Atividade 5: Redes Neurais

Redes Neurais é uma técnica amplamente utilizada para diferentes aplicações e classes de problemas, que algoritmos especialistas não conseguem atender. Este trabalho apresenta um implementação do método em Python, para a disciplina de tópicos especiais em aprendizagem.

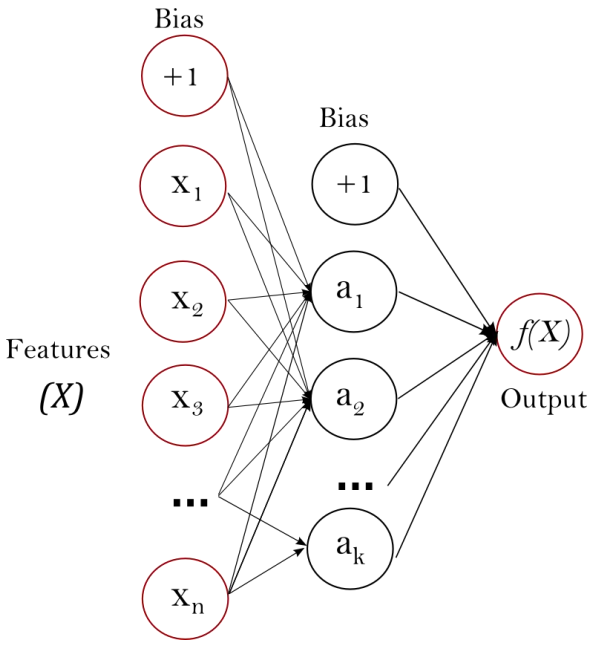
Introdução

Este artigo mostra como a aplicação que implementa RNA foi codificada. Para isto, uma breve explicação é feita sobre cada método desenvolvido, que compõe o exercício. O objetivo do trabalho é implementar um modelo simples que atenda os requisitos da disciplina.

Métodos

Rede Neurais Artificiais é um método de computação inspirados nas redes neurais biológicas que constituem cérebros de animais. Tais sistemas aprendem (melhorando progressivamente o desempenho) tarefas considerando exemplos, geralmente sem programação específica de tarefas. É um algoritmo de aprendizagem supervisionado que aprende uma função F por treinamento em um conjunto de dados, onde m é o número de dimensões para entrada e o é o número de dimensões para saída. Dado um conjunto de características X = {x_1, x_2, ..., x_m} e um alvo y, pode aprender um aproximador de função não linear para classificação ou regressão.

É diferente da regressão logística, naquela entre a camada de entrada e a camada de saída, pode haver uma ou mais camadas não-lineares, chamadas camadas ocultas. A Figura 1 mostra uma camada oculta MLP com saída escalar.



A camada mais à esquerda, conhecida como camada de entrada, consiste em um conjunto de neurônios representando os recursos de entrada. Cada neurônio na camada oculta transforma os valores da camada anterior com uma soma linear ponderada, seguida de uma função de ativação não linear, como o hiperbólico. A camada de saída

recebe os valores da última camada oculta e transforma-os nos valores mais a direita (output).
Suponha que existam n amostras de treinamento, características m, k camadas ocultas, cada uma contendo neurônios h - para simplicidade e o neurônio de saída. A complexidade do tempo de backpropagation é vista a seguir, onde i é o número de iterações. Uma vez que a backpropagation tem uma complexidade de tempo elevada, é aconselhável começar com um número menor de neurônios escondidos e algumas camadas ocultas para treinar.

$$O(n \cdot m \cdot h^k \cdot o \cdot i)$$

As principais vantagens de Multi-layer Perceptron são:

- Capacidade de aprender modelos não-lineares.
- Capacidade de aprender modelos em tempo real (aprendizagem on-line).

Desenvolvimento

Os métodos foram implementados na linguagem de programação pyhton (como permitido para este exercício). Diferente métodos foram criados que em conjunto implementam um método para classificação de dois digitos manuscritos.

BackPropagation

```
/def backPropagation(train, teste, txTraining, nEp,
hideNumber):
    predict = list()
    num_input = len(train[0]) - 1
    nSds = len(set([row[-1] for row in train]))

    net = startNet(num_input, hideNumber,
nSds)
    trainner(net, train, txTraining, nEp, nSds)
    for row in teste:
        pred = predicao(net, row)
        predict.append(pred)
    return predict
```

Erro BackPropagation

```
fdef bpError(net, nxt):
    for i in reversed(range(len(net))):
        camada = net[i]
        errs = list()
```

```
            if i != len(net)-1:
                for j in range(len(camada)):
                    err = 0.0
                    for n in net[i + 1]:
                        err += (n['p'][j] *
n['delta'])
                    errs.append(err)
            else:
                for j in range(len(camada)):
                    n = camada[j]
                    errs.append(nxt[j] - n['sd'])

for j in range(len(camada)):
                n = camada[j]
                n['delta'] = errs[j] *
transferD(n['sd'])
```

FowardPropagate

```
def fProp(net, row):
    entrada = row
    for camada in net:
        nova_entrada = []
        for n in camada:
            a = activated(n['p'], entrada)
            n['sd'] = transfer(a)
            nova_entrada.append(n['sd'])
        entrada = nova_entrada
    return entrada
```

Resultados

Abaixo estão representados de forma matricial os resultados obtidos a partir da base de imagens americana de dígitos manuscritos.

As imagens estão normalizadas, binalizadas, com visto na amostra a seguir:

A classificação foi realizada sobre os dígitos 1 e 3, de modo a verificar a acertividade do algoritmo para números

visualmente distintos. Os resultados obtidos foram agrupados na tabela a seguir:

Testes	**1**
Taxa Aprendizado	0.2
N de Épocas	10
N de camadas escondidas	15
Acertividade (%)	98.23

Testes	**2**
Taxa Aprendizado	0.5
N de Épocas	5
N de camadas escondidas	20
Acertividade (%)	97.49

Testes	**3**
Taxa Aprendizado	0.8
N de Épocas	3
N de camadas escondidas	7
Acertividade (%)	96.81

Conclusão

A partir do algoritmo de Redes Neurais Artificiais com BackPropagation implementado foi possível executar os exemplos indicados com sucesso, absorvendo os conceitos teóricos e as aplicações. Seus resultados se mostraram com alta acertividade, podendo-se notar que ao aumentar o número de camadas escondidas, com baixa taxa de aprendizado, os resultados tendem a ficar mais precisos.

Em contra-partida ao aumentar o número de camadas escondidas e o número de épocas o tempo de treinamento aumenta de forma drástica, que deve ser levada em consideração quando aplicada em diferentes problemas.

Referências

1. Scikit. 1.7, RedesNeurais - Disponível em:http://scikit-learn.org/stable/modules/neural_networks_supervised.html: 11 de Nov. de 2017

2. Artificial Neural Networks - Disponível em (https://en.wikipedia.org/wiki/Artificial_neural_network). Acesso em: 11 de Nov. de 2017