Minerva Schools At KGI
SVM for Classifying The Godfather II's reviews
Quang Tran

Problem Description

In this project I will explore the problem of text classification using the method of Support Vector Machine (SVM). Specifically, given a review for the movie The Godfather, the problem will classify the review as positive or negative.

Data Description

The data set used in this project is self-synthesized. 20,000 audience (as opposed to critics) reviews are pulled from RottenTomatoes website (URL link: 'https://www.rottentomatoes.com/m/godfather/reviews?type=user). To accomplish this, I created a web driver using the `selenium` Python package and simulated clicking the "Next" button on the website to see all the audience reviews. The scraping stops when 20,000 reviews are in the `rottentomatoes.csv` file.

Rates that are larger than 3 (>=3.5) are considered positive, otherwise not positive. However, I will denote "not positive" as "negative", because there should have been a third "neutral" class and in this project I will explore binary classification only.

Pre-processing Data

Data needs to be processed so that it is in a form ready to feed into the SVM model. The pre-processing procedure heavily borrows from the following Medium article:

https://medium.com/@bedigunjit/simple-guide-to-text-classification-nlp-using-svm-and-naive-bayes-with-python-421db3a72d34

The process steps are as follows:

1. Remove rows of null data.
2. Convert all the characters in text to lower case.
3. Break the texts down words and phrases and put them into a list.
4. Remove stop words, such as "the", "a", and "an". These words are believed to not contribute to the sentiment/meaning of the sentence.
5. Remove non-alpha text.
6. Lemmatize, which is to only retain the most basic form of words. For example, "playing" will become simply "play". This is to reduce the final size of the input vectors.

   The end result of this process is each data row now has a list of words that are critical in assessing its sentiment. For example, ['play', 'volleyball'] corresponds to review "I am playing volleyball."
7. Relabel, which is to say "positive" reviews have label 1, and "negative" reviews 0.

8. Extract features for each data point: we use the bag-of-words technique. The features for each review are the frequencies of all the words in our dictionary (5,000 words in total).
9. **Train-test split:** The training data has 14,000 data points, while the testing data 6,000 data points.

At the end of this process, we have sparse matrices for training data (of shape (14k, 5k)) and for testing data (of shape (6k, 5k)).

Solution Specification: Support Vector Machine (SVM)

The linear separable case

Suppose we have data $x \in \mathbb{R}^{n \times m}$, a vector of label $y \in \mathbb{R}^n$ (where $y_i \in \{-1,1\}$). Our goal is to find a hyperplane that separates the two classes $w^T x_k + b = 0$ $(w \in \mathbb{R}^m, b \in \mathbb{R})$. If a point x_k of unknown class is above this decision boundary ($w^T x_k + b \geq 0$), we classify it as 1. If $w^T x_k + b \leq 0$, we classify as -1. The decision rule can be condensed into $y_k(w^T x_k + b) \geq 0$.

If the data are linearly separable, there is some distance between the decision boundary and the nearest points of the classes. Readjusting the decision boundary into the "middle", there exists a positive β where $y_i(w^T x_i + b) \geq \beta$. Rescaling, we have $y_i(w^T x_i + b) \geq 1$ as our constraints.

What to optimize? It's intuitive we maximize this distance between a nearest point of a class to the decision boundary. This distance is half of $\frac{2}{||w||} = \frac{2}{\sqrt{w^T w}}$ (See Appendix A). Because square root is a monotonically increasing function, our problem is thus:

$$\max_{w,b} \frac{2}{\sqrt{w^T w}} \equiv \min_{w,b} \sqrt{w^T w}\,/2 \equiv \min_{w,b} w^T w\,/2$$

subject to

$$y_i(w^T x_i + b) \geq 1 \; \forall i = 1,2,\dots,n$$

The non-linear separable case

If the data is not linearly separable, the above constraints are not feasible. We introduce a slack variable $\varepsilon \in \mathbb{R}^n$ to relax the constraints. Therefore, rather than $y_i(w^T x_i + b) \geq 1$, we have

$$y_i(w^T x_i + b) \geq 1 - \varepsilon_i, \forall i = 1,2,\dots,n$$

where $\varepsilon_i \geq 0$.

The ε_i represents the extent of violation of the data point x_i to the boundary of its class. We want to minimize these violations as much as possible by minimizing the total errors ε_i. Therefore, in a revised optimization problem, we will have a tradeoff between maximizing the width between the two boundaries and minimizing the total error sum:

$$\min_{w,b,\varepsilon} \frac{w^T w}{2} + C \sum_{i=1}^{n} \varepsilon_i$$

subject to:

$$y_i(w^T x_i + b) \geq 1 - \varepsilon_i, \forall i = 1,2,\dots,n$$
$$\varepsilon_i \geq 0, \forall i = 1,2,\dots,n$$

The parameter C is to give us the flexibility of determining to what extent we tolerate the violations. Based on the formulation, a larger C means we have low tolerance towards the errors and thus the margin is narrower to reduce the cost incurred by the violations. [1]

This is a convex optimization problem, because:

1. The objective function is convex. This is because:
    a. $w^T w$ is the square of the norm-2 of w. Since norm-2 is convex and a square of a convex function is also convex, $\frac{w^T w}{2}$ is convex.
    b. The latter term of the objective is the scaled linear (identity) function. Since linear functions are convex (it's also concave), and scaling a convex function also gives a convex function, the latter term is also convex.
    c. Finally, the sum of two convex functions is convex.
2. The feasible set is also convex (shown below).

The above is a quadratic programming problem with linear constraints, because it can be rewritten in the standard form:

$$\min_r r^T \begin{bmatrix} 1/2\mathbb{I}_{m\times m} & 0_{(n+1)\times m} \\ 0_{(n+1)\times m} & 0_{(n+1)\times m} \end{bmatrix} r + \begin{pmatrix} 0_m \\ C1_n \\ 0 \end{pmatrix}^T r$$

subject to

$$Ar \leq p$$

Where $r \in \mathbb{R}^{m+n+1}$ is a vector of all variables ($r = [w\ \varepsilon\ b]$) , $C1_n$ is a vector of n C's,

$$A = \begin{bmatrix} -y_1 x_1 & y_2 x_2 & \cdots & -y_n x_n & 0_m & 0_m & 0_m & 0_m \\ -1 & 0 & \cdots & 0 & -1 & 0 & \cdots & 0 \\ 0 & -1 & \cdots & 0 & 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & -1 & 0 & 0 & \cdots & -1 \\ -y_1 & -y_2 & \cdots & -y_n & 0 & 0 & 0 & 0 \end{bmatrix}^T \in \mathbb{R}^{2n\times(m+n+1)}$$

and

$$p = \begin{pmatrix} -1_n \\ 0_n \end{pmatrix} \in \mathbb{R}^{2n}$$

Based on the rewritten form above, because the inequality constraint is affine, the feasible set is a polyhedron, which means it is convex.

---

[1] **#hypothesisdevelopment:** develop a plausible hypothesis about the effect of C on the optimization problem; later provided an empirical demonstration.

Results and evaluation of the nature of optimal values

Optimization code: https://gist.github.com/quangntran/61c9a7a69a6d99b306e7c63f91056f7c

The effects of the parameter C

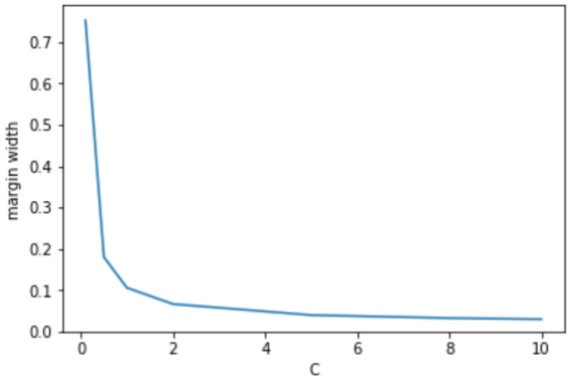

Figure 1. Relationship between C and the width of margin.

A larger C means we have low tolerance towards the errors and thus the margin is encouraged to be narrower as when it is, the violations are fewer. The experiment confirms this intuition.

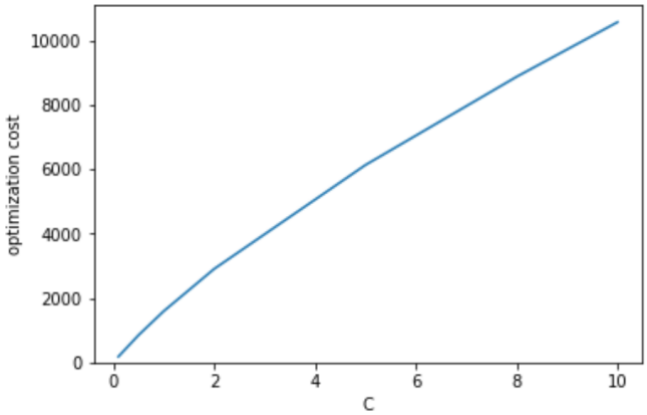

Figure 2. The effect of C on the optimal cost

When C=0, we incurr zero loss for the error terms. As such, the margin can be the widest possible, and thus the cost for the inverse width is low. When C increases, the increase in the cost is double-fold: the cost from a narrower width of the margin and the cost from a now heavier penalization by a larger C. This explains why the optimal cost increased rapidly as C increases.

However, we should not always strive for the lowest optimal cost (when C=0), for the reasons in the following section.

Evaluation metrics: precision and recall

The ultimate goal is to get the learned decision boundary perform well in an unseen data set. Because The Godfather has really rate, which means the data is skewed towards the positive class, accuracy is not a good metric to evaluate on the test set. The machine could predict all reviews to be positive and obtain a high accuracy. In the context of product review, we don't want to miss negative reviews this way ("hard feedback is good feedback.") Also, we might not want to misjudge a positive review as a bad review either (maybe because once a review is deemed "negative", we as a company would spend a lot of effort and money following up on that bad review – such as contacting the customer about the bad review and offering them a gift as a compensation). In the code, I adjusted the class so the meaning of precision and recall is as follows:

1. Recall: how well we detect negative reviews
2. Precision: off all the reviews we predict to be negative, how well we have not misjudged them

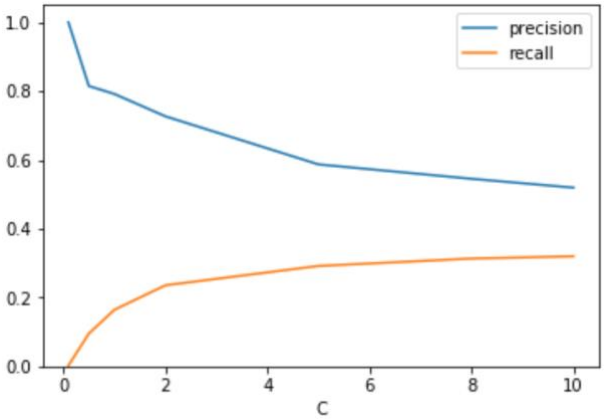

Figure 3. Precision and recall (on negative reviews) as a function of C

We see that there is a tradeoff. For a lower C, as analyzed above, because there is no toll in the datapoint violations of the classes of the data points, the margin width is large and the decision boundary separates most of the data points to one side of the hyperplane, causing the predictions to be mostly positive, leading to a low recall and high precision. Conversely, when C is higher, recall is raised and precision suffers. Depending on our goals and threshold, we may choose an appropriate value for C.

References

Calafiore, G., & Ghaoui, L. E. (2014).*Optimzation models*. Cambridge: Cambridge.

Scikit-learn.org. *1.4. Support Vector Machines — scikit-learn 0.22.2 documentation.* Retrieved from https://scikit-learn.org/stable/modules/svm.html

Appendix A - The distance between a nearest point of a class to the decision boundary is $\frac{1}{||w||}$

Let x_1 be a point on the boundary for +1 class, then $y_1(w^T x_1 + b) = 1$ or $w^T x_1 + b = 1$.
Let x_2 be a point on the boundary for -1 class, then $y_2(w^T x_2 + b) = 1$ or $w^T x_2 + b = -1$.
The vector connecting x_2 and x_1 is $q = x_2 - x_1$. Because w is perpendicular to the boundaries, the width between the two boundaries for the classes is the magnitude of the projection of q onto w. This projection is:

$$q \cdot \frac{w}{||w||}\hat{w} = (x_2 - x_1) \cdot \frac{w}{||w||}\hat{w} = (w^T x_2 - w^T x_1)\frac{\hat{w}}{||w||} = [(-1 - b) - (1 - b)]\frac{\hat{w}}{||w||} = \frac{-2\hat{w}}{||w||}$$

The magnitude of this projection is $\frac{2}{||w||}$. Half of this distance is the distance between a boundary of a class to the decision boundary.

Appendix B – The equivalence of the CVXPY implementation and the Sklearn implementation

In the section CVXPY vs Sklearn in the linked notebook, I demonstrated that the cvxpy implementation works properly by testing whether the predictions in the test set differ between the two implementations with various values of C. The result is that no difference is spotted