

DIPLOMADO EN CIENCIA DE DATOS

# MODELACIÓN SUPERVISADA SEMANA 3

Facultad de Estudios Superiores Acatlán

# OUTLINE

## **Módulo 2 - Semana 3**

- 1.- Regresión Ridge
- 2.- Regresión Lasso
- 3.- Red Elástica
- 4.- Regresión Logística
- 5.- Máquinas Vector Soporte

# QUIZ 2



**<https://b.socrative.com/login/student/>**  
**Room name: IRENE2290**

# QUIZ 1 - PREGUNTAS ABIERTAS



P.- Te dan un set de datos ( $x,y$ ) para llevar a cabo una tarea de clasificación, y solo puede tomar los valores de  $\{0,1\}$  y presenta únicamente 2% de valores positivos. Entrenas un clasificador que tiene 60% de efectividad. ¿Cuál es tu reacción?

P.- Tengo una variable categórica con 5 categorías distintas. ¿Cuántas variables dummy tendrá después de llevar a cabo el proceso de one-hot encoding?

# SHOW TIME!



**Bootstrap para evaluar y seleccionar  
modelos de aprendizaje de máquina**

# 1.- Regresión Ridge

# SOBREAJUSTE Y DESAJUSTE

## OTRA FORMA DE MITIGAR EL SOBREAJUSTE DE DATOS



Agregar un término de regularización a la función de costo



### Término de regularización para mitigar el sobreajuste de datos

- El término Omega nos ayuda a regularizar la función de costo, penalizando a los modelos complejos y favoreciendo a los simples

$$\min_{\mathbf{W}} \mathcal{L}(\mathbf{W}) + \underbrace{\Omega(\mathbf{W})}_{\text{Término regularizador}}$$

### Término de regularización comunes

- Norma L2
  - Parámetros grandes para  $\mathbf{W}$  serán penalizados, mientras que los pequeños estarán ok
- Norma L1
  - La solución óptima para  $\mathbf{W}$  será poco densa (sparse)

$$\Omega(\mathbf{W}) = \lambda \|\mathbf{W}\|_2^2$$

$$\Omega(\mathbf{W}) = \lambda \|\mathbf{W}\|_1$$

# SOBREAJUSTE Y DESAJUSTE

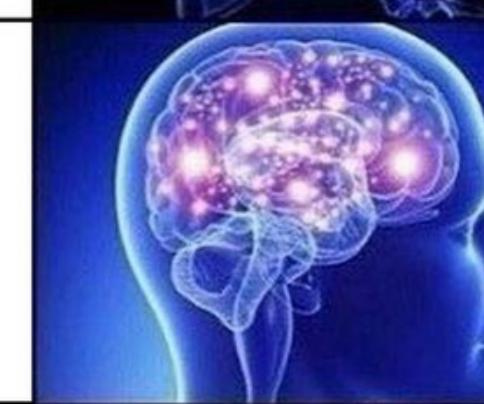
## FORMAS DE EVALUAR SI NUESTRO MODELO SUFRE DE SOBREAJUSTE



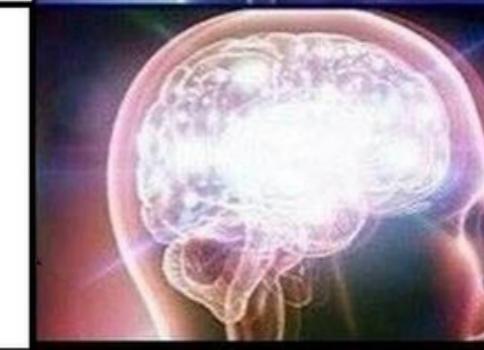
Entrenar y evaluar el modelo en un mismo set de datos de entrenamiento



Partir el set de datos original en train y test. Entrenar en el set de entrenamiento y evaluar en el de validación



Partir el set de datos original en train y test. Entrenar en train y evaluar en test. Obtener un set de datos independiente (holdout) y hacer una evaluación extra



Utilizar K-Fold Cross-Validation para determinar los parámetros óptimos para el término de regularización que nos ayude a mitigar el sobreajuste



K-Fold Cross-Validation nos ayuda a obtener un estimación insesgada de la generalización del error y de su varianza. Por lo tanto, nos ayuda a ver qué tan bien o qué tan mal nuestro modelo generalizará a nuevos sets de datos

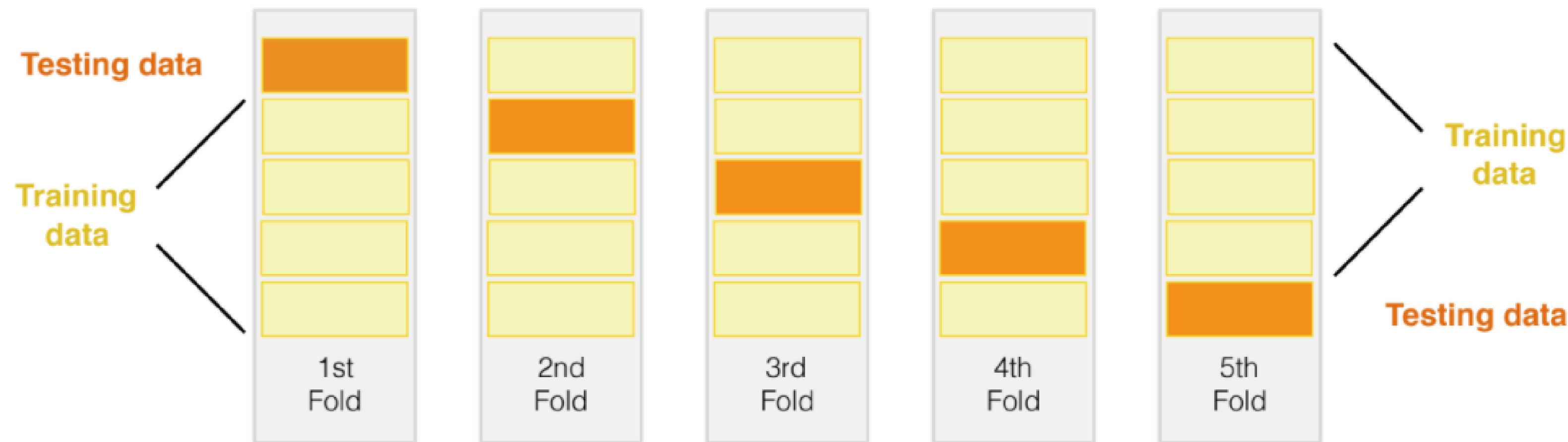
# K-FOLD CROSS-VALIDATION

## FORMAS DE EVALUAR SI NUESTRO MODELO SUFRE DE SOBREAJUSTE



Divide el set de datos K veces, tal que todos los datos sirvan para entrenar y evaluar un modelo de manera distinta cada vez. Al final del ciclo iterativo, el promedio de las métricas obtenidas en cada iteración será la generalización del performance del modelo (error, otras métricas). También puedes calcular la desviación estándar de las métricas, para determinar que tan estable es el modelo

El mismo set de datos será utilizado 5 veces por separado



# K-FOLD CROSS-VALIDATION

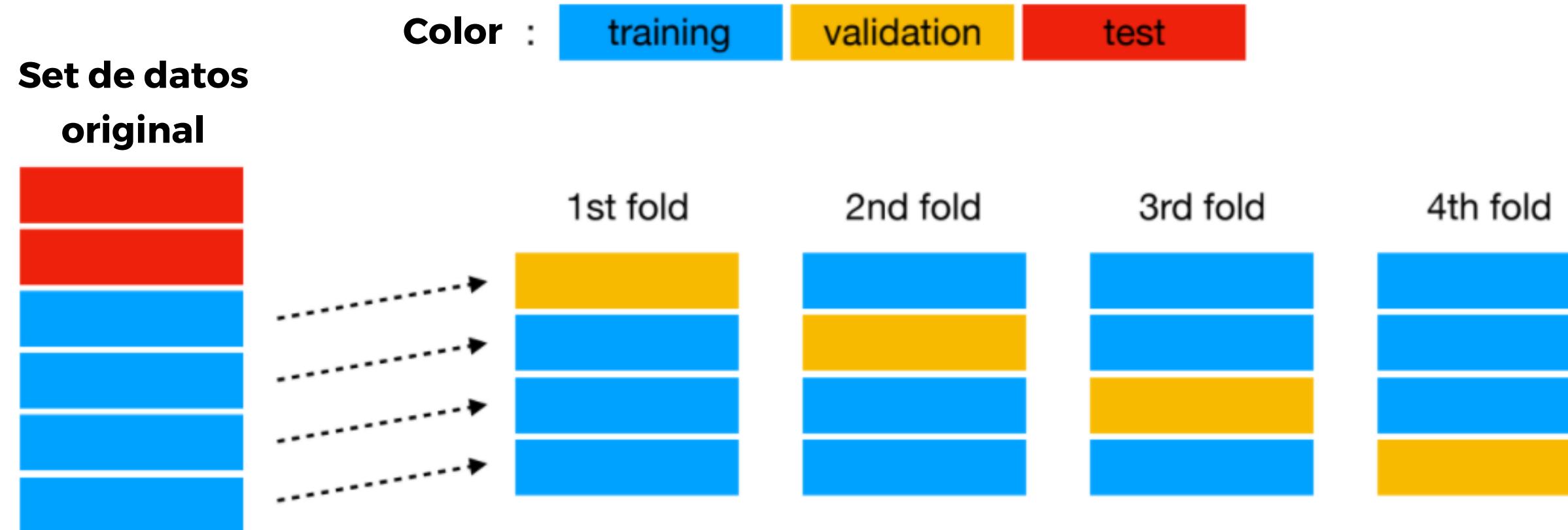
## PARA AFINAR HYPER PARÁMETROS (HYPERPARAMETER TUNING)



Algoritmo para seleccionar hyper parámetros óptimos para nuestro modelo. Seleccionar el que nos de un mejor trade-off entre sesgo y varianza



1. Partir el set de datos en **train** y **test**
2. Partir el set de **train** k veces en un **train** y **validation**, tal que todos los datos en el set de **train original** son utilizados para validación
3. Evaluar el performance de cada hyper parámetro en cada una de las particiones

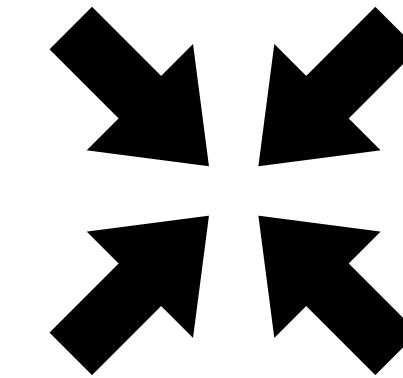


# REGRESIÓN RIDGE

## CONCEPTOS BÁSICOS



Es una de las versiones regularizadas de la regresión lineal



### Definición

- Target numérica → Regresión
- Un set de datos está compuesto por el par  $(\mathbf{x}_n, y_n)$
- Función de costo

$$\mathcal{L}(\mathbf{W}) = \frac{1}{N} \sum_{i=1}^N (y_i - \mathbf{x}_n^T \mathbf{W})^2 + \alpha \|\mathbf{W}\|_2^2$$

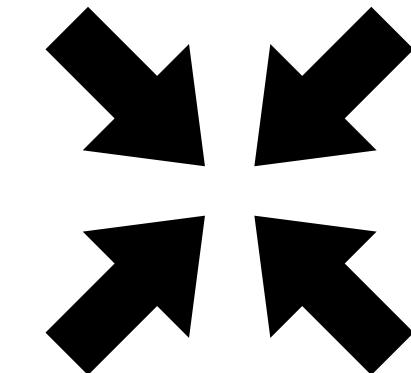
### Características

- Utiliza la norma L2 multiplicada por un hyper parámetro como término regularizador  $\alpha$
- Ayuda a mitigar el sobreajuste
- Forma de regularizar muy frecuentemente usada
- El término regularizador obliga al modelo a aprender de los datos, pero también a mantener pequeños a los pesos del modelo (parámetros W)
- El hyper parámetro alfa define cuánto queremos regularizar el modelo. ¿Qué pasa si alfa = 0?
- Pesos W grandes serán penalizados, mientras que los pesos W pequeños serán favorecidos
- "Encoge" los pesos W al imponer una penalización en su tamaño

# REGRESIÓN RIDGE

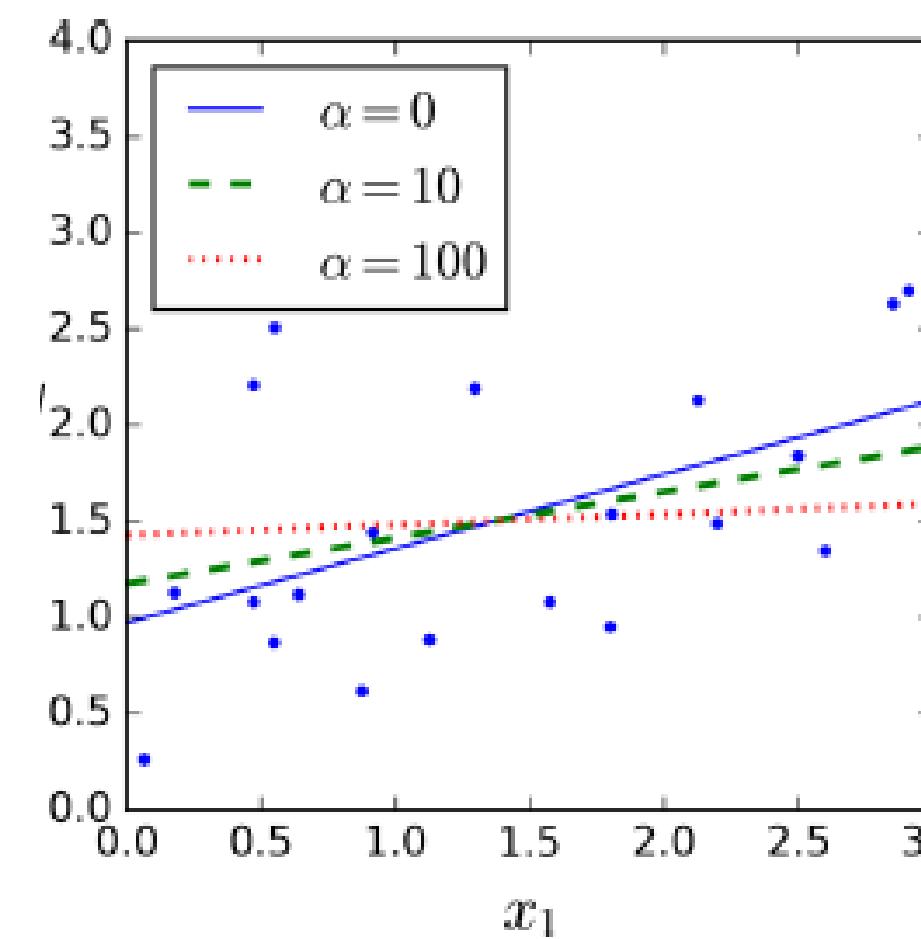
## EJEMPLO\*

Si el valor de alfa es muy grande, entonces los pesos W terminan siendo muy cercanos a cero.  
Lo anterior hace que las predicciones se aplaten.



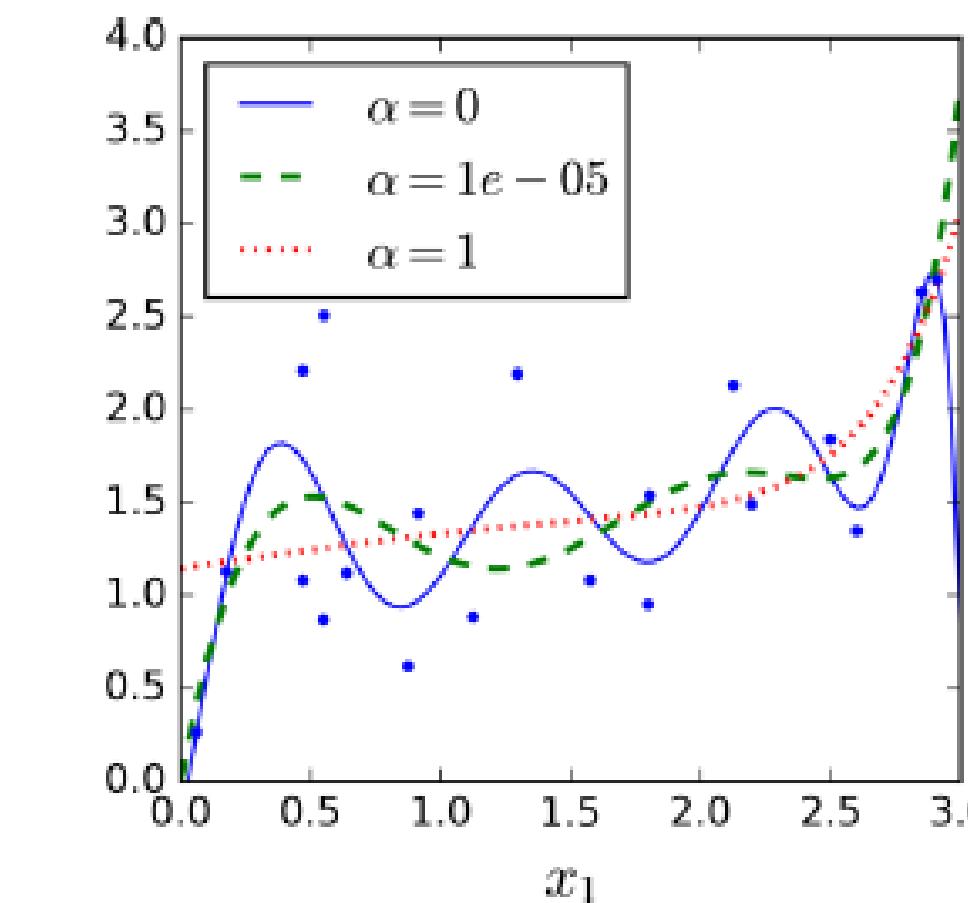
### Ridge sin extender set de datos

- Al incrementar alfa se tienen predicciones más planas, lo cual reduce la varianza pero incrementa el sesgo



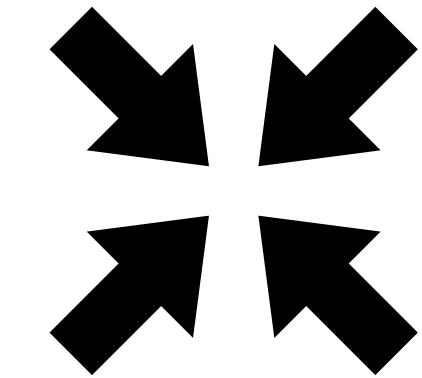
### Ridge con un set de datos extendido (polinomio grado 10)

- Al incrementar alfa se tienen predicciones más planas, lo cual reduce la varianza pero incrementa el sesgo

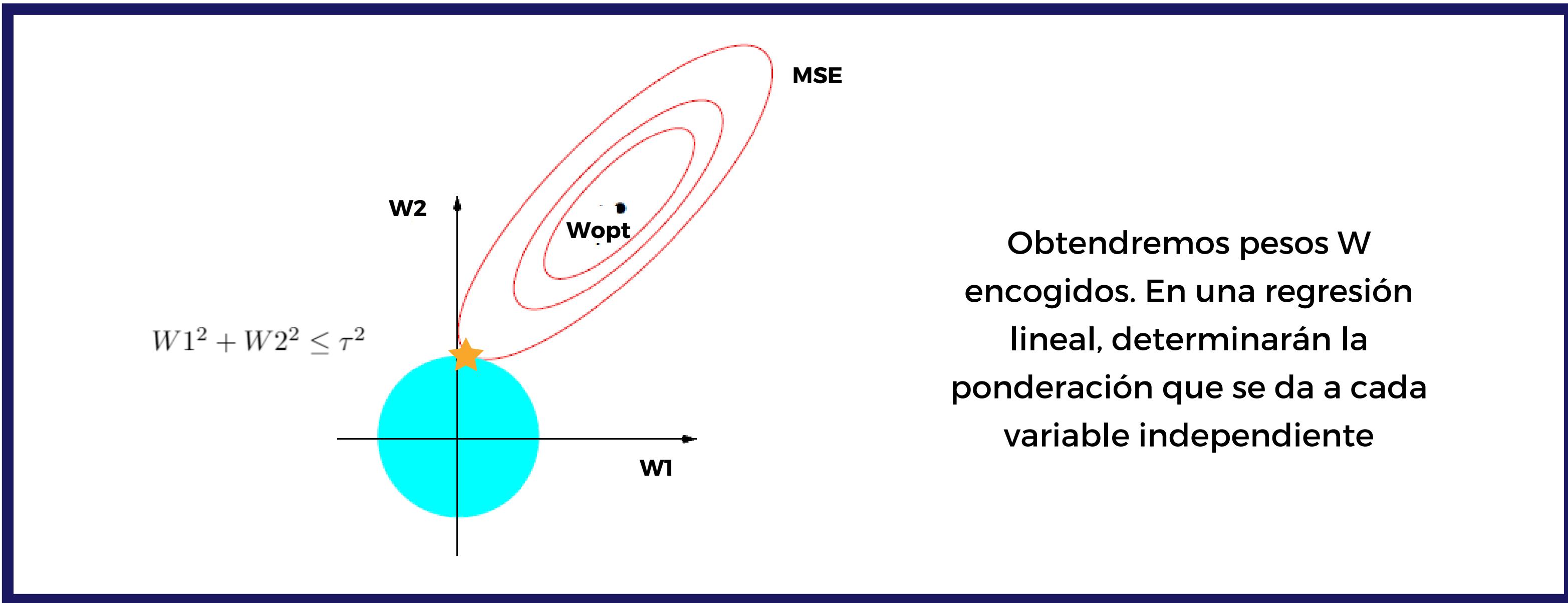


# REGRESIÓN RIDGE

## INTERPRETACIÓN GEOMÉTRICA

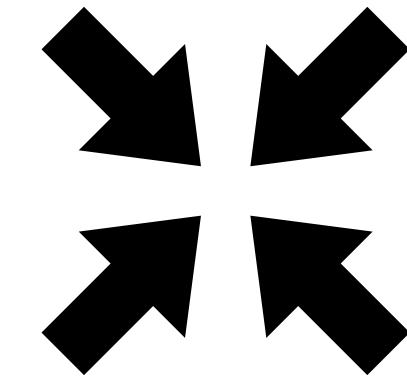


La regresión Ridge encoge los parámetros W al imponer una penalización (término regularizador)



# REGRESIÓN RIDGE

## ENTRENAMIENTO DE UN MODELO



Para entrenar un modelo se puede utilizar la librería Sklearn.

### Clase Ridge con parámetros por default

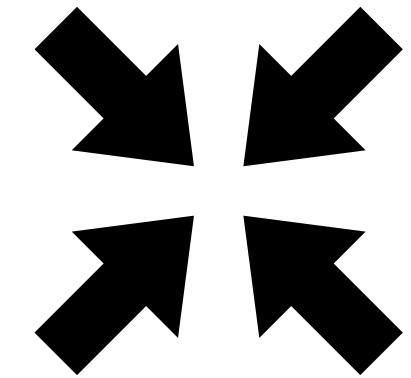
```
from sklearn.linear_model import Ridge
ridge_reg = Ridge(alpha=1.0, *, fit_intercept=True, normalize='deprecated',
                   copy_X=True, max_iter=None, tol=0.001, solver='auto',
                   positive=False, random_state=None)
ridge_reg.fit(X_train, y_train)
ridge_reg.predict(X_test)
ridge_reg.score(X_test, y_test)
```

hyper parámetro a optimizar con K-Fold CV

{'auto', 'svd', 'cholesky', 'lsqr',  
'sparse\_cg', 'sag', 'saga', 'lbfgs'},  
default='auto'

# REGRESIÓN RIDGE

## ENTRENAMIENTO DE UN MODELO



Para entrenar un modelo se puede utilizar la librería Sklearn.

### Atributos de clase Ridge

```
ridge_reg.coef_  
ridge_reg.intercept_
```



Arreglo con pesos W óptimos



Arreglo o número float con  
intercepto W0 óptimo

¿Cuáles otros existen?

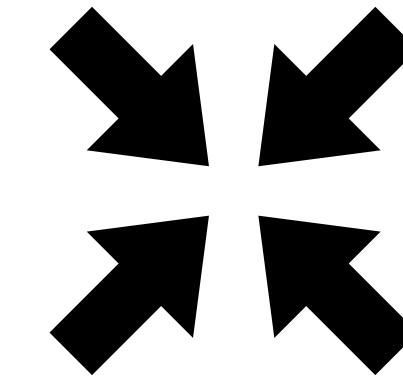
## 2.- Regresión Lasso

# REGRESIÓN LASSO

## CONCEPTOS BÁSICOS



Otra versión regularizada de la regresión lineal



### Definición

- Target numérica → Regresión
- Un set de datos está compuesto por el par  $(\mathbf{x}_n, y_n)$
- Función de costo

$$\mathcal{L}(\mathbf{W}) = \frac{1}{N} \sum_{i=1}^N (y_i - \mathbf{x}_n^T \mathbf{W})^2 + \alpha \|\mathbf{W}\|_1$$

### Características

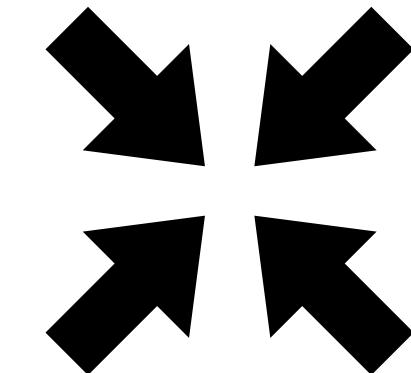
- Utiliza la norma L1 multiplicada por un hyper parámetro como término regularizador  $\alpha$
- Ayuda a mitigar el sobreajuste
- Tiende a eliminar pesos  $\mathbf{W}$  de menor importancia para el modelo (lleva dichos pesos a cero)
- El hyper parámetro alfa define cuánto queremos regularizar el modelo. ¿Qué pasa si alfa = 0?
- Lasso lleva a cabo una forma de selección de variables, y tiene como output un modelo poco denso (sparse model) ¿Por qué?

# REGRESIÓN LASSO

## EJEMPLO\*

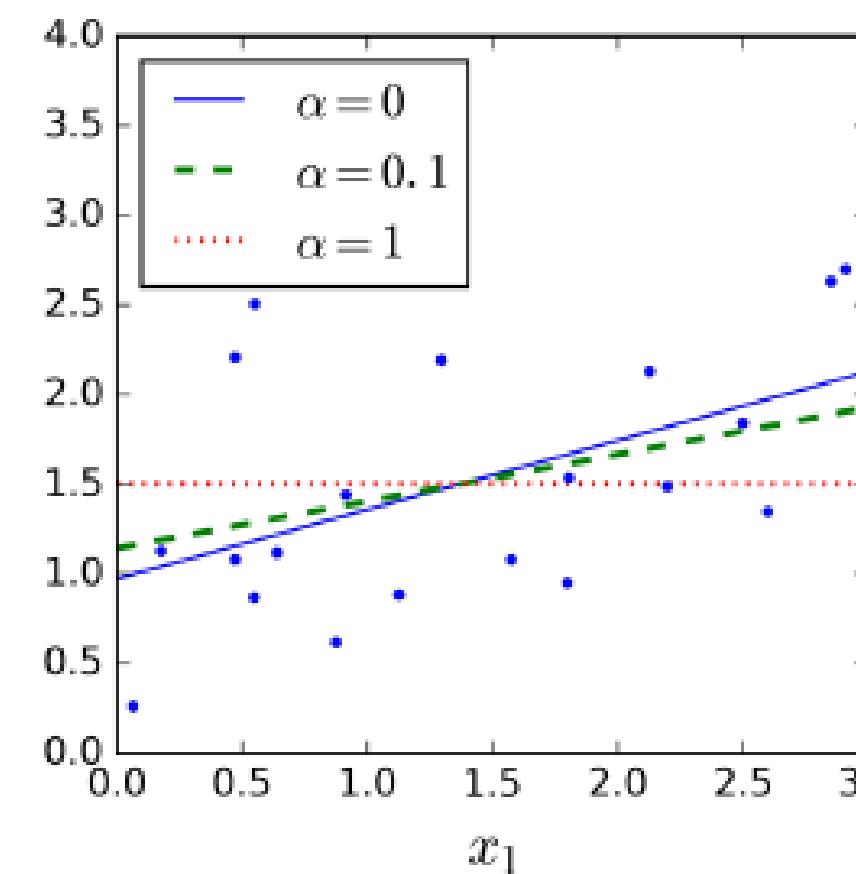


Lasso hace que los pesos W de las variables independientes menos importantes sean iguales a cero, llevando a cabo una forma de selección de variables



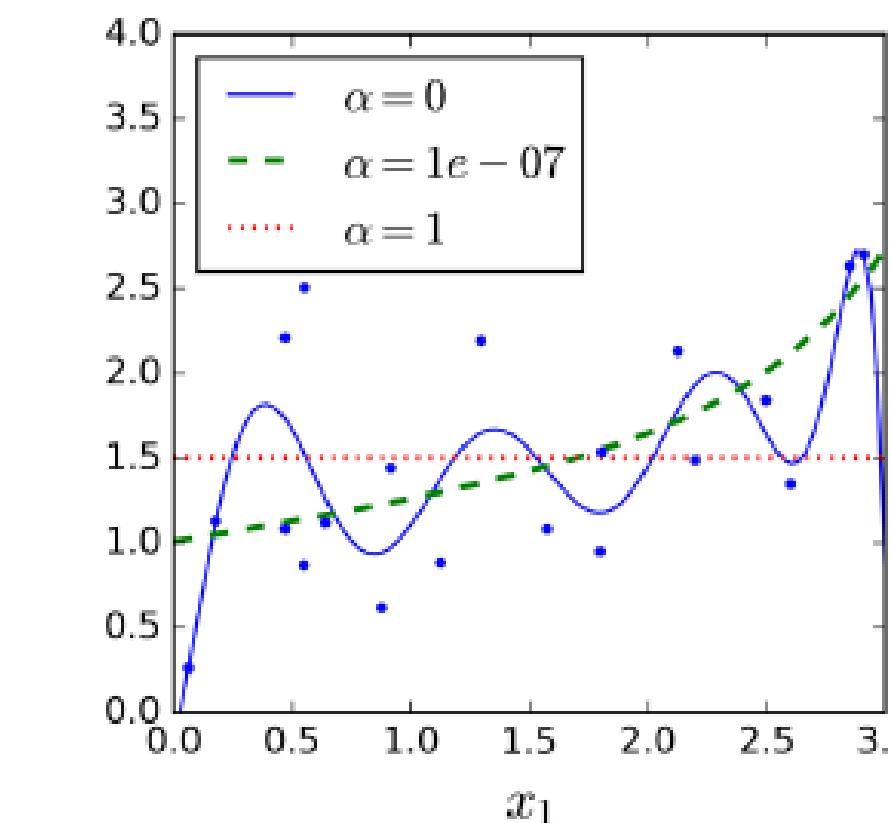
### Lasso sin extender set de datos

- Al incrementar alfa se tienen predicciones más planas, lo cual reduce la varianza pero incrementa el sesgo



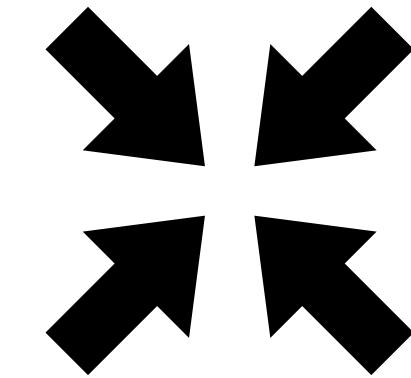
### Lasso con un set de datos extendido (polinomio grado 10)

- Al incrementar alfa se tienen predicciones más planas, lo cual reduce la varianza pero incrementa el sesgo

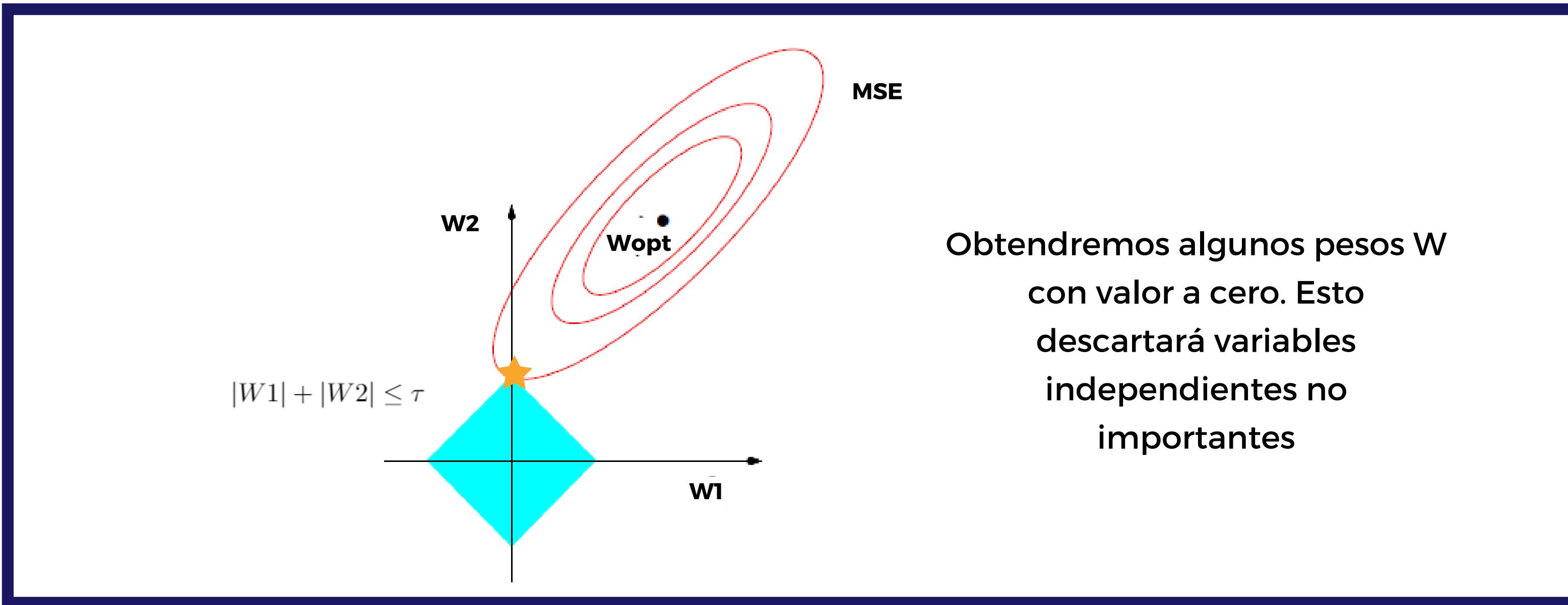


# REGRESIÓN LASSO

## INTERPRETACIÓN GEOMÉTRICA



La regresión Lasso elimina parámetros W de variables independientes no importantes al imponer una penalización (término regularizador)

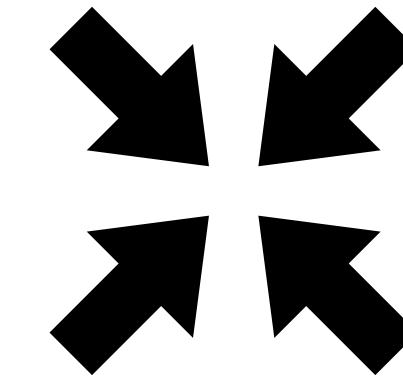


# REGRESIÓN LASSO

## ENTRENAMIENTO DE UN MODELO



Para entrenar un modelo se puede utilizar la librería Sklearn.



### Clase Lasso con parámetros por default

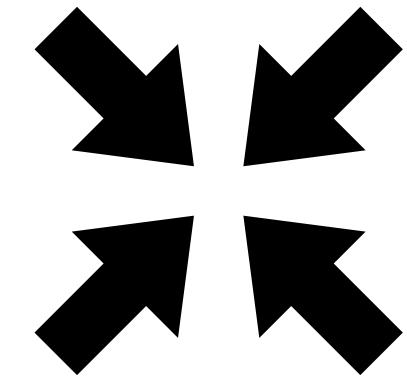
```
from sklearn.linear_model import Lasso
lasso_reg = Lasso(alpha=1.0, *, fit_intercept=True, normalize='deprecated',
                  precompute=False, copy_X=True, max_iter=1000,
                  tol=0.0001, warm_start=False, positive=False,
                  random_state=None, selection='cyclic')
lasso_reg.fit(X_train, y_train)
lasso_reg.predict(X_test)
lasso_reg.score(X_test, y_test)
```

hyper parámetro a optimizar con K-Fold CV

{'cyclic', 'random'}, default='cyclic'

# REGRESIÓN LASSO

## ENTRENAMIENTO DE UN MODELO



Para entrenar un modelo se puede utilizar la librería Sklearn.

### Atributos de clase Lasso

```
lasso_reg.coef_  
lasso_reg.intercept_  
lasso_reg.sparse_coef_
```

Arreglo con pesos W óptimos

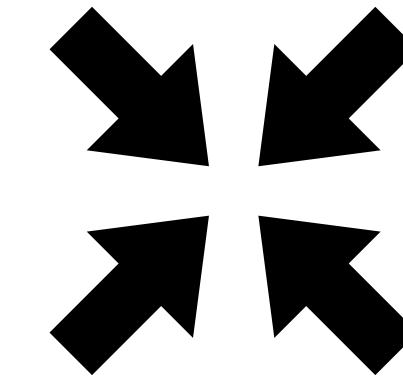
Arreglo o número float con  
intercepto W0 óptimo

¿Para qué sirve el atributo  
`sparse_coef_`?

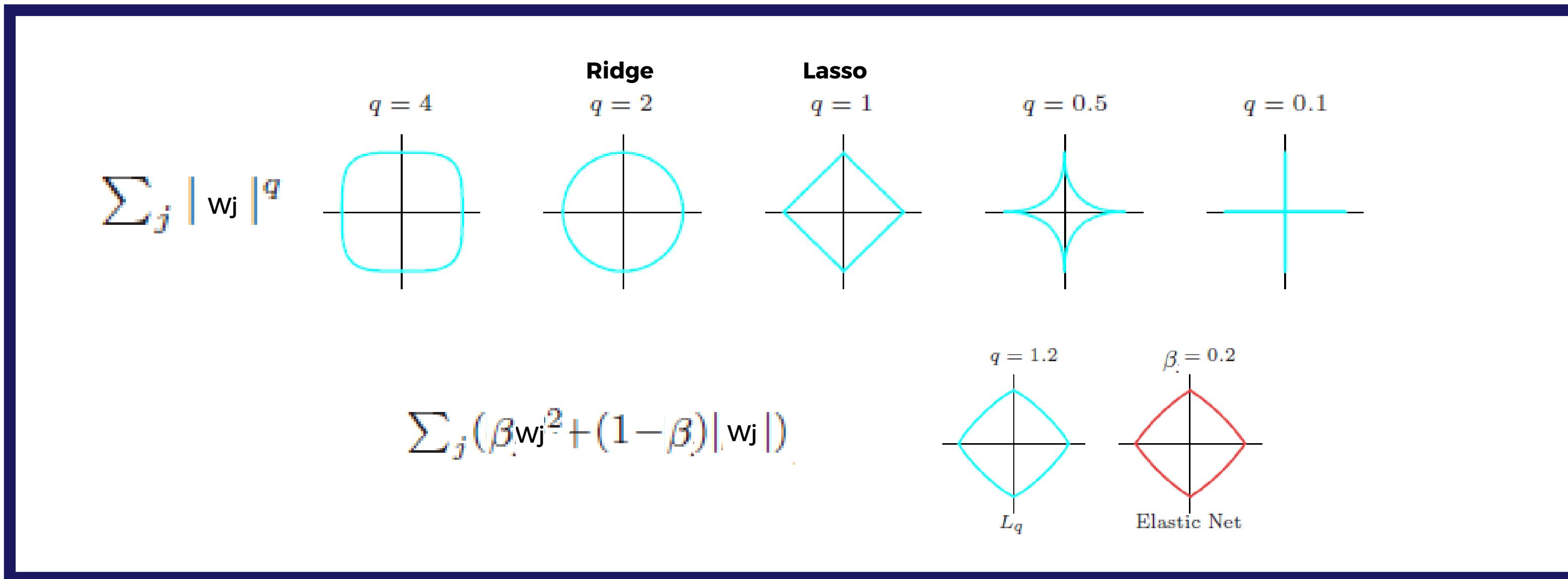
### 3.- Red Elástica

# TÉRMINO REGULARIZADOR

## INTERPRETACIÓN GEOMÉTRICA



🎯 El término regularizador determinará el nombre de la regresión. Para la norma L2 tenemos la regresión Ridge; para la norma L1 es Lasso; y para una combinación de ambas, tenemos la red elástica

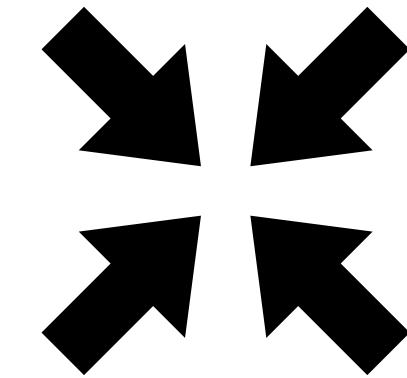


# RED ELÁSTICA

## CONCEPTOS BÁSICOS



Otra versión regularizada de la regresión lineal



### Definición

- Target numérica → Regresión
- Un set de datos está compuesto por el par  $(\mathbf{x}_n, y_n)$
- Función de costo

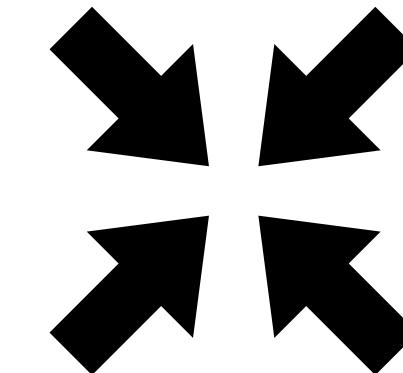
$$\mathcal{L}(\mathbf{W}) = \frac{1}{N} \sum_{i=1}^N (y_i - \mathbf{x}_n^T \mathbf{W})^2 + \alpha \left( \sum_j (\beta w_j^2 + (1-\beta) |w_j|) \right)$$

### Características

- Mezcla entre Ridge y Lasso, el parámetro beta ayuda a controlarla
- Ayuda a mitigar el sobreajuste
- Así como Lasso, también tiende a eliminar pesos  $\mathbf{W}$  de menor importancia para el modelo (lleva dichos pesos a cero)
- Es recomendable utilizar una Red elástica en lugar de Lasso, especialmente cuando se tienen más variables independientes que ejemplos en el set de entrenamiento o cuando éstas fuertemente correlacionadas.

# RED ELÁSTICA

## ENTRENAMIENTO DE UN MODELO



Para entrenar un modelo se puede utilizar la librería Sklearn.

### Clase ElasticNet con parámetros por default

hyper parámetro a optimizar con K-Fold CV

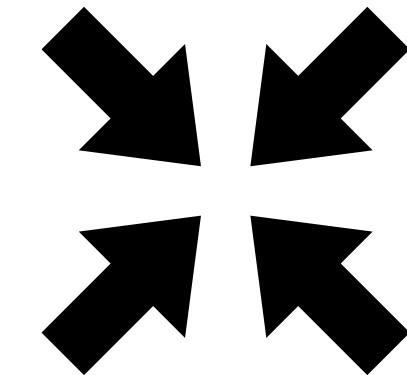
hyper parámetro para determinar mezcla entre Ridge y Lasso

```
from sklearn.linear_model import ElasticNet
elastic_net = ElasticNet(alpha=1.0, *, l1_ratio=0.5, fit_intercept=True, normalize='deprecated',
                         precompute=False, max_iter=1000, copy_X=True, tol=0.0001,
                         warm_start=False, positive=False, random_state=None, selection='cyclic')
elastic_net.fit(X_train, y_train)
elastic_net.predict(X_test)
elastic_net.score(X_test, y_test)
```

{'cyclic', 'random'}, default='cyclic'

# RED ELÁSTICA

## ENTRENAMIENTO DE UN MODELO



Para entrenar un modelo se puede utilizar la librería Sklearn.

### Atributos de clase ElasticNet

```
elastic_net.coef_
```

Arreglo con pesos W óptimos

```
elastic_net.intercept_
```

Arreglo o número float con  
intercepto W0 óptimo

```
elastic_net.sparse_coef_
```

# 4.- Regresión Logística



# CLASIFICACIÓN

## DEFINICIÓN



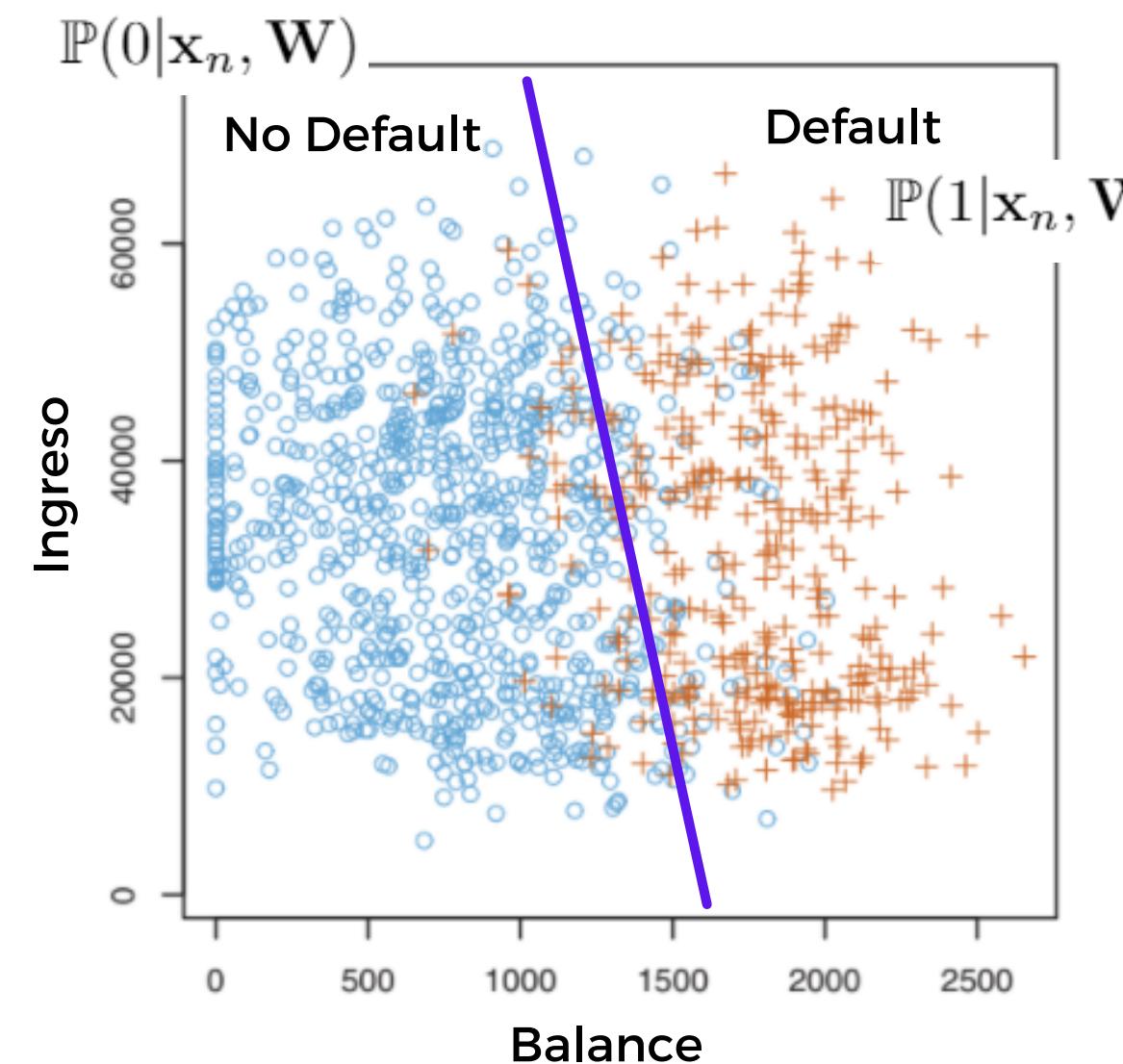
Similar a los problemas de regresión, los problemas de clasificación relacionan un input con la variable output, pero la variable respuesta es categórica



## Clasificadores

Dividen el espacio de variables independientes en una colección de regiones para cada clase en la variable respuesta

¿Qué tipos de problemas de clasificación existen?



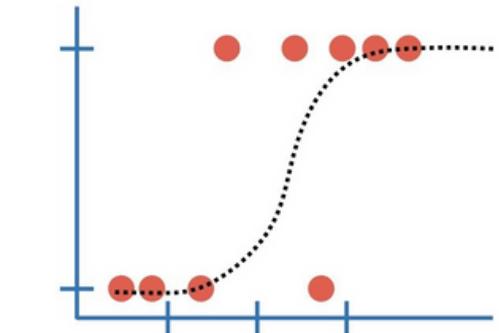
Un modelo de clasificación buscará obtener la probabilidad del evento en cuestión

# REGRESIÓN LOGÍSTICA

## UNA CLASIFICACIÓN NO ES UN CASO ESPECIAL DE UN MODELO DE REGRESIÓN

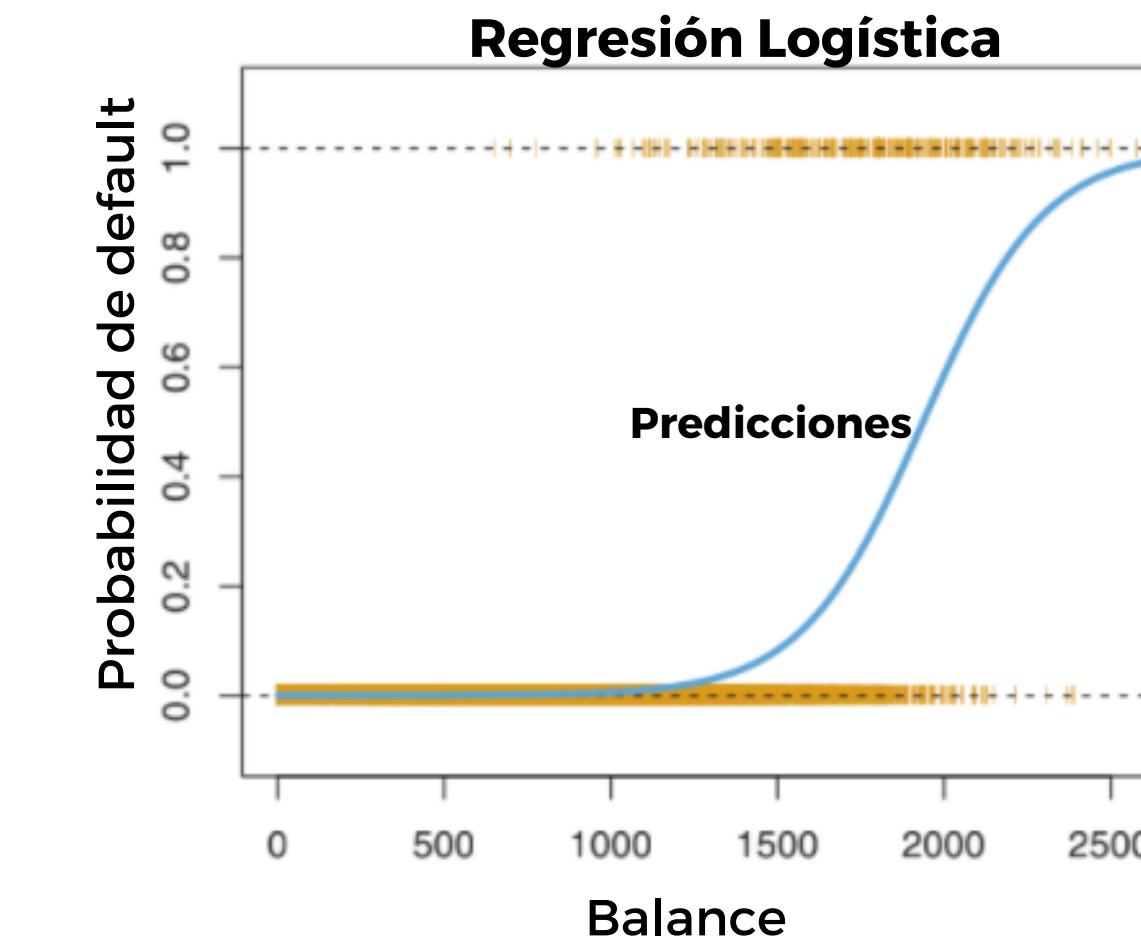
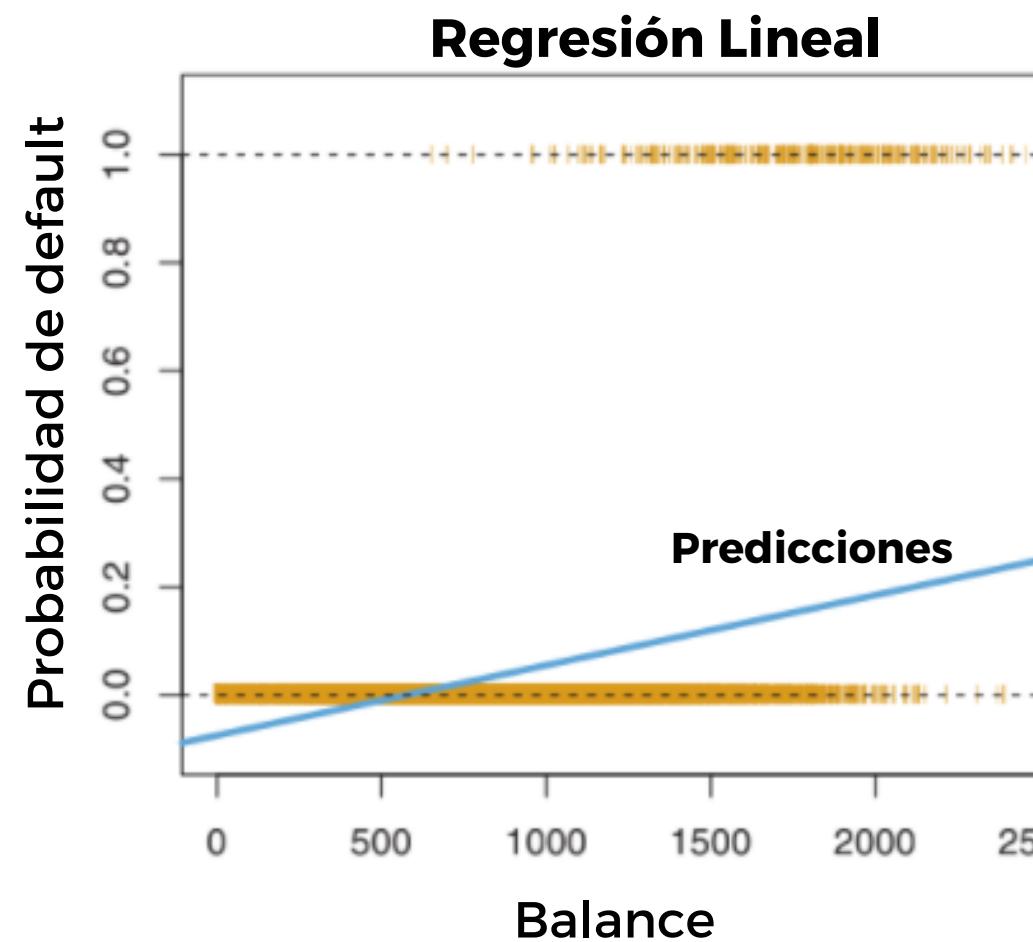


Justo como la regresión lineal, calcula la suma ponderada de las variables independientes, pero transforma el output utilizando la función logística



### Intuición

¿Por qué las predicciones de una regresión lineal no sirve para predecir la probabilidad de default?

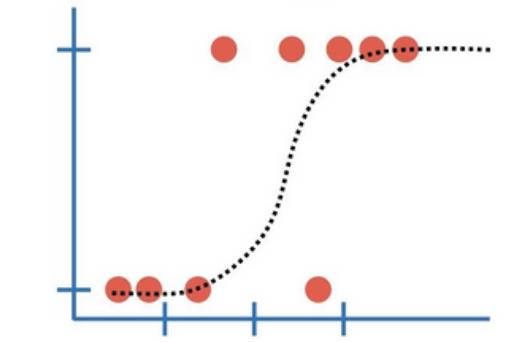


# REGRESIÓN LOGÍSTICA

## CONCEPTOS BÁSICOS



La intuición es transformar las predicciones de una regresión lineal utilizando la función logística (una especie de función sigmoide) para poder obtener la probabilidad  $\mathbb{P}(1|x_n, W)$



### Definición

- Target categórica → Clasificación
- Un set de datos está compuesto por el par  $(x_n, y_n)$
- Función de costo

$$\mathcal{L}(W) = \sum_{n=1}^N \ln[1 + \exp(x_n^T W)] - y_n x_n^T W$$

- Utilizamos la función logística para transformar  $x_n^T W$  en probabilidades

$$\sigma(x_n^T W) = \frac{1}{1 + \exp(-x_n^T W)}$$

### Características

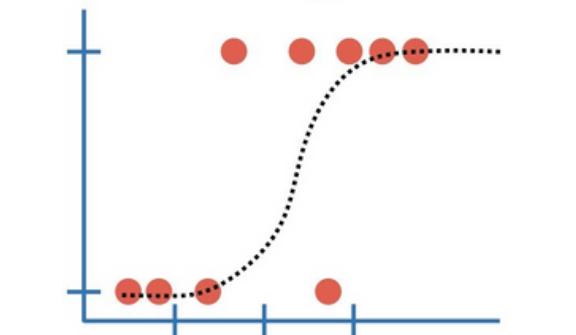
- Función de costo es convexa ¿Qué método podrías usar para optimizarla?
- Otros métodos de optimización: Método de Newton, Hessiana de la función Log-likelihood
- Se pueden ocupar métodos de regularización. ¿Cuáles conoces?
- Si los datos son linealmente separables, es necesario utilizar un término de regularización, de lo contrario, no existirán pesos  $W$  finitos que minimicen la función de costo
- Es uno de los métodos más comúnmente usados

# REGRESIÓN LOGÍSTICA

## INTERPRETACIÓN GEOMÉTRICA

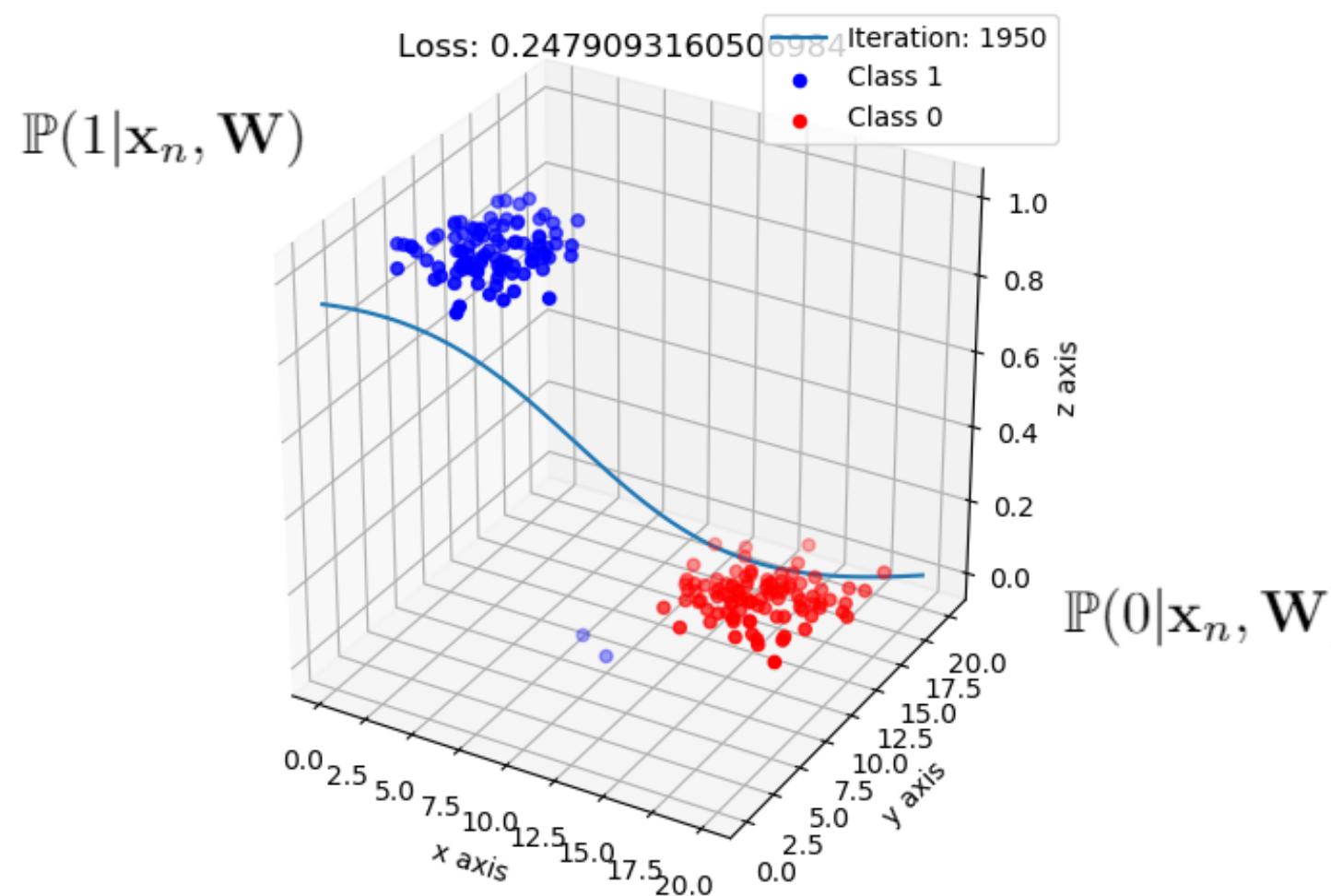


El objetivo es encontrar el o los hyper planos necesarios para partir el espacio de variables input en una colección de regiones para las categorías de la variable respuesta

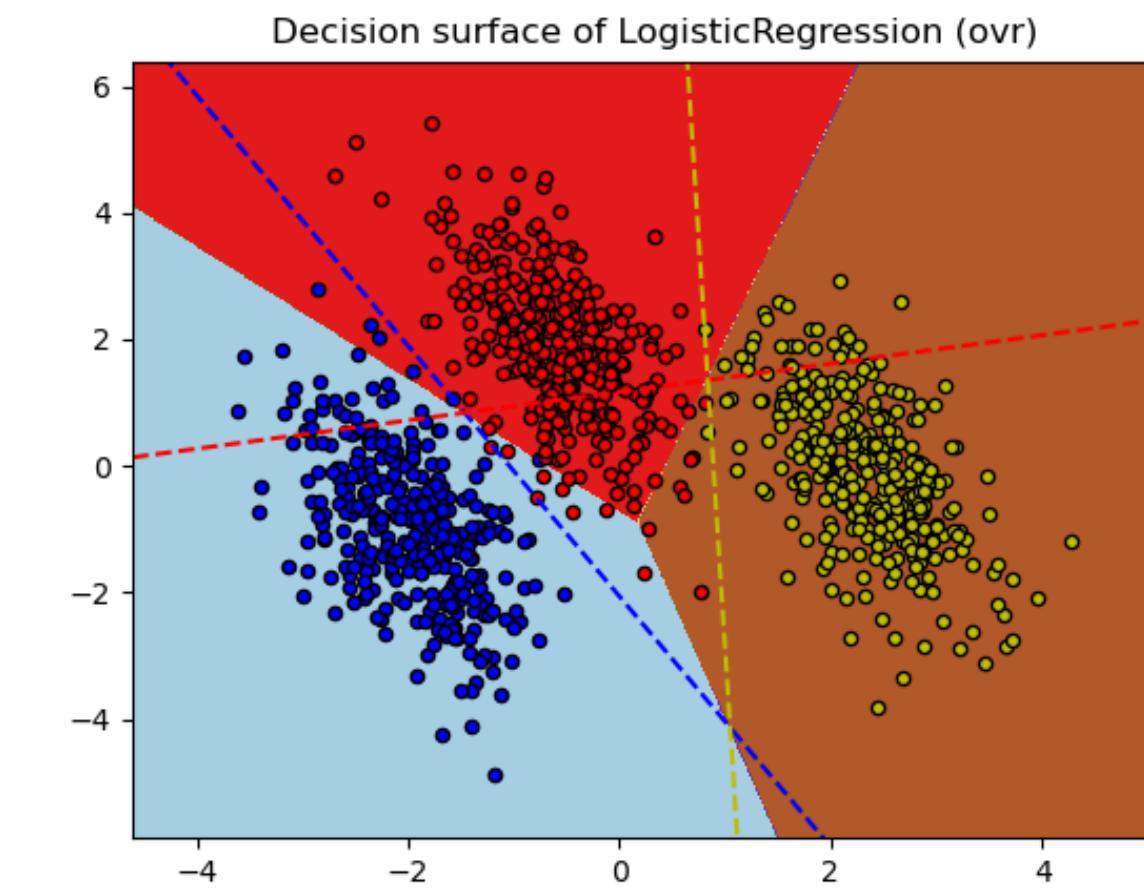


### Intuición geométrica\*

Clasificación Binaria



Clasificación Multiclas

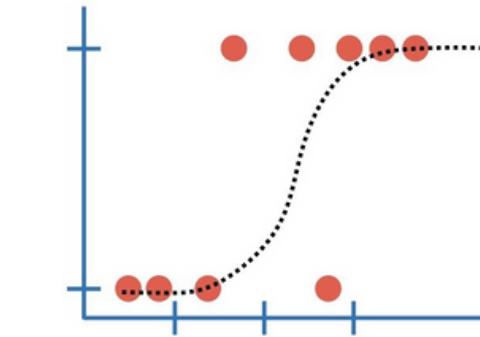


# REGRESIÓN LOGÍSTICA

## ENTRENAMIENTO DE UN MODELO



Para entrenar un modelo se puede utilizar la librería Sklearn.



### Clase LogisticRegression con parámetros por default

```
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(penalty='l2', *, dual=False, tol=0.0001,
                               C=1.0, fit_intercept=True, intercept_scaling=1,
                               class_weight=None, random_state=None, solver='lbfgs',
                               max_iter=100, multi_class='auto', verbose=0,
                               warm_start=False, n_jobs=None, l1_ratio=None)
```

hyper parámetro a optimizar con K-Fold CV

Parámetro que define el tipo de regularización

Tipo de solver a utilizar. Depende directamente del tipo de regularización que ocupemos

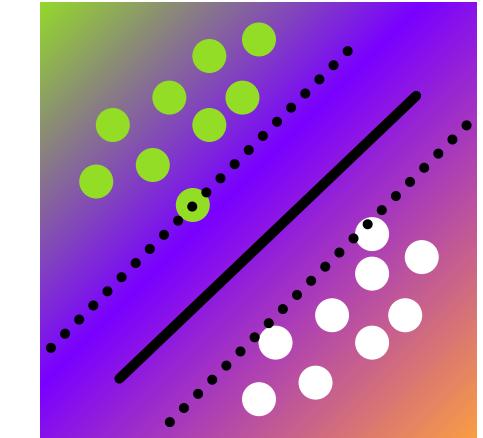
# 5.- Máquinas Vector Soporte

# MÁQUINAS DE SOPORTE VECTORIAL

## CONCEPTOS BÁSICOS



Este modelo intenta separar el espacio de variables con hyper planos y un margen alrededor de ellos



### Definición

- Target numérica/categórica → Regresión/Clasificación
- Un set de datos está compuesto por el par  $(\mathbf{x}_n, y_n)$
- Problema de optimización

$$\min_{\mathbf{w}} \sum_{n=1}^N [1 - y_n \mathbf{x}_n^\top \mathbf{w}]_+ + \frac{\lambda}{2} \|\mathbf{w}\|^2$$
$$[1 - yz]_+ = \max\{0, 1 - yz\}$$

### Características

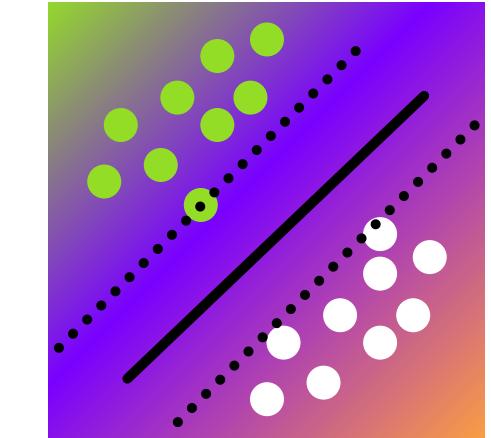
- Algoritmo de aprendizaje de máquina muy frecuentemente usado
- Puede ser usado para llevar a cabo tareas de clasificación lineal y no lineal, regresión, y detección de outliers
- Funciona muy bien para tareas de clasificación en sets de datos pequeños o medianos
- Funciona mejor cuando se tiene un espacio linealmente separable ¿Qué es esto?
- Este clasificador no devuelve la probabilidad para cada clase a predecir
- No existe una interpretación probabilística obvia
- La extensión a clasificación de multclases no es trivial

# MÁQUINAS DE SOPORTE VECTORIAL

## INTUICIÓN - MODELO PARA CLASIFICAR DATOS



Este modelo intenta ajustar una "amplia calle" entre clases. Dicha calle se llama gran margen de clasificación



### Intuición gráfica\*

- Las observaciones situadas en el borde del margen de clasificación ayudarán al modelo a determinar la región de decisión

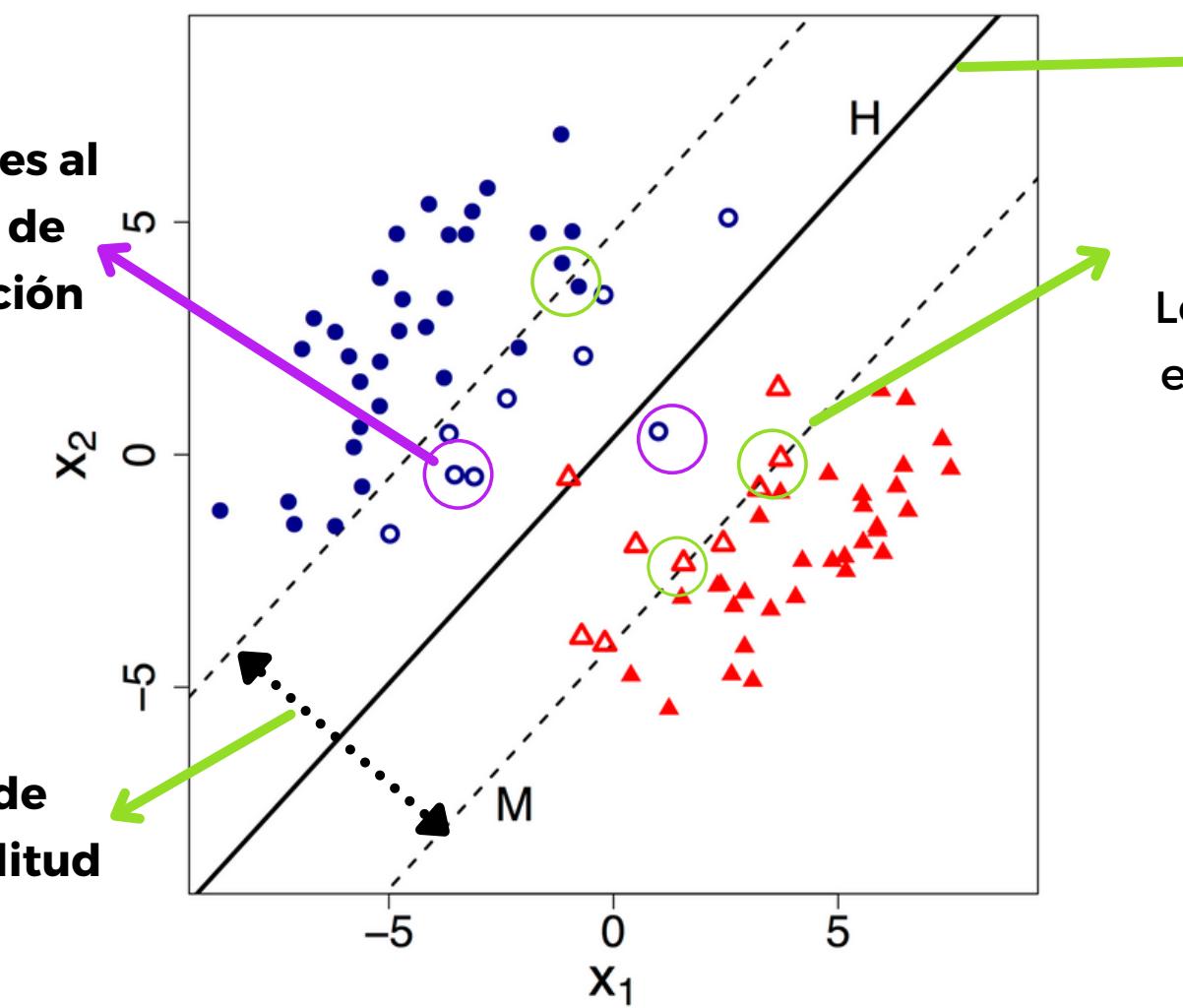


#### Objetivo

Encontrar el margen de clasificación más grande que podamos, limitando las veces que una observación viola dicho margen (cuando una observación que a mitad de la calle o incluso queda al otro lado)

Margin amplio de clasificación (amplitud de la calle)

Violaciones al margen de clasificación



Hyper plano óptimo

Vectores de soporte

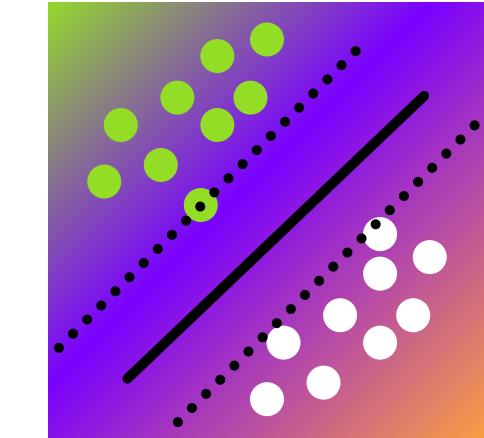
Los límites de la región de decisión están totalmente determinados (o soportados) por estos vectores

# MÁQUINAS DE SOPORTE VECTORIAL

## TIPOS DE MARGENES

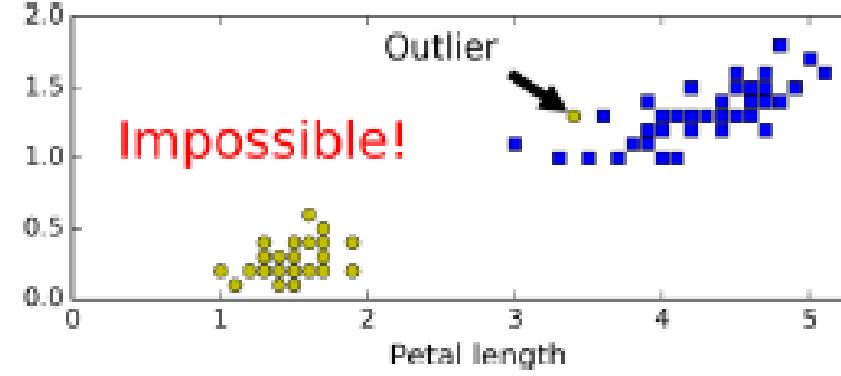
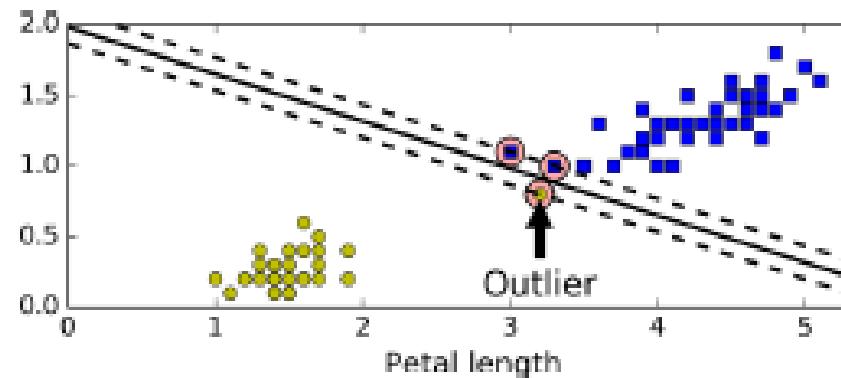


Utilizar un margen "duro" o poco flexible puede hacer que el modelo sea vulnerable ante la presencia de outliers



### Margen duro

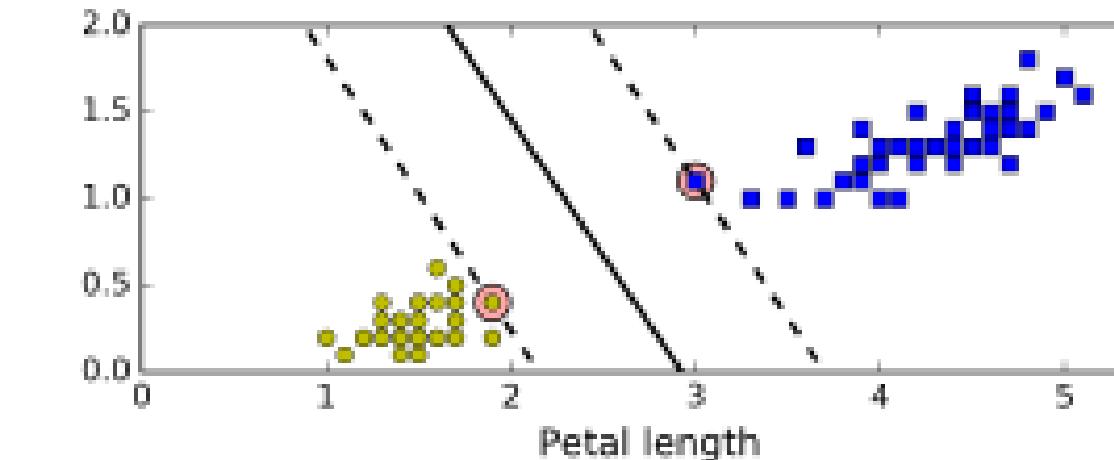
- Es cuando tratamos de hacer que todas las observaciones queden fuera del margen de clasificación



- **Problema 1:** cuando el espacio no es linealmente separable
- **Problema 2:** cuando existen outliers

### Margen suave

- Para evitar los problemas anteriores, podemos utilizar un margen suave que ayude a encontrar la mayor amplitud para el mismo, y limitando el número de veces que observaciones que quedan en medio, o que terminan al otro lado

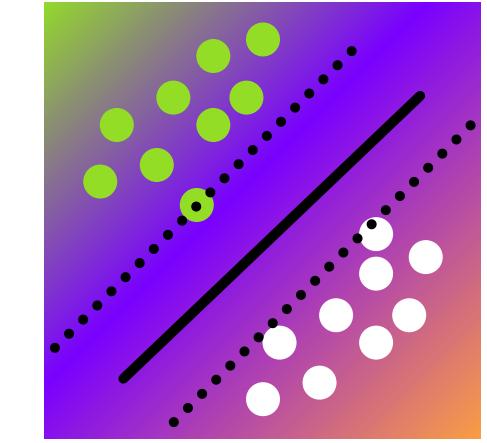


# MÁQUINAS DE SOPORTE VECTORIAL

## COSAS QUE PUEDEN AFECTAR

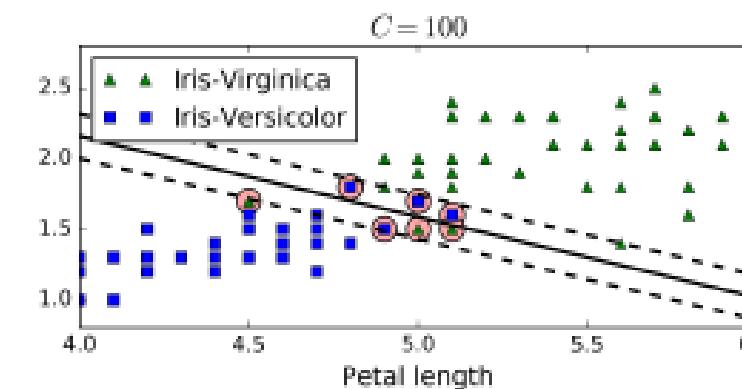
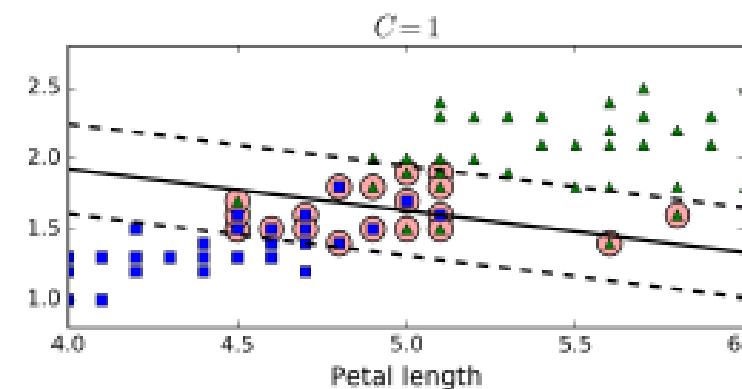


Cuando un modelo de máquinas de soporte vectorial presenta sobreajuste a los datos de entrenamiento, se puede regularizar al reducir un hyper parámetro llamado "C"



### Margin poco amplio

- Puede llevar a que un modelo no generalice de manera correcta a nuevos sets de datos (sobreajuste). Lo anterior se puede mitigar al optimizar el hyper parámetro C

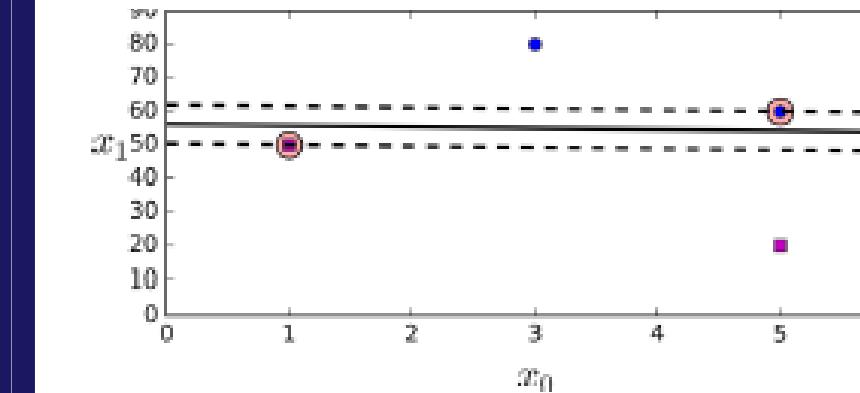


- Si C es bajo, tendremos un mayor número de violaciones al margen, pero el modelo generalizará mejor
- Si C es alto, nuestro margen será menos amplio y tendrá menos violaciones. Sin embargo, no generalizará bien

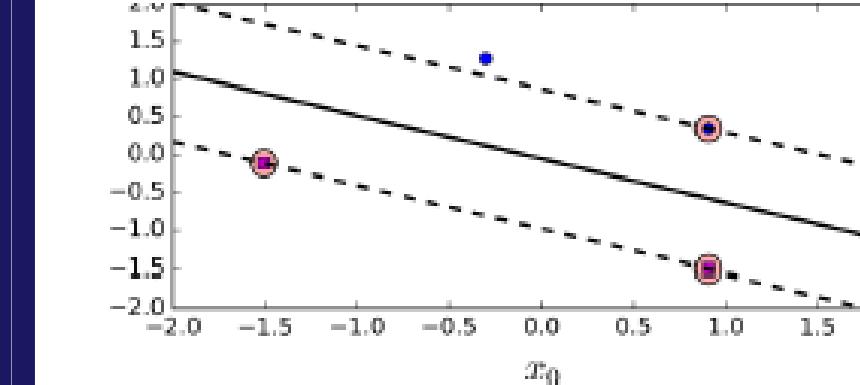
### Sensibilidad a la escala de las variables

- Para evitar los problemas anteriores

#### Variables no estandarizadas



#### Variables estandarizadas



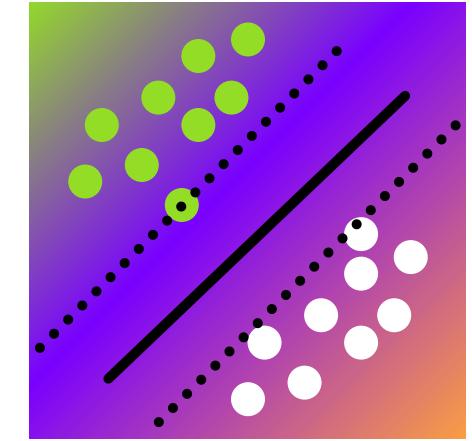
- Este modelo es sensible a variables con escalas muy diferentes ¿Por qué?

# MÁQUINAS DE SOPORTE VECTORIAL

## TRUCO DEL KERNEL



Encontrar una forma equivalente de escribir el problema de optimización de una máquina de soporte vectorial, y que la solución esté "kernelizada"



**¿Cómo podríamos encontrar los parámetros óptimos para el problema de optimización de una máquina de soporte vectorial?**

- Utilizando el algoritmo de Gradiente estocástico descendente con subgradientes (función máxima)

$$\min_{\mathbf{w}} \sum_{n=1}^N [1 - y_n \mathbf{x}_n^\top \mathbf{w}]_+ + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

$$[1 - yz]_+ = \max\{0, 1 - yz\}$$



Truco del Kernel

Pero!

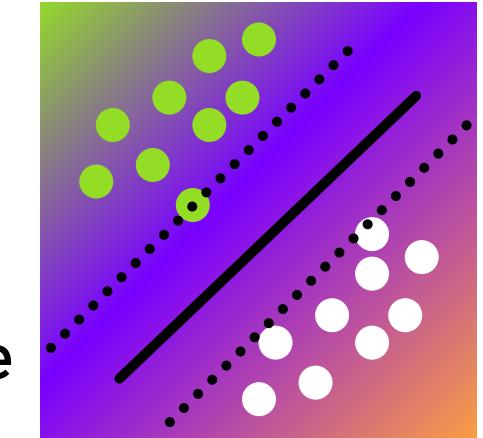
Para algunos casos, es más eficiente optimizar el problema dual

# MÁQUINAS DE SOPORTE VECTORIAL

## TRUCO DEL KERNEL



Algunas veces es más eficiente resolver la forma dual que el primal de un problema de optimización



### Problema Primal

$$\min_{\mathbf{w}} \sum_{n=1}^N [1 - y_n \mathbf{x}_n^\top \mathbf{w}]_+ + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

$$[1 - yz]_+ = \max\{0, 1 - yz\}$$

### Problema Dual

$$\max_{\alpha \in [0,1]^N} \min_{\mathbf{w}} \sum_{n=1}^N \underbrace{\alpha_n(1 - y_n \mathbf{x}_n^\top \mathbf{w})}_{\text{Función convexa}} + \frac{\lambda}{2} \|\mathbf{w}\|^2.$$

Función convexa

¿Cómo llegamos a este formulación?

$$[z]_+ = \max\{0, z\} = \max_{\alpha \in [0,1]} \alpha z.$$

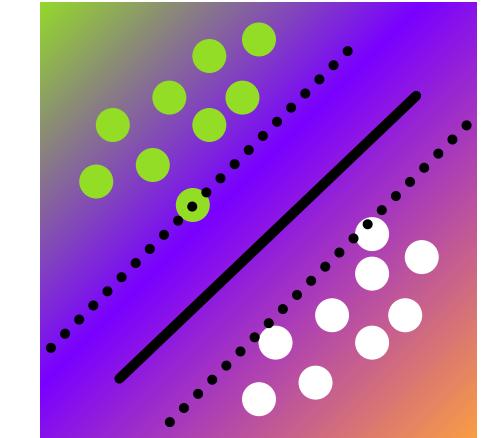
$$[1 - y_n \mathbf{x}_n^\top \mathbf{w}]_+ = \max_{\alpha_n \in [0,1]} \alpha_n(1 - y_n \mathbf{x}_n^\top \mathbf{w}).$$

# MÁQUINAS DE SOPORTE VECTORIAL

## TRUCO DEL KERNEL



La solución del problema de optimización dual utilizará los datos del set de variables independientes en una forma kernelizada



### Solución al Problema Dual

$$\max_{\alpha \in [0,1]^N} \min_w \sum_{n=1}^N \alpha_n (1 - y_n x_n^\top w) + \frac{\lambda}{2} \|w\|^2.$$

Obteniendo la derivada con respecto de  $W$ , igualándola a cero y demás pasos... encontramos esta solución:

$$\max_{\alpha \in [0,1]^N} \alpha^\top 1 - \frac{1}{2\lambda} \alpha^\top Y \boxed{XX^\top} Y \alpha$$

Kernel

#### Tipos de Kernel

Lineal, cuadrático, cúbico, exponencial, sigmoide

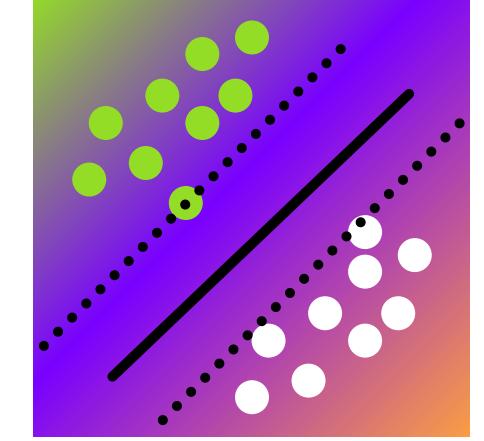


#### Truco del Kernel

Nos permitirá aumentar o expandir el set de datos  $X$  con diferentes funciones sin incurrir en ningún costo extra

# MÁQUINAS DE SOPORTE VECTORIAL

SHOW TIME!



Show time!