UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Simulación de datos de sensores industriales

TRABAJO DE FIN DE MÁSTER

Máster en Big Data Analytics

Autor: Pedro Henrique Mano Figueiredo Fernandes
Tutor: Francisco Sánchez Cid

Curso 2015-2016

Abstract

Los entornos industriales hacen uso de sensores para obtener mediciones de varios factores en su cadena de producción. En las fases iniciales, la cantidad de datos generados es muy reducida, por lo que no es significativa para permitir la aplicación de técnicas de Big Data. Estas técnicas pueden descubrir muchos patrones en los datos, útiles para detección de anomalías o predicción futura. El proyecto descrito en este documento busca simular nuevos datos a través de un pequeña cantidad de datos generados por sensores industriales. La aplicación de estas técnicas se extiende a todo el contexto de proyectos IoT (*Internet of Things*), donde los sensores representan la principal fuente de datos.

Palabras clave: Machine learning, Big Data, PCA

Abstract

The industrial environments make use of sensors for acquiring metrics on several variables in their production pipeline. In the initial phases, the generated data has very low volume, making it hard to apply Big Data techniques. These techniques can bring much insight over the data, useful for anomaly detection or future prediction. The project described in this document seeks the simulation of new data using a small amount of data generated by industrial sensors. The application of such techniques can be extended to the entire context of IoT (*Internet of Things*), where sensors represent the main source of data.

Key words: Machine learning, Big Data, PCA

Índice general

Índice de figuras

Índice de tablas

Índice general

CAPÍTULO 1
Introducción

Este proyecto tiene como fuente de datos uno o varios sensores de un mecanizado industrial de inyección de plástico. Los sensores efectúan mediciones de varios factores con una regularidad temporal, generando así muestras de datos en determinados intervalos de tiempo. Cuando se trata de sensores, es normal que los datos no tengan la calidad necesaria para aplicar técnicas matemáticas. Hay que tener en cuenta que la frecuencia de las mediciones no es necesariamente constante, que pueden existir interrupciones, ruido y redundancias. Además, en el ámbito de IoT (*Internet of Things*), se añade la inherente conectividad de los aparatos como factor de complejidad.

Las técnicas de Big Data, en particular *Machine Learning*, necesitan gran cantidad de datos para poder inducir modelos matemáticos que expliquen esos datos. Cuanto más datos mejor, para un aprendizaje más robusto y fiable. Sin embargo, al principio de los proyectos, la cantidad de datos recolectados suele ser pequeña y insuficiente para poder extraer conocimiento significativo de los mismos. El objetivo de este proyecto es aprender y simular el comportamiento de uno o varios sensores a través de una cantidad reducida de datos recolectados en los mismos.

Una vez preparado el *dataset*, es necesario aprender un modelo matemático que lo describa. Las técnicas usadas para ese efecto son del ámbito de *Machine Learning*.

1.1 Motivación

Los datos recolectados en sensores industriales tienen poco volumen al principio de la implantación. Con estos datos, es posible aprender un modelo para predecir el comportamiento futuro. Sin embargo, antes hay que tener en cuenta que la calidad de datos no siempre es la adecuada para aplicar técnicas matemáticas. Así que es muy importante conocer el *dataset*, limpiarlo y normalizarlo. Además hay que lidiar con frecuencias de mediciones inconstantes, con el ruido y repetición de mediciones. Para estos problemas, las técnicas de *Machine Learning* ofrecen buenas soluciones.

1.2 Objectivos

El objetivo de este proyecto es aprender y simular datos de sensores industriales. Los nuevos datos, de mucho mayor volumen, permitirán aplicar técnicas de Big Data. Como los datos iniciales son provenientes de sensores, la calidad no está asegurada, por lo que se tendrá que emplear técnicas de limpieza y normalización. El aprendizaje del *dataset* pasa por aplicación de *Machine Learning*, para inducir un modelo matemático y

poder inferir datos futuros. Para comprobar la validez de la solución propuesta, se tiene que aplicar métodos estadísticos. Potencialmente, las herramientas convencionales no tengan capacidad para un volumen muy grande de datos- en ese caso se exige el uso de herramientas más poderosas (Big Data).

Los pasos descritos son muy comunes en análisis de datos y en particular en el *pipeline* de Big Data.

1.3 Estructura del documento

Este documento se estructura de la siguiente forma: un análisis sobre el *dataset* y respectivas transformaciones para adecuar los datos; el desarrollo del problema, con las soluciones propuestas y respectiva base teórica; conclusiones del trabajo realizado y posibles mejoras; y termina con las fuentes bibliográficas que han servido de base del estudio.

CAPÍTULO 2

Descripción del problema

TODO

CAPÍTULO 3
Estado del Arte

TODO

CAPÍTULO 4
Solución propuesta

La solución propuesta para simular datos empieza con la proyección de los datos del espacio multi-variante original a un espacio ortogonal más reducido. Esta técnica de Machine Learning se conoce por PCA (*Principal Component Analysis*). El nuevo espacio (con menos dimensiones) contiene las componentes que mejor explican los datos, por lo que se eliminan ruido y redundancias.

A través de los valores de cada componente, se generan nuevos datos con distribución Gaussiana. Esta simulación puede ser de miles o millones de nuevos puntos - no es una tarea computacional exigente y además tendrá utilidad más adelante. La simulación Gaussiana servirá como base del estudio, una vez que, al ser invertida, genera una población que es idéntica a la original. La comprobación de dicha similitud de poblaciones se hace a través del estadístico T-cuadrado de Hotelling.

Los datos originales tienen un sentido temporal. Las componentes lo tendrán también, así que se procede al análisis de la serie temporal de esas componentes. El modelo propuesto es ARIMA (*Autoregressive Integrated Moving Average*). Los modelos de series temporales se ajustan a los datos para mejor comprensión de estos o para predicción de nuevos puntos en la serie. El objetivo es dotar la simulación de un patrón temporal aprendido previamente.

Teniendo en cuenta que el estudio se sustenta en la hipótesis nula de la simulación Gaussiana, para cada predicción de las varias componentes con marca temporal, se buscará el punto multi-dimensional más cercano en la simulación Gaussiana. Los puntos obtenidos de la distribución Gaussiana se usarán para re-alimentar el modelo ARIMA aprendido. Llegados aquí, tenemos nuevos valores simulados de las componentes y además con carácter temporal. Al re-proyectar este espacio al espacio original obtenemos nuevos valores (simulados) para las variables originales.

La simulación Gaussiana debe tener gran volumen para que la simulación final sea efectiva. Cuantos más puntos tenga mejor, para que la búsqueda encuentre puntos más cercanos. Con un volumen de datos muy grande, los paradigmas de programación convencionales no tienen capacidad de respuesta. Se adoptan herramientas Big Data (*Spark* y *MongoDB*) para permitir escalar la parte de simulación Gaussiana, su almacenamiento y la búsqueda de puntos cercanos.

4.1 Reducción de dimensionalidad - PCA

PCA pertenece a la familia de aprendizaje no supervisado (*unsupervised learning*), donde no existen etiquetas o clases a predecir. Los datos de entrenamiento son simplemente observaciones, sin estar clasificados. Las técnicas de unsupervised learning se usan para

descubrir grupos de similitud (clusters) en los datos; o para determinar la distribución de los datos en el espacio original; o también, que interesa a este estudio, para proyectar los datos en espacios de menos dimensiones.

Pocas dimensiones suelen ofrecer mejor insight sobre los datos. Como primera ventaja, permiten la aplicación de técnicas de visualización - resulta muy difícil visualizar datos en más de 3 dimensiones (incluso en 3 dimensiones puede ser difícil). Además, las componentes que no son principales (menos explicativas) suelen ser ruido o redundancia - con esta técnica se pueden atenuar los efectos de estos. La simulación de datos puede usar la ventaja de la eliminación de ruido y redundancia, generando así datos más relevantes.

PCA decompone el dataset en una série de componentes ortogonales que explican la variabilidad. Por ejemplo, si dos variables son directamente proporcionales, no hace falta tener en cuenta las dos para visualizar, porque conllevaría una complejidad inecesaria. PCA está muy relacionado con una técnica matemática llamada Singular Value Decomposition (SVD), tanto que muchas veces los nombres se usan intercambiados. De hecho, el algoritmo de PCA de scikit-learn usa la decomposición SVD de numpy. Sin embargo, el cálculo de SVD es muy intensivo para matrices grandes, porque calcula la matriz de varianzas-covarianzas para todos los componentes. Esto implica un gran consumo de memoria y CPU. Como alternativa, se propone usar una técnica iterativa llamada NIPALS (Nonlinear Iterative Partial Least Squares), que usa el numero de componentes reducido.

El nuevo espacio está constituido por variables independientes y ortogonales. Para cada variable (componente), se calculan valores aleatorios con distribución normal. Estos son los scores de las componentes de los nuevos datos. Al proyectar la nueva matriz de componentes de vuelta al espacio original, se obtienen datos simulados para todas las características. Si la validación de esta técnica demuestra que es correcta, se puede usar para la simulación de nuevos datos. La validación de los datos generados pasa por la aplicación de estadísticos de comparación de las muestras originales y las nuevas. Las técnicas utilizadas son los tests de Hotelling T-squared y Box's M. En el caso de Hotelling T-squared (o T2) se prueba la igualdad en las médias de las poblaciones. Box's M test tiene como suposición la igualdad de matrices de varianzas-covarianzas de las poblaciones. Los resultados demuestran que es viable usar esta técnica de simulación de datos.

4.2 Séries temporales - ARIMA

TODO

CAPÍTULO 5
Experimentación y resultados

TODO

5.1 Dataset

El dataset proporcionado es de reducida dimensión, tiene tan solo 5MB, por lo que la tarea de ingestión no es intensiva. Sin embargo, hay que tratar los datos en bruto antes de empezar a usarlos. El fichero de datos es de texto, así que hay que hacer determinadas conversiones para poder usar tipos más específicos, como sean fechas y números. El primer problema tiene que ver con los formatos de fecha, que no son correctos para el *locale* España, lo que exige una adaptación de los algoritmos de *parsing*, como se describe en el apartado *Transformación*.

5.1.1. Estructura

Un breve análisis del *dataset* en un editor de texto muestra que tiene un formato de campos separados por espacios y tabulaciones. La cantidad de espacios es variable:

```
␣␣␣␣␣␣␣␣Tiempoinicio␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣APHu␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣APVs␣
    ...

␣␣␣␣␣␣␣␣06-oct-2015␣21:57:03␣␣44.6␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣69.3␣...
␣␣␣␣␣␣␣␣06-oct-2015␣21:57:12␣␣45.1␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣69.0␣...
␣␣␣␣␣␣␣␣06-oct-2015␣21:57:21␣␣44.8␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣69.8␣...
␣␣␣␣␣␣␣␣...
␣␣␣␣␣␣␣␣
```

Listing 5.1: Ejemplo del *dataset*.

La primera línea contiene un *header* (cabecera), con 15 nombres:

```
Tiempoinicio APHu APVs ACPv ZSx ZUs H7x H1x H2x H6x H3x H4x H5x ACPx Svo
```

Listing 5.2: *Header* del *dataset*.

Se puede verificar también que hay una línea vacía después del *header*. El primer campo tiene un formato de fecha/hora y los demás campos tienen formato decimal.

5.1.2. Transformación

Para la ingestión y transformación de los datos se han usado librerías muy útiles y con muchas funcionalidades que facilitan bastante esas tareas: en los scripts Python se ha usado el paquete *Pandas* y en R la función *read.csv2* del paquete *utils*.

Las 14 variables decimales no ofrecen problemas en la ingestión del *dataset*. Para asegurar el formato decimal, es conveniente definir el separador decimal como punto ('.'). Con eso es suficiente para un em parsing correcto.

Las fechas son más complejas de procesar. El formato de mes da indicios de estar escrito en castellano: `oct, dic, mar, abr, may, jun`. Sin embargo, en el *locale* de España, el *standard* de abreviación de mes es con punto ('.'), por ejemplo `oct.`. Así que el *parsing* de fechas tuvo que ser ajustado. La estratégia ha sido el uso de una expresión regular para añadir el punto ('.') necesario en las abreviaciones de meses, como se puede ver en los siguientes *snippets* Python y R:

```
def parse_date(date_string):
    locale_date_string = re.sub("(.+-)(.+)(-.+)", "\\1\\2.\\3",
        date_string)
    return datetime.strptime(locale_date_string, "%d-%b-%Y␣%H:%M:%S")
```

Listing 5.3: *Parsing* de fechas en Python.

```
data$Tiempoinicio <- sub("(\\d+-)(\\w+)(-\\d+\\s\\d+:\\d+:\\d+)",
    "\\1\\2.\\3", data$Tiempoinicio)
data$Tiempoinicio <- as.POSIXct(data$Tiempoinicio, format="%d-%b-%Y␣
    %H:%M:%S")
```

Listing 5.4: *Parsing* de fechas en R.

El resultado final es un *dataset* estructurado con una marca temporal como índice y 14 características.

5.1.3. Distribución de los datos

Las mediciones de los sensores se distribuyen por varios días, no siempre consecutivos. Los días con mediciones son algunos

- 6-9, 12 de octubre de 2015
- 14 de diciembre de 2015
- 7, 8, 11, 14 de marzo de 2016
- 6, 29 de abril de 2016
- 2, 26 de mayo de 2016
- 8, 17, 20 de junio de 2016

En cada día las muestras son dispares y en momentos distintos del día, probablemente por corresponder a distintas pruebas. Así, también las frecuencias de muestreo son variables, en algunos días parece indicar una frecuencia de 10 segundos pero hay otros donde son de 1 minuto.

5.2 PCA

TODO

5.2.1. PCA iterativo - NIPALS

TODO

5.2.2. Test de hipótesis - Estadístico T-cuadrado de Hotelling

TODO

5.2.3. Simulación con distribución Gaussiana - hipótesis nula

TODO

5.3 ARIMA

TODO

5.3.1. Simulación con todo el dataset

TODO

5.3.2. Simulación con datos de 1 día

TODO

5.3.3. Simulación con búsqueda de puntos cercanos de la Gaussiana

TODO

CAPÍTULO 6
Big Data

TODO

6.1 MongoDB

TODO

6.2 Spark

TODO

CAPÍTULO 7
Conclusiones

TODO

CAPÍTULO 8

Trabajos futuros

Bibliografía

[1] Jon Shlens. A TUTORIAL ON PRINCIPAL COMPONENT ANALYSIS. Derivation, Discussion and Singular Value Decomposition. Version 1, 25 March, 2003.

[2] Unknown Authors. The Truth about Principal Components and Factor Analysis. 28 September, 2009.

[3] Christopher M. Bishop. Pattern Recognition and Machine Learning. Springer, 2006.

[4] Principal component analysis (PCA). Consultar http://scikit-learn.org/stable/modules/decomposition.html#principal-component-analysis-pca.