Learning and Nonlinear Models - Journal of the Brazilian Computational Intelligence Society, Vol. 11, Iss. 2, pp. 56–73, 2013.

APRENDIZADO POR TRANSFERÊNCIA PARA APLICAÇÕES ORIENTADAS A USUÁRIO: UMA EXPERIÊNCIA EM LÍNGUA DE SINAIS

Lucas F. Brunialti, Sarajane M. Peres, Clodoaldo A. M. Lima
Escola de Artes, Ciências e Humanidades - Universidade de São Paulo
{lucas.brunialti, sarajane, c.lima}@usp.br

Clodis Boscarioli
Centro de Ciências Exatas e Tecnológicas - Universidade Estadual do Oeste do Paraná
{clodis.boscarioli}@unioeste.br

Resumo – Um grande desafio atual é a interpretação automatizada de gestos relacionados à comunicação em Línguas de Sinais. Nesse contexto, a execução dos gestos assume um caráter de grande variabilidade, o que o diferencia daqueles onde os gestos de interesse são simples, como no caso de automação de "casas inteligentes" ou jogos de computadores. Devido a essa complexidade, para indução de modelos de reconhecimento de gestos com grande variabilidade de execução, são necessários conjuntos de dados rotulados provenientes de diferentes pessoas executando os gestos. É fato que este tipo de aquisição e rotulação de dados é uma tarefa difícil, então, este trabalho apresenta um estudo sobre uma estratégia para minimizar este problema. Com o uso dos algoritmos *Multilayer Perceptron* (aprendizado supervisionado) e *TrAdaBoost* (aprendizado por transferência), este artigo discute uma abordagem para construção de aplicações orientadas à usuário, melhorando o reconhecimento de gestos provenientes de novos usuários.

Palavras-chave – Análise de Movimentos, Análise de Gestos, Aprendizado por Transferência, *TRAdaBoost*, *Multilayer Perceptron*, Língua de Sinais.

Abstract – Nowadays, a major challenge is the interpretation of Sign Languages gestures in an automated way. In this context, the gestures assume great variability, which differs Sign Language applications from those where the gestures of interest are simple, such as automation of "smart houses" or computer games. Due to such complexity, in order to build inductive gesture recognition models with a great variability of execution, labeled datasets from different users are needed. Actually, this kind of data acquisition and labeling is a laborious task, thus, this paper presents a study about a strategy to minimize this problem. Making use of the algorithms Multilayer Perceptron (supervised learning) and TrAdaBoost (transfer learning), this paper discusses an approach to build user-oriented applications, improving the recognition of gestures from new users.

Keywords – Movement Analysis, Gesture Analysis, Transfer Learning, TrAdaboost, Multilayer Perceptron, Sign Language.

1. INTRODUÇÃO

A área de pesquisa em análise de gestos tem recebido uma forte atenção de diferentes setores da sociedade. Esse aumento de interesse se justifica pelas recentes iniciativas de desenvolvimento, e fabricação em larga escala para venda comercial, de dispositivos de baixo custo para sensoriamento de movimentos. Muitas pesquisas na área de análise de gestos têm focado no desenvolvimento de novos métodos para interação humano computador, suportando a criação de aplicações das mais diversas naturezas. Jogos, tecnologias assistivas, controle de equipamentos via gestos e softwares de monitoramento de ambientes são alguns exemplos das áreas de aplicação mais conhecidas pela sociedade. Embora existam muitas aplicações que já usam esse tipo de dispositivo de sensoriamento acompanhado de softwares que interpretam gestos, tem-se percebido que as aplicações mais populares, em geral, tratam de um escopo limitado de gestos, e trazem algumas restrições como posicionamento em frente aos sensores ou velocidade de execução do gesto. Algumas revisões sobre assuntos relacionados à análise de gestos foram desenvolvidas por Pickering [1], Mitra e Acharya [2], Moni e Ali [3], Sowa [4], Liu e Kavakli [5] e Madeo e Peres [6].

Especificamente para o caso de interpretação de gestos das Línguas de Sinais, existe pouco desenvolvimento já disponível na indústria. Aplicações existentes são, geralmente, suportadas por equipamentos com muitos sensores (luvas, acelerômetros e giroscópios), que precisam ser "usados" pelo usuário. Existem alguns fatores que contribuem para que aplicações baseadas em dispositivos de sensoriamento menos invasivos tenham um escopo de atuação mais limitado na área de Língua de Sinais, entre eles estão: (a) o nível de detalhe que diferencia um sinal de outro, que muitas vezes não são captados com a devida granularidade (ou resolução) por dispositivos menos invasivos como *webcams* e dispositivos como o dispositivo *Kinect*™[1]; (b) a grande diversidade de tipos de movimentos e configurações corporais diferentes que uma língua de sinais exige[2]; (c) a forma como tais movimentos

[1] *Kinect* é um dispositivo para sensoriamento de movimentos, desenvolvido pela *Microsoft*, originalmente para uso acoplado ao video game Xbox 360. O sensoriamento realizado por ele é baseado, basicamente, em uma câmera RGB e um sensor de profundidade.

[2] Para conhecer os movimentos e configurações corporais que uma língua de sinais engloba, veja a Enciclopédia da Língua de Sinais Brasileira [7].

e configurações se combinam para compor os sinas que são usados na língua; (d) a própria forma de executar tais movimentos e configurações que variam de pessoa para pessoa, criando situações de formulação de sinais incompletos e imprecisos quando comparados a modelos de execução esperada.

A abordagem discutida no presente artigo está relacionada ao estudo de alternativas que, usando dispositivos do tipo *Kinect*™, possam minimizar o efeito do fator (d), tratando o problema de análise de movimentos sob um ponto de vista de uma técnica que tem boa capacidade de generalização, tratando a questão de informação imprecisa e incompleta; e usando uma estratégia que atenda o problema de variabilidade de comportamento de usuário na execução dos movimentos. A fim de fornecer modelos computacionais de análise de movimentos que tenham boa capacidade de generalização, trabalhos anteriores têm aplicado Redes Neurais Artificiais e Teoria de Conjuntos *Fuzzy* ([8], [9] e [10]) para análise de movimentos do contexto de Línguas de Sinais. Embora tenha-se alcançado resultados interessantes nesses trabalhos, pecebe-se que os erros dos modelos são principalmente referentes a dados de novos usuários. Para atender a questão de incorporar a flexibilidade de adaptação de aplicações à diferentes e novos usuários, no estudo descrito neste artigo, foram executados testes com "re" treinamento de modelos de indução clássicos, usando a Rede Neural *Multilayer Perceptron*, e também usando esta mesma Rede Neural junto da estratégia de Aprendizado por Transferência. Desta forma, intencionou-se viabilizar a construção de modelos adaptados a novos usuários, sem perda da generalização já adquirida - caracterizando a disponibilização de uma estratégia de resolução para um problema multitarefa.

A fim de esclarecer como tal abordagem pode ser inserida no contexto de uma aplicação de soletração manual para Língua de Sinais (Figura 1), considere o Jogo da Forca Multimídia em Libras (Língua Brasileira de Sinais), de Madeo *et al*. [9]. Na implementação original, a aplicação é disponibilizada com um modelo de reconhecimento de padrões já previamente treinado. Supondo a estratégia apresentada no presente artigo, a aplicação poderia ser disponibilizada com um "módulo de treinamento" do motor da máquina de reconhecimento de padrões. Na Figura 1 é apresentado um esquema em que esse módulo é baseado na estratégia de Aprendizado por Transferência, e faz uso de uma grande base de dados com movimentos de vários usuários previamente representados e rotulados já disponível na aplicação, mais uma pequena base de dados, também rotulada, fornecida por um novo usuário.

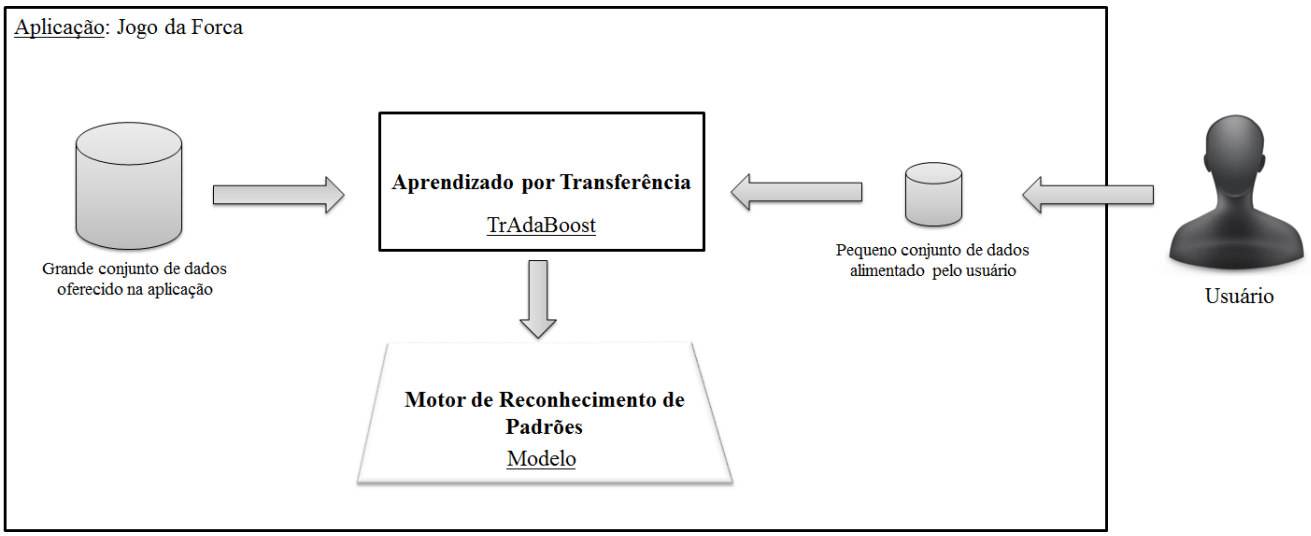


Figura 1: Exemplo de contexto de aplicação da abordagem de Aprendizado por Transferência suportando a adaptação de um motor de reconhecimento de padrões para tratar novos usuários adequadamente. O módulo de Aprendizado por Transferência realiza o treinamento do modelo que compõe o motor de reconhecimento de padrões da aplicação. Alternativamente, outras técnicas e estratégias de construção do modelo de indução podem ser aplicadas. No presente trabalho, uma Rede Neural *Multilayer Perceptron* também foi testada para implementar tal módulo.

Geralmente, para que se tenha bons modelos de reconhecimento de padrões, é necessário prover uma massa de dados grande e rotulada (para o caso do uso do aprendizado supervisionado) e, esperar que o usuário forneça tal base num contexto de aplicação orientada ao usuário, não é uma opção. Portanto, o destaque da abordagem discutida neste artigo é que ela implementa uma forma de tornar orientada ao usuário uma aplicação originalmente independente do usuário, usando (poucos) dados do usuário. Nesta abordagem, mantem-se o desempenho do motor da máquina de reconhecimento de padrões em relação ao conjunto de dados já embarcado na aplicação.

Objetivando expor os estudos realizados em Aprendizado por Transferência para implementação e análise da viabilidade da abordagem supracitada, este artigo é organizado da seguinte forma: a Seção 2 discute brevemente algumas iniciativas correlatas ao uso de Aprendizado por Transferência e análise de movimentos; a teoria básica sobre Aprendizado por Transferência é apresenta na Seção 3, juntamente com um resumo sobre a Rede Neural *Multilayer Perceptron*, a qual é usada sozinha e também como componente do Aprendizado por Transferência; a experimentação suportando a orientação ao usuário em aplicações de análise de movimentos é apresentada e discutida na Seção 4; finalmente algumas considerações são apresentadas na última seção, a qual é seguida pelas referências bibliográficas.

2. TRABALHOS CORRELATOS

As técnicas de Aprendizado por Transferência têm sido aplicadas em muitas áreas. Pan e Yang [11] apresentam uma revisão tanto em relação à teoria correlata a Aprendizado por Transferência, quanto em relação às suas possibilidades de aplicação. Nesse artigo, os autores citam trabalhos que aplicaram Aprendizado por Transferência em problemas de classificação de texto, análise de emoções, classificação de imagens, classificação de e-mails (em *spam* ou não), localização de redes sem fio, tradução e revisão de opiniões sobre produtos. Ainda são listados alguns conjuntos de dados públicos, apropriados para realização de testes de *benchmarking*, e uma *toolbox* que traz algoritmos já implementados para realização de experimentos. Embora muitos trabalhos tenham sido realizados no contexto da aplicação de Aprendizado por Transferência, a área de Reconhecimento de Gestos tem sido pouco explorada por essa estratégia. Nessa seção são apresentados alguns trabalhos correlatos ao Aprendizado por Transferência aplicado ao reconhecimento de gestos ou a áreas afins.

No problema de reconhecimento de ações, FarajiDavar *et al.* [12] fazem uso de Aprendizado por Transferência do tipo Transdutivo para o reconhecimento das ações de saque, golpe e não-golpe em um jogo de tênis. O objetivo foi melhorar o desempenho de modelos de classficação gerados a partir de um conjunto de dados antigo, através do uso de novos dados não-rotulados. O método de classificação das ações usado nesse trabalho é *Kernelised Linear Discriminant Analysis* combinado aos métodos de Aprendizado por Tranferência por reponderação e por translação e escalonamento de características (ambos brevemente descritos no referido artigo). Os experimentos foram conduzidos usando vídeos de jogos de tênis simples (gravado no sistema PAL) e em dupla (gravados no sistema NTSC), caracterizando cenas em domínios diferentes e também qualidade de vídeos diferentes. O Aprendizado por Transferência se deu de duas formas: treinamento com os dados de jogo de tênis simples e testes com os dados do jogo de tênis em dupla; e o contrário. Os melhores resultados foram obtidos na primeira situação, usando a estratégia de translação e escalonamento de características. A taxa média máxima de classificações corretas chegou a $78,14\%$.

Ainda sobre reconhecimento de ações, Lopes *et al.* [13] fazem uso da estratégia de Aprendizado por Transferência para reconhecer ações presentes em vídeos do conjunto de dados Hollywood[3], a partir do conhecimento adquirido em outro conjunto de dados de imagens que possuem as mesmas ações (Caltech256[4]). Os autores estavam particularmente interessados em responder duas questões: a) se era possível **transferir** o conhecimento aprendido em uma base de dados para resolver o problema de classificação na outra base de dados; b) se o conhecimento adquirido em uma base poderia ser usado para **melhorar** o reconhecimento de padrões na outra base. As respostas para ambas as perguntas foram positivas. As ações que foram reconhecidas nessa iniciativa foram: atender o telefone, dirigir um carro, comer, lutar, sair do carro, apertar as mãos, abraçar, beijar, correr, sentar, deitar e levantar. O trabalho apresenta uma análise sobre como a transferência de conhecimento se deu em cada uma das ações, relatando em quais casos houve benefício na transferência e em quais não houve. Classificadores construídos com *Support Vector Machine* foram usados nos experimentos.

Em [14], é proposto um novo método baseado na estratégia de Aprendizado por Transferência, em Redes Neurais Artificiais e em Campos Condicionais Aleatórios, para reconhecimento de padrões no contexto de gestos feitos com os olhos e gestos feitos com a cabeça. A ideia apresentada pelos autores é criar um modelo que analisa sequências de *frames* de vídeo de forma não supervisionada, capturando informações referentes as dependências entre os *frames*; essa tarefa é considerada como uma tarefa auxiliar. Em seguida, o conhecimento adquirido nesta tarefa auxiliar é usado para refinar o modelo de reconhecimento dos gestos, considerado como a tarefa principal. A arquitetura proposta é considerada um modelo de aprendizado semi-supervisionado. A abordagem apresentada no referido trabalho foi testada em três conjuntos de dados, foi comparada com implementações de técnicas clássicas (sem aprendizado por transferência) e obteve resultados superiores. Os resultados são mostrados em termos de Curvas ROC e da análise da área sob a curva.

Dentre os trabalhos encontrados, a iniciativa apresentada por Farhadi, Forsyth e White [15] é a que mais se aproxima da estratégia estudada no presente artigo. Essa estratégia é aplicada no contexto de reconhecimento de palavras da Língua de Sinais Americana e faz uso da estratégia de Aprendizado por Transferência para aprender as palavras transferindo conhecimento de um conjunto de dados feito por um avatar $3D$ para um conjunto de dados feito por um humano. O conjunto de dados obtido a partir do uso de avatares é considerado um conjunto modelo, e por ser "sintético", é gerado por procedimentos de baixo custo (sem necessidade de esforço humano para captação de dados) e, consequentemente, apresenta a vantagem de possibilitar a construção de conjuntos de dados grandes, ideal para implementação de modelos indutivos. O modelo gerado pelos autores trabalha com um conjunto finito de 50 palavras, e usa uma descrição de *frames* baseada em informações de posição, deslocamento, velocidade e orientação das mãos e da cabeça. Por ser um problema localizado (com o objetivo específico de reconhecer determinadas palavras), uma abordagem baseada em similaridade de modelo e de dado a ser reconhecido é aplicada. Os experimentos foram realizados considerando imagens frontais de avatares, imagens frontais de humanos e imagens de humanos levemente virados para a direita ($3/4$ *view*, como especificado no artigo). Os resultados obtidos em termos de porcentagem de classificações corretas foram: 64.17% para o caso das imagens frontais e 62.5% para o caso das imagens com $3/4$ de visão.

Além dos trabalhos citados, vale ainda mencionar a iniciativa referente à competição de Reconhecimento de Gestos *ChaLearn Gesture Challenge* [16]. Os organizadores desta competição disponibilizaram um conjunto de dados[5] captado pelo dispositivo *Kinect*™da *Microsoft*, com o intuito de formentar o estudo de Aprendizado por Transferência nessa área.

[3]Conjunto de dados disponível em `http://www.di.ens.fr/~laptev/actions/hollywood2/`.
[4]Conjunto de dados disponível em `http://authors.library.caltech.edu/7694`.
[5]Conjunto de dados disponível em `http://gesture.chalearn.org/data`.

3. APRENDIZADO POR TRANFERÊNCIA

Aprendizado por Transferência, estudado na área de Aprendizado de Máquina, é uma técnica inspirada em uma estratégia humana para resolução de novos problemas, que tem chamado a atenção de pesquisadores desde meados da década de 90 [11]. Ainda segundo esse mesmo artigo, em linhas gerais, nessa estratégia faz-se uso de um conhecimento adquirido em algum domínio ou contexto, adaptando-o para resolver um problema não necessariamente localizado no mesmo domínio ou contexto, porém que apresenta uma natureza similar, em algum aspecto, ao primeiro.

Na Figura 2 é mostrado um esquema gráfico que ilustra a estratégia de Aprendizado por Transferência comparando-a com a estratégia tradicionalmente usada em Aprendizado de Máquina, e motivando-a em relação ao comportamento humano: pessoas são capazes de adquirir conhecimento em diferentes domínios e também de adaptar e aplicar o conhecimento adquirido em um domínio para resolver um problema pertencente a um domínio diferente.

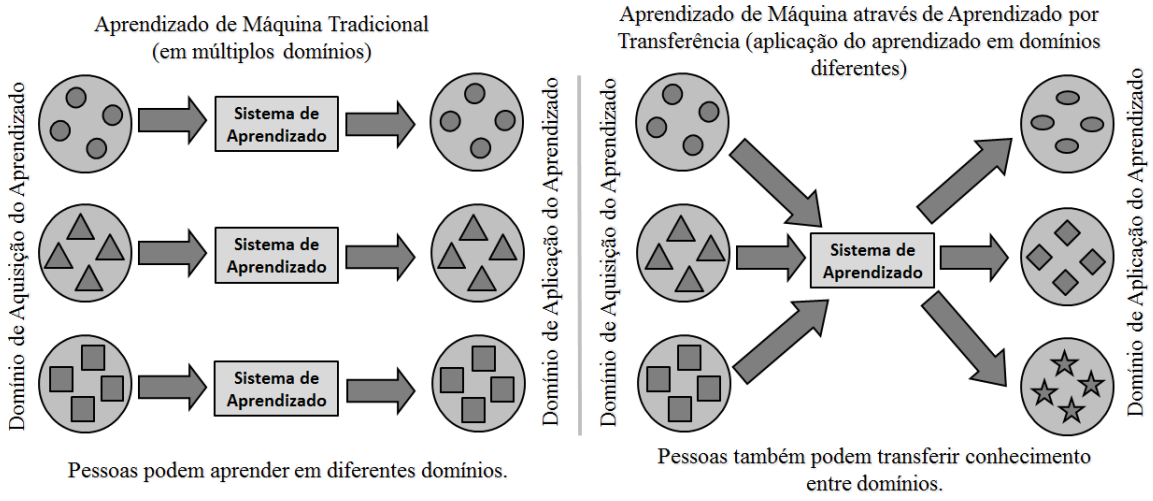


Figura 2: Diferença entre Aprendizado de Máquina Tradicional e usando Aprendizado por Transferência (adaptado de [11]).

O presente trabalho está contextualizado na análise de dados implementada por meio da resolução de uma tarefa de predição, em que se define:

- um domínio D composto por um espaço de características χ e uma distribuição de probabilidade $P(X)$, em que $X = \{x_1, x_2, ..., x_n\} \in \chi$;
- um espaço de rótulos Υ;
- uma tarefa T representada por função de predição $f(.)$, a qual é obtida a partir do aprendizado do mapeamento entre $P(X)$ e $Y = \{y_1, y_2, ..., y_n\} \in \Upsilon$; em que cada y_i corresponde ao rótulo do i-ésimo exemplo de $P(X)$.

Assim, neste contexto, a estratégia de Aprendizado por Transferência considera:

- um domínio de origem $D_o = \{\chi_o, P_o(X)\}$, em que $X_o = \{x_1, x_2, ..., x_{no}\} \in \chi_o$;
- uma tarefa de origem $T_o = \{\Upsilon_o, f_o(.)\}$;
- um domínio de destino $D_d = \{\chi_d, P_d(X)\}$, em que $X_d = \{x_1, x_2, ..., x_{nd}\} \in \chi_d$;
- uma tarefa de destino $T_d = \{\Upsilon_d, f_d(.)\}$

Então, de forma geral, a estratégia de Aprendizado por Transferência pode ser formalmente definida da seguinte forma [11]: dado um domínio de origem D_o e uma tarefa de origem T_o, um domínio de destino D_d e uma tarefa de destino T_d, o Aprendizado por Transferência suporta a função de predição do destino $f_d(.)$ em D_d usando o conhecimento *a priori* obtido em D_o e representado por T_o, sendo que $D_o \neq D_d$ e $T_o \neq T_d$.

De forma mais específica, Aprendizado por Transferência pode assumir diferentes facetas, a depender das características dos contextos onde ele é aplicado. O restante desta seção é dedicado a apresentar os diferentes tipos de Aprendizado por Transferência, dando uma atenção especial ao Aprendizado por Transferência Indutivo baseado na abordagem que transfere conhecimento de instância, usado na experimentação apresentada neste artigo. Também, como forma de deixar este artigo auto-contido, o algoritmo que implementa uma Rede Neural *Multilayer Perceptron* (usada nos experimentos) e aquele que implementa o Aprendizado por Transferência também são discutidos aqui.

3.1 CATEGORIZAÇÃO DE APRENDIZADO POR TRANSFERÊNCIA

Segundo a taxonomia apresentada por Pan e Yang [11], estratégias de Aprendizado por Transferência podem ser categorizadas em três grandes classes[6]: Aprendizado Indutivo, Aprendizado Transdutivo e Aprendizado Não-Supervisionado. E, ainda segundo o mesmo autor, qual conhecimento a ser transferido (o que transferir) pode estar baseado em quatro opções[7]: em instâncias, em características, em parâmetros, em conhecimento relacional.

3.1.1 ESTRATÉGIAS

O Aprendizado por Transferência Indutivo é caracterizado por trabalhar livremente em domínios de origem D_o e de destino D_d iguais ou diferentes, porém a tarefa de destino T_d é diferente da tarefa de origem T_o. Nesta categoria de aprendizado, quando dados rotulados de ambos domínios estão disponíveis para **induzir** o modelo preditivo para a tarefa de destino T_d, tem-se um aprendizado similar ao que ocorre na área de Aprendizado Multitarefa. No caso dos dados rotulados estarem disponíveis apenas no domínio de destino D_d, tem-se um aprendizado de mesmo tipo daquele implementado na área de Auto-Aprendizado, para **induzir** o modelo de predição para a tarefa de destino T_d. O Aprendizado por Transferência Indutivo é aplicado no presente trabalho, de forma que o Experimento 6 descrito na Seção 4 constitue-se como exemplo desta estratégia.

Também no Aprendizado pro Transferência Transdutivo tem-se duas situações a considerar. A primeira em que o espaço de características no domínio de origem D_o e no domínio de destino D_d são diferentes ($\chi_o \neq \chi_d$); a segunda em que ambos espaços de características são os mesmos, porém as distribuições de probabilidade são diferentes ($P(X_o) \neq P(X_d)$). No caso do Aprendizado por Transferência Transdutivo, o conhecimento obtido no domínio de origem D_o e na tarefa de origem T_o é usado para **melhorar** o aprendizado da função de predição para a tarefa de destino T_d, usando ainda alguns dados não-rotulados disponíveis no domínio de destino D_d. O segundo conjunto de experimentos apresentado em [13] implementa esta estratégia.

A terceira categoria é a de Aprendizado por Transferência Não-Supervisionado. Esta categoria tem definição semelhante à da categoria de Aprendizado por Transferência Indutivo, contudo, não assume dados rotulados em nenhum dos dois domínios (D_o ou D_d), e é adequada para tarefas de agrupamento. Um experimento para resolução da tarefa de agrupamento de imagens usando essa estratégia é descrito em [18]. O mesmo artigo traz um conjunto de trabalhos correlatos no quais experimentos em diferentes contextos são apresentados.

3.1.2 O QUE TRANSFERIR

Quando o Aprendizado por Transferência está baseado na transferência do conhecimento presente nas instâncias do problema (X_o ou X_d), partes do conjunto de dados do domínio de origem D_o podem ser interessantes para reúso na indução do modelo de predição para o domínio de destino D_d. Os algoritmos que implementam esta abordagem possuem estratégias específicas para determinar, durante o aprendizado, quais dados são mais interessantes e para atribuir pesos a eles de forma a influenciarem mais, ou menos, no modelo que está sendo induzido.

Transferir conhecimento referente às características significa usar algum conhecimento aprendido sobre a representação das características que descrevem o domínio de origem D_o para aprender uma representação de características interessante para ser usada como descrição dos dados no domínio de destino D_d. Os algoritmos de aprendizado nesta abordagem minimizam o erro obtido no modelo induzido para o domínio de destino D_d, usando uma codificação de representação aprendida no domínio de origem D_o.

A abordagem de transferência baseada em parâmetros assume que o domínio de origem D_o e de destino D_d compartilham algum subconjunto de parâmetros ou uma distribuição *a priori* para hiperparâmetros dos respectivos modelos de indução. Assumindo isso, o papel dos algoritmos de Aprendizado por Transferência é melhorar o desempenho dos modelos induzidos para o domínio de destino D_d fazendo uso também dos dados do domínio de origem D_o para descobrir e aplicar a informação comum de ambos domínios.

Por fim, a transferência de conhecimento relacional é uma abordagem que assume que existem relações similares entre os dados em cada domínio. Esta abordagem faz sentido em problemas onde os dados não são independentemente e identicamente distribuídos (não são i.i.d), e podem ser representados em estruturas com relações com algum significado. Desta forma, uma vez aprendidas as relações em um domínio de origem (D_o), o conhecimento (relacional) pode ser usado no domínio de destino (D_d) como forma de otimizar o modelo a ser induzido. Em [11], os autores trazem uma lista extensa de exemplos de aplicação de cada situação aqui descrita.

3.2 *MULTILAYER PERCEPTRON*

A arquitetura de Rede Neural *Multilayer Perceptron* é aplicada no presente artigo em dois momentos: para implementar uma estratégia de aprendizado simples; para compor os modelos de indução usados na estratégia de Aprendizado por Transferência implementados pelo algoritmo *TRAdaBoost*. A Rede Neural *Multilayer Perceptron* é capaz de criar superfícies de decisão não-lineares com o uso da retropropagação do erro, assim, é apresentado nesta seção o algoritmo (Algoritmo 1) de treinamento de uma *Multilayer Perceptron* usando *Backpropagation* com gradiente descendente, *momentum* e taxa de aprendizado adaptativa.

[6] Uma taxonomia diferente é apresentada em por Torrey e Shavlik [17].

[7] Veja em [11] uma relação de iniciativas que combinam as categorias de Aprendizado por Transferência com as opções de conhecimento a ser transferido.

Algorithm 1 *Backpropagation* com gradiente descendente, *momentum* e taxa de aprendizado adaptativa (adaptado de [19])

Entrada:
Um conjunto de dados para treinamento $\mathcal{D} = \langle x, y \rangle$ em que $x \in \chi$ é um vetor de entradas e $y \in \Upsilon$ é o vetor de saídas esperadas;
Uma taxa de aprendizado inicial η;
O número de neurônios na camada escondida $n_{escondida}$;
O *momentum* α;
O acréscimo e decréscimo da taxa de aprendizado, η_{dec} e η_{acres}, respectivamente.

Inicialização:
Inicialize um vetor de pesos sinápticos w com valores pequenos;
% a entrada do neurônio i em j é denotado x_{ji} e o peso do neurônio i em j é denotado w_{ji}.
Inicialize um contador de épocas $t = 1$;

Treinamento:
Enquanto (*condição de parada*)
% A condição de parada é determinada pelo número de épocas, erro mínimo atingido ou testes de performance no conjunto de validação.
Para cada $\langle x, y \rangle$ em $\mathcal{D}$ faça

1. Entre com a instância x para a rede neural e compute a saída s_u de cada neurônio u;
2. Para cada neurônio da camada de saída u_{out}, calcule o gradiente $\delta_{u_{out}} = s_{u_{out}}(1 - s_{u_{out}})(y_{u_{out}} - s_{u_{out}})$;
3. Para cada neurônio da camada escondida u_{hid}, calcule o gradiente $\delta_{u_{hid}} = s_{u_{hid}}(1 - s_{u_{hid}}) \sum_{u_{out} \in \text{Saídas}} w_{u_{out}u_{hid}} \delta_{u_{out}}$;
4. Atualize cada peso $w_{ji} = w_{ji} + \Delta w_{ji}(t)$ onde:

$$\Delta w_{ji}(t) = \eta \delta_j x_{ji} + \alpha \Delta w_{ji}(t-1)$$

% em que $\Delta w_{ji}(t)$ significa o termo Δw_{ji} calculado na t-ésima época; e δ_j corresponde a $\delta_{u_{out}}$ ou $\delta_{u_{hid}}$, a depender do neurônio cujos pesos estão sendo atualizados.

5. Atualize a taxa de aprendizado $\eta = \begin{cases} \eta.\eta_{dec}, & \Delta w_{ji}(t-1) < \Delta w_{ji}(t) \\ \eta.\eta_{acres}, & \text{caso contrário} \end{cases}$

A estratégia de aprendizado apresentada no Algoritmo 1 faz o treinamento de uma rede neural com base no erro cometido pelos neurônios da camada de saída, e tem o seu treinamento melhorado e acelerado pela estratégia de adicionar: o *momentum*, que guia a convergência do algoritmo para um ponto de mínimo, mesmo que o gradiente esteja passando por um platô na função a ser minimizada; e a taxa de aprendizado adaptativa (veja uma discussão em [20]), que faz o gradiente dar passos maiores em direção a um ponto de mínimo de acordo com a magnitude do erro apresentado pela rede neural .

3.3 *TRADABOOST*

O algoritmo *Transfer AdaBoost* (*TRAdaBoost*), proposto por Dai *et al.* [21], implementa a classe de Aprendizado por Transferência Indutivo baseado na abordagem de transferência por instância. Em linhas gerais, o *TRAdaBoost* trabalha com dados rotulados no domínio de origem D_o e de destino D_d, e resolve uma tarefa de indução binária[8] para o domínio de destino (D_d) fazendo uso dos dados de ambos os domínios. Os dados provenientes do domínio de origem (D_o) são ponderados adequadamente pelo algoritmo, baseado no estudo do quão úteis eles são para melhorar o aprendizado no domínio de destino (D_d). O algoritmo está ilustrado no Algoritmo 2.

Este algoritmo é especialmente útil para lidar com "novos" domínios, ou com "novas variações" na distribuição de probabilidade de um domínio. Nessas novas situações, em muitos casos, existem poucos dados ou tem-se um custo alto para rotular uma grande quantidade de dados. Essas situações, aliadas à necessidade de massa de dados para construção de uma boa função de predição, pode tornar inviável o estabelecimento de um modelo indutivo. Então, a estratégia implementada pelo *TRAdaBoost* supre essa limitação de poucos dados rotulados induzindo um modelo para o novo contexto fazendo uso, adequado, de recursos disponíveis em um domínio de origem (D_o).

Note que neste algoritmo são usados dois conjuntos de dados, sendo que o que é chamado de "origem" é um conjunto "ultrapassado" em relação ao conjunto de dados de interesse atual (o de destino). Isso significa que, embora seja construído um grande conjunto de dados ($\mathcal{D}$) para treinamento a fim de fortalecer o aprendizado de um modelo de predição, nem todos os dados presentes nesse conjunto contribuem para que a superfície de decisão induzida pelo algoritmo seja adequada para resolver a tarefa do destino T_d. A descoberta sobre a contribuição que cada dado "ultrapassado" e o uso desses dados ponderado por esta contribuição caracterizam a transferência de conhecimento implementada no *TRAdaBoost*.

O procedimento implementado pelo *TRAdaBoost* possui algumas estratégias que são interessantes de serem discutidas. São elas:

- N indica o número de modelos classificadores que serão gerados pelo *TRAdaBoost*. Esses classificadores formam a hipótese $h_f(x)$ a partir da aplicação de um esquema de votação;

[8] O algoritmo original está preparado para resolver um problema de classificação binário.

Algorithm 2 *TRAdaBoost* (traduzido e adaptado de [21])

Entrada:
Um conjunto de dados rotulados no domínio de origem $\mathcal{D}_o = (x_i^o, y(x_i^o))$ onde $x_i^o \in \chi_o, i = 1, \cdots, n$, e $y \in \Upsilon$;
Um conjunto de dados rotulados de destino $\mathcal{D}_d = (x_j^d, y(x_j^d))$ onde $x_j^d \in \chi_d, j = 1, \cdots, m$ e $y \in \Upsilon$;
% os conjuntos de dados de origem e destinos compõem juntos o conjunto de dados de treinamento de tamanho $n + m$.
Um conjunto de dados S não rotulado;
% os rótulos dos dados conjunto S devem estar disponíveis para fins de aferição do desempenho do algoritmo no caso de experimentações como as discutidas no presente artigo.
Um **Algoritmo Básico de Aprendizado**;
% qualquer algoritmo capaz de induzir um modelo de predição.
Um número máximo de iterações N.

Inicialização:
Inicialize um vetor de pesos $w^1 = (w_1^1, ..., w_{n+m}^1)$;
% a decisão sobre como inicializar o vetor w^1 é deixada para o usuário do algoritmo.
O conjunto completo de treinamento $\mathcal{D} = \mathcal{D}_o \cup \mathcal{D}_d$
Inicialize um fator de ponderação $\beta = \frac{1}{1+\sqrt{2ln(n)/N}}$

Treinamento:
Para $t = 1 \rightarrow N$ Faça

1. Determine uma distribuição de probabilidade $p^t = w^t / \sum_{i=1}^{n+m} w_i^t$;
2. Execute o **Algoritmo Básico de Aprendizado** com os paramêtros: o conjunto $\mathcal{D}$ modificado segundo a distribuição de probabilidade p^t e o conjunto não-rotulado S. Obtenha a hipótese h_t: $\chi \rightarrow \Upsilon$.
3. Calcule o erro (ϵ_t) de h_t em $\mathcal{D}_d$: $\epsilon_t = \sum_{i=n+1}^{n+m} \frac{w_i^t . |h_t(x_i) - y(x_i)|}{\sum_{i=n+1}^{n+m} w_i^t}$
4. Determine a variável de poderação $\beta_t = \epsilon_t / (1 - \epsilon_t)$;
5. Calcule um novo vetor de pesos: $w_i^{t+1} = \begin{cases} w_i^t . \beta^{|h_t(x_i) - y(x_i)|}, & 1 \leq i \leq n \\ w_i^t . \beta_t^{-|h_t(x_i) - y(x_i)|}, & n+1 \leq i \leq n+m \end{cases}$

Teste:
Saída: $h_f(x) = \begin{cases} 1, \prod_{t=\lceil N/2 \rceil}^{N} \beta_t^{-h_t(x)} \geq \prod_{t=\lceil N/2 \rceil}^{N} \beta_t^{-1/2} \\ 0, \text{caso contrário} \end{cases}$

- Na distribuição de probabilidades p^t, cada elemento representa o quanto um determinado dado do conjunto de dados de treinamento $\mathcal{D}$ pode contribuir para a definição de uma superfície de decisão que resolve a tarefa T_d. As probabilidades são calculadas, para cada hipótese gerada (na iteração t), a partir da informação ponderada sobre erros de classificação cometidos pela hipótese criada na iteração $t - 1$. As contribuições dos dados são refletidas para o **Algoritmo Básico de Aprendizado** como a probabilidade de cada dado ocorrer no conjunto de treinamento a ser usado por ele.

- O **Algoritmo Básico de Aprendizado** é a estratégia escolhida para a construção de uma hipótese - no caso do presente trabalho a estratégia é a Rede Neural *Multilayer Perceptron* discutida na Seção 3.2. Esta hipótese é construída usando o conjunto de treinamento D que, como mencionado, é formado pelo conjunto de "destino" completo, adicionado dos dados do conjunto de "origem" indicados pela distribuição de probabilidades p^t.

- O erro ϵ_t diz respeito ao erro da hipótese h_t em relação aos dados do conjunto de dados de destino $\mathcal{D}_d$ usados no treinamento. Trata-se de um erro ponderado, para o qual o erro cometido em cada dado (x_i) contribui de acordo com a sua importância no modelo gerado, contribuição essa representada por w_i^t.

- O fator β e a variável β_t tem o papel de calibrar os pesos que serão atribuídos aos dados, definindo a contribuição que cada dado terá para a indução da próxima hipótese. Para ponderação referente ao conjunto de domínio de origem $\mathcal{D}_o$ tem-se estabelecido no algoritmo o fator β que enfraquece a contribuição dos dados, erroneamente classificados, a cada iteração. Para ponderação referente ao conjunto de destino $\mathcal{D}_d$, a variável β_t é definida em termos da qualidade da resposta da hipótese h_t: quanto melhor a h_t resolve o problema de destino T_d menor são os valores assumidos por essa variável.

- A atualização dos pesos w^{t+1} é realizada considerando a resposta da hipótese h_t e as variáveis de ponderação.

  - Para cada dado do conjunto de origem $\mathcal{D}_o$: se a hipótese fornecer uma resposta de classificação correta, o peso w^{t+1} correspondente continuará o mesmo, uma vez que β será elevado a um expoente igual a 0; caso contrário, o peso sofrerá uma ponderação que diminuirá o seu impacto na indução da hipótese h_{t+1} (realizando uma transferência negativa), uma vez que a variável de ponderação é menor do que 1 e o seu expoente será igual a 1. Observe então que com o passar das iterações, dados do conjunto de origem $\mathcal{D}_o$ que as hipóteses não conseguem aprender serão cada vez mais desprezados no aprendizado, evidenciando que estes dados não são úteis na transferência de conhecimento.

  - Para cada dado do conjunto de destino $\mathcal{D}_d$: o peso w_{t+1} será maior para os dados que a hipótese atual tem dificuldade em fornecer a resposta correta de classificação (observe que β_t é menor que 1 e o expoente aplicado a ele é negativo); ou seja, dados difíceis de serem aprendidos em T_d aparecerão com uma probabilidade maior no conjunto de treinamento $\mathcal{D}$ para a indução da próxima hipótese; já quando a resposta da hipótese atual é correta, o peso para o respectivo dado continuará o mesmo.

- O algoritmo tem como saída a hipótese $h_f(x)$. Para instanciá-la, é proposto por Dai *et al.* [21], um esquema de votação sobre o resultado dos $\lceil N/2 \rceil$ últimos classificadores. Nesse esquema, se a maioria dos classificadores $h_t(x)$ responder com classe 0, o produtório de $\prod_{t=\lceil N/2 \rceil}^{N} \beta_t^{-h_t(x)}$ não será alterado, contribuindo para que seu resultado seja menor do que o limiar $\prod_{t=\lceil N/2 \rceil}^{N} \beta_t^{-1/2}$ e $h_f(x)$ seja 0; caso contrário, se a maioria dos classificadores $h_t(x)$ responder com classe 1, o resultado do produtório aumentará contribuindo para que ele seja maior do que o limiar e $h_f(x)$ seja 1.

A Figura 3 representa o contexto dos dados e classificadores em algumas das iterações do algoritmo *TRAdaBoost*. Na Figura 3(a) é mostrado o classificador obtido na primeira iteração do algoritmo, assumindo o conjunto de dados original (sem aplicação de pesos). Após pelo menos uma iteração, a distribuição de probabilidade ponderada por pesos cria uma situação como ilustrado na Figura 3(b), onde os dados representados por quadrados dizem respeito ao conjunto do domínio de destino ($\mathcal{D}_d$) e, se estes são representados por quadrados maiores, significa que o classificador da iteração anterior não obteve sucesso em classificá-los. A representação com quadrados de tamanho maiores indica que estes dados aparecerão em maior número no conjunto de dados de treinamento para o classificador atual, de forma que a dificuldade em aprendê-los seja minimizada pela contribuição que eles darão para a indução do classificador. Já os dados representados por círculos dizem respeito ao conjunto de domínio de origem ($\mathcal{D}_o$). Quando os círculos diminuem de tamanho de uma iteração para outra, significa que os dados do domínio de origem $\mathcal{D}_o$ estão representando uma dificuldade ao classificador e portanto, não são úteis para a predição do modelo objetivado e devem ter sua influência minimizada, ou mesmo desconsiderada, no processo. Após várias iterações, o resultado das aplicações das distribuições de probabilidades ponderadas sobre o conjunto de dados ($\mathcal{D}$) é ilustrado na Figura 3(c); o conjunto de classificadores obtidos no final da execução do *TRAdaBoost* é ilustrado na Figura 3(d). Por gerar um conjunto de classificadores, a fase de treinamento no *TRAdaBoost* é computacionalmente custosa, entretanto é esperado que o ganho em desempenho de classificação supere as desvantagens inerentes a tal custo.

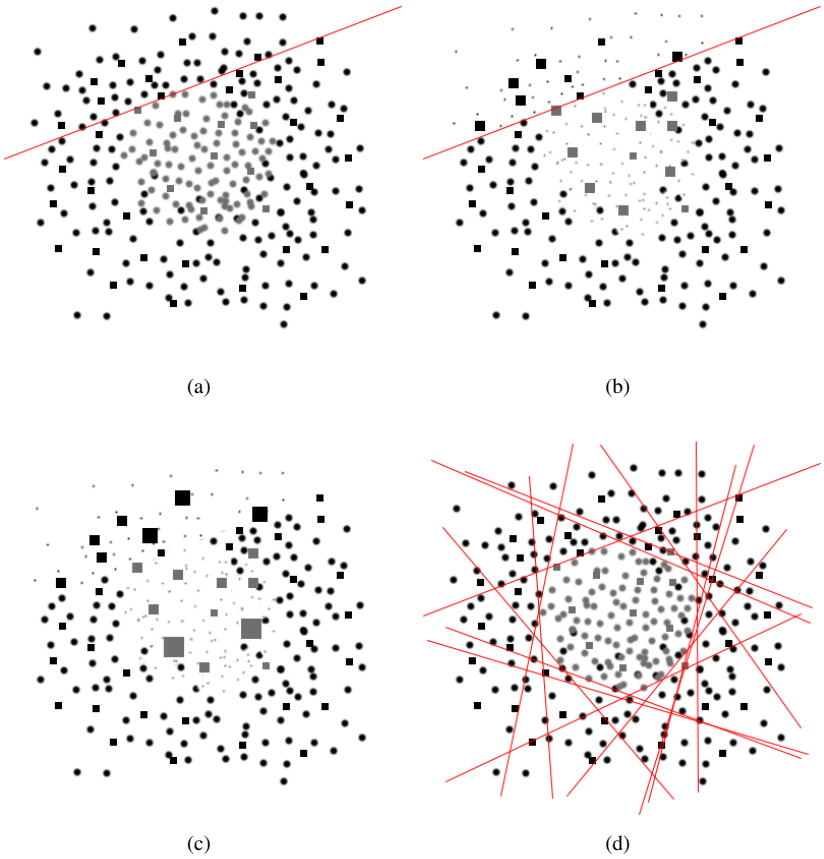


Figura 3: Reprentação gráfica ilustrando um contexto genérico de iterações do *TRAdaBoost* em relação à situação dos conjuntos de dados e o papel de cada classificador. Os quadrados representam o conjunto do domínio de destino ($\mathcal{D}_d$) e o conjunto do domínio de origem ($\mathcal{D}_o$) é representado pelos círculos. A cor preta representa exemplos de uma classe e a cor cinza exemplos de outra classe.

Cada classificador obtido nas N iterações do algoritmo *TRAdaBoost* pode ser considerado, neste contexto, como um classificador fraco, visto que ele consegue discriminar bem, muito provavelmente, apenas uma parte do conjunto de dados a ser aprendido. O conjunto de todos os N classificadores fracos é chamado de classificador forte, e este deve ser capaz de discriminar bem todo o conjunto de dados.

4. EXPERIMENTOS

Os experimentos aqui discutidos estão relacionados ao contexto de análise de gestos, mais especificamente o conjunto é formado por movimentos que compõem o conjunto básico de movimentos usados na construção dos sinais da Língua Brasileira de Sinais (LIBRAS). São 28 movimentos diferentes, sendo que alguns deles assumem trajetórias de mesmo formato, porém com direções diferentes. A Figura 4 traz as gráficos representativos dos diferentes tipos de movimentos usados no presente trabalho. As cores usadas nas figuras mostram a direção assumida em cada movimento: cores escuras indicam o início do movimento e cores claras indicam o final dos mesmos.

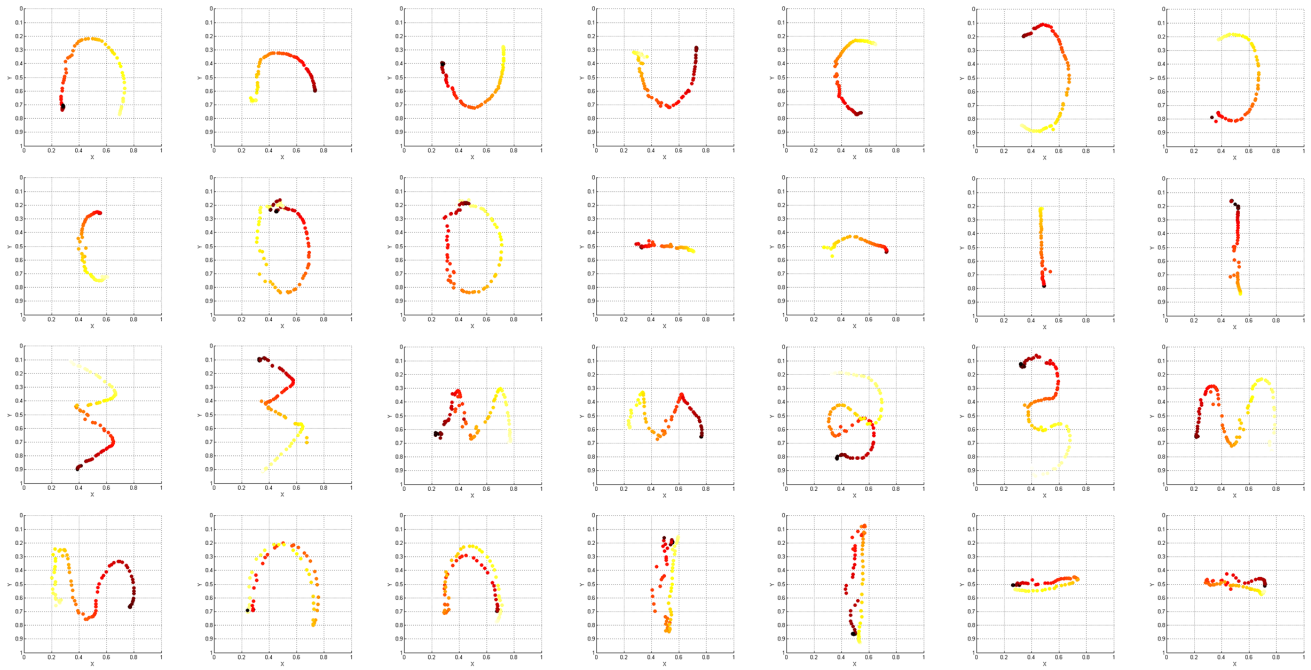


Figura 4: Os 28 movimentos usados na construção dos sinais da Libras. Observando a figura de cima para baixo e da esquerda para a direita, tem-se os seguintes movimentos: curva para cima para direita; curva para cima para esquerda; curva para baixo para direita; curva para baixo para esquerda; arco horário para cima; arco horário para baixo; arco anti-horário para cima; arco anti-horário para baixo; círculo horário; círculo anti-horário; horizontal para direita; horizontal para esquerda; vertical para cima; vertical para baixo; zig-zag vertical para cima; zig-zag vertical para baixo; zig-zag horizontal para direita; zig-zag horizontal para esquerda; ondulatório vertical para cima; ondulatório vertical para baixo; ondulatório horizontal para direita; ondulatório horizontal para esquerda; balançar curvo a partir da esquerda; balançar curvo a partir da direita; balançar vertical a partir de cima; balançar vertical a partir de baixo; balançar horizontal a partir da esquerda; balançar horizontal a partir da direita.

No presente artigo, a análise dos movimentos é implementada como uma tarefa de classificação, resolvida por um algoritmo de Aprendizado de Máquina em uma estratégia clássica (sozinho) e dentro de uma estratégia de Aprendizado por Transferência baseado em instância. O esquema gráfico mostrado na Figura 5 explica como o Aprendizado por Transferência foi instanciado nos experimentos. Como entradas para o Aprendizado por Transferência Indutivo tem-se o conjunto de dados da aplicação (conjunto de dados de origem $\mathcal{D}_o$) – um conjunto grande, e o conjunto de dados rotulado pelo usuário (conjunto de dados de destino $\mathcal{D}_d$) – um conjunto pequeno. Tendo como saída a resolução da tarefa de origem (T_o) e destino (T_d) com transferência baseada em instâncias.

O restante desta seção está organizado em três partes: na primeira é apresentado o conjunto de dados que compõe o universo de experimentação e como os dados do conjunto foram captados, representados, pré processados e então armazenados; na segunda parte são apresentados alguns experimentos usando a arquitetura de Rede Neural *Multilayer Perceptron*; na última parte é descrito o experimento que testa a hipótese de que o Aprendizado por Tranferência oferece um meio adequado para implementar aplicações multiusuários, que envolvam reconhecimento de padrões em língua de sinais. No total foram realizados seis experimentos, cinco usando o algoritmo *Multilayer Perceptron* e um usando o algoritmo *TRAdaBoost*.

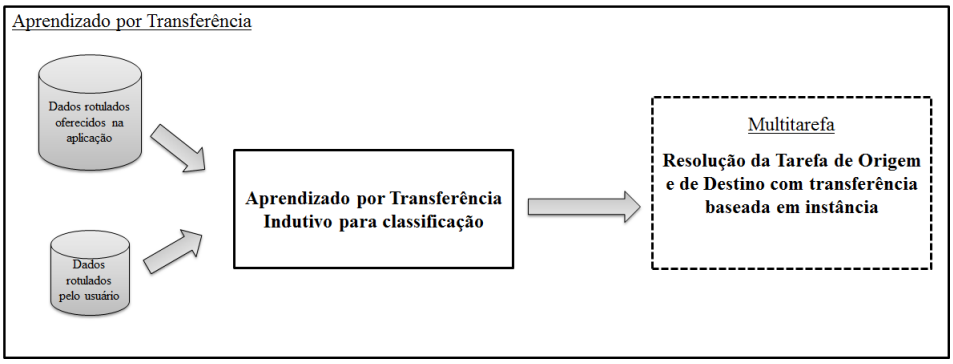


Figura 5: Aprendizado por Transferência instanciado no experimento discutido no presente artigo.

4.1 CONJUNTO DE DADOS

O conjunto de dados é composto por movimentos executados por sete pessoas diferentes (aqui denominados usuários), em diferentes sessões (cada sessão corresponde a uma data diferente de captação de dados). A Tabela 1 lista as principais informações descritivas da composição do conjunto de dados. O conjunto é composto de 725 dados (cada dado é uma execução de um dos 28 movimentos). Dois usuários participaram de duas sessões diferentes de captação, em dias diferentes. A última coluna dessa tabela traz a informação referente à quantidade de execuções de cada tipo de movimento realizadas em cada sessão. No caso das sessões do usuário **G**, algumas execuções foram desconsideradas por conterem erros de captação. O usuário **G** possui experiência na execução dos movimentos; os demais usuários não possuem experiência na execução dos movimentos. Os experimentos realizados neste estudo consideraram todo o conjunto de dados, os dados do usuário **A** (usuário inexperiente), os dados do usuário **G** (usuário experiente), e a combinação dos dados desses dois usuários (**A** – usuário inexperiente e com a menor quantidade de dados[9] que possibilitaria experimentos de treino e teste, e **G** – usuário experiente e com a maior quantidade de dados).

Tabela 1: Conjunto de dados separados por usuários e sessões.

Identificador do usuário	Identificador das sessões	Quantidade de dados	Distribuição por tipo de movimento
A	Sessão 1	56	2 dados
B	Sessão 1	84	3 dados
C	Sessão 1	28	1 dados
D	Sessão 1	84	3 dados
E	Sessão 1	84	2 dados
F	Sessão 1	112	4 dados
	Sessão 2	56	2 dados
G	Sessão 1	138	5 dados / os movimentos 23 e 11 possuem apenas um dado
	Sessão 2	83	2 dados / o movimento 8 possui apenas uma dado

A captação foi realizada usando o dispositivo *Kinect*™da *Microsoft* e o *Kinect for Windows SDK 1.0*[10]. Os usuários, posicionados na frente do dispositivo de captação, executavam os movimentos usando o braço direito. Assim, para gravar as informações sobre a trajetória do movimento, apenas as coordenadas do ponto de interesse referente à mão direita (dos 20 pontos oferecidos pelo dispositivo) foi necessário. Alguns parâmetros livres do dispositivo foram calibrados para que a captação obtivesse sucesso. A calibração seguiu um processo de inspeção de vários parâmetros e também as indicações de [22]. São eles: suavização em $0,5$; correção em $0,5$; predição em $0,5$; raio de tremulação em $0,05$ (em metros); raio de desvio máximo em $0,04$ (em metros).

Como apresentado em [8], um movimento da mão no espaço caracteriza uma curva executada pela mão em um período de tempo. Segundo os mesmos autores, essa curva pode ser formalizada como uma função contínua $f : G = [a, b] \subset \mathbb{R} \rightarrow \mathbb{R}^2$, ou seja, para cada ponto $t \in [a, b]$, é associado um ponto $f(t) = (u(t), v(t)) \in \mathbb{R}^2$.

Contudo, a captação de dados fornece alguns pontos no espaço bidimensional que podem ser vistos como uma discretização a curva f. Cada movimento, neste trabalho, é uma curva amostrada com uma quantidade diferente de pontos bidimensionais, a depender do período de tempo de amostragem (de execução do movimento).

Devido às diferentes quantidades de pontos captados em cada movimento e também à alta variabilidade de altura e amplitude dos movimentos executados, alguns procedimentos de pré-processamento foram aplicados aos dados originais antes que eles

[9]Poucos dados deste usuário simula uma situação real de uso da aplicação, na qual um usuário forneceria uma quantidade mínima de dados como entrada.
[10]`http://www.microsoft.com/en-us/kinectforwindows/develop/developer-downloads.aspx`.

fossem incorporados ao conjunto de dados. O pré-processamento aplicado aos dados originais, captados do dispositivo, consistiu de três etapas: padronizar a quantidade de pontos que amostra a curva f para a quantidade de 100 pontos bidimensionais (x, y)[11]; transladar os movimentos para o centro de todos os gestos; normalizar os movimentos para o intervalo [0,1].

A padronização executada na primeira etapa de pré-processamento foi baseada na inserção de novos pontos na amostragem de f ou na retirada de pontos da amostragem de f, a depender se o dado original era composto, respectivamente, por mais ou por menos de 100 pontos. A inserção de novos pontos se deu por meio do cálculo de um ponto médio entre dois pontos consecutivos do dado original, de maneira uniformemente distribuída por toda a extensão do dado. Já a retirada de pontos foi realizada pela exclusão de pontos escolhidos de maneira também uniformemente distribuída. O passo n para inserção ou retirada de um ponto foi calculado por

$$n = \left\lceil \frac{\textit{tamanho do gesto}}{|\textit{tamanho do gesto} - 100|} \right\rceil,$$

sendo *tamanho do gesto* o número de pontos que compõe a amostragem final da curva f correspondente ao movimento.

A segunda etapa, para centralizar todos os gestos em um ponto, foi utilizada a estratégia de calcular o *centro do gesto* e transportá-lo para a coordenada que é o *centro de todos os gestos*. Para calcular o *centro do gesto* foi usada a seguinte fórmula:

$$\textit{centro do gesto} = \frac{(x, y)_{min} + (x, y)_{max}}{2},$$

assim, para calcular o *centro de todos os gestos* foi usada a mesma fórmula com a diferença de que $(x, y)_{min}$ e $(x, y)_{max}$ será de todos os gestos e não apenas um gesto. Então, para transladar o gesto, cada ponto (x, y) do gesto era subtraído pelo fator *centro do gesto* − *centro de todos os gestos*.

Na terceira etapa foi realizada uma normalização segundo sugerido em [23]:

$$(x, y)_{normalizado} = \frac{(x, y) - (x, y)_{min}}{(x, y)_{max} - (x, y)_{min}},$$

sendo $(x, y)_{min}$ e $(x, y)_{max}$ a coordenada mínima e máxima de todos os movimentos captados.

Assim, o dado que de fato ficou armazenado no conjunto de dados possui uma representação vetorial localizada na dimensão 200. O rótulo do movimento representado pelo dado também é armazenado no conjunto de dados, juntamente com o vetor de representação do movimento. Foram usados números de 1 a 28 para rotulação, seguindo a ordem de apresentação dos diferentes tipos de movimentos da Figura 4.

4.2 EXPERIMENTOS COM *MULTILAYER PERCEPTRON*

Para implementar a Rede Neural *Multilayer Perceptron* optou-se por usar a estratégia de treinamento *Backpropagation* com gradiente descendente, *momentum* e taxa de aprendizado adaptativa. Em todos os experimentos os seguintes parâmetros se mantiveram constantes: 200 neurônios na camada de entrada; 28 neurônios na camada de saída; função de ativação tangente hiperbólica sigmóide; taxa de aprendizado η igual a $0,01$; taxa de acréscimo η_{acres} para taxa de aprendizado igual a $1,05$; taxa de decréscimo η_{dec} da taxa de aprendizado igual a $0,7$; e *momentum* α igual a $0,9$[12].

Já o estudo dos parâmetros correlatos ao número de neurônios e ao número de épocas de treinamento foi realizado. Assim, o primeiro experimento consistiu de uma rotina para descobrir a quantidade de neurônios mais adequada (melhor relação entre média de acerto e variância obtida nas execuções de testes) para a camada escondida da rede neural. Foram criadas 30 arquiteturas de rede neural, cada uma com uma quantidade de neurônios na camada escondida, variando de 10 a 300 neurônios, com passo de 10 neurônios entre cada montante. Dez treinamentos para cada arquitetura de rede neural foram executados, usando todo o conjunto de dados disponível, dividido em três partes: 70% do conjunto para treino, 15% para teste e os outros 15% do conjunto para validação. Para cada um dos treinamentos, uma amostragem aleatória foi executada para compor essas três partições. O critério de parada do treinamento das redes neurais foi o desempenho do modelo no teste de validação, ou seja, se o desempenho do modelo, considerando o conjunto de validação, piorasse por 30 épocas consecutivas (isto é, o erro aumentasse nessas 30 épocas), o treinamento era parado e o modelo final considerado era aquele obtido na última época antes do desempenho começar a sofrer essa piora.

O resultado dessa inspeção é mostrado na Figura 6(a) e Figura 6(c). As variações de desempenho ilustradas pelos gráficos referentes à variância indicam que os modelos ainda apresentam potencial para serem refinados. Observando as figuras é possível verificar que existem diferentes arquiteturas que alcançaram desempenhos similares em termos de média de classificações corretas. A arquitetura com 110 neurônios na camada escondida alcançou a maior média de acertos, e a menor variância. Esta

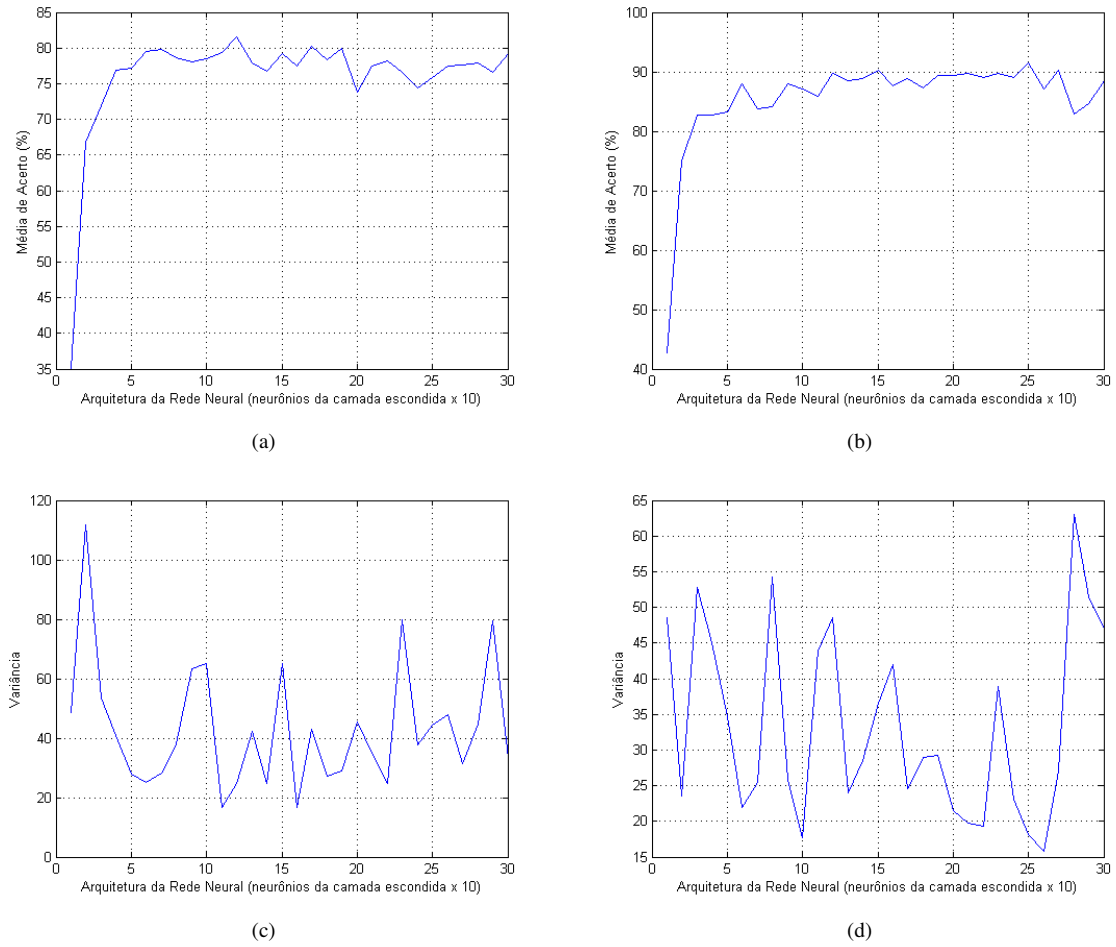


Figura 6: Média de classificações corretas e variância referente aos diferentes modelos *Multilayer Perceptron* treinados: (a) e (c) se referem ao primeiro experimento; (b) e (d) se referem ao segundo experimento. Note que, na escala dos gráficos no eixo x, para saber o número total de neurônios na camada escondida é necessário multiplicar o número no eixo por 10.

arquitetura alcançou um desempenho médio de $79,4\%$ de classificações corretas para os conjuntos de teste, considerando um número médio de épocas de treinamento de 150 épocas.

Um segundo experimento consistiu da utilização da mesma rotina descrita no primeiro experimento, porém usando apenas o conjunto de dados do usuário **G**. A motivação para a execução deste procedimento é construir um modelo sobre um subconjunto de dados que, posteriormente, no sexto experimento será usado como o conjunto de dados do domínio de origem $\mathcal{D}_o$, para possibilitar comparações entre esta forma clássica de resolver problemas e a forma implementada pelo Aprendizado por Transferência.

Nesse experimento, devido ao baixo número de instâncias no subconjunto de dados, foi necessário realizar uma estratificação dos dados para garantir que todos as partições (treino, teste ou validação) tivessem instâncias de todas as 28 classes. A arquitetura com 250 neurônios na camada escondida apresentou a maior média de acertos com a menor variância (Figura 6(b) e Figura 6(d)), alcançando uma média de $91,5\%$ de classificações corretas nos conjuntos de teste, com uma quantidade média de 137 épocas de treinamento.

A fim de atestar a existência de alguma relação entre os desempenhos das arquiteturas testadas, foi realizado um teste estatístico (Wilcoxon Rank-Sum Test [24]) sobre as distribuições dos erros de teste obtidos nas 10 execuções de cada arquitetura. Os resultados obtidos nesse teste estatístico não permitiram tirar conclusões sobre diferenças estatísticas no desempenho das arquiteturas (o teste falha em rejeitar a hipótese nula) para a maior parte das comparações. Então, os demais experimentos descritos neste artigo foram realizados sobre as arquiteturas com a maior média de acertos de classificações nos dois experimentos iniciais, e um teste empírico foi realizado usando a arquitetura um camada escondida menor.

Também foram geradas uma matrizes de confusão a partir dos melhores modelos dentre os dez obtidos com a arquitetura de 110 neurônios (Figura 7(a) – Experimento 1), e dentre os dez obtidos com a arquitetura de 250 (Figura 7(b) – Experimento 2). Observe na Figura 7(a) que o respectivo modelo tem dificuldade em tratar os movimentos ondulatórios e de zig-zag, os quais

[11] A coordenada z (profundidade) foi retirada porque não trazia informação rrelevante para os tipos de gestos em questão no presente estudo.

[12] Os valores para estes parâmetros não foram inspecionados em nossos testes. Usamos as sugestões de valores padrões oferecida na ferramenta (*Neural Network Toolbox* do software Matlab®) usada para implementar a Rede Neural *Multilayer Perceptron*. Portanto, nossas análises de desempenho e de adequação da abordagem não considera nenhum processo de refinamento relacionado a tais parâmetros.

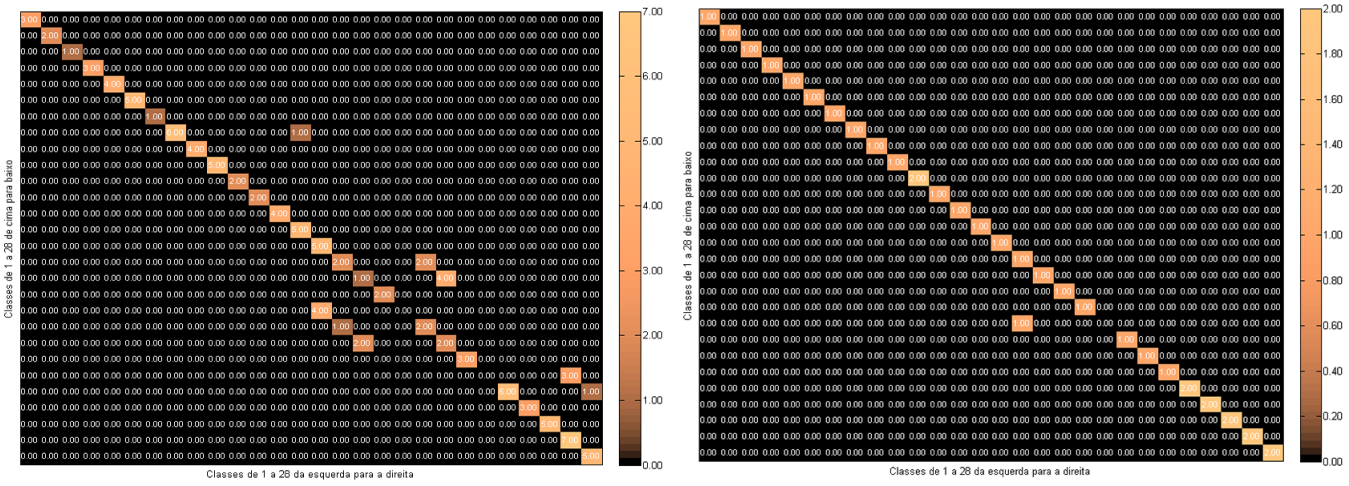
Tabela 2: Resultados obtidos para o teste estatístico realizado sobre os erros observados no conjunto de teste, para cada arquitetura do Experimento 1. O teste considera a arquitetura com 110 neurônios (arquitetura 11, que obteve a maior média de classificações corretas no experimento) para a realização das comparações par a par. O valor abaixo do identificador da arquitetura é referente ao $p - value$ obtido em cada comparação.

Arq.1	Arq.2	Arq.3	Arq.4	Arq.5	Arq.6	Arq.7	Arq.8	Arq.9	Arq.10
0,0002	0,0031	0,0111	0,3832	0,2703	0,9093	0,7901	0,9395	0,9394	0,8494
Arq.11	Arq.12	Arq.13	Arq.14	Arq.15	Arq.16	Arq.17	Arq.18	Arq.19	Arq.20
–	0,3827	0,3432	0,2870	0,9092	0,3055	0,9092	0,6203	0,9091	0,0484
Arq.21	Arq.22	Arq.23	Arq.24	Arq.25	Arq.26	Arq.27	Arq.28	Arq.29	Arq.30
0,2879	0,3624	0,3237	0,1109	0,1391	0,3624	0,5932	0,5429	0,4480	0,8791

Tabela 3: Resultados obtidos para o teste estatístico realizado sobre os erros observados no conjunto de teste, para cada arquitetura do Experimento 2. O teste considera a arquitetura com 250 neurônios (arquitetura 25, que obteve a maior média de classificações corretas no experimento) para a realização das comparações par a par. O valor abaixo do identificador da arquitetura é referente ao $p - value$ obtido em cada comparação.

Arq.1	Arq.2	Arq.3	Arq.4	Arq.5	Arq.6	Arq.7	Arq.8	Arq.9	Arq.10
0,0002	0,0002	0,0017	0,0124	0,0052	0,1201	0,0052	0,0239	0,1243	0,0389
Arq.11	Arq.12	Arq.13	Arq.14	Arq.15	Arq.16	Arq.17	Arq.18	Arq.19	Arq.20
0,0542	0,8772	0,2029	0,2476	0,7292	0,2442	0,2468	0,0777	0,2868	0,4181
Arq.21	Arq.22	Arq.23	Arq.24	Arq.25	Arq.26	Arq.27	Arq.28	Arq.29	Arq.30
0,3818	0,2233	0,5826	0,2635	–	0,0185	0,3874	0,0171	0,0416	0,3515

são muito parecidos e de difícil gesticulação. Na (Figura 7(b)) esta ilustrado o único erro de teste cometido pelo modelo que a segunda matriz de confusão: movimentos da classe ondulatório para baixo (rótulo número 20).



(a) (b)

Figura 7: Matriz de confusão: (a) referente ao primeiro experimento; (b) referente ao segundo experimento. As células em cor laranja fora da diagonal indicam as situações de classificação errada.

No terceiro experimento foi utilizada o melhor modelo gerado no segundo experimento para testar o seu desempenho no conjunto de dados do usuário **A**. O resultado foi um desempenho de $66,1\%$ de taxa de classificações corretas. Observa-se com esses testes que o modelo aprendido com os dados de um usuário pode não ser adequado para ser aplicado nos dados de um segundo usuário. Muito provavelmente, o modelo induzido com os dados do usuário **G** está super ajustado às características de execução de movimentos deste usuário.

Também para fins de comparação com a estratégia de Aprendizado por Transferência, foi treinada uma rede neural *Multilayer Perceptron* (110 neurônios escondidos e 200 épocas) com dados de dois usuários (usuário **G**, e $1/2$ dos dados do usuário **A**), testando com os $1/2$ restantes do usuário **A**. Neste teste, a abordagem alcançou 64% de classificações corretas com os dados de testes e $84,3\%$ de classificações corretas para um teste de resubstituição (um teste com dados que participaram do treinamento). O objetivo do teste de resubstituição será abordado na Seção 4.4.

Completando a investigação, foi feito um experimento usando dados de treino e teste de um mesmo usuário (usuário **A**),

usando um conjunto de dados pequeno. A rede neural (110 neurônios escondidos e 200 épocas) foi treinamento com 28 dados, um representante de cada classe, e foi testado também com 28 dados também com um representante por classe (formando conjuntos de treino e teste disjuntos). O desempenho foi $85,7\%$ de classificações correta.

4.3 EXPERIMENTO COM *TRADABOOST*

O experimento com o *TRAdaBoost* teve o objetivo de verificar a hipótese de que o Aprendizado por Transferência pode suportar a adaptação de um modelo de predição em uma aplicação, fazendo-o se tornar orientado ao usuário, porém sem perda de generalidade. Para isso, a adaptação é realizada a partir de um conjunto de dados fornecido na aplicação (teoricamente contendo uma boa amostra de dados sobre o domínio de análise de movimento) adicionado de um pequeno conjunto de exemplos fornecido pelo usuário. Assim, esse experimento tratou da transferência de conhecimento do subconjunto de dados formado por: movimentos do usuário **G** (em quantidade maior) para ajudar na indução de um modelo capaz de predizer os movimentos feitos pelo usuário **A** (em quantidade menor).

Os parâmetros utilizados para o *TRAdaBoost* foram: o conjunto de dados do domínio de origem ($\mathcal{D}_o$) como o subconjunto formado pelos dados do usuário **G**, o conjunto de dados do domínio de destino ($\mathcal{D}_d$) como um subconjunto de dados do usuário **A** formado pelos 28 primeiros dados; o S como o subconjunto de dados rotulados para teste dos classificadores criados durante a execução do *TRAdaBoost*, extraídos dos próximos 28 dados do usuário **A**; $N = 10$; o vetor w^1 foi inicializado com $1s$; e o **Algoritmo Básico de Aprendizado** como uma Rede Neural *Multilayer Perceptron* configurada da mesma forma que aquelas usadas no segundo e terceiro experimentos da seção anterior, usando um total de 137 épocas[13]. Além dos parâmetros, tomou-se a decisão de trabalhar com a distribuição de probabilidade p^t multiplicando-a por 10 a fim de possibilitar um aumento no volume do conjunto de dados $\mathcal{D}$ e garantir que a força de contribuição esperada para cada dado tomasse efeito no conjunto de dados passado para o Algoritmo Básico de Aprendizado.

O algoritmo *TRAdaBoost* é projetado para resolver problemas de apenas duas classes e, para aplicá-lo ao problema de classificação de 28 movimentos, foi necessário utilizar a estratégia de *One-vs-all*[14], a exemplo de Lorena e Carvalho [25]. Portanto, neste experimento, o algoritmo *TRAdaBoost* foi executado 28 vezes, sendo que em cada uma delas uma das 28 classes foi escolhida para compor a classe positiva $y+$ de um classificador binário, enquanto as demais compuseram a classe negativa $y-$ do mesmo.

No entanto, essa abordagem inseriu uma dificuldade na interpretação dos resultados obtidos na saída h_f do algoritmo pois, para um determinado dado x de entrada, 280 classificadores serão testados da seguinte forma:

1. o dado x precisa ser avaliado por cada um dos 28 classificadores binários fortes construídos na estratégia *One-vs-all* aplicada no algoritmo *TRAdaBoost*;

2. cada um desses classificadores binários fortes é composto de N classificadores fracos construídos nas N iterações do *TRAdaBoost*, e o dado x precisa ser testado em todos eles (veja o cálculo da saída do *TRAdaBoost* no Algoritmo 2 para indicar o resultado h_f);

3. cada h_f obtida no item 2 é organizada em um vetor de classificações que indica as respostas de cada um dos 28 classificadores binários fortes para o dado x, e;

   - se apenas um classificador binário responder com o valor 1, então entende-se que a abordagem indica a classe $y+$ daquele classificador como sendo a classe do dado x;
   - se nenhum dos classificadores binários responder com o valor 1, então entende-se que nenhum deles soube reconhecer o dado x e este fato é computado como um erro da abordagem;
   - se mais de um classificador binário responder com o valor 1, então dentre os classificadores que responderam com 1, verifica-se aquele que possui o valor máximo de $\prod_{t=\lceil N/2 \rceil}^{N} \beta_t^{-h_t(x)}$ para determinar a resposta da abordagem para a classificação do dado x;

O experimento resultou em uma taxa de classificações corretas de $78,6\%$. Na Figura 8 é mostrada uma matriz de confusão referente a este experimento, sendo que as linhas e colunas que contém apenas zeros representam classes para as quais não houve nenhum reconhecimento de movimento (é o caso para o qual nenhum classificador binário respondeu positivamente para dados daquela classe), embora o conjunto de teste tivesse um representante de cada classe de movimento.

Finalmente, um novo experimento com o TRAdaBoost foi executando usando uma arquitetura com 60 neurônios na camada escondida da rede neural *Multilayer Perceptron*. O intuito deste experimento foi verificar empiricamente se uma arquitetura mais simples, com desempenho similar porém inferior à melhor arquitetura nos experimento 1 e 2 , poderia alcançar um desempenho também similar no Aprendizado por Transferência. O resultado alcançado nessa verificação foi $71,4\%$ de classificações corretas para os dados de teste e $87,7\%$ no teste de resubstituição. Estes resultados mostraram um efeito de sobreajuste do modelo aos

[13] Nos experimentos realizados com a Rede Neural *Multilayer Perceptron* sozinha, essa quantidade de épocas foi alcançada antes que o modelo começasse a ficar super especializado (*overfitting).*

[14] Outra estratégia possível seria a *one-vs-one* ou *all-vs-all*. Neste caso seria necessário construir $k(k-1)/2$ classificadores e ainda implementar um método para obtenção de uma resposta final. Na estratégia *One-vs-all*, escolhida para ser usada no presente trabalho, apenas k classificadores são construídos e a resposta mais forte dentre as k respostas determina a classe do dado sob análise.

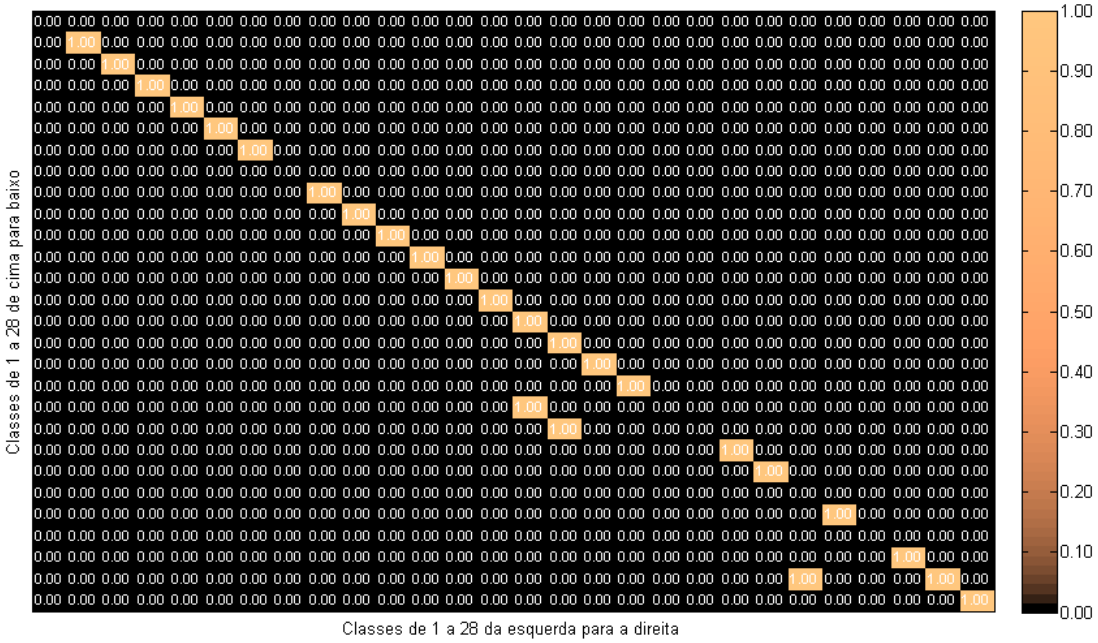


Figura 8: Matriz de confusão referente ao experimento com o *TRAdaBoost*.

dados de treinamento mais acentuado do que aquele obtido no Experimento 6. Por não ter mostrado um desempenho melhor, este experimento não consta na Tabela 4 e para ele não foram realizadas instâncias de comparação com arquiteturas simples de *Multilayer Perceptron*.

4.4 ANÁLISES

Na Tabela 4 estão listadas as principais características dos experimentos realizados, bem como os resultados obtidos em cada um deles.

Tabela 4: Resumo dos experimentos. MLP - experimentos executados com uma Rede Neural *Multilayer Perceptron*; TR - experimento executado com o algoritmo *TRAdaBoost*.

Id. experimento	**Estratégia de indução**	**Dados do conjunto de treinamento**		**Número de instâncias (treinamento)**		**Dados do conjunto de teste**	**Número de instâncias (teste)**	**Taxa de acerto (teste)**	**Taxa de acerto (resubstituição)**
1	MLP	70% de Todos		507		15% de Todos	108	79,4%	
2	MLP	70% de G		196		30% de G	163	91,5%	
3	MLP	G		221		A	56	66,1%	
4	MLP	50% de A		28		50% de A	28	85,7%	
5	MLP	G,50% de A		249		50% de A	28	64%	84,3%
		$\mathcal{D}_o$	$\mathcal{D}_d$	$\mathcal{D}_o$	$\mathcal{D}_d$				
6	TR	G	50% de A	221	28	50% de A	28	78,6%	83,2%

O Experimento 1 fornece uma noção da dificuldade inerente ao problema. Usando dados de 7 usuários diferentes, tanto para treinamento quanto para teste do modelo, sob a estratégia de teste *holdout*, obteve-se uma taxa de classificações corretas interessante, porém não alta o suficiente para permitir que uma aplicação real fosse colocada em uso. Já o Experimento 2 apresenta um desempenho que pode ser considerado bom, contudo, se trata de um modelo que foi construído com base em um único usuário. É um contexto onde existe pouca variabilidade na forma como a execução dos movimentos é realizada.

Já o Experimento 3 diz respeito a uma situação que pode ocorrer para uma aplicação real. Os dados de teste são de um usuário do qual não houve exemplos de execução de movimentos no conjunto de treinamento. O desempenho do modelo de reconhecimento cai para um valor bastante baixo, que inviabiliza o uso dele em uma aplicação real.

A situação ilustrada pelo Experimento 4 é referente a disponibilizar em uma aplicação uma forma de treinar um modelo de reconhecimento de movimentos totalmente centrado nos usuários, porém sem exigir do usuário o fornecimento de uma massa de dados de treinamento grande. O desempenho neste caso pode não ser considerado bom para uma aplicação real porém é um

bom desempenho no ambiente de testes. Um desempenho "bom" foi obtido por se tratar de uma situação de teste dependente do usuário (os dados no conjunto de treinamente e de teste são provenientes de uma mesma pessoa). Contudo, o desempenho foi inferior ao obtido no Experimento 2 (também dependente do usuário) e isso pode ter ocorrido por causa da pouca quantidade de dados para treinamento do modelo.

O Experimento 6 traz algumas evidências que motivam o uso de Aprendizado por Transferência para resolver o problema de balancear o desempenho do modelo de reconhecimento, considerando tanto a necessidade de atender à variabilidade específica trazida pela apresentação de dados de um novo usuário, quanto a necessidade de manter um nível de generalização que permita ainda que o modelo atenda diferentes usuários (e não apenas o novo). Este experimento foi executado em um contexto semelhante ao do Experimento 5, e obteve um desempenho melhor no teste de exemplos do usuário novo, e levemente inferior no teste de resubstituição.

Observe que no Experimento 5 uma arquitetura de MLP foi treinada e testada com ao mesmos conjuntos de treino e teste usados no teste do Aprendizado por Transferência, e o desempenho da MLP no conjunto de teste foi bastante baixo. Ainda sobre estes dois experimentos (5 e 6) foi realizado o teste de resubstituição a fim de verificar se havia algum efeito de sobreajuste do modelo aos dados de treinamento. O desempenho das duas estratégias foi similar nesse teste, evidenciando que o Aprendizado por Transferência se mostrou, de fato, mais adequado à esta situação – efeito de sobreajuste equiparável ao do modelo MLP porém com desempenho de teste superior.

5. CONCLUSÃO

Este artigo apresentou um estudo sobre a aplicação da Rede Neural Artificial *Multilayer Perceptron* e da estratégia de Aprendizado por Transferência em aplicações centradas no usuário. O domínio de aplicação diz respeito ao reconhecimento automatizado de movimentos usados na composição de sinais da Libras. Nesse estudo, a Rede Neural *Multilayer Perceptron* foi usada como modelo de indução de um classificador de movimentos, tendo sido aplicada diretamente na indução do modelo reconhecedor e também como algoritmo básico de aprendizado dentro da estratégia implementada no algoritmo *TRAdaBoost*. O artigo ainda trouxe uma pequena apresentação da área de Aprendizado por Transferência, dando ênfase à discussão do algoritmo escolhido para implementar a transferência de conhecimento baseada em instâncias.

O problema de reconhecimento de movimentos, contextualizado na Libras, foi apresentado, juntamente com uma proposta de representação de dados para este domínio. Os experimentos foram executados usando uma massa de dados composta por movimentos executados por diferentes usuários, em duas sessões distintas de captação realizadas com apoio do dispositivo *Kinect*™da *Microsoft*.

Os resultados obtidos são promissores. A abordagem de Aprendizado por Transferência tem potencial para suportar a construção de aplicações adequadas para o uso real, assumindo que novos usuários estarão em contato com a aplicação, e este inserirão alguma variabilidade de execução de movimentos. Os resultados obtidos neste trabalho superam resultados obtidos anteriormente, em um problema similar ao aqui tratado (movimentos da Libras), discutidos em [8]. No referido artigo, o reconhecimento de movimentos ficou abaixo de 60% de desempenho em taxas de classificação corretas, considerando conjuntos de dados com a presença de movimentos executados por diferentes usuários.

Além disso, os experimentos aqui descritos mostraram resultados comparáveis àqueles alcançados em trabalhos que também aplicaram Aprendizado por Transferência na análise de gestos. Como descrito na Seção 2, as taxas de acerto de classificações é da ordem de $78,14\%$ em [12] e $64,17\%$ em [15], considerando o desempenho das estratégias de Aprendizado por Transferência; nossa abordagem alcançou $78,6\%$, estando portanto, em um patamar similar. Porém, é necessário ressaltar que essas comparações devem ser consideradas com ressalvas, uma vez que as tarefas de reconhecimento, os conjuntos de dados e os algoritmos testados são diferentes. Em tempo, vale ressaltar que os trabalhos correlatos citados no presente artigo, o presente artigo, e muito do que se encontra na análise automatiza de gestos, apresentam limitações referentes a estudos comparativos, principalmente por conta da escassez de conjuntos de dados públicos. Uma breve discussão sobre este tema pode ser encontrada em [26].

A abordagem aqui estudada faz parte de um esforço de pesquisa mais amplo, relacionado à análise de gestos. Dentro do contexto desta pesquisa, algumas ações estão sendo executadas e os resultados esperados para estas ações vêm ao encontro da evolução do estudo apresentado no presente artigo. Dentre estas ações estão:

- aprimorar o conjunto de dados referente aos movimentos usados na Libras, a fim de construir uma massa de dados com mais instâncias e também mais diversificada (em termos de usuários); com o aprimoramento do conjunto de dados, outros experimentos serão projetados e executados no âmbito de Aprendizado por Transferência, incluindo modelagens de outras classes desta estratégia de aprendizado ("transdutivo" e "não-supervisionado");

- incluir mais uma dimensão de representação espacial para os dados - a coordenada "Z"; esta informação já está disponível para os dados que compõem o conjunto de dados atual porém não foi estudada nos experimentos aqui relatados; e usá-la tornará a abordagem mais aderente ao contexto real, pois movimentos envolvendo deslocamentos "para frente" e "para trás" poderão ser adicionados ao conjunto de movimentos; com esta inclusão, ter-se-á um problema mais complexo, com mais classes de movimentos;

- alterar a estratégia do *TRAdaBoost* referente a criação da distribuição de probabilidades (p^t) sobre $\mathcal{D}$, incorporando os pesos de cada exemplo de $\mathcal{D}$ na correção do erro da Rede Neural *Multilayer Perceptron*.

REFERÊNCIAS

[1] C. A. Pickering. "The search for a safer driver interface: a review of gesture recognition human machine interface". *Computing Control Engineering Journal*, vol. 16, no. 1, pp. 34 –40, feb-mar 2005.

[2] S. Mitra and T. Acharya. "Gesture Recognition: A Survey". *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Trans. on*, vol. 37, no. 3, pp. 311 –324, may 2007.

[3] M. Moni and A. B. M. S. Ali. "HMM based hand gesture recognition: A review on techniques and approaches". In *Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on*, pp. 433 –437, aug 2009.

[4] T. Sowa. "The recognition and comprehension of hand gestures: a review and research agenda". In *Proc. of the Embodied communication in humans and machines, 2nd ZiF research group international conference on Modeling communication with robots and virtual humans*, ZiF'06, pp. 38–56, Berlin, Heidelberg, 2008. Springer-Verlag.

[5] J. Liu and M. Kavakli. "A survey of speech-hand gesture recognition for the development of multimodal interfaces in computer games". In *Multimedia and Expo (ICME), 2010 IEEE International Conference on*, pp. 1564 –1569, jul 2010.

[6] R. C. B. Madeo and S. M. Peres. "Análise Automatizada de Gestos: uma Revisão Sistemática considerando Aspectos Temporais". Technical Report PPgSO-002/2012, Universidade de São Paulo, 2012.

[7] F. D. Capovilla, W. D. Raphael and M. A. C. L. *Novo Deit-Libras: Dicionário Enciclopédico Ilustrado Trilíngue da Língua de Sinais Brasileira (Libras) baseado em Linguística e Neurociências Cognitivas*. Edusp, 2010.

[8] D. B. Dias, R. C. B. Madeo, T. Rocha, H. H. Bíscaro and S. M. Peres. "Hand movement recognition for brazilian sign language: a study using distance-based neural networks". In *Proceedings of the 2009 international joint conference on Neural Networks*, IJCNN'09, pp. 2355–2362, Piscataway, NJ, USA, 2009. IEEE Press.

[9] R. C. B. Madeo, S. M. Peres, H. H. Bíscaro, D. B. Dias and C. Boscarioli. "A committee machine implementing the pattern recognition module for fingerspelling applications". In *Proc. of the 2010 ACM Symposium on Applied Computing*, SAC '10, pp. 954–958, New York, NY, USA, 2010. ACM.

[10] R. C. B. Madeo, S. M. PERES, C. A. M. Lima and C. Boscarioli. "Hybrid Architecture for Gesture Recognition: Integrating Fuzzy-Connectionist and Heuristic Classifiers using Fuzzy Syntactical Strategy". In *International Joint Conference on Neural Networks in WCCI 2012*, pp. 617–624. IEEE, 2012.

[11] S. J. Pan and Q. Yang. "A Survey on Transfer Learning". *Knowledge and Data Engineering, IEEE Transactions on*, vol. 22, no. 10, pp. 1345 –1359, oct. 2010.

[12] N. FarajiDavar, T. de Campos, J. Kittler and F. Yan. "Transductive transfer learning for action recognition in tennis games". In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pp. 1548 –1553, nov. 2011.

[13] A. Lopes, E. Santos, E. Valle Jr., J. Almeida and A. Araújo. "Transfer learning for human action recognition". In *Proceedings - 24th SIBGRAPI Conference on Graphics, Patterns and Images*, pp. 352–359, 2011.

[14] J. Liu, K. Yu, Y. Zhang and Y. Huang. "Training Conditional Random Fields Using Transfer Learning for Gesture Recognition". In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pp. 314 –323, dec. 2010.

[15] A. Farhadi, D. Forsyth and R. White. "Transfer Learning in Sign Language". In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pp. 1 –8, june 2007.

[16] I. Guyon, V. Athitsos, P. Jangyodsuk, B. Hamner and H. Escalante. "ChaLearn gesture challenge: Design and first results". In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pp. 1 –6, june 2012.

[17] L. Torrey and J. Shavlik. *Transfer Learning*, chapter Hershey, PA, USA, pp. 242–264. IGI Global, 2010.

[18] W. Dai, Q. Yang, G.-R. Xue and Y. Yu. "Self-taught clustering". In *Proceedings of the 25th international conference on Machine learning*, ICML'08, pp. 200–207, New York, NY, USA, 2008. ACM.

[19] T. Mitchell. *Machine Learning*. McGraw-Hill series in Computer Science. McGraw-Hill, 1997.

[20] V. Rao and R. Damle. "Identification and control of smart structures using neural networks: a survey". In *Decision and Control, Proceedings of the 33rd IEEE Conference on*, volume 1, pp. 91–96, 1994.

[21] W. Dai, Q. Yang, G. rong Xue and Y. Yu. "Boosting for transfer learning". In *In ICML*, 2007.

[22] MSDN. "Joint Filtering". `http://msdn.microsoft.com/en-us/library/jj131024.aspx`, 2012.

[23] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. The Morgan Kaufmann Series In Data Management Systems. Morgan Kaufmann Publishers, 2001.

[24] F. Wilcoxon. "Individual Comparisons by Ranking Methods". *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[25] A. Lorena and A. de Carvalho. "Comparing Techniques for Multiclass Classification Using Binary SVM Predictors". In *MICAI 2004: Advances in Artificial Intelligence*, edited by R. Monroy, G. Arroyo-Figueroa, L. Sucar and H. Sossa, volume 2972 of *Lecture Notes in Computer Science*, pp. 272–281. Springer Berlin / Heidelberg, 2004.

[26] P. K. Wagner, G. O. Borges, R. C. B. Madeo and S. M. Peres. "Uma Ferramenta para Construção de Conjuntos de Dados de Referência para Sistemas de Análise de Gestos Baseados em Imagens". In *Anais do VIII Simpósio Brasileiro de Sistemas de Informa??o*, pp. 607–618, São Paulo, SP, Brasil, may 2012.