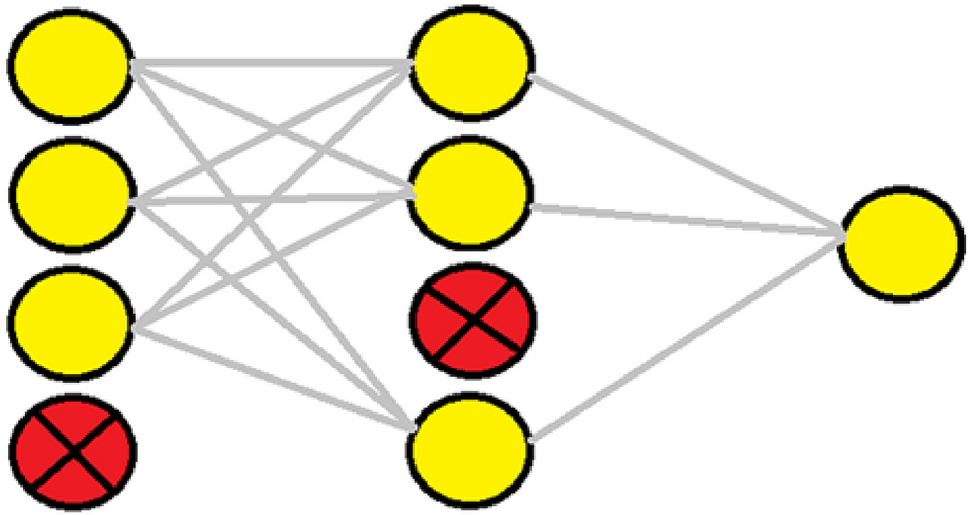
Parada Anticipada

La técnica de abandono (Dropout), es una estrategia de regularización utilizada en redes neuronales para prevenir el sobreajuste al reducir la dependencia entre neuronas. Esta técnica implica "abandonar" aleatoriamente un conjunto de neuronas durante el entrenamiento, lo que significa que se excluyen temporalmente de la propagación hacia adelante y hacia atrás.

Durante cada iteración del entrenamiento, cada neurona tiene una probabilidad asociada de ser "abandonada", esta probabilidad se establece antes del entrenamiento y se conoce como la tasa de abandono; la tasa de abandono típicamente varía entre 0.2 y 0.5, lo que significa que cada neurona tiene una probabilidad del 20% al 50% de ser excluida en cada iteración.



La idea detrás del abandono es que, al forzar a la red a aprender con neuronas ausentes en diferentes momentos, se evita que las neuronas se vuelvan demasiado dependientes entre sí. Esto mejora la generalización del modelo, ya que las neuronas deben aprender a ser útiles incluso cuando otras están "apagadas". Además, el abandono actúa como una especie de ensamble de modelos, ya que cada iteración de entrenamiento utiliza un conjunto diferente de neuronas.

El abandono se aplica típicamente en capas ocultas durante la fase de entrenamiento, y durante la inferencia (o evaluación), todas las neuronas se utilizan. Esto se hace para asegurar que el modelo se beneficie completamente de todas sus capacidades durante la predicción.

Ejemplo de aplicación efectiva del dropout en redes neuronales

Imaginemos que estamos construyendo una red neuronal para la clasificación de imágenes. Supongamos que la arquitectura de nuestra red incluye una capa oculta con Dropout. Aquí hay un ejemplo simplificado utilizando la biblioteca TensorFlow en Python:

```python
import tensorflow as tf
from tensorflow.keras import layers, models

# Definición del modelo
model = models.Sequential()

# Capa de entrada
model.add(layers.Flatten(input_shape=(28, 28)))  # Supongamos imágenes de 28×28 píxeles

# Capa oculta con Dropout
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dropout(0.5))  # Tasa de abandono del 50%

# Capa de salida
model.add(layers.Dense(10, activation='softmax'))  # Supongamos 10 clases para la clasificación

# Compilación del modelo
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Entrenamiento del modelo (supongamos que tenemos datos de entrenamiento x_train, y_train)
model.fit(x_train, y_train, epochs=10, batch_size=32, validation_split=0.2)
```

En el anterior ejemplo, se ha agregado una capa oculta con Dropout al modelo, la capa Dense con activación 'relu' tiene 128 neuronas, y después de eso, hemos agregado una capa Dropout con una tasa del 50%. Esto significa que durante el entrenamiento, cada neurona de esta capa tiene un 50% de probabilidad de ser "abandonada" en cada iteración.

Este Dropout ayuda a prevenir el sobreajuste. Durante la inferencia, cuando no están entrenando la red, TensorFlow automáticamente desactiva Dropout, y todas las neuronas contribuyen a las predicciones finales.

Este es un ejemplo simplificado, y en problemas más complejos, ajustar la tasa de Dropout y su posición en la red puede requerir experimentación para encontrar la configuración óptima.