MODELOS BASEADOS EM REDES NEURAIS NÃO-SUPERVISIONADAS PARA APLICAÇÃO EM PROBLEMAS DE SPIKE SORTING

Francinei Gomes de Morais

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Elétrica.

Orientador: José Gabriel Rodríguez Carneiro Gomes

Rio de Janeiro
Março de 2018

MODELOS BASEADOS EM REDES NEURAIS NÃO-SUPERVISIONADAS PARA APLICAÇÃO EM PROBLEMAS DE SPIKE SORTING

Francinei Gomes de Morais

TESE 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 DOUTOR EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

Prof. José Gabriel Rodríguez Carneiro Gomes, Ph.D.

Prof. Mariane Rembold Petraglia, Ph.D.

Prof. Sergio Lima Netto, Ph.D.

Prof. Juliana Guimarães Martins Soares, D.Sc.

Prof. Marley Maria Bernardes Rebuzzi Vellasco, Ph.D.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2018

Morais, Francinei Gomes de

Modelos baseados em Redes Neurais Não-Supervisionadas para Aplicação em Problemas de Spike Sorting/Francinei Gomes de Morais. – Rio de Janeiro: UFRJ/COPPE, 2018.

XVI, 84 p.: il.; 29, 7cm.

Orientador: José Gabriel Rodríguez Carneiro Gomes

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2018.

Referências Bibliográficas: p. 61 – 69.

1. Redes Neurais Artificiais. 2. Kullback-Leibler. 3. Spike Sorting. I. Gomes, José Gabriel Rodríguez Carneiro. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

"O homem só está completo quando
encontra a sua metade".
À minha esposa Sabrina,
Aos meus filhos Arthur e Bernardo.

Agradecimentos

Agradeço a Deus por estar sempre ao meu lado e ter posto em minha vida pessoas maravilhosas que me ajudaram a colocar mais um tijolo no muro do conhecimento. Essas pessoas são meu orientador José Gabriel Rodríguez Carneiro Gomes, todos os professores que tive até hoje, meus amigos da Universidade Federal do Rio de Janeiro (UFRJ), Roberto Campos, Edgar Medina, Pedro Riascos, José Motta, Juvisan Aguedo e Evelyn, e meus amigos pessoais Mariana, Wilson, Rafael e Taísa.

Gostaria de agradecer ao Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia (COPPE), aos funcionários do Programa de Engenharia Elétricas (PEE), ao Laboratório de Processamento Analógico e Digital de Sinais (PADS), e ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), por seu apoio financeiro.

Por último, meus especiais agradecimentos vão para toda a minha família, minha mãe Inês Berto, meu pai Francisco Gomes, minha irmã Elizabeth Gomes, minha sogra Liana Ribeiro, minha esposa Sabrina Ribeiro e aos meus filhos Arthur Ribeiro e Bernardo Ribeiro, que Deus me deu ao longo do Doutorado.

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

MODELOS BASEADOS EM REDES NEURAIS NÃO-SUPERVISIONADAS PARA APLICAÇÃO EM PROBLEMAS DE SPIKE SORTING

Francinei Gomes de Morais

Março/2018

Orientador: José Gabriel Rodríguez Carneiro Gomes

Programa: Engenharia Elétrica

Registros extracelulares são compostos por sinais pulsados (*spikes*) gerados por vários neurônios biológicos. A correta detecção e separação destes *spikes* é chamada de *spike sorting*. A tarefa de *spike sorting* é de grande importância para estudos que são baseados na análise de sinais de *spike*. Muitos métodos de *spike sorting* têm sido propostos, mas ainda não há um método universalmente adotado. Nesta tese é proposto um método baseado em perceptrons multicamadas (MLPs) que são treinados, de modo não supervisionado, usando a divergência de Kullback-Leibler (KLD) como função-custo para resolver o problema de *spike sorting*. O algoritmo proposto MLP-KLD aprende um mapeamento, a partir do espaço de dados original para um espaço com duas dimensões, capaz de revelar os agrupamentos implícitos no conjunto de dados original. Os algoritmos *t-distributed stochastic neighbor embedding* (t-SNE), *principal component analysis* (PCA), *linear discriminant analysis* (LDA) e *locality preserving projections* (LPP) foram tomados como referência de comparação. Para as comparações de desempenho foram utilizadas duas bases de dados simulados, publicamente disponíveis: a primeira contém *spikes* simulados por um grupo de 2 até 20 neurônios biológicos, e a segunda contém *spikes* simulados por um grupo de três neurônios biológicos sob diferentes condições de ruído. Os mapeamentos para o espaço 2-D (onde a clusterização é realizada através do algoritmo K-means) via MLP-KLD e t-SNE são significativamente melhores. O desempenho da clusterização para mapeamentos 2-D via MLP-KLD é mantido conforme o número de neurônios ou nível de ruído aumenta, o que indica que o método proposto é potencialmente útil para aplicações de análise de *spikes* baseadas em *spike sorting*.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

UNSUPERVISED NEURAL NETWORKS-BASED MODELS APPLIED TO PROBLEMS IN SPIKE SORTING

Francinei Gomes de Morais

March/2018

Advisor: José Gabriel Rodríguez Carneiro Gomes

Department: Electrical Engineering

Extracellular recordings contain neural spike signals that come from different biological neurons. The proper detection and separation of the spikes according to the neurons they originate from is usually referred to as spike sorting. The spike sorting task is crucial for subsequent studies that are based on spike analysis. Many spike sorting methods have been proposed, but universally adopted methods are not available yet. In this thesis, we introduce multilayer perceptrons (MLPs) that are trained, in unsupervised fashion, to minimize the Kullback-Leibler divergence (KLD) between original data and low-dimensional data probability distributions, thus leading to a low-dimensional data representation from which spike sorting problems are efficiently solved. More specifically, the proposed KLD-MLP algorithm learns a map from the original data space to a 2-D space where otherwise implicit spike clusters are revealed. For overall spike sorting performance comparison, four other algorithms have been applied at the data mapping stage: t-distributed stochastic neighbor embedding (t-SNE), principal component analysis (PCA), linear discriminant analysis (LDA), and locality preserving projections (LPP). The performance comparison is based on two publicly available synthetic datasets: the first one contains simulated spikes from a number of biological neurons ranging from two to twenty, and the second one contains simulated spikes from three neurons under different noise conditions. The KLD-MLP and t-SNE approaches yield significantly improved maps into the two-dimensional space where clustering is performed based on conventional K-means. The performance of basic K-means clustering based on KLD-MLP maps is maintained as the number of neurons or noise level are increased, which indicates that the method is potentially useful for spike analysis applications based on spike sorting.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O desenvolvimento de sistemas inteligentes, baseados em computador, que sejam capazes de substituir ou ao menos auxiliar o homem na execução de tarefas complexas como a realização de diagnósticos médicos a partir da análise de sinais fisiológicos, ou o reconhecimento de pessoas em vídeos, tem sido um dos maiores desafios enfrentados por diversas áreas de pesquisa.

Nos últimos anos, com os avanços nas pesquisas e a proposição de novos algoritmos, muitos trabalhos que pareciam apenas um conjunto de hipóteses estão se tornando realidade. Já é possível encontrar sistemas baseados em imagens que são capazes de detectar pessoas, objetos ou até mesmo descrever o conteúdo da imagem (*image captioning*) [1, 2]. Têm surgido sistemas de assistência virtual que interagem automaticamente, estabelecendo conversações simples, ou até mesmo sugerindo conteúdo, baseando-se para isso no conjunto de palavras utilizadas durante a conversação. Na área médica os avanços continuam, com o desenvolvimento de sistemas de análise de imagens para auxiliar na detecção de doenças como câncer de pulmão [3] e câncer de pele [4]. Sistemas baseados em inteligência artificial que ajudam na interpretação de exames médicos, minimizando as chances de erros em diagnósticos, e permitindo que os tratamentos sejam realizados de forma mais rápida, precisa e personalizada.

Grande parte do sucesso desses sistemas inteligentes advém do uso de redes neurais artificiais para realizar sua etapa de inteligência. A popularização no uso das redes neurais tem como principais fatores: a) evolução do poder computacional do *hardware* utilizado para o processamento dos algoritmos de redes neurais; b) proposição de novos algoritmos, arquiteturas e topologias de redes neurais; e c) o aumento do volume e da variedade dos dados produzidos pelos dispositivos eletrônicos atuais. O desempenho e a complexidade que as redes neurais podem atingir estão diretamente relacionados à quantidade e à qualidade dos dados disponíveis.

Nesta tese é proposto um novo algoritmo de aprendizado utilizado para o treinamento de redes neurais artificiais do tipo perceptron multicamadas (*multilayer*

perceptron – MLP). O algoritmo utiliza a divergência de Kullback-Leibler (KLD) como função-custo a ser minimizada durante o processo de ajuste dos parâmetros do MLP. A esse algoritmo foi dado o nome de MLP-KLD.

Os MLPs treinados via algoritmo MLP-KLD são interpretados como funções de mapeamento $f : X \to Y$, que para cada padrão de entrada $\mathbf{x} \in \mathbb{R}^D$ geram um padrão de saída $\mathbf{y} \in \mathbb{R}^L$, onde $\mathbf{y}$ corresponde ao vetor de características aprendidas a partir do padrão de entrada $\mathbf{x}$. Os mapeamentos realizados pelos MLPs são do tipo não-linear e podem se tornar mais complexos à medida em que são acrescidos o número de camadas ou de neurônios artificiais por camada nos MLPs.

Tradicionalmente os algoritmos de aprendizado utilizados para o treinamento de redes neurais adotam como funções-custo as funções erro médio quadrático ou entropia cruzada. Durante a elaboração desta tese, foram encontrados outros trabalhos que também adotam a divergência de Kullback-Leibler como função-custo para o treinamento de modelos baseados em redes neurais. Em [5] foi proposto um algoritmo para o treinamento de modelos formados pelo empilhamento de pequenas redes neurais do tipo máquina de Boltzmann restrita (*restricted Boltzmann machine* – RBM), os modelos foram avaliados na tarefa de redução de dimensionalidade a partir de bases de dados de imagens e bases de dados de texto. Em [6] foi proposto um algoritmo para o treinamento de modelos formados pelo empilhamento de pequenas redes neurais do tipo autoencoder, os modelos foram avaliados na tarefa de clusterização a partir de bases de dados de imagens e bases de dados de texto. É possível também encontrar algoritmos não baseados em redes neurais que utilizam a divergência de Kullback-Leibler nas mais diversas tarefas tais como classificação de textos [7], análise de sinais de eletroencefalograma (EEG) [8], análise de sentimentos [9], separação cega de fontes [10], inferência variacional [11], rastreamento de objetos em imagens [12] e remoção de ruído de *Poisson* em imagens [13].

Nesta tese os MLPs treinados via MLP-KLD serão avaliados na tarefa de *spike sorting* [14]. Será mostrado que é possível treinar MLPs contendo um grande número de camadas e que para o MLP aprender o mapeamento não é necessário o conhecimento do neurônio biológico[1] que produziu o *spike* a ser mapeado, ou seja, o aprendizado é não supervisionado. Será mostrado também que o mapeamento aprendido via MLP-KLD é capaz de revelar possíveis clusteres que existam naturalmente no conjunto de *spikes* utilizados para o treinamento dos MLPs.

[1]Nesta tese o nome "neurônio artificial"é utilizado como referência aos neurônios que constituem as redes neurais artificiais, enquanto que o nome "neurônio biológico"é utilizado como referência aos neurônios que produzem as atividades neurais que compõem os registros extracelulares.

1.1 Contribuições desta Tese

As contribuições mais significativas obtidas a partir dos estudos desenvolvidos ao longo da tese são:

1. É proposto um algoritmo de aprendizado baseado em perceptron multicamadas (MLPs) treinados, de modo não supervisionado, usando a divergência de Kullback-Leibler como função-custo, para resolver o problema de *spike sorting*. A maior parte dos trabalhos que utilizam redes neurais para a tarefa de *spike sorting* realizam o treinamento das redes neurais de modo supervisionado, o que demanda tempo e profissionais qualificados para a correta identificação dos alvos (identificar o tipo de neurônio que gerou cada um dos *spikes* do conjunto de dados utilizado para o treinamento do MLP). Talvez seja esse um dos principais motivos para o pequeno número de trabalhos, publicados ao longo dos últimos anos, que aplicam redes neurais para a tarefa de *spike sorting*. Desta forma, consideramos que a proposição de um método baseado em MLPs treinados de modo não supervisionado para resolver o problema de *spike sorting* contribui para estender a área de aplicação das redes neurais e incentivar o desenvolvimento de novos métodos de *spike sorting* baseados em redes neurais;

2. Para lidar com o problema de conjuntos de dados que apresentam desbalanceamento significativo nas contagens de *spikes* gerados por diferentes classes de neurônios, foi proposta a métrica DCM (*data-counting metric*). Os resultados obtidos via DCM, quando comparados aos resultados obtidos via métrica AMI (*adjusted mutual information*), mostram que a representação gráfica fornecida pela DCM torna mais evidente a variação de desempenho dos sistema de *spike sorting* à medida em que os problemas ficam mais difíceis (com mais neurônios biológicos, ou com mais ruído).

1.2 Resumo dos Capítulos Restantes

O Capítulo 2 apresenta uma visão geral sobre o processo de aquisição e classificação da atividade neural produzida por diferentes tipos de neurônios. A Seção 2.1 mostra um esquema básico para aquisição de registros extracelulares e fala sobre os elementos que compõem um registro extracelular. A Seção 2.2 apresenta o problema de *spike sorting*, as etapas realizadas por um sistema de detecção e classificação de *spikes*, e descreve alguns dos problemas tipicamente encontrados. A Seção 2.3 trata sobre as limitações dos experimentos realizados e dos algoritmos avaliados.

O Capítulo 3 apresenta uma visão geral sobre algoritmos de aprendizados de máquina. Para uma melhor descrição dos algoritmos optou-se por dividi-los nas categorias *supervisionados* e *não supervisionados*. Os algoritmos apresentados no Capítulo 3 são utilizados como técnicas para o aprendizado de funções de mapeamento. O objetivo da função de mapeamento aprendida é, para um determinado conjunto de dados, gerar uma nova representação que favoreça a etapa de clusterização dos sistemas de *spike*. O problema da complexidade das funções de mapeamento é discutido ao longo da Seção 3.1.1.

O Capítulo 4 apresenta um novo algoritmo para o treinamento de redes neurais do tipo perceptron multicamadas (MLP) utilizadas para a tarefa de *spike sorting*. O treinamento do MLP com função-custo KLD é apresentado na Seção 4.1. O critério adotado para a escolha do algoritmo de clusterização é apresentado na Seção 4.2. As métricas de desempenho utilizadas nesta tese são apresentadas na Seção 4.3.

O Capítulo 5 apresenta todos os experimentos realizados nesta tese. A Seção 5.1 avalia o desempenho dos algoritmos de *spike sorting* em condições de registros neurais contendo até 20 neurônios biológicos. A Seção 5.2 avalia o desempenho dos algoritmos de *spike sorting* aplicados a registros neurais contendo nível de ruído de fundo não desprezível. O experimento apresentado na Seção 5.3 avalia se existem espaços de mapeamento que tenham mais do que duas dimensões e que favoreçam o processo de *spike sorting*. O experimento apresentado na Seção 5.4 investiga se o aumento da complexidade da função de mapeamento implementada pelo MLP treinado via MLP-KLD favorece a tarefa de *spike sorting*. O algoritmo MLP-KLD proposto inicialmente utiliza a métrica Euclideana para a construção das funções de distribuição de probabilidade comparadas via KLD. O experimento apresentado na Seção 5.5 avalia a influência da substituição da métrica Euclideana pelas métricas (Cosseno ou *City Block*) sobre o desempenho dos MLPs treinados via MLP-KLD.

O Capítulo 6 apresenta as conclusões do presente trabalho e discute possíveis caminhos para futuras pesquisas.

Capítulo 2

Estudo da Atividade Neural via Registros Extracelulares

Durante décadas as pesquisas científicas têm utilizado sinais eletrofisiológicos para estudar as propriedades elétricas de células e tecidos. O primeiro trabalho estabelecendo uma relação entre estímulos elétricos e o processo de comunicação neural foi apresentado em [15]. O trabalho mostrou que, através de eletroestimulações, era possível induzir a contração muscular da perna de uma rã. Somente após 1920 foi possível medir a atividade elétrica das células do sistema nervoso por meio de circuitos de amplificação. Em [16], os estudos realizados produziram descobertas consideradas inovadoras acerca do movimento iônico através das membranas das células nervosas durante a geração da atividade elétrica/neural celular; e [17] foi o primeiro trabalho a mostrar como as atividades elétricas de neurônios individuais contribuíam para o processamento da visão. O objetivo deste capítulo é apresentar uma visão geral sobre os desafios relacionados ao estudo da atividade neural via registros extracelulares. Para isso, este capítulo está organizado como segue.

A Seção 2.1 apresenta um esquema básico para aquisição de registros extracelulares e também apresenta os elementos que compõem o registro extracelular. Problemas típicos em *spike sorting* e as etapas desenvolvidas pelos sistemas de detecção e classificação de *spikes* são discutidos na Seção 2.2. Limitações dos algoritmos avaliados são discutidas na Seção 2.3.

2.1 Medição da Atividade Neural

O progresso das pesquisas científicas que envolvem o estudo da atividade neural está diretamente relacionado à qualidade e quantidade de sinais gravados simultaneamente pelos sistemas de aquisição de sinais. Os primeiros sistemas de aquisição de sinais desenvolvidos eram capazes de registrar sinais obtidos através de apenas

um eletrodo de aquisição, e a qualidade dos sinais registrados permitia detectar no máximo cinco neurônios [18]. A tecnologia atual dos sistemas de aquisição permite que sejam registrados simultaneamente sinais obtidos por conjuntos de eletrodos contendo dezenas ou até centenas de eletrodos de aquisição [19].

A atividade neural pode ser medida em relação à região interna da célula (registro intracelular) ou em relação à região externa da célula (registro extracelular). A maior facilidade em se obter os registros extracelulares, se comparados aos registros intracelulares, tem favorecido a popularização do uso desse tipo de registro ao longo dos últimos anos. Os registros extracelulares (objeto de estudo desta tese), são obtidos através da inserção de eletrodos de aquisição no tecido do cérebro para gravar a atividade elétrica dos neurônios ao redor dos eletrodos [20]. A Figura 2.1 apresenta um esquema básico para aquisição de registros extracelulares. O formato e a composição física do eletrodo de aquisição exercem influência sobre quais tipos de sinais podem ser medidos.

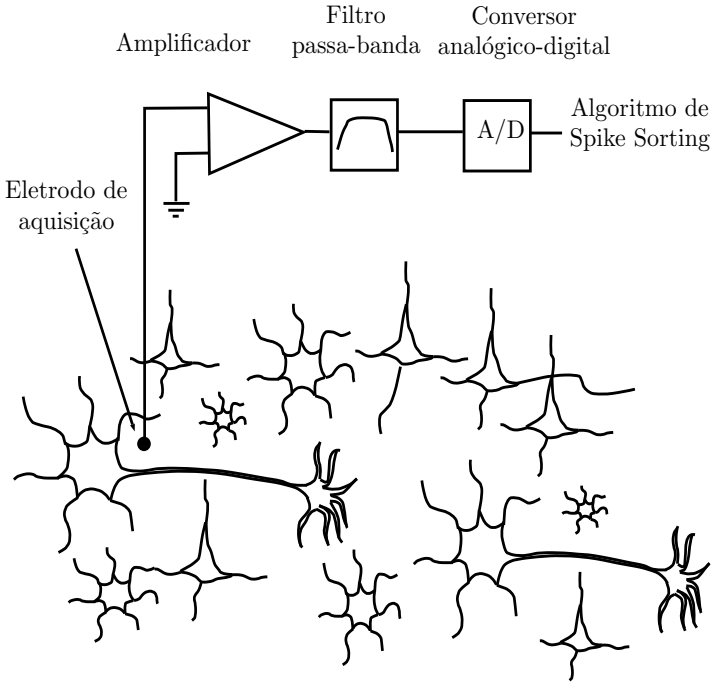


Figura 2.1: Esquema básico para aquisição de registros extracelulares.

Composição do Registro Extracelular

O registro extracelular é composto por ruído de fundo, potencial de campo local e potenciais de ação (*spikes*) dos tipos *multiunit* e *single unit* [21]. Usualmente os registros neurais são submetidos às etapas de filtragem no domínio analógico e conversão analógico-digital através de um processo de amostragem. As taxas de amostragem tipicamente utilizadas variam de 20 a 30 kHz. Existem diversos mecanismos neurais envolvidos no processo de geração das componentes de frequência

presentes no registro extracelular. Do ponto de vista das componentes de frequência, as frequências abaixo de 300 Hz, entre 300 Hz e 5 kHz, e acima de 5 kHz são consideradas, respectivamente, potencial de campo local, potencial de ação e ruído de fundo [22].

Os potenciais de ação do tipo *multiunit* correspondem a *spikes* gerados por neurônios considerados distantes do "eletrodo de aquisição". O meio extracelular percorrido pelos potenciais de ação até alcançarem o eletrodo de aquisição é responsável pela atenuação do sinal; desse modo, quanto maior for a distância entre o neurônio e o eletrodo de aquisição, maior será a atenuação sofrida pelo potencial de ação. Por conta da atenuação do sinal ao longo do percurso percorrido até alcançar o eletrodo de aquisição, os potenciais de ação do tipo *multiunit*, em muitos casos, podem ser confundidos com o ruído de fundo. Os potenciais de ação do tipo *single unit* correspondem aos *spikes* gerados por neurônios próximos ao eletrodo de aquisição. A proximidade com o eletrodo de aquisição permite que estes *spikes*, mesmo após a atenuação produzida pelo meio extracelular, sejam registrados com amplitude suficiente para serem selecionados na etapa de detecção de *spikes*, realizada pelos sistemas de processamento de sinais de *spike* (sistemas de *spike sorting*) [23].

2.2 *Spike Sorting*

O estudo de registros extracelulares e a correta detecção e separação (*clustering*) dos sinais de *spike* são muito importantes para a compreensão dos mecanismos de processamento cerebral, assim como do processo de codificação e decodificação da informação [24, 25]. Em outras palavras, os estudos das atividades elétricas produzidas por um neurônio, ou por uma população de neurônios, permitiram ampliar o conhecimento sobre as funções de alta complexidade executadas pelo cérebro, tais como percepção, compreensão, movimento e memória. Esse conhecimento tem levado a novas possibilidades de tratamento para problemas como epilepsia, paralisia e perda de memória, e também tem sido utilizado para o desenvolvimento de tecnologias como *brain machine interfaces* (BMI) [26] e próteses neurais [27, 28].

De forma simplificada, o *spike sorting* é o processo de detectar os sinais de *spike* e separá-los em grupos, com base em medidas de similaridade, de modo que cada grupo inclua *spikes* gerados por uma única classe de neurônio [29].

Etapas do Sistema de *Spike Sorting*

As etapas de um sistema completo de *spike sorting* tipicamente envolvem [14, 30]: a) filtragem do registro extracelular na faixa de frequência entre 300 e 5 kHz – o objetivo desta etapa é manter apenas as componentes de frequência produzidas pelos potenciais de ação *multiunit* e *single unit*; b) detecção do sinal de spike (*threshold detection*) – o objetivo desta etapa é detectar apenas as atividades do tipo *single unit*; c) segmentação – o objetivo desta etapa é extrair o sinal de spike do registro extracelular; d) alinhamento dos picos dos *spikes* extraídos; e) extração de características (redução do número de dimensões/aprendizado de características) – o objetivo desta etapa é manter apenas a informação que favoreça a etapa de clusterização; e f) clusterização (processo de separação de um conjunto de dados em grupos, com base em alguma medida de similaridade). A Figura 2.2 apresenta as etapas de um sistema de *spike sorting* genérico.

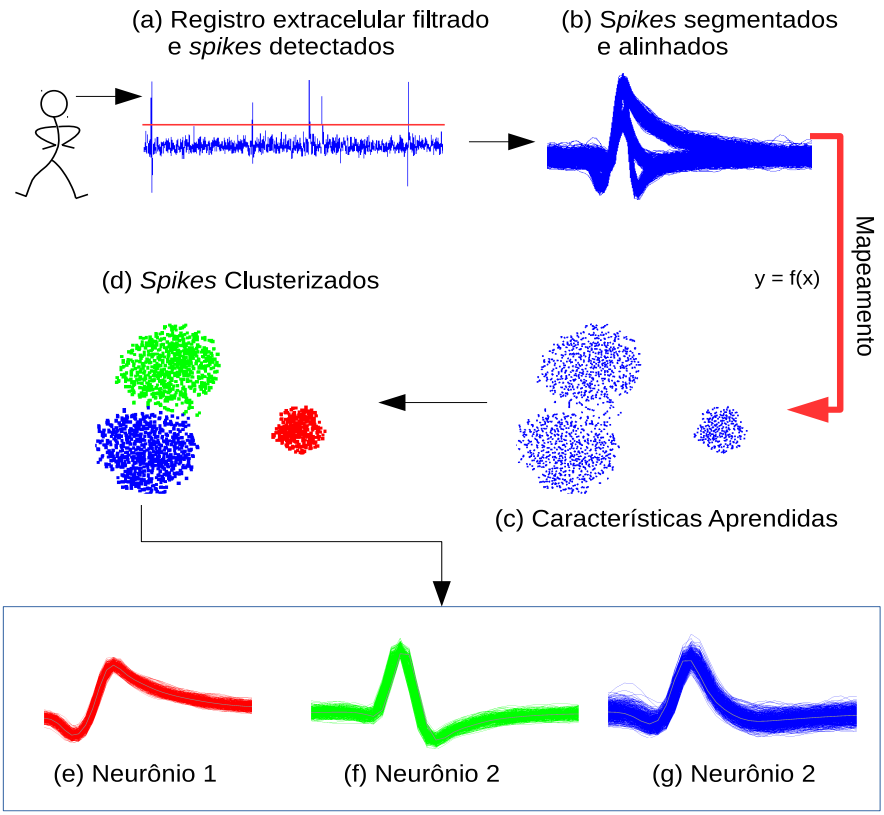


Figura 2.2: Etapas de um sistema de *spike sorting* genérico: (a) registro extracelular filtrado e *spikes* detectados; (b) *spikes* segmentados e alinhados pelo pico; (c) características extraídas/número de dimensões reduzido, onde cada ponto no espaço 2-D corresponde a um *spike* detectado; (d) clusteres encontrados, e (e)-(g) sinais de *spikes* separados em grupos com base em um algoritmo de clusterização. Para colorir estes grupos nesta figura, o conhecimento sobre o grupo correto foi utilizado.

2.2.1 Problemas Típicos em *Spike Sorting*

Além da atenuação de intensidade sofrida pelo sinal de *spike* ao percorrer o meio extracelular até alcançar o eletrodo de aquisição, existem outros problemas inerentes ao processo de *spike sorting* que podem influenciar no resultado da etapa de clusterização [14]:

- *Burst-firing neurons*: tipicamente se considera que o formato dos *spikes* gerados por um determinado tipo de neurônio se mantém fixo ao longo do período de aquisição do registro extracelular, ou seja, que os *spikes* são estacionários. Contudo é sabido que existem neurônios que geram rápidas sequências de *spikes* (*bursting neurons*), com uma pequena diferença de amplitude entre eles. Em muitos casos, a diferença na forma desses *spikes* pode levá-los a serem atribuídos a clusteres distintos.

- *Electrode drift*: a movimentação do eletrodo (*electrode drift*) durante o processo de aquisição do registro extracelular é outro fator que pode provocar modificação no formato do *spike* registrado, e consequentemente resultar que *spikes* produzidos por um mesmo tipo de neurônio sejam atribuídos a clusteres distintos.

- *Ruído de fundo não estacionário*: muitos algoritmos de detecção de *spikes* utilizam um determinado período do registro extracelular para definir o *threshold* aplicado para a detecção dos sinais de *spike*. Se o ruído de fundo mudar ao longo do registro extracelular sob análise, o *threshold* calculado com base em um período fixo pode se tornar inadequado para a correta detecção dos *spikes*. Como consequência, muitos *spikes* podem não ser mais detectados ou, até mesmo, trechos contendo apenas ruído podem ser considerados *spikes*. Uma possível forma de contornar esse problema é definir *thresholds* dinâmicos ao longo do registro extracelular.

- *Sobreposição de spikes*: ocorre quando *spikes* gerados por diferentes neurônios alcançam o eletrodo de aquisição no mesmo instante de tempo ou em instantes de tempo muito próximos. As formas resultantes da combinação linear dos *spikes* podem produzir um novo *cluster* ao final da etapa de clusterização. Como consequência, o "*cluster* extra"pode levar a uma interpretação errada sobre a existência de um nova classe de neurônio. O uso de múltiplos eletrodos (pequenas matrizes de eletrodos) durante o processo de aquisição dos registros extracelulares pode ser adotado como uma alternativa para facilitar o processamento das ocorrências de sobreposição de *spikes*. A informação espacial dos múltiplos registros, realizados simultaneamente em uma pequena

região do cérebro, pode ser utilizada para identificar com maior facilidade os casos de sobreposição de *spikes*, visto que os *spikes* que se sobrepõem em um determinado eletrodo podem não se sobrepor em outro eletrodo.

2.3 Algoritmos de *Spike Sorting*

Nos últimos anos, vários algoritmos tais como *principal component analysis* (PCA) [31], *linear discriminant analysis* (LDA) [32], *wavelet transform* [33], *template matching* [34], *bayesian classification* [35], *fourier transform* [36], *locality preserving projections* (LPP) [37], *artificial neural networks* [38] e *convolutive independent component analysis* [39] foram utilizados em métodos de *spike sorting*, porém a maioria dos trabalhos avaliaram seus métodos em registros neurais contendo limitações como: a) nível de ruído de fundo considerado desprezível, b) registros extracelulares contendo atividades neurais de apenas três neurônios.

Com o avanço na tecnologia dos equipamentos de aquisição de sinais é esperado que os registros extracelulares apresentem um número cada vez maior de neurônios, tanto pela redução no ruído quanto pelo aumento do número de canais gravados simultaneamente. Esse aumento no volume de dados implica a necessidade de que os novos e os já tradicionais algoritmos de *spike sorting* sejam avaliados em condições mais realistas o possível, como por exemplo, em registros extracelulares contendo um grande número de neurônios e/ou nível de ruído de fundo não desprezível.

Apesar do *spike sorting* ser uma técnica amplamente utilizada em muitos trabalhos científicos e aplicações clínicas, ainda existem poucos estudos que exploram os limites dos algoritmos de *spike sorting*. Com o crescente aumento da quantidade e da complexidade dos dados registrados pelos novos sistemas de aquisição, é cada vez mais importante o desenvolvimento de novos algoritmos de *spike sorting* que exijam o mínimo, ou se possível, que operem de forma independente de auxílio do pesquisador. Devido ao grande volume de dados produzidos pelos novos sistemas de aquisição, a tendência é que se torne inviável a realização de *spike sorting* por meio de algoritmos que dependam da supervisão do pesquisador. A supervisão é uma tarefa que demanda tempo, necessita do auxílio de especialistas na área de pesquisa e no algoritmo de *spike sorting* utilizado, e também está sujeita a erros que podem ser produzidos pelo pesquisador durante as etapas de pré-processamento dos dados e de configuração dos parâmetros do algoritmo de *spike sorting*.

2.3.1 Limitações dos Algoritmos de *Spike Sorting*

Estudos baseados nas propriedades biofísicas e anatômicas celulares destacam que o número de neurônios presentes nos registros extracelulares é da ordem das centenas

[40]; no entanto, o número de neurônios identificados pelos algoritmos de *spike sorting*, em geral, está abaixo de dez. Diversas razões foram sugeridas na tentativa de explicar tal discrepância [41], como por exemplo: danos ao tecido neural causados pela inserção dos eletrodos, isolamento elétrico devido ao substrato do eletrodo e baixa taxa de disparos de um grande número de neurônios (neurônios silenciosos).

Experimentos realizados em [29] avaliaram algoritmos de *spike sorting* em condição de aumento do número de neurônios. Os resultados obtidos mostram que, conforme o número de neurônios aumenta, o desempenho dos algoritmos avaliados fica limitado, alcançando um máximo de dez neurônios detectados para registros neurais contendo 20 neurônios. Esses resultados sugerem que possíveis limitações dos atuais algoritmos de *spike sorting* contribuam para a diferença entre o número de neurônios estimados e o número correto de neurônios.

Capítulo 3

Aprendizado de Características

O aprendizado de máquina é um campo da ciência da computação que tenta transferir ao computador a capacidade de aprender sem ser expressamente programado. Dependendo do tipo de tarefa a ser realizada pelo sistema/modelo, os algoritmos de aprendizado de máquina, utilizados para o treinamento do sistema, podem ser classificados como supervisionados ou não supervisionados.

De modo geral, as mais diversas tarefas realizadas pelos sistemas baseados em aprendizado de máquina podem ser pensadas como funções de mapeamento (lineares ou não) entre um domínio de entrada e um domínio de saída. Por exemplo, a partir da imagem de uma pessoa qualquer, pode ser desejado que o sistema gere como saída os valores "zero"ou "um", onde o valor "zero"corresponda a indivíduo do "sexo masculino"e o valor "um"corresponda a indivíduo do "sexo feminino". Nesse exemplo, o domínio de entrada é o domínio da imagem (domínio do vetor formado pelos pixels da imagem) e o domínio de saída é o domínio binário (zero ou um).

Em muitos casos, o aprendizado da função de mapeamento pode representar um grande desafio. Diversos fatores exercem influência no processo de aprendizado, como por exemplo: a tarefa a ser realizada (predição, classificação, clusterização), a disponibilidade de dados (padrões) em quantidade suficiente para o treinamento do sistema, a forma como os dados estão representados (*data representation*, *feature representation*, *feature extraction*, *manifold learning*)[42] e o conhecimento prévio ou não do alvo (saída a ser gerada pelo sistema) associado a cada padrão de entrada.

Muitos dos atuais esforços nas pesquisas em aprendizado de máquina estão no desenvolvimento de algoritmos aplicados para o "pré-processamento"ou para a "transformação de dados". O objetivo é que esses algoritmos, ao realizarem as etapas de pré-processamento ou de transformação dos dados, sejam capazes de preservar somente a informação relevante (aquela que produza a manutenção do desempenho dos sistemas nas mais diversas condições em que o padrão de entrada se apresentar).

Uma abordagem ainda muito utilizada para a extração das informações (características) relevantes é o uso de extratores de características manualmente desen-

volvidos "*feature engineering*"[43, 44]. No entanto, esta abordagem apresenta as seguintes limitações: a) as características extraídas precisam ser definidas por um especialista tanto na tarefa a ser realizada pelo sistema (predição, classificação, clusterização), quanto no tipo de dado utilizado (imagem, áudio, sinal biológico), e b) para tarefas complexas como visão computacional, processamento de linguagem natural e processamento de sinais biológicos é extremamente difícil usar conhecimento prévio para definir as características a serem extraídas.

Devido às limitações implícitas na abordagem via *feature engineering*, algoritmos mais complexos, não dependentes de conhecimento prévio de especialistas, e que podem ser aplicados para os mais variados tipos de dados, têm sido propostos ao longo dos últimos anos [45–48]. Nas próximas seções são apresentados os principais algoritmos para o aprendizado de características utilizados nesta tese.

3.1 Algoritmos Supervisionados para Aprendizado de Características

Dados o conjunto $\mathbf{P}$ contendo m observações $\mathbf{x}_i \in \mathbb{R}^D$, $i = 1, 2, \ldots, m$, e o conjunto $\mathbf{Q}$ contendo m alvos $\mathbf{y}_i \in \mathbb{R}^L$. Os algoritmos supervisionados, utilizados para o aprendizado de características, tentam aprender uma função de mapeamento $\hat{\mathbf{y}} = f(\mathbf{x}, \theta)$ que seja capaz de associar cada observação $\mathbf{x}_i \in \mathbf{P}$ ao seu respectivo alvo $\mathbf{y}_i \in \mathbf{Q}$. O conjunto $\theta = \{\theta_0, \theta_1, \ldots, \theta_n\}$ corresponde ao conjunto de parâmetros treináveis da função de mapeamento, que podem ser ajustados através de algoritmos de aprendizado.

3.1.1 Complexidade da Função de Mapeamento

Em aprendizado de máquina é desejado que a função de mapeamento, aprendida de modo supervisionado, seja capaz de fornecer o alvo correto, mesmo para as observações não utilizadas durante o processo de ajuste de parâmetros. A esta habilidade é dado o nome de "generalização". Para selecionar uma família de funções que representem possíveis soluções (boa capacidade de generalização) para a tarefa a ser executada, é necessário levar em consideração o "*tradeoff*"existente entre a complexidade da função de mapeamento e a real complexidade da tarefa a ser realizada.

Se a função de mapeamento aprendida for muito simples, assim como

$$\hat{\mathbf{y}} = \theta_0 + \theta_1 \mathbf{x}, \tag{3.1}$$

então a função só será capaz de realizar mapeamentos simples, como por exemplo,

funções lineares. No caso da função de mapeamento apresentar maior complexidade, como por exemplo

$$\hat{\mathbf{y}} = \theta_0 + \sum_{i=1}^{T} \theta_i \mathbf{x}^i, \tag{3.2}$$

então a função será capaz de realizar mapeamentos mais complexos, como por exemplo, funções quadráticas.

Se a complexidade da função de mapeamento for muito superior à complexidade da tarefa a ser executada, então a função de mapeamento tenderá a apresentar baixa capacidade de generalização. A Figura 3.1 apresenta a saída de três sistemas de predição baseados em famílias de funções de mapeamento com diferentes complexidades. A Figura 3.1(a) corresponde ao caso em que a complexidade da função é inferior à complexidade da tarefa. A Figura 3.1(b) corresponde ao caso em que a complexidade da função é adequada à complexidade da tarefa. A Figura 3.1(c) corresponde ao caso em que a complexidade da função é superior à complexidade da tarefa.

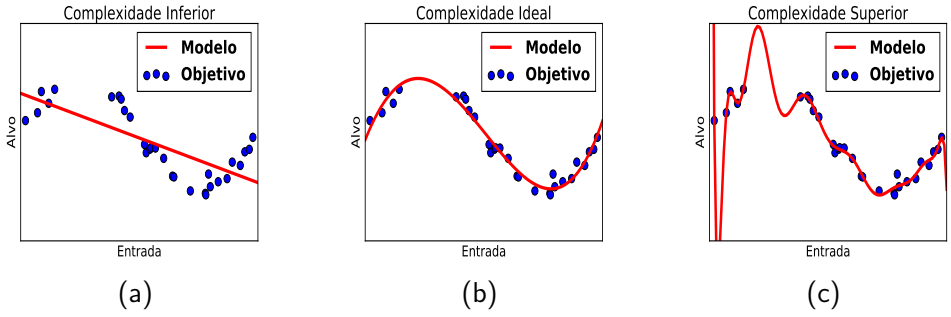


Figura 3.1: Influência da complexidade da função de mapeamento no desempenho de um sistema de predição. A curva "Objetivo"mostra a real complexidade do mapeamento a ser aprendido. A curva "Modelo"corresponde à função de mapeamento aprendida pelo sistema de predição. (a) função de mapeamento com complexidade inferior à real complexidade do problema. (b) função de mapeamento com complexidade equivalente à real complexidade do problema. (c) função de mapeamento com complexidade superior à real complexidade do problema.

3.1.2 Análise de Discriminante Linear (LDA)

Análise de discriminante linear é um algoritmo supervisionado muito utilizado em áreas como aprendizado de máquina, reconhecimento de padrões e estatística. O seu objetivo é, a partir de um conjunto de observações $\mathbf{x}_i \in \mathbb{R}^D$, $i = 1, 2, \ldots, m$, calcular uma função de mapeamento $\hat{\mathbf{y}}_i = f(\mathbf{x}_i)$ que maximize a separação entre classes (*between-class variance* – $\mathbf{\Sigma}_B$) e maximize a compactação intra-classe (*within-class variance* – $\mathbf{\Sigma}_W$) dos grupos formados no novo domínio de representação $\mathbf{y}_i \in \mathbb{R}^L$, onde $L \leqslant D$.

Considerando o conjunto de observações $\mathbf{x}_i$ organizadas em uma matriz $\mathbf{X}$, onde as linhas representam as observações e as colunas representam o número de dimensões das observações $\mathbf{x}_i$ (todas as observações contendo igual número de dimensões), então a matriz $\mathbf{X}$ formada possuirá m linhas e D colunas. Sem perda de generalidade, considerando que a matriz $\mathbf{X}$ seja dividida em k partições, ou seja, $\mathbf{X} = [\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_k]$ e que todas as partições possuam o mesmo número de observações N_i (número de observações na partição $\mathbf{X}_i$), então as métricas $\mathbf{\Sigma}_B$ e $\mathbf{\Sigma}_W$ são obtidas, respectivamente, através das Equações (3.3) e (3.4).

$$\begin{aligned} \mu &= \frac{1}{m} \sum_{\mathbf{x}_i \in \mathbf{X}}^{m} \mathbf{x}, \\ \mu_i &= \frac{1}{N_i} \sum_{\mathbf{x} \in \mathbf{X}_i}^{N_i} \mathbf{x}, \\ \mathbf{\Sigma}_B &= \sum_{i=1}^{k} N_i (\mu_i - \mu)(\mu_i - \mu)^T, \end{aligned} \tag{3.3}$$

onde m é o total de observações contidas na matriz $\mathbf{X}$, μ é o vetor correspondendo à média de todas as observações organizadas na matriz $\mathbf{X}$ (vetor médio do conjunto completo), μ_i é o vetor correspondendo à média de todas as observações organizadas na matriz $\mathbf{X}_i$ (vetor médio da partição) e N_i é o número de observações contidas na partição $\mathbf{X}_i$.

$$\begin{aligned} \mathbf{\Sigma}_i &= \sum_{\mathbf{x} \in \mathbf{X}_i}^{N_i} (\mathbf{x} - \mu_i)(\mathbf{x} - \mu_i)^T, \\ \mathbf{\Sigma}_W &= \sum_{i=1}^{k} \Sigma_i, \end{aligned} \tag{3.4}$$

onde Σ_i é a matriz de covariância obtida a partir da partição $\mathbf{X}_i$.

Se Σ_W for não singular, ou seja, possuir inversa, então a matriz de transformação, $\mathbf{W}$, utilizada na construção da função de mapeamento

$$\hat{\mathbf{y}} = f(\mathbf{x}) = \mathbf{W}^T \mathbf{x}, \tag{3.5}$$

deverá satisfazer à Equação (3.6),

$$\mathbf{W} = \arg\max_{\mathbf{W}} \frac{\mathbf{W}^T \Sigma_B \mathbf{W}}{\mathbf{W}^T \Sigma_W \mathbf{W}}. \tag{3.6}$$

A matriz de transformação $\mathbf{W}$ é calculada através da solução do problema de autovetores generalizados, Equação (3.7),

$$\Sigma_W \mathbf{v}_n = \lambda_n \Sigma_B \mathbf{v}_n, \tag{3.7}$$

onde $n = 1, 2, \ldots, D$, o autovetor $\mathbf{v}_n$ é uma possível solução para a Equação (3.7), e λ_n é o autovalor associado ao autovetor $\mathbf{v}_n$.

Construção da Matriz de Transformação W

Após o cálculo de todos os autovetores e autovalores associados que satisfazem a Equação (3.7), os autovetores são organizados, na matriz $\mathbf{W}$, segundo ordem decrescente de autovalores associados, ou seja, a primeira coluna da matriz $\mathbf{W}$ corresponderá ao autovetor associado ao maior autovalor calculado anteriormente, a segunda coluna da matriz $\mathbf{W}$ corresponderá ao autovetor associado ao segundo maior autovalor calculado anteriormente, e assim por diante. O algoritmo LDA é considerado supervisionado pois para o cálculo da matriz de transformação $\mathbf{W}$ é necessário o conhecimento prévio da partição à qual cada observação $\mathbf{x}_i$ pertence.

3.1.3 Perceptron Multicamadas

O perceptron multicamadas (*multilayer perceptron* – MLP) [49], também conhecido como *feedforward neural network*, é um sistema de computação inspirado em redes neurais biológicas, inicialmente proposto para trabalhar como um aproximador de funções contínuas. Dados uma função $\mathbf{y} = f^*(\mathbf{x})$, onde $f^* : \mathbb{R}^D \to \mathbb{R}^L$, e um conjunto de observações $\mathbf{x}_i \in \mathbb{R}^D$, $i = 1, 2, \ldots, m$ para o qual a função $f^*(\cdot)$ esteja definida, o objetivo do MLP é aprender a função de mapeamento $\hat{\mathbf{y}} = f(\mathbf{x}, \theta)$ que mais se aproxime da função $f^*(\cdot)$.

Tradicionalmente, o treinamento do MLP é supervisionado, pois para o ajuste dos parâmetros θ da função de mapeamento, $\hat{\mathbf{y}}_i = f(\mathbf{x}_i, \theta)$, realizada pelo MLP, é necessária a informação da saída $\mathbf{y}_i = f^*(\mathbf{x}_i)$ associada à observação $\mathbf{x}_i$. Usualmente $\mathbf{x}_i$ é chamado de padrão de entrada ou simplesmente de entrada e $\mathbf{y}_i$ é chamado de alvo. Os valores ótimos do conjunto de parâmetros, θ, são aqueles que minimizam o valor médio de uma função de perda ("*loss function*") $\mathcal{L}(\theta; \hat{\mathbf{y}}, f^*(\mathbf{x}))$ definida para o conjunto de pares entrada-alvo $(\mathbf{x}_i, f^*(\mathbf{x}_i))$:

$$\theta = \arg\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\theta; \hat{\mathbf{y}}_i, f^*(\mathbf{x}_i)). \tag{3.8}$$

Processamento do Padrão de Entrada

Com o objetivo de simplificar e dar maior clareza às explicações, o MLP contendo uma única camada escondida será detalhado nesta seção, contudo a teoria pode ser estendida para MLPs contendo múltiplas camadas escondidas. O MLP pode ser treinado para aproximar qualquer tipo de função contínua. Existem diversos fatores que influenciam na sua capacidade de aprendizado, tais como: número de

camadas escondidas, número de neurônios artificiais por camada, funções de ativação utilizadas e número de observações disponíveis para o treinamento do modelo (ajuste dos parâmetros do MLP).

A função de mapeamento realizada pelo MLP pode ser representada através da Equação (3.9):

$$\hat{\mathbf{y}} = f(\mathbf{x}_i, \theta) = g(\mathbf{W}^{(2)} h(\mathbf{W}^{(1)} \mathbf{x}_i + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}) \tag{3.9}$$

onde $\theta = \{\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \mathbf{W}^{(2)}, \mathbf{b}^{(2)}\}$ é o conjunto de parâmetros ajustáveis do mapeamento. $\mathbf{W}^{(1)}$ é a matriz de pesos sinápticos que conectam a camada de entrada com a camada escondida. O peso sináptico pode ser interpretado como a força da ligação entre duas unidades de processamento (neurônios artificiais) ou entre a camada de entrada e o neurônio artificial, conforme é mostrado na Figura 3.2. O vetor $\mathbf{b}^{(1)}$ representa o *bias* dos neurônios da primeira camada. O *bias* pode ser interpretado como a quantidade mínima de ativação do neurônio, na ausência de padrão de entrada. Os pesos sinápticos e *bias* da segunda camada (saída do MLP) são dados por $\mathbf{W}^{(2)}$ e $\mathbf{b}^{(2)}$, respectivamente. O número de componentes do vetor $h(\cdot)$ é igual ao número de neurônios artificiais na camada escondida. O número de componentes do vetor $g(\cdot)$ é igual ao número de neurônios artificiais na camada de saída. Cada uma das componentes dos vetores $h(\cdot)$ ou $g(\cdot)$ corresponde à saída de uma função linear ou não linear, chamada de função de ativação, conforme é mostrada na Tabela 3.1. O processamento de um determinado padrão de entrada ocorre quando, dado um padrão $\mathbf{x}_i$, este é transmitido ao longo das camadas da rede neural ("*forward pass*"), passando por etapas de transformações, lineares ou não lineares, realizadas pelas funções de ativação dos neurônios artificiais até produzir a saída $\hat{\mathbf{y}} = g(\cdot)$ (saída do MLP).

Função-Custo do MLP

A escolha da função-custo apropriada é outro fator determinante no desempenho do sistema final. Quando o MLP é treinado para realizar predições (aproximar uma função de predição), a função-custo mais indicada é a função erro médio quadrático com regularização, mostrada na Equação (3.10).

$$J(\theta; \mathbf{x}_i, \mathbf{y}_i) = \frac{1}{2m} \sum_{i=1}^{m} \parallel \mathbf{y}_i - \hat{\mathbf{y}}_i \parallel^2 + \frac{\lambda}{2} \sum_{k=1}^{K} \theta_k^2, \tag{3.10}$$

onde K é o número de parâmetros ajustáveis da função de mapeamento implementada pelo MLP e λ é um multiplicador de Lagrange escolhido arbitrariamente de modo a limitar a magnitude dos parâmetros θ_k. A forma lagrangeana adotada pela Equação (3.10) corresponde à minimização da função erro médio quadrático

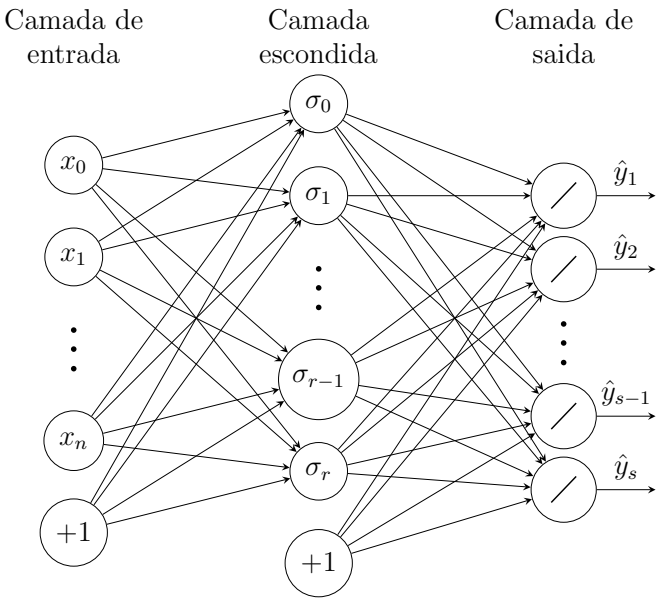


Figura 3.2: Topologia de um MLP contendo apenas uma camada escondida. Os neurônios da camada escondida são do tipo sigmoidal e os neurônios da camada de saída são do tipo linear. A camada de entrada corresponde ao padrão de entrada utilizado para o treinamento da rede. As ligações entre a camada de entrada e a camada escondida, ou entre a camada escondida e a camada de saída, correspondem aos pesos sinápticos da rede (parâmetros da função de mapeamento implementada pelo MLP).

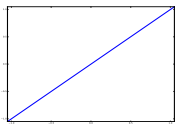
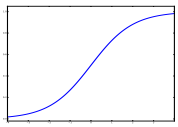
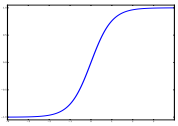
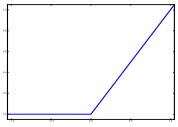
Funções de Ativação		
Nome	Função Matemática	Representação Gráfica
Linear	$f(x) = x$	
Sigmoidal	$f(x) = \frac{1}{1+e^{-x}}$	
Tangente Hiperbólica	$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	
ReLU	$f(x) = max(0, x)$	

Tabela 3.1: Exemplos de funções de ativação realizadas pelos neurônios artificiais utilizados na construção de redes neurais do tipo MLP.

com regularização aplicada à norma quadrática, $\mathcal{L}_2$, do vetor de parâmetros a ser encontrado por otimização.

Quando o MLP é treinado para realizar tarefa de classificação (aproximar uma função de classificação), a função custo mais indicada é a função entropia cruzada (*cross-entropy*), mostrada na Equação (3.11).

$$J(\theta; \mathbf{x}_i, \mathbf{y}_i) = -\sum_{i=1}^{m}\sum_{n=1}^{L} y_i^{(n)} \log \hat{y}_i^{(n)} + \frac{\lambda}{2}\sum_{k=1}^{K}\theta_k^2, \tag{3.11}$$

onde $y_i^{(n)}$ é a n-ésima componente do vetor $\mathbf{y}_i$ e $\hat{y}_i^{(n)}$ é a n-ésima componente do vetor $\hat{\mathbf{y}}_i$.

Ajuste dos Parâmetros do MLP via Gradiente Descendente

Normalmente o ajuste dos parâmetros do MLP é realizado via métodos de gradiente [50], sendo o método "gradiente descendente"o mais utilizado, conforme mostrado na Equação (3.12):

$$\theta_k^{(t)} = \theta_k^{(t-1)} - \alpha \frac{\partial}{\partial \theta_k} J(\theta), \tag{3.12}$$

onde $\alpha \in \mathbb{R}$ corresponde à taxa de aprendizado. A variável t, que é progressivamente incrementada ao longo do treinamento, é chamada de época. $J(\theta)$ é a função custo a ser minimizada e $\frac{\partial}{\partial \theta_k} J(\theta)$ corresponde ao valor médio do gradiente da função custo em relação ao k-ésimo parâmetro do MLP:

$$\frac{\partial}{\partial \theta_k} J(\theta) = \left[\frac{1}{m}\sum_{i=1}^{m} \frac{\partial}{\partial \theta_k} J(\theta; \mathbf{x}_i, \mathbf{y}_i)\right] + \lambda\theta_k. \tag{3.13}$$

Cálculo do Gradiente da Função-Custo via *Backpropagation*

Dado um par entrada-alvo ($\mathbf{x}_i$,$f^*(\mathbf{x}_i)$), o gradiente da função custo, em relação a cada um dos parâmetros θ_k da função de mapeamento implementada pelo MLP, é calculado via algoritmo *backpropagation*, conforme descrito nas seguintes etapas:

1. Realizar o *forward pass* do padrão de entrada $\mathbf{x}_i$ e calcular as saídas de cada um dos neurônios artificiais do MLP. As saídas são dadas pelas componentes dos vetores $h(\cdot)$ e $g(\cdot)$.

2. Calcular o quanto cada neurônio influencia no erro apresentado entre o alvo correto $\mathbf{y}_i$ e o alvo estimado $\hat{\mathbf{y}}_i$. A esse valor é dado o nome de *sinal-erro*, δ.

   (a) Para a camada de saída o vetor de *sinal-erro*, δ_g, é dado pela Equação (3.14):

$$\delta_g = -(\mathbf{y}_i - \hat{\mathbf{y}}_i)g', \tag{3.14}$$

onde as componentes do vetor $\delta_g(\cdot)$ correspondem aos *sinais-erro* associados a cada um dos neurônios da camada de saída; g' é o gradiente do vetor $g(\cdot)$. A n-ésima componente do vetor gradiente g' corresponde à derivada da função de ativação implementada pelo n-ésimo neurônio da camada de saída tomada em relação à entrada da função de ativação.

(b) Para a camada escondida o vetor de *sinal-erro*, δ_h, é dado pela Equação (3.15):

$$\delta_h = ((\mathbf{W}^{(2)})^T \delta_g) h', \tag{3.15}$$

onde $(.)^T$ representa a operação de transposição matricial. As componentes do vetor $\delta_h(\cdot)$ correspondem aos *sinais-erro* associados a cada um dos neurônios da camada escondida. h' é o gradiente do vetor $h(\cdot)$. A n-ésima componente do vetor gradiente h' corresponde à derivada da função de ativação implementada pelo n-ésimo neurônio da camada escondida.

3. Calcular os gradientes.

O vetor de gradientes para os parâmetros $\mathbf{W}^{(2)}$ e $\mathbf{b}^{(2)}$ é dado por:

$$\frac{\partial}{\partial \theta} J(\theta) = \delta_g h^T. \tag{3.16}$$

O vetor de gradientes para os parâmetros $\mathbf{W}^{(1)}$ e $\mathbf{b}^{(1)}$ é dado por:

$$\frac{\partial}{\partial \theta} J(\theta) = \delta_h \mathbf{x}_i^T. \tag{3.17}$$

A extensão do processo descrito acima, para o caso de MLPs contendo um número qualquer de camadas escondidas, é obtida de maneira simples. Por exemplo, dado um MLP contendo l camadas escondidas, o gradiente da função custo em relação aos parâmetros θ da camada $l-2$ é dado por:

$$\frac{\partial}{\partial \theta} J(\theta) = (\delta_{l-1})(p_{l-2})^T. \tag{3.18}$$

onde cada componente do vetor p_{l-2} corresponde a uma saída dada por um diferente neurônio artificial da camada $l-2$ durante o *forward pass* de um determinado padrão de entrada $\mathbf{x}_i$. O vetor de *sinais-erro*, δ_{l-1}, corresponde ao vetor de *sinais-erro*, δ_g (*sinais-erro* da saída), retropropagado até a entrada da camada $l-1$.

3.2 Algoritmos Não Supervisionados para Aprendizado de Características

Dado o conjunto $\mathbf{P}$ contendo m observações $\mathbf{x}_i \in \mathbb{R}^D$, $i = 1, 2, \ldots, m$, o objetivo dos algoritmos de aprendizado não supervisionado é aprender uma função de mapeamento $\hat{\mathbf{y}} = f(\mathbf{x}, \theta)$ sem alvos associados a cada uma das observações $\mathbf{x}_i$. Esta forma de aprendizado pode ser vista como um processo de *data representation*. Usualmente a nova representação dada para o conjunto de observações $\mathbf{P}$, através do mapeamento de seus elementos via função de mapeamento $\hat{\mathbf{y}} = f(\mathbf{x}, \theta)$, é utilizada como um novo conjunto de observações para o treinamento de sistemas baseados em algoritmos supervisionados.

3.2.1 Análise de Componentes Principais (PCA)

Análise de componentes principais é um algoritmo não supervisionado utilizado para obter um mapeamento linear $\hat{\mathbf{y}} = f(\mathbf{x})$ capaz de converter um conjunto de observações $\mathbf{x}_i \in \mathbb{R}^D$, $i = 1, 2, \ldots, m$, de variáveis correlacionadas, em um novo conjunto de observações de variáveis linearmente descorrelacionadas. As variáveis correspondem às dimensões do conjunto de observações.

Calculando as Componentes Principais

Considerando o conjunto de observações $\mathbf{x}_i$ organizado em uma matriz $\mathbf{X}$, onde as linhas representam as observações e as colunas representam as dimensões das observações $\mathbf{x}_i$ (todas as observações contendo igual número de dimensões), então a matriz $\mathbf{X}$ formada possuirá m linhas e D colunas. Inicialmente cada uma das D colunas da matriz $\mathbf{X}$ passa por uma normalização que faz com que cada coluna tenha média zero. A partir da matriz $\mathbf{X}$ (já na forma normalizada) calcula-se a matriz de covariância $\mathbf{\Sigma}$. Assumindo que a matriz $\mathbf{\Sigma}$ seja não singular, os autovetores v e autovalores associados λ podem ser calculados pela Equação (3.19)

$$\mathbf{\Sigma}\mathbf{v}_n = \lambda_n \mathbf{v}_n, \tag{3.19}$$

onde $n = 1, 2, \ldots, D$.

Construindo a matriz de transformação W

Os autovetores são então organizados em uma matriz $\mathbf{W}$ segundo ordem decrescente de autovalores associados, ou seja, a primeira coluna da matriz $\mathbf{W}$ corresponderá ao autovetor associado ao maior autovalor calculado anteriormente, a segunda coluna da matriz $\mathbf{W}$ corresponderá ao autovetor associado ao segundo maior autovalor

calculado anteriormente, e assim por diante. A função de mapeamento será então dada pela Equação (3.20)

$$\hat{\mathbf{y}} = f(\mathbf{x}) = \mathbf{W}^T\mathbf{x}. \tag{3.20}$$

Cada um dos autovetores da matriz $\mathbf{W}$ é interpretado como um eixo de projeção (mapeamento). O autovetor associado ao maior autovalor corresponde ao eixo de maior variância das observações armazenadas na matriz $\mathbf{X}$, ou seja, os autovalores correspondem às variâncias. Os autovetores também são chamados de componentes principais da matriz $\mathbf{W}$. Os autovetores têm módulo unitário.

Redução de Dimensionalidade

Normalmente após a construção da matriz $\mathbf{W}$, apenas um subconjunto de autovetores é mantido, ou seja, algumas colunas da matriz $\mathbf{W}$ são excluídas. Como resultado, a projeção de cada observação $\mathbf{x}_i$ através da função $f(\mathbf{x}_i)$ levará a uma nova representação com menor número de dimensões em relação ao seu domínio original. Esse processo é chamado de redução de dimensionalidade.

O objetivo da redução de dimensionalidade é manter apenas a informação útil, ou seja, aquela que ajude na interpretação ou discriminação dos dados (observações) e que também permita a reconstrução dos dados (transformação do domínio com dimensão inferior de volta para o domínio original) com o mínimo erro entre a observação original e a observação reconstruída. O restante do conteúdo de informação é considerado não discriminativo.

A escolha dos autovetores a serem mantidos é realizada com base nos seus autovalores associados. O autovalor pode ser interpretado como a quantidade de energia do conjunto de observações em relação ao autovetor associado. Os autovalores são organizados em sequência decrescente de valor e então são somados os L primeiros autovalores; se a desigualdade mostrada na Equação (3.21) for satisfeita, o processo termina; caso contrário o autovalor λ_{L+1} é incluído no somatório, e assim por diante, até que a desigualdade seja satisfeita.

$$\frac{\sum_{n=1}^{L}\lambda_n}{\sum_{n=1}^{D}\lambda_n} \geqslant 0,9, \tag{3.21}$$

onde $1 \leqslant L \leqslant D$ e 0,9 é um valor tipicamente utilizado, o qual representa o limiar de energia a ser mantido após a etapa de redução de dimensionalidade.

3.2.2 Projeções de Preservação Local (LPP)

LPP é um algoritmo não supervisionado, muito utilizado como uma alternativa ao PCA quando aplicado em problemas de análise de dados e classificação de padrões.

Inicialmente o LPP constrói um grafo que descreve as relações de vizinhança entre os pontos de um determinado conjunto de observações $\mathbf{x}_i$. Em seguida, através da solução de um problema de autovalores e autovetores generalizados, encontra-se uma transformação linear/função de mapeamento $\hat{\mathbf{y}} = f(\mathbf{x})$ que minimiza a função custo associada ao "Laplacian Eigenmap"[51]. A função de mapeamento obtida através do algoritmo LPP é construída de modo a preservar a informação da vizinhança local. Em outras palavras, pontos próximos no domínio original são mantidos próximos no novo domínio (nova representação do conjunto de observações iniciais).

Etapas do Algoritmo LPP

Dado um conjunto de observações $\mathbf{x}_i \in \mathbb{R}^D$, $i = 1, 2, \ldots, m$, é desejado obter uma função de mapeamento $\hat{\mathbf{y}} = f(\mathbf{x})$ que mapeie cada observação $\mathbf{x}_i$ em sua nova representação $\mathbf{y}_i \in \mathbb{R}^L$, onde $L \leqslant D$.

Grafo de adjacências: considerando que $\mathbf{G}$ seja um grafo contendo m nós, sempre que dois pontos (observações) $\mathbf{x}_i$ e $\mathbf{x}_j$ estiverem "próximos", uma borda (ligação) será estabelecida entre os nós i e j. Como possíveis critérios de proximidade tem-se:

(a) ϵ-neighborhoods, $\epsilon \in \mathbb{R}$. Os nós i e j são conectados através de uma borda se $\| \mathbf{x}_i - \mathbf{x}_j \|^2 \leq \epsilon$ onde $\epsilon \in \mathbb{R}$;

(b) k vizinhos mais próximos, $k \in \mathbb{N}$. Os nós i e j são conectados através de uma borda se i está entre os k vizinhos mais próximos de j ou se j está entre os k vizinhos mais próximos de i.

Pesos: os pesos do grafo $\mathbf{G}$ são armazenados em uma matriz m-dimensional, $\mathbf{W}$, esparsa e simétrica. A posição W_{ij} armazena o valor do peso da ligação entre os nós i e j. Os casos de ausência de conexão entre os nós i e j são representados por $W_{ij} = 0$. Existem duas possibilidades para determinar os pesos das bordas do grafo:

(a) *Heat kernel*, com $t \in \mathbb{R}$. Se os nós i e j forem considerados conectados, então

$$W_{ij} = \exp^{-\frac{\|x_i - x_j\|^2}{t}}; \tag{3.22}$$

(b) *Simple-minded*, com $t = 0$. $W_{ij} = 1$ se e somente se i e j forem considerados conectados.

Eigenmaps: a Equação (3.23) corresponde à forma generalizada do problema de autovetores

$$\mathbf{X}L\mathbf{X}^T v = \lambda \mathbf{X}D\mathbf{X}^T, \tag{3.23}$$

onde D é a matriz diagonal. Cada elemento da matriz D é calculado pela equação $D_{ij} = \sum_i W_{ij}$. $L = D = W$ é a matriz laplaciana. A i-ésima coluna da matriz $\mathbf{X}$ é dada por $\mathbf{x}_i$.

Seja o conjunto de autovetores $\mathbf{A} = [\mathbf{v}_0, \ldots, \mathbf{v}_{l-1}]$, ordenados de acordo com seus autovalores associados, $\lambda_0, <, \ldots, <, \lambda_{l-1}$, uma solução da Equação (3.23). Então, a função de mapeamento aprendida através do algoritmo LPP é dada pela equação

$$\hat{\mathbf{y}} = f(\mathbf{x}) = \mathbf{A}^T\mathbf{x}, \tag{3.24}$$

onde $\mathbf{A}$ é a matriz de transformação, contendo $n = D$ linhas e $l = L$ colunas; e $\mathbf{y}_i \in \mathbb{R}^L$ corresponde à nova representação associada à observação $\mathbf{x}_i \in \mathbb{R}^D$ obtida através do mapeamento

$$\mathbf{y}_i = \mathbf{A}^T\mathbf{x}_i. \tag{3.25}$$

Capítulo 4

Aprendizado de Características via MLP-KLD

Neste capítulo é apresentado um novo algoritmo de aprendizado de características (MLP-KLD), baseado em perceptrons multicamadas (MLPs), treinados de forma não supervisionada, usando a divergência de Kullback-Leibler (KLD) como função-custo, para resolver o problema de *spike sorting*. Conforme descrito no Capítulo 2 as etapas de um sistema completo de *spike sorting* tipicamente envolvem: detecção, segmentação, normalização, redução de dimensionalidade (mapeamento) e clusterização. O algoritmo proposto será utilizado para o treinamento dos MLPs aplicados na etapa de mapeamento. O MLP treinado via MLP-KLD terá como objetivo, para cada padrão de entrada, produzir um vetor de características que corresponda à nova representação do padrão em um espaço de mapeamento com dimensionalidade diferente da dimensionalidade original do padrão mapeado. Usualmente o espaço de mapeamento adotado é 2-D. A vantagem do uso de espaço de mapeamento 2-D é que ele permite a visualização de possíveis agrupamentos (clusteres) existentes no conjunto de padrões mapeados [1]. Este capítulo é organizado como segue. O algoritmo proposto é apresentado na Seção 4.1. O algoritmo de clusterização e as métricas de avaliação de desempenho do sistema completo de *spike sorting* são apresentados, respectivamente, nas Seções 4.2 e 4.3.

4.1 Treino de MLP com Função-Custo KLD

Seja $\mathbf{x} \in \mathbb{R}^D$ o vetor associado a um exemplar de *spike* no domínio original. Neste trabalho, o domínio original é o domínio do tempo, no qual o *spike* é segmentado de um registro com duração maior do que a do *spike*. Seja $\hat{\mathbf{y}} \in \mathbb{R}^L$ o vetor resultante

[1] As bases de dados utilizadas nesta tese são apresentadas no Apêndice A, e a extração de característica é discutida no Apêndice B.

do mapeamento de $\mathbf{x}$ para um domínio com número de dimensões, L. Podemos representar este mapeamento através da Equação (4.1):

$$\hat{\mathbf{y}} = f(\mathbf{x}, \theta) = g(\mathbf{W}^{(2)} h(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}), \tag{4.1}$$

onde $\theta = \left\{\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \mathbf{W}^{(2)}, \mathbf{b}^{(2)}\right\}$ é o conjunto de parâmetros ajustáveis do mapeamento. O mapeamento tem a forma de um MLP com uma camada escondida, chamada de camada 1, e uma camada de saída, com dois neurônios artificiais, chamada de camada 2. Os pesos sinápticos e os *biases* da camada 1 são $\mathbf{W}^{(1)}$ e $\mathbf{b}^{(1)}$. Os pesos sinápticos e os *biases* da camada 2 são $\mathbf{W}^{(2)}$ e $\mathbf{b}^{(2)}$. O número de componentes do vetor $h(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$ é igual ao número, S, de neurônios artificiais na camada escondida. As funções de ativação dos neurônios artificiais da camada escondida são funções sigmoidais de uma entrada escalar u, com expressão $1/(1 + e^{-u})$. O vetor $\hat{\mathbf{y}}$ é igual à saída g do MLP. As funções de ativação dos neurônios artificiais da camada de saída são lineares.

Treinamento

Para o ajuste dos parâmetros θ, os métodos convencionais minimizam funções-custo tais como erro médio quadrático ou entropia cruzada, no contexto de problemas de regressão ou classificação. Em problemas de regressão ou classificação, o treinamento usa os alvos disponíveis na base de dados, ou seja, o aprendizado é supervisionado. No presente trabalho, o ajuste dos parâmetros é feito de forma a minimizar a função-custo dada pela Equação (4.2):

$$J(\theta) = \sum_{i=1}^{T} \sum_{j \neq i} p_{ij} \log \frac{p_{ij}}{q_{ij}} + \frac{\lambda}{2} \sum_{k=1}^{K} \theta_k^2. \tag{4.2}$$

O primeiro termo da função-custo corresponde à KLD entre a p.m.f. (*probability mass function*) $\mathbf{p}$, associável às distâncias entre pontos $\mathbf{x}_i$ e $\mathbf{x}_j$ no espaço original, e a p.m.f. $\mathbf{q}$, associável às distâncias entre os pontos $\hat{\mathbf{y}}_i$ e $\hat{\mathbf{y}}_j$ mapeados para o espaço de dimensão L. A p.m.f. $\mathbf{p}$ pode ser representada pelos números p_{ij} definidos conforme as Equações (4.3) e (4.4). No mapeamento das distâncias Euclideanas entre $\mathbf{x}_i$ e $\mathbf{x}_j$ para probabilidades condicionais $p_{j|i}$ são usadas funções Kernel Gaussianas centralizadas em $\mathbf{x}_i$ e com desvios-padrão σ_i escolhidos arbitrariamente pelo usuário, conforme [52]. Os números p_{ij} podem ser pensados como elementos de uma matriz quadrada com tamanho $T \times T$, onde T é o número de pontos disponíveis na base de dados utilizada para o ajuste dos parâmetros θ:

$$p_{j|i} = \frac{\exp\left(-||\mathbf{x}_i - \mathbf{x}_j||^2/(2\sigma_i^2)\right)}{\sum_{l\neq i}\exp\left(-||\mathbf{x}_i - \mathbf{x}_l||^2/(2\sigma_i^2)\right)}, p_{i|i} = 0, \text{e} \tag{4.3}$$

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2T} \tag{4.4}$$

A p.m.f. $\mathbf{q}$ é representada pelos números q_{ij} definidos conforme a Equação (4.5). Neste caso, as distâncias Euclideanas são mapeadas em probabilidades através de uma função Kernel com forma 1/(1+x) centralizadas em $\hat{\mathbf{y}}_i$.

$$q_{ij} = \frac{(1 + ||\hat{\mathbf{y}}_i - \hat{\mathbf{y}}_j||^2)^{-1}}{\sum_{l\neq i}(1 + ||\hat{\mathbf{y}}_i - \hat{\mathbf{y}}_l||^2)^{-1}} \tag{4.5}$$

Como estas distâncias que aparecem nas Equações (4.3) e (4.5) não dependem dos alvos disponíveis na base de dados, o treinamento é não-supervisionado. Esta ideia é inspirada no algoritmo *t-distributed stochastic neighbor embedding* (t-SNE) [52], que também tem seus parâmetros ajustados através da minimização de KLD entre as p.m.f.s mencionadas. O k-ésimo elemento do conjunto de parâmetros θ é representado por θ_k. No segundo termo da Equação (4.2), λ é um multiplicador de Lagrange escolhido arbitrariamente de modo a limitar as magnitudes dos parâmetros θ. A forma Lagrangeana adotada pela Equação (4.2) corresponde à minimização de KLD com regularização aplicada à norma quadrática ($\mathcal{L}_2$) do vetor de parâmetros a ser encontrado por otimização. O número de parâmetros ajustáveis é K. Durante o treinamento do MLP, além da regularização quadrática também foi aplicada uma variação do método de *dropout* [53]. Nesta variação é selecionado, para cada época e a partir de variáveis aleatórias com densidade uniforme, um subconjunto de parâmetros a serem mantidos fixos. Juntas, a regularização $\mathcal{L}_2$ e a variação do método de *dropout* empregada levam, conforme é esperado, à limitação na magnitude dos parâmetros θ_k.

Os parâmetros θ são inicializados aleatoriamente conforme uma distribuição gaussiana com média igual a zero e desvio-padrão igual a 0,1. Eles são ajustados iterativamente através de um algoritmo de gradiente descendente com momento, de acordo com a Equação (4.6):

$$\begin{aligned}
\frac{\partial J(\theta)}{\partial \theta_k} &= \left[\frac{1}{T}\sum_{i=1}^{T}\frac{\partial J(\theta, x_i, \hat{y}_i)}{\partial \theta_k}\right] + \lambda\theta_k, \\
v_k^{(t)} &= \beta v_k^{(t-1)} + (1-\beta)\frac{\partial J(\theta)}{\partial \theta_k}, \\
\Delta\theta_k^{(t)} &= -\alpha v_k^{(t)}, \\
\theta_k^{(t)} &= \theta_k^{(t-1)} + \Delta\theta_k^{(t)},
\end{aligned} \tag{4.6}$$

As constantes α e β são chamadas de taxa de aprendizado e fator de esquecimento. A variável t, que é progressivamente incrementada ao longo do treinamento, é chamada de época.

A limitação mostrada na Equação (4.7) é aplicada ao incremento $\Delta\theta_k^{(t)}$:

$$\Delta\theta_k^{(t)} = \begin{cases} \gamma \text{sgn}(v_k^{(t)})|\theta_k^{(t-1)}| & \text{se } |-\alpha v_k^{(t)}| > |\theta_k^{(t-1)}| \\ -\alpha v_k^{(t)} & \text{caso contrário,} \end{cases} \tag{4.7}$$

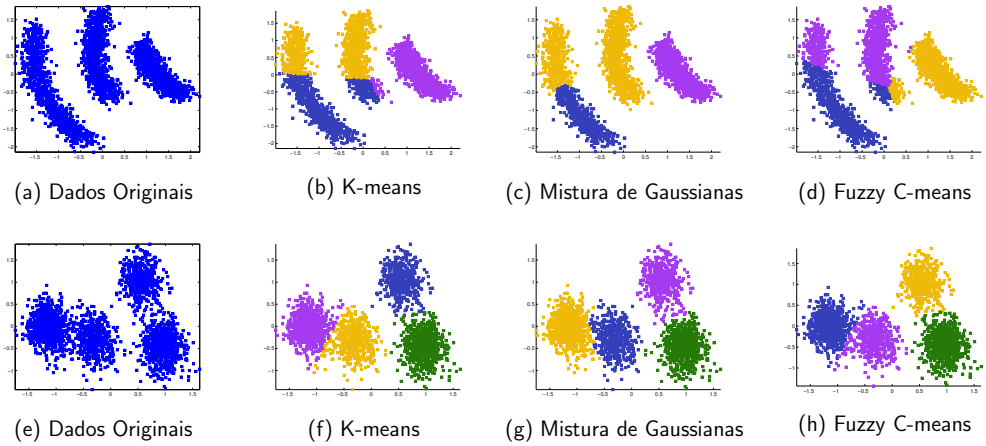
onde sgn $(\cdot)$ é a função sinal e γ é a taxa de aprendizado aplicada no caso de condição verdadeira. A limitação indicada na Equação (4.7) leva aos seguintes efeitos: a) evita $\theta_k^{(t)}$ nulo, pois o incremento somado ou subtraído é sempre menor do que $|\theta_k^{(t-1)}|$. O parâmetro nulo significa que a informação não é transmitida através das conexões entre os neurônios artificiais, e b) o tamanho (passo) máximo do incremento dependerá não somente da taxa de aprendizado α, mas também do tamanho relativo entre $|-\alpha v_k^{(t)}|$ e $|\theta_k^{(t-1)}|$, e c) restringe a região de ajuste dos parâmetros. Após a inicialização da rede, valores de θ positivos somente poderão assumir valores positivos, e valores de θ negativos somente poderão assumir valores negativos. Do ponto de vista dos parâmetros, a rede passa a ser otimizada dentro de um espaço de busca reduzido.

4.2 Clusterização

Nos experimentos apresentados no Capítulo 5, Seções 5.1 e 5.2, a inspeção visual dos dados mapeados em duas dimensões sugere que os clusteres 2-D têm formato circular ou elíptico. A Figura 4.1 indica que o formato dos clusteres influencia os resultados dos algoritmos de clusterização. No caso dos clusteres com formato circular, algoritmos de clusterização populares, como por exemplo K-means [54], mistura de Gaussianas [55] e fuzzy C-means [56] alcançam resultados semelhantes. Para reduzir ainda mais a possível influência do algoritmo de clusterização sobre a comparação entre os algoritmos de mapeamento (MLP-KLD, t-SNE, PCA, LDA e LPP), somente o algoritmo K-means básico foi utilizado em todos os experimentos, por causa de sua implementação conhecidamente mais simples.

4.3 Avaliação de Desempenho

A qualidade de um método de *spike sorting* está diretamente relacionada à sua etapa de mapeamento. É desejado que o processo de mapeamento seja capaz de destacar os clusteres naturalmente presentes no registro extracelular sob análise, portanto facilitando a etapa de clusterização. Após a etapa de mapeamento ser realizada



(a) Dados Originais (b) K-means (c) Mistura de Gaussianas (d) Fuzzy C-means

(e) Dados Originais (f) K-means (g) Mistura de Gaussianas (h) Fuzzy C-means

Figura 4.1: Influência do formato dos clusteres sobre os resultados alcançados por algoritmos de clusterização diferentes: clusteres com formatos curvos (a) e clusteres com formatos circulares (e); resultados obtidos através do algoritmo K-means básico (b) e (f); resultados obtidos através do algoritmo de misturas Gaussianas (c) e (g); e resultados obtidos através do algoritmo fuzzy C-means (d) e (h). Para clusteres circulares, os resultados obtidos pelos três algoritmos são semelhantes.

através do MLP treinado via algoritmo MLP-KLD, e dos métodos tomados como referência (t-SNE, PCA, LDA e LPP), a tarefa de *spike sorting* requer a aplicação de algum algoritmo de clusterização (K-means). Quando os algoritmos de clusterização são aplicados à nova representação do conjunto de dados, diversas situações podem ocorrer: o número correto de clusteres (número de neurônios biológicos contribuindo com potenciais de ação do tipo *single unit* para o registro extracelular) pode ou não ser encontrado, e os clusteres encontrados podem ou não apresentar mistura de dados (dados mapeados em clusteres errados).

A determinação, de forma não supervisionada, do correto número de clusteres não é uma tarefa trivial. Uma possível estratégia é executar o K-means diversas vezes (ou seja, de 1 até m), indicando a cada nova execução um número diferente de clusteres a serem formados. Os resultados das diferentes clusterizações são submetidos a algoritmos de avaliação de clusterização tais como Gap [57], Davies-Bouldin [58], Calinski-Harabasz [59] e Silhouette [60]. Os algoritmos de avaliação de clusterização fornecem um valor numérico para cada resultado diferente de clusterização. Esses valores são utilizados para auxiliar a tomada de decisão sobre qual é, provavelmente, o correto número de clusteres. A determinação automática do correto número de clusteres não está no escopo desta tese. O foco desta tese é na comparação entre algoritmos de mapeamento utilizados em sistemas de *spike sorting*.

Após os clusteres serem determinados, deve-se verificar a correspondência entre um conjunto de *spikes* que vêm de um mesmo neurônio biológico e um conjunto de pontos L-dimensionais contidos em um mesmo cluster. Para cada ponto contido no i-ésimo cluster (ou seja, com i variando de 1 até M, onde M representa o número de

neurônios biológicos presentes no registro extracelular sob análise), deve-se verificar o alvo do seu ponto de origem na base de dados. Em outras palavras, é feita a contagem do número de *spikes* provenientes de cada neurônio biológico presente na base de dados, mapeados no i-ésimo cluster. Após a descoberta do alvo de todos os pontos mapeados dentro de i-ésimo cluster é possível saber qual é o neurônio biológico com maior número de spikes mapeados para o i-ésimo cluster, qual é o segundo mais popular neurônio biológico dentro do i-ésimo cluster, e assim por diante.

Nesta tese são utilizadas algumas métricas popularmente adotadas em *spike sorting* para avaliar a qualidade do mapeamento e o desempenho na classificação. Também é proposta uma nova métrica (*Data-Counting Metric* - DCM) que visa a ponderar igualmente neurônios biológicos com diferentes taxas de disparos.[2]

4.3.1 Qualidade do Mapeamento

A qualidade dos mapeamentos obtidos através dos algoritmos MLP-KLD, t-SNE, PCA, LDA e LPP é avaliada indiretamente, estimando-a através do desempenho de um algoritmo de clusterização executado sobre os dados mapeados. O algoritmo de clusterização utilizado nesta tese é, em todos os casos e para todos os testes realizados, o K-means básico. Depois de realizada a etapa de clusterização, o significado de cada cluster (ou seja, o neurônio biológico correspondente no registro extracelular sob análise) é verificado com base nos alvos disponíveis na base de dados utilizada. A partir desta verificação, cada cluster é contabilizado como um "*hits*", "*misses*"ou "falso-positivos".

- *Hits* [23]: a contagem do número de "*hits*"(N_{hits}) é feita sobre os clusteres encontrados. Para um cluster ser considerado um "*hit*", ele precisa satisfazer duas condições, que são chamadas de *critério da maioria* e *critério do mapeamento*. Um cluster atende ao critério da maioria se 50% ou mais dos

[2]O *spike train* são sequências de impulsos ou pulsos muito pequenos no domínio do tempo. Para criar um *spike train* que represente a sinalização do i-ésimo neurônio presente no registro extracelular, os impulsos são colocados exatamente nos mesmos instantes de tempo em que os *spikes* do i-ésimo neurônio foram detectados. Os *spikes* que não forem detectados representarão perda de informação. À medida que a taxa de disparo associada à sinalização de um neurônio torna-se menor, a não detecção de *spikes* torna-se um problema crítico pois a informação transportada por cada *spike* tende a ser mais relevante. A informação neural pode ser codificada como variação da taxa de disparos, como padrões de disparo temporal, como disparos simultâneos gerados por uma população de neurônios, e assim por diante. Estas observações são importantes para a pesquisa que utiliza multiplos *spike trains* [24] como uma abordagem para estudar o processo de codificação de informações neurais. As bases de dados que são utilizadas nesta tese, não foram geradas para estudos do processo de codificação de informações neurais, mas, no entanto, introduzimos o DCM como uma contribuição original para avaliar o desempenho da clusterização em condições de desbalanceamento significativo na taxa de disparos entre os neurônios presentes em um mesmo registro extracelular.

seus pontos forem provenientes de um mesmo neurônio na base de dados sob análise. Esse neurônio é chamado de *neurônio vencedor* do respectivo cluster. Um cluster satisfaz ao critério do mapeamento se o número de pontos dele atribuídos ao neurônio vencedor corresponder a pelo menos 50% do total de pontos do neurônio vencedor na base de dados sob análise. Se o cluster possuir um neurônio vencedor e o cluster cobrir pelo menos 50% dos *spikes* desse neurônio na base de dados sob análise, então o cluster é contabilizado como "*hit*". O neurônio vencedor em questão passa a ser chamado de *neurônio associado* ao respectivo cluster. O número de "*hits*"equivale ao número de neurônios, dentre os neurônios que contribuem para a base de dados com *spikes* do tipo *single-unit*, que são associados a clusteres.

- *Falso-Positivos*: o número de clusteres "falso-positivos"é igual ao número de clusteres que atendem somente ao critério da maioria, ou seja, que possuem um neurônio vencedor e que não cobre mais do que 50% dos pontos deste neurônio na base de dados sob análise. Considerando que o termo "subcluster"seja usado para descrever clusteres cujo neurônio vencedor corresponde a um neurônio que está associado a outro cluster, então tais subclusteres farão parte da contagem de clusteres "falso-positivos". Clusteres cujo neurônio vencedor não é associado a cluster algum compõem a outra parcela desta contagem. Dependendo da distância entre os seus centróides, o cluster com um neurônio associado e o seu subcluster poderiam ser unidos em um só cluster, o que reduziria a contagem de clusteres "falso-positivos"em uma unidade, mas esta união não é considerada neste trabalho.

- *Misses*: os clusteres que não forem contabilizados como "*hits*"ou "falso-positivos"são contabilizados como "misses".

4.3.2 Desempenho na Classificação

Se classificadores de spikes forem implementados com base nos resultados (isto é, clusteres) gerados pelo K-means básico, então é possível que dois classificadores estejam associados a bons resultados em termos de "*hits*", embora um deles classifique incorretamente muitos pontos dentro de cada cluster. Por exemplo, pode ser que os dois classificadores estejam gerando N saídas diferentes no caso de uma base de dados contendo N neurônios biológicos, e um dos classificadores corretamente atribua ao neurônio 1 aproximadamente 100% dos spikes mapeados no cluster 1, enquanto o outro classificador atribua ao neurônio 1 pouco mais de 50% dos *spikes* mapeados no cluster 1 (e assim por diante). Para comparar sistemas de mapeamento, baseados nos algoritmos (MLP-KLD, t-SNE, PCA, LDA e LPP), com respeito às

taxas de acerto dos classificadores, são consideradas as seguintes definições para a "acurácia"e "número total de erros":

- *Acurácia* (ACC): seja N_i o número de pontos ("spikes") corretamente atribuídos ao neurônio associado ao i-ésimo cluster encontrado (conforme explicado no item "Hits") e seja T o número total de pontos na base de dados sob análise. A acurácia, ou taxa de acertos, é a soma dos valores N_i, $i = 1, 2, \ldots N_{\text{hits}}$ dividida por T:

$$\text{ACC} = \frac{1}{T} \sum_{i=1}^{N_{hits}} N_i \tag{4.8}$$

- *Número Total de Erros*: o número total de erros N_{E} é igual ao número total de pontos na base de dados sob análise menos o somatório dos valores N_i, ou seja, $N_{\text{E}} = T(1 - \text{ACC})$.

4.3.3 Medidas Adicionais

Além dos métodos de avaliação descritos nas Seções 4.3.1 e 4.3.2, podemos também avaliar os algoritmos (MLP-KLD, t-SNE, PCA, LDA e LPP) através de medidas de desempenho que são mais especificamente voltadas para a avaliação de algoritmos de clusterização. A seguir descrevemos a métrica AMI e propomos uma nova métrica (DCM):

- *Adjusted Mutual Information* (AMI) [61]: a AMI compara os alvos (rótulos), disponíveis para os pontos na base de dados sob análise, com as saídas obtidas após a aplicação do algoritmo de clusterização (K-means básico, no trabalho atual), conforme a Equação (4.9). Os alvos e as saídas são considerados como variáveis aleatórias U e V, respectivamente:

$$\text{AMI}(U, V) = \frac{I(U, V) - E[I(U, V)]}{\max\left(H(U), H(V)\right) - E[I(U, V)]}, \tag{4.9}$$

  onde E representa a esperança matemática, H representa a entropia e I representa a informação mútua. O valor esperado da informação mútua, considerando-se todas as possíveis "Tabelas de contingência"[61] entre duas soluções diferentes de clusterização, é representado por $E[I]$. A AMI é utilizada usualmente para a avaliação comparativa da qualidade entre dois sistemas de clusterização, assumindo valores reais entre zero (quanto os sistemas de clusterização divergem de maneira completamente aleatória entre si) e um (quando os sistemas de clusterização concordam completamente entre si) [62] e [63].

- *Data-Counting Metric* (DCM): em alguns conjuntos de dados utilizados para *spike sorting*, os números de *spikes* podem estar desbalanceados. À medida em que o desbalanceamento aumenta, pode acontecer de soluções de clusterização (obtidas por um mapeamento, $\mathbb{R}^D \rightarrow \mathbb{R}^L$, seguido por K-means, por exemplo) negligenciarem as classes com um número reduzido de elementos. Essas soluções poderiam não conter clusteres para estas classes e, mesmo assim, alcançar valores razoavelmente altos nas medidas de *hits*, acurácia, ou AMI. A métrica aqui proposta leva em consideração o problema apresentado. Para isso, ela trata com igual importância cada um dos i ($i = 1, 2, \ldots, N_{\text{hits}}$) clusteres considerados *hits*, segundo os critérios do (mapeamento) e da (maioria), independente do número de elementos contidos no i-ésimo cluster corretamente atribuídos ao seu neurônio associado. Mais objetivamente, a participação, via critério do mapeamento, do neurônio associado ao i-ésimo cluster é dada por N_i/T_i, onde N_i é o número de elementos no i-ésimo cluster corretamente atribuídos ao seu neurônio associado, e T_i é o número de elementos, na base de dados sob análise, referentes ao neurônio associado. A participação, via critério da maioria, do neurônio associado ao i-ésimo cluster é dada por N_i/K_i, onde K_i é o número total de elementos no i-ésimo cluster. Conforme a Equação (4.10), o valor da DCM é definido como N_{hits}/M^3 multiplicado pelo produto da soma das contribuições, onde M é o correto número de neurônios biológicos, presentes na base de dados sob análise, que geram *spikes* do tipo *single-unit*.

$$\text{DCM} = \frac{N_{\text{hits}}}{M^3} \sum_{i=1}^{N_{\text{hits}}} \frac{N_i}{T_i} \sum_{i=1}^{N_{\text{hits}}} \frac{N_i}{K_i}. \tag{4.10}$$

Segundo a métrica proposta, o erro de classificação dos elementos originários do neurônio associado ao cluster i torna-se menos relevante conforme T_i aumenta, e mais relevante conforme T_i diminui. Em outras palavras, a métrica proposta para a avaliação de desempenho não negligencia clusteres quando a base de dados sob análise apresenta um desbalanceamento significativo com relação ao número de spikes gerados por diferentes classes de neurônios.

Capítulo 5

Resultados e Discussões

O Capítulo 5 pode ser dividido em duas partes. A primeira parte, considerada a principal, compreende as Seções 5.1 e 5.2. Nelas os desempenhos dos algoritmos de mapeamento MLP-KLD, t-SNE, PCA, LPP e LDA foram avaliados em condições de: a) registros extracelulares contendo até 20 neurônios biológicos, Seção 5.1. b) registros extracelulares com diferentes níveis de ruído, Seção 5.2. A segunda parte, compreendendo as Seções 5.3, 5.4 e 5.5, apresenta experimentos complementares. Na Seção 5.3 investiga-se a existência de espaços de mapeamento com mais do que duas dimensões, que favoreçam o processo de *spike sorting*. Na Seção 5.4 foi realizado um estudo sobre o desempenho do clusterizador K-means básico, quando aplicado a mapeamentos obtidos a partir de MLPs com até sete camadas escondidas. Na Seção 5.5 foi proposta uma modificação na função-custo minimizada pelo algoritmo MLP-KLD. Originalmente o algoritmo MLP-KLD utiliza a métrica de distância "Euclideana"para o cálculo da p.m.f., conforme descrito no Capítulo 4. No experimento apresentado na Seção 5.5 a métrica de distância Euclideana é substituída pelas métricas "Cosseno"ou "*City Block*", e o desempenho do clusterizador K-means é avaliado quando aplicado a mapeamentos obtidos via algoritmo MLP-KLD, com métricas de distância "Euclideana", "Cosseno"ou "*City Block*".

Diferente dos métodos convencionais que, ao treinarem MLPs para tarefas de classificação ou de regressão, subdividem um determinado conjunto de dados em: conjunto de treinamento (utilizado para o ajuste dos parâmetros θ do MLP), conjunto de validação (utilizado para verificar a capacidade de generalização do MLP durante o treinamento) e conjunto de teste (utilizado para verificar o desempenho do MLP em condições reais). Neste trabalho, o MLP é treinado a partir de um conjunto de dados completo. Optou-se por não subdividir o conjunto de dados em conjunto de treinamento e conjunto de teste pelos motivos descritos a seguir: a) tradicionalmente as comparações de desempenho entre diferentes algoritmos de mapeamento, aplicados para *spike sorting*, são feitas em relação ao conjunto de dados completo [32], b) o MLP é treinado de forma não supervisionada (via MLP-KLD)

para aprender a superfície intrínseca de um conjunto de dados completo. Este tipo de aprendizado é conhecido como (*feature learning* ou *data representation*). Algoritmos de *feature learning* são tradicionalmente avaliados de forma indireta, através do desempenho de classificadores que são treinados de forma supervisionada, utilizando como entrada a nova representação do conjunto de dados (características aprendidas) [64, 65]. O critério de parada do algoritmo MLP-KLD é determinado pelo número de iterações (épocas), e o desempenho do MLP é avaliado de forma indireta pelas métricas apresentadas no Capítulo 4.

5.1 Desempenho do MLP-KLD em Função do Número de Neurônios Biológicos

O objetivo do experimento apresentado nesta seção é verificar se o algoritmo MLP-KLD pode ser utilizado para problemas de *spike sorting* em condição de aumento do número de neurônios presentes nos registros extracelulares. A topologia escolhida para os MLPs treinados via algoritmo MLP-KLD foi a mais simples possível, ou seja, MLPs contendo apenas uma camada escondida composta de neurônios artificiais do tipo sigmoidal e camada de saída composta por dois neurônios artificiais do tipo linear. Como o objetivo desta seção não é treinar o melhor MLP possível, mas apenas verificar se o algoritmo MLP-KLD pode ser aplicado para problemas de *spike sorting* em condição de aumento do número de neurônios, o número de neurônios artificiais da camada escondida dos MLPs pode ser escolhido de forma arbitrária. A Seção 5.1.2 descreve um experimento que foi realizado para auxiliar na determinação do número de neurônios artificiais da camada escondida dos MLPs.

5.1.1 Formação dos Conjuntos de Dados

Para este experimento foi utilizada a base de dados "Data Set 1". Conforme descrito no Apêndice A , o Data Set 1 é composto de 95 registros extracelulares. Há cinco registros contendo *spikes* gerados por dois neurônios distintos, cinco registros contendo *spikes* gerados por três neurônios distintos, e assim por diante até cinco registros contendo *spikes* gerados por 20 neurônios distintos. Para cada registro extracelular, são realizados os seguintes procedimentos: a informação do instante em que o *spike* ocorre ao longo do registro extracelular é utilizada para a sua correta detecção e segmentação. O *spike* é extraído com comprimento de 79 amostras (dimensões), de modo que seu ponto de máximo (pico) esteja localizado na amostra 40, e armazenado em uma matriz (conjunto de dados). Cada linha da matriz formada corresponde a um *spike* diferente, e as colunas da matriz correspondem às dimensões dos *spikes* armazenados. Cada uma das 79 colunas da matriz formada passa por

uma normalização linear que faz com que a coluna tenha valor mínimo igual a zero e valor máximo igual a um. Portanto, os 95 registros extracelulares dão origem a 95 matrizes, com diferentes números de linhas correspondendo ao total de *spikes* presentes em cada registro extracelular. A partir deste ponto a matriz gerada, na sua forma normalizada, será chamada de "conjunto de dados".

O conjunto de dados é submetido a um algoritmo de spike sorting que consiste em um mapeamento de 79 para duas dimensões, a ser projetado (com base em MLP-KLD, t-SNE, PCA, LDA ou LPP), seguido de clusterização através do algoritmo K-means básico. Alguns resultados de mapeamentos de 79 para duas dimensões podem ser vistos na Figura 5.1. Esses resultados foram obtidos a partir de conjuntos de dados contendo 7, 14 e 20 neurônios distintos. Pontos próximos entre si no espaço 2-D são interpretados como spikes similares. Em seguida, é aplicado o K-means básico com K dado a priori ($K = 7$, $K = 14$ ou $K = 20$, no exemplo dos conjuntos de dados da Figura 5.1) e são calculadas as medidas de desempenho descritas no Capítulo 4. O número de clusteres encontrados pelo K-means estará correto, mas pode haver erros que levem a N_{hits} inferior a K, e assim a um número de misses ou clusteres falso-positivos diferente de zero.

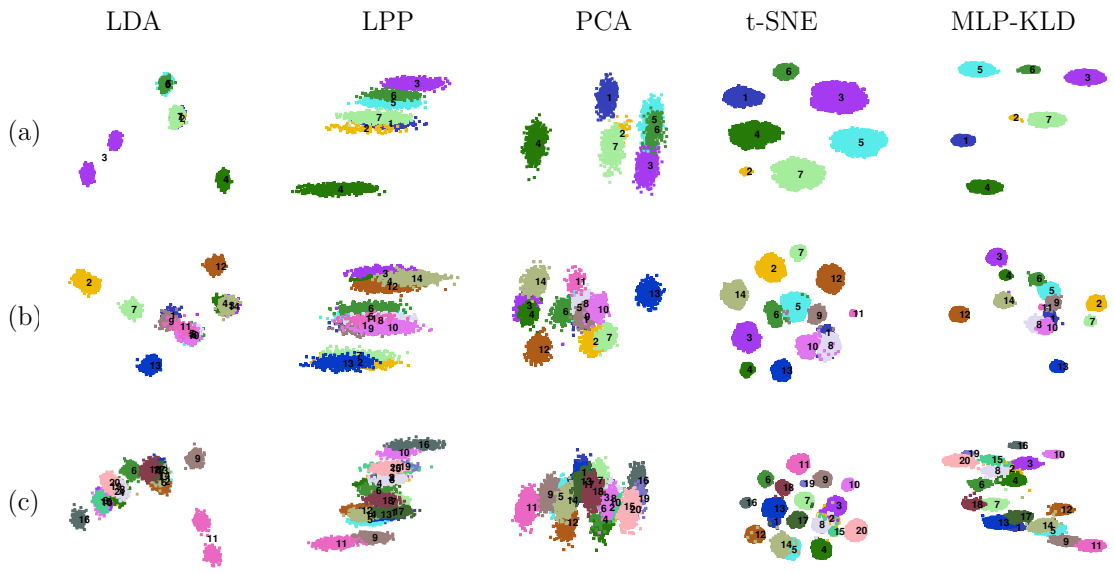


Figura 5.1: Comparação visual entre mapeamentos de 79 para duas dimensões obtidos através dos algoritmos: MLP-KLD, t-SNE, PCA, LDA e LPP. Conjuntos de dados contendo (a) 7 neurônios; (b) 14 neurônios, e (c) 20 neurônios distintos. Percebe-se visualmente que, no caso de 20 neurônios, os mapeamentos *MLP-KLD* e *t-SNE* favorecem a clusterização com sobreposição mínima entre as classes (subconjuntos de pontos que vêm de um mesmo neurônio) presentes no conjunto de dados. Para colorir estas classes nesta figura, as etiquetas originais foram utilizadas.

5.1.2 Número de Neurônios Artificiais da Camada Escondida

Para este experimento foram selecionados todos os conjuntos de dados gerados a partir de registros extracelulares contendo 3, 7, 14 e 20 neurônios, o que representa um total de 20 conjuntos de dados. Para cada um dos 20 conjuntos de dados foram treinados MLPs contendo 30, 200, 400, 600 e 800 neurônios artificiais na camada escondida, o que corresponde a um total de 100 MLPs. Os parâmetros de treinamento adotados foram: $\alpha = 0.1$, $\beta = 0.1$, $\lambda = 10^{-4}$, $\gamma = 10^{-2}$ e máximo de épocas igual a 1.000, sendo que a cada época 10% das sinapses foram mantidas fixas[1]. Para cada MLP treinado, sua saída (pontos mapeados para o espaço 2-D) foi clusterizada via K-means básico e o número de hits (N_{hits}) foi calculado.

Tabela 5.1: Erro médio quadrático entre o número médio de hits e o número exato de neurônios presentes em conjuntos de dados contendo 3, 7, 14 ou 20 neurônios biológicos. São considerados sistemas de *spike sorting* completos com MLP-KLD e clusterização. Os MLPs treinados possuem camada escondida contendo 30, 200, 400, 600 ou 800 neurônios artificiais.

	# Neurônios				
	3	7	14	20	MSE
# Médio de N_{hits}(79-30-2)	3	6,6	12,2	16,0	4,85
# Médio de N_{hits}(79-200-2)	3	6,6	12,2	16,0	4,85
# Médio de N_{hits}(79-400-2)	3	6,6	12,8	16,6	3,29
# Médio de N_{hits}**(79-600-2)**	3	6,6	12,4	16,8	**3,24**
# Médio de N_{hits}(79-800-2)	3	6,6	12,4	16,4	3,92

Na Tabela 5.1, o número de neurônios artificiais na camada escondida dos diferentes MLPs treinados a partir dos conjuntos de dados selecionados muda de uma linha para outra. As colunas "# Neurônios"indicam os números de neurônios presentes nos conjuntos de dados. O valor apresentado na linha (79-30-2) e coluna (3) corresponde à média entre os números de *hits* obtidos a partir de mapeamentos produzidos por cinco diferentes MLPs contendo 30 neurônios artificiais na camada escondida (para cada conjunto de dados distinto foi treinado um MLP). O valor na linha (79-30-2) e coluna MSE corresponde ao erro médio quadrático entre os números médios de hits, contidos na linha (79-30-2), e o exato número de neurônios presentes nos conjuntos de dados selecionados.

É possível notar que a topologia 79-600-2 está associada ao mínimo MSE entre número médio de hits e número exato de neurônios. Esta topologia foi, então,

[1]Detalhes sobre a escolha dos hiperparâmetros são descritos no Apêndice C.

usada para o treinamento dos MLPs com todos os 95 conjuntos de dados disponíveis (conjuntos de dados contendo de 2 até 20 neurônios), levando assim aos resultados referentes a MLP-KLD apresentados nas Figuras 5.1 e 5.2 e nas Tabelas 5.2 e 5.3. Nos treinamentos realizados para a obtenção destes resultados, os valores dos parâmetros α, β, λ e γ foram mantidos. O número máximo de épocas foi aumentado para 3.000.

Tabela 5.2: Algoritmo K-means básico aplicado a conjuntos de dados 2-D obtidos através do mapeamento de 79 para duas dimensões via MLP-KLD, t-SNE, PCA, LDA e LPP. Os resultados mostrados correspondem a número de *hits*, número de *misses* e número de clusteres falso-positivos, em problemas de *spike sorting* com número de neurônios variando de 2 a 20. Em negrito estão marcados os melhores resultados de cada coluna.

	# Neurônios	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
MLP-KLD	Hits	**2**	**3**	**3,6**	**4,8**	**6**	6,6	**7,6**	**8,4**	9,4	**10**	11,2	**12,2**	**13,2**	**14**	14,8	**16**	16,4	**17**	**17,8**
	Misses	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Falso Positivos	0	0	0,4	0,2	0	0,4	0,4	0,6	0,6	1	0,8	0,8	0,8	1	1,2	1	1,6	2	2,2
t-SNE	Hits	**2**	**3**	**3,6**	**4,8**	5,8	**6,8**	7,4	6,25	**9,8**	9,6	**11,8**	**12,2**	12,8	13,6	**15**	15,4	**17**	16,8	**17,8**
	Misses	0	0	0	0	0	0	0	2	0	0,4	0	0	0,2	0	0	0	0	0	0
	Falso Positivos	0	0	0,4	0,2	0,2	0,2	0,6	0,75	0,2	1	0,2	0,8	1	1,4	1	1,6	1	2,2	2,2
PCA	Hits	**2**	**3**	**3,6**	4,6	5,2	6	6,6	7,6	8,4	8,2	9,4	9,2	10,4	11	11,6	11,4	13	10,2	13,4
	Misses	0	0	0	0	0,2	0	0	0	0,4	1,2	0,8	1	1,6	0,4	1,4	2,8	1,6	3	1,6
	Falso Positivos	0	0	0,4	0,4	0,6	1	1,4	1,4	1,2	1,6	1,8	2,8	2	3,6	3	2,8	3,4	5,8	5
LPP	Hits	**2**	2,8	**3,6**	4,6	5,2	5,8	6,6	6,8	7	9	9,4	9	9	9,4	7,2	10,4	9,4	8	10,8
	Misses	0	0	0	0	0	0,4	0,2	1	1,2	0,4	0,2	1,6	1,6	2,4	3,6	2,6	5,2	4,6	3,6
	Falso Positivos	0	0,2	0,4	0,4	0,8	0,8	1,2	1,2	1,8	1,6	2,4	2,4	3,4	3,2	5,2	4	3,4	6,4	5,6
LDA	Hits	**2**	**3**	**3,6**	4,2	4,4	4,6	5,4	5,8	5,2	6	6,6	7,2	7,6	7,6	7,8	7,4	6,8	9,6	7,2
	Misses	0	0	0	0	0	0,4	0,8	1,4	1,8	3	2,2	2	3	3,2	4,6	5,8	6	3,8	7,8
	Falso Positivos	0	0	0,4	0,8	1,6	2	1,8	1,8	3	2	3,2	3,8	3,4	4,2	3,6	3,8	5,2	5,6	5

5.1.3 Avaliação dos Algoritmos

A variação do desempenho do mapeamento, que é estimado através do número de *hits*, do número de *misses* e do número de clusteres falso-positivos do K-means básico aplicado aos dados mapeados, é mostrada na Tabela 5.2. O número de *hits* propiciado pelo mapeamento MLP-KLD aumenta conforme o aumento do número de neurônios, chegando à média de 17,8 *hits* com conjuntos de dados contendo 20 neurônios. Também no caso do mapeamento MLP-KLD, o número de *misses* igual a zero para registros com todos os números de neurônios indica que há pouca sobreposição entre os dados de clusteres diferentes. O baixo número de clusteres

falso-positivos indica que, após o mapeamento de 79 para duas dimensões, os *spikes* provenientes de um mesmo neurônio tipicamente se agrupam em um mesmo cluster e a formação de sub-clusteres não acontece. O desempenho do K-means básico aplicado aos dados provenientes do mapeamento t-SNE é semelhante ao propiciado pelo mapeamento MLP-KLD. O número de *hits* também aumenta conforme o aumento do número de neurônios, atingindo o mesmo valor médio máximo (17,8 *hits*), com conjuntos de dados contendo 20 neurônios. O número de clusteres falso-positivos também não ultrapassa 2,2. O número de *misses* é igual a zero em quase todos os casos. Já o desempenho propiciado pelo mapeamento PCA é consideravelmente inferior, especialmente para conjuntos de dados contendo mais do que dez neurônios. A partir de 11 neurônios, os números de clusteres falso-positivos e de *misses* aumentam conforme o número de neurônios. O valor médio do número de *hits* não ultrapassa 13,4 (para conjuntos de dados contendo 20 neurônios). Os piores desempenhos são obtidos a partir dos mapeamentos LDA e LPP. Os números de *hits* ficam limitados a 10,8 *hits* (LPP) e 7,2 *hits* (LDA), para conjuntos de dados contendo 20 neurônios. Os números de clusteres falso-positivos e de *misses* também são maiores que os obtidos com PCA.

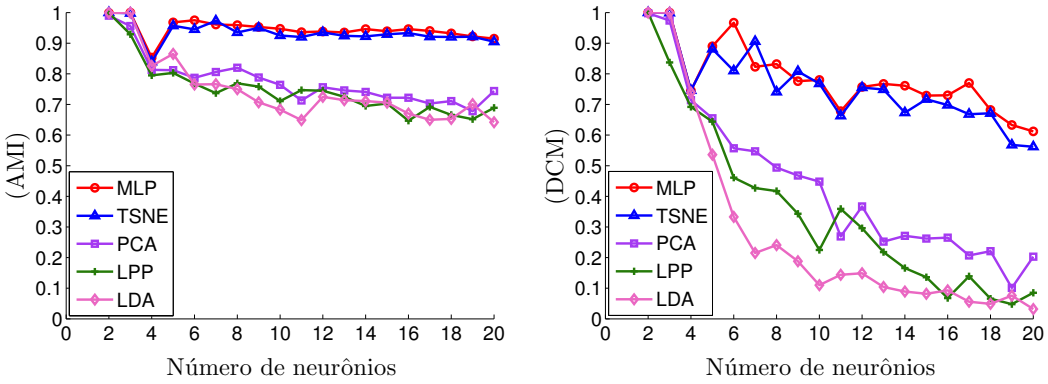
Na Tabela 5.3, são mostrados valores médios dos números de erros e das acurácias de classificação. Os valores médios são calculados a partir dos resultados (número de erros ou acurácia) obtidos de conjuntos de dados contendo o mesmo número de neurônios. A coluna "#Spikes" indica o número médio de *spikes* para conjuntos de dados contendo o mesmo número de neurônios. O mapeamento baseado em MLP-KLD levou ao clustering/*spike sorting* com melhores resultados de classificação (menor erro e maior acurácia) em quase todos os casos. O erro se manteve baixo e a acurácia se manteve alta (isto é, erro inferior e acurácia superior aos alcançados com base nos outros mapeamentos) à medida em que conjuntos de dados com mais neurônios são sucessivamente considerados. A última linha da Tabela 5.3 apresenta os valores médios globais, isto é, os valores médios de número de erros e acurácia calculados sobre todos os números de neurônios (de 2 até 20). Os mapeamentos MLP-KLD e t-SNE são claramente superiores, em termos da acurácia alcançada pelos respectivos sistema de *spike sorting*. O pior desempenho é obtido pelos sistemas de *spike sorting* baseados em mapeamento LDA, que chegam a ter erro 6.8 vezes superior ao do sistema de *spike sorting* baseado em mapeamento MLP-KLD.

A Figura 5.2 mostra comparações entre a qualidade da clusterização K-means básica, avaliada através de AMI, Figura 5.2(a), e a mesma qualidade avaliada por DCM, conforme a Figura 5.2(b), alcançada em sistemas de *spike sorting* baseados em MLP-KLD, t-SNE, PCA, LDA ou LPP. Em ambas as avaliações (AMI ou DCM), o desempenho se mantém razoavelmente constante à medida em que o número de neurônios aumenta, quando o mapeamento para duas dimensões é feito através

Tabela 5.3: Número de erros e acurácia alcançados pelo algoritmo K-means básico executado sobre dados 2-D gerados através dos algoritmos MLP-KLD, t-SNE, PCA, LDA e LPP. Os valores indicados correspondem às médias calculadas sobre cinco registros diferentes para cada número de neurônios. Os valores de acurácia são mostrados entre parênteses. Em negrito estão marcados os melhores resultados.

# Neurônios	# Spikes	LDA	LPP	PCA	t-SNE	MLP-KLD
2	1.369,8	**0,20 (99,99)**	**0,20 (99,99)**	2,40 (99,88)	0,40 (99,98)	**0,20 (99,99)**
3	1.583,4	**0 (100)**	142,60 (92,31)	15,40 (99,05)	0,40 (99,98)	**0 (100)**
4	1.784,8	204,2 (84,1)	303,2 (82,6)	208,8 (84,2)	**170,2 (86,7)**	180,20 (85,9)
5	3.616	843,4 (78,8)	650,4 (82,5)	578 (85,02)	125,4 (96,6)	**93,60 (97,5)**
6	3.543,8	1.214,4 (63,6)	830,2 (76,1)	739,4 (79,9)	210 (94,2)	**46 (98,74)**
7	4.152	1.777,4 (57,8)	1.264,2 (69,8)	792,6 (80,2)	**108 (96,6)**	220,40 (94,3)
8	4.211,6	1.771,8 (58,7)	1.245,4 (71,5)	892,4 (78,8)	403 (91,1)	**209,80 (95,1)**
9	5.356,4	2.543,4 (50,4)	1.843,2 (64,8)	1.218,2 (76,8)	**292,50 (94,1)**	360,80 (93,1)
10	5.752,4	3.491,8 (39,4)	2.597,4 (56,3)	1.667,2 (70,9)	447,4 (91,9)	**341,40 (93,9)**
11	6.256	3.773,2 (39,9)	2.399,8 (62,7)	2.433,4 (61,3)	697,2 (88,6)	**643,20 (89,6)**
12	6.945,4	3.747 (47,5)	2.555,2 (62,8)	2.322,6 (66,4)	603 (91,1)	**571,20 (91,6)**
13	7.786,8	4.320 (43,2)	3.372,4 (55,5)	2.948,2 (60,1)	680 (89,9)	**611 (90,8)**
14	8.693,4	5.212,4 (40,0)	4.642,8 (46,3)	3.334 (61,8)	889,4 (89,7)	**643 (92,5)**
15	9.330,8	5.716,8 (39,0)	5.019,6 (46,7)	3.856,8 (58,6)	870,4 (90,6)	**832,60 (90,9)**
16	9.024,6	5.840,2 (36,3)	6.310 (29,8)	3.850 (57,3)	881,6 (89,5)	**743,40 (91,1)**
17	10.640,8	7.169,4 (32,5)	5.957 (43,8)	5.038,6 (52,4)	1.363 (87,0)	**764,40 (92,7)**
18	10.477,6	7.670,6 (27,2)	6.493,4 (37,9)	4.611,2 (56,5)	1.388 (87,0)	**959,40 (90,8)**
19	11.222,6	6.842,2 (38,3)	7.868,2 (29,4)	6.562,2 (42,2)	1.466 (86,9)	**1.407,60 (87,4)**
20	11.497,6	8.462,6 (26,6)	6.987,4 (39,4)	5.103,6 (55,4)	2.011 (82,2)	**1.646,60 (85,5)**
Valor Médio	6.486,6	3.715,8 (52,8)	3.183,1 (60,5)	2.430,3 (69,8)	663,5 (91,3)	**540,9 (92,7)**

de MLP-KLD ou t-SNE. Por outro lado, o desempenho piora claramente, com o aumento do número de neurônios, nos casos em que o mapeamento é feito por PCA, LDA ou LPP. A diferença entre os valores de DCM para baixo número de neurônios (próximos de 1, para 2 ou 3 neurônios) e para alto número de neurônios (inferiores a 0,8, para 10 ou mais neurônios) é maior do que a diferença entre os valores de AMI, pois para o cálculo da DCM são considerados o número de *hits*, e as participações via critérios de maioria e mapeamento dadas pelos neurônios associados. Quanto maior for o número de *hits* e quanto menor for o número de *spikes* mapeados para clusteres errados, menor será o decaimento descrito pela curva de DCM. A Figura 5.2(b) apresenta de maneira mais evidente a diferença entre os resultados de clusterização.



(a) AMI (b) DCM

Figura 5.2: Desempenho de algoritmo de clusterização básico (K-means) em função do número de neurônios presentes nos conjuntos de dados, indicado através de AMI (a) e DCM (b). Em ambas as avaliações (AMI ou DCM), o desempenho se mantém razoavelmente constante à medida em que o número de neurônios aumenta, quando o mapeamento para duas dimensões é feito através de MLP-KLD ou t-SNE. Por outro lado, o desempenho piora claramente nos casos em que o mapeamento é feito por PCA, LDA ou LPP. A diferença entre os maiores e menores valores de DCM é maior do que a diferença entre os maiores e menores valores de AMI.

5.1.4 Discussão

O experimento realizado na Seção 5.1 mostrou a manutenção do desempenho da clusterização obtida sobre dados 2-D mapeados através de MLP-KLD ou através de t-SNE, ambos apresentaram desempenho médio de classificação de 17.8 *hits* para conjuntos de dados contendo 20 neurônios. Os resultados da clusterização sobre dados 2-D mapeados através de PCA, LDA ou LPP apresentaram redução de desempenho com o aumento do número de neurônios presentes nos registros extracelulares, o que sugere que estes três métodos têm maior dificuldade para realizar um mapeamento adequado à medida em que os *spikes* provenientes de neurônios diferentes se tornam similares.

Outros trabalhos também fizeram uso do "Data Set 1"para avaliar seus métodos de *spike sorting*. Em [23] o *spike sorting* foi realizado com o auxílio do programa Wave-clus [33] e a etapa de classificação foi realizada de maneira independente por três especialistas. O desempenho médio de classificação obtido pelos especialistas foi inferior à 10 *hits* em conjuntos de dados contendo 20 neurônios. Em [66] foi proposto um método automático de *spike sorting*, o qual apresentou um desempenho médio de classificação ligeiramente superior ao descrito no trabalho [23]. O desempenho médio de classificação obtido foi de 10,5 *hits* em conjuntos de dados contendo 20 neurônios. Em [67] uma nova versão do programa Wave-clus foi proposta, com a inclusão de novas métricas de similaridade entre clusteres. O desempenho médio de

classificação obtido foi de 14 *hits* para conjuntos de dados contendo 20 neurônios.

5.2 Desempenho do MLP-KLD em condição de Aumento do Nível de Ruído.

O objetivo do experimento apresentado nesta seção é verificar se o algoritmo MLP-KLD pode ser utilizado para problemas de *spike sorting* quando a componente de ruído presente no registro extracelular não é considerada desprezível. Em outras palavras, os algoritmos MLP-KLD, t-SNE, PCA, LDA e LPP são avaliados em uma base de dados cujos registros extracelulares apresentam diferentes níveis de ruído. Assim como na Seção 5.1, a topologia escolhida para os MLPs foi a mais simples possível, ou seja, MLPs contendo apenas uma camada escondida composta de neurônios artificiais do tipo sigmoidal e camada de saída composta por dois neurônios artificiais do tipo linear. Para auxiliar na escolha do número de neurônios artificiais da camada escondida, um experimento foi realizado, conforme apresentado na Seção 5.2.2.

5.2.1 Formação dos Conjuntos de Dados

Para este experimento foi utilizada a base de dados "Data Set 2". Conforme descrito no Apêndice A, o Data Set 2 é composto por 20 registros extracelulares. Cada um dos registros contém *spikes* gerados por três neurônios distintos. Os registros são separados nas categorias "easy1", "easy2", "difficult1"e "difficult2", sendo a primeira considerada a menos desafiadora e a última considerada a mais desafiadora. Para cada registro extracelular são realizados os seguintes procedimentos: a informação do instante em que o *spike* ocorre ao longo do registro extracelular é utilizada para a sua correta detecção e segmentação (os *spikes* considerados sobrepostos foram excluídos). O *spike* é extraído com comprimento de 64 amostras, de modo que seu ponto de máximo esteja localizado na amostra 31. Após a extração, o *spike* é convoluído com uma wavelet do tipo "sym2"(família Symlets) e fator de escalamento $a = 1$, via transformada wavelet contínua [2], resultando em um vetor de coeficientes com o mesmo número de dimensões que o do *spike* original. O vetor de coeficientes é armazenado em uma matriz (conjunto de dados). Cada linha da matriz corresponde a um diferente vetor de coeficientes, e as colunas da matriz correspondem às dimensões dos vetores armazenados. Cada uma das 64 colunas da matriz passa por uma normalização linear que faz com que a coluna tenha valor mínimo igual a zero e valor máximo igual a um. Portanto, os 20 registros extracelulares dão origem

[2]Detalhes sobre o uso da transformada wavelet contínua como um extrator de características são descritos no Apêndice B.

a 20 matrizes, com diferentes números de linhas correspondendo ao total de *spikes* presentes em cada registro extracelular.

O conjunto de dados é submetido a um algoritmo de *spike sorting* que consiste em um mapeamento de 64 para duas dimensões, a ser projetado (com base em MLP-KLD, t-SNE, PCA, LDA ou LPP), seguido de clusterização através do algoritmo K-means básico. Alguns resultados de mapeamentos de 64 para duas dimensões podem ser vistos na Figura 5.3. Esses resultados foram obtidos a partir de registros extracelulares da categoria "difficult2"e níveis de ruído (0,05; 0,1; 0,15 e 0,2). Pontos próximos entre si no espaço 2-D são interpretados como *spikes* similares. Em seguida, é aplicado o K-means básico com K dado a priori ($K = 3$, no exemplo dos conjuntos de dados da Figura 5.3) e são calculadas as medidas de desempenho descritas no Capítulo 4.

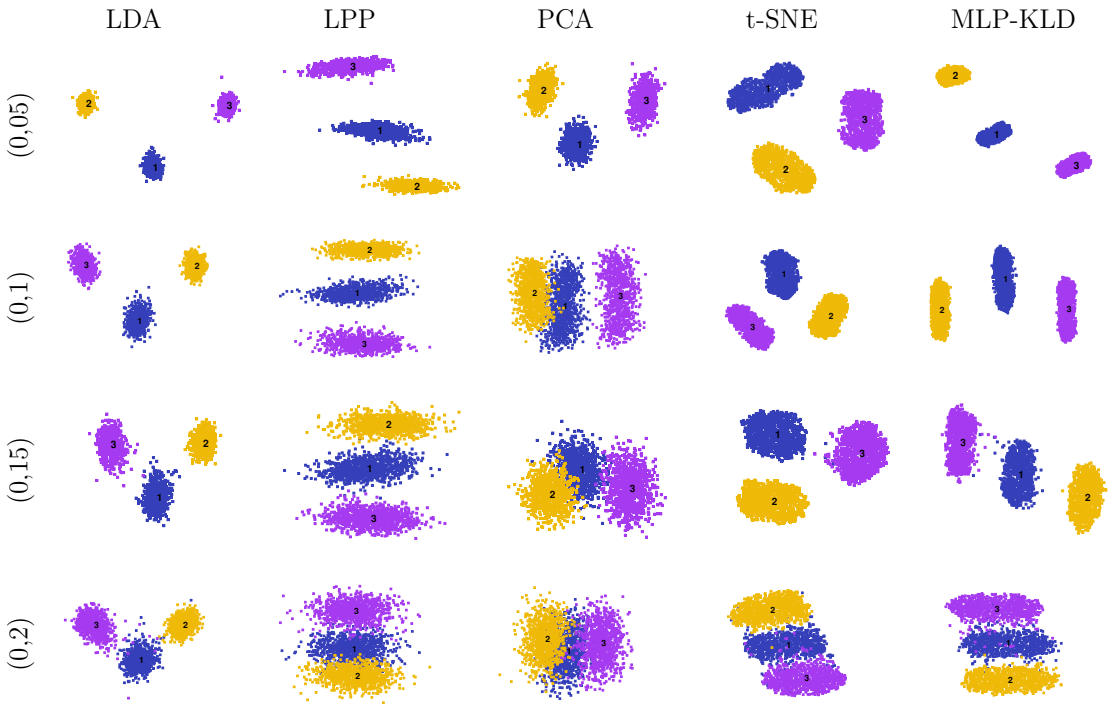


Figura 5.3: Comparação visual entre mapeamentos de 64 para duas dimensões obtidos através dos algoritmos MLP-KLD, t-SNE, PCA, LDA e LPP. Resultados obtidos para registros extracelulares da categoria "difficult2"e níveis de ruído (0,05; 0,1; 0,15 e 0,2). Percebe-se visualmente que os mapeamentos geram clusteres mais próximos a medida em que o nível de ruído aumenta; sugerindo que os métodos comparados apresentam maior dificuldade para realizar um mapeamento adequado dado o aumento do nível de ruído. Os mapeamentos MLP e LDA favorecem a clusterização com sobreposição mínima entre as classes (subconjuntos de pontos que vêm de um mesmo neurônio) presentes no conjunto de dados. Para colorir estas classes nesta figura, as etiquetas originais foram utilizadas.

5.2.2 Número de Neurônios Artificiais da Camada Escondida

Para este experimento foram selecionados os conjuntos de dados formados a partir dos registros extracelulares considerados os mais difíceis dentro de cada categoria, sendo eles: (a) "easy1"com nível de ruído 0,4, e (b) "easy2", "difficult1"e "difficult2"ambos com nível de ruído 0,2. Para cada um dos quatro conjuntos de dados foram treinados MLPs contendo 30, 200, 400, 600 e 800 neurônios artificiais na camada escondida, o que corresponde a um total de 20 MLPs. Os parâmetros de treinamento adotados foram: $\alpha = 0.1$, $\beta = 0.1$, $\lambda = 10^{-4}$, $\gamma = 10^{-2}$ e máximo de épocas igual a 3.000, sendo que a cada época 30% das sinapses foram mantidas fixas [3]. Para cada MLP treinado, sua saída (pontos mapeados para o espaço 2-D) foi clusterizada via K-means básico e o número de hits (N_{hits}) foi calculado.

Tabela 5.4: Número de erros e acurácia alcançados pelo algoritmo K-means básico executado sobre dados 2-D gerados por MLPs contendo 30, 200, 400, 600 ou 800 neurônios na camada escondida, treinados através do algoritmo MLP-KLD. Os conjuntos de dados selecionados são considerados os mais difíceis. Os valores de acurácia são mostrados entre parênteses. Em negrito estão marcados os melhores resultados.

Data Set 2	Nível de Ruído	64-30-2	64-200-2	64-400-2	64-600-2	64-800-2
easy1	0,40	617 (76,7)	536 (79,7)	**510 (80,7)**	520 (80,3)	515 (80,5)
easy2	0,20	46 (98,3)	40 (98,5)	**12 (99,5)**	48 (98,2)	51 (98,1)
difficult1	0,20	10 (99,6)	10 (99,6)	**3 (99,8)**	4 (99,8)	4 (99,8)
difficult2	0,20	122 (95,5)	99 (96,3)	**71 (97,3)**	81 (97,0)	72 (97,3)
Valor Médio		198,8 (92,5)	171,3 (93,5)	**149 (94,3)**	163,3 (93,8)	160,5 (93,9)

A primeira coluna da Tabela 5.4 indica a categoria dos registros selecionados, a segunda coluna indica o nível de ruído, e as demais colunas indicam os erros e acurácia (entre parênteses) de classificação, para MLPs contendo 30, 200, 400, 600 ou 800 neurônios artificais em sua camada escondida. Diferente da Seção 5.1 em que o valor do MSE entre o número médio de *hits* e o número exato de neurônios presentes no conjunto de dados foi utilizado como critério de seleção da topologia, na Seção 5.2 a topologia é selecionada com base no valor médio da acurácia de classificação.[4]

[3]Detalhes sobre a escolha dos hiperparâmetros são descritos no Apêndice C.

[4] O valor do MSE entre o número médio e o número exato de neurônios presentes no conjunto de dados não foi utilizado como critério para selecionar a topologia, pois para todos os conjuntos de dados selecionados o número de *hits* calculados foi igual ao número exato de neurônios presentes no conjunto de dados, levando a um MSE igual a zero para todos os MLPs treinados, o que impossibilitou a escolha da topologia pelo critério do MSE.

Pode-se ver que a topologia 64-400-2 está associada ao máximo valor médio de acurácia de classificação. Esta topologia foi então utilizada para o treinamento dos MLPs com todos os 20 conjuntos de dados disponíveis (conjuntos de dados contendo diferentes níveis de ruído, porém apenas três neurônios distintos), levando assim aos resultados referentes ao MLP-KLD apresentados nas Figuras 5.4 e 5.5 e na Tabela 5.5. Nos treinamentos utilizados para a obtenção destes resultados, os valores dos parâmetros α, β, λ, γ e número de iterações foram mantidos.

5.2.3 Avaliação dos Algoritmos

Para todos os 20 conjuntos de dados e para todos os mapeamentos gerados via algoritmos MLP-KLD, t-SNE, LDA, PCA e LPP, os números de *hits*, *misses* e clusteres falso-positivos obtidos foram iguais a (3, 0 e 0) respectivamente. Portanto, pelo critério de mapeamento, em condições de baixo número de neurônios presentes nos conjuntos de dados, os algoritmos comparados apresentaram desempenhos iguais independente da variação do nível de ruído.

Na Tabela 5.5, são mostrados valores dos números de erros e acurácias de classificação. A primeira coluna indica a categoria do conjunto de dados, a coluna "#Spikes" indica o número de *spikes* para cada conjunto de dados e a coluna "Nível de Ruído"indica o nível de ruído presente no registro extracelular. O mapeamento baseado em MLP-KLD levou a resultados de clustering/*spike sorting* equivalentes aos obtidos pelos algoritmos t-SNE e LDA à medida em que o nível de ruído aumenta. A última linha da Tabela 5.5 apresenta os valores médios globais, isto é, os valores médios de número de erros e acurácia calculados sobre todos os conjuntos de dados. O mapeamento LDA apresentou desempenho ligeiramente superior com acurácia média de (97,34%), seguido pelos algoritmos MLP-KLD (97,29%) e t-SNE(97,24%) praticamente empatados. Os menores desempenhos são obtidos pelos sistemas de *spike sorting* baseados em PCA (91,43%) ou LPP (91,32%). A maior diferença de desempenho entre os algoritmos LDA e MLP-KLD ocorreu para o conjunto de dados da categoria "difficult2"e nível de ruído 0,2. Para este conjunto de dados o LDA apresentou apenas 18 erros comparados aos 71 erros do MLP-KLD. Para os demais conjuntos de dados as acurácias do LDA e do MLP-KLD foram consideradas equivalentes.

As Figuras 5.4 e 5.5 mostram, respectivamente, comparações entre a qualidade da clusterização via K-means básico, avaliada através das métricas AMI e DCM, alcançada em sistemas de *spike sorting* baseados em MLP-KLD, t-SNE, PCA, LDA ou LPP. As curvas mostram a manutenção do desempenho dos algoritmos MLP-KLD, t-SNE e LDA diante da variação do nível de ruído, e uma maior dificuldade de discriminação dos dados por parte dos algoritmos PCA e LPP. Os resultados

Tabela 5.5: Número de erros e acurácia alcançados pelo algoritmo K-means básico executado sobre dados 2-D gerados através dos algoritmos MLP-KLD, t-SNE, PCA, LDA e LPP. Os conjuntos de dados não incluem casos de sobreposição de *spikes*. Os números em negrito correspondem aos melhores desempenhos para cada conjunto de dados. Os valores de acurácia são mostrados entre parênteses. Em negrito estão marcados os melhores resultados.

Data Set 2	Nível de Ruído	# Spikes	LDA	LPP	PCA	t-SNE	MLP-KLD
easy1	0,05	2.729	**0 (100)**	**0 (100)**	**0 (100)**	**0 (100)**	**0 (100)**
	0,10	2.753	**0 (100)**	**0 (100)**	**0 (100)**	**0 (100)**	**0 (100)**
	0,15	2.693	5 (99,8)	**3 (99,8)**	5 (99,8)	5 (99,8)	5 (99,8)
	0,20	2.678	**50 (98,1)**	716 (73,3)	51 (98,1)	51 (98,1)	51 (98,1)
	0,25	2.586	128 (95,1)	629 (75,4)	128 (95,0)	**128 (95,0)**	**128 (95,0)**
	0,30	2.629	261 (90,1)	472 (82,0)	261 (90,1)	**259 (90,1)**	**259 (90,1)**
	0,35	2.702	395 (85,4)	849 (68,6)	395 (85,4)	**391 (85,5)**	**391 (85,5)**
	0,40	2.645	514 (80,5)	760 (71,3)	525 (80,1)	**498 (81,2)**	510 (80,7)
easy2	0,05	2.619	**0 (100)**	**0 (100)**	**0 (100)**	**0 (100)**	**0 (100)**
	0,10	2.694	**0 (100)**	**0 (100)**	5 (99,8)	**0 (100)**	**0 (100)**
	0,15	2.648	7 (99,7)	9 (99,6)	89 (96,6)	**2 (99,9)**	**2 (99,9)**
	0,20	2.715	29 (98,9)	50 (98,2)	244 (91,0)	26 (99,0)	**12 (99,5)**
difficult1	0,05	2.616	**0 (100)**	**0 (100)**	7 (99,7)	**0 (100)**	**0 (100)**
	0,10	2.638	**0 (100)**	384 (85,4)	521 (80,2)	**0 (100)**	**0 (100)**
	0,15	2.660	**0 (100)**	524 (80,3)	697 (73,8)	**0 (100)**	**0 (100)**
	0,20	2.624	**3 (99,8)**	9 (99,6)	765 (70,8)	5 (99,8)	**3 (99,8)**
difficult2	0,05	2.535	**0 (100)**	**0 (100)**	**0 (100)**	**0 (100)**	**0 (100)**
	0,10	2.742	**0 (100)**	2 (99,9)	140 (94,9)	**0 (100)**	**0 (100)**
	0,15	2.631	**1 (99,9)**	32 (98,7)	167 (93,6)	5 (99.8)	5 (99,8)
	0,20	2.716	**18 (99,3)**	168 (93,8)	559 (79,4)	97 (96,4)	71 (97,4)
Valor médio		2.662,7	**70,55 (97,34)**	230,35 (91,32)	227,95 (91,43)	73,35 (97,24)	71,85 (97,29)

concordam com os apresentados na Figura 5.3, onde os mapeamentos obtidos via PCA e LPP produziram clusteres visualmente com maior sobreposição.

5.2.4 Discussão

Vários trabalhos [68–70] têm sido desenvolvidos na tentativa de minimizar ou modelar as diversas fontes de ruído presentes no registro extracelular. Segundo [68], as principais fontes de ruído são: impedância da interface eletrodo-eletrólito [71], ruído biológico (*spikes* de baixa amplitude), ruído eletrônico (térmico e *flicker noise* [72]). A compreensão de tais fontes é de extrema importância para o desenvolvimento de novos equipamentos ou técnicas de registro de sinal. Mesmo com os avanços na tecnologia de aquisição, não se pode garantir por completo a ausência de ruído. A avaliação dos algoritmos de *spike sorting* em condição de variação do nível de ruído

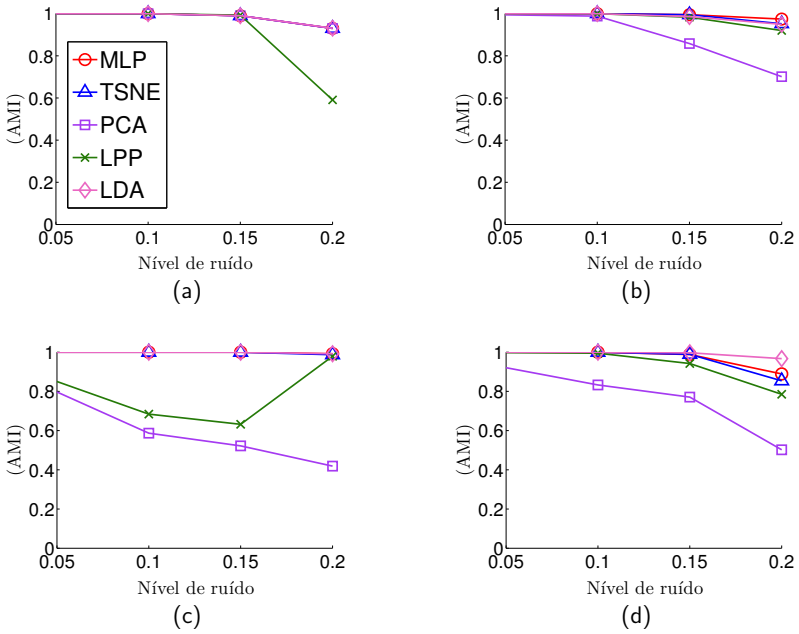


Figura 5.4: Desempenho do algoritmo de clusterização básico (K-means) em função do nível de ruído, para quatro graus de dificuldade (categorias dos conjuntos de dados), indicado através de AMI: (a) "easy1"; (b) "easy2"; (c) "difficult1", e (d) "difficult2". O desempenho se mantém razoavelmente constante à medida em que o nível de ruído aumenta, quando o mapeamento para duas dimensões é feito através dos algoritmos MLP-KLD, t-SNE ou LDA. Os algoritmos PCA e LPP apresentam desempenhos inferiores na maioria dos casos.

ainda se mostra um grande desafio [5].

O experimento realizado na Seção 5.2 mostrou que, em condição de variação do nível de ruído os algoritmos MLP-KLD e t-SNE apresentam, respectivamente, acurácias de classificação iguais a 97,29% e 97,24%.

Outros trabalhos também fizeram uso do "Data Set 2"para avaliar seus métodos de *spike sorting*. Em [74] foi proposto um algoritmo de *spike sorting* baseado na seleção de subconjuntos ótimos de características geométricas e de características obtidas via transformada wavelet discreta e análise de componentes principais. Foram realizados experimentos com registros extracelulares da categoria "difficult2"e níveis de ruído 0,05 e 0,15. As acurácias de classificação obtidas foram de 99,67% e 97,95% respectivamente. Em [37] foi proposto um algoritmo baseado em LPP e clusterização espectral. Foram realizadas diversas comparações e a maior acurácia média de classificação obtida foi de 81,63%. Em [75] foi proposto um algoritmo que utilizou a técnica de mapas de difusão para realizar a etapa de extração de características e o algoritmo *Mean shift* na etapa de clusterização. A acurácia média

[5]Um experimento complementar é apresentado no Apêndice D. Diferente do experimento apresentado na Seção 5.2, que utiliza a transformada wavelet contínua como um extrator de características. O experimento apresentado no Apêndice D utiliza a transformada wavelet discreta [73] com o objetivo de minimizar o ruído presente no registro extracelular.

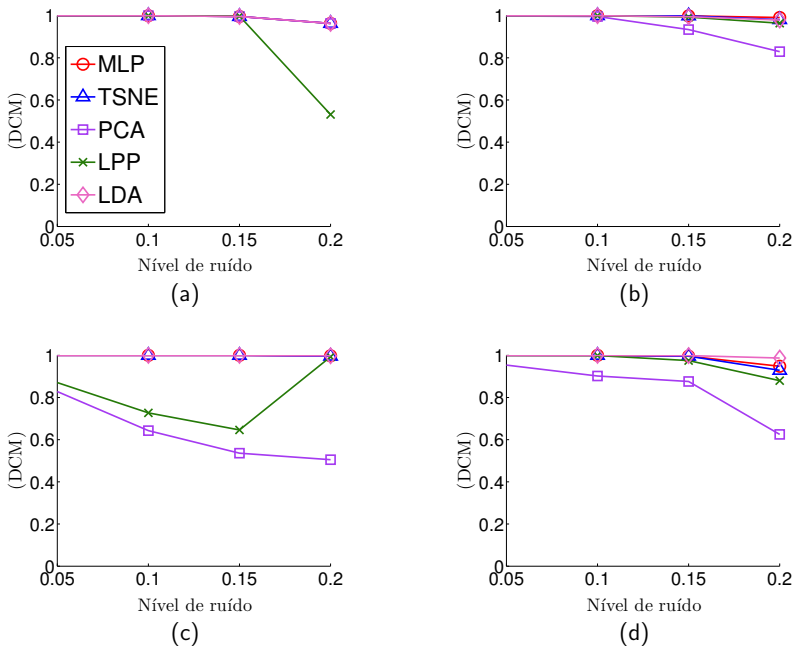


Figura 5.5: Desempenho do algoritmo de clusterização básico (K-means) em função do nível de ruído, para quatro graus de dificuldade, indicado através de DCM: (a) "easy1"; (b) "easy2"; (c) "difficult1", e (d) "difficult2". O desempenho se mantém razoavelmente constante à medida em que o nível de ruído aumenta, quando o mapeamento para duas dimensões é feito através dos algoritmos MLP-KLD, t-SNE ou LDA. Os algoritmos PCA e LPP apresentam desempenhos inferiores na maioria dos casos. As curvas apresentadas concordam com os resultados mostrados na Figura 5.4, o que reforça o potencial de aplicação do DCM como uma alternativa à métrica AMI.

de classificação, se considerada a base de dados completa, foi de 88,09%. Em [32] foi proposto um algoritmo baseado em LDA e clusterização via mistura de gaussianas. O algoritmo realiza buscas iterativas pelo melhor subespaço de mapeamento, de modo que os grupos formados estejam o mais afastados e compactados possível, favorecendo assim a etapa de clusterização. A acurácia média de classificação obtida foi de 97,9%. Em [76] foi proposto um algoritmo baseado em casamento de padrões e discriminante de *Fisher*, que obteve acurácia média de classificação de 97,5%.

5.3 Dimensionalidade do Espaço de Mapeamento

Diferente dos experimentos apresentados nas Seções 5.1 e 5.2, onde o desempenho do método MLP-KLD é avaliado em um espaço de mapeamento 2-D, o objetivo do experimento apresentado nesta seção é avaliar se existem espaços de mapeamento com mais do que duas dimensões, que favoreçam o processo de *spike sorting*.

5.3.1 Formação dos Conjuntos de Dados

O procedimento para a obtenção do registro celular utilizado nesta seção é idêntico ao procedimento descrito no primeiro parágrafo da Seção 5.1.1. Neste estudo do número de dimensões do espaço de mapeamento, somente o conjunto que contém todos os 20 neurônios é utilizado.

O conjunto de dados é então submetido aos algoritmos MLP-KLD, t-SNE, PCA, LPP e LDA, que realizam o mapeamento dos dados de 79 para 500, 400, 300, 200, 100, 50, 30, 20, 10 e 2 dimensões. Em seguida, cada um dos novos conjuntos de dados, já com o novo número de dimensões, é submetido ao algoritmo de clusterização K-means básico.

Foram treinados MLPs contendo apenas uma camada escondida formada por 600 neurônios artificiais do tipo sigmoidal e camada de saída contendo 500, 400, 300, 200, 100, 50, 30, 20, 10 ou 2 neurônios artificiais do tipo linear, referentes ao número de dimensões do espaço de mapeamento. Os parâmetros de treinamento adotados foram: $\alpha = 0.1$, $\beta = 0.1$, $\lambda = 10^{-4}$, $\gamma = 10^{-2}$ e máximo de épocas igual a 3.000, sendo que a cada época 10% das sinapses são mantidas fixas[6].

5.3.2 Avaliação dos Algoritmos

A variação do desempenho do mapeamento, que é estimado através do número de *hits*, número de *misses* e número de clusteres falso-positivos do K-means básico aplicado aos dados mapeados é mostrada na Tabela 5.6. O número de *hits*, *misses* e clusteres falso-positivos obtidos via algoritmos MLP-KLD e t-SNE correspondem a valores médios. Para cada número de dimensões avaliado, os algoritmos MLP-KLD e t-SNE foram executados três vezes e os resultados médios foram incluídos na Tabela 5.6. Esse procedimento é necessário pois, diferente dos algoritmos PCA, LDA e LPP, os algoritmos MLP-KLD e t-SNE são iterativos e podem convergir para resultados diferentes a cada execução.

O número de *hits* propiciado por todos os algoritmos avaliados apresentou um aumentou em casos em que o número de dimensões do espaço de mapeamento é maior que 2. Os algoritmos PCA e LPP permitem apenas mapeamentos para espaços de mapeamento com número de dimensões igual ou inferior ao número de dimensões do conjunto de dados inicial (79 dimensões). O algoritmo LDA permite apenas mapeamentos para espaços de mapeamento com número de dimensões inferior ao número de neurônios biológicos presentes no conjunto de dados. Na Tabela 5.6, as posições sem o valor numérico preenchido correspondem a números de dimensões não permitidos pelos algoritmos PCA, LPP e LDA· Assim como na Seção 5.1, os algoritmos MLP-KLD e t-SNE apresentaram número de *hits* superior ao propiciado

[6]Detalhes sobre a escolha dos hiperparâmetros são descritos no Apêndice C.

pelos algoritmos PCA, LPP e LDA. O baixo número de clusteres falso-positivos obtido pelos mapeamentos MLP-KLD e t-SNE indica que o número de dimensões do espaço de mapeamento exerce pouca influência na formação de sub-clusteres. Para o conjunto de dados utilizado neste experimento, o desempenho médio do MLP-KLD foi inferior ao do t-SNE, porém superior aos demais algoritmos.

Tabela 5.6: Algoritmo K-means básico aplicado a conjuntos de dados obtidos através do mapeamento de 79 para 500, 400, 300, 200, 100, 50, 30, 20, 10 ou 2 dimensões via MLP-KLD, t-SNE, PCA, LDA e LPP. As posições sem valor numérico preenchido correspondem a números de dimensões não permitidos pelos algoritmos avaliados.

	# Dimensões	500	400	300	200	100	50	30	20	10	2
MLP-KLD	*Hits*	18	19,67	18,67	18,33	19	19,33	19,33	19,67	19,67	17,33
	Misses	0,3	0,3	0	0	0	0	0	0	0	1
	Falso Positivos	1,7	0	1,3	1,7	1	0,7	0,7	0,3	0,3	1,7
t-SNE	*Hits*	20	20	20	20	20	19,67	20	20	20	19,33
	Misses	0	0	0	0	0	0	0	0	0	0
	Falso Positivos	0	0	0	0	0	0,33	0	0	0	0,67
PCA	*Hits*	-	-	-	-	-	17	17	17	16	11
	Misses	-	-	-	-	-	0	1	0	1	6
	Falso Positivos	-	-	-	-	-	3	2	3	3	3
LPP	*Hits*	-	-	-	-	-	17	19	19	19	10
	Misses	-	-	-	-	-	0	0	0	0	5
	Falso Positivos	-	-	-	-	-	3	1	1	1	5
LDA	*Hits*	-	-	-	-	-	-	-	-	18	13
	Misses	-	-	-	-	-	-	-	-	0	2
	Falso Positivos	-	-	-	-	-	-	-	-	2	5

Na Tabela 5.7 são mostrados valores dos números de erros e da acurácia de classificação. No caso dos algoritmos MLP-KLD e t-SNE o número de erros e da acurácia correspondem a valores médios. O mapeamento baseado em MLP-KLD levou a resultados de classificação claramente superiores aos dos algoritmos PCA, LPP e LDA, porém inferiores aos do algoritmo t-SNE. As informações consideradas mais relevantes, fornecidas pela Tabela 5.7, são: a) o desempenho do algoritmo MLP-KLD sofre pouca variação com o aumento do número de dimensões do espaço de mapeamento, e b) todos os algoritmos comparados apresentam um aumento no desempenho, em termos de número de erros e acurácia de classificação, quando o número de dimensões do espaço de mapeamento é superior a 2-D.

Tabela 5.7: Número de erros e acurácia alcançados pelo algoritmo K-means básico executado sobre dados obtidos através do mapeamento de 79 para 500, 400, 300, 200, 100, 50, 30, 20, 10 ou 2 dimensões via MLP-KLD, t-SNE, PCA, LDA e LPP. Os valores de acurácia são mostrados entre parênteses. Em negrito estão marcados os melhores resultados.

		LDA	LPP	PCA	t-SNE	MLP-KLD
# Dimensões	500	-	-	-	**247,67 (98)**	1.631 (86,90)
	400	-	-	-	**247 (98,01)**	864,67 (93,40)
	300	-	-	-	**247,67 (98)**	1.127,67 (90,93)
	200	-	-	-	**246,33 (98,02)**	1.448 (88,35)
	100	-	-	-	**246,67 (98,01)**	1.244,67 (89,99)
	50	-	2.949 (76,28)	2.975 (76,07)	**537,67 (95,67)**	909,33(92,69)
	30	-	840 (93,24)	2.741 (77,95)	**246 (98,02)**	795,33 (93,60)
	20	-	860 (93,08)	3.006 (75,82)	**248 (98)**	297 (97,61)
	10	1.244 (89,99)	1.194 (90,39)	3.953 (68,20)	**244 (98,04)**	511,67 (95,88)
	2	6.314 (49,21)	8.457 (31,97)	7.186 (42,19)	**898 (92,78)**	2.057 (83,45)

5.3.3 Conclusões

Tipicamente os trabalhos de *spike sorting* realizam mapeamento para um espaço de mapeamento contendo duas dimensões. O experimento realizado na Seção 5.3 sugere a existência de espaços de mapeamento com mais do que duas dimensões, que favorecem o *spike sorting*. Como consequência direta do experimento é possível supor que métodos de *spike sorting*, que tenham como propósito a detecção automática do número de clusteres, podem apresentar uma melhora em seus desempenhos se aplicados a dados mapeados em espaços com mais do que duas dimensões.

5.4 Número de Camadas

As características extraídas por uma rede neural tornam-se mais complexas à medida em que são obtidas de camadas mais profundas da rede. O experimento aqui proposto tem como objetivo verificar se, para problemas de *spike sorting*, o aumento da complexidade das características extraídas favorece a tarefa de *spike sorting*. É verificado se o desempenho do clusterizador, implementado via algoritmo K-means básico, melhora quando aplicado a mapeamentos obtidos a partir de MLPs contendo duas ou mais camadas escondidas.

Este experimento utiliza o mesmo conjunto de dados aplicado no experimento

da Seção 5.3. São treinados MLPs contendo de uma até sete camadas escondidas de neurônios artificiais do tipo sigmoidal. Todos os MLPs treinados possuem apenas dois neurônios artificiais do tipo linear na camada de saída, o que significa que o mapeamento realizado é de 79 dimensões para duas dimensões.

5.4.1 Número de Neurônios Artificiais das Camadas Escondidas

Não existe um critério bem definido que determine o número ideal de camadas, bem como o número de neurônios artificiais em cada camada. Para determinar as topologias a serem treinadas, o experimento apresentado na Seção 5.3 foi tomado como referência. Naquele experimento foram treinados MLP-KLDs com apenas uma camada escondida contendo 600 neurônios artificiais, e camada de saída contendo 500, 400, 300, 200, 100, 50, 30, 20, 10 ou 2 neurônios artificiais. Então, para o experimento realizado na Seção 5.4 optou-se pelo treinamento de topologias contendo 600, 200, 100, 50, 30, 20 ou 10 neurônios artificiais dispostos ao longo das camadas escondidas.

A Figura 5.6 mostra os resultados de alguns mapeamentos obtidos a partir de MLPs contendo de uma até sete camadas escondidas. Os parâmetros de treinamento adotados foram: $\alpha = 0.1$, $\beta = 0.1$, $\lambda = 10^{-4}$, $\gamma = 10^{-2}$ e máximo de épocas igual a 3.000, sendo que a cada época 10% das sinapses são mantidas fixas [7].

Tabela 5.8: Topologias utilizadas para o treinamento de MLPs via algoritmo MLP-KLD. Todos os MLPs possuem 79 entradas, 600 neurônios artificiais na primeira camada escondida e dois neurônios artificiais na camada de saída. As demais camadas escondidas possuem um número decrescente de neurônios.

# Camadas escondidas	Topologias
1	79-600-2
2	79-600-10-2
3	79-600-20-10-2
4	79-600-30-20-10-2
5	79-600-50-30-20-10-2
6	79-600-100-50-30-20-10-2
7	79-600-200-100-50-30-20-10-2

[7] Detalhes sobre a escolha dos hiperparâmetros são descritos no Apêndice C.

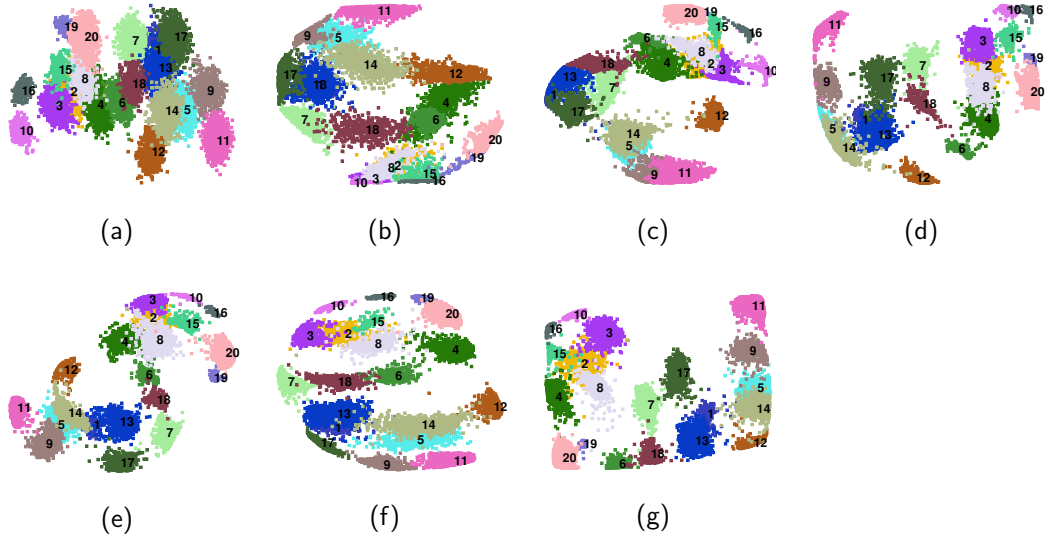


Figura 5.6: Comparação visual entre mapeamentos de 79 para duas dimensões obtidos através MLPs contendo de uma até sete camadas escondidas. MLPs contendo (a) uma camada escondida, (b) duas camadas escondidas até (g) sete camadas escondidas. Percebe-se visualmente que os mapeamentos foram capazes de revelar os agrupamentos implícitos no conjunto de dados. Para colorir estas classes nesta figura, as etiquetas originais foram utilizadas.

5.4.2 Avaliação das Topologias

A Tabela 5.8 mostra todas as topologias utilizadas para o treinamento dos MLPs. Para cada topologia, são treinados três MLPs e os resultados médios são apresentados nas Tabelas 5.9 e 5.10. A Tabela 5.9 mostra os valores médios do número de *hits*, *misses* e clusteres falso-positivos em função do número de camadas escondidas dos MLPs treinados via algoritmo MLP-KLD. Com exceção da topologia contendo uma camada escondida, o aumento do número de camadas escondidas proporcionou aumento de desempenho em termos de número de *hits*. O número de *misses* e clusteres falso-positivos apresentaram redução com o aumento do número de camadas.

Tabela 5.9: Algoritmo K-means básico aplicado a um conjunto de dados 2-D obtido através do mapeamento de 79 para duas dimensões via MLPs contendo de uma até sete camadas escondidas. Os resultados mostrados correspondem a valores médios do número de *hits*, *misses* e clusteres falso-positivos.

# Camadas escondidas	1	2	3	4	5	6	7
Hits	18	11,33	18	18,67	18,67	19,5	19,5
Misses	0	3,67	0,33	0	0	0	0
Falso Positivos	2	5	1,67	1,33	1,33	0,5	0,5

Na Tabela 5.10, são mostrados valores médios dos números de erros e das acurácias de classificação para MLPs contendo de uma até sete camadas escondidas. A topologia contendo sete camadas escondidas apresentou desempenho superior a todas as topologias com menor número de camadas escondidas. Em média os MLPs contendo duas camadas escondidas apresentaram desempenho inferior aos MLPs contendo uma camada escondida. Os resultados obtidos mostram que o aumento do número de camadas é uma estratégia que pode ser adotada com o objetivo de melhorar o desempenho médio dos MLPs treinados via algoritmo MLP-KLD.

Tabela 5.10: Número de erros e acurácia alcançados pelo algoritmo K-means básico executado sobre dados 2-D gerados através de MLPs contendo de uma até sete camadas escondidas. Os valores indicados correspondem à médias dos resultados obtidos a partir de três MLPs contendo igual número de camadas escondidas. As acurácias são mostradas entre parênteses.

# Camadas escondidas	1	2	3	4	5	6	7
Valor médio	1.738(86)	6.477,67(47,90)	1.967,67(84,20)	1.747,33(85,90)	1.034(91,7)	772(94)	639(95)

5.4.3 Conclusões

O experimento realizado mostrou que, para problemas de *spike sorting*, é possível treinar, via algoritmo MLP-KLD, MLPs contendo um grande número de camadas escondidas. Não foi possível afirmar que o aumento do número de camadas implica necessariamente no aumento da acurácia. Foram encontradas evidências de que é possível obter topologias contendo um grande número de camadas escondidas que favoreçam o processo de *spike sorting*.

5.5 Cálculos Alternativos de Distância

Conforme apresentado no Capítulo 4, o algoritmo MLP-KLD minimiza uma função custo $J(\theta)$ que corresponde à forma regularizada da KLD entre a p.m.f. (*probability mass function*) $\mathbf{p}$, associável às distâncias Euclideanas entre os pontos $\mathbf{x}_i$ e $\mathbf{x}_j$ no espaço original $\mathbb{R}^D$, e a p.m.f. $\mathbf{q}$, associável às distâncias Euclideanas entre os pontos $\hat{\mathbf{y}}_i$ e $\hat{\mathbf{y}}_j$ no espaço de mapeamento $\mathbb{R}^L$. O objetivo do experimento apresentado nesta seção é verificar a influência da métrica de distância no desempenho dos MLPs treinados via algoritmo MLP-KLD. Na comparação, são considerados os cálculos de distância Euclideana, "Cosseno"e baseada em norma unitária ("*City Block*").

5.5.1 Formação dos Conjuntos de Dados

Para este experimento foram selecionados cinco registros extracelulares obtidos a partir da base de dados "Data Set 1". Os registros selecionados contêm *spikes* gerados por 20 neurônios distintos. As etapas de detecção, segmentação e construção dos conjuntos de dados foram as mesmas adotadas nos experimentos descritos nas Seções 5.1 e 5.3. Os cinco registros extracelulares deram origem a cinco conjuntos de dados.

Os cinco conjuntos de dados são submetidos ao MLP-KLD, que realiza o mapeamento de 79 dimensões para duas dimensões, seguido de clusterização pelo algoritmo K-means básico. A topologia escolhida foi a mesma utilizada para o experimento apresentado na Seção 5.1, ou seja, MLPs contendo apenas uma camada escondida composta por 600 neurônios artificiais do tipo sigmoidal, e dois neurônios artificiais do tipo linear na camada de saída. Foram utilizados os mesmos parâmetros de treinamento adotados na Seção 5.1.2.

Tabela 5.11: Algoritmo K-means básico aplicado a conjuntos de dados 2-D obtidos através do mapeamento de 79 para duas dimensões via MLP-KLD com métricas de distância "Cosseno", "*City Block*"ou "Euclideana". Os resultados mostrados correspondem a valores médios do número de *hits*, *misses* e clusteres falso-positivos. Em negrito estão marcados os melhores resultados.

# Rodadas		1ª	2ª	3ª
Cosseno	*Hits*	14,8	15	14,2
	Misses	1,8	2,2	1,6
	Falso Positivos	3,4	2,8	4,2
City Block	*Hits*	**17,2**	**17,8**	**17,4**
	Misses	0,6	0,4	0,4
	Falso Positivos	2,2	1,8	2,2
Euclideana	*Hits*	16,4	16	16,2
	Misses	0,8	0,4	0,4
	Falso Positivos	2,8	3,6	3,4

5.5.2 Avaliação dos Algoritmos

Na Tabela 5.11, são mostrados valores médios do número de *hits*, *misses* e clusteres falso-positivos. Por exemplo, para a coluna "1ª"e métrica de distância "Cosseno", os números médios de *hits*, *misses* e clusteres falso-positivos, são obtidos a partir dos mapeamentos realizados por cinco MLPs distintos, treinados via algoritmo MLP-KLD com métrica de distância "Cosseno"(cada MLP é treinado a partir de um conjunto de dados diferente). Cada "Rodada"corresponde ao trei-

Tabela 5.12: Valores médios do número de erros e da acurácia calculados a partir de mapeamentos de 79 para duas dimensões obtidos via algoritmo MLP-KLD com diferentes métricas de distância. Em negrito estão marcados os melhores resultados.

# Rodadas	Cosseno	*City Block*	Euclideana
1ª	3.676,2 (67,42)	**2.113,8 (81,26)**	2.787,8 (75,38)
2ª	3.415,6 (69,76)	**1.867,6 (83,57)**	3.066,6 (72,77)
3ª	3.772 (66,75)	**2.174 (80,89)**	3.022 (73,29)
Valor Médio	3.621,26 (67,97)	**2.051,8 (81,90)**	2.958,8 (73,81)

namento de cinco MLPs. O algoritmo MLP-KLD pode convergir para resultados diferentes a cada execução. Os resultados das diferentes "Rodadas"para diferentes métricas de distância mostram que MLPs treinados com métrica de distância "*City Block*"apresentam desempenho superior, em termos de número de *hits*, *misses* e clusteres falso-positivos, quando comparados a MLPs treinados via algoritmo MLP-KLD baseado nas métricas de distância "Cosseno"ou "Euclideana".

Na Tabela 5.12, são mostrados valores médios do número de erros e da acurácia de classificação. Os mapeamentos baseados em MLP-KLD com métrica de distância "*City Block*"levam a melhores resultados de classificação nas três rodadas realizadas. Os erros obtidos com o uso da métrica de distância"*City Block*"se mantêm menores que os erros obtidos com as métricas de distância "Cosseno"ou "Euclideana". A última linha da Tabela 5.12 apresenta os valores médios globais.

5.5.3 Conclusões

O experimento apresentado nesta seção indicou que a métrica de distância utilizada para o cálculo da p.m.f pode influenciar no desempenho dos MLPs treinados via algoritmo MLP-KLD. O experimento realizado indicou, que para os conjuntos de dados utilizados, o algoritmo MLP-KLD com métrica de distância "*City Block*"favoreceu o treinamento de MLPs com desempenho superior em termos de *hits*, *misses*, clusteres falso-positivos e acurácia de classificação, se comparados a MLPs treinados via algoritmo MLP-KLD com métrica de distância "Euclideana".

Capítulo 6

Conclusões

O objetivo desta tese foi propor um método, para o mapeamento de dados (*spikes*) de registros neurais extracelulares, capaz de revelar os agrupamentos implícitos no conjunto original de *spikes*. O método proposto é baseado em perceptrons multicamadas (MLP). O treinamento dos MLPs utiliza uma função custo composta por dois termos: o primeiro termo corresponde à divergência de Kullback-Leibler (KLD) e segundo termo corresponde à regularização aplicada à norma $\mathcal{L}_2$ do vetor de parâmetros θ a serem ajustados. Em adição à regularização $\mathcal{L}_2$, o método proposto também inclui uma variante do método Dropout. Esta variante consiste em, a cada época, manter fixo um subconjunto de elementos do vetor de parâmetros θ. O ajuste dos parâmetros é realizado via algoritmo gradiente descendente com momento, e uma regra de limitação é aplicada ao incremento $\Delta\theta_k^{(t)}$ calculado para o parâmetro θ_k do vetor θ a cada época. Para validar o método proposto um conjunto de experimentos foi realizado e apresentado ao longo do Capítulo 5. Os experimentos apresentados nas Seções 5.1 e 5.2 são considerados os principais.

Com o avanço na tecnologia dos sistemas de aquisição de registros neurais extracelulares é esperado a presença de um número cada vez maior de neurônios presentes no registro neural. Na Seção 5.1 o algoritmo MLP-KLD foi avaliado em condição de aumento de número de neurônios presentes nos registros extracelulares. Para isso foi utilizada uma base de dados simulados, "Data Set 1", composta por 95 registros extracelulares. Cada registro deu origem a um conjunto de dados formados pelos *spikes* extraídos do respectivo registro extracelular. Após a etapa de normalização dos *spikes* contidos em cada registro, os conjuntos de dados foram submetidos aos algoritmos MLP-KLD, t-SNE, PCA, LDA e LPP, que realizaram o mapeamento de 79 para duas dimensões. Cada conjunto de dados, já em duas dimensões, foi submetido à clusterização via algoritmo K-means básico. As métricas de desempenho calculadas mostraram a manutenção do desempenho dos algoritmos MLP-KLD e t-SNE, enquanto os algoritmos PCA, LDA e LPP apresentaram redução de desempenho dado o aumento do número de neurônios.

Na Seção 5.2 o algoritmo MLP-KLD foi avaliado em uma base de dados simulados, "Data Set 2", cujos registros extracelulares apresentam diferentes níveis de ruído. Inicialmente, para este experimento, cada *spike* foi extraído com 64 dimensões e convoluído com uma wavelet do tipo Symlet, dando origem a um vetor de 64 coeficientes. O vetor de coeficientes foi utilizado como a nova representação do *spike*. Os conjuntos de dados foram formados pelos vetores de coeficientes. Cada conjunto de dados foi normalizado e submetido aos algoritmos MLP-KLD, t-SNE, PCA, LDA e LPP, que realizaram o mapeamento de 64 para duas dimensões. As métricas de desempenho, calculadas a partir do resultado da clusterização obtida sobre os dados 2-D, mostraram que, para a maioria dos conjuntos de dados, o desempenho do algoritmo MLP-KLD foi superior ou igual aos obtidos pelos algoritmos t-SNE e LDA. Os algoritmos PCA e LPP apresentaram resultados inferiores.

Para complementar o experimento apresentado na Seção 5.1, os algoritmos de *spike sorting* foram avaliados a partir de conjuntos de dados formados por vetores de coeficientes obtidos via convolução dos sinais de *spike* com uma wavelet do tipo Symlet. Os resultados obtidos foram equivalentes aos apresentados nas Tabelas 5.2 e 5.3. De forma similar o experimento apresentado na Seção 5.2 também foi realizado a partir do domínio de entradas original, ou seja, sem convolução com wavelet. Neste caso, os resultados obtidos foram inferiores aos apresentados na Tabela 5.5, embora a ordenação dos desempenhos tenha sido igual: LDA (96,6%),MLP-KLD (89,7%), t-SNE (89,1%), PCA (75,2%) e LPP (69,9%).

Tradicionalmente os métodos de *spike sorting* incluem a passagem pelo espaço de mapeamento 2-D. No entanto, o experimento apresentado na Seção 5.3 avaliou os algoritmos de spike sorting considerando espaços de mapeamento com mais do que duas dimensões. Para este experimento foi selecionado um registro extracelular contendo 20 neurônios biológicos. O registro foi extraído do "Data Set 1". As etapas de normalização e formação do conjunto de dados foram as mesmas utilizadas na Seção 5.1. Os mapeamentos realizados pelos algoritmos MLP-KLD, t-SNE, PCA, LPP e LDA, foram de 79 para 500, 400, 300, 200, 100, 50, 30, 20, 10 e 2 dimensões. Após o mapeamento, os dados foram clusterizados via K-means básico e as métricas de desempenho foram calculadas. Para o registro neural utilizado, os resultados mostraram a existência de espaços de mapeamento com mais do que duas dimensões, que favorecem o *spike sorting*.

O experimento apresentado na Seção 5.4 avaliou o desempenho do clusterizador, implementado pelo algoritmo K-means básico, quando aplicado a mapeamentos obtidos via MLPs contendo de duas até sete camadas escondidas. Neste experimento utilizou-se o mesmo conjunto de dados aplicado no experimento da Seção 5.3. Os MLPs treinados apresentavam camadas escondidas contendo 600, 200, 100, 50, 30, 20 ou 10 neurônios artificiais. Os mapeamentos foram de 79 para duas dimensões.

Para o conjunto de dados utilizado, o resultado indicou que é possível treinar MLPs contendo até sete camadas escondidas, que favoreçam o *spike sorting*. Contudo o experimento também indicou que o aumento do número de camadas não é uma condição suficiente para o aumento no desempenho do clusterizador.

A Seção 5.5 avaliou o desempenho de MLPs treinados via algoritmo MLP-KLD, com métricas de distância "*City Block*"ou "Cosseno". O experimento utilizou registros extracelulares obtidos da base de dados "Data Set 1". Todos os MLPs treinados possuem topologia 79-600-2, ou seja, uma camada escondida contendo 600 neurônios artificiais e camada de saída contendo dois neurônios artificiais. Os mapeamentos são de 79 para duas dimensões. Para o conjunto de dados utilizado neste experimento, os resultados indicam que diferente do que foi inicialmente proposto nesta tese (MLP-KLD com métrica Euclideana), a métrica *City Block* leva a MLPs com desempenho superiores em termos de número de *hits*, *misses*, clusteres falso-positivos e acurácia de classificação.

Por fim, o Apêndice D apresentou um experimento que avaliou se através da filtragem, via transformada wavelet discreta, dos registros extracelulares contidos no "Data Set 2"era possível obter um aumento no desempenho dos MLPs treinados via algoritmo MLP-KLD. Os resultados obtidos mostraram que a filtragem via transformada wavelet reduziu o desempenho dos MLPs. Uma possível interpretação para os resultados é que o processo de filtragem pode ter eliminado informação utilizada pela rede neural para aprender as características que permitem uma melhor discriminação entre os diferentes tipos de neurônios biológicos presentes nos registros extracelulares. O Apêndice E apresentou um experimento que avaliou o tempo computacional médio de uma época de iteração dos algoritmos MLP-KLD e t-SNE em diferentes condições de experimento. Em todos os casos em que o tempo computacional dos algoritmos MLP-KLD e t-SNE foi comparado, o algoritmo MLP-KLD apresentou tempo computacional médio superior ao do algoritmo t-SNE.

Na sua maioria, os trabalhos encontrados na literatura apresentam resultados obtidos a partir de registros extracelulares contendo até três neurônios distintos. Os resultados apresentados nesta tese consideraram registros neurais contendo até 20 neurônios. Os resultados obtidos a partir de mapeamentos via MLPs concordam com o conhecimento geral de que modelos não-lineares são adequados para o aprendizado da superfície intrínseca ao longo da qual os dados se distribuem. Os resultados comparativos apresentados nas Seções 5.1 e 5.2 mostraram o potencial de contribuição do algoritmo proposto quando aplicado a registros extracelulares contendo até 20 neurônios ou nível de ruído não desprezível. Já os resultados apresentados nas Seções 5.3, 5.4 e 5.5 reforçam o potencial de contribuição do algoritmo MLP-KLD ao mostrarem que é possível melhorar o desempenho dos MLPs através da escolha do número de dimensões do espaço de mapeamento, do número de camadas

escondidas ou da métrica de distância utilizada para o cálculo da p.m.f.

6.1 Trabalhos Futuros

Existem diversos caminhos que podem ser desenvolvidos como pesquisas futuras, como por exemplo: investigar o desempenho dos algoritmos de *spike sorting* quando aplicados a casos de sobreposição de *spikes*; estudar a influência da técnica de extração de características ou da função de divergência adota sobre o desempenho do classificador. Devido ao crescente número de trabalhos propondo novas arquiteturas, topologias e algoritmos de treinamento de redes neurais [77–79], também seria interessante investigar o potencial de contribuição dessas novas técnicas se aplicadas à tarefa de *spike sorting*.

Referências Bibliográficas

[1] WU, J., HU, H. "Cascade recurrent neural network for image caption generation", *Electronics Letters*, v. 53, n. 25, pp. 1642–1643, 2017. ISSN: 0013-5194.

[2] SHAH, P., BAKROLA, V., PATI, S. "Image captioning using deep neural architectures". In: *2017 International Conference on Innovations in Information, Embedded and Communication Systems*, pp. 1–4, March 2017.

[3] ROSSETTO, A. M., ZHOU, W. "Deep learning for categorization of lung cancer CT images". In: *2017 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies*, pp. 272–273, July 2017.

[4] ARIK, A., GÖLCÜK, M., KARSLIGIL, E. M. "Deep learning based skin cancer diagnosis". In: *2017 25th Signal Processing and Communications Applications Conference*, pp. 1–4, May 2017.

[5] VAN DER MAATEN, L. "Learning a parametric embedding by preserving local structure." In: Dyk, D. A. V., Welling, M. (Eds.), *AISTATS*, v. 5, *JMLR Proceedings*, pp. 384–391. JMLR.org, 2009.

[6] XIE, J., GIRSHICK, R., FARHADI, A. "Unsupervised deep embedding for clustering analysis". In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 478–487. JMLR.org, 2016.

[7] BIGI, B. "Using kullback-leibler distance for text categorization". In: *Proceedings of the 25th European Conference on IR Research*, ECIR'03, pp. 305–319, Berlin, Heidelberg, 2003. Springer-Verlag.

[8] KHODAYARI-ROSTAMABAD, A., REILLY, J. P., HASEY, G. M. "Latent variable dimensionality reduction using a kullback-leibler criterion and its application to predict antidepressant treatment response". In: *2013 International Workshop on Pattern Recognition in Neuroimaging*, pp. 148–151, June 2013.

[9] ABUAIADAH, D., RAJENDRAN, D., JARRAR, M. "Clustering arabic tweets for sentiment analysis". In: *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, pp. 449–456, Oct 2017.

[10] AMIRI, A., SEYEDIN, S., AHADI, S. M. "Multi-layer kullback-leibler-based complex NMF with LPC error clustering for blind source separation". In: *2017 7th International Conference on Computer and Knowledge Engineering (ICCKE)*, pp. 272–276, Oct 2017.

[11] KHAN, M. E., BAQUE, P., FLEURET, F., et al. "Kullback-leibler proximal variational inference". In: Cortes, C., Lawrence, N. D., Lee, D. D., et al. (Eds.), *Advances in Neural Information Processing Systems 28*, Curran Associates, Inc., pp. 3402–3410, 2015.

[12] HANG, Y., DERONG, C., JIULU, G. "Object tracking using both a kernel and a non-parametric active contour model", *Neurocomputing*, v. 295, pp. 108 – 117, 2018. ISSN: 0925-2312.

[13] ZHANG, X., NG, M. K., BAI, M. "A fast algorithm for deconvolution and poisson noise removal", *Journal of Scientific Computing*, v. 75, n. 3, pp. 1535–1554, Jun 2018. ISSN: 1573-7691.

[14] LEWICKI, M. S. "A review of methods for spike sorting: the detection and classification of neural action potentials", *Network: Computation in Neural Systems*, v. 9, n. 4, pp. 53–78, 1998.

[15] KIPNIS, N. "Luigi Galvani and the debate on animal electricity, 1791–1800", *Annals of Science*, v. 44, n. 2, pp. 107–142, 1987.

[16] HODGKIN, A., HUXLEY, A. "A quantitative description of membrane current and its application to conduction and excitation in nerve", *Bulletin of Mathematical Biology*, v. 52, n. 1, pp. 25 – 71, 1990. ISSN: 0092-8240.

[17] HUBEL, D. H., WIESEL, T. N. "Ferrier lecture. Functional architecture of macaque monkey visual cortex." *Proceedings of the Royal Society of London B: Biological Sciences*, v. 198, n. 1130, pp. 1–59, 1977. ISSN: 0080-4649. doi: 10.1098/rspb.1977.0085.

[18] GERSTEIN, G. L., CLARK, W. A. "Simultaneous studies of firing patterns in several neurons", *Science*, v. 143, n. 3612, pp. 1325–1327, 1964. ISSN: 0036-8075.

[19] HILGEN, G., SORBARO, M., PIRMORADIAN, S., et al. "Unsupervised spike sorting for large-scale, high-density multielectrode arrays", *Cell Reports*, v. 18, n. 10, pp. 2521 – 2532, 2017. ISSN: 2211-1247.

[20] GOLD, C., HENZE, D. A., KOCH, C., et al. "On the Origin of the Extracellular Action Potential Waveform: A Modeling Study", *Journal of Neurophysiology*, v. 95, n. 5, pp. 3113–3128, 2006. ISSN: 0022-3077.

[21] CAMUÑAS-MESA, L. A., QUIROGA, R. Q. "A detailed and fast model of extracellular recordings", *Neural Computation*, v. 25, n. 5, pp. 1191–1212, 2013.

[22] WANG, M., QU, Q., HE, T., et al. "Distinct temporal spike and local field potential activities in the thalamic parafascicular nucleus of Parkinsonian rats during rest and limb movement", *Neuroscience*, v. 330, n. Supplement C, pp. 57 – 71, 2016. ISSN: 0306-4522.

[23] PEDREIRA, C., MARTINEZ, J., ISON, M. J., et al. "How many neurons can we see with current spike sorting algorithms?" *Journal of Neuroscience Methods*, v. 211, n. 1, pp. 58–65, 2012. ISSN: 0165-0270.

[24] QUIROGA, R. Q., PANZERI, S. "Extracting information from neuronal populations: information theory and decoding approaches." *Nature reviews. Neuroscience*, v. 10, n. 3, pp. 173–185, 2009. ISSN: 1471-0048.

[25] BORST, A., THEUNISSEN, F. E. "Information theory and neural coding." *Nature neuroscience*, v. 2, n. 11, pp. 947–957, 1999. ISSN: 1097-6256.

[26] COLLINGER, J. L., WODLINGER, B., DOWNEY, J. E., et al. "High-performance neuroprosthetic control by an individual with tetraplegia", *The Lancet*, v. 381, n. 9866, pp. 557–564, 2013. ISSN: 01406736.

[27] DEADWYLER, S. A., HAMPSON, R. E., SONG, D., et al. "A cognitive prosthesis for memory facilitation by closed-loop functional ensemble stimulation of hippocampal neurons in primate brain", *Experimental Neurology*, v. 287, n. Part 4, pp. 452 – 460, 2017. ISSN: 0014-4886.

[28] MELONI, P., RUBATTU, C., TUVERI, G., et al. "A Custom dual-processor System for Real-time Neural Signal Processing", *IFAC-PapersOnLine*, v. 49, n. 25, pp. 61 – 67, 2016. ISSN: 2405-8963.

[29] REY, H. G., PEDREIRA, C., QUIROGA, R. Q. "Past, present and future of spike sorting techniques", *Brain Research Bulletin*, v. 119, pp. 106–117, 2015. ISSN: 1873-2747.

[30] NENADIC, Z., BURDICK, J. W. "Spike detection using the continuous wavelet transform", *IEEE Transactions on Biomedical Engineering*, v. 52, n. 1, pp. 74–87, Jan, 2005. ISSN: 0018-9294.

[31] ADAMOS, D. A., KOSMIDIS, E. K., THEOPHILIDIS, G. "Performance evaluation of PCA-based spike sorting algorithms", *Computer Methods and Programs in Biomedicine*, v. 91, n. 3, pp. 232–244, 2008. ISSN: 0169-2607.

[32] KESHTKARAN, M. R., YANG, Z. "Noise-robust unsupervised spike sorting based on discriminative subspace learning with outlier handling", *Journal of Neural Engineering*, v. 14, n. 3, pp. 036003, 2017.

[33] QUIROGA, R. Q. "Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering", *Neural Computation*, v. 1687, pp. 1661–1687, 2004.

[34] ZHANG, P.-M., WU, J.-Y., ZHOU, Y., et al. "Spike sorting based on automatic template reconstruction with a partial solution to the overlapping problem", *Neuroscience Methods*, v. 135, pp. 55–65, 2004.

[35] WOOD, F., BLACK, M. J. "A nonparametric Bayesian alternative to spike sorting", *Journal of Neuroscience Methods*, v. 173, n. 1, pp. 1–12, 2008. ISSN: 0165-0270.

[36] YANG, C., YUAN, Y., SI, J. "Robust spike classification based on frequency domain neural waveform features", *Journal of Neural Engineering*, v. 10, n. 6, pp. 066015, 2013.

[37] NGUYEN, T., KHOSRAVI, A., CREIGHTON, D., et al. "Spike sorting using locality preserving projection with gap statistics and landmark-based spectral clustering", *Journal of Neuroscience Methods*, v. 238, n. 6, pp. 43 – 53, 2014. ISSN: 0165-0270.

[38] KIM, K. H., KIM, S. J. "Neural spike sorting under nearly 0-dB signal-to-noise ratio using nonlinear energy operator and artificial neural-network classifier", *IEEE Transactions on Biomedical Engineering*, v. 47, n. 10, pp. 1406–1411, Oct, 2000. ISSN: 0018-9294.

[39] LEIBIG, C., WACHTLER, T., ZECK, G. "Unsupervised neural spike sorting for high-density microelectrode arrays with convolutive independent component analysis", *Journal of Neuroscience Methods*, v. 271, pp. 1–13, 2016. ISSN: 0165-0270.

[40] BUZSÁKI, G. "Large-scale recording of neuronal ensembles", *Nature neuroscience*, v. 7, pp. 446–451, Apr, 2004.

[41] SHOHAM, S., O'CONNOR, D. H., SEGEV, R. "How silent is the brain: is there a "dark matter" problem in neuroscience?" *Journal of Comparative Physiology A*, v. 192, n. 8, pp. 777–784, Aug, 2006. ISSN: 1432-1351.

[42] BENGIO, Y., COURVILLE, A., VINCENT, P. "Representation learning: A review and new perspectives", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 35, n. 8, pp. 1798–1828, Aug, 2013. ISSN: 0162-8828.

[43] KHURANA, U., TURAGA, D., SAMULOWITZ, H., et al. "Cognito: Automated feature engineering for supervised learning". In: *2016 IEEE 16th International Conference on Data Mining Workshops*, pp. 1304–1307, Dec, 2016.

[44] HEATON, J. "An empirical analysis of feature engineering for predictive modeling". In: *SoutheastCon 2016*, pp. 1–6, Mar, 2016.

[45] UMER, S., DHARA, B. C., CHANDA, B. "A novel cancelable iris recognition system based on feature learning techniques", *Information Sciences*, v. 406-407, pp. 102–118, 2017. ISSN: 0020-0255.

[46] FANG, C., ZHAO, Z., ZHOU, P., et al. "Feature learning via partial differential equation with applications to face recognition", *Pattern Recognition*, v. 69, pp. 14–25, 2017. ISSN: 0031-3203.

[47] GULTEPE, E., CONTURO, T. E., MAKREHCHI, M. "Predicting and grouping digitized paintings by style using unsupervised feature learning", *Journal of Cultural Heritage*, 2017. ISSN: 1296-2074.

[48] SU, S., GE, H., TONG, Y. "Multi-graph embedding discriminative correlation feature learning for image recognition", *Signal Processing: Image Communication*, v. 60, pp. 173–182, 2018. ISSN: 0923-5965.

[49] HAYKIN, S. S. "Neural networks and learning machines". Third ed., Pearson Education, 2009.

[50] RUDER, S. "An overview of gradient descent optimization algorithms", *CoRR*, v. abs/1609.04747, 2016.

[51] HE, X., NIYOGI, P. "Locality Preserving Projections". In: Thrun, S., Saul, L. K., Schölkopf, B. (Eds.), *Advances in Neural Information Processing Systems 16*, MIT Press, pp. 153–160, 2004.

[52] MAATEN, L. V. D., HINTON, G. "Visualizing data using t-SNE", *Journal of Machine Learning Research*, v. 9, pp. 2579–2605, 2008. ISSN: 0254-5330.

[53] SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., et al. "Dropout: A simple way to prevent neural networks from overfitting", *Journal of Machine Learning Research*, v. 15, pp. 1929–1958, 2014.

[54] LLOYD, S. "Least squares quantization in PCM", *IEEE Transactions on Information Theory*, v. 28, n. 2, pp. 129–137, Mar, 1982. ISSN: 0018-9448.

[55] REYNOLDS, D. "Gaussian mixture models". In: Li, S. Z., Jain, A. (Eds.), *Encyclopedia of Biometrics*, pp. 659–663, Boston, MA, Springer US, 2009. ISBN: 978-0-387-73003-5.

[56] ZOU, K., WANG, Z., HU, M. "An new initialization method for fuzzy c-means algorithm", *Fuzzy Optimization and Decision Making*, v. 7, n. 4, pp. 409–416, Dec, 2008. ISSN: 1573-2908.

[57] TIBSHIRANI, R., WALTHER, G., HASTIE, T. "Estimating the number of clusters in a data set via the gap statistic", *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, v. 63, n. 2, pp. 411–423, 2001. ISSN: 1467-9868.

[58] DAVIES, D. L., BOULDIN, D. W. "A cluster separation measure", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. PAMI-1, n. 2, pp. 224–227, Apr, 1979. ISSN: 0162-8828.

[59] CALIŃSKI, T., HARABASZ, J. "A dendrite method for cluster analysis", *Communications in Statistics*, v. 3, n. 1, pp. 1–27, 1974.

[60] ROUSSEEUW, P. J. "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis", *Journal of Computational and Applied Mathematics*, v. 20, n. Supplement C, pp. 53–65, 1987. ISSN: 0377-0427.

[61] VINH, N. X., EPPS, J., BAILEY, J. "Information theoretic measures for clusterings comparison: Is a correction for chance necessary?" In: *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 1073–1080, New York, USA, 2009. ISBN: 978-1-60558-516-1.

[62] WILD, J., PREKOPCSAK, Z., SIEGER, T., et al. "Performance comparison of extracellular spike sorting algorithms for single-channel recordings", *Journal of Neuroscience Methods*, v. 203, n. 2, pp. 369–376, 2012. ISSN: 0165-0270.

[63] STEINMETZ, P. N. "Comparison of combined spike detection and clustering using mutual information", *Journal of Neuroscience Methods*, v. 291, n. Supplement C, pp. 166–175, 2017. ISSN: 0165-0270.

[64] TIELEMAN, T. "Training restricted boltzmann machines using approximations to the likelihood gradient". In: *Proceedings of the 25th International Conference on Machine Learning*, pp. 1064–1071, New York,USA, 2008. ISBN: 978-1-60558-205-4.

[65] TURIAN, J., RATINOV, L., BENGIO, Y. "Word representations: A simple and general method for semi-supervised learning". In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 384–394, Stroudsburg,USA, 2010.

[66] NIEDIEK, J., BOSTRÖM, J., ELGER, C. E., et al. "Reliable analysis of single-unit recordings from the human brain under noisy conditions: Tracking neurons over hours", *PLOS ONE*, v. 11, n. 12, pp. 1–26, 12 2016.

[67] CHAURE, F. J., QUIROGA, R. Q., KOCHEN, S. S., et al. "A novel unsupervised spike sorting implementation with variable number of features". In: *2017 XVII Workshop on Information Processing and Control*, pp. 1–7, Sept, 2017.

[68] YANG, Z., ZHAO, Q., KEEFER, E., et al. "Noise characterization, modeling, and reduction for in vivo neural recording". In: *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, pp. 2160–2168, USA, 2009. ISBN: 978-1-61567-911-9.

[69] LÓPEZ, C. M., WELKENHUYSEN, M., MUSA, S., et al. "Towards a noise prediction model for in vivo neural recording". In: *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 759–762, Aug, 2012.

[70] CHEN, C. H., PUN, S. H., MAK, P. U., et al. "Circuit models and experimental noise measurements of micropipette amplifiers for extracellular neural recordings from live animals", *Biomed Research International*, v. 2014, pp. 135026, Jul, 2014. ISSN: 2314-6133.

[71] HASSIBI, A., NAVID, R., DUTTON, R. W., et al. "Comprehensive study of noise processes in electrode electrolyte interfaces", *Journal of Applied Physics*, v. 96, n. 2, pp. 1074–1082, 2004.

[72] RAZAVI, B. *Design of Analog CMOS Integrated Circuits*. New York, USA, McGraw-Hill, Inc., 2001. ISBN: 0072380322, 9780072380323.

[73] AKANSU, A. N., HADDAD, R. A. *Multiresolution signal decomposition: Transforms, subbands, and wavelets*. Orlando,USA, Academic Press, Inc., 1992. ISBN: 012047140X.

[74] BESTEL, R., DAUS, A. W., THIELEMANN, C. "A novel automated spike sorting algorithm with adaptable feature extraction", *Journal of Neuroscience Methods*, v. 211, n. 1, pp. 168–178, 2012. ISSN: 0165-0270.

[75] NGUYEN, T., BHATTI, A., KHOSRAVI, A., et al. "Automatic spike sorting by unsupervised clustering with diffusion maps and silhouettes", *Neurocomputing*, v. 153, pp. 199–210, 2015. ISSN: 0925-2312.

[76] FRANKE, F., QUIAN QUIROGA, R., HIERLEMANN, A., et al. "Bayes optimal template matching for spike sorting — combining fisher discriminant analysis with optimal filtering", *J. Comput. Neurosci.*, v. 38, n. 3, pp. 439–459, jun. 2015. ISSN: 0929-5313.

[77] ZHANG, W., LIU, Q. "Using the center loss function to improve deep learning performance for EEG signal classification". In: *2018 Tenth International Conference on Advanced Computational Intelligence*, pp. 578–582, Mar, 2018.

[78] WEN, T., ZHANG, Z. "Deep convolution neural network and autoencoders-based unsupervised feature learning of EEG signals", *IEEE Access*, v. 6, pp. 25399–25410, 2018.

[79] SCHIRRMEISTER, R. T., SPRINGENBERG, J. T., FIEDERER, L. D. J., et al. "Deep learning with convolutional neural networks for EEG decoding and visualization", *Human Brain Mapping*, Aug, 2017. ISSN: 1097-0193.

[80] GIBSON, S., JUDY, J. W., MARKOVIĆ, D. "Spike sorting: The first step in decoding the brain", *IEEE Signal Processing Magazine*, v. 29, n. 1, pp. 124–143, 2012. ISSN: 1053-5888.

[81] TEETERS, J. L., HARRIS, K. D., MILLMAN, K. J., et al. "Data sharing for computational neuroscience", *Neuroinformatics*, v. 6, n. 1, pp. 47–55, Mar, 2008. ISSN: 1559-0089.

[82] MCDOUGAL, R. A., BULANOVA, A. S., LYTTON, W. W. "Reproducibility in computational neuroscience models and simulations", *IEEE transactions on bio-medical engineering*, v. 63, n. 10, pp. 2021–2035, Oct, 2016. ISSN: 0018-9294.

[83] JAN, S., URS, F., DAVID, J., et al. "Independent-component-analysis-based spike sorting algorithm for high-density microelectrode array data processing". In: *2009 IEEE Sensors*, pp. 384–386, Oct, 2009.

[84] IAN GOODFELLOW, YOSHUA BENGIO, A. C. "Deep Learning". Mit Press, 2016.

[85] CITI, L., CARPANETO, J., YOSHIDA, K., et al. "On the use of wavelet denoising and spike sorting techniques to process electroneurographic signals recorded using intraneural electrodes", *Journal of Neuroscience Methods*, v. 172, n. 2, pp. 294–302, 2008. ISSN: 0165-0270.

[86] VERMA, N., VERMA, A. "Performance analysis of wavelet thresholding methods in denoising of audio signals of some indian musical instruments", v. 4, 05 2012.

[87] VALENCIA, D., OREJUELA, D., SALAZAR, J., et al. "Comparison analysis between rigrsure, sqtwolog, heursure and minimaxi techniques using hard and soft thresholding methods". In: *2016 XXI Symposium on Signal Processing, Images and Artificial Vision*, pp. 1–5, Aug, 2016.

Apêndice A

Base de Dados

A comparação entre algoritmos de *spike sorting* é difícil, principalmente, porque há poucas bases de dados previamente e corretamente classificados (*ground truth*) [80]. Grande parte dos trabalhos sobre *spike sorting* utiliza bases de dados próprias [81, 82], geradas com configurações específicas para os processos de aquisição. Mesmo que se tente reproduzir uma configuração específica, não há garantia de que o sinal registrado terá número de neurônios ou nível de ruído iguais aos do sinal que se tenta aproximar. É comum usar bases de dados simuladas, que permitem avaliar comparativamente as diferentes características dos algoritmos de *spike sorting*. A seguir se apresenta uma breve descrição das bases de dados, simuladas, utilizadas nesta tese. Em ambas bases de dados, os registros celulares estão amostrados a 24 kHz. São conhecidos: o número de neurônios (*single-unit activity*) presentes em cada registro celular, os instantes em que os *spikes* ocorrem e a classe à qual cada *spike* pertence.

Base de Dados 1

A base de dados "Data Set 1"permite a comparação entre algoritmos de *spike sorting* considerando diferentes números de neurônios. Ela contém 95 registros extracelulares. Cada registro extracelular tem 10 minutos de duração, o que corresponde a aproximadamente 6.486 *spikes* se considerados os 95 registros. Há cinco registros extracelulares contendo *spikes* provenientes de dois neurônios distintos, cinco registros extracelulares contendo *spikes* provenientes de três neurônios distintos, e assim por diante até 20 neurônios distintos.

Essa base de dados foi proposta originalmente em [23]. Os registros incluem ruído de fundo e potenciais de ação do tipo *multiunit* e *single unit*. O ruído de fundo e as atividades *multiunit* e *single unit* foram gerados a partir de uma base de dados formada por 594 *spikes* obtidos de registros neurais realizados nas regiões do gânglio basal e do neocortex do cérebro de um macaco.

Para gerar um *spike* que será considerado ruído de fundo pela etapa de detecção, um subconjunto de *spikes* é selecionado. Os *spikes* são então escalados, sobrepostos e inseridos no registro extracelular em instantes de tempo aleatórios. Os fatores de escalamento foram gerados a partir de uma distribuição gaussiana com média igual a 1 e desvio-padrão igual a 0,2. Do sinal resultante da sobreposição do *spikes* foi subtraído o valor médio dos *spikes* incluídos na sobreposição. O resultado foi escalado pelo valor 0,1. Em média, cada segundo de sinal de ruído foi gerado pela sobreposição de 48.000 *spikes*.

A atividade do tipo *multiunit* foi incluída para produzir um cenário mais realista. Para a sua simulação foram selecionados aleatoriamente 20 *spikes* com formas de onda diferentes e amplitude 0,5. Cada um dos *spikes* foi escalado e somado ao sinal de ruído seguindo uma distribuição de Poisson independente, com taxa de disparo média de 0,25 Hz .

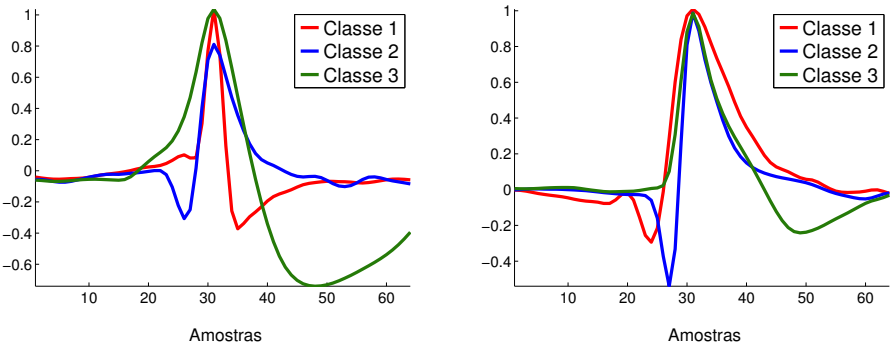
Para gerar cada atividade do tipo *single unit*, um *spike* é aleatoriamente selecionado dentre os 594 possíveis e então é escalado e inserido no sinal simulado. A taxa com que o *spike* é inserido no sinal é gerada por uma distribuição de Poisson, de média randomicamente selecionada a partir de uma distribuição uniforme com limites [0,1;2] Hz. Antes de cada inserção do *spike* selecionado, ele é escalado por um valor obtido via uma distribuição gaussiana com média 1,1 e desvio padrão 0,5, e limitada à faixa [0,9;2]. Esta base de dados não inclui casos de sobreposições de atividades do tipo *single unit*.

Base de Dados 2

A base de dados "Data Set 2"permite a comparação entre algoritmos de *spike sorting* considerando diferentes níveis de ruído. Ela contém 20 registros extracelulares. Cada registro extracelular tem 1 minuto de duração, o que corresponde a aproximadamente 2.662 *spikes* se considerados os 20 registros, e contém *spikes* provenientes de três neurônios distintos. Em termos do nível de dificuldade, estes registros extracelulares podem ser divididos nas categorias "easy1", "easy2", "difficult1"e "difficult2". A categoria "easy1"possui níveis de ruído (desvios padrões do ruído de fundo) iguais a (0,05; 0,1; 0,15; 0,2; 0,25; 0,3; 0,35; 0,4). As demais categorias apresentam níveis de ruído iguais a (0,05; 0,1; 0,15; 0,2). A base de dados apresentada foi proposta originalmente em [33].

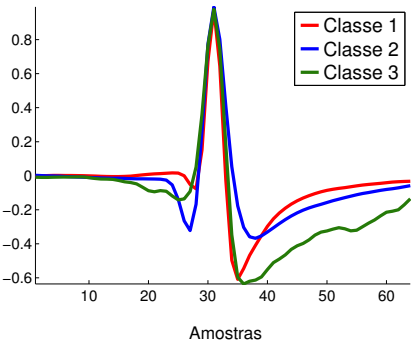
Os processos para simular os ruídos de fundo, bem como as atividades do tipo *multiunit* e *single unit* são os mesmos utilizados no "Data Set 1". A diferença é que a base de dados "Data Set 2"simula ruídos de fundo com diferentes valores de desvio padrão e também inclui casos de sobreposições de atividades do tipo *single unit*.

A Figura A.1 mostra as formas de onda dos *spikes* utilizados para gerar quatro registros extracelulares com diferentes níveis de dificuldade para a tarefa de *spike sorting*. A Figura A.1(a) corresponde ao registro "easy1"e nível de ruído $0,4$. A Figura A.1(b) corresponde ao registro "easy2"e nível de ruído $0,2$. A Figura A.1(c) corresponde ao registro "difficult1"e nível de ruído $0,2$ e a Figura A.1(d) corresponde ao registro "difficult2"e nível de ruído $0,2$. Em todos os casos as formas de onda foram alinhadas pelo ponto de máximo.

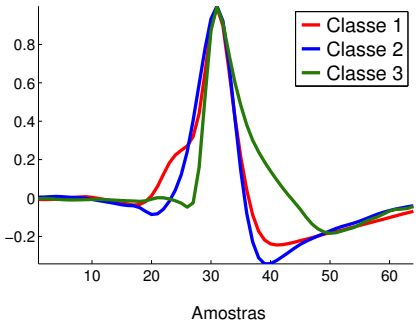


(a) easy 1 - Nível de ruído 0,4

(b) easy 2 - Nível de ruído 0,2



(c) difficult 1 - Nível de ruído 0,2



(d) difficult 2 - Nível de ruído 0,2

Figura A.1: Formas de onda dos *spikes* utilizados para gerar registros extracelulares com diferentes níveis de ruído.

Apêndice B

Extração de Características

Extração de características é um processo que consiste em transformar um conjunto de dados em um novo conjunto de valores, com o objetivo de manter apenas a informação relevante. A extração de características é uma etapa usualmente aplicada em métodos de *spike sorting* para, a partir de cada sinal de *spike*, extrair características que possibilitem uma melhor discriminação dos *spikes* gerados por diferentes tipos de neurônios, e também favoreçam a etapa de clusterização. Isso leva o clusterizador empregado no método de *spike sorting* a um melhor desempenho.

Características tais como diferença entre valor máximo e mínimo, máxima amplitude e energia do sinal de spike, foram utilizadas em estudos iniciais [14], porém, nos últimos anos, técnicas mais complexas de extração de características, tais como *independent component analysis* (ICA) [83] e transformada wavelet [37], têm se tornado populares.

Transformada Wavelet

A transformada wavelet permite avaliar o conteúdo de frequência de um sinal ao longo do seu domínio temporal. Ela possui uma boa resolução tanto no domínio do tempo quanto no domínio da frequência, e também pode ser aplicada para sinais não estacionários. A transformada wavelet contínua, definida na Equação (B.1), é dada pela convolução entre o sinal analisado $x(t)$ (*spike*) e versões escaladas e deslocadas de função wavelet base (wavelet mãe) $\psi(t)$.

$$C(a,b;x(t),\psi(t)) = \frac{1}{|a|^{1/2}} \int_{-\infty}^{\infty} x(t)\overline{\psi}\left(\frac{t-b}{a}\right) dt \tag{B.1}$$

onde $a > 0$, $a \in \mathbb{R}^{+*}$ e $b \in \mathbb{R}$ correspondem, respectivamente, ao fator de escalamento e ao valor de translação da wavelet base. A função $\overline{\psi}(t)$ corresponde à forma conjugada complexa da função contínua $\psi(t)$.

Diversas famílias de wavelet, tais como Haar, Daubechies e Symlets podem ser aplicadas para extração de características de sinais de *spike*. Nos trabalhos [33, 37] a

forma discreta da transformada wavelet (DWT) foi aplicada na etapa de extração de características de sinais *spike*. A transformada wavelet contínua (CWT) possibilita uma análise mais detalhada (escala) no domínio tempo-frequência e também permite a precisa localização de transientes. O vetor de coeficientes resultante da convolução de um sinal de spike $x(t)$ com uma wavelet $\psi(t)$, via transformada wavelet contínua, possuirá o mesmo número de dimensões que a de seu *spike* de origem.

Neste trabalho foi utilizada a wavelet "sym2"(família Symlets) devido ao seu suporte compacto. O suporte de uma wavelet dever ser pequeno o suficiente para permitir detectar características próximas no domínio do tempo. A utilização de wavelet com suporte grande pode resultar em coeficientes que impossibilitem uma boa discriminação dos spikes gerados por diferentes tipos de neurônios [33].

Apêndice C

Ajuste dos Hiperparâmetros do MLP-KLD

Existem diversas estratégias automáticas ou manuais que podem ser utilizadas no processo de seleção dos hiperparâmetros do algoritmo de aprendizado. Uma estratégia automática tipicamente adotada é a busca em *grid* [84]. Para realizar a busca em *grid*, inicialmente devem ser definidos os possíveis valores assumidos por cada hiperparâmetro, por exemplo, $\alpha = \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ e $\lambda = \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$, e então constrói-se o conjunto (*grid*) de possibilidades contendo todos os pares (α,λ) a serem avaliados. Para o exemplo apresentado existem 16 possíveis combinações (α,λ).

A seleção dos hiperparâmetros é um processo que deve ser repetido para cada topologia ou conjunto de dados utilizado para o treinamento de um modelo, ou seja, sempre que uma nova topologia ou conjunto de dados for utilizado, o processo de seleção dos hiperparâmetros deve ser realizado novamente. Portanto, considerando somente os hiperparâmetros α e λ, e definidos a topologia do MLP e o conjunto de dados (conjunto de *spikes*), é possível realizar o treinamento de 16 diferentes modelos – um para cada combinação (α,λ). Após o treinamento de todos os modelos, a combinação de hiperparâmetros selecionada será aquela que produziu o modelo com a maior acurácia de classificação na tarefa de *spike sorting*.

Para os experimentos realizados ao longo desta tese foram utilizadas duas bases de dados, que juntas correspondem a 115 conjuntos de dados, e foram treinadas 20 topologias distintas de MLPs. Considerando apenas as 16 possíveis combinações para os hiperparâmetros α e λ utilizados no exemplo, seria necessário realizar o treinamento de no mínimo 36.800 (20x115x16) modelos. Tal tarefa além de ser exaustiva, consumiria um tempo de processamento não disponível.

Sendo assim, optou-se pela seleção manual dos hiperparâmetros. Para entender a influência de cada hiperparâmetro em relação ao desempenho alcançado pelos MLPs treinados via MLP-KLD, foi tomada como referência a topologia contendo

79 entradas, uma camada escondida composta por 30 neurônios artificiais do tipo sigmoidal e camada de saída composta por dois neurônios artificiais do tipo linear. Para o treinamento dos MLPs foram selecionados cinco registros extracelulares obtidos a partir da base de dados "Data Set 1". Os registros selecionados contêm *spikes* gerados por quatro neurônios biológicos distintos.

Os hiperparâmetros avaliados foram: a taxa de aprendizado α, o multiplicador de Lagrange λ e o fator de esquecimento β. Os demais hiperparâmetros foram selecionados de maneira empírica. Adotou-se número de épocas igual a 1.000 como critério de parada algoritmo MLP-KLD.

Os resultados mostrados na Tabela C.1, correspondem a médias calculadas sobre cinco registros diferentes, todos gerados por quatro neurônios biológicos distintos. Cada uma das linhas corresponde a um diferente valor para o hiperparâmetro α, porém para todos os casos os hiperparâmetros β, λ são considerados nulos. A coluna "Médias"corresponde à média global, resultante da média das três rodadas realizadas para cada valor atribuído ao hiperparâmetro α.

É possível notar que o hiperparâmetro α exerce grande influência sobre a acurácia de classificação alcançada pelos MLPs treinados via MLP-KLD. A maior acurácia de classificação foi obtida para $\alpha = 10^{-1}$ e a menor acurácia foi obtida para $\alpha = 10^{-4}$.

Tabela C.1: valores médios de acurácia de classificação, número de *hits* e erros, em conjuntos de dados contendo 4 neurônios biológicos. Hiperparâmetro avaliado α.

# Rodadas		1ª	2ª	3ª	Médias
$\alpha = 10^{-1}$	Acurácia	0,7334	0,7297	0,7315	**0,7315**
	Hits	3,2	3	3,2	3,1
	Erros	357	369,6	360	362,2
$\alpha = 10^{-2}$	Acurácia	0,7139	0,7166	0,6792	0,7032
	Hits	3,2	3,2	2,8	3,1
	Erros	387	389,2	462,2	412,8
$\alpha = 10^{-3}$	Acurácia	0,5389	0,4945	0,5915	0,5416
	Hits	2,4	2,4	2,6	2,5
	Erros	740	862,6	621,8	741,5
$\alpha = 10^{-4}$	Acurácia	0,29	0,3645	0,3452	0,3332
	Hits	1,4	1,6	1,6	1,5
	Erros	1.295,4	1.079,6	1.164	1.179,7

A Tabela C.2 apresenta a influência hiperparâmetro λ no desempenho dos modelos treinados. Para este caso considerou-se $\alpha = 10^{-1}$ e $\beta = 0$. Cada uma das linhas corresponde a um diferente valor para o hiperparâmetro λ. É possível notar que diferente do hiperparâmetro α, o hiperparâmetro λ exerceu menor influência sobre a acurácia de classificação alcançada pelos MLPs. A maior acurácia foi obtida para $\lambda = 10^{-2}$ e a menor acurácia foi obtida para $\lambda = 10^{-1}$.

Tabela C.2: valores médios de acurácia de classificação, número de *hits* e erros, em conjuntos de dados contendo 4 neurônios biológicos. Hiperparâmetro avaliado λ.

# Rodadas		1ª	2ª	3ª	Médias
$\lambda = 10^{-1}$	Acurácia	0,7033	0,7429	0,7276	0,7246
	Hits	3	3,4	3,2	3,2
	Erros	410	347,8	365,2	374,3
$\lambda = 10^{-2}$	Acurácia	0,7297	0,7495	0,7438	**0,7410**
	Hits	3,2	3,4	3,2	3,3
	Erros	361,8	333	337	343,9
$\lambda = 10^{-3}$	Acurácia	0,7434	0,7239	0,7380	0,7351
	Hits	3,2	3	3,2	0,31
	Erros	340,2	372,6	351	354,6
$\lambda = 10^{-4}$	Acurácia	0,7425	0,7295	0,7320	0,7347
	Hits	3,2	3,2	3,2	3,2
	Erros	337,6	362,2	355,8	351,9

A Tabela C.3 apresenta a influência do hiperparâmetro β no desempenho dos modelos treinados. Para este caso considerou-se $\alpha = 10^{-1}$ e $\lambda = 10^{-4}$. Assim como o hiperparâmetro λ, o hiperparâmetro β exerceu menor influência sobre a acurácia de classificação alcançada pelos MLPs. A maior acurácia foi obtida para $\beta = 0,1$ e a menor acurácia foi obtida para $\beta = 0,4$.

Tabela C.3: valores médios de acurácia de classificação, número de *hits* e erros, em conjuntos de dados contendo 4 neurônios biológicos. Hiperparâmetro avaliado β.

# Rodadas		1ª	2ª	3ª	Médias
$\beta = 0,1$	Acurácia	0,7570	0,7442	0,8366	**0,7793**
	Hits	3,4	3,2	3,6	3,4
	Erros	318,8	338,4	217,2	291,5
$\beta = 0,2$	Acurácia	0,7589	0,7369	0,7358	0,7439
	Hits	3,2	3,2	3,2	3,2
	Erros	311,2	357,2	349,2	339,2
$\beta = 0,3$	Acurácia	0,7529	0,7325	0,7163	0,7339
	Hits	3,2	3	3	3,1
	Erros	329	356,6	394,6	360
$\beta = 0,4$	Acurácia	0,7368	0,7271	0,7332	0,7324
	Hits	3,2	3,2	3,2	3,2
	Erros	346,8	362,4	357,2	355,5

Por fim, optou-se por utilizar os mesmos valores de α, β e λ para todas as topologias e conjuntos de dados utilizados ao longo desta tese. Os valores adotados foram $\alpha = 10^{-1}$, $\lambda = 10^{-4}$ e $\beta = 0,2$.

Apêndice D

Filtragem de Registros Extracelulares via Transformada Wavelet Discreta

O objetivo do experimento apresentado nesta seção é avaliar se o uso da transformada wavelet discreta [73, 85] para a filtragem de ruído presente nos registros extracelulares contidos no "Data Set 2", pode contribuir para um aumento no desempenho dos MLPs treinados via algoritmo MLP-KLD.

Para a formação dos conjuntos de dados, inicialmente, cada um dos 20 registros extracelulares contidos no "Data Set 2"é filtrado via transformada wavelet discreta usando o método *heursure* [86, 87] com um nível de decomposição. A wavelet utilizada foi do tipo "sym2". Após a filtragem, os *spikes* são extraídos com comprimento de 64 amostras de modo que o seu ponto de máximo esteja localizado na amostra 31. Os *spikes* são armazenados como linhas de uma matriz. Cada uma das 64 colunas da matriz passa pelo processo de normalização linear que faz com que a coluna tenha valor mínimo igual a zero e valor máximo igual a um. Os 20 registros extracelulares dão origem a 20 conjuntos de dados.

Os MLPs treinados via algoritmo MLP-KLD possuem uma camada escondida composta por 400 neurônios artificiais do tipo sigmoidal e camada de saída composta por dois neurônios artificiais do tipo linear. Os parâmetros de treinamento utilizados foram $\alpha = 10^{-1}$, $\beta = 0,1$, $\lambda = 10^{-4}$ e $\gamma = 10^{-2}$ e máximo de épocas igual a 3.000, sendo que a cada época 30% das sinapses foram mantidas fixas. Para cada MLP treinado, sua saída (pontos mapeados para o espaço 2-D) foi clusterizada via K-means e as medidas de desempenho foram calculadas.

Na Tabela D.1, as colunas 1ª, 2ª e 3ª (referentes a três rodadas completas do algoritmo de treinamento) mostram o número de erros e acurácia de classificação. Na última linha da tabela é possível notar que a acurácia média (entre parênteses)

é de aproximadamente 83,03%. Este valor é menor do que a acurácia média obtida quando os MLP são treinados a partir dos dados sem a etapa de filtragem, ou seja, dados com ruído. Uma possível interpretação para os resultados obtidos é que a etapa de filtragem pode ter reduzido a quantidade de informação utilizada pela rede neural para aprender as características que permitem uma melhor discriminação entre os diferentes tipos de neurônios biológicos presentes nos registros extracelulares.

Tabela D.1: Número de erros e acurácia alcançados pelo algoritmo K-means básico executado sobre dados 2-D gerados através do algoritmo MLP-KLD. A transformada wavelet discreta foi utilizada na etapa de filtragem dos registros extracelulares.

Data Set 2	Nível de Ruído	# Spikes	1ª	2ª	3ª
easy1	0,05	2.729	0 (100)	0 (100)	0 (100)
	0,10	2.753	0 (100)	0 (100)	0 (100)
	0,15	2.693	3 (99,89)	0 (100)	3 (99,89)
	0,20	2.678	34 (98,73)	38 (98,58)	42 (98,43)
	0,25	2.586	90 (96,52)	73 (97,18)	110 (95,75)
	0,30	2.629	248 (90,57)	206 (92,16)	199 (92,43)
	0,35	2.702	453 (83,23)	384 (85,79)	363 (86,57)
	0,40	2.645	1246 (52,89)	868 (67,18)	857 (67,60)
easy2	0,05	2.619	0 (100)	0 (100)	0 (100)
	0,10	2.694	21 (99,22)	12 (99,55)	18 (99,33)
	0,15	2.648	114 (95,69)	112 (95,77)	149 (94,37)
	0,20	2.715	812 (70,09)	475 (82,50)	482 (82,25)
difficult1	0,05	2.616	0 (100)	2 (99,92)	1 (99,96)
	0,10	2.638	109 (95,87)	92 (96,51)	93 (96,47)
	0,15	2.660	1943 (26,95)	1940 (27,07)	1947 (26,80)
	0,20	2.624	2004 (23,63)	2005 (23,59)	2001 (23,74)
difficult1	0,05	2.535	3 (99,88)	2 (99,92)	2 (99,92)
	0,10	2.742	560 (79,58)	803 (70,71)	583 (78,74)
	0,15	2.631	761 (71,08)	758 (71,19)	750 (71,49)
	0,20	2.716	996 (63,33)	1390 (48,82)	952 (64,95)
Valor médio		2.662,7	469,85 (82,35)	458 (82,82)	427,6 (83,93)

Apêndice E

Avaliação do Tempo Computacional

Conforme descrito na Seção 5.3, diferente dos algoritmos PCA, LDA e LPP, os algoritmos MLP-KLD e t-SNE são iterativos. Sua execução é baseada em um conjunto de etapas que são repetidas ao longo de cada época de iteração. O objetivo dos experimentos apresentados nesta seção é avaliar o tempo computacional médio de uma época de iteração dos algoritmos MLP-KLD e t-SNE em diferentes condições de experimento. Deste ponto em diante o "tempo computacional médio de uma época de iteração"será chamado de "tempo médio de iteração". Para calcular o tempo médio de iteração, os algoritmos foram executados por 1.000 épocas, o tempo de execução de cada época foi armazenado e ao final das 1.000 épocas calculou-se o valor médio dos tempos armazenados.

Todos os experimentos apresentados ao longo desta tese foram executados em computadores com processador Intel(R) Core(TM) i7-3770 CPU 3.4 GHz, 32 GB de memória RAM e sistema operacional Linux.

Para a formação dos conjuntos de dados, a partir do "Data Set1", selecionou-se um registro extracelular contendo 20 neurônios biológicos e deste registro foram extraídos 12.000 *spikes*. Os *spikes* extraídos foram organizados como as linhas de uma matriz (conjunto de dados completo).

A Figura E.1 apresenta a comparação entre o tempo médio de iteração do algoritmo MLP-KLD quando aplicado para o treinamento de um MLP com topologia 79-600-2, e o tempo médio de iteração do algoritmo t-SNE. Neste experimento os mapeamentos são de 79 para duas dimensões. Os algoritmos são avaliados na condição de aumento do número de *spikes* presentes no conjunto de dados.

É possível notar que a diferença entre o tempo médio de iteração dos algoritmos MLP-KLD e t-SNE aumenta a medida em que são utilizados conjuntos de dados com um maior número de *spikes*. A maior diferença ocorre quando os algoritmos são submetidos ao conjunto de dados completo (12.000 *spikes*). Neste caso, o tempo

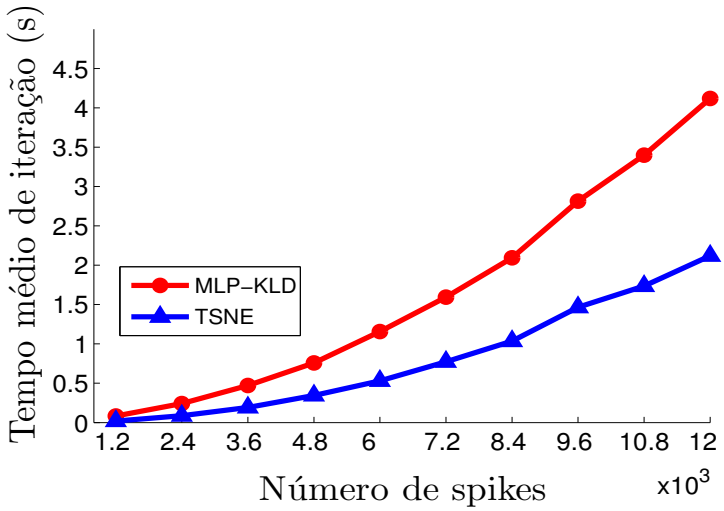


Figura E.1: Tempo médio de iteração dos algoritmos MLP-KLD e t-SNE em função do número de *spikes* presentes nos conjuntos de dados. O algoritmo MLP-KLD foi aplicado para o treinamento de um MLP contendo 79 entradas, 600 neurônios artificiais na camada escondida e dois neurônios artificiais na camada de saída.

médio de iteração do MLP-KLD é aproximadamente o dobro do tempo médio de iteração do t-SNE.

A Figura E.2 apresenta a evolução do tempo médio de iteração do algoritmo MLP-KLD quando aplicado para o treinamento de MLPs contendo 30, 200, 400, 600 ou 800 neurônios artificiais na camada escondida. Todos os MLPs treinados possuem 79 entradas, uma camada escondida e dois neurônios artificiais na camada de saída.

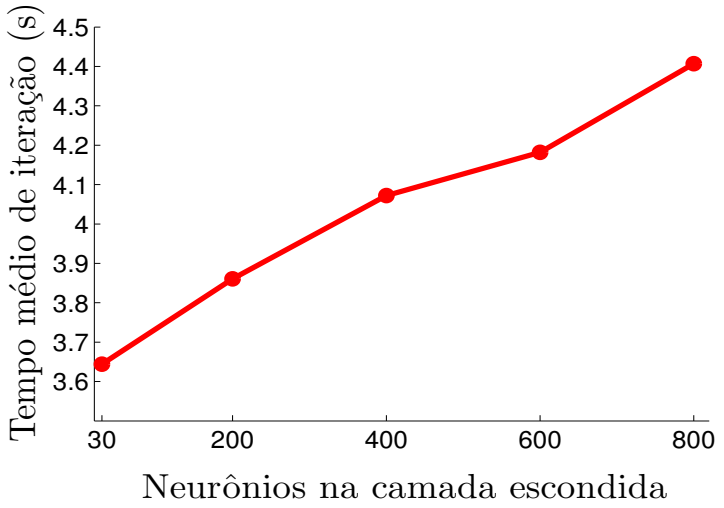


Figura E.2: Tempo médio de iteração do algoritmo MLP-KLD em função do número de neurônios artificiais presentes na camada escondida dos MLPs treinados via algoritmo MLP-KLD.

É possível notar uma relação aproximadamente linear entre o número de neurônios na camada escondida e o tempo médio de iteração. Para este experimento foi utilizado o conjunto de dados completo.

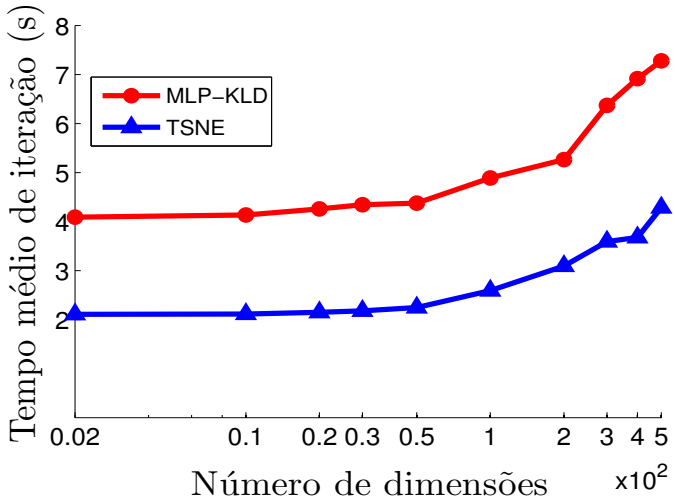


Figura E.3: Tempo médio de iteração dos algoritmos MLP-KLD e t-SNE em função do número de dimensões do domínio de mapeamento.

A Figura E.3 apresenta a comparação entre o tempo médio de iteração dos algoritmos MLP-KLD e t-SNE, e o número de dimensões do domínio de mapeamento. O eixo das ordenadas está em escala logarítmica para permitir uma melhor representação. É possível notar que os tempos de iteração são aproximadamente constantes para 2, 10, 20 e 30 dimensões no espaço de mapeamentos. A partir de 50 dimensões o tempo de iteração apresenta uma tendência de crescimento. A diferença entre os tempos médios de iteração dos algoritmos permanece aproximadamente constante independente do número de dimensões do espaço de mapeamento. Todos os MLPs treinados via MLP-KLD possuem 79 entradas, 600 neurônios artificiais na camada escondida e 2, 10, 20, 30, 50, 100, 200, 300, 400 ou 500 neurônios artificiais na camada de saída.

A Figura E.4 apresenta a evolução do tempo médio de iteração do algoritmo MLP-KLD quando aplicado para o treinamento de MLPs contendo 1, 2, 3, 4, 5, 6 ou 7 camadas escondidas.

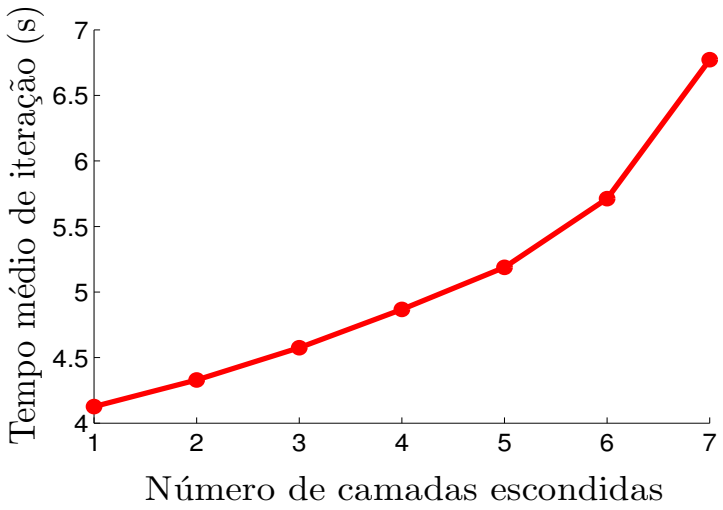


Figura E.4: Tempo médio de iteração do algoritmo MLP-KLD em função do número de camadas escondidas presentes nos MLPs treinados via algoritmo MLP-KLD.

É possível notar uma relação exponencial entre o número de camadas escondidas e o tempo médio de iteração. As topologias avaliadas neste experimento foram as mesmas topologias avaliadas na Seção 5.4.