Trabalho de Sistemas Inteligentes
Implementação de rede perceptron multicamada
Gabriel Martins de Miranda – 13/0111350

Entrada do programa: taxa de aprendizagem, número de iterações para treino, número de neurônios na camada escondida, função de ativação.

Database: base de dados "MNIST", disponível em http://yann.lecun.com/exdb/mnist/
Set de treino: 60000 imagens de 28x28 pixels
Set de teste: 10000 imagens de 28x28 pixels

O código foi baseado no exercício 4 do curso Machine Learning de Andrew Ng.

1) Initialization

Todas as variáveis são liberadas e gui fechadas.

2) Parameters setup

Aqui são definidos os parâmetros fixos e variáveis da rede. Como são usadas na entrada imagens 28x28, as variáveis de entrada da rede correspondem a 784 valores. Além disso, a saída também é fixa, sendo 10 neurônios correspondentes aos dígitos de 0 a 9.

Pede-se como entrada do usuário os valores da taxa de aprendizagem, número de iterações para treino, número de neurônios na camada escondida e a função de ativação, que pode ser logística ou tangente hiperbólica.

3) Loading and visualizing the training data

Nessa parte do código, é carregado o set de treino da base MNIST e selecionado randomicamente 100 exemplos para ser mostrado na tela.

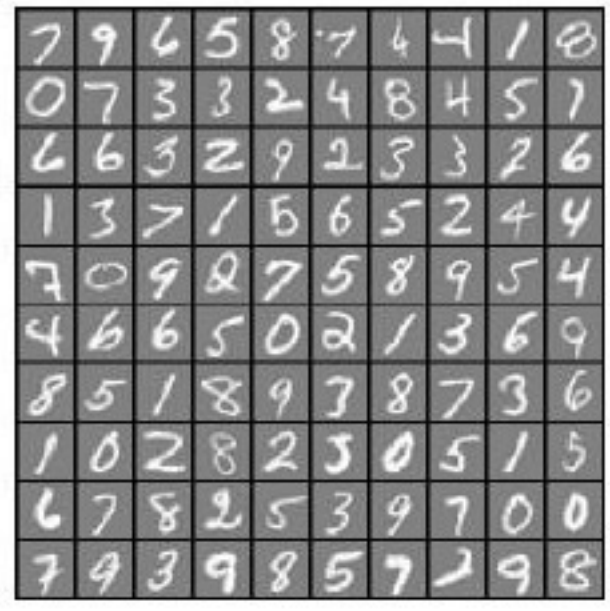


Figura 1. Alguns exemplos do set de treino MNIST

X

60000x784 double

	10	541	542	543	544	545	546	547	548	549	550	551	552	553
3351	0	0	0	0	0	0.2745	0.9922	0.3176	0	0	0	0	0	0
3352	0.0471	0.5725	0.9804	0.9922	0.9608	0.4902	0.0431	0	0	0	0	0	0	0
3353	0	0	0	0.2824	0.2784	0.2784	0.2784	0.7843	0.7882	0.2784	0.2784	0.2784	0.2784	0.4863
3354	0.8353	0.7373	0.1569	0	0.3137	0.9608	0.6588	0.0235	0	0	0	0	0	0
3355	0	0	0.1725	0.8471	0.9961	0.9961	0.9961	0.4000	0	0	0	0	0	0
3356	0	0.1490	0	0	0	0	0	0	0	0	0	0	0	0
3357	0	0.7804	0.9922	0.7098	0	0	0	0	0	0	0	0	0	0
3358	0.9922	0.9922	0.9922	0.9176	0.6039	0.2471	0.1176	0.0275	0	0.0431	0.8667	0.9922	0.9922	0.9412
3359	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3360	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3361	0.1608	0	0	0	0	0	0	0	0	0	0	0	0	0
3362	0.9922	0.9882	0.4784	0	0	0	0	0	0	0	0	0	0	0
3363	0	0	0	0	0	0	0	0	0	0	0.3255	0.9922	0.9922	0.9922

y

60000x1 double

	1
1	5
2	10
3	4
4	1
5	9
6	2
7	1
8	3
9	1
10	4
11	3
12	5
13	3

Figura 2. Entradas e saídas conhecidas carregadas.

As variáveis de entrada foram escaladas entre 0 e 1. Como o matlab não possui índice 0, o dígito correspondente ao índice zero do valor esperado foi substituido por 10 para facilitar a computação.

Para que fosse carregado o set no programa, uma função auxiadora foi utilizada. Esta está disiponível em :
http://ufldl.stanford.edu/wiki/index.php/Using_the_MNIST_Dataset

4) Initializing parameters

Nesta parte são gerados os valores para os pesos iniciais da rede. Como a rede é composta de duas camadas (a escondida e a de saída), duas matrizes de pesos são necessárias, a que representa os pesos da entrada para a camada escondida, e da camada escondida para a camada de saída. As dimensões dessas matrizes são [hidden_layer_size , input_layer_size + 1] e [num_labels , hidden_layer_size + 1] respectivamente, sendo o termo unitário correspondente ao bias.

Seus valores foram gerados de forma randômica dentro do intervalo [-Einit, Einit], em que Einit é definido como:

$$\epsilon_{init} = \frac{\sqrt{6}}{\sqrt{L_{in}+L_{out}}}$$

Sendo Lin e Lout o número de unidades na camada adjacente ao peso.

5) Training NN

Aqui é realizado o treino propriamente dito, sendo o feedfoward e o backpropagation. Tanto o backpropagation quando a função custo foram regularizadas.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left[-y_k^{(i)}\log((h_\theta(x^{(i)}))_k) - (1-y_k^{(i)})\log(1-(h_\theta(x^{(i)}))_k)\right] + \frac{\lambda}{2m}\left[\sum_{j=1}^{25}\sum_{k=1}^{400}(\Theta_{j,k}^{(1)})^2 + \sum_{j=1}^{10}\sum_{k=1}^{25}(\Theta_{j,k}^{(2)})^2\right].$$

Figura 3. Função custo. Neste caso, são 400 variáveis de entrada com 25 neurônios na camada escondida e 10 na de saída.

$$\frac{\partial}{\partial\Theta_{ij}^{(l)}}J(\Theta) = D_{ij}^{(l)} = \frac{1}{m}\Delta_{ij}^{(l)} \qquad \text{for } j = 0$$

$$\frac{\partial}{\partial\Theta_{ij}^{(l)}}J(\Theta) = D_{ij}^{(l)} = \frac{1}{m}\Delta_{ij}^{(l)} + \frac{\lambda}{m}\Theta_{ij}^{(l)} \qquad \text{for } j \geq 1$$

Figura 4. Backpropagation com gradiente regularizado.

6) Visualizing cost and weight

Para efeitos de vizualização, são plotados os gráficos de custo por iteração e a vizualização dos pesos da rede

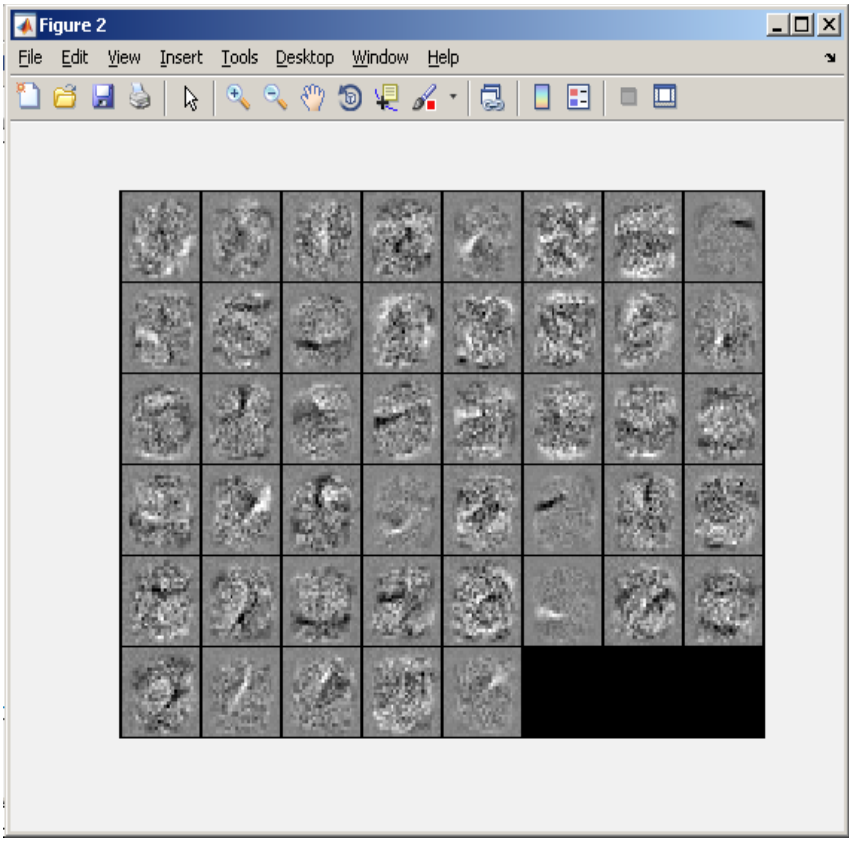


Figura 5. Vizualização dos pesos após treinamento.

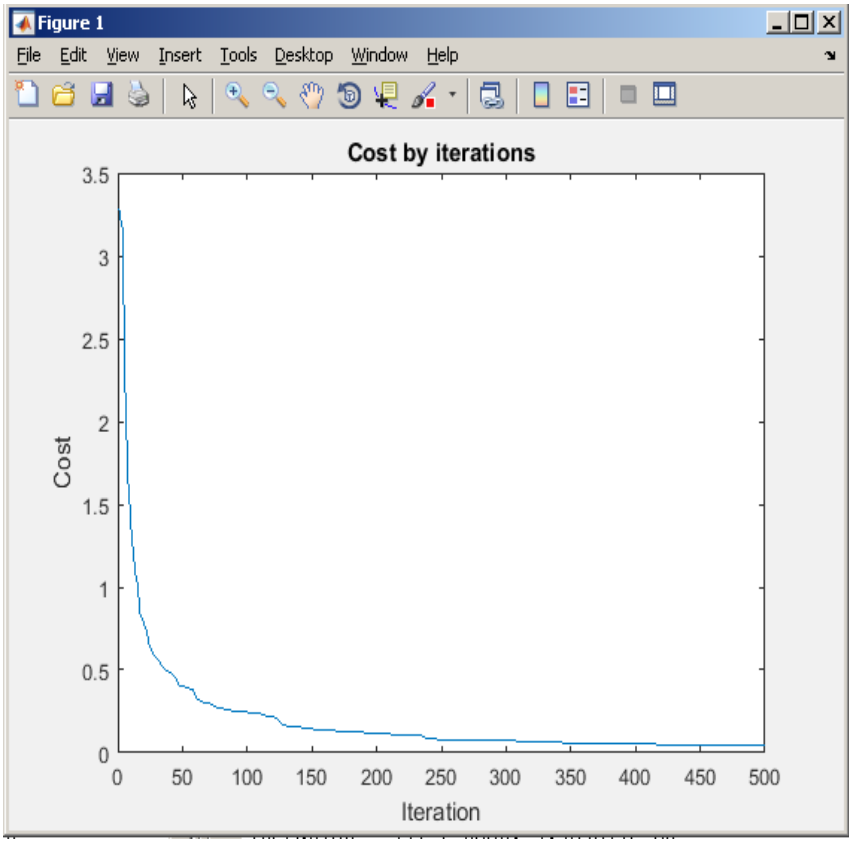


Figura 6. Vizualização do custo por iterações. Nesse caso foram usadas 500 iterações.

7) Loading and visualizing the test data

Aqui é carregado o set de teste que será usado para validar a a taxa de acerto da rede.

8) Predict

Aqui é usada a rede para predizer os resultados tanto do set de treino quanto do set de teste. Resultados ao final.

9) Randomly permute examples

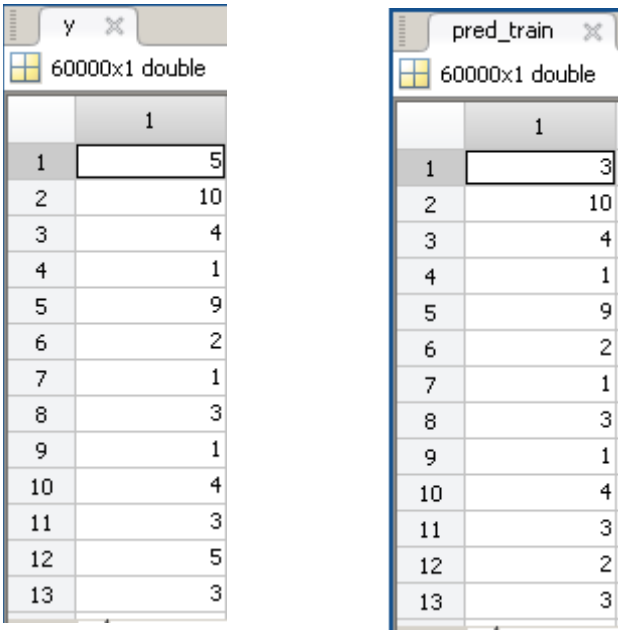
São selecionadas imagens aletórias do set de teste e mostradas na tela junto com a predição da rede.

10) Resultados

Treino 1

→ Com taxa = 0.1
→ iterações = 15
→ neurônios na camada escondida = 25
→ logistica

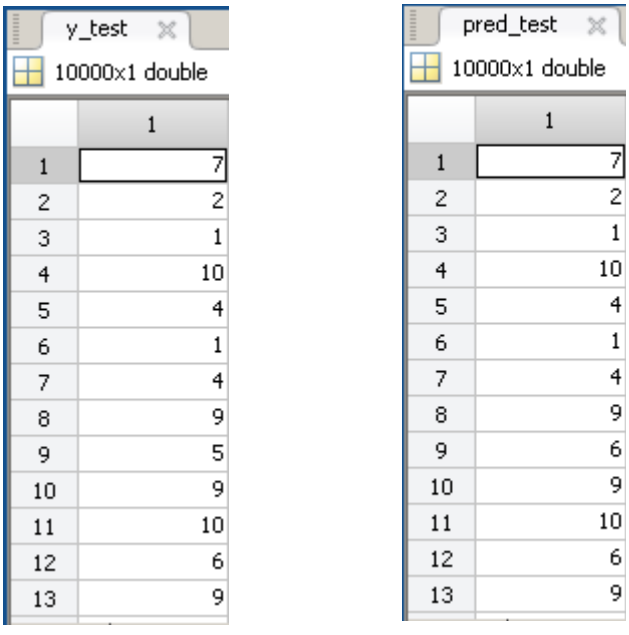
Training Set Accuracy: 83.535000
Test Set Accuracy: 84.470000



y	
60000x1 double	
	1
1	5
2	10
3	4
4	1
5	9
6	2
7	1
8	3
9	1
10	4
11	3
12	5
13	3

pred_train	
60000x1 double	
	1
1	3
2	10
3	4
4	1
5	9
6	2
7	1
8	3
9	1
10	4
11	3
12	2
13	3

Figura 7. Alguns resultados entre os valores esperados e os encontrados pela rede no set de treino.



y_test	
10000x1 double	
	1
1	7
2	2
3	1
4	10
5	4
6	1
7	4
8	9
9	5
10	9
11	10
12	6
13	9

pred_test	
10000x1 double	
	1
1	7
2	2
3	1
4	10
5	4
6	1
7	4
8	9
9	6
10	9
11	10
12	6
13	9

Figura 8. O mesmo para o set de teste.

Treino 2

→ Com taxa = 0.1
→ iterações = 500
→ neurônios na camada escondida = 45
→ logistica

Training Set Accuracy: 99.885000
Test Set Accuracy: 96.930000

Treino 3

→ Com taxa = 0.32
→ iterações = 100
→ neurônios na camada escondida = 25
→ logistica

Training Set Accuracy: 95.646667
Test Set Accuracy: 94.830000

11) Considerações

O código foi realizado no Matlab R2015a numa máquina windows 7 com 4GB de RAM e processador intel core i3.