UNIVERSIDAD
DE GRANADA

Aprendizaje Automático
grado en ingeniería informática

TRABAJO 3

Cuestiones de Teoría

Autor
Vladislav Nikolov Vasilev

Rama
Computación y Sistemas Inteligentes

Escuela Técnica Superior de Ingenierías Informática y de
Telecomunicación

Curso 2018-2019

Índice

Ejercicio 1

¿Podría considerarse Bagging como una técnica para estimar el error de predicción de un modelo de aprendizaje? Diga si o no con argumentos. En caso afirmativo compárela con validación cruzada.

Solución

Bagging se podría considerar como una técnica para estimar el error de predicción de un modelo de aprendizaje. Esto se debe a que, a partir de una muestra, se eligen mediante *bootstrapping* un número B de conjuntos de entrenamiento (es decir, conjuntos en los que hay datos con repetición), y se entrenan B árboles con estas muestras de entrenamiento. El resto de elementos de la muestra que no han sido escogidos para formar las muestras de entrenamiento se utilizan como test (es decir, para cada árbol, se mira cuáles son los elementos de la muestra original que no han sido utilizados en el entrenamiento, y se escogen posteriormente para hacer el test y obtener alguna métrica del error o de la precisión del árbol de manera individual). Finalmente, se realiza la media con todos los resultados obtenidos para cada modelo.

Esto que se ha descrito en el párrafo anterior es muy parecido a lo que se hace con la validación cruzada. Ambos tienen una parte de los datos con la que entrenan el modelo y una parte con la que obtienen alguna métrica sobre el modelo, para luego juntar todas las métricas, hacer una media de éstas y obtener un valor que nos permita estimar el error de predicción del modelo.

Sin embargo, existen algunas diferencias:

- En la validación cruzada no hay datos repetidos ya que se hacen una serie de k particiones disjuntas. En cambio, al utilizar *bootstrapping* en Bagging, al escoger datos de la muestra con repetición, existen muy altas probabilidades de que se repita algún dato.
- En la validación cruzada se hacen k particiones disjuntas y se prueba el mismo modelo con todas ellas agrupando los datos en una parte de training y una de test, haciendo que la partición de test sea cada vez diferente para las k particiones que se han creado. En cambio, con Bagging se entrenan una serie de B modelos con un solo conjunto de entrenamiento cada uno y se prueba con un conjunto de test solo. Es decir, en vez de tener un único modelo que se va entrenando con cada una de las particiones de entrenamiento, se tienen B modelos.
- En la validación cruzada se puede elegir qué parte de los datos estará en la parte de test, al poder elegir el tamaño de las particiones. En cambio,

en Bagging aproximadamente 2/3 de los datos de la muestra original serán usados como training, mientras que aproximadamente el 1/3 restante será utilizado para test para cada modelo.

Ejercicio 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

Entradas: $(\mathbf{x}_i, y_i) = 1, \ldots, n\ ,\ w = 0,\ k = 0$
repeat
  $k \leftarrow (k+1) \mod n$
  if $\text{sign}(y_i) \neq \text{sign}(\mathbf{w}^T\mathbf{x}_i)$ **then**
    $\mathbf{w} \leftarrow \mathbf{w} + y_i\mathbf{x}_i$
  end if
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.

Solución

Algorithm 2 Perceptron adaptado a SVM

Entradas: $(x_i, y_i), i = 1, ..., n;\ w = 0$
repeat
  $peor \leftarrow 1$
  for $k \leftarrow 1$ **to** n **do**
    if $y_k\mathbf{w}^T\mathbf{x}_k < y_{peor}\mathbf{w}^T\mathbf{x}_{peor}$ **then**
      $peor \leftarrow k$
    end if
  end for
  if $y_{peor}\mathbf{w}^T\mathbf{x}_{peor} < 1$ **then**
    $\mathbf{w} \leftarrow \mathbf{w} + y_{peor}\mathbf{x}_{peor}$
  end if
until condición de parada

Con el pseudocódigo ya visto, vamos a intentar justificar brevemente el por qué de cada cosa. SVM, al igual que hace el Perceptrón, comprueba si los signos del valor predicho y el real se corresponden. Esto se hace mediante el producto $y_n \mathbf{w}^T \mathbf{x}_n$. Si los signos coinciden, el producto es positivo, y si no, es negativo. Sin embargo, en SVM también se intenta maximizar el margen que se deja a cada lado del hiperplano. Los vectores soporte siempre se encontrarán encima del margen, y por tanto para ellos se da que $y_n \mathbf{w}^T \mathbf{x}_n = 1$. Para el resto de puntos que están más allá de los márgenes, el valor del producto será siempre mayor estricto que 1, independientemente de a qué lado caiga. Sin embargo, si un punto cae dentro del margen, su distancia será menor que 1.

El algoritmo recorre todos los puntos y busca aquél cuyo producto $y_n \mathbf{w}^T \mathbf{x}_n$ sea el más pequeño, el cuál considerará como el peor clasificado. Después de eso, comprueba si es menor que 1, y en caso de serlo, modifica $\mathbf{w}$ en una cierta cantidad. Aquél punto que se considere el peor clasificado será o bien uno que esté **incorrectamente clasificado** y esté muy alejado del hiperplano, o bien uno que, estando todos los otros puntos correctamente clasificados, éste se encuentre dentro del margen, siendo por tanto su distancia al hiperplano menor que 1. De esta forma, al principio el hiperplano se irá ajustando para que todos los puntos estén bien clasificados, ya que siempre el peor clasificado será uno que no haya sido clasificado correctamente. Después de eso, los peores clasificados serán aquellos que caigan dentro del margen, y por tanto, el hiperplano irá iterando sobre los vectores soporte hasta encontrar los correctos y conseguir ajustarse correctamente. Con lo cuál, como resumen, primero se irá situando el hiperplano en un lugar adecuado, y una vez hecho esto, se irá ajustando para maximizar el margen. Es importante decir que en este algoritmo no se contempla que los datos no sean **linealmente separables** y que haya puntos que no se puedan extraer del margen. Para ello, habría que considerar una tolerancia de violación del margen (un valor de error) y ver cuál es su valor para todos los puntos.

Ejercicio 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.

Solución

Primero vamos a dibujar los puntos para ver como se ditribuyen en el espacio:

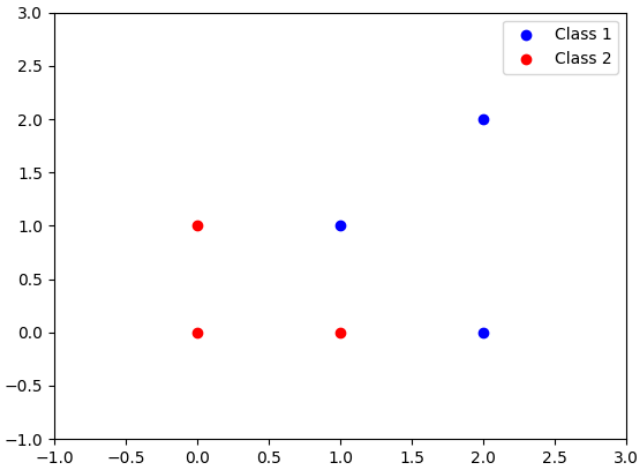

Figura 1: Dibujo con los puntos de las dos clases en el espacio.

Se puede ver claramente que los puntos de los dos clases son linealmente separables, ya que perfectamente se pueden separar mediante un hiperplano que pase en medio de ellas.

Para intentar obtener un hiperplano óptimo, vamos a suponer que éste tiene que pasar entre los puntos de las dos clases que estén más cerca entre sí (es decir, que tiene que pasar entre los vectores soporte). Estos puntos son, para la Clase-1, el $(1, 1)$ y el $(2, 0)$, y para la Clase-2 son el $(0, 1)$ y el $(1, 0)$. Por tanto, sabiendo que el hiperplano óptimo tiene que pasar entre estos puntos, dejando la mayor cantidad de margen a cada lado, podemos suponer que pasará justo en el punto medio para cada par de puntos que están a la misma altura y son de clases diferentes. Es decir, que para los puntos $(0, 1)$ y $(1, 1)$ (los cuáles son de diferente clase), sabemos que seguramente ese hiperplano pasará por el $(0.5, 1)$. Para los dos puntos de abajo, sabiendo que tiene que pasar entre los puntos $(0, 0)$ y $(1, 0)$, seguramente pasará por el punto $(1.5, 0)$, el cuál está justo en medio de los dos anteriores. Esto se puede ver mejor en la siguiente imagen:

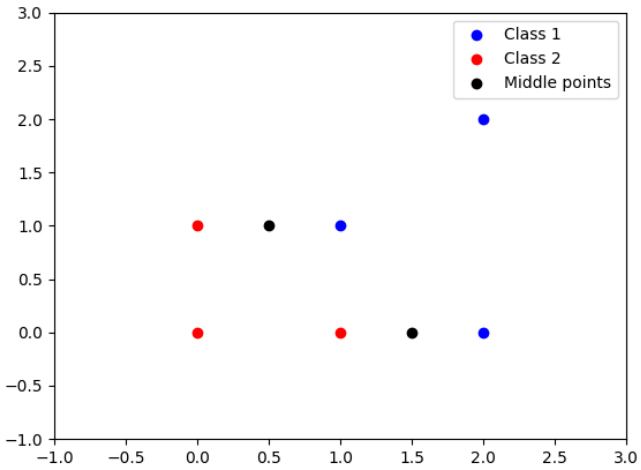

Figura 2: Dibujo con los puntos medios entre los vectores soporte de las dos clases, representados en negro.

Ahora lo único que nos queda es obtener el hiperplano que separa las dos clases. Como ya sabemos los puntos por los que puede pasar, lo único que tenemos que obtener es la recta que pasa por esos dos puntos. Para eso podemos partir de la ecuación de la recta, la cuál viene dada por la forma:

$$y = ax + b \tag{1}$$

donde a es la pendiente de la recta y b el término independiente. Sustituyendo los valores de los puntos por x e y en la expresión dada por (1), obtenemos el siguiente sistema de ecuaciones:

$$\left.\begin{aligned} 1 &= 0.5a + b \\ 0 &= 1.5a + b \end{aligned}\right\} \tag{2}$$

Ahora resolvemos el sistema de ecuaciones para obtener la solución:

$$\left.\begin{aligned} 1 - 0.5a &= b \\ -1.5a &= b \end{aligned}\right\} \tag{3}$$

$$\begin{aligned} 1 - 0.5a &= -1.5a \\ 1 &= -a \end{aligned} \tag{4}$$

De aquí, obtenemos que:

$$\left.\begin{aligned} a &= -1 \\ b &= 1.5 \end{aligned}\right\} \tag{5}$$

Y finalmente, con los resultados obtenidos en (5), sustityendo en la expresión dada en (1), obtenemos que la ecuación de la recta es la siguiente:

$$y = -x + 1.5 \tag{6}$$

Esta recta es, en un principio, el hiperplano óptimo que separa las dos clases. Para obtener los márgenes, lo único que tenemos que hacer es obtener rectas paralelas a las del hiperplano que pasen por los vectores soporte. Para ello, lo único que tenemos que modificar es el valor de b que hemos obtenido con tal de obtener cada margen (el coeficiente libre indica el desplazamiento en el eje X que hace la recta para cortar con este eje en $x = 0$). En los dos casos es muy fácil obtener estos valores de b, ya que algunos de los vectores soporte están sobre el eje X. Por tanto, tenemos que para la Clase-1, la recta que representa el margen es la siguiente:

$$y = -x + 2 \tag{7}$$

Para la Clase-2, la recta es la siguiente:

$$y = -x + 1 \tag{8}$$

Por tanto, veamos como quedaría gráficamente el resultado de pintar el hiperplano óptimo y los márgenes:

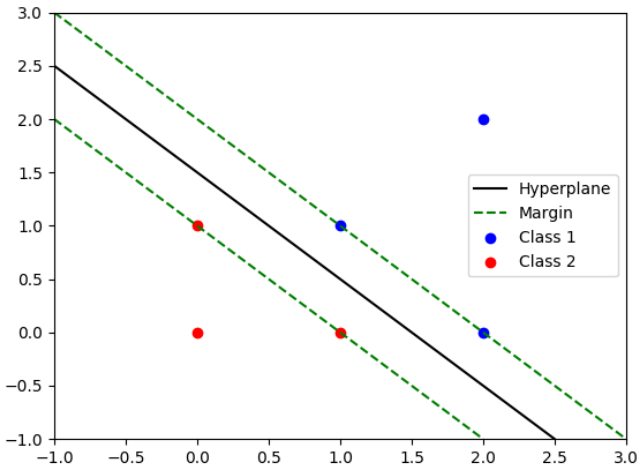

Figura 3: Dibujo del hiperplano óptimo con las dos clases y los márgenes para cada clase.

b) ¿Cuáles son los vectores soporte?

Solución

Los vectores soporte son los siguientes:

- Para la **Clase-1**, los vectores soporte son $(1, 1)$ y $(2, 0)$.
- Para la **Clase-2**, los vectores soporte son $(0, 1)$ y $(1, 0)$.

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

Solución

Ejercicio 4

¿Cúal es el criterio de optimalidad en la construcción de un árbol? Analice un clasificador en árbol en términos de sesgo y varianza. ¿Que estrategia de mejora propondría?

Solución

En un principio, el árbol óptimo és el árbol más pequeño capaz de clasificar correctamente todas las instancias. Sin embargo, intentar obtener este árbol representa un problema NP-completo, con lo cuál computacionalmente es inviable intentar obtenerlo y hay que intentar utilizar otras técnicas para intentar conseguir árboles subóptimos. Para intentar aprender estos árboles, podemos utilizar técnicas basadas en heurísticas greedy, las cuáles han demostrado ofrecer unos buenos resultados en general. La construcción de árboles de esta forma es muy simple: para cada variable, se obtiene la forma en la que esa variable particiona los datos en clases y se escoge la mejor variable según un determinado criterio. Esa variable forma un nuevo nodo e indica que para decidir sobre la clase a la que pertenece una instancia, se va a preguntar sobre su valor. Esto se hace de forma recursiva hasta que no queden más instancias que clasificar.

Existen distintas métricas que se puden utilizar para ver cómo de bien una variable divide un conjunto de datos. Una de ellas son por ejemplo la **entropía**, que mide el desorden o falta de información en un nodo, lo cuál significa que cuanto más equilibradas estén las clases en un conjunto de datos (que las proporciones de datos que pertenecen a una u otra clase sean más o menos las mismas) significa que hay una entropía alta, y por tanto, no se tiene mucha información, ya las dos clases son casi equiprobables o están muy cerca de serlo. Existen otras medidas también, como el **índice de Gini** y el **error de clasificación**, las cuáles se utilizan en algunos algoritmos de aprendizaje. Sin embargo, la que nos interesa es la entropía, ya que uno de los algoritmos más importantes, el **ID3** utiliza esta métrica. Más concretamente, utiliza la **ganancia de información**, que es el concepto opuesto a la entropía.

La idea básica del **ID3** es escoger aquellas variables que maximicen la ganancia de información. Es decir, queremos que la variable separe los datos restantes en conjuntos que tengan una entropía lo más pequeña posible, y que por tanto, la mayoría o todos los elementos de ese conjunto sean de la misma clase (con lo cuál habrá más orden). Así que, para eso, se van escogiendo aquellos atributos que ofrezcan una mayor ganancia de información y se va construyendo el árbol, de forma que si en alguno de los conjuntos solo quedan elementos de una clase se crea un nodo hoja con la etiqueta de esa clase. Esto se hace de forma recursiva hasta que no queden ejemplos por clasificar; es decir, se repite el proceso de escoger una variable con la mayor ganancia de información y ver como se distribuyen posteriormente los datos. Si se da que quedan ejemplos por clasificar pero no quedan atributos, se escoge la clase mayoritaria en ese nodo.

Analizando un árbol con los valores de sesgo y varianza, podemos ver que los árboles van a presentar en general un **sesgo bajo**, ya que no están asumiendo nada o casi nada sobre la función objetivo, con lo cuál son capaces de aprender casi

cualquier función que se desee a diferencia de otros modelos como por ejemplo los lineales, los cuáles sí que imponen restricciones sobre la función objetivo (que esta sea lineal). Sin embargo, los árboles suelen presentar una **alta varianza** debido a que entrenarlos con una muestra de datos o con otra nos produciría modelos muy diferentes (los nodos y las hojas podrían ser completamente diferentes).

Debido a que existe una alta varianza en este modelo, podemos intentar disminuirlo a costa de aumentar un poco el sesgo, lo cuál se conoce como el *bias-variance trade-off*. Para realizarlo, lo que se suele hacer es podar el árbol, eliminando aquellos nodos que no son muy informativos. Con esto, además, se consigue hacer que el tamaño del árbol se vea reducido efectivamente. Esta poda puede realizarse mientras se construye el árbol o una vez construido, comprobando el error que se obtiene de validación cruzada. En el primer caso, se comprueba como va disminuyendo el error a medida que se crea el árbol, y si se da el caso de que en un momento disminuye muy poco, se deja de construir (lo cuál es conocido como **early stopping**). En el otro caso, a partir del árbol construido, se quita cada vez el nodo del árbol que más mejore la precisión con el conjunto de validación hasta que la precisión que se obtiene sobre este conjunto empiece a empeorar. Esta segunda técnica también es conocida como **post-prunnning**.

Ejercicio 5

¿Cómo influye la dimensión del vector de entrada en los modelos: SVM, RF, Boosting y NN?

Solución

- En **SVM** la dimensión no influye en una gran medida. En caso de clasificación lineal, y si los datos son linealmente separables, se puede conseguir un hiperplano óptimo que deje suficiente margen a los lados, con lo cuál la dimensión VC de ese modelo será menor a la dimensión del vector de entrada. Si en cambio no se pueden separa linealmente los datos, se pueden utilizar *kernels*, los cuáes transformarán el espacio de entrada de dimensión d en un espacio con una dimensión mucho mayor. Por tanto, en estos casos al trabar con un número de dimensiones mucho mayor que el de entrada, no influye mucho cuántas dimensiones tenía el vector de entrada, ya que ahora habrán muchas más independientemente de que fuesen muchas o pocas.
- En **RF** la dimensión no influye mucho ya que se utilizan un subconjunto de las características para aprender un árbol en vez de intentar utilizarlas todas, con lo cuál, por muy grande que sea el vector de entrada, solo se usará una parte de él.

- **Boosting** no se ve influido por la dimensión del vector de entrada. Esto se debe a que el enfoque de este modelo es construir un modelo complejo a partir de modelos más simples los cuáles son *weak classifiers* (clasificadores que funcionan un poco mejor que uno aleatorio). Como este tipo de clasificadores normalmente son *stumps* (árboles de un solo nodo), solo se mira una característica. Con lo cuál, tener muchas o pocas no influye mucho.
- En **NN** la dimensión del vector de entrada influye en el modelo, ya que va a influir tanto en la dimensión VC como en el error de generalización. También influye a la hora de escoger el número de *hidden layers*, ya que hay que intentar escogerlo teniendo en cuenta el número de datos que tengamos y el número de dimensiones que tengan éstos.

Ejercicio 6

El método de Boosting representa una forma alternativa en la búsqueda del mejor clasificador respecto del enfoque tradicional implementado por los algoritmos PLA, SVM, NN, etc. a) Identifique de forma clara y concisa las novedades del enfoque; b) Diga las razones profundas por las que la técnica funciona produciendo buenos ajustes (no ponga el algoritmo); c) Identifique sus principales debilidades; d) ¿Cuál es su capacidad de generalización comparado con SVM?

Solución

a) Este nuevo enfoque pretende conseguir crear un modelo robusto a partir de modelos simples o *weak classifiers*. Es decir, pretende obtener un modelo más complejo a partir de modelos muy simples que clasifican un poco mejor que un modelo que clasifica de forma aleatoria. Se realiza mediante un proceso iterativo en el que se ajusta un clasificador simple a una muestra de datos ponderada (los datos tienen un peso según su historia). Si el nuevo modelo clasifica bien los datos, se disminuye el peso asociado a estos. Si clasifica incorrectamente una serie de datos, se les añade más peso para que posteriormente se pueda corregir ese error.

b) Esta técnica produce unos buenos ajustes ya que en cada iteración se intenta ajustar un clasificador que sea capaz de clasificar bien los datos en los que anterioremnte fallaba. Al componer luego múltiples clasificadores simples, se tiene un modelo que es capaz de explicar muy bien todos los datos, ya que cada uno de los clasificadores simples explica una parte de los datos.

c) Una de las principlaes debilidades es que si se utiliza un *weak classifier* demasiado complejo se puede producir *overfitting*, ya que cada uno de los clasificadores intentará explicar los datos lo mejor posible, o si utiliza un *weak*

classifier demasiado simple se puede llegar a que el modelo se queda muy corto (*underfitting*) o a que se produzca *overfitting* debido a que no se deje suficiente margen. Otro problema que tiene es que es una técnica muy susceptible al ruido uniforme, ya que es muy probable que le asigne muchas veces pesos altos a datos con ruido y que por tanto se termine adaptando a éstos, clasificando posteriormente peor.

d) Boosting tiene una muy buena capacidad de generalización comparándolo con otros modelos como SVM, siendo aproximadamente igual de buena que la de SVM. Es una técnica que, partiendo de un *weak classifier* adecuado, permite también obtener un muy buen margen al igual que hace SVM con datos linealmente separables. Por tanto, Boosting va a permitir generalizar muy bien, ya que con un número suficiente de iteraciones, se puede obtener un buen valor de margen. También, es uno de los pocos modelos que llegado un punto en el que se da que $E_{in} = 0$, si se siguen realizando iteraciones se puede seguir reduciendo E_{out} más y más, hasta un cierto límite.

Ejercicio 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.

Solución

Ejercicio 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.

Solución

Ejercicio 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.

Solución

En un principio, parece que todo el planteamiento es correcto, y por tanto, en un principio, las conclusiones que se vayan a extraer al haber tomado la muestra parecen ser correctas. Sin embargo, al volver a pensar en todo el proceso, nos damos cuenta de que hay ciertos aspectos que puede que no hayan sido considerados, ya que no se han mencionado en el planteamiento del problema.

El primero de ellos es que en ningún momento se ha hablado del tamaño de la red utilizada. Si la red es muy grande lo que va a pasar es que vamos a poder capturar los peces más grandes del lago sin ningún tipo de problema. Sin embargo, al ser tan grande, los peces pequeños van a pasar de largo y no van a poder se capturados. Por tanto, nuestra muestra solo contendrá peces grandes de cierto tamaño que no sean capaces de pasar por nuestra red. Puede que haya muchos peces grandes, pero muchos más peces que sean menores que éstos y que no hayamos podido capturar por nuestra red. Si se da esto, podemos decir que la muestra que hemos extraído no representa lo suficientemente bien a la población, y que al intentar algún tipo de conclusión estadística sobre ésta como por ejemplo la distribución de los peces por tamaño, nos estaremos equivocando.

Otro aspecto que a lo mejor no ha sido considerado a la hora de realizar el experimento es la zona del lago en la que se ha realizado. Si el experimento se ha realizado a lo largo de todo el lago, entonces no hay ningún problema, ya que se extraen peces de zonas diferentes del lago, las cuáles pueden tener sus propias peculiaridades y hacer que los peces que haya en esa zona sean diferentes. En cambio, si los peces solo se han extraído de una zona del lago, estamos tomando una muestra de solo una pequeña parte de la población, con lo cuál, las conclusiones que extraigamos a partir de esta muestra pueden ser ciertas solo para **una parte de la población**, y difícilmente se podría concluir algo sobre toda la población en general a partir de estas observaciones.

Ejercicio 10

Identifique que pasos daría y en que orden para conseguir con el menor esfuerzo posible un buen modelo de red neuronal a partir una muestra de datos. Justifique los pasos propuestos, el orden de los mismos y argumente que son adecuados para conseguir un buen óptimo. Considere que tiene suficientes datos tanto para el ajuste

como para el test.

Solución

Referencias

[1] Wikipedia. *Decision Tree Learning*
https://en.wikipedia.org/wiki/Decision_tree_learning

[2] Jason Brownlee. *Gentle Introduction to the Bias-Variance Trade-Off in Machine Learning*
https://machinelearningmastery.com/gentle-introduction-to-the-bias-variance-trade-off-in-machine-learning/

[3] DISPLAYR. *Machine Learning: Pruning Decision Trees*
https://www.displayr.com/machine-learning-pruning-decision-trees/

[4] The University of Utah. *Support Vector Machines*
http://svivek.com/teaching/machine-learning/fall2018/slides/svm/svm-intro.pdf

[5] Wikipedia. *Support Vector Machine*
https://en.wikipedia.org/wiki/Support-vector_machine

[6] Wikipedia. *Boosting*
https://en.wikipedia.org/wiki/Boosting_(machine_learning)