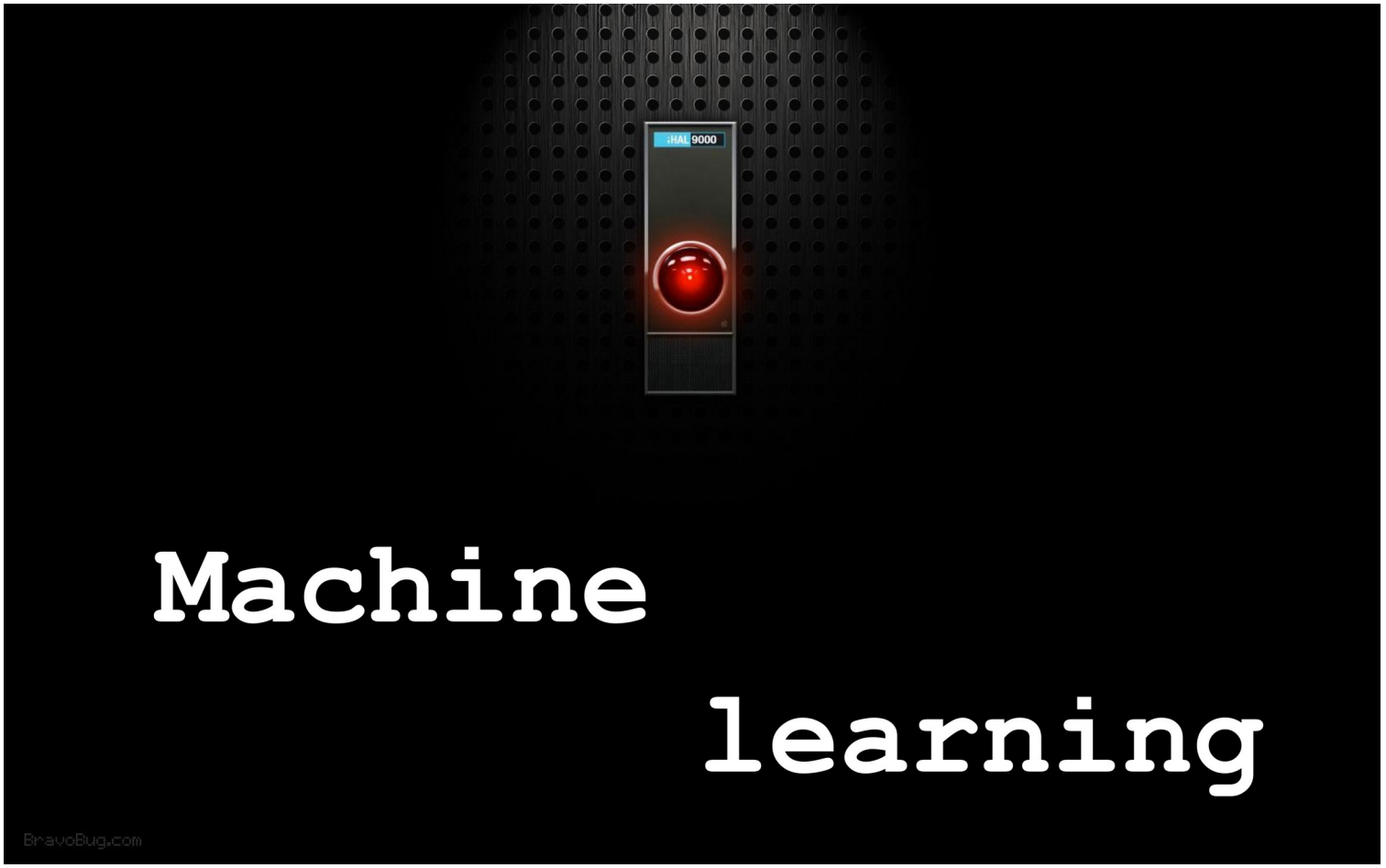
Introducción



Qué es Machine Learning?

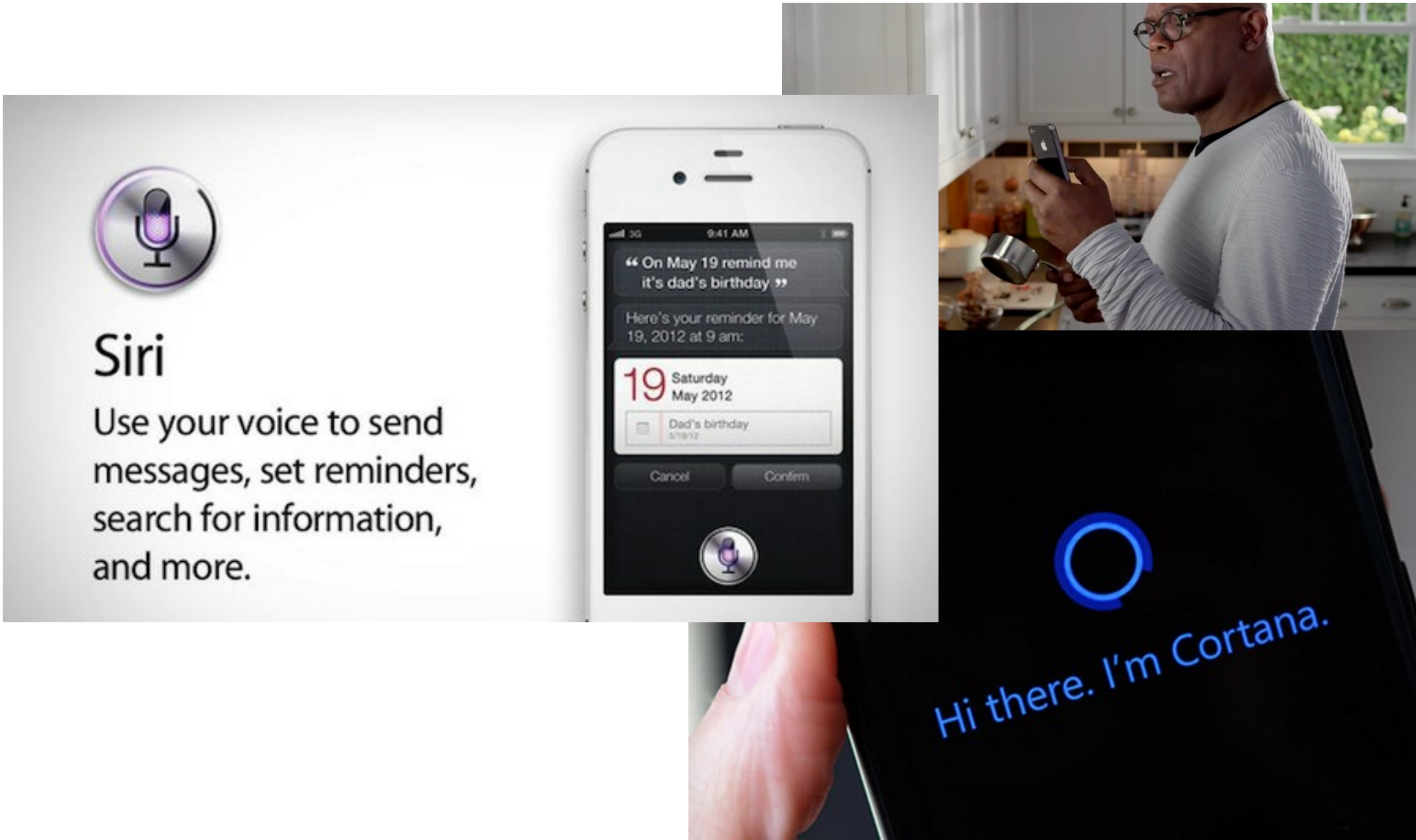
Introducción

- Hay problemas en Informática que se pueden “definir” concretamente y son simples de convertir en un algoritmo
  - Ejemplo: Ordenar alfabéticamente una lista, calcular el balance de una cuenta.

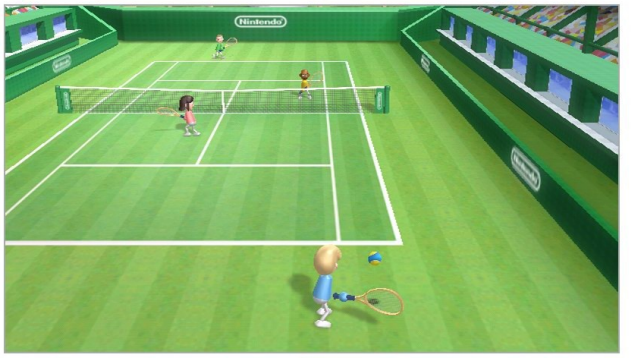
- Hay otros que son simples de “entender” pero muy difíciles de “definir” y convertir en algoritmo
  - Ejemplo: Detectar una sonrisa en una cara, interpretar un gesto del lápiz como una letra dada

El Aprendizaje Automatizado introduce métodos que pueden resolver esas tareas “aprendiendo” la solución a partir de ejemplos de cómo se realiza la misma

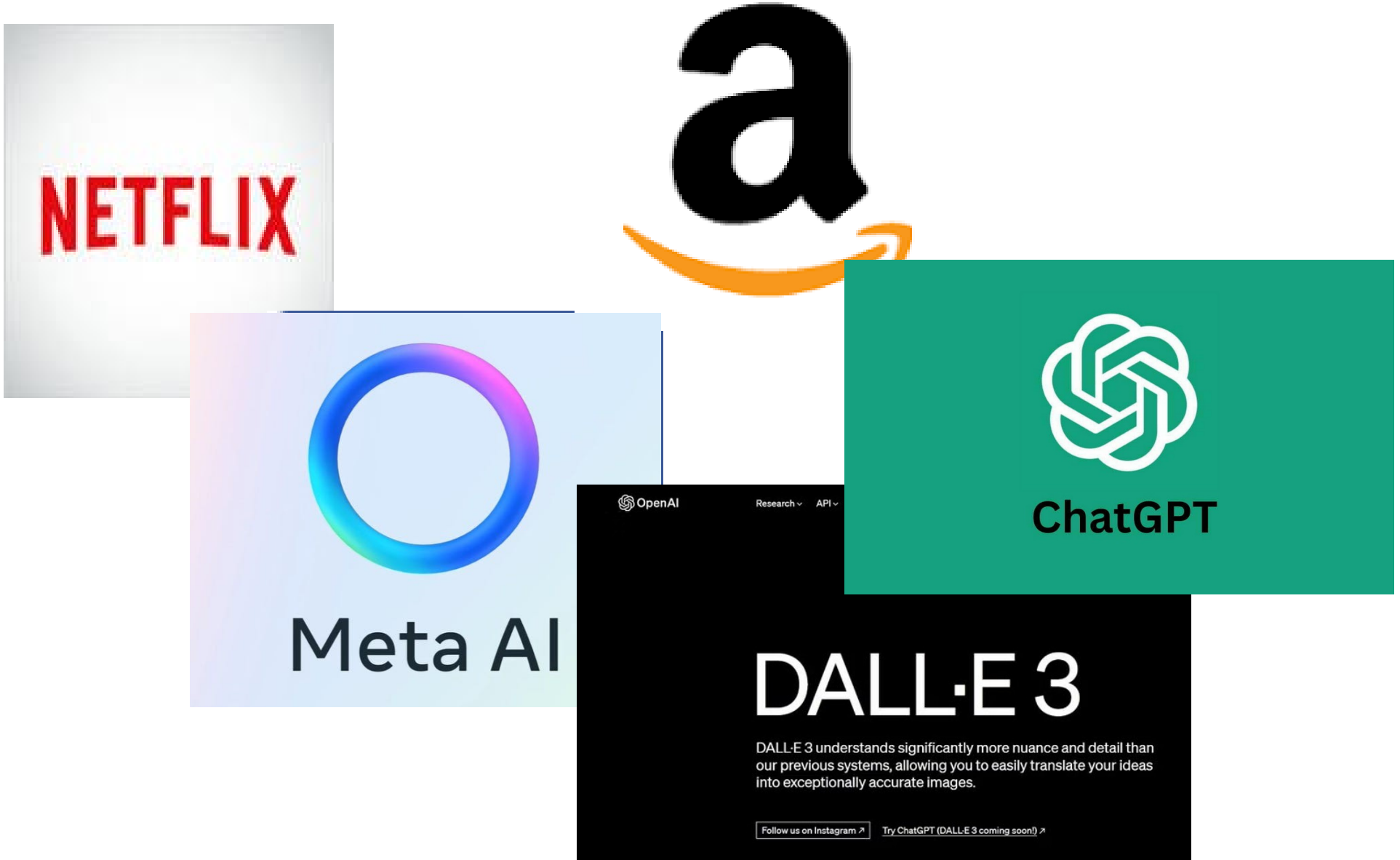
Introducción



Introducción



Introducción



Problemas en ML

- Clasificación
- Regresión
- Ranking-Retrieval
- Detección de novedades
- Clustering
- Identificación de inputs relevantes
- Etc, etc.

Relevantes para Ciencia de Datos

Clasificación

Problema:

Dado un objeto (conjunto de características medidas de alguna forma) asignarle una (o varias) etiqueta de un conjunto finito.

Ejemplo:

asignar un símbolo alfanumérico a una secuencia de movimientos del lápiz en la pantalla táctil

Asignar automáticamente una noticia a diferentes grupos de interés (una o más clases)

Regresión

Problema:

Dado un objeto asignarle un número real.

Ejemplo:

Predecir la relación euro-dolar de mañana.

Predecir niveles de stock/ventas a futuro.

Búsqueda y Ranking

Problema:

Dado un objeto, asignarle y ordenar las respuestas más probables dentro de una base de datos.

Ejemplo:

Buscadores en Internet

Sistemas de recomendación

Detección de novedades

Problema:

Detectar "outliers", objetos que son diferentes a los demás.

Ejemplo:

Alarmas de comportamiento en compras con tarjeta.

Detección de fallas en equipos críticos.

Clustering

Problema:

Detectar grupos de objetos que tienen características similares.

Ejemplo:

Segmentación de consumidores/clientes a partir de sus patrones de compra/búsqueda. Marketing "dirigido".

Detección de inputs relevantes

Problema:

Dado uno de los problemas anteriores (u otro) y sus datos, averiguar cuales de las variables son responsables de la solución.

Ejemplo:

El "nuevo método científico": tomar muestras sanas y con alguna enfermedad. Analizar miles de variables con un método automático (MALDI-TOF, DNA-microchips) y buscar cuales de las variables monitoreadas son relevantes al problema.

Programas que aprenden?

“Se dice que un programa aprende si mejora su performance en una cierta tarea al incorporar experiencia”

Programas que aprenden?

Memorizar no es aprender

Generalizar es aprender

Como logramos generalizar?

Tengo estos datos:

8 – T

2 – T

5 – F

9 – F

4 – T

13 – F

Como logramos generalizar?

Tengo estos datos:

8 – T

2 – T

5 – F

9 – F

4 – T

13 – F

Cual es la respuesta para 12?

Como logramos generalizar?

Tengo estos datos:

8 – T

2 – T

5 – F

9 – F

4 – T

13 – F

Cual es la respuesta para 12?

Y si agrego los datos:

14 – F

16 - T

Como logramos generalizar?

Para generalizar incorporamos “algo” a los datos: un bias.

En general usamos la “navaja de Occam”: La respuesta más simple que explica las observaciones es la válida

Distintos métodos de ML usan distintos bias

Métodos básicos

Arboles de decisión

- Probablemente el método más conocido para resolver problemas de clasificación
- Muy asociado a nuestra forma de proceder
- Uno de los primeros desarrollos en ML

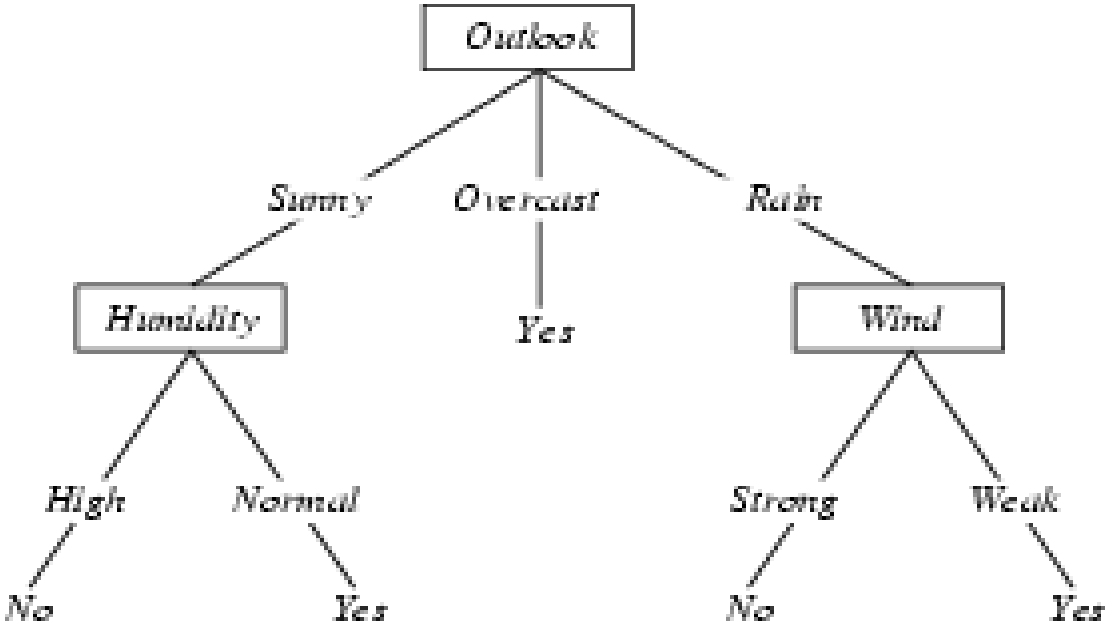
Arboles de decisión

Ejemplo: “Play Tennis”

Day	Outlook	Temperature	Humidity	Wind	PlayTenr
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Arboles de decisión

Ejemplo: “Play Tennis”



Arboles de decisión

Cómo construímos el árbol?

- Hay variables más relevantes que otras.
- Si ponemos más alto las más relevantes, seguramente el árbol llegara antes a la solución, será mas simple.
- Un árbol más simple seguramente generalizará mejor (Occam).

Arboles de decisión

Cómo elegir que variable usar para dividir?

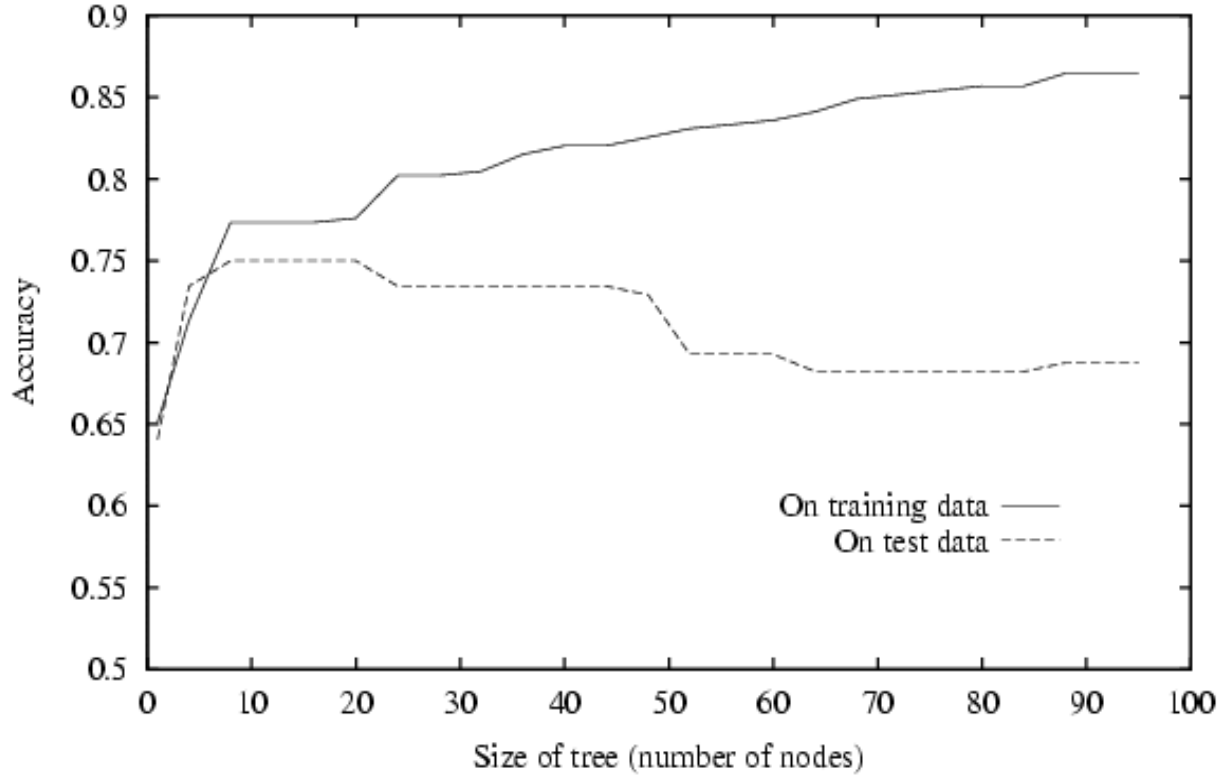
- Necesitamos la más “relevante”
- Busquemos la que da más información sobre la clase->Information Gain, correlación, etc.
- Dividimos e iteramos

Arboles de decisión

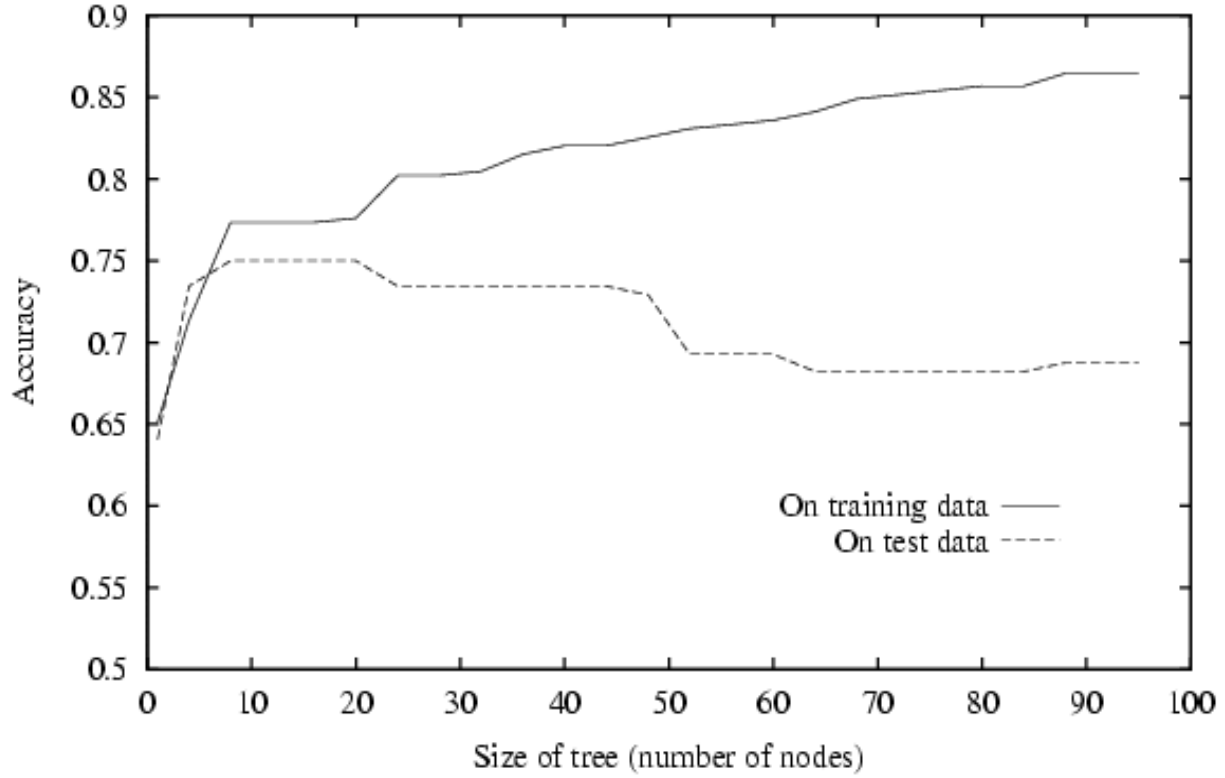
Hasta cuando dividimos?

- Hasta que tenga clases puras en todos los nodos
- Hasta que no tenga más variables disponibles
- O hay algo mejor?

Arboles de decisión



Arboles de decisión



Sobreajuste!

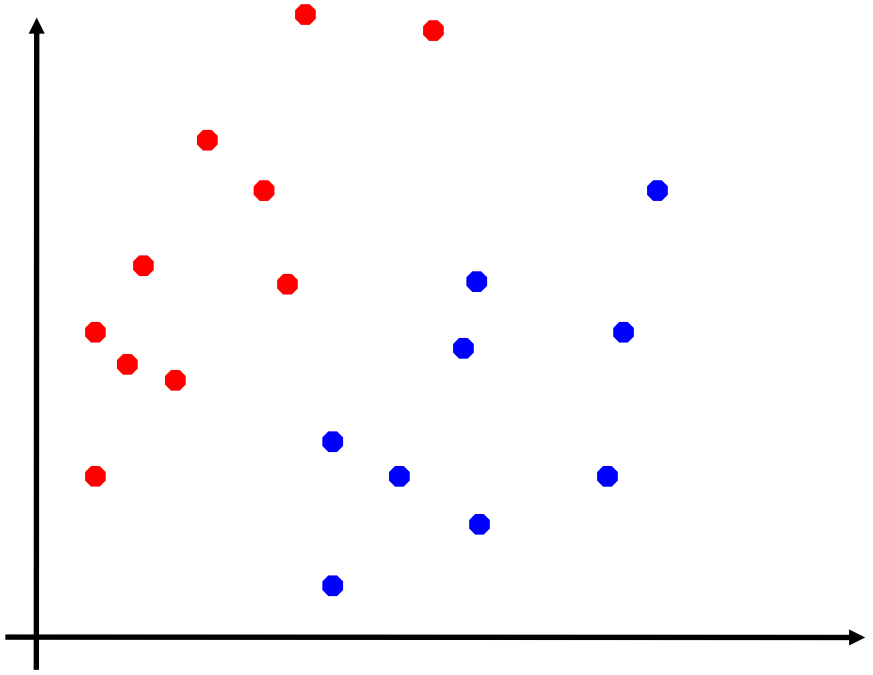
Arboles de decisión

Cómo controlar el sobreajuste?

- Necesitamos un conjunto independiente de datos, sobre el que podamos controlar la capacidad de generalización (“Validación”).
- Cuando deja de mejorar, detenemos el proceso.

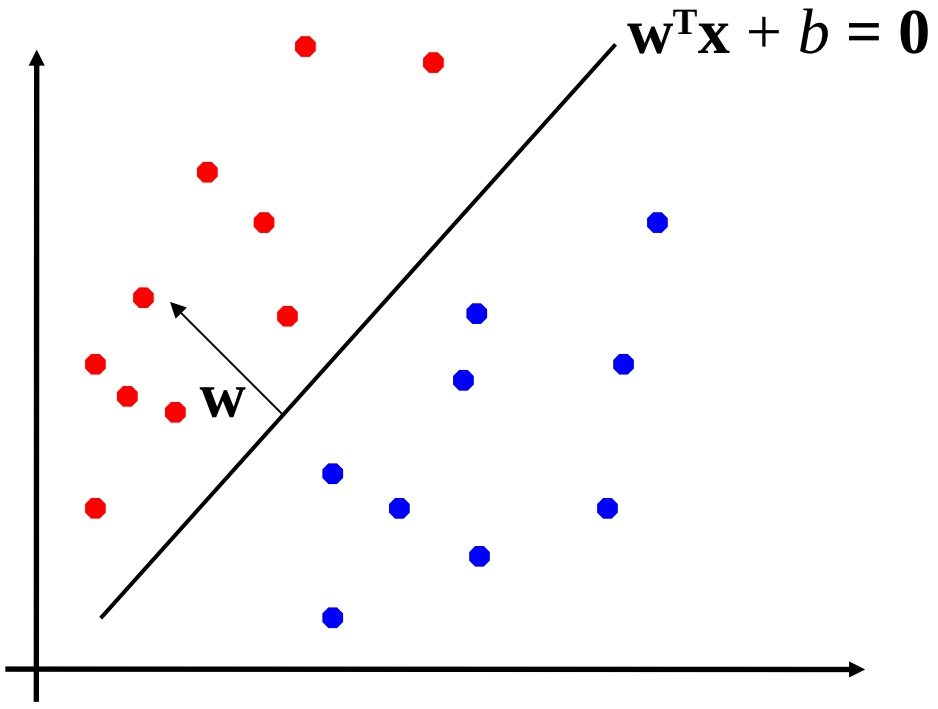
El perceptron

Si los datos están en un espacio vectorial...



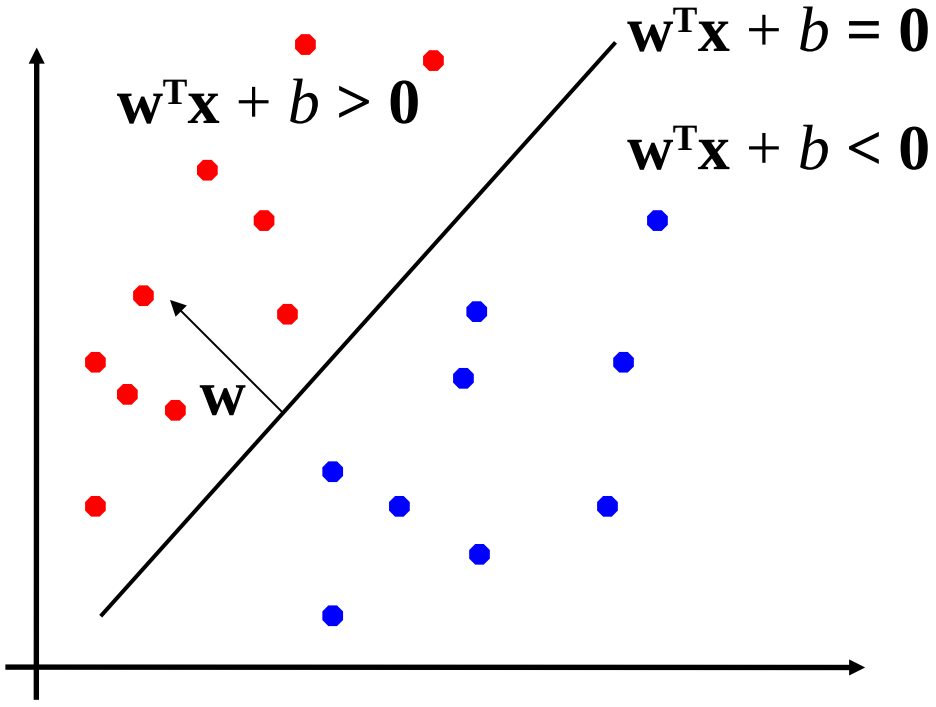
El perceptron

Si los datos están en un espacio vectorial...



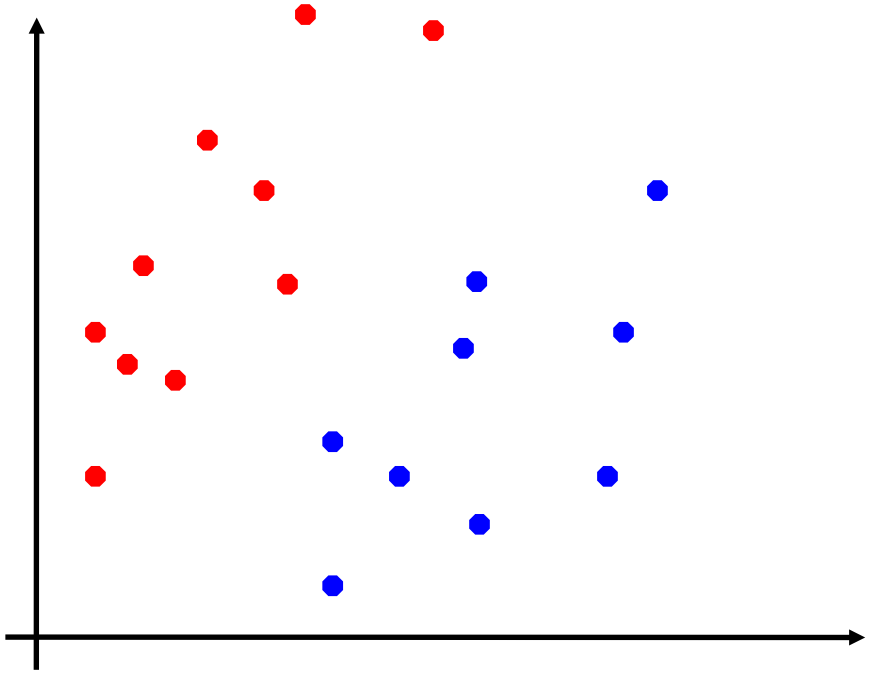
El perceptron

Si los datos están en un espacio vectorial...



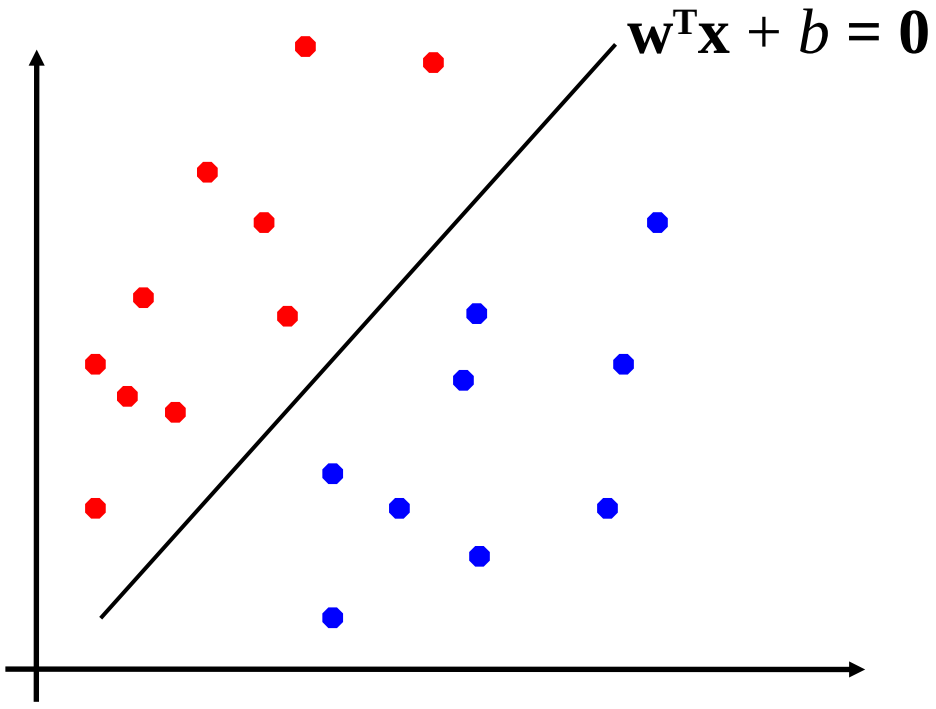
$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$$

El perceptron



- Tenemos una regla de clasificación de forma fija
- Aprender: encontrar el mejor W y b para el problema
- Regla de aprendizaje del perceptron: si es incorrecto muevo W hacia el ejemplo
- Converge a la solución

El perceptron

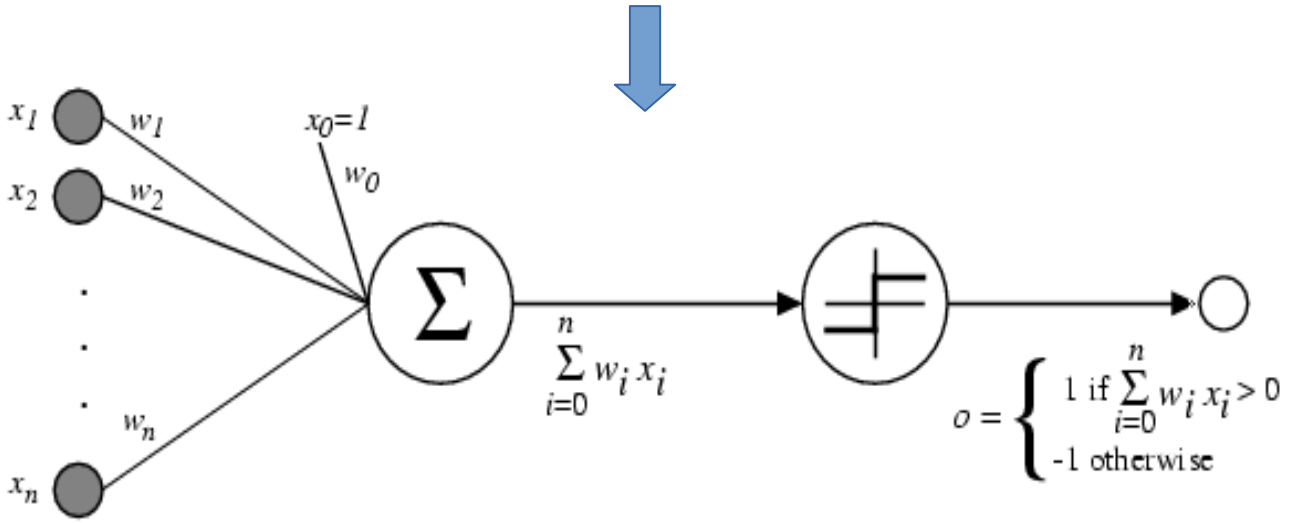


$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$$

- Tenemos una regla de clasificación de forma fija
- Aprender: encontrar el mejor W y b para el problema
- Regla de aprendizaje del perceptron: si es incorrecto muevo W hacia el ejemplo
- Converge a la solución

El perceptron

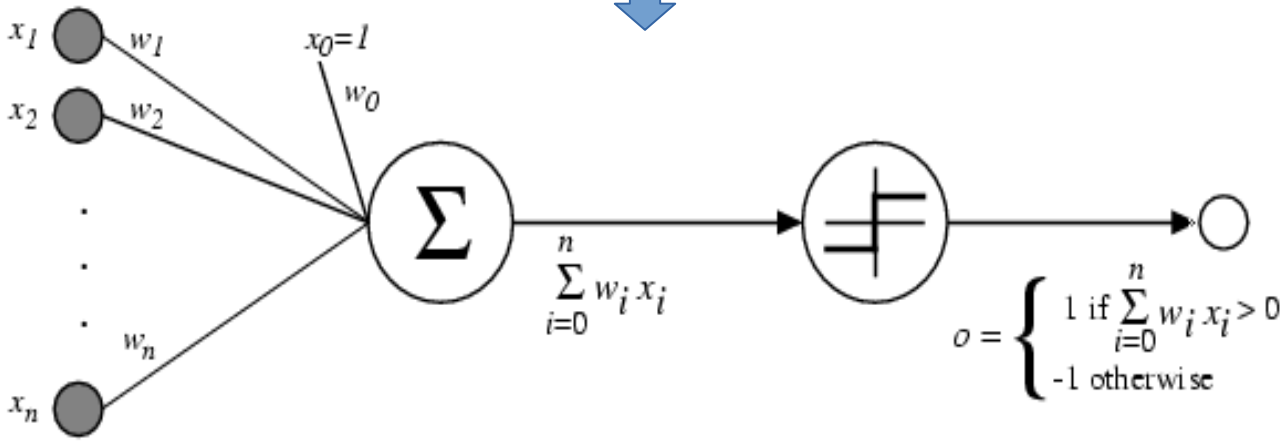
$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b) \qquad si\ b=w_0$$



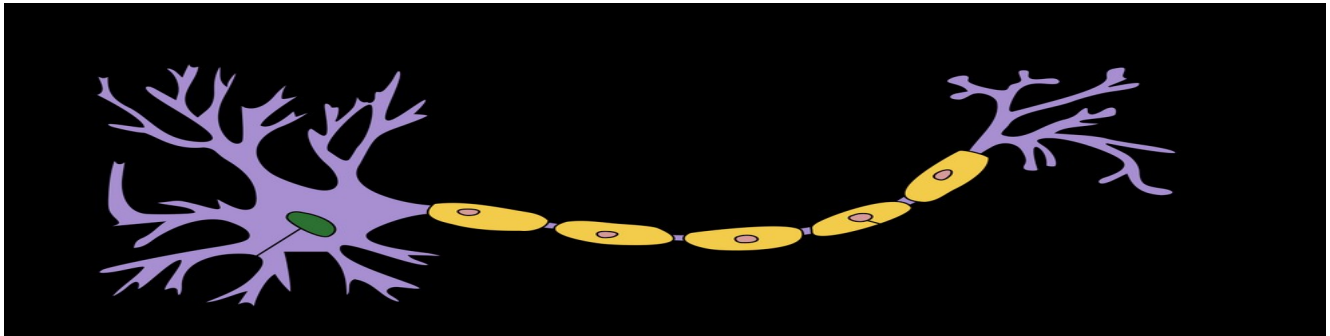
$$o(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$

El perceptron

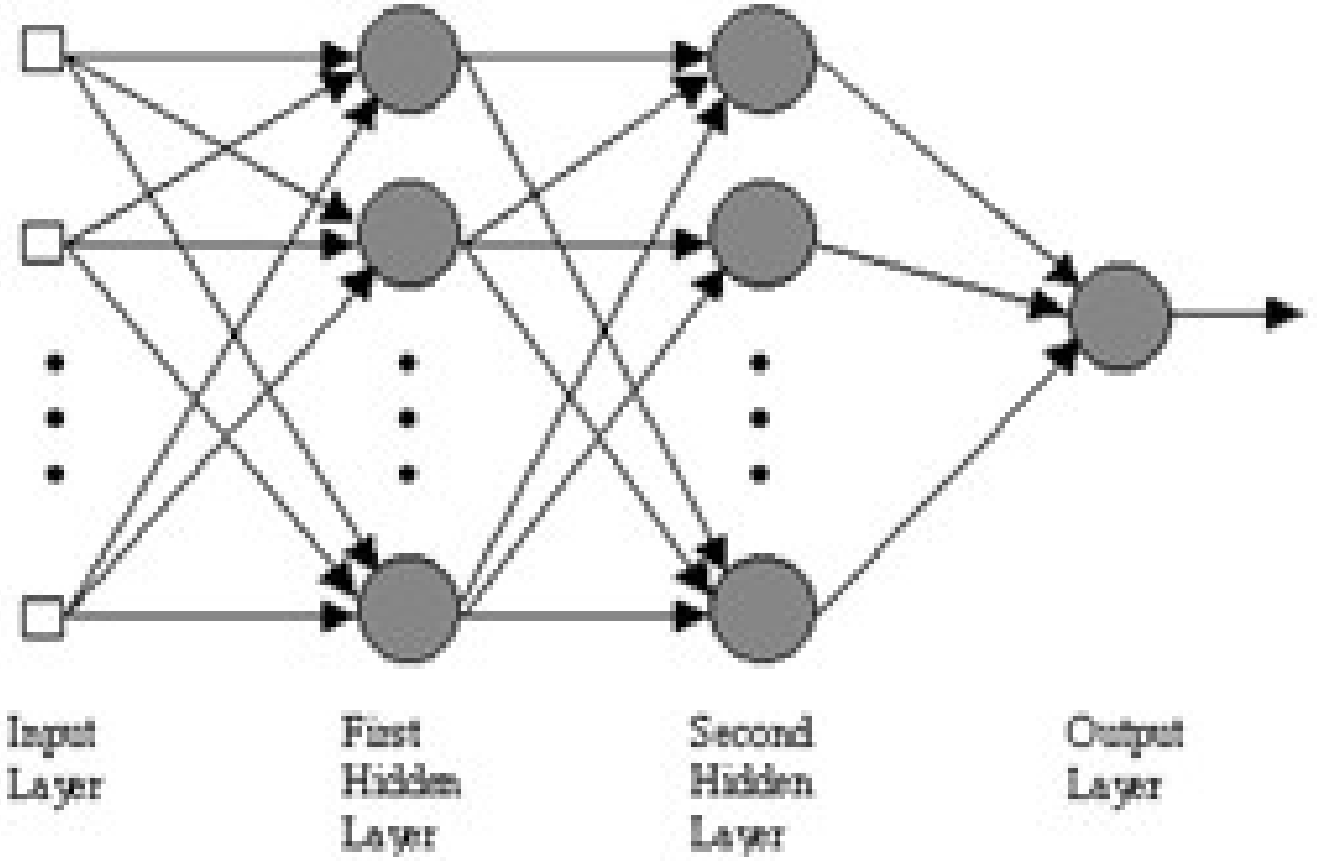
$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b) \qquad si\ b=w_0$$



$$o(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$



El perceptron multicapa o red neuronal



El perceptron multicapa

Como elegir los parámetros para algo tan complejo como la función implementada por un MLP?

Hay que buscar por algún método valores que hagan mínimo el error sobre los datos, que tengan el valor correcto en la salida

Descenso por el gradiente!

El perceptron multicapa

Descenso por el gradiente:

Definir la función error a minimizar

- Calcular el gradiente
- Mover los parámetros en la dirección que minimiza el error
- Iterar hasta converger

Heurísticas para mejorar la convergencia

Aprendizaje por instancias

Aproximar la función objetivo sólo localmente

Modelar sólo al momento de hacer predicciones

Ejemplo: k-vecinos

Aprendizaje por instancias

Aprendizaje con k-vecinos

-Dado un ejemplo X, buscar sus k puntos más cercanos con alguna distancia.

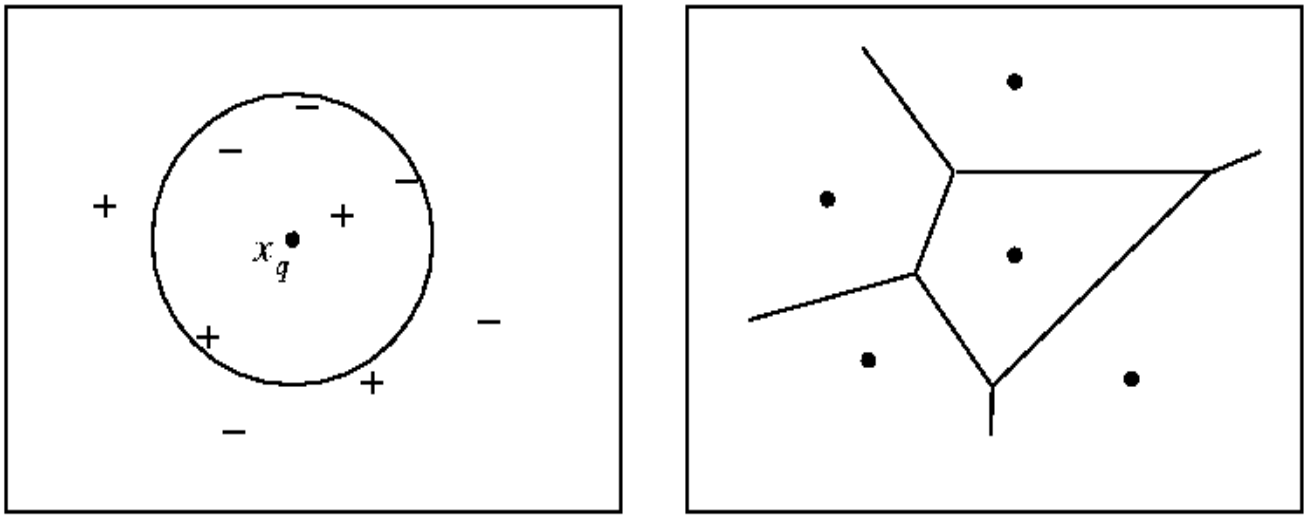
-Predecir usando sólo los k-vecinos:

-Clasificación: Voto sobre la clase

-Regresión: Promedio de los valores

Aprendizaje por instancias

Voronoi Diagram



Aprendizaje por instancias

k es un parámetro a elegir, no cualquier valor sirve, el resultado depende de k.

Cómo elegir k (o cualquier parámetro general)?

- Separar un conjunto de validación dentro de los datos que se usan para aprender.
- Evaluar los posibles valores sobre ese conjunto, elegir el que da menor error.

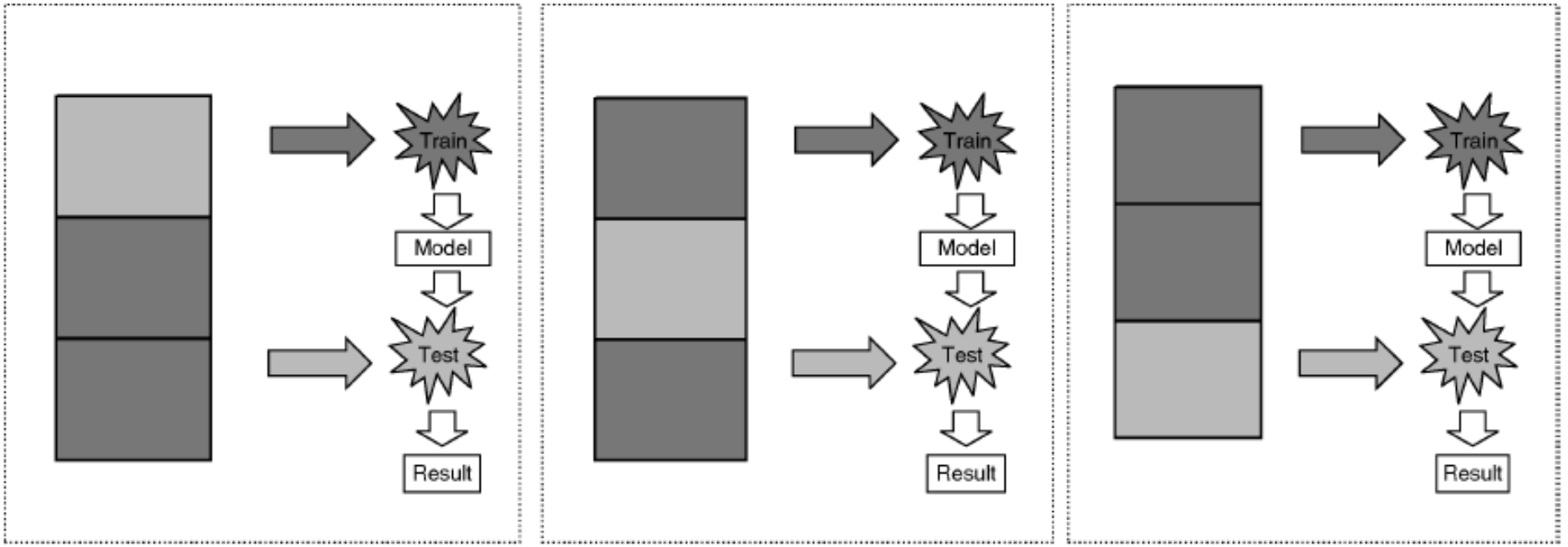
Estimación del error

Como estimar qué tan bueno es lo que aprendí?

La solución más simple es medirlo en una muestra de datos similar a la que voy a usar a futuro: Conjunto de test

Estimación del error

Si no tengo un conjunto de test, estimo con Cross-Validation



Cross-Validation. Figure 1. Procedure of three-fold cross-validation.

Límite: Leave-one-out

Regression: RMSE

- The Root-Mean Squared Error (RMSE) is a cost function for regression. The formula for the RMSE is:

$$RMSE(f) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(f(\mathbf{x}_i) - y_i)^2}$$

where m is the number of test examples, $f(x_i)$, the predicted output on x_i and y_i the actual values.

Regression: RMSE

$$RMSE(f) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(f(\mathbf{x}_i) - y_i)^2}$$

Let’s say my method has an RMSE of 2.13. Is that any good?

Regression: RMSE

$$RMSE(f) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(f(\mathbf{x}_i) - y_i)^2}$$

Let's say my method has an RMSE of 2.13. Is that any good?

RMSE has no scale other than data scale.

Normalized Root-Mean Squared Error (NRMSE), defined as RMSE over the variance of the data, is a better measure.

NRMSE is given in fraction of variance of the data. Predicting with the mean of the outputs gives NRMSE = 1

Clasification

- The basic measures are plain accuracy and error levels:

$$Error(h) = (1/n) \sum_{x \in S} \{ c(x) = h(x)] \}$$

$$Accuracy(h) = (1/n) \sum_{x \in S} \{ c(x) = h(x) \}$$

where n is the number of test examples, h(x) is the predicted output on c(x) is the actual value of the concept.

Confusion Matrix-Based Performance Measures

True class-> Hypothesized class	Pos	Neg
Yes	TP	FP
No	FN	TN
	P=TP+FN	N=FP+TN

Confusion Matrix

- Multi-Class Focus:
  - **Accuracy** = (TP+TN)/(P+N)
  - **Error** = (FP+FN)/(P+N)
- Single-Class Focus:
  - **Precision** = TP/(TP+FP)
  - **Recall** = TP/P
  - **Fallout** = FP/N
  - **Sensitivity** = TP/(TP+FN)
  - **Specificity** = TN/(FP+TN)

Some issues with performance measures

True class->	Pos	Neg
Yes	**200**	100
No	300	**400**
	P=500	N=500

True class ->	Pos	Neg
Yes	**400**	300
No	100	**200**
	P=500	N=500

- Both classifiers obtain 60% accuracy
- They exhibit very different behaviours:
  - On the left: weak positive recognition rate/strong negative recognition rate
  - On the right: strong positive recognition rate/weak negative recognition rate

Some issues with performance measures (cont'd)

True class →	Pos	Neg
Yes	**0**	0
No	5	**500**
	P=5	N=500

True class →	Pos	Neg
Yes	**4**	50
No	1	**450**
	P=5	N=500

- The classifier on the left obtains 99.01% accuracy while the classifier on the right obtains 89.9%
  - Yet, the classifier on the right is much more sophisticated than the classifier on the left, which just labels everything as negative and misses all the positive examples.

Some issues with performance measures (cont'd)

True class →	Pos	Neg
Yes	200	100
No	300	**400**
	P=500	N=500

True class →	Pos	Neg
Yes	200	100
No	300	**0**
	P=500	N=100

- Both classifiers obtain the same precision and recall values of 66.7% and 40% (Note: the data sets are different)
- They exhibit very different behaviours:
  - Same positive recognition rate
  - Extremely different negative recognition rate: strong on the left / nil on the right
- Note: Accuracy has no problem catching this!

Making sense

- How to know if your method works well?

Making sense

- How to know if your method works well?
  - Compare with chance
  - Compare with trivial methods
  - Compare with previous methods (including human performance)
  - Compare with best theoretical value (Bayes level)

Como resolver un problema en ML

- Identificar el problema y conseguir conocimiento experto
- Conseguir datos, muchos datos!
- Elegir un método adecuado (o varios)
- Entrenar varios modelos con el conjunto de train, evaluarlos con el conjunto de validación, elegir el mejor
- Estimar el error con el conjunto de test