ISSN: 2007-9958

REVISTA IZTATL COMPUTACIÓN

1. Selección de Modelo Completo en Conjuntos de Datos de gran Volumen

10. Aprendizaje de Modelos Difusos para Predicción de Series de Tiempo

27. Programación Genética en Problemas Multidimensionales

37. Aprendizaje con Hadoop

45. Selección de Estrategias en Juegos Deportivos de Equipo

54. Reconocimiento Bimodal de Emociones en un Ambiente Inteligente de Programación

Año 4 No. 8
Julio-Diciembre 2015
Publicación semestral
Reserva de Derechos No. 04-2014-070814424700-203

Universidad Autónoma de Tlaxcala
Facultad de Ciencias Básicas, Ingeniería y Tecnología

Universidad Autónoma de Tlaxcala
Facultad de Ciencias Básicas, Ingeniería y Tecnología

Mtro. Rubén Reyes Córdoba
Rector

Dr. Luis Armando González Placencia
Secretario Académico

Mtra. María Samantha Viñas Landa
Secretaria de Investigación Científica y Posgrado

Lic. Edilberto Sánchez Delgadillo
Secretario de Extensión Universitaria y Difusión Cultural

Mtro. José Antonio Durante Murillo
Secretario Técnico

Mtro. Efraín Ortiz Linares
Secretario Administrativo

Dr. Ernesto Meza Sierra
Secretario de Autorrealización

Mtro. Marlon Luna Sánchez
Coordinador de la División de Ciencias Básicas, Ingeniería y Tecnología

Dr. Sergio Eduardo Algarra Cerezo
Coordinador General de Cuerpos Académicos

Mtro. Carlos Santacruz Olmos
Director de la Facultad de Ciencias Básicas, Ingeniería y Tecnología

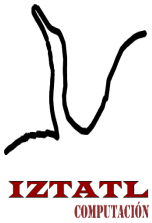
Mtro. Roberto Carlos Cruz Becerril
Secretario de la Facultad de Ciencias Básicas, Ingeniería y Tecnología

Dr. Alberto Portilla Flores
Coordinador de Posgrados en Computación y Electrónica

Mtro. Juventino Montiel Hernández
Coordinador de Ingeniería en Computación

Revista Iztatl Computación

Revista Iztatl Computación, año 4, No. 8, Julio-Diciembre 2015, es una publicación semestral editada por la Universidad Autónoma de Tlaxcala en coordinación con la Facultad de Ciencias Básicas, Ingeniería y Tecnología. Calle del Bosque s/n Colonia Tlaxcala centro C.P. 90000, Tlaxcala, Tlax, México. Teléfono (246) 4621422, http://ingenieria.uatx.mx/iztatl-computacion/35-2/, iztatl.computacion@gmail.com. Editor Responsable: Marva Angélica Mora Lumbreras. Reserva de Derechos al Uso Exclusivo No.04-2014-070814424700-203, ISSN: 2007-9958, ambos otorgados por el Instituto Nacional del Derecho de Autor. Responsables de la última actualización de este número, Universidad Autónoma de Tlaxcala en coordinación con la Facultad de Ciencias Básicas, Ingeniería y Tecnología. Calle del Bosque s/n Colonia Tlaxcala centro C.P. 90000, Tlaxcala, Tlax, México. Teléfono (246) 4621422, Dra. Marva Angélica Mora Lumbreras, fecha de última modificación, 12 de noviembre de 2015.



Editorial

La revista Iztatl Computación es una revista semestral que inició en el 2012, desde su inicio ha buscado tener publicaciones de alta calidad. En este número Iztatl Computación se congratula al presentar una edición especial con seis trabajos seleccionados del Seminario Nacional de Aprendizaje e Inteligencia Computacional SNAIC 2015, descritos brevemente a continuación:

- El proyecto *Selección de Modelo Completo en Conjuntos de Datos de gran Volumen* realizado por Díaz Pacheco A., González Bernal J. A., Reyes García C. A. tiene como objetivo procesar grandes bases de datos como Facebook y Twitter, bajo el enfoque de BIG DATA por medio de técnicas computacionales inteligentes buscando configuraciones óptimas de los hiperparámetros, conocido como enfoque de selección del modelo.

- En el artículo *Aprendizaje de Modelos Difusos para Predicción de Series de Tiempo* de Juan J. Flores, Félix Calderón, José Ortiz y Carlos Lara se propone una mejora para el algoritmo de vecinos más cercanos por medio de una modificación del cálculo de la distancia a través de funciones de membresía de lógica difusa. Se realizan comparaciones contra redes neuronales resultando más efectivo el método de vecinos más cercanos.

- En el proyecto *Programación Genética en Problemas Multidimensionales* realizado por Mario Gradd y Eric S. Téllez se realiza un análisis del estado del arte en materia de programación genética. Se analiza principalmente el tratamiento para problemas de alta dimensión. Se toman como casos de estudio dos problemas reales con diferentes ejemplares a analizar. Finalmente se propone un nuevo enfoque de programación genética para tratar problemas específicos comparables a los casos de estudio discutidos.

- En el trabajo *Aprendizaje con Hadoop* de Eugenia Méndez Macías y René MacKinney Romero se comparan medidas de eficacia (exactitud y F-M) de tres algoritmos de clasificación (Bayes Ingenuo, Bosques Aleatorios, K-vecinos más cercanos) desarrollados en Hadoop, aplicados en la base de datos de correo Enron (Ham/Spam).

- En el trabajo *Selección de Estrategias en Juegos Deportivos de Equipo* de Matías Alvarado se aplican modelos matemáticos de equilibrio de Nash y el de eficiencia de Pareto para modelar estrategias cooperativas en los juegos mencionados, cuya traducción algorítmica se usa en simulaciones computacionales de duelos entre equipos.

- En el artículo *Reconocimiento Bimodal de Emociones en un Ambiente Inteligente de Programación* de María Lucía Barrón Estrada, Margarita Aranda Ortega se describen los resultados obtenidos de los estados afectivos experimentados por un grupo de estudiantes, en interacción con un Sistema Tutor Inteligente (STI) orientado a la generación de código para la solución de problemas en el Lenguaje Java.

Conociendo la calidad de los artículos los invito a que disfruten de esta edición, al mismo tiempo invito a investigadores, académicos y estudiantes del área de computación para que continúen sometiendo sus artículos en esta revista.

Editora Responsable
Dra. Marva Angélica Mora Lumbreras

Índice

1. Selección de Modelo Completo en Conjuntos de Datos de Gran Volumen
Díaz Pacheco A., González Bernal J. A., Reyes García C. A.

10. Aprendizaje de Modelos Difusos para Predicción de Series de Tiempo
Juan J. Flores, Félix Calderón, José Ortiz, Carlos Lara

27. Programación Genética en Problemas Multidimensionales
Mario Gradd, Eric S. Téllez

37. Aprendizaje con Hadoop
Eugenia Méndez Macías, René MacKinney Romero

45. Selección de Estrategias en Juegos Deportivos de Equipo
Matías Alvarado

54. Reconocimiento Bimodal de Emociones en un Ambiente Inteligente de Programación
María Lucía Barrón Estrada, Margarita Aranda Ortega

Selección de Modelo Completo en Conjuntos de Datos de Gran Volumen

Díaz Pacheco A., González Bernal J. A., Reyes García C. A.

Instituto Nacional de Astrofísica, Óptica y Electrónica
Luis Enrique Erro # 1, Sta María Tonanzintla, 72840 Puebla, Pue., México
{diazpacheco,jagonzalez,kargaxxi}@ccc.inaoep.mx http://www.inaoep.mx/

Recibido: 15 de Octubre de 2015, Aceptado: 30 de Octubre de 2015, Versión final: 15 de Noviembre de 2015

Resumen El término Big Data hace referencia a bases de datos de gran magnitud que no pueden ser procesadas con los equipos y algoritmos convencionales. Parte de las tareas de análisis son realizadas con técnicas de aprendizaje automático, por lo cual se ha buscado mejorar su precisión predictiva a través de extensiones y la configuración óptima de sus hiperparámetros. Este último enfoque es conocido como selección de modelo y existen varios trabajos que lo abordan pero relativos a bases de datos convencionales. El presente trabajo tiene como objetivo investigar este problema siendo la principal contribución un conjunto de algoritmos para la selección de modelo completo en Big Data.

Palabras Clave: Big Data, MapReduce, Selección de modelo completo

1. Introducción

En muchas de las áreas de la actividad humana existe un gran interés en el análisis y almacenamiento de información, la cual es generada cada vez en mayores cantidades. Con el advenimiento de nuevas tecnologías como las redes sociales, la cantidad y la variedad de la información se incrementó a escalas sin precedentes. Se estima que en 2013 Twitter y Facebook generaron diariamente 7 y 10 terabytes de información respectivamente [22]. Este fenómeno popularizó el término Big Data, el cual es

utilizado para referirse a cantidades de datos que son demasiado grandes como para ser procesadas por los algoritmos y hardware convencionales [4]. Una definición muy extendida describe el concepto por medio de 3 rasgos distintivos de la información: Volumen, Velocidad y Variedad [6] [7] [13]. Volumen por la gran cantidad de datos, la Velocidad hace referencia a la rapidez con la que deben ser procesados y la Variedad indica que la información puede encontrarse en diversos formatos. Existe una gran cantidad de trabajos donde se ha empleado el paradigma de Big Data. Por ejemplo Mukkamala et al. [15] crearon un modelo de análisis de sentimientos mediante información obtenida de 11,348 publicaciones del perfil de Facebook de una compañía y Robertson et al. [18] realizaron un análisis de patrones de participación en mensajes de los muros de Facebook de Barack Obama, Hillary Clinton y John McCain. Las técnicas de aprendizaje automático son utilizadas para el analisis de información y se han realizado extensiones para poder procesar cantidades de información de mayor tamaño en menor tiempo. Algunos factores que tienen un gran impacto en la mejora de la capacidad de generalización de un algoritmo de clasificación son: la selección de características y el pre procesamiento de los datos. Estos, en combinación con la selección del algoritmo de clasificación y sus hiperparámetros conforman el paradigma de la selección de modelo completo [9], el cual ha sido entendido como un problema de optimización variando la técnica de búsqueda empleada, incluyendo búsquedas en rejilla [12], búsqueda aleatoria [2] , algoritmos evolutivos [3] [16] [19] [20] y bio inspirados [1] [9] [11] entre otros, pero ha sido poco explorada para el ámbito de Big Data. El resto de este trabajo está organizado como se detalla a continuación. En la sección 2 se presentan algunos antecedentes del paradigma de programación MapReduce. En la sección 3 se presenta el algoritmo propuesto para realizar selección de modelo completo en Big Data. En la sección 4 se presentan los experimentos desarrollados y en la sección 5 se presentan las conclusiones.

2. Big Data y el modelo de programación MapReduce

En esta sección se presentan algunos antecedentes sobre el modelo de programación de aplicaciones para el análisis de información en Big Data más extendido conocido como MapReduce. Dicho modelo fue introducido por Dean & Ghemawat en 2004, con el fin de posibilitar la paralelización y distribución del cómputo a gran escala. El diseño del modelo MapReduce fue desarrollado considerando los siguientes principios fundamentales [21]:

- *Hardware básico y de bajo costo.* En lugar de utilizar equipos costosos y de alto desempeño, MapReduce fue diseñado para ejecutarse en clústeres de equipos de cómputo básicos.
- *Clústeres extremadamente escalables RAIN (arreglo redundante de nodos independientes y baratos).* Un nodo MapReduce utiliza su disco duro local. Estos nodos pueden interconectarse mediante conexiones de red estándar y pueden ser puestos fuera de servicio casi sin afectar a algún proceso en ejecución.
- *Tolerantes a fallos.* MapReduce aplica mecanismos sencillos para replicar la información con el fin de mantener los procesos en ejecución en caso de fallo.

En el modelo MapReduce un proceso de cómputo es especificado como una secuencia de etapas *map, shuffle y reduce* que operan en un conjunto de datos $X = \{x_1, x_2, ..., x_n\}$. La etapa *map* aplica una función μ a cada valor x_i para producir un conjunto finito de pares (k, v) clave-valor. Para permitir el cómputo paralelo de una función $\mu(x_i)$, dicha función solo debe depender de x_i. La etapa *shuffle* recolecta todos los pares clave-valor producidos en la etapa map ejecutada anteriormente y produce un conjunto de listas $L_k = (k; v_1, v_2, ..., v_n)$, donde cada lista consiste de todos los valores v_j, de tal manera que $k_j = k$ para una determinada clave k asignada en la etapa *map*. La etapa *reduce* aplica una función ρ, a cada lista $L_k = (k; v_1, v_2, ..., v_n)$ formada en la etapa anterior (*shuffle*) para producir un conjunto de valores $y_1, y_2, ..., y_n$. La función de reducción ρ es definida secuencialmente sobre L_k pero siendo independiente de otras listas L_k, donde $k^{'} \neq k$ [10].

3. Selección de modelo completo mediante algoritmo genético basado en MapReduce.

En esta sección se describe el algoritmo propuesto para selección de modelo. Dicho algoritmo fue implementado en Apache Spark 1.3.0, un marco de desarrollo de aplicaciones basado en el modelo de programación MapReduce. Para el proceso de búsqueda se desarrolló un algoritmo genético. Dicho AG fue desarrollado en base a la variante propuesta por Eshelman [8] conocida como CHC (Crossgenerational elitist selection, Heterogeneous recombination, Cataclysmic mutation). El algoritmo genético CHC es altamente elitista ya que realiza una competencia entre generaciones, esto significa que los descendientes deben competir contra sus padres por su supervivencia, con lo cual se incrementa la probabilidad de los padres de permanecer dentro de la población. Para evitar una

pronta convergencia hacia un óptimo local, el CHC mantiene la diversidad por medio de un operador de cruza que asegura la variedad en la descendencia. Este operador solo permite la cruza entre individuos que cumplan con cierto nivel de diferencia mayor a un umbral, de esta forma es posible asegurar que la descendencia es significativamente diferente de ambos padres, más diversa y por tanto existe mayor exploración. Cada vez que en toda una generación no se realiza una cruza, se decrementa un umbral que al llegar a cero producirá que se genere una población nueva[17]. A diferencia del enfoque del algoritmo CHC en su estado puro donde no se utiliza la mutación en su sentido clásico sino como un recurso para generar nuevas poblaciones, en el algoritmo propuesto para selección de modelo completo, se sigue ocupando el operador de mutación en su sentido tradicional y el operador de macro mutación del enfoque CHC como un recurso para evitar el estancamiento en la convergencia del algoritmo. Cada individuo codificado en el AG representa un modelo potencial, esto significa que para cada individuo es necesario aplicar ciertas transformaciones al conjunto de datos. Es en este punto donde el modelo de programación MapReduce permite hacer frente a las dificultades de efectuar transformaciones en conjuntos de datos de gran magnitud. En el Algoritmo **1** se detalla el algoritmo propuesto en este trabajo denominado Algoritmo Genético para Selección de Modelo Completo Basado en MapReduce en adelante **AGSMCMR**.

```
Data: G← 0
Crear_Población(G);
evalúaMR(G);
while Condición_Terminación do
    descendencia(G) ← Cruza(Población(G));
    if Umbral_Incesto ≤ 0 then
        RestableceUmbral();
        descendencia(G) ← mutCataclismica(descendencia(G));
    else
        evalúaMR(descendencia(G));
        Población(G + 1) ← selecciónElitista(descendencia(G));
    end
end
```

Algorithm 1: AGSMCMR.

Como se mencionó anteriormente, la evaluación de cada modelo potencial involucra múltiples transformaciones en el conjunto de datos, además de la construcción de los clasificadores. A causa del gran volumen de los datos, en este trabajo, la función de evaluación del desempeño fue desarrollada utilizando el modelo de programación MapReduce, denotada en el Algoritmo **1** como ***evaluaMR()***. La función ***evaluaMR()*** está conformada por las sub funciones que llevan a cabo el pre procesamiento, selección de características e hiperparámetros de los algoritmos de cla-

sificación, las cuales al estar basadas en el modelo MapReduce, constan de las etapas Map y Reduce descritas con anterioridad. Como parámetro de desempeño del individuo se evaluó el promedio del área bajo la curva (AUC) de la validación cruzada a 10 pliegos. La validación cruzada es realizada en el conjunto de datos que fue transformado por las funciones de pre procesamiento y selección de características con los valores de los hiperparámetros del algoritmo de clasificación codificados en el individuo. Los algoritmos de clasificación empleados en este trabajo se encuentran en el Cuadro **1**.

Cuadro 1. Métodos de pre procesamiento, algoritmos de selección de características y algoritmos de clasificación utilizados en este trabajo

	Estandarización de características
	Normalización
Métodos de pre procesamiento	Análisis de componentes principales (PCA)
	Escala y desplazamiento
	Discretización
	Información mutua conjunta (JMI)
	Mín. redundancia - máx. relevancia (mrMR)
Algoritmos de selección de características	Interacción bloqueo (ICAP)
	Maxim. de inf. condicional mutua (CMIM)
	Fragmentos informativos (IF)
	Máquina de soporte vectorial (SVM)
	Regresión logística (LR)
Algoritmos de clasificación	Clasificador Bayes ingenuo (NB)
	Árbol de decisión (DT)
	Bosque aleatorio (RF)
	Árboles de impulso gradiente (Boo)

4. Experimentos y resultados

Con el fin de evaluar la factibilidad del algoritmo propuesto para selección de modelo completo, se utilizaron los conjuntos de datos que aparecen en el Cuadro **2**. Dichos conjuntos de datos han sido utilizados en diversos trabajos que abordan problemas de clasificación en Big Data [6] [7] [13] [14]. También se crearon 2 conjuntos de datos sintéticos, los cuales constan de 50,000 instancias y dos clases. El primero fue creado con 5 atributos con poder discriminativo, 25,121 instancias de la clase uno y 24,879 instancias en la clase dos. El segundo consta de 10 atributos, 5 no informativos y 5 con poder discriminativo, 25,108 instancias de la clase uno y 24,892 instancias de la clase 2.

Los parámetros del algoritmo genético fueron determinados experimentalmente al analizar el desempeño de las combinaciones de los fac-

Cuadro 2. Conjuntos de datos utilizados en los experimentos.

Conjunto de datos	Ejemplos	Atributos	Muestras por clase
Covtype_2_vs_1	495,141	54	(283,301 211,840)
Census–income	199,523	40	(187,141 12,382)
Fars_Fatal_Inj_vs No_Inj	50,164	52	(17,445 32,719)
Sintético 1	50,000	5	(25,121 24,879)
Sintético 2	50,000	10	(25,108 24,892)

tores: **Tamaño de Población = 4, 6, 8 Porcentaje de cruza: 0.5, 0.7, 0.9** y **Porcentaje de mutación = 0.1, 0.2, 0.3** en la décima parte del conjunto *Covtype_2_vs_1*. El tratamiento que alcanzó el máximo grado de desempeño fue la combinación *población = 8*, *porcentaje de cruza = 0.5* y *0.1 de porcentaje de mutación*. Se realizaron 10 replicas de cada experimento con cada conjunto de datos, estos fueron divididos en 60 % de las muestras para entrenamiento del algoritmo y 40 % para el conjunto de pruebas. El desempeño del individuo con el mayor grado de aptitud en la última generación del algoritmo genético fue comparado con los algoritmos disponibles en las librerías de aprendizaje automático de **Spark 1.3.0** (mostrados en el Cuadro **1**) con sus hiperparámetros por defecto. En el Cuadro **3** se muestran los resultados obtenidos.

Cuadro 3. Resultados obtenidos en el conjunto de prueba por el mejor individuo del algoritmo propuesto y los obtenidos por los algoritmos de clasificación en Big Data: SVM, Árbol de decisión (DT), Bosque aleatorio (RF), Bayes ingenuo (NB), Regresión logística (LR) y Árboles de impulso gradiente (Boo), tras 10 replicaciones. Los mejores resultados para cada conjunto de datos están en **negritas**.

Conjunto	Propuesto	SVM	DT	RF	NB	LR	Boo
Sintético 1	**0.99**	0.88	0.89	0.91	0.65	0.97	0.95
Sintético 2	**0.99**	0.89	0.89	0.90	0.78	0.96	0.95
Fars...	**0.99**	0.58	0.99	0.95	0.58	0.5	0.99
Census...	**0.89**	0.68	0.60	0.53	0.61	0.60	0.67
Covtype...	**0.89**	0.54	0.76	0.69	0.5	0.69	0.81

Posteriormente se realizó una prueba ANOVA con un 95 % de confianza y una prueba Pos hoc Tukey. En el Cuadro **4** se muestran los resultados.

En el Cuadro **4** y **5** puede verse que el modelo codificado en el mejor individuo del algoritmo propuesto tiene un nivel de desempeño superior al de los algoritmos de la línea base en casi todos los conjuntos de datos analizados. Estos resultados muestran el impacto sobre el desempeño de un proceso de clasificación cuando se realiza una adecuada selección de características y pre procesamiento de datos. Sin embargo, en el conjunto

Cuadro 4. Estadístico F de la prueba ANOVA y valores "q"de la prueba pos hoc Tukey con todas las comparaciones por pares entre el mejor individuo (MI) del algoritmo propuesto y los algoritmos de la línea base. El valor crítico de la prueba ANOVA al nivel de importancia 0.05 es 2.2464 (F(6,63)) para todos los conjuntos de datos. El valor crítico de la prueba Tukey al nivel de importanca de 0.05 es de 4.3069 con 63 grados de libertad para todos los conjuntos de datos. Los casos que exceden el valor crítico son considerados como que poseen una diferencia estadísticamente significativa y son marcados con * (PARTE 1).

Conjunto	ANOVA F	MIvsSVM	MIvsDT	MIvsRF	MIvsNB
Sintético 1	**23,895.9***	**147.8***	**137.7***	**117.6***	**463.3***
Sintético 2	**15,718.6***	**188.5***	**185.2***	**171.5***	**388.7***
Fars...	**35,887.0***	**334.2***	0.1	**34.9***	**337.2***
Census...	**288.4***	**31.1***	**42.5***	**53.0***	**41.4***
Covtype...	**1,111.7***	**83.4***	**31.1***	**47.9***	**93.3***

Cuadro 5. Estadístico F de la prueba ANOVA y valores "q"de la prueba pos hoc Tukey con todas las comparaciones por pares entre el mejor individuo (MI) del algoritmo propuesto y los algoritmos de la línea base. El valor crítico de la prueba ANOVA al nivel de importancia 0.05 es 2.2464 (F(6,63)) para todos los conjuntos de datos. El valor crítico de la prueba Tukey al nivel de importanca de 0.05 es de 4.3069 con 63 grados de libertad para todos los conjuntos de datos. Los casos que exceden el valor crítico son considerados como que poseen una diferencia estadísticamente significativa y son marcados con * (PARTE 2).

Conjunto	ANOVA F	MIvsLR	MIvsBoo
Sintético 1	**23,895.9***	**30.0***	**58.6***
Sintético 2	**15,718.6***	**68.4***	**78.6***
Fars...	**35,887.0***	**407.4***	0.2
Census...	**288.4***	**42.9***	**32.8***
Covtype...	**1,111.7***	**47.2***	**19.8***

de datos *Fars_Fatal_Inj_vs No_Inj*, el desempeño del mejor individuo no consiguió diferencias estadísticamente significativas en comparación a los algoritmos DT y Boo. A pesar de que el desempeño no fuera del todo superior en este conjunto de datos, los resultados dan evidencia de que la selección de modelo en conjuntos de datos de grandes dimensiones es posible mediante el paradigma MapReduce, además de que es posible obtener modelos más eficientes para el ámbito de Big Data.

5. Conclusiones y trabajos futuros

En este trabajo se abordó el problema de selección de modelo completo para conjuntos de datos de grandes dimensiones. Se propuso el desarrollo de un algoritmo de selección de modelo basado en un algoritmo genético, desarrollado bajo el modelo de programación MapReduce. Los resultados experimentales muestran que, es posible realizar un algoritmo de selección de modelo bajo el paradigma antes mencionado. Los modelos obtenidos demuestran una importante mejoría en el poder pre-

dictivo en comparación con los algoritmos de clasificación en Big Data con los parámetros por defecto. Como trabajo futuro se buscará reducir el número de funciones de evaluación necesarias para obtener modelos eficientes y se explorarán medidas para evitar o penalizar el sobre ajuste de los modelos obtenidos con este algoritmo.

Referencias

1. Bansal B., Sahoo A., "Full Model Selection Using Bat Algorithm". Cognitive Computing and Information Processing (CCIP), 2015 International Conference on, 3–4 March 2015 , pp. 1–4, 2015
2. Bergstra, J., & Bengio, Y. (2012). Random search for hyperparameter optimization. The Journal of Machine Learning Research, 13(1), 281–305.
3. Chatelain, C., Adam, S., Lecourtier, Y., Heutte, L., & Paquet, T. (2010). A multi-model selection framework for unknown and/or evolutive misclassification cost problems. Pattern Recognition, 43(3), 815–823.
4. Cox, M., and Ellsworth, D., "Managing Big Data for Scientific Visualization", ACM Siggraph, vol. 97, pp. 1–17, 1997.
5. Dean, J., and S. Ghemawat, (2008), "MapReduce", Commun. ACM, vol. 51, no. 1, pp. 107–113, 2008.
6. del Río, S., López, V., Benítez, J. M., & Herrera, F. (2014). On the use of MapReduce for imbalanced big data using Random Forest. Information Sciences, 285, 112–137.
7. del Río, S., López, V., Benítez, J. M., & Herrera, F. (2015). A MapReduce Approach to Address Big Data Classification Problems Based on the Fusion of Linguistic Fuzzy Rules. International Journal of Computational Intelligence Systems, 8(3), 422–437.
8. Eshelman, L. J. (1991). The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging. Foundations of Genetic Algorithms 1991 (FOGA 1), 1, 265.
9. Escalante, H., J., M. Montes, and L. E. Sucar, (2009),"Particle Swarm Model Selection,"J. Mach. Learn. Res., vol. 10, no. June, pp. 405–440, 2009.
10. Goodrich, M. T., Sitchinava, N., & Zhang, Q. (2011). Sorting, searching, and simulation in the mapreduce framework. In Algorithms and Computation (pp. 374–383). Springer Berlin Heidelberg.
11. Guo ,X., C., J. H. Yang, C. G. Wu, C. Y. Wang, and Y. C. Liang, (2008). "A novel LSSVMs hyperparameter selection based on particle swarm optimization", Neurocomputing, vol. 71, pp. 3211–3215, 2008., vol. 71, pp. 3211–3215
12. Kaneko , H., and K. Funatsu, (2015). "Fast optimization of hyperparameters for support vector regression models with highly predictive ability", Chemom. Intell. Lab. Syst., vol. 142, pp. 64–69.
13. López, V., del Rio, S., Benitez, J. M., & Herrera, F. (2014, July). On the use of MapReduce to build linguistic fuzzy rule based classification systems for big data. In Fuzzy Systems (FUZZIEEE), 2014 IEEE International Conference on (pp. 1905–1912). IEEE.
14. López, V., del Río, S., Benítez, J. M., & Herrera, F. (2015). Costsensitive linguistic fuzzy rule based classification systems under the MapReduce framework for imbalanced big data. Fuzzy Sets and Systems, 258, 5–38.
15. Mukkamala, R. R., Hussain, A., & Vatrapu, R. (2014, September). FuzzySet Based Sentiment Analysis of Big Social Data. In Enterprise Distributed Object Computing Conference (EDOC), 2014 IEEE 18th International (pp. 71–80). IEEE.
16. Oh, S. K., Kim, W. D., Pedrycz, W., & Seo, K. (2014). Fuzzy Radial Basis Function Neural Networks with information granulation and its parallel genetic optimization. Fuzzy Sets and Systems, 237, 96–117.
17. Parmee, I. C. (2012). Evolutionary and adaptive computing in engineering design. Springer Science & Business Media.

18. Robertson, S. P., Vatrapu, R. K., & Medina, R. (2010). Off the wall political discourse: Facebook use in the 2008 US presidential election. Information Polity, 15(1), 11–31.
19. RosalesPérez, A., Gonzalez, J. A., Coello, C. A. C., Escalante, H. J., & ReyesGarcia, C. A. (2014a). Multiobjective model type selection. Neurocomputing, 146, 83–94.
20. RosalesPérez, A., Gonzalez, J. A., Coello, C. A. C., Escalante, H. J., & ReyesGarcia, C. A. (2015). Surrogateassisted multiobjective model selection for support vector machines. Neurocomputing, 150, 163–172.
21. Sakr, S., Liu, A., & Fayoumi, A. G. (2013). The family of MapReduce and largescale data processing systems. ACM Computing Surveys (CSUR), 46(1), 11.
22. Tlili, M., & Hamdani, T. M. (2014, August). Big data clustering validity. In Soft Computing and Pattern Recognition (SoCPaR), 2014 6th International Conference of (pp. 348–352). IEEE.

Aprendizaje de Modelos Difusos para Predicción de Series de Tiempo

Juan J. Flores, Félix Calderón, José Ortiz Carlos Lara

Universidad Autónoma de Tlaxcala, DEP-FIE, Universidad Michoacana
Morelia, Mexico
{juanf,calderon}@umich.mx; job@dep.fie.umich.mx, carlos.lara@cimat.mx
http://dep.fie.umich.mx/

Recibido: 15 de Octubre de 2015, Aceptado: 15 de octubre de 2015,
Versión final: 15 de noviembre de 2015

Resumen El método de Vecinos Cercanos (NN, por sus siglas del inglés Nearest Neighbors) se basa en una idea muy simple: situaciones similares deben tener valores siguientes similares. Se debe buscar a los vecinos cercanos de acuerdo a un criterio de distancia y con ellos determinar el siguiente valor de la serie de tiempo. La primera idea que cruza la mente al estudiar la técnica de Vecinos Cercanos es ponderar la contribución de los vecinos de acuerdo a la distancia con la observación actual. La versión difusa de Vecinos Cercanos pondera de manera implícita la contribución de los diferentes vecinos al momento de realizar la predicción. Esto se hace mediante la función de membresía difusa de los términos lingüísticos, que en éste caso actúan como una forma de distancia a la observación actual. La fase de entrenamiento reúne todos los escenarios diferentes presentes en las observaciones anteriores de las series de tiempo, y lo mapea en forma de reglas difusas. Cuando se encuentra una nueva situación y se requiere predecir la siguiente observación, se procede como en la inferencia difusa normal, la observación actual es fuzzificada, se recorren las reglas para ver cuales se activan (i.e., sus antecedentes se satisfacen) y el resultado de la predicción es defuzzificado por la regla del centro de gravedad. FNN se prueba con un conjunto de series de tiempo no lineales, caóticas y los resultados son comparados con ARIMA y NN. Su desempeño es satisfactorio hasta este momento y nos impulsa a continuar con la evaluación y mejoras del método.

Palabras Clave: Series de Tiempo, Modelos Difusos, Aprendizaje Máquina, Vecinos más Cercanos, Pronóstico.

1. Introducción

Una serie de tiempo se define como un conjunto de observaciones cuantitativas, arregladas en orden cronológico [1], donde generalmente asumimos que el tiempo es una variable discreta. Ejemplos de series de tiempo se encuentran en los campos de ingeniería, ciencia, sociología y economía, entre otros [2]; las series de tiempo se analizan para entender el pasado y predecir el futuro [3]. Esta predicción debe ser tan precisa como sea posible, dado que puede estar ligada a actividades que involucren decisiones de mercado, ventas de productos, índices de la bolsa de valores y demanda eléctrica, entre otras.

En muchas ocasiones es simple y directo predecir el siguiente valor de la serie de tiempo. Sin embargo, mientras más a futuro se requiera predecir, se tiene mayor incertidumbre y los errores de predicción se hacen mayores. Algunos modelos estadísticos, como los autorregresivos [4] pueden ser usados para modelar y predecir series de tiempo. Estas técnicas tradicionales de predicción se basan en modelos lineales, sin embargo, muchas de las series de tiempo que encontramos en la práctica, pueden exhibir características no presentes en procesos lineales. Si sabemos que la serie de tiempo es no lineal, ésta puede exhibir características dinámicas muy ricas, incluyendo sensibilidad a condiciones iniciales, conocida como comportamiento caótico [5]. En el pasado se han propuesto varios métodos para predecir series de tiempo caóticas. Uno de los primero métodos propuestos fue el uso de modelos no lineales (Nonlinear Models - NLM) [6], sin embargo, los principales problemas de estos métodos era la selección adecuada de modelos y la dependencia de datos. Otro método que ha sido utilizado para resolver el problema de predicción en series de tiempo caóticas son las Redes Neuronales Artificiales (Artificial Neural Networks - ANNs) [7]. Las ANNs se entrenan para aprender la relación entre las variables de entrada y las de salida, en base a patrones históricos. Sin embargo, la desventaja principal de ANNs es el proceso de aprendizaje requerido para formar el modelo.

Más recientemente, se han aplicado de manera muy exitosa técnicas de clasificación basadas en los vecinos más cercanos. Los Vecinos más Cercanos (Nearest-Neighbors - NN) [8] es un algoritmo que puede ser empleado para pronosticar series de tiempo; es muy intuitivo y simple. NN busca escenarios similares, obteniéndolos de espacios de fase o características altamente dimensionales, en muchos casos con datos incompletos [6]. El algoritmo NN asume que las subsecuencias de la serie de tiempo que se han observado en el pasado muy probablemente se

parezcan a subsecuencias futuras y puedan ser usadas para generar las predicciones NN.

Una desventaja del método NN es la sensibilidad a cambios en los parámetros de entrada (i.e., la distancia máxima para considerar dos subsecuencias como vecinas y la dimensión de embebido). Si los parámetros de entrada no se seleccionan apropiadamente, la precisión de las predicciones puede reducirse considerablemente.

Una manera de seleccionar los parámetros de entrada óptimos para el método NN es usando un algoritmo de la familia de Computación Evolutiva (Evolutionary Computation - EC) [9]. EC es una tecnología computacional formada de una colección de paradigmas aleatorios de búsqueda global para encontrar soluciones óptimas a un problema dado. En partícular, Evolución Diferencial (Differential Evolution - DE) es una estrategia de búsqueda poblacional que ha probado ser un método valioso de optimización para problemas en dominios reales con funciones objetivo multimodales [10,11]. DE es un método de búsqueda directa que presenta buenas propiedades de convergencia y simplicidad en su implementación. El diseño de modelos para la predicción de series de tiempo se ha hecho tradicionalmente usando modelos estadísticos. En el área de modelado de series de tiempo, encontramos ARIMA (Auto-Regressive Integrated Moving Average), ARMA y AR, entre otros [12]. Estos modelos se definen en términos de observaciones pasadas y corrección de errores. Las técnicas estadísticas como la auto-correlación y auto-correlación parcial, han ayudado a los científicos a identificar cuales de las observaciones pasadas y/o errores son significativos en la construcción de modelos de predicción. En la última década, se han usado las ANN de manera exitosa para modelar y predecir series de tiempo. Diseñar una ANN que proporcione una buena aproximación es un problema de optimización. Dados los numerosos parámetros a seleccionar dentro de su diseño, el espacio de búsqueda en ésta tarea de optimización es enorme. Por otro lado, los algoritmos de aprendizaje usados para entrenar ANN solo son capaces de determinar los pesos de las conexiones sinápticas y no resuelven problemas de diseño arquitectónico de la red. De modo que un científico que requiera un modelo neuronal tiene que diseñar la red por prueba y error. Cuando se diseña una ANN a mano, los científicos solo pueden probar unas cuantas variantes, seleccionando la mejor del conjunto de las que probaron.

La primera idea que cruza la mente cuando vemos el método de vecinos cercanos para la predicción de series de tiempo es pesar la contribución de los diferentes vecinos, de acuerdo a la distancia a la observación actual. En este artículo proponemos la versión difusa del método de vecinos cercanos para la predicción de series de tiempo. Este modelo

implícitamente pondera las contribuciones de los diferentes vecinos en la predicción, usando la membresía difusa de los términos lingüisticos como un tipo de distancia a la observación actual. La fase de entrenamiento compila todos los diferentes escenarios de lo que ha sido observado en el pasado de la serie de tiempo, en un conjunto de reglas difusas. Cuando encontramos una nueva situación y requerimos predecir el valor futuro, al igual que en cualquier sistema de inferencia difusa, la observación actual se fuzzifica, el conjunto de reglas se recorren, verificando cuales de ellas se activan (i.e., sus antecedentes se satisfacen) y el valor a predecir se desfuzzifica por la regla del centro de gravedad.

Una versión preliminar de esta idea fue publicada por Flores et al. en el congreso SIGEF 2015 [13], donde los algoritmos fueron presentados, mientras su implementación estaba en progreso. Una vez implementados y con pruebas posteriores, se publicaron en el congreso ROPEC 2015 [14]. Los autores de este trabajo continuarán refinando los algoritmos y su implementación, probándolos contra otras técnicas de predicción del estado del arte.

El resto del artículo está organizado como sigue. La Sección 2 describe el método de predicción de series de tiempo de Vecinos Cercanos. La Sección 3 expone los conceptos básicos necesarios para producir razonamiento difuso. La Sección 4 propone el método de predicción de series de tiempo que llamamos Vecinos Cercanos Difusos (FNN del inglés Fuzzy Nearest Neighbor). La Sección 5 presenta un caso de estudio donde se ilustra el uso del método propuesto. La Sección 6 presenta los resultados y los compara con los obtenidos usando el método ARIMA. Finalmente, la Sección 7 concluye el trabajo.

2. Método de Pronóstico de Vecinos Cercanos

El método de aproximación NN (Vecinos más Cercanos) es muy simple, pero poderoso. Se ha usado en muchas aplicaciones, particularmente en tareas de clasificación [15]. Una variación del método NN es no determinar el número de vecinos deseados, sino recuperar todos los que estén en su vecindad, a una distancia de la observación actual. Para resolver el problema de pronóstico, el método NN (Vecinos Cercanos) usa el promedio de las predicciones de los vecinos de la observación actual, sin hacer ninguna suposición acerca de la distribución de las variables a predecir en el proceso de aprendizaje.

La idea detrás del pronóstico NN es que ejemplos de entrenamiento similares muy probablemente tendrán futuros similares. Se debe buscar a todos los vecinos cercanos a la observación actual, de acuerdo a

un criterio de distancia. La distancia que comúnmente se usa es la distancia Euclideana. Sin embargo, el método también permite el uso de otras distancias, como las de Chevichev, Manhattan y Mahalanobis, entre otras [16]. Una vez que hemos encontrado los vecinos más cercanos, se calcula una estimación de la variable de salida (el pronóstico) usando el promedio de las salidas de los vecinos más cercanos.

En contraste con los métodos estadísticos, que tratan de identificar un modelo a partir de los datos, el método NN usa el conjunto de entrenamiento como el modelo [17]. La ventaja principal del método NN es su efectividad en situaciones donde el conjunto de entrenamiento es grande y contiene estructuras deterministas. El resto de esta sección describe el método de predicción NN.

Sea $\mathbb{X} = (x_t)_{t=1}^{N}$ una serie de tiempo, donde x_t es el valor registrado de la variable x en el instante de tiempo t. El problema de pronóstico busca estimar Δn valores futuros consecutivos, i.e. $(x_f)_{f=N+1}^{N+\Delta n}$, usando cualesquiera de las observaciones disponibles en $\mathbb{X}$ [18].

Definition 1 (vectores de retardo)**.** *Dado un punto de referencia r, su correspondiente vector de retardo $X_r \in R^m$ es una secuencia de la forma*

$$X_r = \left(x_{r-(m-k)\tau}\right)_{k=1}^{m}, \tag{1}$$

donde $\tau > 0$ es el tiempo de retardo, $m > 0$ es la dimensión de embebido y $(m-1)\tau < r \leq N$.

El valor pronosticado $\hat{x}_{N+h}$ puede ser estimado a partir de

$$Q = \{\, i \mid (m-1)\tau < i \leq N-h, ||X_i - X_N|| < \epsilon \,\}, \tag{2}$$

donde ϵ es un umbral para decidir cuando dos vectores de retardo son lo suficientemente cercanos para ser considerados vecinos. Q representa el conjunto de situaciones, i.e., vectores de retardo X_i, que no se encuentran mas lejos que ϵ de la situación actual X_N. Finalmente, $\hat{x}_{N+h}$ puede ser calculado como se muestra en la Ecuación (3).

$$\hat{x}_{N+h} = \frac{1}{||Q||} \sum_{i \in Q} x_{i+h} \tag{3}$$

La Ecuación (3) proporciona el mecanismo del método de pronóstico NN. Los parámetros involucrados en los modelos de pronóstico son τ, m, y ϵ. τ, el parámetro de retardo de tiempo, especifica un periodo de submuestreo. Para entender el significado de τ imagine que debe pronosticar el nivel de agua de una presa. Suponga que la serie de tiempo está formada por mediciones tomadas cada minuto; las lecturas (i.e., componentes

de la serie de tiempo) permanecen básicamente constantes a través del tiempo. Se requiere un submuestreo cada τ mediciones, de manera que estas exhiban un cambio substancial de una a otra. m, el tamaño del vector de retardo especifica cuanta información del pasado se requiere para producir pronósticos precisos, manteniendo la complejidad del proceso de pronóstico en niveles aceptables. De esta manera, τ y m, definen los vectores de retardo, los cuales identifican la estructura del sistema y en algún momento nos permiten reconstruir diagramas de fase del sistema dinámico subyacente al proceso que produjo la serie de tiempo. Un vector de retardo juega el papel de un vector de estado. NN pronostica el siguiente valor identificando estados pasados similares al estado actual. ϵ especifica que tan cercanos deben estar dos estados para ser considerados vecinos.

FNN supone que τ, m, y ϵ ya fueron determinados para usarse en NN. τ y m pueden ser determinados usando los algoritmos propuestos por Kantz y Schreiber [19] o por medio de Computación Evolutiva, como lo proponen De La Vega et al. [20]. Kantz y Schreiber no proporcionan un algoritmo para determinar ϵ. De La Vega et al. proponen la inclusión de ϵ como parte de la codificación en el proceso evolutivo, para determinar los tres parámetros simultáneamente.

3. Lógica Difusa

Esta sección le proporciona al lector una introducción breve al tema de lógica difusa. Aquí se presentan los conceptos de conjuntos difusos y sus operaciones. Estos conceptos serán usados tanto en la compilación de la base de reglas, como en la inferencia para la predicción de series de tiempo.

3.1. Conjuntos Clásicos y Difusos

Un conjunto clásico A se define como una colección de elementos u objetos. Cualquier elemento u objeto x o pertenece o no a A. La función de membresía $\mu_A(x)$ de x en A es un mapeo $\mu_A : X \mapsto \{\, 0, 1 \,\}$. Del mapeo de membresía concluimos que $A \cap \neg A = \{\ \}$.

La lógica difusa es una lógica basada en conjuntos difusos, i.e., conjuntos de elementos u objetos caracterizados por valores en el intervalo $[0, 1]$, en lugar del conjunto $0, 1$, como en la teoría de conjuntos convencional. La función de membresía de un conjunto asigna un número en el intervalo $[0, 1]$ a cada elemento del universo de discurso de un conjunto difuso. El conjunto difuso A es caracterizado por su función de

membresía $\mu_A : X \mapsto [0,1]$; se define como un conjunto de parejas: $A = \{ (x, \mu_A(x)) \}$.

Aunque las funciones de membresía de un conjunto difuso pueden tomar varias formas, en este trabajo usamos la forma triangular, la cual es la mas común. La función de membresía triangular $\text{Tri}(x, \alpha, \beta, \gamma)$ ($\alpha < \beta < \gamma$) tienen valores positivos en $[\alpha, \gamma]$ con un máximo en $x = \beta$ [21,22].

3.2. Operaciones Difusas

Las operaciones de conjuntos difusos utilizadas en este trabajo son Fuzzificación, And, Implicación y Defuzificación. Dado que hay muchas definiciones para estas operaciones difusas, las definiremos de manera precisa en esta sección, para establecer un marco de referencia común con el lector, a partir del cual se pueda establecer el método de predicción de Vecinos Cercanos Difusos.

Definition 2 (fuzzificación)**.** *Sea $(A_i)_{i=1}^N$ una secuencia de conjuntos difusos y $x \in X$ un valor dado; la **fuzzificación** de x se define como* $\text{fuzz}(x) = (\mu_{A_i}(x))_{i=1}^N$.

Definition 3 (intersección difusa)**.** *Sean A y B dos conjuntos difusos definidos en X con funciones de membresía μ_A y μ_B, la **intersección** $A \cap B$ se define por la función de membresía $\mu_{A \cap B} = \min(\mu_A(x), \mu_B(x))$.*

Definition 4 (reglas difusas e inferencia)**.** *El modelo difuso determinado por FNN contiene un conjunto de reglas de inferencia difusas. La regla j tiene la forma dada por la Ecuación (4).*

$$\begin{aligned} \textit{If } (X_n \textit{ is } A_{j,0}) \wedge (X_{n-1} \textit{ is } A_{j,1}) \wedge \cdots \wedge (X_{n-m} \textit{ is } A_{j,m}) \\ \textit{then } (X_{n+1} \textit{ is } A_{j,m+1}) \end{aligned} \tag{4}$$

donde X_t es el valor de la serie de tiempo $t \in \{ 1, \ldots, n+1 \}$ y $A_{j,k}$ son los términos lingüísiticos a los cuales deben pertenecer para satisfacer los antecedentes de la regla. $A_{j,m+1}$ es el consecuente o resultado de la inferencia. La semántica de la regla establece que si los valores observados previos pertenecen a los conjuntos requeridos por la regla (en cuyo caso la regla se activa), podemos decir que el valor pronosticado de la serie de tiempo (i.e., X_{n+1}) pertenecerá al conjunto $A_{j,m+1}$ con un valor de membresía igual a la magnitud en que los antecedentes son satisfechos (fuerza de activación). A este valor le llamaremos a_j.

Existirán situaciones donde más de una regla se active. En esas situaciones, necesitamos defuzzificar los resultados difusos para proporcionar un solo valor real para el pronóstico. El método del centro de gravedad

es el método más común e intuitivo. Si n reglas se activan con fuerzas a_j, y el centro de la función de membresía triangular $A_{j,m+1}$ es β_j, el valor pronosticado está dado por la ecuación (5).

$$X_{n+1} = \frac{\sum_j^n a_j \beta_j}{\sum_j^n a_j} \tag{5}$$

Note que el utilizar conjuntos difusos triangulares, implica que sus centros de gravedad coinciden con el centro del triángulo.

4. Modelo Difuso de Vecinos Cercanos

Esta sección presenta el método de predicción de series de tiempo propuesto en este artículo. FNN es la versión difusa del método de predicción de vecinos cercanos (NN). NN no requiere una fase de entrenamiento (ver la Sección 2); cuando las últimas observaciones de la serie de tiempo están disponibles, el método las compara con la historia del sistema, buscando patrones similares, lo cuales hayan ocurrido en el pasado. Estos patrones similares se usan para predecir la magnitud de las siguientes observaciones. A diferencia de NN, FNN sí requiere una fase de entrenamiento; en esa fase, se aprenden un conjunto de reglas difusas, i.e., se produce un modelo difuso. Una vez que el modelo de reglas difusas se ha producido, se usa cada vez que se requiere una predicción.

4.1. Términos Lingüísticos Difusos y Reglas Difusas

La idea principal de FNN es determinar un conjunto de reglas difusas que nos permitan predecir el comportamiento de una serie de tiempo. Una regla difusa tiene la forma dada por la Ec. (4). De cada estado (vector de retardo) en el pasado, con su observación siguiente, podemos derivar una regla. Una regla de la forma dada por la Ec. (4), derivada de un vector de retardo, toma la forma de la Ec. (6). Adicionalmente, podemos asumir que el intervalo de muestreo no necesariamente es 1; y se define del siguiente modo:

$$\begin{aligned} \text{If } (X_n \text{ is } A_0) \wedge (X_{n-\tau} \text{ is } A_1) \wedge \cdots \wedge (X_{n-m\tau} \text{ is } A_m) \\ \text{then } (X_{n+1} \text{ is } A_{m+1}) \end{aligned} \tag{6}$$

donde los A_i son los términos lingüísticos difusos (FLT, del inglés Fuzzy Linguistic Terms). Por ahora suponemos que conocemos m y τ. Estos parámetros pueden ser determinados como lo indican Kantz y Schreiber [19], o por medio de un optimizador metaheurístico, como lo describen De la Vega et al. [20].

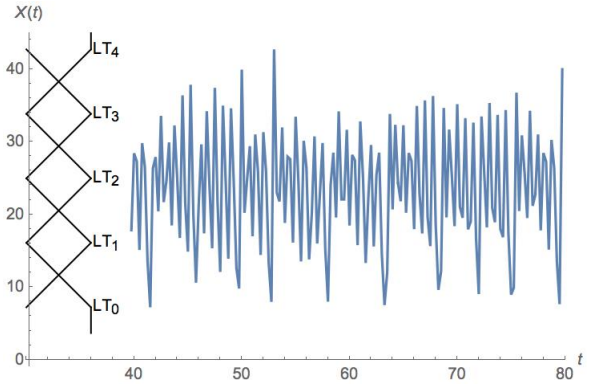


Figura 1. Series de tiempo y sus correspondientes términos lingüisticos.

Para obtener los TLD, dividimos el rango de la serie de tiempo en 3, 5, u 11 valores, para formar las escalas difusas correspondientes a los términos lingüísticos difusos o conjuntos difusos. Mientras mas términos lingüísticos usemos, será mas precisa la predicción. Al mismo tiempo, el proceso es mas costoso en tiempo cuando se incrementa el número de escalas. La Figura 1 muestra una serie de tiempo y sus correspondientes términos lingüísticos difusos. Note que los TLD se traslapan de manera que cualquier valor real pertenece a exactamente dos conjuntos (excepto en los extremos) y su función de membresía nunca es cero.

Una regla difusa, como la de la Ecuación (6), captura el comportamiento de la serie de tiempo en una ventana de tiempo dada. La Figura 2 ilustra una porción de una serie de tiempo que contribuye a la formación de la regla mostrada en la Ec. (7).

$$\begin{aligned} \text{If } (X_n \text{ is } \mathrm{LT}_1) \wedge (X_{n-1} \text{ is } \mathrm{LT}_1) \wedge (X_{n-2} \text{ is } \mathrm{LT}_4) \\ \wedge (X_{n-3} \text{ is } \mathrm{LT}_2) \text{ then } (X_{n+1} \text{ is } \mathrm{LT}_0) \end{aligned} \tag{7}$$

La forma de esta regla se basa en dos parámetros, el tamaño de ventana m y el intervalo de muestreo τ. El tamaño de ventana indica cuantos elementos de la serie de tiempo son considerados para predecir el valor siguiente y τ indica que tan alejados deben estar esos elementos de la serie de tiempo. ($m = 4$ y $\tau = 1$ para la regla en el ejemplo de la Ec. (7)).

4.2. Aprendizaje del Modelo

Esta sección propone un algoritmo para aprender el conjunto de reglas difusas a partir de la serie de tiempo. El Algoritmo 1 muestra pseudocódigo del procedimiento para aprender el conjunto de reglas. LearnRules toma como argumentos X Ð la serie de tiempo, m Ð el tamaño de ventana

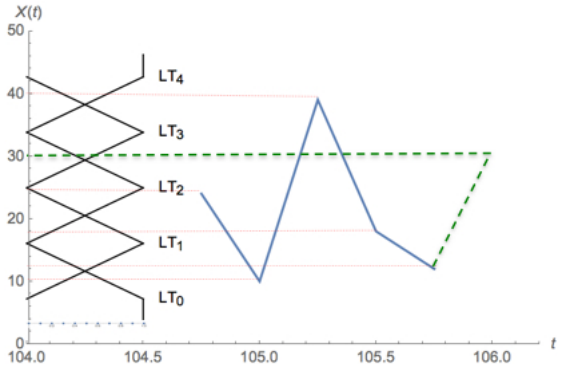


Figura 2. Parte de las series de tiempo y sus correspondientes valores difusos.

y τ Ð el intervalo de muestreo, y regresa el conjunto de reglas derivadas de la serie de tiempo.

Algorithm 1 LearnRules (X, m, τ)

1: Rules $\leftarrow$ { }
2: LTS $\leftarrow$ { }
3: **for each** $W_j = (X_{i+1}, X_i, X_{i-\tau}, X_{i-2\tau}, \ldots, X_{i-(m-1)\tau})$ in X **do**
4: $\quad$ LT$_j \leftarrow$ { }
5: $\quad$ **for each** w in W_j **do**
6: $\quad\quad$ LT$_j \leftarrow$ LT$_j \cup$ Fuzzify(w)
7: $\quad$ **end for**
8: $\quad$ LTS $\leftarrow$ LTS + {LT$_{j0}\times$ LT$_{j1} \times \cdots \times$ LT$_{jm}$}
9: **end for**
10: **for each** LT in LTS **do**
11: $\quad$ rule $\leftarrow$ LT$_1 \wedge \cdots \wedge$ LT$_{m-1} \wedge$ LT$_m \rightarrow$ LT$_0$
12: $\quad$ Rules $\leftarrow$ Rules $\cup$ rule
13: **end for**
14: Agrupar cada regla en Reglas, de tal forma que sus antecedentes y consecuentes coincidan.
15: Crear una nueva regla con los antecedentes/consecuente que coincidan y cuya fuerza de activación sea el promedio de todas las reglas que coincidan.
16: Remplazar las reglas coincidentes con la regla generada.

Para producir el conjunto de reglas difusas (FR del inglés Fuzzy Rules) recorremos la serie de tiempo X, reconociendo los patrones encontrados en la misma y verificando donde se haya encontrado patrones similares. Esos patrones encontrados se recolectan para formar las reglas. En el Algoritmo 1, las líneas 1 y 2 inicializan el conjunto de reglas y un conjunto temporal LTS, usado para almacenar todos los términos lingüísticos posibles para la serie de tiempo. El ciclo que comienza en la línea 3 recorre la serie de tiempo X, generando todas las posibles ven-

tanas (W_j) de tamaño m con retardo τ. En la línea 5 se inicializa LT_j para la j-ésima ventana. El ciclo de la línea 6 calcula los términos lingüísticos para cada término w en W_j (la manera en que distribuimos los conjuntos difusos dentro del universo de discurso garantiza que cada w en W pertenece a no más de dos conjuntos difusos, i. e. Fuzzify(w) regresa una pareja de cardinalidad 2). Una vez que se calculan los conjuntos LT, en la línea 8 todas las posibles combinaciones para una regla se anexan a LTS. Por ejemplo, si $m = 2$, LT_j contendrá el conjunto $\{\{lt_{1,1}, lt_{1,2}\}, \{lt_{2,1}, lt_{2,2}\}, \{lt_{c,1}, lt_{c,2}\}\}$, donde la última tupla contiene los dos conjuntos difusos posibles para el valor X_{i+1} (i. e., el valor a predecir, el consecuente). El valor añadido a LTS en la línea 8 está dado por el producto cartesiano de todos los elementos en LT_j; para el ejemplo, estos conjuntos son $S = \{\{lt_{1,1}, lt_{2,1}, lt_{c,1}\}, \{lt_{1,1}, lt_{2,1}, lt_{c,2}\}, \ldots, \{lt_{1,2}, lt_{2,2}, lt_{c,2}\}$. El ciclo en la línea 10 construye una regla para cada tupla en LTS; por ejemplo, para el primer elemento en S se crea la regla $lt_{1,1} \wedge lt_{2,1} \rightarrow lt_{c,1}$. Finalmente, todas las reglas con los mismos antecedentes/consecuente se colapsan en una sola regla promediada. El registro de reglas idénticas se puede realizar mediante un hash que registre como llave una lista de antecedentes consecuentes. Las agrupaciones mencionadas en la línea 14 corresponden a las entradas del hash. La línea 15 se implementa recorriendo el hash y promediando las colisiones de cada llave. Se sustituyen todas las colisiones por la regla resultante en la línea 16.

4.3. Pronóstico Difuso

Dado un conjunto de FR, el pronóstico a 1 paso (i.e., pronosticar solamente el siguiente valor de la serie de tiempo) se reduce al uso del conjunto de reglas producidas en la fase de aprendizaje. Para el conjunto de FR y los m valores observados en la historia de la ST, mas de una regla puede activarse. Como en cualquier sistema de inferencia difusa (FIS), el resultado (i.e., el pronóstico) se defuzzifica usando el método del centro de gravedad. El Algoritmo 2 muestra el pseudocódigo para producir el pronóstico de un valor a partir de las últimas observaciones y el conjunto de reglas difusas.

El procedimiento Fuzzy Forecasting toma como argumentos el conjunto de reglas difusas Ð FR, la serie de tiempo Ð X, hasta el punto donde se requiere producir el pronóstico, m, el tamaño de ventana. La línea 1 inicializa el conjunto de reglas activadas y determina el tamaño de la serie de tiempo. El ciclo en la línea 2 recorre el conjunto de reglas difusas. La línea 3 verifica que la última observación de la serie de tiempo satisfaga la i-ésima regla difusa. Si es el caso, se registra el consecuente de la regla y la fuerza de activación (línea 4). Cuando todas las reglas se

han visitado, las que fueron activadas se defuzzifican usando el método del centro de gravedad (línea 5). El valor pronosticado se regresa en la línea 6.

Algorithm 2 FuzzyForecasting (FR, X, m)

```
1: Fired ← { }
2: for i = m to size(FR) do
3:     μ_i ← Satisfies (X, FR_i)
4:     if μ_i > 0 then
5:         Fired ← Fired ∪ Consequent(FR_i, μ_i)
6:     end if
7: end for
8: return COG(Fired)
```

5. Ejemplo

Como ejemplo ilustrativo, suponemos que en una serie de tiempo dada, se presenta la situación mostrada en la Figura 2. Se produce una regla que refleja ese escenario. El primer escenario produce la regla dada en la Ec. (6); las funciones de membresía de la variable X en los diferentes instantes de tiempo son: $\mu_{\mathrm{LT}_0}(X45{,}75) = 0{,}68$, $\mu_{\mathrm{LT}_2}(X_{45,5}) = 0{,}6$, $\mu_{\mathrm{LT}_4}(X_{45,25}) = 0{,}71$, $\mu_{\mathrm{LT}_1}(X_{45,0}) = 0{,}90$, $\mu_{\mathrm{LT}_1}(X_{44,75}) = 0{,}53$. La membresía de la regla es $\min(0{,}68, 0{,}6, 0{,}71, 0{,}9, 0{,}53)$. El algoritmo añade el consecuente $(\mathrm{LT}_2, 0{,}53)$.

Al recorrer la serie de tiempo, en un tiempo posterior encontramos el escenario mostrado por la Figura 2. Las funciones de membresía de la variable X en los diferentes instantes de tiempo son: $\mu_{\mathrm{LT}_0}(X_{105,75}) = 0{,}45, \mu_{\mathrm{LT}_2}(X_{105,5}) = 0{,}15, \mu_{\mathrm{LT}_4}(X_{105,25}) = 0{,}55, \mu_{\mathrm{LT}_1}(X_{105,0}) = 0{,}25, \mu_{\mathrm{LT}_1}(X_{104,75}) = 0{,}09$, la membresía de la regla es $\min(0{,}68, 0{,}6, 0{,}71, 0{,}9, 0{,}53)$. El algoritmo añade el consecuente $(\mathrm{LT}_3, 0{,}51)$.

Los puntos medios y las fuerzas de activación de los patrones originales se usan para calcular una media ponderada (Eq. (8)), y el resultado es fuzzyficado nuevamente.

$$X_{n+1} = \frac{24{,}94 * 0{,}53 + 33{,}81 * 0{,}51}{0{,}53 + 0{,}51} = 29{,}28 \qquad (8)$$

$$X_{n+1} \text{ is } \mathrm{LT}_3$$

El modelo difuso de esta serie de tiempo es el conjunto de todas las reglas difusas colectadas por el algoritmo. Supongamos que las magnitudes de la variable X en las lecturas mas recientes son: (22, 12.5, 41, 20,

11). Las funciones de membresía de la variable X en los diferentes instantes de tiempo son: $\mu_{\mathrm{LT}_0}(X_n) = 0{,}49, \mu_{\mathrm{LT}_2}(X_{n-1}) = 0{,}49, \mu_{\mathrm{LT}_4}(X_{n-2}) = 1{,}0, \mu_{\mathrm{LT}_1}(X_{n-3}) = 0{,}25, \mu_{\mathrm{LT}_1}(X_{n-4}) = 0{,}31$. Estos valores activan la regla (posiblemente entre otras reglas) con una fuerza de activación de $\min(0{,}49, 0{,}49, 1{,}0, 0{,}25, 0{,}31) = 0{,}25$. Esto indica que el resultado final del pronóstico es $33{,}81 * 0{,}25/0{,}25. = 33{,}81$. Si se hubiesen activado mas reglas, habrían más términos en el método del centro de gravedad.

6. Resultados

Para probar la eficacia y precisión de FNN aplicamos el algoritmo a varias series de tiempo no lineales caóticas. Estas series de tiempo son Henon, Lorenz, Mackey, y Rossler. Las series de tiempo fueron producidas mediante simulación de sus respectivos modelos dinámicos, produciendo 50,000 valores para cada una.

Los resultados producidos por FNN fueron comparados contra los obtenidos mediante modelos ARIMA y el método de Vecinos Cercanos (NN). Los coeficientes ARIMA y los parámetros para NN y FNN se muestran en la Tabla 1.

Cuadro 1. Coeficientes ARIMA y Parámetros usados en NN y FNN

Series de Tiempo	Longitud	Coeficientes ARIMA (p, d, q)	Parámetros FNN y NN (m, τ)
Henon	50,000	(4, 0, 5)	(2, 14)
Lorenz	50,000	(0, 0, 0)	(3, 23)
Mackey	50,000	(0, 0, 0)	(4, 60)
Rossler	50,000	(0, 0, 0)	(4, 16)

Para cada una de las series de tiempo, los últimos 50 puntos fueron usados como conjunto de prueba, para pronóstico. La Figura 3 muestra la parte de validación de la serie de tiempo de Henon (azul) y los pronósticos producidos por FNN (verde).

Los errores de pronóstico fueron medidos usando MAPE y MSE. Los resultados fueron comparados contra la versión tradicional de Vecinos Cercanos y ARIMA. La Tabla 2 muestra la comparación entre estos métodos. Para cada serie de tiempo, las primeras tres columnas muestran el MAPE para los diferentes modelos. Las siguientes tres columnas muestran el MSE para los mismos modelos. Las medidas de error para el método que obtuvo el error mas pequeño (i.e., alcanzó la mejor precisión de pronóstico) se muestran en negritas.

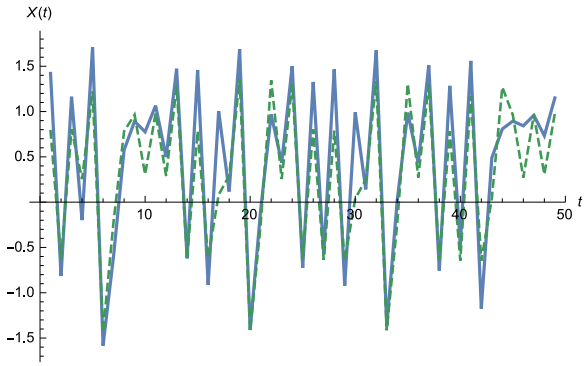


Figura 3. Serie de Tiempo de Henon (azul) y el Pronóstico producido por FNN (verde)

Cuadro 2. Resultados MAPE y MSE Results para FNN, NN, y ARIMA

	MAPE			MSE		
Series de Tiempo	FNN	NN	Arima	FNN	NN	ARIMA
Henon	**65.91**	113.99	169.68	**0.14**	1.13	0.55
Lorenz	41.64	65.61	**4.90**	1.63	**0.11**	0.13
Mackey	**0.75**	3.11	2.08	**0.000079**	0.0006	24.39
Rossler	11.78	**8.17**	103.52	0.36	0.011	**0.0004**

La Tabla 2 muestra que FNN supera a los otros dos métodos para las series de Henon y Mackey-Glass. Este artículo presenta resultados preliminares. FNN está aún en desarrollo y debe ser probado con un conjunto mas grande de series de tiempo para poder estableces diferencias estadísticamente significativas en el comportamiento de los métodos comparados.

7. Conclusiones

Este artículo presenta el métoddo FNN (Fuzzy Narest Neighbors), un método difuso para pronosticar series de tiempo. Este método representa la versión difusa del método de NN (Nearest Neighbors) para pronóstico de series de tiempo. La idea principal del método de NN es verificar qué escenarios en el pasado son suficientemente cercanos al escenario actual y promediar sus salidas (siguiente medición), suponiendo que escenarios similares en la historia deben haber producido salidas similares. Esta versión difusa compila un conjunto de reglas difusas de la serie de tiempo. Esas reglas capturan los diferentes escenarios observados en la serie de tiempo y los convierte en reglas difusas. Cuando se requiere un pronóstico, presentamos las observaciones más recientes al conjunto de reglas y tomamos en cuenta los resultados de aquellas que fueron activadas. El resultado final es el promedio ponderado, tomando en cuenta la fuerza

de activación de cada regla; esta fuerza de activación de alguna manera mide la similaridad entre la situación actual y la representada por la regla. Las reglas son extraídas de la serie de tiempo siguiendo el mismo criterio de similaridad. La similaridad en este contexto es proporcionada por las funciones de membresía de los términos lingüísticos.

Una diferencia entre NN y FNN es que NN no requiere entrenamiento, pero cuando se calcula un pronóstico, NN tiene que buscar todos las ventanas similares en la serie de tiempo. En contraste, FNN requiere entrenamiento para aprender el conjunto de reglas difusas. Sin embargo, cuando se requiere un pronóstico, solamente tenemos que recorrer el conjunto de reglas, cuyo tamaño es mucho menor en número que las observaciones en la serie de tiempo. Se puede demostrar fácilmente que la complejidad de ambos, el proceso de búsqueda para NN y el entrenamiento en FNN son Cuadráticos.

Otra diferencia importante es que NN requiere de un umbral para discriminar entre las ventanas que son consideradas cercanas a la situación actual (vecinas) y aquellas que no lo son. Ese umbral está representado por el parámetro ϵ. En contraste, FNN determina la cercanía de dos escenarios usando las funciones de membresía de los conjuntos difusos. Este hecho nos permite prescindir del parámetro ϵ.

Una similaridad entre NN y FNN es que aprovechan las series de tiempo largas, donde el sistema exhibe una gran variedad de escenarios. Esta situación puede llegar a ser prohibitiva para otros métodos (e.g., ARIMA, ANN, etc.), donde por lo general los recursos computacionales no son suficientes para derivar el modelo apropiado. En estos días, donde los dispositivos de colección de datos son ubicuos y los dispositivos de almacenamiento han crecido tanto en capacidad, necesitamos estar preparados para minar información de repositorios grandes. Esta idea se conoce como Big Data (grandes datos).

Los modelos producidos por FNN son nolineales, implementados como el parcheo típico de las reglas difusas en un sistema de control. La fortaleza del modelo es su habilidad para reconocer el escenario presente como algo ocurrido en el pasado y usar las experiencias pasadas, expresada en la serie de tiempo, para predecir lo que sucederá en seguida. Reconociendo estas fortalezas, FNN ha sido probado con algunas series de tiempo no lineales y caróticas, producidas mediante simulación. Estas series de tiempo se conocen como Henon, Lorenz, Mackey-Glass, y Rossler, por los nombres de sus inventores.

La calidad de los pronósticos producidos por FNN fue medida usando MAPE, un estadístico que mide el error como un porcentaje de la magnitud de la señal. Esta propiedad nos permite comparar el error pro-

ducido en los pronósticos en series de tiempo de diferentes magnitudes. Los pronósticos producidos con FNN fueron comparados con los pronósticos producidos por NN y ARIMA. FNN produjo mejores resultados que ARIMA en 3 de los 4 experimentos. Aún se requieren más experimentos para poder aseverar propiedades estadísticas del algoritmo propuesto.

De acuerdo al conocimiento de los autores, al momento de escribir este artículo no se han detectado otro trabajos que aprendan un conjunto de reglas difusas a partir de una serie de tiempo y las usen para producir pronósticos. Los artículos que usan conceptos de conjuntos y lógica difusa, se refieren al manejo de la incertidumbre en la adquisición de la serie de tiempo. Es decir, modelan el ruido producido en el proceso de adquisición como cierto tipo de incertidumbre.

Reconocimientos

Este trabajo de investigación fue parcialmente patrocinado por el Consejo Nacional de Ciencia y Tecnología (CONACyT), por medio del proyecto "Optimization of industrial processes based on simulators, interfaces and software assurance" (CATEDRAS-3163).

Referencias

1. G. Kirchgässner, J. Wolters, and U. Hassler, *Introduction to Modern Time Series Analysis*, 2nd ed. Springer-Verlag Berlin Heidelberg, 2013.
2. P. J. Brockwell and R. A. Davis, *Introduction to Time Series and Forecasting*, 2nd ed. Springer, Mar. 2002.
3. P. S. P. Cowpertwait and A. V. Metcalfe, *Introductory Time Series with R*, 1st ed. Springer Publishing Company, Incorporated, 2009.
4. R. H. Shumway and D. S. Stoffer, *Time Series Analysis and Its Applications With R Examples*. Springer, 2006.
5. "Chaotic behaviour of a parametrically excited damped pendulum," *Physics Letters A*, vol. 86, no. 2, pp. 71 – 74, 1981.
6. J. D. Farmer and J. J. Sidorowich, "Predicting chaotic time series," *Physical review letters*, vol. 59, no. 8, p. 845, 1987.
7. L. P. Maguire, B. Roche, T. M. McGinnity, and L. J. McDaid, "Predicting a chaotic time series using a fuzzy neural network," *Inf. Sci.*, vol. 112, no. 1-4, pp. 125–136, Dec. 1998.
8. M. Casdagli, "Nonlinear prediction of chaotic time series," *Physica D: Nonlinear Phenomena*, vol. 35, no. 3, pp. 335–356, 1989.
9. A. K. Palit and D. Popovic, *Computational intelligence in time series forecasting: theory and engineering applications*. Springer Science & Business Media, 2006.
10. R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.
11. K. Price, R. M. Storn, and J. A. Lampinen, *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media, 2006.

12. S. C. Wheelwright, S. G. Makridakis *et al.*, *Forecasting methods for management*. Wiley, 1985.
13. J. J. Flores, F. Gonzalez-Santoyo, B. Flores, and R. Molina, "Fuzzy NN time series forecasting," in *Scientific Methods for the Treatment of Uncertainty in Social Sciences*, ser. Advances in Intelligent Systems and Computing, J. Gil-Aluja, A. Terceno-Gomez, J. C. Ferrer-Comalat, J. M. Merigo-Lindahl, and S. Linares-Mustares, Eds. Springer International Publishing, 2015, vol. 377, pp. 167–179.
14. J. J. Flores, J. Ortiz, J. R. C. no González, C. Lara, and R. L. Farőas, "FNN a fuzzy version of the nearest neighbor time series forecasting technique," in *Power, Electronics and Computing (ROPEC), 2015 IEEE International Autumn Meeting on*. IEEE, 2015, p. to appear.
15. A. Sorjamaa and A. Lendasse, "Time series prediction using dirrec strategy," in *ESANN*, vol. 6, 2006, pp. 143–148.
16. P. Baldi and S. Brunak, *Bioinformatics: The Machine Learning Approach*, ser. A Bradford book. A Bradford Book, 2001.
17. A. T. Lora, J. C. R. Santos, R. J. L. Martinez, J. R. Santos, and A. G. Exposito, "Influence of kNN-based load forecasting errors on optimal energy production." in *EPIA*, ser. Lecture Notes in Computer Science, F. Moura-Pires and S. Abreu, Eds., vol. 2902. Springer, 2003, pp. 189–203.
18. D. Yankov, D. DeCoste, and E. Keogh, "Ensembles of nearest neighbor forecasts," in *Machine Learning: ECML 2006*. Springer, 2006, pp. 545–556.
19. H. Kantz and T. Schreiber, *Nonlinear time series analysis*. Cambridge university press, 2004, vol. 7.
20. E. De La Vega, J. J. Flores, and M. Graff, "k-nearest-neighbor by differential evolution for time series forecasting," in *Nature-Inspired Computation and Machine Learning*. Springer, 2014, pp. 50–60.
21. C. P. Pappis and C. I. Siettos, "Fuzzy reasoning," in *Search Methodologies*. Springer, 2005, pp. 437–474.
22. W. Siler and J. J. Buckley, *Fuzzy expert systems and fuzzy reasoning*. John Wiley & Sons, 2005.

Programación Genética en Problemas Multidimensionales

Mario Gradd y Eric S. Téllez

Catedrático CONACYT-Centro de investigación e Inovación en Tecnologías de la Información y Comunicación (INFOTEC)
Circuito Tecnopolo Sur No. 112,
Col. Fracc. Tecnopolo Pocitos C.P. 20313, Aguascalientes, Ags. México.
{mario.graff,eric.tellez}@infotec.com.mx;

Recibido 15 de Octubre de 2015, Aceptado 30 de Octubre de 2015,
Versión final 15 de Noviembre de 2015

Resumen Programación Genética (PG) es una técnica evolutiva que permite crear programas de manera automática. Recientemente, ha tomado un gran auge dada su capacidad de resolver problemas de importancia mundial. Sin embargo, el uso de PG para resolver problemas de gran escala y alta dimensión es escaso. Dada la poca y dispersa literatura en el tema, este trabajo presenta una revisión actualizada del estado del arte en PG para problemas con las características mencionadas. Adicionalmente, éste trabajo presenta dos problemas reales de aprendizaje supervisado que se distinguen por tener una gran cantidad de ejemplos y cada individuo mantiene una representación compleja. Se propone un planteamiento de los problemas, su solución con PG, y se analizan cuidadosamente los resultados. Finalmente, se propone un nuevo esquema de PG para ser utilizado en dominios con características similares a nuestros problemas.

Keywords: Programación Genética, Operadores Semánticos Multidimensionales, Regresión, Clasificación

1. Introducción

Dentro de la gran diversidad de algoritmos evolutivos, este estudio se enfoca en **Programación Genética** (PG) [1]. La diferencia que existe entre PG y otros algoritmos evolutivos (por ejemplo, Algoritmos Genéticos) es que PG busca en el espacio de programas. Uno de los problemas que mejor puede ejemplificar la diferencia entre PG y otros algoritmos evolutivos es el de *regresión simbólica*. Regresión simbólica consiste en encontrar una función que se ajuste mejor a un conjunto de pares (x, y), donde x es la entrada o variables independientes, y y es la salida o variable dependiente. La función buscada podría ser la composición de las variables dependientes con las operaciones aritméticas. Por otro lado, los Algoritmos Genéticos se utilizan para resolver el problema de *regresión simple*; el cual trata de identificar el valor de un conjunto de coeficientes.

Programación Genética ha recibido mucha atención recientemente por su capacidad de resolver problemas difíciles con un impacto económico, social y/o industrial (ver [1]). También es conocido que PG es capaz de proveer soluciones competitivas a las de un humano, ejemplo de estas investigaciones son lo ganadores del los premios "Hummies" dados en la conferencia "Genetic and Evolutionary Computation Conference (GECCO)". Aun con estos antecedentes el uso de PG en problemas de alta dimensionalidad y/o con un gran número de ejemplos es todavía escaso.

Una clase de problemas con estas características, es decir, alta dimensionalidad con un gran numero de ejemplos, es el problema de análisis de sentimientos; que es el problema de identificar la polaridad (positiva, negativa y neutra, entre otras) de un texto de forma automática. Se puede observar que el uso de PG es casi nulo siendo la excepción [2]. En un dominio más general como es el análisis automático de texto, la cantidad de trabajos de PG es escaso, siendo algunas excepciones nuestro trabajo previo (ver [3]) donde se usa PG para optimizar los pesos en representaciones vectoriales de los textos; el trabajo de [4] donde se evolucionan características para reducir el número de dimensiones y lo siguientes trabajos [5, 6] donde se trata el problema de resúmenes automáticos.

Haciendo una búsqueda más amplia del uso de PG en problemas de gran escala, i.e., gran numero de ejemplos, podemos encontrar

a [7] donde se usa PG para crear los clasificadores *base* de un ensamble, el problema tiene 300,000 ejemplos con 41 dimensiones. En [8–10] se plantea un problema de regresión simbólica en un conjunto de 1,000,000 de ejemplos con 20 dimensiones. En [11] se trata de aprender un multiplexor de 135 bits lo cual son 2^{135} ejemplos utilizando en la función objetivo alrededor de 1,000,000 de todos los casos posibles. Por otro lado, el aprendizaje en problemas de múltiples dimensiones se ha tratado en [12] donde se propone un nuevo método de regresión simbólica sobre problemas de 340 atributos (i.e., dimensiones) con 600 ejemplos. En [13] se co-evolucionan clasificadores en problemas que van desde 649 a 102,660 atributos en 7,000 ejemplos.

2. Problemas de PG en Problemas de Gran Escala

Esta revisión del estado del arte es una muestra que PG ha sido sub-explorada en problemas con un gran número de ejemplos y de altas dimensiones. Es posible que una de las limitantes del uso de PG en este tipo de problemas es el tiempo de procesamiento que lleva crear una solución aceptable mediante PG, esto ha sido mencionado en [14].

Un posible camino para resolver el tiempo de ejecución es mediante los nuevos operadores genéticos propuestos en la literatura; los cuales tienen una implementación muy eficiente, capaces de evaluar un nuevo individuo en $O(n)$ donde n es el número de ejemplos en el conjunto de entrenamiento. Estos nuevos operadores genéticos son los operadores semánticos geométricos propuestos por Moraglio *et al.* [15] con la implementación de Vanneschi *et al.* [16] y las extensiones a estos operadores propuestas por Castelli *et al.* [17] y Graff *et al.* [18]. En particular estos dos últimos operadores han mostrado tener una alta tasa de convergencia, superior a todos los operadores propuestos en la literatura; sin embargo, han mostrado el problema de sobre-entrenamiento.

Por otro lado, los problemas con un gran número de atributos donde la gran mayoría de ellos son ceros requieren que la representación de estos datos sea en forma de vectores dispersos, por ejemplo,

en el problema de análisis de sentimientos es común tener $> 80,000$ dimensiones y $> 7,000$ ejemplos donde cada uno de estos ejemplos tienen 10 dimensiones diferentes de cero. Claramente, cualquier sistema de aprendizaje, como PG, que no representa sus datos en vectores dispersos hace que la memoria se una limitante y por lo mismo imposibilita su uso. Desafortunadamente, el uso de arreglos dispersos no es una realidad en los sistemas de PG más populares como ECJ [19], GPLAB [20] y TinyGP [21], entre otros.

3. Problemas de Prueba

Antes de describir el enfoque de PG propuesto, es conveniente describir los problemas donde estos enfoques serán estudiados. El primer problema es especificado por la compañía *Springleaf* en la plataforma Kaggle[1], la idea es determinar si un posible cliente responderá a una oferta de crédito enviada directamente a su correo electrónico. Para este efecto la compañía ofreció un gran número $> 145,000$ de clientes cada uno caracterizado por un conjunto de características anónimas $1,782$. El reto consiste en predecir si el cliente responderá al correo aceptando la oferta de préstamo.

El segundo problema se propuso en el "Taller de Análisis de Sentimientos en SEPLN (TASS2015)" [22] la tarea consiste en hacer clasificación de sentimientos para determinar el nivel de polaridad (seis niveles, positivo, muy positivo, neutro, negativo muy negativo y desconocido) de un conjunto de tweets. Este problema tiene $> 80,000$ características y $> 7,000$ ejemplos.

4. Enfoque Propuesto

Como se ha mencionado el uso de operadores semánticos como los propuestos por [17,18] es una de las posibles variantes de PG que puede ser aplicada a los problemas de gran escala con altas dimensiones. De manera general la idea de estos operadores semánticos es crear un hijo que sea una combinación lineal de los padres.

Para explicar el operador semántico propuesto en [18] (denominado *PrGP*) se empieza por describir los operadores semánticos geométricos propuestos por Moraglio *et al.*

[1] http://kaggle.com

Cruza Geométrica Semántica Sean p_1 y p_2 la representación vectorial del primero y segundo padre, con esta notación el hijo producido por estos padres es $o = p_1 r + p_2(1 - r)$, donde r es una función aleatoria o una constante en el rango $[0, 1]$.

Mutación Geométrica Semántica Sea p_1 la representación vectorial del padre a ser mutado y r_1 y r_2 dos funciones aleatorias. En estas circunstancias el hijo producido es $o = p_1 + m(r_1 - r_2)$ donde m es el paso de mutación.

Usando la restricción de que r es una constante, entonces los operadores semánticos geométricos se pueden escribir como $o = \alpha p_1 + \beta p_2$ donde $\alpha = r$ y $\beta = 1 - \alpha$; y la mutación es $o = \alpha p_1 + \beta(r_1 - r_2)$ donde $\alpha = 1$ y $\beta = m$.

Usando esta notación es evidente que los operadores semánticos geométricos imponen una restricción a las constantes α y β. Los nuevos operadores de cruza y mutación remueven estas restricciones perdiendo la definición de operadores geométricos hecha por Moraglio *et al.* [15].

Cruza Sea p_1 y p_2 la representación vectorial (vector renglón) del primero y segundo padre, respectivamente. Entonces el hijo es $o = \alpha p_1 + \beta p_2$ donde α y β se calculan resolviendo la siguiente ecuación $A(\alpha, \beta)' = \mathbf{t}'$ donde $A = (\mathbf{p}_1', \mathbf{p}_2')$ y $\mathbf{t}$ es el vector objetivo.

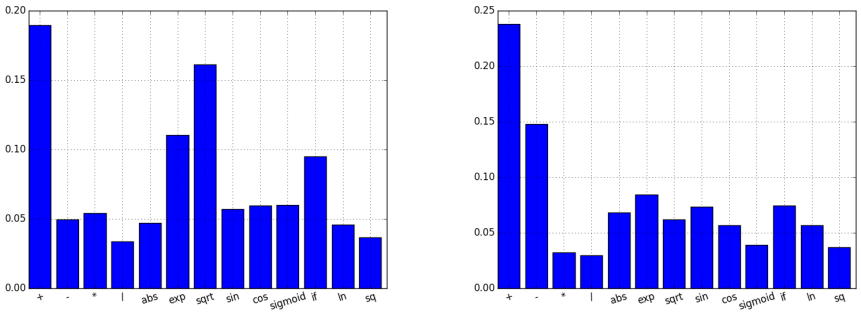
PrMut Sea p_1 la representación vectorial del padre a ser mutado, el hijo es $o = \alpha p_1 + \beta(r_1 - r_2)$ donde r_1 y r_e son dos individuos aleatorios y α y β se obtienen resolviendo $A(\alpha, \beta)' = \mathbf{t}'$ donde $A = (\mathbf{p}_1', (\mathbf{r}_1 - \mathbf{r}_2)')$.

Aunque estos operadores semánticos han mostrado una tasa de convergencia alta, superando a todos los operadores semánticos geométricos, estos presentan dos inconvenientes. El problema de sobre-entrenamiento y la creación de una población inicial factible en los problemas de vectores dispersos de alta dimensionalidad.

El problema de sobre-entrenamiento no es tan agudo cuando el número de ejemplos es grande, en los dos casos de estudio de este trabajo se tienen más de $> 7,000$ ejemplos. Además el problema de sobre-entrenamiento se puede atacar mediante el uso de un conjunto de validación, es decir, se usa un conjunto de validación para identificar el momento donde se presenta el sobre-entrenamiento a esta técnica se le conoce como *early-stopping*.

Cuando se ataca un problema que se representa usando vectores dispersos, como lo es el problema de TASS2015, entonces es necesario crear de manera guiada la población inicial esto porque es muy factible crear individuos los cuales son el vector nulo. Para ejemplificar este problema se presentan en la Figura 1 el histograma (normalizado entre 0 y 1) del uso de funciones en los mejores individuos encontrados por PG en 50 corridas independientes.

Se puede observar en la Figura 1 que los histogramas de Springleaf y TASS2015 son diferentes siendo la característica mas sobresaliente que en el conjunto de TASS2015 la función producto es la que menos utilizada del conjunto de funciones. Además las funciones suma y resta se usan en una mayor proporción en el conjunto TASS2015 que en Springleaf. Continuando con estas diferencias, la función resta es la segunda función más usada en TASS2015 y es una de las menos frecuentes en Springleaf. Estas características indican la necesidad de crear la población inicial guiada o sesgada a mejores individuos.



(a) Springleaf (b) TASS2015

Figura 1. Histograma normalizado de funciones utilizadas en el mejor individuo en 50 corridas independientes en el problema de Springleaf y TASS2015.

Una manera de crear esta población inicial es mediante la ejecución de sistema de PG tradicional y usar los mejores individuos como la población inicial de PrGP. Este es el camino seguido en esta contribución, la única diferencia es que no se uso PG tradicional, en su lugar usamos un sistema memético de PG (denominado *EGP*) que

mostró tener un convergencia mejor que la mayoría de los operadores semánticos propuestos en la literatura, a excepción de PrGP.

Este sistema memético de PG fue propuesto en [23], la idea es que cada individuo en la población sea un bosque, es decir, cada individuo tiene varias expresiones y el resultado final es la combinación lineal de dichas expresiones. Los coeficientes de esta combinación lineal son obtenidos mediante mínimos cuadrados.

5. Resultados

La Tabla 1 muestra los resultados de EGP que se uso como población inicial para PrGP. Para el concurso Springleaf, los resultados son medidos mediante el área bajo la curva ROC, en el conjunto de prueba. Para TASS2015 se utiliza la medida F1 sobre los diferentes niveles de polaridad. Cada problema tiene una medida de aptitud diferente puesto que así fue específicado en los estatutos de cada concurso. En ambos casos se utilizó validación cruzada con 10 folds.

Cuadro 1. Resultados en de Springleaf usado area bajo la curva ROC y del TASS2015 medidos con F1 en los diferentes niveles de polaridad

	Springleaf	P	P+	NEU	N	N+	NONE	Macro
EGP	0,761	0,231	0,452	0,275	0,305	0,126	0,450	0,306
PrGP	0,774	0,289	0,482	0,308	0,344	0,183	0,54	0,358

Se puede observar como en ambos casos de prueba la combinación de EGP con PrGP da mejores resultados que el uso solamente de EGP. Esto indica que EGP no está aprendiendo lo suficiente y además que PrGP logra aprender y no es afectado por sobre-aprendizaje.

Es importante mencionar que aunque esta combinación da mejores resultados no son los mejores resultados encontrados en la literatura. Por ejemplo, en el concurso Springleaf los mejores resultados se han obtenido con otras técnicas como ensambles de árboles de decisión mediante *boosting*. En el caso de TASS2015 se puede comparar contra nuestro trabajo previo (ver [24]) donde se usó una máquina de soporte vectorial lineal y desafortunadamente PG no obtiene mejores resultados.

6. Conclusiones

Como primera contribución, este trabajo presentó una revisión del estado del arte del uso de Programación Genética en problemas de gran escala (muchos ejemplos) de altas dimensiones. La brevedad de la revisión hace evidencia de que el uso de PG es escaso en este tipo de problemas. También se listaron varios de los problemas que afectan a PG y otras técnicas de aprendizaje supervisado. Finalmente, se propuso una combinación de sistemas de PG para atacar problemas de gran escala con múltiples dimensiones. Este nuevo esquema de PG se probó en dos concursos; mostrando que el esquema propuesto es viable. Sin embargo, es necesario hacer modificaciones para superar los resultados obtenidos por otras técnicas de aprendizaje supervisado.

Referencias

1. Riccardo Poli, William B. Langdon, and Nicholas Freitag McPhee. *A field guide to genetic programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`, 2008. (With contributions by J. R. Koza).
2. Shilpa Arora, Elijah Mayfield, Carolyn Penstein-RosÃľ, and Eric Nyberg. Sentiment Classification Using Automatically Extracted Subgraph Features. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, CAAGET '10, pages 131–139, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. 00030.
3. Hugo Jair Escalante, Mauricio A. Garcia-Limon, Alicia Morales-Reyes, Mario Graff, Manuel Montes-y Gomez, Eduardo F. Morales, and Jose Martinez-Carranza. Term-weighting learning via genetic programming for text classification. *Knowledge-Based Systems*, 2015. 00000.
4. Elijah Mayfield and Carolyn Penstein-Rosé. Using Feature Construction to Avoid Large Feature Spaces in Text Classification. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO '10, pages 1299–1306, New York, NY, USA, 2010. ACM. 00013.
5. Carlos Nascimento Silla Jr., Gisele L. Pappa, Alex Alves Freitas, and Celso A. A. Kaestner. Automatic text summarization with genetic algorithm-based attribute selection. In Christian Lemaître, Carlos A. Reyes, and Jesús A. González, editors, *Advances in Artificial Intelligence - IBERAMIA 2004, Proceedings 9th Ibero-American Conference on AI*, volume 3315 of *Lecture Notes in Computer Science*, pages 305–314, Puebla, Mexico, November 22-26 2004. Springer.
6. Nguyen Quang Uy, Pham Tuan Anh, Truong Cong Doan, and Nguyen Xuan Hoai. A study on the use of genetic programming for automatic text summarization. In Hung Dang-Van and Jeff Sanders, editors, *The Fourth International Conference on Knowledge and Systems Engineering, KSE 2012*, pages 93–98, Danang, Vietnam, 17-19 August 2012.

7. Yifeng Zhang and Siddhartha Bhattacharyya. Genetic programming in classifying large-scale data: an ensemble method. *Information Sciences*, 163(1-3):85–101, June 2004. 00061.
8. Michael F. Korns. Large-Scale, Time-Constrained Symbolic Regression. In Rick Riolo, Terence Soule, and Bill Worzel, editors, *Genetic Programming Theory and Practice IV*, Genetic and Evolutionary Computation, pages 299–314. Springer US, 2007. 00019 DOI: 10.1007/978-0-387-49650-4_18.
9. Michael F. Korns. Large-Scale, Time-Constrained Symbolic Regression-Classification. In Rick Riolo, Terence Soule, and Bill Worzel, editors, *Genetic Programming Theory and Practice V*, Genetic and Evolutionary Computation Series, pages 53–68. Springer US, 2008. 00020 DOI: 10.1007/978-0-387-76308-8_4.
10. Michael F. Korns and Loryfel Nunez. Profiling Symbolic Regression-Classification. In *Genetic Programming Theory and Practice VI*, Genetic and Evolutionary Computation, pages 1–14. Springer US, 2009. 00011 DOI: 10.1007/978-0-387-87623-8_14.
11. M. Iqbal, W.N. Browne, and Mengjie Zhang. Reusing Building Blocks of Extracted Knowledge to Solve Complex, Large-Scale Boolean Problems. *IEEE Transactions on Evolutionary Computation*, 18(4):465–480, August 2014. 00019.
12. Trent McConaghy. Latent Variable Symbolic Regression for High-Dimensional Inputs. In Rick Riolo, Una-May O'Reilly, and Trent McConaghy, editors, *Genetic Programming Theory and Practice VII*, Genetic and Evolutionary Computation, pages 103–118. Springer US, 2010. 00007 DOI: 10.1007/978-1-4419-1626-6_7.
13. John Doucette, Peter Lichodzijewski, and Malcolm Heywood. Evolving Coevolutionary Classifiers Under Large Attribute Spaces. In Rick Riolo, Una-May O'Reilly, and Trent McConaghy, editors, *Genetic Programming Theory and Practice VII*, Genetic and Evolutionary Computation, pages 37–54. Springer US, 2010. 00008 DOI: 10.1007/978-1-4419-1626-6_3.
14. Pedro G Espejo, Sebastián Ventura, and Francisco Herrera. A survey on the application of genetic programming to classification. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 40(2):121–144, 2010.
15. Alberto Moraglio, Krzysztof Krawiec, and Colin G. Johnson. Geometric semantic genetic programming. In Carlos A. Coello Coello, Vincenzo Cutello, Kalyanmoy Deb, Stephanie Forrest, Giuseppe Nicosia, and Mario Pavone, editors, *Parallel Problem Solving from Nature - PPSN XII*, number 7491 in Lecture Notes in Computer Science, pages 21–31. Springer Berlin Heidelberg, January 2012.
16. Leonardo Vanneschi, Mauro Castelli, Luca Manzoni, and Sara Silva. A new implementation of geometric semantic GP and its application to problems in pharmacokinetics. In Krzysztof Krawiec, Alberto Moraglio, Ting Hu, A. Åđima Etaner-Uyar, and Bin Hu, editors, *Genetic Programming*, number 7831 in Lecture Notes in Computer Science, pages 205–216. Springer Berlin Heidelberg, January 2013.
17. Mauro Castelli, Leonardo Trujillo, Leonardo Vanneschi, Sara Silva, Emigdio Z-Flores, and Pierrick Legrand. Geometric Semantic Genetic Programming with Local Search. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*, GECCO '15, pages 999–1006, New York, NY, USA, 2015. ACM. 00000.
18. Mario Graff, Eric Sadit Tellez, Elio Villasenor, and Sabino Miranda-Jiménez. Semantic genetic programming operators based on projections in the phenotype space. *Research in Computing Science*, 94:73–85, 2015.
19. DavidR. White. Software review: the ecj toolkit. *Genetic Programming and Evolvable Machines*, 13(1):65–67, 2012.

20. Sara Silva. Gplab: A genetic programming toolbox for matlab. http://gplab.sourceforge.net.
21. Riccardo Poli. TinyGP. See Genetic and Evolutionary Computation Conference (GECCO-2004) competition at http://cswww.essex.ac.uk/staff/sml/gecco/TinyGP.html, June 2004.
22. Julio Villena RomÃąn, Janine GarcÃŋa Morera, Miguel ÃĄngel GarcÃŋa Cumbreras, Eugenio MartÃŋnez CÃąmara, M. Teresa MartÃŋn Valdivia, and L. Alfonso UreÃśa LÃşpez. Overview of tass 2015. *CEUR Workshop Proceedings*, 1397:13–21, 2015.
23. Mario Graff, Eric S. Tellez Hugo Jair Escalante, and Jose Ortiz-Bejar. Memetic genetic programming based on orthogonal projections in the phenotype space. IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC2015) (under review).
24. Oscar S. Siordia, Daniela Moctezuma, Mario Graff, Sabino Miranda-Jimenez, Eric S. Tellez, and Elio-Atenogenes Villaseñor. Sentiment analysis for twitter: Tass 2015. *CEUR Workshop Proceedings*, 1397:65–70, 2015.

Aprendizaje con Hadoop

Eugenia Méndez Macías y René MacKinney Romero

Universidad Autónoma Metropolitana, Departamento de Ingeniería Eléctrica,
San Rafael Atlixco 186, C.P. 09340 Vicentica, Iztapalapa, México, D.F.
memeeugenia@gmail.com
rene@xanum.uam.mx
http://www.izt.uam.mx//

Recibido: 15 de Octubre de 2015, Aceptado: 15 de octubre de 2015,
Versión final: 15 de noviembre de 2015

Resumen El Aprendizaje Maquinal (*Machine Learning*) desarrolla técnicas que permiten a las computadoras aprender. Para poder generar este aprendizaje se analiza mucha información la cual se trata de convertir en conocimiento. Dentro del aprendizaje se desarrolla un nuevo termino llamado Minería de Datos. La minería analiza información extrayendo de manera automática conocimiento y descubre patrones complejos. Dentro de la minería de datos existen algoritmos de clasificación. La clasificación en la última década se realiza sobre miles y millones de datos a lo cual se le conoce como Big Data. Hadoop es un marco que ayuda a paralelizar los algoritmos de clasificación de una manera sencilla siguiendo el esquema de Mapeo-Reducción. En este trabajo se comparan medidas de eficacia (exactitud y F-M) de tres algoritmos de clasificación (Bayes Ingenuo, Bosques Aleatorios, K-vecinos más cercanos) desarrollados en Hadoop, aplicados en la base de datos de correo Enron (*Ham/Spam*).

Palabras Clave: Big Data, Bayes Ingenuo, ID3 (RandomForest), KNN, Hadoop

1. Introducción

Desde la primera generación de computadoras en los años cuarentas, la cual no fue un modelo comercial, estaba llena de tubos y era del tamaño de todo un sótano, los deseos que se tenían eran demasiados, ya que no

se sabía de que eran capaces dichas maquinas. Surgió la ilusión de que las computadoras podrían aprender, sin embargo era demasiado ambicioso creer que una maquina creada por el ser humano fuera capaz de mejorar automáticamente con la experiencia. La tarea se había vuelto vuelto más compleja ya que no sólo tenía que aprender, sino una vez aprendida la acción esta debía ser mejorada al repetirla constantemente.

Poco tiempo después surgió el Aprendizaje Maquinal (*Machine Learning*) del cual surgieron diversas ramas como la Minería de Datos. Que sirve para ayudar a analizar información extrayendo de manera automática conocimiento a partir de conjuntos de ejemplos y descubrir patrones complejos. Dentro de la minería de datos existen cuatro categorías de algoritmos desarrollados: agrupamiento (*clustering*), regresión, métodos por regla de asociación y clasificación. La clasificación en la última decada se realiza sobre miles y millones de datos a lo cuál se le conoce como (*Big Data*).

En este trabajo se abordaran tres algoritmos de clasificación: Bayes Ingenuo (*Naive Bayes NB*), K vecinos más cercanos (*K-nearest neighbors KNN*) y árboles de decisión (*ID3*) con el esquema de Bosques Aletorios (*RandomForest RF*). Dentro de los algoritmos de clasificación el esquema que se sigue es asociar material no conocido dadas sus características con material previamente clasificado. Los algoritmos generalmente tienen una etapa de entrenamiento y una de prueba.

El objetivo del trabajo es encontrar trabajos donde se ocupen los mismos tres algoritmos de clasificación (NB, RF, KNN) desarrollados en una estructura secuencial utilizando la bases de datos Enron, y comparar las medidas de eficacia (exactitud y F-M) con los algoritmos implementados por nosotros en Hadoop.

El esquema de Hadoop esta basado en el método original llamado MapReduce realizado sobre el esquema divide y vencerás, este esquema se divide en dos partes, una llamada Mapeo y la otra Reducción:

Mapeo: El nodo principal reparte el problema en pequeños subproblemas. Un nodo llamado nodo-trabajador, procesa los subproblemas bajo el control de un nodo que administra y almacena el resultado en el sistema local donde la fase de reducción accede [1].

Reducción: En este paso se analizan y combinan todos los datos de entrada de la fase de mapeo. Pueden existir múltiples tareas a reducir las cuales pueden ejecutarse de forma paralela, estas tareas se ejecutan en el nodo-trabajador bajo el control de un nodo-administrador [1].

2. Trabajos Relacionados

Se han mencionado las grandes cantidades de datos por su nombre (*Big Data*), pero no se ha especificado que estos datos pueden ser representados en muchas formas. Nosotros nos interesamos en el desmesurado crecimiento de correos electrónicos los cuales, como mencionan Almeida et al. [2], existen diversos métodos para deshacerse de los correos no deseados (*Spam*) y dentro de los mejores algoritmos de aprendizaje maquinal se encuentra Bayes Ingenuo y Maquinas de Soporte Vectorial (*Support Vector Machines SVM*). Debido a que tiene tanto peso NB en la clasificación de correos, los autores decidieron implementar siete versiones distintas de este método sobre la base de datos Spam/Ham Enron. Para validar los siete diferentes clasificadores se propone utilizar la memoría, precisión (tanto de los Spam como de los Ham), exactitud y el coefficiente de correlación de Matthews (*Matthews Correlation Coefficient MCC*).

Por otra parte debido a el desbalance de clases en diversas bases en el trabajo de Kumor et al. [3], ellos nombran que otra medida muy valiosa dentro del área de clasificación de texto es la tasa de falsos positivos. Esta medida se comparó en distintos algoritmos entre los que estan Programación Genética (*Genetic Programs*), NB, J48 (variación de árboles), RF, SVM. Para probar esta implementación se utilizará la base de Enron y SpamAssassin.

Para mejorar las medidas de desempeño de los filtros en correo Spam, los algoritmos van teniendo atributos más específicos, como en el reporte de Shams et al. [4] donde se describe una técnica llamada Sentinel: se compone de atributos comúnes más atributos relacionados con rasgos característicos y preferencias del autor. Se útilizan otras técnicas como AdaBoosTM1, Bagging RF, RF, SVM y NB sobre Enron.

Para nuestro último clasificador KNN se tuvo que pre-procesar la base Spam/Ham con el fin de calificar a cada palabra en base a su importancia en el texto. Podemos apreciar con más detalle en el proceso de filtrado de Harisinghaney et al. [5] donde manejan KNN, NB y DBSCAN. Dentro de la forma de obtener el procesamiento se cuenta la frecuencia de cada palabra y se sigue un algoritmo de *Stemmer Porter*, donde se quitan los plurales y sufijos, preparando los números de incidencias de las palabras para ocupar el método de KNN.

Un problema muy marcado en KNN es que al no tener etapa de entrenamiento todos los ejemplos del conjunto son guardados y esto crea alto requerimiento de almacenamiento. Esta complejidad en distancia de computación aumenta abruptamente a medida que la dimensión de datos crece y cuando los valores K grandes se consideran para la clasificación.

En este trabajo se utilizó un método llamado *KNNJoin* el cual se describirá posteriormente pero Chakrabarty et [6], desarrollaron un algoritmo mejorado de *K*NN donde se crean centros conglomerados los que se toman como puntos representativos que hacen el conjunto de entrenamiento más pequeño y con el fin de superar el defecto de diferenciar las características de cada palabra les otorgaron un valor de peso que puede indicar diferentes grados de importancia de la muestra.

3. Descripcion de la Aplicación

3.1. Algoritmos

Bayes Ingenuo es un método que utiliza la regla de Bayes y algunas medidas estadísticas, las cuales son la media y la varianza. El modelo trata de definir cuantas veces se repite un ejemplo dado que pertenezca a una clase u otra. La suposición fundamental de Bayes Ingenuo es que los valores son independientes entre sí, lo cual no es siempre cierto. Bayes Ingenuo trabaja con probabilidades y siempre la probabilidad mayor será la que clasifique al ejemplo, se espera, adecuadamente en su clase.

Los árboles de decisión han sido uno de los métodos más desarrollado en los últimos años debido a que su estructura es muy fácil de leer (parecida a un diagrama de flujo) una vez que ha sido creado el árbol de entrenamiento, sin embargo su valor computacional, en tiempo es muy alto, por lo que se ha optado en ocasiones por métodos más sencillos.

ID3 crea un árbol de decisión el cual utiliza medidas de información para generar su estructura, utiliza la Entropía de la Información para saber cual es el mejor atributo y la Ganancia de la Información que ocupa los valores de la Entropía para decidir cual será el nodo que aporta más información. Al inicio se escoge un nodo raíz y se continua recorriendo las ramas creando nuevos nodos hasta llegar a una Entropía igual a cero u otro criterio de paro.

En el presente trabajo se utilizó una variante de los árboles de desición llamada *RandomForest* (RF) donde se crean árboles de distintas longitudes y al momento de clasificar los ejemplos de prueba se recorren cada uno de los árboles y se usa la moda de la clase.

El algoritmo de *K*NN es de los más sencillos ya que no utiliza etapa de entrenamiento porque trabaja con ejemplos clasificados y requiere de ejemplos de prueba a clasificar. Pero los beneficios de este paso son pagados por un alto costo en memoria ya que se necesita guardar todos los ejemplos previos para realizar la clasificación.

*K*NN es un método que utiliza la regla de vecinos más cercanos (*Nearest Neighbors*) para clasificar al valor que se encuentre más cercano del

punto de prueba. Se utilizan distancias para diferenciar entre los ejemplos parecidos y los que no lo son. La K es una variable que se creo en el caso de existir un sesgo en los datos generado por tener un densidad no equivalente entre clases.

3.2. Base de Datos

La base de datos utilizada para nuestro trabajo es la de Enron la cual cuenta con 33716 correos de los cuales 16551 son correos valiosos (con contenido valioso en el correo para el lector, *Ham*) y 17165 son correos basura (sin ningún contenido valioso, *Spam*). La base cuenta con un peso de 53 MB de datos. Los datos se encuentran divididos en seis mangas de aproximadamente el mismo tamaño.

En la tabla 1 se muestra la distribucíon de los datos de la base Enron. Se puede observar que el sesgo de cada manga varía siendo a veces más de Ham que de Spam y en otras ocasiones viceversa.

Cuadro 1. Base de Datos Enron

Enron 1		Enron 2		Enron 3		Enron4		Enron 5		Enron 6	
Ham	Spam	Ham	Spam	Ham	Spam	Ham	Spam	Ham	Spam	Ham	Spam
3673	1499	4362	1495	4013	1499	1501	4499	1501	3674	1501	4499

3.3. Hadoop

Bayes Ingenuo: La estructura de contar palabras parece muy sencilla ya que es un algoritmo básico, sin embargo en hadoop es mucho más complicado ya que al inicio el código comenzó con un Mapeo y Reducción para cada variable que se quería calcular. Esto posteriormente se cambio para que las variables en la parte de Mapeo y Reducción se realicen de forma paralela. Con lo que se obtuvo un código más simple y eficiente.

Bosques Aleatorios: En la clasificación se ocuparon librerias de Weka para formar la estructura de ramas, sin embargo Hadoop generá una partición natural en generar árboles y es muy sencillo crear desde un arbol de longuitud pequeña hasta uno de longitud enorme. Se eligió la técnica de Bosques Aletorios por la facilidad con la que la paralelización de Hadoop crea un bosque completo.

K vecinos más cercanos: Dentro de la implementación de este algoritmo en Hadoop al inicio se presentaron varios problemas de tiempo y de memoria debido a las características de la técnica. Se decidio cambiar el

algoritmo a uno más especializado llamado KNNJoin, el cual divide el el conjunto de datos de cubetas (*buckets*) y cada cubeta trabaja de manera paralela. El método que en un inicio tardo días en sacar resultados más tarde era cuestión de minutos para obtenerlos.

4. Conceptos Básicos

Hadoop esta basado en el diseño de MapReduce siendo una herramienta útil para implementar de forma muy sencilla cualquier tipo de algoritmo de forma paralela, el almacenamiento de las bases y los archivos generados tienen un almacenaje distribuido. Es de suma importancia la escalabilidad de este sistema ya que en la última década los datos han sido desmesurados y la escalabilidad con la que cuenta Hadoop nos auxilia en trabajar con bases muy pequeñas hasta volumenes de Terabytes. La estructura de Hadoop tiene tolerancia a fallos ya que crea replicas de los datos que son repartidos a cada nodo exclavo y se encuentra pendiente actualizando el estado de cada unos de sus colaboradores.

Al elegir la base Spam/Ham Enron se planeó buscar los métodos con los cuales se ha trabajado esta base. Se escogieron tres clasificadores los cuales se adaptaron de forma sencilla y encajando con la estructura de MapReduce para competir con las medidas de eficacia existentes en el estado del arte. Al ocupar Hadoop los 53 MB con los que cuenta Enron no tuvieron ningun problema en tiempo, ni la forma caracterizar las pruebas por hacer.

5. Resultados y Pruebas

Dentro de una base desbalanciada como Enron la literatura sugiere obtener más cantidad de medidas de eficacia, si embargo fue dificil encontrar trabajos que reportaran todas estos valores por los cuales el trabajo de comparación se vio limitado a la exactitud y la F-M.

En la Figura 1 se presenta la comparación de cuatro autores que utilizaron el algoritmo de Bayes Ingenuo en la base de Enron, podemos observar que el algoritmo de Hadoop (color rojo) es competitivo con autores como Almeida et al.[2] donde se probo con siete distintas variaciones de este algoritmo y se concluyó que el algoritmo Básico de Naive Bayes sigue siendo categorizado como uno de los mejores clasificadores de texto.

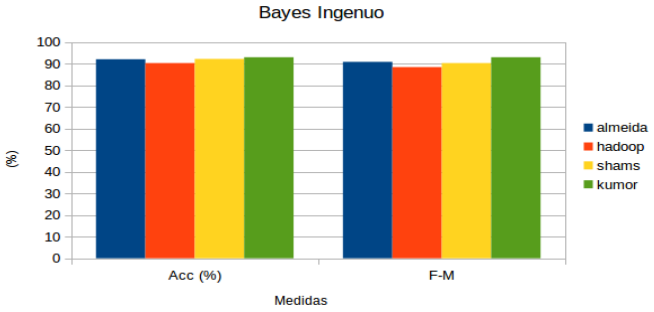


Figura 1. Comparación de 4 autores con Bayes Ingenuo

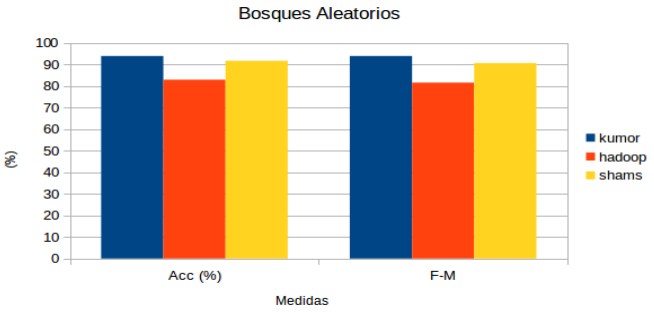


Figura 2. Comparación de 3 autores con Bosques Aleatorios

En la Figura 2 se presenta la comparación de tres autores que utilizaron el algoritmo de Bosques Aleatorios en la base de Enron, podemos observar que el algoritmo de Hadoop (color rojo) es competitivo con autores como Kumor et al. [2] y Shams et al. [4].

En la Figura 3 se presenta la comparación de tres autores que utilizaron el algoritmo de K vecinos más cercanos en la base de Enron, podemos observar que el algoritmo de Hadoop (color rojo) es competitivo con autores incluso en la gráfica se alcanza a notar que tuvo la estimación más alta en la exactitud. Podemos agregar que Harisinghaney et al. [5] tuvo un mejor rendimiento en la F-M pero en la exactitud no y Chakrabarty et al. [6] no tuvo mejor rendimiento ni en la F-M ni en la exactitud.

6. Conclusiones

La medidas de eficacia, exactitud y F-M fueron competitivas para los tres algoritmos de minería de datos obtenidas por los cinco autores en

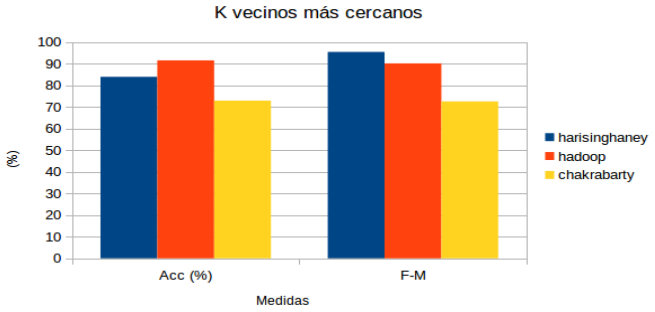


Figura 3. Comparación de 3 autores con K vecinos más cercanos

la base de datos Enron. Es importante recalcar lo fácil que fue convertir estos algoritmos que son generalmente secuenciales a un esquema paralelo, ya que Hadoop es responsable de esta estructura. Hadoop, aparte de tener una alta tolerancia a fallos, trabajar en una estructura multinodal y ser de fácil implementación, se encuentra competitivo con valores de otros autores, siendo que en algunos su principal objetivo es obtener mejoras en las medidas que obtienen de la clasificación. La base Enron es muy práctica para este tipo de trabajos ya que no es pequeña y como es texto es fácil aplicar cualquier tipo de pre-procesamiento.

Referencias

1. Dean, Jeffrey and Ghemawat, Sanjay (2008). *MapReduce: simplified data processing on large clusters*, Communications of the ACM, volume 51, number 1, pages 107–113, ACM
2. Almeida, Tiago and Yamakami, Akebo and others (2010). *ontent-based spam filtering*, Communications of the ACM, Neural Networks (IJCNN), The 2010 International Joint Conference on, pages 1-7, IEEE
3. Trivedi, Shrawan Kumar and Dey, Shuvashis (2013). *An Enhanced Genetic Programming Approach for Detecting Unsolicited Emails*, Computational Science and Engineering (CSE), 2013 IEEE 16th International Conference on, pages 1153–1160, IEEE
4. Shams, Reza and Mercer, Robert E (2013). *Personalized Spam Filtering with Natural Language Attributes*, Machine Learning and Applications (ICMLA), 2013 12th International Conference on, volume 2, pages 127-132, IEEE
5. Harisinghaney, Anirudh and Dixit, Abhishek and Gupta, Swastik and Arora, Abhishek (2014). *ext and image based spam email classification using KNN, Naïve Bayes and Reverse DBSCAN algorithm*, Optimization, Reliabilty, and Information Technology, International Conference on, pages 153–155, IEEE
6. Chakrabarty, Ankush and Roy, Sandip (2014). *An optimized k-NN classifier based on minimum spanning tree for email filtering*, Business and Information Management (ICBIM), 2014 2nd International Conference on, pages 47-52, IEEE

Selección de Estrategias en Juegos Deportivos de Equipo

Matías Alvarado

Centro de Investigación y de Estudios Avanzados del IPN, Departamento de Computación,
Av. IPN 2508 C.P. 07360 San Pedro Zacatenco, DF, México
matias@cs.cinvestav.mx
http://www.cs.cinvestav.mx/~matias

Recibido 15 de Octubre de 2015, Aceptado 30 de Octubre de 2015,
Versión final 15 de Noviembre de 2015

Resumen Una adecuada selección de estrategias en juegos deportivos, como el béisbol y el futbol Americano, es fundamental para aumentar la posibilidad de triunfo. En tales juegos es necesario coordinar las jugadas entre jugadores, de alta complejidad debido a la cantidad de jugadores involucrados y las muy diversas combinaciones de jugadas. Nosotros aplicamos los modelos matemáticos de equilibrio de Nash y el de eficiencia de Pareto para modelar estrategias cooperativas en los juegos mencionados, cuya traducción algorítmica se usa en simulaciones computacionales de duelos entre equipos. Considerar la frecuencia de ocurrencia de las jugadas, es esencial para obtener simulaciones que correspondan a lo que ocurre en partidos reales. Usualmente, en los perfiles de estrategia que satisfacen el equilibrio de Nash se integran jugadas con mayor frecuencia de ocurrencia, y en los perfiles de estrategia que satisfacen la eficiencia de Pareto se incorporan las mejores jugadas, pero con menor probabilidad de ocurrencia.

Palabras Clave: Juegos de equipo, equilibrio de Nash, Eficiencia de Pareto, Estrategias, Cooperación.

1. Introducción

En los juegos de múltiples jugadores organizados en equipos, la cooperación eficaz entre los jugadores puede modelarse utilizando el equilibrio

de Nash [5], el cual establece que la estrategia de cada jugador deber ser la de menor riesgo frente a las de los demás. Tal cooperación es el punto en nuestro trabajo: utilizar un modelo cuyo objetivo original, planteado por el autor en el título del trabajo *Non Cooperative Games*, para tales tipos de juegos [5], puede aplicarse de manera significativa para el diseño de estrategias de cooperación eficaces en juegos de equipo como es el beisbol [1, 2, 3] y el futbol Americano [6, 8].

Un juego comprende un curso de acciones por parte de los jugadores, tale que, bajo reglas bien definindas compiten, individualmente o en equipo, para lograr un mismo objetivo [2]. Un juego es susceptible de análisis matemático, en tanto sistema de reglas establecidas sin ambiguedad, incluyendo el conteo del resultado final del juego. En los juegos de múltiples jugadores, como el beisbol y el futbol, la selección de estrategias implica un análisis altamente complejo, con un gran abanico de combinaciones posibles. Por esta razón es relevante computarizar, con algoritmos eficientes un modelo formal suficientemente expresivo y preciso [4, 7, 8].

En un juego, las estrategias son conjuntos de acciones organizadas y priorizadas entre si con el objetivo de obtener el beneficio en disputa de manera eficaz [8]. Cada jugador debe ajustarse a las reglas del juego para determinar sus acciones y la diferencia a favor de un equipo se debe uso de estrategias para ganar el partido en disputa. Para ganar en equipo se requiere el diseño de estrategias colectivas como combinación positiva de las estrategias individuales. La Teoría de Juegos y técnicas de Inteligencia Artificial se combinan para el análisis formal y la simulación computacional de los comportamiento de individuos en los juegos [8]. Junto con estadísticas y resultados experimentales verificables, se sustentan recomendaciones para aumentar las opciones de ganar partidos.

El equilibrio de Nash (EN) [5] es un concepto en Teoría de Juegos [1], relevante para formalizar la interacción entre jugadores, para cooperar si son del mismo equipo o para competir si son de equipos diferentes. El EN habilita el caracterizar las estrategia colectivas tales que cada jugador, individualmente, elija la jugada que menor riesgo le conlleva. Bajo esta premisa es posible lograr una estrategia colectiva eficaz. Utilizando EN el modelo de coordinación de los jugadores es de tal manera que cada uno de ellos actúe para potenciar el beneficio del equipo.

Nosotros utilizamos el equilibrio de Nash para realizar el análisis de estrategias en los juegos de beisbol y futbol Americano y de esta manera guiar el proceso de toma de decisiones y seleccionar un curso de acción entre diferentes alternativas [8]. El objetivo es identificar las situaciones y condiciones durante el desarrollo de un juego, tal que aplicando el modelo de equilibrio de Nash o de eficiencia de Pareto, las estrategias

para fortalecer las opciones ganadoras del equipo correspondan a lo que estadísticamente es de mayor viabilidad.

1.1. Juego en formal normal

La representación formal de un juego es la llamada forma normal que incluye, para todos y cada uno de los jugadores, las estrategias que puede utilizar y una función de valuación individual, que le permite califica cada una de las combinaciones o perfiles de estrategia de las jugadas del equipo, y puede compar su calificación con la asignada por los otros jugadores a cada perfil de estrategia. El perfil de estrategias es una n-tupla de estrategias donde cada posición corresponde a cada uno de los n jugadores para especificar la acción que puede realizar en el juego [8].

Un juego se representa en forma normal $G = \{S_1, \ldots, S_n; u_1, \ldots, u_n\}$, donde: los jugadores son $1, \ldots, n$, para cada jugador i su conjunto de estrategias es S_i, y $u_1, \ldots, u_n$ son las funciones de costo-beneficio, una por jugador. El espacio de combinaciones de estrategias de los jugadores está dado en el producto cartesiano $S = S_1 \times \ldots \times S_n$, de tal manera que todo vector elemento $s \in S$ es un perfil de estrategias del juego. Para cada jugador, su función de costo-beneficio o rentabilidad (*payoff*) u_i le permite calcular el beneficio a obtener $u_i(s)$ en cada perfil de estrategia.

1.2. El equilibrio de Nash

EN [8] es un concepto ampliamente utilizado en Teoría de Juegos para encontrar perfiles de estrategia que sean solución a juegos de dos o más jugadores, tomando en cuenta que los perfiles deben ser la mejor de menor riesgo de cada jugador, y condicionadas con las estrategias de los demás. En general, para un juego en forma normal de n jugadores, $G = \{s_1, \ldots, s_n; u_1, \ldots, u_n\}$, el perfil de estrategias $(s_1*, \ldots, s_n*)$, satisface el EN, si para cada jugador $i, s_i* \in S_i$ es una respuesta de menor riesgo frente a las estrategias de los otros jugadores, $(s_1*, \ldots, s_{i-1}*, s_{i+1}*, \ldots, s_n*)$.

$$u_i(s_1*, \ldots, s_{i-1}*, s_i*, s_{i+1}*, \ldots, s_n*) \geq u_i(s_1*, \ldots, s_{i-1}*, s_i, s_{i+1}*, \ldots, s_n*). \quad (1)$$

$\forall si \in Si$ la solución EN es $Max\{u_i(s_1*, \ldots, s_{i-1}*, s_i, s_{i+1}*, \ldots, s_n*)\}$.

Es relevante observar que el equilibrio de Nash no necesariamente identifica que cada jugador esté alcanzando el mejor resultado posible, sino el mejor resultado condicionado por el hecho de que los demás jugadores jueguen las estrategias elegidas por ellos en dicho perfil [3].

En la Sección 2 se introducen los juegos multi-jugador: el dilema del prisionero para 2 jugadores, el beisbol y el futbol Americano. En la Sección 3 se describe el equilibrio de Nash y su aplicación para elegir estrategias en los juegos antes mencionados. En la Secciónn 4 se hace la discusión, cerrando el documento con las conclusiones en la Sección 5.

2. Juegos multi-jugador

2.1. El dilema del prisionero

El ejemplo clásico para ilustrar el uso del EN es el Dilema del Prisionero: la policía arresta a dos sospechosos sin pruebas suficientes para inculparlos de un delito; tras separarlos, a cada uno se le ofrece un mismo trato: si uno colabora con la policia recibe la mitad de la pena ($r/2$) o se va libre ($r = 0$), dependiendo si el cómplice, respectivamente, colabora o no, en cuyo caso el complice recibe, asimismo, la mitad o el total de la pena. Observese que si ambos colaboran con la policia comparten la pena, y si ninguno colobora con la policia, no habiendo elementos para inculparlos, ambos se van libres o tendrán una pena menor.

Los perfiles de estrategia son: (callar, callar), (callar, confesar), (confesar, callar) y (confesar, confesar). Las posiciones ordenadas corresponde al prisionero p_1 y p_2. El EN indica elegir como estrategia la que conlleva el menor riesgo posible. Cada prisionero no sabe con certeza lo que hará el otro, pero si sabe que si no colabora con la policia se arriesga a absorber la pena total si el otro colabora con la policia. Tales situaciones las describen los perfiles de estrategia (callar, confesar), (confesar, callar), respectivamente, simétricos entre sí. También sabe que si colabora, puede tener un escenario de media o nula pena, respectivamente (confesar, confesar), (callar, callar).

2.2. El beisbol

El béisbol es jugado entre dos equipos, de 9 jugadores cada uno, sobre campo donde los jugadores ofensivos corren entre las 4 bases ubicadas en los vértices de un área cuadrangular en forma de diamante. El área del lanzador *(pitcher)* de la bola en el campo es una pequeña loma de tierra frente al área *(home)* del bateador, cuyo objetivo es golpear la pelota con un bate, tal que se desplace lo más posible a través del campo y correr las bases 1a, 2a, 3a, volver a *(home)* y así anotar una carrera. Los jugadores defensivos buscan la pelota bateada para eliminar al jugador que bateó la pelota o a otros corredores, antes que estos lleguen a alguna de las bases o consigan anotar la carrera. El equipo que anote más carreras al cabo de

los nueve entradas *(innings)* del encuentro, es el ganador. Si al término de los nueve entradas regulares hay empate, el encuentro se extiende cuanto sea necesario hasta haber un ganador (no existe el empate), en ligas profesionales [7].

En este caso los vectores con combinaciones de jugadas posibles para el equipo a la ofensiva, sondel tipo (*homerun*, alcanzar-base-1, alcanzar-base-2, alcanzar-base-3, alcanzar-*home*); y para el equipo a la defensiva son del tipo (lanzar-bola, atrapar-bola-base, atrapar-bola-jardin).

2.3. El futbol Americano

El Futbol Americano se juega con equipos de 11 jugadores, a la ofensiva tiene posesión del balón y busca avanzar con el ovoide por el terreno, acarreando o pasando el balón a un receptor, y anotar puntos al entrar con el balón a un área del terreno de juego conocido como la zona de anotación. El equipo a la defensiva busca frenar la ofensiva y obligarle a ceder la posesión del balón. Si la ofensiva anota o es obligada a ceder la posesión, los equipos intercambian papeles, y el equipo a la ofensiva coloca su cuadro defensivo en el terreno mientras el equipo que defendía coloca su cuadro ofensivo) El proceso se repite, en una dirección y otra, hasta que se jueguen los cuatro cuartos del partido. Si el marcador está empatado tras el fin del último cuarto, el partido pasa a la prórroga.

El avance en un terreno de juego de futbol americano se mide en yardas. Hay cuatro maneras distintas de anotar: *touchdown* vale seis puntos y el ovoide debe acarreado a través de la línea de gol, o atrapado dentro de la zona de anotación. Punto Extra y la Conversión, de 1 o 2 puntos. Tras la anotación de un *touchdown*, el balón es colocado en la yarda dos del oponente, y la ofensiva tiene dos opciones: Gol de Campo (3 puntos), tal que frecuentemente es la jugada que decide un partido reñido en sus últimos segundos; generalmente se intentan en las últimas 45 yardas por recorrer en cuarta oportunidad, y el pateador debe patear el balón entre los postes y sobre el travesaño. Safety (2 puntos), es cuando el jugador ofensivo porta el balón y es tacleado detrás de su propia línea de anotación o si la ofensiva comete un castigo detrás de su propia línea de anotación. En [4] se desarrolla el modelado algoritmico, basado en autómatas de pila UEPS y lenguajes libres de contexto, del juego de futbol Americano.

Los vectores con combinaciones de jugadas posibles para el equipo a la ofensiva, sondel tipo (pase-balón-mariscal, pase-balón-corredor, atrapa-balon-corredor, correr-balón, anotación); y para el equipo a la defensiva, son del tipo (interceptar-balón, derribar-adversario, atrapar-adversario).

3. Equilibrio de Nash para elegir estrategias

Para aplicar el EN en el dilema del prisionero, el beisbol y el futbol Americano es necesario entoces identificar, el conjunto de estrategias de cada jugador, el espacio vectorial de perfiles de estrategia y las funciones de utilidad. A partir de estos elementos es viable la representación de cada juego en forma normal y por tanto aplicar el equilibrio de Nash y la eficiencia de Pareto.

Teniendo en mente el EN, analicemos el perfil (callar, callar). Si p_1 prevé que p_2 juega callar, tiene como alternativas afectarlo (y beneficiarse solo) o solidarizarse. Si p_1 prefiere desviarse al perfil (confesar, callar) obtiene el pago superior $u_1(confesar, callar) = 10 > u_1(callar, callar) = 5$. Este argumento también aplica a p_2 por simetría del juego. Por tanto el perfil de estrategia (callar, callar) no es un EN: cualquiera de los prisionero puede desviar su estrategia y obtener un mayor beneficio, y es, por el contrario, un peril de riesgo. Si bien, es el óptimo para fines de cooperación en los prisioneros si lo desearan.

Si tenemos $u_2(confesar, confesar) = 5 > u_2(confesar, callar) = 0$, (confesar, callar), bajo u_2, tampoco es un EN. El caso (callar, confesar) es análogo al anterior intercambiando la posición de los prisioneros. El caso (confesar, confesar) si es un perfil de EN, ya que ninguno de los prisioneros tiene el incentivo para desviarse, unilateralmente, de su estrategia en dicho perfil, pues perdería utilidad en relación al perfil (confesar, confesar), dado que $u_1(callar, confesar) = 0 < u_1(confesar, confesar) = 1$, así como, $u_2(confesar, callar) = 0 < u_2(confesar, confesar) = 1$.

El perfil de estrategia (confesar, confesar) es de EN, pues ningún prisionero *racional* tiene el incentivo de desviar su estrategia en este perfil. En general, la desviación de un perfil de estrategia se realiza tal que, dado un perfil, se cambia cada estrategia de un jugador y se fijan las estrategias de los demás. Si se observa que algún jugador j, al desviar su estrategia, obtiene mayor beneficio según su función u_j, el perfil dado se descarta por ser un perfil dominado. Un perfil dominado es aquel en el que par alguna desviación de un jugador, el valor de beneficio de la desviación es mayor al del perfil dado.

3.1. Equilibrio de Nash para beisbol

El algoritmo para encontrar los perfiles de estrategias que sean de EN consta de los siguientes pasos: 1) Aplicar la función de rentabilidad de cada jugador a cada perfil del juego. Esto proceso genera una matriz de rentabilidad para cada jugador. 2) Para cada perfil de estrategia se

realizan las desviaciones cambiando las estrategias, respectivamente, para cada jugador. Si la evaluación en alguno de los perfiles de estrategia resultantes de la desviación es mejor valorado, el perfil de estrategia original es descartado al ser un perfil dominado. 3) Los perfiles que no hayan sido dominados ni descartado, son los que cumplen el EN, y se utilizan como las opciones estrategias de juegos. Aplicar esas estrategias conlleva una respuesta de menor riesgo a las estrategias de los demás jugadores.

En [1] se utiliza el equilibrio de Nash para identificar las estrategias de equipo en tales partidos, analizando el impacto positivo al utilizarlo: el equipo aumenta las probabilidades de exito frente a un equipo que juegue sin un método para elegir estrategias. En [3] se analiza el uso de estrategias de cooperación y de no cooperación durante partidos de beisbol, en ambos casos a partir de utilizar el equilibrio de Nash. En particular al identificar las jugadas de sacrificio como estrategia, aumenta la probabilidad de aplicar una estrategia ganadora. Aplicar las jugadas de sacrificio como estrategias es recomendable si: el equipo se encuentra en las ultimas entradas y la diferencia en el marcador es muy ajustada. Esto ocurre porque los llamados *flies* de sacrificio son golpes de bat sencillos, con mayor probabilidad de ocurrencia como lo constatan las estadísticas de juego. Un *homerun,* vistoso y muy eficaz, es poco probable que ocurra y conlleva alto riesgo como solución factible.

3.2. Equilibrio de Nash para el futbol Americano

Utilizando la misma metodología que para el análisis de selección de estrateigas en beisbol, en [8] se desarrolla el uso de EN y el concepto de eficiencia de Pareto para modelar las estrategias de juego en partidos de futbol Americano. Se identifican los perfiles de estrategia para jugador defensivo, ofensivo y especial, tales que satisfacen EN y EP. Los perfiles EN integran jugadas de mayor frecuencia de ocurrencia, jugadas más factibles, que al equipo le fortalecen sus alternativas de éxito. Complementario, los perfiles de EP integran jugadas de baja frecuencia de ocurrencia (una anotación por pase largo), y son menos factibles. Nuestro simulador, en la final del supertazón 2015, hubiera indicado como perfil EP la jugada final, de pase, practicada por los *Sea Hawks* y como EN una jugada poco vistosa pero más probable: acarrear el balón y proteger a quién lo llevaba.

4. Discusión

El beisbol y el futbol Americano son juegos estratégicos. El EN y la EP habilitan el encontrar estrategias para aumentar la probabilidad de

ganar un partido. Originalmente, J. Nash define su equilibrio para juegos no cooperativos, entre jugadores que compiten por el mismo objetivo. Nosotros lo aplicamos para caracterizar la cooperación entre jugadores de un mismo equipo. Cuando un equipo utiliza el EN bajo diversas circunstancias, fortalece sus alternativas de éxito. Es así porque el EN, por principio de definición, integra jugadas de menor riesgo, y por tanto de mayor probabilidad de ocurrencia, en contrapunto con la EP. El identificar las probabilidades de ocurrencia tiene un alto impacto en el diseño de las estrategias. Hay juegos con múltiples soluciones de EN, y tal multiplicidad ofrece elegir la que mejor resuelva un problema. En [1] y [6] se usan autómatas de pila último-en-entrar primero-en-salir (UEPS) y lenguajes libres de contexto, para la simulación de partidos de beisbol y futbol Americano. Este modelado, inusual en la literatura, da base sólida para el análisis preciso de tales juegos y su posterior elección de estrategias.

5. Conclusiones

En el beisbol y el futbol Americano, una buena selección de estrategias aumenta la posibilidad de triunfo. Coordinar jugadas entre más de una decena de jugadores, siendo muy complejo al involucra muy diversas jugadas y eventos inesperados, es viable y eficiente utilizando el equilibrio de Nash y la eficiencia de Pareto. Considerar la frecuencia de ocurrencia de las jugadas, es esencial para obtener simulaciones que correspondan a lo que ocurre en partidos reales. Usualmente, en los perfiles de estrategia que satisfacen el equilibrio de Nash se integran jugadas con mayor frecuencia de ocurrencia, y en los perfiles de estrategia que satisfacen la eficiencia de Pareto se incorporan las mejores jugadas, pero con menor probabilidad de ocurrencia.

Referencias

1. M. Alvarado, A.Y. Rendon, *Nash equilibrium for collective strategic reasoning*, Expert Syst. Appl.; 2012.
2. M. Alvarado, A. Yee, *Baseball sacrifice play: towards the Nash Equilibrium based strategies*, International Conference of 21st Game Theory Festival, Stony Brook University; 2010.
3. Alvarado, M., Yee Rendon, A., & Cocho, G., *Simulation of baseball gaming by cooperation and non-cooperation strategies*, Computación y Sistemas; 2014, 693-708 pp.
4. Alvarado, M., Yee, A., & Fernandez, J., *Simulation of American football gaming.* Advances in Sport Science and Computer Science; 2014, 57, 227 pp.
5. Nash, J., *Non cooperative games.* The annals of Mathematics, Second Series 54 (2); 1951, 286 - 295 pp.

6. Yee, A., Rodríguez, R., & Alvarado, M. *Analysis of Strategies in American Football Using Nash Equilibrium.* In Artificial Intelligence: Methodology, Systems, and Applications; 2014, 286-294 pp.
7. Yee, A., & Alvarado, M. *Methodology for the modeling of multi-player games*, Proceedings of the 18th International Conference on Computers (part of CSCC '14); 2014.
8. Yee, A., *Selection of strategies in complex games*, Tesis doctoral, Departamento de Computación, CINVESTAV, IPN; Mayo de 2014.

Reconocimiento Bimodal de Emociones en un Ambiente Inteligente de Programación

María Lucía Barrón Estrada, Margarita Aranda Ortega

Instituto Tecnológico de Culiacán
lbarron@itculiacan.edu.mx

Recibido 15 de Octubre de 2015, Aceptado 30 de Octubre de 2015,
Versión final 15 de Noviembre de 2015

Resumen El presente trabajo describe los resultados obtenidos de los Estados afectivos experimentados por un grupo de estudiantes, en interacción con un Sistema Tutor Inteligente (STI) orientado a la generación de código para la solución de problemas en el Lenguaje Java. La captura de los Estados afectivos se realizó a través de la diadema Emotiv EPOC, integrada al STI junto con la captura de imágenes del usuario a través de WebCam. Los datos se graficaron a través de series de tiempo para preservar el orden temporal y por intervalos de acuerdo a las etapas del Método 6 D's. Los resultados más relevantes consisten en el registro y la identificación de los Estados afectivos que prevalecen en cada una de las etapas del Método 6 D's, durante la generación de código para la solución de problemas presentados por el STI y la captura de expresiones faciales al desarrollar la actividad.

Palabras Clave: Sistema Tutor Inteligente, Método 6D's, Bimodal

1. Introducción

En las ciencias computacionales, la enseñanza y aprendizaje de un lenguaje de programación es un factor muy importante que determina el éxito o fracaso de los estudiantes en la carrera para instituciones de educación superior y universidades de cualquier país. Algunos factores generales que contribuyen a este éxito o fracaso son las habilidades matemáticas

del estudiante, la motivación del mismo, los métodos de enseñanza empleados por el profesor y la naturaleza misma del arte de la programación (Haungs, Clark et al. 2012, Ala-Mutka, 2012; Bennedsen & Caspersen, 2007; Gerdes, Heeren, & Jeuring, 2009; Jenkins, 2002; Matthíasdóttir, 2006). Por otra parte y siguiendo una forma diferente de analizar el problema, este trabajo se enfoca en estudiar y analizar las emociones que los estudiantes de ciencias computacionales experimentan al diseñar y desarrollar programas que resuelven un problema en el lenguaje de programación Java. Para llevar a cabo este estudio, se utilizó una diadema especial Emotiv/EPOC (`https://emotiv.com/`) para lectura de señales cerebrales, la cual cuenta con un software que permite leer los estados afectivos que experimenta una persona al estar realizando una actividad mental. La motivación que lleva a realizar este trabajo de investigación, es la de contribuir en el mejoramiento de tecnologías aplicadas al proceso de enseñanza-aprendizaje, en flexibilidad y adaptabilidad a las necesidades del usuario. La idea principal es identificar características que sean de utilidad, para el reconocimiento de emociones en las nuevas tecnologías, que apoyan la solución de problemas en actividades de programación. Esta investigación se distingue por la integración de dos tipos de captura de datos, las señales electroencefalográficas (EEG) [1] o actividad cerebral que contiene los estados afectivos y la captura de imagen sobre las expresiones faciales del usuario al desarrollar la actividad de programación en el lenguaje Java dentro de un ambiente de aprendizaje. La contribución principal del trabajo presentado en este artículo es, la implementación de un ambiente inteligente de programación con identificación de emociones por medio de señales del cerebro y expresiones faciales, en donde se identificarán las diferentes emociones que un programador siente durante cada una de las diferentes fases de desarrollo en la construcción de un programa.

1.1. Método Propuesto

La motivación de contribuir en el mejoramiento de tecnologías aplicadas al proceso de enseñanza-aprendizaje en flexibilidad y adaptabilidad al usuario, conlleva a proponer un método que integre las señales EEG dentro de un Sistema Tutor Inteligente. Con lo anterior, se pretende conocer el comportamiento de los estados afectivos al desarrollar actividades de programación en el lenguaje Java, para identificar las emociones que están presentes en los estados afectivos, tomando como referencia a

[1] EEG del inglés Electroencephalography término acuñado por el fisiólogo alemán Hans Berger (1873-1941).

(Gonzalez-Sanchez, Chavez-Echeagaray et al. 2011, Dŕnello and Graesser 2013). La Figura 1 muestra el modelo general de la propuesta, donde se observa que un usuario (sujeto) frente a un equipo de cómputo, realiza actividades de programación dentro de un STI al mismo tiempo que es capturada la actividad cerebral con el dispositivo Emotiv EPOC y la captura de imagen con cámara Web, siendo registrada cada 5 segundos. La actividad cerebral refleja el comportamiento de los estados afectivos, mientras que la imagen captura las expresiones faciales que el sujeto realiza al desarrollar múltiples ejercicios, seleccionados bajo el método de las 6 D's (Cueto). El modelo se orienta a integrar los estados afectivos de un sujeto en particular, dentro de un STI enfocado al desarrollo de código en el lenguaje Java. Por otra parte, se realiza el análisis del comportamiento de los estados afectivos al desarrollar de los ejercicios propuestos para experimento. Para llevar a cabo la tarea antes mencionada, es necesario contar con ejemplos de los datos que proporciona el dispositivo, es decir, de los estados afectivos grabados mientras los sujetos realizan la solución del ejercicio.

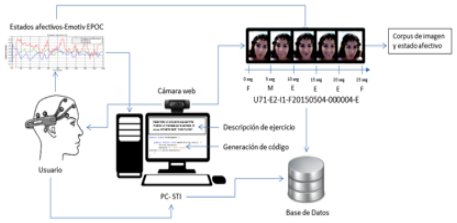


Figura 1. Modelo general del método propuesto

2. Metodología

La estrategia metodológica empleada es de tipo experimental para el reconocimiento de emociones en la cual se abordaron las siguientes etapas:

- Pre-experimento: La intervención no se realizó a propósito de la investigación, sino que obedece a la necesidad de conocer qué tipo de datos proporciona el dispositivo Emotiv EPOC al incluir las instancias de las Suites Expresiva, Afectiva y Cognitiva. Se ha aplicado un ejercicio para identificar el tiempo que requiere un sujeto en el desarrollo de la actividad y la definición de un protocolo.

- Cuasi-experimento: La intervención se realizó con el propósito de realizar la captura de los estados afectivos que presenta un grupo que consta de 8 sujetos, en el cual se han realizado dos mediciones las cuales se identifican por los ejercicios E1 y E2, para analizar el comportamiento de cada Estado afectivo.
- Integración de aplicaciones: En esta etapa de realiza el desarrollo de un STI enfocado al área de programación en el lenguaje Java. Se integran los estados afectivos al STI y se anexan los ejercicios que serán mostrados a los sujetos de manera aleatoria en una base de datos.
- Experimento final: La intervención se realiza con el propósito de comprobar que el sistema realiza la captura de los Estados afectivos y la captura de imagen al mismo tiempo, mientras el sujeto desarrolla ejercicios de programación presentados por el STI. El grupo de control ha sido de 3 sujetos, de los cuales se ha generado un Corpus de datos que integra el id, ejercicio, fecha, tiempo y estado afectivo registrado a los 5 segundos. Este corpus de imagen con estado afectivo pasa a otro tipo de procesamiento con el desarrollo de otra investigación.

2.1. Adquisición de la Actividad Cerebral

En la Figura 2, se muestran los elementos empleados y el área de trabajo donde se lleva a cabo la adquisición de señales del cerebro en cada una de las etapas del experimento. El área corresponde a un espacio dentro del edificio de posgrado del Instituto Tecnológico de Culiacán.

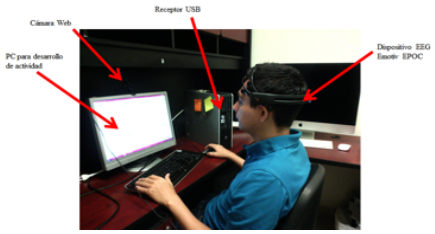


Figura 2. Elementos empleados en la captura de datos

La suite afectiva (Affective Suite) del software de Emotiv/EPOC, visualiza la evolución en tiempo real de los cambios en los estados afectivos experimentados por el usuario. Cuenta con dos graficas que visualizan la evolución de las señales, por defecto la primera muestra un lapso de 30

segundos y la segunda visualiza los últimos 5 minutos. Las señales que se visualizan corresponden a los siguientes estados afectivos que se muestran en la Tabla 1.

Tabla 1. Estados afectivos Detectados por Emotiv/EPOC

Compromiso/Aburrimiento (Engagement/Boredom)
Frustración (Frustration)
Meditación (Meditation)
Emoción instantánea (Instantaneous Excitement)
Emoción a largo plazo (Long-Term Excitement)

3. Integración de Aplicaciones

Tomando en cuenta las investigaciones recientes en el estado del arte sobre STI's con enfoque de programación en busca de mejorar el aprendizaje, determinar el estado afectivo predominante al desarrollar las actividades y la captura de expresiones faciales que presenta el sujeto, se realiza lo siguiente: Se integran las instancias de los estados afectivos de Emotiv Epoc en tiempo real, librerías de Open CV y Java CV para el uso de la cámara Web, el listado de ejercicios, técnicas de inteligencia artificial y una serie de funciones que permiten realizar los registro en el STI de prueba, en conjunto con la base de datos de PostgreSQL versión 1.20.0. La finalidad de la integración es automatizar el registro de los datos que se requieren, para realizar el análisis a través de la generación de un Corpus que contenga el estado afectivo y las expresiones faciales presentadas en diferentes ejercicios. A continuación se presentan los elementos del sistema.

4. Arquitectura

El Sistema Tutor Inteligente es un sistema de prueba con enfoque de programación, que permite desarrollar código en el ambiente Java, identificar errores de compilación, registrar el nivel de progreso del usuario y realizar la captura de Imagen-Estado afectivo para la generación del Corpus. El sistema contiene una arquitectura de capas relajadas, la cual se muestra en la Figura 3. El Módulo de Reconocimiento Facial Afectivo corresponde a un estudio más profundo realizado por separado, por lo tanto no se incluye en este documento.

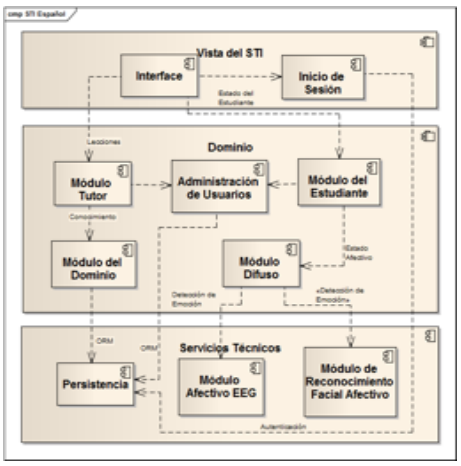


Figura 3. Arquitectura del ITS para Programación Java

4.1. Capa de visualización (Vista del STI)

Esta capa realiza la presentación del tutor donde el usuario interactúa. Se integra por dos componentes que son la Interface y el Inicio de sesión. La Interface corresponde a la presentación y bienvenida del usuario, así mismo permite la comunicación con el componente de Inicio de Sesión, que permite el registro del usuario que por primera vez utiliza el sistema.

4.2. Capa de dominio

En esta capa se implementa la lógica del Sistema Tutor Inteligente. La información almacenada corresponde a las acciones del STI y a los estados afectivos generados por el usuario. Esta capa se encuentra implementada en lenguaje JAVA. El STI permite al experto introducir ejercicios o problemas en los niveles de principiante, intermedio o avanzado, mismos que son presentados al usuario conforme avanza en la solución. Esto permite al usuario incrementar el grado de complejidad. Un componente neuro-difuso aplica el conjunto de reglas, que permite al usuario cambiar de ejercicio en caso de presentar errores de compilación o por el deseo de cambiar el ejercicio. El STI realiza el registro de tiempo y el número de intentos para desarrollar la solución del ejercicio. Los elementos que interactúan con el STI y la misma configuración del STI se encuentran implementados en el IDE de Java dentro del ambiente de NetBeans IDE 7.3.1.

4.3. Capa de servicios técnicos

En esta capa se encuentran todos los servicios de bajo nivel que corresponde a los módulos de comunicación con los sistemas afectivos (Reconocimiento facial afectivo y EEG afectivo), así como también el acceso a la base de datos.

5. Interface Principal del ITS

La figura 4 muestra la interface principal del ITS. La interface consta de 5 áreas principales, que se describen a continuación:

a) Un usuario. En esta sección se muestra el nombre del estudiante.
b) Descripción del ejercicio. En esta sección se muestra la descripción del ejercicio a resolver por el alumno.
c) Sección del código. Consta de 2 pestañas: una que muestra el código que el estudiante está escribiendo para resolver el ejercicio y otra (depuración), que muestra los errores producidos durante la compilación.
d) Aspectos cognitivos. Esta sección muestra los indicadores utilizados para evaluar el aspecto cognitivo del estudiante: tiempo empleado en la resolución del ejercicio, la cantidad de errores y el nivel actual de la dificultad en el ejercicio. Hay 3 niveles de dificultad para clasificar los ejercicios: fácil, medio y difícil.
e) Aspectos afectivos. En esta sección se muestra la emoción y la valencia identificado por los reconocedores.

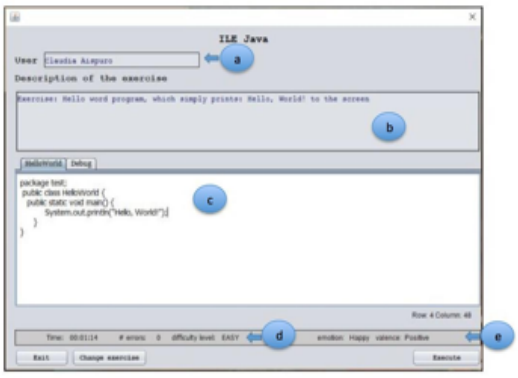


Figura 4. Interface principal del ITS

5.1. Componentes

Los componentes que intervienen en el STI para el reconocimiento de emociones son la diadema Emotiv EPOC, que captura los Estados afectivos y la cámara Web que captura la imagen y la envía a un registro que genera el Corpus de Imagen-Estado afectivo. El usuario interactúa con el STI en la lectura del problema, análisis, escritura, compilación, ejecución y corrección de código. En la interacción se presenta la GUI de la Figura 5, que permite al usuario interactuar con el STI. En primer lugar el STI presenta un mensaje de bienvenida al usuario, el cual realiza el registro o bien la entrada al sistema si ya existe, posteriormente el sistema solicita la activación de la cámara web y presenta la pantalla principal donde el usuario realizará la codificación que da solución al problema presentado en la parte superior. Cada vez que el usuario concluye satisfactoriamente un ejercicio, el sistema le presenta un mensaje indicando que se ha resuelto correctamente. Durante el desarrollo de la actividad se realiza el registro de los estados afectivos y la captura de imagen a través de la cámara Web y se genera un Corpus de Imagen-Estado afectivo.

Figura 5. Navegación por el Sistema

6. Resultados de Experimentos

Para la etapa de captura de los estados afectivos en los sujetos participantes dentro del experimento, se contó con un área específica dentro de las instalaciones del posgrado del Instituto Tecnológico de Culiacán. Se contó con la participación de personas voluntarias, siendo estos alumnos de posgrado con nivel avanzado de programación. Físicamente se

trabajó con 8 sujetos (6 hombres y 2 mujeres) (diestros y zurdos) de diferente edad. Como material de estímulo se aplicaron dos ejercicios básicos seleccionados previamente, desde el clásico Hola Mundo y otro con aplicación de fórmulas básicas de perímetro y área de un cuerpo. Se obtuvieron unidades de análisis de los 8 sujetos, de los cuales después de saber que no existe relación entre estos y que son casi independientes, de determinó analizar los estados afectivos generados por Emotiv Engagement/Boredom (E/B), Meditation (M) y Frustation (F). A nivel exploratorio se han graficado las series de datos para observar el comportamiento de cada estado afectivo, por sujeto y ejercicio. La Figura 6, muestra los estados afectivos durante el desarrollo del programa, indicando en color azul Engagement/Boredom (E/B), el color verde para Meditation (M) y en color rojo la Frustation (F). El nivel de amplitud registrado en los estados afectivos va de 0.0 a 1.0, donde los valores intermedios son los diferentes grados de amplitud que alcanza cada estado afectivo del sujeto al realizar la actividad indicada.

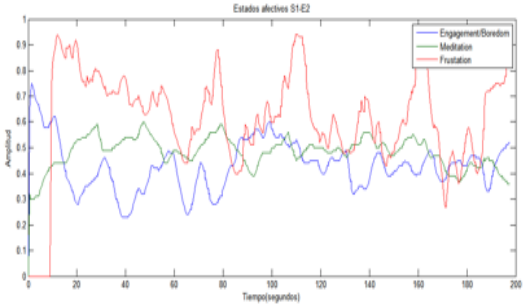


Figura 6. Resultado de Experimento con Emotiv

El ejemplo de la Figura 6 se muestra que el tiempo en desarrollar uno de los ejercicios fue de 197.68 segundos (casi 200 seg.) = 3.394 minutos. También podemos observar que la frustración de los estudiantes en promedio se dispara al inicio del problema y después tiende a bajar y a subir conforme los estudiantes al estar programando pasan por diferentes estados afectivos, de acuerdo a como ellos van enfrentando errores en su solución y como los van resolviendo. También la curva representando el estado afectivo de enganchado nos muestra un ligero incremento a la mitad del problema mientras que la curva de meditación se sostiene regularmente sin muchas grandes variaciones. Un punto importante es que al final la curva de frustración se dispara hacia arriba por la razón de que

la mayoría de los estudiantes no pudieron terminar bien el ejercicio. En otro experimento se realiza la prueba para 6 diferentes ejercicios, donde el STI es capaz de realizar el registro de los usuarios en la base de datos. La Tabla 2 muestra los tiempo asignados a cada etapa con el Método de las 6 D's.

Tabla 2. Intervalos de tiempo asignados a los registros del STI.

Usuarios	No. Reg.	Tiempo (Seg.)	Intervalos de tiempo en Etapas del Método 6D's									
			Etapa1		Etapa2		Etapa3		Etapa4		Etapa5	
U71-E2	2288	398.48	0	79.7	79.7	159.39	159.39	239.09	239.09	318.79	318.79	398.48
U72-E1	135	21.1	1	4.22	4.22	8.44		12.66	12.66	16.88	16.88	21.1
U72-E2	5298	668.58	3	133.72	133.72	267.43	267.43	401.15	401.15	534.87	534.87	668.58
U72-E3	1304	204.19	4	40.84	40.84	81.68	81.68	122.52	122.52	163.36	163.36	204.19
U72-E4	1297	180.99	5	36.2	36.2	72.4	72.4	108.6	108.6	144.8	144.8	180.99
U74-E6	2265	330.27	7	66.05	66.06	132.11	132.11	198.16	198.17	264.22	264.22	330.27
U75-E4	4467	707.51	8	141.5	141.5	283.01	283.01	424.51	424.51	566.01	566.01	707.51
U76-E1	2506	304.6	9	60.92	60.92	121.84	121.84	182.76	182.76	243.68	243.68	304.6

Realizada la asignación del intervalo de tiempo se obtienen las medias del comportamiento de los estados afectivos en cada uno de los ejercicios a nivel global de los usuarios. A continuación en la figura 7 se presenta el comportamiento para el estado afectivo *Enganchado*. Como se aprecia en la Figura dependiendo del ejercicio es el comportamiento de las emociones en los estudiantes pues por ejemplo en el ejercicio 2 (E2) en la etapa 3 el enganchamiento de los estudiantes decae, mientras en el ejercicio 1 (E1) en la etapa 4 el enganchamiento en promedio de los estudiantes se incrementa.

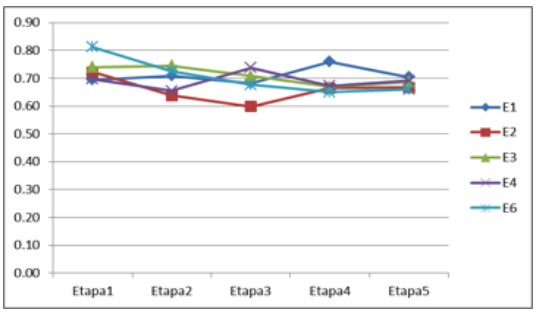


Figura 7. Comportamiento de E/B en los seis ejercicios

7. Conclusiones y Trabajo a Futuro

Como conclusión principal podemos decir que el estado afectivo de los estudiantes varía dependiendo del tipo de ejercicio y de la etapa en que se encuentre el estudiante de acuerdo al método de las 6 D's. Con esta información un profesor puede tomar nota y realizar cambios para por ejemplo tratar de mantener enganchado a los estudiantes cuando estos están resolviendo un ejercicio. Actualmente, el sistema o ITS ya registra los estados afectivos por medio de la diadema Emotiv y ya se registran los estados afectivos del reconocimiento usando expresiones faciales. Como trabajo a futuro inmediato está el realizar una integración de ambos estados afectivos para comprobar el éxito del segundo reconocedor y también para producir un corpus de forma automática con una buena tasa de éxito en el reconocimiento

Referencias

1. Haungs, M., et al., *Improving first-year success and retention through interest-based CS0 courses.* , Proceedings of the 43rd ACM technical symposium on Computer Science Education. Raleigh, North Carolina, USA, ACM: 589-594.
2. Ala-Mutka, K. *Problems in learning and teaching programming*, Codewitz Needs Analysis.
3. Bennedsen, J., & Caspersen, M. E. *Failure rates in introductory programming.*, ACM SIGCSE Bulletin, 39(2), 32-36.
4. Gerdes, A., Heeren, B., & Jeuring, J., *Constructing Strategies for Programming.* . Paper presented at the CSEDU (1).
5. Jenkins, T. *On the difficulty of learning to program.*. Paper presented at the Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences.
6. Matthíasdóttir, Á. *How to teach programming languages to novice students?*. Lecturing or not. Paper presented at the International Conference on Computer Systems and Technologies-CompSysTech.
7. Gonzalez-Sanchez, J., et al. *ABE: An Agent-Based Software Architecture for a Multimodal Emotion Recognition Framework. Software Architecture (WICSA)*, 2011 9th Working IEEE/IFIP Conference on.
8. D'mello, S. and A. Graesser. *AutoTutor and affective autotutor: Learning by talking with cognitively and emotionally intelligent computers that talk back.*, ACM Trans. Interact. Intell. Syst. 2(4): 1-39.
9. Cueto, J. J. F. *Método de las 6'D. UML - Pseudocódigo - Java. (Enfoque algorítmico)*, Método de las 6'D. Universidad de San Martín de Porres.

IZTATL
COMPUTACIÓN

Comité Revisor

Dr. Alberto Portilla Flores

Dr. Brian Manuel González Contreras

Dr. Carlos Sánchez López

Dr. Francisco Javier Albores Velasco

Dra. Leticia Flores Pulido

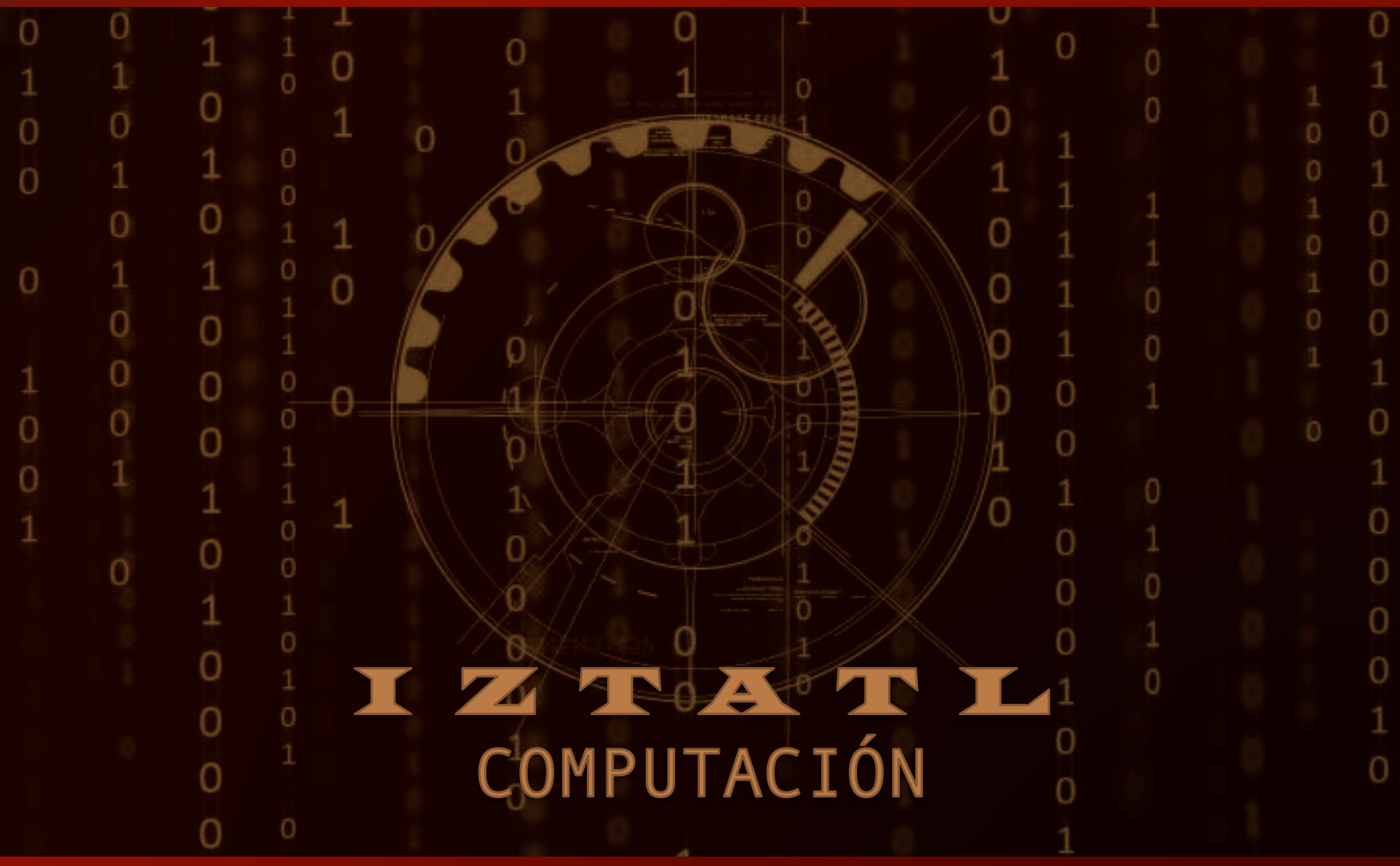
Dra. Marva Angélica Mora Lumbreras

M.C. Carolina Rocío Sánchez Pérez

M.C. María del Rocio Ochoa Montiel

M.I.A. Norma Sánchez Sánchez

M.C. Patrick Hernández Cuamatzi


IZTATL
COMPUTACIÓN