Estudio numérico de un algoritmo que usa redes neuronales para resolver el problema de programación lineal

WILMER SANCHEZ GRUESO

Universidad del Cauca
Facultad de Ciencias Naturales, Exactas y de la Educación
Departamento de Matemáticas
Popayán, Cauca

2017

Estudio numérico de un algoritmo que usa redes neuronales para resolver el problema de programación lineal

TRABAJO DE GRADO

En la modalidad de seminario, presentado como requisito parcial para optar al título de Matemático.

WILMER SANCHEZ GRUESO

DIRECTOR:
Mg. FAVIÁN ARENAS APARICIO

Universidad del Cauca
Facultad de Ciencias Naturales, Exactas y de la Educación
Departamento de Matemáticas
Popayán, Cauca

2017

Nota de aceptación

_____

_____

_____

_____

Director _____

Mg. Favián Arenas Aparicio

Jurado _____

Mg. Hevert Vivas

Jurado _____

Mg. Mauricio Maca

Fecha de sustentación: 8 de Mayo de 2017, Popayán.

Índice general

Capítulo 1

Introducción

La *programación lineal* (PL) es una técnica de la investigación de operaciones (IO) que está diseñada para modelos que tengan una función objetivo y un conjunto de restricciones lineales; lo que se busca es maximizar o minimizar dicha función, las soluciones a este problema son de gran importancia en muchas disciplinas como la planeación estratégica, problemas combinatorios, investigación operativa, etc. [3] [5].

Los métodos tradicionales para resolver problemas de PL son el *método simplex* que consiste en un sucesión de operaciones de pivoteo en el cual la función objetivo decrece continuamente hasta alcanzar un mínimo [3]. Otra alternativa es utilizar los *métodos de penalización* y de *barrera* los cuales consisten en transformar problemas de optimización con restricciones a uno que ya no tenga restricciones, añadiéndole a la función objetivo un término de penalización $P(\boldsymbol{x})$ que toma un valor muy grande cuando no se cumplen las restricciones. El problema de minimización sin restricciones se puede abordar con los métodos de optimización clásicos [8] como: *Los multiplicadores de Lagrange* y el *método de redes neuronales artificiales*. Este último se estudiará en este documento.

Otro concepto importante es la red neuronal artificial (RNA), la cual es un sistema de procesamiento de información que funciona de manera similar a las redes neuronales biológicas. El elemento más importante de éstas, es la célula nerviosa o neurona y su misión es recoger la información que llega a ellas en forma de impulsos eléctricos o químicos procedentes de otras neuronas o receptores, por medio de un código propio, ella se activa o inhibe y trasmite el mensaje codificado en forma de frecuencia o de impulsos por medio de su axón.Por medio de sus ramificaciones el axón efectúa la distribución espacial de los mensajes a las demás neuronas siguientes [6].

Las RNA es una herramienta muy importante para resolver problemas de optimización. Entre los tipos de redes encontramos las redes de *Hopfield* que fue presentada en 1982, basándose en los modelos de redes de *McCulloch* y *Pitts*; dicha red tiene asociada una función de energía, la cual es equivalente a la función objetivo (penalizada) que necesita ser minimizada [14]. La función de energía permite transformar el problema de programación lineal a un sistema de ecuaciones diferenciales no lineal, en la que los puntos estacionarios del sistema son la solución al problema de programación lineal [2] la ventaja de utilizar este modelo es que presenta una alternativa en forma continua para encontrar la solución del problema.

Capítulo 2

Preliminares

En este capítulo pretendemos familiarizar al lector con la teoría necesaria para solucionar el problema programación lineal mediante un método basado en redes neuronales artificiales. Se presentarán conceptos de estabilidad de sistemas dinámicos, método de penalización y definición de un problema de programación lineal.

1. Estabilidad de sistemas dinámicos

Dado un sistema de ecuaciones diferenciales de la forma

$$\boldsymbol{x}'(t) = f(\boldsymbol{x}) \tag{2.1}$$

Decimos que es un sistema es autónomo, si la expresión (2.1) no depende explícitamente del tiempo. Si existen puntos $\boldsymbol{x} \in \mathbb{R}^n$ tales que $f(\boldsymbol{x}) = \mathbf{0}$ entonces a estos se les llama puntos críticos del sistema autónomo; además, si ésto se cumple, se tendrán soluciones constantes en el tiempo.

Definición 2.1 (Estabilidad)**.** *Dado el sistema (2.1), se dice que* $\boldsymbol{x}^*$ *es*

un punto estable si:
Para todo $\epsilon > 0$*, existe* $\delta > 0$ *tal que la solución del sistema* $\boldsymbol{x} = \phi(t)$ *satisface:*

$$\|\phi(0) - \boldsymbol{x}^*\| < \delta \text{ entonces } \|\phi(t) - \boldsymbol{x}^*\| < \epsilon \text{ para todo } t \geq 0.$$

La **Figura 2.1**, ilustra una curva solución que presenta estabilidad cerca del origen.

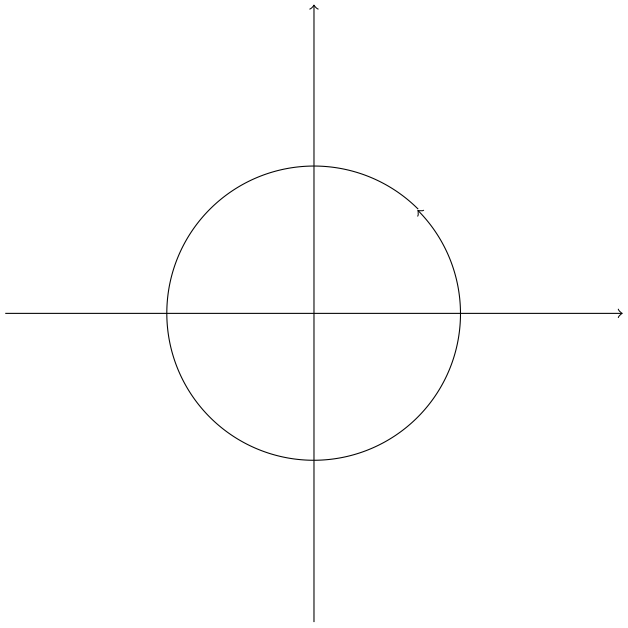

Figura 2.1: Estabilidad

Definición 2.2 (Asintóticamente estable)**.** *Si* $\boldsymbol{x}^*$ *es estable y si existe* $\delta > 0$ *con* $0 < \delta' < \delta$ *tal que, si una solución* $\boldsymbol{x} = \boldsymbol{\phi}(t)$ *satisface:*

$$\|\boldsymbol{\phi}(0) - \boldsymbol{x}^*\| < \delta' \text{ entonces } \lim_{t \to \infty} \boldsymbol{\phi}(t) = \boldsymbol{x}^*,$$

entonces se dice que $\boldsymbol{x}^*$ *es asintóticamente estable.*

En la **Figura 2.2** se ilustra una curva solución que representa la estabilidad asintótica del origen.

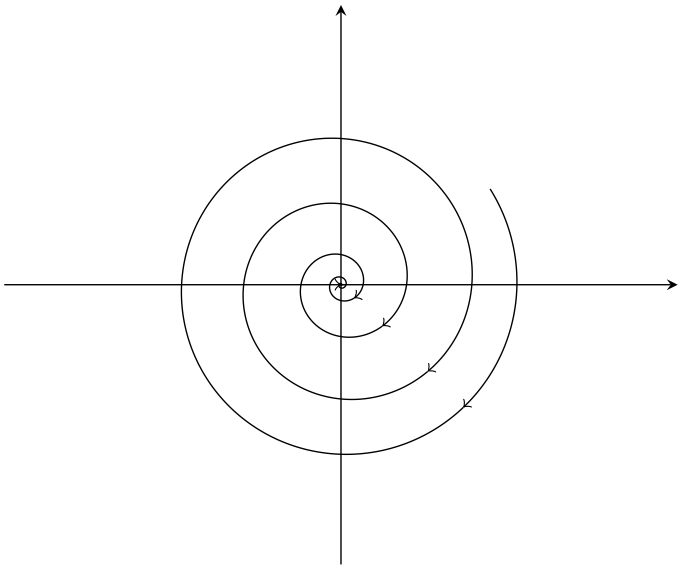

Figura 2.2: Estabilidad asintótica

El siguiente teorema permite estudiar la estabilidad de un sistema dinámico mediante una funcion de energía [9]

Teorema 1.1. *Sea E un subconjunto abierto de $\mathbb{R}^n$ que contiene a $\boldsymbol{x}^*$. Supongamos que $f \in C^1(E)$ y que $f(\boldsymbol{x}^*) = \mathbf{0}$. Supongamos que existe una función de valor real $V \in C^1(E)$ que satisface $V(\boldsymbol{x}^*) = 0$ y $V(\boldsymbol{x}) > 0$ si $\boldsymbol{x} \neq \boldsymbol{x}^*$. Entonces*

1. *Si $V'(\boldsymbol{x}) \leq 0$ para todo $\boldsymbol{x} \in E$, $\boldsymbol{x}^*$ es estable.*
2. *Si $V'(\boldsymbol{x}) < 0$ para todo $\boldsymbol{x} \in E \setminus \{\boldsymbol{x}^*\}$, $\boldsymbol{x}^*$ es asintóticamente estable.*
3. *Si $V'(\boldsymbol{x}) > 0$ para todo $\boldsymbol{x} \in E \setminus \{\boldsymbol{x}^*\}$, $\boldsymbol{x}^*$ es inestable.*

La función del teorema anterior que cumple $V'(\boldsymbol{x}) \leq 0$ para todo $\boldsymbol{x} \in E$, se le denomina ***función de Liapunov*** y en el caso $V'(\boldsymbol{x}) < 0$ para todo $\boldsymbol{x} \in E \setminus \{\boldsymbol{x}^*\}$ se le denomina ***función de Liapunov estricta***.

2. Problema de programación lineal

La programación lineal es una técnica de la investigación de operaciones que está diseñada para modelos que tengan una función objetivo y un conjunto de restricciones lineales, lo que se busca es maximizar o minimizar dicha función. A este modelo se le conoce como problema de programación lineal(PL).
Un problema de programación lineal puede expresarse de la siguiente forma:

$$\begin{array}{ll} Minimizar & z = \boldsymbol{c}^T \boldsymbol{x} \\ Sujeto\, a & A\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x}_\alpha \leq \boldsymbol{x} \leq \boldsymbol{x}_\beta, \end{array} \tag{2.2}$$

[1] donde:

$$\begin{array}{l} \boldsymbol{c} \in \mathbb{R}^n \\ A \in \mathbb{R}^{m \times n} \\ \boldsymbol{x} \in \mathbb{R}^n \\ \boldsymbol{b} \in \mathbb{R}^m \\ \boldsymbol{x}_\alpha, \boldsymbol{x}_\beta \in \mathbb{R}^n \end{array}$$

Definición 2.3. *Sea* $X = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{x}_\alpha \leq \boldsymbol{x} \leq \boldsymbol{x}_\beta\}$. *Al conjunto* $\widehat{X} = \{\boldsymbol{x} \in X : A\boldsymbol{x} = b\}$ *se le denomina* ***región factible*** *y diremos que* $\boldsymbol{x}^*$ *es una* ***solución factible*** *si* $\boldsymbol{x}^* \in \widehat{X}$. *A una solución del sistema*

[1] En este contexto dos vectores $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$ satisfacen $\boldsymbol{u} \leq \boldsymbol{v}$ si y solo si $u_i \leq v_i$ para todo $i = 1, 2, ..., n$.

(2.2) se le llamará ***solución factible óptima*** *y por último a la función* $z = \boldsymbol{c}^T\boldsymbol{x}$ *se le denominará* ***función objetivo****.*

3. Método de penalización

El método de penalización es un procedimiento para aproximar problemas de optimización con restricciones a uno que ya no tenga restricciones, tal aproximación se logra añadiéndole a la función objetivo un término de penalización que tiene un valor muy grande cuando no se cumplen las restricciones.

Consideremos el problema

$$\begin{array}{ll} Minimizar & f(\boldsymbol{x}) \\ Sujeto\,a & \boldsymbol{x} \in S, \end{array} \tag{2.3}$$

donde f es una función continua en $\mathbb{R}^n$ y S es un conjunto de restricción en $\mathbb{R}^n$. La idea del método de penalización es remplazar el problema (2.3) por un problema sin restricciones de la forma

$$Minimizar\ f(\boldsymbol{x}) + cP(\boldsymbol{x}), \tag{2.4}$$

donde c es una constante y P es una función en $\mathbb{R}^n$ que satisface:

1. $P(\boldsymbol{x})$ es continua.
2. $P(\boldsymbol{x}) \geq 0$ para todo $\boldsymbol{x} \in \mathbb{R}^n$.
3. $P(\boldsymbol{x}) = 0$ si, y sólo si, $\boldsymbol{x} \in S$.

4. Diferentes funciones de penalización

La función de penalización puede representarse de la siguiente forma:

$$P(\boldsymbol{x}) = \sum_{i=1}^{m} h(r_i), \tag{2.5}$$

donde r_i son las componentes del vector $\boldsymbol{r} = A\boldsymbol{x} - \boldsymbol{b}$, con $A, \boldsymbol{b}$ definidas en (2.2) y h es una función real que puede ser una de las siguientes:

1. **Cuadrática**
$$h(r) = \frac{1}{2}r^2. \tag{2.6a}$$

2. **P-norma**
$$h(r) = \frac{1}{p} \mid r \mid^p . \tag{2.6b}$$

3. **Función de *Huber***
$$h(r) = \begin{cases} \dfrac{1}{2}r^2 & \text{si} \quad \mid r \mid \leq \beta. \\ \\ \beta \mid r \mid -\dfrac{\beta^2}{2} & \text{si} \quad \mid r \mid > \beta. \end{cases} \tag{2.6c}$$

4. **Logística**
$$h(r) = \beta^2 \ln\left(cosh(\frac{r}{\beta})\right) \quad \beta > 0. \tag{2.6d}$$

Si se escoje h como la función cuadrática, entonces la función (2.5) queda de la forma:

$$P(\boldsymbol{x}) = \sum_{i=1}^{n} \frac{1}{2}r_i^2 = \frac{1}{2} \parallel A\boldsymbol{x} - \boldsymbol{b} \parallel^2 . \tag{2.7}$$

Al reformular el problema de programación lineal, tenemos ahora un problema de optimización sin restricciones, para los cuales, podemos resol-

verlos usando técnicas iterativas que permiten encontrar el mínimo de la función objetivo del problema reformulado como son; Método de Newton, métodos cuasi-Newton y otros. En el presente trabajo se utiliza una técnica basada en redes neuronales artificiales, en el cual se propone un sistema dinámico que solucionara el problema de optimización, esto es lo que estudiaremos en el siguiente capítulo.

Capítulo 3

Redes neuronales

1. Redes neuronales artificiales

El comporta del ser humano despierta mucho interés en particular en la forma como estos pueden tomar decisiones ante problemas, ya sea en reconocimiento de imágenes, en la comunicación, en memorizar, etc. A partir de conocimientos previos se logra tomar decisiones y dar respuesta ante un estímulo, que a pesar de ser cosas aparentemente simples, le llevaría una gran cantidad de cálculos y tiempo a los ordenadores más avanzados. Esta es la razón por la que se ha creado una rama de las matemáticas computacionales denominada **redes neuronales artificiales** la cual tiene su fundamento en imitar el funcionamiento de las redes neuronales biológicas de la cual haremos una breve descripción.

Las redes neuronales biológicas tienen como elemento fundamental la *neurona*, esta es una célula que tiene una característica especial que la hace diferente a las demás se trata de la capacidad de comunicarse con otras células. La forma de esta célula está dada por un cuerpo celular o soma,

éste mide de entre 5 y 10 micras de diámetro esférico y tiene dos tipos de ramificaciones denominadas axón y dendritas.
La **Figura 3.1** ilustra las partes de una neurona y su conexión con otra neurona.

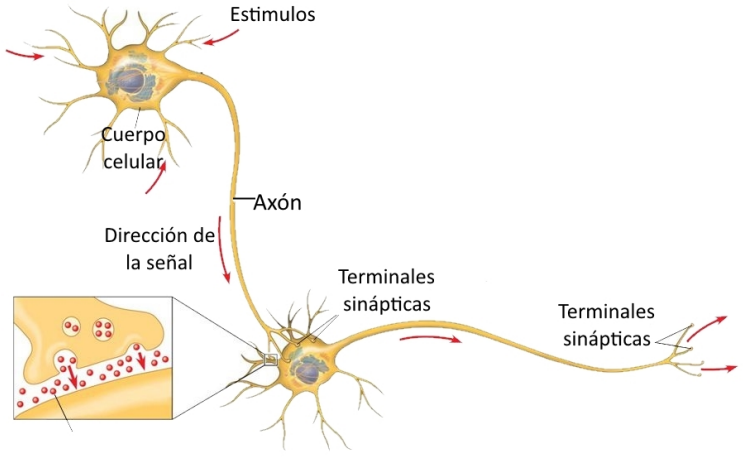

Figura 3.1: Neuronas biologicas [7]

Las neuronas se comunican por medio de impulsos eléctricos los cuales son recibidos por las dendritas, éstas son conducidas al núcleo de la célula, en donde se procesa la información hasta producir un impulso de salida que es trasmitido por el axón hacia otras neuronas. La conexión entre las dendritas de una neurona y el axón de otra se le denomina sinapsis, el cual es un espacio compuesto por una sustancia conductora que permite el paso de señales electro-químicas; esta sustancia es un líquido ionizado que permite o regula el paso de los impulsos eléctricos convirtiéndose así en inhibidores o potenciadores de las señales provenientes del axón, actuando como amplificadores o como aislantes, siendo las únicas que pueden ofrecer una resistencia, ya que el axón tiene unas propiedades que hacen que el impulso no pierda información. La propagación de estas señales por

todas las neuronas van modificando la concentración iónica de la sinapsis, ésto produce el aprendizaje, en el que la neurona funciona a saturación; es decir, si supera cierto umbral, la célula produce una respuesta, de lo contrario permanece inhibida.
Al parecer la neurona tiene un funcionamiento simple que podría ser emulado por un circuito electrónico simple, pero la verdadera potencia de la red neuronal radica en la masiva conexión entre las neuronas. Se estima que el cerebro humano tiene 100.000 millones de neuronas cada una conectada con cerca de 10.000 neuronas. Además esta red es robusta, en otras palabras, si se pierden algunas neuronas, ésta sigue funcionando normalmente sin problemas.
Basándose en la idea del funcionamiento de la red neuronal biológica se ha construido un modelo matemático, el cual tiene las características más importantes de la neurona biológica a saber: impulso eléctrico, la sinapsis, umbral de activación y conexión con otras neuronas. Este modelo se denomina red neuronal artificial (RNA).

El modelo asemeja los impulsos eléctricos con entradas numéricas, es decir, nuestra neurona artificial recibirá valores numéricos. Una vez definido ésto, se procede con la siguiente característica correspondiente a la sinapsis la cual, como se ha dicho, actúa como regulador de la entrada de impulsos al cuerpo celular; es decir, a la conexión que proviene de la neurona j-ésima y llega a la neurona i-ésima se le asocia un valor numérico que se denominará peso w_{ij} sináptico, el cual regulará las entradas numéricas a la neurona artificial; además, se tiene otra entrada adicional θ_i denominada umbral o bias la cual tiene asociado un peso de -1. Determinadas todas las entradas a la neurona, se procede a hacer la sumatoria de todas las entradas ponderadas por los pesos como se ve en la **Figura 3.2**. Para producir la salida de la neurona se hace uso de una función de $\mathbb{R}$ en $\mathbb{R}$. Dicha función de activación evalúa la suma ponderada, dando como resultado un valor en uno de los siguientes conjuntos:

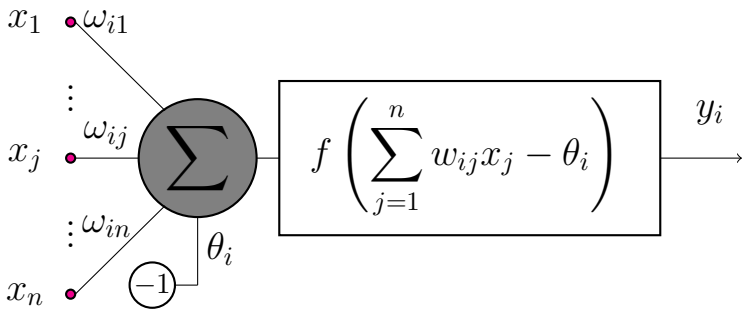

Figura 3.2: Neurona artificial

$[0, 1]$, $[-1, 1]$, los números reales o cualquier otro intervalo que se considere pertinente. También se puede escoger un recorrido discreto $\{1, 0\}$ o $\{-1, 1\}$; todo dependerá, como ya se ha dicho del modelo. Las funciones utilizadas en los diferentes problemas son por lo general las mostradas en la **Figura 3.3**.

Por último, el valor numérico de la función de activación se convierte en la nueva entrada de una rama sináptica de otra neurona, realizando la conexión entre ellas.

En la neurona biológica, las sinapsis van modificando las concentraciones iónicas; a esta propiedad se le puede denominar aprendizaje; dicho de otra manera, en base a eventos pasados la concentración iónica se ha ido modificado en toda la red biológica y estas experiencias previas le permiten resolver o tomar decisiones. Este es la más importante característica que resaltamos en la neurona artificial ya que el "aprendizaje" o "entrenamiento" se realiza cuando se modifican los pesos w_{ij} a partir de una muestra o datos previos.

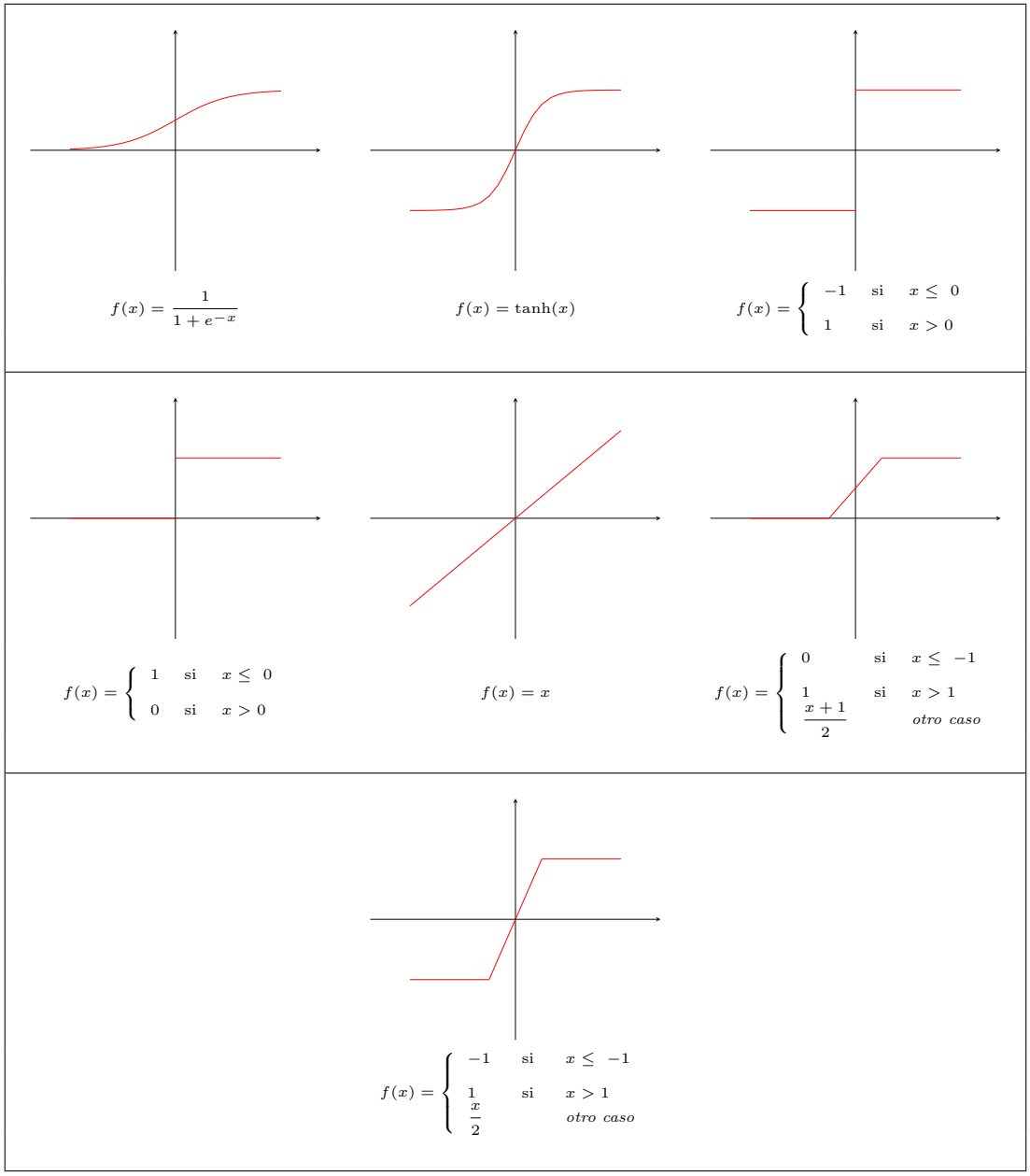

Figura 3.3: Funciones de activación más utilizadas

2. Aprendizaje de las redes neuronales artificiales

Como se indicó en la sección anterior, el aprendizaje de una red neuronal artificial consiste en la modificación de los pesos de las conexiones, la forma de como realizamos el ajuste de los pesos se denomina esquema de aprendizaje, siendo este una parte fundamental de la red ya que este nos dice que tipo de problemas se puede resolver.

Las redes neuronales artificiales son sistemas de aprendizaje basados en muestras [6] esto quiere decir que estas muestras deben ser significativas y representativas por lo que debemos de ser capaces de escoger muestras que representen toda la población o el conjunto de estudio del que vamos a sacar el problema, además, tenemos que estar seguros que el número de elementos de la muestra sea suficiente.

Describiremos a continuación algunos esquemas de aprendizaje.

2.1. Redes neuronales con aprendizaje supervisado

Las redes neuronales artificiales que siguen este esquema necesitan de un supervisor para hacer la modificación de los pesos y además que las muestras seleccionadas para el aprendizaje ya tengan respuesta, dicho de otra manera, si el problema que tenemos consiste en el reconocimiento de imágenes debemos saber cuál es esa imagen y a partir de la introducción de esa muestra en la red debemos comparar la respuesta de la red con la salida real, de aquí el nombre aprendizaje supervisado, si la respuesta de la red difiere mucho de la salida real se hará mayor modificación en los pesos y si la respuesta es muy cercana la modificación será menor.

2.2. Redes neuronales con aprendizaje no supervisado

Este tipo de aprendizaje consiste en la modificación de los pesos por medio de un mecanismo interno. A partir de un ejemplo, la red es capaz de extraer rasgos o características para así dar respuesta conocida, es decir, que no todas las arquitecturas de red neuronal artificial necesitan muestras para el aprendizaje, sino que estas a partir del esquema de aprendizaje van evolucionando o acomodando los pesos de una forma automática, ya sea por un número fijo de repeticiones o en el caso que se pueda definir una función error la cual nos dice que si su valor está por debajo de cierta tolerancia este se detiene y en ese caso el aprendizaje estará completado. Otra forma que puede parar el aprendizaje es cuando ya no se produzcan modificaciones relevantes en los pesos.

3. Arquitectura de una red

El aprendizaje de la RNA depende de la arquitectura de la red, esta tiene que ver en cuanto a cómo se realiza la conexión de las neuronas, pues bien, la RNA tiene por lo general una capa de entrada encargada de recibir las señales ya sean ingresadas de una muestra o de sensores, etc. Unas capas intermedias son las encargadas de hacer el procesamiento de la información recibida, que funcionan de manera independiente al exterior de la red y una capa de salida encargada de enviar la información al exterior de la RNA. El número de capas puede ir desde uno hasta varios miles, todo depende del problema al que se enfrenta la red y la forma como se hace la conexión de estas capas dan como resultado las siguientes arquitecturas:

Feedforward: Esta se refiere a una conexiona hacia adelante, es decir, que cada neurona de una capa está conectada solamente con las neuronas

de la capa siguiente, esta arquitectura se muestra en la **Figura 3.4**.

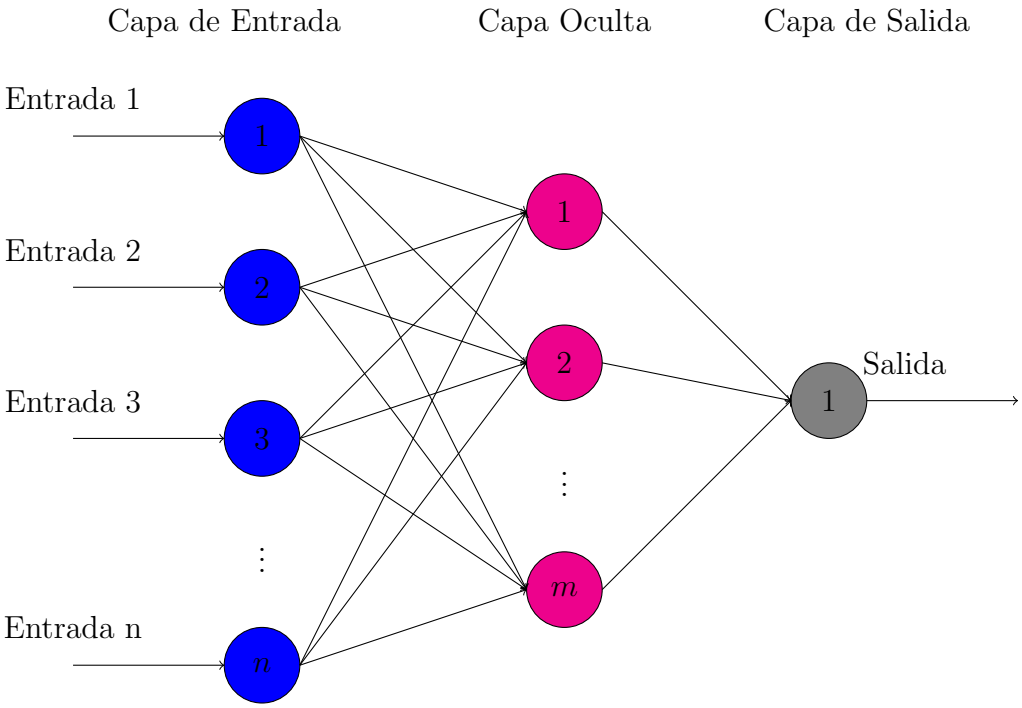

Figura 3.4: Red neuronal con conexión *Feedforward*

Feedback: En este caso la conexión se da entre neuronas de una capa anterior e inclusive permite conexión con neuronas de la misma capa, en la **Figura 3.5** se muestra esta arquitectura

Las redes neuronales recurrentes (RNR) se ubican en la arquitectura *Feedback* y es además una red no supervisada, esta se caracteriza por crear bucles en las capas y neuronas. Estas conexiones recurrentes hacen que el estado de una neurona dependa del estado de las neuronas anteriores, es por esto que se hace necesario considerar una variable temporal y por ende

esta red presenta un comportamiento dinámico en el cual el cambio de las activaciones de la red alcanzan un punto estable, es decir, las activaciones van de un estado inicial a un estado en el cual las neuronas de la red no se modifiquen. Esta red tiene asociada una función que representa la energía del sistema, a esta se le conoce como *función de energía computacional*.

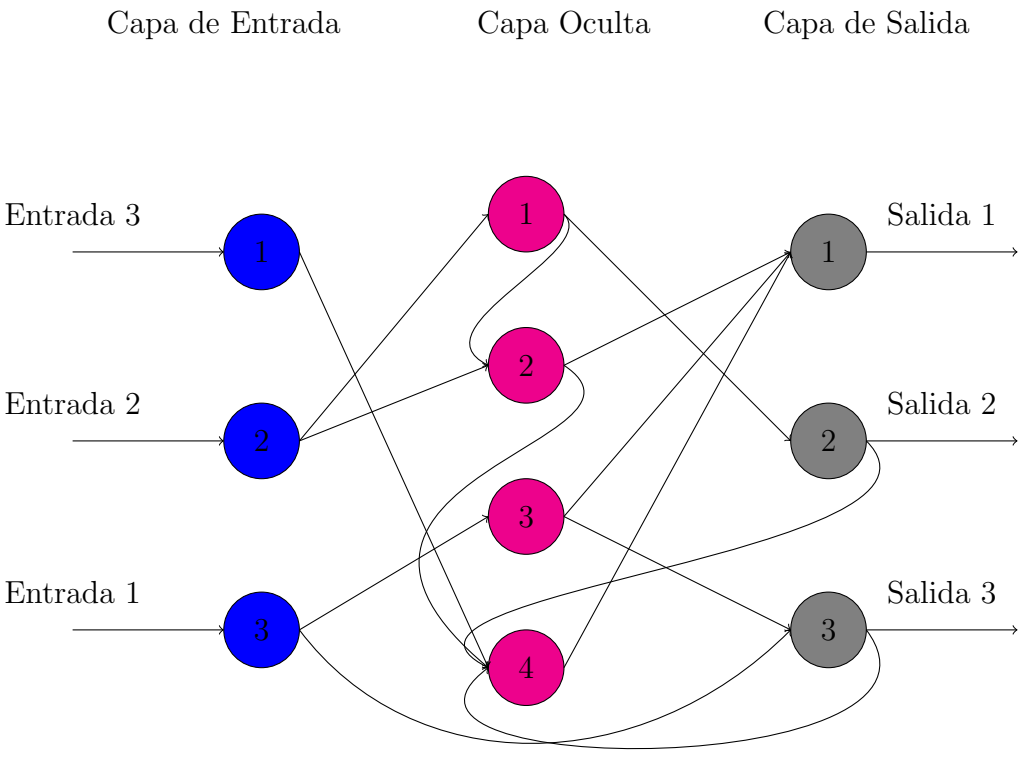

Figura 3.5: Red neuronal con conexión *Feedback*

La estabilidad de esta red es representada por el mínimo de esta función de energía computacional como se vera en capítulos siguientes. La forma de conexión hace que estas redes se puedan convertir en excesivamente lentas para alcanzar su estado de equilibrio como sistema dinámico, salvo que se utilice electrónica para su implementación.

3.1. Redes neuronales recurrentes para optimización

En este documento resolveremos el problema de optimización de una función lineal con restricciones lineales, mediante una red neuronal recurrente, en particular usando una red conocida como red de Hopfield, por lo cual, describiremos cómo reformular un problema de optimización y en particular un problema de programación lineal como un sistema dinámico.

Una representación formal de una red neuronal recurrente para optimización puede ser descrita como sigue[12]:

$$v(t) = E(\boldsymbol{x}(t), \lambda(t)) \tag{3.1}$$
$$\boldsymbol{x}(t) = F(\boldsymbol{u}(t)) \tag{3.2}$$
$$\boldsymbol{u}(t) = G(t, \boldsymbol{x}(t), \lambda(t)) \tag{3.3}$$
$$\lambda(t) = H(t, \lambda(t)) \tag{3.4}$$

Donde $v \in \mathbb{R}$ es el índice de desempeño de la RNR, $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{u} \in \mathbb{R}^n$ corresponde a las entradas de agregación de las neuronas; $\lambda \in \mathbb{R}$ es el parámetro o variable de penalización; E, F, G y H hacen referencia a la regla de evaluación, regla de activación, regla de agregación y regla de penalización, respectivamente. Las condiciones iniciales $\boldsymbol{u}(0), \lambda(0)$ y las reglas E, F, G, H, definen el comportamiento dinámico de la red.

4. Red de *Hopfield*

El modelo de *Hopfield* es utilizado para resolver problemas de optimización, el cual consiste en una red mono capa en la que cada neurona está conectada con todas las demás pero no consigo misma (ver **Figura 3.6**). Además, los pesos asociados a las conexiones entre dos neuronas son simétricos; es decir que el peso de la conexión de una neurona i con la

neurona j es igual a peso de la conexión de la neurona j con la i.

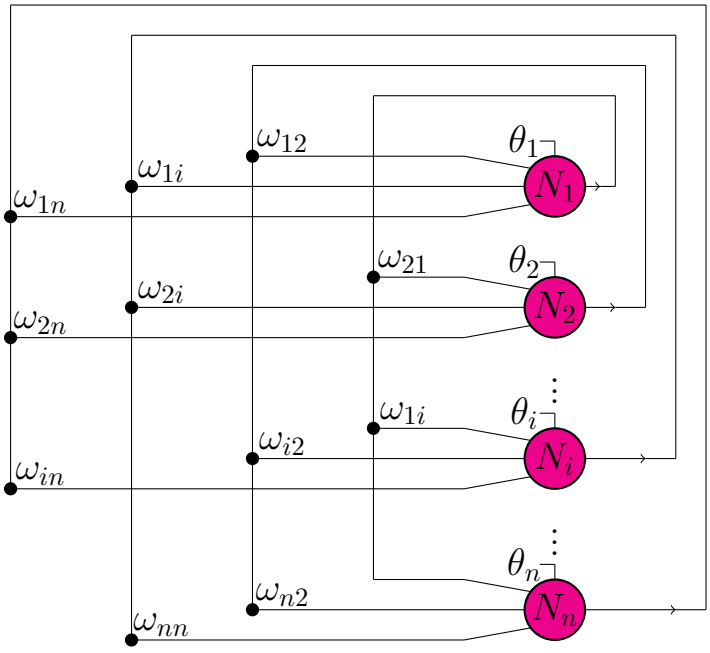

Figura 3.6: Red de *Hopfield*

Como se ha dicho antes, la propuesta de la red para optimización, está dada por las ecuaciones (3.1), (3.2), (3.3), (3.4). Para el caso de la red de *Hopfield*, la función de energía (*Evaluación*) propuesta tiene la siguiente forma:

$$E(\boldsymbol{x}(t)) = -\sum_{i=1}^{n}\sum_{j\neq i}\frac{1}{2}\omega_{ij}x_i(t)x_j(t) - \sum_{i=1}^{n}\theta_i x_i(t) + \sum_{i=1}^{n}\int_0^{x_i} F_i^{-1}(s_i(t))\,ds_i. \tag{3.5}$$

Donde
x_i: Es el valor de la salida de la neurona i.
ω_{ij}: Son los pesos sinápticos que representan la conexión que va de la neurona j a la neurona i

θ_i: Entrada adicional de la neurona i.
F_i^{-1} : Es la inversa de la función de activación de la neurona i.

En general, la función (3.5) no es de *Liapunov*.Para que lo sea, la función de activación debe ser no decreciente con respecto al tiempo. Esta función sirve como medida de abstracción de la energía del sistema dinámico. Además es una función acotada por inferiormente y monótona decreciente con respecto al tiempo.
La regla de activación utiliza una función sigmoidea de la forma:

$$F_i(u_i(t)) = \frac{1}{1 + e^{-u_i(t)/T_i}}.$$

Donde T_i es un escalar positivo. La agregación está dada mediante un sistema de ecuaciones diferenciales, que para la i-ésima neurona es:

$$\begin{aligned} C_i u_i'(t) &= -\nabla_{x_i} E(\boldsymbol{x}(t)) \\ &= \sum_{j=1}^{n} \omega_{ij} x_j(t) - u_i(t)/R_i + \theta_i \end{aligned}$$

Donde C_i y R_i son parámetros de capacitancia y resistividad para la neurona i. en el contexto de optimización, *Hopfield* utiliza el parámetro de penalización constante, es decir $H \equiv constante$.

La red de*Hopfield* está diseñada para obtener minimos locales de la función de energía; para lo cual, él probó que dicha red es asintóticamente estable, si la función de activación es continua y monótona decreciente, además, que las neuronas no tengan auto conexión y que los pesos sean simétricos [12]. También se ha demostrado que, si se tiene una RNR con parámetro de penalización monótonamente creciente con respecto al tiempo entonces la red es asintóticamente estable [12].

Capítulo 4

Teoría de convergencia

En este capítulo, veremos cómo se reformula un problema de programación lineal en un sistema dinámico, mediante redes neuronales recurrentes. También estudiaremos el comportamiento de la trayectoria solución obtenida, utilizando este método.

1. Red neuronal para solucionar el problema de PL

En el articulo [12], *Jun Wang* y *Vira Chankong*, proponen una red neuronal artificial recurrente que utiliza los mismos principios de la red de *Hopfield* en el cual, como se ha estudiado antes, está definido por las expresiones (3.1), (3.3), (3.2), (3.4).

Como función de evaluación, se escoge la función basada en el método de penalización.

$$E(\boldsymbol{x}(t), \lambda(t)) = f(\boldsymbol{x}(t)) + \lambda(t)P(\boldsymbol{x}(t)). \tag{4.1}$$

x_i	$(-\infty, \infty)$	$(-\infty, 0]$	$[-M_i, 0]$	$[-M_i, M_i]$	$[0, \infty)$	$[0, M_i]$
$F_i(u_i)$	$\eta_i u_i$	$-e^{-\eta_i u_i}$	$-\dfrac{M_i e^{-\eta_i u_i}}{1+e^{-\eta_i u_i}}$	$\dfrac{M_i(1-e^{-\eta_i u_i})}{1+e^{-\eta_i u_i}}$	$e^{\eta_i u_i}$	$\dfrac{M_i}{1+e^{-\eta_i u_i}}$

Tabla 4.1: Funciones de activación

La función de agregación se define en términos del gradiente respecto $\boldsymbol{x}$ de la función de evaluación, es decir,

$$\begin{aligned} \boldsymbol{u}'(t) &= -\left[\nabla_{\boldsymbol{x}} E(\boldsymbol{x}(t), \lambda(t))\right] P(\boldsymbol{x}(t)). \\ &= -\left[\boldsymbol{c} + \lambda(t)(A^T A\boldsymbol{x}(t) - A^T \boldsymbol{b})\right] P(\boldsymbol{x}(t)). \end{aligned} \tag{4.2}$$

La configuración de la red en cuanto a la función de activación se escoge de la ***Tabla 4.1*** dependiendo de cómo son las cotas en el problema de PL, es decir, depende de $\boldsymbol{x}_\alpha$ y $\boldsymbol{x}_\beta$.

F_i representa la i-ésima componente de la función F para la neurona i-ésima, M_i, η_i son constantes para todo i.

La regla de penalización se elige una de las siguientes funciones:

$$\lambda(t) = \frac{1}{C_\lambda} \int_0^t \frac{\mu}{P(\boldsymbol{x}(\tau))}\, d\tau, \tag{4.3}$$

$$\lambda(t) = \frac{1}{C_\lambda} \int_0^t \mu P(\boldsymbol{x}(\tau))\, d\tau, \tag{4.4}$$

$$\lambda(t) = \frac{1}{C_\lambda} \alpha^{\mu t}, \tag{4.5}$$

donde $C_\lambda \in \mathbb{R}, \alpha > 1$ y $\mu > 0$.

Es conveniente calcular las derivadas de estas funciones para que la construcción del sistema dinámico no tenga involucrada la integral.

$$C_\lambda \lambda'(t) = \frac{\mu}{P(\boldsymbol{x}(t))} \tag{4.6}$$

$$C_\lambda \lambda'(t) = \mu P(\boldsymbol{x}(t)) \tag{4.7}$$

De (4.2) y (4.6)

$$\begin{aligned} v(t) &= E(\boldsymbol{x}(t), \lambda(t)) = \boldsymbol{c}^T \boldsymbol{x}(t) + \lambda(t) P(\boldsymbol{x}(t)) \\ C_u \boldsymbol{u}'(t) &= -\frac{\boldsymbol{c} + \lambda(t)(A^T A \boldsymbol{x}(t) - A^T \boldsymbol{b})}{C_\lambda \lambda'(t)} \\ C_\lambda \lambda'(t) &= \frac{2}{\boldsymbol{x}^T(t) A^T A \boldsymbol{x}(t) - 2\boldsymbol{x}^T(t) A^T \boldsymbol{b} + \boldsymbol{b}^T \boldsymbol{b}} \\ \boldsymbol{x}(t) &= F(\boldsymbol{u}(t)), \end{aligned}$$

donde $C_u << C_\lambda$ son escalares que se escogen suficientemente pequeños.

Este sistema dinámico fue propuesto en [12] para la solución del problema de programación lineal en donde se utilizó como función de penalización la ecuación (2.5) a continuación estudiará en detalle la convergencia del método.

2. Convergencia del método

En esta sección comenzaremos demostrando que la RNR es asintóticamente estable[13],[11].

Teorema 2.1. *Sea E y G la regla de evaluación y agregación dadas por*

las ecuaciones:

$$E(\boldsymbol{x}(t), \lambda(t)) = f(\boldsymbol{x}(t)) + \lambda(t)P(\boldsymbol{x}(t)) \tag{4.8}$$

$$\boldsymbol{u}'(t) = G(\boldsymbol{x}(t), \lambda(t)) = -[\nabla_{\boldsymbol{x}}^T f(\boldsymbol{x}(t)) + \lambda(t)\nabla_{\boldsymbol{x}}^T P(\boldsymbol{x}(t))]q(\boldsymbol{x}(t)) \tag{4.9}$$

Donde $f(\boldsymbol{x})$ y $P(\boldsymbol{x})$ son la función objetivo y la función de penalización respectivamente, $q(\boldsymbol{x}(t))$ es una función arbitraria no negativa es decir, para todo $t \geq 0$ y para todo $\boldsymbol{x} \in S$, $q(\boldsymbol{x}(t)) \geq 0$.

Supóngase que:

1. *Para todo $t \geq 0$ la función de activación $F(\boldsymbol{u})$ es diferenciable y monótonamente creciente respecto a cada entrada finita u_k, es decir,*

$$\frac{d}{du_k} F_k(u_k) > 0$$

   para $-\infty < u_k(t) < +\infty$, $k = 1, 2, ..., n$.

2. *Para todo $t \geq 0, \lambda(t) > 0$.*

3. *$\lambda'(t) \geq 0$ o $\lambda'(t) \leq 0$.*

Entonces un punto crítico de la RNR es asintóticamente estable, es decir,

$$\lim_{t \to \infty} \boldsymbol{x}'(t) = \boldsymbol{0}$$

Demostración. En caso de $q(\boldsymbol{x}(t)) = 0$ tenemos que $\boldsymbol{u}'(t) = \boldsymbol{0}$ y $\boldsymbol{x}'(t) = \boldsymbol{0}$, por lo tanto tenemos que la red es estable en sí, es decir, la red es asintóticamente estable.
Por esto se considerará el caso $q(\boldsymbol{x}(t)) > 0$, $\boldsymbol{u}'(t) \neq \boldsymbol{0}$ y $\boldsymbol{x}'(t) \neq \boldsymbol{0}$
Si $\lambda'(t) \geq 0$
Definamos la función L de la siguiente forma,

$$L(\boldsymbol{x}(t), \lambda(t)) = \lambda^{-1}(t) f(\boldsymbol{x}(t)) + P(\boldsymbol{x}(t))$$

Por otro lado observe que la función f es continua en el conjunto S y este es acotado, entonces podemos agregar una constante adecuada tal que $\tilde{f} = f + R \geq 0$, así $\tilde{f}(\boldsymbol{x}(t)) \geq 0$ para todo $\boldsymbol{x} \in S$, por tanto podemos considerar que $f(\boldsymbol{x}(t)) \geq 0$. Dado que para todo $t \geq 0, \lambda(t) > 0$ y para todo $\boldsymbol{x} \in S$, $P(\boldsymbol{x}(t)) \geq 0$. Entonces para todo $t \geq 0$ y para todo $\boldsymbol{x}(t)$ tenemos que $L(\boldsymbol{x}(t), \lambda(t)) \geq 0$.

Ahora observemos que:

$$\begin{aligned}
\frac{dL}{dt} &= -\lambda^{-2}(t)\lambda'(t)f(\boldsymbol{x}(t)) + \lambda^{-1}(t)\nabla_{\boldsymbol{x}} f(\boldsymbol{x}(t))\boldsymbol{x}'(t) + \nabla_{\boldsymbol{x}} P(\boldsymbol{x}(t))\boldsymbol{x}'(t) \\
&= -\lambda^{-2}(t)\lambda'(t)f(\boldsymbol{x}(t)) + \lambda^{-1}(t)[\nabla_{\boldsymbol{x}} f(\boldsymbol{x}(t)) + \lambda(t)\nabla_{\boldsymbol{x}} P(\boldsymbol{x}(t))]\boldsymbol{x}'(t) \\
&= -\lambda^{-2}(t)\lambda'(t)f(\boldsymbol{x}(t)) + \lambda^{-1}(t)\left[-\frac{\boldsymbol{u}'(t)}{q(\boldsymbol{x}(t))}\right]\boldsymbol{x}'(t) \\
&= -\lambda^{-2}(t)\lambda'(t)f(\boldsymbol{x}(t)) - \lambda^{-1}(t)\boldsymbol{u}'(t)^T\boldsymbol{x}'(t)/q(\boldsymbol{x}(t))
\end{aligned}$$

Sabemos que $x_k(t) = F_k(u_k(t))$ y esta es monótonamente creciente para $k = 1, ..., n$, además

$$\frac{d}{dt}x_k(t) = \frac{d}{du_k}F_k(u_k(t))\frac{d}{dt}u_k(t)$$

que notado de otra forma queda:

$$x_k'(t) = F_k'(u_k(t))u_k'(t)$$

multiplicando por $u_k'(t)$ tenemos,

$$u_k'(t)x_k'(t) = F_k'(u_k(t))(u_k'(t))^2 \geq 0$$

ahora debe existir un k tal que $u_k'(t) \neq \mathbf{0}$ ya que $\boldsymbol{u}'(t) \neq \mathbf{0}$, asi

$$\boldsymbol{u}'(t)^T\boldsymbol{x}'(t) > 0$$

por tanto, podemos concluir que $dL(\boldsymbol{x}(t), \lambda(t))/dt < 0$ lo que esto quiere decir es que $L(\boldsymbol{x}(t), \lambda(t))$ es una función, estricta de *Liapunov* y por consiguiente, si existe un punto crítico del sistema, éste será asintóticamente estable.
En el caso $\lambda'(t) \leq 0$ tenemos que,

$$\begin{aligned} \frac{dE}{dt} &= [\nabla_{\boldsymbol{x}} f(\boldsymbol{x}(t)) + \lambda(t)\nabla_{\boldsymbol{x}} P(\boldsymbol{x}(t))]\boldsymbol{x}'(t) + \lambda'(t)P(\boldsymbol{x}(t)) \\ &= -\boldsymbol{u}'(t)^T \boldsymbol{x}'(t)/q(\boldsymbol{x}(t)) + \lambda'(t)P(\boldsymbol{x}(t)) \end{aligned}$$

Dado que $E(\boldsymbol{x}(t), \lambda(t)) \geq 0, \quad \lambda'(t) \leq 0, \quad P(\boldsymbol{x}(t)) \geq 0, \quad q(\boldsymbol{x}(t)) > 0$ y $\boldsymbol{u}'(t)^T \boldsymbol{x}'(t) > 0$, concluimos que,

$$\frac{d}{dt} E(\boldsymbol{x}(t), \lambda(t)) < 0$$

Así, $E(\boldsymbol{x}(t), \lambda(t))$ es una función estricta de *Liapunov*, lo que completa la prueba.

★

El siguente teorema garantiza que el punto asintóticamente estable es una solución factible.

Teorema 2.2. *Sean la función de penalización* $P(\boldsymbol{x}) = \frac{1}{2} \parallel A\boldsymbol{x} - \boldsymbol{b} \parallel^2$ *y sea* E *y* G *la regla de evaluación y regla de agregación dadas respectivamente, por las ecuaciones:*

$$E(\boldsymbol{x}(t), \lambda(t)) = f(\boldsymbol{x}(t)) + \lambda(t)P(\boldsymbol{x}(t)) \tag{4.10}$$

$$\begin{aligned} \boldsymbol{u}(t) &= G(\boldsymbol{x}(t), \lambda(t)) \\ &= -\frac{1}{C_u} \int_0^t \big[\boldsymbol{c} + \lambda(\tau)(A^T A\boldsymbol{x}(\tau) - A^T \boldsymbol{b})\big] P(\boldsymbol{x}(\tau))\, d\tau \end{aligned} \tag{4.11}$$

Donde C_u *es un parámetro capacitivo.*
Supóngase

1. *La función de activación es diferenciable a trozos y monótonamente creciente.*
2. *Para todo* $t \geq 0$, $\lambda(t) > 0$, $\lambda(t)$ *es diferenciable a trozos.*
3. $P(\boldsymbol{x}(t)) > 0$ *implica que* $\lambda'(t) \geq 0$.

Entonces un punto estable de la RNR representa una solución factible para el problema de PL.

Demostración. Sea $\boldsymbol{x}^*$ un punto crítico de la RNR, de acuerdo al teorema (2.1) dadas las condiciones anteriores el estado de la RNR es asintóticamente estable, es decir,

$$\lim_{t\to\infty} \boldsymbol{x}(t) = \boldsymbol{x}^* \quad \text{o} \quad \lim_{t\to\infty} \boldsymbol{x}'(t) = \mathbf{0}$$

Observe que de la hipótesis 1:

$$\lim_{t\to\infty} \boldsymbol{x}'(t) = \mathbf{0} \text{ es equivalente a } \lim_{t\to\infty} \boldsymbol{u}'(t) = \mathbf{0}$$

De la ecuación (4.11) obtenemos que:

$$C_u \boldsymbol{u}'(t) = -[\boldsymbol{c} + \lambda(t)(A^T A\boldsymbol{x} - A^T \boldsymbol{b})]P(\boldsymbol{x}(t))$$

de donde,

$$\lim_{t\to\infty} [\boldsymbol{c} + \lambda(t)(A^T A\boldsymbol{x} - A^T \boldsymbol{b})]P(\boldsymbol{x}(t)) = 0.$$

Luego

$$\lim_{t\to\infty} \boldsymbol{c} + \lambda(t)(A^T A\boldsymbol{x} - A^T \boldsymbol{b}) = 0 \quad \text{o} \quad \lim_{t\to\infty} P(\boldsymbol{x}(t)) = 0$$

Como $\boldsymbol{c} \neq 0$, para todo $\boldsymbol{x} \in X$, existe $\lambda(t) > 0$ tal que:

$$\lim_{t\to\infty} \boldsymbol{c} + \lambda(t)(A^T A\boldsymbol{x} - A^T \boldsymbol{b}) \neq 0$$

En otras palabras, si suponemos que

$$\lim_{t\to\infty} P(\boldsymbol{x}(t)) \neq 0$$

entonces

$$\lim_{t\to\infty} \boldsymbol{u}'(t) \neq \mathbf{0} \text{ y por consiguiente, } \lim_{t\to\infty} \boldsymbol{x}'(t) \neq \mathbf{0}$$

por tanto, de

$$\lim_{t\to\infty} \boldsymbol{u}'(t) = \mathbf{0}, \lambda(t) \geq 0 \text{ y } \lambda'(t) \neq 0$$

concluimos que

$$\lim_{t\to\infty} P(\boldsymbol{x}(t)) = 0$$

así,

$$P(\boldsymbol{x}^*) = 0 \Leftrightarrow A\boldsymbol{x}^* = \boldsymbol{b}$$

que es lo mismo decir que $\boldsymbol{x}^* \in \widehat{X}$. ★

Por último, tenemos el teorema que garantiza la solución óptima del problema de programación lineal.

Teorema 2.3. *Sea la función de penalización* $P(\boldsymbol{x}) = \frac{1}{2} \parallel A\boldsymbol{x} - \boldsymbol{b} \parallel^2$ *y sean la regla de evaluación y la regla de agregación dadas por (4.10) y (4.11) respectivamente.*
Supóngamos que:

1. *El estado inicial es un punto no factible.*
2. *La función de activación es diferenciable a trozos y monótonamente creciente.*
3. *Para todo* $t \geq 0$, $\lambda(t) > 0$, $\lambda(t)$ *es diferenciable a trozos.*
4. $\boldsymbol{x}'(t) \neq 0$ *implica* $\lambda'(t) = 0$, *y* $\boldsymbol{x}'(t) = \mathbf{0}$ *y* $P(\boldsymbol{x}(t)) > 0$ *implica que* $\lambda'(t) > 0$.

Entonces el punto estacionario de la RNR representa la solución óptima del problema de PL.

Demostración. Sea $\widehat{\boldsymbol{x}}$ una solución factible optima, es decir,

$$\boldsymbol{c}^T\widehat{\boldsymbol{x}} = \min_{\boldsymbol{\chi}\in\widehat{X}} \boldsymbol{c}^T\boldsymbol{\chi}.$$

y sea $\boldsymbol{x}^*$ un punto estable de la RNR. Debemos probar que $\boldsymbol{c}^T\boldsymbol{x}^* = \boldsymbol{c}^T\widehat{\boldsymbol{x}}$. Si la condición inicial $\boldsymbol{x}(0)$ inicia en una región no factible, es decir, $A\boldsymbol{x}(0) \neq \boldsymbol{b}$ implica que $\boldsymbol{u}'(t) \neq 0$, para $t \geq 0$ y de acuerdo a la ecuación (4.11) ésto implica que el estado de transición siempre será activado. Aplicando el teorema (2.2), tenemos que el punto estable de la RNR representa una solución factible para el problema de PL, es decir,

$$\lim_{t\to\infty} \boldsymbol{x}(t) = \boldsymbol{x}^* \in \widehat{X} \text{ o } \lim_{t\to\infty} P(\boldsymbol{x}(t)) = 0$$

así,

$$\begin{aligned}\lim_{t\to\infty} E(\boldsymbol{x}(t), \lambda(t)) &= \lim_{t\to\infty} \boldsymbol{c}^T\boldsymbol{x}(t) + \lambda(t)P(\boldsymbol{x}(t)) \\ &= \boldsymbol{c}^T\boldsymbol{x}^* + \lim_{t\to\infty} \lambda(t)P(\boldsymbol{x}(t)) \\ &= \boldsymbol{c}^T\boldsymbol{x}^*\end{aligned}$$

Ahora

$$\boldsymbol{x}'(t) = -D(t)\boldsymbol{u}'(t) = D(t)\nabla_x^T E(\boldsymbol{x}(t), \lambda(t))P(\boldsymbol{x}(t)),$$

donde

$$D(t) = \begin{pmatrix} \dfrac{dF_1(u_1)}{du_1} & 0 & \cdots & 0 \\ 0 & \dfrac{dF_2(u_2)}{du_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \dfrac{dF_n(u_n)}{du_n} \end{pmatrix}$$

El estado de la RNR se aproxima a un mínimo local de la función de evaluación en una forma que depende del gradiente cuando t tiende a infinito y, como la función objetivo $f(\boldsymbol{x}) = \boldsymbol{c}^T\boldsymbol{x}$ es lineal y la función de penalización $P(\boldsymbol{x})$ es una forma cuadrática con matriz hessiana A^TA semi-definida positiva, entonces $f(\boldsymbol{x})$ y $P(\boldsymbol{x})$ son convexas.
Por ende la función $E(\boldsymbol{x}(t), \lambda(t))$ es también convexa respecto a $\boldsymbol{x}$; ésto significa, que el mínimo local $\boldsymbol{x}^*$ es un mínimo global de $E(\boldsymbol{x}, \lambda)$ en X.
Por lo tanto, para todo $t \geq 0$ y para todo $\boldsymbol{x} \in X$

$$\boldsymbol{c}^T\boldsymbol{x}^* = \lim_{t\to\infty}[\boldsymbol{c}^T\boldsymbol{x} + \lambda(t)P(\boldsymbol{x}(t))] \leq \boldsymbol{c}^T\boldsymbol{x}(t) + \lambda(t)P(\boldsymbol{x}(t)).$$

como $\widehat{\boldsymbol{x}} \in \widehat{X} \subseteq X$, entonces $P(\boldsymbol{x}(t)) = 0$, luego

$$\boldsymbol{c}^T\boldsymbol{x}^* \leq \boldsymbol{c}^T\widehat{\boldsymbol{x}} + \lambda(t)P(\widehat{\boldsymbol{x}}) = \boldsymbol{c}^T\widehat{\boldsymbol{x}}$$

Por otro lado, $\boldsymbol{x}^* \in \widehat{X}$ y $\widehat{\boldsymbol{x}}$ minimiza a $\boldsymbol{c}^T\boldsymbol{x}$ en $\widehat{X}$, luego $\boldsymbol{c}^T\widehat{\boldsymbol{x}} \leq \boldsymbol{c}^T\boldsymbol{x}^*$ y por tanto,

$$\boldsymbol{c}^T\widehat{\boldsymbol{x}} = \boldsymbol{c}^T\boldsymbol{x}^* = \min_{\boldsymbol{\chi}\in\widehat{X}} \boldsymbol{c}^T\boldsymbol{\chi}.$$

★

Capítulo 5

Pruebas numéricas

En este capítulo, analizamos los resultados numéricos del método de redes neuronales recurrentes para solucionar problemas de optimización estudiados en el **capítulo 4**. Para ello se muestran una serie de ejemplos en los que se hace toda la construcción del sistema dinámico y luego se utiliza el método de *Runge Kutta* de orden 4 [4] para obtener los resultados.

El código del método fue desarrollado en el *software* **MATLAB® R2014a**. La experimentación numérica se realizo utilizando un computador con procesador INTEL CELERON CPU 560 @2.13 GHz y 2 GB de RAM.

Utilizamos cinco problemas que se proponen en diferentes artículos [12], [1] y en el libro [10] para las pruebas. Debemos hacer una aclaración referente a las pruebas numéricas, la variable t se refiere al tiempo de convergencia del método a diferencia de la variable T, que se mide el tiempo que tarda ejecutándose el método en solucionar el problema y esta dado en segundos.

Problema 5.1.

$$\begin{array}{rc} Minimizar & z = -2x_1 - 3.5x_2 \\ Sujeto\, a & -x_1 + 4x_2 \leq 1 \\ & 2x_1 + 3x_2 \leq 3.5 \\ & 2x_1 + x_2 \leq 3 \\ & x_1 \geq 0,\ x_2 \geq 0 \end{array} \tag{5.1}$$

La solución del problema utilizando el método simplex es $\boldsymbol{x}^{*T} = (1\ \ 0.5)$. *Ahora utilizando el algoritmo de RNR el primer paso es la conversión de las restricciones de desigualdad a igualdad*

$$\begin{array}{rcl} -x_1 + 4x_2 + x_3 & = & 1 \\ 2x_1 + 3x_2 + x_4 & = & 3.5 \\ 2x_1 + x_2 + x_5 & = & 3 \end{array}$$

$x_3 \geq 0, x_4 \geq 0, x_5 \geq 0$ *luego la matriz y los vectores* A, b, c *son:*

$$A = \begin{pmatrix} -1 & 4 & 1 & 0 & 0 \\ 2 & 3 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 & 1 \end{pmatrix}$$

$$b = \begin{pmatrix} 1 \\ 3.5 \\ 3 \end{pmatrix}$$

$$c = \begin{pmatrix} -2 \\ -3.5 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Se puede deducir un rango de acotación de las variables por ejemplo $0 \leq x_1 \leq 2,\ 0 \leq x_2 \leq 3,\ 0 \leq x_3, x_4, x_5 \leq 1$ *y según la* ***Tabla 4.1*** *las*

funciones de activación que utilizamos son:

$$\begin{aligned} x_1(t) &= 2(1+e^{-u_1(t)})^{-1} \\ x_2(t) &= 3(1+e^{-u_2(t)})^{-1} \\ x_i(t) &= (1+e^{-u_i(t)})^{-1} \text{ para } i=3,4,5. \end{aligned}$$

Las constantes que aparecen en las ecuaciones del sistema dinámico se toman $c_\lambda = 1$, $c_u = 0.01, \eta = 1, \mu = 2$ *por lo que el sistema dinámico queda así:*

$$\begin{aligned} \boldsymbol{u}'(t) &= -100\frac{c+\lambda(t)(A^TA\boldsymbol{x}(t)-A^T\boldsymbol{b}^T)}{\lambda'(t)} \\ \lambda'(t) &= \frac{2}{\boldsymbol{x}^T(t)A^TA\boldsymbol{x}(t)-2\boldsymbol{x}^T(t)A^T\boldsymbol{b}+\boldsymbol{b}^T\boldsymbol{b}} \\ x_1(t) &= 2(1+e^{-u_1(t)})^{-1} \\ x_2(t) &= 3(1+e^{-u_2(t)})^{-1} \\ x_i(t) &= (1+e^{-u_i(t)})^{-1} \text{ para } i=3,4,5. \end{aligned}$$

Tomando como condiciones iniciales
$\boldsymbol{x}(0)^T = \begin{pmatrix} 1.5 & 1.5 & 0.84 & 0.25 & 0.81 \end{pmatrix}$, $\lambda(0) = 0.2$
Se obtuvo la solución

$$\boldsymbol{x}^* = \begin{pmatrix} 1.000123686661055 \\ 0.500001962378657 \\ 0.0 \\ 0.0 \\ 0.499456693349832 \end{pmatrix}$$

$f(\boldsymbol{x}^*) = -3.750001504383$, $P(\boldsymbol{x}^*) = 2.8781810 \times 10^{-12}$.
Con tiempo de convergencia $t = 8$, *tiempo de ejecución* $T = 8.442$ *seg y* 15925 *iteraciones. En la* ***Figura 5.1*** *se grafican las funciones de Penalización, Energía y Objetivo. En la* ***Figura 5.2*** *se muestra la relación*

*entre x_1 y x_2. En la **Figura 5.3** podemos ver como se estabilizan las variables $x_1(t)$ y $x_2(t)$ a medida que avanza el tiempo, además, podemos ver que se puede mejorar el tiempo de convergencia, esto en cuanto a reducir el tiempo t.*

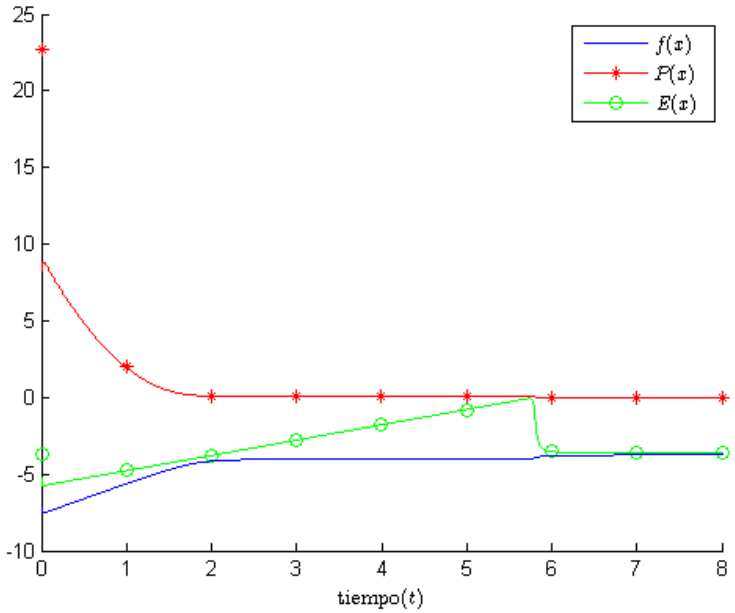

Figura 5.1: Gráfica funciones problema 5.1

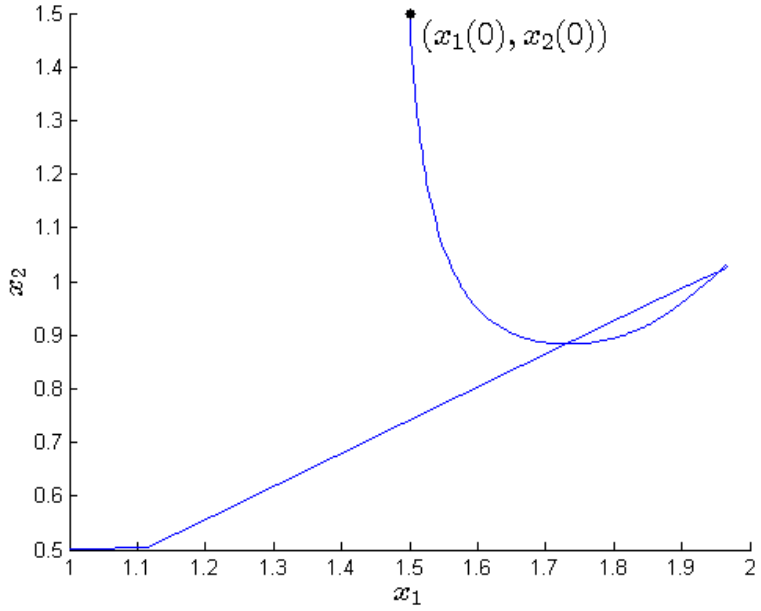

Figura 5.2: Relación entre las variables x_1 y x_2 del problema 5.1

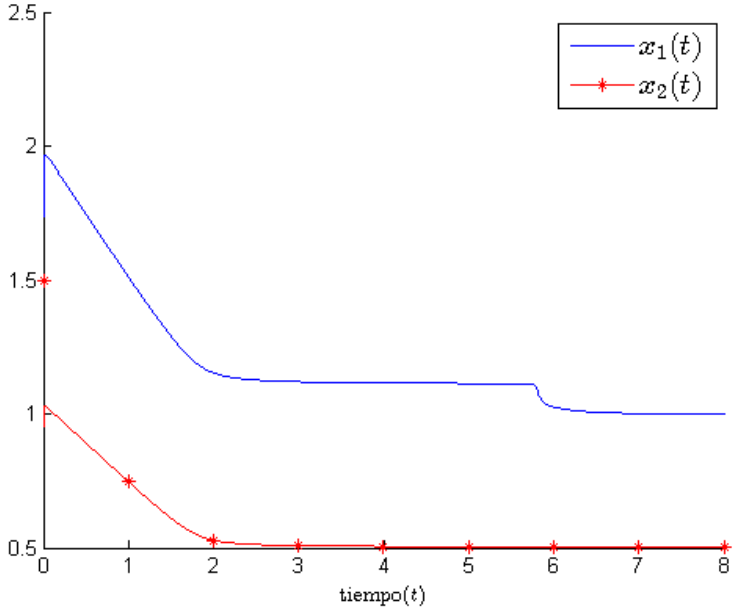

Figura 5.3: Evolución de x_1 y x_2 respecto a t en el problema 5.1

Problema 5.2.

$$\begin{array}{rc} Minimizar & z = -3x_1 - x_2 - 3x_3 \\ Sujeto\,a & 2x_1 + x_2 + x_3 \leq 2 \\ & x_1 + 2x_2 + 3x_3 \leq 5 \\ & 2x_1 + 2x_2 + x_3 \leq 6 \\ & x_1 \geq 0,\ x_2 \geq 0\, x_3 \geq 0 \end{array} \tag{5.2}$$

La solución del problema utilizando el método simplex es $\boldsymbol{x}^{*T} = \begin{pmatrix} 0.2 & 0 & 1.6 \end{pmatrix}$. *Ahora utilizando el algoritmo de RNR el primer paso es la conversión de las restricciones de desigualdad a igualdad*

$$\begin{aligned} 2x_1 + x_2 + x_3 + x_4 &= 2 \\ x_1 + 2x_2 + 3x_3 + x_5 &= 5 \\ 2x_1 + 2x_2 + x_3 + x_6 &= 6 \end{aligned}$$

$x_1, x_2, x_3, x_4, x_5, x_6, \geq 0$ *luego la matriz y los vectores* A, b, c *son:*

$$A = \begin{pmatrix} 2 & 1 & 1 & 1 & 0 & 0 \\ 1 & 2 & 3 & 0 & 1 & 0 \\ 2 & 2 & 1 & 0 & 0 & 1 \end{pmatrix}$$

$$b = \begin{pmatrix} 2 \\ 5 \\ 6 \end{pmatrix}$$

$$c = \begin{pmatrix} -3 \\ -1 \\ -3 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Según la ***Tabla 4.1*** *las funciones de activación que utilizamos son:*

$$x_i(t) \;=\; e^{u_i(t)} \text{ para i=1,2,...,6.}$$

Las constantes que aparecen en las ecuaciones del sistema dinámico se toman $c_\lambda = 1$, $c_u = 0.01, \eta = 1, \mu = 2$ *por lo que el sistema dinámico queda así:*

$$\begin{aligned}
\boldsymbol{u}'(t) &= -100\frac{c + \lambda(t)(A^T A\boldsymbol{x}(t) - A^T\boldsymbol{b}^T)}{\lambda'(t)} \\
\lambda'(t) &= \frac{2}{\boldsymbol{x}^T(t)A^T A\boldsymbol{x}(t) - 2\boldsymbol{x}^T(t)A^T\boldsymbol{b} + \boldsymbol{b}^T\boldsymbol{b}} \\
x_i(t) &= e^{u_i(t)} \text{ para i=1,2,...,6.}
\end{aligned}$$

Las condiciones iniciales elegidas son:
$\boldsymbol{x}(0)^T = \begin{pmatrix} 1.5 & 0.001 & 0.7791 & 0.9340 & 0.1299 & 0.5688 \end{pmatrix}$, $\lambda(0) = 0.4$
Se obtuvo la solución

$$\boldsymbol{x}^* = \begin{pmatrix} 0.200000353680000 \\ 0.0 \\ 1.599999824876559 \\ 0.0 \\ 0.0 \\ 3.999999403327199 \end{pmatrix}$$

$f(\boldsymbol{x}^*) = -5.400$, $P(\boldsymbol{x}^*) = 4.33873497 \times 10^{-30}$.
Con tiempo de convergencia $t = 5$, *tiempo de ejecución* $T = 4.311$ *seg y* 8889 *iteraciones. En la* ***Figura 5.4*** *se grafican las funciones de Penalización, Energía y Objetivo. En la* ***Figura 5.5*** *se muestra la relación entre las variables* x_1, x_2 *y* x_3 *y el la* ***Figura 5.6*** *podemos ver la estabilización de estas variables respecto al tiempo.*

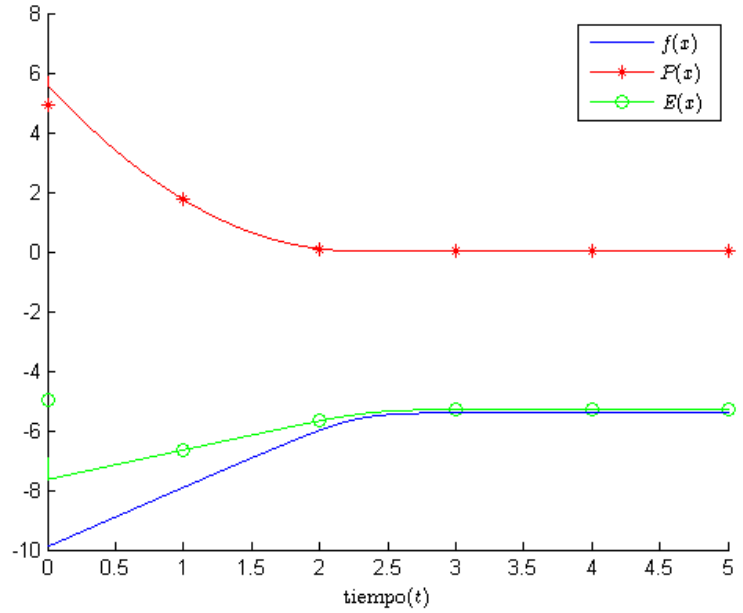

Figura 5.4: Gráfica de las funciones del problema 5.2

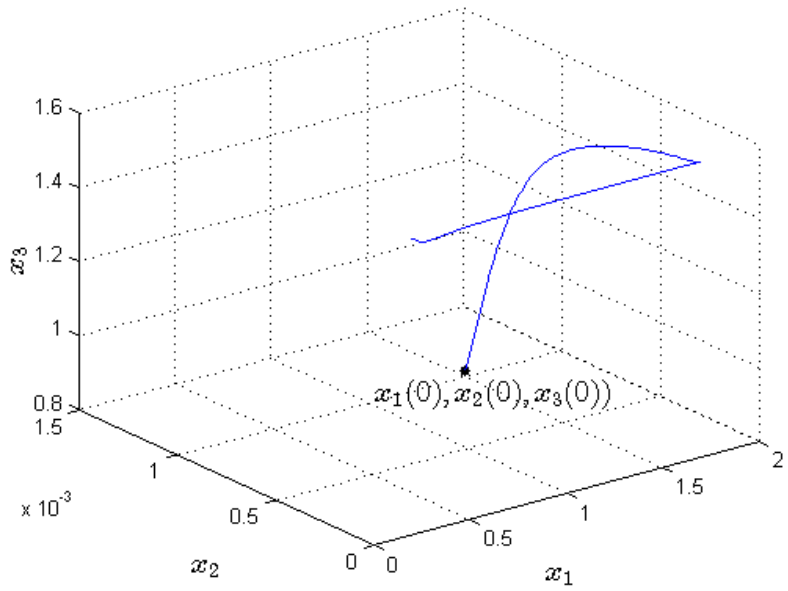

Figura 5.5: Relación entre las variables x_1, x_2 y x_3 del problema 5.2

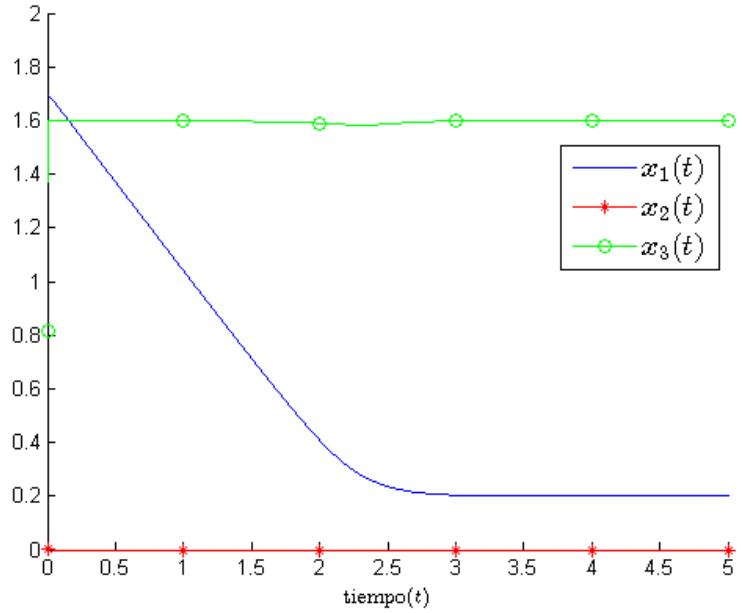

Figura 5.6: Evolución de x_1, x_2 y x_3 respecto a t en el problema 5.2

Problema 5.3.

$$\begin{array}{ll} \textit{Minimizar} & z = -3x_1 - 2x_2 \\ \textit{Sujeto a} & x_1 + 3x_2 \leq 5 \\ & 3x_1 + 2x_2 \leq 8 \\ & 2x_1 + x_2 \leq 3 \\ & x_1 \geq 0,\ x_2 \geq 0 \end{array} \tag{5.3}$$

La solución del problema utilizando el método simplex es $\boldsymbol{x}^{*T} = \begin{pmatrix} 0.8 & 1.4 & 0 \end{pmatrix}$. *Ahora utilizando el algoritmo de la RNR el primer paso es la conversión de las restricciones de desigualdad a igualdad*

$$\begin{aligned} x_1 + 3x_2 + x_3 &= 5 \\ 3x_1 + 2x_2 + x_4 &= 8 \\ 2x_1 + x_2 + x_5 &= 3 \end{aligned}$$

$x_1, x_2, x_3, x_4, x_5 \geq 0$ *luego la matriz y los vectores* A, b, c *son:*

$$A = \begin{pmatrix} 1 & 3 & 1 & 0 & 0 \\ 3 & 2 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 & 1 \end{pmatrix}$$

$$b = \begin{pmatrix} 5 \\ 8 \\ 3 \end{pmatrix}$$

$$c = \begin{pmatrix} -3 \\ -2 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Según la ***Tabla 4.1*** *las funciones de activación que utilizamos son:*

$$x_i(t) \;=\; e^{u_i(t)} \text{ para } i=1,2,3,4,5.$$

Las constantes que aparecen en las ecuaciones del sistema dinámico se eligen $c_\lambda = 1$, $c_u = 0.01, \eta = 1, \mu = 2$ *por lo que el sistema dinámico queda así:*

$$\begin{aligned} \boldsymbol{u}'(t) &= -100\frac{c + \lambda(t)(A^T A\boldsymbol{x}(t) - A^T\boldsymbol{b}^T)}{\lambda'(t)} \\ \lambda'(t) &= \frac{2}{\boldsymbol{x}^T(t)A^T A\boldsymbol{x}(t) - 2\boldsymbol{x}^T(t)A^T\boldsymbol{b} + \boldsymbol{b}^T\boldsymbol{b}} \\ x_i(t) &= e^{u_i(t)} \text{ para } i=1,2,3,4,5. \end{aligned}$$

Las condiciones iniciales elegidas son
$\boldsymbol{x}(0)^T = \begin{pmatrix} 0.1589 & 0.3351 & 0.3062 & 1.284 & 0.1231 \end{pmatrix}$, $\lambda(0) = 0.6$
Se obtuvo la solución

$$\boldsymbol{x}^* = \begin{pmatrix} 0.800062551627947 \\ 1.399977128610856 \\ 0.000000000000186 \\ 2.799810389206293 \\ 0.0 \end{pmatrix}$$

$f(\boldsymbol{x}^*) = -5.20014191$, $P(\boldsymbol{x}^*) = 6.3816369068 \times 10^{-9}$.
Con tiempo de convergencia $t = 3$, *tiempo de ejecución* $T = 3.226$ *seg y* 6745 *iteraciones. En la* ***Figura 5.7*** *se grafican las funciones de Penalización, Energía y Objetivo. en la* ***Figura 5.8*** *se mustra la relación entre* x_1 *y* x_2 *y en la* ***Figura 5.9*** *se muestra la evolución de las variables importantes respecto al tiempo*

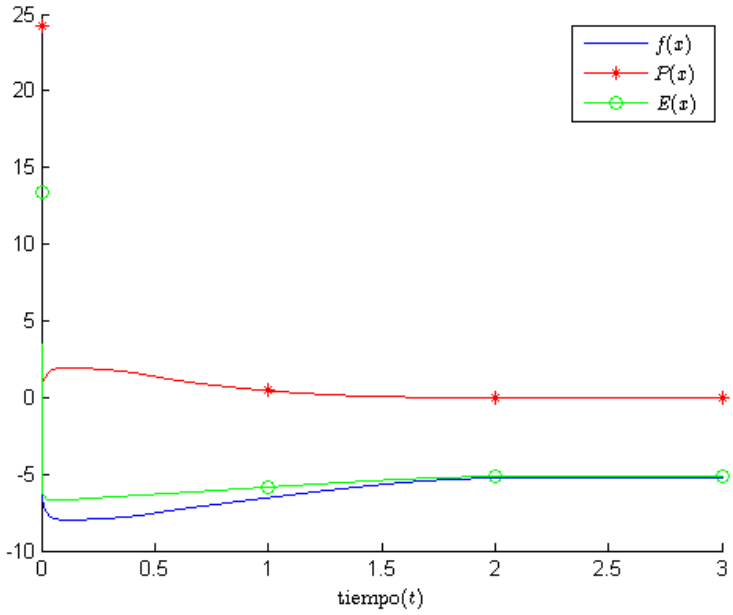

Figura 5.7: Gráfica funciones problema 5.3

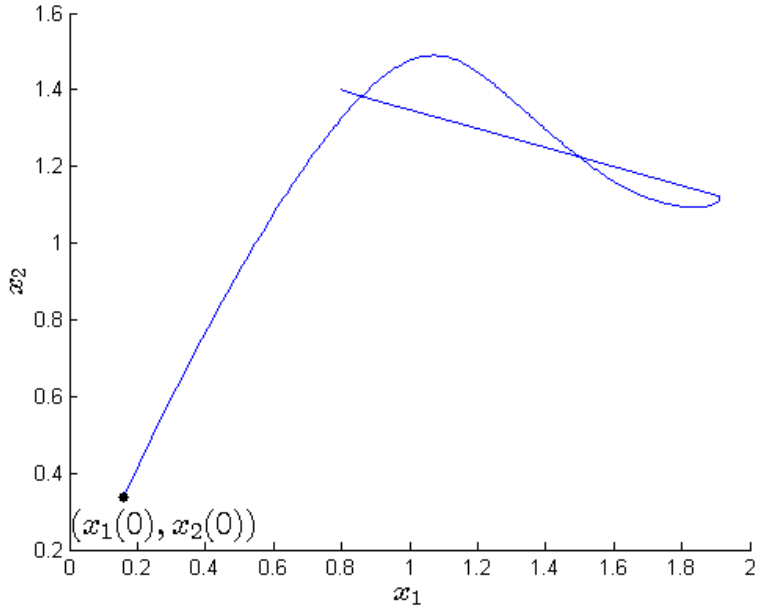

Figura 5.8: Relación entre las variables x_1 y x_2 del problema 5.3

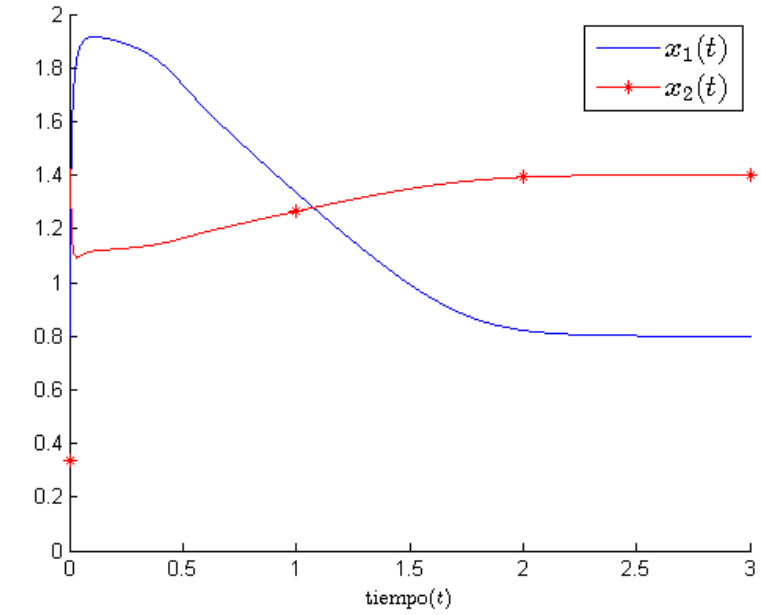

Figura 5.9: Evolución de x_1 y x_2 respecto a t en el problema 5.3

Problema 5.4. ***Problema de producción óptima***

Una fábrica requiere determinar la mezcla de producción de sus 7 productos para la próxima semana. Para cada producto, cada unidad producida requiere una cantidad conocida de tiempo de producción en cada una de las 3 máquinas de fabricación. Cada máquina tiene cierto número de unidades de tiempo disponibles por semana. Cada producto proporciona cierta ganancia por unidad producida. La ***Tabla 5.2*** *muestra los datos relacionados con el tiempo de producción de las maquinas, de igual forma muestra el tiempo del que dispone en la semana y por último la ganancia por cada unidad producida. El objetivo es determinar la política de producción óptima de tal forma que se maximice la ganancia total sin exceder la capacidad de producción de cada máquina.*

	Unidades de Producción							Tiempo disponible por semana
	Producto							
Máquina	P_1	P_2	P_3	P_4	P_5	P_6	P_7	
M_1	1	1	3	1	1	2	4	15
M_2	3	2	1	2	2	2	3	20
M_3	2	2	1	3	1	3	1	25
Ganancia	5	6	5	9	7	6	8	

Tabla 5.1: Datos del Problema 5.4

El modelo queda expresado matemáticamente de la siguiente forma:

$$\begin{aligned} Minimizar \quad & z = -\sum_{j=1}^{7} c_j x_j \\ Sujeto\, a \quad & \sum_{j=1}^{7} a_{ij} x_j \leq b_i \ para\ i = 1, 2, 3. \\ & x_j \geq 0, \ para\ j = 1, 2, ..., 7. \end{aligned} \tag{5.4}$$

donde,

x_j*: Niveles de producción para el producto* P_j*.*
c_j*: Ganancia unitaria por producto* P_j*.*
a_{ij}*: Tiempo de producción en la máquina* M_i *por unidad de producto* P_j*.*
b_i*: Tiempo de producción disponible por semana en la máquina* M_i*.*

La solución del problema utilizando el método simplex es

$$\boldsymbol{x}^{*T} = \begin{pmatrix} 0 & 0 & 2 & 7 & 2 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Ahora utilizando el algoritmo de la RNR el primer paso es la conversión de las restricciones de desigualdad a igualdad e introduciendo las variables de holgura, la matriz A *y los vectores* b, c *son:*

$$A = \begin{pmatrix} 1 & 1 & 3 & 1 & 1 & 2 & 4 & 1 & 0 & 0 \\ 3 & 2 & 1 & 2 & 2 & 2 & 3 & 0 & 1 & 0 \\ 2 & 2 & 1 & 3 & 1 & 3 & 1 & 0 & 0 & 1 \end{pmatrix}$$

$$b = \begin{pmatrix} 15 \\ 20 \\ 25 \end{pmatrix}$$

$$c^T = \begin{pmatrix} 5 & 6 & 5 & 9 & 7 & 6 & 8 & 0 & 0 & 0 \end{pmatrix}$$

Según la ***Tabla 4.1*** *las funciones de activación que utilizamos son:*

$$x_i(t) \;=\; e^{u_i(t)} \text{ para i=1,2,...,10.}$$

Las constantes que aparecen en las ecuaciones del sistema dinámico se eligen $c_\lambda = 1$, $c_u = 0.01, \eta = 1, \mu = 2$*. Luego el sistema dinámico queda*

así:

$$\begin{aligned} \boldsymbol{u}'(t) &= -100\frac{c+\lambda(t)(A^TA\boldsymbol{x}(t)-A^T\boldsymbol{b}^T)}{\lambda'(t)} \\ \lambda'(t) &= \frac{2}{\boldsymbol{x}^T(t)A^TA\boldsymbol{x}(t)-2\boldsymbol{x}^T(t)A^T\boldsymbol{b}+\boldsymbol{b}^T\boldsymbol{b}} \\ x_i(t) &= e^{u_i(t)} \text{ para } i=1,2,\ldots,10. \end{aligned}$$

Las condiciones iniciales elegidas son:
$\boldsymbol{x}(0)^T = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0.3793 & 1.6382 & 0.8960 \end{pmatrix}$, $\lambda(0) = 1.4$

Se obtuvo la solución

$$\boldsymbol{x}^* = \begin{pmatrix} 0.0 \\ 0.0 \\ 1.999999999814719 \\ 6.999999999675760 \\ 2.000000000741124 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \end{pmatrix}$$

$f(\boldsymbol{x}^*) = 87.000000001343295$, $P(\boldsymbol{x}^*) = 3.068128292401 \times 10^{-19}$.
Con tiempo de convergencia $t = 5$, *tiempo de ejecución* $T = 40.227$ *seg y* 76845 *iteraciones. En la* ***Figura 5.10*** *se grafican las funciones de Penalización, Energía y Objetivo. En la* ***Figura 5.11*** *vemos el comportamiento de las variables respecto al tiempo.*

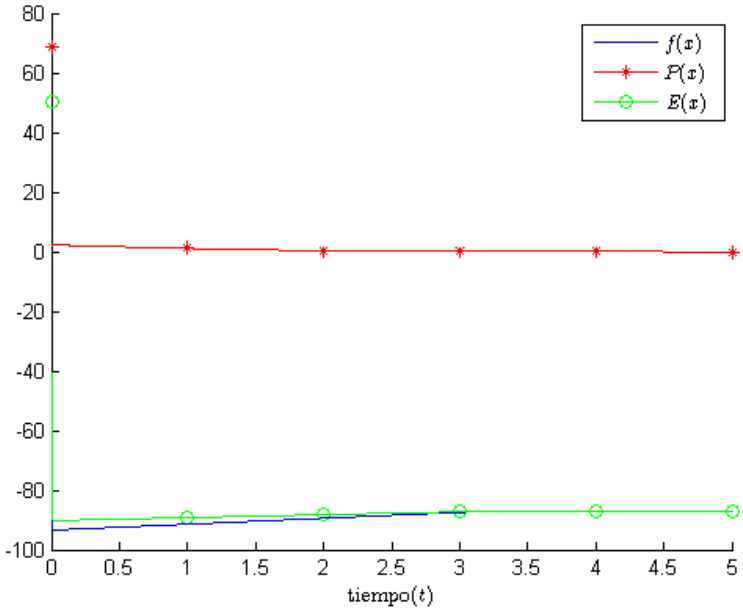

Figura 5.10: Gráfica funciones problema 5.4

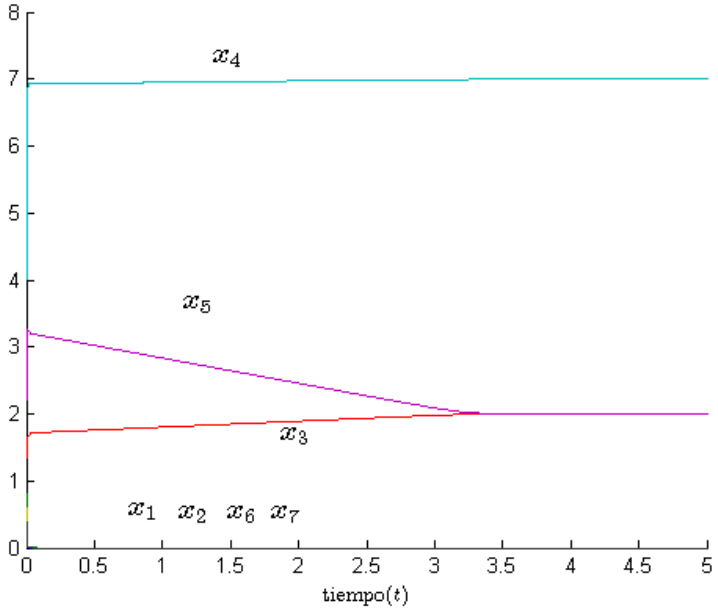

Figura 5.11: Evolución de las variables respecto al tiempo en el problema 5.4

Problema 5.5. *Asignación óptima de tareas*

Una compañía de software tiene a cargo un proyecto a desarrollar un compuesto de 5 grandes módulos para ser trabajados por 5 programadores diferentes. Se desea que cada módulo sea desarrollado por un solo programador y que cada programador desarrolle un solo modulo del software. Debido a los diferentes grados de dificultad de los módulos y a las diferencias individuales de las capacidades y aptitudes de los programadores, el tiempo (en días) que ellos emplean es diferente y se muestran en la ***Tabla 5.2****.*

	Prog 1	Prog 2	Prog 3	Prog 4	Prog 5
Módulo 1	2	4	4	3	6
Módulo 2	2	6	5	4	6
Módulo 3	5	6	5	3	7
Módulo 4	3	5	7	2	4
Módulo 5	8	5	6	2	1

Tabla 5.2: Datos del Problema 5.5.

La gerencia de la compañía desea tomar la decisión adecuada referente a cuales módulos asignar a cada uno de los programadores de tal forma que se logre minimizar el tiempo total de desarrollo del software.

El modelo expresado matemáticamente

La decisión a tomar es asignar los módulos a cada uno de los programadores, teniendo en cuenta que cada módulo debe ser desarrollado por un solo programador y que cada programador debe desarrollar un solo módulo de software, entonces se define:

x_{ij}*: Asignación del módulo* i *al programador* j*.*

c_{ij}: *Tiempo que tarda el programador* j *en desarrollar el módulo* i.

La solución del problema debe dar en los valores 0 *si no se asigna la tarea y* 1 *si la tarea es asignada.*

$$\begin{aligned} Minimizar \quad & z = \sum_{i=1}^{5}\sum_{j=1}^{5} c_{ij}x_{ij} \\ Sujeto\,a \quad & \sum_{j=1}^{5} x_{ij} = 1 \ para\ j = 1,2,3,4,5. \\ & \sum_{i=1}^{5} x_{ij} = 1 \ para\ i = 1,2,3,4,5. \\ & x_{ij} \geq 0, \ para\ i,j = 1,2,3,4,5. \end{aligned} \tag{5.5}$$

La solución del problema utilizando el método simplex es:

$x_{11} = 0$	$x_{12} = 1$	$x_{13} = 0$	$x_{14} = 0$	$x_{15} = 0$
$x_{21} = 1$	$x_{22} = 0$	$x_{23} = 0$	$x_{24} = 0$	$x_{25} = 0$
$x_{31} = 0$	$x_{32} = 0$	$x_{33} = 1$	$x_{34} = 0$	$x_{35} = 0$
$x_{41} = 0$	$x_{42} = 0$	$x_{43} = 0$	$x_{44} = 1$	$x_{45} = 0$
$x_{51} = 0$	$x_{52} = 0$	$x_{53} = 0$	$x_{54} = 0$	$x_{55} = 1$

Ahora utilizando el algoritmo de RNR y ya que las restricciones están en igualdad, definimos la matriz A *y los vectores* b, c*:*

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$b^T = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$c^T = \begin{pmatrix} 2 & 4 & 4 & 3 & 6 & 2 & 6 & 5 & 4 & 6 & 5 & 6 & 5 & 3 & 7 & 3 & 5 & 7 & 2 & 4 & 8 & 5 & 6 & 2 & 1 \end{pmatrix}$$

Según la ***Tabla 4.1*** *las funciones de activación que utilizamos son:*

$$x_{ij}(t) \;=\; \frac{1}{1+e^{-u_{ij}(t)}} \text{ para i,j=1,2,3,4,5.}$$

Las constantes que aparecen en las ecuaciones del sistema dinámico se elige $c_\lambda = 1$, $c_u = 0.01, \eta = 1, \mu = 2$. *Por lo que el sistema dinámico queda así:*

$$\begin{aligned} \boldsymbol{u}'(t) &= -100\frac{c+\lambda(t)(A^TA\boldsymbol{x}(t)-A^T\boldsymbol{b}^T)}{\lambda'(t)} \\ \lambda'(t) &= \frac{2}{\boldsymbol{x}^T(t)A^TA\boldsymbol{x}(t)-2\boldsymbol{x}^T(t)A^T\boldsymbol{b}+\boldsymbol{b}^T\boldsymbol{b}} \\ x_{ij}(t) &= \frac{1}{1+e^{-u_{ij}(t)}} \text{ para i,j=1,2,3,4,5.} \end{aligned}$$

En este caso,

$$\boldsymbol{x}^T = \begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{21} & x_{22} & \cdots & x_{51} & x_{52} & x_{53} & x_{54} & x_{55} \end{pmatrix}.$$

La herramienta que utilizaremos para resolver el sistema dinámico es SIMULINK de MATLAB, para esto es necesario mostrar el diseño de la red en la ***Figura 5.13***

Las condiciones iniciales usadas fueron $x_{ij}(0) = 0.5$ *para* $i, j = 1, 2, 3, 4, 5$.

$y\ \lambda(0) = 1.2$.

La solución obtenida

$x_{11} = 0$	$x_{12} = 0.999277141541583$	$x_{13} = 0.001155885901717$
$x_{21} = 0.999853237905967$	$x_{22} = 0$	$x_{23} = 0$
$x_{31} = 0$	$x_{32} = 0$	$x_{33} = 0.998699110411423$
$x_{41} = 0$	$x_{42} = 0$	$x_{43} = 0$
$x_{51} = 0$	$x_{52} = 0$	$x_{53} = 0$

$x_{13} = 0.001155885901717$	$x_{14} = 0$
$x_{24} = 0$	$x_{25} = 0$
$x_{34} = 0.001155788773906$	$x_{35} = 0$
$x_{44} = 0.999277507357916$	$x_{45} = 0$
$x_{54} = 0$	$x_{55} = 0.999854835864023$

$f(\boldsymbol{x}^*) = 14.000000001343295$, $P(\boldsymbol{x}^*) = 3.068128292401 \times 10^{-19}$.

Con tiempo de convergencia $t = 30$, *tiempo de ejecución* $T = 58.653$ *seg y* 1081 *iteraciones. En la* ***Figura 5.12*** *se grafican las funciones de Penalización, Energía y Objetivo.*

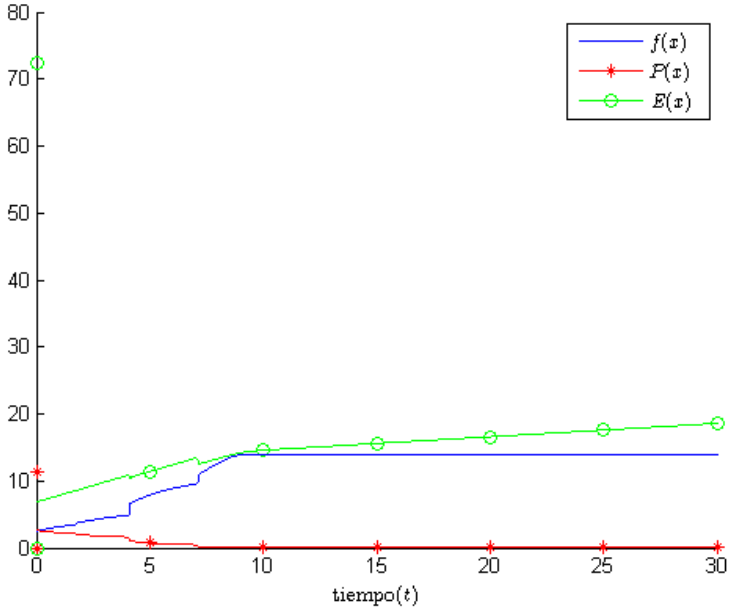

Figura 5.12: Gráfica funciones problema 5.5

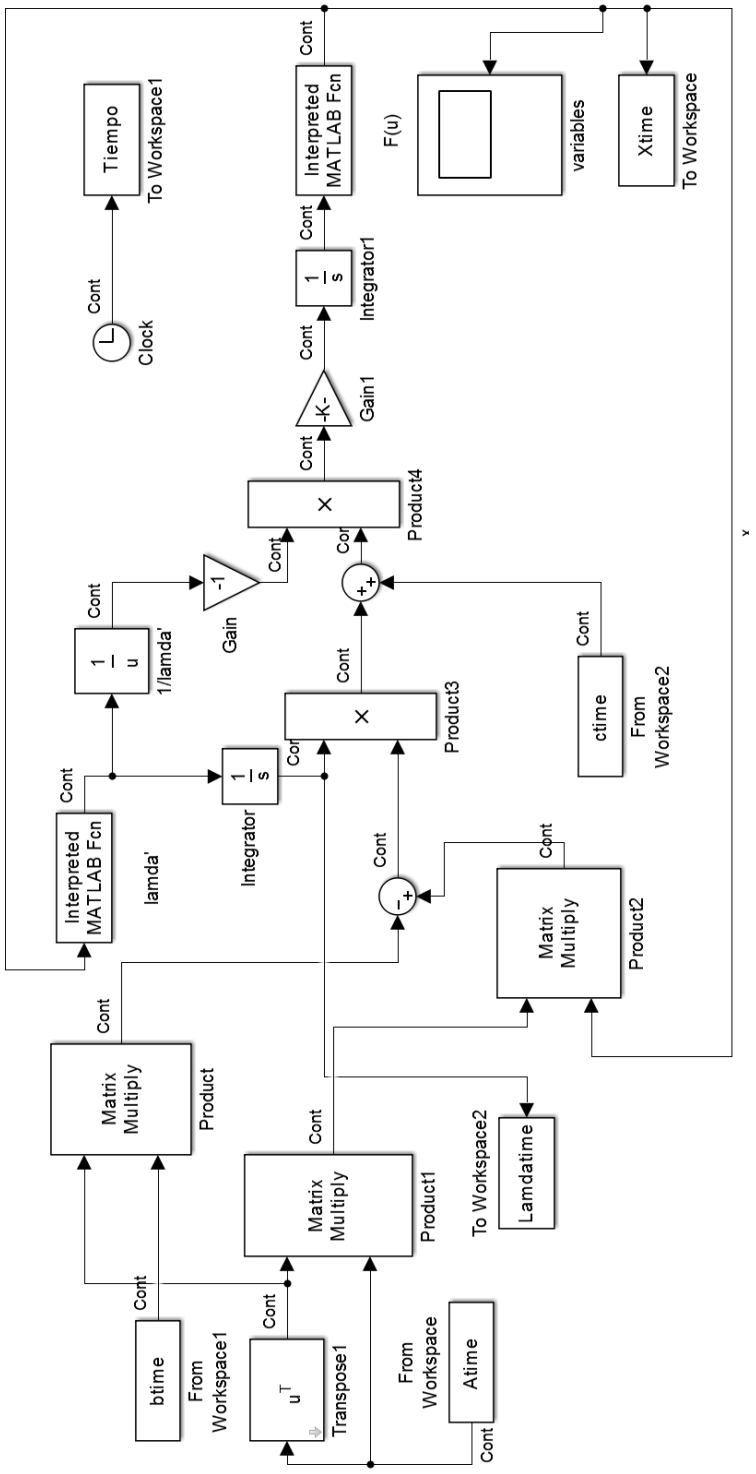

Figura 5.13: Simulador simulink

Capítulo 6

Conclusiones

En este documento se presenta un método para solucionar problemas de programación lineal basado en redes neuronales, esta técnica consiste en reformular el problema de programación lineal como un sistema dinámico por lo que se convierte en una herramienta muy importante al ofrecer una alternativa a los métodos discretos existentes y es además una técnica que se puede extender no solo a problemas lineales sino también a problemas, no lineales. En cuanto a las condiciones iniciales del sistema dinámico, éstas se toman de forma arbitraria sin tener un precedente acerca de cómo debe ser esta elección además de esto, según el tipo de problema, el valor inicial de λ influye en la convergencia.

Al resolver el sistema dinámico con problemas que tuvieran involucradas más variables se evidenció un inconveniente en cuanto que el método de *Runge kutta* necesita mucho recurso computacional al incrementarse el número de variables; por lo cual el tiempo de ejecución es demasiado alto para hacer las pruebas. La ventaja del método es la implementación física ya que se puede llevar a circuitos electrónicos lo que aumentaría la velocidad de convergencia de problemas de gran escala en tiempo real.

En trabajos futuros, se puede estudiar, cómo el método de redes neuronales artificiales se pude extender a problemas no lineales, usando la función de energía vista en este documento y bajo el supuesto de que la función objetivo que sea convexa. la misma teoría usada garantiza la convergencia. Por lo cual se puede estudiar el comportamiento numérico para estos problemas y ver si se tiene una mejor alternativa a los métodos más utilizados actualmente.

Bibliografía

[1] A. Cichocki and A. Bargiela, *Neural networks for solving linear inequality systems*, Parallel Computing **22** (1997), no. 11, 1455 – 1475.

[2] A Cichocki, R Unbehauen, K Weinzierl, and R Hölzel, *A new neural network for solving linear programming problems*, European Journal of Operational Research **93** (1996), no. 2, 244 – 256.

[3] GB Danzig, *Linear programming and extensions. 1963.*

[4] Michael R. Cullen Denniz G. Zill, *Ecuaciones diferenciales con problemas con valores en la frontera*, CENGAGE Learning, 2009.

[5] Clovis C Gonzaga, *Path-following methods for linear programming*, SIAM review **34** (1992), no. 2, 167–224.

[6] Pedro Isasi Viñuela and IM Galván León, *Redes de neuronas artificiales*, Un Enfoque Práctico, Editorial Pearson Educación SA Madrid España (2004).

[7] Structure of Neuron, *http://www.sciencedirect.com/science/article/pii/s0167819196000658.*

[8] Rosana Pérez and Tomás Díaz, *Minimización sin restricciones*, Editorial Universidad del Cauca (2010).

[9] Lawrence Perko, *Differential equations and dynamical systems*, Springer, 2000.

[10] Hamdy A Taha, *Investigación de operaciones*, Pearson Educación, 2004.

[11] Jun Wang, *On the asymtotic properties of recurent neural networks for optimization*, Departament of industrial Technology University of North Dakota (1991).

[12] Jun Wang and Vira Chankong, *Recurrent neural networks for linear programming: Analysis and design principles*, Computers and Operations Research **19** (1992), no. 3, 297 – 311.

[13] Ph.D Wang, Jun, *On the trainability, stability, representability, and realizability of artificial neural networks*, 1991.

[14] Jacek M Zurada, *Introduction to artificial neural networks*, New York, West Pub. Co (1992).