Sistemas Inteligentes
Prof. Elder Rizzon Santos
Alunos: Pablo Vicente, Rolf Zambon, Roberto Rivelino
Universidade Federal de Santa Catarina
Sistemas de Informação

Trabalho sobre Redes Neurais 2019-2

1. Descrever brevemente o problema e qual o propósito da utilização de uma rede neural nesse contexto.

Escolhemos a rede neural do Tensor Flow Fashion MNIST por ser um ótimo exemplo para quem está iniciando na área, e por ser brevemente mais complexo que o MNIST, que é normalmente utilizado como o "Hello World" das redes neurais.

O problema que o Fashion MNIST busca solucionar é o reconhecimento de tipos diferentes de roupas, como tênis e camisetas, e a dificuldade no aprendizado dos iniciantes e entusiastas na área de redes neurais. O propósito do algoritmo é desenvolver uma rede neural que reconheça diversas roupas, para isso, é utilizando como exemplo 60.000 imagens e 10.000 imagens teste.

2. Descrever a modelagem das entradas.

Para as entradas são utilizadas duas listas de treinamento, uma para imagens e outra para labels, com 60.000 entradas cada, e outras duas para teste, ambas com 10.000 entradas.

As imagens presentes nos treinamentos e testes possuem:

- A lista de imagens, um array de matrizes, onde cada imagem é representada por uma matriz 28x28, e cada célula da matriz, podendo ser um valor entre 0-255, indicando a escala de cinza do pixel.

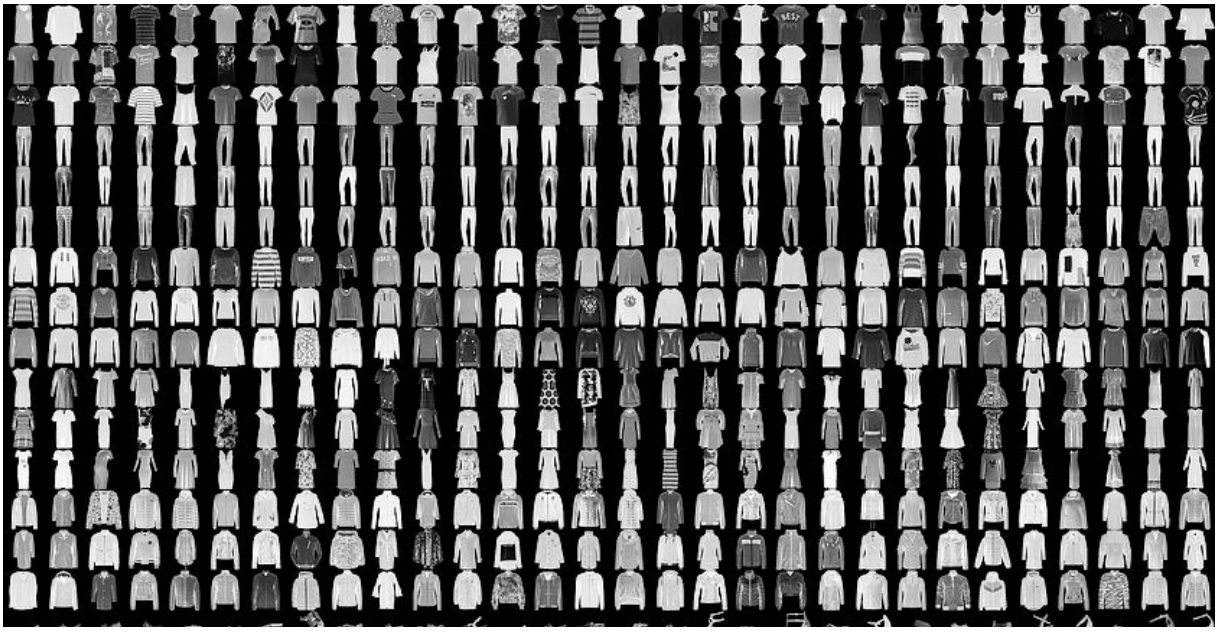

Figura 1. Amostra do conjunto de imagens

Antes da lista de imagens ser passada para a rede ela é pré-processada, dividindo todos os valores por 255.0, alterando a escala de suas possíveis cores de 0-255 para 0-1.

- A lista de labels, um array de inteiros, onde seus valores podem ser de 0 a 9, e cada valor representa um tipo de roupa:

Label	Classe
0	Camisetas/Top (T-shirt/top)
1	Calça (Trouser)
2	Suéter (Pullover)
3	Vestidos (Dress)
4	Casaco (Coat)
5	Sandálias (Sandal)
6	Camisas (Shirt)
7	Tênis (Sneaker)
8	Bolsa (Bag)
9	Botas (Ankle boot)

Figura 2. Labels possíveis para as imagens

3. Explicar a topologia da rede neural (existem vários tipos de redes, ex.: feedforward, convolutional, cíclica, ...) – qual é o tipo utilizado e qual a sua aplicação principal.

Há 3 camadas nessa rede:

1. Camada Flatten, onde é formatado os dados que serão seguidos nas próximas camadas.
   Nessa camada é feito o "achatamento" das imagens, transformando-as de matrizes 28x28 em vetores de 784 valores.
2. Camada Densa Escondida, onde há 128 neurônios, e é a responsável pelos cálculos ocultos com ajustes de pesos da rede.
3. Camada Densa de Saída, onde há 10 neurônios, também com ajustes de pesos, porém agora cada um representando a probabilidade da entrada ser um possível tipo de roupa.

Como há 3 camadas e o fluxo contínuo consiste em passar a entrada para a camada escondida, ocorrendo o processamento pesado, e depois para a saída, de modo que sempre todos os dados das camadas anteriores são passados para a camada seguinte, a topologia utilizada pode ser classificada como **feedforward.**

4. Explicar a escolha das funções de saída e ativação.

A primeira camada Densa, que é a camada escondida e contêm 128 nós, utiliza a **função de ativação ReLu**, uma função simples e ágil, ideal para o contexto do problema. Sua definição é *ReLU(x) = max{0, x}* (Se a soma do neurônio foi maior que 0 é ela que passa pra frente, se não é o valor 0). É uma função eficiente e eficaz, pois permite ajustes no erro de camadas anteriores, possuindo um intervalo de saída contínuo, e não 0 ou 1, portando também um cálculo extremamente rápido.
(https://matheusfacure.github.io/2017/07/12/activ-func/#relu)

Já a segunda camada Densa, que é a camada de saída que contém 10 nós, utiliza a **função de ativação SoftMax**, pois é a função ideal para saída de redes neurais de classificação.
A função softmax transforma as saídas para cada classe com valores entre 0 e 1, agrupando-as e dividindo-as pela soma das saídas, resultando no valor 1. Isso essencialmente dá a probabilidade, entre 0 e 1, de a entrada estar em uma determinada classe, visto que a soma de todas as saídas totaliza em 1.
(http://deeplearningbook.com.br/funcao-de-ativacao/)

5. Descrever qual foi o algoritmo de aprendizagem utilizado juntamente com a descrição do conjunto de testes e do conjunto de treinamento.

Foi utilizado o **algoritmo de aprendizagem supervisionada**.

O conjunto de treinamento e de testes é o Fashion MNIST, que foi importado diretamente do TensorFlow. Foi utilizado 60.000 entradas para o treinamento e 10.000 entradas para testes, que foram descritos na resposta da questão 2.

6. Verifique o desempenho* da RN com uma quantidade diferente de camadas ocultas, com relação à proposta no tutorial ou rede original.

A proposta no tutorial utiliza uma camada oculta com 128 neurônios, e 10 épocas. O tempo de treinamento foi de 31,87s. A acurácia no conjunto de treinamento foi de 91,11%, e no conjunto de teste foi de 87,65%.

```
Epoch 9/10
60000/60000 [==============================] - 3s 54us/sample - loss: 0.2477 - accuracy: 0.9072
Epoch 10/10
60000/60000 [==============================] - 3s 54us/sample - loss: 0.2383 - accuracy: 0.9111
10000/1 - 0s - loss: 0.3315 - accuracy: 0.8765

Test accuracy: 0.8765

Tempo de treinamento: 31.87199115753174
```

Figura 3. Saída obtida ao treinar a rede com as configurações originais

Adicionamos mais 2 camadas ocultas, uma com 256 neurônios e outra com 128, e mantemos as 10 épocas. O tempo de treinamento foi de 43,01s. A acurácia no conjunto de treinamento foi de 91,22%, e no conjunto de teste foi de 88,35%.

```
Epoch 8/10
60000/60000 [==============================] - 4s 70us/sample - loss: 0.2501 - accuracy: 0.9059
Epoch 9/10
60000/60000 [==============================] - 4s 71us/sample - loss: 0.2404 - accuracy: 0.9100
Epoch 10/10
60000/60000 [==============================] - 4s 71us/sample - loss: 0.2300 - accuracy: 0.9122
10000/1 - 0s - loss: 0.2230 - accuracy: 0.8835

Test accuracy: 0.8835

Tempo de treinamento: 43.01565933227539
```

Figura 4. Saída obtida com duas camadas ocultas adicionais

7. Verifique o desempenho* da RN realizando modificações no conjunto de testes e/ou conjunto de treinamento, testando entradas diferentes, saídas esperadas, etc..

Foi alterado o conjunto de treinamento inteiro, gerando um novo conjunto no mesmo formato porém completamente aleatório.

```
19
20    train_images = np.random.randint(0, 255, (60000, 28, 28))
21
```

Figura 5. Comando que gera um novo conjunto de treino

Com o treinamento completamente aleatório, o desempenho da rede foi péssimo, com acurácia próxima a 10% tanto para o treinamento quanto para o teste, porém o tempo de treinamento permaneceu por volta de 30s.

```
Epoch 9/10
60000/60000 [==============================] - 3s 50us/sample - loss: 2.3028 - accuracy: 0.0993
Epoch 10/10
60000/60000 [==============================] - 3s 50us/sample - loss: 2.3027 - accuracy: 0.0998
10000/1 - 0s - loss: 2.3089 - accuracy: 0.1386

Test accuracy: 0.1386

Tempo de treinamento: 29.257147789001465
```

Figura 6. Saída obtida com conjunto de treinamento completamente aleatório

Referências

Tutorial utilizado: https://www.tensorflow.org/tutorials/keras/basic_classification (classificação de artigos de moda (modificação no conjunto MNIST))

GitHub deste trabalho: https://github.com/led479/AprendendoTensorFlow

Github do Fashion MNIST: https://github.com/zalandoresearch/fashion-mnist

Instalar TensorFlow: https://www.tensorflow.org/install/pip?lang=python3