PU Learning: Aprendizaje supervizado con una sola clase

Magdalena Navarro
ex alumna de Matemáticas Aplicadas
Néstor Sánchez
ex alumno de Matemáticas Aplicadas

Supongamos que eres un catador de vinos y tu trabajo consiste en probar una nueva cosecha de vinos y encontrar a los vinos de buena calidad para recomendarlos a tu clientela. Tú ya conoces algunos vinos de buena calidad de cosechas anteriores y cuentas con ciertas características químicas tanto de los buenos vinos como de los de la nueva cosecha, como la cantidad de ácido cítrico y los grados de alcohol, el problema es que la nueva cosecha es tan grande que te tomaría mucho tiempo probar cada vino [1], ¿cómo podrías seleccionar un conjunto de buenos vinos de todos éstos que nunca han sido clasificados y quedar bien con tu elegante clientela?

Si te gusta el aprendizaje estadístico probablemente pensaste en entrenar un modelo de aprendizaje supervisado que te ayude a separar los buenos vinos de los malos, el problema es que todos los metodos estándar necesitan ejemplos de ambas clases (buenos y malos) para entrenarse, y en este problema sólo tenemos ejemplos de la clase positiva. Éste problema es bastante común, por lo que ya ha sido estudiado y se han propuesto métodos para resolver este inconveniente. De aquí surgió una subrama relativamente nueva de aprendizaje estadístico: **Aprendizaje PU** (de *Positive-Unlabeled Learning*); que estudia problemas de clasificación en donde sólo un subconjunto de los ejemplos están clasificados y todos ellos pertenecen a una misma clase.

En este artículo vamos a implementar dos métodos de Aprendizaje PU: Spy-SVM y SVM-Sesgada. Ambos basados en el método de máquinas de soporte vectorial pero con ciertos ajustes para compensar la falta de observaciones negativas etiquetadas, y después, para comparar su desempeño con aprendizaje supervisado, vamos a entrenar a una máquina de soporte vectorial sin estos ajustes y comparar las clasificaciones que cada una hace.

La base de datos que vamos a utilizar tiene información de 4,898 vinos blancos. Tiene como variables ciertas características químicas (como su pH, acidez y dióxido de carbono) y un valor del 1 al 10 calificando la calidad del vino. Para simplificarlo, tomamos a vinos con calidad de 7 en adelante como buenos (clase positiva) y el resto como malos (clase negativa). [2]

Para simular el conjunto de observaciones no etiquetadas de Aprendizaje PU vamos a tomar un porcentaje de los vinos de la clase positiva y la vamos a mezclar con la clase negativa, tomando a este nuevo conjunto como el conjunto no etiquetado y vamos a ver qué tan bueno

[1] O en nuestro caso, agregarías a esta restricción de tiempo el que lo que diferencía a un vino bueno de uno malo es un misterio y el probarlos no ayudaría en nada

[2] La base se puede encontrar en http://archive.ics.uci.edu/ml/datasets/Wine+Quality

es el algoritmo para encontrar a los vinos de buena calidad que escondimos aquí. Primero, una breve introducción a SVM:

0.1. Máquinas de Soporte Vectorial (SVM)

Las máquinas de soporte vectorial (o SVM por sus siglas en inglés) son unos de los algoritmos de clasificación más sofisticado que se tienen actualmente, y su finalidad es encontrar el hiperplano que mejor separa a las clases en el espacio de variables. Entrenar un SVM con los datos observados, es equivalente a encontrar dicho hiperplano como la solución de un problema de optimización:
Supongamos que tenemos un conjunto de observaciones

$$C = \{(X_i, y_i)|i = 1, ..., n, y_i \in \{-1, 1\}\} \subset \mathbb{R}^{n+1}$$

Podemos caracterizar un hiperplano como el conjunto de $x \in \mathbb{R}^n$ tales que $w^T x - b = 0$ para alguna constante $b \in \mathbb{R}$, donde w es un vector ortogonal al hiperplano. Además queremos clasificar correctamente todas las observaciones, es decir, $w^T x_i - b \geq 1$ si $y_i = 1$ y $w^T x_i - b \leq -1$ si $y_i = -1$; Esto se puede resumir como

$$y_i(w^T x_i - b) \geq 1 \ \forall i = 1, ..., n$$

Si las clases fueran separables, podríamos encontrar un hiperplano que tuviera cierto margen entre él y las observaciones de cada clase (ver Figura 1). Estos margenes pueden ser caracterizados como $w^T x - b = 1$ y $w^T x - b = -1$ (para alguna $b \in \mathbb{R}$), y resulta ser que la distancia entre ambos es $\frac{2}{||w||}$.Lo que buscamos es maximizar la distancia entre los margenes, pero clasificando correctamente todas las observaciones. Esto se puede formular de la siguiente manera:

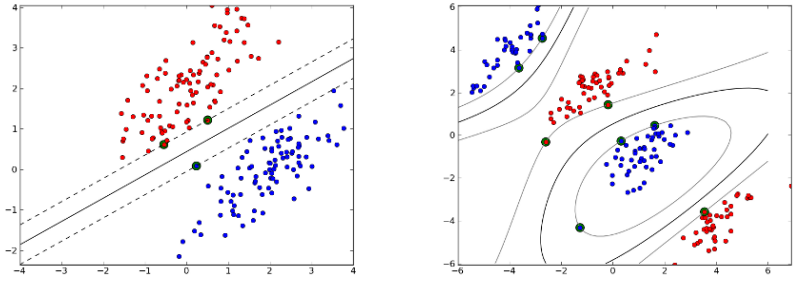
$$\begin{aligned} \min_{w \in \mathbb{R}^n, b \in \mathbb{R}^+} \quad & \tfrac{1}{2}||w||^2 \\ \text{s.a.} \quad & y_i(w^T x_i - b) \quad \geq \quad 1 \, \forall \, i = 1, ..., n \end{aligned} \tag{1}$$

La mayoría de las veces las clases no son separables y el problema de optimización de arriba no tiene solución, es por esto que la versión más usada de SVM son los llamados **soft margin SVM**, donde se permite la clasificación errónea pero se penaliza en la función de costos, de manera que las observaciones incorrectamente clasificadas se mantengan al mínimo posible:

$$\begin{aligned} \min_{w \in \mathbb{R}^n, b \in \mathbb{R}} \quad & \tfrac{1}{2}||w||^2 + C\sum_{i=1}^{n} \epsilon_i \\ \text{s.a.} \quad & y_i(w^T x_i - b) \quad \geq \quad 1 - \epsilon_i \ \ \forall \quad i = 1, ..., n \\ & \epsilon_i \geq 0 \end{aligned} \tag{2}$$

Lo que hace más interesante a este método, es que si se analiza el problema de optimización dual, se puede comprobar que la solución sólamente depende del producto punto entre los **vectores de soporte**, es decir, aquellos que caen en los margenes (ver Figura 1). Usando

esto junto con funciones **Kernel**, que mapea el producto punto en la dimensión original en un producto punto de otros espacios de dimensión mayor, incluso infinita, los SVM se pueden transformar en clasificadores no lineales muy podersos[3]:



(a) SVM ordinario

(b) SVM con kernel radial

Figura 1: Fronteras de clasificación para SVM con distintos kernel. en (b) las clases no son linealmente separables, pero esto se puede lograr usando kernel radial. En ambos podemos observar los margenes (línea punteada) y los vectores de soporte (puntos resaltados)

0.2. Biased SVM

Supongamos que tenemos un conjunto de observaciones positivas:

$$P = \{(X_i, 1)|i = 1, ..., k\}$$

y un conjunto de ejemplos no calificados:

$$U = \{x_i|i = k + 1, ...n\}$$

Asignamos la etiqueta $y_i = -1 \ \forall\, i = k + 1, ..., n$, de tal modo que los ejemplos no calificados sean tomados como ejemplos negativos. La meta es poder extraer de U las observaciones positivas que no conocemos.

El método de SVM sesgado o **biased SVM** consiste en entrenar un SVM asignando penalizaciones diferentes a errores de clasificación distintos, esto es, penalizamos más fuertemente clasificar como -1 una observación positiva que al revés. Entonces, del problema (2) pasamos a resolver el siguiente problema de optimización para obtener nuestro nuevo SVM:

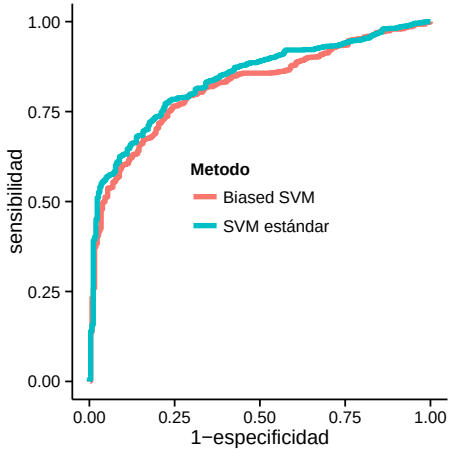
[3]Imagen obtenida de http://www.mblondel.org/

$$\begin{aligned} \underset{w\in\mathbb{R}^n, b\in\mathbb{R}}{\text{mín}} \quad & \tfrac{1}{2}||w||_2^2 + C_1\sum_{i=1}^{k}\epsilon_i \; + \mathrm{C}_2\sum_{i=k+1}^{n}\epsilon_i \\ \text{s.a.} \quad & y_i(w^T x_i - b) \;\geq\; 1-\epsilon_i \;\; \forall \qquad i = 1, ..., n \\ & \epsilon_i \geq 0 \end{aligned} \tag{3}$$

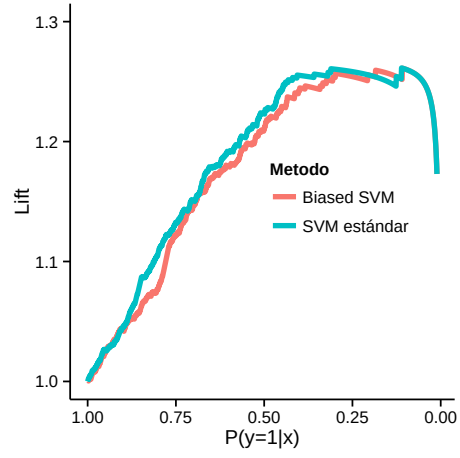
En la práctica, C_1 suele ser mucho más grande que C_2. Esto quiero decir que estamos fijándonos en clasificar correctamente a las observaciones positivas que conocemos, sin fijarnos mucho en separar al conjunto U de P. Intuitivamente, esto se basa en que las observaciones positivas en U que no conocemos probablemente serán muy cercanas en el espacio a las de P que sí conocemos, mientras que las negativas estarán más alejadas, y así, al esforzarse en clasificar correctamente las observaciones en P, de paso clasificará correctamente a las demás. Para la selección de pesos se usa validación cruzada, y la métrica de desempeño es el F1 score, aunque algunas otras han sido propuestas.

0.3. Resultados

La pregunta obvia es ¿Qué tan buenos son estos métodos? Para esto, compararemos los resultados con los métodos estándar de aprendizaje estadístico, usando las etiquetas reales de un conjunto de prueba:



(a) curva ROC de ambos modelos



(b) lift de ambos modelos

Como vemos en ambos gráficos, el desempeño de ambos modelos es muy parecido; Ambos nos proporcionan un aumento de casi 30 % en la captura de observaciones positivas con respecto a las proporciones a priori según la gráfica de lift. Esto es notable si tomamos en cuenta que el modelo de PU learning (línea roja) fue entrenado usando mucha menos información que el SVM normal. Las matrices de confusión generadas son las siguientes:

	malo	bueno
malo	526	18
bueno	438	243

Cuadro 1: Biased SVM

	malo	bueno
malo	707	51
bueno	257	210

Cuadro 2: SVM estándar

Las matrices de confusión (etiquetas reales por columnas, etiquetas de los modelos por filas) nos dejan ver que el SVM estándar es mejor clasificando correctamente los ejemplos negativos (malos), pero nuestro modelo es mejor detectando ejemplos positivos (buenos). Estas proporciones se pueden ajustar cambiando la probabilidad de corte de los modelos; En las tablas ambos cortes son de 0.5. En conclusión el metodo de biased SVM y otros métodos de

PU learning son extremadamente útiles, pues permiten tener un desempeño muy parecido a los algoritmos estándar de aprendizaje estadístico en casos donde, por falta de información suficiente, dichos algoritmos no podrían ser utilizados. De hecho, en [1] se ha llegado a afirmar que estos métodos pueden dar mejores resultados que los métodos entrenados con ejemplos de ambas clases, en casos donde los ejemplos negativos en los conjuntos de prueba y entrenamiento no vienen de la misma distribución de probabilidad. [4]

Bibliografía

[1] Xiao-Li Li, Bing Liu, and See-Kiong Ng. Negative training data can be harmful to text classification. In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP '10, pages 218–228, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[2] Bing Liu, Yang Dai, Xiaoli Li, Wee Sun Lee, and Philip S. Yu. Building text classifiers using positive and unlabeled examples. In In: Intl. Conf. on Data Mining, pages 179–186, 2003.

[3] Dell Zhang and Wee Sun Lee. Learning classifiers without negative examples: A reduction approach. In ICDIM, pages 638–643. IEEE, 2008.

[4] Este proyecto puede ser descargado en https://github.com/nestorSag/PU-learning-project