Clasificación Bayesiana

Práctica 5

César Borja

15/04/2022

Aprendizaje Automático

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Índice

1. Objetivo

El objetivo de la práctica es resolver mediante clasificación Bayesiana el reconocimiento de dígitos manuscritos. Se utilizará una versión reducida del conjunto de datos MNIST.

2. Estudio previo

El algoritmo de entrenamiento y clasificación multi-clase utilizando clasificación Bayesiana con atributos Gaussianos consiste simplemente en llamar a la función **entrenarGaussianas** y **clasificacionBayesiana** explicadas en la sección 3

3. Entrenamiento y clasificación con modelos Gaussianos regularizados

Se pide programar las siguientes funciones:

1. **entrenarGaussianas**: aprende el modelo Gaussiano de cada clase, es decir, calcula el vector de medias y la matriz de covarianza de los datos de entrenamiento para cada clase. También permite entrenar modelos bayesianos tanto completos como ingenuos, diagonalizando la matriz de covarianzas en el caso de Bayes ingenuo. Además, aplica el factor de regularización pasado por parámetro. En la Figura 1 se muestra la implementación de esta función en MATLAB.

2. **clasificacionBayesiana**: hace la clasificación de un conjunto de muestras utilizando los modelos Gaussianos entrenados con la función entrenarGaussianas. Para ello se calcula la salida predicha para cada muestra de la siguiente manera: se calcula la probabilidad de que cada muestra pertenezca a cada clase, haciendo uso de la función **gaussLog** y la probabilidad a priori. La salida predicha para cada muestra es la que tiene la máxima probabilidad. En la Figura 2 se muestra la implementación de esta función en MATLAB.

Además, se va a utilzar regularización para evitar sobreajuste. Para ello se ha implementado la función **entrenarYclasificarBayes** que elige el mejor factor de regularización para el entrenamiento de los modelos. Se comparan las tasas de error de los datos de validación para elegir el mejor *lambda*. El código de esta función se encuentra en la Figura 3.

```
function modelo = entrenarGaussianas( Xtr, ytr, nc, NaiveBayes, landa )
% Entrena una Gaussana para cada clase y devuelve:
% modelo{i}.N      : Numero de muestras de la clase i
% modelo{i}.mu     : Media de la clase i
% modelo{i}.Sigma : Covarianza de la clase i
% Si NaiveBayes = 1, las matrices de Covarianza ser n diagonales
% Se regularizar n las covarianzas mediante: Sigma = Sigma + landa*eye(D)

[~,D] = size(Xtr);

for i=1:nc
    [Ni,~] = size(find(ytr==i));
    Xi = Xtr((ytr==i),:);
    modelo{i}.N = Ni;
    modelo{i}.mu = mean(Xi);
    modelo{i}.Sigma = cov(Xi);

    if NaiveBayes == 1
        modelo{i}.Sigma = diag(diag(modelo{i}.Sigma));
    end

    modelo{i}.Sigma = modelo{i}.Sigma + landa*eye(D);

end
```

Figura 1: función entrenarGaussianas en MATLAB

```
function yhat = clasificacionBayesiana(modelo, X)
% Con los modelos entrenados, predice la clase para cada muestra X
    ypred = [];
    for i = 1:width(modelo)
        Pi = modelo{i}.N / height(X);
        % se multiplica por la probabilidad a priori de la clase i
        ypred(:, i) = gaussLog(modelo{i}.mu,modelo{i}.Sigma,X) * Pi;
    end
    [~,yhat] = max(ypred,[],2);
end
```

Figura 2: función clasificacionBayesiana en MATLAB

```
function [Etr,Ecv,best_lambda] = entrenarYclasificarBayes(Xtr,ytr,Xcv,ycv, ...
    N_clases,lambda,NaiveBayes)

    Etr = [];
    Ecv = [];
    best_lambda = 0;
    best_errV = inf;

    for l = 1:length(lambda)
        etr = 0; ecv = 0;

        modelo = entrenarGaussianas(Xtr,ytr,N_clases,NaiveBayes,lambda(l));
        ytr_pred = clasificacionBayesiana(modelo,Xtr);
        ycv_pred = clasificacionBayesiana(modelo,Xcv);

        etr = tasa_error(ytr_pred,ytr);
        ecv = tasa_error(ycv_pred,ycv);

        Etr = [Etr;etr];
        Ecv = [Ecv;ecv];

        if ecv < best_errV
            best_errV = ecv;
            best_lambda = l;
        end
    end
end
```

Figura 3: función entrenarYclasificarBayes en MATLAB

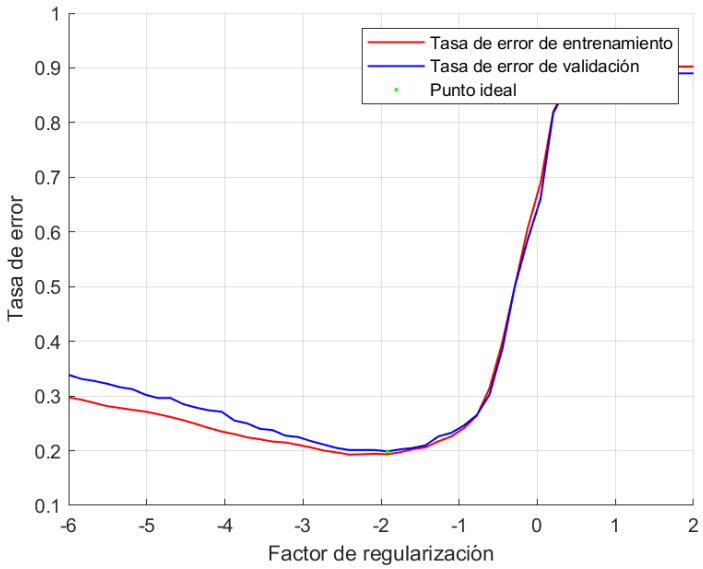

Figura 4: Evolución de las tasas de error con datos de entrenamiento y datos de validación en función del factor de regularización utilizando Bayes ingenuo

4. Bayes ingenuo

Se pide programar el entrenamiento y la clasificación multi-clase del conjunto de datos MNIST utilizando Bayes ingenuo, separando un 20 % de los datos para validación.

Haciendo uso de la función **entrenarYclasificarBayes** se obtiene el mejor valor para el factor de regularización para Bayes ingenuo: $\lambda = 0{,}0121$. En la Figura 4 se muestra la evolución de las tasas de error con los datos de entrenamiento y validación en función de λ.

Como se puede observar en la imagen, ambas tasas de error aumentan para valores pequeños de λ. Este subajuste es debido a que se está usando un modelo de Bayes ingenuo. Estos modelos suponen la independencia de los atributos entre sí. En un ejemplo simplificado en el que tuviéramos dos atributos por clase el modelo se podría representar gráficamente como una elipse vertical u horizontal (en el caso a tratar, se trataría de un elipsoide de 400 dimensiones), dependiendo de la escala de ambos atributos. Sin embargo, no sería posible representarla como una elipse diagonal por la suposición mencionada.

$$\begin{pmatrix}
98 & 6 & 5 & 2 & 3 & 3 & 4 & 8 & 2 & 0 \\
1 & 75 & 5 & 3 & 1 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 75 & 0 & 12 & 0 & 0 & 3 & 1 & 0 \\
0 & 0 & 1 & 75 & 5 & 1 & 2 & 3 & 4 & 0 \\
0 & 0 & 1 & 1 & 61 & 0 & 0 & 2 & 0 & 3 \\
0 & 9 & 1 & 2 & 4 & 95 & 0 & 1 & 0 & 4 \\
0 & 0 & 1 & 0 & 1 & 0 & 86 & 0 & 1 & 0 \\
0 & 8 & 6 & 3 & 6 & 1 & 1 & 75 & 3 & 4 \\
0 & 0 & 5 & 14 & 5 & 0 & 7 & 7 & 89 & 0 \\
0 & 2 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 89
\end{pmatrix}$$

Figura 5: Matriz de confusión obtenida utilizando los datos de test para el conjunto de datos MNIST con el modelo de Bayes ingenuo con factor de regularización $\lambda = 0.0121$

.

Por ello, en problemas de clasificación donde los atributos sí están correlacionados (como el tratado en esta práctica), al no poder dibujar una elipse diagonal, el área de dicha elipse tiene que ser más grande, y por tanto, el error será mayor.

Esto último se ve acentuado si no se utiliza regularización, puesto que los datos espurios harán que dicha elipse sea menos precisa. Por este motivo, la tasa de error con los datos de entrenamiento aumenta con valores de λ muy pequeños.

La matriz de confusión se muestra en la Figura 5. Las filas representan las clases predichas mientras que en las columnas se representan las clases reales. Además, se pide calcular los valores de precisión y recall para cada dígito. Dichos valores se pueden ver en el Cuadro 1.

Por último, en la Figura 6 se visualizan algunas de las confusiones obtenidas con la función verConfusiones() administrada como material de la práctica.

¿Qué dígitos son los más problemáticos?

En la Figura 6 se puede ver como el 5 es el número que más se confunde. Esto se corrobora obteniendo el recall del 5, el cual es de 0.61, mucho más bajo que el de los demás dígitos.

Dígito	Precisión	Recall
1	0.7481	0.9800
2	0.8721	0.7500
3	0.8152	0.7500
4	0.8242	0.7500
5	0.8971	0.6100
6	0.8190	0.9500
7	0.9663	0.8600
8	0.7009	0.7500
9	0.7008	0.8900
0	0.9570	0.8900

Cuadro 1: Precisión y recall para cada dígito utilizando los datos de test con el modelo de Bayes ingenuo

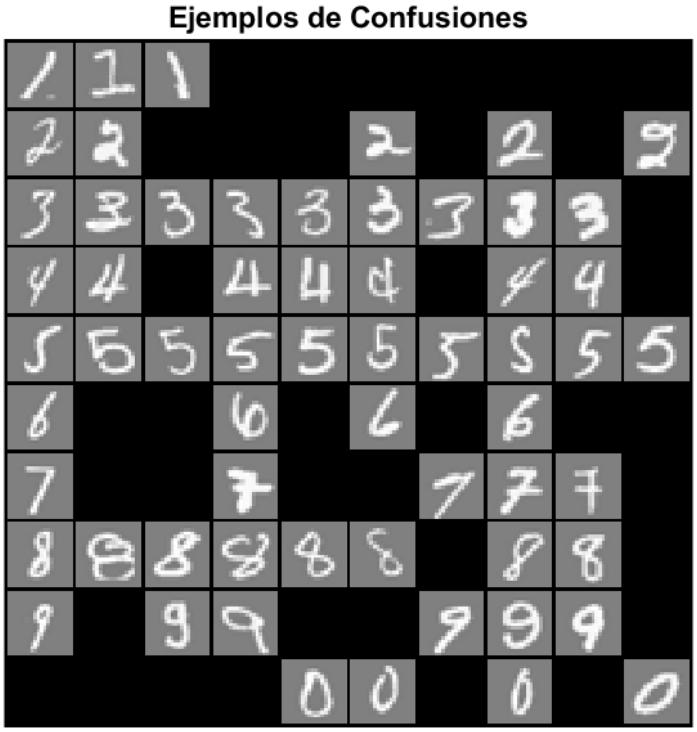

Figura 6: Ejemplos de las confusiones habidas utilizando Bayes ingenuo

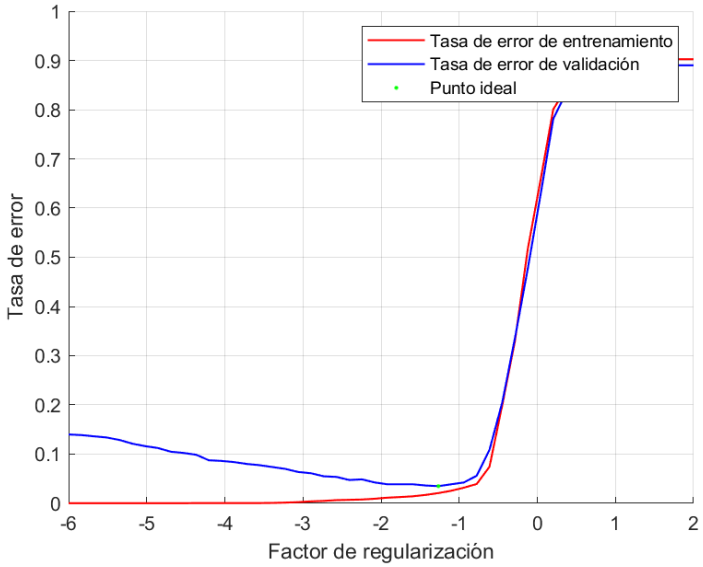

Figura 7: Evolución de las tasas de error con datos de entrenamiento y datos de validación en función del factor de regularización utilizando Bayes completo

5. Covarianzas completas

Se pide ahora repetir el apartado anterior pero utilizando Bayes completo. Utilizando la función **entrenarYclasificarBayes** se obtiene el mejor valor para el factor de regularización para Bayes completo: $\lambda = 0{,}0543$. En la Figura 7 se muestra la evolución de las tasas de error con los datos de entrenamiento y validación en función de λ.

En este caso se puede apreciar claramente como aparece sobreajuste para valores pequeños de λ y subajuste para valores grandes.

Esto se debe a que en los modelos de Bayes completo sí que se tienen en cuenta las correlaciones entre los atributos, y por tanto, la representación gráfica del modelo sería un elipsoide de 400 dimensiones diagonal, ofreciendo una mayor precisión.

La matriz de confusión obtenida se muestra en la Figura 8 y los valores de precisión y recall se muestran en el Cuadro 2.

Por último, en la Figura 9 se visualizan algunas de las confusiones obtenidas.

$$\begin{pmatrix} 99 & 3 & 1 & 1 & 0 & 0 & 2 & 2 & 1 & 0 \\ 1 & 94 & 2 & 1 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 1 & 94 & 0 & 1 & 0 & 0 & 3 & 2 & 0 \\ 0 & 0 & 0 & 98 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 95 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 99 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 96 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 3 & 0 & 0 & 91 & 2 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 2 & 1 & 94 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 100 \end{pmatrix}$$

Figura 8: Matriz de confusión obtenida utilizando los datos de test para el conjunto de datos MNIST con el modelo de Bayes completo con factor de regularización $\lambda = 0{,}0543$

Dígito	Precisión	Recall
1	0.9083	0.9900
2	0.9400	0.9400
3	0.9307	0.9400
4	1.0000	0.9800
5	1.0000	0.9500
6	0.9802	0.9900
7	0.9796	0.9600
8	0.9286	0.9100
9	0.9592	0.9400
0	0.9804	1.0000

Cuadro 2: Precisión y recall para cada dígito utilizando los datos de test con el modelo de Bayes completo

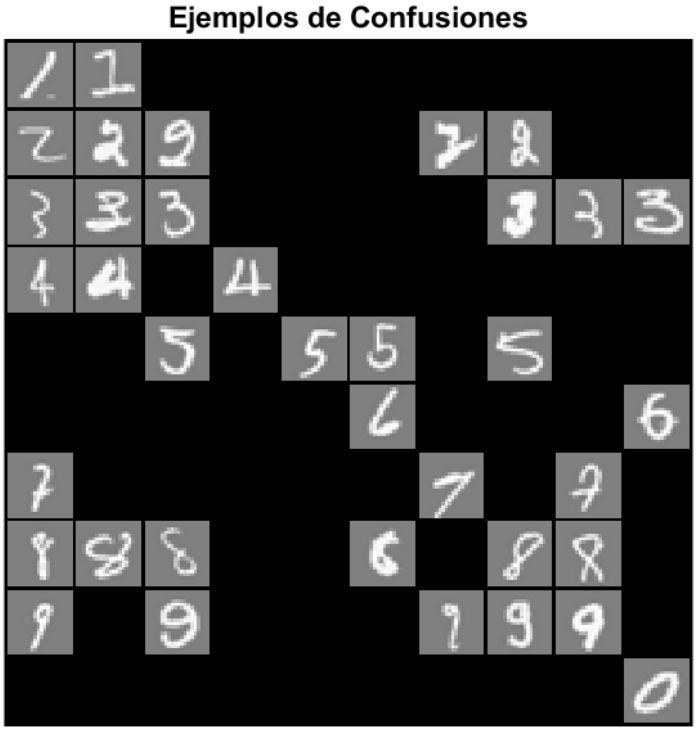

Figura 9: Ejemplos de las confusiones habidas utilizando Bayes completo

.

¿Qué dígitos son los más problemáticos?

El 8 es el más problemático en este caso. En el Cuadro 2 se observa como el dígito 8 es el que tiene mayor recall y es el que más apariciones tiene en en la Figura 9.

¿Qué modelo es mejor?

El modelo que utiliza Bayes completo es evidentemente mejor debido a que los atributos de los datos están correlacionados.