**¿Cuáles son sus causas?**

Las **causas del overfitting** pueden ser complicadas. Generalmente, podemos categorizarlas en tres  
tipos:

* Aprendizaje de ruido en el conjunto de entrenamiento: cuando el conjunto de entrenamiento es demasiado pequeño o tiene menos datos representativos o demasiados ruidos. Esta situación hace que los ruidos tengan grandes posibilidades de ser aprendidos, y luego actuar como base de predicciones. Por lo tanto, un algoritmo que funcione bien debería poder distinguir los datos representativos de los ruidos.
* Complejidad de la hipótesis: la compensación en complejidad, un concepto clave en el aprendizaje de la estadística y el mecanizado, es un compromiso entre Varianza y Sesgo. Eso se refiere a un equilibrio entre precisión y coherencia. Cuando los algoritmos tienen demasiadas hipótesis (demasiadas entradas), el modelo se vuelve más preciso en promedio con menor consistencia. Esta situación significa que los modelos pueden ser drásticamente diferentes en diferentes conjuntos de datos.
* Múltiples procedimientos de comparación que son omnipresentes en los algoritmos de inducción, así como en otros Algoritmos de inteligencia artificial. Durante estos procesos, siempre comparamos varios elementos en función de puntuaciones de una función de evaluación y seleccionamos el elemento con la puntuación máxima. Sin embargo, este proceso probablemente elegirá algunos elementos que no mejorarán o incluso reducirán la precisión de la clasificación.

El overfitting puede tener muchas causas y normalmente es una combinación de las siguientes:

* Modelo demasiado potente: por ejemplo, permite polinomios hasta el grado 100. Con polinomios hasta el grado 5, tendría un modelo mucho menos potente que es mucho menos propenso a sobreajustarse.
* No hay suficientes datos: obtener más datos a veces puede solucionar problemas de sobreajuste.

**¿Qué consecuencias negativas puede tener?**

El **sobreajuste u overfitting**, en pocas palabras, significa tener en cuenta **demasiada información** de sus datos y / o conocimientos previos, y utilizarla en un modelo. Para hacerlo más sencillo, considera el siguiente ejemplo: algunos científicos te contratan para proporcionarles un modelo para predecir el crecimiento de algún tipo de plantas. Los científicos te han brindado información recopilada de su trabajo con dichas plantas durante todo un año, y continuamente te brindarán información sobre el desarrollo futuro de su plantación.

Entonces, revisa los datos recibidos y crea un modelo a partir de ellos. Ahora supón que, en tu modelo, has considerado tantas características como sea posible para averiguar siempre el comportamiento exacto de las plantas que viste en el conjunto de datos inicial. Ahora, a medida que continúa la producción, siempre tendrás en cuenta esas características y producirás resultados muy precisos. Sin embargo, si la plantación eventualmente sufre algún cambio estacional, los resultados que recibirás pueden ajustarse a tu modelo de tal manera que tus predicciones comenzarán a fallar.

Además de ser incapaz de detectar variaciones tan pequeñas y de clasificar sus entradas de manera incorrecta, el detalle del modelo, es decir, la gran cantidad de variables, puede hacer que el procesamiento sea demasiado costoso. Ahora, imagina que tus datos ya son complejos. Sobreajustar tu modelo a los datos no solo hará que la clasificación / evaluación sea muy compleja, sino que probablemente hará que te equivoques en la predicción por la más mínima variación que pueda tener en la entrada.

El sobreajuste es empíricamente malo. Supón que tienes un conjunto de datos que divides en dos, prueba y entrenamiento. Un modelo sobreajustado es aquel que tiene un rendimiento mucho peor en el conjunto de datos de prueba que en el conjunto de datos de entrenamiento. A menudo se observa que modelos como ese también, en general, funcionan peor en conjuntos de datos de prueba adicionales que los modelos que no están sobreajustados.

**¿Cómo detectar el overfitting en un modelo predictivo?**

**Detectar el overfitting** es casi imposible antes de probar los datos. Puede ayudar a abordar la característica inherente del sobreajuste, que es la incapacidad de generalizar conjuntos de datos. Por lo tanto, los datos se pueden separar en diferentes subconjuntos para facilitar el entrenamiento y las pruebas. Los datos se dividen en dos partes principales, es decir, un conjunto de prueba y un conjunto de entrenamiento.

El **conjunto de entrenamiento** representa la mayoría de los datos disponibles (alrededor del 80%) y entrena el modelo. El conjunto de prueba representa una pequeña parte del conjunto de datos (aproximadamente el 20%) y se utiliza para probar la precisión de los datos con los que nunca antes interactuó. Al segmentar el conjunto de datos, podemos examinar el rendimiento del modelo en cada conjunto de datos para detectar el sobreajuste cuando ocurre, así como ver cómo funciona el proceso de entrenamiento.

El rendimiento se puede medir utilizando el porcentaje de precisión observado en ambos conjuntos de datos para concluir sobre la presencia de sobreajuste. Si el modelo funciona mejor en el conjunto de entrenamiento que en el conjunto de prueba, significa que es probable que el modelo esté sobreajustado.

**¿Cómo evitar el overfitting o sobreajuste?**

A continuación, se muestran algunas de las formas de evitar el overfitting:

**Entrenamiento con más datos**

Es posible que esta técnica no funcione todas las veces. Básicamente, ayuda al modelo a identificar mejor la señal.

Pero en algunos casos, el aumento de los datos también puede significar alimentar más ruido al modelo. Cuando entrenamos el modelo con más datos, debemos asegurarnos de que los datos estén limpios y libres de aleatoriedad e inconsistencias.

**Detención anticipada**

Cuando el modelo se está entrenando, puedes medir el rendimiento del modelo en función de cada iteración. Podemos hacer esto hasta un punto en el que las iteraciones mejoren el rendimiento del modelo. Después de esto, el modelo sobreajusta los datos de entrenamiento a medida que la generalización se debilita después de cada iteración.

Entonces, básicamente, la detención anticipada significa detener el proceso de entrenamiento antes de que el modelo pase el punto donde el modelo comienza a sobreajustarse a los datos de entrenamiento. Esta técnica se utiliza principalmente en el aprendizaje profundo.

**Eliminación de funciones**

Aunque algunos algoritmos tienen una selección automática de funciones. Para un número significativo de aquellos que no tienen una selección de características incorporada, podemos eliminar manualmente algunas características irrelevantes de las características de entrada para mejorar la generalización.

Una forma de hacerlo es derivar una conclusión sobre cómo encaja una característica en el modelo. Es bastante similar a depurar el código línea por línea.

En caso de que una característica no pueda explicar la relevancia en el modelo, simplemente podemos identificar esas características. Incluso podemos usar algunas heurísticas de selección de características para un buen punto de partida.

**Ensamblaje**

Esta técnica básicamente combina predicciones de diferentes modelos de Machine Learning. A continuación se enumeran dos de los métodos más comunes para ensamblar:

* El embolsado intenta reducir la posibilidad de sobreajuste de los modelos
* Impulsar los intentos de mejorar la flexibilidad predictiva de modelos más simples

Aunque ambos son métodos de conjunto, el enfoque comienza totalmente en direcciones opuestas. Bagging usa modelos base complejos e intenta suavizar sus predicciones mientras que el impulso usa modelos base simples e intenta aumentar su complejidad agregada.

**Parada anticipada**

Cuando entrenas un algoritmo de aprendizaje de forma iterativa, puedes medir qué tan bien se desempeña cada iteración del modelo.

Hasta cierto número de iteraciones, las nuevas iteraciones mejoran el modelo. Sin embargo, después de ese punto, la capacidad del modelo para generalizar puede debilitarse a medida que comienza a sobreajustarse a los datos de entrenamiento.

La detención anticipada se refiere a detener el proceso de capacitación antes de que el alumno pase ese punto.

Hoy en día, esta técnica se utiliza principalmente en el aprendizaje profundo, mientras que otras técnicas se prefieren para el aprendizaje automático clásico.

**Validación cruzada**

La validación cruzada es una poderosa medida preventiva contra el sobreajuste.

La idea es inteligente: usa tus datos de entrenamiento inicial para generar múltiples divisiones de prueba de mini tren. Utiliza estas divisiones para ajustar tu modelo.

En la validación cruzada estándar de k veces, dividimos los datos en k subconjuntos, llamados pliegues. Luego, entrenamos iterativamente el algoritmo en k-1 pliegues mientras usamos el pliegue restante como conjunto de prueba (llamado “pliegue reservado”).

La validación cruzada te permite ajustar los hiperparámetros solo con su conjunto de entrenamiento original. Esto permite mantener tu conjunto de prueba como un conjunto de datos realmente invisible para seleccionar tu modelo final.