VII CONNEPI
Congresso Norte Nordeste de Pesquisa e Inovação
PALMAS . TOCANTINS . 2012
19 a 21 de outubro - Ciência, tecnologia e inovação: ações sustentáveis para o desenvolvimento regional

Abordagem a um classificador de dígitos manuscritos baseado em redes neurais

David Clifte da Silva Vieira[1], Renata Passos Machado[2]

[1]Graduando em Engenharia de Telecomunicações – IFCE. e-mail: cliftedavid@gmail.com
[2]Graduanda em Engenharia de Telecomunicações – IFCE. e-mail: re.passosm@gmail.com

Resumo: Este artigo descreve o uso de redes neurais e visão computacional para analisar imagens recebidas de um scanner, ou outra forma de digitalização de imagem, reconhecer um dígito manuscrito e classificá-lo de 0-9, ou indefinido, de acordo com o valor correspondente. As principais características do sistema são: Controlar o grau de acerto, um sistema de autoaprendizagem de fácil implementação e modularizado. O sistema se mostrou um ótimo exemplo da aplicação de redes neurais mostrando um acerto médio de 66%, e com uma das classes apresentando um acerto máximo de 85%.

Palavras–chave: dígitos manuscritos, OCR, redes neurais, visão computacional, Java

1. INTRODUÇÃO

OCR é o acrônimo de Optical Character Recognition, reconhecimento ótico de caracteres em português. Essa tecnologia permite que uma máquina possa reconhecer automaticamente um caractere através de um mecanismo óptico. As tentativas da engenharia em reconhecer caracteres impressos, ou manuscritos, iniciaram antes da Segunda Guerra Mundial, mas isso não foi possível até a década de 50, quando a associação dos Bancos e a Indústria dos serviços financeiros criaram fundos para a pesquisa e desenvolvimento da tecnologia [1].Diante da necessidade de uma abordagem mais simples aos temas de visão computacional e algoritmos de inteligência computacional, ambos necessários para aplicações como OCR, foi desenvolvido um sistema para uma boa exemplificação dos dois temas. Com a necessidade apenas de um conhecimento básico em Java, foi desenvolvido e aqui apresentado, um sistema classificador de dígitos manuscritos.

O MNIST é uma base de dados de dígitos manuscritos que tem um conjunto de 60.000 amostras de imagens prontas para o treinamento e mais 10.000 amostras que podem ser usadas para se fazer testes de eficiência de técnicas de classificação de dígitos manuscritos. O MNIST é um subconjunto das amostras do NIST (Instituto nacional de padrões e tecnologia). Os dígitos nessa amostra foram digitalizados, normalizados pelo tamanho e centralizados em uma imagem de tamanho fixo [2].

É uma boa amostra de dados, pois permite que técnicas de aprendizagem de máquina e reconhecimento de padrões sejam experimentadas com dados reais enquanto se economiza esforços em pré-processamento e formatação dos dados.

As imagens pretas e brancas, originalmente colhidas do NIST foram normalizadas para caberem em um quadrado de 20x20 pixels mantendo ainda a proporção do caractere. As imagens resultantes contêm uma variação em escala de cinza, devido a técnica antisserrilhamento aplicada na técnica de redimensionamento e normalização. Após a normalização, a imagem foi colocada no centro de uma janela de 28x28, usando como parâmetro o centro de massa do caractere.

A base de dados é composta por dois arquivos. O arquivo de imagens, onde cada linha do arquivo é uma imagem que sofreu o pré-processamento referido no paragrafo anterior, e o arquivo de identificadores, onde cada linha se refere ao valor do caractere inserido na linha de mesmo número do arquivo de imagens.

Computadores podem executar muitas operações com um tempo consideravelmente menor que os humanos poderiam fazer. Contudo, nem sempre, essa rapidez é a melhor escolha para resolver um problema. Muitas tarefas com as quais os computadores falham consideravelmente os humanos fazem melhor. Muitas dessas tarefas, nas quais os computadores perdem estão relacionadas à natureza interpretativa e de multi-processamento do cérebro. Uma maneira simples de caracterizar bem a

ISBN 978-85-62830-10-5

diferença entre o computador e o Homem seria comparar o computador, que é uma máquina serial, com o nosso cérebro, que é altamente paralelo e possui como característica principal a capacidade de aprender coisas [3].

A expressão rede neural, como é normalmente usada, é empregada de forma incompleta. Computadores tentam simular redes neurais biológicas através da implementação de redes neurais artificiais. Contudo, grande parte das publicações usam o termo rede neural ao invés do termo rede neural artificial (RNA).

Uma rede neural artificial é um paradigma baseado na forma biológica do sistema nervoso e de como ele processa a informação. O elemento mais importante desse paradigma é a forma como a informação é processada. A rede é composta por um grande número de elementos de processamento interconectados atuando juntos para resolver uma tarefa. RNAs, bem como pessoas, aprendem baseados em experiências passadas e ou exemplos. A rede normalmente é configurada para uma aplicação específica, como reconhecimento de padrões ou classificação de dados, através de um processo de treinamento. O sistema de aprendizagem biológico baseia-se em ajustar as conexões sinápticas que existem entre os neurônios. Esse modelo é o mesmo seguido pelas RNAs, como mostra a figura 1.

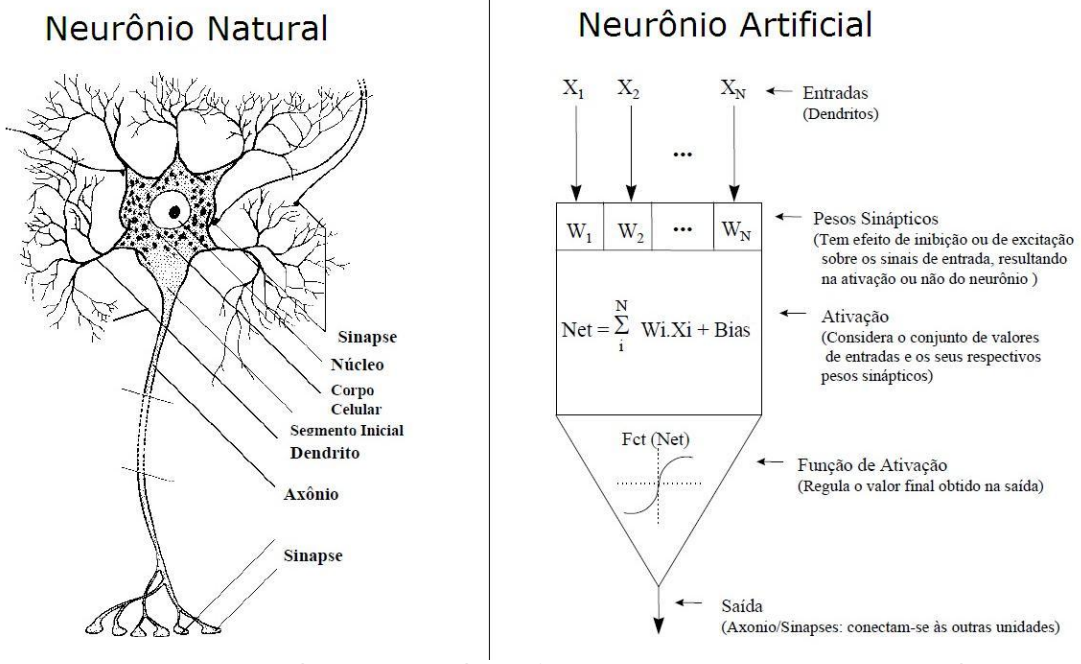

Figura 1 - Comparativo entre neurônio natural e neurônio artificial. As sinapses das conexões chegam ao neurônio pelos dendritos em ambos os modelos. Os pesos sinápticos junto com a função de ativação vão determinar se haverá uma sinapse de saída pelo axônio

Nesse projeto foi utilizada uma rede neural do tipo alimentação progressiva com retro propagação, frequentemente chamada de feedforward backpropagation. Esse tipo de arquitetura de rede neural é a mais popular, pois pode ser aplicada a várias tarefas diferentes. Para um melhor entendimento dessa arquitetura, examinou-se como é feito o treinamento e como ela processa um determinado padrão.

O termo alimentação progressiva descreve como esse tipo de rede neural processa e relembra os padrões para os quais ela foi treinada. Em uma rede neural desse tipo, todos os neurônios são conectados com a camada seguinte. Veja a figura 2 que exemplifica esse tipo de conexão.

ISBN 978-85-62830-10-5

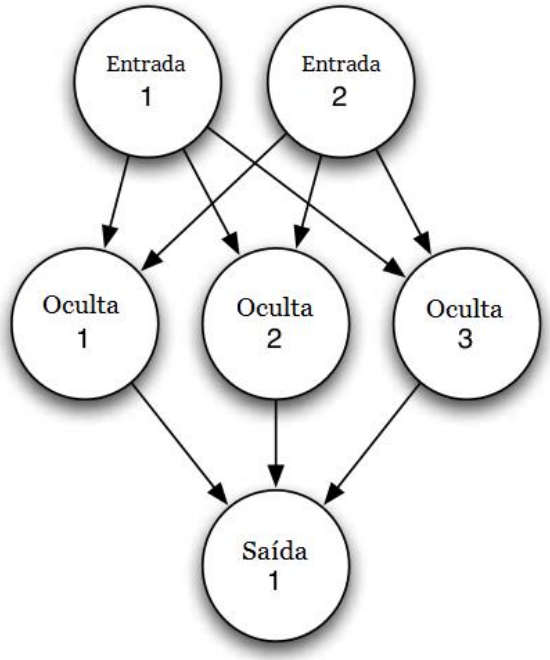

Figura 2 - Rede Neural de alimentação progressiva Nesse tipo de rede os neurônios são conectados somente a camada seguinte

O termo retro propagação descreve como esse tipo de rede neural é treinada. Quando é usado um treinamento supervisionado, uma rede neural deve apresentar tanto entradas quanto as saídas correspondentes para se realizar o treinamento. Essas saídas são chamadas de saídas antecipadas. A rede, inicialmente tem os pesos de cada camada ajustados com valores aleatórios. A rede então calcula as saídas baseadas nos pesos iniciais e então cada saída calculada é comparada com a saída antecipada, fornecida para a rede durante o treinamento. Essa comparação é feita através do cálculo do erro. Após esse cálculo, uma pequena alteração deve ser feita nos pesos da rede. O quão pequeno é essa alteração é chamado de taxa de aprendizagem. O ajuste dos pesos é feito da última camada para a primeira, isso origina o nome do algoritmo. Todo esse procedimento é executado para cada para entrada-saída fornecida ao algoritmo de treinamento. Ao final espera-se que o erro calculado entre os processos diminuam, e o algoritmo é dado como treinado após esse valor obtido seja menor que um dado limiar [3].

2. MATERIAL E MÉTODOS

O desenvolvimento do classificador ficou bem dividido em quatro fases, são elas: Coleta de dados, desenvolvimento de ferramentas para preparação dos dados, desenvolvimento do classificador e análise dos resultados.

Na primeira fase sugeriu-se que pessoas, escolhidas aleatoriamente, preenchessem um formulário contendo uma matriz em que as linhas representavam o número da amostra e as colunas os dígitos de 0-9. Após o preenchimento do formulário os dados seriam segmentados e passados para a fase seguinte. Porém surgiram problemas, como a capacidade de processar os formulários de forma eficiente. A quantidade de dados a ser processada superou as expectativas e a forma que os dados seriam pré-processados iria levar muito tempo, tanto de desenvolvimento quanto de importação dos dados. Tendo em vista um melhor aproveitamento dos esforços tanto computacional quanto de recursos humanos, optou-se por pesquisar uma base de dados já existente. O MNIST se mostrou melhor que outras bases encontradas ou coletadas, pois já havia um pré-processamento aplicado às imagens.

ISBN 978-85-62830-10-5

Na segunda fase o focou-se no desenvolvimento de ferramentas para auxiliar no progresso do sistema. A priori era inevitável desenvolver rotinas que decodificassem os arquivos da base de dados do MNIST, isso se fez necessário devido à forma que as imagens são armazenadas dentro de um único arquivo. Foi necessária a criação de dois métodos para ler os arquivos, um para converter o arquivo de identificadores, saídas antecipadas, e outro para o de imagens para um formato CSV padrão, como mostrado na figura 3.

```
public void convertIndexImages2CSV(File f,File out)
                                        throws IOException {
    DataInputStream dis = new DataInputStream(new FileInputStream(f));
    PrintWriter pw = new PrintWriter(out);

    magicNum = dis.readInt();
    nIm = dis.readInt();
    nRow = dis.readInt();
    nCol = dis.readInt();
    for(int i=0;i<nIm;i++){
        for(int a=0;a<nRow*nCol;a++)
            pw.printf("%d;", dis.read());

        pw.print("\n");
    }
    pw.close();
}
```

```
public void convertIndexLabels2CSV(File f,File out)
                                        throws IOException {
    DataInputStream dis = new DataInputStream(new FileInputStream(f));
    PrintWriter pw = new PrintWriter(out);

    magicNum = dis.readInt();
    nIm = dis.readInt();

    for(int i=0;i<nIm;i++){
        pw.printf("%d\n", dis.read());
    }
    pw.close();
}
```

Figura 3 – Métodos utilizados ambos implementados em Java

A fim de se obter mais flexibilidade e permitir o reaproveitamento de muitas das ferramentas desenvolvidas nesse trabalho o formato CSV foi escolhido.

Não se podiam levar as informações vindas da base de dados do jeito que estavam para a fase seguinte. A preparação tanto do arquivo de identificadores quanto do arquivo de imagens era necessária. Para o arquivo de identificadores era preciso: Ler o arquivo CSV e transformar cada entrada em um vetor no formato mostrado na figura 4. Após a formatação, montou-se cada saída em uma matriz onde cada linha era um identificador, porém já convertido em um vetor. Para o arquivo de imagens além da leitura do arquivo CSV, já convertido pela fase anterior, era necessário a normalização dos dados. Os dados gerados pelo MNIST continham valores variando na faixa de 0 a 255, esses valores foram normalizados entre 0 e 1 devido ao classificador suportar somente esse tipo de entrada.

Além da necessidade em formatar os dados para poder passar para o classificador, foi necessária a exibição das imagens, tanto da base de dados quando as imagens de teste. Para conseguir isso, foi necessário, a priori, converter uma imagem que está na forma de vetor para um arquivo de imagem. A situação inversa, converter arquivos de imagens em vetores, também era necessária, pois as imagens de teste poderiam ter qualquer origem, não somente do MNIST. Foi usada uma biblioteca de processamento de imagens para encurtar os esforços e manter o foco no classificador.

Valor	Saida	Valor	Saida
0	1000000000	5	0000010000
1	0100000000	6	0000001000
2	0010000000	7	0000000100
3	0001000000	8	0000000010
4	0000100000	9	0000000001

Figura 4 – vetor correspondente a cada entrada

ISBN 978-85-62830-10-5

O ponto chave da terceira fase do projeto foi o desenvolvimento do classificador. Após o recebimento dos dados da fase anterior o algoritmo foi preparado e projetado inicialmente para classificá-lo com uma taxa de aprendizagem e o momentum igual a 0,1. O parâmetro taxa de aprendizado tem uma enorme influência durante o processo de treinamento da rede neural. Uma taxa de aprendizado muito baixa torna o aprendizado da rede muito lento, ao passo que uma taxa de aprendizado muito alta provoca oscilações no treinamento e impede a convergência do processo de aprendizado. O momentum tem por objetivo aumentar a velocidade de treinamento da rede neural e reduzir o perigo de instabilidade. Foi utilizada uma biblioteca de aprendizagem de máquina para executar o treinamento, porém foi necessário a implementação de algumas rotinas a fim de o treinamento funcionar adequadamente.

A quarta fase será abordada na sessão seguinte como resultado.

3. RESULTADOS E DISCUSSÃO

Ao fim da fase anterior as amostras para o treinamento já poderiam ser preparadas para serem encaminhadas para o classificador. Foi usado inicialmente todo o banco de imagens para fazer a classificação, Porém o tempo de treinamento se tornou inviável. Cada época levava aproximadamente 25 minutos, foi feito um breve estudo para verificar que levaria cerca de um mês e meio para o algoritmo convergir com um erro de 1%. Optou-se por reduzir drasticamente a quantidade de amostras a serem usadas. 15 amostras de cada classe foram escolhidas ao acaso e colocadas para o classificador treinar.

Durante a fase de treinamento, informações como o erro e os arquivos de treinamento foram armazenados para que se fosse possível gerar estatísticas e acompanhar o treinamento. O tempo médio de treinamento para cada época foi de um minuto, o que fez com que o treinamento levasse aproximadamente 50 horas. A princípio a taxa de erro desejável era de 10% porém resolveu-se continuar para verificar o comportamento da curva de treinamento, como mostrado na figura 5.

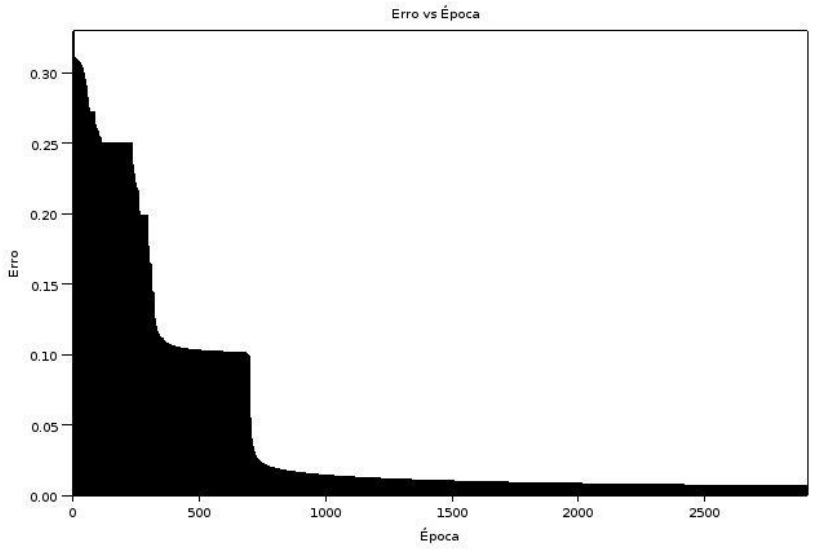

Figura 5 – Curva de comportamento do algoritmo de treinamento

É importante notar as regiões em que o gráfico apresenta certa constância. Nessas regiões enquanto observava-se os valores do erro do treinamento, chegou-se a inferir que o algoritmo tinha

ISBN 978-85-62830-10-5

convergido cedo demais e agiu-se de forma errada, mudando parâmetros e reverificando os dados de treinamento. Estudos mais tarde comprovaram que na verdade se tratavam de mínimos locais, nos quais nessas regiões o algoritmo levava mais tempo para convergir. A partir da milésima época, o erro foi considerado desprezível para a aplicação, porém continuou-se novamente o treinamento, a fim de verificar alguma anomalia. Nenhuma anomalia foi notada e o algoritmo continuou convergindo, porém de maneira significativamente mais lenta, tendo que ser interrompido o treinamento.

Após o treinamento era hora de testar a eficiência do algoritmo para o resto das imagens do arquivo de imagem. Com o arquivo de identificadores, já fornecido pelo MNIST, não era necessário uma validação humana dos resultados. Uma matriz confusão foi criada para um monitoramento do resultado, e assim poder explorar qualquer eventual falha de classificação.

Uma matriz confusão é um formato de tabela usado para visualizar o desempenho de um algoritmo, normalmente um de treinamento supervisionado. Cada linha na matriz representa uma das classes e cada coluna representa o valor que dada uma entrada o algoritmo de classificação julgou ser.

O resultado do classificador é mostrado na figura 6.

x	0	1	2	3	4	5	6	7	8	9	Desconhecido
0	84,80	0,00	1,33	1,12	0,20	2,35	1,12	0,51	0,20	0,00	8,37
1	0,00	82,20	0,26	0,09	2,82	0,00	0,00	0,09	9,34	0,00	5,20
2	1,45	0,97	60,66	4,46	4,75	0,19	0,68	0,48	6,49	0,78	19,09
3	4,75	2,87	8,61	47,03	0,89	2,08	0,40	4,06	2,57	1,09	25,64
4	0,00	1,83	0,10	0,00	76,48	0,31	0,71	0,00	0,10	12,42	8,04
5	5,04	2,24	3,03	2,02	1,91	46,97	1,57	2,24	5,16	2,24	27,58
6	1,57	0,63	3,24	1,04	2,30	1,57	67,33	0,10	2,09	0,10	20,04
7	0,00	3,50	0,29	2,04	1,56	0,00	0,78	68,48	0,29	10,60	12,45
8	1,64	2,36	4,41	0,62	3,59	5,85	0,21	0,41	60,57	0,51	19,82
9	0,50	2,28	0,20	0,40	9,61	1,49	0,40	3,07	0,10	67,89	14,07

Figura 6 – Matriz confusão

A diagonal principal da matriz confusão, ressaltada em verde, representa o acerto do algoritmo dado o valor da linha ou coluna. A exemplo disso nota-se que o acerto máximo foi de 84,80% na classificação do número zero e o acerto mínimo se encontra na classificação do número cinco. Além de informações como o acerto para determinada entrada, a matriz confusão exibe o grau com que determinados valores de entrada se confundem com os outros. Na matriz exibida vê-se que: Não há confusão entre zeros e uns, a confusão entre oitos e cincos também é zero por cento, entre nove e quatro o grau de confusão é de 9,61 por cento.

O algoritmo usado para classificação também permite classificar a entrada em desconhecida. O resultado é exibido na última coluna da matriz confusão, percebe-se através dessa coluna que o número de amostras usadas para o treinamento não foram suficientes para representar todas as classes principalmente a classe do número cinco que teve o pior resultado dentre as outras. Nessa classe vê-se que além do alto grau de confusão com as outras classes, 27% dos valores 5 foram identificados como desconhecidos. O acerto total do classificador foi de 66%.

4. CONCLUSÕES

O sistema apresentado mostrou-se um ótimo exemplo de como se usar uma rede neural para se fazer uma simples classificação de caracteres.

O resultado obtido após a classificação deixou claro que melhoras devem ser feitas principalmente na fase de treinamento. O uso de um treinamento simultâneo para tornar viável o uso de um maior número de amostras já está em fase de desenvolvimento, mostrando-se uma saída promissora para reduzir o tempo de aprendizagem do algoritmo.

ISBN 978-85-62830-10-5

REFERÊNCIAS

[1]HERBERT SCHANTZ, The History of OCR. Manchester Center, VT: Recognition Technologies Users Association, 1982.

[2]THE MNIST DATABASE**,** http://yann.lecun.com/exdb/mnist/
An introduction to neural computing. Aleksander, I. and Morton, H. 2nd edition

[3] Deolinda M. P. Aguieiras de Lima, http://mesonpi.cat.cbpf.br/naj/redesneurais.pdf/

[4]JEFF HEATON, Introduction to Neural Networks with Java, second edition

ISBN 978-85-62830-10-5