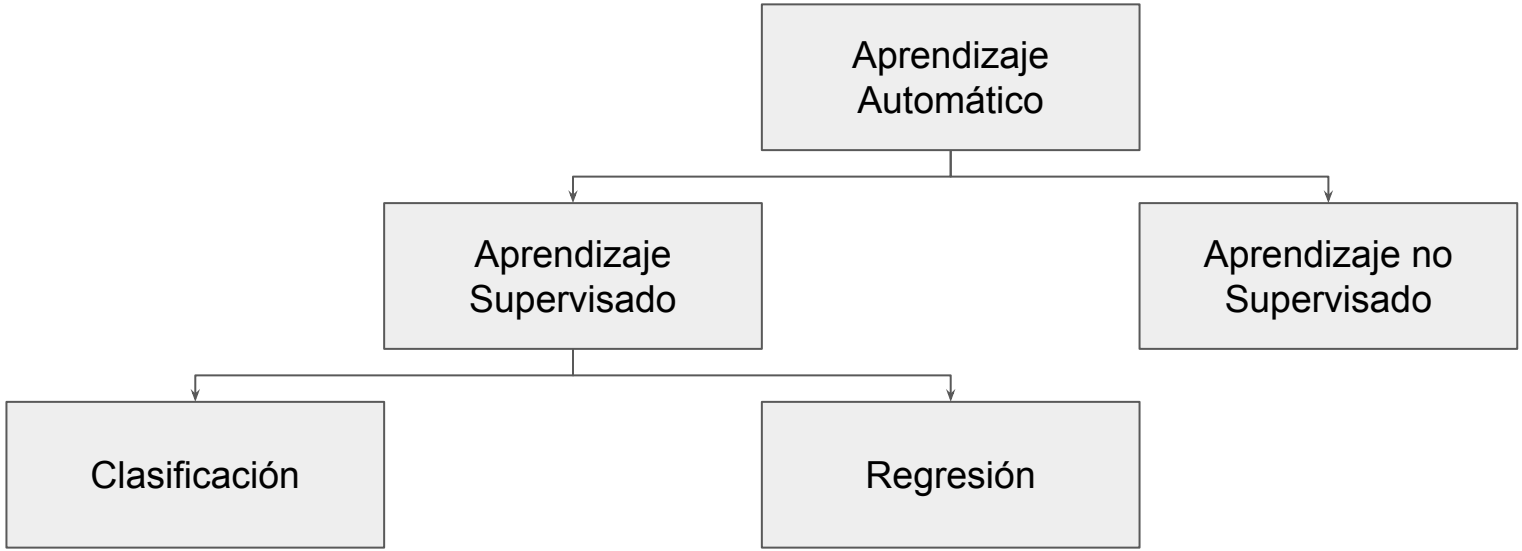
Aprendizaje supervisado 1

Métricas, MLP y SVM

Fabián Villena & Mauricio Cerda

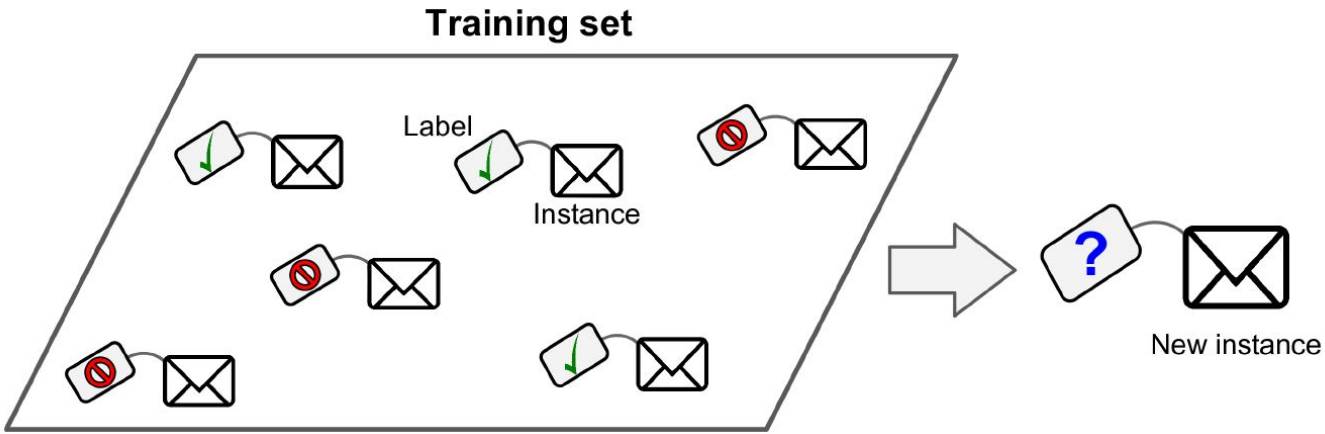
Aprendizaje supervisado

En el aprendizaje supervisado, los datos de entrenamiento que se ingresan al algoritmo incluyen las soluciones.



Clasificación

Una de las tareas que se resuelven con aprendizaje supervisado es la clasificación, en donde la etiqueta asociada a cada ejemplo es una categoría o *clase*.



Métricas de rendimiento

Para poder entender el rendimiento de un modelo y poder valorar si está funcionando de manera correcta, debemos calcular un valor numérico. Cada métrica de rendimiento va a tener un significado específico y debemos interpretarlas de manera correcta.

	precision	recall	f1-score	support
0	0.77	0.86	0.81	37584
1	0.84	0.75	0.79	37577
accuracy			0.80	75161
macro avg	0.81	0.80	0.80	75161
weighted avg	0.81	0.80	0.80	75161

Matriz de confusión

Una manera de evaluar el rendimiento de un clasificador es observar la matriz de confusión. La idea general es contar la cantidad de veces que los ejemplos de la clase A son clasificados como clase B. Este cálculo normalmente se debe realizar en el subconjunto de prueba.

		Predicho	
		Negativo	Positivo
Real	Negativo	Verdaderos negativos (FN)	Falsos positivos (FN)
	Positivo	Falsos negativos (FN)	Verdaderos positivos (FN)

Accuracy

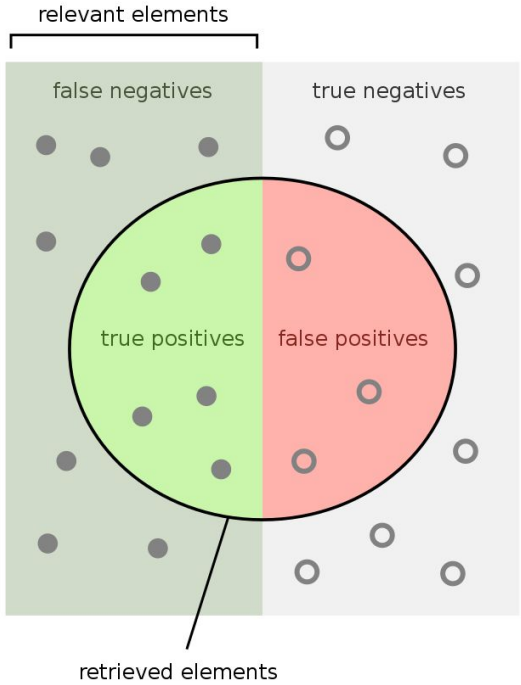
La *accuracy* es una de las métricas que podemos utilizar para evaluar el rendimiento de un modelo. La *accuracy* es la fracción de predicciones que nuestro modelo realizó correctamente. El problema es que en problemas desbalanceados, esta métrica puede llevarnos a falsas conclusiones.

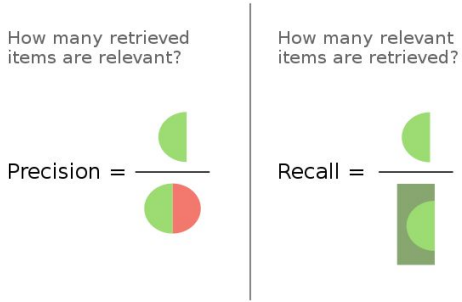
$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

Precision y *recall*

- La *precision* busca identificar la proporción de predicciones positivas que realmente eran positivas.
- El *recall* busca identificar la proporción de elementos realmente positivos que se predijeron como positivos.

Para evaluar el rendimiento de nuestro modelo debemos evaluar ambas métricas. Típicamente existe un compromiso entre ambas métricas que debemos manejar con cuidado.





F-score

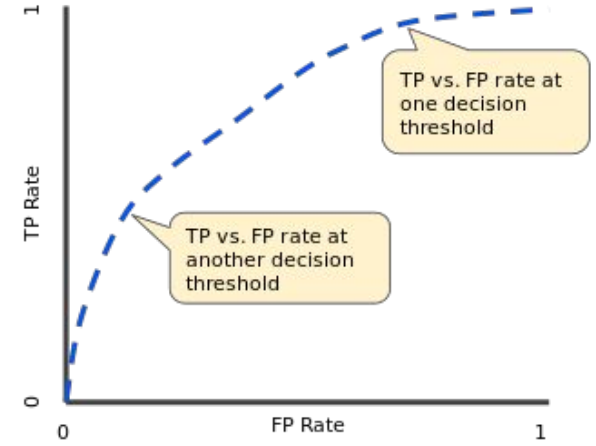
El F-score es una métrica resumen que nos comunica el resultado de la *precision* y el *recall* en un sólo valor. La versión más utilizada es el F_1-score, que pondera con el mismo peso la *precision* y el *recall*.

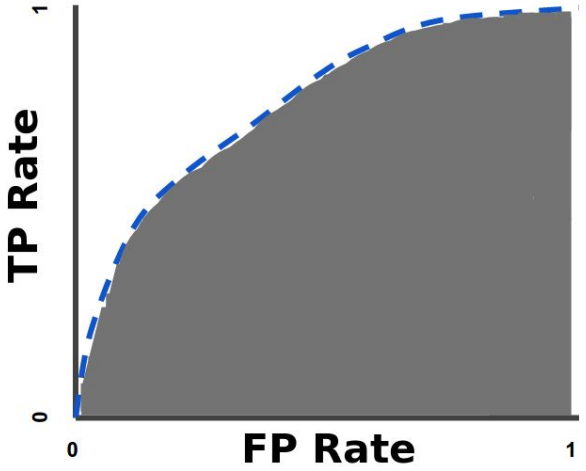
$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

$$F_1 = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}} = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Área bajo la curva ROC

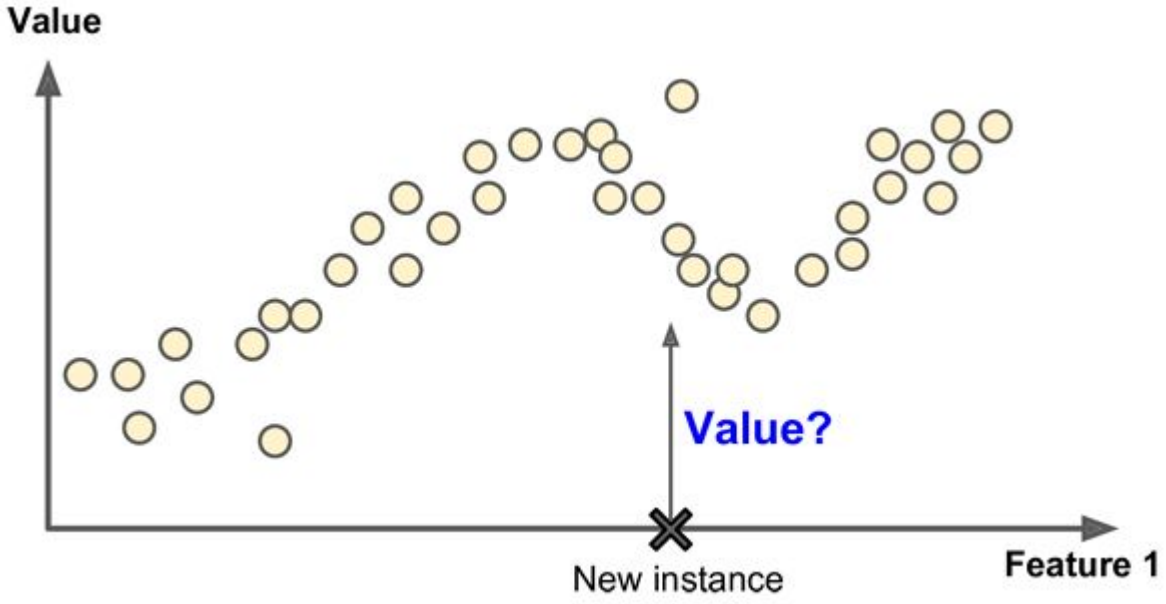
La curva ROC es una representación del cambio en el False Positive Rate (FPR; la proporción de elementos negativos que fueron incorrectamente clasificados como positivos) y el True Positive Rate (TPR; Recall) de un clasificador binario al modular su umbral de decisión. La métrica derivada de esta curva es el área bajo la misma.





Regresión

Una de las tareas que se resuelven con aprendizaje supervisado es la regresión, en donde la etiqueta asociada a cada ejemplo es un valor continuo.



Métricas de rendimiento

Las métricas más utilizadas para evaluar el rendimiento de un regresor son el *Mean Absolute Error* (MAE), *Mean Squared Error* (MSE) y el *Root Mean Squared Error* (RMSE).

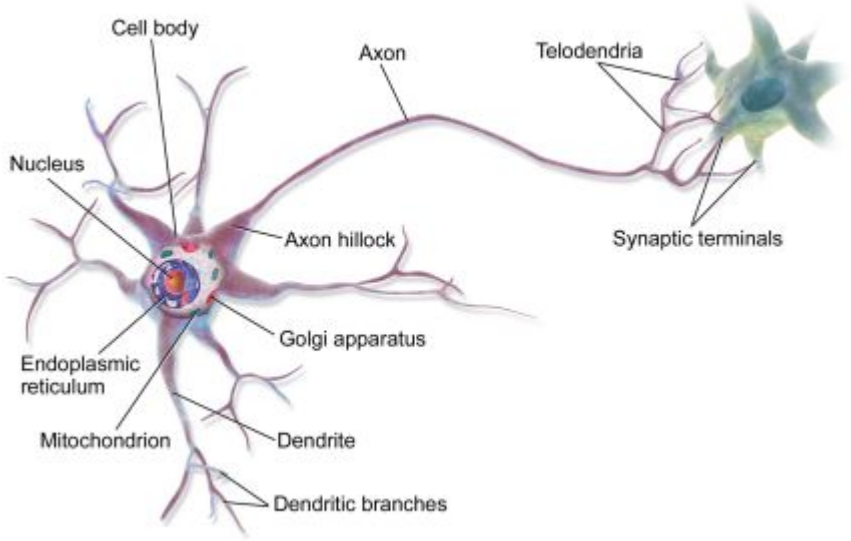
$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} \left| y_i - \hat{y}_i \right|}{n}$$

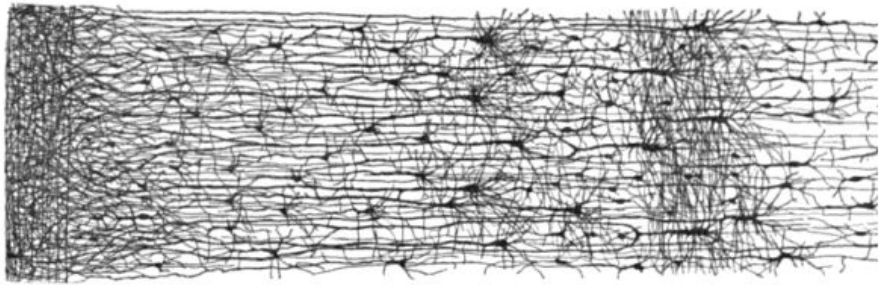
$$\mathrm{MSE} = \frac{\sum_{i=1}^{n} \left| (y_i - \hat{y}_i)^2 \right|}{n}$$

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} \left| (y_i - \hat{y}_i)^2 \right|}{n}}$$

Neurona Biológica

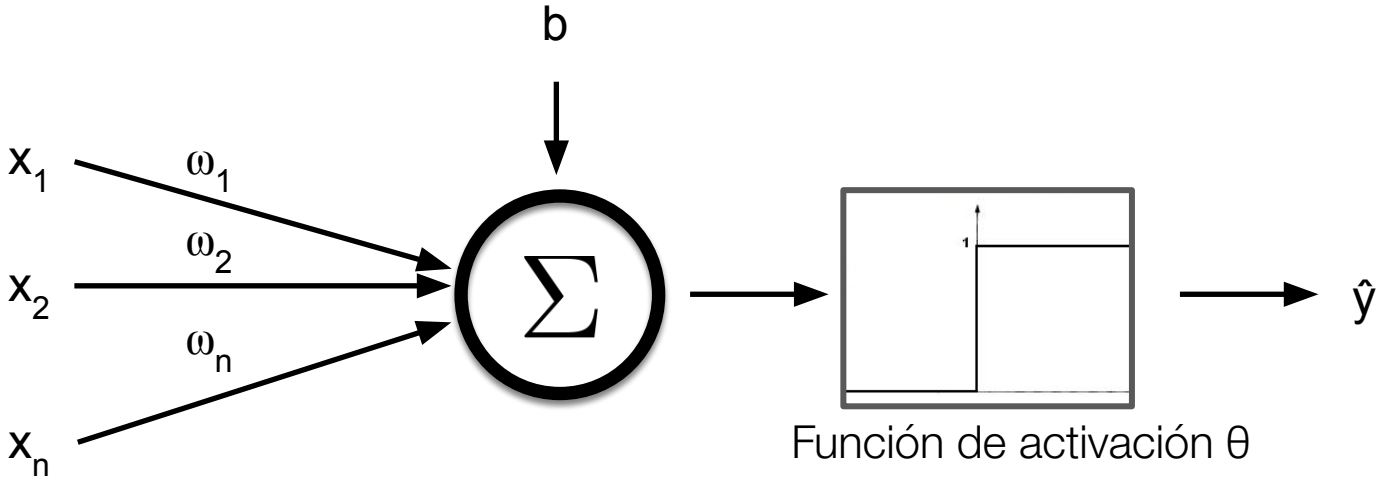
Una célula compuesta por dendritas, unas prolongaciones que reciben pulsos eléctricos que son procesados en el cuerpo y pueden ser transmitidos a través del axón, el cual puede conectarse a más neuronas. Complejas computaciones pueden realizarse en una red de neuronas.





Perceptrón

Modelo de funcionamiento simple de una neurona única.



$$\hat{y} = \theta(x_1*\omega_1 + x_2*\omega_2 + \ldots + x_n*\omega_n + b)$$

Aprendizaje con un perceptrón

$$\omega_i = \omega_i + \Delta\omega_i$$
$$\Delta\omega_i = \rho * (y - \hat{y}) * x_i$$

Donde ρ es la tasa de aprendizaje, $\hat{y}$ es la salida del perceptrón.

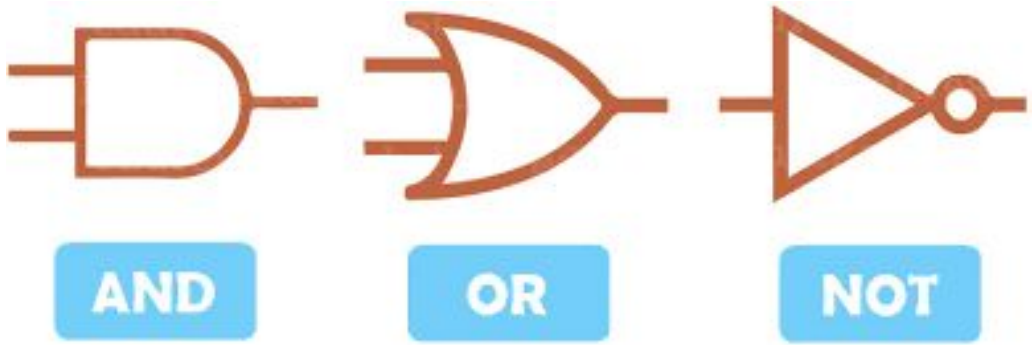
- Si y = $\hat{y}$ los pesos ω_i no cambian
- Si y != $\hat{y}$ los pesos ω_i cambian para hacer la salida lo más parecida al objetivo.

El algoritmo converge si:

- Los datos son linealmente separables.
- ρ es suficientemente pequeño.

Perceptrones conocidos

- AND: $\omega_1 = 1$, $\omega_2 = 1$, $b = -1.5$
- OR: $\omega_1 = 1$, $\omega_2 = 1$, $b = -0.5$
- NOT: $\omega_1 = -1$, $b = 0.5$



Ejemplo de entrenamiento de un perceptrón

Iteración 1, $\theta(x) = x > 0.5$, $w = (0.1, 0.2, 0.3)$, $\rho = 0.1$

x_1	x_2	x_3	$\theta(\mathbf{w} * \mathbf{x})$	y	$\hat{y}$
-1	0	0		0	
-1	0	1		0	
-1	1	0		0	
-1	1	1		1	

Iteración 2, $\theta(x) = x > 0.5$, $w = (?, ?, ?)$, $\rho = 0.1$

x_1	x_2	x_3	$\theta(\mathbf{w} * \mathbf{x})$	y	$\hat{y}$
-1	0	0		0	
-1	0	1		0	
-1	1	0		0	
-1	1	1		1	

Ejemplo de entrenamiento de un perceptrón

Iteración 1, $\theta(x) = x > 0.5$, $w = (0.1, 0.2, 0.3)$, $\rho = 0.1$

x_1	x_2	x_3	$\theta(\mathbf{w} * \mathbf{x})$	y	$\hat{y}$
-1	0	0	(0.1*-1 + 0.2*0 + 0.3*0) > 0.5	0	0
-1	0	1	(0.1*-1 + 0.2*0 + 0.3*1) > 0.5	0	0
-1	1	0	(0.1*-1 + 0.2*1 + 0.3*0) > 0.5	0	0
-1	1	1	(0.1*-1 + 0.2*1 + 0.3*1) > 0.5	1	0

Iteración 2, $\theta(x) = x > 0.5$, $w = (?, ?, ?)$, $\rho = 0.1$

x_1	x_2	x_3	$\theta(\mathbf{w} * \mathbf{x})$	y	$\hat{y}$
-1	0	0		0	
-1	0	1		0	
-1	1	0		0	
-1	1	1		1	

Ejemplo de entrenamiento de un perceptrón

$$\omega_i = \omega_i + \Delta\omega_i$$
$$\Delta\omega_i = \rho * (y - \hat{y}) * x_i$$

$\Delta\omega_1 = 0.1 * (1 - 0) * -1 = -0.1$
$\Delta\omega_2 = 0.1 * (1 - 0) * \ \ 1 = \ \ 0.1$
$\Delta\omega_3 = 0.1 * (1 - 0) * \ \ 1 = \ \ 0.1$

$\omega_1 = 0.1 + -0.1 = 0.0$
$\omega_2 = 0.2 + \ \ 0.1 = 0.3$
$\omega_3 = 0.3 + \ \ 0.1 = 0.4$

Ejemplo de entrenamiento de un perceptrón

Iteración 1, $\theta(x) = x > 0.5$, $w = (0.1, 0.2, 0.3)$, $\rho = 0.1$

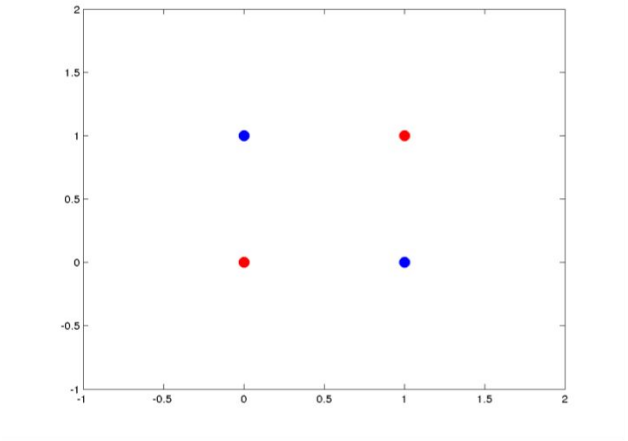
x1	x2	x3	$\theta(\mathbf{w} * \mathbf{x})$	y	$\hat{y}$
-1	0	0	(0.1*-1 + 0.2*0 + 0.3*0) > 0.5	0	0
-1	0	1	(0.1*-1 + 0.2*0 + 0.3*1) > 0.5	0	0
-1	1	0	(0.1*-1 + 0.2*1 + 0.3*0) > 0.5	0	0
-1	1	1	(0.1*-1 + 0.2*1 + 0.3*1) > 0.5	1	0

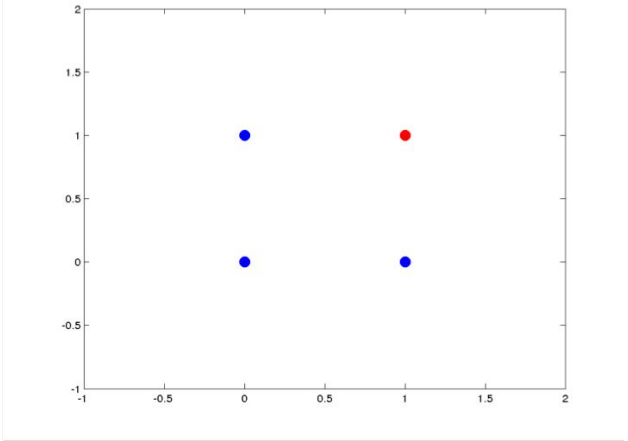
Iteración 2, $\theta(x) = x > 0.5$, $w = (0.0, 0.3, 0.4)$, $\rho = 0.1$

x1	x2	x3	$\theta(\mathbf{w} * \mathbf{x})$	y	$\hat{y}$
-1	0	0	(0.0*-1 + 0.3*0 + 0.4*0) > 0.5	0	0
-1	0	1	(0.0*-1 + 0.3*0 + 0.4*1) > 0.5	0	0
-1	1	0	(0.0*-1 + 0.3*1 + 0.4*0) > 0.5	0	0
-1	1	1	(0.0*-1 + 0.3*1 + 0.4*1) > 0.5	1	1

El problema del perceptrón

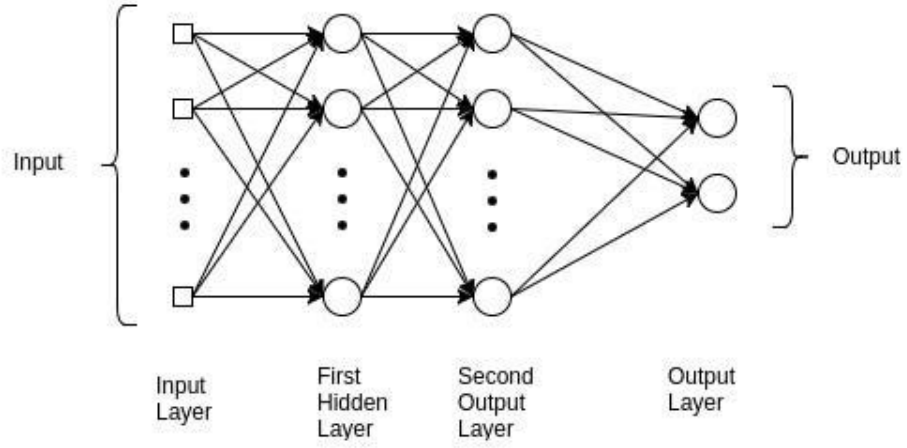
El perceptrón puede hacer sólo una separación lineal. Esto provocó desinterés en las redes neuronales en la década del 70 y 80.



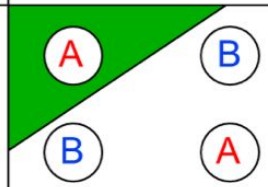
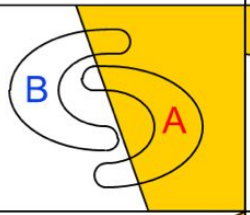
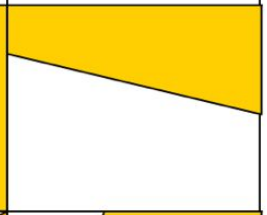
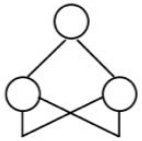
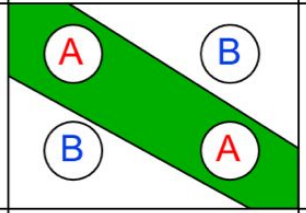
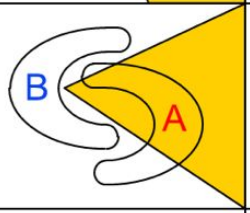
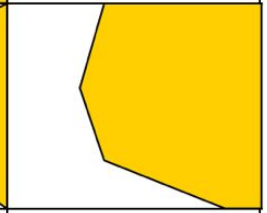
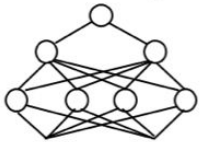
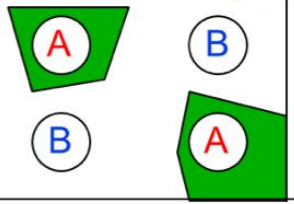
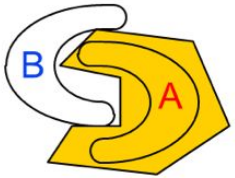
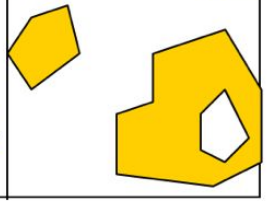


Perceptrón multicapa (MLP)

Si nosotros somos capaces de apilar distintas neuronas en distintas disposiciones, es posible resolver problemas más complejos, como los problemas en donde no existe una separación lineal. Ahora el aprendizaje se debe propagar entre distintas capas.

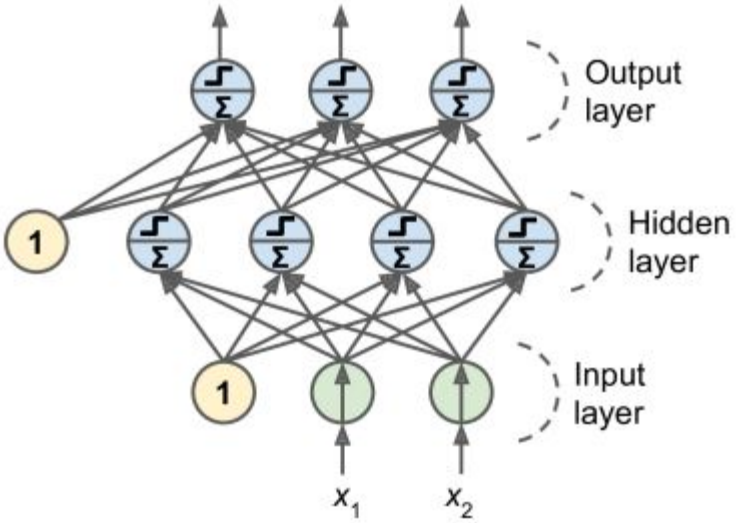


Tipos de problema y perceptrón multicapa

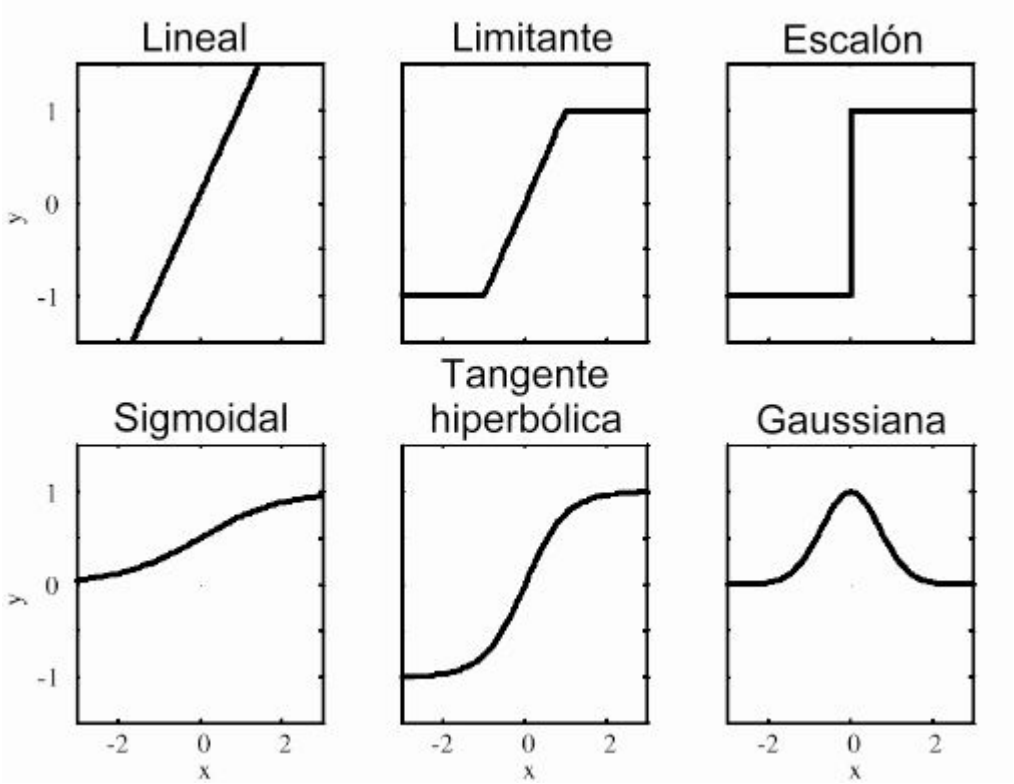
Structure	Types of Decision Regions	Exclusive-OR Problem	Classes with Meshed regions	Most General Region Shapes
Single-Layer 	Half Plane Bounded By Hyperplane			
Two-Layer 	Convex Open Or Closed Regions			
Three-Layer 	Arbitrary (Complexity Limited by No. of Nodes)			

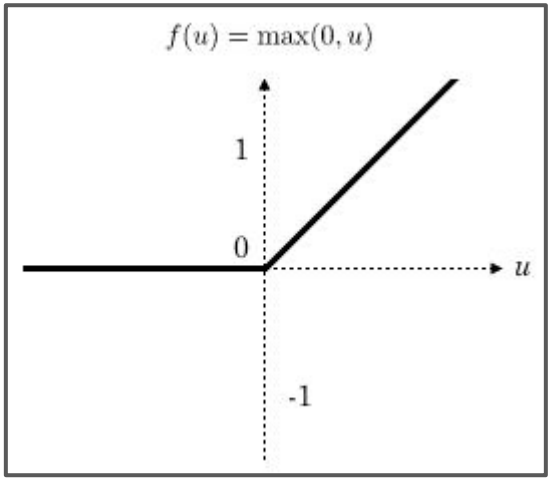
Arquitectura de un perceptrón multicapa

- Capa de entrada
  - Recibe las entradas
  - Las introduce a la red
- Capas ocultas
  - Realizan clasificación
  - Más características de entrada puede implicar más capas
- Capa de salida
  - Similar a la capa de entrada
  - Retorna los resultados de la red hacia el exterior. Debe ser diseñada en función al problema a resolver.



Funciones de activación





ReLU

Entrenamiento de un perceptrón multicapa

- Se ajustan los pesos de la red neuronal para mapear entradas a salidas.
- Usar un conjunto de patrones de ejemplo donde la salida es conocida para ciertas entradas.
- El objetivo es que la red pueda también generalizar.
- Reconocer características que son buenas/malas para clasificar.

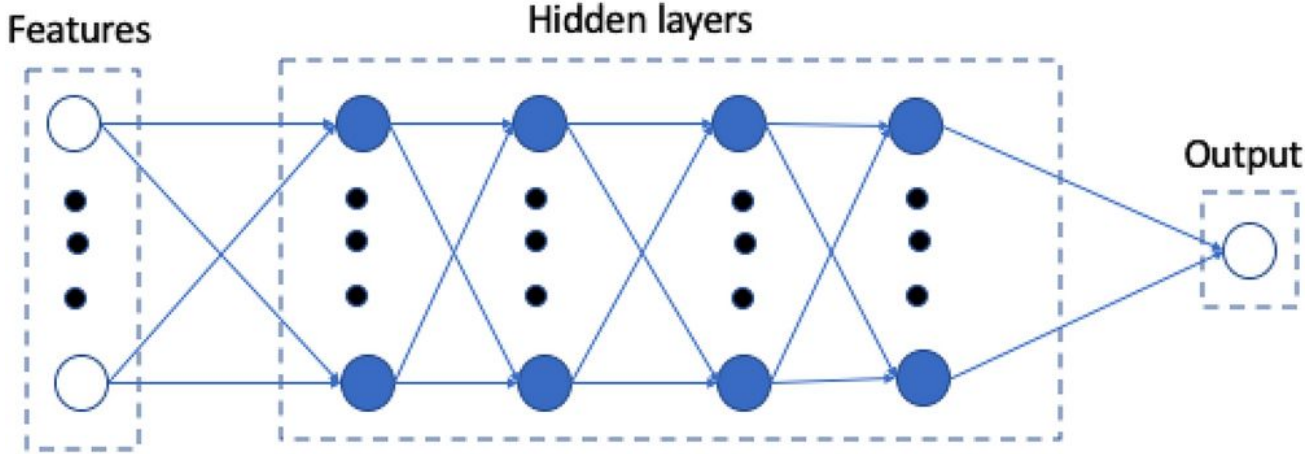
Backpropagation

Es un método para ajustar una red neuronal. Esta técnica es una forma eficiente de ajustar los pesos de las conexiones.

1. Pasamos un grupo de ejemplos de entrenamiento por la red y calculamos las predicciones.
2. Se calcula el error respecto a las respuestas reales.
3. Se calcula cuánto cada conexión aportó a ese error.
4. Finalmente se ajustan los pesos de cada conexión en función del aporte al error de cada conexión.

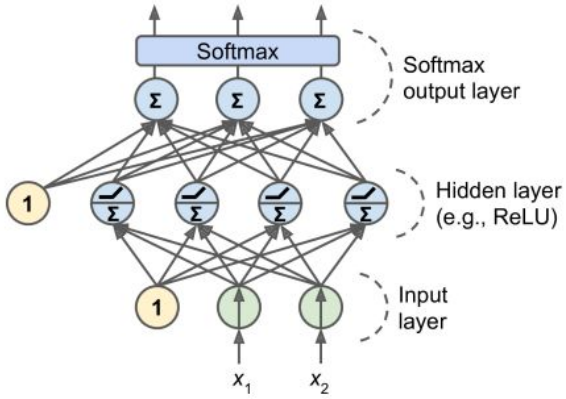
Regresión con MLP

Si se necesita predecir un valor único, entonces se necesita sólo una unidad de salida en la red neuronal. Podemos no sólo predecir un único valor, sino que también podríamos predecir un vector de valores, simplemente al colocar tantas unidades como sea necesario en la capa de salida.



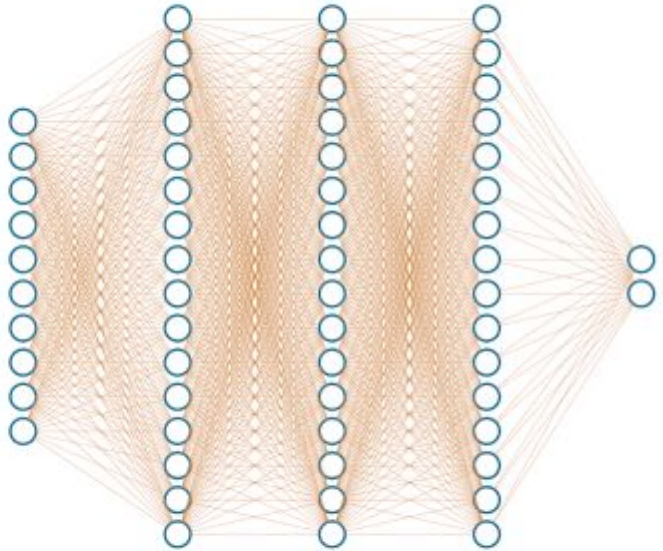
Clasificación con MLP

Para una clasificación binaria, podemos colocar una unidad única en la capa de salida y aplicar una función de activación sigmoide para obtener valores entre 0 y 1 para poder interpretar la salida como la probabilidad de pertenecer a la clase positiva.
Podemos también resolver problemas multiclase (múltiples unidades con activación softmax) y multietiqueta (múltiples unidades con activaciones sigmoides).



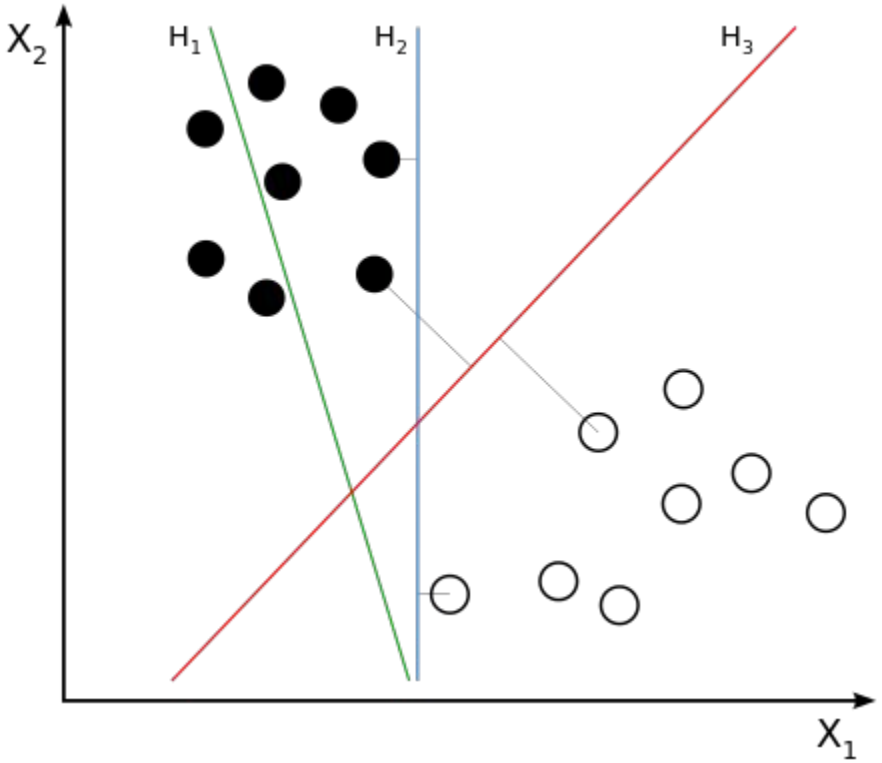
Desventajas de un MLP

- Sobreajuste
- Largo período de entrenamiento
- Múltiples parámetros a ajustar
  - Número de capas
  - Número de neuronas
  - Más neuronas => más entrenamiento
  - Tasa de aprendizaje (experimental 0.1)



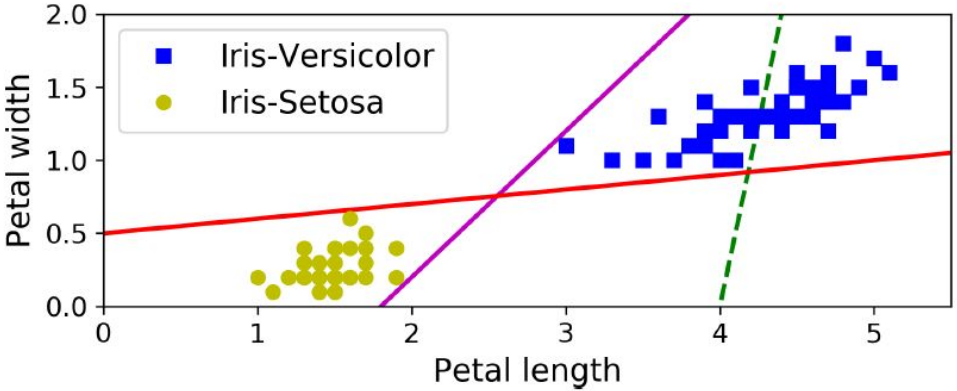
Support vector machines

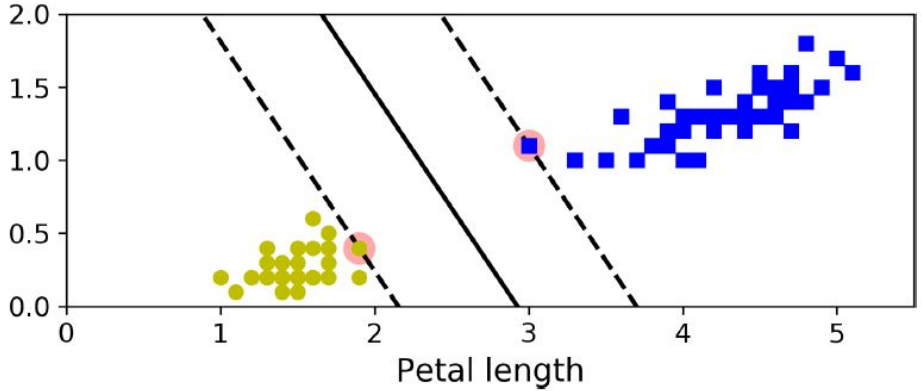
Las *support vector machines* son modelos muy versátiles para resolver tareas de clasificación y de regresión. Es uno de los modelos más utilizados en el área del aprendizaje automático.



SVM lineal

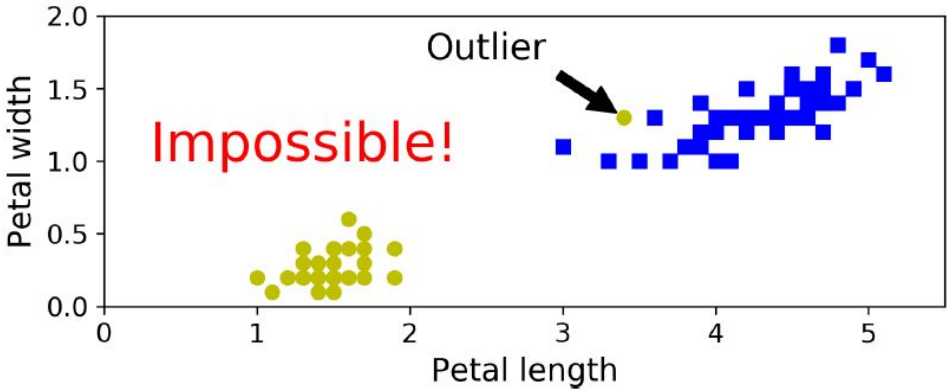
Con las SVM buscamos ajustar una línea que separe las clases con el mayor margen posible. Notar que al añadir ejemplos fuera del margen, no afectaría el umbral de decisión, porque está completamente soportado por los elementos localizados en el borde del margen.

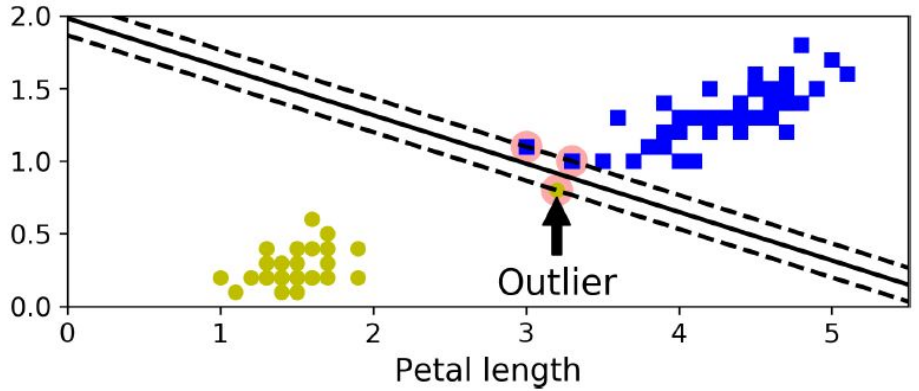




Clasificación de margen suave

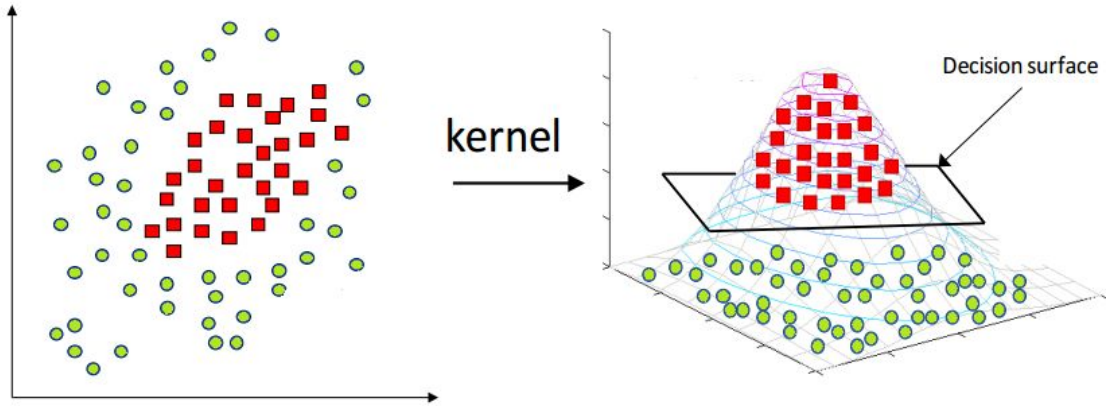
Si imponemos que estrictamente todos los elementos deben quedar a cada lado del margen, el modelo se vuelve muy sensible a valores atípicos y pueden existir casos que sea imposible ajustarlo. Por eso es que se busca un balance entre mantener el margen lo más ancho posible con ojalá las menores violaciones posibles.





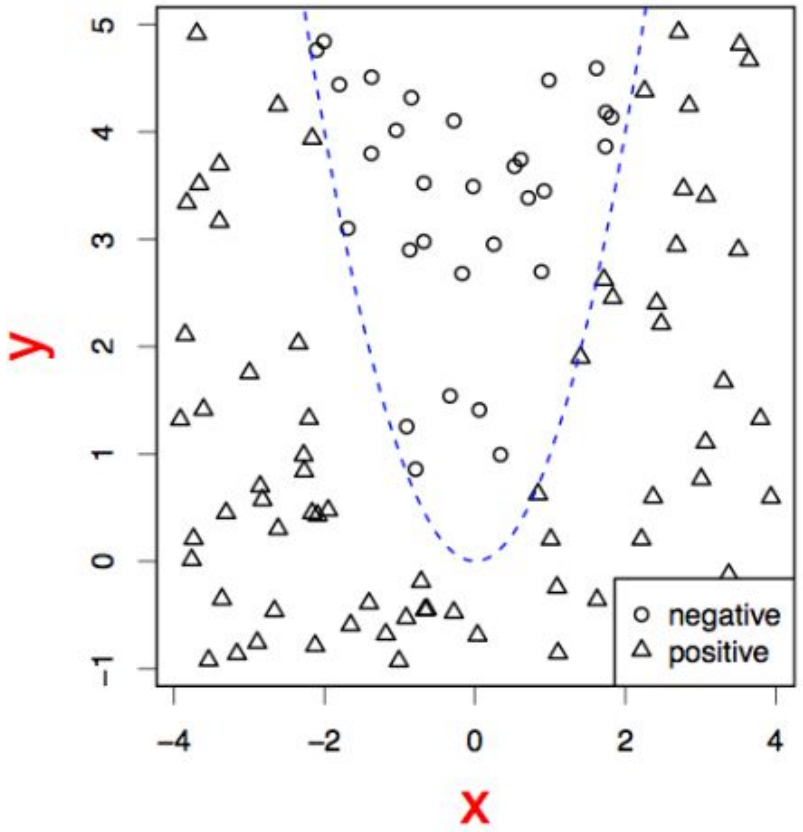
SVM no lineal

Por defecto las SVM sólo son capaces de separar clases con una línea, pero no siempre es posible realizar este tipo de separación. El truco del kernel representa los datos de una manera que es linealmente separable.



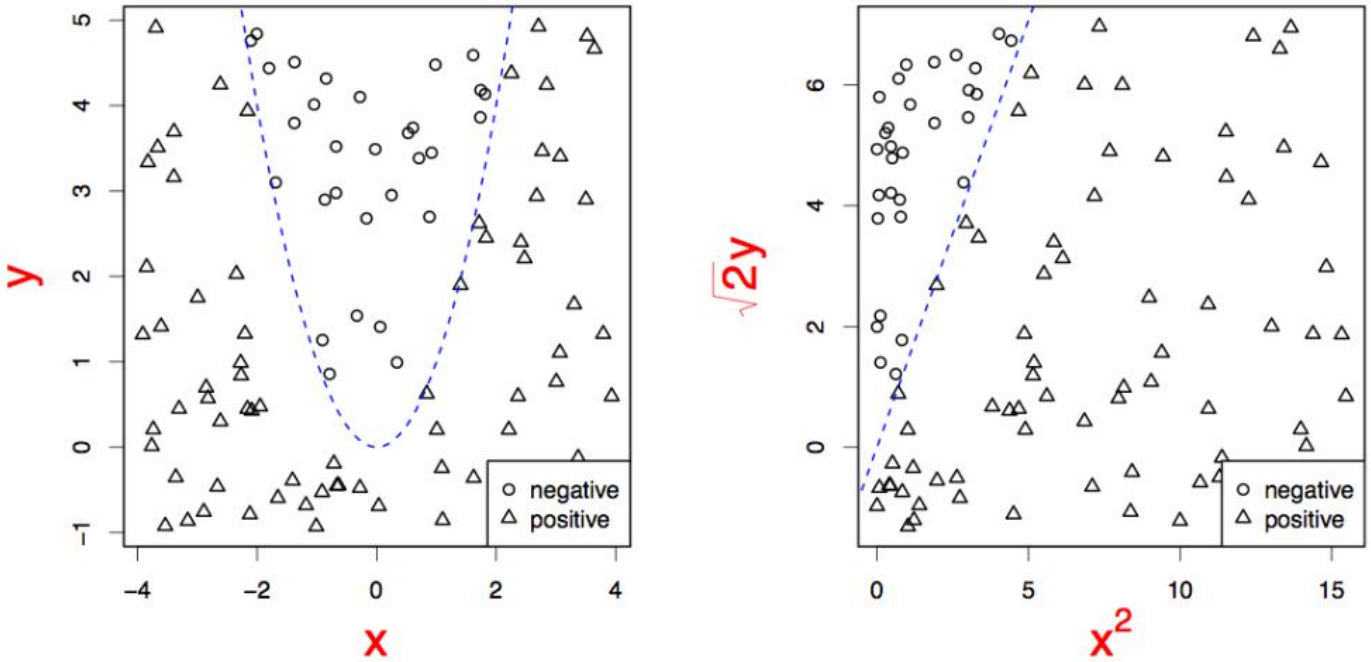
SVM no lineal

Una idea es encontrar una transformación no-lineal de los datos en un espacio de mayor dimensión, donde sí sean separables.



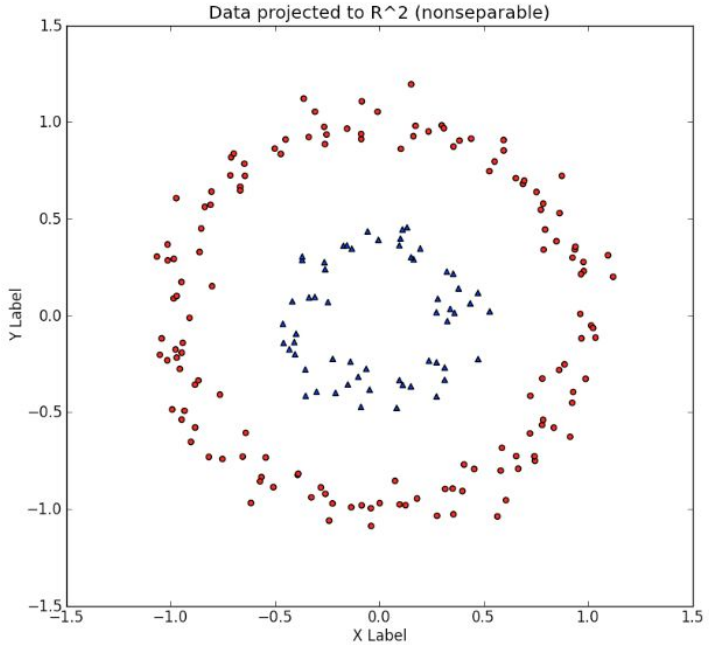
Ejemplo de transformación en un SVM no lineal

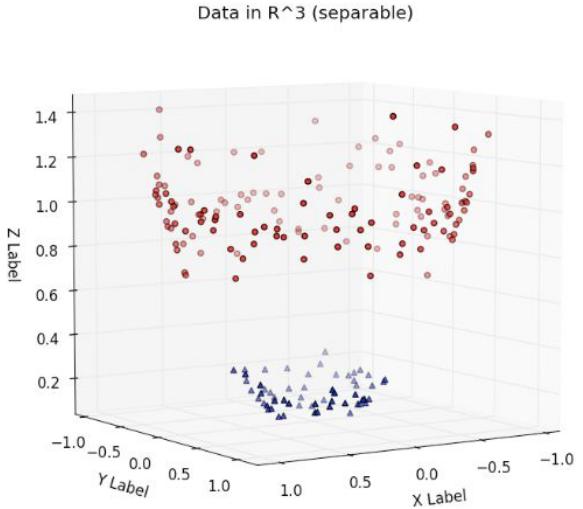
Transformar un punto (x, y) a $(x^2, (2y)^{0.5})$, donde el espacio sí es separable.



SVM no lineal

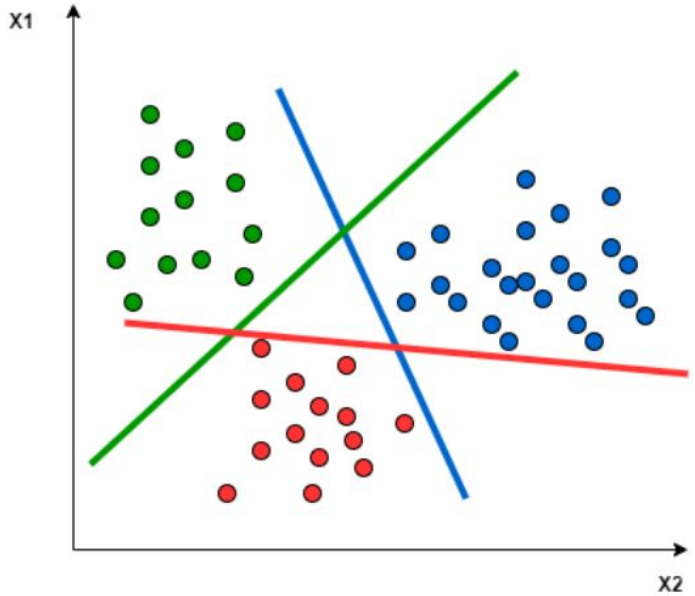
¿Cómo se calculó la tercera dimensión?





SVM multiclase

Un clasificador basado en SVM es un clasificador binario, pero muchos problemas se modelan como multiclase, en donde existen 2 o más clases en el conjunto de salida.



SVM multiclase

En un problema con `i` clases.

One versus all

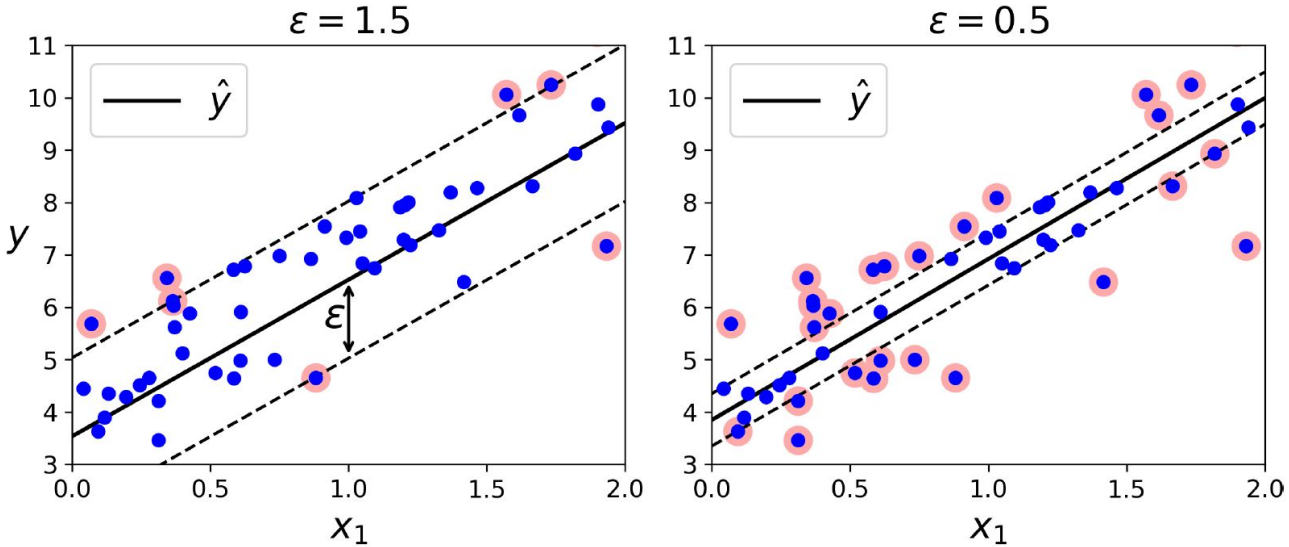
Se entrenan `i` clasificadores donde al clasificador k_i se le entregan todos los ejemplos de la clase i como positivos y el resto como negativos

Ove versus one

Se entrenan `i(i-1)/2` clasificadores y cada clasificador es entrenado con ejemplos pares de clases. La clasificación se realiza por una votación de todos los clasificadores

Regresión con SVM

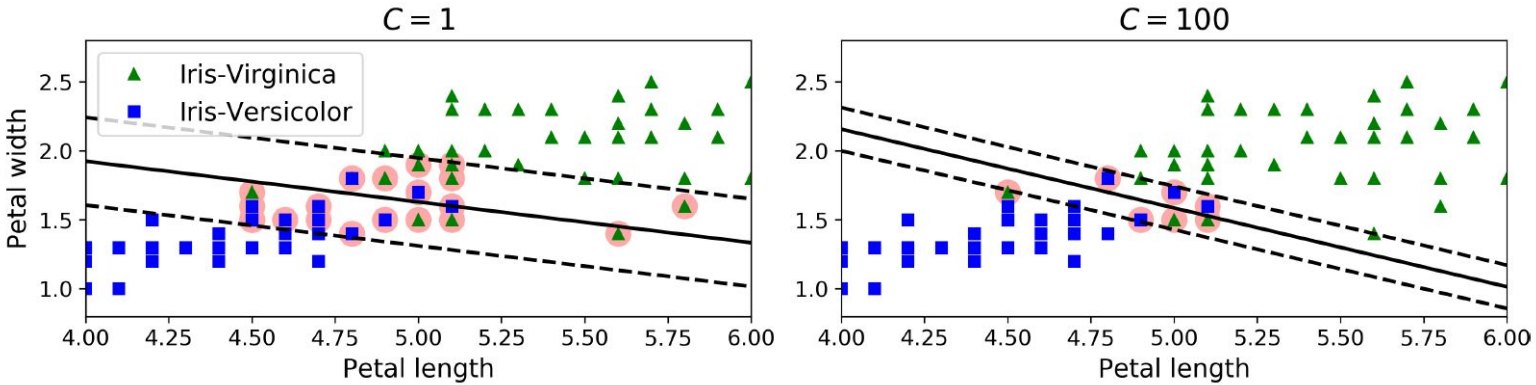
Para realizar regresión con SVM buscamos tener la mayor cantidad de ejemplos dentro del margen al mismo tiempo de disminuir la cantidad de ejemplos fuera del margen.



Hiperparámetros de las SVM

C: Este hiperparámetro controla el balance entre el tamaño del margen y la violación del margen.

Kernel: Podemos usar distintas distorsiones del espacio para generar una separación lineal de las clases.



Support Vector Machines

- El plano de separación SVM no depende de la inicialización(como MLP).
- Tiene menos parámetros libres (en la práctica se utilizan un número reducido de kernels)
- Rápido de entrenar (comparado a MLP).
- Funciona bien con 2 (o pocas) clases.
- Al tener múltiples clases SVM presenta problemas (zonas donde varios clasificadores entregan respuestas).