Preprocesado de flujos de datos para aprendizaje automático mediante reglas CEP

Aurora Ramírez, Nathalie Moreno, Manuel F. Bertoa y Antonio Vallecillo

Dpto. Lenguajes y Ciencias de la Computación, Universidad de Málaga
{aramirez,moreno,bertoa,av}@lcc.uma.es

Resumen El procesamiento de flujos de información constituye un área de gran relevancia dentro de la gestión de datos, pues sus métodos deben ser ágiles y eficientes para soportar el volumen y la velocidad con la que los datos se generan actualmente. Las técnicas de minería de datos han necesitado adaptarse a estas circunstancias, que no solo afectan al modo en el que se aprende de los datos, sino también a la preparación de los mismos. En este contexto, los sistemas de procesamiento de eventos complejos (CEP) pueden facilitar el tratamiento de los datos en tiempo real. Este trabajo propone abordar el preprocesamiento de flujos de datos mediante CEP. El estudio experimental revela que los datos, convenientemente transformados y enriquecidos con información temporal, mejoran la predicción de los algoritmos de aprendizaje automático.

Keywords: Flujos de datos · procesamiento de eventos complejos · preprocesado · aprendizaje automático.

1. Introducción

La velocidad con la que los sistemas informáticos generan datos hoy en día, unido al creciente interés en métodos que los analicen automáticamente, han hecho que exista una gran demanda de tecnologías eficientes para el procesamiento de flujos de datos. El hecho de que los datos se generen a gran velocidad y que las decisiones deban tomarse en tiempo real supone una serie de retos tanto a nivel de almacenamiento como de las técnicas de descubrimiento de conocimiento [14].

En la era del *big data*, donde los datos pueden llegar a ser ilimitados, los sistemas de procesamiento de eventos complejos (*complex event processing*, CEP) abren nuevas posibilidades para manejarlos eficientemente [5]. Este tipo de sistemas permite procesar cada dato en el momento que llega, tomar decisiones y propagarlas rápidamente [4]. Para ello es necesario que el experto defina de antemano las situaciones de interés a detectar, las cuales son expresadas en forma de reglas y patrones. CEP tiene gran aplicación en sistemas de monitorización de sensores [3] o en aplicaciones basadas en el internet de las cosas [11], entre otras.

Sin embargo, cuando el objetivo es extraer conocimiento útil y previamente desconocido, son las técnicas de minería de datos (*data mining*, DM) las que que entran en escena [6]. Algo que no varía por el hecho de trabajar con datos estáticos o en tiempo real es la necesidad de preparar los datos para la fase de aprendizaje. De hecho, realizar

un buen preprocesado de los datos puede suponer una mejora considerable en los resultados, si bien puede llegar a ser una tarea laboriosa y costosa [16]. En el caso de los flujos de datos, el preprocesado debe ser además ligero y lo más automático posible [6].

Por ello en este artículo se explora la posibilidad de utilizar CEP como mecanismo de preprocesado para flujos de datos en el contexto de DM. Dado que muchas de las transformaciones y filtrados que se le deben realizar a los datos son establecidos a priori en función del dominio de aplicación, un experto podría expresarlas en forma de reglas e integrarlas en un sistema CEP. Además, el alcance de las reglas puede acotarse mediante ventanas, sobre las que realizar cálculos aritméticos. Esta funcionalidad va a permitir la generación de características con información temporal, lo cual puede ayudar a realizar predicciones más ajustadas en un entorno de aprendizaje incremental.

Este trabajo supone un primer estudio donde se plantean y analizan las posibles aplicaciones de CEP al preprocesado de flujos de datos. Además, se presenta una implementación Java que integra un sistema CEP con una librería de aprendizaje automático. Para validar la propuesta, se han desarrollado dos casos de estudio en diferentes dominios, cuyos resultados revelan que el preprocesamiento de los datos realizado mediante reglas CEP ayuda a mejorar notablemente la predicción de los algoritmos.

La sección 2 introduce conceptos previos relacionados con el aprendizaje automático y CEP. La sección 3 describe los posibles usos de CEP para el preprocesado, así como la implementación de la propuesta. Los casos de estudio se desarrollan en la sección 4. Finalmente, la sección 5 recoge las conclusiones y el trabajo futuro.

2. Conceptos previos

2.1. Aprendizaje automático a partir de flujos de datos

Dentro del denominado proceso de extracción de conocimiento desde bases de datos (*Knowledge Discovery from Databases*, KDD), la preparación de los datos y la minería de datos son dos fases de gran importancia. La primera de ellas consiste en adecuar el formato o contenido de los datos provenientes de una o más fuentes, con el objetivo de facilitar su posterior procesamiento [7]. Durante esta fase, por ejemplo, se elimina ruido o se tratan inconsistencias y valores perdidos. En la tabla 1 se recogen las principales actividades que se llevan a cabo durante el preprocesado de los datos.

Una vez que los datos han sido preparados, se debe elegir la técnica de DM más apropiada para extraer conocimiento útil. Aquí destacan los métodos de aprendizaje automático, que suelen clasificarse en base al tipo de tarea que se vaya a realizar. Así, los algoritmos de *clasificación* tratan de asignar una categoría predefinida, llamada clase, a una nueva instancia recibida. Por el contrario, los métodos de *regresión* infieren el valor de una variable numérica a partir de otras.

El hecho de que las instancias puedan recibirse de forma continua hace necesario que se realicen ciertas adaptaciones tanto en su preprocesado como en el aprendizaje a partir de ellas. En cuanto al preprocesado, es necesario considerar que tanto la distribución de los datos como sus propiedades pueden cambiar a lo largo del tiempo [13]. Desde el punto de vista del aprendizaje, se debe optar por un enfoque incremental, donde el modelo de decisión aprendido se actualice conforme llegan nuevas instancias [2]. Además, los requisitos de tiempo y memoria se vuelven más exigentes, pues estos métodos se conciben para ser desplegados en sistemas en tiempo real o en escenarios donde

Tabla 1. Categorización de técnicas de preprocesado de datos (adaptada de [7]).

Preparación de datos	
Limpieza	Corrección y filtrado de datos incorrectos.
Transformación	Conversión del formato de los datos.
Normalización	Modificación de los datos para escalar sus valores.
Integración	Unión de datos provenientes de varias fuentes.
Reducción de datos	
Selección de características	Eliminación de características irrelevantes o redundantes.
Selección de instancias	Muestreo de instancias para, por ejemplo, validación.
Discretización	Partición de los valores continuos en intervalos.
Generación de valores	Reemplazo de características y creación de instancias.

la cantidad de datos de entrada y la velocidad con la que se generan hacen imposible su almacenamiento. Cabe mencionar que tanto los algoritmos de preprocesado como los de aprendizaje para flujos de datos que pueden encontrarse en la literatura son, o bien específicos, o bien adaptaciones de métodos utilizados para datos estáticos.

2.2. Procesamiento de eventos complejos

El procesamiento de eventos complejos (*Complex Event Processing*, CEP) consiste en la detección de situaciones de interés en base a reglas y patrones definidos por expertos [9]. Bajo el término CEP se engloban una serie de sistemas y tecnologías capaces de procesar flujos de información, donde cada muestra que llega es tratada como un evento simple. Con el objetivo de inferir la ocurrencia de eventos complejos, el sistema es capaz de filtrar y combinar los eventos simples para determinar qué está ocurriendo [4].

Los eventos y patrones se definen en CEP mediante un lenguaje de procesado de eventos (*Event Processing Language*, EPL), similar a SQL. En general, las reglas que se definen en los sistemas CEP tienen la siguiente estructura:

1. *Selection*: representa el antecedente de la regla, donde se identifica la ocurrencia de los eventos simples o compuestos que disparan la regla.
2. *Matching*: fase en la que se decide si los eventos, o sus atributos, cumplen ciertas condiciones. Para ello se utilizan operadores aritméticos, lógicos o de precedencia.
3. *Production*: consiste en la generación de eventos derivados y el cálculo de sus atributos mediante funciones de agregación u otras definidas por el experto.

```
select *|<atributo(s)>
from <flujo(s)>|<patron(es)>
where <condicion(es)>
```

1. Sintaxis de una regla CEP en lenguaje EPL.

El código 1 muestra cómo se expresa una regla CEP en lenguaje EPL, la cual consta de las siguientes partes: 1) `select`, para especificar los atributos de interés o el evento completo (*); 2) `from`, para indicar uno o más flujos de entrada (o bien patrones sobre ellos); y 3) `where`, para establecer restricciones. Ante la llegada de nuevos eventos al flujo de entrada, el motor CEP analiza todas las reglas registradas en el sistema y,

Tabla 2. Tipos de operadores en CEP (adaptados de [4]).

Selección	Filtrado de eventos en base a restricciones en su contenido.
Proyección	Extracción de partes de la información contenida en el evento.
Lógica	Construcción de expresiones de conjunción, disyunción y negación.
Secuencia	Selección en base a criterios de orden (temporalidad).
Flujo	Manejo de flujos (unir, dividir, ordenar).
Creación	Generación de nuevos flujos e inserción de eventos en ellos.
Aritmético	Cálculo de propiedades en series de eventos.

si se cumplen sus condiciones, las dispara. En la tabla 2 se recogen los principales tipos de operadores que se pueden utilizar en las reglas. Los sistemas CEP tradicionales funcionan bajo un modelo de *publicación-subscripción* [4], donde los subscriptores recogen y procesan los resultados producidos por las reglas.

Un aspecto destacado de CEP es la posibilidad de establecer relaciones temporales entre los eventos mediante el uso de ventanas. Las ventanas pueden ser: *espaciales*, compuestas por un número prefijado de eventos; o *temporales*, definidas en base al tiempo transcurrido en el sistema. Un ejemplo del uso de una ventana espacial sería procesar conjuntamente las últimas n transacciones de un banco, mientras que una ventana temporal sería útil para filtrar todas las transacciones realizadas en la última hora.

La definición manual de las reglas CEP es sin duda la tarea más laboriosa y depende en gran medida del conocimiento del experto. No obstante, una tendencia reciente consiste precisamente en utilizar técnicas de aprendizaje automático para asistir al experto en este proceso. Entre las propuestas existentes cabe destacar iCEP [10], donde la definición de las reglas se divide en varios subproblemas, como identificar los tipos de eventos y atributos más interesantes o determinar el tamaño de la ventana más apropiado. Otra propuesta relevante es autoCEP [12], capaz de aprender secuencias temporales y expresarlas como reglas CEP. Finalmente, el objetivo de Adaptive CEP [8] es actualizar las reglas CEP utilizando técnicas de agrupamiento y modelos probabilísticos. Todas estas propuestas se centran en incrementar las capacidades de CEP con técnicas de DM. Por el contrario, la siguiente sección plantea una nueva sinergia entre ambas áreas, donde son las funcionalidades de CEP las que van a ayudar al proceso KDD.

3. CEP para el preprocesado de flujos de datos

3.1. Aplicaciones de CEP al preprocesado de datos

Basándonos en la categorización de tareas presentada en la sección 2.1 y en los tipos de operadores definidos en la sección 2.2, la tabla 3 recopila las posibilidades que CEP ofrece al preprocesado de datos. Para cada tarea, se enumeran los tipos de operadores que consideramos aplicables, así como ejemplos de las cláusulas y funciones EPL concretas que los implementan. Para ello hemos tomado como referencia la API de Esper[1], una de las implementaciones de CEP más conocidas, y que está desarrollada en Java.

Respecto a la limpieza de los datos, las reglas CEP van a actuar como filtros, dejando pasar únicamente aquellas instancias (eventos en CEP) que cumplan una determinada

[1] Esper v8: `http://www.espertech.com/esper/` (último acceso: 5/6/2019)

Tabla 3. Operadores, cláusulas y funciones útiles para cada tipo de preprocesado.

Preparación de datos		
Tarea	**Tipo de operador**	**Cláusulas y/o funciones**
Limpieza	- Operadores de selección - Operadores lógicos	`any`, `all`, `like`, `some` `distinctOf`, `except`, `exists`
Transformación	- Operadores de selección - Operadores aritméticos	`all`, `any`, `like`, `some` `cast`, `Math.round`, `toDate`
Normalización	- Operadores aritméticos - Operadores lógicos	`avg`, `maxever`, `minever`, `stddev` `>`, `>=`, `<`, `<=`

Integración	- Operadores de flujo
	- Operadores de secuencia
	- Operadores de creación
		`join`, `order by`, `limit`
		`first`, `last`, `prev`, `take`, `->`
		`insert into`, `output`

Reducción de datos		
Tarea	**Tipo de operador**	**Cláusulas y/o funciones**
Selección de características	- Operadores de proyección	`select`<*atributo*>

Selección de instancias	- Operadores de selección
	- Operadores lógicos
	- Operadores de flujo
	- Operadores de secuencia
		`any`, `all`, `like`, `some`
		`distinctOf`, `except`
		`group by`
		`first`, `last`, `prev`, `take`, `->`

Discretización	- Operadores de selección - Operadores lógicos	`between`, `in`, `not in` `>`, `>=`, `<`, `<=`

Generación de valores	- Operadores aritméticos
	- Operadores de creación
	- Operadores de secuencia
		`avg`, `min`, `max`, `count`, `sum`
		`insert into`
		`leastFrequent`, `mostFrequent`

condición. Para ello se debe utilizar la cláusula `where`, acompañada de expresiones lógicas sobre los atributos. Aparte de los operadores lógicos habituales ($>$, $<$, $\neq$), CEP proporciona funciones como `exists`, que permitiría la detección de valores perdidos. Las comparaciones también se pueden establecer en base a subconsultas, utilizando para ello operadores como `any` o `some`, seguidos de una expresión que devuelva los valores concretos con los que comparar. Funciones como `distinctOf` y `except` van a servir para filtrar ruido, pues permiten indicar qué condiciones debe tener un atributo para que el evento siga siendo considerado en el flujo. Para la transformación de valores, los operadores de selección indicados en la tabla 3 se basan en expresiones regulares. Por tanto, son útiles para identificar, de forma flexible, instancias con valores concretos que se quieran alterar. El operador aritmético `cast` permite convertir entre tipos de datos, incluidas fechas con distinto formato. En el caso de utilizar Esper como motor CEP, las funciones del paquete *Math* de Java, como `round`, pueden incluirse en las reglas. Para la normalización de valores, funciones para el cálculo de la media (`avg`) y la desviación estándar (`stddev`) están también disponibles en CEP. En cuanto al cómputo de máximos y mínimos con los que acotar, la implementación de Esper distingue tres funciones para obtener el máximo (equivalente para el mínimo): `max`, que devuelve el valor máximo en el ámbito de la ventana; `fmax`, que permite incorporar un filtro; y `maxever`, que devuelve el máximo histórico.

La posibilidad de trabajar con múltiples flujos de entrada es una característica de CEP especialmente relevante para abordar la integración de los datos. Una primera forma consiste en seleccionar múltiples flujos de entrada (utilizando `from`) y establecer restricciones comunes sobre ellos. De esta forma se podría, por ejemplo, unir en un mismo flujo las transacciones de un cliente recibidas a través de dos medios de pago diferentes. Una forma equivalente consiste en utilizar la cláusula `join`. Aparte, otras funciones interesantes permiten ordenar o limitar los eventos a la salida (combinando `output` con `order by` o `limit`, respectivamente), o crear nuevos eventos (`insert into`), posiblemente en un flujo diferente. Finalmente, si se quiere acceder a eventos concretos de una ventana se pueden utilizar funciones como `first`, `last`, etc., mientras que el operador de precedencia `followed by` (`->`) permite detectar si un evento sucede antes que otro. A modo de ejemplo, esto sería útil para detectar secuencias de compras en el contexto de un sistema de recomendación.

Centrándonos en las tareas de reducción de datos, la cláusula `select` nos permite extraer el subconjunto de características de interés. Cabe señalar que, al igual que en SQL, las cláusulas `select` pueden anidarse, lo cual permite crear expresiones más complejas. Para la selección de instancias existen más alternativas, ya que la elección se puede hacer en base a su contenido, a la clase a la que pertenece (si es conocida) o a su orden de llegada al flujo. En primer lugar, el uso de operadores lógicos en la cláusula `where` ofrece la posibilidad de establecer restricciones sobre los valores que toman los atributos. En aquellas situaciones en las que los datos contengan atributos categóricos, esto es, un valor entre un número finito de opciones, la cláusula `group by` puede ser útil para separar las instancias por categoría. Con ello se podría realizar un tratamiento particular de los datos de una categoría o incluso descartarlos.

En cuanto a la discretización de valores numéricos, CEP permite expresar condiciones en base a intervalos. Para el caso de Esper, estos se indican como condición en la cláusula `where` utilizando `between`. También define los operadores `in` y `not in`, válidos para atributos numéricos o resultados de subconsultas. Igualmente, los intervalos pueden definirse con operadores lógicos simples ($>$, $<$). Una vez identificados las instancias que se encuadran en el rango establecido, el valor del atributo deberá sustituirse por el identificador del intervalo. Por ejemplo, si se quiere discretizar la altura de una persona, el atributo resultante podría tomar los valores *baja*, *media* o *alta*.

Finalmente, para la generación de valores puede distinguirse entre creación de características y de instancias. Para lo primero, funciones aritméticas como las mostradas en la tabla 3 pueden formar parte de la cláusula `select`, indicando el atributo para el cual se calculan. Dicho cálculo se efectuará para todos los eventos anteriores del flujo o de la ventana definida, incluyendo también el actual. De esta forma se van a generar nuevas características con información histórica más o menos reciente, permitiendo que el algoritmo de aprendizaje se adapte a fluctuaciones en los datos. Esto tiene aplicación directa en sistemas de predicción meteorológica o de análisis de mercado, donde el factor temporal es fundamental. Respecto a la generación de instancias, estas mismas funciones, junto a otras que tengan en cuenta la frecuencia de los datos (`mostFrequent`, `leastFrequent`), pueden servir para crear instancias artificiales, en lugar de usar valores aleatorios. Este mecanismo es habitual cuando se quieren reemplazar valores

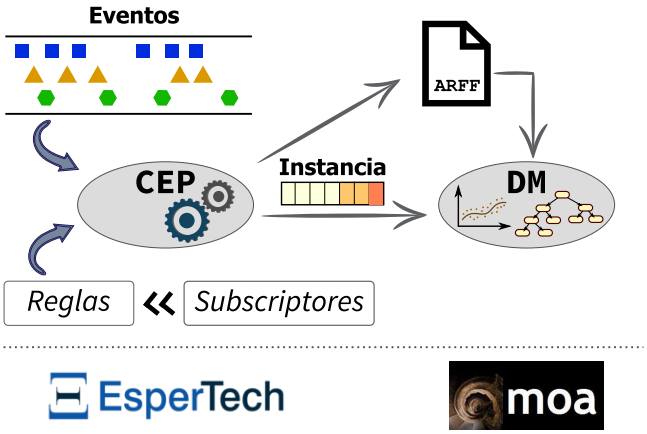

Figura 1. Elementos principales de la solución propuesta y su relación.

perdidos o cuando se desea balancear los datos, esto es, conseguir que haya un número equivalente de instancias perteneciente a cada clase.

Todas estas opciones se complementan con la posibilidad de usar ventanas para establecer el alcance de los operadores, así como la potente parametrización que permiten. Además, muchos sistemas CEP permiten al usuario definir sus propias funciones de agregación, lo cual facilita la definición de transformaciones más complejas.

3.2. Estructura de la solución propuesta

La figura 1 muestra la estructura de la solución propuesta a alto nivel. Como puede observarse, consta de dos bloques principales, uno que implementa el preprocesado de datos con CEP y otro encargado de invocar a los algoritmos de minería de datos.

El componente *CEP* es el encargado de recibir los datos en forma de eventos, los cuales pueden provenir de uno o más flujos (representados con distintas formas en la figura). A su vez, es necesario disponer de un conjunto de reglas que codifiquen las acciones de preprocesado a realizar. A cada una de las reglas se le asocia un subscriptor, encargado de recoger los eventos producidos por las reglas tras dispararse. A partir de dichos eventos, se construyen las instancias que sirven de entrada al algoritmo de aprendizaje. Cada instancia puede contener tres tipos de características (representadas con distinto tono): 1) atributos originales, 2) atributos transformados, y 3) atributos derivados. Igualmente, algunos de los atributos del evento pueden ser descartados.

El componente *DM* contiene los algoritmos y utilidades necesarias para llevar a cabo la fase de minería de datos. Dado que la propuesta se centra en el aprendizaje en tiempo real, las instancias son directamente recibidas por el algoritmo para entrenar y validar el modelo de decisión correspondiente (clasificación, regresión, etc.). No obstante, también se contempla la carga de las instancias desde un fichero en formato ARFF [2], habitual entre las herramientas del área.

[2] Especificación del formato ARFF: `https://waikato.github.io/weka-wiki/arff_stable` (último acceso: 5/6/2019)

La solución propuesta se ha implementado en Java, desarrollando dos *wrappers*, uno por componente. Por un lado, se ha utilizado Esper, mencionado anteriormente, como motor CEP. Y, por otro lado, la herramienta MOA [1], la cual proporciona la implementación de los algoritmos de aprendizaje incremental. Además, MOA también da soporte a la definición de experimentos con los que evaluar su rendimiento.

4. Validación experimental de la propuesta

En esta sección se realiza una validación inicial de la propuesta mediante el desarrollo de dos casos de estudio. Antes, se describe la metodología experimental seguida.

4.1. Metodología

Los datos de entrada se han tomado de conjuntos de datos disponibles en dos conocidos repositorios públicos: UCI[3] y OpenML[4]. En ambos casos se trata un fichero en formato ARFF, cuyas instancias son leídas una a una, inyectándolas como eventos simples en el flujo de entrada de CEP. Los conjuntos de datos utilizados se describen en cada caso de estudio, al igual que las reglas definidas, ya que son específicas de cada dominio de aplicación. Un aspecto común es el uso de ventanas, para lo cual se van a definir varios tamaños a fin de analizar su influencia. Respecto a la fase de aprendizaje, se va a seguir el enfoque *test-then-train*, consistente en validar primero el modelo de predicción con la nueva instancia recibida, y después actualizar el modelo entrenando con ella [1]. Los resultados son evaluados desde las tres dimensiones habituales en aprendizaje en tiempo real: precisión en la predicción, tiempo y memoria. Los algoritmos y medidas de precisión dependen de si la tarea es de regresión o de clasificación:

1. Primer caso de estudio: regresión
   a) Algoritmos. MOA proporciona una implementación del método *stochastic gradient descent* (SGD), que puede ser parametrizado con tres funciones: SVM (*support vector machine*), logística y lineal.
   b) Medida. Diferencia entre los valores reales y estimados, obtenido mediante el *root mean squared error* (RMSE).
2. Segundo caso de estudio: clasificación
   a) Algoritmos. Se han seleccionado cuatro algoritmos: *Hoeffding tree*, que induce un árbol de decisión; *k-nearest neighbours*, que clasifica de acuerdo a la instancia más parecida; *naive Bayes*, que se fundamenta en el teorema de Bayes; y *rule-based classifier*, que genera un conjunto de reglas de clasificación.
   b) Medidas. Porcentaje de muestras clasificadas correctamente (*accuracy*).

Para garantizar la representatividad de los resultados en cuanto al tiempo y la memoria, cada algoritmo se ha ejecutado 10 veces. Para todos los algoritmos se utilizan los parámetros por defecto definidos en MOA. Finalmente, las pruebas se han ejecutado en un ordenador con sistema operativo Windows 10, procesador Intel Core i7-4790 a 3.60GHz y 8GB de memoria RAM.

[3] Repositorio UCI: `https://archive.ics.uci.edu/ml` (último acceso: 5/6/2019)
[4] Repositorio OpenML: `https://www.openml.org` (último acceso: 5/6/2019)

4.2. Primer caso de estudio: transformación de datos

Los sistemas de predicción meteorológica representan un ejemplo de aprendizaje en tiempo real. En este caso de estudio se aborda la estimación de la temperatura interior de una casa inteligente para ajustar automáticamente su sistema de calefacción. El estudio en el que nos basamos [15], describe tanto el conjunto de datos como el proceso de preprocesado seguido, el cual va a ser reproducido aquí con CEP a modo ilustrativo.

El conjunto de datos[5] recoge mediciones de sensores meteorológicos ubicados en el interior y exterior de la casa. De los 23 atributos, 21 se corresponden con mediciones de temperatura, humedad, CO2, etc. Los otros dos son la marca temporal (*timestamp*) y el día de la semana. En total cuenta con 2.764 muestras tomadas cada 15 minutos. Para estimar la temperatura, los autores del estudio se basan en las mediciones previas de temperatura e irradiación solar, para las cuales indican el siguiente preprocesado:

1. Normalizar de la diferencia de temperatura (*temp*), y tomar los T valores previos.
2. Escalar la medida de irradiación solar (*irrad*), y tomar los I valores previos.
3. Incorporar la hora de la medición como variable binaria.

Por tanto, el conjunto de datos resultante consta de T características con las diferencias de temperatura, I características con los valores de irradiación solar, y 24 características para representar la hora, donde solo una de ellas tomará el valor 1. Como variable a predecir, se guarda la temperatura actual. Los autores del estudio proponen el uso de 15 valores de temperatura y 4 de irradiación solar. Nos referiremos a esta configuración de referencia como $\{15, 4\}$.

Definición de reglas CEP. Para la normalización de las diferencias de temperatura, es necesario conocer la media y la desviación estándar, mientras que para escalar la irradiación solar se necesitan sus valores máximos y mínimos. Sin embargo, la forma en la que se obtienen estos valores no se describe en el estudio, por lo que aquí van a considerarse varias alternativas para su comparación.

Una primera opción consiste en suponer que estos parámetros son conocidos a priori. Basándose en este supuesto, la regla mostrada en el código 2 sirve para preprocesar las muestras. En ella, `normalize` y `scale` son funciones definidas por el programador. Aparte, se recogen las muestras completas de la ventana para que el subscriptor pueda realizar esas mismas operaciones sobre las muestras anteriores. La ventana espacial, que se indica con el comando `length`, se configura al valor máximo entre T e I. Finalmente, la función `getHourOfDay` permite extraer la hora del *timestamp*.

```
select  Function.normalize(temp, prev(temp)), window(*),
        Function.scale(irrad), timestamp.getHourOfDay()
from    Temperatura#length(tam)
```

2. Regla para transformar las mediciones en base a estadísticas conocidas.

Como segunda opción, las estadísticas van a calcularse en base a las muestras que conforman la ventana. El código 3 muestra cómo se codifica la regla CEP. Debido a que

[5] `https://archive.ics.uci.edu/ml/datasets/SML2010` (último acceso: 5/6/2019)

la media y la desviación estándar requieren la diferencia de temperatura, no es posible utilizar las funciones `avg` y `stddev` en la cláusula `select`, por lo que son calculadas por el subscriptor. En este caso se ha decidido tomar los valores de temperatura e irradiación en la ventana por separado.

```
select   temp, prev(temp), irrad, timestamp.getHourOfDay(),
         window(temp), window(irrad),
         min(irrad), max(irrad)
from     Temperatura#length(tam)
```

3. Regla para transformar las mediciones en base a la ventana.

Finalmente, se va a considerar la actualización de las estadísticas a medida que se reciben las muestras. Para ello, en la regla anterior se van a utilizar las funciones `minever` y `maxever` para obtener el mínimo y el máximo histórico, respectivamente. En el caso de la media y la desviación estándar, el subscriptor va almacenando el resultado de acumular cada nueva diferencia de temperatura.

Resultados experimentales. Se van a comparar las tres opciones de preprocesado con diferentes tamaños de ventana, incluyendo valores superiores e inferiores a los propuestos originalmente. La tabla 4 recoge los valores de RMSE obtenidos, así como el tiempo medio de la fase de aprendizaje. Las celdas sombreadas representan mejoras respecto a la configuración de referencia ({15, 4}, en cursiva), mientras que en negrita se indica el mejor resultado para el algoritmo correspondiente.

Respecto al error cometido, se observa que los valores de RMSE se reducen ligeramente al considerar el preprocesado con las estadísticas históricas frente a las absolutas. Sin embargo, si las estadísticas se calculan sobre la ventana, el error se incrementa en exceso cuando se utilizan ventanas pequeñas, no siendo comparables (se omiten de la tabla). Esto puede deberse a que el número de muestras no es lo suficientemente grande como para obtener una media y desviación estándar representativas, pues solo se están considerando las diferencias de temperatura en un intervalo de 2-3 horas.

Por otra parte, el método de regresión que mejores resultados obtiene es SGD con regresión lineal, independientemente del tipo de preprocesado realizado. Respecto a la configuración de las ventanas, se pueden lograr ciertas mejoras cambiando los valores utilizados en el estudio original. En concreto, la ganancia obtenida varía entre 0,07 % y el 7,68 % para las estadísticas absolutas, y entre el 0,07 % y el 7,40 % para las estadísticas históricas. Contrario a lo que cabría esperar, considerar un número mayor de muestras no siempre ayuda a mejorar la estimación.

Finalmente, no existen grandes diferencias respecto al tiempo (ver tabla 4), ya que los tres métodos de regresión tienen la misma base algorítmica. Se aprecia un ligero incremento conforme aumenta el tamaño de la ventana, pues al tener más características de entrada, el algoritmo debe ajustar un mayor número de coeficientes. La desviación estándar entre las 10 ejecuciones es siempre inferior a 0,01, mientras que los valores de memoria oscilan entre 4.991 y 9.896 bytes. Este incremento se debe al aumento en el número de características.

Tabla 4. Error y tiempo medio obtenidos para la predicción de la temperatura.

	Ventana	RMSE			Tiempo medio (s)		
	{T, I}	*Reg. SVM*	*Reg. Log.*	*Reg. Lin.*	*Reg. SVM*	*Reg. Log.*	*Reg. Lin.*
Absoluto	{10, 2}	1,3831	1,2569	0,6456	0,0375	0,0391	0,0422
	{10, 4}	1,3761	1,2529	0,6444	0,0453	0,0406	0,0453
	{10, 8}	1,3587	1,2452	0,6440	0,0391	0,0406	0,0344
	{15, 2}	1,4692	1,3139	0,6449	0,0328	0,0344	0,0297
	{15, 4}	*1,4631*	*1,3098*	*0,6437*	*0,0313*	*0,0359*	*0,0344*
	{15, 8}	1,4499	1,3014	0,6433	0,0422	0,0438	0,0438
	{20, 2}	1,4926	1,3391	0,6405	0,0438	0,0438	0,0406
	{20, 4}	1,4873	1,3349	0,6392	0,0422	0,0453	0,0453
	{20, 8}	1,4714	1,3261	0,6385	0,0484	0,0531	0,0469
Ventana	{20, 2}	1,6492	1,5583	0,9379	0,0313	0,0328	0,0313
	{20, 4}	1,6435	1,5553	0,9380	0,0391	0,0359	0,0328
	{20, 8}	1,6496	1,5605	0,9357	0,0375	0,0375	0,0375
Histórico	{10, 2}	1,1585	1,0626	0,6011	0,0313	0,0266	0,0281
	{10, 4}	1,1472	1,0569	0,5988	0,0281	0,0313	0,0297
	{10, 8}	**1,1288**	**1,0458**	0,5974	0,0375	0,0375	0,0375
	{15, 2}	1,2266	1,1030	0,5999	0,0375	0,0359	0,0359
	{15, 4}	*1,2123*	*1,0970*	*0,5978*	*0,0359*	*0,0328*	*0,0359*
	{15, 8}	1,1837	1,0851	0,5967	0,0469	0,0438	0,0391
	{20, 2}	1,2504	1,1196	0,5969	0,0422	0,0438	0,0359
	{20, 4}	1,2350	1,1135	0,5947	0,0469	0,0453	0,0438
	{20, 8}	1,2014	1,1011	**0,5932**	0,0469	0,0484	0,0531

4.3. Segundo caso de estudio: generación de características

Otro posible dominio de aplicación consiste en predecir la demanda o el consumo de algún servicio, sobre todo cuando el precio se calcula en base a ello. Como ejemplo práctico de este caso, se va a utilizar un conjunto de datos con registros de la demanda de electricidad en dos estados de Australia (*New South Wales* y *Victoria*)[6]. Este conjunto de datos consta de 45.312 muestras, tomadas cada 30 minutos. Entre otros, contiene cuatro atributos numéricos que indican la demanda de electricidad y el precio de la electricidad en cada estado. El objetivo es predecir si el precio va a subir o bajar, por lo que el problema se puede abordar desde una perspectiva de clasificación binaria.

```
select   *, avg(nswprice), avg(nswdemand)
         avg(vicprice), avg(vicdemand)
from     Electricidad#length(tam)
```

4. Regla CEP para calcular la media del precio y de la demanda de electricidad.

Definición de reglas CEP. En este ejemplo se pretende estudiar si incorporar información temporal es útil para predecir el cambio de precio. Para ello se propone incluir la media de la demanda y del precio como nuevas características, estableciendo una ventana para calcularlos. El código 4 muestra la regla CEP que genera las instancias a

[6] `https://www.openml.org/d/151` (último acceso: 5/6/2019)

partir de los eventos. Como puede apreciarse, en este caso la regla recupera el evento completo (mediante el símbolo *) y se calculan los cuatro atributos. De forma similar, se ha definido otra regla que, en lugar de la media, obtiene el mínimo y el máximo de los cuatro atributos mostrados.

Con el objetivo de estudiar si las nuevas características son más efectivas que considerar el precio y la demanda actual por sí solos, también se han generado dos conjuntos de datos adicionales donde el precio y la demanda puntual son eliminados. Por tanto, en total se dispone de cuatro conjuntos de datos: enriquecido con medias, enriquecido con mínimos y máximos, reducido con medias, y reducido con mínimos y máximos.

Resultados experimentales. Se han ejecutado los cuatro algoritmos de clasificación sobre el conjunto de datos original y los generados con CEP. La tabla 5 muestra los resultados de precisión y el tiempo medio. Al igual que en el primer caso de estudio, se han considerado varios tamaños de ventana. Los símbolos $(+)$ y $(-)$ se utilizan para indicar si el conjunto de datos es el enriquecido o el reducido, respectivamente.

En primer lugar, los resultados muestran que es apropiado mantener las características originales de demanda y precio. Además, incluir características con información temporal conlleva una mejora en la predicción en tres de los cuatro algoritmos considerados. No obstante, la ganancia concreta depende del algoritmo utilizado: entre un 0,64 % y 17,58 % para *Hoeffding tree*, de 0,05 % a 1,05 % para *naive Bayes* y entre 0,06 % y 8,22 % para *rule-based classiffier*. El único para el que no se consiguen mejoras es *kNN*, debido a que se basa en la similitud entre instancias. Dado que todas las instancias cercanas en tiempo tienen un valor medio (o mínimo/máximo) parecido, el algoritmo tenderá a asignarles la misma clase, no detectando correctamente el cambio de una a otra. Este hecho se acrecienta cuando las características originales de precio y demanda son eliminadas. Respecto al tamaño de la ventana, se observa que los valores más adecuados son 50 y 100. Esto indica que es preferible utilizar la información más reciente, lo que puede ser sintomático de una fluctuación frecuente del precio.

Las diferencias en cuanto al tiempo se deben principalmente al tipo de algoritmo utilizado. El caso más significativo es de nuevo *kNN*, pues el incremento en el número de características eleva el tiempo necesario para calcular la distancia entre instancias. A pesar de que el tiempo suele aumentar respecto al necesario para aprender del conjunto original, la mejora en la precisión compensa este hecho, especialmente al utilizar un algoritmo rápido como es *Hoeffding tree*. Tanto este algoritmo como *naive Bayes* son muy estables (la desviación estándar no es superior a 0,03). Para *kNN*, la desviación estándar aumenta ligeramente (entre 0,03 y 0,26), mientras que el clasificador basado en reglas es el que más dispersión presenta (entre 0,31 y 1,23).

Finalmente, aunque la memoria puede llegar a ser casi el doble que la original, nunca es superior a 11MB. Además, cabe destacar que los modelos de decisión generados con los datos preprocesados pueden llegar a ser más pequeños que el original, facilitando su comprensión. Por ejemplo, se ha logrado aumentar la precisión cerca de un 10 % reduciendo el número de nodos del árbol de decisión (de 57 nodos a 51), mientras que la mejor ganancia obtenida con el clasificador basado en reglas (8,22 %) se ha conseguido con un modelo de 25 reglas, frente a las 36 del conjunto de datos original.

Tabla 5. Precisión y tiempo medio obtenidos para predecir el cambio de precio de la electricidad.

	Ventana	Accuracy				Tiempo medio (s)			
		HTree	*kNN*	*NBayes*	*Rule Cl.*	*HTree*	*kNN*	*NBayes*	*Rule Cl.*
Original		*79,20*	***78,38***	*73,36*	*73,21*	*0,21*	*9,53*	*0,14*	*13,55*
Media (−)	10	68,22	72,51	62,79	63,81	0,25	9,54	0,20	11,57
	50	64,97	72,03	60,32	61,29	0,25	9,85	0,20	10,55
	100	66,93	71,87	60,77	63,46	0,24	10,04	0,20	12,54
	500	66,76	71,39	59,47	62,54	0,24	11,35	0,19	11,32
	1000	66,60	71,37	59,49	61,73	0,25	13,01	0,20	11,78
Media (+)	10	82,37	77,09	70,76	71,91	0,36	12,32	0,28	12,71
	50	**93,12**	77,74	73,76	**79,23**	0,35	12,82	0,26	14,65
	100	86,15	77,32	**74,13**	73,25	0,34	13,06	0,27	11,81
	500	81,73	76,33	73,40	73,06	0,35	15,34	0,26	13,61
	1000	81,07	75,92	73,27	72,21	0,35	17,78	0,26	11,62
Min/Max (−)	10	68,54	72,89	63,97	62,94	0,29	12,09	0,20	13,01
	50	66,98	71,46	60,91	61,67	0,26	13,28	0,21	10,59
	100	63,85	71,27	60,98	61,80	0,28	13,78	0,21	12,29
	500	63,37	70,91	58,97	62,46	0,29	19,13	0,21	11,45
	1000	64,92	71,35	58,53	62,78	0,28	20,41	0,21	10,85
Min/Max (+)	10	79,70	76,68	69,40	74,69	0,36	14,73	0,27	16,38
	50	86,95	76,59	73,78	74,38	0,37	16,46	0,28	13,18
	100	83,17	76,08	73,46	74,81	0,38	17,27	0,27	15,47
	500	81,09	75,09	72,33	72,33	0,36	24,89	0,28	14,57
	1000	81,43	75,48	71,97	72,77	0,35	26,62	0,27	15,30

5. Conclusiones

En este trabajo se ha propuesto el uso del procesamiento de eventos complejos como un nuevo mecanismo para preprocesar flujos de datos con los que abordar tareas de minería de datos. La riqueza y expresividad del lenguaje EPL permiten definir de forma flexible reglas de preprocesamiento intuitivas para los expertos del dominio. Con ellas se pueden construir filtros basados en el contenido de los datos, transformar sus características o incorporar información temporal de forma rápida y comprensible.

La solución implementada integra dos sistemas conocidos: Esper, como motor CEP, y MOA, una librería para aprendizaje automático incremental. Así, los datos recibidos en forma de eventos son preprocesados por Esper y transformados en instancias que sirven de entrada a los algoritmos de clasificación y regresión disponibles en MOA. El desarrollo de dos casos de estudio muestra cómo se pueden definir distintas reglas de preprocesado para tratar datos meteorológicos y de consumo energético. Gracias a los operadores de CEP y al uso de ventanas, se consigue incorporar fácilmente información temporal al proceso de aprendizaje, la cual ha contribuido en la mejora, en algunos casos superior al 15 %, de la precisión de modelos de decisión.

Como trabajo futuro, se contempla la realización de más casos de estudio para abordar otras tareas de preprocesado, así como comparar el rendimiento de CEP frente a otras técnicas de preprocesado existentes. Además, se pretende profundizar en el estudio de la influencia de las ventanas y abordar cómo CEP puede ayudar a detectar el

denominado *concept drift*, esto es, el cambio que puede experimentar la distribución de los datos.

Agradecimientos

Trabajo financiado por el Ministerio de Economía y Competitividad (proyecto TIN2014-52034-R).

Referencias

1. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: Massive Online Analysis. Journal of Machine Learning Research **11**, 1601–1604 (2010)
2. Bolon-Canedo, V., Fernández-Francos, D., Peteiro-Barral, D., Alonso-Betanzos, A., Guijarro-Berdiñas, B., Sánchez-Maroño, N.: A unified pipeline for online feature selection and classification. Expert Systems with Applications **55**, 532–545 (2016)
3. Boubeta-Puig, J., Bravetti, M., Llana, L., Merayo, M.G.: Analysis of temporal complex events in sensor networks. J. Information and Telecommunication **1**(3), 273–289 (2017)
4. Cugola, G., Margara, A.: Processing Flows of Information: From Data Stream to Complex Event Processing. ACM Computing Surveys **44**(3), 15:1–15:62 (2012)
5. Flouris, I., Giatrakos, N., Deligiannakis, A., Garofalakis, M., Kamp, M., Mock, M.: Issues in complex event processing: Status and prospects in the Big Data era. Journal of Systems and Software **127**, 217–236 (2017)
6. Gaber, M.M., Zaslavsky, A., Krishnaswamy, S.: Mining Data Streams: A Review. SIGMOD Rec. **34**(2), 18–26 (2005)
7. García, S., Luengo, J., Herrera, F.: Data Preprocessing in Data Mining. Springer (2014)
8. Lee, O.J., Jung, J.E.: Sequence Clustering-based Automated Rule Generation for Adaptive Complex Event Processing. Future Generation Computer Systems **66**, 100–109 (2017)
9. Luckham, D.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley (2001)
10. Margara, A., Cugola, G., Tamburrelli, G.: Learning from the Past: Automated Rule Generation for Complex Event Processing. In: Proc. 8th ACM Int. Conference on Distributed Event-Based Systems (DEBS). pp. 47–58. ACM (2014)
11. Moreno, N., Bertoa, M.F., Barquero, G., Burgueño, L., Troya, J., García-López, A., Vallecillo, A.: Managing Uncertain Complex Events in Web of Things Applications. In: Proc. Int. Conference on Web Engineering (ICWE). pp. 349–357 (2018)
12. Mousheimish, R., Taher, Y., Zeitouni, K.: Automatic Learning of Predictive CEP Rules: Bridging the Gap Between Data Mining and Complex Event Processing. In: Proc. 11th ACM Int. Conference on Distributed and Event-based Systems (DEBS). pp. 158–169 (2017)
13. Ramírez-Gallego, S., Krawczyk, B., García, S., Woźniak, M., Herrera, F.: A survey on data preprocessing for data stream mining: Current status and future directions. Neurocomputing **239**, 39–57 (2017)
14. Roudjane, M., Rebaïne, D., Khoury, R., Hallé, S.: Real-Time Data Mining for Event Streams. In: Proc. of IEEE Enterprise Computing Conference (EDOC). pp. 123–134 (2018)
15. Zamora, F., Romeu, P., Botella, P., Pardo, J.: On-line learning of indoor temperature forecasting models towards energy efficiency. Energy and Buildings **83**, 162–172 (2014)
16. Zhang, S., Zhang, C., Yang, Q.: Data preparation for data mining. Applied Artificial Intelligence **17**(5-6), 375–381 (2003)