Trabajo final
Minería de Datos
Clasificar los datos Cleveland

Uxue Ayechu

Ruben Sesma

Clasificador KNN

Lo primero que hemos hecho ha sido aplicar los algoritmos vistos en clase al problema propuesto. En primer lugar el algoritmo de KNN para todas las posibles variaciones.

KNN

	Cleveland
K1	0.51666667
K1 KF1	0.51666667
K1 KF3	0.51666667
K1 KF5	0.51666667
K1 KF7	0.51666667
K3	0.53333333
K3 KF1	0.56666667
K3 KF3	0.58333333
K3 KF5	0.58333333
K3 KF7	0.56666667
K5	0.6
K5 KF1	0.55
K5 KF3	0.6
K5 KF5	0.58333333
K5 KF7	0.56666667
K7	0.6
K7 KF1	0.55
K7 KF3	0.56666667
K7 KF5	0.55
K7 KF7	0.55

Todos los resultados mostrados en la tabla son para el conjunto de test. Como podemos observar las mejores tasas de acierto las obtenemos para k=5, k=5 KF=3 y k=7, aunque solo alcanzamos un 60% de aciertos.

Reglas de Chi

A continuación hemos probado el clasificador de reglas de Chi. Hemos probado todas las combinaciones para agregación suma o máximo y para confianza penalizada y no penalizada por la suma del resto.

Chi

	Cleveland
confianza no penalizada Máximo	0.55
confianza no penalizada Suma	0.56666667
confianza penalizada Máximo	0.48333333
confianza penalizada Suma	0.48333333

Todos los resultados mostrados en la tabla son para el conjunto de test. Como podemos observar la mejor tasa de acierto es del 56% para el operador de agregación de la suma y la confianza para los pesos no penalizada por la suma del resto.

Híbrido KNN + Chi

Con lo visto en clase tal cual, la mayor tasa de aciertos es tan solo del 60% y la conseguimos para varias combinaciones de KNN, ya que los resultados de Chi son peores. A partir de estos resultados, hemos pensado mejorarlos haciendo un "híbrido" del KNN y el método de las reglas de Chi.

Lo primero que hace nuestro híbrido es aplicar el mejor KNN clásico, para k=5 y para k=7, ya que obtenemos mejores resultados que con Chi. A continuación sobre los ejemplos dudosos, aquellos cuyas distancias son muy similares para varias clases, aplicamos para la mitad el clasificador de Chi y para la otra mitad el clasificador obtenido con el KNN. ¿Por qué la mitad? Porque hemos descubierto observando las predicciones de Chi, las predicciones de 5NN y las cases reales del conjunto de entrenamiento, que en los ejemplos dudosos unas veces Chi los clasifica correctamente al contrario que KNN y otras veces sucede exactamente al revés. Sobre qué mitad empleamos uno u otro, lo dejamos en manos del azar.

Hemos creado otro híbrido similar para KNN difuso con k=5 y KF=3 para el cual también obteníamos un 60% de aciertos. El procedimiento es similar pero ahora los ejemplos dudosos son aquellos que tienen votos parecidos.

Además hemos de comentar que la tasa de aciertos varía mucho de unas iteraciones a otras como podemos observar en la tabla. Esto se debe a la aleatoriedad en la mezcla de los dos clasificadores. También hay que tener en cuenta el umbral que elijamos para determinar si un ejemplo es dudoso o no. (Para KNN clásico hemos elegido 5 y para NN Difuso hemos elegido el 1), probar a normalizar

KNN + Chi

	Cleveland
k = 5	0.6
k=5 + Chi	0.616666666666667
k=5 KF=3	0.6
k=5 KF=3 + Chi	0.616666667
k = 7	0.6
K=7 + Chi	0.583333333

Todos los resultados mostrados en la tabla son para el conjunto de test. Con el híbrido que hemos creado, el mejor resultado que hemos conseguido obtiene un 1% de mejoría. Hemos realizado numerosas pruebas y esta es la mejor, no siempre obtenemos 61% sino que a veces empeora, esto es debido a la aleatoriedad.

Genético KNN

Implementación de un algoritmo genético para la búsqueda de la mejor configuración de KNN

Hemos implementado un algoritmo genético el cual optimiza el número de aciertos en el conjunto entrenamiento sobre el dataset Cleveland, este valor es usado como función fitness del algoritmo genético. He utilizado para estas pruebas el KNN implementado en clase.

Representación de los cromosomas

Nuestro algoritmo, en cada población debe ejecutar el algoritmo KNN ó KNN difuso para obtener el fitness de cada cromosoma. Cada alelo de los cromosomas es un número.

Cada cromosoma se compone en total de 252 números, los dos primeros enteros y los restantes binarios.

Los dos primeros genes del cromosoma son números enteros y representan los parámetros de configuración de KNN. El primer gen corresponde al parámetro KF de KNN difuso usado para calcular las pertenencias de cada ejemplo a cada clase, el cual se moverá en el rango [0,10], si este valor es 0, corresponde a KNN clásico. El segundo gen corresponde al parámetro K, se moverá en el rango [1,10].

Si nos fiamos en el dataset, apreciamos que se compone de 13 variables y 237 ejemplos, por lo que los siguientes genes corresponden a la selección de variables y ejemplos.

Los 13 genes siguientes (posiciones 3 a 15 inclusive) son números binarios que indican la selección de variables.

El resto de genes (posición 16 hasta la 252 inclusive) corresponden a la selección de instancias.

Ejemplo de cromosoma:

Genes de configuración:

3	7

se ejecutaría 7NN difuso con KF=3

Genes de selección variables:

1	1	1	1	0	0	0	0	0	0	0	0	0

Indica que solo se usan las 4 primeras variables para aprender y clasificar

Genes de selección ejemplos: Similar a la selección de variables pero para instancias.

Método de selección de progenitores

Los progenitores se eligen por torneo de K participantes con K fijado a la mitad del tamaño de la población actual.

Métodos de cruce implementados

Cruce simple: Cada descendiente se obtiene de dos progenitores, padre y madre. Obtenemos 2 descendentes. Primer descendiente es idéntico al padre pero con la selección de variables de la madre. Para el otro descendiente se invierte el papel de los progenitores.

Cruce HUX modificado: Un descendiente hereda la parte de configuración de KNN del padre y el otro de la madre. Para el resto de la cadena binaria, buscamos las posiciones que son diferentes en el padre y la madre. Para cada posición diferente, la rellenamos con probabilidad 50% de la madre o del padre.

Mutaciones implementadas

Hemos implementado una mutación que lo que hace es: Para cada gen del cromosoma lo modifica aleatoriamente dentro de su rango. Debido a que los métodos de cruce no modifican los parámetros de configuración de KNN del cromosoma, hemos fijado la probabilidad de mutación a 0.15 para que se produzcan más mutaciones que en un algoritmo genético común.

Métodos de reemplazo de población implementados

Brecha generacional: se ordenan progenitores y descendentes por fitness y se mantiene el mejor 20% de los progenitores y el mejor 80% de los progenitores.

Generacional: Los descendientes reemplazan a los progenitores.

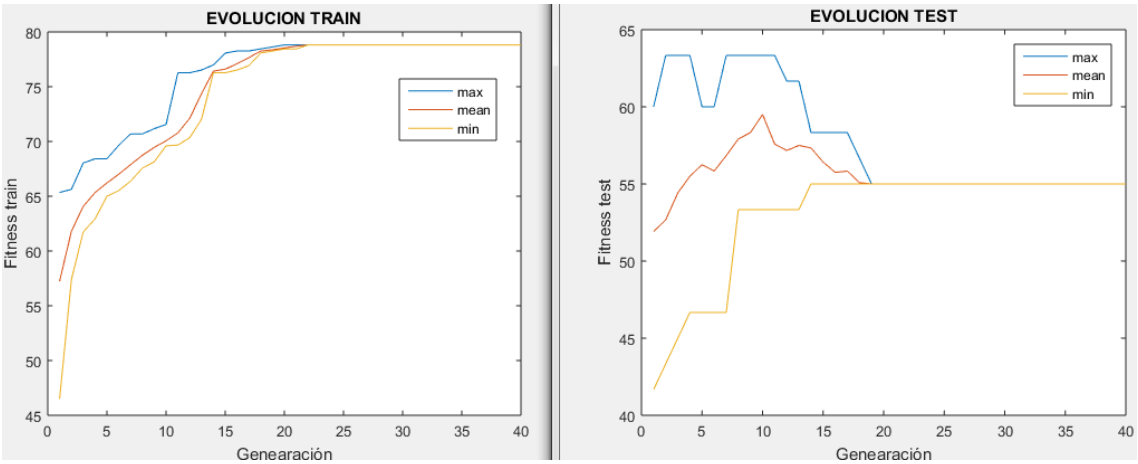
Elitista: Se ordenan progenitores y descendientes por fitness y se seleccionan los mejores para la siguiente generación.

Validación de la población: La selección de ejemplos puede dar la casualidad que sea menor que los parámetros de K o KF lo que provocaría un error de ejecución en el código, para resolver este problema me aseguro de introducir en el conjunto de entrenamiento al menos 10 ejemplos (rango máximo del valor de K y KF). Manualmente se incluyen 2 ejemplos de cada clase

Pruebas experimentales

Tamaño población: 20; Número de generaciones: 40; Cruce por binarios: Reemplazo elitista

Probabilidad de mutación 0.15. Obtenemos la siguiente evolución de porcentaje de aciertos:



Podemos observar claramente que al tratar de optimizar el porcentaje de aciertos en train, el algoritmo genético puede converger a soluciones que se han sobre ajustado a los datos de entrenamiento, por esto se reduce el porcentaje de aciertos en test. Por esta razón, para los siguientes experimentos para buscar una buena configuración para el KNN hemos decidido fijar el número de generaciones a 15 y con él, e tamaño de la población a 20, ya que alrededor de esta generación es cuando comienza a reducirse el porcentaje de aciertos en test.

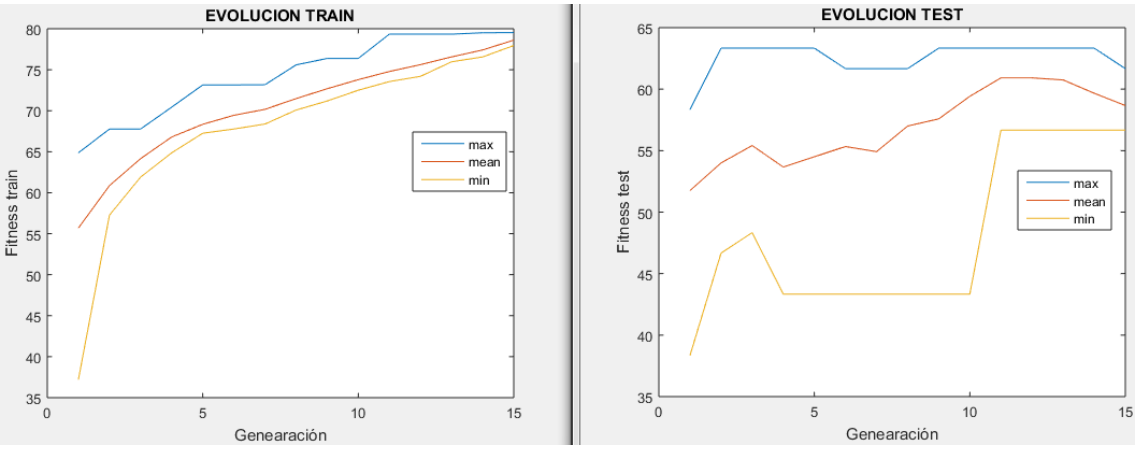
Pruebas fijado el número de generaciones

He realizado varias pruebas con los diferentes métodos de cruce, y reemplazo. Los métodos de cruce no afectan demasiado al comportamiento del algoritmo. Los métodos de reemplazo sí que afectan al funcionamiento del algoritmo, siendo el reemplazo elitista el que mejores resultados parece que ofrece. Por esto usaré en las siguientes pruebas el cruce por binarios y el reemplazo elitista.

Tamaño población: 20; Número de generaciones: 15; Cruce por binarios: Reemplazo elitista. Usaremos una configuración anterior para realizar las siguientes pruebas:

Prueba 1: Lanzamiento aleatorio

```
Creando poblacion y fitness inicial
Generacion 1: AccTrain:6.486486e+01 | AccTest:5.833333e+01
Generacion 2: AccTrain:6.776860e+01 | AccTest:6.333333e+01
Generacion 3: AccTrain:6.776860e+01 | AccTest:6.333333e+01
Generacion 4: AccTrain:7.043478e+01 | AccTest:6.333333e+01
Generacion 5: AccTrain:7.314815e+01 | AccTest:6.333333e+01
Generacion 6: AccTrain:7.314815e+01 | AccTest:6.166667e+01
Generacion 7: AccTrain:7.317073e+01 | AccTest:6.166667e+01
Generacion 8: AccTrain:7.559055e+01 | AccTest:6.166667e+01
Generacion 9: AccTrain:7.637795e+01 | AccTest:6.333333e+01
Generacion 10: AccTrain:7.637795e+01 | AccTest:6.333333e+01
Generacion 11: AccTrain:7.933884e+01 | AccTest:6.333333e+01
Generacion 12: AccTrain:7.933884e+01 | AccTest:6.333333e+01
Generacion 13: AccTrain:7.933884e+01 | AccTest:6.333333e+01
Generacion 14: AccTrain:7.950820e+01 | AccTest:6.333333e+01
Generacion 15: AccTrain:7.952756e+01 | AccTest:6.166667e+01
Elapsed time is 700.598200 seconds.
```



Se observa que hay mejore soluciones en la generación 14 que en la 15, esto es debido a que en la generación 14 tenemos muy parecido train a la generación 15, pero mejor resultados en test en la generación 14.

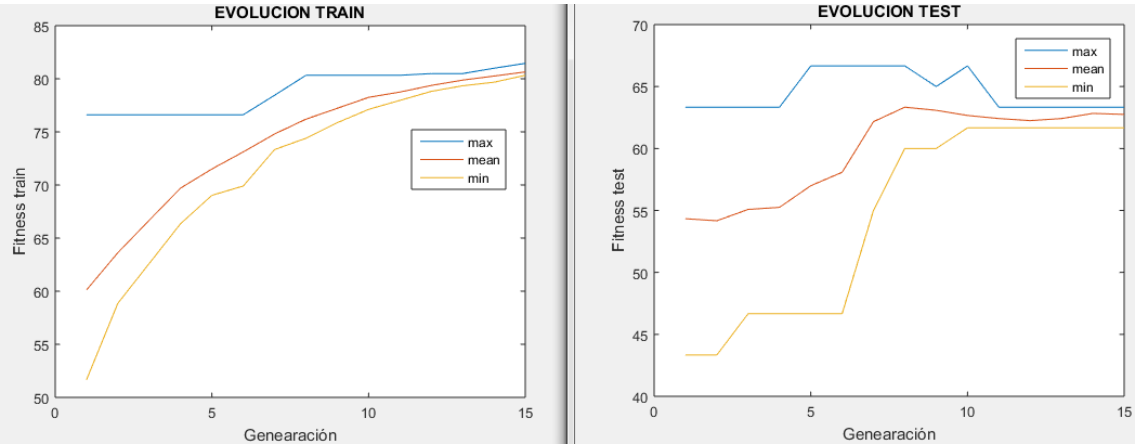
El mejor cromosoma en test de esta prueba lo adjunto en el fichero *sol63-3.mat*, el cual obtiene un 63.33% de acierto en test y 76.71% en train. Son porcentajes admisibles para este clasificador.

Prueba 2: Introducimos conocimiento en la población inicial

Consiste en lanzar el algoritmo introduciendo en la población inicial el cromosoma obtenido en la primera prueba. En esta prueba los fitness máximos de la primera generación, muy probablemente serán los mejores obtenidos en la primera prueba.

Tras esta segunda prueba obtenemos:

```
Creando poblacion y fitness inicial
Generacion 1: AccTrain:7.661290e+01 | AccTest:6.333333e+01
Generacion 2: AccTrain:7.661290e+01 | AccTest:6.333333e+01
Generacion 3: AccTrain:7.661290e+01 | AccTest:6.333333e+01
Generacion 4: AccTrain:7.661290e+01 | AccTest:6.333333e+01
Generacion 5: AccTrain:7.661290e+01 | AccTest:6.666667e+01
Generacion 6: AccTrain:7.661290e+01 | AccTest:6.666667e+01
Generacion 7: AccTrain:7.844828e+01 | AccTest:6.666667e+01
Generacion 8: AccTrain:8.032787e+01 | AccTest:6.666667e+01
Generacion 9: AccTrain:8.032787e+01 | AccTest:65
Generacion 10: AccTrain:8.032787e+01 | AccTest:6.666667e+01
Generacion 11: AccTrain:8.032787e+01 | AccTest:6.333333e+01
Generacion 12: AccTrain:8.048780e+01 | AccTest:6.333333e+01
Generacion 13: AccTrain:8.048780e+01 | AccTest:6.333333e+01
Generacion 14: AccTrain:8.099174e+01 | AccTest:6.333333e+01
Generacion 15: AccTrain:8.145161e+01 | AccTest:6.333333e+01
Elapsed time is 735.275551 seconds.
```



Observamos que sigue ocurriendo que las soluciones que ofrece el algoritmo en la última generación, si hacemos un balance entre los aciertos en entrenamiento y test las mejores soluciones estarían en las generaciones 8, 9 o 10.

Conclusiones de las primeras pruebas experimentales

Debido al problema del sobre aprendizaje comentado antes. A pesar de no ser una práctica habitual, al finalizar la ejecución, si encuentro algún cromosoma con resultados mejores de los obtenidos anteriormente, selecciono el cromosoma que mejor accuracy obtiene en test.

El mejor resultado obtenido después de muchas pruebas ha sido un cromosoma que obtiene 56.48% En train y 68.33% en test, el cual lo adjunto en el fichero *sol.mat* corresponde al 7NN difuso con KF=10 usando 6 variables y 108 ejemplos.

Pruebas KNN sin bucles

He implementado el mismo algoritmo usando como función fitness el entrenamiento que devuelve el knn_sin_bucles. Los mejores accuracy en test solo llega a 63.33% puede ser debido a que solo se prueba el KNN clásico y no el difuso, no tiene en cuenta el primer gen de los

cromosomas. En tiempo lo mejora muchísimo, pasa de tardar 500 segundos a 150 con la misma configuración de parámetros.

Híbrido genético KNN + Chi

La última prueba que hemos realizado se trata simplemente de unificar las pruebas anteriores. Con el mejor cromosoma obtenido para el genético KNN hemos aplicado el híbrido realizado anteriormente entre KNN y Chi. Con esta opción es como hemos conseguido obtener el 70% de aciertos. Hemos de mencionar que no siempre lo consigue ya que a la hora de mezclar los métodos KNN y Chi influye la aleatoriedad.

Mejor solución obtenida:

Partimos del mejor cromosoma obtenido en el apartado anterior con el cual obteníamos un 68'3% de aciertos. Dicho cromosoma especifica las variables y los ejemplos que hemos de emplear para crear el clasificador, además de los valores de K y KF para el clasificador k-means.

1º Creamos un clasificador con las variables, los ejemplos y los valores de K y KF que indica el cromosoma. En este caso K=7 y KF = 10.

2º También creamos el clasificador de Chi con los ejemplos y variables que indica el cromosoma anterior.

3º Creamos un vector binario con tantas posiciones como casos dudosos genera el KNN.

4º Clasificamos los ejemplos de entrenamiento con KNN y en los casos dudosos usamos según indique el vector binario Chi o KNN.

5º La prueba 4 la repetimos para todos los posibles Chi, los operadores de agregación suma y máximo y con confianzas penalizadas y no penalizadas. Obtenemos la siguiente tabla:

Entrenamiento			
	KNN	Chi	CHI+KNN
Confianza no penalizada, máximo	0,564814814814815	0,731481481481482	0,574074074074074
Confianza penalizada, máximo		0,648148148148148	0,564814814814815
Confianza no penalizada, suma		0,703703703703704	0,574074074074074
Confianza penalizada, suma		0,620370370370370	0,564814814814815

6º Escogemos el mejor clasificador para CHI+KNN y lo aplicamos sobre Test obteniendo los siguientes resultados:

Test	
CHI + KNN	
Confianza no penalizada, máximo	0.691246667