Redes Perceptron - 3

10.1 Processo de Treinamento

Aspectos do Processo de Treinamento (Regra de Aprendizado de Hebb)

O ajuste dos pesos (wi) e do limiar (θ) de um Perceptron, com o objetivo de classificar padrões em duas classes possíveis, é realizado através da regra de aprendizado de Hebb.

- Se a saída produzida pelo Perceptron coincide com a saída desejada, os pesos sinápticos e o limiar são incrementados (ajuste excitatório) proporcionalmente aos valores de suas entradas.
- Se a saída produzida pelo Perceptron é diferente do valor desejado, os pesos sinápticos e o limiar serão decrementados (ajuste inibitório).
- Este processo é repetido sequencialmente para todas as amostras de treinamento, até que a saída produzida pelo Perceptron seja similar à saída desejada de cada amostra.

Em termos matemáticos, tem-se:

$$\begin{cases} w_i^{atual} = w_i^{anterior} + \eta \cdot (d^{(k)} - y) \cdot \boldsymbol{x}^{(k)} \\ \theta_i^{atual} = \theta_i^{anterior} + \eta \cdot (d^{(k)} - y) \cdot \boldsymbol{x}^{(k)} \end{cases}$$

- w_i são os pesos sinápticos.
- θ é limiar do neurônio.
- $\boldsymbol{x}^{(k)}$ é o vetor contendo a *k*-ésima amostra de treinamento
- $d^{(k)}$ é saída desejada para a *k*-ésima amostra de treinamento.
- y é a saída do ***Perceptron***.
- η é a taxa de aprendizagem da rede.

A taxa de aprendizagem {η} exprime o quão rápido o processo de treinamento da rede estará sendo conduzido rumo à sua convergência.

10.2 Processo de Treinamento

Aspectos de implementação computacional (adequações algorítmicas)

- Em termos de implementação, torna-se mais conveniente tratar as expressões anteriores em sua forma vetorial.
- Como a mesma regra de ajuste é aplicada tanto para os pesos w_i como para o limiar θ, pode-se então inserir θ dentro do vetor de pesos:

$$\boldsymbol{w} = [\theta \quad w_1 \quad w_2 \dots w_n]^T$$

- De fato, o valor do limiar é também uma variável a ser ajustada a fim de se realizar o treinamento do ***Perceptron***. Portanto, tem-se:

$$\begin{cases} w_i^{atual} = w_i^{anterior} + \eta \cdot (d^{(k)} - y) \cdot \boldsymbol{x}^{(k)} \\ \theta_i^{atual} = \theta_i^{anterior} + \eta \cdot (d^{(k)} - y) \cdot \boldsymbol{x}^{(k)} \end{cases}$$

notação vetorial →

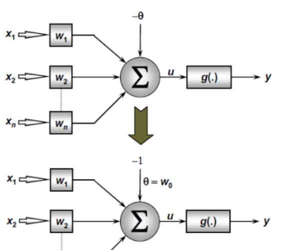
$$\boldsymbol{w}^{atual} = \boldsymbol{w}^{anterior} + \eta \cdot (d^{(k)} - y) \cdot \boldsymbol{x}^{(k)}$$

notação algorítmica ↓

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta \cdot (d^{(k)} - y) \cdot \boldsymbol{x}^{(k)}$$

Formato de cada amostra de treinamento:

$$\boldsymbol{x}^{(k)} = [-1 \quad x_1^{(k)} \quad x_2^{(k)} \dots x_n^{(k)}]^T$$



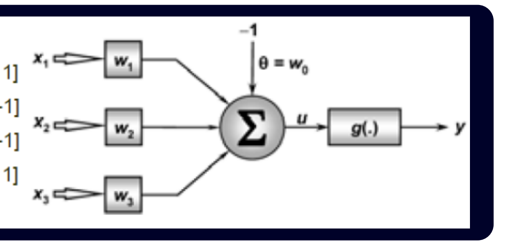
10.3 Processo de Treinamento

Aspectos de implementação computacional (montagem de conjuntos de treinamento)

Supõe-se que um problema a ser mapeado pelo Perceptron tenha três entradas {x1, x2, x3}, conforme a figura abaixo

Assume-se que se tem quatro amostras, constituída dos seguintes valores de entrada

Amostra 1 → Entrada: [0,1 0,4 0,7] → Saída desejada: [1]
Amostra 2 → Entrada: [0,3 0,7 0,2] → Saída desejada: [-1]
Amostra 3 → Entrada: [0,6 0,9 0,8] → Saída desejada: [-1]
Amostra 4 → Entrada: [0,5 0,7 0,1] → Saída desejada: [1]



10.4 Processo de Treinamento

Aspectos de implementação computacional (montagem de conjuntos de treinamento)

Então, pode-se converter tais sinais para que os mesmos possam ser usados no treinamento do ***Perceptron***:

Conjunto de treinamento

$\boldsymbol{x}^{(1)} = [-1 \quad 0,1 \quad 0,4 \quad 0,7]^T$ → com $d^{(1)} = 1$
$\boldsymbol{x}^{(2)} = [-1 \quad 0,3 \quad 0,7 \quad 0,2]^T$ → com $d^{(2)} = -1$
$\boldsymbol{x}^{(3)} = [-1 \quad 0,6 \quad 0,9 \quad 0,8]^T$ → com $d^{(3)} = -1$
$\boldsymbol{x}^{(4)} = [-1 \quad 0,5 \quad 0,7 \quad 0,1]^T$ → com $d^{(4)} = 1$

forma matricial →

$$\Omega^{(\boldsymbol{x})} = \begin{bmatrix} -1 & -1 & -1 & -1 \\ 0,1 & 0,3 & 0,6 & 0,5 \\ 0,4 & 0,7 & 0,9 & 0,7 \\ 0,7 & 0,2 & 0,8 & 0,1 \end{bmatrix}; \quad \text{(linhas } x_0, x_1, x_2, x_3\text{; colunas } \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \boldsymbol{x}^{(4)}\text{)}$$

$$\Omega^{(\boldsymbol{d})} = [1 \quad -1 \quad -1 \quad 1] \quad (\boldsymbol{d}^{(1)}, \boldsymbol{d}^{(2)}, \boldsymbol{d}^{(3)}, \boldsymbol{d}^{(4)})$$

- Geralmente, as amostras de treinamento são disponibilizadas em sua forma matricial (por meio de arquivo texto ou planilha).

10.5 Processo de Treinamento

Algoritmo de Aprendizagem (fase de treinamento)

Início {Algoritmo *Perceptron* – Fase de Treinamento}
<1> Obter o conjunto de amostras de treinamento $\{\boldsymbol{x}^{(k)}\}$;
<2> Associar a saída desejada $\{d^{(k)}\}$ para cada amostra obtida;
<3> Iniciar o vetor **w** com valores aleatórios pequenos;
<4> Especificar a taxa de aprendizagem {η};
<5> Iniciar o contador de número de épocas {*época* ← 0};
<6> Repetir as instruções:
<6.1> *erro* ← "inexiste";
<6.2> Para todas as amostras de treinamento $\{\boldsymbol{x}^{(k)}, d^{(k)}\}$, fazer:
<6.2.1> $u \leftarrow \boldsymbol{w}^T \cdot \boldsymbol{x}^{(k)} - (\theta)$;
<6.2.2> $y \leftarrow \text{sinal}(u)$;
<6.2.3> Se $y \neq d^{(k)}$
<6.2.3.1> Então $\begin{cases} \boldsymbol{w} \leftarrow \boldsymbol{w} + \eta \cdot (d^{(k)} - y) \cdot \boldsymbol{x}^{(k)} \\ erro \leftarrow \text{"existe"} \end{cases}$
<6.3> *época* ← *época* + 1;
Até que: *erro* ← "inexiste"
Fim {Algoritmo *Perceptron* – Fase de Treinamento}

Época de treinamento → É cada apresentação completa de todas as amostras pertencentes ao subconjunto de treinamento, visando, sobretudo, o ajuste dos pesos sinápticos e limiares de seus neurônios.

Pseudocódigo para fase de operação

Início {Algoritmo *Perceptron* – Fase de Operação}
<1> Obter uma amostra a ser classificada $\{\boldsymbol{x}\}$;
<2> Utilizar o vetor **w** ajustado durante o treinamento;
<3> Executar as seguintes instruções:
<3.1> $u \leftarrow \boldsymbol{w}^T \cdot \boldsymbol{x}$;
<3.2> $y \leftarrow \text{sinal}(u)$;
<3.3> Se $y = -1$
<3.3.1> Então: amostra $\boldsymbol{x} \in$ *{Classe A}*
<3.4> Se $y = 1$
<3.4.1> Então: amostra $\boldsymbol{x} \in$ *{Classe B}*
Fim {Algoritmo *Perceptron* – Fase de Operação}

Obs. 1: A "Fase de Operação" é usada somente após a fase de treinamento, pois aqui a rede já está apta para ser usada no processo.

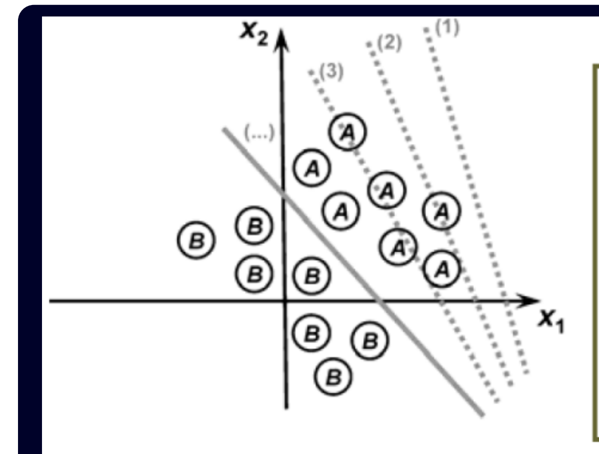
Obs. 2: A "Fase de Operação" é então utilizada para realizar a tarefa de classificação de padrões frente às novas amostras que serão apresentadas em suas entradas.

10.6 Processo de Treinamento

Ilustração do processo de convergência

Processo de treinamento tende a mover continuamente o hiperplano de classificação até que seja alcançada uma fronteira de separação que permite dividir as duas classes.

Como exemplo, seja uma rede composta de apenas duas entradas { x1 e x2 }



- Após a primeira época de treinamento (1), constata-se que o hiperplano está ainda bem longínquo da fronteira de separabilidade das classes.
- A distância tende ir cada vez mais decrescendo na medida em que se transcorre as épocas de treinamento.
- Quando o ***Perceptron*** já estiver convergido, isto estará então significando que tal fronteira foi finalmente alcançada.

10.7 Processo de Treinamento

Região de separabilidade (aspectos de convergência)

A reta de separabilidade a ser produzida após o treinamento do Perceptron não é única.

O número de épocas pode também variar de treinamento para treinamento.