Los datos de nuestro dataset son finitos y debemos distribuirlos para que el entrenamiento se haga con la máxima cantidad de ejemplos posibles a la vez que podamos verificar la veracidad del modelo con datos reales no vistos anteriormente, para esto creamos los subsets de entrenamiento, validación y pruebas.

**¿Por qué distribuir datos?**

Para entender esta necesidad, pasemos a una analogía con helados: de niño estás aprendiendo sobre los diferentes tipos de helados, tu padre tiene 100 helados, de los cuales usará 70 para enseñarte y 30 para ponerte a prueba; cada día te mostrará un helado diferente y te dirá su sabor hasta que se terminen y luego te preguntará por aquellos que no has visto.

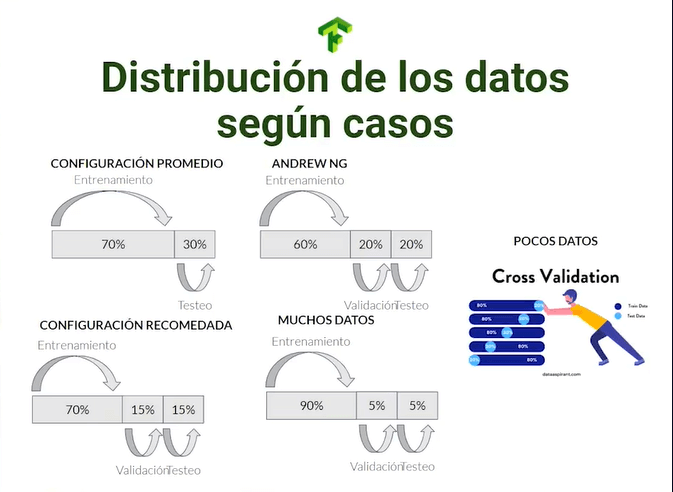
Lo anterior hace alusión a los datos de entrenamiento y prueba, donde los primeros se usarán para entrenar el modelo (tendrán acceso a las etiquetas de salida) mientras que los segundos serán para predecir, el problema con esto es que solo estaremos comprendiendo la eficacia del modelo una vez finalizado el entrenamiento.

Para solucionar este problema y tener feedback en vivo del desempeño del modelo creamos el subset de validación, que hará el papel de pruebas durante cada época del entrenamiento, permitiendo monitorear el rendimiento de la red a través de las iteraciones.

**Determinando los porcentajes de cada subset**

La distribución de los datos a los diferentes subsets se puede determinar de diferentes maneras, donde la configuración promedio será de 70% para entrenamiento y 30% para pruebas (la mitad de este conjunto podrían destinarse a validación). Andrew NG (de las figuras más importantes del Deep Learning moderno) propone una estructura de 60% de entrenamiento, 20% de validación y 20% pruebas.

En caso de poseer pocos datos es recomendable aplicar la técnica de *cross validation*, que nos permitirá iterar el subset de validación entre los datos de entrenamiento, mientras que si tienes muchos datos puedes maximizar la cantidad de datos a entrenamiento en una estructura 90%/5%/5%.



**Errores comunes al distribuir datos**

Cuando distribuyas datos es posible encontrarte con errores altamente mortales en tiempo de ejecución porque no son de lógica ni compilación sino de estructuración, no serán detectados por la máquina y pueden ser muy costosos de detectar y solucionar.

Un error común es el de combinar erróneamente los datos de entrenamiento con los de testeo, lo que resultará en un rendimiento artificialmente alto para la red. Otro error común es el de clases desbalanceadas, es decir, la cantidad de ejemplos de diferentes clases es diferente.

Supongamos 95 ejemplos de la clase A con 5 ejemplos de la clase B, incluso si todos los ejemplos los clasificamos como A, tendremos una precisión artificial de 95%. Si tienes muy pocos datos, el modelo no podrá entrenarse dado que no tendrá ejemplos suficientes para abstraer los patrones a enseñar.



Con esto tienes las intuiciones necesarias para distribuir tus datasets, los valores exactos los podrás decidir basándote en las recomendaciones e intuiciones personales, pero ya puedes partir con total seguridad desde los hombros de los gigantes del machine learning.

**Contribución creada por** Sebastián Franco Gómez.