

CAPÍTULO V

PROGRAMAÇÃO LINEAR E REDES NEURAIIS

V.1 - Introdução:

Uma nova topologia de redes neurais é apresentada aqui para solução do problema de Programação Linear baseado no método de Direções Viáveis em dois estágios desenvolvido inicialmente para o problema de Programação Não-Linear (vide[03]). Tomamos um ponto interior de nossa região viável e inicializamos a rede gerando uma trajetória que converge para a solução ótima de nosso problema. Como veremos, o número de neurônios da rede cresce linearmente com o tamanho do problema e os neurônios são conectados de forma esparsa.

V.2 - Programação Linear e o Método das Direções Viáveis em Dois Estágios:

Trabalharemos com a seguinte formulação para nosso problema de Programação Linear (PPL):

$$\begin{aligned} \min & c^T x \\ \text{s.a.} & Ax \leq b \\ \text{onde } & c, x \in \mathfrak{R}^n, A \in \mathfrak{R}^{m \times n} \text{ e } b \in \mathfrak{R}^m \end{aligned} \quad (1)$$

O conjunto de restrições de desigualdades $Ax \leq b$ define um poliedro convexo contido em \mathfrak{R}^n . Temos que, se nosso problema admite uma solução ótima então pelo menos um vértice do conjunto de restrições também será ótimo (vide [04]). Procuraremos então uma solução ótima entre os vértices do poliedro de restrições.

O método de direções viáveis discutido aqui (analogamente ao método gradiente) parte de um ponto inicial interior ($Ax^0 < b$) e gera uma direção de busca a partir desse ponto. Repetimos o processo, construindo então uma sequência $x^0, x^1, x^2 \dots$ e esperamos que ela convirja para uma solução ótima de nosso problema.

Sejam $f(x) = c^T x$ e $g(x) = Ax - b$ duas funções correspondentes à função objetivo e as restrições de (1) respectivamente. Como discutido em [03], buscaremos a minimização da função lagrangeana $l(x, \lambda)$ associada ao problema. Assim:

$$\begin{aligned} \min & l(x, \lambda) = c^T x + \lambda^T (Ax - b) \\ \text{s.a.} & x \in \mathfrak{R}^n \end{aligned}$$

Neste caso, se x^* é um ponto de mínimo local para o problema, existirá um vetor $\lambda \geq 0$ (condição necessária de otimalidade) tal que:

$$\nabla f(x^*) + \lambda^T \nabla g(x^*) = 0$$

$$\text{ou ainda: } c^T + \lambda^T A = 0.$$

Assim, se tomamos um ponto $x_0 \in \mathfrak{R}^n$, a direção de descida correspondente à minimização da função lagrangeana será:

$$d = -A^T \lambda - c$$

Note entretanto, que precisamos computar o vetor λ antes de obter a direção d . Na sugestão apresentada em [03], devemos resolver o sistema:

$$d = -\left(\nabla f(x) - \sum_{i=1}^m \mathbf{I}_i \nabla g_i(x)\right)$$

$$d^T \nabla g_i(x) = -r_i(x) \mathbf{I}_i g_i(x) \quad \text{para } i = 1, \dots, m$$

onde $r_i(x) > 0$.

Note em nosso caso que: $\nabla g_i(x) = A_i$ e $g_i(x) = (Ax - b)_i$. Como $r_i(x) > 0$, teremos $d^T \nabla g_i(x) \geq 0$ (condição necessária de otimalidade - Luemberger[04]).

Reescrevendo o sistema acima em notação matricial temos:

$$\begin{cases} d = -c - A^T \mathbf{I} & (a) \\ d^T A = -R \mathbf{G} \mathbf{I} & (b) \end{cases}$$

$$\text{onde: } R = \begin{pmatrix} r_{11} & & 0 \\ & \ddots & \\ 0 & & r_{mm} \end{pmatrix}; \quad G = \begin{pmatrix} (Ax - b)_1 & & 0 \\ & \ddots & \\ 0 & & (Ax - b)_m \end{pmatrix}; \quad \mathbf{I} = \begin{pmatrix} \mathbf{I}_1 \\ \vdots \\ \mathbf{I}_m \end{pmatrix}$$

Substituindo (a) em (b) temos:

$$\begin{aligned} A(-c - A^T \mathbf{I}) &= -R \mathbf{G} \mathbf{I}; \\ -Ac - AA^T + R \mathbf{G} \mathbf{I} &= 0; \\ (AA^T - R \mathbf{G}) \mathbf{I} &= -Ac; \end{aligned} \quad (c)$$

Portanto, devemos resolver o sistema (c) acima para obtenção do vetor λ .

O processo iterativo obtido dessa forma, nos conduz para uma solução ótima do problema. Logo, quando nos aproximamos da solução ótima, teremos \mathbf{d} tendendo a zero, e \mathbf{I} (multiplicadores de lagrange) tendendo para valores não-negativos.

V.3 - A Rede Neural para o Sistema de Equações Lineares:

Antes de pensarmos na elaboração da topologia para o problema de programação linear, utilizada no método de direções viáveis em dois estágios, devemos resolver o sistema $(AA^T - R \mathbf{G}) \mathbf{I} = -Ac$. Para resolvê-lo usaremos uma idéia semelhante à apresentada no capítulo II.

Assim, fazendo $Y = (AA^T - R \mathbf{G})$ e $z = -Ac$, devemos computar:

$$\mathbf{I}_u = \sum_{i=1}^m y_{ui} \mathbf{I}_i - z_u$$

$$C_I \frac{d\mathbf{I}_i}{dt} = \sum_{u=1}^m y_{ui} \mathbf{I}_u - \frac{x_i}{R_I} = -\sum_{u=1}^m y_{ui} \left(\sum_{k=1}^m y_{uk} \mathbf{I}_k - z_u \right) - \frac{\mathbf{I}_i}{R_I}$$

onde C_I e R_I correspondem a capacitância e resistência respectivamente.

Temos portanto a seguinte situação:

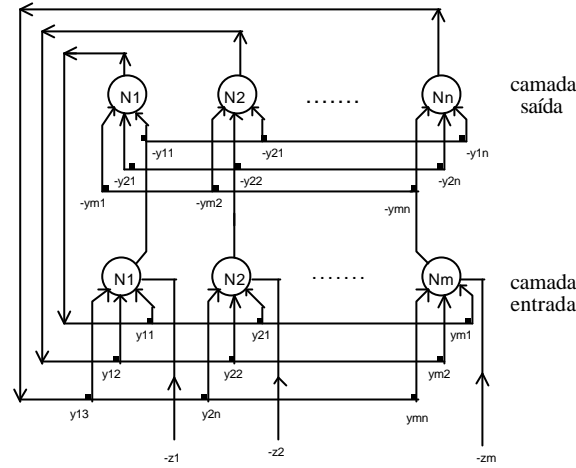


Figura V.1: Sistema Linear $Y\mathbf{I} = \mathbf{z}$

Note que $\mathbf{I}_u = \sum_{i=1}^m y_{ui} \mathbf{I}_i - z_u$ para $u=1, \dots, m$ (camada inferior) representa os estados de ativação da camada de entrada.

Para obter os estados de ativação da camada de saída devemos calcular $\mathbf{I}_j^{k+1} = \Delta t \cdot \frac{d\mathbf{I}_j^k}{dt} + \mathbf{I}_j^k$, onde $j=1, 2, \dots, m$ (para $\Delta t \ll 0$). Como discutido no capítulo II, utilizamos Δt para efeito de simulação.

V.4 - A Rede Neural p/ o Problema de Programação Linear:

Antes de definirmos propriamente uma topologia para a rede neural, vejamos o seguinte algoritmo, onde fazemos uma simulação dos passos presentes na implementação utilizando o método das direções viáveis em dois estágios:

Algoritmo: Direções-Viáveis;

Início

Leia $r \in \mathbb{R}^m$, \mathbf{I}_0 , x_0 ;

Repita

$d \leftarrow -A^T \mathbf{I} - c$; {calcula direção}

$x \leftarrow x + \Delta t \cdot d$;

Calcula: $G = \text{diag}((Ax - b)_i)$;

Calcula: $G\lambda$;

$\mathbf{I}_u \leftarrow AA^T \mathbf{I} - RG\mathbf{I} + Ac$; {obtem camada de saída do sistema linear}

$\frac{d\mathbf{I}_i}{dt} = -(AA^T - RG)^T \mathbf{I}_u - \frac{\mathbf{I}_i}{R_i}$; {para $i=1, \dots, m$ }

$\mathbf{I} \leftarrow \mathbf{I} + \Delta t \cdot \frac{d\mathbf{I}}{dt}$; {calcula camada de saída}

Até que $\|d\| \approx 0$;

Imprima (x);

fim.

Note no algoritmo que fazemos uma atualização alternada entre os vetores \mathbf{d} e \mathbf{I} , ou seja, não resolvemos completamente o sistema $(AA^T - RG)\mathbf{I} = -Ac$ antes da atualização do vetor \mathbf{d} !

Utilizaremos uma rede de 6 camadas na representação do problema de programação linear. Com o objetivo de simplificar a notação, representamos (na figura V.2) os índices correspondentes a cada coluna de nossa rede:

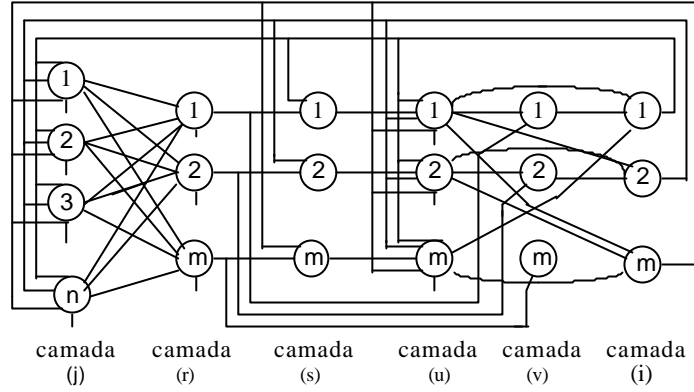


Figura V.2: A Rede Neural p/ o Método de Direções Viáveis em P. Linear

Representaremos por V_k os estados de ativação da camada k . Assim:

$$\left\{ \begin{array}{ll} \text{camada 1:} & 1 \leq j \leq n \\ \text{" 2:} & 1 \leq r \leq m \\ \text{" 3:} & 1 \leq s \leq m \\ \text{" 4:} & 1 \leq u \leq m \\ \text{" 5:} & 1 \leq v \leq m \\ \text{" 6:} & 1 \leq i \leq m \end{array} \right.$$

Cada uma das seis camadas tem uma função específica no cálculo da direção d . Para $t \geq 0$, os estados de ativação V_j (da camada 1) correspondem às coordenadas $x_j(t)$ de $x(t)$. Analogamente, os estados de ativação V_i (da camada 6) representam os estados $I_i(t)$ correspondentes à camada de saída do sistema $(AA^T - RG)I = -Ac$.

As camadas 2 e 3 (V_r e V_s) representam as diagonais das matrizes G e λG respectivamente.

As camadas 4 e 5, são utilizadas conjuntamente com a camada 6 na solução do sistema linear. Observe neste caso, que utilizamos 3 camadas, ao invés de 2, como discutido na seção anterior. Note entretanto, que λ é dependente das atualizações da matriz G (que também varia continuamente no tempo). Teremos portanto sinapses ligando os neurônios da camada 2 aos neurônios da camada 5 (vide Figura V.2).

Os estados de ativação V_u (da camada 4), são obtidos calculando-se:

$$(AA^T - RG)I + Ac$$

Para o cálculo de $d\lambda/dt$ (camada 6), devemos calcular:

$$-(AA^T - RG)^T((AA^T - RG)I = -Ac)$$

Resumindo, apresentamos abaixo as equações correspondendo a cada camada e sua respectiva notação matricial:

$$\begin{aligned}
C_x \frac{dV_j}{dt} &= - \sum_{i=1}^m a_{ij} V_i - c_j - \frac{V_j}{R_x} \quad \Leftrightarrow \quad d = -A^T \mathbf{I} - c; \\
V_r &= \sum_{j=1}^n a_{rj} V_j - b_r \quad \Leftrightarrow \quad G = \text{diag}((Ax - b)_i); \\
V_s &= w_{si} w_{sr} V_i V_r = V_i V_r \quad (\text{para } i = s = r) \quad \Leftrightarrow \quad \mathbf{I}G; \\
V_u &= \sum_{i=1}^m \sum_{k=1}^n a_{uk} a_{ik} V_i - r_{us} V_s \sum_{j=1}^n a_{uj} x_j + \sum_{k=1}^n a_{uk} C_k \quad (\text{para } s = u) \quad \Leftrightarrow \quad AA^T \mathbf{I} - RG \mathbf{I} + Ac; \\
V_v &= r_{vr} V_u V_r \quad (\text{para } u = v = r) \quad \Leftrightarrow \quad G((AA^T - RG) \mathbf{I} + Ac) \\
C_I \frac{dV_i}{dt} &= - \sum_{u=1}^m \sum_{k=1}^n a_{uk} a_{ik} V_u - r_{iv} V_s \left(\sum_{j=1}^n a_{ij} x_j - b_i \right) - \frac{V_i}{R_I} \quad (\text{para } v = i) \quad \Leftrightarrow \quad -(AA^T - RG)^T ((AA^T - RG) \mathbf{I} + Ac)
\end{aligned}$$

Observe, em uma simulação, que os estados de ativação das camadas 1 e 6 são obtidos após fazermos $x \leftarrow x + \Delta t \cdot d$ e $\mathbf{I} \leftarrow \mathbf{I} + \Delta t \cdot (d\mathbf{I} / dt)$ respectivamente.

Como discutido em [02], para garantir a convergência da rede para uma solução ótima devemos escolher com cuidado as capacitâncias C_x e C_I , já que estes parâmetros regulam a velocidade de atualização dos pontos \mathbf{x} e \mathbf{I} . As resistências R_I , podem ser consideradas arbitrariamente grandes.

A rede proposta em [02], possui um número de neurônios que cresce linearmente com $m+n$, e é relativamente esparça. Isto possibilita implementações analógicas (circuitos VLSI) tornando o método apresentado, bastante atrativo para problemas que envolvam programação linear.