Trabajo Final
Inteligencia Artificial I

Año 2017

Franco Palau

Legajo N° 11194

Índice

Resumen

Este trabajo final consiste en el diseño e implementación de un agente de aprendizaje para su posterior utilización en un robot encargado de pintar piezas mecánicas. Para lograr esto, el agente recibe por comandos de voz los tres colores a pintar: Rojo, Verde y Amarillo. En estos casos el agente debe saber diferenciar cada palabra para poder pintar la pieza del color correspondiente. Para esto, se utilizaron dos algoritmos de aprendizaje, Kmeans y K-NN, donde luego se evaluó el rendimiento de cada uno.

Para aplicar los algoritmos se tomaron los valores de la transformada inversa de Fourier del espectro de la señal sonora, también denominados "Cepstrums". La implementación de los algoritmos se realizó en MatLab y el prototipo se desarrolló en una placa Arduino Uno.

Introducción

El agente a implementar es del tipo "Agentes que aprenden" ya que a este agente se lo entrenará con la voz del operario y luego aplicando lo aprendido reconocerá la orden a ejecutar (en este caso que color pintar la pieza).

El aprendizaje será no supervisado debido a que el agente solo aprenderá a partir de patrones de entrada sin considerar sus valores de salida.

A continuación, se muestra la tabla REAS del agente, esta considera el rendimiento, el entorno, los actuadores y los sensores.

Agente	Rendimiento	Entorno	Actuadores	Sensores
Robot Industrial con capacidad de pintar piezas mecánicas por comando de voz	Maximizar la cantidad de piezas pintadas correctamente	Industria	Pistola neumática de pintura	Sensor de sonido, de posición e infrarrojo

Las propiedades del entorno de trabajo son las siguientes:

Entorno	Observable	Determinista	Episódico	Estático	Discreto
Industrial	Parcialmente Observable	Si	Si	No	Si

Durante el desarrollo del proyecto, se buscó simular mediante algoritmos de aprendizaje y un prototipo, al agente explicitado anteriormente.

Diseño del Sistema

El diseño del agente está compuesto por una parte de software, en la que se implementan los algoritmos de control y otra de hardware, donde se puede visualizar los resultados.

La implementación del software se realizó en Matlab y se subdividió a este en distintas funciones con tareas específicas cada una. Se implementaron dos funciones una encargada de aplicar el algoritmo Kmeans y otra

encargada de K-nn. Además, se crearon funciones para la grabación de voz, para el procesamiento de las grabaciones de voz, para la visualización de las grabaciones y para el control de la placa Arduino. Matlab permite descargar un Toolbox que permite manejar la placa Arduino desde el mismo sin tener que recurrir a otros programas.

Para el diseño de los algoritmos de aprendizaje se recurrió a los métodos brindados por Matlab que consisten en la siguiente función para utilizar K-means:

```
[idx,C] = kmeans(___)
```

Esta devuelve la ubicación de los centroides de los k clusters en una matriz C.

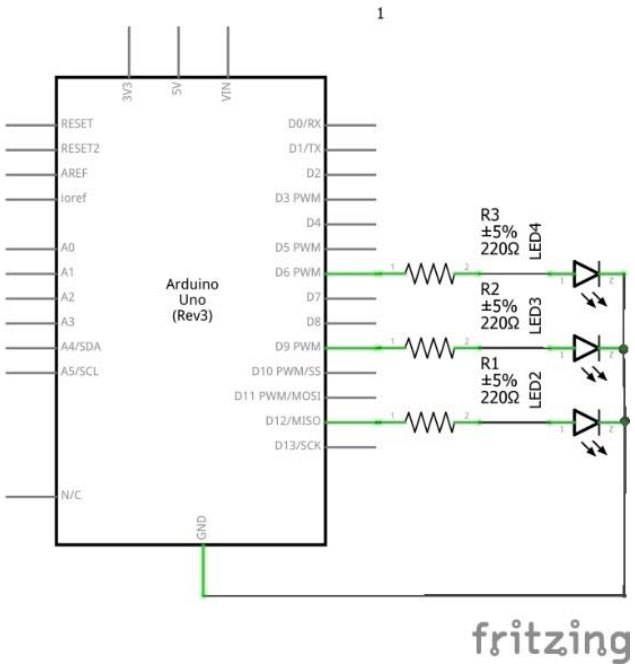
Las otras funciónes correspondiente a K-nn son:

```
Mdl = fitcknn(X,Y)
```

Esta función devuelve el modelo de clasificación según K-nearest neighbor basado en la información X y la respuesta Y.

```
Class = predict(Mdl,audioActual)
```

Esta función toma como parámetro el modelo de clasificación y la información a clasificar. Devuelve a que clase corresponde la información.
En el caso del hardware, el circuito a implementar fue el siguiente:



Código

El siguiente código es la implementación de la solución en Matlab, en este se pueden apreciar las distintas funciones con sus tareas dedicadas.

```
function trabajoFinal
    contKmeans=0;
    contKNN=0;
    errKmeans=0;
    errKNN=0;
    activado=true;
    opciones = input('1_Grabar nueva base de datos \n2_Usar base de datos
existente \n');
    if (opciones == 1)
        grabarBaseDatos;
        AudiosProcesados=procesarBaseDeDatos;
        DisplayBaseDeDatos(AudiosProcesados);
    elseif (opciones == 2)
        AudiosProcesados=procesarBaseDeDatos;
        DisplayBaseDeDatos(AudiosProcesados);
    end

    while (activado)
        opciones2 = input('0_Salir \n1_ K-MEANS \n2_ K-NN\n');
        if (opciones2 == 0)
            activado = false;
        elseif (opciones2 == 1)
            disp('K-Means');
            contKmeans = contKmeans + 1;
            grabarAudioActual;
            audioActualProcesado=procesarAudioActual;
            I=myKmeans(audioActualProcesado,AudiosProcesados);
            arduinoControl(I);
            Error=input('1-Lectura correcta \n2-Lectura incorrecta \n');
            if (Error == 2)
                errKmeans=errKmeans+1;
            end
        elseif (opciones2 == 2)
            disp('K-NN');
            contKNN = contKNN + 1;
            grabarAudioActual;
            audioActualProcesado=procesarAudioActual;
            I=knn(audioActualProcesado,AudiosProcesados);
            arduinoControl(I);
            Error=input('1-Lectura correcta \n2-Lectura incorrecta \n');
            if (Error == 2)
                errKNN=errKNN+1;
            end
        end
    end

    fprintf('Porcentaje de Error con K-means:
%g\n',(errKmeans*100/contKmeans));
    fprintf('Porcentaje de Error con Knn: %g\n',(errKNN*100/contKNN));
```

```
end

function grabarBaseDatos
    for(i=1:9)
        recObj = audiorecorder(44100,16,2); %objeto recObj,
param(rate,bits,stereo)
        disp(strcat('Comience a hablar ',num2str(i)));
        recordblocking(recObj, 2); %segundo parametro tiempo de grabacion
        disp('Fin de la grabación.');
        % almacena los datos en un arreglo de doble precisión.
        myRecording = getaudiodata(recObj);
        audiowrite(strcat('Audio', num2str(i),
'.wav'),myRecording,44100);
    end
end

function [CRecordings]=procesarBaseDeDatos
     for i=1:9
         Recordings = audioread(strcat('Audio', num2str(i), '.wav'));
         CRecordings(:,i)=rceps(Recordings(:,1));
         CRecordings(:,i)=CRecordings(:,i)./max(CRecordings(:,i));
     end
end

function grabarAudioActual
    % Grabe su voz por 2 segundo
    recObj = audiorecorder(44100,16,2); %objeto recObj,
param(rate,bits,stereo)
    disp('Comience a hablar.')
    recordblocking(recObj, 2); %segundo parametro tiempo de grabacion
    disp('Fin de la grabación.');
    % almacena los datos en un arreglo de doble precisión.
    myRecording = getaudiodata(recObj);
    %se crea un archivo de audio con el arreglo
    audiowrite('AudioActual.wav',myRecording,44100);
end

function [cepstrum]=procesarAudioActual
    myRecording=audioread('AudioActual.wav');
    cepstrum=rceps(myRecording);
    %Normalizamos
    cepstrum(:,1)=cepstrum(:,1)./max(cepstrum(:,1));
    cepstrum(:,2)=cepstrum(:,2)./max(cepstrum(:,2));
end

function DisplayBaseDeDatos(AudiosProcesados)
    figure(1)
    for (i = 1 : 9)
        Recordings = audioread(strcat('Audio', num2str(i), '.wav'));
        subplot(3,3,i)
        if i < 4
            plot(Recordings,'r');
        elseif i >= 4 && i < 7
            plot(Recordings,'g');
        else
```

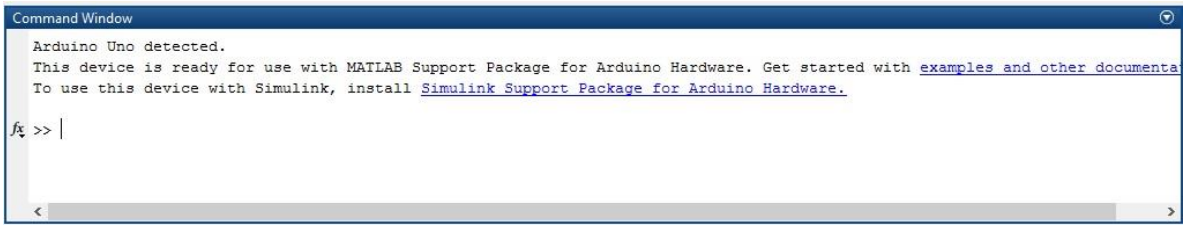
```
            plot(Recordings,'y');
        end
    end
    t=0:length(Recordings)-1;
    figure(2)
    for (i = 1 : 9)
        subplot(3,3,i)
        if i < 4
            plot(t,AudiosProcesados(:,i),'r');
        elseif i >= 4 && i < 7
            plot(t,AudiosProcesados(:,i),'g');
        else
            plot(t,AudiosProcesados(:,i),'y');
        end
    end

    figure(3)
    for (i = 1 : 9)
        subplot(3,3,i)
        if i < 4
            histogram(AudiosProcesados(:,i));
        elseif i >= 4 && i < 7
            histogram(AudiosProcesados(:,i));
        else
            histogram(AudiosProcesados(:,i));
        end
    end
end

function [I]=knn(audioActual,baseDeDatos)
    grupos=[1;1;1;2;2;2;3;3;3];
    Mdl=fitcknn(baseDeDatos',grupos);
    Class=predict(Mdl,audioActual');
    I=Class(1,1);
end

function [I]=myKmeans(audioActual,baseDeDatos)
    for (i=1:3)
        X1(i,:) = baseDeDatos(:,i);   %rojos
        X2(i,:) = baseDeDatos(:,i+3); %verdes
        X3(i,:) = baseDeDatos(:,i+6); %amarillos
    end

    [idx1,C1] = kmeans(X1,1,'Distance','cosine','Start','plus');
    [idx2,C2] = kmeans(X2,1,'Distance','cosine','Start','plus');
    [idx3,C3] = kmeans(X3,1,'Distance','cosine','Start','plus');

    [idxN,CN] =
kmeans(audioActual',1,'Distance','cosine','Start','plus');
    centroides=[C1;C2;C3];

    for (j=1:3)
        for (i=1:88200)
            diferencia(j,i) = abs(centroides(j,i) - CN(1,i));
        end
    end
```

```
    K=sum(diferencia');
    [M,I] = min(K);

end

function arduinoControl(I)
    a=arduino();
    RED_PIN='D12';
    GREEN_PIN='D6';
    YELLOW_PIN='D9';
    writeDigitalPin(a,RED_PIN,0);
    writeDigitalPin(a,GREEN_PIN,0);
    writeDigitalPin(a,YELLOW_PIN,0);
    if I == 1
        disp('ROJO');
        writeDigitalPin(a,RED_PIN,1);
        pause(2);
    elseif I == 2
        disp('VERDE');
        writeDigitalPin(a,GREEN_PIN,1);
        pause(2);
    elseif I == 3
        disp('AMARILLO');
        writeDigitalPin(a,YELLOW_PIN,1);
        pause(2);
    end
    clear a;
end
```

Ejemplo de Aplicación

A continuación, se demuestra la ejecución del programa creado en Matlab

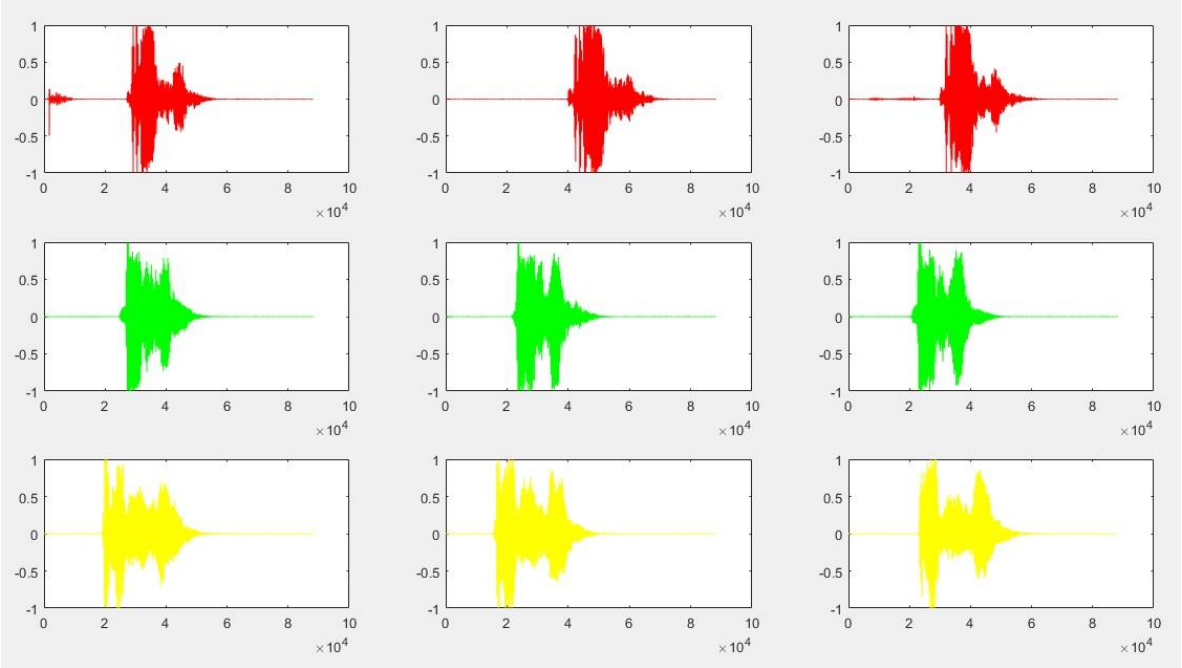
Inicialmente se conecta la placa Arduino y se espera a que el programa la reconozca.

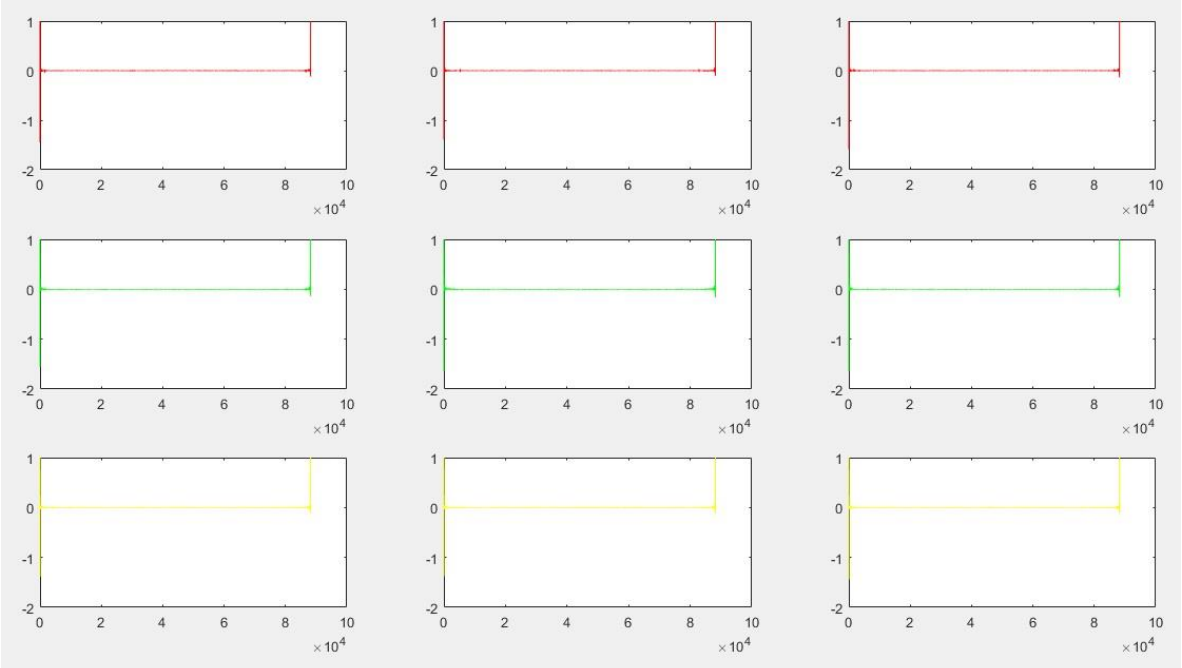


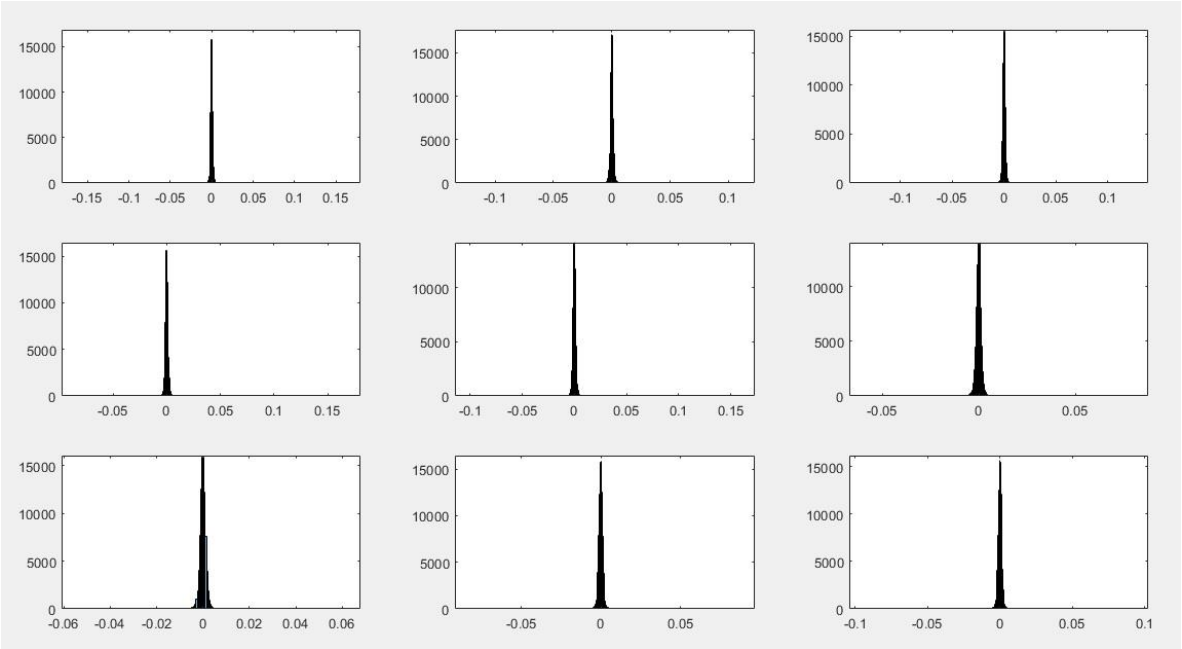
Luego se corre el programa, se da la opción de grabar los colores con los que se entrenará a los agentes o de utilizar archivos de audio ya grabados.

```
>> trabajoFinal
1_Grabar nueva base de datos
2_Usar base de datos existente
```

Se grafican los archivos de los colores grabados, los audios procesados mediante los cepstrums y su respectivo histograma







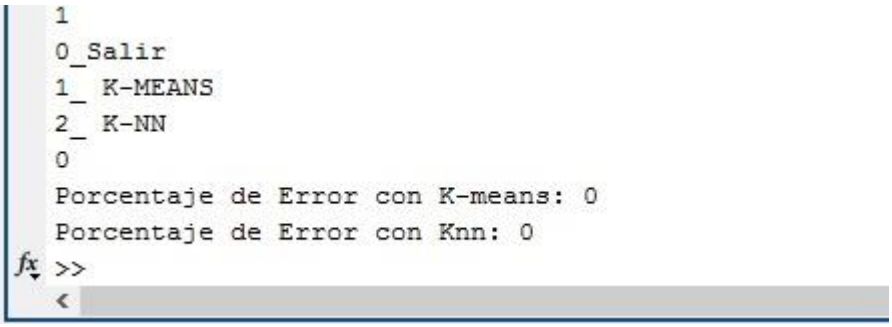
Se procede a seleccionar el algoritmo que deseamos utilizar

```
0_Salir
1_ K-MEANS
2_ K-NN
```

Al seleccionar un algoritmo, se muestra el nombre del cual se seleccionó, e inmediatamente comienzan los 2 segundos de grabación. Luego se imprime el color leído (al mismo tiempo que se prende el led correspondiente). Finalmente nos pregunta si la lectura fue correcta o no.

```
Command Window
1
K-Means
Comience a hablar.
Fin de la grabación.
ROJO
1-Lectura correcta
2-Lectura incorrecta
1
```

El proceso continúa hasta que apretamos el "0" y el programa finaliza. Inmediatamente se muestran en pantalla el porcentaje de error de cada algoritmo, con el cual podremos decidir cuál es el más eficiente



Conclusiones

Como primera conclusión, luego de correr varias pruebas en distintos ambientes, se observa que ambos algoritmos se ven afectados por el ruido introducido en las grabaciones. Esto se solucionaria filtrando la señal y a su vez con hardware de mejor tecnología.

Por otra parte, en la comparación del rendimiento, si bien, frente a condiciones favorables para la reducción del ruido ambos algoritmos presentan un rendimiento elevado, K-NN se comportó mejor frente a los cambios de ambiente, sin tener que volver a grabar la base de datos nuevamente, como lo tendríamos que hacer si quisiéramos tener un K-means eficiente.

Por último, al cambiar la base de datos de 9 grabaciones a 18 grabaciones los resultados mejoraron notablemente en ambos algoritmos, donde en este caso, la brecha de rendimiento entre ambos fue menor.

Bibliografía

- Fit k-nearest neighbor classifier (https://www.mathworks.com/help/stats/fitcknn.html)

- k-means clustering (https://www.mathworks.com/help/stats/kmeans.html#buefthh-2)

- Cepstrum Analysis (https://www.mathworks.com/help/signal/ug/cepstrum-analysis.html)

- Apuntes de la catedra de Inteligencia Artificial 1 de la Facultad de Ingeniería, versión 2017