Machine Learning

75.06 Organización de Datos

Aprendizaje Automático

Supervisado (tenemos labels)

- Regresión (labels numéricos)
- Clasificación (labels categóricos)

No-Supervisado

- Detección de anomalías
- Clustering

Aprendizaje Supervisado

El Set de Datos

- El set de datos tiene “m” filas (rows u observaciones) y “n” columnas (features o variables).
- Al proceso de creación de features lo llamamos “feature engineering”.

El Set de Datos

- A partir del set de datos queremos “entrenar” un modelo de Machine Learning que nos permita predecir el label a partir de los features.

Validación del Modelo

¿Cómo podemos evaluar el modelo entrenado?

Idea: Hacer predicciones con datos para los cuáles conocemos el valor a predecir (label)

Validación del Modelo

- Dividimos el set de datos en dos:
  - Training set
  - Test set
- El training set lo usamos para entrenar, el test set lo usamos para medir la performance de nuestro modelo.
- Esta medición implica el uso de alguna métrica de performance.

Métricas

- Accuracy
- Precisión
- Recall
- RMSE
- MAE
- Auc
- Otras...

Split Train-Test

- En algunos casos podemos hacerlo al azar (ejemplo 80% train, 20% test)
- En otros casos es importante hacerlo por tiempo (evitar time-travelling)
- Para las métricas es importante que el set de test tenga distribución productiva.

Sets Desbalanceados

- Si el set de datos está muy desbalanceado puede ser difícil entrenar el modelo.
- Posibles soluciones:
  - Oversamplear la clase minoritaria.
  - Subsamplear la clase mayoritaria.
  - Manejar el desbalanceo con hiper-parámetros del modelo.

Resumen hasta aquí

- Tenemos un set de datos y necesitamos:
  - Un set de train (que dependiendo del modelo podemos necesitarlo balanceado)
  - Un set de test con distribución productiva.
  - Evitar problemas de time-traveling (jamás tener en train algo que sucedió luego de algo en test)
  - Elegir la métrica con la cual vamos a evaluar nuestro modelo. (hay muchas opciones)
  - Es importante que el set de test nunca sea usado para entrenar

Evaluando el modelo

- Error de entrenamiento
  - Entrenamos con el train-set y luego hacemos predicciones para ese mismo train-set evaluando la métrica que elegimos.
- Error de test
  - Entrenamos con el train-set y luego hacemos predicciones para el test-set evaluando la métrica que elegimos.

¿Qué es entrenar?

- El entrenamiento de un modelo es un problema de optimización.
- Buscamos los parámetros óptimos para minimizar o maximizar la métrica elegida.
- Dependiendo del modelo varía la técnica de optimización a usar.

Modelo ya entrenado

- Un modelo ya entrenado queda representado por sus parámetros.
- Por ejemplo si entrenamos un polinomio de grado 2 el resultado son los coeficientes del polinomio.

Hiper-Parámetros

- Los hiper-parámetros son valores que parametrizan el entrenamiento del modelo.
- Son valores que debemos indicar por afuera, no son parte del proceso de optimización del modelo.

Hiper-Parámetros: Ejemplos

- Polinomios: grado del polinomio.
- Árboles de decisión: Profundidad máxima, cantidad mínima de nodos por hoja, etc.
- Redes Neuronales: Cantidad de capas, cantidad de neuronas en cada capa, función de activación a usar, dropout, método de optimización a usar, etc.

Hiper-Parámetros: Ejemplos

- KNN: Cantidad de vecinos a evaluar, distancia a emplear.
- Random Forests: Cantidad de árboles a usar, cantidad de atributos a usar, etc.
- Gradient Boosting: Numero de estimadores, gamma, factor de aprendizaje, profundidad máxima, subsample de rows y columnas, etc.

Búsqueda de Hiper-Parámetros

- Necesitamos encontrar el conjunto óptimo de hiper-parámetros para luego encontrar el modelo (parámetros) óptimo.
- A la búsqueda de hiper-parámetros se la conoce también como “tuning”
- En algunos modelos el proceso de “tuning” es más crítico que en otros.

Búsqueda de Hiper-Parámetros

- Grid-Search
- Random-Search
- Optimización Bayesiana

Grid-Search

- Establecemos un conjunto de valores a probar por cada hiper-parámetro.
- Ejemplo: grado del polinomio {2,3,4} y factor de regularización {0,0.05,1}
- Probamos cada combinación de hiper-parámetros y nos quedamos con la mejor.
- En nuestro ejemplo son 3*3 = 9 casos.

Grid Search

- Una vez que encontramos los valores óptimos podemos refinar la búsqueda.
- Por ejemplo si el óptimo está en un extremo deberíamos probar valores aun más grandes o pequeños (según el caso)
- Si el óptimo está entre dos valores deberíamos probar con un poco más de resolución.

Grid-Search

- El principal problema del método de Grid-Search es qué consume mucho tiempo al tener que evaluar todas las combinaciones posibles de valores elegidos para cada hiper-parámetro.
- Algunos modelos tienen muchos hiper-parámetros.

Random Search

- Al igual que en grid-search definimos un conjunto de valores posibles por cada hiper-parámetro.
- Probamos “k” combinaciones aleatorias de hiper-parámetros y nos quedamos la mejor.
- Decidimos cuánto tiempo invertir pero no probamos todas las combinaciones.

Búsqueda Bayesiana

- Definimos una distribución de probabilidades y rangos por cada hiper-parámetro.
- Ejemplo grado del polinomio uniforme entre 1 y 5.
- El algoritmo se encarga de ir probando combinaciones de hiper-parámetros de forma bayesiana.

“The Bayesian Way”

- El algoritmo prueba combinaciones y en base a los resultados asigna pesos.
- En la siguiente iteración samplea los valores de hiper-parámetros a elegir en base a estos pesos.
- De esta forma a medida que aprende tiene que explorar menos valores sin sentido.

Probando Hiper-parámetros

- Tanto en grid-search como en random-search o en la optimización bayesiana tenemos que “probar” un conjunto de hiper-parámetros que elegimos.
- ¿Cómo?

Probando Hiper-parámetros

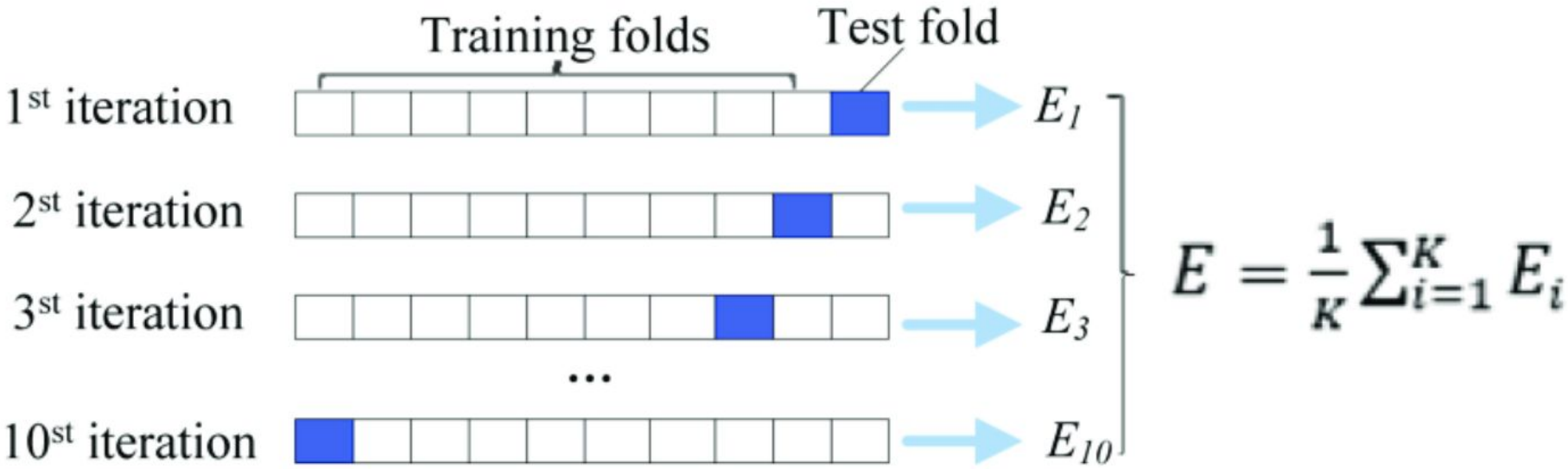
- No podemos usar el test-set (no debemos tocarlo)
- Tampoco tiene sentido entrenar con el train-set y probar con el mismo set ya que los hiper-parámetros serían óptimos para el set de entrenamiento pero no para datos por afuera del mismo.

Set de Validación

- Idea: Dividir el train-set en dos: train y validation.
- Es decir que tendremos un set de train, uno de test y otro de validación.
- Pero si el set de validación es fijo los hiper-parámetros solo serán óptimos para ese conjunto de registros.

Cross-Validation

- Idea: dividir el set de entrenamiento en “k” particiones del mismo tamaño (folds)
- Usar cada fold como set de validación entrenando con el resto.
- Promediar la métrica en las “k” validaciones.


Training folds
Test fold
1st iteration
2st iteration
3st iteration
...
10st iteration
E_1
E_2
E_3
E_{10}
$E = \frac{1}{K}\sum_{i=1}^{K} E_i$

K-fold cross validation

- Cuando k=1 usamos cada registro como set de validación. (LOOCV: Leave one out cross-validation). Pero suele ser muy costoso.
- En general valores como k=10 son populares.

Overfitting y Underfitting

Underfitting

- Se produce cuando tenemos un error de entrenamiento alto.
- El modelo ajusta mal al set de entrenamiento.
- El modelo tiene “visión borrosa”

Underfitting

- El modelo no tiene suficiente capacidad expresiva.
- El modelo no tiene suficiente complejidad o grados de libertad.

Underfitting: Soluciones

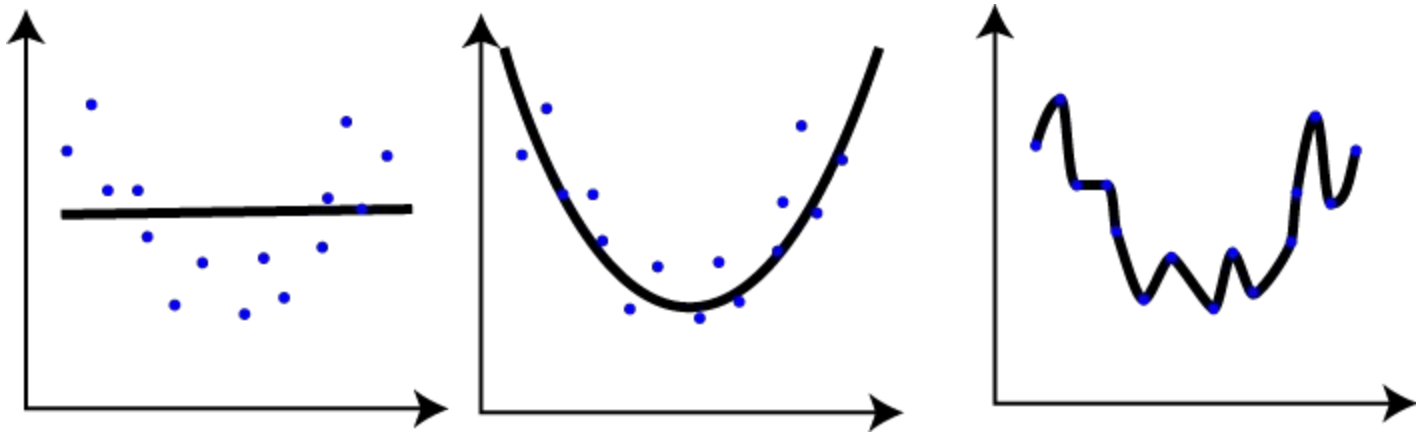
- En general la solución es cambiar a un modelo más complejo, más expresivo.

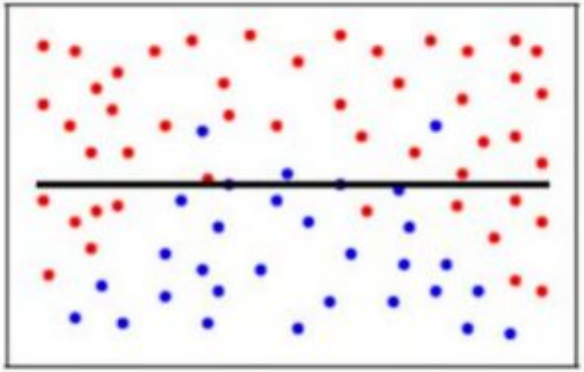
Overfitting

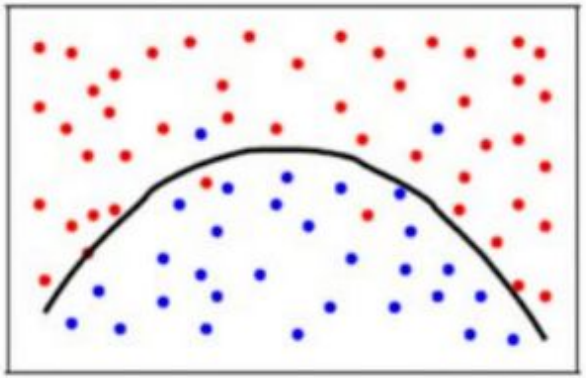
- El modelo tiene muy buen resultado para el set de entrenamiento pero no es tan bueno para el set de test.
- El modelo generaliza mal.
- El modelo “alucina”.
- El modelo es demasiado expresivo.

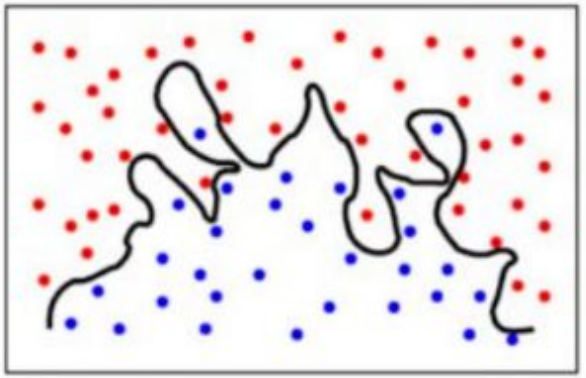
Overfitting

- Si el modelo es muy complejo puede haber memorizado el set de entrenamiento.
- Un modelo con muchos grados de libertad puede tener demasiadas soluciones posibles al mismo problema de optimización









Overfitting: Soluciones

- Conseguir más datos.
- Disminuir la complejidad del modelo.
- Usar regularización.

Regularización

- La regularización es una técnica por la cuál penalizamos a un modelo de machine learning en función de su complejidad.
- Ejemplo: Penalizamos un polinomio mediante lambda * sumatoria de todos los coeficientes al cuadrado.

Bias y Variance

Bias y Variance

- Es otra forma de ver el problema de underfitting y overfitting.

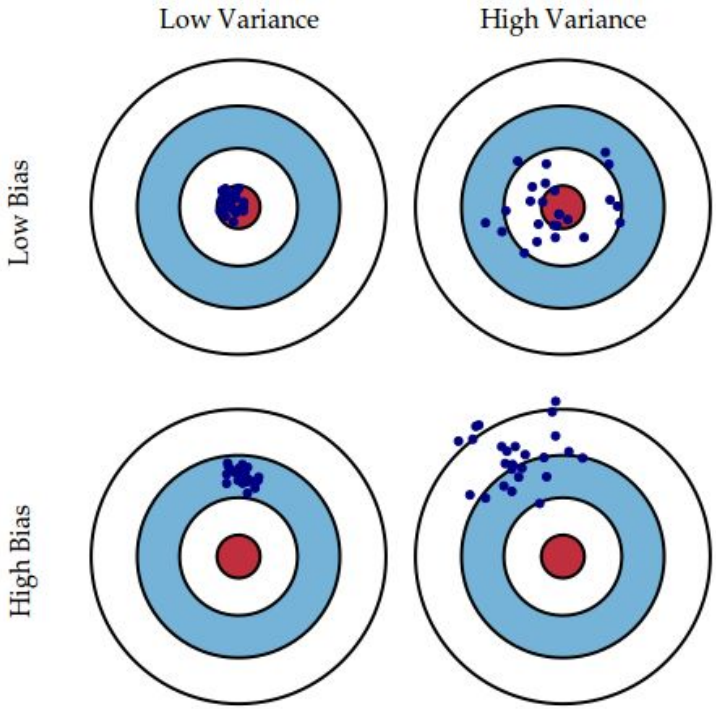
Bias & Variance

Bias:
Cuanto aproxima la estimación al set de entrenamiento.
Asociado al error que tenemos en el set de entrenamiento.

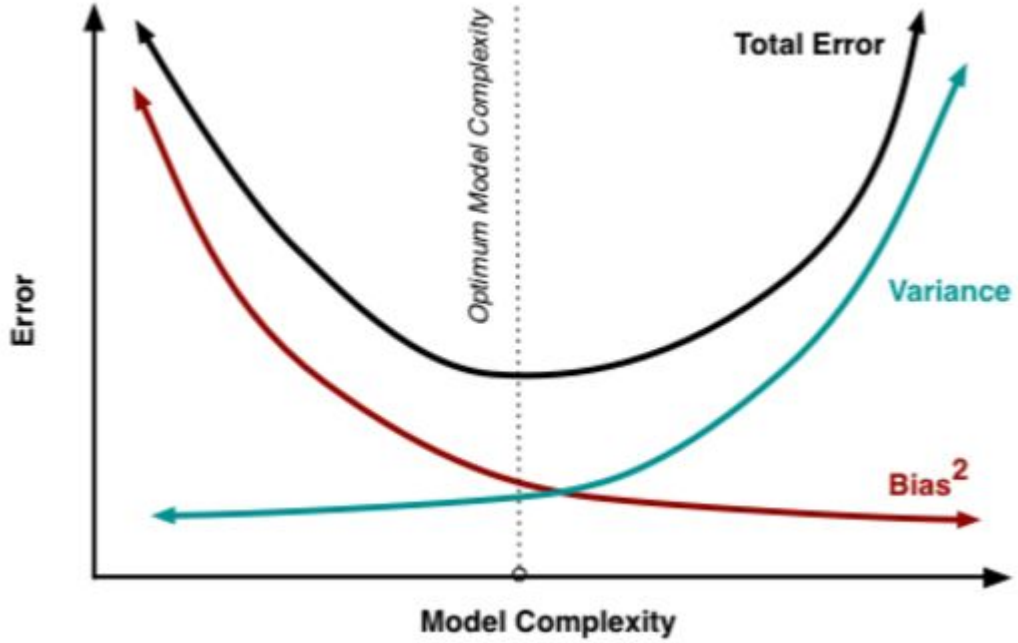
Variance: Cuantos grados de libertad tiene el resultado de la estimación.
Asociado al error que tenemos en el set de test o validacion.

	Low Bias	**High Bias**
Low Variance	Good 😊	Underfitting
High Variance	Overfitting	Not good 😞

Bias & Variance



Bias & Variance: Model complexity



No Free Lunch theorem

Wolpert (1997)

Dos algoritmos de optimización son equivalentes, si los promediamos para todos los problemas posibles.

Esto implica que no existe un algoritmo que sea bueno para cualquier problema, y el hecho de que un algoritmo sea bueno para un problema específico, necesariamente implica que es malo para otro problema

No Free Lunch

Observación	Label	Predicción
A	0	0
B	1	1
C	0	1
D	0	0

No Free Lunch

Observación	Label	Predicción
A	1	0
B	0	1
C	1	1
D	1	0

No Free Lunch

Conclusiones equivocadas.