Uptc
Universidad Pedagógica y Tecnológica de Colombia

Ingeniería de Sistemas y Computación

Taller No 2 - Inteligencia Computacional

Conceptos de Inteligencia y Redes Neuronales

David Felipe Huertas Flechas, German Andrés Torres Cánchala, Lina Alejandra Melo Patiño.

Punto 1: Leyendo... ando.

REPRESENTACIÓN DEL CONOCIMIENTO

Los softwares de Inteligencia Artificial trabajan con información que se ha adquirido. Para que una máquina se comporte de manera inteligente debe tener una Base de conocimiento, pero no es suficiente con esto, pues la máquina debe ser capaz de utilizarlo, para ello hay que se proveen mecanismos que le permitan razonar con el conocimiento almacenado. La representación del conocimiento es el área de la IA que estudia cómo representar el conocimiento adquirido y a las formas, estructuras o técnicas que permiten representarlo, se les llama Técnicas de Representación del Conocimiento.

CARACTERÍSTICAS DE UN BUEN SISTEMA DE REPRESENTACIÓN DEL CONOCIMIENTO

- Suficiencia de la representación: Capacidad de representar todos los conocimientos necesarios.
- Suficiencia deductiva: Capacidad de manipular las estructuras de representación para obtener nuevas estructuras que puede ser visto como nuevo conocimiento deducido del antiguo.
- Eficiencia deductiva: Capacidad de incorporar información adicional en el conocimiento con el fin de que se realicen inferencias más prometedoras.
- Eficiencia en la adquisición: Capacidad de adquirir nueva información con facilidad

Ya que no se puede encontrar un formalismo o técnica que cumpla con las anteriores características, por lo que se puede exigir que posea.

- Exactitud: El modelo debe ser fiel a la realidad para no distorsionarla.

- Adecuación: Poder expresivo y eficacia del formalismo, la primera es todo lo que se puede decir con la notación, si la representación permite realizar o evitar distinciones sutiles. La eficacia, hace referencia a las estructuras concretas.

TIPOS DE CONOCIMIENTOS

- Conocimiento Declarativo: conocimiento relacional simple, pueden almacenarse en estructuras del mismo tipo que las utilizadas en los sistemas de bases de datos.
- Conocimiento operacional o procedimental, que especifica qué hacer cuando se da una determinada situación
- Conocimiento heredable: Una de las formas más útiles de inferencia en la herencia de propiedades, donde los elementos de una clase heredan los atributos y los valores de otras clases más generales[1]

TÉCNICAS DE REPRESENTACIÓN DEL CONOCIMIENTO

Basados en conceptos

Representan los objetos del dominio y sus posibles valores que pueda tomar cada una de sus propiedades

- Marcos: Están conformados por un nombre y un conjunto de propiedades también llamadas ranuras, se pueden relacionar con otros marcos y formar una jerarquía entre ellos.

Basados en relaciones

Son capaces de representar las relaciones que existen entre las entidades y los conceptos.

- Lógica proposicional: Una preposición es un enunciado o frase que tiene un significado determinado que puede ser verdadera o falsa.
- Lógica de predicados: Hay argumentos que su validez no puede ser probada por la lógica proposicional.
- Redes semánticas: son grafos formados por nodos y arcos unidireccionales, los nodos representan objetos y los arcos, la relación entre estos
- Ontologías: Es una especificación formal y explícita de una conceptualización compartida. Las Ontologías son: formales, compartidas y explícitas. Están compuestas por:

[1] Julio Cesar Ponce Gallegos, Aurora Torres Soto, et al. Inteligencia Artificial.1ra. Iniciativa Latinoamericana de Libros de Texto Abiertos (LATln), 2014.

- Clases
- Selecciones
- Funciones
- Instancias
- Taxonomía

- Axiomas
- Propiedades

Basados en acciones

Permiten representar el conocimiento por medio de acciones básicas

- Reglas de producción: La representación basada en reglas parte del principio que la relación lógica entre un conjunto de objetos puede ser representadas mediante una regla, cada regla tiene la forma “si P entonces Q”, donde P es la es el antecedente, premisa, condición o situación de la regla y Q es el consecuente, conclusión, acción o respuesta.

AGENTES INTELIGENTES

Los agentes han surgido por la necesidad de abordar problemas muy complejos (sistemas distribuidos) ya que en la actualidad muchos de los problemas están físicamente distribuidos ;el crecimiento de sistemas multiagentes están compuestos por sistemas de agentes autónomos que interactúan entre sí.

La noción de agencia está aplicada a diferentes líneas de trabajo como se muestra en la siguiente imagen.

Figura 1: Líneas de trabajo de “Agencia”.

¿QUÉ ES UN AGENTE?

Un agente es una entidad que posee características generales como la capacidad de percibir el entorno, actuar sobre el entorno, comunicarse con otros agentes, tienen objetivos y recursos propios. Norvig y Russel dicen que "la meta de la IA es diseñar un agente inteligente o racional que opere o actúe adecuadamente en sus ambientes." Es decir debe hacer lo correcto en su entorno y hará la mejor opción posible.[2]

Wooldridge & Jennings definen a un agente como: "un sistema de computación situado en algún entorno, que es capaz de una acción autónoma y flexible para alcanzar sus objetivos de diseño." de donde salen características como:

- Autonomía.
- Habilidad social.
- Reactividad.
- Proactividad.

Se puede representar la interacción del Agente con su entorno mediante su historia, la cual se define como una secuencia estado-acción-estado. Dos agentes serán equivalentes respecto a un entorno, si sus historias son iguales; sin embargo hay agentes que no toman decisiones basados en su historia.

ARQUITECTURAS DE AGENTES

Para dar soporte a los sistemas multiagentes se han propuesto teorías y arquitecturas a seguir. Las teorías son especificaciones del comportamiento de los agentes y se deben satisfacer mientras que las arquitecturas representan un punto medio entre la especificación (Las teorías del agente basadas en un enfoque intencional son las más utilizadas y estudiadas) y la implementación. En ellas se identifican las principales funciones que determinan su comportamiento y se definen las interdependencias que existen entre ellas; en las arquitecturas se definen preguntas como: "¿Cómo se va a construir sistemas informáticos que satisfagan las propiedades especificadas por una particular teoría de agentes? ¿Cómo el agente se puede descomponer en la construcción de un conjunto de módulos componentes y cómo estos módulos deben interactuar? ¿Qué estructuras de software / hardware son apropiadas?"[3]

[2] Julio Cesar Ponce Gallegos, Aurora Torres Soto, et al. Inteligencia Artificial.1ra. Iniciativa Latinoamericana de Libros de Texto Abiertos (LATIn), 2014.
[3] Julio Cesar Ponce Gallegos, Aurora Torres Soto, et al. Inteligencia Artificial.1ra. Iniciativa Latinoamericana de Libros de Texto Abiertos (LATIn), 2014.

De la misma manera existen diferentes tipos de arquitecturas:

- Arquitecturas basadas en la lógica - Agentes deductivos.
- Arquitecturas reactivas - Agentes reactivos.
- Arquitecturas en capas - Agentes híbridos.
- Arquitecturas de razonamiento práctico - Agentes PRS y Agentes BDI.
- Agentes de Razonamiento Procedural (PRS). **Es la más estudiada y utilizada.**

LENGUAJES DE AGENTES

Entre estos se pueden destacar los siguientes:

- Agent0
- Concurrent METATEM
- JASON **Se hace enfoque en este lenguaje.**
- JACK
- OPENPRS
- APL -2APL, 3APL

EL lenguaje JASON es multiplataforma hecho en JAVA y cuenta con las siguientes características: negación fuerte de fórmulas, manejo de planes de falla, comunicación entre agentes, anotaciones en las creencias, soporte del desarrollo de entornos (Programados en JAVA), anotaciones en las etiquetas o labels de los planes que pueden ser usadas por funciones selectoras modificadas o extendidas, posibilidad de correr un sistema multiagente distribuido sobre la red, extensible (funciones selectoras, funciones de verdad y la arquitectura completa) , librería de acciones internas esenciales y acciones internas extensibles por el usuario, programadas en JAVA.[4]

SISTEMAS MULTIAGENTES

Un sistema multiagente como su nombre lo indica, está compuesto de agentes que tiene una funcionalidad y que interactúan entre sí y su entorno. Actualmente se implementan en muchos sistemas y a que el paradigma de sistemas multiagentes ofrece un conjunto de conceptos y técnicas para modelar, diseñar e implementar sistemas distribuidos complejos y a eso se suma el hecho de que cada vez más los sistemas se diseñan en forma más distribuida y en múltiples dispositivos.

[4] Julio Cesar Ponce Gallegos, Aurora Torres Soto, et al. Inteligencia Artificial.1ra. Iniciativa Latinoamericana de Libros de Texto Abiertos (LATIn), 2014.

Los protocolos de comunicación permiten a los agentes intercambiar y entender mensajes, se debe tener el cuenta que necesitan para que 2 agentes puedan comunicarse es necesario acordar la terminología. (FIPA ACL es considerado un estándar). Por otra parte, los protocolos de interacción permiten a los agentes tener conversaciones, que serán intercambios de mensajes estructurados.

Algunas de las características más importantes de los sistemas multiagentes son:

- Proveen una infraestructura que especifica los protocolos de comunicación y de interacción entre agentes.
- Pueden tener intereses propios o ser cooperativos.
- Suelen ser abiertos y suelen no tener un diseño centralizado.
- Pueden compartir conocimientos sobre el problema y las posibles soluciones entre agentes.

Para que 2 agentes realicen una actividad debe existir la coordinación o negociación dependiendo del tipo de agentes (no-antagonistas o competitivos).

2: Y ahora, redes neuronales.

En el siguiente enlace encontrará todos los recursos (código, datos y videos) de todos los problemas del taller:

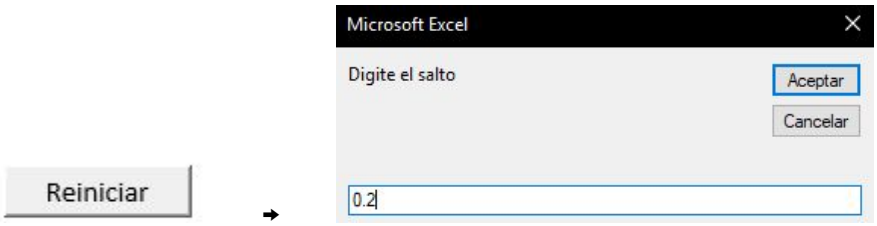
https://drive.google.com/drive/folders/11Yzxak_A1flHQbZjsnZAFCUkLTCVdpkc?usp=sharing

CUADRO DE INFORMACIÓN	
Lenguaje de programación	Phyton y C#
Sistema Operativo	Windows 10 Pro-64 bits
Procesador del computador	Intel® Core ™ i5-6200U CPU 2.30GHz (4CPUs), ^2.4GHz
Memoria del computador	8192MB RAM
Placa del computador	ASUSTek COMPUTER INC X555UB

2.1 Como primera medida vamos a minimizar. Para ello se requiere que se demuestre la implementación de una función de costo, usando descenso de gradiente, de forma numérica y usando reglas de cálculo, teniendo en cuenta las siguientes funciones:

2.1 Se generó un programa que generara una función de costos en Visual Basic For Applications gestionado desde una hoja de excel. Este programa consta de las siguientes funcionalidades:

- Al abrir el archivo de excel, o dar click en el botón "*Reiniciar*", un componente pide cada uno de los parámetros necesarios para desarrollar la función.



Este botón ejecuta una función la cual limpia la hoja, recibe los datos, y genera una nueva tabla con los resultados procesados.

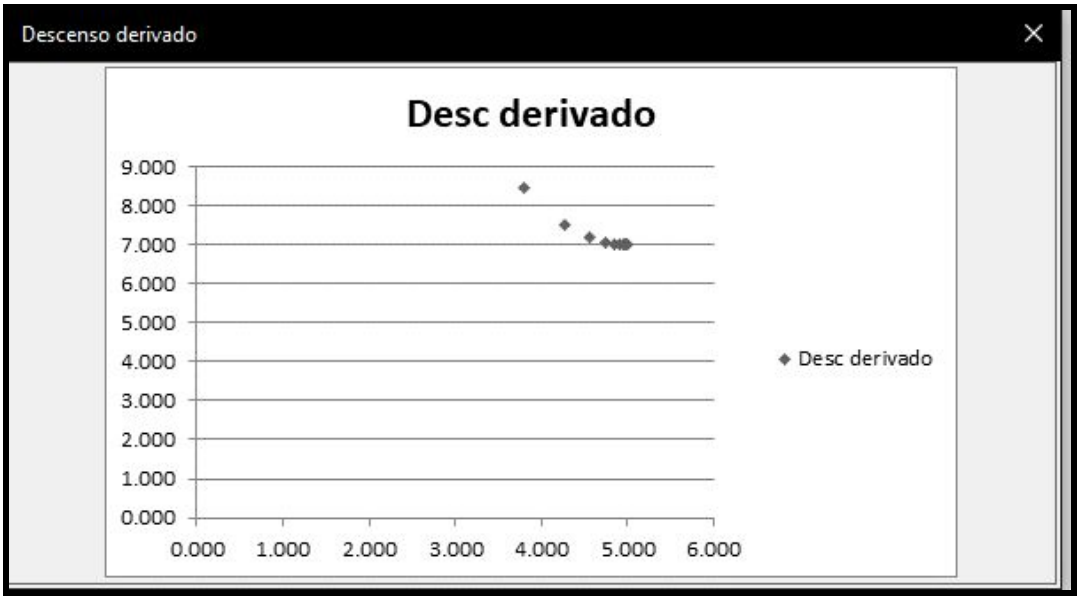
```
For i = 1 To ciclos
 b = b - salto * (2 * (b - Deltax))
 Cells(i + 8, 6).Select
 Cells(i + 8, 6).Select

 Cells(i + 8, 6).Select
 Cells(i + 8, 6).Value = (((b - Deltax) ^ 2) + Deltay)

 Cells(i + 8, 4).Select
 Cells(i + 8, 4).Select

 Cells(i + 8, 4).Select
 Cells(i + 8, 4).Value = i


 Cells(i + 8, 5).Select
 Cells(i + 8, 5).Select

 Cells(i + 8, 5).Select
 Cells(i + 8, 5).Value = b

 If i > 1 Then Cells(i + 8, 7).Value = (Cells((i - 1) + 8, 9).Value) - b
Next i
```

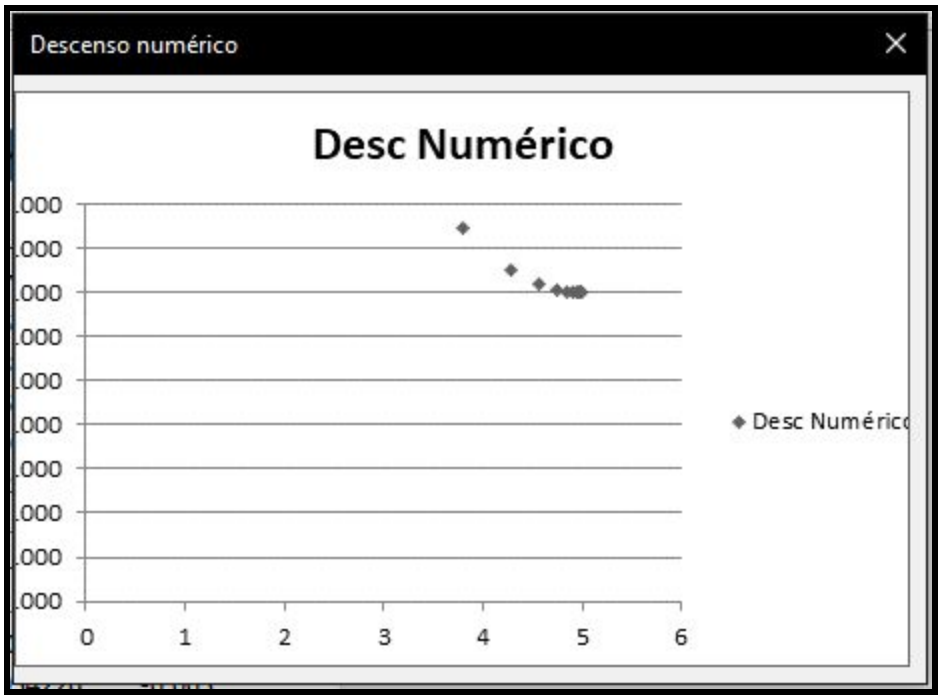
La tabla tiene los datos necesarios para encontrar el descenso derivado y el numérico.

	Desc derivado	Y	Delta X	)esc Numéric	Y	Delta Y
1	3.800	8.440		4	8.454	
2	4.28	7.518	-4.280	4	7.532316091	-0.476
3	4.568	7.187	-4.568	5	7.196922865	-0.286
4	4.7408	7.067	-4.741	5	7.074123274	-0.172

La función llama a otra función en la que se generan las gráficas de los descensos, mostrándose en nuevas ventanas.

```
Sub MostrarGrafico()
Dim myChart As Chart
Dim Data As Range
Dim tmpImage As String

Application.ScreenUpdating = False

    Set Data = Range(Cells(8, 5), Cells(8, 6).End(xlDown))
    Set myChart = ActiveSheet.Shapes.AddChart.Chart
    myChart.SetSourceData Source:=Data
    myChart.ChartStyle = 1
     myChart.ChartType = xlXYScatter


    tmpImage = ThisWorkbook.Path & Application.PathSeparator & "imgChart.gif"
    myChart.Export Filename:=tmpImage, FilterName:="GIF"
    UserForm2.Image1.Picture = LoadPicture(tmpImage)

    ActiveSheet.ChartObjects(1).Delete

Application.ScreenUpdating = True

End Sub
```





2.2 Dado el dataset del taller IC2, se generan los programas con el fin de observar el comportamiento de una red neuronal por clasificación y por regresión.

X1	X2	X3	Y
42	2	1	0
46	3	1	0
50	1	1	0
30	1	1	0
40	1	1	0
39	2	1	0
15	1	0	0
16	1	1	0
20	2	0	0
18	3	1	0
24	3	0	1
30	1	1	0
37	2	1	0
19	1	1	0
23	1	1	0
49	2	0	1
40	1	1	0
24	2	0	1
22	2	0	1
26	3	0	1

2.2.1 KneighborsClassifier

Se realiza el código para el método en el lenguaje de programación Python, que nos da un score de 1.0 y donde al pasar el set de datos de pruebas nos da como resultado:

```
In [4]: ClasificadorFeliz.score(ME,MR)

Out[4]: 1.0
```

```
In [6]: MV = np.array([[28.0,3.0,0.0],
                       [22.0,3.0,1.0],
                       [24.0,1.0,1.0],
                       [17.0,2.0,1.0],
                       [50.0,2.0,0.0],
                       [18.0,1.0,1.0],
                       [25.0,2.0,1.0],
                       [40.0,2.0,1.0],
                       [33.0,1.0,1.0],
                       [29.0,1.0,0.0],
                       [31.0,2.0,0.0],
                       [20.0,3.0,1.0],
                       [25.0,3.0,0.0],
                       [49.0,2.0,0.0]
                      ])
        salida = ClasificadorFeliz.predict(MV)
        salida

Out[6]: array([1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1])
```

Donde solo se equivoca en 3 de los 14 resultados, aproximadamente en un 22% de las respuestas dadas.

2.2.2 Linear Regressor

Se realiza el código para el método en el lenguaje de programación Python, que nos da un error de 0.21 aproximadamente y donde al pasar el set de datos de pruebas nos da como resultado, valores muy cercanos a los esperados, pero no lo suficiente:

```
In [5]: import numpy as np
        print("Error: %r" %np.sqrt(np.mean((predicted - expected) ** 2)))
        plt.show()

        Error: 0.21110130099000743
```

```
In [6]: print(expected)

        [1 0 0 0 1 0 0 0 0 0 1 0 1 1]
```

```
In [7]: print(predicted)

        [ 0.82518408  0.09663318 -0.11684826 -0.04715597  0.84811287 -0.15389669
          0.00224194  0.09486302 -0.06127561  0.6055279   0.73079284  0.0842837
          0.80665986  0.84193813]
```

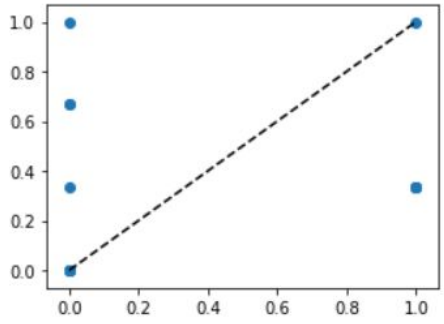
2.2.3 KneighborsRegressor

El proceso es realmente similar al algoritmo anterior salvo que se cambia la librería importada.

```
In [1]: #Importamos el KNeighbors Regresor y np para manejo de vectores
        from sklearn.neighbors import KNeighborsRegressor
        import matplotlib.pyplot as plt
        import numpy as np
```

```
In [3]: #Definimos la cantidad de vecinos
        quantity_neighbors = 3
        #Creamos el regresor y pasamos la cantidad de vecinos
        kneighborsRegresor = KNeighborsRegressor(quantity_neighbors)
```

```
In [4]: #Alimentamos al regresor
        kneighborsRegresor.fit(X, target)

Out[4]: KNeighborsRegressor(algorithm='auto', leaf_size=30, metric='minkowski',
                            metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                            weights='uniform')

In [5]: #Verificamos el puntaje de exactitud
        kneighborsRegresor.score(X,target)

Out[5]: 0.46666666666666656
```




```
In [8]: kneighborsRegresor.predict(x_test)

Out[8]: array([0.33333333, 0.66666667, 0.66666667, 0.        , 0.33333333,
               0.        , 1.        , 0.        , 0.        , 0.33333333,
               0.33333333, 0.        , 1.        , 0.33333333])

In [9]: #Imprimos los valores esperados
        y_test

Out[9]: [1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1]
```

2.2.4 NNR – Suma

Se realiza el código para el método en el lenguaje de programación Python, que nos da como resultado los pesos obtenidos y el baias, tras 100 iteraciones.

```
In [6]: print('pesos obtenidos',w)
        print('biases obtenido',b)

        pesos obtenidos [[2.64585005e+129]
         [1.34566965e+128]
         [5.39774848e+127]]
        biases obtenido [7.66066221e+127]
```

Preguntas:

- ¿Cuál de los métodos es más efectivo respecto de los demás? ¿Por qué?

  Este depende de la cantidad de datos, por ejemplo el regressor tiene un rendimiento mucho mayor (al igual que el nearlessNeighbors) cuando la cantidad de datos son pequeñas o no la cantidad de variables no son muchas.

- ¿Cómo funcionan los métodos de regresión y clasificación para redes neuronales?

  Clasificador: Debe ser entrenado para que cuando se le presente una versión distorsionada ligeramente del patrón, pueda ser reconocida correctamente sin problemas. De la misma forma, la red debería presentar cierta inmunidad contra el ruido, esto es, debería ser capaz de recuperar una señal limpia de ambientes o canales ruidosos.

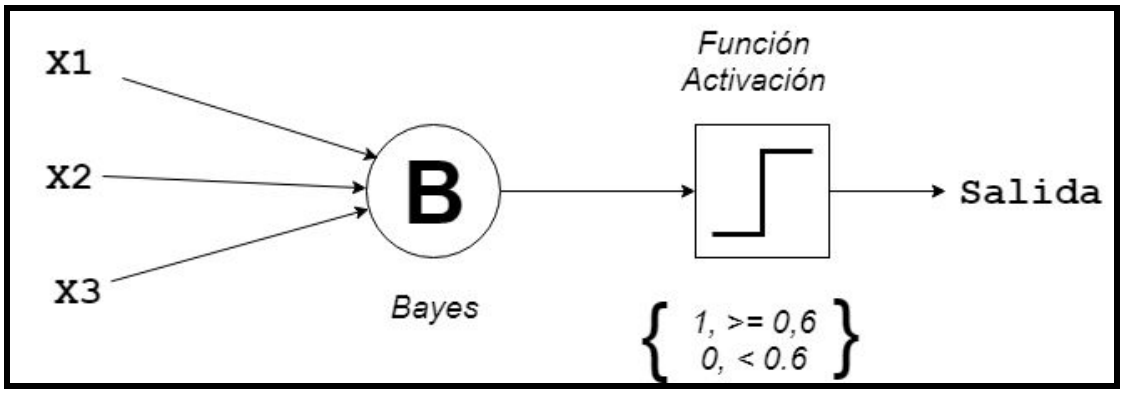
  Regressor:

  Es un modelo matemático usado para aproximar la relación de dependencia entre una variable dependiente Y, las variables independientes Xi y un término aleatorio. Genera un modelo matemático que puede apegarse al comportamiento de la función.

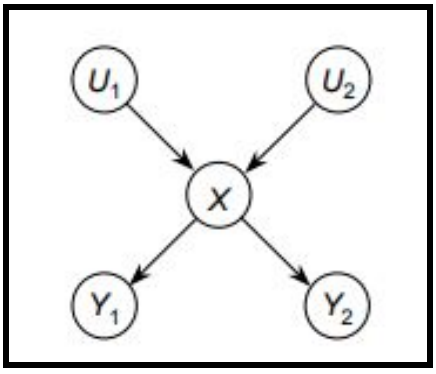
3 Bayesiano

Nuestro perceptrón trata de evaluar si se tiene gripa o no basados en 5 variables:

1. Zona de origen (alto medio o bajo riesgo de contagio). [U1]
2. Tipo sanguineo (alta o baja inmunidad). [U2]
3. Secreción nasal (positivo o negativo). [Y1]
4. Fiebre (presente o ausente). [Y2]
5. Gripe (positivo o negativo). [X]



Nuestro grafo es el siguiente:



In [1]:
```
#Probalididades riegso
przAlta = 0.10
przMedia = 0.10
przBaja = 0.8
#Porbabildiades de que tipo sanguineo
psAlta = 0.6
psBaja = 0.4
```

In [2]:
```
#Probabilidad  de gripa dependiente de tipo de sangre y zona de origen
# Tipo de sangre con mayor inmunidad en zonas
prsBajazAlta = 0.015
prsBajazMedia = 0.003
prsBajazBaja = 0.0003
# Tipo de sangre con menor inmunidad en zonas
prsAltazAlta = 0.022
prsAltazMedia = 0.012
prsAltazBaja = 0.0008
```

In [3]:
```
#Probabilidad de tener o no gripa dependiendo de moqueo
#Prob de gripa con Moqueo
pgsiMoqueo = 0.992
#Prob de gripa sin moqueo
pgnoMoqueo = 0.006
```

In [4]:
```
#Probabilidad de tener o no gripa dependiendo de fiebre
#Prob de gripa con fiebre
pgsiFiebre = 0.98
#Prob de gripa sin fiebre
pgnoFiebre = 0.017
```

In [6]:
```
#Probabilidad de tener gripa en una zona de alto riesgo con sangre menos inmunde, sin moqueo y con fiebre
# p = pszMedia * psBaja *p(gripa | zmedia y sBaja ) * p(sin moqueo|gripa) * p(siFiebre| gripa)
prGripaconzAltasBajanoMoqueoSiFiebre= przAlta*psBaja*0.12*0.008*pgsiFiebre
print(prGripaconzAltasBajanoMoqueoSiFiebre)
```

```
3.7632e-05
```

In [7]:
```
#Probabilidad de no tener gripa en zona media, menos inmunidad, sin  moqueo y con fiebre
#
prnoGripazMediasBajanoMoqueoSiFiebre = przMedia * psBaja * pgnoFiebre * 0.88 * 0.994
print(prnoGripazMediasBajanoMoqueoSiFiebre)
```

In [9]:
```
#Prob de estar en zona media,sangre menos inmune, sin moqueo y con fiebre
przMedianoMoqueosBajaSiFiebre = 0.0000376 + 0.0005948
print(przMedianoMoqueosBajaSiFiebre)
```

```
0.0006324000000000001
```

In [10]:
```
#Prib de teniendo fiebre, sin moqueo, sangre menos inmune y zona de medio riego tenga gripa
# p = (zMedia, sBaja, gripa,noMoqueo, siFiebre)/(zMedia, sBaja,noMoqueo, siFiebre)
p = 0.0000376/przMedianoMoqueosBajaSiFiebre
print(p)
```

```
0.059456040480708405
```

Conclusiones

- Las relaciones causales pueden desempeñar un papel o no, dependiendo de la interpretación que se le dé a los eventos planteados.

- Se debe conocer información sobre un experimento para poder modificar los valores de probabilidad y así lograr una consistencia en los datos.

- Al utilizar la función de costo puede utilizarse un algoritmo genérico, ya que al tener las variables de implementación su desarrollo se basa en procesos simultáneos y relacionados.

Bibliografía.

1. Julio Cesar Ponce Gallegos. (2014). Inteligencia Artificial 1a ed. (1).