma a dispor letras idênticas próximas umas das outras, tornando-as mais previsíveis (essa reordenação pode ser desfeita, claro, durante a decodificação). Sua principal vantagem é a rapidez com que a codificação é feita, comparável à de métodos que não usam um modelo probabilístico, tais como o Lempel-Ziv [56] (amplamente utilizado, por exemplo, nos computadores pessoais através do comando `gzip`).

O segundo baseia-se em modelar o texto como um processo de Pitman-Yor hierárquico, cujos parâmetros vão sendo aprendidos progressivamente à medida que o texto é lido e codificado, utilizando para isso uma abordagem bayesiana. Ele é responsável por algumas da melhores compressões já obtidas até hoje. Os níveis de compressão alcançados pelos diferentes métodos são comparados na Tabela 4.1).

Seria interessante aproveitar nosso algoritmo de descoberta de padrões para, de alguma forma, gerar um modelo probabilístico que pudesse ser utilizado na compressão de texto. Entretanto, o que ele nos fornece são simplesmente os trechos que se repetem com frequência significativa em uma sequência. Isso não é informação suficiente para prever que palavra viria após outra, por exemplo, para o que teoricamente o PPM seria muito mais propício, pois é independente das palavras em si: só leva em conta sequências arbitrárias de caracteres.

Outra razão para não depender das palavras do texto para aprender as probabilidades é que, para poder comprimir outros domínios (tais como os do *Calgary*

Tabela 4.1: Métodos de compressão e suas respectivas taxas de compressão médias obtidas para o *Calgary corpus*. A numeração usada para o PPM refere-se a diferentes versões do método.

Método	bits/símbolo
PPM1 [51]	2.48
Block-sorting [54]	2.43
`gzip` [55, 56]	2.11
PPM2 [52]	2.04
PPM3 [53]	1.93
Sequence Memoizer [55]	1.89

corpus, citado acima), não teríamos conhecimento sobre quais seriam as entidades análogas a palavras. No entanto, é justamente a isso que se propôs nosso algoritmo! Logo, se pudermos pensar num jeito de aprender probabilidades eficazmente com base na sequência de palavras de um texto, talvez esse método também sirva para aprender as probabilidades de símbolos em qualquer outro domínio, resultando em um compressor universal. Mais que isso, teríamos um algoritmo de aprendizado universal de modelos.

Primeiramente, repare que, munidos apenas de uma lista com todos os padrões formados em nossa rede, já podemos prever o próximo símbolo em um texto. Para isso, basta procurar todos os padrões cujas primeiras letras correspondam ao final do segmento de texto já lido e verificar as frequências das letras seguintes. Por exemplo: se já lemos "`mai`" e conhecemos os seguintes padrões: "`mais`", "`maio`" e "`maionese`", faz sentido atribuir às letras "`s`" e "`o`" probabilidades 1/3 e 2/3, respectivamente (já que, nos padrões conhecidos, elas aparecem uma e duas vezes após "`mai`", respectivamente, e temos três padrões no total).

Entretanto, observe que, quanto mais nossos padrões se assemelharem a palavras exatas (e isso foi um de nossos objetivos no capítulo anterior), piores serão nossas previsões quando o segmento de texto lido terminar no final de uma palavra (como em "`nunca-`", por exemplo). Uma forma de contornar esse problema seria armazenar, de alguma forma, a informação de como nossos padrões costumam suceder uns aos outros em um texto ou sequência.

4.2 Memorizando sequências de padrões

Uma das concepções mais tradicionais para a modelagem de eventos ocorrendo em uma sequência é o processo de Markov. Nele, temos uma sequência de observações cuja probabilidade é condicionada somente às n observações anteriores, onde n é a ordem do modelo. Embora isso possa soar pouco realista, inúmeros fenômenos costumam ser satisfatoriamente modelados por processos de Markov, até mesmo

para $n = 1$, ainda que de forma aproximada (um exemplo clássico de aplicação é no reconhecimento de fala [57]). Tendo isso em mente, podemos pensar que, dada a sequência de palavras em um texto, é possível prever cada um delas a partir de uma distribuição de probabilidades de palavras condicionadas à palavra anterior. Para isso, bastaria anotar todos os pares de palavras consecutivas ocorrendo em um grande corpo de texto e, a partir deles, calcular, para cada palavra w ocorrida no texto, as probabilidades de cada uma das palavras que ocorreram após w, dado o número de vezes que cada uma delas de fato ocorreu após w. Podemos criar, portanto, um dicionário D relacionando cada palavra a uma distribuição de probabilidades para a palavra seguinte.

No entanto, para compressão de texto precisamos de um modelo que nos forneça as probabilidades de letras, e não de palavras. Para isso, para cada palavra w em D, ao invés de guardamos as probabilidades de cada palavra que sucede w, guardarmos as probabilidades de cada primeira letra das palavras que sucedem w.

Isso já é suficiente para tornar nossas previsões muito mais sofisticadas. Para começar, montamos um dicionário D contendo os padrões aprendidos por nosso algoritmo. Agora, suponha que estejamos prevendo uma a uma as letras de um certo texto T. Vamos chamar o segmento de T que já foi lido de T'. Para prever a próxima letra, primeiramente usamos a mesma estratégia explicada ao final da seção anterior—verificamos se o final de T' corresponde ao início de algum dos padrões aprendidos. Caso contrário, é provável que T' tenha chegado ao final de um padrão. Portanto, verificamos agora se há alguma correspondência exata de algum dos padrões em D com o final de T'. Em caso afirmativo, lemos as probabilidades da próxima letra diretamente em D.

Note que, a partir dessa ideia, é imediato ampliar nosso contexto para duas ou mais palavras (ou padrões) anteriores. Além da ideia intuitiva de que com isso estamos levando em conta mais informação para julgar melhor a continuação do texto, podemos ilustrar o ganho potencial de se aumentar o contexto com o seguinte exemplo: é quase impssível adivinhar o que vem após a palavra "de", em um texto em português; entretanto, se soubermos que a palavra "Rio" precedeu "de", então podemos prever que a próxima palavra deverá ser "Janeiro" com bastante segurança.

4.3 Implementação

Para tanto, uma estrutura mais indicada que o dicionário seria uma *trie* tradicional, ou "árvore de prefixos", com profundidade igual ao número m de palavras, ou padrões, que queremos utilizar como contexto, e na qual cada nó corresponde a um padrão reconhecido. Ela pode ser montada à medida que vamos reconhecendo cada padrão no texto, da seguinte maneira: a cada sequência de $m + 1$ padrões reconhe-

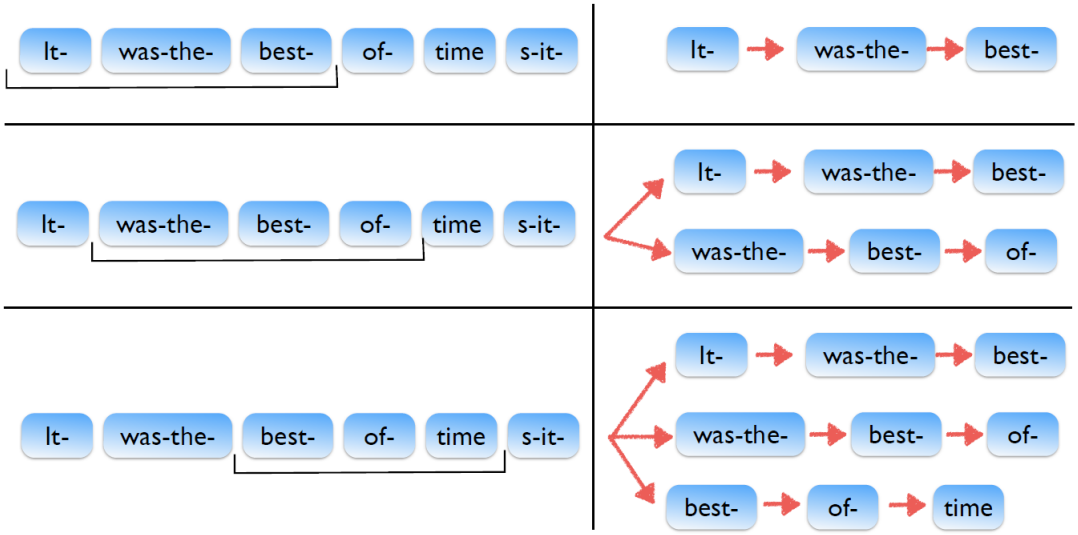


Figura 4.1: Primeiros passos da construção de uma árvore de prefixos de profundidade três para o livro "A Tale of Two Cities", de Charles Dickens. À esquerda temos cada sequência de três padrões sendo reconhecidos, e, à direita, a árvore resultante após a inserção dos mesmos. Os padrões utilizados aqui foram aprendidos após treinamento com o livro "David Copperfield".

cidos, adicionamos à árvore tal sequência partindo da raiz e reutilizando quaisquer padrões já presentes na estrutura.

Como exemplo, digamos que $m = 2$. Então, a cada três padrões reconhecidos, digamos **A**, **B** e **C**, partimos da raiz e vemos se **A** já é um dos sucessores. Se não for, conectamos o caminho **A** $\rightarrow$ **B** $\rightarrow$ **C** à raiz. Caso contrário, seguimos para o nó **A** e vemos se **B** é um dos sucessores. Se não for, adicionamos o caminho **B** $\rightarrow$ **C** a ele. Caso contrário, seguimos para o nó **B**, ao qual conectamos o nó **C**, caso ele ainda não seja um sucessor de **B**. Uma vez que a árvore foi construída, podemos calcular as probabilidades para o próximo símbolo dado um contexto **AB** percorrendo o caminho **A** $\rightarrow$ **B** e contando os primeiros símbolos de todos os sucessores de **B**. Esse processo é ilustrado na Figura 4.1.

Para tornar essas probabilidades mais precisas, nossa árvore de memória pode guardar, em cada nó, uma variável que conta o número de vezes que o caminho da raiz até ele já foi encontrado. Assim, quando fizermos uma busca pelos sucessores de um certo caminho C, ao invés de cada sucessor contribuir igualmente para o cálculo das probabilidades, ponderamos essa contribuição pelo número de vezes que C foi encontrado durante o treinamento. Um exemplo é dado na Figura 4.2.

Embora a árvore de memória torne rápida a busca pelo símbolo seguinte quando T' termina em um padrão exato, precisamos ainda de uma maneira eficiente de fazer o mesmo quando não há um padrão completo ao final de T'. Para isso, mantemos

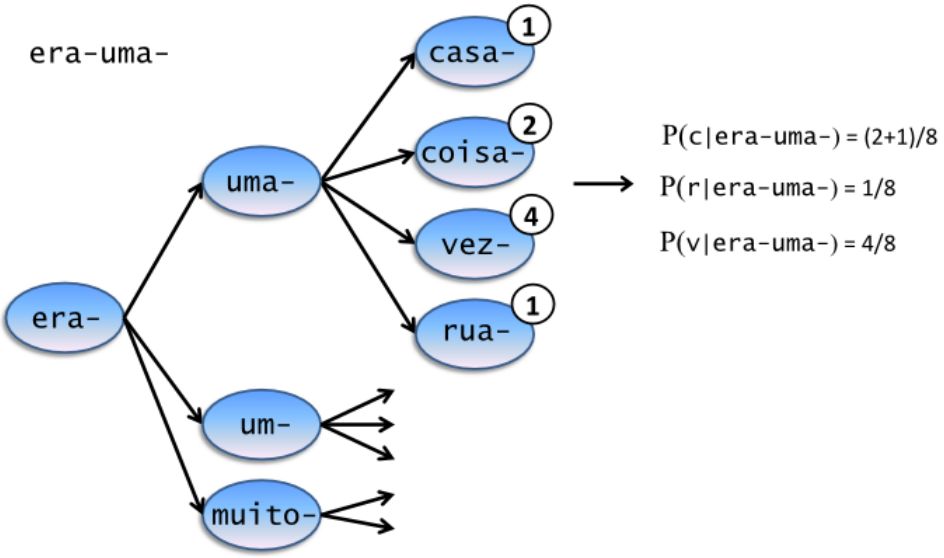


Figura 4.2: Previsão do próximo símbolo na sequência "`era-uma-`" utilizando a árvore de memória. Como "`era-`" e "`uma-`" são nós nessa árvore (são padrões conhecidos), podemos percorrê-la e construir uma distribuição de probabilidades para o símbolo seguinte baseada nos sucessores do caminho `era` → `uma-` e suas respectivas frequências.

uma lista L com todos os padrões presentes na árvore em ordem lexicográfica. Se queremos descobrir se o trecho ao final de T' (digamos, "`fra`"), corresponde ao começo de algum padrão, basta fazer uma busca binária em L por "`fra`", a qual nos retornará em tempo $O(\log |L|)$ a posição i do elemento mais próximo de "`fra`": percorremos então L, partindo de i em direção ao final da lista e, enquanto encontrarmos padrões começados por "`fra`", anotamos o símbolo seguinte (por exemplo, se encontramos "`fragile `", anotamos "`g`"). Ao final desse processo, basta dividir o número de vezes que cada símbolo foi visto pelo número total de padrões encontrados começando por "`fra`". Note que o número de comparações feitas para cada padrão que testamos ao percorrer a lista é $O(n)$, onde n é o tamanho do trecho t sendo buscado (no nosso exemplo com "`fra`", $n = 3$); fazemos n comparações somente quando o padrão sendo testado de fato começa por t, caso contrário interrompemos a verificação antes disso, ao detectar a primeira não-correspondência (ainda utilizando nosso exemplo, se o padrão sendo testado for "`fugue`", interrompemos a comparação logo na segunda letra. De fato, uma vez que L é ordenada, assim que isso acontece podemos parar de percorrê-la e calcular as probabilidades. Estas também serão ponderadas pelo número de vezes que cada padrão foi encontrado na etapa de construção da memória. O processo é ilustrado na Figura 4.3 com o exemplo dado ao final da Seção 4.1.3.

Algo que ainda não foi explicado, entretanto, é como escolher o trecho T' que

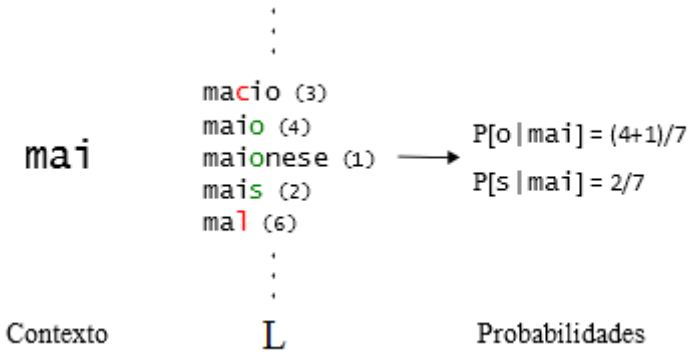


Figura 4.3: Previsão do próximo símbolo na sequência "`mai`" a partir de uma lista L contendo todos os padrões conhecidos. Como L está em ordem lexicográfica, podemos encontrar o primeiro padrão cujo início corresponde ao contexto atual em $O(\log |L|)$. No exemplo dado, encontramos primeiramente "`maio`", que foi visto quatro vezes no texto. Basta prosseguir em L para encontrar os demais padrões começando por "`mai`". Ao final desse processo, computamos as probabilidades de cada símbolo a partir do número de vezes que cada um deles ocorreu, ponderando cada ocorrência pela frequência do padrão.

é usado como contexto no decorrer do processo de codificação. Queremos sempre condicionar as probabilidades para o próximo símbolo utilizando o maior contexto possível. Logo, a princípio T' poderia ser simplesmente todo o trecho de T que já foi lido. Entretanto, o máximo de contexto possível de ser utilizado é determinado pela profundidade de nossa árvore de memória. Podemos pensar em T' como uma janela que termina sempre o mais à direita possível e começa num ponto que delimita o trecho de máxima correspondência com algum sub-padrão ou sequência de padrões na memória.

Nossa janela T' será delimitada à esquerda e à direita, respectivamente, por dois índices, i e j, que marcam posições no texto sendo codificado. Quando iniciamos o processo de codificação, ainda não lemos nenhum símbolo do texto T, logo T' está vazia ($i = j = 0$). Entretanto, após cada passo do algoritmo, estamos prevendo um símbolo novo de T, e podemos avançar a posição de j uma unidade à frente. Assim, a janela T' vai aumentando e com ela o contexto disponível como base para as previsões dos símbolos seguintes. Enquanto isso i permanece parado, até que T' já tenha se tornado grande demais para ainda poder ser usada como contexto (isso ocorre sempre que T' é maior que qualquer padrão ou caminho da memória existentes). Nesse caso, avançamos i até que T' se torne útil mais uma vez. Dessa forma, observe que a janela vai alternando expansões de sua extremidade direita com contrações de sua extremidade esquerda, de forma análoga a uma lagarta, enquanto percorre o texto do início ao fim.

Já podemos inspecionar o pseudo-código do algoritmo completo, que nos aju-

Algoritmo 4.1 Algoritmo de codificação

```
Input: texto, Memo, padroes_ordenados
  codigo ← [ ] // inicializa o código a ser retornado
  for n ← 0 to texto.size do
    proximo_simbolo ← texto[n] // próximo símbolo a ser previsto
    if i = 0 or i = j then
      lagarta ←'-' // se a lagarta estiver vazia, usa um espaço como contexto para
      o próximo símbolo
    else
      lagarta ← texto[i : j]
    end if
    probs, novo_i ← OBTER_PROBS(lagarta, Memo, padroes_ordenados)
    i ← novo_i // atualiza a extremidade esquerda da lagarta (i)
    // move a extremidade direita da lagarta (j) para frente
    j ← j + 1
    codigo ← ATUALIZAR_CODIGO(proximo_simbolo, probs, codigo) // codi-
    ficação aritmética
  end for
  return codigo
```

dará a fazer sentido dos detalhes adicionais envolvidos. Ele está dividido entre os Algoritmos 4.1 e 4.2.

Note que o Algoritmo 4.1 é apenas uma casca cuja função básica é administrar o contexto (personificado na imagem de uma lagarta) que é fornecido à sub-rotina 4.2 para gerar as probabilidades de cada símbolo. Com isso, a cada símbolo lido do texto sendo codificado obtém-se uma distribuição de probabilidades, que por sua vez é traduzida em um dígito binário pela codificação aritmética.

Na realidade, o cerne do algoritmo está em *OBTER_PROBS*, que produz o modelo probabilístico implícito nas estruturas da rede de padrões e da árvore de memória construídas. Cada vez que essa sub-rotina é chamada ela recebe uma lagarta, cuja partição em padrões é solicitada à rede que deu origem aos mesmos (isso é feito basicamente fornecendo-se a lagarta como entrada para o Algoritmo 3.1). Batizamos essa partição de "ladrilhos": cada um de seus elementos é um padrão, um sub-padrão ou então um trecho que não está contido em qualquer padrão da rede. A partir daí, vamos fazer sucessivas buscas a nossos padrões e à árvore de memória ("Memo", no pseudo-código) experimentando contextos diferentes para obter as probabilidades de cada símbolo de nosso alfabeto ser o seguinte no texto.

Começamos tentando usar todos os ladrilhos (máximo de contexto) para obter probabilidades. Essas probabilidades são obtidas conforme discutido acima: primeiro, verificamos se há um padrão que contenha o contexto atual; caso contrário, fazemos uma busca na memória por algum caminho que corresponda aos ladrilhos atuais. Sempre que a busca não retornar probabilidades ou ainda faltar algum

Algoritmo 4.2 Sub-rotina *OBTER_PROBS*

Input: lagarta, Memo, padroes_ordenados

```
ladrilhos ← FORMAR_LADRILHOS(lagarta, Rede) // particiona a lagarta em "ladrilhos" baseando-se nos padrões presentes na Rede (cada trecho q nao corresponde a nenhum padrão é tratado como um ladrilho)
lista_probs ← [ ] // inicializa uma lista de probabilidades
while not TODOS_OS_SIMBOLOS_ENCONTRADOS(lista_probs) do
  novas_probs ← BUSCAR_PROBS(ladrilhos, padroes_ordenados)
  if novas_probs ≠ false then
    achou_probs ← true
    lista_probs.INSERIR(novas_probs)
  end if
  if TODOS_OS_SIMBOLOS_ENCONTRADOS(lista_probs) = true then
    break
  end if
  // tenta encontrar o maior caminho na Memo seguindo os ladrilhos atuais
  novas_probs ← BUSCAR_PROBS(ladrilhos, Memo)
  if novas_probs ≠ false then
    achou_probs ← true
    lista_probs.INSERIR(novas_probs)
  end if
  // atualiza ladrilhos para a próxima iteração dentro do while-loop
  if TODOS_OS_SIMBOLOS_ENCONTRADOS(lista_probs) = true then
    break
  end if
  if ladrilhos.size = 1 then
    // se só sobrou um ladrilho
    if achou_probs = false then
      // se ainda não achou prob nenhuma, descarta a primeira letra dele e continua
      ladrilho ← ladrilho[1 :]]
      novo_i ← novo_i + 1 // atualiza a lagarta
    else
      break // se ja achou alguma, sai do while-loop
    end if
  else
    // se ainda há ladrilhos à frente, passa para o próximo
    ladrilhos ← ladrilhos[1 :]
    if achou_probs = false then
      novo_i ← lagarta.BUSCAR(ladrilhos[0]) // atualiza a lagarta se o ultimo ladrilho usado nao forneceu probs
    end if
  end if
end while
probs ← COMPUTAR_PROBS_FINAIS(lista_probs) // compila a distribuição total das probs de cada símbolo, ponderando pela qtdd de contexto usada
// inclui os símbolos faltantes
return probs, novo_i
```

símbolo, vamos diminuindo o contexto utilizado eliminando o ladrilho mais à esquerda (se isso ocorrer quando só resta um ladrilho, passamos a eliminar o símbolo mais à esquerda).

Em ambos os casos, isso nos diz que aquele símbolo ou ladrilho não nos dá mais qualquer informação útil para previsão do que vem após a lagarta atual. Logo, sempre que uma dessas eliminações é feita, atualizamos o índice i que marca a extremidade esquerda da lagarta (e por isso ele é retornado à rotina principal, junto com as probabilidades obtidas). Guardamos cada conjunto de probabilidades obtido dessa maneira em uma lista.

Após a lagarta atual ter sido explorada ao máximo, temos possivelmente uma lista de distribuições, cada uma obtida utilizando-se contextos progressivamente menores. O problema é que cada distribuição dessas costuma incluir apenas alguns símbolos, e para a codificação aritmética precisamos de uma probabilidade para cada símbolo do alfabeto (afinal, se a probabilidade para o próximo símbolo do texto for zero, nossa codificação passaria a ser infinitamente longa!). Para inferir uma distribuição única a partir dessa lista de distribuições, seguimos uma abordagem inspirada na que é usada pelo PPM.

Para cada distribuição presente na lista, estipulamos uma probabilidade de "escape", a qual será atribuída a todos os símbolos ausentes naquela distribuição. Essa probabilidade será dividida entre os símbolos faltantes de acordo com as demais distribuições presentes na lista, obtidas com contextos menores. A probabilidade de escape deve ter um valor pequeno, afinal contextos menores costumam gerar previsões piores. Seguimos a heurística usada no algoritmo PPM conhecida como "método C", que descrevemos a seguir.

Lembre-se de que as probabilidades são calculadas a partir de contagens do número de vezes que cada símbolo aparece na busca como possível candidato a ser o próximo no texto. Suponha que a busca tenha retornado a seguinte contagem: {a: 3, b: 5} e que nosso alfabeto seja $A = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$. Atribuímos uma contagem para "escape" igual ao número de símbolos presentes no conjunto retornado pela busca (nesse caso, 2). Com isso, teríamos como contagens finais: $\{\mathtt{a} : 3, \mathtt{b} : 5, \mathit{escape} : 2\}$ e, portanto, as seguintes probabilidades: $P(\mathtt{a}) = 3/10$, $P(\mathtt{b}) = 5/10$ e $P(\text{escape}) = 2/10$. Agora, como temos os símbolos "c" e "d" ausentes, vamos querer fazer uma nova busca usando um contexto menor. Os resultados dessa busca serão ponderados por essa probabilidade de escape. Fazemos isso sucessivamente, para todas as distribuições obtidas. Essa é a etapa realizada por *COMPUTAR_PROBS_FINAIS*. Note que pode ser que mesmo ao final dessa processo ainda fiquem faltando símbolos, caso em que usamos a última probabilidade de escape computada e a dividimos uniformemente entre todos os símbolos faltantes.

Nosso algoritmo está finalmente completo e pronto para ser testado. Um pano-

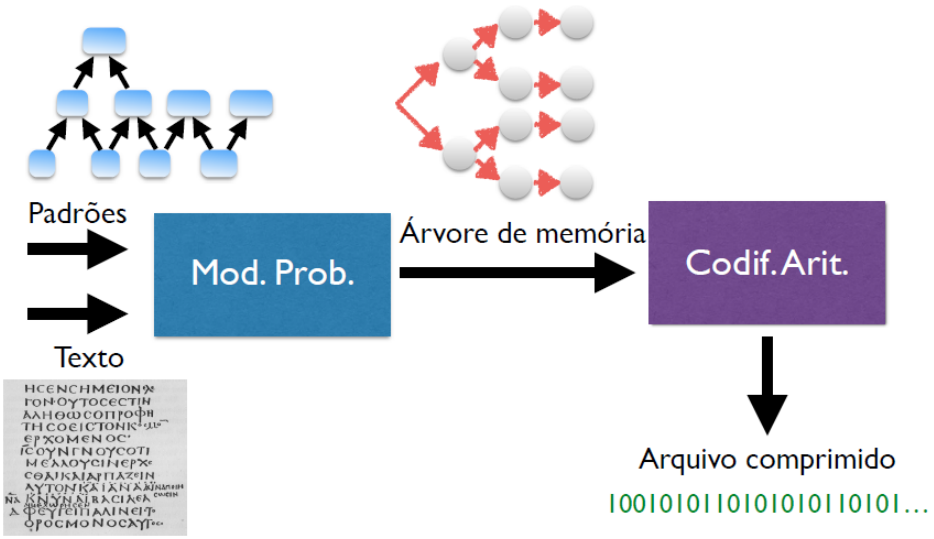


Figura 4.4: Panorama geral das etapas necessárias para se realizar a compressão: uma rede de padrões é construída a partir da leitura de um corpo de texto (ou de qualquer sequência de símbolos). Esses padrões são reorganizados em uma árvore de memória construída tendo como base o texto que se deseja comprimir. Ela representa o modelo probabilístico usado pela codificação aritmética para gerar um código binário correspondente ao texto original comprimido.

rama geral das etapas necessárias para realizar a compressão se encontra na Figura 4.4. Os resultados obtidos encontram-se a seguir.

4.4 Resultados

Primeiramente, utilizamos o algoritmo para comprimir os mesmos textos utilizados no capítulo anterior, "David Copperfield" e "Os Maias", com uma árvore de memória de profundidade 2. As taxas obtidas foram de 0.32 bits/símbolo para o primeiro e 0.34 bits/símbolo para o segundo. Esses valores são surpreendentemente baixos: uma ordem de grandeza menores que os mostrados na Tabela 4.1. Como cada símbolo em um arquivo de texto usa um *byte* para ser representado na memória do computador, estamos comprimindo cada *megabyte* de texto em aproximadamente 40 *kilobytes*.

Podemos variar a profundidade da árvore de memória utilizada, comparando o nível de compressão conseguido para "David Copperfield" e o número total de nós na árvore. Isso é feito na Tabela 4.2. Fica evidente a importância da árvore para a capacidade preditiva do modelo: sem ela (profundidade zero), estamos usando apenas os padrões aprendidos, caso em que a compressão obtida é consideravelmente pior do que a dos algoritmos da literatura.

Além disso, fica claro também que a escolha da profundidade da memória se traduz num compromisso entre a taxa de compressão obtida e o tamanho da estrutura

Tabela 4.2: Taxas de compressão e número total de nós na árvore de memória M obtidos para o livro "David Copperfield", utilizando-se diferentes profundidades para M. Note que usar uma profundidade igual a 0 é o mesmo que ignorar o uso da árvore, fazendo nossas previsões com base apenas nos padrões aprendidos no texto.

Profundidade	Número de nós	bits/símbolo
3	290800	0.092
2	194118	0.32
1	99901	1.12
0	17377	3.03

Tabela 4.3: Taxas de compressão obtidas com nosso algoritmo para cada arquivo do *Calgary corpus* (profundidade da árvore de memória igual a 2). Os nomes dos arquivos foram mantidos conforme os utilizados em [48].

Arquivo	bits/símbolo
bib	0.87
book1	0.28
book2	0.43
geo	0.33
news	0.68
obj1	0.45
obj2	1.1
paper1	0.45
paper2	0.39
pic	1.0
progc	0.67
progl	0.83
progp	0.60
trans	1.2
Média	0.66

necessária para isso. Lembre-se de que, para decodificarmos o arquivo comprimido, precisamos da árvore que foi usada na compressão: é ela que representa o modelo probabilístico. Logo, se a árvore é criada especificamente para um arquivo—ao invés de uma árvore genérica, uma possibilidade que será aventada mais à frente—, seu tamanho deverá ser incluído no tamanho total do arquivo comprimido, o que acaba por reduzir, na prática, a compressão obtida.

Resta saber se esse mesmo nível de compressão pode ser conseguido para entradas de natureza diversa ou se funciona somente para texto. A Tabela 4.3 responde a essa pergunta mostrando as taxas de compressão obtidas para cada arquivo do *Calgary corpus*, cuja média é de 0.66 bits/símbolo. Essa é a melhor taxa de compressão já reportada.

4.4.1 Discussão

Como explicar os resultados obtidos? O papel da árvore de memória se mostrou crucial para alcançarmos valores tão baixos. Um motivo para isso é que usando ela estamos fazendo mais do que simplesmente aumentar o contexto. Os autores em [58] mostraram, para o algoritmo PPM, que aumentar o tamanho do contexto só melhora a previsão até certo ponto (de acordo com suas análises, contextos maiores que 5 não resultam em qualquer ganho de compressão, e podem de fato até mesmo piorá-la!). Isso ocorre porque, embora em teoria maiores contextos dariam origem a previsões mais exatas, há uma grande chance de, ao se buscar por um contexto grande, termos pouquíssimos resultados (uma distribuição incluindo poucos ou até mesmo nenhum símbolo). Nesse caso, estaremos usando a probabilidade de escape frequentemente, o que contribui para deteriorar a qualidade da compressão. No nosso algoritmo, embora muitas vezes utilizemos contextos grandes, estes são particionados nas palavras ou padrões conhecidos, o que torna a própria escolha do contexto menos rígida e resulta em distribuições de escape mais representativas (quando "escapamos", em geral removemos do contexto uma palavra inteira, e não apenas uma letra, como no PPM).

É claro que o PPM melhoraria suas previsões se construísse seu modelo percorrendo o texto inteiro antes de começar a compressão. Mas isso em geral se torna impraticável dado o tamanho da estrutura necessária para armazenar todos os contextos encontrados. No PPM é usada uma *trie* onde cada nó é um símbolo, similar à que usamos para aprender os padrões no capítulo anterior. Em contraste, nossa *trie* de memória fica muitas vezes menor que a do PPM uma vez que usamos palavras ao invés de letras.

De todo modo, nosso procedimento se parece muito mais com a maneira que nós mesmos usamos para sermos capazes de prever texto do que os demais algoritmos de compressão citados. Afinal, ninguém que começa a ler um livro em uma língua desconhecida consegue passar a adivinhar as letras seguintes a partir de algumas páginas de leitura. Nosso conhecimento é construído na memória ao custo de longos períodos de aprendizado. Além disso, nosso algoritmo se utiliza de um recurso que, se usado por um humano, certamente também irá melhorar significativamente sua performance: ele lê previamente o livro antes de tentar prevê-lo. Isso pode parecer trivial, mas nenhum dos outros algoritmos da literatura pode efetivamente se aproveitar desse método—eles simplesmente não possuem o tipo de conhecimento alto-nível que nosso algoritmo consegue construir no período de aprendizado, isto é, a capacidade de segmentar a entrada em padrões de alto poder descritivo (ainda que usemos uma passada completa do texto para treinar o PPM e façamos a compressão apenas numa segunda passada, sua taxa de compressão não é muito reduzida, con-

tinuando acima de 2 bits/símbolo [53]).

Com base nessa analogia entre o uso de padrões e nosso conhecimento prévio sobre uma língua, podemos pensar em construir um grande banco de padrões referentes a uma ou mais linguagens. Se tivermos algo assim, quando quisermos comprimir um certo arquivo naquela(s) linguagem(ns), bastaria reutilizar esse banco para "carregar na memória" a(s) linguagem(ns) em questão, sem a necessidade de treinar padrões especificamente para aquele arquivo. Isso seria ótimo, uma vez que o treinamento de padrões é muitas vezes mais lento do que o treinamento da memória. Além disso, se outro usuário que vá utilizar o arquivo já possuir esse banco, só é necessário transmitir a ele o código comprimido, permitindo, assim, que o banco seja grande sem que isso implique um tamanho dos arquivos.

Na Figura 4.1 vemos uma indicação de que isso seria possível, já que os padrões reconhecidos em "A Tale of Two Cities" são bastante razoáveis muito embora tenham sido aprendidos a partir de outro livro. Isso sugere que nossa rede de padrões está de certa forma aprendendo a linguagem em que o livro está escrito, sem ficar restrita ao material específico usado no treinamento.

As semelhanças observadas acima entre o comportamento do nosso algoritmo e as facilidades e dificuldades que um humano teria para realizar a mesma tarefa são interessantes se pensarmos que foi justamente a isso que nos propusemos no começo do Capítulo 3: tentar modelar aspectos do processamento cortical de forma plausível, ainda que extremamente simplificada. Mais ainda, são um sinal de que estamos fazendo alguma coisa certa. Poderia isso significar que a estrutura de caminhos direcionados em forma de árvore nos dá uma pista sobre a maneira com que nossa memória é realmente organizada?

4.5 Organização da memória no córtex

Tradicionalmente, modelos para a memória no cérebro baseiam-se na distinção entre memórias de curto e longo prazo (como no modelo clássico proposto em [59]). Entretanto, ainda há pouco consenso sobre de que forma essa informação é armazenada e acessada [60]. A ciência da computação nos ensina que a escolha de uma estrutura de dados define tanto a forma de armazenagem como seu acesso posterior, e tal escolha em geral é determinante para a eficiência dessas operações.

Podemos imaginar que, analogamente à nossa rede, onde cada nó representa um padrão, o mesmo princípio ocorre no córtex, no entanto com neurônios ou grupos de neurônios representando padrões em diversos níveis de abstração. De fato, existem evidências disso em estudos que descobrem, por exemplo, o "neurônio que representa imagens da Jennifer Anniston junto ao Brad Pitt", ou que codificam um objeto bastante específico, como "casa" [61]. Certamente esses nós fazem parte do

que chamamos de memória: na verdade, eles seriam as peças fundamentais com as quais poderíamos construir estruturas mais versáteis que permitissem armazená-los e buscá-los posteriormente de forma eficiente. Visto que demonstramos uma grande capacidade para codificação de uma sequência de símbolos, será que poderíamos estender essa analogia também à estrutura em forma de árvore direcionada que utilizamos? Isto é, será que caminhos direcionados fariam sentido dado o que sabemos sobre nosso cérebro?

É fácil pensar em exemplos do dia-a-dia em que nossos acessos à memória parecem depender de uma sequência fixa de conceitos: por exemplo, em geral as pessoas só conseguem listar as letras do alfabeto em ordem alfabética (a ordem em que o alfabeto foi decorado). É bem fácil começar de qualquer ponto: se ouvimos "M", imediatamente nos vem à mente: "N, O, P, Q,...". Porém, uma tarefa muito mais complicada é enunciar o alfabeto em ordem contrária: um exercício de introspecção parece nos revelar que, para fazer isso, precisamos buscar sucessivamente pequenas sequências de letras em ordem alfabética para serem usadas como "cola", a partir da qual conseguimos "ler" as letras em ordem contrária. (Por exemplo: "Z? (u → v → x → z) X? V? U? (r → s → t → u) T? S? R? ...").

Desafio praticamente impossível, entretanto, é listar todas as letras do alfabeto em uma ordem qualquer, ou "aleatória", como se as sorteássemos, uma a uma, de dentro de uma sacola. Isso parece não ser possível pois simplesmente não há em nosso cérebro uma estrutura que corresponda ao alfabeto e que seja organizada na forma de um conjunto não-ordenado. Letras podem ser acessadas diretamente, entretanto para percorrê-las sequencialmente estamos restritos à ordem com que elas foram aprendidas. (Bem parecido como em um vetor ou *array* em linguagens de programação similares a C.) É possível aprender outras ordens, claro, como a sequência de vogais: "A, E, I, O, U" que todos sabemos de cor. Mas note que até mesmo essa curta sequência de cinco letras já impõe um desafio quando queremos listá-la em outra ordem: imediatamente perdemos a conta de quais já foram citadas e quais ainda falta citar.

Mas não é somente dessa forma que a memória se manifesta. Embora as considerações acima façam sentido quando estamos aprendendo ou decorando sequências, muitas vezes nossos pensamentos parecem navegar incessantemente por um mar de memórias que não parecem possuir qualquer relação direta entre si, e independentemente da ordem em que elas foram criadas. Trata-se, conforme já foi adiantado no começo desta seção, da memória associativa, que será o assunto do próximo capítulo.

Capítulo 5

Agrupamento de padrões

5.1 Memória associativa

A discussão no final do capítulo anterior sobre um modelo de organização da memória em árvores ou caminhos direcionados deixou de fora um outro aspecto igualmente importante e que também é constantemente vivenciado por todos nós: a sensação de que, ao pensar em alguma coisa, nos lembramos automaticamente de outra coisa direta ou indiretamente relacionada à primeira (se ouvimos a palavra "gorgonzola", imediatamente lembramos de "queijo"), muitas vezes inclusive uma sobre a qual há muito tempo não pensávamos. Um exemplo típico disso é quando percebemos um cheiro que não sentíamos há algum tempo: temos imediatamente a nítida sensação de estar de volta àquele lugar ou situação evocados pelo cheiro.[1]

Não há dúvida de que para isso usamos nossa memória, entretanto de uma forma bastante distinta daquela usada nos caminhos direcionados vistos anteriormente. Na associação livre de palavras, em que uma pessoa vai dizendo o mais rápido possível as palavras que lhe vêm à mente, em geral não há qualquer semelhança entre a sequência de palavras produzida com uma sequência do tipo que se encontra em um texto, por exemplo. Essa memória é normalmente chamada de "associativa".

Modelos para esse tipo de memória não são novos: desde [63], que estabeleceu a plasticidade sináptica de longo termo como sendo a essência da formação de memórias (quando um neurônio A ativa outro neurônio B repetidamente, a conexão entre os dois é fortalecida), eles vêm sendo propostos, alguns mais biologicamente inspirados, outros mais abstratos. Para citar apenas alguns, o autor de [64] propõe que, sempre que passamos por uma experiência marcante, tudo que está ativo em nossa mente naquele momento ficaria associado em uma estrutura denominada "k-

[1] De fato, existe uma explicação neurocientífica para esse fenômeno que relaciona a intensidade das recordações acionadas por cheiros ao fato de nosso bulbo olfativo possuir conexões diretas com o hipocampo e a amígdala (áreas envolvidas, respectivamente, com as memórias espacial e emocional [62]).

line", e o de [65] inspirou-se na memória associativa humana para construir um modelo utilizando redes neurais artificiais recorrentes (conhecidas como "redes de Hopfield"). Este último é a base do paradigma *attractor-memory* para modelos do córtex.

5.2 Um algoritmo de agrupamento

Uma ideia simples para tentar reproduzir tal efeito baseia-se na intuição de que eventos entre os quais decorreu um curto intervalo de tempo devam ser mais fortemente associados do que eventos mais defasados temporalmente. Isso é facilmente observado com um exemplo inspirado no condicionamento clássico: se um animal aperta um botão e logo em seguida recebe comida, é muito mais razoável que ele associe a comida ao acionamento do botão do que a algum outro evento que tenha ocorrido horas (ou até mesmo minutos) antes.

Para emular essa ideia tomando proveito do nosso algoritmo de detecção de padrões, podemos imaginar que tudo que acontece dentro de uma certa janela de tempo ficaria associado de alguma forma. (É claro que nós, humanos, somos capazes de associar eventos separados por qualquer intervalo de tempo, mas antes de pensar em como realizar associações tão complexas assim, seria interessante simplesmente ver até onde uma regra simples pode nos levar.) Se pensarmos que os padrões vão sendo encontrados enquanto percorremos uma linha do tempo imaginária, e supondo que cada padrão leve o mesmo tempo para ser reconhecido, podemos criar uma janela de tamanho fixo dentro tal que todos os padrões detectados dentro dela passariam a estar associados na memória.

Representaremos essa associação através de arestas em uma rede não-direcionada, onde cada nó representa um padrão. A escolha da ausência de direção se dá pelo fato de que gostaríamos de que, se encontramos os padrões "`rato`" e "`queijo`", tanto o primeiro nos lembre do segundo quanto o segundo nos lembre do primeiro.[2] Não obstante, uma vez mais iremos focar em modelos os mais simples possível, os quais sempre poderão ser aperfeiçoados futuramente. Com isso, se esse tamanho de janela for igual a m, vamos conectando sucessivamente grupos de m padrões, à medida que eles vão sendo encontrados na sequência de entrada. Esses padrões vão sendo reconhecidos um a um de forma análoga à que usamos para construir a árvore de memória no Capítulo 4: alimentamos a entrada ao Algoritmo **??**, que vai nos retornando cada padrão reconhecido. Mantemos uma fila de tamanho fixo m, à qual vamos adicionando cada padrão. Cada vez que isso é feito, conectamos o padrão recém-inserido aos demais $m - 1$ padrões presente na fila (ver Figura

[2]Muito embora, é claro, existam muitas associações que vêm acompanhadas de uma ideia de consequência ou causalidade, para as quais arestas direcionadas poderiam fazer mais sentido.

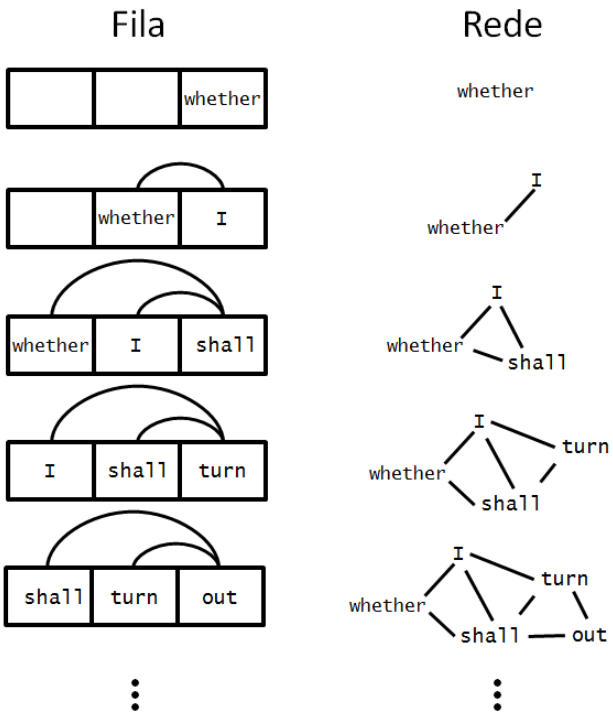


Figura 5.1: Início da formação da rede para o livro "David Copperfield". A cada passo, recebemos um padrão que é inserido em uma fila Q de tamanho $m = 3$ (nesse exemplo, estamos assumindo que cada palavra corresponde a um padrão). Sempre que inserimos um novo padrão em Q, criamos arestas entre ele e os demais. Cada vez que um padrão aparece, portanto, ele acaba por ser conectado, em geral, aos $m - 1$ padrões anteriores e aos $m - 1$ padrões seguintes (exceto para os $m - 1$ primeiros padrões encontrados, e também quando há padrões repetidos, já que não queremos considerar *self-loops*).

5.1).

De que forma uma rede formada por esse processo pode ser útil? Para começar, ela revela facilmente quais são os padrões mais "centrais" no domínio sendo alimentado ao algoritmo, no sentido de possuírem um número maior de padrões aos quais costumam aparecer próximos. Essa é uma maneira simples de diferenciar os diversos padrões detectados e que nos dá mais informação do que simplesmente o número de vezes que cada um deles foi encontrado. Afinal, é possível que um padrão seja bastante frequente porém apareça sempre junto dos mesmos poucos padrões vizinhos, isto é, tenha pequena variedade de vizinhos, enquanto que outro padrão pode ter uma frequência menor porém ser muito mais versátil no que diz respeito à variedade de padrões que ele pode ter como vizinhos. Um exemplo de rede construída por nosso algoritmo após a leitura da primeira frase de "David Copperfield" é mostrado na Figura 5.2.

No entanto, gostaríamos de extrair mais informações a partir dessa rede. Se alimentarmos o algoritmo com dois textos em línguas diferentes, por exemplo, um após

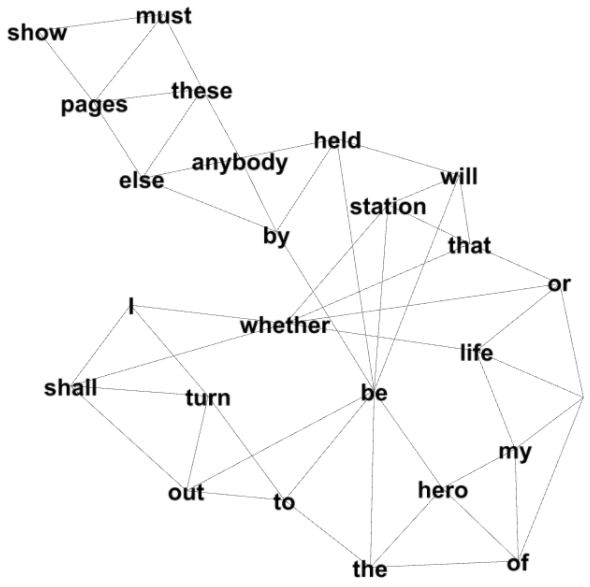


Figura 5.2: Rede formada com a primeira frase do livro "David Copperfield", "`whether I shall turn out to be the hero of my own life or whether that station will be held by anybody else these pages must show`" , usando $m = 3$. (De forma a ilustrar o método com maior clareza, consideramos que os padrões conhecidos são as próprias palavras presentes no texto.) Mesmo nesse pequeno trecho algumas palavras já despontam como mais centrais: "`whether`" e "`be`", que aparecem duas vezes cada. Note que com isso elas estão agrupando palavras do começo da frase com outras que aparecem mais à frente. À medida que a leitura do livro avança e a rede cresce, essas palavras mais centrais vão contribuindo para "clusterizar" todo o livro.

o outro, é de se esperar que a própria estrutura da rede reflita essa dicotomia presente na entrada alimentada (afinal, como as palavras usadas em uma língua costumam ser consideravelmente diferentes das usadas em outra, intuitivamente imaginamos que os nós de uma língua estarão conectados apenas a outros nós daquela mesma língua).

Mais que isso: seria possível através de uma análise da rede formada reconhecer, por exemplo, que recebemos dois textos distintos, ainda que escritos na mesma língua? Em outras palavras, se alimentarmos dois livros ao nosso algoritmo, seria possível descobrir automaticamente que naquela entrada existem dois domínios[3], tendo como base apenas a estrutura da rede? Isso seria o equivalente de receber um sinal extraterrestre e detectar automaticamente que há mais um assunto diferente sendo transmitido, mesmo sem saber o significado do sinal recebido. O mesmo

[3]Por "domínio" queremos denominar um conjunto de palavras ou padrões que possuam alguma relação maior entre si do que com padrões de outros conjuntos.

poderia ser aplicado ao domínio da música: seria possível diferenciar compositores automaticamente?

5.3 Resultados

Primeiramente, vamos olhar para as redes geradas pelo algoritmo com alguns conjuntos de entradas, cada um envolvendo dois domínios: um livro em português *vs.* um livro em inglês, um livro de Charles Dickens *vs.* um livro de Marcel Proust (ambos em inglês) e corais de Bach *vs.* madrigais de Monteverdi. A Figura 5.3 deixa claro que há alguns nós comuns aos dois domínios e muitos nós exclusivos a apenas um deles. Mais que isso, o uso de um algoritmo de posicionamento automático dos nós [66] já aproxima nós de um mesmo dómínio, sinalizando que separar os dois domínios não deve ser tão difícil.

5.3.1 Dividindo a rede em diferentes domínios

Gostaríamos de um procedimento que dividisse automaticamente a rede nesses dois domínios, já que, ao contrário das redes mostradas na Figura 5.3, em uma aplicação real não teríamos cores para nos dizer se dividimos as redes corretamente. Mas como conseguir isso? Uma ideia inicial seria a busca por módulos ou comunidades, conforme visto no Capítulo 2. Entretanto, após tentativas com diversos métodos (*Link Communities* [30], SLPA [67], *Chinese Whispers* [68], *Walk-trap* [69] e Girvan-Newman [70]), tivemos como resultado um número de comunidades muito maior que duas (dados não mostrados). Mesmo os métodos de agrupamento hierárquico não parecem muito promissores. O grande problema com essa abordagem é que dentro de um mesmo contexto há diversos módulos de nós mais fortemente conectados, os quais acabam sendo reconhecidos como comunidades.

Gostaríamos de poder ignorar esses módulos localizados e encontrar uma divisão da rede em um nível mais alto. Embora os métodos de agrupamento hierárquico sejam capazes, teoricamente, de dividir a rede em diferentes números de comunidades—bastando para isso cortar o dendrograma no nível de agrupamento desejado—, quando chegamos a níveis mais altos os dois domínios já estão demasiadamente misturados. Esses algoritmos são "confundidos" pelos padrões mais centrais na rede (de alto grau).[4]

[4]Isso ilustra o fato crucial de que, embora haja muitos trabalhos que se propõem a avaliar a qualidade intrínseca de algoritmos de divisão de redes em comunidades, o sucesso dessa tarefa é diretamente dependente do que queremos dizer efetivamente com "comunidade". Muitas vezes busca-se um conceito que defina "comunidade", mas, tanto no Capítulo 2 como neste, vimos que tal definição depende largamente do tipo de informação que queremos obter. Logo, não faz sentido buscar comunidades em uma rede sem ter uma pergunta prévia que queiramos responder.

5.3.2 Uma métrica para descobrir nós inter-domínios

Vamos desenvolver um outro procedimento, baseado na seguinte observação: dada a regra de formação de nossa rede, todo nó possui vizinhos que apareceram dentro de uma mesma janela de tempo. De fato, um nó v_i deverá inclusive ter vários grupos de vizinhos, um grupo para cada vez que ele apareceu (a não ser que ele sempre apareça cercado pelos mesmos padrões). Isso contribui em muito para o aumento do seu coeficiente de clusterização C_i (dado pela fração de pares de vizinhos conectados entre si dentre todos os pares de vizinhos) [71]:

$$C_i = \frac{|\{e_{jk} : v_j, v_k \in N(v_i), e_{jk} \in E\}|}{d(v_i)[d(v_i) - 1]/2},$$

onde e_{jk} representa uma aresta entre os nós v_j e v_k, E é o conjunto de todas as arestas da rede e $N(v_i)$ e $d(v_i)$ denotam, respectivamente, a vizinhança e o grau de v_i.

Agora, note que esses grupos podem possuir diversas conexões entre si (o que contribui ainda mais para o aumento de C_i), porém somente entre os vizinhos que apareceram num mesmo domínio. Note também os nós que aparecem em ambos os domínios provavelmente terão grupos de vizinhos pertencentes a somente um deles. Isso irá diminuir drasticamente seus coeficientes de clusterização, pois estamos aumentando muito o número total de pares de vizinhos possíveis. Essa ideia é ilustrada na Figura 5.4. Na prática, descobrimos que uma pequena alteração na fórmula de C_i torna a disparidade entre seus valores para nós inter- e intra-domínios ainda maior. Denotamos essa nova métrica por C_i', na qual, ao invés de somar 1 para cada aresta entre dois nós v_j e v_k vizinhos de v_i, vamos somar $1/(d^*(v_j) + d^*(v_k))$, onde $d^*(v)$ se refere ao grau de v decrescido de uma unidade (para não contarmos o próprio nó v_i, para o qual estamos calculando o coeficiente).

Com isso, fazemos com que vizinhos de alto grau não contribuam muito para aumentar o valor da métrica. Para ilustrar a vantagem disso, suponha que v_i é um nó inter-domínios, que tem dois vizinhos intra-domínios, v_A, pertencente apenas ao domínio A, e v_B, pertencente apenas ao domínio B, e ainda um terceiro vizinho v_{AB}, pertencente a ambos os domínios e que também é ligado a v_A, v_B e a outros nós da rede. Como v_A e v_B são intra-domínio, eles não são adjacentes, logo isso contribui para diminuir C_i. Mas ainda assim o nó v_{AB} vai fechar dois triângulos ao redor de v_i, o que não seria bom. Porém, a grande maioria dos nós inter-domínios possui grau muito alto (se eles aparecem em ambos os domínios, costumam ser "universais", logo aparecem frequentemente). Nossa adaptação à formula de C_i faz com que os triângulos fechados utilizando esse tipo de nó passem a contribuir muito pouco para

Tabela 5.1: Vinte nós com os menores valores de C'_i na rede Dickens *vs.* Proust.

C'_i	Nó	Livro
0.00075	`time-`	ambos
0.00092	`ed-`	ambos
0.00094	`ing-`	ambos
0.00095	`Dombey-`	Dickens
0.0010	`s`	ambos
0.0012	`and-`	ambos
0.0014	`in-`	Dickens
0.0014	`ex`	ambos
0.0014	`of-`	ambos
0.0014	`in-his-`	ambos
0.0015	`ance-`	ambos
0.0015	`eyes-`	ambos
0.0015	`y-`	Dickens
0.0015	`son-`	Dickens
0.0016	`would-`	ambos
0.0017	`ist`	ambos
0.0018	`my-`	ambos
0.0020	`to-`	ambos
0.0021	`x`	ambos
0.0021	`myself-`	Proust

o aumento do valor final de C'_i. Podemos escrever sua fórmula como

$$C'_i = \frac{\sum_{v_j \in N(v_i)} \sum_{\substack{v_k \in N(v_i) \setminus \{v_i\} \\ e_{jk} \in E}} \frac{1}{(d^*(v_j)+d^*(v_k))}}{d^*(v_i)\left[d^*(v_i)-1\right]/2}.$$

Na Tabela 5.1, observamos a identidade dos vinte nós com menor C'_i para a rede Dickens *vs.* Proust. Vemos que de fato a maioria dos nós de baixo C'_i representam padrões comuns a ambos os livros. Entretanto, alguns nós que aparecem somente em um dos livros mas que são muito centrais tamém estão nessa lista. A razão é que os padrões que eles representam, em geral os personagens principais, aparecem repetidas vezes ao longo de todo o livro—como "`Dombey-`" e "`son-`" no livro de Dickens (as personagens principais da história), e "`myself-`" no de Proust (um livro escrito em primeira pessoa)—, e portanto adquirem vizinhos de várias partes ou capítulos diferentes do mesmo livro. Logo, eles também acabam por ter grupos de vizinhos não conectados entre si, embora pertencentes ao mesmo livro.

5.3.3 Agrupamento dos domínios

Agora basta ir removendo sucessivamente os nós de menor C'_i até que a rede seja particionada em duas componentes conexas—cada uma delas corresponderá a uma

"comunidade" referente a um livro ou domínio. Em seguida, para cada componente, podemos readicionar os nós e arestas removidos para recuperar os nós intra-domínio removidos indevidamente (tomando o cuidado de duplicar os nós que pertenciam a ambas, a fim de que elas não se reconectem!).

O processo de particionamento da rede pode ser visto passo a passo nas Figuras 5.5, 5.6 e 5.7. Ele permitiu agrupar corretamente os dois domńios em todas as redes estudadas. Uma observação: na remoção sucessiva de nós, os valores de C_i' usados devem ser aqueles calculados para a rede original: caso contrário (se eles fossem recalculados após cada remoção), a remoção dos nós inter-livros acaba aumentando os C_i' de outros nós inter-livros, comprometendo a qualidade da separação.

Uma pergunta que nos devemos fazer é: o que acontece se houver somente um contexto? Ele acabaria se separando em várias componentes, e nesse caso cada uma delas estará revelando sub-contextos, como por exemplo capítulos ou partes que forem menos semelhantes entre si dentro do mesmo livro. Também é o que ocorre se continuarmos a remover nós após a separação em duas componentes, o que, aliás, devemos sempre fazer pois teoricamente não sabíamos, para começar, que havia dois domínios no sinal de entrada.

Podemos tentar descobrir isso plotando a fração de nós já removidos, f_n, contra a fração de componentes conexas obtidas, f_{CC}. A Figura 5.8 mostra a diferença entre as curvas obtidas para dois domínios e para apenas um, ainda que sutil. Vemos que, no caso de dois domínios, após a primeira quebra da rede em duas componentes conexas, f_{CC} segue variando mais frequentemente. Em contraste, quando só há um domínio, f_{CC} passa por períodos em que permanece constante perante a remoção de nós. Isso é explicado pelo fato de que, no primeiro caso, estamos gerando componentes simultaneamente em dois domínios, portanto a alteração no número total de componentes fica mais frequente.

5.3.4 Discussão

O método apresentado conseguiu inferir satisfatoriamente uma disparidade entre dois trechos presentes em uma mesma sequência de entrada através da criação de uma rede de associação de padrões. Em resumo, ao criar conexões entre os padrões que ocorrem próximos uns dos outros estamos contribuindo para agrupar trechos desse sinal de entrada que têm maior relação entre si. O resultado é que, partindo-se de uma entrada para a qual descobrimos os padrões pertinentes, é possível descobrir, sem qualquer supervisão, o melhor agrupamento desses padrões em módulos de padrões que fazem mais sentido juntos do que separados.

O agrupamento é realizado por uma divisão da rede baseada na identificação dos nós "fronteira" entre os dois domínios. Para isso, criamos uma métrica simples que

permite identificar esses nós de forma eficiente e com relativamente poucos falsos-positivos.

Outras métricas, como *edge betweenness*, também foram testadas mas, embora tenham tido desempenho próximo ao da que usamos, eram demasiadamente custosas de serem computadas. O C_i' de cada nó v_i, por outro lado, pode ser calculado de forma totalmente localizada, bastando para isso percorrer o subgrafo limitado à vizinhança de v.

Observe que, uma vez mais, o fato de não termos palavras exatas, e sim padrões aprendidos com o algoritmo do Capítulo 3, se mostrou fundamental: além de permitir que obtenhamos resultados para entradas de natureza desconhecida, isso também faz com que livros, ainda que escritos na mesma língua, tenham menos padrões comuns entre si do que se usássemos palavras exatas, tais como as conhecemos. Isso ajuda a rede formada a ser separada pelo nosso método, que tem resultados um pouco piores quando usamos palavras exatas (dados não mostrados).

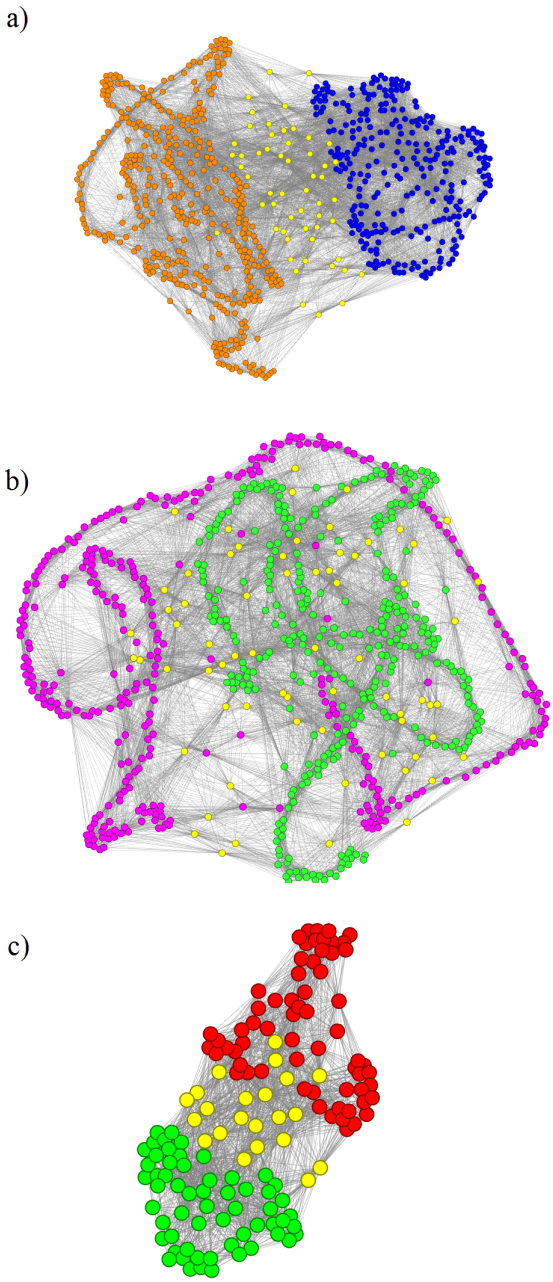


Figura 5.3: Três redes geradas pelo algoritmo usando entradas contendo dois domínios distintos cada uma. Em (a), temos línguas diferentes: um trecho de um livro em português ("Os Maias") (nós em laranja) seguido de um trecho de um livro em inglês ("David Copperfield") (nós em azul). Em (b), temos dois livros em inglês porém de autores diferentes: "Dombey and Son", de Charles Dickens (nós em verde), e "Swann's Way" (o primeiro volume de "In Search of Lost Time"), de Marcel Proust, traduzido do francês (nós em rosa). Em (c), temos música de dois compositores diferentes: dez corais de Bach (nós em vermelho) seguidos de cinco madrigais de Monteverdi (nós em verde). Nas três redes, os nós em amarelo correspondem aos padrões comuns a ambos os domínios. A disposição dos nós é dada por um algoritmo de posicionamento automático [66] que simula um modelo físico no qual os nós repelem uns ao outros e as arestas aproximam seus nós incidentes.

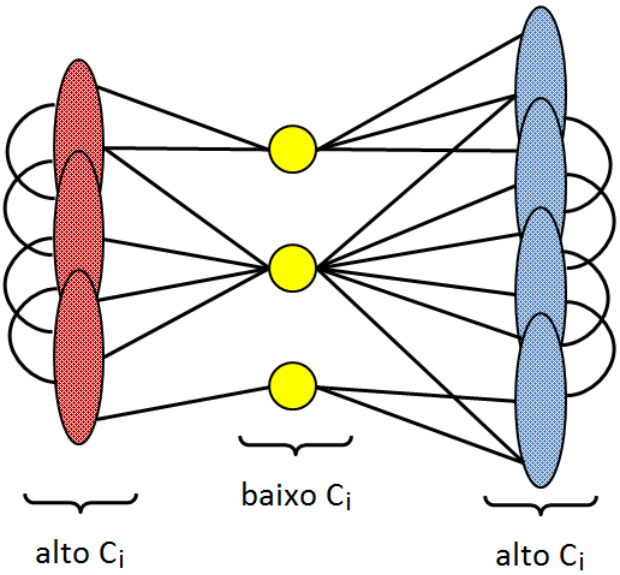
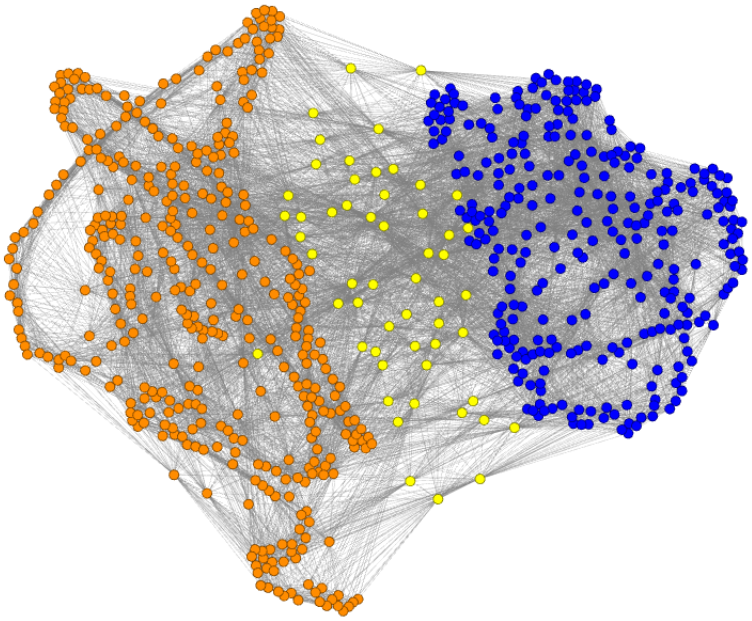
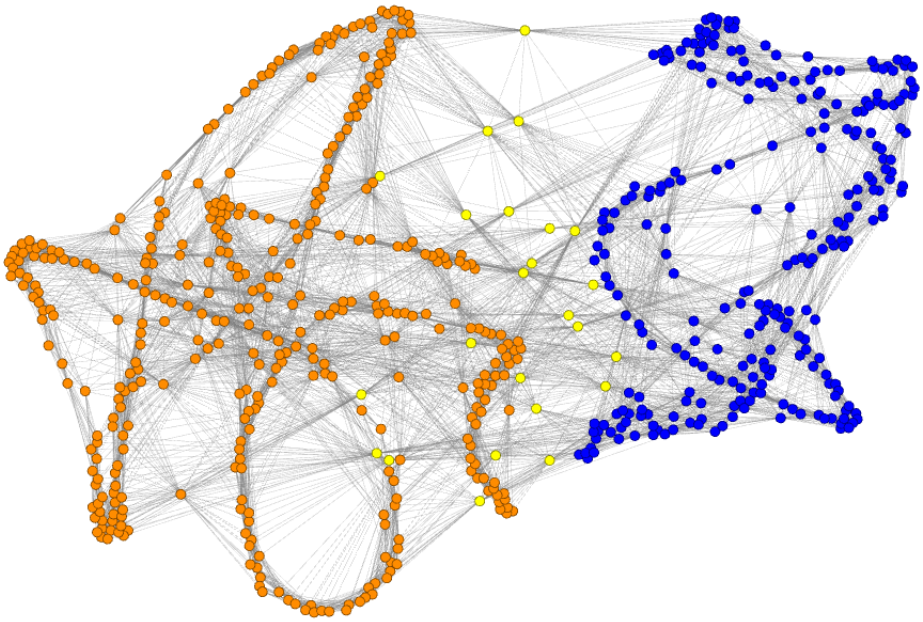


Figura 5.4: Diagrama ilustrando a razão pela qual nós inter-domínios (em amarelo) possuem baixo coeficiente de clusterização C_i. Os nós de dois domínios distintos estão representados por elipses vermelhas e azuis, respectivamente. Todos os caminhos que ligam um domínio ao outro obrigatoriamente passarão pelos nós amarelos. Além disso, nós intra-domínio possuem muitos vizinhos dentro do prprio domínio, porém poucos vizinhos inter-domínio. Portanto, os nós amarelos obrigatoriamente possuirão pares de vizinhos não-adjacentes (e, em geral, muitos deles), o que diminui consideravelmente seus valores de C_i.

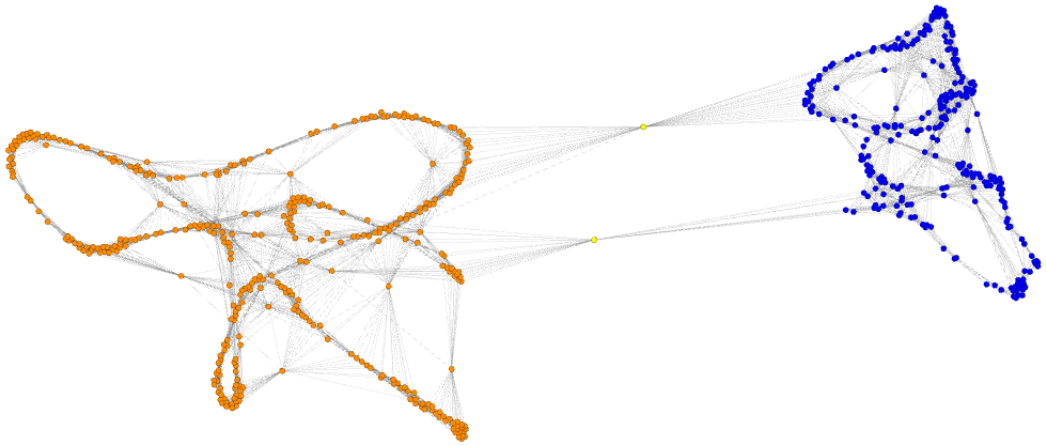


(a)

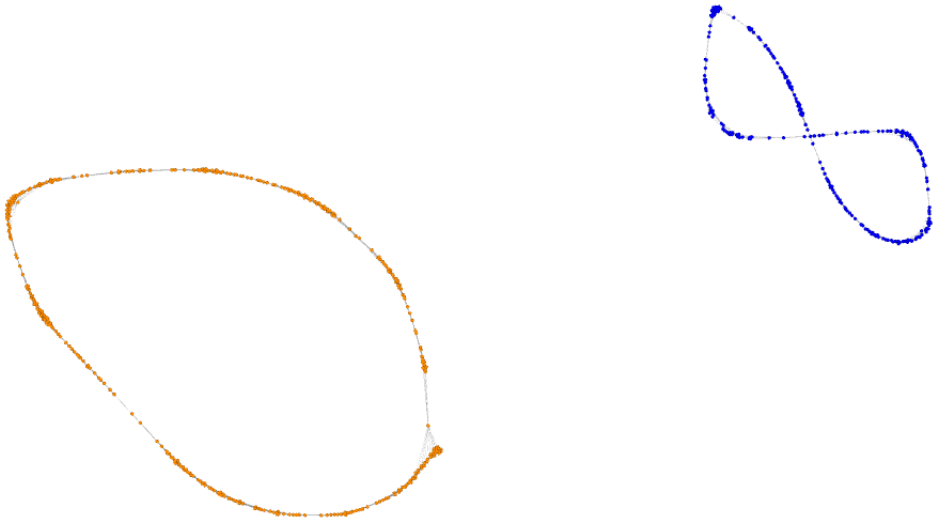


(b)

Figura 5.5: Quatro momentos do processo de separação da rede inglês *vs.* português em dois domínios (painéis (a)–(d)). Note como ao final temos duas componentes conexas, cada uma contendo apenas um domínio (representados por cores diferentes, conforme as usadas na Figura 5.3).

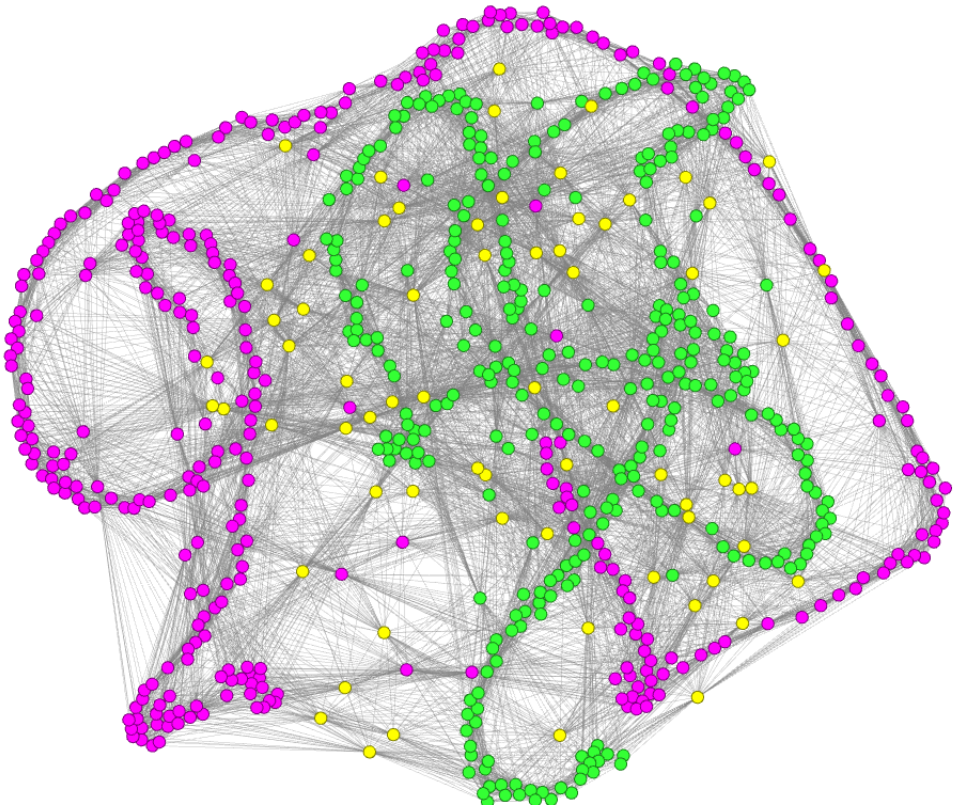


(c)

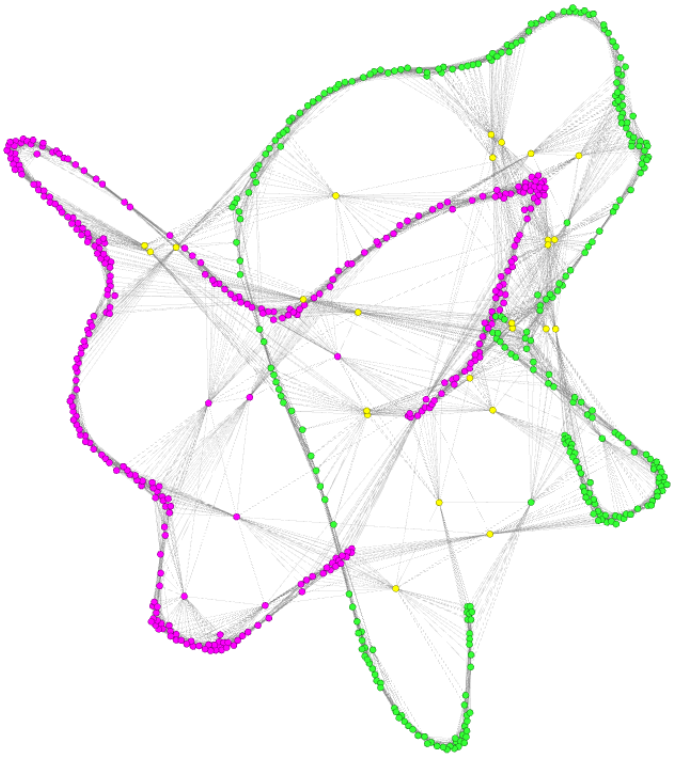


(d)

Figura 5.5: (continuação)

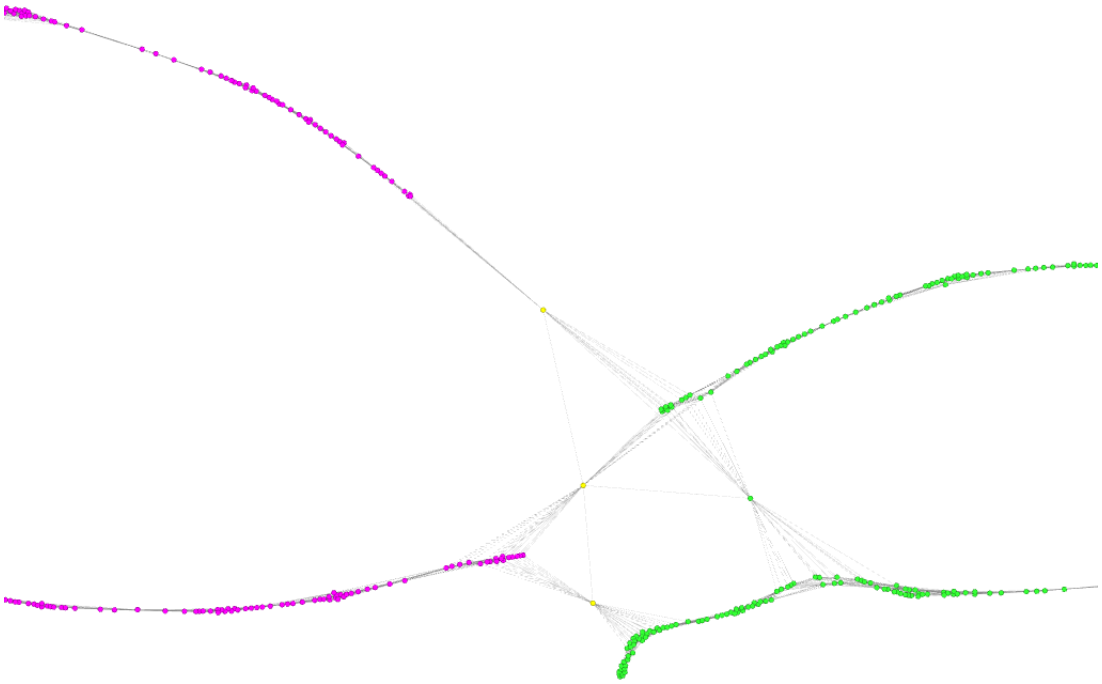


(a)

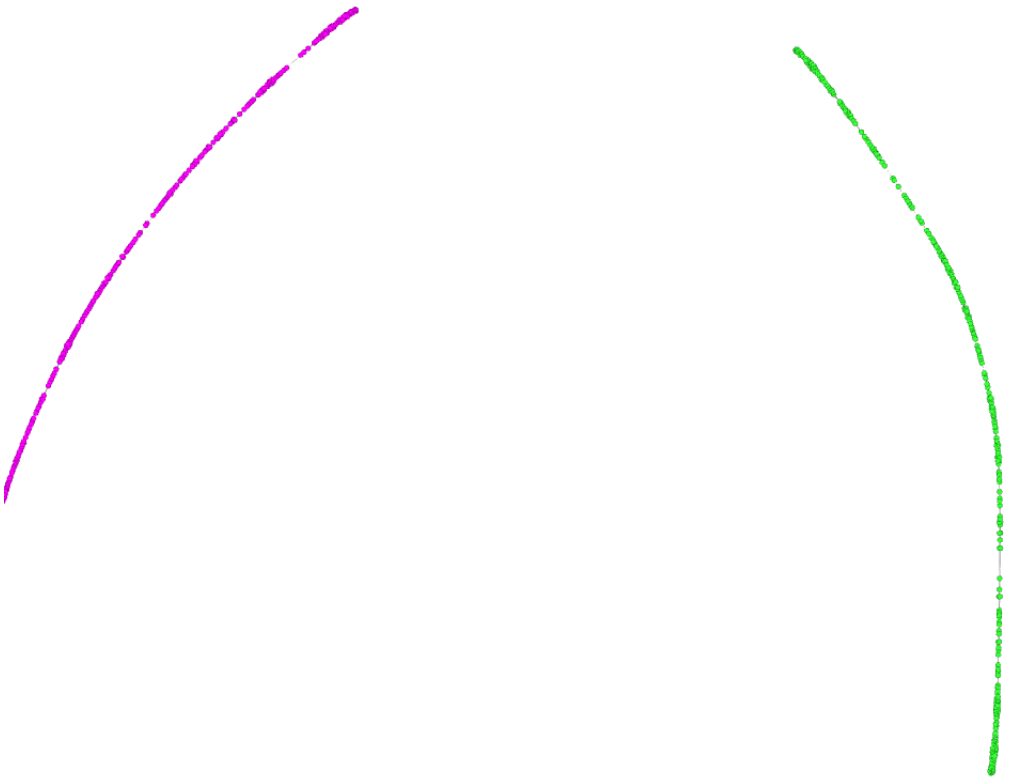


(b)

Figura 5.6: Quatro momentos do processo de separação da rede Dickens *vs.* Proust em dois domínios (painéis (a)–(d)). Note como ao final temos duas componentes conexas, cada uma contendo apenas um domínio (representados por cores diferentes, conforme as usadas na Figura 5.3).

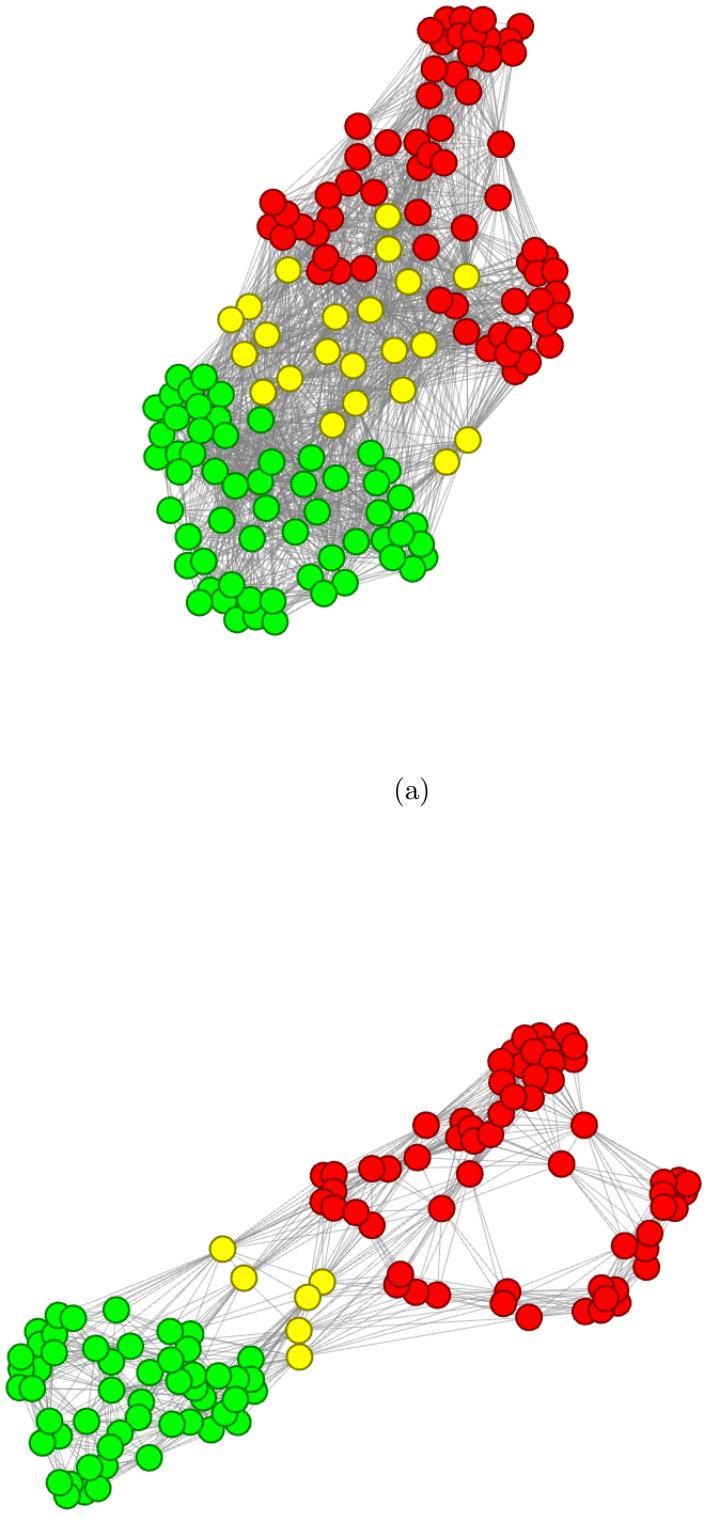


(c)



(d)

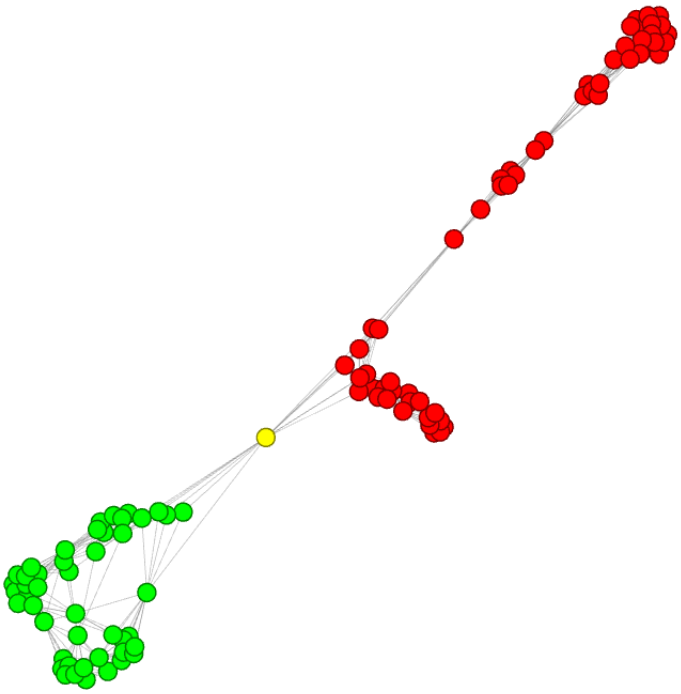
Figura 5.6: (continuação)



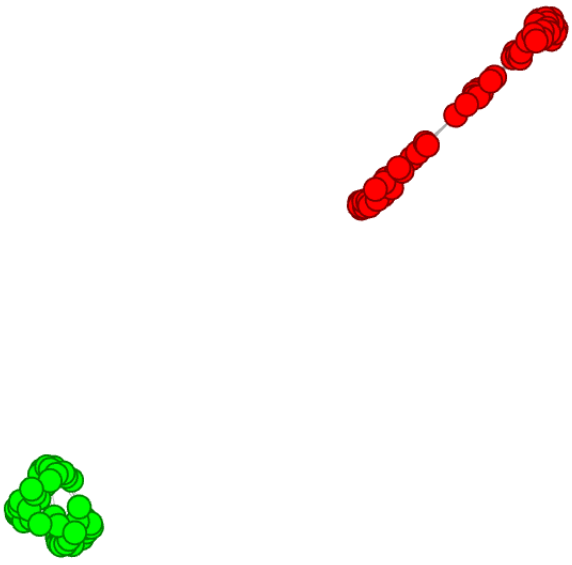
(a)

(b)

Figura 5.7: Quatro momentos do processo de separação da rede Bach *vs.* Monteverdi em dois domínios (painéis (a)–(d)). Note como ao final temos duas componentes conexas, cada uma contendo apenas um domínio (representados por cores diferentes, conforme as usadas na Figura 5.3).



(c)



(d)

Figura 5.7: (continuação)

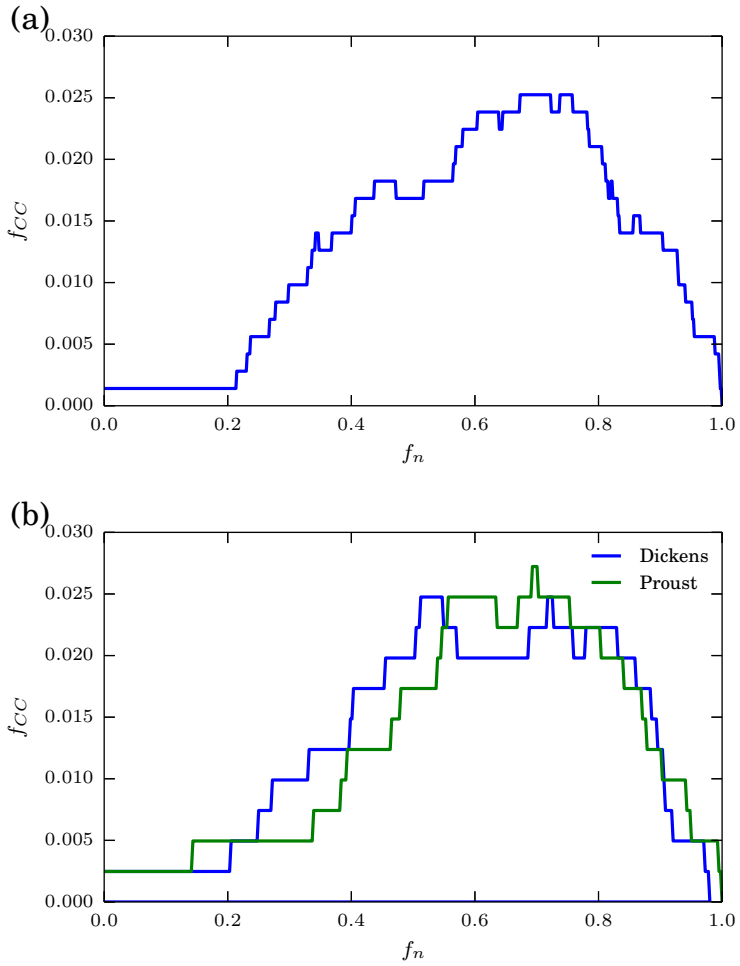


Figura 5.8: Fração de nós já removidos, f_n, contra a fração de componentes conexas obtidas, f_{CC}. Esta última é tomada com relação ao máximo possível de componentes conexas (o número total de nós da rede). Frações foram usadas de forma a possibilitar o uso das mesmas escalas em ambos os gráficos, já que cada rede possui um número distinto de nós. No painel (a) temos a rede Dickens *vs.* Proust, e em (b) temos as redes geradas separadamente para cada domínio.

Capítulo 6

Conclusão

6.1 Discussão geral dos resultados

Nesse trabalho, inicialmente observamos tendências e regularidades peculiares a respeito da rede cortical de macacos do gênero *Macaca*, tendo como base seus dados de conectividade. Em particular, encontramos indicações de que o nível hierárquico de uma dada área cortical pode ser aproximado pela distância desta área até uma das áreas sensoriais primárias, principais portas de entrada de informação externa no córtex.

Vimos também que essa rede pode ser particionada em algumas poucas comunidades principais compostas por áreas responsáveis por funções similares, e que a comunicação entre essas comunidades é feita de maneira especializada por outras comunidades formadas por interseções entre as comunidades principais. No que diz respeito a essa comunicação, existe uma tendência geral de que ela ocorra nos dois sentidos, e de que para isso empreguem quantidades similares de neurônios.

Tais resultados serviram como inspiração para tentarmos desenvolver modelos biologicamente razoáveis que reproduzissem algumas das capacidades do nosso cérebro para o processamento de sinal sensorial. Na verdade, a quantidade crescente de dados publicados sobre a atividade cerebral vem motivando e inspirando também muitos outros a elaborar modelos do processamento feito no cortex (por exemplo, [72]).

No Capítulo 2, apresentamos um mecanismo universal para aprendizado de padrões inspirado em algumas caracterśticas de redes neurais biológicas. Vimos que, tal como ocorre na natureza, a representação de conceitos progressivamente mais complexos é desencadeada naturalmente através de nossa rede, sendo guiada exclusivamente pelas características do sinal que chega até ela (distribuição de probabilidades dos símbolos).

É importante mencionar que de forma alguma queremos implicar que os circuitos

corticais funcionem como nossa rede de nós—nosso modelo é extremamente simples comparado ao mar de detalhes e complexidade presentes nos mecanismos celulares envolvendo neurônios. Por exemplo, até mesmo a noção de *threshold*, tão explorada em outros modelos desde os primeiros neurônios artificiais propostos em [73], não é utilizada em nosso modelo: ao invés de esperar até que um sinal seja suficientemente consistente para ativar outro nó, criamos uma nova conexão de imediato e posteriormente a desfazemos caso ela venha a se mostrar inadequada.

Embora nosso trabalho não tenha buscado criar modelos fiéis, nossos resultados de certa forma fortalecem a ideia de que talvez eles possuam alguma base de verdade. Se por um lado ignoramos inúmeros detalhes amplamente conhecidos sobre os mecanismos celulares usados por neurônios, por outro os algoritmos criados puderam ser utilizados para aplicações reais e com resultados satisfatórios. Essa é uma abordagem bastante utilizada no campo da "vida artificial", onde buscam-se os princípios mais gerais por trás da emergência da complexidade vista em sistemas biológicos.

Com eles, pudemos descobrir os elementos básicos de formação existentes em um sinal de entrada, seja este na forma de texto ou notação musical, ou ainda dos diversos domínios presentes no *Calgary corpus*. A aplicação a problemas de outros tipo também deve ser possível, bastando para isso que haja um fluxo de informação sequencial o qual desejemos entender melhor. Um exemplo seria o genoma de um organismo (sequência de símbolos representando nucleotídeos), ou então uma sequência de parâmetros operacionais fornecidos por um equipamento industrial em operação. Na verdade, de forma similar ao que é feito no *deep learning*, nosso padrões também poderiam ser usados como *features* para geração de vetores de entrada em um método tradicional de aprendizado de máquina.

Note que o sucesso do algoritmo perante diferentes domínios sugere que, assim como textos são formados a partir de uma linguagem, outros domínios, como música ou imagens, também possuiriam sua própria "linguagem". Se essas outras linguagens possuem a complexidade gramatical que sabemos existir em nossos idiomas, esta já é uma questão que ainda não sabemos decidir. Nosso algoritmo consegue aprender padrões estatisticamente relevantes, e inclusive agrupá-los em módulos com maior coerência interna (conforme Capítulo 5), mas ele não consegue inferir automaticamente o papel sintático de cada padrão, por exemplo.

Acreditamos que para isso a presença de múltiplas entradas sensoriais seria essencial, pois no caso dos humanos o aprendizado semântico de substantivos e verbos depende fortemente de associação sensorial (em especial com a visão ou com o tato, no caso de cegos). No entanto, pensamos que o presente trabalho tenha contribuído para que novos patamares de aprendizado não-supervisionado sejam atingidos, em especial aqueles que se propõem a emular as capacidades humanas.

Um aprimoramento importante para o algoritmo de formação de padrões seria incorporar a capacidade de lidar com ruído (na forma de símbolos trocados ou faltantes). Repare que isso não é tão trivial quanto se possa imaginar, afinal de contas como saber quando uma aparente inconsistência trata-se de ruído ou de um novo padrão? Uma ideia seria tratá-la como ruído mas guardar durante algum tempo na rede essa variação observada, para que, caso ela passe a aparecer repetidamente, possa ser "promovida" a um padrão propriamente dito.

Os padrões aprendidos foram organizados de maneiras distintas para gerar resultados bem diferentes. A primeira delas foi em forma de caminhos direcionados, que permitiram expandir o tipo de informação guardada para sequências de padrões. A qualidade da compressão conseguida a partir disso serviu como um atestado da capacidade de aprendizado alcançada por nossa rede.

Para tornar nosso algoritmo de compressão ainda mais eficiente, observe que é perfeitamente possível estender a ideia usada no aprendizado de padrões, onde excluímos os padrões menos frequentes de modo a limitar a quantidade de nós presentes, para a árvore de memória. Para isso basta designar tempos de vida a cada sequência inserida na árvore e ir excluindo-as sempre que elas não aparecerem com a frequência necessária. Isso deve permitir diminuir o espaço usado por essas estruturas sem que isso acarrete grandes perdas na capacidade de compressão.

No Capítulo 5, organizamos esses padrões na forma de uma rede. Ao formar pequenas cliques com os padrões detectados entre os quais decorreram-se intervalos curtos de tempo, contribuímos para um agrupamento sucessivo desses padrões. Isso gerou redes que naturalmente mantêm estruturalmente separados domínios distintos que eventualmente estejam presentes na entrada sendo lida.

Supreendentemente, nosso algoritmo conseguiu separar até mesmo dois livros escritos na mesma língua. Outra coisa curiosa é que, no exemplo com música, embora tenhamos alimentado diversas músicas em sequência (dez corais e cinco madrigais), o algoritmo separou a rede nos dois compositores dessas peças, e não em cada uma delas. Isso nos leva a crer que diferentes peças de um mesmo compositor "contam" uma mesma "história" musical, em que os personagens principais seriam os motivos musicais que se repetem em todas elas.

Não é incomum reconhecermos o estilo de um compositor mesmo quando estamos ouvindo uma música inédita composta por ele. É claro que isso nem sempre será verdade; muitos artistas mudam seu estilo ao longo da carreira ou propositalmente variam suas técnicas em diferentes peças. Além disso, para cada compositor no nosso exemplo, escolhemos apenas peças obedecendo a uma mesma forma: apenas corais para o primeiro, e apenas madrigais para o segundo. Entretanto, o resultado poderia ser diferente se tomássemos, para Bach, algumas fugas misturadas aos corais, peças que possuem diferenças marcantes. (Nesse caso, talvez o algoritmo agrupasse corais

e fugas separadamente?)

É importante observar ainda que, muito embora o método que criamos para agrupar a rede nesses domínios tenha sido bem sucedido em todos os casos testados, se a natureza da entrada sendo lida for totalmente desconhecida é difícil dizer com certeza o número de domínios envolvidos. Por exemplo, se tivermos três domínios, como saber ao certo que se tratam de três e não de dois? Nesse caso a análise qualitativa que foi sugerida usando-se o gráfico de número de nós removidos *vs.* número de componentes conexas certamente seria mais difícil. Ainda assim, o resultado visual dado pelo algoritmo de posicionamento dos nós nos sugere que haja uma maneira sistemática ainda mais objetiva para separar esses domínios em "comunidades" do que a que foi utilizada.

6.2 Agrupamentos sucessivos

Com base no que foi aprendido, podemos agora retomar o panorama geral da organização do córtex que adiantamos no Capítulo 1.

6.2.1 Comunidades centradas em áreas sensoriais primárias

Dado que conseguimos aprender padrões com sucesso no Capítulo 3, é possível imaginar que mecanismos minimamente análogos ocorram nas áreas sensoriais do córtex. De modo geral, isso foi conseguido no nosso algoritmo partindo-se de padrões extremamente simples e agrupando-os progressivamente em padrões mais complexos através da formação de pares. Um procedimento similar poderia ocorrer no córtex: sempre que dois neurônios (ou dois grupos de neurônios) representando padrões simples fossem ativados em sincronia, ou sequencialmente, a combinação dos dois passaria a ativar um novo neurônio (ou conjunto de neurônios) em um nível mais alto. Logo, este último estará usando menos neurônios para representar o mesmo padrão resultante da combinação ocorrida em um nível mais baixo. Essa ideia é bastante similar ao que se conhece sobre a forma com que imagens são formadas no córtex visual, em que áreas de diferentes níveis hierárquicos representam padrões de diferentes níveis de complexidade [74].

Embora não tenhamos feito isso explicitamente em nossa rede de formação de padrões, é possível dividi-la em níveis de abstração baseados na complexidade dos padrões (número de pares necessários para formá-los). Se fizermos isso, veremos que os padrões de um mesmo nível possuirão características semelhantes: mesmo número aproximado de símbolos, e valores de k semelhantes.

Podemos imaginar esses conjuntos como equivalentes a áreas corticais. Afinal, cada área possui características próprias, como distribuição laminar, densidade ce-

lular e tipos de neurônios usados, que podem ser vistos como parâmetros ajustados evolutivamente para permitir o aprendizado de padrões naquele nível de complexidade. Sob essa ótica, é interessante pensar que nosso sistema vai se tornando modular justamente devido à modularidade inerentemente presente nos sinais sendo recebidos.

Em cada um desses níveis, ou áreas, estamos agrupando padrões para o descobrimento de novos padrões num nível acima. Na nossa rede, cada vez que isso acontece, criam-se conexões entre o par de padrões combinados e o novo padrão formado. Isso provoca o surgimento de conexões entre os dois níveis, ou, de forma análoga, a formação de conexões físicas entre duas áreas. Naturalmente, isso contribui para que todas as áreas responsáveis por um mesmo tipo de entrada sensorial, ainda que cada qual em um nível, acabem por estar todas fortemente conectadas umas às outras. Em outras palavras, do ponto de vista estrutural estamos formando uma comunidade entre essas áreas.

Agora, imagine que tenhamos várias redes desse tipo em paralelo, cada uma recebendo seu próprio sinal de entrada (no nosso algoritmo, só consideramos uma rede por vez). Então, num nível de observação mais alto, poderíamos ver cada uma delas como uma comunidade própria. No córtex isso equivaleria à ideia de que cada comunidade corresponderia a um sentido.

Ora, vimos anteriormente que o córtex de mamíferos tem como característica marcante uma alta modularidade. Seria possível embasar as conjecturas feitas acima a partir do que descobrimos a respeito da identidade das comunidades identificadas no Capítulo 2? Acreditamos que a resposta é sim: se olharmos com atenção, duas das chamadas comunidades principais são aproximadamente centradas em pontos do cérebro onde há entrada de informação: temos uma comunidade claramente visual (comunidade 2) e outra envolvendo uma área somatossensorial primária (comunidade 4). A comunidade 1 envolve alguma áreas relacionadas ao córtex auditivo, mas também inclui todo o córtex pré-frontal (que, por sua vez, também recebe grande *input* dos gânglios da base e sistema límbico). Talvez, quando o conjunto de dados utilizado passar a incluir uma parcela maior de áreas corticais, possamos testar melhor nossa hipótese (os experimentos feitos pelo grupo de [16] seguem sendo realizados).

6.2.2 Córtex motor

Note que até agora só discutimos o papel das áreas sensoriais na organização modular do córtex. Mas o que dizer sobre o córtex motor? Essa é a parte do córtex responsável pela tarefa oposta: emitir sinais que serão levados até nossos membros e atuadores para execução de ações por nosso corpo. Logo, à primeira vista seu me-

canismo não precisaria ter qualquer semelhança com o responsável por reconhecer padrões. Entretanto, é fácil imaginar como nossa rede poderia também ser usada para aprender ações progressivamente mais complexas.

Imagine um robô que possua apenas quatro tipos de locomoção: frente, trás, esquerda e direita, e que, para desencadear cada um desses movimentos básicos, um conjunto de comandos precise ser acionado. Começaríamos aprendendo nós capazes de emitir as sequências de sinais que ativariam essas quatro ações mais elementares. Nesse caso, esses nós seriam reforçados não pelo reencontro do padrão que eles codificam, já que não estão recebendo padrões, mas sim pelo sucesso ou utilidade da ação desempenhada. Esse reforço precisaria chegar até os nós por outro caminho, o que no nosso cérebro é feito em geral pelo mesencéfalo através do uso de dopamina [75].

Uma vez que tivéssemos aprendido esses movimentos básicos, seria apenas uma questão de combinar padrões (mais uma vez, em pares, por exemplo), para aprender ações mais complexas como andar em diagonal, ou em círculo.[1] Ainda assim, ficaria faltando um fator crucial: repare que o ato de receber um sinal de entrada, no caso de nossa rede original, é passivo, isto é, a ativação dos nós não requer qualquer ação por parte do organismo. Entretanto, para aprender ações motoras, cabe ao próprio organismo ativar seus nós e "experimentar" novas ações que poderão ou não ser úteis.

Nos animais, isso é feito através de um *drive* inato (mais uma vez proporcionado por substâncias liberadas periodicamente na corrente sanguínea) [74] que nos motiva a nos movimentarmos para buscar comida, ou a sair de uma posição incômoda, por exemplo. O ato de brincar, inclusive, é natural a todos os mamíferos [76], e é uma das principais maneiras pelas quais aprendemos a aperfeiçoar nossos movimentos e ganhar coordenação motora (codificação de mais alto nível e mais eficiente de nossas sequências de ações).

6.2.3 Evolução do córtex

Talvez possa parecer ingênuo tentar buscar uma origem comum a todas as comunidades do córtex, acreditando que todas as áreas seguiriam um mesmo princípio de organização centrado no aprendizado de padrões. Afinal elas são tantas e possuem funções aparentemente tão diversas, e além disso o próprio mecanismo evolutivo acaba por sobrepor tantas inovações sobre outras mais antigas que o cenário real tem tudo para ser muito menos comportado do que conseguimos conceber.

[1] É possível perceber um processo análogo quando aprendemos a tocar um instrumento: começamos dominando apenas movimentos básicos, sem conseguir encadeá-los precisamente. Pouco a pouco, sequências de posições de nossos braços, mãos ou boca inicialmente desajeitadas se tornam automáticas e passamos a tocar o instrumento com mais destreza.

Mas, ao mesmo tempo, o fato de termos uma organização em seis camadas encontrada ao longo de todo o neocórtex (a parte mais recente do córtex, à qual a maioria das áres presentes na rede do Capítulo 2 pertence) mostra que, apesar de tudo, alguns princípios são de fato comuns a todas as áreas. Olhar para a história evolutiva do córtex certamente nos ajudaria a avaliar quão plausíveis são essas teorias.

Entretanto, sabe-se pouco a respeito de sua evolução [77]. Como os tecidos moles não costumam deixar vestígios fósseis, nossa melhor fonte de informação para isso é olhar para o sistema nervoso de animais atuais mas cuja origem é mais primitiva. É o caso dos monotremados, ou mamíferos que põem ovos (como o ornitorrinco e a equidna). Seu córtex visual possui muito menos áreas que o de primatas [78], por exemplo, o que sugere que partimos de um córtex visual primitivo que foi se especializando através da criação de novas áreas para representaçõs mais complexas.

De fato, em [79] especula-se que nossa comunidade visual poderia ter se desenvolvido ao redor de uma área primária original, equivalente ao teto óptico ainda existente nos anfíbios atuais. Em [80], contribuiu-se para essa teoria ao sugerir-se que, nos primatas, áreas visuais mais avançadas (V2, V3, V4, MT) herdaram características estruturais surgidas originalmente no córtex visual primário (V1). Ainda, em [77, 81] e [82] propõm-se, respectivamente, mecanismos genéticos para o surgimento de novas áreas corticais através de duplicação e coalescência, explicando como isso poderia ocorrer em um tempo evolutivo relativamente curto. Todas essas informações corroboram o que sugerimos como princípio básico de organização para gerar a modularidade vista no córtex.

Estamos ainda no começo de uma longa trajetória rumo ao entendimento dos mecanismos por trás da inteligência e dos processos cognitivos manifestados por animais. A natureza evolutiva do surgimento dessas capacidades dá margem a inúmeras dificuldades para sua compreensão, já que adaptações vão sendo adquiridas pelos organismos sem qualquer compromisso com coerência ou elegância matemáticas.

Não obstante, acreditamos que nosso trabalho tenha avançado mais um passo nesse caminho e que ele possa ser usado como inspiração para trabalhos futuros. Somente uma combinação dos conhecimentos e metologias acumulados ao longo do tempo por diferentes áreas do conhecimento poderá dar origem a descrições mais completas dos processos ocorrendo em nosso cérebro.

Referências Bibliográficas

[1] BRESSLER, S. L., MENON, V. "Large-scale brain networks in cognition: emerging methods and principles", *Trends in Cognitive Sciences*, v. 14, pp. 277–290, 2010.

[2] BULLMORE, E., SPORNS, O. "Complex brain networks: graph theoretical analysis of structural and functional systems", *Nature Reviews Neuroscience*, v. 10, pp. 186–198, 2009.

[3] STAM, C. J., VAN STRAATEN, E. C. W. "The organization of physiological brain networks", *Clinical Neurophysiology*, v. 123, pp. 1067–1087, 2012.

[4] SPORNS, O. *Networks of the Brain*. Cambridge, MA, The MIT Press, 2010.

[5] HAGMANN, P., CAMMOUN, L., GIGANDET, X., et al. "Mapping the structural core of human cerebral cortex", *PLoS Biology*, v. 6, pp. e159, 2008.

[6] FELLEMAN, D. J., VAN ESSEN, D. C. "Distributed hierarchical processing in the primate cerebral cortex", *Cerebral Cortex*, v. 1, pp. 1–47, 1991.

[7] SCANNELL, J. W., BLAKEMORE, C., YOUNG, M. P. "Analysis of connectivity in the cat cerebral cortex", *The Journal of Neuroscience*, v. 15, pp. 1463–1483, 1995.

[8] OH, S. W., HARRIS, J. A., NG, L., et al. "A mesoscale connectome of the mouse brain", *Nature*, v. 508, pp. 207–214, 2014.

[9] PARK, H. J., FRISTON, K. "Structural and Functional Brain Networks: From Connections to Cognition", *Science*, v. 342, pp. 1238411, 2013.

[10] ERCSEY-RAVASZ, M., MARKOV, N. T., LAMY, C., et al. "A predictive network model of cerebral cortical connectivity based on a distance rule", *Neuron*, v. 80, pp. 184–197, 2013.

[11] ZAMORA-LÓPEZ, G., ZHOU, C., KURTHS, J. "Cortical hubs form a module for multisensory integration on top of the hierarchy of cortical networks", *Frontiers in Neuroinformatics*, v. 4, 2010.

[12] VAN DEN HEUVEL, M. P., SPORNS, O. "An anatomical substrate for integration among functional networks in human cortex", *The Journal of Neuroscience*, v. 33, pp. 14489–14500, 2013.

[13] MEUNIER, D., LAMBIOTTE, R., BULLMORE, E. T. "Modular and hierarchically modular organization of brain networks", *Frontiers in Neuroscience*, v. 4, pp. 200, 2010.

[14] KAISER, M., MARTIN, R., ANDRAS, P., et al. "Simulation of robustness against lesions of cortical networks", *European Journal of Neuroscience*, v. 25, pp. 3185–3192, 2007.

[15] BASSETT, D. S., BULLMORE, E., VERCHINSKI, B. A., et al. "Hierarchical organization of human cortical networks in health and schizophrenia", *The Journal of Neuroscience*, v. 28, pp. 9239–9248, 2008.

[16] MARKOV, N. T., ERCSEY-RAVASZ, M. M., RIBEIRO GOMES, A. R., et al. "A Weighted and Directed Interareal Connectivity Matrix for Macaque Cerebral Cortex", *Cerebral Cortex*, v. 24, pp. 17–36, 2014.

[17] SCANNELL, J., BURNS, G., HILGETAG, C., et al. "The connectional organization of the cortico-thalamic system of the cat", *Cerebral Cortex*, v. 9, pp. 277–299, 1999.

[18] MODHA, D. S., SINGH, R. "Network architecture of the long-distance pathways in the macaque brain", *Proceedings of the National Academy of Sciences USA*, v. 107, pp. 13485–13490, 2010.

[19] MARKOV, N. T., VEZOLI, J., CHAMEAU, P., et al. "Anatomy of hierarchy: feedforward and feedback pathways in macaque visual cortex", *Journal of Comparative Neurology*, v. 522, pp. 225–259, 2014.

[20] NEWMAN, M. E. J. "Analysis of weighted networks", *Physical Review E*, v. 70, pp. 056131, 2004.

[21] MARKOV, N. T., ERCSEY-RAVASZ, M., VAN ESSEN, D. C., et al. "Cortical high-density counterstream architectures", *Science*, v. 342, pp. 1238406, 2013.

[22] MARKOV, N. T., MISERY, P., FALCHIER, A., et al. "Weight Consistency Specifies Regularities of Macaque Cortical Networks", *Cerebral Cortex*, v. 21, pp. 1254–1272, 2011.

[23] ESSEN, D. C. V., GLASSER, M. F., D. L. DIERKER, J. H. "Cortical parcellations of the macaque monkey analyzed on surface-based atlases", *Cerebral Cortex*, v. 22, pp. 2227–2240, 2012.

[24] HILGETAG, C. C., GRANT, S. "Cytoarchitectural differences are a key determinant of laminar projection origins in the visual cortex", *NeuroImage*, v. 51, pp. 1006–1017, 2010.

[25] BARONE, P., BATARDIERE, A., KNOBLAUCH, K., et al. "Laminar distribution of neurons in extrastriate areas projecting to visual areas V1 and V4 correlates with the hierarchical rank and indicates the operation of a distance rule", *The Journal of Neuroscience*, v. 20, pp. 3263–3281, 2000.

[26] REID, A. T., KRUMNACK, A., WANKE, E., et al. "Optimization of cortical hierarchies with continuous scales and ranges", *NeuroImage*, v. 47, pp. 611–617, 2009.

[27] SPORNS, O. "Network attributes for segregation and integration in the human brain", *Current Opinion in Neurobiology*, v. 23, n. 2, pp. 162–171, 2013.

[28] VAN DEN HEUVEL, M. P., SPORNS, O. "Network hubs in the human brain", *Trends in Cognitive Sciences*, v. 17, pp. 683–696, 2013.

[29] GOULAS, A., SCHAEFER, A., MARGULIES, D. S. "The strength of weak connections in the macaque cortico-cortical network", *Brain Structure and Function*, p. to appear, 2014.

[30] AHN, Y.-Y., BAGROW, J. P., LEHMANN, S. "Link communities reveal multiscale complexity in networks", *Nature*, v. 466, pp. 761–764, 2010.

[31] KARRER, B., LEVINA, E., NEWMAN, M. E. J. "Robustness of community structure in networks", *Physical Review E*, v. 77, pp. 046119, 2008.

[32] LANCICHINETTI, A., FORTUNATO, S., KERTÉSZ, J. "Detecting the overlapping and hierarchical community structure in complex networks", *New Journal of Physics*, v. 11, pp. 033015, 2009.

[33] ASHBY, W. R. "Principles of the Self-Organizing System". In: von Foerster, H., Zopf, Jr., G. W. (Eds.), *Principles of Self-Organization: Transactions of the University of Illinois Symposium*, Pergamon Press, pp. 255–278, New York, NY, 1962.

[34] KNIERIM, J. J., VAN ESSEN, D. C. "Visual cortex: cartography, connectivity, and concurrent processing", *Current Opinion in Neurobiology*, v. 2, pp. 150–155, 1992.

[35] RUSSELL, S., NORVIG, P. *Artificial Intelligence: a Modern Approach*. New Jersey, Prentice Hall, 1995.

[36] AREL, I., ROSE, D. C., KARNOWSKI, T. P. "Deep machine learning—a new frontier in artificial intelligence research", *Computational Intelligence Magazine, IEEE*, v. 5, pp. 13–18, 2010.

[37] KOHONEN, T. *Associative memory: A system-theoretic approach*. West Berlin, Springer-Verlag, 1997.

[38] RUMELHART, D. E., ZIPSER, D. "Competitive learning", *Cognitive Science*, v. 9, pp. 75–112, 1985.

[39] MCCLELLAND, J. L., RUMELHART, D. E. "Distributed Memory and the Representation of General and Specific Information", *Journal of Experimental Psychology: General*, v. 114, pp. 159–188, 1985.

[40] JOANISSE, M. F., MCCLELLAND, J. L. "Connectionist perspectives on language learning, representation and processing", *WIREs Cogn Sci*, v. 435, pp. 1102–1107, 2005.

[41] GASSER, M., COLUNGA, E. "Pattern learning in infants and neural networks". In: Quinlan, P. T. (Ed.), *Connectionist models of development: Developmental processes in real and artificial neural networks*, Psychology Press, pp. 233–255, New York, 2003.

[42] DOUMAS, L. A. A., HUMMEL, J. E., SANDHOFER, C. M. "A Theory of the Discovery and Predication of Relational Concepts", *Psychological Review*, v. 115, pp. 1–43, 2015.

[43] RUMELHART, D. E., MCCLELLAND, J. L. "On learning the past tenses of English verbs". In: Rumelhart, D. E., McClelland, J. L. (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. II*, MIT Press, pp. 216–271, Cambridge, MA, 1986.

[44] WEINER, P. "Linear pattern matching". In: *IEEE 14th Annual Symposium on Switching and Automata Theory*, pp. 1–11, 1973.

[45] UKKONEN, E. "On–line construction of suffix trees", *Algorithmica*, v. 14, pp. 249–260, 1995.

[46] DEUTSCHER, G. *The Unfolding of Language: An Evolutionary Tour of Mankind's Greatest Invention*. New York, Metropolitan Books, 2005.

[47] UKKONEN, E., LEMSTRÖM, K., MÄKINEN, V. "Geometric Algorithms for Transposition Invariant CBMR". In: *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR 2003)*, pp. 193–199, 2003.

[48] BELL, T. C., CLEARY, J. G., WITTEN, I. H. *Text compression*. New Jersey, Prentice Hall, 1990.

[49] WITTEN, I. H., NEAL, R. M., CLEARY, J. G. "ARITHMETIC CODING FOR DATA COIUPRESSION", *Communications of the ACM*, v. 30, pp. 520–540, 1987.

[50] MACKAY, D. J. C. *Information Theory, Inference, and Learning Algorithms*. Cambridge, Cambridge University Press, 2003.

[51] J.CLEARY, WITTEN, I. . "Data compression using adaptive coding and partial string matching", *IEEE Trans. Commun.*, v. COM–32, pp. 396–402, 1984.

[52] SHKARIN, D. "PPM: One step to practicality". In: *Data Compression Conference*, pp. 202–212, 2002.

[53] CLEARY, J. G., TEAHAN, W. J. "Unbounded length contexts for PPM", *The Computer Journal*, v. 40, pp. 67–75, 1997.

[54] BURROWS, M., WHEELER, D. J. *A block-sorting lossless data compression algorithm*. Palo Alto, CA, Digital Equipment Corporation, 1994.

[55] WOOD, F., GASTHAUS, J., ARCHAMBEAU, C., et al. "Unbounded length contexts for PPM", *Communications of the ACM*, v. 54, pp. 91–98, 2011.

[56] ZIV, J., LEMPEL, A. "A universal algorithm for sequential data compression", *Communications of the ACM*, v. IT–23, pp. 337–343, 1977.

[57] GALES, M., YOUNG, S. "The Application of Hidden Markov Models in Speech Recognition Foundations and Trends in Signal Processing", *Brain Research*, v. 1, pp. 195–304, 2007.

[58] CLEARY, J. G., TEAHAN, W. J., WITTEN, I. H. "Unbounded Length Contexts for PPM". In: *Proceedings of the Conference on Data Compression*, pp. 52–. IEEE Computer Society, 1995.

[59] ATKINSON, R. C., SHIFFRIN, R. M. "Human memory: A proposed system and its control processes". In: Spence, K. W., Spence, J. T. (Eds.), *The*

psychology of learning and motivation (Volume 2), Academic Press, pp. 89–195, New York, 1968.

[60] BADDELEY, A. "Working memory: looking back and looking forward", *Nat. Rev. Neurosci.*, v. 4, pp. 829–839, 2003.

[61] QUIROGA, R. Q., REDDY, L., KREIMAN, G., et al. "Invariant visual representation by single neurons in the human brain", *Nature*, 2015. doi: 10.1002/wcs.1340.

[62] HERZ, R. S., ELIASSEN, J., BELAND, S., et al. "Neuroimaging evidence for the emotional potency of odor-evoked memory", *Neuropsychologia*, v. 42, pp. 371–378, 2004.

[63] HEBB, D. *The Organization of Behavior*. New York, John Wiley and Sons, 1949.

[64] MINSKY, M. "K-lines: A Theory of Memory", *Cognitive Science*, v. 4, pp. 117–133, 1980.

[65] HOPFIELD, J. "Neural networks and physical systems with emergent collective computational properties", *Proc. Natl. Acad. Sci. U. S. A.*, v. 81, pp. 3088–3092, 1982.

[66] JACOMY, M., VENTURINI, T., HEYMANN, S., et al. "ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualization Designed for the Gephi Software", *PLOS One*, 2014. doi: 10.1371/journal.pone.0098679.

[67] XIE, J., SZYMANSKI, B. K., , et al. "SLPA: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process". In: *Proc. ICDM Workshop*, pp. 344–349, 2011.

[68] BIEMANN, C. "Chinese Whispers: An Efficient Graph Clustering Algorithm and Its Application to Natural Language Processing Problems". In: *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*, pp. 73–80, 2006.

[69] PONS, P., LATAPY, M. "Computing communities in large networks using random walks", *pre-print*, 2006. arXiv:cs/0608050.

[70] GIRVAN, M., NEWMAN, M. E. J. "Community structure in social and biological networks", *Proc. Natl. Acad. Sci. USA*, v. 99, pp. 7821—-7826, 2002.

[71] WATTS, D. J., STROGATZ, S. "Collective dynamics of 'small-world' networks", *Nature*, v. 393, pp. 440–442, 1998.

[72] KOCH, C., SEGEV, I. *Methods in Neuronal Modeling: From Ions to Networks*. Cambridge, MA, MIT Press, 1998.

[73] MCCULLOCH, W. S., PITTS, W. H. "A Logical Calculus of the Ideas Immanent in Nervous Activity", *Bulletin of Mathematical Biophysics*, v. 7, pp. 115–133, 1943.

[74] PURVES, D., AUGUSTINE, G. J., FITZPATRICK, D., et al. *Neuroscience*. Sunderland, MA, Sinauer Associates, 2012.

[75] NIV, Y. "Reinforcement learning in the brain", *J. Math. Psychol.*, v. 53, pp. 139–154, 2009.

[76] PANKSEPP, J. *Affective Neuroscience: The Foundations of Human and Animal Emotions*. Oxford, Oxford University Press, 1998.

[77] ALLMAN, J. M. *Evolving Brains*. New York, Scientific American Library - W. H. Freeman, 2000.

[78] KRUBITZER, L. "What can monotremes tell us about brain evolution?" *Philos Trans R Soc Lond B Biol Sci*, v. 353, pp. 1127–1146, 1998.

[79] ALLMAN, J. M. "The origin of the neocortex", *Seminars in The Neurosciences*, v. 2, pp. 257–262, 1990.

[80] KAAS, J. H. "Evolution of columns, modules, and domains in the neocortex of primates", *Proc Natl Acad Sci U. S. A.*, v. 109, pp. 10655–10660, 2012.

[81] ALLMAN, J. M., KAAS, J. H. "A crescent-shaped cortical visual area surrounding the middle temporal area (MT) in the owl monkey (Aotus trivirgatus)", *Brain Research*, v. 81, pp. 199–213, 1974.

[82] SERENO, M., ALLMAN, J. M. "Cortical Visual Areas in Mammals". In: Cronly-Dillon, J. (Ed.), *Neural Basis of Visual Function*, C R C Press, pp. 160–172, Boca Raton, 1991.

Apêndice A

Métodos

A fórmula para cálculo da similaridade, S, entre duas arestas e_{ik} e e_{jk} em uma rede não-direcionada, sem pesos e sem *self-loops*, é

$$S(e_{ik}, e_{jk}) = \frac{|n_+(i) \cap n_+(j)|}{|n_+(i) \cup n_+(j)|}, \tag{A.1}$$

onde $n_+(i)$ é o conjunto de vizinhos do nó i, incluindo o próprio i (ou, simplesmente, o conjunto dos chamados "vizinhos inclusivos" de i). A razão para incluirmos o nó em sua própria vizinhança fica clara quando dois nós i e j compartilham exatamente os mesmos vizinhos: sem a inclusão, S seria igual a 1 independentemente do número de vizinhos em comum.

A presença de pesos nas arestas pode ser levada em conta utilizando-se uma versão da Equação (A.1) em forma de vetor,

$$S(e_{ik}, e_{jk}) = \frac{a_i \cdot a_j}{|a_i|^2 + |a_j|^2 - a_i \cdot a_j}, \tag{A.2}$$

onde a_i é a i-ésima linha da matriz de adjacências com pesos (exceto pelo elemento da diagonal). Isto é, se N é o número de nós na rede, então $a_i = (w_{i1}, w_{i2}, \ldots, w_{iN})$, com cada elemento sendo referente ao peso da aresta correspondente (exceto por w_{ii}, o qual discutiremos a seguir).

Para sermos coerentes com o uso dos "vizinhos inclusivos" na Equação (A.1), precisamos escolher um valor adequado para w_{ii}. Os autores em [30] propuseram que esse elemento fosse a média aritmética dos pesos das arestas incidentes a i. Faremos uso de um exemplo ilustrativo para sugerir que uma opção melhor seria o peso máximo entre as arestas incidentes a i.

Considere uma rede simples composta de três nós, i, j e k, e de duas arestas, e_{ik} e e_{jk}, ambas de peso unitário. Como i e j não possuem qualquer vizinho em comum além de k, temos $S(e_{ik}, e_{jk}) = 0.333$, já que $w_{ii} = 1$ e $w_{jj} = 1$ qualquer que seja o método escolhido para lidar com esses pesos. Entretanto, se adicio-

narmos um nó l e uma aresta e_{il} a essa rede, os dois métodos fornecerão valores distintos para $S(e_{ik},\ e_{jk})$, dependendo do peso w_{il}. Se $w_{il} = 1$, ambos os métodos resultam em $S(e_{ik}, e_{jk}) = 0.25$—o que faz todo sentido, já que i agora possui um vizinho não adjacente a j. Mas, se fizermos $w_{il} = 0.1$, por exemplo, então usando o peso médio $w_{ii} = (1 + 0.1)/2 = 0.55$ teríamos $S(e_{ik}, e_{jk}) = 0.43$, um valor ainda maior do que a similaridade original de 0.333 quando i e j tinham a mesma vizinhança "não-inclusiva". Usando o peso máximo, $w_{ii} = \max\{1, 0.1\} = 1$, teremos $S(e_{ik}, e_{jk}) = 0.332$, que é ligeiramente menor devido à nova aresta introduzida. Em outras palavras, o uso do peso máximo reflete assimetrias nas vizinhanças dos dois nós, levando em consideração a magnitude dos pesos nessas vizinhanças de forma mais razoável. Portanto, doravante usaremos

$$w_{ii} = \max_{i' \in n(i)} w_{ii'}, \tag{A.3}$$

onde $n(i)$ é a vizinhança ("não-inclusiva") de i.

Agora, para incorporar direções nos cálculos acima, acreditamos que vizinhos de mesma identidade mas interagindo em direções diferentes devem ser tratados como vizinhos distintos. Afinal, suponha que estejamos calculando a similaridade entre duas arestas direcionadas e_{ik} e e_{jk}, e que tanto i quanto j têm l como vizinho, mas em direções opostas (ou seja, l é vizinho de entrada de i e vizinho de saída de j). O valor de similaridade deveria ser menor do que se l interagisse com i e j na mesma direção.

Portanto, uma maneira direta de adaptar a Equação (A.1) à presença de direções é fazer

$$S(e_{ik}, e_{jk}) = S(e_{ki}, e_{kj}) = \frac{|n_+^{\text{in}}(i) \cap n_+^{\text{in}}(j)| + |n_+^{\text{out}}(i) \cap n_+^{\text{out}}(j)|}{|n_+^{\text{in}}(i) \cup n_+^{\text{in}}(j)| + |n_+^{\text{out}}(i) \cup n_+^{\text{out}}(j)|}, \tag{A.4}$$

ou, em forma de vetor,

$$S(e_{ik}, e_{jk}) = \frac{a_i^{\text{in}} \cdot a_j^{\text{in}} + a_i^{\text{out}} \cdot a_j^{\text{out}}}{|a_i^{\text{in}}|^2 + |a_j^{\text{in}}|^2 - a_i^{\text{in}} \cdot a_j^{\text{in}} + |a_i^{\text{out}}|^2 + |a_j^{\text{out}}|^2 - a_i^{\text{out}} \cdot a_j^{\text{out}}}, \tag{A.5}$$

onde ambas as arestas têm a mesma direção (isto é, para k ou vindo de k).

Quanto a pares de arestas possuindo direções distintas com relação ao nó que elas têm em comum, nenhum deles é levado em conta em nossos cálculos. Tomamos essa decisão porque parece razoável que tais arestas não devam possuir similaridades maiores que qualquer outro par de arestas incidentes a um mesmo nó na mesma direção.

Após calcular os valores de similaridade para os pares de arestas relevantes no *dataset*, utilizamos o agrupamento hierárquico tipo *single-linkage* para construir um

dendrograma de arestas (durante esse processo, quando há arestas empatadas incorporamos ambas simultaneamente). As comunidades resultantes foram selecionadas no ponto de máxima densidade de partição, D, conforme proposto por [30]. Essa densidade é dada por

$$D = \frac{1}{M} \sum_{c} m_c D_c, \tag{A.6}$$

onde M é o número de arestas da rede, c representa cada uma das comunidades, e D_c, após uma adaptação direta para o caso direcionado, é dado por

$$D_c = \frac{m_c - (n_c - 1)}{n_c(n_c - 1) - (n_c - 1)}. \tag{A.7}$$

Nessa expressão, m_c e n_c são, respectivamente, o número de arestas e nós na comunidade c. Então, D é o valor médio de D_c, com cada comunidade ponderada pela fração de M a que correspondem suas arestas. Para entender melhor o significado da quantidade D_c, note que $n_c - 1$ é o número mínimo de arestas requerido para que n_c nós estejam conectados em uma única componente. Portanto, D_c pode ser visto como o número de arestas que uma comunidade c possui além desse mínimo, normalizado com relação ao máximo excesso que pode haver (ou seja, se todas as $n_c(n_c - 1)$ arestas direcionadas possíveis estão presentes). Uma discussão detalhada de comunidades de arestas é dada por [30].