Projecto Final: Máquinas de Soporte Vectorial

Facultad de Ciencias UNAM
Física Computacional — 14/12/2023

Antonio Moreno Jonni, López Villegas Fernando Maximiliano, Rubio Ruiz Claudia Daniela

***Abstract*—En este trabajo abordamos la teoría y aplicación de las Máquinas de Soporte Vectorial (MSV) en la clasificación de la estabilidad de una red eléctrica. Se discuten conceptos como la clasificación binaria por medio de un hiperplano, la unicidad del hiper plano de margen maximal dependiente de los vectores de soporte y, con ello, se desarrolla el Clasificador de Margen Maximal. Se introduce el concepto de vectores de soporte y de margen suave para lidiar con los casos en que el clasificador anteriormente mencionado es muy sensible a nuevas observaciones, construyendo el Clasificador de Soporte Vectorial. Una vez que abandonamos el territorio lineal y ampliamos el espacio de características, tenemos como resultado un alto costo computacional. Finalmente, para añadir la no-linealidad de manera eficiente, se construyen las Máquinas de Soporte Vectorial con el concepto de Kernel.**

Se aplicó la MSV para clasificar la estabilidad de una red eléctrica. Se llegó a la conclusión de que las MSV son efectivas en la clasificación de problemas complejos y se destaca la importancia de ajustar cuidadosamente los hiperparámetros mediante la búsqueda exhaustiva para mejorar el rendimiento del modelo, aunque las MSV presentan casi una nula necesidad.

I. Introducción

El objetivo de este ensayo es presentar la teoría pertinente a las Máquinas de Soporte Vectorial, los conceptos básicos requeridos para comprenderla, y emplearla para elaborar un programa que resuelva un problema de aplicación en física en el cual se requiera clasificar en distintas categorías un conjunto de datos, como lo es la estabilidad de red eléctrica, el cual consiste en determinar si la red eléctrica es estable o inestable en base a parámetros establecidos.

II. Clasificadores de Margen Maximal

El principal objetivo del Clasificador de Margen Maximal es obtener el hiperplano que separe al máximo las diferentes clases en un conjunto de datos.

Un hiperplano es un espacio afín de dimensión n-1 que divide el espacio en dos, correspondientes a las entradas de las dos clases distintas. Su definición matemática en dos dimensiones es:

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 = 0 \tag{1}$$

Con $\beta_0, \beta_1, \beta_2$ parámetros. Nótese que tiene la misma forma que la ecuación de la recta, más aún, en 2D el hiperplano es una línea. En un espacio p-dimensional, el hiperplano está dado por:

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p = 0 \tag{2}$$

Si $\vec{X} = (x_1, x_2, ..., x_p)^T$ un vector p-dimensional satisface (2), decimos que es un punto del hiperplano, de lo contrario, tendríamos alguno de los siguientes casos:

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p > 0 \tag{3}$$

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p < 0 \tag{4}$$

y esto nos indicaría a qué lado del hiperplano se encuentra.

El margen es la distancia perpendicular entre el hiperplano y el vector de soporte más cercano.

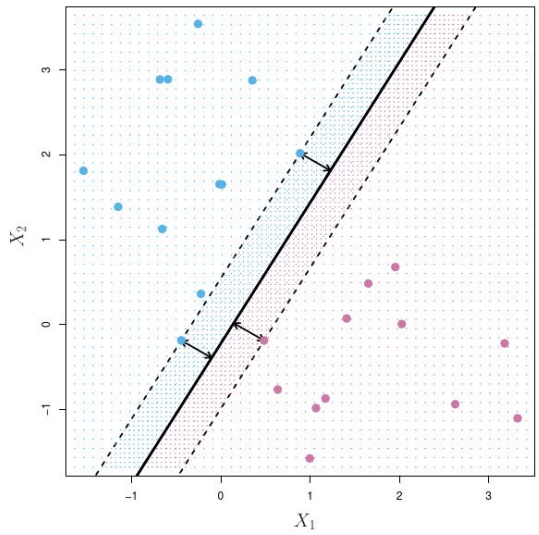

Fig. 1. El hiperplano de margen maximal está indicado con una línea sólida. El margen es la distancia del hiperplano a cualquiera de las líneas punteadas. Los puntos que yacen sobre esta línea son los vectores de soporte. *Nota*. Adaptado de *Support Vector Machines* (p.372), por G.James et.al., 2021, Springer Texts in Statistics.

En general, si los datos pueden ser separados perfectamente usando un hiperplano, entonces existe un número infinito de tales hiperplanos. Una decisión natural, es elegir el **hiperplano de margen maximal** (*maximal margin hyperplane*), es decir, el hiperplano de separación para el cual el margen es el mayor.

Podemos entonces clasificar nuestros datos dependiendo de en qué lado del hiperplano de margen maximal se encuentren, esto se denomina entonces *Clasificador de Margen Maximal.*

En la **Figura 1** podemos notar que tres observaciones de entrenamiento son equidistantes al hiperplano, las líneas punteadas indican la anchura del margen. Estos vectores p-dimensionales son conocidos como *vectores de soporte*, ya que "soportan" al hiperplano de margen maximal en el sentido de que si éstos se movieran, el hiperplano lo haría también, es decir, depende directamente de ellos.

A. Construcción del Clasificador de Margen Maximal

Partiendo de un conjunto de n observaciones de prueba, $x_1, ...x_n \in R^p$, con etiquetas de clase asociadas $y_1, ...y_n \in \{-1, 1\}^p$, entonces el hiperplano de margen maximal es la solución al problema de optimización siguiente: Maximizar M, con:

$$\beta_0, \beta_1, ..., \beta_p, M \tag{5}$$

sujeta a:

$$\sum_{j=1}^{p} \beta_j^2 = 1 \tag{6}$$

$$y_i(\beta_0 + \beta_1 xi_1 + \beta_2 x_{i2} + ... + \beta_p xi_p) \geq M \forall i = 1, ..., n \tag{7}$$

La ecuación (7) garantiza que si M es positivo, cada observación estará en el lado correcto del hiperplano. Observemos que (6) no constituye realmente una limitación, ya que si $\beta_0+\beta_1 xi_1+\beta_2 x_{i2}+...+\beta_p xi_p = 0$, define un hiperplano, entonces la multiplicación de esta ecuación por cualquier k constante diferente de cero, también lo hace. Esta condición, en conjunto con (7) garantiza que la distancia perpendicular desde la i-observación al hiperplano está dada por:

$$y_i(\beta_0 + \beta_1 xi_1 + \beta_2 x_{i2} + ... + \beta_p xi_p) \tag{8}$$

Por lo que se asegura que cada observación en el lado correcto del hiperplano se encuentra a una distancia a lo menos M de él. M representa el margen y los coeficientes $\beta_0, \beta_1, ...\beta_p$, son los parámetros de maximización dependientes del problema. En general, es deseable un margen mayor, puesto que indica una separación más sólida entre clases.

B. *El caso no-separable*

En casos en que el hiperplano de separación no existe, el problema de optimización establecido por (5) y (6) carece de solución para M positivo, es decir, no podemos separar *exactamente* las dos clases.

III. Clasificadores de Soporte Vectorial

En casos donde el hiperplano de margen maximal conduce a sobreajustes, debemos considerar otro tipo de clasificador.

El clasificador de vectores de soporte, aunque no separa *perfectamente* las dos clases, sí ofrece mayor robustes ante variaciones individuales y clasifica mejor la mayoría de las observaciones de entrenamiento. El **clasificador de soporte vectorial** es a veces denominado *clasificador de margen suave*, ya que permite algunas observaciones en el lado incorrecto del margen.

A. *Especificaciones matemáticas*

El clasificador de soporte vectorial, selecciona una observación de prueba en función de que lado de un hiperplano se ubique. El hiperplano separa la mayoría de las observaciones de *entrenamiento* en dos clases, aunque puede separar erróneamente algunas de ellas, es la solución al problema de optimización *(7)*, sujeto a la ecuación *(6)* con

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots \beta_p x_{ip}) \geq M(1 - \epsilon_i) \tag{9}$$

$$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C \tag{10}$$

donde C es un parámetro de ajuste no negativo. En *(9)* $\epsilon_1 \dots \epsilon_n$ son llamadas *"variables de holgura"* que permiten que las observaciones individuales estén en el revés del margen o del hiperplano, una vez resueltas *(5)-(10)*, procedemos a clasificar una observación de prueba como x^* simplemente observando en que parte del hiperplano se encuentra, es decir, clasificamos la observación de prueba con una función $f(x^*) = \beta_0 + \beta_1 x_1^* + ... + \beta_p x_p^*$. Todo esto parece complejo pero se puede entender perfectamente haciendo unas cuantas observaciones al problema de optimización, pero principalmente enfoquémonos en *(10)*.

B. *Análisis del parámetro C*

De la expresión *(10)* tenemos que si $\epsilon_i = 0$ entonces la í-esima observación se encuentra en el lado correcto del margen, si $\epsilon_i > 0$ se encuentra en lado incorrecto, mientras que si $\epsilon_i > 1$ ésta se encuentra en lado equivocado del hiperplano.

Ahora considerando el parámetro C, el cual lo podemos pensar como un tipo de presupuesto que se tiene para la cantidad que se puede violar el margen por las n observaciones. ¿Pero que pasa si $C = 0$? Resulta que no hay presupuesto para violaciones al margen, entonces debe darse $\epsilon_1 = \dots = \epsilon_n = 0$ lo cual es equivalente al problema de optimización visto en la sección anterior. Ya aclaramos que podemos ajustar C, lo interesante viene cuando al parámetro le asignamos valores como $c > 0$, para este caso c observaciones pueden estar en el lado equivocado del hiperplano, porque si esto sucede entonces $\epsilon_i > 1$ y por la expresión *(10)* queremos que C sea lo suficientemente grande, y esto es genial porque a medida que el parámetro aumente nos volveremos más tolerantes a la violación del margen y al margen se ampliará, ocurre lo contrario cuando C disminuye, el margen se estrecha.
Al analizar al parámetro C y ver que se relaciona con las violaciones al margen, es natural hacer la siguiente pregunta ¿Hay algo que nos brinde información acerca de las violaciones permitidas? De la expresión *(10)*, observamos que acota la suma de los ϵ_i y así determina el número y que tan graves son las violaciones al margen y al hiperplano que podemos tolerar.

Podemos decir que ajustar C permite que el modelo equilibre el deseo de un margen más grande con la tolerancia a los errores de clasificación.

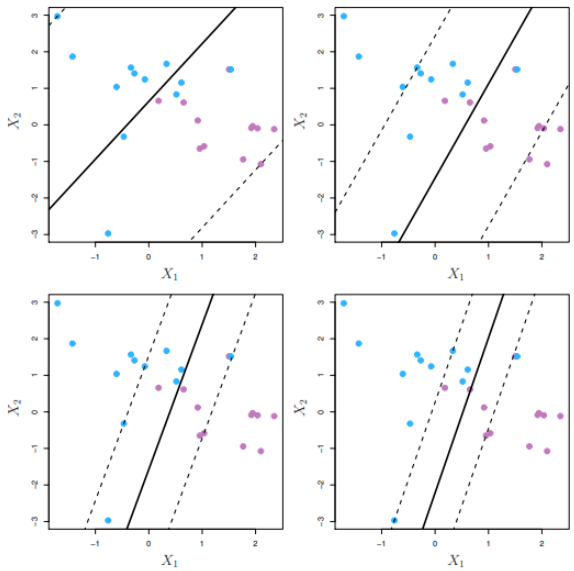

Fig. 2. Se muestra la varianza de C comenzando por un valor muy grande y haciendo pequeño. *Nota.* Adaptado de *Support Vector Machines* (p.378), por G.James et.al., 2021, Springer Texts in Statistics..

En la **Figura 2**, se observan 4 ajustes del parámetro C, comenzando por un valor grande el cual se va haciendo cada vez más pequeño. Cuando C es grande hay una mayor tolerancia para que las observaciones estén en el lado incorrecto del margen, a medida que C disminuye la tolerancia también lo hace.

C. *Empleo del parámetro C en un clasificador de Soporte Vectorial*

Generalmente, el parámetro C se elige a través de validación cruzada y lo más importante es que controla el equilibrio sesgo-varianza de la técnica de aprendizaje estadístico. Un clasificador

que se ajusta en gran medida a los datos se obtiene cuando C es pequeño lo que conlleva a un sesgo bajo pero a una varianza alta, ocurre lo contrario cuando C es muy grande, tenemos un clasificador que es potencialmente más sesgado pero su varianza es más baja. Una propiedad interesante del problema de optimización es la siguiente: una observación que se encuentre estrictamente en el lado correcto del margen no afectará al clasificador de vectores de soporte, incluso si se cambia la posición de observación, siempre y cuando se respete que su posición se mantenga en el lado correcto del margen. Las observaciones que se encuentran en el margen o en el lado equivocado del margen, se les conoce como vectores de soporte y afectan al clasificador de vectores de soporte. Este tipo de clasificadores son menos sensible a la influencia de puntos de datos atípicos, aunque si llega a haber observaciones lejanas al hiperplano el **CSV** se centra en un pequeño subconjunto llamado *vectores de soporte* para definir la decisión de clasificación, lo cual es muy bueno ya que permite generalizar mejor a datos nuevos y desconocidos por que no se ve influenciado por observaciones que estén muy lejos del hiperplano, contrario pasa con la regresión logística, de hecho la regresión logística y el clasificador de vectores de soporte están estrechamente relacionados.

IV. Maquinas de Soporte Vectorial

A. Aumento

Ha quedado claro que el **Clasificador de Soporte Vectorial (CSV)** es muy efectivo a la hora de separar datos cuya frontera resulte ser lineal, desgraciadamente sabemos que esto no es muy común, casi en ninguna situación, consideremos el siguiente conjunto de datos (**Figura 3**):

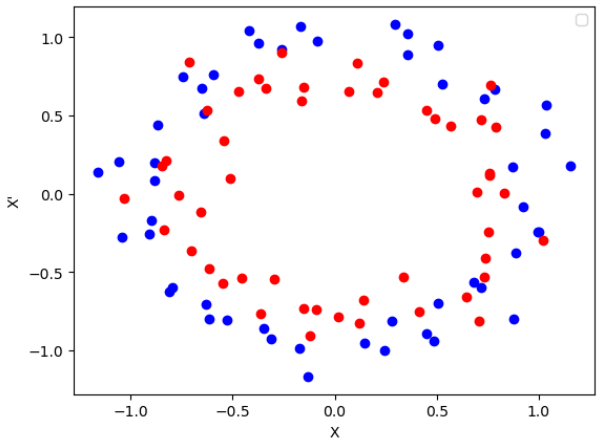

Fig. 3. Conjunto de datos no linealmente separables.

Si nosotros intentáramos usar una linea para separar los datos en sus respectivas clases *(Rojo y Azul)* vemos que tendríamos muchos errores de clasificación (**Figura 4**).

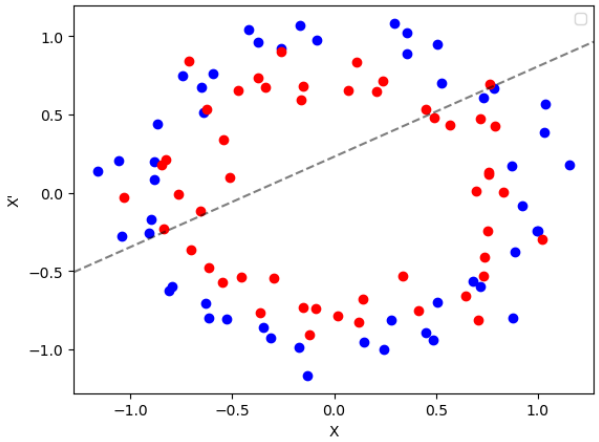

Fig. 4. Conjunto de datos no linealmente separables 1.

Vamos a empezar con un conjunto menos agresivo (**Figura 5**), consideremos las observaciones azules y las observaciones rojas, es obvio que aquí no podemos usar una linea. Pero podemos atacar este problema de la siguiente manera, tomemos a X y elevémosla al cuadrado, grafiquemos los puntos (X, X^2) y obtendremos la gráfica de la **Figura 6**.

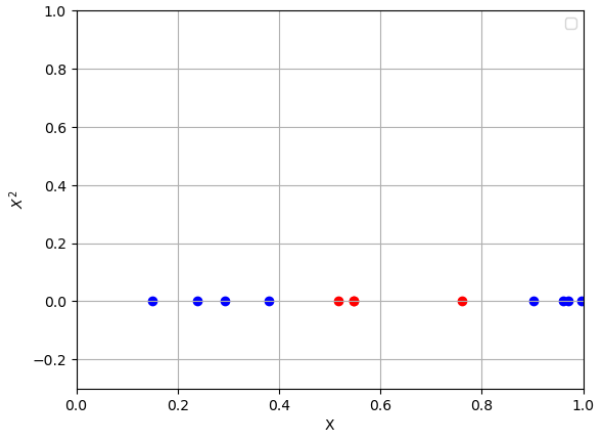

Fig. 5. Conjunto de datos no linealmente separables 2.

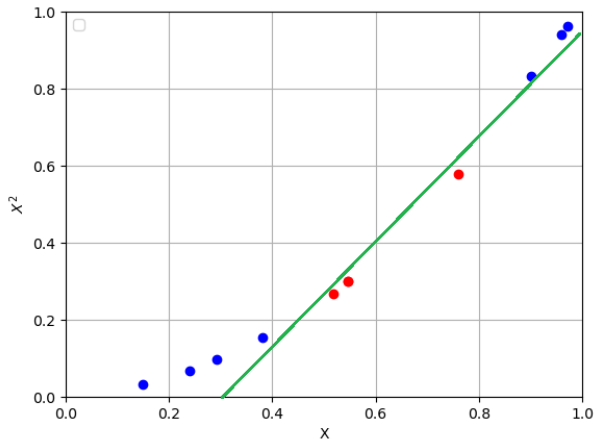

Fig. 6. Conjunto de datos 2 en el espacio aumentado de características..

Podemos ver que ahora sí podemos separar los datos y con una linea...,¡Woah!, ¿qué fue eso?, lo que hicimos fue aumentar el espacio de características.

Podemos aumentar el espacio de características usando funciones cuadradas, cúbicas e incluso de órdenes mayores de los predictores (X) (pueden ser funciones más extravagantes, como módulos o trigonométricas).

En lugar de hacer el ajuste de un **CSV** con p características, podríamos usar $2p$ características, entonces el problema de optimización que se vio para el **CSV** seria:

$$\underset{\beta_0;\beta_{11};\beta_{12};...;\beta_{\mathrm{p1}};\beta_{\mathrm{p2}};\epsilon_1;...\epsilon_{\mathrm{n}};\mathrm{M}}{\text{maximizar}} M \tag{11}$$

con la condición de que:

$$y_i \left(\beta_0 + \sum_{j=1}^{p} \beta_{ji} x_{ij} + \sum_{j=1}^{p} \beta_{j2} x_{ij}^2 \right) \geq M(1 - \epsilon_i) \tag{12}$$

$$\sum_{i=1}^{n} \epsilon_i \leq C; \qquad \epsilon_i \geq 0; \qquad \sum_{j=1}^{p} \sum_{k=1}^{2} \beta_{jk}^2 = 1 \tag{13}$$

En el espacio de características ampliado, la frontera de decisión que resulta de lo anterior de hecho sí es lineal. Pero en el espacio de características original, la frontera de decisión es de la forma $q(x) = 0$, donde q es un polinomio cuadrático y sus soluciones generalmente no son lineales.

B. Eficiencia

El aumentar el espacio de características parece nuestra salvación, pero no es así, nótese que al hacer esto estamos aumentando también en gran medida la cantidad de características que la computadora va a procesar

Resulta que la solución al problema de optimización del **CSV** involucra solamente a los productos internos de las observaciones.

El producto interno de dos observaciones $x_i, x_{i'}$ está dado por:

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j} \tag{14}$$

El **CSV** lineal puede ser representado como:

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x_i, x_{i'} \rangle \tag{15}$$

donde hay n parámetros α_i, con $i = \{1, ..., n\}$. Para estimar los parámetros $\alpha_1, ..., \alpha_n$ y β_0 necesitamos $\frac{n(n-1)}{2}$ productos internos $\langle x_i, x_{i'} \rangle$ sobre todos los pares de observaciones de entrenamiento.

Resulta que α_i es diferente de cero solo para los vectores de soporte en la solución.

Si consideramos a S como el conjunto de los índices de soporte, entonces:

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i \langle x_i, x_{i'} \rangle \tag{16}$$

C. Kernel

Cada vez que el producto interno aparezca en la representación, o en la solución para el **CSV**, lo remplazaremos con una generalización del producto interno de la forma $K(x_i, x_{i'})$, donde K es alguna función a la que le llamaremos **Kernel**. Un kernel es una función que cuantifica la similaridad de dos observaciones. Por ejemplo:

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'} x_j \tag{17}$$

lo que nos devolvería el clasificador de soporte vectorial. Lo anterior es conocido como un Kernel lineal porque el **CSV** es lineal en sus características.

De hecho podríamos poner cualquier otra cosa, por ejemplo:

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'} x_j)^d \tag{18}$$

Esto es conocido como el **Kernel Polinomial de grado d** con d en los enteros positivos.

Si usamos un Kernel con $d > 1$ en lugar del Kernel Lineal, entonces el algoritmo de CSV nos otorga a una frontera de decisión muchísimo más flexible.

Esencialmente, provoca que el ajuste del **CSV** sea un espacio de mayor dimensión que usa polinomios de grado d, en lugar del espacio de características original (como ya vimos al principio de esta sección), pero aquí ya tenemos la ventaja de no tener que transformar los puntos para poder calcular su relación en su espacio aumentado, resolviendo el problema de la eficiencia.

Cuando el **CSV** se fusiona con un Kernel no lineal (hay que enfatizar, no sólo es el hecho de que sea no lineal sino de que es óptimo para los cálculos), al clasificador resultante se le conoce como **Maquina de Soporte Vectorial**.

Otro Kernel no lineal es el Kernel Radial dado por:

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2) \tag{19}$$

Con γ constante positiva, notemos que su comportamiento (clasificación) se ve influenciado en su mayoría por sus **vecinos más cercanos** y no por aquellos que están lejos.

V. Implementación de las Maquinas de Soporte Vectorial en Python con aplicación relacionada a la Física.

Para esto necesitamos un problema en física o al menos relacionado que se le haya asociado una base de datos. Encontramos la siguiente base de datos: Datos simulados de estabilidad de la red eléctrica, donde el objetivo es determinar si la red eléctrica es estable o no en base a los siguientes parámetros:

- $\tau[x]$: Tiempo de reacción de los participantes en un rango de 0.5 a 10 segundos.
- $p[x]$: Poder nominal consumido o producido (negativo o positivo respectivamente).
- $g[x]$: Coeficiente proporcional a la elasticidad del precio en un rango de 0.05 a 1 s^{-1}
- $stab$: La parte real máxima de la raíz de la ecuación característica, si es positiva el sistema es linealmente inestable.

Comenzamos limpiando y asegurándonos de que la base de datos esté en condiciones de ser usada por una **MSV**. Una vez que terminamos con eso, se reduce el número de observaciones que se va a usar aplicando *downsampling*, para balancear la contribución de los datos de cada clase y mejorar el rendimiento de la **MSV**. Dividimos en dos conjuntos, de entrenamiento y de prueba, se ajustan y posteriormente se le proporcionan a la **MVS**.

```
clf_svm = SVC(random_state=42) #creamos el
    clasificador
clf_svm.fit(X_train_scaled, y_train) #entrenamos
    el clasificador
```

Haciendo uso de una matriz de confusión podemos ver que la **MSV** es no solo muy eficiente si no que no necesita de muchos hiperparametros (**Figura 7**).

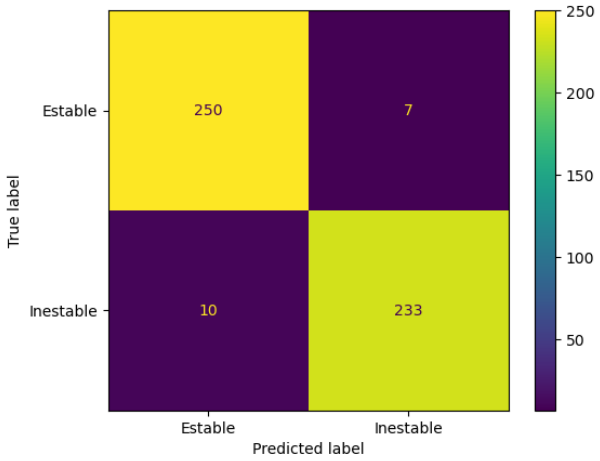

Fig. 7. Matriz de Confusión sin parámetros optimizados.

Y la pregunta obvia es ¿qué tanto mejora si ajustamos esos parámetros? Recordemos el problema de optimización en el **CSV** donde teníamos el parámetro C que nos indicaba que tanto se puede violar el margen, y ahora con la introducción del Kernel tenemos un nuevo parámetro γ que no especificamos que indica en realidad pero tiene que ver con la forma de la frontera final.

¿Cómo logramos esto?,lo que hace es lo siguiente, imaginemos que queremos cocinar el mejor sandwich con cierto tipo de ingredientes **param_grid** es solo una lista de que posibles combinaciones de ingredientes se pueden usar, siendo estos:

- **C**: que tanto se puede violar el margen,
- γ: que tan alocada se pone la frontera,
- **Kernel**: que viene a ser como el pan que usaremos.
- **optimal_params**: es un señor que nos encontramos en la calle y obligaremos a probar todas esas posibles combinaciones de sándwiches.
- **SVC**: es el cocinero,
- **cv**: vamos a obligar al señor a probar 5 veces el mismo sandwich para estar seguros de que si sabe bien.
- **scoring**: tiene estándares el señor, es como la forma en la que va a juzgar que tan bien o no están.
- **optimal_params.fits** aquí es cuando se está llevando a cabo la degustación de sándwiches.

```
param_grid = [
  {'C': [0.5, 1, 10, 100],
   'gamma': ['scale', 1, 0.1, 0.01, 0.001,
       0.0001],
   'kernel': ['rbf']},
]
optimal_params = GridSearchCV(
      SVC(),
      param_grid,
      cv=5,
      scoring='accuracy',
      verbose=2
   )
optimal_params.fit(X_train_scaled, y_train)
print(optimal_params.best_params_)
```

Una vez acabando esa degustación, encontramos que el señor decidió que esta es la mejor combinación de sándwich:

```
{'C': 100, 'gamma': 0.001, 'kernel': 'rbf'}
```

Insertamos esos datos en la **MSV**

```
clf_svm = SVC(random_state=42, C=100, gamma=0.001)
clf_svm.fit(X_train_scaled, y_train)
```

Podemos ver que no cambió mucho (**Figura 8**), la MSV le dió casi al clavo desde el principio.

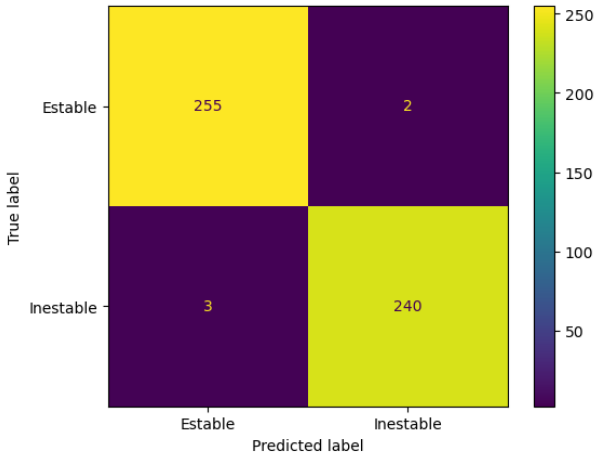

Fig. 8. Matriz de Confusión con parámetros optimizados..

El Analisis de Componentes Principales nos permite visualizar está compleja base de datos (por sus dimensiones), en la **Figura 9**:

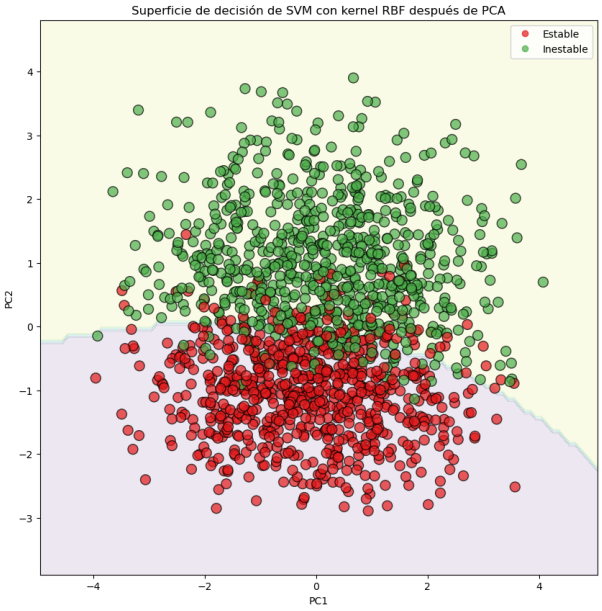

Fig. 9. Uso de PCA para visualizar la superficie de decisión..

VI. Conclusiones

Una vez presentada la teoría acerca de las *Maquinas de Soporte Vectorial* se aplicó dicha teoría a la resolución de un problema físico, para ello se recurrió a una base de datos simulados de la estabilidad de una red eléctrica. El propósito fundamental fue determinar si la red es estable o no con base a parámetros establecidos.

Pudimos observar mediante una matriz de confusión que las **(MSV)** son demasiado eficientes necesitando de pocos hiperparámetros. Ajustando el parámetro C y con la inclusión de un kernel a través del GridSearch obtuvimos los parámetros óptimos para nuestro

modelo. Gracias a la implementación de una **(MSV)** optimizada se pudieron determinar los escenarios posibles en los que la red eléctrica era estable o inestable con gran precisión.

Appendix A
MSV con distintos Kernel

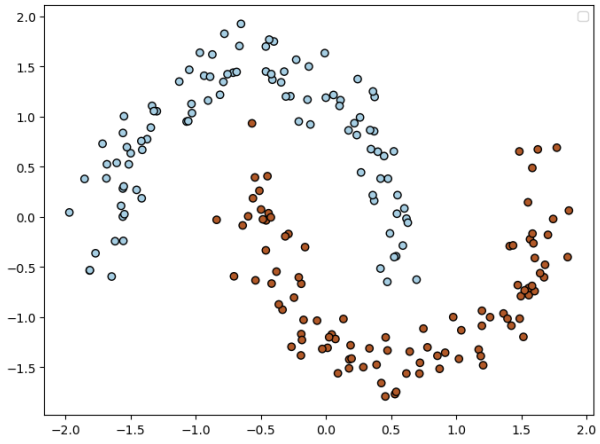

Fig. 10. Conjunto de datos no linealmente separables 3 .

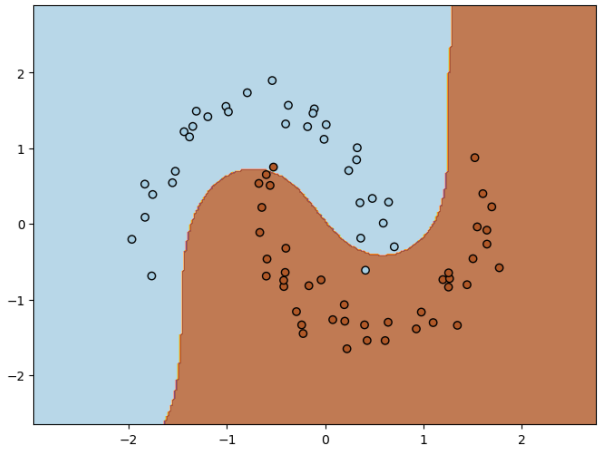

Fig. 11. MSV con Kernel RBF.

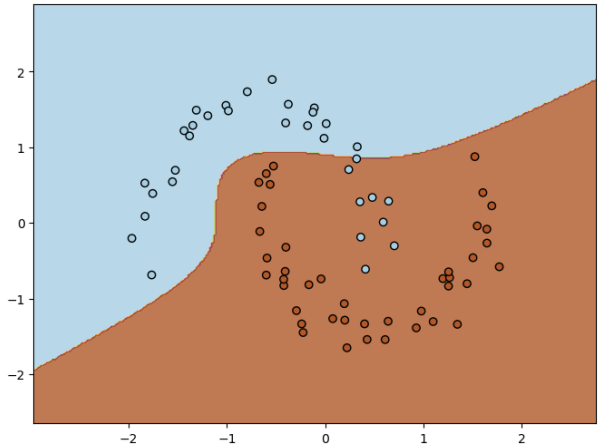

Fig. 12. MSV con Kernel Polinomico.

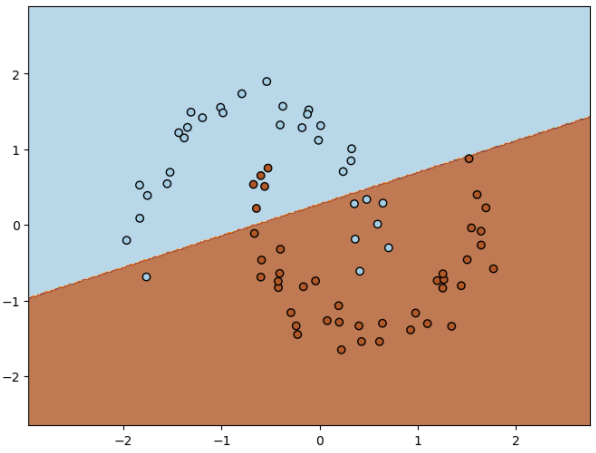

Fig. 13. MSV con Kernel Lineal.

Appendix B
Código de preparación de datos.

```
df =
    pd.read_csv('ELECTRODATA\Data_for_UCI_named.csv',
            header=None)
df.columns =
    ["tau1","tau2","tau3","tau4","p1","p2","p3",
"p4",
"g1","g2","g3","g4","stab","Estabilidad"]
dflimpio = df.drop(index=0).copy()
dflimpio.rename({'stabf' : 'Estabilidad'},
    axis='columns', inplace=True)
dflimpio.Estabilidad =
    dflimpio.Estabilidad.eq('unstable').mul(1)
X = dflimpio.drop('Estabilidad', axis=1).copy()
y = dflimpio['Estabilidad'].copy()
df_estable = dflimpio[dflimpio['Estabilidad'] ==
    0]
df_inestable = dflimpio[dflimpio['Estabilidad']
    == 1]
df_estable_downsampled = resample(df_estable,
                          replace=False,
                          n_samples=1000,
                          random_state=42)
df_inestable_downsampled = resample(df_inestable,
                          replace=False,
                          n_samples=1000,
                          random_state=42)
df_downsample =
    pd.concat([df_estable_downsampled,
    df_inestable_downsampled])
X = df_downsample.drop('Estabilidad',
    axis=1).copy()
y = df_downsample['Estabilidad'].copy()
X_train, X_test, y_train, y_test =
    train_test_split(X,y, random_state=42)
scaler =
    preprocessing.StandardScaler().fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

References

[1] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An Introduction to Statistical Learning: with Applications in R* (Springer Texts in Statistics). Springer Science+Business Media. DOI: 10.1007/978-1-0716-1418-1

[2] Cristianini, N., & Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press. DOI: 10.1017/CBO9780511801389

[3] Kecman, V. (Autor), & Wang, L. (Ed.). (2005). *Support Vector Machines: Theory and Applications* (Studies in Fuzziness and Soft Computing, 177). Springer-Verlag Berlin Heidelberg. DOI: 10.1007/10984697