Redes Neurais Artificiais

AULA 11 – Redes LVQ e Counter-Propagation
– Arquitetura e Aspectos de Treinamento –

Prof. Ivan Nunes da Silva

1. Introdução

Aspectos introdutórios da arquitetura

- A rede *LVQ* (*Learning Vector Quantization* // Quantização Vetorial por Aprendizagem) é também aplicada tipicamente em problemas de classificação de padrões.
- O aprendizado da *LVQ* é processado de maneira supervisionada.
- A rede *LVQ* possui também certa similaridade em seu processo de aprendizado quando comparada com a rede auto-organizável de Kohonen.
- O treinamento desta rede neural, composta em sua forma convencional por uma única camada de neurônios, realiza um processo de Quantização Vetorial, frente ao espaço em que as amostras se encontram, a fim de ponderar as regiões de domínio de cada uma das classes.

2. Processo de Quantização Vetorial

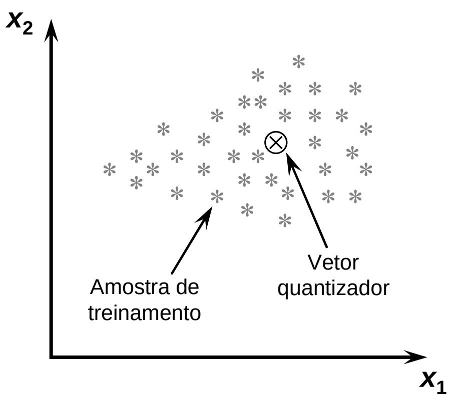
Princípios básicos e objetivos

- Considera-se aqui um problema de classificação de padrões, cujas amostras são divididas em ***n* classes**, conhecidas a priori (aprendizado supervisionado).
- O processo de quantização vetorial consiste em atribuir a cada uma dessas classes um único vetor, denominado de quantizador (vetor referência).
- O quantizador representa então o perfil que permeia o respectivo grupo frente às operações de classificação de uma nova amostra.
- Assim sendo, uma nova amostra será classificada como pertencente àquela classe em que o vetor quantizador esteja mais próximo.
- Neste caso, levando em conta uma classe específica, o objetivo da quantização vetorial seria a obtenção de um quantizador, que esteja alocado numa posição estratégica, cuja soma de sua distância em relação a todas as amostras que compõem a classe seja a menor possível.

2. Processo de Quantização Vetorial

Ilustração do processo (para uma classe)

- A figura seguinte ilustra a idéia envolvida com o processo de quantização vetorial frente a um conjunto de amostras constituídas de duas componentes {x_1 e x_2}.

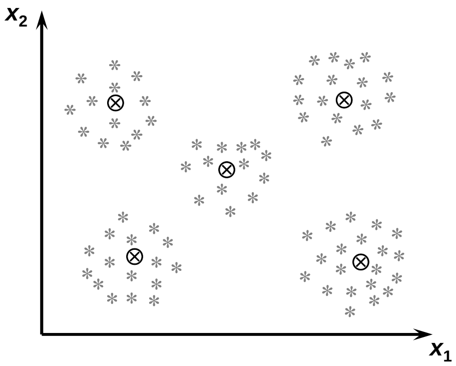


- Observa-se aqui que o vetor quantizador ficou posicionado numa localização em que o somatório de sua distância aos elementos da classe é o mínimo possível.
- Verifica-se ainda que o quantizador está alocado mais proximamente ao local onde há o maior adensamento de amostras.
- Por outro lado, se as amostras da classe tiverem uma distribuição normal, o vetor quantizador tenderia então à média de seus valores.

2. Processo de Quantização Vetorial

Ilustração do processo (para várias classes)

- Considerando agora a existência de um total de *n* classes, haverá então *n* vetores quantizadores que serão responsáveis por representar os perfis discriminantes de cada uma delas.
- A figura seguinte ilustra 5 classes distintas com seus respectivos vetores quantizadores já previamente calculados.



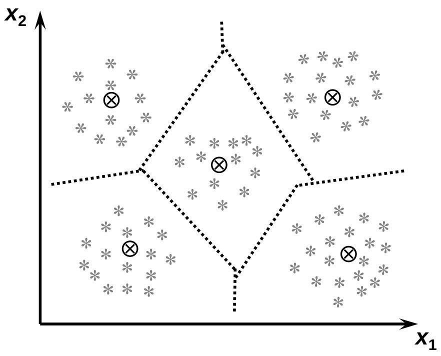
- Cada um dos 5 vetores quantizadores estará aqui representando o perfil que domina cada uma das classes.
- Considera-se agora a situação de que uma nova amostra tenha que ser classificada dentre uma dessas 5 classes possíveis.
- Então, o único parâmetro a ser usado na tarefa será o próprio vetor quantizador daquela classe.
- A principal medida utilizada para tal propósito é o nível de proximidade.
- De fato, quão mais próximo esteja um vetor quantizador de determinada amostra, mais parecido estará com a mesma.

5

2. Processo de Quantização Vetorial

Diagramas de Voronoy

- Com a adoção do critério de distância mínima para definir o vetor quantizador dominante à determinada amostra, tornar-se-á então possível delimitar as suas fronteiras de dominação frente aos demais quantizadores.
- Assim, dado que o ponto médio entre dois quaisquer vetores quantizadores seja um divisor natural de proximidade, conclui-se então que bastaria obter o hiperplano, perpendicular ao segmento de reta que liga tais vetores, impondo-se como condição a sua passagem pelo respectivo ponto médio.



- Cada um dos 5 vetores quantizadores estará agora representando o perfil que domina cada uma das classes.
- As fronteiras de dominância, quando se assume o critério de proximidade, possibilitam decompor todo o espaço amostral métrico em sub-regiões que delimitam as referidas classes.
- O conjunto total destes retículos derivados de tal particionamento, e que permite também a redução da dimensionabilidade do problema, é conhecido como diagrama de Voronoy ou mosaico de Dirichlet.

6

2. Processo de Quantização Vetorial
Aspectos práticos

- Em termos práticos, após a alocação dos vetores quantizadores, a definição da classe a que pertence uma nova amostra é feita a partir do cálculo de sua distância frente a todos vetores quantizadores.
- A amostra será então classificada como pertencente àquela classe que estará sendo inteiramente representada pelo vetor quantizador com menor distância.
- Então, a principal tarefa da quantização vetorial está em determinar a posição espacial de cada quantizador dentro do conjunto amostral de cada classe, sendo também este o alvo das redes *LVQ*.
- Em termos comparativos, a quantização vetorial é bem diferente da clusterização, em que cujo propósito está tão somente em encontrar e agrupar amostras em classes que possuem qualidades em comum.
- Em contraste, além de encontrar grupos com características similares, a quantização vetorial consiste de converter todo o espaço de entradas, em que as amostras estão definidas, para um espaço discreto de saída, com considerável redução de dimensionabilidade.

7

3. Redes LVQ
Arquitetura, treinamento, aplicabilidade

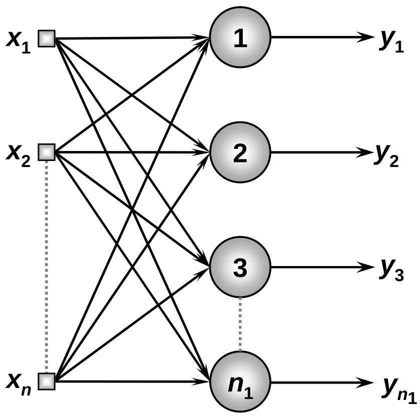
- A rede LVQ foi também idealizada por Kohonen, sendo ainda considerada uma versão supervisionada dos mapas auto-organizáveis (Self-Organization Maps // SOM).
- Assim, diferentemente da topologia SOM, exige-se então, para propósitos de treinamento das redes LVQ, de um conjunto de pares entradas e saídas representativos do processo a ser mapeado.
- O treinamento de redes LVQ é também executado de maneira competitiva, similarmente àquele utilizados para as redes SOM, de modo que os vetores de pesos dos neurônios estarão representando os respectivos vetores quantizadores de classes.
- Assim, para utilização desta topologia, as diversas classes associadas à representação do processo devem ser conhecidas.
- A LVQ é aplicada tipicamente em problemas de classificação de padrões.

8

3. Redes LVQ

Ilustração topológica da LVQ

- A figura mostra uma rede LVQ composta de n entradas e n_1 neurônios, os quais estarão representando todas as classes envolvidas com o referido problema de classificação de padrões.



- Em contraste à arquitetura de Kohonen, a rede LVQ não possui conexões laterais entre neurônios.
- Assim, aqueles neurônios vizinhos ao vencedor deixarão também de ter os seus pesos ajustados.
- De fato, a existência de conexões laterais é desnecessária, pois os estímulos excitatório e inibitório serão computados pelo algoritmo de treinamento.
- Os algoritmos de treinamento mais utilizados são denominados de LVQ-1 e LVQ-2.

9

4. Treinamento LVQ-1

Aspectos de aprendizado

- Os passos envolvidos com o algoritmo de treinamento LVQ-1 são implementados de maneira a ajustar somente os pesos sinápticos dos neurônios vencedores, frente aos processos de competição que envolve cada uma das amostras disponíveis.
- Neste caso, considera-se que cada vetor de entrada $\{\boldsymbol{x}^{(k)}\}$, utilizado durante o treinamento da LVQ, pertence somente a uma das classes j previamente conhecida, pois o mecanismo de aprendizado é do tipo supervisionado.
- Os procedimentos inerentes ao algoritmo de aprendizagem são, portanto, idênticos àqueles da rede de Kohonen, modificando apenas o critério de seleção dos neurônios que terão os seus pesos ajustados:
  - No algoritmo de treinamento LVQ-1 somente os pesos do neurônio vencedor serão devidamente sintonizados.
- Os dois passos principais do algoritmo consistem da obtenção do neurônio vencedor, assim como da especificação da regra de ajuste de seus respectivos pesos sinápticos.

10

4. Treinamento LVQ-1

Passos do algoritmo – Declaração do vencedor

- Em relação à obtenção do <u>vencedor</u>, aquele neurônio cujo vetor de pesos $\{\boldsymbol{w}^{(j)}\}$ tiver o maior nível de proximidade com uma determinada amostra $\{\mathbf{x}^{(k)}\}$, será declarado vitorioso.
- Similarmente à rede de Kohonen, uma das medidas de proximidade mais utilizadas é a <u>norma euclidiana</u> entre esses dois parâmetros, ou seja:

$$dist_j^{(k)} = \sqrt{\sum_{i=1}^{n} \left(x_i^{(k)} - w_i^{(j)}\right)^2}, \quad \text{com } j = 1, \ldots, n_1$$

onde $dist_j^{(k)}$ fornece a distância entre o vetor de entrada representando a k-ésima amostra $\{\boldsymbol{x}^{(k)}\}$ em relação ao vetor de pesos do j-ésimo neurônio $\{\boldsymbol{w}^{(j)}\}$.

- Após a declaração do neurônio vencedor, aplica-se então a <u>regra de ajuste</u> de seus pesos.

11

4. Treinamento LVQ-1

Passos do algoritmo – Regra de ajuste dos pesos (I)

- Após a declaração do neurônio vencedor, aplica-se então a <u>regra de ajuste</u> de seus pesos.
- Para a situação do neurônio vencedor $\{\boldsymbol{w}^{(j)}\}$ <u>estiver representando a própria classe</u> atribuída à respectiva amostra $\{\boldsymbol{x}^{(k)}\}$, isto é, $\boldsymbol{x}^{(k)} \in C^{(j)}$, ajustam-se então os seus pesos com a finalidade de aproximá-lo ainda mais daquela amostra.
  - Nesta circunstância, o mesmo terá como recompensa uma grande chance de vencer a competição quando a referida amostra for novamente apresentada no decorrer do treinamento.
- Caso contrário, para a condição do neurônio vencedor $\{\boldsymbol{w}^{(j)}\}$ <u>não estiver representando a classe</u> da referida amostra $\{\boldsymbol{x}^{(k)}\}$, ou seja, $\boldsymbol{x}^{(k)} \notin C^{(j)}$, então os seus pesos serão ajustados com o intuito de se afastar da mesma.
  - Neste caso, por conseqüência, há a chance de que outro neurônio possa vencer a competição na próxima época de treinamento.

12

4. Treinamento LVQ-1

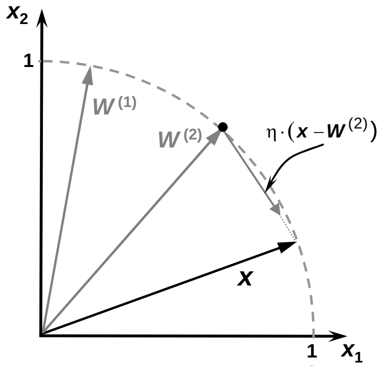
Passos do algoritmo – Regra de ajuste dos pesos (II)

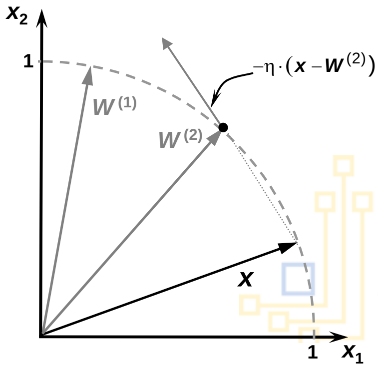
- Em termos algorítmicos, tais procedimentos podem ser sintetizados por meio da seguinte regra:

Se $\boldsymbol{x}^{(k)} \in C^{(j)}$

Então : $\boldsymbol{w}^{(j)} \leftarrow \boldsymbol{w}^{(j)} + \eta \cdot (\boldsymbol{x}^{(k)} - \boldsymbol{w}^{(j)})$

Senão : $\boldsymbol{w}^{(j)} \leftarrow \boldsymbol{w}^{(j)} - \eta \cdot (\boldsymbol{x}^{(k)} - \boldsymbol{w}^{(j)})$





4. Treinamento LVQ-1

Algoritmo (Fase de treinamento)

Início {Algoritmo *LVQ*-1 // Fase de Treinamento}

<1> Obter o conjunto de amostras de treinamento { $\boldsymbol{x}^{(k)}$ };

<2> Associar cada amostra { $\boldsymbol{x}^{(k)}$ } com sua respectiva classe (saída) desejada { $C^{(j)}$ }, com j variando de um até o total de classes;

<3> Iniciar o vetor de pesos de cada neurônio { $\boldsymbol{w}^{(j)}$ } com um dos valores das amostras em { $C^{(j)}$ };

<4> Normalizar os vetores de amostras { $\boldsymbol{x}^{(k)}$ } e de pesos { $\boldsymbol{w}^{(j)}$ };

<5> Especificar a taxa de aprendizagem { η };

<6> Iniciar o contador de número de épocas { *época* $\leftarrow$ 0 };

<7> Repetir as instruções:

- <7.1> Para todas as amostras de treinamento { $\boldsymbol{x}^{(k)}$ }, fazer:
  - <7.1.1> Calcular as distâncias euclidianas entre $\boldsymbol{x}^{(k)}$ e $\boldsymbol{w}^{(j)}$, conforme a expressão (9.1);
  - <7.1.2> Declarar como vencedor aquele neurônio j que contenha a menor distância euclidiana:

    $$vencedor = \arg\min_{j} \{ \| \boldsymbol{x}^{(k)} - \boldsymbol{w}^{(j)} \| \};$$
  - <7.1.3> Ajustar o vetor de pesos do vencedor, conforme a regra explicitada na expressão (9.2);
  - <7.1.4> Normalizar o vetor de pesos que foi ajustado no passo anterior;
- <7.2> *época* $\leftarrow$ *época* + 1;

Até que: não haja mudanças significativas nos vetores de pesos.

Fim {Algoritmo *LVQ*-1 // Fase de Treinamento}

4. Treinamento LVQ-1

Algoritmo (Fase de operação)

- A fase de operação consiste em determinar tão somente o neurônio vencedor, cujo rótulo estará representando a classe a ser atribuída à respectiva amostra.

Início { Algoritmo *LVQ*-1 – Fase de Operação}

<1> Apresentar a amostra { $\boldsymbol{x}$ } a ser classificada e normalizar;

<2> Assumir os vetores de pesos { $\boldsymbol{w}^{(j)}$ } já ajustados durante a fase de treinamento;

<3> Executar as seguintes instruções:

<3.1> Calcular as distâncias euclidianas entre $\boldsymbol{x}$ e $\boldsymbol{w}^{(j)}$, conforme a expressão (9.1);

<3.2> Declarar como vencedor o neurônio *j* que contenha a menor distância euclidiana;

<3.3> Associar a amostra à classe em que o neurônio vencedor estiver representando;

<4> Disponibilizar a eventual classe em que a amostra foi associada.

Fim {Algoritmo *LVQ*-1 – Fase de Operação}

15

5. Treinamento LVQ-2

Passos do algoritmo – Regra de ajuste dos pesos

- Em relação ao algoritmo de treinamento LVQ-2, os procedimentos de ajuste são agora aplicados tanto para o neurônio vencedor $\boldsymbol{w}^{(j)}$ como para o seu vice $\boldsymbol{w}^{(m)}$.

Primeiro Caso ➔ Neurônio vencedor {$\boldsymbol{w}^{(j)}$} está representando a própria classe atribuída à respectiva amostra {$\boldsymbol{x}^{(k)}$}:

Se $(\boldsymbol{x}^{(k)} \in C^{(j)})$ *AND* $(\boldsymbol{x}^{(k)} \notin C^{(m)})$

$$\text{Então}: \begin{cases} \boldsymbol{w}^{(j)} \leftarrow \boldsymbol{w}^{(j)} + \eta \cdot (\boldsymbol{x}^{(k)} - \boldsymbol{w}^{(j)}) \\ \boldsymbol{w}^{(m)} \leftarrow \boldsymbol{w}^{(m)} - \eta \cdot (\boldsymbol{x}^{(k)} - \boldsymbol{w}^{(m)}) \end{cases}$$

Segundo Caso ➔ Neurônio vice-vencedor estiver representando a referida classe em que pertence a amostra {$\boldsymbol{x}^{(k)}$}:

Se $(\boldsymbol{x}^{(k)} \notin C^{(j)})$ *AND* $(\boldsymbol{x}^{(k)} \in C^{(m)})$

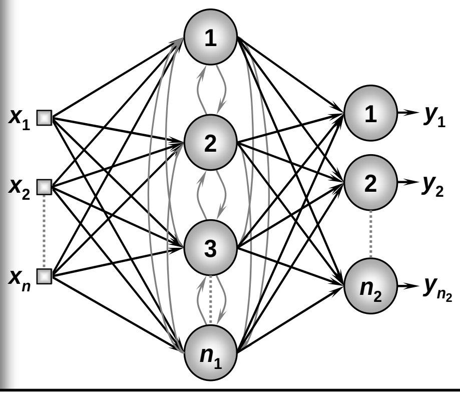
$$\text{Então}: \begin{cases} \boldsymbol{w}^{(j)} \leftarrow \boldsymbol{w}^{(j)} - \eta \cdot (\boldsymbol{x}^{(k)} - \boldsymbol{w}^{(j)}) \\ \boldsymbol{w}^{(m)} \leftarrow \boldsymbol{w}^{(m)} + \eta \cdot (\boldsymbol{x}^{(k)} - \boldsymbol{w}^{(m)}) \end{cases}$$

16

6. Rede *Counter-Propagation*

Arquitetura, treinamento, aplicabilidade

- As redes *counter-propagation* (contra-propagação) são normalmente constituídas de duas camadas neurais, cujo treinamento pode ser também considerado híbrido, tal como acontece com as redes *RBF*.
- Como o próprio sugere, a rede *counter-propagation*, diferentemente do PMC, faz nenhuma retropropagação de erro com o intuito de ajustar os pesos dos neurônios da camada intermediária.



- **1ª Camada** ➔ Constitui um arranjo de Kohonen:
  - Treinamento similar (aprendizado competitivo).
- **2ª Camada** ➔ Constitui um arranjo de Grossberg, denominado *Outstar*:
  - Função de ativação rampa e ausência de limiar.
  - Treinamento supervisionado, similar à regra de Hebb.

17

6. Rede *Counter-Propagation*

Aspectos de treinamento

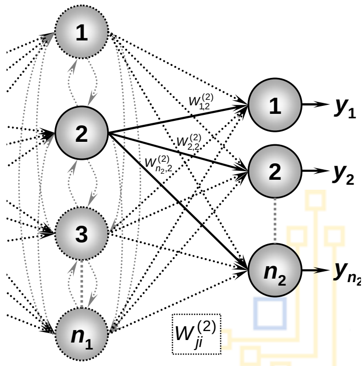
- Assim sendo, o treinamento da rede *counter-propagation* é realizado em dois estágios.
- Primeiramente, a camada escondida é ajustada a fim de realizar a identificação dos agrupamentos de dados.
- Em seguida, a camada de saída (*Outstar*) é então treinada com o intuito de associar as respostas da rede com as respectivas saídas desejadas.
- Para tanto, as entradas da camada de saída serão os valores produzidos pela própria camada escondida (Kohonen), em que haverá apenas um neurônio vencedor em relação a um estímulo de entrada.

18

6. Rede Counter-Propagation
Aspectos da camada Outstar

- O treinamento da camada ***Outstar*** da rede *counter-propagation* deve ser realizado somente após a finalização do aprendizado da camada de Kohonen.
- Serão ajustados somente os pesos daqueles neurônios cujas entradas (advindas da camada de Kononen) sejam diferentes de zero.
- Para fins de ilustração,assume-se que o neurônio 2 da camada de Kohonen seja o vencedor em relação a um estímulo de entrada {***x***}.
- Nesta circunstância, somente aqueles pesos associados à saída do vencedor (setas em linha contínua) serão ajustados em relação à camada ***Outstar***, isto é:

$$W_{ji}^{(2)} \leftarrow W_{ji}^{(2)} + \eta \cdot (d_j^{(k)} - W_{ji}^{(2)}) \cdot u_i^{(k)}$$



6. Rede Counter-Propagation
Algoritmo (Treinamento)

Início {Algoritmo *Counter-Propagation* – Fase de Treinamento}

<1> Obter o conjunto original de amostras de treinamento { $\boldsymbol{x}^{(k)}$ };
<2> Obter o vetor de saída desejada { $\boldsymbol{d}^{(k)}$ } para cada amostra;
<3> Executar o treinamento da camada intermediária (Kohonen) usando o algoritmo competitivo // {conforme Subseção 8.2};
<4> Iniciar $W_{ji}^{(2)}$ com valores aleatórios pequenos;
<5> Especificar taxa de aprendizagem { η };
<6> Iniciar o contador de número de épocas { *época* $\leftarrow$ 0 };
<7> Repetir as instruções:
<7.1> Para todas as amostras { $\boldsymbol{x}^{(k)}$ }, fazer:
<7.1.1> Obter os valores de saída do neurônio vencedor da camada intermediária (Kohonen) em relação a cada amostra $\boldsymbol{x}^{(k)}$ // {conforme Subseção 8.2};
<7.1.2> Ajustar somente os pesos individuais daqueles neurônios da camada de saída (*Outstar*) que estão conectados ao neurônio vencedor da camada intermediária (Kohonen) // {conforme expressão (9.6)};
<7.2> *época* $\leftarrow$ *época* + 1;
Até que: não haja mudanças significativas na matriz de pesos $W_{ji}^{(2)}$.

Fim {Algoritmo *Counter-Propagation* – Fase de Treinamento}

6. Rede Counter-Propagation

Algoritmo (Operação)

Início {Algoritmo *Counter-Propagation* – Fase de Operação}

<1> Apresentar uma amostra { $\boldsymbol{x}$ };

<2> Obter os valores de saída do neurônio vencedor { $u_i^{(v)}$ } da camada intermediária (Kohonen) // {conforme Subseção 8.2};

<3> Calcular a resposta da rede, advinda dos próprios neurônios da camada *Outstar*, por meio da seguinte expressão:

$$y_j = W_{ji}^{(2)} \cdot u_i^{(v)}, \text{ onde } j = 1, \ldots, n_2;$$

<4> Disponibilizar as saídas da rede dadas pelos elementos contido em y_j.

Fim {Algoritmo *Counter-Propagation* – Fase de Operação}

21

Fim da Apresentação

Núcleo de Pesquisa em
AUTOMAÇÃO INTELIGENTE
de Processos e Sistemas

22