Universidad Simón Bolívar
Inteligencia Artificial II
Proyecto 2
GABIL

Grupo 05

Stefano De Colli	Juliana León	Karen Troiano
09-10203	08-10608	09-10855

Noviembre, 2014

Resumen

Se utilizó el lenguaje **Python** y la librería **Pyevolve** para construir un algoritmo genético (GA) que sea capaz de clasificar las tres clases de la flor Iris *(Setosa, Versicolour, Virginica)*. Los datos de las clases de flores cuentan con cuatro *(4)* atributos reales que fueron llevados a un sistema de strings de ceros *(0s)* y unos *(1s)*.

Se probaron dos 2 versiones de la función de selección de padres y dos 2 versiones de la función de selección sobrevivientes. Una de las funciones de selección es **rueda de ruleta** y la otra es **selección del mejor individuo**. Para las funciones de fitness se prueba el tradicional *(fitness en gabil.py)* y una función de fitness que controla el tamaño del individuo *(fitness_by_side en gabil.py).*

A manera de conclusión, lo más relevante que se puede afirmar que entre más chico sea el set de entrenamiento, más altas deben ser las tasas de mutación y cruce.

Descripción del uso de la librería: Pyevolve

Pyevolve fue escrito en Python puro, es un API fácil de usar y es extensible, es decir, el usuario puede crear nuevas representaciones, operadores genéticos como cruce, mutación y etc. Esenciales para la adaptación que debimos realizar para implementar GABIL, dentro de la librería existían clases de mutación y cruce las cuales fueron alteradas en el archivo gabil.py dos funciones: **crossover_gabil** y **G1DBinaryStringMutatorFlip_GABIL** para cruce y mutación a utilizar en nuestro clasificador.

También se utiliza Pyevolve dado a las características comunes de funciones y aprendizaje que se utilizó en el curso de Inteligencia Artificial II, en el período de Septiembre-Diciembre del 2014, implementando las características más comunes como lo selectores: ruleta, el mejor de la población, la clasificación y uniforme.

Descripción del AG y parámetros bases utilizados

La siguiente tabla representa la distribución de los rangos para la representación de los cuatro (4) atributos a determinar.

Sepal Length	Sepal Width	Petal Length	Petal Width
5 bits	**6 bits**	**8 bits**	**11 bits**
[4.0 , 5.0): 10000 **[5.0 , 5.5)**: 01000 **[5.5 , 6.0)**: 00100 **[6.0 , 6.5)**: 00010 **[6.5 , ∞)**: 00001	**[2.0 , 2.5)**: 100000 **[2.5 , 2.8)**: 010000 **[2.8 , 3.0)**: 001000 **[3.0 , 3.25)**: 000100 **[3.25 , 3.75)**: 000010 **[3.75 , ∞)**: 000001	**[1.0 , 1.25)**: 10000000 **[1.25 , 1.5)**: 01000000 **[1.5 , 3.0)**: 00100000 **[3.0 , 4.0)**: 00010000 **[4.0 , 4.5)**: 00001000 **[4.5, 5.0)**: 00000100 **[5.0 , 5.75)**: 00000010 **[5.75 , ∞)**: 00000001	**[0.0 , 0.15)**: 10000000000 **[0.15 , 0.3)**: 01000000000 **[0.3 , 0.50)**: 00100000000 **[0.5 , 1.00)**: 00010000000 **[1.0 , 1.25)**: 00001000000 **[1.25, 1.35)**: 00000100000 **[1.35 , 1.5)**: 00000010000 **[1.5 , 1.75)**: 00000001000 **[1.75 , 2.0)**: 00000000100 **[2.0 , 2.35)**: 00000000010 **[2.35 , ∞)**: 00000000001

Tabla 1: Distribución de los atributos de la flor Iris.

Para la clasificación de la Iris Setosa, Versicolour o Virginica, se utilizaron dos (2) bits de representación donde el Iris Setosa es 00, Iris Versicolour es 01 y la Iris Virginica es 10.

Es decir, el individuo: 5.1 3.5 1.4 0.2 1 es para el algoritmo genético el individuo se transforma en: 0100000001001000000100000000000.

Para la implementación del cruce se reciben dos individuos denominados “Mom” y “Dad” *(denominación de la librería de Pyevolve)* y se retornan dos individuos denominados “Brother” y “Sister” *(denominación de la librería de Pyevolve)* los cuales son el producto del cruce de “Mom” y “Dad”, donde se buscan dos puntos de corte en el individuo más largo *(que denominamos Mom)*, luego utilizamos una función mod para ubicar en qué posición de la regla de la madre te encuentras, luego eliges dos reglas random en el padre y modificas el número de bits que se obtienen de la ubicación de la madre.

Para la implementación de la mutación se utilizó la función de la librería de Pyevolve: **G1DBinaryStringMutatorFlip**; el cual modifica al menos un bit del individuo recibido, cambiando un 0 por 1 y viceversa. Esta función fue modificada y llamada **G1DBinaryStringMutatorFlip_GABIL** para que funcionara con la implementación de los individuos que estábamos utilizando.

Para la implementación, de las funciones del fitness se hizo la clásica la cual consiste en elevar al cuadrado la cantidad de que se logró un match que cumplen con una regla del training set. En la otra, a la misma cantidad se le suma la inversa del tamaño del individuo para desfavorecer si el tamaño es muy grande.

Por último, para la implementación de los selectores se utilizaron las funciones de la libería de Pyevolve.

Descripción y análisis de los experimentos realizados

NOTA *Cabe acotar que por la aleatoriedad de las funciones de mutación, cruce y selección los mejores individuos lo resultados expuestos funcionan un alto porcentaje de las veces que se corre el algoritmo, pero podría no retornar los mismos resultados expuestos aquí presentes.*

Para responder estas preguntas, se utilizaron varias una tasas de mutación y cruce del 10% hasta el 100% con 6, 9, 45 y 105 individuos de entrenamiento y prueba con los 150 individuos y verificando la clasificación para cada caso y los siguientes cuadros se muestran los mejores resultados con las mejores combinaciones entre tasas de mutación y cruce:

Para ***105 individuos*** en el set de entrenamiento, tenemos que las tasas mínimas de mutación y cruce se podría utilizar pueden ser de entre 30% a 50% para clasificar al menos un 1% de los datos correctamente, si se entrena con las siguientes combinaciones, tenemos:

Algoritmo de selección Ruleta **Función de fitness** Normal **Mejor resultado**	**Algoritmo de selección** Ruleta **Función de fitness** Por tamaño **Mejor resultado**	**Algoritmo de selección** Mejor individuo **Función de fitness** Normal **Mejor resultado**	**Algoritmo de selección** Mejor individuo **Función de fitness** Por tamaño **Mejor resultado**
Mutación 0.8	**Mutación** 0.8	**Mutación** 0.8	**Mutación** 0.7
Cruce 0.8	**Cruce** 0.8	**Cruce** 0.8	**Cruce** 0.7
Fitness 2.0	**Fitness** 22050.0	**Fitness** 2.0	**Fitness** 22050.0
Acertadas 100%	**Acertadas** 100%	**Acertadas** 100%	**Acertadas** 100%

Tabla 2: Datos de resultados para la prueba con 105 individuos en el set de entranamiento.

Para ***45 individuos*** en el set de entrenamiento, tenemos que las tasas mínimas de mutación y cruce se podría utilizar pueden ser de entre 40% a 50%, para clasificar al menos un 1% de los datos correctamente, si se entrena con las siguientes combinaciones, tenemos:

Algoritmo de selección Ruleta **Función de fitness** Normal **Mejor resultado**	**Algoritmo de selección** Ruleta **Función de fitness** Por tamaño **Mejor resultado**	**Algoritmo de selección** Mejor individuo **Función de fitness** Normal **Mejor resultado**	**Algoritmo de selección** Mejor individuo **Función de fitness** Por tamaño **Mejor resultado**
Mutación 0.8	**Mutación** 0.8	**Mutación** 0.7	**Mutación** 0.7
Cruce 0.8	**Cruce** 0.8	**Cruce** 0.7	**Cruce** 0.6
Fitness 2.0	**Fitness** 3499.44	**Fitness** 2.0779	**Fitness** 4050.0
Acertadas 100%	**Acertadas** 100%	**Acertadas** 100%	**Acertadas** 100%

Tabla 3: Datos de resultados para la prueba con 45 individuos en el set de entrenamiento.

Para ***9 individuos*** en el set de entrenamiento, tenemos que las tasas mínimas de mutación y cruce se podría utilizar pueden ser de entre 70% a 85%, para clasificar al menos un 1% de los datos correctamente , si se entrena con las siguientes combinaciones, tenemos:

Algoritmo de selección Ruleta **Función de fitness** Normal **Mejor resultado** **Mutación** 0.95 **Cruce** 0.95 **Fitness** 2.0 **Acertadas** 100%	**Algoritmo de selección** Ruleta **Función de fitness** Por tamaño **Mejor resultado** **Mutación** 0.95 **Cruce** 0.95 **Fitness** 162.0 **Acertadas** 100%	**Algoritmo de selección** Mejor individuo **Función de fitness** Normal **Mejor resultado** **Mutación** 0.95 **Cruce** 0.95 **Fitness** 2.0 **Acertadas** 100%	**Algoritmo de selección** Mejor individuo **Función de fitness** Por tamaño **Mejor resultado** **Mutación** 0.85 **Cruce** 0.95 **Fitness** 162.0 **Acertadas** 100%

Tabla 4: Datos de resultados para la prueba con 9 individuos en el set de entrenamiento.

Para ***6 individuos*** en el set de entrenamiento, tenemos que las tasas mínimas de mutación y cruce se podría utilizar pueden ser de entre 85% a 95%, para clasificar al menos un 1% de los datos correctamente, si se entrena con las siguientes combinaciones, tenemos:

Algoritmo de selección Ruleta **Función de fitness** Normal **Mejor resultado** **Mutación** 1.0 **Cruce** 1.0 **Fitness** 2.0 **Acertadas** 100%	**Algoritmo de selección** Ruleta **Función de fitness** Por tamaño **Mejor resultado** **Mutación** 1 **Cruce** 1 **Fitness** 1.913 **Acertadas** 100%	**Algoritmo de selección** Mejor individuo **Función de fitness** Normal **Mejor resultado** **Mutación** 0.95 **Cruce** 0.95 **Fitness** 2.0 **Acertadas** 100%	**Algoritmo de selección** Mejor individuo **Función de fitness** Por tamaño **Mejor resultado** **Mutación** 0.85 **Cruce** 0.95 **Fitness** 78.48 **Acertadas** 100%

Tabla 5: Datos de resultados para la prueba con 6 individuos en el set de entrenamiento.

En conclusión, se puede apreciar que entre más bajo sea el porcentaje de individuos en el set de entrenamiento más altas deberán ser las tasas de mutación y cruce, para lograr una evolución más variada y poder alcanzar una clasificación óptima de los 3 tipos de Iris Setosa

Preguntas

Para responder estas preguntas, se utilizaron varias una tasas de mutación y cruce del 10% hasta el 100% con 9 individuos de entrenamiento y prueba con los 150 individuos y verificando la clasificación:

¿Cuál es la mejor configuración de su algoritmo genético para clasificar los datos estudiados?

La mejor configuración fue un 100% de tasa de mutación y cruce y el algoritmo de selección es el del mejor individuo de la población para ese momento y para el fitness fue el fitness que penaliza por tamaño.

¿Cuál es el mejor conjunto de reglas hallado por el algoritmo genético? Considerando el número de ejemplos clasificados correcta e incorrectamente

El mejor individuo obtenido fue:

Longitud:	4507
Fitness:	162.0
Acertadas:	150.0
Set de prueba:	150.0
Porcentaje acertadas:	100%

Se considera como el mejor dado que es el de tamaño más chico que brinda y clasifica mejor los 150 individuos con sólo 9 individuos de referencia en el set de entrenamiento.

Describa la función de fitness utilizada. ¿Es útil incluir en la función de fitness un factor de penalización a clasificadores muy grandes?

Para la implementación, de las funciones del fitness se hizo la clásica la cual consiste en elevar al cuadrado la cantidad de que se logró un match que cumplen con una regla del training set. En la otra, a la misma cantidad se le suma la inversa del tamaño del individuo para desfavorecer si el tamaño es muy grande.

A nivel de desempeño la función de fitness con factor de penalización a clasificadores muy grandes arrojó mejores resultados que sólo determinar el porcentaje de aciertos.