Informe Proyecto Final

Proyecto Estándar : Reconocimiento de letras

Reconocimiento de Patrones

Alicia Fernández
Pablo Cancela
Guillermo Carbajal
Ignacio Ramírez

Alumno: Pablo Rodríguez

Instituto de Ingeniería Eléctrica
Facultad de Ingeniería
Universidad de la República

Montevideo, Uruguay
8 de Diciembre de 2014

Indice

Descripción del Problema

Se eligió trabajar con el proyecto estándar propuesto en el curso. El problema consistirá en identificar letras a partir de un gran número de datos definidos por un conjunto de 16 características.

1 x-box posición horizontal del recuadro (int)
2 y-box posición vertical del recuadro (int)
3 width ancho del recuadro (int)
4 high altura del recuadro (int)
5 onpix # total de pixels encendidos (int)
6 x-bar media de x de pixels encendidos dentro del recuadro (int)
7 y-bar media de y de pixels encendidos dentro del recuadro (int)
8 x2bar varianza de x (int)
9 y2bar varianza de y (int)
10 xybar correlación entre x e y (int)
11 x2ybr valor medio de x * x * y (int)
12 xy2br valor medio x * y * y (int)
13 x-ege conteo medio de bordes horizontales izq a der (int)
14 xegvy correlación de bordes horizontales con y (int)
15 y-ege conteo medio de bordes verticales abajo a arriba (int)
16 yegvx correlación de bordes verticales con x (int)

Objetivo

1. Se desea lograr un sistema que sea capaz de clasificar correctamente, y en forma automática, a qué letra corresponde cada vector de características. El requerimiento mínimo a alcanzar es el de lograr que la media aritmética (Amean) de las tasas de acierto por clase esté por encima de 90%. Este requisito se impone para un sorteo de los vectores de test, asumiendo equiprobabilidad entre las clases.
2. Se desea construir un sistema para transcribir un texto en español (tomando la Ñ como una N) a partir de los vector de características de imágenes provenientes de las letras de un texto. Se establece como nuevo objetivo que el sistema maximice su tasa de acierto, conociendo que el conjunto de testeo proviene de un texto en idioma español.

2

Enfoque

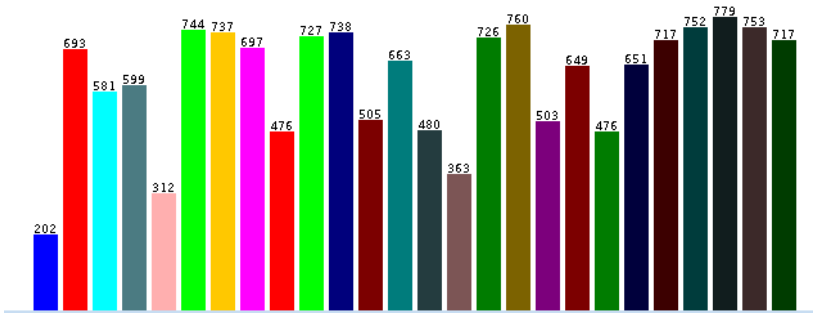
Se decidió hacer pruebas de desempeño con varios algoritmos clásicos para tener un punto de partida. Para luego seleccionar a los posibles candidatos a optimizar.
En la etapa de optimización se buscó tratar el problema del desbalanceo de clases mediante la aplicación de dos técnicas: submuestreo de las clases mayoritarias y generación de nuevas muestras con SMOTE.

Inicialmente se aplicaron herramientas de selección y extracción de características pero se descarto la opción por no mostrar mejoras.

La solución que se implementó estuvo compuesta por varias pruebas iniciales con Weka, pero la versión final se hizo completamente matlab/octave, usando la librería libsvm.

Características de los datos

- Los datos son discretos en el rango de 0 a 15.
- Se tiene un buen numero de patrones con relación a la cantidad de características.
- El conjunto de datos no esta perfectamente balanceado.



Pruebas con algoritmos

La siguiente tabla presenta los resultados de los desempeños de los clasificadores clásicos que se probaron. Las pruebas se hicieron usando weka. Los que presentaron mejor rendimiento fueron C-SVM y Knn.

Algoritmo	Desempeño
NaiveBayes	64,00%
NaiveBayesMultinomial	53,00%
Vecino mas cercano	95,00%
K vecinos, k=2	94,00%
k vecinos, k=3	95,00%
K vecinos, k=4	94,80%
k vecinos, k=5	95,00%
SVM	96,70%
C4.5	86,00%
RandomForest	93,00%
Clasificacion via Regresion	90,00%

Sobre la elección del clasificador

La idea inicial fue intentar combinar ambos clasificadores pero luego de analizar las matrices de confusión, se encontró que no había complementariedad entre ambos.//TODO anexo matrices de confusión.

Para elegir el clasificador que se aplicaría al problema, se realizo una comparación de desempeño de los clasificadores candidatos frente a datos ruidosos, con el fin de quedarse con el mas robusto.

Generación de conjuntos de entrenamiento

Se sabe que el primer conjunto de datos que es necesario clasificar tiene distribución uniforme mientras que el conjunto de entrenamiento brindado tiene la distribución que se ve en la figura 1.

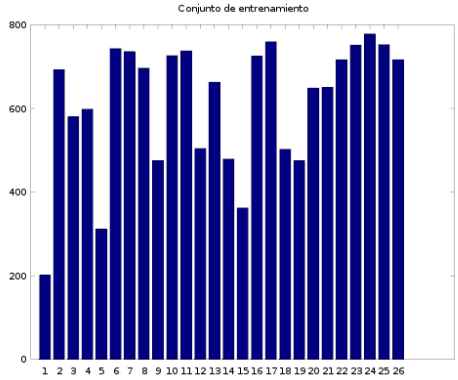


Figura 1: Distribución entrenamiento

Para encarar la generación de los conjuntos de entrenamiento con la alguna distribución arbitraria(i.e:uniforme) se uso la técnica de Smote (Figura 2) y otra variante,(Figura 3) para generar nuevas muestras sintéticas a partir de muestras reales. Las imágenes siguientes ejemplifican datos sintéticos (rojos) a partir de reales(azules) con las dos técnicas para el caso 2D.

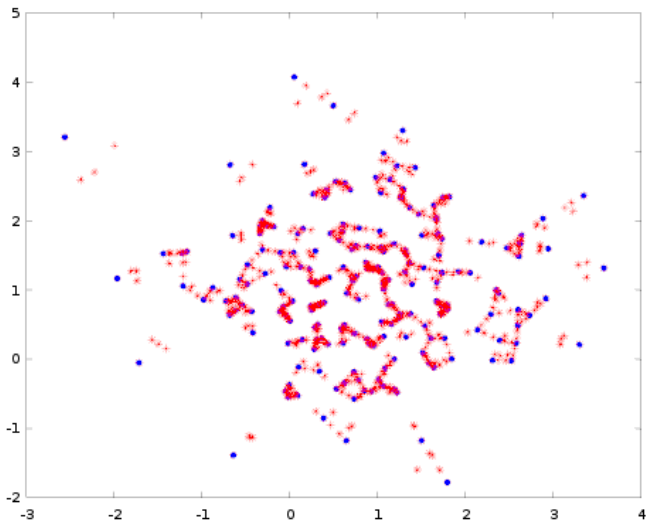


Figura 2: Smote

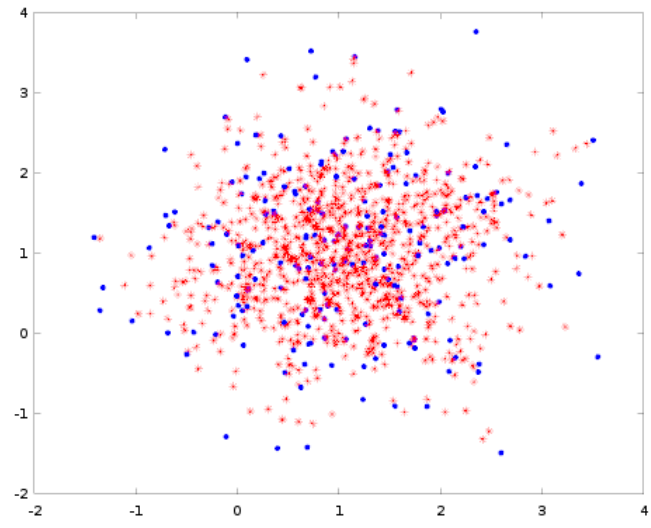


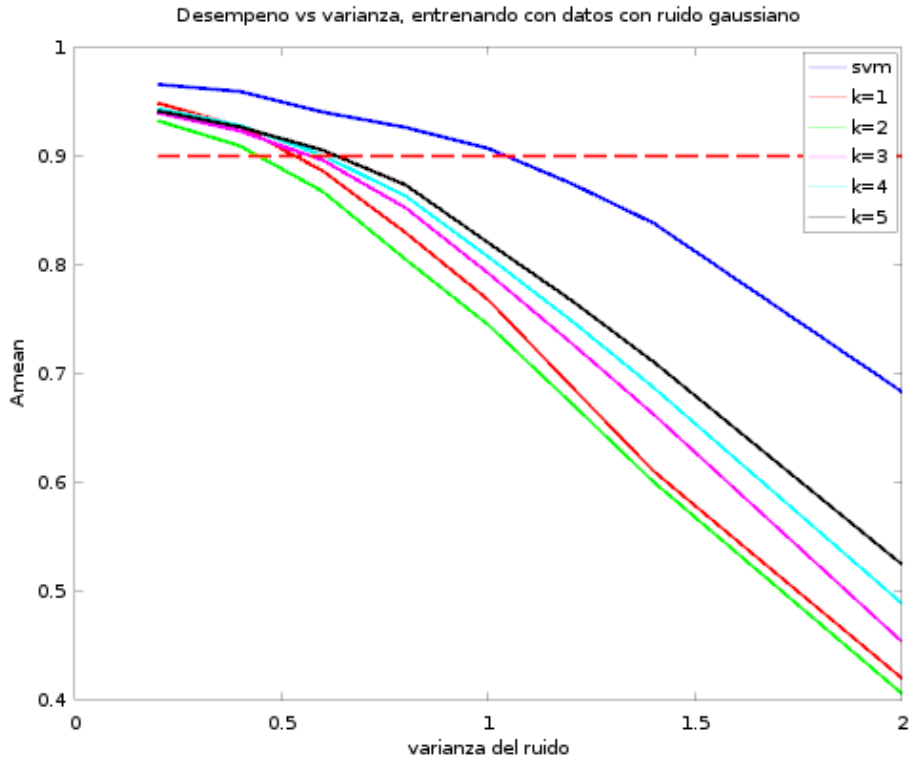
Figura 3: Ponderación randómica entre k=3 muestras

Elección de C-SVM sobre Knn

El conjunto de entrenamiento usado en las siguientes pruebas presento las siguientes características.
*Distribución uniforme.
*El numero de patrones por clase se eligió como la media de la cantidad de patrones por clase de las distribución original, resultando aproximadamente 600.
*Se generarían muestras de una clase solo en el caso que no hubiesen las necesarias para hacer un subsampling de muestras reales.
*Se agrega ruido blanco de media 0 y varianza variable.

La técnica de generación que se eligió fue la que se ve en la figura 3. Dado que se prefirió mas aleatoriedad en la generación para no favorecer a Knn.

Se implemento el script *Parte1_DesempenoVsRuidoParaSVMyKNN.m*, de donde se obtuvo que C-SVM es menos afectado por el ruido que K vecinos para k=[1 2 3 4 5] con este conjunto de entrenamiento dado. En la siguiente gráfica se pueden ver los desempeños en función de la varianza del ruido para los distintos clasificadores. La linea de color azul, es la que mejor se comporta, siendo svm.



Se puede ver que aproximadamente el limite alto de la varianza del ruido en el que el desempeno de C-Svm baja a menos de 0,9 es 1 aproximadamente. Mas adelante se agrega ruido al conjunto de entrenamiento para disminuir chances de sobre entrenamiento. Se decidió que la varianza seria igual a 0.4, degrada en 2% el desempeño, de los 6% que se tienen por arriba del limite que se busca.

Búsqueda del Valor de C

Una vez definido que se usaría C-SVM, se decidió encontrar el valor de C que mejor desempeño lograse en las pruebas de entrenamiento.

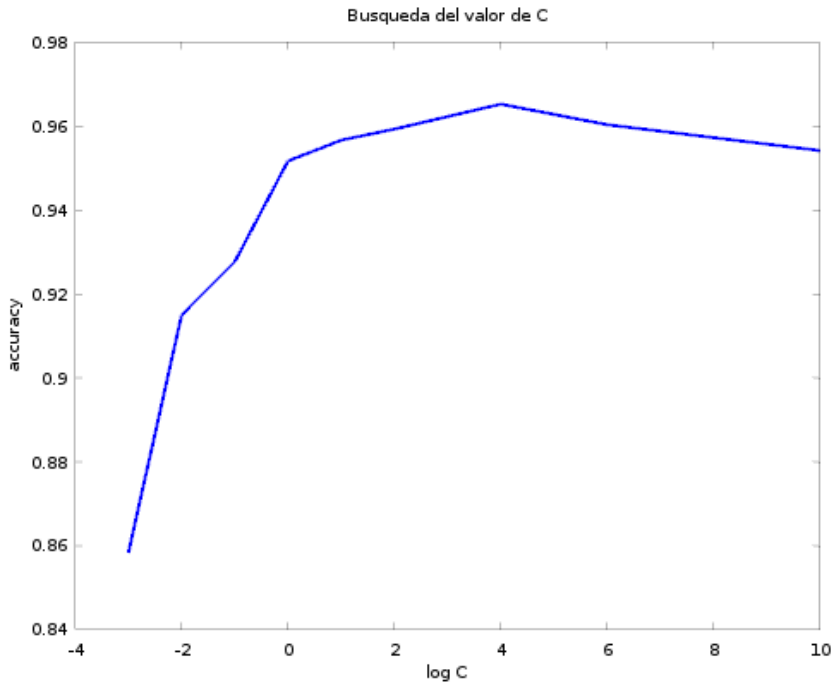
Se implemento un script que realiza validación cruzada, con la particularidad que genera conjuntos de entrenamiento y testing con distribución uniforme a partir del conjunto de entrenamiento real de la siguiente manera:

Se dividen los datos originales en dos conjuntos, entrenamiento y test, en proporciones 0.85 y 0.15 del conjunto original.
A partir del subconjunto de entrenamiento se genera otro con distribución uniforme, el cual se usa para entrenar el clasificador. La cantidad por clase fue la media del subconjunto de entrenamiento, siendo necesario agregar muestras sintéticas.

El conjunto usado para testear se crea subsampleando el subconjunto de test .
La cantidad de muestras por clases es igual a la que tiene la clase minoritaria. Con lo que se usan solo muestras reales.

Para cada valor de C con el que se testeo, se realizaron varias repeticiones para compensar la de las particiones de los conjuntos.
La medida de desempeño que se calculó fue el Accuracy promedio por clase, Amean.

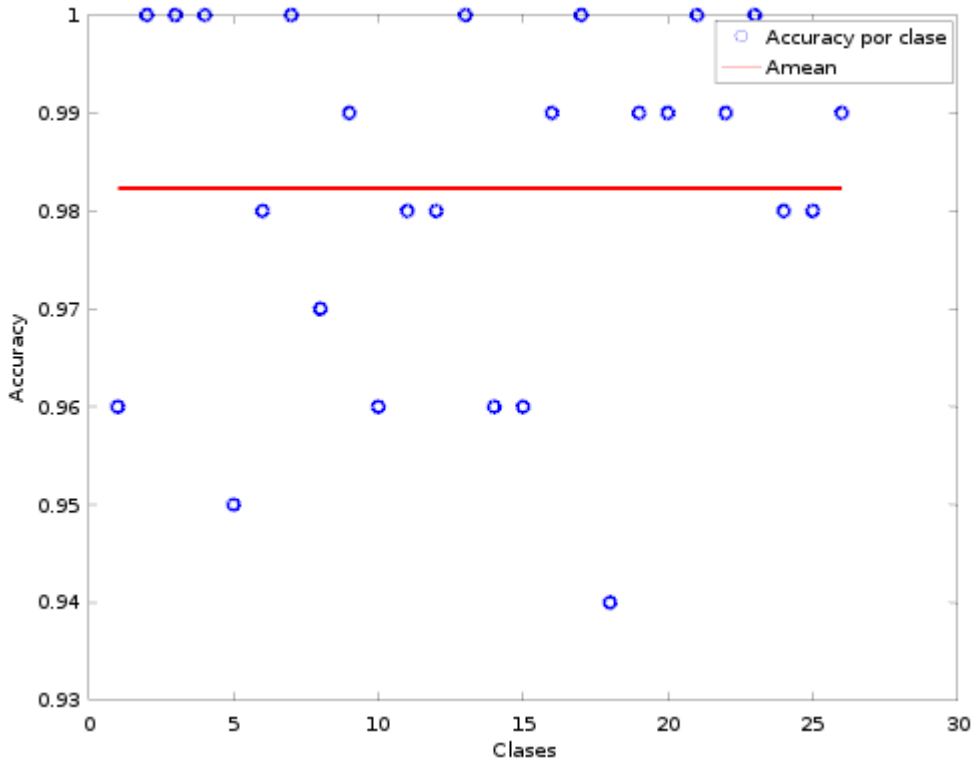
El script Parte1_BusquedaCSvm implementa lo anterior.



Parte 1-Entrenamiento y Test

Siguiendo las ideas anteriores de usar una distribución uniforme para entrenar el clasificador, con el valor de C hallado en la parte anterior, se entrenó un modelo svm y luego se uso para clasificar los datos de test reales.
La siguiente imagen muestra el accuracy logrado por clase y el Amean.

Amean = 98.23 %



Se obtuvó un resultado 2% mejor que los que mostraban los test de entrenamiento.

Parte 2-Entrenamiento y Test

Ahora se agrega la información a priori de que el conjunto de test proviene del idioma español.

Para aprovechar este nuevo dato, se obtuvieron las probabilidades de ocurrencia por cada letra y también la probabilidad de ocurrencia de dos letras seguidas para el idioma español, con la idea de probar dos alternativas para este problema.
El script Parte2_ProbLetrasYSecuencias calcula la probabilidad a partir de un libro en español con 600 mil letras.

El primer encare solo usa la probabilidad de ocurrencia de las letras. Se la incluirá en la distribución de los datos de entrenamiento.
La figura 5 muestra una representación de las ocurrencia por clases si el conjunto total tiene 16000 muestras. Se observó que hay clases que tienen probabilidades tan bajas que a lo sumo necesitan 1 muestra si se sigue estrictamente esta distribución. Clases con tan pocas muestras pueden no ser representativas fieles de su tipo, por lo que esto puede ocasionar errores de clasificación.

Se probaron distintos valores de elementos por clases y se analizaron los desempeños testeados con el resto de los elementos de la clase. En el entorno de 200 patrones se observó que es un buen compromiso entre desempeño y poca cantidad de muestras.
A partir de esto se estableció un mínimo de elementos por clases, con esto se altera la distribución de ocurrencia de las letras en el idioma español, pero como se puede ver en la figura 4, aun se mantiene cierta semejanza con la original.

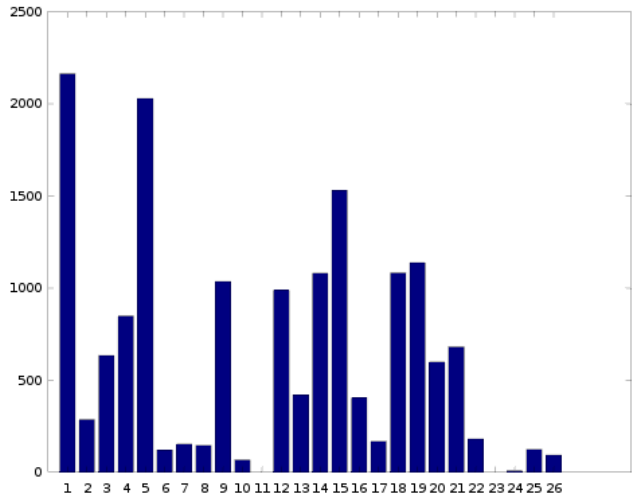


Figura 5: Distribución letras Idioma Español

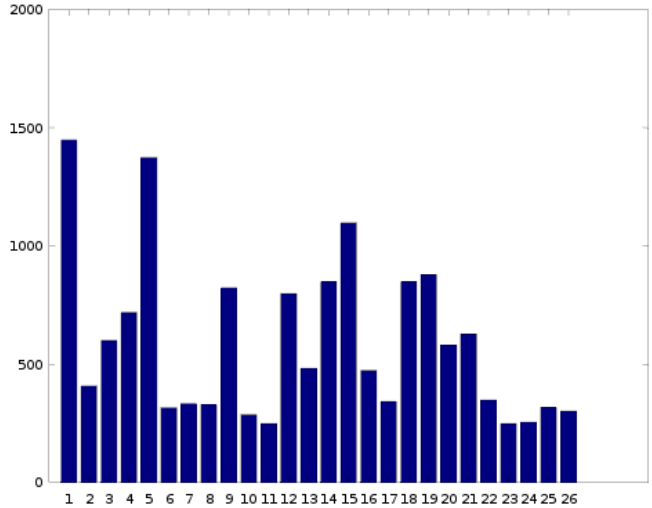


Figura 4: Distribución con cota mínima

Análogo a la parte 1 se busco el valor C que mejor optimizaba el entrenamiento con los datos de entrenamiento. Se dividió el conjunto de patrones originales en 2 subconjuntos. Con uno se creo el conjunto de entrenamiento, con la distribución que se describió anteriormente. Fue necesario incluir muestras sintéticas.
El conjunto de test se hizo en semejanza a una secuencia de caracteres extraída del libro donde se calcularon las probabilidades de ocurrencia por letra. Se usaron los patrones del 2do subconjunto, subsampleados y ordenados como el texto. Los mismos patrones repetidos y re ordenados muchas veces, no es lo mejor, pero se prefirió esto a testear con muestras sintéticas.

La figura 6 muestra la evolución del desempeño al variar C

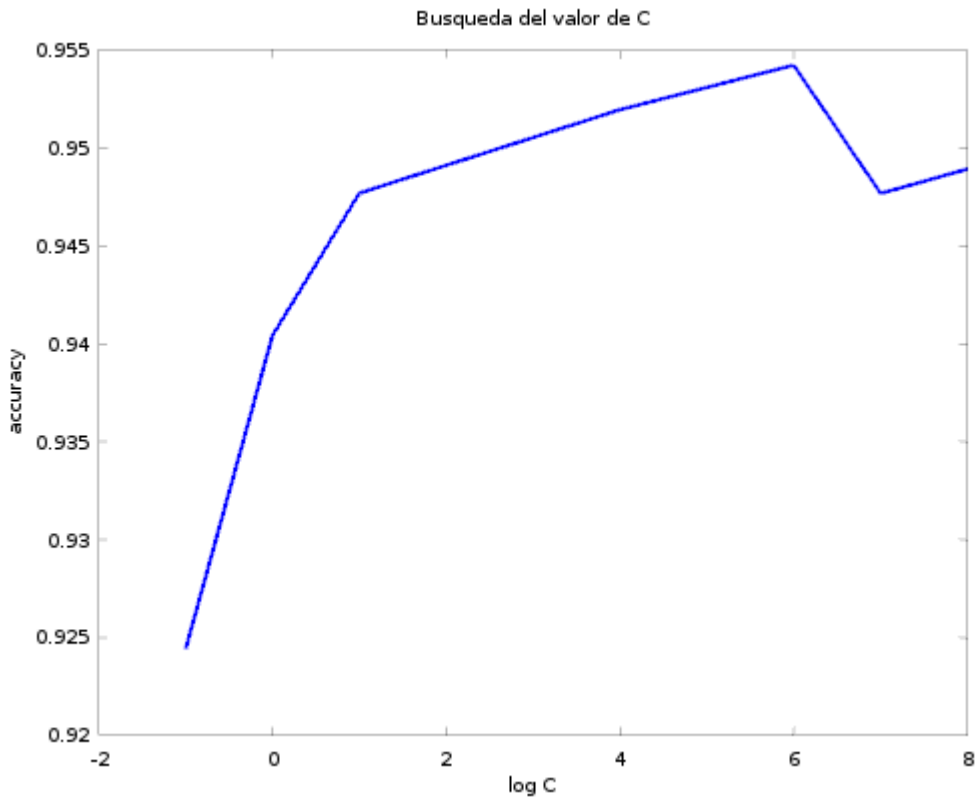


Figura 6: Parte 2 optimización C

Con el valor de C hallado se entreno un clasificador con la distribución con cota mínima por clase, con todo el conjunto de entrenamiento y se clasificaron las muestras del conjunto de test real. También se clasificaron los nuevos patrones con el modelo entrenado en la parte 1 para comparar resultados.

Modelo Parte 1 Amean=0.9861 , Accuracy=0.974
Modelo Parte 2 Amean=0.980 , Accuracy=0.983

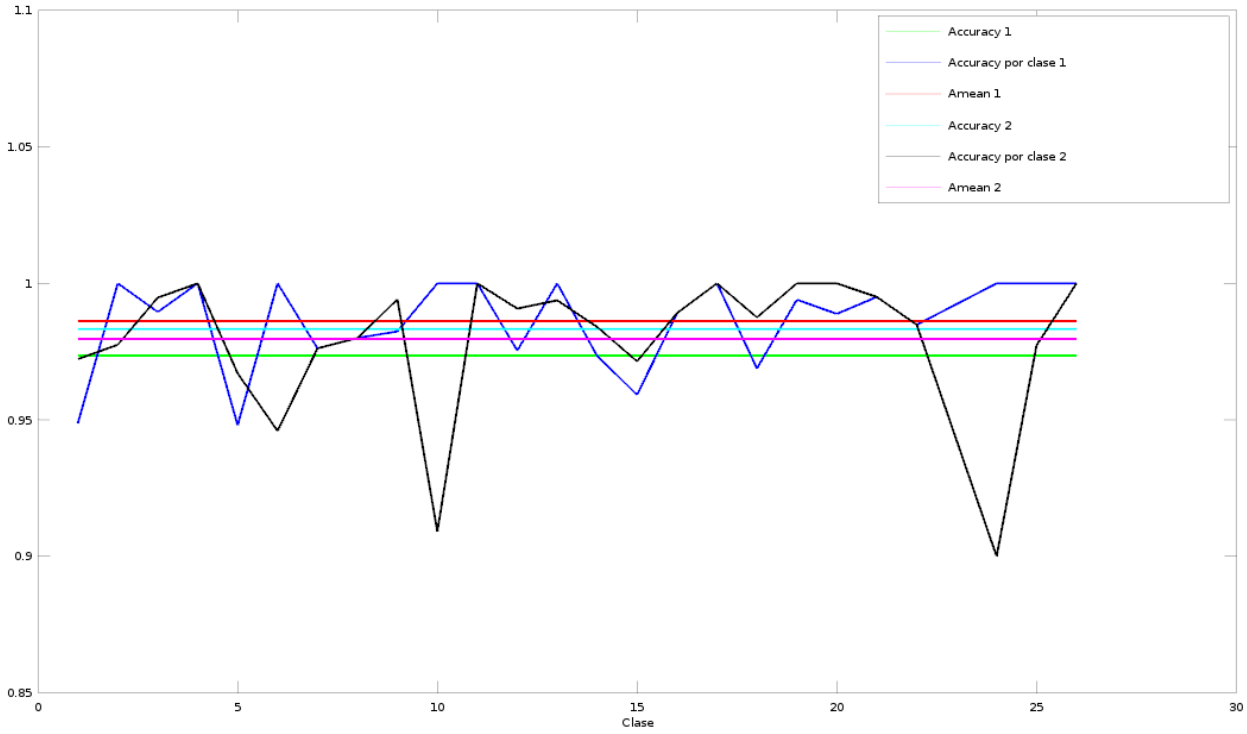


Figura 7: Comparación de resultados. En azul (1)y negro(2), Accuracy por clase

En rojo(1) y magenta(2) : Amean

En verde(1) y cyan(2): Accuracy

De la comparación de los resultados se puede ver que el Amean disminuyó levemente con respecto a la prueba anterior, pero el accuracy tuvo una leve mejora.

El Amean bajó por que hubieron clases en las que se tuvo peores desempeño. Principalmente los dos grandes valles en negro que sobresalen.
El Accuracy total aumento como consecuencia de que las clases mayoritarias, las letras a y e, en la parte 2, fueron mejor clasificadas que en la parte 1.

El segundo encare al problema es una variante del caso anterior. La idea es que en vez de utilizar la misma distribución de frecuencias de letras para todas las letras, se tendrá en cuenta cual fue la letra anterior y se usará una distribución acorde.
Esto involucra entrenar 26 clasificadores, uno por cada letra anterior.
En este caso no se hizo búsqueda exhaustiva de un optimo valor de C, dado el tiempo y complejidad de entrenamiento que requiere esta alternativa. Para simplificar se uso el mismo valor hallado en la parte anterior.

El programa se encuentra implementado en el archivo Parte2_Alternativa2.m

Se obtuvieron los siguientes resultados:

Amean = 0.985 , Accuracy = 0.983

Comparando con los 2 métodos anteriores. Este presenta el Accuracy mas alto que se logró y el segundo valor mas alto Amean.

Conclusiones

- Se logró cumplir el objetivo propuesto del requerimiento de Amean con buen margen.

- Los tres clasificadores se comportaron correctamente. Hubo una leve mejora en la cantidad total de letras correctamente clasificadas entre el clasificador de la parte 1 y los 2 de la parte 2 cuando se usan datos en español. Lo cual era de esperarse, dado que estos últimos usaban información extra.

- La diferencia de desempeño entre los clasificadores de la parte 2 es muy poca como para justificar el costo de entrenar 25 clasificadores extras.

- Para compensar el desbalance de clase de los conjuntos de entrenamiento y test se usaron técnicas como Smote y subsampling con éxito.

- Se optimizaron los clasificadores usando una búsqueda lineal sobre C, la constante de C-Svm.

Anexo 1.

Matriz de confusión para vecino mas cercano

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	<--	classified	as	
197	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	|	a	=	A
0	662	0	1	2	0	1	4	0	1	0	0	0	0	0	0	0	5	0	1	1	14	0	0	0	1	|	b	=	B
0	0	570	0	2	0	2	0	0	0	0	0	0	0	2	0	0	0	2	0	0	1	2	0	0	0	|	c	=	C
1	3	0	570	0	0	2	11	0	0	0	0	0	2	4	1	1	2	0	2	0	0	0	0	0	0	|	d	=	D
0	3	5	0	278	0	9	3	0	0	4	0	0	0	0	0	1	0	0	1	0	0	1	4	0	3	|	e	=	E
0	1	0	0	0	699	0	2	2	1	0	0	0	1	0	23	1	1	9	1	0	2	0	0	1	0	|	f	=	F
0	5	6	4	3	0	710	0	0	0	1	1	1	0	1	0	1	0	0	0	0	2	1	1	0	0	|	g	=	G
0	5	1	10	3	0	2	632	0	1	20	1	2	4	5	2	0	8	0	0	0	0	0	1	0	0	|	h	=	H
0	0	0	0	0	5	0	0	449	19	0	0	0	1	0	0	0	0	0	0	0	0	0	2	0	0	|	i	=	I
0	1	0	0	0	2	0	2	17	700	0	1	0	2	0	0	0	0	0	0	2	0	0	0	0	0	|	j	=	J
0	3	0	0	1	0	1	26	0	0	681	0	0	0	0	0	0	11	0	0	0	0	0	15	0	0	|	k	=	K
0	0	1	0	1	0	3	2	0	2	0	490	0	0	0	1	0	2	0	0	0	0	0	2	0	1	|	l	=	L
0	2	0	0	0	0	1	0	0	0	0	0	651	2	1	0	0	0	0	0	0	4	2	0	0	0	|	m	=	M
0	1	0	4	0	0	0	4	0	1	0	0	2	453	1	0	1	9	0	0	0	4	0	0	0	0	|	n	=	N
0	1	1	11	0	0	2	3	0	1	0	0	1	3	327	1	6	0	0	0	2	2	2	0	0	0	|	o	=	O
0	1	0	3	0	31	1	2	0	0	0	2	1	1	0	680	2	1	0	0	0	0	0	0	1	0	|	p	=	P
0	1	0	2	0	0	3	0	0	0	0	0	0	0	7	2	739	2	0	0	0	0	0	1	1	2	|	q	=	Q
0	17	1	0	0	0	1	11	0	0	9	3	0	5	0	0	2	452	0	1	0	1	0	0	0	0	|	r	=	R
0	1	3	1	0	2	0	1	0	1	0	0	0	0	0	0	0	0	626	0	0	0	0	2	11	1	|	s	=	T
0	3	0	1	6	0	2	1	0	2	0	1	0	0	0	0	0	3	0	451	1	0	0	0	0	5	|	t	=	S
0	4	0	1	0	0	0	6	0	0	1	0	1	0	0	0	0	0	0	0	635	1	0	0	2	0	|	u	=	U
0	14	1	0	0	1	1	0	0	0	0	0	1	1	1	1	0	1	0	0	0	690	2	0	3	0	|	v	=	V
0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	0	1	3	745	0	0	0	|	w	=	W
0	3	0	2	2	0	0	0	0	0	20	1	0	0	2	0	1	1	1	2	0	0	0	744	0	0	|	x	=	X
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	2	2	1	1	737	0	|	y	=	Y
0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0	1	0	0	0	1	0	704 |	z	=	Z

Matriz de confusión para C-SVM

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	<--	classified	as	
195	0	1	0	0	0	0	2	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	2	0	|	a	=	A
0	674	0	2	0	0	1	2	0	0	1	0	0	0	0	0	0	1	0	1	0	8	0	3	0	0	|	b	=	B
0	0	567	0	2	0	4	2	0	0	0	0	0	0	4	0	0	0	1	0	0	0	1	0	0	0	|	c	=	C
0	1	0	587	0	0	0	7	0	1	0	0	0	2	1	0	0	0	0	0	0	0	0	0	0	0	|	d	=	D
0	1	4	0	284	0	13	0	0	0	2	0	0	0	0	0	1	0	0	1	0	0	0	1	0	5	|	e	=	E
0	0	0	1	0	726	2	1	0	1	0	0	0	0	0	6	0	0	3	0	0	2	0	1	0	1	|	f	=	F
0	2	0	9	1	0	721	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	2	0	0	0	|	g	=	G
0	0	0	15	0	0	5	641	0	1	14	0	1	0	1	0	1	16	0	0	1	0	1	0	0	0	|	h	=	H
0	0	0	0	0	3	0	0	443	29	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	|	i	=	I
0	0	0	1	0	0	0	1	5	716	0	0	0	1	2	0	0	0	0	0	1	0	0	0	0	0	|	j	=	J
0	0	0	0	0	0	0	8	0	0	711	0	0	0	0	0	0	10	0	0	1	0	0	8	0	0	|	k	=	K
0	0	0	0	1	0	4	2	0	0	1	489	0	0	0	0	0	3	0	0	0	0	0	5	0	0	|	l	=	L
0	5	0	0	0	0	2	0	0	0	0	0	652	1	0	0	0	0	0	0	1	0	2	0	0	0	|	m	=	M
0	1	0	2	0	0	0	5	0	0	0	0	4	453	5	0	0	7	0	0	0	2	1	0	0	0	|	n	=	N
0	1	2	11	0	0	2	1	0	0	0	0	0	0	333	1	5	0	0	0	0	0	7	0	0	0	|	o	=	O
0	2	0	0	0	21	3	4	0	0	0	0	0	0	0	692	2	0	0	0	0	0	1	0	1	0	|	p	=	P
0	0	0	1	0	0	0	1	0	0	0	0	0	0	2	1	752	1	0	0	0	0	0	0	2	0	|	q	=	Q
0	9	1	2	0	0	0	2	0	1	10	0	0	1	0	0	2	473	0	0	0	1	0	1	0	0	|	r	=	R
0	1	0	2	0	3	0	1	0	0	0	1	0	0	0	0	0	0	638	0	0	0	0	1	2	0	|	s	=	T
0	1	0	0	2	2	0	0	0	0	0	0	1	0	0	0	0	1	0	468	0	0	0	0	0	1	|	t	=	S
0	1	0	0	0	0	0	2	0	0	0	0	1	0	0	0	0	0	0	0	646	1	0	0	0	0	|	u	=	U
0	18	0	0	0	2	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	692	1	0	1	0 |	v	=	V
0	0	0	0	0	0	1	1	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	747	0	0	|	w	=	W
0	0	0	2	0	0	0	0	0	0	2	0	0	0	0	0	0	2	1	0	0	0	0	0	771	0	1 |	x	=	X
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	2	0	0	1	747	0 |	y	=	Y
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	713 |	z	=	Z

Anexo 2.

Diferencias entre datos discretizados con y sin ruido gaussiano de distintos sigmas.

Sigma= 0.2

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0

Sigma =0.4

0	0	-1	1	0	0	0	0	0	0	-1	-1	0	0	0	0
-1	-1	-1	-1	0	0	0	0	0	-1	1	0	0	0	0	0
0	0	0	0	0	-1	0	0	0	0	1	0	0	0	0	-1
0	0	0	-1	0	0	0	0	0	0	0	0	0	0	-1	0
0	0	1	1	0	0	-1	0	-1	0	0	0	0	0	0	0
0	0	0	0	0	-1	1	0	-1	0	0	0	1	0	-1	0
-1	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0

Sigma=0.6

-1	0	0	0	0	0	0	-1	0	1	0	0	0	1	0	1
0	0	-1	0	-1	0	0	-1	-1	0	-1	0	1	0	0	-1
1	-1	0	1	1	-1	0	-1	0	0	1	0	-1	1	0	0
0	-1	0	0	0	0	0	1	-1	0	0	0	0	1	0	1
0	0	1	-1	0	0	-1	1	0	0	1	0	1	0	-1	0
0	0	0	0	-1	0	0	0	0	-1	0	0	0	-1	0	1
-1	0	0	-1	1	0	0	1	0	0	0	0	0	0	0	0
0	0	-1	0	0	0	0	0	0	-1	0	0	0	-1	-1	-1

Sigma=0.8

-1	0	1	0	-1	1	2	-1	1	1	0	-2	0	0	1	1
-1	0	-1	0	0	1	0	0	0	0	0	0	-2	0	0	0
-2	1	0	0	-1	-1	0	0	-1	1	0	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	-1	0	-1	0
0	2	-1	0	1	0	0	0	1	1	0	-1	0	1	1	-2
1	0	0	-1	2	3	1	0	0	0	2	1	2	0	0	0
0	0	-2	0	1	0	-1	1	0	0	-1	0	0	0	0	0
0	0	-1	0	1	0	1	0	0	1	0	-1	0	2	1	1
0	1	-2	0	-1	-1	0	0	0	0	0	0	0	1	-1	0

Sigma=1

-1	0	0	1	-1	0	1	1	2	-1	0	1	-1	0	0	-1
-1	0	1	1	1	0	0	1	-1	1	0	1	-1	-1	0	0
0	2	-1	-1	0	0	1	-1	-2	-1	0	-2	1	0	0	1
0	0	-1	-2	0	2	0	1	-1	-3	-1	1	-1	0	1	-3
0	-2	0	0	-1	0	-2	-1	-1	0	1	0	0	1	1	0
1	1	0	-1	0	0	-1	-1	0	-1	0	0	0	-1	0	0
-1	0	0	-1	-1	0	0	1	0	-1	1	0	-1	1	0	0
0	-1	1	1	-3	1	-2	1	0	0	0	0	0	0	1	1
-1	-1	0	1	-2	-1	-1	0	1	-1	2	1	0	-1	1	-2

Sigma =1.2

0	0	-2	1	-1	-1	0	2	-1	-1	-1	-1	0	1	1	0
-1	0	-1	1	1	2	2	0	1	-2	1	1	-3	0	0	2
2	1	0	1	0	0	0	-1	0	1	0	1	0	-1	-1	0
-3	2	2	-1	0	-2	2	0	1	-1	0	-2	0	0	0	-2
2	1	-1	0	-2	0	1	-1	0	0	-1	-1	-1	3	1	1
0	-1	-1	1	0	1	-2	1	-3	0	1	2	0	1	0	1
0	3	0	3	1	0	0	1	3	-3	2	-2	0	-1	0	-2
-1	1	0	1	-2	1	0	0	2	0	0	0	-2	0	0	-3

Sigma =1.4

-1	0	-2	1	0	-1	2	-1	-2	-2	1	0	2	-2	1	-1
1	0	1	1	1	1	1	1	0	1	0	0	-1	-1	0	0
2	0	1	-2	0	3	-2	0	-2	1	-1	0	-3	-3	-3	2
1	-2	1	2	0	0	-1	0	-2	-1	0	-1	-1	4	-1	0
-1	-1	1	0	3	-3	1	1	1	0	-3	0	-1	-1	1	-3
-1	1	-2	0	2	1	1	1	0	2	-1	1	0	-1	1	0
-1	1	2	0	-2	3	-1	1	0	-2	0	2	0	2	3	2
-2	3	-1	-2	1	0	-1	-1	-1	0	-2	0	0	2	-2	2
1	-1	-2	4	1	-2	1	2	-2	0	-2	-1	0	1	2	-1

Sigma = 1.6

2	-1	0	-1	-2	3	1	1	2	0	-1	0	2	0	0	-1
1	3	1	1	1	2	0	0	-1	-1	1	-2	-3	-1	0	1
2	1	-1	-2	1	-2	3	1	0	0	-2	1	-1	-1	0	1
-1	-1	2	2	-1	-2	0	0	2	2	1	2	0	1	-3	2
2	0	-2	-1	-2	1	-1	-1	0	2	0	-1	1	2	-2	-1
-1	-2	-1	2	0	1	2	1	-2	2	1	0	2	2	-3	2
3	-2	1	1	1	1	-1	3	1	1	-3	0	-1	0	0	0
1	-2	-1	-2	-2	4	2	0	-1	0	-3	0	0	-2	2	2
-1	4	-2	0	-2	1	1	-1	0	0	0	1	2	-2	2	1

Sigma = 1.8

-1	0	3	4	-2	0	-1	-3	-1	0	0	-1	-1	1	-1	2
0	-1	-3	0	-2	3	-1	3	-2	0	1	-1	-1	1	-4	0
-2	-2	5	1	0	1	-3	2	1	-1	-4	1	3	0	-3	-2
-2	-3	-1	-1	-1	3	2	1	1	1	-2	2	1	3	0	-2
-2	0	-3	1	3	1	2	2	1	1	0	0	-3	-1	-2	1
2	-4	-1	-4	-2	0	3	-2	-1	0	-1	-2	-1	3	-2	2
0	0	1	1	-2	-1	-1	1	-1	-1	-2	-1	0	0	2	-2
2	1	-2	-4	-1	-2	0	1	0	-1	0	-2	-1	-2	2	0
-1	-1	0	-1	2	-2	6	0	-1	0	0	-1	0	1	0	-1
-4	-1	1	-2	-2	0	0	-2	1	-2	4	0	-1	1	-2	0

Sigma = 2.0

-4	0	2	2	-2	1	-5	0	3	3	0	-1	1	3	0	1
1	4	-4	0	-2	-3	1	-1	0	4	0	1	-2	2	-2	4
1	0	-2	0	1	-1	1	1	0	0	2	-2	-4	5	-3	3
1	0	1	-2	2	0	2	4	1	-2	-1	-2	1	2	-4	1
-2	2	2	1	-1	-1	1	0	2	2	3	0	-1	-2	-2	-1
4	-1	2	1	1	-4	0	-1	-3	0	-2	3	-2	2	1	6
-2	0	0	0	0	-2	3	2	-1	1	2	0	3	1	3	-2
3	-1	-1	-1	3	1	3	1	1	-1	0	1	5	-1	0	0
-1	2	-2	-3	0	-1	2	3	-1	5	-1	-1	0	-1	3	-3
1	0	3	2	-1	-1	4	-4	-3	4	1	3	3	-3	3	0