Aprendizaje Automático (2016-2017)
Grado en Ingeniería Informática
Universidad de Granada

Cuestionario de teoría 3

David Criado Ramón

1 Considere un modelo de red neuronal con dos capas totalmente conectadas: d unidades de entrada, n_H unidades ocultas y c unidades de salida. Considere la función de error definida por $J(w) = \frac{1}{2}\sum_{k=1}^{c}(t_k - c_k)^2 = \frac{1}{2}||t - z||^2$, donde el vector t representa los valores de la etiqueta, z los valores calculados por la red y w los pesos de la red. Considere que las entradas a la segunda capa se calculan como $z_k = \sum_{j=0}^{N_H} y_j w_{kj} = w_k^t y$ donde el vector y representa la salida de la capa oculta. Usar θ para notar la función de activación

1.a) Deducir con todo detalle la regla de adaptación de los pesos entre la capa oculta y la salida.

1.b) Deducir con todo detalle la regla de adaptación de los pesos entre la capa de entrada y la capa oculta.

2 Tanto "bagging" como validación-cruzada cuando se aplican sobre una muestra de datos nos permiten dar una estimación del error de un modelo ajustado a partir de dicha muestra de datos. Discuta cuál de los dos métodos considera que obtendrá una mejor estimación del error. Especificar con precisión las razones.

En casos extremos (tamaño de muestra pequeño y muy grande) creo que es mejor *bagging*. En otro caso no creo que exista una gran diferencia entre usar uno u otro.

En el caso de ser demasiado pequeño habría que particionar la muestra muy pequeña en subconjuntos todavía más pequeños de ellos por lo que cada uno de los ajustes que se hagan aunque ciertamente sean una cota superior del error que se dará fuera de la muestra estará considerablemente inflado mientras que gracias al muestreo con reemplazamiento que se aplica en *bagging* eso no ocurrirá.

En casos muy grandes incluso usar *validazción cruzada K-fold* puede ser mucho más costoso computacionalmente que *bagging*.

En el resto de los casos, exceptuando árboles de decisión con mucha varianza dónde realizar *bagging* es especialmente potente (gracias a la reducción de la varianza proporcionada por la media) no puedo concluir que uno u otro sea especialmente mejor y probablemente dependerá de las condiciones con las que trabajos y el clasificador que usemos.

3 Considere que dispone de un conjunto de datos linealmente separable. Recuerde que una vez establecido un orden sobre los datos, el algoritmo perceptron encuentra un hiperplano separador iterando sobre los datos y adaptando los pesos de acuerdo al algoritmo. Modificar el pseudo-código del perceptron para adaptarlo a un algoritmo simple de SVM, considerando que en cada iteración adaptamos los pesos de acuerdo al caso peor clasificado de toda la muestra. Justificar adecuadamente/matemáticamente el resultado, mostrando que al final del entrenamiento solo estaremos adaptando los vectores soporte.

4 Considerar un modelo SVM y los siguientes datos de entrenamiento: Clase-1:{(1,1),(2,2),(2,0)}, Clase-2:{(0,0),(1,0),(0,1)}

4.a) Dibujar los puntos y construir por inspección el vector de pesos para el hiperplano óptimo y el margen óptimo.

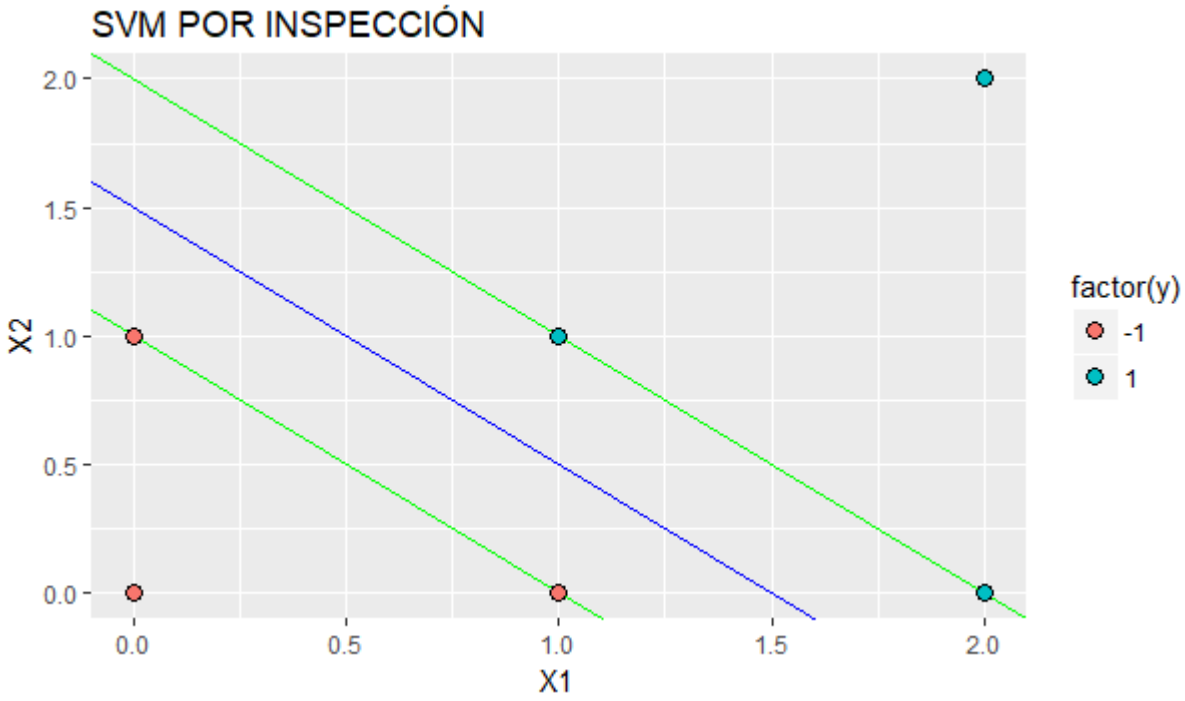
Dicho vector de peso vendría dado por la recta que divide a ambas clases, y puesto que existen cuatro puntos cercanos a esa recta, la recta paralela justo a la mitad entre las rectas formadas por los puntos [(0,1) y (1,0)] y [(1,1) y (2,0). Para ver las ecuaciones de las rectas y pintar las gráficas vamos a usar el siguiente script en R.

```
library(ggplot2)

x = matrix(data = c(1,1,
2,2,
2,0,
0,0,
1,0,
0,1), nrow = 6,ncol = 2, byrow = T)
y = matrix(data = c(1,1,1,-1,-1,-1))

calcula_recta = function(ptos) {
pendiente = (ptos[1,2] - ptos[2,2]) / (ptos[1,1]-ptos[2,1])
corte = ptos[1,2]-pendiente*ptos[1,1]
list(pendiente = pendiente, corte = corte)
}

rd = calcula_recta(x[c(1,3),])
ri = calcula_recta(x[5:6,])

# La recta paralela (que bajo inspección es la que da lugar al vector de pesos)
# es la que tiene la misma pendiente pero el punto de corte está justo en la
```

```
# mitad de ambos
rp = ri
rp$corte = ri$corte + (rd$corte-ri$corte)/2

# Mostramos la gráfica
mostrar_recta = function(r,text) {
cat(text, " es ", r$pendiente, "x + ", r$corte, "\n", sep = "")
}
mostrar_recta(ri, "El margen izquierdo")
mostrar_recta(rd, "El margen derecho")
mostrar_recta(rp, "La recta del ajuste")
```

El margen izquierdo es -1x + 1
El margen derecho es -1x + 2
La recta del ajuste es -1x + 1.5

```
inspeccion <- ggplot(data.frame(x), aes(x=X1, y=X2)) +
ggtitle("SVM POR INSPECCIÓN") +
# Pintamos los margenes de verde
geom_abline(slope = rd$pendiente, intercept = rd$corte, colour = "green") +
geom_abline(slope = ri$pendiente, intercept = ri$corte, colour = "green") +
# Pintamos el vector de pesos de azul
geom_abline(slope = rp$pendiente, intercept = rp$corte, colour = "blue") +
# Pintamos los puntos
geom_point(aes(fill=factor(y)), size=3, pch=21)

print(inspeccion)
```



4.b) ¿Cuáles son los vectores soporte?

Los vectores soporte son los puntos que dan lugar a las líneas de los márgenes, es decir: [(1,0), (0,1), (1,1) y (2,0)].

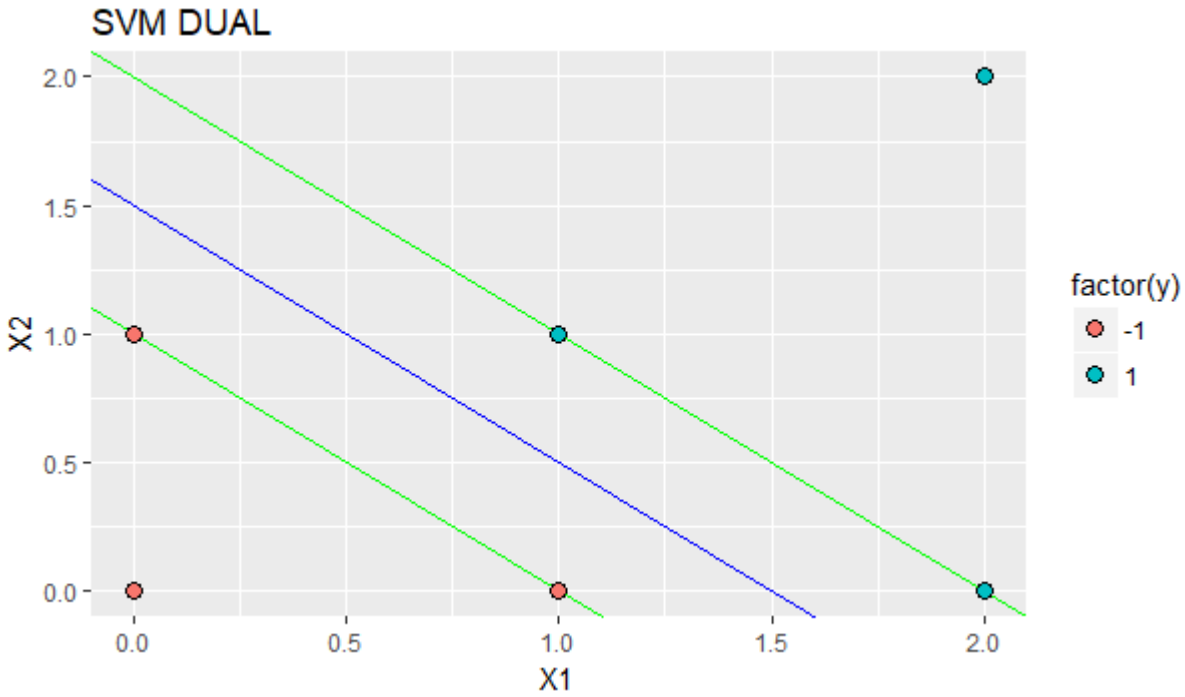
4.c) Construir la solución en el espacio dual. Compara la solución con la del apartado (a).

Para asegurarnos de que la construida en el espacio dual es la correcta vamos a realizar otro script en R para calcularlo utilizando la libería de programación cuadrática *quadprog*

```
library(quadprog) # Libería de programación quadrática
library(ggplot2) # Para pintar gráficas

# Datos de entrada
x = matrix(data = c(1,1,
2,2,
2,0,
0,0,
1,0,
0,1), nrow = 6,ncol = 2, byrow = T)
y = matrix(data = c(1,1,1,-1,-1,-1))

# Epsilon para dejar la matriz definida positivamente y luego comprobar
```

```
# números mayores que 0
eps = 1e-5
# Tamaño de la muestra
n = nrow(x)
# Montamos la matriz Qd
Qd = sapply(1:n, function(i) y[i]*t(x)[,i])
Qd = t(Qd) %*% Qd
# Montamos la matriz fila de unos de tamaño N (Ojo: en la diapositiva pone -1)
unos = matrix(1, nrow = n)
# Montamos la matriz A
A = t(rbind(t(y), t(-y),diag(nrow = n)))
# Montamos la matriz fila de ceros de tamaño N + 2
ceros = rbind(matrix(0,nrow = n+2))

# Usamos la librería de programación cuadrática
solucion = solve.QP(Qd + eps * diag(n), unos, A, ceros )
solucion = matrix(solucion$solution, nrow = n)

# Buscamos el ajuste de la recta
encontrarAjuste <- function(a, y, X){
nozero <-  abs(a) > eps
W <- rowSums(sapply(which(nozero), function(i) a[i]*y[i]*X[i,]))
b <- mean(sapply(which(nozero), function(i) X[i,]%*%W- y[i]))
pendiente <- -W[1]/W[2]
corte <- b/W[2]
list(pendiente = pendiente, corte = corte)
}
rectaDual <- encontrarAjuste(solucion, y, x)


rectaDualPlot <- ggplot(data.frame(x), aes(x=X1, y=X2)) +
ggtitle("SVM DUAL") +
# Pintamos los margenes de verde
geom_abline(slope = rd$pendiente, intercept = rd$corte, colour = "green") +
geom_abline(slope = ri$pendiente, intercept = ri$corte, colour = "green") +
# Pintamos el vector de pesos de azul
geom_abline(slope = rectaDual$pendiente, intercept = rectaDual$corte, colour = "blue") +
# Pintamos los puntos
geom_point(aes(fill=factor(y)), size=3, pch=21)

mostrar_recta(rectaDual, "La recta obtenida en el ajuste del SVM dual")
print(rectaDualPlot)
```

La recta obtenida en el ajuste del SVM dual es -1.00001x + 1.50001



Debido al uso de un epsilon para conseguir que la matriz Qd esté definida positivamente (necesario para el paquete de programación cuadrática usado) podemos observar ligeras diferencias de 1e-5 en el ajuste que obtuve por inspección y este pero concluyo que realmente si no hubiese sido necesario eso ambos habrían dado exactamente el resultado obtenido por inspección que, por tanto, es el mejor hiperplano que separa ambas clases.

5 Una empresa está valorando cambiar su sistema de proceso de datos, para ello dispone de dos opciones, la primera es adquirir dos nuevos sistemas idénticos al actual a 200.000 euros cada uno, y la segunda consiste en adquirir un sistema integrado por 800.000 euros. Las ventas que la empresa estima que tendrá a lo largo de la vida útil de sus equipos son de 5.000.000 euros en el caso positivo, a lo que la empresa le asigna una probabilidad de que suceda del 30 %, en caso contrario, las ventas esperadas son de 3.500.000 euros. ¿Qué opción debería de tomar la empresa?

Tenemos un 70 % de probabilidad de que las ganancias sean 3.500.000 euros y un 30 % de que sean 5.000.000 euros. Por tanto, vamos a calcular la esperanza de los beneficios durante la vida útil.

$0,7 \cdot 3{,}500{,}000 + 0,3 \cdot 5{,}000{,}000 = 3{,}950{,}000$
En principio hay que invertir el doble de dinero para comprar el sistema integrado que

para comprar dos sistemas idénticos al que tienen actualmente. Considero que, a no ser que lo estén cambiando por tener problemas con dicho sistema, es mejor obtener dos sistemas idénticos al actual ya que obtendrían mayores beneficios al tener que invertir menos en el sistema dando un margen a esta opción de que se estropee el sistema dos veces entre ambos, si alguno de los dos sistemas falla tienen otro para no dejar a sus clientes sin servicio y la ventaja de que los trabajadores ya conocen dicho sistema a diferencia del integrado.

6 El método de Boosting representa una forma alternativa en la búsqueda del mejor clasificador con respecto del enfoque tradicional implementado por los algoritmos PLA, SVM, NN, etc.

6.a) Identifique de forma clara y concisa las novedades del enfoque.

En vez de complicar el clasificador para poder conseguir el mejor ajuste posible combina múltiples clasificadores simples para entrenar y dar los resultados.

6.b) Diga las razones profundas por la que la técnica funciona produciendo buenos ajustes (no ponga el algoritmo).

La razón por la que da buenos ajuste es porque el método de entrenamiento de boosting tiende a maximizar el número de muestras de entrenamiento con un margen mayor. Así pues, si aumenta considerablemente el número de iteraciones hasta un punto en el que ajuste a la perfección el error de training podría continuar reduciendo el error de generalización fuera de la muestra buscando clasificadores que sigan clasificando correctamente pero con mayores márgenes. El hecho de que un mayor margen sea mejor, es especialmente bueno por el hecho de los datos ruidosos, ya que, si existe una ligera diferencia en la ubicación en el espacio de una muestra, un gran margen evita que dicho elemento sea clasificado incorrectamente.

6.c) Identifique sus principales debilidades.

Sus debilidades dependen principalmente de los clasificadores débiles.

Si los clasificadores débiles son muy complejos ocurre un sobreajuste debido a la gran varianza generada al usarlos.

Si los clasificadores débiles no son los suficientemente potente ocurre justo lo opuesto. El ajuste obtenido, incluso en entrenamiento, no se aproxima lo suficiente a la función objetivo que desconocemos por lo que el error es alto tanto dentro como fuera de la muestra.

Además, el rendimiento (para entrenar) depende de los clasificadores débiles usados y, por supuesto, el tamaño de la muestra de entrenamiento.

6.d) ¿Cuál su capacidad de generalización comparando con SVM?

Tanto SVM con kernel como boosting son capaces de representar cualquier función y problemas tanto de clasificación como regresión. Mientras que el error de generalización en ambos casos es menor cuanto mayor es el tamaño de la muestra (como cabe esperar), en ambos dicho error también aumenta según la complejidad del método. En SVM, aumenta conforme mayor es el número de vectores soporte. En boosting, aumenta conforme a la raíz cuadrada de la complejidad de los clasificadores simples.

7 ¿Cuál es a su criterio lo que permite a clasificadores como Random Forest basados en un conjunto de clasificadores simples aprender de forma más eficiente? ¿Cuáles son las mejoras que introduce frente a los clasificadores simples? ¿Es Random Forest óptimo en algún sentido? Justifique con precisión las contestaciones

A mi criterio, aprenden de forma más eficiente gracias a que al no buscar un modelo más complejo la varianza no crece tanto como si usamos muchos clasificadores más simples.

Random Forest es eficiente con tamaños de muestra grande, nos da información de cuáles son las variables del vector de características más importantes y el proceso de generación nos va a proporcionar un error de generalización con varianza pero sin sesgo.