UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

MESTRADO ACADÊMICO EM CIÊNCIAS DA COMPUTAÇÃO

WESLLEY LIOBA CALDAS

PROPOSTA DE DOIS MÉTODOS SEMI-SUPERVISIONADOS BASEADOS NA MÁQUINA DE APRENDIZAGEM MÍNIMA UTILIZANDO CO-TRAINING

FORTALEZA

2017

WESLLEY LIOBA CALDAS

PROPOSTA DE DOIS MÉTODOS SEMI-SUPERVISIONADOS BASEADOS NA MÁQUINA DE APRENDIZAGEM MÍNIMA UTILIZANDO CO-TRAINING

Dissertação apresentada ao Curso de Mestrado Acadêmico em Ciências da Computação do Programa de Pós-Graduação em Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Ciência da Computação

Orientador: Prof. Dr. João Fernando Lima Alcântara

Co-Orientador: Prof. Dr. João Paulo Pordeus Gomes

FORTALEZA

2017

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

C15p Caldas, Weslley Lioba.
Proposta de dois métodos semi-supervisionados baseados na Máquina de Aprendizagem Mínima utilizando Co-Training / Weslley Lioba Caldas. – 2017.
59 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação, Fortaleza, 2017.
Orientação: Prof. Dr. João Fernando Lima Alcântara.
Coorientação: Prof. Dr. João Paulo Pordeus Gomes.

1. Classificação. 2. Aprendizado semi-supervisionado. 3. Maquina de aprendizagem mínima. 4. Co-Training. I. Título.

CDD 005

WESLLEY LIOBA CALDAS

PROPOSTA DE DOIS MÉTODOS SEMI-SUPERVISIONADOS BASEADOS NA MÁQUINA DE APRENDIZAGEM MÍNIMA UTILIZANDO CO-TRAINING

Dissertação apresentada ao Curso de Mestrado Acadêmico em Ciências da Computação do Programa de Pós-Graduação em Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Ciência da Computação

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. João Fernando Lima Alcântara (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. João Paulo Pordeus Gomes (Co-Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. João Paulo do Vale Madeiro
Universidade da Integração Internacional da Lusofonia Afro-Brasileira (UNILAB)

Prof. Dr. Ajalmar Rego da Rocha Neto
Instituto Federal do Ceará (IFCE)

A minha família, por acreditar em meu potencial. Mãe, a senhora me incentivou a cada minuto pela busca do conhecimento. Vó. a senhora me viu crescer, me educou e foi uma segunda mãe. Vô, o senhor foi mais que um pai, foi meu maior referencial de caráter e amigo. Michelle, a mulher da minha vida, obrigado por me tornar um homem melhor e estar sempre ao meu lado.

AGRADECIMENTOS

Ao Prof. Dr.João Paulo Pordeus Gomes por me orientar nesta tese, ser meu amigo, professor e mentor, fornecendo todas as ferramentas necessária para que eu pudesse crescer academicamente.

Ao Prof. Dr.Joao Fernando Lima Alcântara por fazer parte da minha orientação nesta tese.

Ao Prof. Dr. Carlos Eduardo Fisch de Brito, pelas difíceis porém proveitosas horas na disciplina de processos estocásticos.

Aos membros da banca Prof. Dr. João Paulo Pordeus Gomes, Prof. Dr. João Fernando Lima Alcântara, Prof. Dr. João Paulo do Vale Madeiro e Prof. Dr. Ajalmar Rego da Rocha Neto, por me prestigiarem com sua presença.

Aos meus colegas do LOGIA, Diego Parente, Marcelo Veras, Alisson Alencar e Jônatas Aquino, por me auxiliarem em minhas dúvidas, e permitirem meu crescimento no programa de mestrado.

A todos os meus professores tanto da graduação quando da pós-graduação, que me ajudaram a estar aqui hoje, me dando não somente conhecimento, mas também importantes lições de vida.

Ao Google, por me ajudar em horas que ninguém mais poderia.

Aos meus irmãos Odmir Fortes e Asley Caldas, por estarem comigo em todos os momentos que precisei.

A minha família por me proporcionar tudo o que precisei, em todos os aspectos, para estar aqui hoje. Em especial ao meu avô Raimundo Lioba, por ser avô, pai e amigo, e por me ensinar o valor de se ter caráter.

A minha parceira de uma vida toda Michelle Cacais, que se tornou minha nova família, alguém com quem posso contar sempre, me aguentando todos esses anos que passaram e tantos outros que virão, por me ajudar em todos os momentos e por sempre acreditar em mim, mesmo quando eu tive dúvidas.

A Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), por me apoiar financeiramente, o que permitiu que eu me concentrasse em meus estudos.

"É muito melhor lançar-se em busca de conquistas grandiosas, mesmo expondo-se ao fracasso, do que alinhar-se com os pobres de espírito, que nem gozam muito nem sofrem muito, porque vivem numa penumbra cinzenta, onde não conhecem nem vitória, nem derrota."

(Theodore Roosevelt)

RESUMO

O aprendizado semi-supervisionado é um importante ramo de aprendizado de máquina, que combina dados rotulados com dados não rotulados, tendo ganho bastante atenção da comunidade acadêmica nos últimos anos. Isso se deve principalmente a grande quantidade de dados disponíveis e o trabalho necessário para rotular estes dados, tornando o aprendizado semi-supervisionado uma metodologia atrativa por necessitar de uma quantidade relativamente reduzida de dados rotulados. Acerca das diversas abordagens de aprendizado semi-supervisionado, Co-Training tem se tornado popular devido a sua formulação simples e resultados promissores em diferentes áreas. Neste trabalho propõem-se Co-MLM, um método semi-supervisionado que utiliza o recente método supervisionado Máquina de Aprendizagem Mínima, do inglês *Minimal Learning Machine* (MLM) em conjunto com a metodologia Co-Training. Além disso, propõem-se também uma versão rápida deste mesmo método, nomeada de Fast Co-MLM, usando como classificador base NN-MLM, uma variante do MLM. Ambos os métodos foram comparados utilizando conjuntos de dados proveniente dos repositórios UCI, UCF e DataGov, demonstrando capacidade de aprender sobre dados não rotulados, além de resultados promissores quando comparados com outros algoritmos baseados em Co-Training.

Palavras-chave: Classificação, Aprendizado Semi-Supervisionado, Máquina de Aprendizagem Mínima

ABSTRACT

Semi-supervised learning is an important fild of machine learning, combining the use of labeled data with unlabelled data, and has gained attention of academic community in the last years. This is mainly due to the large amount of data available and the work required to label these data, making semi-supervised learning an attractive methodology because it requires a reduced amount of labeled data. Regarding the various approaches of semi-supervised learning, Co-Training has become popular because of its simple formulation and promising results in different areas. In this work, we propose Co-MLM, a semi-supervised method that uses the Minimal Learning Machine (MLM), a recent proposed supervised method, in conjunction with the Co-Training methodology. In addition, we also propose a quick version of this same method, named Fast Co-MLM, using as base classifier the NN-MLM method, an MLM variant. Both methods were compared using data sets from the UCI, UCF and DataGov repositories, demonstrating ability to learn about unlabeled data, and promising results when compared with other Co-training based algorithms.

Keywords: Classification. Semi-Supervised Learning. Minimal Learning Machine

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE CÓDIGOS-FONTE

LISTA DE ABREVIATURAS E SIGLAS

AM	Aprendizado de Máquina
AMV	Aprendizado Multi-Visão
ANS	Aprendizado Não Supervisionado
AS	Aprendizado Supervisionado
ASS	Aprendizado Semi-Supervisionado
Co-MLM	*Co-Training Minimal Learning Machine*
DCL	*Democratic Co-learning*
EM	*Expectation Maximization*
Fast Co-MLM	*Fast Co-Training Minimal Learning Machine*
IA	Inteligência Artificial
KNN	*K-Nearest Neighbor*
LM	Levenberg-Marquardt
MLM	*Minimal Learning Machine*
MQO	Mínimos Quadrados Ordinário
MQR	Mínimos Quadrados Recursivo
NN-MLM	*Nearest Neighbor Minimal Learning Machine*
PAC	Probabilidade Aproximadamente Correta
PEC	Probabilidade Estimada da Classe
Rasco	*Random Subspace Co-Training*
Rel-Rasco	*Relative Random Subspace Co-Training*
SSL-MLM	*Semi-Supervised Learning-Minimal Learning Machine*

LISTA DE SÍMBOLOS

$\mathscr{Y}$	Espaço de saída
$\mathscr{X}$	Espaço de entrada
X	Conjunto de dados no espaço de entrada
Y	Conjunto de dados no espaço de saída
U	Conjunto de dados não rotulados
L	Conjunto de dados rotulados
Z	Conjunto de dados contendo $L \cup U$
Q	Número de iterações na fase *Co-Training*
q	Iteração dado um determinado momento
K	Número de pontos de referência
M	Conjunto de pontos de referência no espaço de entrada
T	Conjunto de pontos de referência no espaço de saída
D_x	Matriz de distancias dos elementos do espaço de entrada
Δ_y	Matriz de distancias dos elementos do espaço de saída
B	Matriz de coeficientes de peso de uma regressão
x	Instância de exemplo de entrada
y	Resposta correspondente a instância de entrada
$\hat{y}$	Saída aproximada correspondente a instância de entrada
Y	Vetor contendo as saídas exatas referente a classificação de uma instância x
$\hat{Y}$	Vetor contendo as saídas aproximadas referente a classificação de uma instância x
C_s	Índice do elemento do vetor $\hat{Y}$ que representa a provável classe de x.
h_j	Classificador ou hipótese j

SUMÁRIO

1 INTRODUÇÃO

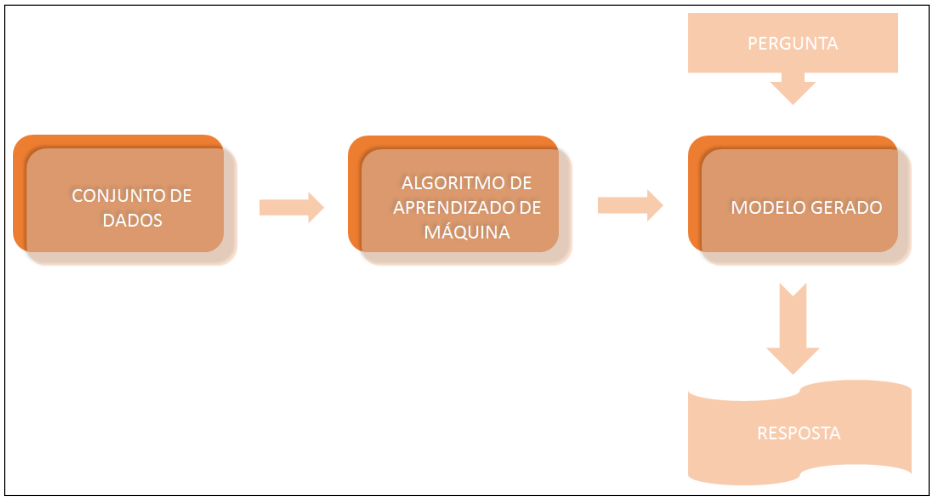
Com o passar dos anos, a presença de programas de Inteligência Artificial (IA) tem se difundido cada vez mais na sociedade, seja na medicina, robótica, telefonia ou a própria internet. Dentre as diversas metodologias existentes de IA, o Aprendizado de Máquina (AM) tem se tornado uma das mais relevantes, sendo objeto de estudo dessa dissertação.

Conforme (MITCHELL, 1997) AM, foca em como construir programas de computador que automaticamente adquiram conhecimento. Neste caso, o conhecimento provém de dados fornecidos ao computador, que farão um programa aprender qual decisão tomar em determinado problema. Existem diversas definições quanto ao que é aprender. Para (SIMON, 1983), a aprendizagem denota mudanças em um sistema de forma a adapta-lo a fazer tarefas iguais ou similares de forma mais eficiente. (MITCHELL, 1997) ainda propõe uma definição mais formal de aprendizagem no contexto de AM como:

> Um programa de computador aprende a partir de uma experiência E em relação a uma classe de tarefas T, com medida de desempenho P, se seu desempenho em T, medido por P, melhora com E.

Ao analisar as características presentes em uma base de dados, algorítimos de AM buscam por padrões existentes, e nunca em nenhuma outra época houve tantos dados disponíveis para serem analisados. Isso se deu graças a popularização da internet que proporcionou o surgimento de uma grande massa de informações sobre praticamente qualquer coisa.

Figura 1 – Construção de um modelo de aprendizado de máquina



Fonte: O autor

A maioria dos autores divide o Aprendizado de Máquina em duas categorias sendo elas o Aprendizado Supervisionado (AS) e o Aprendizado Não Supervisionado (ANS). Para (FRIEDMAN *et al.*, 2001) o objetivo do Aprendizado de Máquina é predizer um valor de saída mensurável tomando como base os valores de entrada, em outras palavras, existe um "professor"que avalia as respostas dadas pelo programa ao conjunto de dados de entrada, que logo em seguida ajustará a resposta do programa de forma que a mesma parecerá mais com a resposta do "professor". A Figura 1 apresenta as etapas básicas na construção de um modelo de aprendizado de máquina supervisionado: ao obter um conjunto de dados com características de entrada e saída, selecionas-se um algoritmo de seu interesse para então criar um modelo baseado nos dados disponibilizados. Então, dado um novo exemplo com características similares as descritas pelo problema, o modelo fornecerá uma saída aproximada da saída real deste exemplo. É importante ressaltar que existem ainda alguns problemas de AS em que cada exemplo pertence a uma categoria, ou seja contém um rótulo que o identifica a um determinado grupo. Neste tipo de problema, um modelo é construído a partir de uma base de dados contendo os atributos e rótulos de cada exemplo, que após adquirir informações poderá ser utilizado para descobrir os rótulos de novos exemplos, a esse tipo de modelo nos referenciaremos como classificador.

Segundo o mesmo autor, o Aprendizado Não Supervisionado, é uma metodologia onde a resposta não é uma saída mensurável, mas sim a descrição de associações do conjunto de informações fornecidas na entrada. Diferente do visto em AS, um modelo de ANS não contém nenhuma informação da saída de um exemplo, porém é possível agrupa-los por semelhança, diferença ou qualquer outro critério pré-estabelecido. Nesta forma de aprendizagem o programa deve encontrar sozinho padrões, regularidades ou agrupamentos, sem a necessidade de um "professor".

Existe ainda um outro paradigma de AM chamada de Aprendizado Semi-Supervisionado (ASS, (ZHU, 2005)) que combina dados rotulados com dados não rotulados. Este ultimo paradigma, um intermédio do Aprendizado Não Supervisionado e do Aprendizado Supervisionado, representa o foco desta dissertação.

Neste trabalho adotaremos a terminologia mais presente nos livros sobre AM, em que o conjunto de dados de entrada (variáveis independentes) é denominado de matriz de atributos, enquanto que a saída (variável dependente) será chamada de reposta, ou no caso de classificação, será chamada de: categoria, classe ou rótulo. Além disso, denominaremos de exemplo uma instância qualquer presente no banco de dados.

1.1 Aprendizado Semi-Supervisionado (ASS)

Atualmente, a maioria dos dispositivos, como *smartphones*, câmeras digitais, ou *smartwatches*, provém formas simples de se obter os mais variados tipos de dados, como fotos, áudios, mensagens, informações de contato, pesquisas feitas e etc. Infelizmente, utilizar esses dados nem sempre é uma tarefa fácil, uma vez que categorizá-los, requer tempo, dinheiro e esforço humano. Consequentemente, o Aprendizado Semi-Supervisionado, que explora a capacidade de combinar dados rotulados com dados não rotulados, tem obtido atenção da comunidade por ser uma alternativa atrativa para reduzir custos, uma vez que dados não rotulados são obtidos a custos inferiores, e a quantidade de dados rotulados necessária é reduzida.

Diversas abordagens para ASS tem sido propostas com o passar dos anos, como modelos genéricos (RABINER, 1989), modelos baseados em grafos (BLUM; CHAWLA, 2001), *Semi-Supervised Support Vector Machines* (S3VM (BENNETT *et al.*, 1999)), e Co-Training (BLUM; MITCHELL, 1998). Acerca de todas estas abordagens, Co-Training é uma das mais atrativas, sendo considerada uma parte fundamental do Aprendizado Multi-Visão (AMV) (XU *et al.*, 2013), obtendo resultados satisfatórios em diferentes áreas, como por exemplo diagnóstico por computador (LI; ZHOU, 2007), processamento de linguagem natural (PIERCE; CARDIE, 2001), recuperação de imagens baseada em conteúdo (ZHA *et al.*, 2013) e classificação de imagens (YU *et al.*, 2012).

A ideia básica do Co-Training é construir dois classificadores utilizando visões independentes (i.e os conjuntos de atributos são independentes dados os rótulos), selecionar alguns dos exemplos de dados não rotulados, para então atribuir rótulos aos mesmos e aumentar o número de dados rotulados. Na maioria dos casos, os conjuntos de dados não contém dois conjuntos de atributos originalmente formados, então uma forma simples de criá-los é repartir o conjunto original de atributos em dois subconjuntos de tamanhos iguais aleatoriamente. É importante notar que para (BREFELD, 2015) tal medida, só pode ser feito sob algumas condições e que estes subconjuntos não necessariamente serão independentes. Por simplicidade, todos os experimentos adotados neste trabalho utilizaram conjuntos de dados que possuem duas visões distintas, ou que seu conjunto original de atributos possa ser dividido em dois, a fim de formar as duas visões.

Escolher o algoritmo base é uma tarefa extremamente importante para o Co-Training. Isto encorajou muitas variações de Co-Training baseadas em algoritmos que obtiveram resultados promissores em Aprendizado Supervisionado (FEGER; KOPRINSKA, 2006; WAN, 2009; LI *et*

al., 2013) . Máquina de Aprendizagem Mínima, do inglês *Minimal Learning Machine* (MLM, (A.H. *et al.*, 2013)) é um recente método supervisionado que obteve resultados promissores em relação a outros algoritmos do estado-da-arte para Aprendizado Supervisionado. Por sua simplicidade e facilidade de implementação, MLM tem ganhado atenção da comunidade acadêmica proporcionando diversos outros trabalhos (MESQUITA *et al.*, 2015a; MESQUITA *et al.*, 2015b; ALENCAR *et al.*, 2015) .

Neste trabalho propõe-se dois métodos de ASS utilizando o *framework* Co-Training. O primeiro método, batizado de Co-MLM, usa como classificador base o método MLM. Somado a isso, propõe-se também Fast Co-MLM, uma variação de Co-MLM contendo duas modificações importantes em relação ao primeiro: primeiramente, no uso da formulação recursiva do algoritmo mínimos quadrados; a segunda, em usar como classificador base o *Nearest Neighbor Minimal Learning Machine* (NN-MLM, (MESQUITA *et al.*, 2017)), uma variação rápida do MLM. Uma vez que MLM obteve bons resultados em AS, espera-se, como hipótese, que a mesma situação ocorra para ASS, e em adicional será oferecida uma biblioteca funcional com os métodos propostos.

1.2 Objetivos Gerais

O principal objetivo deste trabalho é propor Co-MLM e Fast Co-MLM, dois métodos de ASS, e então avaliar seu desempenho em diversos problemas reais, assim comprovando sua capacidade de combinar informações de dados rotulados e não rotulados. Fornecendo assim, duas novas ferramentas a disposição da comunidade.

1.3 Objetivos Específicos

Os objetivos específicos deste trabalho são apresentados a seguir.

1. Propor duas extensões do método MLM baseadas em Co-Training para o ASS, sendo elas Co-MLM e Fast Co-MLM;
2. Realizar um estudo comparativo entre Co-MLM e Fast Co-MLM, com o intuito de verificar a ganho de velocidade de Fast Co-MLM em relação a Co-MLM.
3. Realizar um estudo comparativo de desempenho entre Co-MLM e Fast Co-MLM, e diversas outras técnicas padrões de ASS, sendo elas TriTraining, Co-Forest e o Co-Training usando KNN como classificador base.

1.4 Organização

Este trabalho está organizado da seguinte forma. O capitulo 2 apresenta as definições formais do Co-Training e sua principais variações. Além disso, será definido formalmente o método MLM e sua variação NN-MLM. Capitulo 3 apresenta os métodos propostos Co-MLM e Fast Co-MLM. O capitulo 4 demonstra os experimentos e resultados a fim de validar os métodos propostos. O capitulo 5 apresenta as conclusões da presente dissertação, perspectivas e trabalhos futuros . Por fim, o apêndice A, apresenta a biblioteca disponibilizada, fornecendo também a documentação necessária para a reprodução dos algoritmos.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo apresentaremos os principais embasamentos teóricos para esta dissertação. Inicialmente na seção 2.1 cobriremos o algoritmo Co-Training, suas principais variações, formulação teórica e por fim, apresentaremos um *framework* genérico para aplicações voltadas a Co-Training. Na seção 2.2 cobriremos o método *Minimal Learning Machine*, sua extensão para classificação e sua variação NN-MLM, bem como a complexidade de ambos os métodos.

2.1 Co-training

O algoritmo Co-Training padrão opera em duas visões distintas, isto é, duas hipóteses independentes dados os rótulos de uma base de dados. Conforme Blum e Mitchell (1998) dois classificadores são construídos usando o conjunto de dados rotulado existente, em que cada classificador usa somente seu respectivo conjunto de atributos. Neste caso pode haver bases de dados diferentes para um mesmo problema, como visto na figura 2, em que existem informações textuais e gráficas sobre um mesmo individuo, ou simplesmente uma divisão aleatória dos atributos para formar as visões (subconjunto de atributos) a partir do conjunto dos atributos originais.

Na metodologia proposta pelo autor, cada classificador prediz os rótulos de alguns exemplos escolhidos aleatoriamente dos dados não rotulados, para então selecionar os exemplos em que cada classificador esta mais seguro sobre o rótulo fornecido. Estes rótulos serão então atribuídos aos seus respectivos exemplos presentes na base de dados do outro classificador. Em outras palavras, cada classificador seleciona alguns exemplos para colocar no conjunto de treinamento do outro classificador. Depois disso, usando o novo conjunto de dados rotulados aumentado pela outra visão, ambos os classificadores são treinados novamente. Esta rotina continuará até que algum critério de parada seja encontrado.

Nesta abordagem, segundo Blum e Mitchell (1998) fortes premissas sobre os atributos dos dados são necessárias para garantir o sucesso do Co-Training:

1. Cada visão necessita ser suficiente para construir um bom classificador (Premissa de suficiência);
2. As visões devem ser condicionalmente independentes dados os rótulos dos dados (Premissa de independência).

Nas próximas seções deste capitulo, abordaremos as principais variações do Co-

Algoritmo 1: Co-Training

Entrada: Conjunto de treinamento $Z = L \cup U$, em que L contém l exemplos de treinamento rotulados e U contém u exemplos de treinamento não-rotulados; número de iterações Q. Número p de exemplos positivos a serem rotulados por iteração; Número n de exemplos negativos a serem rotulados por iteração;

Saída: Classificador $H = \{h_1, h_2\}$, novos exemplos rotulados:

início

Construir as visões V_1 e V_2 a partir de L;

Inicializar classificadores h_1, h_2, usando a base de dados rotulada em L;

para *q=1...Q* **faça**

para *j=1...2* **faça**

Selecionar uma amostra U' de exemplos aleatoriamente de U;

Treinar h_j usando somente V_j;

Usar h_j para rotular p exemplos positivos e n negativos de $U^{'}$;

fim

Adicionar os exemplos recém-rotulados em L;

Aleatoriamente recolher $2p + 2n$ exemplos de U para reabastecer U';

fim

retorne $H = \{h_1, h_2\}$;

fim

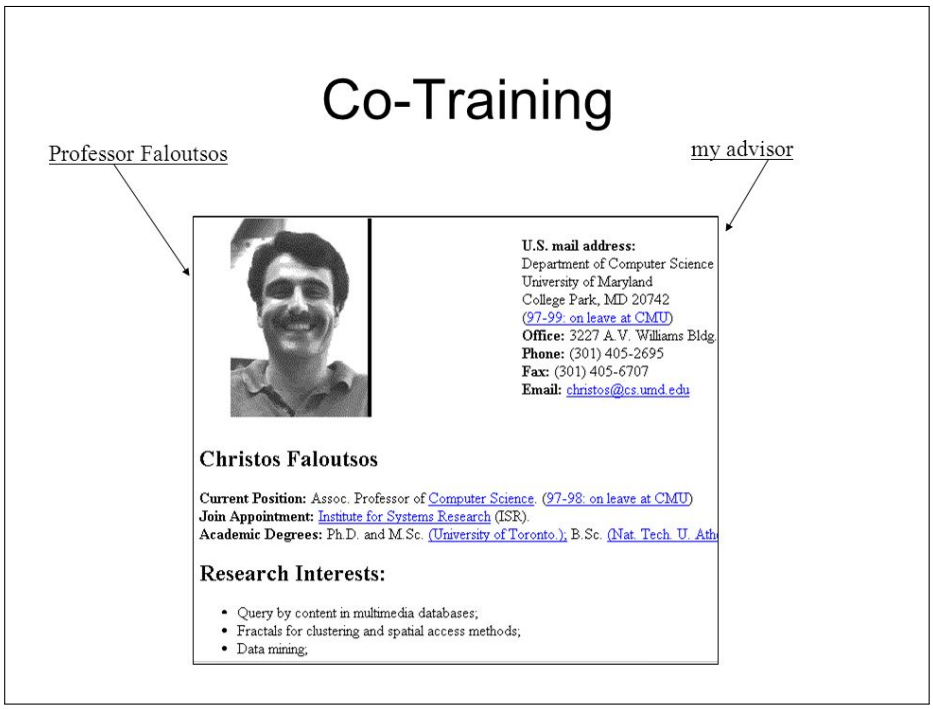
Training, suas aplicações bem como um *framework* genérico voltado a sua aplicações.

2.1.1 Variações do Co-training

Cada classificador do Co-Training necessita de um algoritmo base para formular sua hipótese, e dada a sua simplicidade, a abordagem foi incorporada a diversos classificadores. Alguns autores simplesmente utilizaram a essência do Co-Traning para formular novos estilos de Co-Training, alcançando resultados encorajadores em diferentes áreas. Outros simplesmente utilizaram a abordagem padrão em conjunto com algoritmos supervisionados para estendê-los ao paradigma semi-supervisionado. Nesta subseção, serão vistas algumas das principais metodologias e algoritmos propostos.

Pouco tempo depois do Co-Traning ser apresentado à comunidade cientifica, as premissas de suficiência e independência foram justificadas através de um estudo teórico e um novo modelo de Probabilidade Aproximadamente Correta (PAC) de generalização do erro foi formulado (DASGUPTA *et al.*, 2002). Entretanto, no contexto real, essas premissas são

Figura 2 – Diferentes tipos de visões para um mesmo problema.



Fonte: Blum e Mitchell (1998).

dificilmente encontradas, e em virtude disso outros antecedentes menos rigorosos sobre a base de dados foram fornecidas pela comunidade ao longo do tempo, como uma forma de relaxamento das premissas originais do Co-Traning. Conforme Abney (2002), em alguns casos, mesmo com uma condição menos rigorosa da premissa de independência, isto é, visões que não são totalmente independentes, dado o rótulo de um exemplo, Co-Traning pode funcionar bem. Já para Balcan *et al.* (2004), a premissa relaxada chamada ε-expansão apresentou uma boa aproximação do modelo de aprendizado PAC, garantindo o sucesso do Co-Traning sobre certas distribuições dos dados.

Muitas outras abordagens foram propostas na literatura para solucionar o problema da suficiência e independência. Uma destas foi o Co-EM, um algoritmo que incorpora as ideias de *Expectation Maximization* dentro do Co-Traning, mostrando uma pequena melhora de performance em relação ao Co-Traning em conjuntos de dados de classificação de textos, além de demonstrar também menor sensibilidade à independência dos classificadores (NIGAM; GHANI, 2000). Neste mesmo contexto, surge o *Democratic Co-learning* (DCL), um algoritmo operando em somente uma visão dos dados, e não necessitando de duas visões independentes e redundantes de atributos (ZHOU; GOLDMAN, 2004). Conforme o autor, no *Democratic Co-learning* um conjunto de classificadores é treinado com a base de dados original usando

todos os atributos. Além disso, este método usa voto majoritário e intervalos de confiança estatística para inferir rótulos aos dados não rotulados e decidir quais exemplos podem ser adicionados no novo conjunto de dados rotulados. De forma similar, TriTraining foi proposto usando três classificadores diferentes treinados com reposição de amostras (ZHOU; LI, 2005). Mais tarde, os mesmos autores propuseram outro algoritmo chamado Co-Forest, onde um conjunto de classificadores formados usando o algoritmo *Random Forest* se aliam ao paradigma multivisão (LI; ZHOU, 2007). O algoritmo começa com uma seleção de amostras dos dados rotulados originais, para então treinar diversos classificadores que serão refinados usando os novos exemplos rotulados em cada iteração durante o processo de treino. No final, a predição de um exemplo qualquer será feita usando voto majoritário.

Anos mais tarde, uma nova abordagem denominada Co-Training por comitê, obteve bons resultados ao ser testada com três diferentes classificadores base, sendo eles *Bagging*, *Adaboost* e *Random Subspace*. Nesta metodologia, um conjunto de classificadores é formado usando o conjunto de dados rotulados original. Depois disso, alguns exemplos não rotulados são escolhidos aleatoriamente, e usando voto majoritário, os dados não rotulados ganham novos rótulos. Após isso, uma pequena amostra de exemplos confiáveis é adicionada no novo conjunto de treino do comitè. Esta metodologia difere do DCL ao utilizar algoritmos base que aplicam subconjuntos de atributos dos dados originais ou atributos extras gerados artificialmente, gerando maior diversidade entre os classificadores.

Recentemente, um novo método chamado DCPE-Co-Training , baseado nas premissas originais de independência e suficiência, ganhou a atenção da comunidade ao usar a diferença da Probabilidade Estimada da Classe (PEC) , para aumentar a diversidade dos classificadores, obtendo resultados promissores (XU *et al.*, 2012). Primeiro, ambos os classificadores são treinados e usados para classificar uma pequena amostra aleatória dos dados não rotulados. Os exemplos em que ambos os classificadores concordam sobre o rótulos, com grande diferença na PEC são escolhidos por cada classificador, maximizando a PEC do exemplo para o classificador em questão. Logo após esse procedimento, os classificadores são treinados usando o conjunto de dados recém aumentado, repetindo este processo até que não sobrem mais exemplos ou um outro critério de parada seja cumprido.

Por fim, mas longe de cobrir toda a gama de varições do Co-Training, foi proposta pela comunidade um novo algoritmo chamado *Random Subspace Co-Training* (Rasco, (HO, 1998)), uma abordagem que combina as ideias do Co-Training com *Random Subspace*, um

método que gera subconjuntos de atributos aleatoriamente. O autor forma um conjunto de classificadores com subespaços do conjunto original de dados, apresentando uma melhora de performance quando comparado a algoritmos tradicionais como o TriTraining e o Co-Training clássico para alguns conjuntos de dados (WANG *et al.*, 2008). Algum tempo depois, *Relative Random Subspace Co-Training* (Rel-Rasco, (YASLAN; CATALTEPE, 2010)), uma extensão do trabalho anterior, foi proposta usando informação mútua para gerar os subespaços. Esta versão apresentou melhor performance nos casos em que existiam atributos irrelevantes, número de classificadores reduzido ou quando o numero de atributos era pequeno. De forma geral, Rel-Rasco apresentou melhores resultados que Rasco.

Como visto, existem inúmeras variações de Co-Training em diversos paradigmas a fim de construir as visões de forma independente quando não disponibilizadas. Algumas abordagens ainda utilizam diferentes algoritmos base em seus classificadores, outras criam diferentes subconjuntos de atributos ou simplesmente usam reposições de amostras de dados diferentes.

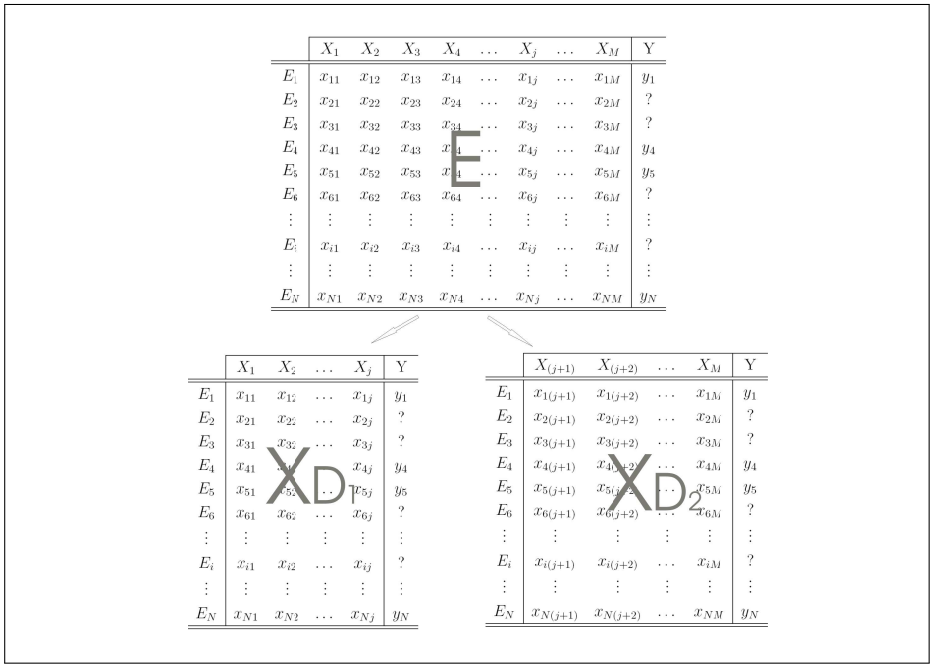
2.1.2 Formalização e Framework

Recentemente, um *framework* genérico para aplicações voltadas ao Co-Training foi apresentado como forma de solucionar a grande variedade de algoritmos baseados nesta abordagem (LIU *et al.*, 2015). Neste *framework*, foram propostos 3 arquétipos de adaptação: multivisão, multi-algoritmos e multi-subconjuntos, representando a grande variedade existente de modificações. O intuito deste trabalho é utilizar o primeiro arquétipo (multivisão) para propor um novo algoritmo baseado em Co-Training.

Formalmente falando, sejam $\mathscr{X} \subset \mathbb{R}^D$ e $\mathscr{Y} \subset \mathbb{R}^S$, respectivamente os espaços de entrada e saída, e dado o conjunto de dados $Z = L \cup U$, onde $L = \{(x_i, y_i)\}_{i=1}^{l} \in \mathscr{X} \times \mathscr{Y}$ é a base de dados rotulada e $U = \{(x_i)\}_{i=l+1}^{n} \in \mathscr{X}$ a base de dados não rotulada. Então, os l primeiros exemplos de Z serão considerados o conjunto de dados rotulados enquanto que os $u = n - l$ últimos exemplos serão os dados não rotulados, ambos contidos no conjunto de treinamento. Assuma que $H = \{h_1, h_2\}$, é nossa função objetivo ou hipótese, onde h_1 e h_2 serão as hipóteses definidas sobre diferentes visões, ou seja diferentes classificadores, subconjunto de atributos ou de amostras de dados.

Inicialmente, dois classificadores h_1 e h_2 são formados usando um dos 3 arquétipos. Após isso, cada classificador atribui rótulos a todos os dados não rotulados ordenando os

Figura 3 – Criação das visões com subconjuntos de atributos.



Fonte: Blum e Mitchell (1998)

exemplos a partir da confiança de suas predições, para então adicionar os *e* exemplos rotulados mais confiáveis dentro do novo conjunto de treinamento.

Depois deste processo, ambos os classificadores são treinados novamente usando o conjunto de dados aumentado com os novos dados rotulados fornecidos anteriormente. Este processo será repetido até que os classificadores convirjam ou um numero de interações seja alcançado. O algoritmo 2 sumariza o funcionamento do *framework*.

2.2 Máquina de Aprendizagem Mínima

Recentemente, o algoritmo Máquina de Aprendizagem Mínima, do inglês *Minimal Learning Machine* (MLM, (A.H. *et al.*, 2013)) foi proposto para solucionar problemas de regressão e classificação, tendo alcançado resultados competitivos no paradigma de Aprendizado Supervisionado. Algumas das vantagens do MLM estão em sua formulação simples, fácil implementação e a existência de somente um hiperparâmetro, necessitando portanto de ajustes em uma única variável. Neste seção, será descrito brevemente o funcionamento do MLM, bem como sua variação rápida o *Nearest Neighbor Minimal Learning Machine* (NN-MLM).

Algoritmo 2: *Framework* genérico para Co-Training

Entrada: Conjunto de treinamento $Z = L \cup U$, onde L contém l exemplos de treinamento rotulados e U contém u exemplos de treinamento não rotulados. Número de exemplos e a serem adicionados por iteração

Saída: Classificador $H = \{h_1, h_2\}$: **início**

Inicializar classificadores h_1, h_2, usando a base de dados rotulada L.

repita

Aplicar o classificador $h_i (i = 1,2)$ para predizer rótulos dos dados não rotulados em U;

Estimar a confiança de rotulagem para cada classificador;

Escolher e exemplos relativamente confiáveis de U, removê-los e adicioná-los em L;

Atualizar os classificadores $H = \{h_1, h_2\}$ usando o novo conjunto atualizado de treinamento;

até *Critério de parada*;

retorne $H = \{h_1, h_2\}$, conjunto L de dados rotulados;

fim

2.2.1 Introdução ao MLM

Dada uma quantidade finita de exemplos N, sendo $\mathscr{X} \subset \mathbb{R}^D$ e $\mathscr{Y} \subset \mathbb{R}^S$, respectivamente, os espaços de entrada e saída, define-se o conjunto de dados $\mathscr{D} = \{(\mathbf{x}_n, \mathbf{y}_n = f(\mathbf{x}_n))\}_{n=1}^N$, de tal forma que $\mathbf{x}_n \in \mathscr{X}$ e $\mathbf{y}_n \in \mathscr{Y}$. Supondo a existência de uma função f que mapeia o espaço de entrada $\mathscr{X}$ para o espaço de saída $\mathscr{Y}$, define-se para esta função a seguinte expressão:

$$\mathscr{Y} = f(\mathscr{X}) + \mathbf{R},$$

Em que a matriz R representa a quantidade de resíduo resultante da aproximação do mapeamento f, em que cada linha $N \times S$ de $\mathbf{R}$, refere-se ao resíduo do mapeamento de uma instância $\mathbf{x}$ para sua respectiva saída $\mathbf{y}$. O principal objetivo do MLM é encontrar a função de mapeamento linear $\hat{f}$ que mais se aproxime de f. Para Junior *et al.* (2013), MLM pode ser dividida em duas etapas: na primeira etapa, modificamos o espaço de entrada e de saída usando funções substitutas $\delta_k : \mathscr{Y} \to \mathbb{R}_+$ e $d_k : \mathscr{X} \to \mathbb{R}_+$, que podem ser obtidas a partir da distância dos exemplos entre alguns pontos fixos selecionados, chamados de pontos de referência $\{(\mathbf{m}_k, \mathbf{t}_k = f(\mathbf{m}_k)) \in \mathscr{D}\}_{k=1}^K$, e após isso, supondo a existência de um mapeamento entre os novos espaços de entrada e saída, MLM o calculará. Conforme o mesmo autor, o segundo passo

consiste em usar a aproximação fornecida das distâncias de saída (obtidas no passo anterior) para, através de sua configuração geométrica, estimar a resposta $\mathbf{y}$ no espaço original $\mathscr{Y}$. Ambos os passos serão discutidos nas próximas seções:

2.2.1.1 Regressão das distâncias

Inicialmente, define-se o conjunto de pontos de referência como um número K de pontos aleatoriamente selecionados do conjunto de dados $\mathscr{D}$. Seja $R = \{\mathbf{m}_k\}_{k=1}^{K}$, um subconjunto de pontos no espaço de entrada $\mathscr{X}$, e $T = \{\mathbf{t}_k\}_{k=1}^{K}$, o respectivo subconjunto de pontos de saída de R no espaço $\mathscr{Y}$, ambos contidos no conjunto de treinamento $\mathscr{D}$. Neste trabalho, como forma de evitar ambiguidades, serão referenciados em eventos futuros o conjunto R, como pontos de referência de entrada, e o conjunto T, como pontos de referencia de saída. O próximo passo consiste em mapear todos os pontos presentes em $\mathscr{D}$ dos espaços $\mathscr{X}$ e $\mathscr{Y}$, a partir dos conjuntos de pontos de referência R e T respectivamente. Esse remapeamento consiste na distância euclidiana entre um ponto qualquer presente em $\mathscr{D}$ a todos os pontos referência. Para tal, definimos $\mathbf{D}_x \in \mathbb{R}^{N\times K}$, de tal maneira que a k-th coluna $\mathbf{d}(X,\mathbf{m}_k)$ contenha as distâncias $d(\mathbf{x}_i,\mathbf{m}_k)$ entre os N pontos de entrada $\mathbf{x}_i$ e o k-th ponto de referência $\mathbf{m}_k$. Analogamente, definiremos $\Delta_y \in \mathbb{R}^{N\times K}$ de tal forma que a k-ésima coluna $\delta(Y,\mathbf{t}_k)$ contenha a distância $\delta(\mathbf{y}_i,\mathbf{t}_k)$ entre os N pontos de saída $\mathbf{y}_i$ e a saída $\mathbf{t}_k$ do k-ésimo ponto de referência.

Supondo a existência de um mapeamento g entre as funções de distâncias d e δ, será reconstituída a função $g : \mathbb{R}_+^K \to \mathbb{R}_+$ entre $\mathbf{D}_x$ e Δ_y a partir do modelo de regressão multi-responsivo $\Delta_y = g(\mathbf{D}_x) + \mathbf{E}.$, para $n = 1,\ldots,N$, em que a matriz $\mathbf{E} \in \mathbb{R}^{N\times K}$ representa os resíduos do mapeamento g, de tal forma que cada resíduo presente na linha n representa a diferença entre as instâncias das linhas $\Delta_{y(i)}$ e $g(\mathbf{D}_{x(i)})$.

No sistema matricial adotado, cada linha de índice n da matriz $\mathbf{D}_x$ representa um vetor de entrada correspondente a uma instância do problema, após a reconfiguração das distâncias. De forma semelhante cada linha de índice n da matriz Δ_y representa a saída associada a esta instância. MLM supõe que um modelo linear é suficiente para descrever as conexões do espaço de entrada para o de saída, sendo o mesmo definido como: $\Delta_y = \mathbf{D}_x\mathbf{B} + \mathbf{E}$. É importante notar que a matriz de regressão $\mathbf{B} \in \mathbb{R}^{K\times K}$, contém em cada uma de suas colunas os coeficientes para cada uma das K respostas referentes aos K pontos de referência.

Os coeficientes da matriz $\mathbf{B}$ podem ser computados minimizando-se a seguinte

função de custo abaixo:

$$J(\mathbf{B}) = ||\mathbf{D}_x\mathbf{B} - \Delta_y||_F. \tag{2.1}$$

Uma vez que os pontos de referência são coletados da base de dados, tem-se que $K \leq N$. O número de pontos de referência é um importante fator na construção de um modelo baseado em MLM. Enquanto que um número pequeno pode não reproduzir um modelo muito confiável, um número muito grande pode construir um modelo enviesado. De uma maneira geral, o numero de pontos de referência será menor que o numero de pontos disponíveis, ou seja $K < N$. Para estes casos a matriz $\mathbf{B}$ pode ser aproximada usando-se o algoritmo de Mínimos Quadrados Ordinário (MQO), (DISMUKE; LINDROOTH, 2006)):

$$\hat{\mathbf{B}} = (\mathbf{D}_x'\mathbf{D}_x)^{-1}\mathbf{D}_x'\Delta_y. \tag{2.2}$$

Seja $\mathbf{d}(\mathbf{x},R) = [d(\mathbf{x},\mathbf{m}_1)\ldots d(\mathbf{x},\mathbf{m}_K)]$, o vetor que contém as distâncias de um ponto de teste $\mathbf{x} \in \mathbb{R}^D$ aos K pontos de referência de entrada, $\{\mathbf{m}_k\}_{k=1}^K$. O objetivo do MLM é encontrar uma estimativa aproximada da saída $\mathbf{y}$, no entanto é conhecida somente uma estimativa das respectivas distancias entre $\mathbf{y}$ e os $\mathbf{K}$ pontos de referência de saída, $\{\mathbf{t}_k\}_{k=1}^K$, a qual pode ser obtida a partir da equação abaixo:

$$\hat{\delta}(\mathbf{y},T) = \mathbf{d}(\mathbf{x},R)\hat{\mathbf{B}}. \tag{2.3}$$

Conforme (JUNIOR *et al.*, 2013) o vetor $\hat{\delta}(\mathbf{y},T) = [\hat{\delta}(\mathbf{y},\mathbf{t}_1)\ldots\hat{\delta}(\mathbf{y},\mathbf{t}_K)]$, pode ser entendido como uma estimativa da configuração geométrica de $\mathbf{y}$ e do conjunto de referência $\mathbf{T}$, dentro do espaço de saída $\mathscr{Y}$.

2.2.1.2 Estimação da saída

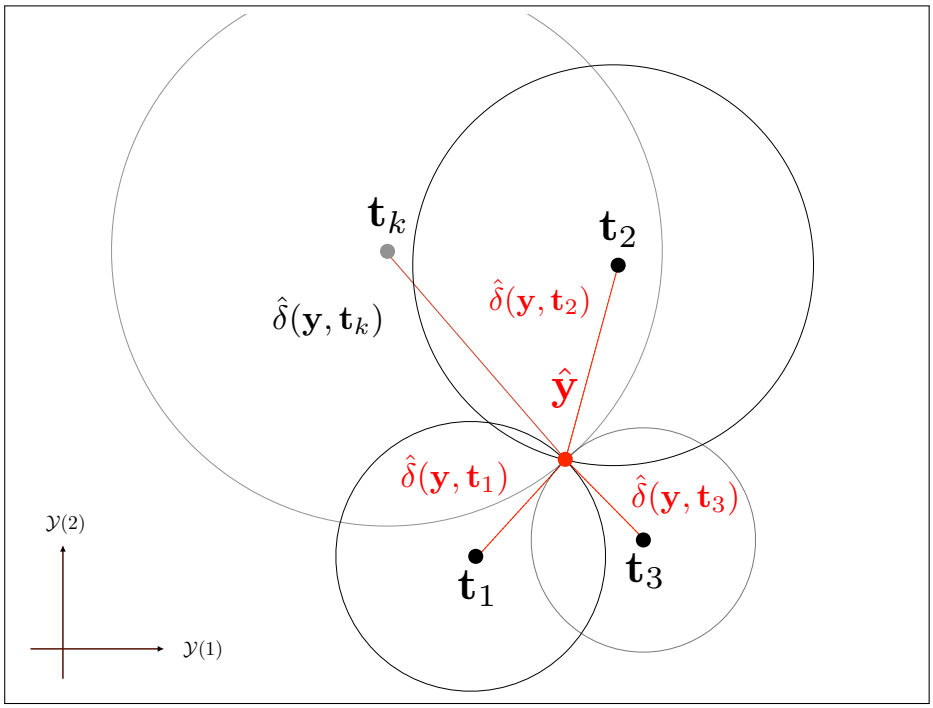
Conforme (NIEWIADOMSKA-SZYNKIEWICZ EWA, 2009), em uma rede de sensores, um problema de multilateração tem interesse em determinar geograficamente a localização de um nó (i.e um sensor), baseada nas distâncias desde nó aos demais nós presentes na rede.

A etapa anterior do MLM não fornece uma estimativa de $\mathbf{y}$, calculando somente uma aproximação das distâncias de $\mathbf{y}$ aos K pontos de referência de saída. Essa informação no entanto, pode ser utilizada para encontrar a saída esperada $\mathbf{y}$ no espaço $\mathscr{Y}$, uma vez que é

possível reformular este problema como um problema de multilateração, tratando cada ponto de referência como um nó, e a saída $\mathbf{y}$, como o nó a ser encontrado.

Conforme Junior *et al.* (2013), a partir da configuração geométrica dos pontos, localizar $\mathbf{y} \in \mathbb{R}^S$ é equivalente a solucionar um sistema sobredeterminado de um conjunto de K equações não lineares correspondentes a $(S+1)$ hiperesferas, no plano $(S+1)$-dimensional, centralizadas em $\mathbf{t}_k$ e passando por $\mathbf{y}$.

Figura 4 – Estimativa de saída representada graficamente para o problema $S = 2$.



Fonte: A.H. *et al.* (2013)

O número de hiperesferas será igual ao número de pontos de referência escolhido para a construção do modelo, onde cada hiperesfera $\mathbf{k}$ terá raio igual a distância entre o ponto de referência de saída $\mathbf{t_k}$ e o ponto de saída procurado $\mathbf{y}$. Em outras palavras cada uma das hiperesferas de $k = 1, \ldots, K$ terá raio igual a $\hat{\delta}(\mathbf{y}, \mathbf{t}_k)$ e uma vez que:

$$(\mathbf{y} - \mathbf{t}_k)'(\mathbf{y} - \mathbf{t}_k) = \hat{\delta}^2(\mathbf{y}, \mathbf{t}_k), \tag{2.4}$$

a localização de $\mathbf{y}$ será estimada a partir da minimização da seguinte função de custo

$$J(\mathbf{y}) = \sum_{k=1}^{K} \left((\mathbf{y} - \mathbf{t}_k)'(\mathbf{y} - \mathbf{t}_k) - \hat{\delta}^2(\mathbf{y}, \mathbf{t}_k) \right)^2. \tag{2.5}$$

Conforme Junior *et al.* (2013) a função de custo tem mínimo igual a 0 e pode ser resolvida se, e somente se, $\mathbf{y}$ é uma solução de (2.4). Se a solução existir, ela é global e única.

Segundo o mesmo autor, devido à incerteza introduzida ao estimar $\hat{\boldsymbol{\delta}}(\mathbf{y}, \mathbf{t}_k)$, uma solução ótima de (2.5) pode ser encontrada minimizando $\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmin}}\, J(\mathbf{y})$ a partir de métodos baseados no gradiente descendente para a solução de sistemas não lineares, sendo adotado o método de Levenberg-Marquardt (LM, (MARQUARDT, 1963)).

2.2.2 MLM para classificação

O método MLM clássico oferece um modelo de predição para problemas de regressão. Uma vez que problemas de classificação são de extrema importância em Aprendizado de Máquina de uma maneira geral, uma extensão para problemas de classificação será discutida nesta subseção. Seja $\mathscr{D}$ um conjunto de dados com N pontos de entrada $X = \{\mathbf{x}_i\}_{i=1}^{N}$, com $\mathbf{x}_i \in \mathbb{R}^D$, e um conjunto $L = \{l_i\}_{i=1}^{N}$, contendo as respectivas saídas. Neste caso define-se que as saídas serão denominadas classes ou rótulos de tal forma que $l_i \in \{C_1, \ldots, C_S\}$, no qual C_j denota a j-ésima classe. O padrão de classificação binária, consiste no caso em que $S = 2$. Para qualquer valor acima desse, o problema será tratado como uma classificação multiclasse.

A extensão do MLM para classificação pode ser feita usando o tradicional sistema de codificação 1-por-S, que consiste em utilizar um vetor como variável qualitativa, formado por S variáveis binárias, em que S representa o número de classes. Neste tipo de abordagem, cada variável binária pode assumir somente dois valores, 0 ou 1, sendo estes designados como pulso alto ou baixo. No sistema de classificação tradicional, uma classe pode ser representada por um pulso alto na variável que designa a classe à qual a instância pertence, enquanto que as demais variáveis devem assumir um pulso baixo. Em outras palavras, o vetor conterá uma ou mais variáveis de 0's e uma única variável com o valor 1, representando a classe pertencente do exemplo em questão. Formalmente falando, o j-ésimo componente de saída do vetor $\mathbf{y}_i$ terá o valor 1 se $l_i = C_j$ e 0 caso contrário.

O teste de uma instância pode ser feito da seguinte forma: coleta-se uma observação $\mathbf{x}$ cuja classe é desconhecida $l \in \{C_1, \ldots, C_S\}$, a classe estimada $\hat{l}$ associada à saída extipulada $\hat{\mathbf{Y}}$ será $\hat{l} = C_{s^*}$, na qual:

$$s^* = \underset{s=1,\cdots,S}{\operatorname{argmax}} \{\hat{\mathbf{Y}}^{(s)}\}, \tag{2.6}$$

em que $\hat{\mathbf{Y}}^{(s)}$ denota o s-ésimo componente do vetor $\hat{\mathbf{Y}}$, enquanto que s^*, o índice associado à variável com maior valor.

2.2.3 Nearest Neighbor-Minimal Learning Machine (NN-MLM)

Apesar de obter resultados satisfatórios, tanto em problemas de regressão quanto classificação, MLM contém etapas em sua fase de treinamento e teste que consomem considerável intervalo de tempo. Para Mesquita *et al.* (2017), uma possível forma de aumentar a velocidade do método para problemas de classificação está em combinar MLM com o algoritmo de K-Vizinhos Mais Próximos, do inglês *K-Nearest Neighbor* (KNN, (ALTMAN, 1992)). Neste caso, descarta-se a segunda etapa do MLM para utilizar o KNN. Para achar a classe l de uma instância $\mathbf{x}$, basta selecionar o ponto de referência $\mathbf{t_k}$ mais próximo da saída $\mathbf{y}$ e associar a classe de $\mathbf{t_k}$ a $\mathbf{x}$. Uma vez que a matriz δ_y já contém as distâncias aproximadas de $\mathbf{y}$ a todos os pontos de referência de saída $\mathbf{t_k}$, basta utilizar o método KNN para achar o ponto $\mathbf{t_k}$ mais próximo de $\mathbf{y}$. Esta variação do MLM, foi batizada de NN-MLM.

Conforme (MESQUITA *et al.*, 2017), foi demonstrado que para problemas de classificação, s^* será a classe associada a $\hat{\mathbf{y}}$ se e somente se, s^* for o índice diferente de zero da entrada de $\mathbf{t_k}$, no qual

$$S = \underset{s\in\{1,\cdots,S\}}{\operatorname{argmin}} \hat{\delta}(y,t_s).$$

Este resultado implica que s^* deve ser a mesma classe do ponto $\mathbf{t_k}$ mais próximo em $\mathbf{T}$. Ou seja, as saídas de MLM e NN-MLM serão necessariamente as mesmas. Em outras palavras, NN-MLM se torna vantajoso por não necessitar de métodos baseados em gradiente descendente, agilizando a etapa de treinamento e teste, mas sem perder acurácia.

2.2.4 Análise de complexidade

Como já descrito anteriormente, o algoritmo tradicional do MLM pode ser dividido em duas etapas:

1. Regressão das distâncias;
2. Problema de multilateração.

A primeira parte consiste no cálculo das matrizes de distâncias de entrada e saída, D_x e δ_y, e no cálculo da solução de mínimos quadrados para achar o coeficiente B da regressão linear presente na equação (2.2). Uma vez que dispõe-se de K pontos de referência e N exemplos, o cálculo das matrizes de distância levará tempo da ordem de $\Theta(KN)$. O calculo o coeficiente B

requer o cálculo de uma pseudoinversa de *Moore-Penrose*, que leva através do algoritmo SVD o tempo da ordem de $\Theta(K^2N)$.

A segunda parte do MLM requer a solução de um problema de multilateração para achar o ponto aproximado $\hat{\mathbf{y}}$ através da distâncias deste ponto aos pontos de referência. Considera-se que o algoritmo usado para este problema é Levenberg-Marquardt, devido à sua convergência ser rápida e estável, apesar de qualquer método baseado no gradiente descendente poder ser utilizado para solucionar o problema de minimização em (2.1) (MESQUITA *et al.*, 2017). Conforme o mesmo autor, o método MLM necessita da computação de uma matriz jacobiana $\mathbf{J} \in \mathbb{R}^{K \times S}$ e a inversa $\mathbf{J}^T\mathbf{J}$, tendo MLM uma complexidade aproximada de $\Theta(I(KS^2 + S^3))$, em que S representa a dimensionalidade de $\mathbf{y}$ e I o numero de iterações do método.

A versão NN-MLM escolhe o ponto de referência mais próximo de $\mathbf{y}$. Uma vez que dispõe-se de somente K pontos de referência e desprezando os fatores constantes, sua complexidade para achar a classe associada a $\mathbf{y}$ será aproximadamente de $\Theta(K)$. Somando os temos dos dois passos, a complexidade final do MLM será de $\Theta(K^2N + I(KS^2 + S^3))$ enquanto que para a versão NN-MLM será de $\Theta(K^2N)$.

2.3 Considerações finais

Neste capítulo abordou-se o *framework* Co-Training, o qual constitui parte integral deste trabalho. Além disso, definiram-se formalmente os métodos MLM e NN-MLM, cobrindo toda a fundamentação teórica necessária para a compreensão desta dissertação.

3 METODOLOGIA

Nesta seção, será apresentada uma proposta de variação do Co-Training usando como classificador base o método MLM, nomeada de Co-MLM. Além disso a fim de otimizar a velocidade do método Co-MLM, será detalhada também uma versão rápida, chamada Fast Co-MLM, baseada no método NN-MLM.

3.0.1 Co-MLM

A variação proposta, usará como metodologia aplicar o *framework* já mencionado anteriormente, escolhendo-se o arquétipo multi visão. Neste caso, ambas as hipóteses presentes em $H = \{h_1, h_2\}$ terão como base o classificador MLM, sendo formadas sobre diferentes sub conjunto de atributos.

Supondo a existência do conjunto de dados já detalhado anteriormente, dá-se inicio a formalização do Co-MLM. Inicialmente o conjunto de dados $X \in \mathbb{R}^{N \times D}$, que representa somente a parte que contém os atributos de Z, é aleatoriamente dividido em dois subconjuntos mutuamente exclusivos de atributos V_1 e V_2 de mesmas dimensões, representando duas visões nas quais $V_j \in \mathbb{R}^{N \times \frac{D}{2}}$. Então os dois classificadores fracos (i.e, classificadores construídos com somente um subconjunto de atributos), h_1 e h_2, com diferentes visões, são treinados utilizando somente o conjunto de dados rotulados original L. Para cada classificador, escolhe-se um conjunto K de pontos de referência a partir do algoritmo *k-medoids*, uma tradicional técnica de particionamento para encontrar os exemplos centrais de um conjunto de dados (PARK; JUN, 2009). Uma vez que os subconjuntos de atributos são diferentes, é esperado que diferentes *medoids* sejam escolhidos, aumentando-se a diversidade entre os classificadores.

Então, cada classificador atribui rótulos a todos os dados não rotulados, e juntos selecionam os e exemplos mais confiantes de cada classe para inserirem no conjunto de dados rotulados. Observa-se que neste ponto Co-MLM difere da metodologia tradicional do Co-Training, por não escolher exemplos separados para cada classificador, neste caso a saída combinada dos dois classificadores é preferida, por ter menor chance de inserção de um rótulo errado no conjunto de dados rotulados, uma vez que uma instância será avaliada pelos dois classificadores simultaneamente. A seleção dos melhores exemplos se dará através do uso da saída ponderada dos dois classificadores, baseada no esquema de classificação visto na subseção

2.2.2. Neste caso, $s*$ será calculado como se segue:

$$s^* = \underset{s=1...S}{\operatorname{argmax}} \sum_{j=1}^{2} [W_j * h_{j,s}(V_j(x))] \tag{3.1}$$

Em que o vetor $\hat{Y}_{(j)} = [\hat{Y}_{(j)}^{(1)}, \hat{Y}_{(j)}^{(2)}, ..., \hat{Y}_{(j)}^{(S)}]$ denota a saída estimada para o j-ésimo classificador no esquema de saída 1-por-S, $V_j(x)$ é a visão j para a observação x, $h_{j,s}(V_j(x))$ é a saída provida pelo classificador h_j para a classe s para esta observação, isto é $\hat{Y}_{(j)}^{s}$, e W_j é um fator de peso fornecido pela taxa de acerto de h_j sobre os dados rotulados originais.

Algoritmo 3: Co-MLM

Entrada: Conjunto de treinamento $Z = L \cup U$, onde L contém l exemplos de treinamento rotulados e U contém u exemplos de treinamento não rotulados. Número k de pontos de referência, e número p de exemplos para seleção. Número de exemplos e a serem adicionados por iteração

Saída: Classificador $H = \{h_1, h_2\}$, novos exemplos rotulados:

início

Inicializar classificadores h_1, h_2, usando a base de dados rotulada em L.

repita

para *j=1...2* **faça**

Construir V_j a partir de L;

Usar *K-medoids* para selecionar k pontos de referência, R, de V_j e suas respectivas saídas, T, de Y, usando somente os dados rotulados;

Calcular D_x: A matriz de distância entre V_j e R;

Calcular Δ_y: A matriz de distância entre Y e T;

Calcular $\hat{B}_j = (D_x' D_x)^{-1} D_x' \Delta_y$;

Calcular $\hat{\delta}(\mathbf{y}, T) = d(x, R)\hat{B}_j$; Usar T e $\hat{\delta}(\mathbf{y}, T)$ para achar uma estimativa de $\mathbf{y}$, para todos os dados não rotulados em U;

Calcular W_j;

$h_j = \{B_j, R\}$;

fim

Selecionar e exemplos para cada classe, usando a regra (3.1), e formar o novo conjunto de treinamento, L.

até $\{$*repetições*$\}$ *ou* $\{$*convergência*$\}$ *ou* $\{U = \emptyset\}$;

retorne $H = \{h_1, h_2\}$;

fim

As e observações com maior valor $s*$ para cada classe serão selecionadas e colocadas no novo conjunto de treinamento. Esta medida foi adotada para colocar maior peso de decisão no classificador que obteve melhores resultados, com o intuito de prevenir acúmulo de ruido, ou seja, atribuição de rótulos errados aos novos dados rotulados nas iterações do Co-MLM.

Após isso, ambos os classificadores são treinados novamente, porém com um conjunto de dados aumentado fornecido a cada iteração do algoritmo. Este ciclo será repetido até os classificadores convergirem ou até que um número de iterações seja alcançado.

Finalmente, um classificador forte (i.e que usa todos os atributos), será construído utilizando os dados fornecidos por $H = \{h_1, h_2\}$, adquiridos nas fases Co-Training após Q iterações. O pseudocódigo de Co-MLM pode ser visto em 3, após concluído o processo de rotulação dos novos exemplos, basta-se treinar um novo classificador MLM com o conjunto de dados aumentado.

É importante notar que, a cada iteração, o calculo da matriz de coeficientes B_j deve ser repetida para cada classificador, o que acarreta um grande custo computacional, além de que a obtenção de $\hat{y}$ requer resolver o método *Levenberg–Marquardt*. O elevado custo computacional de Co-MLM motivou a sua modificação a fim de mitigar este problema. Tais modificações podem ser vistas na próxima seção, no método Fast Co-MLM.

3.1 Fast Co-MLM

O algoritmo proposto na seção prévia usa MLM como classificador base, e como já visto anteriormente, tanto o *framework* Co-Training quando o próprio MLM acarretam alta complexidade. O primeiro por necessitar de diversas iterações, e o segundo por precisar do cálculo de um método baseado no gradiente descendente. Para aliviar este problema, foi desenvolvida uma versão rápida, chamada Fast Co-MLM, com duas modificações principais: usar uma fórmula recursiva para calcular e atualizar a matriz de coeficientes B_j em cada iteração; e usar a versão NN-MLM para calcular de maneira mais rápida as saídas esperadas dos classificadores MLM's.

Na primeira modificação, utiliza-se o algoritmo Mínimos Quadrados Recursivo (MQR, (HAYES, 1996)) formulando-se uma atualização para cada iteração q, dos coeficientes de regressão $B_j^{(q)}$ de cada classificador j, a fim de reduzir o custo computacional presente no cálculo dos Mínimos Quadrados Ordinário (MQO, (DISMUKE; LINDROOTH, 2006)). Em outras palavras, ao invés de calcular-se repetidas vezes os valores de B para cada classificador (MQO

Algoritmo 4: Fast Co-MLM

Entrada: Conjunto de treinamento $Z = L \cup U$, onde L contém l exemplos de treinamento rotulados e U contém u exemplos de treinamento não rotulados. Número k de pontos de referência. Número de exemplos e a serem adicionados por iteração

Saída: Classificador $H = \{h_1, h_2\}$, novos exemplos rotulados:

início

Construir as visões V_1 e V_2 a partir de L;

Usar o algoritmo *K-medoids* em V_1 e V_2 para obter,respectivamente, os pontos de referência para h_1 e h_2.

Computar as matrizes de distâncias $\left[D_x^{(0)}\right]_1$, $\left[D_x^{(0)}\right]_2$, $\left[\Delta y^{(0)}\right]_1$ e $\left[\Delta y^{(0)}\right]_2$.

Inicializar $\hat{B}_1^{(0)}$ e $\hat{B}_2^{(0)}$ conforme a equação. (3.2). Inicializar $P_1^{(0)}$ e $P_2^{(0)}$ usando a equação (3.4).

$q \leftarrow 0$

repita

$q \leftarrow q+1$

Computar os pesos W_1 e W_2 dos classificadores h_1 e h_2 baseado em sua acurácia em L.

Selecionar (e remover) de U os e elementos com maiores valores associado de $C^{(s)}$.

Aumentar o conjunto de treinamento L com o elemento selecionado e seu rótulo predito, estimado usando a equação (3.1).

Atualizar as matrizes de distâncias com o novo exemplo adicionado ao conjunto de dados.

Computar $P_1^{(q)}$ e $P_2^{(q)}$ usando a equação (3.4).

Estimar $\hat{B}_1^{(q)}$ e $\hat{B}_2^{(q)}$ usando a equação (3.3).

até $\{repetições\}$ *ou* $\{convergência\}$ *ou* $\{U = \emptyset\}$;

retorne $H = \{h_1, h_2\}$;

fim

em cada iteração), faz-se isto uma única vez, e então, a cada iteração dada na fase Co-Training, apenas atualizaremos os pesos dos coeficientes citados.

Para este propósito, antes da fase interativa do Co-Training, para cada classificador j, inicializa-se a matriz de coeficientes de regressão $B_j^{(0)}$ usando o algoritmo tradicional MQO, como visto abaixo:

$$\hat{B}_j^{(0)} = \left(\left[D_x^{(0)}\right]_j^T \left[D_x^{(0)}\right]_j \right)^{-1} \left[D_x^0\right]_j^T \left[\Delta y^{(0)}\right]_j \tag{3.2}$$

em que $\left[D_x^{(q)}\right]_j$ e $\left[\Delta y^{(q)}\right]_j$ respectivamente denotam as matrizes de distâncias dos espaços de entrada e saída do j-ésimo classificador na iteração q, que neste caso será a primeira iteração.

Além disso, para qualquer iteração $q > 0$, aplica-se a seguinte fórmula recursiva a fim de obter $B_j^{(q)}$:

$$\hat{B}_j^{(q)} = \hat{B}_j^{(q-1)} + P_j^{(q)} \left[D_x^{(q)}\right]_j^T \left(\left[\Delta y^{(q)}\right]_j - \left[D_x^{(q-1)}\right]_j \hat{B}_j^{(q-1)} \right) \tag{3.3}$$

em que $\mathbf{I}$ denota a matriz identidade e $P_j^{(q)}$ é dado por:

$$P_j^{(q)} = \begin{cases} \left[D_x^{(0)}\right]_j^T \left[D_x^{(0)}\right]_j, & \text{se } q = 0; \\ P_j^{(q-1)} - P_j^{(q-1)} \left[D_x^{(q)}\right]_j^T \left(\mathbf{I} + \left[D_x^{(q)}\right]_j P_j^{(q-1)} \left[D_x^{(q)}\right]_j^T \right)^{-1} & \\ \qquad + \left[D_x^{(q)}\right]_j P_j^{(q-1)}, & \text{caso contrário.} \end{cases} \tag{3.4}$$

Utilizando a formulação recursiva vista para o calculo da matriz de coeficientes, torna-se possível, a cada iteração q, calcular a matriz $\hat{B}_j^{(q)}$ sem a necessidade de calcular novamente a equação 2.2. A formulação do algoritmo MQR é usada de forma a diminuir o custo computacional do Co-MLM durante as iterações da fase *Co-traning*, uma vez que seu custo computacional só depende do tamanho do vetor que comporta os atributos e do número de exemplos.

A fim de acelerar a fase de teste, substitui-se o algoritmo MLM por sua versão mais rápida NN-MLM, como classificador base. Vale lembrar que foi demonstrado que o método NN-MLM apresenta exatamente o mesmo resultado do MLM para problemas de classificação, porém com um custo computacional reduzido (MESQUITA *et al.*, 2017). Enquanto que o passo de saída original requer um número arbitrário I de iterações para a convergência, tendo complexidade total de $O(I(KS^2 + S^3))$, a versão rápida reproduz o mesmo resultado com tempo da ordem de $O(K)$, em que S representa a dimensionalidade do vetor de saída.

É importante notar que a nova saída fornecida pelo NN-MLM informa somente a classe de um exemplo, não refletindo o grau de confiança da classificação. Em outras palavras, o método provê s^*, mas não $\hat{Y}$, necessitando-se que a equação (3.1) seja atualizada. No entanto, vale ressaltar que conforme (DALITZ, 2009), diversas maneiras de aproximar a probabilidade a posteriori da classe, podem ser utilizadas quando em conjunto com o método KNN. A partir

disso, defini-se uma nova saída do classificador NN-MLM, como sendo $\hat{Y} = [\hat{Y}_1 \dots \hat{Y}_K]$, de tal forma que para cada entrada l de $\hat{Y}$, a probabilidade da classe a posteriori será de:

$$\hat{Y}_l = \frac{\hat{\delta}^{-1}(Y, t_l)}{\sum_{k=1}^{K} \hat{\delta}^{-1}(Y, t_k)} \tag{3.5}$$

em que $\hat{\delta}^{-1}(Y, t_k)$ denota o inverso distância euclidiana do vetor de saída Y no esquema de codificação 1-por-S ao ponto de referência t_k. O método proposto é resumido no Algoritmo 4.

3.2 Considerações finais

Neste capítulo, apresentamos formalmente os dois métodos de ASS propostos neste trabalho: Co-MLM e Fast Co-MLM. Em adição, foram definidos os algorítimos de ambos os métodos e toda a sua formulação teórica.

4 RESULTADOS

Esta seção é composta pela descrição e resolução de diversos experimentos, separados em duas etapas, a fim de averiguar a eficácia dos métodos propostos. Na primeira etapa, será comparado o método Co-MLM com a versão mais rápida Fast Co-MLM, com o intuito de averiguar a capacidade de aprendizado semi supervisionado de ambos os métodos bem como comprovar a eficácia de Fast Co-MLM em termos de rapidez, sem perda significativa na acurácia.

A segunda etapa consiste em comparar Fast Co-MLM com outros métodos baseados em Co-Training, com o intuito de verificar a taxa de acurácia média de todos os métodos utilizados, destacando os resultados competitivos dos métodos propostos.

4.1 Experimentos Co-MLM vs Fast Co-MLM

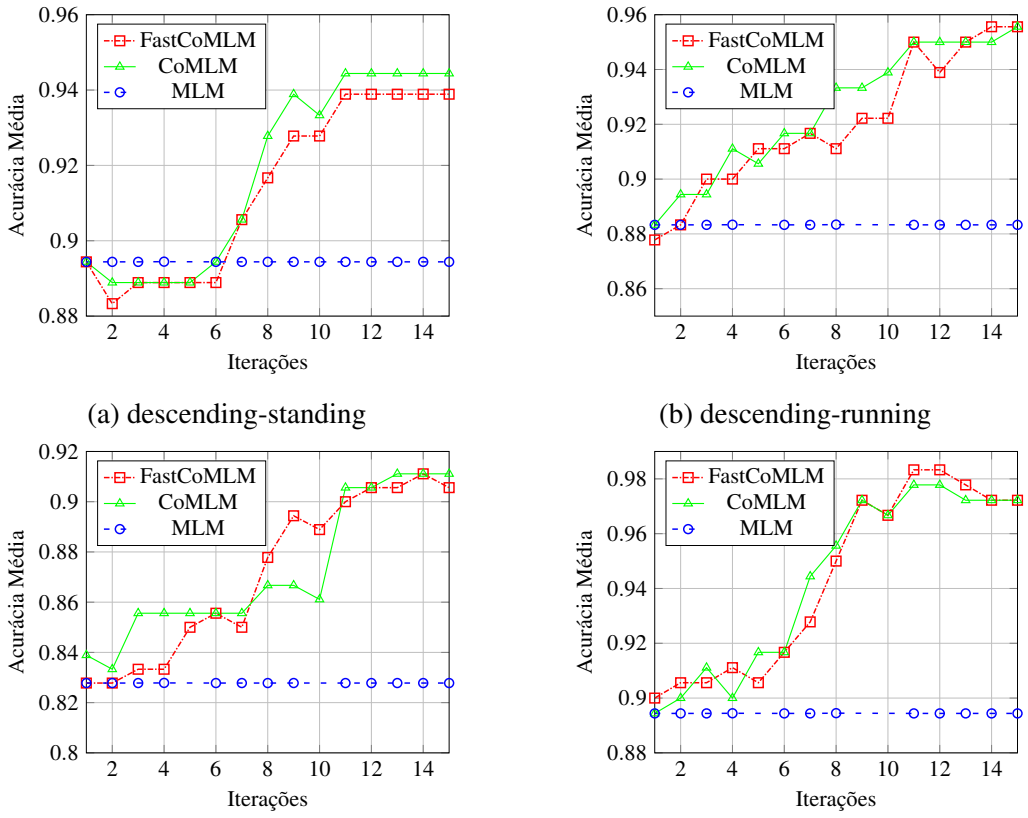
Inicialmente utiliza-se o conjunto de dados *UCF-DataPhone*, uma base real de dados contendo duas visões dos mesmos dados, a fim de ilustrar a capacidade de aprendizado sobre dados não rotulados do algoritmo proposto. A base de dados *UCF-DataPhone* é um conjunto de ações aeróbicas (passear com a bicicleta, escalada, descida de escadas, exercício com a bicicleta, pular, correr, ficar de pé, andar e caminhada na esteira) fornecido pela *University of Central Florida*, obtida a partir de usuários de um *smart phone iPhone* 4 (MCCALL *et al.*, 2012). Foi utilizada a Unidade de Medição Inercial (UMI) a partir um acelerômetro 3D, a velocidade angular (giroscópio) e orientação (magnetômetro) para obtenção dos dados de movimentos. As amostras foram feitas a 60 Hz, e manualmente recortadas em 500 amostras. Em seguida, foram transformadas em exemplo com 1500 atributos que representam uma ação.

Nos experimentos dessa dissertação, foram selecionadas as 6 classes com maior quantidade de exemplos referentes ao acelerômetro e ao giroscópio (45 exemplos por classe), e foi adotado um procedimento "um-vs-todos", que consiste na combinação dois a dois de todas as classes, totalizando 15 diferentes bases de dados. Conforme (LIU *et al.*, 2015), os dados coletados pelo acelerômetro e pelo giroscópio podem ser entendidos como duas visões distintas e suficientes de um mesmo problema.

A condição de suficiência de uma visão pode ser obtida facilmente verificando-se a acurácia do classificador, uma vez que um classificador será suficiente se obtiver bons resultados. Já para (FEGER; KOPRINSKA, 2006), uma forma de calcular a independência entre as visões pode ser obtida através de cálculos baseados na informação mútua. Então, 4 bases de dados com

elevada taxa de acurácia (condição de suficiência) e pequena informação mútua condicionada a classe (condição de independência) entre as visões foram selecionadas para serem usadas neste experimento.

Para validar esta metodologia, foi utilizada validação cruzada com 10 *folds*, e 10 *splits* de cada conjunto de dados, separados em conjunto de treino, teste e validação. Para o conjunto de validação, foram separados 10% do conjunto de dados original, 20% para o conjunto de teste e o restante para o conjunto de treino, em que cada conjunto contém ao menos um exemplo de cada classe. O conjunto de treino foi dividido em dois subconjuntos, L, com l exemplos rotulados, e U com u exemplos não rotulados, com respectivamente 60% e 40% do conjunto de treinamento original. A intenção deste experimento é investigar como Co-MLM e Fast Co-MLM aprendem sobre dados não rotulados na medida em que as iterações avançam, e então comparar ambos os métodos em termos de acurácia e velocidade.



(a) descending-standing

(b) descending-running

(c) climbing-running

(d) climbing-jumping

Figura 5 – Comparação entre Co-MLM e Fast Co-MLM sobre o número de iterações na fase Co-Training. Ambos os métodos foram treinados com 60% da base de dados rotulada (54 exemplos, 27 por classe), sendo refinados adicionando 2 exemplos (um por classe) para cada iteração.

A Figura 5 mostra que quando as iterações avançam, a taxa de acurácia média de

Fast Co-MLM e Co-MLM aumentam, demonstrando a capacidade de aprender usando dados não rotulados de ambos os métodos. Em algum casos, é possível ver uma pequena queda da taxa média de acurácia, que pode ser explicada pela possível adição de dados erroneamente rotulados. Entretanto, na medida que as iterações avançam a quantidade de dados rotulados corretamente tende a compensar este efeito (NIGAM *et al.*, 2000). É importante notar que, Co-MLM e Fast Co-MLM mostraram uma performance similar o que reforça a hipótese inicial de que a proposta modificação não afetará significativamente a performance do Co-MLM. Entre entanto, as modificações definidas para Fast Co-MLM obtiveram alto impacto em termos de velocidade a cada iteração, como pode ser visto na tabela 1 em que cada base de dados representa a junção de duas das quatro classes utilizadas, *climbing*, *jumping*, *descending* e *standing*.

Tabela 1 – Tempo médio de iteração (segundos) para Co-MLM e Fast Co-MLM na base de dados da UCF

Conjunto de dados	Co-MLM	Fast Co-MLM
climbingjumping	7.5677	0.1175
climbingrunning	7.5447	0.1176
descendingrunning	7.5501	0.1191
descendingstanding	7.2634	0.1176

Fonte: o autor

Um segundo experimento foi conduzido a fim de comparar Co-MLM e Fast Co-MLM utilizando 5 (car, german, dbwrold, optdigits e sat) conjuntos de dados reais disponibilizados no repositório UCI (ASUNCION; NEWMAN, 2007), duas bases de dados artificiais (g10n e g50c) usadas em outros trabalhos similares (GRANDVALET *et al.*, 2004; CHAPELLE; ZIEN, 2005) e por fim, o conjunto de dados *uspst* (somente o conjunto de teste), disponível no repositório norte americano Data.Gov (ADMINISTRATION, 2017). Os detalhes acerca do número de atributos, exemplos e classes pode ser visto na tabela 2.

Neste experimento avalia-se o desempenho de cada método quando o número de dados não rotulados varia. Estabelece-se uma porcentagem entre 20% a 80% na taxa de dados não rotulados. Todos os hiper parâmetros de ambos os métodos foram ajustados por um procedimento de validação cruzada de 10 *folds*, semelhante ao do experimento anterior. A Figura 6 mostra a acurácia de 10 execuções onde cada execução conteve 5 iterações na fase Co-Training.

É possível notar que, semelhante ao experimento anterior, não foram observadas diferenças drásticas na taxa média de acurácia entre Co-MLM e Fast Co-MLM, porém com

Tabela 2 – Características dos conjuntos de dados

Conjunto de dados	Instâncias	Atributos	Classes
Car Evaluation (car)	1728	6	4
Gaussians10n (g10n)	548	10	2
German Credit Data (german)	1000	24	2
Landsat Satellite (sat)	1994	36	6
DBWorld e-mails(dbwrold)	62	4702	2
Gaussians50c (g50c)	548	50	2
Optical Recognition of Handwritten Digits (optdigits)	1787	64	10
United States Postal Service Post Office (uspst)	1997	256	10

Fonte: o autor

significativa redução no tempo médio de cada iteração como pode ser observado na tabela 3.

Tabela 3 – Tempo médio (segundos) por iteração para Co-MLM e Fast Co-MLM nas bases de dados da UCI e DataGov

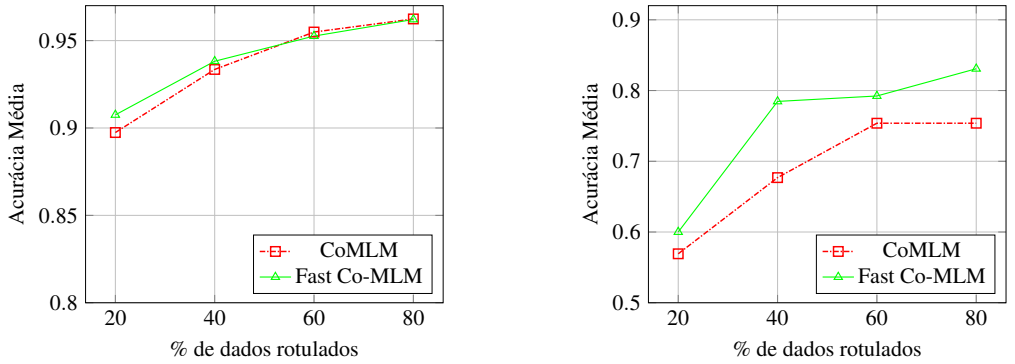
	Fast Co-MLM				Co-MLM			
Conjunto de dados	20%	40%	60%	80%	20%	40%	60%	80%
g10n	0.594	1.323	2.688	4.725	83.788	82.762	97.838	96.406
car	1.520	5.617	16.352	39.581	274.064	306.447	526.848	511.049
german	1.701	6.126	32.082	65.230	130.372	162.059	225.861	398.399
sat	4.912	25.296	113.492	173.255	475.447	628.698	1054.03	1732.99
dbworld	0.085	0.039	0.036	0.059	1.046	0.903	0.943	0.917
g50c	0.537	1.218	2.592	6.222	72.323	108.527	90.154	110.440
optdigits	4.779	20.488	60.914	145.503	561.137	960.134	1361.65	2136.8
uspst	7.934	28.996	84.664	192.489	692.692	1097.632	1788.46	2751.87

Fonte: o autor

4.2 Comparação entre Fast Co-MLM e outros métodos semi-supervisionados

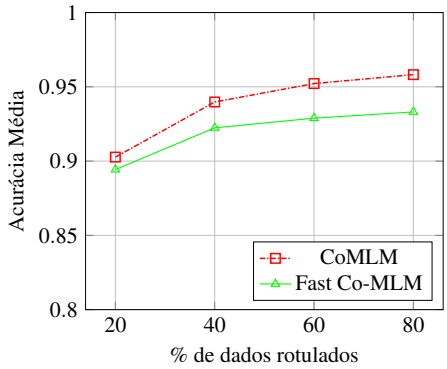
O objetivo deste experimento é comparar a precisão de Fast Co-MLM com outros métodos semi-supervisionados baseados em Co-Training. Configurou-se o experimento com os mesmos 8 conjuntos de dados utilizados na seção anterior. O número de exemplos marcados variou de 20% a 80% e cada método executou 5 iterações. Comparou-se Fast Co-MLM com 3 métodos bem conhecidos de ASS: TriTraining, Co-Forest e o Co-Training padrão com método K-Vizinhos Mais Próximos como classificador base.

Para promover uma comparação justa, todos os métodos tiveram seus hiper parâmetros ajustados por um procedimento de validação cruzada de 10 *folds*. Similar aos demais experimentos, repetiu-se 10 vezes cada uma das avaliações a fim de garantir a sua validade, sendo a acurácia apresentada na Figura 7.

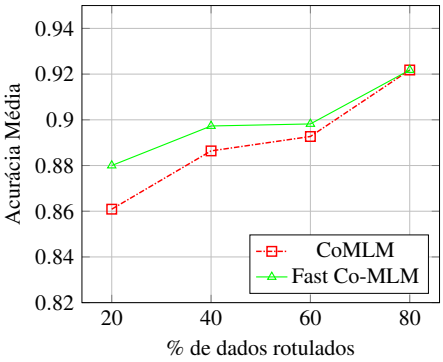


(a) car

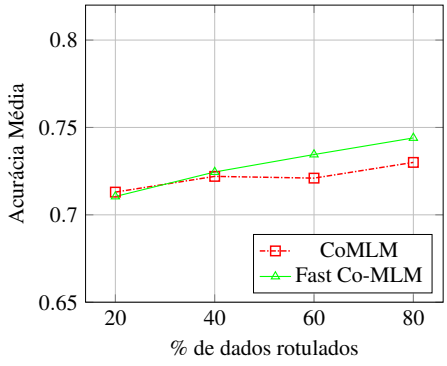
(b) dbworld



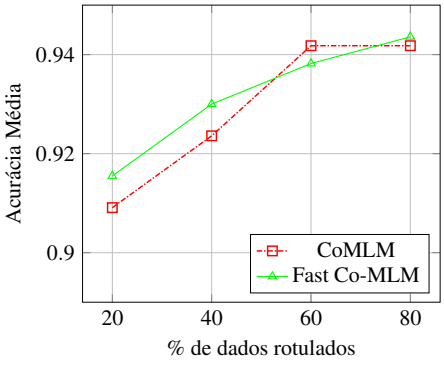
(c) uspst



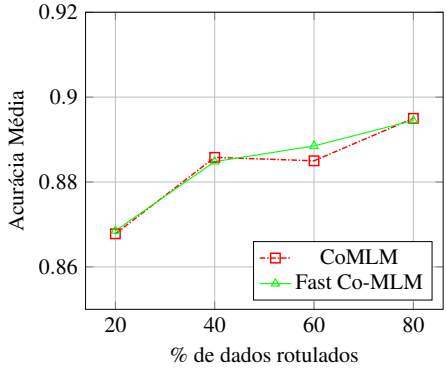
(d) g10n



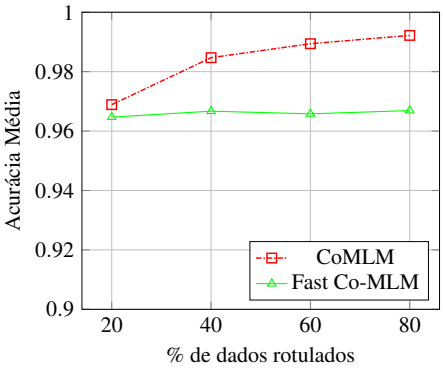
(e) german



(f) g50c



(g) sat



(h) optdigits

Figura 6 – Comparação entre Co-MLM e Fast Co-MLM em diferentes conjuntos de dados para diferentes porcentagens de dados rotulados.

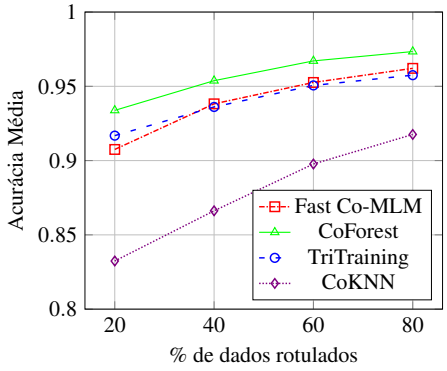
Fonte: o autor.

Considerando os resultados apresentados na Figura 7, observa-se ver que, na maioria dos casos, os métodos obtêm taxas de precisão mais altas à medida que o número de instâncias rotuladas é aumentado. Outro fato importante que deve ser observado é que as taxas de acerto mais baixas são obtidas no banco de dados *dbworld*. Isto é esperado uma vez que este conjunto de dados compreende um número reduzido de exemplos de treino com alta dimensionalidade.

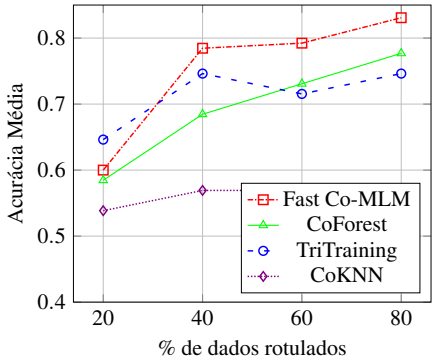
Em relação aos desempenhos de cada método, uma observação imediata é que Fast Co-MLM obteve os melhores resultados em quase todas as situações(6 das 8 bases de dados). Cabe aqui destacar essa observação, uma vez que CoForest é baseado em um modelo de conjunto de classificadores (neste experimento, foram utilizados 30 classificadores basados no *Random Forest*) e, portanto, é esperado que alcance taxas de precisão mais elevadas. Por outro lado, Fast Co-MLM, TriTraining e CoKnn usam modelos baseados em um único classificador (ou dois classificadores fracos), sendo estes MLM, J48 e K-NN, respectivamente.

4.3 Considerações finais

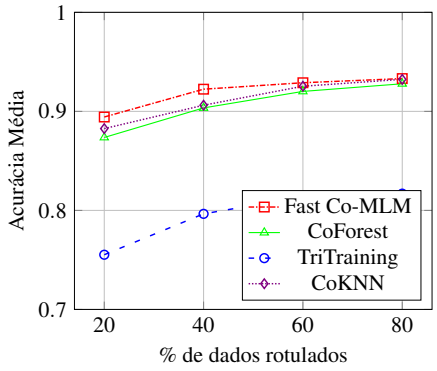
Neste capítulo, foram validadados os métodos propostos Co-MLM e Fast Co-MLM utilizando-se diversos experimentos que comprovaram a sua eficácia. Tais experimentos demonstraram a capacidade de aprender sobre dados não rotulados dos métodos mencionados, além de demonstrarem resultados promissores quando comparados a outras metodologias semelhantes.



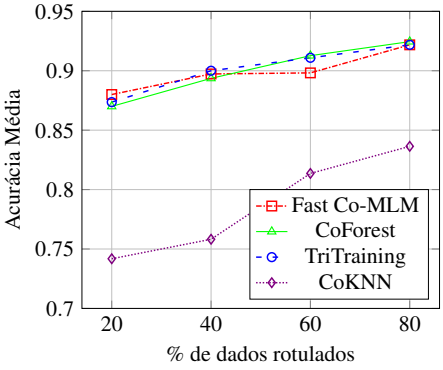
(a) car



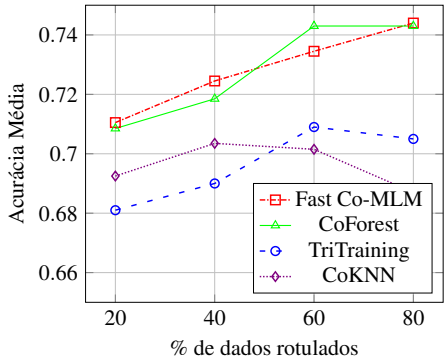
(b) dbworld



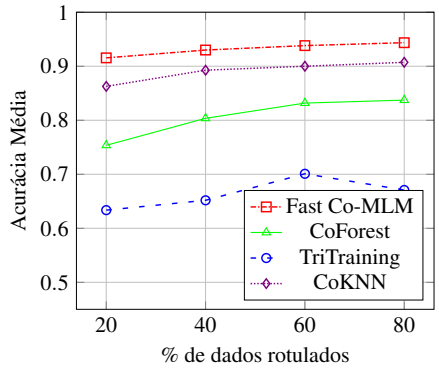
(c) uspst



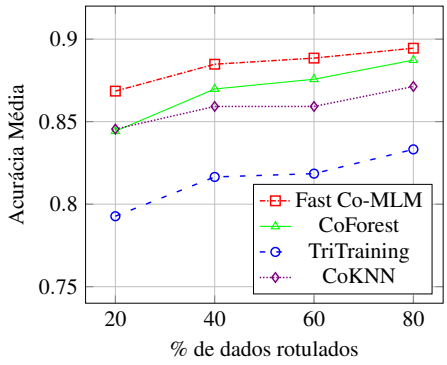
(d) g10n



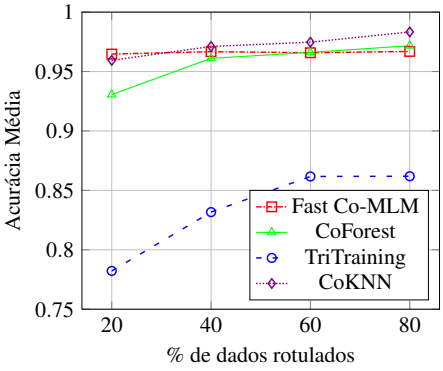
(e) german



(f) g50c



(g) sat



(h) optdigits

Figura 7 – Comparação entre Fast Co-MLM e outros métodos de aprendizagem semi-supervisionados em diferentes conjuntos de dados para diferentes quantidades de dados não rotulados.

5 CONCLUSÕES E TRABALHOS FUTUROS

Nesta seção são apresentadas as principais contribuições desta dissertação, suas limitações, bem como os futuros trabalhos. É realizada uma revisão das hipóteses iniciais, realçando os resultados alcançados e fornecendo uma conclusão do ponto de vista do autor.

O Aprendizado Semi-Supervisionado consiste em utilizar a combinação de uma pequena porção de dados rotulados em conjunto com um grande volume de dados não rotulados. Tal paradigma tem ganhando muitos adeptos nos últimos anos, uma vez que existe uma gigantesca massa de volume de dados disponíveis (muito devido à popularização dos *smarth phones*, redes sociais e afins). Como rotular dados requer um certo custo em tempo e dinheiro, uma medida atrativa é utilizar ASS para reduzir o número de exemplos rotulados necessários.

Co-Training é uma das metodologias mais bem sucedidas em ASS, em que a ideia central está em usar um pequeno conjunto de dados rotulados, construir um modelo de predição para, então, aumentar a quantidade de dados rotulados, predizendo os rótulos dos dados que não os possuem. Escolher qual o classificador base para o Co-Training e adaptá-lo é uma tarefa de extrema importância e um dos objetos de estudo deste trabalho. É importante frisar que optou-se por selecionar MLM como o classificador base, o qual tem obtido bons resultados no paradigma de Aprendizado Supervisionado, além de possuir uma terminologia simples e de fácil implementação.

Nossa hipótese inicial era de que se MLM obteve bons resultados em AS, também poderia ser repetido com sucesso para ASS. Além da escolha do classificador base, optou-se por propor uma nova regra de decisão específica para a saída do MLM, bem como uma forma (também específica para este método) de diversificar os classificadores MLM's utilizados no processo do Co-Training.

Foram propostos dois métodos semi-supervisionados batizados de Co-MLM e Fast Co-MLM. Diversos experimentos foram realizados para verificação da eficácia dos métodos propostos, bem como a eficácia da escolha de MLM como classificador base. Nestes experimentos, ficou evidente a capacidade de ambos os métodos em aprender acerca de dados não rotulados, confirmando-se as expectativas inicias.

Vale a pena ressaltar que Co-MLM possui alto custo computacional, tanto devido à própria metodologia do Co-Training, que consiste em repetir diversas vezes o mesmo procedimento, quanto na escolha do classificador base MLM, que possui um elevado custo computacional na sua fase de teste, necessitando resolver um problema de regressão não-linear. Fast Co-MLM,

no entanto, foi proposto para minimizar este problema, ao utilizar a variante NN-MLM com menor custo computacional do que o método clássico MLM, além de aplicar um procedimento recursivo para o cálculo da matriz B de coeficientes de regressão das matrizes de distâncias. Outro fator a se mencionar é que Fast Co-MLM foi comparado com Co-MLM em termos de acurácia e tempo de execução, e apesar de apresentar um significativo ganho de velocidade, não demonstrou perda da generalização do erro. Com isso, conclui-se que Fast Co-MLM serve de uma forma geral como substituto do Co-MLM, mantendo resultados similares, porém com um significativo ganho de desempenho.

Vale ressaltar que, como experimento complementar, selecionaram-se alguns dos algoritmos mais conhecidos baseados em Co-Training, como Co-Forest, TriTraining e a própria metodologia tradicional Co-Training, a fim de averiguar a competitividade dos métodos propostos. Fast Co-MLM apresentou resultados competitivos quando comparado a estes métodos, sendo por tanto uma alternativa às abordagens já encontrados na literatura. Destaca-se, no entanto, que existem muitas abordagens em ASS com resultados superiores, em alguns campos, a estes algoritmos citados. Porém, os resultados obtidos de Fast Co-MLM demonstram que a técnica merece ser objeto de estudo, podendo futuramente ser aprimorada.

Além da contribuição teórica, uma implementação foi desenvolvida em forma de *toolbox* com o intuito de disponibilizar os métodos propostos ao público em geral. Optou-se por usar a linguagem MATLAB, por ser comumente utilizada na comunidade internacional de Aprendizado de Máquina. Além da implementação dos dois métodos já referidos, a *toolbox* contém uma gama de funções utilitárias e diversas opções de personalização. No apêndice A, é possível encontrar uma documentação básica de como usar as principais funções e parâmetros disponíveis, além de contar também com um tutorial didático a fim de instruir o usuário.

Mesmo com resultados promissores, ainda existem diversas lacunas a serem exploradas. Fast Co-MLM necessita, por sua própria formulação, de métodos mais sofisticados de seleção dos pontos de referência dos classificadores MLM's. Além disso, necessita-se investigar metodologias alternativas e mais robustas de como construir as duas visões, do que apenas dividir o conjunto original de atributos aleatoriamente em dois subconjuntos. Outro problema com elevado grau de interesse está em como selecionar os melhores exemplos a serem rotulados e adicionados ao novo conjunto de treino.

Por fim, conclui-se que o presente trabalho alcançou os resultados iniciais desejados. Fornece-se a comunidade acadêmica novos métodos de Aprendizado Semi-Supervisionado em

conjunto com sua implementação computacional. Melhores estratégias de selecionar pontos de referência, construção das visões e seleção dos exemplos a serem rotulados, serão abordados em trabalhos futuros, com o intuito de aprimorar Fast Co-MLM.

REFERÊNCIAS

ABNEY, S. Bootstrapping. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the 40th Annual Meeting on Association for Computational Linguistics**. [S.l.], 2002. p. 360–367.

ADMINISTRATION, U. G. S. **Data.Gov**. 2017. Accessed: 2017-001-31. Disponível em: <https://www.data.gov>.

A.H., S. J.; F., C.; Y., M.; A., L.; G., B.; O., S. Minimal learning machine: A new distance-based method for supervised learning. In: **12th International Work Conference on Artificial Neural Networks (IWANN'2013)**. [S.l.: s.n.], 2013. p. 408–416.

ALENCAR, A. S. C.; CALDAS, W. L.; GOMES, J. P. P.; SOUZA, A. H. d.; AGUILAR, P. A. C.; RODRIGUES, C.; FRANCO, W.; CASTRO, M. F. d.; ANDRADE, R. M. C. Mlm-rank: A ranking algorithm based on the minimal learning machine. In: **2015 Brazilian Conference on Intelligent Systems (BRACIS)**. [S.l.: s.n.], 2015. p. 305–309.

ALTMAN, N. S. An introduction to kernel and nearest-neighbor nonparametric regression. **The American Statistician**, Taylor & Francis, v. 46, n. 3, p. 175–185, 1992.

ASUNCION, A.; NEWMAN, D. **UCI machine learning repository**. 2007.

BALCAN, M.-F.; BLUM, A.; YANG, K. Co-training and expansion: Towards bridging theory and practice. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2004. p. 89–96.

BENNETT, K.; DEMIRIZ, A. *et al.* Semi-supervised support vector machines. **Advances in Neural Information processing systems**, MIT; 1998, p. 368–374, 1999.

BLUM, A.; CHAWLA, S. Learning from labeled and unlabeled data using graph mincuts. 2001.

BLUM, A.; MITCHELL, T. Combining labeled and unlabeled data with co-training. In: ACM. **Proceedings of the eleventh annual conference on Computational learning theory**. [S.l.], 1998. p. 92–100.

BREFELD, U. Multi-view learning with dependent views. In: ACM. **Proceedings of the 30th Annual ACM Symposium on Applied Computing**. [S.l.], 2015. p. 865–870.

CHAPELLE, O.; ZIEN, A. Semi-supervised classification by low density separation. In: **AISTATS**. [S.l.: s.n.], 2005. p. 57–64.

DALITZ, C. Document image analysis with the gamera framework. In: ____. [S.l.]: Shaker Verlag, 2009. cap. Reject Options and Confidence Measures for kNN Classifiers, p. 16–38.

DASGUPTA, S.; LITTMAN, M. L.; MCALLESTER, D. Pac generalization bounds for co-training. **Advances in neural information processing systems**, MIT; 1998, v. 1, p. 375–382, 2002.

DISMUKE, C.; LINDROOTH, R. Ordinary least squares. **Methods and Designs for Outcomes Research**, ASHP Bethesda, v. 93, p. 93–104, 2006.

FEGER, F.; KOPRINSKA, I. Co-training using rbf nets and different feature splits. In: IEEE. **Neural Networks, 2006. IJCNN'06. International Joint Conference on**. [S.l.], 2006. p. 1878–1885.

FRIEDMAN, J.; HASTIE, T.; TIBSHIRANI, R. **The elements of statistical learning**. [S.l.]: Springer series in statistics Springer, Berlin, 2001. v. 1.

GRANDVALET, Y.; BENGIO, Y. *et al.* Semi-supervised learning by entropy minimization. In: **NIPS**. [S.l.: s.n.], 2004. v. 17, p. 529–536.

HAYES, M. Recursive least squares. **Statistical Digital Signal Processing and Modeling**, p. 541, 1996.

HO, T. K. The random subspace method for constructing decision forests. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, IEEE, v. 20, n. 8, p. 832–844, 1998.

JUNIOR, A. S.; CORONA, F.; MICHE, Y.; LENDASSE, A.; BARRETO, G. Extending the minimal learning machine for pattern classification. In: **Computational Intelligence and 11th Brazilian Congress on Computational Intelligence (BRICS-CCI CBIC), 2013 BRICS Congress on**. [S.l.: s.n.], 2013. p. 236–241.

LI, K.; ZHANG, J.; XU, H.; LUO, S.; LI, H. A semi-supervised extreme learning machine method based on co-training. **J. Comput. Inf. Syst**, v. 9, n. 1, p. 207–214, 2013.

LI, M.; ZHOU, Z.-H. Improve computer-aided diagnosis with machine learning techniques using undiagnosed samples. **Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on**, IEEE, v. 37, n. 6, p. 1088–1098, 2007.

LIU, W.; LI, Y.; TAO, D.; WANG, Y. A general framework for co-training and its applications. **Neurocomputing**, Elsevier, v. 167, p. 112–121, 2015.

MARQUARDT, D. W. An algorithm for least-squares estimation of nonlinear parameters. **Journal of the Society for Industrial and Applied Mathematics**, v. 11, n. 2, p. 431–441, 1963. Disponível em: <http://dx.doi.org/10.1137/0111030>.

MCCALL, C.; REDDY, K. K.; SHAH, M. Macro-class selection for hierarchical k-nn classification of inertial sensor data. In: . [S.l.: s.n.], 2012.

MESQUITA, D. P. P.; GOMES, J. P. P.; JR., A. H. S. A minimal learning machine for datasets with missing values. In: **Neural Information Processing: 22nd International Conference, ICONIP 2015, Istanbul, Turkey, November 9-12, 2015, Proceedings, Part I**. [S.l.: s.n.], 2015. p. 565–572.

MESQUITA, D. P. P.; GOMES, J. P. P.; JUNIOR, A. H. S. Ensemble of minimal learning machines for pattern classification. In: **Advances in Computational Intelligence: 13th International Work-Conference on Artificial Neural Networks, IWANN 2015, Palma de Mallorca, Spain, June 10-12, 2015. Proceedings, Part II**. [S.l.: s.n.], 2015. p. 142–152.

MESQUITA, D. P. P.; GOMES, J. P. P.; JUNIOR, A. H. S. Ensemble of efficient minimal learning machines for classification and regression. **Neural Processing Letters**, p. 1–16, 2017.

MITCHELL, T. M. Machine learning. **Artificial Intelligence**, 1997.

NIEWIADOMSKA-SZYNKIEWICZ EWA, M. M. Optimization schemes for wireless sensor network localization. **International Journal of Applied Mathematics and Computer Science**, University of Zielona Gora Press, v. 19, n. 2, p. 291–302, 2009. Disponível em: <http://eudml.org/doc/207936>.

NIGAM, K.; GHANI, R. Analyzing the effectiveness and applicability of co-training. In: ACM. **Proceedings of the ninth international conference on Information and knowledge management**. [S.l.], 2000. p. 86–93.

NIGAM, K.; MCCALLUM, A. K.; THRUN, S.; MITCHELL, T. Text classification from labeled and unlabeled documents using em. **Machine learning**, Springer, v. 39, n. 2-3, p. 103–134, 2000.

PARK, H.-S.; JUN, C.-H. A simple and fast algorithm for k-medoids clustering. **Expert Systems with Applications**, Elsevier, v. 36, n. 2, p. 3336–3341, 2009.

PIERCE, D.; CARDIE, C. Limitations of co-training for natural language learning from large datasets. In: **Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing**. [S.l.: s.n.], 2001. p. 1–9.

RABINER, L. R. A tutorial on hidden markov models and selected applications in speech recognition. **Proceedings of the IEEE**, IEEE, v. 77, n. 2, p. 257–286, 1989.

SIMON, H. A. Why should machines learn? In: **Machine learning**. [S.l.]: Springer, 1983. p. 25–37.

WAN, X. Co-training for cross-lingual sentiment classification. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1**. [S.l.], 2009. p. 235–243.

WANG, J.; LUO, S.-w.; ZENG, X.-h. A random subspace method for co-training. In: IEEE. **Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on**. [S.l.], 2008. p. 195–200.

XU, C.; TAO, D.; XU, C. A survey on multi-view learning. **arXiv preprint arXiv:1304.5634**, 2013.

XU, J.; HE, H.; MAN, H. Dcpe co-training for classification. **Neurocomputing**, Elsevier, v. 86, p. 75–85, 2012.

YASLAN, Y.; CATALTEPE, Z. Co-training with relevant random subspaces. **Neurocomputing**, Elsevier, v. 73, n. 10, p. 1652–1661, 2010.

YU, J.; WANG, M.; TAO, D. Semisupervised multiview distance metric learning for cartoon synthesis. **Image Processing, IEEE Transactions on**, IEEE, v. 21, n. 11, p. 4636–4648, 2012.

ZHA, Z.-J.; ZHANG, H.; WANG, M.; LUAN, H.; CHUA, T.-S. Detecting group activities with multi-camera context. **Circuits and Systems for Video Technology, IEEE Transactions on**, IEEE, v. 5, n. 5, p. 856–869, 2013.

ZHOU, Y.; GOLDMAN, S. Democratic co-learning. In: IEEE. **Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on**. [S.l.], 2004. p. 594–602.

ZHOU, Z.-H.; LI, M. Tri-training: Exploiting unlabeled data using three classifiers. **Knowledge and Data Engineering, IEEE Transactions on**, IEEE, v. 17, n. 11, p. 1529–1541, 2005.

ZHU, X. Semi-supervised learning literature survey. Citeseer, 2005.

APÊNDICE A – TOOLBOX SSL-MLM

Com o intuito de avaliar empiricamente a eficiência dos métodos propostos, Co-MLM e Fast Co-MLM, foi desenvolvida uma *toolbox* utilizando a linguagem MATLAB/OCTAVE contendo as principais funcionalidades descritas na seção 3. Nesta seção será apresentado um tutorial de como utilizar tal *toolbox*, bem como suas funções e parâmetros. A referida *toolbox* foi batizada de *Semi-Supervised Learning-Minimal Learning Machine* (SSL-MLM) sendo disponibilizada neste *link*.

A.1 Estrutura da *toolbox*

A *toolbox* pode ser divida em três grandes blocos: objeto parametrizado,funções base e funções utilitárias. O objeto parametrizado serve como forma de padronização aos usuários, oferecendo um escopo de como utilizar todos os parâmetros e funções oferecidas pela *toolbox* de forma simples e efetiva. As funções bases correspondem ao principal foco da *toolbox*, referindo-se às funções **FastCoMLM** e **CoMLM**, responsáveis pela fase de treino e teste dos métodos propostos, enquanto que as funções utilitárias trazem facilidades aos usuários da *toolbox*.

A.2 Objeto parametrizado

A *toolbox* conta com uma gama de opções disponíveis aos usuários, de forma a personalizar tanto o classificador MLM quanto os parâmetros do Co-Training. A seguir pode ser vista uma lista das opções disponibilizadas, divididas em duas subseções.

A.2.1 Opções para MLM

As opções de modelagem do algoritmo MLM são as mesmas disponibilizadas pela ferramenta computacional do autor original, uma vez que para a implementação desta *toolbox* utilizamos parte das funções fornecidas originalmente. O número de pontos de referência é o único hiper parâmetro do MLM, porém, como disponibilizado na versão original, é possível escolher o termo constante da matriz coeficientes de regressão B das matrizes de distâncias e o seu fator λ de regularização. Em adicional, permiti-se também o tipo de seleção dos pontos de referência. Fora isso, para o método Co-MLM, ainda são oferecidas duas funções de otimização para o cálculo do problema de multilateração. Assim, são listados os seguintes parâmetros:

a) Parâmetro *K*: número de pontos de referência para o MLM. Por padrão, o valor é de 10% do conjunto de dados total;
b) Parâmetro *bias*: valor constante de *B*. Por padrão o valor é igual a 0, ou seja, sem constante;
c) Parâmetro λ: valor de regularização. Por padrão o valor do parâmetro é 0, ou seja, sem regularização;
d) Parâmetro *selectMethod*: Método de seleção dos pontos de referência. Atribua 1, para usar o método de seleção aleatório, e 0, para o método *k-medoids*. Por padrão usa-se 0;
e) Parâmetro *type*: tipo de método de otimização: usa-se *fsolve* para aplicar Levenberg-Marquardt e *lsq* para aplicar o método de aproximação jacobiano. Por padrão o método LM é preferido;

A.2.2 Opções para Co-Training

O *framework* Co-Training opera com um número *Q* de iterações preestabelecido pelo usuário. Outra opção relevante é o número de exemplos a serem rotulados, podendo este ser especificado para ambos os métodos Fast Co-MLM e Co-MLM. Em adicional, a construção das visões pode ser feita repetidas vezes utilizando-se o conjunto de validação para selecionar o melhor conjunto de visões possível. Nesse caso, um número *v* de iterações será estabelecido, representando o número de vezes em que o modelo construirá essas visões aleatoriamente. Apesar de adotar-se o método de separação aleatória dos atributos, é possível, através de um vetor, ordenar a posição dos atributos da forma que o usuário desejar, a única restrição é o fato das visões terem tamanhos iguais. Como Co-Training recebe tanto dados rotulados quando dados não rotulados, o parâmetro **paramLabel** definirá a porcentagem dos dados contendo rótulos. Assim, são listados os seguintes parâmetros:

a) Parâmetro *Q*: número de iterações na fase Co-Training. Por padrão o valor é de 5 iterações;
b) Parâmetro *n*: número de exemplos por classe a serem rotulados por cada iteração. Por padrão o valor é 1;
c) Parâmetro *v*: número de vezes em que o algoritmo dividirá aleatoriamente o conjunto original de atributos em dois subconjuntos para formar as visões. Por padrão o valor é 1;

d) Parâmetro *permutationVector*: vetor contendo a ordem dos atributos para cada visão, de tal forma que a primeira metade corresponderá a primeira visão, e o restante à segunda. Caso o vetor esteja vazio, a divisão será aleatória. Além disso caso este vetor não seja nulo, o parâmetro *v* obviamente será definido como 1;

e) Parâmetro *paramLabel*: define a porcentagem de dados rotulados, podendo ser um valor unitário ou um vetor contendo a quantidade de dados para cada experimento. Por padrão a porcentagem é de 60%;

A.3 Funções base

Nesta seção será descrito o uso das principais funções desenvolvidas nesta *toolbox*, responsáveis por construir e avaliar os modelos propostos. Inicialmente tem-se a função **FastCoMLM**, que recebe um objeto parametrizado e retorna um modelo avaliado. Nesta função ambas as etapas, treino e teste são efetuadas e o modelo retornado contém todas as informações necessárias, como taxa de acerto, matriz de confusão, tempo de execução dentre outras. Caso o usuário deseje separar as fases de treino e teste, a função **TrainFastCoMLM** pode ser utilizada para construir o modelo, enquanto que a função **testnnMLM** para avaliá-lo. É importante notar que **testnnMLM** é também a função que avalia um modelo *NN-MLM* qualquer, não necessitando de nenhuma função extra para avaliar **FastCoMLM**, uma vez que **Co-Training** em nada altera o método NN-MLM. Apenas aumenta o número de dados rotulados disponíveis. As funções **CoMLM**,**TrainCoMLM** e **testMLM** cumprem respectivamente as mesmas atribuições das funções **FastCoMLM**,**TrainFastCoMLM** e **testnnMLM**, com a mudança de que o fazem para Co-MLM. No código 1 observa-se um exemplo de chamadas destas funções.

Código-fonte 1 – Chamada das principais funções disponibilizadas.

```
1 %Exemplo de chamada de treino e teste%
2 [MODEL]=FastCoMLM(objParam);
3 %Exemplo de chamada somente do treino%
4 [MODEL]=TrainFastCoMLM(objParam);
5 %Fun   o teste do metodo NN-MLM que tambem serve para o
     FastCoMLM
6 [~,Yh]=test_nnMLM(Beta,testData,refPoints);
```

A.4 Funções utilitárias

Algumas funções, que normalmente não serão vistas pelo usuários, estarão disponíveis, como por exemplo, as funções de treino e teste dos algoritmos MLM e NN-MLM, bem como a função que calcula o método de mínimos quadrados recursivamente. Porém, algumas outras funções, como **constructData**, que serve para adequar o conjunto de dados aos termos necessários (ver subseção A.6), serão de extrema utilidade. Além desta função, **remap**, que calcula a distância entre uma matriz e os pontos de referência, e **Sencoding**, que transforma um vetor unidimensional no sistema de codificação 1-por-S, estão disponíveis na *toolbox*. No código 2 observa-se um exemplo de chamadas destas funções.

Código-fonte 2 – Chamada de algumas funções utilitárias.

```
1 %Exemplo de chamada da funcao recursiva do minimos
     quadrados%
2 [ B,Pk ] = RLS_Online(Bo,Po,H,T);
3 %Exemplo de chamada da funcao remap%
4 [ Dx,Dy ] = remap(refPoints,data);
5 %exemplo de chamda da fun   o constructData.
6 [ dataset ] = constructData( data, iterations, rate_train,
     rate_test);
```

A.5 Exemplo de uso

Nesta seção, será detalhado o uso da *toolbox* através do código fonte 3. Como pode ser observado, um objeto nomeado de **objParam** foi criado a fim de coletar todos os parâmetros necessários (linhas 1 a 15). A chamada da função **FastCoMLM** (linha 18) passa os parâmetros através do objeto **objParam** e retorna o modelo final produzido no objeto *MODEL*, que conterá todas as informações pertinentes, como taxa de acerto, parâmetros selecionados na validação, matriz de confusão, tempo de execução, etc. Nesta mesma função, ocorrem as etapas de treinamento, validação e teste. Para tal, o conjunto de dados importado (linha 8) deve atender às seguintes exigências que serão vistas na subseção A.6. As fases de treino (linhas 20 e 21) e teste (linhas 23 a 38) podem ser executadas separadamente, neste caso a mesma função

testnnMLM, usada para testar um modelo NN-MLM, pode ser aplicada. De forma similar a vista anteriormente todos os parâmetros devem ser fornecidos usando o objeto **objParam**.

Código-fonte 3 – Tutorial de uso.

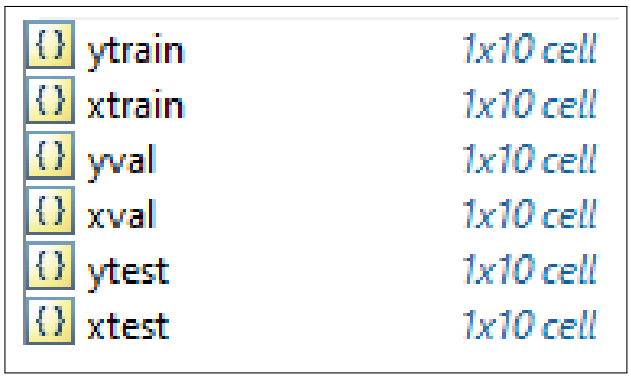
```
%Exemplo de Parametros B sicos%
objParam.numberRepetitions=10;
objParam.numberRepetitionsForCotraining=5;
objParam.normalization=0;
objParam.selectMethod=0;
objParam.paramLabel=[6];
objParam.paramPoints=[0.5:0.1:1];
objParam.dataSet=load( Smartphone_Dataset/descendingrunning
   .mat );
objParam.dataSet=objParam.dataSet.dataset;
objParam.dataName= default ;
objParam.type= fsolve ;
objParam.bias=0;
objParam.lambda=0;
objParam.validation=1;
objParam.permutationVector=1:3000;

%Exemplo de chamada de treinamento e teste
[FullMODEL]=FastCoMLM(objParam);

%Exemplo de chamada somento do treino%
[MODEL]=TrainFastCoMLM(objParam);

%Exemplo de chamada somento do teste%

%Matriz de coeficientes do primeiro modelo gerado
finalBeta=MODEL{1}.finalBeta;
finalrefPoints=MODEL{1}.finalrefPoints;
```

```
testData.x=objParam.dataSet.xtest{1};
testData.y=S_encoding(objParam.dataSet.ytest{1});

%Funcao teste do m todo NN-MLM
[~,Yh]=test_nnMLM(finalBeta,testData,finalrefPoints);

%Comparacao do modelo B*X vs B  *X
for j = 1: length(testData.y),
    [~, index(j)] = max(Yh(j,:));
    [~, target(j)] = max(testData.y(j, :));
end
%Taxa de acerto
rate = length(find(index == target))/length(testData.y)
```

A.6 Base de dados de entrada

A base de dados utilizadas na *toolbox* tem uma estrutura a ser seguida, e para tal, é fornecida uma função que adéque um conjunto de dados as especificações. A função **constructData** retorna um conjunto de dados contendo 3 sub conjuntos: treino, teste e validação. Além disso, cada sub conjunto conterá ao menos uma instância de cada classe, podendo opcionalmente construir aleatoriamente quantos *folds* o usuário quiser, ou seja, caso o usuário escolha obter 10 *folds*, a função retornará um objeto contendo 10 conjuntos de dados de treino, teste e validação, cujas instâncias foram escolhidas de forma aleatória. Para tal, a função **constructData** transforma uma matriz *lxc* (l linhas por c colunas) de dados em que as primeiras $c-1$ colunas representam os atributos, e a última coluna representa a variável resposta, no conjunto de dados com as especificações/formatações necessárias. A Figura 8 apresenta um objeto com 10 *folds* aleatoriamente divididos, com cada um deles contendo os conjuntos de treino, teste, e validação. Outra importante observação é que a base de dados não rotulada U deve estar em conjunto com a base de dados rotulada L. Neste caso, considera-se apenas os l primeiros exemplos como sendo rotulados. Para simplificar o uso da *toolbox*, os exemplos não rotulados terão seu valor resposta igual a zero.

Figura 8 – Exemplo de objeto retornado ao usar-se a função *constructData*



Fonte: o autor

A.7 Considerações finais

Neste capítulo foi apresentada a *toolbox Semi-Supervised Learning-Minimal Learning Machine*, contendo uma implementação MATLAB/OCTAVE de Co-MLM e Fast Co-MLM. Somado a isso, forneceram-se as instruções necessárias para usá-la, apresentando-se exemplos e tutoriais.