UNIVERSIDADE FEDERAL RURAL DO RIO DE JANEIRO
INSTITUTO MULTIDISCIPLINAR

LÍVIA DE AZEVEDO DA SILVA

Uma análise empírica do efeito do Ruído de Classe no aprendizado de Redes Neurais Artificiais

Prof. Filipe Braida do Carmo, D.Sc.
Orientador

Nova Iguaçu, Abril de 2022

Uma análise empírica do efeito do Ruído de Classe no aprendizado de Redes Neurais Artificiais

Lívia de Azevedo da Silva

Projeto Final de Curso submetido ao Departamento de Ciência da Computação do Instituto Multidisciplinar da Universidade Federal Rural do Rio de Janeiro como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Apresentado por:

_____

Lívia de Azevedo da Silva

Aprovado por:

_____

Prof. Filipe Braida do Carmo, D.Sc.

_____

Prof. Leandro Guimarães Marques Alvim, D.Sc.

_____

Prof. Bruno José Dembogurski, D.Sc.

NOVA IGUAÇU, RJ - BRASIL

Abril de 2022

Agradecimentos

Primeiramente, queria agradecer a Deus e a Nossa Senhora Aparecida por estarem comigo e terem me dado forças para chegar até aqui.

Agradeço à toda minha família, pelo apoio e preocupação. Em especial, agradeço aos meus pais, Eurídice e José Henrique, pelo apoio e suporte em vários aspectos que me permitiram fazer a faculdade da melhor forma possível e agradeço a minha tia Rosana, que no ano de 2021 ficou ao meu lado me dando suporte em um dos momentos mais difíceis da minha vida, em que quase perdi a minha mãe para a COVID-19, e também me ouviu e acolheu com conversas em um momento em que eu não acreditava que iria conseguir terminar a faculdade e passar por outros obstáculos, me incentivando a voltar a pensar positivo e que as coisas dariam certo.

Agradeço aos meus amigos que conheci na Rural: Andressa Oliveira, Desiree Araújo, Fernanda Oliveira, Hosana Gomes, Mayara Marques, Samara Almendane, Thiago Frazão, Vinícius Rigoni e William Gomes. Muito obrigada pela companhia e apoio, pelas conversas sobre diversos assuntos e por todos os momentos divertidos que passamos. Vocês tornaram essa minha caminhada mais leve e divertida. Agora, respondendo a pergunta de uma das nossas festinhas: "E o TCC ta como?", eu digo: agora ta finalizado!

Um agradecimento especial à Hosana e William, com quem compartilhei vários trabalhos de faculdade desde o início, começando pelo jogo de Computação I. Não somos um pólipo, mas obrigada por me ouvirem cantar Lua de Cristal durante o trabalho de Compiladores (haha!).

Agradeço ao meu orientador e professor Filipe Braida, por toda a paciência e por

ter me dado todo o suporte e orientação necessários para a execução deste trabalho. Obrigada também pelos conselhos dados que fugiam do escopo do TCC.

Agradeço ao professor Leandro Alvim, pela orientação durante a Iniciação Científica que realizei, sendo um evento marcante durante minha trajetória na graduação.

Agradeço à professora Lígia, pela orientação durante o tempo em que fui monitora, uma experiência gratificante e que me trouxe aprendizados. Obrigada também pelas conversas que tivemos e conselhos que você me deu que me ajudou a refletir e tomar uma decisão de seguir um determinado caminho que, até o momento, já me trouxe muitos aprendizados e crescimento que eu não imaginava que eu teria.

Agradeço ao corpo docente do curso de Ciência da Computação, pelo conhecimento compartilhado durante as disciplinas.

Agradeço ao curso de Ciência da Computação da Rural como um todo, por tudo que me proporcionou que ajudou no meu desenvolvimento tanto intelectual e também como pessoa.

A todos os citados, muito obrigada por tudo.

A dedicação é a chave para o sucesso.

RESUMO

Uma análise empírica do efeito do Ruído de Classe no aprendizado de Redes Neurais Artificiais

Lívia de Azevedo da Silva

Abril/2022

Orientador: Filipe Braida do Carmo, D.Sc.

O Aprendizado de Máquina é uma área com muito vislumbre e que fornece diversas soluções atualmente. As Redes Neurais Artificiais são modelos preditivos bastante conhecidos em Aprendizado de Máquina por serem a ideia base na construção de vários modelos de Aprendizado Profundo, uma área com muitas contribuições nos últimos tempos. Muitos dos problemas de Aprendizado de Máquina são de classificação, os quais tem como objetivo aprender a relação entre as características disponíveis nos dados e a uma classe associada a este dado. O sucesso de qualquer modelo de classificação depende da qualidade dos dados que são fornecidos, principalmente a informação associada a classe daquele dado. Pelo fato do processo de rotulação dessas classes possuir fatores que podem levar a erros, a presença de uma classe diferente da real em algum dado se torna possível, o que caracteriza o Ruído de Classe. Na literatura, existem diversos trabalhos experimentais para evidenciar o quanto os modelos de Aprendizado de Máquina são impactados negativamente pelo Ruído de Classe e o quanto eles podem ser naturalmente resistentes ao mesmo em diferentes contextos, sendo importante para deixar evidências do quanto e como o Ruído de Classe pode afetar o aprendizado destes modelos. A proposta deste trabalho é realizar experimentos com diferentes modelos de Redes Neurais Artificiais treinados em algumas bases de dados com a presença do Ruído de Classe gerado artificialmente, para no fim avaliar a influência do Ruído de Classe nas redes treinadas. Os resultados experimentais mostraram que a dificuldade inerente ao domínio dos dados pode ajudar a piorar o impacto do Ruído de Classe; que a complexidade do modelo neural pode afetar a resistência ao Ruído de Classe e a escolha da função de ativação dos neurônios da rede pode impactar na resistência ao Ruído de Classe.

ABSTRACT

Uma análise empírica do efeito do Ruído de Classe no aprendizado de Redes Neurais Artificiais

Lívia de Azevedo da Silva

Abril/2022

Advisor: Filipe Braida do Carmo, D.Sc.

Machine Learning is a very promising area that provides many solutions nowadays. Artificial Neural Networks are well known predictive models in Machine Learning for being the basic idea in building many Deep Learning models, an area with many contributions in recent times. Many Machine Learning problems are classification problems, which aim to learn the relationship between features available in the data and a class associated with that data. The success of any classification model depends on the quality of the data that is provided, especially the information associated with the class of that data. Because the process of labeling these classes has factors that can lead to errors, the presence of a class different from the real one in some data becomes possible, which characterizes the Label Noise. In the literature, there are several experimental works to evidence how Machine Learning models are negatively impacted by Label Noise and how much they can be naturally resistant to it in different contexts, being important to leave evidence of how Label Noise can affect the learning of these models. The purpose of this work is to perform experiments with different Artificial Neural Network models trained on some data sets with the presence of artificially generated Label Noise, in order to evaluate the influence of Label Noise on the trained networks. The experimental results showed that the inherent difficulty of the data domain can help to worsen the impact of Label Noise; the complexity of the neural model can affect Label Noise resistance and the choice of activation function of the network neurons can impact Label Noise resistance.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

RNA	*Rede Neural Artificial*
IA	*Inteligência Artificial*
MLP	*Multilayer Perceptron*
ReLU	*Rectified Linear Unit*
kNN	*k-nearest neighbors*
SVM	*Support Vector Machines*
PAC	*Probably Approximately Correct*
NCAR	*Noise Completely at Random*
NAR	*Noise at Random*
NNAR	*Noise not at Random*
CNN	*Convolutional Neural Network*
ResNet	*Residual Neural Network*
VGG16	*Visual Geometry Group*
DC	*Dependente da Classe*
CL	*Concentrado Localmente*
RDA	*Ruído Dependente do Atributo*
Adam	*Adaptive Moment Estimation*
t-SNE	*t-Distributed Stochastic Neighbor Embedding*

Sumário

Capítulo 1

Introdução

Um dos tópicos de bastante interesse e que dá suporte a soluções em vários contextos é o Aprendizado de Máquina. Aplicações como reconhecimento de imagens, reconhecimento de voz e processamento de linguagem natural são exemplos práticos bem evidentes nos quais Aprendizado de Máquina é a base para solucionar as suas demandas. (RUSSELL; NORVIG, 2021)

Os exemplos anteriores têm em comum o fato de que um *software* programado explicitamente para resolver tais problemas seria quase impossível de ser feito, pois os conceitos e regras necessários para obter sucesso nas aplicações são muito complexos, específicos e flexíveis para serem definidos em código. Acaba sendo necessário ter algoritmos que possam aprender esses conceitos e regras implícitas e difíceis e, dado este conhecimento, estar disponível para ser aplicado em futuros casos, e isto que o Aprendizado de Máquina propõe. (MITCHELL, 1997; RUSSELL; NORVIG, 2021)

Além disso, nos últimos tempos um grupo de algoritmos de Aprendizado de Máquina vem ganhando bastante notoriedade pelos grandes resultados obtidos, sendo este o de maior aplicação nos exemplos de aplicações citadas anteriormente também. Este grupo refere-se a um subtópico do Aprendizado de Máquina chamado de Aprendizado Profundo e a ideia base na construção destes algoritmos vem de um modelo chamado de Redes Neurais Artificiais, um tipo de modelo de Aprendizado de Máquina. (RUSSELL; NORVIG, 2021; GOODFELLOW; BENGIO; COURVILLE,

2016)

Para que um algoritmo de Aprendizado de Máquina aprenda, dados de exemplos precisam existir. Sem dados, não há como ter um aprendizado. Assim, estes dados são coletados e construídos de acordo com o domínio do problema em questão, para serem usados pelo algoritmo de aprendizado para aprender os conceitos do problema.

Um tipo de problema comum em Aprendizado de Máquina é o de classificação: dada algumas informações sobre os dados, este dado deve ser classificado para uma determinada classe dentre um conjunto de classes possíveis para o problema. Um exemplo prático é o reconhecimento de imagens: tendo imagens de cachorros e gatos, se quer classificar corretamente uma imagem nova como cachorro ou gato.

Percebe-se que os dados são cruciais para o sucesso destes algoritmos de Aprendizado de Máquina. Qualquer erro nos dados causaria o fornecimento de informações erradas para o algoritmo que consequentemente aprenderia conceitos incorretos e iria realizar predições erradas. Na literatura, esse tipo de erro nos dados é chamado de ruído. Mais especificamente, o erro relacionado a classe em um dado, ou seja, o dado está definido com uma classe diferente da original é chamado de Ruído de Classe. A intervenção humana é o fator chave para que o Ruído de Classe ocorra, pelo fato da determinação das classes dos dados estarem sujeitos ao erro humano. (FRÉNAY; VERLEYSEN, 2013)

O Ruído de Classe vem tendo uma atenção na literatura em propor ações ou técnicas que possam mitigar ou corrigir este ruído nos dados, além de trabalhos experimentais que ratificam o efeito negativo do Ruído de Classe na classificação, tendo ênfase nos últimos tempos no campo dos algoritmos de Aprendizado Profundo. (FRÉNAY; VERLEYSEN, 2013; HAN et al., 2020; SONG et al., 2020; CORDEIRO; CARNEIRO, 2020)

Os trabalhos experimentais são uma base importante para evidenciar comportamentos específicos que um algoritmo de Aprendizado de Máquina, dado certas características, pode ter em domínios de dados específicos, e há muitos que podem ser feitos para confirmar um indício ou apresentar novos pontos a serem analisados

em conjunto ao Ruído de Classe presente nos dados durante o aprendizado.

1.1 Objetivo

O objetivo deste trabalho é oferecer resultados empíricos que forneçam uma evidência prática do efeito do Ruído de Classe no aprendizado de Redes Neurais Artificias, respaldados em diferentes tipos de Ruído de Classe e domínios de dados distintos. Diferentes modelos neurais serão treinados para contemplar análises sob algumas perspectivas que vão contribuir para a melhor consolidação dessas evidências práticas.

1.2 Organização do Trabalho

Os capítulos seguintes serão organizados desta forma:

- O Capítulo 2 trará a fundamentação teórica mais detalhada dos tópicos sobre Aprendizado de Máquina, Redes Neurais Artificiais e Ruído de Classe, essenciais para o entendimento geral do trabalho;
- O Capítulo 3 detalhará a proposta deste trabalho, apresentando a ideia geral da metodologia adotada para alcançar o objetivo principal, além de explanar alguns trabalhos relacionados;
- O Capítulo 4 apresentará quais experimentos foram feitos e explicará como foram feitos, junto dos resultados obtidos e suas análises;
- O Capítulo 5 trará as conclusões e considerações finais a respeito dos resultados obtidos.

Capítulo 2

Fundamentação Teórica

Neste capítulo, vamos fornecer os principais conceitos teóricos necessários para o entendimento da proposta deste trabalho. Entre eles, vamos apresentar conceitos relacionados a Aprendizado de Máquina, Aprendizado Supervisionado, Redes Neurais Artificias e Ruído de Classe.

2.1 Aprendizado de Máquina

Certamente uma das características que torna o ser humano um agente bastante flexível e suscetível a melhoras é a sua capacidade de aprendizado. Por muito tempo se pensou se este tipo de característica fosse possível para uma máquina simular, mesmo de uma forma limitada. Isto seria essencial em soluções computacionais que auxiliam ou resolvem alguma determinada questão. (RUSSELL; NORVIG, 2021; MITCHELL, 1997)

Podemos citar alguns exemplos reais de utilidade desta capacidade de aprendizado: um *software* para servir como um assistente virtual, o qual o mesmo teria aprendido previamente e vai aprendendo a fazer reconhecimento de voz para poder entender comandos de voz dados por um usuário; um *software* voltado para uma tradução automática de texto de uma linguagem para outra, o qual o mesmo poderia aprender a traduzir tendo como base referencial um conjunto de dados com uma relação de

texto da linguagem "X" para a linguagem "Y"; um *software* voltado para detecção se um e-mail é considerado como *spam* ou não, aprendendo com outros exemplos de e-mails que são certamente um *spam*. (RUSSELL; NORVIG, 2021; MITCHELL, 1997)

Esta capacidade de aprendizado é um chamariz maior ainda porque é impossível prever todas as situações futuras que a solução teria que lidar e, em exemplos como o reconhecimento de voz, não se saberia como programar explicitamente esta solução para todos os casos, pelo problema de ter um número grande de variáveis a serem consideradas. Sendo assim, Aprendizado de Máquina (*Machine Learning*) é o campo de estudo voltado em desenvolver soluções computacionais com esta capacidade de aprendizado (RUSSELL; NORVIG, 2021).

Para formalizar a ideia de aprendizado no contexto computacional e com isso a ideia geral sobre Aprendizado de Máquina, Mitchell (1997) deu a seguinte definição:

> "Um programa de computador aprende a partir de uma experiência E com relação a alguma classe de tarefas T e uma medida de performance P, se a performance do programa nas tarefas em T, medido por P, melhora com a experiência E." (MITCHELL, 1997, p.2, tradução própria)

Em outras palavras, a medida que o programa vai passando por experiências, a sua performance nas tarefas que lhe são designadas melhora cada vez mais. Com esta ideia, pode-se estruturar como seria definido um problema de aprendizagem. Por exemplo, considerando um programa que deve aprender a jogar damas, T seria o ato de jogar damas, P a porcentagem de jogos ganhos contra oponentes e E os jogos praticados contra ele mesmo. Um outro exemplo: considerando o problema de detecção se um e-mail é *spam* ou não, T seria o reconhecimento de um e-mail como sendo *spam* ou não, P a porcentagem de vezes que o programa reconheceu um e-mail como *spam* corretamente e E um conjunto de dados de e-mails acompanhado com a classificação se aquele e-mail é *spam* ou não. (MITCHELL, 1997)

Existem três tipos diferentes de aprendizado: aprendizado supervisionado, aprendizado não supervisionado e aprendizado por reforço. Os três se diferenciam depen-

dendo do tipo de *feedback* disponível relacionado aos dados recebidos pelo algoritmo de aprendizado. (RUSSELL; NORVIG, 2021; GHAHRAMANI, 2003)

No aprendizado supervisionado é disposto um conjunto de pares de entrada de dados e saída de dados, o qual o programa irá se basear para aprender uma função que irá representar a relação entre a entrada e saída. O *feedback* neste caso são as saídas associadas as suas respectivas entradas. Por exemplo, no caso do problema da detecção de um e-mail como *spam*, seria fornecido um conjunto de pares de e-mails e um rótulo dizendo se aquele e-mail é *spam* ou não.

No aprendizado não supervisionado é disposto um conjunto de dados de entrada, sem ter alguma saída especificada e assim, neste caso, não há nenhum tipo de *feedback*. O objetivo principal do programa neste tipo de aprendizado é aprender padrões presentes nos dados de entrada para eventuais análises. Por exemplo, dado um conjunto de textos de notícias, um programa poderia aprender a separar estas notícias em grupos diferentes de acordo com a temática da notícia, por exemplo, sem ter tido acesso a categoria real daquela notícia.

No aprendizado por reforço é disposto uma série contínua de *feedbacks* vindas do ambiente que podem ser recompensas ou punições. A ideia deste aprendizado é o programa ir executando ações contínuas que irão mudar o estado do ambiente o qual ele atua e, dependendo do novo estado do ambiente, é fornecido ao programa um reforço de recompensa ou uma punição se o novo estado do ambiente foi desejável ou não, respectivamente. O objetivo principal é o programa, com este processo sucessivo, aprenda sempre a tomar as ações que maximizem o ganho de recompensas. Por exemplo, no problema de aprender a jogar xadrez, é dito ao programa no fim de uma partida se ele ganhou (recompensa) ou perdeu (punição) e, de acordo com isto, o programa iria reavaliar as jogadas (ações) feitas durante o jogo dado o *feedback* para nos próximos jogos mudar as estratégias das jogadas.

Na próxima seção, discutiremos com mais detalhes a respeito do aprendizado supervisionado, que será o foco deste trabalho.

2.1.1 Aprendizado Supervisionado

Formalmente, definimos a tarefa do aprendizado supervisionado da seguinte forma: Dado um conjunto de treinamento $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ de exemplos (dados) que representam uma função desconhecida $y = f(x)$, o objetivo é obter uma hipótese h que mais se aproxima da função real f. (RUSSELL; NORVIG, 2021; MONARD; BARANAUSKAS, 2003)

Dada a definição acima, definimos x como sendo o atributo (*feature*) do exemplo, que descreve alguma característica específica deste exemplo e que será usado como base para o aprendizado de padrões que regem os dados. O elemento y é um atributo especial chamado de rótulo ou alvo que informa o aspecto de interesse do qual se deseja aprender a partir dos outros atributos x. Quando y é um valor que pertence a um conjunto discreto finito de rótulos, chamado de classes, define-se como um problema de classificação, enquanto que se y for um valor contínuo (real), define-se como um problema de regressão. (RUSSELL; NORVIG, 2021; MONARD; BARANAUSKAS, 2003)

Além do conjunto de treinamento que representa o domínio da tarefa que se deseja aprender, temos um conjunto de classe de hipóteses $\mathcal{H}$ onde a hipótese h será procurada; um algoritmo de aprendizado que irá, a partir de $\mathcal{H}$ e de acordo com o conjunto de treinamento fornecido, aprender os padrões que mapeiam x para y; e um modelo que será gerado a partir do algoritmo de aprendizado, correspondendo ao conhecimento concreto consolidado, e que será usado para prever o y em exemplos novos de acordo com o conhecimento disponível. A Figura 2.1 mostra a relação entre estes componentes anteriores em uma tarefa de aprendizado supervisionado. (RUSSELL; NORVIG, 2021; MONARD; BARANAUSKAS, 2003)

É importante ressaltar que $\mathcal{H}$ é determinado previamente pelo tipo de modelo de Aprendizado de Máquina que é escolhido para se construir o modelo final e, consequentemente, define o tipo da função que se terá no fim do processo de aprendizado. Por exemplo, se considerar um modelo de Regressão Linear, tem-se a restrição de aprender uma função da forma $y = ax + b$, nada mais além disto. (RUSSELL;

NORVIG, 2021; MONARD; BARANAUSKAS, 2003)

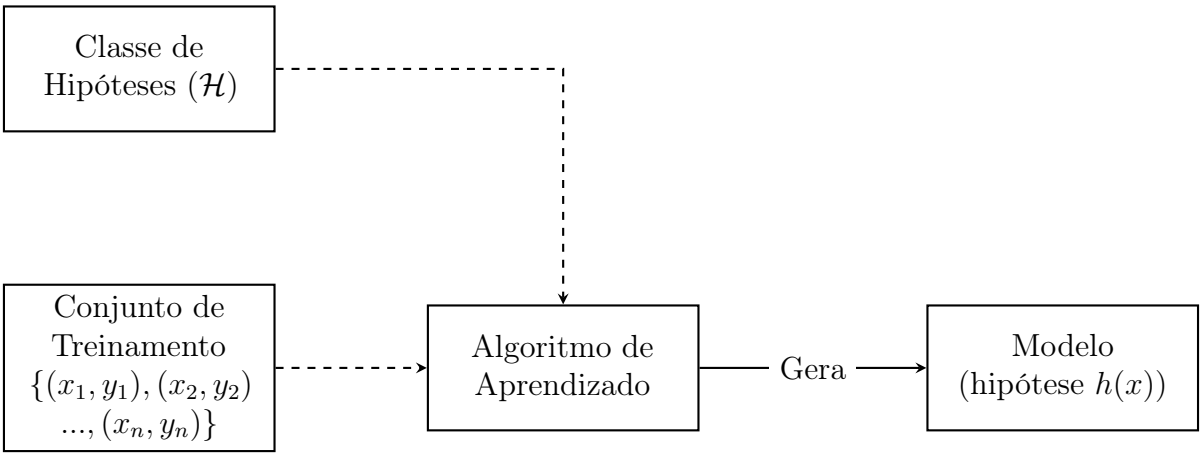


Figura 2.1: Principais componentes em uma tarefa de Aprendizado Supervisionado.

O grande objetivo de uma tarefa de Aprendizado Supervisionado é gerar modelos que tenham uma boa generalização, ou seja, a hipótese final consiga fazer as predições corretas para novos dados além daqueles usados no treinamento. Isto pode ser alcançado na escolha ideal de $\mathcal{H}$ com hipóteses flexíveis o suficiente para representar o padrão desejável que representa àquele conjunto de dados. Quando há uma má escolha de $\mathcal{H}$, tem-se um modelo com baixa generalização. (RUSSELL; NORVIG, 2021; MONARD; BARANAUSKAS, 2003; ALPAYDIN, 2014)

Aliado a isto, tem-se dois casos em que a hipótese final possui uma baixa generalização: subajuste (*underfitting*) e sobreajuste (*overfitting*). O subajuste ocorre quando a hipótese resultante do aprendizado que caracteriza o modelo é rígida e limitada com relação ao ajuste necessário no conjunto de dados considerado, sugerindo uma insuficiência para o aprendizado do padrão requerido que, consequentemente, leva a uma baixa generalização. Por exemplo, um modelo com uma hipótese como sendo uma função linear ($y = ax + b$) que tenta atuar em cima de dados que consistem em uma função polinomial de terceira ordem ($y = ax^3 + bx^2 + cx + d$) sofreria com o subajuste. (RUSSELL; NORVIG, 2021; MONARD; BARANAUSKAS, 2003; ALPAYDIN, 2014)

O sobreajuste é o inverso do subajuste: a hipótese resultante do aprendizado que caracteriza o modelo é muito flexível e abrangente com relação ao ajuste necessário no conjunto de dados considerado, o que causa o comportamento do modelo "decorar"

os dados e que ele aprende, além do padrão requerido, outros conceitos indesejáveis que se somam ao ideal, ocasionando uma baixa generalização dos dados. Sem um controle externo no aprendizado que evite o sobreajuste, na maioria dos casos o modelo final correria riscos de sofrer com este problema. Por exemplo, um modelo com uma hipótese definida pela função $y = ax^5 + bx^4 + cx^3 + dx^2 + ex + f$ que tenta atuar em cima de dados que consistem na função $y = ax^3 + bx^2 + cx + d$ poderia sofrer com o sobreajuste. (RUSSELL; NORVIG, 2021; MONARD; BARANAUSKAS, 2003; ALPAYDIN, 2014)

É comum em tarefas de aprendizado supervisionado separar uma parte dos dados de treinamento disponíveis e tirá-los do conjunto que de fato vai ser utilizado para treinar um modelo. O modelo final irá prever os dados desse conjunto a parte e, com isto, pode-se ter uma medida da generalização do modelo final a dados que não lhe foram apresentados durante o treinamento. Chama-se este conjunto a parte de conjunto de teste. (RUSSELL; NORVIG, 2021; MONARD; BARANAUSKAS, 2003)

Alguns algoritmos de aprendizado, para medir o quanto uma hipótese está em concordância com os dados de treinamento, utilizam o que se chama de função de custo (*loss function*). Ela consiste em uma função na forma $\ell : \mathcal{H} \times Z \to \mathbb{R}_+$, o qual Z é o conjunto de dados do domínio em questão, sendo $Z : X \times Y$. Quanto menor o valor de ℓ, melhor a hipótese em questão para o conjunto de treinamento. A função de custo permite uma abordagem de aprendizagem geral que engloba tanto tarefas de problemas de classificação e regressão, tendo apenas que considerar uma função que possa fornecer uma distinção entre uma hipótese boa e ruim. (SHALEV-SHWARTZ; BEN-DAVID, 2014)

Normalmente, para se obter uma medida geral considerando todo o conjunto de treinamento T, o algoritmo de aprendizado busca a hipótese que obtém o menor valor possível do risco empírico (*empirical risk*) em T, que consiste na média dos valores retornados pela função de custo considerando cada exemplo de treinamento, dado pela Equação 2.1. (SHALEV-SHWARTZ; BEN-DAVID, 2014)

$$L_T(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(h, z_i) \tag{2.1}$$

2.2 Redes Neurais Artificiais

Redes Neurais Artificiais (RNAs) é um assunto bastante estudado quando, principalmente, adentramos sobre Aprendizado Profundo (*Deep Learning*). Aprendizado Profundo é a área na qual se tem a ideia de construir modelos que consigam aprender por si mesmos atributos complexos e latentes dos dados de um problema, os quais, na maioria das vezes, não se tem uma relação direta dos seus atributos relevantes. (GOODFELLOW; BENGIO; COURVILLE, 2016)

Além disso, modelos de Aprendizado Profundo consistem na ideia de que conceitos complexos podem ser aprendidos através de conceitos mais simples que vão se complementando de forma sucessiva, gerando a cada etapa uma representação diferente dos dados. Esta abordagem normalmente resulta em várias camadas de processamento os quais cada uma é responsável por trabalhar um determinado conceito que faz sentido ao problema. Conceitos complexos resultam em várias camadas (o que deriva o termo "profundo"). Em evidência disto, as RNA's são o tipo de modelo que se aproxima desta ideia e por isso muito dos modelos de Aprendizado Profundo se utilizam de RNA's. Facilmente vemos aplicações nas mais diversas áreas de grande interesse, como, por exemplo, Processamento de Linguagem Natural e Visão Computacional. (GOODFELLOW; BENGIO; COURVILLE, 2016)

O tema de RNA's é um subtópico de Aprendizado de Máquina, e Aprendizado de Máquina é uma subárea da Inteligência Artificial (IA). As RNA's deram início as discussões que culminaram no começo da área da IA. Um dos objetivos da IA é construir abordagens e máquinas capazes de obter um certo nível de inteligência para solucionar tarefas que, até então, são simples para seres humanos. Sendo assim, estudar e tentar entender os tópicos que constituem e permitem a nossa inteligência é um dos caminhos seguidos para adquirir inspirações em desenvolver soluções computacionais que atendam a estas tarefas. (RUSSELL; NORVIG, 2021)

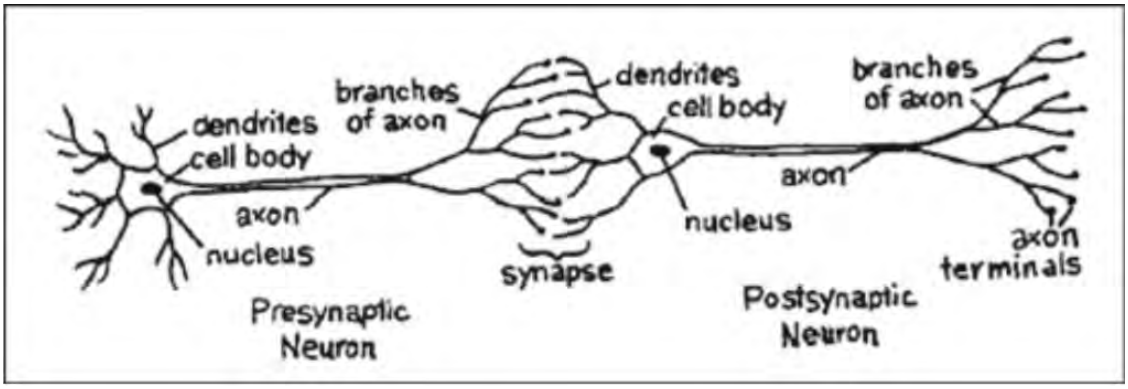


Figura 2.2: O neurônio biológico e um exemplo de conexão com outro neurônio. Os detritos (*dendrites*) recebem os estímulos externos e repassam para o corpo celular (*cell body*), onde é feito o "cálculo" considerando todos os estímulos recebidos. Após isso, o resultado é passado ao longo do axônio (*axon*) até que as vias finais do axônio (*branches of axon*) repassem esses estímulos a outros neurônios, o que caracteriza a sinapse (*synapse*). Imagem retirada de Aggarwal (2018).

O elemento crucial que torna o ser humano inteligente é o cérebro. Nele as RNAs tiveram a sua grande inspiração. O cérebro humano é um computador complexo, paralelo e não-linear, assim definido em Haykin (2007). (BRAGA; FERREIRA; LUDERMIR, 2007; HAYKIN, 2007)

O neurônio é a célula nervosa que, em conjunto com vários outros, formam esta rede que nos proveem o raciocínio e inteligência. Cada neurônio recebe impulsos elétricos (ou nervosos) pelas suas vias de entrada chamadas de dendritos e une todos os sinais recebidos no corpo celular. O resultado desta união é repassado para frente através do axônio. É através das conexões entre os neurônios com emissões de impulsos nervosos, denominadas sinapses, que permitem que os neurônios recebam estímulos que determinarão se irão emitir novos sinais para outros neurônios ou não. Um exemplo simplificado do neurônio e sua conexão com outros é mostrado na figura Figura 2.2. (BRAGA; FERREIRA; LUDERMIR, 2007; HAYKIN, 2007)

A flexibilidade da estrutura neuronal que inclui a redefinição das sinapses entre outros pontos permitem ao cérebro uma capacidade de mudança e, assim, o aprendizado. (BRAGA; FERREIRA; LUDERMIR, 2007; HAYKIN, 2007)

Pode-se definir tecnicamente uma RNA como sendo uma abordagem computacional constituinte de unidades de processamento, denominados neurônios artificiais,

que interconectados possuem o poder de resolver problemas complexos. Este poder é obtido graças a um conhecimento aprendido através de um processo de aprendizado utilizando dados de um determinado contexto. Os neurônios artificias podem ser divididos em camadas para definir uma organização entre eles. (BRAGA; FERREIRA; LUDERMIR, 2007; HAYKIN, 2007)

Nas próximas seções serão explicados com mais detalhes como funciona e como é a arquitetura de uma RNA.

2.2.1 Neurônio Artificial

O modelo estrutural do neurônio artificial conhecido hoje, também conhecido como modelo MCP, foi proposto por McCulloch e Pitts (1943) em uma tentativa de representar o neurônio biológico em uma estrutura artificial.

O processamento em um neurônio artificial é feito da seguinte forma: os sinais de entrada $x_1, x_2, x_3, ..., x_n$, vindos de outros neurônios ou então da entrada em si da rede, são recebidos pelo neurônio. Cada um destes sinais possui um peso sináptico associado $(w_1, w_2, w_3, ..., w_n)$ que determinará o nível de influência daquele sinal no neurônio em si. O somatório representa a etapa em que a soma dos produtos entre os sinais de entrada e dos pesos sinápticos é calculada. A última etapa do processamento pega este resultado do somatório e aplica uma função para gerar o resultado final y_k. A Figura 2.3 mostra graficamente o modelo do neurônio artificial.

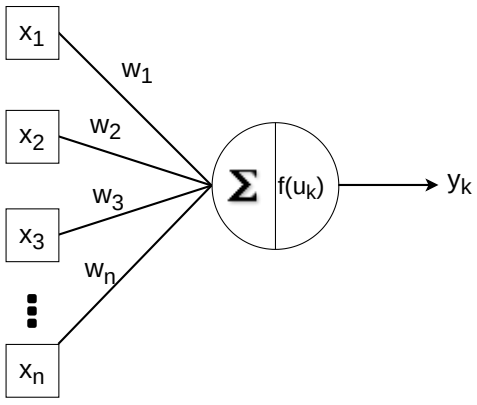


Figura 2.3: Modelo do neurônio artificial.

A função aplicada denomina-se função de ativação, e previamente na modelagem da rede deve-se definir qual tipo de função a ser utilizada como função de ativação do neurônio. A Equação 2.2 representa matematicamente, para um neurônio k na rede, o cálculo do resultado final y_k usando uma função de ativação f.

$$y_k = f(u_k) = f(\sum_{i=1}^{n} w_i x_i) \tag{2.2}$$

O somatório vem da ideia biológica da soma dos estímulos recebidos pelo neurônio de outros através das sinapses. A função de ativação trás a intuição de "ativar" ou não um estímulo para outros neurônios dado um certo nível de estímulo atual. (BRAGA; FERREIRA; LUDERMIR, 2007; HAYKIN, 2007)

Os pesos sinápticos de uma RNA são os parâmetros que serão ajustados e aprendidos por um processo de aprendizagem e representam a experiência e o conhecimento que a rede obteve. Este conhecimento e experiência são utilizados para realizar as previsões dado o contexto do problema. (BRAGA; FERREIRA; LUDERMIR, 2007; HAYKIN, 2007)

Um parâmetro adicional aplicado no modelo do neurônio artificial é o viés (*bias*). O viés é um artifício matemático que permite o deslocamento vertical no sistema de coordenadas da função linear definida por u_k com variáveis x_i. Este deslocamento impede que a função linear resultante sempre passe pela origem. Isto é representado no modelo do neurônio artificial colocando um parâmetro adicional w_0 que sempre recebe como entrada o valor 1. A Figura 2.4 mostra o efeito do viés. A Figura 2.5 mostra o modelo neuronal com este novo parâmetro e a Equação 2.3 mostra o cálculo com o viés presente. (HAYKIN, 2007)

$$y_k = f(u_k) = f(\sum_{i=1}^{n} w_i x_i + w_0) \tag{2.3}$$

Este modelo de neurônio artificial foi a base para o desenvolvimento do Perceptron, a primeira tentativa bem sucedida de construção de uma RNA. Esta rede será brevemente explicada na próxima seção.

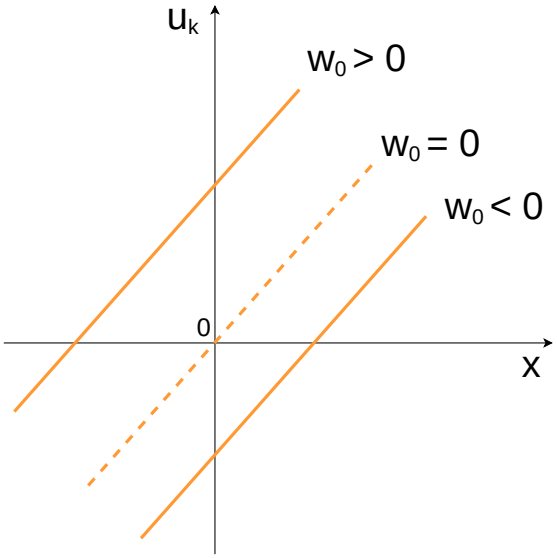


Figura 2.4: Exemplo gráfico do efeito do viés (w_0). Este exemplo considera que o neurônio tenha apenas uma única entrada x, mas a ideia se estende também quando há mais entradas. Imagem baseada em Haykin (2007).

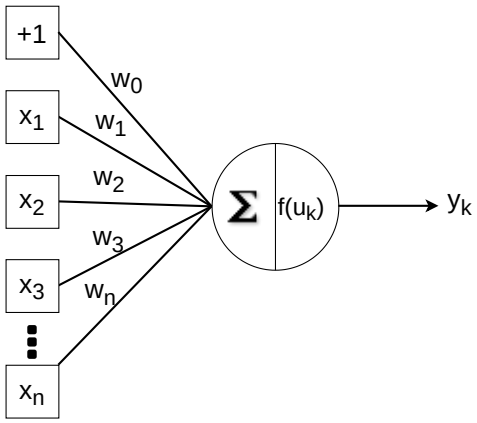


Figura 2.5: Modelo do neurônio artificial com o parâmetro viés.

2.2.2 Perceptron de uma Camada

O trabalho de Rosenblatt (1958) inaugurou o primeiro modelo de RNA funcional. Ele possuía um algoritmo de aprendizado próprio para treinar a rede. Este algoritmo, anos depois, teve a sua convergência provada pelo próprio Rosenblatt, o chamado Teorema de Convergência do Perceptron. (BRAGA; FERREIRA; LUDERMIR, 2007; HAYKIN, 2007; MITCHELL, 1997)

O Perceptron é uma rede de uma única camada de neurônios MCP, que utiliza

uma função de ativação do tipo degrau, definida na Equação 2.4. Um exemplo de uma rede com uma camada é mostrada na Figura 2.6.

$$f(u) = \begin{cases} 1 & \sum_{i=1}^{n} w_i x_i > 0 \\ -1 & \sum_{i=1}^{n} w_i x_i \leq 0 \end{cases} \tag{2.4}$$

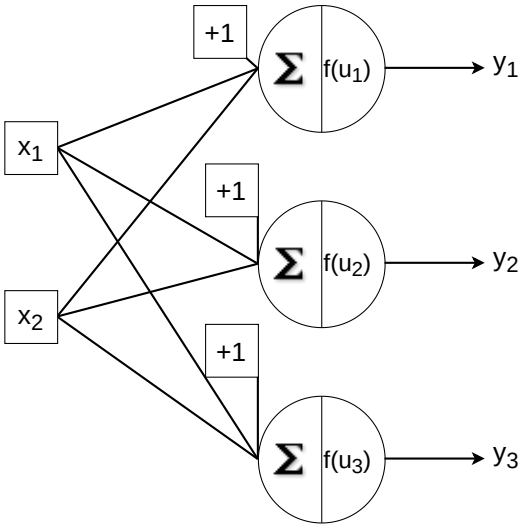


Figura 2.6: Exemplo de uma rede com uma camada com duas entradas e três neurônios.

O algoritmo de aprendizado do Perceptron consiste na atualização iterativa dos pesos associados ao neurônio. A Equação 2.5 e Equação 2.6 representam o cálculo deste aprendizado.

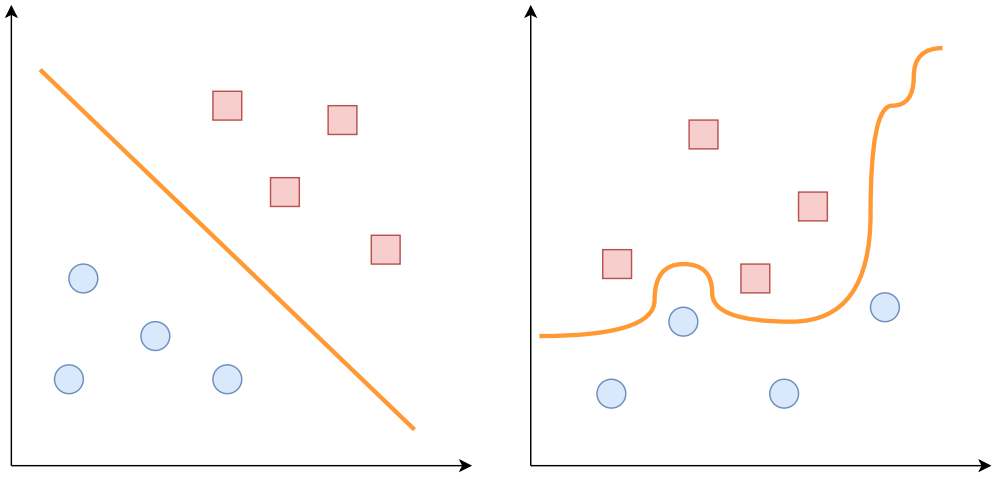
$$w_i \leftarrow w_i + \Delta w_i \tag{2.5}$$

$$\Delta w_i = \eta(y_k - \hat{y}_k)x_i \tag{2.6}$$

A constante η é denominado taxa de aprendizado (*learning rate*) e determina o quanto este peso será atualizado. Valores muito grandes de η podem levar a instabilidades no treinamento e valores muito pequenos uma demora a mais para a convergência. O termo $y_k - \hat{y}_k$ consiste no erro de predição, onde y_k é o valor esperado e $\hat{y}_k$ é o valor da predição atual. O algoritmo de aprendizado é executado

para cada neurônio presente na rede e os pesos sinápticos serão atualizados enquanto houver algum erro de predição diferente de 0. Isto é feito para cada exemplo presente no conjunto de treinamento da rede.

O modelo do Perceptron tem a grande limitação de resolver apenas problemas considerados linearmente separáveis. Um problema é linearmente separável quando é possível dividir os dados do problema por completo utilizando uma reta, de acordo com as suas classes. Esta reta seria o limitador que permitiria classificar corretamente um novo dado na classe correspondente. Esta limitação traz ao Perceptron um escopo reduzido de problemas que podem ser resolvidos por ele, sendo que a maioria dos problemas reais não são linearmente separáveis. Um exemplo visual de como é este conceito é apresentado na Figura 2.7, o qual é usado um exemplo de um pequeno conjunto de dados que possui apenas duas classes, diferenciadas pela cor. (BRAGA; FERREIRA; LUDERMIR, 2007; HAYKIN, 2007; MITCHELL, 1997)



(a) Linearmente Separável (b) Não Linearmente Separável

Figura 2.7: Exemplo de problema linearmente separável e não linearmente separável.

O trabalho do Perceptron ocasionou um reforço bastante positivo e chamou mais atenção para a área de RNAs na época. O trabalho de Minsky e Papert (1969) fez críticas ao Perceptron devido as suas limitações e demonstrou uma visão pessimista com relação as RNAs. Isto acabou resultando na queda do interesse no estudo da área durante a década de 70 e 80. No pós década de 80, surgiram trabalhos inovadores como o do Rumelhart, Hinton e Williams (1986) que ocasionaram o ressurgimento

do interesse na área. (BRAGA; FERREIRA; LUDERMIR, 2007; HAYKIN, 2007)

2.2.3 Perceptron de Múltiplas Camadas

Como comentado anteriormente, um Perceptron com uma única camada possui grandes limitações, e para isso uma rede com mais camadas é necessária. A ideia da construção de redes com mais camadas já existia até então, mas havia o problema em aberto da falta de abordagens e algoritmos de aprendizado funcionais que pudessem lidar com este tipo de arquitetura (HAYKIN, 2007). Com o surgimento do algoritmo de aprendizado de Retropropagação (*Back-propagation*), inaugurou-se uma oportunidade de treinar de fato RNAs com mais de uma camada e permitir a construção destes modelos. (HAYKIN, 2007; BRAGA; FERREIRA; LUDERMIR, 2007)

Uma rede Perceptron de Múltiplas Camadas (*Multilayer Perceptron* - MLP) ou uma Rede Neural Alimentada Diretamente (*Feedforward Neural Network*) é uma arquitetura de RNA composta por mais de uma camada de neurônios. As MLPs possuem uma ou mais camadas ocultas e uma camada de saída que consiste no resultado da rede. É comum que essas redes sejam totalmente conectadas, ou seja, cada neurônio da camada anterior tem uma ligação com cada neurônio da camada logo depois. A Figura 2.8 mostra um exemplo de uma rede com dessa estrutura. (GOODFELLOW; BENGIO; COURVILLE, 2016)

Uma MLP é capaz de resolver problemas não lineares, como já exemplificado em Figura 2.7b. A organização em camadas permite a rede a aplicação de transformações sucessivas no conjunto de entrada em outras representações mais úteis desses dados. Graças a estrutura natural de funções encadeadas que a MLP tem, diversas representações podem ser geradas. Estas representações são codificações internas criadas pela rede que são resultado da combinação das saídas dos neurônios de uma camada anterior nos neurônios da camada atual. Isto concede a rede a habilidade de se trabalhar com uma representação mais acessível a solução do problema em si. Um exemplo esquemático desta ideia pode ser visto na Figura 2.9. (HAYKIN, 2007; BRAGA; FERREIRA; LUDERMIR, 2007; GOODFELLOW; BENGIO; COURVILLE, 2016; MITCHELL, 1997)

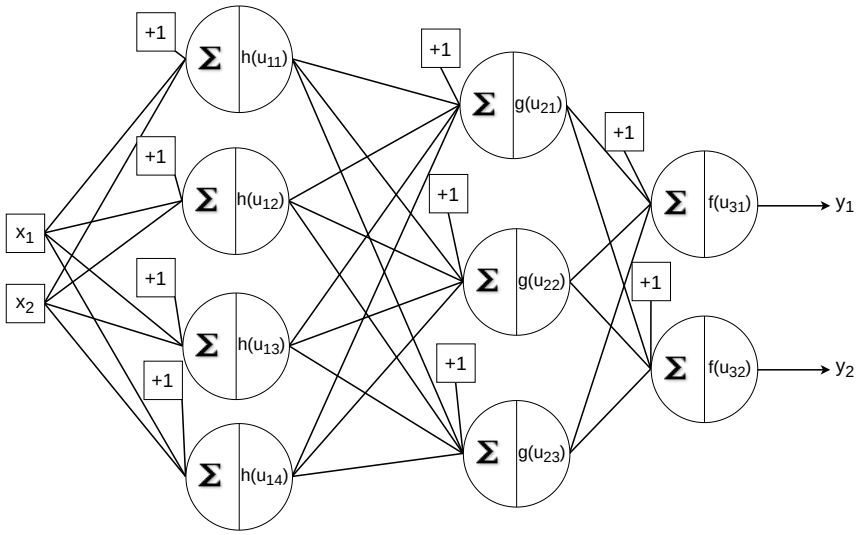


Figura 2.8: Exemplo de uma rede com três camadas. As duas primeiras camadas são as camadas ocultas e a última a camada de saída.

Nessa lógica, podemos dizer também que essas redes funcionam como detectores de características latentes da entrada e elas são usadas para solucionar o problema em questão. Por mais que esses elementos sejam visíveis na rede, o significado real dos resultados intermediários das camadas ocultas é, na maioria das vezes, difícil de se entender. Isto torna as MLPs uma espécie de "caixa preta" quando tentamos interpretar as suas decisões até chegar a solução final. (HAYKIN, 2007; BRAGA; FERREIRA; LUDERMIR, 2007; GOODFELLOW; BENGIO; COURVILLE, 2016; MITCHELL, 1997)

Os trabalhos de Cybenko (1988), Cybenko (1989) e Hornik, Stinchcombe e White (1989) explicitam resultados importantes no contexto das MLP's. Eles dizem que uma rede MLP com uma camada oculta é capaz de aproximar qualquer função contínua e uma rede com duas camadas ocultas é suficiente para aproximar qualquer função. Assim, uma MLP pode ser considerada um Aproximador Universal de Funções. Isto é um chamariz a competência computacional que as MLPs possuem. (BRAGA; FERREIRA; LUDERMIR, 2007; MITCHELL, 1997; GOODFELLOW; BENGIO; COURVILLE, 2016)

A aptidão desse comportamento não linear da rede é permissível apenas se trabalharmos, nas camadas, com funções de ativações não lineares. Não existe

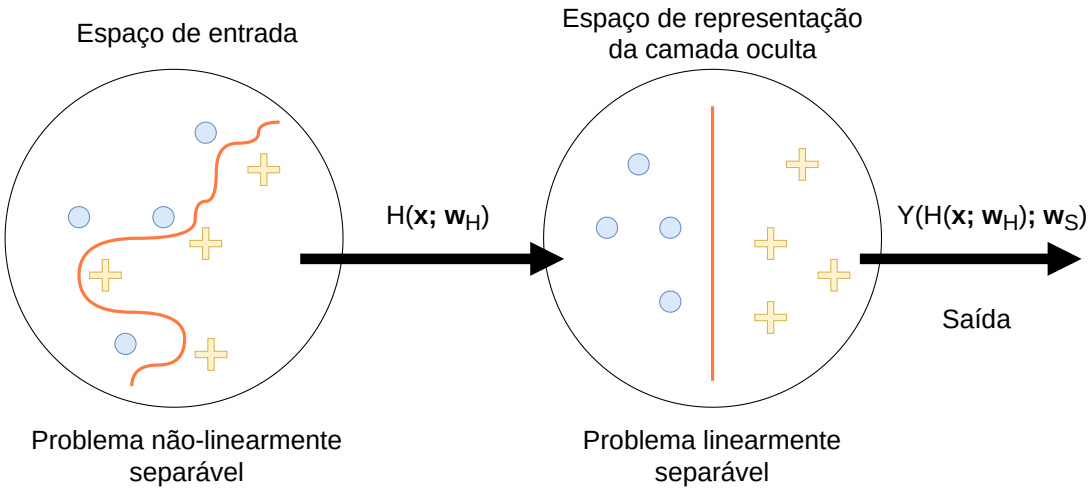


Figura 2.9: Esquema exemplificando o comportamento de uma MLP. Neste caso, a camada oculta da MLP aplica uma transformação nos dados ($H(x; w_H)$) e os mapeia para uma representação em que a camada de saída seja capaz de classificar esses dados ($Y(H(x; w_H); w_S)$). Inspirado e adaptado de Braga, Ferreira e Ludermir (2007).

uma restrição da escolha da função de ativação a ser usada nas camadas, mas há a obrigação de que pelo menos alguma camada oculta tenha funções de ativação não lineares. A aplicação exclusiva de transformações lineares nas camadas resultaria em soluções que só atenderiam casos com características lineares, questões que o Perceptron de uma única camada resolve. (BRAGA; FERREIRA; LUDERMIR, 2007; GOODFELLOW; BENGIO; COURVILLE, 2016)

Uma observação a ser feita é sobre a função de ativação do tipo degrau (Equação 2.4): diferente de um Perceptron de camada única, em uma MLP ela é inviável, pois o resultado de sua derivada em todos os pontos é igual a 0, o que torna inexecutável o treinamento da rede. Isto será esclarecido mais adiante na Subsubseção 2.2.3.1. (BRAGA; FERREIRA; LUDERMIR, 2007; GOODFELLOW; BENGIO; COURVILLE, 2016)

Funções de ativações comuns que são utilizadas: Sigmoide Logística (Equação 2.7), Tangente Hiperbólica (Equação 2.8) e a Unidade Linear Retificada (*Rectified Linear Unit - ReLU*) (Equação 2.9). (BRAGA; FERREIRA; LUDERMIR, 2007; GOODFELLOW; BENGIO; COURVILLE, 2016)

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.7}$$

$$f(x) = tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.8}$$

$$f(x) = max(0, x) \tag{2.9}$$

Quando falamos de MLP, temos algumas decisões adicionais a serem tomadas, como a decisão da quantidade de camadas a serem utilizadas e o número de neurônios em cada uma delas. É importante dizer que ambas as decisões impactarão no nível de generalização do modelo neural, pois afeta o número de pesos sinápticos presentes no modelo. Não há na literatura um método específico que determina de maneira exata a disposição da quantidade de camadas e neurônios para um determinado problema. A experimentação com diferentes parâmetros variando o número de camadas e neurônios ainda é o caminho adotado para a definição heurística de um melhor modelo. (BRAGA; FERREIRA; LUDERMIR, 2007; GOODFELLOW; BENGIO; COURVILLE, 2016)

Um dilema relevante também é a relação direta entre a quantidade de camadas e o número de neurônios. Um problema altamente complexo demandará um modelo igualmente mais flexível para atender a esta complexidade. Mesmo com a prova do Aproximador Universal com apenas duas camadas, poderia ser necessário introduzir muitos neurônios nessas poucas camadas para atender a esta flexibilidade. Uma rede desta magnitude seria custosa tanto para treinamento e também para uso, pelo excesso de computação necessária para efetuar seu trabalho. Como já explicitado anteriormente, as camadas em uma MLP tem um papel importante em realizar transformações nos dados para lidar com esta complexidade e assim resolver o problema. (BRAGA; FERREIRA; LUDERMIR, 2007; GOODFELLOW; BENGIO; COURVILLE, 2016)

Existem alguns trabalhos na literatura, expostos por Goodfellow, Bengio e Cour-

ville (2016), que mostraram experimentalmente que o uso de redes com muitas camadas e menos neurônios nelas demonstra uma melhor capacidade de generalização em várias tarefas do que uma rede com poucas camadas, mas com um número grande de neurônios em cada uma delas. Isto ajuda a fomentar a ideia de que as funções que solucionam os problemas complexos de interesse envolve sempre a composição de várias funções simples, o que caracteriza uma rede MLP. (BRAGA; FERREIRA; LUDERMIR, 2007; GOODFELLOW; BENGIO; COURVILLE, 2016)

2.2.3.1 Retropropagação

O algoritmo de Retropropagação (*Back-propagation*) é um dos principais algoritmos que participa no âmbito do aprendizado destas redes e foi introduzido pela primeira vez por Rumelhart, Hinton e Williams (1986).

O objetivo principal é fornecer uma maneira de calcular a derivada parcial de um peso sináptico com relação a um risco empírico, tendo em conta os erros gerados pelos resultados dos neurônios intermediários das camadas presentes. O uso da derivação é indispensável para o algoritmo de otimização mais comum usado para treinar MLPs, o Gradiente Descendente (GOODFELLOW; BENGIO; COURVILLE, 2016). O algoritmo de otimização, no nosso caso, busca iterativamente o conjunto de pesos os quais resultam no valor mínimo do risco empírico. Em termos matemáticos, se quer calcular:

$$\frac{\partial J(W)}{\partial w}, \tag{2.10}$$

sendo $J(W)$ o risco empírico que recebe todos os pesos sinápticos como entrada (W) e w um peso da rede que precisa ser atualizado.

A Retropropagação é feita para cada exemplo presente na base de treinamento por vez. Assim, na derivação se considera a função de custo aplicada a um único exemplo de treinamento, que representaremos como $J_e(W)$. No fim, essas derivações individuais dos exemplos poderão ser utilizadas para o cálculo da derivada com relação ao risco empírico $J(W)$.

Há duas etapas a serem feitas para o cálculo daquela derivada: a fase de propagar para frente (*forward*) e a fase de propagar para trás (*backward*). A fase *forward* computa o resultado final da rede tendo em conta o dado de treinamento atual e a fase *backward* computa as derivadas de cada peso considerando o erro obtido na saída da rede.

A ideia da Retropropagação é calcular o ajuste dos pesos sinápticos de acordo com o nível de influência que eles tiveram no resultado da MLP, e esta influência está encadeada nas saídas dos neurônios das camadas posteriores em que o valor dele tem dependência direta ou indiretamente. Para computar esta influência, é usado em abundância uma fórmula do Cálculo Diferencial denominada Regra da Cadeia. A Equação 2.11 representa esta regra.

$$\frac{df}{dx} = \frac{df}{dg} * \frac{dg}{dx} \tag{2.11}$$

A Figura 2.10 mostra um esquema exemplo de rede MLP que será usado como base para as próximas explicações da Retropropagação. A partir de agora, será explicado o passo a passo de como a Retropropagação aconteceria de acordo com o esquema da Figura 2.10. Antes de tudo, a fase *forward* é feita e após ela a fase *backward* se inicia. Para o cálculo do peso w_{ji} na camada de saída, tem-se o seguinte:

$$\begin{aligned} \frac{\partial J_e(W)}{\partial w_{ji}} &= \frac{\partial J_e(W)}{\partial out_j} * \frac{\partial out_j}{\partial w_{ji}} \\ &= \frac{\partial J_e(W)}{\partial out_j} * \frac{\partial out_j}{\partial u_j} * \frac{\partial u_j}{\partial w_{ji}} \\ &= \frac{\partial J_e(W)}{\partial out_j} * f'(u_j) * out_i, \end{aligned} \tag{2.12}$$

sendo $f'(u_j)$ a derivada da função de ativação f com relação a u_j.

Aplicando a Regra da Cadeira, transformamos $\frac{\partial J_e(W)}{\partial w_{ji}}$ de uma forma em que se pode calcular essas derivadas mais fácil e diretamente, pois cada derivada tem o termo ao qual está derivando explícito na função do qual está relacionado. O valor

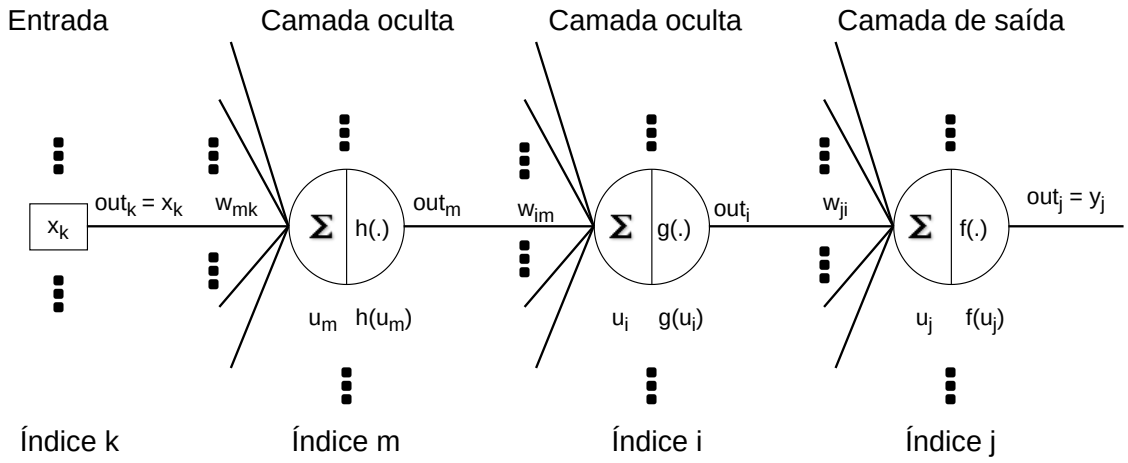


Figura 2.10: Esquema de rede MLP exemplo com índices associados a cada camada. As variáveis *outs* dos neurônios serão importantes para fase *backward* e cada w destacado no esquema sofrerá influência direta do *out* anterior correspondente. Inspirado e adaptado de Braga, Ferreira e Ludermir (2007)

$\frac{\partial J_e(W)}{\partial out_j}$ dependerá de qual $J_e(W)$ será escolhido para treinar a rede.

Agora para o cálculo de um peso w_{im} da camada oculta, tem-se o seguinte:

$$\begin{aligned}
\frac{\partial J_e(W)}{\partial w_{im}} &= \frac{\partial J_e(W)}{\partial out_i} * \frac{\partial out_i}{\partial w_{im}} \\
&= \frac{\partial J_e(W)}{\partial out_i} * \frac{\partial out_i}{\partial u_i} * \frac{\partial u_i}{\partial w_{im}} \\
&= \sum_j \left(\frac{\partial J_e(W)}{\partial u_j} * \frac{\partial u_j}{\partial out_i} \right) * \frac{\partial out_i}{\partial u_i} * \frac{\partial u_i}{\partial w_{im}} \\
&= \sum_j \left(\frac{\partial J_e(W)}{\partial out_j} * \frac{\partial out_j}{\partial u_j} * \frac{\partial u_j}{\partial out_i} \right) * \frac{\partial out_i}{\partial u_i} * \frac{\partial u_i}{\partial w_{im}} \\
&= \sum_j \left(\frac{\partial J_e(W)}{\partial out_j} * f'(u_j) * w_{ji} \right) * g'(u_i) * out_m,
\end{aligned} \tag{2.13}$$

sendo j o índice do neurônio da camada de saída.

A ideia deste somatório é expressar que a saída out_i tem influência na alteração dos resultados de todos os neurônios na camada posterior que, neste caso, é a camada de saída. Se for computar o peso w_{mk} de outra camada oculta:

$$
\begin{aligned}
\frac{\partial J_e(W)}{\partial w_{mk}} &= \frac{\partial J_e(W)}{\partial out_m} * \frac{\partial out_m}{\partial w_{mk}} \\
&= \frac{\partial J_e(W)}{\partial out_m} * \frac{\partial out_m}{\partial u_m} * \frac{\partial u_m}{\partial w_{mk}} \\
&= \sum_i \left(\frac{\partial J_e(W)}{\partial u_i} * \frac{\partial u_i}{\partial out_m} \right) * \frac{\partial out_m}{\partial u_m} * \frac{\partial u_m}{\partial w_{mk}} \\
&= \sum_i \left(\frac{\partial J_e(W)}{\partial out_i} * \frac{\partial out_i}{\partial u_i} * \frac{\partial u_i}{\partial out_m} \right) * \frac{\partial out_m}{\partial u_m} * \frac{\partial u_i}{\partial w_{mk}} \\
&= \sum_i \left(\sum_j \left(\frac{\partial J_e(W)}{\partial out_j} * f'(u_j) * w_{ji} \right) * \frac{\partial out_i}{\partial u_i} * \frac{\partial u_i}{\partial out_m} \right) * \frac{\partial out_m}{\partial u_m} * \frac{\partial u_m}{\partial w_{mk}} \\
&= \sum_i \left(\sum_j \left(\frac{\partial J_e(W)}{\partial out_j} * f'(u_j) * w_{ji} \right) * g'(u_i) * w_{im} \right) * h'(u_m) * out_k,
\end{aligned}
\tag{2.14}
$$

sendo i o índice de um neurônio da camada oculta posterior.

Aqui percebe-se de fato que o elo que conecta essa influência nas camadas posteriores é o termo $\frac{\partial J_e(W)}{\partial out}$, que permite relacionar os valores obtidos das influências anteriores.

Até que se chegue na última camada de trás para frente, este mesmo raciocínio é aplicado. Com isto, é possível calcular todas as derivadas relacionadas aos pesos sinápticos e depois usá-los para alterar o valor atual dos pesos sinápticos da MLP. Até que um critério de parada no algoritmo de treinamento seja atendido, este processo de *forward* e depois o *backward* é executado.

2.3 Ruído de Classe

Quando há a necessidade de se construir um modelo de Aprendizado de Máquina, um dos requisitos é a obtenção de dados de um determinado contexto para a construção de um conjunto de dados (*dataset*). Este conjunto de dados é essencial para uso no treinamento de modelos de Aprendizado de Máquina, para que eles aprendam os padrões corretos e possam fazer as melhores predições. Por isso, é importante que estes dados sejam livres de quaisquer erros ou inconsistências que possam afetar o

registro salvo nestes dados. Quando se levanta esta questão, a qualidade destes dados é questionada e nisso o tópico sobre ruído nesses dados é discutido. Na literatura, os problemas de classificação que envolvem o Aprendizado Supervisionado são os mais explorados nesse contexto do ruído. (FRÉNAY; VERLEYSEN, 2013; NIGAM; DUTTA; GUPTA, 2020; HAN et al., 2020)

O processo de aquisição de dados é um passo naturalmente custoso em questão de recursos e tempo, enviesado e sujeito a erros. Com a crescente demanda por grandes conjuntos de dados, este problema torna-se mais relevante. Alguns estudos estimaram que, mesmo em ambientes com bastante controle para evitar erros no processo, há pelo menos 5% de erros presentes no conjunto de dados final (WU, 1995; ORR, 1998; MALETIC; MARCUS, 2000). Mais ainda, o ruído pode impactar drasticamente os modelos de Aprendizado de Máquina para classificação, aumentando a sua complexidade e deteriorando a performance de sua predição em novos dados (LORENA; CARVALHO, 2004). Usado em ambientes com aplicações críticas, o uso desses modelos podem causar problemas de confiabilidade e segurança do sistema (STRONG; LEE; WANG, 1997).

Uma das características importantes que surgem devido ao ruído é o quanto um algoritmo de aprendizado é robusto. Robustez é a capacidade do algoritmo de criar modelos que sejam insensíveis ao exemplos ruidosos e consequentemente não sofram tanto com o impacto negativo do ruído. Algoritmos robustos criam modelos mais próximos dos modelos ideais, como se fossem treinados em um conjunto de dados com ausência de ruído. (HUBER, 2004)

O tema sobre ruído começou a ser reconhecido e justificado como uma linha de pesquisa relevante no artigo de Angluin e Laird (1988), o qual o mesmo questiona como um algoritmo de aprendizado pode lidar com dados de treinamento incorretos. Além disso, afirmou que é possível um algoritmo de aprendizado lidar com dados incorretos.

O trabalho de Angluin e Laird (1988) foi um passo importante para abrir uma linha de pensamento que contesta se os dados estão corretos ou não, o que reflete em um cenário real que constantemente há erros ou imperfeições nos dados obtidos.

Apoiado a isto, já existem várias abordagens desenvolvidas na literatura que tentam lidar com o problema de ruído nos dados, alguns tendo bons resultados (FRÉNAY; VERLEYSEN, 2013; HAN et al., 2020).

Pode-se definir o ruído como qualquer coisa que obscureça a relação entre os atributos de uma instância no conjunto de dados com a sua classe correspondente (HICKEY, 1996). Devido a isto, relações ou padrões importantes dos atributos com sua classe são ocultados durante o aprendizado do modelo de classificação, trazendo consequências ao seu resultado final.

O ruído pode ser separado em duas categorias: o Ruído de Atributo e o Ruído de Classe (ZHU; WU, 2004). O Ruído de Atributo afeta os valores reais dos atributos do conjunto de dados, enquanto o Ruído de Classe altera as classes observadas dos exemplos dentro do conjunto de dados. É plausível que um possa existir e o outro não, e também de ambos ocorrem ao mesmo tempo.

Quando há a comparação entre qual tipo de ruído é mais danoso, o Ruído de Classe mostra-se superior neste aspecto, o que aumenta a importância de tratá-lo. Isso pode ser explicado pelo fato de normalmente termos muitos atributos e apenas uma classe associada ao exemplo e também que a importância de cada atributo para o aprendizado é diferente, enquanto que as classes sempre terão relevância, pois são elas que deverão ser preditas pelo classificador. (ZHU; WU, 2004; SÁEZ et al., 2014)

Considerando todos os pontos levantados até o momento, podemos ratificar o tópico sobre ruído da seguinte maneira: O ruído é um erro que mascara a real relação dos atributos com a classe da instância, impedindo o aprendizado efetivo desta relação. Sendo assim, é importante criar soluções que lidem com este fator antes, durante ou após o processo de construção do modelo de Aprendizado de Máquina. A Figura 2.11 sumariza o relacionamento do ruído com os demais elementos do Aprendizado Supervisionado.

Como conseguinte, nas próximas seções serão explicados com mais detalhes pontos relevantes sobre o Ruído de Classe, como suas causas, consequências e algumas soluções para lidar com ele, além de alguns pontos essenciais para melhor

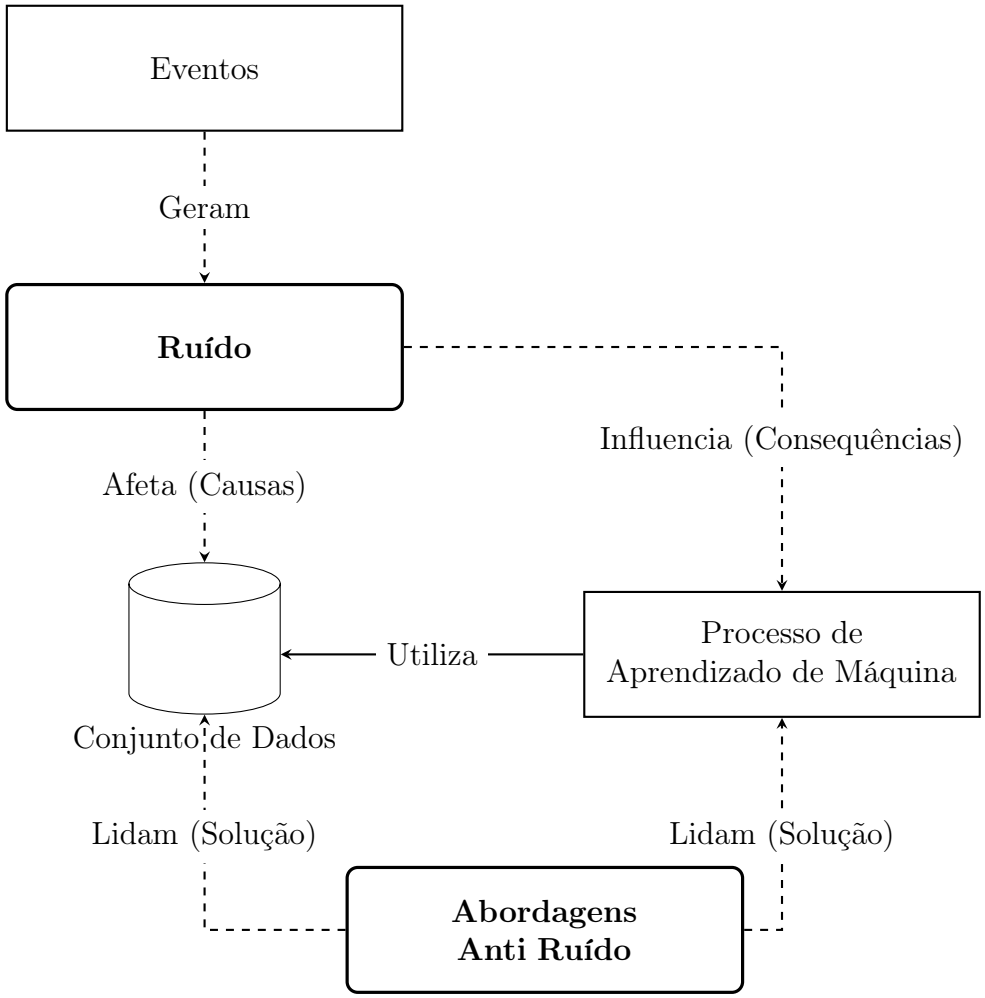


Figura 2.11: Diagrama relacionando os principais pontos quando entramos no tema sobre ruído.

entendimento sobre o ruído.

2.3.1 Causas do Ruído de Classe

Em muitos contextos, a rotulação das classes vinda de um especialista humano é crucial, e justamente por esta intervenção humana o ruído ocorre de forma natural. De acordo com Frénay e Verleysen (2013), existem quatro causas comuns o qual o Ruído de Classe acontece, e que serão explicitadas nos próximos parágrafos.

A primeira causa se refere a insuficiência da informação dada ao especialista para que o mesmo realize uma rotulação correta. Como exemplo, no domínio de aplicações médicas os resultados de alguns testes médicos podem ser desconhecidos, inerente a dificuldade ou impossibilidade de obtê-los por questões de custo ou tempo

(PECHENIZKIY et al., 2006). Em algumas situações, a informação está em péssima ou com alguma variação em sua qualidade, como em casos os quais as imagens não estão com uma resolução boa o suficiente para distinguir certas características que podem ser relevantes na rotulação da mesma em alguma circunstância.

A segunda causa se refere ao próprio especialista errar a rotulação. A coleta de muitos dados com uma confiança extremamente alta é bastante custosa e com uma alta demanda de tempo, e por isto é comum a adoção de alternativas mais baratas e fácies utilizando procedimentos os quais vários outros não especialistas possam rotular os dados, conhecido como *crowdsourcing*. Um exemplo de ferramenta que adota esta ideia é o *Amazon Mechanical Turk*[1]. Classes identificadas por estes indivíduos são menos confiáveis, mas a abundância destas classes podem aliviar o problema de se ter poucos dados confiáveis a disposição (SNOW et al., 2008).

A terceira causa se refere a variabilidade da rotulação final de cada especialista referente ao mesmo dado, o qual ressalta-se a subjetividade de cada especialista, causando dúvidas da rotulação correta. Pode-se ver esses problemas em aplicações médicas, como, por exemplo, numa análise de eletrocardiograma os especialistas raramente concordam sobre os limites exatos dos padrões de sinal (HUGHES; ROBERTS; TARASSENKO, 2004).

A quarta causa se refere a problemas provenientes de codificação dos dados ou de comunicação. Como exemplo, temos que um usuário pode acidentalmente clicar em um e-mail definindo-o como *spam*, ou então pela falta de entendimento do mecanismo de *feedback* acaba marcando o e-mail como spam, trazendo uma rotulação final incorreta (SCULLEY; CORMACK, 2008).

2.3.2 Consequências do Ruído de Classe

Como já brevemente apresentado, o Ruído de Classe tem seu principal motivador de estudo justamente a consequência que ele causa: impactos diretos nos algoritmos e modelos de classificação, além de impactar estatísticas relevantes feitas baseados nesses dados ruidosos. De acordo com Frénay e Verleysen (2013), pode-se dar uma

[1] <https://www.mturk.com>

visão geral sobre essas consequências negativas focando em cinco fatos, que serão explicitados nos próximos parágrafos. Estas podem não ser as únicas existentes, mas são as mais relevantes.

A primeira consequência é a diminuição da performance de predição dos classificadores, ou seja, a acurácia do modelo treinado na presença de ruído é menor comparado a um modelo treinado nos dados sem ruído. Alguns trabalhos realizaram um estudo teórico formal que demonstra esse impacto em modelos simples, como os de regressão logística e os baseados no algoritmo k vizinhos mais próximos (*k-nearest neighbors* - kNN) (FRÉNAY; VERLEYSEN, 2013). Outros trabalhos apresentaram resultados de forma empírica mostrando esta consequência, e que o impacto difere dependendo do algoritmo do classificador (FRÉNAY; VERLEYSEN, 2013). Por exemplo, o trabalho de Nettleton, Orriols-Puig e Fornells (2010) efetua uma análise do impacto do ruído em quatro algoritmos diferentes: Naïve Bayes, kNN, Árvore de Decisão e Máquina de Vetores de Suporte (*Support Vector Machines* - SVM), concluindo que o algoritmo Naïve Bayes e kNN se mostram menos sensível ao ruído e com sensibilidades parecidas na maioria dos casos, seguidos pela Árvore de Decisão e o SVM como o mais sensível.

A segunda consequência se refere ao Ruído de Classe afetar alguns requisitos de aprendizagem necessários para se obter o melhor modelo possível. Por exemplo, o trabalho de Angluin e Laird (1988) expõe que a presença de ruído uniforme dentro do *framework* Provavelmente Aproximadamente Correto (*Probably Approximately Correct* - PAC) aumenta o número de exemplos necessários no conjunto de treinamento para se identificar a hipótese PAC que vai caracterizar o modelo final. O aprendizado PAC, resumidamente, é um conceito matemático formal vindo da Teoria da Aprendizagem que formaliza a possibilidade de um algoritmo de aprendizado, com um determinado conjunto de dados de treino, consiga retornar uma hipótese que tem uma alta probabilidade de ser a hipótese que provê um baixo erro de generalização, sendo ela chamada de hipótese PAC (VALIANT, 1984).

A terceira consequência é o aumento da complexidade dos modelos inferidos, ou seja, a geração de modelos com mais parâmetros presentes para armazenar o

aprendizado obtido dos dados, que consequentemente aumenta a flexibilidade desses modelos. Por exemplo, os trabalhos de Brodley e Friedl (1999) e Libralon et al. (2009) mostram experimentalmente um crescimento no número de nós em Árvores de Decisão e no número de vetores de suportes no SVM quando são criados modelos a partir de dados ruidosos.

A quarta consequência é a alteração das frequências das classes presentes nos dados. Frénay e Verleysen (2013) citaram um exemplo prático a respeito de estudos médicos que tem como objetivo medir a incidência de uma doença em uma população e que esses resultados teriam um viés equivocado afetado pelo Ruído de Classe, o que levaria a conclusões incorretas. Além disso, esse desvio das frequências reais influencia negativamente na validação e avaliação dos modelos, através de métricas de performance calculadas em cima de dados de testes ruidosos, o que pode levar a comparações enganosas.

A quinta consequência é a dificuldade na identificação e seleção de atributos relevantes. Por exemplo, o trabalho de Zhang, Rekaya e Bertrand (2006) revelou o impacto negativo do Ruído de Classe na seleção de atributos nos dados de microarranjos de DNA, o que é mais preocupante pelo fato de que há poucos dados de microarranjos disponíveis.

2.3.3 Taxonomia do Ruído de Classe

Para distinguir as diferentes condições de como o Ruído de Classe é gerado no conjunto de dados, Frénay e Verleysen (2013) o classificou em três tipos: Ruído Completamente Aleatório (*Noise Completely at Random* - NCAR), Ruído Aleatório (*Noise at Random* - NAR) e Ruído não Aleatório (*Noise not at Random* - NNAR). Esta classificação fornece possíveis modelos estatísticos de como o Ruído de Classe se apresenta dentro do conjunto de dados.

Nos modelos anteriores, considera-se quatro variáveis aleatórias: Y que consiste na classe verdadeira do dado, X que representa o vetor dos atributos, $\tilde{Y}$ que é a classe observada do dado e E uma variável binária que diz se existe ou não um erro de rotulação, ou seja, se há ruído. É importante notar que a presença da variável $\tilde{Y}$

é indispensável para diferenciar a classe real a qual àquele dado pertence da classe que realmente se mostra a disposição para uso, para contemplar os casos de $Y \neq \tilde{Y}$. A Figura 2.13 ilustra as dependências de cada variável dependendo do modelo.

O NCAR consiste na ocorrência do erro de rotulação (E) independente de qual é a classe verdadeira (Y) e os atributos (X) da instância. Na prática, todas as instâncias sem distinção de classe terão a mesma probabilidade de serem rotuladas erroneamente. Este tipo também é conhecido como ruído simétrico (CORDEIRO; CARNEIRO, 2020).

O NAR consiste na ocorrência do erro de rotulação (E) independente dos atributos da instância (X), mas dependendo da classe verdadeira (Y). Na prática, algumas instâncias de determinadas classes podem ter uma probabilidade maior ou menor de serem rotuladas erroneamente em relação as outras instâncias das demais classes. Por exemplo, os erros de classificação para determinar se uma imagem se refere a uma onça-pintada ou não estariam mais concentrados como sendo da classe do leopardo comparado ao um gato, por exemplo, pois é muito mais fácil confundir uma onça-pintada com um leopardo. O NCAR é um caso especial do NAR, quando todas estas probabilidades são iguais entre si. Este tipo também é conhecido como ruído assimétrico (CORDEIRO; CARNEIRO, 2020).

Além disso, pela dependência por Y, este modelo pode ser representado por uma matriz de transição (Equação 2.15), que expressa as probabilidades de cada combinação possível dos casos de ruído em $\tilde{Y}$ considerando Y:

$$\gamma = \begin{pmatrix} P(\tilde{Y}=1|Y=1) & \dots & P(\tilde{Y}=n|Y=1) \\ \vdots & \ddots & \vdots \\ P(\tilde{Y}=1|Y=n) & \dots & P(\tilde{Y}=n|Y=n) \end{pmatrix}, \tag{2.15}$$

sendo n o número de classes consideradas e que o somatório de cada linha da matriz seja 1, ou seja: $\sum_{\tilde{y}=1}^{n} P(\tilde{Y}=\tilde{y}|Y=y) = 1$.

O ruído uniforme, que representa o caso especial NCAR, é representado na matriz

de transição da seguinte forma:

$$\gamma_{NCAR} = \begin{pmatrix} 1-p_e & \dots & \frac{p_e}{n_y-1} \\ \vdots & \ddots & \vdots \\ \frac{p_e}{n_y-1} & \dots & 1-p_e \end{pmatrix}, \tag{2.16}$$

sendo n_y o número de classes do problema e p_e a probabilidade da classe ser ruidosa.

O NNAR consiste na ocorrência do erro de rotulação (E) dependente da classe verdadeira (Y) e dos atributos (X). Na prática, instâncias presentes em certos locais dentro do espaço de atributos, ou seja, com certos valores e, consequentemente, de determinadas classes terão uma probabilidade maior ou menor de serem rotuladas erroneamente. A Figura 2.12 mostra um exemplo de classificação de imagens sendo cachorros ou não, que em alguns exemplos a tendência de erro na rotulação para a classe lobo pode ser diferente. A imagem do cachorro com uma bola na boca é de autoria própria e as outras foram retiradas da internet[1].

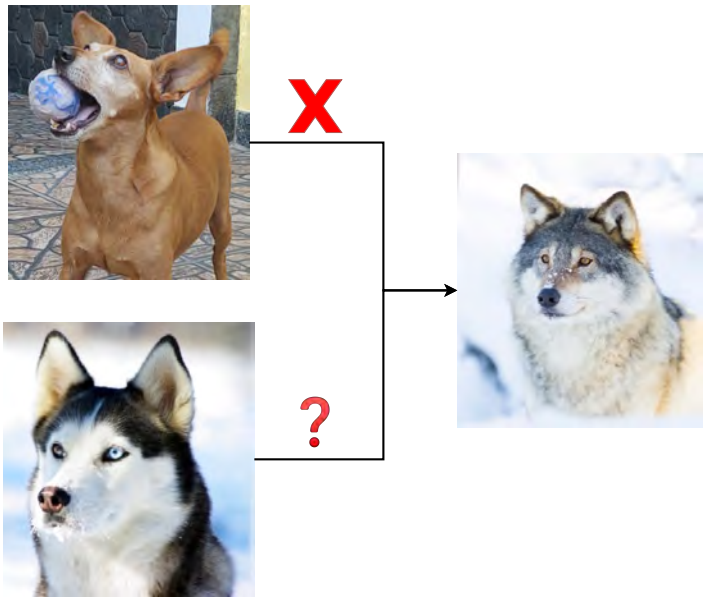


Figura 2.12: Exemplo em que o modelo NNAR é aplicado. Neste caso, o objetivo é classificar imagens de cachorros como sendo cachorros, mas se sabe que alguns cachorros são muito parecidos com lobos e outros não, remetendo que para alguns exemplos o erro para um rótulo como sendo lobo é maior, o que significa dizer que dependendo do atributo da imagem do cachorro, ele pode sofrer um ruído para a classe do lobo.

[1] <https://pt.differbetween.com/article/difference_between_wolf_and_husky>

Este é o cenário mais geral possível, sendo o NCAR e NAR casos especiais do NNAR, quando a influência da posição da instância dentro do espaço de atributos é nula, podendo restringir a análise a apenas ao conjunto das classes do problema. Este tipo também é conhecido como ruído semântico (CORDEIRO; CARNEIRO, 2020) ou como ruído dependente da instância (SONG et al., 2020).

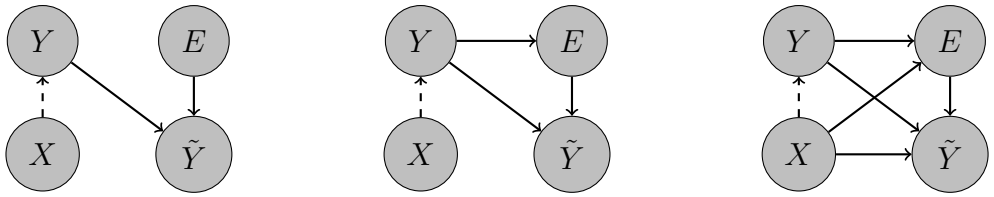


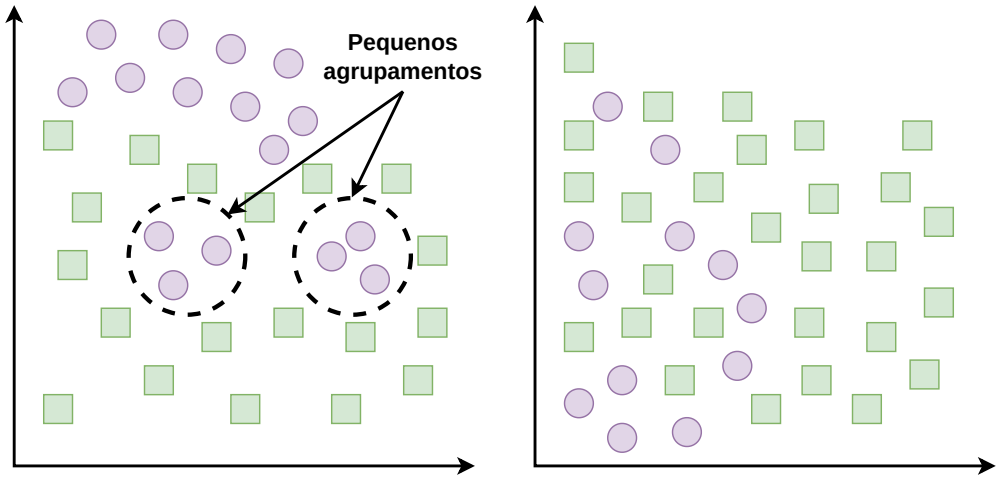
Figura 2.13: Taxonomia do Ruído de Classe proposta por (FRÉNAY; VERLEYSEN, 2013). (a) Ruído Completamente Aleatório (NCAR), (b) Ruído Aleatório (NAR) e (c) Ruído não Aleatório (NNAR). As setas correspondem às dependências estatísticas. A dependência natural entre X e Y foi colocada como uma seta tracejada.

2.3.4 Impactos do Ruído de Classe na Classificação

Como já citado anteriormente, o Ruído de Classe acaba por diminuir a performance dos classificadores, e o motivo direto a isto é a mudança intrínseca que ele provoca na disposição dos dados dentro do espaço o qual eles estão organizados. Esta mudança acaba destoando o problema real de classificação a ser resolvido para um problema não representativo, o que leva a aprendizagens equivocadas por parte dos algoritmos, tornando o desafio mais complexo. (GARCÍA; LUENGO; HERRERA, 2015)

Como concretização dessa disposição, García, Luengo e Herrera (2015) explanaram duas configurações possíveis destas disposições que deixam a classificação mais difícil: a criação de pequenos agrupamentos (*clusteres*) de dados de uma classe particular em uma área de domínio que pertence a uma classe diferente e a sobreposição de dados entre as áreas de domínio das classes. A Figura 2.14 mostra exemplos destes casos.

A sobreposição das classes está inteiramente ligada ao caso de exemplos que estão ao redor e próximos da fronteira de decisão que separa de forma clara estas classes, denominados exemplos de fronteira. Foi identificado que a maioria dos erros



(a) Pequenos agrupamentos (b) Sobreposição entre classes

Figura 2.14: Exemplo dos casos que tornam a classificação mais difícil. Adaptado de García, Luengo e Herrera (2015).

dos classificadores ocorrem em exemplos de fronteira (GARCÍA et al., 2006). A quantidade de exemplos de fronteira presentes contribui bastante para a degradação da performance do classificador e portanto a presença de instâncias ruidosas entre esses exemplos pode agravar ainda mais a degradação, causando um aumento na sobreposição das classes e consequentemente mudando a linha de fronteira do problema real (NAPIERAŁA; STEFANOWSKI; WILK, 2010). A Figura 2.15 mostra um exemplo visual de exemplos de fronteira.

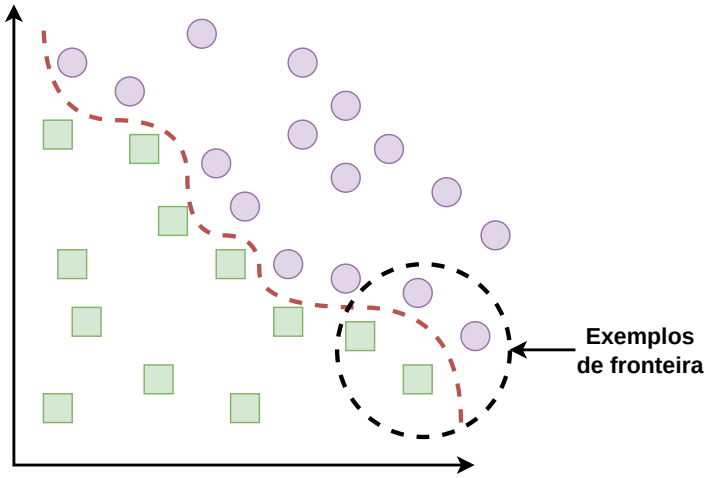


Figura 2.15: Exemplo que contém instâncias de fronteira.

2.3.5 Abordagens para lidar com o Ruído de Classe

Com o problema do Ruído de Classe evidenciado, várias abordagens já foram propostas na literatura para lidá-lo. Como explorado em Frénay e Verleysen (2013), o grupo destas soluções pode ser separado em três categorias: modelos naturalmente robustos ao ruído, métodos de limpeza dos dados e algoritmos tolerantes ao ruído.

A primeira abordagem se baseia na assunção de que os algoritmos de aprendizado terão uma certa robustez a presença de ruído. Esta forma de encarar o ruído é aceita pois é evidenciado que existem algoritmos mais sensíveis ao ruído com relação a outros, como é mostrado em alguns trabalhos experimentais na literatura, que expõem estes algoritmos a presença de ruído e compara-os com relação a sua performance.

A vantagem desta abordagem é a ausência de passos adicionais para se lidar com o ruído, podendo este ser tratado utilizando técnicas que em teoria já seriam usadas durante o processo de aprendizagem, como as estratégias para se lidar com o sobreajuste. A desvantagem seria que esse tipo de tática é efetiva apenas quando há um baixíssimo nível de ruído no conjunto de dados.

A segunda abordagem se baseia na melhora dos classificadores aplicando estratégias para identificar e limpar os exemplos ruidosos no conjunto de dados. Esta limpeza nos dados pode se referir a uma remoção completa desse exemplo do conjunto ou então uma nova rotulação naquele exemplo errôneo, mas a maioria dos trabalhos se envolveram na estratégia de remoção completa, pois aparentemente isto é mais efetivo do que o outro caso. Em adição, estes trabalhos usam de distintas abordagens para lidar com o problema, como o uso de medidas e limiares para identificação, o aproveitamento das predições finais feitas pelo classificador, a análise do impacto no aprendizado que estes exemplos fazem e em métodos baseados em algoritmos de aprendizado conhecidos, como o kNN, *Boosting*, etc.

A grande vantagem da limpeza dos dados é a eliminação completa da influência desses exemplos ruidosos no modelo final obtido, mas há um ponto desvantajoso de que abordagens que resultam em um excesso de remoção de dados no conjunto podem ocasionar uma queda considerável na performance final do classificador, pois o mesmo

seria treinado com pouquíssimos exemplos. Este problema pode ser mais facilmente alcançado quando lidamos com conjuntos de dados desbalanceados, ou seja, os quais a discrepância entre as quantidades de exemplos de determinadas classes é muito alta. Este caso dificulta ainda mais o aprendizado do classificador, pois a probabilidade dos exemplos pertencentes a classes minoritárias de serem removidas no processo de limpeza é muito alta, justamente por existir poucos exemplos destas classes no conjunto de dados.

A terceira abordagem consiste na definição de um método explícito para lidar com o ruído que será atribuído ao algoritmo de aprendizagem escolhido. Neste caso, teríamos uma modelagem de técnicas para capturar ou identificar o ruído que seriam embutidas no passo de aprendizagem do modelo, tornando-o tolerante ao ruído. Frénay e Verleysen (2013) explicitam vários trabalhos que propuseram soluções que entram nesse aspecto, que envolvem técnicas de modelagem probabilística para modelar a presença de ruído e abordagens baseadas diretamente no determinado algoritmo de aprendizado, o qual uma determinada característica específica do algoritmo é adaptada considerando a presença de ruído nos dados, se utilizando dos efeitos de algumas consequências causadas pelo Ruído de Classe como uma base para desenvolver estas adaptações.

A grande vantagem deste tipo de abordagem é a definição clara entre a modelagem do ruído e a modelagem do classificador em si, o que permite um desenvolvimento específico e possibilidade de aprimoramentos, caso necessário, no modelo de ruído para atacar os mais diversos tipos de ruído que podem existir, em contraste com a primeira abordagem explicada anteriormente, que a robustez natural dos algoritmos só consegue lidar com baixos níveis de ruído nos dados. A desvantagem, em contrapartida, é o aumento expressivo da complexidade do aprendizado do modelo final, aumentando a quantidade de parâmetros e passos adicionais para tal objetivo.

Capítulo 3

Uma análise empírica do efeito do Ruído de Classe no aprendizado de Redes Neurais Artificiais

Neste capítulo será apresentado detalhes sobre a base da proposta deste trabalho. Será explicado a motivação e os trabalhos relacionados que o inspiraram, além de expor a proposta geral dele em si.

3.1 Motivação

Assim como já introduzido na Seção 2.3, o empecilho do ruído nos dados traz um problema desafiador no quesito de obter modelos de Aprendizado de Máquina eficientes em relação a sua performance preditiva.

Em um contexto atual de se ter uma grande quantidade de dados a disposição em um curto espaço de tempo, construir conjuntos de dados massivamente grandes se torna cada vez mais fácil, mas rotular essa enorme quantidade de dados pode ser bastante custoso no sentido de geralmente ser necessário uma intervenção de um especialista no domínio para uma rotulação precisa aliado a grande quantidade de dados para passar por essa rotulação, o que consumiria bastante tempo e, dependendo

do caso, não ser escalável, podendo ocasionar na adoção de alternativas de rotulação com não especialistas, como já levantado na Subseção 2.3.1. Sendo um processo árduo esta rotulação, a chance de existir um Ruído de Classe no conjunto de dados é bem grande, fazendo com que ele acabe se tornando mais evidente e um fator preocupante para que se possa garantir uma melhor qualidade dos dados e, consequentemente, dos modelos. Song et al. (2020) apresentaram alguns trabalhos que mostraram uma estimativa da proporção de Ruído de Classe presente nos conjunto de dados reais variando entre 8% e 38,5%.

Nos últimos tempos, abordagens baseadas em Aprendizado Profundo tornaram-se frequentes e eficientes na resolução de problemas em visão computacional, recuperação de informação e processamento de linguagem natural, se utilizando dessa grande quantidade de dados e tendo tido a atenção nesses últimos tempos para o desenvolvimento de soluções para lidar com o Ruído de Classe (SONG et al., 2020; HAN et al., 2020). Um ponto interessante é que estas abordagens usam de RNA's como parte de suas soluções, dando-as uma grande relevância.

Como visto anteriormente na Seção 2.3, uma das abordagens para se lidar com o Ruído de Classe é a adoção da robustez natural inerente dos modelos ao ruído. Dentro deste tópico, vários trabalhos realizaram análises empíricas para prover evidências sobre o nível de robustez natural que estes modelos possuem.

Mesmo que esta abordagem tenha a desvantagem de que esta robustez natural nem sempre é contemplada, é importante realizar trabalhos experimentais deste tipo para reforçar, conhecer, validar e trazer intuições sobre os efeitos do ruído nestes modelos. Experimentos distintos permitem uma testagem em diferentes contextos de como o Ruído de Classe pode atuar nos modelos com diferentes características, deixando mais rica a literatura com dados e evidências do quanto o ruído é danoso no aprendizado destes modelos e como é importante tratá-lo.

Devido a relevância das RNA's e o quanto o Ruído de Classe tem de impacto negativo, o objetivo deste trabalho é oferecer uma análise experimental que forneça uma observação sobre os efeitos do Ruído de Classe no aprendizado das RNA's, o qual diferentes contextos de Ruído de Classe serão considerados e parâmetros da

RNA serão variados para contemplar diferentes RNAs.

3.2 Trabalhos Relacionados

Na literatura, alguns trabalhos tiveram uma proposta experimental para avaliar os efeitos do ruído em modelos de Aprendizado de Máquina e também para RNA's.

O trabalho de Nettleton, Orriols-Puig e Fornells (2010) efetuou uma análise do efeito do ruído em quatro técnicas diferentes: Naïve Bayes, kNN, Árvore de Decisão e SVM. O objetivo principal do artigo foi fornecer uma comparação entre estas técnicas com relação ao seu comportamento na presença de ruído, considerando vários contextos de como o ruído pode se apresentar, e que isto sirva como uma base intuitiva de como estes algoritmos se comportam em um ambiente ruidoso, ao contrário de ser uma conclusão definitiva.

Sobre os conjuntos de dados usados nos experimentos, 6 foram obtidos no *University of California at Irvine (UCI) Machine Learning Repository*[1] e um foi gerado sinteticamente. O conjunto de dados foi construído para ter a garantia de ausência de ruído, algo que em conjuntos de dados reais não é garantido; um nível alto de desbalanceamento entre as classes, ou seja, algumas classes no conjunto de dados terão pouquíssimas ou quase ocorrências nulas; e um atributo redundante, ou seja, ele não é necessário para aprender o padrão desejado, com o objetivo de avaliar o comportamento das técnicas nesses aspectos. Além disso, as bases do UCI contendo mais do que duas classes foram divididas em m bases com 2 classes, onde m é o número de classes. Isto foi feito para ter uma avaliação baseada na performance dos modelos na predição de cada classe específica do problema em questão. Disto, foram criados 9 conjunto de dados.

Dado as questões anteriores, resultou-se no total de 13 conjunto de dados para os experimentos, todos levando em conta 2 classes e separados em 3 grupos: um com bases de alta complexidade dado a quantidade de atributos e as relações entre os mesmos; outro com bases com classes desbalanceadas e o último com os restantes

[1] <https://archive.ics.uci.edu/>

que não englobaram as características anteriores.

Para a geração de ruído, os autores empregaram os dois tipos de ruído: atributo e o de classe. Para o Ruído de Classe, foi aplicada uma geração de ruído uniforme do tipo NCAR, mas com uma condição: caso a instância selecionada para ser ruidosa fosse da classe majoritária, isto é, com mais ocorrências no conjunto de dados, ela seria trocada para a outra classe minoritária. Em outras palavras, apenas as instâncias com a classe majoritária tiveram as suas classes trocadas, enquanto que as instâncias com classe minoritária permaneceram as mesmas.

Para avaliar o efeito do ruído em diferentes situações, o artigo variou os níveis de ruído entre 0%, 10% e 50% e o ruído foi aplicado tanto no conjunto de treino e no de teste. Para o caso do Ruído de Classe, apenas o conjunto de treinamento foi contemplado.

Considerando o Ruído de Classe e a média dos resultados dentre todos os conjuntos de dados, foi observado que com 10% de ruído as técnicas tiveram performances semelhantes, com o Naïve Bayes com uma pequena vantagem e performando melhor que os outros. No caso de 50%, as performances foram bem variadas, seguindo a ordem, do melhor para o pior: kNN, Naïve Bayes, Árvore de Decisão e SVM.

Para evidenciar a análise sobre o efeito do desbalanceamento de classe com a presença de ruído, o artigo selecionou dois conjuntos de dados representativos dentre os 13 totais, sendo que um sofre com o problema de desbalanceamento e o outro não. Para o Ruído de Classe sem o desbalanceamento, todas as técnicas, com exceção do Naïve Bayes, tiveram uma perda de performance considerável, seguindo a ordem, do melhor para o pior: Naïve Bayes, Árvore de Decisão, kNN e SVM. Com o desbalanceamento, os resultados foram semelhantes ao anterior, diferenciando apenas no aumento da diferença de performance entre 0% e 10%, comparado ao resultado anterior. Com isto, é possível induzir que o problema do desbalanceamento não agrava tanto o problema do ruído nos dados.

Tendo em mente todos os resultados dos experimentos, os autores organizaram as técnicas em grupos respeitando a semelhança do comportamento delas dado

os experimentos. No caso do Ruído de Classe, as técnicas Naïve Bayes e kNN permaneceram em um grupo, enquanto que Árvore de Decisão teria um grupo só e o SVM também.

O trabalho de Rolnick et al. (2017) realiza diversos experimentos com o objetivo de analisar o efeito do Ruído de Classe no aprendizado e também a relação do aumento de exemplos não ruidosos usados no treinamento com o nível do impacto do Ruído de Classe. Para os experimentos sobre o efeito do Ruído de Classe, se empregou uma geração de ruído uniforme do tipo NCAR e também um tipo de ruído estruturado definido pelo próprio artigo do tipo NAR.

A geração do ruído estruturado consistiu na ideia de que, dependendo da ordem em que a lista de rótulos ruidosos fosse posta, o primeiro rótulo ruidoso da lista teria uma maior chance de ocorrer com relação aos outros dado o rótulo correto considerado, e as probabilidades dos outros rótulos ruidosos iam diminuindo, seguindo a ordem, de forma linear. O nível do quanto o primeiro rótulo ruidoso seria mais provável é controlado por um parâmetro δ que varia entre 0 e 1; quando $\delta = 0$, seria o equivalente a uma distribuição uniforme entre as probabilidades dos rótulos ruidosos; quando $\delta = 1$, o primeiro rótulo da lista é certamente garantido caso o exemplo do rótulo original seja ruidoso. A diminuição linear dos restantes é obtida baseada neste δ.

Para os experimentos de análise do efeito do ruído, foram usados tanto o ruído uniforme e o ruído estruturado, enquanto que para que os experimentos do aumento de exemplos não ruidosos apenas o ruído uniforme foi considerado. Foram considerados três conjuntos de dados para os experimentos, alguns não entrando em todos eles: MNIST (LECUN, 1998), CIFAR-10 (KRIZHEVSKY; HINTON et al., 2009) e ImageNet (DENG et al., 2009). Foram usados algumas arquiteturas específicas de RNA's: Perceptron, MLP, Redes Neurais Convolucionais (*Convolutional Neural Network* - CNN) e Redes Neurais Residuais (*Residual Neural Network* - ResNet).

Diferente da abordagem usual de aplicar o ruído diretamente nos exemplos já existentes no conjunto de dados, o artigo utilizou da abordagem de preservar os exemplos corretos já existentes no conjunto de treinamento e, para implantar o

ruído, adicionou ao conjunto de treinamento exemplos ruidosos que são idênticos aos exemplos já existentes, mas com os rótulos trocados. Além disso, os autores usaram uma medida de proporção para se referir o quanto de ruído eles estavam considerando no experimento. Por exemplo: para cada um exemplo correto, adiciona-se 50 exemplos ruidosos, que consiste nos mesmos atributos do exemplo correto considerado, mas apenas mudando para um rótulo ruidoso. O objetivo desta abordagem foi de remeter a um cenário o qual tem-se uma abundância de dados disponíveis com uma qualidade ruim de rotulação. Em outras palavras, quando a quantidade de dados ruidosos supera bastante a quantidade de exemplos corretos.

Eles observaram que arquiteturas mais profundas de RNA's tem a tendência de serem mais robustas ao Ruído de Classe, e que, quanto mais há exemplos ruidosos, é necessário uma quantidade maior de exemplos corretos mínimos para alcançar uma determinada acurácia e, quanto maior a acurácia desejada, maior a quantidade de exemplos corretos presentes durante o treinamento.

O trabalho de Rusiecki (2020) busca verificar empiricamente se a técnica de *Dropout* aplicada em RNA's consegue tornar o modelo mais robusto. O trabalho se utilizou de uma geração de ruído uniforme do tipo NCAR nos conjunto de dados MNIST e CIFAR-10, variando a taxa de *Dropout* e o nível de ruído em alguns intervalos específicos. Foram usados duas arquiteturas específicas de CNN's usadas em um artigo do qual o trabalho se baseou, sendo uma arquitetura para cada conjunto de dados. Concluiu-se que o uso de *Dropout* poderia sim ajudar na robustez da Rede Neural de alguma forma quando o conjunto de dados está na presença de ruído.

Dropout é uma técnica aplicada durante o treinamento de RNA's para se evitar o sobreajuste e melhorar a generalização, que, resumidamente, consiste em anular aleatoriamente a influência de alguns neurônios da camada considerada durante as etapas de aprendizado (HINTON et al., 2012). A anulação desta influência implica em uma saída igual a 0 deste neurônio que foi desconsiderado. O evento de um neurônio ser anulado ou não é definido por uma probabilidade desta anulação ocorrer, que é escolhida previamente antes do treinamento da RNA, sendo esta probabilidade a taxa de *Dropout*. Diferentes camadas da RNA podem ter taxas de *Dropout* distintas,

assim como uma camada específica pode ter influência do *Dropout* e outra não, sendo isto um critério de escolha prévia em quais camadas o *Dropout* será aplicado. O efeito do *Dropout* é aplicado apenas durante a etapa de treinamento da RNA, sendo que o modelo final terá todos os neurônios levados em conta.

O trabalho de Algan e Ulusoy (2020) visa analisar o impacto negativo dos três tipos de Ruído de Classe (NCAR, NAR e NNAR) utilizando uma geração de ruído específica para representar cada um deles. Outra contribuição que o artigo traz é uma proposta própria para gerar ruído do tipo NNAR. Dentre os pontos analisados, foram considerados a performance das RNAs na presença de diferentes níveis de ruído nos dados, com a presença ou não de *Dropout* e também com diferentes porcentagens dos dados disponíveis para treinamento, com o objetivo de observar se o aumento na quantidade dos dados de treinamento na presença de ruído melhora ou piora a sua performance. Para os experimentos mostrados no artigo, foram utilizados os conjuntos de dados Fashion-MNIST (XIAO; RASUL; VOLLGRAF, 2017) e CIFAR-100 (KRIZHEVSKY; HINTON et al., 2009), utilizando uma CNN com duas camadas convolucionais e duas totalmente conectadas e um tipo de arquitetura de CNN específica denominada VGG16 (*Visual Geometry Group*), cada modelo aplicado a um único conjunto de dados, respectivamente.

Na geração do ruído, para o NCAR foi empregado o uso do ruído uniforme; para o NAR, foi usado uma estratégia de geração Dependente da Classe (DC) que consiste no uso da matriz de confusão derivada a partir da predição no conjunto de teste para criar a matriz de transição final para representar a distribuição do ruído nos dados.

Para o caso do NNAR, foram adotados dois tipos de geração: Concentrado Localmente (CL) e o Ruído Dependente do Atributo (RDA), sendo este último a geração nova proposta pelo artigo. O CL é uma abordagem proposta por Inouye et al. (2017) que teve a sua inspiração no algoritmo do kNN. A ideia é o seguinte: escolhendo aleatoriamente um exemplo no conjunto de dados de treinamento, é selecionado um k de instâncias próximas dentro do conjunto de instâncias referentes a classe correspondente, e estes k exemplos próximos teriam os seus rótulos corrompidos para uma outra classe diferente da original, escolhida previamente de forma aleatória,

sendo que todas as classes têm a mesma probabilidade de serem escolhidas. Isto é feito considerando todas as classes possíveis do conjunto de dados e até completar, para cada classe, a quantidade desejada de exemplos ruidosos dado pelo parâmetro da taxa de ruído. Desta forma, há vários *clusteres* pequenos de dados ruidosos ao redor do conjunto de dados.

O RDA se baseou na ideia de Hinton, Vinyals e Dean (2015), que se embasa na transferência do conhecimento de uma RNA grande, isto é, com muitos parâmetros para uma RNA menor sem degradar a performance. A proposta do RDA se fundamenta no treinamento de uma RNA no conjunto de dados considerado e depois são extraídos, para cada instância, as saídas finais produzidos pela camada de *softmax* da RNA, que corresponde a uma lista de probabilidades associadas à chance daquela instância ser daquela classe. Estas saídas são chamados de *soft labels* e os rótulos originais que seria uma lista com todos os valores iguais a 0 com exceção a classe correspondente que seria igual a 1, são chamados de *hard labels*.

Uma nova RNA é treinada levando em conta esses *hard labels* e *soft labels* no cálculo da função de custo, com a adição de um parâmetro α que fornece o nível de influência dos valores das *soft labels* no resultado do cálculo final da função de custo. Por último, verificando as novas probabilidades dadas pela camada de *softmax* dessas instâncias, exemplos os quais a maior probabilidade é uma classe diferente da classe original presente no conjunto de dados de treinamento são identificadas, e seriam estes os exemplos com certa similaridade com outras classes além da original. Dado isto, o rótulo do exemplo é corrompido de acordo com a classe mais similar identificada. Este processo é feito até que a porcentagem de exemplos ruidosos desejados dado pelo nível de ruído seja alcançada. O resultado deste tipo de ruído é ter exemplos ruidosos mais concentrados nas linhas de fronteira entre as classes.

Com os resultados experimentais, concluiu-se que o CL e o RDA se mostraram com a menor acurácia no conjunto de teste dentre os tipos de ruído testados e as variações do nível de ruído, sendo que nos testes feitos no Fashion-MNIST o RDA se mostrou mais danoso, enquanto que nos testes no CIFAR-100 o LC foi o mais danoso; o aumento do tamanho do conjunto de dados resultou em um aumento na

acurácia em todos os tipos de ruído, com exceção do RDA; o ruído uniforme e o DC tiveram uma melhora na acurácia com a presença do *Dropout*, enquanto que para o LC e o RDA não houve uma melhora significativa, tendo na maioria dos casos um aumento variando entre 1% a 6%.

Além destas conclusões, houve uma repetição de todos os experimentos realizados em uma MLP com três camadas, para verificar se o impacto do ruído dependeria da arquitetura do modelo. Os resultados foram muito parecidos com os originais, chegando a conclusão que os resultados apresentados podem ser analisados de maneira agnóstica ao modelo.

3.3 Proposta

Na atual seção, será apresentada a ideia geral da metodologia adotada neste trabalho para avaliar o efeito do Ruído de Classe das RNA que, em outras palavras, significa avaliar a robustez da RNA. Antes de tudo, assim como feito nos trabalhos relacionados explicados anteriormente, será escolhido algumas bases de dados para restringir os experimentos e os modelos de RNAs que serão criados. Após isto, uma parte desses dados será dedicado para o treinamento das RNAs, sendo o conjunto de treinamento, e outra parte para futuramente testar os modelos finais e obter as métricas de desempenho dos mesmos, para que possam ser comparados, caracterizando o conjunto de teste.

Antes de efetuar o treinamento dos modelos, uma parte essencial do processo é a geração de ruído artificial no conjunto de treinamento. Será aplicado esta geração em vários níveis de intensidade diferentes para se ter uma maior riqueza de resultados e maior margem para conclusões mais assertivas. Além disso, ruídos do tipo NCAR e NAR serão gerados.

É importante ressaltar que esta forma de trabalho é comum na literatura neste ramo experimental, permitindo uma comparação idônea entre os resultados das performances entre os modelos, pois se terá o caso original e os casos ruidosos presentes na avaliação. Até onde se sabe dado o escopo deste trabalho, não há

trabalhos experimentais que utilizem um conjunto de dados com presença de ruído que não seja gerado artificialmente e compara-os aos resultados obtidos utilizando o conjunto de dados com ausência de ruído. Isso pode ser explicado pela dificuldade na identificação exata se um dado é ruidoso ou não, para que se possa removê-lo da base de dados originalmente ruidosa, com o propósito de ter uma base de controle para avaliar o efeito negativo que o ruído traz.

Obtido os conjuntos de treinamento ruidosos, pode-se iniciar o treinamento das RNAs considerando cada conjunto ruidoso recebido. Lembrando que o caso com o conjunto sem ruído é um caso especial na geração de ruído quando o nível de ruído a ser gerado nos dados é igual a zero. No fim desta etapa, ter-se-á os modelos treinados, cada um representando a situação ruidosa o qual foi submetido. Os modelos resultantes e o conjunto de teste serão utilizados na parte de geração e avaliação dos resultados, testando as performances dos modelos e com os resultados formular uma análise sobre a robustez das RNAs no contexto submetido ao experimento. Por fim, a saída da etapa anterior servirá para tirar conclusões a respeito do comportamento daquele experimento.

Todos os pontos levantados nesta seção e como eles estão relacionados entre si estão sumarizados na Figura 3.1. No próximo capítulo, mais detalhes sobre os experimentos serão apresentados.

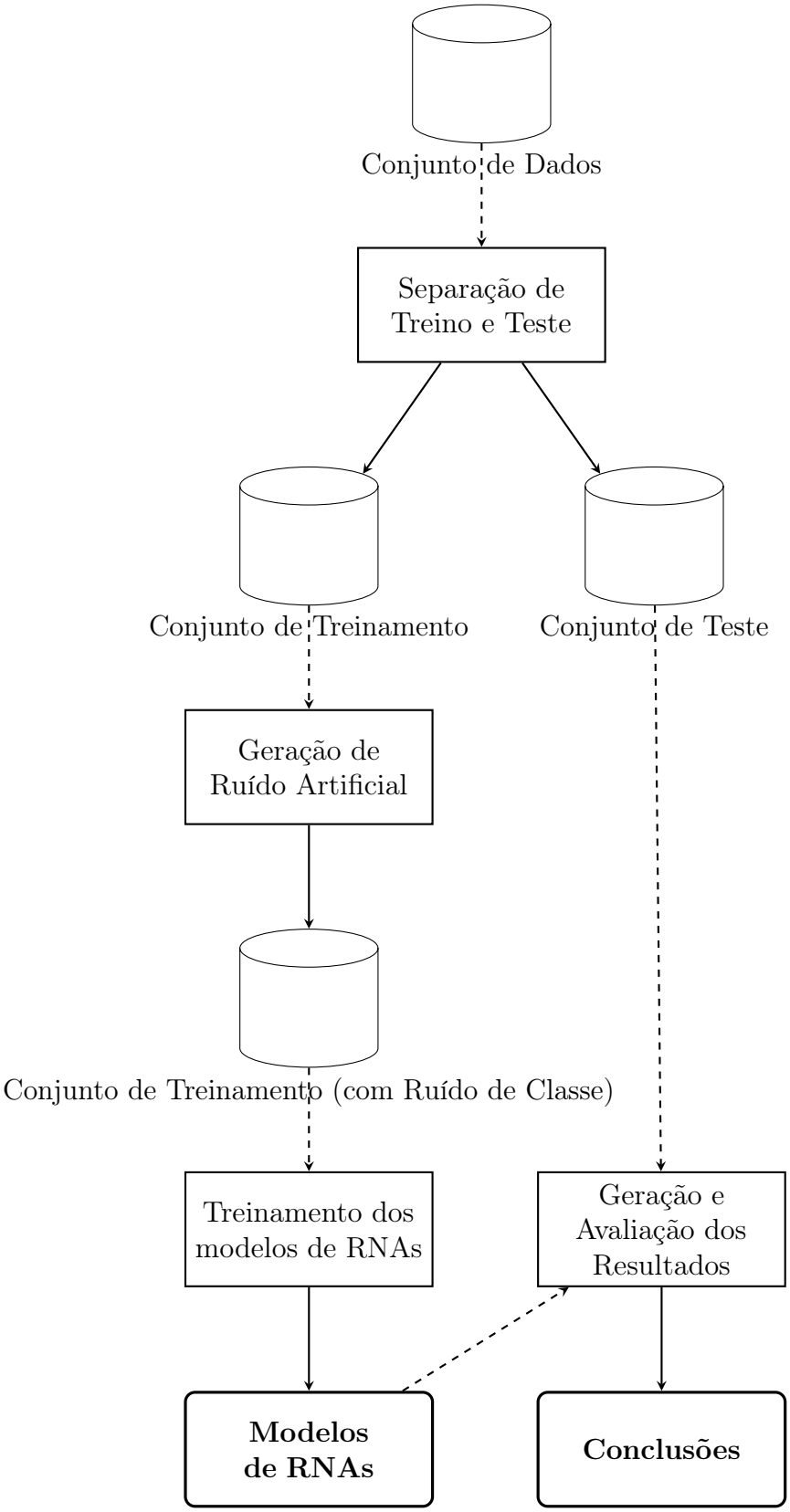


Figura 3.1: Fluxo que descreve a metodologia adotada neste trabalho.

Capítulo 4

Experimentos

Neste capítulo será apresentado detalhes a respeito dos experimentos feitos. Nestes serão incluídos as bases de dados utilizadas, as métricas de avaliação aplicadas, como foi feita a geração de ruído nas bases, a descrição da metodologia e organização dos experimentos e, por fim, a análise de quatro experimentos distintos: o primeiro voltado para a verificação do impacto do Ruído de Classe no treinamento de modelos de RNAs com arquiteturas pré-estabelecidas e o segundo, terceiro e quarto direcionados para o impacto do Ruído de Classe quando há variação na quantidade de neurônios, na quantidade de camadas ocultas e no tipo de função de ativação usada, respectivamente.

4.1 Base de Dados

Existem na literatura algumas bases de dados populares utilizadas para Aprendizado de Máquina e, mais especificamente, para o Aprendizado Profundo. Nesta seção será definido as duas bases de dados usadas neste trabalho: MNIST e Fashion-MNIST.

4.1.1 MNIST

O conjunto de dados MNIST (*Mixed National Institute of Standards and Technology*) foi apresentado por LeCun (1998)[1]. Ele consiste no total de 70000 imagens de dígitos de 0 a 9 escritos a mão, sendo que 60000 foram separadas para o conjunto de treinamento e as 10000 restantes para o conjunto de teste. Cada exemplo do conjunto é uma imagem binária de dimensão 28 x 28, a qual cada pixel varia a sua intensidade de cinza entre 0 e 255. A Figura 4.1 mostra exemplos de dígitos presentes no conjunto de dados.

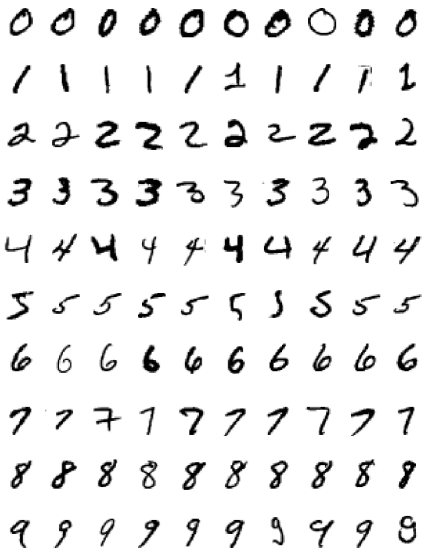


Figura 4.1: Exemplos de 100 imagens vindas do MNIST.

4.1.2 Fashion-MNIST

O conjunto de dados Fashion-MNIST foi apresentado por Xiao, Rasul e Vollgraf (2017)[2], com a proposta de ser uma alternativa ao MNIST. Ele consiste no total de 70000 imagens de produtos de moda retirados do acervo do site da Zalando[3], sendo que 60000 são destinadas para treino e as 10000 restantes para teste. Existem 10 tipos diferentes de produtos presentes no conjunto de dados, variando entre roupas, calçados e bolsas. Assim como no MNIST, cada exemplo é uma imagem binária de dimensão 28 x 28 variando a sua intensidade do pixel entre 0 e 255. A Figura 4.2

[1] <http://yann.lecun.com/exdb/mnist>
[2] <https://github.com/zalandoresearch/fashion-mnist>
[3] <https://zalando.com/>

mostra exemplos de produtos existentes no conjunto de dados.

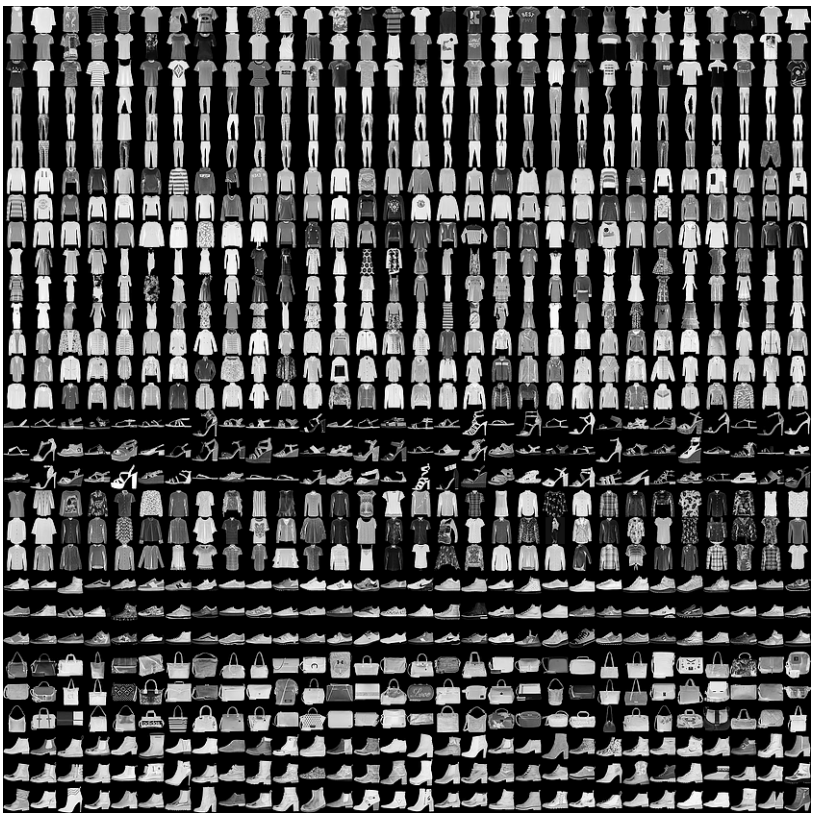


Figura 4.2: Exemplos de imagens vindas do Fashion-MNIST.

4.2 Métricas de Avaliação

Para avaliar o impacto do ruído nos experimentos, será utilizada a acurácia. A acurácia é comumente utilizada na literatura nesse tipo de análise sobre o ruído e para avaliar a performance do modelo final, além de particularmente ter aparecido em todos os trabalhos relacionados explanados anteriormente. A acurácia é definida na Equação 4.1.

$$\text{acurácia} = \frac{\text{Total de predições corretas}}{\text{Total de predições}} \tag{4.1}$$

4.3 Geração Artificial do Ruído

Para simular a presença de ruído nos dados, é comum na literatura destes trabalhos experimentais corromper as bases de dados artificialmente, principalmente quando se trabalha com base de dados definidamente livres de ruído, os quais estão entre os mais utilizados nestes experimentos. O MNIST e o Fashion-MNIST, apresentados anteriormente, são exemplos de conjuntos livres de ruído. (SONG et al., 2020; CORDEIRO; CARNEIRO, 2020). Para a geração de ruído, é comum os trabalhos utilizarem o modelo NCAR e o NAR nos seus experimentos. (FRÉNAY; VERLEYSEN, 2013; PRATI; LUENGO; HERRERA, 2019)

No caso do NCAR, o padrão do ruído uniforme é empregado. Dado uma taxa de ruído p_e, representando a probabilidade de uma classe ser ruidosa, em cada exemplo do conjunto de dados é aplicado um ruído artificial e, caso esse exemplo seja ruidoso, uma nova classe, diferente da classe do exemplo considerado, será escolhida para ser a nova classe do exemplo. Todas as classes tem a mesma probabilidade de serem escolhidas, caracterizando uma probabilidade uniforme. O algoritmo 1 mostra o algoritmo de geração do ruído uniforme.

Algoritmo 1: Geração do ruído uniforme.

Entrada conjunto de dados D, conjunto de classes C, taxa de ruído p_e

Saída conjunto de dados ruidoso $\tilde{D}$.

$\tilde{D} \leftarrow \{\}$

for $(x, y) \in D$ **do**

  Seja $p \in \mathbb{R}$ escolhido aleatoriamente no intervalo $[0, 1]$

  if $p < p_e$ **then**

    Seja $\tilde{y}$ escolhido aleatoriamente no conjunto $C \setminus \{y\}$

    $\tilde{D} \leftarrow \tilde{D} \cup \{(x, \tilde{y})\}$

  else

    $\tilde{D} \leftarrow \tilde{D} \cup \{(x, y)\}$

  end

end

No caso do NAR, o padrão adotado será o mesmo utilizado em Patrini et al. (2017) e Zhang e Sabuncu (2018), podendo defini-lo como um ruído de troca (*flip*

noise). Neste ruído, duas classes distintas c_a e c_b são selecionadas e a classe c_a tem uma probabilidade p_a de sofrer um erro de rotulação como sendo da classe c_b. Com isto, a classe c_a só teria como opção de ruído apenas a classe c_b, enquanto as restantes teriam uma probabilidade zero de serem ruidosas como c_a sendo a classe verdadeira. Esse padrão permite modelar casos reais os quais classes que podem possuir certas semelhanças tem uma tendência de serem rotuladas como sendo da outra classe. O algoritmo 2 mostra o algoritmo de geração do *flip noise*.

Algoritmo 2: Geração do *flip noise*.

Entrada conjunto de dados D, classe c_a, classe c_b, taxa de ruído p_a

Saída conjunto de dados ruidoso $\tilde{D}$.

$\tilde{D} \leftarrow \{\}$

for $(x, y) \in D$ **do**

  if $y = c_a$ **then**

    Seja $p \in \mathbb{R}$ escolhido aleatoriamente no intervalo $[0, 1]$

    if $p < p_a$ **then**

      $\tilde{D} \leftarrow \tilde{D} \cup \{(x, c_b)\}$

    else

      $\tilde{D} \leftarrow \tilde{D} \cup \{(x, y)\}$

    end

  else

    $\tilde{D} \leftarrow \tilde{D} \cup \{(x, y)\}$

  end

end

Como já explanado anteriormente, a geração de ruído do tipo NAR pode ser representando por uma matriz de transição, a qual cada linha representa a classe verdadeira e as colunas a classe observada, e cada elemento é uma probabilidade condicional da classe observada ser uma classe dada a classe verdadeira.

Como o NCAR é um caso especial do NAR, o ruído uniforme pode ser retratado como uma matriz de transição (Equação 2.16). No caso do *flip noise* que é NAR, a matriz de transição pode ser definida como:

$$\gamma_{troca} = \begin{pmatrix} 1-p_1 & p_1 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 1 \end{pmatrix}, \tag{4.2}$$

sendo p_1 a probabilidade de c_1 ser ruidosa e a classe c_2 a ser escolhida no lugar de c_1.

A título de exemplo, o par de classes c_1 e c_2 foram selecionadas, mas qualquer outro par que não seja de classes consecutivas dado a ordem da matriz poderia determinar o *flip noise*, como c_2 e c_4, por exemplo. Além disso, como apenas duas classes tem uma relação ruidosa neste tipo de modelo, as classes restantes serão isentas de ruído e mais de um par pode ser considerado na mesma matriz, como foi feito nos artigos de Patrini et al. (2017) e Zhang e Sabuncu (2018).

O procedimento genérico para geração de ruído se baseia no uso da matriz de transição para obter a distribuição de probabilidade de uma classe específica sofrer um erro de rotulação para alguma outra classe diferente. Isto é obtido acessando a linha da matriz referente a classe do exemplo a gerar o ruído, e esta linha retornaria estas probabilidades. Com elas, a escolha aleatória respeitando-as é feita e assim um retorno da nova classe, podendo ser a mesma que a original, é obtido, e esta será a nova classe do exemplo. Isto é feito iterando sobre todos os exemplos do conjunto de dados, para que todos passem pelo mesmo processo.

Esta geração com a matriz de transição consegue atender as gerações do ruído uniforme e o *flip noise* mostradas nos algoritmos 1 e 2, na devida ordem. O algoritmo 3

estabelece o algoritmo desta geração genérica.

Algoritmo 3: Geração de ruído artificial usando uma matriz de transição.

Entrada conjunto de dados D, conjunto de classes C, matriz de transição T

Saída conjunto de dados ruidoso $\tilde{D}$.

$\tilde{D} \leftarrow \{\}$

for $(x, y) \in D$ **do**

  $probs \leftarrow T[y]$

  Seja $\tilde{y} \in C$ escolhido aleatoriamente seguindo as probabilidade em $probs$

  $\tilde{D} \leftarrow \tilde{D} \cup \{(x, \tilde{y})\}$

end

4.4 Metodologia e Organização dos Experimentos

A metodologia experimental para avaliar a robustez da RNA consistirá de quatro experimentos. O primeiro experimento permitirá observar como o Ruído de Classe impacta negativamente o treinamento de uma RNA em diferentes contextos, caracterizados por diferentes conjunto de dados e a variação do nível de intensidade do ruído nos conjuntos, simulando casos em que o nível de ruído vai crescendo até chegar a um ponto extremo, com modelos sendo treinados com base nesses dados ruidosos.

O segundo experimento permitirá verificar como o impacto do ruído se comportará a medida que o número de neurônios da rede vai aumentando e também quando vai diminuindo. O terceiro experimento busca observar o efeito do ruído quando há um aumento na quantidade de camadas ocultas na RNA. O quarto e último experimento visa conferir se a mudança na função de ativação da rede tem alguma influência na robustez da RNA.

Os conjuntos de dados que serão usados nos experimentos serão o MNIST e o Fashion-MNIST. O nível de ruído será variado de 0% até 90%, em intervalos de 10%. Isso permitirá ter uma visão geral de como a RNA se comportará nos diferentes casos de ruído.

Para se ter uma medida numérica da performance dos modelos, a acurácia será utilizada, aplicada em um conjunto de teste. O conjunto de teste usado será o mesmo

já fornecido pelas referências originais das bases utilizadas e, da mesma forma, o conjunto de treinamento original será usado como o conjunto de treinamento também. Além da acurácia, para viabilizar uma exatidão maior aos resultados, cada contexto terá 10 modelos distintos treinados e a acurácia final do contexto específico será a média aritmética da métrica entre os 10 modelos. É importante fazer isto pois o treinamento de uma RNA poder gerar modelos diferentes, dado a natureza heurística da definição e do processo de treinamento destas RNAs. Para avaliar a variação da acurácia destes 10 modelos, o desvio padrão será calculado também e complementado aos resultados.

Como parte da criação desses modelos, o processo de treinamento é feito e este processo têm alguns parâmetros a serem estabelecidos. No conceito de Aprendizado de Máquina, esses parâmetros que não são obtidos como resultado do treinamento e precisam ser preenchidos previamente antes dele são chamados de hiperparâmetros. Portanto, esta definição será feita nos seguintes hiperparâmetros: otimizador, o tamanho do lote, o número de épocas, a função de custo, a definição de Parada Prematura e o tamanho de um conjunto de validação.

O otimizador baseado em gradiente utilizado neste trabalho foi o Adam (*Adaptive Moment Estimation*) (KINGMA; BA, 2014), atualmente um dos mais utilizados e com melhores resultados em comparação com outros (FATIMA et al., 2020; GOODFELLOW; BENGIO; COURVILLE, 2016). Foi aplicado os valores padrões nos parâmetros do Adam recomendados pelo seu próprio artigo. Além disso, com este otimizador o parâmetro da taxa de aprendizado não precisaria ser ajustado, pois a ideia dele é de ir ajustando dinamicamente este parâmetro durante o treinamento da melhor forma.

O tamanho do lote (*batch size*), corresponde a quantidade de exemplos a serem considerados no conjunto de treinamento para calcular o valor do gradiente para alterar os pesos da RNA. Para este trabalho, foi empregado o valor de 128 para o tamanho do lote, o mesmo usado nos experimentos do trabalho relacionado de Rolnick et al. (2017).

O número de épocas corresponde a quantidade de vezes no treinamento em que

todo o conjunto de treinamento será utilizado para atualizar os pesos da RNA. Para este trabalho, foi adotado o valor de 300 épocas, seguindo a mesma linha de 200 para o treinamento do MNIST em Rolnick et al. (2017) adicionando mais 100, para dar uma margem para o treino usando o Fashion-MNIST. Este valor alto não culminará na criação de modelos com sobreajuste pois o critério de Parada Prematura evitaria isto.

A Parada Prematura (*Early Stopping*) é uma técnica de regularização para lidar com o sobreajuste, por vezes utilizada na literatura. A Parada Prematura consiste no acompanhamento de alguma métrica pré-definida que indicaria que o modelo esteja sofrendo de sobreajuste e quando uma condição relacionada a esta métrica é atendida, o treinamento é interrompido e o melhor modelo encontrado até então é considerado. (PRECHELT, 1998; GOODFELLOW; BENGIO; COURVILLE, 2016). Neste trabalho, foi selecionado um critério de Parada Prematura o qual depois de 20 épocas sem ter alguma melhora na acurácia sobre um conjunto de validação, o treinamento da RNA é interrompido. O conjunto de validação corresponde a 10% do conjunto de treinamento, construído de forma aleatória antes do início do treinamento de fato.

Por último, a função de custo selecionada foi a Entropia Cruzada (*Cross-entropy*), a mesma empregada no artigo de Rolnick et al. (2017). A Tabela 4.1 sumariza a escolha destas variáveis, agrupando os hiperparâmetros citados previamente.

Tabela 4.1: Lista de hiperparâmetros usados nos experimentos.

Hiperparâmetro	Valores
Otimizador	Adam
Tamanho do lote	128
Épocas	300
Função de Custo	Entropia Cruzada
Parada Prematura (Paciência)	20
Conjunto de Validação	10% do Treinamento

Para a geração do ruído *flip noise*, os pares escolhidos foram os mesmos usados nos artigos do Patrini et al. (2017) e Zhang e Sabuncu (2018) para o MNIST e Fashion-MNIST, respectivamente. A Tabela 4.2 mostra os pares de classes afetados

por este ruído.

Tabela 4.2: Pares de classes usadas para a geração do *flip noise*. A notação $a \rightarrow b$ significa que a classe a será trocada pela classe b, e a notação $a \leftrightarrow b$ significa que haverá a troca de a para b e de b para a.

Base de Dados	Pares de Classes
MNIST	$2 \rightarrow 7$ $3 \rightarrow 8$ $7 \rightarrow 1$ $5 \leftrightarrow 6$
Fashion-MNIST	Bota de Tornozelo $\rightarrow$ Tênis Tênis $\rightarrow$ Sandália Pulôver $\rightarrow$ Camisa Casaco $\leftrightarrow$ Vestido

Nas próximas seções, serão detalhados os quatro experimentos feitos neste trabalho, sendo eles:

- O que utilizará de arquiteturas de RNAs pré-estabelecidas, o qual serão escolhidos duas arquiteturas consideradas boas para o MNIST e Fashion-MNIST, respectivamente, e ambas serão treinadas nos dados a medida que o nível de ruído vai aumentando nos casos do ruído uniforme e *flip noise*;

- O que irá aumentar e diminuir o número de neurônios das mesmas arquiteturas utilizadas no experimento de arquiteturas pré-estabelecidas, tanto considerando o MNIST e o Fashion-MNIST nos casos do ruído uniforme e *flip noise* mudando os níveis de ruído;

- O que irá aumentar o número de camadas ocultas mantendo a quantidade de neurônios total que foi definido pela arquitetura de RNA considerada, levando em conta o MNIST e o Fashion-MNIST e também os ruídos uniforme e *flip noise* em diferentes níveis de ruído;

- O que irá alterar a função de ativação da RNA sob as perspectivas das base de dados MNIST e Fashion-MNIST e dos diferentes níveis de ruído nos casos do ruído uniforme e *flip noise*.

4.4.1 Experimento 1: Efeito do Ruído de Classe em RNAs pré-estabelecidas

Como dito anteriormente, o objetivo deste experimento é ter uma evidência do quanto o ruído impacta negativamente nos treinamentos futuros de RNAs com arquiteturas já definidas.

Para simular tal fato, foram escolhidas arquiteturas de MLPs com boas acurácias registradas na literatura de acordo com o conjunto de dados considerado. Como critério adicional de escolha, as acurácias das MLPs selecionadas foram obtidas sem nenhum pré-processamento específico nos dados, de acordo com as informações disponíveis nas referências de onde foram aplicadas. É relevante isto ser levado em consideração para que a margem de erro da acurácia que será encontrada no experimento não destoe tanto da acurácia original.

Para o MNIST, foi escolhida a MLP do artigo Simard et al. (2003), listada no próprio site referência do MNIST (LECUN, 1998). Pela falta de informações no próprio artigo fonte da arquitetura anterior, nas camadas ocultas e na camada de saída foram utilizadas as funções de ativação *ReLU* e *Softmax*, respectivamente, por serem as mais bem consolidadas na literatura para uma escolha padrão (SHARMA; SHARMA; ATHAIYA, 2017; GOODFELLOW; BENGIO; COURVILLE, 2016).

Para o Fashion-MNIST, foi escolhida uma MLPs com melhor acurácia presente na lista de *benchmark* da própria referência da base de dados (XIAO; RASUL; VOLLGRAF, 2017). Assim como na arquitetura do MNIST, tanto a *ReLU* e *Softmax* foram usadas nas camadas ocultas e de saída, respectivamente.

Ambas as arquiteturas são MLPs com uma única camada oculta, tendo diferença apenas na quantidade de neurônios presentes na camada. Os detalhes das MLPs selecionadas estão descritas na Tabela 4.3.

Tabela 4.3: MLPs escolhidas para o experimento de arquiteturas fixas.

Base de Dados	# Neurônios
MNIST	800
Fashion-MNIST	100

Sendo assim, o experimento consiste em treinar os modelos de acordo com as arquiteturas apontadas e com o que foi descrito antes a respeito do treinamento, conseguindo a acurácia e o desvio padrão final para cada nível de ruído no intervalo. Tanto o ruído uniforme e o *flip noise* serão gerados para este experimento, com o *flip noise* considerando os pares, em cada base de dados, conforme apresentados anteriormente.

4.4.2 Experimento 2: Efeito do Ruído de Classe variando o número de neurônios

O objetivo deste experimento é treinar RNAs sob o efeito do ruído mudando a quantidade de neurônios presentes na camada oculta, para se ter a noção de como a performance da rede vai mudar quando se aumenta ou diminui a quantidade de neurônios.

Para alcançar tal objetivo, foram escolhidas as mesmas MLPs do experimento anterior como base para variar a quantidade de neurônios: a do MNIST vinda artigo do Simard et al. (2003) e a do Fashion-MNIST vinda na referência do Xiao, Rasul e Vollgraf (2017). Tanto a $ReLU$ e o *Softmax* foram adotadas na camada oculta e de saída, respectivamente.

Para o MNIST, os modelos foram definidos aumentando e diminuindo em 200 neurônios por vez do número de neurônios da MLP base de 800 neurônios. Para o Fashion-MNIST, os modelos foram definidos aumentando em 200 neurônios por vez também mas diminuindo em 25 neurônios por vez do número de neurônios da MLP base de 100 neurônios. Chegando ao valor de 35 neurônios, foi diminuindo em 5 neurônios até chegar ao valor mínimo de 10 neurônios.

Como se pode observar, devido a MLP base do Fashion-MNIST ter apenas 100 neurônios foi necessário colocar um valor menor de variação para ainda ser possível analisar este caso de decaimento da quantidade de neurônios no caso do Fashion-MNIST. Para se equiparar com a diminuição dos neurônios no Fashion-MNIST, na MLP do MNIST houve o mesmo decaimento na quantidade de neurônios a partir de 100 como ocorreu no caso do Fashion-MNIST. As listas das quantidades de neurônios

usadas estão descritos na Tabela 4.4.

Tabela 4.4: MLPs escolhidas para o experimento de variação do número de neurônios. A primeira lista em cada linha da base de dados corresponde ao evento em que o número de neurônios vai diminuindo, enquanto que a segunda lista corresponde ao aumento do número de neurônios.

Base de Dados	Listas com # Neurônios
MNIST	10, 15, 25, 35, 50, 75, 100, 200, 400, 600, 800 800, 1000, 1200, 1400, 1600
Fashion-MNIST	10, 25, 50, 75, 100 100, 300, 500, 700, 900

Deste modo, o experimento vai treinar todos os modelos possíveis descrito na Tabela 4.4 seguindo as mesmas diretivas para o treinamento explicadas logo no início desta seção, obtendo a acurácia e o desvio padrão final para cada nível de ruído no intervalo. A respeito dos tipos de ruído usados, tanto o ruído uniforme e o *flip noise* serão aplicados, com o *flip noise* considerando os pares, em cada base de dados, conforme apresentados anteriormente.

4.4.3 Experimento 3: Efeito do Ruído de Classe variando o número de camadas

O objetivo deste experimento é treinar RNAs sob o efeito do ruído alterando a quantidade de camadas ocultas presentes, para verificar se o número de camadas influencia de alguma maneira na robustez do modelo.

Como adotado no experimento 1 e no experimento 2, as mesmas MLPs clássicas do Simard et al. (2003) e Xiao, Rasul e Vollgraf (2017) foram usadas como base para o MNIST e Fashion-MNIST, respectivamente. A $ReLU$ e $Softmax$ foram adotadas como as funções de ativações na camada oculta e na camada de saída, nessa ordem.

Foi decidido uma variação de uma até quatro camadas ocultas para as MLPs de cada base de dados. Para evitar resultados fora do escopo original do modelo base, a quantidade de neurônios original vindas do modelo base foi dividida igualmente entre as camadas ocultas de cada modelo com duas ou mais camadas ocultas. Os modelos contendo três camadas ocultas, pela divisão dos neurônios não ser exata, seguiram o

padrão de distribuir os neurônios restantes nas primeiras camadas ocultas. A função de ativação $ReLU$ foi utilizada nessas camadas ocultas, seguindo o mesmo padrão da MLP base dos conjuntos de dados. A Tabela 4.5 mostra os modelos usados, cada um com uma quantidade de camadas ocultas diferentes.

Tabela 4.5: MLPs escolhidas para o experimento de variação do número de camadas.

Base de Dados	# Neurônios da(s) camada(s) oculta(s)
MNIST	800 400 - 400 267 - 267 - 266 200 - 200 - 200 - 200
Fashion-MNIST	100 50 - 50 34 - 33 - 33 25 - 25 - 25 - 25

Deste modo, o experimento vai treinar todos os modelos descrito na Tabela 4.5 seguindo as mesmas diretivas para o treinamento explicadas logo no início desta seção, obtendo a acurácia e o desvio padrão final para cada nível de ruído no intervalo. A respeito dos tipos de ruído usados, tanto o ruído uniforme e o *flip noise* serão aplicados, com o *flip noise* considerando os pares, em cada base de dados, conforme apresentados anteriormente.

4.4.4 Experimento 4: Efeito do Ruído de Classe variando a função de ativação

O objetivo deste experimento é treinar os modelos mudando a função de ativação usada nas RNAs sob o efeito do ruído nos dados e observar se a robustez do modelo pode ser afetado pela troca da função de ativação.

Tal como no experimento 1, foi aproveitado as arquiteturas clássicas vindas dos artigos Simard et al. (2003) e Xiao, Rasul e Vollgraf (2017) para o MNIST e o Fashion-MNIST. Para tentar isolar mais o efeito de robustez relacionado apenas a função de ativação, cada base de dados teve dois modelos treinados com 800 neurônios e 100 neurônios, as mesmas quantidade de neurônios das arquiteturas

clássicas consideradas. O *Softmax* foi usado como função de ativação na camada de saída.

Restringiu-se o experimento para considerar apenas três funções de ativação: a *ReLU*, a Tangente Hiperpólica (TanH) e a Sigmoide Logística, por serem bastante conhecidas na literatura, assim como já exposto na Seção 2.2. A Tabela 4.6 expõe a configuração dos modelos treinados neste experimento.

Tabela 4.6: MLPs escolhidas para o experimento de variação da função de ativação.

Base de Dados	# Neurônios (Função de Ativação)
MNIST	800 (ReLU) 800 (TanH) 800 (Sigmoide) 100 (ReLU) 100 (TanH) 100 (Sigmoide)
Fashion-MNIST	100 (ReLU) 100 (TanH) 100 (Sigmoide) 800 (ReLU) 800 (TanH) 800 (Sigmoide)

Deste modo, o experimento vai treinar todos os modelos descrito na Tabela 4.6 seguindo as mesmas diretivas para o treinamento explicadas logo no início desta seção, obtendo a acurácia e o desvio padrão final para cada nível de ruído no intervalo. A respeito dos tipos de ruído usados, tanto o ruído uniforme e o *flip noise* serão aplicados, com o *flip noise* considerando os pares, em cada base de dados, conforme apresentados anteriormente.

4.5 Análise dos Experimentos

Nesta seção irá se apresentar os resultados e conclusões dos experimentos enumerados na seção anterior, juntamente com as suas respectivas análises.

4.5.1 Experimento 1: Efeito do Ruído de Classe em RNAs pré-estabelecidas

O primeiro experimento teve como objetivo avaliar o efeito do ruído em arquiteturas clássicas. Nele, serão usados os conjunto de dados MNIST e Fashion-MNIST para a avaliação e o ruído uniforme e o *flip noise* como representantes do ruído aplicado.

A Figura 4.3 mostra o resultado do primeiro experimento nos conjuntos de dados escolhidos. No MNIST, pode-se observar comportamentos diferentes na degradação da acurácia do ruído uniforme e no *flip noise*. No ruído uniforme, o modelo manteve uma perda branda de acurácia até 60% de ruído até uma queda brusca de acurácia de 60% para 80% de ruído, enquanto que no *flip noise* o modelo manteve uma perda branda de acurácia até 40% de ruído até uma queda brusca de acurácia de 40% para 50% de ruído.

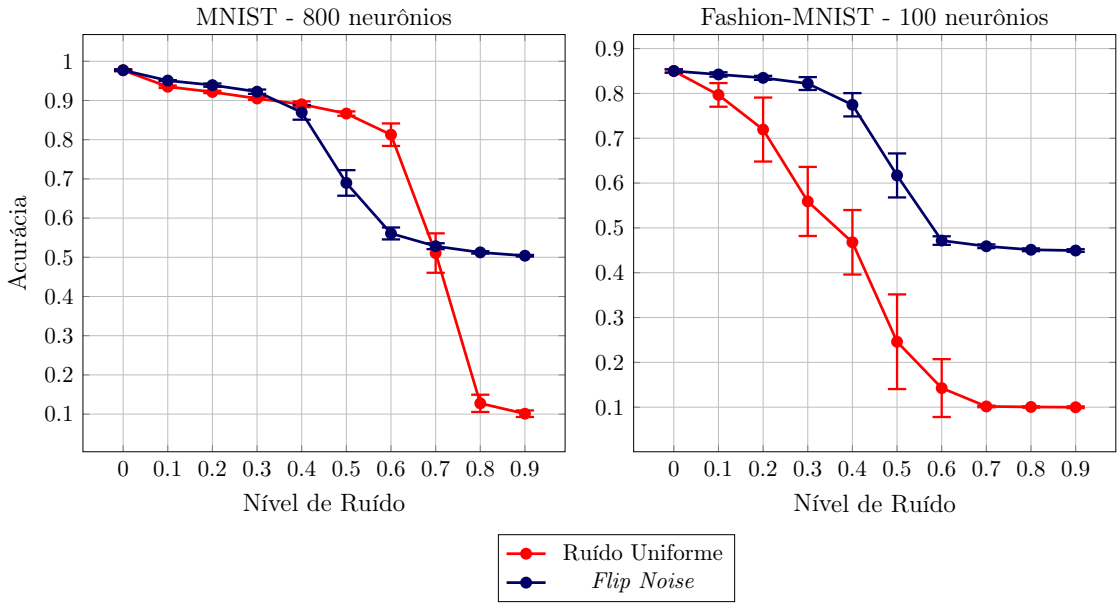


Figura 4.3: Resultado do primeiro experimento nas bases de dados MNIST e no Fashion-MNIST.

O interessante a se observar é a diferença no nível limite de ruído entre o ruído uniforme e o *flip noise* até se ter uma queda brusca de acurácia: no ruído uniforme, o modelo nesta base de dados manteve uma robustez até os 60% de ruído, enquanto que no *flip noise* a robustez se mostrou até os 40% de ruído.

No caso do *flip noise*, o ato de trocar a classe dos exemplos de uma classe a diretamente para uma única possibilidade de classe b ocasiona numa mudança mais

significativa na fronteira de decisão entre a classe a e b, pois cada vez mais exemplos que são a serão considerados como b devido ao *flip noise*, e o modelo vai aprender este padrão de forma mais evidente a medida que o nível de ruído aumenta. A Figura 4.4 ilustra um exemplo de como o *flip noise*, quando vai aumentando, pode afetar o aprendizado do modelo.

Para esta RNA no MNIST deste experimento, 50% de ruído foi o início de uma degradação mais acentuada, o que significa que a RNA começou a predizer numa quantidade maior os exemplos da classe a como sendo da classe b, até a estabilização desta degradação em 50% de acurácia com 90% de ruído. Estes 50% de acurácia fazem sentido porque o *flip noise* foi aplicado em 5 classes distintas, e estes 50% de acurácia correspondem aos acertos das classes não afetadas pelo *flip noise*.

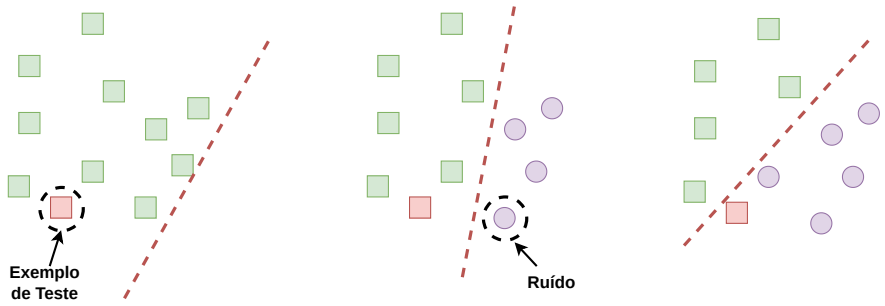


Figura 4.4: Exemplo de como o *flip noise* afeta os dados. Há 10 quadrados verdes como dados de treinamento. A medida que o nível de *flip noise* aplicado a este *cluster* aumenta, em um dado momento a fronteira de decisão irá se ajustar de forma que o quadrado vermelho seja classificado como sendo da classe ruidosa, causando uma perda de generalização do modelo na classe dos quadrados verdes.

No caso do ruído uniforme, há dois pontos que poderiam influenciar esta robustez do modelo até os 60%: a natureza distribuída do ruído uniforme e a boa separação das classes no MNIST.

Para ilustrar a boa separação das classes, a Figura 4.5 mostra uma visualização 2D do conjunto de dados do MNIST, usando uma abordagem chamada t-SNE (*t-Distributed Stochastic Neighbor Embedding*) (MAATEN; HINTON, 2008). O t-SNE é uma técnica voltada para redução de dimensionalidade bastante utilizada para visualização de dados dispostos em altas dimensões. A ideia principal é agrupar os exemplos de acordo com as similaridades locais com os seus vizinhos próximos, que

como consequência identificaria os *clusteres* de classes do conjunto. É um método estocástico e iterativo que precisa ser treinado. Vê-se na figura que os *clusteres* de classes estão bem formados e definidos, e que a sobreposição entre classes, dado o nível de similaridade, é bastante baixa, o que mostra que os padrões das classes no MNIST estão bem definidos.

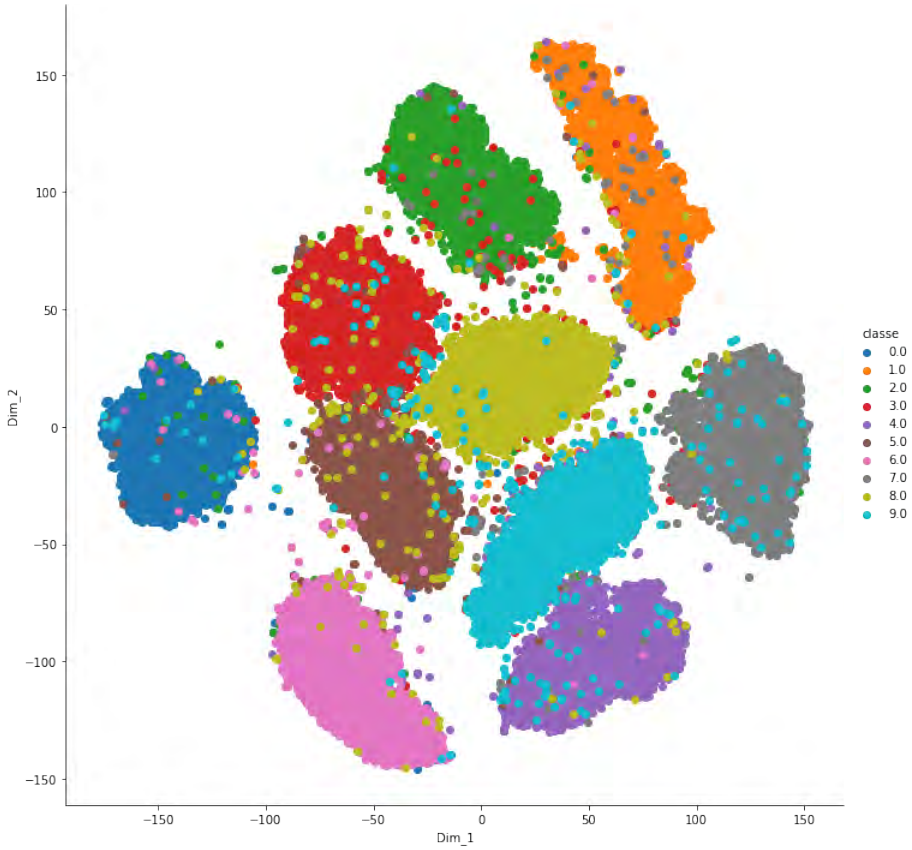


Figura 4.5: Representação do MNIST pelo t-SNE. A concentração de cada *cluster* colorido com pouquíssimos pontos sobrepostos em cada um dos *clusteres* mostra o quanto esta base está bem definida. Os pontos sobrepostos nos *clusteres* indicam que eles tem uma certa similaridade com os exemplos dos *clusteres* correspondentes, mas a quantidade é irrisória para mudar a fronteira de decisão que classifica corretamente cada um dos *clusteres*.

O ruído uniforme tem uma natureza distribuída por si só: quando falamos em aplicar, por exemplo, 60% de ruído nos dados que estão separadas em 10 classes diferentes, é quase o mesmo de se dizer que 60% dos dados de uma determinada classe estarão com ruído, e esses 60% serão divididos igualmente entre as 9 classes restantes que são diferentes da classe em questão. Neste exemplo, 6,67% dos exemplos ruidosos seriam separados para cada um das 9 classes. Diferentemente do *flip noise*

que tem um comportamento mais concentrado, o ruído uniforme tem esta natureza distribuída.

Assim, como a boa separação das classes deixa mais fácil o aprendizado das relações corretas dos atributos com as suas classes e o ruído uniforme apresenta um comportamento não concentrado nos exemplos ruidosos, estes fatores podem fazer com o que a quantidade de ruído presente nos *clusteres* do conjunto até aquele nível de 60% de ruído não seja o suficiente para mudar de forma drástica a fronteira de decisão que faz com que o modelo consiga aprender de forma expressiva o padrão daquele determinado *cluster*.

Quando se chega em 70% indo até 90%, a RNA já não consegue ter os exemplos suficientes para aprender os padrões das classes corretos devido ao ruído e começa a entender o padrão uniforme presente nos dados que foi incorporado pelo ruído uniforme.

No Fashion-MNIST, assim como no MNIST, pode-se observar comportamentos diferentes na degradação da acurácia do ruído uniforme e no *flip noise*. No ruído uniforme, o modelo teve quedas bem acentuadas desde os 10% de ruído até os 70% de ruído, enquanto que no *flip noise* o modelo manteve uma perda branda de acurácia até 40% de ruído até uma queda brusca de acurácia de 40% para 50% de ruído.

O interessante a se notar aqui é que no ruído uniforme o modelo no Fashion-MNIST apresentou uma sensibilidade considerável ao ruído uniforme, sempre tendo quedas relevantes entre um nível de ruído e outro, algo diferente com o que aconteceu com o MNIST.

Ao contrário do MNIST, o conjunto do Fashion-MNIST exibe uma separação das classes não tão bem definidas, tendo sobreposições muito grandes entre algumas classes. A Figura 4.6 mostra o resultado do t-SNE do Fashion-MNIST que ilustra essa questão. Com isto, o Fashion-MNIST demonstra um desafio maior, comparado ao MNIST, para aprender as suas classes, pois muitas das classes podem facilmente ser confundidas com outras classes que são bem similares. Como levantado na Subseção 2.3.4, alguns trabalhos da literatura mostram que uma das configurações

que deixam a classificação mais difícil é justamente a sobreposição entre as classes e que a maioria dos erros dos classificadores ocorre nos exemplos de fronteira. Ainda tendo a presença de ruído, o aprendizado poderia se tornar mais árduo que o normal.

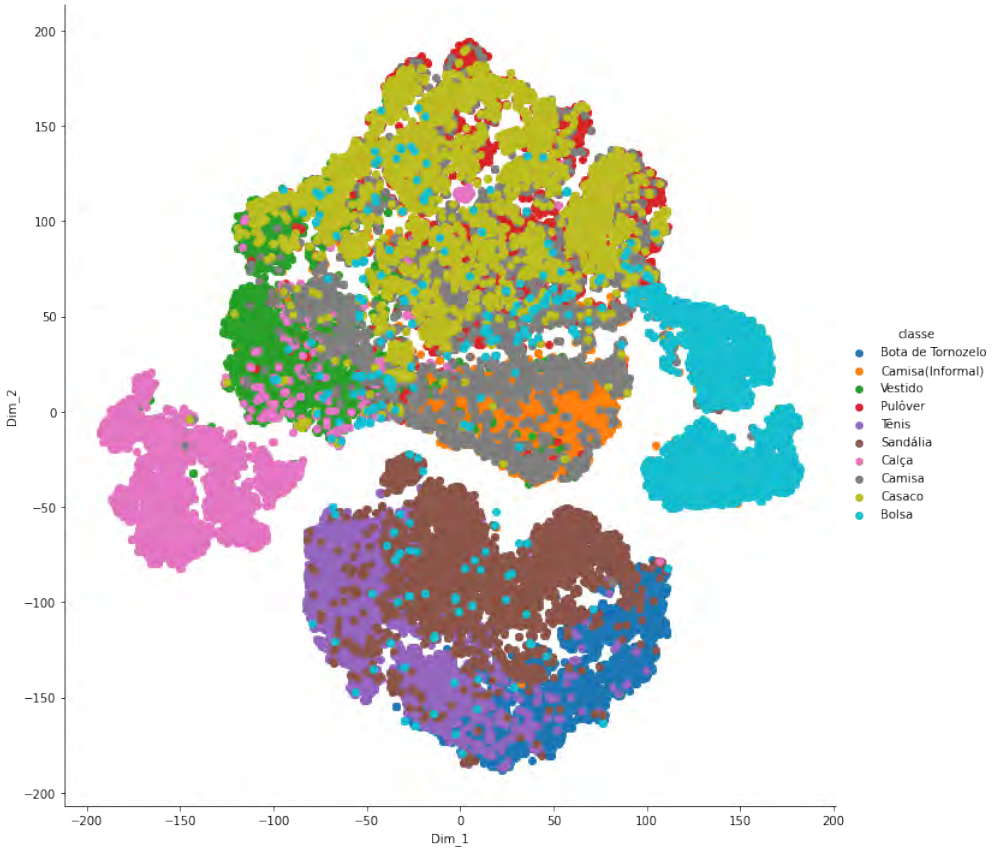


Figura 4.6: Representação do Fashion-MNIST pelo t-SNE. Pode-se observar que apenas as classes "Bolsa" e "Calça" estão bem isoladas e com praticamente nenhuma sobreposição de nenhuma outra classe, ao contrário das classes restantes que apresentam esta sobreposição muito fortemente.

Assim sendo, a dificuldade inerente ao domínio do Fashion-MNIST mais o obstáculo imposto pelo ruído uniforme aplicado nos dados podem ser um indício de um impacto mais forte na performance desta RNA no Fashion-MNIST, pois torna o aprendizado destes padrões ainda mais desafiador do que o normal.

Para reforçar esta ideia da dificuldade inerente ao domínio, a Figura 4.7 apresenta a execução de um pequeno experimento o qual foi treinada uma RNA com 800 neurônios sob a aplicação do ruído uniforme no Fashion-MNIST, para se ter uma comparação mais idônea entre os resultados da RNA entre as bases de dados no ruído uniforme.

Observamos que, como esperado, mesmo com a melhora na robustez, esta RNA de 800 neurônios no Fashion-MNIST é mais sensível ao ruído uniforme do que a do MNIST, pois o gráfico tem uma queda mais acentuada do que o gráfico do MNIST, o que reforça ainda mais a conclusão de que a dificuldade inerente ao domínio do conjunto de dados pode ser um indicador para ajudar a piorar ainda mais a performance da rede junto com certos tipos de ruído.

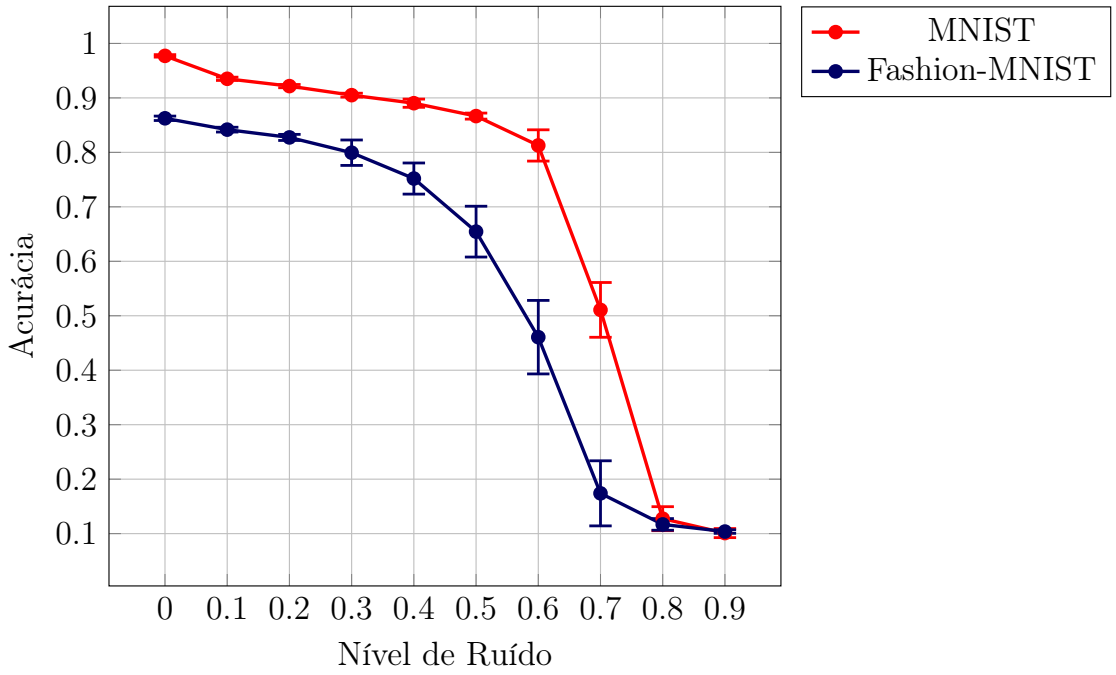


Figura 4.7: Comparação do MNIST com Fashion-MNIST no Ruído Uniforme com uma RNA de 800 neurônios.

O comportamento do *flip noise* deste modelo no Fashion-MNIST teve uma tendência de queda de performance parecida com a do MNIST, diferenciando em alguns valores de perda de acurácia no aumento dos níveis de ruído e pela estabilização em 45% de acurácia quando chegou a 90% de ruído. Como no MNIST, 5 classes do Fashion-MNIST sofreram com o *flip noise*, mas como este conjunto de dados tem esta questão da definição não tão clara dos *clusteres* das classes, estes 5% a menos de acerto pode ter esta referência dos erros inerentes ao domínio em si.

Em suma, pode-se tirar algumas conclusões a respeito deste experimento: a dificuldade inerente ao domínio do conjunto de dados pode ser um indicador que ajuda a intensificar o impacto negativo de certos tipos de ruído na performance e diferentes tipos de ruído podem promover efeitos distintos no aprendizado das RNAs.

4.5.2 Experimento 2: Efeito do Ruído de Classe variando o número de neurônios

O segundo experimento visa pegar as arquiteturas clássicas escolhidas e variar a quantidade de neurônios de cada uma. O MNIST e o Fashion-MNIST foram usados neste experimento. As figuras 4.8 e 4.9 mostram o resultado da variação dos neurônios nos modelos base escolhidos para o MNIST e Fashion-MNIST, respectivamente, a medida que o nível de ruído uniforme e o *flip noise* vão aumentando.

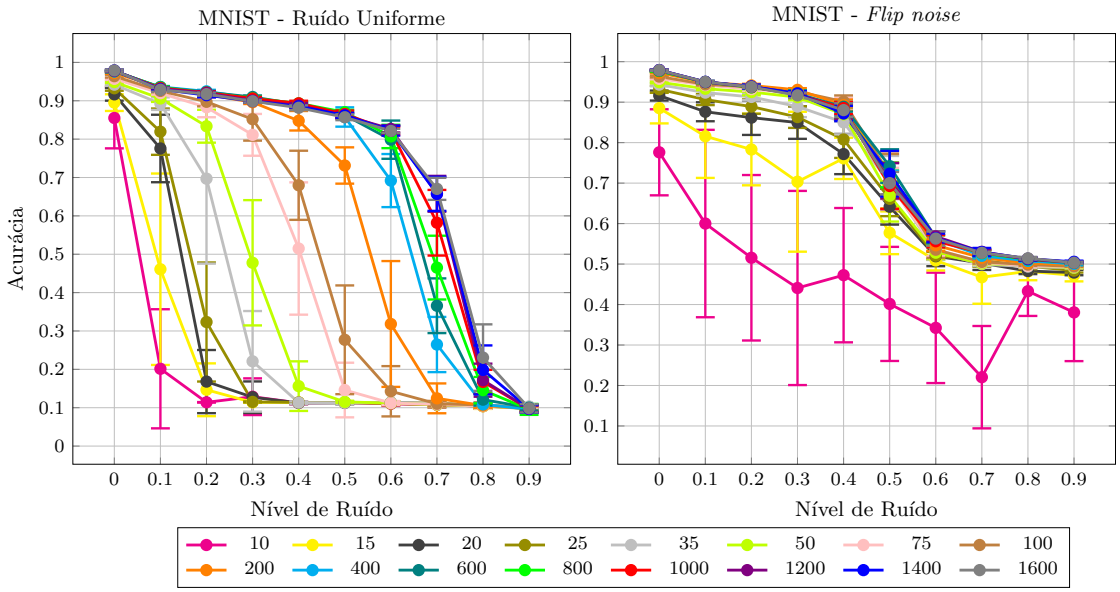


Figura 4.8: Resultado da variação dos neurônios a partir da RNA de 800 neurônios para o MNIST.

Pode-se observar um comportamento bem interessante nos gráficos que mostram o efeito sob o ruído uniforme: a medida que se aumenta o número de neurônios, percebe-se que os gráficos das performances dos modelos vão se tornando menos íngrimes com relação ao gráfico anterior a ele, dado a ordem de crescimento dos valores dos neurônios. Nesta ideia, se tornar menos íngrime traz o sentido de que a diferença do erro das acurácias entre os níveis de ruído adjacentes vai diminuindo. Isto significa o modelo vai ficando mais robusto a medida que o número de neurônios aumenta para as bases de dados consideradas no ruído uniforme.

A quantidade de neurônios para uma RNA define a sua complexidade, ou seja, quanto mais neurônios, maior será a sua capacidade de aprender funções mais

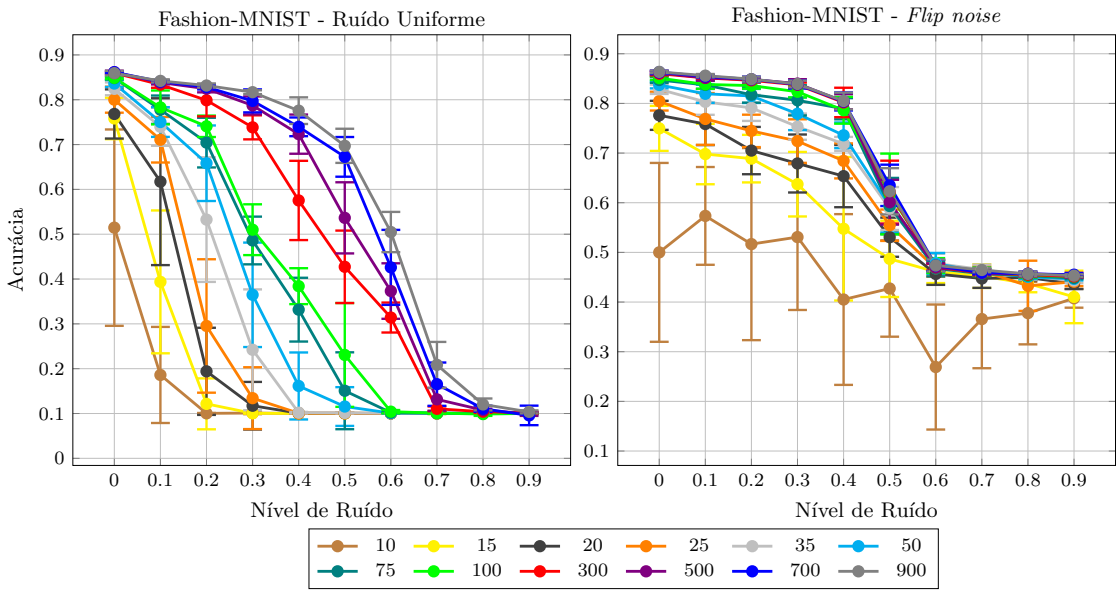


Figura 4.9: Resultado da variação dos neurônios a partir da RNA de 100 neurônios para o Fashion-MNIST.

complexas, pelo o aumento de parâmetros livres para serem treinados dado o aumento dos neurônios. Assim, para este escopo de experimento, pode-se dizer que uma RNA mais complexa tornou-a robusta, ou seja, mais resistente ao ruído uniforme.

Como já apresentado na Subseção 2.3.2, Frénay e Verleysen (2013) pontuaram como uma das consequências do Ruído de Classe sendo justamente o aumento da complexidade dos modelos preditivos quando há a presença do Ruído de Classe nos dados, com trabalhos da literatura mostrando resultados que ajudam a reforçar este fato e também o resultado obtido agora neste experimento.

Intuitivamente, pode-se explicar o comportamento deste aumento de complexidade da seguinte forma: pelo fato do modelo ser mais complexo, significa que as fronteiras de decisão que esse modelo poderia criar são mais maleáveis porque há mais parâmetros para serem treinados, o que dá ao modelo uma maior capacidade de separação das classes. Dada esta maior abrangência, é possível que o modelo mais complexo possa aprender um separador que mesmo com alguns dados ruidosos dentro do espaço de uma outra classe distinta ainda seja capaz de predizer corretamente os exemplos de teste. A Figura 4.10 ilustra com um exemplo esse comportamento do aumento da complexidade.

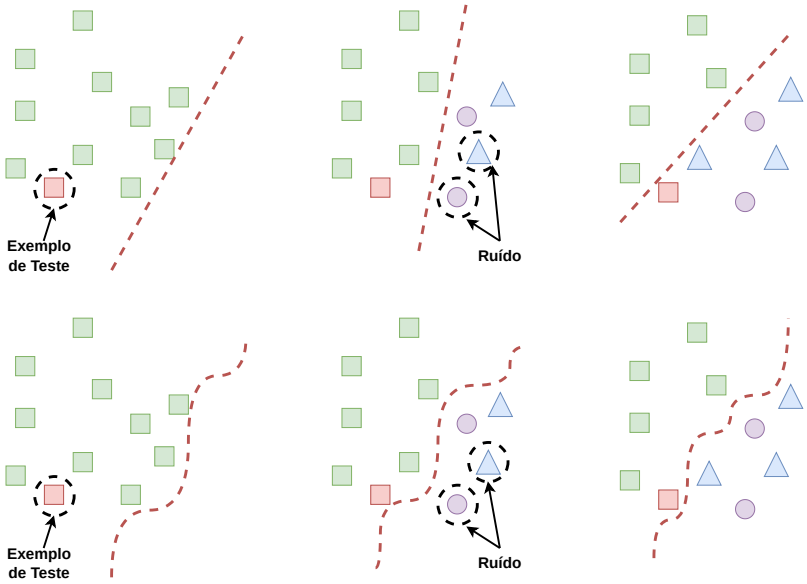


Figura 4.10: Exemplo de como o aumento da complexidade de um modelo pode torná-lo mais robusto. A primeira linha da imagem supõe que um modelo só consiga criar fronteiras de decisão lineares, sendo menos complexo, enquanto que a segunda linha supõe que um modelo já consiga criar fronteiras de decisão com curvas, sendo mais complexo. Quando o ruído chega em um determinado nível, o modelo menos complexo começa a errar os exemplos de teste ao mesmo tempo que o modelo mais complexo consegue definir uma fronteira de decisão que ainda possa generalizar.

Quando analisamos os gráficos do *flip noise*, assim como no ruído uniforme, pode-se ver um comportamento incremental de melhora nítida na robustez quando a complexidade do modelo neural vai aumentando desde a quantidade de 10 neurônios até 50 neurônios para o MNIST e 100 para o Fashion-MNIST, devido ao efeito do gráfico cada vez mais assumir um formato menos íngrime e constante quando vai aumentando a quantidade de neurônios.

É interessante notar os comportamentos bem peculiares do gráfico da RNA de 10 neurônios tanto no MNIST e no Fashion-MNIST: quedas e subidas irregulares a medida que o nível de ruído vai aumentando, além dos valores altos dos desvios padrões. Isso mostra o quanto a rede teve muita dificuldade em aprender os padrões nas bases de dados a medida que o nível de ruído aumentava, o que significa que o ruído presente realmente dificultou o aprendizado, aliado mais ao fato desta RNA não ser tão complexa comparada aos outras.

Depois dos 50 neurônios para o MNIST e dos 100 para o Fashion-MNIST,

os gráficos começam a se interpolar com maior intensidade, mostrando que suas acurácias são muito próximas e por causa disto não é possível averiguar uma certeza de melhora ou não sempre que a quantidade de neurônios aumenta. Existe a chance dessa melhorara ser bastante pequena ou então foi alcançado um limite máximo para que o alto número de neurônios possa impactar na robustez destas RNAs nestas bases de dados. A suposição do limite máximo faria sentido pois chegará um momento que a grande quantidade de dados ruidosos se torna tão expressiva no espaço que mesmo aumentando a capacidade de criar fronteiras de decisões mais complexas não iria adiantar, pois a RNA só busca aprender os padrões evidentes naquele momento, e para ele o certo é o padrão exibido pelos dados, independente se há ruído ou não nos dados.

Seria plausível como uma evolução futura a testagem destes experimentos envolvendo o *flip noise* considerando uma granularidade maior nos valores do intervalo do nível de ruído como também o aumento de vezes para treinar os modelos para cada nível de ruído, o que poderia amplificar mais as acurácias e assim ter um resultado mais assertivo.

Em suma, pode-se tirar a seguinte conclusão a respeito deste experimento: O aumento dos neurônios nestas bases de dados para o ruído uniforme ocasionou uma melhora na robustez dos modelos treinados nestas bases de dados, e no *flip noise* a melhora ocorreu em certos momentos. Isso dá um indício que quando a complexidade da RNA aumenta a robustez do mesmo pode melhorar, como exemplificado nos resultados deste experimento.

4.5.3 Experimento 3: Efeito do Ruído de Classe variando o número de camadas

O terceiro experimento envolve variar o número de camadas ocultas de cada uma das arquiteturas base escolhidas para o MNIST e para o Fashion-MNIST. As figuras 4.11 e 4.12 exibem os resultados dos modelos com a quantidade de camadas ocultas distintas para o MNIST e Fashion-MNIST sob o efeito do ruído uniforme e do *flip noise*.

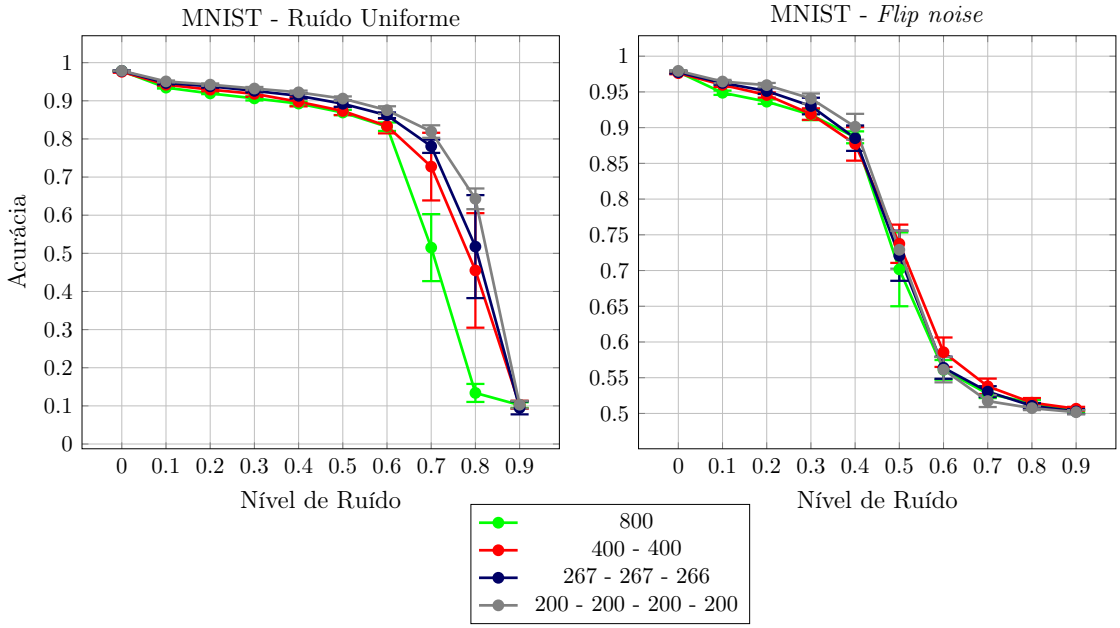


Figura 4.11: Resultado da variação das camadas a partir da RNA de 800 neurônios para o MNIST.

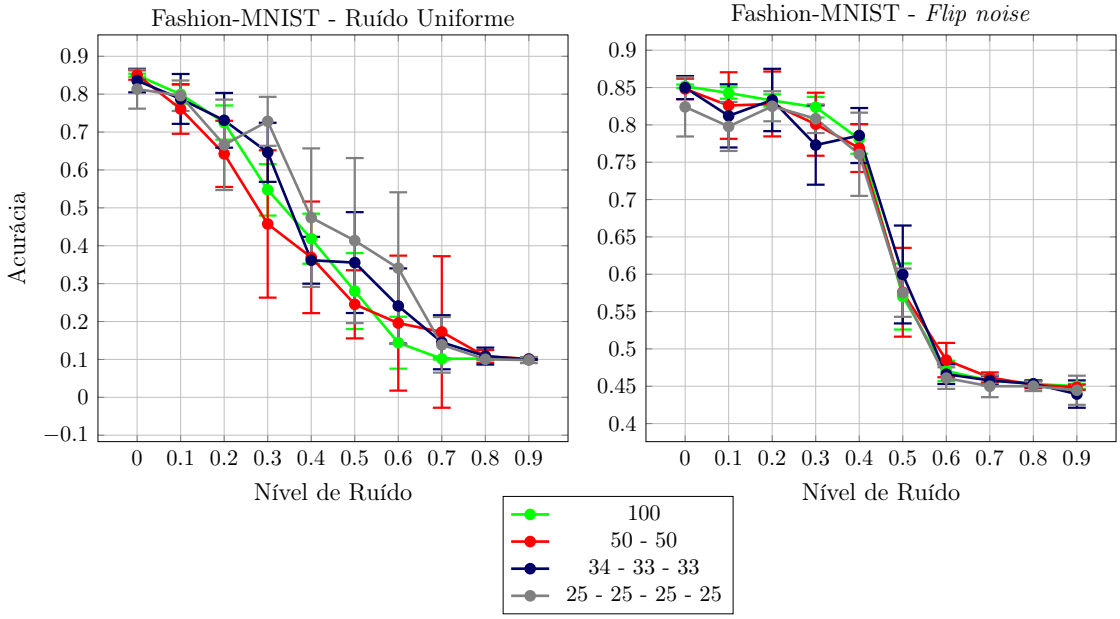


Figura 4.12: Resultado da variação das camadas a partir da RNA de 100 neurônios para o Fashion-MNIST.

No MNIST, quando colocamos mais camadas ocultas no modelo, houve uma certa melhora da robustez do modelo: no ruído uniforme, os níveis de ruído onde ocorre uma queda muito brusca da acurácia do modelo sobe quando o modelo é treinado com mais camadas. No de uma camada, esse ponto é de 60%; no de duas camadas

este ponto é de 70%; no de três camadas é de 70% e no de quatro é de 80%. No *flip noise*, observa-se que a inclinação do gráfico entre os pontos de 0% e 40% ficam menos íngrimes quando o modelo ficou com mais camadas. No Fashion-MNIST, os gráficos tiveram um comportamento instável: em ambos os tipos de ruído, não houve um padrão constante de melhora ou piora da robustez nos gráficos quando houve a adição de camadas.

Dado estes resultados, não se pode concluir algo genérico sobre a robustez desses modelos com relação aos ruídos usados neste experimento com as bases usadas, pois um fator pode ter contribuído para isto: a forma como foi definido os neurônios nas camadas para a construção dos modelos usados neste experimento. Como apresentado na Subseção 2.2.3, a adição de camadas na rede dá a RNA a capacidade de aplicar transformações sucessivas nos dados para criar outras representações internas da rede que facilitem a classificação do problema em si. A questão em si é que a flexibilidade destas transformações são controladas pela quantidade de neurônios presentes nas camadas, e a combinação de camadas com quantidade de neurônios diferentes podem ter graus de liberdade distintos para criar fronteiras de decisões entre as classes.

Neste experimento foi considerado que as camadas teriam uma quantidade de neurônios iguais para cada camada e que a quantidade total de neurônios não seria maior do que a arquitetura base para tentar permitir uma comparação mais idônea, mas pelos resultados bem diferentes entre as duas bases para os dois tipos de ruído, mostrou-se necessário a construção de um experimento o qual seria levado em conta a criação de modelos variando as camadas, mas que por vez seria variado a quantidade de neurônios naquela camada por vez para se ter um controle maior do aumento da capacidade de aprendizado da RNA. Por exemplo, quando for testar o caso com as RNAs com duas camadas, a primeira camada teria um número fixo de neurônios enquanto que outros modelos seriam criados variando o número de neurônios apenas da segunda camada, e assim seria possível ter um controle a mais na variação do grau de liberdade do aprendizado da RNA.

Em suma, pode-se tirar a seguinte conclusão a respeito deste experimento: Com o que foi apresentado, pode-se apenas ter um indício pequeno que aumentando o

número de camadas pode fazer alguma diferença no caso do MNIST, mas para se obter um indício mais assertivo e geral tanto no MNIST e no Fashion-MNIST é necessário a execução de um experimento mais focado na variação de neurônios por camada oculta para ter um controle maior no aumento do grau de liberdade da RNA e assim ter o resultado dessa evolução gradual da capacidade de aprendizado da RNA.

4.5.4 Experimento 4: Efeito do Ruído de Classe variando a função de ativação

O quarto experimento envolve mudar a função de ativação usada nas arquiteturas base escolhias para o MNIST e para o Fashion-MNIST. Dentre as funções de ativações, foram consideradas as funções ReLU, Sigmoide Logística e a Tangente Hiperbólica. As figuras 4.13 e a 4.14 exibem os resultados dos modelos com 800 e 100 neurônios com a mudança da função de ativação sob o efeito do ruído uniforme e *flip noise* para o MNIST e o Fashion-MNIST.

Vê-se nos gráficos de ambas as bases de dados no ruído uniforme uma tendência clara de uma queda mais branda da acurácia a medida que o nível de ruído aumenta nos modelos que usaram a função Sigmoide Logística e Tangente Hiperbólica com relação ao modelos que utilizaram a ReLU. A Sigmoide Logística se mostrou um pouco mais robusta que a Tangente Hiperbólica e a Tangente Hiperbólica mais robusta que a ReLU para as bases do MNIST e do Fashion-MNIST no ruído uniforme.

A Figura 4.15 apresenta com mais detalhes sobre a resistência ao ruído que cada função de ativação teve no ruído uniforme. Os gráficos metrificam o quanto de acurácia foi perdida até aquele nível de ruído correspondente, tendo como base a acurácia do modelo treinado sem a presença de ruído de acordo com a função de ativação analisada. Portanto, quando mais rápido a linha do gráfico crescer até os 90%, mais sensível ao ruído o modelo será. Vê-se que as linhas dos gráficos referentes a ReLU ficaram sempre acima das demais, e as linhas dos gráficos referentes a Tangente Hiperbólica quase sempre ficaram acima das linhas da Sigmoide Logística, o que significa dizer que a Tangente Hiperbólica se mostrou mais sensível ao ruído

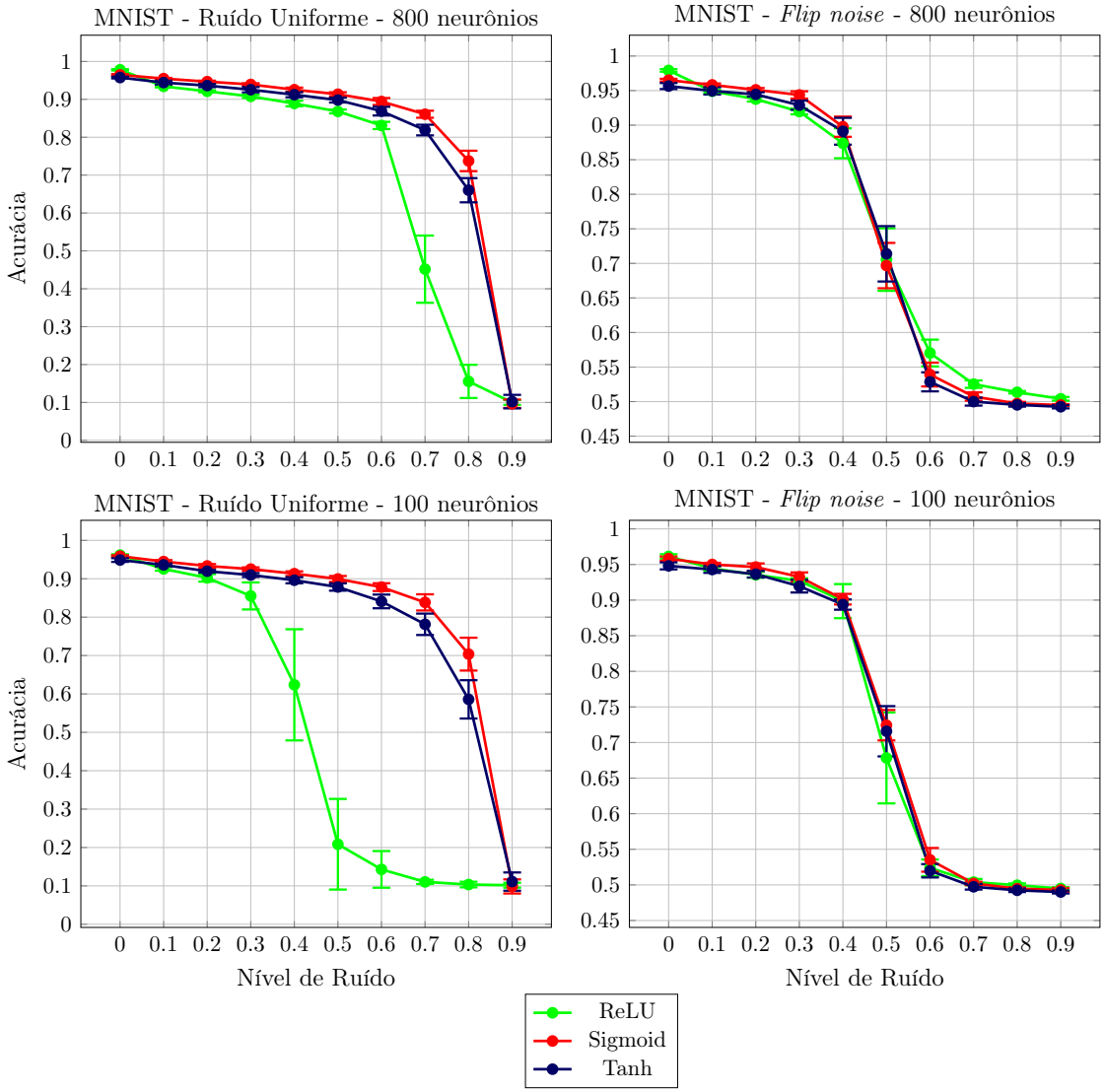


Figura 4.13: Resultado da mudança da função de ativação para o MNIST.

do que a Sigmoide Logística tanto para o MNIST e para o Fashion-MNIST no ruído uniforme.

Quando observamos a tendência dos gráficos do *flip noise*, a performance de robustez não se mantém a mesma como ocorreu no ruído uniforme. A Figura 4.16 mostra os gráficos de perda de acurácia até então dado o nível de ruído para o caso do *flip noise*. Quando se observa a tendência dos gráficos do *flip noise* entre os pontos de 0% e 40% de nível de ruído, para o MNIST a ReLU se mostrou menos robusta do que as outras funções de ativação e no Fashion-MNIST a ReLU se mostrou mais robusta do que as demais. No MNIST praticamente a Sigmoide Logística e a Tangente Hiperbólica tiveram a mesma robustez, sendo que no Fashion-MNIST em

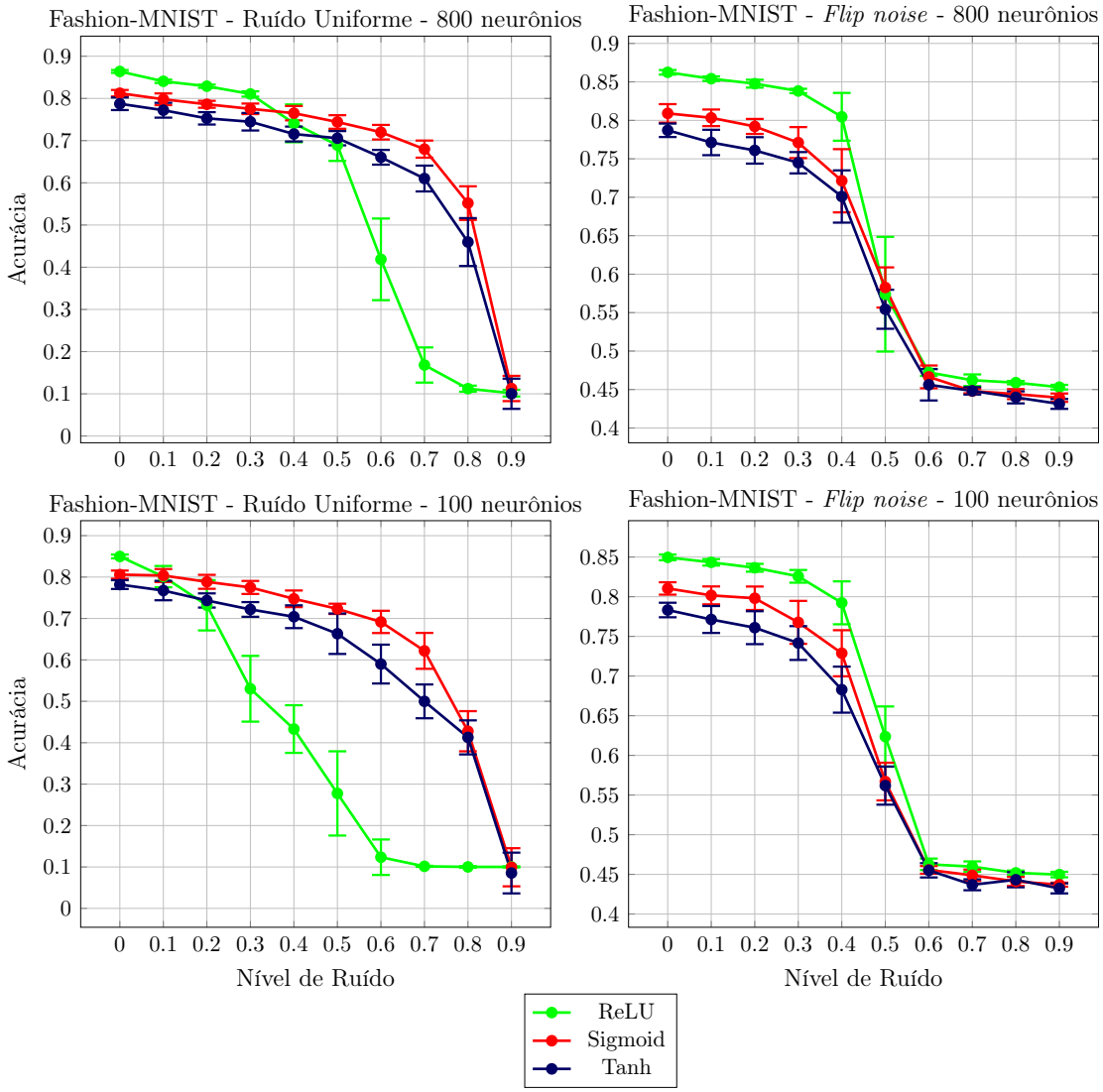


Figura 4.14: Resultado da mudança da função de ativação para o Fashion-MNIST.

alguns momentos a Sigmoide Logística se mostrou um pouco mais robusta que a Tangete Hiperbólica. Quando se analisa níveis de ruído mais altos, como no caso entre 60% e 90%, a ReLU revela ser menos robusta do que as demais, seguida pela Sigmoide Logística e a Tangente Hiperbólica.

O interessante a se observar é que para o *flip noise* a robustez dos modelos mudando a função de ativação foi diferente para o MNIST e para o Fashion-MNIST, ao contrário do caso do ruído uniforme que o mesmo padrão de robustez foi visto em ambas as bases de dados.

É importante fazer uma observação: a função ReLU originalmente foi adotada

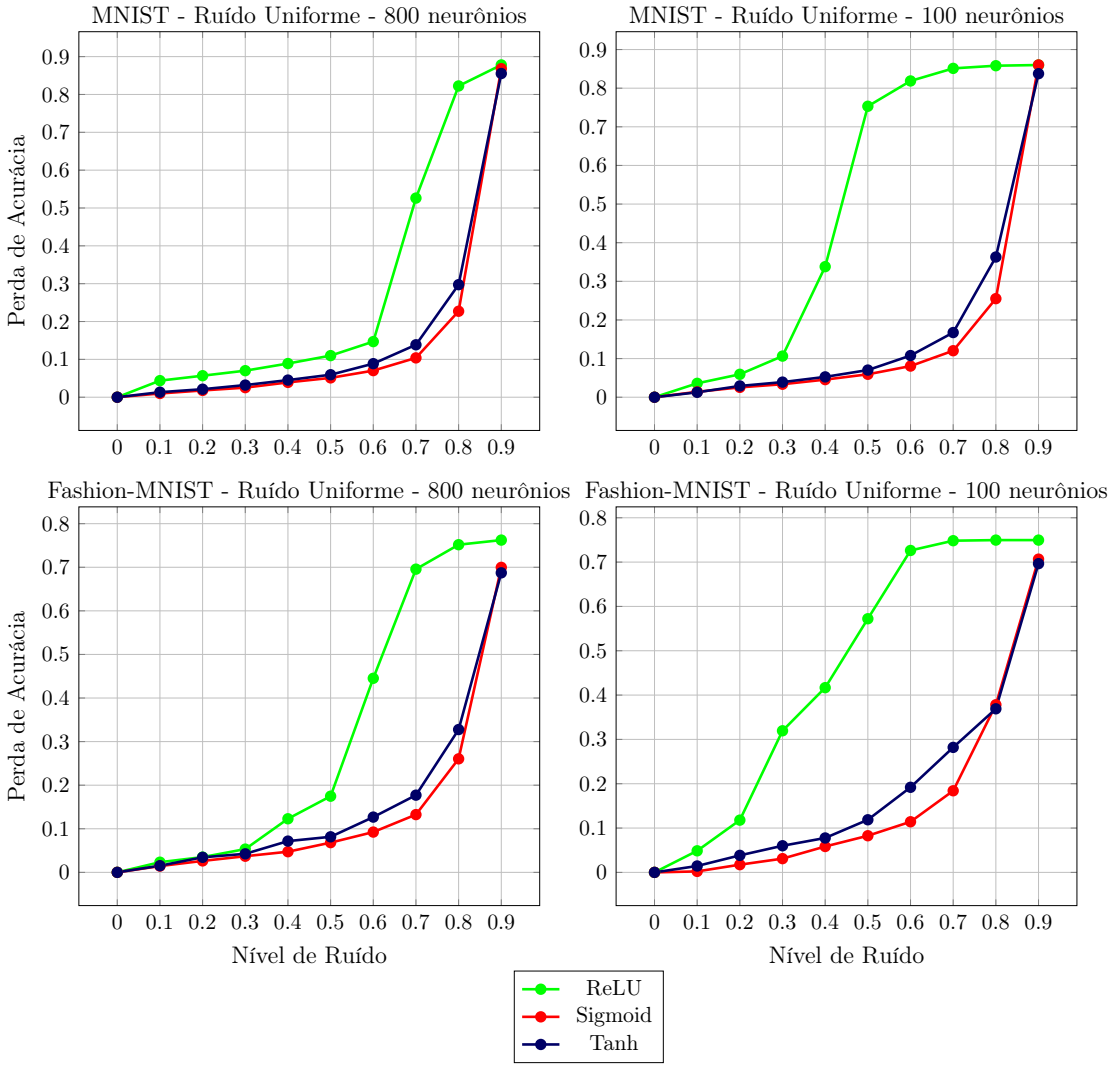


Figura 4.15: Diferença de acurácia do modelo base com relação ao modelo treinado no nível de ruído correspondente no ruído uniforme.

para evitar o problema de saturação da saída dos neurônios, ou seja, valores muito altos de entrada fazem a função de ativação chegar muito próximo do seu valor máximo e valores muitos baixos fazem a função de ativação chegar muito próximo do seu menor valor possível. Isso pode ocasionar em uma dificuldade maior no aprendizado da rede, principalmente quando a rede possui mais de uma camada, porque o valor final do ajuste dos pesos pode assumir um valor baixíssimo devido a intensa quantidade de multiplicações feitas mais o efeito do valor baixo da derivada da função de ativação para este ajuste no aprendizado usando a Retropropagação. A ReLU foi feita para evitar este problema quando se trabalha com redes de mais de uma camada, e o seu comportamento linear considerando o seu domínio de valores

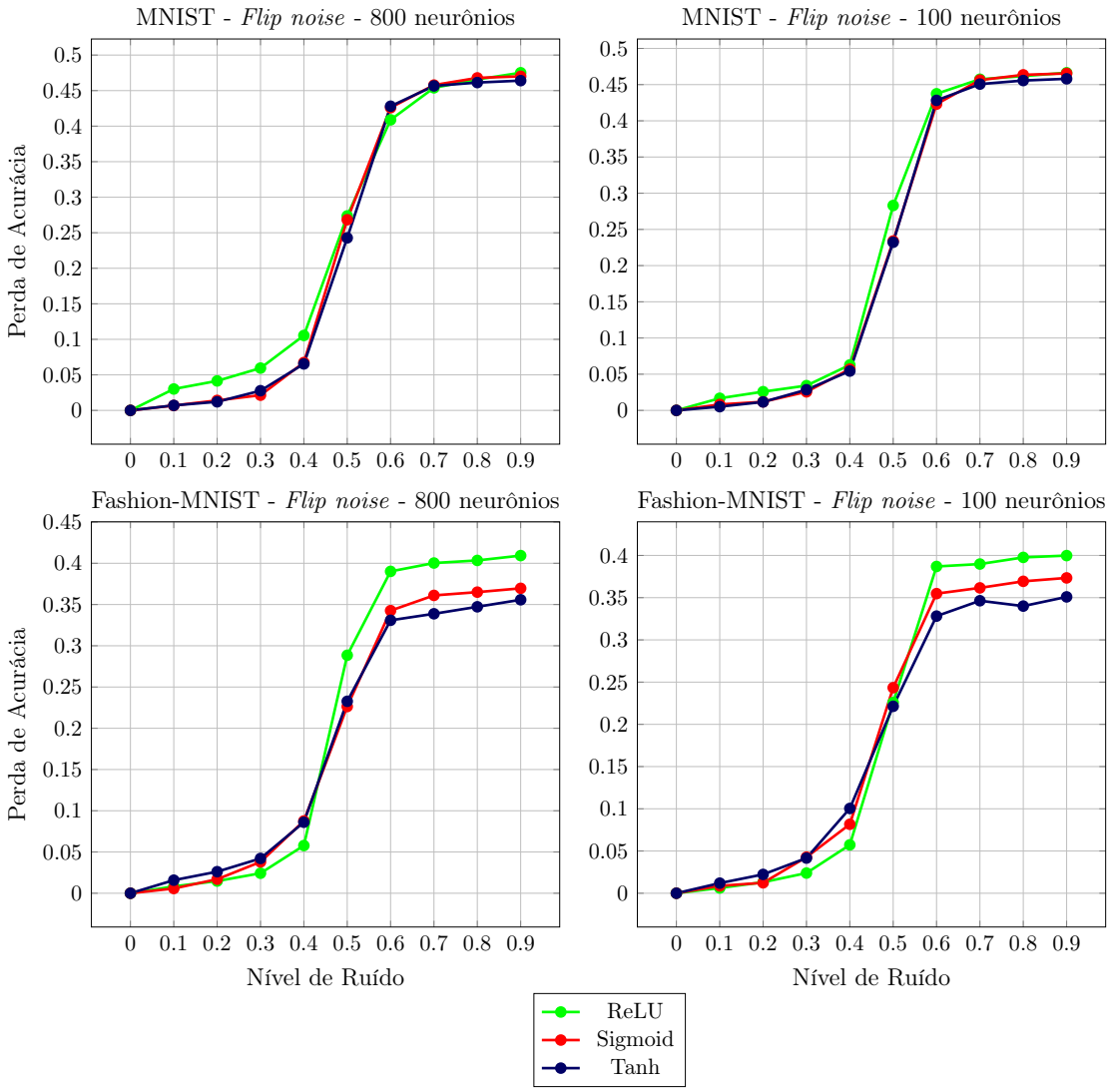


Figura 4.16: Diferença de acurácia do modelo base com relação ao modelo treinado no nível de ruído correspondente no *flip noise*.

positivos ataca este problema o qual a Sigmoide Logística e a Tangente Hiperbólica enfrentam. (GOODFELLOW; BENGIO; COURVILLE, 2016)

Por conseguinte, este experimento releva que dado uma RNA tendo apenas uma única camada oculta a ReLU mostrou uma desvantagem de robustez dado o contexto do experimento comparado as outras duas funções, e que seria relevante realizar outros experimentos mudando a função de ativação levando em conta mais camadas ocultas na RNA.

Em suma, pode-se tirar a seguinte conclusão a respeito deste experimento: a escolha da função de ativação a ser usada nas camadas ocultas ocasionou em um

impacto na robustez dos modelos treinados no MNIST e no Fashion-MNIST, o que traz um indício de que esta escolha poderia fazer a diferença em determinar a robustez do modelo neural, exemplificado com os resultados deste experimento.

Capítulo 5

Conclusão

Neste capítulo será apresentado as conclusões finais a respeito do que foi feito no trabalho e os resultados obtidos nos experimentos, junto com as limitações encontradas durante o processo e possíveis extensões que poderiam ser trabalhos futuros para o referente trabalho.

5.1 Considerações finais

Neste trabalho diversos experimentos foram feitos em algumas bases de dados bem conhecidas na literatura com o fim de analisar o efeito do Ruído de Classe em modelos baseados em RNAs, os quais hiperparâmetros referentes as RNAs foram variados e testados para criar estes modelos e assim observou-se como a robustez desses modelos foram alteradas.

Uma importante lição a ser tirada é o quão relevante é levar em consideração a presença do Ruído de Classe no conjunto de dados, que se não tratado pode causar consequências negativas no aprendizado do modelo neural que se refletem na sua performance final, como bem exemplificado nos experimentos feitos anteriormente.

Ademais, os resultados dos experimentos trouxeram uma ideia de que algumas alterações nos hiperparâmetros específicos da RNA como a quantidade de neurônios, o número de camadas e a função de ativação usada nos neurônios podem alterar a

robustez do modelo da RNA, o que ajuda a inferir um indício de que na busca de uma robustez natural do modelo sem a intervenção de nenhum tipo de abordagem externa para lidar com o Ruído de Classe a escolha desses hiperparâmetros tem a sua dada importância.

5.2 Limitações e trabalhos futuros

Como já levantado em alguns momentos deste trabalho, existem limitações associados ao escopo dos experimentos que impediram de se tirar conclusões mais assertivas sobre alguns fatores, como já esperado deste tipos de trabalhos experimentais como este a também vários da literatura.

Uma limitação é a restrição da quantidade de bases de dados usadas. Neste trabalho foram usadas apenas duas bases: o MNIST e o Fashion-MNIST. Realizar a testagem com várias bases diferentes aumenta o escopo de análise e ajuda a deixar as conclusões mais assertivas. Um trabalho futuro seria a execução dos experimentos deste trabalho em outras bases de dados distintas para incrementar os resultados aqui obtidos. Uma outra adição interessante seria considerar bases com contextos distintos aos que foram usadas no trabalho que no caso se restringiu ao problema de classificação de imagens. A mudança de contexto ajudaria a eliminar mais o viés da análise focada em um único tipo de problema de classificação.

Outra limitação é a falta de um tipo de ruído que representasse o modelo NNAR. Neste trabalho apenas os modelos NCAR e o NAR foram considerados, retratados pelo ruído uniforme e o *flip noise*, respectivamente. O modelo NNAR é o caso mais geral da taxonomia do Ruído de Classe a abre margem para modelar diversos problemas reais que podem ser de grande interesse, pois ele inclui a influência dos atributos para que o Ruído de Classe ocorra. Um trabalho futuro seria a execução dos experimentos deste trabalho nas bases de dados com um tipo de ruído NNAR aplicado e ver como a robustez dos modelos neurais iriam se comportar. O trabalho de Algan e Ulusoy (2020) propôs uma forma de gerar um tipo de ruído NNAR, podendo ele ser usado nesta proposta de trabalho futuro.

Outra limitação é o intervalo de ruído considerado e a quantidade de vezes para treinar um mesmo modelo. Esta limitação já foi pontuada no experimento de variação de neurônios durante a análise dos resultados em cima do *flip noise*, que com um detalhamento maior os resultados para este caso seriam mais assertivos. Não só para este caso mas de forma geral, a determinação de uma granularidade maior do intervalo de ruído e o aumento de vezes para treinar um modelo trariam certamente resultados mais detalhados para qualquer experimento que fosse feito, e isso sempre é bem-vindo. Um trabalho futuro seria a escolha de intervalos menores de variação de ruído como 1% ou 5% ao invés de 10% usado neste trabalho e um aumento de 10 para 20 vezes o treinamento de cada modelo neural.

Outra limitação é a abordagem de como foi feita a variação da quantidade de camadas em um dos experimentos do trabalho. Como detalhado na Subseção 4.5.3, houve uma limitação de análise pela forma de como foi criado os modelos com quantidade de camadas ocultas distintas. Um trabalho futuro seria atacar este experimento para atender a demanda levantada, que no caso seria variar a quantidade de camadas mantendo fixo os neurônios das camadas anteriores, apenas variando o número de neurônios da camada oculta da vez, fazendo o mesmo processo para todas as camadas ocultas que estiverem sendo consideradas.

Outra limitação é a quantidade de funções de ativações escolhidas. Para definir um escopo e um início de trabalho para analisar essa parte da função de ativação, foram selecionadas as funções de ativação mais conhecidas na literatura, mas existem outras que poderiam ser exploradas, como a *leaky ReLU* e a *Swish*, por exemplo (SHARMA; SHARMA; ATHAIYA, 2017). Um trabalho futuro seria considerar estas outras funções de ativações no experimento e verificar a robustez da RNA na presença destas outras funções de ativação, podendo comparar com os resultados já obtidos com a ReLU, Sigmoide Logística e a Tangente Hiperbólica atingidos neste trabalho.

Referências

AGGARWAL, C. C. *Neural Networks and Deep Learning*. [S.l.]: Springer, 2018. v. 10. 978–3 p.

ALGAN, G.; ULUSOY, I. Label noise types and their effects on deep learning. *arXiv preprint arXiv:2003.10471*, 2020.

ALPAYDIN, E. *Introduction to Machine Learning*. [S.l.: s.n.], 2014.

ANGLUIN, D.; LAIRD, P. Learning from noisy examples. *Machine Learning*, Springer, v. 2, n. 4, p. 343–370, 1988.

BRAGA, A. de P.; FERREIRA, A. C. P. de L.; LUDERMIR, T. B. *Redes neurais artificiais: teoria e aplicações*. [S.l.]: LTC Editora Rio de Janeiro, Brazil:, 2007.

BRODLEY, C. E.; FRIEDL, M. A. Identifying mislabeled training data. *Journal of artificial intelligence research*, v. 11, p. 131–167, 1999.

CORDEIRO, F. R.; CARNEIRO, G. A survey on deep learning with noisy labels: How to train your model when you cannot trust on the annotations? In: IEEE. *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. [S.l.], 2020. p. 9–16.

CYBENKO, G. Continuous valued neural networks with two hidden layers are sufficient, department of computer science. *Tufts University*, 1988.

CYBENKO, G. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, Springer, v. 2, n. 4, p. 303–314, 1989.

DENG, J. et al. Imagenet: A large-scale hierarchical image database. In: IEEE. *2009 IEEE conference on computer vision and pattern recognition*. [S.l.], 2009. p. 248–255.

FATIMA, N. et al. Enhancing performance of a deep neural network: A comparative analysis of optimization algorithms. Ediciones Universidad de Salamanca (España), 2020.

FRÉNAY, B.; VERLEYSEN, M. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, IEEE, v. 25, n. 5, p. 845–869, 2013.

GARCÍA, S.; LUENGO, J.; HERRERA, F. *Data Preprocessing in Data Mining*. [S.l.]: Springer, 2015. v. 72.

GARCÍA, V. et al. Combined effects of class imbalance and class overlap on instance-based classification. In: SPRINGER. *International Conference on Intelligent Data Engineering and Automated Learning*. [S.l.], 2006. p. 371–378.

GHAHRAMANI, Z. Unsupervised learning. In: SPRINGER. *Summer School on Machine Learning*. [S.l.], 2003. p. 72–112.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.

HAN, B. et al. A survey of label-noise representation learning: Past, present and future. *arXiv preprint arXiv:2011.04406*, 2020.

HAYKIN, S. *Redes neurais: princípios e prática*. [S.l.]: Bookman Editora, 2007.

HICKEY, R. J. Noise modelling and evaluating learning from examples. *Artificial Intelligence*, Elsevier, v. 82, n. 1-2, p. 157–179, 1996.

HINTON, G.; VINYALS, O.; DEAN, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

HINTON, G. E. et al. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. *Neural networks*, Elsevier, v. 2, n. 5, p. 359–366, 1989.

HUBER, P. J. *Robust statistics*. [S.l.]: John Wiley & Sons, 2004. v. 523.

HUGHES, N. P.; ROBERTS, S. J.; TARASSENKO, L. Semi-supervised learning of probabilistic models for ecg segmentation. In: IEEE. *The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. [S.l.], 2004. v. 1, p. 434–437.

INOUYE, D. I. et al. Hyperparameter selection under localized label noise via corrupt validation. In: *NIPS Workshop*. [S.l.: s.n.], 2017.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

KRIZHEVSKY, A.; HINTON, G. et al. Learning multiple layers of features from tiny images. Citeseer, 2009.

LECUN, Y. The mnist database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*, 1998.

LIBRALON, G. L. et al. Pre-processing for noise detection in gene expression classification data. *Journal of the Brazilian Computer Society*, SpringerOpen, v. 15, n. 1, p. 3–11, 2009.

LORENA, A. C.; CARVALHO, A. C. de. Evaluation of noise reduction techniques in the splice junction recognition problem. *Genetics and Molecular Biology*, SciELO Brasil, v. 27, p. 665–672, 2004.

MAATEN, L. Van der; HINTON, G. Visualizing data using t-sne. *Journal of machine learning research*, v. 9, n. 11, 2008.

MALETIC, J. I.; MARCUS, A. Data cleansing: Beyond integrity analysis. In: CITESEER. *Iq*. [S.l.], 2000. p. 200–209.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943.

MINSKY, M.; PAPERT, S. Perceptrons: An introduction to computational geometry. *MIT Press, Massachusetts*, 1969.

MITCHELL, T. M. *Machine Learning*. New York: McGraw-Hill, 1997. ISBN 978-0-07-042807-2.

MITCHELL, T. M. *Machine Learning*. New York: McGraw-Hill, 1997, p.2, tradução própria. ISBN 978-0-07-042807-2.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. *Sistemas inteligentes-Fundamentos e aplicações*, Manole Ltda, v. 1, n. 1, p. 32, 2003.

NAPIERAŁA, K.; STEFANOWSKI, J.; WILK, S. Learning from imbalanced data in presence of noisy and borderline examples. In: SPRINGER. *International conference on rough sets and current trends in computing*. [S.l.], 2010. p. 158–167.

NETTLETON, D. F.; ORRIOLS-PUIG, A.; FORNELLS, A. A study of the effect of different types of noise on the precision of supervised learning techniques. *Artificial intelligence review*, Springer, v. 33, n. 4, p. 275–306, 2010.

NIGAM, N.; DUTTA, T.; GUPTA, H. P. Impact of noisy labels in learning techniques: a survey. In: *Advances in data and information sciences*. [S.l.]: Springer, 2020. p. 403–411.

ORR, K. Data quality and systems theory. *Communications of the ACM*, ACM New York, NY, USA, v. 41, n. 2, p. 66–71, 1998.

PATRINI, G. et al. Making deep neural networks robust to label noise: A loss correction approach. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2017. p. 1944–1952.

PECHENIZKIY, M. et al. Class noise and supervised learning in medical domains: The effect of feature extraction. In: IEEE. *19th IEEE symposium on computer-based medical systems (CBMS'06)*. [S.l.], 2006. p. 708–713.

PRATI, R. C.; LUENGO, J.; HERRERA, F. Emerging topics and challenges of learning from noisy data in nonstandard classification: a survey beyond binary class noise. *Knowledge and Information Systems*, Springer, v. 60, n. 1, p. 63–97, 2019.

PRECHELT, L. Early stopping-but when? In: *Neural Networks: Tricks of the trade*. [S.l.]: Springer, 1998. p. 55–69.

ROLNICK, D. et al. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*, 2017.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, American Psychological Association, v. 65, n. 6, p. 386, 1958.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *nature*, Nature Publishing Group, v. 323, n. 6088, p. 533–536, 1986.

RUSIECKI, A. Standard dropout as remedy for training deep neural networks with label noise. In: SPRINGER. *International Conference on Dependability and Complex Systems*. [S.l.], 2020. p. 534–542.

RUSSELL, S. J.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. Global edition. [S.l.]: Pearson, 2021.

SÁEZ, J. A. et al. Analyzing the presence of noise in multi-class problems: alleviating its influence with the one-vs-one decomposition. *Knowledge and information systems*, Springer, v. 38, n. 1, p. 179–206, 2014.

SCULLEY, D.; CORMACK, G. V. Filtering email spam in the presence of noisy user feedback. In: CITESEER. *CEAS*. [S.l.], 2008.

SHALEV-SHWARTZ, S.; BEN-DAVID, S. *Understanding machine learning: From theory to algorithms*. [S.l.]: Cambridge university press, 2014.

SHARMA, S.; SHARMA, S.; ATHAIYA, A. Activation functions in neural networks. *towards data science*, v. 6, n. 12, p. 310–316, 2017.

SIMARD, P. Y. et al. Best practices for convolutional neural networks applied to visual document analysis. In: *Icdar*. [S.l.: s.n.], 2003. v. 3, n. 2003.

SNOW, R. et al. Cheap and fast–but is it good? evaluating non-expert annotations for natural language tasks. In: *Proceedings of the 2008 conference on empirical methods in natural language processing*. [S.l.: s.n.], 2008. p. 254–263.

SONG, H. et al. Learning from noisy labels with deep neural networks: A survey. *arXiv preprint arXiv:2007.08199*, 2020.

STRONG, D. M.; LEE, Y. W.; WANG, R. Y. Data quality in context. *Communications of the ACM*, ACM New York, NY, USA, v. 40, n. 5, p. 103–110, 1997.

VALIANT, L. G. A theory of the learnable. *Communications of the ACM*, ACM New York, NY, USA, v. 27, n. 11, p. 1134–1142, 1984.

WU, X. *Knowledge acquisition from databases*. [S.l.]: Intellect books, 1995.

XIAO, H.; RASUL, K.; VOLLGRAF, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

ZHANG, W.; REKAYA, R.; BERTRAND, K. A method for predicting disease subtypes in presence of misclassification among training samples using gene expression: application to human breast cancer. *Bioinformatics*, Oxford University Press, v. 22, n. 3, p. 317–325, 2006.

ZHANG, Z.; SABUNCU, M. Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems*, v. 31, 2018.

ZHU, X.; WU, X. Class noise vs. attribute noise: A quantitative study. *Artificial intelligence review*, Springer, v. 22, n. 3, p. 177–210, 2004.