UNIVERSIDAD DE CHILE

Universidad de Chile

Facultad de Ciencias Físicas y Matemáticas

Departamento de Ciencias de la Computación

CC5114

Redes Neuronales y Programación Genética

Reporte Tarea 2

Autores:	Felipe Lizama Rodríguez
Profesor:	Alexandre Bergel
Auxiliar:	Juan-Pablo Silva
Ayudantes:	Gabriel Chandía
	Alonso Reyes

Fecha de entrega: 3 de diciembre de 2018

Santiago, Chile

Índice

1. Introducción

El problema escogido es el conocido Traveling Salesman Person (TSP), se utilizó algoritmos genéticos para encontrar una aproximación a la solución del problema.
El problema consiste en lo siguiente:
Dada una cantidad de ciudades fijas en un mapa en 2D, se desea pasar por todas las ciudades sin repetir ninguna excepto la primera ciudad. Es decir, si lo analizamos como un grafo, se busca un camino hamiltoniano.
Para la solución presentada en este reporte, se asume que desde cada ciudad se puede llegar a cualquiera de las otras siguiendo una línea recta y la distancia entre dos ciudades está dada por su distancia euclideana.

El link al código en github es:
https://github.com/flizamar/tarea2RedesNeuronales.git

1.1. Hipótesis

Se pretende corroborar con datos experimentales que con algoritmos genéticos es posible llegar a la solución exacta o en su defecto, a una buena solución aproximada.

2. Ejecución de los programas

Para correr la tarea se necesita tener las librerías: random, itertools, numpy y matplotlib. En caso de querer que en los gráficos aparezca la distania del algoritmo genético con respecto a la distancia exacta, se debe realizar con una cantidad de ciudades menor a 10, esta decisión se tomó ya que el algoritmo de fuerza bruta implementado se demora mucho para más ciudades.
Para correr los experimentos basta escribir en la terminal:
python AlgGen.py
Se ejecutará el algoritmo genético con los parámetros indicados al inicio del archivo. Si se desea cambiar los parámetros se debe realizar manualmente, en caso de querer realizar varios gráficos o cambiar los parámetros del gráfico actual, esto se puede hacer directamente al invocar las funciones dist_AlgGen() y plot_res().

3. Diseño experimental

El dataset utilizado fue creado artificialmente, enumerando las ciudades desde 0 a n y dándoles coordenadas en el plano x-y al azar.
Para calcular la solución exacta se calcularon todas las permutaciones posibles y se calculó la menor distancia entre ellas.
Para realizar el cross-over, se decidió analizar ambos caminos hamiltonianos al mismo tiempo y lanzar una moneda para cada posición, la moneda indicará cuál de los padres heredará esa ciudad en esa posición al hije.
En caso de que el hijo ya posea ambas ciudades que ofrecen los padres para dicha posición, entonces se escoge una ciudad al azar entre las que queda por elegir. Añadiendo así la posibilidad de mutaciones dentro del cross-over.
Sin embargo, no sólo se realiza mutación en el cross-over, sino que dado que se eligen 3 ganadores de cada generación y entre ellos 3 se generan 3 hijos. El resto de candidatos es creado mutando estos 6 candidatos utilizando la una función llamada swap_2values() que intercambia 2 ciudades al azar en un camino hamiltoniano.

La configuración de la máquina usada para los testings es:
Sistema Operativo: macOS Sierra.
Procesador: 1.6 GHz Intel Core i5.
Memoria: 4 GB 1600 MHz DDR3.

4. Resultados y Análisis

El primer gráfico nos muestra la curva de distancias para 9 ciudades y 9 candidatos considerados en cada generación
Con un total de 80000 generaciónes con el objetivo de comprobar convergencia. Sin embargo, no se puede apreciar una convergencia clara para una cantidad tan pequeña de ciudades.
El gráfico puede ser engañoso, ya que cada punto muestra el resultado cada 1000 generaciones, por lo que podría haber tendencias intermedias no perceptibles por el gráfico actual.
A pesar de que a veces el algoritmo empeore la distancia encontrada, podemos notar que en numerosas ocasiones se encuentra la respuesta correcta, sin embargo, no converge a ella.

El segundo gráfico nos muestra la curva de distancias para el algoritmo genético.
En esta instancia sí es posible observar una tendencia.
Se mantiene el algoritmo pon una cantidad menor de generaciones, 800 para ser exactos y graficando los datos cada 20 generaciones.
No es posible mostrar la solución exacta para este caso (90 ciudades y 18 candidatos) ya que la solución realizada por fuerza bruta se demoraría mucho.

El tercer gráfico nos muestra otra curva para los mismos parámetros que el gráfico 2. Es distinto al gráfico anterior ya que la elección de los candidatos y las mutaciones son aleatorias.
Se puede observar una clara convergencia.

Figura 1: Distancia versus cantidad de generaciones, cada 5 valores en el eje x equivale a 1000 generaciones

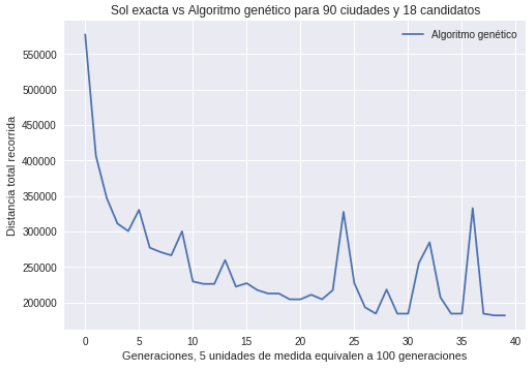

Figura 2: Distancia versus cantidad de generaciones, cada 5 valores en el eje x equivale a 100 generaciones

5. Conclusiones

Debido a que no se logró implementar la solución exacta o alguna heurística buena para distancia mínima dada una cantidad relativamente grande de ciudades, no se puede medir la eficiencia correctamente.
Para el caso de un número de ciudades menor a 10 sí es posible y notamos que el algoritmo es capaz de encontrar la solución exacta varias veces a lo largo de las generaciones. Sin embargo, por algún error en la implementación que no pudo ser descubierto, el algoritmo no siempre se quedaba con la menor distancia, lo que afectaba la convergencia e hiciera caótico la distancia mínima en función de las generaciones.
Para pocas ciudades es posible encontrar la solución exacta en menos de 1000 generaciones, pero no es posible determinar si ya se encontró la solución sin saber de antemano cuál es la solución exacta, debido a lo errático de los resultados.
Para una mayor cantidad de ciudades es posible apreciar cómo se va disminuyendo la distancia con el paso de las generaciones. Sin embargo, como ya se indicó anteriormente, al no tener la respuesta exacta no podemos saber qué tan eficiente es, aunque por la forma de la curva podemos suponer que nos encontramos ante una solución aproximada relativamente buena. Ya que se puede apreciar que la curva cada vez decrece con una mayor pendiente.
Se concluye que los algoritmos genéticos pueden ser muy útiles tanto para encontrar soluciones aproximadas como para obtener soluciones exactas.
La dificultad está en encontrar una función de fitness adecuada, idear una forma en que se puedan cruzar los padres de tal manera de que el hijo pueda obtener información útil de los padres.
Se observa que la rapidez del algoritmo implementado supera con creces al algoritmo de fuerza bruta. Pudiendo realizar 8000 generaciones para caminos de largo 90 y 18 candidatos en menos de 5 segundos.

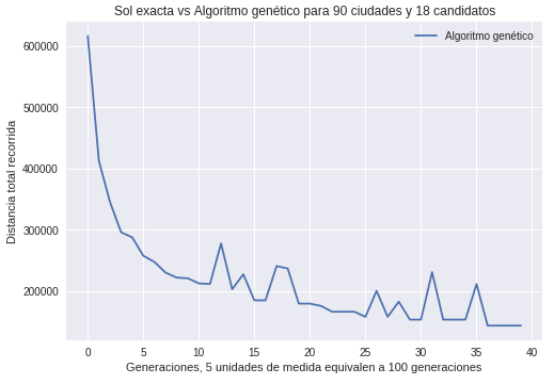

Figura 3: Distancia versus cantidad de generaciones, cada 5 valores en el eje x equivale a 100 generaciones