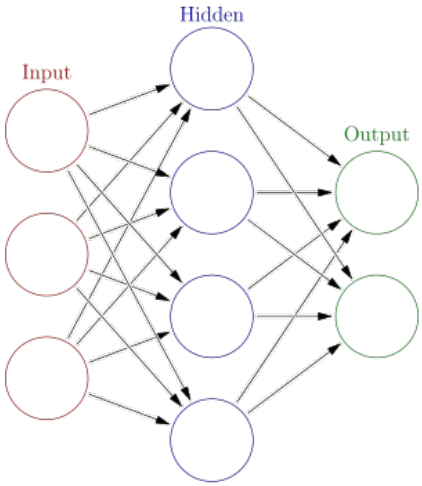
Desafio de Programação Paralela - ERAD-SP 2016

Aprendizado de Redes Neurais Multicamadas

Introdução



Redes neurais artificiais são compostas por um conjunto de neurônios simplificados, que recebem um conjunto de entradas de outros neurônios e geram uma saída. Uma rede neural é chamada de *feed-forward* quando é dividida em camadas e possui apenas conexões unidirecionais de uma camada para a seguinte, como na figura ao lado (fonte: en.wikipedia.org).

Redes como esta podem ser capazes de aprender mapeamentos arbitrários do tipo `saída = f(entrada)`, onde saída é um vetor contendo a saída de cada um dos neurônios da rede e `entrada` é um vetor com a entrada da rede. Mas para tal é necessário definir os pesos das conexões entre os neurônios, sendo que existem diversos algoritmos de aprendizado com esta finalidade.

Algoritmo de Retro-propagação (backpropagation)

O algoritmo de retro-progagação (*backpropagation*) é um método de aprendizado supervisionado, isto é, são fornecidos pares (`entrada,saída`) para a rede, que deve aprender a realizar o mapeamento. O algoritmo determina o erro na camada de saída da rede e propaga este erro para as camadas anteriores, de modo que estas possam ajustar os seus pesos de modo a diminuir o erro da rede.

Existem diversas fontes de explicações detalhadas sobre o algoritmo, que podem ser consultadas:
https://page.mi.fu-berlin.de/rojas/neural/chapter/K7.pdf
https://en.wikipedia.org/wiki/Backpropagation
http://neuralnetworksanddeeplearning.com/chap2.html

Note que o aprendizado pode ser feito de modo *online* ou *offline*. No aprendizado *online*, os pesos são modificados na rede imediatamente após a apresentação de cada exemplo de treinamento. Já no *offline*, primeiro são calculadas as alterações dos pesos para todos os exemplos de entrada para, só então, os pesos na rede serem atualizados. Na implementação *híbrida* do algoritmo, são calculadas as alteração para `k` entradas e, então, estas alterações são colocadas na rede. O processo é repetido até que a rede atinja um erro mínimo de classificação dentro do conjunto de aprendizado. Isto permite que o aprendizado seja paralelizado entre múltiplos núcleos.

Código Serial Fornecido

É fornecido uma versão serial do programa que deve ***obrigatoriamente*** ser usada como programa de referência pelos competidores.. A implementação está dividida em três arquivos fontes (e seus cabeçalhos), que devem ser compilados em conjunto:

`NeuralNet.cpp`: contém o método principal de execução. O programa lê as configurações da rede e os conjuntos de treinamento e testes dos arquivos de entrada, cria a rede neural e executa as

etapas de treinamento e teste da rede.

`backprop.cpp`: implementação da rede neural. A principal função desta classe é `backpropagation()` que realiza o treinamento da rede. É esta a função que deve ser paralelizada no código.

`database.cpp`: realiza a leitura dos arquivos de entrada e carrega em matrizes bidimensionais. A matriz de treinamento é a variável matrix[][] e a matriz de testes é a variável teste[][].

Entrada

O programa recebe um arquivo de entrada através da entrada padrão:

```
./erad < Input
```

O conjunto de entrada são caracteres descritos por 16 atributos. Os dados foram retirados do site as http://archive.ics.uci.edu/ml/datasets/Letter+Recognition, que descreve também como os atributos foram definidos. A rede deve ser capaz de aprender a distinguir 2 caracteres a partir de seus atributos. No exemplo abaixo, vemos conjuntos de testes e treinamento para diferenciação das letras I e R.

```
I,5,12,3,7,2,10,5,5,4,13,3,9,2,8,4,10
...
R,2,3,3,2,2,7,7,5,5,7,5,6,2,7,4,8
I,5,12,3,7,2,10,5,5,4,13,3,9,2,8,4,10
...
R,2,3,3,2,2,7,7,5,5,7,5,6,2,7,4,8
```

O arquivo Input deve conter o seguinte conteúdo em ordem:
`M` número de amostras de treinamento
`N` número de amostras para Testes
`K` número de atributos de cada caracter
`M` entradas de treinamento no formato "`A a b c ... n`" → onde o caractere indica a classe da letra e os seguintes `K` valores os atributos.
`N` entradas para teste no formato "`A a b c ... n`"
`4 16 10 10 1` → onde o primeiro valor indica o número de camadas da rede e, os seguintes, o número de neurônios em cada camada.

O tempo de execução é dependente da configuração da rede.

Tarefa

O objetivo das equipes deve ser realizar a paralelização da etapa de aprendizado da rede neural. Para tal, a versão submetida deve conter alterações ***exclusivamente***: (1) no código de `backpropag.cpp` e (2) na seção demarcada do método `main()` do arquivo `NeuralNet.cpp`. Para alterações em outras partes do código é necessário consultar o comitê julgador, justificando a necessidade. Submissões que não se atendam esta restrição serão desclassificadas.

Note que a paralelização não pode alterar o comportamento do algoritmo, caso contrário ele dará um resultado diferente do código sequencial. Por exemplo, alterar o valor de `stepSize` altera o comportamento do algoritmo, assim como alterar o ponto onde o teste do `mse`, indicado no código como `if( bp->mse(database->targetSample) < Thresh)`, é realizado. Sempre confira se a versão paralela produz o mesmo resultado da sequencial.

Avaliação

Serão fornecidos dados na entrada padrão do programa, conforme descrito anteriormente. O tamanho e composição dos conjuntos de treinamento e teste podem variar, assim como o tamanho das redes. Estas poderão ter entre 3 e 5 camadas e cada camada poderá conter entre 5 e 20 neurônios (exceto pela primeira, que tem sempre 16 neurônios e a última, que tem sempre 1 neurônio).

Os resultados do programa devem ser escritos na saída padrão e deverão indicar quais as letras presentes na seção de testes da entrada padrão. O programa deverá acertar a saída em cada caso. Será também verificado se durante o aprendizado a rede realmente atingiu o erro mínimo `mse`, definido como critério de parada.

Testes na plataforma

Será fornecida uma plataforma para realização de testes, similar à que será usada para avaliação. Mais informações serão publicadas em breve.