Universidad de Granada

Ingeniería Informática

Practica 2: Reconocimiento de personas con V-REP y SVM

Autor: José Antonio Ruiz Millán
email: jantonioruiz@correo.ugr.es
Curso: 2018-2019
Asignatura: Programación Técnica y Científica
28 de diciembre de 2018

Índice

1. Introducción

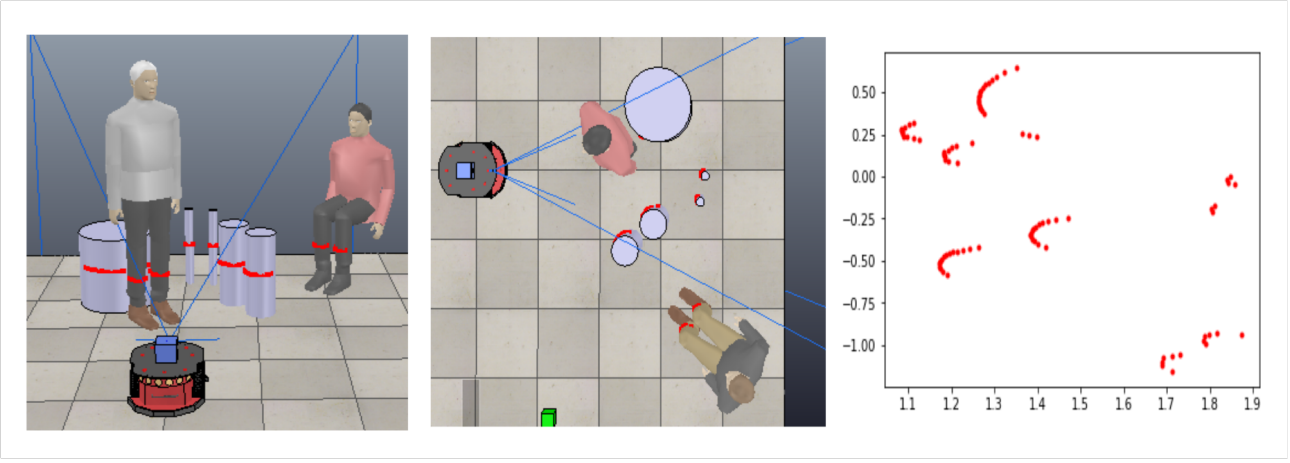
Esta práctica se basa en la deteccíon de personas utilizando el simulador V-REP[1] y el algoritmo SVM[2] para la clasificación de los mismos. El proceso se basa en obtener unos datos de entrenamiento utilizando el simulador basándose en la simulación de una persona sentada, un persona de pie y utilizando cilindros de distinto tamaño para obtener datos de rechazo. Una vez se obtengan los datos, se crean clusters donde consideraremos cada uno de los clusters como posible pierna, que en nuestro caso sabremos si lo es o no porque tenemos los datos. Una vez estén creados los clusters, se crean una serie de características para cada uno de los clusters y así tendremos características para los clusters de piernas y para lo clusters de no piernas. Una vez hecho esto, entrenamos un modelo con el algoritmo SVM que posteriormente utilizaremos para predecir si lo que detecta el laser es o no una pierna.

2. Simulación utilizada

Para el entrenamiento he seguido los pasos que se indican en el guión y por lo tanto he utilizado la simulación de ejemplo que nos pasan en la práctica, haciendo traslaciones y rotaciones de los objetos.

Para comprobar los resultados finales, he utilizado la escena simulada *simulacion-jp2.ttt* que tendremos en el zip donde se encuentran todos los ficheros.

La escena **es la siguiente**:



Donde podemos ver a dos personas algo rotadas, una sentada y otra de pie, podemos también ver un cilindo grande detrás de la persona de pie y luego dos "simulaciones" de piernas con cilindros algo mas anchos y mas estrechos que las piernas de una persona. La segunda imagen es el mismo escenario pero visto desde arriba que es como podemos ver los puntos del laser en la última imagen de la derecha.

3. Desarrollo

En esta sección intentaré mostrar el proceso que he seguido para el realizamiento de la práctica.

3.1. Obtención de datos

Como he comentado en apartados anteriores, este proceso es algo laborioso en tiempo y algo difícil de obtener unos resultados que realmente sean buenos debido a la complejidad en el espacio de la simulación y a las diferentes posturas que podemos poner a los objetos. No obstante este proceso consta de:

- Para cada uno de los objetos, hacer 50 medidas del láser obteniendo todos los puntos en el espacio y alamcenándolos en ficheros sabiendo qúe puntos son para cada objeto. Este proceso se realiza a $< 1{,}5$ metros del sensor.
- Lo mismo que en paso anterior pero para distancias $1{,}5 < d < 2{,}5$.
- Por último, realizar lo mismo que anteriormente pero para distancias $2{,}5 < d < 3{,}5$

Con esto tenemos puntos para cada uno de los objetos en distintas posiciones y distintas ubicaciones.

Para ello, **he creado un fichero en Python llamado "crearDatosLaser.py"** que se encarga de realizar este proceso.

3.2. Creación de los clusters

Despúes de tener todos los puntos correspondientes a cada fichero, tenemos que crear los distintos clusters siguiendo los pasos que se indican en el guión.

En mi caso, **he creado un fichero llamado "crearClusters.py"** que se encarga de realizar todo este proceso.

3.3. Calculo de características para cada cluster

Una vez tenemos todos los clusters creados, tenemos que obtener las características de "perímetro", "profundidad" y "anchura" para cada uno de los clusters y crear de nuevo un fichero con estos valores para cada cluster.

En mi caso, **he creado un fichero llamado "caracteristicasClusters.py"** que se encarga de realizar este proceso.

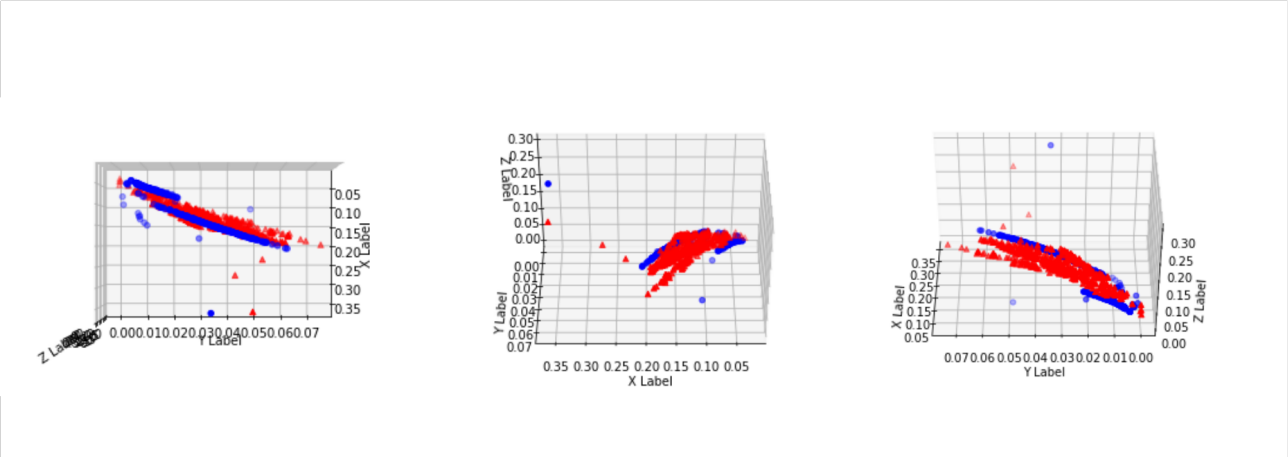
3.4. Entrenamiento del modelo

Para este apartado, tenemos que entrenar un modelo SVM que sea capaz de predecir si unos puntos en el espacio son de una pierna o no.

Para ello se utilizan los ficheros de características anteriores donde cada una de las clases está indicada para cada conjunto de características.

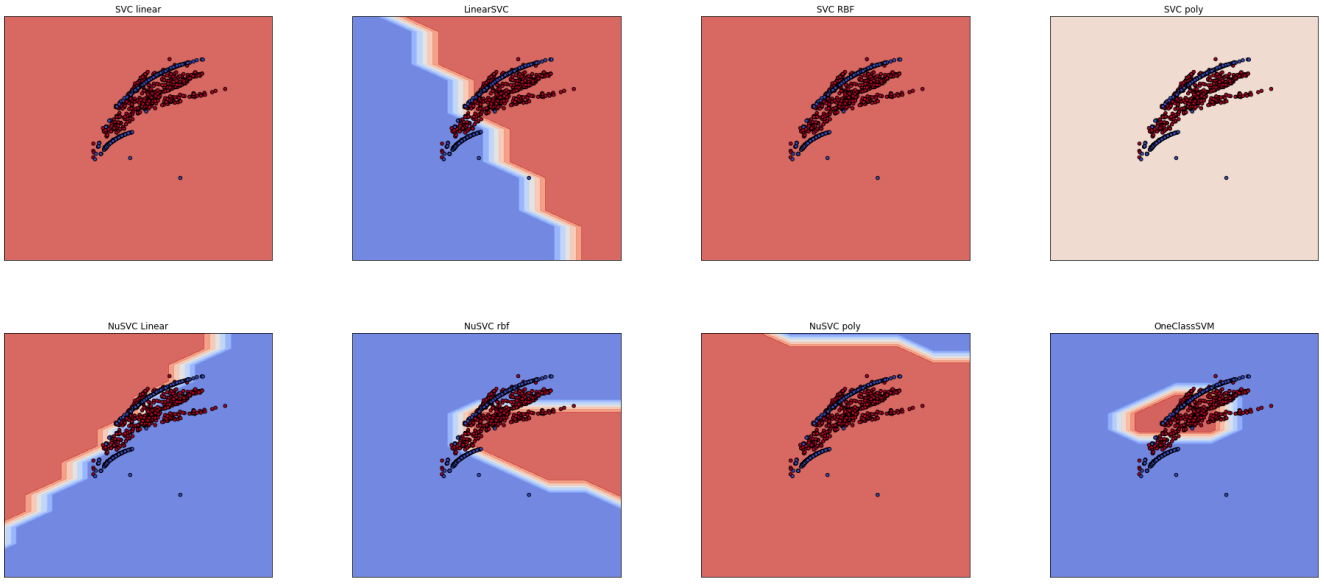
Para realizar este proceso, **he creado un fichero llamado "SVM.py"** donde entreno el modelo y hago un ajuste simple de parámetros para la mejora del modelo. La elección del modelo fué algo laboriosa ya que el SVC no obtenía buenos resultados y tuve que investigar un poco el los datos.

En primer lugar visualicé los datos que tenemos, ya que al tener sólo 3 características, podemos visualizarlos, obteniendo:



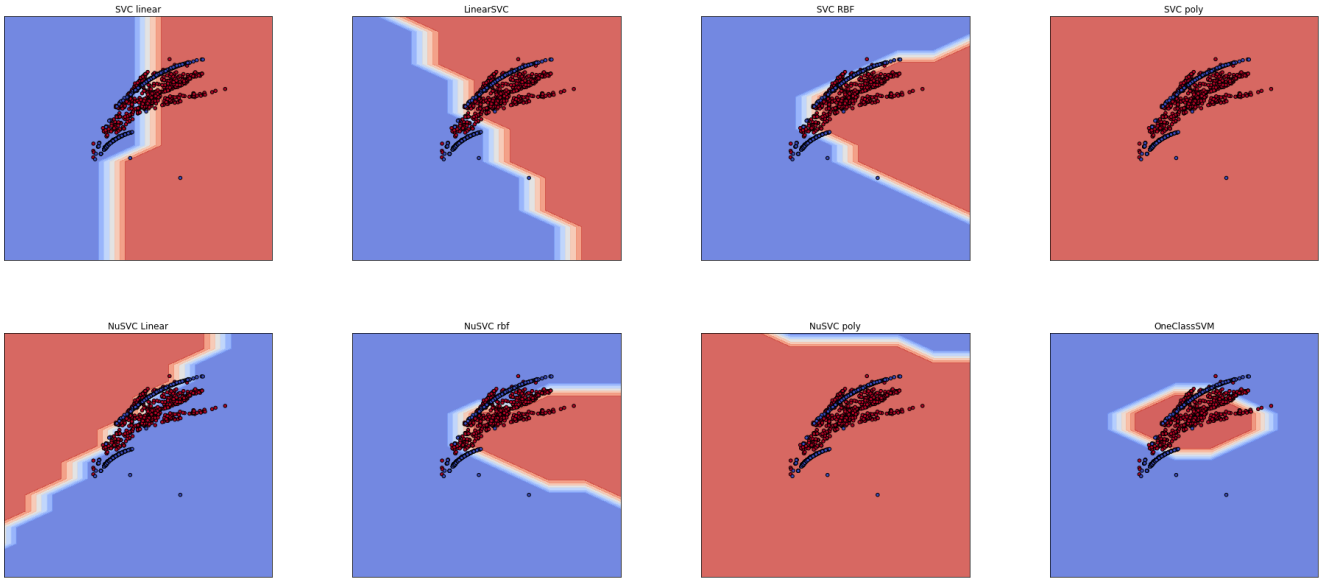
Como podemos ver, tenemos como tres zonas claramente separables ya que lo que son piernas (rojo) están en el centro y los que no son piernas (azul) están en los lados.

Para poder visualizar los distintos modelos, tuve que reducir el espacio aunque luego lo aplicase con las 3 características. Lo que hice fué visualizar la matriz de correlación viendo que entre "perímetro" y "anchura" había una correlación muy grande (0.98) y por lo tanto decidí eliminar "perímetro" **sólo para la visualización**.



Como podemos ver en el resultado, obtenemos unos ajustes un poco malos a excepción del **NuSVC**. Esto se debe a que estoy utilizando los parámetro por defecto para cada uno de los algoritmos,

modificando esto parámetros conseguí mejorar algo el ajuste, obteniendo:



Como podemos ver hemos mejorado el ajuste en algunos de estos algoritmos, no obstante, utilizando métricas para comparar estos algoritmo obtuve que el mejor que clasificaba era **NuSVC** con los siguientes resultados:

	V	**F**
F	100	28
V	33	114

Tabla 1: Confusion Matrix NuSVC

	Precision	**Recall**	**F1-score**	**Support**
F	0.75	0.78	0.77	128
V	0.8	0.78	0.79	147
Average	0.78	0.78	0.78	275

Tabla 2: Métricas NuSVC

Y obteniendo de *Accuracy 5-cross validation* = **0.79**,

y *Accuracy 5-cross validation (test 0.25)* = **0.79**

4. Modo de uso

Una vez tenemos ya todo lo anterior, sólo nos queda ponerlo en funcionamiento, por lo que para ello **he creado el fichero llamado "main.py"** que se encarga de:

- Recibir los datos de laser
- Convertirlos en clústeres
- Generar las tres características de cada cluster

- Utilizar el predictor para cada cluster a partir de sus características
- Dibujar en color rojo aquellos clústeres que sean piernas según el predictor y en azul aquellos que no.

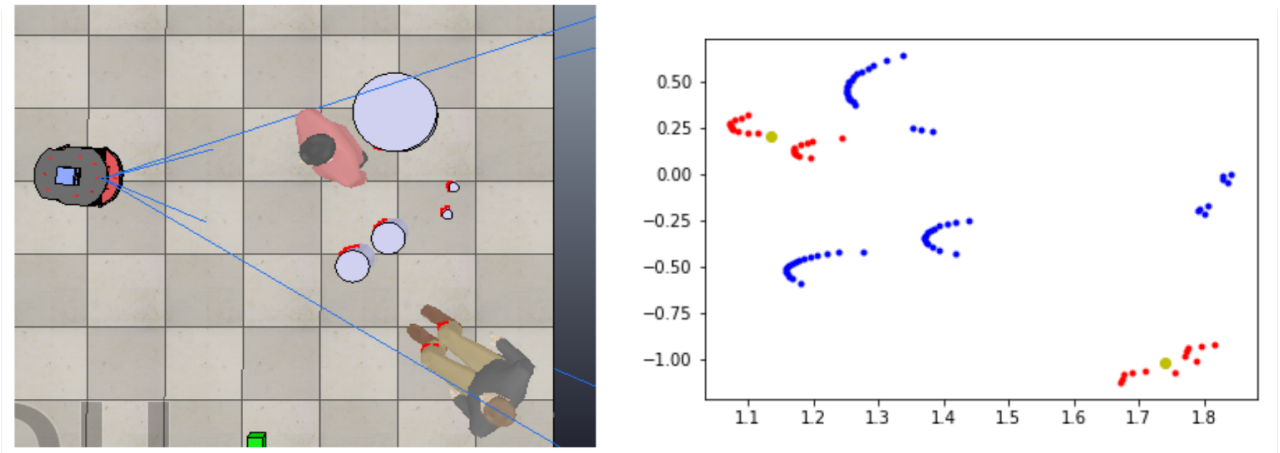
Todos los métodos utilizados en los ficheros anteriores, están bien organizados y explicados en este fichero.

Una vez ejecutado este fichero, empezamos ha recibir los datos del láser, ha organizarlos en clústeres y ha crear las disitintas características para cada uno de ellos. Una vez tenemos esto, el clasificador se encarga de determinar si es pierna o no.

Después de un intenso estudio sobre los parámetros, llegué a la conclusión de que los mejores valores son:

- **Número mínimo de nodos = 3**
- **Número máximo de nodos = 20**
- **Umbral distancia puntos = 0.06**
- **Umbral distancia persona = 0.2**

Obteniendo los siguientes resultados:



5. Conclusiones

Despúes de un proceso laborioso de obtencion de datos que implica una gran parte del tiempo, necesitamos que estos datos sean de calidad ya que si no, es imposible que el modelo final sea capaz de crear un buen ajuste, por ejemplo en mi caso, hay partes en rotaciones que no sabe interpretar por falta de elementos en el entrenamiento ya que como digo es un proceso muy laborioso.

Para fijar los parámetros, estube probando varios fijandome en cuantos puntos están definidos las personas y los cilindros, utilizando en una primera vez la mitad de los cilindros mayores para el máximo y la mitad de los cilindros menores para el mínimo. Sin embargo, cuando probé lo mismo

pero utilizando los datos de las piernas, obtuve en la práctica un mejor resultado ya que haciendo distintas pruebas comprobé que era más efectivo. Finalmente acabé poniendo los parámetros indicados en los apartados anteriores.

Finalmente, el modelo acaba haciendo un ajuste bastante bueno teniendo un acierto del 80 %, lo que hace que ajuste bastante bien. No obstante, me he percatado de que es algo sensible a las rotaciones y si el humano esta situado a 90º de la camara le cuesta decir que es un humano, pero por lo general lo hace bastante bien.

Referencias

[1] Simulador V-REP, coppeliarobotics.com, `http://www.coppeliarobotics.com/downloads.html`, Accedido el 28 de diciembre de 2018.

[2] Sklearn references, scikit-learn.org, `https://scikit-learn.org/stable/modules/svm.html`, Accedido el 28 de diciembre de 2018.