Argentina programa 4.0

Universidad Nacional de San Martín

Módulo 3
Aprendizaje Automático

Argentina programa 4.0

Universidad Nacional de San Martín

Módulo 3
Aprendizaje Automático

Semana 3. *Support Vector Machines (SVM)*

Contenidos del módulo

ML Clásico

- Árboles de Decisión
- Métodos de Ensemble
  - Bagging / Pasting —> Random Forests
  - Boosting
- Support Vector Machines

Deep Learning

- Redes Neuronales
- Redes Neuronales Convolucionales
- Auto-Encoders / Auto-Encoders Variacionales
- Redes Neuronales Recurrentes (LSTM, otras)
- Extras:
  - Generative Adversarial Networks (GAN)
  - Reinforcement Learning

Contenidos del módulo

ML Clásico

- Árboles de Decisión
- Métodos de Ensemble
  - Bagging / Pasting —> Random Forests
  - Boosting
- Support Vector Machines

Deep Learning

- Redes Neuronales
- Redes Neuronales Convolucionales
- Auto-Encoders / Auto-Encoders Variacionales
- Redes Neuronales Recurrentes (LSTM, otras)
- Extras:
  - Generative Adversarial Networks (GAN)
  - Reinforcement Learning

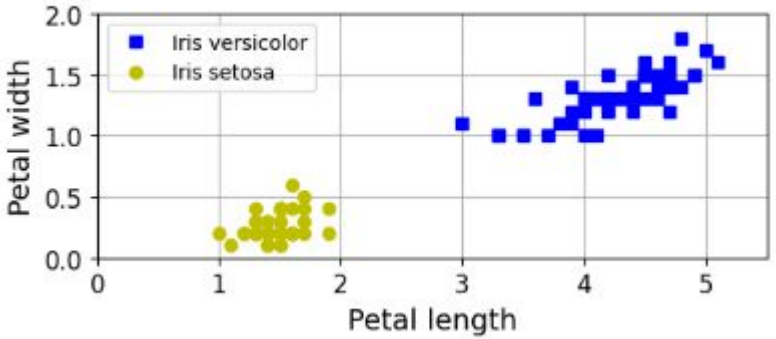
Support Vector Machines (SVM)

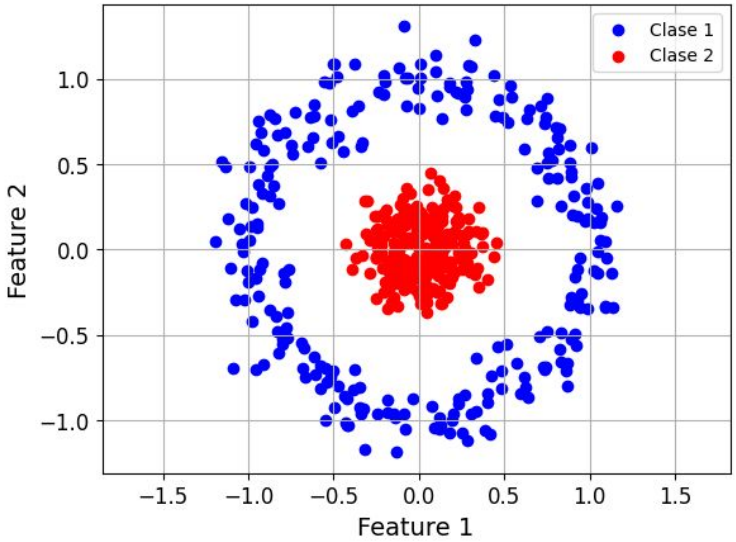
Las Máquinas de Vectores de Soporte o *Support Vector Machine (SVM)* es un modelo de aprendizaje automático versátil y poderoso, capaz de realizar clasificación lineal/no lineal y regresión.
Es muy útil, principalmente para clasificación, en conjuntos de datos pequeños o medianos (cientos o miles de instancias)

- `Pros`: Fácil de usar, útil para baseline
- `Contras`: No escala bien en muchos datos

Support Vector Machines (SVM)

- ¿Se acuerdan qué significaba **linealmente separable**?

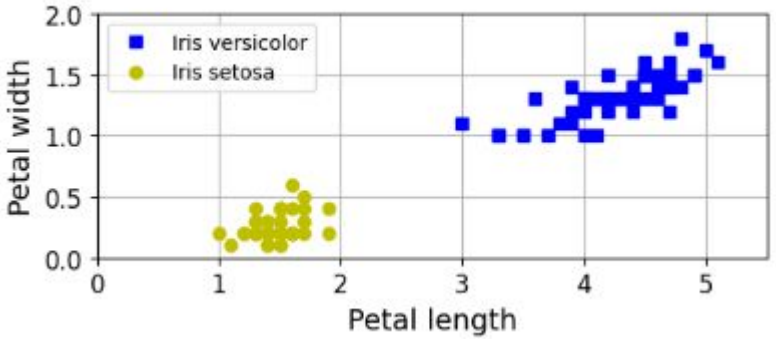


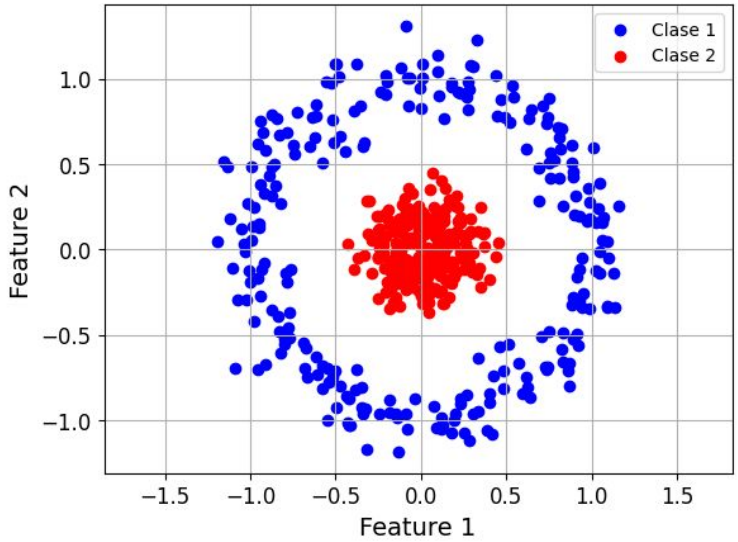


Imágenes adaptadas del Géron

Support Vector Machines (SVM)

- ¿Se acuerdan qué significaba **linealmente separable**?
- Era “si podemos trazar una línea que separe **perfectamente** las clases”
- Vamos a pensar también en una *buena separación*

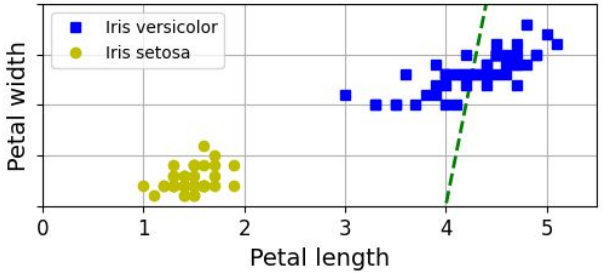




Imágenes adaptadas del Géron

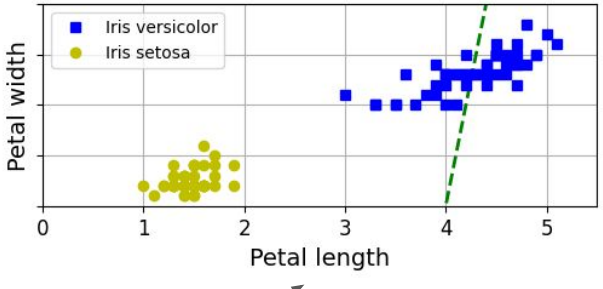
Support Vector Machines (SVM)

¿Qué es un **buena separación**?



Support Vector Machines (SVM)

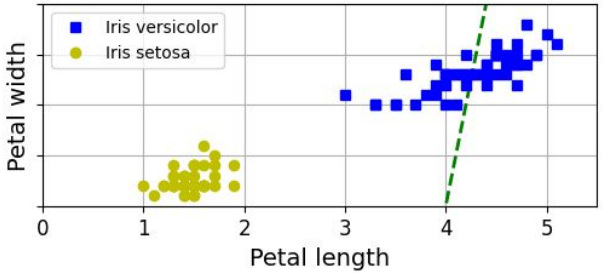
¿Qué es un **buena separación**?



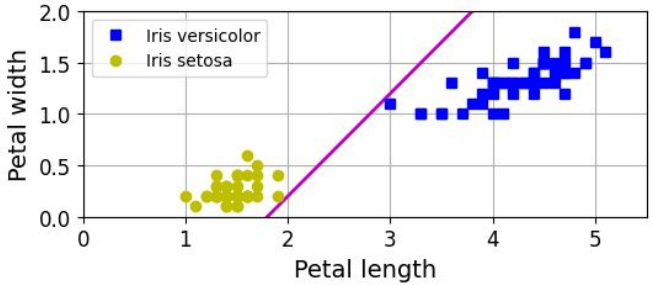
Ni siquiera separa bien

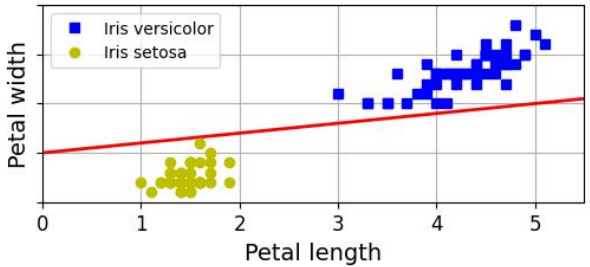
Support Vector Machines (SVM)

¿Qué es un **buena separación**?



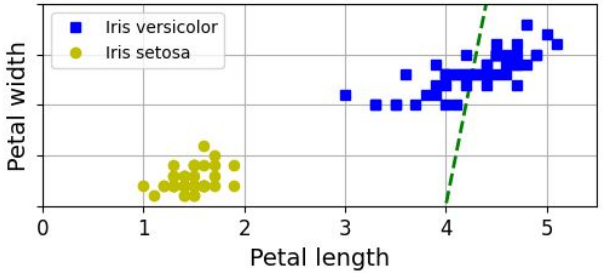
Ni siquiera separa bien





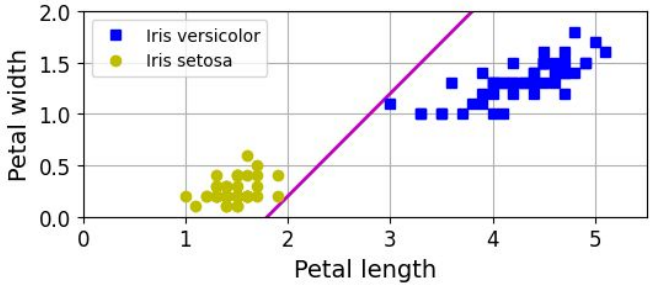
Support Vector Machines (SVM)

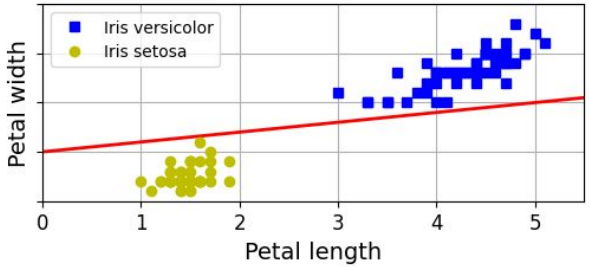
¿Qué es un **buena separación**?



Ni siquiera separa bien

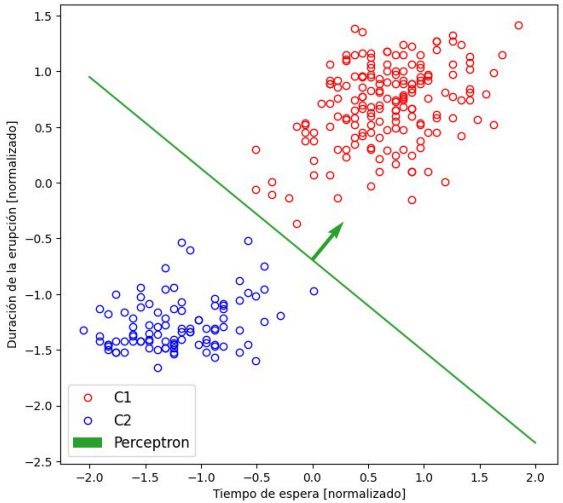
Separan bien, pero esto es el conjunto de entrenamiento, ¿qué puede pasar con datos nuevos?

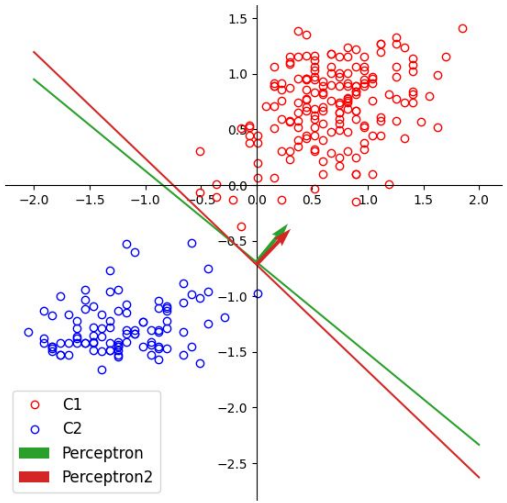


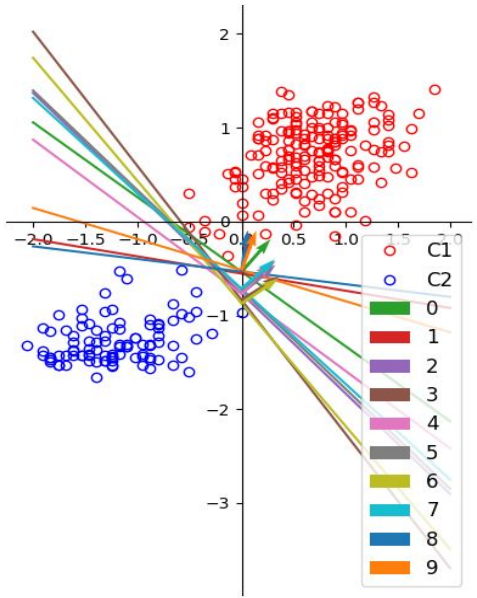


Support Vector Machines (SVM)

Ya se toparon este tema ¿se acuerdan del perceptrón y cómo variaba la frontera con la inicialización?



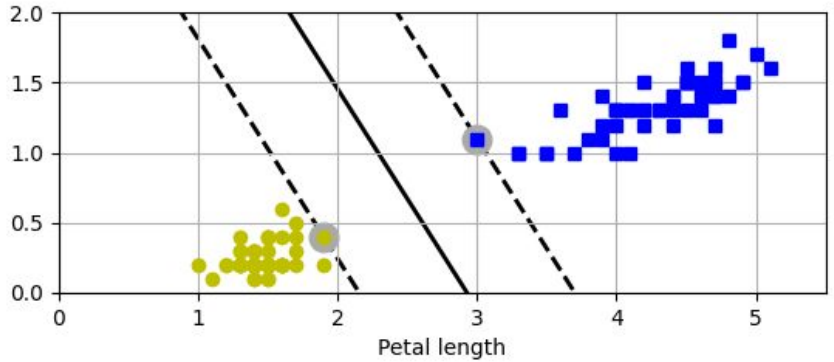




Support Vector Machines (SVM)

Clasificación de margen ancho

- Un SVM busca la separación que se mantiene lo más alejada posible de las instancias de entrenamiento más cercanas.
- Se puede pensar que busca la *calle más ancha* entre los datos
- Añadir más instancias entrenamiento fuera "de la calle" no afecta la frontera, ya que está totalmente determinada por las instancias situadas en el borde
- Estas instancias se denominan **vectores de soporte** (*support vectors*)



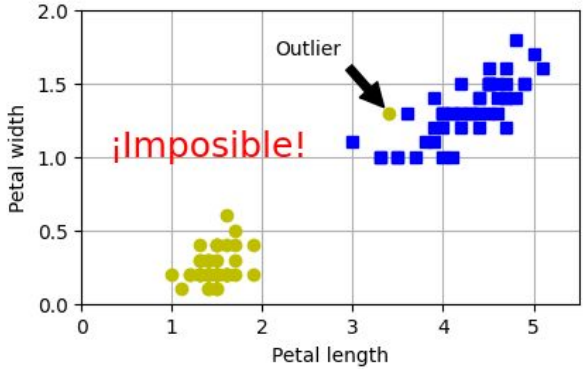
Margen duro: todas las instancias están *fuera de la calle* y del lado correcto

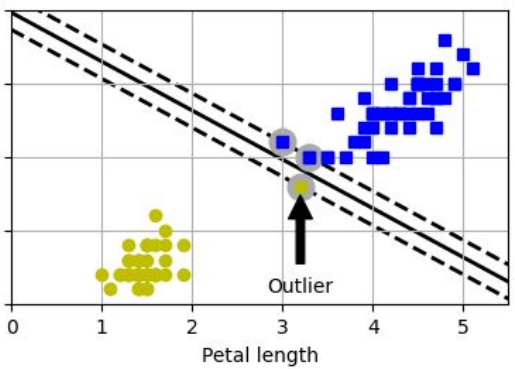
Support Vector Machines (SVM)

Clasificación de margen suave

Usar un margen duro puede traer algunos problemas:

- Solamente funciona si los datos son linealmente separables
- Es sensible a *outliers*
- Hay que flexibilizar un poco el modelo, buscar un buen balance entre “mantener la calle lo más ancha posible” y limitar los puntos que pueden caer fuera de los límites
- Esto se llama **clasificación de margen suave**



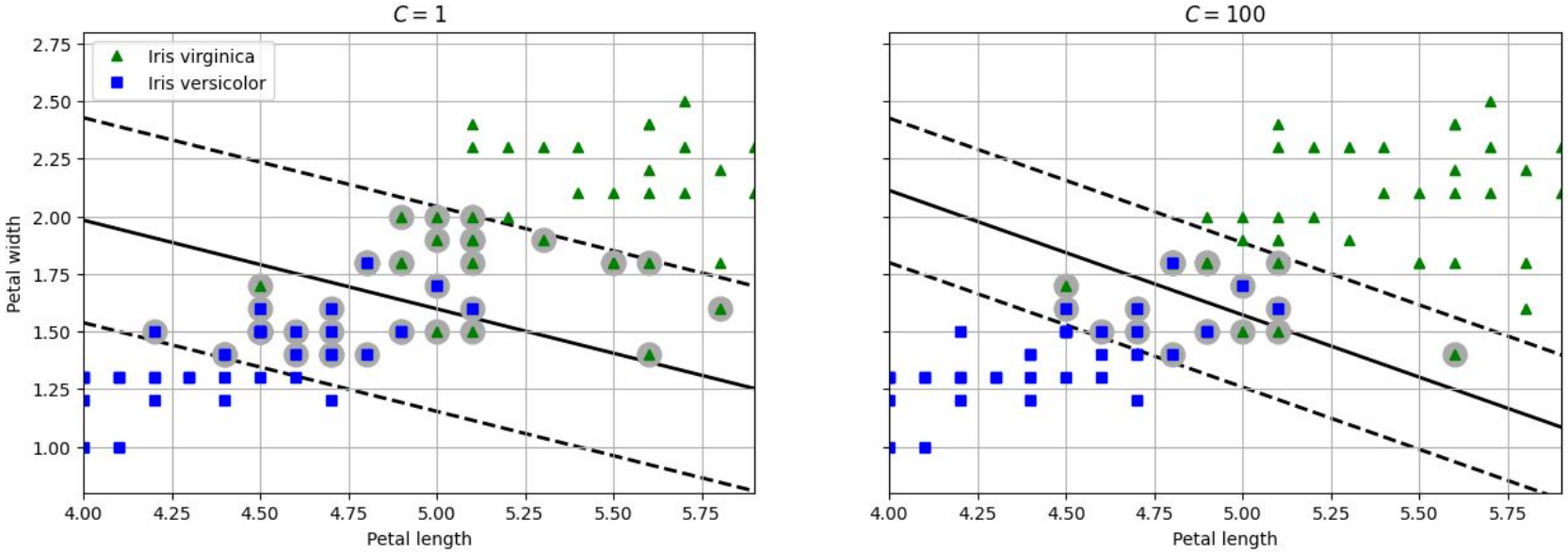


Support Vector Machines (SVM)

Clasificación de margen suave

- El hiperparámetro principal que se pueden especificar cuando sea crea el modelo SVM es el de regularización *C*
- Achicar el *C* hace la calle más ancha, reduciendo el riesgo de sobreajuste al permitir más violaciones al margen, pero si es muy chico puede terminar en un subajuste (underfitting)
- Aumentar el *C* hace lo contrario, endurece el margen haciendo la calle más angosta y por ende permitiendo menos puntos dentro del margen

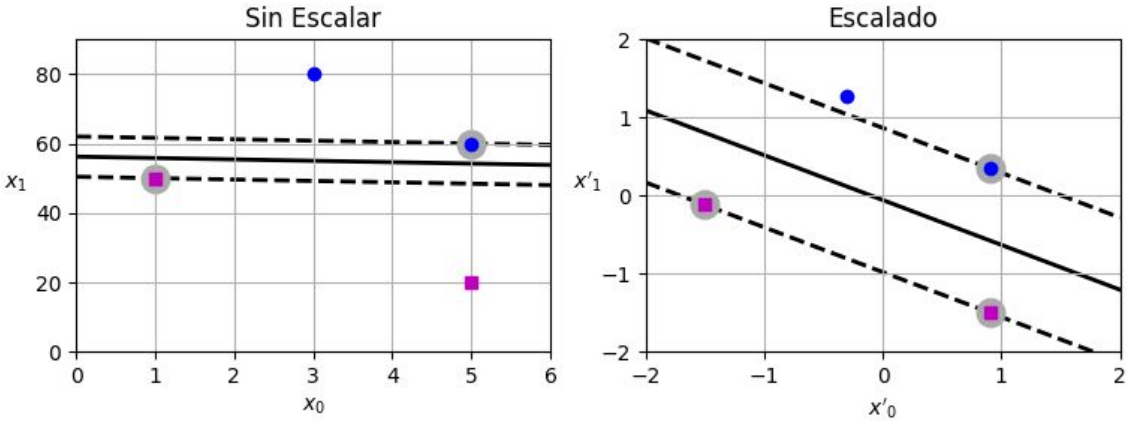
Support Vector Machines (SVM)



Support Vector Machines (SVM)

TIP: Escalar los datos

- Los SVM son sensibles a la escala de las características
- En el gráfico de la izquierda la escala vertical es mucho más grande que la horizontal, entonces la *calle más ancha* es casi horizontal
- Si se hace un escalado, por ejemplo con *StandardScaler,* la frontera queda mucho mejor (Incluso más adelante va a ser mandatorio)



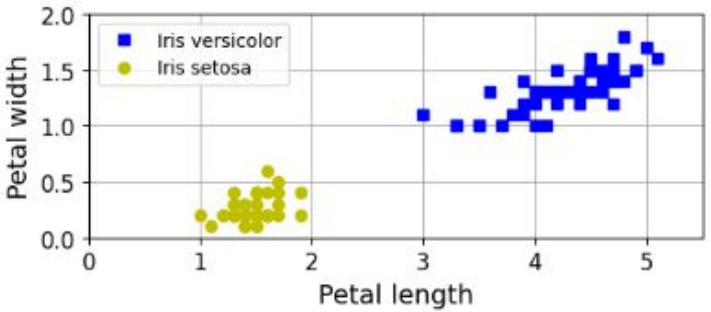

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Una pequeña digresión

(

Una pequeña digresión

- Generalmente los ejemplos que estuvimos viendo durante el curso son de conjuntos de datos más bien pequeños y con ejemplos de gráficos en dos dimensiones
- Esto es por una cuestión de tiempo de exposición y recursos computacionales, pero sobre todo didáctica, una representación visual ayuda a entender que hacen las técnicas y busca desarrollar al menos una intuición de como trabajan con conjuntos de datos más grandes o de dimensionalidad más alta
- Hagamos un pequeño paréntesis para hablar de este tema usando Iris



- Por ejemplo, en estos gráficos sólo estamos viendo dos características para cada flor (y solamente dos especies):

  flor_0 = (1.4, 0.2)
  flor_1 = (1.5, 0.4)
  …

- ¿Pero cuántas características tenía cada flor?

Una pequeña digresión

	sepal_length	sepal_width	petal_length	petal_width	species	species_id
0	5.1	3.5	1.4	0.2	setosa	1
1	4.9	3.0	1.4	0.2	setosa	1
2	4.7	3.2	1.3	0.2	setosa	1
3	4.6	3.1	1.5	0.2	setosa	1
4	5.0	3.6	1.4	0.2	setosa	1
...	...	...	...	...	...	...

Una pequeña digresión

	sepal_length	sepal_width	petal_length	petal_width	species	species_id
0	5.1	3.5	1.4	0.2	setosa	1
1	4.9	3.0	1.4	0.2	setosa	1
2	4.7	3.2	1.3	0.2	setosa	1
3	4.6	3.1	1.5	0.2	setosa	1
4	5.0	3.6	1.4	0.2	setosa	1
...	...	...	...	...	...	...

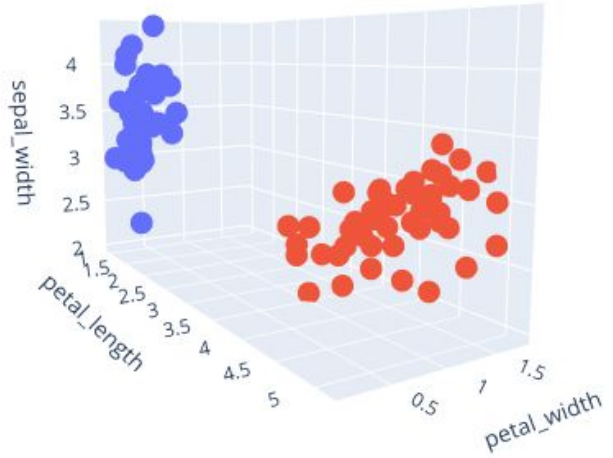
- Cada flor tiene cuatro características numéricas que podemos usar en nuestros análisis (la especie es la etiqueta), usando estas características cada punto *vive* en un espacio de cuatro dimensiones:

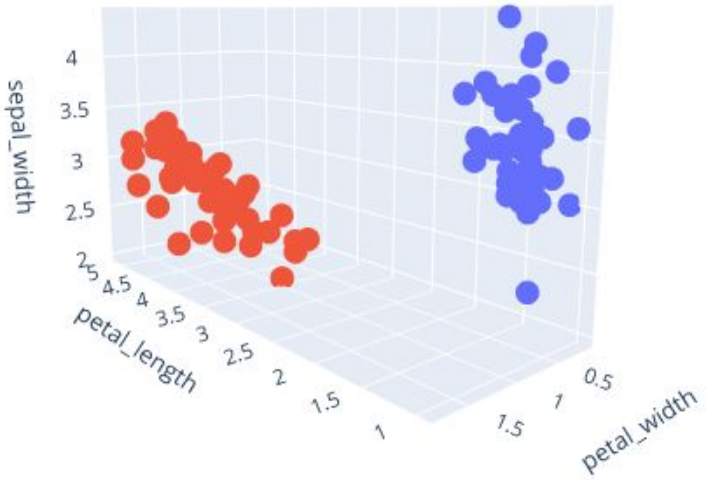
  flor_0 = (5.1, 3.5, 1.4, 0.2)
  flor_1 = (4.9, 3.0, 1.5, 0.4)
  …

- Se complica graficarlos, pero al menos podríamos elegir tres características y ver las flores en un gráfico de tres dimensiones. Por ejemplo tomemos el ancho y el largo del pétalo junto con el ancho del sépalo:

  flor_0 = (3.5, 1.4, 0.2)
  flor_1 = (3.0, 1.5, 0.4)
  …

Una pequeña digresión



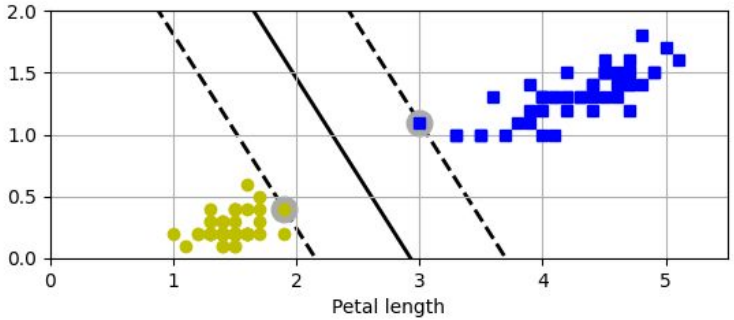


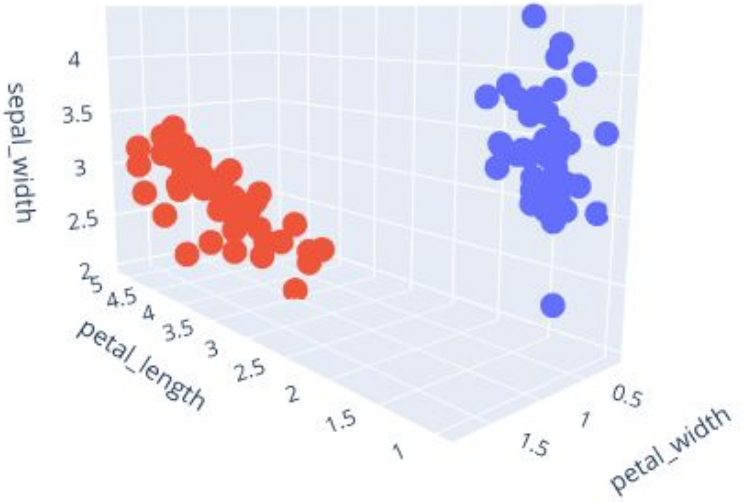
- En el notebook van a poder jugar con esto, elegir otra característica, otra especie, etc.
- Ahora volvamos a los SVM

Una pequeña digresión

La pregunta del millón...

- Dijimos que cuando trabajamos con puntos en dos dimensiones un SVM hace una división como esta:

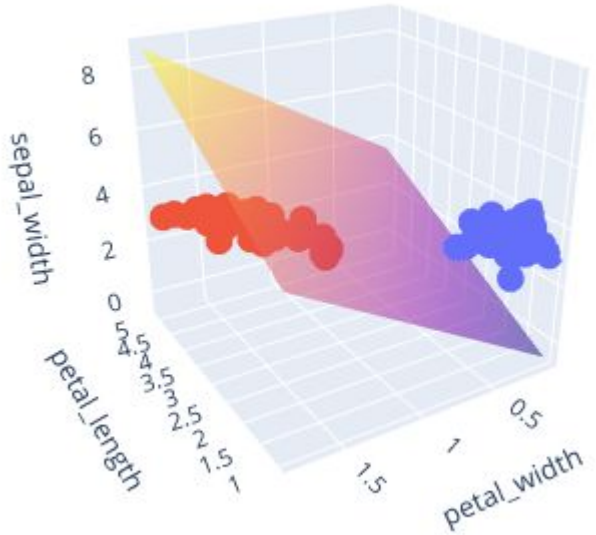


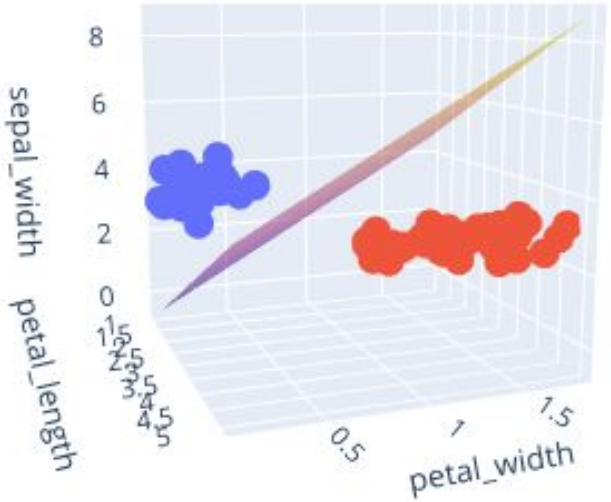


- ¿Se puede usar con estos puntos en tres dimensiones?
- ¿Cómo piensan que se vería la frontera de separación?

Una pequeña digresión

Si pensaron en un plano (vamos a llamarlo *hiperplano,* mejor) acertaron





Una pequeña digresión

¿Y si queremos usar las cuatro características que tiene cada punto del conjunto de datos? (o sea puntos en cuatro dimensiones)

- ¿Podemos usar SVM? ->
- ¿Cómo sería la frontera? ->
- ¿Cómo se vería eso con los puntos? ->

Una pequeña digresión

¿Y si queremos usar las cuatro características que tiene cada punto del conjunto de datos? (o sea puntos en cuatro dimensiones)

- ¿Podemos usar SVM? -> **SIP**
- ¿Cómo sería la frontera? ->
- ¿Cómo se vería eso con los puntos? ->

Una pequeña digresión

¿Y si queremos usar las cuatro características que tiene cada punto del conjunto de datos? (o sea puntos en cuatro dimensiones)

- ¿Podemos usar SVM? -> **SIP**
- ¿Cómo sería la frontera? -> **Un hiperplano en cuatro dimensiones, obvio…**
- ¿Cómo se vería eso con los puntos? ->

Una pequeña digresión

¿Y si queremos usar las cuatro características que tiene cada punto del conjunto de datos? (o sea puntos en cuatro dimensiones)

- ¿Podemos usar SVM? -> **SIP**
- ¿Cómo sería la frontera? -> **Un hiperplano en cuatro dimensiones, obvio…**
- ¿Cómo se vería eso con los puntos? ->

Una pequeña digresión

¿Y si queremos usar las cuatro características que tiene cada punto del conjunto de datos? (o sea puntos en cuatro dimensiones)

- ¿Podemos usar SVM? -> **SIP**
- ¿Cómo sería la frontera? -> **Un hiperplano en cuatro dimensiones, obvio…**
- ¿Cómo se vería eso con los puntos? ->

Por eso vamos a tener que confiar en el resultado de métricas, no siempre podemos VER el resultado

Una pequeña digresión

En resumen

- No hay impedimentos en usar los algoritmos en puntos de cualquier dimensionalidad (solo restricciones computacionales)
- La mayoría de las veces no vamos a poder “ver” las cosas, vamos a tener que confiar en las métricas que usemos
- **NOTA**: Cuando hablamos de “el espacio de características” nos referimos el espacio donde se mueven nuestro datos

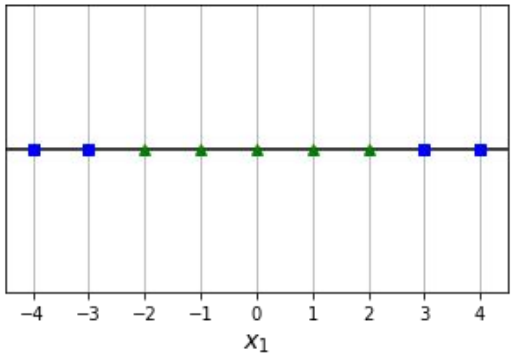
Una pequeña digresión

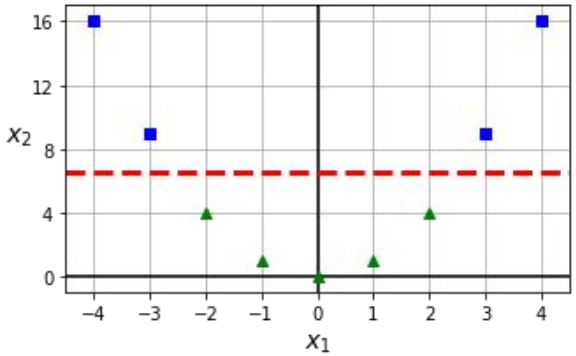
)

Clasificaciones no lineales

Los SVM pueden funcionar eficientemente para muchos casos, pero hay conjuntos de datos que no están ni cerca de ser linealmente separables

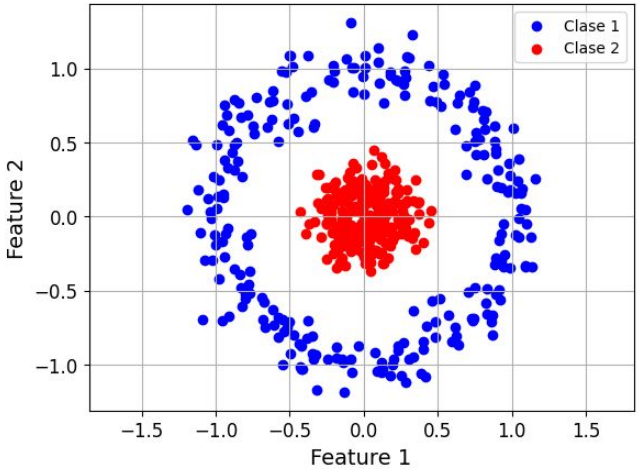
- Una forma de trabajar con esto, además de suavizar el margen, es agregar más características a los puntos (como hicieron con regresión polinomial) o sea, **aumentar la dimensionalidad de los datos**
- En algunos casos esto puede resultar en un conjunto de datos linealmente separable
- Se puede implementar definiendo un pipeline con un `PolynomialFeatures(...)`

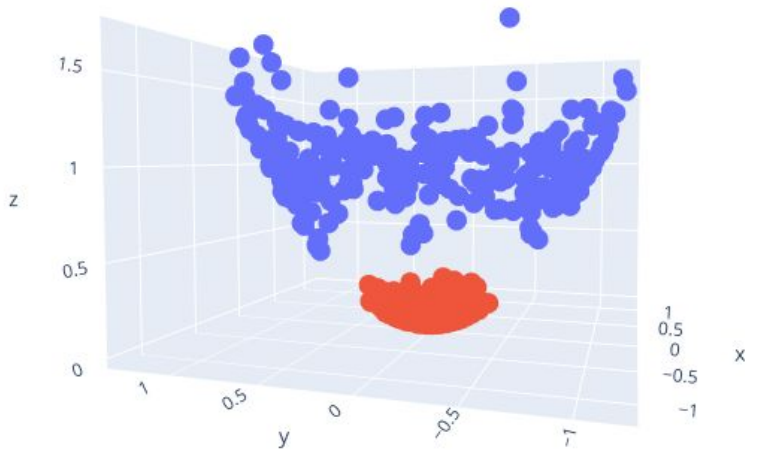




- Los puntos en el gráfico 1
  p_0 = -4
  P_1 = -3
  …
- Los puntos en el gráfico 2
  p_0 = (-4, -4^2)
  P_1 = (-3, -3^2)
  …

Clasificaciones no lineales





(x, y) → (x, y, x^2+y^2)

Estamos llevando nuestros datos a otro espacio porque ahí es más fácil clasificarlos. Mantengan eso en mente...

El poder de los kernels

- Agregar características polinomiales es fácil de implementar y puede funcionar con todo tipo de algoritmos (no solo con SVM)
- Pero con un grado bajo esta táctica no puede tratar con datos muy complejos e incrementarlo demasiado aumenta mucho la cantidad de características haciendo el modelo muy lento
- Afortunadamente, al trabajar con SVMs, se puede usar una técnica casi milagrosa[1], llamada el *truco del kernel,* con la que se obtiene el mismo resultado que agregar muchas características polinomiales, incluso potencialmente infinitas, sin tener que agregarlas (!!!)
- Los kernels permiten operar en un espacio de características implícito y de alta dimensión sin tener que calcular nunca las coordenadas de los datos en ese espacio, sino simplemente *haciendo las cuentas* en el espacio de características. Esta operación es a menudo más barata que el cálculo explícito de las coordenadas.
- O sea, **no hay explosión combinatoria en el número de características**

[1] Géron's dixit

El poder de los kernels

Hay varios tipos de kernels disponibles para usar (incluso se pueden definir) cada uno tiene sus hiperparámetros particulares que vamos a ver en el notebook:

- Lineal
- Polinomial
- Kernel Gaussiano RBF
- Hay otros tipos más específicos dependiendo el tipo de problema e.g. String Kernels para texto

Notebook

Vamos al notebook!

`Notebook_Semana_3_SVM.ipynb`