Universidad de Granada

Ingeniería Informática

Computación y Sistemas Inteligentes

Cuestionario 3

Autor: José Antonio Ruiz Millán
Asignatura: Aprendizaje Automático
31 de mayo de 2018

1. **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. Enuncie las diferencias y semejanzas entre ambas técncias. Diga cual de ellas considera que nos proporcionará una mejor estimación del error en cada caso concreto y por qué.**

   Primero, para dar unas diferencias basadas en algo, voy a definir qué realiza cada uno de ellos.

   - **Bagging**[2]:

     Dado un conjunto de tamaño n, se crea una **muestra bootstrap** al muestrear n instancias de manera uniforme a partir de los datos (con reemplazamiento). Como el conjunto de datos se muestrea con reemplazo, la probabilidad de que una instancia determinada no se elija después de n muestras es $(1 - 1/n)^n \approx e^{-1} \approx 0{,}368$; El número esperado de instancias distintas del conjunto de datos original que aparece en el conjunto de prueba es, por lo tanto, $0{,}632n$. La estimación de precisión $\epsilon 0$ se obtiene utilizando la muestra de arranque para el entrenamiento y el resto de las instancias para la prueba. Dado un número b el número de muestras de arranque, deje que $\epsilon 0_i$ sea la estimación de precisión para la muestra de arranque i La estimación de arranque de 0,632 se define como:

     $$acc_{boot} = \frac{1}{b} \sum_{i=1}^{b} (0{,}632 \cdot \epsilon 0_i + ,368 \cdot acc_S)$$

     donde acc_S es la estimación de precisión de sustitución en el conjunto de datos completo (es decir, la precisión en el conjunto de entrenamiento). La varianza de la estimación se puede determinar calculando la varianza de las estimaciones para las muestras.

   - **Validación cruzada**[2]:

     Utilizaremos la versión de validación cruzada de k folds. En la validación cruzada k folds, a veces llamada estimación de rotación, el conjunto de datos D se divide aleatoriamente en k subconjuntos mutuamente excluyentes (k folds) $D_1, D_2, ..., D_k$ de tamaño aproximadamente igual. EL inductor se entrena y se prueba k veces; cada vez $t \in \{1, 2, ..., k\}$, se entrena en D/D_t y probado en D_t. La estimación de precisión de validación cruzada es la cantidad total de clasificaciones correctas, dividida por el número de instancias en el conjunto de datos. Formalmente, que D_i sea el conjunto de prueba que incluye la instancia $x_i = (v_i, y_i)$, entonces la estimación de acierto en validación cruzada se define como:

     $$acc_{cv} = \frac{1}{n} \sum_{(v_i, y_i \in D)} \delta(I(D/D_{(i)}, v_i), y_i)$$

     Finalmente lo que busca es que cada una de las particiones esté correctamente distribuida, es decir, que en cada una de ellas aparezcan en mayor medida una igualdad de elementos que corresponden a cada una de las clases.

   Semejanzas:

   - Tanto "bagging" como validación cruzada son computacionalmente costosos, aunque validación cruzada es algo más costoso debido a que realiza k pruebas con los datos para cada una de las divisiones distintas.
   - Ambos se utilizan como una técnica para validar un modelo concreto, aunque en este caso, validación cruzada es utilizado más que "bagging".

- Ambos separan el conjunto de datos en varios subconjuntos para tener un resultado más "real".

Diferencias

- "Bagging" clona el conjunto de datos original y crea varios subconjuntos de datos (con repetición), mientras que validación cruzada coje el conjunto de datos y lo divide en distintas partes.
- Validación cruzada no tiene elementos repetidos en cada una de las particiones, mientras que "bagging" sí.
- El principal objetivo de validación cruzada es estimar el error en el test para intentar obtener el mínimo y así tener un modelo considerablemente correcto, mientras que "bagging" se suele utilizar para muestrear el training, creando varios modelos para reducir la varianza del conjunto de datos original, con el fin de conseguir simular una correcta distribución de los datos.

A mi parecer, creo que validación cruzada al ser una técnica como he comentado para estimar el E_{out}, cuando tenemos un conjunto lo suficientemente grande de datos podremos llegar a obtener unos resultados considerablemente buenos, es por ello, que es una de las técnicas más utilizadas en éste ámbito. Por otra parte, creo que "bagging" nos va a dar un mejor rendimiento cuando el conjunto de datos sea más pequeño, aunque debemos de tener una pequeña precaución, ya que si el conjunto de entrenamiento es muy distinto al de test, tendríamos unos resultados no deseados.

2. **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 interando sobre los datos y adaptando los pesos de acuerdo al algoritmo.**

Algorithm 1 Perceptron

1: **Entradas:**$(\mathbf{x}_i, y_i), i = 1, ..., n, \mathbf{w} = 0, k = 0$
2: **repeat**
3: $k \leftarrow (k + 1) \bmod n$
4: **if** $\text{sign}(y_i) \neq \text{sign}(\mathbf{w}^T \mathbf{x}_i)$ **then**
5: $\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i$
6: **end if**
7: **until** todos los puntos bien clasificados

Modificar este pseudo-código 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/matematicamente el resultado, mostrando que al final del entrenamiento solo estaremos adaptando los vectores soporte.

Como nos dice el enunciado, utilizaremos el caso del peor clasificado, por lo que podemos definir este punto como el punto que corresponde al siguiente mínimo, siendo x_i el vector columna correspondiente al punto i e $y_i \in \{-1, 1\}$ la etiqueta del punto i.

$$min_i y_i(x_i^T w)$$

Con esto, los elementos que estén mal clasificados tendrán valores ≤ 0 y si todos están bien clasificados devolverá el valor del punto más próximo a la frontera de decisión. Si aplicamos esto al algoritmo perceptron que nos muestran en el enunciado, tendremos ahora un algoritmo que intentará aplicar transformaciones sobre w para intentar alejarlo de los vectores más proximos a la frontera óptima:

Algorithm 2 Perceptron-SVM

1: **Entradas:**$(\mathbf{x}_i, y_i), i = 1, ..., n, \mathbf{w} = 0, k = 0$
2: **repeat**
3: $k \leftarrow (k + 1) \bmod n$
4: $i \leftarrow$ indice del peor clasificado
5: $\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i$
6: **until** condicion de parada

Podemos comprobar que he eliminado la condición del signo dentro del algoritmo, esto es porque si hay alguno mal clasificado, el mínimo corresponderá necesariamente al peor de éstos y si no empujará la frontera en la dirección del punto óptimo. Como podemos comprobar, la condición de parada también la he modificado, esto se debe a que ahora con la condición de que todos los puntos estén bien clasificados es insuficiente, ya que ahora el objetivo es maximizar el margen entre la frontera de decisión y los vectores soporte, cuantas más iteraciones tengamos, mas póximo estará el punto medio entre los vectores soporte con la frontera de decisión.

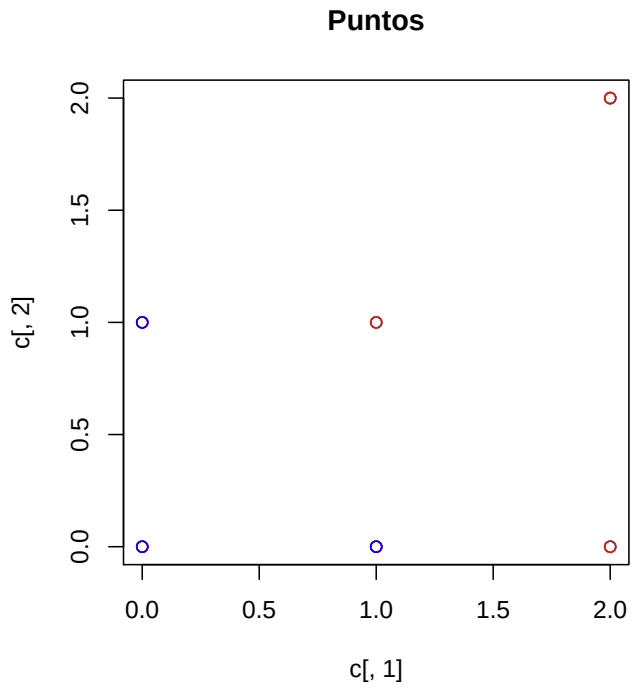
Lo que conseguimos con esto es que una vez tenemos todos los puntos bien clasificados, el resultado de

$$min_i y_i(x_i^T w)$$

son aquellos que verifican la igualdad $y_n(\mathbf{w}^T \mathbf{x}_n + b) = 1$, con $n = 1, ..., N$ en SVM, pero no sabemos el valor de b. Si tenemos $y_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1$, con $n = 1, ..., N$ en SVM, si todos los puntos están bien clasificados, donde almenos 1 de ellos satisface la igualdad, el menor valor de $y_n(\mathbf{w}^T \mathbf{x}_n + b)$ es 1, pero como no sabemos el valor de b, el menor valor de $y_n(\mathbf{w}^T \mathbf{x}_n)$ es un vector soporte. El algoritmo mostrado anteriormente, al clasificar todos los puntos correctamente irá iterando por los distintos vectores soporte de tal forma que irá empujando la frontera en sentido opuesto debido a la actualización de los propios pesos ($\mathbf{w} = \mathbf{w} + y_i \mathbf{x}_i$)

3. **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)}.**

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



Se puede ver claramente en la imagen donde estaría el margen y frontera para separa correctamente estos puntos. Podemos apreciar que tenemos cuatro puntos que son los más conflictivos. Éstos son el (0,1) y (1,0) de la clase 2 y el (2,0) y (1,1) de la clase 1. Podemos observar que las líneas que unen ambos puntos de cada clase tienen la misma pendiente, por lo que la frontera será una misma línea pero dejando ambas a la misma distancia.

Para la clase 2, el margen es $y = 1 - x$ y para el margen de la clase 1, $y = 2 - x$. La recta en la que se encuentra la frontera es $y = \frac{3-2x}{2} = 1{,}5 - x$, vamos a obtener ahora $\mathbf{w}$ resolviendo el sistema de ecuaciones con dos puntos conocidos y la restriccion de la minima distancia entre cualquier vector soporte y el hiperplano sea igual a 1.

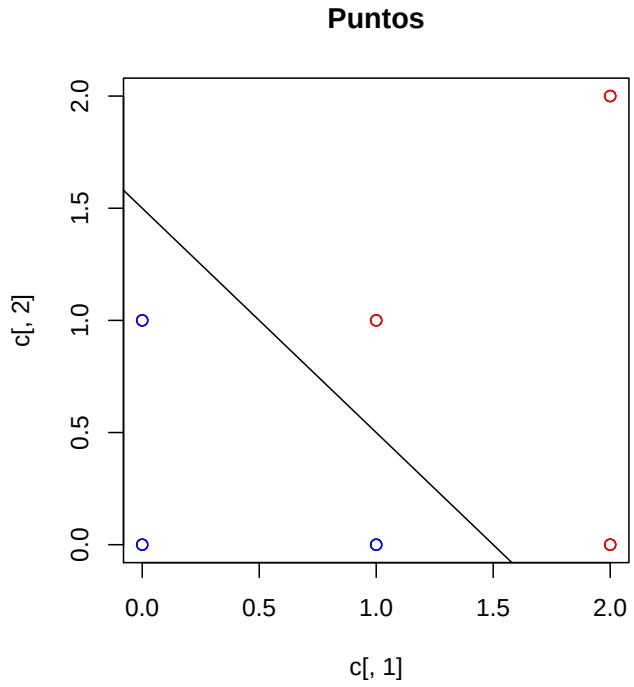
Definimos $\mathbf{w}' = (w_0, w_1, w_2)$ y $\mathbf{x} = (1, x_1, x_2)$, conocemos que en (0,1,5) $\mathbf{w}' = 0$, en (1,5,0) $\mathbf{w}' = 0$ y que en (1,1) debe ser exactamente 1.

$$\begin{cases} w_0 + 1{,}5w_2 = 0 \\ w_0 + 1{,}5w_1 = 0 \\ w_0 + w_1 + w_2 = 1 \end{cases}$$

Finalmente obtenemos que $\mathbf{w}' = (-3, 2, 2) = (-1{,}5, 1, 1)$

Nos quedamos con la forma de la derecha pues sólo buscamos la dirección del vector normal al hiperplano y ya tenemos el vector de pesos para SVM $\mathbf{w}$, compuesto por las componentes w_1 y w_2.

$$\mathbf{w} = (1, 1)$$

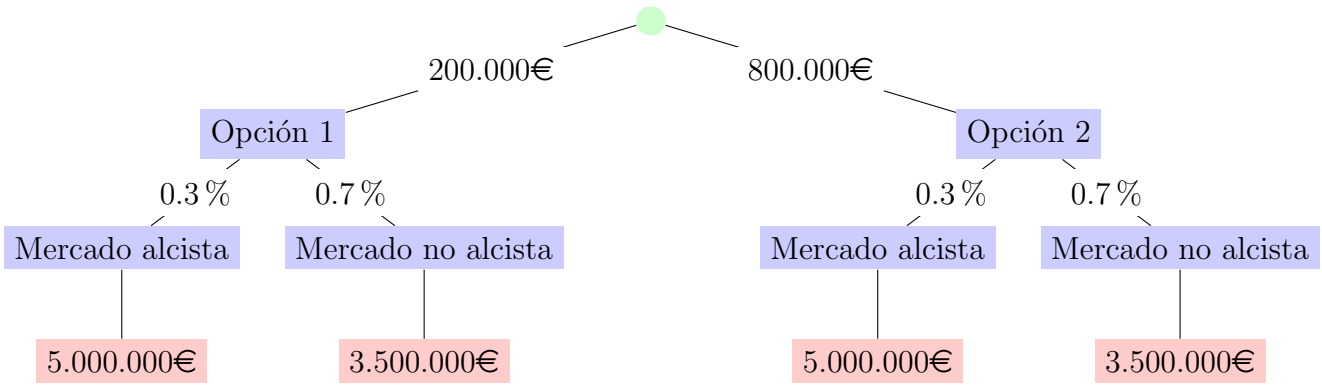


b) **¿Cuáles son los vectores soporte?**

Hay cuatro puntos que pueden ejercer de vectores de soporte y que cumplen la igualdad $y_n(\mathbf{w}^T\mathbf{x}_n + b) = 1$, con $n = 1, ..., N$, por lo tanto, los cuatro son vectors soporte, ya que se encuentran en la frontera del margen y están a distancia 1 de la frontera de decisión, aunque cualesquiera tres de ellos son suficientes para calcular la $\mathbf{w}$

c) **Construir la solución en el espacio dual. Comparar la solución con la del apartado (a)**

4. **Una empresas esta valorando cambiar su sistema de proceso de datos, para ello dispone de dos opciones, la primera es adquirir un nuevo sistema compuesto por dos sistemas idénticos al actual a 200.000 euros cada uno, y la segunda consiste en adquirir un nuevo sistema mucho mayor por 800.000 euros. Las ventas que la empresa estima que tendrá a lo largo de la vida útil de cualquiera de sus nuevos equipos es de 5.000.000 de euros en el caso de un mercado alcista, 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. Construir el árbol de decisiones y decir que opción es la más ventajosa para la empresa.**



Voy a calcular ahora el beneficio de cada rama (de izquierda a derecha en la imagen).

- **Rama 1:** $5{,}000{,}000 - 200{,}000 = 4{,}800{,}000$
- **Rama 2:** $3{,}500{,}000 - 200{,}000 = 3{,}300{,}000$
- **Rama 3:** $5{,}000{,}000 - 800{,}000 = 4{,}200{,}000$
- **Rama 4:** $3{,}500{,}000 - 800{,}000 = 2{,}700{,}000$

Como la última división del árbol es probabilística, calculamos la esperanza matemática de beneficios para cada opción.

- **Opción 1:** $4{,}800{,}000 \cdot 0{,}3 + 3{,}300{,}000 \cdot 0{,}7 = 3{,}750{,}000$
- **Opción 2:** $4{,}200{,}000 \cdot 0{,}3 + 2{,}700{,}000 \cdot 0{,}7 = 3{,}150{,}000$

Por lo que finalmente, debemos seleccionar la *Opción 1* que es la de renovar el sistema por uno de 200,000€para obtener una esperanza de beneficio mayor.

5. **¿Que algoritmos de aprendizaje no se afectan por la dimensionalidad del vector de características? Diga cuáles y por qué.**

6. **Considere la siguiente aproximación al aprendizaje. Mirando los datos, parece que los datos son linealmente separables, por tanto decidimos usar un simple perceptron y obtenemos un error de entrenamiento cero con los pesos óptimos encontrados. Ahora deseamos obtener algunas conclusiones sobre generalización, por tanto miramos el valor d_{vc} de nuestro modelo y vemos que es $d + 1$. Usamos dicho valor de d_{vc} para obtener una cota del error de test. Argumente a favor o en contra de esta forma de proceder identificando los posible fallos si los hubiera y en su caso cual hubiera sido la forma correcta de actuación.**

   Con lo estudiado en el temaria durante todo el curso, puedo detectar varios errores en los procedimientos que se han especificado en el enunciado.

   Comenzando por el principio, ya nos están diciendo que las conclusiones que se sacan para obtener una clase de funciones y un algoritmo de aprendizaje se hacen basándose en los datos. Éste es un error claramente, ya que hemos aprendido que debemos seleccionar la clase de funciones sin ver los datos para evitar escoger unos métodos incorrectos. Nos dicen que se "aprecia" que los datos son linealmente separables y por ello se decide usar el perceptron. Partiendo de que como he comentado, mirar los datos antes de elegir la clase de funciones es un error, otro error es basándonos en ellos, escoger el perceptron porque "parece" que los

datos son linealmente separables. No sabemos nada sobre la muestra, es decir, no sabemos si la muestra es representativa y si es lo suficientemente grande como para poder decirnos algo sobre estos datos. Por lo que si escogemos como en este caso el perceptron porque "parecen" linealmente separables, como no nos dicen nada más sobre la muestra, puede ser que finalmente no tengan nada que ver los datos reales con esta muestra y sean practicamente imposible de separar linealmente.

Por otra parte, nos dice que utilizando esos métodos (que son erróneos), consigue obtener un 0 % de error. Claramente esto nos está indicando un sobreajuste a los datos, lo que nos puede producir que en el futuro, cuando tengamos de verdad que clasificar datos reales, podamos llegar a obtener unos resultados nefastos. Para evitar este tipo de problemas, podemos usar métodos de regularización que nos meten un cierto error a la hora de clasificar y así evitar el sobreajuste a una muestra o usar validación cruzada para tener un porcentaje más real.

Finalmente, utilizan el d_{vc} para obtener una cota de error del test, pero este d_{vc} se obtiene de todos los pasos anteriormente comentados, por lo que ya vienen contaminados desde un principio. Por lo que aunque el método utilizado para obtener una cota de error del test es correcto, obtendremos unos resultados no fiables ya que todos los pasos anteriores se han realizado erroneamente y esto hace que éste último paso tampoco tengo un buen final.

7. **Discuta pros y contras de los clasificadores SVM y Random Forest (RF). Considera que SVM por su construcción a través de un problema de optimización debería ser un mejor clasificador que RF. Justificar las respuestas.**

   Primero comenzaré por el SVM. EL SVM es un modelo lineal cuyo objetivo es encontrar el hiperplano separador que esté mas alejado de ambas clases, es decir, que separe lo máximo posible las clases pero que a su vez, haya la máxima separación entre la frontera y el elemento perteneciente a cada una de las clases. El máximo grosor lo conseguiremos cuando el hiperplano esté a la misma distancia de una clase que de otra.

   SVM:

   a) Pros:

   - Este método propone encontrar el hiperplano óptimo mediante un problema de optimización, por lo que si el problema es linalmente separable en el que no haya un ruido que entorpezca el ajuste, este modelo obtendrá el óptimo y no podrá ser superado por otro método en lo que a certeza se refiere.
   - De todos los puntos que tengamos, solo son necesarios los vectores de soporte (puntos mas cercanos a la frontera de clasificacion de cada clase) ya que el hiperplano del SVM se calcula a través de éstos que son los que realmente afectan.
   - Un pro claramente es el funcionamiento del mismo, es decir, consigue separar linealmente como cualquier modelo lineal pero a su vez, el rango que hace entre las distintas clases es el máximo posible, esto hace que el modelo sea mejor que un modelo lineal simple ya que conseguimos hacerlo más general y que sea capaz de clasificar mejor datos que no ha visto. Esto también nos permite ser más tolerantes al ruido ya que al tener el mayor margen, podemos soportar más ruido que otro modelos.
   - SVM permite convinarse con la herramienta Kernel, que nos brinda una forma

eficiente de utiliazar transformaciones no lineales de alta dimensión, consiguiendo una pontencia de aprendizaje bastante elevada.

b) Contras:

- Si tomamos SVM como lo que es, al crear un hiperplano, tenemos un modelo que es capaz de separar el espacio en dos partes, lo que nos dice que este modelo esta orientado a clasificacion binaria. Existen técnicas para adaptarlo a problemas multiclase pero ya tendríamos un aumento de complejidad y dificultad.
- Un problema del SVM es simplemente que pertenece a la clase de modelos lineales, es decir, sólo podrá separar correctamente si los datos que tenemos son linealmente separables. Ésto, en un problema normalmente no sucede. A veces es necesario aplicar transformaciones no lineales sobre lo datos para poder clasificarlos correctamente, por lo que donde métodos que no son lineales tendrían una facilidad grande para clasificar los datos, el SVM tiene una capacidad reducida al no poder realizar estas transformaciones en los datos.
- Un problema claro es el propio desconocimiento de los vectores de soporte antes de analizar los datos, ya que éstos son realmente importantes para poder realizar el hiperplano óptimo.
- Este método se diseño originalmente para trabajar con problemas numéricos donde se puede sacar su máximo partido. Sin embargo, para problemas no numéricos parte con algo de desventaja contra métodos preparados para ello.
- Hemos hablado en los pros de la posible utilización de la herramienta Kernel, y esto nos lleva a la contra de la elección del Kernel apropiado para el modelo y para obtener el máximo beneficio posible.

Random Forest

Pasamos ahora a analizar el Random Forest. Este método es un multiclasificador compuesto por varios clasificadores simples (árboles de decisión), cuta predicción final se obtiene poniendo en común las predicciones de cada uno de los clasificadores simples que lo componenen.

a) Pros:

- Uno de los pros más improtante que hace que este método tenga un buen resultado, es que consigue que cada uno de los clasificadores simples que contiene no tengan correlacion entre ellos, lo que hace es que todos ellos sean diferentes entre sí para así tener diversidad y poder estudiar distintos escenarios para finalmente sacar una conclusión sobre todos ellos.
- Este método consigue unos buenos resultados teniendo en cuenta la simpleza del mismo, ya que está compuesto por clasificadores simples y únicamente añadiendo algunos parámetros para nada costosos ni difíciles, consigue unos resultados notoriamente satisfactorios en cuanto a relacion certeza-simpleza se refiere (Navaja de Ockham).
- Lleva incorporado en el propio proceso una "validacion cruzada" que la realiza mediante la técnica de remuestreo, ya que ésta le permite el uso del estimador de error *out-of-bag*, el cual aprobecha los datos sobre los que no se ha entrenado

un determinado árbol para hacer una estimación del E_{out}. En conjuntos grandes de datos llega incluso a ser más eficiente computacionalmente que la validación cruzada.

- En un modelo no lienal, lo que le proporciona una mejora significativa respecto a los modelos lineales que sólo pueden ser utilizados en esos casos.
- La forma que tiene de elegir las variables en cada nodo de los árboles nos reduce en gran medida el número de variables a elegir, por lo que nos proporciona muchas ventajas como por ejemplo trabajar con conjunto de datos de alta dimensión, sin tener que hacer un filtro previo de varaibles, ya que en cada paso elige unsubconjunto pequeño de ellas.

b) Contras:

- Uno de los contras que he encontrado es la interpretación del modelo final, es decir, todos los clasificadores que contiene son simples y fáciles de interpretar pero cuando queremos tener el modelo final, donde utiliza la combinación de todos ellos, la interpretación del modelo se hace más compleja.

Respecto al último apartado que nos comenta el ejercicio, SVM ajusta un hiperplano lineal que será optimo simpre que los datos sean linealmente separables, que como he comentado en apartados anteriores es un caso poco común en la realidad. La calidad del RF no depende esencialmente de si los datos son linealmente separables. Para que SVM pueda afrontar problemas que no son linealmente separables debemos de aplicar modificaciones que tienen un costo adicional, que no garantiza que finalmente sea posible la clasificación. Por lo tanto, afirmar que SVM es mejor clasificador que RF es aventurarnos demasiado ya que ésto no siempre será así, dependerá bastante de los datos con los que estemos trabajando y de cómo se puedan representar.

8. **¿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? ¿Cuales son las mejoras que introduce frente a los clasificadores simples? ¿Es Random Forest óptimo en algún sentido? Justifique con precisión las contestaciones.**

a) Para mi parecer, la gran ventaja que ofrece Random Forest es básicamente su funcionamiento, es decir, el hecho de usar multiclasificadores para obtener una diversidad y una generalización que un modelo individual por sí solo tiene más dificultad. Al estar formado por un conjunto de clasificadores independientes, cada uno de ellos puede estar entrenando en escenarios distintos, lo que hace que a la hora de obtener un modelo final, hemos estado entrenando en distintintas situaciones, lo que hace que sea más complicado encontrarse una situación en el exterior que ya no conozcamos o que hayamos estado cerca. Además Random Forest obliga a que los distintos clasificadores simples no estén correlacionados entre sí para así obtener una diversidad mayor.

b) Una de las mejoras es la comentada en el apartado anterior. Lo que esto conlleva utilizar algunos métodos como realizar la media de todos los clasificadores simples en caso de la regresión o el voto mayoritario en caso de clasificación. Con esto conseguimos reducir la varianza de los datos ya que si cada uno contempla un escenario distinto, es probable que la varianza sea bastante amplia, por ello, si realizamos la media entre ellos, conseguimos agrupar y disminuir esta varianza de los datos.

c) Para hablar de optimalidad, debemos saber como funciona el Random Forest, ya que partimos de que la generación de aleatorios en grandes cantidades no es óptima. Partiendo de esto, Random Forest se basa en la aleatoriedad en la toma de algunas decisiones para así ganar diversidad y obtener un modelo final bastante general y adaptrativo a diferentes escenarios. Para entrenar a cada uno de sus árboles simples, se basa en la elección aleatoria con repetición de instancias del conjunto de datos original. También a la hora de elegir el subconjunto de variables que se tendrán en cuenta a la hora de seleccionar una variable para cada nodo de los árboles simples, utiliza aleatoriedad. Esto hace que al utilizar tanta aleatoriedad en el proceso de aprendizaje, no podamos hablar de optimalidad del mismo ya que como he comentado anteriormente, la generación aleatoria es costosa si se utiliza constantemente.

9. **En un experimento para determinar la distribución del tamaño de los peces en un lago, se decide echar una red para capturar una muestra representativa. Así se hace y se obtiene una muestra suficientemente grande de la que se pueden obtener conclusiones estadísticas sobre los peces del lago. Se obtiene la distribución de peces por tamaño y se entregan las conclusiones. Discuta si las conclusiones obtenidas servirán para el objetivo que se persigue e identifique si hay algo que lo impida.**

   Nos dicen que obtenemos una muestra suficientemente grande, pero no nos dicen nada sobre la distribción de la propia muestra, es decir, puede ser que aunque la muestra sea muy grande, al echar la red, caiga en un banco de peces muy grande pero donde todos los peces o la mayoría de ellos son del mismo tipo, por lo que obtendríamos un resultado sesgado.

   Podemos suponer que lo dicho anteriormente se cumpla, por lo que tendríamos una muestra totalmente representativa y bien distribuida. El ejercicio nos dice que entrega unas conclusiones basándose en resultados estadísticos sobre los peces del lago, pero estamos dando unas conclusiones concretas y fijas, es decir, no estamos dando un margen de error, por lo que aquí tenemos otro error. Nosotros debemos obtener unas conslusiones que podemos asegurar asumiendo un cierto error δ, donde se debe de cumplir que dado un d_{vc} y un δ:

   $$E_{out}(h) \leq E_{in}(h) + \sqrt{\frac{8}{N}\log\left(\frac{4((2N)^{d_{vc}}+1)}{\delta}\right)}$$

   Por lo que finalmente deberíamos solucionar estos errores para que las conclusiones finales sean válidas.

10. **Identifique dos razones de peso por las que el ajuste de un modelo de red neuronal a un conjunto de datos puede fallar o equivalentemente obtener resultados muy pobres. Justifique la importancia de las razones expuestas.**

    Hay varias razones para ello, pero en mi parecer, creo que las dos razones con más peso serían:

    - **Criterio de parada:**

      Este criterio es muy importante, ya que si no definimos una buena condicion de parada, podemos sobrepasarnos y con ello obtener un sobreajuste, lo que puede conllevar a obtener unos resultados muy pobres en los test o al contrario, si la condición de parada hace que la red pare antes de lo necesario, aprendiera mucho menos de lo que tenemos y esto nos lleva de nuevo a unos resultados muy pobres.

Algunas condiciones de parada podrían ser:

- Número concreto de iteraciones.
- Valor de E_{in}.
- Tamaño del gradiente.
- Conbinación de varios de estos criterios (recomendable).

- **Inicializacion de los pesos**:

  La segunda razón para mí sería la inicialización de los pesos, ya que esta puede tener un gran impacto para obtener unos resultados de calidad en la red. La forma de proceder con este parámetro y sus posibles erratas son:

  - Lo recomendable es inicializar los pesos con valores cercanos a cero. Por ejemplo, podemos usar una distribución normal con varianza inversamente proporcional al número de entradas para cada neurona.
  - El inicializar los pesos a 0 nos llevaría a un aprendizaje nulo ya que la red no aprendería nada ya que todas las neuronas realizan el mismo cálculo dando el mismo resultado.
  - Tampoco inicializar los pesos a valores muy grandes, ya que la función sigmoidal se saturaría y como consecuencia de tener estos valores tan grandes en el gradiente, se podrían ocasionar desbordamientos dando lugar a cálculos incorrectos.

Referencias

[1] Learning from Data Short Course, Yser S. Abu-Mostafa; Malik Magdon-Ismail; Hsuan-Tien Lin, `https://libsvm.com/data/learn.from.data.pdf`, Accedido el 31 de mayo de 2018.

[2] A study of cross-validation and bootstrap for accuracy estimation and model selection, In Proceedings of the 14th International Joint Conference on Artificial intelligence (IJCAI ' 9 5), volume 2, pages 1137 1 143, 1995. R. Kohavi, `http://robotics.stanford.edu/~ronnyk/accEst.pdf`, Accedido el 31 de mayo de 2018.