UNIVERSIDADE FEDERAL DE MINAS GERAIS
ENGENHARIA DE SISTEMAS

MACHINE LEARNING AUTOMATED MODEL COMPARISON: UM SOFTWARE PARA AUTOMATIZAR A COMPARAÇÃO ENTRE MÉTODOS DE APRENDIZADO

PAULO CIRINO RIBEIRO NETO

Orientador: Antônio de Pádua Braga
Universidade Federal de Minas Gerais

Coorientador: Gustavo Rodrigues Lacerda Silva
Universidade Federal de Minas Gerais

BELO HORIZONTE
NOVEMBRO DE 2017

PAULO CIRINO RIBEIRO NETO

MACHINE LEARNING AUTOMATED MODEL COMPARISON: UM SOFTWARE PARA AUTOMATIZAR A COMPARAÇÃO ENTRE MÉTODOS DE APRENDIZADO

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Sistemas da Universidade Federal de Minas Gerais, como requisito parcial para a obtenção do título de Doutor em Engenharia de Sistemas.

Orientador: Antônio de Pádua Braga
Universidade Federal de Minas Gerais

Coorientador: Gustavo Rodrigues Lacerda Silva
Universidade Federal de Minas Gerais

UNIVERSIDADE FEDERAL DE MINAS GERAIS
ENGENHARIA DE SISTEMAS
BELO HORIZONTE
NOVEMBRO DE 2017

Esta folha deverá ser substituída pela cópia digitalizada da folha de aprovação fornecida pelo Programa de Pós-graduação.

Dedico esse trabalho à meus familiares, amigos, colegas e professores.

“The only excuse for making a useless thing is that one admires it intensely.” (Oscar Wilde, prefácio, O retrato de Dorian Gray)

Resumo

O tema desse trabalho de conclusão de curso é a criação de um software para automatizar testes de modelos de aprendizado de máquina.

Ao fim do projeto, existirá um software, escrito na linguagem de programação R, capaz de comparar modelos de aprendizado de máquina consagrados na literatura com modelos criados pelo usuário. O software fará o tratamento das base de dados padrões, suas chamadas e os testes estatísticos de qualidade e tempo para os problemas de classificação, regressão e clusterização.

Será discutido nesse trabalho o impacto social de software livre e os aspectos técnicos e definições matemáticas do aprendizado supervisiona e não supervisionado.

Palavras-chave: Software Livre. Aprendizado de Máquina. Linguagem R. Teste de Modelos.

Abstract

The theme of this work is the creation of a software capable of automating machine learning models testing.

At the end of the project, there will be a software written in the R programming language, capable of comparing established machine learning model with ones created by the user. The software will make standard database treatments, create function calls for testing, and define statistical tests routines of bolth quality and time for classification, regression and clustering problems.

The discussion of social impact of free software and the technical and mathematical aspects of supervised and unsupervised learning, will also be part of the project.

Keywords: Free Software. Machine Learning. R programming Language. Model Testing.

Lista de Figuras

Lista de Tabelas

Lista de Quadros

Lista de Algoritmos

Lista de Abreviaturas e Siglas

ABNT	Associação Brasileira de Normas Técnicas
UFMG	Universidade Federam de Minas Gerais
GNU	*GNU's Not Unix*
GPL	*General Public License*
MIT	*Massachusetts Institute of Technology*
BSD	*Berkeley Software Distribution*
NSA	*National Security Agency*
SO	Sistema Operacional
CRAN	*Comprehensive R Archive Network*
CPU	*Central Processing Unit*
GPU	*Graphical Processing Unit*
POO	Programação Orientada à Objetos
RNA	Redes Neurais Artificiais
ML	*Machine Learning*
KNN	*K Nearest Neighbours*
NN	*Nearest Neighbour*
MLP	*Multilayer Perceptron*
PCA	*Principal Component Analasys*
SVM	*Support Vector Machines*

Lista de Símbolos

Γ	Letra grega Gama
λ	Comprimento de onda
$\in$	Pertence

Sumário

Capítulo 1

Introdução

1.1 Justificativa

Blá blá blá ....

1.2 Objetivo

Blá blá blá ....

1.3 Organização do trabalho

Normalmente ao final da introdução é apresentada, em um ou dois parágrafos curtos, a organização do restante do trabalho acadêmico. Deve-se dizer o quê será apresentado em cada um dos demais capítulos.

Capítulo 2

Software Livre

2.1 O Que é Software Livre

A definição de software livre apresenta os critérios para determinar se um programa de software é qualificado como software livre. Essa definição pode mudar conforme o momento histórico, mas atualmente é definida pela GNU como *software que os usuários têm a liberdade de executar, copiar, distribuir, estudar, alterar e melhorar* (INC, 2012) .

Nesse contexto, o conceito de livre diz respeito à "liberdade de expressão", não como gratuito. Com essas liberdades, os usuários, tanto individualmente ou coletivamente, controlam o programa e o que ele pode fazer. Quando os usuários não controlam o programa, chamamos ele de programa *não livre* ou *proprietário*.

Em linhas gerais, um software livre deve, obrigatoriamente, obedecer a quatro liberdades (INC, 2012; WILLIAMS; STALLMAN, 2010; LESSIG; STALLMAN, 2002) :

1. A liberdade de executar o programa como desejar, para qualquer propósito;
2. A liberdade de estudar como funciona o programa e mudá-lo para que ele faça a sua computação como você deseja;
3. A liberdade de redistribuir cópias para que você possa ajudar seu vizinho;
4. A liberdade de distribuir cópias de suas versões modificadas para outros.

Naturalmente, essas liberdades estão relacionadas às informações divulgadas entre desenvolvedor e usuário. A liberdade **2**, por exemplo, implica na necessidade de o desenvolvedor divulgar abertamente o código fonte de um software, e permitir que esse seja modificado. Além disso, é obvio que para o usuário terem a liberdade de decidir sobre sua computação, o software livre, por sua vez, não pode utilizar nenhum código *não livre*.

É importante notar que não existem, nessas liberdades, quaisquer limitações relacionadas ao uso comercial do software. Isso significa que empresas podem cobrar pelo uso do

software, em versões modificadas ou não.

Além das liberdades básicas de um software livre, existem abaixo delas, licenças de uso. Cada licença específica, restringe ou garante mais liberdades sobre o software, sem afetar as quatro fundamentais. De forma forma geral, existem quatro famílias de licenças : *Permissiva*, *Fracamente Protetiva*, *Fortemente Protetiva* e *Protetiva de Rede* (WILLIAMS; STALLMAN, 2010).

2.2 História do Software Livre

O surgimento do movimento de software livre está altamente atrelado a academia e ao desenvolvimento dos sistemas operacionais UNIX, GNU e Linux.

O UNIX têm suas origens na *joint venture*, lançada no final da década de 1960 pela *Bell Labs* e MIT para criar um novo sistema operacional chamado *Multics*. Utilizando o conhecimento adquirido nesse projeto, alguns dos programadores desenvolveram paralelamente um sistema operacional para oferecer mais flexibilidade aos usuários, que nomearam UNIX.

Em 1975, Ken Thompson juntamente com Bill Joy e Chuck Haley começaram a distribuir uma versão *open source* do UNIX chamada BSD. No ano seguinte, o lançamento de uma edição revista foi denominada 2BSD.

Em 1984, o programador Richard Stallman fundou o Projeto GNU. A GNU GPL permitia aos usuários modificar o código e distribuir a versão melhorada sob a mesma licença. O sistema operacional GNU não tinha um *kernel*, até que Linus Torvalds desenvolveu o *kernel* do Linux. Em 1992, o *kernel* do Linux foi integrado no sistema operacional GNU.

Nos anos seguintes surgiu a introdução de muitas versões comerciais e aprimoradas do sistema operacional Linux por fornecedores como Red Hat, Mandriva e Novell.

Com a criação de sistemas operacionais que poderiam ser utilizados com total liberdade, surgiu então uma demanda por softwares livres que funcionassem nesses sistemas. Da mesma forma que a comunidade se juntou para aprimorar a base do *kernel* do Linux, eles juntaram e fizeram os mais diversos softwares para atender a demanda.

Nos dias atuais existem diversas comunidades que fazem os mais variados tipos de softwares utilizando os mesmos princípios criados por Richard Stallman. Os softwares livres difundiram na sociedade, e hoje são peças fundamentais para infraestrutura computacional, periféricos, celulares e virtualmente qualquer outro dispositivo computacional.

2.3 A Importância Social do Software Livre

Atualmente, os softwares livres são fundamentais principalmente em três áreas sociais: à proteção da liberdade individual, ao avanço da computação e à acessibilidade da educação e conhecimento.

2.3.1 Software Livre à Favor da Proteção da Liberdade Individual

Uma celebre frase da comunidade de software livre diz, 'Os softwares não livres, onde o usuários não controlam o programa, o programa controla os usuários' (WILLIAMS; STALLMAN, 2010).

Essa frase, resume uma preocupação crescente com o software proprietário, a de que sempre há alguma entidade, que controla o programa e através dele, exerce poder sobre seus usuários.

Esse poder é enxergado por alguns como uma afronte ao direito de privacidade do indivíduo. Hoje, serviços de busca e redes sociais, utilizam dos dados de navegação para gerar propagandas sob-medidas. Muitas pessoas, consideram a forma que essas empresas manipulam as informações como uma forma de censura e venda de informação confidencial.

Em alguns casos, empresas que desenvolvem softwares proprietários foram ligadas a escândalos onde propositalmente construíram *backdoors* em seus produtos que dão acesso a informações sem as devidas permissões dos usuários. Um exemplo é o caso do *Kindle*, que possui uma *backdoor* que permite apagar livros (GNU Operating System, 2017) .

Um outro senário onde é importante que o software seja livre, é para proteger os usuários de acesso externo indesejado. Um exemplo disso é o caso do ex analista da NSA (TATE et al., 2013), Edward Joseph Snowden, que em junho de 2013, revelou como a agência americana utilizava de falhas de seguranças, propositais ou não, para espiar na população mundial.

O principio que o software livre protege a liberdade Individual, contra empresas mau intencionadas ou governos abusivos, é que quando o código de um programa é aberto, a comunidade pode ver oque ele faz e testar todas as falhas que o mesmo possa ter.

2.3.2 Software Livre à Favor do Avanço da Computação

Após a construção dos sistemas operacionais livres, surgiram varias distribuições e variações, cada qual para atender um nicho. Esses avanços, tornaram possível que hoje, os sistemas operacionais baseados no Linux e UNIX dominassem o setor de infra estrutura computacional. Possibilitando que empresas e órgãos governamentais customizassem

esses softwares para criar soluções específicas para suas necessidades, que por sua vez não são necessariamente livres.

Essa abordagem se tornou tão prática que, dados da *W3Cook* e *TOP500*, mostram que esses sistemas operacionais são utilizados em 98.3% de todos os servidores públicos de interne e 99.88% dos supercomputadores do mundo.

A ultima grande plataforma que popularizou a utilização do Linux foi o sistema operacional Android. Construído inicialmente para ser um software de telefones celulares, esse sistema operacional criado com base no *kernel* do Linux, se espalhou pelos mais diversos aparelhos, como televisões, *tablets* e até mesmo geladeiras. Esse software se popularizou tanto que, segundo o CEO da Google, Sundar Pichai, é utilizado em mais de 2 bilhões de dispositivos ativos.

Além dos avanços em sistemas operacionais, o movimento de software livre alavancou o desenvolvimento comunitário de softwares de computação livres. Um exemplo disso é a *Apache Software Foundation*, que é uma corporação americana sem fins lucrativos formada por uma comunidade descentralizada de desenvolvedores de código aberto.

Os projetos Apache são feitos em desenvolvimento colaborativo, baseado em consenso e uma licença de software aberta e pragmática. Cada projeto é gerenciado por uma equipe de especialistas técnicos auto-selecionados que são contribuidores ativos para o projeto.

O projeto inicial da Apache foi o *HTTP Server*, que era um sistema para processar protocolos web básicos na internet. Contudo, hoje são 315 projetos nas mais diversas áreas da computação, e incluem softwares de *Big Data* como o *Spark* e *Hadoop*, gerenciamento de projetos como o *Maven* e até mesmo software de escritório como o *Open Office*.

2.3.3 Software Livre à Favor da Acessibilidade da Educação e Conhecimento

As escolas e universidades, influenciam o futuro da sociedade através do que ensinam. Ensinar um programa proprietário é implantar a dependência de um artifício que não é de sua propriedade. Isso diminui a capacidade do futuro profissional em exercer os conhecimentos que lhe foram ensinados.

A utilização de software livre como ferramente de ensino, empodera os estudantes a utilizarem os conhecimentos técnicos na vida após a universidade. Utilizando software livre, um profissional é capaz de fazer uso de programas como ferramenta básica de trabalho gratuitamente, ou ainda utilizar soluções livres como base para criar um produto próprio.

Além de ser útil para o estudante, os softwares livres são importantes para o avanço

da pesquisa acadêmica na universidade. A comunidade de Software livre têm em suas liberdades básicas, a liberdade de estudar como um programa funciona e permitir que os usuários melhorem e redistribuam esse programa. Este espírito de comunidade permite que pesquisadores em locais diferentes do mundo, sem quaisquer dificuldades, compartilharem suas pesquisas e conhecimentos.

Além de promover o compartilhamento da informação, os softwares livres também promovem uma acessibilidade universal dos avanços técnico-científicos, uma vez que permitem pesquisadores e empresas de ponta à compartilhar seus resultados e códigos com o resto do mundo. Isso permite que mesmo pesquisadores com limitantes de recursos, façam aplicações ou pesquisa com esses recursos.

Um exemplo disso, é o caso do software *Tensor Flow* (ABADI et al., 2016), feito pela Google. Esse é um programa extremamente complexo que funciona como motor de operações numéricas, utilizando a abstração de computação em grafos de forma escalável para CPU's e GPU's. Além de compartilhar o código do *Tensor Flow* com a comunidade, a empresa também disponibilizou inúmeros modelos de RNA, como a *LeNet* (SZEGEDY et al., 2015) que é um modelo treinado para identificar objetos em imagens. Utilizando esse avanço, pesquisadores do mundo inteiro foram capazes de utilizar esses modelos em suas próprias pesquisas.

Capítulo 3

Linguagem de Programação R

R é uma linguagem e ambiente para computação estatística e gráfica, que foi criada na década de 90 por Ross Ihaka e Robert Gentleman, enquanto ambos trabalham na Universidade de Auckland. Esse é um projeto GNU que é semelhante a linguagem e ao ambiente *S* desenvolvida na *Bell Laboratories* por John Chambers.

O *R* pode ser considerado como uma implementação diferente da linguagem de programação *S*, ao ponto que código escrito em *S* pode ser executado de forma inalterada pelo interpretador de *R*. Contudo, a principal diferenças entre os dois ambientes, é que o *R* é um projeto de software livre sob licença GPL GNU e o *S* possui licença proprietária.

Atualmente esse ambiente é utilizando principalmente nas áreas de estatística e análise de dados. O sucesso da linguagem nessas tarefas pode ser explicado pela grande variedade de algorítimos em modelagem linear e não-linear, testes estatísticos clássicos, análise de séries temporais e técnicas gráficas. Nos dias atuais, a linguagem é utilizada na indústria e academia,de forma que já é a sexta linguagem de programação mais popular (CASS, 2017).

Apesar de ser uma linguagem livre e aberta, o desenvolvimento e manutenção do ambiente é controlada pela *R Foundation* e pelo grupo de 20 curadores chamados de *R core team*. Contudo, qualquer pessoa pode contribuir com o avanço da linguagem por meio de pacotes, tradicionalmente disponibilizados no repositório oficial CRAN e divulgados na revista chamada *The R Journal*.

3.1 Organização da Linguagem de Programação *R*

O *R* é uma linguagem de programação que não foi planejada para ser tão versátil. Inicialmente, foi criado um interpretador que executava comandos da linguagem *S*, onde o ambiente era composto por funções de matemáticas, estatísticas, gráficas e de leitura de

dados.

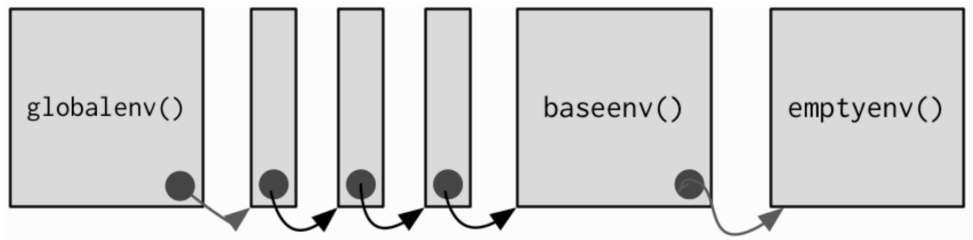
Como a linguagem foi criada por estatísticos para estatísticos, não foram definidos padrões de programação. Dessa forma ela é considerada uma linguagem de quarta geração, ou seja, uma linguagem de domínio.

Possui suporte para os principais paradigmas de programação, incluindo os modelos declarativo, imperativo, orientado à objeto, procedural, funcional e outros. É um linguagem naturalmente lenta, por conta de ser interpretada e concorrente. Entretanto, fornece interface para conectar com linguagens de alta performance, *C*, *C++* e *Fortran*, e programação paralela pela plataforma *OpenMP* .

Segundo o projeto *OpenHub*, o código do interpretador *R* possui mais de 750 mil linhas, das quais 39% são escritas em *C*, 27.1% em *Fortran*, 19.7% em *R*, 8.1% em *Autoconf* e 2% em *shell*.

Esse interpretador foi definido para trabalhar com o conceito de *ambientes*, que são grupos de funções e variáveis organizadas por precedência de contexto. O formato dos contextos podem ser observados na figura 1 .

Figura 1 – *Ambientes* da linguagem de programação R



Fonte: Wickham (2015)

Em linhas gerais, o ambiente onde o usuário trabalha é chamado de *globalenv*, as funções da linguagem *R* estão definidas no ambiente de *baseenv* e os gerenciadores de memória, o coletor de lixo e os demais controladores do próprio interpretador estão ambiente *emptyenv*.

A linguagem *R*, interpreta as funções de forma a percorrer os ambientes hierarquicamente. Quando o usuário chama o interpretador, ele inicialmente procura as funções no *globalenv*, quando não encontra, ele busca recursivamente nos ambientes *pai*.

Todos os ambientes entre o *globalenv* e o *baseenv*, são chamados de *ambientes de pacote*. Esses pacotes foram a forma encontrada pelos desenvolvedores da linguagem para permitir que a comunidade contribuísse com o avanço da linguagem, sem afetar as funções base.

3.2 A comunidade R e seus Pacotes

Capítulo 4

Aprendizado de Máquina

4.1 Aprendizado Supervisionado

4.1.1 Decisor Bayesiano

4.1.2 Árvores de Decisão

4.1.3 Vizinhos Mais Próximos

4.1.4 Redes Neurais Artificiais

4.1.5 Maquinas de Vetores de Suporte

4.2 Aprendizado Não Supervisionado

4.2.1 K Médias

4.2.2 Cluster Aglomerativo

4.3 Métricas de Aprendizado

4.3.1 Métricas Supervisionadas

4.3.2 Métricas Não Supervisionadas

Capítulo 5

Estatísticas de Teste

Capítulo 6

O Pacote

Referências

ABADI, M. et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. **CoRR**, abs/1603.04467, 2016. Disponível em: <http://arxiv.org/abs/1603.04467>. Citado na página 6.

CASS, S. The 2017 Top Programming Languages. **IEEE Spectrum**, 2017. Disponível em: <https://spectrum.ieee.org/computing/software/the-2017-top-programming-languages>. Citado na página 7.

GNU Operating System. **Proprietary Software Is Often Malware**. 2017. Disponível em: <https://www.gnu.org/proprietary/proprietary.html>. Citado na página 4.

INC, F. S. F. **What is free software?** 2012. Disponível em: <http://www.gnu.org/philosophy/free-sw.html>. Citado na página 2.

LESSIG, L.; STALLMAN, R. **Free Software, Free Society: Selected Essays of Richard M. Stallman**. [s.n.], 2002. Disponível em: <https://www.gnu.org/philosophy/fsfs/rms-essays.pdf>. Citado na página 2.

SZEGEDY, C. et al. Going deeper with convolutions. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2015. p. 1–9. Citado na página 6.

TATE, J. et al. **Edward Snowden says motive behind leaks was to expose 'surveillance state'**. 2013. Disponível em: <https://www.washingtonpost.com/politics/edward-snowden-says-motive-behind-leaks-was-to-expose-surveillance-state/2013/06/09/aa3f0804-d13b-11e2-a73e-826d299ff459{_}story.html?tid=pm{_}politics{_}pop{&}utm{_}ter>. Citado na página 4.

WICKHAM, H. **Advanced R**. CRC Press, 2015. Disponível em: <https://englianhu.files.wordpress.com/2016/05/advanced-r.pdf>. Citado na página 8.

WILLIAMS, S.; STALLMAN, R. M. Free as in Freedom (2.0): Richard Stallman and the Free Software Revolution Second edition revisions. 2010. Disponível em: <https://sagitter.fedorapeople.org/faif-2.0.pdf>. Citado 3 vezes nas páginas 2, 3 e 4.