

UNIVERSIDAD DE CÓRDOBA

ESCUELA POLITÉCNICA SUPERIOR

INTRODUCCIÓN AL APRENDIZAJE AUTOMÁTICO

DEPARTAMENTO DE INFORMÁTICA Y ANÁLISIS NUMÉRICO



Documentación de Prácticas

Juan Jesús Carmona Tejero

Gregorio Corpas Prieto

Índice general

Índice de Figuras	VI
Índice de Tablas	VII
1 Introducción a WEKA	1
1.1 Filtros no supervisados	1
1.1.1 Normalize	1
1.1.2 ReplaceMissingValues	1
1.1.3 RemoveUseless	2
1.1.4 PrincipalComponents	2
1.1.5 RandomProjection	2
1.1.6 NominalToBinary	2
1.1.7 RemoveMissclassified	3
1.1.8 RemovePercentage	3
1.1.9 Resample	3
1.2 Filtros supervisados	4
1.2.1 AttributeSelection	4
1.2.2 Discretize	4
1.2.3 NominalToBinary	4
1.2.4 SpreadToBinary	4
1.2.5 ClassBalancer	4

1.2.6	Resampler	5
1.3	Base de datos Wine	6
1.3.1	Detalles de la base de datos	6
1.3.2	Modificación del fichero con nombres de atributo descriptivos	6
1.3.3	Descripción de atributos y clases	6
1.3.4	Tratamiento de elementos perdidos	6
1.3.5	Diferencia entre Distinct y Unique	6
1.3.6	Eliminar atributos identificadores	6
1.3.7	Relaciones visualmente significativas en entorno Visualice	6
1.4	Aplicación de 6 filtros a la base de datos	7
1.4.1	Filtro de selección de características	7
1.4.2	Filtro de selección de patrones	7
1.4.3	Filtro de filters/supervised/attribute/*	7
1.4.4	Filtro de filters/supervised/instance/*	7
1.4.5	Filtro de filters/unsupervised/attribute/*	7
1.4.6	Filtro de filters/unsupervised/instance/*	8
1.5	Conversión de todo atributo nominal a codificación binaria . . .	9
1.6	División de la base de datos en particiones	10
1.6.1	División en 10-Holdout 75-25	10
1.6.2	División en 10-Fold	10
2	Clasificación y Regresión en WEKA	11
2.1	Algoritmo IB1 con 10-fold crossvalidation	11
2.1.1	Visualización de la clasificación	11
2.1.2	Interpretación de los resultados	11
2.2	Algoritmo IBK ($k=1, k=3, k=5$) con 10-fold crossvalidation a 1	12
2.2.1	Cálculo de media y desviación típica de las medidas . .	12
2.2.2	Visualización de la clasificación	12
2.2.3	Interpretación de los resultados	12

2.3	Base de datos house.arff	13
2.3.1	Carga de la base de datos	13
2.3.2	Variable Granite	13
2.3.3	Variable binaria Bathroom	13
2.3.4	Variable Bedrooms	13
2.3.5	Variable HouseSize	13
2.4	Base de datos autoMpg.arff	14
2.4.1	Aplicación del algoritmo LinearRegression con un 80-20	14
2.4.2	Resultados obtenidos	14
2.4.3	Atributo que aporta más información de la variable dependiente	14
2.4.4	Atributo que no aporta información al modelo	14
2.4.5	Errores cometidos	14
2.4.6	Modificación del parametro attributteSelectionMethod .	14
2.5	Método SimpleLinearRegression en base de datos autoMpg.arff	15
2.6	Algoritmos Logistic y SimpleLogistic	16
2.6.1	Análisis	16
2.6.2	Visualización gráfica de errores cometidos	16
2.6.3	Análisis de parámetros	16
3	MOEA memético utilizando evolución diferencial	17
3.1	Introducción a la evolución diferencial	17
3.1.1	Variantes de la evolución diferencial	20
3.2	Evolución diferencial para optimización multi-objetivo utilizando el concepto de dominacia de Pareto	24
3.3	Evolución diferencial de Pareto con MOANNs	26
3.4	Caminos futuros en la evolución diferencial multi-objetivo . . .	30
3.5	El algoritmo MPANN	31
3.6	El algoritmo MPDE	36
3.6.1	Funciones objetivo	37

3.6.2	Operadores	37
3.6.3	Búsqueda local	37
3.6.4	Etapas y aspectos relevantes de MPDE	38
3.6.5	Diferencias con el algoritmo MPANN	41
3.6.6	Diseño experimental	42
3.6.7	Resultados	44

Índice de Figuras

3.1	Cruce discreto binomial (izquierda) y cruce discreto exponencial (derecha). Los individuos \mathbf{x} e \mathbf{y} dan lugar a otro individuo \mathbf{z}	22
3.2	Principales modelos de DE. p es el número de pares de vectores que conforman la diferencia, j_r es un valor aleatorio generado en el intervalo $[0, n]$, donde n es el número de variables del problema. x_{r3} es el parente principal y x_{r1} y x_{r2} los padres secundarios. F y K son valores de escala, u_i es el hijo creado y x_{best} significa que se ha seleccionado como parente principal el mejor individuo o solución de la población en una determinada generación, <i>bin</i> representa cruce binario y <i>exp</i> cruce exponencial, y <i>dir</i> indica que se incluye información de alguna función de aptitud al cruce y a la mutación.	24
3.3	Cálculo de la distancia <i>crowding</i> . Los puntos azules son soluciones de un mismo frente.	26
3.4	Pseudocódigo del algoritmo MPANN.	33
3.5	Pseudocódigo del algoritmo MPDE.	39
3.6	Frente de Pareto en entrenamiento (A_1, A_2), y valores asociados a (A_1, C) en generalización para el conjunto de datos Balance.	45

Índice de Tablas

3.1 Características de los conjuntos de datos de la UCI	42
3.2 Resultados estadísticos para MPDE, MPENSGAI y SVM en media y desviación típica sobre el conjunto de generalización para C y MS	44

Capítulo 1

Introducción a WEKA

1.1. Filtros no supervisados

Son filtros que no tienen en cuenta la variable marca de clase.

1.1.1. Normalize

Realiza una normalización de todos los valores numéricos en el conjunto de datos.

- **Uso**

Los valores son modificados al rango [0,1], tomando el valor 0 el dato más pequeño del conjunto y tomando el valor 1 el dato mayor del mismo, quedando el resto de valores en valores continuos dentro del rango.

- **Ejemplo**

1.1.2. ReplaceMissingValues

Reemplaza todo valor perdido, los cuales son representados con el signo '?', de los atributos nominales y numéricos.

1.1. Filtros no supervisados

- **Uso**

Busca aquellos valores perdidos y los reemplaza con los valores de las modas y medias para dicho atributo.

Para ilustrar el uso de este filtro se ha usado una pequeña base de datos que contiene notas de 3 asignaturas para varias instancias que son los alumnos, y se observa como los datos perdidos son reemplazados.

- **Ejemplo**

1.1.3. RemoveUseless

Elimina atributos nominales donde la varianza es muy grande o muy pequeña y que por tanto, no tienen utilidad.

- **Uso**

Parar ilustrar este filtro se ha usado una pequeña base de datos que contiene 1 atributo nominal que representa un color, y 2 atributos numéricos que representan otros datos asociados. Se puede observar como el atributo nominal tiene un valor distinto para cada instancia, y por tanto, la varianza es la máxima, así pues el filtro actúa descartando dicho atributo nominal.

- **Ejemplo**

Realiza el análisis y transformación de los componentes principales de los datos. Utilizar junto con una búsqueda de Ranker. La reducción de la dimensionalidad se logra eligiendo suficientes vectores propios para tener en cuenta algún porcentaje de la varianza en los datos originales — por defecto 0.95 (95

1.1.4. PrincipalComponents

Realiza el análisis y transformación de las componentes principales de los datos.

- **Uso**

La reducción de la dimensionalidad se logra eligiendo suficientes vectores propios para tener en cuenta algún porcentaje de la varianza en los datos originales (por defecto 0.95). El ruido de los atributos puede filtrarse transformándolo en el espacio de la Componente Principal, eliminando algunos de los vectores propios peores, y luego transformando de nuevo al espacio original.

- **Ejemplo**

1.1.5. RandomProjection

Reduce la dimensionalidad de los datos proyectándolos en un subespacio de menor dimensión usando una matriz aleatoria con columnas de longitud unitaria.

- **Uso** Primero aplica el filtro NominalToBinary para convertir todos los atributos a numérico antes de reducir la dimensión. Conserva el atributo de marca de clase.

- **Ejemplo**

1.1.6. NominalToBinary

Convierte todos los atributos nominales en atributos binarios numéricos.

- **Uso**

1.1. Filtros no supervisados

Un atributo con k posibles valores se transforma en k atributos binarios (0-1) si la clase es nominal. Los atributos binarios se dejan binarios. Si la clase es numérica, es posible que desee utilizar la versión supervisada de este filtro.

- **Ejemplo**

1.1.7. RemoveMissclassified

Elimina aquellas instancias que han sido incorrectamente clasificadas, de modo que no existan valores atípicos.

- **Uso**

Permite escoger la marca de clase en la que se basan las clasificaciones erróneas, el clasificador sobre el que se basarán las clasificaciones erróneas, si el resultado será descartado o aceptado, número de iteraciones, pliegues y umbral de error permisible.

- **Ejemplo**

1.1.8. RemovePercentage

Permite eliminar un porcentaje de la información de la base de datos.

- **Uso**

- **Ejemplo**

1.1.9. Resample

Produce una submuestra aleatoria de un conjunto de datos utilizando el muestreo con reemplazo o sin reemplazo.

- **Uso**

Se puede especificar el número de instancias en el conjunto de datos generado. Cuando se utilizan en modo por lotes, los lotes posteriores no son remuestreados.

- **Ejemplo**

1.2. Filtros supervisados

1.2. Filtros supervisados

Son filtros que tienen en cuenta la variable marca de clase.

1.2.1. AttributeSelection

- Uso

- Ejemplo

1.2.2. Discretize

- Uso

- Ejemplo

1.2.3. NominalToBinary

- Uso

- Ejemplo

1.2.4. SpreadToBinary

- Uso

- Ejemplo

1.2.5. ClassBalancer

- Uso

- Ejemplo

1.2.6. Resampler

- Uso

- Ejemplo

1.3. Base de datos Wine

- 1.3.1. Detalles de la base de datos
- 1.3.2. Modificación del fichero con nombres de atributo descriptivos
- 1.3.3. Descripción de atributos y clases
- 1.3.4. Tratamiento de elementos perdidos
- 1.3.5. Diferencia entre Distinct y Unique
- 1.3.6. Eliminar atributos identificadores
- 1.3.7. Relaciones visualmente significativas en entorno Visulaice

1.4. Aplicación de 6 filtros a la base de datos

1.4.1. Filtro de selección de características

- Uso
- Resultados:
- Ejemplo

1.4.2. Filtro de selección de patrones

- Uso
- Resultados:
- Ejemplo

1.4.3. Filtro de filters/supervised/attribute/*

- Uso
- Resultados:
- Ejemplo

1.4.4. Filtro de filters/supervised/instance/*

- Uso
- Resultados:
- Ejemplo

1.4.5. Filtro de filters/unsupervised/attribute/*

- Uso

1.4. Aplicación de 6 filtros a la base de datos

- Resultados:
- **Ejemplo**

1.4.6. Filtro de filters/unsupervised/instance/*

- **Uso**
- Resultados:
- **Ejemplo**

1.5. Conversión de todo atributo nominal a codificación binaria

1.6. División de la base de datos en particiones

1.6. División de la base de datos en particiones

1.6.1. División en 10-Holdout 75-25

- **Uso**
- Resultados:
- **Ejemplo**

1.6.2. División en 10-Fold

- **Uso**
- Resultados:
- **Ejemplo**

Capítulo 2

Clasificación y Regresión en WEKA

2.1. Algoritmo IB1 con 10-fold crossvalidation

2.1.1. Visualización de la clasificación

2.1.2. Interpretación de los resultados

2.2. Algoritmo IBK (k=1, k=3, k=5) con 10-fold crossvalidation a 1

2.2. Algoritmo IBK (k=1, k=3, k=5) con 10-fold cross-validation a 1

2.2.1. Cálculo de media y desviación típica de las medidas

- Accuracy:
- Kappa:
- RMSE:
- F-Measure:
- Media ponderada AUC:

2.2.2. Visualización de la clasificación

2.2.3. Interpretación de los resultados

2.3. Base de datos house.arff

2.3.1. Carga de la base de datos

2.3.2. Variable Granite

- Influencia en el modelo
- Conclusiones

2.3.3. Variable binaria Bathroom

- Influencia en el modelo
- Aplicación de algoritmo de regresión lineal
- Conclusiones

2.3.4. Variable Bedrooms

- Influencia en el modelo
- Conclusiones

2.3.5. Variable HouseSize

- Influencia en el modelo
- Aportación al modelo de regresión lineal
- Conclusiones

2.4. Base de datos autoMpg.arff

2.4. Base de datos autoMpg.arff

2.4.1. Aplicación del algoritmo LinearRegression con un 80-20

2.4.2. Resultados obtenidos

- Conclusiones
- Tablas comparativas

2.4.3. Atributo que aporta más información de la variable dependiente

2.4.4. Atributo que no aporta información al modelo

2.4.5. Errores cometidos

- Visualización
- Representación de las diferentes cruces

2.4.6. Modificación del parametro attributeSelectionMethod

- Conclusión en el nuevo modelo
- Análisis de menor peso de "weight" frente "acceleration"

2.5. Método SimpleLinearRegression en base de datos autoMpg.arff

- Respuesta
- Solución
- Visualización
- Comentario de resultados

2.6. Algoritmos Logistic y SimpleLogistic

2.6.1. Análisis

- Conclusión
- Métricas
- Variables más influyentes (beta)
- Variables no usadas
- Visualización
- Asociación de fórmulas con modelos obtenidos según el algoritmo

2.6.2. Visualización gráfica de errores cometidos

2.6.3. Análisis de parámetros

- maxBoostingIterations
 - Análisis
 - Modificación
- heuristicStop
 - Análisis
 - Modificación

Capítulo 3

MOEA memético utilizando evolución diferencial

3.1. Introducción a la evolución diferencial

A continuación se dará una breve introducción de la evolución diferencial (*Differential Evolution*, DE) ? para entender el algoritmo MPANN (*Memetic Pareto Artificial Neural Networks*, MPANN) de Abbass, el cual describiremos en las siguientes secciones, y a partir del cual hemos desarrollado un algoritmo propio basado en DE llamado MPDE (*Memetic Pareto Differential Evolution*), utilizando las medidas (MS, C) descritas en el capítulo ??.

La DE fue propuesta por Storn y Price ? como nueva heurística para la minimización de funciones no lineales y no diferenciables en espacios totalmente ordenados. La DE es un tipo de técnica de optimización global que usa selección y cruce como sus operadores primarios para la optimización de problemas sobre dominios continuos, e incluso mutación, aunque este último operador fue introducido posteriormente al trabajo inicial de Storn y Price, por ejemplo en ?.

Los algoritmos tradicionales deterministas son insuficientes cuando se abordan problemas de optimización que presentan características como: no linealidad, alta dimensionalidad, existencia de múltiples óptimos locales (multimodal), no diferenciabilidad o ruido. La DE es un método alternativo en la solución a problemas con estas características.

3.1. Introducción a la evolución diferencial

Dentro de las características fundamentales de la DE se encuentran ?:

- La capacidad de manejar funciones objetivo no lineales, no diferenciales y multimodales.
- La paralelización, puesto que el algoritmo es fácilmente paralelizable y resulta útil cuando la evaluación de la función objetivo es computacionalmente costosa.
- La falta de predefinición de las distribuciones de probabilidad, como en el caso de las estrategias evolutivas.
- El uso de una codificación real y de una precisión determinada por el formato de punto flotante empleado.
- La convergencia a un valor óptimo (posiblemente local) de manera consistente a lo largo de una secuencia de ejecuciones independientes.
- La escasa utilización de parámetros de control, puesto que el algoritmo básico ? emplea únicamente tres parámetros de control, además de un criterio de terminación.
- Así mismo, posee una amplia gama de aplicaciones, de entre las que se encuentran el entrenamiento de ANNs, el diseño de filtros digitales, la optimización de procesos químicos no lineales, el diseño de redes de transmisión de aguas, etc ?.

La DE, como heurística evolutiva, tiene algunas características básicas que comparte con los EAs ?:

- Es una aproximación basada en poblaciones.
- El cruce y en ocasiones la mutación, se utilizan como operadores para generar nuevas soluciones.
- Hay un mecanismo de reemplazo para mantener un tamaño fijo en la población de individuos.

La DE difiere ? trabaja creando una población inicial aleatoria de soluciones, donde está garantizado, mediante reglas de reparación, que el valor de cada variable esta dentro de unos límites. Entonces un individuo es seleccionado aleatoriamente para ser reemplazado y se seleccionan tres padres diferentes para formar un nuevo hijo. Uno de estos padres se elige como parente principal y cada alelo en el parente principal se cambia al azar, donde al menos una variable debe ser cambiada. Esto se lleva a cabo añadiendo al valor de las variables un valor ponderado de la diferencia entre los dos valores de esta variable en los otros dos padres. En esencia, el vector asociado al parente principal es perturbado con el vector de los otros dos padres. Esto representa el operador de cruce en la DE. Si el vector resultante es mejor que el escogido para reemplazar, se reemplaza, en caso contrario el vector escogido permanece en la población.

Detallado de manera más formal, y en la misma notación que propuso Storn y Price ?, una solución l , en la generación i , es un vector multidimensional $\mathbf{x}_{G=i}^l = (x_1^l, \dots, x_N^l)^T$. Una población, $P_{G=k}$, en la generación $G = k$ es un vector de M soluciones ($M > 4$). La población inicial, $P_{G=0} = \{\mathbf{x}_{G=0}^1, \dots, \mathbf{x}_{G=0}^M\}$ se inicializa como:

$$x_{i,G=0}^l = inferior(x_i) + aleatorio[0, 1] \cdot (superior(x_i) - inferior(x_i)),$$

$$l = 1, \dots, M, \quad i = 1, 2, \dots, N,$$

donde M es el tamaño de la población, N es la dimensión de la solución, y cada variable i en un vector de soluciones l en la generación inicial $G = 0$, $x_{i,G=0}^l$, se inicializa dentro de los límites ($inferior(x_i)$, $superior(x_i)$). La selección se lleva a cabo seleccionando 4 soluciones con índices diferentes (3 padres más la solución a reemplazar), r^1, r^2, r^3 y $j \in [1, M]$. Los valores de cada variable en el hijo se cambian mediante un operador de cruce con una probabilidad CR , de la siguiente manera:

$$\forall i \leq N, x'_{i,G=k} = \begin{cases} x_{i,G=k-1}^{r^3} + F \cdot (x_{i,G=k-1}^{r^1} - x_{i,G=k-1}^{r^2}) \\ \text{si } (aleatorio[0, 1] < CR \wedge i = i_{aleatorio}) \\ x_{i,G=k-1}^j \quad \text{en otro caso} \end{cases}$$

3.1. Introducción a la evolución diferencial

donde $F \in (0, 1)$ es un parámetro que representa la cantidad de perturbación añadida al padre principal (r^3 en este caso). La nueva solución reemplaza a la antigua seleccionada si es mejor que ella, y al menos una de las variables se debe cambiar. Esta última está representada en el algoritmo de forma aleatoria, seleccionando una variable, $i_{aleatorio} \in (1, N)$. Después del cruce, si una o más de las variables en la nueva solución se encuentran fuera de su límite inferior o superior, se aplica la siguiente regla de reparación:

$$x_{i,G=k} = \begin{cases} \frac{x_{i,G} + inferior(x_i)}{2} & \text{si } x_{i,G+1}^j < inferior(x_i) \\ inferior(x_i) + \frac{x_{i,G} + superior(x_i)}{2} & \text{si } x_{i,G+1}^j > superior(x_i) \\ x_{i,G+1}^j & \text{en otro caso} \end{cases}$$

Una vez comentados los detalles esenciales de la DE, decir que la DE está restringida a dominios donde el espacio de búsqueda está completamente ordenado y en especial subespacios de \Re^n . Un individuo se representa como una n -tuple, llamada vector objetivo $\mathbf{x} = (x_1, \dots, x_n)$, donde $x_i \in \Re$ ($i = 1, \dots, n$) son los valores escalares que representan las variables de diseño del problema. Emplear una codificación real permite generar perturbaciones acotadas en las variables de diseño, a consecuencia del orden determinado por \Re^n . La codificación real de la DE, a diferencia las estrategias de evolución (*Evolution Strategies*, ES) no utiliza una distribución fijada (como la distribución Gaussiana fijada en ES) para controlar el comportamiento del operador de mutación, en lugar de ello, la distribución de las soluciones en el espacio de búsqueda determina el tamaño de paso y la dirección de búsqueda para cada individuo.

El lector puede consultar una revisión completa y detallada sobre DE y sobre aplicaciones en [????](#) y en un reciente trabajo de Ferrante Neri y Ville Tirronen en [?](#).

3.1.1. Variantes de la evolución diferencial

Hay algunas variantes [?](#) del algoritmo básico de DE, que se diferencian en:

- El tipo de criterio para seleccionar uno de los individuos a usar en el operador de cruce como padre principal.
- El número de diferencias computadas en la operación de cruce.
- El operador de cruce escogido.

La variante más popular se llama “*DE/rand/1/bin*”, donde DE se refiere al nombre del algoritmo, *rand* indica una elección aleatoria de los vectores que conforman el vector de mutación, la cifra *1* señala el número de pares de vectores que conforman la diferencia y *bin* establece un proceso de cruce binomial. Así, por ejemplo, un algoritmo de DE que selecciona aleatoriamente a cuatro vectores (2 pares) que componen al vector de mutación, y que son recombinados con un proceso exponencial, se representa como ”*DE/rand/2/exp*”. La figura 3.1 muestra gráficamente el cruce binomial y exponencial.

Además de los parámetros típicos de los EAs, la DE adopta dos nuevos parámetros: *CR*, que representa una probabilidad de cruce, normalmente entre $[0, 1]$ y controla la influencia de los padres en la generación de los hijos, y *F*, un parámetro que regula las magnitudes relativas de las diferencias del vector mutación, y que suele ser un valor aleatorio obtenido de una distribución normal $N(0, 1)$ o uniforme $U(0, 1)$. Estos parámetros dependen en cierta medida de las características de las funciones objetivo y del tamaño de la población.

3.1. Introducción a la evolución diferencial

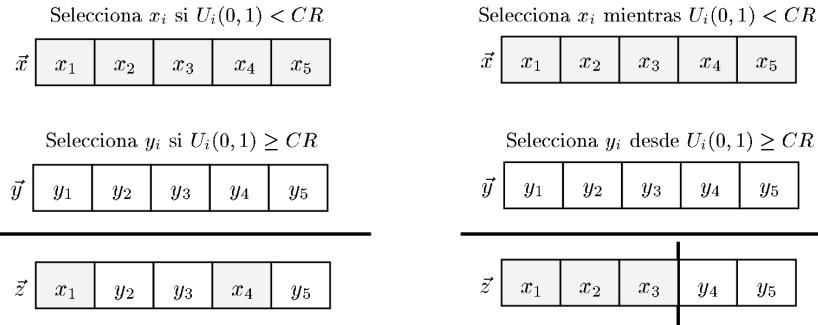


Figura 3.1: Cruce discreto binomial (izquierda) y cruce discreto exponencial (derecha). Los individuos **x** e **y** dan lugar a otro individuo **z**.

Definimos a continuación las variantes de la DE (la figura 3.2 muestra un resumen de las principales variantes):

- Variantes con operador de cruce discreto (binomial o exponencial):
 - *DE/rand/1/bin*
 - *DE/rand/1/exp*
 - *DE/best/1/bin*
 - *DE/best/1/exp*

Las variantes con *rand* seleccionan el parente principal y un par de padres secundarios para calcular la mutación diferencial aleatoriamente. En contraste, las variantes con *best* utilizan la mejor solución de la población como parente principal, y un par de individuos seleccionados aleatoriamente como padres secundarios.

- Variantes con cruce aritmético:
 - *DE/current-to-rand/1*
 - *DE/current-to-best/1*

La diferencia entre ellos es que el primero selecciona el parente principal y los padres secundarios de manera aleatoria en la población actual,

mientras que el segundo utiliza la mejor solución de la población actual como padre principal, y los padres secundarios se eligen aleatoriamente.

- Variantes con cruce combinado aritmético discreto (similar a las anteriores pero utiliza cruce binomial):

- $DE/current-to-rand/1/bin$

3.2. Evolución diferencial para optimización multi-objetivo utilizando el concepto de dominacia de Pareto

Nomenclatura	Modelo
rand/p/bin	$u_{i,j} = \begin{cases} x_{r_3,j} + F \cdot \sum_{k=1}^p (x_{r_1^p,j} - x_{r_2^p,j}) & \text{si } U_j(0,1) < CR \text{ o } j = j_r \\ x_{i,j} & \text{en caso contrario} \end{cases}$
rand/p/exp	$u_{i,j} = \begin{cases} x_{r_3,j} + F \cdot \sum_{k=1}^p (x_{r_1^p,j} - x_{r_2^p,j}) & \text{desde } U_j(0,1) < CR \text{ o } j = j_r \\ x_{i,j} & \text{en caso contrario} \end{cases}$
best/p/bin	$u_{i,j} = \begin{cases} x_{best,j} + F \cdot \sum_{k=1}^p (x_{r_1^p,j} - x_{r_2^p,j}) & \text{si } U_j(0,1) < CR \text{ o } j = j_r \\ x_{i,j} & \text{en caso contrario} \end{cases}$
best/p/exp	$u_{i,j} = \begin{cases} x_{best,j} + F \cdot \sum_{k=1}^p (x_{r_1^p,j} - x_{r_2^p,j}) & \text{desde } U_j(0,1) < CR \text{ o } j = j_r \\ x_{i,j} & \text{en caso contrario} \end{cases}$
current-to-rand/p	$\vec{u}_i = \vec{x}_i + K \cdot (\vec{x}_{r_3} - \vec{x}_i) + F \cdot \sum_{k=1}^p (\vec{x}_{r_1^p} - \vec{x}_{r_2^p})$
current-to-best/p	$\vec{u}_i = \vec{x}_i + K \cdot (\vec{x}_{best} - \vec{x}_i) + F \cdot \sum_{k=1}^p (\vec{x}_{r_1^p} - \vec{x}_{r_2^p})$
current-to-rand/p/bin	$u_{i,j} = \begin{cases} \vec{x}_{i,j} + K \cdot (\vec{x}_{r_3,j} - \vec{x}_{i,j}) + F \cdot \sum_{k=1}^p (\vec{x}_{r_1^p,j} - \vec{x}_{r_2^p,j}) & \text{si } U_j(0,1) < CR \text{ o } j = j_r \\ x_{i,j} & \text{en caso contrario} \end{cases}$
rand/2/dir	$\vec{v}_i = \vec{v}_1 + \frac{F}{2}(\vec{v}_1 - \vec{v}_2 + \vec{v}_3 - \vec{v}_4) \quad \text{donde } f(\vec{v}_1) < f(\vec{v}_2) \text{ y } f(\vec{v}_3) < f(\vec{v}_4)$

Figura 3.2: Principales modelos de DE. p es el número de pares de vectores que conforman la diferencia, j_r es un valor aleatorio generado en el intervalo $[0, n]$, donde n es el número de variables del problema. x_{r_3} es el parente principal y x_{r_1} y x_{r_2} los padres secundarios. F y K son valores de escala, u_i es el hijo creado y x_{best} significa que se ha seleccionado como parente principal el mejor individuo o solución de la población en una determinada generación, bin representa cruce binario y exp cruce exponencial, y dir indica que se incluye información de alguna función de aptitud al cruce y a la mutación.

3.2. Evolución diferencial para optimización multi-objetivo utilizando el concepto de dominacia de Pareto

Para aplicar la estrategia de la DE a problemas multi-objetivo, hay que modificar el esquema original ?, ya que el conjunto de soluciones de un problema con múltiples objetivos no consiste en una sola solución (ver sección ?? del capítulo ??).

Hay varios aspectos que se deben considerar para extender la DE a un problema de optimización multi-objetivo:

- ¿Cómo promover la diversidad de la población?
- ¿Cómo seleccionar o retener los mejores individuos, es decir, cómo realizar elitismo?

Para promover la diversidad hay que tener en cuenta el proceso de selección por medio de mecanismos basados en alguna medida de calidad, la cual indique la cercanía entre los individuos que forman la población. Las dos medidas de diversidad más usadas en optimización multi-objetivo son:

Distancia *crowding* ?: Esta medida da una idea de cómo de agrupados están los vecinos de un determinado individuo en el espacio de la función objetivo y hace que los frentes sean lo más uniformes posible, sin agrupar muchos individuos en una zona y dejar ninguno o muy pocos en otras. La distancia *crowding* se estima en función de la media de las caras de un cubo formado al tomar como vértices los vecinos más cercanos a un individuo i (ver figura 3.3 y el trabajo de K. Deb ? para más información).

Compartición de aptitud o *fitness sharing*: Cuando un individuo comparte valores de su función de aptitud con otros, ésta se degrada en proporción al número y a la proximidad de los individuos que lo rodean dentro de un determinado perímetro. La vecindad de un individuo se define en términos de un parámetro llamado σ_{share} , que indica el radio de la vecindad, a la cual se le llama nicho. ??.

El lector puede obtener en ? un estado del arte actualizado en métodos de agrupamiento para obtener diversidad en ANNs.

Para promover el elitismo en optimización multi-objetivo se suele utilizar un archivo externo, llamado población secundaria, que almacena los individuos no dominados encontrados a lo largo de la búsqueda. Uno de los métodos

3.3. Evolución diferencial de Pareto con MOANNs

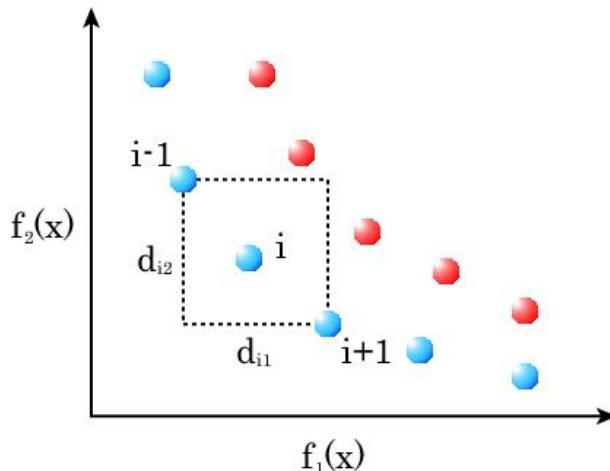


Figura 3.3: Cálculo de la distancia *crowding*. Los puntos azules son soluciones de un mismo frente.

más populares para seleccionar los mejores individuos de una población formada por padres e hijos es la ordenación de no dominados. Esta técnica se basa en el mecanismo de orden que se le da a los diferentes individuos de una población en forma de niveles. Por ejemplo, en el nivel 1 estarán los individuos no dominados. En el segundo nivel, estarán los individuos no dominados si no se tienen en cuenta los del primer nivel, y así sucesivamente. Según Goldberg ?, para mantener una diversidad apropiada, la metodología de ordenación de no dominados se debería usar en conjunción con alguna técnica de nichos como las mencionadas anteriormente. El algoritmo NSGAII ? es un claro ejemplo de esto.

Existen varios trabajos interesantes que utilizan DE con MOEAs y aplicaciones en ???.

3.3. Evolución diferencial de Pareto con MOANNs

En primer lugar vamos a considerar brevemente los artículos más interesantes sobre el uso de DE para el diseño de ANNs mediante técnicas multiobjetivo basadas en el concepto de dominancia y óptimo de Pareto, y en

los cuales nos hemos basado para la construcción de un nuevo MOEA basado en DE para el diseño de ANNs con unidades de base sigmoide.

El referente principal de este tipo de aplicaciones, usando una variación del algoritmo de DE original ?, es H. Abbass, creador del algoritmo multi-objetivo PDE (*Pareto Differential Evolution*) ???. En el algoritmo PDE se utiliza un caso especial de la variante *DE/current-to-rand/1/bin* con $K = 0$ (ver penúltima fila de la figura 3.2), ya que el padre principal se utiliza para la creación de un nuevo hijo y también para un tipo de cruce discreto. Los objetivos a minimizar son el *MSE* y la complejidad de la red.

El algoritmo trabaja como sigue: La población inicial se inicializa utilizando una distribución Gaussiana de media 0,5 y de desviación típica 0,15. Solamente las soluciones no dominadas se retienen en la población para el cruce y las dominadas se eliminan. Se seleccionan tres padres de manera aleatoria (uno de ellos como parente principal y también como solución hija) para generar un nuevo hijo. Los hijos se incluyen en la población solo si dominan al parente principal, en caso contrario, se hace un nuevo proceso de selección. Si el número de soluciones no dominadas excede un umbral, se adopta una métrica de distancia para eliminar padres que están muy cercanos unos de otros (esto se puede ver como un procedimiento de nichos en el cual la métrica de distancia es el radio del nicho). En esta aproximación, el tamaño de paso F se genera a partir de una distribución Gaussiana $N(0,1)$, y las restricciones de frontera se preservan, ya sea mediante un cambio de signo si la variable es ≤ 0 o mediante restas repetitivas, restando 1 si es ≥ 0 , hasta que la variable esté dentro de los límites permitidos. El algoritmo PDE también incorpora un operador de mutación que se aplica con una determinada probabilidad, después del operador de cruce, mediante la suma a cada variable de una pequeña perturbación aleatoria.

A partir de la aparición del algoritmo PDE, y casi en paralelo, Abbass desarrolla el algoritmo MPANN (*Memetic Pareto Artificial Neural Networks*) ?, que es una versión del PDE añadiéndole un algoritmo de LS basado en gradiente como es BP, con algunas mejoras, para así aumentar la velocidad de convergencia. MPANN trata de obtener modelos de ANNs que tengan buena

3.3. Evolución diferencial de Pareto con MOANNs

capacidad de generalización sin aumentar demasiado el tamaño de su arquitectura. Concretamente trata de minimizar el *MSE* y el número de neuronas en capa oculta. MPANN evoluciona conjuntamente la arquitectura y los pesos de la red y utiliza operadores cruce y mutación para la obtención de los hijos, codificando cada ANN en un cromosoma que representa la estructura y el valor de los pesos. Otras versiones de MPANN se utilizan para problemas reales como la diagnosis del cáncer ?, diferenciándose del MPANN original en la incorporación de un operador de mutación, ya que los algoritmos PDE y MPANN carecían de ello.

Otra variación del PDE es el algoritmo SPDE (*Self-adaptive Pareto Differential Evolution*) ?, la cual adapta de manera automática las probabilidades de cruce y mutación. Ambas probabilidades se heredan de los padres de la misma forma que se realiza el cruce para las variables de decisión. Si las probabilidades de cruce y de mutación no están entre (0, 1), se modifican automáticamente de acuerdo a unas reglas de reparación. Al igual que se realizó una versión auto-adaptativa del PDE con el algoritmo SPDE, Abbass propuso una versión auto-adaptativa del algoritmo MPANN, llamada SPANN (*Self-adaptive Pareto Artificial Neural Networks*) ?.

Un algoritmo que se debe mencionar a pesar de que sea un algoritmo evolutivo mono-objetivo con término de regularización es el de J. Illonen ?, donde se propone un estudio de la DE en el diseño de ANNs para encontrar el óptimo global de un problema. El algoritmo utiliza el *MSE* medio regularizado mediante la media de los pesos y los sesgos para entrenar a las ANNs. Illonen compara su metodología con métodos basados en gradiente y concluye que la DE puede ser más útil en el caso especial de algunas superficies de error, pero que la inclusión de alguna metodología híbrida que utilice conjuntamente optimización evolutiva e información basada en gradiente podría ser más beneficiosa.

En ?, se proponen una serie de experimentos utilizando el algoritmo multi-objetivo PDE de Abbass para evolucionar ANNs aplicadas a juegos de inteligencia artificial. La metodología propuesta contiene tres subsistemas: Un subsistema PDE canónico, un subsistema que introduce coevolución en

PDE con tres configuraciones posibles (PCDE), y un subsistema también de coevolución con PDE usando un archivo con tres configuraciones diferentes (PCDE-A). El primer subsistema se trata del algoritmo PDE con operadores de cruce y mutación y sin LS. El segundo sistema introduce coevolución, siendo la evaluación de cada individuo la principal diferencia con el algoritmo PDE. Para la ejecución del PCDE, cada ANN se compara con un número constante de ANNs elegidas al azar de la población de la generación actual. Si la puntuación de la ANN es mayor o igual a la de su oponente (elegidas al azar), recibirá una victoria. Por otra parte, la clasificación del primer frente de Pareto (mediante el etiquetado de soluciones no dominadas) se basa en el número de victorias como criterio de evaluación principal. En el algoritmo PDCE-A, al igual que en PCDE, después de la puntuación de cada ANN se lleva a cabo un segundo computo. Sin embargo, PCDE-A tiene un archivo extra que se utiliza para almacenar las soluciones de Pareto cada 50 generaciones. En consecuencia, cada ANN se compara con un número mínimo de ANNs elegidas al azar (sin repeticiones), a partir del archivo extra. Sólo si el número de ANNs en el archivo es menor que el número mínimo exigido de oponentes elegidos al azar, entonces la lista de oponentes se completa de ANNs elegidas al azar de la población. Del mismo modo, una ANN recibirá una victoria si su puntación es mayor o igual que la de su competidora. El número de victorias se utilizará como criterio de evaluación principal para etiquetar las soluciones no dominadas para el primer frente de Pareto. Este valor de evaluación será menor al depender del etiquetado de las soluciones dominadas, a causa del conjunto acotado de evaluadores. De la experimentación realizada con una serie de juegos de inteligencia artificial, se concluye que los mejores resultados los obtiene el algoritmo PDE. El pobre desempeño de los sistemas PCDE, incluso la versión que utilizan un archivo extra, en la producción de una buena distribución de soluciones a lo largo del frente de Pareto, es una prueba más de que los métodos co-evolutivos no son especialmente beneficiosos para la síntesis de agentes inteligentes para juegos en la evolución de Pareto.

3.4. Caminos futuros en la evolución diferencial multi-objetivo

En ? se propone un algoritmo llamado NSDE (*Non-dominated Sorting Differential Evolution*), que consiste en una modificación del algoritmo NSGAII ?, al que se introduce DE con la variante *DE/current-to-rand/1* (ver sección 3.1.1) en el cruce y la mutación. NSDE se utiliza para resolver problemas de rotación de funciones en el plano, atendiendo a dos funciones objetivo, una para cada eje de coordenadas del plano basándose en los grados de rotación. El algoritmo NSDE se compara con NSGAII en una serie de problemas de rotación, mostrando mejores soluciones por el proceso de cruce y mutación realizado en este tipo de problemas.

A continuación exponemos algunos de los caminos futuros de la DE usando MOEAS.

3.4. Caminos futuros en la evolución diferencial multi-objetivo

Según se puede ver en ?, la DE debe tener en cuenta estas futuras mejoras:

Diversidad: A pesar de que la DE tiene una alta convergencia, no posee suficiente robustez y tiene problemas para alcanzar el verdadero frente de Pareto, pudiendo quedar atrapada en óptimos locales. Además parece que la DE tiene problemas para crear un frente de Pareto homogéneo, con lo que se deberían aplicar alternativas de diversidad cuando se utilice con problemas multi-objetivo.

Variantes: A día de hoy no se sabe que variante de DE es mejor para problemas multi-objetivo para alcanzar el verdadero frente de Pareto de manera más efectiva.

Operador de mutación: Se deben tomar algunos nuevos criterios a la hora de seleccionar pares de soluciones en el proceso de mutación que sean más efectivos ?. En este momento estamos estudiando la selección de soluciones utilizando intervalos de confianza asociados a las distribuciones de los mejores individuos de la población ?.

Adaptación de los parámetros: Debe haber nuevas propuestas que no sean solamente las autoadaptativas ?? para optimizar los parámetros CR y F.

Alternativas en la codificación: DE se propuso para espacios se búsquedas continuos, por lo que se debería buscar una alternativa de codificación que permita el uso de la DE en problemas de optimización combinatoria.

Teoría: Los estudios sobre la convergencia de las variantes de DE y análisis en tiempo de ejecución, mejorarían la teoría actual.

3.5. El algoritmo MPANN

A continuación exponemos el algoritmo MPANN ? en su versión adaptada al reconocimiento de diagnosis del cáncer ?. Ésta versión se diferencia del algoritmo MPANN original en que incorpora un operador de mutación, del cual carecían los algoritmos PDE y MPANN originales. Además, tendremos en cuenta el algoritmo NSGAII ?, que nos servirá de base para la creación de nuestro algoritmo MPDE para el diseño de ANNs en multi-clasificación de patrones.

En la figura 3.4, se presenta el pseudocódigo del algoritmo MPANN que comentamos a continuación:

El primer paso de MPANN es generar una población inicial al azar siguiendo una distribución Gausiana (0, 1) (Paso 1).

Acto seguido, el algoritmo comienza su proceso evolutivo:

La primera acción (Paso 4) que tiene lugar al comienzo de una generación es la evaluación de los individuos, en el caso de MPANN utilizando la minimización del *MSE* y la complejidad de la red (número de neuronas en capa oculta). Una vez evaluados, se etiquetan los individuos no dominados.

Si el número de individuos no dominados es menor que 3, se busca un individuo no dominado de entre las soluciones que no están etiquetadas y se

3.5. El algoritmo MPANN

etiqueta como no dominado. Esto se repite hasta que el número de no dominados sea igual a 3 (Paso 5).

- 1) Generar una población P_0 , de manera que los pesos asignen aleatoriamente de acuerdo a una distribución Gaussiana $N(0,1)$ y la existencia de neurona en capa oculta dependa de una probabilidad de 0.5.
- 2) $k = 1$
- 3) Repetir
- 4) Evaluar los individuos de la población P_{k-1} (error+complejidad) y etiquetar aquellos que son no dominados.
- 5) Si el número de soluciones no dominadas es menor que 3, repetir
 - 1) Encontrar una solución no dominada entre las que no están etiquetadas.
 - 2) Etiquetar la solución como no dominada.
 hasta que haya 3 soluciones no dominadas etiquetadas.
- 6) Borrar de la población P_{k-1} todas las soluciones que sean dominadas.
- 7) Marcar el 20% del conjunto de entrenamiento como conjunto de validación para BP.
- 8) Repetir
 - 1) Seleccionar un individuo P_{k-1} aleatoriamente como padre principal α_1 , y dos individuos α_2, α_3 como padres secundarios (sin repetición).
 - 2) Aplicar el operador de cruce a cada neurona de la capa oculta y de la capa de salida, y con una probabilidad uniforme (0,1), si es menor que CR hacer
 - a) Aplicar (6.1) y (6.2) o (6.5) según la capa a la que pertenezca la neurona.
 - b) Aplicar (6.3) y (6.4) o (6.6) según la capa a la que pertenezca la neurona.
 Al menos se debe cambiar una variable.
 - 3) Aplicar el operador de mutación cada neurona de la capa oculta y de la capa de salida con una probabilidad uniforme (0,1) y si es menor que MR hacer
 - a) Aplicar (6.7) y (6.9) o (6.8) según la capa a la que pertenezca la neurona.
 - 4) Aplicar búsqueda local (BP) al hijo usando el conjunto de validación.
 - 5) Evaluar al hijo (error+complejidad).
 - 6) Si el hijo domina al padre principal entonces
 - a) Añadir hijo a la población P_{k-1} .
 mientras $\text{tamaño}(P_{k-1}) < M$
- 9) $k = k + 1$
- 10) hasta que no se cumpla la condición de parada.

Figura 3.4: Pseudocódigo del algoritmo MPANN.

Seguidamente (Paso 6), se eliminan todas las soluciones dominadas de la población.

A continuación (Paso 7), se marca un 20 % del conjunto de patrones de entrenamiento como conjunto de validación para la LS.

El siguiente paso (Paso 8) es el más importante de todo el algoritmo, ya que se trata de la generación de hijos. Este paso se repetirá hasta que la pobla-

3.5. El algoritmo MPANN

ción alcance un tamaño máximo, M , fijado de antemano.

La primera acción a realizar (Paso 8.1) es la selección aleatoria de tres padres. Uno de ellos se etiqueta como padre principal (α_1) y los otros dos como secundarios (α_2, α_3).

A continuación (Paso 8.2), tiene lugar la operación de cruce. En la operación de cruce se calcula aleatoriamente una probabilidad uniforme en el intervalo $(0, 1)$ para cada una de las neuronas de la capa oculta en conexión con la capa de entrada. Si el valor obtenido es menor que el valor de CR (*Crossover Probability*), se aplican las expresiones (3.1) y (3.2), donde w_{ih}^{hijo} se refiere, en el hijo, a los pesos asociados a cada una de las neuronas de entrada, i , con cada una de las neuronas de la capa oculta, h . ρ_h^{hijo} se refiere a la existencia o no en la capa oculta de las neuronas del nuevo hijo. Todo este proceso se hace para cada neurona de la capa oculta en conexión con la capa de entrada, siendo el número máximo de neuronas ocultas prefijado al inicio del algoritmo.

$$w_{ih}^{hijo} \leftarrow w_{ih}^{\alpha_1} + N(0, 1)(w_{ih}^{\alpha_2} - w_{ih}^{\alpha_3}) \quad (3.1)$$

$$\rho_h^{hijo} \leftarrow \begin{cases} 1 & \text{si } \rho_h^{\alpha_1} + N(0, 1)(\rho_h^{\alpha_2} - \rho_h^{\alpha_3}) \geq 0.5 \\ 0 & \text{en otro caso} \end{cases} \quad (3.2)$$

En el caso de que el valor aleatorio obtenido en el intervalo $(0, 1)$ no sea menor que CR , se aplican las expresiones (3.3) y (3.4).

$$w_{ih}^{hijo} \leftarrow w_{ih}^{\alpha_1} \quad (3.3)$$

$$\rho_h^{hijo} \leftarrow \rho_h^{\alpha_1} \quad (3.4)$$

Una vez finalizada la operación de cruce para todas las neuronas de la capa oculta en conexión con la capa de entrada, es el turno de las neuronas de capa oculta en conexión con la capa de salida. De nuevo se obtiene un valor aleatorio probabilístico uniforme en el intervalo $(0, 1)$, y si el valor obtenido es menor que el valor de CR , se aplica la expresión (3.5), donde w_{ho}^{hijo} significa el peso asociado a la conexión que va desde la neurona h de la capa oculta hasta la neurona o de la capa de salida del nuevo hijo. Esto se hace, al igual que antes, para todas las neuronas que haya en capa oculta en conexión con

la capa de salida.

$$w_{ho}^{hijo} \leftarrow w_{ho}^{\alpha_1} + N(0, 1)(w_{ho}^{\alpha_2} - w_{ho}^{\alpha_3}) \quad (3.5)$$

En el caso de que el valor aleatorio obtenido en el intervalo $(0, 1)$ no sea menor que CR , se aplica la siguiente expresión:

$$w_{ho}^{hijo} \leftarrow w_{ho}^{\alpha_1} \quad (3.6)$$

Durante la operación de cruce, al menos una variable del hijo se debe modificar para que sea distinto al padre principal.

A continuación se realiza la operación de mutación (Paso 8.3), calculándose una probabilidad uniforme $(0, 1)$ para cada una de las neuronas de capa oculta del hijo, tanto las que están en conexión con la capa de entrada como las que están en conexión con la capa de salida. Si el valor obtenido es menor que el valor de MR (*Crossover Probability*), se aplican las expresiones (3.7) y (3.9), o la expresión (3.8) si el cambio que se va a producir es para una conexión de la capa de oculta con la capa de entrada, o si es para una conexión de la capa de oculta con la capa de salida respectivamente. Si el valor aleatorio obtenido es mayor que MR , no se hace la mutación. La nomenclatura que se sigue es la misma que para la operación de cruce.

$$w_{ih}^{hijo} \leftarrow w_{ih}^{hijo} + N(0, \text{porcentaje mutación}) \quad (3.7)$$

$$w_{ho}^{hijo} \leftarrow w_{ho}^{hijo} + N(0, \text{porcentaje mutación}) \quad (3.8)$$

$$\rho_h^{hijo} \leftarrow \begin{cases} 1 & \text{si } \rho_h^{hijo} = 0 \\ 0 & \text{en otro caso} \end{cases} \quad (3.9)$$

Una vez realizadas las operaciones de cruce y mutación, se aplica al hijo resultante la LS (Paso 8.4), usando el algoritmo de retropropagación BP, aplicando el conjunto de validación del paso 7.

Este hijo se evalúa y se añade a la población si presenta una relación de dominancia con respecto al padre principal (Paso 8.5 y Paso 8.6)

Este proceso se repetirá a lo largo de las generaciones hasta que se cumpla la condición de parada (Paso 10).

3.6. El algoritmo MPDE

3.6. El algoritmo MPDE

A continuación exponemos nuestro algoritmo basado en DE para multiclasi-ficación de patrones usando el concepto de dominancia de Pareto. Dicho algoritmo se llama MPDE (*Memetic Pareto Differential Evolution*) ?. Nuestro procedimiento, al igual que el algoritmo MPENSGAII descrito en el capítulo ??, evoluciona simultáneamente los pesos y la arquitectura de la red, y se encarga de diseñar modelos de red para multi-clasificación de patrones.

MPDE obtiene diferentes conjuntos de clasificadores no dominados que presentan un buen balance entre precisión y *MS* (ver capítulo ??), que son los dos objetivos a optimizar.

La población de individuos está sujeta a operaciones de cruce y de mutación. En cuanto a la codificación de los modelos de red, se sigue la misma codificación explicada en los algoritmos CBFEP y MPENSGAII del capítulo ?? y ?? respectivamente.

MPDE está hibridado con un algoritmo de LS, y utiliza funciones de base sigmoides (SUs).

Para llevar a cabo un proceso de elitismo, MPDE se basa en algunos aspectos de NSGAII ?, utilizando como característica más representativa el ordenamiento rápido de no-dominados para la obtención del frente de Pareto.

Con respecto a la diversidad se utiliza la distancia *crowding* de NSGAII, para el caso en que haya que completar la población hasta alcanzar un número determinado de individuos, y también una ecuación de cálculo de la distancia de un individuo a los dos vecinos más cercanos, la cual comentaremos en las siguientes secciones.

En cuanto a la variante de la DE, se trata de la variante *DE/rand/1/bin*.

3.6.1. Funciones objetivo

Las funciones objetivo a optimizar con MPDE son las mismas que se utilizan con MPENSGAII, E y MS , ya que pensamos que un buen clasificador debería obtener un alto nivel de precisión global, así como un aceptable nivel de clasificación para cada clase de un determinado problema:

- **Objetivo 1:** La mínima sensibilidad de todas las clases de un problema, MS :

$$A_1(g, \Theta) = MS(g)$$

- **Objetivo 2:** La entropía cruzada como medida de error global, E .

Concretamente, como función de aptitud a la hora de evaluar un individuo se usará la siguiente expresión:

$$A_2(g, \Theta) = \frac{1}{1 + E(g, \Theta)},$$

es decir, maximizar una transformación estrictamente decreciente de E .

A la hora de asignar una clase a una nueva observación se sigue el esquema “1 de Q ” explicado en los algoritmos CBFEP y MPNENSGAII.

3.6.2. Operadores

En cuanto a los operadores de cruce y mutación se utilizarán los mismos que los del algoritmo MPANN, pero adaptados a nuestra representación.

Las expresiones (3.1) a (3.6) representan el operador de cruce y las expresiones (3.7) a (3.9) el operador de mutación. El significado de la nomenclatura seguida es la misma que la explicada en el algoritmo MPANN.

3.6.3. Búsqueda local

Como algoritmo de búsqueda local usamos el algoritmo iRprop+ utilizado con el algoritmo MPENSGAII (ver sección ?? del capítulo ??). En la siguiente sección se explica detalladamente cuándo se utiliza el algoritmo iRprop+ y se comenta cada una de las etapas de nuestra metodología.

3.6. El algoritmo MPDE

3.6.4. Etapas y aspectos relevantes de MPDE

En la figura 3.4 se muestran las etapas del algoritmo MPDE, que pasamos a comentar:

El algoritmo comienza con la creación de una población de individuos tomados al azar, siendo M el tamaño de la población (Paso 1).

Empieza el proceso evolutivo hasta que se cumpla la condición de parada (Paso 3). Los individuos se evalúan en base a las dos funciones objetivo que guían el algoritmo y se realiza un ordenamiento rápido de no dominados equivalente al del algoritmo NSGAII ?. Se etiquetan entonces aquellos que sean no dominados. (Paso 4)

Si el número de soluciones no dominadas es menor que 3 (Paso 5), entonces se repite el siguiente proceso hasta que haya al menos 3 soluciones: Encontrar una solución no dominada entre las que no están etiquetadas, en función del orden asignado en el ordenamiento rápido de no dominados. En caso de empate en orden, se utiliza el valor de la distancia *crowding* de NSGAII (Paso 5.1) y se elige la solución con mayor distancia. A continuación se etiqueta el individuo escogido como no dominado (Paso 5.2).

Si el número de soluciones no dominadas ya era de al menos 3 individuos, se comprueba si el número de soluciones es mayor que $M/2$ (Paso 6). Si es así, se calcula la distancia de cada individuo a sus dos vecinos más cercanos (Paso 6.1), y se elimina el individuo con menor distancia (Paso 6.2). El cálculo de la distancia viene dado por:

$$D(x) = \frac{(\min\|x - x_i\| + \min\|x - x_j\|)}{2}$$

siendo x el individuo al que se le va a calcular la distancia a sus dos vecinos más cercanos, siendo estos x_i y x_j . Esto nos permite mantener un mayor grado de diversidad y que el algoritmo no quede estancado en el caso de que el número de soluciones no dominadas sea cercano a M , ya que el tamaño de la población suele ser pequeño, con lo que obtendríamos de una generación a otra muy pocos individuos mejorados.

- 1) Generar una población P_0 , asignando los pesos de cada individuo en función de una distribución uniforme en dos intervalos: $[-5, 5]$ para las conexiones entre capa de entrada y capa oculta y $[-10, 10]$ para conexiones entre la capa oculta y salida.
- 2) $k = 1$; creados = 0
- 3) Repetir
 - 4) Evaluar los individuos de la población P_{k-1} en base a A_1 y A_2 , y etiquetar aquellos que son no dominados (ordenamiento rápido de no dominados).
 - 5) Si el número de soluciones no dominadas es menor que 3 entonces

Repetir

 - 1) Encontrar una solución no dominada entre las que no están etiquetadas.
 - 2) Etiquetar la solución como no dominada.

hasta que haya 3 soluciones no dominadas etiquetadas.
 - 6) Sino si el número de soluciones no dominadas es mayor que $(M/2)$ entonces

Repetir

 - 1) Calcular la distancia de cada individuo con su dos vecinos más cercanos.
 - 2) Eliminar el individuo con menor distancia.

hasta que el número de soluciones no dominadas sea igual a $(M/2)$.
 - 7) Borrar de la población P_{k-1} todas las soluciones que sean dominadas.
 - 8) Repetir
 - 1) Seleccionar un individuo P_{k-1} aleatoriamente como padre principal α_1 , y dos individuos α_2, α_3 como padres secundarios (sin repetición).
 - 2) Aplicar el operador de cruce a cada conexión existente entre capa de entrada y capa oculta, y entre la capa oculta y la capa de salida, con una probabilidad uniforme (0,1) y si es menor que CR hacer
 - a) Aplicar (6.1) y (6.2) o (6.5) según la capa a la que pertenezca la neurona.
 - Sino
 - b) Aplicar (6.3) y (6.4) o (6.6) según la capa a la que pertenezca la neurona.
 - 3) Evaluar al hijo obtenido en base a A_1 y A_2 .
 - 4) Si es igual al padre principal entonces

Se cambia un enlace aleatoriamente añadiéndole un valor de una distribución Gaussiana $N(0,1)$.
 - 5) Aplicar el operador de mutación a cada conexión existente entre la capa oculta y la capa de salida, con una probabilidad uniforme (0,1), y si es menor que MR hacer
 - a) Aplicar para cada neurona en capa oculta la ecuación (6.9)
 - 6) El hijo se ha creado, creados = creados + 1
 - 7) Evaluar al hijo en base a A_1 y A_2 .
 - 8) Si el hijo domina al padre principal entonces
 - a) Aplicar búsqueda local con iRprop+ al hijo.
 - b) Añadir hijo a la población P_{k-1} .
 - Sino si no hay relación de dominancia entre padre e hijo entonces
 - c) Añadir hijo a la población P_{k-1} .
 - Sino si creados=100 (aquí el padre principal domina al hijo)
 - d) Añadir el mejor de los 100 hijos almacenados en base a A_1 .
 - e) creados = 0
 - Sino
 - f) El candidato es descartado.

mientras $tamano(P_{k-1}) < M$
 - 9) $k = k+1$
 - 10) hasta que no se cumpla la condición de parada.

Figura 3.5: Pseudocódigo del algoritmo MPDE.

3.6. El algoritmo MPDE

En el paso 7 se eliminan de la población actual todas las soluciones no etiquetadas, es decir, las soluciones dominadas.

Se pasa ahora a completar la población actual para prepararla para la siguiente generación hasta que el número de individuos sea igual a M (Paso 8).

Primero se selecciona aleatoriamente un individuo como padre principal y otros dos como padres secundarios, todo ello sin repetición, para que los 3 sean distintos (Paso 8.1).

A partir de esos tres individuos realizamos una serie de operaciones hasta obtener un nuevo hijo que se añade a la población. En primer lugar aplicamos el operador de cruce a partir del padre principal y a partir de los padres secundarios, con una probabilidad de cruce uniforme designada como CR (Paso 8.2). El hijo que se obtenga, tendrá características de los tres padres.

El hijo obtenido se evalúa en base a las dos funciones objetivo que guían al algoritmo (Paso 8.3), y si el individuo es igual al padre porque no se hayan producido cambios con las operaciones realizadas anteriormente, se le fuerza a cambiar aleatoriamente un enlace, añadiéndole un valor de una distribución Gaussiana $N(0, 1)$ (Paso 8.4).

A continuación, se aplica al hijo el operador de mutación en cada una de sus neuronas de la capa oculta (Paso 8.5). Al comenzar el algoritmo se establece un número máximo de neuronas como en el algoritmo MPENSGAII (ver capítulo ??). Para el total del máximo de neuronas, si la neurona i existe se elimina, y si no, se añade, estableciendo enlaces y pesos de la misma forma que la mutación añadir neurona del algoritmo MPENSGAII. Todo ello aplicando la mutación con una probabilidad determinada.

Cuando el nuevo hijo se ha creado, aumentamos en 1 el valor de una variable llamada “creados”, que nos servirá, en un momento dado de la evolución. Concretamente cuando se hayan creado muchos hijos y ninguno de ellos por las circunstancias que se explican a continuación se pueda añadir a la población actual (Paso 8.6).

Evaluamos al hijo en base a las dos funciones objetivo que guían a MPDE (Paso 8.7).

Si el hijo domina al padre principal se le aplica la LS con el algoritmo iR-prop+ y se añade a la población actual (Paso 8a y 8b). En caso contrario, si no hubiera relación de dominancia entre padre e hijo, también se añade el hijo a la población actual (Paso 8c). En caso contrario, si el padre principal domina al hijo, se comprueba si *creados* = 100. En ese caso se elige el mejor de los 100 hijos almacenados, en base a la función objetivo A_1 , y la variable *creados* se establece a 0 (Paso 8d y 8e). Sino se produce nada de lo anterior, el candidato es descartado (Paso 8f). El paso 8 se repite entero hasta completar el tamaño de la población que se establece en la variable M .

Cuando se complete el tamaño de la población, ésta queda preparada para la siguiente generación (Paso 9), y el proceso evolutivo continúa hasta que se cumpla la condición de parada, que en nuestro caso es un número determinado de generaciones (Paso 10).

3.6.5. Diferencias con el algoritmo MPANN

Las diferencias fundamentales con respecto al algoritmo MPANN ? son las siguientes:

- El operador de cruce que nosotros utilizamos, también calcula aleatoriamente una probabilidad uniforme en el intervalo (0, 1), y si el valor obtenido es menor que el valor de CR no se aplica el operador. La diferencia está en que nuestro método no aplica el valor de probabilidad obtenido aleatoriamente a todos los enlaces y neuronas de la capa oculta, sino que utilizamos un nuevo valor aleatorio dentro del intervalo uniforme definido para cada neurona y no para la capa oculta entera, como en el caso de MPANN. En este caso, nuestro operador de cruce es menos agresivo con los cambios en las ANNs, ya que en alguna ocasión puede que el valor aleatorio obtenido a partir de una probabilidad uniforme no sea menor que CR , con lo que la neurona que se esté tratando en ese momento no cambia.

3.6. El algoritmo MPDE

- La probabilidad de mutación MR , también se utiliza de manera independiente, al igual que en el cruce, para cada neurona, y no para la capa oculta entera, como en el caso del algoritmo MPANN.
- La manera en que se añaden los individuos a la población en el algoritmo MPANN (solo los que dominan al padre principal), hace que el algoritmo pueda quedar estancado durante un buen número de generaciones (se ha comprobado experimentalmente) hasta que se pueda añadir un nuevo hijo. Nosotros añadimos individuos de una manera más “relajada”, de manera que hijos que no dominen al padre principal tienen la opción de añadirse a la población. De esta manera también se reduce el coste computacional sin disminuir la calidad de los resultados.

3.6.6. Diseño experimental

Para analizar el rendimiento de MPDE hemos utilizado 6 conjuntos de datos del repositorio de la UCI ?.

En la tabla 3.1 se muestran las características de cada conjunto: Número total de patrones por cada conjunto de datos, número de patrones en entrenamiento y en generalización, número de variables de entrada, número de clases, número total de patrones por clase y valor de p^* .

Tabla 3.1: Características de los conjuntos de datos de la UCI.

Conjunto	Patrones entrena.	Patrones generaliz.	Variables de entrada	Clases	Patrones por clase	p^*
Autos	205	152	53	72	6	67-3-22-54-32-27
Balance	625	469	156	4	3	288-49-288
BreastC	286	215	71	15	2	201-85
HeartStatlog	270	202	68	13	2	150-120
Newthyroid	215	161	54	5	3	150-35-30
Pima	768	576	192	8	2	500-268

El diseño experimental consiste en una partición estratificada del conjunto de datos con $3n/4$ patrones para el conjunto de entrenamiento y $n/4$ patrones para el conjunto de generalización, siendo n el tamaño del conjunto.

El proceso de obtención de resultados es el mismo que el utilizado con MPENS-GAII (ver sección ?? del capítulo ??). Una vez se forma el frente de Pareto

se utilizan dos estrategias de selección automática de individuos, el mejor modelo en E y el mejor modelo en MS (extremos del frente de Pareto). En cada ejecución del algoritmo (hacemos 30, dado que el proceso de entrenamiento de la red es estocástico), una vez que tenemos el frente de Pareto de la última generación del proceso evolutivo, se escogen los extremos del frente en entrenamiento. Esto es, el mejor individuo en E , y el mejor individuo en MS . A estos individuos se les llamamos individuo EI , para el primer caso, e individuo MSI , para el segundo. Cuando tenemos los individuos del paso anterior calculamos su valor de C y de MS , sobre el conjunto de generalización. De esta manera, tenemos para los extremos del frente dos pares de valores, $EI = (C_{EI}, MS_{EI})$ y $MSI = (C_{MSI}, MS_{MSI})$ de una ejecución de las 30 realizadas.

Al repetirse el proceso anterior 30 veces obtenemos la media y la desviación típica de los dos pares de valores para los individuos EI y MSI , es decir, $\overline{EI} = (\overline{C}_{EI}, \overline{MS}_{EI})$ y $\overline{MSI} = (\overline{C}_{MSI}, \overline{MS}_{MSI})$, de forma que la primera expresión muestra el rendimiento medio obtenido teniendo en cuenta solo los mejores individuos en E , mientras que la segunda expresión muestra el rendimiento medio teniendo en cuenta solo los mejores individuos en MS . A la manera de obtener automáticamente el rendimiento medio teniendo en cuenta los mejores individuos en E (parte superior del frente) le hemos llamado MPEDEE, a y la forma de obtener el rendimiento medio teniendo en cuenta los mejores individuos en MS (parte inferior del frente) la hemos llamado MPDES.

La probabilidad de cruce se estableció a $CR = 0.8$ y la de mutación a $MR = 0.1$, que es la adoptada por Abbass en MPANN, y el tamaño de la población se estableció como $M = 25$.

Para iRprop+, los parámetros adoptados son $\eta^- = 0.5$ (tamaño de paso para el factor de decremento), $\eta^+ = 1.2$ (tamaño de paso para el factor de incremento), $\Delta_0 = 0.0125$ (valor inicial de tamaño de paso para los pesos, Δ_{ij}), $\Delta_{min} = 0$ (tamaño mínimo de paso para los pesos), $\Delta_{max} = 50$ (tamaño máximo de paso para los pesos), $Epochs = 5$ (número de épocas para la optimización local).

3.6. El algoritmo MPDE

3.6.7. Resultados

Hemos comparado MPDE con nuestro algoritmo MPENSGAI y con la metodología SVM, a partir del algoritmo SMO que proporciona Weka¹?

La tabla 3.2 presenta los valores de media y desviación típica para C y MS obtenidos de los mejores modelos en E en cada ejecución. Observar que en Balance y Breast Cancer, el algoritmo MPDES obtiene los mejores valores en MS , encontrándose muy cercano al algoritmo MPENSGAI en valores de C . En Autos, el mejor resultado en C lo obtiene MPDEE, pero el mejor resultado en MS lo consigue el algoritmo MPENSGAII. En Newthyroid, MPDE obtiene los mejores valores en MS y C , y en Pima y Heart Statlog, MPDES obtiene los mejores valores en MS , y muy similares en C a los que obtiene MPENSGAIIE.

Tabla 3.2: Resultados estadísticos para MPDE, MPENSGAI y SVM en media y desviación típica sobre el conjunto de generalización para C y MS .

Conjunto	Algoritmo	$C(%)$	$MS(%)$	Conjunto	Algoritmo	$C(%)$	$MS(%)$
Autos	MPDEE	68.79 ± 5.59	28.75 ± 21.40	Balance	MPDEE	91.43 ± 1.01	54.36 ± 26.25
	MPDES	64.15 ± 5.63	12.26 ± 20.54		MPDES	91.41 ± 1.53	87.42 ± 4.32
	MPENSGAIIE	66.67 ± 4.07	39.64 ± 14.92		MPENSGAIIE	94.01 ± 1.52	42.66 ± 17.00
	MPENSGAIIIS	66.04 ± 4.78	42.28 ± 10.98		MPENSGAIIIS	92.47 ± 2.16	83.72 ± 8.19
	SVM	67.92	0.00		SVM	88.46	0.00
BreastC	MPDEE	67.27 ± 2.71	38.09 ± 11.59	Newthy	MPDEE	96.66 ± 2.02	81.42 ± 10.74
	MPDES	65.39 ± 3.40	57.04 ± 7.01		MPDES	96.66 ± 1.84	81.64 ± 9.76
	MPENSGAIIE	69.34 ± 2.30	28.88 ± 9.09		MPENSGAIIE	95.12 ± 2.30	74.81 ± 10.07
	MPENSGAIIIS	63.99 ± 3.10	59.08 ± 6.57		MPENSGAIIIS	95.55 ± 2.15	75.07 ± 10.66
	SVM	64.79	23.81		SVM	88.89	55.56
Pima	MPDEE	78.59 ± 1.59	61.94 ± 4.10	HeartStlg	MPDEE	76.17 ± 1.41	61.11 ± 2.20
	MPDES	77.11 ± 2.20	73.12 ± 2.98		MPDES	76.27 ± 1.57	63.66 ± 2.37
	MPENSGAIIE	78.99 ± 1.80	60.44 ± 2.59		MPENSGAIIE	78.28 ± 1.75	61.88 ± 2.08
	MPENSGAIIIS	76.96 ± 2.08	72.68 ± 3.06		MPENSGAIIIS	77.5 ± 1.73	62.66 ± 2.38
	SVM	78.13	50.75		SVM	76.47	60.00

Los mejores resultados se muestran en **negrita** y los segundos mejores resultados se muestran en *cursiva*.

Como ejemplo gráfico en la figura 3.6, presentamos los resultados obtenidos por el algoritmo MPDE en el conjunto de datos Balance, y al igual que en el caso de MPENSAII (ver sección ?? del capítulo ??), están divididos en gráficos de entrenamiento (A_1, A_2), y en gráficos de generalización (A_1, C).

En nuestra opinión, el uso de la DE junto con métodos de LS usando E y MS como objetivos a optimizar, puede ser un nuevo punto de vista para tra-

¹<http://www.cs.waikato.ac.nz/ml/weka/>

tar problemas multi-clase en clasificación, con resultados muy prometedores.

En el siguiente capítulo se expone una aplicación realiza con el algoritmo MPENSGAII que estudiamos y detallamos en el capítulo ??, concretamente una aplicación sobre microbiología predictiva ?.

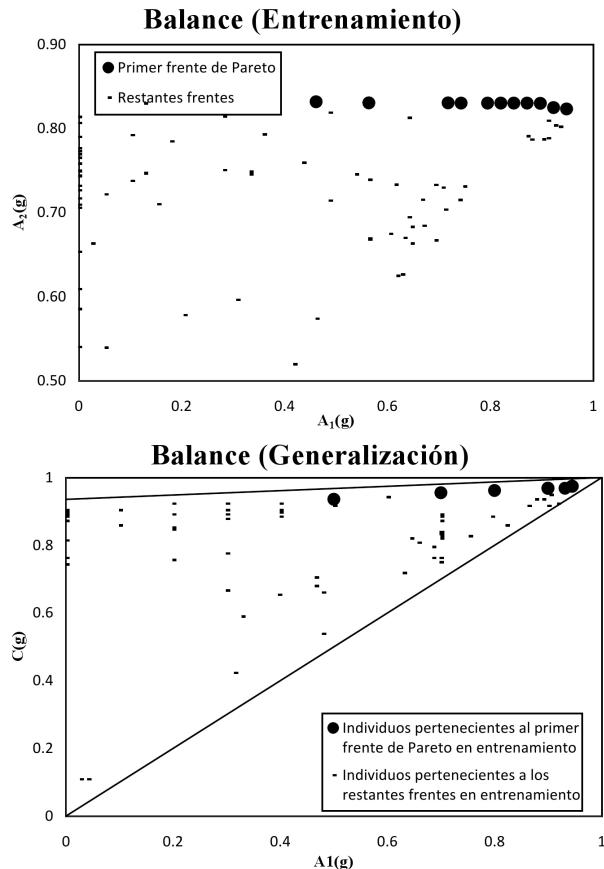


Figura 3.6: Frente de Pareto en entrenamiento (A_1, A_2), y valores asociados a (A_1, C) en generalización para el conjunto de datos Balance.

