UNIVERSIDAD DE BUENOS AIRES
Facultad de Ciencias Exactas y Naturales
Departamento de Matemática

Tesis de Licenciatura

Análisis del rendimiento de un clasificador en datos de alta dimensión y múltiples clases

Carlos Manuel Maldonado

Directora: Daniela Rodriguez

Septiembre 2025

Agradecimientos

A Daniela, mi directora de tesis, por acompañarme en el proceso. Su paciencia y dedicación fueron claves para llegar hasta el final.

A mi familia, por su apoyo incondicional. Han estado para darme energía cuando más lo necesitaba. Me han bancado económicamente en este caótico país.

A todas mis amistades de la facultad. Ha sido una experiencia inolvidable. Hemos compartido todo, desde una pasión por la matemática, horas de ping pong, momentos de mucha alegría, largas horas de estudio y momentos de sufrimiento.

Ha habido esos momentos de quiebre, en los cuales no siempre creí que concluiría el proceso. Gracias a todas esas personas que sin saberlo hicieron de mi día a día muy llevadero y hermoso. Desde mis compañeros de teatro, que han sido mi cable a tierra, a todas amistades que me escucharon y aconsejaron. Varias de ellas inclusive sin estar en el país han sido parte imprescindible del proceso.

A la educación pública y en particular la universidad pública, que a pesar de todo siguen dando ejemplo de calidad y resiliencia.

Índice general

Introducción

En esta tesis se aborda el problema de la clasificación multiclase en contextos de alta dimensión, caracterizados por un gran número de variables y clases, y un número reducido de observaciones por clase. Este escenario se ha vuelto cada vez más común en los últimos años, impulsado por la capacidad creciente de almacenar y procesar grandes volúmenes de datos, como ocurre en áreas como la bioinformática o el procesamiento de imágenes. Si bien este problema ha sido explorado desde una perspectiva computacional, aún persisten importantes desafíos teóricos y prácticos.

Nos enfocamos específicamente en el trabajo de Abramovich y Pensky (2019), quienes proponen un marco teórico sólido para estudiar la exactitud de la clasificación en este tipo de entornos. A partir de un modelo normal multivariado, los autores derivan condiciones que permiten analizar la selección de variables y la exactitud del clasificador. Un hallazgo particularmente interesante de su análisis es que, bajo ciertas condiciones, un gran número de clases puede mejorar la exactitud de la clasificación, lo que resulta contraintuitivo.

En esta tesis se presenta una exposición de los resultados teóricos del artículo y se replican los estudios de simulación propuestos por los autores, comparando su rendimiento con otros algoritmos. Este trabajo contribuye a una mejor comprensión del comportamiento de los clasificadores en escenarios complejos y ofrece herramientas útiles para su aplicación en problemas reales de alta dimensión.

Capítulo 1

Introducción al problema de clasificación

El problema de clasificación es parte de la Ciencia de Datos. El objetivo es intentar predecir variables cualitativas de nuevas observaciones, es decir, elegir a que categoría pertenecen de todas las posibles. Es un problema supervisado, es decir, tendremos datos de entrenamiento para los cuales sabremos cómo clasificarlos.

Existen diversos tipos de clasificadores. Entre ellos, destacan dos grupos de algoritmos. Los primeros están basados en estimar las probabilidades de pertenecer a cada grupo y asignarlos a la categoría más probable. Los segundos, en intuición geométrica. Presentaremos a continuación un algoritmo perteneciente a cada grupo, Linear Discriminant Analysis (LDA) y Support Vector Machines (SVM). Los elegimos debido a que, por su sencillez, se pueden ejecutar en altas dimensiones y con pocos datos. Más adelante en la tesis, los usaremos para competir frente al algoritmo que se describirá en el siguiente capitulo.

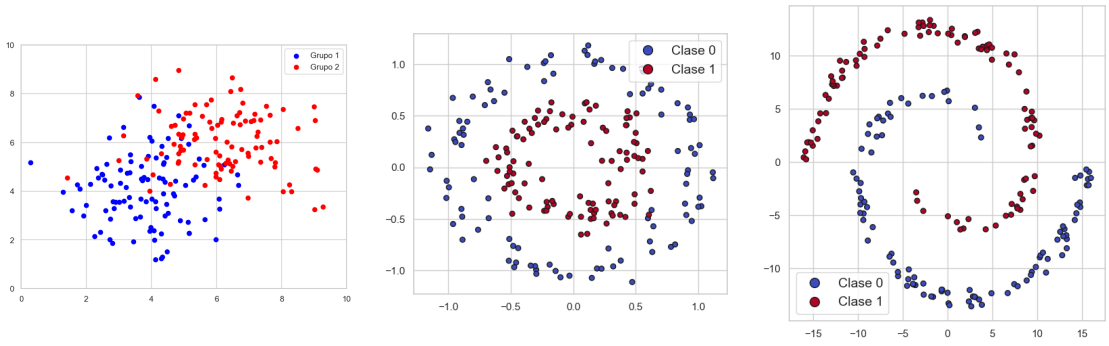
Primero, describiremos su funcionamiento interno para dos categorías. Luego, abordaremos las situaciones en las que haya muchas clases. Para finalizar, explicaremos cómo elegir la mejor receta para cada uno mediante validación cruzada. Esta técnica nos resultará muy útil en los próximos capítulos, cuando la cantidad de datos sea escasa. Este capítulo está basado en el libro [4].

1.1. Clasificación con 2 categorías

Abordaremos el problema de clasificación más sencillo en el que hay solamente dos grupos posibles. Posee diversas aplicaciones, entre las que están: decidir si se le da un préstamo personal a una persona, detectar casos de fraude bancarios, decidir si un mail es o no spam, entre otras.

El problema a resolver será poder clasificar nuevas observaciones $x^* \in \mathbb{R}^p$ en su categoría correcta (podrá ser $Y = 0$ ó $Y = 1$). Para ello, haremos uso de nuestro

conjunto de datos de entrenamiento, $(x_i, y_i) \ : \ i \in \{1, \ldots, n\}$ para generar la regla de clasificación. A la cantidad de datos de cada clase, respectivamente, la llamaremos n_0 y n_1. Estos mismos pueden tener diversas formas:



A continuación, vamos a focalizarnos en cómo funcionan los algoritmos Linear Discriminant Analysis (LDA) y Support Vector Machine (SVM). El primero servirá para casos más sencillos, mientras que el segundo, puede adaptarse mejor a más escenarios.

Para simplificar las ideas, pensaremos que en caso que haya parámetros que puedan obtener distintos valores y los algoritmos funcionen igual, están fijos. Más adelante en este capítulo vemos como obtener su mejor combinación para los datos a través de validación cruzada.

Los ejemplos presentes en capítulo serán datos sintéticos, generados para ilustrar algún comportamiento específico.

1.1.1. Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) pertenece al grupo de algoritmos de clasificación que se basan en estimar probabilidades. En este caso, se calcula $P(Y = l|X = x) \ \forall \ l \in \{0, 1\}$. Dada una nueva observación x^*, se le asignará el grupo l^* para el cual dicha probabilidad sea mayor ($P(Y = l^*|X = x^*) \geq P(Y = l|X = x^*) \ \forall \ l \in \{0, 1\}$). Es decir, a partir de lo se conoce como la probabilidad de posteriori se genera la regla de clasificación. Es la bien conocida como Regla de Clasificación óptima de Bayes.

Una manera natural de enfrenar el problema es calcular $P(Y = l|X = x^*) \ \forall \ l \in \{0, 1\}$ a partir del Teorema de Bayes, para ello y a modo ilustrativo asumiremos que las covariables X son discretas. Por lo tanto,

$$P(Y = l|X = x^*) = \frac{P(Y = l) \ P(X = x^*|Y = l)}{\sum\limits_{l' \in \{0,1\}} P(Y = l') \ P(X = x^*|Y = l')} \quad \forall \ l \in \{0, 1\} \tag{1}$$

Esta reescritura de la probabilidad a posteriori nos induce un modo de estimar y así construir el clasificador en la práctica. De este modo será necesario proponer estimadores para:

- $\pi_l = P(Y = l)$ la probabilidad *a priori* de que una observación pertenezca a la categoría l. Que puede ser fácilmente estimada por la proporción de datos de entrenamiento de dicha categoría, es decir,

$$\hat{\pi}_l = \frac{1}{n} \sum_{i=1}^{n} 1_{Y_i = l} \quad \forall\, l \in \{0, 1\} \tag{2}$$

- $P(X = x^* | Y = l) \quad \forall\, l \in \{0, 1\}$ la probabilidad de haber observado a x^* dado que pertenecía a la categoría l. Como tenemos datos que sabemos que pertenecen a esa categoría, luego es natural estimar esas probabilidades a partir de los datos de esa categoría.

Si suponemos que X condicional a cada clase tiene densidad condicional, entonces es posible obtener una expresión similiar a (1) pero en lugar de $P(X = x^* | Y = l)$ en términos de $f_l(X)$ la densidad de X condicional a la clase l.

Por lo tanto, el desafió será estimar $f_l(X)$. Para este último punto es donde aparecen la mayor diversidad de opciones y modelos. En LDA se asume que la distribución para $f_l(X)$, es decir la distribución condicional de X dada la clase, sigue una distribución normal y con covarianzas idénticas en todas las clases. Es decir, $f_l(X) \sim \mathcal{N}(\mu_l, \Sigma)$. Luego la estimación en este caso se reduce a estimar los parámetros.

A continuación veremos como queda la regla de clasificación óptima, descripta anteriormente bajo el supuesto de normalidad. Para llegar a la fórmula general, comencemos por escribirla en el caso que haya una única co-variable. Luego, la generalizamos para el caso en donde haya más de una.

Escenario con una única co-variable

Nuestro objetivo es hallar la regla de clasificación basada en atribuir la clase l^* que maximice (1) en el escenario que $p = 1$.

Como mencionamos previamente, el método asume que $f_l(X) \sim \mathcal{N}(\mu_l, \Sigma)$. Si $p = 1$, luego $f_l(x) \sim \mathcal{N}(\mu_l, \sigma^2)$ es decir

$$f_l(x^*) = \frac{1}{\sqrt{2\pi}\ \sigma} \exp^{-\frac{1}{2\sigma^2}(x^* - \mu_l)^2} \quad \forall l \in \{0, 1\} \tag{3}$$

Bajo el modelo normal, es conocido como estimar los parámetros desconocidos:

- $\mu_l \ \forall\ l \in \{0, 1\}$ la media poblacional de cada categoría y se estima usando la muestral:

$$\hat{\mu}_l = \frac{1}{n_l} \sum_{i : y_i = l} x_i \quad \forall\, l \in \{0, 1\} \tag{4}$$

- σ la varianza en común para todas las categorías y se estima así:

$$\hat{\sigma}^2 = \frac{1}{n - 2} \sum_{l=0}^{1} \sum_{i : y_i = l} (x_i - \hat{\mu}_l)^2 \tag{5}$$

El algoritmo atribuye una nueva observación x^* a la clase l^* que maximice (1). En este caso, tenemos densidades condicionales. Veamos cómo quedan las fórmulas al realizar el método plug-in, que consiste en simplemente reemplazar los valores por las estimaciones correspondientes.

$$P(Y = l|X = x^*) \approx \frac{\hat{\pi}_l \ \hat{f}_l(x^*)}{\sum\limits_{l' \in \{0,1\}} \hat{\pi}_{l'} \ \hat{f}_{l'}(x^*)}$$

$$P(Y = l|X = x^*) \approx \frac{\hat{\pi}_l \ \frac{1}{\sqrt{2\pi}\ \hat{\sigma}} \exp^{-\frac{1}{2\hat{\sigma}^2}(x^* - \hat{\mu}_l)^2}}{\sum\limits_{l' \in \{0,1\}} \hat{\pi}_{l'} \ \frac{1}{\sqrt{2\pi}\ \hat{\sigma}} \exp^{-\frac{1}{2\hat{\sigma}^2}(x^* - \hat{\mu}_{l'})^2}} \quad \forall\ l \in \{0,1\} \tag{6}$$

En consecuencia, la regla de clasificación es hallar l^* que maximice (6). Veamos a continuación, cómo podemos reescribir el criterio de decisión de manera más sencilla (sacando los valores que sean constantes para todas las clases):

Proposición 1.1.1. Dado un nuevo x^* que se desea clasificar a través de LDA entre 2 categorías con $p = 1$. La clase l^* que maximiza las aproximaciones de $P(Y = 0|X = x^*), P(Y = 1|X = x^*)$ dadas en 6 es equivalente a la que maximiza:

$$\delta_l(x^*) = x^* \ \frac{\hat{\mu}_l}{\hat{\sigma}^2} - \frac{\hat{\mu}_l{}^2}{2\hat{\sigma}^2} + \ln \hat{\pi}_l \qquad l \in \{0,1\} \tag{7}$$

Demostración. Queremos encontrar l^* tal que se maximice

$$\hat{f}_l(x^*) = \frac{\hat{\pi}_l \ \frac{1}{\sqrt{2\pi}\ \hat{\sigma}} \exp^{-\frac{1}{2\hat{\sigma}^2}(x^* - \hat{\mu}_l)^2}}{\sum\limits_{Y \in \{0,1\}} \hat{\pi_Y} \ \frac{1}{\sqrt{2\pi}\ \hat{\sigma}} \exp^{-\frac{1}{2\hat{\sigma}^2}(x^* - \hat{\mu}_Y)^2}} \quad \forall\ l \in \{0,1\}$$

.

Notemos que $\forall l \in \{0,1\}$, $\hat{f}_l(x^*)$ tienen el mismo denominador, por ende, podemos prescindir de él para obtener el máximo. Luego bastara analizar,

$$\hat{\pi}_l \ \frac{1}{\sqrt{2\pi}\ \hat{\sigma}} \exp^{-\frac{1}{2\hat{\sigma}^2}(x^* - \hat{\mu}_l)^2}.$$

Notemos que ahora todos están multiplicados por $\frac{1}{\sqrt{2\pi}\ \hat{\sigma}}$, luego basta con encontrar quien maximice:

$$\hat{\pi}_l \ \exp^{-\frac{1}{2\hat{\sigma}^2}(x^* - \hat{\mu}_l)^2}$$

Si aplicamos ln a todas las opciones:

$$\ln \hat{\pi}_l - \frac{1}{2\hat{\sigma}^2}(x^* - \hat{\mu}_l)^2 \quad \forall\ l \in \{0,1\}$$

Y luego acomodando los términos,

$$\ln \hat{\pi}_l - \frac{1}{2\hat{\sigma}^2}\left((x^*)^2 - x^*\hat{\mu}_l - \hat{\mu}_l{}^2\right) \ \forall l \in \{0,1\}$$

$$\ln \hat{\pi}_l - \frac{(x^*)^2}{2\hat{\sigma}^2} + \frac{x^*\hat{\mu}_l}{\hat{\sigma}^2} - \frac{\hat{\mu}_l^2}{2\hat{\sigma}^2} \ \forall l \in \{0,1\}$$

Como todas tienen $-\frac{(x^*)^2}{2\hat{\sigma}^2}$, se puede simplificar a encontrar l^* tal que maximice:

$$\delta_l(x^*) = x^* \ \frac{\hat{\mu}_l}{\hat{\sigma}^2} - \frac{\hat{\mu}_l^2}{2\hat{\sigma}^2} + \ln \hat{\pi}_l$$

□

Hemos hallado una manera más sencilla de expresar el clasificador. Luego, para implementarlo es necesario calcular:

- $\hat{\pi_0}, \hat{\pi_1}$: descripta en (2)
- $\hat{\mu_0}, \hat{\mu_1}$: descripta en (4)
- $\hat{\sigma}$: descripta en (20)

Veamos como podemos caracterizar el clasificador en el caso en el cual tengamos la misma cantidad de datos de cada categoría:

Proposición 1.1.2. Si $n_0 = n_1$, luego el punto de frontera en donde termina una región de clasificación y comienza la siguiente viene dado por:

$$\frac{\hat{\mu_0} + \hat{\mu_1}}{2}$$

El resultado de la proposición se ilustra en la siguiente figura.

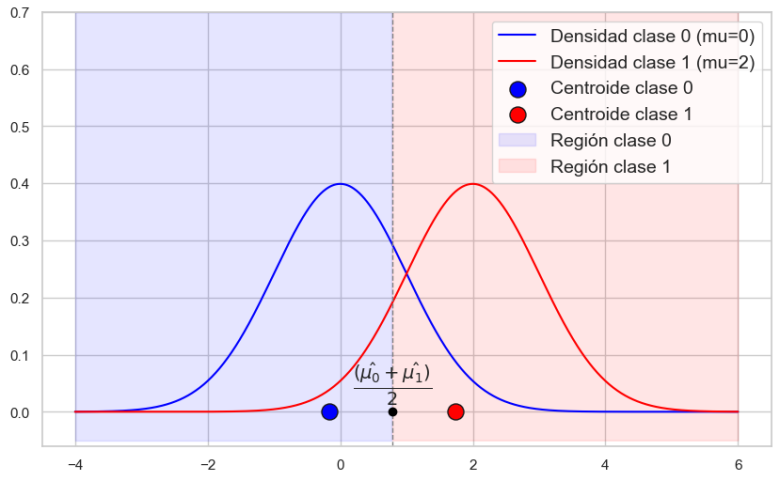


Figura 1.1: Clasificación LDA con una sola co-variable
Clasificación generada con 20 datos de normales con media 0 y 2 y varianza 1.

Demostración. Para asignar el punto x al grupo $Y = 0$, debe suceder que:

$$\delta_0(x) > \delta_1(x)$$

Si reemplazamos por la definiciones de $\delta(x)$ en (7), obtenemos:

$$x\ \frac{\hat{\mu_0}}{\sigma^2} - \frac{\hat{\mu_0}^2}{2\sigma^2} + \ln \hat{\pi_0} > x\ \frac{\hat{\mu_1}}{\sigma^2} - \frac{\hat{\mu_1}^2}{2\sigma^2} + \ln \hat{\pi_1}$$

Si asumimos que tenemos la misma cantidad de observaciones pertenecientes al grupo 0 que al grupo 1, $\hat{\pi_0} = \hat{\pi_1}$:

$$x\ \frac{\hat{\mu_0}}{\sigma^2} - \frac{\hat{\mu_0}^2}{2\sigma^2} > x\ \frac{\hat{\mu_1}}{\sigma^2} - \frac{\hat{\mu_1}^2}{2\sigma^2}$$

Acomodando los términos,

$$\begin{aligned} x\ (\frac{\hat{\mu_0} - \hat{\mu_1}}{\hat{\sigma}^2})\ &>\ \frac{\hat{\mu_0}^2 - \hat{\mu_1}^2}{2\hat{\sigma}^2} \\ x\ (\hat{\mu_0} - \hat{\mu_1})\ &>\ \frac{\hat{\mu_0}^2 - \hat{\mu_1}^2}{2} \\ x\ &>\ \frac{\hat{\mu_0}^2 - \hat{\mu_1}^2}{2\ (\hat{\mu_0} - \hat{\mu_1})} \end{aligned}$$

Aplicando diferencia de cuadrados

$$x\ >\ \frac{\hat{\mu_0} + \hat{\mu_1}}{2}$$

Luego, la frontera que divide ambas regiones es el punto $x = \frac{\hat{\mu_1} + \hat{\mu_2}}{2}$ □

Obtuvimos que, para el escenario en el que $n_0 = n_1$, clasificaremos x^* a la categoría 0 o 1, según sea más grande o más pequeña que el promedio de los centroides:

$$x^* > \frac{1}{2\ n_0} \left(\sum_{i:y_i=0} x_i - \sum_{j:y_j=1} x_j \right)$$

Escenario con múltiples co-variables

Nuestro objetivo es hallar la regla de clasificación basada en atribuir una nueva observación x^* a la clase l^* que maximice 1 en el escenario en el que haya múltiples co-variables, o equivalentemente, $p > 1$ y dos categorías, $l \in \{0, 1\}$.

Para poder tener una fórmula cerrada, nos faltaba estimar $f_l(x^*)$ que supusimos que tenía distribución $f_l(X) \sim \mathcal{N}(\mu_l, \Sigma)\ \forall\ l \in \{0, 1\}$. Luego,

$$f_l(x^*) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp^{-\frac{1}{2}(x^* - \mu_l)^T \Sigma^{-1} (x^* - \mu_l)} \quad \forall l \in \{0, 1\} \tag{8}$$

Los estimadores de los parámetros desconocidos son:

- $\mu_l \ \forall \ l \in \{0,1\}$ la media poblacional de cada categoría y se estima usando la muestral como en (4)
- Σ la matriz de covarianza en común para todas las categorías y se estima así:

$$\hat{\Sigma} = \frac{1}{n}\sum_{l=0}^{1}\sum_{i:y_i=l}(x_i - \hat{\mu_l})(x_i - \hat{\mu_l})^T \tag{9}$$

El criterio para clasificar es asignar x^* a l^* que maximice $P(Y=0|X=x^*), P(Y=1|X=x^*)$. Al realizar el plug-in de la estimación de (8) en (1) tenemos:

$$P(Y=l|X=x^*) \approx \frac{\hat{\pi_l}\frac{1}{(2\pi)^{p/2}|\hat{\Sigma}|^{1/2}}\exp^{-\frac{1}{2}(x^*-\hat{\mu_l})^T\hat{\Sigma}^{-1}(x^*-\hat{\mu_l})}}{\sum_{Y\in\{0,1\}}\hat{\pi_Y}\frac{1}{(2\pi)^{p/2}|\hat{\Sigma}|^{1/2}}\exp^{-\frac{1}{2}(x^*-\hat{\mu_Y})^T\hat{\Sigma}^{-1}(x^*-\hat{\mu_Y})}} \tag{10}$$

En consecuencia, la regla de clasificación es hallar l^* que maximice (10). Análogamente a el caso $p=1$, reescribimos el clasificador:

Proposición 1.1.3. Dado un nuevo x^* que se desea clasificar a través de LDA entre 2 categorías con $p>1$. La clase l^* que maximiza las aproximaciones de $P(Y=0|X=x^*), P(Y=1|X=x^*)$ dadas en 10 es equivalente a la que maximiza:

$$\delta_l(x^*) = (x^*)^T\ \hat{\Sigma}^{-1}\hat{\mu_l} - \frac{1}{2}\hat{\mu_l}^T\hat{\Sigma}^{-1}\hat{\mu_l} + \ln\hat{\pi}_l \qquad l\in\{0,1\}$$

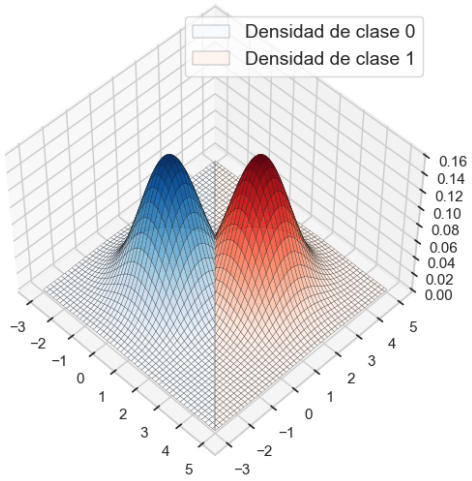
donde

- $\hat{\pi_0}, \hat{\pi_1}$ son los estimadores definido en (2)
- $\hat{\mu_0}, \hat{\mu_1}$ son los estimadores definido en (4)
- $\hat{\Sigma}$: es el estimador definido en (9)

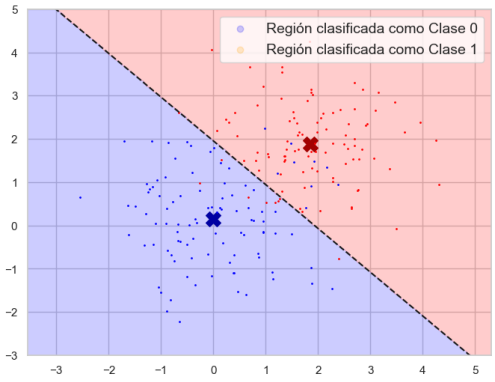
La prueba es idéntica al caso $p=1$, lo que es ligeramente distinto es el relacionado con su geometría. Si asumimos que tenemos la misma cantidad de observaciones en cada categoría, la separación de los dos grupos es una recta. La misma no necesariamente es aquella que separa $\hat{\mu_0}$ de $\hat{\mu_1}$ y viene dada por los x que cumplen:

$$x^T\ \hat{\Sigma}^{-1}\hat{\mu_0} - \frac{1}{2}\hat{\mu_0}^T\hat{\Sigma}^{-1}\hat{\mu_0} = x^T\ \hat{\Sigma}^{-1}\hat{\mu_1} - \frac{1}{2}\hat{\mu_1}^T\hat{\Sigma}^{-1}\hat{\mu_1}$$

En el gráfico 1.2 se puede apreciar la recta que los separa.



(a) Densidad de las clases
En azul, clase 0, con media (0,0). En rojo, clase 1, con media (2,2). Ambas poseen una identidad como matriz de covarianza.



(b) Regiones de clasificación
Clasificación generada con 20 datos creados a partir de las distribuciones del gráfico que está a la izquierda

Figura 1.2: Clasificación con LDA con dos co-variables

1.1.2. Support Vector Machine

Support Vector Machine (SVM) pertenece al grupo de algoritmos de clasificación que se basan en geometría. La idea es separar el espacio de co-variables en 2 regiones y asignar la categoría a una nueva observación según a la región que pertenezcan.

A continuación veremos cómo generar las regiones. Por un lado, primero analizaremos el caso en donde los datos estén en regiones separables. Luego, abordaremos las situaciones donde no lo sean. Por último, detallaremos cómo generalizar el método para casos no lineales.

Para facilitar la notación, supondremos que las dos categorías son $\{-1, 1\}$ en vez de $\{0, 1\}$.

Caso separable

Comencemos por el caso en el que las clases son separables por un hiperplano, es decir, $\exists\ (\beta_0,\ ..,\ \beta_p)$ tal que los puntos x que cumplen

$$\beta_0\ +\ \beta_1 x_1\ +\ ...\ +\ \beta_p x_p = 0 \tag{11}$$

separan el espacio en 2 regiones que dejan de un lado todas las observaciones de una categoría, y las del otro, del lado opuesto. En el caso de dimensión dos, será una recta. Si la dimensión es tres, será un plano.

La idea que los dos grupo de puntos queden en regiones distintas se puede expresar matemáticamente como:

$$\beta_0 + \beta_1 x_{j,1} + ... + \beta_p x_{j,p} > 0 \quad \forall\, j/Y_j = 1, \tag{12}$$

$$\beta_0 + \beta_1 x_{j,1} + ... + \beta_p x_{j,p} < 0 \quad \forall\, j/Y_j = -1. \tag{13}$$

Naturalmente, si ambos grupos de puntos son separables por el hiperplano, podemos clasificar una nueva categoría según la región en la que caigan. Dicha idea se expresa como, dado x^* una nueva observación, evaluaremos $f(x^*) = \beta_0 + \sum_{i=0}^{p} \beta_i x_i^*$ y según el signo del resultado será catalogado como una u otra región.

Como podemos observar, existe más de un hiperplano que las separa:

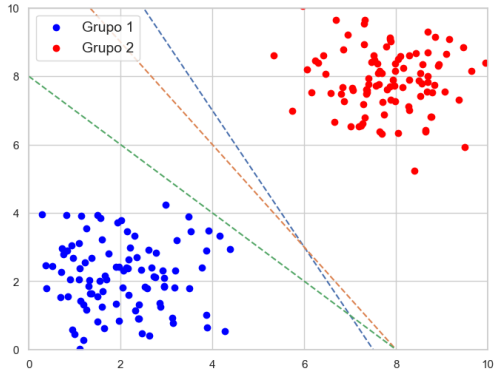
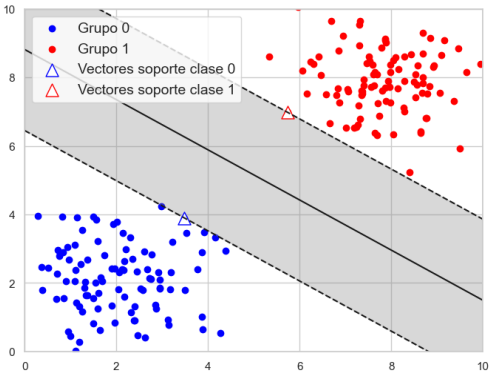


Figura 1.3: Infinitas rectas separan los dos grupos

Vamos a elegir aquel que, además de seperarlas, mantiene la mayor distancia posible con todos puntos, sean de una u otra clase. Es decir, el que maximiza la distancia mínima con todos los puntos:



A los puntos marcados con triángulos, que realicen la distancia M de cada grupo, serán denominados vectores de soporte, dándole origen al nombre del método. Además,

cabe destacar que la ubicación del separador de categorías depende exclusivamente de ellos. Perturbar un poco el resto de los valores de otras observaciones que no sean de soporte no modificará el resultado. En particular, implica que es robusto en cuanto a *outliers* se refiere.

Para encontrar los valores $(\beta_0, ..., \beta_p)$ que representen el hiperplano separador, se debe resolver el siguiente sistema de ecuaciones lineales:

$$\begin{aligned} \underset{\beta_0,\beta_1,...,\beta_p}{\text{máx}} \quad & M \\ \text{sujeto a} \quad & \sum_{j=1}^{p} \beta_j^2 = 1, \\ & y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M \quad \forall \quad i \in \{1, \ldots, n\} \end{aligned} \tag{14}$$

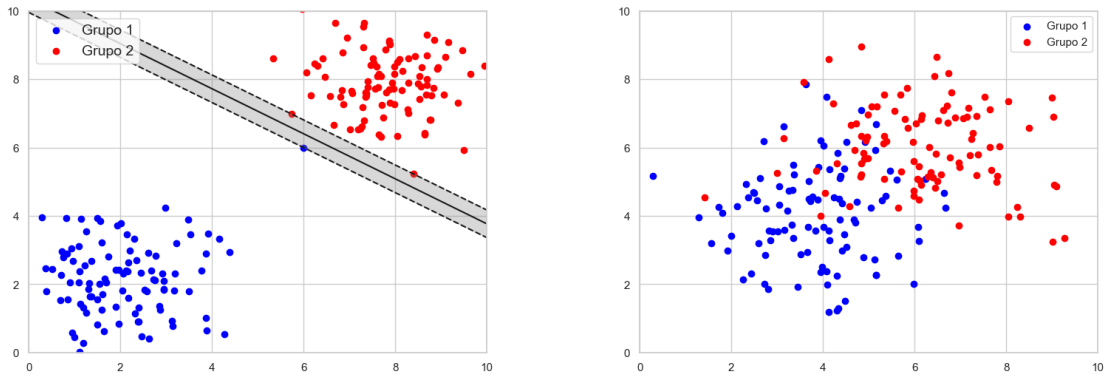
En particular:

- si $y = 1$, la segunda ecuación implica (12) y si $y = -1$, (13)
- La primera ecuación, que restringe a que la suma de los coeficientes sea 1, se agrega para darle unicidad a la solución. La motivación es que de lo contrario podríamos agrandar los β para agrandar M. Esto se debe a que $y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M$, luego $y_i(\beta_0 + 2\beta_1 x_{i1} + 2\beta_2 x_{i2} + \cdots + 2\beta_p x_{ip}) = y_i(\beta_0 + 2 < \beta, x_i >) \geq 2M - |\beta_0|$ En otras palabras, asegura que $(\beta_1, \ .., \ \beta_p)$ este en la esfera unitaria y evita soluciones triviales.
- M es el ancho del margen, la distancia a todos los puntos a maximizar.

Este método es conocido como el clasificador de máximo margen. Una pregunta natural que surge al plantear un sistema de ecuaciones no lineales es si es posible encontrar la solución al problema. La respuesta es afirmativa, pero no lo abordaremos en esta tesis.

Caso no separable

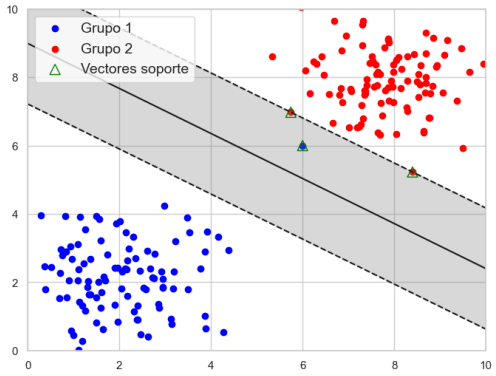
Para comenzar, veamos conjuntos para los cuales es necesario una clasificación no lineal. Por un lado, están los conjuntos de datos que no son linealmente separables (como en la figura 1.4b). E inclusive en el caso separable, no siempre es deseable catalogar a todas las observaciones correctamente. Esto puede llevar a agruparlos de manera anti natural, causando *overfitting* como se ve en a figura 1.4a:



(a) Clasificación antinatural (b) Datos no separables

Figura 1.4: Ejemplos motivadores para SVM no lineal

A partir de ahora, nos interesa permitirle a una observación que esté tanto en el margen como del lado incorrecto. Para ello vamos a penalizar a aquellas que lo hagan. Si lo aplicamos a los datos de la figura 1.4a, a cambio de tener una observación mal clasificada, las regiones de clasificación parecen más naturales:



Al agregar está penalización al modelo matemático descrito en (14), el problema se expresa como encontrar el hiperplano dado por $(\beta_0, ..., \beta_p)$ que cumpla:

$$\begin{aligned} \underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{máx}} \quad & M \\ \text{subject to} \quad & y_i \left(\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} \right) \geq M(1-\epsilon_i) \quad \forall \quad i \in \{1,\ldots,n\}, \\ & \sum_{i=1}^{n} \epsilon_i \leq C, \quad \epsilon_i \geq 0 \quad \forall i \in \{1,\ldots,n\}, \\ & \sum_{j=1}^{p} \beta_j^2 = 1. \end{aligned} \tag{15}$$

En particular:

- La primera y la última ecuación representan, al igual que antes, el cumplimiento de la clasificación de cada observación y evitar soluciones triviales, respectivamente. A diferencia de antes, ahora se le permite quedar dentro de la franja, o del lado equivocado, como buscábamos.

- Los ϵ_i representan por cuánto incumple cada par x_i, y_i la restricción de estar en su región y fuera del margen.

  Si $\epsilon_i = 0$, implica que el par x_i, y_i fue correctamente clasificado.

  Si $0 < \epsilon_i \leq 1$, está incumpliendo el margen pero no la región.

  Si $\epsilon_i > 1$, está en la región equivocada.

- C es la tolerancia máxima que vamos a permitirle a todas las instancias. Si $C = 0$, luego estamos en el caso previamente descrito.

- M sigue siendo el margen que cumplen la mayoría de las observaciones. A más grande sea C, menos tenderán a cumplirlo.

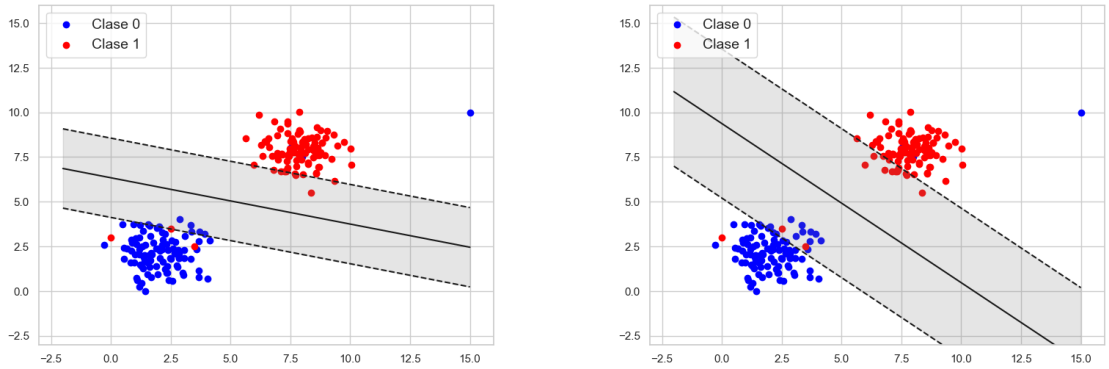
- Una vez que obtenemos el hiperplano, seguimos clasificando a las nuevas observaciones x^* según la región en la que caen, es decir, el signo de $f(x^*) = \beta_0 + \sum_{i=1}^{p} \beta_i x_i^*$. Las tolerancias son usadas para generarlo solamente.

Esta manera de clasificar sigue dependiendo de los vectores *de soporte*, que son aquellos que realizan el margen, están dentro del mismo o en la región incorrecta. Va a poseer más vectores de soporte que antes, por lo cual, es aún más robusto frente a *outliers*. Ya sean perturbando puntos que no sean de soporte, como antes, o para modificaciones de los de soporte.

Como mencionamos en la introducción, hay parámetros (como C) para los cuales el algoritmo funciona independientemente de su valor. En nuestro caso supondremos que C que está fijo, pero su valor impactara en el trade-off entre sesgo y varianza.

- A valores bajos de C, pediremos que los puntos se clasifiquen de manera correcta y tendrá poca capacidad de generalización.

- Si C es muy grande, estamos permitiendo que muchos valores incumplan las restricciones, pero a cambio, una mayor capacidad de generalización.



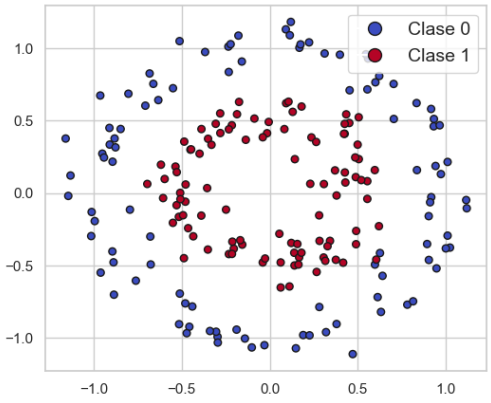
(a) C pequeño
Pocos mal clasificados, poca generalización

(b) C grande
Varios mal clasificados, pero generaliza mejor

Figura 1.5: Clasificación SVM con varios valores de C

Caso no lineal

Hasta ahora, planteamos que *Support Vector Machine* se origina de pensar el problema para dos clases y pensando en separar ambas de manera lineal. Hay diversos escenarios en donde esa forma de clasificar separando en un hiperplano no es suficiente. Trabajemos con el siguiente ejemplo y veamos cómo se adapta el algoritmo para luego ver la generalización:



En este caso, ninguna recta separa el espacio de manera que la clasificación resulte natural.

La generalización del método comienza por pensar en agrandar el espacio de co-variables. Al igual de lo que sucede en otros algoritmos, como las regresiones lineales o logísticas, SVM pide linealidad en los $\beta_0,\ ..,\ \beta_p$, pero los x_i pueden tomar la forma que deseemos. Entonces podríamos pensar el espacio de co-variables como

$$X_1,\ X_1^2,\ ..,\ X_p,\ X_p^2 \tag{16}$$

El algoritmo encontrará un hiperplano que separe ambas clases en el espacio elegido. Para clasificar una nueva observación x^*, deberemos llevarla al espacio agrandado y ver en qué región cae. Lo interesante es que al pensar en las regiones de clasificación en el espacio original, es decir, $x \in \mathbb{R}^p$, resultan no ser lineales. En nuestro ejemplo se puede apreciar como son adentro y afuera de un círculo:

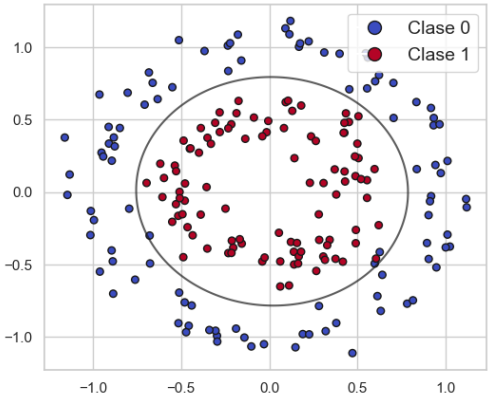


Figura 1.6: Clasificación SVM con kernel polinomial de grado 2
Si bien no parece una recta, en la dimensión dada por el polinomio de grado 2, lo es.

Para poder obtener $(\beta_0, \beta_{11}, ...\beta_{2p})$ que representan el hiperplano en el nuevo espacio, debemos resolver las ecuaciones para casos no separables (15), pero agregando las nuevas co-variables:

$$\begin{aligned} &\max_{\beta_0,\beta_{11},\beta_{12},\ldots,\beta_{p1},\beta_{p2},\epsilon_1,\ldots,\epsilon_n} \quad M \\ &\text{subject to} \quad y_i \left(\beta_0 + \sum_{j=1}^{p} \beta_{j1} x_{ij} + \sum_{j=1}^{p} \beta_{j2} x_{ij}^2 \right) \geq M(1-\epsilon_i) \forall \quad i \in \{1, \ldots, n\}, \\ &\sum_{i=1}^{n} \epsilon_i \leq C, \quad \epsilon_i \geq 0, \\ &\sum_{j=1}^{p} \sum_{k=1}^{2} \beta_{jk}^2 = 1. \end{aligned} \tag{17}$$

El problema se reduce a resolver el sistema de ecuaciones y a clasificar nuevas observaciones x^*, según el signo de $f(x^*) = \beta_0 + \sum_{j=1}^{p} \beta_{j1} x_j^* + \sum_{j=1}^{p} \beta_{j2} (x_j^*)^2$.

La manera en la que acabamos de expandir el espacio para hallar las regiones de clasificación es arbitraria. A continuación, detallamos el método general.

Hasta ahora siempre que hemos hablamos del planteo del algoritmo, lo hemos hecho a través de las restricciones. Existen además certezas sobre cómo luce la función f, esa

que utilizamos para determinar en que región cae una nueva observación. En el caso lineal, se puede expresar como, dado x^* una nueva observación:

$$f(x^*) = \alpha_0 + \sum_{i=1}^{n} \alpha_i \ < x^*, x_i > \tag{18}$$

Si bien es cierto que tenemos $n+1$ valores de α, la clasificación solamente depende de los vectores soporte en el espacio transformado. Y, entonces, los únicos casos donde $\alpha_i \neq 0$ serán para los cuales x_i sea un vector de soporte.

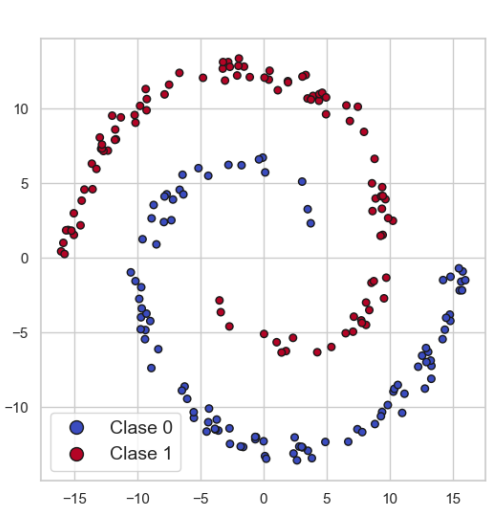
Para generalizar el algoritmo, debemos reemplazar el producto interno de $\mathbb{R}^p$ en 18. Existe una generalización del mismo, los *kernels*, $K(x, y)$. Cuantifican similitud entre vectores. Entonces, la función f que determina la clase en el caso general se puede expresar como:

$$f(x^*) = \alpha_0 + \sum_{i=1}^{n} \alpha_i \ K(x^*, x_i) \tag{19}$$

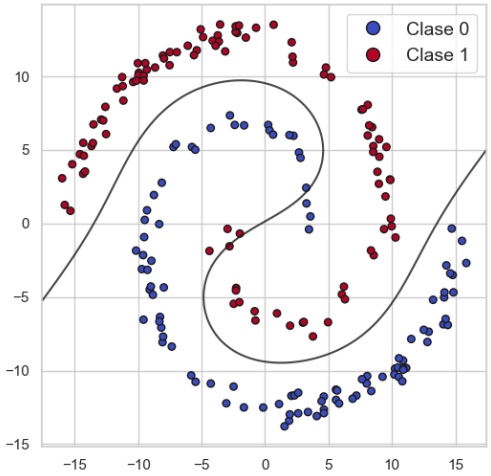
Los más conocidos son:

- lineal: $K(x_i, x_{i'}) =< x_i, x_{i'} >$. Miden similitud semejante a la correlación de Pearson. Un ejemplo práctico en donde se usa este kernel es la figura 1.5.
- polinomial de grado d: $K(x_i, x_{i'}) = (1+ < x_i, x_{i'} >)^d$. Un ejemplo práctico en donde se usa este kernel es la figura 1.6.
- radial: $K(x_i, x_{i'}) = exp\,(-\gamma \ ||x_i - x_{i'}||)$. La similitud que miden es de cercanía o lejanía. Valores cerca de 0 implica que están muy lejos, mientras que a más juntos de 1, más cerca estarán. Un ejemplo práctico en donde se usa este kernel es en la figura 1.7.

La información proporcionada contempla los elementos fundamentales para comprender el algoritmo, si se deseara conocer la intuición del procedimiento para obtener los α_i se aconseja leer la bibliografía.



(a) Datos en forma de espiral



(b) Regiones de clasificación dada por radial kernel

Figura 1.7: Clasificación radial kernel

Al igual que para el caso no separable, poseemos parámetros para los cuales el algoritmo funciona para cualquier valor que supusimos fijos en las definiciones, pero que en la práctica habrá que hacer validación cruzada para elegir el que mejor se ajuste a nuestros datos. Estos son el *kernel* y C.

1.2. Clasificación con L categorías

Abordaremos ahora el problema de clasificación multiclase. Consta de asignar una categoría posible a observaciones nuevas entre todas las presentes. Posee diversas aplicaciones, entre las que están: cuál es la enfermedad de una persona basada en sus síntomas, tenemos imágenes o fotos por ejemplo de diversas frutas y queremos decidir para futuras fotos cuáles frutas son.

El problema a resolver será poder clasificar nuevas observaciones $x^* \in \mathbb{R}^p$ en su categoría correcta (podrá ser $Y \in \{1, ..., L\}$). Para ello, haremos uso de nuestro conjunto de datos de entrenamiento, $(x_i, y_i) : \ i \in \{1, \ldots, n\}$ para generar la regla de clasificación. A la cantidad de datos por categoría la llamaremos $n_i : \ i \in \{1, \ldots, L\}$.

En el apartado anterior, vimos cómo clasificar usando LDA y SVM para dos categorías. A continuación, vamos a focalizarnos en cómo generalizarlos para el caso multiclase.

Los ejemplos presentes en esta sección del capítulo serán datos sintéticos, generados para ilustrar algún comportamiento específico.

1.2.1. Análisis de discriminación lineal

Comencemos por recordar que LDA, en el caso de dos categorías, se basa en la idea de clasificar a una nueva observación x^* según la clase para la cual se maximice $P(Y = l|X = x^*)$ entre $l = 0$ y $l = 1$. Si en vez de dos categorías tenemos L, bastará con aproximar $P(Y = l|X = x^*)\ \forall\ l \in \{1, ..., L\}$. y asignarlo a la clase que maximice dichas probabilidades.

Siguiendo con la linea de razonamientos realizada al analizar el caso para dos categorías, podemos reescribir dicha probabilidad usando Bayes, llegando a (1). Al igual que antes, hay cantidades que desconocemos y tendremos que estimar. Son $\pi_l,\ f_l(x)\ \forall\ \{1,\ ...,\ L\}$. Asumimos que $f_l(X) \sim \mathcal{N}(\mu_l, \Sigma)$ (o $f_l(X) \sim \mathcal{N}(\mu_l, \sigma^2)$ en $p = 1$). Usaremos las mismos valores muestrales de los parámetros μ_l y σ/Σ. La aproximación de σ^2 en su versión generalizada para L clases es:

$$\hat{\sigma}^2 = \frac{1}{n - L} \sum_{l=0}^{L-1} \sum_{i:y_i=l} (x_i - \hat{\mu}_l)^2 \tag{20}$$

Luego, si suponemos que tenemos una única co-variable, obtenemos de modo equivalente a (1.1.2), que el problema se reduce en encontrar l^* que maximice:

$$\delta_l(x^*) = x^*\ \frac{\hat{\mu}_l}{\hat{\sigma}^2} - \frac{{\hat{\mu}_l}^2}{2\hat{\sigma}^2} + \ln \hat{\pi}_l \quad \text{con } l \in \{1,\ ...,\ L\}.$$

Si prestamos especial atención a su demostración, en ningún momento usamos que eran solamente dos categorías.

Análogamente, para el caso que $p > 1$, se sostiene la equivalencia dada en (1.1.3) y el problema se traduce a encontrar l^* tal que se maximice:

$$\delta_l(x^*) = (x^*)^T\ \hat{\Sigma}^{-1}\hat{\mu}_l - \frac{1}{2}{\hat{\mu}_l}^T\hat{\Sigma}^{-1}\hat{\mu}_l + \ln \hat{\pi}_l \quad \text{con } l \in \{1,\ ...,\ L\}.$$

En conclusión, para resolver el problema de LDA con múltiples categorías, dada una nueva observación x^* basta con calcular $\delta_1(x^*),\ ..,\ \delta_L(x^*)$ usando los valores muestrales del conjunto de entrenamiento $\hat{\mu}_l,\ \hat{\Sigma},\ \hat{\pi}_l$ y asignárselo a la categoría que realice su máximo. Veamos un ejemplo:

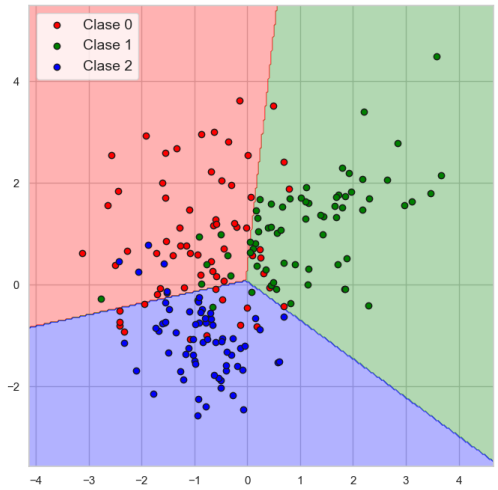


Figura 1.8: Clasificación LDA con 3 categorías

Podemos observar que, al igual que cuando eran tan solo dos categorías, vale que la separación de las regiones de clasificación viene dada por una recta. Esto valdrá siempre que se trabaje en esta dimensionalidad.

1.2.2. Support Vector Machine

Como mencionamos anteriormente Support Vector Machine es un algoritmo basado en intuición geométrica, su idea principal es separar categorías con hiperplanos, ya sea en el espacio de origen o uno más grande, y clasificar x^* según la región en la que esté. Resulta que esta idea no se puede extender para múltiples clases de manera natural. Por suerte, se puede abordar realizando varias comparaciones del caso más sencillo. Dos maneras entre las más conocidas de usarlo son:

- 1 vs 1: Dada una nueva observación, x^*, ejecutamos SVM para saber si dadas dos categorías, a cuál lo asignaríamos (usando solamente los datos de entrenamientos provenientes de las mismas). Lo realizamos con todas las combinaciones posibles, es decir, $\binom{L}{2}$ veces. Contabilizamos en cuántas instancias fue asignado a cada una. Lo asignaremos a la que más veces haya ganado. A continuación, vemos un ejemplo:

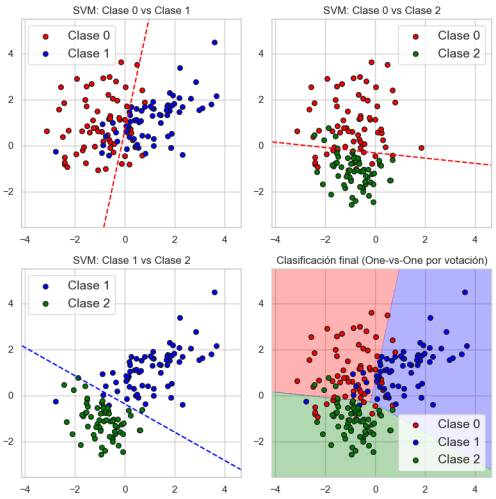


Figura 1.9: Clasificación SVM generada con uno vs uno
Se puede apreciar como se generan los votos que terminan dando pie a la clasificación.

- 1 vs el resto: Dada una nueva observación, x^*, vamos a ejecutar L comparaciones. En cada una, entrenaremos el clasificador para distinguir entre una clase sola y el resto de los puntos. Esto nos generará L clasificaciones con sus respectivos generadores de valores de decisión $f_l(x) \ : \ i \in \{1, \ldots, L\}$.

  Las potenciales clases a las que asignar x^* son aquellas en donde al compararlas contra el resto, lo asignamos a la clase solitaria. En particular, entre las opciones que ganen, elegimos l^* la clase cuyo valor de decisión $f_l(x^*)$ sea el más alto. En el caso que el kernel fuese lineal, representaría a la clase cuya distancía al hiperplano separador vs el resto fuese mayor.

  A continuación, un ejemplo de su funcionamiento:

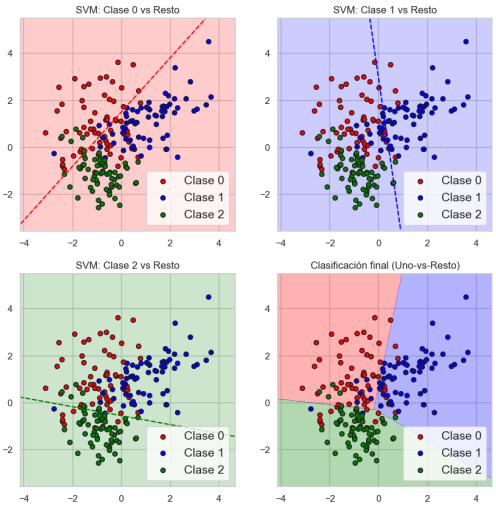


Figura 1.10: Clasificación SVM uno vs el resto
Se puede apreciar como se compara los datos de cada categoría vs los del resto.

1.3. Validación cruzada

En las secciones anteriores vimos como clasificar una nueva observación x^* en los casos de LDA y SVM. En la realidad, vamos a tener un único conjunto de n datos y diversos algoritmos (supongamos LDA y SVM). Queremos poder decidir qué método usaremos para clasificar nuevas instancias. Podríamos pensar por ejemplo, que tenemos un conjunto de datos históricos de pacientes en un hospital. Usándolos, queremos decidir que cómo vamos a diagnosticar que enfermedad padecen los nuevos pacientes que vengan en el futuro.

Para poder usar un algoritmo, debemos elegir su *receta*. La misma está compuesto por:

- PARÁMETROS INTERNOS: son aquellos que son necesarios obtener para el funcionamiento del algoritmo (como π_l, μ_l y Σ en LDA)
- HIPERPARÁMETROS: parámetros que debemos elegir para el funcionamiento del algoritmo. Un ejemplo es el *kernel* en SVM. El mismo podría funcionar con cualquiera de las opciones, pero necesitamos elegir una para completar la receta.

  Pueden ser continuos o discretos.

Cada conjunto de hiperparámetros, tiene sus parámetros internos necesarios para funcionar.

Nuestro objetivo es elegir el algoritmo junto con su receta que mejor se ajuste a nuestros datos de cada uno.

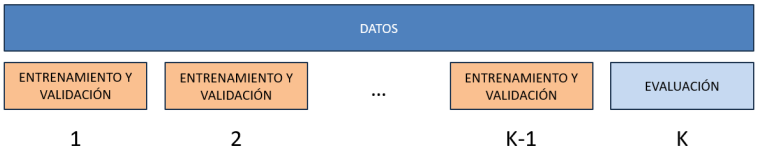
Para comparar dos recetas, que pueden o no provenir del mismo algoritmo, vamos a usar una métrica de confianza. Dado una combinación algoritmo/receta f, evaluaremos su capacidad de generalización en datos que no hayan sido usados para entrenarlo $(x_1, y_1), ...(x_{n'}, y_{n'})$. Cuantificamos la confianza como la tasa de aciertos:

$$Exactitud(f, (x_1, y_1), ..., (x_{n'}, y_{n'})) = \frac{\sum_{i=1}^{n'} 1_{f(x_i)=y_i}}{n'} \tag{21}$$

Una vez que tengamos la mejor receta para cada algoritmo, debemos elegir el ganador.

Un proceso que resuelve todo el problema es *validación cruzada*. Consiste en dividir el conjunto de datos en K grupos.

Serán utilizados para:

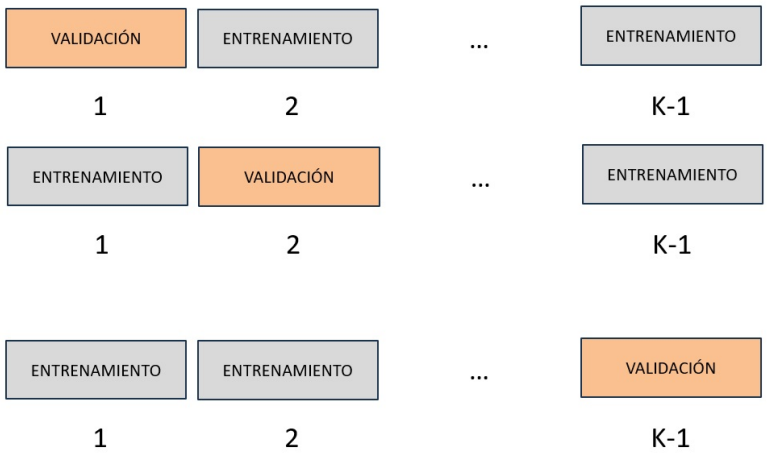


- DATOS DE ENTRENAMIENTO Y VALIDACIÓN: son $K-1$ grupos de datos. Serán utilizados para encontrar la receta de cada algoritmo que mejor se ajuste a la naturaleza de nuestro problema.

  Supongamos que queremos decidir en SVM si el *kernel* será lineal o cuadrático y que este es el único hiperparámetro a decidir (fijemos $C = 1$).

  Comencemos por obtener el valor de confianza del *kernel* lineal. Para obtener un valor necesitamos un conjunto de datos en donde entrenar el algoritmo y otro en donde evaluarlo. Para esto último, hay $K-1$ grupos disponibles para elegir. Entonces, generaremos $K-1$ valores de exactitud y los promediaremos. Para cada uno, seleccionaremos los conjuntos como:

  - *entrenamiento:* usaremos $K-2$ conjuntos. Rotamos el que no usamos.
  - *validación*: usaremos el i-ésimo conjunto para generar el valor de confianza para los datos de entrenamiento.



Promediando los $K-1$ valores de exactitud obtenidos, obtenemos la métrica. Hacemos lo mismo con el *kernel* cuadrático. Tenemos entonces valores para comparar y elegir si preferimos ir con el lineal o el cuadrático.

Ahora que tenemos el conjunto de hiperparámetros ganadores para SVM, nos faltan los parámetros internos. Para no desperdiciar datos, usamos los $K-1$ grupos para conseguirlos.

Análogamente, podemos encontrar recetas para otros algoritmos.

Resta elegir que combinación receta - algoritmo usaremos para nuevas observaciones.

- DATOS DE EVALUACIÓN: 1 grupo o *fold*. Son las observaciones nuevas, es decir, que no han sido utilizadas para decidir las recetas de ningún algoritmo. Se usarán para comparar sus rendimientos y ver su capacidad de generalización. Elegimos la que posea una mayor exactitud y la utilizamos para clasificar nuevas observaciones que vayan surgiendo.

A lo largo de esta tesis nos centraremos en casos en los cuales no tenemos muchas observaciones en donde esta técnica nos será de mucha utilidad.

En conclusión, ya sabemos como funcionan internamente los algoritmos y como compararlos entre sí para elegir cuál usar.

Capítulo 2

Propuesta cuando el número de clases aumenta con el tamaño de muestra

Los problemas de clasificación multiclase frecuentemente surgen en entornos de alta dimensión, donde no solo el número de variables (p) es grande, sino también la cantidad de clases (L), sin embargo el tamaño de muestra por clase (n) puede ser limitado. Este escenario, es cada vez más frecuente en aplicaciones como la genómica, el reconocimiento de imágenes o el procesamiento de lenguaje natural. A pesar de su relevancia, y de los avances recientes en el desarrollo de técnicas para abordarlo el análisis teórico es algo limitado.

En este Capítulo estudiamos la propuesta presentada en [1], quienes abordan este problema desde un enfoque teórico y ofrecen un marco analítico sólido para evaluar la exactitud de la clasificación multiclase en alta dimensión. Los autores consideran un escenario ampliamente utilizado: la clasificación de vectores normales de alta dimensión, permitiendo que el número de clases crezca con la dimensión del espacio de características. En este contexto, derivan condiciones no asintóticas sobre el efecto de las variables relevantes, así como cotas inferiores y superiores para las distancias entre clases necesarias para lograr una selección de variables y una clasificación precisa. En las secciones siguientes presentaremos el modelo, describiremos la propuesta y resumiremos los resultados obtenidos en trabajo de Abramovich y Pensky [1].

2.1. Escenario analizado

El problema de clasificación a analizar es multiclase con L categorías y vectores observados de dimensión p. Utilizaremos la notación propuesta por los autores, donde dada una categoría, l, su conjunto de datos será $\{Y_{lj}\}_{1\leq j\leq n_j}$. Nos enfocamos en el estudio

del caso en donde los datos de cada clase provengan de distribuciones normales $Y_{l=l'} \sim \mathcal{N}(m_{l'}, \Sigma)\ \forall\ l' \in \{1, ..., L\}$. La matriz de covarianza $\Sigma \in \mathbb{R}^{p \times p}$ se asume invertible y común a todas las clases.

El escenario de interés corresponde a situaciones donde tanto el número de clases L como la cantidad de variables p son elevadas, mientras que el número de muestras por clase n_l pueda ser reducido.

2.1.1. Limitaciones de LDA

Si recordamos lo visto en el capítulo anterior, el modelo Análisis del Discriminante Lineal (LDA) asumía la misma distribución normal sobre los datos de entrenamiento con co-varianza compartida. Veamos qué sucede con el algoritmo al exponerlo a las condiciones a analizar.

En el siguiente ejemplo, analizamos el comportamiento al aumentar la cantidad de categorías. Trabajamos en dimensión $p = 500$, de las cuales 10 son información. Asumimos 20 datos por clase. Empezamos con 20 clases y vamos agregando de a una. Al tener más opciones para elegir, la exactitud del modelo disminuye.

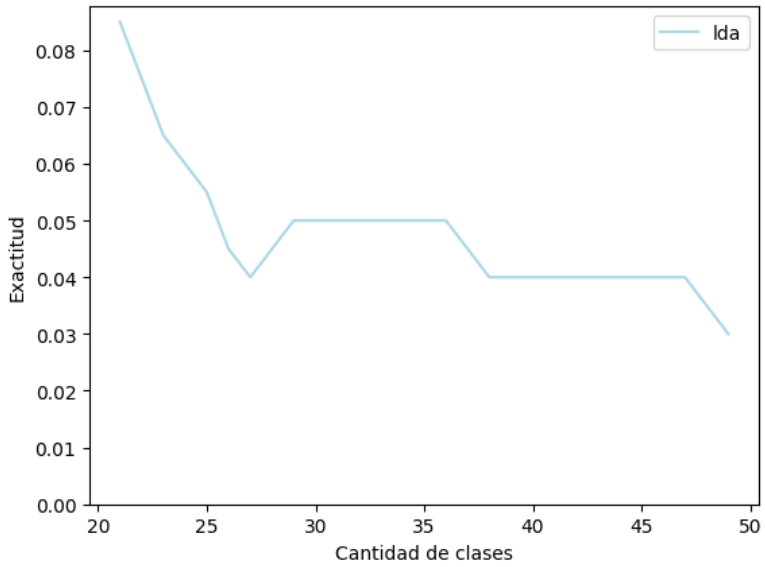


Figura 2.1: Problemas de LDA al agregar variables
Evolución de la exactitud al agregar las categorías en p=500 para LDA.

Para más detalles sobre cómo se generan los datos, ver el capítulo siguiente.

Como podemos observar, el algoritmo no es capaz de sostener su exactitud al tener más categorías. En la siguiente sección, se detalla un algoritmo generado para poder superar esta limitación.

2.2. El algoritmo propuesto

A lo largo de esta sección estudiaremos el algoritmo propuesto en el *paper* analizado. Es importante destacar que el mismo busca ser fuerte en esos escenarios en los cuales LDA no es robusto. Una de las claves que explota el procedimiento es no usar todas las co-variables para clasificar. Manteniendo solamente las que distingan a las clases. Reduciendo el impacto negativo que suele tener trabajar en una elevada dimensión.

Primero, supondremos que Σ es conocida. Veremos cómo construimos el algoritmo. Desde el modelado de los datos, a cómo haremos para clasificar nuevas observaciones y, además, cómo seleccionar las variables a usar. Finalmente, abordaremos el caso donde Σ sea desconocida y veremos cómo se adapta el algoritmo.

2.2.1. Modelos

A continuación, damos una intuición del origen del algoritmo. Además, mostraremos los modelos que se utilizan para modelar los datos.

Como supusimos que los datos son vectores normales, para la observación i-ésima en la clase l, un modelo razonable es:

$$Y_{li} = m_l + \varepsilon_{li}, \tag{1}$$

donde $l \in \{1, \ldots, L\}$, $i \in \{1, \ldots, n_l\}$, el vector $m_l \in \mathbb{R}^p$ representa el la media de la clase l, y ε_{li} corresponde al error que asumiremos gaussiano independiente con distribución $\mathcal{N}(0_p, \Sigma)$. Recordemos que $\Sigma \in \mathbb{R}^{p\times p}$ se asume invertible y común a todas las clases, lo que constituye un supuesto fuerte del modelo.

Notemos que promediando las observaciones dentro de cada clase, obtenemos los *centroides*, cuyo modelo es:

$$\bar{Y}_l = m_l + \varepsilon_l^* \ \forall \ l \in \{1, ..., L\},$$

con $\varepsilon_l^* \sim \mathcal{N}(0_p, n_l^{-1}\Sigma)$ y $\bar{Y}_l = \frac{1}{n_l}\sum_{i=1}^{n_l} Y_{li}$.

Dada una nueva observación $Y_0 \in \mathbb{R}^p$, independiente a las n utilizadas para el entrenamiento, lo natural es asignar Y_0 a la clase l^* cuyo centroide $\bar{Y}_{l^*}$ sea el más cercano. Se podrían usar diversas nociones de distancia, en particular podemos elegir la de Mahalanobis. Asumamos momentáneamente a la matriz Σ como conocida. Notemos que la matriz de varianza de $Y_0 - \bar{Y}_l$ es $\rho_l^{-1}\Sigma$, donde $\rho_l = \frac{n_l}{n_l+1}$. En particular, nos queda:

$$l^* = \arg\min_{1\leq l\leq L} \rho_l (Y_0 - \bar{Y}_l)^\top \Sigma^{-1} (Y_0 - \bar{Y}_l). \tag{2}$$

Antes de continuar denotemos por $N = \sum_{l=1}^{L} n_l$, y $L_1 = L - 1$, y notemos que $\rho_l = \frac{n_l}{n_l+1} \in [\frac{1}{2}, 1)$.

Cabe destacar que esta idea está implícita en LDA. Si asumimos que tenemos la misma proporción de datos de cada clase, $\hat{\pi}_l$, podemos reescribir su regla de clasificación 10. Obteniendo así que asignamos una nueva observación Y_0 a la clase:

$$l^* = \arg \min_{1 \leq l \leq L} \delta_l(Y_0) = \arg \min_{1 \leq l \leq L} d_M^2(Y_0, \hat{\mu}_l)$$

Es decir, esta idea está presente en el planteo de un algoritmo que ya vimos. La diferencia es que cómo estamos pensando en un escenario específico, que es el de alta dimensión, nos surge una pregunta natural de si hacen falta tener todas las co-variables al generar los centroides o hay algunas que nos den más información que otras.

Buscamos una manera de medir qué variables son más importantes para discriminarlos que otras. Para ello, lo primero que hacemos es reescribir el modelo de medias en el marco del análisis multivariado de varianza (MANOVA) de un factor. Es decir,

$$\bar{Y}_l = \delta + \beta_l + \varepsilon_l^* \quad \forall\, l \in \{1, ..., L\}$$

donde estamos descomponiendo el centroide en:

- $\delta \in \mathbb{R}^p$ es el vector de efectos medios globales. Se calcula como el promedio de los $m_l : \ l \in \{1, \ldots, L\}$.
- $\beta_l \in \mathbb{R}^p$ representa los efectos específicos de la clase l.

Para este modelo es necesario imponer condiciones de identificabilidad, en particular, para cada característica j supondremos que

$$\sum_{l=1}^{L} \beta_{lj} = 0.$$

Con los elementos de este último modelado basado en MANOVA podemos generar la medida de importancia de una variable. Dada una variable j, queremos identificar si hay mucha o poca dispersión interclase. En particular, está relacionado con cuánto varían los centroides respecto de δ, el promedio de las medias de cada clase. Se representa con los coeficientes $\beta_{lj} : \ l \in \ \{1, \ldots, L\}$. De esta forma podemos considerar una medida global de la contribución de la variable j a la clasificación como:

$$b_j^2 = \sum_{l=1}^{L} \beta_{lj}^2 \tag{3}$$

Coloquialmente podemos decir que si una variable no varía significativamente entre clases, es decir, b_j^2 es pequeño y se parecen al promedio de los c, entonces tendrá poca influencia en la clasificación. Esta última observación es la base de la propuesta. El objetivo de la selección de variables consiste en identificar un subconjunto reducido y significativo de ellas que sean realmente discriminantes.

2.2.2. Regla de clasificación

En el apartado anterior, vimos una idea de cómo clasificar las nuevas observaciones y, también, los modelos que vamos a utilizar para poder detectar las variables significativas. Ahora, nos focalizaremos en entender mejor cómo es la regla de clasificación.

Para entender que es lo que el algoritmo realiza, nos concentraremos en un escenario teórico ideal en el que se conoce de antemano cuáles son las variables verdaderamente relevantes para la clasificación, es decir, aquellas para las cuales se cumple que $b_j^2 > 0$. En tal caso, lo más razonable sería utilizar únicamente esas variables informativas, lo que permitiría reducir de manera efectiva la dimensión del problema sin perder poder discriminativo.

Supongamos que las primeras p_1 variables corresponden a las significativa y las $p_0 = p - p_1$ variables restantes son irrelevantes. Consideremos $x_j = \mathbb{I}\{b_j^2 > 0\}$, las variables que indican si una variable es o no irrelevante, es decir $p_1 = \sum_{j=1}^{p} x_j$.

Bajo esta situación hipótetica, el procedimiento de clasificación dado en la ecuación (2) puede reescribirse considerando únicamente las variables relevantes, es decir:

$$\hat{l} = \arg \min_{1 \leq l \leq L} \left\{ \rho_l \left(Y_0^* - \bar{Y}_l^* \right)^\top (\Sigma^*)^{-1} \left(Y_0^* - \bar{Y}_l^* \right) \right\},$$

donde $Y_0^*, \bar{Y}_l^* \in \mathbb{R}^{p_1}$ corresponden solo a las variables importantes de Y_0 y $\bar{Y}_l$, es decir los vectores formados por las primeras p_1 coordenadas de Y_0 y $\bar{Y}_l$ respectivamente. Y la matriz $\Sigma^* \in \mathbb{R}^{p_1 \times p_1}$ es la submatriz principal superior izquierda de Σ correspondiente a las variables seleccionadas.

Por lo tanto si se conoce de antemano las variables relevantes para la clasificacción, el problema se reduce simplemente a clasificar con esas variables. Qué sucede entonces si no sabemos quienes son las variables importantes. La idea consiste en realizar primero una selección de variables a partir de testar la hipótesis de b_j es o no significativa para cada j y luego clasificar en base a esa selección.

2.2.3. Selección de variables

Hemos visto que, al tener las variables significativas elegidas, sabemos cómo vamos a clasificar. Además, estudiamos el modelo 3 a partir del cual podremos definir la medida de importancia de una co-variable.

Nos centramos ahora en el escenario donde no se conocen las variables significativas a utilizar en la regla de clasificación. Describiremos el proceso para poder identificarlas.

Comenzamos por separar el conjunto de los n datos de entrenamiento en dos. El primer conjunto de n_l^{SV} observaciones por categoría lo usaremos para hallar las variables importantes, mientras que los restantes n_l^C serán utilizados para calcular las distancias para poder clasificar nuevas observaciones según proximidad. Asumimos, sin pérdida de generalidad, que $n_l^{SV} = n_l^C$. Es decir, que partimos en partes iguales los datos disponibles de cada categoría.

Vamos a utilizar la dispersión de cada una de las variables respecto de los L centroides. La idea es semejante a lo que sucede en ANOVA, vamos a analizar la variabilidad entre los grupos. Para cada variable j de las p posibles, haremos un test de hipótesis. Más precisamente, el procedimiento de inferencia para corroborar la importancia de una variable j consiste en testear si:

$$H_0 : b_j^2 = 0 \quad \text{vs} \quad H_1 : b_j^2 > 0.$$

donde $b_j^2 = \sum_{l=1}^{L} \beta_{l,j}^2$, al igual que como lo definimos en 3. Un estadístico natural para este problema es considerar

$$\zeta_j = \sigma_j^{-2} \sum_{l=1}^{L} n_l^{SV} \left(\bar{Y}_{lj}^{SV} - \bar{Y}_{\cdot j}^{SV} \right)^2 \tag{4}$$

donde $\sigma_j^2 = \Sigma_{jj}^{SV}$ y $\bar{Y}_{\cdot j}^{SV} = \frac{1}{N^{sv}} \sum_{l=1}^{L} n_l^{SV} \bar{Y}_{lj}^{SV}$.

Bajo la hipótesis nula, $\zeta_j \sim \chi_{L_1}^2$, mientras que bajo la alternativa $\zeta_j \sim \chi_{L_1}^2(\mu_j)$, donde $\chi_{L_1}^2(\mu_j)$ es la distribución chi-cuadrado no central con parámetro de no-centralidad $\mu_j = \sigma_j^{-2} \sum_{l=1}^{L} n_l^{SV} \beta_{lj}^2$.

Notar que en general los ζ_j estarán correlacionados salvo el caso particular que Σ^{SV} sea diagonal.

Para obtener el punto de corte, los autores referencian a la siguiente propiedad de una variable chi-cuadrado. Sea $\zeta \sim \chi_k^2(\mu)$, con $\mu \geq 0$. Entonces, para cualquier $x > 0$, se tienen dos propiedades:

$$\Pr\left(\zeta > \mu + k + 2\sqrt{(k + 2\mu)x} + 2x \right) \leq e^{-x}$$

$$\Pr\left(\zeta < \mu + k - 2\sqrt{(k + 2\mu)x} \right) \leq e^{-x}.$$

En particular, en la primera si $x = \ln \frac{2p}{\alpha}$, $k = L_1$ y $\mu = 0$, se tiene que:

$$\Pr\left(\zeta > L_1 + 2\sqrt{L_1 \ln \frac{2p}{\alpha}} + 2 \ln \frac{2p}{\alpha} \right) \leq \frac{\alpha}{2p}$$

Luego, dado $0 < \alpha \leq 1$ podemos utilizar el umbral

$$\lambda = L_1 + 2\sqrt{L_1 \ln \left(\frac{2p}{\alpha} \right)} + 2 \ln \left(\frac{2p}{\alpha} \right). \tag{5}$$

De este modo para cada variable j, se la seleccione como significativa (es decir, se rechaza H_{0j}) si

$$\zeta_j > \lambda.$$

Y el test de hipótesis tendrá nivel $\frac{\alpha}{2p}$.

Haremos esta comparación p veces, eligiendo cada una de las variables que ayuden a discriminar los centroides. El nivel del test ha sido elegido para que la probabilidad de elegir bien el conjunto de dimensiones significativas sea mayor a $1-\alpha$, condicionado a que los centroides estén lo suficientemente separadas. Para más detalles, ver la sección de contribuciones teóricas de este Capítulo.

2.2.4. El paso a paso con Σ conocida

En los anteriores segmentos se explica cómo modelar y calcular los centroides, y cómo identificar variables significativas. Recordemos que hasta acá suponíamos que la covarianza Σ es conocida. Dada una observación Y_0, el algoritmo de clasificación propuesto para asignarla es:

Paso 1: Dividir los datos de entrenamiento: dada una categoría l, debemos elegir qué datos utilizaremos para la selección de variables $Y_{lj}^{SV}, j \in \{1,\dots,n_l^{SV}\}$ y cuáles para el cálculo de las distancias a los centroides para poder clasificar $Y_{lj}^{C}, j \in \{1,\dots,n_l^{C}\}$. Los dividimos en partes iguales, es decir, $n_l^{SV} = n_l^C \ \forall\ l \in \{1,\dots,L\}$.

Paso 2: Identificar las variables significativas. A partir de los datos $Y_{lj}^{SV} \ : \ l \in \{1,\dots,L\}\ ; j \in \{1,\dots,n_l^{SV}\}$ seleccionamos las variables importantes utilizando un test de hipótesis para cada dimensión. Notemos por $\hat{p}_1$ el número de variables importantes halladas por los tests.

Paso 3: Entrenamiento del algoritmo. Encontrar los L centroides $\overline{Y}_l^C$. Luego, truncamos los centroides utilizando solamente las $\hat{p}_1$ variables importantes. Los denominamos $\overline{Y}_l^{C*}$. Se tiene que $\overline{Y}_l^{C*} \in R^{\hat{p}_1}\ \forall\ l:\ 1 \le l \le L$.

Paso 4: Clasificar la nueva observación. Para poder comparar Y_0 con los centroides truncados, seleccionamos sus $\hat{p}_1$ variables importantes y lo denominamos Y_0^*. Asignaremos el nuevo dato a la categoría l^* que cumpla que su centroide truncado sea el más cercano. Es decir, que minimice la distancia de Mahalanobis. Se calcula como,

$$l^* = arg \min_{1\le l\le L} \left\{ \rho_l^C \ (Y_0^* - \overline{Y}_l^{C*})^T \ (\Sigma^{C*})^{-1} \ (Y_0^* - \overline{Y}_l^{C*}) \right\} \tag{6}$$

donde Σ^{C*} es la matriz de covarianza restringida a las $\hat{p}_1$ variables importantes.

Como ya habíamos mencionado, en el Paso 1 la se podría haber tomado otra proporción de datos para definir las muestras de selección de variables y cálculo de centroides arbitraria. Simplemente tomamos la mitad a modo de simplificación.

Si ninguna variable fuese elegida como importante, entonces el algoritmo falla y la nueva observación Y_0 debe ser atribuida a una clase al azar. Se asumo que se está trabajando en dimensión elevada, en donde utilizar la distancia de Mahalanobis con todas las variables podría implicar problemas numéricos.

2.2.5. Σ desconocido.

Previamente, describimos el funcionamiento del algoritmo en el caso en el cual la varianza Σ (que es común para todas las categorías) es conocida. A continuación, abordaremos el caso en el que la misma sea desconocida. Nuestro objetivo es establecer un procedimiento para poder clasificar una nueva observación Y_0.

Para comenzar, exponemos cómo vamos a aproximar dicha varianza. Recordemos que trabajamos con dos conjuntos de datos de entrenamiento diferentes, uno para la selección de variables y otro para calcular los centroides utilizados para clasificar. Luego, vamos a generar una aproximación para cada uno:

$$\hat{\Sigma}^{SV} = \frac{1}{N^{SV}} \sum_{l=1}^{L} \sum_{i=1}^{n_l^{SV}} (Y_{il} - \overline{Y_l})(Y_{il} - \overline{Y_l})^T$$

$$\hat{\Sigma}^{C} = \frac{1}{N^{C}} \sum_{l=1}^{L} \sum_{i=1}^{n_l^{C}} (Y_{il} - \overline{Y_l})(Y_{il} - \overline{Y_l})^T$$

donde N^{SV} y N^C son la totalidad de los datos de cada subconjunto, es decir, $\sum_{l=1}^{L} n_l^{SV}$ y $\sum_{l=1}^{L} n_l^C$, respectivamente. Notemos que la estimación descrita está fuertemente basada en el supuesto que las matrices de covarianzas de cada categoría son iguales.

Para continuar, observemos cómo se adaptan los pasos del algoritmo. Los únicos que sufren modificaciones son el segundo, en donde se eligen las variables significativas y el cuarto, donde se elige a que categoría asignar a la nueva observación. Estos mismos son aquellos en donde se utiliza la matriz de covarianza Σ.

Selección de variables:

El objetivo es adaptar el método de selección de variables para el caso en que la varianza sea desconocida. Las modificaciones que se realizan en cada test de hipótesis a realizar en el segundo paso del algoritmo son:

- Modificaremos el estadístico del test de hipótesis definido en 4. Donde previamente usaba la matriz de covarianza, ahora usará su versión aproximada:

$$\hat{\zeta}_j = \frac{1}{\hat{\sigma^{SV}}_j^2} \sum_{l=1}^{L} n_l^{SV} \left(\bar{Y}_{lj}^{SV} - \bar{Y}_{\cdot j}^{SV} \right)^2 \quad \forall\, j \in \{1, \ldots, p\} \tag{7}$$

  donde $\hat{\sigma}_j^2 = \hat{\Sigma_{jj}^{sv}}$ y $\bar{Y}_{\cdot j}^{SV} = \frac{1}{N^{sv}} \sum_{l=1}^{L} n_l^{SV} \bar{Y}_{lj}^{SV}$.

- Por otro lado, consecuencia de modificar el estadístico se debe también modificar el valor crítico.

Para poder rechazar el test de hipótesis e identificar a la variable j como discriminante, el valor crítico propuesto λ_1 será mayor al λ definido en (5):

$$\lambda_1 = \frac{\lambda}{1-\kappa} \tag{8}$$

con $\kappa = 2\sqrt{\frac{\ln\frac{2p}{\alpha}}{N_1 - L}} + 2\,\frac{\ln\frac{2p}{\alpha}}{N_1 - L}$.

Notemos que el nivel estará asociado con la dimesión, más precisamente pediremos que la dimensión del problema cumpla con $p \leq \frac{\alpha}{2}\mathrm{e}^{\left(\frac{N_1-L}{4}\right)}$.

Este valor crítico se elige de manera tal que, si si se cumplen suspuestos de variabilidad de las dimensiones, la probalidad de elegir correctamente las variables importantes correctas sea mayor a $1-2\alpha$. Para más detalles, ver la sección de este capítulo donde se detalla la teoría.

En resumen, el test de hipótesis para determinar la importancia de una variable j es:

$$H_0 : b_j^2 = 0 \quad \text{vs} \quad H_1 : b_j^2 > 0.$$

Y lo aceptaremos si el estadístico ζ (7) supera el umbral λ_1 (8).

En la sección siguiente, se verán las condiciones necesarias para que esta adaptación las seleccione de manera correcta con una alta probabilidad.

Clasificar la nueva observación:

Para clasificar una nueva instancia Y_0, la asignamos a la clase l^* que tenga su centroide más cerca de Y_0. En particular, para calcular la distancia de Mahalanobis se utiliza la matriz de covarianza de $(Y_0^* - \bar{Y}_l^{C*})$, que es $\rho_l\ \Sigma^C$.

Como desconocemos el valor de Σ^C, basta reemplazarlo por su aproximación en 6. Y entonces, l^* será:

$$l^* = arg\ \min_{1\leq l\leq L}\left\{\rho_l^C\ (Y_0^* - \overline{Y}_l^{C*})^T\ (\hat{\Sigma}^{C*})^{-1}\ (Y_0^* - \overline{Y}_l^{C*})\right\}$$

El paso a paso:

Finalmente, si agregamos las modificaciones previamente descritas, obtenemos que el algoritmo para clasificar una nueva observación Y_0, cuando la matriz de covarianza Σ es desconocida, es:

Paso 1: Dividir los datos de entrenamiento: dada una categoría l, debemos elegir qué datos utilizaremos para la selección de variables $Y_{lj}^{SV}, j \in \{1,\ldots,n_l^{SV}\}$ y cuáles para el cálculo de las distancias a los centroides para poder clasificar $Y_{lj}^{C}, j \in \{1,\ldots,n_l^{C}\}$. Los dividimos en partes iguales, es decir, $n_l^{SV} = n_l^C\ \forall\ l \in \{1,\ldots,L\}$

Paso 2: Identificar las variables significativas. A partir de los datos $Y_{lj}^{SV} \ : \ l \in \{1, \ldots, L\} \, ; j \in \{1, \ldots, n_l^{SV}\}$ seleccionamos las variables importantes utilizando la variante de los tests de hipótesis donde la matriz es desconocida. Notemos por $\hat{p}_1$ el número de variables importantes halladas.

Paso 3: Entrenamiento del algoritmo. Encontrar los L centroides $\overline{Y}_l^C$. Luego, truncamos los centroides utilizando solamente las $\hat{p}_1$ variables importantes. Los denominamos $\overline{Y}_l^{C*}$. Se tiene que $\overline{Y}_l^{C*} \in R^{\hat{p}_1} \ \forall \ l: \ 1 \leq l \leq L$.

Paso 4: Clasificar la nueva observación. Para poder comparar Y_0 con los centroides truncados, seleccionamos sus $\hat{p}_1$ variables importantes y lo denominamos Y_0^*. Asignaremos el nuevo dato a la categoría l^* que cumpla que su centroide truncado sea el más cercano. Es decir, que minimice la distancia de Mahalanobis. Se calcula como,

$$l^* = arg \ \min_{1 \leq l \leq L} \left\{ \rho_l^C \ (Y_0^* - \overline{Y}_l^{C*})^T \ (\Sigma^{\hat{C}*})^{-1} \ (Y_0^* - \overline{Y}_l^{C*}) \right\} \tag{9}$$

donde $\Sigma^{\hat{C}*}$ es la matriz de covarianza estimada, restringida a las $\hat{p}_1$ variables importantes.

Si ninguna variable fuese elegida como importante, entonces el algoritmo falla y la nueva observación Y_0 debe ser atribuida a una clase al azar. Se asumo que se está trabajando en dimensión elevada, en donde utilizar la distancia de Mahalanobis con todas las variables podría implicar problemas numéricos.

2.3. Contribuciones teóricas

Uno de los aportes en los que se destaca el trabajo de [1] es que obtienen condiciones explícitas que garantizan que el aumento del número de clases puede mejorar la exactitud de la clasificación. Recordemos que este es el contexto para el cual queremos que el algoritmo supere las limitaciones de LDA. Este resultado, que a primera vista puede parecer contra intuitivo, se explica por una selección de variables más precisa que en principio cuando estamos en un contexto de pocas clases las variables que no contribuyen a la clasificación podrían pasar desapercibidas.

Para el caso donde la varianza Σ es conocida, si hay cierta separación de los centroides, se puede probar que el procedimiento para seleccionar las variables funciona. En el sentido que elige las variables significativas con probabilidad mayor a $1 - \alpha$. Más precisamente,

Teorema 2.3.1. Sea p_1 el número de variables importantes y $\hat{p}_1$ su estimación obtenida por el procedimiento descrito en la Sección 2.2.4 con $0 < \alpha < 1$. Si

$$\mu_* = \min_{1 \leq j \leq p_1} \sigma_j^{-2} \sum_{l=1}^{L} n_l^{SV} \beta_{lj}^2 \geq 4 \left(3 \ln \frac{2p}{\alpha} + \sqrt{L_1 \ln \frac{2p}{\alpha}} \right).$$

Entonces

$$Pr(\hat{p_1} = p_1) \geq 1 - \alpha$$

Un resultado similar obtienen para el caso en que la matriz de varianzas es desconocida.

Teorema 2.3.2. Sea p_1 el número de variables importantes y $\hat{p}_1$ su estimación obtenida por el procedimiento descrito en la Sección 2.2.5 con $0 < \alpha \leq \frac{1}{2}$ y supongamos que $p \leq \frac{\alpha}{2} e^{\left(\frac{N_1 - L}{4}\right)}$. Si

$$\mu_* + \sqrt{(L_1 + 2\mu^*) \ln \frac{2p}{\alpha}} \geq \lambda_1(1 + \kappa).$$

Entonces

$$Pr(\hat{p_1} = p_1) \geq 1 - 2\alpha.$$

Otro resultado que presenta el paper en estudio, esta relacionado con la bondad de la clasificación post selección de variables. Consideremos la regla de clasificación aplicada al segundo conjunto de datos, utilizando $\bar{Y}_l^{C*}$, donde las verdaderas p_1 variables importantes desconocidas son reemplazados por las $\hat{p}_1$ estimadas siguiendo el procedimiento propuesto de selección de variables. Ademas podemos suponer que ordenamos las variables de modo que las $\hat{p}_1$ variables seleccionadas como significativas sean las primeras. Entonces la regla de clasificación resultante se puede escribir como:

$$l^* = \arg \min_{1 \leq l \leq L} \left\{ \rho_l \left(Y_0^* - \bar{Y}_l^{C*} \right)^\top (\Sigma^*)^{-1} \left(Y_0^* - \bar{Y}_l^{C*} \right) \right\}, \tag{10}$$

donde los vectores truncados $Y_0^*, \bar{Y}_l^{C*} \in \mathbb{R}^{\hat{p}_1}$, para $l \in \{1, \ldots, L\}$, están definidos como $Y_{0j}^* = Y_{0j}$, $\bar{Y}_{lj}^{C*} = \bar{Y}_{lj}^C$, para $j \in \{1, \ldots, \hat{p}_1\}$, y $\Sigma^* \in \mathbb{R}^{\hat{p}_1 \times \hat{p}_1}$ es la submatriz superior izquierda correspondiente de Σ o a $\hat{\Sigma}$ según se conozca o no y $\rho_l = \frac{n_l^C}{n_l^C + 1}$.

Se tiene que

$$\Pr(\hat{l} \neq l) \leq \Pr(\hat{l} \neq l \mid \hat{p}_1 = p_1) + \Pr(\hat{p}_1 \neq p_1), \tag{11}$$

donde, debido a que se utilizan conjuntos de datos distintos para la selección de variables y para la clasificación, por los Teoremas 2.3.1 y 2.3.2, se puede acotar la $P(\hat{p}_1 = p_1)$. Por lo tanto, se obtiene el siguiente resultado:

Teorema 2.3.3. Sea $0 < \alpha \leq \frac{1}{2}$ y sean m_k^* los centroides truncados luego de la selección de variables. Si se satisfacen las hipótesis del 2.3.1 (para que la selección de variable sea precisa) y los centroides están separados:

$$d_M(m_k^*, m_{k'}^*) \geq \frac{8 \ln \frac{L_1}{\alpha}}{\min \rho_k, \rho_{k'}} \left\{ 1 + \frac{1}{\sqrt{2 \min (n_k^C, n_{k'}^C)}} \left(1 + \sqrt{\frac{2p_1}{\ln \frac{L_1}{\alpha}}} \right) \right\}.$$

Sea Y_0 una nueva observación proveniente de la clase l, y sea l^* la clase asignada a Y_0 de acuerdo con la regla de clasificación. Entonces, el error de clasificación está acotado por:

$$P(l^* = l) \geq 1 - 2\alpha.$$

Finalmente un resultado similar se puede obtener en el caso de matriz de varianza desconocida.

Teorema 2.3.4. Dada Y_0 una nueva observación proveniente de la clase l. La clasificamos a l^* mediante el algoritmo descrito en 2.2.5. Para poder tener certeza de la precisión del resultado, necesitamos pedir que la selección de variables la selección de variables también lo sea. Y entonces, pedimos que se cumplan las hipótesis de 2.3.2.

Además, pedimos que los centroides estén separados. Para ello, pedimos que

$$\text{máx}\left\{L,\, 2\ln\left(\frac{2}{\alpha}\right)\right\} < p_1 < \frac{1}{4C_1}\left(\frac{\lambda_{\text{mín}}(\Sigma^{C*})}{\lambda_{\text{máx}}(\Sigma^{C*})}\right)^4 N^C$$

para algún $0 < \alpha < \frac{1}{4}$ y donde C_1 es una constante. Luego, denotamos

$$\gamma_{p_1,N_2} = 2\frac{\lambda^2_{\text{máx}}(\Sigma^{C*})}{\lambda^2_{\text{mín}}(\Sigma^{C*})}\sqrt{\frac{C_1 p_1}{N^C}}.$$

Por la condición anterior, sabemos que $\gamma_{p_1,N_2} < 1$. Luego, la distancia que les pedimos a los centroides es:

$$\begin{aligned} d_M(m_k^*, m_{k'}^*) = (m_k^* - m_{k'}^*)^\top (\Sigma^*)^{-1}(m_k^* - m_{k'}^*) \quad &\geq \quad \frac{8\ln(L_1/\alpha)}{(1-\gamma_{p_1,N^C})\,\text{mín}(\rho_k, \rho_{k'})} \\ &\times \quad \left\{1 + \sqrt{2\,\text{mín}(n_k^C, n_{k'}^C)} + \gamma^2_{p_1,N^C}\left(1 + \sqrt{\frac{2p_1}{\ln(L_1/\alpha)}}\right)\right\}. \end{aligned}$$

Entonces, el error de clasificación está acotado por:

$$P(l^* = l) \geq 1 - 4\alpha$$

En conclusión, hemos estudiado el algoritmo propuesto. El mismo tiene garantías de precisión inclusive cuando se esté estimando la varianza (escenario más realista). En el caso que los centroides estén lo suficientemente separados, elige bien tanto las variables importantes para la clasificación, como la categoría.

Capítulo 3

Simulaciones y comparaciones

En este capítulo estudiaremos a partir de simulación el algoritmo descrito previamente. El objetivo es, por un lado, entender en qué circunstancias tiene un buen desempeño y en cuáles encuentra problemas. Además, veremos cómo afecta a la calidad del resultado el aumentar la cantidad de categorías y su comparación con sus competidores.

El paper que motiva esta tesis especifica que no es el mejor algoritmo. Pero si dice que es el primero para el cuál se tienen certezas teóricas ante el aumento de cantidad de clases. La pregunta a responder es si podrá consistentemente ganarle a LDA y a SVM.

3.1. Generación de Datos

En primer lugar, debemos comenzar por establecer cómo vamos a generar los datos en los que vamos a evaluar los algoritmos. Utilizaremos el escenario descripto en [1]. Es decir, consideraremos

$$Y_{li} = m_l + \varepsilon_{li}$$

con $1 \leq l \leq L$ y $1 \leq i \leq n_l$ y $Y_{li} \in R^p$. Es decir, nos centraremos en la clasificación de vectores normales de alta dimensión $p = 500$, donde solamente p_1 de las variables distinguirán a las categorías. Las otras serán ruido. Además, supondremos que tenemos pocos datos por clase $n_l = 20 \; \forall \; l \in \{1, \ldots, L\}$. Para poder simular los datos, necesitamos la media y la varianza de cada grupo.

Al igual que propone [1] generaremos las medias segun el siguiente esquema. Para cada $l \in \{1, \ldots, L\}$, generamos las medias de las clases como vectores normales i.i.d. $m_l \sim \mathcal{N}(0, \sigma_m^2 X)$, con $l \in \{1, \ldots, L\}$, donde σ_m gobierna la dispersión de las variables que distinguirán a las categorías, y $X \in \mathbb{R}^{pxp}$, una matriz diagonal con los primeros p_1 valores iguales a 1,

$$X = \begin{bmatrix} \mathrm{Id}_{p_1} & 0 \\ 0 & 0 \end{bmatrix}$$

Para reducir el impacto de una elección particular de los vectores m_l, generamos M_1 réplicas de las medias de las clases.

La matriz de varianza Σ, al igual que cuando vimos la motivación del algoritmo, vamos a suponer que es idéntica para todas las categorías y distinguiremos 3 posibilidades:

- Independientes:
$$\Sigma = \sigma^2 Id_p$$
- Autorregresivas:
$$\Sigma_{i,j} = \sigma^2 \ 0,5^{|i-j|}$$
- Correlacionadas:
$$\Sigma_{i,j} = \sigma^2 \ (0,5 + 0,5 \ Id_{i=j})$$

Notemos que son simétricas y además, se puede apreciar que en los tres casos las varianzas de todas las dimensiones serán σ^2. Cabe destacar que esto vale tanto para las variables que sean o no importantes. Utilizaremos $\sigma^2 = 1$. Para decidir la varianza σ_m utilizamos el ratio t:
$$t = \frac{\sigma_m^2}{\frac{\sigma^2}{n_l/2}}$$
que vincula la dispersión de las medias con la de los centroides utilizados en el algoritmo (en cada paso usamos la mitad de los datos, por eso ese $\frac{n_l}{2}$).

Si t es pequeño, implica que las categorías no están lo suficientemente separadas y se mezclarán los puntos de las diferentes distribuciones. Mientras que si t es grande, habrá menos solapamiento.

Ahora que ya tenemos la distribución de cada una de las categorías, obtenemos dos grupos de datos. Al primero, lo usaremos para entrenar los algoritmos, mientras que el segundo será el grupo de control.

El grupo para entrenamiento constará de $n_l = 20$ datos por categoría. Mientras que el segundo, serán $M_3 = 50$ datos cuyas clases serán elegidos al azar.

En verdad, obtenidas las medias de cada categoría, las M_2 veces que generemos los n_l datos no serán 100 % independientes. Sino que producimos un conjunto muy grande (1.000 datos de cada categoría), y obtenemos al azar n_l datos M_2 veces. Si bien no son independientes, el conjunto es tan grande que prácticamente lo serán. Análogamente, obtenemos 1.000 datos de control, de los cuales elegiremos $M_3 = 50$ al azar para cada conjunto de datos que generemos.

En resumen, para realizar un experimento, primero elegimos p_1 variables significativas, t la relación de dispersión entre las medias y los datos de las mismas, L la cantidad de categorías y el tipo de covarianza. Luego, generamos $M_1 * M_2$ conjuntos de datos de entrenamiento y para cada uno de ellos, los M_3 datos para evaluar la precisión del modelo.

3.2. Análisis algoritmo paper

En esta sección analizaremos los resultados provenientes del algoritmo. Como ya mencionamos previamente, consta de cuatro pasos. Hay dos de ellos en los cuales podrían llegar a ocasionarse inconvenientes. Son la selección de variables y la clasificación de nuevas instancias.

El objetivo es validar si hay circunstancias en las cuales no funciona correctamente el algoritmo. Y encontrar las que si. Además, analizaremos la convergencia por aumento de cantidad de categorías en acción.

3.2.1. Análisis selección de variables

Recordemos que la selección de variables discriminantes se da a través de realizar un test de hipótesis para cada una de las dimensiones. En cada uno se evalúa la dispersión de los centroides. Dicho test está compuesto por el estadístico utilizado y el valor crítico a partir del cual se considera una variable como importante.

En los resultados obtenidos en el paper estudiado, se concluye que el valor crítico utilizado es muy elevado. Y, en consecuencia, no genera falsos positivos(variables que son ruido consideradas como importantes) pero, tampoco detecta todas las importantes. Primero, veremos qué consecuencias trae a la clasificación cuando hay pocas dimensiones importantes. Luego, estudiaremos cómo evoluciona el valor crítico al agregar más categorías y como se reflejó eso en los resultados de las simulaciones.

Estudiemos qué sucede con la clasificación para los casos en donde una variable significativa no supere el umbral $\lambda(5)$ o λ_1 (8), según corresponda. En ese caso, no se rechazará el test de hipótesis y no será utilizada para clasificar nuevas observaciones.

A modo ilustrativo, simulamos el comportamiento en dos dimensiones discriminantes. En las diferentes filas, podemos observar cómo lucen las regiones de clasificación con distintas cantidades de clases. Tenemos las regiones para el caso en el que detectamos una sola o las dos dimensiones como significativas. Si bien el algoritmo no fue pensado para abordar problemáticas en dimensiones bajas, ilustra el potencial problema en el que se puede caer si hay pocas dimensiones significativas.

Analicemos los resultado de la figura 3.1. El mismo muestra la pérdida de generalidad. Cuantas más categorías consideramos, peor será.

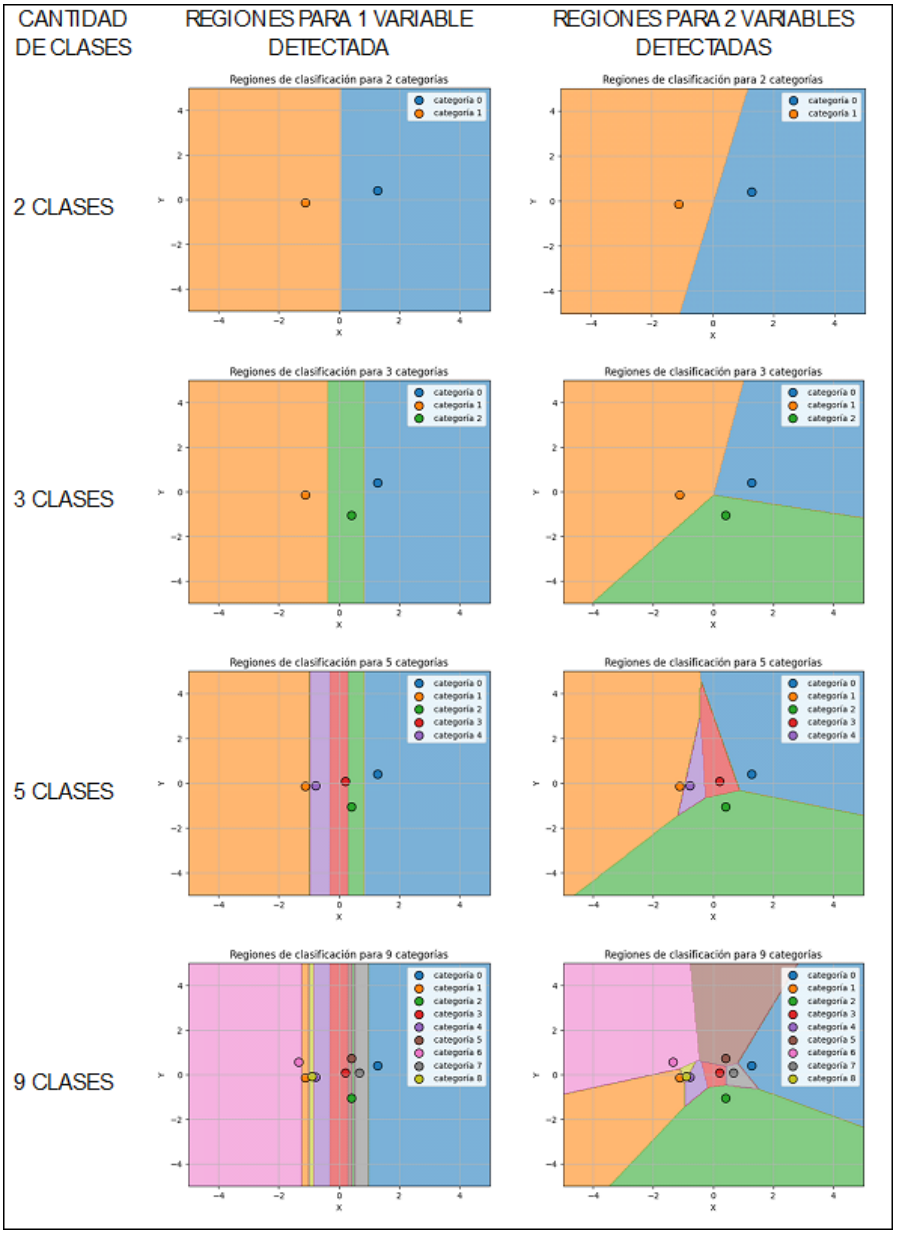


Figura 3.1: Consecuencias de la detección de variables

En general, el impacto en la capacidad de generalizar debido a no identificar correctamente las variables dependerá fundamentalmente del porcentaje de ellas que nos perdemos. Es decir, de la relación entre $\hat{p_1}$ y p_1. Cuanto más dimensiones significativas haya, menor va a ser el impacto.

Supongamos que a partir de ahora trabajamos en dimensión $p = 500$.

Analicemos el impacto de aumentar la cantidad de categorías en el valor crítico. El

mismo depende de

- la cantidad de categorías, L
- la dimensión del espacio de co-variables, p (en este caso $p = 500$)
- el valor α que regula la probabilidad de identificar correctamente el grupo de variables que realmente distingan a las clases

Veamos qué sucede cuando $\alpha = 0,05$ y aumentamos la cantidad de variables:

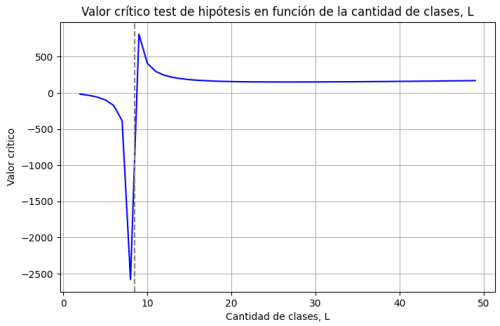


Figura 3.2: Estudio valores críticos

Como se puede apreciar:

- si $L < 9$: el valor crítico queda negativo, dando una imposibilidad de utilizar el algoritmo
- si $9 \leq L < 28$: va decreciendo y, por ende, nos estamos viendo beneficiados por agregar más categorías y estimar mejor la dispersión.
- si $28 \leq L$: comienza lentamente a crecer el valor crítico. Lo hace de manera tan paulatina que se va a ver beneficiado de las nuevas clases.

$L = 9$ será la mínima cantidad de clases para que el algoritmo funcione.

A continuación se muestran los resultados obtenidos en la simulación del siguiente apartado. Aquí podremos observar como evoluciona la cantidad de variables detectadas al ir alejando las medias de las clases, es decir, aumentando t. De las $p = 500$ variables del experimento, solo $p_1 = 10$ serán importantes. Hay 3 resultados según la cantidad de categorías $L \in \{10, 20, 50\}$

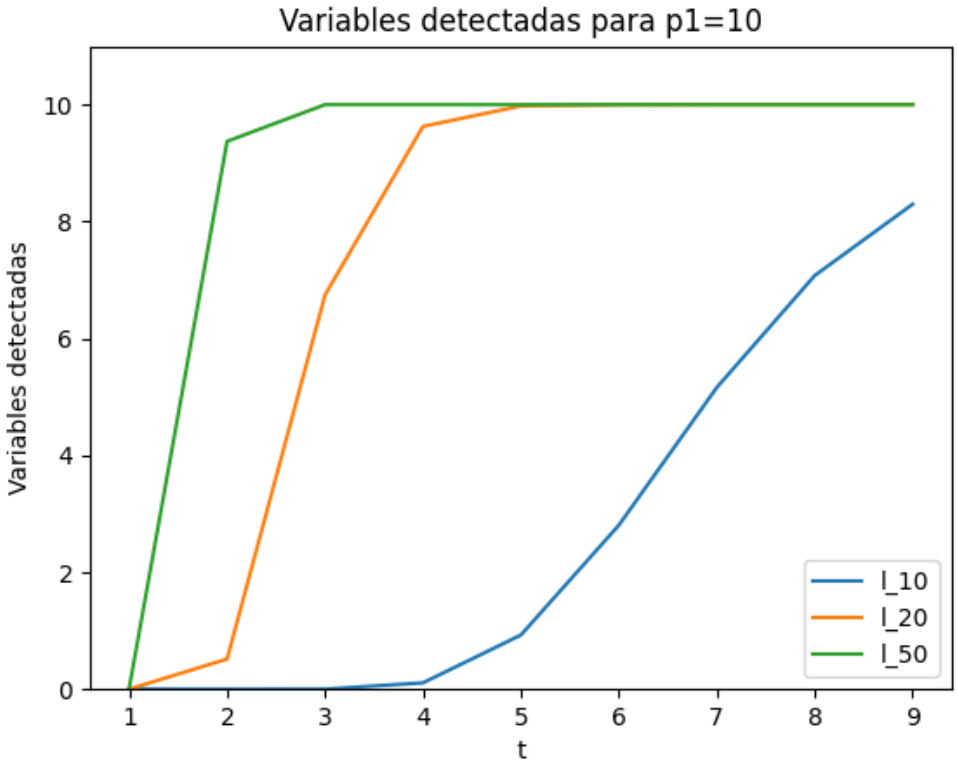


Figura 3.3: Evolución de variables detectadas en función de t para $p_1 = 10$
Resultados obtenidos de las simulaciones del apartado de comparación con otros algoritmos

Si observamos estos resultados, entonces el universo en donde hay muy pocas variables importantes detectadas se vuelve realista. Y se puede caer en los problemas que vimos.

Efectivamente $L = 10$, al que se le exige mucha evidencia, no logra detectarlas. Se puede apreciar como aumentar la cantidad de categorías beneficia en la detección de variables importantes.

3.2.2. Análisis asignación de nuevas observaciones

En esta sección analizaremos el comportamiento del algoritmo respecto de la asignación de nuevas observaciones. Tal como vimos en el capítulo anterior, para asignar una nueva observación Y_0 a una clase, primero seleccionamos sus $\hat{p_1}$ dimensiones importantes y luego la asignamos a la categoría cuyo centroide minimice la distancia de Mahalanobis.

Para poder calcular la distancia de Mahalanobis se invierte la matriz $\Sigma^{\hat{C}*}$, que es la matriz de covarianza aproximada restringida a las $\hat{p_1}$ variables detectadas como importantes. Como esta matriz puede tener una dimensión elevada, en el caso que haya

muchas variables catalogadas como importantes, puede caer en problemas numéricos.

A continuación, se dan los resultados producto de simular a partir de:

- $p = 500$ dimensiones existentes de las cuales solo $p_1 = 200$ distinguen a las variables.
- $L = 50$ clases entre las que clasificar con $n_l = 20$ datos de entrenamiento por categoría
- 600 simulaciones donde se generan $M_1 = 20$ veces las medias de cada categoría y se toman $M_2 = 30$ veces los datos de cada categoría. La matriz de covarianza común de todas las clases es la opción que llamamos correlacionada.

En particular, en donde suceda $\hat{p_1} = p_1$ y entonces, se detecten las 200 variables discriminantes, vamos a tener una matriz cuyo número de condición no es el ideal:

t	Promedio de Variables Detectadas	Promedio de Número de condición	Promedio de Exactitud
1.5	67	195	0.97
2	188	1495	1
2.5	200	1715	1
3	200	1722	1

Tabla 3.1: Análisis asignación de nuevas observaciones

Sin embargo, el algoritmo sigue detectando de manera correcta el centroide más cercano en el conjunto de evaluación, que no fue utilizado para decidir que variables significativas se usarían. Se concluye entonces, si bien hay que tener en cuenta que podría ocasionar problemas, no se encontró escenarios en los cuales si los generara.

3.2.3. Aumentando Categorías

Una característica que destacamos en el capítulo anterior cuando describimos el funcionamiento del algoritmo es que se ve beneficiado por aumentar la cantidad de categorías. La ventaja es que logra detectar más sencillamente a las variables que distinguen las clases. A diferencia de LDA, que no logra sacar ventaja de esta circunstancia.

En el siguiente experimento, comparamos la exactitud de ambos modelos cuando el número de grupos se incrementa. Sus parámetros son :

- empezaremos con $L = 20$ clases, iremos agregando de a una hasta llegar a 50. Asumiremos el modelo con covarianza y $n_l = 20$ para $1 \leq l \leq 20$.
- Vamos a simular 2500 veces cada caso($M1 = 50$, $M2 = 50$), evaluando cada uno de ellos en $M3 = 50$ observaciones

- Los datos serán generados con $p_1 = 10$ variables importantes que distingan las categorías de las $p = 500$ posibles, con cada clase con varianza independiente y la relación entre la dispersión intra e inter clase $t = 2$

Los resultados son:

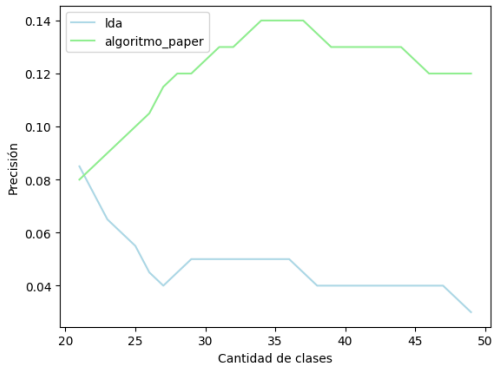


Figura 3.4: Experimento: Aumentar la cantidad de categorías

Primero, se puede destacar que el algoritmo logra vencer a LDA de manera consistente en términos de exactitud. Además, el algoritmo mejora a medida que hay más clases, el siguiente gráfico veamos como aumenta cantidad de variables detectadas en función de la cantidad de categorías.

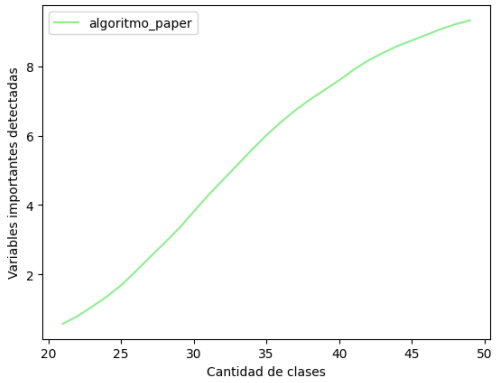


Figura 3.5: Aumento de variables detectadas al aumentar la cantidad de clases

Efectivamente, aumenta la detección de variables que distinguen las clases. Esto se debe a, por un lado, una mejor estimación de la covarianza. Y, por otro, a una mejor medida de dispersión utilizada en el test de hipótesis y su correspondiente menor valor crítico. Pero, es de esperar que si ya se estuviesen detectando todas las importantes, la mejora sea inferior a la vista.

3.3. Comparación entre algoritmos

En este segmento simularemos diferentes escenarios en los cuales compararemos la exactitud del algoritmo propuesto en comparación con LDA y SVM. LDA fue elegido por ser nuestro baseline a superar. Mientras que la elección de SVM, es por su buen funcionamiento en dimensiones altas.

3.3.1. Casos a analizar

Los escenarios elegidos para analizar replican los utilizados en el trabajo de [1]. Vamos a trabajar en dimensión $p = 500$. De las mismas, elegimos a $p_1 \in \{10, 50, 100, 200\}$ como importantes. Esto nos permitirá ver en cuáles casos el algoritmo es mejor, si cuando hay más o menos ruido.

Fijado p_1, elegiremos la cantidad de categorías $L \in \{10, 20, 50\}$. Recordemos que vimos que al menos debe haber 9 categorías para que el algoritmo funcione. Aquí el objetivo será ver cuántas categorías son necesarias para que el rendimiento se asemeje o supere al de los otros dos algoritmos.

Para cada p_1 y L, estudiaremos diferentes ratios t entre la dispersión de las medias de las clases y la dispersión de cada una de las clases. A medida que aumentemos el t, esperemos que la exactitud aumente.

Por último, supondremos las 3 posibilidades de covarianza explicadas al comienzo del capítulo, (independiente, correlacionada y autorregresiva).

Cada uno de los experimentos los repetiremos 600 veces ($M_1 = 20$ y $M_2 = 30$).

Cuando ejecutemos SVM, usaremos validación cruzada para elegir los hiperparámetros con 3 grupos.

3.3.2. Análisis de resultados

El primer efecto que queremos estudiar es el que respecta a la cantidad de variables importantes. Para ello, analicemos cómo impactaron cuando había $L = 50$ clases. En los siguientes gráficos, podemos apreciar la exactitud promedio entre todos los experimentos, incluyendo las tres opciones de covarianza, para $L = 50$ y p_1 en función del parámetro de separación de clases t:

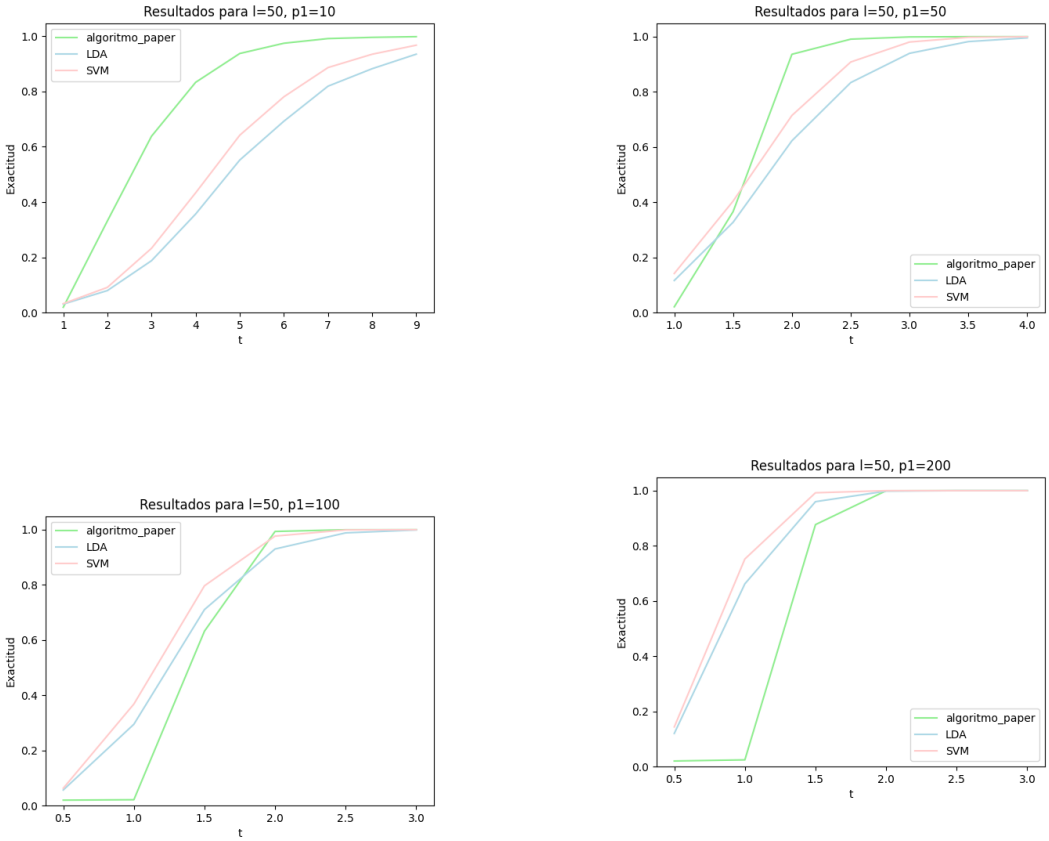


Figura 3.6: Resultados para 50 clases

Además, para poder entender qué está sucediendo con el algoritmo en cada caso, observamos la cantidad de variables detectadas promedio:

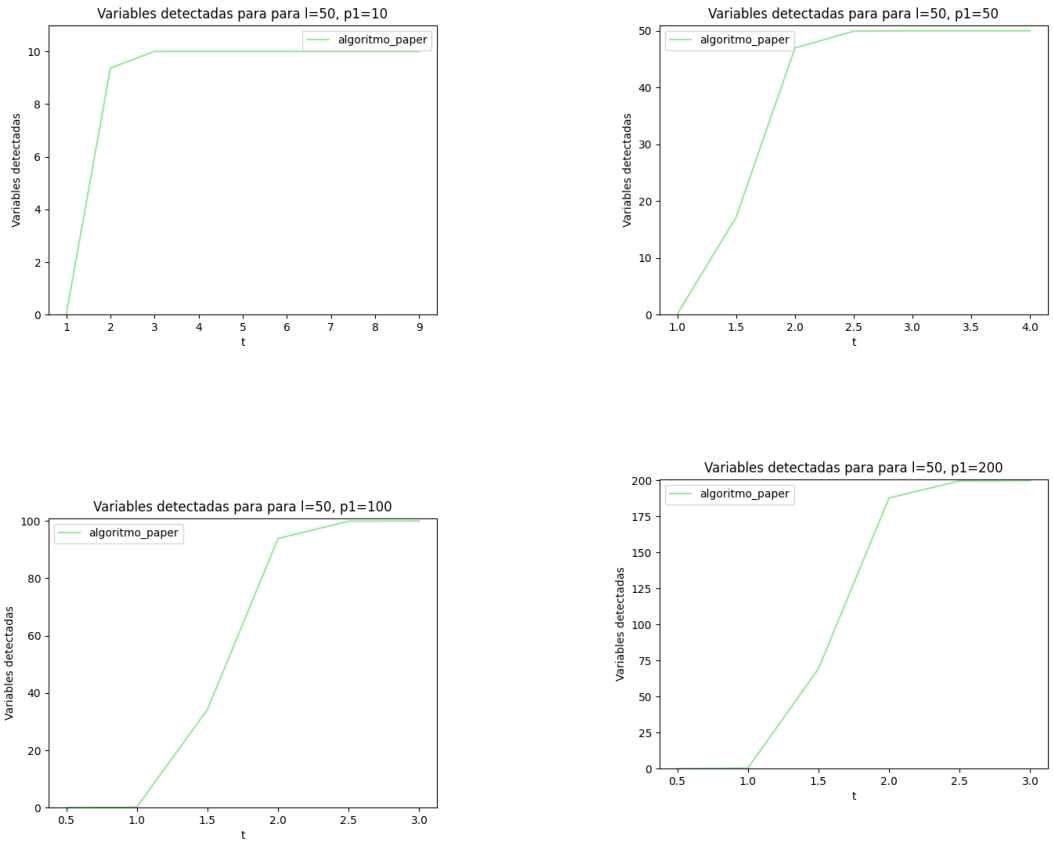


Figura 3.7: Variables detectadas en cada escenario para 50 clases

Como podemos observar, los resultados muestran que le gana a LDA y a SVM cuando hay menos dimensiones importantes. El hecho de que el algoritmo reduzca el espacio de variables cuando hay muchas que representan ruido termina implicando una ventaja comparativa.

En cambio, cuando hay una elevada cantidad de variables que distinguen a las categorías pierde. Recién comienza a distinguir el ruido de la información cuando los otros algoritmos ya obtienen rendimientos elevados, generando así que se vea superado. Por ejemplo, para $p_1 = 100$, comienza a detectarlas cuando $t = 1{,}5$ y para ese entonces, SVM y LDA ya poseen más de un 60 % de exactitud.

En consecuencia, si bien pareciera que el haber más dimensiones importantes le quita importancia a no detectarlas todas, es cuando menos hay cuando el algoritmo mejor rinde.

A partir de ahora, tomaremos en consideración los casos de $p_1 = 10$ y $p_1 = 50$.

Otra pregunta que nos interesa responder es cuál es la cantidad de categorías a partir de las cuales gana consistentemente, y entonces, hace sentido usar el algoritmo.

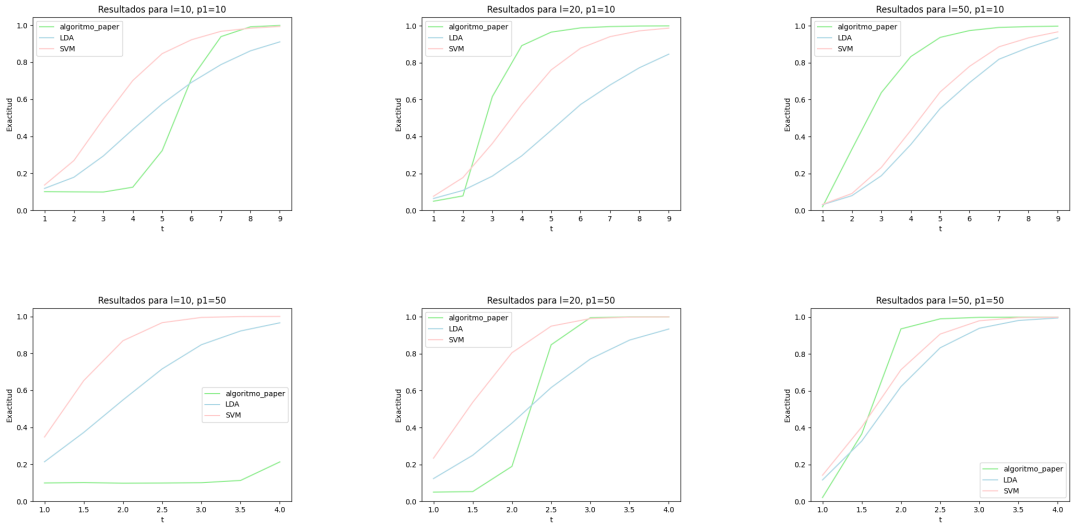


Figura 3.8: Impacto de aumentar las categorías

Tenemos también sus respectivos gráficos con las variables importantes detectadas:

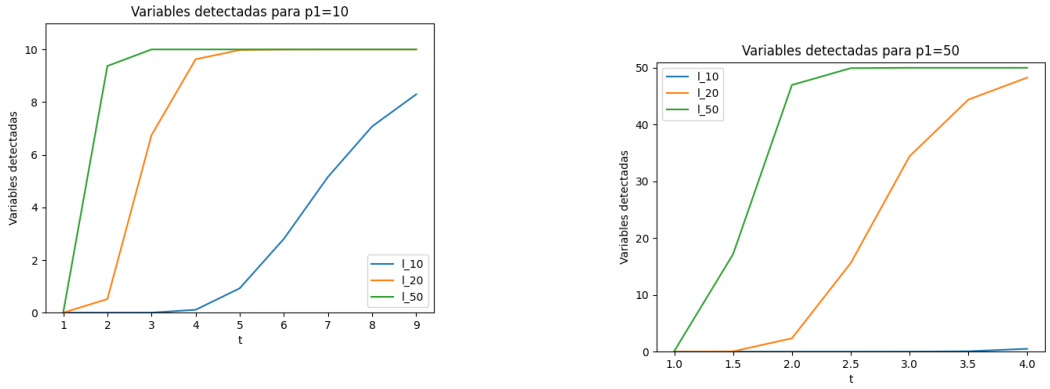


Figura 3.9: Impacto aumentar categorías: variables detectadas

Destaquemos que $L = 10$ parecieran ser muy pocas clases. Tanto para $p_1 = 10$ como $p_1 = 50$, comienza a detectar las variables importantes en escenarios donde las clases ya están muy separadas y cualquier algoritmo funciona bien. Se corresponde con lo analizado previamente, en estos casos le estamos pidiendo mucha evidencia para que superen el test de hipótesis.

Cuando aumentamos la cantidad de clases el algoritmo comienza a brillar. Ya para $L = 50$ supera ampliamente a sus competidores. Y se refleja también en que encuentra la cantidad de categorías importantes más rápido.

Para $L = 20$, el algoritmo da pelea pero no gana consistentemente.

En conclusión, el número de clases para que valga la pena ejecutar el algoritmo es de al menos 20, y cuántas más, mejor. Además si al detectar la cantidad de variables importantes el número es elevado, posiblemente se obtenga mejores resultados con otro algoritmo.

Capítulo 4

Conclusiones

En este trabajo, se llevaron a cabo simulaciones para evaluar un algoritmo de clasificación propuesto en [1]. El objetivo principal fue profundizar en el conocimiento de este método en un contexto de datos de alta dimensión y con un elevado número de categorías. El estudio se centró en un escenario ideal para la teoría del algoritmo, asumiendo que el conjunto de datos de entrenamiento era limitado para cada clase y que cada una de ellas seguía una distribución normal multivariada.

Las simulaciones demostraron que las deficiencias del algoritmo no se manifestaron en este contexto. Por un lado, no se encontraron los típicos problemas numéricos que surgen al trabajar en alta dimensión. Además, se constató que un aumento en el número de clases puede reducir el error del modelo, algo que, por ejemplo, los modelos LDA no logran.

Sin embargo, al compararlo con otros clasificadores como LDA y SVM, el algoritmo no siempre se impuso de manera consistente. Su rendimiento fue superior en los casos donde pocas variables eran las que distinguían las clases. En cambio, cuando el número de variables importantes era elevado, el algoritmo tardó demasiado en identificarlas, permitiendo que sus competidores obtuvieran un mejor rendimiento.

Estos resultados demuestran que el algoritmo es una contribución valiosa. Posee una teoría única que lo hace robusto en escenarios con múltiples clases, donde la mayoría de los algoritmos tienden a disminuir su exactitud. Por ello, es un método que merece ser considerado en cualquier evaluación comparativa.

Como trabajo futuro, se propone mejorar la detección de las variables importantes. Si bien el método actual no genera falsos positivos, en ocasiones requiere demasiada evidencia para detectar estas variables y utilizarlas en la clasificación.

Bibliografía

[1] Abramovich, Felix y Pensky, Marianna. Classification with many classes: Challenges and pluses. *Journal of Multivariate Analysis* 174 (2019), pág. 104536. DOI: 10.1016/j.jmva.2019.104536.

[2] Abramovich, Felix y Pensky, Marianna. High-dimensional classification with many classes: a blessing. *The Annals of Statistics* 49.2 (2021), págs. 1057-1084. DOI: 10.1214/20-AOS1994.

[3] Fan, Jianqing y Lv, Jinchi. High-dimensional variable selection: A review. *Statistica Sinica* 20.1 (2010), págs. 101-148.

[4] James, Gareth, Witten, Daniela, Hastie, Trevor y Tibshiran, Robert. *An Introduction to Statistical Learning with Applications in R*. Springer, 2013.

[5] Tukey, John W. *Exploratory Data Analysis*. Addison-Wesley, 1977.