clusterAI
ciencia de datos en ingeniería industrial
UTN BA
curso I5521

clase_05: clasificación

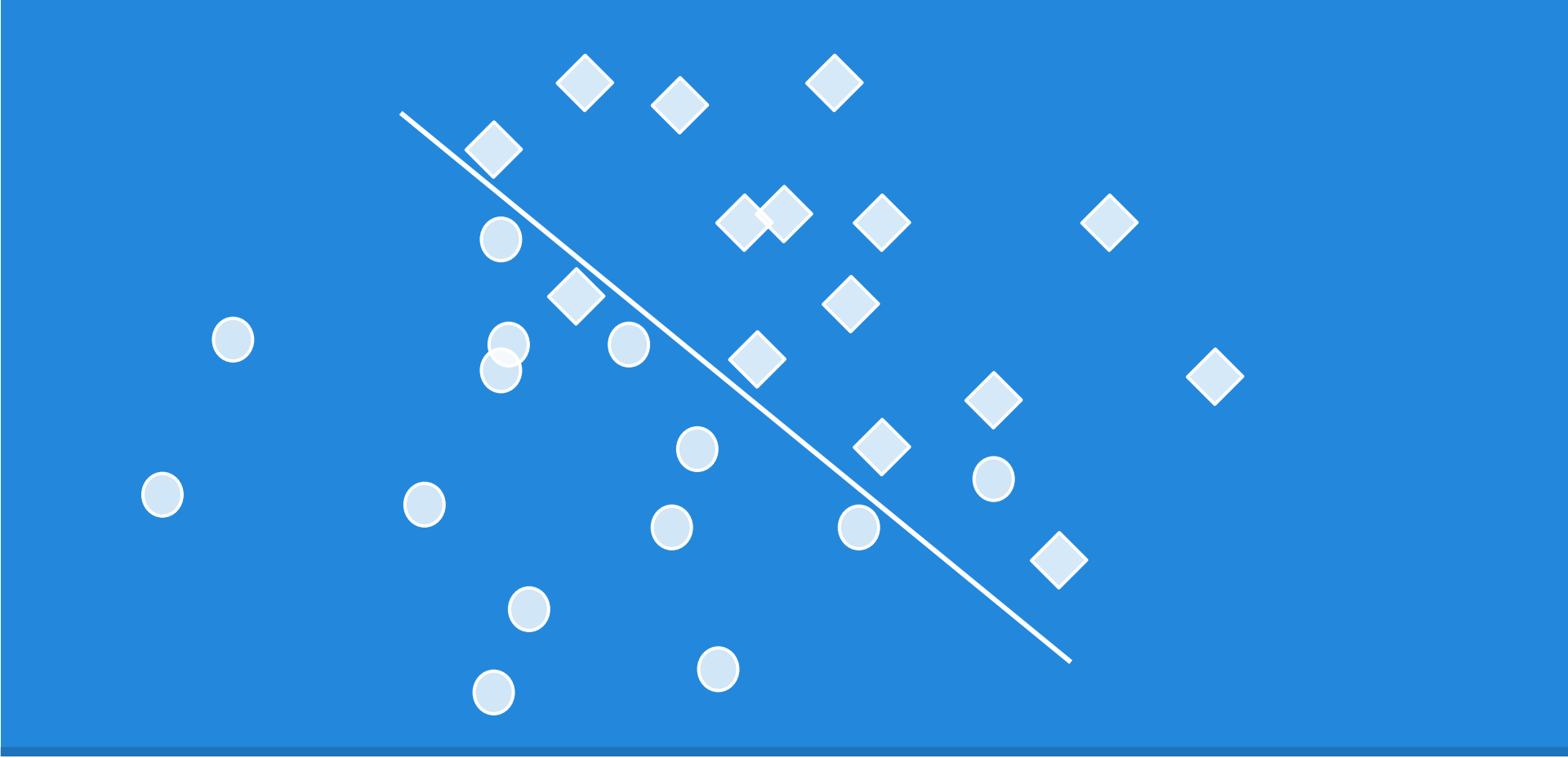
Docente: Martin Palazzo

AI & Art

* Anne Spalter https://cadaf.art/artificial-seascapes

agenda clase03: aprendizaje supervisado

- Clasificación
- Train-test split
- Cross validation
- Grid Search
- Performance metrics (Sens, Spec, ROC)
- Regularizacion
- Clasificadores: SVM, KNN, Logistic Regression



Aprendizaje supervisado: clasificación

Aprendizaje supervisado

$$S = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$$

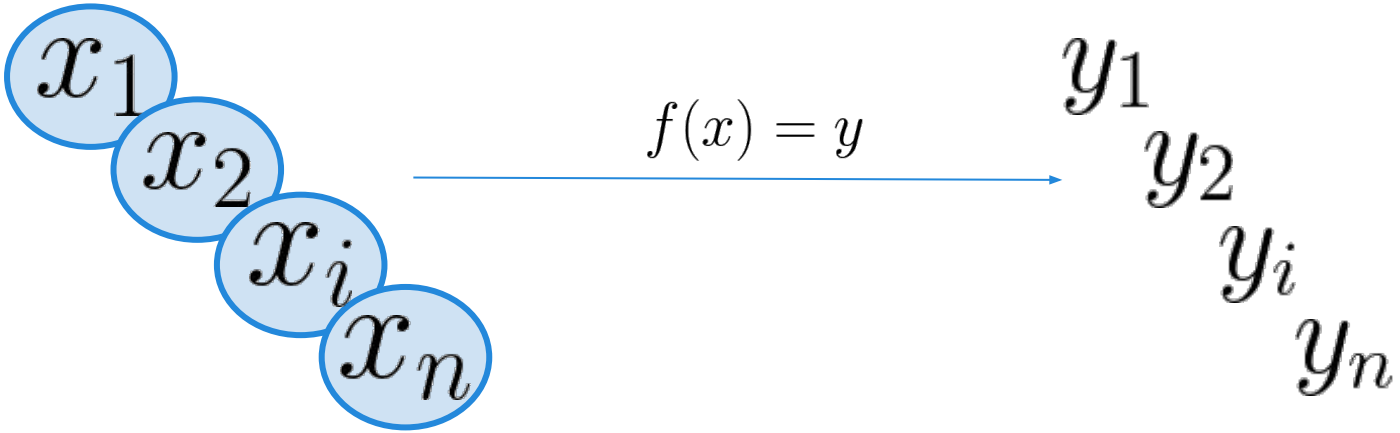
$$\mathcal{X} \in \mathbb{R}^d \qquad y \in \{-1, 1\} \qquad f(x) = y$$

El enfoque de aprendizaje supervisado se basa en disponer datos ordenados en un dataset **S** en pares de instancias y etiquetas (samples 'x' & labels 'y'). Las instancias son vectores d-dimensionales de variables aleatorias i.i.d. (independientes e idénticamente distribuidos). Las etiquetas se suponen variables dependientes que pueden tomar valores discretos (clases) o continuos a partir de distintos valores de x mediante una función f(x) llamada 'ground truth' o función objetivo tal que f(x) = y. Es decir que f(.) explica la relación entre 'x' (input) e 'y' (output) Como la realidad es compleja generalmente no conocemos la verdadera f(x), por lo que trataremos de aproximarla.

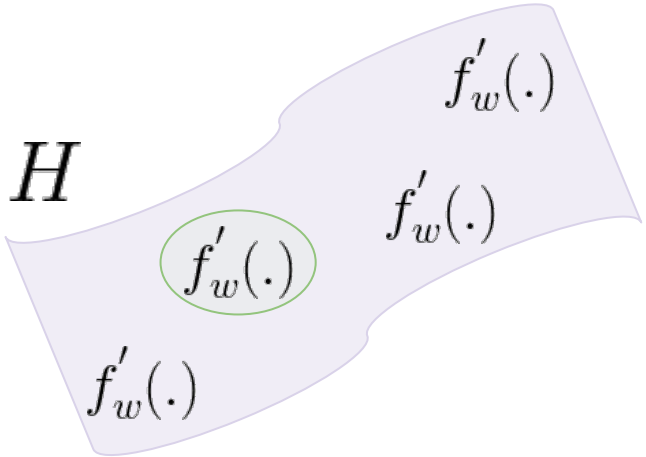
Aprendizaje supervisado

$$S = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$$

$$\mathcal{X} \in \mathbb{R}^d \qquad y \in \{-1, 1\} \qquad f(x) = y$$



Hipotesis



$$H = \left\{ f_w^1(.), f_w^2(.), \ldots, f'_w(.), \ldots \right\}$$

Para aproximarnos a la verdadera f(x) vamos a buscar alguna función f(.) dentro de un espacio de hipótesis que contiene muchas funciones f(.) . De todas las funciones disponibles dentro del espacio de hipótesis H vamos a tratar de encontrar alguna que explique lo mejor posible la relación entre el input 'x' y el output 'y'. La función que vayamos a buscar estará caracterizada por **parámetros (w)** que pueden tomar distintos valores. Entonces existirá una combinación de parámetros que determinen una f'(.) que se aproxime a la verdadera f(.) mas que otras f'(.) .

Aprendizaje supervisado

$$f(x) = y \qquad \hat{f}(x) = \hat{y} \qquad L(y, \hat{y})$$

Suponiendo que tanto el dataset de sample-features y las etiquetas están disponibles s=(x,y) vamos a **aprender** una función f'(x) que explique lo mejor posible la relación (x,y). Es decir aquella que aprenderemos una función que tomando como input las variables aleatorias "x" genere un output y' lo más similar a las etiquetas "y" dadas. Para poder medir cuán cerca están las etiquetas generadas por la función aprendida f'(x) utilizaremos una función L(y,y') de Costo o Pérdida (Loss function) que tomará valores altos cuando y sea muy distinto de y' . Por el contrario cuando y sea muy parecido a y' la función de costo tomará valores bajos. Por esta razón buscamos **minimizar** la función de costo.

Aprendizaje supervisado

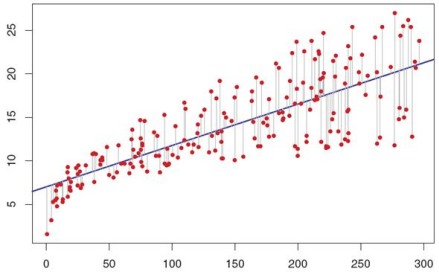
$$L(y, \hat{y})$$

$$\hat{f}(x) = \hat{y}$$

El aprendizaje de los parámetros w para definir una función f(x) durante la busqueda de una frontera de decision se realiza con metodos de optimizacion. Se busca minimizar la funcion objetivo de costo L utilizando las variables de decision w, los parametros de la funcion f(x).

Métodos de aprendizaje supervisado

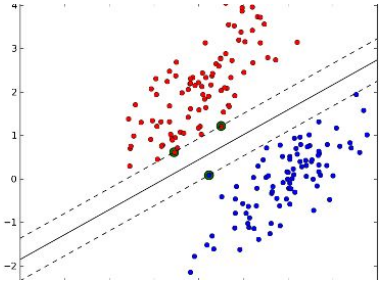
regression



Y es continua

$$y \subseteq \mathbb{R}$$

classification

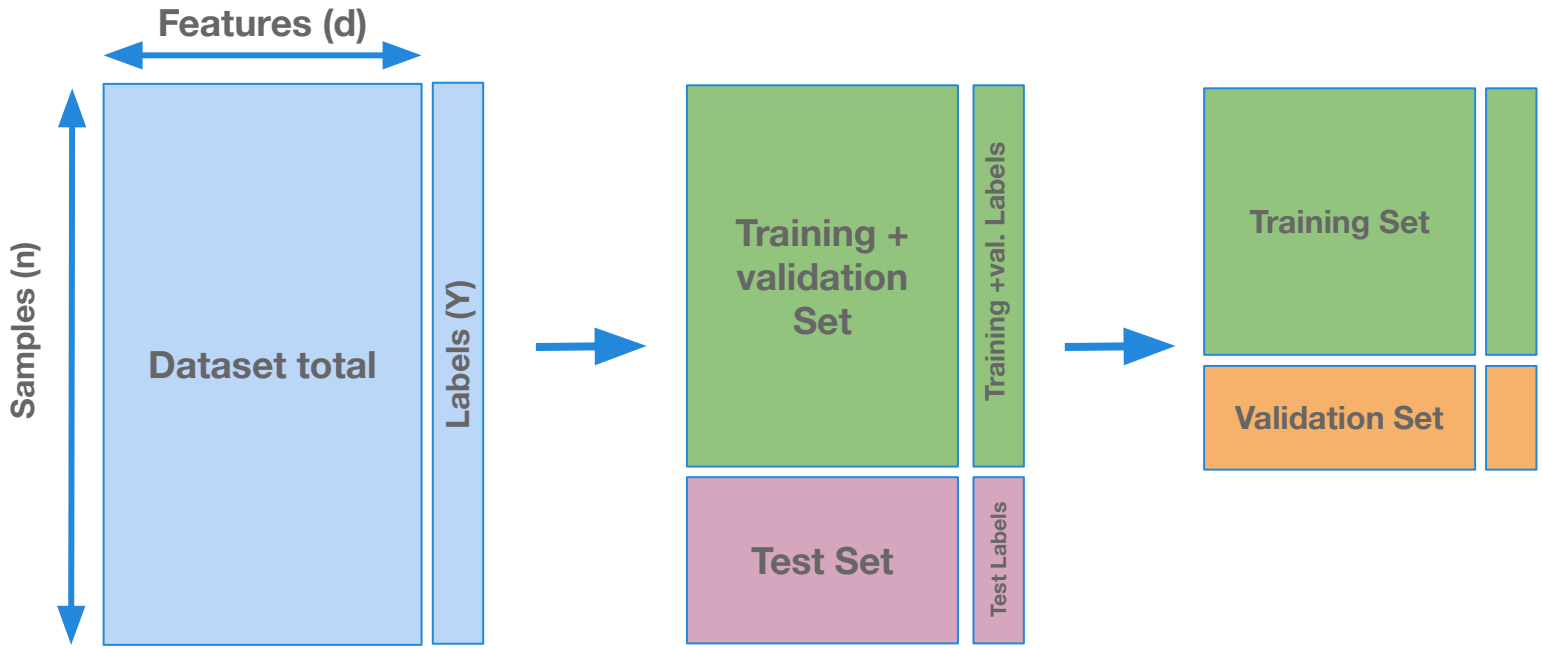


Y es categorica

$$y \in \{-1, 1\}$$

Existen dos enfoques importantes en el aprendizaje supervisado: clasificación y regresión. Cuando las etiquetas toman valores categóricos hablamos de clasificación. Cuando las etiquetas toman valores continuos hablamos de regresión.

Train, Validation, Test sets.

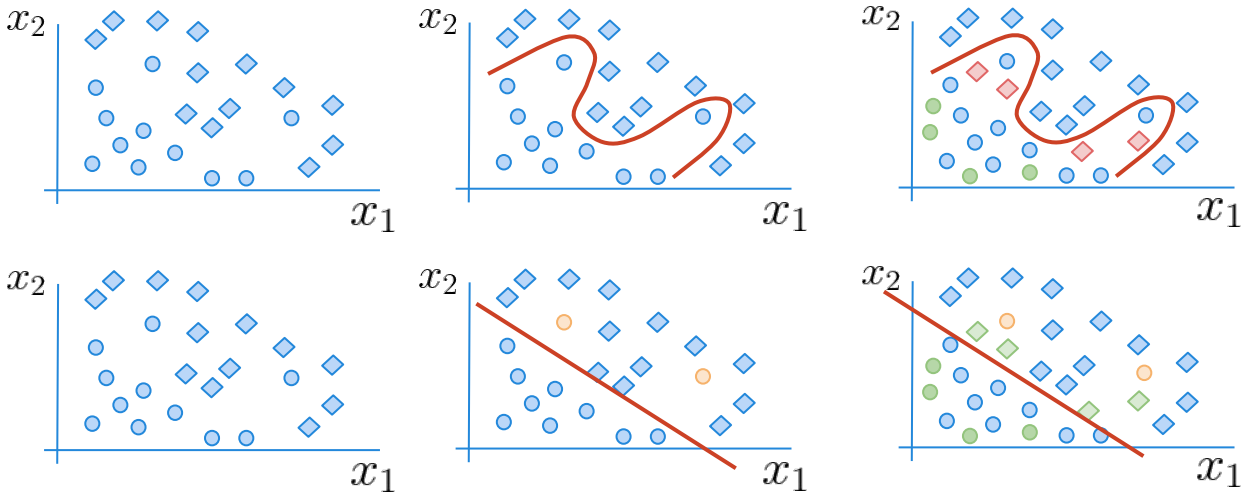


El clasificador aprenderá la regla de decisión utilizando el train set (samples + labels). Luego clasificará las muestras de test (sin mirar las labels de test) y se medirá la exactitud de clasificación en test.

Aprendizaje supervisado: clasificación

Modelo no-lineal

Modelo lineal



Datos de entrenamiento con etiquetas de dos clases.

Funcion de clasificacion: entrenamos un modelo no-lineal y lineal.

Agregamos 4 muestras nuevas de cada clase y vemos cómo clasifican con el modelo entrenado.

Inicialmente el clasificador estará expuesto a muchas muestras de “entrenamiento” y sus respectivas etiquetas de manera tal de que aprenda los parametros **w** que definen una función f’(x)=y’ cuya frontera **minimiza el error de clasificación L(y,y’)**. Simultáneamente esperamos que luego de entrenar, la regla/función de decisión aprendida en los datos de entrenamiento clasifique bien muestras que nunca vió provenientes de la misma distribución/poblacion (test). Es decir, que pueda **generalizar** a instancias ‘x’ nunca vistas clasificándolas correctamente.

En la figura: cuál modelo tiene menor error en el entrenamiento? cual modelo generaliza mejor? Cual es más complejo? Cual es más sencillo?

Frontera de decision lineal

Hiper-plano separador $$f(x) = w^T x + b = 0$$

Funcion de decision $$D(x) = \text{sign}[w^T x + b]$$

Existen muchos tipos de funciones de decisión. La familia de funciones más conocida es la de las funciones lineales. Estas funciones son hiper-planos caracterizados por parámetros **w** (vector w=[w1….wd]) que determinarán cómo se posiciona la frontera de decisión en el hiper-espacio de dimensión d. En la clasificación binaria la función de decisión asignará un valor de **y=1** o **y=-1** según de que lado del hiper-plano se posicionen las muestras **x**.

Variance vs Bias

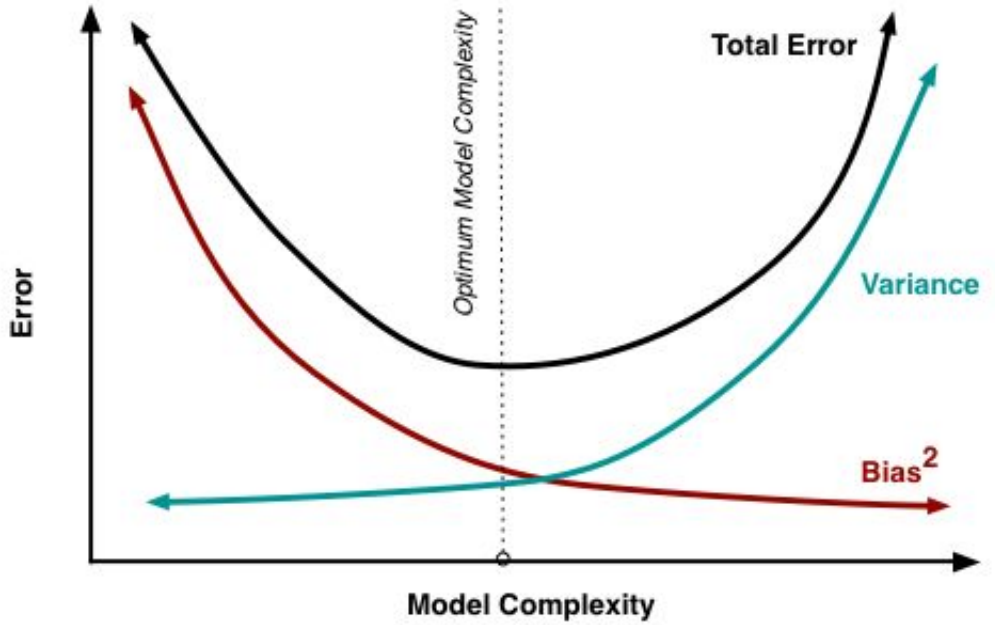
$$\text{Error} = \text{Var}(\hat{f}(x)) + [\text{bias}(\hat{f}(x))]^2$$

Cuando la etiqueta real **y** es distinta a la etiqueta asignada por el clasificador **y'** la función de pérdida-costo (Loss) **L** se incrementará. La causa de que un modelo supervisado tenga error en muestras nuevas se puede causar por su sesgo y/o por su varianza.

Un modelo tiene alta varianza cuando su predicción varía mucho si lo exponemos a muestras distintas provenientes de la misma población. En general la varianza se observa en modelos complejos (polinomios de alto grado), y está asociada al 'sobre-ajuste' (overfitting).

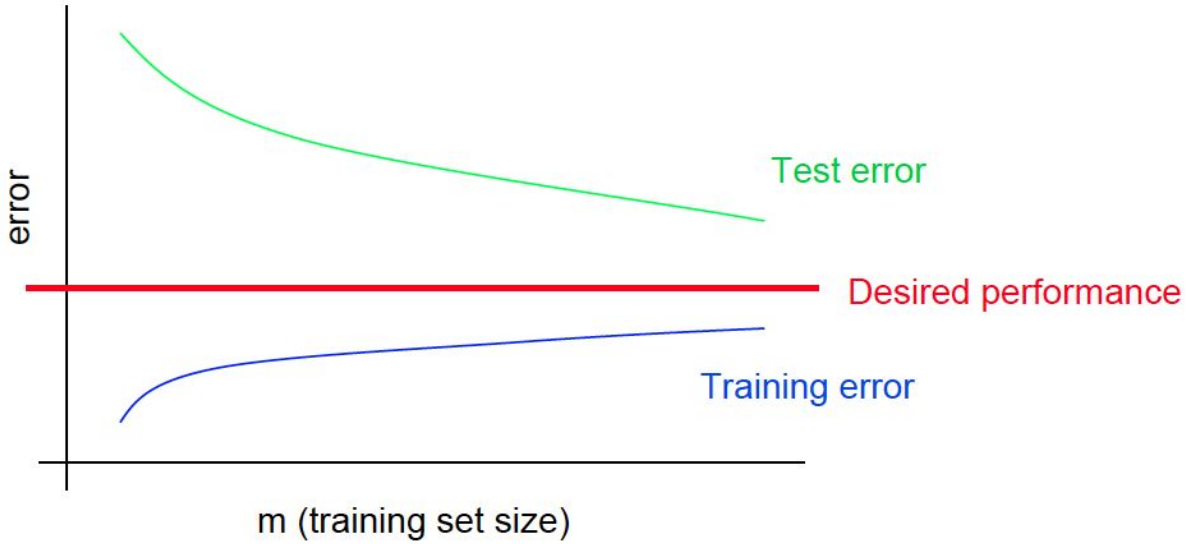
Los modelos con alto sesgo son aquellos que frente a distintos datos provenientes de la misma población no modifican mucho su predicción aunque en general ese valor de predicción no suele ser aceptable. El sesgo está asociado a 'sub-ajuste' (underfitting) y se suele observar en modelos muy sencillos (por ejemplo modelos lineales).

Variance vs Bias



*Andrew Ng.

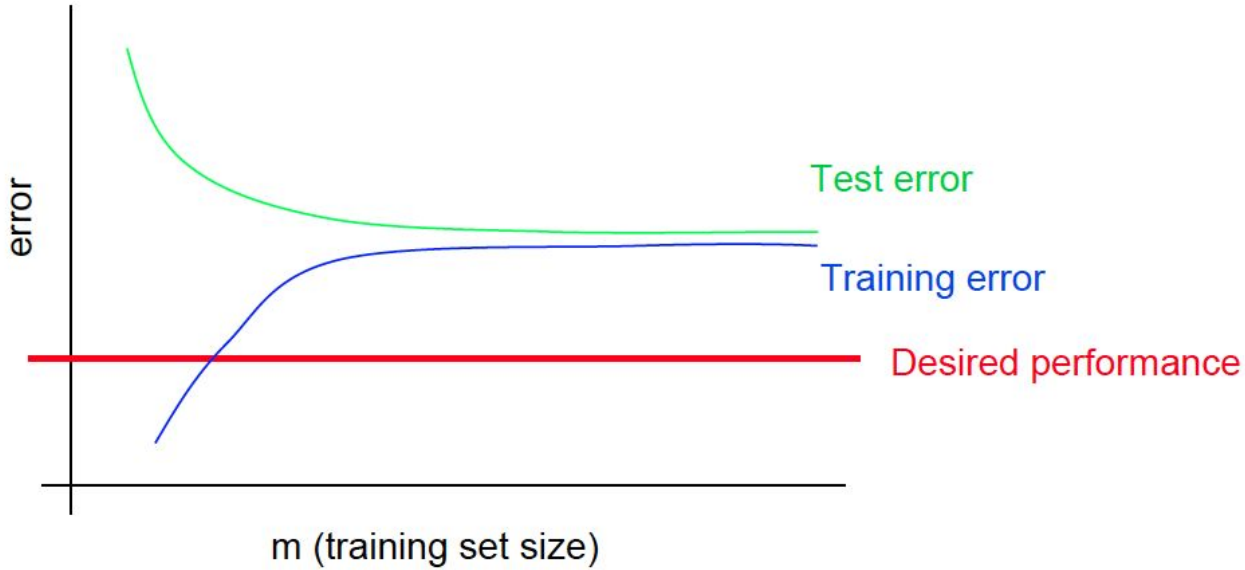
Learning curve: high variance



Higher model complexity might be affected by Variance

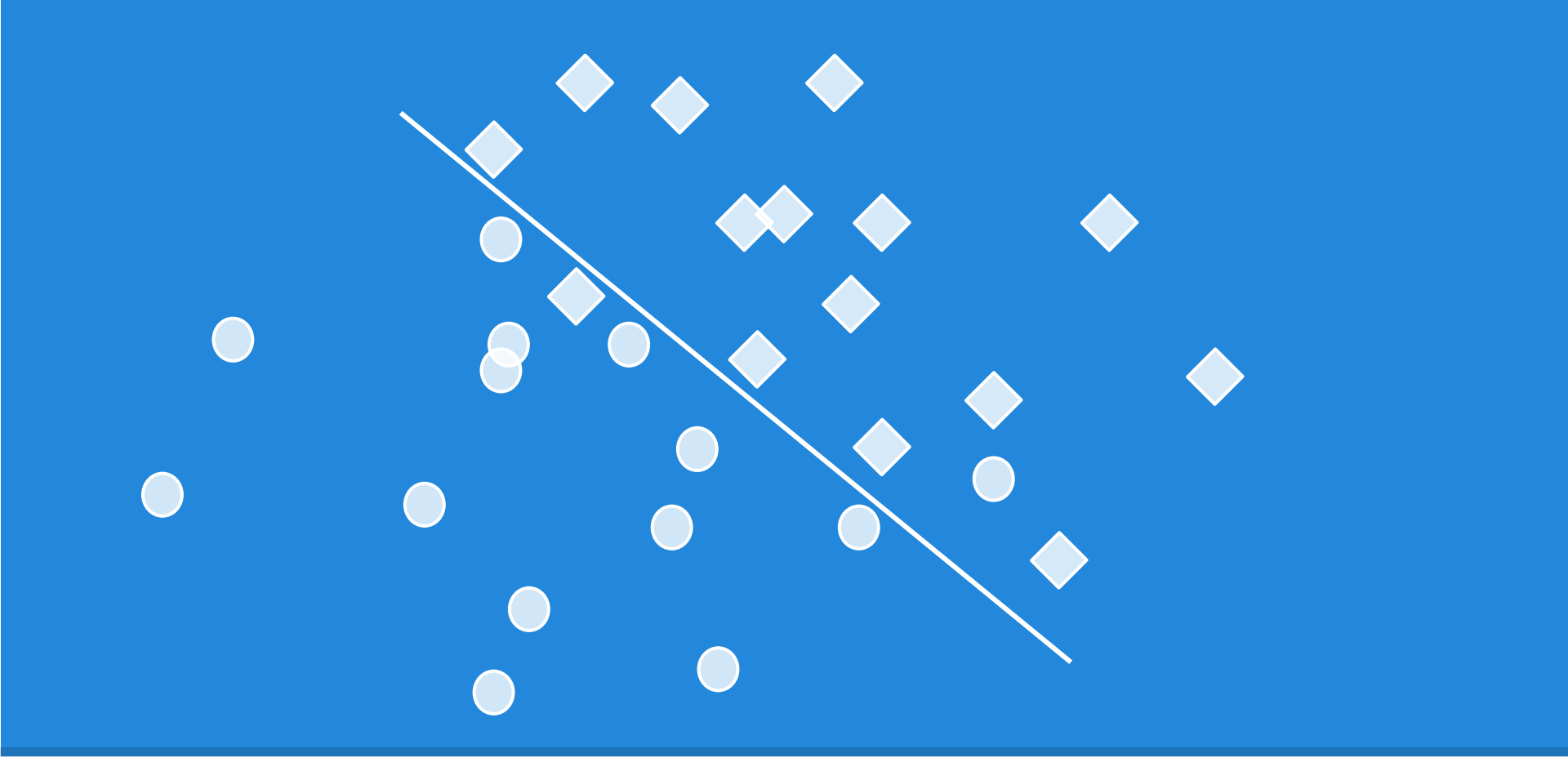
*Andrew Ng.

Learning curve: high bias



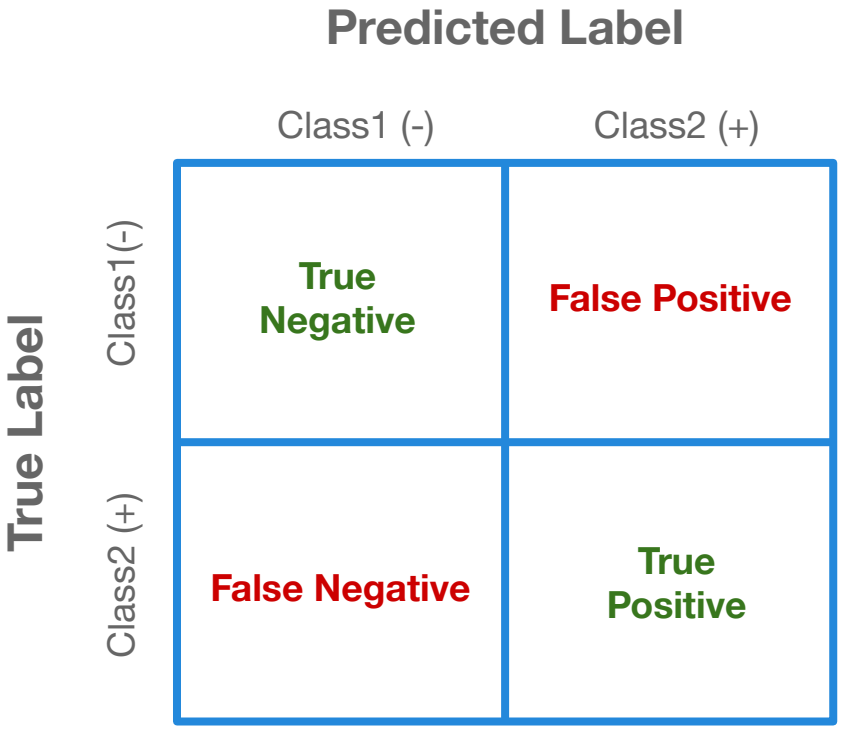
Simpler models might be exposed to Bias

*Andrew Ng.



Aprendizaje supervisado: mediciones de desempeño/performance en clasificación

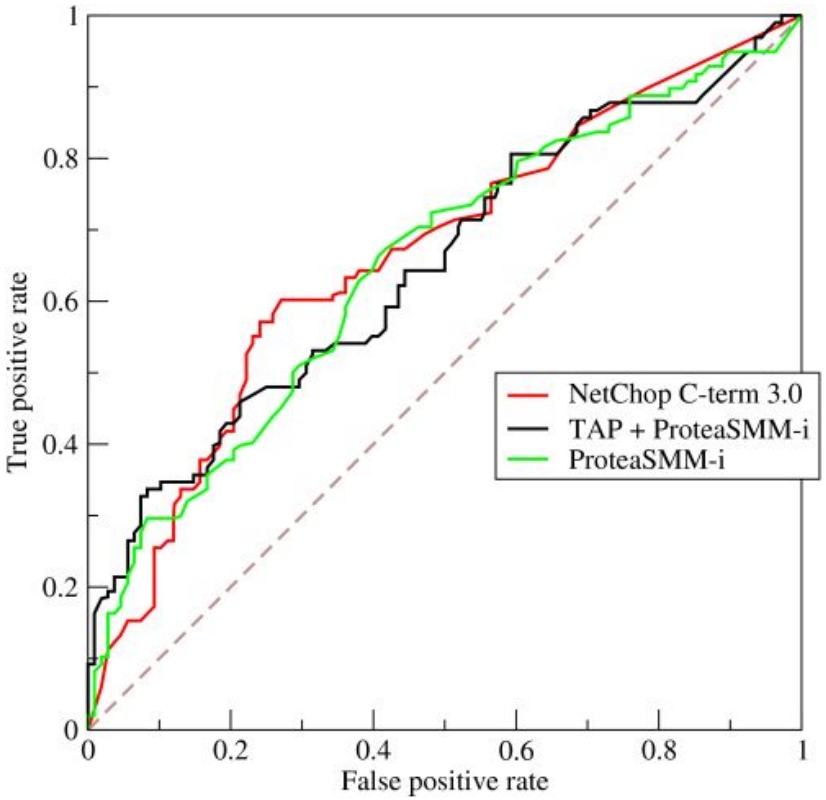
classification results: confusion matrix



Accuracy = (TN+TP) / Total
Sensitivity (recall) = TP/(TP+FN)
Specificity = TN/(TN+FP)

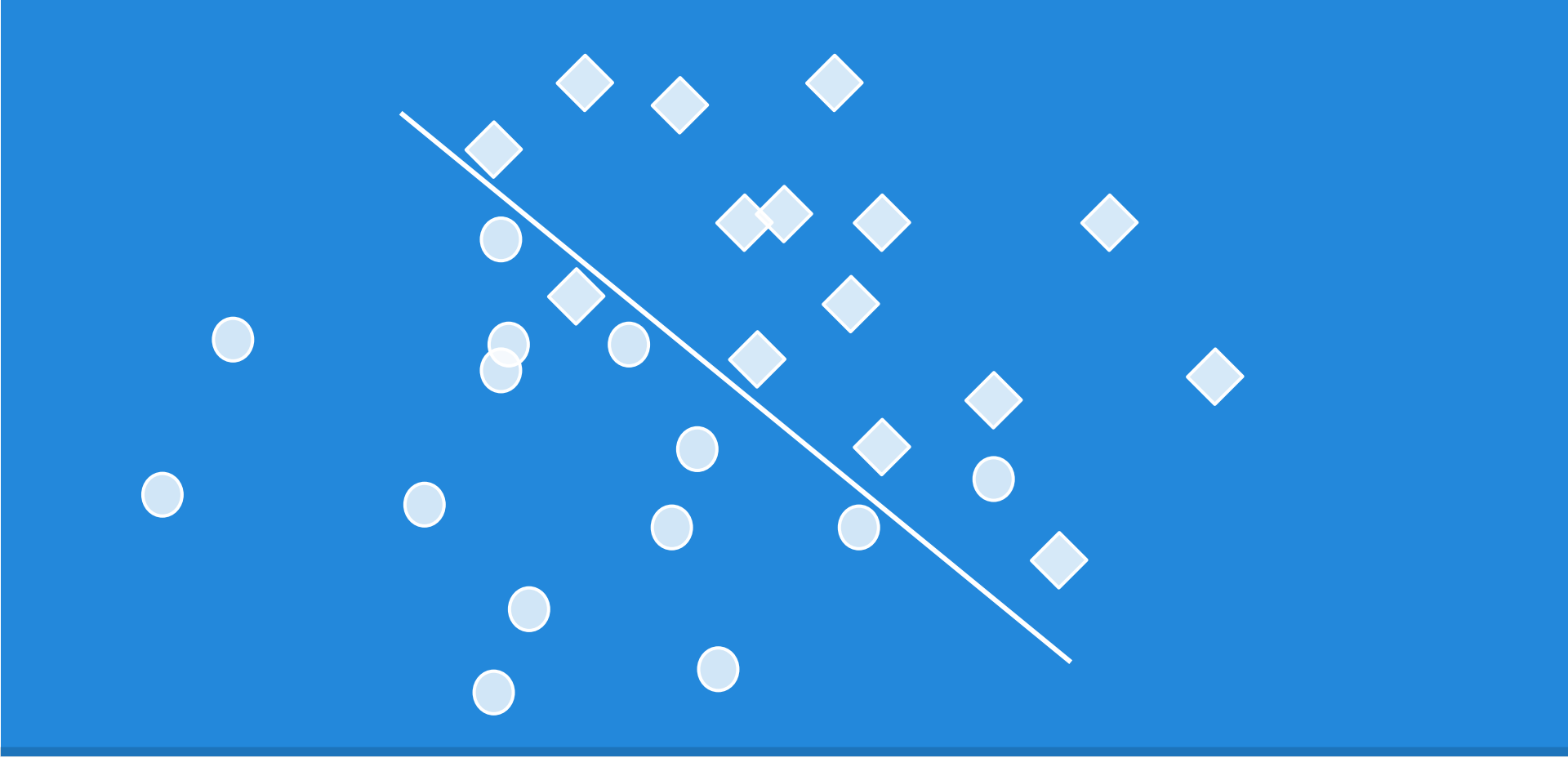
La matriz de confusión es un elemento para evaluar los resultados de la clasificación. En cada posición se cuentan los TP, TN, FP, FN. Luego se obtienen los coeficientes de accuracy, Sensitivity, specificity.

classification results: AUC ROC



- El área bajo la curva ROC (AUC) da una idea de cuan bueno es mi clasificador independientemente del accuracy.
- Tomando una sola clase, se analizan distintos umbrales de clasificación y se contempla la relaciòn entre TP y FP.
- Cuanto más cerca de 0.5 sea el AUC, más similar a “arrojar una moneda” sera mi clasificador. Cuanto más cerca de 1 sea el área bajo la curva, mejor mi clasificador.

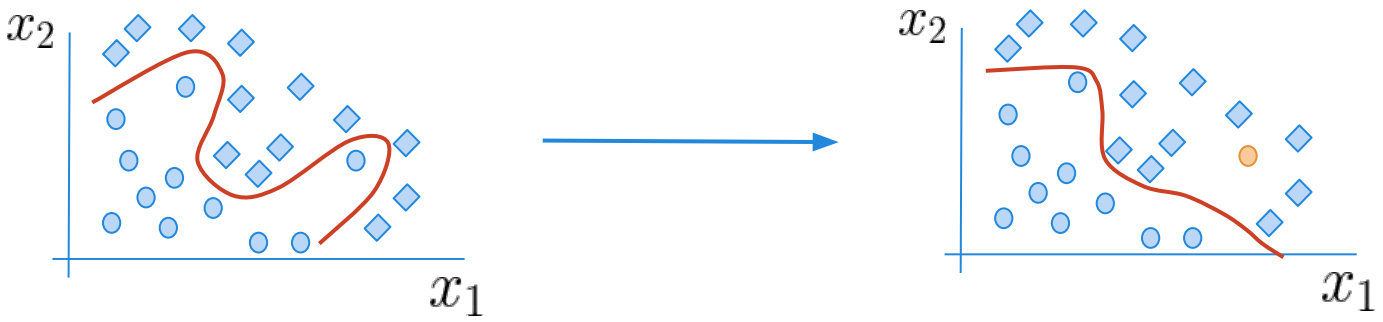
* https://en.wikipedia.org/wiki/Receiver_operating_characteristic#/media/File:Roccurves.png



Aprendizaje supervisado: Regularizacion

Regularizacion

La regularización es un enfoque diseñado para mejorar la generalización de un modelo de aprendizaje supervisado y evitar el sobreajuste (overfitting).

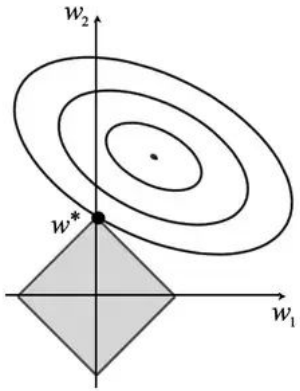


En muchos casos la regularización se asocia a **penalizar** al modelo por su complejidad obligándolo a no ajustarse completamente a los datos de entrenamiento. La regularización puede ser considerada un supuesto donde los datos ruidosos o muy complejos se asocian al ruido. Es decir que regularizar es suponer que la frontera de decisión no es tan compleja y esa suposición se la impone en el modelo como una penalización a la complejidad. Por el contrario una regularización excesiva puede generar un modelo demasiado simple que sub-ajuste a los datos. Por medio de la validación cruzada (cross validation) se tratará de encontrar un nivel de regularización aceptable.

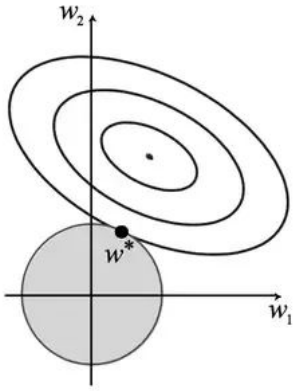
Regularización de la función de decisión

$$\min_w L(y, \hat{y}) = \min_w L(y, \hat{f}(x))$$

$$f(x) = w_1 x_1 + w_2 x_2$$



L1



L2

Regularización con norma L2 $\|\mathbf{w}\|_2 = (|w_1|^2 + |w_2|^2)^{1/2}$ $\|\mathbf{w}\|_2 < C$

Regularización con norma L1 $\|\mathbf{w}\|_1 = |w_1| + |w_2|$ $\|\mathbf{w}\|_1 < C$

Regularización en LR: L1 & L2

Norma L1 : Lasso

$$\|\mathbf{w}\|_1 = |w_1| + |w_2| + \cdots + |w_d|$$

Si le imponemos una restricción/penalización a la norma L1 de un vector lo que sucederá es que muchos valores de w tomarán el valor wi = 0 para poder satisfacer la restricción y solo unos pocos serán distinto de cero.

$$\|\mathbf{w}\|_1 < C$$

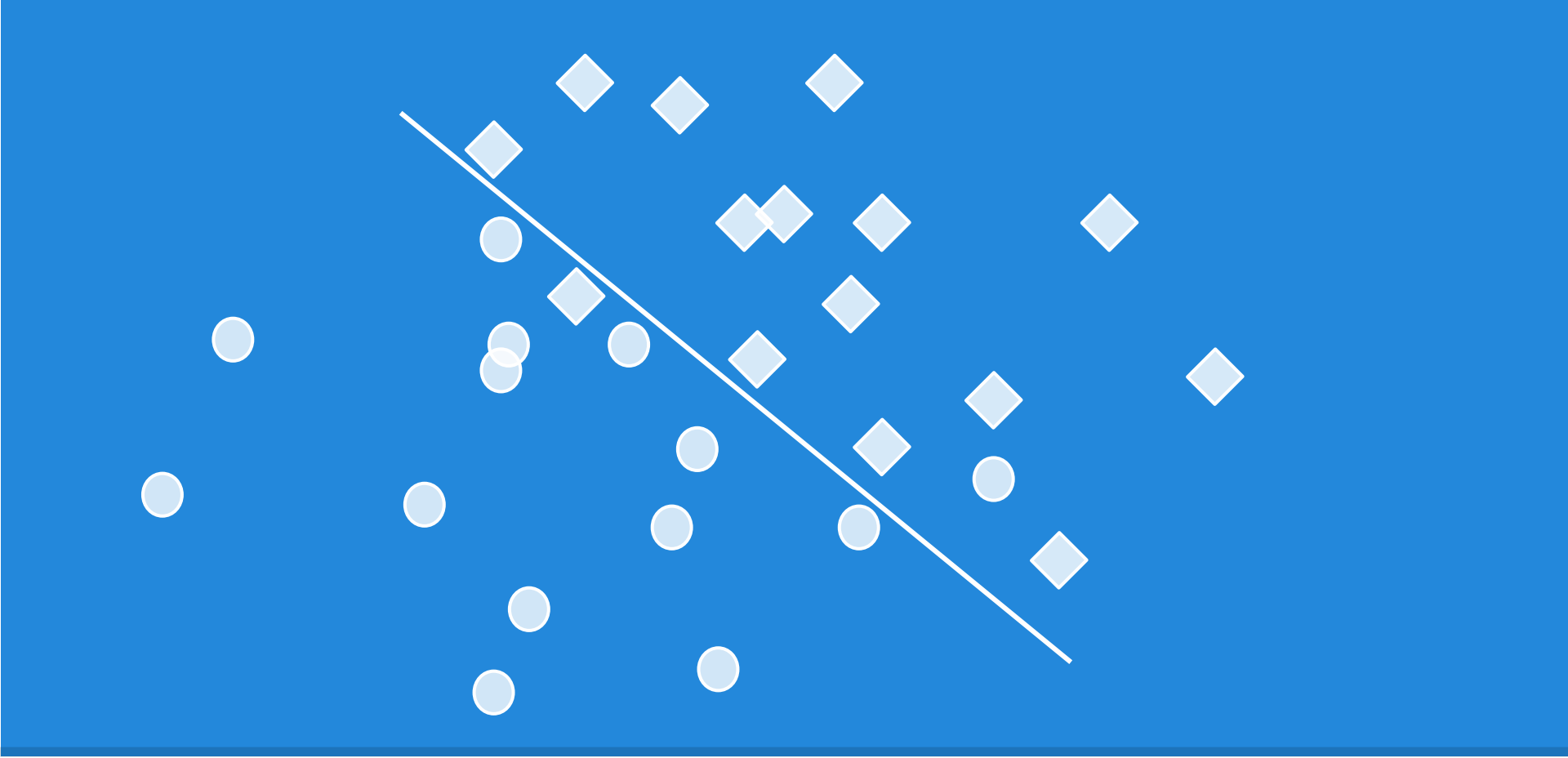
Norma L2 : Ridge

$$\|\mathbf{w}\|_2 = (|w_1|^2 + |w_2|^2 + \cdots + |w_d|^2)^{1/2}$$

Si le imponemos una restricción/penalización a la norma L2 de un vector lo que sucederá es que muchos valores de w tomarán valores cercanos a cero para poder satisfacer la restricción y sólo unos pocos tendrán valores absolutos grandes.

$$\|\mathbf{w}\|_2 < C$$

regularization



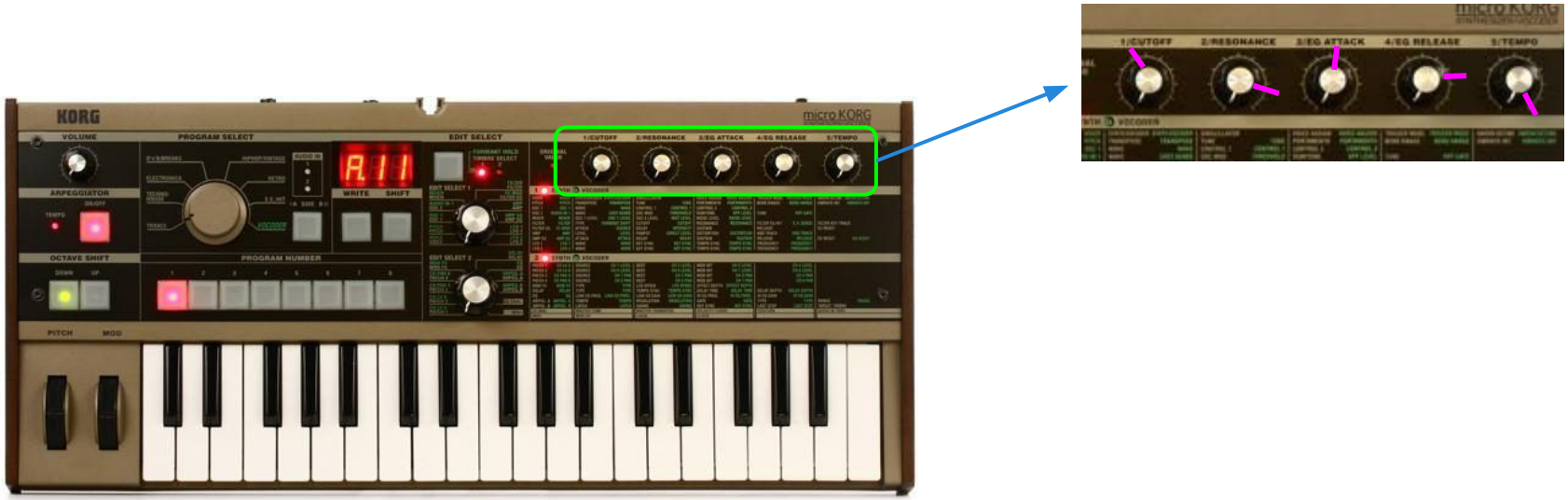
Aprendizaje supervisado: cross validation & hyperparameter tuning

Hiper-parámetros

$$f_{w,\lambda}(x) \quad \text{funcion de decision}$$
$$w \quad \text{parametros}$$
$$\lambda \quad \text{hiper-parametros}$$

Los modelos están caracterizados por parámetros **w** que son aprendidos durante el entrenamiento al ser expuestos a los datos. Adicionalmente los clasificadores tienen hiper-parámetros que definen entre otras cosas la familia de funciones que se pueden aprender. Por ejemplo, un hiper-parámetro podría ser el grado de una función polinomial. Los hiper-parámetros no son aprendidos por un algoritmo, son prefijados por el usuario. Los hiperparametros son útiles para poder determinar la complejidad y flexibilidad del clasificador. Por medio de una técnica llamada validación cruzada (cross validation) determinaremos cual la configuración de los hiper parámetros que minimiza el error de clasificación.

hyperparameters



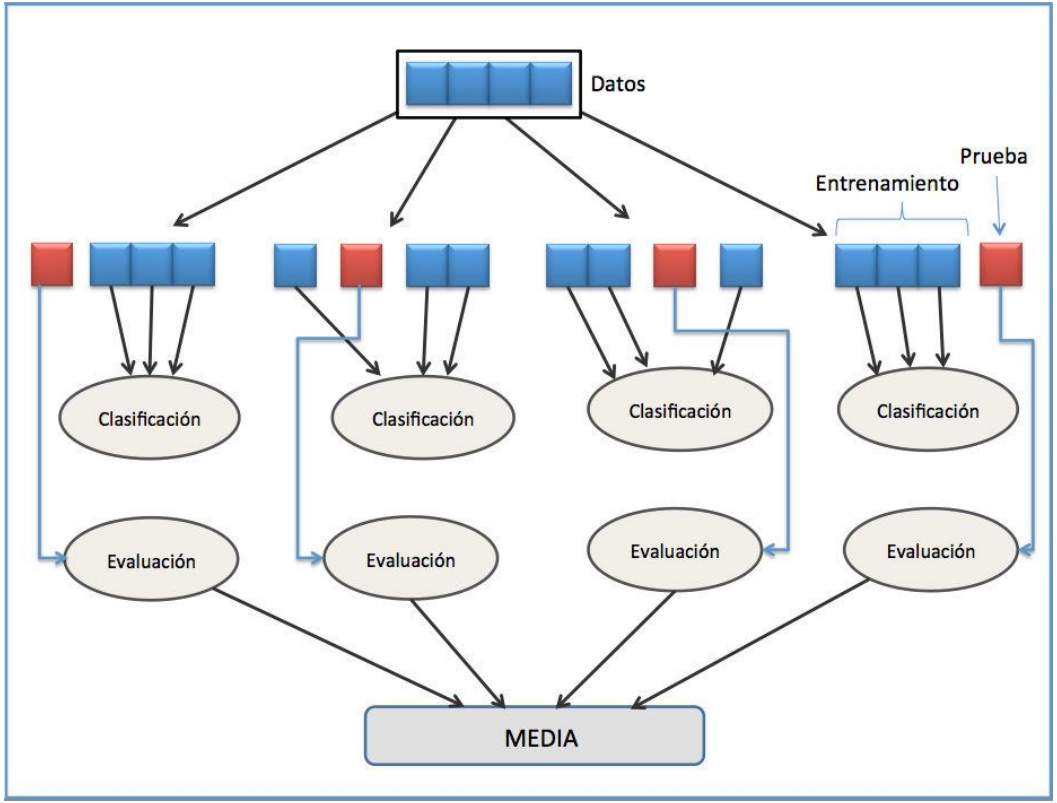
Los hiperparametros son definidos por el usuario y no son determinados durante el aprendizaje interno del modelo. Para encontrar una combinación de hiperparametros que permita un entrenamiento con resultados aceptables buscaremos en una grilla de valores de hiperparametros. Nos quedaremos con la combinación de la grilla que menor media de error tenga luego de K iteraciones de validación cruzada (cross validation).

Cross - Validation en training set

"**Cross validation**" (CV) se realiza con las muestras de entrenamiento. Consiste en dividir nuestro training set en **K folds** (K porciones) e iterar K veces.

En cada iteración, una porción se utiliza como validación independiente y el resto como train. En cada iteración se entrena un modelo con train y se evaluará el resultado de clasificación con validación. Luego se realizará un promedio de la exactitud de clasificación de las k iteraciones.

Cross validation sirve para poder estimar el error estadísticamente lo que es equivalente tratar de entender cómo se comportaría frente a muestras nuevas. Además los valores de los hiperparametros del modelo pueden determinarse por cross validation y se preserva el hiper parámetro que menor error promedio de cross validation genere.



Cross Validation

	$\delta_1 = 1$	$\delta_2 = 10$	$\delta_3 = 100$	$\delta_4 = 1000$
$\lambda_1 = 0.01$	Acc^i_{11}	Acc^i_{12}	Acc^i_{13}	Acc^i_{14}
$\lambda_2 = 0.1$	Acc^i_{21}	Acc^i_{22}	Acc^i_{23}	Acc^i_{24}
$\lambda_3 = 1$	Acc^i_{31}	Acc^i_{32}	Acc^i_{33}	Acc^i_{34}

Fold = 0

	$\delta_1 = 1$	$\delta_2 = 10$	$\delta_3 = 100$	$\delta_4 = 1000$
$\lambda_1 = 0.01$	Acc^i_{11}	Acc^i_{12}	Acc^i_{13}	Acc^i_{14}
$\lambda_2 = 0.1$	Acc^i_{21}	Acc^i_{22}	Acc^i_{23}	Acc^i_{24}
$\lambda_3 = 1$	Acc^i_{31}	Acc^i_{32}	Acc^i_{33}	Acc^i_{34}

Fold = 1

	$\delta_1 = 1$	$\delta_2 = 10$	$\delta_3 = 100$	$\delta_4 = 1000$
$\lambda_1 = 0.01$	Acc^i_{11}	Acc^i_{12}	Acc^i_{13}	Acc^i_{14}
$\lambda_2 = 0.1$	Acc^i_{21}	Acc^i_{22}	Acc^i_{23}	Acc^i_{24}
$\lambda_3 = 1$	Acc^i_{31}	Acc^i_{32}	Acc^i_{33}	Acc^i_{34}

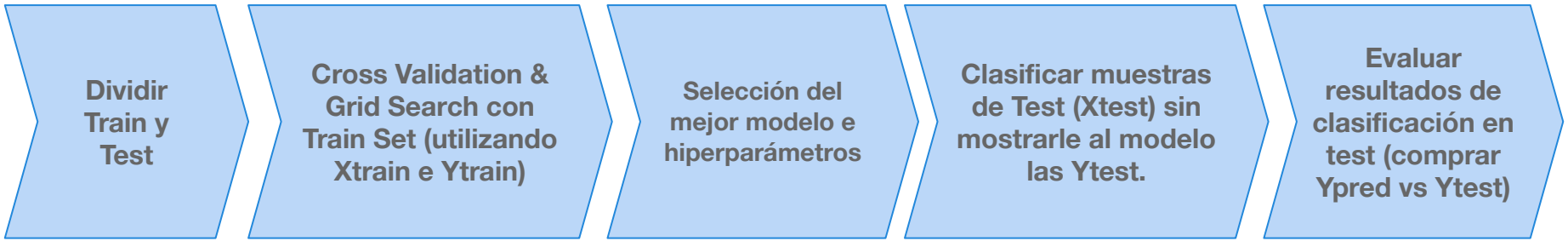
Fold = 2

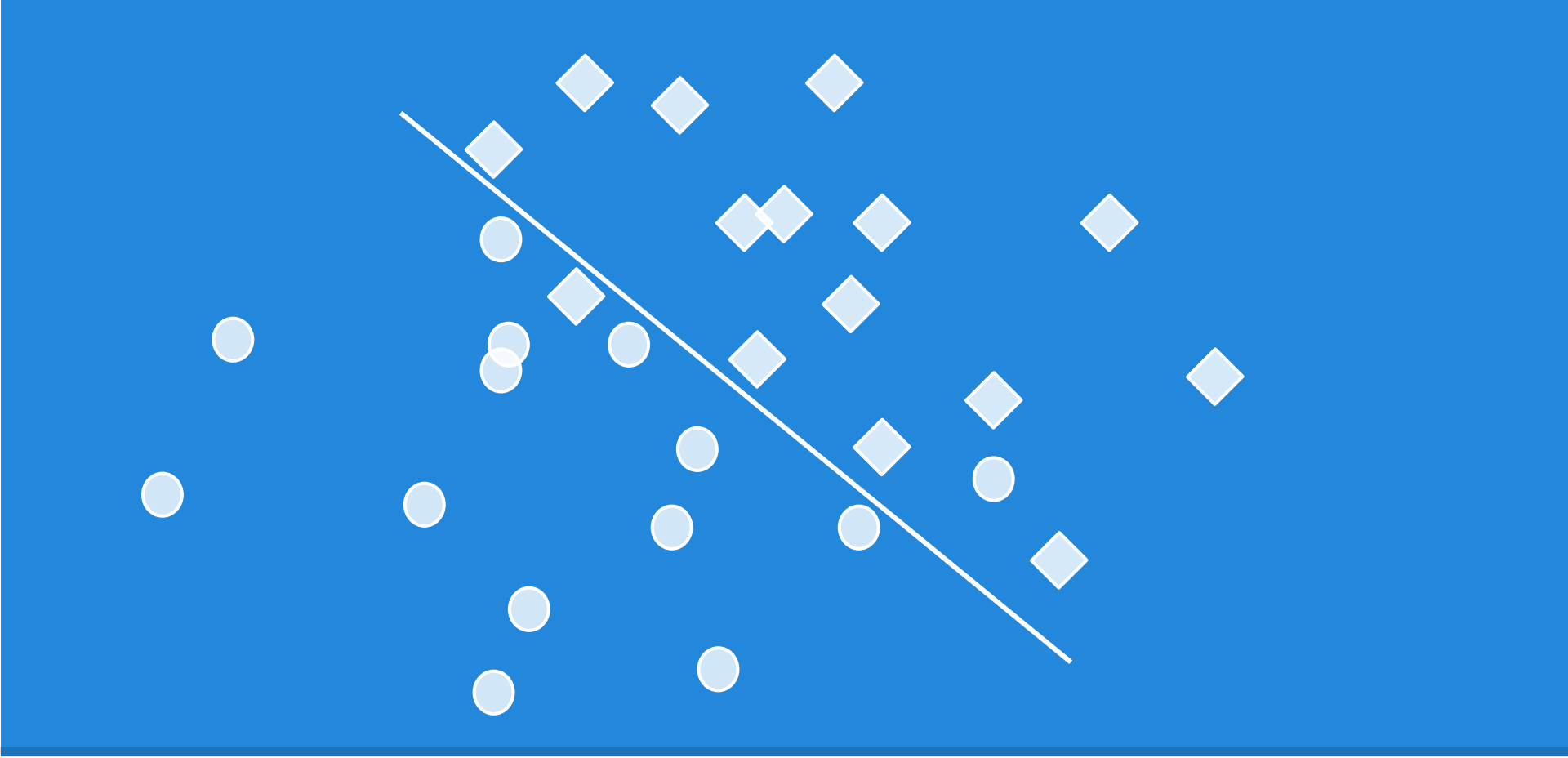
	$\delta_1 = 1$	$\delta_2 = 10$	$\delta_3 = 100$	$\delta_4 = 1000$
$\lambda_1 = 0.01$	Acc^i_{11}	Acc^i_{12}	Acc^i_{13}	Acc^i_{14}
$\lambda_2 = 0.1$	Acc^i_{21}	Acc^i_{22}	Acc^i_{23}	Acc^i_{24}
$\lambda_3 = 1$	Acc^i_{31}	Acc^i_{32}	Acc^i_{33}	Acc^i_{34}

Fold = 3

Supongamos que tenemos dos hiperparametros y una grilla de valores para cada uno. Para cada combinación de hiper-parámetros se entrena un modelo y se calcula si exactitud (Accuracy). Luego se calcula la media de los resultados de cada una de las particiones de cross-validation y se selecciona la combinación que mejor resultado obtiene.

Pipeline: Train, Validate, Test Model





Aprendizaje supervisado, clasificación: Support Vector Machines

Clasificadores

Vamos a estudiar tres modelos de clasificación

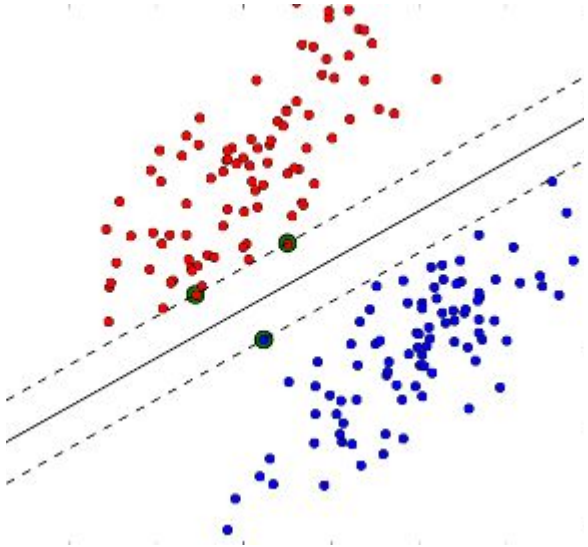
- Support Vector Machines
- K-Nearest Neighbors
- Logistic Regression

Cada uno tendrá ventajas y debilidades respecto a los otros. Hay decenas de clasificadores que por razones de tiempo no incluimos en este curso.

Support Vector Machines

Support Vector Machines

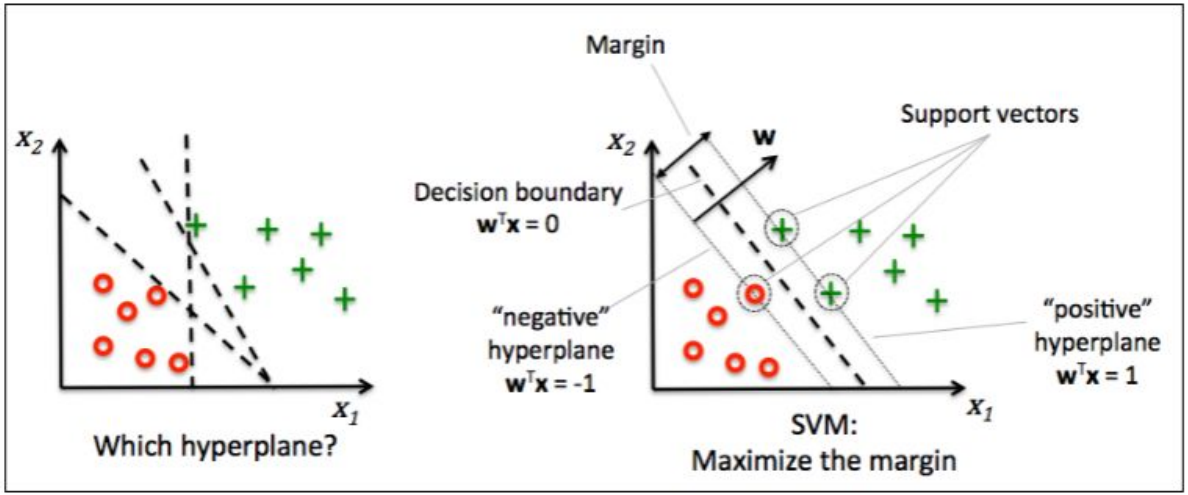
- Clasificador Lineal.
- Busca el hiperplano separador que **maximiza** el margen entre clases.
- Cuando las clases no son separables linealmente se acude al “soft-margin”, penalizador C (costo) que permite muestras “del otro lado”.
- Cada muestra mal clasificada es penalizada por un **costo C** que el usuario selecciona (un ej. de hiper-parámetro).
- El hiperplano separador queda definido por un subconjunto de “s” muestras. **Estas muestras son llamadas support vectors.**



Classification Models: SVM “Kernel Trick”

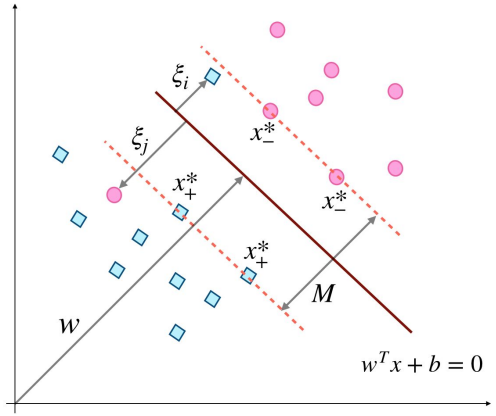
El hiperplano calculado por SVM será el mejor de todas las opciones posibles -> tiene **solución convexa** con un máximo global :)

El margen obtenido siempre es el máximo. El hiper-plano estará determinado por un sub-conjunto de muestras llamadas “support vectors”.



*Sebastian Raschka, Python Machine Learning

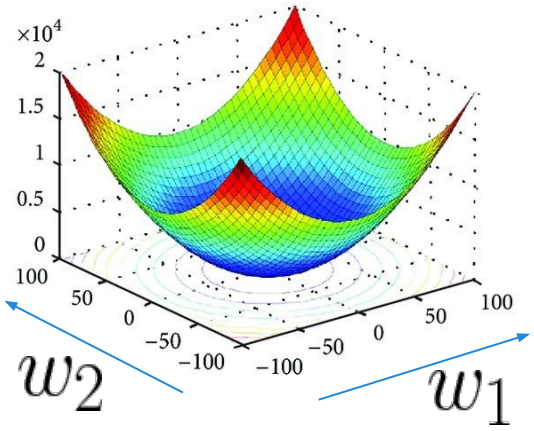
Aprendizaje del hiperplano: optimización convexa



$$\min \quad \frac{1}{2}\|w\|^2$$

$$\text{s.t} \quad y_i(wx_i + b) \geq 1$$

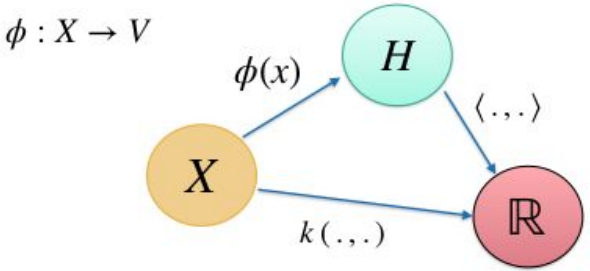
$$x \in \mathbb{R}^d \quad y \in \{-1, 1\}$$



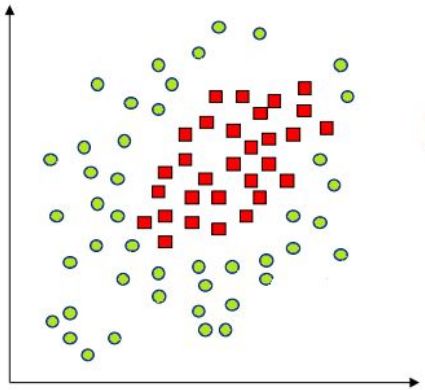
Durante el entrenamiento del SVM se buscara maximizar el margen separador entre las dos clases. Este problema de optimizacion es equivalente a minimizar la norma al cuadrado de los pesos w sujeto a restricciones en la funcion lineal.

Classification Models: SVM “Kernel Trick”

$$k\left(x_i, x_j\right) = \left\langle \phi(x_i), \phi(x_j) \right\rangle_H$$

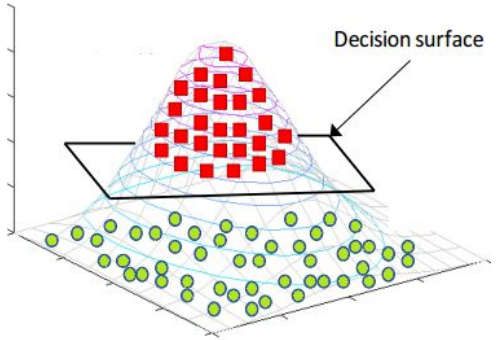


No separables linealmente



kernel

Separables linealmente por un hiperplano.



Los kernels son funciones de similaridad entre muestras. Mapean nuestros datos a un espacio de alta dimensión donde son linealmente separables. Allí en ese nuevo espacio donde son mapeadas las muestras se aplican los productos internos (o similaridad) entre muestras de manera que facilite la clasificación (kernel trick). Luego el resultado de dicha clasificación se representa en el espacio original. Si el kernel es no-lineal la frontera de decisión obtenida también será no lineal.

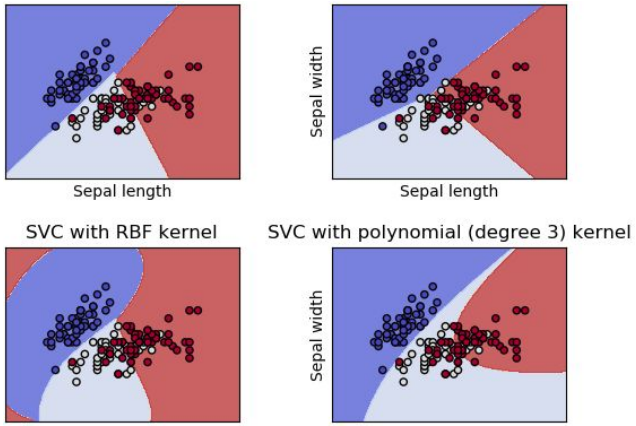
SVM: Hiper Parametros

Kernels más frecuentes: Gaussian, Linar, Polynomial

$$K_{gaussian}(x_i, x_j) = exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

$$K_{lin}(x_i, x_j) = \langle x_i, x_j \rangle$$

$$K_{poly}(x_i, x_j) = (\langle x_i, x_j \rangle + R)^d$$



Cada kernel hará que el SVM genere distintos tipos de frontera de clasificación. Los kernels gaussianos y polinomiales generarán fronteras no lineales mas complejas y el lineal o el polinomial de bajo grado mas sencillas. ¿cuando usamos un kernel complejo y cuando uno sencillo?

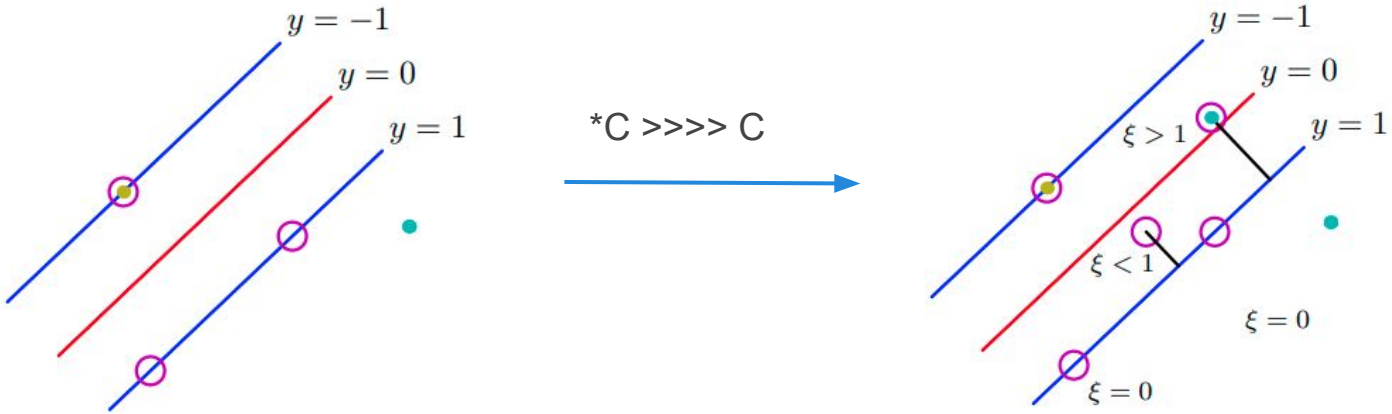
*Kernel Methods for Pattern Analysis, Nello Cristianini

SVM: Hiper Parametros

El SVM puede generar una familia de funciones discriminantes. La función final quedará determinada por los hiper-parámetros que seleccionemos (via cross validation)

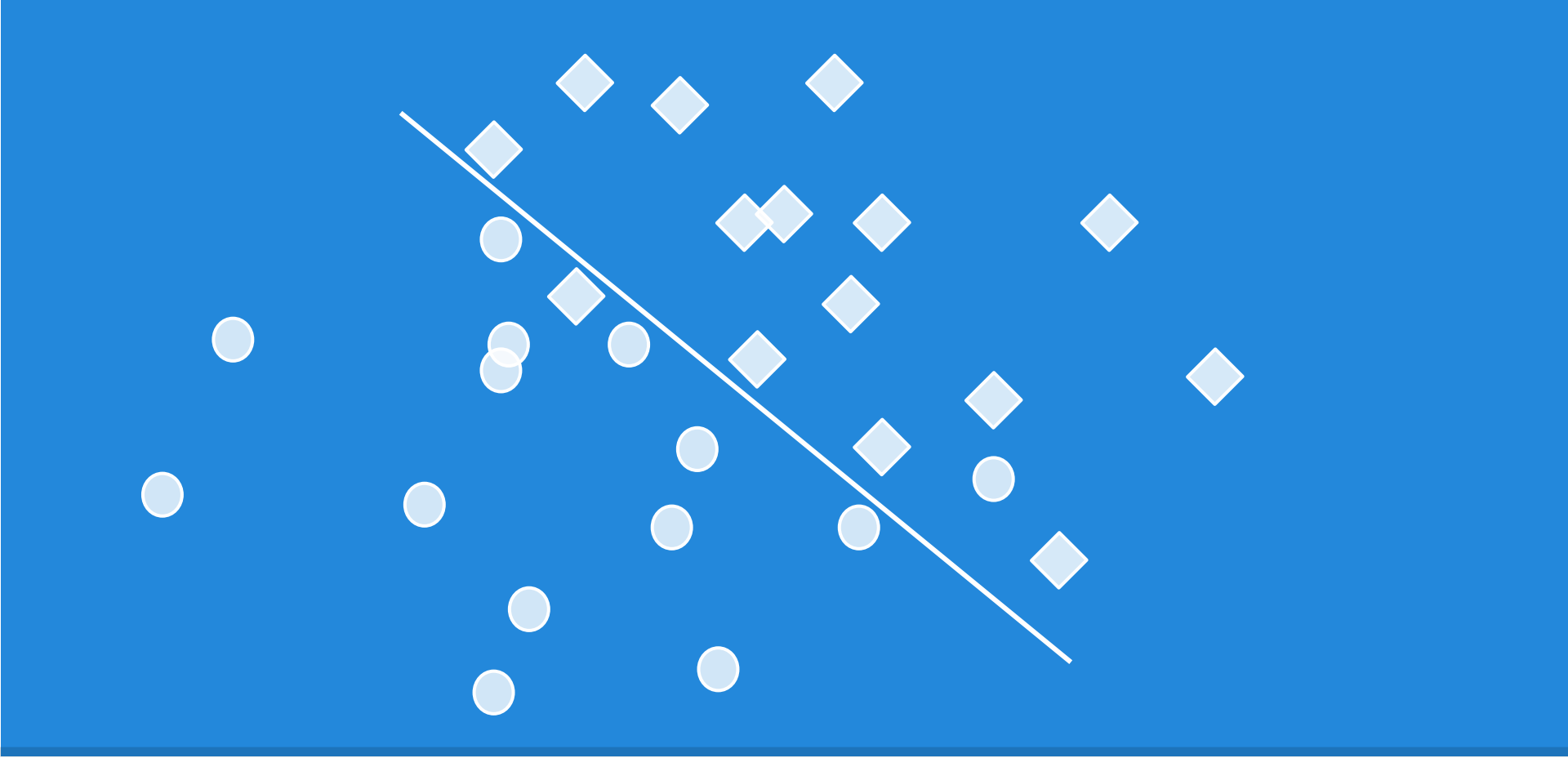
- Tipo de Kernel = Lineal, Polinomial, Gaussiano
- C = “Costo” por mal clasificación (todos los kernels)
- Gamma = Kernel Gaussiano (RBF)
- Degree = Kernel Polynomial

Regularización en SVM: hiperparametro costo C



El parámetro C penalizará al modelo por muestras mal clasificadas. Esto quiere decir que a mayor C el modelo se complejiza lo necesario para reducir las muestras mal clasificadas. A meno C por el contrario el modelo se relajara y permitirá algunas muestras mal clasificadas. Entonces el parametro C regulariza al SVM. A menor C mayor regularización.

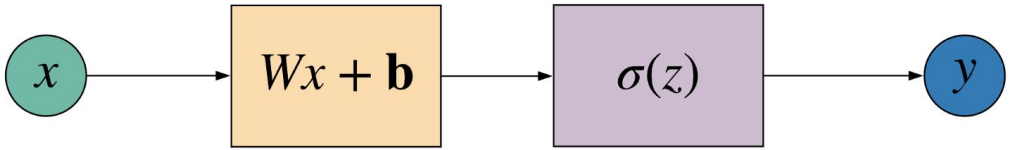
*Bishop, Pattern Recognition



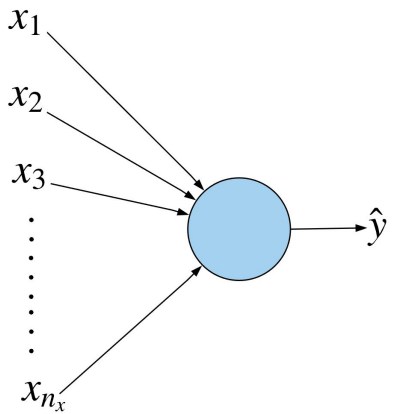
Aprendizaje Supervisado, Clasificación: Logistic Regression

Logistic Regression

- Logistic regression es un clasificador lineal.
- Es una regresión lineal precedida de una función de activación “sigmoid”, lo que genera que el output sea binario y no continuo como una regresión normal.
- Puede entenderse como una red neuronal de una sola capa y una sola neurona.
- A cada muestra clasificada, le asigna una probabilidad de pertenecer a cada clase existente en el problema. Si la probabilidad es mayor a cierto threshold (0.5) entonces pertenece a una clase y viceversa.



Modelo Lineal Funciòn Sigmoid



Logistic Regression

El regresor logístico debe aprender un parámetro interno (no es hiper-parámetro) por cada dimensión del vector de entrada (vector W). Para eso calculará el gradiente del error de clasificación y tratará de minimizarlo.

Probabilidad de la clase Y_i dado un vector de entrada x.

$$p(y_i|\mathrm{x}) = \sigma(w^T \mathrm{x})$$

La suma de las probabilidades de pertenecer a cada clase debe sumar 1.

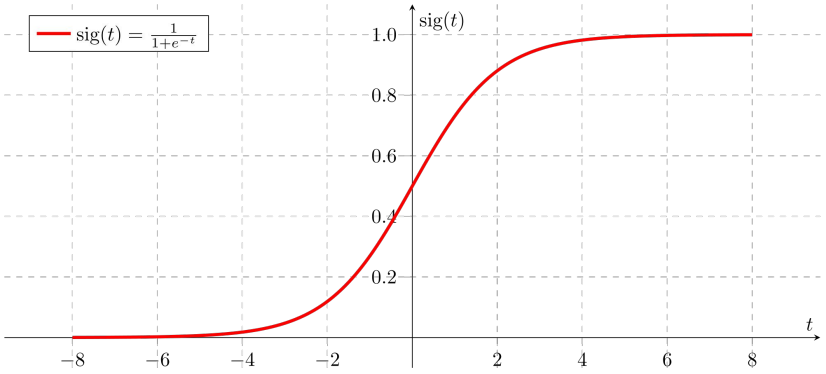
$$p(y_1|x) = 1 - p(y_2|x)$$

Función Sigmoid

$$\sigma(\mathrm{x}) = \frac{1}{1 + \exp(-\mathrm{x})}$$

*Logistic Regression, Pattern Recognition, Bishop

Logistic Regression

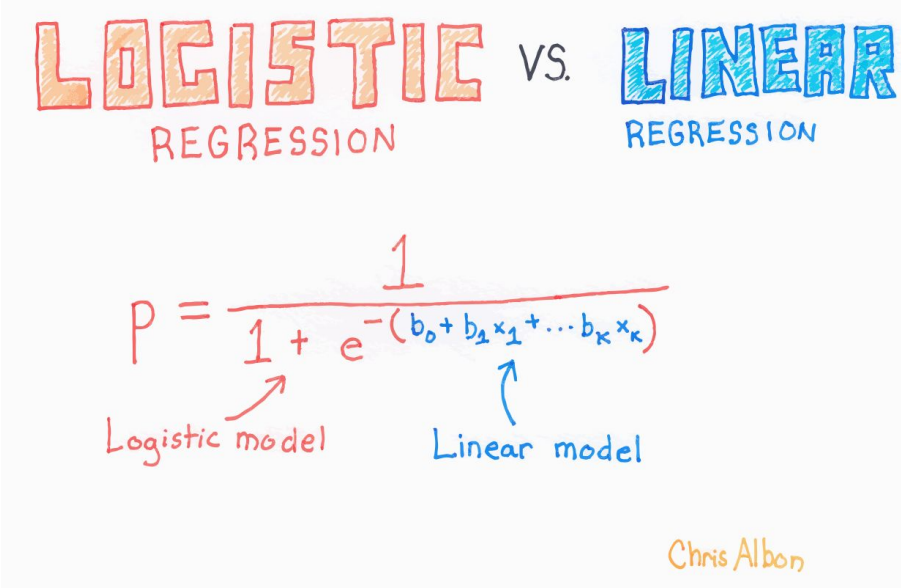


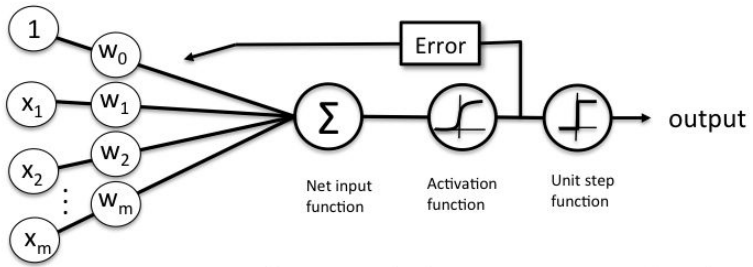
Función de activación “sigmoid”: mapea cualquier valor de X a un valor entre 0 y 1 pero nunca llega a estos extremos.

$$f(x) = \frac{1}{1 - e^{-x}}$$

$$w^T x + b$$

Logistic Regression





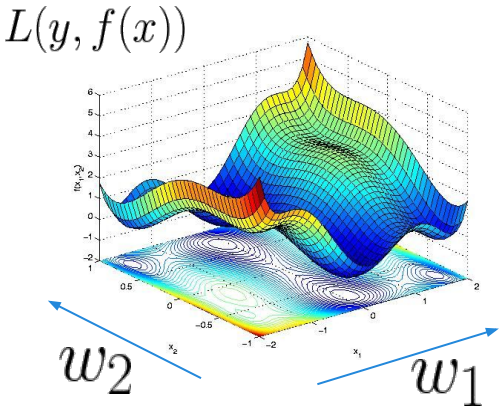
Schematic of a logistic regression classifier.

Lo que “aprende” logistic regression es el vector de “pesos” **W** de cada variable. Queremos que ese vector tenga “poder de generalización” para que clasifique bien nuevas muestras independientes una vez entrenado.

Logistic Regression: Optimizacion de funcion de costo

$$L_{\text{CE}}(\hat{y}, y) = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})]$$

$$L_{\text{CE}}(\hat{y}, y) = -[y \log \sigma(w \cdot x + b) + (1 - y) \log(1 - \sigma(w \cdot x + b))]$$

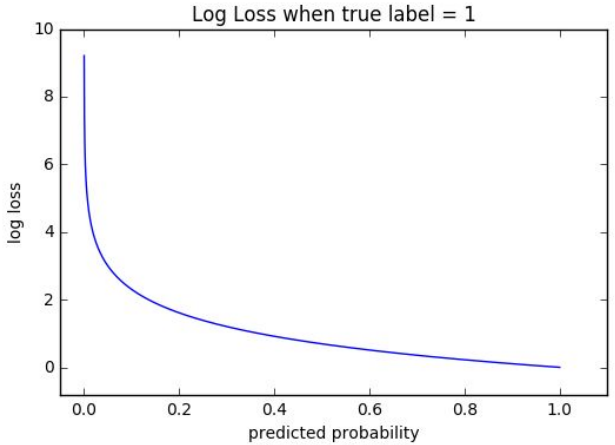


$$\hat{w} = \underset{w}{\text{argmin}} \frac{1}{n} \sum_{i=1}^{n} L_{\text{CE}} \left(f \left(x^{(i)}; w \right), y^{(i)} \right)$$

Para aprender la frontera de decision con logistic regression se buscara encontrar los pesos w de la funcion lineal dentro de la funcion logistica que minimicen el Loss Cross Entropy entre las etiquetas reales vs las etiquetas estimadas. La funcion de costo es no lineal y no convexa por lo que debera utilizarse un optimizador de gradiente descendiente.

Logistic regression Loss function: cross entropy

$$L_{\text{CE}}(\hat{y}, y) = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})]$$

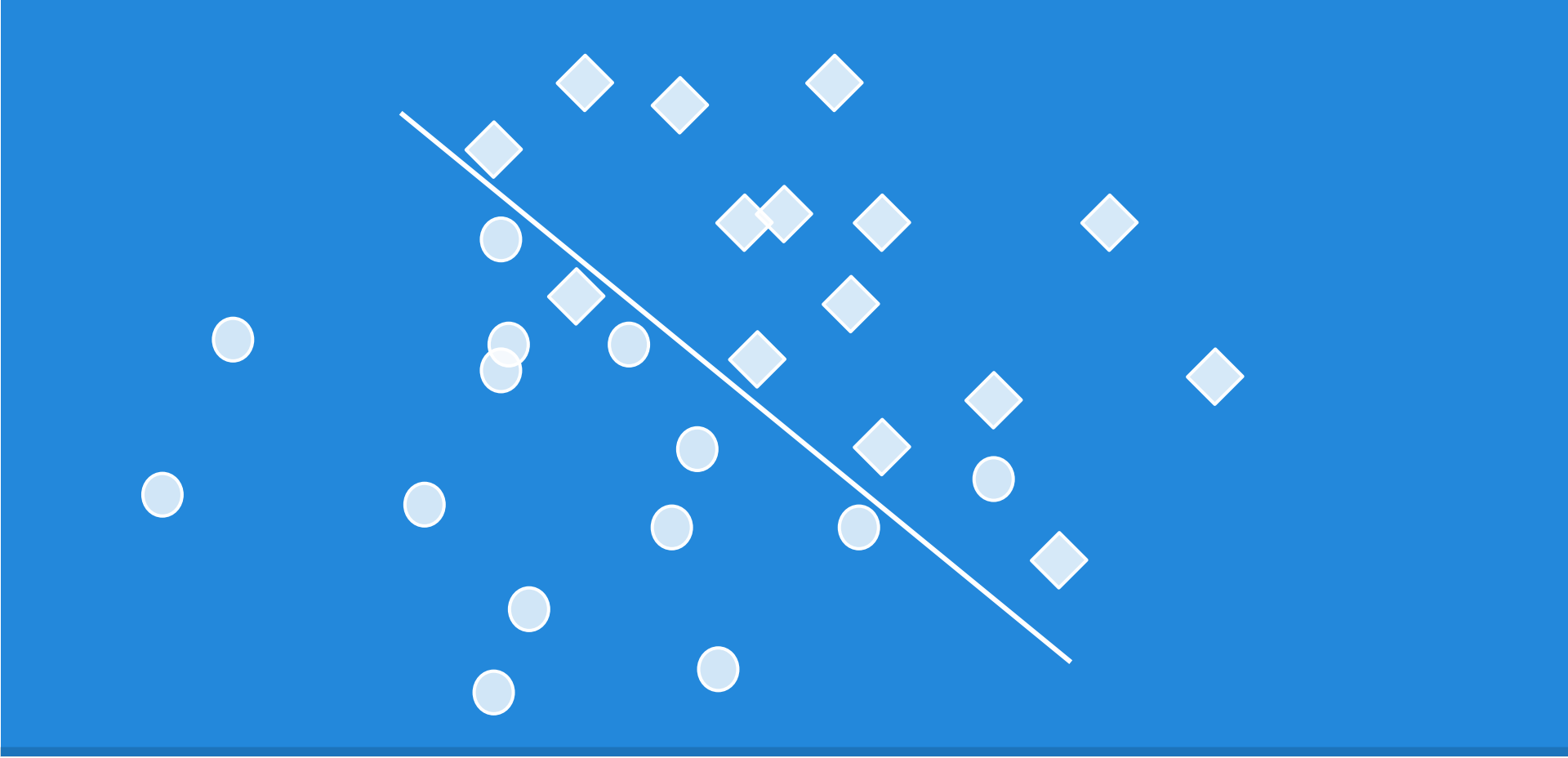


LR: Hiper Parametros

Los hiper-parámetros de Logistic Regression son

- C = “Costo” , al igual que SVM, es un penalizador que regulariza la solución computando clasificaciones erróneas.

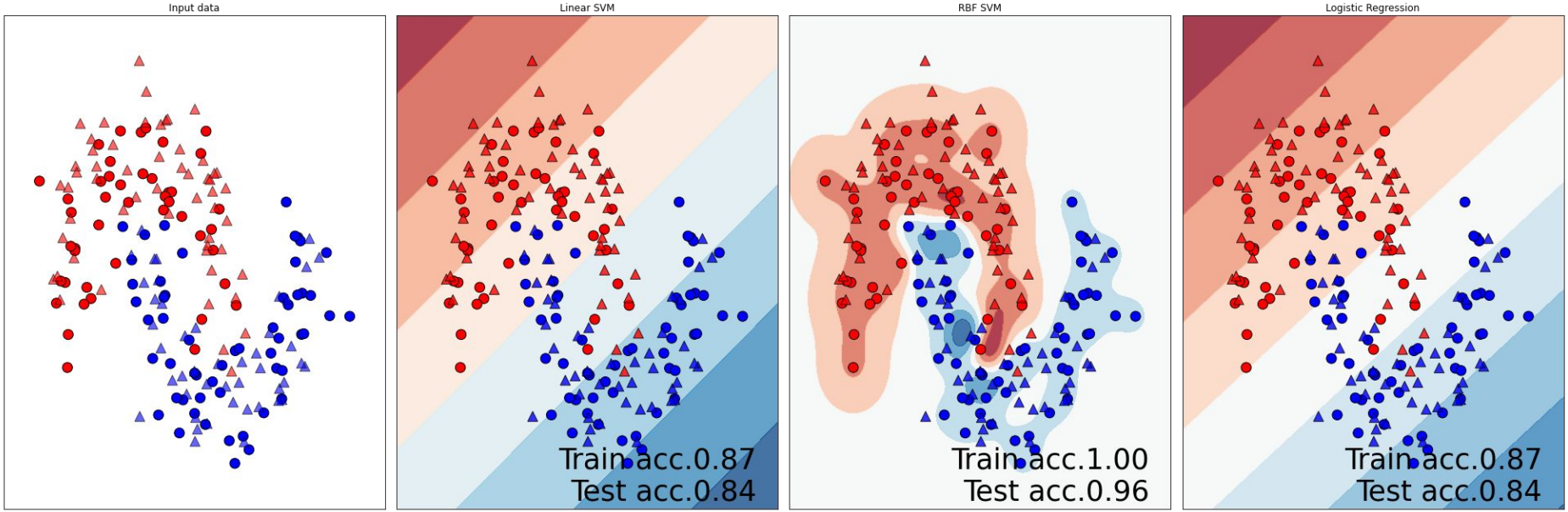
- Penalizador L1 o L2: aplica una penalización bajo la norma L1 o L2 al vector **W**. De esta manera evita que el modelo quiera “sobre-ajustarse” a los datos de entrenamiento. En otras palabras, evita que existan posiciones de W muy muy altas y otras casi nulas, o viceversa, que solo algunas posiciones de W se activen y el resto no.



Comparacion de múltiples modelos de clasificacion

Caso 1: dataset moons

```
classifiers_moons(dset='moons',  test_frac=0.5, param_c = 100, param_gamma = 10, lr_penalization = 'none')
```



Input data
Circle: train
Triangle: test
Color: label

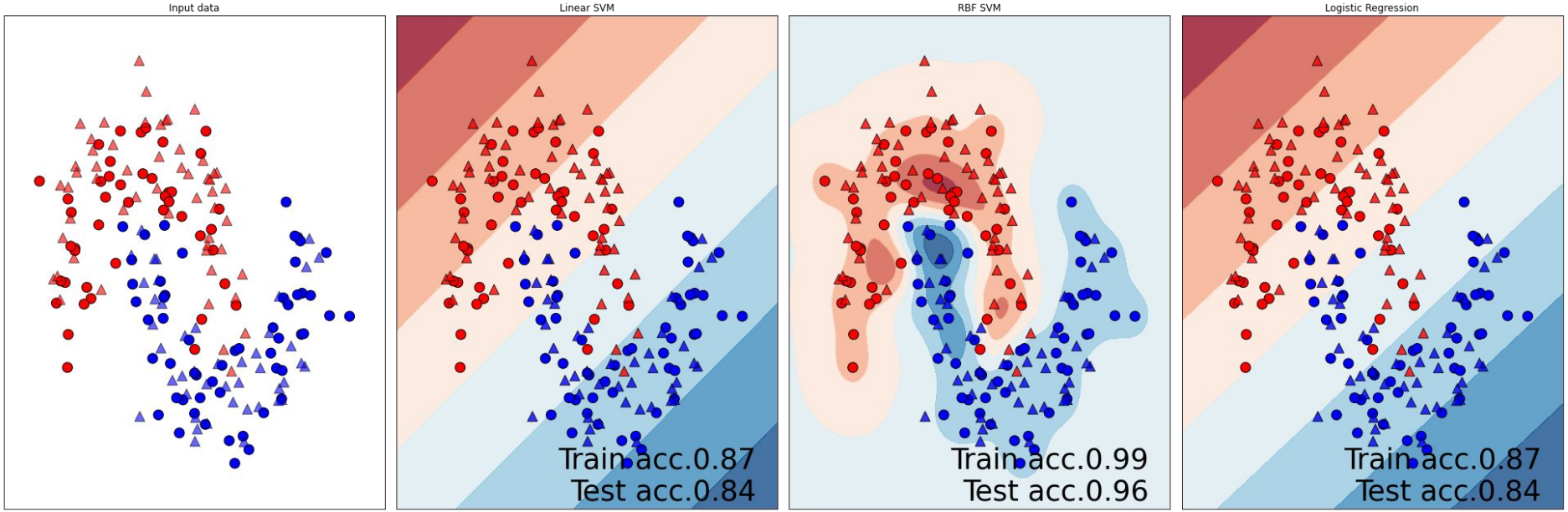
Linear SVM

Gaussian SVM

Logistic Regression

Caso 1: dataset moons

```
classifiers_moons( dset= 'moons', test_frac= 0.5, param_c = 10, param_gamma = 5, lr_penalization = 'l2')
```



Input data
Circle: train
Triangle: test
Color: label

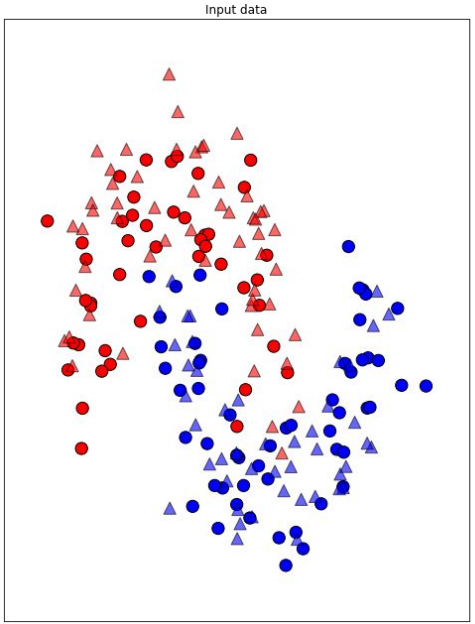
Linear SVM

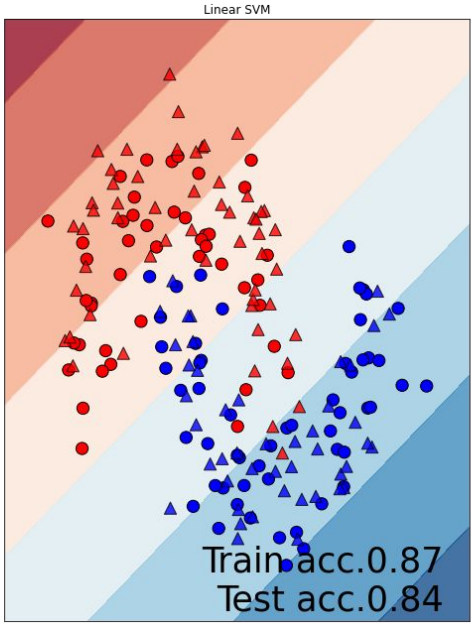
Gaussian SVM

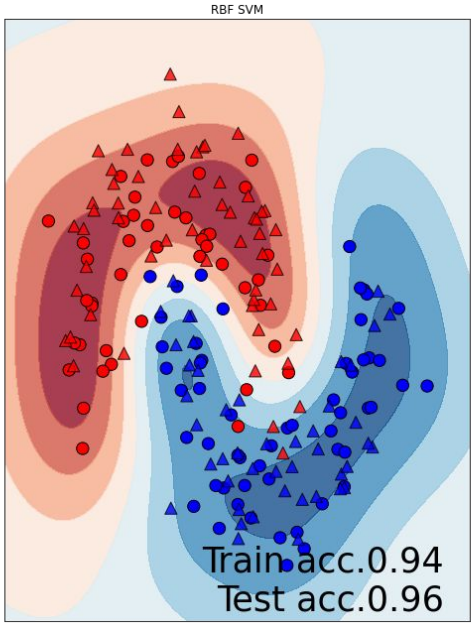
Logistic Regression

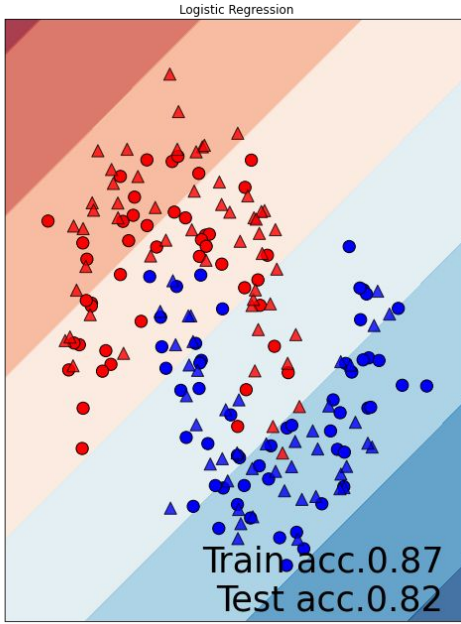
Caso 1: dataset moons

```
classifiers_moons( dset= 'moons', test_frac= 0.5, param_c = 1, param_gamma = 1, lr_penalization = 'l2')
```









Input data
Circle: train
Triangle: test
Color: label

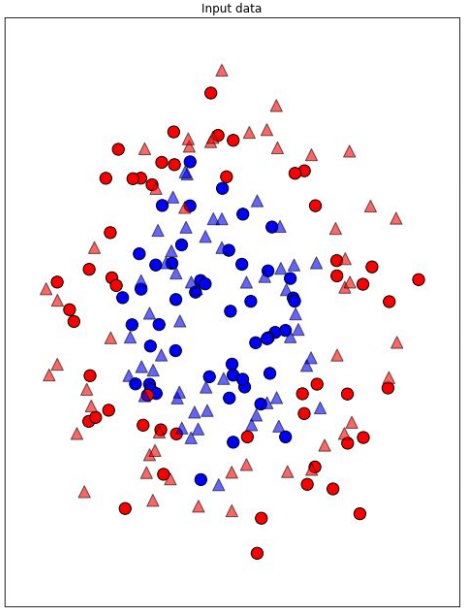
Linear SVM

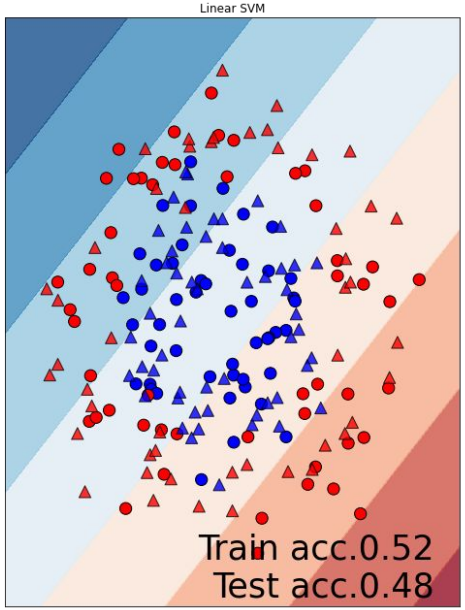
Gaussian SVM

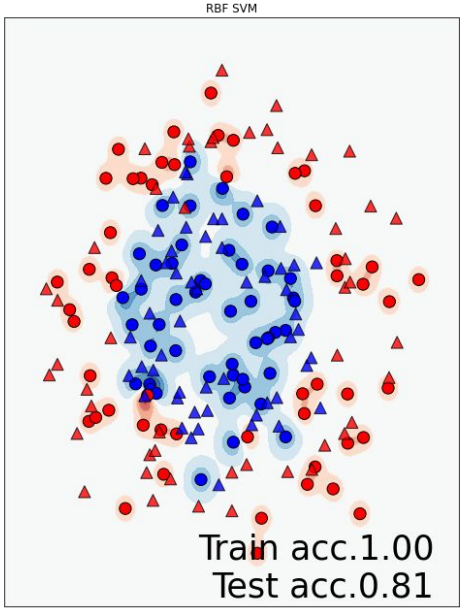
Logistic Regression

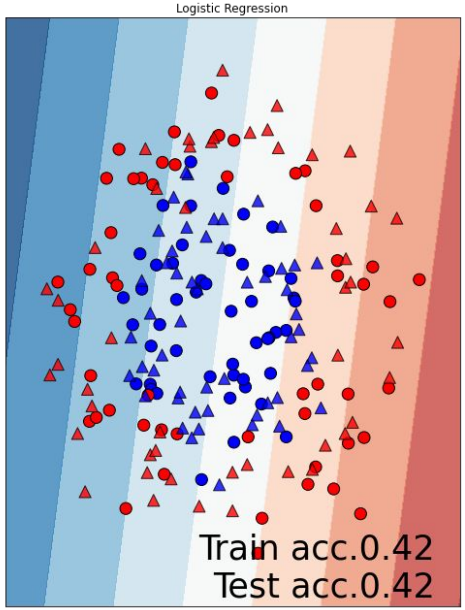
Caso 2: dataset circles

```
classifiers_moons( dset= 'circles', test_frac= 0.5, param_c = 100, param_gamma = 50, lr_penalization = 'none')
```









Input data
Circle: train
Triangle: test
Color: label

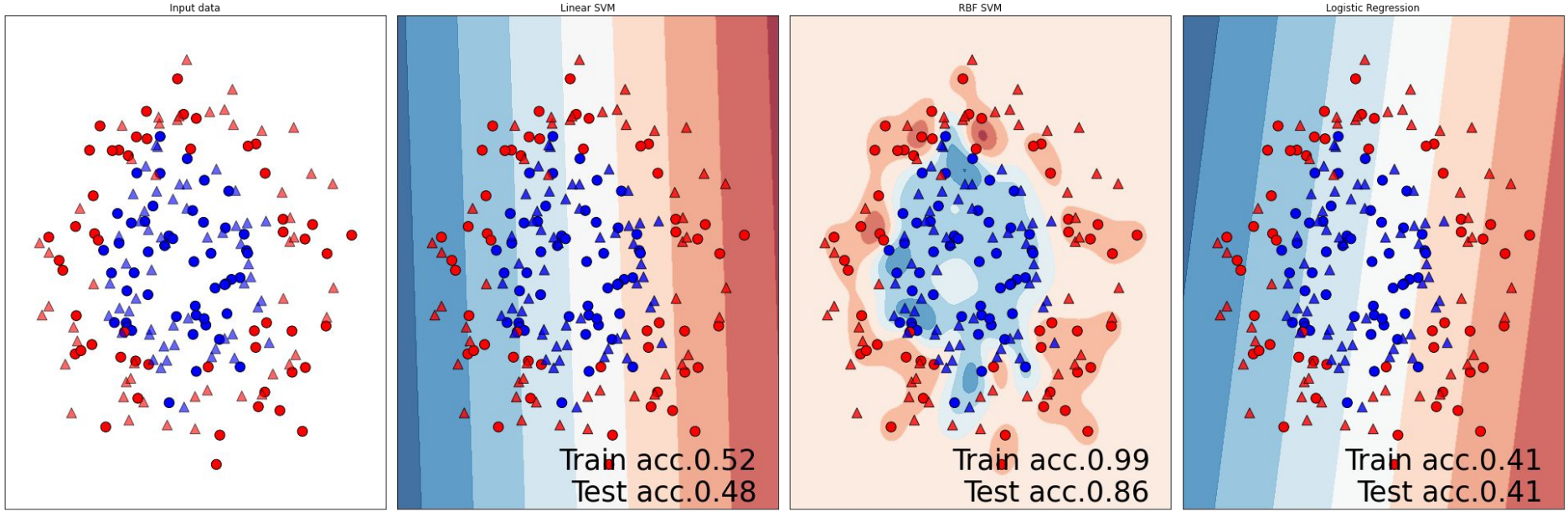
Linear SVM

Gaussian SVM

Logistic Regression

Caso 2: dataset circles

```
classifiers_moons( dset= 'circles', test_frac= 0.5, param_c = 10, param_gamma = 10, lr_penalization = 'l2')
```



Input data
Circle: train
Triangle: test
Color: label

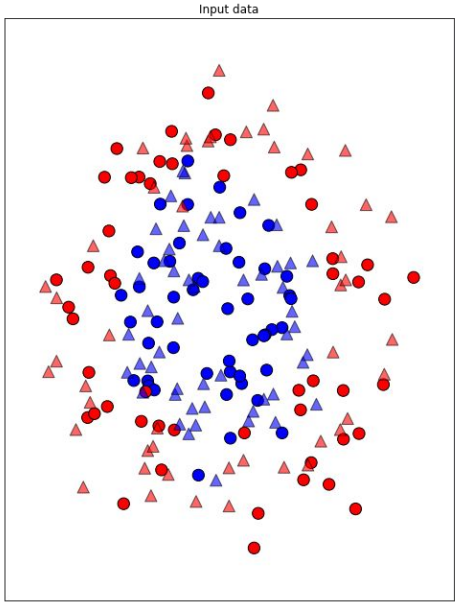
Linear SVM

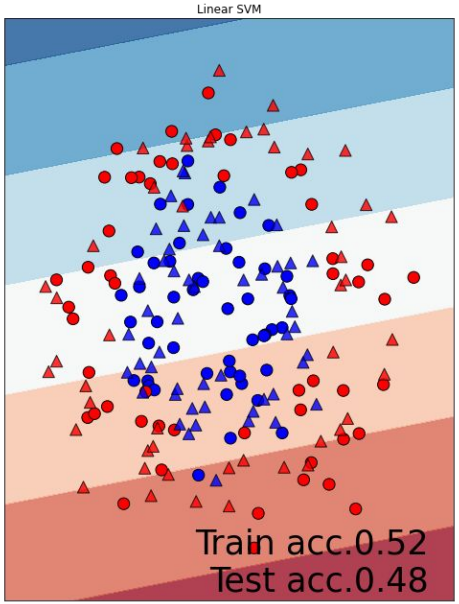
Gaussian SVM

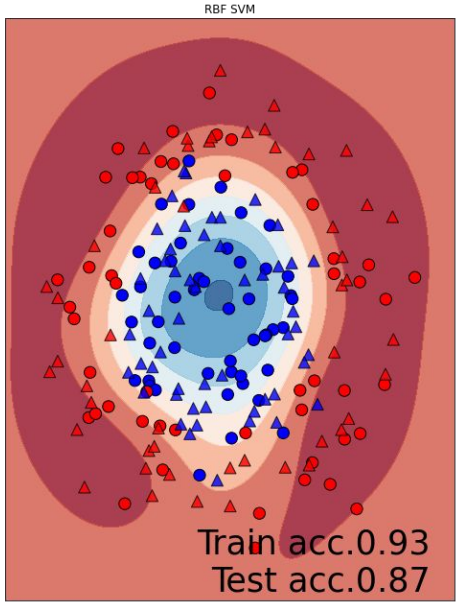
Logistic Regression

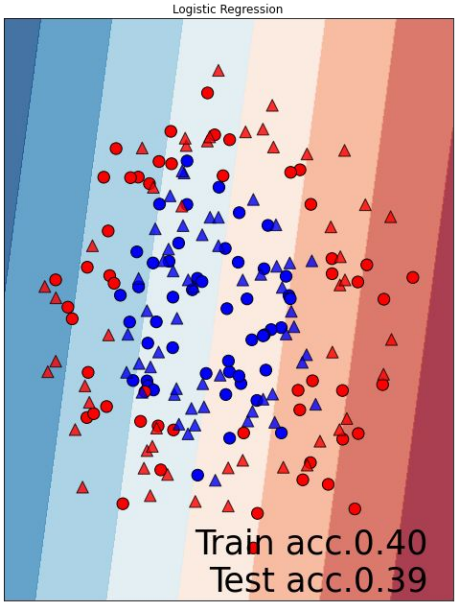
Caso 2: dataset circles

```
classifiers_moons( dset= 'circles', test_frac= 0.5, param_c = 1, param_gamma = 1, lr_penalization = 'l2')
```









Input data
Circle: train
Triangle: test
Color: label

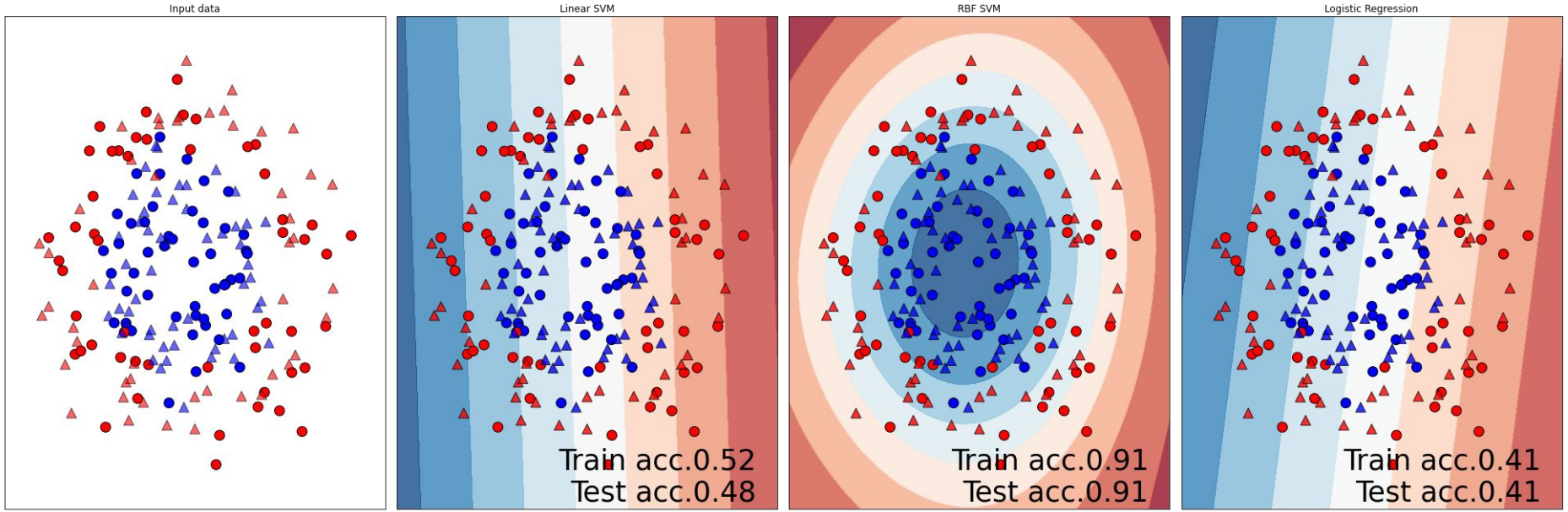
Linear SVM

Gaussian SVM

Logistic Regression

Caso 2: dataset circles

```
classifiers_moons( dset= 'circles', test_frac= 0.5, param_c = 10, param_gamma = 0.1, lr_penalization = 'l2')
```



Input data
Circle: train
Triangle: test
Color: label

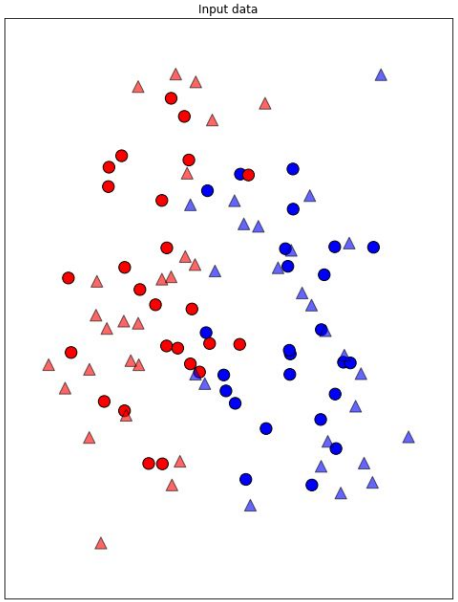
Linear SVM

Gaussian SVM

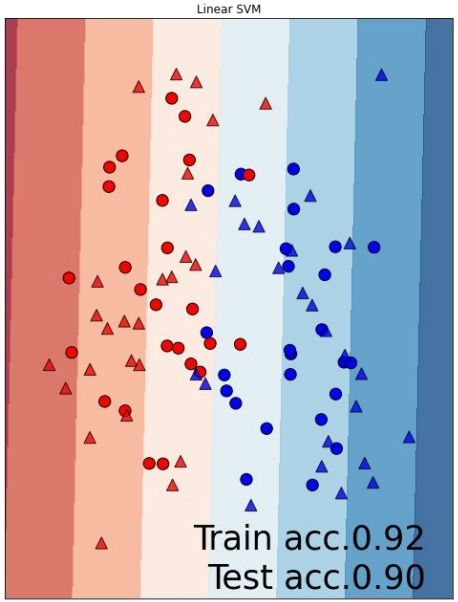
Logistic Regression

Caso 3: dataset linearly separable

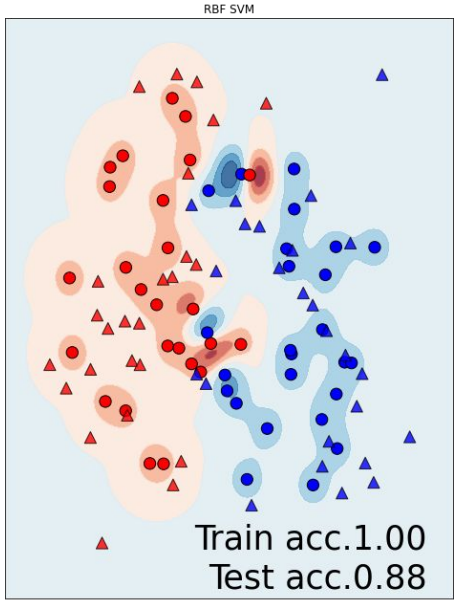
```
classifiers_moons( dset= 'lin_sep', test_frac= 0.5, param_c = 100, param_gamma = 20, lr_penalization = 'none')
```



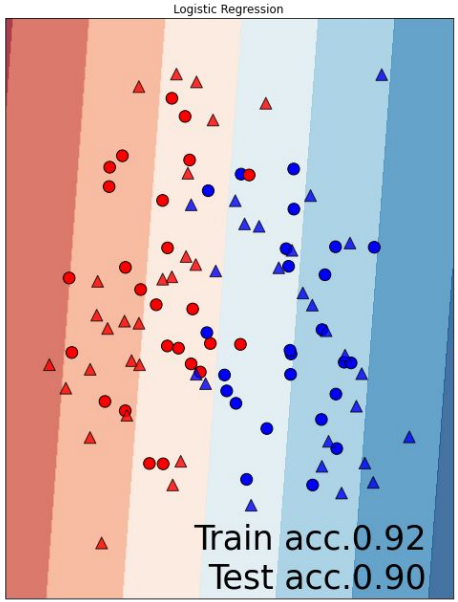
Input data
Circle: train
Triangle: test
Color: label



Linear SVM



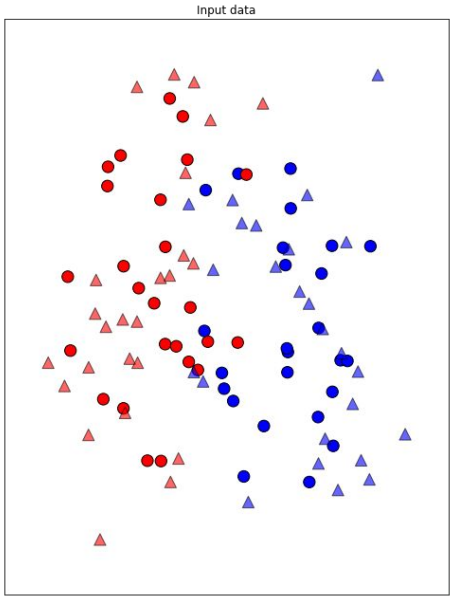
Gaussian SVM



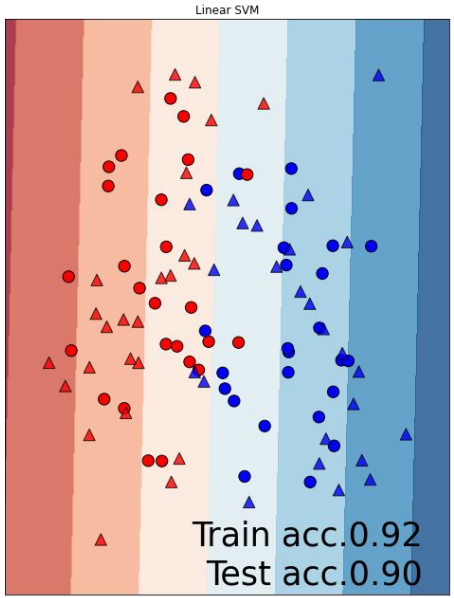
Logistic Regression

Caso 3: dataset linearly separable

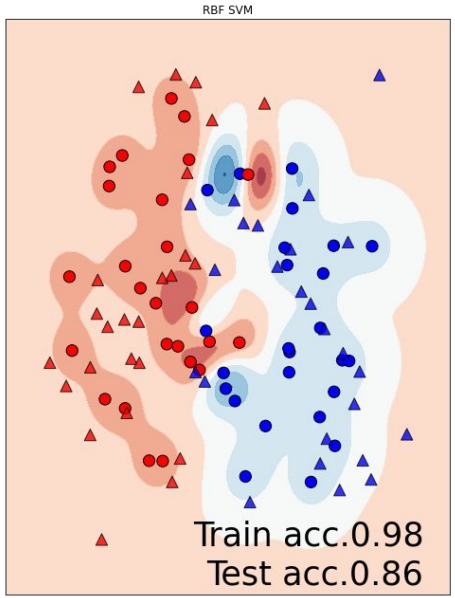
```
classifiers_moons( dset= 'lin_sep', test_frac= 0.5, param_c = 10, param_gamma = 10, lr_penalization = 'l2')
```



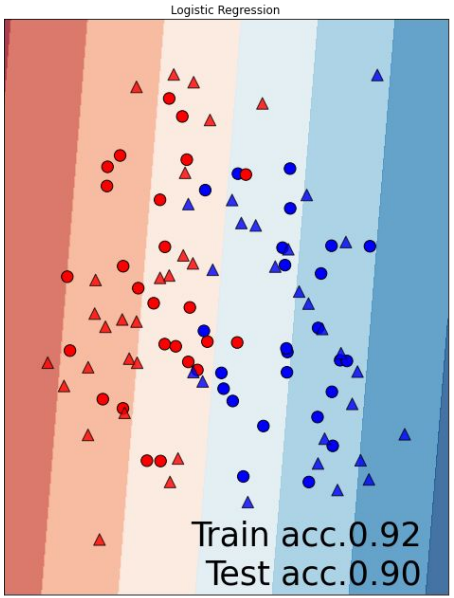
Input data
Circle: train
Triangle: test
Color: label



Linear SVM



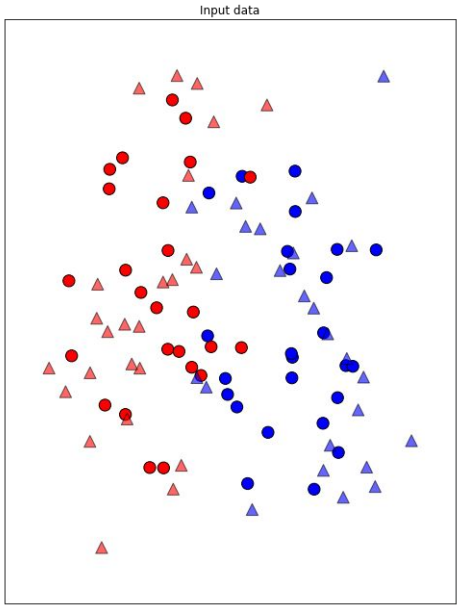
Gaussian SVM



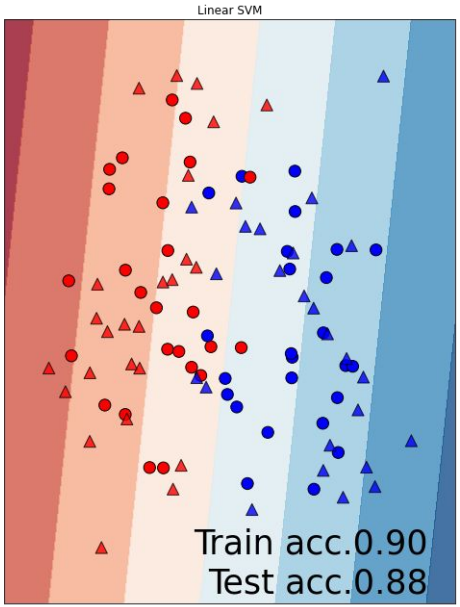
Logistic Regression

Caso 3: dataset linearly separable

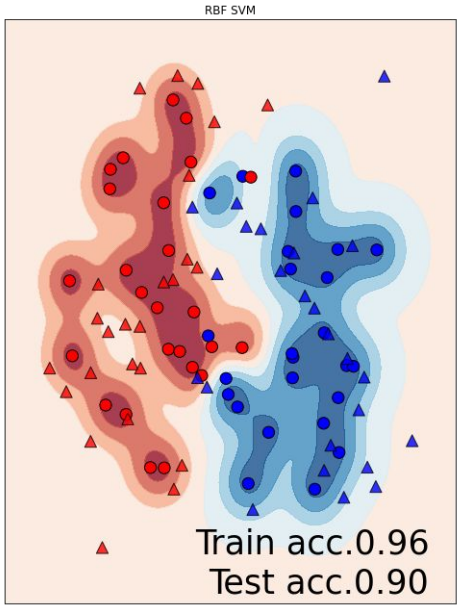
```
classifiers_moons( dset= 'lin_sep', test_frac= 0.5, param_c = 1, param_gamma = 10, lr_penalization = 'l2')
```



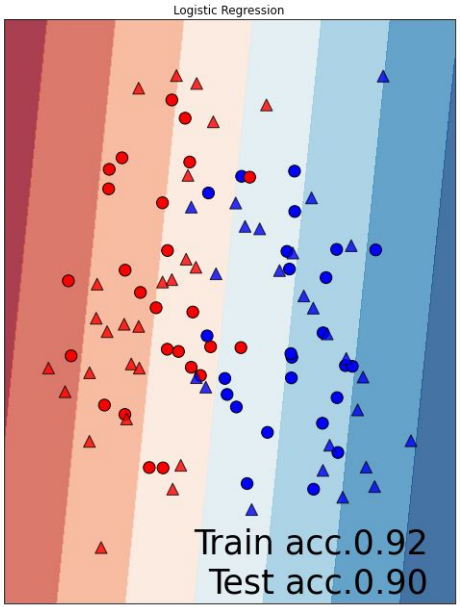
Input data
Circle: train
Triangle: test
Color: label



Linear SVM



Gaussian SVM



Logistic Regression