Aplicación Java para distribución de código en R

Roldán Galán Lafarga
rolgalan@gmail.com

Huesca - 24 de noviembre de 2009

Descripción problema

- Existen muchos problemas, cuya resolución informática requiere de un enorme tiempo de cómputo.
- En la actualidad, la solución más rentable para mejorar la eficiencia, consiste en ejecutar distintas partes del programa en diferentes computadores paralelamente, de modo que este tiempo se reduce enormemente.

¿Por qué no hacerlo también en **R**?

Descripción problema

- Existen muchos problemas, cuya resolución informática requiere de un enorme tiempo de cómputo.
- En la actualidad, la solución más rentable para mejorar la eficiencia, consiste en ejecutar distintas partes del programa en diferentes computadores paralelamente, de modo que este tiempo se reduce enormemente.

¿Por qué no hacerlo también en **R**?

¿Cómo hacerlo?

Máquinas de cálculo

- Programa en el lenguaje Java que lanza una instancia de R en la que realiza los cálculos.
- Conectado a través de la red al equipo principal, desde el que recibirá el código que le corresponda ejecutar.

Servidor principal

- Interfaz gráfico que facilita la construcción de las sentencias R.
- También hace de servidor que las distribuirá a las máquinas anteriores.
- Finalmente, recibe todos los resultados.

¿Cómo hacerlo?

Máquinas de cálculo

- Programa en el lenguaje Java que lanza una instancia de R en la que realiza los cálculos.
- Conectado a través de la red al equipo principal, desde el que recibirá el código que le corresponda ejecutar.

Servidor principal

- Interfaz gráfico que facilita la construcción de las sentencias R.
- También hace de servidor que las distribuirá a las máquinas anteriores.
- Finalmente, recibe todos los resultados.

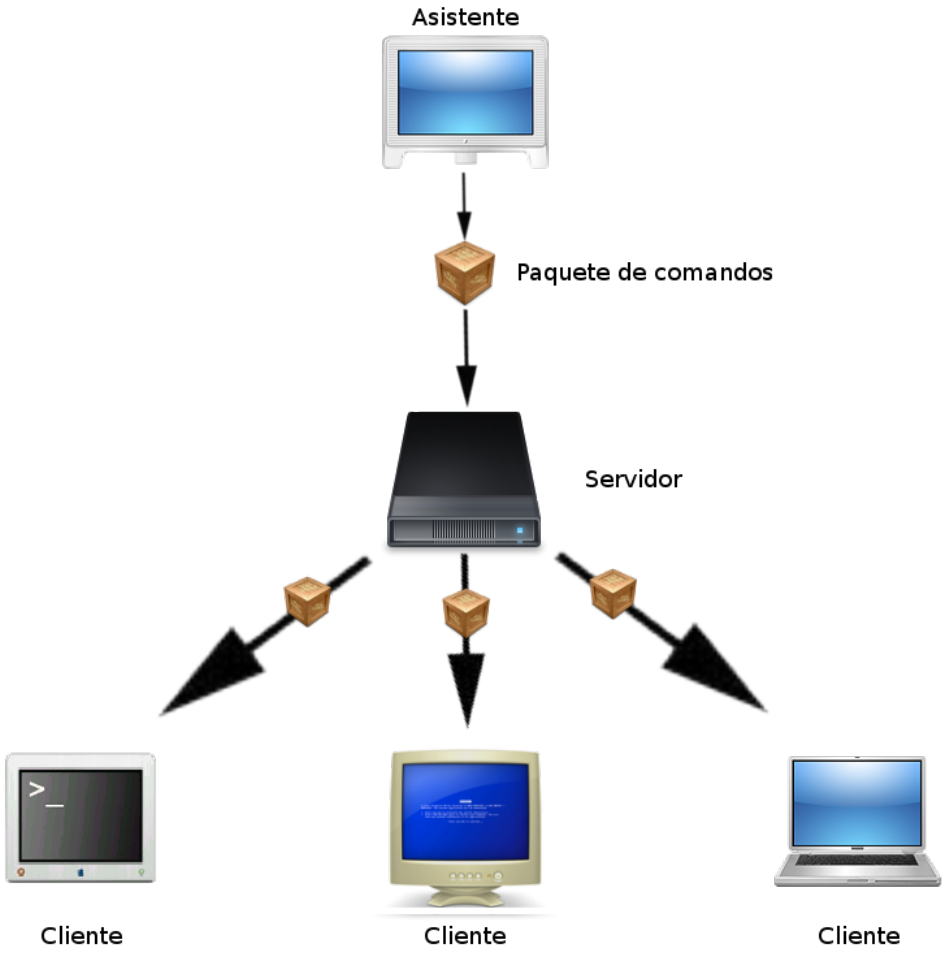
Sintetización de la arquitectura

Asistente/Servidor

Selección/Generación del código a ejecutar y distribución.

Cliente

Cálculo del resultado.



Ventajas

- Reducción del tiempo de cálculo.
- Generación automática de código de calidad.
- Posibilidad de realizar estudios extra al disponer de más tiempo.
- Interfaz gráfica más agradable al uso que la consola de comandos.

Entrenamiento de perceptrón multicapa bajo R

Nuestra primera aplicación estaba orientada a facilitar el entrenamiento de la red Perceptrón Multicapa para resolución de problemas de clasificación binaria.

¿Por qué?

Requieren bastante tiempo, que se incrementa con el número de neuronas, de entradas y el total de datos.
Es necesario un comando distinto para cada red con diferentes parámetros que se desee entrenar.

Comando de ejemplo (paquete *neural*)

```
mlptrain(inp,neurons,out,weight=c(),dist=c(),
alfa=0.2,it=200,online=TRUE, permute=TRUE,thresh=0,
dthresh=0.1,actfns=c(),diffact=c(),visual=TRUE)
```

Entrenamiento de perceptrón multicapa bajo R

Nuestra primera aplicación estaba orientada a facilitar el entrenamiento de la red Perceptrón Multicapa para resolución de problemas de clasificación binaria.

¿Por qué?

Requieren bastante tiempo, que se incrementa con el número de neuronas, de entradas y el total de datos.
Es necesario un comando distinto para cada red con diferentes parámetros que se desee entrenar.

Comando de ejemplo (paquete *neural*)

```
mlptrain(inp,neurons,out,weight=c(),dist=c(),
alfa=0.2,it=200,online=TRUE, permute=TRUE,thresh=0,
dthresh=0.1,actfns=c(),diffact=c(),visual=TRUE)
```

Entrenamiento de perceptrón multicapa bajo R

Nuestra primera aplicación estaba orientada a facilitar el entrenamiento de la red Perceptrón Multicapa para resolución de problemas de clasificación binaria.

¿Por qué?

Requieren bastante tiempo, que se incrementa con el número de neuronas, de entradas y el total de datos.
Es necesario un comando distinto para cada red con diferentes parámetros que se desee entrenar.

Comando de ejemplo (paquete *neural*)

```
mlptrain(inp,neurons,out,weight=c(),dist=c(),
alfa=0.2,it=200,online=TRUE, permute=TRUE,thresh=0,
dthresh=0.1,actfns=c(),diffact=c(),visual=TRUE)
```

Actividades esenciales

- Interfaz gráfica para facilitar el diseño de las diversas redes.
- Generación automática del código de las distintas redes en base a los parámetros.
- Computación distribuida para el entrenamiento de todos modelos deseados.
- Selección automática de la mejor arquitectura utilizando el valor del área bajo la curva ROC.
- Muestra por pantalla los parámetros significativos del mejor modelo.

Diseño de la red

- Selección de los datos de test o entrenamiento de forma manual, o aleatoriamente estableciendo el porcentaje de cada uno.
- Arquitectura
  - Pueden añadirse diversas arquitecturas una por una con sus diferentes configuraciones.
  - O bien seleccionar un rango de neuronas por capa y que la aplicación realice las combinaciones con ciertos parámetros en común.
- Generación automática de distintas redes en base a los parámetros.

Fácil ampliación

- Se ha prestado especial dedicación a la creación de un código Java fácilmente ampliable y escalable.
- Para implementar una versión con otros objetos R sólo sería necesario redefinir tres clases de la aplicación original.
- También sería necesario rediseñar parte de la IGU para concordar con el nuevo modelo de datos.
- Gracias a conceptos de la POO como herencia y polimorfismo, el resto del programa no requerirá modificación para trabajar con las nuevas clases. Y de hecho, escribir las nuevas será muy poco costoso.

Clases a implementar

Modelo de datos

Una de las clases Java representa el objeto R con el que se trabaja. Encapsula sus datos y parámetros.

Generador

Define todo el código R que podrá generar la aplicación. Se puede hacer tan compleja como se desee. Automatizar la creación de código mediante IGU o introducirlo a mano.

Extractor

Otra clase se encarga de enviar los comandos necesarios al espacio de trabajo R para extraer la información necesaria calculada e instanciar el equivalente en Java.