iMobilis
Computação Móvel

SEVA

Self Organizing Maps

André Siqueira Ruela

UFOP
Universidade Federal de Ouro Preto

decom
departamento de computação

Sumário

- Introdução
- Estrutura
- Reconhecimento
- Normalização
- Cálculo da Saída
- Aprendizado
- Implementação

UFOP
Universidade Federal de Ouro Preto

SEVA iMobilis

decom
departamento de computação

Introdução

- *Self Organizing Maps* (SOM) são também conhecidos como redes Kohonen, devido ao seu criador Tuevo Kohonen.

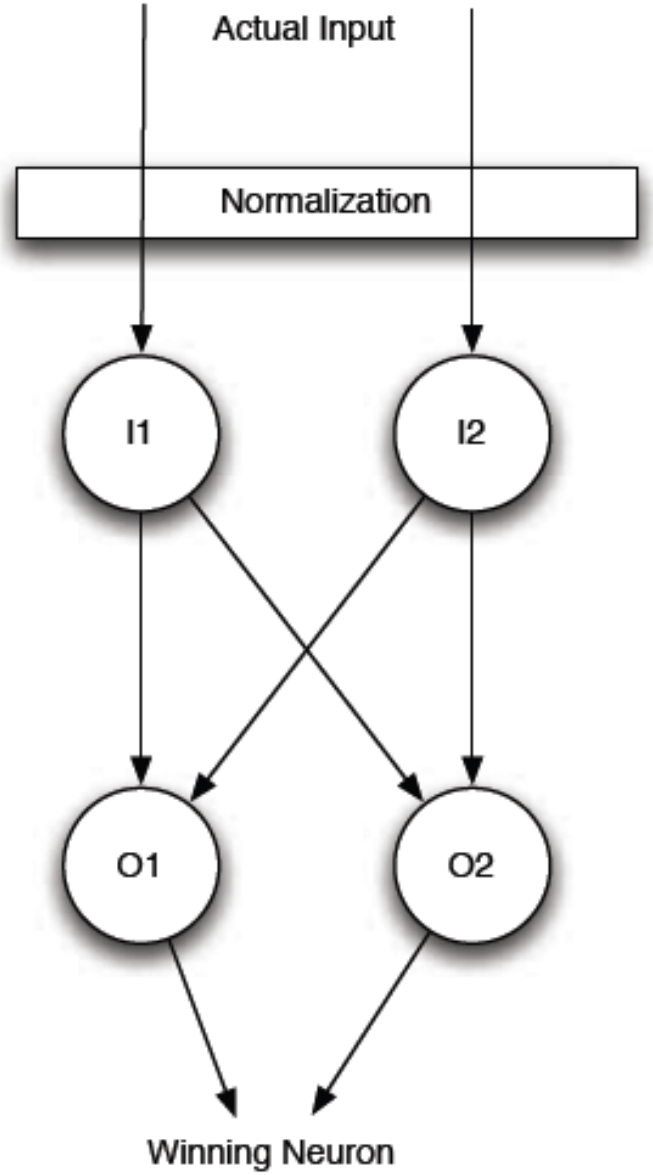
- Realiza um treinamento não supervisionado, geralmente designada a classificar padrões em grupos distintos.

- Difere de redes feedforward tanto em estrutura, quanto em forma de treinamento.

Introdução

- Os neurônios não possuem funções de ativação nem limiares.
- Na saída, apenas um neurônio é selecionado como vencedor.
- Em geral, cada neurônio na camada de saída representa um grupo distinto.
- Adicionalmente, SOM não possuem camada intermediária.

Estrutura



SEVA iMobilis

UFOP Universidade Federal de Ouro Preto

decom departamento de computação

Reconhecimento

- O reconhecimento de padrões em um SOM é bastante simples. Por exemplo:

Input Neuron 1 (I1)	0.5
Input Neuron 2 (I2)	0.75

I1 -> O1	0.1
I2 -> O1	0.2
I1 -> O2	0.3
I2 -> O2	0.4

- O primeiro passo é a normalização do vetor de entrada. Os valores devem estar entre -1 e 1.

Normalização

- Para a normalização do vetor de entrada, existem dois métodos comuns.

- Multiplicativo: método mais simples que consiste calcula o raiz quadrada do somatório dos quadrados das entradas.

$$f = \frac{1}{\sqrt{\sum_{i=0}^{n-1} x_i^2}}$$

```
1.0 / Math.sqrt( (0.5 * 0.5) + (0.75 * 0.75) )
```

  - Esta equação produz o fator de normalização 1,1094.

UFOP Universidade Federal de Ouro Preto

SEVA iMobilis

decom departamento de computação

Normalização

- Normalização Z-Axis: os dados de entrada são multiplicados por uma constante.

$$f = \frac{1}{\sqrt{n}}$$

  - A equação depende do tamanho *n* da entrada e preserva a magnitude da informação.

$$s = f\sqrt{n - l^2}$$

Entrada sintética. Onde *f* é o fator calculado acima, *n* o tamanho da entrada e *l* é o comprimento do vetor.

UFOP Universidade Federal de Ouro Preto

SEVA iMobilis

decom departamento de computação

Normalização

- Em geral, a normalização z-axis é melhor que a multiplicativa.
- Entretanto, quando os valores do vetor de entrada são muito próximos a zero, a z-axis não tem um bom desempenho.
- Nesse caso, é melhor recorrer a normalização multiplicativa.

Cálculo da Saída

- Para calcular a saída, é preciso multiplicar as entrada pelos pesos e calcular a saída em cada neurônio de saída.

$$[0.5 \quad 0.75] * [0.1 \quad 0.2] = (0.5 * 0.75) + (0.1 * 0.2) = 0.395$$

- Multiplicando o resultado pelo fator de normalização 1,1094, temos a saída 0,438213.

- O próximo passo é mapear a saída em um valor bipolar.

Cálculo da Saída

- Uma vez que a entrada foi normalizada, a saída precisa ser normalizada de maneira semelhante.
- Em um sistema bipolar, o binário zero é mapeado em -1 e o binário 1 permanece 1.
- Esse mapeamento consiste em multiplicar por 2 e subtrair 1.
- Para a saída 0,438213 o resultado é -0,123574.

Cálculo da Saída

- O valor -0,123574 é a saída de apenas o primeiro neurônio.
- Este valor deve ser comparado a todas as demais saídas para a escolha do neurônio vencedor.
- Neste exemplo, a segunda saída consiste em 0,0465948 no qual o mapeamento bipolar produz o valor -0.9068104.
- O neurônio vencedor é aquele que produz o maior resultado.

Aprendizado

- Para cada conjunto de entrada, tem-se um neurônio vitorioso.
- O neurônio vitorioso tem os seus pesos ajustados e dessa forma ele reagirá mais forte para uma entrada na próxima vez que ela for apresentada.
- Enquanto neurônios diferentes vencem para diferentes padrões, a sua habilidade de reconhecer tais padrões aumenta.

Aprendizado

- O treinamento se repete até que o erro tenha sido reduzido a uma pequena quantidade, ou até que o erro tenha estagnado em um valor indesejado.

- Assim como em redes feedforward, uma constante denominada taxa de aprendizado precisa ser configurada.

- Os pesos dos neurônios são ajustados a cada época. Se o ajuste dos pesos não produzir bons resultados, a rede é reiniciada e um novo ciclo de treinamento é iniciado.

Aprendizado

- O ajuste dos pesos pode ser realizado de duas diferentes formas:

- Método aditivo: $$w^{t+1} = \frac{w^t + \alpha x}{length\,(w^t + \alpha x)}$$
  - x é o conjunto de entrada, w são os pesos do neurônio vencedor, α é a taxa de aprendizado e o resultado é o valor do novo peso.

$$e = x - w^t$$

- Método subtrativo:

$$w^{t+1} = w^t + \alpha e$$

  - Considera a diferença entre a entrada x e os pesos w do neurônio vencedor.

Cálculo do Erro

- Na realidade, o erro calculado em um treinamento não supervisionado não é um erro no sentido normal da palavra.
- O erro não é utilizado na correção dos pesos, como é no caso do backpropagation.
- O erro apenas indica o quão bem está a rede na tarefa de classificar padrões em grupos distintos.
- Não existe uma forma oficial de se calcular o erro em uma rede de Kohonen.

Implementação

- A implementação de uma aplicação básica de uma rede de Kohonen é feita em três classes:

Class	Purpose
NormalizeInput	Normalizes the input for the self-organizing map. This class implements the normalization method discussed earlier in this chapter.
SelfOrganizingMap	This is the main class that implements the self-organizing map.
TrainSelfOrganizingMap	Used to train the self-organizing map.

Implementação

- NormalizeInput:
  - calculateFactors – método que calcula ambos métodos de normalização (multiplicativo e z-axis) e a entrada sintética.
  - createInputMatrix – método que recebe o vetor de entrada e cria uma matriz de entrada.
- SelfOrganizingMap:
  - winner – calcula a saída de cada neurônio e retorna o índice do neurônio vencedor.

Implementação

- TrainSelfOrganizingMap:
  - iteration – método chamado a cada iteração, geralmente quando o erro ainda não se reduziu a um nível satisfatório.
  - evaluateErrors – método chamado para avaliar o erro atual da rede, o qual será comparado com o melhor erro já encontrado.
  - forceWin – força a vitória de algum neurônio, caso nenhum tenha disparado.
  - adjustWeights – corrige os pesos da matriz de acordo com o método adotado.

ORC – Optical Character Recognition

- Programas ORC são capazes de ler textos. Vamos apresentar uma SOM capaz de reconhecer caracteres escritos manualmente.
- A cada época, uma rede SOM é criada.
- Esta rede possui 64 neurônios na camada de entrada e *n* neurônios de saída, onde *n* é o número de caracteres que se deseja reconhecer.

Aplicações Industriais

- O exemplo apresentado não consiste em uma aplicação industrial prática.
- Possui diversas limitações, por exemplo, quanto a diferenciação dos caracteres ‘o’, ‘O’ e ‘0’.
- Aplicações avançadas para a leitura completa de textos devem no mínimo realizar: detecção de linhas, detecção de espaços, letras maiúsculas e números, elementos contextuais e verificação da ortografia, por exemplo.