UNIVERSIDADE FEDERAL DE MATO GROSSO

CURSO DE ENGENHARIA DE COMPUTAÇÃO

Bruno Fernandes de Lima Santos

Relatório de atividade prática 5: Classificação de imagens com redes neurais artificiais

Cuiabá, 2017

Bruno Fernandes de Lima Santos

Relatório de atividade prática 5:

Classificação de números com redes neurais artificiais

Relatório técnico referente à atividade prática de número 5, da disciplina de Inteligência Artificial, curso de Engenharia de Computação, na Universidade Federal do Mato Grosso.

Prof. Dr. Raoni F. S. Teixeira

Cuiabá, 2017

RESUMO

O documento apresenta.

Palavras-chave: Rede. Redes. Neurais. Backpropagation. Custo.

SUMÁRIO

1 INTRODUÇÃO

Redes neurais artificiais (ou, abreviado, RNAs) são um tipo de classificador paramétrico não-linear, isto é, tratam de problemas de classificação onde os dados não necessariamente são linearmente separáveis. Atualmente são o tipo de classificador mais utilizados no mundo, sendo amplamente utilizadas na constituição de modelos de classificação multiclasse (não-binário).

A problemática deste trabalho é o desenvolvimento, treinamento e teste de uma rede neural artificial capaz de analisar dígitos manuscritos, armazenados em pequenas imagens de dimensão 20×20 pixels, e mapeá-los corretamente ao seu respectivo valor numérico. Por exemplo, na Figura 1.1 a RNA elaborada deve classificar tal imagem como o dígito "6".

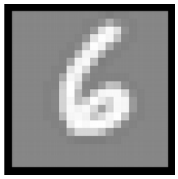

Figura 1.1: Exemplo de dígito manuscrito classificado pela RNA.

A base de dados contendo tais imagens, como as exibidas nas Figura 1.1 e Figura 1.2, é uma pequena amostra da base de dados do MNIST[1] de dígitos manuscrito. Ao todo 5000 imagens são utilizadas para treinamento e teste da rede neural desenvolvida neste experimento.

Naturalmente, como em qualquer modelo de classificação, erros ocorrerão. Sendo assim, o objetivo aqui é treinar a rede tal que a mesma possua uma acurácia de classificação relativamente alta (considere como alta uma acurácia superior a 90%). Mais detalhes sobre o treinamento da rede e os métodos de otimização utilizados estão contidos na seção 2.

1 MNIST. Disponível em: http://yann.lecun.com/exdb/mnist/. Acesso em: 11 abr 2017.

Figura 1.2: Cem imagens de exemplo de dígitos manuscritos.

2 IMPLEMENTAÇÃO E ARQUITETURA

A arquitetura da RNA aqui implementada constitui-se de três camadas e, consequentemente, tem-se duas matrizes de parâmetros entre as camadas. Sejam $A^{(l)}$ a matriz que denota os valores de saída da l-ésima camada e $\Theta^{(l)}$ a matriz dos parâmetros da rede, e que faz a conversão de valores entre as camadas l e $l+1$. A camada de entrada $A^{(1)} = X$ recebe cada um dos pixels de uma determinada imagem mais o valor de viés 1, totalizando 401 valores de entrada, isto é, $X = (1, x_1, x_2, ..., x_{400})$ é o vetor contendo os valores da entrada. A segunda camada $A^{(2)}$ possui 25 unidades mais o viés 1, e recebe os dados "filtrados" da camada de entrada através da multiplicação com a matriz de parâmetros $\Theta^{(1)} \in \mathbb{R}^{401\times26}$. A terceira e camada final da rede, $A^{(3)}$, possui 10 unidades (cada uma representando um dígito na faixa de 0 a 9) e recebe os dados da camada anterior pelo parâmetro $\Theta^{(2)} \in \mathbb{R}^{26\times10}$. A tabela a seguir resume a arquitetura utilizada na implementação desta RNA.

Componente	Notação(ões)	Dimensão
Camada de entrada	$A^{(1)}$, X	401×1
Camada intermediária	$A^{(2)}$	26×1
Camada final	$A^{(3)}$, h_Θ	10×1
Parâmetros de pesos 1	$\Theta^{(1)}$	401×26
Parâmetros de pesos 2	$\Theta^{(2)}$	26×10

Tabela 1: Arquitetura da RNA.

Note que a entrada é um vetor de 401 unidades e a saída da rede é um outro vetor de 10 unidades. Com isto é possível receber como entrada uma amostra de imagens em formato matricial $X \in \mathbb{R}^{401\times N}$, onde N é o número de imagens da amostra (uma imagem em cada vetor/coluna da matriz). Assim, a rede pode processar de uma vez uma amostra contendo N imagens e devolver uma matriz $h_\Theta \in \mathbb{R}^{10\times N}$ contendo as respectivas N respostas. Consequentemente, a dimensão da camada intermediária também muda, sendo $26 \times N$.

2.1 Função de limiar (threshold) e predição (predict)

Uma função de *soft threshold* foi utilizada na saída de cada neurônio da rede. Dentre algumas opções de funções que realizam *soft threshold*, foi utilizada a função sigmoide s, definida como:

$$s(Z) = \left\{ \frac{1}{1 + e^{-z_{i,j}}} \right\}_{i,j}, \forall z_{i,j} \in Z \tag{2.1}$$

Observe pela notação acima que, caso Z seja uma matriz, a função sigmoide é aplicada em cada componente da matriz, logo, $\dim(s(Z)) = \dim(Z)$.

2.2 Função de custo J

Um componente essencial em uma RNA é sua função de custo, isto é, a função que recebe o parâmetro Θ da rede e calcula o erro do modelo. A seguir tem-se a definição de uma função de custo J com regularização.

$$J(\Theta^{(1)}, \Theta^{(2)}) = (C + R)(\Theta^{(1)}, \Theta^{(2)}) \tag{2.2}$$

Onde C é o custo bruto da rede e R é a regularização da rede (prevenindo sobreajuste, ou *overfitting*, dos dados). Ambos são definidos como:

$$C(\Theta^{(1)}, \Theta^{(2)}) = \frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} \left(-y_i^k \log(h_\Theta(x_i)_k) - (1 - y_i^k) \log(1 - h_\Theta(x_i)_k) \right) \tag{2.3}$$

$$R(\Theta^{(1)}, \Theta^{(2)}) = \frac{\lambda}{2m} \left(\sum_{i=1}^{25} \sum_{j=1}^{400} (\theta_{i,j}^{(1)})^2 + \sum_{i=1}^{10} \sum_{j=1}^{25} (\theta_{i,j}^{(2)})^2 \right), \forall \theta_{i,j}^{(l)} \in \Theta^{(l)} \tag{2.4}$$

Um dos objetivos deste experimento foi implementar a mesma função acima no MATLAB.

2.3 Vetor gradiente de J

O outro objetivo aqui foi obter, também em MATLAB, o vetor gradiente do custo ∇J, necessário para a otimização da função de custo. O algoritmo necessário para obtenção do vetor gradiente é o algoritmo Backpropagation.

3 EXPERIMENTOS UTILIZANDO A REDE NEURAL

Texto exemplo.

3.1 Experimento 1: Testando implementação da rede

3.1.1 Parte I: Função de custo J

3.1.2 Parte II: Obtenção do grad J via Backpropagation

3.2 Experimento 2: Treinamento da rede

3.3 Experimento 3: Treinamento e teste de acurácia

4 RESULTADOS E DISCUSSÃO

4.1 Experimento 1: Testando implementação da rede

4.1.1 Parte I: Função de custo J

4.1.2 Parte II: Obtenção do grad J via Backpropagation

4.2 Experimento 2: Treinamento da rede

4.3 Experimento 3: Treinamento e teste da rede com validação cruzada

5 CONCLUSÃO