Práctica 2 PTC

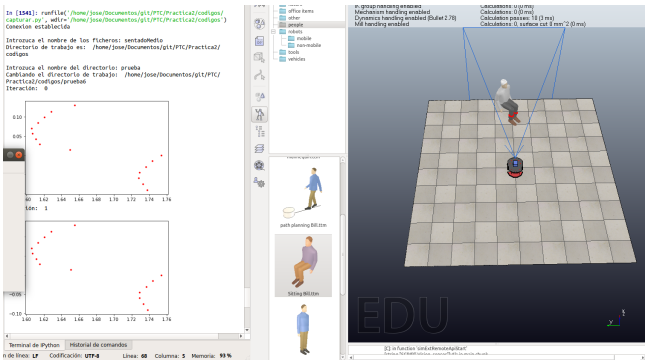
José Manuel Pérez Lendínez, 26051613-l

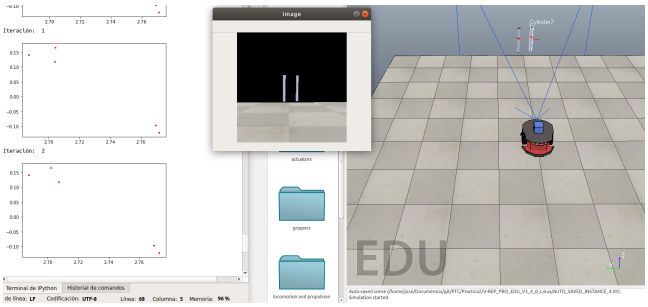
January 6, 2020

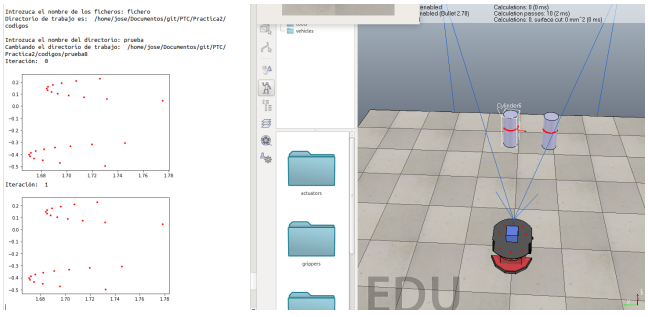
Contents

1 Captura de datos

La captura de datos la he realizado utilizando la escena de ejemplo de láser dada por el profesor. Para esto y puesto que el suelo no tenia la suficiente superficie necesaria para la toma de datos pedida, he quitado las físicas de los objetos de forma que no caigan al vació. La falta de suele en ciertas partes no nos dará problemas porque el láser no capta el suelo al estar mas elevado.







Los cilindros pequeños tienen una anchura de 0.02 y los grandes de 0.5

Para controlar la distancia del objeto al centro, podremos utilizar las variables que se muestran en la equina superior izquierda de la ventana de simu-

lación, que nos da la posición de x e y para el objeto selecciona.

Al ejecutar el script de captación nos pedirá primero el nombre que utilizaremos para el fichero json y las imágenes que guardemos y a continuación nos pedirá también el nombre de la carpeta en el que queremos que sean guardados, añadiéndole el programa el numero que le corresponda si tenemos mas con este nombre. He elegido como mínimo 3, máximo de 20 y distancia máxima valida de 0.05 para tomar los cluster.

2 SVM fase de aprendizaje

Para la fase de aprendizaje tendremos dos parte.

1. **Fase con modelos simples:**En este apartado se probaran los tres tipos de svm dados en el ejemplo del profesor sin cambiar sus parámetros, con esto descartaremos al que peor resultados nos de. Los datos son los siguientes:

```
Valores por defecto


------------------------------------------------------------
Kernel linear
------------------------------------------------------------

Acc_test: (TP+TN)/(T+P)  0.8551
Matriz de confusion Filas: verdad Columnas: prediccion
[[ 62  30]
[  0 115]]
Precision= TP / (TP + FP), Recall= TP / (TP + FN)
f1-score es la media entre precision y recall
precision    recall  f1-score   support

0       1.00      0.67      0.81        92
1       0.79      1.00      0.88       115

accuracy                           0.86       207
macro avg       0.90      0.84      0.84       207
weighted avg       0.89      0.86      0.85       207

Accuracy 5-cross validation: 0.8182 (+/- 0.0195)







------------------------------------------------------------
Kernel Polinomico
------------------------------------------------------------

Clasificacion con kernek polinomico de grado  2
Acc_test: (TP+TN)/(T+P)  0.5556
Matriz de confusion Filas: verdad Columnas: prediccion
[[  0  92]
[  0 115]]
Precision= TP / (TP + FP), Recall= TP / (TP + FN)
f1-score es la media entre precision y recall
precision    recall  f1-score   support
```

```
0       0.00      0.00      0.00        92
1       0.56      1.00      0.71       115

accuracy                           0.56       207
macro avg       0.28      0.50      0.36       207
weighted avg       0.31      0.56      0.40       207

Accuracy 5-cross validation: 0.5774 (+/- 0.0007)


------------------------------------------------------------
Kernel rbf
------------------------------------------------------------

Acc_test: (TP+TN)/(T+P)  0.6618
Matriz de confusion Filas: verdad Columnas: prediccion
[[ 22  70]
[  0 115]]
Precision= TP / (TP + FP), Recall= TP / (TP + FN)
f1-score es la media entre precision y recall
precision    recall  f1-score   support

0       1.00      0.24      0.39        92
1       0.62      1.00      0.77       115

accuracy                           0.66       207
macro avg       0.81      0.62      0.58       207
weighted avg       0.79      0.66      0.60       207

/home/jose/anaconda3/lib/python3.7/site-packages/sklearn/
     metrics/classification.py:1437: UndefinedMetricWarning:
     Precision and F-score are ill-defined and being set to 0.0
     in labels with no predicted samples.
'precision', 'predicted', average, warn_for)
Accuracy 5-cross validation: 0.7572 (+/- 0.0285)
```

Como se ve con nuestros datos el peor kernel a sido el polinomio, quedando muy atrás de el resto. En este caso se uso grado 2. Con grado 3 y 4 mejoran un poco pero tampoco se acercan.

2. **Comparar los dos ganadores:** Para comparar los dos siguientes vamos a realizar mejora a ambos mediante gridSearchCv. Le pasaremos los parámetros que queremos que pruebe para nuestro modelo, el modelo y obtendremos el mejor modelo para dichos parámetros. Cambiaremos el parámetro C(es la penalización por clasificar erróneamente), que le daremos los valores [0.1 ,1, 20 ,30, 10, 100, 1000] y el parámetro gamma(expansión que puede tener el kernel) los valores [0.1,0.5,1,2,3,7,10]. Esto nos devuelve los mejores modelos para estas opciones que nos dan los siguientes resultados:

```
------------------------------------------------------------
Mejoras para linear
------------------------------------------------------------

Acc_test: (TP+TN)/(T+P)  0.8937
Matriz de confusion Filas: verdad Columnas: prediccion
[[ 72  20]
[  2 113]]
Precision= TP / (TP + FP), Recall= TP / (TP + FN)
f1-score es la media entre precision y recall
precision    recall  f1-score   support
```

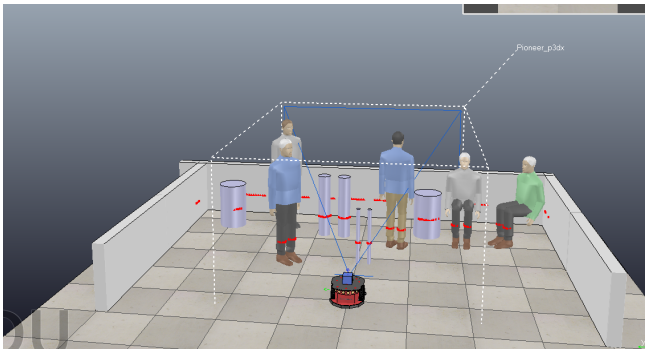
```
0       0.97      0.78      0.87        92
1       0.85      0.98      0.91       115

accuracy                           0.89       207
macro avg       0.91      0.88      0.89       207
weighted avg       0.90      0.89      0.89       207

Accuracy 5-cross validation: 0.8569 (+/- 0.0277)

-----------------------------------------------------------------
Mejoras para rbf
-----------------------------------------------------------------

Acc_test: (TP+TN)/(T+P)  0.9758
Matriz de confusion Filas: verdad Columnas: prediccin
[[ 90   2]
[  3 112]]
Precision= TP / (TP + FP), Recall= TP / (TP + FN)
f1-score es la media entre precision y recall
precision    recall  f1-score   support

0       0.97      0.98      0.97        92
1       0.98      0.97      0.98       115

accuracy                           0.98       207
macro avg       0.98      0.98      0.98       207
weighted avg       0.98      0.98      0.98       207

Accuracy 5-cross validation: 0.9787 (+/- 0.0158)
```

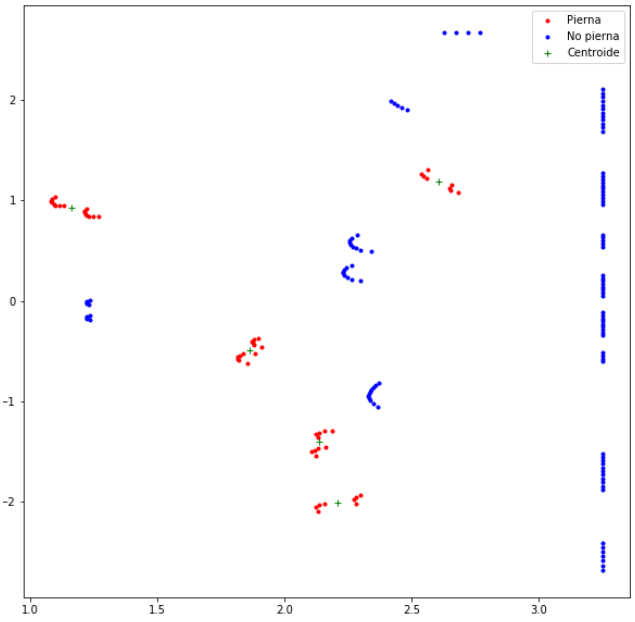
Con estos resultados nos decantaremos por el kernel rbf puesto que obtiene casi un 10% mas en validación cruzada que el linear. El parámetro c seleccionado fue 1000 y el gamma selecciono 10.

3 Ejecutar el modelo en escena de test.

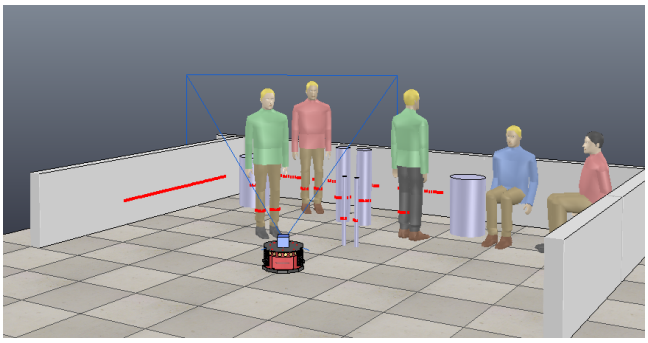
Realizaremos un par de ejecuciones, una con la posición inicial del robot y moviendo un poco el robot para ver que sigue acertando. Marcaremos en rojo las piernas y azul lo que no pertenezca a una pierna. Los centroides entre dos cluster los marcaremos con una cruz verde. Para los centroides he marcado una distancia minima de 0.7 para considerar que esta cerca.

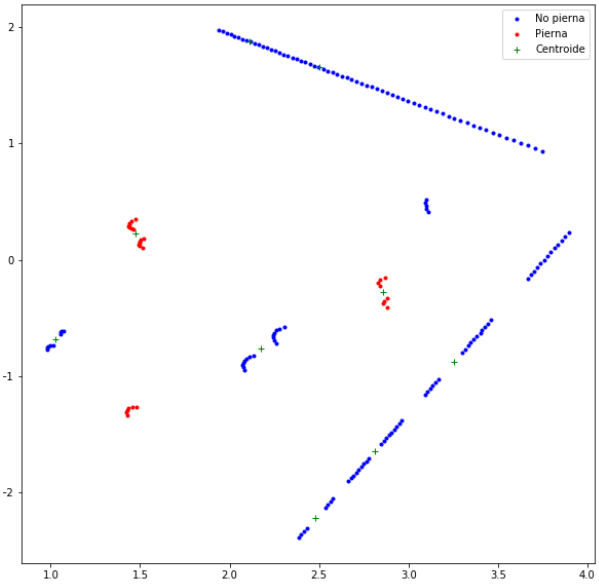
Vamos a ver el primer ejemplo.



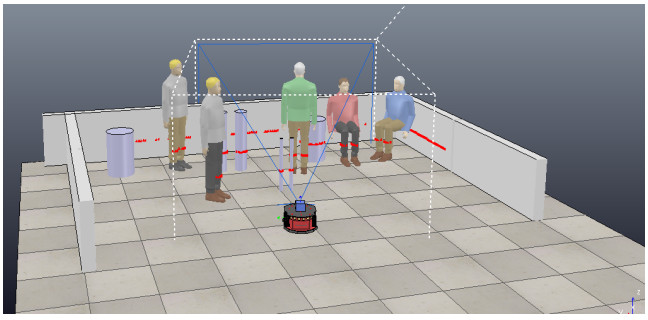


Veamos un par de ejemplo rotando el robot.





Para el siguiente ejemplo rotare el robot en el otro sentido.



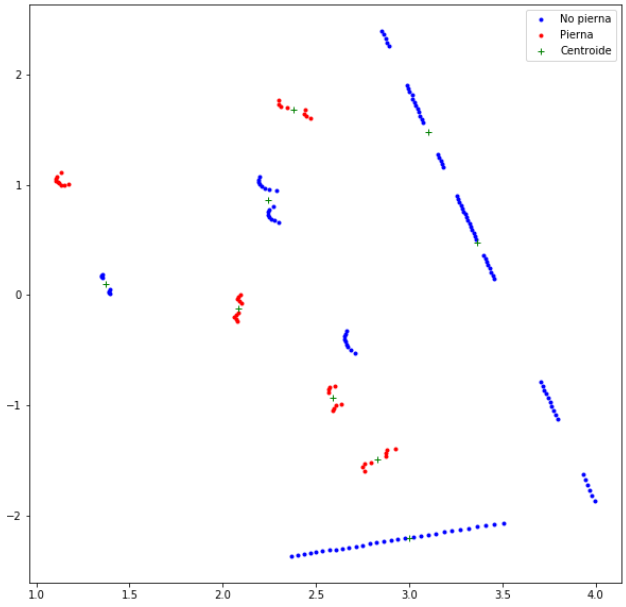


Figure 1