UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENG. DE AUTOMAÇÃO E SISTEMAS
DAS410058 – APRENDIZADO DE MÁQUINA

THAYS DA CRUZ FRANCO

WILLIAN DO NASCIMENTO FINATO BENOSKI

RELATÓRIO: TREINAMENTO DE REDES NEURAIS MULTI-CAMADAS

APLICAÇÃO DO MÉTODO DE RETROPROPAGAÇÃO

FLORIANÓPOLIS – SC

2022/2

UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENG. DE AUTOMAÇÃO E SISTEMAS
DAS410058 – APRENDIZADO DE MÁQUINA

THAYS DA CRUZ FRANCO

WILLIAN DO NASCIMENTO FINATO BENOSKI

RELATÓRIO: TREINAMENTO DE REDES NEURAIS MULTI-CAMADAS

APLICAÇÃO DO MÉTODO DE RETROPROPAGAÇÃO

Relatório desenvolvido para obtenção de nota parcial na disciplina de aprendizado de máquina do programa de pós-graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina

DOCENTE: PROF. DR. ERIC AISLAN ANTONELO

FLORIANÓPOLIS – SC

2022/2

SUMÁRIO

INTRODUÇÃO

Após a segunda guerra mundial houve uma crescente busca para fundamentar os conhecimentos em Inteligência Artificial, para diferentes linhas de pesquisas as aplicações variam desde funções com padrão de repetição à busca por soluções inovadoras. Neste contexto, verificando os dados de entrada e o resultado desejado, é necessária a aplicação de algoritmos que realizem o tratamento dos dados e definam os valores de hiperparâmetros que minimizem os custos e torne o modelo mais preparado para o treinamento. (McCARTHY, 2007)

A área de redes neurais busca representar as atividades cognitivas realizadas pelo cérebro humano. É um conjunto de neurônios, dispostos em camadas e interconectados. Para isto há um processo básico de definição da arquitetura da rede, o número de neurônios e como estão conectados, há a definição da função de erro para quantificar quão bem os objetivos estão sendo alcançados e por fim um algoritmo que minimize o erro. (LILICRAP et al. 2020)

Neste relatório propõe-se analisar o desempenho do algoritmo de retropropagação de erros, usualmente aplicado para reconhecimento de imagens, aprendizagem não supervisionada, predição e modelagem de linguagem, e quando combinado com aprendizado por reforço, gera também bons resultados em controle. Desta forma, inicialmente foi gerada uma estrutura de dados para armazenar os pesos definindo assim a arquitetura da rede, implementado o algoritmo de retropropagação, a aproximação numérica do gradiente através da validação, e por ultimo a parte de descenso do gradiente para minimização do erro, seguindo o fluxo conforme supracitado.

O objetivo é analisar o desempenho do programa, verificando a alteração de parâmetros, implementação do código, análise de erro ao longo das épocas e o desempenho para classificação dos dados através das curvas de nível.

2. METODOLOGIA

O desenvolvimento do código foi realizado seguindo as instruções dadas em sala de aula e utilizando a linguagem Python.

2.1. Etapa 1: Inicializar a rede

Para inicializar a rede, são definidos a quantidade de neurônios, a ordem da camada, o tipo de função (tangente hiperbólica, sigmoide ou relu) e o tipo da camada (oculta, entrada ou saída). Em seguida, é realizada a inicialização do Xavier, que é uma forma de ponderar as entradas da rede neural, igualando a variância da saída à variância da entrada, estabilizando o processo. São determinados os valores de pesos e vieses (bias) e estes são armazenados em lista de matrizes.

2.2. Etapa 2: Propagação direta

Nesta etapa é realizado o processamento em cada neurônio através da função de ativação. Aqui foram especificadas três funções: sigmoide, utilizada quando necessário saída com apenas números positivos; tangente hiperbólica, para saídas entre -1 e 1; e ReLU, que retorna 0 para valores negativos, e a si próprio para valores positivos.

Desta forma, foi definida uma função que tomando como parâmetros a rede, os pesos e as entradas, aplica a função escolhida para os neurônios de cada camada.

2.3. Etapa 3: Retropropagação de erros

Para esta etapa são determinadas as derivadas das funções. É então calculado o erro de cada neurônio j em cada camada l, em seguida calculado o delta que posteriormente é dividido pelo grupo todo, retornando assim um valor médio.

Assim, foi definida uma função que tendo como parâmetros a rede, a saída, os pesos e dados de entrada, retorna os valores para gradiente e os deltas calculados.

2.4. Etapa 4: Gradiente

Nesta etapa inicialmente foi determinado o custo utilizando entropia cruzada. Com a função determinada foi iniciada a função para a aproximação do gradiente, que computa numericamente a aproximação entre os valores de gradiente gerados pelo algoritmo de retropropragação com uma estimação do gradiente. Foi feita uma função que recebe como parâmetros os pesos, um hiperparâmetro épsilon, a rede, entradas e saída, gerando como resultado um valor de gradiente aproximado, que após normalizado, pode ser comparado com os valores de gradiente da retropropagação através de uma razão.

Já o gradiente descendente foi utilizado para minimizar o custo em função da alteração dos pesos. Portanto, dentro do looping para as épocas no treinamento, há a atualização dos pesos de toda a rede no final de cada época, gerando assim novas respostas do desempenho da rede.

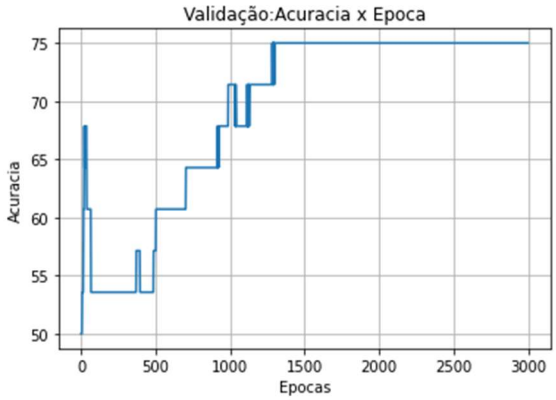
3. RESULTADOS E DISCUSSÕES

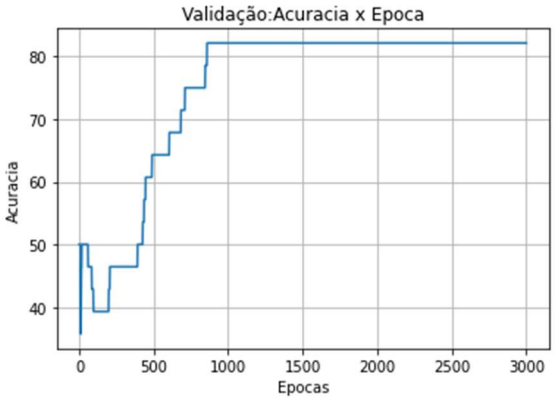
Inicialmente foi realizado um teste para verificar o desempenho de diferentes funções de ativação aplicadas nas camadas ocultas, para duas camadas ocultas com cinco neurônios cada, dois de entrada e um de saída sendo a função de ativação da saída a sigmoide. Com taxa de aprendizado fixado em 0.1, épsilon em 0.001 e 3000 épocas.

Tabela 1 - comparativo funções de ativação

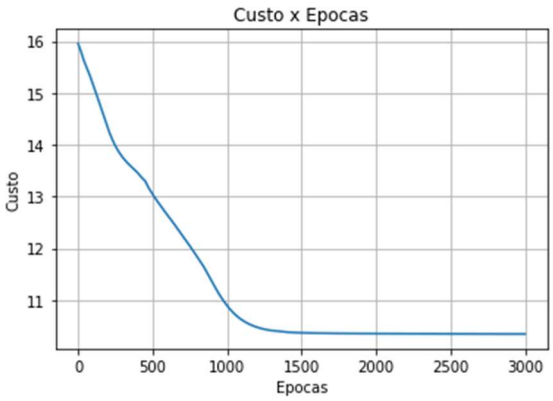
Precisão	ReLU	Tangente hiperbólica
Validação	75.00%	82.14%
Teste	85.00%	85.00%

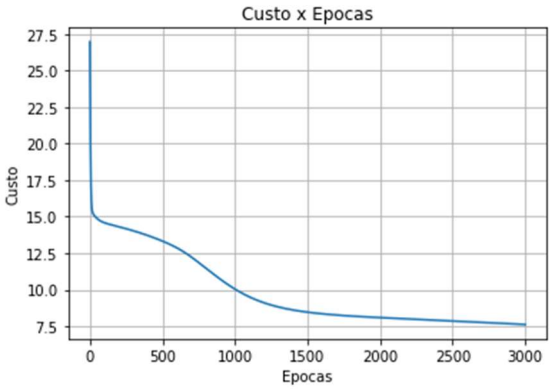
Para os resultados de acurácia x época para validação, utilizando tangente hiperbólica manteve-se numa crescente compassada, o método ReLU obtém uma alta acurácia na época inicial, porém sofre muita instabilidade ao passar das épocas atingindo constante em 75%, já a tangente obtém a constante em 85%.



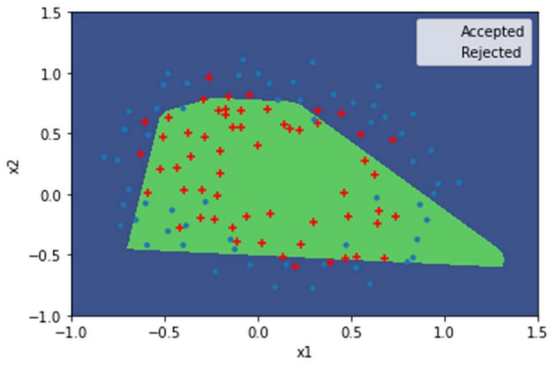


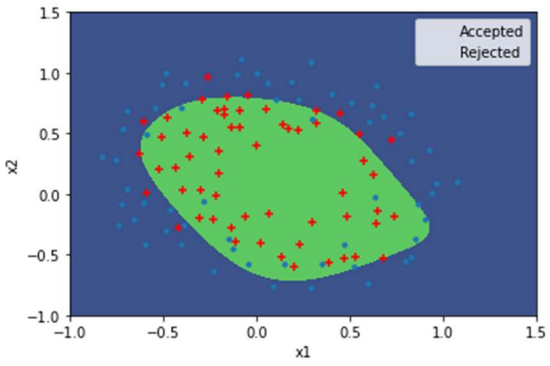
O custo inicia em 27 com a função da tangente e reduz para 15 logo no início dos testes enquanto a função ReLU já inicia com o custo em 16, ambas têm um decrescimento suave, porém a ReLU obtém estabilização já em 1500 épocas, e a tangente estabiliza em torno de 3000 épocas.





Por último o gráfico de classificação demonstra que para o conjunto de dados e o problema de classificação, a função tangente hiperbólica apresenta melhor adaptação se comparado ao ReLU.





A função de saída sigmoide é a que melhor se adapta a problemas de classificação, mas é necessário verificar a combinação com as funções nas camadas ocultas. De acordo com Assunção (2002), a função tangente hiperbólica não é adequada para redes com muitas camadas pois combinada com sigmoide pode surgir o problema de desaparecimento de gradiente, onde durante a regra da cadeia surge a derivada da função de ativação dissolvendo o máximo rapidamente. Para ReLU pode zerar a derivada parcial, mas o produto não será zero como na tangente, por ser idempotente, é o método mais utilizado em redes neurais.

Para verificar a influência de hiperparâmetros, foram verificados dois valores diferentes para cada um deles.

Tabela 2- comparativo mudanças nos hiperparâmetros

	Valor 1	Validação	Teste	Valor 2	Validação	Teste
Número de neurônios	10	82.14%	85.00%	3	35.71%	85%
Número de camadas	10	82.14%	80%	5	78.57%	75%
Taxa de aprendizado	0.1	82.14%	85%	0.9	75%	70%

A partir da tabela, utilizando como função de ativação ReLU, é possível perceber que quanto maior o número de camadas e de neurônios, melhor o desempenho do algoritmo, assim como uma menor taxa de aprendizado também melhora o desempenho. A função ReLU funciona melhor para redes com várias camadas e vários neurônios, então os resultados estão dentro dos esperados para o método.

4. CONSIDERAÇÕES FINAIS

Para dados não lineares o algoritmo de retropropagação busca através das derivadas parciais desenvolver soluções para cenários mais complexos, o desempenho dos algoritmos depende do tamanho e complexidade dos dados de entrada, memória utilizada e tempo de execução. Quando observado para várias camadas e muitos neurônios, o algoritmo exige bastante da máquina, o que é compreensível dado as conexões um a um da rede de neurônios, e a quantidade de substituições e buscas realizadas no código, que encarecem o custo de execução. Utilizado em larga escala, o método atende as expectativas no que se propõe, e com aperfeiçoamento no algoritmo implementado com funções de menor custo possam melhorar o desempenho.

A dificuldade de execução de testes em grupos maiores prejudica a avaliação mais coerente do método, considerando que a função de ativação ReLU tem melhor desempenho para grupos maiores, com mais camadas e neurônios enquanto a tangente hiperbólica tem melhor desempenho para grupos menores. Os testes confirmam os pressupostos na literatura da área.

5. REFERÊNCIAS

LILLICRAP, T. P. et al. Backpropagation and the brain. **Nature Reviews Neuroscience**, v. 21, n. 6, p. 335–346, 17 abr. 2020.

02 -AULA, S. **Deep Learning**. [s.l: s.n.]. Disponível em: <https://homepages.dcc.ufmg.br/~assuncao/AAP/Sem%2002%20-%20Aula%2002.pdf>. Acesso em: 20 out. 2022.

MCCARTHY, J. **What is AI? / Basic Questions**. Disponível em: <http://jmc.stanford.edu/artificial-intelligence/what-is-ai/index.html>.

ANEXO A – RELATÓRIO DE CONTRIBUIÇÃO

	Thays Franco	**Willian Benoski**
Inicialização	-	Implementação
Propagação Direta	-	Implementação
Retropropagação	Auxílio teórico	Implementação
Aproximação Gradiente	Implementação	Auxilio código
Gradiente Descendente	Implementação	Auxílio código
Testes	Realização	Discussão
Gráfico de classificação	Revisão	Realização
Relatório	Produção	Revisão