1. Introducción

Un algoritmo genético es un algoritmo utilizado para optimizar la solución a un problema. Para realizar esto, utiliza un conjunto inicial aleatorio de soluciones (llamado población inicial), las cuales poseen un grado de adaptación (fitness) que denota qué tan buena es.

La implementación de un algoritmo genético está dada de modo tal que se realice un aprendizaje automático y existen muchos casos en donde un algoritmo genético puede ayudar a resolver problemas que serían muy difíciles o hasta imposibles de resolver de forma explícita.

Considerando redes neuronales artificiales que buscan resolver la serie temporal `y = f(x(t-1), x(t-2), ...)` con un paso temporal `t`, se busca mejorar los resultados de esta red mediante un algoritmo genético.

2. Marco Teórico

En un algoritmo genético cada solución (o individuo) está codificado mediante un cromosoma. Dicho cromosoma está compuesto por un conjunto de elementos denominados alelos, cuya posición (y por ende su significado) es denominado locus.

La idea de un algoritmo genético es la de reemplazar la población actual a través de generaciones por una población nueva con el objetivo de que la nueva generación posea mejores soluciones.

Cada solución posee un grado de adaptación (para lo cual se cuenta con una función de Fitness) que se utiliza para evaluar qué tan buena es.

Para generar nuevos individuos se pueden intercambiar alelos entre dos indivudos conocidos (entrecruzamiento) o modificar algún alelo por un valor aleatorio (mutación). Mediante distintos criterios de selección y reemplazo se conforma una nueva población. Es importante destacar que algunos individuos de la población pueden pasar a la siguiente generación sin ser modificados.

Existen diversas maneras de finalizar la ejecución de un algoritmo genético. Es posible definir el corte de dicha ejecución mediante una cantidad máxima de generaciones, al alcanzar un grado de adaptación menor a una cota definida, al detectar que la población no está siendo modificada significativamente a lo largo de cierta cantidad de generaciones o al haber alcanzado una solución que no sea posible mejorar.

3. Desarrollo

El problema a desarrollar es el de tratar de encontrar la mejor red neuronal que solucione el problema de aproximación de una serie temporal.

Para representar nuestro problema, cada individuo estaba conformado por la matriz de pesos correspondiente a una red neuronal, mientras que el valor de cada locus de su codificación equivalía a un peso particular.

La función de adaptación para este problema era el Error Cuadrático Medio obtenido al evaluar la red.

El error cuadrático medio es cálculado como se puede ver en la Figura 1 en donde la resta equivale a la resta entre el valor esperado y el obtenido para cada patrón.

A las operaciones habituales de un algoritmo genético (entrecruzamiento y mutación) se agregó la posibilidad de aplicar una cantidad de pasos del algoritmo de backpropagation a un individuo.

Para poder evaluar el problema se prefijaron ciertos elementos de la red neuronal en base a una buena ejecución de la misma. Estos fueron:

- Estructura de la red: [2 9 8 1]
- Momentum: 0.3
- Parametros Adaptativos: 0.1 para los incrementos del Learning Rate y 0.001 para los decrementos, 0.001 es considerado como la cota para evaluar si una iteración es buena, mala o neutral y 2 pasos buenos son considerados para aumentar el Learning Rate
- Learning Rate: 0.4
- Rollbacks: Activados
- Función de activación: Sigmoidea lógica
- Beta: 1
- Patrones utilizados para el aprendizaje: 300

Con esta configuración originalmente se pudo obtener un error de 0.0003756

Dentro de nuestra implementación existe la posibilidad de configurar los siguientes parámetros para el algoritmo genético:

- Cantidad de individuos en la población
- Criterios de corte: Se puede definir una cantidad máxima de generaciones, una cota mínima para el grado de adaptación de los individuos, un porcentaje para el cual se considera que la población no ha sido modificada siginificativamente entre generaciones y una cantidad de generaciones en las cuales se espera que las soluciones hayan mejorado
- Mutaciones a los individuos: Se cuenta con una probabilidad de mutación donde la misma puede ser de dos tipos: mutando con una misma probabilidad cada locus o mutando algún locus elegido al azar con otra probabilidad. Así mismo, dichas probabilidades pueden ser constantes durante las generaciones (clásica) o reducirse con el paso de las mismas (no uniforme)
- Entrecruzamientos entre individuos: Se cuenta con una probabilidad de entrecruzamiento, pudiéndose elegir el método de entrecruzamiento entre individuos entre: cruce de un punto, cruce de dos puntos, cruce uniforme parametrizado y anular
- Backpropagation: Es posible configurar una probabilidad de ejecución del algoritmo de backpropagation sobre un individuo. Así mismo, es posible configurar cuántas iteraciones del mismo se realizarán
- Métodos de reemplazo: Es posible configurar cuál de los tres métodos de reemplazo dados por la cátedra será utilizado. Para los métodos que lo necesiten, también se puede configurar la brecha generacional
- Métodos de selección: Es posible configurar cuál será el método de selección a utilizar entre ruleta, universal, elitismo, ranking, Boltzmann, torneos, mixto 1 (elitismo y ruleta) y mixto 2 (elitismo y universal).

4. Resultados

Para evaluar las distintas configuraciones del algoritmo, se evaluó cada caracterísitica por separado, prefijando valores por defecto para las otras configuraciones.

La configuración por defecto considera como métodos de selección a ruleta. Para los métodos mixtos, por defecto, se toma una proporción del 50% por método. Por su parte, para el reemplazo, es utilizado el primer método. Para aquellos casos en que se utilizan los otros dos métodos de reemplazo, por defecto, se toma 0.8 como brecha generacional. El método de entrecruzamiento utilizado es el uniforme con una probabilidad de entrecruzamiento de 0.75. En estos casos, donde el método sea uniforme, la probabiladad de entrecruzamiento por locus es de 0.05. La probabilidad de mutación es fijada en 0.1, mutando sólo un locus por individuo. En los casos donde cambia la configuracion para que se evalue cada locus por separado, la probabilidad de mutación por locus es de 0.03.

Respecto a la finalización del algoritmo, se tomaron varios criterios de cortes: al alcanzar un número máximo de 17 generaciones, al conseguir un nivel de adaptación menor a 0.001, en el caso en el cual un 80% o más de la población no haya sido modificado entre las últimas dos generaciones, y en el caso en el cual la mejor solución no se haya podido mejorar en las últimas 10 generaciones.

Además, por defecto, se considera una población de 60 individuos.

Para cada combinación de parámetros probada se hizo un conjunto de 3 corridas. Para cada conjunto se midieron:

- avgMeanError: promedio entre cada corrida del conjunto de los errores medios de las poblaciones finales
- avgBestError: promedio entre cada corrida del conjunto de los menores errores de las poblaciones finales
- bestMeanError: menor entre cada corrida del conjunto de los errores medios de las poblaciones finales
- bestBestError: menor entre cada corrida del conjunto de los menores errores de las poblaciones finales

4.1 Evaluación de métodos de entrecruzamiento

En la Tabla 1 se pueden observar las mejores soluciones y la solución promedio para los distintos métodos de entrecruzamiento.

4.2 Evaluación de métodos de mutación

En la Tabla 2 se pueden observar las mejores soluciones y la solución promedio para los distintos métodos de mutación.

4.3 Evaluación de métodos de selección

En la Tabla 3 se pueden observar las mejores soluciones y la solución promedio para los distintos métodos de selección. Para los métodos mixtos, el parámetro `a` indica qué porcentaje de la selección se realiza

mediante el método elite.

4.4 Evaluación de métodos de reemplazo

En la Tabla 4 se pueden observar las mejores soluciones y la solución promedio para los distintos métodos de reemplazo utilizando además diversos métodos de selección. En los casos de los métodos 2 y 3, el posible utilizar distintos métodos de selección para la fase de selección y para la fase de reemplazo.

4.5 Evaluación de backpropagation

En la Tabla 5 se pueden observar las mejores soluciones y la solución promedio para evaluaciones con probabilidad por defecto de 0.1 de que operar con el algoritmo de backpropagation, como así también para evaluaciones en las que no se utilice dicho algoritmo.

Se debe mencionar, además, que el método de selección utilizado para estas evaluacioes ha sido elitismo, mientras que para el entrecruzamiento, la opción a utilizar ha sido anular. Por otra parte, para la mutación se ha empleado aquel tipo de mutación en el cual cada locus puede ser mutado con misma probabiladad. Dicha mutación además era no uniforme.

4.6 Comparación entre la red neuronal y el algoritmo genético

A modo de contrastar ambas estrategias, se han corrido ejecuciones de la red neuronal como así también del algoritmo genético para el problema a tratar. En la Tabla 6 se pueden ver los resultados obtenidos de dichas corridas.

Es importante destacar que en estas ejecuciones el criterio de finalización por una máxima cantidad de 17 generaciones ha sido modificado elevando este número a 37.

5. Conclusiones

El mejor individuo conseguido con el algoritmo genético tuvo un error cuadrático medio durante el testing (para 300 inputs) de 0.00028 y para todos los inputs, de 0.00039. Esto es ligeramente superior a los 0.00026, 0.00028 (respectivamente) de la RNA. Sin embargo notamos que el AG es mucho más consistente en sus corridas. Esto es, en cerca del 70% de las ejecuciones de éste con los mejores parámetros encontrados se llega a un individuo cercano al óptimo. Contrariamente, con la RNA esto pasa sólo en aproximadamente un 10% de los casos.

El mejor individuo fue hallado con los siguientes parámetros:

```
pMutate = 0.003; % single locus mutation probability
pBack = 0.1; % back propagation probability
pCross = 0.75; % crossing probability
p = 0.05; % uniform crossover probability per locus
c = 0.95; % mutation reduction ratio
N = 60; % population size
nonUniformMutation = true;
maxEpochs = 80;
rollback = true;
momentum = 0.3;
eta = 0.4;
etaEps = 0.001;
etaInc = 0.1;
etaDec = 0.001;
etaSteps = 2;
g = #algorithm.functions.sigmoidLog;
dg = #algorithm.functions.DsigmoidLog;
selection = #algorithm.ga.selection.elite;
crossover = #algorithm.ga.crossover.annular;
mutation = #algorithm.ga.mutation.locus;
replacement = #algorithm.ga.replacement.first;
```

Como se puede observar en las tablas, eliminar la operación de backpropagation hace que la mejora sea extremadamente lenta, por lo cual la consideramos una parte fundamental de nuestro algoritmo.

Por último, otra observación clave que hicimos fue el uso de la estrategia de elitismo ya que sin ésta el algoritmo se comporta casi como una búsqueda aleatoria.