Trabalho de Sistemas Inteligentes
Problema MNIST com pré-processamento auto-organizado

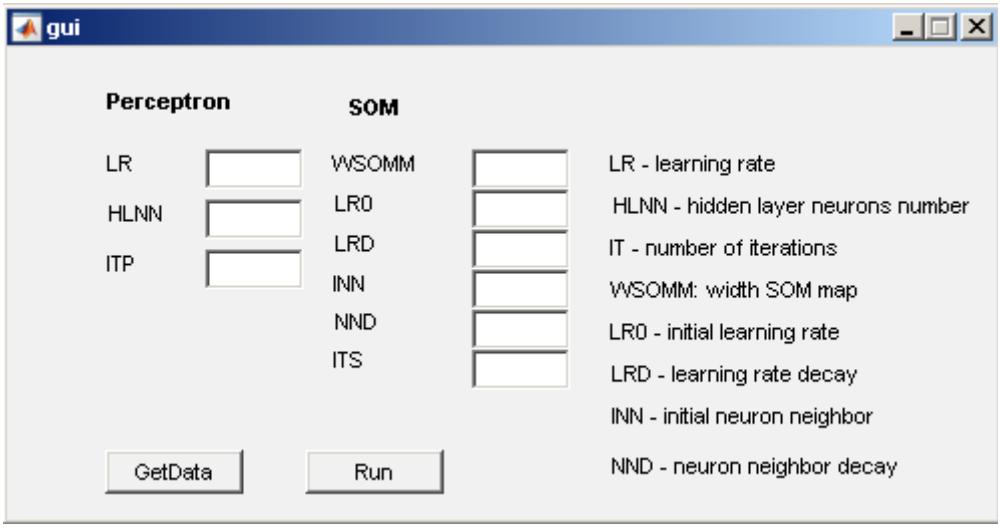
Gabriel Martins de Miranda – 13/0111350
Fernanda Amaral Melo – 14/0019855

O proposto código treina uma rede de Kohonen e usa suas saídas Y como entrada para a rede perceptron.

Por motivos de poder computacional, foi selecionada uma porção da base de dados MNIST de mil exemplos, sempre que a base possui sessenta mil.

A maior parte dos parâmetros tanto da rede de Kohonen quanto da rede Perceptron são ajustáveis.

1) GUI



Através da interface com o usuário é possível setar os parâmetros da rede Perceptron (à esquerda) e da rede Kohonen (à direita).

1.1) Perceptron

→ LR – taxa de aprendizado.
→ HLNN – número de neurônios na camada escondida.
→ ITP – número de iterações para treino.

1.2) Kohonen

→ WSOMM – dimensão do mapa de kohonen. Decidimos fazê-lo como uma matriz quadrada.
→ LR0 – taxa de aprendizado inicial.

→ LRD – decaimento da taxa de aprendizado.

→ INN – tamanho de vizinhança inicial de neurônios. Uma gaussiana centrada no neurônio vencedor (bmu – best matching unit) é usada para definir a vizinhança na iteração. Assim, o parâmetro é o sigma inicial dessa função gaussiana.

→ NND – decaimento da vizinhança.

→ ITS – número de iterações para treino.

1.3) Botões

1.3.1) GetData

Carrega os dados usados para treino das redes (MNIST) e inicializa todos os parâmetros configuráveis com valores padrões. Deve ser o primeiro botão utilizado pelo usuário.

1.3.2) Run

Após inicializar todos os dados com GetData, o usuário pode iniciar o processo de treinamento e predição das redes ou alterar o valor de variáveis.

1.3.3) Campos vazios

Para alterar um campo, após pressionado o GetData, o usuário deve inserir um valor numérico no campo vazio e pressionar Enter. Para saber se o valor foi devidamente alterado, pode-se olhar no Workspace seu valor.

2) Arquivos

2.1) gui.m

Para executar o programa, o usuário deve executar gui.m. Aqui a GUI é carregado, sendo definidas todas as variáveis e chamadas as redes de treinamento kohonen e perceptron.

2.2) som.m

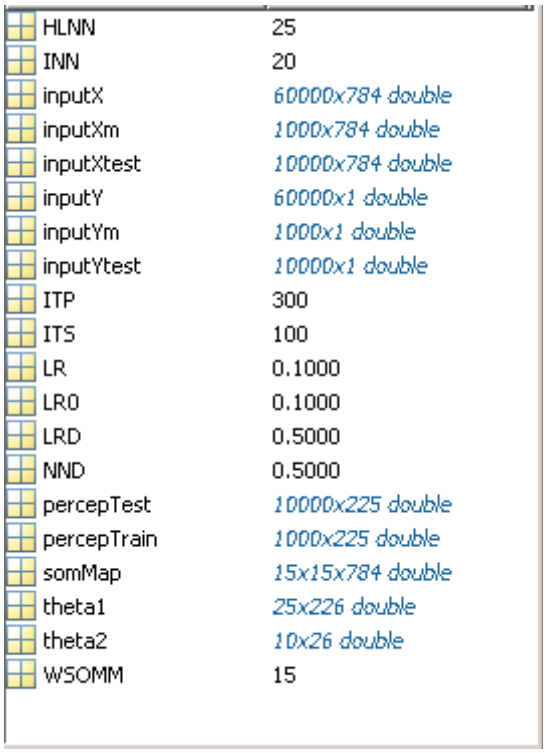
Aqui são definidas todas as funções do mapa de kohonen. Para obtenção do neurônio vencedor foi usada a função mandist padrão do matlab, que calcula a distância de manhatan entre dois vetores. Ela foi preferida devido ao alto custo computacional da distância euclidiana padrão para vetores de altas dimensões. Para cálculo da vizinhança, foi usada uma função gaussiana, que a cada época tem sua base achatada com a diminuição do valor sigma. A cada época é mostrada a organização dos neurônios na rede de acordo com o valor de entrada que mais se aproxima.

2.3) multilayerPerceptron.m

Aqui são definidas todas as função do perceptron multicamada implementado no trabalho 1. Ao final do treinamento é possível vizualizar o custo pelas iterações.

3) Resultados

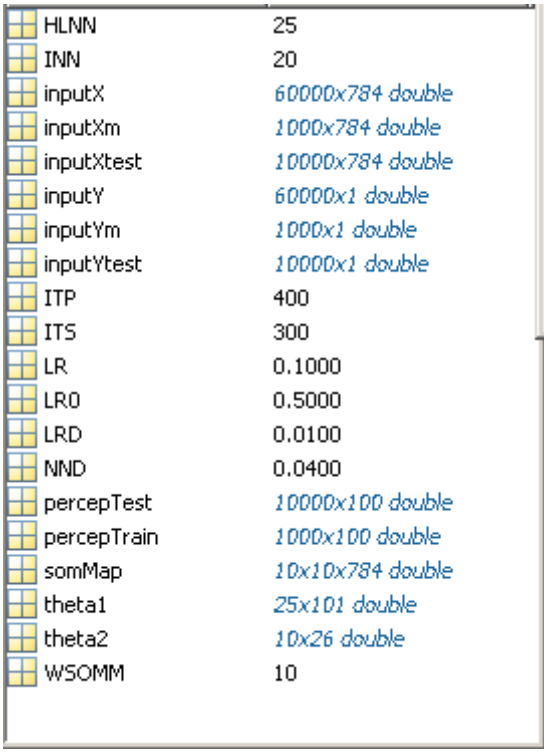
Treino 1



HLNN	25
INN	20
inputX	*60000x784 double*
inputXm	*1000x784 double*
inputXtest	*10000x784 double*
inputY	*60000x1 double*
inputYm	*1000x1 double*
inputYtest	*10000x1 double*
ITP	300
ITS	100
LR	0.1000
LR0	0.1000
LRD	0.5000
NND	0.5000
percepTest	*10000x225 double*
percepTrain	*1000x225 double*
somMap	*15x15x784 double*
theta1	*25x226 double*
theta2	*10x26 double*
WSOMM	15

```
Training Set Accuracy: 54.100000
Test Set Accuracy: 47.080000
```

Treino 2



HLNN	25
INN	20
inputX	*60000x784 double*
inputXm	*1000x784 double*
inputXtest	*10000x784 double*
inputY	*60000x1 double*
inputYm	*1000x1 double*
inputYtest	*10000x1 double*
ITP	400
ITS	300
LR	0.1000
LR0	0.5000
LRD	0.0100
NND	0.0400
percepTest	*10000x100 double*
percepTrain	*1000x100 double*
somMap	*10x10x784 double*
theta1	*25x101 double*
theta2	*10x26 double*
WSOMM	10

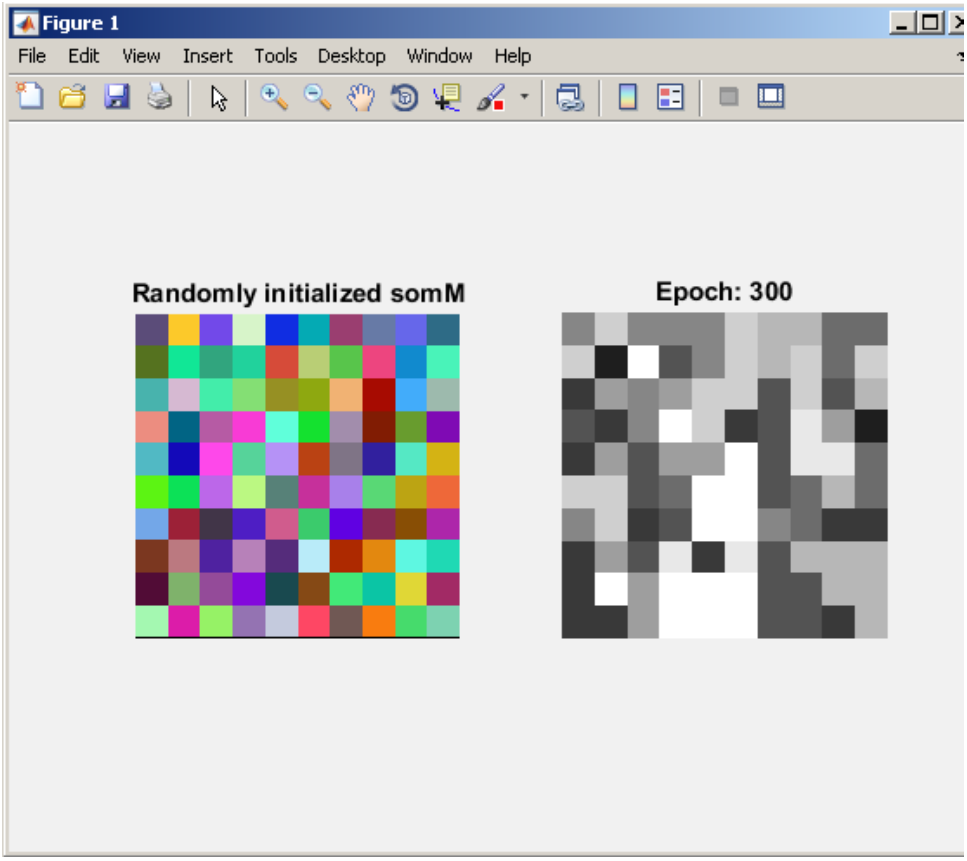


Figure 1
File Edit View Insert Tools Desktop Window Help
Randomly initialized somM
Epoch: 300

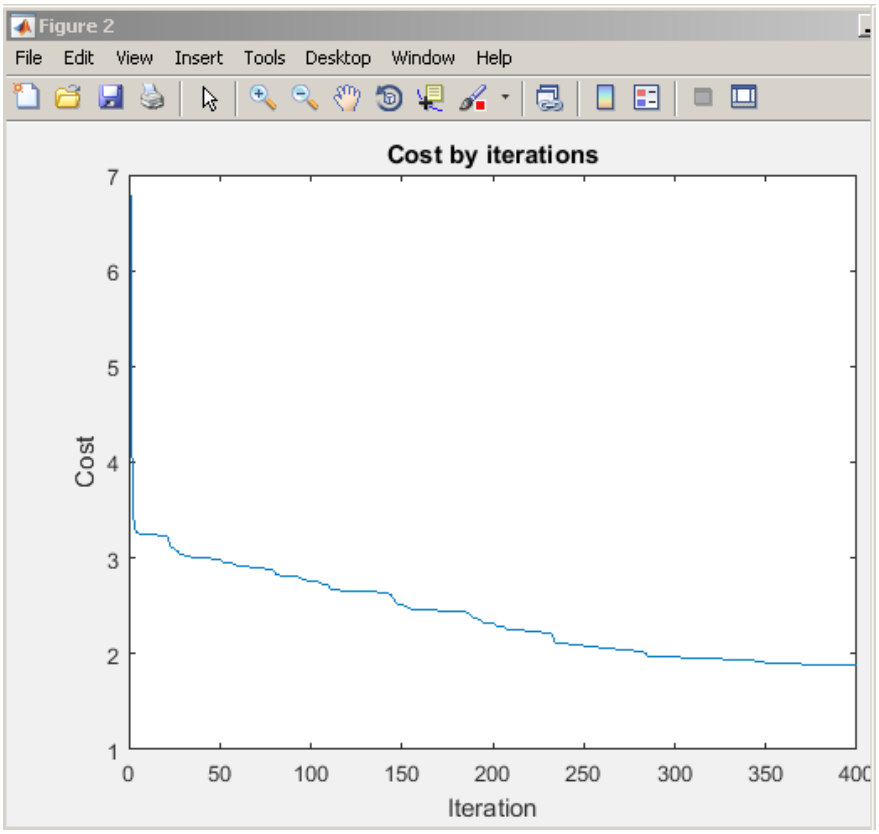
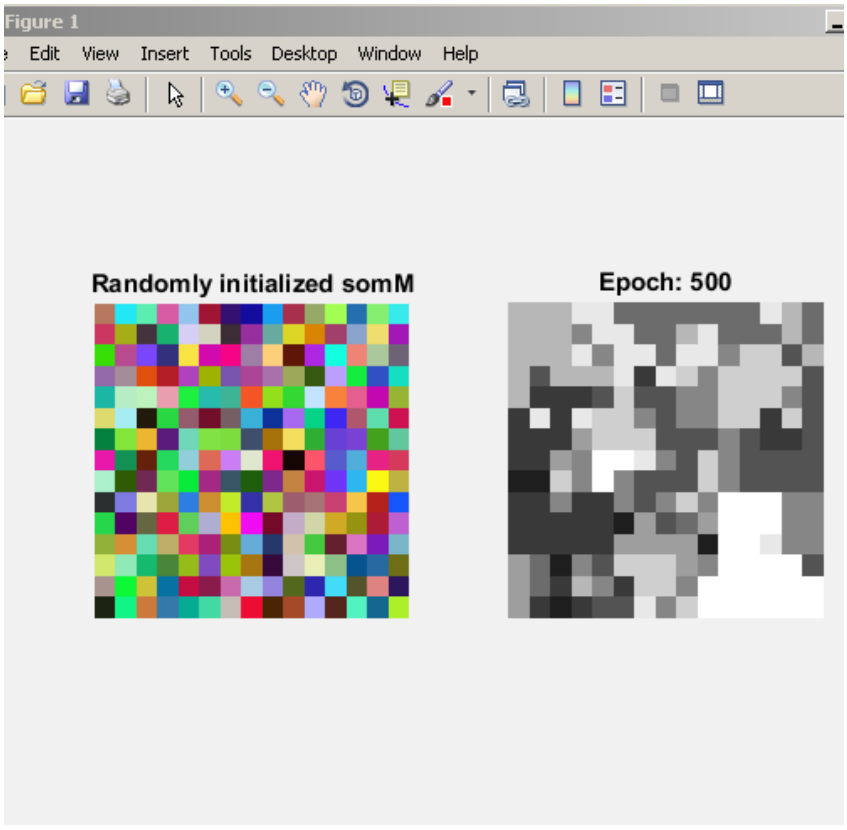


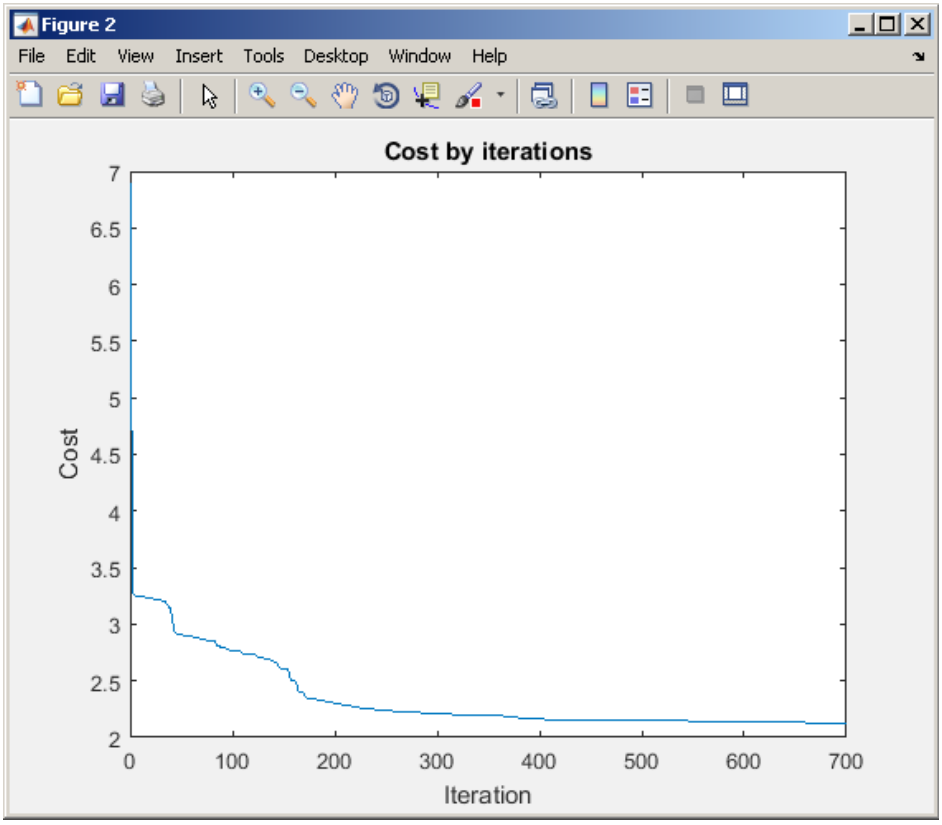
Figure 2
File Edit View Insert Tools Desktop Window Help
Cost by iterations
Cost
Iteration
1
2
3
4
5
6
7
0
50
100
150
200
250
300
350
400

```
Training Set Accuracy: 58.200000
Test Set Accuracy: 51.000000
```

Treino 3

HLNN	25
INN	20
inputX	*60000x784 double*
inputXm	*1000x784 double*
inputXtest	*10000x784 double*
inputY	*60000x1 double*
inputYm	*1000x1 double*
inputYtest	*10000x1 double*
ITP	700
ITS	500
LR	0.1000
LR0	0.5000
LRD	0.0100
NND	0.0200
percepTest	*10000x225 double*
percepTrain	*1000x225 double*
somMap	*15x15x784 double*
theta1	*25x226 double*
theta2	*10x26 double*
WSOMM	15






```
Training Set Accuracy: 47.100000
Test Set Accuracy: 38.120000
```

4) Considerações

Apesar de otimizar parâmetros, os resultados nem sempre melhoraram com o treinamento da rede.