Sistemas de Inteligencia Artificial

Trabajo práctico especial 4

Grupo 5
Julián E. Gutiérrez F. (51141)
Alexis Medvedeff (50066)
Javier Perez Cuñarro (49729)

ITBA
Primer Cuatrimestre 2014

Introducción

El objetivo de este trabajo es implementar un motor que ejecute un algoritmo genético que sea capaz de obtener los pesos de una red neuronal multicapa.

Se buscó obtener los pesos del perceptrón multicapa realizado para el *Trabajo Práctico Especial 2*, que estima una función escalar a partir de un conjunto de puntos graficados en la *figura 1*. Se optó por la arquitectura de red y la función de activación que resultaron más óptimas según los resultados que se obtuvieron mediante aprendizaje supervisado en dicho trabajo.

Configuración de la red elegida

Recordamos de nuestro informe anterior que las mejores tres redes obtenidas fueron las que se muestran en la *tabla 1*.

De esas tres optamos por quedarnos con la configuración de dos capas ocultas con 4 y 3 neuronas respectivamente (ver *figura 2*) ya que agiliza la computabilidad del algoritmo. Para esto volvimos a correr las pruebas con las distintas mejoras al algoritmo backpropagation y los resultados obtenidos se muestran en la *tabla 2* y los *gráficos 1 y 2*.

Se usó la función de activación tangente hiperbólica y se dividieron los datos de entrada en un 60% para la fase de entrenamiento y un 40% restante para la fase de testeo. A su vez se agregó la mejora de momentum y se ejecutó a lo largo de 1000 épocas, obteniendo un error cuadrático medio de 0.0031 y un porcentaje de predicción del 41.66% (*tabla 2*).

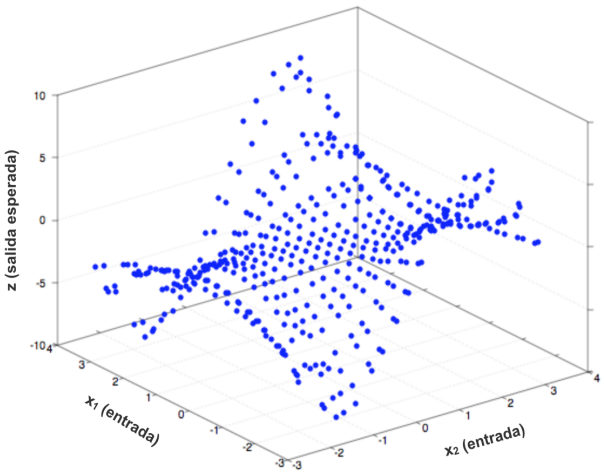

Figura 1: puntos de la función (samples8.txt)

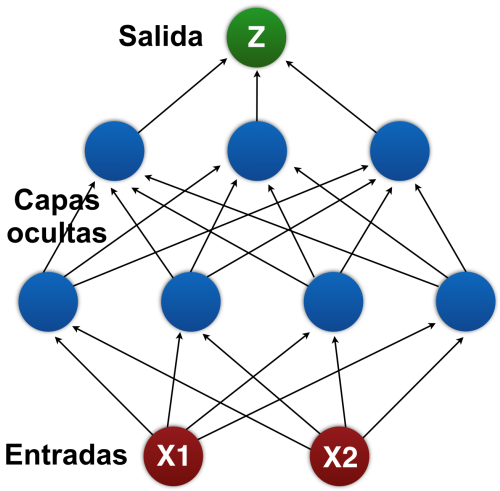

Figura 2: esquema de la red elegida

Implementación del motor

Se implementaron distintas variantes de operadores genéticos (cruce, mutación, backpropagation), criterios de selección y reemplazo (elite, ruleta, universal, Boltzmann, torneos, mixto), métodos de selección analizados en clase (método 1, 2, 3) y criterios de corte (máxima cantidad de generaciones, estructura, contenido, entorno al óptimo). Se compararon los resultados obtenidos con el motor usando distintas variantes entre sí y con los resultados obtenidos por aprendizaje supervisado.

El motor recibe una red pre entrenada una cantidad baja de épocas; en este caso se eligió 100. La razón por la cual se entrena es para guiar el comienzo del algoritmo. Si no fuera así, los pesos iniciales que tomaría el motor genético podría estar muy alejados de algo útil debido a la aleatoriedad. Todos los individuos iniciales de la población se crean de la misma forma e independientemente entre sí. Se generan tantas configuraciones de pesos que representan a las redes neuronales como individuos el algoritmo genético.

Representación de individuos

Para representar al individuo sin modificar la esencia de la estructura elegida para el algoritmo de backpropagation, se eligió transformar la matriz de pesos a un vector de números decimales (*figura 3*). De este modo logramos aplicar los operadores en forma más rápida que si utilizáramos las operaciones aritméticas equivalentes sobre la matriz. Llegado el punto en que necesitáramos la estructura anterior, realizamos el camino inverso pero esta vez con la ayuda de las dimensiones originales de la matriz, que almacenamos previamente.

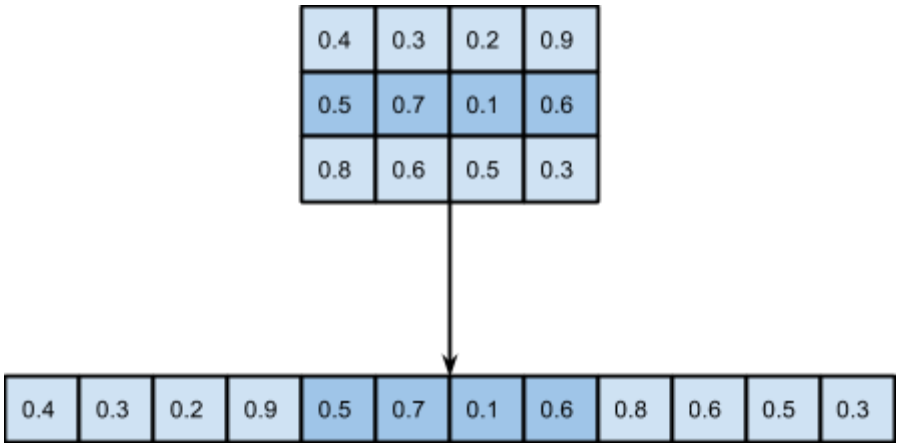

Figura 3: Esquema de la conversión de la matriz de pesos a un arreglo.

Fidelidad del individuo

- **Coherencia**: representa únicamente el dominio del problema, ya que ninguna representación podrá pertenecer a un conjunto externo al dominio.
- **Completitud**: es completa, ya que una solución al problema es una matriz de pesos y en consecuencia cualquiera de ellas es representable (dominio).

- **Localidad**: cada cambio en el arreglo representa un cambio en el peso de la conexión en la red neuronal que representa en la matriz (previo reconvertir el arreglo a matriz como ya se comentó).
- **Uniformidad**: si no lo fuera podríamos representar 2 individuos distintos con la misma cadena, y debido a su método de construcción esto no es posible. Si dos individuos tuvieran la misma cadena entonces serían el mismo.
- **Sencillez**: es sencillo ya que convertir una lista de matrices a un arreglo único es de complejidad O(n^2) por lo que lleva leer una matriz de n x n.

Elección de la población

Una de las primeras cosas que debimos hacer una vez implementado el motor genético fue seleccionar la cantidad de pobladores para evolucionar, en búsqueda de uno que “aprenda” mejor la superficie que representa la función que nos tocó. Para ello corrimos con número de pobladores entre los intervalos [2 50], [50 100] y [100 150]. Notamos que si el número de pobladores superaba los 100 aumentaba significativamente el tiempo de cómputo (dos órdenes de magnitud) sin aportar una diferencia apreciable en el error (no llegaba a un orden). Con un número menor a 50 no llegabamos a apreciar los beneficios del algoritmo genético. Ahora entre 50 y 100 nos encontrábamos en valores aceptables aunque no muy dispersos. Por lo tanto decidimos tomar 50 como valor representativo para lo que resta del informe, a riesgo de perder algo de precisión, pero reduciendo el tiempo de cómputo.

Eleccion del número de progenitores

Fijado el tamaño de la población avanzamos a seleccionar un número de progenitores adecuado. Para ello corrimos las pruebas que se muestran en la *tabla 3*, y nos quedamos con un número de progenitores de 10.

Backpropagation para entrenar a los descendientes

Cabe destacar que la población seleccionada, de 50 individuos, es entrenada durante 100 épocas por individuo. Este número no puede ser muy elevado ya que sino la genética no logrará aplicar y el error se estancará en el obtenido en el trabajo práctico anterior.

Luego, resta discutir cuántas épocas entrenar a los descendientes. Inicialmente corrimos el algoritmo sin backpropagation notando que la genética no mejoraba mucho el error del promedio inicial de la población. Por ello tomamos 100 épocas por descendiente al igual que en el entrenamiento inicial pero notamos que en pocas generaciones se estancaba el error en el mismo del trabajo anterior por lo que introducimos la probabilidad de aplicar o no backpropagation. Con esta modificación logramos que se estancara, pero ahora en un número más alto de generaciones. Finalmente bajamos el número a 30 generaciones manteniendo la probabilidad de aplicar el backpropagation con mejores resultados pero aún se llegaba a un estanco alrededor de las 500 generaciones. Podemos decir entonces que la probabilidad de aplicar backpropagation (p) debería decaer con el paso de las generaciones, por lo que

reemplazamos el valor fijo de p, por la función p/log(nroGeneracion/2) lo que nos disminuye el valor de p a la mitad en 25 generaciones (p = 0.4, para el ejemplo). Los resultados obtenidos para el error cuadrático medio y el porcentaje de aprendizaje se muestran en la *tabla 4***.**

Mutación y selección de las probabilidades

Contamos con dos mecanismos para mutar a los pobladores: la mutación clásica y la no uniforme. En el primer caso sólo uno de los alelos es modificado. En cambio en el segundo se evalúa una probabilidad contra un número pseudo aleatorio sobre cada alelo. En cualquier caso el valor de un alelo nunca se modifica más de un 10% de su valor original. Este "ruido" agregado puede ayudar a destrabar el error en la red, mientras que si se agrega demasiado ruido esa red tendrá "daño cerebral" y podría disminuir drásticamente el fitness del individuo. Los resultados obtenidos en la *tabla 5* y respaldados por la información teórica encontrada[1] nos dicen que la mejor opción es el método de mutación no uniforme con probabilidad de mutación de 0.1 (lo que hará que muten aproximadamente 1 de nuestros 10 descendientes) y 0.15 por alelo (alrededor de 5 alelos de los 31 que contiene la cadena).

Crossover y selección de las probabilidades

Para decidir cómo impacta la probabilidad de crossover contra la predicción de la red se fijaron los parámetros y se corrieron las pruebas que indica la *tabla 6***.** Notamos que en la medida en que la probabilidad aumenta la red se hace más precisa. Esto es porque la genética aumenta la diversidad de los individuos. Se toma 0.85 para que el cambio en la población no sea tan drástico de una generación a la otra.

Métodos de Crossover

Una vez fijada la probabilidad de crossover en 0.85 nos decidimos en comparar los distintos métodos de crossover. Las pruebas y parámetros elegidos se encuentran en la *tabla 7*. Vale destacar que cambiamos el método de selección por ruleta, para evitar que el error se estanque en mínimos locales. Finalmente optamos por el cruce uniforme. Este método introduce más aleatoriedad a la cruza y permite cambiar los alelos de distintas partes de los individuos, a diferencia de los otro tres que intercambian un intervalo, lo que modifica partes cercanas de matriz de pesos de la red neuronal representada.

Criterio de selección, método y criterio de reemplazo

Esta es la sección que más tiempo llevó en las pruebas debido a la cantidad de posibles combinaciones de parámetros. Logramos un mejor resultado con la combinación ruleta + elite con método 1. A esta le sigue Torneos P.+Torneos D. con método 3. con un 10% menos deprecisiónn La mejor alternativa fueron graficada en la figura 3 , de lo que se desprende que el mejor fitness tiende a alejarse del fitness promedio de la población.

[1] http://www.obitko.com/tutorials/genetic-algorithms/recommendations.php

Resultados[2]

Las configuraciones donde más se corrió backpropagation fueron las mejores en cuanto a aprendizaje siendo al mismo tiempo las peores en cuanto al tiempo de ejecución.

Para nuestro problema la mejor combinación obtenida sin contar las que aplican backpropagation en todo momento fue:

> Backpropagation con probabilidad con decrecimiento logarítmico (entrenando 30 épocas si corresponde), 50 generaciones, con selección por torneos probabilística, método de reemplazo 2, seleccionando 10 progenitores en el caso de los métodos de reemplazo 1 y 3. Mutación no uniforme con probabilidad de mutar 0.1 y probabilidad en cada alelo de 0.15.

Con esta configuración se obtuvo una predicción en testeo de 51.1494%.

Conclusiones

La selección de la función de fitness puede hacer que distintos métodos se comporten de distinta forma debido a que según esta función se interpretará la aptitud del individuo.
Debido a la gran cantidad de parámetros modificables resulta muy costoso temporalmente explorar todas las combinaciones posibles por lo que consideramos muy probable encontrar otras configuraciones que produzcan mejores resultados que los mostrados.
Aún en el caso de poder probar todas o gran cantidad de las posibles configuraciones, no es posible encontrar una óptima combinación recomendable para todas las situaciones ya que depende altamente de la condiciones particulares del problema a estudiar.
Los métodos de selección y reemplazo elitistas aceleran el acercamiento a fitness mayores pero reducen la diversidad en la población. El efecto de esto podría ser caer en mínimos locales lo cual no permitiría llegar a soluciones más óptimas alcanzables por otros caminos. Combinar el uso de selección y reemplazo elitistas con corte por contenido permite detener el algoritmo genético al repetirse el mejor fitness (considerado estancamiento).
Sería interesante lograr un balance entre elitismo y diversidad, lo que haría que la evolución se prolongue a más largo plazo y que se cubra el dominio de la función con mayor precisión.
El algoritmo de backpropagation mejora notablemente el fitness de la población. Es preciso utilizarlo en forma prudente para mantenerse en un tiempo de cómputo aceptable.
Es destacable cómo la genética nos ayuda a obtener una red que aprende la función un 10% más (en promedio) que la del trabajo práctico anterior en un tiempo de cómputo similar, o hasta incluso un poco menor.

[2] Obtenidos con procesador Intel Core i5 de 2.6 Ghz bajo Mac OSX 10.9.2.

Anexo A

Tablas de resultados

Arquitectura	**Error cuadrático medio en testeo**	**Predicción en testeo**
[20]	0.9456	36.7647%
[4 3]	0.2237	39.1636%
[10 20 30]	0.6844	40.2934%

Tabla 1: comparación del error de la red neuronal en relación a su arquitectura. Sin mejoras, 60% de entrenamiento, tangente hiperbólica, 500 épocas.

Mejoras	**Error cuadrático medio en testeo**	**Aprendizaje en entrenamiento**	**Predicción en testeo**
Ninguna	0.0100	24.1636%	29.6512%
Momentum	0.0031	46.1538%	41.6667%
ETA Adaptativo	0.0100	19.7026%	20.3488%
Momentum y ETA adaptativo	0.0113	18.5874%	23.8372%

Tabla 2: comparación de las mejoras a backpropagation. Configuración de las neuronas en las capas internas [4 3], 60% de datos para entrenamiento, función de activación tangente hiperbólica, 1000 épocas.

Número de progenitores	Error cuadrático medio en testeo	Predicción en testeo
40	0.9467	0%
26	0.8856	0%
16	0.2707	3.570%
10	0.1026	10.9890%

Tabla 3: análisis del número de progenitores. Parámetros: sin backpropagation, 500 generaciones, classic crossover con p=0.4, elite selection, método de reemplazo 2, mutación no uniforme con pMutar=0.1 y pAlelo=0.15.

Probabilidad de aplicar backpropagation	#épocas	#generaciones	Error cuadrático medio en testeo	Predicción en testeo
0	n/a	200	0.2380	3.2967%
1	100	50	0.0006	66.4835% (*)
1	30	50	0.0052	34.6154% (*)
0.4	30	50	0.0078	14.8352%
0.4*log(nroGeneracion/2)	30	100	0.0059	29.3103%

Tabla 4: análisis de la probabilidad de aplicar backpropagation y número de épocas en caso de que se aplique. Parámetros: progenitores=10, classic crossover con p=0.4, elite selection, método de reemplazo 2, mutación no uniforme con pMutar=0.1 y pAlelo=0.15. **(*) El tiempo de ejecución superó las 3 horas, alrededor de 6 veces superior al tiempo de los otros casos.**

Método de mutación	Probabilidad de mutación	Probabilidad de mutación por alelo	Error cuadrático medio en testeo	Predicción en testeo
Uniforme	1	n/a	0.0071	19.7802%
Uniforme	0.5	n/a	0.0067	30.7033%
No uniforme	0.2	0.3	0.0090	23.0769%
No uniforme	0.1	0.15	0.0060	35.1648%

Tabla 5: análisis del método de mutación vs las distintas probabilidades. Backpropagation p= 0.4*log(nroGeneracion/2), 30 épocas si corresponde, 50 generaciones, classic crossover con p=0.4, elite selection, método de reemplazo 2, 10 progenitores.

Probabilidad de crossover	Error cuadrático medio en testeo	Predicción en testeo
1	0.0020	36.7816%
0.4	0.0105	25.2874%
0.85	0.0014	51.1494%

Tabla 6: análisis de la probabilidad de crossover de dos puntos vs predicción en testeo. Parámetros: backpropagation con p= 0.4*log(nroGeneracion/2), 30 épocas si corresponde, 50 generaciones, **torneos probabilísticos**, método de reemplazo 2, 10 progenitores, mutación no uniforme con pMutar=0.1 y pAlelo=0.15.

Operador genético	Fitness	#generaciones	Corte	Error cuadrático medio en testeo	Predicción en testeo
Cruce clasico	148.39	29	Contenido	0.0077	18.6813%
Cruce dos puntos	32	57	Contenido	0.0106	6.0440%
Cruce uniforme	49	5	Contenido	0.0103	28.5714%
Cruce anular	50	6	Contenido	0.0100	26.3736%

Tabla 7: análisis de los distintos operadores genéticos. Parámetros: progenitores=10, backpropagation con p= 0.4*log(nroGeneracion/2), crossover con p = 0.85 , 30 épocas si corresponde, ruleta, método de reemplazo 2, mutación no uniforme con pMutar=0.1 y pAlelo=0.15.

Criterio de selección	Criterio de reemplazo	Método	Generación	Corte	Fitness
Ruleta	Torneos prob.	1	3	Contenido	91.5420
Ruleta	Elite	1	30	Generaciones	**48.5(*)**
Torneos prob.	Elite	1	3	Contenido	100.3145
Torneos determ.	Elite	3	20	Generaciones	1040
Universal	Ruleta	1	3	Contenido	97.746
Torneos prob.	Torneos determ.	3	20	Generaciones	**41.43(*)**
Elite	Universal	1	3	Contenido	92.691
Universal	Torneos prob.	3	20	Generaciones	**31.52(*)**
Ruleta	Torneos prob	1	10	Generaciones	**24.78 (*)**
Elite	Ruleta	3	20	Contenido	**40.23(*)**

Tabla 8: análisis de los distintos criterios de selección y reemplazo. Parámetros: progenitores=10, backpropagation con p= 0.4*log(nroGeneracion/2), crossover uniforme con p = 0.85, 30 épocas si corresponde, mutación no uniforme con pMutar=0.1 y pAlelo=0.15.

(*) calculado con la función de fitness f(i) = learning_rate con datos de testeo +entrenamiento

Método	**Error cuadrático medio en entrenamiento**	**Error cuadrático medio en testeo**	**Porcentaje de aciertos en entrenamiento**	**Porcentaje de aciertos en testeo**
Aprendizaje supervisado	0.0024	0.0031	46.1538%	41.6667%
Algoritmos genéticos (mejor individuo)	0.0018	0.0027	52.4323%	49.5218%

Tabla 9: comparación de los resultados obtenidos mediante aprendizaje supervisado (en el trabajo práctico especial 2) con los resultados de los algoritmos genéticos. Configuración de las neuronas en las capas internas [4 3], 60% de datos para entrenamiento, función de activación tangente hiperbólica.

Parametros:

-Supervisado: 1000 épocas

-Genetico: Progenitores=10, backpropagation p= 0.4*log(nroGeneracion/2),crossover p = 0.85 , 30 épocas si corresponde, elite selection, método de reemplazo 1, Mutación No uniforme pMutar=0.1, pAlelo=0.15. Criterio de reemplazo: elite, Criterio de seleccion: ruleta metodo de seleccion:1 nro de generaciones:30, Criterio de corte: nro de generaciones

Anexo B

Gráficos

Para configuración: [4 3], momentum activado, 60% entrenamiento, tangente hiperbólica, 1000 épocas máx.

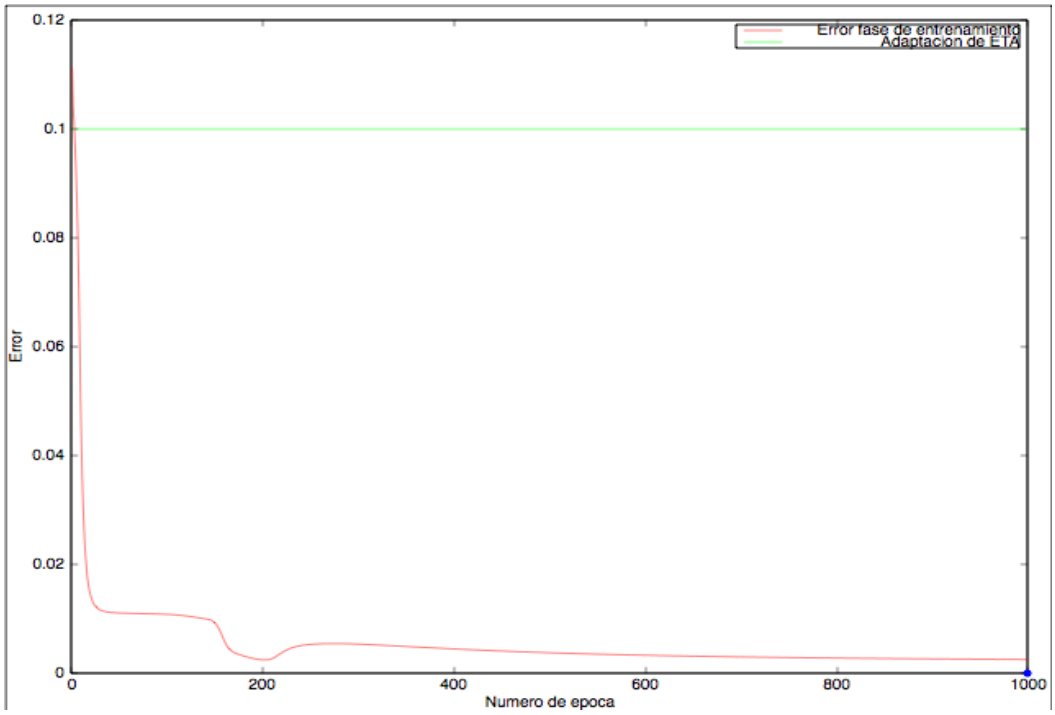

Gráfico 1: error cuadrático por época

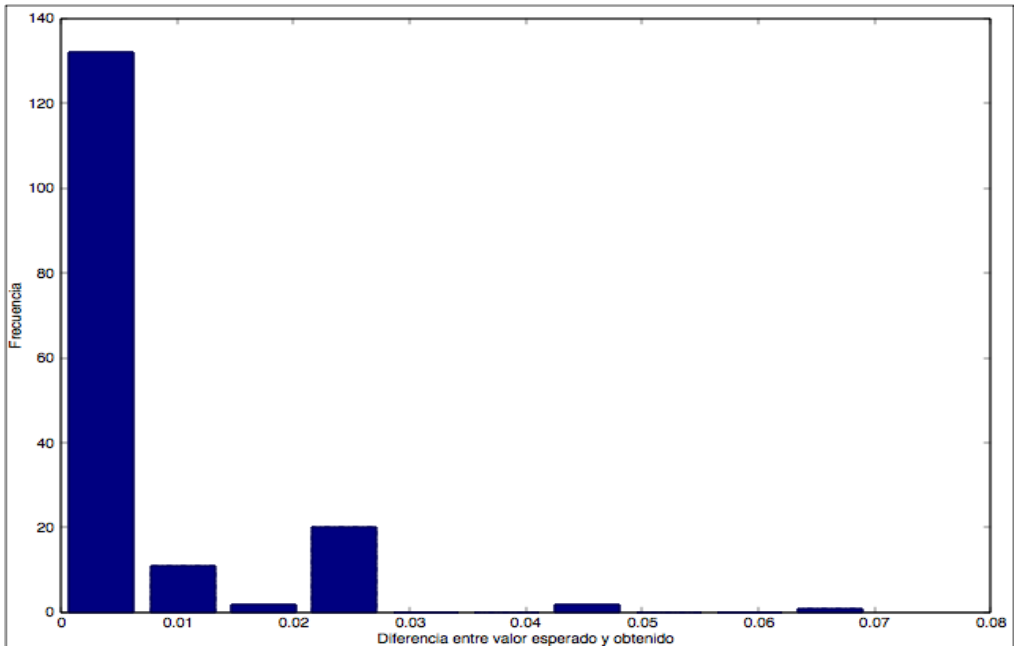

Gráfico 2: diferencia valor calculado y esperado