



UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E
DE COMPUTAÇÃO



Aprendizagem Semissupervisionada por meio de Técnicas de *Deep Learning* e de Teoria da Informação

Bruno Vicente Alves de Lima

Orientador: Prof. Dr. Adrião Duarte Dória Neto

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e de Computação da UFRN (área de concentração: Engenharia de Computação) como parte dos requisitos para obtenção do título de Doutor em Ciências.

Número de ordem PPgEEC: D294
Natal, RN, 09 de junho de 2021

Universidade Federal do Rio Grande do Norte - UFRN
Sistema de Bibliotecas - SISBI
Catalogação de Publicação na Fonte. UFRN - Biblioteca Central Zila Mamede

Lima, Bruno Vicente Alves.

Aprendizagem semissupervisionada por meio de técnicas de Deep Learning e de Teoria da Informação / Bruno Vicente Alves de Lima. - 2021.

155 f.: il.

Tese (doutorado) - Universidade Federal do Rio Grande do Norte, Centro de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica e de Computação, Natal, RN, 2021.

Orientador: Prof. Dr. Adrião Duarte Dória Neto.

1. Semissupervisionado - Tese. 2. Deep Learning - Tese. 3. Rotulação - Tese. 4. Agrupamento - Tese. 5. Teoria da Informação - Tese. I. Dória Neto, Adrião Duarte. II. Título.

RN/UF/BCZM

CDU 004.7

*Dedico essa conquista à minha mãe
Maria Gonçalves que sempre se
dedicou a me fazer crescer como
pessoa e profissional.*

Agradecimentos

Agradeço acima de tudo a Deus, pela família, amigos e as dons e todas as conquistas adquiridas até hoje.

Agradeço imensamente à minha mãe Maria Gonçalves, por todos esses anos de dedicação, amor e confiança depositados que foram de extrema importância para a realização desta conquista, pois sem ela, nada disso seria possível.

À toda minha família por sempre acreditarem em mim e por ela nunca me permitir seguir um caminho diferente.

À minha noiva e futura esposa Elayne Figueredo, que desde quando entrou na minha vida sempre esteve no meu lado me apoiando, inclusive contribuindo com o desenvolvimento desse trabalho.

Ao meu orientador Prof. Dr. Adrião Duarte Dória Neto, por todos esses anos de orientação e paciência para me explicar algo que eu não fazia ideia de como fazer.

À Lúcia Emília, que também contribuiu com o desenvolvimento desse trabalho com seu conhecimento e pela amizade de anos.

Ao Prof. Dr. Vinicius Ponte Machado, a pessoa que me ajudou a entrar nesse mundo acadêmico.

Aos professores Dr. Ivan Nunes, Dr. Jorge Dantas e Dr. Daniel Sabino por se disponibilizarem a contribuir com esse trabalho

Ao Instituto Federal do Maranhão, por permitir meu afastamento para que eu concluísse esta tese.

E por fim, àqueles presentes à defesa desta tese.

Resumo

O crescimento expressivo de conjuntos de dados modernos, combinado à dificuldade de obter informações sobre rótulos, tornou o aprendizado semissupervisionado um dos problemas de importância prática na análise moderna de dados. Na maioria dos casos, obter conjunto de dados com a quantidade de exemplos suficientes para induzir um classificador, pode ser oneroso, pois é necessário que seja realizada uma rotulação dos dados por um especialista. Dados não rotulados são mais fáceis de serem obtidos, porém mais difíceis de serem analisados e interpretados. No problema do aprendizado semissupervisionado, têm-se uma base de dados formada por uma pequena parte rotulada e uma parte maior não rotulada, sendo possível duas vertentes: classificação semissupervisionada e agrupamento semissupervisionado. A partir disso, o objetivo deste trabalho baseia-se na aplicação de modelos que utilizam técnicas de *Deep Learning* no aprendizado semissupervisionado. Utilizando um *deep autoencoder* transformou-se os dados para um espaço de características Z e a partir disso agrupou-se e rotulou-se esses dados, com auxílio dos dados rotulados. Aplicou-se técnicas de Aprendizado por Teoria da Informação para aumentar a robustez do modelo proposto neste trabalho. Experimentos realizados apontaram a eficiência do modelo proposto em rotular e classificar dados a após o treinamento. Comparou-se também o com outros modelos clássicos de aprendizado semissupervisionado, como *co-training*, *tri-training*, *STRED* e *SEEDED K-means*, bem como outros trabalhos mais recente, mostrando a viabilidade do modelo proposto para o problema de aprendizagem semissupervisionada. E por fim, aplicou-se o modelo em um problema real na área de sensoriamento remoto e classificação de dados de *stream*.

Palavras-chave: Semissupervisionado, *Deep Learning*, Rotulação, Agrupamento, Classificação, Teoria da Informação.

Abstract

The expressive growth of modern data sets, combined with the difficulty of obtaining information about labels, has made semi-supervised learning one of the problems of practical importance in modern data analysis. In most cases, obtaining a dataset with enough examples to induce a classifier can be costly, as it is necessary to perform labeling of the data by an expert. Unlabeled data is easier to obtain but more difficult to analyze and interpret. In the semi-supervised learning problem, there is a database formed by a small labeled part and a larger unlabelled part, with two possible aspects: semi-supervised classification and semi-supervised clustering. With this, this work aims to apply models that use deep learning techniques in semi-supervised learning. Using a deep autoencoder, the data was transformed to feature space Z , and, from that, these data were clustered and labeled, with the help of the labeled data. Information Theory Learning techniques were applied to increase the robustness of the model proposed in this work. Experiments performed showed the proposed model efficiency in labeling and classifying data after training. It was also compared to other classic semi-supervised learning models, such as co-training, tri-training, STRED and SEEDED K-means, as well as other more recent works, showing the proposed model feasibility for the semi-supervised learning problem. Finally, the model was applied to a real problem in remote sensing problem and stream data classification.

Keywords: Semi-supervised, Deep Learning, Labeling, Clustering, Classification, Information Theory.

Sumário

Sumário	i
Lista de Figuras	v
Lista de Tabelas	ix
Lista de Símbolos e Abreviaturas	xi
1 Introdução	1
1.1 Contextualização	1
1.2 Motivação	3
1.3 Objetivos	4
1.4 Contribuições	4
1.5 Organização do Documento	5
2 Estado da Arte	7
2.1 Aprendizado Semissupervisionado	7
2.2 Aprendizado Semissupervisionado e Deep Learning	8
2.3 Aprendizado semissupervisionado e Sensoriamento Remoto	11
2.4 Aprendizado Semissupervisionado e dados de <i>Streaming</i>	12
2.5 Considerações	12
3 Referencial Teórico	15
3.1 Aprendizado de Máquina	15
3.1.1 Aprendizado Supervisionado	16
3.1.2 Aprendizado Não Supervisionado	17
3.2 Aprendizado Semissupervisionado (ASS)	17
3.3 Redes Neurais Artificiais	19
3.3.1 Função de Ativação	21
3.3.2 Arquitetura de uma rede neural	25
3.3.3 Processo de treinamento	26
3.3.4 Função de Custo	27
3.4 <i>Deep Learning</i>	28
3.4.1 Regularização	29
3.5 Aprendizado por Teoria da Informação	31
3.5.1 Entropia	32

3.5.2	Entropia Conjunta e Entropia Condicional	33
3.5.3	Divergência de Kullback-Leibler	34
3.5.4	Entropia Cruzada	34
3.6	Considerações	35
4	Materiais e Métodos	37
4.1	Métodos Utilizados	37
4.1.1	<i>Multilayer Perceptron</i> (MLP)	37
4.1.2	Algoritmo <i>Backpropagation</i>	38
4.1.3	Otimizador Adam	39
4.1.4	Redes Neurais Convolucionais (CNN)	39
4.1.5	Camada convolucional	41
4.1.6	Função Relu	42
4.1.7	Camada de <i>Pooling</i>	43
4.1.8	Rede Totalmente Conectada	44
4.1.9	Rede <i>Autoencoder</i> (AE)	44
4.1.10	<i>Deep Autoencoder</i>	45
4.1.11	Convolutional Autoencoder (CAE)	46
4.1.12	Algoritmo <i>K-means</i>	47
4.1.13	<i>K-means++</i>	49
4.1.14	Janelas de Parzen	50
4.2	Base de Dados Utilizadas	51
4.2.1	Bases de Benchmark	51
4.2.2	Bases de Imagens	52
4.3	Métodos de Avaliação Utilizados	54
4.3.1	Matriz de Confusão	55
4.3.2	Acurácia - A	57
4.3.3	Precisão - P	57
4.3.4	<i>Recall</i> - R	57
4.3.5	<i>F-Measure</i> - FM	57
4.3.6	Acurácia de Agrupamento	58
4.3.7	Índice Kappa	58
4.3.8	Validação Cruzada	59
4.3.9	Busca em Grade	59
4.4	Teste de kruskal-Wallis	60
4.5	Considerações	61
5	Modelo Proposto	63
5.1	Fase de Agrupamento	65
5.2	Fase de Rotulação	69
5.2.1	Definição final do rótulo de uma amostra x_i^U	72
5.2.2	Condições de Parada	73
5.3	Ajuste Fino	73
5.4	Algoritmo de Treinamento	74
5.5	Considerações	74

6	Experimentos e Resultados	77
6.1	Hiperparâmetros	79
6.2	Comparativo com modelos clássicos	83
6.2.1	Resultados da fase transdutiva com modelos clássicos	83
6.2.2	Resultados da Fase Indutiva	89
6.2.3	Análise Estatística	94
6.3	Comparativo com modelos da literatura	95
6.3.1	Resultados da fase transdutiva com modelos da literatura	95
6.3.2	Resultado da fase indutiva com modelos da literatura	101
6.3.3	Análise Estatística	106
6.4	Considerações	106
7	Estudo de Casos	109
7.1	Mapeamento de áreas agrícolas no Cerrado brasileiro	109
7.1.1	Classificação Semissupervisionada	109
7.1.2	Resultados da Classificação	112
7.2	Classificação em Dados de <i>Stream</i>	115
7.2.1	Classificação semissupervisionada de dados de <i>stream</i>	116
7.2.2	Resultados Preliminares	118
7.3	Considerações	120
8	Considerações Finais	121
8.1	Conclusões	121
8.2	Trabalhos Futuros	122
8.3	Publicações	123
Referências bibliográficas		124
A	Resultados modelos clássicos fase transdutiva	135
B	Resultados modelos clássicos fase indutiva	141
C	Resultados modelo da literatura	147

Lista de Figuras

3.1	Representação gráfica do aprendizado supervisionado	16
3.2	Representação gráfica do aprendizado não supervisionado	17
3.3	Princípio do aprendizado semissupervisionado	18
19figure.caption.15		
3.5	Modelo de Neurônio Artificial.	20
3.6	Função Degrau.	22
3.7	Função Linear.	22
3.8	Função Sigmoide.	23
3.9	Função Tangente Hiperbólica.	23
3.10	Função Relu.	24
3.11	Arquitetura <i>feedforward</i> de camada simples. Fonte: Silva et al.(2010) . .	26
3.12	Arquitetura <i>feedforward</i> de múltiplas camadas. Fonte: Silva et al.(2010) .	27
28figure.caption.32		
29figure.caption.33		
3.15	Rede Neural Profunda.	30
3.16	Rede com <i>Dropout</i>	31
38figure.caption.37		
41figure.caption.38		
43figure.caption.39		
4.4	Visualização do <i>Pooling</i>	44
4.5	Arquitetura básica de uma Rede <i>Autoencoder</i>	45
4.6	Arquitetura de uma <i>Deep Autoencoder</i>	46
4.7	Exemplo de uma rede tipo CAE	47
48figure.caption.44		
4.9	Visualização das Amostras da Base MNIST.	53
4.10	Visualização das Amostras da Base Fashin MNIST.	54
4.11	Visualização das Amostras da Base CIFAR10.	55
4.12	Visualização das Amostras da Base SLT10.	56
4.13	Representação visual da validação cruzada.	59
5.1	Fluxograma geral do treinamento do modelo proposto.	63
5.2	Representação visual do modelo proposto.	64
5.3	Estrutura do <i>autoencoder</i> utilizado no trabalho.	65
5.4	Fluxograma do treinamento do DSL.	65
5.5	Estrutura da camada de rotulação.	66

5.6	Fase agrupamento: Centroides calculado pelo k-means++ formam os pesos iniciais da camada de rotulação.	67
5.7	Representação visual de uma iteração da fase rotulação.	69
5.8	Representação dos dados rotulados influenciando para definir a rotulação dos elementos não rotulados do grupo.	70
5.9	Execução do treinamento durante a etapa de rotulação.	71
5.10	Representação do resultado da rotulação após a Fase de Rotulação.	73
 78figure.caption.73		
6.2	Estrutura do DAE utilizado nos experimentos.	79
6.3	Estrutura do CAE utilizado nos experimentos.	80
6.4	Resultado para o hiperparâmetro K com Divergência de Kullback-Leibler.	80
6.5	Resultado para o hiperparâmetro K com Entropia Cruzada.	81
6.6	Resultado para o hiperparâmetro t com divergência de <i>Kullback-Leibler</i> . .	81
6.7	Resultado para o hiperparâmetro t com entropia cruzada.	82
6.8	Classificação da Base Epilepsia na fase transdutiva.	84
6.9	Classificação da Base Reuters na fase transdutiva.	85
6.10	Classificação da Base Gás na fase transdutiva.	85
6.11	Classificação da Base Pendigits na fase transdutiva.	86
6.12	Classificação da Base MNIST na fase transdutiva.	87
6.13	Classificação da Base FASHION na fase transdutiva.	87
6.14	Classificação da Base CIFAR10 na fase transdutiva.	88
6.15	Classificação da Base SLT10 na fase transdutiva.	89
6.16	Classificação da Base Epilepsia na fase indutiva.	90
6.17	Classificação da Base Reuters na fase indutiva.	90
6.18	Classificação da Base Gás na fase indutiva.	91
6.19	Classificação da Base Pendigits na fase indutiva.	92
6.20	Classificação da Base MNIST na fase indutiva.	92
6.21	Classificação da Base FASHION na fase indutiva.	93
6.22	Classificação da Base CIFAR10 na fase indutiva.	93
6.23	Classificação da Base SLT10 na fase indutiva.	94
6.24	Rotulação Base EPILEPSIA na fase transdutiva com modelos da literatura	96
6.25	Rotulação Base REUTERS na fase transdutiva com modelos da literatura	96
6.26	Rotulação Base de GÁS na fase transdutiva com modelos da literatura .	97
6.27	Rotulação Base PENDIGITS na fase transdutiva com modelos da literatura	98
6.28	Rotulação Base MNIST na fase transdutiva com modelos da literatura .	99
6.29	Rotulação Base FASHION na fase transdutiva com modelos da literatura .	99
6.30	Rotulação Base CIFAR-10 na fase transdutiva com modelos da literatura .	100
6.31	Rotulação Base SLT-10 na fase transdutiva com modelos da literatura .	100
6.32	Classificação Base EPILEPSIA na fase indutiva com modelos da literatura	101
6.33	Classificação Base REUTERS na fase indutiva com modelos da literatura	102
6.34	Classificação Base de GÁS na fase indutiva com modelos da literatura .	102
6.35	Classificação Base PENDIGITS na fase indutiva com modelos da literatura	103
6.36	Classificação Base MNIST na fase indutiva com modelos da literatura .	104
6.37	Classificação Base FASHION na fase indutiva com modelos da literatura .	104

6.38	Classificação Base CIFAR-10 na fase indutiva com modelos da literatura	105
6.39	Classificação Base SLT-10 na fase indutiva com modelos da literatura	105
7.1	Área de Estudo do MATOPIBA	110
7.2	Área de estudo em composição RGB	111
7.3	Classes definidas no problema	112
7.4	Área do MATOPIBA classificada com o DSL	115
7.5	Mapa de classificação de um ponto da área de estudo nos anos 2013, 2015 e 2019	116
7.6	Adaptação do DSL em cada um dos tempos	117
7.7	Fluxograma de execução do DSL com dados de Stream	118
A.1	Rotulação da base EPILEPSIA na fase transdutiva.	135
A.2	Rotulação da base REUTERS na fase transdutiva.	136
A.3	Rotulação da base GAS na fase transdutiva.	136
A.4	Rotulação da base PENDIGITS na fase transdutiva.	137
A.5	Rotulação da base MNIST na fase transdutiva.	137
A.6	Rotulação da base FASHION na fase transdutiva.	138
A.7	Rotulação da base CIFAR-10 na fase transdutiva.	138
A.8	Rotulação da base SLT-10 na fase transdutiva.	139
B.1	Rotulação da base EPILEPSIA na fase Indutiva.	141
B.2	Rotulação da base REUTERS na fase Indutiva.	142
B.3	Rotulação da base GAS na fase Indutiva.	142
B.4	Rotulação da base PENDIGITS na fase Indutiva.	143
B.5	Rotulação da base MNIST na fase Indutiva.	143
B.6	Rotulação da base FASHION na fase Indutiva.	144
B.7	Rotulação da base CIFAR-10 na fase Indutiva.	144
B.8	Rotulação da base SLT-10 na fase Indutiva.	145
C.1	Rotulação da base EPILEPSIA na fase transdutiva.	147
C.2	Rotulação da base REUTERS na fase transdutiva.	148
C.3	Rotulação da base GAS na fase transdutiva.	148
C.4	Rotulação da base PENDIGITS na fase transdutiva.	149
C.5	Rotulação da base MNIST na fase transdutiva.	149
C.6	Rotulação da base FASHION na fase transdutiva.	150
C.7	Rotulação da base CIFAR-10 na fase transdutiva.	150
C.8	Rotulação da base SLT-10 na fase transdutiva.	151
C.9	Rotulação da base EPILEPSIA na fase Indutiva.	151
C.10	Rotulação da base REUTERS na fase Indutiva.	152
C.11	Rotulação da base GAS na fase Indutiva.	152
C.12	Rotulação da base PENDIGITS na fase Indutiva.	153
C.13	Rotulação da base MNIST na fase Indutiva.	153
C.14	Rotulação da base FASHION na fase Indutiva.	154
C.15	Rotulação da base CIFAR-10 na fase Indutiva.	154

Lista de Tabelas

4.1	Detalhes das bases de dados utilizadas nos experimentos	54
4.2	Exemplo para matriz de confusão para a classificação binária.	55
4.3	Exemplo para matriz de confusão para três classes.	56
4.4	Kappa Index Interpretation Table	58
5.1	Comparação das acurárias do <i>Deep Clustering</i> com o algoritmo <i>K-means</i> para a base de dados MNIST.	67
6.1	Quantidade de amostras em relação à porcentagens de dados rotulados para cada base	78
6.2	Hiperparâmetros usados nos classificadores.	83
7.1	Cores das classes no mapa de classificação	111
7.2	Resultados da rotulação na fase transdutiva	113
7.3	Resultados da classificação na fase induativa	114
7.4	Quantidade de dados em cada instante de tempo	118
7.5	Resultados da rotulação no tempo t_0	119
7.6	Resultado da rotulação no tempo t_1	119
7.7	Resultado da rotulação no tempo t_2	119
7.8	Resultado da rotulação no tempo t_3	119
7.9	Resultado da rotulação no tempo t_4	119

Lista de Símbolos e Abreviaturas

X^L	Conjunto de dados rotulados
X^U	Conjunto de dados não rotulados
AM	Aprendizagem de Máquina
ASS	Aprendizado Semissupervisionado
CAE	Convolucionar <i>Autoencoder</i>
CNN	Rede Neural Convolucional
DAE	<i>Deep Autoencoder</i>
DGL	<i>Deep Graph Learning</i>
DL	<i>Deep Learning</i>
DSL	<i>Deep Self-Labeled</i>
GAN	Redes Adversárias Generativas
IA	Inteligência Artificial
ITL	Aprendizado por Teoria da Informação
RNA	Redes Neurais Artificiais
SAE	<i>stacked autoencoders</i>
SR	Sensoriamento Remoto
TIC	Tecnologia da Informação e Comunicação

Capítulo 1

Introdução

Neste capítulo apresenta-se uma breve contextualização e motivação para o trabalho em desenvolvimento. Além disso, também estão expostos os objetivos e a estrutura deste documento.

1.1 Contextualização

Atualmente nota-se um crescimento expressivo da *web*, e cada vez mais as tecnologias da informação vem sendo utilizadas pela sociedade. Com isso, a quantidade de dados gerados e disponibilizados tem tido um crescimento exponencial nos últimos anos. Neste contexto, o aumento da necessidade de dados e informação coopera para o desenvolvimento das Tecnologias da Informação e Comunicação (TIC) viabilizando a evolução das tecnologias contribuindo para a produção cada vez maior de dados e informações.

Estudos desenvolvidos pela Consultoria EMC *Corporation*, apontam que no período 2006 a 2010, o volume de dados digitais gerados cresceu de 166 para 988 Exabytes. Existia a perspectiva que o volume de dados alcançasse a casa dos 40.000 Exabytes (40 Zettabytes ou 40 trilhões de Gigabytes) até 2020 (Gantz e Reinsel 2012). Nesse sentido, a geração e o acúmulo de dados traz consigo a necessidade de analisar e extrair suas informações, pois isso pode ajudar a explicar o que os dados significam e ajudar a prever novos dados no futuro.

As pessoas lidam todos os dias com diferentes tipos de dados que se originam de diversos ambientes, tais como, científico, industrial, saúde, educação, financeiras, imagens digitais, entre outros. Além do grande número de pessoas conectadas em redes sociais, e geram conteúdos, expõem sua opinião, postam fotos, compartilham imagens e vídeos de diversos segmentos (Tan et al. 2019) (Gupta e Parveen 2019). Esses dados, fornecem a base para uma análise posterior, tomada de decisões e entendimento de situações em que esses dados estão envolvidos (Xu e Wunsch 2009).

Neste cenário de acúmulo de dados, diversas técnicas e ferramentas computacionais têm sido desenvolvidas a fim de auxiliar o processo de análise de dados. Dentre elas destacam-se aquelas baseadas em técnicas de Aprendizagem de Máquina (AM), subárea da Inteligência Artificial (IA) que pode ser descrita como o desenvolvimento de técnicas computacionais que permitam a construção de sistemas capazes de adquirir conhecimento de forma automática (Mitchell 1997).

Técnicas de AM podem ser aplicadas em diversos problemas, tais como classificação de padrões, agrupamento, detecção de fraudes, previsão de preços de casas, auxílio de diagnóstico de doenças, entre outras. Entretanto, no atual cenário de acúmulo de dados, os modelos de AM foram sendo sobrecarregados e em muitos casos perdendo a eficiência. Na situação em que têm-se muitos dados os modelos de *Deep Learning (DL)* (sub-área da AM), que são baseados em redes neurais artificiais, tem sido o estado da arte em várias aplicações, tais como classificação de imagem, detecção de objetos, agrupamento de dados, processamento de linguagem natural, entre outros (Faceli et al. 2011).

DL lida melhor com uma quantidade muito grande de dados do que as técnicas tradicionais de aprendizado de máquina (LeCun et al. 2015). O diferencial tecnológico dessa abordagem está nos excelentes resultados obtidos em determinadas tarefas, que superam até mesmo o desempenho de especialistas, tais como, o reconhecimento de características semânticas em imagens, a vitória em jogos de estratégia, como por exemplo o AlphaGo Zero da *Google* (Silver et al. 2017) e a superação de seres humanos em testes psicométricos de compreensão verbal (Goodfellow et al. 2016).

Muitos autores subdividem AM em supervisionado e não-supervisionado. Na primeira, o conjunto de dados é formado por dados rotulados, tornando possível a tarefa de classificação, enquanto na segunda, dados não-rotulados são utilizados para extração de padrões a partir da similaridade entre eles. Portanto, o que frequentemente acontece é ter uma base de dados com muitos dados não rotulados e poucos dados rotulados. Isso torna inviável induzir um classificador de forma eficiente. Somando-se a isso, a tarefa de rotular dados com um especialista no domínio, com objetivo de induzir um classificador para generalizar o problema pode ser um trabalho dispendioso em tempo e custo (Amini e Gallinari 2003) (Basu et al. 2002). Por outro lado, a análise e interpretação dos padrões encontrados pelos algoritmos na aprendizagem não-supervisionada pode ser uma tarefa complexa para o ser humano.

Desta forma, nos últimos anos outra vertente de AM, o Aprendizado Semissupervisionado (ASS). Essa metodologia representa um intermédio entre o AM supervisionado e a não-supervisionado, utilizando dados rotulados e não rotulados para realizar o treinamento (Zhu 2005) (Faceli et al. 2011). O ASS pode ser realizado de duas formas: classificação semissupervisionada, em que um modelo preditivo treina utilizando dados rotulados e não rotulados, para rotular dados da base sem rótulos (transdutivo) e prever amostras não previstas no treinamento (indutivo); agrupamento semissupervisionado, quando dados rotulados são utilizados para agrupar os dados sem rótulos.

Combinando a geração de grandes quantidades de dados com poucos dados rotulados disponíveis, em trabalhos recentes, o aprendizado semissupervisionado foi direcionado as técnicas de Deep Learning (Ouali et al. 2020) (Kingma et al. 2014). Pois uma rede neural profunda, utilizando ASS, é capaz de lidar com a falta de dados rotulados com dados em grande escala.

Somando-se a isso, a abordagem de Aprendizado por Teoria da Informação (ITL), vem sendo cada vez mais difundido na literatura (Haykin 2001)(Principe 2010). Por exemplo, a função de ativação baseada em entropia, vem sendo bastante utilizada em Redes Neurais Artificiais (RNA). Baseado neste contexto, esse trabalho propõe um modelo semissupervisionado utilizando técnicas de *Deep Learning* combinado com aprendizado por Teoria

da Informação.

Neste trabalho, utilizou-se da capacidade de extrair características de um *deep auto-encoder* (DAE), para realizar a redução de dimensionalidade dos dados, e a partir disso realizar um agrupamento dos dados não rotulados. Para isso, adiciona-se uma camada a mais no *encoder*, que recebe os centroides iniciais como pesos, e estes serão ajustados utilizando o otimizador Adam. A partir deste agrupamento, rotula-se os elementos de cada um dos grupos, dependendo dos dados rotulados atribuídos ao grupo. A similaridade é realizada por meio de medidas não euclidianas, neste caso a divergência de *kullback-Leibler* e entropia cruzada.

1.2 Motivação

A abordagem de modelos clássicos do aprendizado supervisionado e não supervisionado, baseia-se principalmente no aspecto da quantidade de informação existente sobre o problema em questão. Escolhe-se supervisionado se houver informações descritivas acerca dos dados, ou seja, rótulos que representam a classe de uma determinada amostra pertencente a base de dados. Assim, os dados rotulados, que serão parte do conjunto de treinamento, são utilizados para treinar um modelo preditivo. O aprendizado não supervisionado é adotado quando não há informações de rótulo dos dados (Jain 2010). Neste caso, os dados são agrupados de acordo com algum critério que aumente a homogeneidade dos exemplos num mesmo grupo e aumente a heterogeneidade entre exemplos de grupos diferentes (Jain 2010).

As aplicações do mundo real na maioria das vezes ou possuem poucos dados rotulados ou nenhum. Diante disso, o aprendizado semissupervisionado torna-se uma solução mais adequada em relação ao supervisionado e não supervisionado. No de não existir nenhum dado rotulado na base, um especialista pode rotular algumas amostras, sendo possível assim a utilização do ASS.

Somando-se a isso, na literatura trabalhos recentes vem aplicando técnicas de Deep Learning ao problema do aprendizado semissupervisionado. Isso porque, como dito anteriormente, a quantidade de dados gerados vêm crescendo com o tempo, surgindo assim, grandes bases de dados que precisam ser analisadas. Segundo Sun et al. (2017), o desempenho de técnicas de Deep Learning apresentaram-se superior às técnicas tradicionais de AM, quando trabalha-se com uma grande quantidade de dados. As técnicas de DL têm sido bastante aplicadas no problema do aprendizado semissupervisionado

Então, encontrar modelos robustos capazes de rotular e classificar dados tornou-se fundamental dado as diversas aplicações reais. Diante do exposto, este trabalho tem como motivação a contribuição na resolução de problemas de classificação semissupervisionada. Para isso, utiliza-se técnicas de *Deep Learning* combinada com o aprendizado por teoria da informação, com isso, apresentando um modelo de ASS. Outro ponto importante, é a utilização de ITL, para o treinamento do modelo proposto neste trabalho. Pois além de auxiliar no treinamento, ITL é utilizado como medida de similaridade entre dados, de forma que apresenta melhor desempenho do que técnicas euclidianas.

Diante do exposto, nota-se que técnicas de *Deep Learning* trabalhando com modelo de aprendizado semissupervisionado para o problema de classificação, apoiado com aprendi-

zado por teoria da informação, tem grande potencial para inúmeras aplicações do mundo real. Essas aplicações podem ser em diversas áreas do conhecimento, tais como, na educação, saúde, sensoriamento remoto e classificação de dados de *stream*, como mostrado nos resultados deste trabalho.

1.3 Objetivos

O objetivo deste trabalho é propor um modelo semissupervisionado denominado Auto-rotulação profunda (*Deep Self-Labeled (DSL)*) baseado em técnicas de *Deep Learning* que utilizam o aprendizado por meio de Teoria da Informação. O modelo proposto (*DSL*) aborda especificamente a classificação semissupervisionada. Para isso, treina-se um *Deep Autoencoder* (*DAE*) com dados não rotulados, posteriormente extrai-se o *encoder*, adiciona-se uma nova camada e realiza-se um treinamento com o objetivo de agrupar os dados não rotulados. Os rótulos dos elementos em cada grupo são posteriormente definidos considerando os dados rotulados em seus respectivos grupos.

O treinamento do *Autoencoder* é realizando utilizando uma função de custo baseada em aprendizado por teoria da informação. Posteriormente, extrai-se o *encoder* e acopla-se uma camada a mais. Essta camada vai ser responsável por definir um grupo para cada uma das amostras do conjunto não rotulado. Os pesos dessa nova camada são inicializados com os centroides calculados com o algoritmo *K-means*, e realizado um ajuste fino com o algortimo *Backpropagation* também utilizando como função custo técnicas de ITL.

Então, utiliza-se o *encoder* com a nova camada, denominada camada de agrupamento, define um grupo para cada uma das amostras de dados rotulados. Para cada grupo, as amostras rotuladas são utilizadas para definir a classe dos elementos de seus respectivos grupos. Essa definição de classe é baseada também em técnicas de ITL para definir a similaridade entre as amostras. As medidas de teoria da informação não fazem suposições acerca dos dados, pois essas medidas são calculadas em função dos próprios dados. Assim, o modelo proposto torna-se mais robusto, sendo possível de aplicá-lo em diferentes conjuntos de dados.

1.4 Contribuições

Como contribuição desse trabalho, é proposta para auto-rotulação semissupervisionada capaz treinar um modelo preditivo e realizar classificação semissupervisionada usando um pequeno conjunto de dados rotulados e um grande conjunto de dados sem rótulos compatível com outros modelos semissupervisionados auto-rotulados. Podemos apontar também, como contribuição deste trabalho, um estudo empírico realizado detalhadamente, em diversos contextos de aplicações, utilizando bases da literatura e bases reais.

Outra contribuição deste trabalho, é no campo do Sensoriamento Remoto (SR), com a aplicação do modelo proposto, ao mapeamento de atividades agrícolas em uma região específica do nordeste brasileiro. Neste caso, existe a necessidade de se obter um conjunto de dados (pixels) rotulados, para gerar um mapa de classificação. Entretanto, existe uma

dificuldade em coletar as amostras rotuladas. Com isso, aplicando o modelo proposto, obteve-se os mapas de classificação da região estudada.

Também ainda como contribuição, apresenta-se a aplicação do modelo proposto no contexto de data *streaming*. Realizou-se experimentos com bases de dados da literatura, para simular um ambiente onde dados chegam em instantes de tempo, no caso, dados rotulados e não rotulados. Adaptou-se o modelo proposto para esse problema, tendo resultados mostrando-se promissores nessa área.

1.5 Organização do Documento

O restante do documento está organizado em 7 capítulos, são eles:

- Capítulo 2: apresenta trabalhos relacionados à aprendizado semissupervisionado, *Deep Learning* e Teoria da Informação e a relação entre estes trabalhos e o modelo proposto.
- Capítulo 3: descreve os conceitos de Aprendizado de Máquina: supervisionado e não supervisionado, Aprendizado Semissupervisionado, Deep Learning e Teoria da Informação e aprendizado por teoria da informação.
- Capítulo 4: são apresentadas as bases de dados utilizadas nos experimentos desse trabalho, as métricas de avaliação e métodos utilizados.
- Capítulo 5: apresenta em detalhe a proposta desse trabalho, onde é apresentado e mostrado todo o funcionamento do modelo proposto.
- Capítulo 6: descreve todos os experimentos realizados para validar o modelo proposto e os respectivos resultados obtidos.
- Capítulo 7: mostra resultados da aplicação do modelo proposto em um problema de classificação dentro do contexto de sensoriamento remoto e resultados preliminares de experimentos realizados visando mostrar a viabilidade do modelo proposto no contexto de dados de *streaming*.
- Capítulo 8: e finalmente são feitas conclusões sobre o trabalho desenvolvido e ações que possam melhorar o modelo proposto.

Capítulo 2

Estado da Arte

Inúmeros trabalhos sobre aprendizado semissupervisionado e *Deep Learning* vem sendo desenvolvidos nos últimos anos. Neste contexto são apresentados alguns trabalhos que abordam aprendizado semissupervisionado, Agrupamento Profundo e aprendizado por Teoria da Informação. Além disso, alguns trabalhos recentes no contexto de sensoriamento remoto e dados de *streaming*.

2.1 Aprendizado Semissupervisionado

No aprendizado semissupervisionado o primeiro modelo apresentado foi o algoritmo *self-training* (Schwenker e Trentin 2014). Considerando que tem-se um conjunto de dados dividido em rotulados (X^L) e não rotulados (X^U), sendo X^U em maior quantidade em relação à X^L , no algoritmo *self-training*, um classificador é treinado inicialmente com a parte de dados rotuladas (X^L), após isso, classifica-se (rotulação) a parte de dados não rotulada (X^U). Definida uma forma de se verificar se a classificação de uma determinada amostra foi rotulada com grau de certeza, esta é adicionada aos dados rotulados, caso contrário, volta ao conjunto dos não rotulados. Esse processo é realizado iterativamente por algumas épocas até que todo o conjunto de dados esteja rotulado. O processo executado no *self-training* é denominado auto-rotulação, pois os dados são rotulados durante o processo de treinamento (Zhu 2005).

Como evolução do algoritmo *self-training*, Blum e Mitchell (1998) propuseram o algoritmo *Co-training*. Neste caso, utiliza-se a mesma abordagem do algoritmo *self-training*, mas desta vez, utilizando dois classificadores, em que os dois cooperam entre si para definir o rótulo dos dados não rotulados. O algoritmo *co-training* baseia-se na teoria de duas visões, dividindo os dados em dois subconjuntos e cada um contendo apenas metade da entrada do conjunto de dados. O processo é repetido por algumas épocas até atingir uma condição de parada, que pode ser o número de épocas, a rotulação total de X^U .

Além dos métodos clássicos *self-training* e *Co-training*, destacam-se os outros métodos de auto-rotulação, tais como, *Tri-training* (Zhou e Li 2005) e o STRED (Li e Zhou 2005). O primeiro, utiliza mesma metodologia adotada em *co-training*, entretanto, utiliza três classificadores. O STRED utiliza um método específico de edição de dados para identificar e remover os exemplos incorretos dos dados auto-rotulados. Em cada

iteração do processo de autotreinamento, a estatística local do peso da aresta de corte é usada para ajudar a estimar se um exemplo recentemente rotulado é confiável ou não, e apenas os exemplos auto-rotulados confiáveis são usados para aumentar o conjunto de treinamento rotulado.

Outro algoritmo clássico de aprendizado semissupervisionado é o *Label Propagation* (Zhu e Ghahramani 2002), que é um algoritmo iterativo onde atribui-se rótulos a pontos não rotulados, propagando rótulos através do conjunto de dados. O *Label Propagation* trabalha com aprendizagem transdutiva, que é quando define-se os rótulos dos pontos de dados não rotulados a partir dos rótulos que já estão disponíveis.

Além dos métodos clássicos, outro método baseado em aprendizado não supervisionado pode ser destacado como processo de auto-rotulação, o *SEEDED K-means* (Basu et al. 2002). Este algoritmo é similar ao *k-means* padrão, entretanto, não seleciona as sementes iniciais de forma aleatória. No *SEEDED K-means* os dados rotulados são utilizados como sementes iniciais. Desta forma cria-se um grupo para cada classe do problema.

2.2 Aprendizado Semissupervisionado e Deep Learning

Com a evolução dos problemas e o acumulo de dados destacado no Capítulo 1, cada vez mais trabalhos foram surgindo com técnicas de DL adotando o aprendizado semissupervisionado.

Uma abordagem de auto-rotulação foi proposta por Tanha et al. (2017), que consiste no *self-training* utilizando uma árvore de decisão. Essa abordagem usa a proporção de exemplos de treinamento rotulados de cada classe incluídos no nó folha como medida de confiança do exemplo classificado. Por exemplo, se 80% dos exemplos de treinamento rotulados são da classe A e os outros 20% são exemplos da classe B em um nó, a confiança na classificação dos exemplos no nó é de 80%. Se o número de amostras rotuladas nos nós for grande o suficiente, define-se como boa confiança. No entanto, como o *self-training* geralmente é aplicado para condicionar que o número de amostras rotuladas seja pequeno e o número de amostras rotuladas correspondentes a cada nó da folha seja extremamente menor, uma abordagem de *self-training* baseada em árvore de decisão tem problemas para que a seja menos confiável.

Em Lee et al. (2017) foi apresentada uma abordagem baseada em rede neural profunda para auto-treinamento, combinando pré-treinamento, abandono e esquecimento de erros. Primeiramente, um classificador de base é generalizado para melhorar o desempenho do autotreinamento com poucos dados de treinamento inicial. Para melhorar o desempenho da generalização dos classificadores, os autores utilizam dados não rotulados no treinamento a fim de aprender as informações latentes nos dados. Embora as redes neurais profundas sejam pré-treinadas, existe a possibilidade de adaptação excessiva aos dados iniciais rotulados. Se a quantidade de dados de treinamento rotulados iniciais for muito pequena, o classificador treinado tende a aprender detalhes desnecessários, como ruídos de observação. Para evitar este problema, o modelo utiliza *dropout* como método de regularização.

Piroonsup e Sinthupinyo (2018) propuseram um método para determinar a suficiência dos dados rotulados e dois métodos para melhorar o conjunto de dados rotulados na parte

insuficiente. Para determinar a suficiência dos dados rotulados, aplicou-se uma técnica de agrupamento semissupervisionado para estimar a distribuição dos dados rotulados no conjunto de treinamento. O objetivo é resolver o problema de casos em que, adicionando dados não rotulados ao treinamento, a precisão da classificação é degradada no processo.

Do-Omri et al. (2018) apresentaram um método de meta-aprendizado de auto treinamento que pode ser aplicado às Redes Adversárias Generativas (GAN) que usam os dados gerados para aumentar o desempenho da classificação. Baseou-se no GAN Melhorado (Salimans et al. 2016), pois ele já possui mecanismos para lidar com o aprendizado semissupervisionado.

Em Li e Yeh (2019), um modelo de aprendizagem semissupervisionado baseado em um Convolucional Autoencoder (CAE) e uma Rede Neural Convolucional (CNN) complementar foram empregados para apoiar a classificação de imagens. A abordagem aprende automaticamente os recursos de um modelo CAE pré-treinado, o que reduz a incorporação de recursos extremamente grande extraída de camadas profundas. A incorporação de recursos extraídos pelo modelo CAE substitui os dados da imagem original como a entrada CNN. A partir daí, é empregado para treinar o modelo de classificação de imagens.

Sahito et al. (2019) explorou um novo método de treinamento para aprendizado semissupervisionado baseado no aprendizado de função de similaridade usando uma rede siamesa para obter uma incorporação adequada. As representações aprendidas são discriminativas no espaço euclidiano. Portanto, eles podem ser empregados para rotular instâncias não rotuladas usando um classificador de vizinho mais próximo. Previsões confiáveis de instâncias não rotuladas são usadas como rótulos verdadeiros para retreinar a rede siamesa no conjunto de treinamento expandido; este processo é aplicado iterativamente.

Huang e Deng (2019) propôs um método de classificação de representação esparsa generalizada, que era essencialmente uma classificação baseada em representação esparsa de Euler para classificação de imagens. Sua proposta era um método de classificação baseado em representação colaborativa generalizada para problemas de classificação de amostras insuficientes.

Li, Zhu e Wu (2019) propuseram um novo método de auto-treinamento baseado em picos de densidade e um filtro local de ruído estendido sem parâmetros. O objetivo é resolver os problemas dos filtros de ruído locais existentes usados nos métodos de autotreinamento. Estes filtros enfrentam os seguintes defeitos técnicos: dependência de parâmetros e uso apenas de dados rotulados para remover amostras erradas.

Em Iscen et al. (2019) utiliza-se um método de propagação de rótulo transdutivo baseado em múltiplas suposições para fazer previsões em todo o conjunto de dados e usar essas previsões para gerar pseudo-rótulos para os dados não rotulados e treinar uma rede neural profunda. No centro do método transdutivo está um gráfico do vizinho mais próximo do conjunto de dados criado com base nos *embeddings* da mesma rede. Portanto, este processo de aprendizagem é composto por essas duas etapas. O desempenho foi melhorado em vários conjuntos de dados, especialmente no regime de poucos rótulos.

Livieris (2019) propôs um alternativa aos métodos de classificação tradicionais em forma de comitê. Neste trabalho, abordou-se os métodos tradicionais Self-training, Co-Training e Tri-Training em forma de comitê, baseado em esquema de votação de máxima

probabilidade.

Li et al. (2020) propôs uma estrutura semissupervisionada para auto-rotulação eficaz com base em núcleos locais com o objetivo de resolver o problema da quantidade inadequada de dados inicialmente etiquetados em métodos de auto-rotulação. Os principais conceitos deste trabalho incluem dois aspectos: a) dados inicialmente rotulados inadequados são melhorados pela adição de núcleos locais previstos por meio de rotulagem ativa ou co-rotulagem; b) o modelo usa qualquer método semissupervisionado de auto-rotulação para treinar um determinado classificador em dados rotulados aprimorados e dados não rotulados atualizados. Neste modelo é revelada parcialmente a distribuição de dados, tornando a modelo proposto aplicável a conjuntos de dados esféricos ou não esféricos. Além disso, os núcleos locais também apoiam o modelo para melhorar efetivamente a quantidade insuficiente de dados inicialmente rotulados, usando uma quantidade consideravelmente limitada de dados inicialmente rotulados.

Śmieja et al. (2020) introduziu uma estrutura de rede neural para agrupamento semissupervisionado com restrições de pares (*must-link* ou *cannot-link*) para melhorar a classificação. Em contraste com as abordagens existentes, este método decompõe o agrupamento semissupervisionado em duas tarefas de classificação mais simples. O primeiro estágio usa pares de redes neurais siamesas para rotular os pares de pontos como *must-link* ou *cannot-link*. O segundo estágio emprega o conjunto de dados rotulado por pares completo produzido pelo primeiro estágio em um método de agrupamento baseado em rede neural supervisionada. A abordagem de Śmieja et al. (2020) é baseada na observação de que as classificações binárias (por exemplo, atribuição de relações de pares) são geralmente mais fáceis do que o agrupamento multi-classe com supervisão parcial.

Liu et al. (2020) propuseram uma abordagem semissupervisionada combinando aprendizagem de representação não supervisionada e classificação supervisionada para melhorar a classificação semissupervisionada de dados de imagem de ressonância magnética funcional. O modelo de aprendizagem de representação não supervisionada foi construído utilizando uma Máquina de Boltzmann Restrita, e os parâmetros de características do modelo não supervisionado foram fornecidos como restrições no treinamento semissupervisionado do modelo de classificação.

Lin et al. (2020) apresentaram um modelo denominado *deep graph learning* (DGL). O objetivo desse trabalho é construir redes de aprendizado de grafos profundos para capturar dinamicamente o gráfico global por aprendizado de métrica de similaridade e gráfico local por aprendizado de atenção.

Fu et al. (2021) propuseram a utilização de redes convolucionais de aprendizagem de gráfico dinâmico para classificação semissupervisionada. Os objetivos deste método são otimizar constantemente as informações estruturais da amostra durante o processo de treinamento da rede convolucional de aprendizagem de grafos dinâmicos com o aprendizado semissupervisionado.

2.3 Aprendizado semissupervisionado e Sensoriamento Remoto

Na literatura encontra-se também trabalhos de aprendizado semissupervisionado aplicado ao sensoriamento remoto. Em Zhou et al. (2019) é apresentado um novo framework *deep learning*, nomeado, *semisupervised stacked autoencoders* com co-training, para classificação em sensoriamento remoto. Primeiro, dois *stacked autoencoders* (SAE) são pré-treinados com base nas características hiperespectrais e nas características espaciais, respectivamente. Em segundo lugar, o ajuste fino é conduzido alternativamente para os dois SAEs em um modo de co-treinamento semissupervisionado, onde o conjunto de treinamento inicial é ampliado projetando um método eficaz de cultivo da região. Finalmente, as probabilidades de classificação obtidas pelos dois SAEs são fundidas usando um modelo de campo aleatório de Markov resolvido por modos condicionais iterados.

Em Aydav e Minz (2019) é proposta uma versão aprimorada do método *self-training* tradicional, que utiliza propriedades espaciais para uma seleção confiável da amostra. A técnica de autotreinamento é modificada para as imagens de sensoriamento remoto que usam a ajuda de pixels de vizinhança espacial para adicionar amostras rotuladas para o processo de treinamento iterativo.

Em Kothari e Meher (2020), propuseram um modelo de classificação semissupervisionado de autoaprendizagem usando uma rede neural como classificador base. O modelo usa um método de aprendizagem de informação de vizinhança eficiente para a rede neural superar os deméritos de abordagens convencionais existentes. Usando duas abordagens diferentes, Kothari e Meher (2020) apresentaram a geração de matrizes de similaridade para extrair informações de vizinhança que eventualmente melhorem o processo de aprendizagem da rede neural. O primeiro método considera informações mútuas de vizinhança e o segundo método usa o mapa de classes de amostras não rotuladas. Os rótulos de classe das amostras não rotuladas são previstos por um classificador, ou seja, treinados com as amostras rotuladas disponíveis. Finalmente, a informação de vizinhança colaborativa é derivada dessas duas matrizes e usada para o desenvolvimento do modelo de classificação semissupervisionado proposto.

Outro trabalho nessa linha, é o de Wu et al. (2020), que propôs um novo modelo de classificação de imagem hiperespectral semissupervisionada que utiliza *self-training* para atribuir gradualmente pseudo rótulos altamente confiáveis a amostras não rotuladas por agrupamento e emprega restrições espaciais para regular o processo de *self-training*. Restrições espaciais são introduzidas para explorar a consistência espacial dentro da imagem para corrigir e reatribuir os pseudo rótulos erroneamente classificados. Por meio do processo de *self-training*, os pontos de amostra de alta confiança aumentam gradativamente e são adicionados às classes semânticas correspondentes, o que aumenta gradativamente as restrições semânticas. Ao mesmo tempo, o aumento em pseudo rótulos de alta confiança também contribui para a consistência regional dentro das imagens hiperespectrais, o que destaca o papel das restrições espaciais e melhora a eficiência do HSIC.

2.4 Aprendizado Semissupervisionado e dados de *Streaming*

Outra linha, do qual o modelo proposto pode ser aplicado é na classificação de dados de *streaming*. No trabalho de Li, Wang, Liu, Bi, Jiang e Sun (2019), propuseram uma nova abordagem de aprendizado semissupervisionado incremental em *streaming* de dados. Cada camada do modelo é composta por uma rede gerativa, uma estrutura discriminante e a ponte. A rede gerativa usa aprendizado de recurso dinâmico baseado em *autoencoders* para aprender recursos gerativos a partir de dados de *streaming*, o que tem demonstrado seu potencial no aprendizado de representações de recursos latentes. Além disso, a estrutura discriminante regulariza a construção da rede por meio da construção de restrições de similaridade e dissimilaridade aos pares.

Le Nguyen et al. (2019) apresentaram duas novas abordagens para lidar com rótulos ausentes para aprendizagem de classificação em fluxos de dados, nomeadamente *cluster-and-label*, método que se baseia em um algoritmo de agrupamento para agrupar instâncias por rótulos de classe de uma maneira não supervisionada, combinado com um esquema de votação para selecionar o rótulo mais provável para uma instância de dados e *self-training*, que é um "aluno treinado" com suas previsões de dados não rotulados para se reforçar com os erros anteriores.

Em Din et al. (2020), apresentaram um novo algoritmo de aprendizagem semissupervisionado online modelando desvios de conceito com um conjunto de microclusters. Esses microclusters são mantidos dinamicamente para capturar os conceitos em evolução com aprendizagem representativa baseada em erros. Dessa forma, os desvios de conceito locais são capturados mais rapidamente e, finalmente, dão suporte ao aprendizado de fluxo de dados eficaz.

Zhu e Li (2020) considerando o problema que em dados de *streaming*, a parte não rotulada pode existir classes que não estão previstas nos rotulados. Abordando essa questão, Zhu e Li (2020) propuseram o SEEN, que consiste em três componentes principais: um novo detector de classe eficaz baseado em agrupamento de árvores aleatórias, um classificador robusto para previsões nas classes conhecidas e um processo de atualização eficiente que garante que todo o *framework* se adapte o ambiente em mudança automaticamente. O classificador produz rótulos conhecidos por meio da propagação de rótulos que utiliza todos os dados rotulados e parcialmente não rotulados no passado, que naturalmente descrevem todo o fluxo visto até agora.

2.5 Considerações

Este capítulo apresentou uma série de trabalhos tradicionais no contexto de aprendizado semissupervisionado. Posteriormente, listou um conjunto de trabalhos que empregam o aprendizado semissupervisionado no problema de Auto-rotulação utilizando técnicas de *deep learning*. Foram apresentados também, alguns trabalhos relacionados à sensoriamento remoto e dados de *streaming*, com aprendizado semissupervisionado.

Diante dos trabalhos expostos, nota-se a relevância das técnicas de DL no contexto da

aprendizagem semissupervisionada. Considerando os trabalhos citados, assume-se que o problema da auto-rotulação continua relevante. Diversas técnicas podem ser utilizadas para resolver este problema, incluindo o uso de técnicas de *deep learning*, conforme demonstrado neste trabalho.

Capítulo 3

Referencial Teórico

Neste capítulo apresenta-se uma fundamentação teórica sobre os conceitos de Aprendizado de Máquina (AM). Neste trabalho, aplica-se técnicas de AM, especificamente *Deep Learning* para resolver o problema do Aprendizado Semissupervisionado, baseado no aprendizado por Teoria da Informação. As técnicas de *Deep Learning* permitem trabalhar com uma enorme quantidade de dados, extraíndo características, facilitando a tarefa no aprendizado semissupervisionado que trabalha com dados rotulados e não rotulados.

3.1 Aprendizado de Máquina

Geralmente problemas computacionais são resolvidos por meio de algoritmos, que são passos definidos para resolver um determinado problema. Entretanto, algumas tarefas, devido sua complexidade, tornam difícil definir um algoritmo para serem resolvidos, tais como, por exemplo, reconhecimento facial. Neste exemplo, é extremamente difícil definir quais características considerar, pois existem inúmeros formatos de rostos, inúmeras expressões faciais de uma mesma pessoa, dentre outras características e alterações que dificultam a modelagem de um algoritmo para a tarefa de reconhecimento facial (Mitchell 1997) (Faceli et al. 2011). Para esse tipo de problema como reconhecimento facial, têm sido bastante abordado técnicas de Aprendizado de Máquina (AM) que vem apresentando desempenho satisfatório para esse tipo de problema.

Nos últimos anos, inúmeros problemas de alta complexidade vem surgindo para serem resolvidos de forma computacional e uma grande geração de dados por diferentes setores, tais como, economia, indústria, saúde, educação, etc... Para trabalhar com esses problemas e grande quantidade de dados, necessita-se técnicas computacionais capazes de aprender a partir de experiências passadas, uma hipótese, ou função, capaz de generalizar o problema. Isso pode ser definido por um processo de indução de uma hipótese (ou aproximação de função) a partir de experiência passada. Esse processo dá-se o nome de Aprendizado de Máquina (AM) (Mitchell 1997) (Faceli et al. 2011).

Na literatura podem-se encontrar diversas definições para Aprendizado de Máquina. Uma delas definida por Mitchell (1997), apresenta AM como: "*A Capacidade de melhorar o desempenho na realização de alguma tarefa por meio da experiência passada.*". AM é uma sub-área da Inteligência artificial que é associada também a outras áreas de pesquisa, tais como, Probabilidade e Estatística, Teoria da Computação, Neurociência,

Teoria da Informação, entre outras.

Assim dentre o vasto campo científico que pode ser aplicada, a AM conforme destacado abaixo está também relacionada ao sucesso de técnicas aplicadas ao reconhecimento facial, reconhecimento de fala, detecção de fraude de cartão de crédito, em jogos como xadrez e dama, auxílio no diagnóstico de doenças, tais como câncer, entre outras aplicações. Muitos autores dividem AM em: supervisionado, não supervisionado, por reforço e semissupervisionado (Rezende 2003). Esses tipos de aprendizados serão descritos nas Seções a seguir.

3.1.1 Aprendizado Supervisionado

O aprendizado supervisionado é determinado por métodos que são capazes de, a partir de um conjunto de treinamento (dados), modelar padrões existentes. Um elemento pertencente ao conjunto de treinamento é a união de pares de valores formado por um conjunto de atributos (características) e uma classe (rótulo). A quantidade de atributos define a dimensão do problema que vai ser trabalhado e seus valores serão as entradas de um modelo no processo de aprendizado (Mitchell 1997).

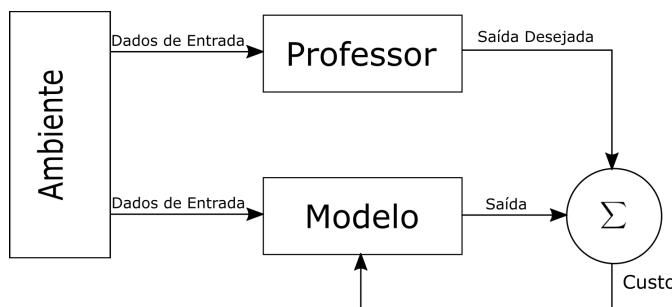


Figura 3.1: Representação gráfica do aprendizado supervisionado

A Figura 3.1 apresenta o diagrama representativo do aprendizado supervisionado. Pode-se observar o modelo de aprendizagem, que recebe uma entrada e emite uma saída. Para auxiliar, pode-se observar um professor que apresenta a saída desejada para comparar com saída emitida para definir um erro, que é utilizado pelo modelo para ajustar o aprendizado.

No Aprendizado supervisionado têm um conjunto de exemplos X , onde X é definido na Equação 3.1.

$$X = \{x_i, y_i\}_{i=1}^n \quad (3.1)$$

Onde, $x_i = [x_1, x_2, x_3, \dots, x_n]$ é o vetor de atributos que representa cada amostra do conjunto de exemplos e y_i representa a resposta esperada referente a cada amostra de X . E n é o tamanho do conjunto de exemplos, ou seja, a quantidade de elementos (Barber 2012).

O aprendizado supervisionado preconiza que, para cada conjunto de valores de entrada, existe um respectivo conjunto de respostas (saídas) que são apresentadas durante a aprendizado. Dessa forma, esse problema pode ser modelado como um problema de

classificação ou regressão, e esses conjuntos de entradas e saídas são utilizados durante as etapas de treinamento e teste do classificador (Faceli et al. 2011).

O objetivo é induzir um mapeamento geral dos vetores de entrada x para os valores y . Desta forma, o sistema de aprendizado gera um modelo $y = f(x)$, sendo f uma função desconhecida que permite predizer futuros valores y para amostras não conhecidas.

3.1.2 Aprendizado Não Supervisionado

O aprendizado não supervisionado possui um conjunto de entradas, e diferente do supervisionado, não existe conhecimento das saídas referente a cada exemplo. Os métodos de aprendizado não-supervisionado tentam detectar padrões existentes nos dados com o objetivo de representá-los de forma resumida. Desta forma, o problema consiste em agrupar diversas amostras em grupos distintos, de tal forma que haja um alto grau de similaridade entre os elementos de um mesmo grupo e o grau de dissimilaridade entre os grupos distintos seja o maior possível.



Figura 3.2: Representação gráfica do aprendizado não supervisionado

A Figura 3.2, apresenta a representação visual do aprendizado não supervisionado. Diferente do supervisionado (Figura 3.1) não existe um professor auxiliando. O Aprendizado é definido a partir das entradas, que representam o estado do ambiente. Desta forma o modelo é capaz de realizar o aprendizado sem auxílio das saídas desejada (professor).

No aprendizado não supervisionado têm-se um conjunto de exemplos X , descrito na Equação 3.2. Onde o vetor $x_i = [x_1, x_2, x_3, \dots, x_n]$, representa o vetor de atributos de cada amostra do conjunto X sem a informação sobre a classe y_i , como no aprendizado supervisionado (Equação 3.1) (Barber 2012).

$$X = \{x_i\}_{i=1}^n \quad (3.2)$$

O objetivo é construir um modelo capaz de encontrar padrões nas amostras, formando grupos de elementos com base em suas características similares. Desta forma, para um conjunto de dados X , formado por vetores x_1, \dots, x_n , deve-se encontrar uma similaridade entre os dados de modo que os grupos sejam formados pelos elementos mais similares possível, definindo um conjunto $G = \{g_i\}_{i=1}^n$ que represente as classes das amostras (Barber 2012).

3.2 Aprendizado Semissupervisionado (ASS)

O aprendizado semissupervisionado (ASS) está entre o aprendizado supervisionado e a não supervisionado. Além de dados não rotulados, o algoritmo é fornecido com algumas informações de supervisão - mas não necessariamente para todos os exemplos. No ASS, dado um conjunto de dados $X = \{X_i^L, X_i^U\} \in \mathbb{R}^n$, sendo $X^L = \{x_i, y_i\}_{i=1}^L$ dados rotulados

e $X^U = \{x_i\}_{i=1}^U$ dados não rotulados, onde $U \gg L$, x_i uma amostra qualquer de X e y_i o rótulo (Zhu 2005) (Zhu e Goldberg 2009).

Na Figura 3.3 é apresentado a representação do princípio do aprendizado semissupervisionado. Um modelo (por exemplo, classificador) é primeiro treinado com poucos dados de treinamento rotulados disponíveis. Este modelo é então usado para classificar e, assim, rotular os muitos dados não rotulados (os dados de teste) disponíveis. Os dados recém-rotulados são combinados com os rotulados originalmente disponíveis para retreinar o modelo com muito mais dados e, assim, obter um modelo melhor.

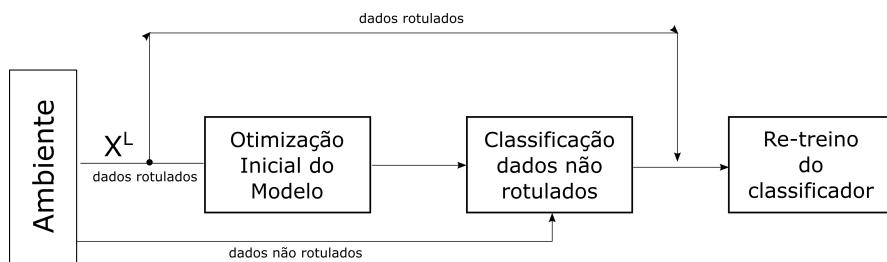


Figura 3.3: Princípio do aprendizado semissupervisionado

O objetivo é encontrar uma função f não linear parametrizada capaz de generalizar o problema Semissupervisionado e encontrar o conjunto $\{y_i\}_{i=1}^U$ para os dados não rotulados X^U a partir do conjunto rotulado X^L (Zhu 2005) (Zhu e Goldberg 2009).

À primeira vista, pode parecer paradoxal que se possa aprender qualquer coisa sobre um modelo preditivo $f : X \rightarrow y$ a partir de dados não rotulados. Afinal, f é sobre o mapeamento da instância x para o rótulo y , embora os dados não rotulados não fornecam nenhum exemplo de tal mapeamento. A resposta está nas suposições que se faz sobre a ligação entre a distribuição de dados não rotulados $P(x)$ e o rótulo de destino.

O aprendizado semissupervisionado pode ser descrito de duas formas:

- **Classificação Semissupervisionada** - também conhecida como classificação com dados rotulados e não rotulados (ou parcialmente rotulados), é uma extensão do problema de classificação supervisionada. O treinamento considera ambos os dados, rotulados (X^L) e não rotulados (X^U). Tipicamente assume-se que o $U \gg L$. Nessa classificação, tem-se a rotulação do conjunto U , durante o treinamento, como é representado na Figura 3.4, e a possibilidade de classificação de amostras não previstas no processo de aprendizado (Zhu e Goldberg 2009).
- **Agrupamento Semissupervisionado** - neste caso, o agrupamento se dá a partir de informações prévias sobre os dados. Uma possibilidade, é utilizar o agrupamento por sementes, onde têm um conjunto L (rotulados) que será utilizado como centroides para agrupamento do conjunto U . Existe também, a versão que utiliza agrupamento por restrições, que podem ser *must-link*, onde duas instâncias x_i e x_j devem estar no mesmo grupo, *cannot-link*, que define que duas instâncias x_i e x_j não podem estar no mesmo grupo. O objetivo do agrupamento com restrição é obter um melhor particionamento do que apenas utilizando de dados não rotulados (Zhu e Goldberg 2009).

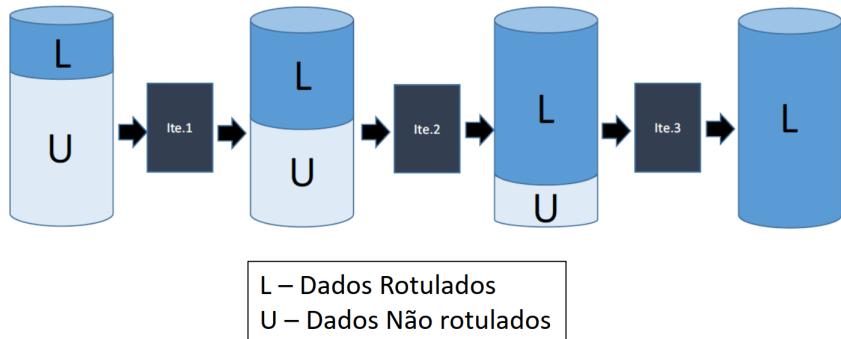


Figura 3.4: Visualização da base de dados na classificação semissupervisionada. Fonte: Lima (2015)

Existem outras configurações de aprendizado semissupervisionado, incluindo regressão com dados rotulados e não rotulados, redução de dimensionalidade com instâncias rotuladas cuja representação reduzida de recurso é fornecida e assim por diante. Este trabalho se concentra na classificação semissupervisionada. Existem também, duas configurações de aprendizagem semi-supervisionada, são elas:

- **Aprendizado semissupervisionado Transdutivo:** Nesse tipo de ASS, têm-se um conjunto de dados X^L (rotulados) e X^U não rotulados. Sabemos que o treinamento, utiliza-se ambos os tipos de dados. No ASS Transdutivo, o modelo ao mesmo tempo que treina, rotula o conjunto X^U , de modo que, ao final do treinamento, têm-se além de um modelo treinado, um conjunto de dados totalmente rotulado (Zhu e Goldberg 2009).
- **Aprendizado semissupervisionado Indutivo:** Nesse tipo, aplica-se o modelo de ASS para classificar amostras não previstas nos conjuntos X^U e X^L , como realizado nos modelos de aprendizado supervisionado.

3.3 Redes Neurais Artificiais

Nos últimos anos *Deep Learning* (DL), uma sub área específica de Aprendizado de Máquina, vem demonstrando destaque na resolução de problemas em diversas áreas, como por exemplo na área de visão computacional. Isso se deve ao fato de surgir bases de dados com milhões de imagens e de computadores capazes processar tais bases de dados (Russakovsky et al. 2014) (Deng et al. 2009). Estas técnicas de DL são baseadas nas redes neurais artificiais.

Os trabalhos iniciais de redes neurais artificiais (RNA) publicados a mais de 50 anos, apenas só a partir dos anos 90 estas técnicas começaram a ser utilizadas mais fortemente (da Silva et al. 2010). Redes neurais artificiais podem ter diversas aplicações, tais como, classificação de padrões, reconhecimento de escrita e voz, reconhecimento facial, identificação de anomalias em imagens médicas, entre outras.

Redes Neurais Artificiais (RNA) são modelos matemáticos inspirados no funcionamento do cérebro humano. São definidas como um conjunto de unidades de processamento

mento chamadas de Neurônios Artificiais. Estes neurônios são conectados entre si por um grande número de interconexões, onde cada conexão possui um peso (da Silva et al. 2010).

O primeiro trabalho de neurocomputação foi publicado em 1943 por McCulloch e Pitts (1943), onde os autores apresentaram o primeiro modelo matemático inspirado no neurônio biológico, resultando assim no conceito de neurônio artificial. Em Hebb (1949) foi proposto o primeiro método de treinamento RNA, denominado de regra de aprendizado de Hebb, sendo baseado em hipóteses e observações de caráter neurofisiológico. Os neurônios artificiais são a base das RNA's, pois elas são formadas por um conjunto de camadas de neurônios artificiais (da Silva et al. 2010). O modelo de neurônio artificial mais simples foi proposto por (McCulloch e Pitts 1988), está representado na Figura 3.5.

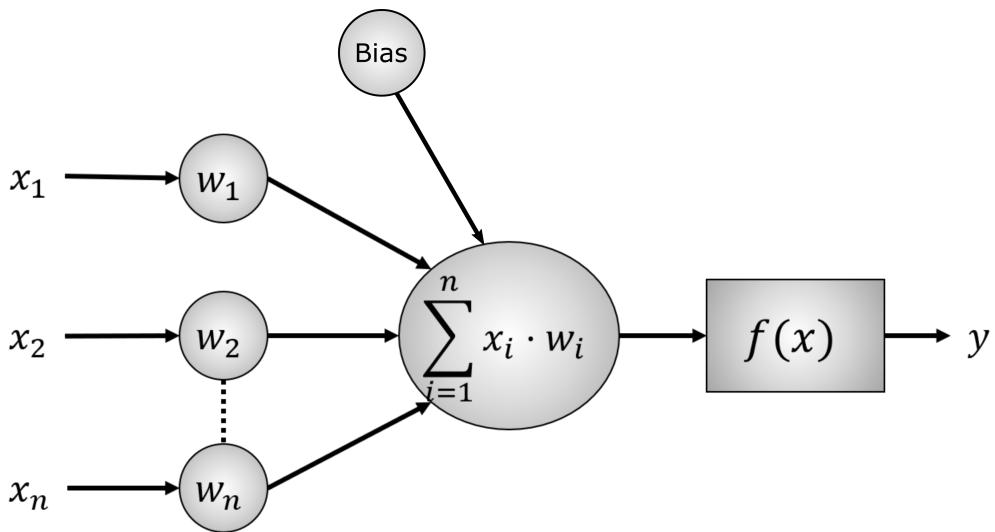


Figura 3.5: Modelo de Neurônio Artificial.

Os neurônios artificiais utilizados nos modelos de redes neurais artificiais são não-lineares e fornecem saídas contínuas, e realizam funções simples, como coletar os sinais existentes em suas entradas, aplicar à uma função de operacional (somatório) e emitir uma saída de acordo com uma função de ativação inerente (da Silva et al. 2010).

No modelo mostrado na Figura 3.5, têm-se o vetor de entrada $x = \{x_1, x_2, \dots, x_n\} \in X$, que são os sinais de entrada advindos do meio externo, ou seja da aplicação e são análogos à impulso elétricos que percorrem o cérebro humano (Figura 3.5). Cada conexão neural humana, que são as junções sinápticas no cérebro, são representadas pelo vetor de pesos $w_i = \{w_1, w_2, \dots, w_n\} \in W$, sendo n o tamanho da dimensão do problema abordado, como pode ser visualizado na Figura 3.5. O modelo de neurônio, inclui um *bias* aplicado externamente (Figura 3.5), onde esse parâmetro tem o efeito de aumentar ou diminuir a entrada líquida da função de ativação, dependendo se ele é positivo ou negativo, respectivamente (Haykin 2001).

A relevância de cada uma das entradas x_i do neurônio é executada pelo somatório da multiplicação de cada entrada por seu respectivo peso (Figura 3.5) e por fim, submete-se o resultado a uma função de ativação, para obter a saída y do neurônio, como pode-se notar na Equação 3.3.

$$y = f\left(\sum_{i=1}^n x_i \cdot w_i + bias\right) \quad (3.3)$$

Assim, de acordo com a Figura 3.5 pode-se observar que o neurônio artificial funciona a partir de alguns elementos básicos:

- Sinais de Entrada $x = \{x_1, x_2, \dots, x_n\} \in X$ que são valores advindos do meio externo e representa valores assumidos pelas variáveis de um problema específico.
- Pesos sinápticos $w_i = \{w_1, w_2, \dots, w_n\} \in W$ que são valores utilizados para ponderar cada uma das variáveis de entrada, fazendo com que sejam quantificadas as suas relevâncias em relação a funcionalidade do neurônio.
- Combinador Linear Σ que agrupa todos os sinais de entrada que foram ponderados pelos seus respectivos pesos sinápticos produzindo assim um valor de potencial de ativação.
- Função de ativação f que tem como objetivo limitar a saída do neurônio dentro de um intervalo de valores razoáveis a serem assumidos pela sua própria imagem funcional.
- sinal de saída y que consiste no valor final produzido pelo neurônio a partir do conjunto de entradas x , podendo ser utilizados por outros neurônios que estão conectados em sequência.

3.3.1 Função de Ativação

Responsável por permitir capacidade representativa às RNA's, introduzindo um componente de não linearidade. Por outro lado, com esse poder a mais surgem algumas dificuldades. Ao introduzir uma ativação não linear, a superfície de custo da RNA deixa de ser convexa, desta forma a otimização torna-se mais custosa. Várias funções podem ser encontradas como ativação das RNA's. As principais funções de ativação são apresentadas abaixo.

Função Degrau

A função degrau ou binária (Equação 3.4) é muito utilizada para criação de um classificador binário. Quando é necessário apenas dizer sim ou não para uma única classe, neste caso ativaría o neurônio ou deixaria zero. A Figura 3.6 apresenta o gráfico que representa a função Degrau.

$$f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \quad (3.4)$$

O resultado produzido pela aplicação da função degrau assumirá valores unitários positivos quando o potencial de ativação do neurônio for maior ou igual a zero, caso contrário, o resultado será valores nulos (da Silva et al. 2010).

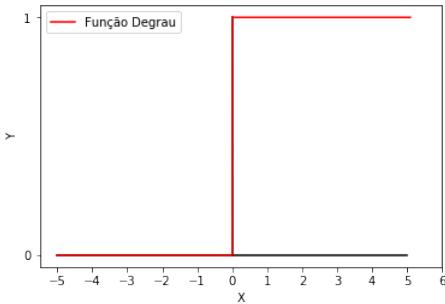


Figura 3.6: Função Degrau.

Função Linear

A função Linear está definida na Equação 3.5 e seu gráfico pode ser visualizado na Figura 3.7.

$$f(x) = ax \quad (3.5)$$

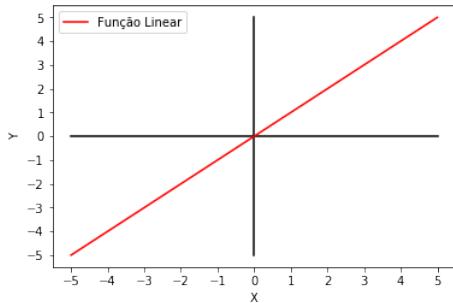


Figura 3.7: Função Linear.

A derivada de uma função linear é constante, isto é, não depende do valor de entrada x . Isso significa que o gradiente será o mesmo para toda execução do *backpropagation*. Uma das aplicabilidades da função de ativação linear se dá quando utilizamos as RNA's como aproximators universais de funções com o objetivo de mapear o comportamento entre variáveis de entrada e saída de processos.

Função Sigmoide

Esta função de ativação é amplamente utilizada em problemas na literatura. Ela pode assumir valores entre 0 e 1 (da Silva et al. 2010) tendo um formato de S como vê-se no gráfico da Figura 3.8. Possui a característica de não linearidade, isso significa essencialmente que quando eu tenho vários neurônios com função sigmoide como função de ativação, a saída também não é linear.

$$f(x) = \frac{1}{1 + \exp^{-x}} \quad (3.6)$$

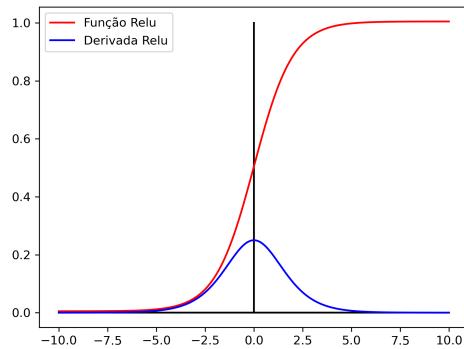


Figura 3.8: Função Sigmoidal.

A função sigmoidal satura quando seu argumento é muito positivo ou muito negativo, o que significa que a função se torna muito plana e insensível a pequenas mudanças em sua entrada (Goodfellow et al. 2016), como podemos visualizar na derivada da função sigmoidal no gráfico da Figura 3.8.

Função Tangente Hiperbólica

Similar à função sigmoidal, a função Tangente Hiperbólica (Equação 3.7) também tem um formato de ‘S’, como mostra o gráfico da Figura 3.9, mas varia de -1 a 1 , em vez de 0 a 1 como na sigmoidal. A tangente hiperbólica se aproxima mais da identidade, sendo assim uma alternativa mais atraente do que a sigmoidal para servir de ativação às camadas ocultas das RNA’s.

$$f(x) = \tanh(x) \quad (3.7)$$

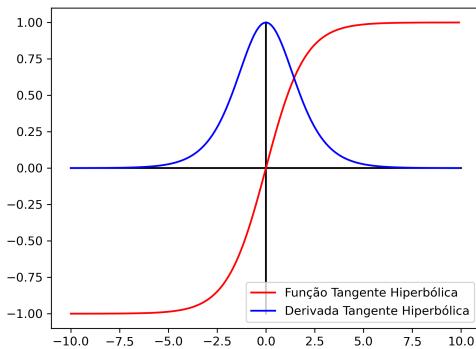


Figura 3.9: Função Tangente Hiperbólica.

As saturações ainda estão presentes, como podemos ver no gráfico da derivada da função na Figura 3.9, mas o valor da derivada é maior, chegando ao máximo de 1 quando

$x = 0$. Por esse motivo, quando uma função Sigmoide precisa ser utilizada, recomenda-se a Tangente Hiperbólica no lugar da sigmoide (Goodfellow et al. 2016).

Função Relu

A função de ativação Sigmoide possui um problema. Quando os valores do potencial se aproxima de 0 ou de 1, o valor da função tende a estabilizar e não mais evoluir, fazendo com que não haja mais aprendizado. De forma similar, a Tangente hiperbólica possui o mesmo problema com valores muito próximo de 1 e -1 .

Para sanar esse problema, temos a função RelU, definida na Equação 3.8. No caso desta função, para todo valor do potencial acima de 0, a resposta será o próprio potencial calculado, fazendo com que o aprendizado não estagne, como na Sigmoide e na Tangente Hiperbólica.

A função Relu, que é a mais amplamente utilizada ao projetar RNA's atualmente, é uma função não linear, o que significa que o erro é facilmente copiado para as camadas anteriores e ter várias camadas de neurônios ativados pela função Relu. O seu gráfico pode ser visto na Figura 3.10.

$$f(x) = \max(0, x) \quad (3.8)$$

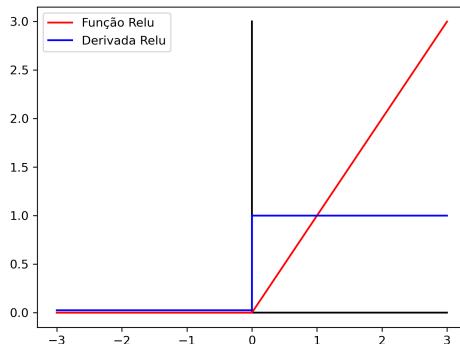


Figura 3.10: Função Relu.

Esta é a função de ativação padrão recomendada para uso com a maioria das redes neurais *feedforward*, no contexto de *Deep Learning*. Aplicar esta função à saída de uma transformação linear produz uma transformação não linear. No entanto, a função permanece muito próxima de linear, no sentido de que é uma função linear por partes com duas peças lineares. Como as unidades lineares retificadas são quase lineares, elas preservam muitas das propriedades que tornam os modelos lineares fáceis de otimizar com métodos baseados em gradiente. Eles também preservam muitas das propriedades que fazem o modelo linear generalizar bem (Goodfellow et al. 2016).

Função Softmax

A função sigmoide é capaz de lidar com problemas de classificação de apenas duas classes, quando trabalhamos com mais de duas classes, existe a função *Softmax* como alternativa. Ela é um tipo de sigmoide que transforma as saídas de cada classe para valores entre 0 e 1 e também divide pela soma das saídas. Isso basicamente é o cálculo da probabilidade da entrada pertencer à uma determinada classe. A função *Softmax* é definida na Equação 3.9.

Sempre que quisermos representar uma distribuição de probabilidade sobre uma variável discreta com n valores possíveis, podemos usar a função *softmax*. Isso pode ser visto como uma generalização da função sigmoide que pode ser usada para representar uma distribuição de probabilidade sobre uma variável binária (Goodfellow et al. 2016). As funções *softmax* são mais frequentemente usadas como saída de um classificador, para representar a distribuição de probabilidade em n classes diferentes. Mais raramente, as funções *softmax* podem ser usadas dentro do próprio modelo, se desejarmos que o modelo escolha entre uma das n opções diferentes para alguma variável interna.(Goodfellow et al. 2016)

A função *softmax* pode então exponenciar e normalizar z para obter o y desejado (Goodfellow et al. 2016). Formalmente, a função *softmax* é dada pela Equação 3.9.

$$\sigma(z)_j = \frac{\exp^{z_j}}{\sum_{k=1}^k \exp^{z_k}} \quad (3.9)$$

Para $j = 1, 2, 3, \dots, k$.

A função *softmax* geralmente é usada na camada de saída da RNA, onde são geradas as probabilidades para definir a classe de cada entrada.

3.3.2 Arquitetura de uma rede neural

A arquitetura de uma rede neural artificial define como os neurônios serão organizados em relação aos outros. Essa organização é baseada em conexões sinápticas dos neurônios. O treinamento de uma RNA consiste da aplicação de um conjunto de passos ordenados com o intuito de ajustar os pesos de seus neurônios, esse processo é chamado de algoritmo de aprendizagem, que visa sintonizar a rede para que seus respostas sejam mais próximas dos valores desejados.

Basicamente uma rede neural pode ser dividida em três partes:

- Camada de entrada que é responsável por receber os sinais de entrada do meio externo para que sejam utilizadas nos cálculos do neurônios.
- Camadas Ocultas que são compostas por neurônios que possuem a responsabilidade de extrair as características associadas ao processo a ser aprendido. A maioria do processamento interno da rede é realizado nessas camadas.
- Camada de Saída é constituída por neurônios que têm a função de produzir a apresentar o resultado final da rede, ou seja, a resposta da rede calculada a partir dos resultados dos processamentos efetuados pelos neurônios das camadas anteriores.

As principais arquiteturas de uma RNA, considerando a organização de seus neurônios, suas formas de interligação entre eles e a constituição de suas camadas, podem ser de dois tipos: *feedforward* de camada simples ou *feedforward* de camadas múltiplas.

Arquitetura *Feedforward* de camada simples

Nesse tipo de arquitetura de RNA, temos a camada de entrada e apenas uma camada de neurônios, que por sua vez é a própria camada de saída. A Figura 3.11 apresenta uma rede *feedforward* de camada simples composta por n entradas e m saídas.

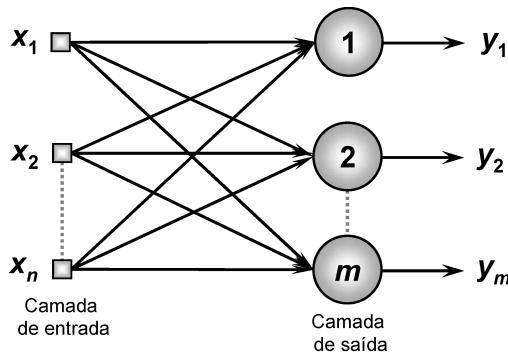


Figura 3.11: Arquitetura *feedforward* de camada simples. Fonte: Silva et al.(2010)

A direção das informações sempre será unicamente da camada de entrada para a camada de saída. De acordo com a Figura 3.11, podemos observar que a quantidade de saídas da rede sempre coincidirá com o número de neurônios.

Arquitetura *feedforward* de múltiplas camadas

Diferentemente da arquitetura discutida anteriormente, as redes *feedforward* de múltiplas camadas são formadas por uma ou mais camadas ocultas, como pode ser visto na Figura 3.12.

A Figura 3.12 mostra uma rede *feedforward* de múltiplas camadas formada por uma camada de entrada com n sinais, duas camadas ocultas formadas por n_1 e n_2 neurônios respectivamente, e por fim uma camada de saída contendo m neurônios.

Podemos observar na Figura 3.12 que a quantidade de neurônio que forma a primeira camada oculta pode ser diferente da quantidade que forma a segunda camada oculta. E que cada neurônio da primeira camada oculta, têm sua saída ligada a cada um dos neurônios da segunda camada oculta. E de forma análoga, a segunda camada oculta ligada à camada de saída. E igualmente na *feedforward* de camadas simples, a quantidade de saídas sempre será igual a quantidade de neurônios na camada de saída.

3.3.3 Processo de treinamento

A grande vantagem das RNA's é o fato delas conseguirem aprender a partir da apresentação de amostras de dados que representam um determinado problema, sendo que,

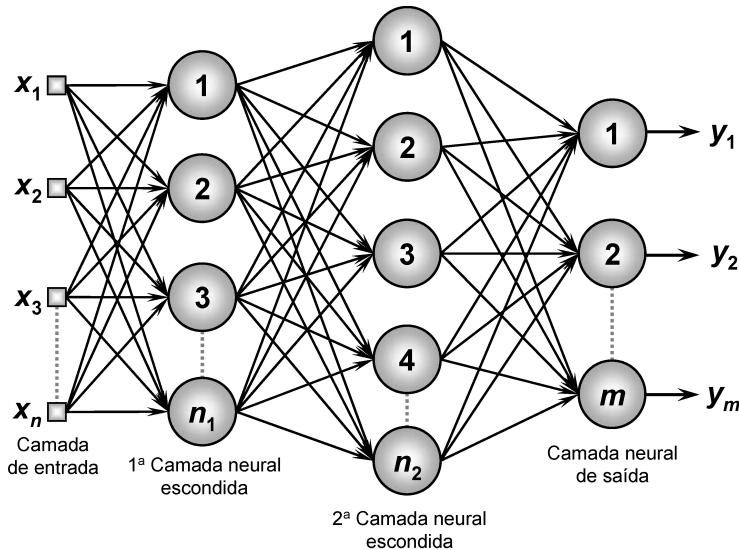


Figura 3.12: Arquitetura *feedforward* de múltiplas camadas. Fonte: Silva et al.(2010)

em seguida, após aprender o relacionamento entre entradas e saídas, e capaz de generalizar soluções e emitir saída para qualquer nova entrada que não tenha sido utilizada no treinamento.

O processo de treinamento de uma RNA consiste na aplicação de passos ordenados que sejam necessários para otimizar os pesos sinápticos de seus neurônios, tendo como objetivo final a generalização de soluções a serem produzidas pela camada de saída. Esse conjunto de passos são denominados algoritmo de aprendizagem (treinamento). Durante o processo de treinamento de uma RNA, a cada apresentação de uma amostra x do conjunto de dados X , visando o ajuste dos pesos sinápticos é denominado época de treinamento.

3.3.4 Função de Custo

A maioria dos algoritmos de aprendizado profundo envolve algum tipo de otimização. Otimização refere-se à tarefa de minimizar ou maximizar alguma função $f(x)$ alterando x . Normalmente, expressamos a maioria dos problemas de otimização em termos de minimizar $f(x)$. A função que queremos minimizar podemos chamá-la de função custo, função de perda ou função de erro (Goodfellow et al. 2016).

A Função de Custo é responsável por dizer o quanto longe o modelo está da predição ideal, ou seja, quantifica o "custo" ou "perda" para uma predição dita por os parâmetros atuais do modelo. Em outras palavras, qual o custo de aceitarmos uma predição y' tendo y como a classe verdadeira do problema?

Existem vários exemplos de Função de Custo, por exemplo, o Erro Quadrático Médio (E), descrito na Equação 3.10. Onde y_i é a saída desejada e y'_i é a saída obtida pela rede e n o número de amostras testadas.

$$E = \sum_{i=1}^n \frac{(y'_i - y_i)^2}{n} \quad (3.10)$$

Outra função de custo muito utilizada na literatura, e também nesse trabalho, é a função entropia Cruzada (CE), mostrada na Equação 3.11, onde p_i é a distribuição de probabilidade desejada e q_i a distribuição de probabilidade obtida.

$$CE(p|q) = - \sum_{i=1}^c p_i \cdot \log q_i \quad (3.11)$$

A função custo de entropia cruzada, ou perda logarítmica, mede o desempenho de um modelo de classificação cuja saída é um valor de probabilidade entre 0 e 1. A perda de entropia cruzada aumenta à medida que a probabilidade prevista diverge do rótulo real. Portanto, prever uma probabilidade de 0,012 quando o rótulo de observação real é 1 seria ruim e resultaria em um alto valor de perda. Um modelo perfeito teria uma perda de \log de 0.

3.4 Deep Learning

Deep Learning é uma sub-área da Aprendizado de Máquina, que emprega algoritmos para processar dados e simular o processamento feito pelo cérebro humano (Goodfellow et al. 2016). Baseados em Redes Neurais Artificiais com maior grau de complexidade, este tipo de solução permite que computadores aprendam a partir de experiências anteriores e compreendam o mundo em termos de uma hierarquia de conceitos, no qual os conceitos mais complexos são definidos e compreendidos em termos de sua relação com conceitos mais simples e já conhecidos (Goodfellow et al. 2016).

As técnicas de DL diferenciam-se de outros métodos de Aprendizado de Máquina, por ter a capacidade de trabalhar com uma quantidade muito maior de dados. Além disso, possui a capacidade de adicionar camadas para extração de características, como mostra Figura 3.13.

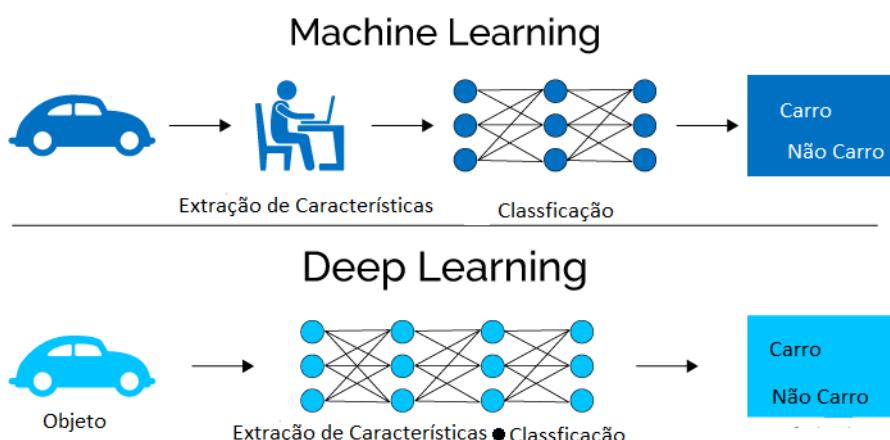


Figura 3.13: Comparativo *Aprendizado de Máquina Comum e Deep Learning*. Fonte: Pilli (2019)

Considerando um problema de reconhecimento facial, para resolver usando técnicas padrão de Aprendizado de Máquina, inicialmente deve-se utilizar algum método para

extração de características da imagem, depois aplicar as características à um modelo de classificação (Figura 3.13). No caso de resolver o problema utilizando técnicas de *Deep Learning*, o extrator de características pode ser descartado, pois as próprias redes neurais profundas, extraem e classificam em um determinado problema (Ponti e da Costa 2018).

Outra diferença entre modelos de aprendizado de máquina e *deep learning* é em relação à arquitetura da rede. Uma rede neural simples, geralmente chamada de rasas (*shallow*), buscam por uma única função $f(\cdot)$ a partir de um conjunto de parâmetros (Goodfellow et al. 2016), para gerar o resultado. Isso porque a rede neural geralmente possui apenas uma camada oculta, como mostra a Figura 3.14. Em que $W = \{w_i\}_{i=1}^n$, que é o vetor de pesos para cada ligação entre os nós (neurônios) da rede.

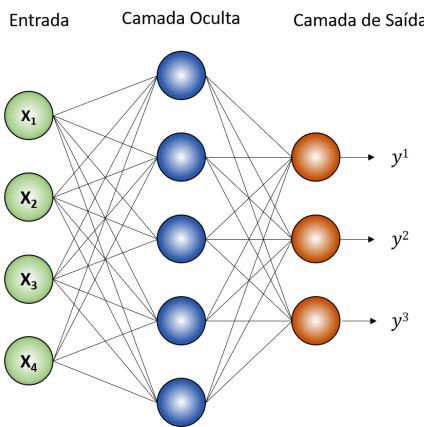


Figura 3.14: Rede Neural Rasa. Fonte: ?

Por outro lado, uma rede DL aprende $f(\cdot)$ por composições de funções, como mostra a Equação 3.12

$$f(x) = f_i(\dots f_2(f_1(x_1)\dots)) \quad (3.12)$$

Em que cada função $f_i(\cdot)$ toma como entrada um vetor x_i , gerando como saída o próximo vetor x_{i+1} . O índice i se refere a cada camada. Isso porque, no caso de uma rede DL, existem várias camadas ocultas, como mostra a Figura 3.15. Desta forma uma das ideias centrais em *Deep Learning* consiste em aprender sucessivas representações dos dados (X), intermediárias, ou seja, os $x_i|_{i=1}^n$.

Matematicamente, pode-se interpretar que essa sequência de transformações separa as múltiplas variedades (do ponto de vista geométrico, do termo inglês *manifolds*) que nos dados originais estariam todas enoveladas (Conway e White 2012).

3.4.1 Regularização

A regularização foi usada por décadas antes do advento do aprendizado profundo. Modelos lineares, como regressão linear e regressão logística, permitem estratégias de regularização simples, diretas e eficazes (Goodfellow et al. 2016).

Muitas abordagens de regularização são baseadas na limitação da capacidade dos modelos, como redes neurais, regressão linear ou regressão logística, adicionando um termo

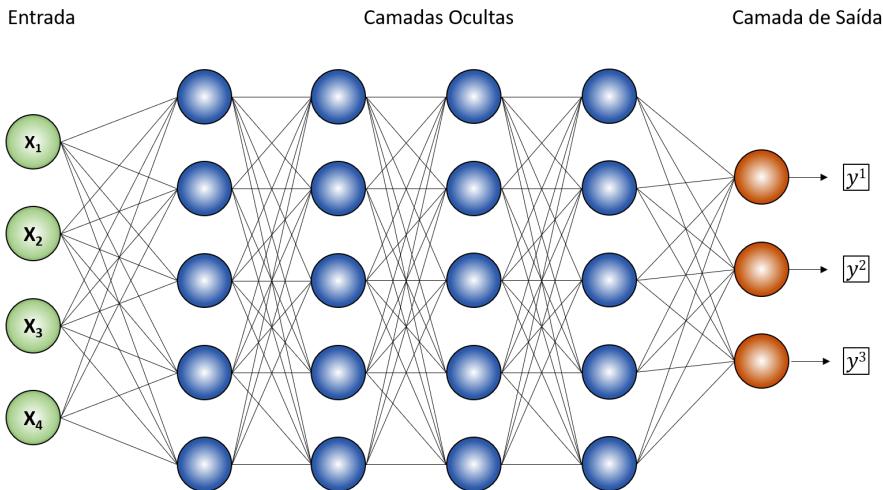


Figura 3.15: Rede Neural Profunda.

de penalidade $\Omega(\theta)$ à função objetivo f . Denotamos a função objetivo regularizada por \tilde{f} , na Equação 3.13 (Goodfellow et al. 2016):

$$\tilde{f}(\theta, X, y) = f(\theta, X, y) + \alpha\Omega(\theta) \quad (3.13)$$

Onde $\alpha \in [0, \infty)$ é um hiperparâmetro que pondera a contribuição relativa do termo de penalidade da norma, Ω , em relação à função objetivo padrão $f(x, \theta)$. Definir α como 0 resulta em nenhuma regularização. Valores maiores de α correspondem a mais regularização (Goodfellow et al. 2016).

Quando o algoritmo de treinamento minimiza a função objetivo regularizada \tilde{f} , ele diminuirá o objetivo original f nos dados de treinamento e alguma medida do tamanho dos parâmetros θ (ou algum subconjunto dos parâmetros). Diferentes escolhas para a norma de parâmetro Ω podem resultar em soluções diferentes sendo preferidas (Goodfellow et al. 2016).

Dropout

Quando aumenta a profundidade de uma rede ela fica suscetível ao *overfitting*, que é quando uma hipótese apresenta uma baixa capacidade de generalização a razão pode ser que ela está superajustada aos dados de treinamento (Hinton et al. 2012).

O *dropout* é uma técnica que aborda esses dois problemas. Ele evita o *overfitting* e fornece uma maneira de combinar de forma exponencial e eficiente muitas arquiteturas de rede neural diferentes. O termo “dropout” refere-se a unidades de dropout (ocultas e visíveis) em uma rede neural. Ao descartar uma unidade, queremos dizer removê-la temporariamente da rede, junto com todas as suas conexões de entrada e saída, conforme mostrado na Figura 3.16 (Srivastava et al. 2014).

A escolha de quais unidades descartar é aleatória. No caso mais simples, cada unidade é retida com uma probabilidade fixa p independente de outras unidades, onde p pode ser escolhido usando um conjunto de validação ou pode simplesmente ser definido em 0,5,

o que parece estar próximo do ideal para uma ampla faixa de redes e tarefas. Para as unidades de entrada, no entanto, a probabilidade ótima de retenção é geralmente mais próxima de 1 do que de 0,5 (Srivastava et al. 2014). A Figura 3.16, representa visualmente a utilização de *dropout* em uma rede neural.

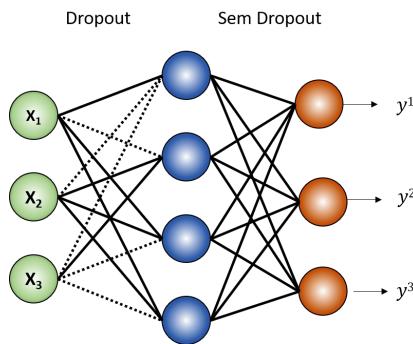


Figura 3.16: Rede com *Dropout*.

Na Figura 3.16, pode-se observar as conexões entre neurônios da camada de entrada e a camada oculta desabilitadas, representadas por linhas pontilhadas. E conexões mantidas representadas por linha preenchida. A ideia do *Dropout* não se limita a redes neurais *feed-forward*. Ele pode ser aplicado de forma mais geral a modelos gráficos, como Máquinas Boltzmann (RBM). Segundo Srivastava et al. (2014), experimentos mostram que RBMs com dropout são melhores do que RBMs padrão em certos aspectos.

3.5 Aprendizado por Teoria da Informação

Um problema comum enfrentado por muitos profissionais de processamento de dados é como extrair melhor as informações a partir dos dados. Diariamente, uma enorme quantidade de dados é gerada, mas na maioria das vezes os dados não são o interesse principal. Os dados ocultam, seja na estrutura do tempo ou na redundância espacial, pistas importantes para responder à questionamentos acerca do processamento de informações que são levantados (Principe 2010).

A teoria da informação é uma parte da matemática que estuda quantificação da informação. Shannon (1948) formalizou conceitos com aplicações na teoria da comunicação e estatística. Shannon (1948) inicialmente tinha como objetivo lidar com o problema de transmissão de mensagens de forma ótima através de um canal ruidoso. Desta forma, mesmo com fatores físicos envolvidos nos sistemas de comunicação, tais como antenas, transmissões, receptores, entre outros, com a teoria da informação seria possível tratar a caracterização da estrutura das mensagens e o limite da transmissão livre de erros do conteúdo dessas mensagens (Principe 2010).

De acordo com Cover e Thomas (2006), a teoria da informação foi desenvolvida originalmente para compressão de dados, para transmissão e armazenamento. Em outras palavras, foi criada para ajudar a responder as questões teóricas de como codificar otimamente mensagens de acordo com suas estruturas estatísticas. Porém, foi planejada para

aplicação ampla e têm sido usada em muitas outras áreas cujo fator de análise são os dados.

Em síntese, a teoria da informação define medidas que tem o objetivo de quantificar a informação contida em uma dada variável aleatória ou até mesmo o montante de informações que uma variável aleatória possui sobre outra.

Segundo Principe (2010), o aprendizado por teoria da informação (ITL) é uma área de estudo que usa descritores da teoria da informação estimados a partir dos dados para substituir descritores estatísticos convencionais como variância e covariância. O grande sucesso do uso de ITL deve-se principalmente ao fato de ser uma técnica baseada em estatística de alta ordem, já que seus descritores são estimados diretamente a partir das amostras de dados. ITL propõe o uso de uma função de custo com eficiência computacional e sem sofrer a limitação de gaussianidade inerente às funções de custo baseadas em momentos de segunda ordem.

Isto é conseguido com o uso de descritores da teoria da informação como a entropia e medidas de dissimilaridades (divergência e informação mútua) combinados com estimadores de função densidade de probabilidade (PDF) não paramétricos, que trazem robustez e generalidade para a função de custo e melhoram o desempenho em muitos cenários realistas.

3.5.1 Entropia

Hartley (1928) definiu que a quantidade de informação existente em um conjunto de mensagens está relacionada com o número de símbolos que aquele conjunto é formado. Portanto, a quantidade de informação (H_0) em um conjunto de N símbolos é definida como mostrado na Equação 3.14, em que S é o número de símbolos possíveis.

$$H_0 = \log S^N = N \log S \quad (3.14)$$

Com isso, Shannon (1948) percebeu que o conteúdo de informação definido por Hartley (1928) é exato apenas quando não existe nenhum conhecimento a cerca dos dados, isto é, assumindo uma probabilidade igual para todos os eventos, ou seja, $p_i = \frac{1}{N}$.

Então, foi estabelecido que se deve ir além da cardinalidade do conjunto de mensagens para quantificar precisamente o total de escolhas que está envolvida na seleção de eventos probabilísticos, considerando que a probabilidade de selecionar cada mensagem deve ser considerada relevante na formulação (Shannon 1948).

Se um determinado evento i ocorre com probabilidade $p(x) = 1$, consequentemente qualquer outro evento possui $p(x_k) = 0$, para todo $i \neq k$. Neste caso, não existe como ter surpresa, então, nenhuma informação é transmitida pela ocorrência do evento x_k , pois já é sabido como a mensagem será (Haykin 2001). Porém, se existir eventos com probabilidades distintas, e um deles possuir probabilidade baixa, então, há uma surpresa com o que pode ocorrer no evento, e desta forma, têm-se mais informação. Portanto, é possível ver que os conceitos de surpresa, informação e incerteza estão intimamente relacionados.

Portanto, para considerar as probabilidades de cada evento no momento de calcular a quantidade de informação e, com isso, caracterizar totalmente o elemento de um conjunto

de símbolos S_x ocorrendo com diferentes probabilidades $p(x_i)$, a quantidade de informação ($I(x_i)$) é definida como mostrado na Equação 3.15.

$$I(x_i) = \log \frac{1}{p(x_i)} = -\log p(x_i) \quad (3.15)$$

A incerteza do conjunto $X = \{x_1, x_2, \dots, x_n\} \in \mathfrak{R}$, segundo Shannon (1948), é definida como a soma das incertezas de todas as mensagens ponderadas pela probabilidade de cada uma, como mostra a Equação 3.16.

$$H(X) = E[I_k] = \sum p(x_i)I(x_i) = -\sum p(x_i)\log p(x_i) \quad (3.16)$$

Onde, $\sum_{i=1}^n p(x_i) = 1$ e $p(x_i) \geq 0$. Shannon (1948) nomeou essa quantidade de incerteza (Equação 3.16) de entropia, supondo que para $p(x_i) = 0$, $p(x_i)\log p(x_i) = 0$. A unidade de informação depende da base logarítmica usada. Inicialmente, Shannon (1948) utilizou a base 2, para indicar que as quantidades deveriam ser expressas em bits. Porém, outras bases podem ser utilizadas, de acordo com o contexto abordado.

A entropia $H(X)$ é a medida de quantidade média de informação transmitida por uma mensagem. A combinação das incertezas são ponderadas pelas suas probabilidades, isso faz a essência do conceito de entropia. Assim, eventos que ocorrem com frequência possuem pouco conteúdo de informação e possuem um valor baixo no cálculo da entropia.

3.5.2 Entropia Conjunta e Entropia Condisional

O conceito de entropia apresentado anteriormente, atende apenas em situações em que existe apenas uma variável aleatória. Considerando um par de variáveis, pode-se criar mais duas novas definições de entropia: conjunta e condicional.

A entropia conjunta é definida na Equação 3.17 (Haykin 2001, Cover e Thomas 2006, Principe 2010).

$$H(X, Y) = -E_{X,Y}[\log p(X, Y)] = -\sum \sum p(x, y) \log p(x, y) \quad (3.17)$$

Em que, $p(x, y)$ é a probabilidade de x e y ocorrerem juntos. De forma semelhante, a entropia condicional é definida na Equação 3.18 (Haykin 2001, Cover e Thomas 2006, Principe 2010).

$$H(Y|X) = -E_{X,Y}[\log p(Y|X)] = -\sum \sum p(x, y) \log p(y|x) \quad (3.18)$$

Em que, $p(y|x)$ é a probabilidade condicional de y dado x . A entropia condicional também pode ser expressada de acordo com a Equação 3.19, considerando a propriedade $0 \leq H(X|Y) \leq H(X)$.

$$H(X|Y) = H(X, Y) - H(Y) \quad (3.19)$$

Vale ressaltar, que a entropia calcula a incerteza de uma distribuição de probabilidade, enquanto que a entropia conjunta e condicional são apenas extensões que calculam a incerteza da distribuição conjunta e distribuição condicional de duas variáveis aleatórias.

3.5.3 Divergência de Kullback-Leibler

A divergência de Kullback-Leibler (D_{KL}) (Kullback e Leibler 1951) é a medida de ineficiência de assumir que uma distribuição é p quando, na verdade, ela é definida como q . Essa divergência é definida na Equação 3.20.

$$D_{KL}(p||q) = E_p \left[\log \frac{p(X)}{q(X)} \right] = \sum p(x) \log \frac{p(x)}{q(x)} \quad (3.20)$$

Analizando a Equação 3.20, é possível ver que D_{KL} é sempre não-negativa e zero se, e somente se, $p = q$ (Cover e Thomas 2006). A divergência de Kullback-Leibler é muito utilizada como medida de distâncias entre distribuições de probabilidade. Porém, como não segue à desigualdade triangular e por não ser simétrica, não pode ser considerada uma métrica.

A divergência de Kullback-Leibler é a quantidade de incerteza que se tem ao observar uma distribuição $f(x)$ usando outra distribuição $g(x)$. D_{KL} é uma medida de divergência entre duas distribuições probabilidades baseada na entropia proposta por (Shannon 1948).

3.5.4 Entropia Cruzada

A entropia cruzada é uma medida da diferença entre duas distribuições de probabilidade para uma determinada variável aleatória ou conjunto de eventos. Como dito anteriormente, as informações quantificam o número de bits necessários para codificar e transmitir um evento. Os de probabilidade mais baixa têm mais informações, eventos de probabilidade mais alta têm menos informações (Murphy 2012).

A informação $h(x)$ pode ser calculada para um evento x , dada a probabilidade do evento $P(x)$, seguindo a Equação 3.21:

$$h(x) = -\log(P(x)) \quad (3.21)$$

Entropia é o número de bits necessários para transmitir um evento selecionado aleatoriamente a partir de uma distribuição de probabilidade. Uma distribuição enviesada tem uma entropia baixa, enquanto uma distribuição onde os eventos têm probabilidade igual tem uma entropia maior (Murphy 2012).

Uma distribuição de probabilidade distorcida tem menos “surpresa” e, por sua vez, uma baixa entropia porque os eventos prováveis dominam. A distribuição balanceada é mais surpreendente e os turnos têm uma entropia mais alta porque os eventos são igualmente prováveis (Murphy 2012).

A entropia $H(x)$ pode ser calculada para uma variável aleatória com um conjunto de $x \in X$ estados discretos e sua probabilidade $P(x)$ seguindo a Equação 3.22.

$$H(X) = -\sum P(x) \cdot \log P(x) \quad (3.22)$$

A entropia cruzada baseia-se na ideia de entropia da teoria da informação e calcula o número de bits necessários para representar ou transmitir um evento médio de uma distribuição em comparação com outra distribuição (Murphy 2012).

A intuição para esta definição vem se considerarmos um alvo ou distribuição de probabilidade subjacente p e uma aproximação da distribuição alvo q , então a entropia cruzada de q de p é o número de bits adicionais para representar um evento usando q em vez de p (Murphy 2012).

A entropia cruzada entre duas distribuições de probabilidades, como q de p , pode ser formalizada segundo a Equação 3.23.

$$H(p, q) = - \sum p(x) \cdot \log q(x) \quad (3.23)$$

Onde $p(x)$ é a probabilidade do evento x em p , $q(x)$ é a probabilidade do evento x em q e \log é o logaritmo de base 2, o que significa que os resultados estão em bits.

3.6 Considerações

Este capítulo apresentou os conceitos necessários para o desenvolvimento deste trabalho. Apresentou-se os conceitos de Aprendizado de Máquina, pois este é base principal do trabalho desenvolvido, explorando o conceito de redes neurais artificiais e *Deep Learning*. E por fim, apresentou-s e o conceito básico de Teoria da Informação, e sua definição dentro do contexto de aprendizado.

Todos esses conceitos fazem parte do desenvolvimento do trabalho apresentado nesta tese. No Capítulo seguinte, serão apresentados os métodos utilizados para desenvolver o modelo proposto. Além disso, as métricas e bases de dados adotadas nos experimentos.

Capítulo 4

Materiais e Métodos

Este capítulo apresenta as bases de dados e métricas utilizadas para validação, bem como os métodos usados para compor o modelo proposto. Para tanto, foram selecionados algumas bases de dados da literatura, dividindo em quatro categorias: bases *benchmark*, Bases de Imagens, Bases de Sensoriamento Remoto e dados de *stream*. São apresentadas também, as métricas de avaliação com objetivo de validar e avaliar o modelo proposto, bem como compará-lo com outros métodos da literatura que foram apresentados no Capítulo 2.

4.1 Métodos Utilizados

Neste trabalho, utilizou-se uma série de métodos para formar o modelo proposto, dentre eles, técnicas de *Deep Learning*, no caso do *Deep Autoencoder* e *Convolucional Autoencoder*. Além do algoritmo *K-means++* e o otimizador Adam.

4.1.1 *Multilayer Perceptron (MLP)*

Para o desenvolvimento do modelo proposto neste trabalho, utilizou-se a rede *Multilayer Perceptron* (MLP), na classificação semissupervisionada. A MLP têm sido bastante aplicadas em problemas difíceis na literatura, tais como, reconhecimento de objetos, classificação de perfis, entre outros, obtendo sucesso nos resultados. MLP trata-se de uma rede neural com uma ou mais camadas ocultas entre a entrada e saída da rede e seu treinamento é realizado de forma supervisionada utilizando o algoritmo *Backpropagation* (Haykin 2001).

A Figura 4.1 apresenta a estrutura básica de uma MLP, onde mostra uma rede com duas camadas ocultas e uma camada de saída. Neste caso a rede é totalmente conectada, ou seja, todos os neurônios de uma camada possuem conexão com todos os neurônios da próxima camada. A direção da operação da MLP progride da esquerda para direita, ou seja, os dados entram pela camada de entrada, percorrem pelas camadas ocultas até ser propagado para camada de saída, onde serão coletadas as saídas (Haykin 2001).

Na Figura 4.1, têm-se o vetor de entrada $\mathbf{x} = \{x_1, x_2, x_3, \dots, x_n\}$ onde n é a dimensão da entrada. Para cada conexão entre os neurônios existe um peso w associado, sendo o conjunto de vetores de pesos $\mathbf{w} = \{w_{j,i}^{(1)}, w_{j,i}^{(2)}, w_{j,i}^{(3)}\}$, onde j e i representam o neurônio

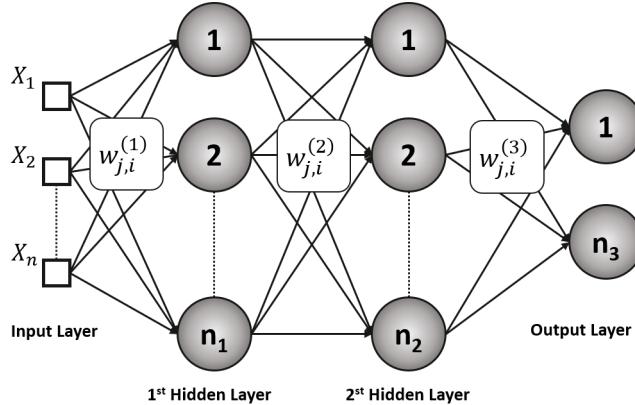


Figura 4.1: Arquitetura básica de uma Rede *Perceptron* Multicamadas. Fonte: da Silva et al. (2010)

da camada anterior e da próxima camada entre as conexões, respectivamente. Na última camada têm-se a saída, onde é respondido o vetor $\mathbf{y} = \{y_1, y_2, y_3, \dots, y_m\}$, sendo m a quantidade de neurônio na camada de saída.

Cada neurônio da MLP, segue a mesma abordagem descrita na Sessão 3.3. Onde realiza-se a combinação linear entre o vetor de entrada X com os vetores de pesos de W . E por fim, o resultado é submetido à uma função de ativação (da Silva et al. 2010).

4.1.2 Algoritmo *Backpropagation*

O algoritmo backpropagation foi originalmente introduzido na década de 1970, mas ganhou importância depois do artigo Rumelhart et al. (1986). Esse artigo descreve várias redes neurais em que o *backpropagation* funciona muito mais rapidamente do que as abordagens anteriores de aprendizado, possibilitando o uso de redes neurais para resolver problemas que antes eram insolúveis.

Para treinar uma rede *multilayer perceptron*, utiliza-se o algoritmo *Backpropagation*, que é realizado em duas fases bem definidas: *forward* e *backward* (da Silva et al. 2010). O treinamento de uma rede neural consiste em atualizar os pesos de suas conexões, de tal forma, conseguir predizer corretamente para qualquer nova amostra que for submetida na rede (Haykin 2001).

A primeira fase a ser aplicada é denominada de *forward*, na qual a entrada $x = x_1, x_2, \dots, x_n \in X$ é inserida na rede e é propagada camada a camada até chegar à camada de saída (da Silva et al. 2010). Portanto, essa fase visa apenas obter as respostas da rede, considerando apenas os valores dos atuais pesos sinápticos e limiares dos neurônios da rede, dos quais permaneceram os mesmos durante cada execução desta fase (da Silva et al. 2010).

Logo em seguida, as respostas produzidas pela rede são comparadas com as respectivas respostas desejadas que são disponibilizadas. Então, podemos calcular o custo baseado em uma função (da Silva et al. 2010).

Desta forma, os valores calculados na função custo são utilizados para a segunda fase

do algoritmo *backpropagation*, denominada de *backward*. Nessa fase, os pesos sinápticos e bias de todos os neurônios são ajustados (da Silva et al. 2010).

Na fase *backward*, o custo calculado na camada de saída é utilizado para corrigir os pesos, posteriormente o erro é retro-propagado para as camadas ocultas até chegar à camada de entrada. Esse cálculo é feito seguindo a Regra do Gradiente mostrado na Equação 4.1.

$$w'_{ij} = w_{ij} + \eta * \delta_j * \frac{df_j(e)}{de} x_j \quad (4.1)$$

sendo,

$$\delta_j = \sum_i w_{ij} x_j \quad (4.2)$$

Em que η é a taxa de aprendizagem da rede, δ é a influência do erro para cada neurônio, w_{ij} é o valor a ser corrigido no peso, w'_{ij} o valor do peso corrigido e f é a função de ativação.

Um ponto importante nas redes neurais multicamadas é a função de ativação, que é encontrada em todos os neurônios da rede. Uma das finalidades da função de ativação é evitar o acréscimo progressivo dos valores de saída ao longo das camadas da rede, considerando que tais funções possuem valores máximos e mínimos estabelecidos em intervalos (Junior e Costa 2007)(Faceli et al. 2011).

4.1.3 Otimizador Adam

Os otimizadores estão presentes na etapa de atualização de pesos da rede, com o objetivo de minimizar a função custo. Existem diversos otimizadores, como Adagrad, Ada-delta e até mesmo o próprio gradiente descendente com a aplicação pura do algoritmo backpropagation. Nessa seção, o otimizador ADAM (Kingma e Ba 2014) será tratado com maior profundidade através do pseudocódigo no Algoritmo 1 que representa o funcionamento do algoritmo, pois foi o otimizador utilizado no desenvolvimento do trabalho.

No Algoritmo 1 pode-se observar os parâmetros de configuração do Adam, que são: α como taxa de aprendizado, β_1 como taxa de decaimento exponencial para as estimativas do primeiro momento, β_2 como taxa de decaimento exponencial para as estimativas do segundo momento e o ϵ que representa um número pequeno para impedir que ocorra alguma divisão por zero na implementação. Segundo o artigo original do otimizador Adam, são boas configurações de parâmetro: $\alpha = 0,001$, $\beta_1 = 0,9$, $\beta_2 = 0,999$ e $\epsilon = 10^{-8}$ (Kingma e Ba 2014). O algoritmo trabalha a todo momento no objetivo de atualizar e melhorar seus parâmetros iniciais.

4.1.4 Redes Neurais Convolucionais (CNN)

Do inglês Convolutional Neural Networks (CNN's), as redes neurais convolucionais são um tipo específico de redes neurais artificiais, proposta pelo pesquisador francês LeCun et al. (1998). Redes convolucionais são um tipo especializado de rede neural

Algoritmo 1: Algoritmo Adam

Result: G **Entrada:** $\alpha, \beta_1, \beta_2, f(\theta), \theta_0$ **Saída :** θ_t (parâmetros resultantes)

```

1  $m_0 \leftarrow 0$  (Inicialização do vetor dos primeiros momentos)
2  $v_0 \leftarrow 0$  (inicialização do vetor de segundos momentos)
3  $f \leftarrow 0$  (Inicialização do passo de tempo)
4 while  $\theta_t$  não convergir do
5    $t \leftarrow t + 1$ 
6    $g_t \leftarrow \nabla_{\theta} f(\theta_{t-1})$  (calcular os gradientes da função objetivo no tempo t)
7    $m_t \leftarrow \beta_1 * m_{t-1} + (1 - \beta_1) * g_t$  (atualizar a estimativa do primeiro momento)
8    $v_t \leftarrow \beta_2 * v_{t-1} + (1 - \beta_2) * g_t^2$  (atualizar a estimativa do segundo momento)
9    $m'_t \leftarrow m_t / (1 - \beta_1^t)$  (computar a estimativa do primeiro momento)
10   $v'_t \leftarrow v_t / (1 - \beta_2^t)$  (computar a estimativa do segundo momento)
11   $\theta \leftarrow \theta_{t-1} - \alpha * m'_t / (\sqrt{v'_t} + \epsilon)$  (atualizar parâmetros)
12 end

```

para processamento de dados que tem uma topologia conhecida, semelhante a uma grade (LeCun e others 1989) (Goodfellow et al. 2016).

As CNNs se mostraram, desde a sua criação, serem muito eficazes para resolver problemas de classificação, se mostrando uma alternativa viável aos métodos tradicionais para esse tipo de problema. As redes convolucionais têm sido extremamente bem-sucedidas em aplicações práticas.

A convolução alavanca três ideias importantes que podem ajudar a melhorar um sistema de aprendizado de máquina: interações esparsas, compartilhamento de parâmetros e representações equiparáveis. Além disso, a convolução fornece um meio de trabalhar com entradas de tamanho variável. Descrevemos agora cada uma dessas ideias (Goodfellow et al. 2016).

Camadas de rede neural tradicionais usam multiplicação de matriz por uma matriz de parâmetros com um parâmetro separado que descreve a interação entre cada unidade de entrada e cada unidade de saída. Isso significa que cada unidade de saída interage com cada unidade de entrada. Redes convolucionais, no entanto, normalmente têm interações esparsas (Goodfellow et al. 2016).

Por exemplo, ao processar uma imagem, a entrada pode ter milhares ou milhões de pixels, mas podemos detectar características pequenas e significativas, como bordas com núcleos que ocupam apenas dezenas ou centenas de pixels. Isso significa que precisamos armazenar menos parâmetros, o que reduz os requisitos de memória do modelo e melhora sua eficiência estatística. Também significa que computar a saída requer menos operações. Essas melhorias em eficiência são geralmente significativas (Goodfellow et al. 2016).

Uma das desvantagens das CNNs é o fato de existir a necessidade de uma grande quantidade de dados rotulados para a extração dos padrões, ou características. Basicamente, para extração dessas características, podem existir três componentes básicos em

uma CNN, camada de convolução, *pooling* e rede totalmente conectada, como mostrado na Figura 4.2 (Patterson e Gibson 2017, Goodfellow et al. 2016). Qualquer arquitetura básica é composta por esses blocos.

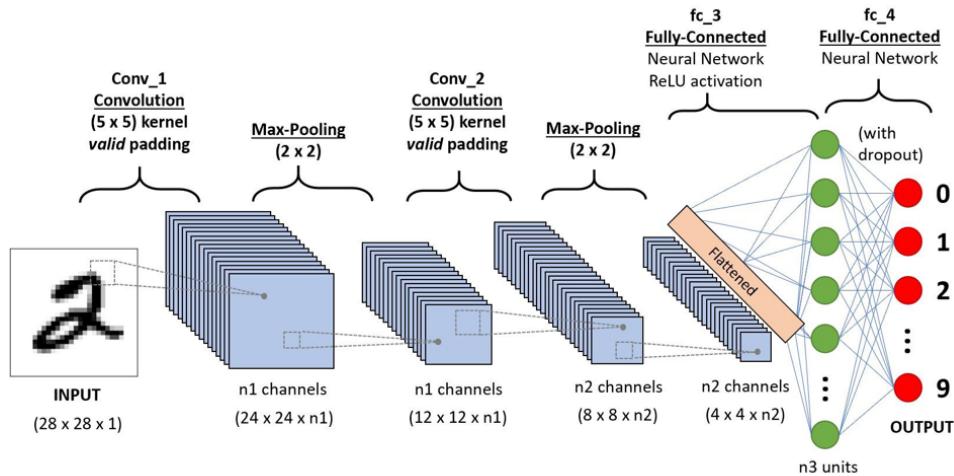


Figura 4.2: Arquitetura básica de uma Rede Neural Convolucional. Fonte: Poliana Reis (2017)

4.1.5 Camada convolucional

A camada convolucional em uma CNN é responsável por extrair as características da entrada. O processo de extração dessas características é realizado por meio de filtros convolucionais de tamanhos reduzidos, onde os filtros percorrem os dados de entrada em largura, altura e profundidade (chamada de dimensão) realizando a operação de convolução sobre os dados (Goodfellow et al. 2016).

Em sua forma mais geral, a convolução é uma operação em duas funções de um argumento reavaliado. Para facilitar a definição de convolução, começamos com exemplos de duas funções que podemos usar.

Suponha que estejamos rastreando a localização de uma nave espacial com um sensor a laser. Nossa sensor a laser fornece uma única saída $x(t)$, a posição da espaçonave no tempo t . Ambos x e t têm valores reais, ou seja, podemos obter uma leitura diferente do sensor de laser a qualquer momento

Agora, suponha que nosso sensor a laser seja um pouco barulhento. Para obter uma estimativa menos ruidosa da posição da nave espacial, gostaríamos de fazer a média de várias medições. Obviamente, as medições mais recentes são mais relevantes, portanto, queremos que essa seja uma média ponderada que dê mais peso às medições recentes. Podemos fazer isso com uma função de ponderação $w(a)$, onde a é a idade de uma medição. Se aplicarmos essa operação de média ponderada a cada momento, obteremos uma nova função s fornecendo uma estimativa suavizada da posição da nave espacial, vide Equação 4.3.

$$\int x(a)w(t-a)da \quad (4.3)$$

Esta operação (Equação 4.3) é chamada de convolução. A operação de convolução é normalmente indicada com um asterisco (Goodfellow et al. 2016), como mostrado na Equação 4.4.

$$s(t) = (x * w)(t) \quad (4.4)$$

No exemplo mostrado, w precisa ser uma função de densidade de probabilidade válida ou a saída não é uma média ponderada. Além disso, w precisa ser 0 para todos os argumentos negativos, ou olhará para o futuro, o que presumivelmente está além de nossas capacidades. No entanto, essas limitações são específicas do nosso exemplo. Em geral, a convolução é definida para quaisquer funções para as quais a integral na Equação 4.3 é definida e pode ser usada para outros fins além de obter médias ponderadas (Goodfellow et al. 2016).

Na terminologia de rede convolucional, o primeiro argumento (neste exemplo, o x da função) para a convolução é frequentemente referido como a entrada e o segundo argumento (neste exemplo, o w da função) como o kernel ou filtro. A saída é chamada de mapa de características, segundo Goodfellow et al. (2016).

A cada iteração de treinamento da rede, os filtros vão sendo ajustados de tal modo a disparar quando a entrada contiver uma determinada característica comum aos lotes de entrada, como por exemplo, arestas, cores, dentre outras características. A cada entrada na rede, os filtros vão aprendendo estruturas cada vez mais complexas, ou seja, quanto mais filtros convolucionais, mais características extraí-se da entrada, porém isso tem um custo de memória e processamento, o que precisa ser balanceado na hora de definir a arquitetura (Goodfellow et al. 2016).

As convoluções funcionam como filtros que enxergam pequenos quadrados e vão “es-corregendo” por toda a imagem captando os traços mais marcantes. Explicando melhor, com uma imagem 32x32x3 e um filtro que cobre uma área de 5x5 da imagem com movimento de 2 saltos (chamado de *stride*), o filtro passará pela imagem inteira, por cada um dos canais, formando no final um mapa de características ou mapa de ativação de 28x28x1 (Patterson e Gibson 2017) (Goodfellow et al. 2016).

A profundidade da saída de uma convolução é igual a quantidade de filtros aplicados. Quanto mais profundas são as camadas das convoluções, mais detalhados são os traços identificados com o mapa de ativação (Patterson e Gibson 2017) (Goodfellow et al. 2016).

O filtro, que também é conhecido por kernel, é formado por pesos inicializados aleatoriamente, atualizando-os a cada nova entrada durante o processo de *backpropagation*. A pequena região da entrada onde o filtro é aplicado é chamada de campo receptivo, e é representada como mostrado na Figura 4.3.

4.1.6 Função Relu

As funções de ativação servem para trazer a não-linearidades ao sistema, para que a rede consiga aprender qualquer tipo de funcionalidade. Há muitas funções, como descrito

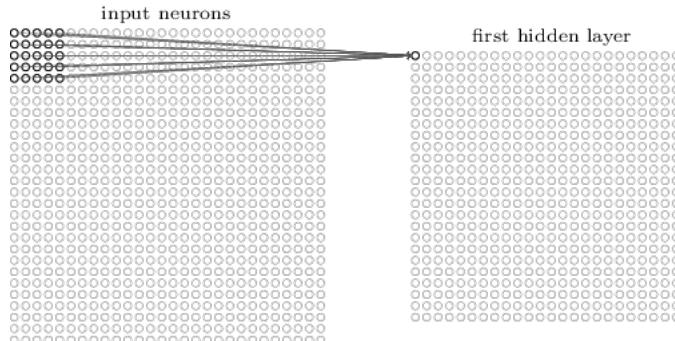


Figura 4.3: Exemplo de uma convolução em uma matriz. Fonte: Galli (2016)

anteriormente, sigmoide, tangente hiperbólica e softmax, a mais indicada para redes convolucionais é a elu por ser mais eficiente computacionalmente sem grandes diferenças de acurácia quando comparada a outras funções.

Com isso, aplica-se a função relu à todos os valores do mapa de característica gerado na camada de convolução, zerando todos os valores negativos da saída da camada anterior.

4.1.7 Camada de *Pooling*

Uma camada típica de uma rede convolucional consiste em três estágios. No primeiro estágio, a camada realiza várias convoluções em paralelo para produzir um conjunto de ativações lineares. No segundo estágio, cada ativação linear é executada por meio de uma função de ativação não linear, como a função de ativação linear retificada. Este estágio às vezes é chamado de estágio do detector. No terceiro estágio, usamos uma função de pool para modificar ainda mais a saída da camada (Goodfellow et al. 2016).

Uma função de agrupamento substitui a saída da rede em um determinado local por uma estatística resumida das saídas próximas (Goodfellow et al. 2016). Por exemplo, o *max pooling* (Zhou e Chellappa 1988) reporta a produção máxima dentro de uma vizinhança retangular, como podemos ver na Figura 4.4.

Camadas de *pooling* são comumente inseridas entre camadas convolucionais sucessivas. As camadas de *pooling* são utilizadas para reduzir progressivamente o tamanho espacial (largura e altura) da representação de dados. Reduzem a representação dos dados progressivamente na rede e ajudam a controlar o *overfitting*. A camada de agrupamento opera independentemente em cada fatia de profundidade da entrada (Patterson e Gibson 2017).

O *pooling* ajuda a tornar a representação aproximadamente invariante para pequenas traduções da entrada. A invariância para a tradução significa que, se traduzirmos a entrada em uma pequena quantidade, os valores da maioria das saídas combinadas não mudam (Goodfellow et al. 2016).

O uso de *pooling* pode ser visto como uma adição significativamente forte de que a função que a camada aprende deve ser invariável para pequenas traduções. Quando essa suposição está correta, ela pode melhorar muito a eficiência estatística da rede. *Pooling* sobre regiões espaciais produz invariância para tradução, mas se aplicarmos *pooling* sobre

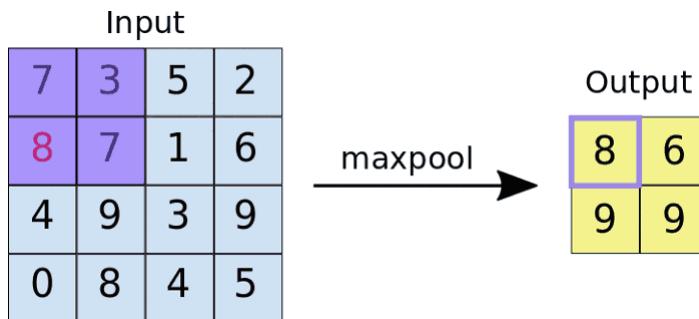


Figura 4.4: Visualização do *Pooling*.

as saídas de convoluções parametrizadas separadamente, os recursos podem aprender a quais transformações se tornam invariáveis (Goodfellow et al. 2016).

Como o pooling resume as respostas de uma vizinhança inteira, é possível usar menos unidades de pooling do que unidades de detector, relatando estatísticas resumidas para regiões de pool espaçadas em k pixels em vez de 1 pixel (Goodfellow et al. 2016). Isso significa que a técnica de *pooling* reduz o custo computacional, pois diminuir a quantidade de dados que será processado de uma camada para outra, consequentemente melhorando a regularização da rede, reduzindo custo de memória e processamento.

4.1.8 Rede Totalmente Conectada

Ao final da rede é colocada uma camada Rede totalmente conectada (Fully Connected na Figura 4.2), onde sua entrada é a saída da camada anterior e sua saída são N neurônios, com N sendo a quantidade de classes do seu modelo para finalizar a classificação.

E por fim, a última parte de uma CNN é a camada totalmente conectada (Figura 4.2). Situada no final da rede de convolução, elas recebem as características extraídas nas camadas de convolução, e são utilizadas para calcular a saída de classificação da rede.

4.1.9 Rede Autoencoder (AE)

Um *autoencoder* é uma rede neural que é treinada para refazer uma cópia da entrada para a saída. Internamente, existem camadas ocultas h que descrevem o código usado para representar a entrada. A rede é formada por duas partes: uma função *encoder* $h = f(\mathbf{x})$ e uma função *decoder* que reconstrói $y = g(h)$.

O *autoencoder* é um tipo de rede neural treinada de forma não supervisionada voltada ao aprendizado de novas representações de um conjunto de dados através da reconstrução da entrada. Para isso, um AE é composto de um *encoder*, que transforma a entrada original \mathbf{x} em uma representação h em um espaço latente de menor dimensionalidade, e um *decoder*, que gera uma versão reconstruída da entrada \mathbf{x}' a partir da representação latente (Zhao et al. 2019). A Figura 4.5 ilustra um *autoencoder* raso, contendo uma única camada oculta.

A arquitetura básica de um *autoencoder* é apresentada na Figura 4.5. Em resumo, uma rede *autoencoder* aprende a definir $g(f(\mathbf{x})) = \mathbf{x}'$.

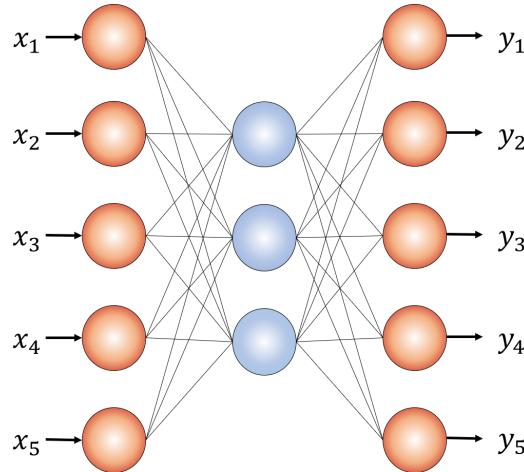


Figura 4.5: Arquitetura básica de uma Rede *Autoencoder*.

autoencoders generalizaram a ideia de um codificador e um decodificador além das funções determinísticas para mapeamentos estocásticos $P_{encoder}(h|x)$ e $P_{decoder}(x|h)$. Tradicionalmente, as redes AE são usadas para redução de dimensionalidade e/ou aprendizado de características.

Os AE podem ser considerados como um caso especial de redes *feedforward* e podem ser treinados com todas as mesmas técnicas, tipicamente descendente gradiente *minibatch* seguindo gradientes computados por retropropagação.

Ao contrário das redes *feedforward* em geral, os *autoencoders* também podem ser treinados usando a recirculação (ao contrário das redes *feedforward* em geral, os AE também podem ser treinados usando a recirculação (E. Hinton e McClelland 1987), um algoritmo de aprendizagem baseado na comparação das ativações da rede na entrada original.), um algoritmo de aprendizagem baseado na comparação das ativações da rede na entrada original para as ativações na entrada reconstruída. A recirculação é considerada mais biologicamente plausível do que a retro-propagação, mas raramente é usada para aplicações de aprendizado de máquina.

4.1.10 Deep Autoencoder

Para o desenvolvimento deste trabalho, utilizou-se o *Deep Autoencoder* (DAE), na fase de agrupamento profundo. O DAE gera um espaço de características latente (Z) reduzindo a dimensionalidade do problema 4.6. Utiliza-se esses espaço Z no agrupamento ao invés de utilizar na dimensão inicial.

Segundo Hinton e Salakhutdinov (2006), um *deep autoencoder* profundo pode ser visto como uma generalização não linear da análise de componentes principais (PCA, do inglês *Principal Component Analysis*), capaz de solucionar as limitações do mesmo quanto à análise de conjuntos de dados contendo relações não lineares. Além disso, por serem modelados de forma paramétrica, AE's oferecem maior flexibilidade em termos de projeto e manutenção, podendo ser aplicados em casos nos quais se trabalha com quantidades extensas de dados.

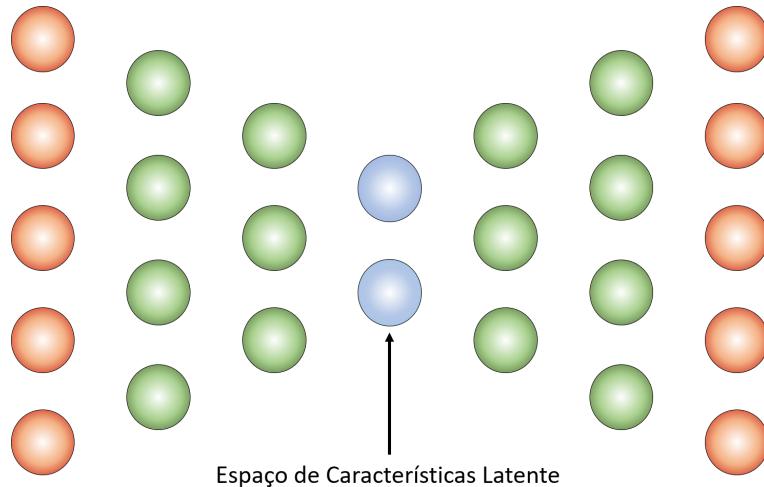


Figura 4.6: Arquitetura de uma *Deep Autoencoder*.

A rede *Deep Autoencoder* fornece mapeamentos flexíveis em ambos os sentidos. O tempo de aprendizagem é linear no número de casos de treinamento, e o modelo de codificação final é bastante compacto e rápido. No entanto, pode ser muito difícil otimizar DAE's usando *backpropagation*. Com pequenos pesos iniciais, o gradiente do *backpropagation* morre. Para tentar resolver isso, utiliza-se o pré-treinamento camada-por-camada sem supervisão ou apenas inicializando os pesos com uma melhor heurística de escolha de pesos iniciais.

Diversas variantes do AE foram criadas visando otimizar estender a sua aplicabilidade, dentre as quais se destacam três básicas, segundo Zhao et al. (2019):

Visando otimizar e estender sua aplicabilidade, diversas variantes do AE foram desenvolvidas ao longo dos anos (Baldi 2012), dentre as quais destacam-se três básicas (Zhao et al. 2019), são elas:

- ***Sparse autoencoder***: nesse tipo de AE existe um termo regularizador que é adicionado à função de erro para não permitir que o AE aprenda a identidade da entrada x . Camadas de *dropout* também podem ser usadas para regularização.
- ***Denoising autoencoder***: nesse AE, é adicionado um ruído parcial ou corrupção do sinal de entrada por meio de *dropout*. Desta forma, o AE é condicionado à aprender representações mais robustas dos dados (Vincent et al. 2008).
- ***Stacked autoencoder***: *encoders* provenientes de um conjunto de AE's treinados sequencialmente são empilhados, formando uma estrutura profunda em que a representação codificada do *autoencoder* l é usada como entrada do AE $l + 1$ (Ma et al., 2018). Trata-se do método de treinamento camada por camada (Ma et al. 2018).

4.1.11 Convolutional Autoencoder (CAE)

O convolutional autoencoder (CAE) é um tipo do AE criada especialmente para o processamento de imagens e outros dados multidimensionais. Assim como um AE padrão,

o CAE possui um *encoder* e um *decoder*, entretanto formados por camadas de convoluções e *MaxPooling*. Assim, o CAE tende a ter um desempenho melhor em relação aos AE's convencionais por incorporar relações espaciais entre pixels em uma imagem. A Figura 4.7 apresenta um exemplo de um CAE.

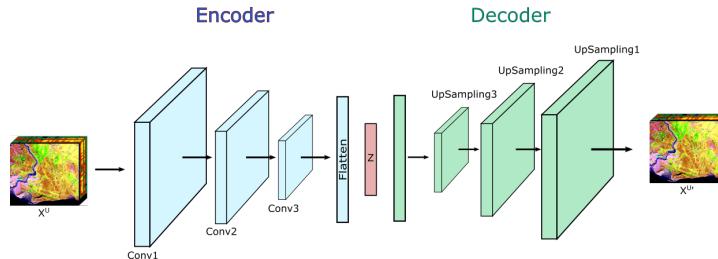


Figura 4.7: Exemplo de uma rede tipo CAE

De maneira similar à classificação com AE, a parte do encoder pode ser achatada e processada por uma camada que retorna a probabilidade referente a cada classe. Além disso, é possível notar que, durante a fase de refinamento, a junção do encoder e da camada de classificação configura uma topologia CNN típica, porém com parâmetros inicializados de acordo com os valores obtidos no processo de pré-treinamento (Goodfellow et al. 2016).

A compressão dos dados de entrada é realizada no encoder pelas operações de convolução e de subamostragem. No decoder, a descompressão é feita através das operações de convolução ou deconvolução, e de sobreamostragem. A deconvolução pode ainda ser vista como uma convolução transposta, em que se utiliza um filtro resultante da transposição e da inversão do tensor referente ao filtro de convolução original (Aggarwal 2018).

A exemplo da CNN, existem diversos modos de se estruturar uma rede do tipo CAE. Em algumas arquiteturas, a amostragem não é aplicada logo após cada camada convolucional, e sim somente após múltiplas convoluções. Na maioria dos modelos, o número de filtros nas camadas convolucionais aumenta conforme se aproxima do gargalo, no entanto, há casos em que o número de filtros diminui entre a entrada da rede e o gargalo, e aumenta entre o gargalo e a saída.

O principal benefício do convolutional autoencoder pode ser observado quanto à capacidade de otimização dos pesos e vieses. Os experimentos conduzidos por Masci et al. (2011) mostraram que CNN's pré-treinadas com o auxílio de CAE's possuem uma leve, porém consistente vantagem em relação às redes inicializadas com parâmetros aleatórios.

4.1.12 Algoritmo *K-means*

O algoritmo *k-means* é um método de agrupamento particional, primeiramente apresentando por MacQueen (1967). O objetivo é partitionar os dados em m grupos mutualmente exclusivos e indicar a qual grupo cada elemento pertence (possui maior similaridade). Este método utiliza uma medida de similaridade para encontrar os elementos mais similares para um determinado grupo. Deve-se indicar o número m , que é a quantidade de grupos a serem criados a partir do conjunto de dados de entrada.

Para o algoritmo *k-means*, têm-se X uma base de dados contendo N elementos em um espaço Euclidiano, os métodos particionais organizam os objetos de X em m grupos, g_1, \dots, g_m , de modo que $g_i \subset X$ e $g_i \cap g_j = \emptyset$ para $1 \leq i, j \leq m$ e $i \neq j$, restrito a $m \leq N$. Assim, cada elemento é associado à apenas um grupo (Han 2005).

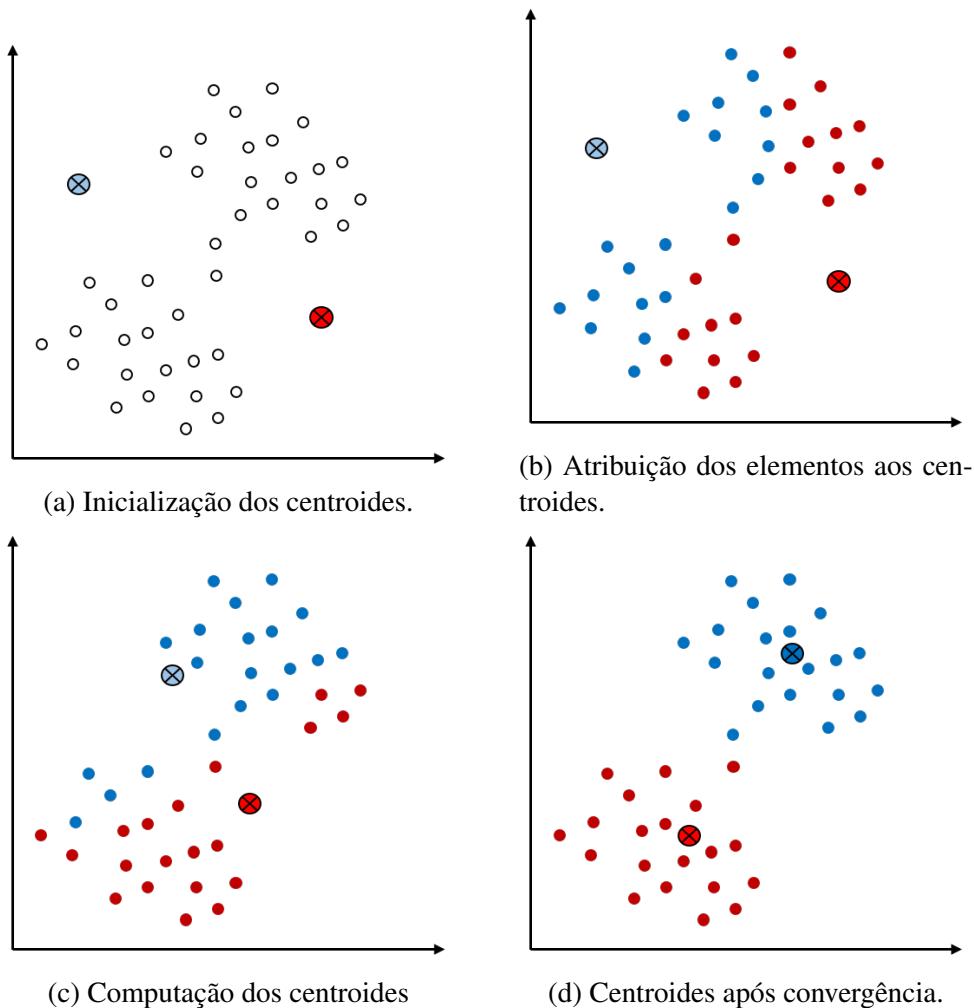


Figura 4.8: *k-means* aplicado a um problema \mathbb{R}^2 com dois grupos. Fonte: Lima (2015)

Segundo Manning et al. (2008), o *k-means* tem como objetivo minimizar a média quadrática da distância Euclidiana (Equação 4.6) entre os centroides de cada grupo e seus respectivos elementos, como mostra na Equação 4.5.

$$J = \sum_{i=1}^k \sum_{x_j \in g_i} D(x_j, \mu_i)^2 \quad (4.5)$$

Onde cada centroide μ de um dado grupo g_i é definido pela média dos elementos que o formam, representados pela Equação 4.7. $D(x_j, \mu_i)$ é a distância euclidiana entre a amostras $x_j \in g_i$ para o centrodides μ_i .

$$D(a, b) = \sqrt{\sum_{i=1}^n (a_i - b_i)^2} \quad (4.6)$$

$$\mu = \frac{1}{n_i} \cdot \sum_{x \in g_i} x \quad (4.7)$$

Em que n_i é número de elementos de cada grupo g_i . Inicialmente, os m centroides são definidos de forma aleatória. Em seguida, cada exemplo, definido por x_i é atribuído ao grupo que possui o centroide mais próximo. Nas iterações seguintes os centroides são recalculados conforme a Equação 4.7 e os elementos são novamente reatribuídos aos centroides mais próximos. Esse processo é repetido até que os centroides não sofram mais alterações, significando que o algoritmo convergiu.

A Figura 4.8 apresenta como o *k-means* se comporta no processo de agrupamento. Na Figura 4.8a, representa um conjunto de dados e dois centroides (bolinhas vermelha e azul) inicializados aleatoriamente. A Figura 4.8b mostra cada elemento sendo atribuído ao centroide mais próximo. Em seguida, na Figura 4.8c, os centroides são recalculados de acordo com os grupos definidos na primeira iteração. A reatribuição dos elementos e o recálculo dos centroides se repetem até que o algoritmo tenha convergido (Figura 4.8d).

Algoritmo 2: Algoritmo *K-means*.

Result: G

Entrada: X, k

Saída : $G_{i=1}^k$

- 1 Definir o conjunto de centroides $\mu_{i=1}^k$;
 - 2 **while** *Não convergiu* **do**
 - 3 **for** *cada* $x \in X$ **do**
 - 4 | Atribuir x definir o centroides mais próximo;
 - 5 **end**
 - 6 Recalcular os novos centroides;
 - 7 **end**
-

O Algoritmo 2 representa o procedimento realizado no *K-means*. Onde X é o conjunto de entradas, $\mu_{i=1}^k$ é o conjunto de k centroide, onde cada grupo tem um centroides associado. E ao final, têm-se um conjunto de grupos $G_{i=1}^k$.

Onde g_i é um determinado grupo pertencente à $G_{i=1}^k$, μ_i é o centroide referente ao g_i .

4.1.13 *K-means++*

Embora exista inúmeros trabalhos que mostram que o algoritmo *K-Means* sempre termina, ele não necessariamente encontra a configuração ótima de grupos e, também, é bastante sensível ao conjunto de centroides inicialmente escolhidos. Para evitar um possível efeito negativo devido à escolha aleatória dos centroides iniciais, Arthur e Vassilvitskii

(2006) propuseram o *K-means++*, que é uma alteração do *k-means* onde os centroides iniciais não são escolhidos aleatoriamente.

De acordo com Arthur e Vassilvitskii (2006) a ideia da modificação do algoritmo *K-Means++* é selecionar um bom conjunto de centroides iniciais. O algoritmo *K-Means++* só difere do algoritmo *K-Means* na escolha inicial dos centroides que é feita usando uma distribuição de probabilidades ponderada na qual uma instância x é escolhida com probabilidade proporcional ao quadrado de sua distância ao centroide mais próximo. As vantagens encontradas com as modificações atribuídas ao K-Means++ em relação ao K-Means são:

- Melhoria no tempo de execução;
- Melhoria na qualidade do resultado;
- Melhoria nos resultados à medida que o número de grupos aumenta.

Algoritmo 3: Algoritmo *K-means++*.

Result: G

Entrada: X, k

Saída : $G_{i=1}^k$

- 1 Pegue um centroide μ_1 , escolhido uniformemente ao acaso em X ;
- 2 **while** *Não foram escolhidos k centroides* **do**
- 3 | Pegue um novo centroide μ_i , escolhendo $x \in X$ com probabilidade $\frac{D(x)}{\sum_{x \in X} D(x)^2}$
- 4 **end**
- 5 Proceda com o algoritmo *k-means* padrão.

O algoritmo 3 define o *K-means++*, onde $D(x)^2$ denota a distância mais curta de um ponto de dados ao centro mais próximo que já escolhido.

4.1.14 Janelas de Parzen

Para se trabalhar com a divergência de *Kullback-Leibler* e a entropia cruzada, torna-se necessário a estimativa não-paramétrica para estimar a função de probabilidade. Fazemos isso direto das amostras utilizando a Janela de Parzen.

O método Janelas de Parzen foi proposto por (Parzen 1962) com o objetivo de definir a função de densidade a partir dos próprios dados, tornando-a assim uma técnica chamada não-paramétrica.

Esta técnica faz basicamente uma interpolação de pontos, ou seja, dada uma amostra de uma variável aleatória, $x \in \mathbb{R}^d$, o objetivo da janela de parzen é estimar uma função densidade de probabilidade (FDP) $f(x)$ de onde essa variável foi retirada. A essência da técnica consiste em sobrepor funções centradas em cada uma das observações, de modo que cada observação, $x_i \in x$ contribui na construção da $f(x)$. A Equação 4.8 formaliza a janela de parzen.

$$P(x) = \frac{1}{n} \sum_{i=1}^n \frac{1}{h_n^d} K\left(\frac{x-x_i}{h_n}\right) \quad (4.8)$$

Em que, $h_n > 0$ é o parâmetro correspondente à "largura" da janela e $K(x)$ é a função janela utilizada obedecendo à restrição:

$$\int_{\mathbb{R}^d} K(x) dx = 1 \quad (4.9)$$

Em geral, as funções de densidades de probabilidades são escolhidas para serem utilizadas como funções janelas, pois garantem que as densidades estimadas obedeçam às propriedades de uma FDP.

Em Parzen (1962) a janela foi definida como um hipercubo centrado em cada ponto e a contribuição é dada pelo número de pontos que se encontram dentro da janela.

4.2 Base de Dados Utilizadas

Para a realização dos experimentos foram utilizadas bases de dados que podem ser encontradas no repositório (Bache e Lichman 2013). Foram escolhidas quatro categorias de bases de dados.

4.2.1 Bases de Benchmark

Base Epilepsia

A base de dados EPILEPSIA contém dados sobre pessoas que sofrem de epilepsia. Consiste em um total de cinco classes e é formado por 4097 pontos de dados (séries temporais). Cada ponto de dados é o valor de registro da eletroencefalografia em momentos diferentes; São obtidos 500 indivíduos, cada um com 4097 pontos de dados para 23,5 s.

Dividiu-se e embaralhou-se cada 4097 pontos de dados em 23 blocos, cada bloco contém 178 pontos de dados por 1 segundo e cada ponto de dados é o valor do registro de EEG em um ponto diferente no tempo. Portanto, agora temos $23 \times 500 = 11.500$ informações (linha), cada informação contém 178 pontos de dados por 1 segundo (coluna), a última coluna representa o rótulo y 1,2,3,4,5.

Assim, a base EPILEPSIA é formada por um total de 11500 amostras sendo 178 características, onde cada amostra é referente à um paciente.

Base Pendigits

O conjunto de dados *Pendigits* original (Reconhecimento Baseado em Caneta de Dígitos Manuscritos) é um conjunto de dados de classificação multiclasse com 16 atributos inteiros e 10 classes (0 a 9). O banco de dados de dígitos é criado com a coleta de 250 amostras de 44 escritores. Neste conjunto de dados, todas as classes têm frequências iguais. Assim, o número de objetos em uma classe (correspondente ao dígito "0") é reduzido por um fator de 10%.

Base Reuters

Criado por Lewis (1997), o conjunto de dados Reuters é uma coleção de 10.788 documentos do serviço de notícias financeiras da Reuters, dividida em um conjunto de treinamento com 7769 documentos e um conjunto de teste com 3019 documentos., no idioma inglês. Contendo 10.788 em formato de arquivo TSV.

Formada por 90 classes, os documentos foram publicados na agência de notícias Reuters em 1987. Os documentos foram reunidos e indexados com categorias por pessoal da Reuters Ltd.

Base Gás

Esta base de dados é formada por 13910 medições de 16 sensores químicos utilizados em simulações para compensação de deriva em uma tarefa de discriminação de 6 gases em vários níveis de concentrações. O conjunto de dados foi coletado no período de janeiro de 2007 a fevereiro de 2011 (36 meses) em uma instalação de plataforma de distribuição de gás situada no Laboratório ChemoSignals no Instituto de BioCircuitos, Universidade da Califórnia em San Diego. Sendo totalmente operado por um ambiente totalmente computadorizado - controlado por um software LabVIEW - National Instruments em um PC equipado com as placas de aquisição de dados seriais apropriadas. O conjunto de dados resultante compreende registros de seis substâncias gasosas puras distintas, ou seja, amônia, acetaldeído, acetona, etileno, etanol e tolueno, cada um dosado em uma ampla variedade de valores de concentração variando de 5 a 1000 ppmv. Com os dados, foi formado um total de 13.910 amostras e 128 características, sendo 6 classes (1: Etanol; 2: Etileno; 3: Amônia; 4: Acetaldeído; 5: Acetona; 6: Tolueno).

4.2.2 Bases de Imagens

Base MNIST

O conjunto de dados MNIST é um dos conjuntos de dados mais comuns usados para classificação de imagens e acessível a partir de várias fontes diferentes. A base MNIST contém 70.000 imagens de números escritos à mão, obtidas de funcionários do *American Census Bureau* e de estudantes americanos do ensino médio.

Cada amostra da MNIST é uma imagem de um dígito, que vão de 0 à 9, como mostra a Figura 4.9. Cada imagem possui escala de cinza 28 x 28, associada a uma classe (0 à 9).

A base MNIST possui uma variação, neste trabalho chamada de MNIST 64. Trata-se da mesma base MNIST, porém com apenas 1700 amostras, sendo cada em escala de cinza 8 x 8. Esta versão pode ser encontrada na API da biblioteca Scikit-learn na linguagem Python (Pedregosa et al. 2011). Neste trabalho foram utilizadas as duas versões nos experimentos.

Base Fashion MNIST

Fashion-MNIST é um conjunto de dados das imagens consistindo um total de 70 mil amostras. Cada amostra é uma imagem em escala de cinza 28 x 28, associada a um rótulo

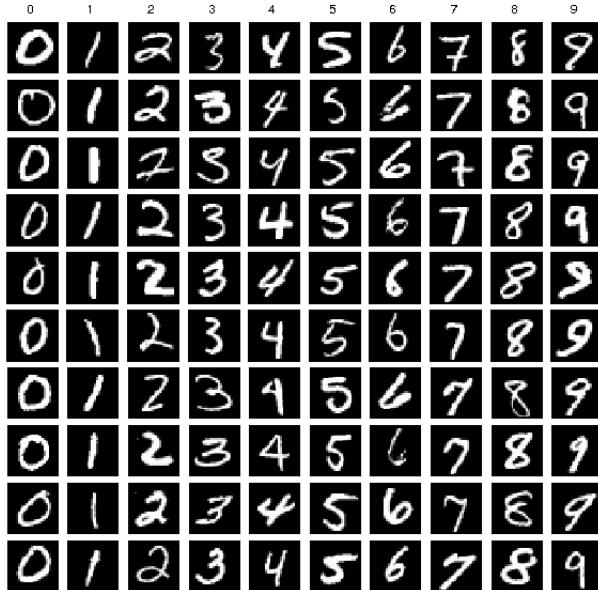


Figura 4.9: Visualização das Amostras da Base MNIST.

de 10 classes. O objetivo desta base é que a *Fashion MNIST* seja um substituto direto do conjunto de dados MNIST original para algoritmos de *benchmarking* no contexto de aprendizado de máquina.

A Figura 4.10 apresenta a visão geral da base com imagens de todas as classes. A base Fashion MNIST é formada com dez classes: camiseta, calças, camisa pulôver, vestido, casaco, sandália, camisa, tênis, bolsa, bota no tornozelo.

Base CIFAR-10

O conjunto de dados CIFAR-10 consiste em 70.000 imagens coloridas de 32x32 em 10 classes, com 7.000 imagens por classe. O conjunto de dados é dividido em cinco lotes de treinamento e um lote de teste, cada um com 10.000 imagens. O lote de teste contém exatamente 1000 imagens selecionadas aleatoriamente de cada classe. Os lotes de treinamento contêm as imagens restantes em ordem aleatória, mas alguns lotes de treinamento podem conter mais imagens de uma classe do que de outra. Entre eles, os lotes de treinamento contêm exatamente 5.000 imagens de cada aula. A Figura 4.11 apresenta algumas amostras da base CIFAR-10.

Base SLT-10

O conjunto de dados STL-10 é um conjunto de dados de reconhecimento de imagem para o desenvolvimento de aprendizado de recursos não supervisionado, aprendizado profundo e algoritmos de aprendizado autodidata. É inspirado no conjunto de dados CIFAR-10, mas com algumas modificações. Em particular, cada classe tem menos exemplos de treinamento rotulados do que no CIFAR-10, mas um grande conjunto de exemplos não rotulados é fornecido para aprender modelos de imagem antes do treinamento supervi-

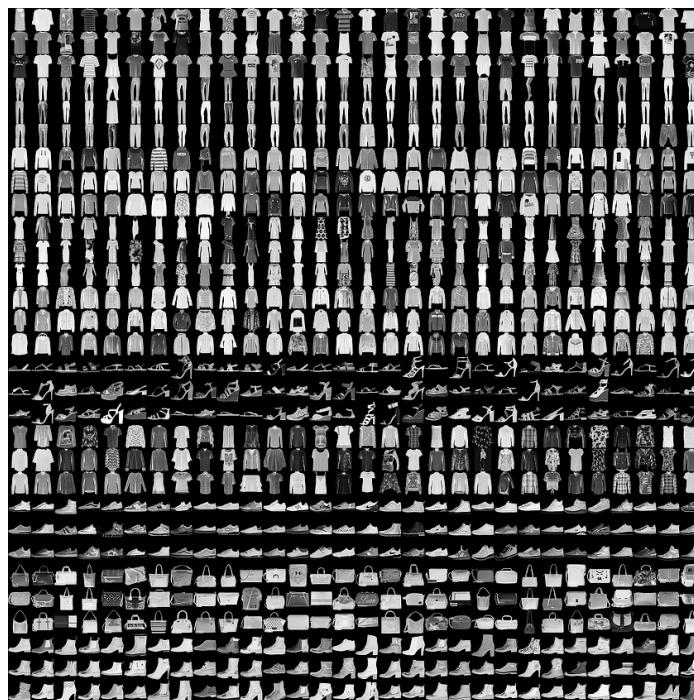


Figura 4.10: Visualização das Amostras da Base Fashin MNIST.

sionado. A base SLT-10 é formada por um total de 60.000 imagens, sendo 10 classes. Algumas amostras podem ser vistas na Figura 4.12.

Os detalhes sobre todas as bases de dados utilizadas nos experimentos são mostrados na Tabela 4.1.

Tabela 4.1: Detalhes das bases de dados utilizadas nos experimentos

Base de Dados	Amostras	Classes	Características
EPILEPSIA	11.500	5	179
REUTERS	11.228	46	500
GÁS	13.910	6	128
PENDIGITS	10.992	10	16
MNIST	70.000	10	784
FASHION MNIST	70.000	10	784
CIFAR-10	60.000	10	452
STL-10	13.000	10	1424

4.3 Métodos de Avaliação Utilizados

Para avaliar o desempenho dos algoritmos na tarefa de rotulação e de classificação foram utilizadas alguns métodos de avaliação comumente abordados por trabalhos encontrados na literatura. Nessa seção são apresentadas as métricas de avaliação dos resultados

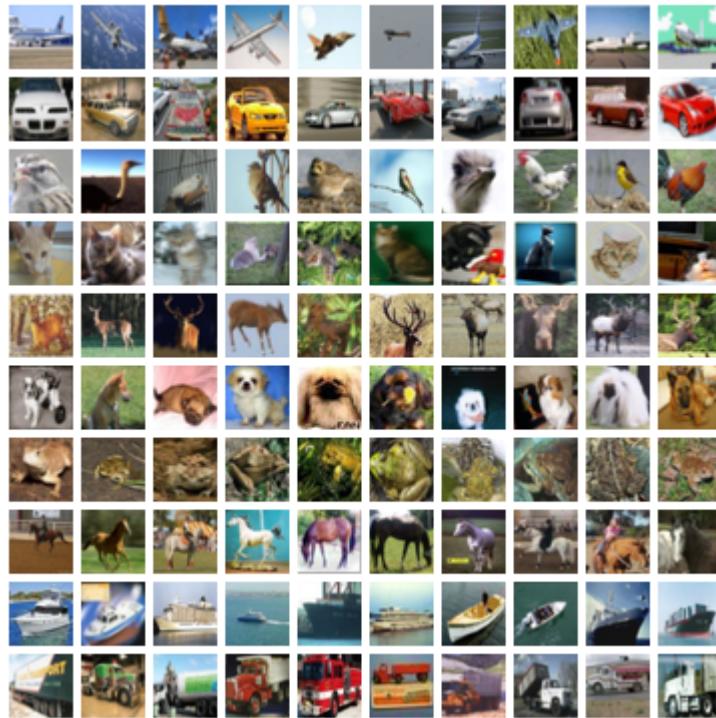


Figura 4.11: Visualização das Amostras da Base CIFAR10.

utilizadas na execução deste trabalho.

4.3.1 Matriz de Confusão

A matriz de confusão é uma tabela que organiza e mostra o resultado obtido com a classificação, comparando-o com o resultado esperado. A Tabela 4.2 é um exemplo de uma matriz de confusão para uma classificação binária.

Nota-se que a matriz de confusão é formada por 4 valores: verdadeiro positivo (VP), falso positivo (FP), falso negativo (FN) e verdadeiro negativo (VN), além de a e b que representam as classes hipotéticas. Sendo:

- VP é o número de objetos da classe a classificados como da classe a;
- FP é o número de objetos da classe b classificados com a;
- FN é o número de objetos da classe a classificados como b;
- VN é o número de objetos da classe b classificados como da classe b.

Tabela 4.2: Exemplo para matriz de confusão para a classificação binária.

	a	b
a	VP	FP
b	FN	VN

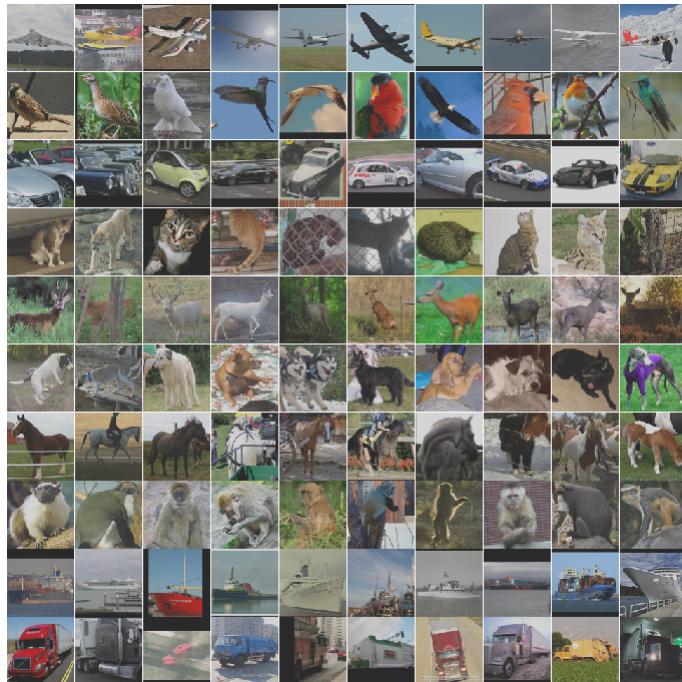


Figura 4.12: Visualização das Amostras da Base SLT10.

A matriz de confusão apresentadas na Tabela 4.2 é utilizada quando o problema de classificação binária, ou seja, existe apenas duas classes. Entretanto neste trabalho a maior das bases de dados utilizadas são formadas por mais de duas classes (classificação multiclasse). Neste caso a matriz de confusão também é aplicável, porém para definir os valores VP, FP, FN e VN, deve-se considerar classe por classe. A Tabela 4.3 mostra uma matriz de confusão para um problema de 3 classes.

Tabela 4.3: Exemplo para matriz de confusão para três classes.

	a	b	c
a	VP	FP	FP
b	FN	VN	
c	FN		VN

A matriz de confusão da Tabela 4.3 apresenta um problema com três classes (a,b e c), com os valores (VP, FP, FN e VN) determinados apenas para a classe a. De forma análoga, pode-se calcular tais valores para as classes b e c. Nos dois casos, classificação binária ou multiclasse, os valores VP, FP, FN e VN são utilizados para calcular a acurácia (A), precisão (P), *recall* (R) e F-Measure (FM) (Powers 2007), que também são utilizadas na avaliação dos resultados.

4.3.2 Acurácia - A

A acurácia de um classificador é a porcentagem de casos correspondentes classificados em um conjunto de teste. A partir dessa medida é possível determinar a qualidade do classificador quanto ao reconhecimento de instância de diversas classes, i.e., basicamente calcula a porcentagem de acerto do classificador no conjunto de teste. Para a classificação binária a acurácia pode ser calculada a partir da Equação 4.10 (Sokolova e Lapalme 2009).

$$A = \frac{VP + VN}{VP + FP + FN + VN} \quad (4.10)$$

Quando têm-se uma classificação multiclasse, para um classe individual c_i , a avaliação é definida por VP_i , FN_i , VN_i e FP_i , acurácia (A_i), precisão (P_i), *recall* (R_i) e são calculadas a partir da contagem de c_i . Desta forma, a Equação 4.11 é utilizada para calcular a acurácia em uma classificação multiclasse, considerando n quantidade de classes.

$$A = \frac{\sum_{i=1}^n \frac{VP_i + VN_i}{VP_i + FN_i + FP_i + TN_i}}{n} \quad (4.11)$$

4.3.3 Precisão - P

A precisão é uma medida que reflete a proporção de verdadeiros positivos em relação a todas as previsões positivas. Essa medida mostra a quantidade de elementos de uma dada classe classificados corretamente em relação a todos os objetos classificados como sendo da classe. Isso significa que quanto maior a precisão, melhor é a classificação considerando as classes. A precisão em uma classificação binária pode ser calculada a partir da Equação 4.12 (Sokolova e Lapalme 2009).

$$P = \frac{VP}{VP + FP} \quad (4.12)$$

4.3.4 Recall - R

O *recall* é uma medida que reflete a proporção de verdadeiros positivos em relação a previsões positivas e as incorretas previsões negativas. Desta forma é possível ver o comportamento dos elementos classificados em a, ou seja, de todos os objetos da classe a, quantos foram classificados como a. Para a classificação binária o *recall* pode ser calculado a partir da Equação 4.13.

$$R = \frac{VP}{VP + FN} \quad (4.13)$$

4.3.5 F-Measure - FM

A F-measure é uma medida que depende da precisão e do *recall*. Em outras palavras, só produz resultados satisfatórios se essas duas taxas forem equilibradas. Desta forma, essa medida permite avaliar os resultados (Chimieski e Fagundes 2013). O resultado da FM é

dado entre 0 e 1, portanto, quanto mais próximo de 1 melhor é o resultado da classificação. Para o tipo binário pode ser calculado a partir da Equação 4.14.

$$FM = \frac{2 * P * R}{P + R} \quad (4.14)$$

4.3.6 Acurácia de Agrupamento

Para demonstrar a eficácia do agrupamento profundo, usou-se a métrica de acurácia de agrupamento (ACC) (Yang et al. 2010). Definiu-se o número de grupos com o número de classes para avaliar o desempenho o desempenho. A acurácia de agrupamento é descrita como mostra a Equação 4.15.

$$ACC = \max_m \frac{\sum_{i=1}^n 1\{l_i = m(c_i)\}}{n} \quad (4.15)$$

Onde l_i é o rótulo verdadeiro, c_i é a atribuição de grupo produzida pelo algoritmo e m varia sobre todos os mapeamentos um-para-um possíveis entre grupos e rótulos. Intuitivamente, essa métrica pega uma atribuição de grupo de um algoritmo não supervisionado e uma atribuição de verdade fundamental e, em seguida, encontra a melhor correspondência entre eles. O melhor mapeamento pode ser calculado com eficiência pelo algoritmo de Kuhn (1955).

4.3.7 Índice Kappa

O índice Kappa, avalia a porção de ocorrências que podem ser atribuídas ao próprio classificador, excluindo ocorrências aleatórias, relativas a todas as classificações que não podem ser atribuídas apenas ao acaso. O índice Kappa varia de -1 (desacordo total) a 0 (classificação aleatória) a 1 (concordância perfeita). Para problemas de várias classes, o índice kappa é um medidor muito útil, porém simples, para medir a precisão de um classificador e compensar sucessos aleatórios. O índice Kappa pode ser interpretado seguindo a Tabela 4.4.

Tabela 4.4: Kappa Index Interpretation Table

KAPPA	Interpretation
< 0	Poor
0.01 - 0.20	Weak
0.21 - 0.40	Regular
0.41 - 0.60	Moderate
0.61 - 0.80	Great
0.81 - 1.00	Very Great

4.3.8 Validação Cruzada

Validação cruzada é uma técnica para avaliar modelos de AM por meio de treinamento de vários modelos de AM em subconjuntos de dados de entrada disponíveis e avaliação deles no subconjunto complementar dos dados.

A Figura 4.13 representa visualmente o funcionamento da validação cruzada. O conjunto de dados total é subdividido em n subconjuntos, em que cada um destes, são particionados em treinamento e teste.

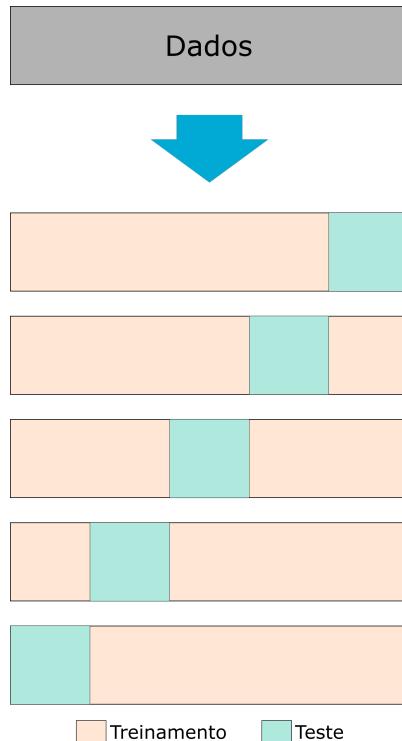


Figura 4.13: Representação visual da validação cruzada.

A cada iteração, utiliza-se um o conjunto de treinamento para treinar um modelo, e posteriormente obtém a predição do conjunto de teste a partir do modelo treinamento. Esse procedimento é realizado n vezes, dependendo da quantidade de subconjuntos definidos.

A validação cruzada permite que o modelo seja treinado e testado com toda a base de dados sem que uma determinada amostra esteja duas ou mais vezes no conjunto de treinamento ou teste. A cada iteração calcula-se o valor da métrica escolhida para avaliar, para posteriormente calcular a média.

4.3.9 Busca em Grade

Técnica de ajuste de hiperparâmetros que pode facilitar a construção de um modelo preditivo e avaliar cada combinação de parâmetros de algoritmo por grade.

Essa técnica aplica diferentes valores dos hiperparâmetros no modelo que se deseja treinar. Em seguida, ele calcula o erro para cada um desses valores de hiperparâmetros,

obtendo assim os melhores valores. Na Busca em grade, todas as misturas de combinações de hiperparâmetros passarão uma a uma para o modelo e verificarão a pontuação de cada modelo, baseado em uma métrica de avaliação.

4.4 Teste de Kruskal-Wallis

O teste de Kruskal-Wallis foi criado por William Kruskal (1919 – 2005), matemático e estatístico americano e por W. Allen Wallis (1912 – 1998) economista e estatístico americano.

O teste de Kruskal-Wallis não trabalha com as hipóteses de comparação dos parâmetros, não testa a hipótese de igualdade de médias e nem testa a igualdade de medianas, como muitos acreditam. O teste de Kruskal-Wallis é indicado para testar a hipótese de que três ou mais populações têm distribuição igual ou não (Conover 1998).

Assim quando se aplica um teste de Kruskal-Wallis, no relatório a princípio, não deveria ser apresentado médias, medianas ou gráficos com essas estatísticas. O teste de Kruskal-Wallis trabalha com postos (rank) e não com os dados coletados.

Só a título de esclarecimento, existe um outro teste não paramétrico que também trabalha com a comparação de vários grupos, que é o teste de Friedman (Conover 1998). Este teste tem uma particular diferença, ele considera que cada grupo está usando os mesmos indivíduos em todos os tratamentos, técnica que é chamada de comparação de medidas repetidas, enquanto o Kruskal-Wallis considera que para cada grupo, os indivíduos são diferentes e independentes.

Há duas formas de calcular a estatística do teste H:

1. Quando há poucos ou nenhum empate dos ranks;
2. Quando há muitos empates (não há uma regra bem definida, mas geralmente > 3).

Para cada grupo obtém-se a soma dos ranks (r_{ij}).

$$R_i = \sum_{j=1}^{n_i} r_{ij} \quad (4.16)$$

Quando existe empates nos valores observados das amostras, ou o nº de empates é muito pequeno, como no exemplo (somente dois resultados empataram), a estatística de teste é:

$$H = \frac{12}{N(N+1)} \sum_{i=1}^k \frac{R_i^2}{n_i} - 3(N+1) \quad (4.17)$$

Onde,

- k = números de grupos;
- N = número total de medições experimentais;
- R_i = soma dos ranks de cada grupo;
- n_i = número de medidas em cada grupo;

- p = valor da estatística de Kruskal-Wallis

A hipótese nula deve ser rejeitada se o valor observado da estatística H for superior ao valor crítico (teste unilateral à direita), para isso se busca esse valor crítico na tabela de Qui-Quadrado (X^2). Desta forma, rejeita-se a hipótese nula, concluindo que existe pelo menos um grupo diferente.

4.5 Considerações

O modelo proposto neste trabalho utiliza um série de métodos para realizar o aprendizado semissupervisionado. Mostramos aqui neste Capítulo, os métodos de *Deep Learning* utilizados, no caso *Deep Autoencoder* convencional e *Convolucional Autoencoder*.

Foi apresentado também o algoritmo *K-means++* que é utilizado para inicializar os pesos da camada de rotulação no modelo proposto, como será discutido no Capítulo 5.

Foram mostradas também, as bases de dados que foram utilizadas nos experimentos realizados para validar o modelo proposto. Sendo estas bases em duas categorias: *benchmark*, que são as bases EPILEPSIA, REUTERS, GÁS e PENDIGITS; e bases de imagens MNIST, FASHION MNIST, CIFAR-10 e SLT-10.

E por fim, mostrou-se as métricas que foram adotadas para comparar o modelo proposto com outros modelos semissupervisionados, tanto da literatura atual quanto os clássicos do ASS.

Capítulo 5

Modelo Proposto

Neste capítulo são descritos detalhadamente os conceitos e definições que compõem o classificação semissupervisionada proposta neste trabalho, bem como o algoritmo de treinamento. O modelo proposto trabalha com auto-rotulação de maneira semissupervisionada para treinar um classificador utilizando dados rotulados e não rotulados simultaneamente. Dado um conjunto de dados $X = \{X^L, X^U\} \in \mathbb{R}^n$, em que $X^L = \{x_i, y_i\}|_{i=1}^L$ é o conjunto de dados rotulados e $X^U = \{x_i\}|_{i=1}^U$ o conjunto de dados não rotulados, onde $U >> L$, x_i é uma amostra qualquer de X e y_i é o rótulo respectivo da amostra.

A Figura 5.1 apresenta um fluxograma do funcionamento do modelo proposto neste trabalho. Nota-se que o modelo funciona basicamente em três etapas, inicialmente realiza um agrupamento, por meio de um *deep autoencoder* (DAE) ou convolucional *autoencoder* CAE, treinado com o conjunto X^U (não rotulados), inicializa-se os centroides da camada de rotulação, que é adicionada ao *encoder*, e por fim agrupa o conjunto X^U , por meio da otimização do *encoder* + camada de rotulação, utilizando as funções custos Entropia Cruzada e Divergência de Kullback-Leibler.

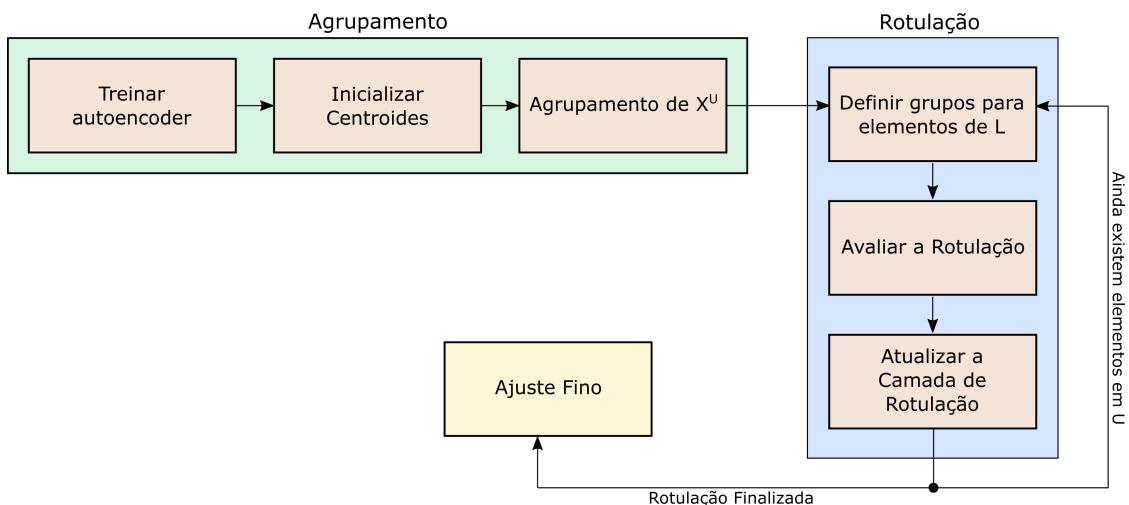


Figura 5.1: Fluxograma geral do treinamento do modelo proposto.

Então, o modelo segue com a rotulação, que são as etapas dentro do retângulo azul na Figura 5.1. Nesse ponto, são utilizados os dados rotulados (X^L) disponíveis na base de treino. Esses dados, são agrupados assim como o conjunto X^U , e em cada um dos grupos

tais dados exercem influência que rotulam os dados do grupo, ou dividem o grupo para tentar a rotulação em uma próxima iteração. E por fim, ocorre o ajuste fino da camada de rotulação, e define-se os rótulos de amostras que não conseguiram rótulos durante o treinamento.

Seja H um classificador modelado para um determinado problema e treinado por um conjunto de dados $D = \{(x_i, y_i)\}$, tal que D contenha elementos suficientes para treinar H e torná-lo capaz de aprender a relação entre os valores de entrada x e os de saída y , permitindo, partir do conhecimento contido em X^L , rotular o conjunto X^U (Zhu 2005).

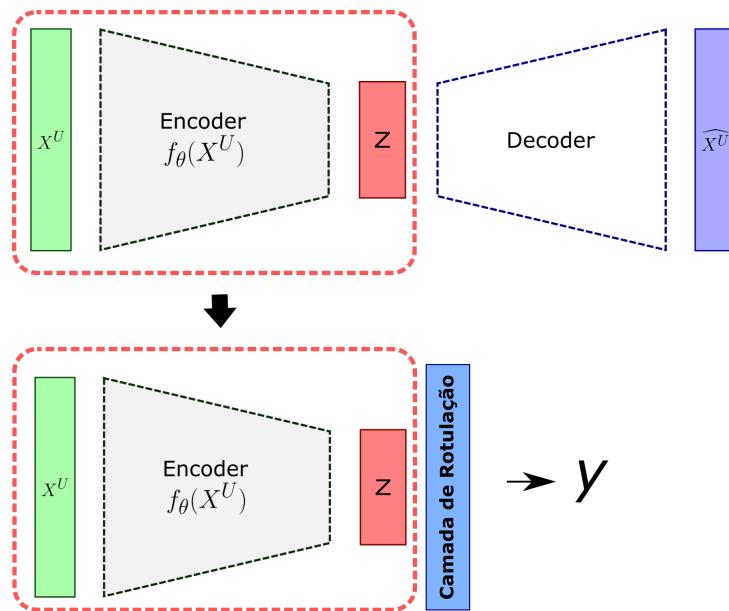


Figura 5.2: Representação visual do modelo proposto.

O modelo, denominado Auto-Rotulação Profunda (*Deep Self-Labeled -DSL*), trata-se um *Encoder*, retirado de um *Deep Autoencoder* (DAE) (Goodfellow et al. 2016) treinado com os dados não rotulados, com uma camada de rotulação acoplada, como mostrado na Figura 5.2. Posteriormente esse *Encoder* + camada de rotulação têm seus parâmetros ajustados com o algoritmo *backpropagation* otimizando função custo baseada em Teoria da Informação.

Inicialmente, treina-se o DAE para inicializar os seus respectivos pesos, como mostra na Figura 5.2. Depois, o *encoder* é usado para reduzir a dimensionalidade dos dados, e obter um classificador capaz de generalizar o problema e predizer futuras amostras não previstas.

Na literatura pode-se encontrar diversos trabalhos que mostram a eficiência da redução de dimensionalidade dos dados para problemas de agrupamento e classificação (Xie et al. 2016) (Ren et al. 2019) (Peng et al. 2020) (Maggia et al. 2020). No caso do DSL, utilizou-se dois autoencoders, o *Deep Autoencoder* (DAE) e *Deep Autoencoder Convolucional* (CAE), cada um em problemas distintos, como mostrado na Figura 5.3, onde X^U é o conjunto dos dados não rotulados, X^U' é a reconstrução da entrada gerada pelo AE, e no meio da rede, o conjunto Z que contém todos os dados de X^U com dimensão reduzida pelo *encoder*.

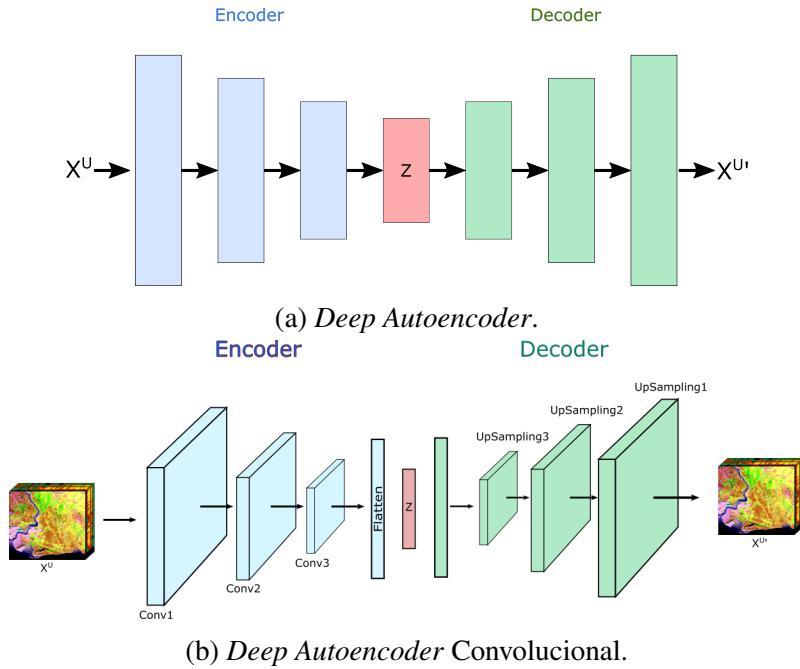
Figura 5.3: Estrutura do *autoencoder* utilizado no trabalho.

Figura 5.4: Fluxograma do treinamento do DSL.

O treinamento do modelo proposto (DSL) é baseado no algoritmo *Backpropagation*, utilizando técnicas de teoria da informação como função de custo, neste caso *Entropia Cruzada* e *Divergência de Kullback-Leibler*. O treinamento do DSL é desempenhado em três fases: **Agrupamento**, **Rotulação** e **Ajuste Fino**, como mostra a Figura 5.4. Ao final do treinamento, obtém-se um modelo capaz de generalizar o problema e um conjunto de dados totalmente rotulados.

5.1 Fase de Agrupamento

A fase de Agrupamento do DSL consiste em definir os parâmetros iniciais de um DAE ou um CAE. Utilizou-se *Deep Autoencoder* para inicializar uma função não linear (f_θ) e os centroides iniciais para a camada de rotulação a partir dos dados não rotulados (X^U). Esta etapa gera c grupos a partir de X^U , onde c é o número de classes no problema, sendo θ o conjunto de parâmetros do *autoencoder*. Pois pesquisas recentes mostraram eficiência do DAE ou CAE em produzir representações consistentes e semanticamente significativas e bem separadas em conjuntos de dados do mundo real (Alelyani et al. 2013) (Enguehard et al. 2019). Assim, a representação não supervisionada aprendida pelo AE naturalmente facilita a aprendizagem de representações para a fase de agrupamento.

Em outras palavras, o DAE é treinado utilizando os dados não-rotulados (X^U), afim de

obter o conjunto de parâmetros θ . A partir do θ , o DAE é capaz de transformar os dados X^U , usando um mapeamento não linear, para obter o espaço de características latente Z , gerando a função f_θ , como mostra a Equação 5.1

$$f_\theta : X^U \rightarrow Z \quad (5.1)$$

Com o objetivo de evitar que ocorra overfitting, utilizou-se *Dropout* como método de regularização da rede, tanto no DAE (Figura 5.3a) quanto no CAE (Figura 5.3b). De forma empírica, definiu-se valor de *Dropout* de 0.2 entre as camadas ocultas do DAE, ou seja, 20% das conexões são desconsideradas entre as camadas ocultas.

Para obter o conjunto de parâmetros θ , da função f_θ (Equação 5.1), utilizou-se o algoritmo *backpropagation*, explanado no Capítulo 3. O treinamento é realizado com base na função de custo Erro Quadrático Médio, como descrita na Equação 3.10 no Capítulo 3, e o otimizador Adam (Kingma e Ba 2014). O resultado final é um *autoencoder* multi-camadas com uma camada de codificação que reduz a dimensionalidade dos dados não rotulados, que serão utilizados na Fase de Agrupamento.

Após a inicialização dos parâmetros do DAE, o DSL agrupa os dados para realizar posteriormente a rotulação grupo à grupo. Utilizou-se o que foi chamado de Agrupamento Profundo (AP) do inglês *Deep Clustering* em vários trabalhos da literatura (Enguehard et al. 2019) (Xie et al. 2016). As características obtidas pela função f_θ (Equação 5.1) são usadas para particionar o conjunto $X^U = \{x_i\}_{i=1}^U$, onde $X_i = \{x_{i1}, x_{i2}, \dots, x_{id}\} \in \mathbb{R}^k$, no espaço Z , obtendo assim o conjunto de grupos $G = \{G_i\}_{i=1}^c$.

Assim, as características aprendidas são usadas para calcular o conjunto de c centroides μ no espaço de características Z , de tal modo que $\mu = \{\mu_i\}_{i=1}^c \in Z$. Esses centroides (μ) serão utilizados para formar os pesos iniciais da camada de rotulação (vide Figura 5.2), como mostrado na Figura 5.5.

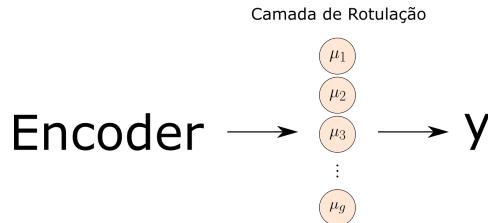


Figura 5.5: Estrutura da camada de rotulação.

Para cada $\mu_i \in \mu$ existe um neurônio na camada de rotulação. A ativação de cada neurônio vai depender da similaridade entre pesos do neurônio com a amostra transformada de X para Z . O neurônio maior similaridade é ativado e os demais desativados, desta forma definindo o grupo ao qual a amostra vai pertencer.

Para não selecionar aleatoriamente um conjunto de centroides, ou seja, pesos iniciais da camada de rotulação, optou-se em utilizar o algoritmo *k-means* (MacQueen 1967) para obtenção dos centroides iniciais do agrupamento. Neste caso, utilizou-se como método de inicialização dos centroides o *k-means++* (Arthur e Vassilvitskii 2006), que especifica um caminho direcionado para a escolha dos centros iniciais do agrupamento, desta forma obtendo melhores resultados no agrupamento.

O método k-means ++ escolhe inicialmente um centro (c_1) do conjunto de dados X aleatoriamente, e o próximo valor c_i é selecionado de acordo com a probabilidade na Equação 5.2, onde x é uma amostra no conjunto de dados X , e $D(x)$ denota a distância mais curta de um ponto de dados ao centro mais próximo que já escolhemos.

$$D(X) = \frac{D(x)^2}{\sum_{x \in X} D(x)^2} \quad (5.2)$$

Esse procedimento pode ser resumido na Figura 5.6. Em resumo, o *encoder* transforma os dados X^U para o espaço de características Z . Depois, utilizando o algoritmo *K-means++* calcula-se os centroides $\mu = \{\mu_i\}_{i=1}^c \in Z$, que serão utilizados como pesos iniciais da camada de rotulação que é acoplado no final do *encoder*.

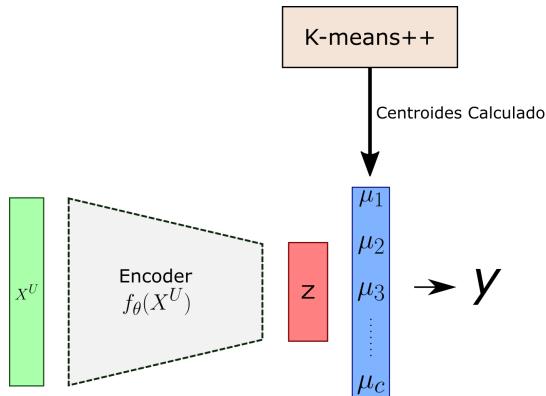


Figura 5.6: Fase agrupamento: Centroides calculado pelo k-means++ formam os pesos iniciais da camada de rotulação.

Neste trabalho, a quantidade de classes foi adotada como a quantidade inicial de grupos. Após o treinamento e agrupamento DAE, os elementos de uma mesma classe tendem a permanecer no mesmo agrupamento; portanto, menos etapas são necessárias para a fase de rotulagem. Na Tabela 5.1, pode-se observar um experimento empírico preliminar mostrando a acurácia do agrupamento quando utilizamos o algoritmo *k-means* comparado com o *Deep Clustering*.

Tabela 5.1: Comparação das acuráncias do *Deep Clustering* com o algoritmo *K-means* para a base de dados MNIST.

	MNIST	FASHION	EPILEPSIA	PENDIGITS
K-MEANS	0,884	0,799	0,865	0,898
Deep Clustering (KL)	0,954	0,842	0,951	0,954
Deep Custering (EC)	0,965	0,895	0,962	0,9745

Aplicou-se os métodos de agrupamento nas base MNIST, FASHION, EPILEPSIA e PENDIGITS, sendo a quantidade de grupos a mesma de classe da base. Nota-se que utilizando o *Deep Clustering*, minimizando no treinamento as funções custo *Kullback-leibler* e Entropia Cruzada, o *Deep Clustering* apresentou melhor resultados, quando comparado

em relação à acurácia do agrupamento. Como o DSL baseia-se no agrupamento, então este deve ser o mais consistente possível, que é o caso do *Deep Clustering*.

Após a definição os centroides iniciais, com o objetivo de refinar o agrupamento, utiliza-se a Divergência *Kullback Leibler* ou Entropia Cruzada para otimizar o agrupamento através do algoritmo *Stochastic Gradient Descent* (SGD). A Equação 5.3 define o agrupamento do modelo. Em que $G(X^U)$ é a função que define o agrupamento do conjunto de dados não rotulados X^U , que recebe como parâmetro os dados com dimensão reduzida para Z calculados pela função f_θ (Equação 5.1).

$$G(X^U) = g(f_\theta) \quad (5.3)$$

Como foi mostrado na Tabela 5.1, o agrupamento utilizando *Deep Learning* apresentou resultados satisfatórios quando utilizou-se técnicas de teoria da informação na função custo.

Na Equação 5.3, g é uma função que calcula a similaridade entre uma amostra $z_i \in f_\theta(X^U)$ e um centroide μ_j (calculado pelo *k-means++*). Esta similaridade entre um ponto $z_i \in Z$ e um centroide μ_j é medida por uma *Student's t-distribuition* como kernel, segundo a Equação 5.4, onde $z_i \in Z$ é $x_i \in X^U$ calculado por $f_\theta(x_i)$, α é o grau de liberdade da *Student's t-distribution* e q_{ij} é a probabilidade de uma amostras i ser atribuída à um grupo j . Como é realizado em Xie et al. (2016)

$$q_{ij} = \frac{(1 + \|z_i - \mu_j\|^2 / \alpha)^{-\frac{\alpha+1}{2}}}{\sum_{j'} (1 + \|z_i - \mu_{j'}\|^2 / \alpha)^{-\frac{\alpha+1}{2}}} \quad (5.4)$$

Após definir os parâmetros iniciais do modelo DSL (Equação 5.3), uma otimização do conjunto de parâmetros θ e os centroides $\{\mu_i\}_{i=1}^c$, consequentemente, obtendo o melhor resultado para o agrupamento do conjunto X^U . Para o processo de otimização, utilizou-se uma distribuição alvo auxiliar, que representa uma forte predição dos dados pois aumenta a influência dos dados atribuídos com alta confiança. Como realizado em (Xie et al. 2016), foi adotado $\alpha = 1$ para os experimentos neste artigo. Calculou-se o p_{ij} usando a Equação 5.5, onde $f_j = \sum_i q_{ij}$ é a frequência suave dos grupos.

$$p_{ij} = \frac{q_{ij}^2 / f_j}{\sum_{j'} q_{ij'}^2 / f'_j} \quad (5.5)$$

Dado q e p , define-se Divergência de Kullback-Leibler (KL) ou Entropia Cruzada (EC) como função custo do modelo na fase de inicialização. Dado modelo $g(f_\theta)$, otimiza-se os pesos usando o *Stochastic Gradient Descent* (SGD) (Xie et al. 2016). O gradiente de L com respeito ao espaço de características latente, cada exemplo z_i e μ_j são computados usando as Equações 5.6 e 5.7, onde $\partial L / \partial z_i$ é obtido pelo algoritmo *backpropagation* para computar o conjunto Θ .

$$\frac{\partial L}{\partial z_i} = \frac{\alpha + 1}{\alpha} \sum_j \left(1 + \frac{\|z_i - \mu_j\|^2}{\alpha}\right)^{-1} \cdot (p_{ij} - q_{ij})(z_i - \mu_j) \quad (5.6)$$

$$\frac{\partial L}{\partial \mu_j} = \frac{\alpha+1}{\alpha} \sum_i \left(1 + \frac{\|z_i - \mu_j\|^2}{\alpha}\right)^{-1} \cdot (p_{ij} - q_{ij})(z_i - \mu_j) \quad (5.7)$$

Por fim é obtido o conjunto de grupo $G = \{G_i\}_{i=1}^c$, onde c é o número de classes no problema.

5.2 Fase de Rotulação

Na fase de rotulação, utiliza os dados rotulados para definir uma classe para os elementos de cada grupo $G_i \in G$. Na fase da rotulação, o modelo é re-treinado para rotular os dados de X^U considerando os dados rotulados X^L . Nesta etapa, os grupos são analisados individualmente para definir a rotulação de seus elementos. Neste fase, os dados do conjunto X^U serão rotulados, na chamada fase transdutiva do modelo. A fase de rotulação é representada na Figura 5.7.

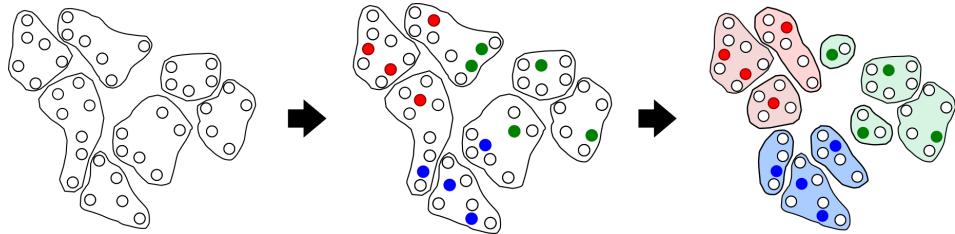


Figura 5.7: Representação visual de uma iteração da fase rotulação.

Na Figura 5.7 têm-se um agrupamento hipotético com dados não rotulados (pontos brancos). A partir da definição do centroide μ de cada grupo, os elementos $x_i \in X^L$ (rotulados, representados pelos pontos azuis, vermelhos e verdes) têm seus respectivos grupos definidos pelo DSL. E por fim, têm esses dados rotulados influenciando cada um dos grupos para realizar a rotulação dos pontos brancos. O agrupamento não garante que apenas elementos de uma mesma classe estarão presentes em cada um dos grupos. Desta forma, cada grupo G_j é analisando considerando a classe dos elementos rotulados atribuídos à ele, com isso definindo a rotulação de seus elementos.

Para cada amostra não rotulada $x_i^U \in G_j$, sendo G_j um determinado grupo, calcula-se o vetor de influência $I_{x_i^U}$, formado pelos k elementos rotulados mais influentes no grupo G_j , em outras palavras, os k elementos rotulados mais similares à x_i^U de acordo com a medida de ITL escolhida. Com base nisso, o vetor $I_{x_i^U}$ é definido pela Equação 5.8. Onde $\{x_i\}_{i=1}^k$ são os k elementos mais influentes de x_i^U . A medida de ITL neste caso, define o quão similar os k elementos rotulados mais influentes são das amostras $x_i^U \in G_j$.

$$I_{x_i^U} = \{x_i\}_{i=1}^k \in X^L \quad (5.8)$$

Como a divergência de Kullback-Leibler e a entropia cruzada são calculadas entre duas distribuições de probabilidades, neste etapa, utilizamos a Janela de Parzen (Parzen

1962) para se calcular a função de distribuição de probabilidade, pelo fato de não termos nenhuma informação sobre a estrutura subjacente dos dados.

A Figura 5.8 representa o processo de rotulação realizado nessa etapa. Os dados rotulados $x_i \in L$ (pontos vermelhos e azuis) exercem influência sobre os não rotulados $x_i^U \in U$, pertencentes ao grupo.

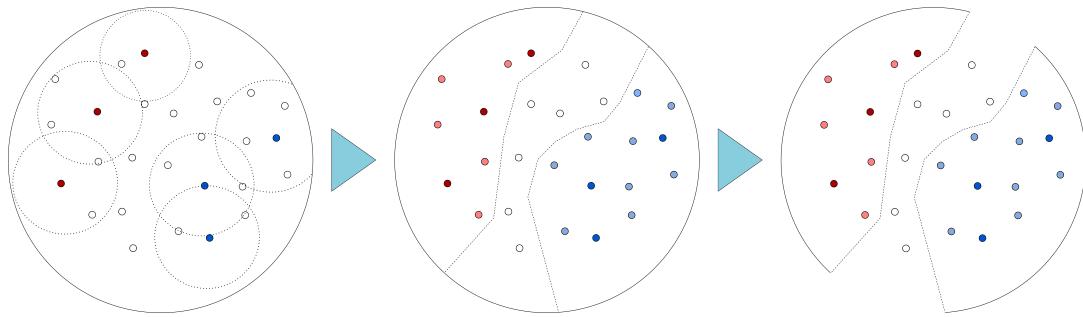


Figura 5.8: Representação dos dados rotulados influenciando para definir a rotulação dos elementos não rotulados do grupo.

Dada uma determinada amostras $x_i^U \in G_j$, com seu respectivo vetor de influência $I_{x_i^U}$, calcula-se o rótulo (classe) para x_i^U . A partir das classes dos elementos pertencentes à $I_{x_i^U}$, define-se o rótulo de x_i^U como a probabilidade de pertencer a cada classe. A Equação 5.9 apresenta o cálculo da probabilidade de x_i^U pertencer à uma classe r ($P(x_i^U|r)$). Sendo f_r a frequência que r aparece nas classes dos elementos de $I_{x_i^U}$.

$$P(x_i^U|r) = \frac{f_r}{k} \quad (5.9)$$

Entretanto, com o objetivo de tornar o cálculo do rótulo mais consistente, acrescentamos na Equação 5.9 a média da medida de ITL (M_r) adotada, como pode ser vista na Equação 5.10. Desta forma, o DSL acrescenta no cálculo da definição dos rótulos a Teoria da Informação, tornando a rotulação mais robusta.

$$P(x_i^U|r) = \frac{f_r}{k.M_r} \quad (5.10)$$

Sendo M_r a média dos valores calculados pela medida de ITL adotada no modelo, entre x_i^U e cada um dos elementos de $I_{x_i^U}$. Então pode-se dizer que:

$$M_r = \frac{\sum_{j=1}^k d(x_i^U, x_j)}{f_r} \quad (5.11)$$

Pois, como a média M_r é calculada para os valores de uma determinada classe r , então divide-se o somatório dos valores da medida de ITL pela quantidade de vezes em que a classe r aparece no vetor $I_{x_i^U}$. Em outras palavras, divide-se o somatório pela frequência da classe r . Substituindo M_r na Equação 5.10, temos:

$$P(x_i^U | r) = \frac{f_r}{k \cdot \left[\frac{\sum_{j=1}^k d(x_i^U, x_j)}{f_r} \right]} \quad (5.12)$$

Desenvolvendo a Equação 5.12, obtemos então a Equação 5.13, que é o cálculo da probabilidade de uma determinada amostra x_i^U pertencer à classe r .

$$P(x_i^U | r) = \frac{f_r}{k} \cdot \frac{f_r}{\sum_{j=1}^k d(x_i^U, x_j)} \quad (5.13)$$

$$P(x_i^U | r) = \frac{f_r^2}{k \cdot \sum_{j=1}^k d(x_i^U, x_j)}$$

Aplicando a Equação 5.13, calcula-se o vetor de probabilidades $P_{x_i} = \{P(x_i^U | c_j)\}_{j=1}^c$, contendo as probabilidades de x_i^U pertencer à cada uma das classes definidas no problema. Em seguida, calcula-se o rótulo do elemento x_i (R_{x_i}), segundo à Equação 5.14, aplicando a Função Degrau (Equação 5.15) à cada um dos elementos de P_{x_i} .

$$R_{x_i^U} = \sum_{c=1}^C f(P(x_i^U | r)) * c \quad (5.14)$$

Onde f é a função Degrau definida na Equação 5.15.

$$f(P_i) = \begin{cases} 0, & x \leq 0.5 \\ 1, & x > 0.5 \end{cases} \quad (5.15)$$

Definidos os novos rótulos dos elementos, o grupo em análise é dividido, colocando elementos juntos à elementos rotulados da sua classe de maior influência, como é mostrado na Figura 5.8. Onde podemos ver, na primeiro grupo, elementos rotulados com suas zonas de influência, no segundo passo, os elementos não rotulados obtendo seus rótulos, e na terceira parte, a divisão do grupo de acordo com as classes pertencentes àquele grupo. Essa fase, é realizada várias vezes até que seja atingido uma condição de parada.

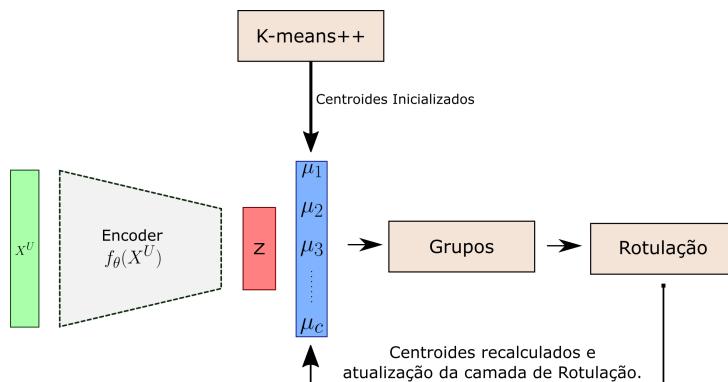


Figura 5.9: Execução do treinamento durante a etapa de rotulação.

Após todos os grupos analisados, sendo seus elementos rotulados ou não, o treinamento segue para próxima iteração. Em resumo, após a inicialização dos centroides utilizando o $k\text{-means}++$, na etapa de rotulação, estes são atualizados de acordo com os novos grupos que serão definidos. Isso significa que, a camada de rotulação vai sofrer alterações durante todo o treinamento, até que cada um dos seus neurônios ative de acordo com a classe. A figura 5.9 exemplifica o processo de treinamento na etapa de rotulação. Têm-se o modelo com seus pesos inicializado pelo $K\text{-means}++$, que por sua vez gera os grupos iniciais, e após a rotulação, esses grupos serão subdivididos em outros ou ter seus elementos rotulados. A cada iteração, quando um grupo têm os rótulos de seus elementos definidos, essas amostras agora rotuladas, são adicionadas ao conjunto X^L . Isso é chamada de fase transductiva do treinamento.

5.2.1 Definição final do rótulo de uma amostra x_i^U

Na fase de rotulação, é quando as amostras pertencentes à X^U obtém um rótulo. Os elementos de um determinado grupo G_j , são rotulados caso apenas uma determinada quantidade de amostras rotuladas sejam atribuídas à um grupo.

No caso do exemplo da Figura 5.8, o grupo não teria seus elementos rotulados de forma definitiva, pois existem mais de uma classe dentre os elementos rotulados atribuídos à esse grupo.

Inicialmente, definimos que quando um grupo recebesse em uma iteração, apenas amostras rotuladas de uma só classe r , os elementos daquele grupo seriam rotulados na classe r . Entretanto, de forma empírica, definimos um novo hiperparâmetro t , chamada de fator de rotulação. Desta forma, define-se o rótulo dos elementos de um grupo:

- Se, em uma iteração, existir apenas elementos de uma classe no grupo;
- Ou, se a porcentagem de elementos de uma determinada classe r atribuída ao grupo, for maior ou igual a um fator de rotulação t , que é descrito em probabilidade (0 à 1).

O fator de rotulação t é um hiperparâmetro, ou seja, deve ser definido previamente. Ele define uma porcentagem mínima de quantidade de elementos de uma determinada classe r para que as amostras do grupo seja rotulada como a classe r . Deve-se escolher esse valor cuidadosamente, pois um t muito grande pode fazer com que o algoritmo não obtenha convergência, e um muito baixo pode definir rotulação de baixa qualidade no grupo.

Após realizar a inicialização dos parâmetros na fase de Agrupamento, o treinamento entra em um *loop* executando a fase de rotulação. Quando um determinado grupo G_j é analisado, duas coisas podem acontecer: 1) Existe uma classe r em maior no grupo que satisfaça o parâmetro t ou existe apenas uma classe, e as amostras do grupos G_j serão rotuladas como r ; 2) Os critérios de rotulação não são satisfeitos e ocorre a divisão do grupo, como mostrado na Figura 5.8, surgindo assim novos grupos, como demonstrado na Figura 5.7.

5.2.2 Condições de Parada

Como dito anteriormente, a etapa de rotulação é realizada várias vezes ajustando os pesos do modelo. Assim, foram definidas algumas condições de parada que podem ser utilizadas para encerrar o treinamento. São elas:

1. Todos os elementos do conjunto X^U forem totalmente rotulados;
2. Não ocorrer mais alterações nos centroides nem no agrupamento dos dados;

5.3 Ajuste Fino

Atingida uma condição de parada para a etapa de rotulação, o treinamento entra então na ultima fase, o Ajuste Fino. Após essa fase ainda poderão existir elementos não rotulados, pois alguns destes elementos não puderam ser rotulados devido não serem atingidos por nenhuma zona de influência de quaisquer elemento rotulado.

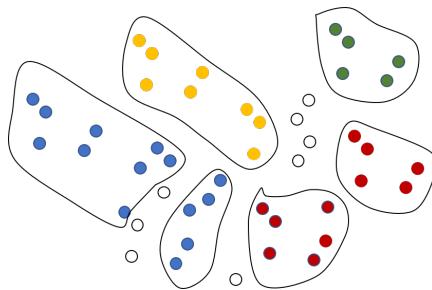


Figura 5.10: Representação do resultado da rotulação após a Fase de Rotulação.

A figura 5.10 apresenta uma representação gráfica de como seria a rotulação após a fase de Rotulação. Considera-se que no problema existem quatro classes: vermelha, azul, amarela e verde e os pontos brancos como sendo elementos não rotulados. Nota-se que em meio aos grupos alguns elementos não foram rotulados devido à condição na Etapa de rotulação, de que é preciso estar na zona de influência de elementos da mesma classe.

Notou-se que os elementos mais próximos da fronteira entre os grupos apresentaram uma maior dificuldade de se rotular, pois estes possuem uma maior probabilidade de estar sob influência de mais de um elemento de classes distintas. Na Fase do Ajuste Fino, treina-se o DSL (*encoder* + camada de rotulação) para classificar e rotular estes elementos remanescentes.

Antes de realizar essa classificação, o modelo é submetido novamente ao algoritmo backpropagation otimizando uma função de custo baseada em teoria da informação. Porém, diferente da etapa de agrupamento, não se utiliza uma distribuição auxiliar, mas os rótulos das amostras rotuladas (X^L), tanto os que já existiam, como os que obtiveram seus rótulos na etapa de rotulação.

5.4 Algoritmo de Treinamento

Dado o modelo proposto, apresenta-se a formalização do algoritmo de treinamento no Algoritmo 4. O algoritmo tem como entrada os conjuntos de dados X^U e X^L , sendo dados não rotulados e rotulados respectivamente, onde y são as classes (rótulos) do conjunto X^L . Além disso, o algoritmo recebe como entrada o valor de k , que é a quantidade de vizinhos rotulados mais influentes que serão considerados para realizar a rotulação no grupo e o valor de t , que define um limite mínimo para a porcentagem que uma classe r deve ter em relação às outras dentro do grupo, para que r seja a classe definida para todos os elementos do grupo. Como saída, os têm-se conjunto X^U totalmente rotulado, durante a fase transductiva do treinamento e o próprio DSL treinamento, de tal forma que seja capaz de predizer amostras que não foram apresentadas durante o treinamento, na chamada fase indutiva.

Algoritmo 4: Algoritmo de Treinamento do DSL.

Result: U totalmente rotulado e DSL com parâmetros ajustados

Entrada: X^L, X^U, y, k, t

Saída : U Totalmente Rotulado e RNA Treinada com X^L e X^U

- 1 Treinar DAE
 - 2 $\mu = \text{calcular_centroides_iniciais}(X^U)$
 - 3 $G = \text{agrupamento}(X^U, \mu)$
 - 4 **while** Não convergiu **do**
 - 5 $G^L = \text{definir_grupos_rotulados}(X^L)$
 - 6 $y_r = \text{calcular_rotulos}(G)$
 - 7 $\text{redefinir_grupos}(X^U, x)$
 - 8 atualizar camada de rotulação
 - 9 ajustar pesos
 - 10 **end**
 - 11 Realizar Ajuste fino
-

5.5 Considerações

O modelo proposto denominado Auto-rotulação Profunda (*Deep Self-Labeled (DSL)*) é uma rede neural formada por um *encoder*, extraído de um *autoencoder* treinado previamente, e uma camada de rotulação adicionada durante o treinamento do DSL para realizar a rotulação e classificação.

Este capítulo apresentou detalhadamente a formação do DSL, bem como o seu algoritmo de treinamento. Para formar o *encoder* do DSL, pode-se utilizar um *Deep Autoencoder* ou um *Autoencoder Convolucional*, no caso de bases de imagens.

Apresentou-se também neste capítulo, os experimentos que definiram os valores ideias para os hiperparâmetros em cada uma das bases adotadas. Desta forma, compara nos próximos capítulos, o DSL com outros modelos semissupervisionados, tanto os tradicionais

quanto modelos de trabalhos mais recentes, no contexto de classificação de imagens, classificação no sensoriamento remoto e dados de *stream*.

Capítulo 6

Experimentos e Resultados

Este capítulo apresenta os resultados experimentais do modelo proposto. Para isso, definiu-se três experimentos básicos: *Avaliação dos Hiperparâmetros*, *Comparação com modelos semissupervisionados clássicos* e *Comparação com modelos da literatura*. Os experimentos apresentados neste capítulo foram realizados utilizando as bases de dados *benchmark* e de imagens, descritas no Capítulo 4.

Neste trabalho, em todos os experimentos, aplicou-se a normalização *MinMax* em todas as bases de dados, para normalizar os dados. A Equação 6.1 define o *MinMax*, onde v é o valor original, v' o valor normalizado, \min_A o menor valor do atributo e \max_A o maior valor do atributo.

$$v' = \frac{v - \min_A}{\max_A - \min_A} \quad (6.1)$$

A normalização *MinMax* transforma todos os atributos para uma escala de 0 a 1, diminuindo assim a probabilidade de interferência no aprendizado dos modelos em decorrência de apenas um atributo com valores acima ou abaixo dos demais.

Cada experimento apresenta os resultados em duas fases: **transdutiva**, que é quando os modelos rotulam o conjunto X^U durante o processo de treinamento; **indutiva**, quando os modelos classificam amostras não previstas no treinamento. Para isso, as bases de dados foram divididas em **conjunto de treino** e **conjunto de teste**, adotando o processo chamado de Validação Cruzada, apresentado no Capítulo 4.

O processo de validação cruzada, da forma como foi utilizada neste trabalho, é descrita como mostra a Figura 6.1, que apresenta a divisão das bases durante os experimentos. Em cada iteração da validação cruzada, tem-se a base dividida em conjunto de treino, que é subdividido em dois: rotulados (L) e não rotulados (U), que serão utilizados na fase transdutiva. O conjunto de teste, é utilizado para testar o modelo após o treinamento, na chamada fase indutiva.

Como as bases de dados apresentadas no Capítulo 4, são todas totalmente rotuladas, removeu-se o rótulo da maioria das amostras, deixando apenas algumas rotuladas. A quantidade de dados em L foi variada na faixa de valores [1%, 2%, 3%, 4%, 6%, 8% e 10%] das amostras, sendo o restante não rotulado. Cada modelo foi submetido nestas condições, avaliando na fase transdutiva a acurácia, precisão, *recall* e *f-score*. A tabela 6.1 mostra a quantidade de amostras em cada uma das porcentagens adotadas para cada base de dados.

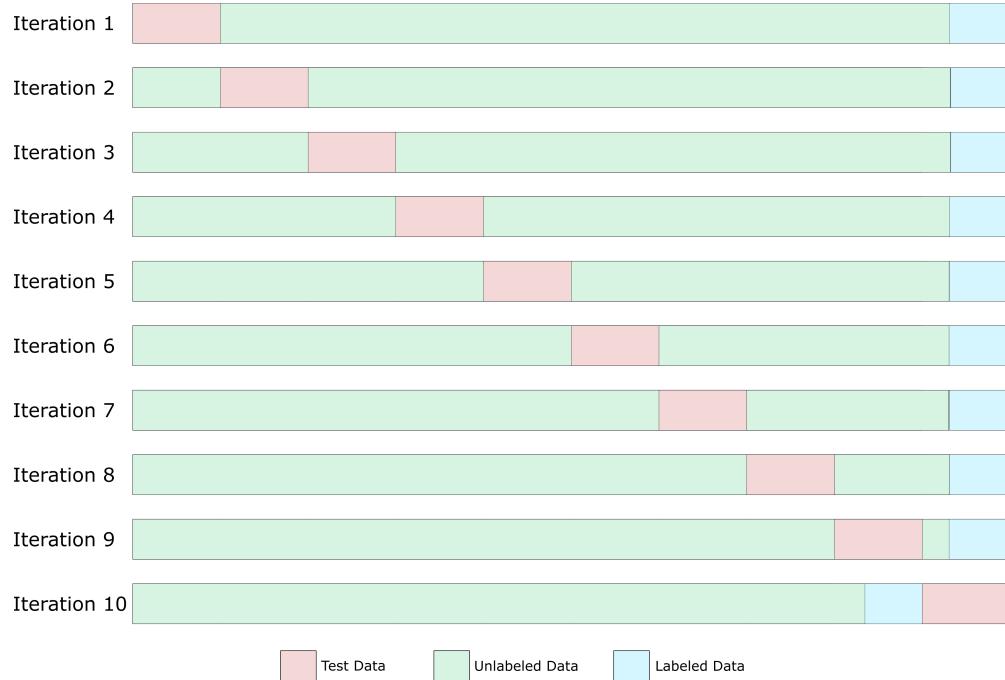


Figura 6.1: Representação visual da divisão das bases de dados nos experimentos para as fases transdutiva e indutiva. Fonte: Lima et al. (2021)

Tabela 6.1: Quantidade de amostras em relação à porcentagens de dados rotulados para cada base

	Total de Amostras	1%	2%	4%	5%	8%	10%
EPILEPSIA	11.500	115	230	460	690	920	1150
REUTERS	11228	112	225	449	674	898	1123
GÁS	13.910	139	278	556	835	1113	1391
PENDIGITS	10992	110	220	440	660	879	1099
MNIST	70.000	700	1400	2800	4200	5600	7000
FASHION	70.000	700	1400	2800	4200	5600	7000
CIFAR-10	70.000	700	1400	2800	4200	5600	7000
SLT-10	60.000	600	1200	2400	3600	4800	6000

Para a implementação do modelo proposto, dos modelos utilizados na comparação e a modelagem dos experimentos foram implementados utilizando a Linguagem *Python*. Utilizou-se as bibliotecas *Scikit-Learn* (Pedregosa et al. 2011), que apresenta uma série de métodos e funções de aprendizagem de máquina e *Keras* (Chollet e others 2015) que é uma biblioteca específica para implementação de modelos *Deep Learning*.

6.1 Hiperparâmetros

Como mostrado anteriormente, o modelo proposto (DSL) possui dois hiperparâmetros, k , que é a quantidade de vizinhos rotulados mais influentes que serão considerados no processo de rotulação e t que é a porcentagem mínima que uma classe r deve ter em quantidade dentro de um determinado grupo para que o grupo seja rotulado como r . Os valores desses hiperparâmetros (k e t) devem ser definidos antes do treinamento e podem variar de acordo com a base de dados adotada. Para definir quais valores serão utilizados para cada base, realizou-se uma série de experimentos empíricos para determinar quais os melhores valores para k e t em cada uma das bases de dados apresentadas no Capítulo 4, tanto para as bases de *benchmark* quanto as bases de imagens.

Para avaliar o impacto de valor de k , realizou-se um experimento, no qual foram testados valores na faixa de 1 à 25. Neste experimento, utilizou-se a validação cruzada para testar o modelo em cada valor de k . Os experimentos foram feitos em duas vertentes, uma com o DSL utilizando a divergência de *Kullback-Leibler* e outra utilizando a Entropia Cruzada, tanto no processo de rotulação quanto na função custo do *autoencoder*.

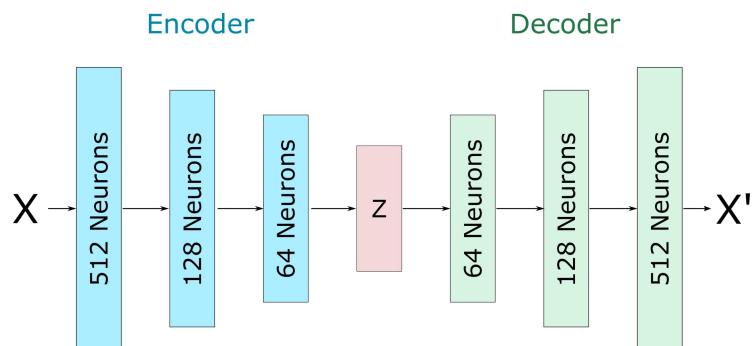


Figura 6.2: Estrutura do DAE utilizado nos experimentos.

Para as bases *benchmark* adotou-se um *deep encoder*, como descrito na Figura 6.2. Um DAE com a primeira camada oculta composta por 512 neurônios, a segunda com 128 e a terceira com 64, sendo a camada central, que define as características no espaço Z , com um total de neurônios igual à quantidade de classes nos dados rotulados. Foi utilizada a Função *Relu*, nas camadas ocultas e função sigmoide na camada de saída. Foi adotada a função Entropia Cruzada como custo para o otimizador Adam. Para definir os hiperparâmetros do DAE, foi adotado o método de busca em grade. Acrescentou entre as camadas ocultas um *dropout* de 0.25 para evitar *overfitting*.

Para as bases de imagens, utilizou-se um convolucional autoencoder descrito na Figura 6.3, com um total de 5 camadas de convolução. Sendo a entrada X , uma imagem, e

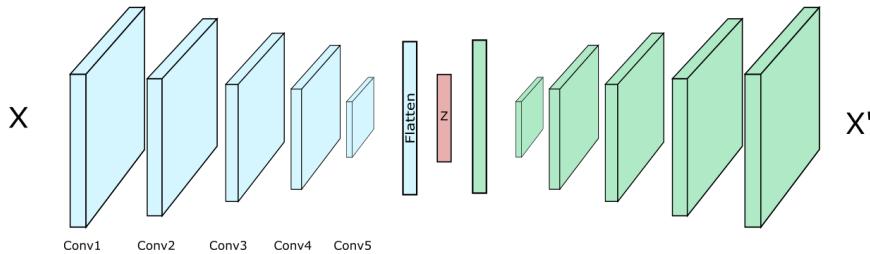
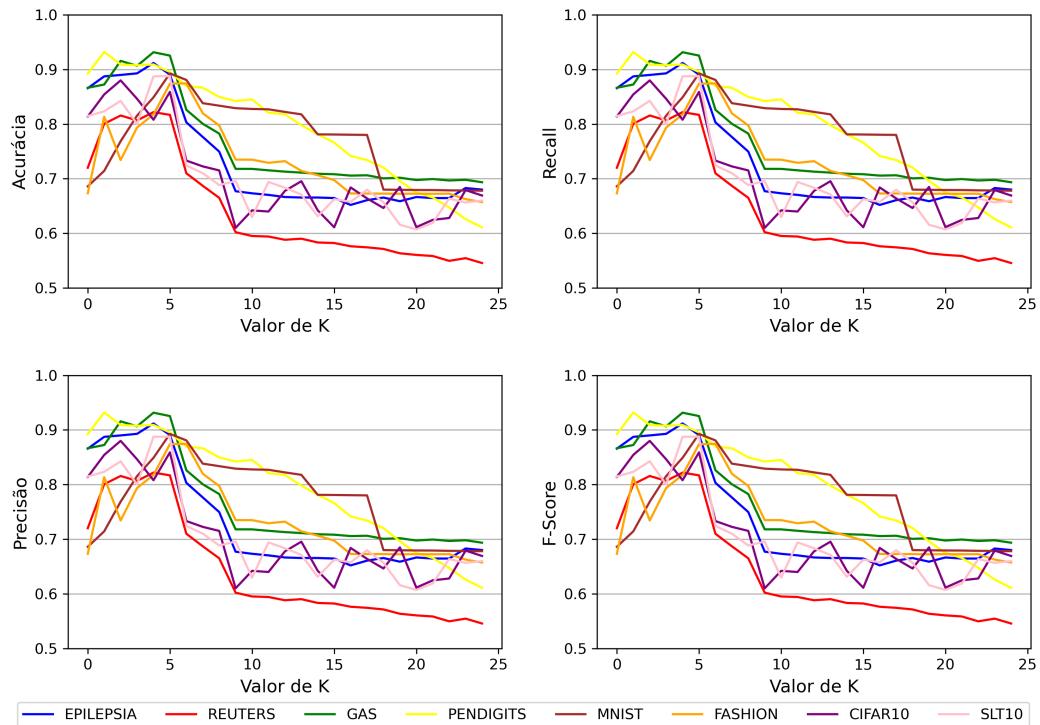


Figura 6.3: Estrutura do CAE utilizado nos experimentos.

Figura 6.4: Resultado para o hiperparâmetro K com Divergência de Kullback-Leibler.

X' , a imagem reconstruída.

As Figuras 6.4 e 6.5, apresenta os resultados dos testes do DSL variando o valor de k , adotando a divergência de Kullback-Leibler como medida de similaridade e na Figura 6.5 a entropia cruzada.

Utilizando um *autoencoder* com a estrutura de camadas $[e \ 512, 256, 128, 64, c]$ neurônios em cada camada respectivamente, no *encoder*, sendo e o tamanho da entrada e c é o número de classes da base de dados adotada. Utilizando um *dropout* de 0.2, realizou-se o experimento.

De acordo com as Figuras 6.4 e 6.5, o modelo apresentou melhores valores de acurácia (acima de 0.9) para k entre 1 e 7. Pode-se observar que a medida que o valor de k aumenta, o modelo perde desempenho, ou seja, a acurácia vai diminuindo. Isso pode ser explicado, devido ao aumento do k , aumenta a quantidade de amostras rotuladas que podem exercer influência sobre as amostras não rotuladas do grupo. As medidas de *recall* e *precisão*

6.1. HIPERPARÂMETROS

81

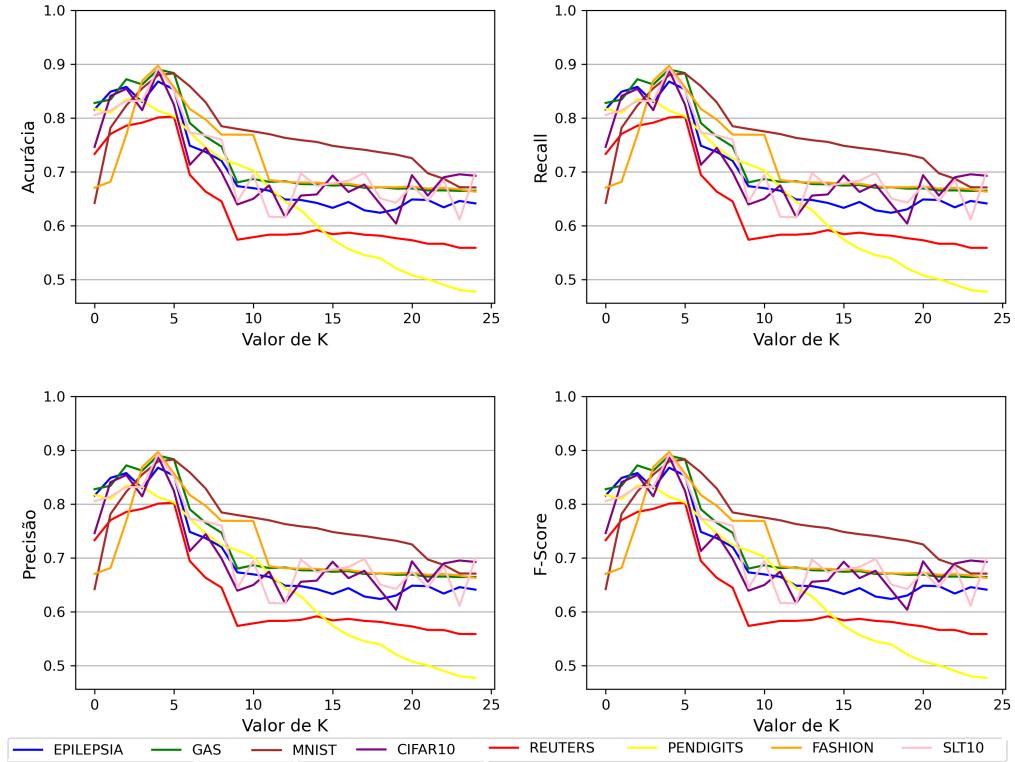


Figura 6.5: Resultado para o hiperparâmetro K com Entropia Cruzada.

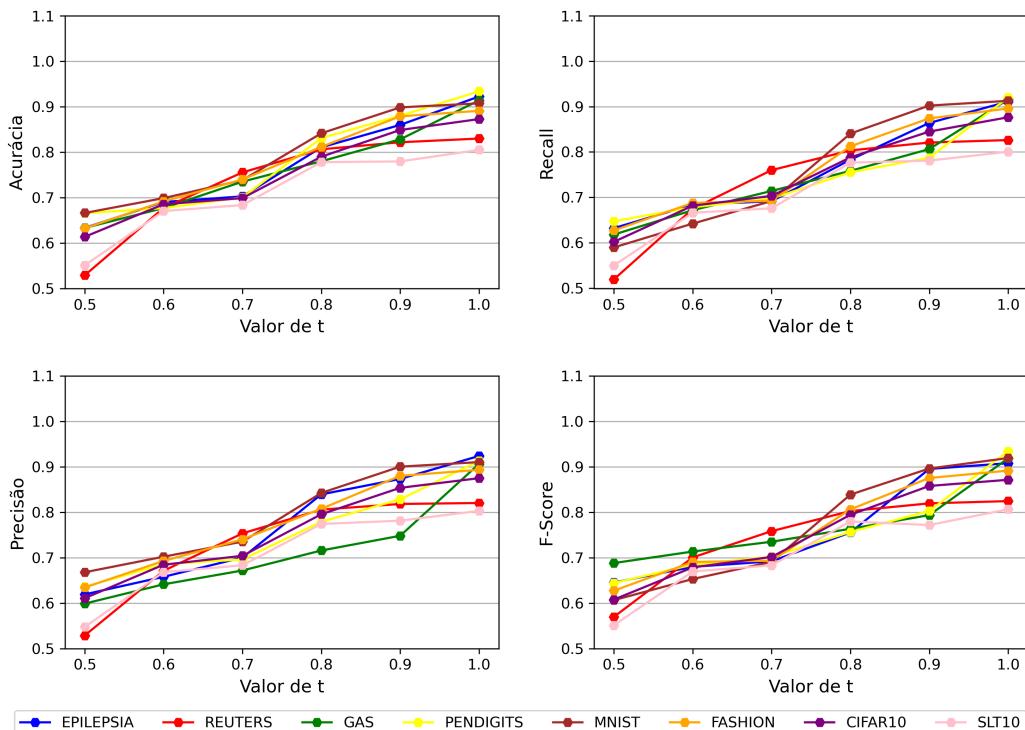
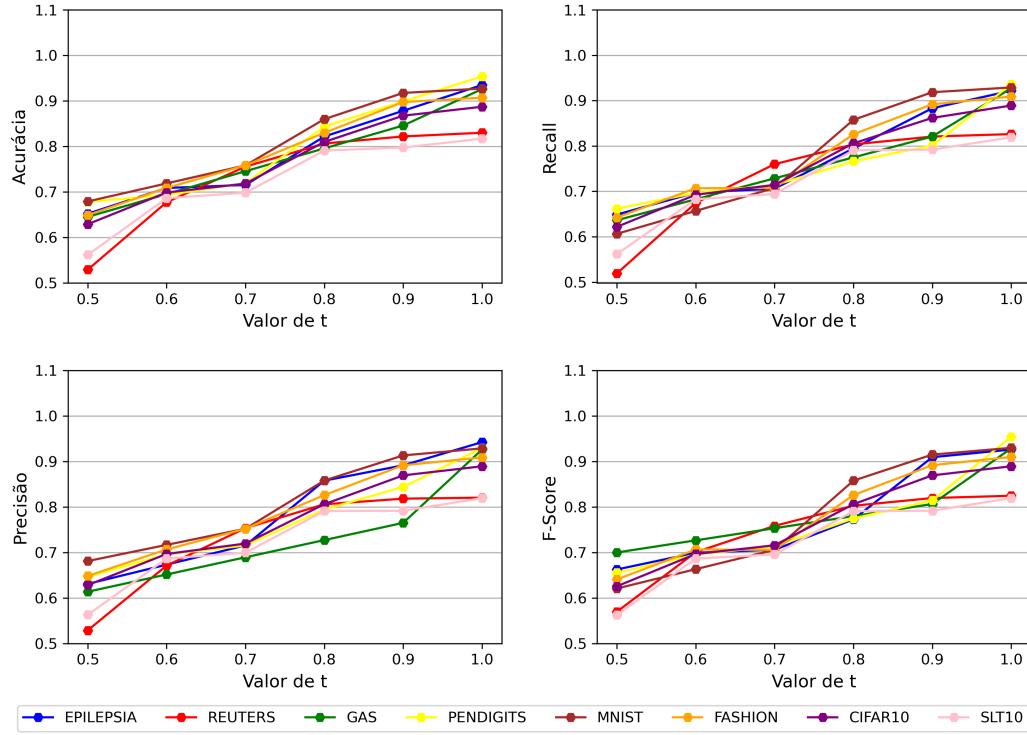


Figura 6.6: Resultado para o hiperparâmetro t com divergência de *Kullback-Leibler*.

Figura 6.7: Resultado para o hiperparâmetro t com entropia cruzada.

confirmam a acurácia do modelo, pois apresentaram valores também acima de 0.9, o que significa que o modelo possui um bom desempenho considerando as classes de maneira individual.

De maneira similar, as Figuras 6.6 e 6.7 mostram resultados do desempenho do modelo variando o valor de t no modelo, utilizando a divergência de *Kullback-Leibler* e Entropia Cruzada respectivamente. Para esse experimento, utilizou-se o mesmo *autoencoder* treinado e testado na avaliação do hiperparâmetro k . Os gráficos das Figuras 6.6 e 6.7 mostram que a acurácia do modelo melhora a medida que vai aumentando o valor de t . Ou seja, caso o valor de t quanto menor o valor de t menos amostras rotuladas de uma determinada classe é são necessárias para definir o rótulo dos elementos de um grupo. Quanto menos amostras eu tenho, menor o grau de confiabilidade do modelo em rotular uma determinada amostra. As medidas de *recall* e precisão, confirmam a acurácia, mostrando a consistência do desempenho do DSL quando analisado classe à classe.

Os resultados deste experimento, apontam que quando considera-se o valor de $t = 1$, o modelo tende a obter o melhor desempenho. Isto é, quando a quantidade de amostras rotuladas que foram atribuídas a um determinado grupo, seja 100% no grupo, significa que o grupo possui uma melhor homogeneidade.

Vale ressaltar, que foram realizados inúmeros testes, onde para cada valor de k , foram testados com cada um dos valores de t , mas para facilitar a visualização separou-se as medidas.

6.2 Comparativo com modelos clássicos

Como dito anteriormente, para validar o DSL, realizou-se experimentos comparando-o com outros modelos de aprendizagem semissupervisionada, neste caso o *self-training*, *co-training*, *tri-training* e *STRED*, e *SEEDED K-means*. Esses modelos foram escolhidos pois são considerados clássicos no contexto de aprendizado semissupervisionado e adotados em diversos trabalhos para efeito de comparação.

Tabela 6.2: Hiperparâmetros usados nos classificadores.

Classificador	Parâmetros
MultiLayer Perceptron (MLP)	1 camada oculta, 100 neurônios; max 500 iterações; Função de Ativação Relu; Função de Ativação softmax na saída.
Máquina de Vetor de Suporte (SVM)	C = 20; Tipo de Kernel = RBF
Random Forest (RF)	200 Árvores na Floresta

Com exceção do *SEEDED K-means*, todos os modelos testados no experimento, utilizam em seus procedimentos um classificador base. Com isso, utilizou-se os classificadores, *Multilayer Perceptron* (MLP), *Random Forest* (RF) e Máquina de Vetor de Suporte (SVM) no experimentos. Para garantir o melhor desempenho, utilizou-se a técnica de Busca em grade para definir os hiperparametros dos classificadores bases, o resultado final pode ser visto na Tabela 6.2. Então, os modelos foram executados variando os classificadores base, mas foram apresentados neste capítulo, apenas combinação modelo e classificador base de melhor desempenho.

Assim como o DSL, os outros modelos de ASS também possuem hiperparâmetros. Para obter os melhores valores para cada classificador, aplicou-se o método de busca em grade (Bergstra e Bengio 2012), explicado no Capítulo 4. Desta forma pôde-se obter os valores ótimos dos hiperparâmetros, apresentados na Tabela 6.2.

6.2.1 Resultados da fase transdutiva com modelos clássicos

Como dito anteriormente, o modelo proposto (DSL) foi comparado com os modelos semissupervisionados clássicos. As Figuras 6.8, 6.9, 6.10, 6.11 mostram os gráficos que representam o desempenho do DSL, bem como os modelos clássicos *co-training*, *tri-training*, *STRED* e *SEEDED K-means*, na fase transdutiva. Ou seja, quando um modelo semissupervisionado classifica (rotula) as amostras do conjunto U (não rotulado). Os modelos apresentados neste trabalho e o modelo proposto (DSL) possuem esta característica.

A Figura 6.8 apresenta os resultados da rotulação da base Epilepsia. Cada gráfico da Figura 6.8 mostra o desempenho dos modelos quando tem-se amostras rotuladas (%) em relação à cada uma das medidas de avaliação, acurácia, precisão, *recall* e *fscore*.

Como observar na Figura 6.8, o modelo obteve desempenho superior a todos os modelos semissupervisionado clássicos utilizados no experimento. O DSL apresentou acurácia

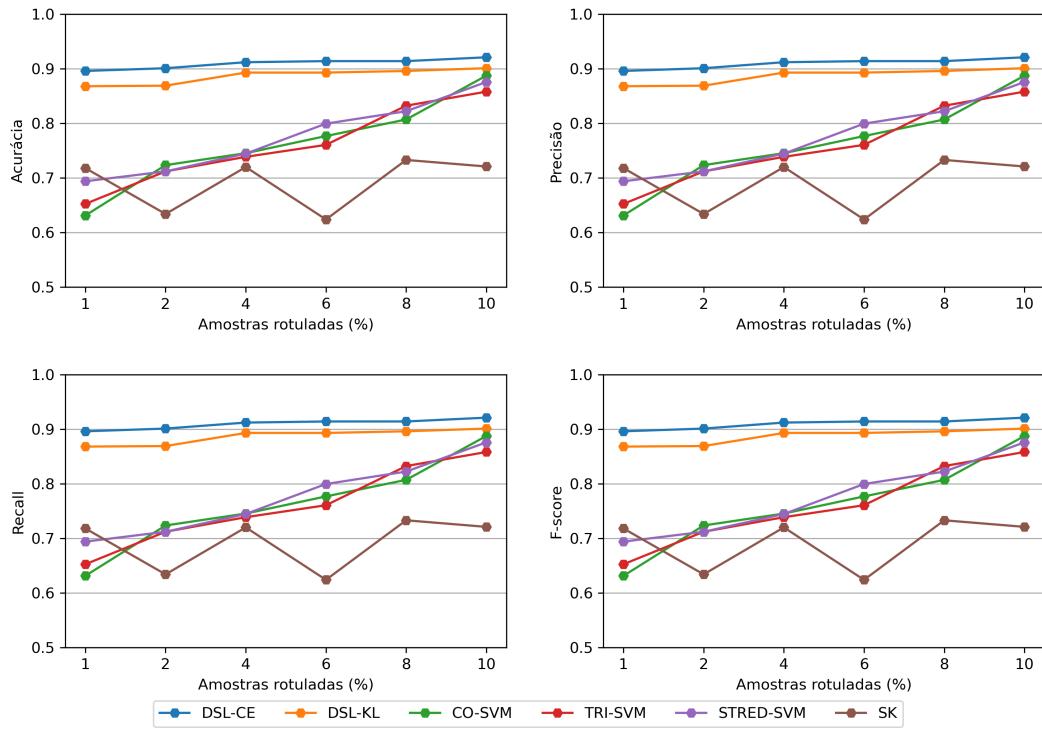


Figura 6.8: Classificação da Base Epilepsia na fase transdutiva.

superior à 0.9 quando utilizou-se a Entropia Cruzada como medida de similaridade no processo de rotulação, em cada uma das quantidades de amostras rotuladas utilizadas.

Pode-se notar, também, que as medidas de precisão e *recall* são similares à acurácia. Isso significa que, os valores de acurácia também são consistentes quando consideramos cada uma das classes em separados. Ou seja, o DSL rotulou corretamente a base Epilepsia, mesmo considerando cada das classes individualmente. Com isso, observou-se também a taxa *fscore*, acima de 0.9, para todas as porcentagens, indicando um equilíbrio entre precisão e *recall*.

Para mostrar a superioridade do modelo proposto, realizou-se testes estatísticos para comparar com os outros modelos. Com um nível de significância de 5% (0.05). O teste estatístico, apresentou valor de p acima de 0.05, para todas métricas avaliadas, mostrando que existe diferença significativa para as porcentagens de 1% à 8% das amostras rotuladas. Com 10% das amostras os testes estatísticos apontaram que apenas o modelo SEEDED K-means apresentou resultados inferiores, mostrando um $p = 0.022$, em todas as métricas.

Na Figura 6.9, pode-se observar os resultados dos modelos na rotulação da base REUTERS. Nesta caso, observou-se que o modelo proposto apresentou resultados inferiores em relação à base de EPILEPSIA. Apenas com 10% das amostras rotuladas, utilizando a entropia cruzada, o DSL obteve acurácia média acima de 0.8. Os outros modelos também, apresentaram dificuldades em trabalhar com a base REUTERS. Os resultados são os mesmos tanto em termos de acurácia, quanto precisão, *recall* e *fscore*.

Apesar disso, o modelo proposto demonstrou melhor consistência nos resultados, pois mesmo com menos amostras rotuladas obteve acurácia acima de 0.85 para todas as por-

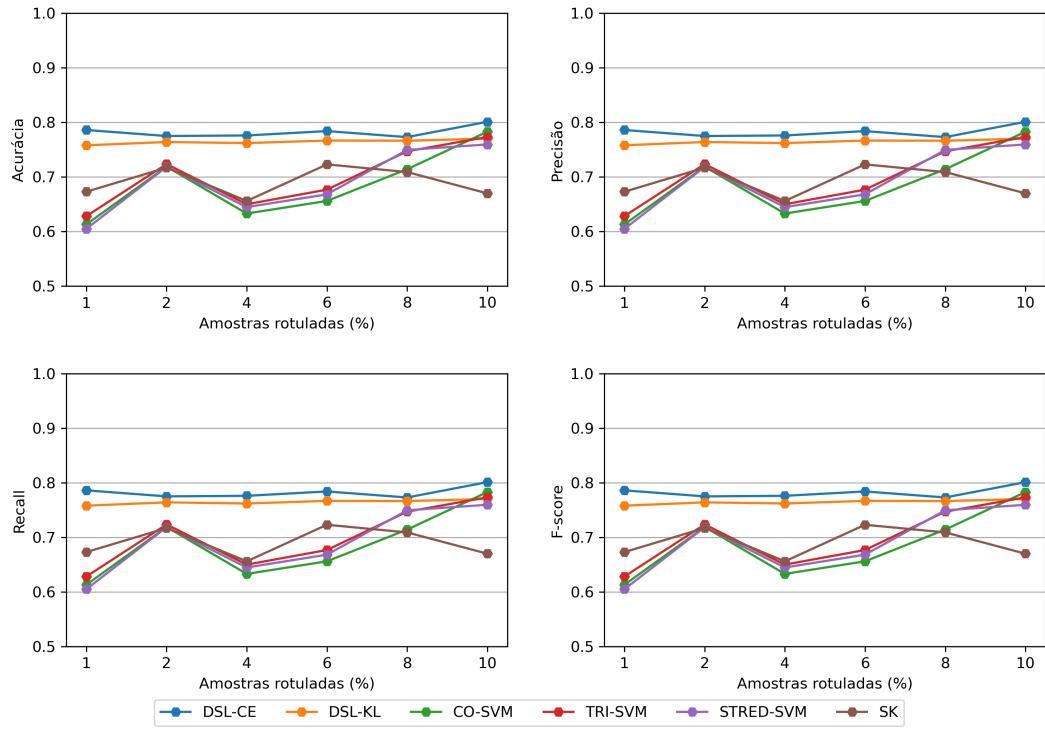


Figura 6.9: Classificação da Base Reuters na fase transdutiva.

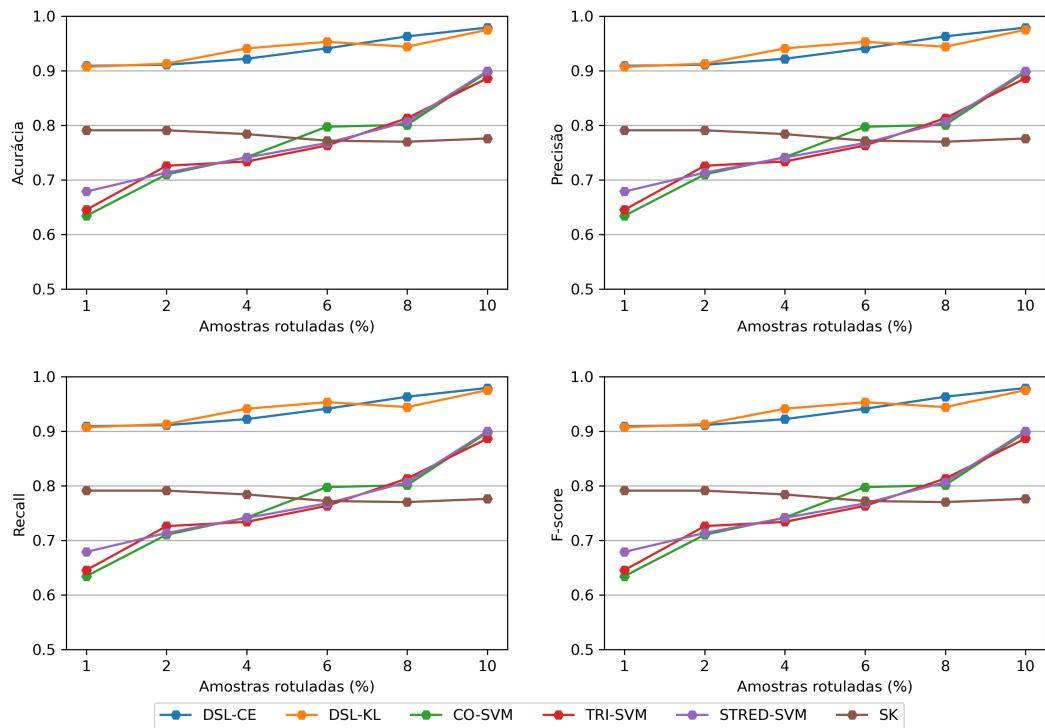


Figura 6.10: Classificação da Base Gás na fase transdutiva.

centagens de L , os outros modelos, com exceção do *SEEDED K-means*, melhoraram o desempenho com o aumento da quantidade de amostras rotuladas. Nota-se também, que o DSL apresentou melhores resultados com a Entropia Cruzada.

Os testes estatísticos, apresentou diferença significativa, pois o modelo proposto apresentou $p > 0.05$ em todas as métricas e porcentagens de dados rotulados. Assim como na base anterior, os modelos apresentaram pouca diferença significativa, com exceção do modelo SEEDED K-Kmeans. Ou seja, mesmo com taxas abaixo de 0.8, o modelo proposto apresentou melhores resultados. Isso pode ser observado tanto na acurácia, quanto precisão, *recall* e *f-score*. Em todas as métricas de avaliação o modelo obteve resultados satisfatórios.

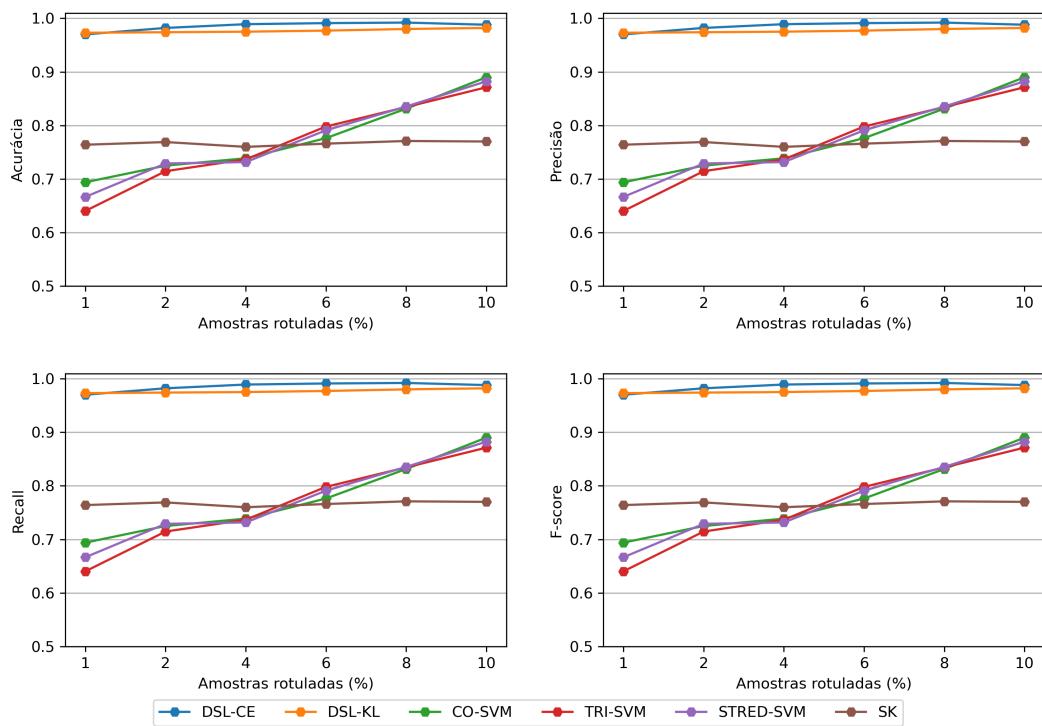


Figura 6.11: Classificação da Base Pendigits na fase transdutiva.

Vale ressaltar, que o modelo proposto demonstrou melhor consistência, pois mesmo com menos amostras rotuladas obteve acurácia acima de 0.85 para todas as porcentagens de L . Nota-se também, que o DSL apresentou melhores resultados com a Entropia Cruzada em relação ao DSL com divergência de *Kullback-leibler*.

Na Figura 6.10, são mostrados os resultados da rotulação da base de Gás na fase transdutiva. Do mesmo modo à base Epilepsia, o modelo proposto mostrou desempenho superior, com acurácia acima de 0.9, tanto para o DSL utilizando divergência de Kullback-Leiber, quanto Entropia Cruzada. No caso da base de gás, o modelo proposto apresentou desempenho similar na rotulação, como pode ser visto nos gráficos da Figura 6.10. O mesmo pode ser observado nas medidas de precisão, *recall* e *f-score*. Os testes estatísticas apontaram que na Base de Gás o modelo proposto apresentou p acima de 0.05, ou seja, existe diferença significativa do DSL para os demais testados.

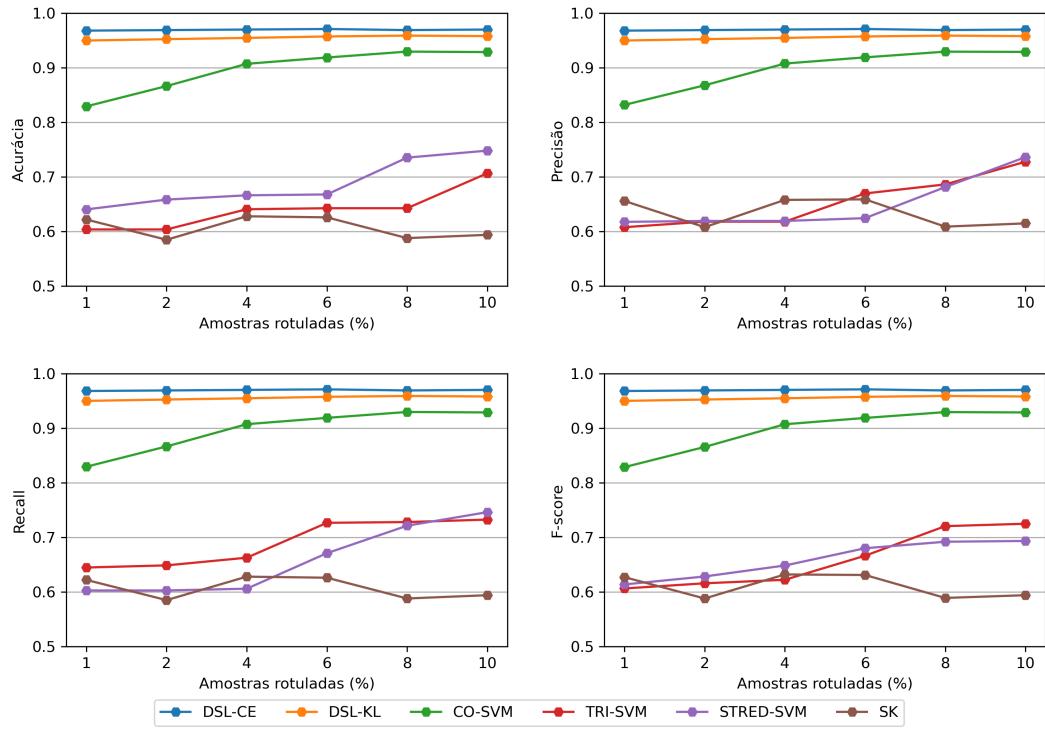


Figura 6.12: Classificação da Base MNIST na fase transdutiva.

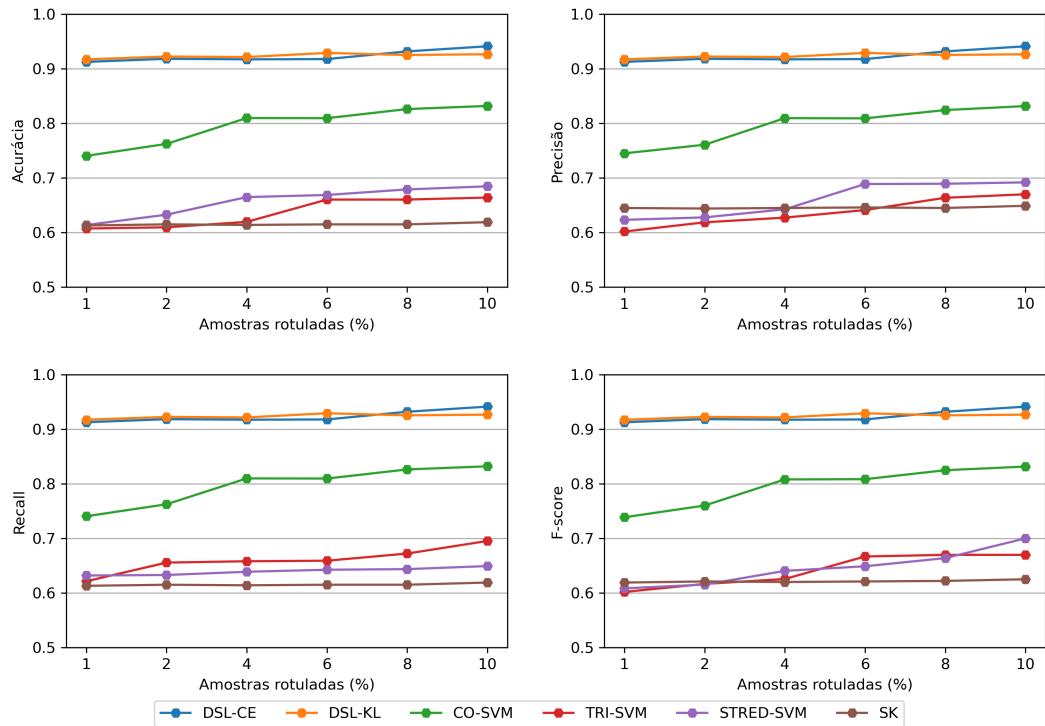


Figura 6.13: Classificação da Base FASHION na fase transdutiva.

E por fim, a Figura 6.11, apresenta os resultados da rotulação da base Pendigits. Igualmente nas outras bases, o DSL apresentou desempenho superior aos modelos semissupervisionados clássicos, sendo melhores taxas de acurácia utilizando a Entropia Cruzada. Em todas as outras métricas, obteve também taxas de precisão, *recall* e *f-score*, acima de todos os outros. Assim como na base PENDIGITS, o DSL apresentou diferença significativa em todas as métricas e porcentagens de dados rotulados, pois obteve $p > 0.05$ em todas as situações.

Como dito anteriormente, além das bases de dados chamadas de *benchmark* (epilepsia, reuters, gás e pendigits), também foram utilizadas bases de imagens, neste caso as bases mnist, fashion mnist, cifar-10 e slt-10. Na Figura 6.12 visualizar os resultados dos modelos na rotulação da base de dados mnist.

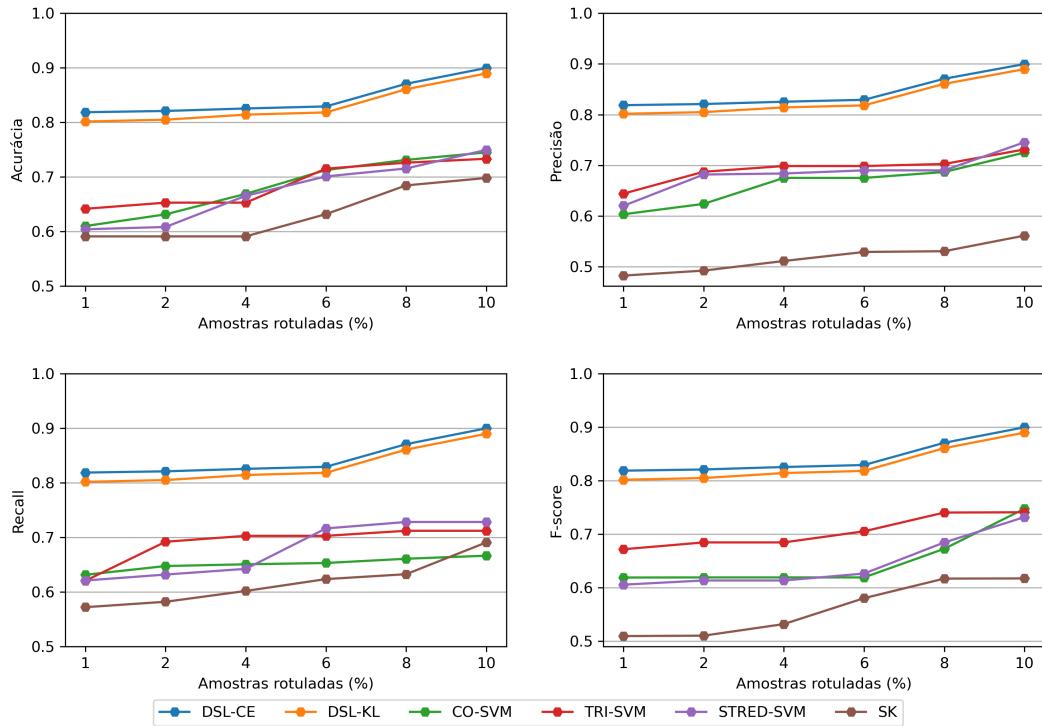


Figura 6.14: Classificação da Base CIFAR10 na fase transdutiva.

Como dito anteriormente, no caso das bases de imagem o DSL é executado utilizando um *autoencoder* Convolucional, pois na literatura inúmeros trabalhos mostram o bom desempenho da CNN em problemas com imagens. Na Figura 6.12, o DSL obteve acurácia média acima de 0.9, tanto utilizando a divergência de *Kullback-Leibler*, quanto a entropia cruzada. A taxa de precisão e *recall* também acompanharam a acurácia e ficaram acima de 0.9, o que significa que o modelo proposto rotula corretamente tanto considerando o geral, quanto classe à classe da base de dados.

De forma similar, nos gráficos das Figura 6.13, é possível observar que o desempenho do DSL também foi satisfatório em relação os outros modelos clássicos. Sendo, MNIST e FASHION MNIST duas bases de dados formadas por imagens em escala de cinza, obtiveram resultados similares na rotulação. Como mostrado na Figura 6.13, na rotulação

da base FASHION MNIST o DSL também apresentou acurácia acima de 0.9, além das outras métricas, precisão, *recall* e *f-score*.

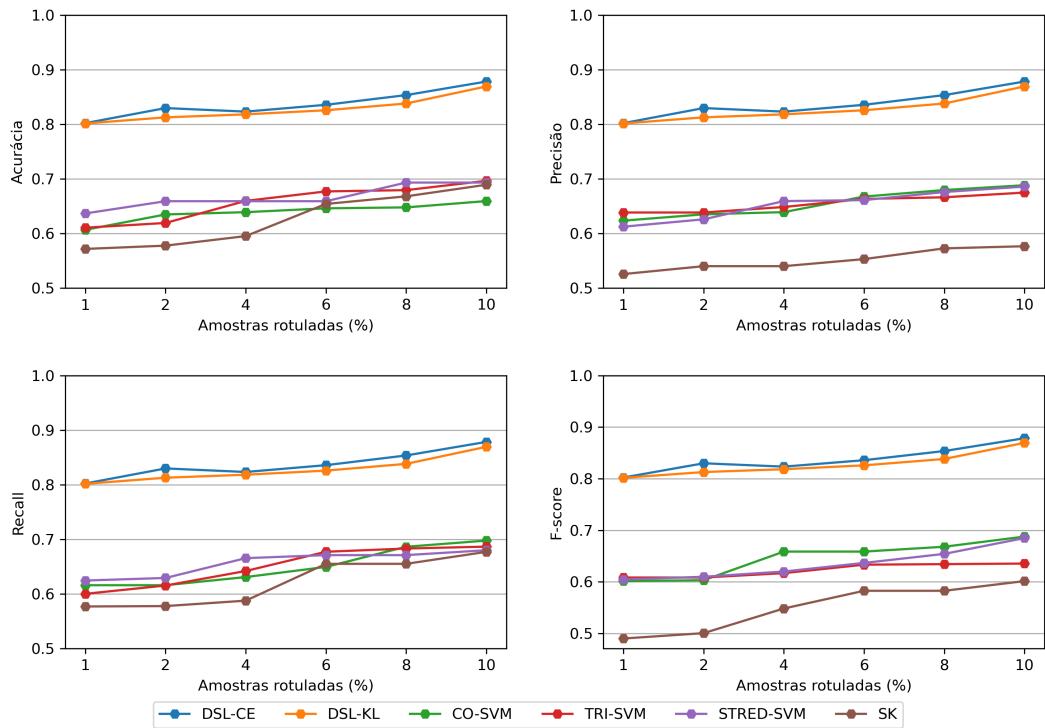


Figura 6.15: Classificação da Base SLT10 na fase transdutiva.

Nas Figuras 6.14 e 6.15 são apresentados os resultados da rotulação das bases CIFAR-10 e SLT-10. Como dito anteriormente, são bases formadas por imagens coloridas, aumentando assim a dificuldade para os modelos conseguirem generalizar o problema. Nesta situação, o modelo proposto apresentou taxa de acurácia acima de 0.9, sendo superior a todos os outros modelos clássicos. Considerando individualmente as classes, através das taxas de precisão e *recall*, pode-se observar que o desempenho do DSL se manteve consistente. A taxa de *fscore*, também acima de 0.85, aponta uma equilíbrio na rotulação das classes.

Para as bases de imagens (MNIST, FASHION, CIFAR-10 e SLT-10) o modelo proposto apresentou diferença significativa para os demais com um valor de $p > 0.05$, isso sendo possível de visualizar nos gráficos das Figuras 6.12, 6.13, 6.14 e 6.15. Na base MNIST, o modelo Co-Training usando máquina de vetor de suporte (CO-SVM) apresentou resultados mais próximos do modelo proposto, porém, o teste estatístico p obteve valor acima de 0.05, mostrando a diferença significativa, mostrando a superioridade do DSL com as bases de imagens na fase transdutiva.

6.2.2 Resultados da Fase Indutiva

Os resultados deste capítulo apresentados até aqui, são referentes à fase transdutiva, onde o modelo rotula dados durante o treinamento. Agora, tem-se os resultados dos mo-

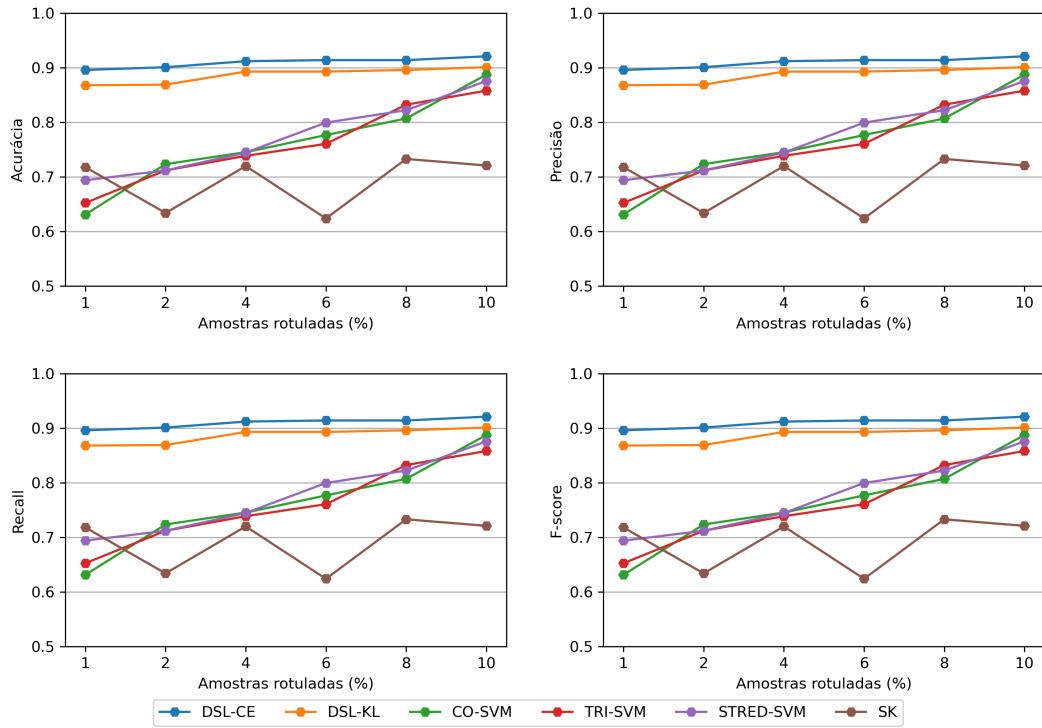


Figura 6.16: Classificação da Base Epilepsia na fase induativa.

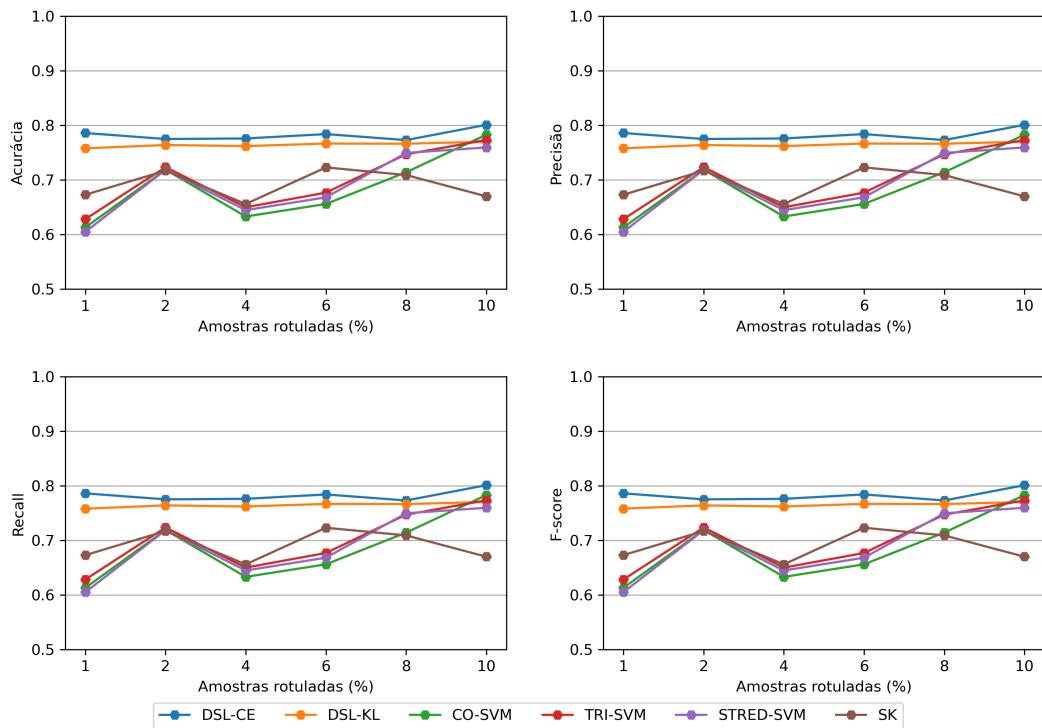


Figura 6.17: Classificação da Base Reuters na fase induativa.

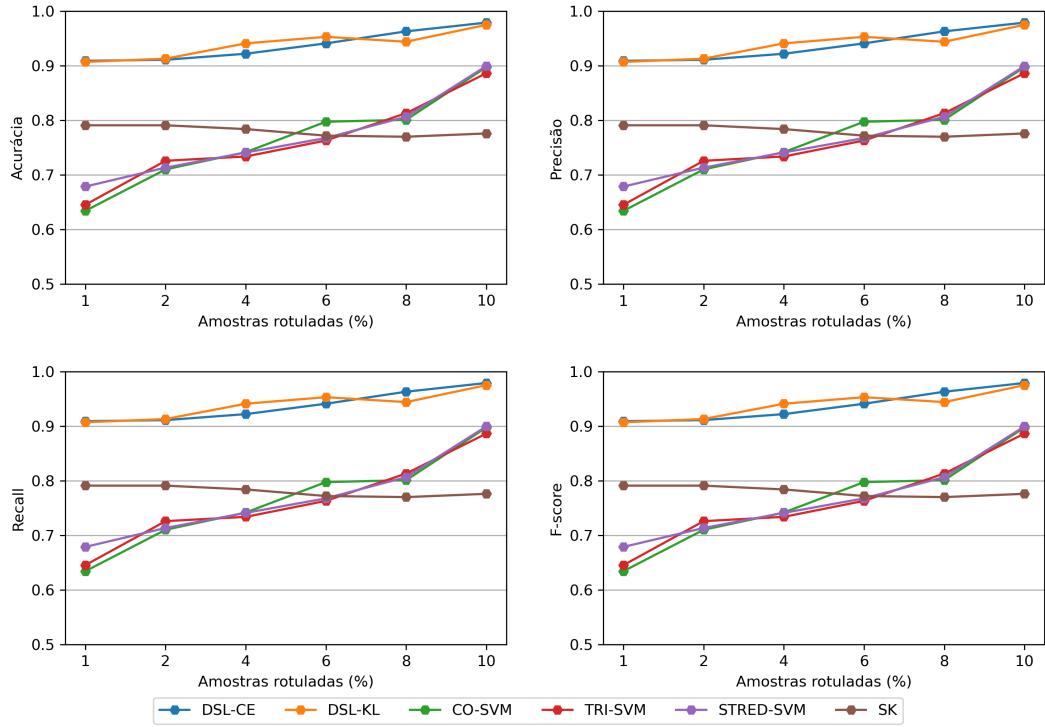


Figura 6.18: Classificação da Base Gás na fase indutiva.

delos na fase indutiva, que é quando classifica-se amostras não previstas no treinamento.

A Figura 6.16 apresenta os resultados da classificação da base EPILEPSIA na fase indutiva. O modelo DSL apresentou desempenho com acurácia aproximadamente de 0.9 quando utilizando a entropia cruzada. Em comparação com os outros modelos, o DSL apresentou desempenho superior, tanto com entropia cruzada quanto com divergência de *Kullback-Leibler*. De forma similar, apresentam-se as taxas de precisão, *recall* e *f-score*.

Na Figura 6.17 tem-se os resultados da classificação da base REUTERS. De forma similar aos resultados da fase transdutiva nesta base, o DSL apresentou desempenho superior em todas as porcentagens de dados rotulados. Mas, com 10% dos dados rotulados, o modelo apresentou resultados superior, mas muitos próximos dos demais modelos, com exceção do *SEEDED-Kmeans*. As taxas de precisão, *recall* e *f-score* seguem similarmente à taxa de acurácia.

Na classificação da Base de GÁS, apresentado na Figura 6.18, o modelo obteve resultado superior em todos as métricas. Sendo todas superior à 0.9. DO mesmo modo acontece nos resultados da base PENDIGITS, mostrados na Figura 6.19. Sendo, tanto com divergência de *Kullback-Leibler* quanto Entropia Cruzada, o DSL apresentou acurácia acima de 0.9, assim como as métricas *precisao*, *recall* e *f-score*. E da mesma forma, nas bases de Imagem apresentadas nas Figuras 6.20, 6.21, 6.22, 6.23.

Na classificação das base MNIST (Figura 6.20), FASHION MNIST (Figura 6.21) e CIFAR-10 (Figura 6.22) o modelo proposto apresentou melhores desempenhos utilizando a Entropia Cruzada. Na base SLT-10 6.23, notamos que os resultados com as duas medidas foram similares. Da mesma forma, segue as taxas de precisão, *recall* e *f-score*.

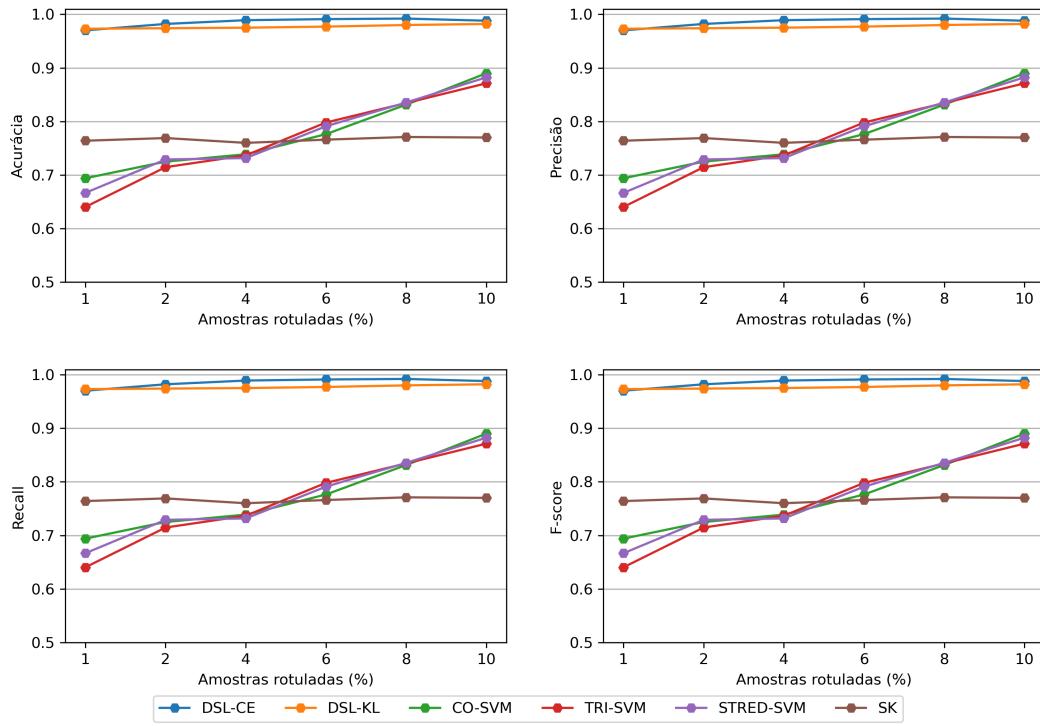


Figura 6.19: Classificação da Base Pendigits na fase indutiva.

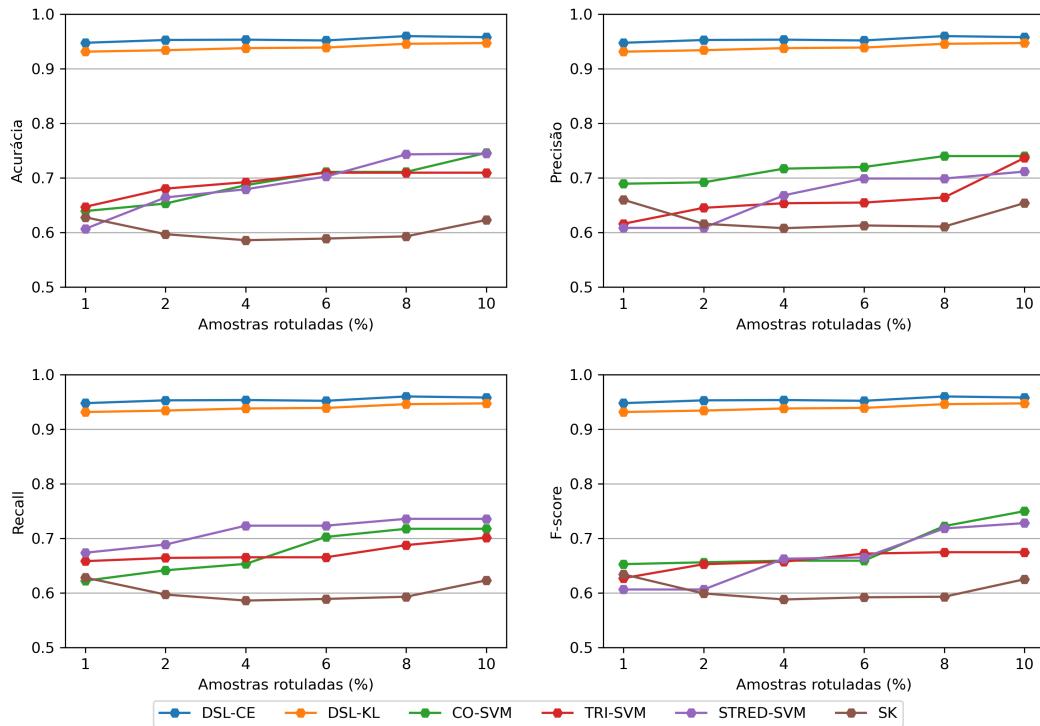


Figura 6.20: Classificação da Base MNIST na fase indutiva.

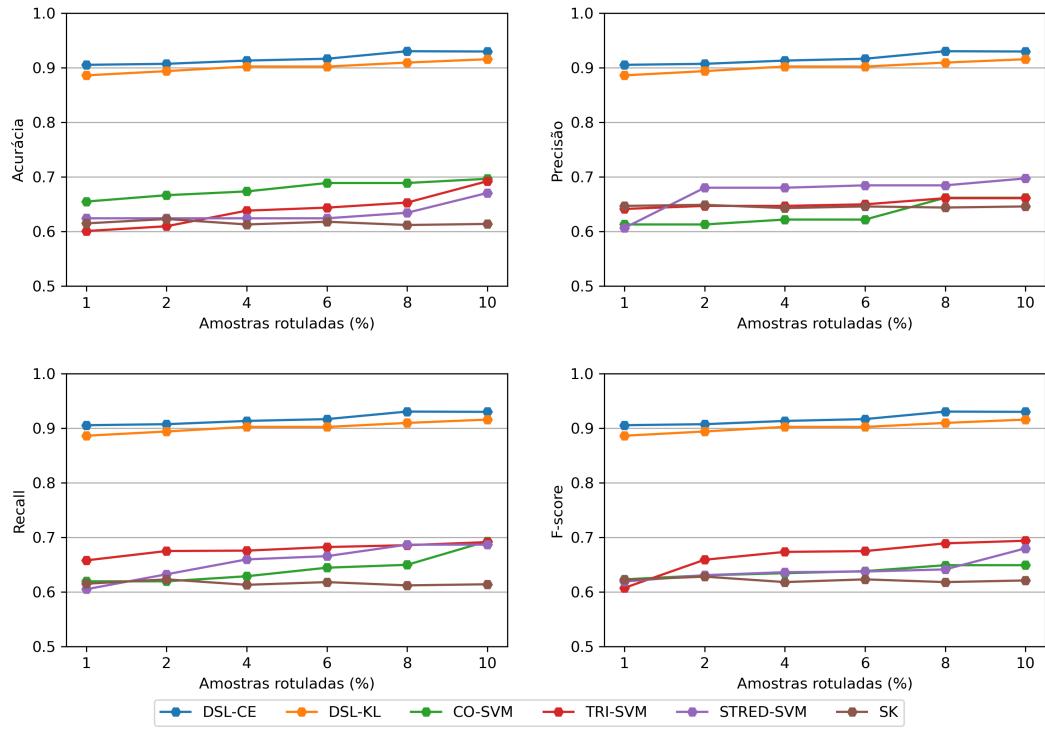


Figura 6.21: Classificação da Base FASHION na fase induativa.

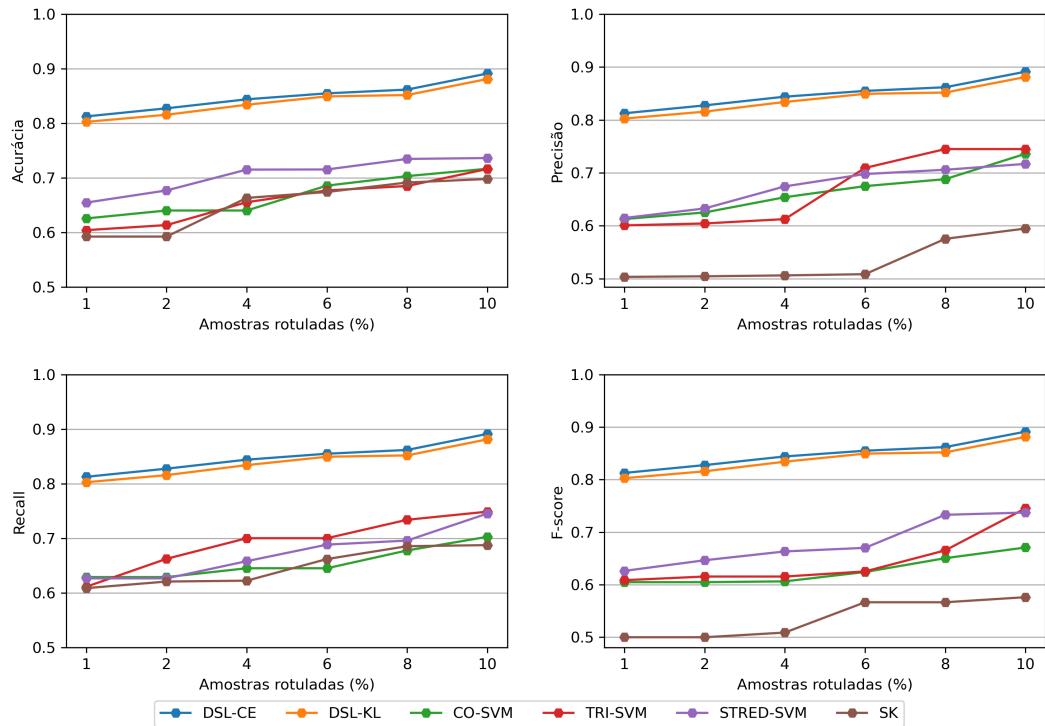


Figura 6.22: Classificação da Base CIFAR10 na fase induativa.

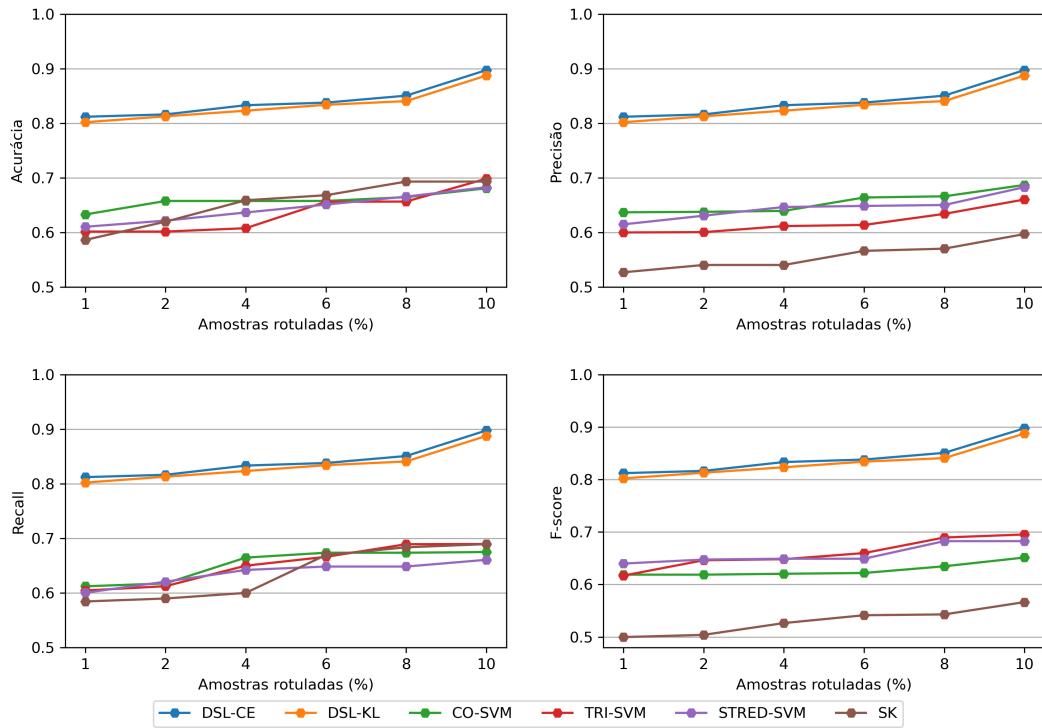


Figura 6.23: Classificação da Base SLT10 na fase indutiva.

De forma geral, tanto a divergência de *Kullback-Leibler* como a Entropia Cruzada foram capazes de ajudar o modelo a definir corretamente a similaridade entre amostras, permitindo definir os vizinhos rotulados mais influentes de forma satisfatória. Desta forma, separando o mais ideal possível grupos, rotulando de forma eficiente e obtendo resultados satisfatórios como mostrado nessa Seção.

O modelo proposto (DSL) apresentou resultados superiores em todas as bases de dados de *benchmark*. Eficientemente, treinou e rotulou todas as bases, bem como classificou de forma satisfatória, amostras após o treinamento. Dentre os modelos testados o *SEEDE K-means* apresentou os piores resultados, principalmente nas bases EPILEPSIA, REUTERS, MNIST, CIFAR-10 E SLT-10.

Os modelos clássicos no geral, apresentaram piores desempenho com as bases de imagem, diferente do modelo proposto que manteve as taxas de acurácia consistente. Os resultados das classificações foram similares aos da fase transdutiva. Isso significa que o modelo DSL é eficiente em rotular o conjunto U e classificar novas amostras.

6.2.3 Análise Estatística

Estatisticamente, o modelo proposto apresentou resultados superiores na classificação da base EPILEPSIA com diferença significativa ($p > 0.05$) em todas as métricas e com 1% à 8% de dados rotulados. Quando têm-se 10% dos dados rotulados o modelo proposto supera apenas o *SEEDED K-means*.

A classificação da base de REUTERS, da qual o modelo proposto apresentou resulta-

dos inferiores em relação às demais bases, o modelo proposto não apresentou diferença significativa em relação às porcentagens 8% e 10% de dados rotulados em todas as métricas. Nestes casos o valor de p ficou abaixo de 0.05.

Para as bases de GÁS e PENDIGITS o DSL apresentou resultados significativamente superiores, pois apresentou $p > 0.05$ para todas as métricas avaliadas em todos os cenários de porcentagens de dados rotulados. Da mesma forma, ocorreu com as bases de imagens MNIST, FASHION, CIFAR-10 e SLT-10.

6.3 Comparativo com modelos da literatura

Os resultados apresentados na Seção anterior, mostraram um comparativo do DSL com modelos clássicos da literatura no aprendizado semissupervisionado. Nesta sessão são mostrados os resultados do modelo proposto em comparação com outros modelos encontrados na literatura. Para esse estudo, selecionou-se os trabalhos que foram descritos no Capítulo 2, são eles:

- **M1** - Algoritmo semissupervisionado baseado em um esquema de votação de probabilidade máxima, entre os modelos de ASS clássico: self-training, co-training e tri-training (Livieris 2019).
- **M2** - Modelo de aprendizado profundo com grafo para classificação semissupervisionada (Lin et al. 2020).
- **M3** - Modelo semi-supervisionado eficaz com base em núcleos locais, com o objetivo de resolver o problema da falta de dados rotulados iniciais (Li et al. 2020).
- **M4** - Redes convolucionais de aprendizagem de grafo dinâmico para classificação semissupervisionada (Fu et al. 2021).

Assim, como no modelo DSL, aplicou-se a validação cruzada nos modelos da literatura escolhidos, como ilustrado na Figura 6.1, avaliando cada modelo na fase transdutiva e na fase indutiva.

6.3.1 Resultados da fase transdutiva com modelos da literatura

A Figura 6.24 mostra os resultados da rotulação da base EPILEPSIA na fase transdutiva. O DSL apresentou desempenho superior em relação aos modelos da literatura, quando utilizou-se a Entropia Cruzada e resultados similares aos dos modelos da literatura com a divergência de Kullback-Leibler. Todos os modelos apresentaram resultados satisfatórios, com taxas de acurácia acima de 0.8, para todas as porcentagens de dados rotulados. Da mesma forma, as medidas de precisão e *recall* apresentaram taxas que confirmam a acurácia em relação às classes individualmente, ou seja, se considerar apenas uma classe o desempenho dos modelos permanece o mesmo. Consequentemente, tem-se uma taxa de *f-score* condizente com precisão e *recall*.

No caso da rotulação da base REUTERS, os da literatura apresentaram melhor desempenho em relação ao DSL, como pode ser visto na Figura 6.25. O modelo DSL apresentou taxa de acurácia abaixo de 0.8, exceto, quando utilizou-se de 10% de amostras rotuladas.

CAPÍTULO 6. EXPERIMENTOS E RESULTADOS

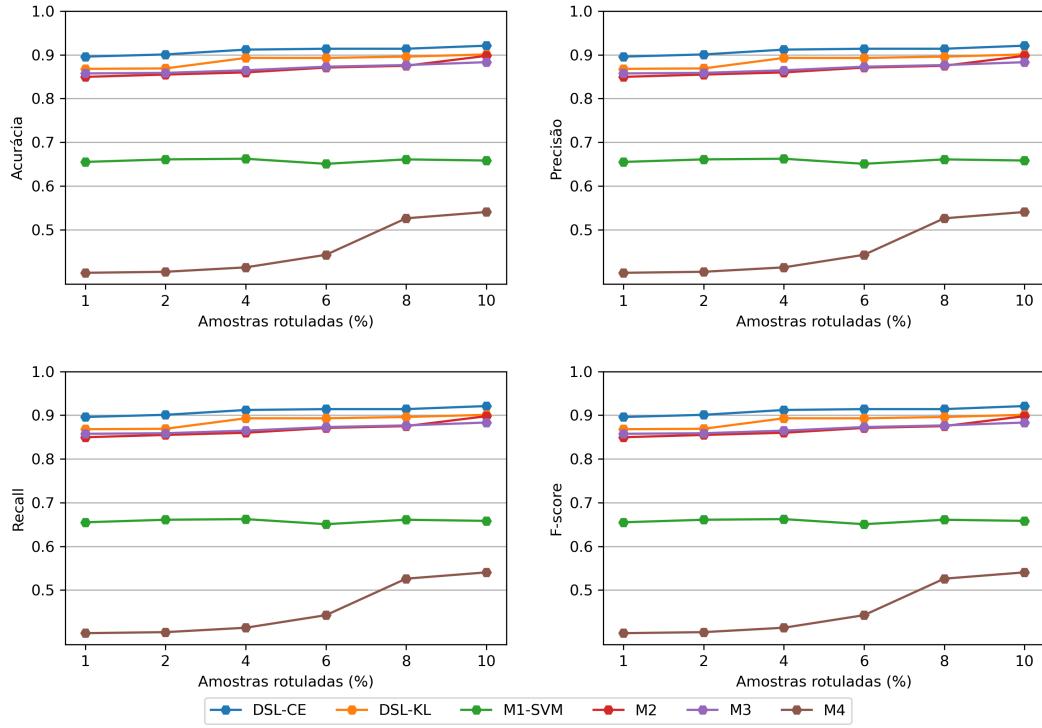


Figura 6.24: Rotulação Base EPILEPSIA na fase transdutiva com modelos da literatura

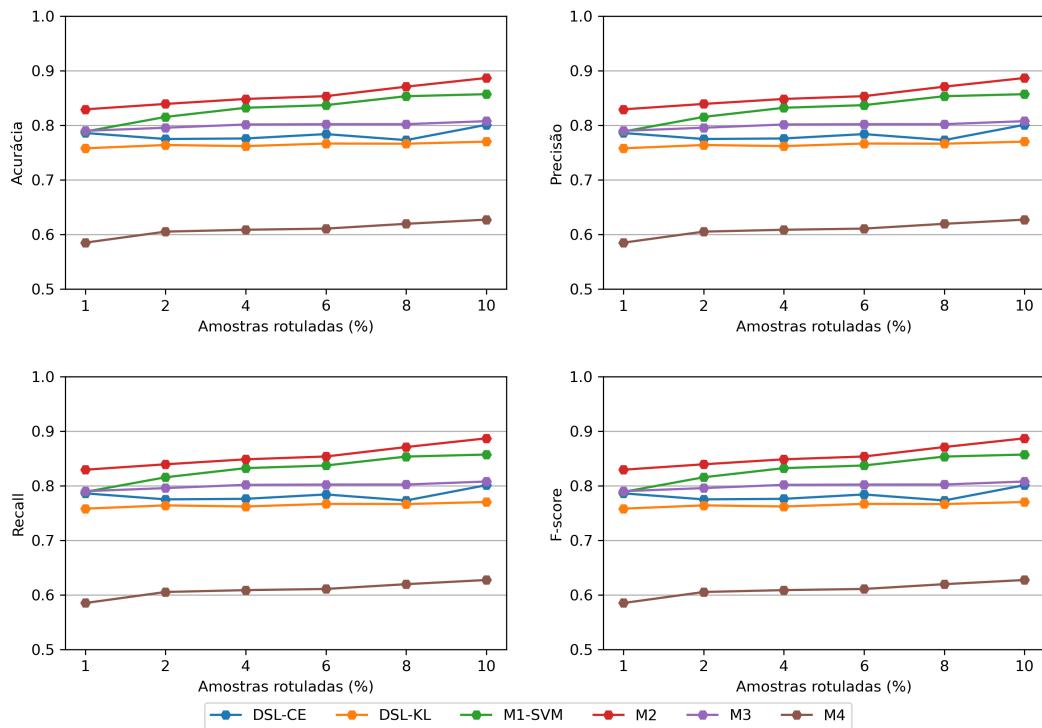


Figura 6.25: Rotulação Base REUTERS na fase transdutiva com modelos da literatura

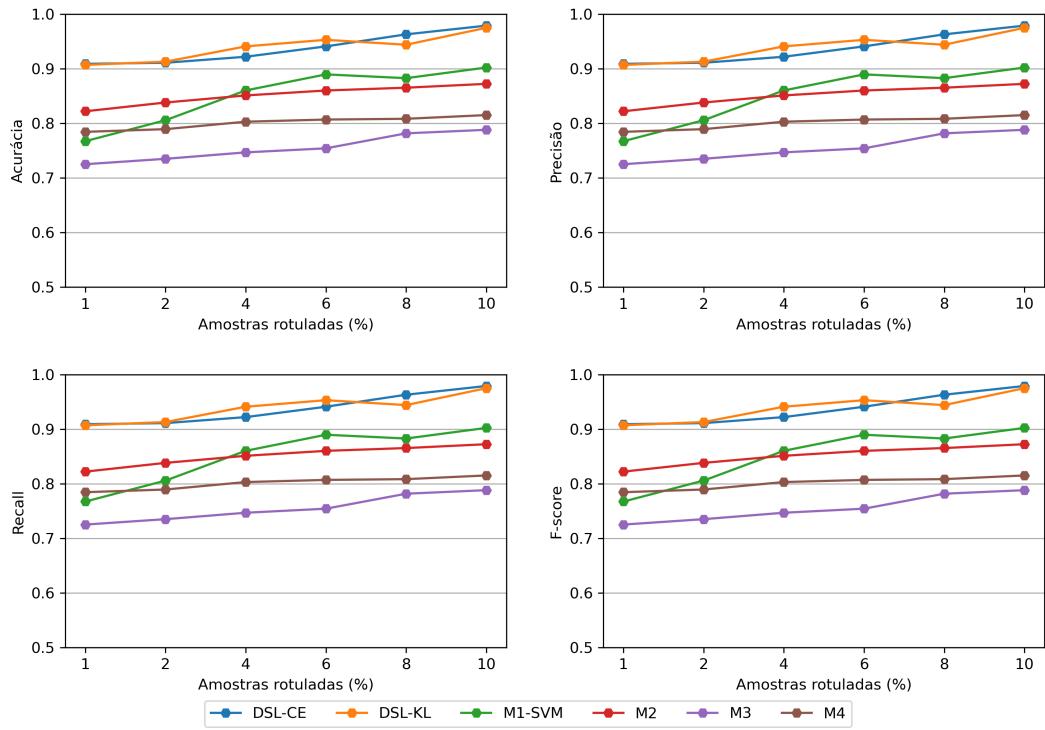


Figura 6.26: Rotulação Base de GÁS na fase transdutiva com modelos da literatura

Da mesma forma, é o comportamento das taxas de precisão e *recall*, que consequentemente afeta a taxa de *f-score*.

Na rotulação das bases de GÁS e PENDIGITS, como podem ser vistas nas Figuras 6.26 e 6.27, o modelo proposto apesentou resultados superiores em todas as métricas avaliadas e em todas as porcentagens de dados rotulados. o DSL obteve taxa de acurácia acima de 0.9 para todas as métricas nas duas bases. Apenas o modelo M2 apresentou acurácia acima de 0.9, com a base PENDIGITS.

Assim como as bases de *benchmark*, tem-se os resultados da rotulação das bases de imagens, comparando o modelo proposto com os modelos citados. ver na Figura 6.28 a rotulação da base MNIST, onde o modelo proposto obteve acurácia acima de 0.95, tanto com a divergência de Kullback-Leibler quanto a entropia cruzada. notar com essa base, que os modelos da literatura apresentaram resultados também satisfatórios, no geral melhores do que os das bases *benchmark*. Apenas o modelo *M1 – SVM* apresentou acurácia abaixo de 0.85 com 1% e 2% das amostras rotuladas. Os resultados são similares nas taxas de precisão e *recall*, indicando que a rotulação também possui bons resultados considerando as classes individualmente.

Na Figura 6.29, tem-se os resultados da rotulação da base MNIST, na fase transdutiva. Neste caso o modelo *M2* apresentou pior desempenho em relação aos outros, com taxa de acurácia abaixo de 0.8 com 1%, 2% e 4% de amostras rotuladas. Os demais, incluindo o modelo proposto, apresentaram acurácia acima de 0.8 em todas as métricas avaliadas.

Nas duas bases de dados (MNIST e FASHION) o modelo proposto apresentou resultados promissores. Nota-se também, que pouco influenciou a quantidade de amostras

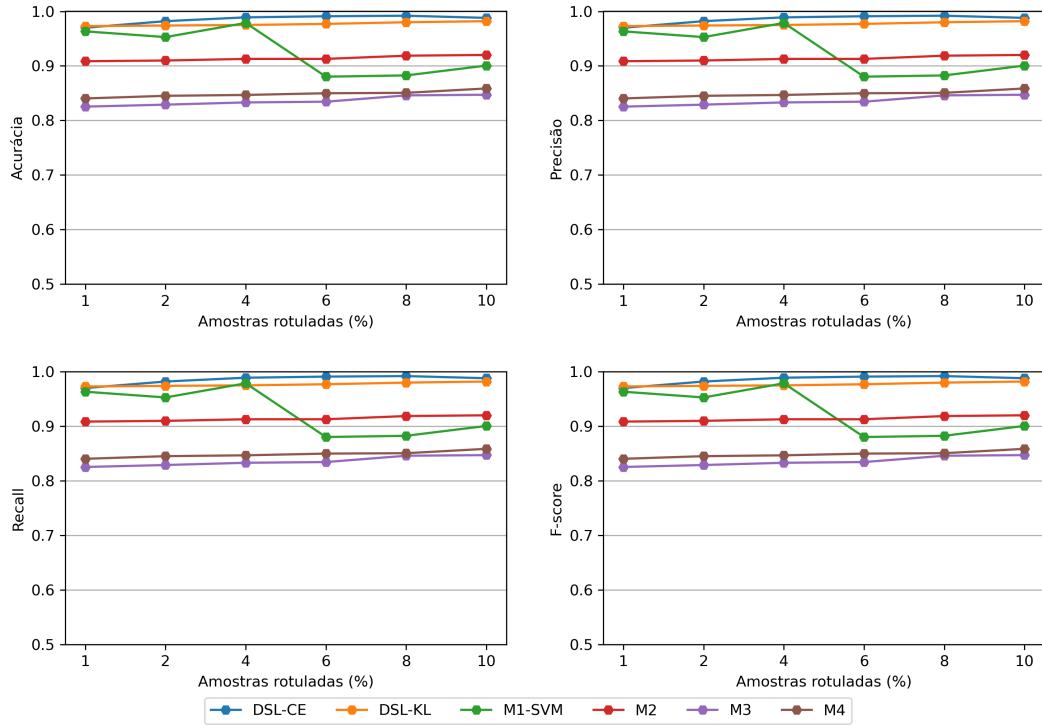


Figura 6.27: Rotulação Base PENDIGITS na fase transdutiva com modelos da literatura

rotuladas no desempenho do DSL.

Nos gráficos das Figuras 6.30 e 6.31 ver os resultados da rotulação das bases CIFAR-10 e STL-10, respectivamente. Para essas duas bases de dados, os modelos da literatura tiveram mais dificuldades em rotular a base. O DSL mesmo com tendo taxa de acurácia menor com 1%, 2% e 4% de amostras rotuladas em relação à 6%, 8% e 10%, ainda sim, foram taxas de acurácia acima de 0.9, com exceção do DSL usando divergência de kullback-Leibler na base STL-10 e 1% das amostras, onde a acurácia obtida foi abaixo de 0.9.

O modelos com melhor desempenho, além do DSL, foi o M3 ((Lin et al. 2020)), que apresentam taxas acima de 0.9. O modelo *M2*, com SVM, apresentou acurácia acima de 0.9 apenas quando foram consideradas 8% e 10% de amostras rotuladas.

A taxa de precisão também foi similar à taxa de acurácia, para o modelo proposto, indicando que o DSL possui uma boa capacidade de rotular uma amostra de uma classe como sendo realmente daquela classe. E uma taxa de *recall* também promissora, mostrando que o DSL rotula em uma frequência boa em uma determinada classe, quando a amostra é da classe. E a taxa de *f-score*, mostrando a harmonia entre a taxa de precisão e acurácia, indicando que os resultados não foram afetados quando se avalia os resultados considerando as classes individualmente.

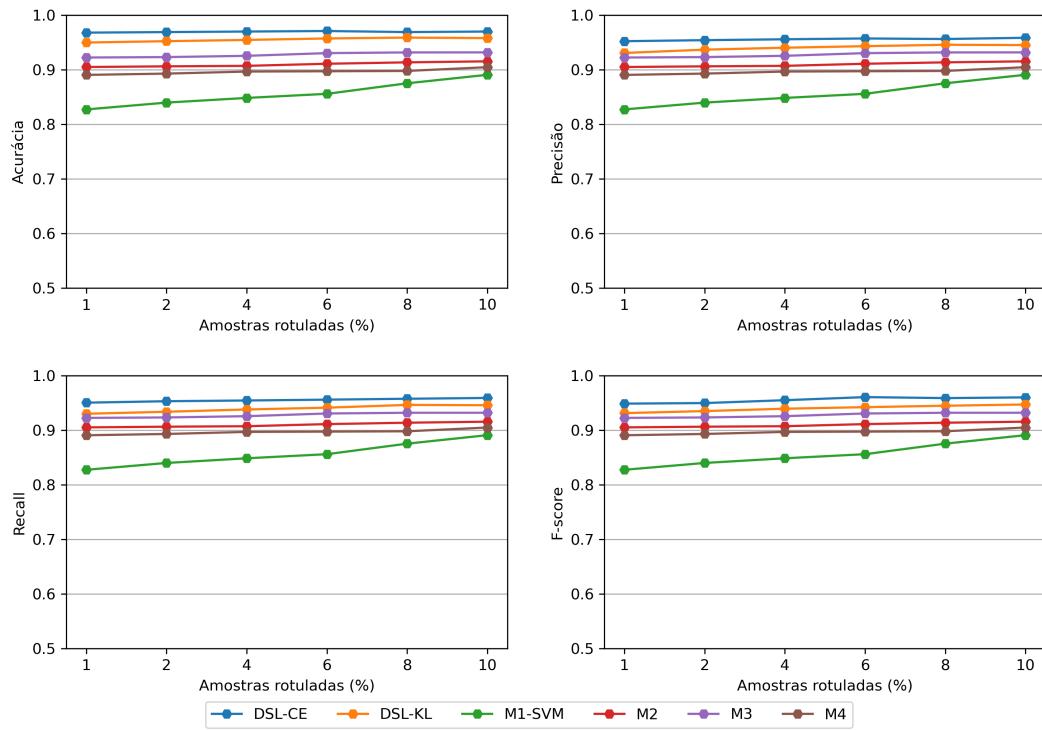


Figura 6.28: Rotulação Base MNIST na fase transdutiva com modelos da literatura

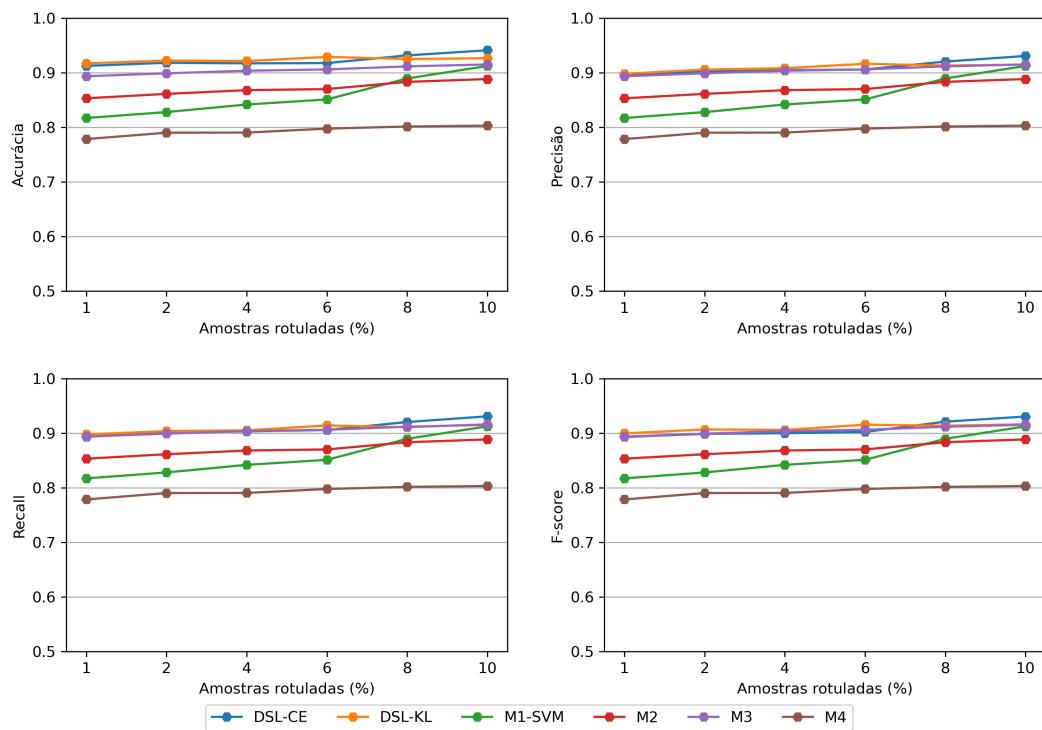


Figura 6.29: Rotulação Base FASHION na fase transdutiva com modelos da literatura

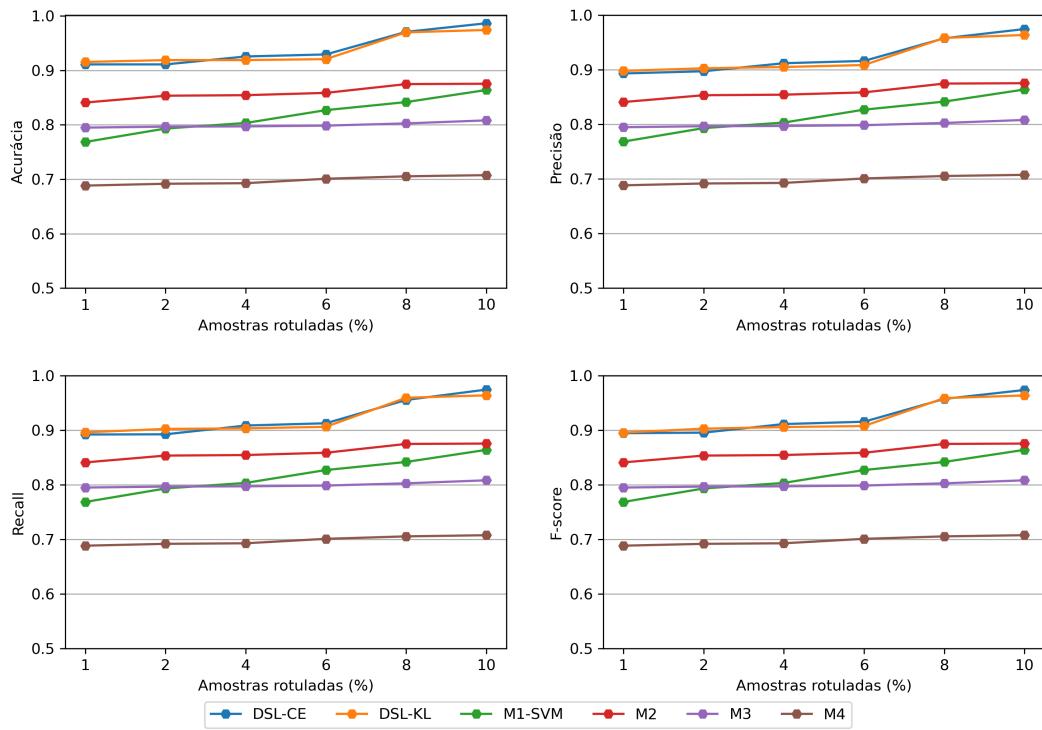


Figura 6.30: Rotulação Base CIFAR-10 na fase transdutiva com modelos da literatura

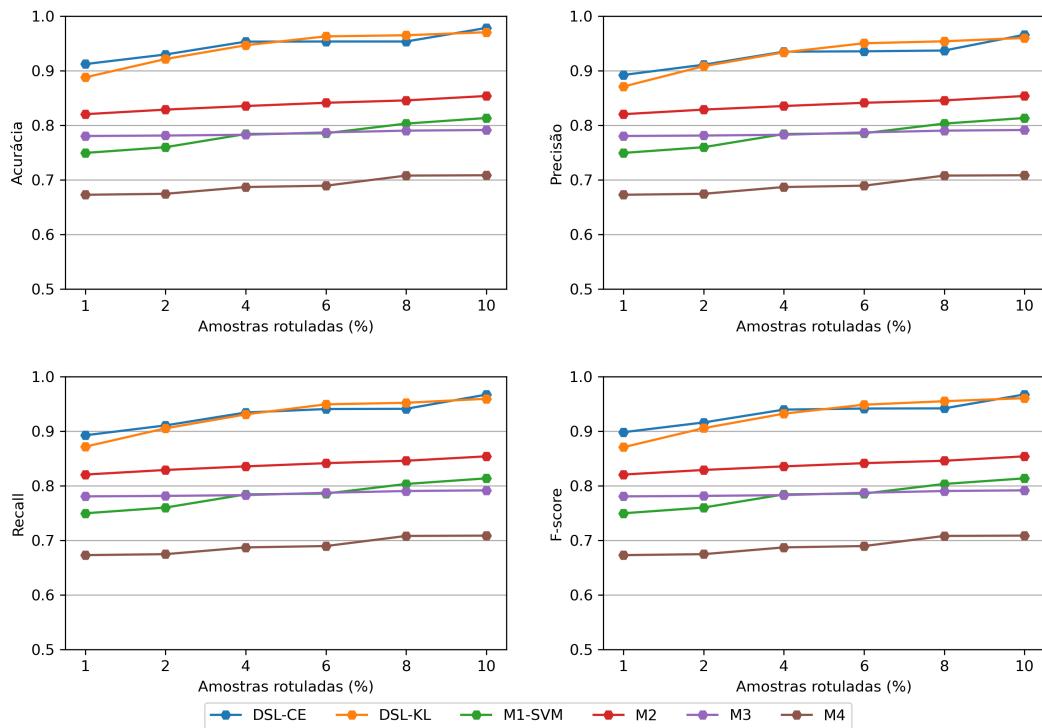


Figura 6.31: Rotulação Base SLT-10 na fase transdutiva com modelos da literatura

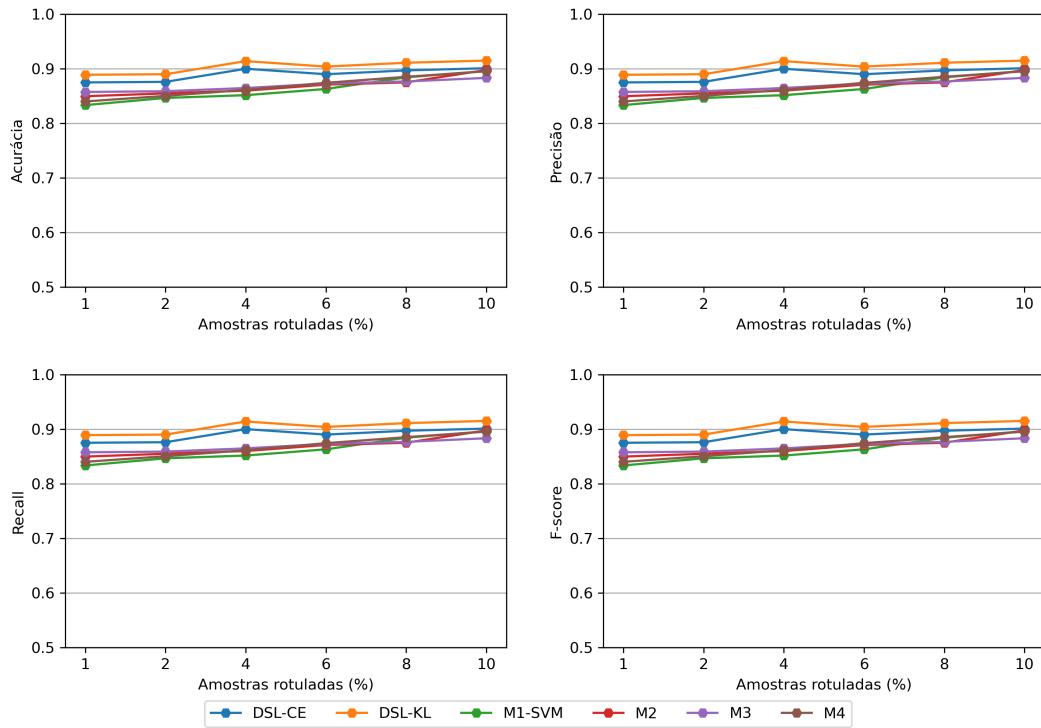


Figura 6.32: Classificação Base EPILEPSIA na fase induutiva com modelos da literatura

6.3.2 Resultado da fase induutiva com modelos da literatura

Nesta subsecção, mostrou-se os resultados da classificação do conjunto de teste por parte dos modelos semissupervisionados, após o processo de treinamento. Observar na Figura 6.24 que a classificação (indutiva) tende a ser satisfatória quando a rotulação (transdutiva) apresenta bons resultados. Os resultados da classificação na fase induutiva mostram que o modelo proposto foi capaz de generalizar o problema e predizer corretamente amostras que não foram utilizadas no treinamento.

Os gráficos das Figuras 6.32, 6.17, 6.34 e 6.19, apresentam os resultados da fase induutiva para as bases EPILEPSIA, REUTERS, GÁS e PENDIGITS respectivamente. Analisando, nota-se que o resultado é similar ao da rotulação apresentada na subsecção anterior.

O modelo proposto obteve na classificação da base 6.32 taxa de acurácia média acima de 0.85, em qualquer uma das duas métricas de similaridade adotada. Na rotulação da base GAS, apresentou desempenho visivelmente superior aos demais modelos, quando com taxa de acurácia acima de 0.9, com todas as porcentagens de dados rotulados consideradas. Na base PENDIGITS, todos os modelos obtiveram resultados satisfatórios, mas mesmo assim, o modelo DSL apresentou melhores taxas de acurácia.

Assim como na rotulação (Figura 6.25), o modelo proposto obteve maiores dificuldades com a base REUTERS, sendo assim, o modelo M2 apresentou resultados superiores ao DSL em todas as porcentagens de dados rotulados, como ver na Figura 6.33.

Em todos os casos, as taxas de precisão e *recall*, apresentaram-se condizentes com a

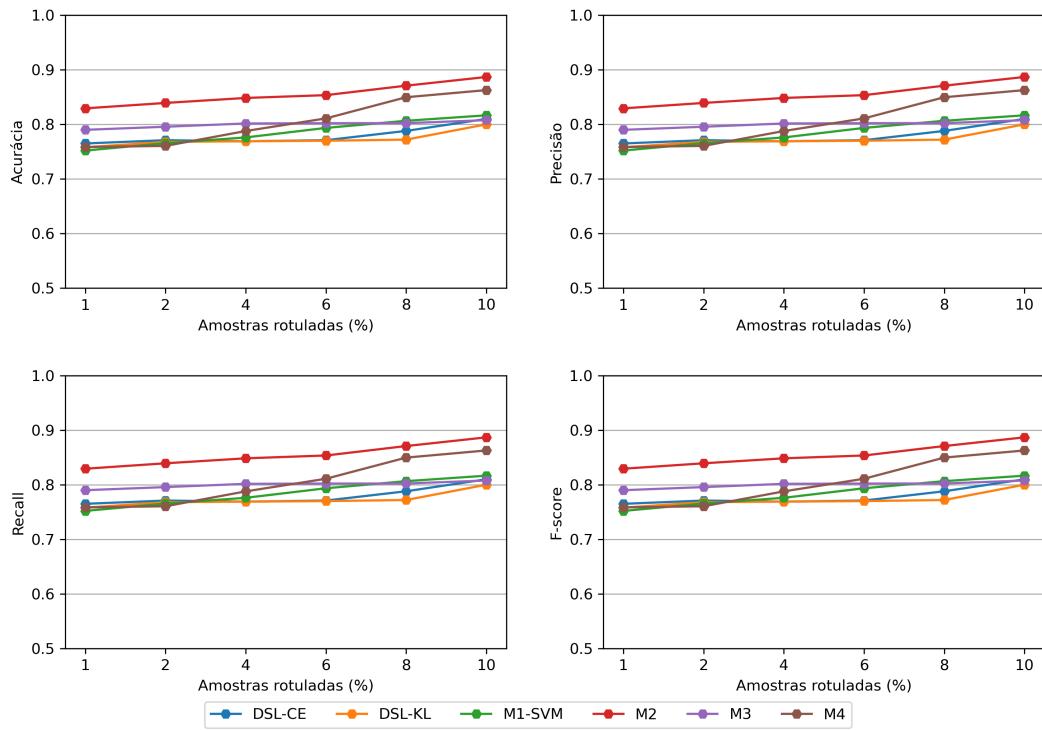


Figura 6.33: Classificação Base REUTERS na fase induutiva com modelos da literatura

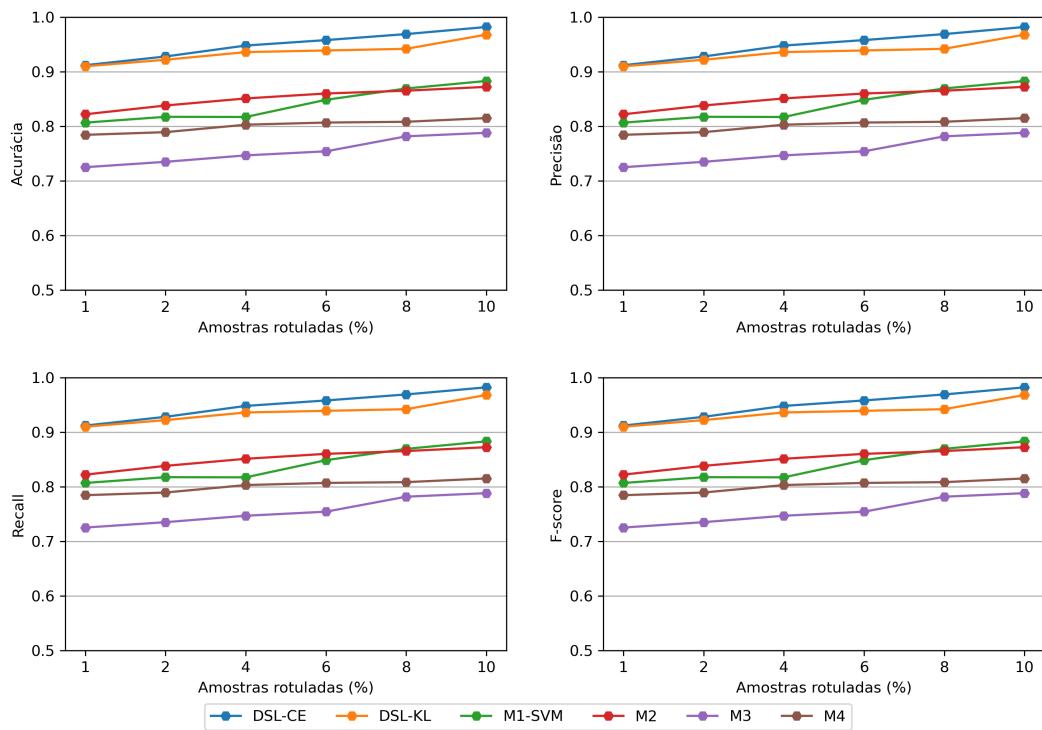


Figura 6.34: Classificação Base de GÁS na fase induutiva com modelos da literatura

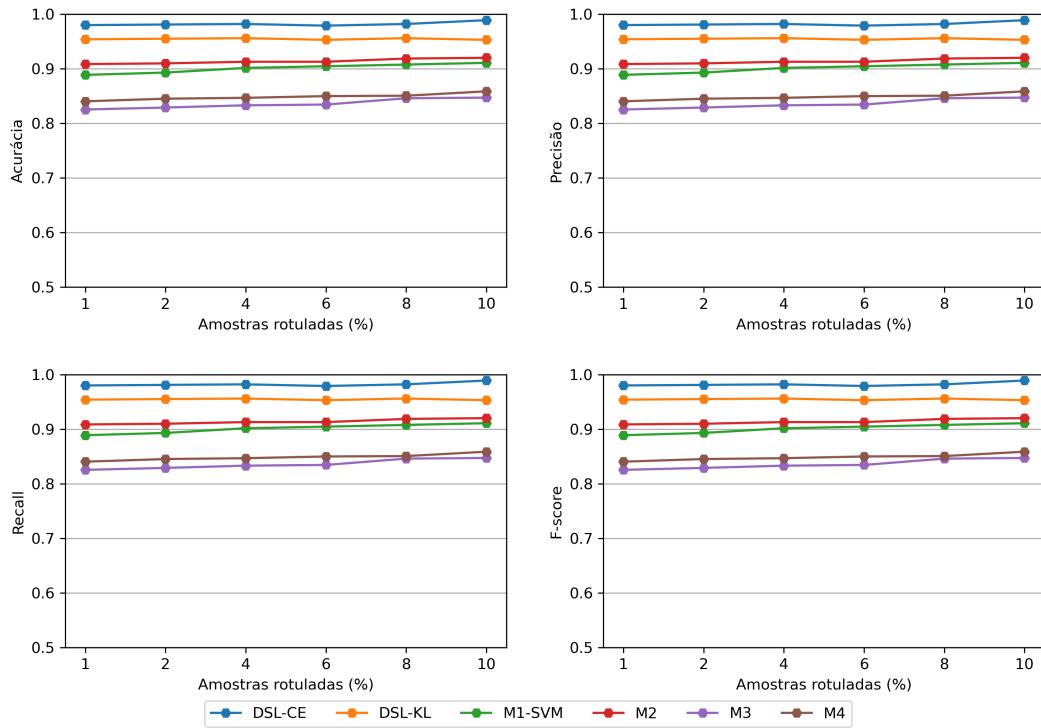


Figura 6.35: Classificação Base PENDIGITS na fase induutiva com modelos da literatura

taxa de acurácia. Isso significa que o modelo classificou corretamente em todas as bases, considerando as classes, não apenas no geral. A taxa de *f-score* mostra, em todos os resultados, uma harmonia entre precisão e *recall*.

De forma similar, os resultados das classificações das bases de imagem, apresentados abaixo, também mostraram um desempenho satisfatório, não só do modelo DSL, quanto dos modelos da literatura.

As Figuras 6.36 e 6.37, mostram os resultados da classificação das bases MNIST e FASHION, respectivamente. Assim como na rotulação (Figuras 6.36 e 6.36), o modelo proposto obteve resultados satisfatórios, significando que foi capaz de generalizar o problema e classificar amostras não previstas no treinamento. O DSL obteve taxa de acurácia acima de 0.9 na classificação das bases MNIST e FASHION, como pode ser visto nas Figuras 6.36 e 6.37.

Observa-se também, que considerando as classes de forma individual, obteve-se bons resultados, pois a taxa de precisão e *recall* foram similares à taxa de acurácia, ou seja, ficaram acima de 0.9, o que reflete na taxa de *f-score*, que ficou também acima de 0.9, indicando que os resultados são promissores.

E por fim, a classificação das bases CIFAR-10 e SLT-10, que também de forma similar à rotulação, os resultados foram que o modelo DSL apresentou melhor desempenho, tanto na CIFAR-10 e SLT-10. Isso pode ser observado nas Figuras 6.38 e 6.35, onde as taxas de acurácia apresentadas são acima de 0.9 na base CIFAR-10, com 2% dos rotulados ou mais (Figura 6.38). Da mesma forma, na base SLT-10 (Figura 6.35). Esses resultados foram obtidos com DSL tanto divergência de *Kullback-leibler* quanto a Entropia Cruzada.

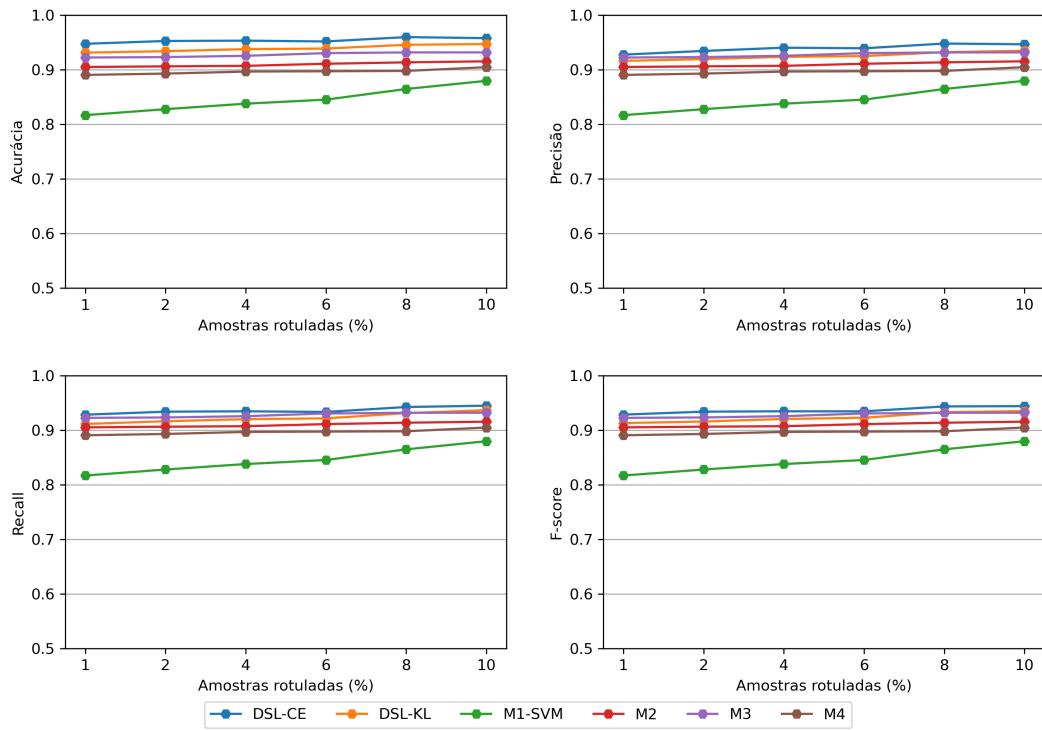


Figura 6.36: Classificação Base MNIST na fase induutiva com modelos da literatura

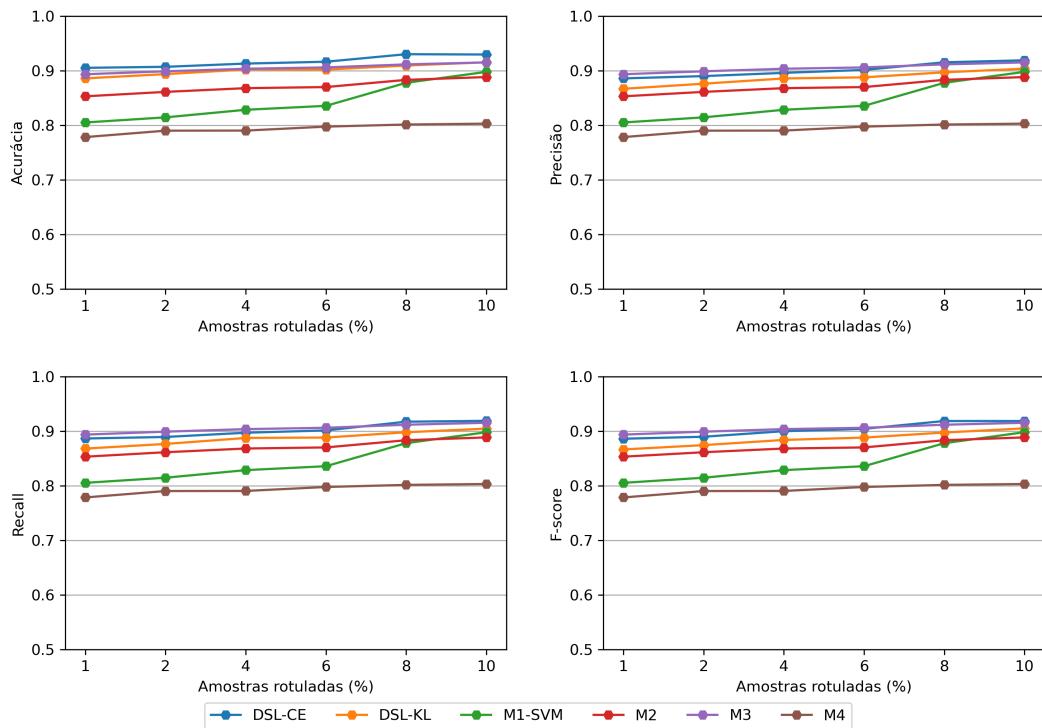


Figura 6.37: Classificação Base FASHION na fase induutiva com modelos da literatura

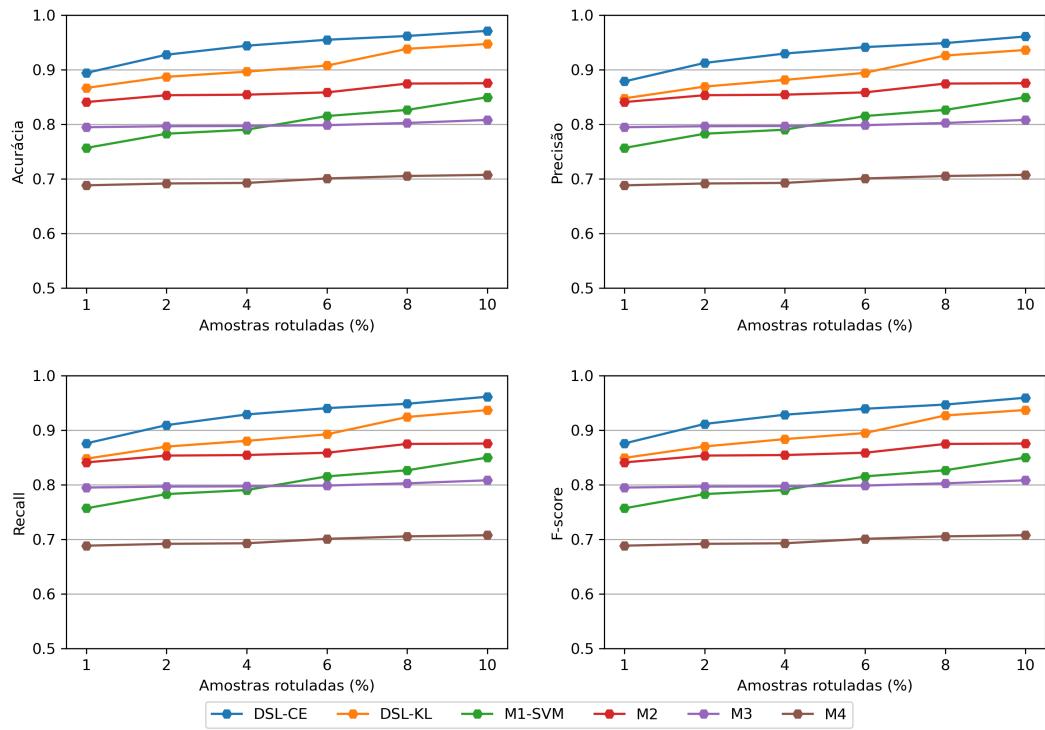


Figura 6.38: Classificação Base CIFAR-10 na fase induutiva com modelos da literatura

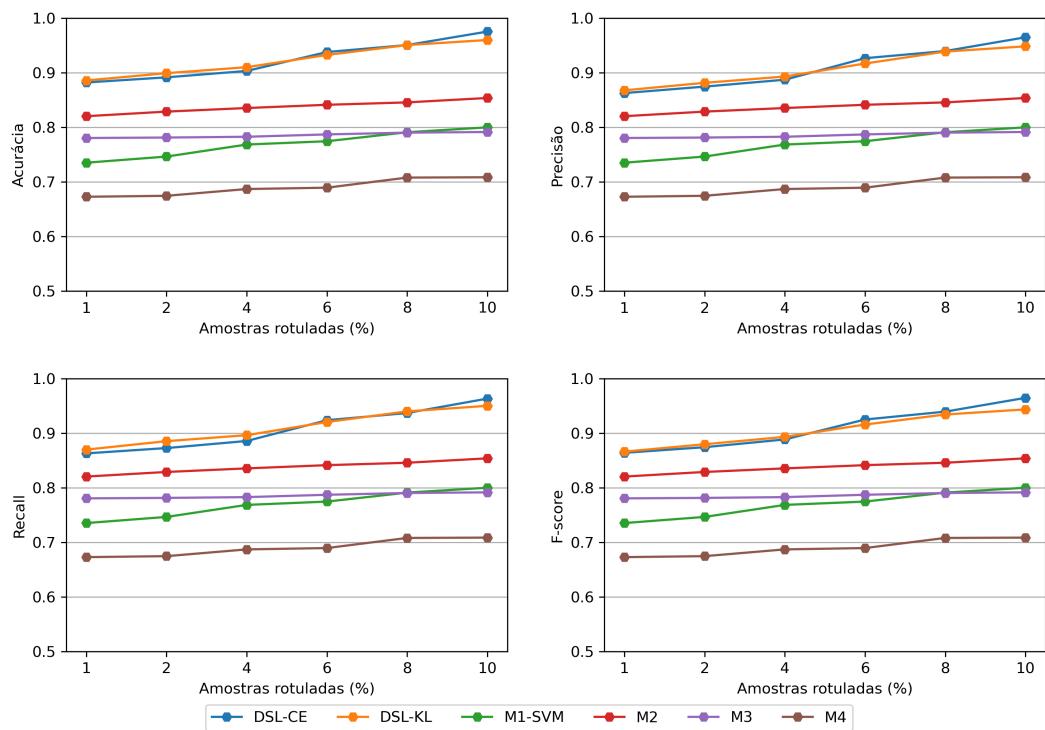


Figura 6.39: Classificação Base SLT-10 na fase induutiva com modelos da literatura

6.3.3 Análise Estatística

Estatisticamente o modelo proposto apresentou diferença significativa para os modelos da literatura (M1, M2, M3 e M4) com 1% à 8% de dados rotulados em todas as métricas avaliadas para a base de EPILEPSIA, pois apresentou valor de $p > 0.05$ nestas situações. De forma similar à fase transductiva, não apresentou diferença significativa com 10% dos dados rotulados, pois apresentou $p < 0.05$ em todas as métricas.

Para a base REUTERS, o modelo proposto apresentou resultados inferiores ao modelo M2, confirmados pelo teste estatístico, apresentando um $p > 0.05$ entre o DSL e M2. Para os demais modelos, o DSL não apresentou diferenças significativas, obtendo um $p < 0.05$. Para as bases GÀS e PENDIGITS, o modelo apresentou resultados superiores estatisticamente, ou seja, para todos os modelos, todas as métricas e todas as porcentagens de dados rotulados, o DSL apresentou $p > 0.05$ comparando com todos os modelos da literatura.

Na classificação da base MNIST, o DSL apresentou resultados estatisticamente superiores apenas aos modelos M1-SVM e M4 com $p > 0.05$, para os demais apresentou resultados equivalentes estatisticamente, ou seja, o valor de p ficou abaixo de 0.05. De forma similar, aconteceu com a base FASHION MNIST.

Para as bases de dados CIFAR-10 e SLT-10, o modelo proposto apresentou resultados significativamente diferentes dos demais modelos, sendo superiores, com p acima de 0.05. Isso, para todas as métricas e porcentagens de dados rotulados.

6.4 Considerações

Neste capítulo, avaliou-se o comportamento do modelo proposto (DSL) com diferentes valores para o hiperparâmetro k (1 à 25). Verifica-se que o valor de k pode influenciar no desempenho do DSL, sendo necessário uma análise prévia para definir um valor de k ideal para o problema abordado. Considerou-se que isso seja, por enquanto, uma limitação do DSL. Avaliou-se também o valor do hiperparâmetro t , que de acordo com os gráficos das Figuras 6.7 e 6.6 pode-se notar que o valor de t muito baixo pode comprometer o desempenho do modelo.

Os resultados dos experimentos, apontam que o DSL foi capaz de rotular na fase transductiva as bases de dados abordadas, ou seja, treinou utilizando dados rotulados e não rotulados simultaneamente e rotulou de forma satisfatória a parte não rotulada das bases. Nota-se também, a eficiência do modelo proposto na fase indutiva, ou seja, quando o modelo é colocado para classificar amostras não previstas no treinamento.

Na maioria dos casos o modelo proposto obteve resultado superior aos modelos da literatura, com algumas poucas exceções. Isso mostra que o modelo proposto apresenta desempenho competitivo com os demais modelos. Em algumas bases de dados os demais modelos, apresentaram desempenho superior, devido as limitações de cada um destes.

Por exemplo, o self-training e Co-training, utilizando algoritmos padrões de aprendizados e máquina (MLP, SVM, e RF), apresentaram dificuldades com bases com muitos atributos, como é o caso das bases CIFAR10 e STL10. Nesses aspectos, o fato do modelo proposto utilizar técnica de *deep learning*, para reduzir a dimensionalidade das bases, faz o DSL ter desempenho superior aos demais, como pode ser visto nos resultados.

O modelo proposto apesar de ter apresentado desempenho satisfatório, ainda apresenta algumas limitações, tais como foi dito anteriormente, é necessário uma investigação prévia para definir os valores ideais do hiperparâmetro k e t . Caso esse estudo não seja realizado de forma adequada, pode levar o DSL à desempenhos insatisfatórios, dependendo do problema abordado.

Nota-se também, no caso das bases com um maior número de classes, no caso da base REUTERS, o DSL apresentou pior resultados em relação às bases de EPILEPSIA, GÁS e PENDIGITS. Vale ressaltar, que essas limitações não invalidam o modelo proposto, pois este apesar disso, apresentou desempenho satisfatório no problema de classificação semissupervisionada, como pode ser visto nos resultados apresentados. Vale ressaltar, que no Anexo A contém resultados do modelo M1 utilizando a Rede *Multilayer Perceptron* e o *Random Forest* como classificadores base.

Capítulo 7

Estudo de Casos

Além dos experimentos realizados com as bases **benchmark** e de imagens, aplicou-se o modelo proposto em um problema de classificação de imagens no contexto de Sensoriamento Remoto.

7.1 Mapeamento de áreas agrícolas no Cerrado brasileiro

A integração de grandes volumes de dados orbitais em aplicações de Sensoriamento Remoto (SR) voltadas para estudos ambientais, vem ganhando espaço na literatura (Colditz et al. 2011)(Feranec et al. 2010). Esse grande volume de dados tem permitido a utilização de técnicas de *Deep Learning* (DL) para realizar a classificação supervisionada de dados espectrais (Yao et al. 2019) (Zhu et al. 2019) (Koda et al. 2019). Com isso, diversos trabalhos aplicam técnicas de Machine Learning no sensoriamento remoto em inúmeras áreas, tais como, mapeamento gestão de recursos, agricultura de precisão, modelagem de serviços ecossistêmicos, mudanças climáticas e planejamento urbano dentre outros.

Neste trabalho, destaca-se aplicação no mapeamento de atividades agrícolas na Região Geográfica Intermediária (RGI) de Uruçuí, no Sul do Estado do Piauí, Brasil. A região faz parte da área chamada MATOPIBA, cujo nome é em referência a zona formada pelos estados brasileiros Maranhão (Ma), Tocantis (To), Piauí (Pi) e Bahia (Ba). O MATOPIBA é uma recente fronteira agrícola no domínio do Cerrado no Nordeste brasileiro. A região da MATOPIBA abrange 337 municípios e 73 milhões de hectares, e em 2015 o governo brasileiro estabilizou o Plano de Desenvolvimento da Agricultura para MATOPIBA.

A área de estudo deste trabalho é a Região Geográfica Imediata (RGI) de Uruçuí, composta por sete municípios, são eles: Antônio Almeida, Baixa Grande do Ribeiro, Berrolândia, Manoel Emídio, Ribeiro Gonçalves, Sebastião Leal e Uruçuí, reconhecidos por intensa atividade agrícola. E está localizada na porção noroeste da mesorregião sudoeste piauiense, com variações de 152 a 651 metros em relação ao nível do mar (Figura 7.1).

7.1.1 Classificação Semissupervisionada

A RGI de Uruçuí é composta por uma multiplicidade de estruturas produtivas (cultivo de soja, milho, algodão, arroz, cana-de-açúcar, dentre outras) a área de estudo foi es-

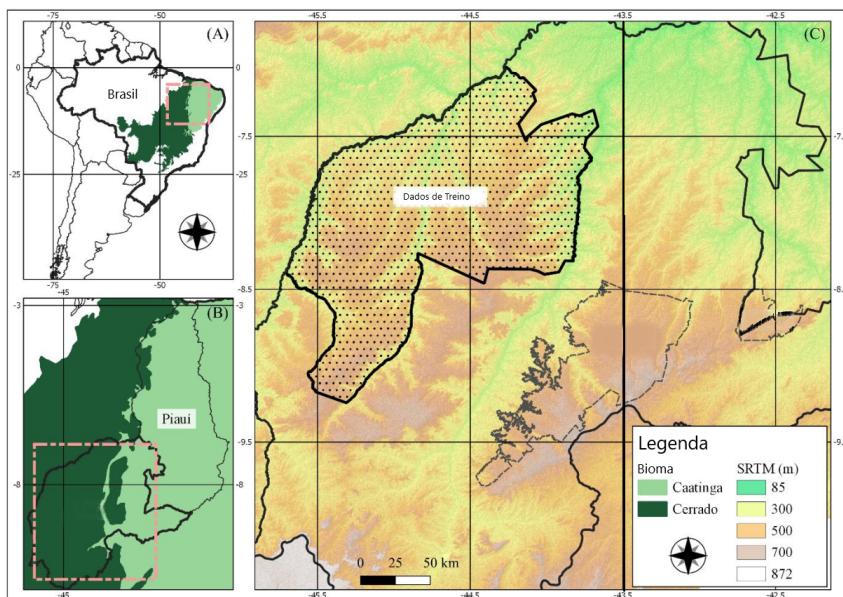


Figura 7.1: Área de Estudo do MATOPIBA

cializada em 6 classes que incluem Água, Vegetação Cerrado, Atividade Agrícola, Solo Exposto, Mata Ciliar e Encosta de Serra. A área de estudo possui 2.718.986,40 ha, sendo considerada uma amostra representativa para a aplicação da metodologia experimental sugerida.

O objetivo é realizar a classificação da região em cada uma das classes definidas para auxiliar na análise posterior em relação à evolução da atividade agrícola. O modelo proposto gera mapas temáticos de uso e cobertura da terra para a região noroeste da mesoregião sudoeste piauiense, assinalado no B) na Figura 7.1, podemos ver a região, com preenchimento pontilhado no mapa, de onde foram retiradas as amostras de treinamento. A Figura 7.2, apresenta a área de estudo em composição RGB, permitindo visualizar melhor a região estudada, onde foi realizada a classificação.

Baseado na Figura 7.2, pode-se definir as classes e visualizar na composição RGB cada uma delas. Na Figura 7.3 apresenta imagens referente à cada uma das classes definidas no problema. Podemos ver a foto de campo panorâmica (A), e do lado a Composição RGB, para melhor visualizar no mapa, como visto na Figura 7.2.

Para realizar o treinamento e classificação, tornou-se necessário ainda coletar amostras, neste caso por um especialista. Para extraír os dados para formar a base de dados, foi utilizada a ferramenta *Google Earth Engine*¹, que é uma ferramenta da *Google*, que permite trabalhar com imagens sem sensoriamento remoto, através do *Google Maps*.

A base de dados extraída é composta por metadados extraídos das bandas Espectrais 2 a 7, estas em níveis de refletância calibrada para o *Top of Atmosphere* (TOA) - camada 1, do satélite Landsat-8, que possui resolução espacial de 30 x 30 m nestas respectivas bandas, adquiridas no período de agosto do ano 2013. Foram coletadas um total de 11966 amostras.

¹<https://code.earthengine.google.com/>

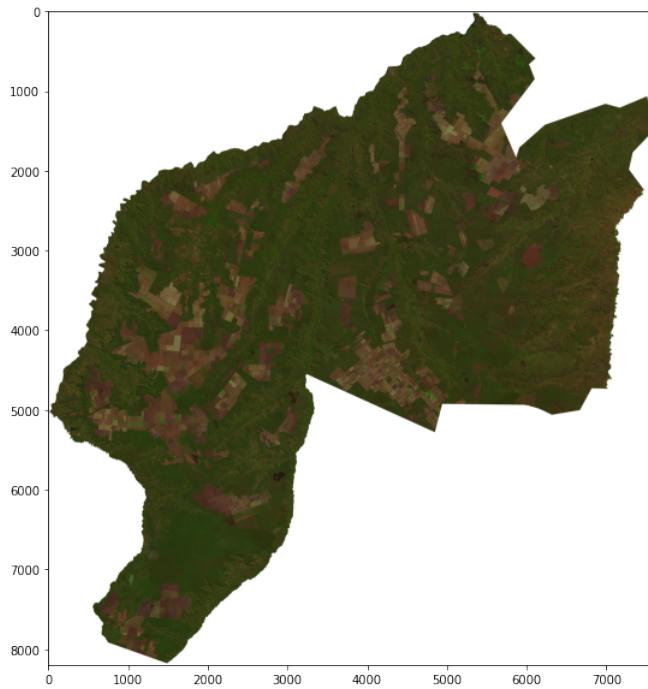


Figura 7.2: Área de estudo em composição RGB

Para realizar a classificação dividiu-se a base de dados em dois subconjuntos: 1) Teste, que foi utilizado para avaliar os modelos na tarefa de classificação de novas amostras; 2) Treino, que foi subdividido em dados rotulados (X^L) e não rotulados (X^U), sendo a parte rotulada variou-se em 50, 100, 150, 200, 250 e 300 amostras. Utilizamos a abordagem de validação cruzada, como nos experimentos anteriores, com isso cada experimento foi realizado 10 vezes. Utilizamos as métricas acurácia e índice kappa (Sokolova e Lapalme 2009) para avaliar os modelos. A base de dados foi normalizada utilizando o método *MinMax*. Para gerar o mapa de classificação, escolhemos uma cor para cada classe do problema como pode ser visto na Tabela 7.1.

Tabela 7.1: Cores das classes no mapa de classificação

	Mata Ciliar
	Atividade Agrícola
	Solo Exposto
	Água
	Floresta de Cerrado
	Encosta da Serra

O modelo proposto, foi executado utilizando um *Autoencoder* Convolucional, com 3 camadas de convolução, e uma rede conectada de 1 camada oculta. Utilizando o método de busca em grade, definimos a Entropia Cruzada como função custo e método de similaridade na rotulação, $k = 5$ e $t = 0.9$.

Para comparação, utilizamos os modelos clássicos semissupervisionados: *self-training*

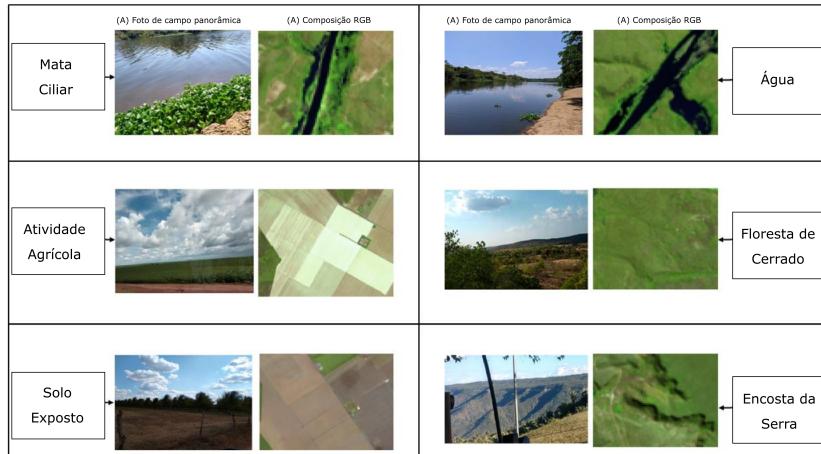


Figura 7.3: Classes definidas no problema

Co-training e Tri-training. Como classificador base utilizamos o Random Forest (RF), Máquina de Vetor de Suporte (SVM) e Multilayer Perceptron (MLP). Todos os Hiperparametros definidos por meio de busca em grade, sendo os mesmos utilizados na Tabela 6.2.

7.1.2 Resultados da Classificação

A Tabela 7.2 apresenta os resultados dos modelos semissupervisionados na fase transductiva, ou seja, a capacidade de rotular os dados X^U (não rotulados) à partir da pequena amostras rotulada X^L . Em comparação aos modelos da literatura, o modelo proposto apresentou melhor acurácia na fase transdutiva, sendo o SEEDED K-means apresentando resultado mais próximo, quando o treinamento foi realizado com 50 amostras rotuladas, porém ainda inferior. Com 100 à 300 amostras, o modelo proposto apresentou acurácia acima de 0.9, superior a todos os modelos clássicos.

Os valores das acuráncias na fase indutiva, podem se confirmados com os valores do índice kappa (Tabela 7.2). Tanto o modelo proposto quanto os modelos da literatura apresentaram em todos os casos índice kappa acima de 0.8, confirmando a qualidade dos desempenhos apresentados pelas acuráncias.

A Tabela 7.3 apresenta os resultados dos modelos semissupervisionados na fase de indução, ou seja, a capacidade dos modelos de predizer amostras que não estavam no treinamento.

O modelo proposto apresentou desempenho superior aos demais modelos clássicos semissupervisionados, em todos os casos, com acuráncias próxima de 0.9 com 50 amostras rotuladas e acima de 0.9 com 100 à 300 amostras rotuladas. O modelo SEEDED K-means e o Self-training com SVM e MLP apresentaram resultado similar com 50 amostras rotuladas, porém ainda inferior ao modelo proposto.

Tabela 7.2: Resultados da rotulação na fase transdutiva

ACCURACY						
Labeled Amount	50	100	150	200	250	300
Proposed Model	0.894	0.905	0.895	0.924	0.919	0.925
SEDED K-means	0.855	0.855	0.855	0.854	0.855	0.854
Self-Training (SVM)	0.813	0.822	0.831	0.834	0.839	0.840
Self-Training (MLP)	0.819	0.878	0.881	0.879	0.886	0.890
Self-Training (RF)	0.791	0.821	0.835	0.838	0.842	0.851
Co-training (SVM)	0.802	0.818	0.821	0.824	0.830	0.834
Co-training (MLP)	0.773	0.794	0.788	0.802	0.803	0.803
Co-training (RF)	0.742	0.803	0.818	0.833	0.835	0.839
Tri-Training (SVM)	0.806	0.820	0.827	0.828	0.836	0.835
Tri-Training (MLP)	0.774	0.799	0.798	0.800	0.804	0.806
Tri-Training (RF)	0.776	0.809	0.821	0.830	0.843	0.843

KAPPA						
Labeled Amount	50	100	150	200	250	300
Proposed Model	0.870	0.883	0.871	0.906	0.901	0.908
SEDED K-means	0.824	0.823	0.823	0.821	0.823	0.822
Self-Training (SVM)	0.893	0.904	0.916	0.919	0.925	0.927
Self-Training (MLP)	0.777	0.850	0.854	0.852	0.860	0.866
Self-Training (RF)	0.867	0.903	0.920	0.925	0.929	0.940
Co-training (SVM)	0.880	0.899	0.903	0.907	0.914	0.919
Co-training (MLP)	0.845	0.870	0.863	0.880	0.882	0.881
Co-training (RF)	0.806	0.881	0.900	0.918	0.920	0.925
Tri-Training (SVM)	0.885	0.902	0.910	0.912	0.921	0.920
Tri-Training (MLP)	0.845	0.876	0.875	0.878	0.882	0.885
Tri-Training (RF)	0.848	0.888	0.903	0.915	0.930	0.931

Mapa de Classificação

Considerando os resultados apresentados na sessão anterior podemos analisar o desempenho do modelo proposto em classificar dados extraídos da região de estudo, através do mapa de classificação, que é a representação visual do desempenho do modelo, visualizando o mapa com as cores de cada pixel baseado na Tabela 7.1.

A Figura 7.4 mostra o mapa de classificação da região de estudo. Do lado esquerdo (a), podemos visualizar a área em composição RGB, no lado direito (b) o resultado da classificação, considerando as cores da Tabela 7.1. Esse mapa de classificação, são da imagem considerando ano de 2013. A partir dos resultados obtidos na fase transdutiva e indutiva, e o mapa de classificação visualizado na Figura 7.4, podemos afirmar que o modelo proposto generalizou o problema e obteve desempenho satisfatório na classificação do mapa a partir de dados espectrais.

Pode-se notar, na Figura 7.4b, que as áreas agrícolas foram destacadas, permitindo assim uma análise das atividades agrícolas na região. Podemos também observar melhor

Tabela 7.3: Resultados da classificação na fase indutiva

Labeled Amount	ACCURACY					
	50	100	150	200	250	300
Proposed Model	0.893	0.911	0.897	0.921	0.920	0.922
SEEDED K-means	0.855	0.850	0.857	0.855	0.854	0.853
Self-Training (SVM)	0.813	0.817	0.827	0.837	0.84	0.842
Self-Training (MLP)	0.820	0.876	0.874	0.880	0.885	0.894
Self-Training (RF)	0.791	0.816	0.833	0.843	0.843	0.851
Co-training (SVM)	0.806	0.817	0.818	0.830	0.831	0.834
Co-training (MLP)	0.778	0.789	0.787	0.805	0.806	0.803
Co-training (RF)	0.747	0.803	0.820	0.835	0.834	0.838
Tri-Training (SVM)	0.805	0.819	0.825	0.830	0.835	0.834
Tri-Training (MLP)	0.775	0.795	0.799	0.798	0.804	0.805
Tri-Training (RF)	0.775	0.811	0.820	0.827	0.841	0.843
Labeled Amount	KAPPA					
	50	100	150	200	250	300
Proposed Model	0.869	0.891	0.873	0.903	0.902	0.905
SEEDED K-means	0.823	0.817	0.825	0.823	0.822	0.821
Self-Training (SVM)	0.893	0.898	0.911	0.922	0.926	0.929
Self-Training (MLP)	0.779	0.848	0.846	0.853	0.859	0.870
Self-Training (RF)	0.866	0.898	0.918	0.930	0.930	0.940
Co-training (SVM)	0.884	0.898	0.900	0.914	0.915	0.919
Co-training (MLP)	0.850	0.864	0.861	0.883	0.885	0.88
Co-training (RF)	0.813	0.881	0.901	0.920	0.919	0.925
Tri-Training (SVM)	0.883	0.901	0.908	0.914	0.920	0.919
Tri-Training (MLP)	0.846	0.871	0.877	0.875	0.882	0.884
Tri-Training (RF)	0.846	0.891	0.902	0.911	0.928	0.930

no mapa temático (Figura 7.4b) em relação à composição RGB (Figura 7.4a), as áreas de atividades agrícolas, auxiliando assim a análise da região em relação à essa classe.

Nota-se também, as áreas de solo exposto, que na composição RGB é mais difícil de se diferenciar das áreas de atividade agrícola. Entretanto, com o mapa de classificação gerado pelo DSL, fica evidente tais regiões.

A Figura 7.5 apresenta os mapas de classificação da região de estudo em três anos, 2013, 2015 e 2019, em um ponto específico do mapa, comparando com suas respectivas composições RGB. Os mapas de classificação da Figura 7.5 foi gerado com o modelo proposto.

Quando aproxima-se a imagem em um ponto específico, fica mais evidente que a classificação gerada pelo DSL, facilitou a análise em relação às atividades agrícolas na região.

Pode-se perceber também, que o modelo proposto conseguiu gerar o mapa de classificação para outros anos, além do ano de 2013, cujo os dados foram extraídos. Permitindo assim uma análise do comportamento das atividades agrícola no passar dos anos.

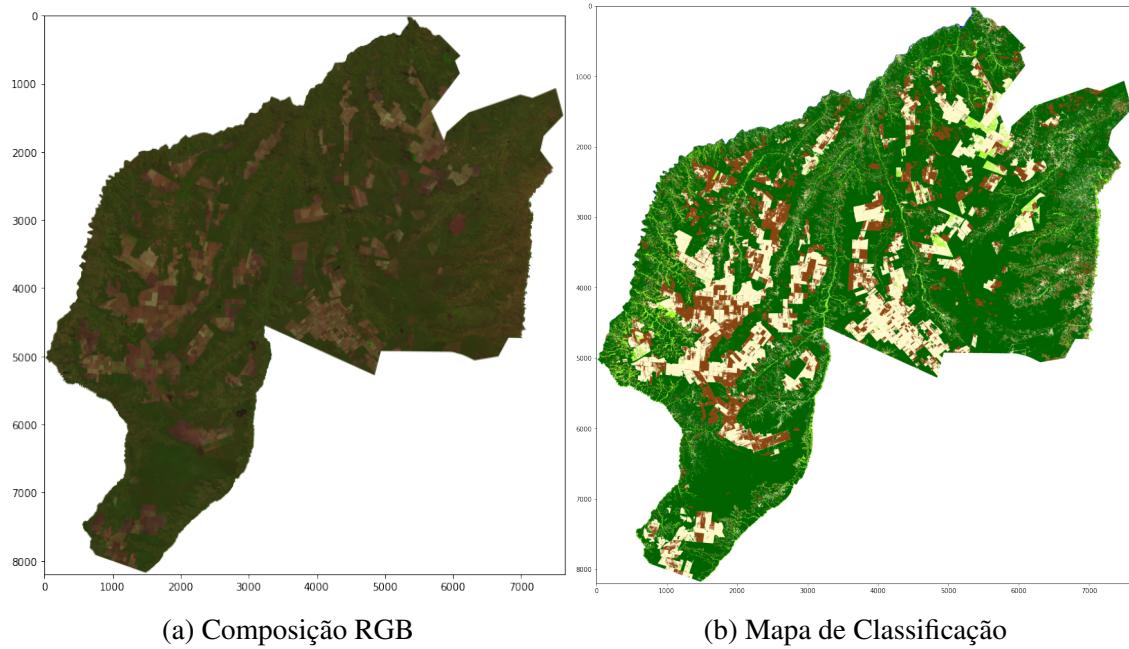


Figura 7.4: Área do MATOPIBA classificada com o DSL

Desta forma pode-se verificar a atividade agrícola na região. Para melhor visualizar o resultado, aproximou-se um ponto específico da área de estudo. Verifica-se que o modelo proposto conseguiu classificar os pixels satisfatoriamente, permitindo desta forma determinar o que é região agrícola e o que não é.

7.2 Classificação em Dados de *Stream*

No aprendizado de máquina tradicional, muitas abordagens avançadas foram propostas com base no pressuposto de que o ambiente de aprendizado é estacionário. No entanto, o ambiente de aprendizagem é muitas vezes dinâmico em aplicações práticas, especialmente quando o aprendizado é com dados de *stream* (Zhou 2016).

Na classificação de dados de *streaming*, a maioria dos métodos existentes assume que todos os dados em evolução que chegaram estão completamente rotulados. Um desafio é que alguns aplicativos em que apenas uma pequena quantidade de exemplos rotulados estão disponíveis para treinamento (Li, Wang, Liu, Bi, Jiang e Sun 2019).

Em problemas de aprendizagem semissupervisionados dinâmicos abertos, as instâncias são coletadas sucessivamente de um fluxo de dados. Devido a questões práticas, como restrições de tempo e recursos, apenas alguns dados são rotulados, enquanto uma grande quantidade deles perde as informações do rótulo (Zhou 2016). No início, todas as amostras pertencem a classes conhecidas.

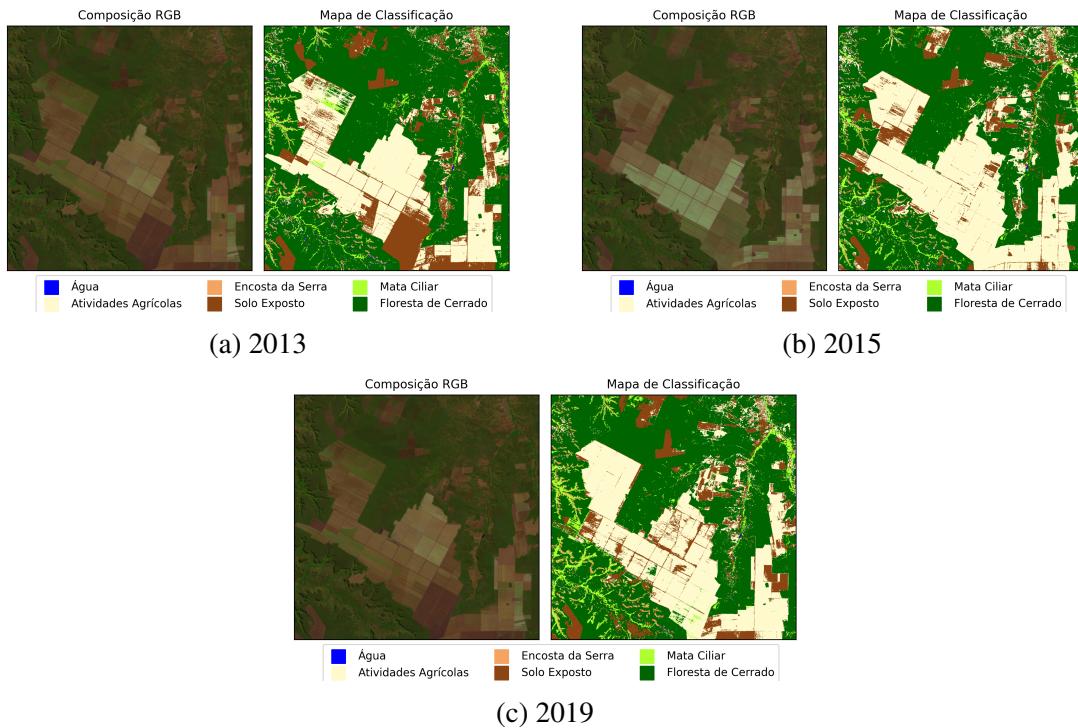


Figura 7.5: Mapa de classificação de um ponto da área de estudo nos anos 2013, 2015 e 2019

7.2.1 Classificação semissupervisionada de dados de *stream*

No problema de classificação de dados de *stream*, temos um conjunto de dados $S = \{X_t\}_{t=1}^{\infty}$, onde cada X_t é uma instância dos dados no tempo t . Como é um problema de aprendizado semissupervisionado, cada temos cada instância $X_t = \{X^U, X^L\}$. Em que $X^L = \{x_i, y_i\}_{i=1}^n$ e $X^U = \{x_i\}_{i=1}^m$, onde $m >> n$. A cada tempo t aumenta a quantidade de dados em S , tanto dados rotulados, quanto dados não rotulados.

Vale ressaltar que a quantidade de classes no problema pode mudar a cada tempo t , pois o fluxo de dados é dinâmico. Ou seja, em um determinado tempo $t + 1$, a quantidade de classes no problema pode ser maior do que no tempo t .

A cada iteração do DSL, o conjunto de grupos podem ser alterados e a camada de rotulação é atualizada. Com isso, podemos adaptar o modelo proposto para a classificação de dados de *stream*. Na Figura 7.6, podemos visualizar o modelo proposto em 3 tempos distintos, sendo a camada de rotulação com quantidades diferentes de neurônios em cada um dos tempos. No caso da Figura 7.6a, temos o DSL com uma camada de rotulação com um total de 3 neurônios no tempo t . Com a chegada de mais dados, a quantidade de classes aumentou para 4, sendo atualizada a camada de rotulação no tempo $t + 1$, como pode ser visto na Figure 7.6b. E na Figura 7.6c temos o DSL com uma camada de rotulação atualizada para 5 classes, no tempo $t + 2$.

No primeiro tempo t , o DSL realiza o treinamento igualmente foi descrito no Capítulo 5. A partir do tempo $t + 1$, o DSL realiza apenas iterações de atualização da camada de rotulação e do conjunto de pesos, otimizando a função custo.

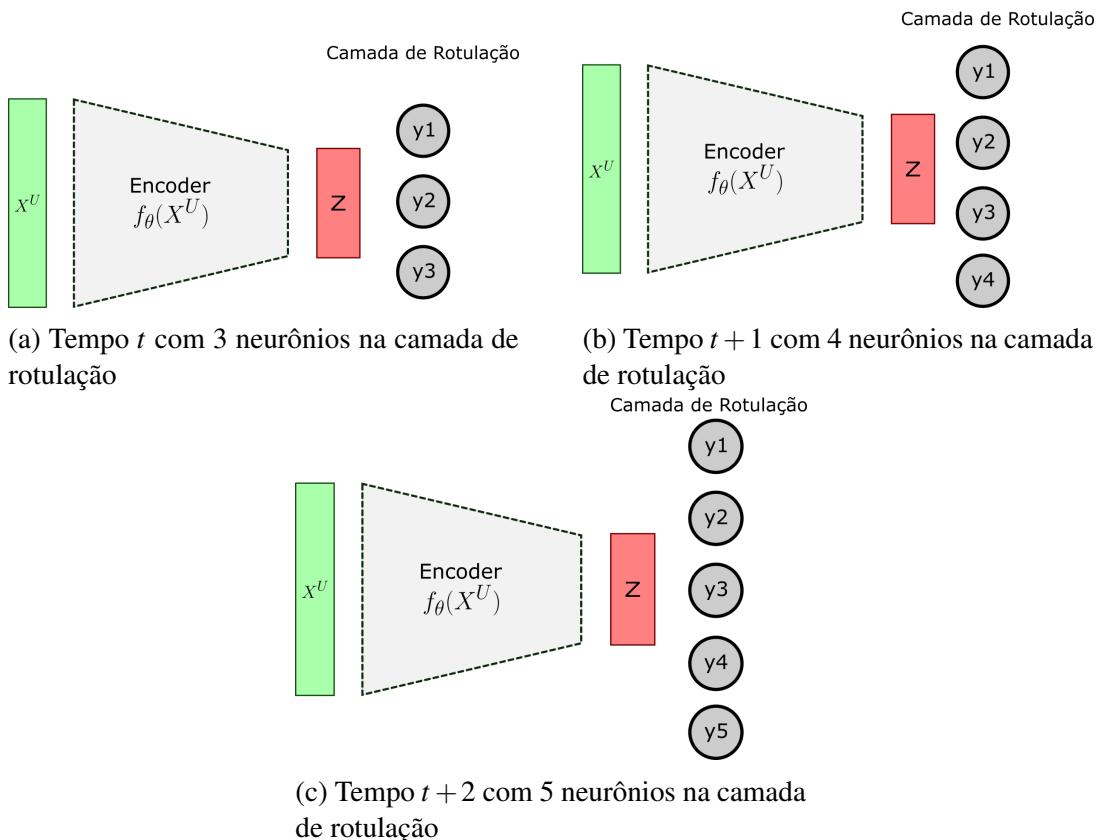


Figura 7.6: Adaptação do DSL em cada um dos tempos

A Figura 7.7 apresenta o procedimento adotado para o DSL para a classificação semissupervisionada dinâmica de dados de stream. Os dados são submetidos no modelo a cada tempo. Caso a amostra analisada pelo DSL não seja rotulada, ela é submetido ao procedimento padrão do DSL, explicado no Capítulo 5, onde o *encoder* reduz a dimensionalidade e a camada de rotulação define uma classe para aquela amostra, posteriormente atualiza-se os pesos da camada de rotulação.

Caso o dado seja rotulado, o DSL tenta definir uma classe para a nova amostra, utilizando as medidas de ITL, é possível analisar caso a amostra não pertença a nenhuma das classes existentes. Caso isso ocorra, cria-se um novo grupo nos dados, e atualiza-se a camada de rotulação adicionando mais um neurônio. Por fim, atualiza-se a camada de rotulação e os pesos por meio do otimizador Adam. Esse processo poderá ocorrer infinitas vezes, pois o fluxo de dados pode nunca parar, o DSL iria treinar e rotular dinamicamente amostras de dados.

Para executar o modelo proposto, adotamos um total de 5 tempos. A cada tempo adicionou-se um parte da base de dados para testar o modelo com diferentes configuração de dados. A Tabela 7.4 apresenta a quantidade amostras em cada um do tempos (t_0, t_1, t_2, t_3 e t_4), e suas respectivas quantidades de classes. Inicialmente, no tempo t_0 , treinamos o modelo com apenas 20% dos dados e 3 classes, depois em cada um dos tempos, aumentamos essa quantidade de dados, como pode ser visto na Tabela 7.4.

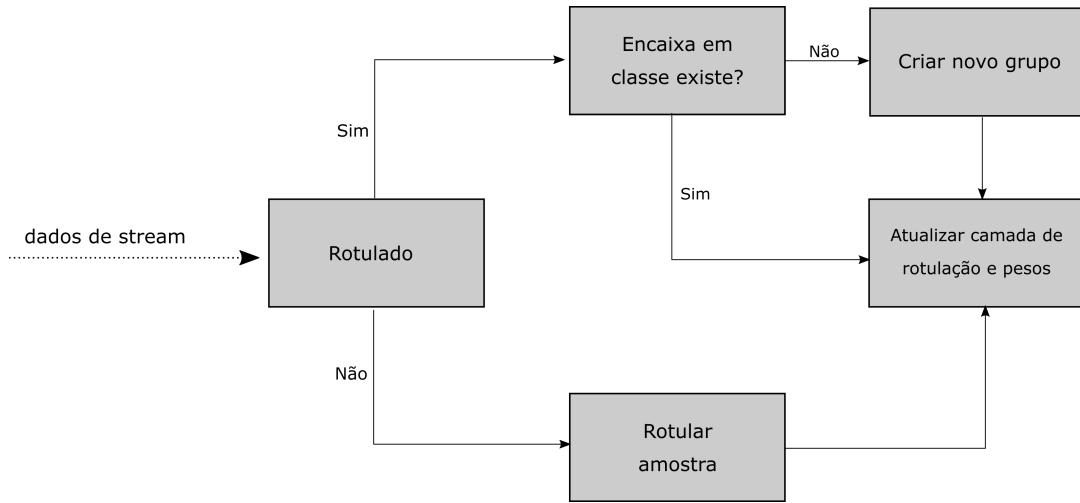


Figura 7.7: Fluxograma de execução do DSL com dados de Stream

Tabela 7.4: Quantidade de dados em cada instante de tempo

	t0	t1	t2	t3	t4
	20%	40%	60%	80%	100%
MNIST	14.000 3 classes	28.000 5 classes	42.000 7 classes	56.000 9 classes	70.000 10 classes
FASHION	14.000 3 classes	28.000 5 classes	42.000 7 classes	56.000 9 classes	70.000 10 classes
CIFAR-10	14.000 3 classes	28.000 5 classes	42.000 7 classes	56.000 9 classes	70.000 10 classes
SLT-10	1.300 3 classes	5.200 5 classes	7.800 7 classes	48.000 9 classes	13.000 10 classes

7.2.2 Resultados Preliminares

Os experimentos foram realizados adotando o método de validação cruzada com 10 *folds*, onde os dados em cada instante t foi dividida em 10 subconjuntos, sendo neste caso, utilizamos um dos *folds* como conjunto rotulado X^L , e os demais como conjunto não rotulado X^U .

A Tabela 7.5 apresenta os resultados da rotulação das bases no tempo t_0 , ou seja, com apenas 20% dos dados iniciais e com 3 classes (Tabela 7.6). Podemos observar, que no caso do tempo t_0 , o modelo proposto rotulou corretamente todas as bases de dado, sendo todas com acurácia acima de 0.9. Nota-se também, a taxa de precisão e *recall* com valores também acima de 0.9, indicando que a rotulação foi bem sucedida quando consideramos classes individualmente. A taxa de *f-score* confirma os resultado, apresentando valores acima de 0.9.

Na Tabela 7.6, podemos ver os resultados da rotulação no tempo t_2 , onde foram adicionados mais classes ao problema, para verificar se o modelo consegue se adaptar à novas classes. Também no tempo t_2 , o modelo DSL apresentou acurácia acima de 0.9. E de

Tabela 7.5: Resultados da rotulação no tempo t_0

BASE	Acurácia	Precisão	Recall	F-score
MNIST	0,952	0,945	0,946	0,954
FASHION	0,915	0,921	0,932	0,935
CIFAR-10	0,942	0,941	0,942	0,941
SLT-10	0,932	0,934	0,933	0,940

Tabela 7.6: Resultado da rotulação no tempo t_1

BASE	Acurácia	Precisão	Recall	F-score
MNIST	0,927	0,927	0,927	0,927
FASHION	0,912	0,912	0,912	0,912
CIFAR-10	0,949	0,949	0,949	0,949
SLT-10	0,934	0,934	0,934	0,934

Tabela 7.7: Resultado da rotulação no tempo t_2

BASE	Acurácia	Precisão	Recall	F-score
MNIST	0,93	0,93	0,93	0,93
FASHION	0,934	0,934	0,934	0,934
CIFAR-10	0,929	0,929	0,929	0,929
SLT-10	0,944	0,944	0,944	0,944

Tabela 7.8: Resultado da rotulação no tempo t_3

BASE	Acurácia	Precisão	Recall	F-score
MNIST	0,93	0,93	0,93	0,93
FASHION	0,934	0,934	0,934	0,934
CIFAR-10	0,929	0,929	0,929	0,929
SLT-10	0,944	0,944	0,944	0,944

Tabela 7.9: Resultado da rotulação no tempo t_4

BASE	Acurácia	Precisão	Recall	F-score
MNIST	0,937	0,937	0,937	0,937
FASHION	0,906	0,906	0,906	0,906
CIFAR-10	0,943	0,943	0,943	0,943
SLT-10	0,904	0,904	0,904	0,904

forma similar, as taxas de *recall* e precisão, assim como a taxa de *f-score*.

Nas Tabelas 7.7, 7.8 e 7.9, os resultados da rotulação nos tempos t_2 , t_3 e t_4 . Assim como nos tempos t_0 e t_1 , o modelo proposto apresentou acurácia média acima 0.9, mesmo adicionando classes no problema. Vemos também, que as taxas de precisão e *recall* não foram afetadas pela a nova quantidade de classes no problema.

7.3 Considerações

Nesse capítulo apresentou-se a aplicação do modelo proposto em um problema de classificação em sensoriamento remoto. A área de estudo foi Região Geográfica Imediata (RGI) de Uruçuí no estado do Piauí. O DSL apresentou desempenho satisfatório para realizar esta tarefa e gerar o mapa de classificação.

Os mapas de classificação, confirmam os bons resultados apresentados nas Tabelas 7.2 e 7.2. Com isso, pode-se afirmar que o DSL consegue desempenhar de forma promissora a tarefa de auxiliar no mapeamento de áreas agrícolas, tornando necessário para o especialista coletar poucas amostras rotuladas, já que coletar amostras não rotuladas é mais fácil.

Além disso, mostrou-se os resultados preliminares da aplicação do DSL em ambiente de *stream*. Colocamos bases de dados da literatura e dividimos os dados em espaços de tempo para treinar o modelo em diferentes situações. O modelo proposto apresentou resultados promissores em todos as situações em que foi aplicado.

A camada de rotulação, adapta-se a quantidade de classe existente no problema a cada instante t , fazendo com que a cada t o modelo consiga rotular e atualizar corretamente os pesos da suas respectivas camadas.

Capítulo 8

Considerações Finais

Esse capítulo apresenta algumas conclusões sobre o trabalho desenvolvido nesta tese, além de apontar algumas perspectiva para trabalhos futuros.

8.1 Conclusões

Como dito no Capítulo 1, o objetivo desse trabalho é propor um modelo de aprendizado semissuperisionado baseado em técnicas de *deep learning* e teoria da informação. Onde foi utilizado um *autoencoder* para gerar representações em um espaço de dimensionalidade menor que os dados originais e realizar o agrupamento dos dados não-rotulados (X^U). O modelo gerado no agrupamento, direciona cada amostra $x \in X^L$ à um grupo gerado. E após isso, realiza-se a rotulação dos elementos de cada grupo, de acordo com uma medida de similaridade, que no nosso caso foi utilizada a entropia cruzada e divergência de *kullback-leibler*. Tais medidas, também foram utilizadas no ajuste fino da camada de rotulação.

Como *autoencoder*, adotou-se no DSL dois tipos, o *autoencoder* convencional e o convolucional *autoencoder*. O primeiro para as bases de *benchmark* (EPILEPSIA, REUTERS, GÁS e PENDIGITS) e o segundo para as bases de imagem (MNIST, FASHION, CIFAR-10, SLT-10). Através dos resultados obtidos, foi possível perceber que o modelo proposto é capaz de realizar a classificação de forma semissupervisionada. O DSL obteve resultado equivalente ou superior a outros modelos da literatura. O modelo foi validado tanto nas bases de *benchmark*, quanto nas bases de imagens.

O modelo DSL, depende do agrupamento que é gerado, pois neste trabalho, assume-se a premissa de que dados de mesma classe tendem a serem agrupadas em mesmo grupo. Utilizando o agrupamento profundo (*deep clustering*), obteve-se melhores agrupamentos, do que utilizamos o algoritmo *k-means*, que utiliza a distância euclidiana. Utilizamos uma variação do algoritmo k-means, no caso o k-means++, apenas para inicializar os centroides, que são os pesos iniciais da camada de rotulação. Com isso, conclui-se que o melhor para o DSL é utilizar agrupamento profundo, devido ao modelo proposto ter objetivos de ser aplicado em problemas de altas dimensionalidades.

A abordagem adotada no DSL, pode ser entendida como um estratégia de dividir para conquistar, onde basicamente os dados não rotulados (X^U) são divididos em grupos e cada grupo é rotulado de forma individual, dependendo de quais dados rotulados foram atribuí-

dos à ele. E ao mesmo tempo, os centroides dos grupos tornam-se os pesos da camada de rotulação, sendo assim, possível classificar amostras não previstas no treinamento, como pode ser visto nos resultados da fase indutiva no Capítulo 6. De todas as bases de dados, a de EPILEPSIA foi que o modelo proposto apresentou desempenho pior. Isso é atribuído ao fato da base ter um maior número de classes.

Diante dos resultados, pode-se concluir que o DSL é um modelo semissupervisionado promissor, que depende de uma avaliação prévia dos hiperparâmetros k e t , onde k , no caso das bases utilizadas, não não foi satisfatório com valores muito próximos de 1 e nem muito próximo de 25. Para o hiperparâmetro t , valores muitos baixo comprometem o desempenho do modelo proposto, pois quanto menor o valo de t , mas amostras de classes distintas podem ter no grupo.

No Capítulo 7, apresentamos também o DSL em uma aplicação no contexto de sensoriamento remoto. Neste trabalho, o objetivo é classificar pixels para gerar mapa de classificação da região do MATOPIBA para auxiliar no mapeamento de atividades agrícola. Podemos notar, que o modelo proposto obteve resultados superiores aos modelos semissupervisionado clássicos. Através dos mapas de classificação, podemos perceber a eficiência do modelo proposto em melhorar a visualização das atividades agrícolas, sendo possível observar nos anos de 2013, 2015 e 2019.

Ainda, no Capítulo 7 foram apresentados resultados preliminares do estudo do modelo proposto no contexto de classificação de dados de *streaming*. Os resultados ainda são iniciais, apenas uma simulação, do modelo proposto, atuando em diferentes instantes de tempo, onde a cada instante t recebe uma nova remessa de dados. Aplicando as bases de imagens, onde a cada instante de tempo, aumentava-se a quantidade de dados e de classes no problema, o DSL apresentou resultados promissores no problema de rotulação de dados de *streaming*.

É válido mencionar, que durante o desenvolvimento do modelo proposto, obteve-se várias críticas e contribuições da comunidade científica. Ao longo da pesquisa, tópicos relacionados ao trabalho foram submetidos à conferências e periódicos científicos, são eles: Lima, Silva, Neto e Machado (2019a), Lima, Silva, Neto e Machado (2019b), Lima, Neto, Silva, Machado e Costa (2019), Lima et al. (2021).

Diante do que foi exposto nesta tese, pode-se concluir que o DSL é capaz de treinar com dados rotulados e não rotulados simultaneamente, rotulando de forma satisfatória o conjunto X^U , na fase transductiva, bem como classificar corretamente amostras não previstas no treinamento na fase indutiva.

8.2 Trabalhos Futuros

Apesar do modelo proposto apresentar desempenho médio satisfatório, ainda existe margem para ampliar os estudos melhorá a técnica. Um dos pontos sugeridos, é a escolha de outro método para selecionar os centroides iniciais, utilizou-se neste trabalho o *k-means++*, poderia utilizar por exemplo a rede de Kohonen.

Utilizou-se duas técnicas como função custo, divergência de *Kullback-Leibler* e entropia cruzada, sugere-se como trabalho futuro, investigar o uso de outros métodos de teoria da informação para ser utilizada como função custo. Outra sugestão de trabalho

futuro, além da função custo, outras formas de verificar a similaridade entre elementos, no momento de calcular o vetor de influência das amostras. Sugere-se também, adaptar o modelo proposto para o agrupamento semissupervisionado, otimizando o encoder + camada de rotulação, com diferentes medidas de ITL, considerando dados rotulados para auxiliar no agrupamento.

E por fim, como trabalho futuro, sugere-se ampliar o estudo do modelo proposto no contexto de classificação de dados de *streaming*. Como vimos no Capítulo 7, os resultados preliminares sugerem que o DSL pode ser promissor nessa área.

8.3 Publicações

Esse trabalho originou as seguintes publicações científicas:

- LIMA, B. V. A.; DORIA NETO, A. D. ; SILVA, L. E. S. ; MACHADO, V. P.. **Deep Semi-supervised Classification based in Deep Clustering and Cross Entropy**. INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, v. 36, p. 1-40, 2021, DOI: <https://doi.org/10.1002/int.22446>.
- LIMA, BRUNO VICENTE ALVES; NETO, ADRIÃO DUARTE DÓRIA ; SILVA, LÚCIA EMÍLIA SOARES ; MACHADO, VINICIUS PONTE . Deep semi supervised classification based in deep clustering and cross entropy. INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, v. n/a, p. 1-40, 2021.
- LIMA, B. V. A.; DORIA NETO, A. D. ; SILVA, L. E. S. ; MACHADO, V. P. ; COSTA, J. G. C.. **Semi-Supervised Approach Using Nearest Neighbors Clustering and Deep Learning**. In: 14º Simpósio Brasileiro de Automação Inteligente, 2019, Ouro Preto - MG. Anais do 14º Simpósio Brasileiro de Automação Inteligente, 2019. v. 1.
- LIMA, B. V. A.; DORIA NETO, A. D. ; SILVA, L. E. S ; MACHADO, V. P. ; COSTA, J. G. C.. **Semi-supervised Classification Using Deep Learning**. In: 2019 8th Brazilian Conference on Intelligent Systems (BRACIS), 2019, Salvador.
- LIMA, B. V. A.; DORIA NETO, A. D. ; SILVA, L. E. S. ; MACHADO, V. P.. **Abordagem Semissuperisionada usando Deep Learning Aplicada à Rotulação e Classificação de Dados**. In: Computer on the Beach, 2019, Florianópolis. Anais do Computer on the Beach, 2019. p. 356.

Referências Bibliográficas

- Aggarwal, Charu C. (2018), *Neural Networks and Deep Learning: A Textbook*, 1st^a edição, Springer Publishing Company, Incorporated.
- Alelyani, S, J Tang e H Liu (2013), ‘Feature selection for clustering: A review. data clustering: Algorithms and applications, editor: Charu aggarwal and chandan reddy’.
- Amini, Massih-Reza e Patrick Gallinari (2003), Semi-supervised learning with explicit misclassification modeling, *em* ‘International Joint Conference on Artificial Intelligence’, IJCAI’03, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 555–560.
- Arthur, David e Sergei Vassilvitskii (2006), k-means++: The advantages of careful seeding, Relatório técnico, Stanford.
- Aydag, Prem Shankar Singh e Sonajharia Minz (2019), Self-training with neighborhood information for the classification of remote sensing images, *em* ‘Information and Communication Technology for Competitive Strategies’, Springer, pp. 465–474.
- Bache, K. e M. Lichman (2013), ‘(uci) machine learning repository’.
URL: <http://archive.ics.uci.edu/ml>
- Baldi, Pierre (2012), Autoencoders, unsupervised learning, and deep architectures, *em* ‘Proceedings of ICML workshop on unsupervised and transfer learning’, JMLR Workshop and Conference Proceedings, pp. 37–49.
- Barber, David (2012), *Bayesian Reasoning and Machine Learning*, Cambridge University Press, New York, NY, USA.
- Basu, Sugato, Arindam Banerjee e Raymond Mooney (2002), ‘Semi-supervised clustering by seeding’, *International Conference on Machine Learning* pp. 19–26.
- Bergstra, James e Yoshua Bengio (2012), ‘Random search for hyper-parameter optimization.’, *Journal of machine learning research* 13(2).
- Blum, Avrim e Tom Mitchell (1998), Combining labeled and unlabeled data with co-training, *em* ‘Proceedings of the eleventh annual conference on Computational learning theory’, pp. 92–100.

- Chimieski, Bruno Fernandes e Rubem Dutra Ribeiro Fagundes (2013), ‘Association and classification data mining algorithms comparicion over medical datasets’, *Journal of health informatics* **5**, 44–51.
- Chollet, Francois et al. (2015), ‘Keras’.
URL: <https://github.com/fchollet/keras>
- Colditz, R.R., M. Schmidt, C. Conrad, M.C. Hansen e S. Dech (2011), ‘Land cover classification with coarse spatial resolution data to derive continuous and discrete maps for complex regions’, *Remote Sensing of Environment* **115**(12), 3264 – 3275.
URL: <http://www.sciencedirect.com/science/article/pii/S0034425711002549>
- Conover, William Jay (1998), *Practical nonparametric statistics*, Vol. 350, John Wiley & Sons.
- Conway, Drew e John Myles White (2012), *Machine learning for hackers*, O'Reilly Media.
- Cover, Thomas M e Joy A Thomas (2006), ‘Elements of information theory second edition solutions to problems’, *Internet Access* .
- da Silva, Ivan Nunes, Danilo Hernane Spatti e Rogério Andrade Flauzino (2010), *Redes neurais artificiais: para engenharia e ciências aplicadas*, ARTLIBER.
- Deng, Jia, Wei Dong, Richard Socher, Li jia Li, Kai Li e Li Fei-fei (2009), Imagenet: A large-scale hierarchical image database, *em ‘In CVPR’*.
- Din, Salah Ud, Junming Shao, Jay Kumar, Waqar Ali, Jiaming Liu e Yu Ye (2020), ‘Online reliable semi-supervised learning on evolving data streams’, *Information Sciences* **525**, 153–171.
- Do-Omri, Alan, Dalei Wu e Xiaohua Liu (2018), ‘A self-training method for semi-supervised gans’.
- E. Hinton, Geoffrey e James McClelland (1987), Learning representations by recirculation., pp. 358–366.
- Enguehard, Joseph, Peter O'Halloran e Ali Gholipour (2019), ‘Semi-supervised learning with deep embedded clustering for image classification and segmentation’, *IEEE Access* **7**, 11093–11104.
- Faceli, Katti, Ana Carolina Lorena, Joao Gama e Andre C. P. L. F. de Carvalho (2011), *Inteligência Artificial: Uma Abordagem de Aprendizado de Máquina*, Rio de Janeiro.
- Feranec, Jan, Gabriel Jaffrain, Tomas Soukup e Gerard Hazeu (2010), ‘Determining changes and flows in european landscapes 1990–2000 using corine land cover data’, *Applied Geography* **30**(1), 19 – 35.
URL: <http://www.sciencedirect.com/science/article/pii/S0143622809000472>

- Fu, Sichao, Weifeng Liu, Weili Guan, Yicong Zhou, Dapeng Tao e Changsheng Xu (2021), ‘Dynamic graph learning convolutional networks for semi-supervised classification’, *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* **17**(1s), 1–13.
- Galli, Filippo (2016), ‘Landmark recognition with deep learning’.
- Gantz, John e David Reinsel (2012), ‘The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east’.
- Goodfellow, Ian, Yoshua Bengio e Aaron Courville (2016), *Deep Learning*, MIT Press. <http://www.deeplearningbook.org>.
- Gupta, Hriday Kumar e Rafat Parveen (2019), Comparative study of big data frameworks, *em ‘2019 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)’*, Vol. 1, IEEE, pp. 1–4.
- Han, Jiawei (2005), *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- Hartley, Ralph VL (1928), ‘Transmission of information 1’, *Bell System technical journal* **7**(3), 535–563.
- Haykin, Simon (2001), *Neural Networks: A Comprehensive Foundation*, 2^a edição, Porto Alegre.
- Hebb, Donald Olding (1949), ‘The organization of behavior; a neuropsychological theory’, *A Wiley Book in Clinical Psychology* **62**, 78.
- Hinton, Geoffrey E., Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever e Ruslan Salakhutdinov (2012), ‘Improving neural networks by preventing co-adaptation of feature detectors’, *CoRR abs/1207.0580*.
- Hinton, Geoffrey E e Ruslan R Salakhutdinov (2006), ‘Reducing the dimensionality of data with neural networks’, *science* **313**(5786), 504–507.
- Huang, B. e K. Deng (2019), Semi-supervised sparse representation classification with insufficient samples, *em ‘2019 6th International Conference on Information Science and Control Engineering (ICISCE)’*, pp. 545–549.
- Iscen, Ahmet, Giorgos Tolias, Yannis Avrithis e Ondrej Chum (2019), Label propagation for deep semi-supervised learning, *em ‘Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition’*, pp. 5070–5079.
- Jain, Anil K (2010), ‘Data clustering: 50 years beyond k-means’, *Pattern recognition letters* **31**(8), 651–666.
- Junior, Oswaldo Ludwig e Eduard Montgomery Meira Costa (2007), *Redes Neurais: Fundamentos e Aplicações com Programas em C*, Rio de Janeiro.

- Kingma, Diederik P., Danilo Jimenez Rezende, Shakir Mohamed e Max Welling (2014), ‘Semi-supervised learning with deep generative models’, *CoRR abs/1406.5298*.
- Kingma, Diederik P e Jimmy Ba (2014), ‘Adam: A method for stochastic optimization’, *arXiv preprint arXiv:1412.6980* .
- Koda, Satoru, Farid Melgani e Ryuei Nishii (2019), ‘Unsupervised spectral-spatial feature extraction with generalized autoencoder for hyperspectral imagery’, *IEEE Geoscience and Remote Sensing Letters* .
- Kothari, Neeta S e Saroj K Meher (2020), ‘Semisupervised classification of remote sensing images using efficient neighborhood learning method’, *Engineering Applications of Artificial Intelligence* **90**, 103520.
- Kuhn, Harold W (1955), ‘The hungarian method for the assignment problem’, *Naval research logistics quarterly* **2**(1-2), 83–97.
- Kullback, Solomon e Richard A Leibler (1951), ‘On information and sufficiency’, *The annals of mathematical statistics* **22**(1), 79–86.
- Le Nguyen, Minh Huong, Heitor Murilo Gomes e Albert Bifet (2019), Semi-supervised learning over streaming data using moa, em ‘2019 IEEE International Conference on Big Data (Big Data)’, IEEE, pp. 553–562.
- LeCun, Yann, Léon Bottou, Yoshua Bengio e Patrick Haffner (1998), ‘Gradient-based learning applied to document recognition’, *Proceedings of the IEEE* **86**(11), 2278–2324.
- LeCun, Yann, Yoshua Bengio e Geoffrey Hinton (2015), ‘Deep learning’, *nature* **521**(7553), 436.
- LeCun, Yann et al. (1989), ‘Generalization and network design strategies’, *Connectionism in perspective* **19**, 143–155.
- Lee, Hye-Woo, Noo-ri Kim e Jee-Hyong Lee (2017), ‘Deep neural network self-training based on unsupervised learning and dropout’, *International Journal of Fuzzy Logic and Intelligent Systems* **17**(1), 1–9.
- Lewis, D. (1997), Reuters-21578 text categorization test collection, distribution 1.0.
- Li, Junnan, Qingsheng Zhu e Quanwang Wu (2019), ‘A self-training method based on density peaks and an extended parameter-free local noise filter for k nearest neighbor’, *Knowledge-Based Systems* **184**, 104895.
- Li, Junnan, Qingsheng Zhu, Quanwang Wu e Dongdong Cheng (2020), ‘An effective framework based on local cores for self-labeled semi-supervised classification’, *Knowledge-Based Systems* p. 105804.
- Li, Ming e Zhi-Hua Zhou (2005), Setred: Self-training with editing, em ‘PAKDD’.

- Li, Y. e H. Yeh (2019), A semi-supervised learning model based on convolutional auto-encoder and convolutional neural network for image classification, *em ‘2019 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)’*, pp. 1–2.
- Li, Yanchao, Yongli Wang, Qi Liu, Cheng Bi, Xiaohui Jiang e Shurong Sun (2019), ‘Incremental semi-supervised learning on streaming data’, *Pattern Recognition* **88**, 383–396.
- Lima, Bruno Vicente Alves (2015), Método semissupervisionado de rotulação e classificação utilizando agrupamento por sementes e classificadores, Dissertação de mestrado, Universidade Federal do Piauí.
- Lima, Bruno Vicente Alves, Adrião Duarte Dória Neto, Lúcia Emilia Soares Silva, Víncius Ponte Machado e João Guilherme Cavalcanti Costa (2019), Semi-supervised classification using deep learning, *em ‘2019 8th Brazilian Conference on Intelligent Systems (BRACIS)’*, IEEE, pp. 717–722.
- Lima, Bruno Vicente Alves, Adrião Duarte Dória Neto, Lúcia Emilia Soares Silva e Víncius Ponte Machado (2021), ‘Deep semi-supervised classification based in deep clustering and cross-entropy’, *International Journal of Intelligent Systems* pp. 1–0.
- Lima, Bruno Vicente Alves, Lucia Emilia Soares Silva, Adrião Duarte Doria Neto e Víncius Ponte Machado (2019a), ‘Abordagem semissupervisionada usando deep learning aplicada à rotulação e classificação de dados’, *Anais do Computer on the Beach* pp. 356–365.
- Lima, Bruno Vicente Alves, Lucia Emilia Soares Silva, Adrião Duarte Doria Neto e Víncius Ponte Machado (2019b), ‘Semi-supervised approach using nearest neighbors clustering and deep learning’, *Anais do 14 Simpósio Brasileiro de Automação Inteligente* pp. 1676–1681.
- Lin, Guangfeng, Xiaobing Kang, Kaiyang Liao, Fan Zhao e Yajun Chen (2020), ‘Deep graph learning for semi-supervised classification’, *arXiv preprint arXiv:2005.14403*.
- Liu, N., L. Yao e X. Zhao (2020), A semi-supervised classification approach based on restricted boltzmann machine for fmri data, *em ‘2020 8th International Winter Conference on Brain-Computer Interface (BCI)’*, pp. 1–4.
- Livieris, Ioannis (2019), ‘A new ensemble semi-supervised self-labeled algorithm’, *Informatica* **43**(2), 221–234.
- Ma, Jian, Hua Su, Wan-lin Zhao e Bin Liu (2018), ‘Predicting the remaining useful life of an aircraft engine using a stacked sparse autoencoder with multilayer self-learning’, *Complexity* **2018**.

- MacQueen, J. (1967), *Some methods for classification and analysis of multivariate observations*, Vol. 1, University of California Press.
- Maggu, Jyoti, Angshul Majumdar, Emilie Chouzenoux e Giovanni Chierchia (2020), ‘Deeply transformed subspace clustering’, *Signal Processing* p. 107628.
- Manning, Christopher D., Prabhakar Raghavan e Hinrich Schütze (2008), *Introduction to Information Retrieval*, Cambridge University Press, New York, NY, USA.
- Masci, Jonathan, Ueli Meier, Dan Cireşan e Jürgen Schmidhuber (2011), Stacked convolutional auto-encoders for hierarchical feature extraction, em ‘International conference on artificial neural networks’, Springer, pp. 52–59.
- McCulloch, Warren S e Walter Pitts (1943), ‘A logical calculus of the ideas immanent in nervous activity’, *The bulletin of mathematical biophysics* 5(4), 115–133.
- McCulloch, Warren S. e Walter Pitts (1988), ‘Neurocomputing: Foundations of research’, pp. 15–27.
- Mitchell, Tom Michael (1997), ‘Machine learning’.
- Murphy, Kevin P (2012), *Machine learning: a probabilistic perspective*, MIT press.
- Ouali, Yassine, Céline Hudelot e Myriam Tami (2020), ‘An overview of deep semi-supervised learning’.
- Parzen, Emanuel (1962), ‘On estimation of a probability density function and mode’, *The annals of mathematical statistics* 33(3), 1065–1076.
- Patterson, Josh e Adam Gibson (2017), *Deep learning: A practitioner’s approach*, "O'Reilly Media, Inc.".
- Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot e E. Duchesnay (2011), ‘Scikit-learn: Machine learning in Python’, *Journal of Machine Learning Research* 12, 2825–2830.
- Peng, Xi, Jiashi Feng, Joey Tianyi Zhou, Yingjie Lei e Shuicheng Yan (2020), ‘Deep subspace clustering’, *IEEE Transactions on Neural Networks and Learning Systems*
- .
- Pilli, Narayana (2019), ‘Difference between machine learning and deep learning’.
- URL:** <https://morioh.com/p/ed56b4fdbf1c>
- Piroonsup, N e Sukree Sinthupinyo (2018), ‘Analysis of training data using clustering to improve semi-supervised self-training’, 143, 65–80.
- Poliana Reis, Savaram Ravindra (2017), ‘Como as redes neurais convolucionais realizam o reconhecimento de imagem’.
- URL:** <https://www.infoq.com.br/articles/redes-neurais-convolucionais/>

- Ponti, Moacir Antonelli e Gabriel B. Paranhos da Costa (2018), ‘Como funciona o deep learning’, *CoRR abs/1806.07908*.
- Powers, David M. W. (2007), Evaluation: From precision, recall and f-factor to roc, informedness, markedness & correlation, Relatório técnico, School of Informatics and Engineering, Flinders University, Adelaide, Australia.
- Principe, Jose C. (2010), *Information Theoretic Learning: Renyi's Entropy and Kernel Perspectives*, 1st^a edição, Springer Publishing Company, Incorporated.
- Ren, Yazhou, Kangrong Hu, Xinyi Dai, Lili Pan, Steven C.H. Hoi e Zenglin Xu (2019), ‘Semi-supervised deep embedded clustering’, *Neurocomputing* **325**, 121 – 130.
- Rezende, Solange Oliveira (2003), *Sistemas inteligentes: fundamentos e aplicações*, Manole.
- Rumelhart, David E, Geoffrey E Hinton e Ronald J Williams (1986), ‘Learning representations by back-propagating errors’, *nature* **323**(6088), 533–536.
- Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg e Li Fei-Fei (2014), ‘Imagenet large scale visual recognition challenge’.
- Sahito, Attaullah, Eibe Frank e Bernhard Pfahringer (2019), Semi-supervised learning using siamese networks, em ‘Australasian Joint Conference on Artificial Intelligence’, Springer, pp. 586–597.
- Salimans, Tim, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford e Xi Chen (2016), Improved techniques for training gans, em ‘Advances in neural information processing systems’, pp. 2234–2242.
- Schwenker, Friedhelm e Edmondo Trentin (2014), ‘Pattern classification and clustering: A review of partially supervised learning approaches’, *Pattern Recognition Letters* **37**, 4–14.
- Shannon, Claude E (1948), ‘A mathematical theory of communication’, *The Bell system technical journal* **27**(3), 379–423.
- Silver, David, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel e Demis Hassabis (2017), ‘Mastering the game of go without human knowledge’, *Nature* **550**, 354–.
- Śmiejka, Marek, Łukasz Struski e Mário AT Figueiredo (2020), ‘A classification-based approach to semi-supervised clustering with pairwise constraints’, *Neural Networks* .

Sokolova, Marina e Guy Lapalme (2009), ‘A systematic analysis of performance measures for classification tasks’, *Information Processing & Management* **45**(4), 427 – 437.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever e Ruslan Salakhutdinov (2014), ‘Dropout: a simple way to prevent neural networks from overfitting’, *The journal of machine learning research* **15**(1), 1929–1958.

Sun, Chen, Abhinav Shrivastava, Saurabh Singh e Abhinav Gupta (2017), ‘Revisiting unreasonable effectiveness of data in deep learning era’, *CoRR abs/1707.02968*.

URL: <http://arxiv.org/abs/1707.02968>

Tan, Pang-Ning, Michael Steinbach, Anuj Karpatne e Vipin Kumar (2019), *Introduction to Data Mining*, Pearson.

Tanha, Jafar, Maarten van Someren e Hamideh Afsarmanesh (2017), ‘Semi-supervised self-training for decision tree classifiers’, *International Journal of Machine Learning and Cybernetics* **8**(1), 355–370.

Vincent, Pascal, Hugo Larochelle, Yoshua Bengio e Pierre-Antoine Manzagol (2008), Extracting and composing robust features with denoising autoencoders, *em ‘Proceedings of the 25th international conference on Machine learning’*, pp. 1096–1103.

Wu, Yue, Guifeng Mu, Can Qin, Qiguang Miao, Wenping Ma e Xiangrong Zhang (2020), ‘Semi-supervised hyperspectral image classification via spatial-regulated self-training’, *Remote Sensing* **12**(1), 159.

Xie, Junyuan, Ross Girshick e Ali Farhadi (2016), Unsupervised deep embedding for clustering analysis, *em ‘Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48’*, ICML’16, JMLR.org, pp. 478–487.

Xu, Rui e Don Wunsch (2009), *Clustering*, Wiley-IEEE Press.

Yang, Yi, Dong Xu, Feiping Nie, Shuicheng Yan e Yuetong Zhuang (2010), ‘Image clustering using local discriminant models and global integration’, *IEEE Transactions on Image Processing* **19**(10), 2761–2773.

Yao, Xiwen, Liuqing Yang, Gong Cheng, Junwei Han e Lei Guo (2019), Scene classification of high resolution remote sensing images via self-paced deep learning, *em ‘IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium’*, IEEE, pp. 521–524.

Zhao, Rui, Ruqiang Yan, Zhenghua Chen, Kezhi Mao, Peng Wang e Robert X Gao (2019), ‘Deep learning and its applications to machine health monitoring’, *Mechanical Systems and Signal Processing* **115**, 213–237.

- Zhou, Shaoguang, Zhaojun Xue e Peijun Du (2019), ‘Semisupervised stacked autoencoder with cotraining for hyperspectral image classification’, *IEEE Transactions on Geoscience and Remote Sensing* **57**(6), 3813–3826.
- Zhou, Yi-Tong e Rama Chellappa (1988), Computation of optical flow using a neural network., *em ‘ICNN’*, pp. 71–78.
- Zhou, Zhi-Hua (2016), ‘Learnware: on the future of machine learning.’, *Frontiers Comput. Sci.* **10**(4), 589–590.
- Zhou, Zhi-Hua e Ming Li (2005), ‘Tri-training: Exploiting unlabeled data using three classifiers’, *IEEE Transactions on knowledge and Data Engineering* **17**(11), 1529–1541.
- Zhu, Qiqi, Xiongli Sun, Yanfei Zhong e Liangpei Zhang (2019), High-resolution remote sensing image scene understanding: A review, *em ‘IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium’*, IEEE, pp. 3061–3064.
- Zhu, Xiaojin (2005), Semi-supervised learning literature survey, Relatório Técnico 1530, Computer Sciences, University of Wisconsin-Madison.
- Zhu, Xiaojin e Andrew B Goldberg (2009), ‘Introduction to semi-supervised learning’, *Synthesis lectures on artificial intelligence and machine learning* **3**(1), 1–130.
- Zhu, Xiaojin e Zoubin Ghahramani (2002), ‘Learning from labeled and unlabeled data with label propagation’.
- Zhu, Yong-Nan e Yu-Feng Li (2020), Semi-supervised streaming learning with emerging new labels, *em ‘Proceedings of the AAAI Conference on Artificial Intelligence’*, Vol. 34, pp. 7015–7022.

Apêndice A

Resultados modelos clássicos fase transdutiva

Os resultados apresentados no Capítulo 6 foram considerando o SVM como classificador base para o modelo M1. Este apêndice apresenta os resultados dos modelos da literatura utilizando MLP e RF como classificador base. Todas as figuras deste apêndice são apenas os resultados para todas bases de dados para o modelo M1 na fase transdutiva.

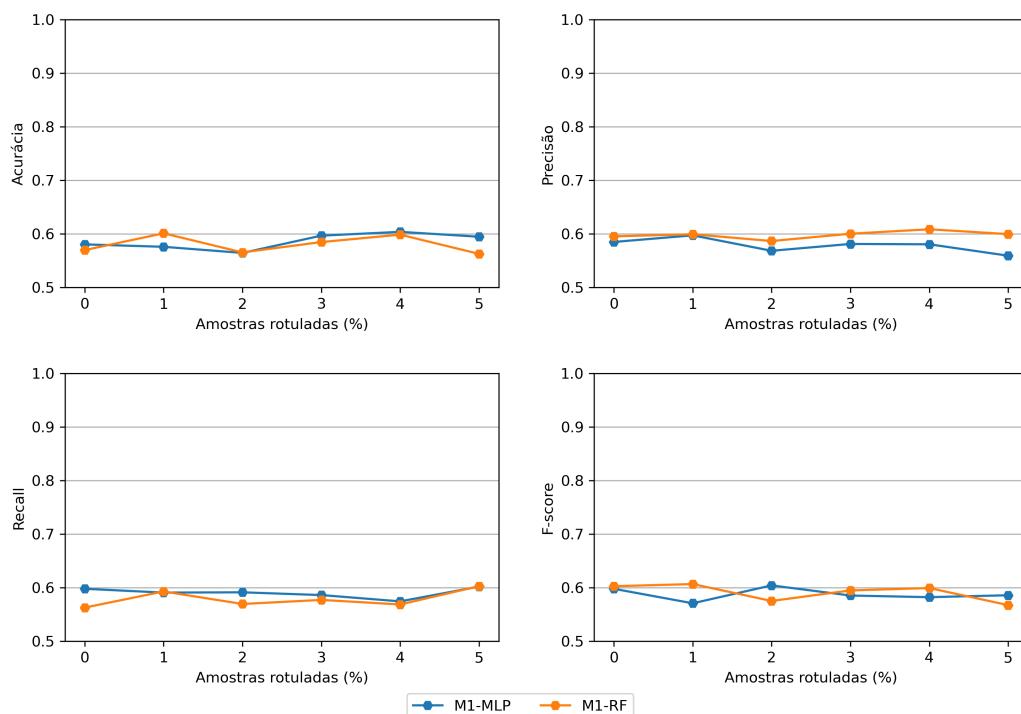


Figura A.1: Rotulação da base EPILEPSIA na fase transdutiva.

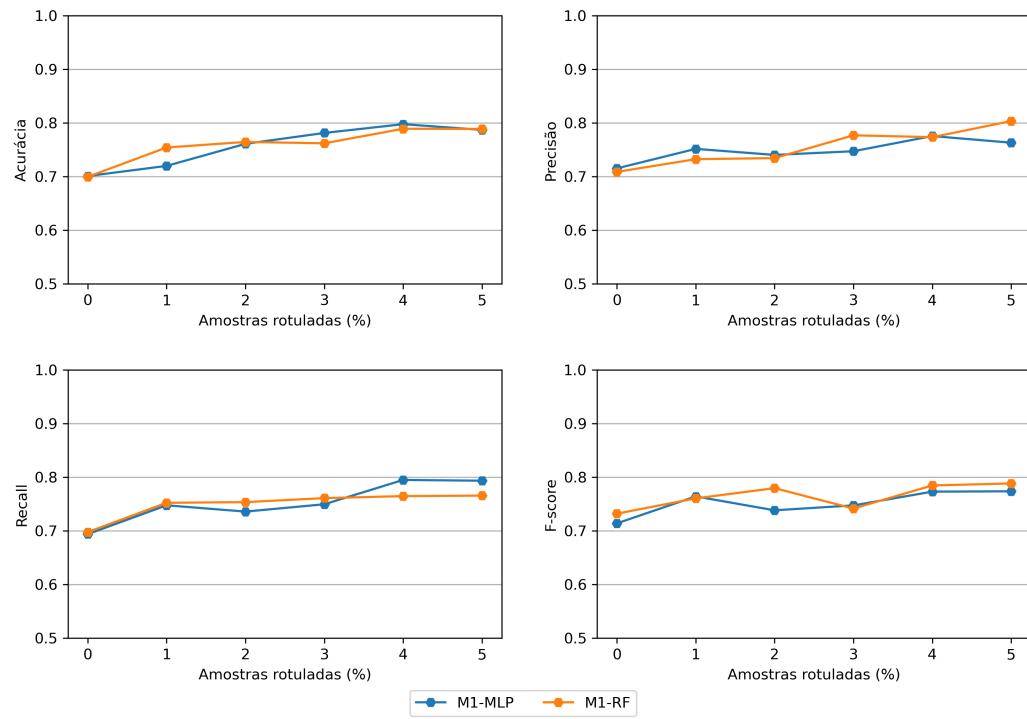


Figura A.2: Rotulação da base REUTERS na fase transdutiva.

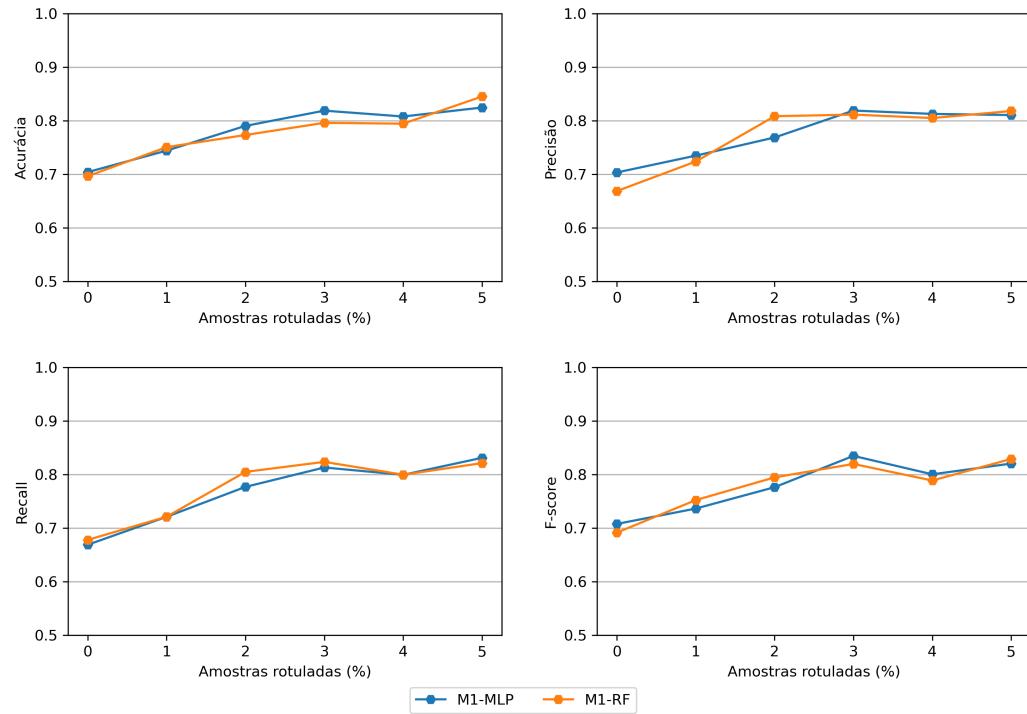


Figura A.3: Rotulação da base GAS na fase transdutiva.

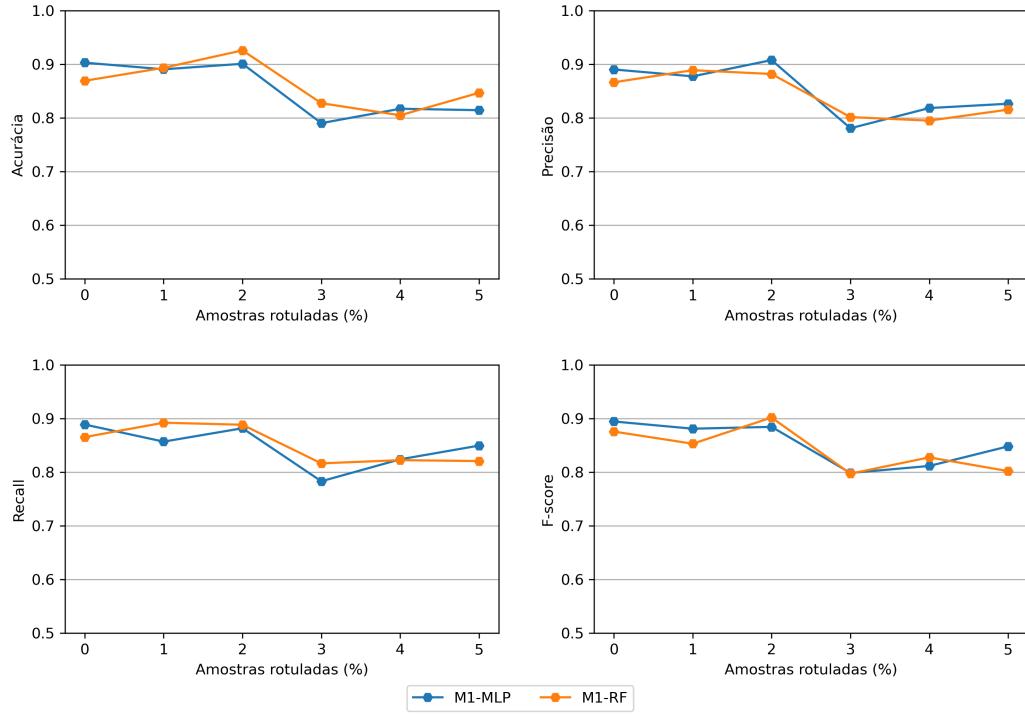


Figura A.4: Rotulação da base PENDIGITS na fase transdutiva.

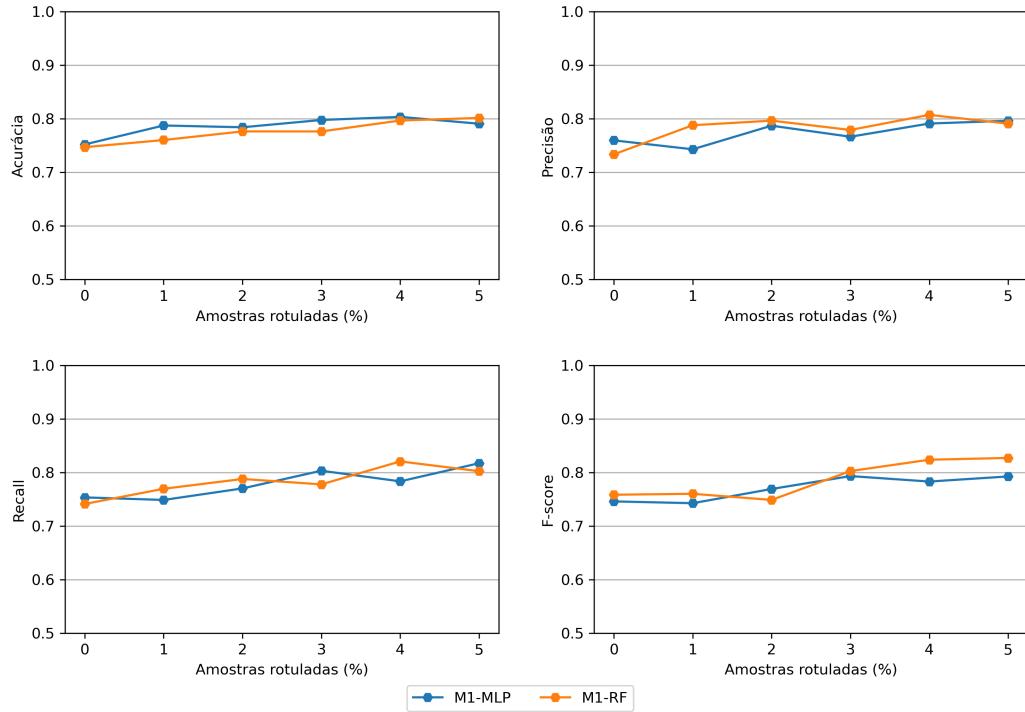


Figura A.5: Rotulação da base MNIST na fase transdutiva.

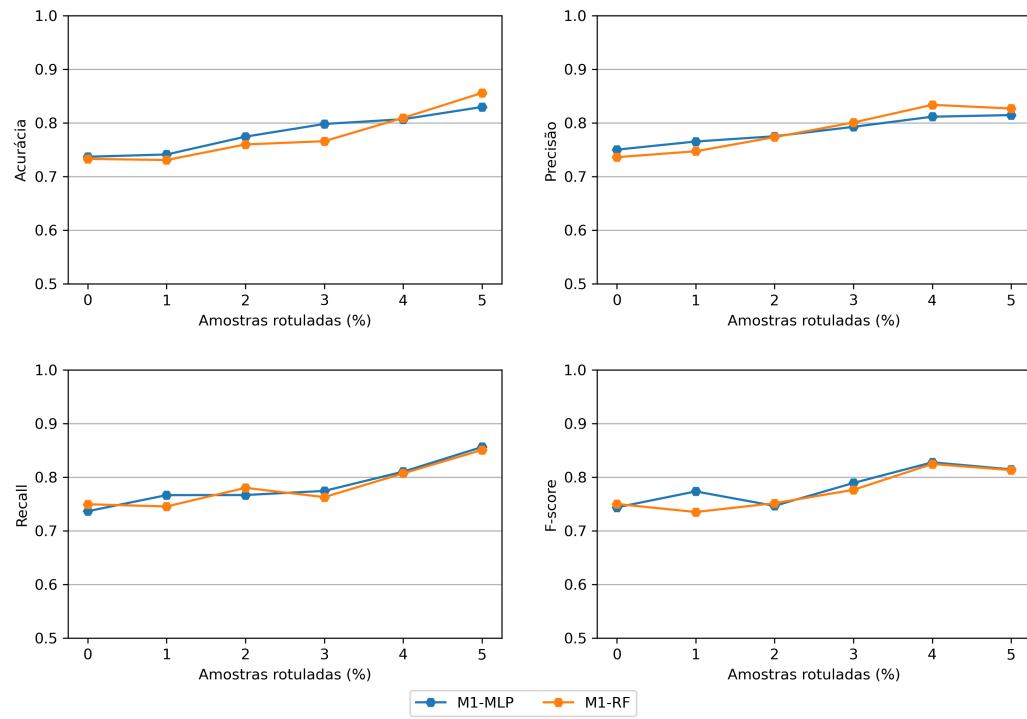


Figura A.6: Rotulação da base FASHION na fase transdutiva.

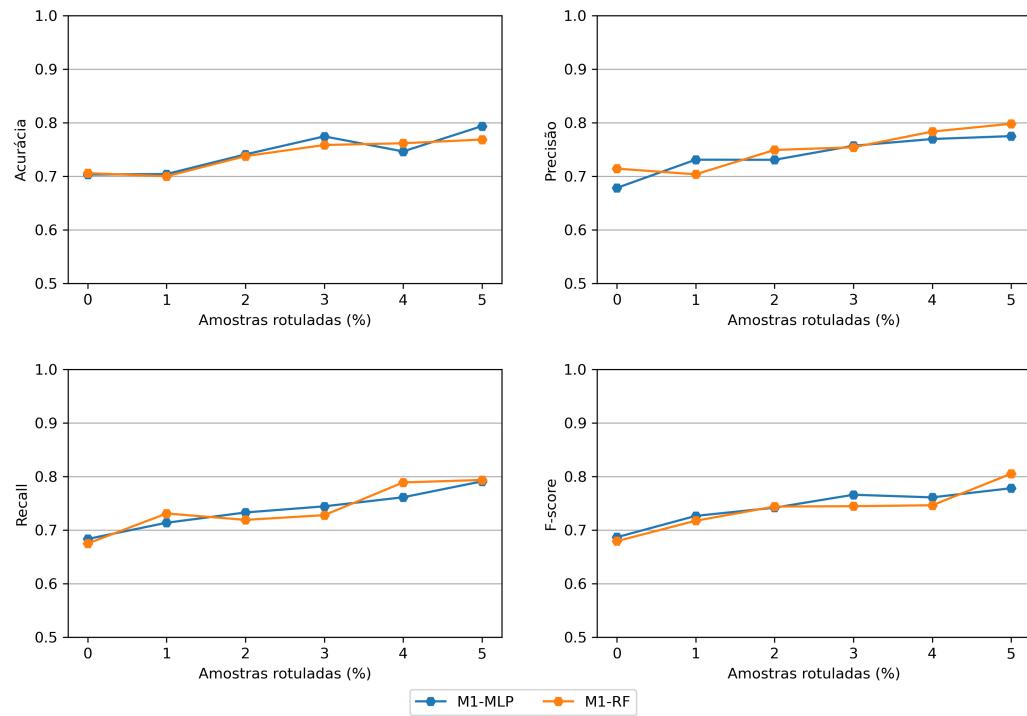


Figura A.7: Rotulação da base CIFAR-10 na fase transdutiva.

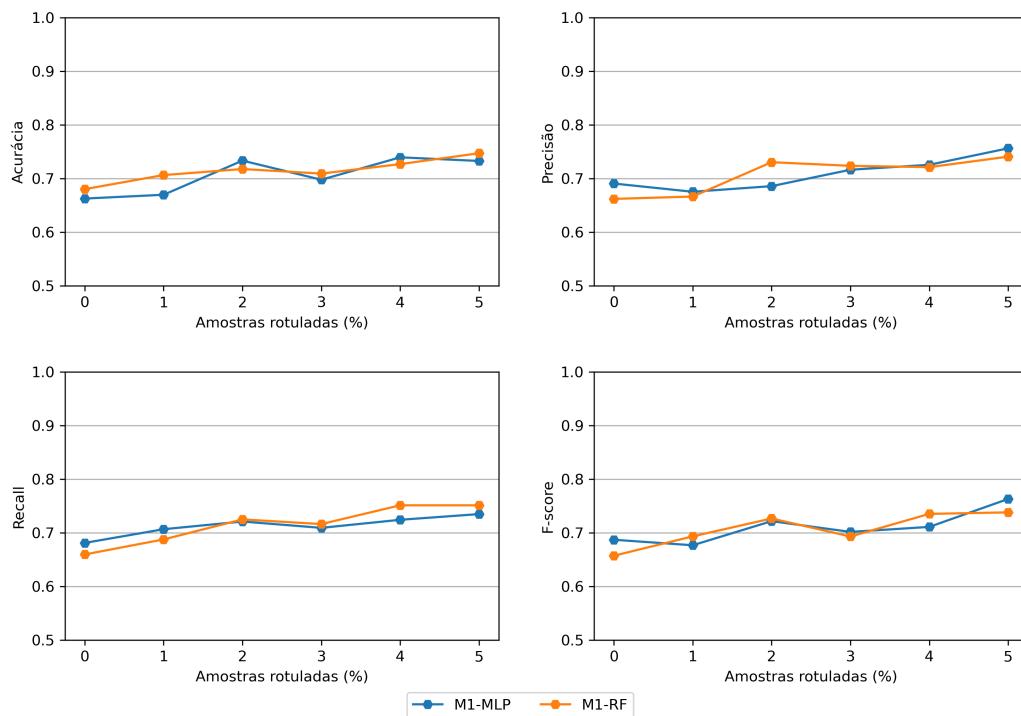


Figura A.8: Rotulação da base SLT-10 na fase transdutiva.

Apêndice B

Resultados modelos clássicos fase indutiva

Os resultados apresentados no Capítulo 6 foram considerando o SVM como classificador base para o modelo M1. Este apêndice apresenta os resultados dos modelos da literatura utilizando MLP e RF como classificador base. Todas as figuras deste apêndice são apenas os resultados para todas bases de dados para o modelo M1 na fase indutiva.

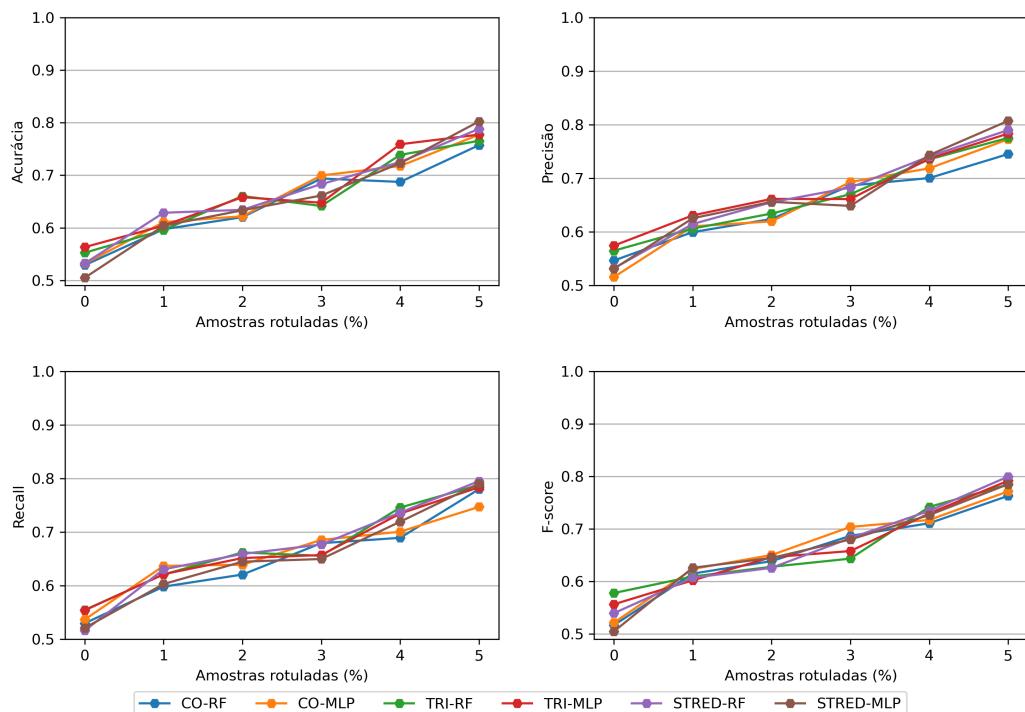


Figura B.1: Rotulação da base EPILEPSIA na fase Indutiva.

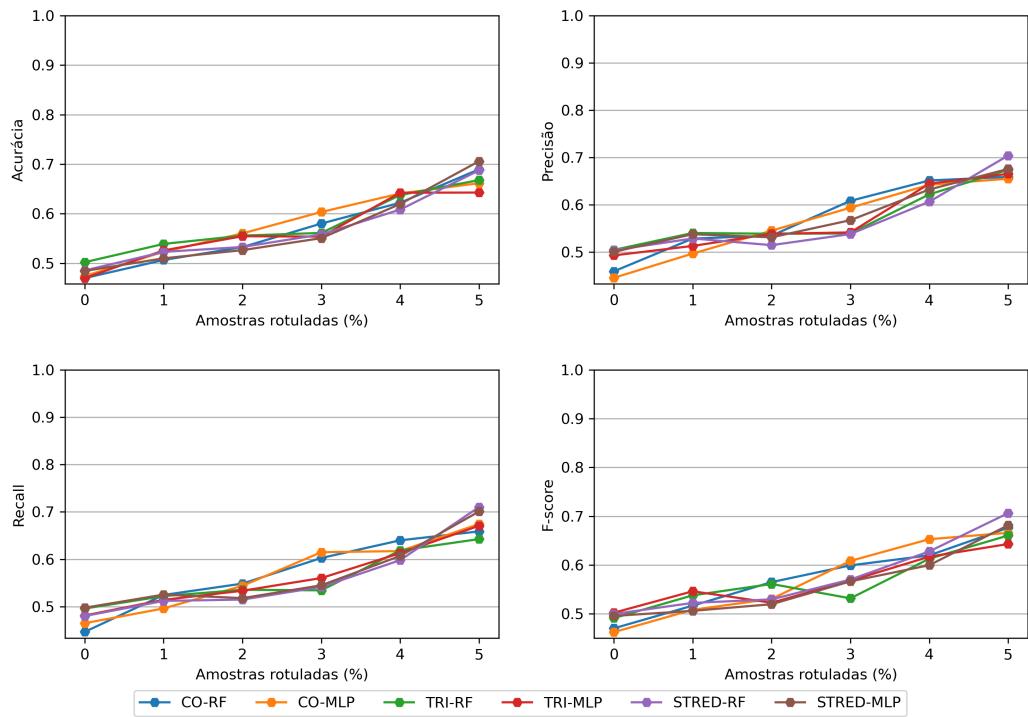


Figura B.2: Rotulação da base REUTERS na fase Indutiva.

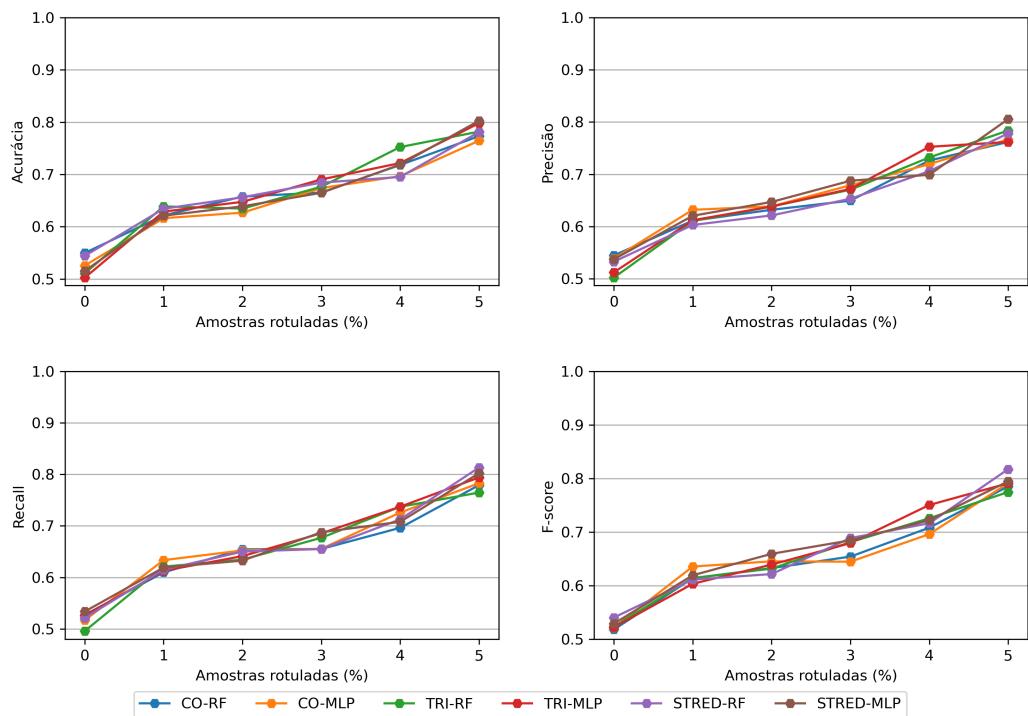


Figura B.3: Rotulação da base GAS na fase Indutiva.

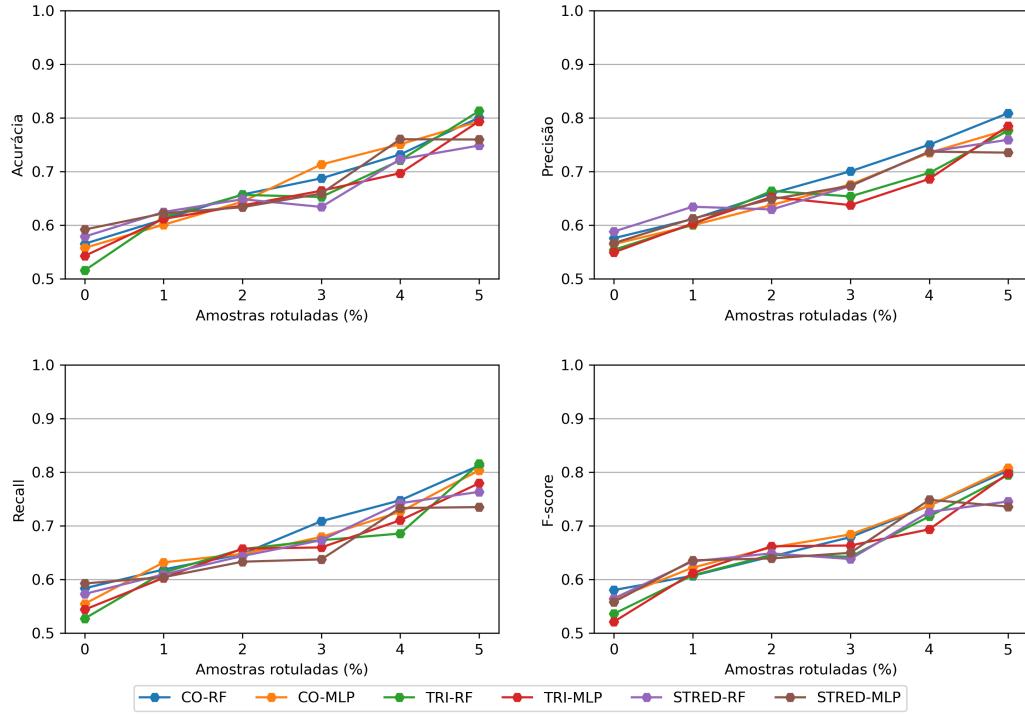


Figura B.4: Rotulação da base PENDIGITS na fase Indutiva.

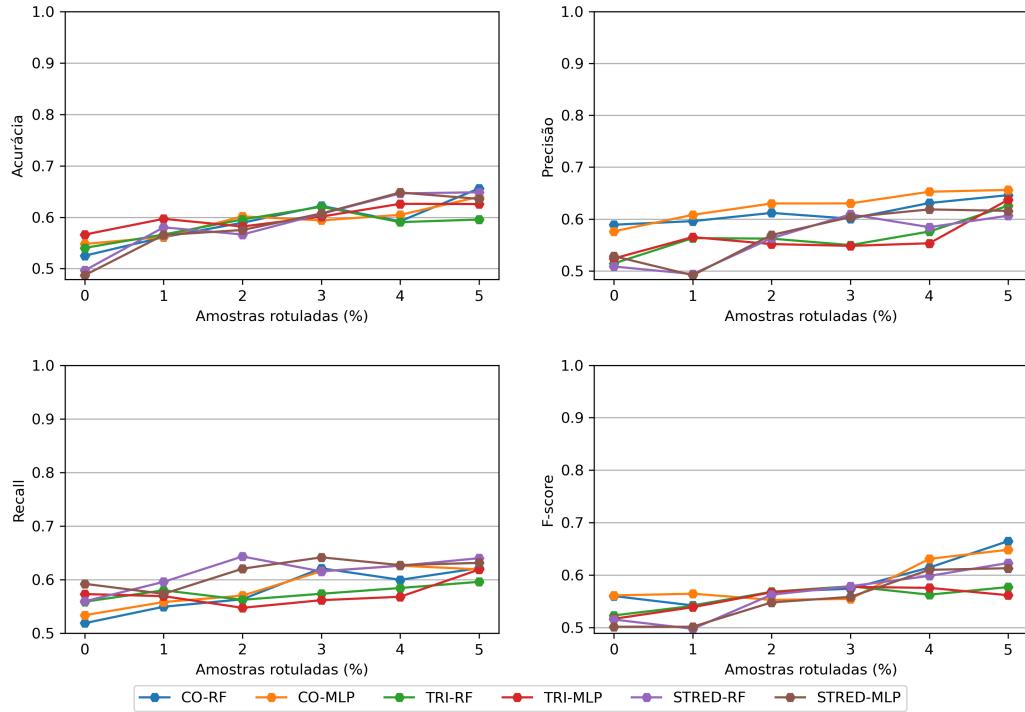


Figura B.5: Rotulação da base MNIST na fase Indutiva.

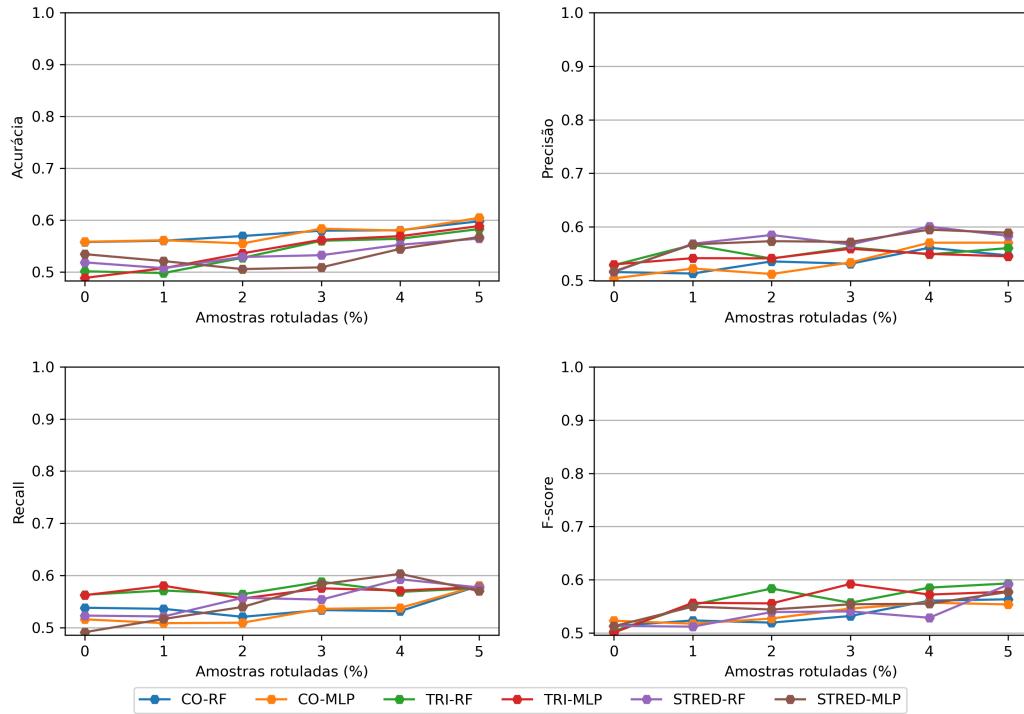


Figura B.6: Rotulação da base FASHION na fase Indutiva.

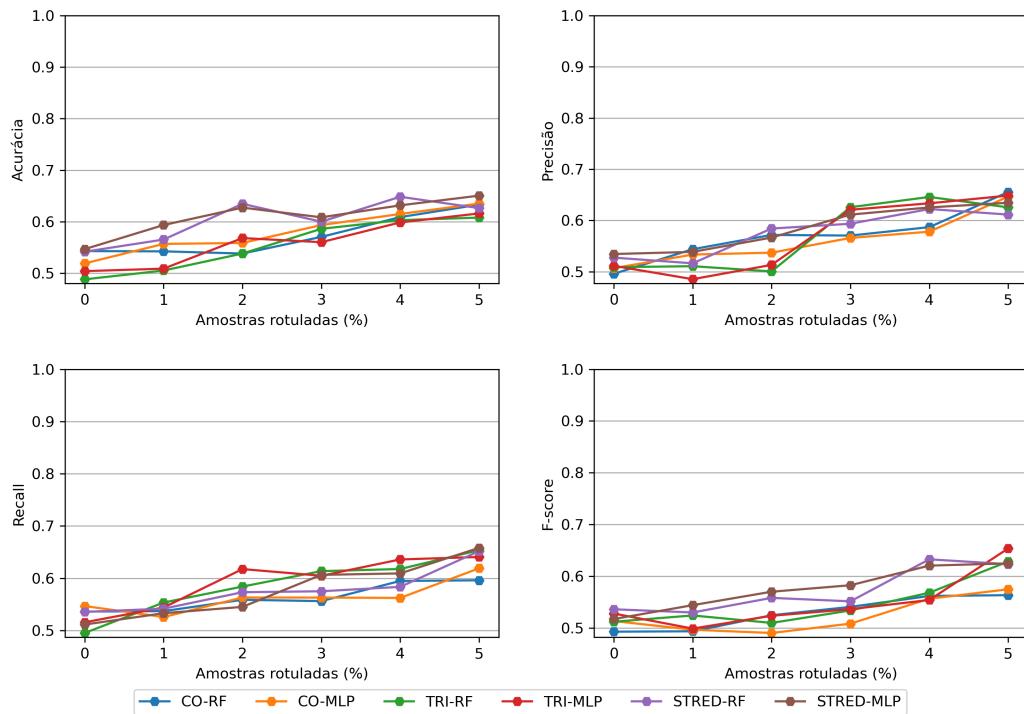


Figura B.7: Rotulação da base CIFAR-10 na fase Indutiva.

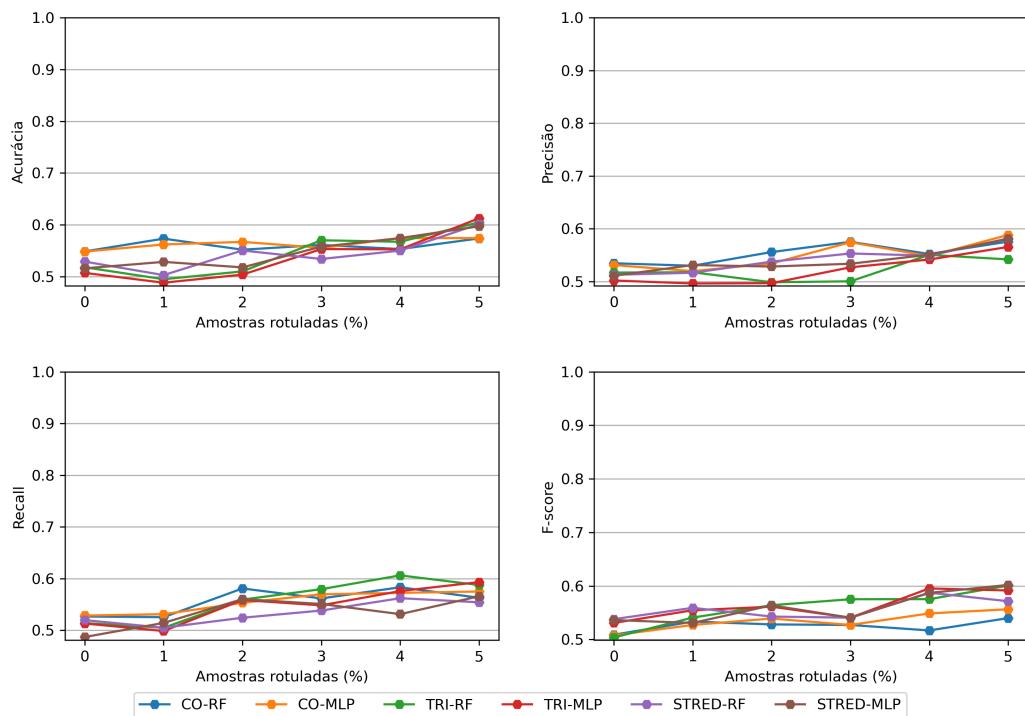


Figura B.8: Rotulação da base SLT-10 na fase Indutiva.

Apêndice C

Resultados modelo da literatura

Os resultados apresentados no Capítulo 6 foram considerando o SVM como classificador base para o modelo de (Livieris 2019). Este apêndice apresenta os resultados dos modelos da literatura utilizando MLP e RF como classificador base. Todas as figuras deste apêndice são apenas os resultados para todas bases de dados para o modelo (Livieris 2019) nas fases transdutiva e indutiva.

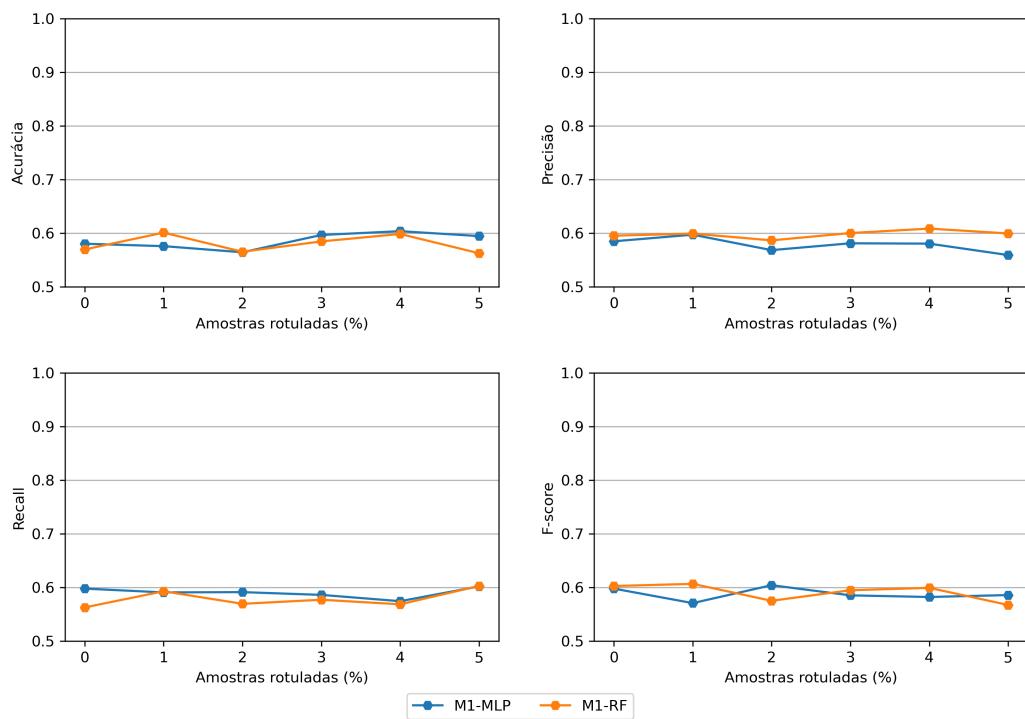


Figura C.1: Rotulação da base EPILEPSIA na fase transdutiva.

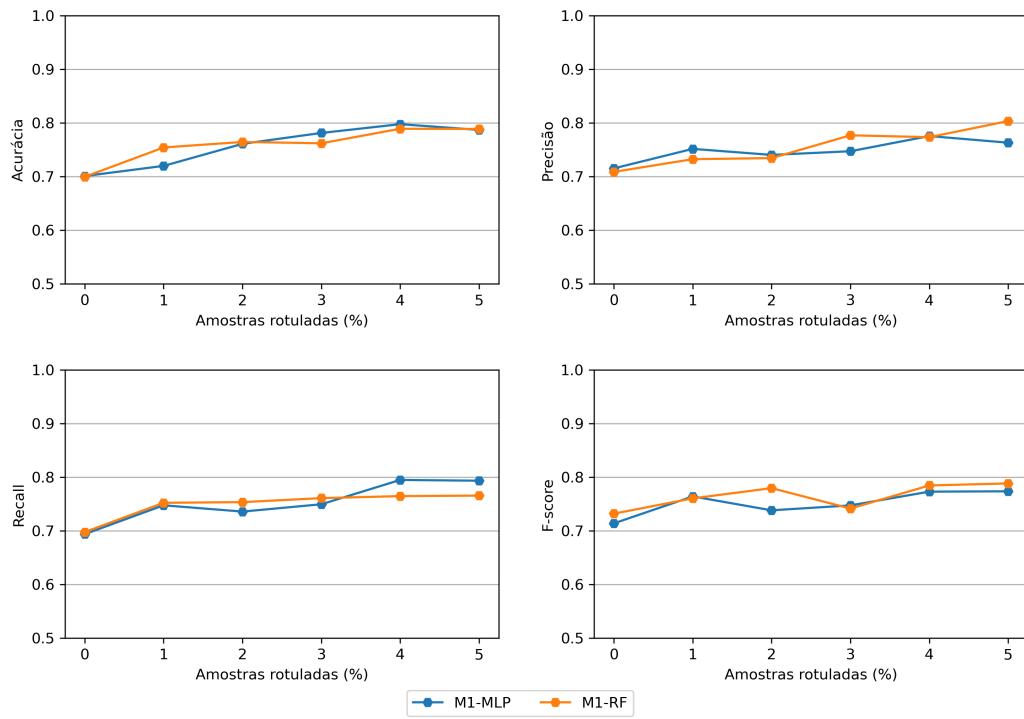


Figura C.2: Rotulação da base REUTERS na fase transdutiva.

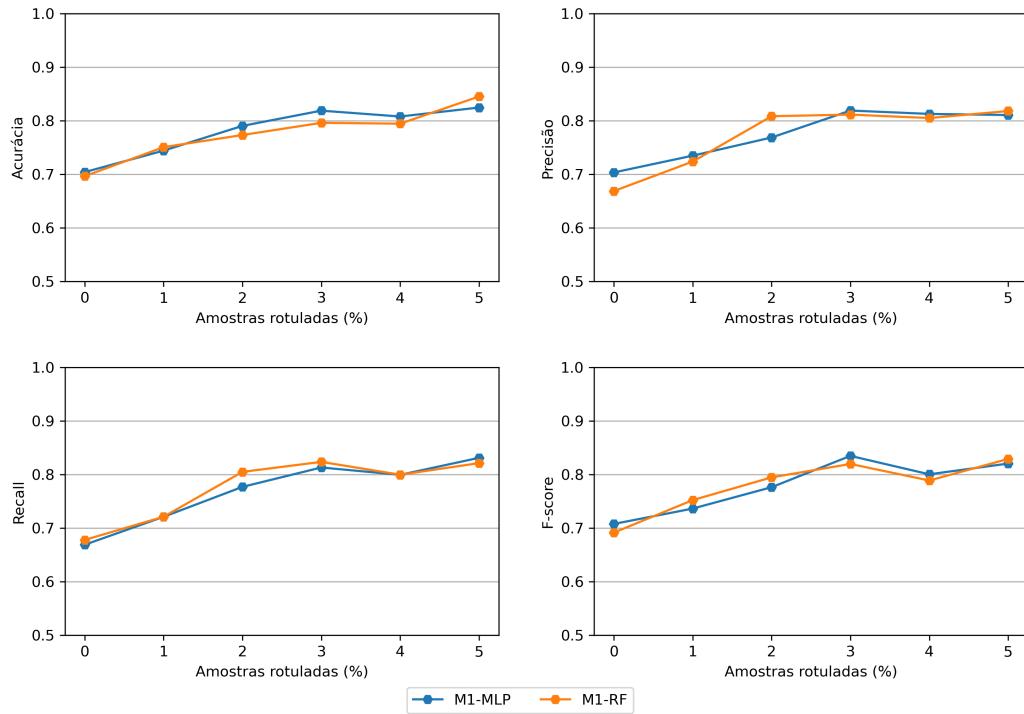


Figura C.3: Rotulação da base GAS na fase transdutiva.

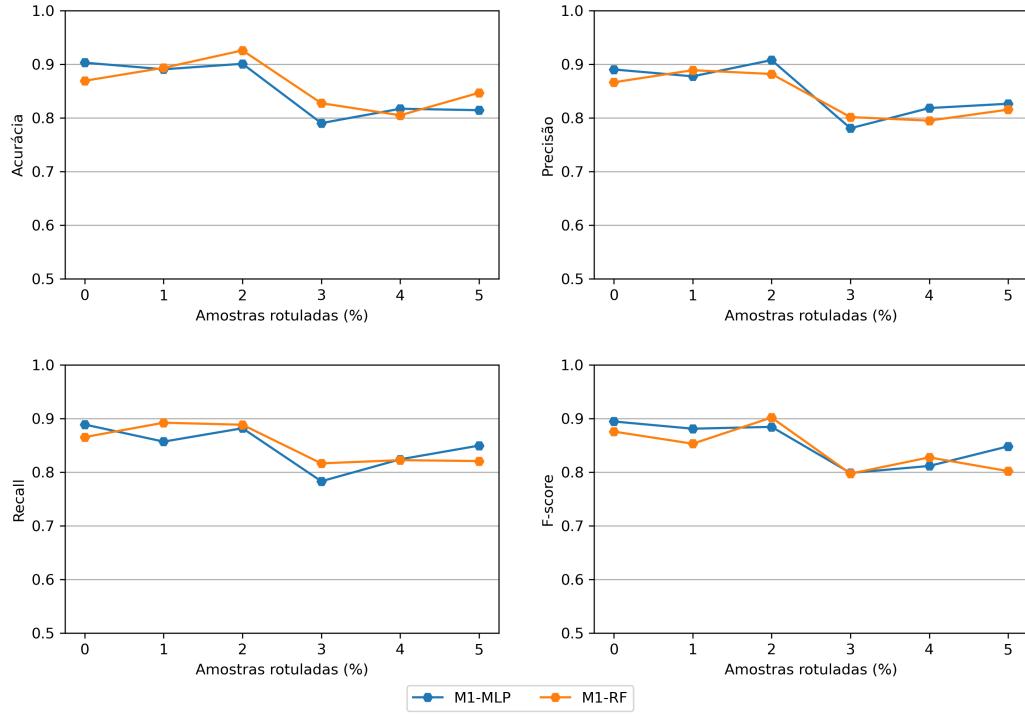


Figura C.4: Rotulação da base PENDIGITS na fase transdutiva.

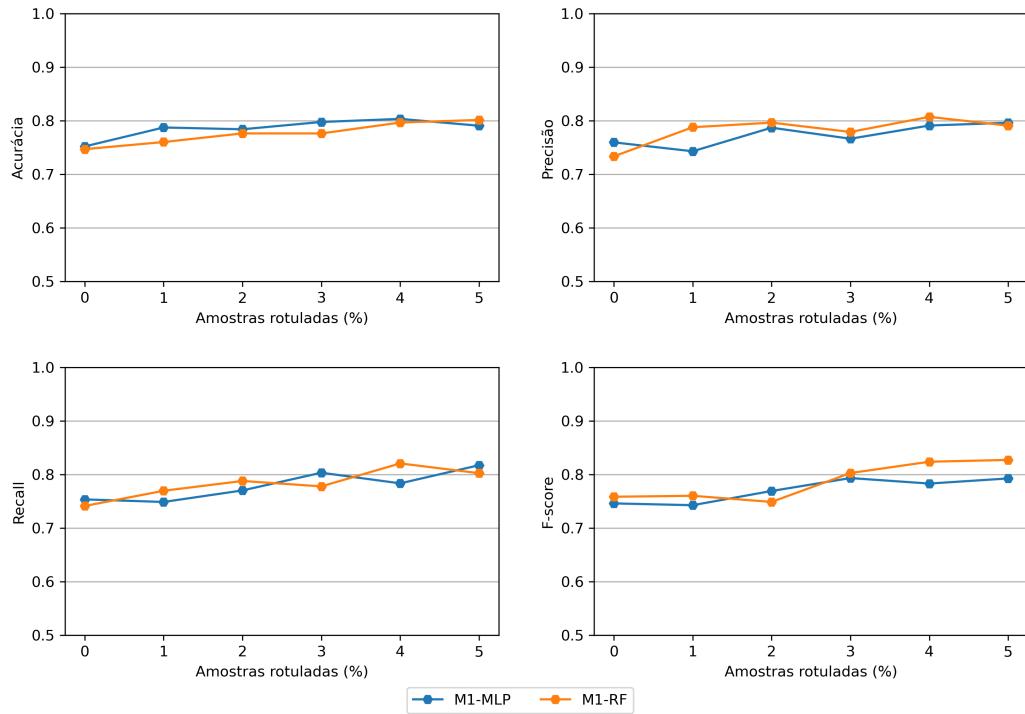


Figura C.5: Rotulação da base MNIST na fase transdutiva.

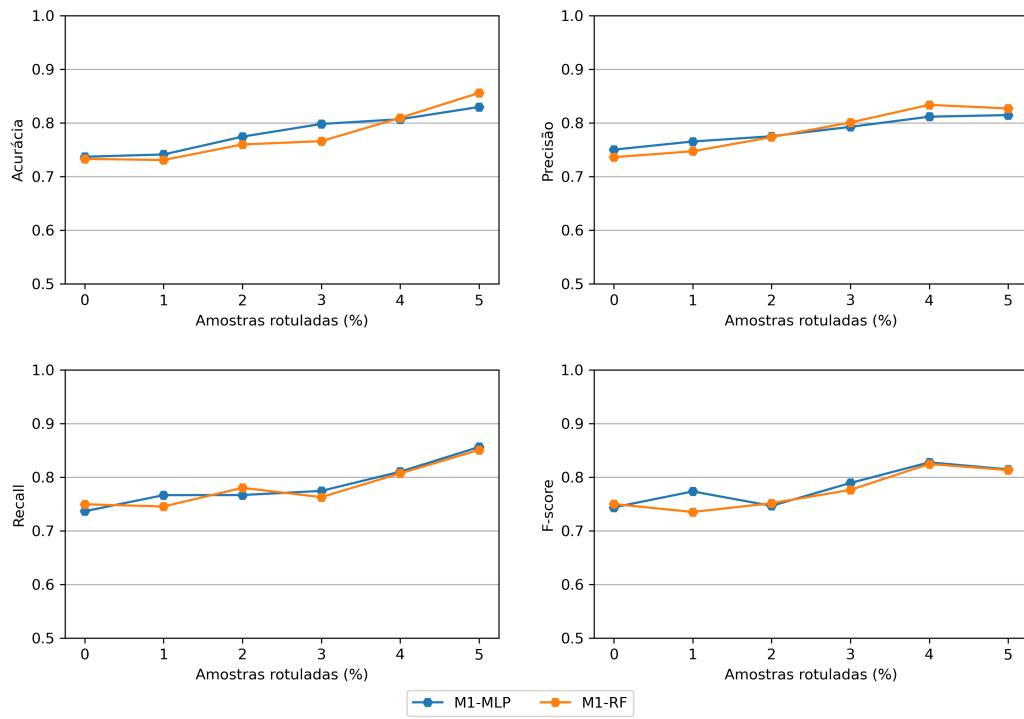


Figura C.6: Rotulação da base FASHION na fase transdutiva.

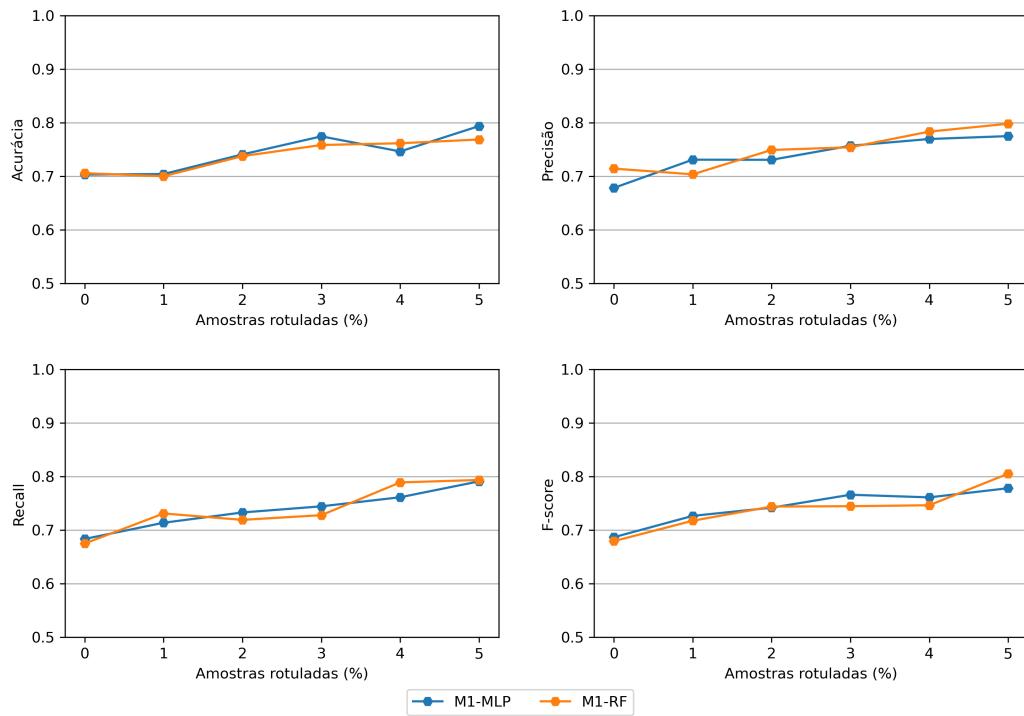


Figura C.7: Rotulação da base CIFAR-10 na fase transdutiva.

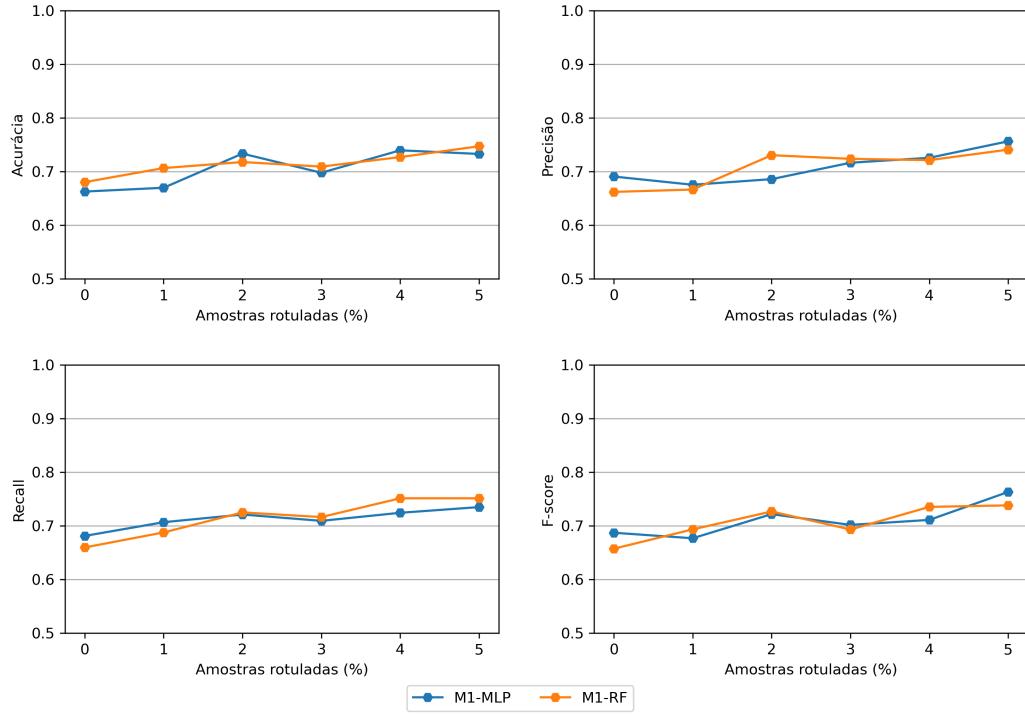


Figura C.8: Rotulação da base SLT-10 na fase transdutiva.

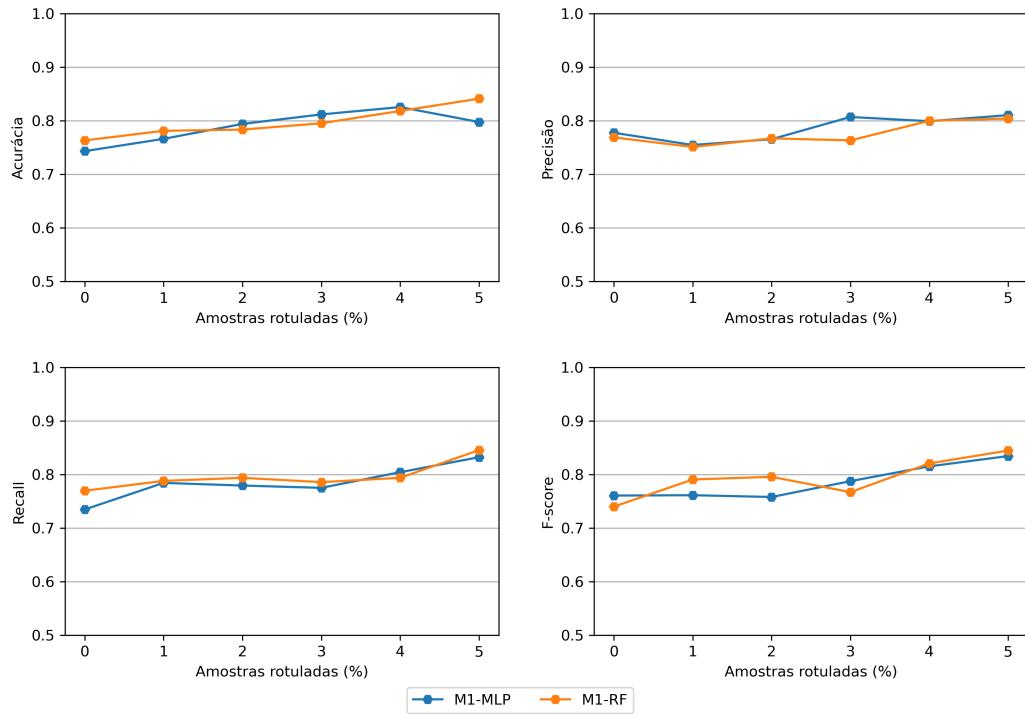


Figura C.9: Rotulação da base EPILEPSIA na fase Indutiva.

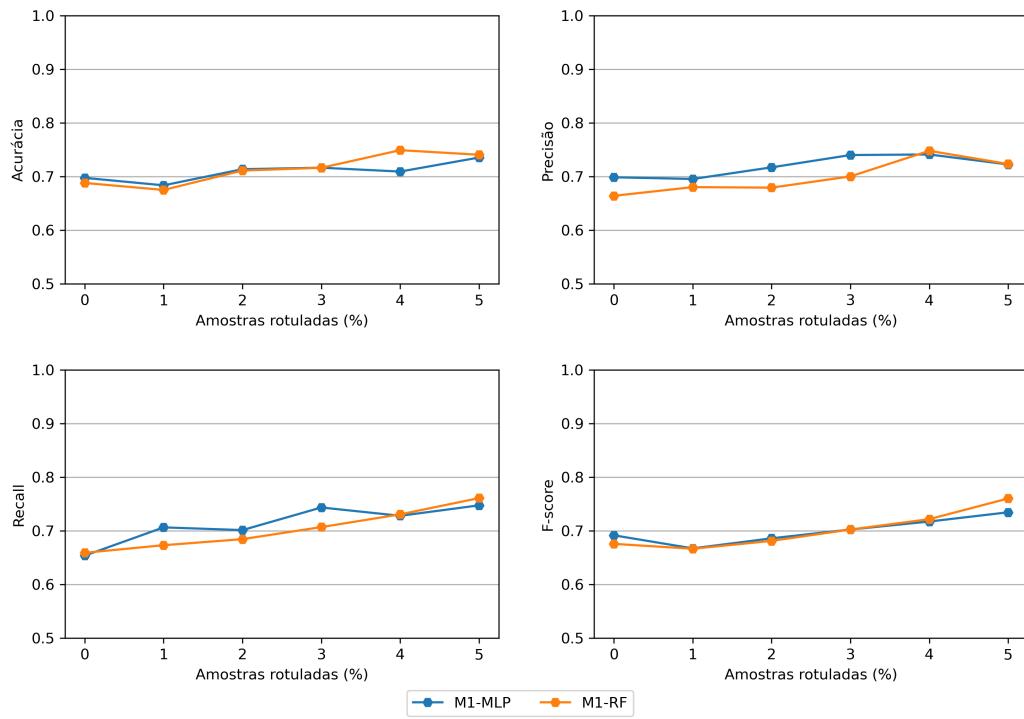


Figura C.10: Rotulação da base REUTERS na fase Indutiva.

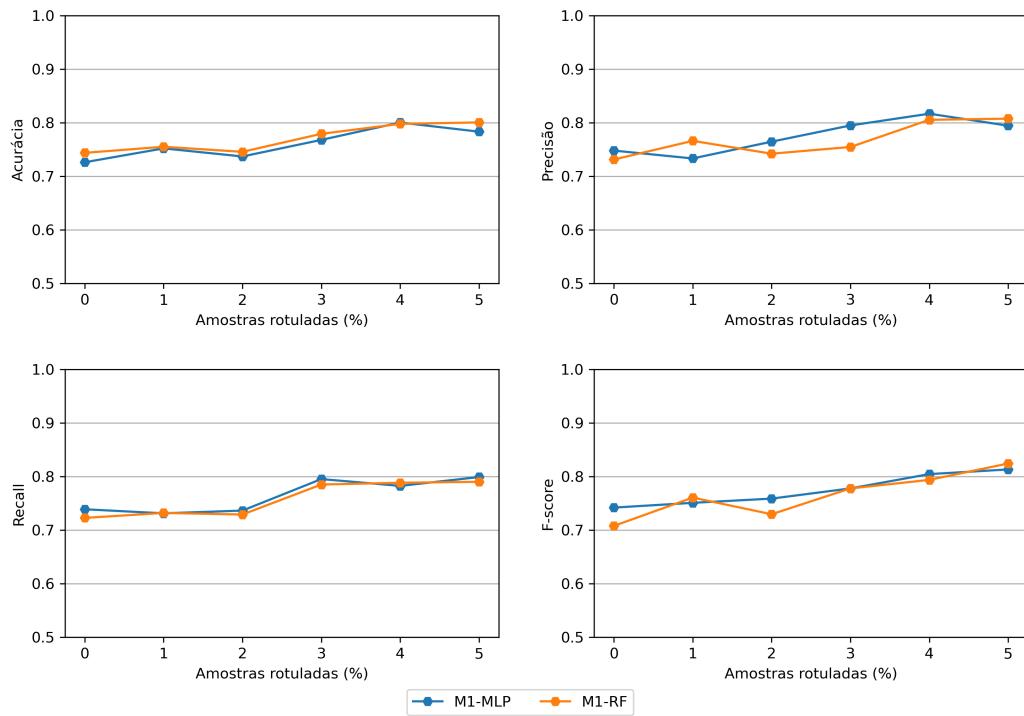


Figura C.11: Rotulação da base GAS na fase Indutiva.

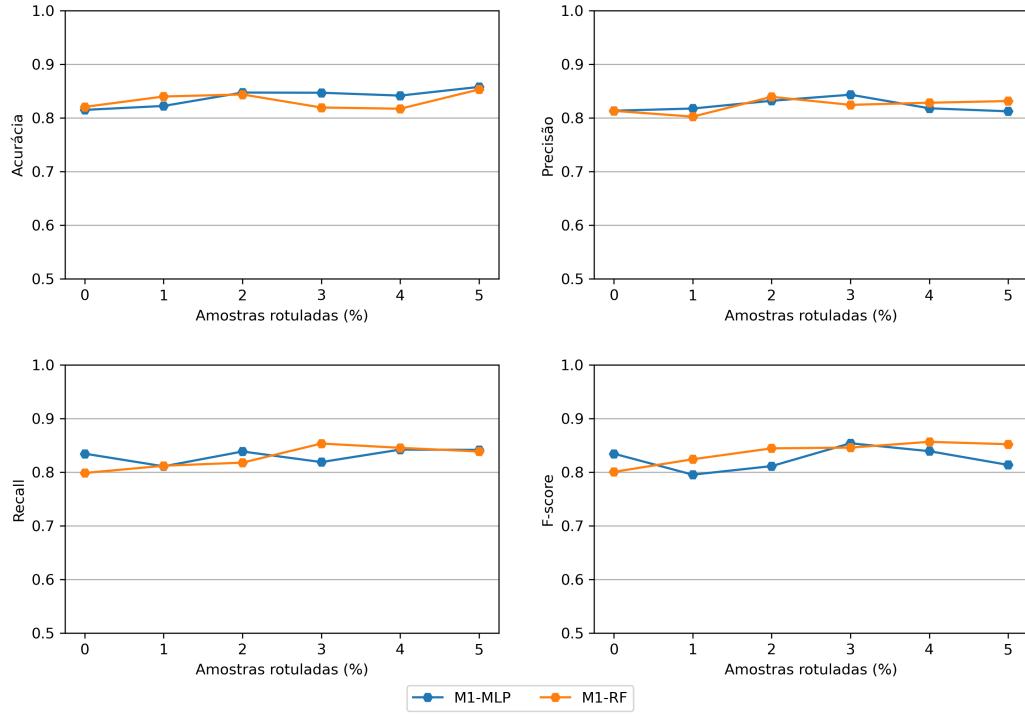


Figura C.12: Rotulação da base PENDIGITS na fase Indutiva.

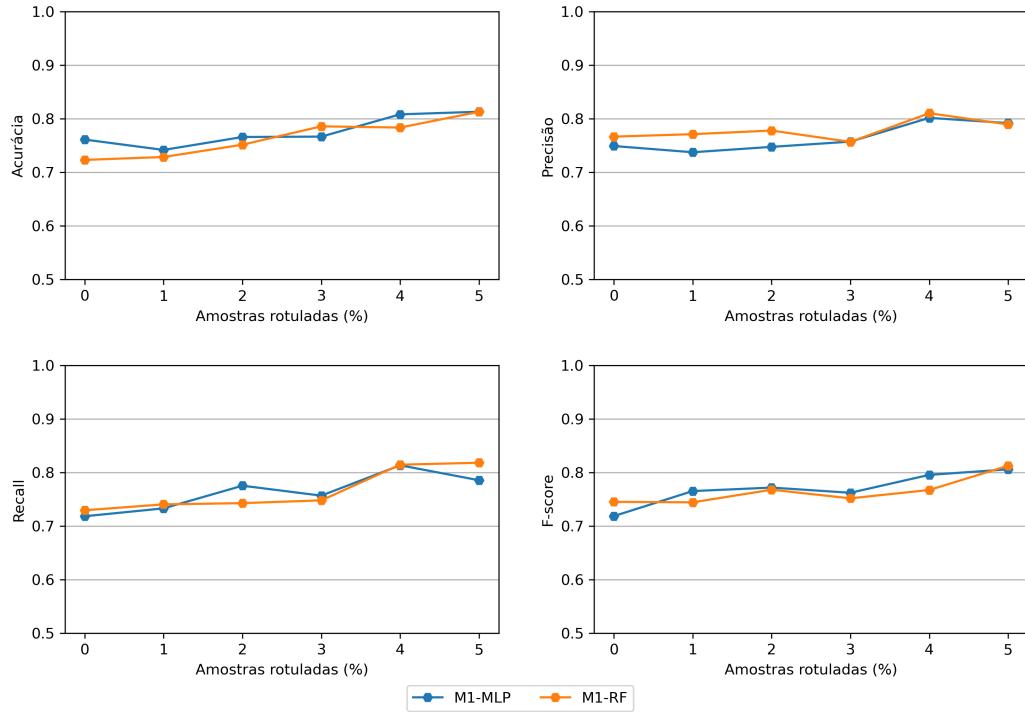


Figura C.13: Rotulação da base MNIST na fase Indutiva.

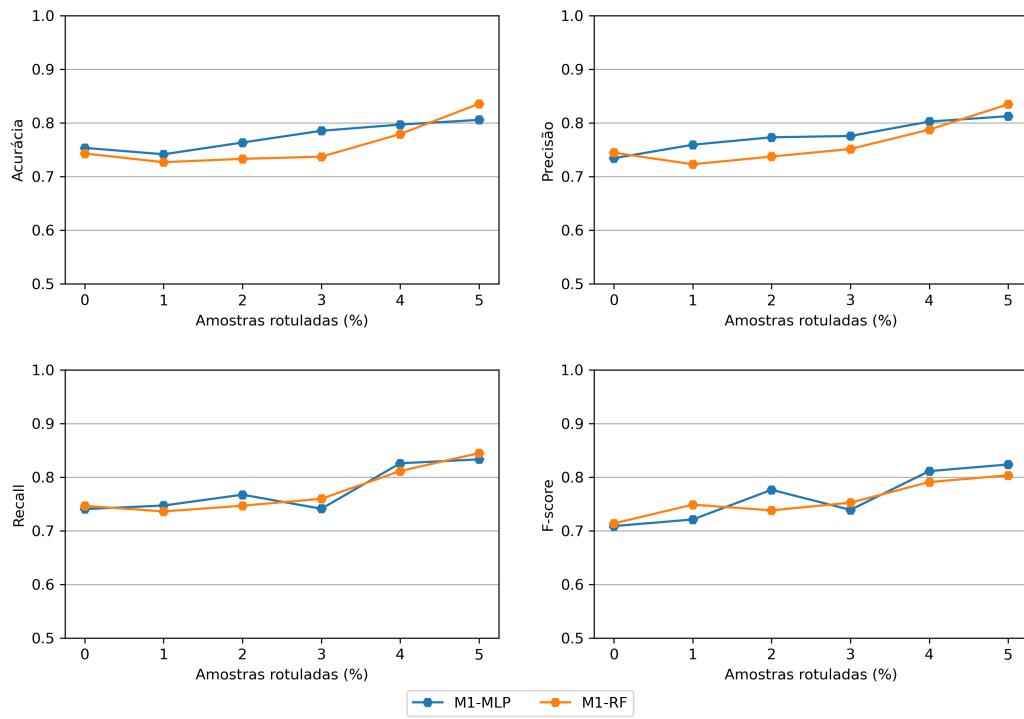


Figura C.14: Rotulação da base FASHION na fase Indutiva.

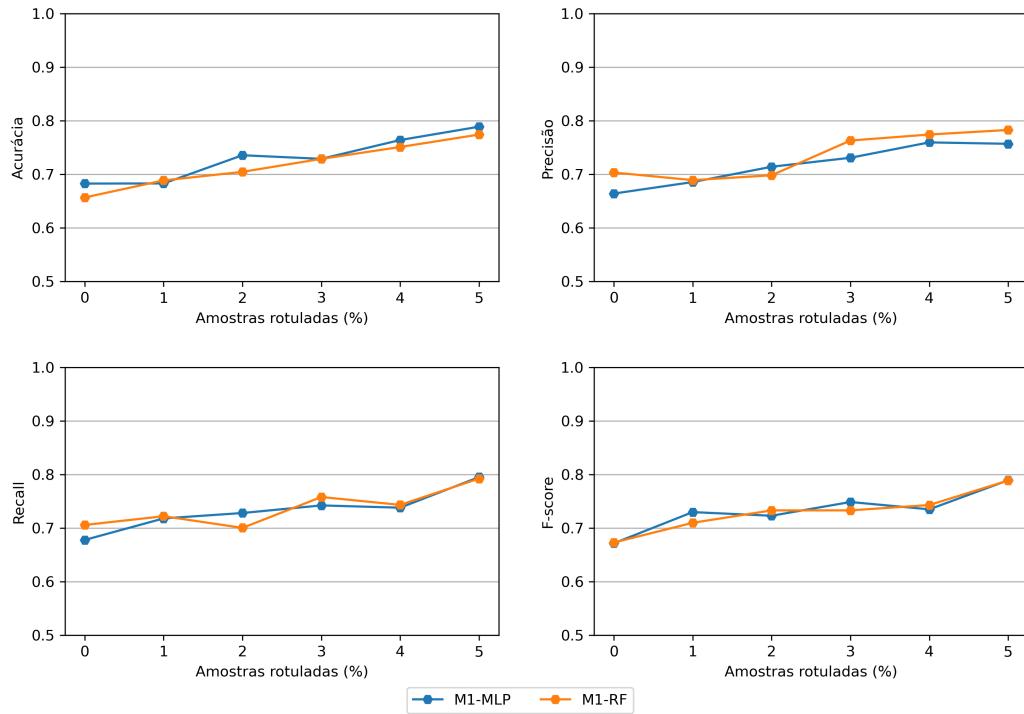


Figura C.15: Rotulação da base CIFAR-10 na fase Indutiva.

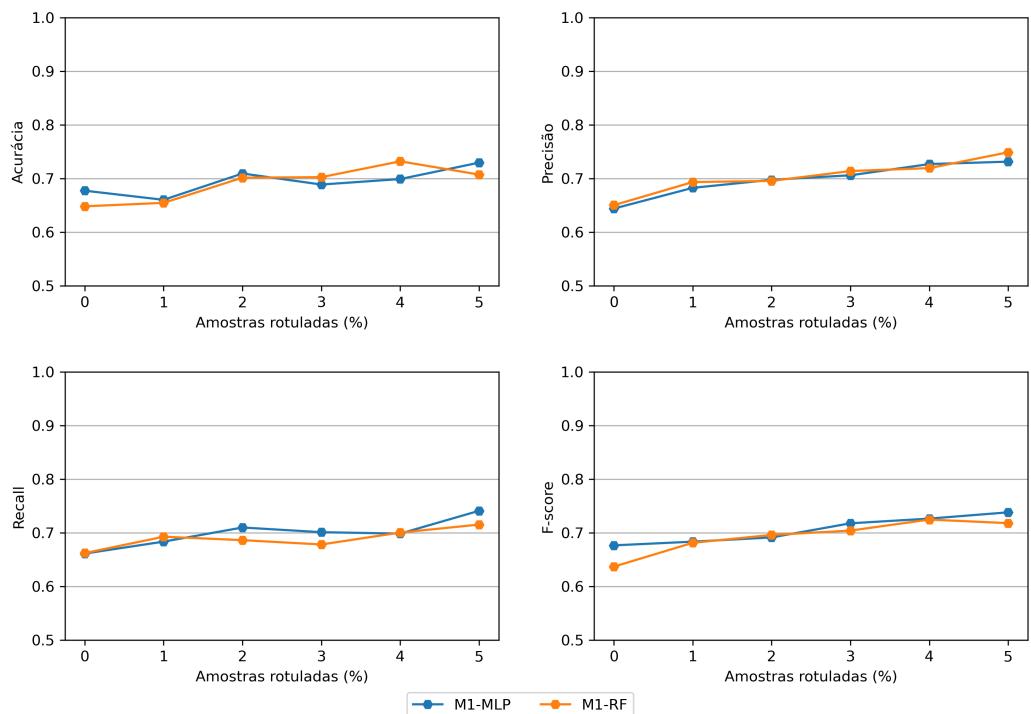


Figura C.16: Rotulação da base SLT-10 na fase Indutiva.