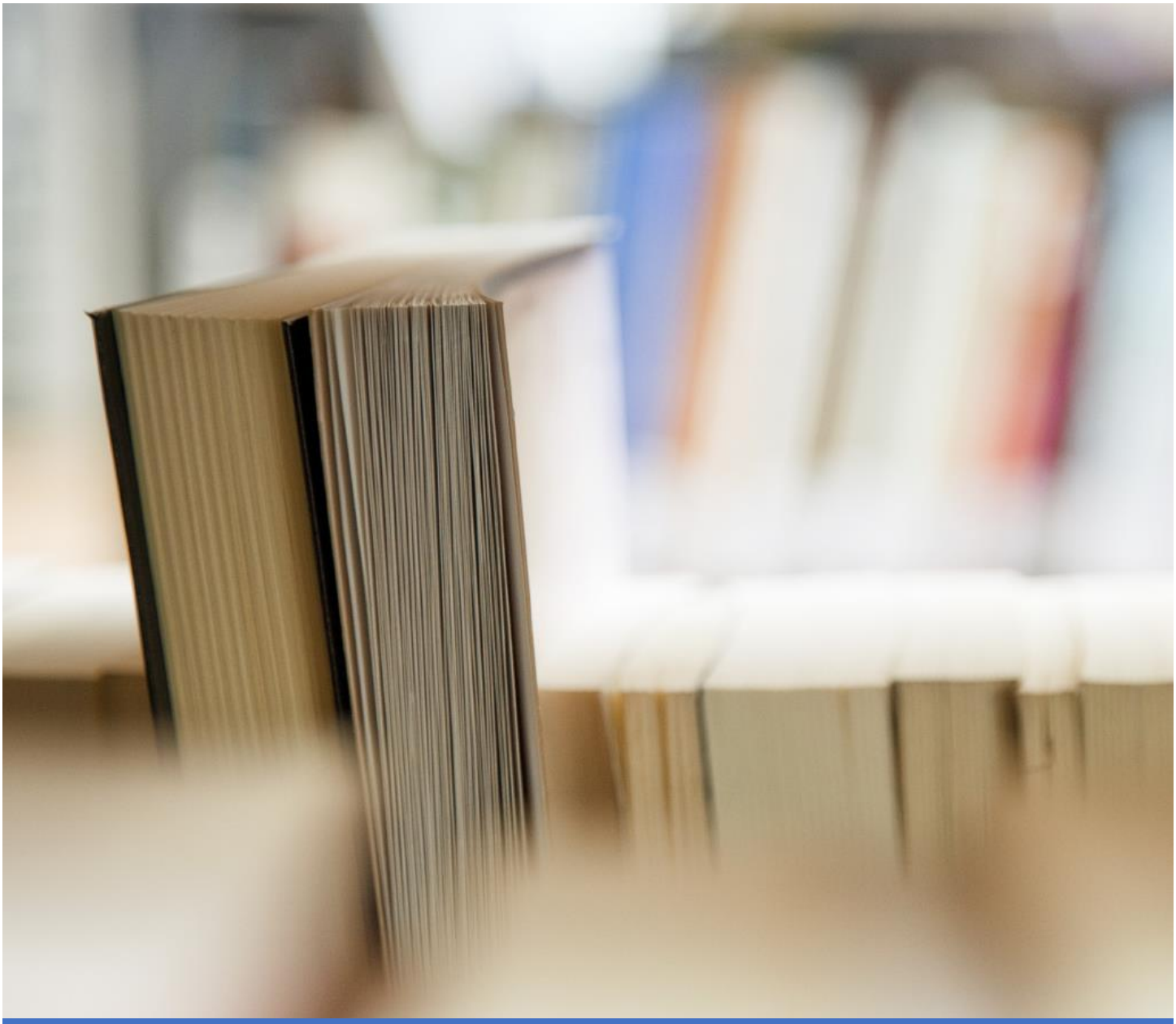


A MODIFIED SELF-TRAING SEMI-SUPERVISED SVM ALGORITHM

Paper Reproduction

2020021327 조경선[대학원석사과정재학 / 산업경영공학과]

12/27/20

BUSINESS ANALYTICS

Paper Summary

이 논문에서는 수정된 self-training semi-supervised SVM 알고리즘을 제시한다.

A Modified Self-training Semi-supervised SVM Algorithm

(A) Former self-training semi-supervised SVM algorithm

Standard SVM classifier 는 다음과 같이 정의된다.

$$\min \frac{1}{2} w^T w + C \sum_{i=1}^{l} \varepsilon_i$$

$$\text{Subject to } y_i(w^T x_i + b) \geq 1 - \varepsilon_i,\ \varepsilon_i \geq 0,\ i = 1, \cdots, l$$

(B) Modified self-training semi-supervised SVM algorithm

수정 된 SVM 은 두 클래스에 대한 constraint least-squares optimization 문제인 Fisher's discriminant optimization 문제에서 모티브를 얻은 모형이다. 클래스 내 분산을 최소화하는 문제는 분리 가능한 경우와 분리 할 수없는 경우 모두에 대해 최적의 hyperplane 을 통해 분리할 수 있도록 재구성되었다.

먼저 수정 된 SVM 의 최적화 문제를 형성하기 위해 training set 의 클래스 내 scatter matrix 를 다음과 같이 정의 한다.

$\mu_k^1,\ \mu_k^2$는 U_k^1, U_k^2 의 mean vector 라고 가정한다. Scatter matrix S_w^k 는 다음과 같이 정의된다.

$$S_w^k = \sum_{g_i U_k^1} (g_i - \mu_k^1)\,(g_i - \mu_k^1)^T + \sum_{g_i U_k^2} (g_i - \mu_k^2)\,(g_i - \mu_k^2)^T$$

수정 된 SVM 은 다음과 같다.

$$\min \frac{1}{2} w_k^T S_w^k w_k + C_k \sum_{j=1}^{l} \varepsilon_j^k$$

$$\text{Subject to } y_i^k \left(w_k^T x_i + b_k\right) \geq 1 - \varepsilon_i^k,\ \varepsilon_i^k \geq 0,\ j = 1, \cdots, l$$

Lagrangian Problem

$$L\left(w_k, b_k, \alpha^k, \beta^k, \varepsilon^k\right)$$

$$= w_k^T S_w^k w_k + C_k \sum_{j=1}^{l} \varepsilon_j^k - \sum_{i=1}^{l} \alpha_i^k \left[y_i^k \left(w_k^T g_i x_i - b_k\right) - 1 + \varepsilon_i^k\right] - \sum_{i=1}^{l} \beta_i^k \varepsilon_i^k$$

KT Condition

$$w_k = \frac{1}{2} S_w^{k-1} \sum_{i=1}^{l} \alpha_i^k y_i^k \, g_i$$

Wolf Dual Problem

$$W(\alpha^k) = \sum_{i=1}^{l} \alpha_i^k - \frac{1}{4} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i^k \alpha_j^k \, y_i^k y_j^k x_i^T g_i^T S_w^{k-1} g_j$$

Subject to $0 \le \alpha_i^k \le C, i = 1, \cdots, l$

Experiments

(A) Parameter Selection

SVM 구현에 있어서 C 값을 먼저 셋팅하여야 한다. training data 로 cross-validation 을 한 결과 C value 의 Recognition Rate 가 아래의 표와 같음을 구할 수 있었다. 그에 따라서 가장 좋은 성능을 보이는 C=0.05 값을 사용한다.

Table 1.The Recognition Rate with Different C Value

C Value	0.01	0.012	0.015	0.02	**0.05**	0.1	0.2	0.5
Recognition Rate	60.9	68.6	77.3	83.2	**83.2**	80.5	78.6	79.5
C Value	1	2	5	10	50	100	200	500
Recognition Rate	77.7	77.7	73.6	71.8	70	74.1	73.6	73.6

(B) Simulation Result

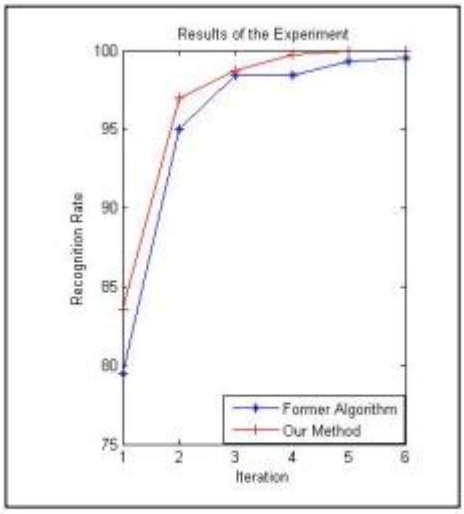


Figure 3. Result of the data set "heart_scale". The lower line with "*" is the results of the former algorithm and the upper line with "+" is the results of our modified algorithm

TABLE III. RECOGNITION RATE OF CLASSIFICATION RESULT OF INDEPENDENT DATA SET D_2

Data	Our Method	Former Algorithm
heart_scale	85.4	81.2
iris	100	100
wine	94.8	93.5
vehicle	89.1	85.6

Reproduction

구현을 위해 Iris Data 를 사용하였다. 논문에서는 unlabeled data 의 비중이 표기되지 않아서, 임의로 30%를 unlabeled data 로 분류하여 테스트 하였다. 또한 Kernel 함수에 설명이 없었으므로 Linear SVM 을 수행하였다.

Data Preperation

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets
from sklearn.semi_supervised import SelfTrainingClassifier
from sklearn.svm import SVC

# Load Data - Iris
iris = datasets.load_iris()

X = iris.data[:, :2]
y = iris.target

rng = np.random.RandomState(0)
y_rand = rng.rand(y.shape[0])
y_30 = np.copy(y)

# 30% unlabeled data 를 Random 하게 추출
y_30[y_rand < 0.3] = -1
```

Original SVM

```
# Origin SVM algorithm
linear_svc = (SVC(kernel='linear', C=0.05).fit(X, y), y, 'SVC with linear
kernel')
```

Former self-training semi-supervised SVM algorithm

```
sssvm_classifier = SVC(kernel='linear', C=0.05, probability=True)
sssvm = (SelfTrainingClassifier(sssvm_classifier).fit(X, y_30), y_30,
'Former SSSVM')
```

Modified self-training semi-supervised SVM algorithm

수정된 Self training Supervised SVM 의 식을 커널함수로 구현하여 대입해 보았다.

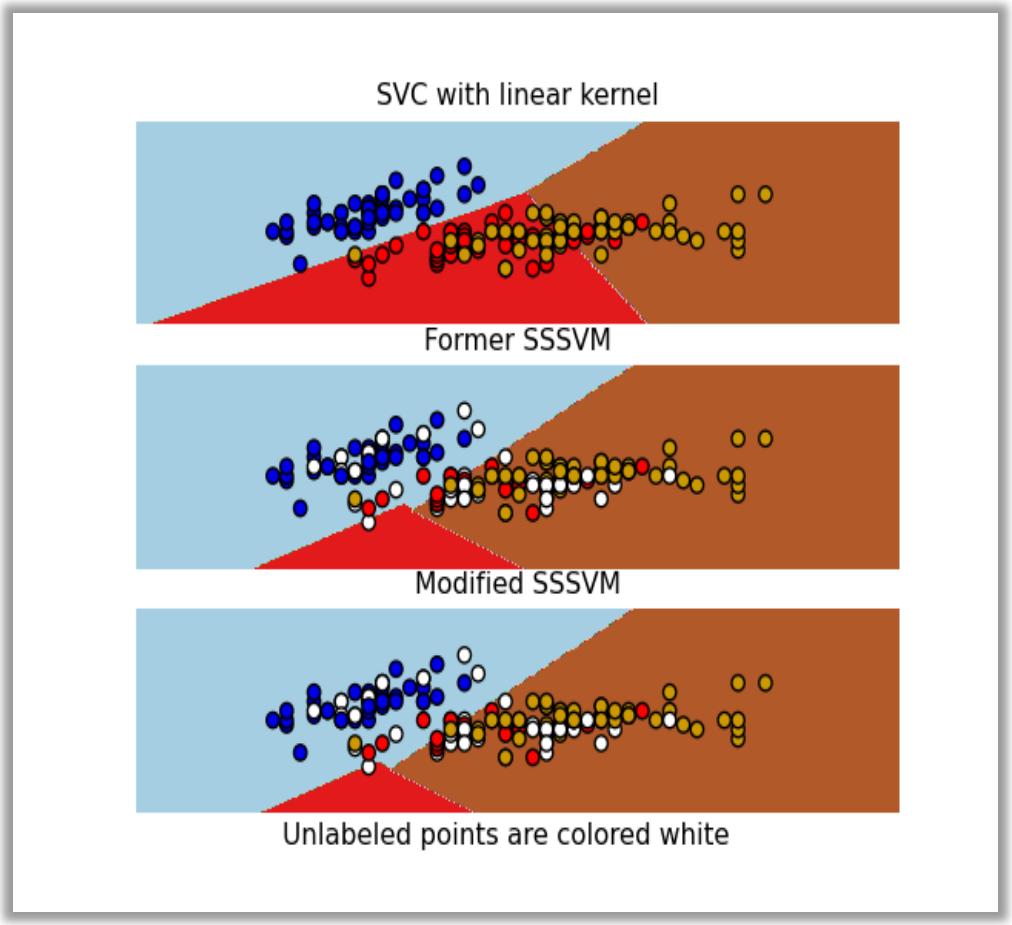
```
def modified_linear(X,Y):
    s = np.subtract(X, np.mean(X))
    S = np.dot(s, s.T)
    return np.divide(np.dot(np.dot(X.T,S), X), 2)

modified_classifier= SVC(kernel='linear', C=0.05, probability=True)
msssvm = (SelfTrainingClassifier(modified_classifier).fit(X, y_30), y_30,
'Modified SSSVM')
```

Result

```
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max,
h))

color_map = {-1: (1, 1, 1), 0: (0, 0, .9), 1: (1, 0, 0), 2: (.8, .6, 0)}

classifiers = (linear_svc, sssvm, msssvm)
for i, (clf, y_train, title) in enumerate(classifiers):
    plt.subplot(3, 1, i + 1)
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

    Z = Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, cmap=plt.cm.Paired)
    plt.axis('off')

    colors = [color_map[y] for y in y_train]
    plt.scatter(X[:, 0], X[:, 1], c=colors, edgecolors='black')

    plt.title(title)

plt.suptitle("Unlabeled points are colored white", y=0.1)
plt.show()
```

Result And Comparison

논문에서 Iris Data 에 대해 Former Self training Semi-Supervised SVM 과 수정된 Self training SVM 에서 성능의 차이가 없다고 하였다. 하지만 실재 구현된 모델에서는 약간의 차이가 존재하는 결과가 나타났다.



Conclusion

기존의 Self-Training Semi-Supervised SVM Algorithm 에서 Fisher's discriminant optimization 에서 영감을 얻은 Modified Self-Training Semi-Supervised SVM 을 구현하였다. 수업시간에도 배웠던 Supervised Learning 기법인 SVM 를 Semi-Supervised 기법으로 변형하여 대입해 보았다는 점에서 의미가 더욱 큰 것 같다. Iris Data 가 아닌 더 복잡한 모형에 대해서도 테스트 해 볼 예정이며, 또한 다른 Algorithm 에 영감을 얻어 변형된 모델을 만들었다는 점에서도 굉장히 흥미로운 논문이었다. 앞으로 나의 연구에서도 이러한 기법을 잘 활용할 수 있길 기대한다.