Computación evolutiva para selección pesada de características en sistemas de detección de intrusiones

F. de Toro, P. García-Teodoro, J.E.Díaz-Verdejo, G.Maciá-Fernández.
Dept. de Teoría de la Señal, Telemática y Comunicaciones
ETS Ingenierías Informática y de Telecomunicaciones
Univ. de Granada
18071 Granada
ftoro@ugr.es, pgteodor@ugr.es, jedv@ugr.es, gmacia@ugr.es

Resumen

El presente trabajo aborda el uso de un algoritmo evolutivo con hacinamiento determinista para la selección pesada de características en un clasificador binario de k vecinas en el contexto del diseño optimizado de sistemas de detección de intrusiones basados en anomalías. La incorporación de una técnica de mantenimiento de diversidad (hacinamiento determinista) en el diseño del algoritmo evolutivo tiene como objeto potenciar la obtención de diferentes soluciones de optimización (subconjuntos de características), de manera que se flexibilice en lo posible la elección de las características a utilizar. El sistema se ha evaluado preliminarmente en una aplicación de detección de ataques de denegación de servicio.

1. Introducción

El aumento creciente de las amenazas a la seguridad ha hecho que el diseño de sistemas eficientes de detección de intrusión en redes sea esencial para salvaguardar a empresas e instituciones de costosos desembolsos en la restauración de daños causados por ataques informáticos, además de para garantizar continuidad en los negocios de las primeras. Un Sistema de Detección de Intrusiones (SDI) es un programa que analiza lo que ocurre o ha ocurrido en un sistema de red y trata de encontrar indicios de que un computador ha sido utilizado con fines maliciosos. Un SDI típicamente monitoriza el tráfico de datos que pasa por la red con vistas a generar una alerta cuando tiene lugar un evento de ataque. Tradicionalmente, se han diferenciado dos esquemas de detección diferentes: (1) Sistemas de detección de intrusiones basados en firmas: dichos sistemas tratan de encontrar la *firma* de determinados ataques en los datos monitorizados; (2) Sistemas de detección de intrusiones basados en anomalías: estos sistemas señalan cualquier desviación de un modelo de tráfico normal (previamente requieren elaborar dicho modelo). En general, la tendencia actual es centrarse en el estudio y diseño de este tipo de sistemas frente a los basados en firmas debido a su mejor comportamiento frente a la aparición de nuevos tipos de ataques. Los algoritmos de aprendizaje automático (*machine learning*) [1] pueden utilizarse para el diseño de sistemas de detección de intrusiones, tanto basados en firmas como en anomalías. Dichos algoritmos pueden ser entrenados con tráfico de red conocido (etiquetado), de manera que puedan reconocer tráfico desconocido con una cierta precisión. Uno de estos sistemas podría consistir en un clasificador binario con capacidad de discriminar dos grupos diferentes de observaciones: tráfico normal y tráfico anormal o malicioso (conteniendo algún tipo de ataque sin especificar). Dicho tipo de sistemas harían uso de ciertas características (*features*) extraídas del medio observado para realizar la decisión de clasificación. Encontrar las características que dan lugar a un porcentaje alto de clasificación correcta del tráfico monitorizado es un tema de importancia capital en tales sistemas. En este sentido, la *selección de características* [2] reduce el número de características que usa el clasificador para realizar su decisión, disminuyéndose así el costo computacional del proceso de clasificación al utilizar un sistema más simple, además de proporcionar información valiosa acerca de las características del tráfico de red más relevantes en el proceso de clasificación.

Los algoritmos evolutivos [3,4] son procedimientos de optimización estocástica que aplican un proceso de transformación inspirado en la evolución natural de las especies (operadores de mutación y cruce) a un conjunto de soluciones

codificadas del problema (población de individuos). Estos procedimientos han mostrado un gran éxito abordando problemas NP-completos, y problemas de optimización con varias soluciones [5,6] debido a su especial habilidad para explorar espacios de búsqueda grandes y capturar múltiples soluciones en una única ejecución.

En este contexto el presente trabajo aborda el diseño de un sistema de detección de intrusiones capaz de discriminar tráfico normal y tráfico anómalo mediante el uso de un clasificador binario basado en un algoritmo de k vecinas. Para el proceso de selección de características en dicho clasificador se utilizará un algoritmo evolutivo de hacinamiento determinista (*deterministic crowding*) [9]. El trabajo está organizado de la siguiente manera: la sección 2 presenta la metodología para detección de ataques basada en anomalías que se propone en este trabajo; la sección 3 muestra algunos resultados preliminares obtenidos con dicho sistema en una aplicación de detección de ataques de denegación de servicio [7]. Finalmente, la sección 4 se dedica a la discusión de resultados y ha establecer las conclusiones.

2. Metodología de detección de ataques

El esquema general de la metodología propuesta en este trabajo se representa en la Figura 1. En primer lugar, se extrae un vector de características n-dimensional que define el estado del entorno de red monitorizado para un periodo de tiempo dado. En principio, no hay restricción en la naturaleza de dichas características pudiendo ser estas extraídas de los *hosts* (ej. ficheros *logs* del sistema), o directamente de la monitorización de la red (ej. paquetes IP o, a nivel de la capa de transporte, de conexiones TCP, o en general, de flujos de datos).

A continuación, un clasificador binario de k-vecinas (cuyo funcionamiento se repasa en la sección 2.1) produce una salida positiva (tráfico anormal) o negativa (tráfico normal) a partir del vector de características. El problema de selección de las características que usará el clasificador (sobre un conjunto de características previamente definido) se formula como un problema de optimación abordado por un algoritmo evolutivo de la siguiente manera: una solución candidata del problema de optimización es un vector de pesos de la misma dimensión que el vector de características. Cada componente del vector peso tiene un valor entre 0 y 1. La componente i-ésima del vector peso expresa el grado de importancia de la característica i-ésima del vector de características. El algoritmo evolutivo parte primeramente de un conjunto de soluciones candidatas (población de individuos) cuyos valores son calculados aleatoriamente según una distribución uniforme. A dicha población se le aplican los operadores de transformación (mutación y cruce) [3], evaluándose la bondad (*fitness*) de las nuevas soluciones. En este caso la bondad de las soluciones viene dada por el rendimiento que un determinado pesado de características tiene en el clasificador. Un gran número de indicadores de rendimiento pueden utilizarse para evaluar el rendimiento de un determinado pesado de características, tales como la precisión de clasificación alcanzada o la sensibilidad en la detección de ataques (ambos indicadores están definidos en la sección 3). Dichos indicadores se calculan comparando las etiquetas de decisión del clasificador (tráfico normal o tráfico anómalo) con las etiquetas reales de dicho tráfico. Una vez calculadas las bondades de todas las soluciones candidatas, se aplica el operador de selección (típicamente se seleccionan las mejores soluciones candidatas con algún criterio opcional adicional para evitar el problema de la convergencia prematura). Normalmente este proceso se repite de manera iterativa hasta que se cumple alguna condición de parada del algoritmo, como por ejemplo que se completen un número dado de iteraciones (generaciones) o que la bondad de las soluciones encontradas no mejore sustancialmente respecto a las soluciones encontradas en la iteración precedente. Una vez concluida esta etapa de optimización con el algoritmo evolutivo se dispone de un conjunto de soluciones que pueden aplicarse para el pesado *on line* de las características del tráfico de la red, durante el funcionamiento normal del clasificador como sistema de detección de intrusiones. Tal sistema puede ser re-entrenado, por ejemplo con nuevos ataques no tenidos en cuenta inicialmente. Igualmente, admite fácilmente la eliminación de características (porque se concluya que no influyen en el rendimiento del clasificador), así como la adición de otras nuevas (que otros investigadores puedan encontrar importantes).

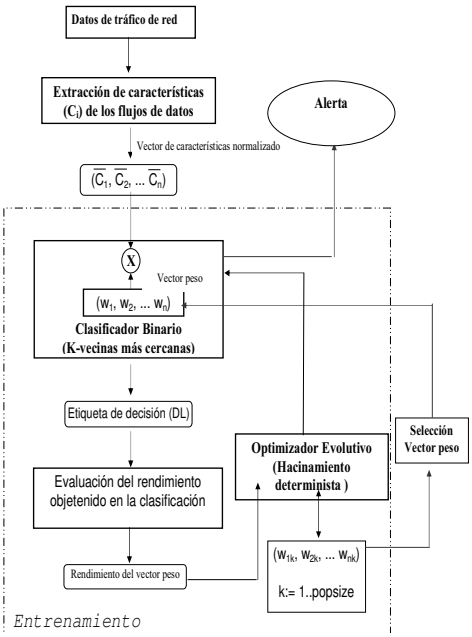

Figura 1. Optimización de un clasificador mediante un algoritmo evolutivo para selección pesada de características.

2.1 Clasificación mediante algoritmo de las K vecinas más cercanas

En este trabajo se ha escogido un algoritmo de k vecinas para realizar el proceso de clasificación debido a la simplicidad de su implementación. Es importante hacer notar, no obstante que la metodología representada en la Figura 1 puede utilizarse con diferentes algoritmos de clasificación de aprendizaje supervisado. Un algoritmo de las k vecinas más cercanas identifica la clase a la que pertenece una observación desconocida como la clase a la que pertenece la mayoría de las k observaciones conocidas más cercanas de entre aquellas utilizadas como modelo de referencia. Dicho algoritmo requiere, pues, el cálculo de la distancias euclídeas entre vectores de características, por lo cual todas las características se normalizan, por ejemplo, dividiendo cada característica original por su varianza en el modelo de referencia.

2.2 Algoritmo Evolutivo con Hacinamiento Determinista

El algoritmo evolutivo que se ha utilizado en este trabajo para la selección pesada de características es un algoritmo al que se ha incorporado una técnica de mantenimiento de diversidad denominada hacinamiento determinista [9]. Las técnicas de mantenimiento de diversidad favorecen que las soluciones encontradas por el algoritmo evolutivo sean diferentes en términos de distancia euclídea, es decir estén separadas en el espacio de características del problema. Encontramos dos ventajas fundamentales en este hecho: (1) puede obtenerse un mayor conocimiento del espacio de diseño del problema, en nuestro caso del espacio de características con el que se trabaja; (2) se dispone de un grado de flexibilidad para escoger aquellas soluciones de pesado más convenientes de acuerdo por ejemplo, a la facilidad de extracción de las características con un mayor pesado, y por lo tanto más decisivas en el proceso de clasificación. Por otro lado, se ha escogido la técnica de hacinamiento determinista por dos razones principales: (1) dicha técnica ha mostrado un gran rendimiento tanto en estudios con problemas de optimización sintéticos [8,10], como en otros dominios diferentes a la detección de intrusiones [15]; (2) Al contrario que otras técnicas como la técnica de aclarado (*Clearing*) [11] o compartición de bondad (*fitness sharing*) [4], no existe necesidad de determinar ningún parámetro de ajuste. El algoritmo utilizado se representa en el Algoritmo 1 . La función D(x,y) hace referencia a la distancia euclidea.

01 Seleccionar dos individuos p_1 y p_2 de la población (sin reemplazamiento)
02 Cruzar p_1 y p_2 para obtener h_1 y h_2
03 Mutar h_1 y h_2 para obtener c_1 y c_2
04 **Si** $[D(p_1,c_1)+D(p_2,c_2)] \leq [D(p_1,c_2)+D(p_2,c_1)]$
05 **Si** c_1 es mejor que p_1 **entonces** sustituir p_1 por c_1
06 **Si** c_2 es mejor que p_2 **entonces** sustituir p_2 por c_2
En otro caso
07 **Si** c_1 es mejor que p_2 **entonces** sustituir p_2 por c_1
08 **Si** c_2 es mejor que p_1 **entonces** sustituir p_1 por c_2

Algoritmo 1.

3. Caso de Estudio

Con el propósito de evaluar la metodología anterior, se ha utilizado el dataset reducido al 10% de la base de datos de DARPA [7], conteniendo 22 tipos diferentes de ataques que caen en alguno de los siguientes grupos:

- DoS (*Denial of Service*): el atacante elige algún recurso de memoria o computación y hace que esté demasiado ocupado o lleno como para atender peticiones legítimas, o simplemente deniega a usuarios legítimos el uso de dichos recursos (ej. inundación SYN, ping de la muerte, smurf, etc.)
- R2U (*Remote To User*): el atacante explota alguna vulnerabilidad para ganar acceso local no autorizado desde una máquina remota (ej. ataques de adivinación de *password*)
- U2R (*User To Root*): el atacante tiene acceso a la cuenta de un usuario (obtenida legítimamente o de otra manera) y usando dicha cuenta es capaz de ganar acceso al sistema como administrador utilizando algún agujero de seguridad del sistema.
- PROBE (*Probing*): el atacante monitoriza la red para reunir información o encontrar vulnerabilidades conocidas al sistema. Un atacante que ha reunido información sobre las máquinas y servicios disponibles en la red puede utilizar esta información para buscar puntos débiles del sistema (ej. *scan* de puertos)

Hay 41 características presentes en esta base de datos de tráfico: las primeras 9 son características "intrínsecas" extraídas directamente de la cabecera de los paquetes monitorizados. El resto de características han sido construidas tal y como se detalla en [12,13]. De esta manera, las características 10 a 22 son características basadas en contenido, obtenidas examinando la porción de datos de los flujos de datos (*payload*). Las características que van de la número 23 a la 41 son características basadas en tráfico y se calculan usando una ventana de tiempo. Dentro de este grupo las características que van de la número 23 a la número 31 usan una ventana de tiempo de dos segundos, mientras que las características que van desde la número 32 a la 41 utilizan una ventana de tiempo de 100 conexiones TCP. La razón para utilizar dos tipos de ventanas es que los ataques de denegación de servicio y los ataques PROBE normalmente realizan múltiples conexiones en un corto periodo de tiempo, mientras que los ataques R2U y U2R están embebidos en los datos de las conexiones TCP y normalmente conllevan una única conexión TCP.

En este trabajo sólo abordamos la detección de ataques de denegación de servicio. Por otro lado, solo se han tenido en cuenta características numéricas del conjunto original de características. De esta manera, únicamente 6 de las 9 características denominadas "intrínsecas" se consideran en los experimento. La Tabla 1 contiene un listado completo de las 38 características utilizadas para la base de datos. Como parte del preprocesamiento de la base de datos, se han eliminado los vectores de características duplicados. El entrenamiento y la evaluación ha sido realizado con el mismo conjunto de entrenamiento mediante el método *leaving one out* [14]. Se han realizado 20 ejecuciones de entrenamiento con conjuntos aleatorios de 1000 elementos de la base de datos (500 vectores de tráfico normal y 500 vectores con ataques).

Los resultados de clasificación caen en uno de los siguientes casos: (1) El algoritmo clasifica el tráfico como malicioso y el tráfico es de hecho malicioso (Verdadero Positivo, VP); (2) El algoritmo clasifica el tráfico como normal y de hecho es normal (Verdadero Negativo, VN); (3) El algoritmo clasifica el tráfico como malicioso cuando en realidad el tráfico es normal (Falso Positivo, FP); (4) el algoritmo clasifica el tráfico como normal cuando en realidad es malicioso (Falso Negativo, FN).

Se han considerado dos indicadores de rendimiento:

- Precisión de clasificación (C): representa el cociente del tráfico correctamente clasificado respecto del tráfico total.

$$C = \frac{VP + VN}{VP + VN + FP + FN} \quad (1)$$

- Sensibilidad (S): representa el cociente del tráfico malicioso detectado respecto del tráfico malicioso total.

$$S = \frac{VP}{VP + FN} \quad (2)$$

Tabla 1: Características del tráfico de red

#	Description
1	Duration of the connection
2	Bytes sent from source to destination
3	Bytes sent from destination to source
4	1 if connection is from/to the same host/port; 0 otherwise
5	# wrong fragment
6	# urgent packets
7	# "hot" indicators
8	# failed logins
9	1 if successfully logged in; 0 otherwise
10	# "compromised" conditions
11	1 if root shell is obtained; 0 otherwise
12	1 if "su root" command attempted; 0 otherwise
13	# "root" accesses
14	# file creations operations
15	# shells prompts
16	# operations on access control files
17	# outbounds commands in an ftp session
18	1 if the login belongs to the "hot" list; 0 otherwise
19	1 if the login is a "guess" login; 0 otherwise
20	# connections to same host in the last two seconds
21	# connections to same host and service as the current connection in the past two seconds
22	% of connections that have SYN errors
23	% of connections to same service having SYN error
24	% of connections that have REJ errors
25	% of connections to same service having REJ error
26	% of connections to the same service
27	% of connections to the different services
28	% of connections to different hosts
29	Count of connections having same dest.
30	Count of connections having same dest. and service
31	% of connections having the same dest. and service
32	% of different services on the current host
33	% of connections having same src port
34	% of connections to the same service coming from different hosts
35	% of connections to the current host having S0 err.
36	% of connections to the current host and specified service that have an S0 error
37	% of connections to current host having RST error.
38	% of connections to the current host and specified service that have an RST errors

Los indicadores C y S se evalúan mediante el método de *leaving one out*: en cada ciclo se selecciona un vector de la base de datos como elemento de test, se utiliza el resto de los vectores como modelo de referencia, se calcula la etiqueta del elemento de test (tráfico normal/malicioso) y se actualizan los contadores VP, VN, FP y FN una vez comparada dicha etiqueta con la etiqueta real del elemento de test. Cuando todos los elementos de la base de datos han sido utilizados como elementos de test, se calculan los indicadores C y S utilizando las expresiones (1) y (2) anteriores. Este proceso se repetiría para cada solución candidata (vector peso) hasta evaluar todas las soluciones candidatas de la población del algoritmo evolutivo. Es importante mencionar que únicamente se tiene en cuenta la precisión de clasificación para guiar la búsqueda del algoritmo evolutivo.

Operadores de Transformación

En la implementación del algoritmo evolutivo se ha utilizado un operador de cruce de un solo punto y dos tipos diferentes de operadores de mutación: mutación uniforme (ecuación 3) y mutación gausiana (ecuación 4). Ambos operadores se calculan respectivamente como:

$$w_j' = U(0,1) \tag{3}$$

$$\begin{aligned} \mu &= w_j \\ \sigma &= \sqrt{\min\left[(1-w_j), w_j\right]} \\ w_j' &= N(\mu, \sigma) \end{aligned} \tag{4}$$

Cada componente del vector peso tiene una probabilidad de transformarse por mutación de 0.6. Los dos operadores de mutación utilizados se usan con probabilidad de 0.5 cada uno. Por último, el algoritmo evolutivo se ejecuta durante 400 generaciones (iteraciones) con 50 soluciones candidatas, obteniéndose precisiones de clasificación entre 0.94 y 0.99 al final de la convergencia. Cuatro de las soluciones encontradas en una ejecución típica del algoritmo evolutivo y un valor de k=3 (número de vecinas) se muestran en la Tabla 2.

Tabla 2. Precisión de Clasificación y Sensibilidad de cuatro soluciones encontradas (vectores peso) para selección pesada de características en una ejecución típica del algoritmo evolutivo.

#Car.	Soluc. #1	Soluc. #2	Soluc. #3	Soluc. #4
1	0.02	0.80	0.37	0.01
2	0.73	**0.00**	**0.90**	0.75
3	0.21	0.75	0.14	0.20
4	0.07	0.06	0.29	0.06
5	0.83	0.96	1.00	0.82
6	0.33	0.85	0.87	0.24
7	0.92	0.46	0.47	0.92
8	0.99	0.18	0.18	0.99
9	0.47	0.40	0.75	0.25
10	0.40	0.10	0.47	0.40
11	0.87	1.00	0.22	0.87
12	**0.98**	**0.04**	0.02	0.98
13	0.42	0.63	0.95	0.42
14	0.75	0.63	0.21	0.96
15	0.60	0.23	0.95	0.99
16	0.94	0.91	0.25	0.59
17	0.46	0.25	0.72	0.91
18	0.99	0.48	0.62	0.50
19	0.61	0.75	0.54	0.12
20	0.90	0.56	0.58	0.86
21	0.52	0.03	0.91	0.75
22	0.30	0.60	0.46	0.74
23	0.75	0.91	0.52	0.50
24	0.45	0.66	0.03	0.05
25	0.75	0.29	0.93	0.62
26	0.15	0.75	0.95	0.16
27	0.25	0.76	0.80	0.68
28	0.19	0.09	0.08	0.90
29	0.64	0.08	0.33	0.58
30	0.45	0.80	0.49	0.89
31	0.75	0.25	0.59	0.43
32	0.75	0.53	0.99	0.95
33	0.12	0.10	0.58	0.29
34	0.90	0.04	0.61	0.76
35	0.42	0.02	0.29	0.78
36	0.50	0.73	0.51	0.20
37	0.42	0.62	0.94	0.29
38	0.98	0.89	0.41	0.93
C (%)	**97.2**	**97.3**	**97.2**	**94.4**
S (%)	**98.1**	**98.4**	**98.1**	**99.0**

Tal como puede observarse, las cuatro soluciones diferenciadas en la Tabla 2 involucran diferentes pesos para las mismas características, así por ejemplo, la característica #12 está pesada con un valor cercano a 1 (0.98) en la solución #1, y aparece al mismo tiempo pesada con un valor cercano al cero (0.04) en la solución #2, sin embargo ambas soluciones dan lugar a precisiones de clasificación y a sensibilidades igualmente altas. Fenómenos similares podemos apreciar en otras características (ej. característica #2 en solución #2 y solución #3). De igual manera, tenemos que a similar precisión de clasificación en las soluciones de la Tabla 2, las soluciones con mejor sensibilidad son preferibles a las soluciones con peor sensibilidad. En este sentido la solución #2 sería mejor que la solución #1 y la solución #3.

4. Conclusiones

Este trabajo aborda la selección pesada de características en un sistema de detección de intrusiones compuesto por un clasificador binario de k vecinas. La selección pesada de características se formula como un problema de optimización abordado por un algoritmo evolutivo de hacinamiento determinista. La evaluación de las soluciones encontradas se ha realizado mediante el método *leaving one out*. Dichas soluciones de ajuste proporcionan precisiones de clasificación superiores al 94% en el problema de detección de ataques de denegación de servicio abordado. La obtención de más de una solución durante la fase de optimización proporciona un grado de flexibilidad para la elección de las características a usar por el sistema de detección de intrusiones propuesto en este trabajo.

Agradecimientos

Este trabajo ha sido parcialmente subvencionado por el Ministerio de Ciencia y Tecnología Español a través del proyecto TSI2005-08145-C02-02.

Referencias

[1] Maheshkumar Sabhnani, Gürsel Serpen: Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context, Proceedings of the International Conference on Machine Learning; Models, Technologies and Applications, pp. 209-215, Las Vegas, 2003.

[2] Liu, H. and Motoda, H., Feature Selection for Knowledge Discovery and Data Mining, Kluwer, 1998.

[3] A. E. Eiben and J.E. Smith, Introduction to Evolutionary Computing, Natural Computing Series, Springer, 2003.
[4] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning. New York: Addison Wesley, 1989.
[5] J.P. Li, M.E Balazs, G.T. Parks, P.J. Clarkson, A species conserving genetic algorithm for multimodal function optimisation, Evolutionary Computation, Vol. 10, No. 3 (2002) 207-234.
[6] F.de Toro, J.Ortega, E.Ros, S.Mota, B.Paechter; Parallel Processing and Evolutionary Algorithms for Multiobjective Optimisation; Parallel Computing; Vol. 30, No.6 , pp. 721-739, 2004
[7] The UCI KDD Archive, Information and Computer Science, University of California, Irvine, "Kdd cup 1999 data set", http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.
[8] B. Sareni and L. Krähenbühl, Fitness Sharing and Niching Methods Revisited, IEEE Transaction on Evolutionary Computation, Vol. 2, No. 3, 1998.
[9] S.W. Mahfoud, Crowding and Preselection Revisited, Parallel Problem Solving from Nature 2, R. Manner, B. Manderick (Eds.), Elsevier Science Publishers, Amsterdam, pp. 27-36, 1992.
[10] S.W. Mahfoud, A Comparison of Parallel and Sequential Niching Methods, Proceedings of the Sixth International Conference on Genetic Algorithms, Morgan Kauffman, San Mateo, CA, 1995.
[11] A. Petrowski. "A Clearing Procedure as a Niching Method for Genetic Algorithms". In Proc. 1996 IEEE Int. Conf. Evolutionary Computation, Nagoya, Japan, pp.798-803, 1996.
[12] S. Stolfo, W. Lee, A. Prodromidis, and P. Chan, "Cost-based modeling for fraud and intrusion detection: Results from the JAM project", in Proc. DARPA Information Survavility Conference and Exposition, 2000, Vol. II, pp. 1130-1144, IEEE Computer Press.
[13] W. Lee, S.Stolfo and K.W. Mok, "A data mining framework for building intrusion detection models", in IEEE Symposium on Security and Privacy, 1999, pp. 120-132.
[14] D. Hand, Discrimination and Classification, Wiley & Sons, New York, 1981.
[15] F. de Toro, E.Ros, S.Mota, J.Ortega, "Evolutionary Algorithms for Multiobjective and Multimodal Optimisation of Diagnostic Schemes", IEEE Transactions On Biomedical Engineering, Vol. 53, No.2, pp. 178-189, 2006