UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

LAURA ANGELICA TOMAZ DA SILVA

FERRAMENTA EXPERIMENTAL PARA ANÁLISE DE PADRÕES DE ATIVIDADE CEREBRAL

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2015

LAURA ANGELICA TOMAZ DA SILVA

FERRAMENTA EXPERIMENTAL PARA ANÁLISE DE PADRÕES DE ATIVIDADE CEREBRAL

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 1, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Profa. Beatriz Terezinha Borsoi

PATO BRANCO

2015

ATA Nº: **272**

DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DO ALUNO **LAURA ANGELICA TOMAZ DA SILVA**.

Às 17:00 hrs do dia 25 de novembro de 2015, Bloco V da UTFPR, Câmpus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Beatriz Terezinha Borsoi (Orientadora), Soelaine Rodrigues Ascari (Convidada) e Kathya Silvia Collazos Linares (Convidada), para avaliar o Trabalho de Diplomação do aluno Laura Angelica Tomaz da Silva, matrícula 1168010, sob o título **Ferramenta experimental para análise de padrões de atividade cerebral** ; como requisito final para a conclusão da disciplina Trabalho de Diplomação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, COADS. Após a apresentação o candidato foi entrevistado pela banca examinadora, e a palavra foi aberta ao público. Em seguida, a banca reuniu-se para deliberar considerando o trabalho **APROVADO**. Às 17:40 hrs foi encerrada a sessão.

Beatriz Borsoi

Profa. Beatriz Terezinha Borsoi, Dr.
Orientadora

Profa. Soelaine Rodrigues Ascari, M.Sc.
Convidada

Kathya S. Collazos

Profa. Kathya Silvia Collazos Linares, Dr.
Convidada

Profa. Soelaine Rodrigues Ascari, M.Sc.
Coordenador do Trabalho de Diplomação

Prof. Edilson Pontarolo, Dr.
Coordenador do Curso

RESUMO

SILVA, Laura Angélica Tomaz da. Ferramenta experimental para análise de padrões de atividade cerebral. 2015. 51f. Monografia (Trabalho de Conclusão de Curso) - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2015.

O cérebro é um órgão muito complexo e os estudos realizados na tentativa de entendimento do seu funcionamento podem ser realizados de maneiras diferentes utilizando diferentes técnicas e experimentos bastante diversificados. Até recentemente apenas as estruturas do cérebro eram estudadas, mas o avanço de tecnologias (como o surgimento de tomografia por emissão de pósitrons e ressonância magnética funcional) tem permitido investigar além da estrutura anatômica do cérebro visando entendimento do funcionamento e relacionamento das estruturas desse órgão em sua visão microscópica. Na Universidade Médica da Carolina do Sul, Campus Charleston, atualmente os profissionais da divisão de neuroimagem do departamento de neurociência utilizam uma ferramenta desenvolvida por estudiosos na Universidade de Princeton. Porém, para a utilização dessa ferramenta é necessário ter um prévio conhecimento da linguagem *MATrix LABoratory* (MATLAB) e, mesmo assim, a ferramenta não oferece um ambiente de fácil interação que é por linha de comando. Assim, considerando as possibilidades de análise de dados obtidos de atividade cerebral e a diversidade de profissionais envolvidos nessas análises e que nem sempre possuem conhecimento aprofundado para lidar com ferramentas computacionais complexas, o desenvolvimento de uma interface gráfica de fácil uso se apresentou como necessário para incentivar o uso da ferramenta MVPA da Universidade de Princeton. Assim, por meio da realização desse trabalho foi implementada uma ferramenta experimental com uso de MATLAB para uso por médicos para diagnosticar pacientes baseando-se apenas em padrões de sua atividade cerebral. Os dados para essa análise são obtidos por meio de dados de ressonância magnética (fMRI).

Palavras-chave: fMRI. Ferramenta MVPA. MATLAB.

ABSTRACT

SILVA, Laura Angélica Tomaz da. Interactive Platform for Analysis of Brain Activity Patterns. 51f. Monografia (Trabalho de Conclusão de Curso) - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco. Pato Branco, 2015.

The brain is a very complex organ and studies performed trying to understanding its operation can be done in different ways using different techniques and very diverse experiments. Until recently only brain structures were studied, but the progress of technologies (such as the uprising of positron emission tomography and functional magnetic resonance imaging) has allowed researchers to investigate beyond the anatomical structure of the brain aiming at understanding the functioning and relationship of structures from this organ in a microscopic view. At the Medical University of South Carolina, Charleston Campus, currently professionals in the neuroimaging division of the department of neuroscience have been using a tool developed by researchers at Princeton University. However, to use this tool it is necessary to have a prior knowledge of language Matrix Laboratory (MATLAB) and, even so, the toolbox does not offer an environment of easy interaction being done by command line. Thus, considering the possibilities of analysis of data obtained from brain activity and the diversity of professionals involved in these analysis that do not always possess an in-depth knowledge to handle complex computational tools, the development of an easy graphical interface is presented as necessary to encourage the use of the MVPA toolbox of Princeton University. Therefore, by performing this work it was implemented an experimental tool with MATLAB to be used by physicians to diagnose patients based only on patterns of their brain activity. The data for this analysis are obtained by magnetic resonance data (fMRI).

keywords: fMRI. MVPA Toolbox. MATLAB.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTAGENS DE CÓDIGO

LISTA DE SIGLAS

2D	Duas dimensões
3D	Três dimensões
4D	Quatro dimensões
ANOVA	*Analise of Variance*
BOLD	*Blood Oxygenation Level Dependente imaging*
fMRI	*Functional Magnetic Resonance Imaging*
MATLAB	*Matrix Laboratory*
MRI	*Magnetic Resonance Imaging*
MUSC	Medical University of South Carolina
MVPA	*Multi-Voxel Pattern Analysis*
PET Scan	*Positron Emission Tomography*
TR	Tempo de Repetição

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais, os objetivos e a justificativa da realização deste trabalho. No final do capítulo é apresentada a organização do texto por meio de uma breve apresentação dos seus capítulos.

1.1 CONSIDERAÇÕES INICIAIS

O cérebro humano sempre foi e ainda o é um dos maiores mistérios que a ciência tenta explicar. Para isso, médicos, pesquisadores e admiradores por esse fascinante e complexo órgão humano trabalham com teorias diferenciadas e tentam criar novos sistemas para facilitar a compreensão do mesmo. Como os pensamentos são codificados no cérebro ainda é uma incógnita e, até recentemente, essa era uma questão deixada apenas para filósofos responderem. Atualmente, no entanto, com o desenvolvimento de novas tecnologias e computadores com capacidade de computação considerável se fez possível obter informações sobre os padrões de atividade cerebral e analisá-los em tempo hábil.

Visto que seria extremamente árduo para um ser humano encontrar os padrões de atividade cerebral provenientes de uma ressonância magnética funcional (*Functional Magnetic Resonance Imaging* (fMRI)), percebeu-se que é necessário um sistema para localizar e examinar essas informações. Hoje já existem algumas técnicas usadas para esse propósito. Entre elas, a que mais se destaca é *Machine Learning,* que é adotada para prever quais padrões de atividade cerebral estão associados a determinados pensamentos.

Na Universidade Médica da Carolina do Sul, Campus Charleston, atualmente os profissionais da divisão de neuroimagem do departamento em neurociência utilizam uma ferramenta desenvolvida por estudiosos na Universidade de Princeton. Porém, para a utilização dessa ferramenta é necessário ter um prévio conhecimento da linguagem *MATrix LABoratory* (MATLAB) e, mesmo assim, a ferramenta não oferece um ambiente de fácil interação para que pudesse ser utilizada por pessoas sem conhecimento prévio da ferramenta ou do ambiente, pois as tarefas são

realizadas por meio de linha de comando. Além disso, essa ferramenta possui algumas limitações tanto no que diz respeito ao tipo de dados que podem ser inseridos quanto a sua adaptação a diferentes experimentos que podem ser realizados, o que torna o seu uso ainda mais difícil para profissionais da área de saúde.

Assim sendo, verificou-se a necessidade de desenvolver um sistema computacional com interface gráfica para os profissionais dessa área tornando a sua utilização mais simples e, também, aprimorar a ferramenta desenvolvida pela Universidade de Princeton de forma a torná-la mais flexível na entrada de dados e experimentos que podem ser realizados.

1.2 OBJETIVOS

A seguir são apresentados os objetivos geral e específicos deste trabalho.

1.2.1 Objetivo Geral

Implementar uma ferramenta experimental para uso por médicos para diagnosticar pacientes baseando-se apenas em padrões de sua atividade cerebral.

1.2.2 Objetivos Específicos

- Disponibilizar uma ferramenta de fácil utilização por profissionais da área de saúde (médicos) ou pessoas que não são familiares com o uso do MATLAB.
- Permitir que neurocientistas, não familiares com ciência da computação ou MATLAB, sejam capazes de usar *Machine Learning* para investigar como pensamentos são codificados no cérebro.

- Tornar flexível o processo de entrada de dados do paciente pela ferramenta.
- Melhorar o processo de *Machine Learning* de uma ferramenta já desenvolvida pela Universidade de Princeton para que possa atender as necessidades dos profissionais da Universidade Médica da Carolina do Sul, a Medical University of South Carolina (MUSC).

1.3 JUSTIFICATIVA

O estudo do cérebro tem se tornado cada vez mais uma área de interesse entre a comunidade cientifica. Recentemente, uma grande quantidade de conhecimento sobre a anatomia e a psicologia do cérebro tem sido acumulada. Existe um extenso entendimento da estrutura organizacional do cérebro humano; porém, a forma como o cérebro utiliza suas conexões anatômicas para codificar e processar informações permanece incerta. Encontrar a relação entre os padrões em pequena escala de atividade cerebral e os estados mentais como emoções e percepção permanece uma questão não respondida em neurociência.

Os adventos recentes de técnicas de imagens funcionais permitem visualizar quais partes do cérebro estão ativas durante determinadas circunstâncias. Ressonância Magnética Funcional, fMRI, possibilita, particularmente, monitorar a atividade cerebral baseada nos níveis de oxigênio nos vasos sanguíneos. Ao contrário da maioria das técnicas de imagem, fMRI fornece várias fotos instantâneas (*snapshots*) em determinado tempo, garantindo a habilidade de representar a evolução dessas imagens dinâmicas ao longo do tempo e permitindo observar os padrões de atividade cerebral. Métodos tradicionais de estudo de funções cerebrais têm focado em estudar a relação entre regiões cerebrais individuais e condições experimentais. O cérebro, entretanto, é uma rede composta por sub-redes altamente inter e intra conectadas. A maior parte das informações não é codificada por regiões que são ativadas, mas sim nos padrões de interação entre essas regiões no decorrer do tempo.

Como é de se esperar, o foco nos últimos anos está voltado para o estudo de padrões de atividade multi-regionais. Uma técnica especialmente bem-sucedida –

Multi Voxel Pattern Analysis (MVPA) – usa sofisticados algoritmos de classificação de padrões juntamente com *Machine Learning* para decifrar as informações carregadas pelos padrões de ativação cerebral.

MVPA tem se mostrado útil em prever e classificar o estado de espírito de um individuo em um novo conjunto de dados com base em padrões previamente aprendidos, às vezes chamados “leitores de mente”, e presumir o diagnóstico de um paciente baseado em padrões de ativação de pacientes anteriores com a mesma condição.

MVPA, no entanto, é um método que poucos têm conhecimento. Código disponível gratuitamente é escasso e geralmente requer tempo e conhecimento em computação para ser possível usá-lo e customizá-lo. Neste trabalho, o propósito é tornar MVPA mais acessível para cientistas que não possuem necessariamente uma vasta experiência com MATLAB ou *Machine Learning*. Isso porque será realizada a construção de uma ferramenta em MATLAB com interface de fácil uso por meio do desenvolvimento de um recurso de importação flexível, permitindo que o usuário possa selecionar interativamente a condição regressora. Fornecendo, assim, ao usuário a oportunidade de escolher entre diferentes algoritmos de Machine Learning.

A ideia de desenvolver esse trabalho surgiu devido ao fato de que durante o verão de 2014, a autora deste documento foi estagiária do departamento em neurociência da divisão de neuroimagem desse na Universidade Médica da Carolina do Sul e manteve contato com um dos pesquisadores departamento após o seu retorno ao Brasil. Em uma das conversas entre a autora e o pesquisador, foi identificada a necessidade de uma plataforma com maior usabilidade para a ferramenta MVPA de Princeton. Esse é, também, o motivo de a interface e comentários no próprio código desenvolvidos estarem em inglês. A ferramenta é disponibilizada para domínio público, mas como a MPVA é em inglês, foi mantida a linguagem para compatibilidade.

1.4 ESTRUTURA DO TRABALHO

Este texto está organizado em capítulos. O capítulo 2 apresenta o referencial teórico que fundamenta o escopo e aplicação da ferramenta modelada como

resultado deste trabalho. As tecnologias e ferramentas utilizadas e o método empregado são apresentadas no Capítulo 3 e os resultados obtidos estão no Capítulo 4. O capítulo é finalizado com as conclusões.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta o referencial teórico. Os assuntos que fundamentam o trabalho são: imagem de ressonância magnética funcional, análise de padrões de multi-voxel e aprendizagem de máquinas.

2.1 fMRI – IMAGEM DE RESSONÂNCIA MAGNÉTICA FUNCIONAL

A mais essencial questão em neurociência cognitiva trata da questão de representação (NORMAN *et al*. 2006, p. 424): qual informação é representada em diferentes estruturas cerebrais; como a informação é representada; e como a informação é transformada nos diferentes estágios de processamento. Para esses autores, imagem de ressonância magnética, ou fMRI, constitui uma ferramenta que oferece recursos significativos para lidar com essas questões. E, ainda, para eles, com uso de fMRI enquanto um sujeito realiza uma tarefa cognitiva, é possível obter estimativas do fluxo sanguíneo local a partir de dezenas e centenas de regiões neuro-anatômicas distintas, em uma questão de segundos.

fMRI é uma técnica usada para obter imagens tridimensionais relacionadas à atividade cerebral. De maneira mais precisa fMRI mede a taxa de hemoglobina oxigenada para hemoglobina desoxigenada no sangue em relação a uma linha base de controle, sendo utilizado como indicador de atividade cerebral (MITCHELL, 2004). fMRI que melhora significativamente a habilidade humana de observar correlações de atividades cerebrais em humanos com alta resolução espacial (alguns milímetros) em todo o cérebro (MITCHELL *et al*., 2004).

Para Song, Iordanescu e Wyrwicz (2007), fMRI é uma ferramenta eficiente para estudo não invasivo da atividade cerebral em resposta a diferentes estímulos.

Uma pequena porção de dados de uma fMRI é ilustrada na Figura 1. Essa Figura mostra os dados coletados de 15 segundos de intervalos durante os quais o sujeito lia uma palavra, decidindo se era um nome ou um verbo (no teste, era um verbo) e aguardando por uma nova palavra (MITCHELL *et al*., 2004).

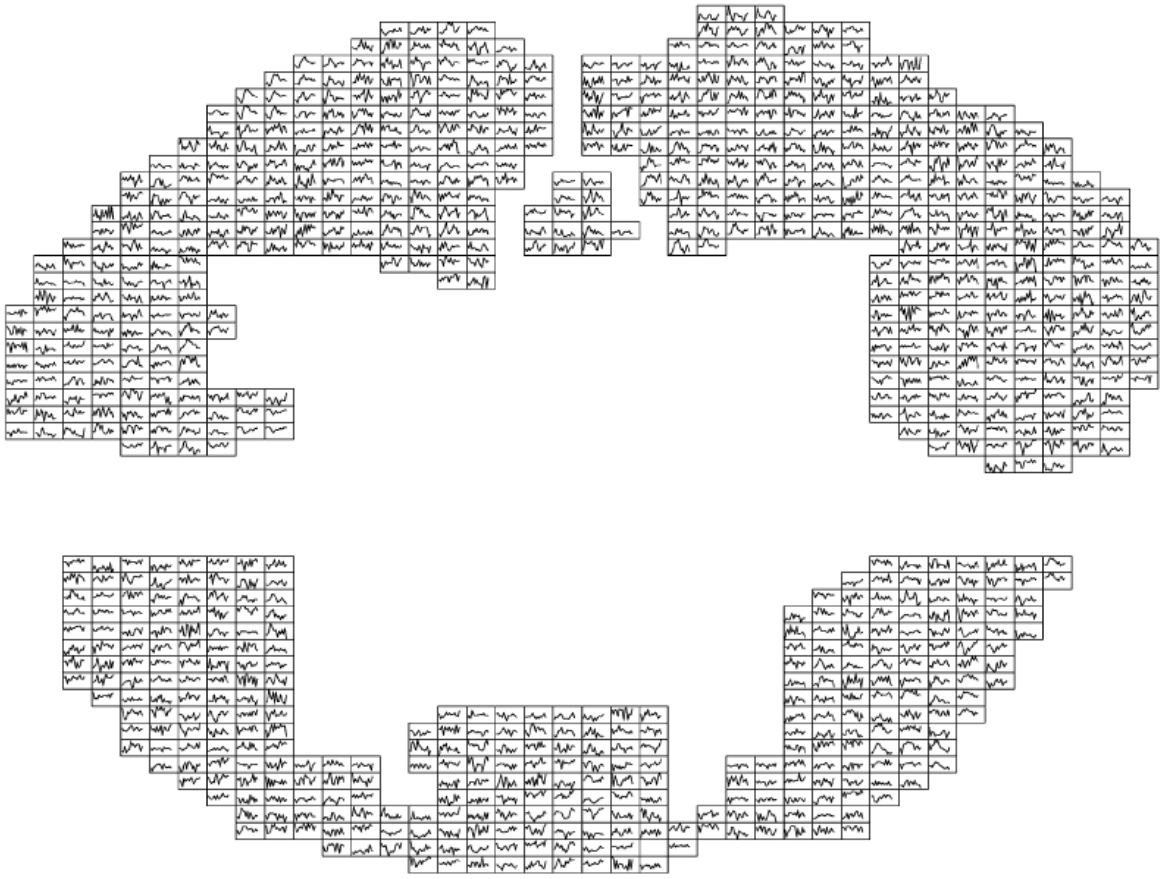

Figura 1 – Dados típicos de fMRI
Fonte: Mitchell *et al*. (2004, p. 148).

O modo de funcionamento dessa técnica é por meio de detecção das mudanças no nível de oxigenação do sangue e fluxos que ocorrem em resposta à atividade neural. Uma região do cérebro que está mais ativa consome mais oxigênio. O cérebro detecta esse aumento da demanda por oxigênio e dilata os vasos sanguíneos que irrigam a região de maior atividade. fMRI pode ser usada para produzir mapas de ativação mostrando quais partes do cérebro estão ativas sob certas condições ou processos mentais.

fMRI não deve ser confundida com o seu homólogo estrutural ou Imagem de Ressonância Magnética (*Magnetic Resonance Imaging* (MRI)) que fornece uma imagem da organização estrutural do cérebro, na qual cada tipo de tecido é representado por um valor de contraste diferente na imagem em três dimensões (3D). MRI proporciona uma única imagem de alta resolução obtida em um longo período de tempo. fMRI, por outro lado, proporciona uma série de imagens obtidas a cada 2 ou 3 segundos – um arquivo em quatro dimensões (4D) e oferece não uma

imagem anatômica, mas funcional, ou seja, uma imagem que mostra a atividade cerebral de uma região em vez de como essa região se parece.

2.1.1 Funcionamento da Imagem de Ressonância Magnética Funcional

Um scanner de ressonância magnética consiste em uma caixa cilíndrica envolvendo um eletroímã muito potente. Um scanner de pesquisa típico tem uma força de campo de 1,5-3 tesla(T), mais de 30.000 vezes maior do que o campo magnético natural da Terra (BUXTON, 2009). O campo magnético dentro do scanner afeta a orientação da rotação magnética dos átomos, particularmente de átomos de hidrogênio em água. Sob condições normais, os núcleos atômicos estão orientados aleatoriamente, mas sob a influência de um campo magnético os núcleos ficam alinhados com a direção do campo. Quanto maior o campo magnético maior o grau de alinhamento dos átomos. Quando todos os átomos estão apontando na mesma direção, os minúsculos sinais magnéticos de núcleos individuais se somam de forma coerente resultando em um sinal grande o suficiente para ser medido. Ressonância magnética estrutural funciona devido aos sinais dos núcleos de hidrogênio variar em força dependendo do que esta ao seu redor. Assim, proporcionando uma forma de discriminar entre a massa cinzenta, massa branca e o líquido vertebral cerebral – ou líquido cefalorraquidiano, em imagens estruturais do cérebro (MARQUES *et al*., 2009).

Oxigênio é fornecido para os neurônios por meio da hemoglobina nos glóbulos vermelhos capilares. Quando a atividade neural aumenta há um aumento da demanda de oxigênio e a resposta local é a dilatação dos vasos sanguíneos e aumento do fluxo de sangue para as regiões de maior atividade neural (MITCHELL *et al.*, 2004). Paradoxalmente, o nível de oxigênio no sangue aumenta quando uma região do cérebro é mais ativa, pois as regiões mais ativas necessitam de mais oxigênio aumentando por sua vez o fluxo sanguíneo dessas regiões.

Hemoglobina é diamagnética quando oxigenada, mas paramagnética quando desoxigenada. Essa diferença nas propriedades magnéticas leva à pequenas diferenças nos sinais de ressonância magnética do sangue em função do grau de oxigenação. Devido a oxigenação do sangue variar de acordo com os níveis de

atividade neural essas diferenças podem ser usadas para detectar a atividade cerebral (MAHMOUDI *et al.*, 2012). Portanto, essa forma de imagem de ressonância magnética é conhecida como imagem dependente do nível de oxigenação do sangue, *Blood Oxygenation Level Dependente imaging* (BOLD) que é a base para fMRI (MITCHELL *et al.*, 2004).

2.1.2 Informações fornecidas pela MRI

A atividade cerebral é monitorada ao longo do tempo, que o indivíduo está na máquina de ressonância magnética, conforme as mudanças nos sinais devido a flutuações do nível de oxigênio em todo o cérebro. Esse padrão de ativação pode mudar em consequência de vários fatores, como por exemplo, determinada tarefa que está sendo realizada pelo indivíduo, um pensamento que o indivíduo pode estar tendo, ou com base em condições neurológicas adquiridas.

fMRI fornece um mapa das regiões que estão menos ou mais ativas devido a determinadas condições experimentais. Por exemplo, uma pessoa lendo enquanto no scanner mostraria um aumento de atividade no lobo occipital, onde as informações visuais são processadas e também, na área de Wernicke, onde a linguagem é decifrada.

A Figura 2 apresenta uma imagem figurativa do cérebro, destacando-se a região do lobo frontal (TROCOLI, 2009).

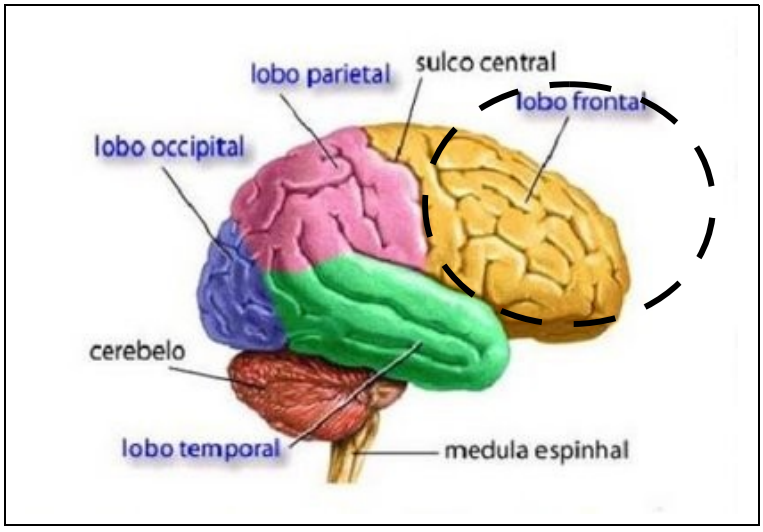

Figura 2 – Lobo frontal
Fonte: Trocoli (2009).

Na Figura 3 está uma imagem com a área de Wernicke destacada e seu relacionamento com a área de Broca (SCHUNEMANN, 2014).

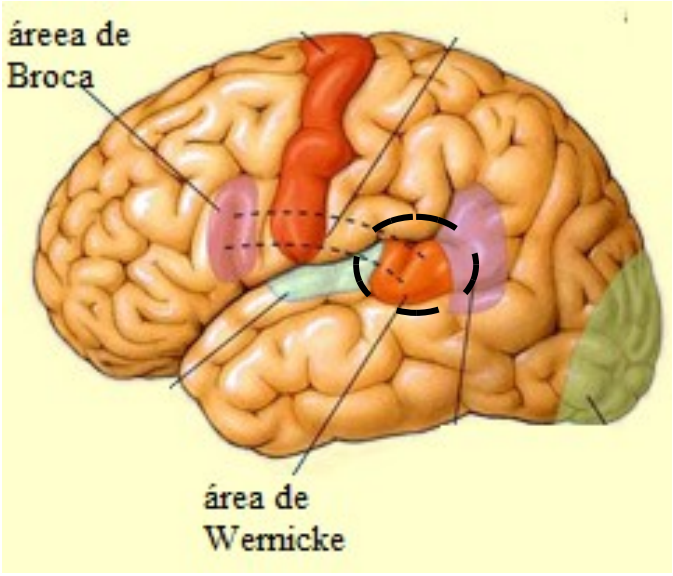

Figura 3 – Área de Wernicke
Fonte: Schunemann (2014).

Essas regiões (lobo frontal e área de Wernicke) são áreas de associação visual de sensitiva a fala, como indicado na Figura 4 (THOR, 2010).

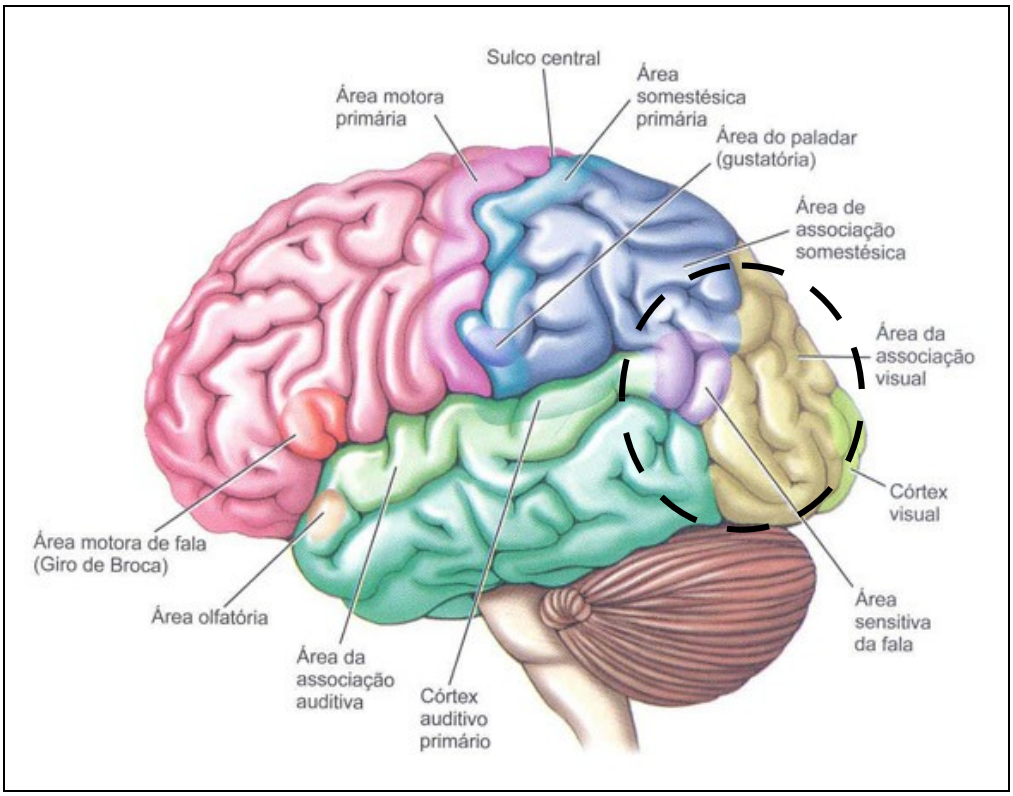

Figura 4 – Áreas de associação visual e sensitiva a fala
Fonte: Thor (2010).

fMRI gera uma representação dinâmica da atividade cerebral ao longo do tempo. Isto permite descobrir quais partes do cérebro estão ativas, como essa ativação muda ao longo do tempo e como ela se relaciona com a atividade de outras regiões. Permitindo, assim, deduzir quais regiões trabalham juntas e construir redes funcionais que são ativadas com propósitos específicos.

2.1.3 Arquivo de dados gerado pela por fMRI

Como explicado anteriormente, fMRI obtém dados da atividade cerebral a cada 2 ou 3 segundos. O tempo entre as aquisições de dados é chamado de Tempo de Repetição (TR). Cada mapeamento consiste em uma matriz de 3 dimensões chamada de um volume e representa uma imagem instantânea do estado de atividade cerebral em um momento específico de tempo. Cada plano da matriz é chamado de um pedaço e cada elemento representa um *pixel* na imagem daquele pedaço. Isto é o que médicos geralmente olham e como são representados nos meios de comunicação.

Quando esses pedaços são empilhando juntos, esses *pixels* duas dimensões (2D) se tornam tridimensionais já que o espaço entre esses pedaços precisa ser preenchido. Esses *pixels* cúbicos são chamados *voxels*. A próxima sessão de digitalização traz informações sobre o estado do cérebro de um TR posterior, e assim por diante. Desta forma, uma sessão de ressonância (aquisição de dados) de 5 minutos com um TR de 2 segundos é como um filme das mudanças de ativação do cérebro durante os 5 minutos com uma taxa de amostragem de 0,5 imagens por segundo.

A Figura 5 ilustra o processo da terminologia de fMRI, vinculando indivíduos, sessões, *runs*, *slices* e *voxels* associados ao tempo de repetição que é necessário para que um volume seja obtido.

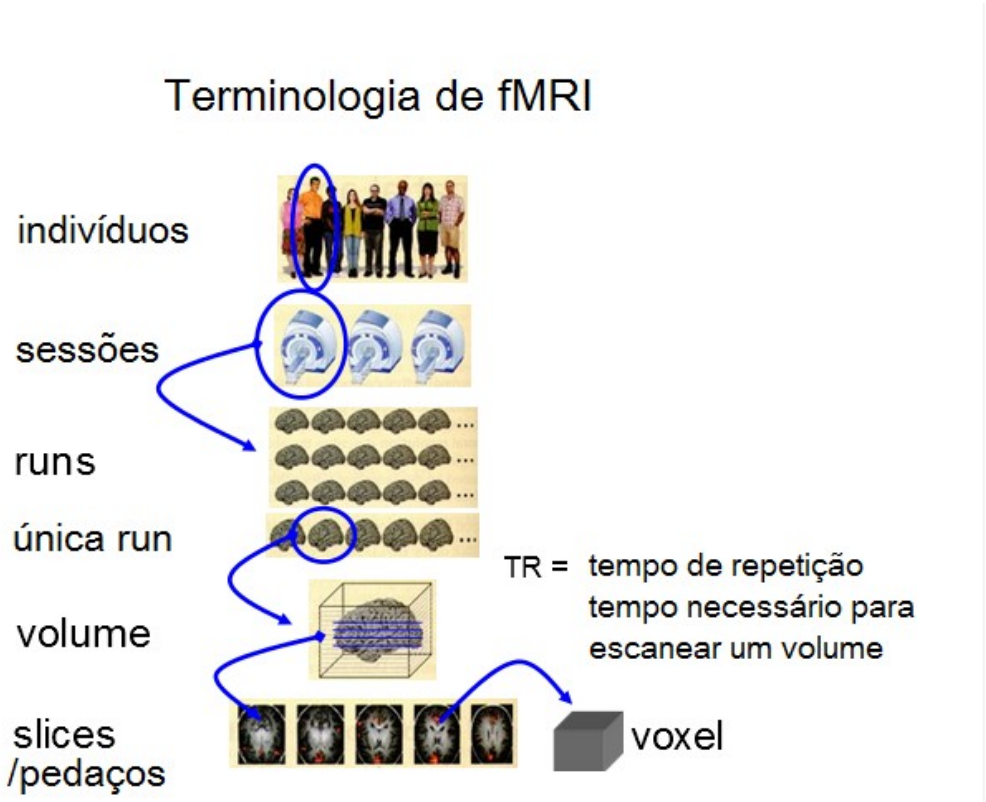

Figura 5 – Terminologia fMRI

Fonte: Stephan (2009, p. 4).

A grande quantidade de dados (geralmente vários gigabytes) e o alto nível de ruído inerente aos dados obtidos com fMRI colocam um desafio para pesquisadores interessados em minerar esses conjuntos de dados para obter informações sobre processos cognitivos (NORMAN *et al*., 2006). Mitchell et al. (2004) destacam que 20 minutos de sessão fMRI com um único sujeito humano produz uma série de imagens do cérebro cada uma contendo aproximadamente 15000 *voxels*, produzindo dezenas de milhões de dados para observação.

Norman *et al*. (2006) destacam que, tradicionalmente, os métodos de análise de fMRI tem focado na caracterização do relacionamento entre variáveis cognitivas e *voxels* cerebrais individuais (*pixels* volumétricos). Embora, ainda de acordo com esses autores, essa abordagem seja muito produtiva há limites no que pode ser aprendizado sobre estados cognitivos por meio da análise de *voxels* isolados.

Vários estudos têm demonstrado que fMRI contém informações que permitem o discernimento entre estados mentais e condições neurológicas. Descobrir e classificar estas diferenças pode permitir “ler mentes” e construir um software que permita diagnosticá-las automaticamente de forma a auxiliar médicos na tomada de

decisões. Além disso, o desenvolvimento desse tipo de software tem sido uma área de pesquisa de grande atividade nos últimos anos. *Multi-Voxel Pattern Analysis* é um desses métodos e se refere à ideia de aplicar métodos multivariados a dados de fMRI, permitindo analisar diversos *voxels* simultaneamente.

2.2 MULTI-VOXEL PATTERN ANALYSIS (MVPA)

Multi-Voxel Pattern Analysis é uma técnica de *machine learning* que analisa e classifica padrões de ativação cerebrais. *Machine learning* significa construir um algoritmo computacional para melhorar o desempenho por meio da experiência ou exemplos (conjuntos de treinamento) (VIDHATE; KULKARNI, 2012).

MVPA envolve buscar padrões espaciais altamente reproduzíveis para atividade que se diferencia por meio de condições experimentais (MAHMOUDI *et al*., 2014). MVPA é considerada um problema de classificação supervisionado no qual um classificador tenta capturar os relacionamentos entre padrões espaciais de atividade fMRI e condições experimentais (DAVATZIKOS *et al*., 2005).

Esse processo ocorre em duas etapas, a primeira etapa de treinamento e a segunda etapa de testes. Na etapa de treinamento, MVPA primeiramente “aprende” os diferentes padrões de ativações cerebrais (entrada) associadas às condições experimentais (alvo), então esse conhecimento é usado para prever a categoria de um novo conjunto de dados na etapa de testes. O resultado é uma pontuação com o nível de precisão calculado para estimar o desempenho do classificador.

Benefícios da MVPA (NORMAN, 2006):

a) Detecção mais sensitiva de estados cognitivos;
b) Caracterização da estrutura do código neural;
c) Relacionar atividade cerebral ao comportamento.

2.2.1 Obtençao de dados com MVPA

Supondo que existam duas condições experimentais: a primeira é quando o indivíduo está pensando em um cavalo, por exemplo, a segunda é quando o

indivíduo está pensando em um bolo de chocolate. A partir daí o indivíduo é colocado no scanner para uma sessão de fMRI que consiste em três subsessões de 5 minutos (também chamadas de *runs*). Durante a primeira *run*, é solicitado ao indivíduo que pense em um cavalo, já na segunda *run* o indivíduo deve pensar em um bolo de chocolate e na terceira *run* é solicitado que o indivíduo tente não pensar em nada. A terceira condição é chamada de estado de repouso e usada como uma forma de controle. Esse experimento é repetido com vários indivíduos para que se possa coletar um tamanho considerável de amostras.

Posteriormente, esses dados são separados em um conjunto de treinamento e um conjunto de testes. Esses dados de entrada serão inseridos no programa MVPA com rótulos correspondentes as tarefas realizadas. O programa então será responsável por analisar todos os dados associados com a condição “cavalo” juntamente com os outros dados e em seguida classificar os padrões em um modelo com base no que foi aprendido. Logo após, o programa atuará da mesma forma para os dois outros rótulos.

Na sequência, durante a fase de testes, todos os dados de indivíduos que não foram utilizados na fase de treinamento serão alimentados ao MVPA para que possa ser previsto no que o indivíduo estava pensando: cavalo, bolo de chocolate ou em “nada”. Se a partir disso a máquina é capaz de prever com precisão o que o indivíduo estava pensando, então é dito que a(s) característica(s) responsável(is) por codificar este estado mental foram extraídas com sucesso. No entanto, se a máquina não tem uma boa precisão é possível que os dados estejam no chamado *overfitting*, não conseguindo captar quais eram as características relevantes, ou também pode simplesmente ser que não existam diferenças nos dados para diferenciar confiavelmente entre as condições.

2.2.2 Diagnóstico de condições neurológicas com o uso de MVPA

MVPA pode ser utilizada para diagnosticar pacientes. Apesar dos pacientes serem orientados a não pensar em nada durante o estado de repouso, é impossível parar de pensar. As pessoas relatam sonhar acordadas, pensando sobre o passado e o futuro e tendo pensamentos introspectivos enquanto no *scanner*.

Padrões de ativação também aparecem em regiões do cérebro responsáveis pela representação de si mesmo e de memória. É possível visualizar esta ativação entre os indivíduos. As regiões do cérebro associadas com as atividades do estado de repouso têm sido chamadas de modo de rede padrão (*default mode network*). Pesquisas têm mostrado que essa atividade do estado de repouso é um tanto quanto confusa em indivíduos com condições neurológicas devido ao seu efeito sobre a cognição. Por exemplo, caso pesquisadores quisessem construir uma ferramenta de diagnóstico para triagem de pacientes com doença de Alzheimer. Seria então utilizado um conjunto de dados do estado de repouso de pacientes com Alzheimer e pacientes saudáveis (o grupo de controle). Portanto, para esse propósito MVPA seria utilizado para prever se o paciente tem Alzheimer ou não, da mesma forma que MVPA é utilizado para prever o estado mental de um indivíduo.

2.3 MACHINE LEARNING

Aprendizado é a abordagem mais importante para um ser humano adquirir conhecimento e este é, também, a mais proeminente característica da inteligência humana (GAO, *et al*., 2013). Para esses autores, *machine learning* (aprendizado de máquina) visa investigar como simular a ação de aprendizado humano para propiciar inteligência computacional e é o aspecto central e fundamental da inteligência artificial. O objeto de estudo de aprendizado de máquina são as leis fundamentais que governam todos os processos de aprendizado, incluindo sistemas artificiais (computadores) e sistemas humanos (pessoas) (GAO, *et al*., 2013).

Para Wang, Ma e Zhou (2009), aprendizado de máquina é um assunto de estudo de como usar computadores para simular as atividades de aprendizado humano para estudar métodos de auto-melhoramento de computadores para obter novos conhecimentos e adicionar novas habilidades, identificar conhecimento existente e melhorar continuamente a aquisição e o aprendizado.

A Figura 5 apresenta um modelo básico do aprendizado de máquina.

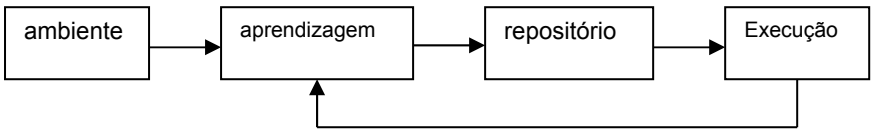

Figura 6 – Modelo básico do aprendizado de máquina
Fonte: traduzido de Wang, Ma e Zhou (2009, p. 1).

No processo de aprendizado, a qualidade da informação proveniente do ambiente para o sistema é um fator essencial.

Na representação da Figura 5:

a) Ambiente representa a origem da informação.
b) Aprendizagem é o processo que transforma a informação em conhecimento e armazena. Essa informação é obtida no ambiente externo, transformada em conhecimento que é armazenado no repositório.
c) Repositório armazena os princípios gerais que guiam a implementação da ação.

Execução é o processo de uso do conhecimento que está armazenado no repositório para realizar uma tarefa, fornecer retorno do processo de aprendizado ao realizar uma tarefa e guiar estudos futuros.

3 MATERIAIS E MÉTODO

Este capítulo apresenta a ferramenta e tecnologia e o método utilizados para a realização deste trabalho.

3.1 MATERIAIS

O Quadro 1 apresenta as ferramentas e as tecnologias que foram utilizadas para modelar e implementar o sistema.

Ferramenta / Tecnologia	**Versão**	**Referência**	**Finalidade**
MATLAB	2010b	http://www.princeton.edu/~bdsinger/pubs/polyn_et_al_mvpa_ohbm2005.pdf	Linguagem usada para construir a ferramenta/aplicação
Princeton MVPA	1.1	http://code.google.com/p/princeton-mvpa-toolbox/	Ferramenta na qual foram realizados ajustes e melhorias

Quadro 1 – Ferramentas e tecnologias utilizadas

3.1.1 MATLAB

MATrix LABoratory (MATLAB) é um software de programação interativa para computação cientifica (MATLAB, 2015a). É muito utilizado em diversos campos técnicos como análise de dados, resolução de problemas e realização de experimentos e para o desenvolvimento de algoritmos. As possibilidades de aplicação do MATLAB se relacionam a computação numérica, análise e visualização de dados, desenvolvimento de algoritmos e programação e desenvolvimento e distribuição de aplicações (MATLAB, 2015b). MATLAB é um ambiente interativo e uma linguagem de programação para computação técnica e científica em geral (BRAVO; ALBUQUERQUE, 2015).

MATLAB permite adicionar extensões (*toolboxes)* de forma a realizar outros tipos de operações, além de cálculo numérico, como, por exemplo, o módulo *Simulink* que é um ambiente de simulação baseado em diagrama de blocos e plataforma para *Model-Based Design* para sistemas dinâmicos e incorporados.

MATLAB também permite a interação com programas escritos em outras linguagens, incluindo C, C ++, Java, Python e Fortran.

3.1.2 Ferramenta Multi-Voxel Pattern Analysis (MVPA)

A ferramenta para *Multi-Voxel Pattern Analysis* desenvolvida em MATLAB pela Universidade de Princeton permite aos seus usuários realizar *Machine Learning Classification* tendo como base padrões de ativação cerebral obtidos a partir de dados de *fMRI* (DETRE *et al*., 2006).

Essa ferramenta foi originalmente desenvolvida para ser capaz de deduzir que tipo de imagem um indivíduo estava olhando enquanto ele estivesse dentro de um scanner com base no estado neural (sinal *BOLD).* As diferentes categorias foram rostos, casas, gatos, garrafas, tesouras, sapatos, cadeiras e imagens caracterizadas como ruídos (como as que representam canais de televisão fora de sintonia, por exemplo) O conjunto de dados foi construído com dados vindo de um estudo publicado anteriormente (HAXBY *et al*., 2001).

A ferramenta funciona com o treinamento de um classificador para prever qual a categoria de uma imagem que o indivíduo está olhando durante cada experimento. Porém, para que isso seja possível são necessários certos itens como, por exemplo, um padrão de classificação, um conjunto de regressores, um seletor e uma máscara. O padrão é a informação que se deseja classificar e é formado pelos dados *BOLD* vindos de exames ressonância magnética funcional. O conjunto de regressores é representado por uma matriz de condições que indica qual condição está presente em determinado momento durante o experimento. Isto é o que permite os classificadores saberem quais padrões estão associados com quais condições. O seletor é um vetor que possibilita a máquina saber a localização da fragmentação dos dados para as etapas de treinamento e testes. A máscara é uma matriz 3D – também chamada *spatial filter* – que permitirá que somente os dados localizados nas regiões cerebrais de interesse irão para análise.

Os passos para realizar a análise envolvem, primeiramente, carregar a máscara que será utilizada. A seguir, usa-se essa máscara para importar o padrão e ao mesmo tempo as partes do padrão que não serão utilizadas são retiradas de

acordo com a máscara. Os *condition regressors* e o seletor precisam ser personalizados de acordo com cada experimento realizado. Eles podem ser carregados antes ou depois da máscara e dos padrões.

Após todas essas informações terem sido carregadas na ferramenta, o pré-processamento dos dados pode, então, ser realizado. Essa etapa ajuda no desempenho da classificação dos dados durante o *ZScore. ZScore* é um meio utilizado para transformar os dados de forma que a média se torna zero e a variância da amostra é representada por uma unidade igual a um desvio padrão de distância da média. O *ZScore* é realizado separadamente em cada fragmento de dados de acordo com o seletor.

Depois de realizar o *ZScore*, os índices de *cross-validation* são definidos. A forma de classificação realizada por essa ferramenta é por meio de um método do tipo n-menos-um *cross-validation*. Ele funciona quebrando os dados em fragmentos com base no seletor e, então, retira um desses fragmentos que será utilizado para teste. A fase de treinamento é realizada nos fragmentos restantes e a fase de testes será realizada no fragmento que foi retirado anteriormente. Esse procedimento é repetido até que cada um dos fragmentos tenha sido utilizado na fase de testes. Como resultado, se existem n fragmentos, cada fragmento será usado n-menos-um vezes para a fase de treinamento e uma vez para a fase de testes.

O passo seguinte é utilizar ANOVA (*Analise of Variance*) para selecionar os *voxels* que mudaram significativamente entre os fragmentos. Esta etapa é chamada de *feature selection* e é usada para permitir que o algoritmo de *Machine Learning* se concentre nas características que são relevantes.

Finalmente, a classificação é realizada usando *backpropagation*, um tipo supervisionado de *Machine Learning* baseado em redes neurais. Esse método compara a saída desejada com a entrada e saída iniciais, e se ajusta até que o erro seja minimizado por gradiente descendente.

Desde o seu lançamento, a ferramenta MVPA desenvolvida pela universidade de Princeton tem sido usada para detectar memórias (RISSMAN *et al.*, 2010), para prever a gravidade de sintomas (COUTANCHE *et al*., 2011), para estudar os déficits de representações mentais em esquizofrenia (YOON, *et al*., 2008) e para decodificar a direção do sentido auditivo em pacientes cegos (WOLBERS *et al.*, 2011).

3.2 MÉTODO

A seguir estão descritas as etapas para o desenvolvimento do trabalho.

a) Estudo do referencial teórico

Estudo do referencial teórico foi realizado para o entendimento dos conceitos envolvidos, das funcionalidades já providas pela ferramenta Princeton MVPA e dos requisitos acrescentados a essa ferramenta. Como resultado do estudo do referencial teórico foi elaborado o Capítulo 2 deste trabalho.

b) Estudo da ferramenta Princeton MVPA

O estudo da ferramenta teve como objetivo o entendimento das suas funcionalidades e da forma de interação que poderia ser proposta visando facilitar a usabilidade do aplicativo.

c) Estudo de MATLAB

O estudo da linguagem utilizada por MATLAB teve como objetivo o aprendizado da sintaxe da linguagem a ser utilizada para os complementos a serem implementados na ferramenta Princeton MVPA.

d) Implementação da interface gráfica

A implementação de uma interface gráfica para a interação com a ferramenta MATLAB visa melhorar a usabilidade, especialmente para usuários não familiarizados com o uso da interface de interação no formato caractere oferecida por MATLAB. No desenvolvimento da interface, componentes serão testados visando facilitar a interação.

e) Implementação de melhorias no algoritmo

Melhorias serão implementadas no algoritmo de aprendizagem de máquinas que faz o processamento dos *voxels*. Outros métodos estatísticos serão avaliados.

4 RESULTADO

Este capítulo apresenta o resultado do trabalho em termos da definição do aplicativo.

4.1 ESCOPO DO SISTEMA

Multi-Voxel Pattern Analysis é uma técnica na qual aprendizagem de máquina é aplicada em padrões de ativação cerebral para classificar pensamentos de pacientes baseados no que já foi previamente ensinado para a máquina. Um classificador de padrões é treinado para associar a correlação entre a atividade de um conjunto de regiões cerebrais com uma condição experimental – por exemplo, se o individuo está olhando para rostos ou pensando em bananas. A máquina é alimentada com diferentes conjuntos de dados e suporia qual era a condição que desencadeou o padrão de ativação cerebral sendo considerado. O objetivo é ter uma ferramenta para ajudar a investigar como o cérebro codifica informações e o grau de consistência entre os indivíduos.

A técnica também pode ser expandida para investigar os padrões de ativação cerebral quando um indivíduo está em repouso ou não está envolvido em nenhuma atividade em particular enquanto está em uma máquina de ressonância magnética funcional. O cérebro é dito estar em um estado de repouso e os padrões de atividade cerebral têm apresentado diferenças entre um individuo considerado saudável e pacientes com certas condições neurológicas (BUCKNER *et al*., 2008). MVPA também pode ser usada para classificar indivíduos baseados no diagnóstico do seu estado de repouso, nesse caso, se implementada essa opção seria utilizada para investigar o seu potencial como uma ferramenta de diagnósticos.

A Figura 7 apresenta uma visão geral da proposta deste trabalho inserida no contexto do sistema já desenvolvido pela Universidade de Princeton, denominado Princeton MVPA.

Figura 7 – Visão geral da MPVA

Legenda da Figura 7: cor vermelha significa implementação importante, cor azul implementação secundária e cor verde implementação opcional.

De acordo com a representação da Figura 6, no desenvolvimento serão realizados:

a) Criação de uma interface: para a ferramenta Princeton MVPA para permitir que usuários não familiarizados com MATLAB possam utilizá-la.

b) Criação de um filtro espacial na interface: criar um repositório no qual os filtros predefinidos serão armazenados e o usuário poderá selecionar aqueles que ele deseja para a análise. Esse filtro espacial permite que os *voxels* da área desejada façam parte da análise.

c) Especificadores de condições (condição regressora): permitir que o usuário defina interativamente enquanto visualiza os dados em vez de criar uma matriz de

regressão. Condição regressora é um arquivo que especifica quais condições experimentais estão ativas em determinado momento.

d) Seletor de características: atualmente está sendo utilizada a ANOVA, que tem sido criticada por predispor os resultados de aprendizagem de máquina favorecendo uma maior precisão, uma forma chamada de “dupla imersão”. O objetivo é mudar do método ANOVA para outra forma de seleção de característica mais adequada à aprendizagem de máquina.

4.2 APRESENTAÇÃO DO SISTEMA

A Figura 8 apresenta a tela principal da ferramenta desenvolvida em MATLAB.

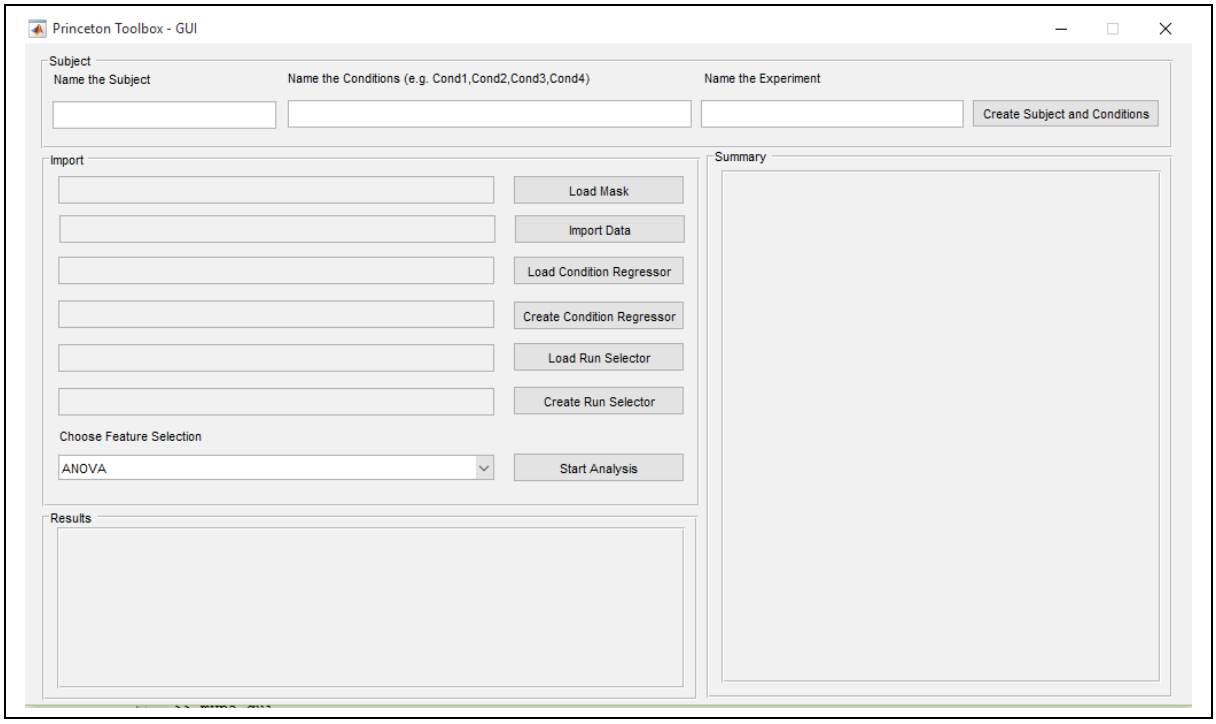

Figura 8 – Tela principal do sistema

O leiaute do sistema é simples e dividido em quatro partes para auxiliar o usuário na sua utilização. Na parte superior está a criação do indivíduo, ou seja, a parte inicial da utilização da ferramenta. Logo abaixo está a localização para a entrada de dados e escolha do método de *Feature Selection* para iniciar a análise dos dados. Abaixo da parte de entrada de dados está a apresentação do resultado

da análise, com a precisão da mesma. Por último, no lado direito da tela há um quadro para trazer o resumo de tudo que está sendo feito na tela, que são as entradas de dados ou objetos criados para auxiliar na análise.

Na Figura 9, o usuário informará o nome do individuo, as condições utilizadas no experimento e qual o nome do experimento que está sendo realizado. Um experimento se refere a uma execução dos aplicativos com dados de ressonância magnética obtidos de um indivíduo. Após isso, basta clicar no botão de criação do individuo e suas condições para dar início ao experimento.

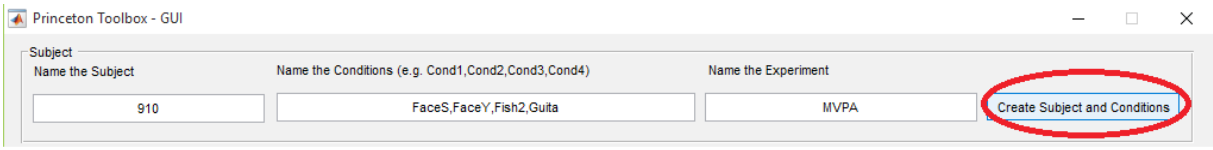

Figura 9 – Criação do experimento

Na Figura 10 é apresentada a entrada de dados na qual o usuário buscará os diretórios que possuem as informações do indivíduo armazenadas. Ele carregará informações de máscara, os dados vindos da fMRI, o *Condition Regressor* e o *Run Selector.* Ainda nessa parte é possível criar o *Condition Regressor* e também o *Run Selector*, como pode ser visualizado nas Figuras 10 e 12. Para a criação do *Condition Regressor* e *Run Selection* devem ser especificados respectivamente, o número de condições ou número de *runs* e o número de *timepoints*. Após isso, será criada uma outra tela para que o usuário informe quando determinada condição ou *run* estava ativa e salvar essa matriz ou vetor criados em um local escolhido por ele mesmo. É importante ressaltar que o número de *timepoints* informado deve ser a mesma dimensão de colunas da matriz de dados provenientes da fMRI. Após carregar todos os dados na tela, o usuário deverá escolher entre os dos métodos de *Feature Selection* e iniciar a análise dos dados.

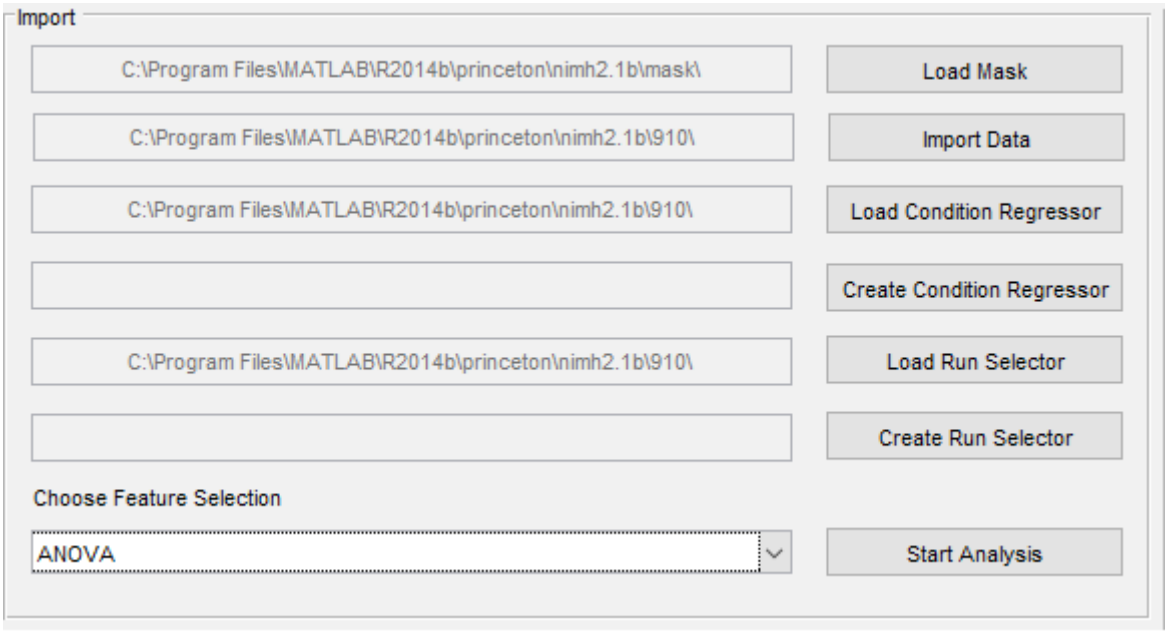

Figura 10 – Importação/entrada de dados

A Figura 11 apresenta a tela para a entrada do número de condições para o experimento e de *timepoints*.

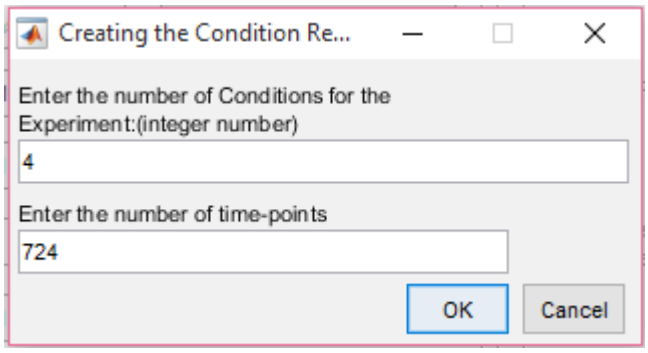

Figura 11 – Criação do *condition regressor*

A Figura 12 apresenta a tela para entrada de dados referentes à especificação do *condition regressor*.

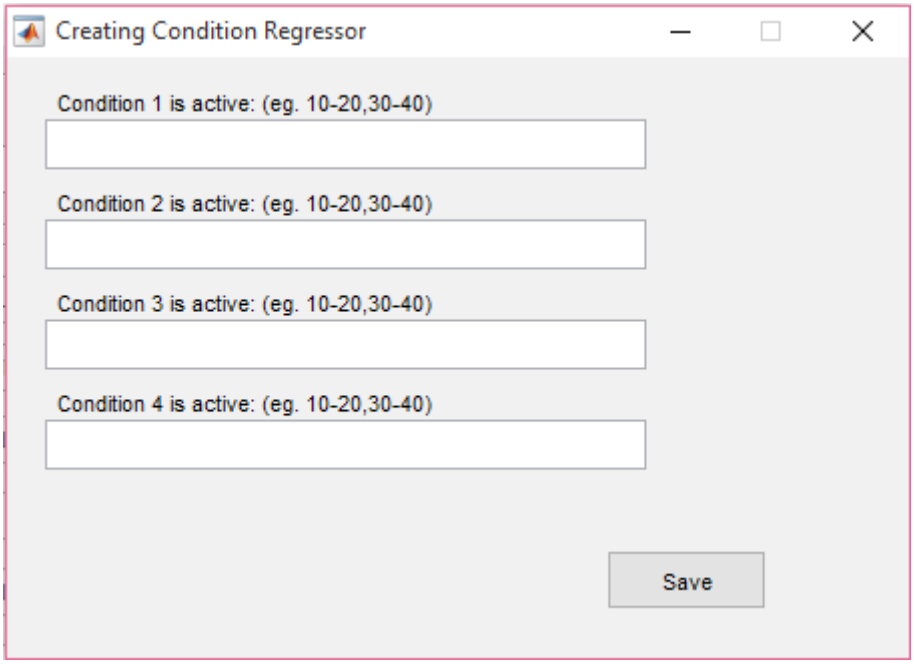

Figura 12 – Tela de especificação dos *condition regressor*

Na Figura 13, é possível verificar a precisão do resultado do desempenho do algoritmo de acordo com os dados na tela e o método de *Feature Selection* escolhidos.

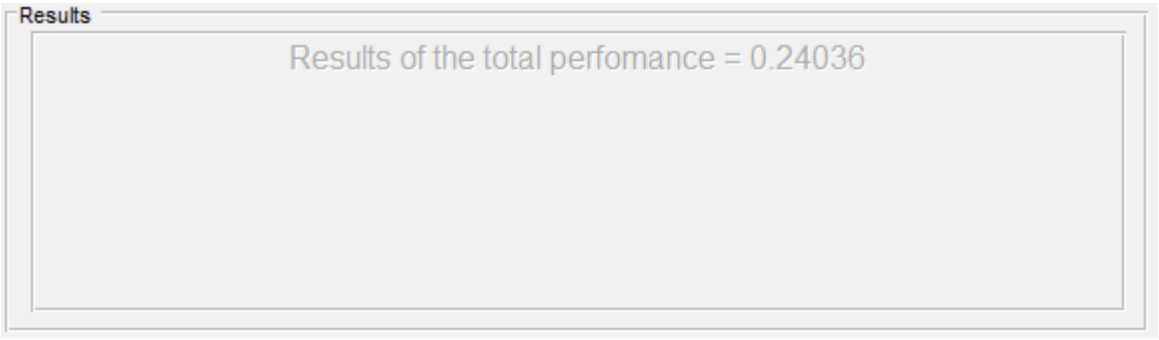

Figura 13 – Resultado do desempenho da análise de dados

Por último, a Figura 14 apresenta resumidamente todas as informações de entradas na tela. Desde os dados escolhidos pelo usuário, assim como os objetos criados para auxiliar na análise.

Figura 14 – Resumo dos dados

4.3 IMPLEMENTAÇÃO DO SISTEMA

Na Listagem 1 é possível verificar como é desenvolvido o conteúdo do painel que traz as informações de resumo de todos os dados de entrada na tela.

```
function [result] = mySummarizeFn(subj)
    diary(fullfile(tempdir, 'mySubject.txt'))
    summarize(subj);
    diary off

    fid = fopen(fullfile(tempdir,'mySubject.txt'), 'r');
    result = textscan(fid, '%s', 'Delimiter', '\n');
    fclose(fid);
    result = result{1};
    result=result(~cellfun('isempty',result));
    delete(fullfile(tempdir, 'mySubject.txt'));
```

Listagem 1 – Código para trazer os dados do *summary*

A função criada é chamada em vários locais do código, pois nas entradas de dados os dados do *summary* devem ser atualizados. Essa função retorna um *array* de *strings* para ser inserido no componente de texto (text). O comando *diary* criará um arquivo de texto no diretório temporário da máquina, salvando todas as informações que são apresentadas na tela de comandos do MATLAB já que a função original da ferramenta de Princeton retorna os dados dentro dessa tela. Após esse arquivo de texto ter sido criado é possível fazer a abertura do mesmo com o comando *fopen* informando os argumentos de local do arquivo e nesse caso leitura 'r'. Os dados lidos são armazenados na variável de retorno da função sendo os mesmos delimitados por linhas. O arquivo de texto é então fechado, a variável de retorno é transformada em uma *array* de *strings* e caso houver espaços em branco dentro desse *array* eles são então removidos. Por último, o arquivo é excluído do diretório temporário.

Na Listagem 2 é apresentada como é realizada a criação da matriz de Condition Regressor. O código a seguir é executado no botão de criação de Condition Regressor, sendo que o primeiro passo é criar uma tela de entrada de dados pelo usuário para serem informados o número de condições e o número total de *timepoints*, sendo estes salvos em uma variável chamada sCondReg.

```
% --- Executes on button press in btnCreateCR.
function btnCreateCR_Callback(hObject, eventdata, handles)
% hObject    handle to btnCreateCR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Ask the user to enter two integer numbers.
sCondReg = inputdlg({'Enter the number of Conditions for the Experiment:
(integer number)', 'Enter the number of time-points'},...
                    'Creating the Condition Regressor', [1 50; 1 40]);
if isempty(sCondReg),return,end; % Bail out if they clicked Cancel.

% Round to nearest integer in case they entered a floating point number.
integerValue1 = round(str2double(sCondReg{1}));
integerValue2 = round(str2double(sCondReg{2}));

%Check if the 1-of-n pattern matrix is the same as the entered value
%for the timepoints for the new Condition Regressor
outCheck = sanity_check(integerValue2);
if(outCheck == true)
    % Check for a valid integer.
    if (isnan(integerValue1) || isnan(integerValue2))
        % They didn't enter a number.
        % They clicked Cancel, or entered a character, symbols, or
something else not allowed.
        message = sprintf('Please enter integer values for the conditions
and time-points.');
        uiwait(warndlg(message));
```

```
    else
       create_condreg(integerValue1, integerValue2);
        set(handles.btnBrowseCR,'Enable','off');
    end
else
message = sprintf('Please enter the same dimension for the time-points.');
uiwait(warndlg(message));

end
```

Listagem 2 – Código de execução do botão de criação de *condition regressor*

Após salvos, uma verificação é feita para saber se o usuário cancelou a entrada de dados e então o restante do código não é executado ou então, as posições do *array* são separadas em duas novas variáveis e uma nova verificação é realizada para saber se o número de *timepoints* informado é o mesmo que o número de colunas da matriz de dados importada pelo usuário (Listagem 3). Caso o retorno dessa validação seja verdadeiro, é verificado se as entradas de dados pelo usuário são números inteiros ou não. Caso não sejam valores numéricos inteiros, uma mensagem é apresentada ao usuário para que ele verifique os dados informados. Se a entrada corresponde ao tipo de dado esperado, a função de criação de condições é chamada e o botão de carregar os dados de *Condition Regressor* é desabilitado. No entanto, se o retorno da validação for falso, o usuário é informado que o número de *timepoints* deve ser o mesmo que o número de colunas dos dados carregados.

```
function [output] = sanity_check(regsSel)

% Check that the timepoints we're going to create for the
% Run Selector and Condition Regressor have the same dimension
% as the timepoints for the pattern matrix
global patName;
global subj;

pat  = get_mat(subj,'pattern', patName);

output = true;

if size(pat,2) ~= regsSel
  message = sprintf('Wrong number of timepoints.');
  uiwait(errordlg(message, 'Error Creating Function', 'modal'));
  output = false;
end
```

Listagem 3 – Código de verificação de *timepoints*

Na Listagem 3 é possível observar que a verificação de *timepoints* é uma função usada tanto para a criação do *Condition Regressor* quanto do *Run Selector*.

Nela os dados de criação do experimento e as informações carregadas são recuperados por meio das variáveis globais em que ambos estão armazenados e a partir delas são recuperadas as dimensões da matriz de dados. Com essas dimensões recuperadas é verificado, então, se o número de colunas dos dados é o mesmo que o número de *timepoints* informado pelo usuário e, caso não seja, uma mensagem de erro é apresentada ao usuário informando que os números não são os mesmos e o retorno da função é falso.

Ainda no código de criação de *Condition Regressor*, está a Listagem 4. Essa listagem apresenta como é feita a criação de uma tela no MATLAB de forma programada sem o auxílio do *Guide, a* ferramenta de auxílio para criação de telas em que os componentes são escolhidos e arrastados dentro de uma tela. A função recebe dois parâmetros: o número de condições e o número de *timepoints*. A partir desses parâmetros uma estrutura *h* é criada com o mesmo número de condições informados e dentro dessa estrutura são criados os campos de 'staticText' e 'editText' também com o mesmo número que o de condições.

O comando *figure* criará uma tela *f* de acordo com as propriedades informadas como, por exemplo, posição (*Position*), nome (*Name)*, tipo da janela (*WindowStyle)*, visibilidade (*Visible*), entre outras. Duas variáveis para controlar a posição y de cada componente de texto estático (*staticText)* e texto (*editText)* são criadas e recebem um valor inicial que será alterado dentro do *for loop* ao final de cada iteração.

Uma matriz de zeros é criada para as dimensões de número de condições e *timepoints*. No *for loop* cada componente é adicionado dentro da posição *i* da estrutura *h* criada anteriormente e de acordo com o nome do componente é anexado à figura *f* cada um com suas propriedades. Após os componentes terem sido criados um botão é adicionado à figura *f* para salvar os dados de entrada pelo usuário e então a figura *f* se tornará visível.

```
function create_condreg(nConditions, nTimePoints)

h(nConditions) = struct('staticText', (nConditions), 'editText',
(nConditions));
f = figure('Position', [500 360 450 300],...
           'Name', 'Creating Condition Regressor',...
           'WindowStyle', 'modal',...
           'Visible', 'off',...
           'Resize', 'off',...
           'NumberTitle', 'off');
posT.y = 260;
pos.y = 245;

matCondReg = zeros(nConditions, nTimePoints);

for i=1:nConditions
    h(i).staticText = uicontrol(f, 'Style', 'text',...
                                   'String', sprintf('Condition %d is
active: (eg. 10-20,30-40)', i),...
                                   'Position', [25 posT.y 300 25],...
                                   'HorizontalAlignment', 'left');
    h(i).editText = uicontrol(f, 'Style', 'edit',...
                                 'Position', [20 pos.y 300 25]);
    pos.y = pos.y - 50;
    posT.y = posT.y - 50;
end
btnSave = uicontrol(f, 'Style', 'pushbutton',...
                       'String', 'Save',...
                       'Position', [300 pos.y-20 80 30],...
                       'Callback', {@btnSave_Callback, h, matCondReg});
f.Visible = 'on';
```

Listagem 4 – Criação da tela de condition regressor

Na Listagem 5 é apresentado o código para a chamada da função do botão de salvar criado anteriormente. Essa função tem os parâmetros do MATLAB e dois novos parâmetros *h* e *matCondReg*, que são respectivamente a estrutura e matriz de zeros criadas na Listagem 4. Uma estrutura (*retData*) no formato *cell* é criada com o mesmo tamanho da estrutura *h*.

No primeiro *for loop*, são salvos todos os dados vindos do *editText* da estrutura *h* dentro da variável *retData* cada um na respectiva posição *i*. Além disso a *String* do *editText* é recuperada pelo comando *get* e para salvar dentro de uma variável do formato *cell* é usada a conversão de *String* para *cell* com o comando *cellstr*. Uma nova variável (*splitData)* é criada para salvar os pontos iniciais e finais de cada condição após terem os seus delimitadores sido removidos. Essa variável tem o mesmo tamanho que o de *retData*.

No segundo *for loop,* o comando *textscan* percorre um *array* de caracteres, por isso da conversão de *cell* para *char*, e quando há delimitadores informados pela

propriedade *'Delimiter'* remove tais valores e separa o que vem antes e depois do delimitador. No terceiro *for loop*, é percorrido o tamanho da variável *splitData* e no quarto e último *for loop é* percorrido o tamanho de *splitData* na posição *m*. Nas variáveis de começo (*nStart)* e fim (*nEnd)* são recuperadas as posições informadas pelo usuário que foram separadas pelos delimitadores e então é realizada a substituição da matriz de zeros por uns de acordo com a posição inicial e final dentro de um *while*. Após isso são recuperados os dados que estão na tela principal do sistema com o comando de *guidata* e o nome da tela.

Com o comando de *uiputfile* é solicitado ao usuário que escolha o local no qual deseja salvar a matriz de *Condition Regressor* e qual o nome do arquivo. É feita a verificação para saber se o usuário informou o arquivo e o local onde o mesmo será salvo e caso ele não os tenha informado é retornado à tela para que a matriz possa ser salva. Se os dados foram informados, a matriz gerada é salda com o nome e no local informados com a ajuda do comando *save*.

Além disso, com os dados recuperados da tela principal é possível passar o local onde a matriz foi salva para o componente de texto e também é possível desabilitar o botão de criação de *Condition Regressor* e habilitar o botão de carregar onde está salvo o *Condition Regressor*.

Por fim, com o comando *close(gcf) é* fechada a figura atual do MATLAB, para entrada de dados do *Condition Regressor.*

```
function btnSave_Callback(hObject, eventdata, h, matCondReg)

retData = cell(1, length(h));

for i=1:length(h)
    retData(i) = cellstr(get(h(i).editText, 'String'));
end
splitData = cell(1,length(retData));
for j=1:length(retData)
    if(j > 1)
        [splitData] = [splitData; textscan(char(retData(j)), '%s',
'Delimiter', '-,')];
    else
        [splitData] = textscan(char(retData(j)), '%s', 'Delimiter',
'-,');
    end
end
for m=1:length(splitData)
    for n=1:+2:length(splitData{m})
        nStart = str2double(splitData{m}(n));
        nEnd = str2double(splitData{m}(n+1));
        while(nEnd >= nStart)
           matCondReg(m,nStart) = 1;
           nStart = nStart + 1;
        end
```

```
    end
end
handles=guidata(mvpa_gui);
[filename, path] = uiputfile('*.mat', 'Save Condition Regressor Matrix
as');
if isequal(filename,0) || isequal(path,0), return;
else
   save([path,filename],'matCondReg');
   set(handles.edtCreateCR, 'String', path);
   set(handles.btnCreateCR, 'Enable', 'Off');
   set(handles.btnBrowseCR, 'Enable', 'On');
close(gcf);
end
```

Listagem 5 – Código para criação da matriz de condition regressor

A Listagem 6 se refere ao que acontece após o usuário ter fornecido entrada em todos os dados e selecionado um método de *Feature Selection*. Ao clicar no botão de *'Start Analysis'*, o sistema iniciará a análise de dados para então retornar a precisão dos resultados. É recuperado qual foi o método selecionado pelo usuário que é salvo na variável *nFeature*. Além disso, são recuperadas as variáveis globais de nomes dados ao indivíduo, *pattern, condition regressor* e *run selector.*

O ponteiro do mouse é alterado para que o usuário do sistema saiba que a análise está em andamento e o ponteiro voltará ao seu estado inicial ao final da mesma. É iniciada então a análise fazendo o *zscore* do *pattern,* que são os dados vindos da fMRI, de acordo com o número de *runs*. Após isso, são criados os índices para o *n-minus-one* número de iterações de acordo com o número de *runs* para realizar o *cross-validation*. No caso estudado, *runs* são substituídos para blocos (*blocks*) para tentar implementar um melhor resultado no desempenho do algoritmo.

Antes de realizar a classificação é utilizada uma análise estatística para ajudar na classificação, o que no caso de uso de *Machine Learning* é realizado por *Feature Selection* que é responsável por verificar quais *voxels* são mais relevantes que outros. A maneira até então utilizada pela ferramenta era a ANOVA que diz qual a probabilidade de atividade que um determinado *voxel* variar significativamente durante as condições em todo o experimento.

ANOVA realiza uma análise separadamente para cada *voxel*. Um segundo método para análise foi implementado que é o *Kruskall-Wallis*, uma análise estatística equivalente a ANOVA, mas não parametrizada. Esse método foi escolhido já que o uso de ANOVA tende a assumir que os dados estão parametrizados, isto é, os valores de ativação dos *voxels* estão dentro de uma distribuição Gaussiana o que frequentemente não é o caso desses experimentos.

Portanto, o uso de ANOVA em dados não parametrizados poderia levar a falsos resultados e com o uso de *Kruskall-Wallis* não houve esse problema.

O passo seguinte é verificar qual o método de *Feature Selection* foi selecionado e chamar a função responsável por tal, a qual possui os parâmetros de informações de individuo, *pattern* após o *zscore, condition regressor* e os índices de iterações criados.

Com esses dados é possível começar a classificação de dados de acordo com a n-minus-one Cross-Validation. É necessário adicionar alguns parâmetros de básicos para o classificador de Backpropagation, sendo este um método comum de treinamento de redes neurais na qual o resultado inicial do sistema é comparado com o resultado desejado e o sistema é ajustado até que a diferença entre os dois seja minimizada. Os dados são separados em teste e treinamento. É chamada a função de *Cross-Validation* que classificará os dados após o *zscore* ter sido realizado de acordo com as condições o qual chamará o *Backpropagation* múltiplas vezes, uma por iteração de *n-minus-one*, iterando pelo número de índices criados e a *mask* definida que auxiliará na classificação informando quais características serão alimentadas ao classificador.

Após a classificação ter sido finalizada, o ponteiro do mouse voltará ao seu estado original e a função para trazer as informações resumidamente para o *Summary* será chamada. Além disso, o resultado total de precisão do desempenho do classificador será armazenado em uma variável e então mostrado ao usuário no quadro de *Results.* Tendo isso sido feito, é iniciada a remoção de todos os objetos auxiliares criados para essa análise, no caso de o usuário querer recomeçá-la de forma que não lhe causará nenhum problema, pois não precisará sair da tela ou carregar todos os dados novamente, basta clicar no botão de *Start Analysis.*

```
% --- Executes on button press in btnStart.
function btnStart_Callback(hObject, eventdata, handles)
% hObject    handle to btnStart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

nFeature = get(handles.mFeatureSelection, 'Value');

global rsName;
global patName;
global crName;
global subj;

oldpointer = get(handles.figure1, 'pointer');
set(handles.figure1, 'pointer', 'watch')
drawnow;
```

```
% the computation goes here

% PRE-PROCESSING - z-scoring in time and no-peeking anova
% we want to z-score the EPI data (called patName),
% individually on each run (using the 'runs' selectors)
subj = zscore_runs(subj,patName,rsName);

% now, create selector indices for the n different iterations of
% the nminusone
%subj = create_xvalid_indices(subj,rsName);
% substituing the runs by the new vector created after the run selector
subj = create_xvalid_indices(subj,'blocks');

switch nFeature
    case 1
% run the anova multiple times, separately for each iteration,
% using the selector indices created above
    %[subj] =
feature_select(subj,sprintf('%s_z',patName),crName,sprintf('%s_xval',
rsName));
        [subj] =
feature_select(subj,sprintf('%s_z',patName),crName,'blocks_xval');
    case 2
        [subj] = feature_select_kruskalwallis(subj,sprintf('%s_z',patName),
                                              crName,'blocks_xval');
    otherwise
     h = msgbox('Please, select a valid option!','Warning', 'warn');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CLASSIFICATION - n-minus-one cross-validation

% set some basic arguments for a backprop classifier
class_args.train_funct_name = 'train_bp';
class_args.test_funct_name = 'test_bp';
class_args.nHidden = 0;
% now, run the classification multiple times, training and testing
% on different subsets of the data on each iteration
[subj results] = cross_validation(subj,sprintf('%s_z',patName),crName,
'blocks_xval',sprintf('%s_z_thresh0.05',patName),class_args);
set(handles.figure1, 'pointer', oldpointer)
result = mySummarizeFn(subj);
res = results.total_perf;
set(handles.edtResult, 'String', sprintf('Results of the total perfomance =
%.5f',
res));

set(handles.edtSummary, 'String', result);

subj = remove_object(subj, 'pattern', sprintf('%s_z',patName));
subj = remove_group(subj, 'mask', sprintf('%s_z_thresh0.05',patName));
subj = remove_group(subj, 'selector', 'blocks_xval');
switch nFeature
    case 1
       subj = remove_group(subj, 'pattern', sprintf('%s_z_anova',patName));
    case 2
       subj = remove_group(subj, 'pattern',
sprintf('%s_z_kruskalwallis',patName));
end
```

Listagem 6 – Código para análise dos dados

Na Listagem 7 está o que é feito para a alteração de *runs* para *blocks.* Ele receberá uma matriz de *Condition Regressor* e criará uma um vetor de 1 pelo número de *timepoints* que é o tamanho da matriz de *condition regressor*. Com o comando de *sum* é verificado qual é a soma dos elementos da matriz de *condition regressor* e essa soma é armazenada na variável *conditionVector.* Uma variável de controle é criada para ter seu valor alterado dentro do *for loop*.

No *for loop*, é verificado se a variável *conditionVector* na posição *i* é maior do que zero e caso seja é realizada outra verificação para saber se o *conditionVector* na posição *i* é maior do que o *conditionVector* na posição *i-1*, o que significa que ele teve seu valor alterado e portanto, a variável de controle *blockCount* é incrementada em 1. É verificado também se a variável de controle é 0. Caso seja, o vetor de zeros na posição *i* receberá 1 do contrário receberá a variável de controle.

```
function blocks=runs_to_blocks(conditionRegressor)

%This function transforms the condition regressor matrix into a
matrix where each
%block (or active condition with the following rest block) will be
given a
%number in the same format as the run selector. This is to create the
cross
%validation indices based on blocks and not based on runs.
%adujsted for the haemodynamic lag

timePoints=length(conditionRegressor);
blocks=zeros(1,timePoints);
conditionVector=sum(conditionRegressor);
blockCount=0;
for i=1:timePoints
    if conditionVector(i)>0
        if conditionVector(i)>conditionVector(i-1)
            blockCount=blockCount+1;
        end
    end
    if blockCount==0
        blocks(i)=1;
    else
        blocks(i)=blockCount;
    end
end

end
```

Listagem 7 – Alteração de runs para blocks

5 CONCLUSÃO

O cérebro é um órgão tão complexo e pode ser abordado de muitas maneiras diferentes utilizando diferentes técnicas e experimentos que estudá-lo pode ser associado com encontrar uma entrada – um fenômeno interessante para ser estudado – mas também muito fácil de perder-se ao tentar encontrar uma saída (ALLPORT, 1986). Até recentemente cientistas estudavam apenas as estruturas do cérebro, mas tecnologias recentes como *PET Scan (Positron Emission Tomography)* ou tomografia por emissão de pósitrons e fMRI tem permitido olhar além da anatomia e estudar como essas partes funcionam e se relacionam.

O cérebro é como uma máquina de processamento de informações. Há uma grande quantidade de informações escondidas dentro dos padrões de ativação cerebral e é por isso que técnicas como *MVPA*, que permitem estudar a dinâmica de ativação do cérebro, têm se tornado cada vez mais valiosas e têm o potencial de auxiliar no entendimento em um nível mais profundo de como o cérebro processa a informação.

Considerando as possibilidades de análise de dados obtidos de atividade cerebral e a diversidade de profissionais envolvidos nessas análises e que nem sempre possuem conhecimento aprofundado para lidar com ferramentas computacionais complexas, o desenvolvimento de uma interface gráfica de fácil uso se apresentou como necessário para incentivar o uso da ferramenta MVPA da Universidade de Princeton. Além disso, tornar essa ferramenta gratuita permitirá que um maior número de cientistas possa utilizá-la para realizar análises sem precisar implementar códigos próprios, agilizando essa análise sem necessidade de conhecimento prévio em ciência da computação ou *Machine Learning.* A ferramenta desenvolvida permitirá experimentação com MVPA por meio de uma interface que propicie interação simples e facilitada.

Primeiramente, para construir a ferramenta um amplo conhecimento foi necessário, o que conduz a uma das dificuldades encontradas neste trabalho. Não apenas conhecimento em computação é necessário, mas também entendimento ou compreensão, ainda que superficial, em biologia, psicologia, estatística e física. Portanto, não apenas a codificação da ferramenta pode ser desafiante devido ao fato de que muitas universidades não possuem o aprendizado e uso de MATLAB

nos seus currículos, mas também devido ao conhecimento necessário em outras áreas.

Considerando o não uso e aprendizado acadêmico de MATLAB, desenvolver a ferramenta modelada como resultado deste trabalho se torna ainda mais desafiador.

Finalmente, desenvolver a ferramenta e adquirir o conhecimento necessário para fazê-lo pode ter sido um enorme desafio, mas sabendo que o tempo gasto na sua implementação auxiliará muitos pesquisadores e médicos para que em suas jornadas possam ajudar pessoas é gratificante. Além disso, saber que essa ferramenta poderá ser utilizada para identificar padrões de atividade cerebral como de pessoas com autismo ou até mesmo de pessoas com esquizofrenia, com os dados vindos da fMRI, é um incentivo a mais para o seu desenvolvimento e continuidade de estudo.

REFERÊNCIAS

ALLPORT, Susan. **Explorers of the black box: the search for the cellular basis of memory.** W W Norton & Co Inc, 1986.

BRAVO, Cedric Marcelo Augusto Ayala; ALBUQUERQUE, Éder Lima de. **Introdução ao MATLAB** . Centro Nacional de Processamento de Alto Desempenho em São Paulo. Universidade Estadual de Campinas. Disponível em: <ftp://www.ufv.br/Dea/Disciplinas/Evandro/Eng691/Material%20Didatico/Apostila %20MATLAB%20Unicamp.pdf>. Acesso em: 4 nov. 2015.

BUCKNER, Randry L.; ANDREWS-HANNA, Jessica R.; SCHACTER, Daniel L. **The brain's default network: anatomy, function, and relevance to disease**. Annals of the New York Academy of Sciences, v.1124, 2008, p. 1-38.

BUXTON, Richard B. **Introduction to functional magnetic resonance imaging: principles and techniques**. Cambridge university press, 2009.

COUTANCHE, Marc N.; THOMPSON-SCHILL, Sharon L.; SCHULTZ, Robert T. Multi-voxel pattern analysis of fMRI data predicts clinical symptom severity. **Neuroimage**, v. 57, n. 1, p. 113-123, 2011.

DAVATZIKOS Christos; RUPAREL Kosha; FAN, YYong; SHEN Dinggang G.; ACHARYYA, Muktish; LOUGHEAD, Jonathan W,; GUR, Raquel C; LANGLEBEN David D. Classifying spatial patterns of brain activity with machine learning methods: application to lie detection, **NeuroImage**, v. 28, n. 3, p. 663–668, 2005.

DETRE, Ggreg J; POLYN, Sean M.; MOORE, Christopher; NATU, Vaidehi; SINGER, Benjamin; COHEN, Jonathan; HAXBY, James V.; NORMAN, Kenneth A. **The multi-voxel pattern analysis (MVPA) toolbox**. In: Annual Meeting of the Organization for Human Brain Mapping, 2006. Disponível em: <http://www.princeton.edu/~bdsinger/pubs/detre_et_al_mvpa_ohbm2006.pdf>. Acesso em: 15 nov. 2015..

GAO, Juan; LI, Chun-Fang; LIU, Zhen-Guo; LIU, Lian-Zhong. **Elicitation of machine learning to human learning from iterative error correcting**. In: International Conference on Machine Learning and Cybernetics, 2013, p. 229-234.

HAXBY, James V.; GOBBINI, Maria Ida; FUREY, Maura L.; ISHAI, Alimit; SCHOUTEN, Jennifer L.; Pietrini, Pietro. Distributed and overlapping representations of faces and objects in ventral temporal cortex. **Science**, 2001, v. 293, n. 5539, p. 2425-2430.

MAHMOUDI, Abdelhak; TAKERKART, Sylvain; REGRAGUI, Fakhita; BOUSSAOUD, Driss; BROVELLI, Andrea. MULTIVOXEL pattern analysis for fMRI data: a review**. Computational and Mathematical Methods in Medicine**, v. 2012, p. 1-14.

MARQUES, Jose P.; MADDAGE, Rajika; MLYNARIK, Vladimir; Gruetter, Rolf. On the origin of the MR image phase contrast: an in vivo MR microscopy study of the rat brain at 14.1 T. **Neuroimage**, v. 46, n. 2, p. 345-352, 2009.

MATLAB. **Matlab: language reference manual**, version 5. Disponível em: <http://ub.cbm.uam.es/publications/teaching/master_biofisica_2011_2012/MATLAB_REFBOOK.pdf>. Acesso em: 4 nov. 2015a.

MATLAB. **MATLAB The language of technical computing**. Disponível em: <http://www.mathworks.com/products/matlab/index-b.html>. Acesso em: 4 nov. 2015b.

MITCHELL. Tomm; HUTCHINSON, Rebecca; NICULESCU, Radu S.; PEREIRA, Francisco; WANG, Xuerui; JUST, Marcel; NEWMAN, Sharlene. Learning to decode cognitive states from brain images. **Machine Learning**, v. 57, p. 145–175, 2004

NORMAN, Kenneth A.; POLYN, Sean M.; DETRE, Greg J.; HAXBY, James V. Beyond mind-reading: multi-voxel pattern analysis of fMRI data. **Trends in Cognitive Sciences,** v.10, n. 9, p. 424-430.

RISSMAN, Jessé; GREELY, HENRY T.; WAGNER, Anthony D. Detecting individual memories through the neural: dDecoding of memory states and past experience. **National Academy of Sciences**, v. 107, n. 21, p. 9849-9854, 2010.

STEPHAN, Klaas Enno. **Spatial preprocessing of fMRI data: methods & models for fMRI data analysis**. 2009, Laboratory for Social and Neural Systems. Disponível em: <http://slideplayer.com/slide/4986788/>. Acesso em: 28 nov. 2015.

SONG, Xiaomu; IORDANESCU, George; WYRWICZ, Alice M. **One-class machine learning for brain activation detection**. In: IEEE Conference on Computer Vision and Pattern Recognition, 2007 (CVPR '07), 2007, p. 1-6.

VIDHATE, Deepak; KULKARNI, Parag. **Cooperative machine learning with information fusion for dynamic decision making in diagnostic applications**. 2012 International Conference on Advances in Mobile Network, Communication and Its Applications. 2012, p. 70-74.

WANG, Hua; MA Cuiqin; ZHOU Lijuan. **A brief review of machine learning and its application**. International Conference on Information Engineering and Computer Science (ICIECS 2009), 2009, p. 1-4.

WOLBERS, Thomas; ZAHORIK, Pavel; GIUDICE, Nicholas. A. **Decoding the direction of auditory motion in blind humans**. Neuroimage, v. 56, n. 2, p. 681-687, 2011

YOON, Jong H.; TAMIR, Diana; MINZENBERG, Michael J.; URSU, Stefan; CARTER, Cameron S. Multivariate pattern analysis of functional magnetic resonance imaging data reveals deficits in distributed representations in schizophrenia. **Biological psychiatry**, v. 64, n. 12, p. 1035-1041, 2008.

TROCOLI, Tathiana. Áreas Cerebrais e Funções Correlatas. 2009. Disponível em: <https://tathianatrocoli.wordpress.com/2009/12/28/areas-cerebrais-e-funcoes-correlatas-2/>. Acesso em: 25 nov. 2015.

SCHUNEMANN, Gustavo Henrique. **Cérebros que se revelam. 2014. Série Ensaios: Aplicação da Etologia**. Disponível em: **<**http://etologia-no-dia-a-dia.blogspot.com.br/2014_03_01_archive.html>. Acesso em: 25 nov. 2015.

THOR. **Medicina y Farmacología**. 2010. Disponível em: <http://medicinafarmacologia.blogspot.com.br/2010/04/area-de-wernicke.html>. Acesso em: 25 nov. 2015.