Redes Neurais Artificiais

PARA ENGENHARIA E CIÊNCIAS APLICADAS

fundamentos teóricos e aspectos práticos

Ivan Nunes da Silva
Danilo Hernane Spatti
Rogério Andrade Flauzino

Artliber
EDITORA

Ivan Nunes da Silva
Danilo Hernane Spatti
Rogério Andrade Flauzino

Universidade de São Paulo

Redes Neurais Artificiais

para engenharia e ciências aplicadas

2ª edição
revisada e ampliada

Artliber
EDITORA

2ª edição - 2016

Revisão:
Maria Antonieta M. Eckersdorff

Capa e composição eletrônica:
Perfil Editorial

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro)

Silva, Ivan Nunes da
Redes neurais artificiais: para engenharia e ciências aplicadas / Ivan Nunes da Silva; Danilo Hernane Spatti; Rogério Andrade Flauzino. – São Paulo: Artliber, 2010.

1. Redes neurais 2. Inteligências artificiais 3. Arquitetura de redes neurais
I. Título II. Spatti, Danilo Hernane; Flauzino, Rogério Andrade

10-0725 CDD-004.032.62

Índices para catálogo sistemático:
1. Redes neurais: Engenharia e ciências aplicadas 004.032.62

2019

Artliber Editora Ltda.
Av. Diógenes Ribeiro de Lima, 3294
05083-010 – São Paulo – SP – Brasil
Tel.: (11) 3641-3893
info@artliber.com.br
www.artliber.com.br

Sumário

Capítulo 3

Rede *Perceptron*

3.1 – Introdução

O *Perceptron*, idealizado por Rosenblatt (1958), é a forma mais simples de configuração de uma rede neural artificial, cujo propósito focava em implementar um modelo computacional inspirado na retina, objetivando-se então um elemento de percepção eletrônica de sinais. Uma de suas aplicações consistia de identificar padrões geométricos.

A figura 3.1 mostra uma ilustração que exemplifica a concepção inicial do elemento *Perceptron*, em que os sinais elétricos advindos de fotocélulas mapeando padrões geométricos eram ponderados por resistores sintonizáveis, os quais podiam ser ajustados durante o processo de treinamento. Em seguida, um somador efetuava a composição de todos os sinais. Desta forma, o *Perceptron* poderia reconhecer diversos padrões geométricos, tais como letras e números.

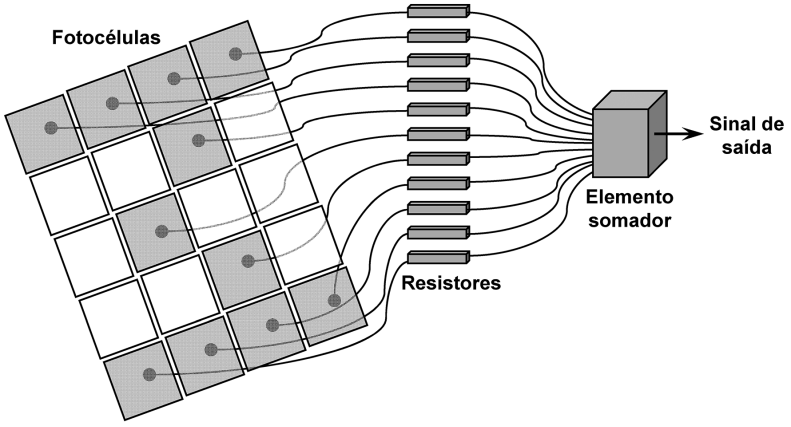

Figura 3.1 – Modelo ilustrativo do *Perceptron* para reconhecimento de padrões

A simplicidade da rede *Perceptron* está associada à sua condição de ser constituída de apenas uma camada neural, tendo-se também somente um neurônio artificial nesta única camada.

A figura 3.2 ilustra uma rede *Perceptron* constituída de *n* sinais de entrada, representativos do problema a ser mapeado, e somente uma saída, pois este tipo de rede é composto de um único neurônio.

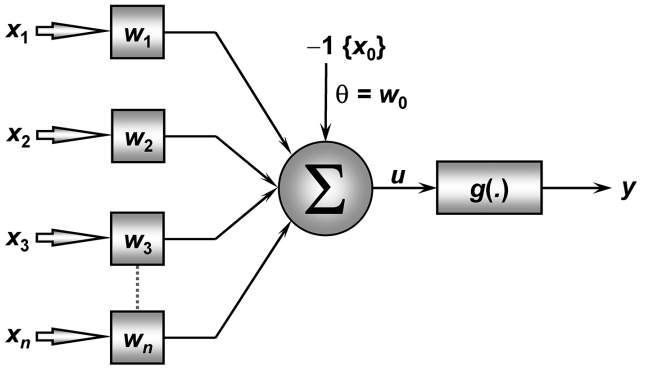

Figura 3.2 – Ilustração da rede *Perceptron*

Embora seja uma rede simples, o *Perceptron* teve o potencial de atrair, quando de sua proposição, diversos pesquisadores que aspiravam investigar essa promissora área de pesquisa para a época, recebendo-se ainda especial atenção da comunidade científica que também trabalhava com inteligência artificial.

Visando aspectos de implementação computacional, conforme será abordado na seção 3.4, observa-se na figura 3.2 que o valor do limiar de ativação $\{\theta\}$ foi assumido (sem qualquer perda de interpretabilidade) como sendo também um termo de ponderação $\{w_0\}$, tendo-se então o valor unitário negativo como respectiva entrada.

Com base no exposto no capítulo anterior, a rede *Perceptron* pertence à arquitetura *feedforward* de camada única, pois o fluxo de informações em sua estrutura reside sempre no sentido da camada de entrada em direção à camada neural de saída, inexistindo-se qualquer tipo de realimentação de valores produzidos pelo seu único neurônio.

3.2 – Princípio de funcionamento do *Perceptron*

Assim como verificado na simplicidade estrutural do *Perceptron*, o seu princípio de funcionamento é também extremamente simples.

A partir da análise da figura 3.2, observa-se que cada uma das entradas $\{x_i\}$, as quais representam informações sobre o comportamento do processo a ser mapeado, será inicialmente ponderada pelos pesos sinápticos $\{w_i\}$ a fim de quantificar a importância de cada uma frente aos objetivos funcionais atribuídos ao neurônio, cujo propósito será então mapear o comportamento entrada/saída do referido processo.

Em seguida, o valor resultante da composição de todas as entradas já devidamente ponderadas pelos seus respectivos pesos, adicionado ainda do limiar de ativação $\{\theta\}$, é repassado como argumento da função de ativação, cujo resultado de retorno será a saída $\{y\}$ produzida pelo *Perceptron*.

Em termos matemáticos, o processamento interno realizado pelo *Perceptron* pode ser descrito pelas seguintes expressões:

$$\begin{cases} u = \sum_{i=1}^{n} w_i \cdot x_i - \theta & (3.1) \\ y = g(u) & (3.2) \end{cases}$$

onde x_i são as entradas da rede, w_i é o peso (ponderação) associado à *i*-ésima entrada, θ é o limiar de ativação, $g(.)$ é a função de ativação e u é o potencial de ativação.

Tipicamente, devido às suas características estruturais, as funções de ativação normalmente usadas no *Perceptron* são a função degrau ou degrau bipolar, conforme apresentadas na subseção 1.3.1. Assim, independente da função de ativação a ser utilizada, tem-se apenas duas possibilidades de valores a serem produzidos pela sua saída, ou seja, valor 0 ou 1 se for considerada a função de ativação degrau, ou ainda, valor -1 ou 1 se for assumida a função degrau bipolar.

Em relação às entradas $\{x_i\}$, estas podem assumir quaisquer valores numéricos, ficando basicamente em função do tipo de problema a ser mapeado pelo *Perceptron*. Na prática, técnicas que normalizam as entradas, considerando os limites numéricos produzidos pelas funções de ativação adotadas, visam melhorar o desempenho computacional do processo de treinamento.

Resumindo, a tabela 3.1 explicita os aspectos característicos dos parâmetros relacionados com a dinâmica de funcionamento do *Perceptron*.

Tabela 3.1 – Aspectos dos parâmetros característicos do *Perceptron*

Parâmetro	**Variável representativa**	**Tipo característico**
Entradas	x_i (*i*-ésima entrada)	Reais ou binárias (advindas externamente)
Pesos sinápticos	w_i (associado a x_i)	Reais (iniciados aleatoriamente)
Limiar	θ	Real (iniciado aleatoriamente)
Saída	y	Binária
Função de ativação	$g(.)$	Degrau ou degrau bipolar
Processo de treinamento	-----	Supervisionado
Regra de aprendizado	-----	Regra de Hebb

O ajuste dos pesos e limiar do *Perceptron* é efetuado utilizando processo de treinamento supervisionado, isto é, para cada amostra dos sinais de entrada se tem a respectiva saída (resposta) desejada. Como o *Perceptron* é tipicamente usado em problemas de reconhecimento de padrões, sendo que sua saída pode assumir somente dois valores possíveis, então cada um de tais valores será associado a uma das duas classes que o *Perceptron* estará identificando.

Mais especificamente, considerando um problema envolvendo a classificação dos sinais de entrada em duas classes possíveis, denominadas de *classe* A e *classe* B, seria então possível (assumindo como exemplo a função de ativação degrau bipolar) atribuir o valor -1 para representar as amostras pertencentes à *classe* A, ao passo que o valor 1 seria usado para aquelas da *classe* B, ou vice-versa.

3.3 – Análise matemática do *Perceptron*

A partir da análise matemática do *Perceptron*, considerando a função de ativação sinal, torna-se possível verificar que tal elemento pode ser considerado um típico caso de discriminador linear. Para mostrar esta situação, assume-se um *Perceptron* com apenas duas entradas, conforme ilustrado na figura 3.3.

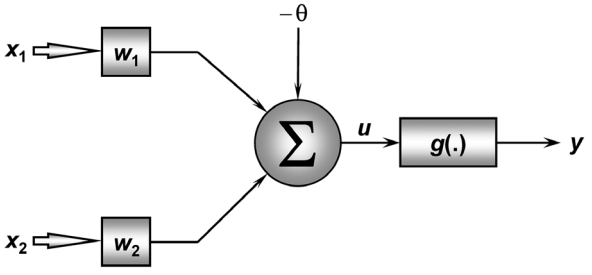

Figura 3.3 – *Perceptron* constituído de duas entradas

Em termos matemáticos, a saída do *Perceptron*, tendo-se assim como ativação a função sinal definida em (1.5), será dada por:

$$y = \begin{cases} 1\,, \text{ se } \sum w_i \cdot x_i - \theta \geq 0 \Leftrightarrow w_1 \cdot x_1 + w_2 \cdot x_2 - \theta \geq 0 \\ -1\,, \text{ se } \sum w_i \cdot x_i - \theta < 0 \Leftrightarrow w_1 \cdot x_1 + w_2 \cdot x_2 - \theta < 0 \end{cases} \tag{3.3}$$

Sendo as desigualdades em (3.3) representadas por uma expressão de primeiro grau (linear), a fronteira de decisão para esta instância (*Perceptron* de duas entradas) será então uma reta cuja equação é definida por:

$$w_1 \cdot x_1 + w_2 \cdot x_2 - \theta = 0 \tag{3.4}$$

Portanto, pode-se concluir que o *Perceptron* se comporta como um classificador de padrões cuja função é dividir classes que sejam linearmente separáveis. Para o caso do *Perceptron* de duas entradas, a figura 3.4 ilustra uma reta posicionada na fronteira de separabilidade.

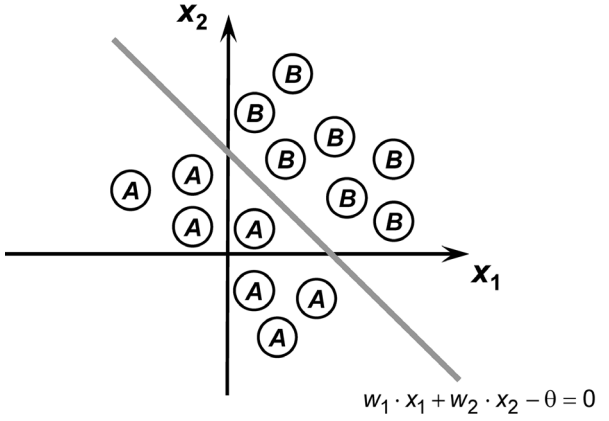

Figura 3.4 – Ilustração de fronteira de separação (neurônio com duas entradas)

Em suma, para a circunstância verificada na figura 3.4, o *Perceptron* consegue então dividir duas classes linearmente separáveis, sendo que quando a saída deste for -1 significa que os padrões (*classe A*) estão localizados abaixo da fronteira (reta) de separação; caso contrário, quando a saída for 1 indica que os padrões (*classe B*) estão acima desta fronteira. A figura 3.5 ilustra uma configuração em que as classes não são linearmente separáveis, isto é, uma única reta seria incapaz de separar as duas classes do problema.

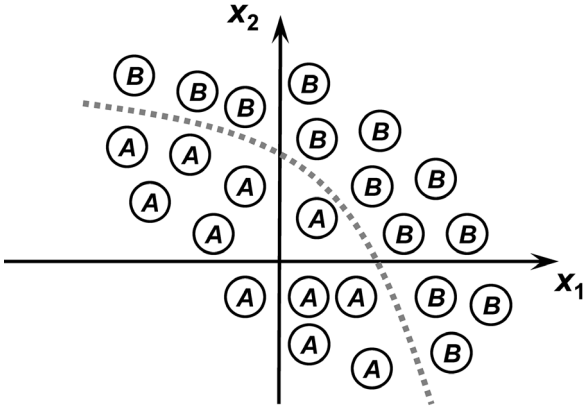

Figura 3.5 – Ilustração de fronteira não linearmente separável

Considerando o fato de o *Perceptron* ser constituído de três entradas (três dimensões), a fronteira de separação seria representada por um plano; sendo que, para dimensões superiores, tais fronteiras seriam hiperplanos.

Finalmente, conclui-se que a condição para que o *Perceptron* de camada simples possa ser utilizado como um classificador de padrões consiste em que as classes do problema a ser mapeado sejam linearmente separáveis. Este princípio condicional foi enunciado como Teorema de Convergência do *Perceptron* [Minsky & Papert, 1969].

3.4 – Processo de treinamento do *Perceptron*

O ajuste dos pesos e limiar do *Perceptron*, visando-se propósitos de classificação de padrões que podem pertencer a uma das duas únicas classes possíveis, é realizado por meio da regra de aprendizado de Hebb [Hebb, 1949].

Resumidamente, se a saída produzida pelo *Perceptron* está não coincidente com a saída desejada, os pesos sinápticos e limiares da rede serão então incrementados (condição excitatória) proporcionalmente aos valores de seus sinais de entrada; caso contrário, ou seja, a saída produzida pela rede é igual ao valor desejado, os pesos sinápticos e limiar permanecerão então inalterados (condição inibitória). 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, as regras de ajuste dos pesos sinápticos $\{w_i\}$ e do limiar $\{\theta\}$ do neurônio podem ser expressas, respectivamente, pelas seguintes expressões:

$$\begin{cases} w_i^{atual} = w_i^{anterior} + \eta \cdot (d^{(k)} - y) \cdot \boldsymbol{x}_i^{(k)} & (3.5) \\ \theta_i^{atual} = \theta_i^{anterior} + \eta \cdot (d^{(k)} - y) \cdot (-1) & (3.6) \end{cases}$$

Entretanto, em termos de implementação computacional, torna-se mais conveniente tratar as expressões anteriores em sua forma vetorial. Como a mesma regra de ajuste é aplicada tanto para os pesos sinápticos como para o limiar, pode-se então inserir o valor do limiar $\{\theta\}$ dentro do vetor de pesos sinápticos. De fato, o valor do limiar é também uma variável que deve ser ajustada a fim de se realizar o treinamento do *Perceptron*. Portanto, as expressões (3.5) e (3.6) podem ser representadas por uma única expressão vetorial dada por:

$$\boldsymbol{w}^{atual} = \boldsymbol{w}^{anterior} + \eta \cdot (d^{(k)} - y) \cdot \boldsymbol{x}^{(k)} \tag{3.7}$$

Em notação algorítmica, a expressão anterior é equivalente à seguinte:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta \cdot (d^{(k)} - y) \cdot \boldsymbol{x}^{(k)} \tag{3.8}$$

onde: $\boldsymbol{w} = [\theta \quad w_1 \quad w_2 \ \ldots \ w_n]^T$ é o vetor contendo o limiar e os pesos;

$\boldsymbol{x}^{(k)} = [-1 \quad x_1^{(k)} \quad x_2^{(k)} \cdots x_n^{(k)}]^T$ é a *k*-ésima amostra de treinamento;

$d^{(k)}$ é o valor desejado para a *k*-ésima amostra de treinamento;

y é valor da saída produzida pelo *Perceptron*;

η é uma constante que define a taxa de aprendizagem da rede.

A taxa de aprendizagem $\{\eta\}$ exprime o quão rápido o processo de treinamento da rede estará sendo conduzido rumo à sua convergência (estabilização). A escolha de η deve ser realizada com cautela para evitar instabilidades no processo de treinamento, sendo que normalmente se adotam valores pertencentes ao intervalo compreendido em $0 < \eta < 1$.

A fim de elucidar a notação utilizada para o vetor $\boldsymbol{x}^{(k)}$ representando a *k*-ésima amostra de treinamento, assim como o respectivo valor desejado $d^{(k)}$, supõe-se que um problema a ser mapeado pelo *Perceptron* tenha três entradas $\{x_1, x_2, x_3\}$. Assume-se que o conjunto de treinamento seja composto de apenas quatro amostras, constituídas dos seguintes valores: $\Omega^{(x)}$ = { [0,1 0,4 0,7]; [0,3 0,7 0,2]; [0,6 0,9 0,8]; [0,5 0,7 0,1] }. Considerando-se ainda que os respectivos valores de saída para cada uma dessas amostras seja dado por $\Omega^{(d)}$ = { [1]; [-1]; [-1]; [1]} , então se pode adotar a seguinte forma matricial para suas representações:

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

Nesta condição, alternativamente, pode-se extrair dessas matrizes cada um dos vetores $\boldsymbol{x}^{(k)}$, com seu respectivo valor $d^{(k)}$, os quais representarão cada uma das amostras de treinamento, da seguinte forma:

$$\boldsymbol{x}^{(1)} = [-1 \quad 0{,}1 \quad 0{,}4 \quad 0{,}7]^T; \text{ com } d^{(1)} = 1$$

$$\boldsymbol{x}^{(2)} = [-1 \quad 0{,}3 \quad 0{,}7 \quad 0{,}2]^T; \text{ com } d^{(2)} = -1$$

$$\boldsymbol{x}^{(3)} = [-1 \quad 0{,}6 \quad 0{,}9 \quad 0{,}8]^T; \text{ com } d^{(3)} = -1$$

$$\boldsymbol{x}^{(4)} = [-1 \quad 0{,}5 \quad 0{,}7 \quad 0{,}1]^T; \text{ com } d^{(4)} = 1$$

Entretanto, em se tratando de implementações computacionais, a disposição dos dados por meio de variáveis indexadas se torna bem mais apropriada para a manipulação dos índices das amostras.

Assim, a sequência passo a passo para o treinamento do *Perceptron* pode ser explicitada por intermédio de algoritmo em pseudocódigo, conforme se segue.

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 $\boldsymbol{w}$ 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> $u \leftarrow \boldsymbol{w}^T \cdot \boldsymbol{x}^{(k)}$;
    <6.2.2> $y \leftarrow \text{sinal}(u)$;
    <6.2.3> Se $y \neq d^{(k)}$
      <6.2.3.1> Então { $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta \cdot (d^{(k)} - y) \cdot \boldsymbol{x}^{(k)}$; *erro* ← "existe"
  <6.3> *época* ← *época* + 1;
  Até que: *erro* = "inexiste"

Fim {Algoritmo *Perceptron* – Fase de Treinamento}

Analisando o algoritmo envolvido com o treinamento do *Perceptron*, verifica-se que a variável ***época*** estará incumbida de contabilizar o próprio número de épocas de treinamento, ou seja, quantas vezes serão necessárias apresentar todas as amostras do conjunto de treinamento visando o ajuste do vetor de pesos $\{\boldsymbol{w}\}$. A rede será considerada treinada (ajustada) quando inexistir erro entre os valores desejados em comparação com aqueles produzidos em suas saídas.

Uma vez que esteja treinada, a rede estará apta para proceder com a tarefa de classificação de padrões frente às novas amostras que serão apresentadas em suas entradas. Portanto, as instruções para colocar o *Perceptron* em operação, após a realização de seu treinamento, são sintetizadas no algoritmo seguinte.

Início {Algoritmo *Perceptron* – Fase de Operação}

<1> Obter uma amostra a ser classificada { $\boldsymbol{x}$ };
<2> Utilizar o vetor $\boldsymbol{w}$ 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}

Portanto, pode-se abstrair que o processo de treinamento do *Perceptron* tende a mover continuamente o hiperplano de classificação até que seja alcançada uma fronteira de separação que permite dividir as duas classes.

A figura 3.6 mostra uma ilustração do processo de treinamento do *Perceptron* visando o alcance desta fronteira de separabilidade. Para fins didáticos, considera-se uma rede composta de apenas duas entradas $\{x_1$ e $x_2\}$.

Após a primeira época de treinamento (1), constata-se que o hiperplano está ainda bem longínquo da fronteira de separabilidade das classes, ao passo que esta distância tende a decrescer cada vez mais na medida em que se transcorre as épocas de treinamento.

Em consequência, quando o *Perceptron* já estiver convergido, significará que tal fronteira foi finalmente alcançada, sendo que as saídas produzidas pelo *Perceptron* a partir deste momento serão todas iguais àquelas desejadas.

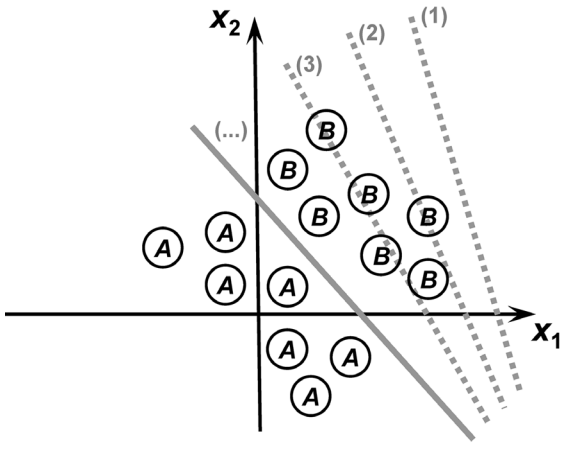

Figura 3.6 – Ilustração do processo de treinamento do *Perceptron*

Analisando a figura 3.6, observa-se ainda a possibilidade de se ter diversas retas que permitam separar as duas classes envolvidas com o problema. A figura 3.7 ilustra um conjunto de eventuais retas que são também passíveis de separar tais classes.

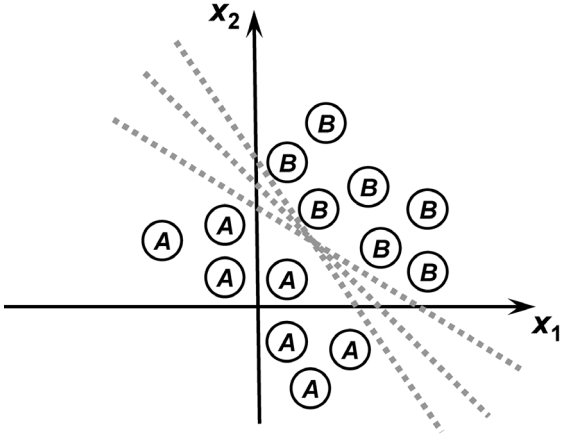

Figura 3.7 – Ilustração de conjunto de retas passíveis de separar as classes

Assim, para este problema de classificação de padrões usando o *Perceptron*, pode-se também abstrair que a reta de separabilidade a ser produzida após o seu treinamento não é única, sendo que, nesses casos, o número de épocas pode também variar.

Na sequência são apresentados alguns aspectos práticos envolvendo o processo de treinamento do *Perceptron*.

- A rede divergirá se o problema for não linearmente separável. A estratégia a ser usada nesta ocorrência é limitar o processo por meio da especificação de um número máximo de épocas;
- Quando a faixa de separabilidade entre as duas classes do problema for muito estreita, o seu processo de treinamento pode implicar em instabilidades. Em tais casos, assumindo-se um valor de taxa de aprendizado $\{\eta\}$ bem pequeno, a instabilidade da convergência pode ser então aliviada;
- A quantidade de épocas necessárias para a convergência do processo de treinamento do *Perceptron* varia em função dos valores iniciais que foram atribuídos ao vetor de pesos $\{\boldsymbol{w}\}$, assim como da disposição espacial das amostras de treinamento e do valor especificado para a taxa de aprendizado $\{\eta\}$;
- Quanto mais próxima a superfície de decisão estiver da fronteira de separabilidade, menos épocas para a convergência do *Perceptron* são geralmente necessárias;
- A normalização das entradas para domínios apropriados contribui para incrementar o desempenho do treinamento.

3.5 – Exercícios

1) Explique como se processa a regra de Hebb no contexto do algoritmo de aprendizado do *Perceptron*.

2) Mostre por intermédio de gráficos ilustrativos como pode ocorrer a instabilidade no processo de convergência do *Perceptron* quando da utilização de valores inapropriados para a taxa de aprendizado.

3) Explique por que o *Perceptron* somente consegue classificar padrões cuja fronteira de separação entre as classes seja linear.

4) Em termos de implementação computacional descreva a importân-

cia de tratarmos o limiar de ativação $\{\theta\}$ como um dos elementos do vetor de pesos $\{\boldsymbol{w}\}$.

5) Seja um problema de classificação de padrões que se desconhece *a priori* se as suas duas classes são ou não-separáveis linearmente. Elabore uma estratégia para verificar a possível aplicação do *Perceptron* em tal problema.

6) Dois projetistas de instituições diferentes estão aplicando uma rede *Perceptron* para mapear o mesmo problema de classificação de padrões. Discorra se é correto afirmar que ambas as redes convergirão com o mesmo número de épocas.

7) Em relação ao exercício anterior, considere-se que ambas as redes já estão devidamente treinadas. Para um conjunto contendo 10 novas amostras que devem ser identificadas, explique se os resultados produzidos por ambas serão os mesmos.

8) Seja um problema de classificação de padrões que seja linearmente separável composto de 50 amostras. Em determinada época de treinamento observou-se que somente para uma dessas amostras a rede não estava produzindo a resposta desejada. Discorra se é então necessário apresentar novamente todas as 50 amostras na próxima época de treinamento.

9) Considere um problema de classificação de padrões composto de duas entradas $\{x_1$ e $x_2\}$, cujo conjunto de treinamento é composto pelas seguintes amostras de treinamento:

x_1	x_2	*Classe*
0,75	0,75	*A*
0,75	0,25	*B*
0,25	0,75	*B*
0,25	0,25	*A*

Mostre se é possível aplicar o *Perceptron* na resolução deste problema.

10) Explique de forma detalhada quais seriam as eventuais limitações do *Perceptron* se considerarmos o seu limiar de ativação nulo.

3.6 – Projeto prático

Pela análise de um processo de destilação fracionada de petróleo observou-se que determinado óleo poderia ser classificado em duas classes de pureza $\{P_1$ e $P_2\}$ a partir da medição de três grandezas $\{x_1, x_2$ e $x_3\}$, que representam algumas de suas propriedades físico-químicas. A equipe de engenheiros e cientistas pretende usar uma rede *Perceptron* para executar a classificação automática das duas classes.

Assim, baseado nas informações coletadas do processo, formou-se o conjunto de treinamento apresentado no apêndice I, tomando por convenção o valor -1 para óleo pertencente à classe P_1 e o valor 1 para óleo pertencente à classe P_2.

Para tanto, o neurônio constituinte do *Perceptron* terá então três entradas e uma saída conforme ilustrado na figura 3.8.

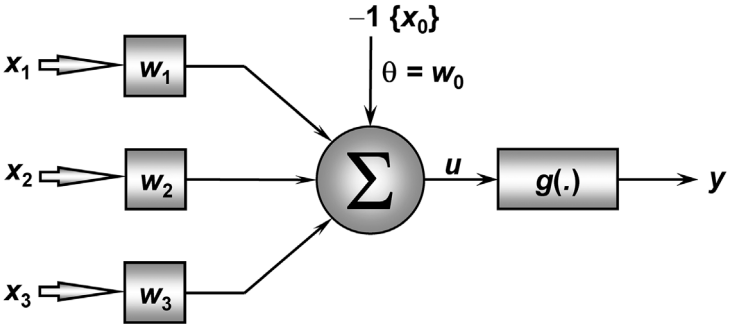

Figura 3.8 – Arquitetura do *Perceptron* para o projeto prático

Utilizando o algoritmo supervisionado de Hebb (regra de Hebb) para classificação de padrões, e assumindo-se a taxa de aprendizagem como 0,01, faça as seguintes atividades:

1) Execute cinco treinamentos para a rede *Perceptron*, iniciando-se o vetor de pesos $\{\boldsymbol{w}\}$ em cada treinamento com valores aleatórios entre zero e um. Se for o caso, reinicie o gerador de números aleatórios em cada treinamento de tal forma que os elementos do vetor de pesos iniciais não sejam os mesmos. O conjunto de treinamento encontra-se no apêndice I.

2) Registre os resultados dos cinco treinamentos na tabela 3.2 apresentada a seguir.

Tabela 3.2 – Resultados dos treinamentos do *Perceptron*

Treinamento	Vetor de pesos iniciais				Vetor de pesos finais				Número de épocas
	w_0	w_1	w_2	w_3	w_0	w_1	w_2	w_3	
1º(T1)									
2º(T2)									
3º(T3)									
4º(T4)									
5º(T5)									

3) Após o treinamento do *Perceptron*, coloque este em operação, aplicando-o na classificação automática das amostras de óleo da tabela 3.3, indicando ainda nesta tabela aqueles resultados das saídas (Classes) referentes aos cinco processos de treinamento realizados no item 1.

Tabela 3.3 – Amostras de óleo para validar a rede *Perceptron*

Amostra	x_1	x_2	x_3	y (T1)	y (T2)	y (T3)	y (T4)	y (T5)
1	-0,3665	0,0620	5,9891					
2	-0,7842	1,1267	5,5912					
3	0,3012	0,5611	5,8234					
4	0,7757	1,0648	8,0677					
5	0,1570	0,8028	6,3040					
6	-0,7014	1,0316	3,6005					
7	0,3748	0,1536	6,1537					
8	-0,6920	0,9404	4,4058					
9	-1,3970	0,7141	4,9263					
10	-1,8842	-0,2805	1,2548					

4) Explique por que o número de épocas de treinamento, em relação a esta aplicação, varia a cada vez que executamos o treinamento do *Perceptron*.

5) Para a aplicação em questão, discorra se é possível afirmar se as suas classes são linearmente separáveis.

Frank Rosenblatt