TDVI: Inteligencia Artificial

Ensamble de Modelos I: Bagging / Random Forest

UTDT - LTD

Estructura de la clase

- Trade-off sesgo varianza
- Bagging
- Random forest

Estructura de la clase

- Trade-off sesgo varianza
- Bagging
- Random forest

Trade-off sesgo varianza

Recordatorio:

Setup:

- Existe un proceso generador de datos (***P(X, y)***). No lo conocemos
- Disponemos de instancias/observaciones generadas por dicho proceso. Las llamamos **train set** (X_{tr} e y_{tr}). OJO: podríamos haber tenido otro train set (primer componente aleatorio)
- Existe una relación del tipo $\boldsymbol{y = f(X) + \varepsilon}$. En donde 1) no conocemos *f* y en donde 2) ε es ruido irreducible (se cumple que $\boldsymbol{E(\varepsilon)=0}$)

Objetivo:

- Tendremos nuevas instancias (generadas por el mismo proceso) para las cuales sólo conoceremos *X*. Queremos estimar bien *y*, Las llamaremos **test set** (X_{ts} e y_{ts}). OJO: podríamos haber tenido otro test set (segundo componente aleatorio)

Estrategia

- Sobre la base de X_{tr} e y_{tr}, entrenamos un modelo de aprendizaje supervisado (e.g., un árbol) y obtenemos una estimación de f. Llamamos **f** al modelo

Trade-off sesgo varianza

Recordatorio:

Esta conceptualización da lugar a dos formas de pensar la performance/error del modelo:

- **Test error** (o generalization error): dado un training set *T*, es el error (*L*) esperado para una nueva observación de test independiente:

$$\mathrm{Err}_{\mathcal{T}} = \mathrm{E}[L(Y, \hat{f}(X))|\mathcal{T}]$$

- **Expected prediction error** (o expected test error): es el error esperado para una nueva observación de testing independiente teniendo en cuenta todo lo que es aleatorio (incluye la aleatoriedad intrínseca del proceso que genera X_{tr} e y_{tr})

$$\mathrm{Err} = \mathrm{E}[L(Y, \hat{f}(X))] = \mathrm{E}[\mathrm{Err}_{\mathcal{T}}]$$

Trade-off sesgo varianza

Supongamos que estamos en el caso de regresión y que medimos el error mediante el cuadrado de los residuos/errores. Se puede demostrar lo siguiente (futuro ejercicio de la guía 2):

$$\begin{aligned}
\mathrm{Err}(x_0) &= E[(Y - \hat{f}(x_0))^2 | X = x_0] \\
&= \sigma_\varepsilon^2 + [\mathrm{E}\hat{f}(x_0) - f(x_0)]^2 + E[\hat{f}(x_0) - \mathrm{E}\hat{f}(x_0)]^2 \\
&= \sigma_\varepsilon^2 + \mathrm{Bias}^2(\hat{f}(x_0)) + \mathrm{Var}(\hat{f}(x_0)) \\
&= \text{Irreducible Error} + \mathrm{Bias}^2 + \text{Variance.}
\end{aligned}$$

Dos aclaraciones:

1. Es el error esperado para una observación cualquiera de testing
2. La esperanza es sobre todos los posibles datasets de entrenamiento

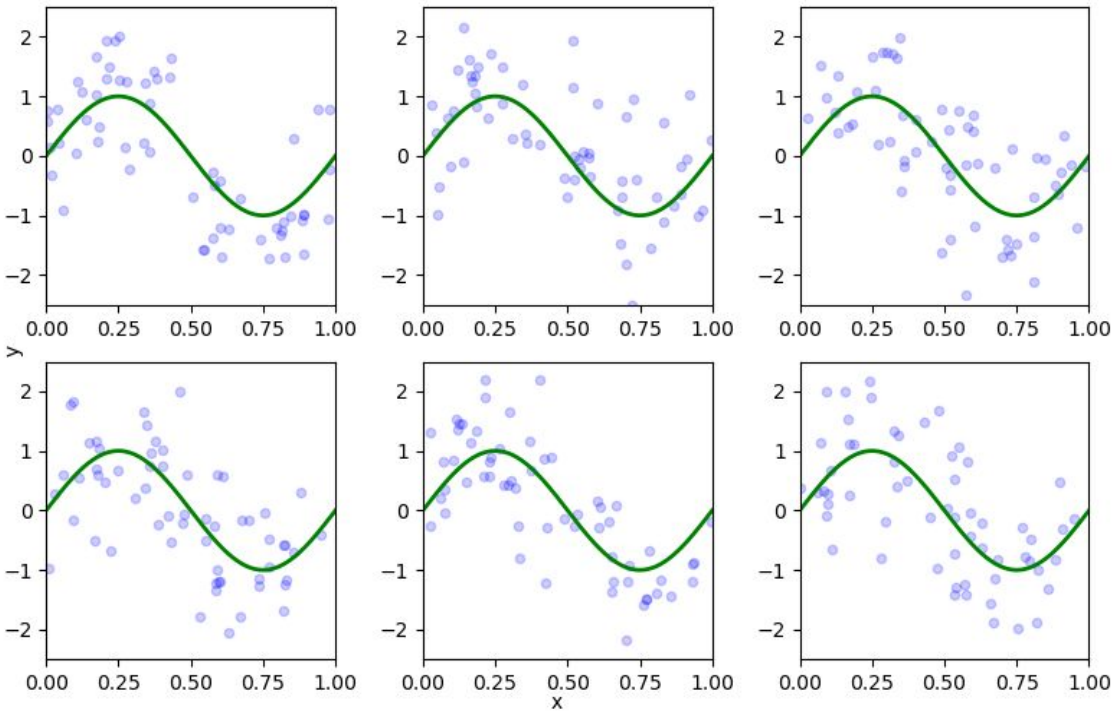
Tratemos de ver esto de una manera más visual

Aclaración: las mismas intuiciones que veremos para regresión son válidas para problemas de clasificación

Trade-off sesgo varianza

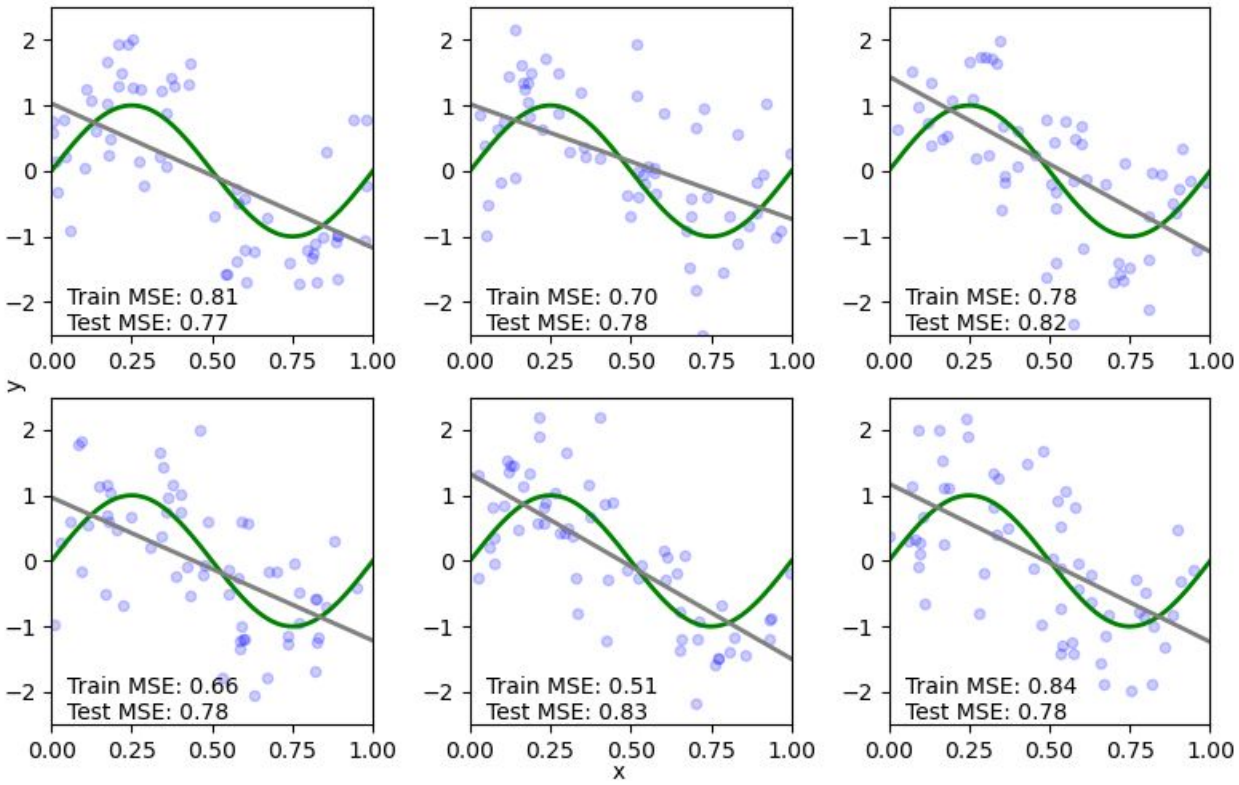
Supongamos que podemos ver seis instanciaciones distintas de $y_i = f(x_i) + \varepsilon_i$, en donde:

1. x ~ U(0, 1)
2. $f(x_i) = sin(2 * \pi * x_i)$ (curva verde)
3. $\varepsilon \sim N(0, 0.7^2)$
4. $N = 100$



Trade-off sesgo varianza

¿Qué hubiera pasado al predecir con una regresión lineal?



¿La función predicha varía mucho según la instanciación del train set?

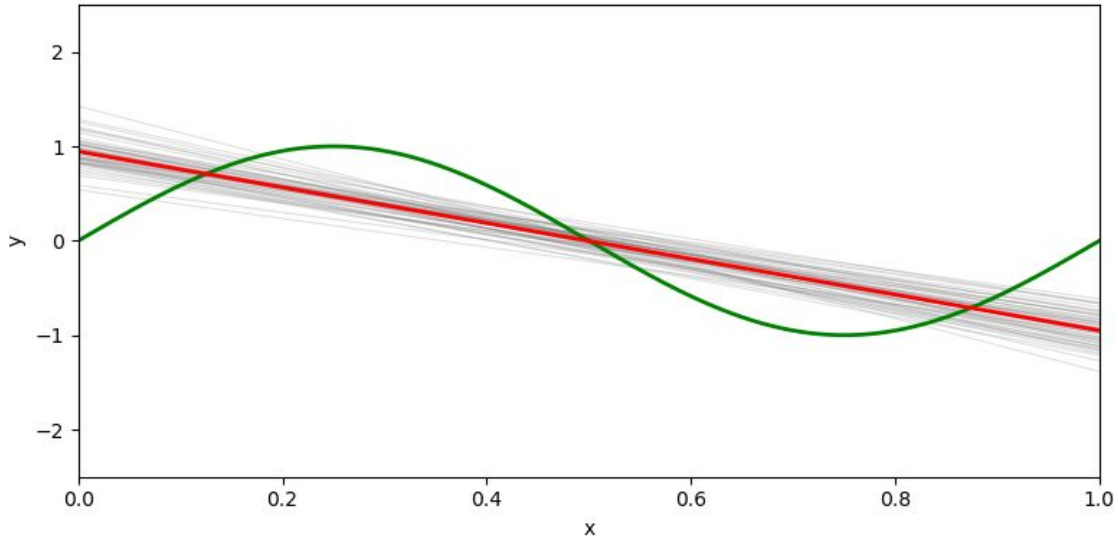
¿Si se promedian las curvas grises, el promedio será cercano a *f*?

¿Esto modelo hará overfitting o underfitting?

Trade-off sesgo varianza

Esta figura muestra el resultado de estimar la regresión lineal sobre 50 instanciaciones distintas del train set

Luego para cada valor de x se promedian las 50 predicciones (línea roja)



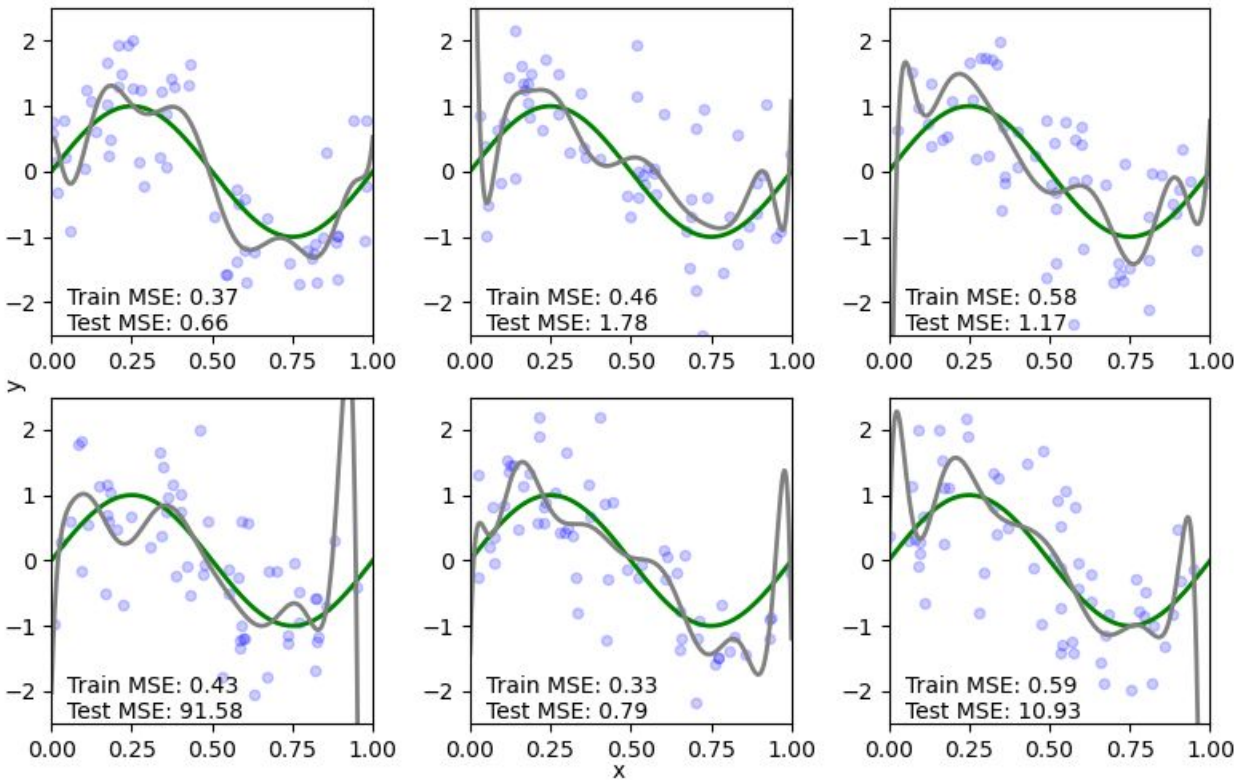
$$\sigma_\varepsilon^2 + [\mathrm{E}\hat{f}(x_0) - f(x_0)]^2 + E[\hat{f}(x_0) - \mathrm{E}\hat{f}(x_0)]^2$$

Patrones interesantes:

- El promedio de las curvas es lejano a la f verdadera (alto sesgo) → Malo
- Las 50 predicciones son cercanas al promedio (baja varianza) → Bueno

Trade-off sesgo varianza

¿Qué hubiera pasado al predecir con una regresión polinómica (grado 12)?



¿La función predicha varía mucho según la instanciación del train set?

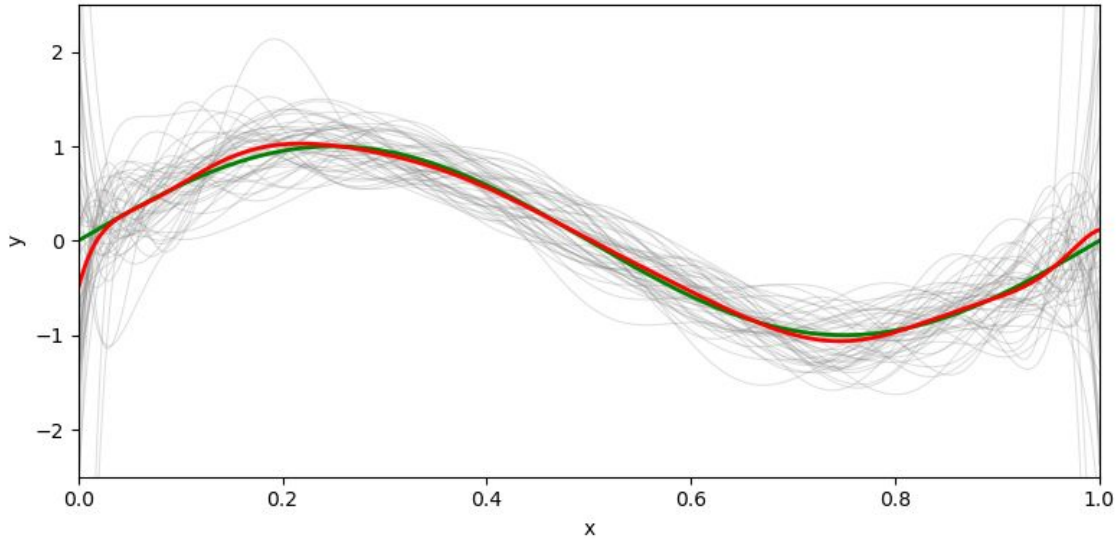
¿Si se promedian las curvas grises, el promedio será cercano a *f*?

¿Esto modelo hará overfitting o underfitting?

Trade-off sesgo varianza

Esta figura muestra el resultado de estimar la regresión polinómica sobre 50 instanciaciones distintas del train set

Luego para cada valor de x se promedian las 50 predicciones (curva roja)



$$\sigma_\varepsilon^2 + [\mathrm{E}\hat{f}(x_0) - f(x_0)]^2 + E[\hat{f}(x_0) - \mathrm{E}\hat{f}(x_0)]^2$$

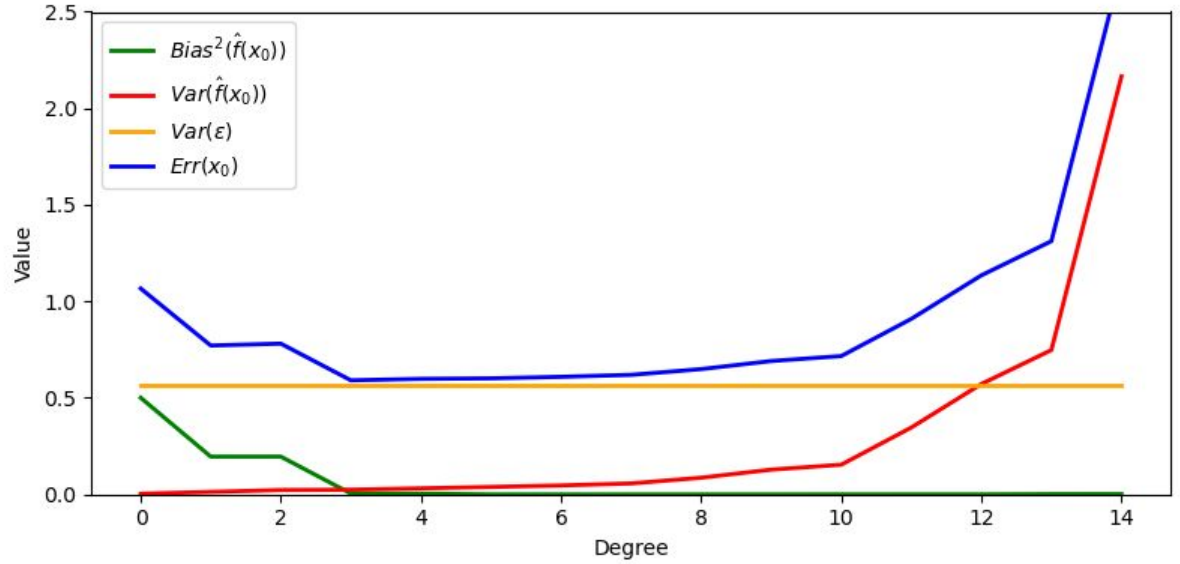
Patrones interesantes:

- El promedio de las curvas es cercano a la f verdadera (bajo sesgo) → Bueno
- Las 50 predicciones varían mucho respecto al promedio (alta varianza) → Malo

Trade-off sesgo varianza

Empíricamente se observa que a mayor flexibilidad de un algoritmo, menor el sesgo, pero mayor será la varianza. Este es el trade-off sesgo varianza

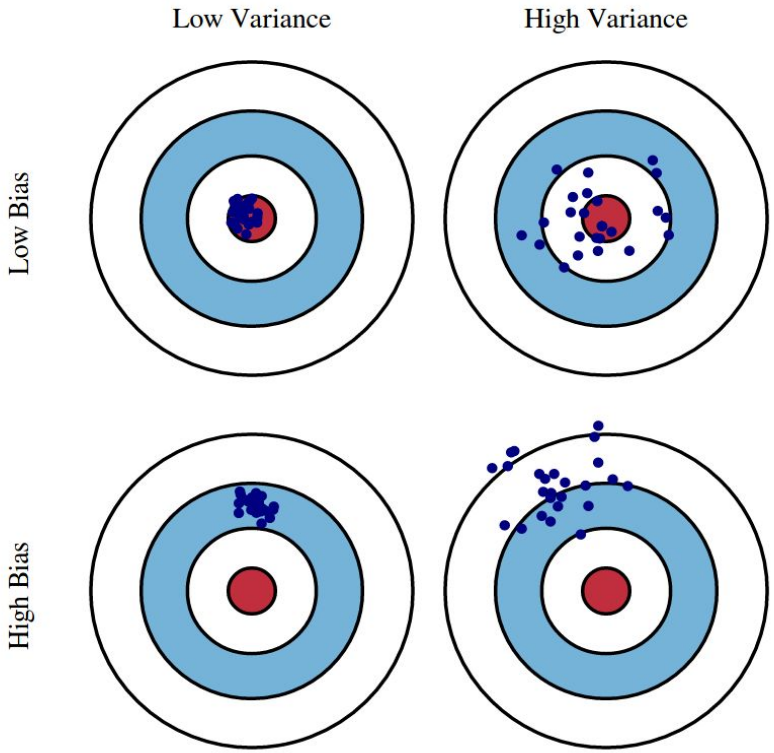
$$Err(x_0) = \sigma_\epsilon^2 + Bias^2(\hat{f}(x_0)) + Var(\hat{f}(x_0))$$



- Falta de flexibilidad → alto sesgo y baja varianza → se asocia a underfitting
- Excesiva flexibilidad → bajo sesgo y alta varianza → se asocia a overfitting

Trade-off sesgo varianza

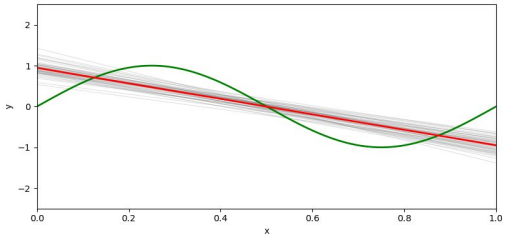
Repaso aún más visual

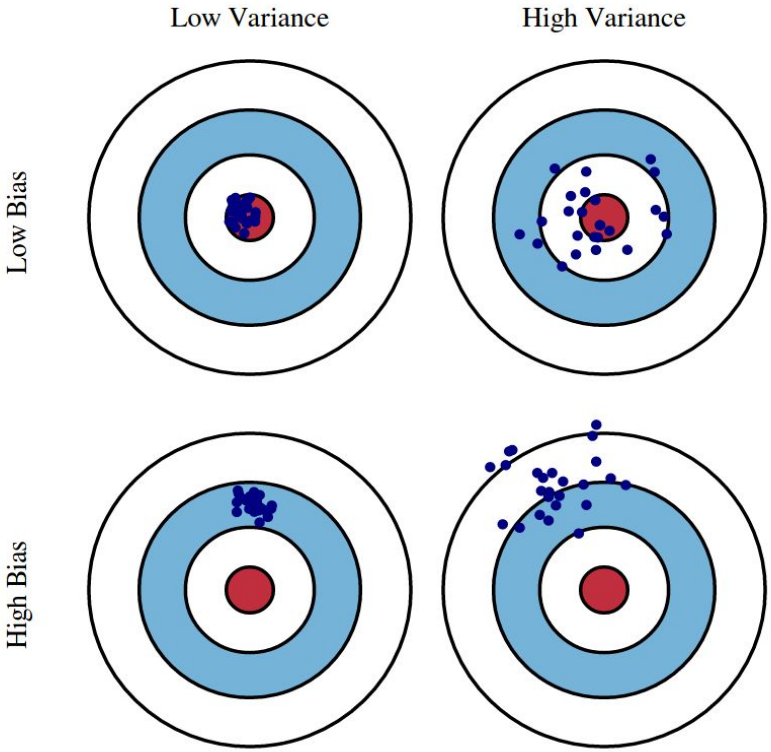


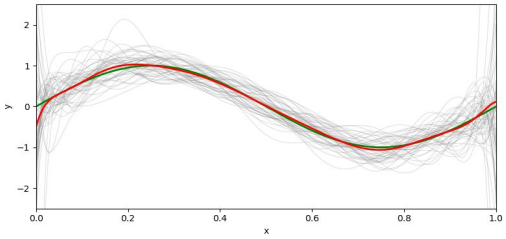
¿En qué caso esperan que un siguiente disparo acierte mejor? ¿Por qué en los otros no lo esperan?

Trade-off sesgo varianza

Repaso aún más visual





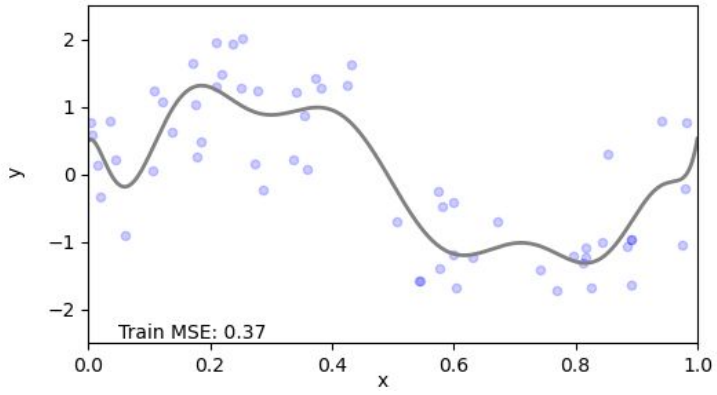


¿Cómo sería para bajo sesgo y baja varianza? ¿Cómo sería para alto sesgo y alta varianza?

Trade-off sesgo varianza

Aclaraciones importantes:

1. No olviden que $Err(X_o)$ es la esperanza del error antes de haber visto el train set, sujeto a 1) un proceso generador de datos y 2) un algoritmo de aprendizaje propuesto
2. En la práctica no conocemos ni a $Err(X_o)$, ni al sesgo del clasificador, ni a la varianza del clasificador, ni a la varianza del error. Este análisis es una forma de modelar cómo se descompone $Err(X_o)$
3. Sólo conocemos una única instanciación del train set y desconocemos la verdadera *f* (¡la queremos predecir!)



¿Cómo podremos aprovechar el trade-off sesgo varianza?

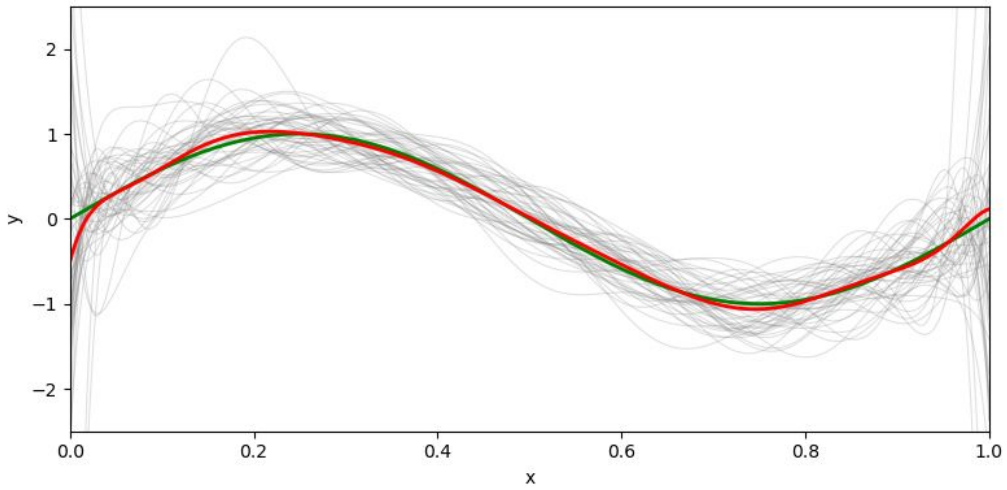
Estructura de la clase

- Trade-off sesgo varianza
- Bagging
- Random forest

Bagging

Enfoquémonos en el promedio de los modelos (curva roja)

Recuerden que si uno tiene Z_1, …, Z_n valores independientes, cada uno con varianza σ_Z^2, el promedio de las mismas tendrá varianza σ_Z^2 / n

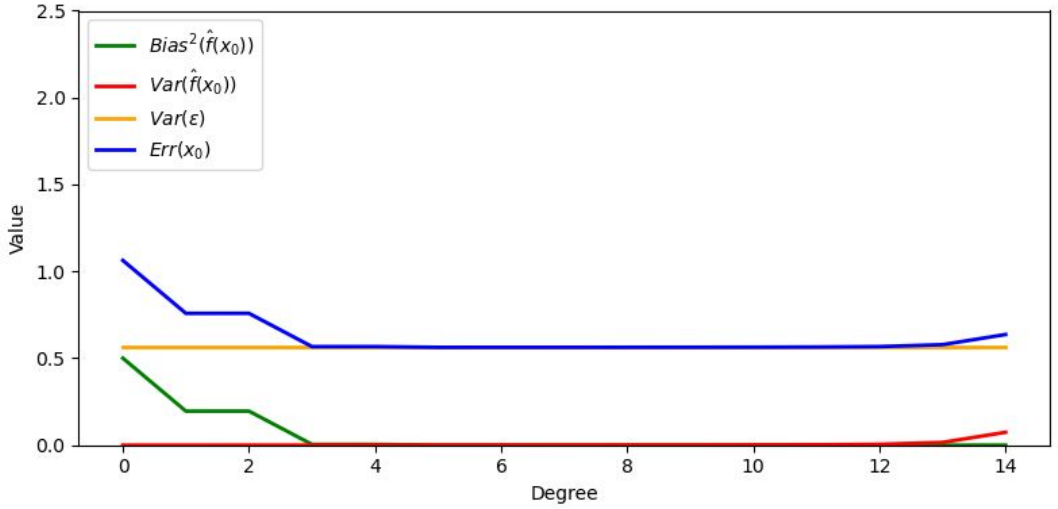


IMPORTANTE: en nuestro contexto "*el promedio de los modelos tendrá menor varianza*" significa que 1) si volvemos a instanciar 50 veces el train set y 2) volvemos a promediar los 50 modelos ajustados → la curva roja no cambiará mucho (tendrá poca varianza, esto es bueno)

Bagging

Idea: 1) instanciemos muchas veces el train set, 2) entrenemos modelos que ajusten bien (o sobreajusten algo) cada instanciación y 3) promediemos sus predicciones. Este promedio será el **modelo final** (tendrá bajo sesgo y también baja varianza!)

¿Cómo se ve esto en nuestro ejercicio de simulación?



¿Pero con datos de verdad, se puede hacer esto? ¿Uno puede tener nuevas instancias de entrenamiento?

Bagging

Realidad: disponemos de una única instanciación del train set (no podemos generar genuinamente nuevas instancias del train set)

Idea: ¡"inventemos" nuevas instanciaciones del train set en donde cada una tenga diferente ruido!

¿¡Cómo inventar nuevas instanciaciones del train set!?

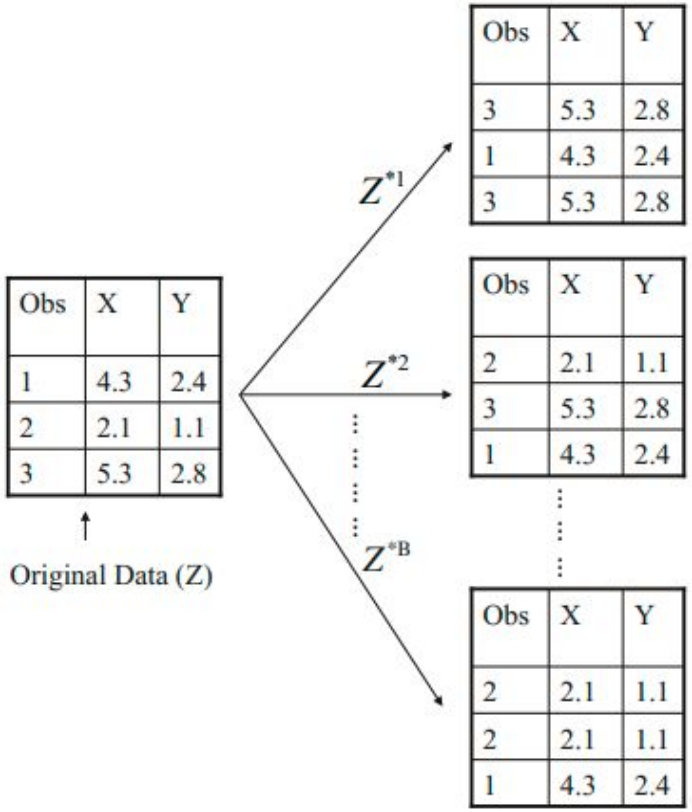
Bagging

Bootstrapping

Generemos nuevos train sets “bootstrapeados” muestreando observaciones del train set original al azar y con reposición (es como simular el proceso generador de datos)

- Para un train set particular, se pueden generar tantos train sets bootstrapeados como se quiera

- Para datasets grandes, en promedio, cada train set bootstrapeado tendrá cerca de 2/3 de las observaciones del original (algunas repetidas). 1/3 no serán usadas (*out-of-bag observations*)



Bagging

Bagging (**b**ootstrap **agg**regat**ing**)

1. Generamos *B* train sets bootstrapeados a partir del train set original
2. Para cada uno de los *B* train sets bootstrapeados entrenamos un modelo muy flexible (i.e., con bajo sesgo y posiblemente alta varianza - tampoco excesiva)
3. Para toda observación x, se toma como predicción final al promedio de las predicciones generadas por cada uno de los *B* modelos entrenados

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}$$

Este modelo debiera tener bajo sesgo y baja varianza

Consideraciones:

1. ¿Qué valor de *B* elegir? Mientras mayor, mejor. En principio, aumentarlo no debiera generar overfitting
2. ¿Qué se promedia en clasificación? Las probabilidades condicionales predichas

Bagging

Bagging es un ejemplo de lo que se conoce como un modelo de ensamble

Un modelo que combina las predicciones de diferentes modelos (a los que se llama modelos base)

Comunmente, ensamblar modelos da lugar a mejoras en performance (siendo uno de los costos a pagar la pérdida de interpretabilidad)

Bagging

Out-of-bag (oob) error estimation (casi de “yapa”)

Con bagging uno puede estimar el error en testeo sin necesidad de armar un validation set

Es válido predecir la performance sobre una observación *i* de entrenamiento utilizando las predicción de aquellos árboles para los cuales *i* fue una out-of-bag observation (aproximadamente de *B*/3 predicciones)

Estructura de la clase

- Trade-off sesgo varianza
- Bagging
- Random forest

Random forest

Problema: el promedio reduce la varianza fuertemente sólo cuando cada valor promediado (Z) es independiente de los demás ($\rho=0$, siendo ρ el índice de correlación de pearson entre pares de valores)

Si no son independientes, la varianza viene dada por la siguiente expresión:

$$Var(\frac{1}{n}\sum_{i=1}^{n} Z_i) = \frac{\sigma_Z^2}{n} + (1 - \frac{1}{n})\rho\sigma_Z^2$$

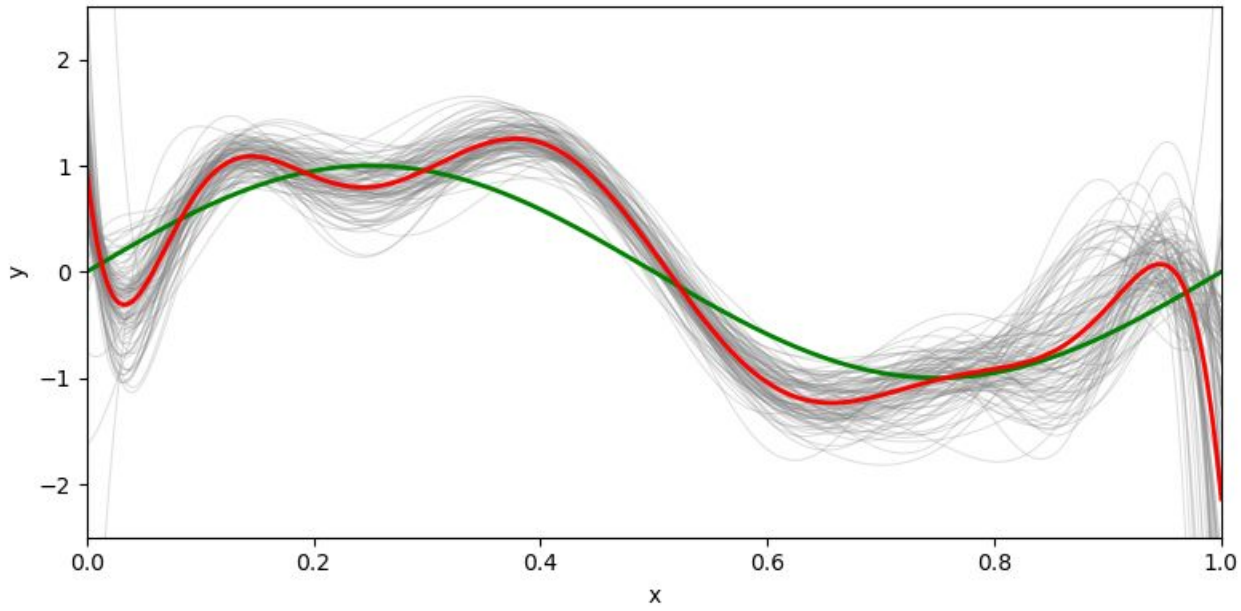
Noten que ahora, cuando $n \to \infty$ la varianza de la media muestral no tiende a 0 (como antes), tiende a $\rho\sigma_Z^2$ ($\rho = 1$ es como sacar siempre la misma muestra)

Si ρ es alta, promediar ya no ayuda tanto

Random forest

¿Qué sucede con bagging? Los modelos entrenados (y sus predicciones) no difieren tanto entre sí, los modelos suelen estar muy correlacionados

Visualización de los modelos base y final predichos utilizando bagging:



Consecuencia: en el nuevo modelo (el ensamble), no se baja tanto la varianza
→el error esperado no baja tanto

Random forest

Ahora nos enfocaremos en el caso de árboles con muchos predictores

¿Por qué los distintos árboles base podrían estar correlacionados?

Primero recordemos cómo se eligen los splits en árboles de decisión:

Para **cada variable *j*** y cada punto de corte *s*, definimos las siguientes regiones y se elige la de mayor ganancia

$$R_1(j,s) = \{X | X_j < s\} \text{ and } R_2(j,s) = \{X | X_j \geq s\}$$

Random forest

Supongamos que existe un predictor fuerte junto a otros predictores moderados, entonces, en bagging, la mayoría de los árboles utilizarán el predictor fuerte en su primer(os) split(s)

Esto hará que los árboles sean estructuralmente similares y, por ende, hará que sus predicciones estén correlacionadas (malo)

Idea: decorrelacionar los árboles perturbando la manera en que se construyen, y así lograr que sean más diversos (pero sin que cada uno deje de ajustar bien - o sobreajustar algo - los train sets bootstrapeados)

¿!Pero cómo hacerlo!?

Random forest

Random forest: casi idéntico a bagging (se entrenan sucesivos árboles sobre train set boostrapeados), pero ahora, en cada split de cada árbol, en vez de considerar todos los predictores, se evalúan sólo un subconjunto muestreado al azar de los mismos (que varía de split a split)

Nueva versión: para **cada variable *j* dentro de una muestra al azar de todas las variables** y cada punto de corte *s*, se definen las siguientes regiones y se elige la de mayor ganancia

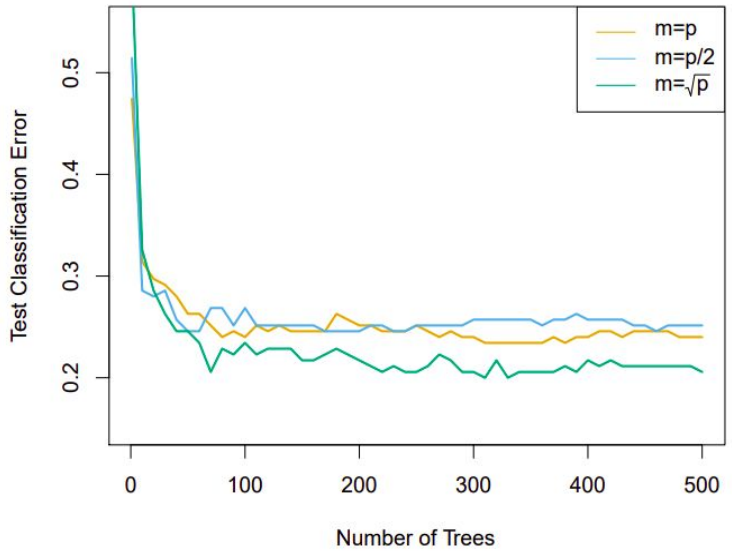
$$R_1(j,s) = \{X|X_j < s\} \text{ and } R_2(j,s) = \{X|X_j \geq s\}$$

- **Efecto I**: cada modelo base tendrán un poco más de varianza (relativo a utilizar siempre todos los predictores), pero al decorrelacionarlos, la varianza del ensamble será menor (generando una reducción de $Err(X_0)$)
- **Efecto II**: el ensamble se entrena más rápido que en el caso de bagging

Random forest

Al igual que en bagging, se puede calcular el *out-of-bag estimator error* y la importancia de atributos

Si p es el número de predictores, suele recomendarse muestrear $\sqrt{p}$ precitores por split. Técnicamente el número a samplear es un hiperparámetro cuyo valor se define experimentalmente



Modificar otros hiperparámetros (por ejemplo: la profundidad de los árboles), puede mejorar la performance del ensamble

Random forest

Para pensar:

Recordemos una vez más cómo se eligen los splits en árboles de decisión:

Para cada variable *j* y cada punto de corte *s*, definimos las siguientes regiones y se elige la de mayor ganancia

$$R_1(j,s) = \{X | X_j < s\} \text{ and } R_2(j,s) = \{X | X_j \geq s\}$$

Random forest hace 1) bagging y 2) muestrea variables al azar en cada split

¿Será buena idea muestrear también puntos de corte (*s*) en cada split candidato?

Random forest

Probemos bagging y random forest en Python

Bibliografía

- ISLP. Capítulo 2 entero y capítulo 8 (salvo lo referido a BART)

Bagging - Extra

Importancia de atributos en bagging

En bagging, con árboles de decisión como modelos base, se puede obtener la importancia de cada atributo

Comúnmente se hace de la siguiente manera:

1) En cada árbol del ensamble se calcula la importancia tal cual se hace en árboles individuales

2) Luego, para cada predictor, se promedia la importancia estimada por los distintos árboles