KSchool Supervised Learning - Notas adicionales

Javier Cañadillas - javier@canadillas.org - Repo Git

Git en Windows

Para aquellos que habéis tenido problemas con la máquina virtual, o queráis clonar el repositorio Git en Windows con el material del curso, descargad e instalad Git desde aquí. Una vez hecho esto podéis seguir las instrucciones del fichero README.md del repositorio Git de las clases.

Configurando tu propio Git remote

Si queréis mantener vuestro propio trabajo en Git, una cosa que podéis hacer es configurar vuestro propio Remote de Git y guardar vuestros cambios allí.

A continuación os explico cómo puede hacerse esto, asumiendo que ya tenéis un usuario en GitHub o GitLab y que ya habéis clonado el repositorio Git de la clase, kschool-supervised-learning.

1. Crea un nuevo proyecto en GitHub (o GitLab). No necesitáis inicializarlo con un README como dicen las instrucciones. Por convención, voy a asumir que habéis llamado a tu proyecto **my-ks-sl** y que vuestro usuario de GitHub se llama **ks-student**. En las instrucciones que van más adelante tendréis que cambiar esto por vuestro nombre de usuario y vuestro nombre de repositorio.
2. Una vez vuestro proyecto está creado, id al directorio local donde está el repo de la clase y añadid el proyecto como nuevo remote (recordad sustituir en el comando que va a continuación vuestro usuario GitHub y el nombre del proyecto que hayáis creado:

```
$ pwd #nos aseguramos que estais en el directorio correcto
kschool-supervised-learning
$ git remote add myremote https://github.com/ks-student/my-ks-sl
$ git remote -v #verificamos que teneis un nuevo remote
> gitlab https://gitlab.com/Sh3llD00m/kschool-supervised-learning.git
(fetch)
https://gitlab.com/Sh3llD00m/kschool-supervised-learning.git (push)
> myremote https://github.com/ks-student/my-ks-sl (fetch)
> myremote https://github.com/ks-student/my-ks-sl (push)
```

3. Ahora ya podéis utilizar vuestro nuevo remote para subir cambios. Cada vez modificais un fichero podéis seguir la secuencia:

```
$ git commit -a -m "New commit" #Anadimos los cambios a Git
```

```
$ git push myremote #Los subimos al nuevo remote
```

Desconexión del repositorio remoto

Una vez finalizado el curso, podéis desconectar mi repositorio (estoy asumiendo que no lo renombrasteis y se llama origin) sin más que hacer:

```
$ git remote rm origin
```

Datasets Scikit-Learn - Conversión a Pandas Dataframes

Ver el fichero **`Dataset_Conversion.ipynb`** en el repositorio Git para una descripción completa de los datasets de Scikit-Learn y cómo convertirlos a DataFrames de Pandas. Actualiza el repositorio haciendo `git pull` para tener la última versión de los ficheros en el mismo.

Técnicas de Ingeniería de Características

Escalado de datos - MinMaxScaler vs StandardScaler

En algunos ejercicios en clase utilizaremos escalado de datos antes de proceder al entrenamiento con nuestros modelos de aprendizaje supervisado. El escalado de datos es una técnica que cae dentro de la ingeniería de características. Aunque no es el objetivo formativo de la sesión, se incluye una breve explicación de en qué consisten. Scikit-Learn cuenta con preprocesadores para las dos técnicas más utilizadas de escalado de los datos, `MinMaxScaler` y `StandardScaler`.

Cuando nos encontramos con características que son muy diferentes en escala, está claro que los clasificadores/regresores que utilicen distancia euclídea (como k-Neighbors Classifiers) no van a funcionar bien o su rendimiento será malo. Esto también va a pasar con otros regresores, especialmente aquellos que que utilizan la técnica de gradiente descendiente para la optimización de la función de coste, como es el caso de la regresión logística, SVM o las redes neuronales. Los únicos clasificadores/regresores que son inmunes al impacto de la diferencia de escala son los árboles de decisión.

MinMaxScaler

Cuando usamos MinMaxScaler, lo que estamos haciendo es restar el mínimo a todos los valores (y por lo tanto marcar una escala que va del mínimo al máximo). Una vez hecho esto, dividimos entre (min - max).

El resultado es que los valores pertenecen al intervalo [0,1], lo que es aceptable en los casos en los que no estamos preocupados por estandarizar a lo largo de los ejes de varianza, por ejemplo en procesamiento de imágenes o en redes neuronales que esperan valores entre 0 ó 1.

Sin embargo, al haber movido todos nuestros valores al intervalo [0,1], tendremos menores desviaciones estándar y por lo tanto estaremos suprimiendo el posible efecto que tengan los outliers.

StandardScaler

Standar Scaler (o normalización z-score) resuelve este último inconveniente que tiene Min Max Scaler, al precio de ser más sensible a los outliers (y por lo tanto no pudiendo garantizar características balanceadas en su presencia).

StandardScaler sustrae primero la media, y por lo tanto atrae a los valores hacia cero (tendrá una media cero). Y luego divide los valores por la desviación estándar asegurando que la distribución resultante es estándar con una media 0 y una desviación de 1.

Cuándo usar una u otra

La respuesta es siempre, "depende del caso y de los datos" :-)

Aún así, como guía general, un StandarScaler irá bien salvo que los outliers sean datos que realmente no queremos computar. Sin embargo, si nuestro caso es que sí que queremos considerar la varianza (regresión logística, SVMs, perceptrones y redes neuronales), es StandardScaler será la opción. Si queremos mitigar el efecto de los outliers, podremos utilizar entonces un MinMaxScaler. Si tenemos árboles de decisión, como ya hemos visto, ninguno de los dos supondrá ninguna ventaja.

Más información

Para una discusión de todos los pre-procesadores de escalado presentes en Scikit-Learn, véase el artículo Compare the effect of different scalers on data with outliers.

Características polinómicas

La creación de características polifónicas es una técnica de ingeniería de características que permite añadir más complejidad a un dataset numérico mediante la combinación de sus características. Como tal, no es tampoco objeto del temario de nuestras clases, pero os incluyo un poco más de información para que sepáis mejor lo que estáis haciendo en el curso.

La técnica se usa frecuentemente cuando queremos señalar que existe una relación no lineal entre las características (features) y el objetivo (labels), o cuando sospechamos que el efecto de una característica sobre el objetivo depende a su vez de otra.

Scikit-Learn dispone de un preprocesador (`PolynomialFeatures`) preparado para esto. Durante el día 1 de clase, cuando ampliamos el dataset de Boston Housing, lo estamos usando (consultar el código para `load_extended_boston` para más detalle).

PolynomialFeatures cuenta con un parámetro importante llamado degree. Este parámetro determina el grado máximo del polinomio en el que vamos a transformar las características de entrada. Por ejemplo, con dos características de entrada, x1 y x2, creará un polinomio 1, x1, x2, x1^2, x1x2 , x2^2.

Lo que esto nos permite es llevar las características a un espacio de mayor dimensión, donde podemos aplicar entonces una predicción lineal para conseguir un mejor ajuste de la tendencia de los datos que una simple recta.

En el cuaderno **`Polynomial_Features.ipynb`** incluido en el repositorio de las clases puede verse más detalle del funcionamiento de la transformación polinómica sobre las características.