

Prácticas de Aprendizaje Automático

Práctica 1

Experimentación con clasificadores y regresores

Pablo Mesejo y Salvador García

Universidad de Granada

Departamento de Ciencias de la Computación e Inteligencia Artificial



UNIVERSIDAD
DE GRANADA



Práctica 1

- Ejercicio 1: el problema del sobreajuste (2/10)
- Ejercicio 2: problema de clasificación (4/10)
- Ejercicio 3: problema de regresión (4/10)
- **Casuística próxima a la realidad:** te llega un problema y... ¿cómo lo resuelves?
 - **Análisis del Problema, Preprocesado de los Datos, Entrenamiento, Validación, y Discusión de Resultados**

Repaso de algunos fundamentos teóricos

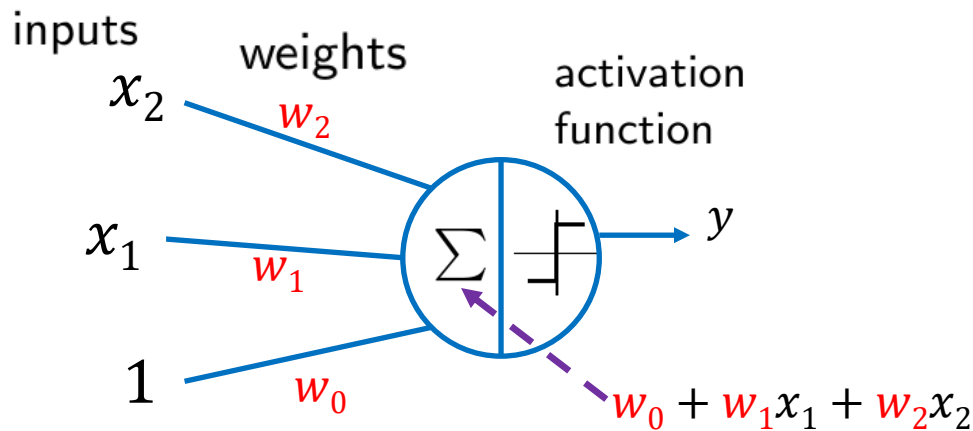
- ¿Qué aprendemos? ¡Pesos!

- Ejemplos:

- *Linear regression*

$$y = w_0 + w_1x_1 + w_2x_2$$

- *Perceptron*



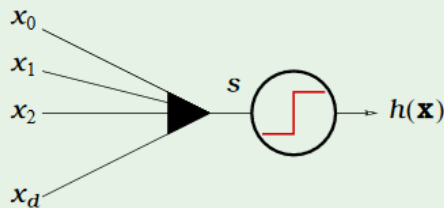
Los modelos de aprendizaje automático pueden ser muy diferentes. Por ejemplo, *k-nearest neighbors* (KNN) no tiene, como tal, un proceso de entrenamiento/aprendizaje de pesos (aunque, en cualquier caso, dados ciertos datos, permite realizar predicciones). El único "training" que ocurre en KNN es la memorización de los datos (creación de copia local), de forma que puedas realizar búsqueda y voto mayoritario.

Repaso de algunos fundamentos teóricos

$$s = \sum_{i=0}^d w_i x_i$$

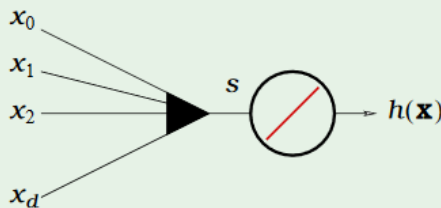
linear classification

$$h(\mathbf{x}) = \text{sign}(s)$$



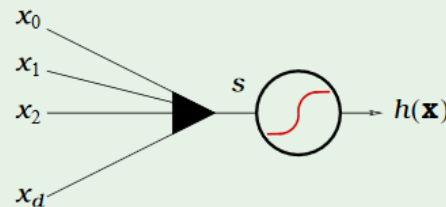
linear regression

$$h(\mathbf{x}) = s$$



logistic regression

$$h(\mathbf{x}) = \theta(s)$$



Repaso de algunos fundamentos teóricos

- ¿Cómo aprendemos?

- Generalmente, empleando alguna versión del **descenso de gradiente**

Tus pesos en la siguiente iteración

Learning rate



$$w_j := w_j - \eta \frac{\partial E_{in}(\mathbf{w})}{\partial w_j}$$

Derivada de nuestra función de pérdida (E_{in}) con respecto a los pesos. Intentamos medir la contribución de cada peso al error cometido.

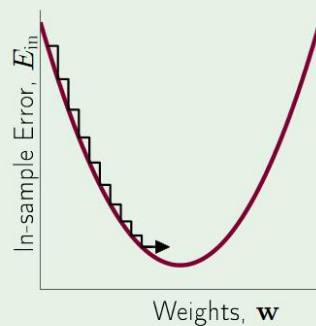
Tus pesos en la iteración actual

El entrenamiento/aprendizaje se plantea como un problema de optimización!!!

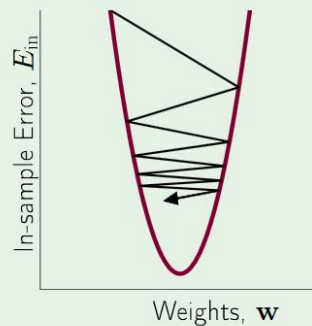
Repaso de algunos fundamentos teóricos

$$w_j := w_j - \eta \frac{\partial E_{in}(\mathbf{w})}{\partial w_j}$$

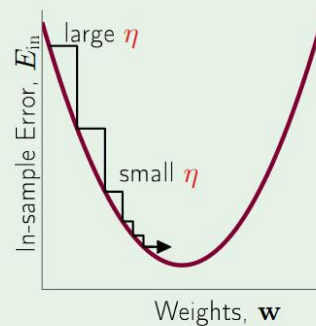
How η affects the algorithm:



η too small



η too large



variable η – just right

η should increase with the slope

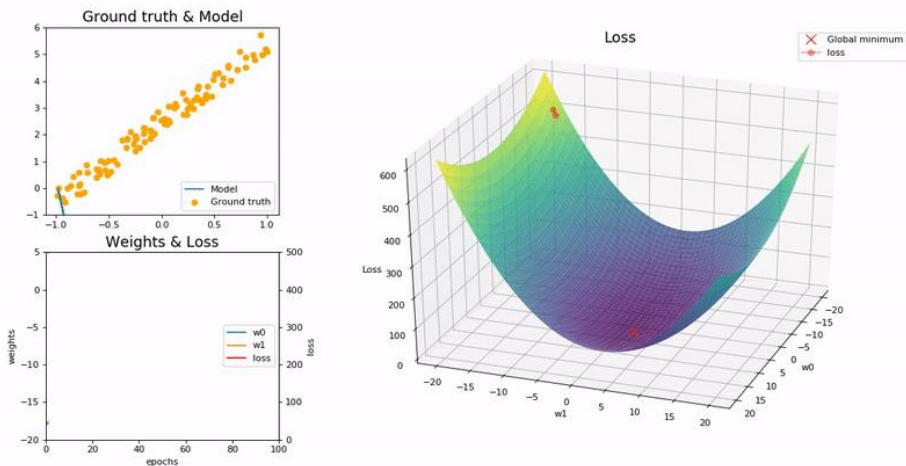
<http://work.caltech.edu/slides/slides09.pdf> (slide 21)

Repaso de algunos fundamentos teóricos

$$w_j := w_j - \eta \frac{\partial E_{in}(\mathbf{w})}{\partial w_j}$$

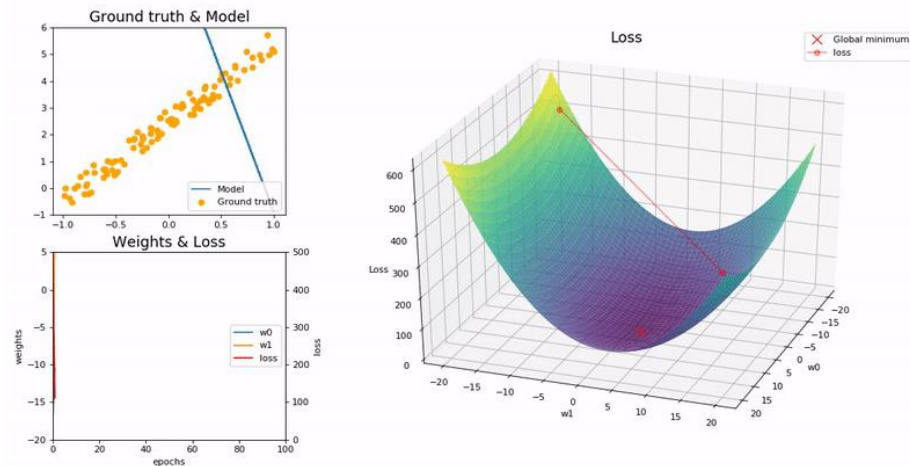
Learning rate adecuado: el modelo converge dentro de los *epochs* determinados.

lr: 0.01 - Epoch: 2/100



Learning rate “óptimo”: el modelo alcanza el óptimo rápidamente (menos de 10 *epochs*).

lr: 0.7 - Epoch: 2/100

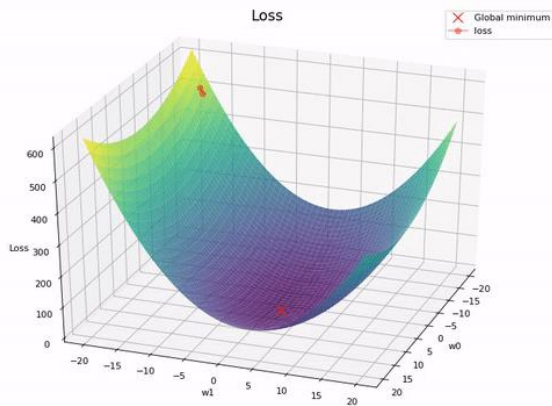
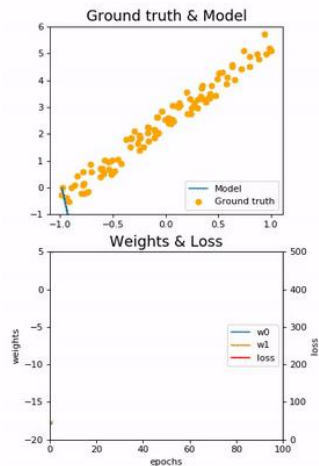


Repaso de algunos fundamentos teóricos

$$w_j := w_j - \eta \frac{\partial E_{in}(\mathbf{w})}{\partial w_j}$$

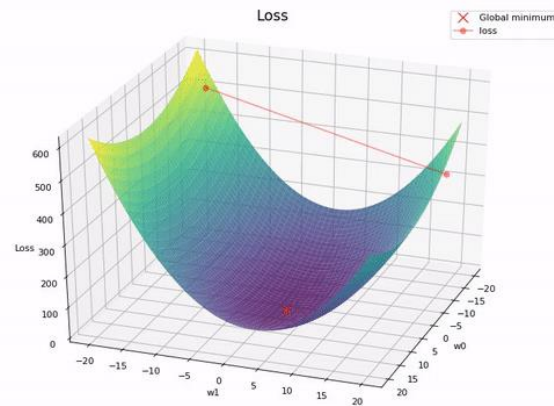
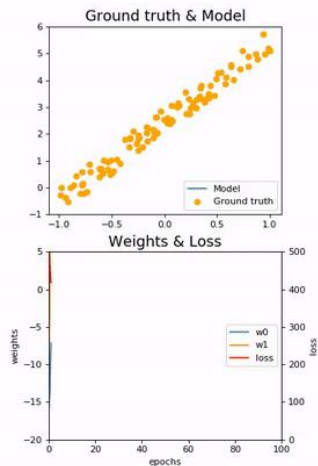
Learning rate demasiado bajo: la optimización progresa con demasiada lentitud.

lr: 0.01 - Epoch: 2/100



Learning rate demasiado alto: el modelo incluso llega a diverger.

lr: 1.01 - Epoch: 2/100



Repaso de algunos fundamentos teóricos

$$w_j := w_j - \eta \frac{\partial E_{in}(\mathbf{w})}{\partial w_j}$$

Función de pérdida

“The function we want to minimize or maximize is called the objective function or criterion. When we are minimizing it, we may also call it the cost function, loss function, or error function.” (“Deep Learning”, Ian J. Goodfellow, Yoshua Bengio and Aaron Courville, MIT Press, 2016 (p.82))

Regression (Mean Squared Error)

$$E = \frac{1}{N} \sum_{i=1}^N (\hat{y}_i - y_i)^2$$

(Binary) Classification (Cross-entropy loss)

$$E = -\frac{1}{N} \sum_{i=1}^N [y_i (\log(\hat{y}_i)) + (1 - y_i) \log(1 - \hat{y}_i)]$$

if $y = 1$ and $\hat{y} = 1 \rightarrow E = 0$

But as $\hat{y} \rightarrow 0, E \rightarrow \infty$

if $y = 0$ and $\hat{y} = 0 \rightarrow E = 0$

But as $\hat{y} \rightarrow 1, E \rightarrow \infty$



Dos de las más comunes!

Repaso de algunos fundamentos teóricos



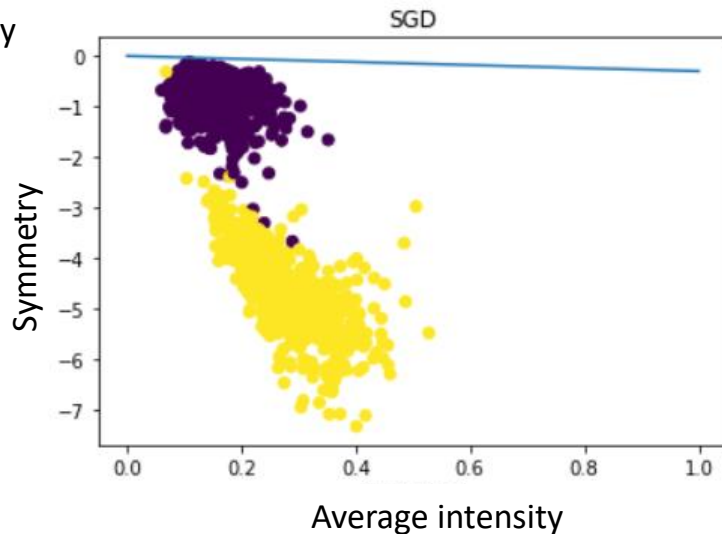
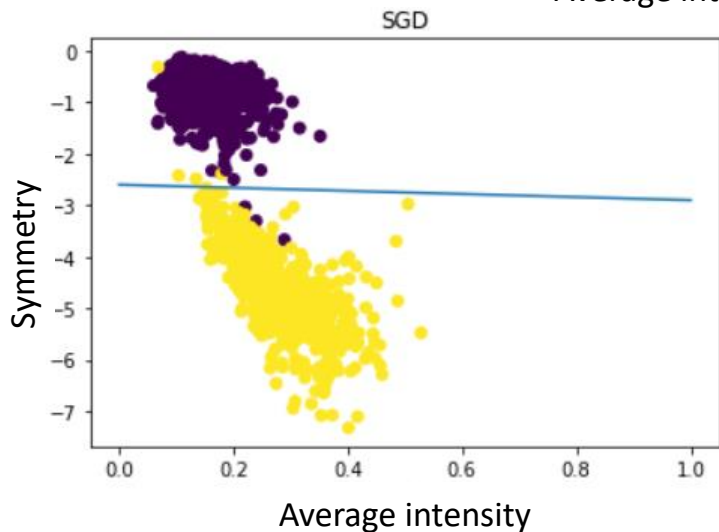
bias o intercept

$$y = w_0 + w_1 x_1 + w_2 x_2$$

Average intensity

Symmetry

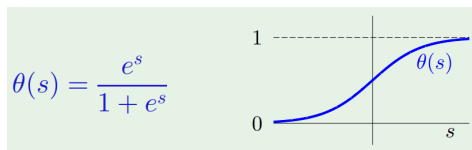
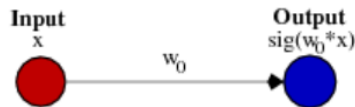
¿Qué es el bias y para qué sirve?



Sin él, perdemos flexibilidad en nuestro modelo: nuestro *decision boundary* se ve forzado a pasar por el origen (0,0)

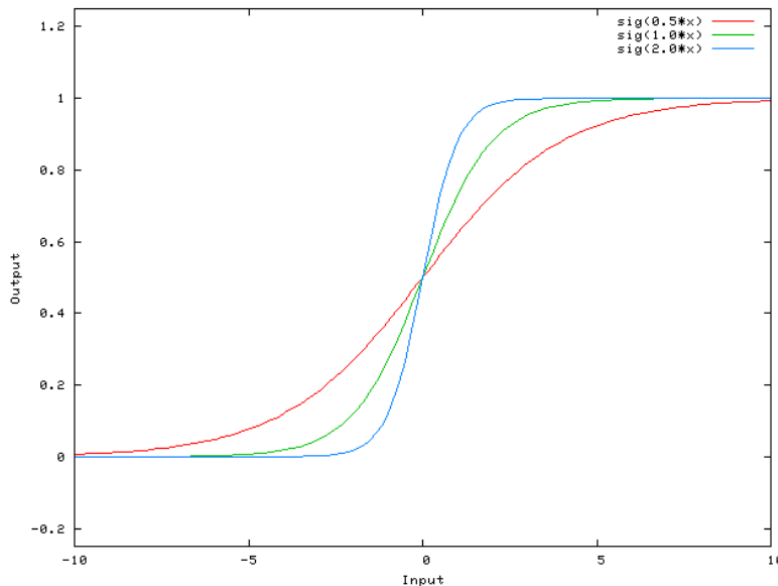
Repaso de algunos fundamentos teóricos

¿Qué ocurre si, en regresión logística, no empleamos bias?



En este ejemplo, el bias sería w_1

$$y = w_0 x$$

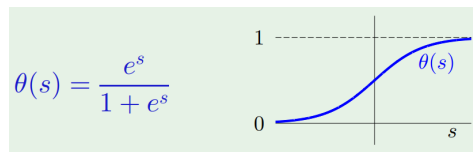
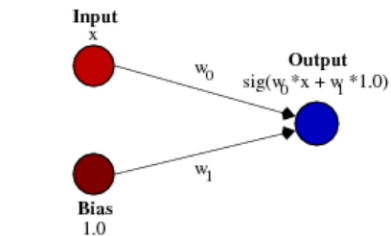


Sin bias, w_0 solo modula la pendiente de la sigmoide. ¿Cómo podríamos desplazar la curva en el eje X? Sin bias, no podemos...



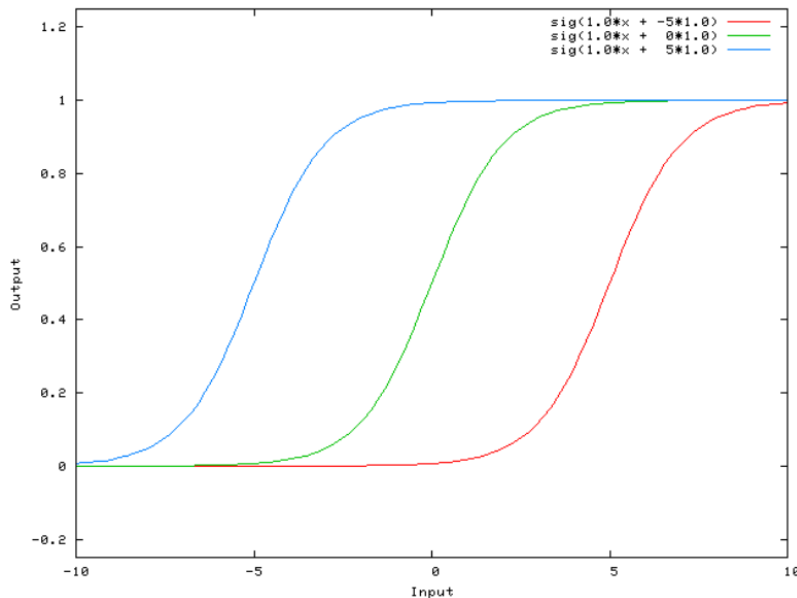
Repaso de algunos fundamentos teóricos

¿Qué ocurre si, en regresión logística, no empleamos bias?




En este ejemplo, el bias sería w_1

$$y = w_1 + w_0 x$$



El bias enriquece y proporciona flexibilidad en relación al número de funciones que podemos aprender.

Repaso de algunos fundamentos teóricos

- ¿De qué hablamos cuando hablamos de modelos lineales? ¿Lineales en qué?
 - ¡En los pesos! 
 - $w_0 + w_1x_1 + w_2x_2^2 = y$ es un modelo lineal, ¡a pesar de que hay entradas al cuadrado! Lo que nos interesa es que los pesos no estén al cuadrado.

Repaso de algunos fundamentos teóricos

- Que un modelo sea lineal no está necesariamente ligado al hecho de que la frontera de decisión sea una línea recta.

Si transformamos nuestras características de entrada, podemos resolver un problema no linealmente separable con un modelo lineal (regresión lineal, p.ej.).

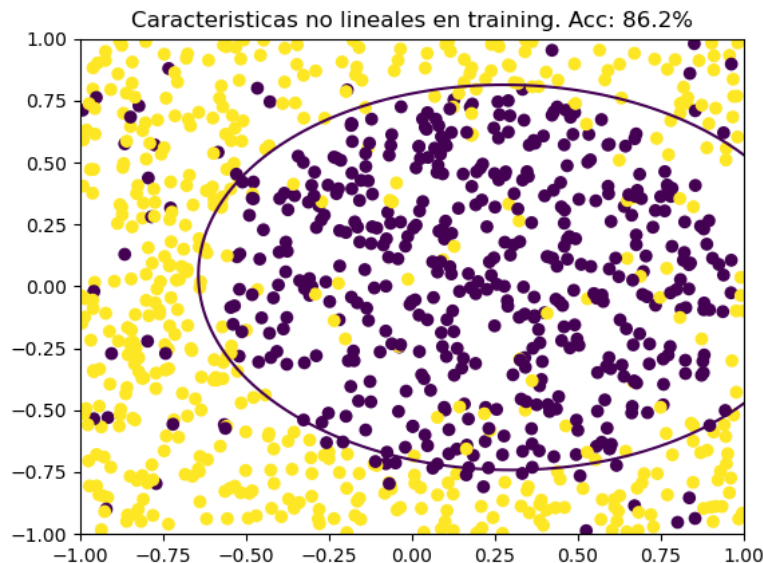


En el fondo, lo único que estaríamos haciendo es

$$y = w_0 + w_1x_1 + w_2x_2 + w_3x_1x_2 + w_4x_1^2 + w_5x_2^2$$

en lugar de

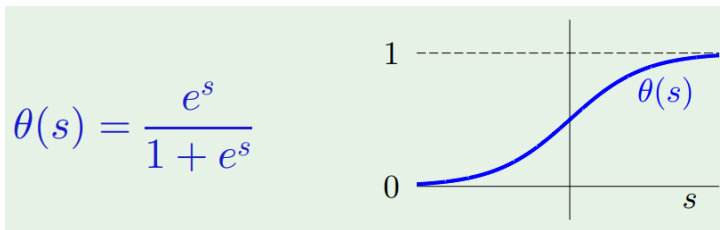
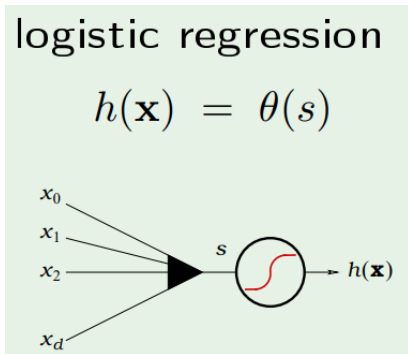
$$y = w_0 + w_1x_1 + w_2x_2$$



Una frontera de decisión no lineal puede ser producida por un modelo lineal.

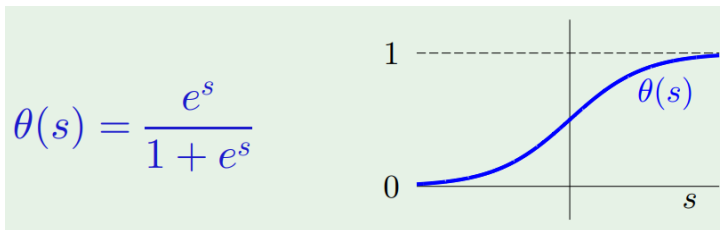
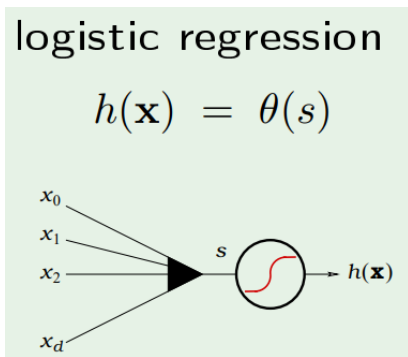
Repaso de algunos fundamentos teóricos

- ¿Qué no sería un modelo lineal?
 - Por ejemplo, $y = w_0 \cdot x^{w_1}$
- ¿Cómo es posible que regresión logística sea un modelo lineal si usa una función de activación no lineal?



Repaso de algunos fundamentos teóricos

- ¿Cómo es posible que regresión logística sea un modelo lineal si usa una función de activación no lineal?



Porque podemos escribir las predicciones en términos de una función lineal

$$h(x) = \frac{1}{1+e^{-\mathbf{w}^T \mathbf{x}}} = 0.5 \rightarrow 2 = 1 + e^{-\mathbf{w}^T \mathbf{x}} \rightarrow 1 = e^{-\mathbf{w}^T \mathbf{x}} \rightarrow \ln(1) = \ln(e^{-\mathbf{w}^T \mathbf{x}}) \rightarrow 0 = -\mathbf{w}^T \mathbf{x}$$

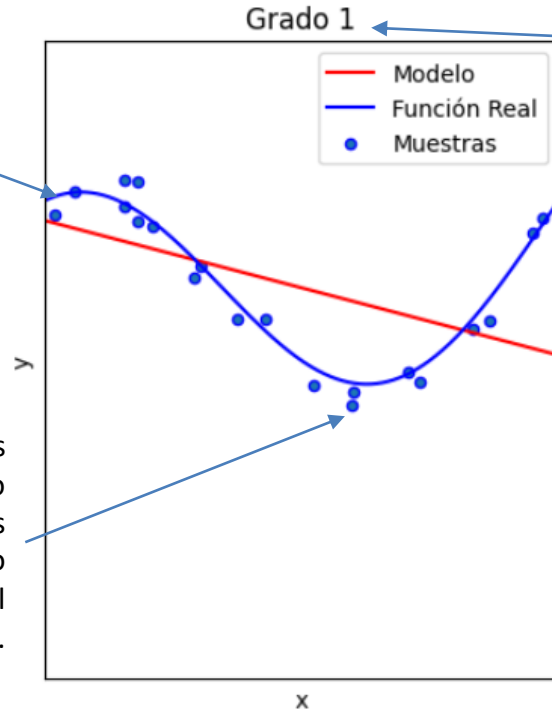
La salida siempre depende de la suma de los productos de entradas por pesos. No hay ninguna interacción del estilo $w_1 x_1 \cdot w_2 x_2$, que haría nuestro modelo no lineal.

Ejercicio 1

- Apartado A: **regresión en 1D** con datos sintéticos (1 punto)

Generamos una senoide que representa nuestra señal ideal (*ground truth*). Función que buscamos aproximar.

A partir de dicha *ground truth* generamos puntos ruidosos. Estos puntos serán nuestro conjunto de datos, sobre el que aplicaremos validación cruzada (`cross_val_score` o `cross_validate`) y MSE para estimar el rendimiento de nuestro modelo.

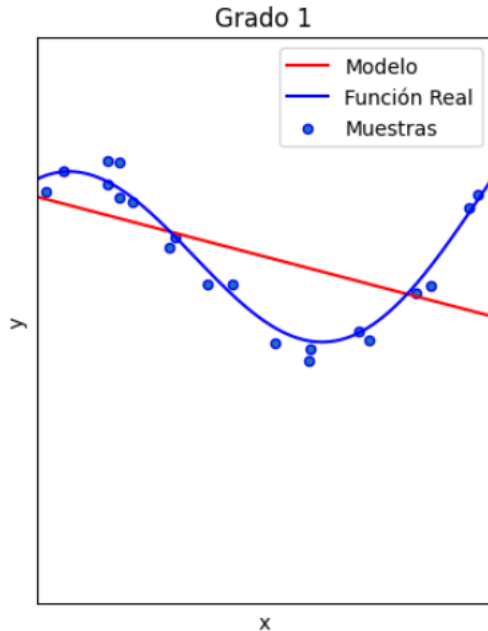


Emplear regresión lineal (`LinearRegression()`) con distintas transformaciones polinomiales de las características de entrada (`PolynomialFeatures()`): grado 1, 2, 4, 8, 16 y 32.

Nuestro modelo de regresión lineal ajustado a los datos.

Ejercicio 1

- Apartado A: regresión en 1D con datos sintéticos (1 punto)



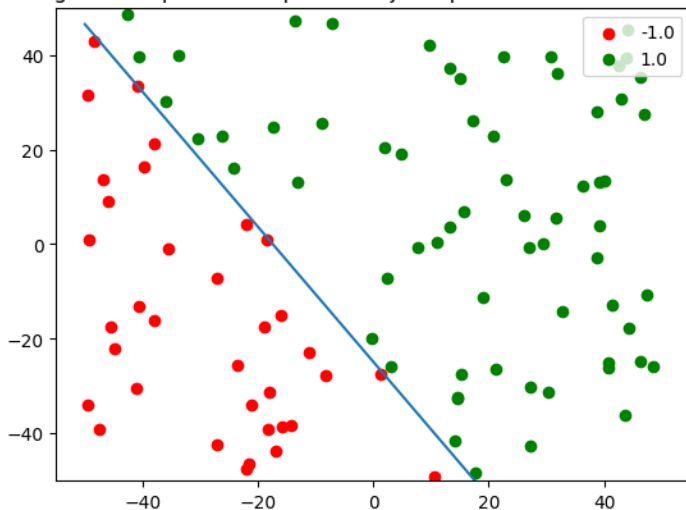
Tenéis que:

- mostrar figuras con ajustes, como la de la izquierda, para cada grado de polinomio solicitado.
- crear una tabla con grado de polinomio, número de pesos del modelo, MSE train, MSE test, y media de los coeficientes encontrados en entrenamiento (en valor absoluto).
- responder una serie de preguntas:
 - ¿Se observa algún patrón en los resultados obtenidos?
 - ¿Se observa algún fenómeno de infraentrenamiento (*underfitting*) o sobreentrenamiento (*overfitting*)?
 - ¿Qué grado considera el más adecuado para resolver este problema? ¿Por qué?

Ejercicio 1

- Apartado B: **clasificación en 2D** con datos sintéticos (1 punto)

Datos y recta generada para su etiquetado. Ajuste perfecto. Misclassification rate: 0.0%

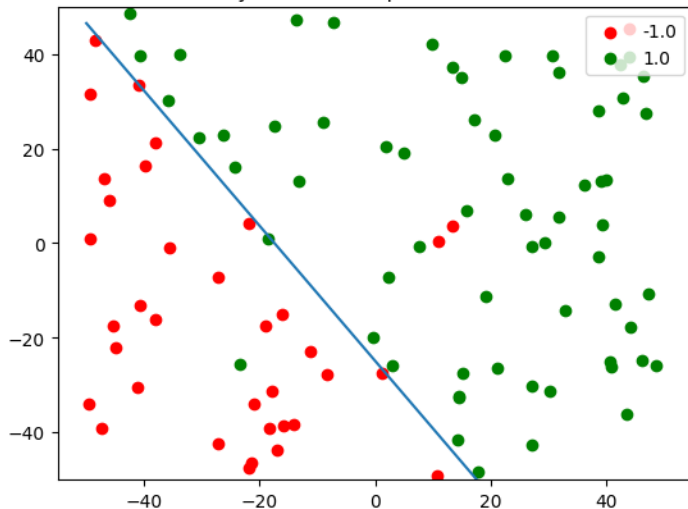


Generamos una recta (frontera de decisión) que emplearemos para etiquetar 100 puntos 2D. Esa recta será nuestra *ground truth*. La función ideal que queremos aproximar.

Ejercicio 1

- Apartado B: clasificación en 2D con datos sintéticos (1 punto)

Datos (con 5% de ruido/clase) y recta de etiquetado inicial. Misclassification rate: 5.0%

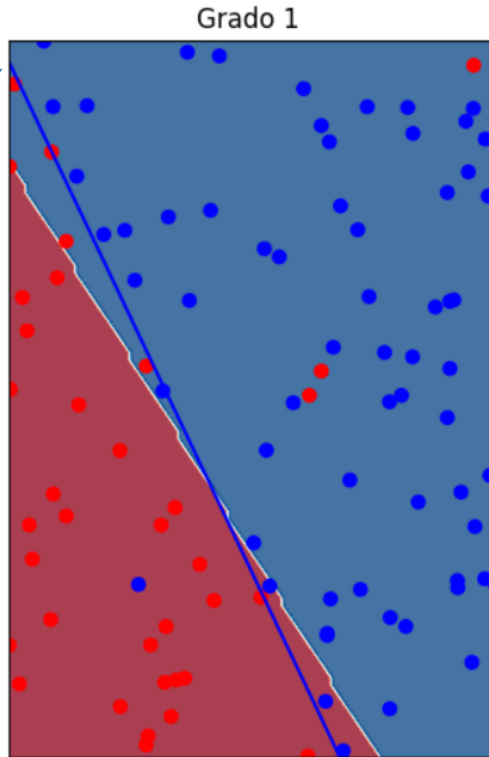


Introducimos ruido en las etiquetas.

Ejercicio 1

- Apartado B: clasificación en 2D con datos sintéticos (1 punto)

Indicamos en azul la función etiquetadora real/ideal a aproximar.



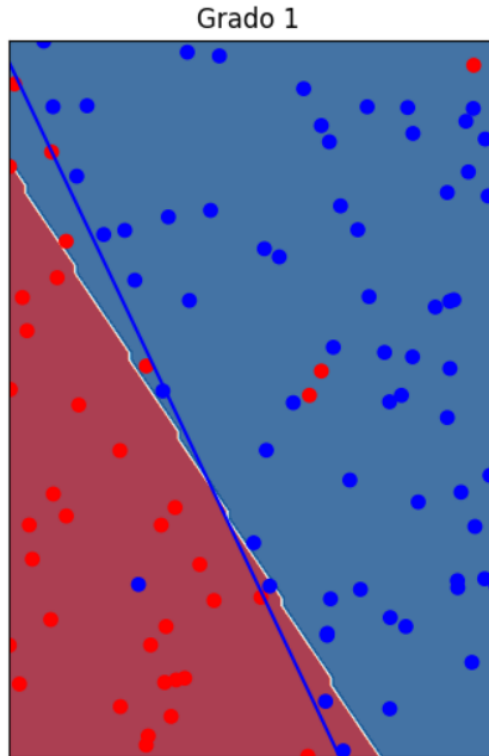
Empleamos SVM con kernel lineal (`sklearn.svm.SVC`) para ajustar estos datos, tras escalarlos (`sklearn.preprocessing.StandardScaler`) y aplicar transformaciones polinomiales (`PolynomialFeatures`) con distintos grados (1, 2, 4 y 8)

Visualizamos la frontera de decisión aprendida, con cada grado de polinomio, empleando funciones como `contourf`, `ListedColormap` o `DecisionBoundaryDisplay`.

Véase https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

Ejercicio 1

- Apartado B: clasificación en 2D con datos sintéticos (1 punto)



Generamos un conjunto de test siguiendo, exactamente, el mismo proceso empleado para generar nuestros datos de entrenamiento

Hipótesis de partida:

nuestros datos de entrenamiento siguen una distribución similar a los de test.

Creamos una tabla con el grado de polinomio, el número de pesos de nuestro modelo, y métricas de error (*accuracy* y *f1-score*) en entrenamiento y test.

¿Qué conclusiones se pueden extraer al respecto de los resultados obtenidos y el trabajo realizado?



Ejercicios 2 y 3



Datos tabulares:

– Problema de clasificación

<https://archive.ics.uci.edu/dataset/80/optical+recognition+of+handwritten+digits>

5620 ejemplos, 64 atributos, y 10 clases de salida

– Problema de regresión

<https://archive.ics.uci.edu/dataset/203/yearpredictionmsd>

515345 ejemplos, 90 atributos, y salida a predecir (año de la canción, 89 posibilidades).



Nota: respetad las particiones de training y test indicadas en el repositorio (`optdigits.train` y `optdigits.test` en el problema de clasificación, por ejemplo)

1. Analizar y comprender el problema

- ¿Qué es X?
- ¿Qué es Y?
- ¿En qué consiste el problema que tengo que resolver ($f: X \rightarrow Y$)?

¡Ojo! Hay una **delgada línea que separa la visualización/exploración de los datos del data snooping/data dredging**.

- a) Está bien visualizar los datos (de entrenamiento) si es para aprender sobre el problema. Está mal visualizarlos si es para guiar nuestra elección del modelo concreto a usar.
- b) Nuestro propósito no es tanto establecer una hipótesis (que se ajuste perfectamente a nuestro conjunto de entrenamiento) como comprender mejor nuestro problema.

1. Analizar y comprender el problema

— **Data Snooping**: una de las claves principales es **no usar los datos de test para absolutamente nada**.

- Ejemplo 1:

- Alguien calcula la media y desviación típica para realizar la estandarización de los datos utilizando todos los ejemplos (incluyendo, sin darse cuenta, el test, que no deberíamos conocer de nada).

- » Primero habría que dividir los datos, luego escalar los datos de entrenamiento y, finalmente, escalar los datos de test con los factores de escalado empleados en el entrenamiento.

- Ejemplo 2:

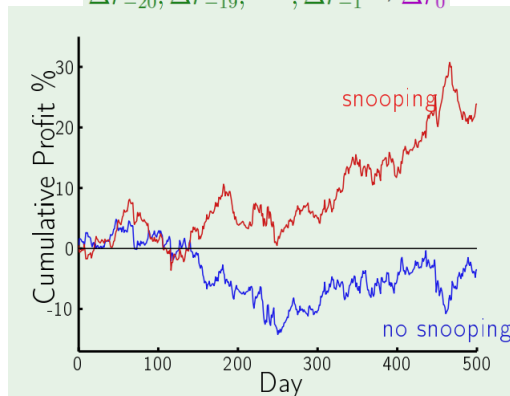
- nuestros datos, al visualizarlos, tenemos la impresión de que son linealmente separables y, por tanto, decidimos usar *perceptron learning algorithm* (PLA). Muy probablemente esa premisa sea demasiado fuerte, esté demasiado apegada a nuestros datos, y sea desaconsejable asumirla.

Predict US Dollar versus British Pound

Normalize data, split randomly: $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{test}}$

Train on $\mathcal{D}_{\text{train}}$ only, test g on $\mathcal{D}_{\text{test}}$

$\Delta r_{-20}, \Delta r_{-19}, \dots, \Delta r_{-1} \rightarrow \Delta r_0$



<https://home.work.caltech.edu/slides/slides17.pdf>

Lecture 17 - Three Learning Principles

<https://www.youtube.com/watch?v=EZBUDG12Nr0>

1. Analizar y comprender el problema

- **Data Snooping:** una de las claves principales es **no usar los datos de test para absolutamente nada.**
- Ejemplo 3:
 - Probamos todas las técnicas que conocemos, una tras otra, sobre el mismo conjunto de test, y nos quedamos con la mejor.
 - » Imaginemos que tenemos 3 hipótesis candidatas (una con SVM, otra con MLP, otra con RL), y decidimos cuál es la mejor mirando su rendimiento con $E_{test} \rightarrow$ *data snooping*! En principio, deberíamos ejecutar el test con una única hipótesis final!

1. Analizar y comprender el problema

- Entonces... ¿qué clase de **análisis** se puede hacer?
 - Descripción de las variables empleadas (¿cuántas entradas tenemos? ¿qué miden? ¿de cuántos ejemplos disponemos? ¿cómo se han obtenido?)
 - Histogramas con el porcentaje de ejemplos de cada clase en entrenamiento
 - Para verificar la existencia o no de desbalanceo en los datos
 - Tablas mostrando un resumen de las variables continuas
 - media, desviación típica, mínimo, máximo, porcentaje de valores faltantes, percentiles,...
 - Y categóricas
 - número de ejemplos para cada categoría, rango de los valores, porcentaje de valores faltantes, ...
 - Matrices de correlación de variables continuas (Pearson, Spearman)
 - etc.
- A nivel de visualización y análisis de los datos, **debéis centraros en cuestiones exploratorias y descriptivas siempre enfocadas a comprender mejor el problema.**
- Debéis dejar muy claro **qué ha aportado la visualización de variables.**



2. Preprocesado de los datos: manipulaciones sobre los datos iniciales para obtener el conjunto de entrenamiento.

No hay reglas universales. Cada problema y característica requiere un procesamiento diferente.
Aquí aportamos algunas referencias generales.

- eliminar datos sin variabilidad (no son discriminantes)
- eliminar datos extremos/atípicos (*outliers*)
- ¿Qué hacemos con los datos faltantes (*missing data imputation*)?
- reducción de dimensionalidad (p.ej. PCA)
 - El objetivo debe ser reducir el número de variables (es decir, la complejidad de tu problema) mientras se mantiene la información relevante de los datos originales.
- transformaciones en los datos
 - Ciertas transformaciones en los datos pueden permitir que un problema no linealmente separable se convierta en linealmente separable.

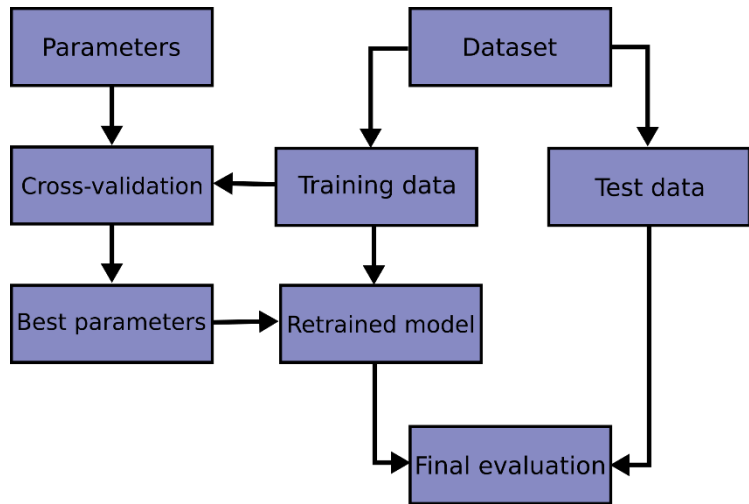
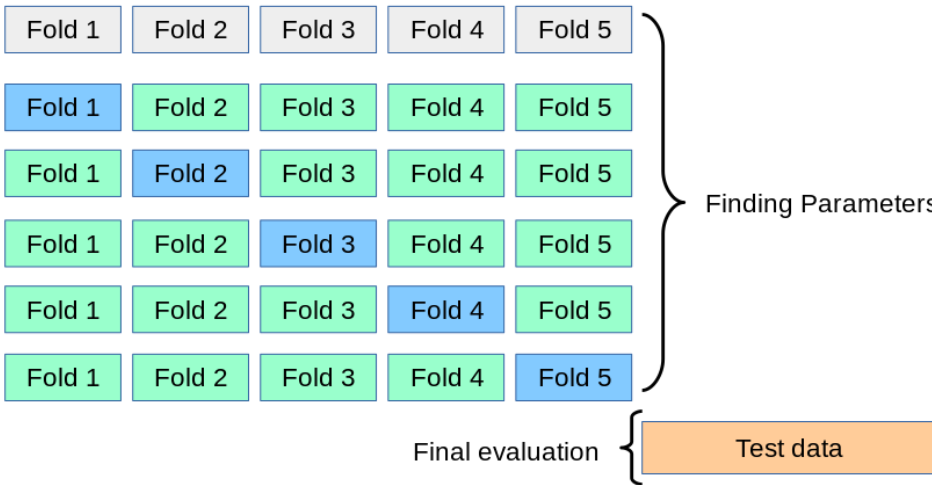
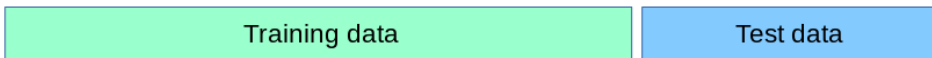
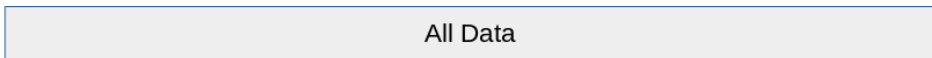
2. **Preprocesado de los datos:** manipulaciones sobre los datos iniciales para obtener el conjunto de entrenamiento.

No hay reglas universales. Cada problema y característica requiere un procesamiento diferente.
Aquí aportamos algunas referencias generales.

- escalado de variables
 - Empleada para que todas las variables estén en un rango de valores similar
 - Objetivo:
 - acelerar la optimización/entrenamiento (véase explicación de Andrew Ng en [Normalizing Inputs \(C2W1L09\)](#))
 - hay técnicas más afectadas por la escala de las variables (KNN, K-Means, SVM,...) que otras (DTs, RF,...)
 - ¿Con qué coeficientes se deben escalar los datos de test (con los calculados en test o con los calculados antes en entrenamiento)?
- codificación de datos
 - a) Si son binarias o numéricas → las podéis dejar como están (a no ser que tengáis algún motivo de peso para codificarlas/transformarlas de otro modo)
 - ¿Tiene sentido normalizar las variables binarias?
 - b) Si son variables categóricas:
 - Nominales → **one-hot encoding** (rojo – 1 0 0, verde – 0 1 0, azul – 0 0 1)
 - Ordinales → codificación entera (bajo – 1, medio – 2, alto – 3) o one-hot encoding
 - Cíclicas (días, meses, p.ej.) → cosine/sine transformation



3. Definición de los conjuntos de **training**, **validación** y **test**



https://scikit-learn.org/stable/modules/cross_validation.html

Algunas ideas sobre particiones de datos

- i. si tenemos muy pocos datos → *leave-one-out*
- ii. si tenemos modelos cuyo entrenamiento es muy costoso → *hold-out*
- iii. si se considera que los datos son relativamente escasos también se podría obviar la separación de un conjunto de test → empleo de **CV sobre todos los datos disponibles, aproximación de E_{out} con E_{cv}**
- iv. en el resto de casos, lo más frecuente es hacer lo mostrado anteriormente → separación de test y entrenamiento, empleo de **CV sobre el conjunto de entrenamiento, aproximación de E_{out} con E_{test}**

4. Modelos de Aprendizaje Automático a utilizar.

Discutir su idoneidad para el problema



5. Discusión y selección de valores para hiperparámetros

- a) Discutir la idoneidad de los valores de los parámetros de la técnica de ajuste. No podéis emplear los valores por defecto para los métodos incluidos en Scikit-learn sin saber qué hacen.
 - learning rate, tamaño de minibatch, criterio de parada, etc.
- b) L2 regularization /weight decay / ridge regression vs L1 regularization / Lasso
 - En problemas de alta dimensionalidad, podría tener sentido usar L1, porque la solución que proporciona es dispersa (aplica selección de características de modo implícito)
 - Si sabemos que todas las variables deben ser tenidas en cuenta, seguramente mejor L2

6. Entrenamiento, discusión de resultados, y estimación del error

- Entrenamiento (E_{in}) y Test (E_{test}). Estimación de E_{out}

- Posibilidades de análisis (entre otras):



- a) Emplear baselines con los que comparar.

- i. Por ejemplo, si tengo un 3% de E_{test} no sé si es mucho, porque a lo mejor un estimador *naive* (la media en regresión, o un clasificador aleatorio en clasificación) ya me da un 4% de error.

- b) Selección de las métricas de error adecuadas (MSE, MAE, Accuracy,...).

- i. No confundir métrica de error (para medir el rendimiento del modelo) y función de pérdida (para optimizar el modelo).

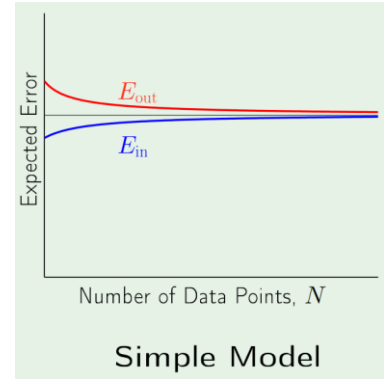
- ii. No confundir métricas de error para problemas de clasificación y regresión

- c) Introducción de curvas de aprendizaje para evaluar el rendimiento de los modelos a medida que son entrenados, progresivamente, con un mayor número de ejemplos:

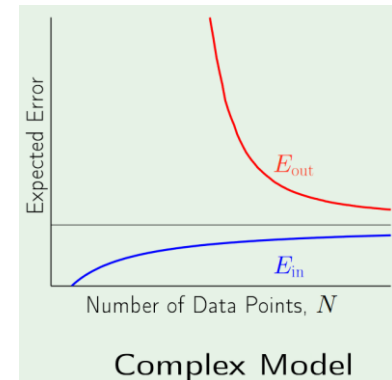
- https://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html
y <https://work.caltech.edu/library/082.html>

6. Entrenamiento, discusión de resultados, y estimación del error

- I. Del conjunto de entrenamiento separamos un conjunto para entrenamiento, propiamente dicho, y otro para validación.
- II. Escogemos, dentro del conjunto de entrenamiento, un conjunto de datos pequeño, entrenamos, guardamos ese error, le pasamos el conjunto de validación, y también guardamos el error de validación correspondiente.
- III. Así sucesivamente hasta que **hemos entrenado X veces con X conjuntos de entrenamiento progresivamente más grandes, y hemos validado X veces** (siempre con el mismo conjunto).
- IV. Visualizamos y analizamos ambas curvas.



<https://work.caltech.edu/library/082.html>



Nota: no confundir estas curvas con las de *early-stopping*, empleadas generalmente en redes neuronales para evitar el sobreentrenamiento, y en donde se emplea un conjunto de validación simultáneamente con el de entrenamiento.

Consejos generales

Presentad correctamente y describir con claridad el trabajo realizado:

- qué problema se aborda,
- detalles del proceso de validación cruzada,
- selección de hiperparámetros,
- regularización,
- valoración de los resultados finales con cada técnica, etc.

Más detalles importantes:

- la memoria no es el relato de ejecución del código
- no debéis emitir meras opiniones o presentar decisiones sin justificación
- evitad también **confundir conceptos básicos**
 - como podría ser considerar SGD como un modelo de aprendizaje automático, o confundir función de pérdida con métricas de evaluación, entre otros.
- no dudéis en **incluir todas las gráficas que consideréis pertinentes, y comentadlas en el cuerpo del texto** (dado que las gráficas no se comentan solas, y es necesario presentarlas y analizarlas)

Consejos generales (fuera del objetivo de la práctica)

Recordad que hay todo un abanico de modelos de aprendizaje:

- Qué método es más adecuado vendrá determinado por vuestro problema y datos concretos
- Revisad la literatura para ver qué se ha hecho antes en ese mismo problema o algún otro similar

Ejemplos (<https://www.asimovinstitute.org/neural-network-zoo/>):

- Muy pocos ejemplos, anotaciones muy costosas, baja dimensionalidad: SVM, RF, MLP, RL
- Imágenes: ConvNets / Transformers
- Etiquetado de secuencias (procesado de audio, series temporales): RNNs / Transformers
- Generación: GANs / Diffusion
- Información estructurada en forma de grafo: GNNs
- Información tabular: SVM, RF, MLP, Gradient Boosting
- Reducción de dimensionalidad o eliminación de ruido: Autoencoders

Entrega

.ipynb = Código, informe y resultados integrados en un Colab notebook



1 fichero incluyendo todo
o
1 fichero por cada ejercicio

Subid la entrega a PRADO, a la actividad creada para ello.

Fecha de entrega: 14 de Abril

Prácticas de Aprendizaje Automático

Práctica 1

Experimentación con clasificadores y regresores

Pablo Mesejo y Salvador García

Universidad de Granada

Departamento de Ciencias de la Computación e Inteligencia Artificial



UNIVERSIDAD
DE GRANADA

