Ayudantía I3

---

1. Verdadero o Falso

a) En Machine Learning modelar los datos es más importante que buscar el aprendizaje. **Falso, los algoritmos buscan aprendizaje más que modelar los datos**.

b) Cuando hablamos de *clustering*, clasificación y reducción de dimensionalidad, estamos hablando de aprendizaje supervisado. **Falso, tanto *clustering* como reducción de dimensionalidad son categorías de algoritmos de aprendizaje no supervisado**.

c) El *set* de entrenamiento nos permite calibrar nuestro modelo. **Verdadero**.

d) Una fila en el set de entrenamiento, puede considerarse como una *feature*. **Falso, una fila es un vector en el espacio de características**.

f) No es posible visualizar el espacio de características de MNIST debido a que son 10 dígitos y solo podemos visualizar 2 o 3 dimensiones. **Falso, no es posible visualizar MNIST por la gran cantidad de dimensiones, pero estas son las *features***.

g) Tener un buen *set* de entrenamiento garantiza tener una buena generalización. **Falso, siempre tenemos que apuntar a generalizar para evitar *overfitting***.

h) *Overfitting* ocurre cuando el modelo comienza a memorizar los datos de entrenamiento. **Verdadero**.

i) Cada valor en la diagonal de una matriz de confusión indica la cantidad de ejemplos mal clasificados para cada clase. **Falso, indica la cantidad de ejemplos correctamente clasificados**.

---

2. Árboles de Decisión

Se desea construir un árbol de decisión para saber si Chile sería capaz de ganar un partido de fútbol. La decisión estará basada en dos atributos binarios:

- A = Llueve.
- B = El estadio está lleno.

Asuma que, históricamente, Chile gana el 70% de los partidos que juega como local (el restante lo empata o pierde). Además, Chile juega 20% de los partidos de local con lluvia y de ellos gana el 80%, mientras que cuando no llueve gana el 85%. Por otro lado, Chile juega de local 60% de sus partidos con estadio lleno y de ellos gana el 85%, mientras que cuando no hay estadio lleno gana sólo el 45% de las veces.

a) Arme el árbol de decisión. Dibuje el árbol completo, especificando los valores para cada nodo.

> *Este ejercicio recomiendo que lo hagan al menos una vez para que entiendan los pasos para armar un árbol. Recuerden cómo calcular la entropía y la ganancia de información, para qué sirve cada fórmula y qué hacemos para evitar overfitting.*

- Entropy:

$$H(S) = -\sum_{c_i} p_i \cdot log_2(p_i)$$

- Gain:

$$G(S, A) \equiv H(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

b) En el entrenamiento de un árbol de decisión, siempre existe un atributo (dimension, *feature*, ...) que está presente en todas las reglas de clasificación representadas por el árbol. ¿Cuál atributo es el que aparece en todas las reglas?¿Por qué esto podría ser una fuente de inestabilidad?

R: El atributo ubicado en la raíz del árbol aparece en todas las reglas. Ésta es una fuente de inestabilidad en el uso de árboles de decisión dada su sensibilidad al correcto valor de este atributo.

c) Supongamos que su modelo no esté obteniendo buenos resultados. ¿Es cierto que aumentar el tamaño del *set* de entrenamiento **siempre** resultará en un mejor modelo?

R: No necesariamente. En primer lugar, se requiere garantizar que los nuevos datos sean representativos del problema y no una muestra sesgada. Aún si este es el caso, puede ser que la herramienta de aprendizaje no sea la decuada y nuevos datos no mejoren el rendimiento del modelo.

---

3. Redes Neuronales

La siguiente es la función de error para problemas de clasificación binaria, en la que se utiliza una función de activiación con output logístico-sigmoide, de forma que $0 \leq y(\mathbf{x}, \mathbf{w}) \leq 1$ y los valores objetivos (*target*) son $t \in \{0, 1\}$:

$$E(\mathbf{w}) = -\sum_{n=1}^{N} \{t_n \cdot \ln(y_n) + (1 - t_n) \cdot \ln(1 - y_n)\}$$

Obtener la función de error correspondiente si consideramos una red con un *output* $-1 \leq y(\mathbf{x}, \mathbf{y}) \leq 1$ y con valores objetivo $t = 1$ para la clase $\mathcal{C}_1$ y $t = -1$ para la clase $\mathcal{C}_2$. Cuál sería una función de activación apropiada?

R: Esto corresponde a simplemente escalar y trasladar los *outputs*, lo que entrega directamente la nueva función de activación:

$$y = 2 \cdot \sigma(a) - 1$$

La función correspondiente se obtiene aplicando la transformación inversa a y_n y t_n, obteniendo

$$E(\mathbf{w}) = -\sum_{n}^{N} \frac{1+t_n}{2} \cdot \ln \frac{1+y_n}{2} + (1 - \frac{1+t_n}{2}) \cdot \ln(1 - \frac{1+y_n}{2})$$

$$E(\mathbf{w}) = -\frac{1}{2} \sum_{n}^{N} (1+t_n) \cdot \ln(1+y_n) + (1-t_n) \cdot \ln(1-y_n) + N \ln 2$$

El último término puede ser ignorado, ya que es independiente de $\mathbf{w}$. Para encontrar la función de activación, simplemente se aplica la transformación a la función logística, lo que entrega:

$$y(a) = 2 \cdot \sigma(a) - 1 = \frac{2}{1+e^{-a}} - 1$$

$$y(a) = \frac{1-e^{-a}}{1+e^{-a}} = \frac{e^{a/2} - e^{-a/2}}{e^{a/2} + e^{-a/2}}$$

$$y(a) = \tanh(a/2)$$

Material Extra

- Como les comenté en la ayudantía, en esta aplicación pueden jugar un poco con las redes neuronales, ver qué es lo que pasa dentro de cada una mientras se entrenan y visualizar el *output* de su modelo. A Neural Network Playground - TensorFlow.
- También me comprometí con un *post* en un blog que encontré donde se explicaba *backpropagation*. Por un tema de tiempo, les recomiendo no leerlo completo (haganlo cuando puedan). Denle una mirada a las secciones que hablan de esto, y al respectivo ejemplo. Hacker's guide to Neural Networks - Andrej Karpathy blog.
- Además, un video que vi hace unos días que podría ayudarles a entender *backpropagation* de una forma intuitiva y correcta. What is backpropagation and what

is it actually doing? (YouTube video).