ESCOLA POLITÉCNICA
DE PERNAMBUCO

Departamento de
Sistemas
Computacionais

Aplicação de Redes Neurais para Análise de Imagens Funcionais em Ambiente Paralelo

Trabalho de Conclusão de Curso

Engenharia da Computação

Antônio Henrique de Melo Costa
Orientador: Prof. Wellington Pinheiro dos Santos

Recife, novembro de 2007

UNIVERSIDADE
DE PERNAMBUCO

ESCOLA POLITÉCNICA
DE PERNAMBUCO

Departamento de
Sistemas
Computacionais

Aplicação de Redes Neurais para Análise de Imagens Funcionais em Ambiente Paralelo

Trabalho de Conclusão de Curso

Engenharia da Computação

Este Projeto é apresentado como requisito parcial para obtenção do diploma de Bacharel em Engenharia da Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Antônio Henrique de Melo Costa
Orientador: Prof. Wellington Pinheiro dos Santos

Recife, novembro de 2007

UNIVERSIDADE
DE PERNAMBUCO

Antônio Henrique de Melo Costa

Aplicação de Redes Neurais para Análise de Imagens Funcionais em Ambiente Paralelo

Resumo

O cérebro humano é, para muitos, uma das entidades mais complexas existentes. Nesse contexto existem inúmeras divisões de estudo. Uma parcela desse estudo é mapear regiões ou áreas responsáveis por determinadas atividades. A análise de imagens funcionais possibilita o estudo deste mapeamento.

Imagens funcionais permitem a percepção de regiões de ativação cerebral em função de algum estímulo, possibilitando o mapeamento da região do cérebro responsável pelos movimentos em função de um estímulo externo, ou da imaginação em função do pensamento em algo fantasioso. Neste trabalho serão utilizadas somente imagens funcionais provenientes de aquisição por ressonância magnética.

A análise das imagens pode ser feita usando técnicas de análise estatística, como o *Student's t-test*, Regressão Linear, Subtração, entre outras, ou através do uso de técnicas inteligentes como Redes Neurais. O uso do primeiro grupo é feito *pixel* a *pixel*, não utilizando informação de vizinhança de cada *pixel* durante a análise. Devido a isto, o uso de redes neurais pode ser bastante interessante, pois permite, neste caso, a análise do conjunto *pixel* e sua respectiva vizinhança, de maneira semelhante ao processo feito pelo cérebro humano.

Os algoritmos de redes neurais podem ser divididos em processo de aprendizagem. Estes algoritmos podem ser *supervisionados* (há uma espécie de professor, que faz ajustes em função da resposta desejada) e *não-supervisionados* (não há uma resposta desejada, nem informação *a priori*).

Independentemente da técnica de análise (estatística ou rede neural) aplicada às imagens funcionais de ressonância magnética usada há um dispêndio quanto ao tempo, espaço de armazenamento e memória utilizados no processamento computacional. Esses fatores são maiores ainda se há o uso de redes neurais.

Assim, é notável a necessidade de alguma ferramenta que permita a fácil comparação entre algoritmos de análise, entre o tipo de arquitetura (uniprocessador ou multiprocessador) e entre o uso de técnicas de pré-processamento.

Este trabalho aborda as áreas de conhecimento acima mencionadas, bem como a implementação de algumas técnicas e métodos relacionados. São mostrados testes e experimentos realizados, apresentando o uso de *clusters* como alternativa a possíveis melhoras de desempenho em computação com baixo custo.

Abstract

The human brain is, for many people, one of the most complex existent entities. In this context, there are countless study areas. One of these areas studies the mapping of regions responsible for determined activities. The analysis of functional images enables the study of this mapping.

Functional images allow the perception of regions of cerebral activation as a stimulus function, enabling the mapping of the region of the brain responsible by the movements in function of this stimulus, or by imagination in function of fanciful thoughts. In this work, only functional images acquired by magnetic resonance will be used.

The analysis of images can be done using statistic analysis techniques, or through the use of intelligent techniques like Neural Networks. Statistic analysis techniques, such as the Student's t-test, Linear Regression or Subtraction, among others, analyze images without using the pixel's neighborhood information during the analysis. Neural networks allow the analysis of pixels and their respective neighborhoods processing information in a similar way as the human brain does, making them a promising approach for this kind of analysis.

Neural network algorithms can be divided by the learning process. These algorithms can be supervised (there is a kind of teacher, performing adjustments depending on the desired response), and non-supervised (there is no desired response or information in advance), among others.

Regardless of the analysis technique applied to the functional images, there is an expenditure regarding the time, storage space, and memory used during computer processing. These factors have more influence if neural networks are used.

Therefore, it is clearly necessary a tool that facilitates the comparison between analysis algorithms, architecture types and usage of pre-processing techniques.

This work addresses the knowledge areas mentioned above, as well as the implementation of methods and techniques related. Tests and experiments are shown, presenting the use of clusters as an alternative to possible performance improvements on low-cost computing.

Sumário

Índice de Figuras

Índice de Tabelas

Tabela de Símbolos e Siglas

RNA – Rede Neural Artificial
fMRI – *functional Magnetic Resonance Imaging*
HPC – High Performance Cluster
HAC – High Avaliability Cluster
E/S – Entrada e Saída
UHN – Unique Home Node

Agradecimentos

Não é em vão que o meu primeiro e mais importante agradecimento é dado a Deus. Fidedigno em todas as ocasiões. Disposto a oferecer tranqüilidade e força nos momentos difíceis, assim como possibilidade de gozo em momentos de alegria. Agradeço também a meus irmãos na fé, que como imitadores de Cristo ofereceram também tranqüilidade, força e alegria. Sem dúvida posso exceder a frase "... *Até aqui nos ajudou o Senhor*. I Sm 7.12", pois sei Ele vai me ofertar muitos outros momentos semelhantes a este.

Agradeço sem dúvida a minha amada amiga, companheira e eterna namorada, Clara Beatriz. Esposa com qualidades, fez criar apoios para me sustentar nessa jornada onde muitas vezes pensei em desistir.

A meus pais, que com exemplos de vida aprendi o que é ter caráter e força de vontade. Obrigado Antônio e Vera.

A minha família, irmão, tios, avó, sogros, cunhados. Torcendo, apoiando, comemorando e aconselhando em cada decisão por mim tomada.

Aos professores do DSC, na qual sinto orgulho de ser aluno. Exemplos de seriedade, competência e amizade. Ressalto agradecimento ao professor Wellington Pinheiro dos Santos por seu excelente caráter, por sua amizade inquestionável, por seu domínio nas áreas de conhecimento. Um exemplo a ser seguido. Muito obrigado professor.

A meus irmãos de faculdade. Bruno, Edésio, Paulo César, Victor, Vinícius, Thiago, dentre tantos outros que contribuíram, e muito, na minha formação como pessoa e como profissional. Quantas noites "viradas" fazendo projetos...

E por fim, mas não menos importante, as pessoas da Cemicro, empresa onde trabalhei, e ao meu novo lar profissional: o Banco do Brasil.

Capítulo 1

Introdução

Problemas diversos sempre foram ferramentas para promover desenvolvimento ao homem, através da necessidade de resolvê-los. Computadores digitais, hoje, são um dos meios mais usados para o auxilio à resolução de problemas diversos: desde contas aritméticas a previsões de tempo. Para isso, há algoritmos (procedimentos bem definidos e objetivos com condição de término) que implementados de alguma maneira, seja em *hardware*, *software* ou ambos, com o intuito de descrever o caminho a solução – esperada ou não – do problema. Entretanto, em muitos dos problemas, não existe um procedimento formal para a sua resolução. Para encontrar formas de resposta a tais, buscou-se inspiração no melhor objeto computacional conhecido no universo: o cérebro humano. Embora lento em muitas situações se comparado a máquinas, o cérebro possui a capacidade até então não sintetizada em máquina alguma: o aprendizado.

Por volta de 1950 foi modelado o primeiro neurônio artificial e pesquisas na área de inteligência artificial foram sendo desenvolvidas. Porém, devido ao fraco *hardware* existente, houve uma estagnação por alguns anos [1].

Com equipamentos mais poderosos em termos computacionais e interação com o meio, algoritmos inteligentes, em especial Redes Neurais Artificiais (RNAs), estão em uso em inúmeras áreas como o reconhecimento de retina (biometria), sistemas de previsão, etc.

RNAs podem aprender de várias maneiras, dentre as principais o aprendizado supervisionado, onde há um professor mostrando a resposta desejada ao sistema, e o aprendizado não-supervisionado, onde não há professores. Em ambos os casos, a computação usada nas fases de pré-processamento dos dados de entrada, de treinamento, de validação e de teste, assim como no uso da rede já treinada pode ser um tanto custosa em termos de desempenho, de recursos e de custos.

Uma das propostas deste trabalho é mensurar o desempenho e estudar a viabilidade do uso de arquitetura multiprocessador com memória dedicada, conhecida como *cluster*, para melhorar o desempenho do uso de RNAs, em especial as técnicas de aprendizado não-supervisionado.

É proposto o uso do conjunto das técnicas acima citadas, tanto quanto a arquitetura de *hardware* quanto a RNAs, para a análise de imagens funcionais de ressonância magnética (fMRI) cerebrais. Imagens funcionais [4] permitem a percepção de regiões de ativação cerebral em função de algum estímulo.

Quanto ao pré-processamento, é pretendido o uso de Morfologia Matemática [3] [4], para a expansão das bandas de cada imagem usada como dado de entrada imagens funcionais de ressonância magnéticas cerebrais divididas em dois conjuntos: um conjunto de imagens inativas e um conjunto de imagens ativas. A ativação ou inativação é em função da motivação do dedo

indicador direito. O uso de Morfologia Matemática é usado neste trabalho para permitir a expansão das bandas como informação de vizinhança de cada *pixel* da imagem original. É esperada a melhora da fase de treinamento da RNA.

Este trabalho propõe o estudo e avaliação do comportamento de algoritmos de classificação não-supervisionado em ambiente paralelo como o auxílio de Morfologia Matemática no pré-processamento da rede. Propõe também a implementação do método *Student's t-test* para comparações, testes e exemplificações.

1.1 Objetivos

Este trabalho tem como objetivo o desenvolvimento aplicações que utilizem as tecnologias e técnicas já citadas como RNAs, *clusters*, Morfologia Matemática à análise de imagens provenientes de aparelhos de fMRI. Para alcançar tais objetivos foram traçadas as seguintes metas:

- Desenvolvimento de uma aplicação em ambiente de *cluster* para a análise de imagens funcionais;
- Desenvolvimento e implementação de módulos que englobem todas as técnicas apresentadas neste capítulo.

1.2 Estrutura do Documento

Este trabalho foi organizado em 7 (sete) capítulos. Os capítulos, com exceção deste, foram dispostos da seguinte maneira:

- O capítulo 2 apresenta conceitos de *clusters* de computadores. Mostra seus princípios, tipos, arquiteturas e pacotes de software, em especial o *openMosix*, na qual foi objeto de estudo deste trabalho;
- No capítulo 3, há a introdução a Redes Neurais Artificiais. Em destaque o *perceptron*, e técnicas com *K-Means*, *Fuzzy C-Means* e *Kohonen SOM*;
- O capítulo 4 apresenta introduções a Imagens Funcionais. Duas técnicas são apresentadas: PET e fMRI sendo a última, foco deste trabalho. É apresentado também técnicas de detecção de atividade em fMRI como a subtração e o Test t de *Student*;
- No capítulo 5 é apresentado a morfologia matemática, bem como as operações de erosão, dilatação e as operações derivadas destas: abertura e fechamento;
- São mostrados no capítulo 6, os resultados dos estudos realizados nas aplicações implementadas e nos testes;
- O capítulo 7 expõe conclusões e interesse a trabalhos futuros.

Capítulo 2

Clusters

Do inglês, *cluster* significa um grupo de coisas semelhantes que estão próximas umas das outras, por vezes com algo ao redor [5].

Em computação, *cluster tem* vários significados. Um grupo com características semelhantes em uma Rede Neural Artificial [6] é chamado de *cluster*. No exemplo clássico das *Bailarinas e Halterofilistas* [7] cada um destes é um *cluster*. Em termos de tecnologia de armazenamento, *cluster* é uma unidade lógica de armazenamento de arquivos em um disco rígido, na qual é gerenciado por um sistema operacional. *Cluster* também pode ser um grupo de computadores interconectados, onde todos os integrantes trabalham juntos como um recurso computacional unificado, na qual cria uma ilusão de haver apenas uma máquina [8]. Esta última definição será abordada neste capítulo.

2.1 Princípios de *Cluster*

Caracteriza-se por *cluster* o ambiente que oferece um recurso computacional por meio de vários computadores interligados por uma rede, que são vistos pelos usuários, aplicativos e outros servidores como um recurso único. Além disso, os usuários não devem saber que estão usando um *cluster* e os demais servidores não devem saber que estão se comunicando com um *cluster* [9].

Segundo *Kopper*, as quatro características básicas de um *cluster* são:

1. Os usuários não devem saber que estão usando um *cluster*;
2. Os computadores não sabem que eles fazem parte do *cluster*;
3. As aplicações em execução, não sabem que fazem parte de um *cluster*;
4. Demais computadores que compõe o *cluster* têm que ser vistos como clientes comuns.

A Figura 1 mostra um *cluster* formado por quatro computadores comuns. Eles estão interligados através de um *switch* e compartilham um dispositivo de armazenamento.

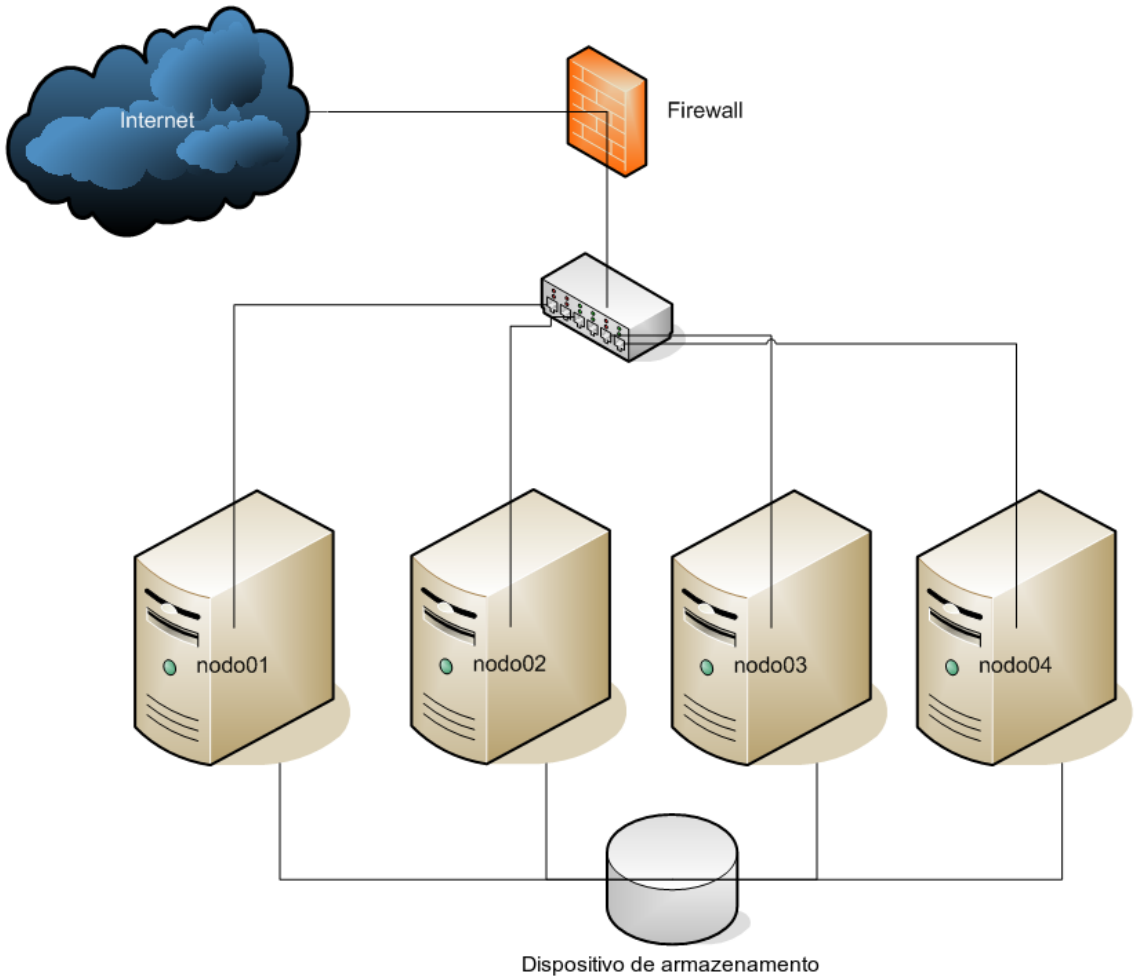


Figura 1. *Cluster com quatro nodos.*

Na literatura, há quatro benefícios que estão associados a *clusters*.

- **Escalabilidade absoluta**: É possível criar *clusters* imensos, que de longe superam o poder de máquinas *standalone* poderosas. Um *cluster* pode ter dezenas de máquinas, cada uma delas multiprocessada, ou seja, com mais de um processador.
- **Escalabilidade incremental**: Um *cluster* é configurado de tal maneira que é possível adicionar novos sistemas ao *cluster* com pequenos incrementos. Assim o usuário pode começar como um sistema modesto (poucos nodos) e expandi-lo conforme necessidade.
- **Alta disponibilidade**: Devido ao fato de que cada nodo do *cluster* é um computador *standalone*, a falha de um dos nodos não significa a perda do serviço.
- **Baixo custo e independência do fornecedor**: o custo de montagem do sistema é baixo, pois pode ser usado *hardware* comum que pode ser de qualquer fabricante. Quanto ao sistema operacional e aplicativos, há a possibilidade da utilização de sistemas de Código Aberto (*Open Source*) onde não existe a necessidade de pagamento de licença de uso.

2.2 Tipos de *Clusters*

Originalmente, *clusters* e computação de alto desempenho são sinônimos. Atualmente, o significado da palavra *cluster* foi expandido para além de alto-desempenho, para clusters de alto-desempenho (HPC – *High Performance Cluster*), *clusters* de alta-disponibilidade (HAC – *High Avaliability Cluster*) e *clusters* de balanceamento de carga (LBC – *Load Balancing Cluster*). Na prática, não há muita diferença, pois, depois de tudo, todos são *clusters*.

Clusters de alta-disponibilidade, também chamado de *clusters* tolerantes a falhas, são geralmente usados em aplicações de missão-crítica. Se um servidor *web* de uma corporação não pode parar de nenhuma maneira, pois se isso acontecer irá acarretar em perda de negócios, é um ótimo caso de aplicação de *clusters* HA, por exemplo [10]. A chave para alta-disponibilidade é redundância. Este tipo de *cluster* é composto por várias máquinas e um subconjunto delas pode oferecer um determinado serviço. Dessa maneira, apenas uma máquina ou servidor está diretamente disponível – todas as outras máquinas estarão em modo de espera. Eles irão monitorar o servidor primário para assegurar que ele esteja operacional, e em caso de falha no sistema, outro servidor assume.

A idéia por trás dos *clusters* de balanceamento de carga é oferecer melhor desempenho pela divisão de trabalho entre os múltiplos computadores no *cluster*. Por exemplo, quando um servidor *Web* é implementado usando *cluster* LB, as diferentes requisições são distribuídas entre os computadores do *cluster*. Isso pode ser feito usando simples algoritmos *round-robin*. O *Round-Robin* DNS pode ser usado para mapear respostas de requisições DNS para diferentes endereços IP, por exemplo. Isto é, quando a requisição DNS é feita o servidor DNS local retorna o endereço da próxima máquina no *cluster*, visitando máquinas de modo *round-robin*. Todavia, essa abordagem pode levar a desequilíbrios de balanceamento de carga. Muitos algoritmos sofisticados usam informações de cada máquina de maneira individual para determinar qual máquina pode melhor manipular a próxima tarefa.

2.2.1 Clusters de alto-desempenho (HPC – *High Performance Cluster*)

Neste tipo de *cluster*, tarefas que exigem um nível elevado de processamento são divididas entre os nodos, isto é, são executadas de maneira paralela com a intenção que o tempo total de processamento seja menor. Quanto mais nodos estiverem ligados ao *cluster*, menos tempo será necessário para executar todo o processamento [11].

A aplicação de *clusters* de alto-desempenho pode ser realizada em ambientes que possuam uma grande demanda de dados a serem computados. Alguns exemplos são banco de dados robustos, aplicações de engenharia e portais com grande volume de acesso.

A utilização de aplicações que fazem o uso de processamento paralelo é uma tarefa complexa, e, devido a isto, necessita ser bem planejada. Além do desenvolvimento de aplicações ser muito trabalhoso, existe grande quantidade de memória e bastante tempo de processamento.
No Brasil, existem cinco Centros Nacionais de Processamento de Alto Desempenho (CENAPAD) espalhados pelo país. Estão instalados junto à Unicamp em Campinas – SP, UFPE em Recife – PE, UFRGS em Porto Alegre – RS, UFMG em Belo Horizonte – MG e UFRJ, no Rio de Janeiro – RJ. Surgiram em 1994 com o objetivo de apoiar o avanço de pesquisas nas áreas de ciência e tecnologia no país. Para isso disponibilizam aos usuários laboratórios com equipamentos de alto desempenho, recursos de hardware e aplicativos, além do suporte técnico [12]. Também a Petrobrás utiliza clusters de alto desempenho nas plataformas, 26 para executar cálculos relacionados à extração de petróleo, e simulação numérica em reservatórios de petróleo.

2.2.2 *Clusters* de alta-disponibilidade (HAC – *High Avaliability Cluster*)

Os *clusters* de alta-disponibilidade – *High Avaliability Clusters* (HAC) tem como objetivo manter um ou mais serviços disponíveis a maior parte do tempo possível e de forma segura [13]. Os recursos têm que estar sempre, ou quase sempre, acessíveis para atender às requisições dos clientes, ou seja, têm que estar acessíveis por um período de tempo muito próximo a 100% [11].

2.2.3 *Clusters* de balanceamento de carga (LBC – *Load Balance Clusters*)

Os *clusters* balanceamento de carga – *Load Balance Clusters* (LBC) são mistos de *clusters* de alto-desempenho com *clusters* de alta-disponibilidade.

Neste tipo de *cluster*, vários nodos, chamados de nodos diretores ou *load balancers*, atendem requisições de um serviço qualquer e repassam aos demais nodos do cluster. Estes por sua vez fazem o processamento das informações e são conhecidos como servidores reais [2]. Esta técnica faz com que não haja um ponto de gargalo. É muito usada em grandes portais de Internet, que recebem um grande número de acessos simultâneos [2]

Um exemplo de empresa que utiliza em grande escala os *clusters* do tipo LBC é o Google. Neste ambiente, diversos nodos recebem às requisições de busca feitas pelos usuários e repassam aos nodos reais, que então fazem a busca e devolvem os resultados ao usuário [10].

2.3 Arquitetura de *Clusters* de Computadores

A Figura 1 mostra uma arquitetura típica de *clusters*. Os computares individuais por alguma *LAN* de alta velocidade ou um *switch*. Cada computador é capaz de operar independentemente. Além disso, uma camada de *software* (*middleware*) é instalada em cada comutador para permitir o *cluster*. O *middleware* do *cluster* fornece uma imagem do sistema unificada para o usuário, conhecida como *single-system image*. O *middleware* é responsável por manter a alta-disponibilidade, por meio de balanceamento de carga e identificação de falhas nos componentes individuais.

A lista de serviços e funções desejáveis em um *middleware* de *cluster* é a seguinte:

- **Único ponto de entrada**: o usuário conecta-se ao *cluster* e não ao computador individual.
- **Única hierarquia de arquivos**: o usuário vê uma simples hierarquia de diretórios de arquivos sobre um mesmo diretório raiz.
- **Ponto de controle único**: há uma estação padrão usada para o gerenciamento e controle do *cluster*.
- **Espaço de memória único**: memória compartilhada distribuída permite aos programas o compartilhamento de variáveis.

2.4 Pacotes de *Software*

Esta sessão descreve a instalação de três pacotes de *software* que, quando instalados, irão oferecer um *cluster* completo e funcional.

Esses pacotes diferem radicalmente. *OpenMosix* oferece extensão ao *kernel* do Linux que transparentemente move processos entre as máquinas para balancear a carga e otimizar o

desempenho. *OSCAR* e *Rocks* são coleções de pacotes de *software* que podem ser instalados uma vez, fornecendo um *cluster* nos padrões *Beowulf*.

2.4.1 *OpenMosix*

O *openMosix* é um *software* que extende o *kernel* do Linux tal que os processos podem migrar transparentemente entre as diferentes máquinas no *cluster* a fim de distribuir a carga de trabalho de maneira mais eficiente [2].

Basicamente o *openMosix* inclue um conjunto *patches* para o *kernel* e ferramentas de suporte. Os *patches* extendem o *kernel* do para fornecer suporte a movimentação de processos entre as máquinas no *cluster*. Tipicamente, os processos migram de maneira transparente para o usuário. Todavia, pelo uso das ferramentas fornecidas pelo *openMosix*, bem como ferramentas de terceiros (*third-party*), é possível ao usuário o controle de migrações de processos.

O *openMosix* pode ser usado como tentativa de diminuição de tempo de processamento de tarefas computacionalmente caras. Como exemplo, se a tarefa a ser executada é a compressão de dezenas de arquivos, onde há uma intensa requisição de CPU e esta tarefa não será executada em um *cluster openMosix*, será necessária a compressão de um arquivo por vez (*batch*). Mesmo que seja aberta uma quantidade de instancias do software de compressão igual ao número de arquivos a serem comprimidos, o tempo de execução, neste caso, será o mesmo da compressão em *batch*. Todavia, se o computador é parte de um *cluster openMosix*, a tarefa de compressão de todos os arquivos será feita da seguinte forma: Primeiro, a inicialização de todos os processos no nodo que recebe a requisição de compressão. Em um *cluster openMosix*, depois de poucos segundos, os processos irão iniciar a migração da máquina que está mais sobrecarregada, para outra que esteja ociosa ou com menos carga de processamento no *cluster* (devido ao fato de que muitos processos terminam rapidamente, pode ser contraproducente a migração rápida). Se há dezenas de máquinas ociosas no *cluster*, cada compressão deve rodar em uma máquina diferente.

Se o número de computadores no *cluster* não é grande, ou alguns computadores são mais lentos que outros, ou alguns destes estão sobrecarregados, o *openMosix* irá mover os *jobs* (porção de computação) com o objetivo de obter o melhor balanceamento de carga. Uma vez que o *cluster* é implantado, isso é feito de maneira transparente pelo sistema. Normalmente o usuário apenas inicia os *jobs* e o *openMosix* faz o restante. Por outro lado, se há a necessidade do controle da migração dos *jobs* de um computador para outro, *openMosix* oferece ferramentas que possibilitam isso.

A granularidade do *openMosix* é processo. Programas individuais, como o exemplo da compressão, podem criar processos. Processos também pode ser resultado de diferentes *forks* de um simples programa. Todavia, se há uma tarefa computacionalmente intensiva, que realiza dezenas de atividades em um simples processo (mesmo com o uso de *threads*), então, devido ao fato de haver somente um processo, este não pode ser migrado. A única contribuição do *cluster* neste caso é a migração para a máquina com a melhor relação entre carga e recursos de *hardware*.

Nem todos os processos migram. Dependendo da configuração do *cluster*, processos que terminam antes do tempo mínimo de migração, não migram. Atualmente, *openMosix*, não trabalha com múltiplos processos que usam memória com escrita compartilhada, tais como servidores *Web* [2]. De maneira similar, processos fazem manipulação direta de dispositivos de E/S, não podem migrar. Processos que usam escalonamento de tempo-real, também não migram. Se um processo já migrou para outro processador, e tenta fazer algum dos casos citados anteriormente, o processo irá migrar de volta para o seu *unique home node* (UHN), nodo onde o processo foi inicialmente criado, antes de continuar.

Para suportar a migração de processos, o *openMosix* divide o processo em duas partes ou contextos. O *contexto de usuário* que contém o código do programa, pilha, dados, etc., e é a parte

que pode migrar. O *contexto do sistema*, na qual contem a descrição dos recursos do processo, é anexado na pilha do *kernel*, não migra, mas permanece no UHN.

OpenMosix usa uma política de alocação de recursos adaptativa. Isto é, cada nodo monitora e comprara sua própria carga e com a carga de parte dos outros computadores no *cluster*. Quando um computador acha outro com carga mais leve (baseada na capacidade total do computador), ele tenta fazer a migração de um processo para a máquina com menos carga, criando um melhor balanceamento de carga entre os dois. Como a carga nos computadores muda, isto é, quando *jobs* iniciam ou terminam, os processos irão migrar entre os computadores para rebalancear a carga no *cluster*, adaptando-se dinamicamente as mudanças nas cargas.

Nodos individuais atuam como sistemas autônomos, decidem quais processos migram. A comunicação entre pequenos conjuntos de nodos em um *cluster* usados para comparar a carga é aleatória. Conseqüentemente, o *cluster* tem boa escalabilidade por causa deste elemento aleatório. Uma vez que a comunicação está dentre deste subconjunto no *cluster*, os nodos são limitados apenas a informações recentes sobre o estado do *cluster* completo. Essa abordagem reduz a sobrecarga (*overhead*) e comunicação.

O pacote de ferramentas do *openMosix* oferece muitas ferramentas gerenciamento (*migrate*, *mosctl*, *mosmon*, *mosrun* e *setpe*), uma versão do *ps* e do *top*, chamados de *mps* e *mtop* e um *script* de inicialização (*/etc/init.d/openmosix* ou */etc/rc.d/init.d/openmosix*).

Nas Figuras 2 e 3, é mostrado respectivamente o *mps* e o *openMosixView.*

```
[ root@zion skyline ] ~# mps
PID    TTY    NODE   STAT   TIME   COMMAND

...

19766  ?      0      R      2:32   ./loop

19767  ?      2      S      1:45   ./loop

19768  ?      5      S      3:09   ./loop

19769  ?      4      S      2:58   ./loop

19770  ?      2      S      1:47   ./loop

19771  ?      3      S      2:59   ./loop

19772  ?      6      S      1:43   ./loop

19773  ?      0      R      1:59   ./loop

...
```

Figura 2. *Saída do comando mps. Na terceira coluna, o processo 19769 está rodando no nodo quatro. O MPs deve rodar na máquina onde o processo foi originado.*

2.4.2 OSCAR

Ajustar um *cluster* pode envolver a instalação e configuração de uma grande quantidade de *software* bem como a reconfiguração do sistema e dos *softwares* pré-instalados. *OSCAR* (*Open Source Cluster Application Resources*) é um pacote de *software* projetado para simplificar a instalação de *clusters*. Uma coleção de *softwares* para *clusters open source*, OSCAR inclui todos os componentes necessários para a implantação de um *cluster* de alto desempenho dedicado [2].

O principal objetivo do OSCAR é reduzia a necessidade de experiência na implantação de um *cluster*.

O OSCAR foi criado e é mantido pelo *Open Cluster Group*, um grupo informal e dedicado para simplificar a instalação e o uso de *cluster* e disseminar o seu uso. Durante os anos, um número de organizações e companhias tem suportado o *Open Cluster Group*, incluindo a Dell, IBM, Intel, NCSA e a ORNL dentre outras.

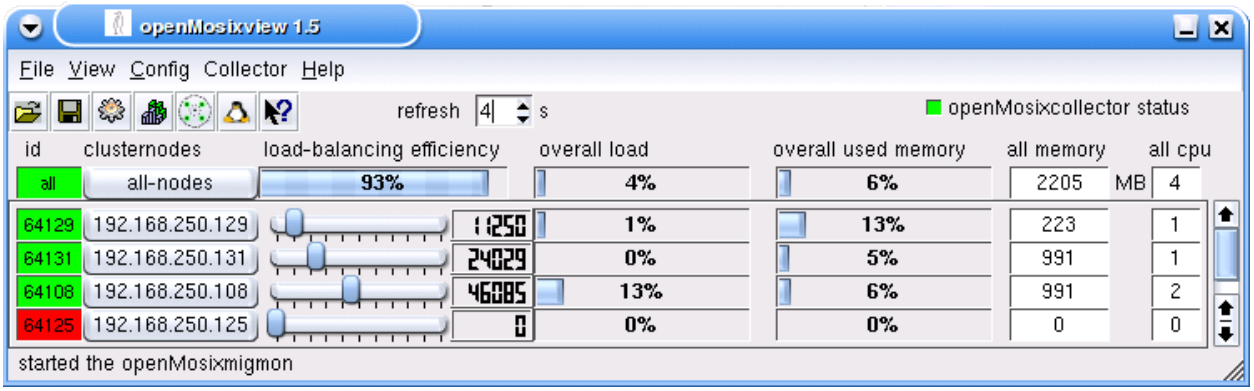


Figura 3. *openMosixView mostrando informações dos quatro nodos componentes do cluster. A primeira coluna o status do nodo pelo número (id). A cor é verde se o nodo está disponível ou vermelha se indisponível. Na segunda coluna, há o endereço IP de cada nodo e ao selecioná-lo é possível a configuração do sistema individual.*

OSCAR é planejado com computação de alto desempenho em mente. Basicamente, ele é designado para ser usado como um *cluster* assimétrico (na qual um computador do *cluster* funciona como *gateway* ou como interface com o usuário, isolando o restante do *cluster* com o exterior).

2.4.3 *Rocks*

Assim como o OSCAR, *Rocks* é uma coleção de *software* de código aberto para a construção de um *cluster* de alto desempenho. A primeira meta do *Rocks* é fazer a instalação de um *cluster* da maneira mais fácil possível.

A instalação padrão tem o propósito de ser bastante rápida e muito suave [2].

Capítulo 3

Redes Neurais Artificiais

Neste capítulo, adentraremos em uma porção da ciência que provoca bastante interesse a pesquisadores de inúmeras áreas: Redes Neurais.

As Redes Neurais, ou redes neurais artificiais mais precisamente, representam uma tecnologia que tem raízes em muitas disciplinas: neurociência, matemática, estatística, física, ciência da computação e engenharia. As redes neurais encontram aplicações em campos tão diversos, como modelagem, análise de séries temporais, reconhecimento de padrões, processamento de sinais e controle, em virtude de uma importante propriedade: a habilidade de aprender a partir de dados de entrada com ou sem professor.

3.1 Introdução

Redes Neurais fornecem uma alternativa poderosa e razoável para os classificadores convencionais. Os benefícios potenciais das RNAs se estendem além da alta taxa de computação fornecida pelo paralelismo massivo. Redes neurais fornecem um grande grau de robustez e tolerância a falhas. Muitas arquiteturas e algoritmos de aprendizado são disponíveis. RNAs são biologicamente inspiradas e fornecem uma poderosa ferramenta para projeção de sistema de visão computacional. Um sistema de visão computacional consiste de muitos estágios, tais como pré-processamento, extração de características, armazenamento associativo, base de conhecimentos e reconhecimento de padrões.

No estágio de pré-processamento, modelos de redes neurais têm sido usados para extração de bordas ou fronteiras, restauração de imagens e filtragem de imagens. Na extração de características, modelos de redes neurais têm sido usados para extrair domínio de características transformado. Redes neurais têm sido usadas como memória associativa para armazenamento e recebimento de informações.

Redes neurais aprendem através de amostras de treinamento e têm sido usadas para reconhecimento de padrões desde 1950s [6]. Redes tais como *perceptrons* e redes *feed-forward* (tais como MLP – *Multilayer Perceptron*) com um algoritmo de aprendizado *backpropagation* tem sido usadas como classificadores supervisionados. Os modelos de *perceptron* usam o conceito de recompensa e punição. No final da década de 60, progressos nos modelos de redes neurais estavam sendo lentamente diminuídos devido à capacidade limitadas dos modelos anteriores de *perceptron* com camada única [6]. Hoje, um número de bem-sucedidas teorias e algoritmos de aprendizado para modelos de redes neurais com multicamadas estão disponíveis.

Uma rede neural artificial pode ser considerada como um sistema de processamento de informações que mapeiam um vetor de dados de entrada em um vetor de dados de saída. Uma rede neural com múltiplas saídas pode ser considerada como uma coleção de multi-entradas independentes, sistemas de saída única [1] [6]. A mais significante vantagem de redes neurais é que a função de mapeamento pode ser determinada por vetores de treinamento, isto é, redes neurais podem aprender com amostras de treinamento. Redes neurais consistem de um grande numero de unidades simples de processamento chamadas de neurônios. O cérebro humano consiste de centenas de bilhões de neurônios, cada um capaz de receber, processar e transmitir sinais elétricos sobre as vias neurais que constituem o sistema de comunicação cerebral. Uma célula nervosa elementar é um bloco de construção fundamental do sistema biológico neural. Dois neurônios são conectados via linha (*link*) de conexão. Cada *link* tem um peso associado, que, em uma típica rede neural multiplica o sinal que é transmitido. Cada neurônio aplica uma função de ativação na sua respectiva entrada para determinar a saída. A rede neural é caracterizada pelos seus padrões de conexão, chamados de arquitetura da rede, e seu algoritmo de aprendizado, na qual determina o peso, ou força das conexões. Algumas das arquiteturas de redes comumente usadas são redes com duas camadas *feed-forward*, três camadas *feed-forward* e redes com única camada e conexões com *realimentação*. Isso é mostrado nas Figuras 4, 5 e 6.

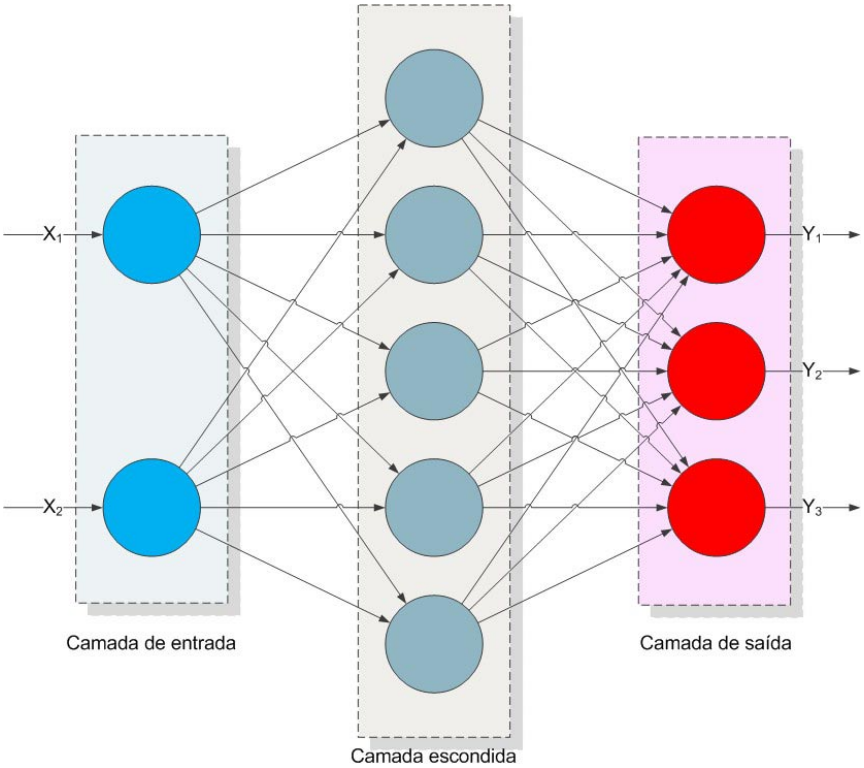


Figura 4. *Rede com 3 camadas feed-forward.*

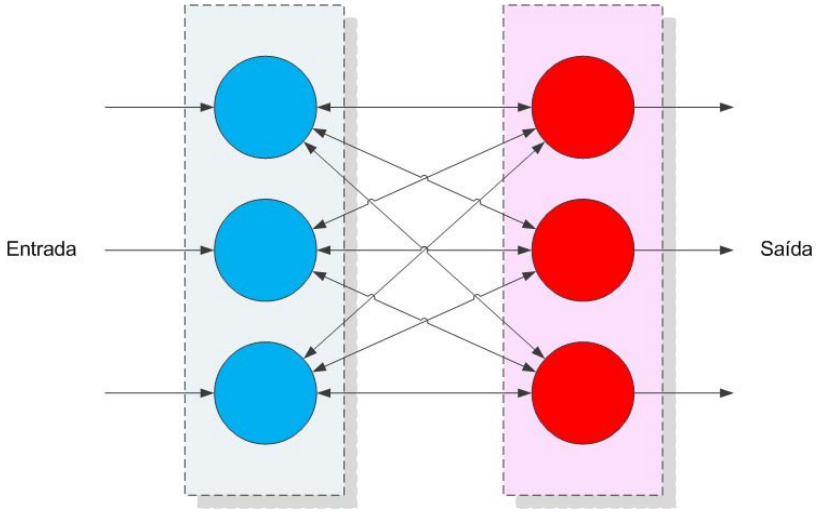


Figura 5. *Rede com duas camadas com conexões realimentação.*

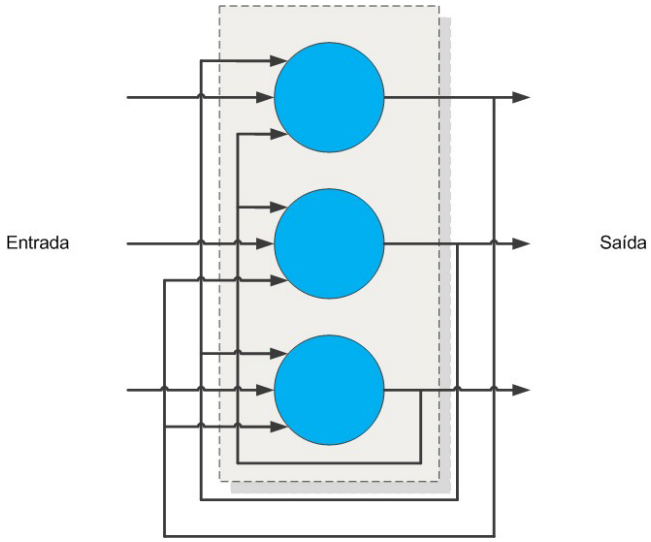


Figura 6. *Uma rede de única camada com conexões de realimentação.*

3.2 Representação do neurônio

Um modelo esquemático de um neurônio biológico é mostrado na Figura 7. Um típico neurônio biológico consiste de quatro partes básicas: dendritos, as sinapses, o corpo celular – na qual também é chamado de *soma* – e o axônio. Dendritos são estruturas ramificadas que fornecem entradas para o corpo celular. Os dendritos recebem sinais dos neurônios vizinhos em pontos de conexão chamados sinapses. Quando a carga de excitação cumulativa do corpo celular excede um limite, a célula dispara, enviando um sinal do axônio para outros neurônios. Axônios são fibras longas que servem como linhas de transmissão [6].

Um diagrama esquemático representando um neurônio artificial com *bias* (polarização) é mostrado na Figura 8. A operação básica de um neurônio artificial envolve o somatório das entradas vezes seus respectivos pesos e a aplicação da função de ativação para gerar a saída. Tomando $x = (x_1, x_2, \ldots, x_n)^T$ representa as n entrada aplicada ao neurônio artificial. A entrada net para o neurônio no tempo t é dada por:

$$net(t) = \sum_{i=1}^{n} x_i w_i \tag{3.1}$$

Onde w_i representa a o peso da entrada x_i

A entrada net, $net(t)$, é processada pela função de ativação F para produzir o sinal de saída do neurônio $o(t)$. Geralmente, os valores de ativação são restringidos ao intervalo [0, 1]. A função mais comumente usada para função de ativação é a função sigmóide, na qual é dada por:

$$F(x) = \frac{1}{1 + \exp(-x)} \tag{3.2}$$

A saída do neurônio, assumindo que a função de ativação é a sigmóide, é obtida pela expressão:

$$o(t) = \frac{1}{1 + \exp(-net(t))} \tag{3.3}$$

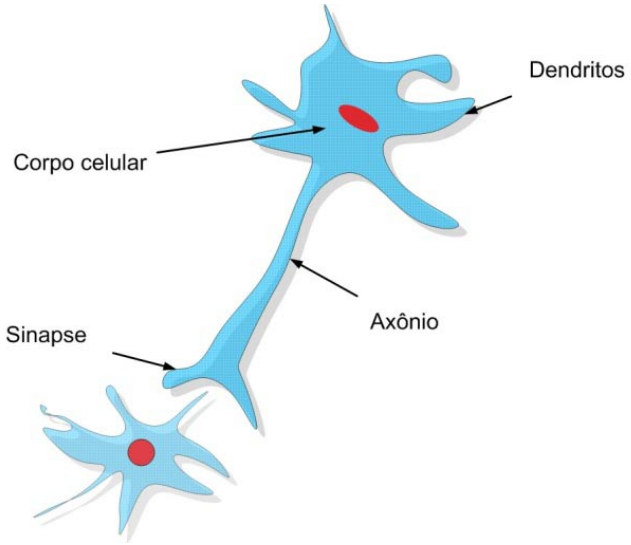


Figura 7. *Neurônio biológico.*

3.3 *Perceptron*

O modelo mais simples de redes neurais é o *perceptron* com um único neurônio. O modelo é mostrado na Figura 8. Os neurônios nessa rede usam a função *degrau* como função de ativação. O modelo simples de *perceptron* pode ser usado para tomar decisões binárias. O modelo pode também ser usado como classificador quando as entradas são linearmente separáveis. O modelo pode ser treinado e pode tomar decisões. Durante a fase de treinamento, pares dos vetores de entrada e saída são usados para treinar a rede. Com cada vetor de entrada, a saída atual é comparada com a saída desejada e o erro entra as duas é usado para atualizar os pesos e o *bias*. Em um problema com duas classes, se a amostra da entrada pertence a w_1, então a saída y é 1, e se a amostra da entrada pertence a w_2, então a saída y é 0. O modelo perceptron trabalha bem

para classes linearmente separáveis. O *bias* e os pesos definem uma superfície linear que separa as duas classes. O algoritmo pode ser descrito pelos seguintes passos:

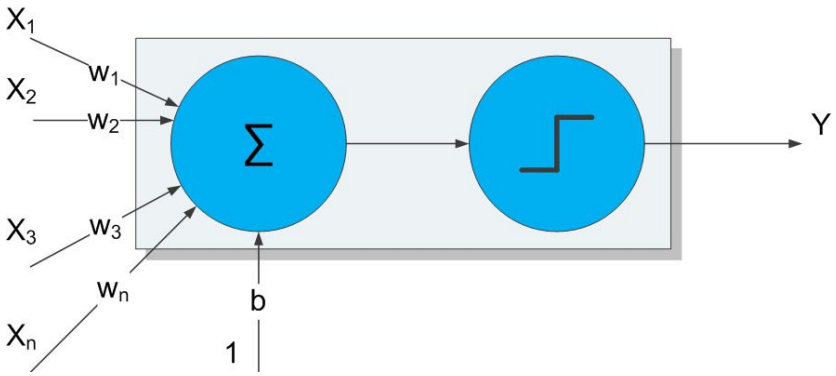


Figura 8. *Modelo perceptron com camada simples.*

Passo 1: Inicialize os pesos e o *bias* usando pequenos valores aleatórios.
Passo 2: Apresente o vetor de entrada $x = (x_1, x_2, \ldots, x_n)^T$ e obtenha a entrada net:

$$net = \sum_{j=1}^{n} (x_i w_i + b) \tag{3.4}$$

Passo 3: Calcule a saída:

$$\begin{aligned} y &= 1 \quad if\ net > 0 \\ y &= 0 \quad if\ net \leq 0 \end{aligned} \tag{3.5}$$

Passo 4: Atualize os pesos e o *bias (b)*:

$$\begin{aligned} w_i(k+1) &= w_i(k) + \varepsilon x_i \\ b(k+1) &= b(k) + \varepsilon \end{aligned} \tag{3.6}$$

Onde k representa o número de iterações e ε representa o erro entre a saída desejada e a saída atual, sendo este dado por:

$$\varepsilon = \bar{y} - y \tag{3.7}$$

Onde $\bar{y}$ é a saída desejada e y é a saída atual.

Passo 5: Para cada vetor de entrada, repetir os passos 2 e 3, até que não haja mudanças nos pesos ou o máximo número de iterações seja atingido.

Passo 6: Repita os passos 2 a 4 para todas as amostras de treinamento.

3.4 Aprendizado Supervisionado

É sinônimo de aprendizado com professor. O aprendizado deste paradigma consiste na presença de uma base de treinamento pré-classificada. O modelo *perceptron* apresentado na sessão 3.3 tem como paradigma o aprendizado supervisionado. Redes neurais que usam este paradigma aprendem pela modificação dos pesos das conexões da rede para classificar mais corretamente os

dados de treinamento. Isso permite que as redes neurais sejam aptas a generalização com extrema exatidão nas muitas situações de um conjunto de dados de treinamento para o conjunto completo de possíveis entradas.

Em ocasiões onde a base de treinamento é reduzida o bastante para não permitir a generalização, ocorre o problema conhecido como *underfitting* [1].

Em outras ocasiões, quando há ruído nos dados de treinamento, ou quando estes dados não são adequadamente representados no espaço inteiro de dados possíveis, ou até quando critérios de parada no treinamento não estão bem ajustados (permitindo, neste caso, que a rede decore os dados de treinamento), acontece o *overfitting*. Nestas situações, pode ser possível para a árvore de decisão classificar corretamente todos os dados de treinamento, mas a desempenho cai quando aplicada a dados não vistos durante o treinamento. Ou seja, se os dados do treinamento não representarem adequadamente e exatamente o conjunto inteiro de dados, a árvore da decisão que é aprendida deste conjunto pode não combinar dados não-vistos. Esse problema não se aplica somente a árvores de decisão, mas a outros métodos de aprendizado.

Na Figura 9, os pontos representam elementos de duas classes de dados: uma classe é representada por pontos pretos, e outra por pontos brancos. As duas linhas (uma azul e outra vermelha) representam duas possíveis hipóteses de distinção dos dados de treinamento. No primeiro diagrama, a linha azul é relativamente simples e classifica incorretamente alguns pontos da base de treinamento (pontos pretos acima da linha e pontos brancos abaixo da linha são erros de classificação). No mesmo diagrama, a linha vermelha, classifica corretamente todos os dados do treinamento e usa uma hipótese mais complexa (a complexidade se deve ao ruído dos dados). No próximo diagrama (superior, à direita) a linha azul é razoavelmente eficaz no conjunto completo dos dados. Essa hipótese erra em alguns casos, mas descreve bem o conjunto. O terceiro diagrama exemplifica que hipóteses complexas podem não representar completamente todo o conjunto de dados, pois apesar de acertar totalmente a classificação no primeiro diagrama, esta hipótese não serviu para um conjunto maior de dados, ocasionando em uma alta taxa de erro (neste caso, muitos pontos brancos abaixo da linha vermelha).

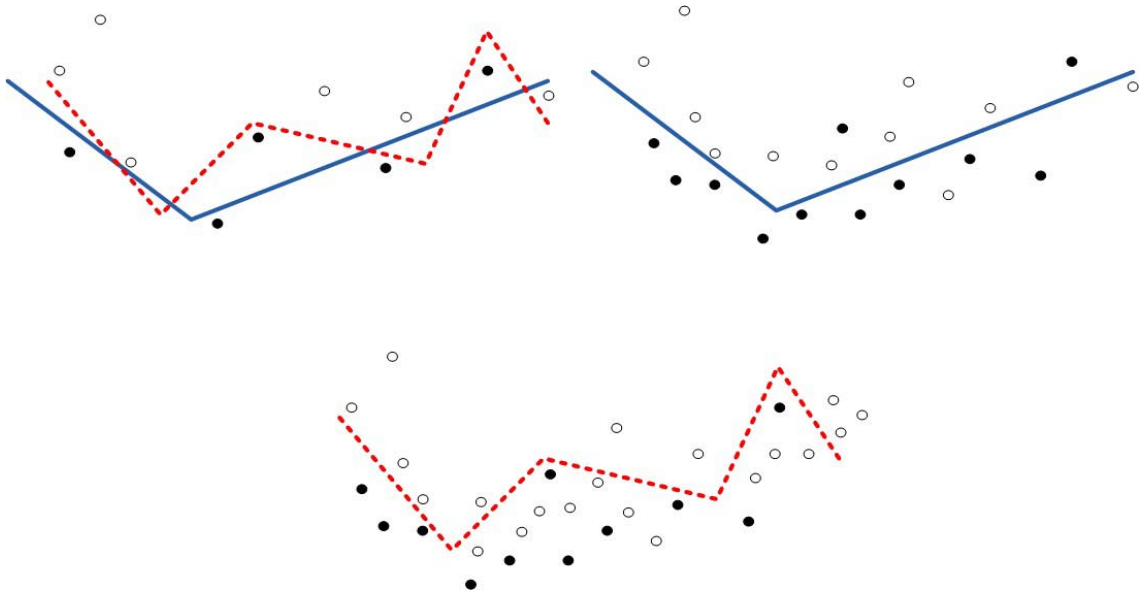


Figura 9. *Três possíveis soluções geradas por uma rede neural para um espaço de amostras.*

3.5 Aprendizado Não-supervisionado

Métodos de aprendizado não-supervisionado aprendem sem quaisquer intervenções humanas. Os mapas *Kohonen* são bons exemplos destes. Um mapa de *Kohonen* (*Kohonen map*) é uma rede que é apta a aprender a classificar um conjunto de dados de entrada sem informações sobre respostas desejadas nos dados de treinamento. Em situações onde dados precisam ser classificados ou agrupados (c*lustered*) em um conjunto de classificações, mas estas não são conhecidas, esse método é extremamente útil.

Dado um conjunto de documentos recebidos da internet, por exemplo, os mapas de *Kohonen* podem agrupar documentos similares e automaticamente fornecer uma indicação de assuntos distintos que são tratados nos documentos.

Outro método de aprendizado não-supervisionado é conhecido com aprendizado *Hebbiano* (*Hebbian learning*) e foi proposto por *Donald Hebb* em 1949. Baseado na idéia de que se dois neurônios em uma rede neural estão juntamente conectados (em ambos os lados de uma sinapse) são ativados simultaneamente, isto é, sincronamente, então a força daquela sinapse é seletivamente aumentada. Caso contrário, ou seja, se ambos os neurônios de uma sinapse são ativados assincronamente, então aquela sinapse é seletivamente enfraquecida ou eliminada.

3.6 Técnicas de Aprendizado Não-supervisionado

Nesta sessão, serão apresentadas algumas técnicas de aprendizado não-supervisionado, tais como Mapas Auto-organizáveis (*Kohonen* SOM), *Fuzzy C-means* e *K-means*.

3.6.1 Mapas Auto-Organizáveis

Um mapa auto-organizável ou rede de *Kohonen* (*self-organizing map* - SOM) é um modelo de rede neural desenvolvido na década de 80 por Teuvo Kohonen [1] [6]. Diferentemente dos outros modelos de redes neurais, redes SOM têm uma forte inspiração fisiológica. Este tipo de rede baseia-se no mapa topológico existente no córtex cerebral. Neurônios topologicamente próximos uns dos outros tentem a produzir respostas para o mesmo tipo de estímulo no córtex cerebral. Devido a esse fato, redes SOM são amplamente empregadas em reconhecimentos de padrões visuais. A idéia dos mapas auto-organizáveis é projetar dados com “n” dimensões em algo que seja mais bem entendido visualmente, como por exemplo, uma imagem com três dimensões (largura, altura e cor). Redes SOM reduzem dimensões produzindo mapas com uma ou duas dimensões comumente na qual representam similaridades entre os dados agrupando itens dos dados similares juntos. Dessa forma, redes SOM possuem duas características: reduzir dimensões e mostrar similaridades.

A rede *Kohonen* consiste de duas camadas. A primeira é a de camada de entrada e a segunda é a camada *Kohonen*. Cada unidade na camada de entrada tem uma conexão unidirecional para cada unidade da camada *Kohonen*. O método de treinamento deste tipo de rede é baseado no aprendizado competitivo, que é um tipo de aprendizado não-supervisionado de redes neurais. Neurônios na camada de *Kohonen* competem entre si quando um vetor de entrada é apresentado. Estes vetores são mostrados seqüencialmente a camada de entrada. O aprendizado em redes *Kohonen* SOM consiste em cada unidade computar o valor de melhor casamento dos vetores de pesos com o vetor de entrada. A unidade, ou neurônio, com maior valor de casamento é declarada como vencedora. Apenas à unidade vencedora é permitido aprender. Assumindo a camada de entrada como L_1 e a camada de *Kohonen* como L_2, o algoritmo está descrito a seguir:

Passo 1: Inicializar os elementos da matriz de peso W com pequenos valores aleatórios. O elemento w_{ij} representa a força da conexão entre a unidade j de L_1 e a unidade i de L_2.

Passo 2: Apresentar o vetor de entrada $x = (x_1, x_2, \dots, x_n)^T$ e computar o grau de não-combinação para cada unidade na camada L_2. O grau de não-combinação é essencialmente a distância entre os vetores w_i e x, e a unidade com menor valor de não-combinação é a vencedora da competição. O grau de não-combinação é dado por:

$$v_i = \sum_j \left(w_{ij} - x_j\right)^2 \tag{3.8}$$

Passo 3: Atualizar os vetores de pesos correspondentes à unidade vencedora e suas unidades vizinhas. A variação no peso w_{ij} para a unidade vencedora é dado:

$$\Delta w_{ij} = \alpha\left(x_j - w_{ij}\right) \tag{3.9}$$

Se a unidade não for vencedora, o valor Δw_{ij} é zero, ou seja, não houve aprendizado. Na Equação 3.9, α representa a taxa de aprendizado tal que $0 \leq \alpha \leq 1$. Essa taxa é decrementada de maneira pré-determinada. Os pesos são atualizados como:

$$w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij} \tag{3.10}$$

onde k indica o número de iterações. Os pesos são atualizados com cada amostra da entrada.

Passo 4: Repetir os passos dois e três para todas as amostras da entrada.

3.6.2 K-Means

K-means está entre os mais clássicos métodos de algoritmos de agrupamento não-supervisionado para resolução de problemas fundamentais de *clustering*.

O algoritmo tenta descobrir possíveis categorias de dados em um conjunto de objetos, de forma que haja um grupo de objetos com alguma semelhança. Nesta técnica, um *cluster* nada mais é do que uma coleção de objetos similares, e simultaneamente diferentes dos outros objetos pertencentes aos outros *clusters*. O agrupamento *K-means* pode ser considerado a mais importante abordagem de aprendizado não-supervisionado [1].

Dentre suas potenciais vantagens estão:

- Capacidade de tratar com diferentes tipos de dados;
- Capacidade de descoberta de *cluster* com formatos arbitrários;
- Necessidade mínima de conhecimento do domínio na determinação dos parâmetros de entrada;
- Robustez no tratamento de ruídos e *outliers*;
- Minimização de dissimilaridade entre dados.

Entretanto, algumas falhas devem ser superadas. O método de inicialização das médias na fase inicial é usando aleatoriedade, pois não há nenhuma especificação quanto a isso, e como os resultados dependem desses valores iniciais, freqüentemente há particionamentos subótimos encontrados. Caso o conjunto de dados próximos ao centro de algum *cluster* seja vazio, então este não poderá ser atualizado. Os resultados dependem da métrica usada na medida de dissimilaridade entre uma dada amostra e um determinado centro do *cluster* (usar desvio-padrão na normalização é uma solução popular). E por fim, o resultado depende do valor de k.

A função objetivo a ser minimizada é:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2 \tag{3.11}$$

Onde na Equação 3.11, k é o número de *clusters* e n é o número de amostras. A norma $\left\| x_i^{(j)} - c_j \right\|$ representa a distância entre o ponto $x_i^{(j)}$ e o centro do *cluster* c_j.

3.6.3 *Fuzzy C-means*

A técnica conhecida como *Fuzzy C-means* consiste em um método de agrupamento que permite que um elemento em um conjunto de dados pertença a mais de um grupo (*cluster*). FCM (*Fuzzy C-means*) ou *Fuzzy* ISODATA é usado em reconhecimento de padrões. Utiliza técnica de lógica difusa (*fuzzy*) para permitir que algo pertença a mais de um domínio, atribuindo um grau de pertinência a cada domínio, ou seja, FCM emprega lógica difusa de modo que cada dado pertença a todos os grupos com diferentes graus de pertinência, entre 0 e 1. Foi desenvolvido por Dunn [6] em 1973 e melhorado por Bezdek [6]. O objetivo deste método é minimizar a seguinte função:

$$J_m = \sum_{i=1}^{N} \sum_{j=1}^{C} u_{ij}^m \left\| x_i - c_j \right\|^2 \quad , 1 \leq m < \infty \tag{3.12}$$

Na Equação 3.12, N é o número de elementos presentes no conjunto de dados e C é o número de *clusters*, na qual deve ser conhecido previamente. O valor u_{ij}^m, na qual possui valores entre 0 e 1, é o grau de pertinência do elemento x_i no *cluster* j, x_i é o i-ésimo elemento do conjunto d-dimensional de dados medidos e c_j é o centro d-dimensional do *cluster*.

O agrupamento é feito através de uma otimização iterativa da função objetivo mostrada acima – função de dissimilaridade. Isto é feito através da atualização do grau de pertinência u_{ij}^m e dos centros, ou centróides, dos *clusters* c_j. A norma $\left\| x_i - c_j \right\|$ é a distância euclidiana entre o centróide c_j e o iésimo dado. O mínimo da função de dissimilaridade é encontrado através de duas condições. Estas são calculadas nas equações (3.13 e 3.14):

$$u_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{\left\| x_i - c_i \right\|}{\left\| x_i - c_j \right\|} \right)^{\frac{2}{m-1}}} \quad , \tag{3.13}$$

$$c_j = \frac{\sum_{i=1}^{N} u_{ij}^m \cdot x_i}{\sum_{j=1}^{C} u_{ij}^m} \tag{3.14}$$

Quando a diferença entre o valor de u da iteração atual e da anterior for menor que um critério ε (valor entre 0 e 1), pode-se parar o processo de aprendizado.

No final do processo de treinamento, haverá uma relação entre cada elemento do espaço de entrada e cada *cluster*, na qual esta relação é o grau de pertinência u, propiciando a criação de uma matriz U.

Os seguintes passos determinam o algoritmo:

Passo 1: Inicializar aleatoriamente a matriz de pertinência U, com valores entre 0 e 1.

Passo 2: Calcular os centróides através da Equação 3.14.

Passo 3: Computar a dissimilaridade entre os centróides e cada dado usando a Equação 3.12. Parar se este valor está abaixo de um limiar ε estabelecido previamente.

Passo 4: Computar um novo U usando a Equação 3.13. Ir para o passo 2.

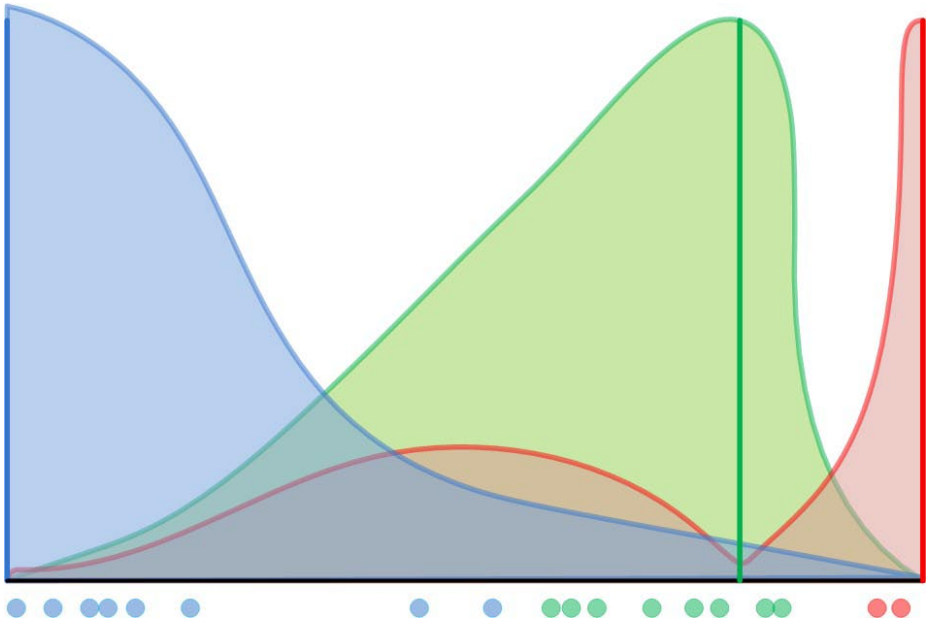


Figura 10. *Classificação resultante da aplicação da técnica Fuzzy C-means a uma base com 18 elementos. É observável a formação de tres clusters.*

Na Figura 10, observa-se na abscissa 18 elementos, na qual correspondem ao vetor de entrada. Há também 3 linhas verticais. Cada uma destas representa um *cluster*. A área abaixo de cada uma das três curvas representa o grau de pertinência de cada elemento a cada *cluster* e, quanto mais próximo do centro do *cluster*, maior o grau (próximo de 1). É interessante notar que há sobreposições entre cada curva, característica de métodos baseados em lógica difusa.

Não é certo que FCM convirja para uma solução ótima. Isto é devido à inicialização aleatória da matriz U, podendo atingir mínimo locais.

A posição inicial dos centros dos clusters influencia o desempenho da FCM. O uso de algoritmos que determinam todos os centróides (como o uso de médias aritméticas entre todos os dados, por exemplo) ou rodar a FCM várias vezes para com diferentes centróides iniciais são abordagens para a melhora a robustez da FCM.

Capítulo 4

Imagens Funcionais

Imagem funcional é na medicina diagnóstica, o estado-da-arte da análise por imagem. A ressonância magnética deu seu grande salto nesta direção através das técnicas de aquisição ultra-rápidas, que permitiram medir as variações no nível de consumo de oxigênio decorrentes do efeito *BOLD* [4]. Imagem funcional vem sendo empregada no estudo de funcionamento cerebral, podendo até mesmo fornecer informações para o planejamento cirúrgico. Utilizando a radiação ionizante, o PET possibilitou à Medicina Nuclear medir o nível de consumo de glicose no tecido.

Sua principal aplicação consiste na investigação de tumores e metástases (as metástases são resultado de uma proliferação descontrolada de células tumorais que acabam rompendo barreiras teciduais até chegar a vias de transporte pelo organismo), que naturalmente tem aumentado o consumo de glicose devido ao maior nível metabólico. Mais recentemente, o PET Dinâmico tem permitido avaliar o consumo de glicose em função do tempo e medir de forma mais precisa as regiões mais ativas.

Neste capítulo, serão apresentadas duas técnicas de estudo e análise de imagens funcionais: PET e fMRI.

4.1 Introdução

A possibilidade de visualizar o interior do organismo de uma forma não invasiva, sempre representou uma fonte de interesse para o homem. A concretização desta possibilidade foi, seguramente, um dos maiores feitos da humanidade. A descoberta do raio-X por *Röntgen*, em 1895, como um fenômeno físico capaz de propiciar as primeiras inferências sobre o interior do corpo humano, e constituiu-se sem dúvida o grande passo. Depois desse advento, obtiveram-se as primeiras imagens reveladoras das estruturas internas do corpo humano, tais como órgãos e esqueleto [14].

A ciência evoluiu, trouxe conhecimento sobre o fenômeno do raio-X, proporcionou a evolução tecnológica e o domínio de sua geração. Desenvolveram-se equipamentos novos e melhores para fornecer imagens com qualidade superior. Mas, ainda assim, as imagens eram projeções planas do volume, pois representavam o mapa da atenuação de raio-X que atravessava o objeto em estudo ao longo de uma única direção.

A tomografia por raio-X foi o passo seguinte na direção do mapeamento do interior do corpo humano. Sua evolução dependeu principalmente do desenvolvimento de algoritmos numéricos responsáveis pela reconstrução matemática, de equipamentos capazes de obter vistas

parciais (projeções) em toda a volta do objeto, do desenvolvimento tecnológico que integrava os dispositivos eletromecânicos aos computacionais e do desenvolvimento dos tubos de raio-X.

Com todas as vantagens obtidas na aquisição de imagens de tomografia computadorizada, considerando inclusive os tempos atuais, há uma restrição nesse método diagnóstico que o torna inapto para determinados fins. Pelo fato de sua imagem significar o mapa de atenuação do objeto, o que ali se observa corresponde eminentemente à anatomia. Só se podem detectar alterações funcionais que provocam alteração anatômica (entenda-se, alteração na densidade do tecido) E, mesmo nesse caso, não é possível avaliar o funcionamento dinâmico do tecido, isto é, seu nível de metabolismo.

Não há dúvidas de que os métodos diagnósticos capazes de fornecer esse tipo de informação são complementos importantes, muitas vezes até mais importantes.

A descoberta da radioatividade ocorreu aproximadamente no mesmo período da história da ciência em que se descobriu o raio-X. *Henry Becquerel* e o casal Curie foram pioneiros na descoberta e na pesquisa da radioatividade. Eles observaram que o urânio emitia radiação com alto poder de penetração.

Algumas décadas mais tarde, *Hal Anger* criou um equipamento capaz de realizar imagens da distribuição de substâncias metabolizadas nos órgãos. Para tanto, as substâncias têm, adicionadas à sua estrutura molecular, um isótopo radioativo emissor de radiação gama. O princípio de funcionamento do equipamento desenvolvido por *Anger*, conhecido inicialmente por Câmara *Anger* e, mais tarde, por Câmara de Cintilação, era transformar a energia da radiação gama em energia luminosa num cristal (fenômeno chamado de cintilação, daí o último nome dado ao equipamento). Um conjunto de fotomultiplicadoras transforma a energia luminosa em elétrica e, em seguida, um conjunto de circuitos forma a imagem. Esse equipamento representou um grande avanço para a Medicina Diagnóstica conhecida como Medicina Nuclear.

Em 1946, um grupo liderado por *Felix Bloch*, da *Stanford University*, e outro liderado por *Edward Purcell*, do *Massachusetts Institute of Technology* e, mais tarde, da *Harvard University*, publicaram suas descobertas em ressonância nuclear magnética. Mas somente em 1973 dois grupos independentes, um liderado por Paul Lauterbur, Professor de Química da *New York State University*, e outro por *Peter Mansfield*, Professor da *Nottingham University*, fizeram as primeiras imagens utilizando ressonância magnética. Novamente, o avanço tecnológico foi, sem dúvida, fundamental para o desenvolvimento desta técnica, que utiliza o fenômeno físico da radiofreqüência para a formação da imagem, apesar do nome ‘nuclear’ [14]. As imagens obtidas são as respostas do tecido à estimulação por radiofreqüência e dependem basicamente da concentração de átomos de hidrogênio em diferentes regiões, bem como da forma como os átomos de hidrogênio estão ligados nas variadas substâncias do corpo, como gordura e água, por exemplo. Em qualquer um desses métodos, tomografia por raio-X, tomografia por medicina nuclear ou ressonância magnética (método tomográfico no princípio, que hoje fornece diretamente cortes do objeto), pode-se destacar uma seqüência de desafios ocorridos em seqüência quase cronológica.

O primeiro desafio foi conseguir formar uma imagem a partir do fenômeno físico pertinente. Em seguida, como conseguir formar uma imagem com boa resolução espacial, isto é, capaz de revelar detalhes do objeto em dimensões cada vez menores. Finalmente, como derivar tecnologias, softwares ou hardwares que, empregando os mesmos fenômenos físicos, fornecessem informações sobre a dinâmica de funcionamento do objeto em estudo. Com efeito, todos esses problemas ainda são desafios; mas não há como negar que já se evoluiu muito, principalmente nos dois primeiros casos. Hoje, o grande desafio, que tem mostrado grande número de possibilidades, é conseguir obter informações dinâmicas do funcionamento do corpo humano. Nessa linha, duas grandes técnicas de imagem vêm se destacando: o PET (*Positron Emission Tomography*) dinâmico e a fMRI (*Functional Magnetic Resonance Imaging*). A

primeira deriva de a Medicina Nuclear e a segunda da Ressonância Magnética. Para a primeira, desenvolveu-se um equipamento destinado à aquisição das imagens e, para a segunda, tecnologias de hardware e de software, utilizando a mesma base de equipamento.

4.2 PET e PET Dinâmico

O PET (*Positron Emission Tomography*) é uma técnica diagnóstica desenvolvida com o objetivo de mapear o consumo de glicose do tecido, tornando-a assim uma eficiente ferramenta para a detecção de tumores. O isótopo comumente utilizado é o flúor ligado à desoxiglicose, conhecido como FDG. O flúor 18 é um emissor de pósitrons, partícula que rapidamente interage com os elétrons do meio (interação denominada aniquilação), produzindo um par de fótons gama, com energia de 511 KeV cada, que viajam a partir do ponto de aniquilação numa mesma direção, mas em sentidos opostos. A evolução dos equipamentos, traduzida, sobretudo na sensibilidade de detecção de eventos e na capacidade de armazenamento e processamento de dados, tem proporcionado o desenvolvimento do PET dinâmico. Obtém-se uma seqüência de cortes tomográficos em função do tempo a partir do instante em que se administra o FDG ao paciente.

4.3 fMRI

O desenvolvimento tecnológico no campo da radiofreqüência e o aumento do conhecimento dos fenômenos eletromagnéticos e quânticos, incorporados aos equipamentos, tornaram possível detectar, por meio das imagens de ressonância magnética, pequenas alterações na intensidade do fluxo sangüíneo nas regiões do cérebro associadas a funções específicas. Daí vem o nome fMRI (*functional Magnetic Resonance Imaging*). Quando se executa alguma atividade, a região do cérebro ativada aumenta o consumo de ATP (trifosfato de adenosina), gerando uma maior demanda de oxigênio e glicose, suprida pelo aumento do fluxo sangüíneo na região cerebral. Desse modo, cresce o número de moléculas de hemoglobina, que transportam o oxigênio através dos vasos capilares e transformam-se em desoxi-hemoglobina, ao liberarem O_2. Dá-se assim um aumento da concentração de desoxi-hemoglobina na região. Esta, por sua vez, possui propriedades paramagnéticas, que reforçam localmente os efeitos do campo magnético externo. Segue-se um aumento no fluxo e no volume sangüíneo para suprir a demanda de oxigênio, reduzindo a concentração de desoxi-hemoglobina para aquém do nível basal. Este fenômeno é denominado efeito Bold (*Blood Oxygenation Level Dependent effect*)(13). Pode-se constatar, portanto, que há variação da intensidade de resposta na região de acordo com o momento que se adquire a imagem.

Em outras palavras, fMRI, é uma técnica que mede a atividade cerebral. Ela é realizada pela detecção das mudanças na oxigenação e fluxo sangüíneo que ocorre em resposta a atividade neural – quando uma área cerebral esta mais ativa ela consome mais oxigênio e para satisfazer demanda há um aumento do fluxo sangüíneo para a área ativada.

Essa possibilidade vem promovendo inúmeros estudos para entender a dinâmica cerebral. Muitos deles se realizam analisando as áreas do cérebro ativadas quando o ser humano executa tarefas.

Na Figura 11, há uma imagem gerada pela reconstrução tridimensional de imagens de ressonância magnética funcional, adquiridas durante a prática de atividades mobilizadoras da inteligência criativa.

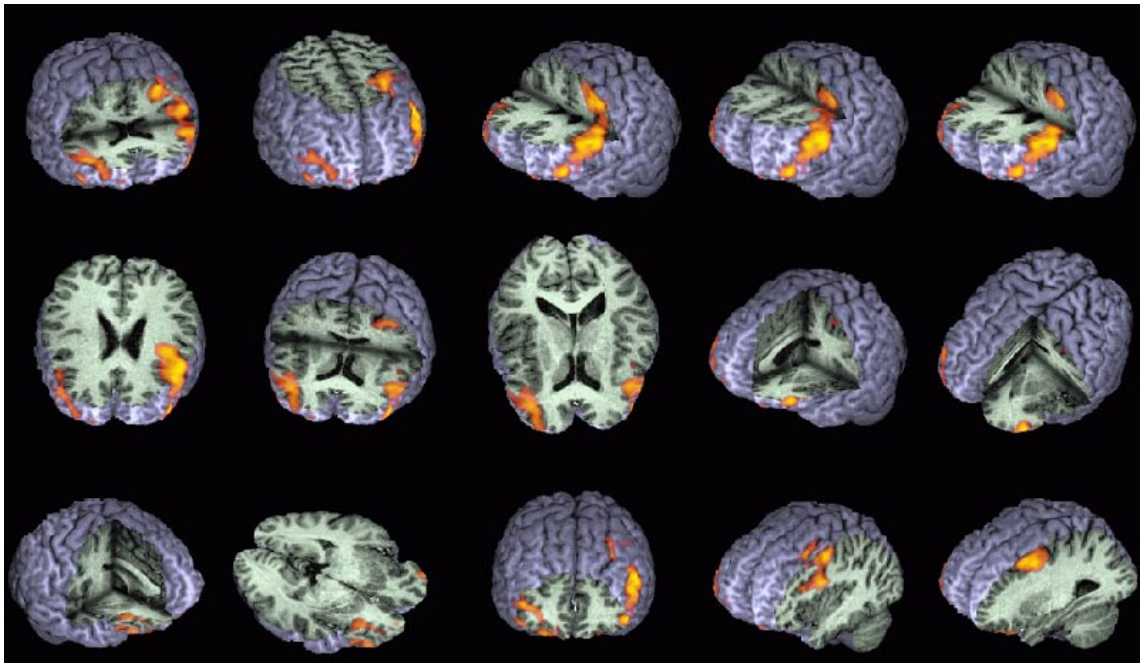


Figura 11.*Imagem gerada pela reconstrução tridimensional de imagens de ressonância magnética funcional, adquiridas durante a prática de atividades mobilizadoras da inteligência criativa. Áreas com tonalidade amarela indicam as áreas cerebrais ativadas durante a atividade [14].*

A aquisição das imagens é feita equipamento moderno: um *scanner* MRI. Tal equipamento está abrigado em um recinto especial, na qual deve resistir e isolar o enorme campo magnético produzido pelo *scanner*. Um *scanner* de pesquisa típico (tal como o scanner do Centro FMRIB) tem um campo magnético com três teslas (T), em torno de 50.000 vezes maior que o campo magnético terrestre. O campo magnético dentro do *scanner* afeta o núcleo magnético dos átomos. Normalmente os núcleos atômicos estão aleatoriamente orientados, mas sobre a influência de um campo magnético o núcleo é alinhado na direção do campo. Quanto mais forte for o campo, maior será o grau de alinhamento. Quando apontados na mesma direção, os minúsculos sinais magnéticos dos núcleos individuais adicionam coerência, resultando em um sinal grande o suficiente para ser medido. Em fMRI, é o sinal magnético do núcleo de hidrogênio presente na água (H_2O) que é detectado.

A chave para a IRM é que o sinal do núcleo hidrogênio varia em intensidade, dependendo do ambiente. Isso proporciona um meio de discriminar entre massa cinzenta, massa branca e fluido espinhal nas imagens estruturais do cérebro.

A Figura 12 mostra um aparelho de fMRI, enquanto a Figura 13 mostra a imagem sagital de um cérebro humano proveniente de um aparelho de fMRI.

Figura 12.*Foto do Varian 3T Scanner [14].*

Figura 13.*Imagem sagital de um cérebro humano, proveniente de um aparelho de fMRI [14].*

Neste trabalho, tem-se como um conjunto com um total de 48 imagens de extensão BMP, sendo 24 delas referente ao movimento do dedo indicador direito e 24 referente à inatividade. Na Figura 13 há uma amostra referente à ativação do dedo indicador direito.

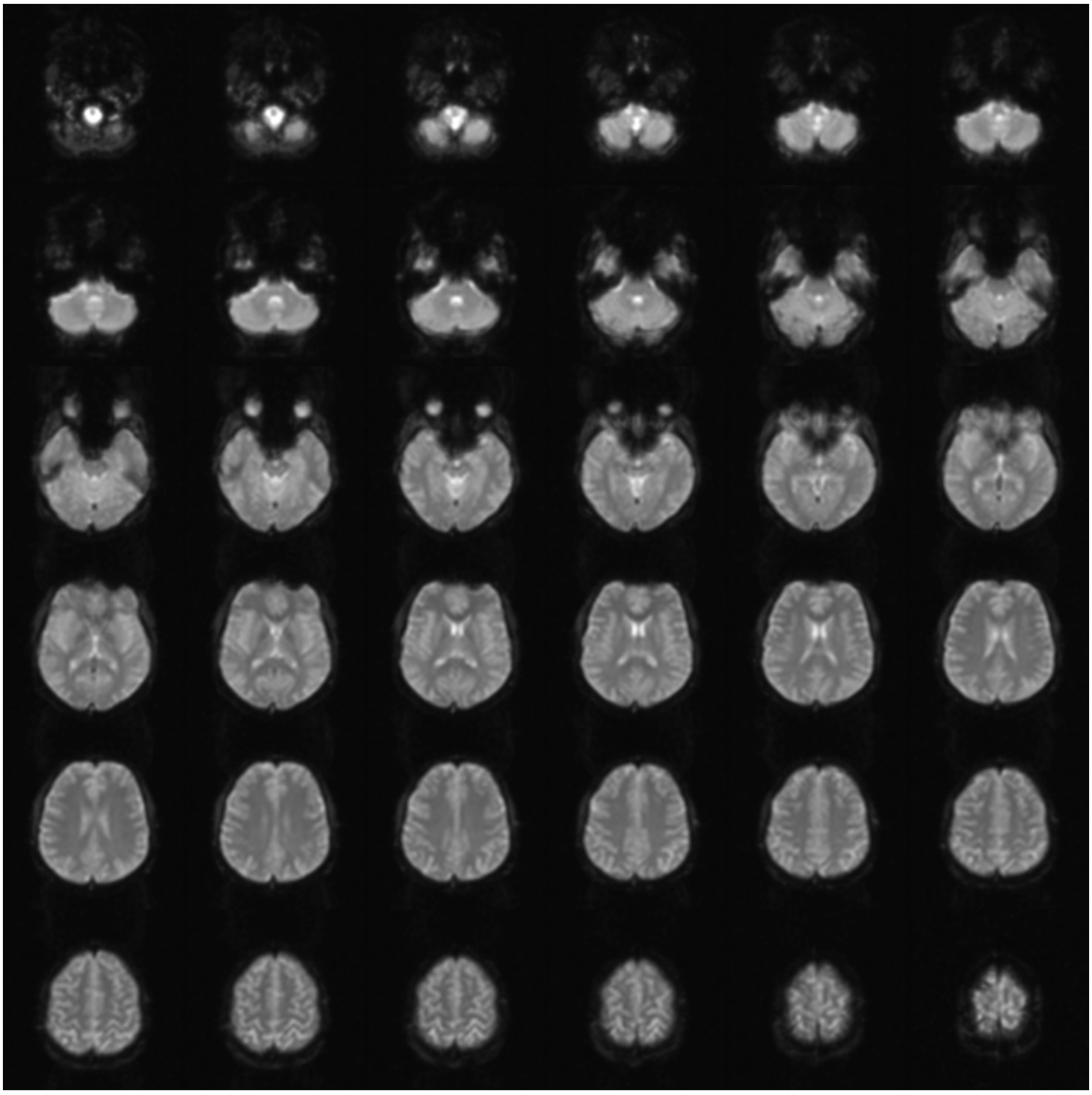


Figura 14.*Imagem proveniente de um aparelho de fMRI referente a período de movimentação do dedo indicador direito.*

4.3.1 Métodos Clássicos de Detecção em fMRI

Nesta subseção serão apresentados dois métodos para a detecção de *voxels* ativados: o método da *Subtração* e do *Teste t de Student* (*Student's t-test*).

4.3.1.1 Subtração

O método mais simples para a detecção de *voxels* ativados. Consistem em calcular a média de todas as imagens adquiridas durante os períodos de ativação e subtrair todas as imagens adquiridas em repouso [Javier]. Para $j = 1 \ldots J$ simplesmente é aplicada a seguinte fórmula:

$$s_j = \overline{Y_{j(ativa)}} - \overline{Y_{j(repouso)}} \tag{4.1}$$

O método de subtração é simples, e permite visualizar as áreas de ativação neurais mais claras que as não ativadas, em uma imagem em tons de cinza. A desvantagem fundamental é a sensibilidade ao movimento da cabeça gerando artefatos na imagem.

A imagem obtida é um mapa de ativação, mas não é um mapa estatístico. Em um mapa estatístico, uma imagem onde o valor de cada *voxel* seja uma estatística associada ao método de detecção, nos permitiria decidir se o *voxel* está ativado e também determinar a probabilidade do erro na classificação.

4.3.1.2 *Student's t-test*

Um teste-t é algum teste de hipótese estatística na qual o teste estatístico tem uma distribuição de *Student* se a hipótese nula é verdade.

No caso da análise estatística de imagens de ressonância magnética funcional, para cada *pixel* da imagem calcula-se uma estatística baseada na série temporal deste *pixel*. A imagem obtida é um mapa estatístico paramétrico baseado na distribuição *t* de *Student*, permitindo assim determinar se o *pixel* está ativado ou não ativado e o nível de significância do resultado.

O princípio desse teste é considerar duas amostras, a primeira formada pela série temporal em ativação e a segunda pelos valores da série temporal correspondente a repouso ou a condição alternativa de controle. Consideramos os dados não-emparelhados (populações não-correlacionadas), com distribuição normal, e o desvio padrão igual para as duas populações. Calcula-se a variância amostral combinada (*pooled variance*), $S_{p(j)}{}^2$, para o *pixel* j como uma média ponderada das variâncias amostrais, usando como pesos os graus de liberdade de cada uma, resultando em:

$$S^2_{p(j)} = \frac{(N_{j(ativa)} - 1)S^2_{j(ativa)} + (N_{j(repou)} - 1)S^2_{j(repou)}}{N_{j(ativa)} + N_{j(repou)} - 2}$$

onde $S^2_{j(ativa)}$ e $S^2_{j(repou)}$ são as variâncias amostrais dos conjuntos, $Y_{j(ativa)}$ e $Y_{j(repou)}$ respectivamente. $N_{j(ativa)}$ e $N_{j(repou)}$ são o número de pontos dos dois conjuntos correspondentes a ativação e repouso.

O teste estatístico t_j do *pixel i* será realizado através de:

$$t_j = \frac{\overline{Y_{J(ativa)}} - \overline{Y_{J(repou)}} - (\Delta_0)}{S_{p(j)}\sqrt{\frac{1}{N_{j(ativa)}} + \frac{1}{N_{j(repou)}}}},$$

Onde $\overline{Y_{J(ativa)}}$ e $\overline{Y_{J(repou)}}$ são médias amostrais de $Y_{j(ativa)}$ e $Y_{j(repou)}$ respectivamente. É designado H_0 a hipótese nula ou hipótese a ser testada, e H_1 a hipótese alternativa. O teste-t permite aceitar ou rejeitar a hipótese nula.

O valor-p ("*p-value*") é o valor de significância estatística observado, é o menor nível de significância para o qual H_0 seria rejeitado. Uma vez que o valor p foi determinado para um conjunto de dados se:

$$p \leq \alpha \Rightarrow Rejeitar\ H_0\ para\ o\ nível\ \alpha$$
$$p > \alpha \Rightarrow Não\ rejeitar\ H_0\ para\ o\ nível\ \alpha$$

As vantagens mais importantes desse método são a redução dos artefatos de movimento e a determinação do nível de significância estatística, para decidir se um *pixel* será classificado como ativado ou não ativado.

Capítulo 5

Morfologia Matemática

O estudo morfológico concentra-se na estrutura geométrica (ou forma) das imagens. Áreas como realce, filtragem, segmentação, esqueletizaçao e outras áreas afins, são campos de aplicação onde a morfologia pode ser usada. Morfologia é a forma e estrutura de um objeto ou os arranjos e inter-relacionamentos entre as partes de um objeto [3].

Morfologia digital é uma maneira para a descrição ou análise a forma de um objeto digital. A morfologia digital é uma ciência relativamente recente, pois só os computadores digitais permitiram seu uso na pratica. [4].

A idéia de morfologia digital é que uma imagem consiste de um conjunto de *pixels* que são reunidos em grupos tendo uma estrutura bidimensional (forma). Certas operações matemáticas em conjuntos de pixels podem ser usadas para ressaltar aspectos específicos das formas permitindo que sejam contadas ou reconhecidas.

O princípio da morfologia é extrair as informações relativas à geometria e a topologia de um conjunto desconhecido (no caso, uma imagem) pela transformação através de outro conjunto bem-definido, chamado *elemento estruturante*, ou seja, consiste em extrair de uma imagem desconhecida a sua geometria através da utilização da transformação de outra imagem completamente definida. Com isso, torna importante ao contexto a utilização de teoria dos conjuntos, pois esta é a base utilizada na morfologia, assim é com esta teoria que será descrita e apresentada uma imagem. Por exemplo, a definição de um vetor bidimensional onde serão expostas as coordenadas (x, y) para sua representação gráfica.

As operações básicas da morfologia digital são a *erosão*, em que *pixels* que não atendem a um dado padrão são apagados da imagem, e *dilatação*, em que uma pequena área relacionada a um *pixel* é alterada para um dado padrão. Todavia, dependendo do tipo de imagem sendo processada (preto e banco, tons de cinza ou colorida), a definição destas operações muda, assim cada tipo deve ser considerado separadamente.

A Morfologia Matemática se caracteriza por um conjunto específico de operações sobre imagens enquanto conjuntos de *pixels*. Podemos dividi-las em operações realizadas sobre imagens binárias e operações realizadas em imagens em tons de cinza e coloridas. Serão mostradas primeiras as operações binárias. Para cada operação binária existe uma correspondente para tons de cinza e imagens em cores.

5.1 Dilatação Binária (⊕)

A dilatação, também chamada de *dilação*, é uma transformação morfológica que combina dois conjuntos usando adição vetorial. Seu símbolo é ⊕. Como o nome diz, o resultado será uma imagem dilatada ou expandida.

A dilatação de um conjunto A pelo conjunto B e definida por:

$$A \oplus B = \{ c \mid c = a + b, a \in A, b \in B\}, \tag{5.1}$$

onde A representa a imagem sendo operada e B é um segundo conjunto onde é chamado *elemento estruturante* e sua composição define a natureza especifica da dilatação, sendo assim a dilatação expande uma imagem.

Ela pode ser representada pela união $A \oplus B = A \cup B$.

Seja o conjunto $A = \{(0,1),(1,1),(2,1),(2,2),(3,0)\}$ *e* $B = \{(0,0),(0,1)\}$, então o resultante da dilatação é:

$$A \oplus B = \{A + \{(x_1 \in B)\}\} \bigcup \{A + \{(x_2 \in B)\}\}$$

$$A \oplus B = \{(0,1),(1,1),(2,1),(3,0),(0,2),(1,2),(2,2),(2,3),3,1\}$$

A seguir, um exemplo ilustrativo demonstrando a operação para os conjuntos apresentados ($A = \{(0,1),(1,1),(2,1),(2,2),(3,0)\}$ *e* $B = \{(0,0),(0,1)\}$).

Na Figura 15, há uma representação no formato matricial dos conjuntos A e B.

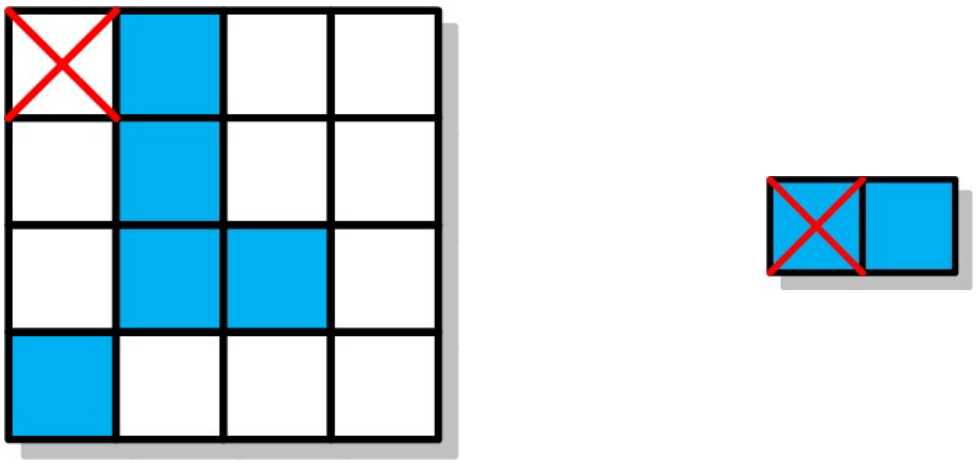
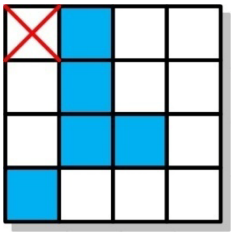


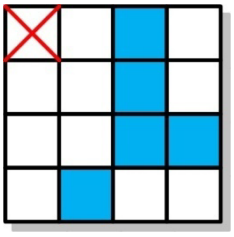
Figura 15.*Imagens utilizadas nos exemplos de dilatação e erosão. A imagem da direita representa o conjunto* $\boldsymbol{A} = \{(\mathbf{0},\mathbf{1}),(\mathbf{1},\mathbf{1}),(\mathbf{2},\mathbf{1}),(\mathbf{2},\mathbf{2}),(\mathbf{3},\mathbf{0})\}$*, enquanto a da direita, o elemento estruturante* $\boldsymbol{B} = \{(\mathbf{0},\mathbf{0}),(\mathbf{0},\mathbf{1})\}$.

$$A + \{(0,0)\} = \{(0,1),(1,1),(2,1),(2,2),(3,0)\}$$



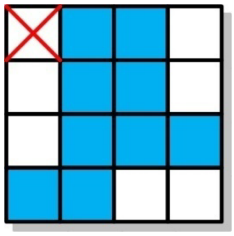
$$(0,1) + (0,0) = (0,1)$$
$$(1,1) + (0,0) = (1,1)$$
$$(2,1) + (0,0) = (2,1)$$
$$(2,2) + (0,0) = (2,2)$$
$$(3,0) + (0,0) = (3,0)$$

$$A + \{(0,1)\} = \{(0,2),(1,2),(2,2),(2,3),(3,1)\}$$



$$(0,1) + (0,1) = (0,2)$$
$$(1,1) + (0,1) = (1,2)$$
$$(2,1) + (0,1) = (2,2)$$
$$(2,2) + (0,1) = (2,3)$$
$$(3,0) + (0,1) = (3,1)$$

$$A \oplus B = \{(0,1),(1,1),(2,1),(3,0),(0,2),(1,2),(2,2),(2,3),3,1\}$$



O *pixel* marcado com um "x" representa a origem (0,0) da imagem. É notável na imagem resultante a expansão da imagem original com as dimensões do elemento estruturante.

5.2 Erosão Binária (⊖)

A erosão, basicamente encolhe a imagem e pode ser vista como uma transformação morfológica que combina dois conjuntos de usando vetores de subtração. Seu símbolo é ⊖. Ela é expressa como a intersecção de A e B. assim é definido $A \ominus B = A \cap B$

A erosão da imagem A pelo elemento estruturante B pode ser definida como:

$$A \ominus B = \{ x \mid x + b \in A \ \forall \ b \in B\} \tag{5.2}$$

Seja o conjunto $A = \{(0,1),(1,1),(2,1),(2,2),(3,0)\}$ *e* $B = \{(0,0),(0,1)\}$, então o resultante da erosão é:

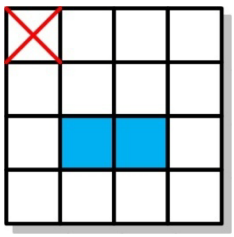
$$A \ominus B = \{(2,2),(2,3)\}$$

A seguir, um exemplo ilustrativo demonstrando a operação para os conjuntos apresentados ($A = \{(0,1),(1,1),(2,1),(2,2),(3,0)\}$ *e* $B = \{(0,0),(0,1)\}$).

$$A\ \{(0,1),(1,1),(2,1),(2,2),(3,0)\}$$
$$B\ \{(0,1),(1,1)\}$$

$B_{(0,1)} = \{(0,1),(0,2)\}$,	Onde o conjunto $\{(0,1),(0,2)\}$ não pertence à imagem A, e por isso não fará parte da imagem resultante
$B_{(1,1)} = \{(1,1),(1,2)\}$	Onde o conjunto $\{(1,1),(1,2)\}$ não pertence à imagem A, e por isso não fará parte da imagem resultante
$B_{(2,1)} = \{(2,1),(2,2)\}$	Neste caso, o conjunto $\{(2,1),(2,2)\}$ faz parte do conjunto A, e desta maneira, fará parte da imagem final
$B_{(2,2)} = \{(2,2),(2,3)\}$	Onde o conjunto $\{(2,2),(2,3)\}$ não pertence à imagem A, e por isso não fará parte da imagem resultante
$B_{(3,0)} = \{(3,0),(3,1)\}$	Onde o conjunto $\{(3,0),(3,1)\}$ não pertence à imagem A, e por isso não fará parte da imagem resultante

$$A \ominus B = \{(2,2),(2,3)\}$$



5.3 Abertura Binária (°)

A abertura em geral suaviza o contorno de uma imagem, quebra estreitos e elimina proeminências delgadas, a operação de abertura e usada também para remover ruídos da imagem. A abertura de um conjunto A por elemento estruturante B e denotado A ° B e definida como:

$$A \circ B = (A \ominus B) \oplus B \tag{5.3}$$

A aplicação de uma erosão imediatamente seguida de uma dilatação usando o mesmo elemento estrutural é uma operação de abertura, ela tende a abrir pequenos vazios ou espaços entre objetos próximos numa imagem. Em outras palavras uma abertura e uma erosão seguida de uma dilatação usando um mesmo elemento estruturante, relembrando que a erosão acha todos os lugares onde o ajuste do elemento estruturante esta dentro a imagem, mas isto somente marca esta posição a origem de um elemento.

Entretanto, a sucessão de uma erosão por uma dilatação, é preenchido de brancos os lugares onde o ajuste do elemento estruturante estiver dentro do objeto.

Na Figura 16, há uma ilustração da operação de abertura em uma imagem.

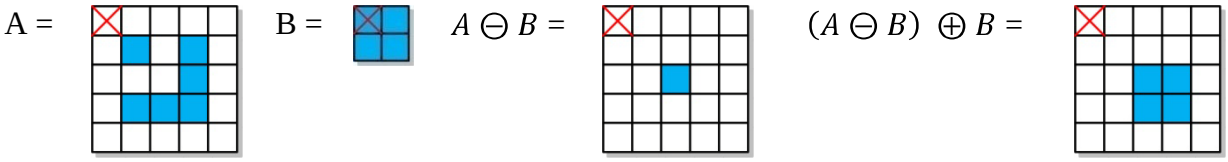


Figura 16.*Figura ilustrativa demonstrando a operação de abertura.*

5.4 Fechamento Binário (·)

O fechamento funde pequenas quebras, alarga golfos estreitos, elimina pequenos orifícios. Se uma abertura cria pequenos vazios na imagem, um fechamento irá preencher ou fechar os vazios. Estas operações podem remover muitos dos *pixels* brancos com ruídos, ou seja, basicamente o fechamento é igual à abertura só que primeiramente é feita a dilatação e depois a erosão. Assim, esta operação se define como:

$$A \cdot B = (A \oplus B) \ominus B \tag{5.3}$$

Na Figura 17, há uma ilustração da operação de fechamento em uma imagem.

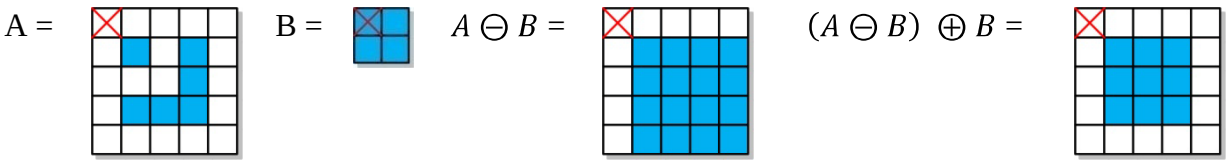


Figura 17.*Figura ilustrativa demonstrando a operação de fechamento.*

Capítulo 6

Estudo de Caso: Detecção de Ativações em fMRI

Para a concretização deste trabalho foi necessário o estudo de métodos de aprendizado não-supervisionado. Entre inúmeros métodos, *K-Means*, *Kohonen SOM* e *Fuzzy C-means* foram abordados neste trabalho, sendo o último implementado.

As duas técnicas foram implementadas utilizando paradigmas de programação para arquiteturas paralelas, sendo C a linguagem de programação escolhida para isso, pois permite o desenvolvimento de programas para ambiente paralelo além de oferecerem um bom desempenho.

A implementação das técnicas de maneira paralelizada não necessitará de grandes ajustes, pois os algoritmos já são por natureza, paralelos.

Foi implementado também a técnica de detecção de ativação *Student's t-test*, utilizando as mesmas técnicas já mencionadas.

Com o objetivo de melhorar o pré-processamento dos dados de entrada – imagens funcionais de ressonância magnética – para os algoritmos acima citados, foi escolhido o uso de Morfologia Matemática [3] [4] devido ao fato de lidar naturalmente com problemas de sobreposição, pois preservam a inclusão de objetos, ao invés das relações de soma presentes nos processamentos lineares, como filtros de convolução. A Morfologia Matemática é interessante, como descrito no capítulo 5, porque permite informações de vizinhança de cada elemento (contextualmente, *pixels*) no resultado e não simplesmente uma média ponderada de valores. Especificamente, as operações de *abertura* e *fechamento* de Morfologia Matemática foram foco nesta parte do trabalho, ambas sendo implementadas.

Como mencionado, este trabalho objetiva a experimentação e comparação de algoritmos de detecção de ativações em fMRI em arquitetura uniprocessada e em *clusters*. Houve então a necessidade de estudar soluções comerciais e *open-source* para a implementação de tais *clusters* usando PCs.

Houve a implantação do *cluster* escolhido em um conjunto composto de 4 (quatro) computadores em uma LAN, como infra-estrutura de rede.

6.1 Abordagens

Nesta seção serão apresentadas as duas abordagens realizadas neste trabalho: a implementação do *Student's t-test* e a implementação de uma rede *Fuzzy C-means* com expansão de bandas para aumentar e melhorar a base de dados. O aplicativo que implementa a técnica FCM e expansão de bandas é chamada foi batizada como *2cann*.

6.1.1 Implementação do *Student's t-test* em *clusters*

Com o objetivo de melhorar o desempenho no processo de análise de imagens funcionais, optou-se pela implementação de alguma técnica de análise em ambiente paralelo. A técnica escolhida foi o *Student's t-test* no *cluster openMosix*.

Inicialmente, houve a necessidade de adaptar a abordagem do problema para que ele pudesse ser distribuído entre vários processos e, assim, ser realizado em paralelo. Para isso, foram delegadas faixas de linhas das matrizes, que compunham as imagens, para cada processo, sendo este responsável pela aplicação do *Student's t-test* para cada vetor de *pixel* que compõe a linha. A Figura 18 ilustra a relação entre os processos e os dados de entrada, bem como a maneira como foram divididas as imagens para aplicação do *Student's t-test*.

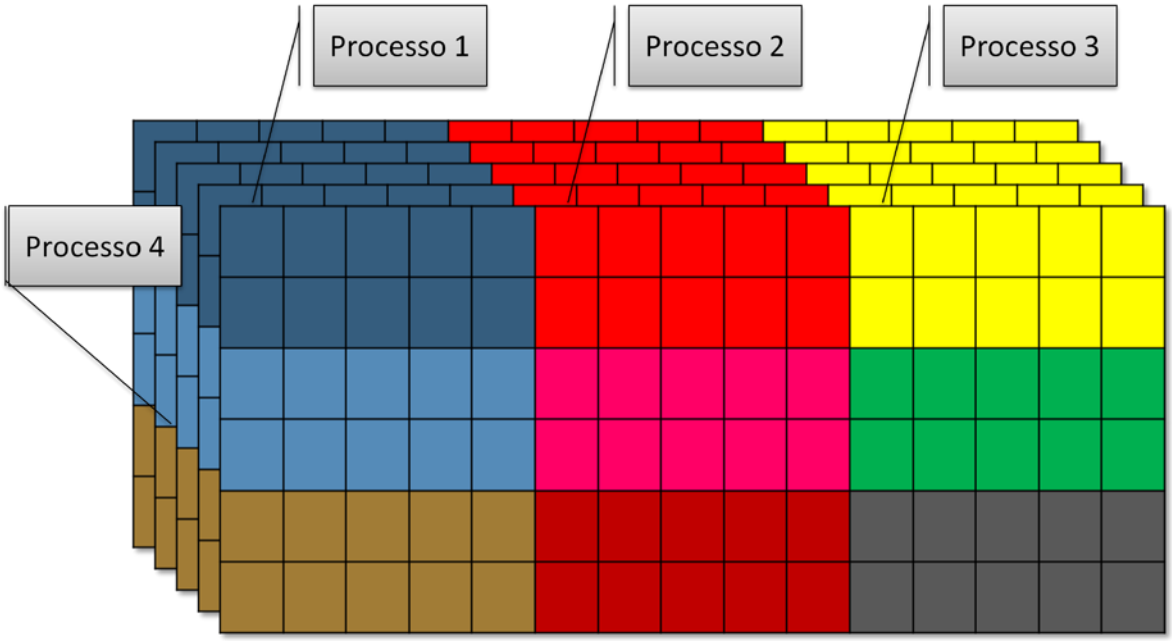


Figura 18. *Bloco de imagens para cada processo.*

Cada processo é criado como o uso da chamada de sistema *fork*, presente nos sistemas operacionais derivados do UNIX, tais como Linux, Solaris, HP-UX, Minix, Mac OS e BSD, na qual simplesmente copia o processo "pai".

O *openMosix*, para cada novo processo gerado, tenta migrá-lo para o nodo com menos carga, sem necessidade de intervenção do usuário. Para a implantação do *openMosix*, foi usado a distribuição Linux *Cluter Knoppix*, na qual não necessita de instalação alguma, pois roda direto do CD. Também não foi necessário a configuração do *openMosix*, bastando apenas a iniciação do CD e, posteriormente da aplicação a ser estudada, que no caso foi a implementação do *Student's t-test*.

De um modo geral, cada *pixel* de uma imagem pertencente ao grupo das que representam o cérebro em estado de atividade e seus correspondentes nas outras imagens do mesmo grupo é agrupado em um vetor. O mesmo é realizado para as imagens que ilustram o cérebro em um estado de relativa inatividade. Com isso, tem-se dois vetores de 24 posições, um para as imagens ativas e outro para as inativas, sendo cada uma composta de valores variando entre 0 e 255, que representam o valor RGB do *pixel* na imagem em questão. Logo, como as dimensões das imagens são de 768 x 768 pixels, há um total de 48 x (768 x 768) vetores.

Para determinar se um *pixel* está ativo ou não, são levantadas duas hipóteses: H_0 e H_1, hipótese nula e hipótese alternativa, respectivamente. Em seguida, aplica-se o *Student's t-test* para os dois vetores de 24 posições, representando o agrupamento de todos os pixels em uma mesma posição para ambos os conjuntos de imagens. O valor *p*, menor nível de significância para o qual H_0 seria rejeitada, obtido da aplicação do *Student's t-test* é comparado a uma constante ρ chamada nível de significância. Caso $p \leq \rho$, a hipótese H_0 é rejeitada para o nível ρ, ou seja, ele está ativo. Caso contrário, o pixel é tido como inativo.

A Figura 19 é um exemplo de imagem que é usada como entrada para o problema.

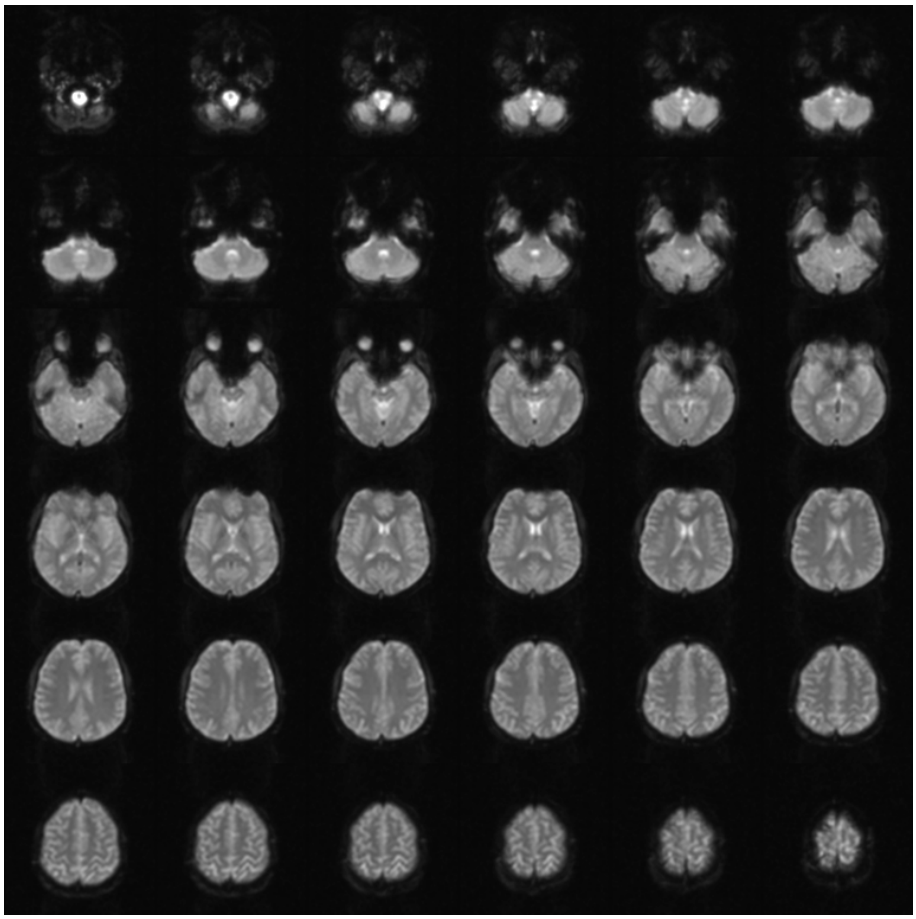


Figura 19.*Imagem de ressonância apenas com informação anatômica.*

Para essa abordagem, determinou-se que da aplicação do método proposto resultaria uma matriz 768 x 768 com valores 0 (zero) ou 1 (um). O valor 1 (um) em uma posição informa que o *pixel* em questão para a imagem original está ativo e, para o valor zero, inativo. A imagem original é então reescrita, tendo os pixels ativos substituídos pela cor vermelha e os *pixels* inativos mantidos de acordo com os da imagem original. A Figura 19 representa a imagem resultante do da aplicação do processo acima descrito.

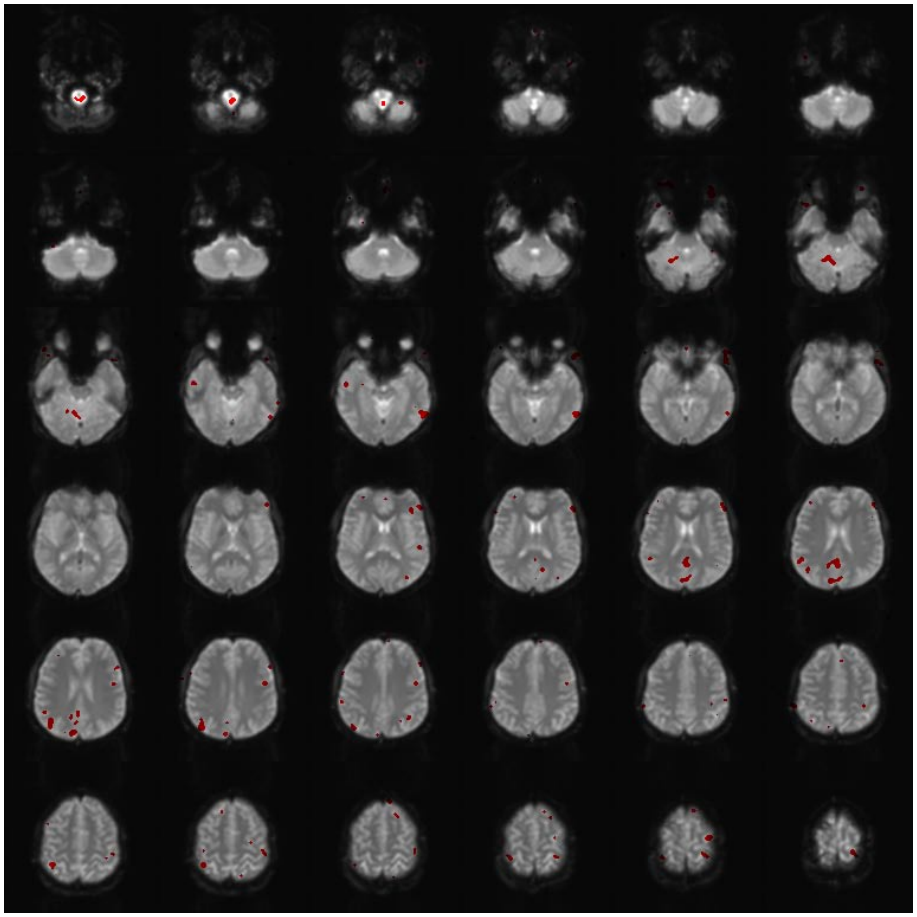


Figura 20.*Imagem de ressonância com marcação dos pixels correspondentes as regiões ativadas resultado da implementação do Student's t-test.*

6.1.2 *2cann* – Implementação de uma Rede *Fuzzy C-Means* com Expansão de Bandas

Para melhorar a base de dados de treinamento da rede *Fuzzy C-Means* foi usado a técnica de expansão de bandas nas 48 imagens de entrada. Foram geradas mais 48 imagens com a operação de dilatação (primeira expansão de banda), e mais 48 com a dilatação das imagens já dilatadas (segunda expansão de banda) totalizando 144 imagens para o treinamento. Com a expansão de bandas é esperado uma melhoria na fase de treinamento da rede FCM, visto que cada expansão leva em consideração informações de vizinhança.

O aplicativo *2cann* possibilita o uso de expansões de bandas para uma imagem ou um conjunto de imagens. No *2cann* é possível a aplicação dos seguintes elementos estruturantes quadrado, cruz, vertical, horizontal, diagonal principal e diagonal secundária conforme a Figura 21. No aplicativo *2cann* os elementos estruturantes possuem dimensão 3 x 3.

Após cada expansão, é disponibilizada a opção de guardar as imagens resultantes em um diretório qualquer.

Nas figuras 22 e 23 há imagens resultantes da dilatação pelo Quadrado 3 x 3 a imagem da Figura 18 através do processo de primeira e segunda expansão de bandas, aplicadas no aplicativo. A aplicação proposta neste trabalho contém, além do módulo de pré-processamento já mostrado, um módulo inteligente, na qual constitui a técnica *Fuzzy C-Means* e uma interface gráfica em GTK. A linguagem escolhida para o desenvolvimento do modelo proposto, foi a linguagem C devido a sua performance, abstração de *syscalls* e possibilidade de criação de processos para o

proveito de *cluster*. Devido à biblioteca *Glade*, gerador de interface para ambiente GNOME, não possuir compilador para C++, esta não pode ser usada.

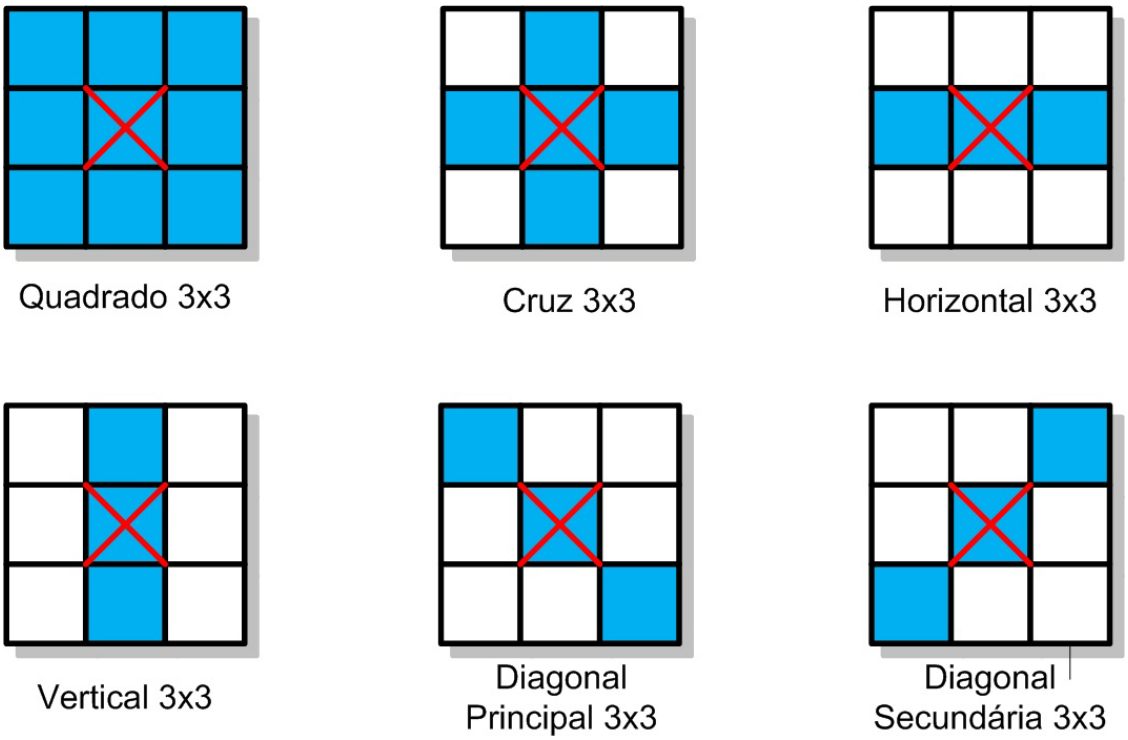


Figura 21.*Representação dos elementos estruturantes implementados no aplicativo 2cann.*

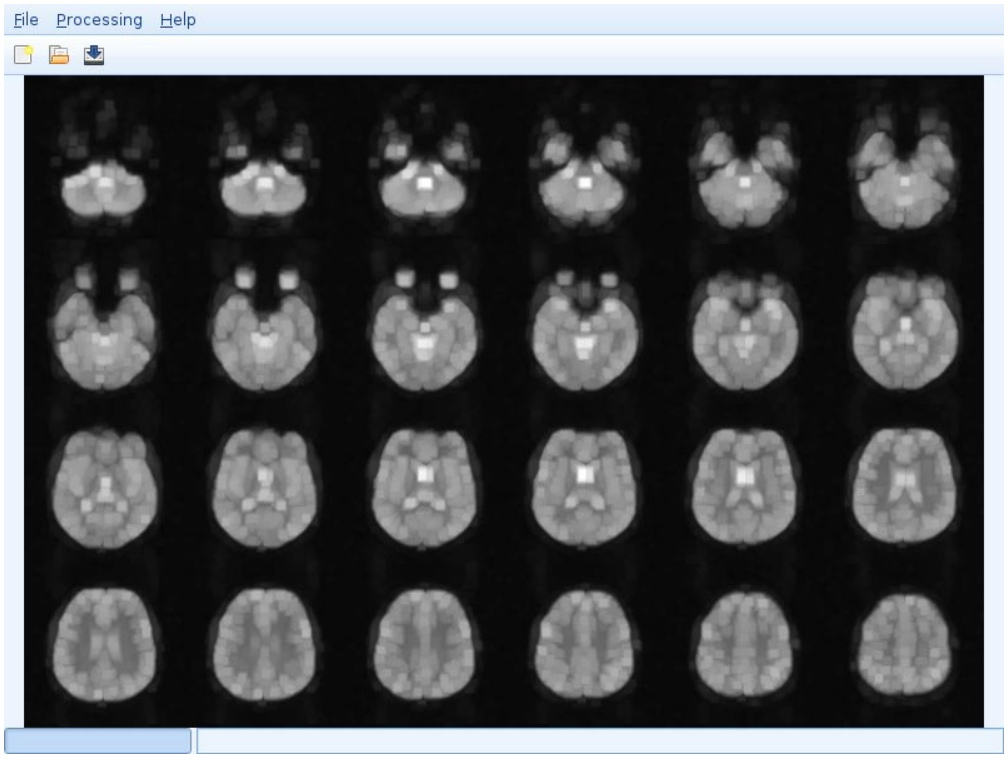


Figura 22.*Imagem resultante da primeira expansão usando a operação de abertura na imagem da Figura 18.*

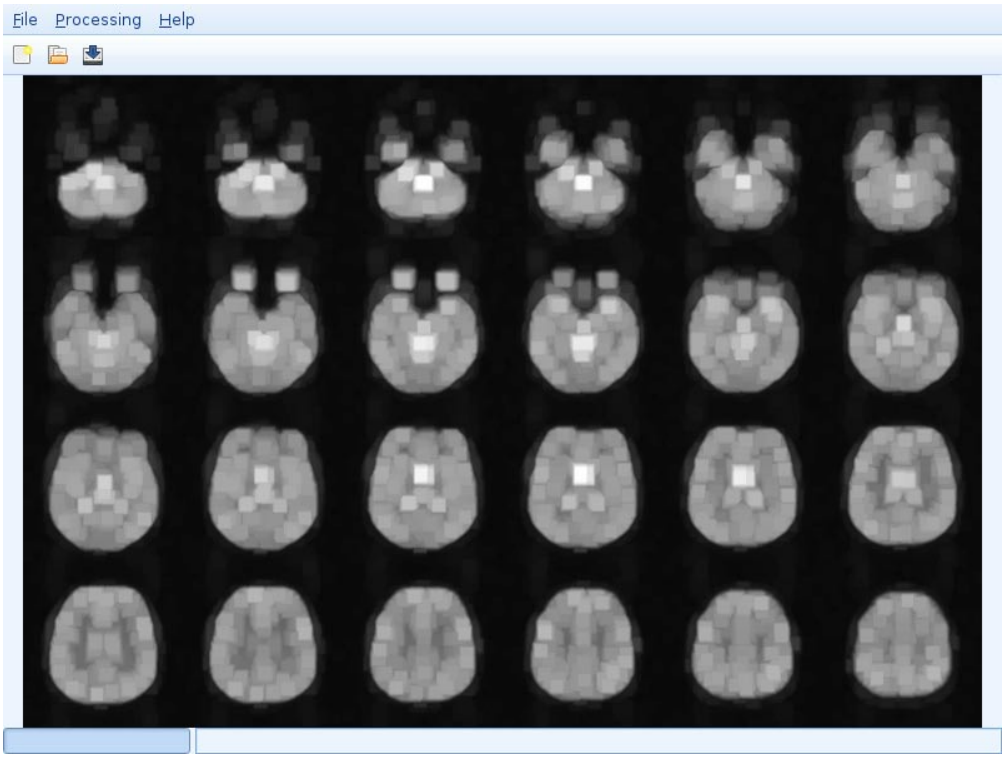


Figura 23.*Imagem resultante da segunda expansão usando a operação de abertura na imagem da Figura 18.*

Entretanto não foi conseguido a interligação dos módulos a tempo da apresentação deste trabalho devido à complexidade de implementação em C de cada módulo.

6.2 Experimentos

Os experimentos do software que implementa a técnica *Student's t-test* foi feito em uma LAN com quatro computadores. As configurações de cada máquina são apresentadas na tabela a seguir.

Nodo	Processador	*Clock* (GHz)	Memória (MB)
1	Intel Xeon Hyper Threading	3.06	1024
2	AMD Duron	1.3	256
3	AMD Athlon XP 2800+	2.0	512
4	Intel Celerom M	2.6	1024

Um CD do *Cluster Knoppix* foi rodado em cada máquina constituindo um *cluster openMosix* como 4 nodos.

Em cada experimento, foi executada a aplicação em uma máquina diferente, tomando o cuidado de anotar o tempo de execução, a carga de cada nodo e o tráfego da rede.

O testes foram executados 1 nodo *openMosix*, 4 nodos e 128 processos, 4 nodos e 256 processos. A hipótese era que o experimento realizado no *cluster* com 4 nodos e 256 processos.

6.2.1 Análise dos Resultados

Com um único nó *openMosix*, obtivemos uma média de 14,339 s na execução do programa de testes. A tabela 1 exibe a execução dos experimentos em 10 simulações, assim como o desvio médio obtido. Com 4 nós e 128 processos, o tempo caiu para 9,608 s e com 256 processos a performance foi melhorada, reduzindo o tempo para 6,675 s. A tabela 2 mostra todas as simulações realizadas para obter a média com 128 processos, enquanto a tabela 3 exibe os resultados para 256 processos. Conforme a expectativa, o aumento do poder computacional trouxe melhoria nos resultados, e mantendo o número de nós e aumentando os processos, a performance também foi melhorada. Isso indica que o balanceamento de carga do *openMosix* foi eficiente, e aproveitou bem a estrutura do *cluster*.

Tabela 1: Resultados obtidos com 1 nodo no openMosix

Simulação	Tempo (s)
1	17,410
2	16,254
3	13,308
4	11,789
5	12,453
6	14,462
7	14,296
8	13,757
9	14,049
10	15,608
Média	**14,339**
Desvio Padrão	**1,709**

Tabela 2: Resultados obtidos com 4 nodos e 128 processos no *openMosix*

Simulação	Tempo (s)
1	10,193
2	9,169
3	10,308
4	9,789
5	8,453
6	10,462
7	10,296
8	9,757
9	8,049
10	9,608
Média	**9,608**
Desvio Padrão	**0,82**

Tabela 3: Resultados obtidos com 4 nodos e 256 processos no *openMosix*

Simulação	Tempo (s)
1	7,490
2	6,098
3	5,525
4	6,906
5	7,570
6	7,731
7	6,913
8	5,999
9	5,987
10	7,564
Média	**6,675**
Desvio Padrão	**0,744**

Capítulo 7

Conclusão

É notável a necessidade de melhorar o entendimento do cérebro humano, e a detecção de atividades em fMRI, é um bom caminho para isso. Esse entendimento é importante para mapeamento das regiões cerebrais controladoras de alguma atividade, possibilitando possíveis seqüelas em uma cirurgia de câncer cerebral, para a implantação de um *chip* para permitir que deficientes visuais enxerguem, ou simplesmente para a estimulação de algumas áreas, por exemplo.

Como a computação necessária para o estudo de regiões de ativação é dispendiosa em memória e processamento. O uso de *Mainframes* é adequado no quesito computação, mas como é sabido, é caro financeiramente. Este trabalho propôs os uso de *clusters* pois oferece alto poder de processamento com baixo custo, sendo acessível popularmente, pois pode usar ferramentas de código aberto e computadores domésticos.

Este trabalho propôs, o desenvolvimento de ferramentas que implementa alguma técnica de análise de regiões ativadas em fMRI.

Nesse capitulo serão discutidos as contribuições desse projeto, assim como as dificuldades encontradas durante sua construção e como ele pode ser estendido e aperfeiçoado com trabalhos futuros.

7.1 Contribuições

As contribuições desse trabalho quanto a implementação foram:

- Módulo de expansão de bandas utilizando as técnicas de abertura e fechamento com vários tipos de elementos estruturantes;
- Desenvolvimento da técnica de aprendizado não supervisionado *Fuzzy C-Means*;
- Interface gráfica em GTK para facilitar o uso;
- Implementação do método de análise estatística *Student's t-test*.
- Implantação de um *cluster openMosix*.

Quanto aos benefícios oferecidos a comunidade foi:

- Possibilidade de uso de *clusters* para a análise de fMRI;
- Uso de expansão de bandas para o melhoramento da fase de pré-processamento.

7.2 Dificuldades Encontradas

No decorrer deste trabalho foram encontradas algumas dificuldades que impossibilitaram em tempo hábil, a realização de alguns os objetivos propostos.

Devido a necessidade do uso de linguagens de programação com suporte a chamadas de sistemas (*syscalls*), e devido ao gerador de interface gráfica para ambiente GNOME não possuir compilador para C++, foi necessário o uso da linguagem C. Por ser uma linguagem orientada a procedimentos, foi bastante custosa a implementação de componentes simples em outras linguagens. Como exemplo, no módulo de expansão de bandas, há a necessidade do uso de matrizes para a representação de imagens. Entretanto houve momentos que foi necessário a aquisição das dimensões da matriz, na qual a linguagem C, por padrão, não oferece, tendo que ser implementados estruturas para tal. Outro problema foi a necessidade do uso de orientação a objeto para a implementação dos algoritmos de aprendizagem de máquina. Desta maneira embora necessária, a linguagem C trouxe dificuldade na realização total da proposta inicial deste trabalho, pois apesar de todos os módulos necessários estarem implementados, estes não foram agregados a tempo de entrega deste trabalho.

7.3 Trabalhos Futuros

Neste trabalho é proposto o desenvolvimento de atividades futuras relativas a objetivos não realizados, ao melhoramento das idéias concretizadas e a implementação e desenvolvimento de novas propostas. Dentre elas:

- Agregação dos módulos desenvolvidos;
- Implementação de outras técnicas de aprendizado de máquina, tais como *Kohonen* SOM;
- Implementação de outras técnicas de pré-processamento;
- Realização de testes em outros tipos de clusters;
- Comparação entre técnicas para a detecção de atividades baseadas em estatísticas assim como em técnicas inteligentes;
- Criação de uma distribuição Linux em *LiveCD* para popularização da ferramenta.

Bibliografia

[1] HAYKIN, Simon. *Redes Neurais Princípios e Prática*. Bookman, 2001.

[2] SLOAN, Joseph D. *High Performance Linux Clusters with OSCAR, Rocks, OpenMosix, and MPI*. O'Reilly Media, 2004.

[3] BANON, G.J.F.; BARRERA, J. *Bases da Morfologia Matemática para Análse de Imagens Binárias*. 2. Ed. 1998.

[4] SANTOS, Wellington Pinheiro dos. *Análise de imagens digitais em patologia utilizando Morfologia Matemática e Lógica Nebulosa*. 2003. Dissertação apresentada à Universidade Federal de Pernambuco para a obtenção do título de Mestre em Engenharia Elétrica, Recife.

[5] Cambridge Dictionaries Online. Disponível em <http://dictionary.cambridge.org/> Acesso em 8 de novembro de 2007.

[6] KULKARNI, Arun D. *Computer Vision and Fuzzy-Neural Systems*. New Jersey. Prentice Hall PTR, 2001.

[7] BEAGLE, R. e Jackson, T. *Neural Computing*: *An Introduction IOP Publishing*. Bristol, UK, 1992.

[8] STALLINGS, William. *Computer Organization and Architecture - Designing for Performance*. Prentice.Hall, 2003.

[9] KOPPER, K. *The Linux Enterprise Cluster*. San Francisco: No Starch Press, 2005.

[10] How Google Works. Disponível em <http://www.baselinemag.com/print_article2/0,1217,a=182560,00.asp> Acesso em 9 de novembro de 2007.

[11] PEREIRA, N. A. *Serviços de pertinência para clusters de alta disponibilidade*. São Paulo. 2004. Dissertação de Mestrado apresentada na USP.

[12] CENAPAD. Guia do Usuário CENAPAD. Disponível em <http://www.cenapad.unicamp.br>. Acesso em: 12 de novembro de 2007.

[13] PITANGA, M. *Construindo Supercomputadores com Linux*. Rio de Jaineiro: Brasport, 2002.

[14] FMRIB Centre, University of Oxford. Disponível em < http://www.fmrib.ox.ac.uk/> Acesso em 16 de novembro de 2007.