Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ingeniería Mecánica

Robótica

Tarea 3

Autor: Joaquín Aliaga González
Profesor de cátedra: Juan C. Zagal
Ayudantes: Bastián Fuenza
Diego Hidalgo

31 de Octubre de 2017
Santiago, Chile

Configuraciones previas

Esta tarea está diseñada para ser desarrollada bajo el lenguaje de programación Python ®, dada la gran cantidad de módulos open-source disponibles en Internet.

Por lo tanto primero es necesario instalar Python, el cual se descarga directamente desde la página de los creadores (`https://www.python.org/downloads/release/python-2714/`). Cabe destacar que **los códigos de ejemplo de esta tarea están escritos en la version 2.7.14 de Python**, por tanto puede que con versiones más recientes existan errores de compilación, por lo que se **recomienda** utilizar dicha versión

Una vez instalado Python, es necesario instalar una serie de módulos o paquetes. Para esto se hace lo siguiente:

1. Abrir "Símbolo del sistema": En el buscador de Windows escribir "cmd". Se recomienda abrir como administrador.
2. Escribir el comando "cd .." (con los dos puntos incluidos) hasta llegar al directorio C: (disco local, en el cual se debió haber instalado Python por defecto).
3. Instalar o actualizar pip, escribiendo: pip install - -upgrade pip
4. Instalar el módulo "ejemplo", escribiendo: pip install ejemplo

Los módulos necesarios para el desarrollo de esta tarea son:

- scikit-learn : Por tanto para instalarlo se debe escribir en la ventana de comandos : pip install scikit-learn. También puede ser instalado manualmente desde `http://scikit-learn.org/stable/index.html#`
- numpy (`http://www.numpy.org/`)
- matplotlib (`http://matplotlib.org/`)
- pydot (`https://pypi.python.org/pypi/pydot`)
- scipy (`http://scipy.org/`)

Es importante destacar que esta tarea está inspirada en un set de videos de Google Developers®, los cuales sirven bastante como material de apoyo a este documento. Para encontrar los videos basta con buscar "Google Developers Machine Learning" en Youtube ®

Índice de Contenidos

1. Clasificadores

Un problema común en robótica es la clasificación de objetos. Por ejemplo, un robot de servicio de hogar debería ser capaz de distinguir entre cucharas, cuchillos y tenedores para poder separarlos y ordenarlos en un cajón.

Para simplificar el problema piensa en que tienes fotos de frutas y necesitas escribir un programa que tome las fotos y distinga entre duraznos y manzanas. Podrías escribir un código con distintas reglas de diferenciación como peso, textura, color, volumen etc.

El problema que inmediatamente surge de esta solución es que la realidad es muy variable y podrías tener manzanas que tengan un color parecido al de un durazno o un durazno con una textura y peso similar a una manzana o podrías tener un conjunto de frutas en que no hayan ni manzanas ni naranjas. Por último, si escribieras un código lo suficientemente bueno para resolver este problema y te pidieran distinguir entre duraznos, manzanas, plátanos y sandías, tendrías que reescribir todo el código y hacerlo aún más complicado.

Una forma de solucionar esto es mediante un método llamado Aprendizaje Supervisado, el cual crea una función que, a través de aprendizaje, establece reglas de diferenciación por si misma. Dicha función se llama clasificador y su misión, en este caso, es tomar datos de frutas y clasificarlas en manzanas o duraznos.

Cabe destacar que existen varios tipos de clasificadores como “Árboles de decisión”,“KNeighbors”, “Redes Neuronales”,etc. Para el desarrollo de esta tarea se usarán Redes Neuronales MLP (Multilayer Perceptron), que consisten básicamente en redes neuronales de al menos 3 capas que ocupan retropropagación en su proceso de aprendizaje.

El aprendizaje supervisado se implementa mediante la siguiente metodología:

1. Recolectar datos de entrenamiento: Imagina que vas a la feria y anotas el peso y la textura de 4 manzanas y 4 duraznos, obteniendo los datos de la tabla mostrada en la Figura 1.1.

2. Crear y entrenar al clasificador: Usando una parte de los datos que has recolectado y usando la función “fit” del módulo **sklearn**, debes “entrenar” al clasificador. El proceso de entrenamiento fue revisado en clases.

3. Probar al clasificador: Usando la parte restante de los datos recolectados, debes poner a prueba al clasificador ingresando las características, en este caso peso y textura, y ver si los resultados que entrega el clasificador son o no correctos.

Nota: Al momento de escribir el programa, es importante codificar las etiquetas. Para el caso anterior, podrías usar la codificación manzana $= 0$ y durazno $= 1$

Fruta	Peso (g)	Textura
Manzana	120	Suave
Durazno	170	Aspera
Durazno	180	Aspera
Manzana	140	Suave
Manzana	150	Suave
Durazno	200	Suave

Figura 1.1: Datos recolectados de frutas

1.1. Pregunta 1

A. Usando el módulo **sklearn** genere un clasificador que distinga entre bicicletas de pista y mountain bikes. Dicho clasificador debe ser una Red Neuronal MLP. Considere que los datos recolectados son los siguientes (Figura 1.2)

Bicicleta	Velocidad Max (km/hr)	Peso (kg)
Pistera	60	1
Mountain Bike	30	2.5
Pistera	70	1.5
Mountain Bike	48	2
Mountain Bike	55	1.5
Pistera	58	2

Figura 1.2: Datos recolectados de bicicletas

B. Considera que tienes los siguientes datos de bicicletas (Tabla 1.3).

Velocidad Max (km/hr)	Peso (kg)
57	1.5
60	1.7

Figura 1.3: Datos recolectados sin clasificar

i. ¿Cuál es la predicción del clasificador para los datos?.

ii. ¿Implementarías el clasificador que creaste en una fábrica de bicicletas?,¿Crees que necesita ser mejorado?,¿Cómo lo mejorarías? Fundamenta.

Nota: El código adjunto “ejemploP1” puede servir de guía.

1.2. Pregunta 2

Con los ejemplos anteriores te puedes dar cuenta de que las predicciones que entregan los clasificadores pueden no ser lo suficientemente confiables o muy arbitrarias. Una de las causas de esto es que la cantidad de datos de entrenamiento fue muy pequeña.

En problemas reales, los clasificadores son entrenados con cientos, miles o millones de datos para hacerlos certeros y confiables. En Internet existen distintos sets de datos tomados de muestras reales y están a libre disposición para poder entrenar a tus propios clasificadores.

Un set de datos bastante conocido es "iris", que contiene una gran cantidad de datos asociados a 3 especies distintas de la flor iris: *setosa, versicolor y virginica.*

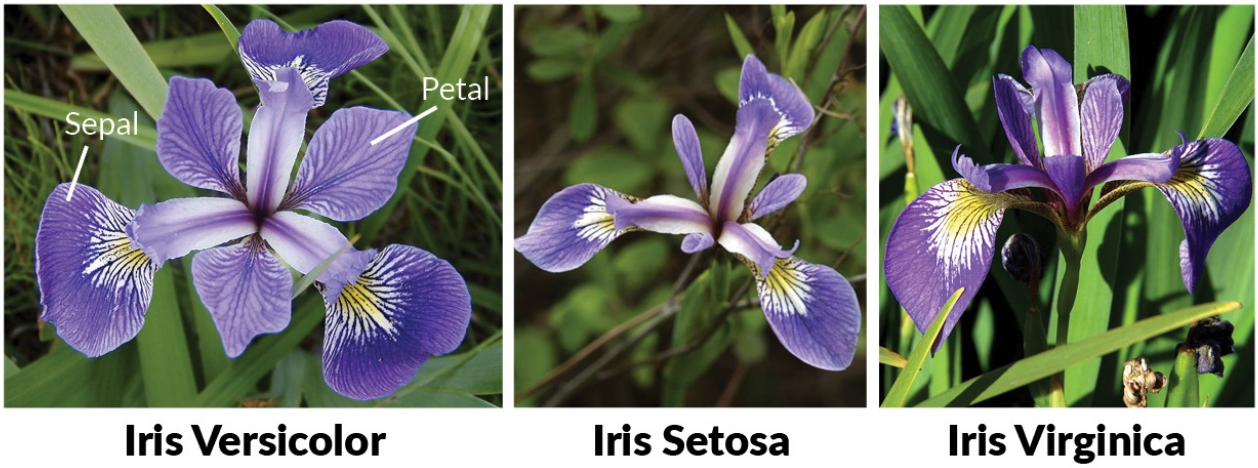

Figura 1.4: Especie de flor iris

El módulo scikit-learn contiene distintos sets de datos para aprendizaje de máquinas, entre ellos está el dataset iris.

Para importar y cargar los datos se usa el siguiente código:

```
from sklearn.datasets import load_iris
iris=load_iris()
```

Puedes ver qué datos contiene el dataset y cuales son las etiquetas usadas con el siguiente código:

```
print iris.feature_names
print iris.target_names
```

Y con el siguiente código puedes visualizar los datos correspondientes a la primera fila y su respectiva etiqueta:

```
print iris.feature_names
print iris.target_names
print iris.data[0]
print iris.target[0]
```

Nota: Al igual que en los ejemplos anteriores, se ha usado una codificación para las etiquetas. En este caso el valor 0 corresponde a la especie *setosa*

Puedes comprobar que los datos que obtuviste son correctos comparando con la tabla de datos de este link : `https://es.wikipedia.org/wiki/Iris_flor_conjunto_de_datos`

Al tener una gran cantidad de datos es bastante útil separarlos en dos grupos: datos de entrenamiento y datos de prueba. Una forma de hacer esto es elegir manualmente cuales datos pertenecerán a cierto grupo, pero para una cantidad de datos considerable (>500) esto puede ser muy tedioso. Por suerte para ti, el módulo sklearn contiene una función para dividir los datos. El código para utilizar dicha función es el siguiente (aplicado al ejemplo de iris).

```
X = iris.data
y= iris.target
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test (X,y,test_size = .3)
```

Dicho código generará datos de entrenamiento (X_train,y_train) y datos de prueba (X_test,y_test) tomados de forma aleatoria del conjunto total de datos. Además se puede indicar el porcentaje de datos usados para pruebas con el parámetro test_size, que en este caso indica un 30 % (70 % de los datos serán usados para entrenamiento).

A. Carga el dataset "load_wine" incluido en el módulo sklearn.

B. Determina cuales son los datos y las etiquetas que contiene el dataset. Además determina la codificación usada para representar a las etiquetas

C. Genera una partición de los datos, tomando un 50 % de ellos para entrenamiento y el otro 50 % para prueba.

Los datos de este dataset en específico no tienen una escala común, por tanto al entrenar a la red neuronal podrían ocurrir errores graves que afectarían el desempeño de la misma

Para llevar todos los datos a la misma escala usamos la herramienta **StandardScaler** incluida en sklearn, con el código siguiente:

```
from sklearn.preprocessing import StandardScaler

scaler=StandardScaler()
scaler.fit(x_train)
x_train=scaler.transform(x_train)
x_test=scaler.transform(x_test)
```

D. Crea un clasificador tipo "Red Neuronal MLP" y entrénala con los datos de entrenamiento.

E. Toma cualquier fila de datos de la tabla y prueba el clasificador que creaste. ¿La predicción del clasificador fue correcta?. Si es así, ¿Consideras confiable el clasificador creado? Fundamenta.

Una forma de medir que tan bueno es el clasificador creado es obteniendo su porcentaje de precisión. Esta métrica viene incorporada en sklearn y se implementa con el siguiente código

```
predicciones=clasificador.predict(X_test)
from sklearn.metrics import accuracy_score
print accuracy_score (y_test, predicciones)
```

F. ¿Cual es el porcentaje de precisión del clasificador que creaste? ¿Cómo afecta la cantidad de datos de entrenamiento en el porcentaje de precisión?.

G. Genera un gráfico del porcentaje de precisión del clasificador en función de la cantidad de datos de aprendizaje. Has un análisis del gráfico que obtuviste.

Nota: El código adjunto "ejemploP2" puede servir de guía.

1.3. Pregunta 3

Usando cualquier base de datos (excluyendo iris y wine), ya sea de los incorporados en sklearn, de los disponibles en Internet o de alguna otra fuente (puedes crear una), repite el procedimiento de la Pregunta 2. (no olvides incorporar en tu informe la bibliografia correspondiente a los datasets usados).