Práctica II. Parte1

1. Esquema VC Estratificada

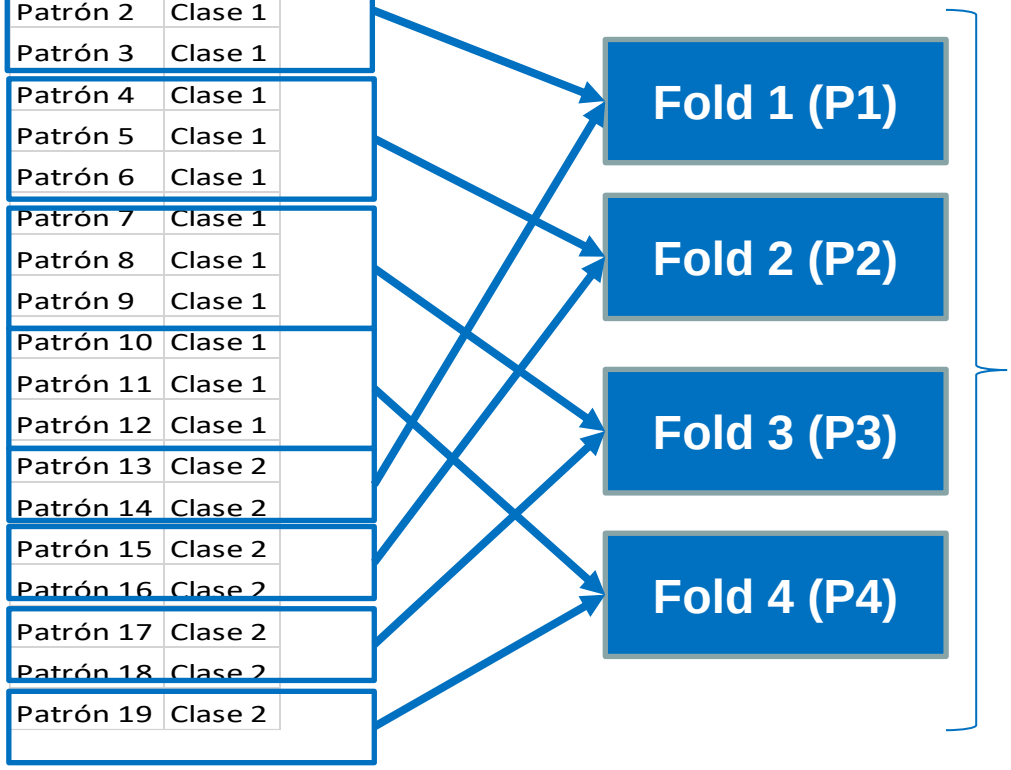
2. Metodología a seguir

3. Script en R

Esquema VC Estratificada

Se ordenan los patrones por clase

Patrón 1	Clase 1
Patrón 2	Clase 1
Patrón 3	Clase 1
Patrón 4	Clase 1
Patrón 5	Clase 1
Patrón 6	Clase 1
Patrón 7	Clase 1
Patrón 8	Clase 1
Patrón 9	Clase 1
Patrón 10	Clase 1
Patrón 11	Clase 1
Patrón 12	Clase 1
Patrón 13	Clase 2
Patrón 14	Clase 2
Patrón 15	Clase 2
Patrón 16	Clase 2
Patrón 17	Clase 2
Patrón 18	Clase 2
Patrón 19	Clase 2



4 parejas de ficheros:

- (P2+P3+P4) train y P1 test
- (P1+P3+P4) train y P2 test
- (P1+P2+P4) train y P3 test
- (P1+P2+P3) train y P4 test

Desordenar los 4 conjuntos de train

Metodología a seguir

Para cada combinación de hiperparámetros (ocultas, RA y ciclos)

Train/test	Error Train	% de aciertos Train	Error Test	% de aciertos Test
P2+P3+P4 P1	-	-	-	-
P1+P3+P4 P2	-	-	-	-
P1+P2+P4 P3	-	-	-	-
P1+P2+P3 P4	-	-	-	-
	media	**media**	**media**	**media**

En este caso se eligirá la mejor combinación utilizando la media en test

Se podría extraer de cada conjunto de train (P2+P3+P4, P1+P3+P4, P1+P2+P4, P1+P2+P3) un conjunto de validación, que podría utilizarse para elegir los mejores hiperparámetros para cada pareja, pero no se utilizará esta metodología en la práctica

Script en R

```
fold <- 1


# usar read.table si los campos están separados por espacios o tabuladores.
# Si están separados por ; o , usar read.csv

#trainSet <- read.table(paste("Train",fold,".txt",sep=""),header = T)
#testSet  <- read.table(paste("Test", fold,".txt",sep=""),header = T)

trainSet <- read.csv(paste("Train",fold,".csv",sep=""),dec=".",sep=",",header = T)
testSet  <- read.csv(paste("Test", fold,".csv",sep=""),dec=".",sep=",",header = T)
```

Se define el fold

Se cargan los datos

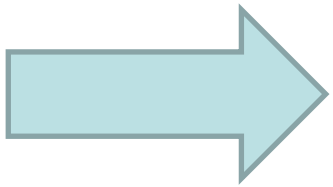
```
#SELECCION DE LA SALIDA. Num de columna del target.
nTarget <- ncol(trainSet)

#SEPARAR ENTRADA DE LA SALIDA
trainInput <- trainSet[,-nTarget]
testInput <-  testSet[,-nTarget]
```

Número de columnas

Este paso es necesario para poder codificar la salida en formato numérico

Script en R

Salida deseada
nube
nube
multinube
cieloDespejado
nube
multinube
cieloDespejado
multinube



Salida deseada		
0	0	1
0	0	1
0	1	0
1	0	0
0	0	1
0	1	0
1	0	0
0	1	0

```
#TRANSFORMAR LA SALIDA DISCRETA A NUMERICA (Matriz con c
trainTarget <- decodeClassLabels(trainSet[,nTarget])
testTarget <-  decodeClassLabels(testSet[,nTarget])
```

Script en R

```
#SELECCION DE LOS HIPERPARAMETROS DE LA RED
topologia        <- c(10)
razonAprendizaje <- 0.01
ciclosMaximos    <- 2000

## generar un nombre de fichero que incluya los hiperparámetros
fileID <- paste("fold_",fold,"_topol",paste(topologia,collapse="-"),"_ra",
               razonAprendizaje,"_iter",ciclosMaximos,sep="")
```

Hiperpámetros

Se genera un nombre de fichero con los hiperparámetros

fold_1_topol10_ra0.01_iter2000

Script en R

```
set.seed(1)
#EJECUCION DEL APRENDIZAJE Y GENERACION DEL MODELO
model <- mlp(x= trainInput,
             y= trainTarget,
             inputsTest= testInput,
             targetsTest= testTarget,
             size= topologia,
             maxit=ciclosMaximos,
             learnFuncParams=c(razonAprendizaje),
             shufflePatterns = F
)

#GRAFICO DE LA EVOLUCION DEL ERROR
#
#plotIterativeError(model)

#TABLA CON LOS ERRORES POR CICLO de train y test correspondientes a las 4 salidas
iterativeErrors <- data.frame(MSETrain= (model$IterativeFitError/nrow(trainSet)),
                              MSETest= (model$IterativeTestError/nrow(testSet)))

graficaError(iterativeErrors)
```

Se entrena la red

Gráfico de evolución en términos de SSE

Gráfico de evolución en términos de MSE

Script en R

```
#GENERAR LAS PREDICCIONES en bruto (valores reales)
trainPred <- predict(model,trainInput)
testPred  <- predict(model,testInput)
```

Salida de la red		
0.63423	0.000013	0.012467
0.05256	0.322300	0.652106
0.93423	0.100013	0.412467
….	…	…

```
#poner nombres de columnas "cieloDespejado" "multinube" "nube"
colnames(testPred)<-colnames(testTarget)
colnames(trainPred)<-colnames(testTarget)
```

Salida de la red		
cieloDespejado	multinube	nube
0.63423	0.000013	0.012467
0.05256	0.322300	0.652106
0.93423	0.100013	0.412467
….	…	…

Script en R

Transformaciones de la salida de la red (si se desea hacerlo)

```
# transforma las tres columnas reales en la clase 1,2,3
#segun el maximo de los tres valores.

trainPredClass<-as.factor(apply(trainPred,1,which.max))
testPredClass<-as.factor(apply(testPred,1,which.max))


#transforma las etiquetas "1", "2", "3" en "cieloDespejado" "multinube" "nube"
levels(testPredClass)<-c("cieloDespejado", "multinube","nube")
levels(trainPredClass)<-c("cieloDespejado", "multinube","nube")
```

Salida de la red		
0.63423	0.000013	0.012467
0.05256	0.322300	0.652106
0.93423	0.100013	0.412467
….	…	…

Salida de la red		
1	0	0
0	0	1
1	0	0
….	…	…

Salida de la red
cieloDespejado
nube
cieloDespejado
….

Script en R

```
#CALCULO DE LAS MATRICES DE CONFUSION
trainCm <- confusionMatrix(trainTarget,trainPred)
testCm  <- confusionMatrix(testTarget, testPred)

trainCm
testCm

#VECTOR DE PRECISIONES
accuracies <- c(TrainAccuracy= accuracy(trainCm), TestAccuracy=  accuracy(testCm))

accuracies
```

Matriz de Confusión y % de aciertos

```
#MODELO
saveRDS(model,              paste("nnet_",fileID,".rds",sep=""))

#tasa de aciertos (accuracy)
write.csv(accuracies,       paste("finalAccuracies_",fileID,".csv",sep=""))

#Evolución de los errores MSE
write.csv(iterativeErrors,paste("iterativeErrors_",fileID,".csv",sep=""))

#salidas esperadas de test con la clase (Target) (última columna del fichero de test)
write.csv( testSet[,nTarget] ,      paste("TestTarget_",fileID,".csv",sep=""), row.names = T

#salidas esperadas de test codificadas en tres columnas (Target)
write.csv(testTarget ,       paste("TestTargetCod_",fileID,".csv",sep=""), row.names = TRUE)

#salidas de test en bruto (nums reales)
write.csv(testPred ,        paste("TestRawOutputs_",fileID,".csv",sep=""), row.names = TRUE)

#salidas de test con la clase
write.csv(testPredClass,    paste("TestClassOutputs_",fileID,".csv",sep=""),row.names = TRUE)

# matrices de confusión
write.csv(trainCm,          paste("trainCm_",fileID,".csv",sep=""))
write.csv(testCm,           paste("testCm_",fileID,".csv",sep=""))
```

Guardando Resultados