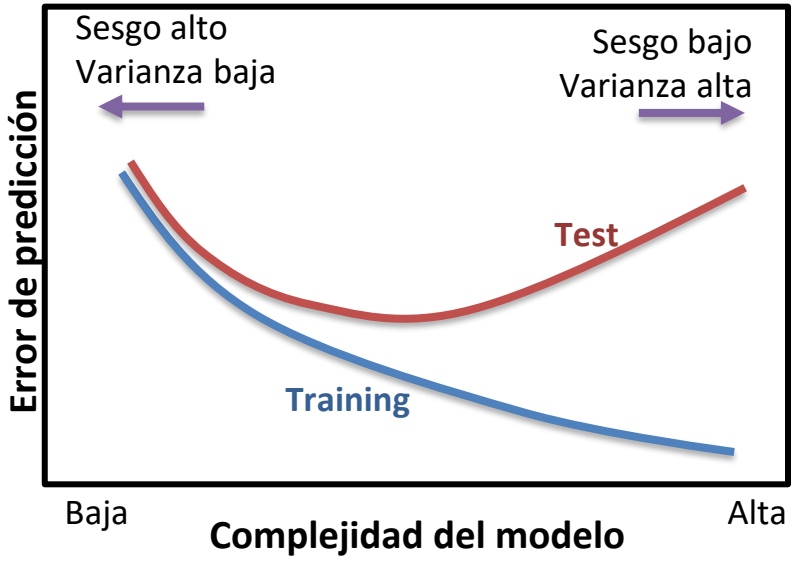
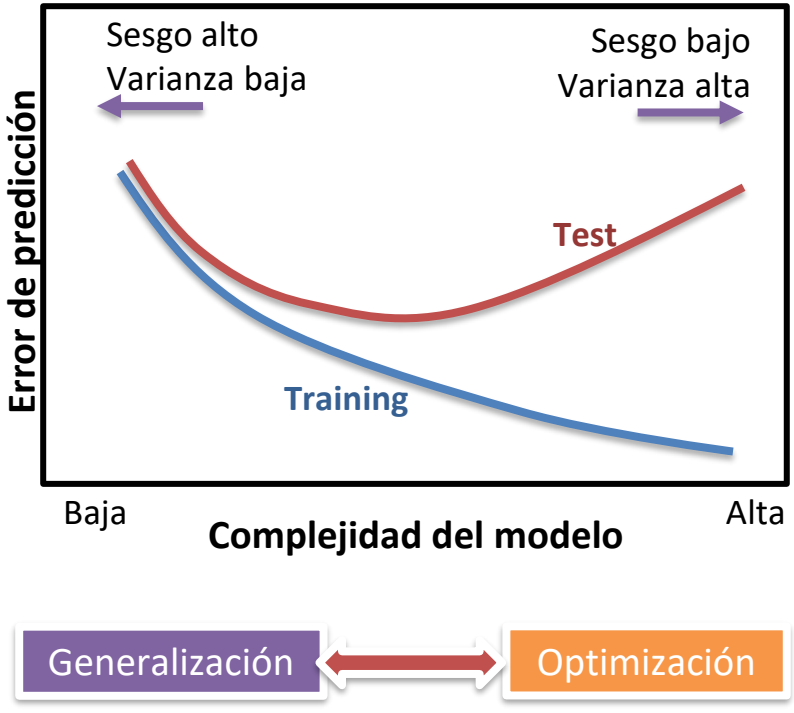
Overfitting / Underfitting

- En general, los errores de “training” siempre tienden a disminuir.
- Sin embargo, los errores de “test” disminuyen al inicio (tanto como dominen las reducciones en el sesgo), pero entonces comienzan a incrementarse de nuevo (tanto como domine el incremento en la varianza).

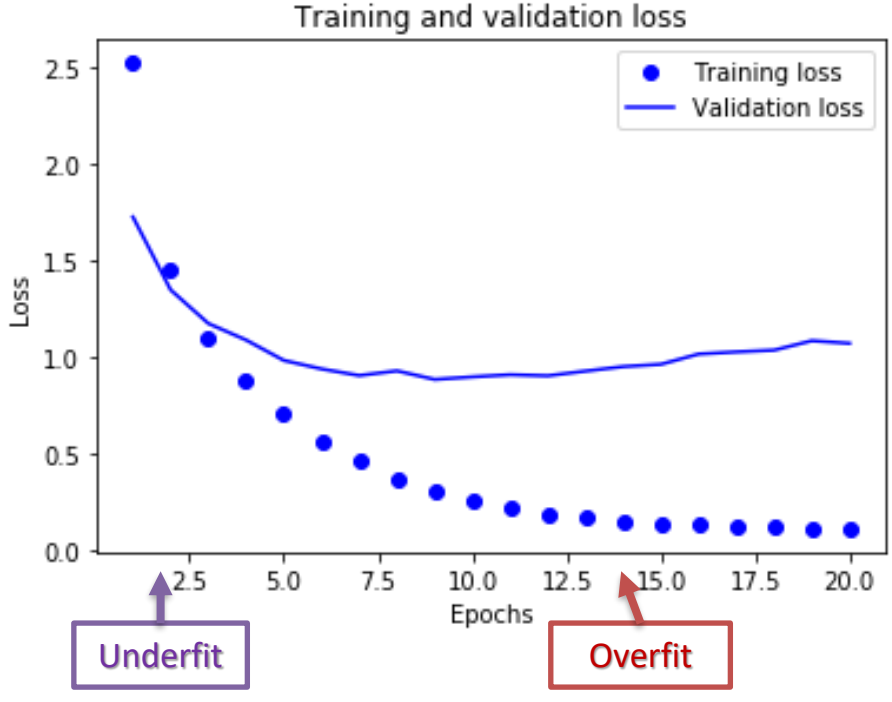


Debemos recordar esta gráfica cuando se escoja un método de aprendizaje. Una mayor flexibilidad/complejidad no es siempre lo mejor.

Overfitting / Underfitting



Generalización ⟷ Optimización



¿Cómo evitar el overfitting?

- ¿Cómo evitar el overfitting? (evitar que aprenda patrones falsos o irrelevantes de los datos de entrenamiento)
- La mejor solución: obtener más datos de entrenamiento.
- La siguiente mejor solución:
  - **Regularización**: *modular la cantidad de información* que el modelo puede almacenar o agregar *restricciones* sobre qué información puede almacenar el modelo.
- Si una red solo puede permitirse memorizar una pequeña cantidad de patrones, el proceso de optimización la obligará a centrarse en los patrones más destacados, que tienen una mayor probabilidad de generalizarse bien.

Reducir la capacidad de la red

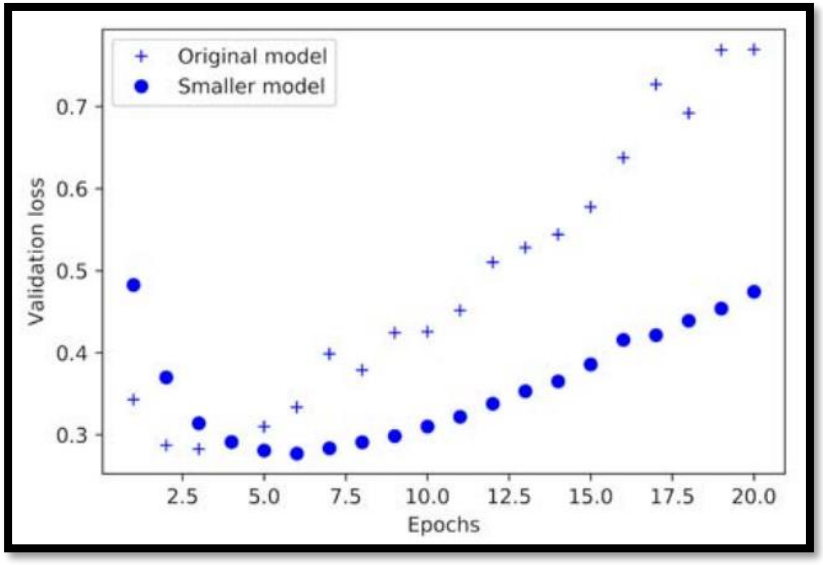
Regularización de los pesos

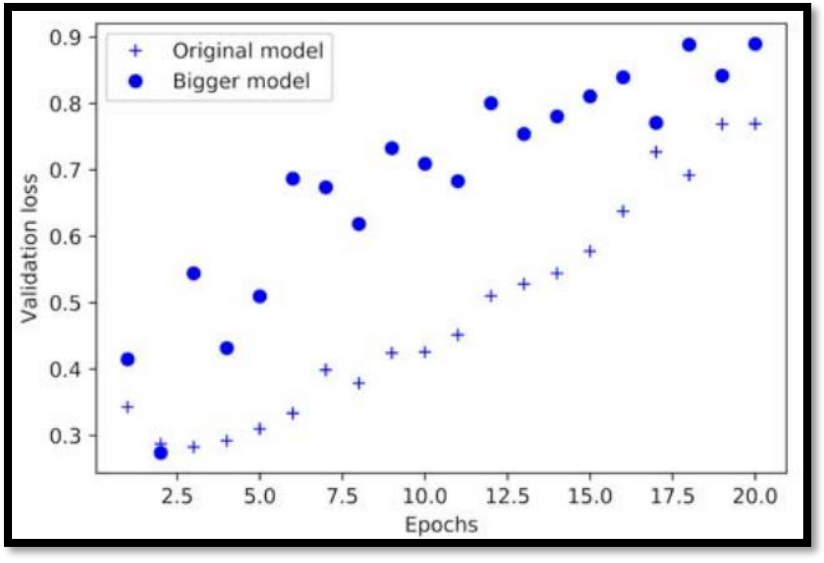
Dropout

Regularización: reducir el tamaño de la red

- El tamaño de una red (neuronas y capas) define su capacidad de memoria para aprender.
- Una red con tamaño muy grande memorizará todos los datos y no tendrá generalización alguna (overfitting).
- Una red con tamaño muy pequeño no tendrá capacidad para memorizar los patrones mínimos (underfitting).
- Si la red tiene un tamaño medio adecuado (con recursos de memoria limitados), no será capaz de hacer una representación tan fácil y tendrá que recurrir a aprender representaciones comprimidas con poder predictivo.

Regularización: reducir el tamaño de la red

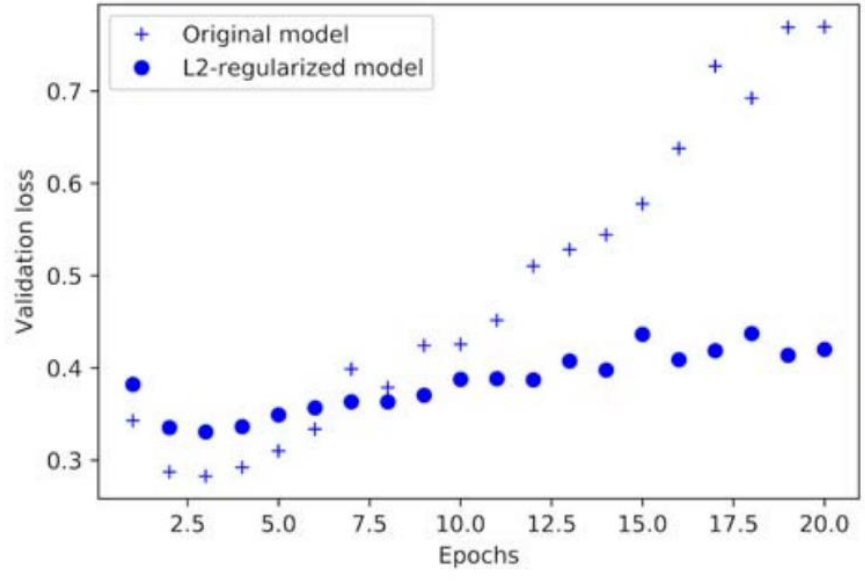




Regularización de los pesos -> Weight Decay

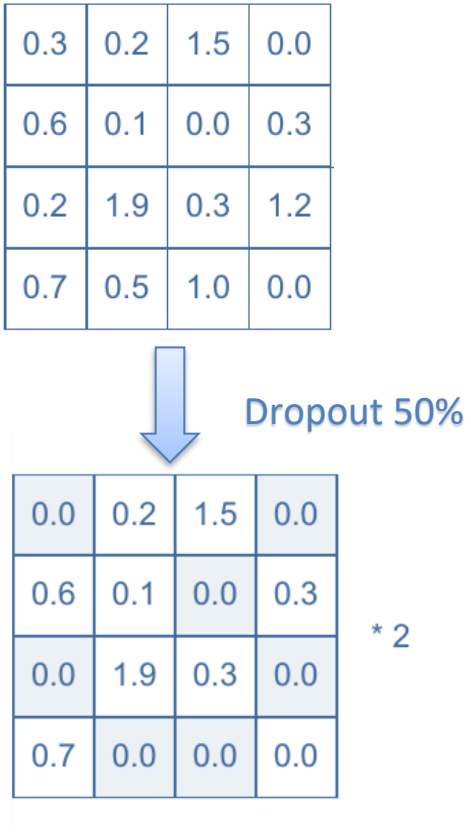
$$Loss\ completa = Loss + weightdecay * Norma\ L2\ \ weights$$

```
optimizer = torch.optim.SGD(model.parameters(), lr=1e-3, weight_decay=1e-4)
```



Dropout

- Dropout es una de las técnicas más efectivas y más utilizadas para las NN.
- Consiste en anular aleatoriamente varios descriptores de salida (output features) durante el entrenamiento (colocar un valor de cero).
- La tasa de dropout es la fracción de descriptores de salida que se reducen a cero (generalmente entre 0.2 y 0.5).
- La idea central es que introducir ruido en los valores de salida de una capa puede romper los patrones de casualidad que no son significativos y que la red comenzaría a memorizar si no hay ruido presente.




```
torch.nn.Dropout(p=0.5)
```

Y hay más…

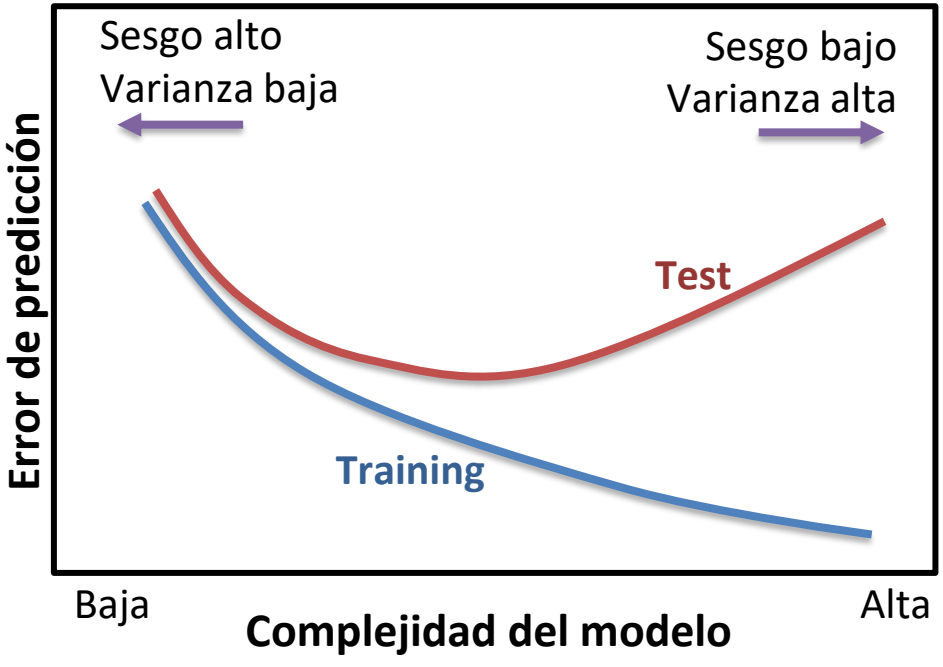
- **Scheduling**: https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate
- **Batch normalization:** https://www.machinecurve.com/index.php/2020/01/14/what-is-batch-normalization-for-training-neural-networks/
- **Early stopping**: https://clay-atlas.com/us/blog/2021/08/25/pytorch-en-early-stopping/
- **Optimizadores**: https://ruder.io/optimizing-gradient-descent/
- **Weight initialization**: https://www.deeplearning.ai/ai-notes/initialization/
- Y mucho más: CNN, RNN, LSTM, GANs, Transformers, ViT, …

Contenido

- Conjunto de validación
- Validación cruzada dejando uno fuera
- Validación cruzada dejando K fuera
- Repaso de métricas de clasificación
- Bootstrap

RESUMEN: Un esquema fundamental

- En general, los **errores de "training"** siempre tienden a disminuir.
- Sin embargo, los **errores de "test"** disminuyen al inicio (tanto como dominen las reducciones en el sesgo), pero entonces comienzan a incrementarse de nuevo (tanto como domine el incremento en la varianza).

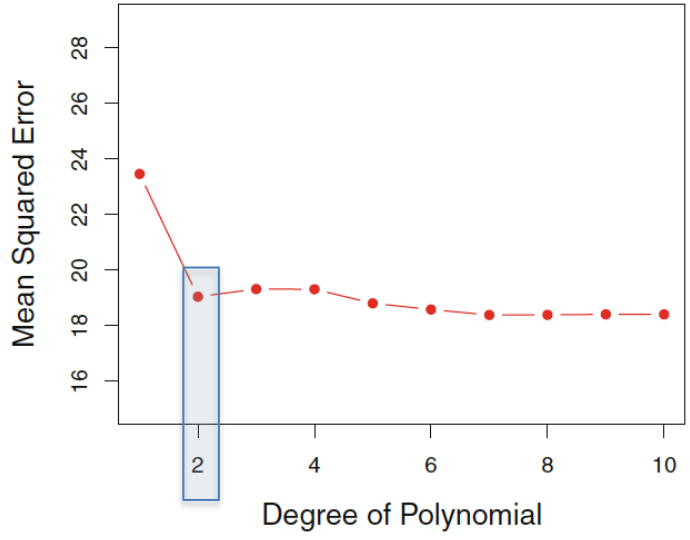


Debemos recordar esta gráfica cuando se escoja un método de aprendizaje. Una mayor flexibilidad/complejidad no es siempre lo mejor.

Ejemplo de regresión usando un conjunto de validación

Ahora el conjunto de observaciones se **separa aleatoriamente** en dos conjuntos con 196 observaciones cada uno.

Error cuadrático medio para una única validación con varios modelos según el grado del polinomio

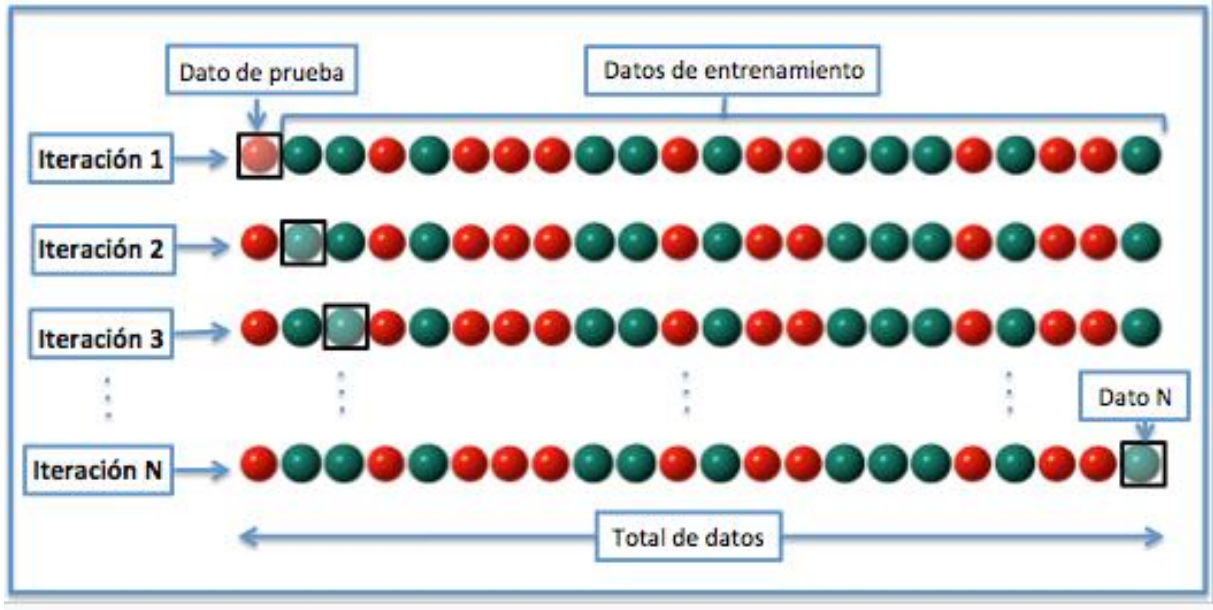


Validación cruzada

Método de validación cruzada dejando uno fuera
Leave-one-out cross-validation

- Se deja un elemento fuera del conjunto de training: (x_1, y_1)
- El resto se usa para el training: $\{(x_2, y_2), \ldots, (x_n, y_n)\}$

Esto se realiza iterativamente dejando fuera un nuevo elemento en cada iteración



El error cuadrático medio de la validación es

$$\mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\mathrm{MSE}_i$$

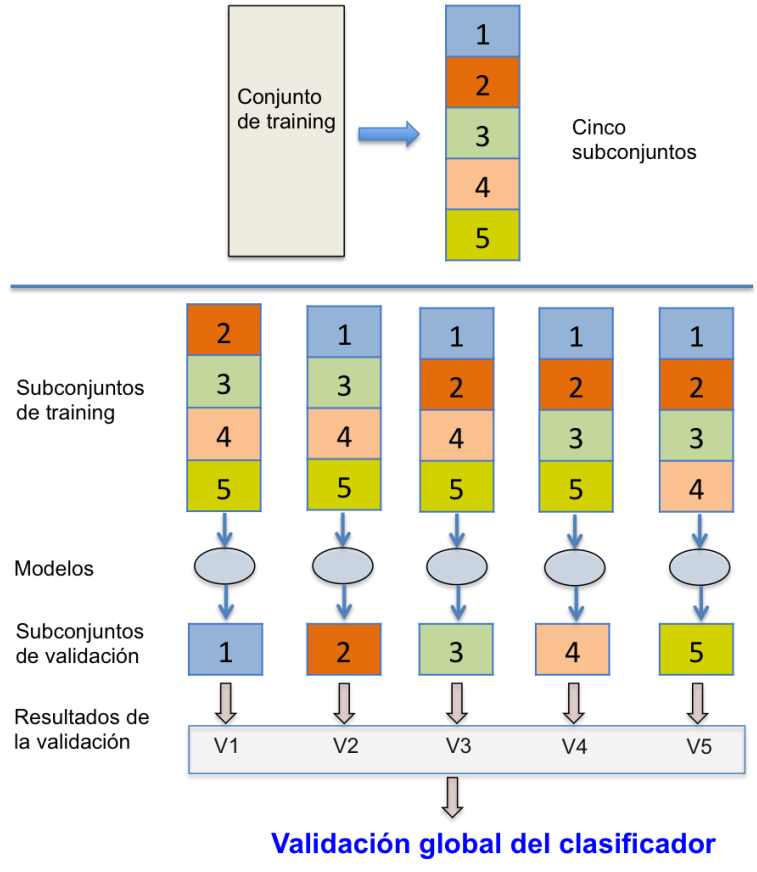
Validación cruzada

Método de validación cruzada dejando K fuera
K-fold cross-validation

Es una extensión del caso anterior.

El **conjunto de entrenamiento se divide aleatoriamente en varias particiones** de igual tamaño (por ejemplo 5).

Cada partición se usa una vez como **conjunto de validación,** con los otros cuatro grupos como **conjunto de entrenamiento**.



Validación cruzada

Método de validación cruzada dejando K fuera
K-fold cross-validation

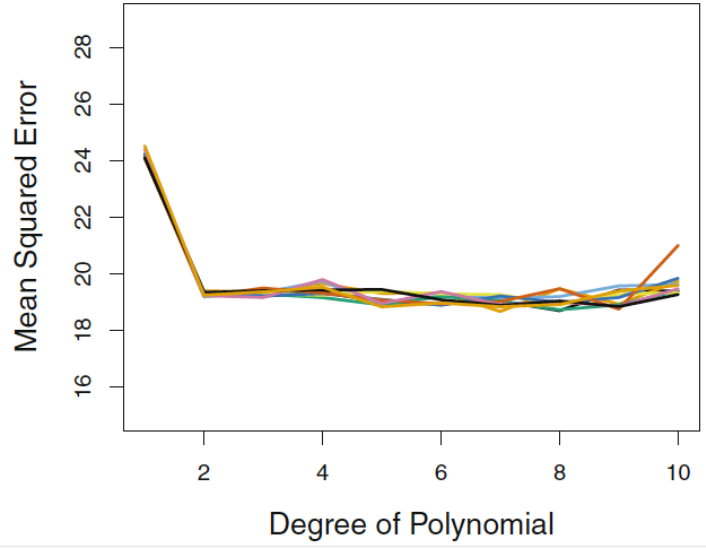
En cada iteración se realiza la validación **prediciendo las respuestas** correspondientes al conjunto que se ha dejado y su **error cuadrático MSEi**
El error cuadrático medio de la validación es

$$\mathrm{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k}\mathrm{MSE}_i$$

Ejemplo anterior con k=10

Curvas del error para cada iteración

Se observa poca variabilidad en el error



Validación cruzada – algunos comentarios prácticos

Si se divide el conjunto de datos disponibles en un conjunto de training y otro de validación, y ésta se realiza una única vez, **el error de validación no será una buena estimación del error de las predicciones futuras** porque se habrán usado solo la mitad de los datos disponibles. **Aumenta el sesgo.**

Según esto, la **validación cruzada dejando uno fuera** (*equivale a K=n*), parece la mejor opción porque se usa prácticamente todo el conjunto de training. El inconveniente es doble: **aumenta la varianza** y tiene mucho mayor **coste computacional** (se repite n veces).

La **validación cruzada con k=5 o K=10 ofrece un buen compromiso entre sesgo y varianza.** No tiene excesivo sesgo ni excesiva varianza.

Sirve para ajustar hiperparámetros del clasificador.

Es el caso más utilizado en la práctica.

Medidas del rendimiento de un clasificador

Matriz de confusión

Ejemplo de clasificación de 60000 imágenes MIST

Clasificación

Clases Verdaderas	CLASIFICACIÓN NEGATIVA	CLASIFICACIÓN POSITIVA
CLASE NEGATIVA	Verdadero Negativo VN = 53272	Falso Positivo FP = 1307
CLASE POSITIVA	Falso Negativo FN = 1077	Verdadero Positivo VP = 4344

Medidas del rendimiento de un clasificador

Exactitud (*accuracy*)

Es la proporción de aciertos:

$$exactitud = \frac{VN + VP}{total\ de\ objetos} = \frac{53272 + 4344}{60000} = 0.96 = 96\%$$

Precisión (valor predictivo positivo)

Es la exactitud de las predicciones positivas. Es decir, la proporción de verdaderos positivos (aciertos) sobre el total de objetos que han sido clasificados como positivos, esto es

$$Precisión = \frac{VP}{VP + FP} = \frac{4344}{4344 + 1307} = 0.77 = 77\%$$

Sensibilidad (Proporción de verdaderos positivos - PVP) o Recall

Es la proporción de objetos positivos que son clasificados correctamente. Es decir, la proporción de aciertos dentro del conjunto total de objetos que son realmente positivos:

$$Sensibilidad = PVP = \frac{VP}{VP + FN} = \frac{4344}{4344 + 1077} = 0.79 = 79\%$$

Medidas del rendimiento de un clasificador

Especificidad (Proporción de verdaderos negativos - PVN)

Es la proporción de objetos negativos que son clasificados correctamente. Es decir, la proporción de aciertos dentro del conjunto total de objetos que son realmente negativos:

$$Especificidad = PVN = \frac{VN}{VN + FP} = \frac{53272}{53272 + 1307} = 0.98 = 98\%$$

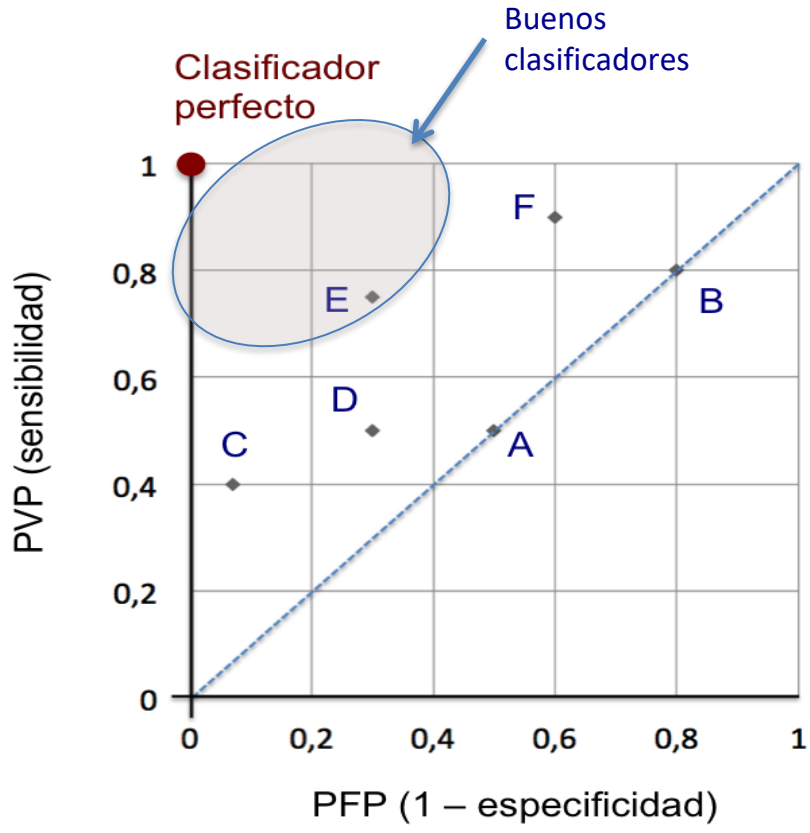
Proporción de falsos positivos (PFP)

Es la proporción de objetos negativos que son clasificados incorrectamente, es decir como si fueran positivos. Este valor es complementario con la especificidad y se calcula en la forma

$$PFP = 1 - Especificidad = \frac{FP}{VN + FP} = \frac{1307}{53272 + 1307} = 0.02 = 2\%$$

COORDENADAS ROC *(Receiving Operating Characteristic)*

Cada punto corresponde al resultado de un clasificador aplicado al mismo conjunto de objetos

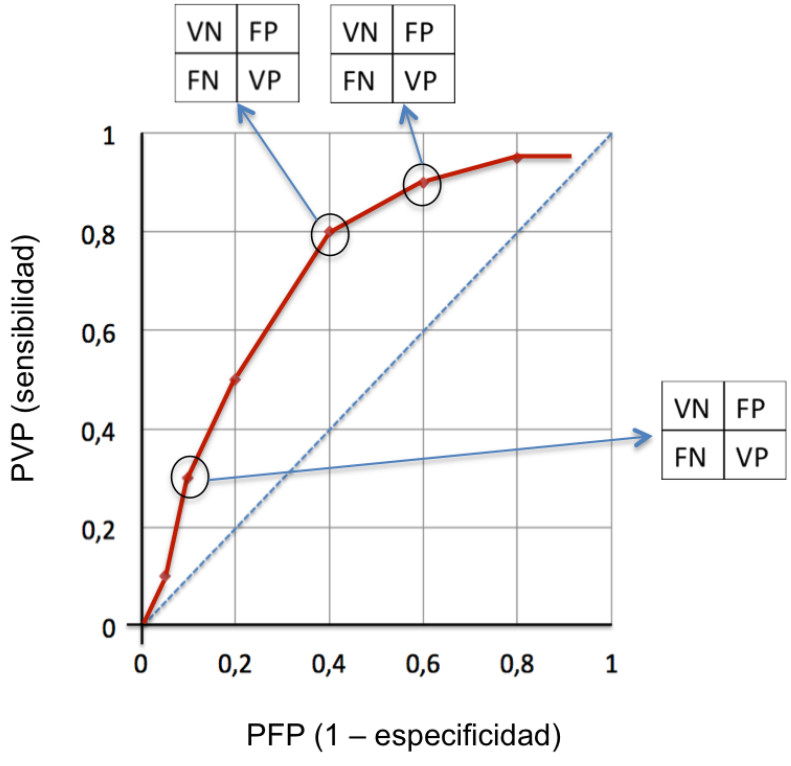


CURVA ROC *(Receiving Operating Characteristic)*

Cada punto se obtiene a partir de la matriz de confusión para un **valor umbral** que se va variando gradualmente.

El clasificador **toma una decisión** dependiendo del resultado del cálculo de una función y de un **valor umbral** prefijado por el diseñador.

Un objeto se clasifica como **positivo** si el valor del cálculo es **superior** al umbral, y negativo en caso contrario.

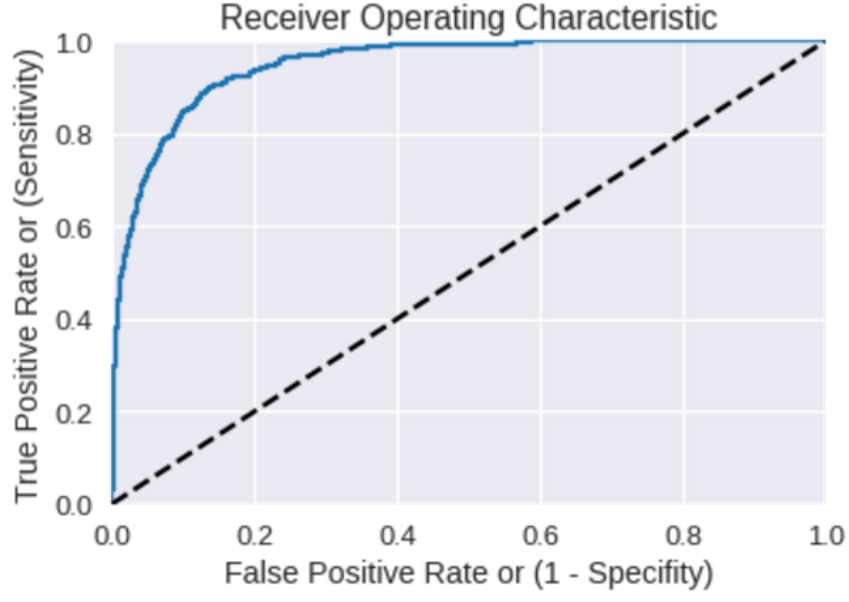


INDICADOR: Área bajo la ROC

Mejor cuanto más próximo a 1

Curva ROC

```
1  from sklearn.metrics import roc_curve
2
3  # calcula fpr, tpr, thresholds
4  fpr, tpr, thresholds = roc_curve(y, y_prob[:,1], pos_label='Yes')
5
6  # Grafica ROC curve
7  plt.plot(fpr, tpr)
8  plt.plot([0, 1], [0, 1], 'k--')# curva de predicciones aleatorias
9  plt.xlim([0.0, 1.0])
10 plt.ylim([0.0, 1.0])
11 plt.xlabel('False Positive Rate or (1 - Specifity)')
12 plt.ylabel('True Positive Rate or (Sensitivity)')
13 plt.title('Receiver Operating Characteristic')
14 plt.show()
```



BOOTSTRAPING

Bootstrap

En un **contexto estadístico** general, este término se refiere a métodos que usan muestras aleatorias para estimar propiedades de una población o conjunto mayor, cuantificando su incertidumbre (variabilidad).

En el **aprendizaje estadístico,** se refiere a la selección aleatoria de una serie de conjuntos de training y validación según la siguiente estrategia:

- A partir del conjunto inicial con n observaciones, se obtiene un número L de nuevos conjuntos de entrenamiento seleccionando n observaciones con remplazo, es decir que algunas estarán repetidas.

- Se realizan entrenamientos con cada uno de los L conjuntos y se promedian los resultados.

El nombre viene de la frase *"To pull oneself up by one's bootstraps",* usada coloquialmente para expresar la idea de que alguien resuelve un problema por sus propios medios, "espabilándose".

Bootstrap

Ejemplo ilustrativo

Se quiere invertir una cantidad fija de dinero en dos activos X e Y.
Se establece la siguiente fórmula:

$$I = \alpha X + (1 - \alpha)Y$$

Como X e Y son variables aleatorias, la inversión I está sujeta a variabilidad. Se quiere determinar la proporción 'alfa' que minimice la varianza, que es una medida del riesgo:

$$Var(\alpha X + (1 - \alpha)Y)$$

Se puede comprobar que la mínima varianza se obtiene para

$$\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}}$$

Var (X) Var (Y) Cov (X,Y)

Bootstrap

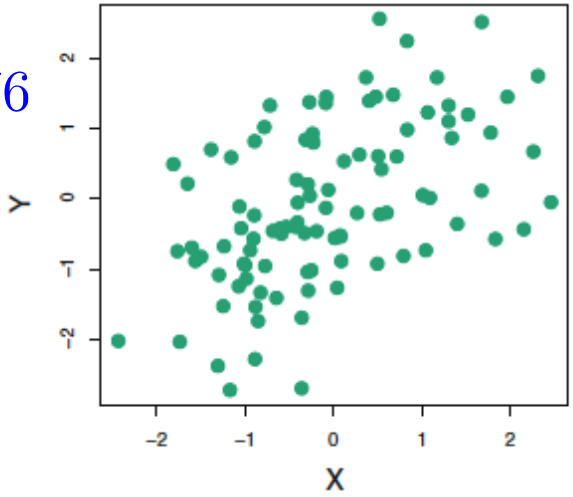
Ejemplo ilustrativo

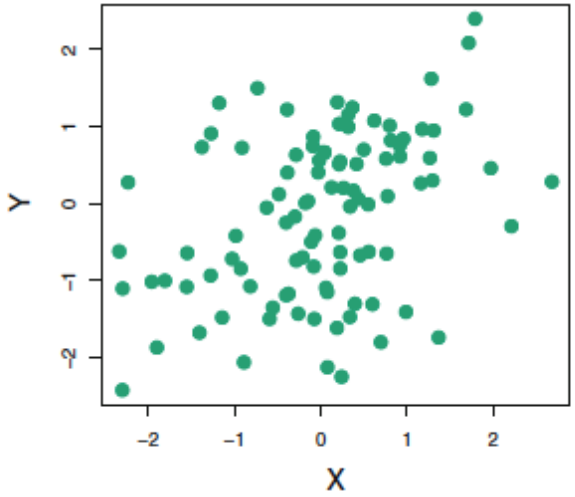
Estas cantidades son desconocidas. Se realizan **estimaciones** a partir de muestras y con ellas se estima el valor deseado de 'alfa':

$$\hat{\alpha} = \frac{\hat{\sigma}_Y^2 - \hat{\sigma}_{XY}}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2\hat{\sigma}_{XY}}$$

Para evaluar la exactitud de la estimación, se consideran 1000 muestras de 100 pares de observaciones (X, Y) y se obtienen 1000 estimaciones usando la fórmula anterior: $\hat{\alpha}_1, \hat{\alpha}_2, \ldots, \hat{\alpha}_{1,000}$

$\hat{\alpha} = 0.576$





$\hat{\alpha} = 0.532$

Bootstrap

Ejemplo ilustrativo

Se calcula la media de las 1000 estimaciones:

$$\bar{\alpha} = \frac{1}{1,000} \sum_{r=1}^{1,000} \hat{\alpha}_r = 0.5996$$

Su desviación estándar es

$$\sqrt{\frac{1}{1,000-1} \sum_{r=1}^{1,000} (\hat{\alpha}_r - \bar{\alpha})^2} = 0.083$$

Este ejemplo se ha obtenido mediante simulaciones con los valores

$$\sigma_X^2 = 1, \sigma_Y^2 = 1.25, \sigma_{XY} = 0.5 \longrightarrow \alpha = 0.6$$

Por tanto, la estimación 0.5996 es muy buena.

En la práctica este procedimiento no puede implementarse: no pueden generarse nuevas muestras de la población original.

Bootstrap

Solución alternativa

El bootstrap es un procedimiento computacional para **emular** el proceso de obtención de nuevos conjuntos de muestras (artificiales), de forma que puede estimarse la variabilidad de α sin necesitar muestras adicionales verdaderas.

En lugar de obtener repetidamente nuevos conjuntos independientes de datos de la población, **obtenemos distintos conjuntos muestreando de forma repetida observaciones del conjunto de datos original**.

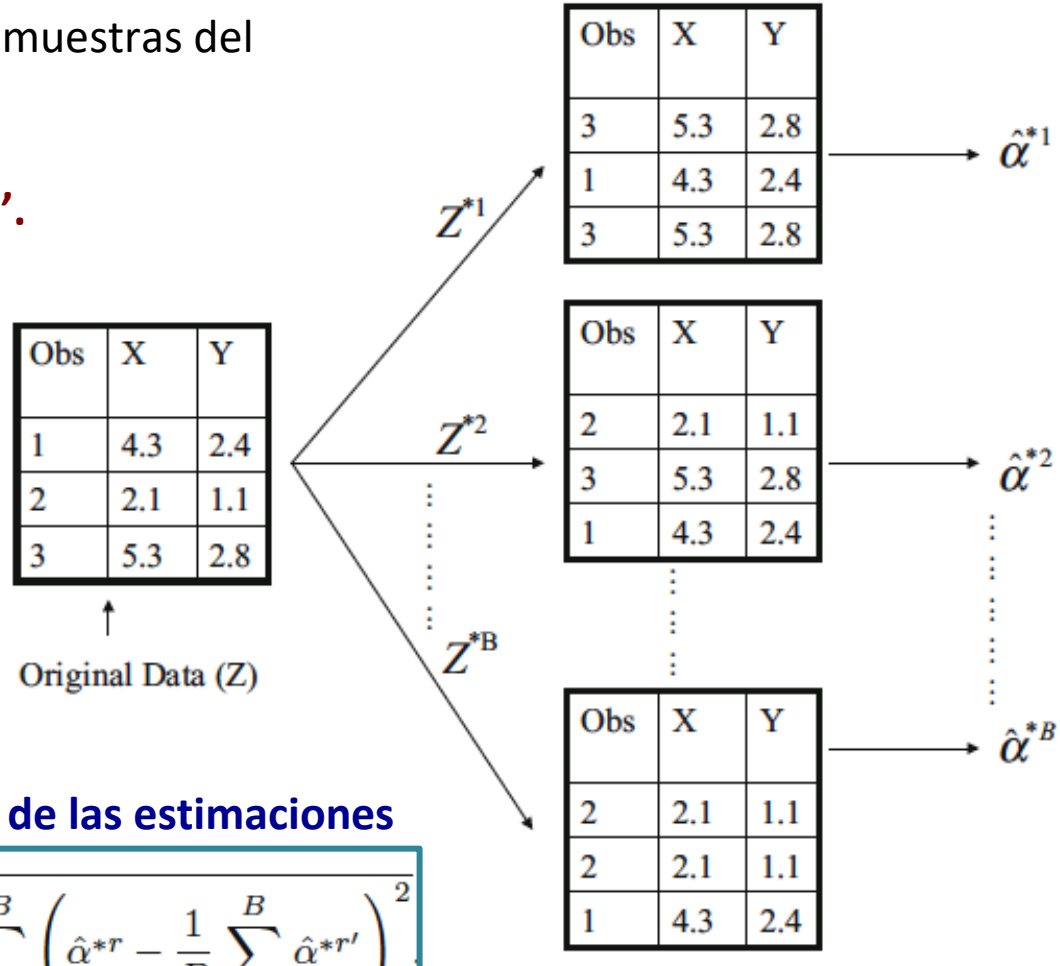
Bootstrap

Ejemplo de un conjunto de datos con n=3

Se seleccionan aleatoriamente n muestras del conjunto inicial con **remplazo**.

Se obtiene la estimación de 'alfa'.

El proceso se repite hasta un número B de veces



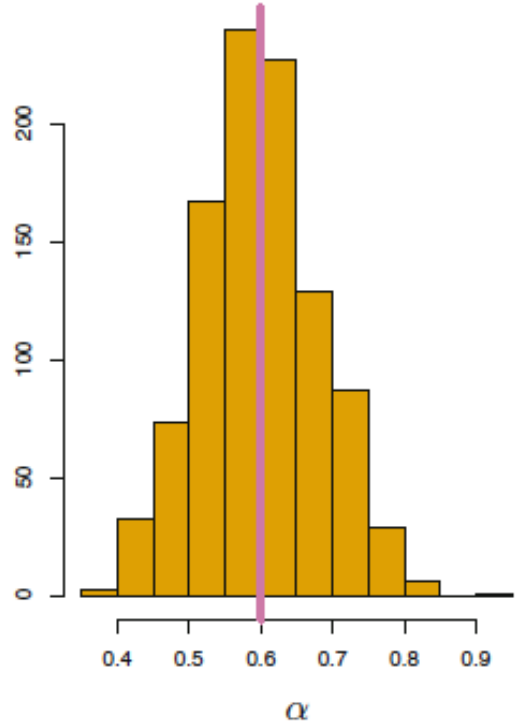
Se calcula la desviación estándar de las estimaciones

$$\mathrm{SE}_B(\hat{\alpha}) = \sqrt{\frac{1}{B-1}\sum_{r=1}^{B}\left(\hat{\alpha}^{*r} - \frac{1}{B}\sum_{r'=1}^{B}\hat{\alpha}^{*r'}\right)^2}$$

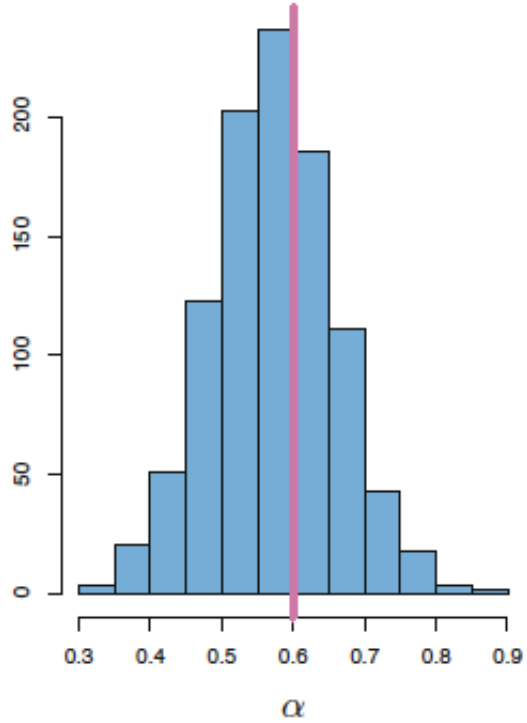
Bootstrap

Ejemplo ilustrativo

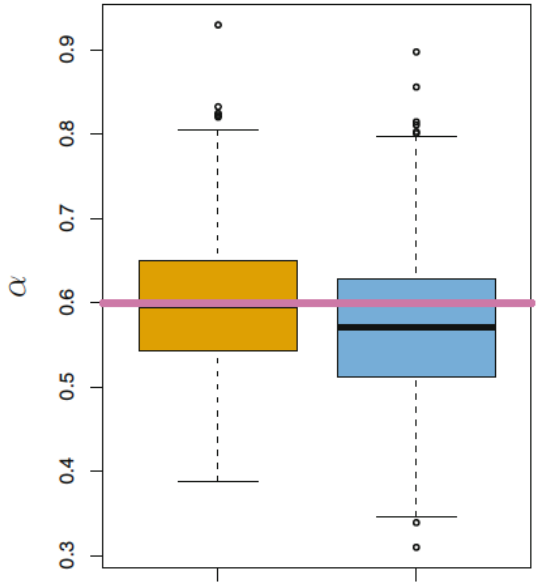
Histograma de las estimaciones de ‘alfa’ con 1000 datos simulados *(verdad)*



Histograma de las estimaciones de ‘alfa’ con 1000 muestras bootstrap



Comparación mediante boxplots



Resultados muy parecidos

¿Y en el mundo real?

- El procedimiento de tomar muchas muestras de la población no puede ser aplicado en la vida real.

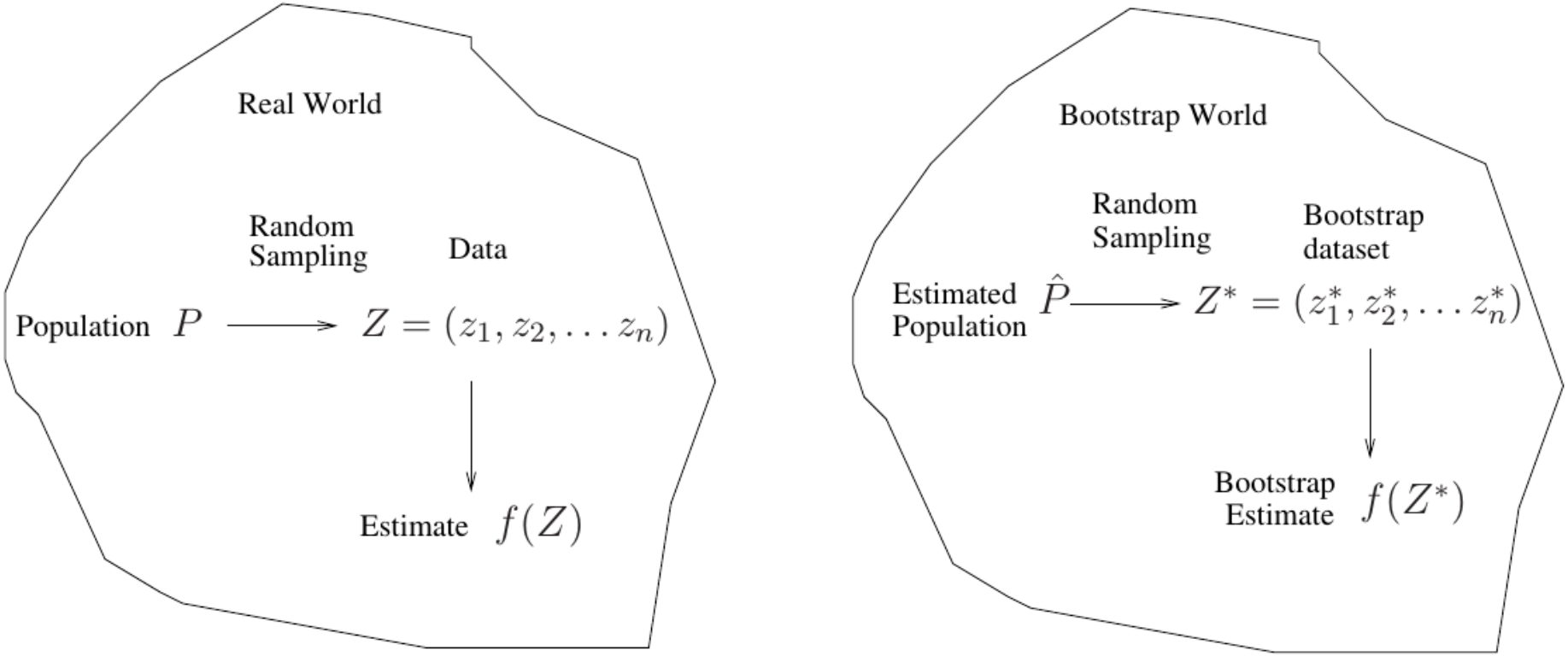
- Si embargo, el método de *bootstrap* nos permite imitar el proceso de obtener nuevos conjuntos de datos, tal que podamos estimar las variabilidad de nuestra estimación sin generar muestras adicionales.

¿Y en el mundo real?

- En vez de obtener repetidamente nuevos conjuntos de datos independientes a partir de la población, obtenemos distintos conjuntos de datos mediante repetidos muestreos **con reemplazo** partiendo del conjunto de datos original.

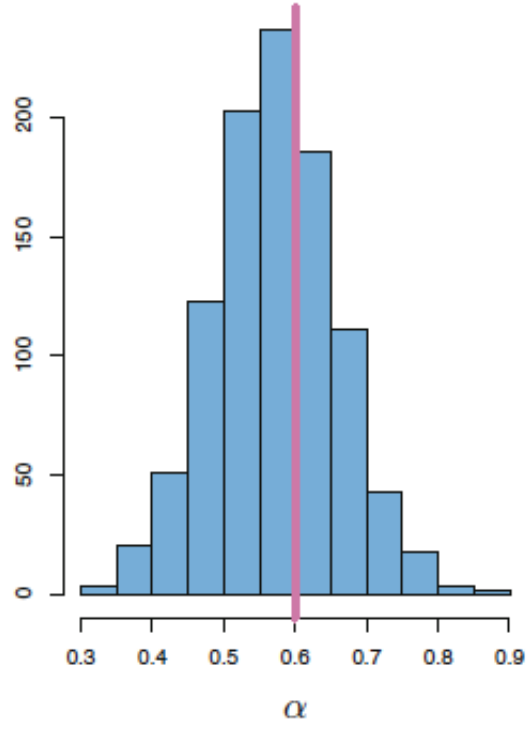
- Cada uno de esos conjunto de datos “bootstrap” es creado mediante el muestreo **con reemplazo**, y es del mismo tamaño que el conjunto de datos original. Como resultado, las observaciones pueden aparecer más de una vez en un conjunto de datos “bootstrap” y en algunos puede no aparecer.

Bootstrap en una figura



Otros usos del bootstrap

- Es usado principalmente para estimar errores estándar.
- También puede suministrar de forma aproximada intervalos de confianza para un parámetro poblacional.



- Por ejemplo, mirando en el histograma, los cuantiles del 5% y 95% de los 1000 valores son (0.43, 0.72), respectivamente.

- Esto representa un intervalo de confianza aproximado del 90% para el verdadero α.

¿Puede el bootstrap estimar el error de predicción?

- En validación cruzada, cada uno de los K conjuntos de validación es distinto de los otros K-1 conjuntos usados para el entrenamiento. Entonces, no hay superposición.
- Para estimar el error de predicción usando bootstrap, podriamos usar cada conjunto de datos bootstrap como el conjunto de entrenamiento, y la muestra original como el conjunto de validación.
- Pero cada muestra bootstrap se superpone significativamente con los datos originales. Cerca de las 2/3 partes de los puntos de datos originales aparecen en cada muestra bootstrap.
- Esto causa que el bootstrap subestime seriamente el error de predicción verdadero.
- La otra forma, usando los datos originales como entrenamiento, y la muestra boostrap como validación, es incluso peor.