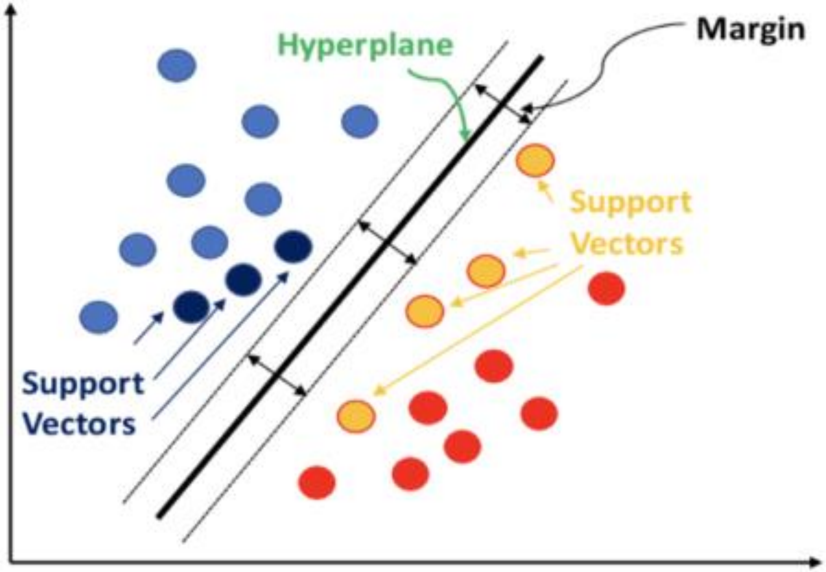
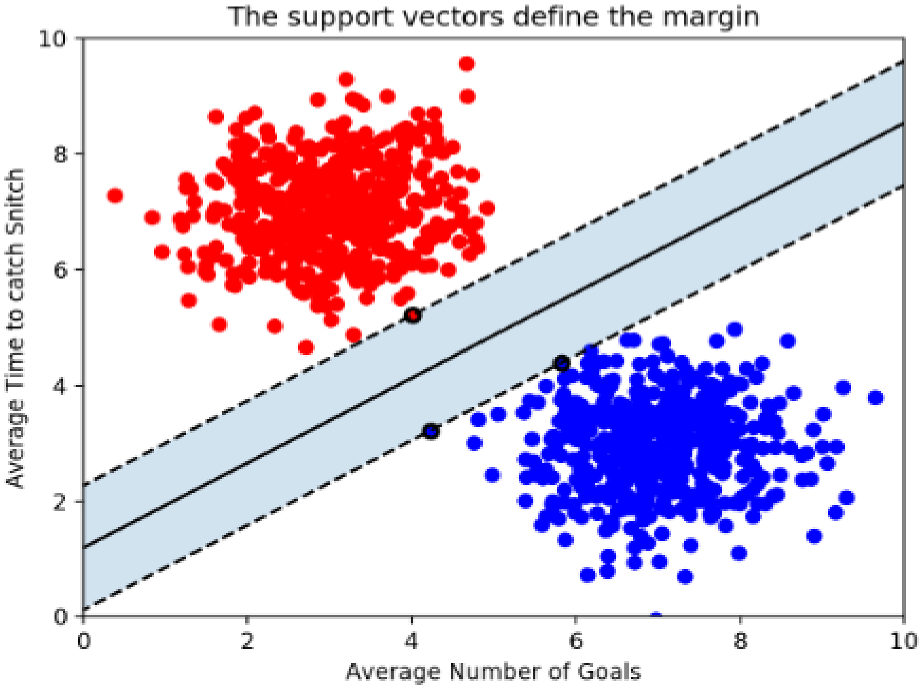
제2장 머신러닝-SVM
(Support Vector Machine)

SVM이란?

- 이진 분류(Binary Classification)에 사용
- 마진(Margin)의 최대화



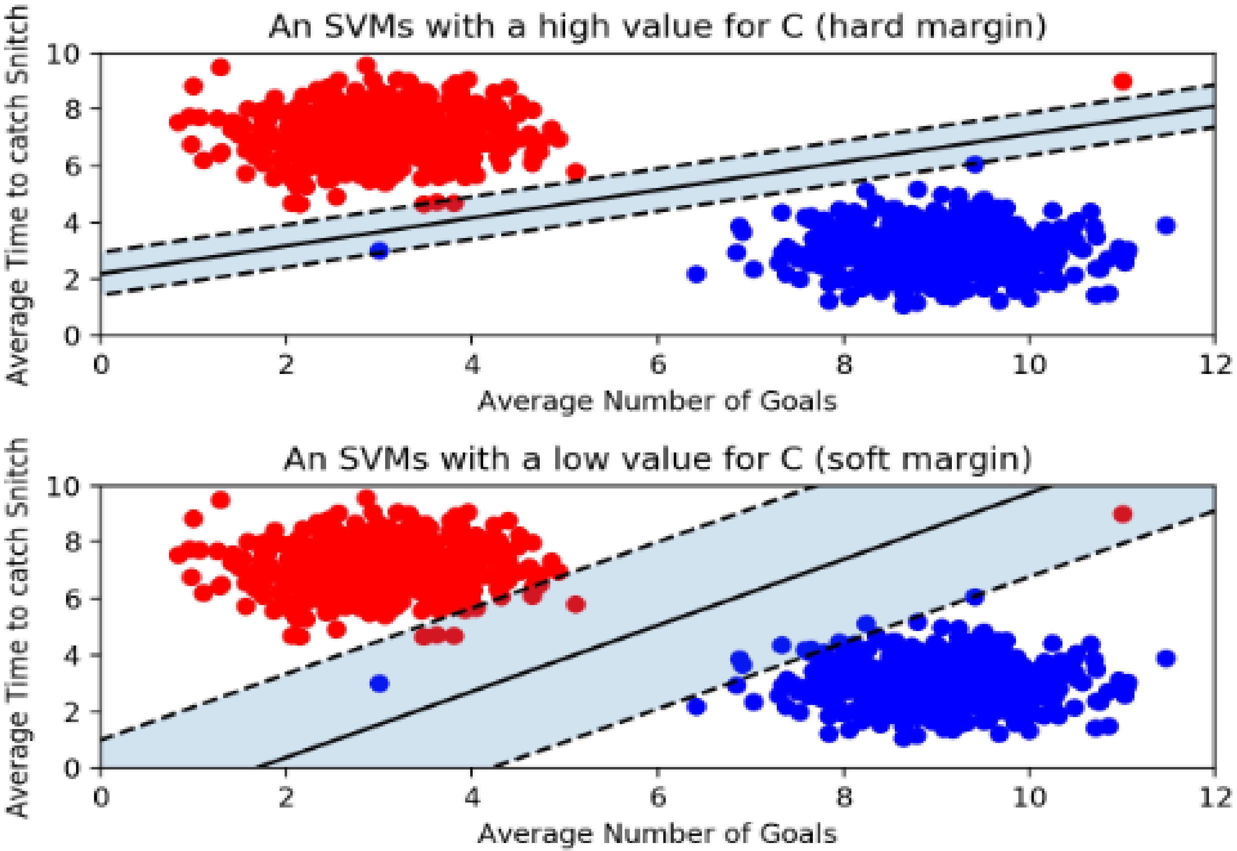
출처: https://datatron.com/what-is-a-support-vector-machine/



출처 : https://blog.eunsukim.me/posts/understanding-support-vector-machine

SVM이란?

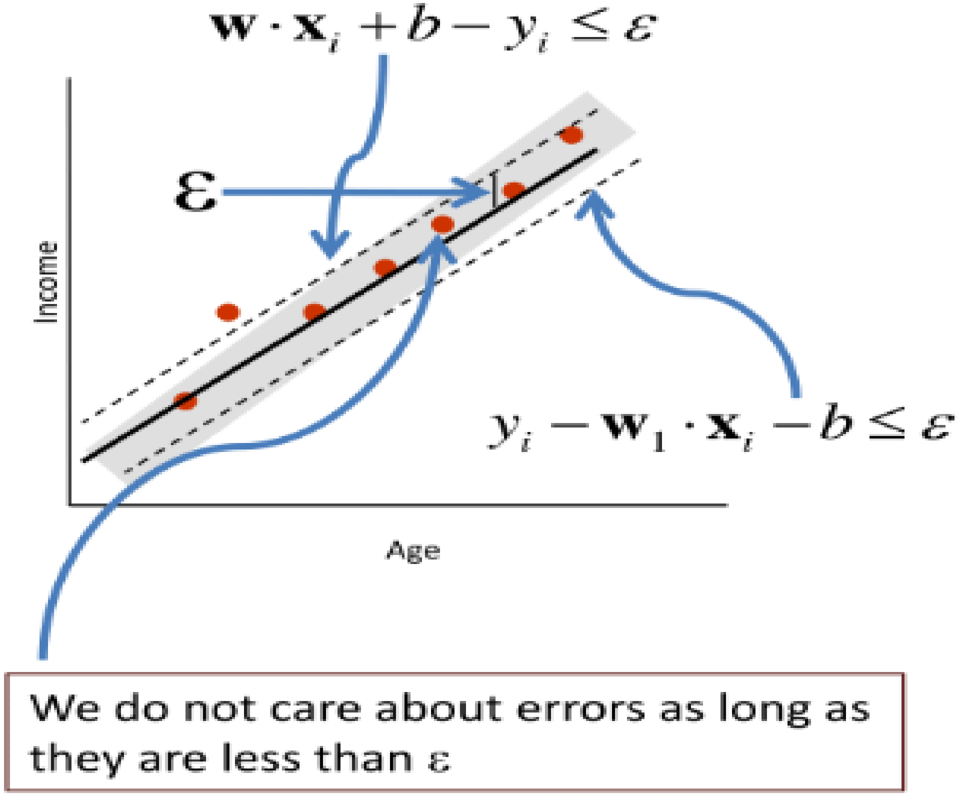
- Outliers(이상값)의 영향을 받음 : marin 최대화로



출처: https://blog.eunsukim.me/posts/understanding-support-vector-machine

SVM이란?

- Outliers(이상값)의 영향을 받음 : marin 최대화로



$$\mathbf{w}\cdot\mathbf{x}_i + b - y_i \le \varepsilon$$

ε

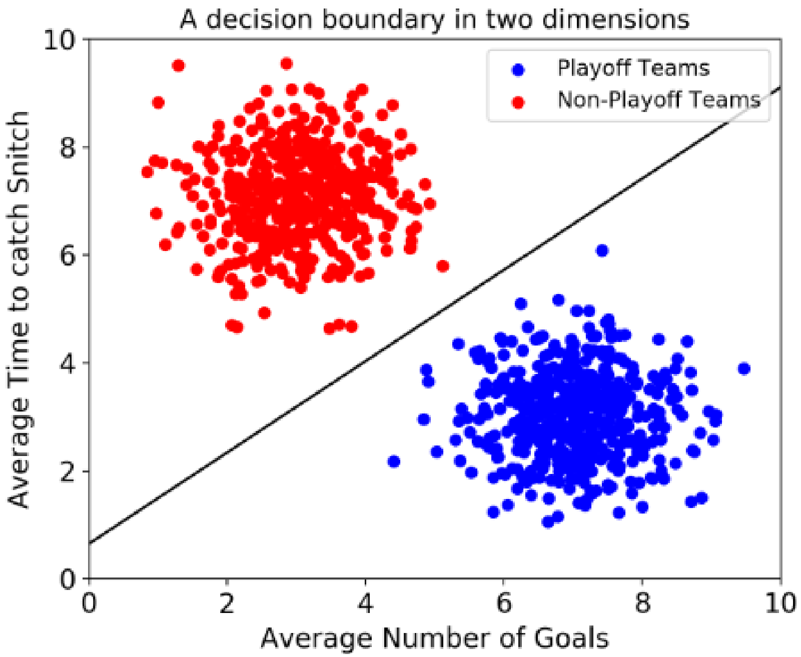
$$y_i - \mathbf{w}_1 \cdot \mathbf{x}_i - b \le \varepsilon$$

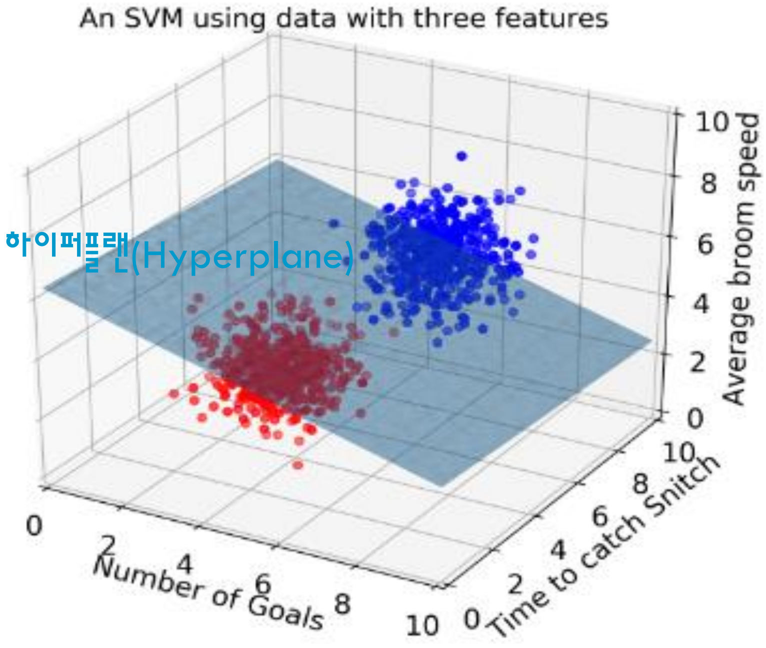
We do not care about errors as long as they are less than ε

출처: https://www.linkedin.com/pulse/role-svm-model-current-data-science-deepak-kumar

SVM이란?

- 2D / 3D





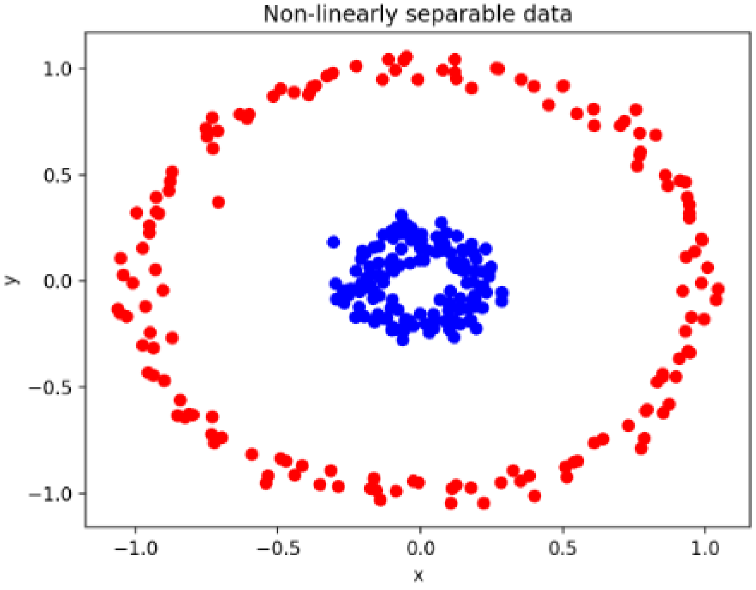
출처 : https://blog.eunsukim.me/posts/understanding-support-vector-machine

SVM이란?

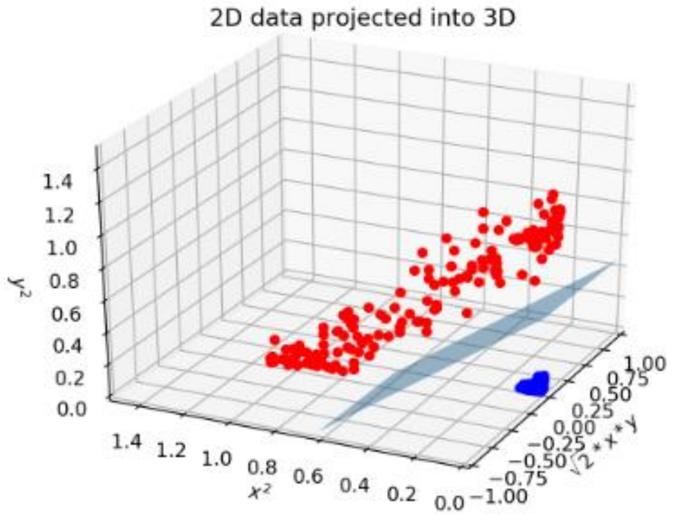
- Kernels / Polynomial Kernel
- kernel paremeter “Ploy”로 설정하면 모든 data를 transform 함

$$(x, y) \rightarrow (\sqrt{2} \cdot x * \cdot y, x^2, y^2)$$

$$(1, 2) \rightarrow (2\sqrt{2}, 1, 4)$$



Non-linearly separable data



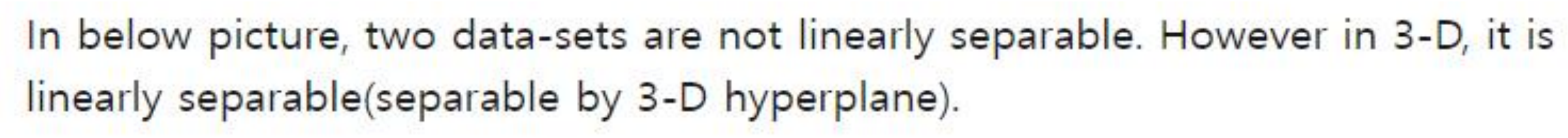
2D data projected into 3D

하이퍼플랜
(Hyperplane)

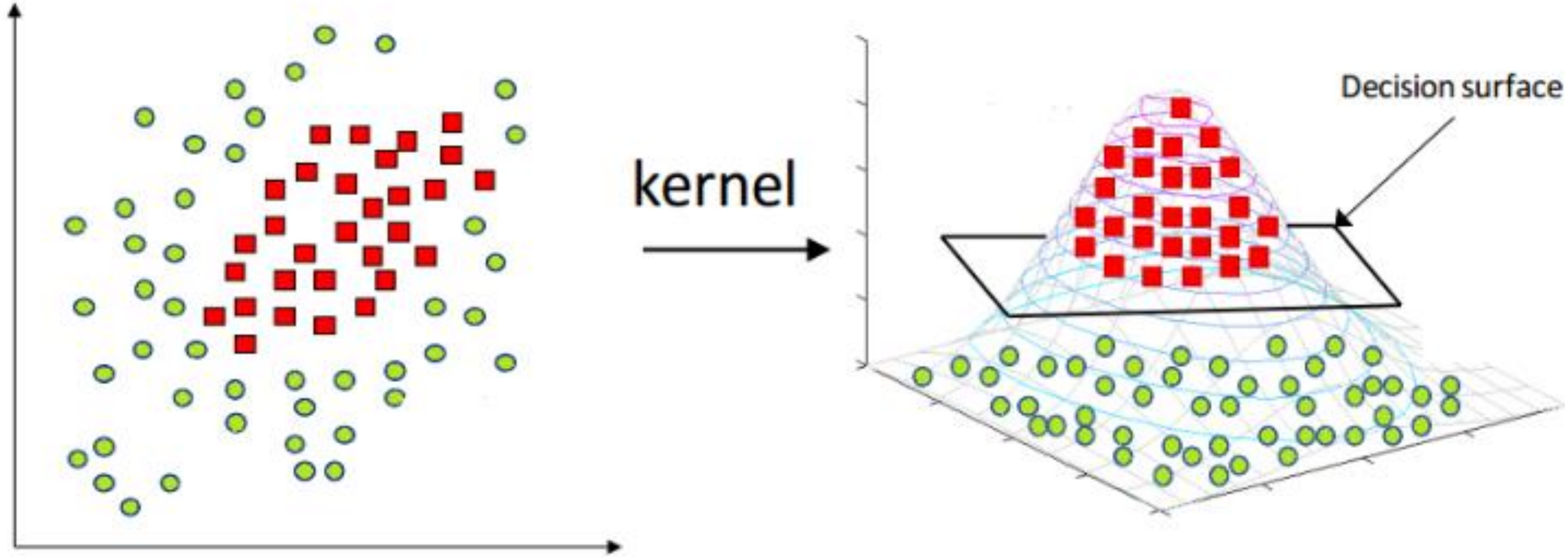
출처 : https://blog.eunsukim.me/posts/understanding-support-vector-machine

SVM이란?

- How SVM can solve non-linear problem?



In below picture, two data-sets are not linearly separable. However in 3-D, it is linearly separable(separable by 3-D hyperplane).



하이퍼플랜
(Hyperplane)

출처 : https://www.linkedin.com/pulse/role-svm-model-current-data-science-deepak-kumar

SVM 코드

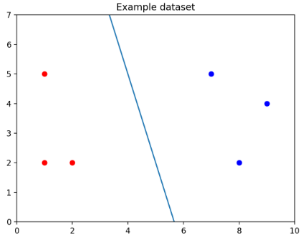
- from sklearn.svm import SVC
- classifier = SVC(kernel = 'linear')   # ‘poly’ #”rbf” : 자동 설정

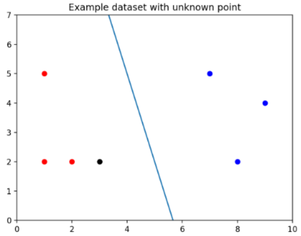
- training_points = [[1, 2], [1, 5], [2, 2], [7, 5], [9, 4], [8, 2]]
- labels = [1, 1, 1, 0, 0, 0]
- classifier.fit(training_points, labels)

- rint(classifier.predict([[3, 2]]))

- print(classifier.support_vectors_)
- *# [[7, 5],*
- *# [8, 2],*
- *# [2, 2]]*

- print(classifier.predict([[3, 2]]))





SVM 코드

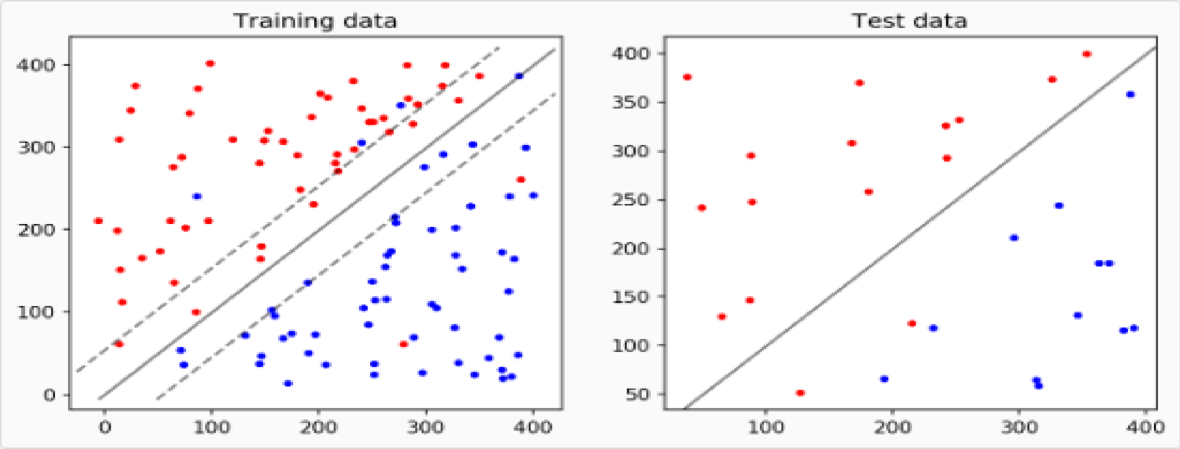
- 2

SVM 코딩

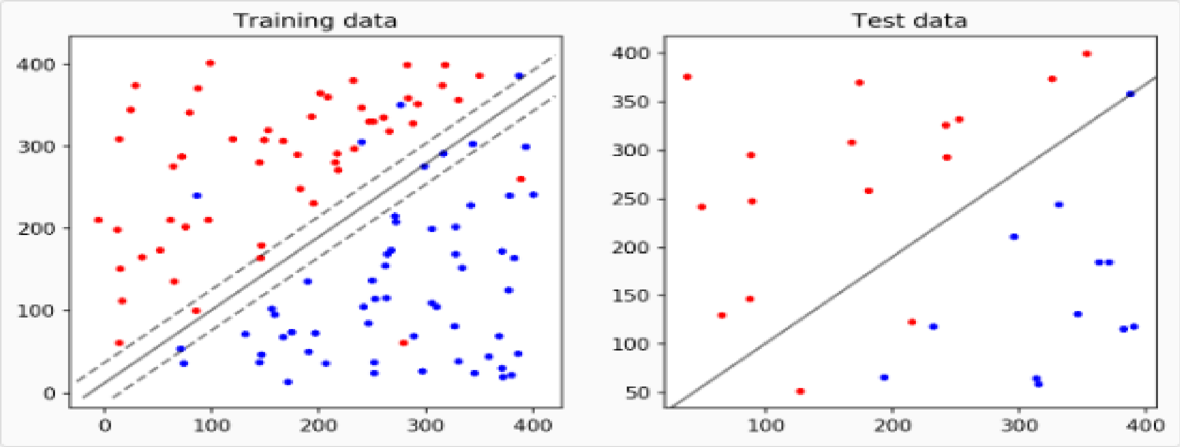
```
# Read data  and split data on 8:2 ratio
x, labels = read_data("points_class_0.txt", "points_class_1.txt")
X_train, X_test, y_train, y_test = train_test_split(x, labels, test_size = 0.2, random_state=0)

print("Displaying data. Close window to continue.")
# Plot data
plot_data(X_train, y_train, X_test, y_test)

# make a classifier and fit on training data
clf = svm.SVC(kernel='linear‘, C=1)
clf_1.fit(X_train, y_train)

print("Display decision function (C=1) ...\n The SVM classifier will choose a large margin decision boundary at the expense of larger number of misclassifications")
# Plot decision function on training and test data
plot_decision_function(X_train, y_train, X_test, y_test, clf_1)

# make a classifier and fit on training data
clf_100 = svm.SVC(kernel='linear', C=100)
clf_100.fit(X_train, y_train)

print("Accuracy(C=1): {}%".format(clf_1.score(X_test, y_test) * 100 ))
print("\n")
print("Display decision function (C=100) ...\nThe classifier will choose a low margin decision boundary and try to minimize the misclassifications")
# Plot decision function on training and test data
plot_decision_function(X_train, y_train, X_test, y_test, clf_100)

print("Accuracy(C=100): {}%".format(clf_100.score(X_test, y_test) * 100 ))

# Make predictions on unseen test data
clf_1_predictions = clf_1.predict(X_test)
clf_100_predictions = clf_100.predict(X_test)
```

SVM 코딩

- *C = 1 / C= 100*



kernel='linear', C = 1



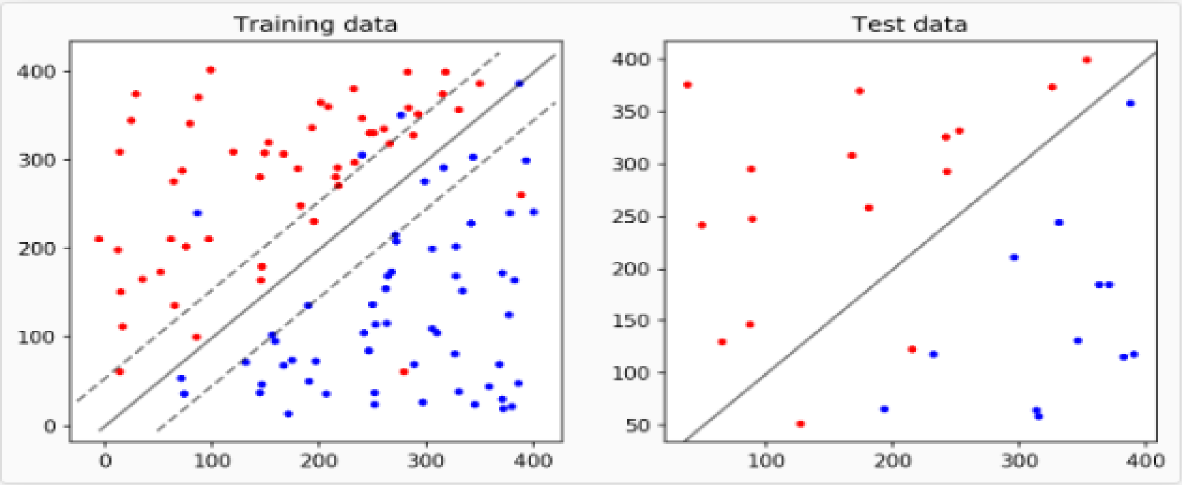
kernel='linear', C = 100

SVM 코딩

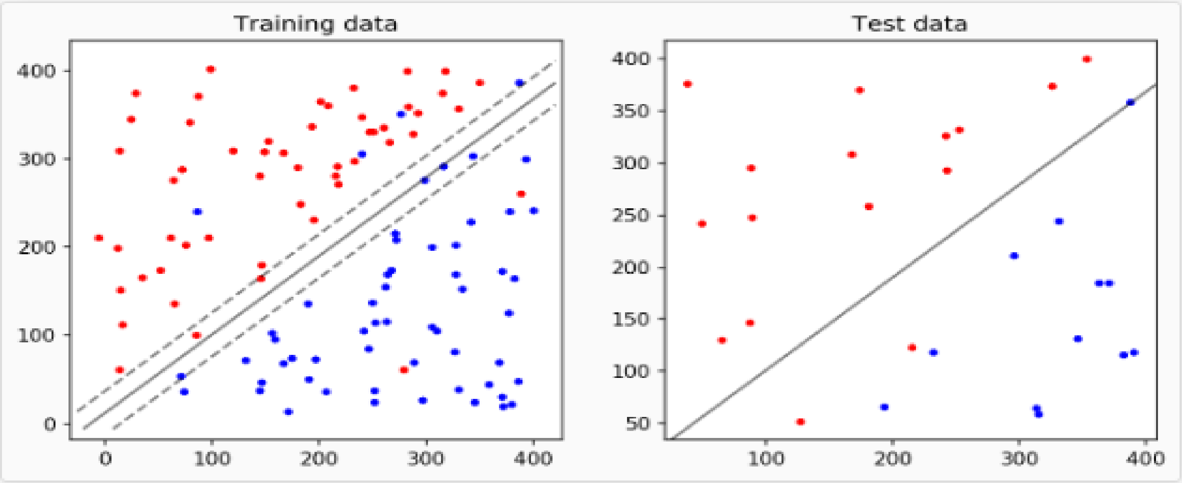
** C의 값은 적절하게 선택*

- *C = 1*
- *training data 분류 : 부정확*
- *Margine : 큼*
- *사용 : Noise 많은곳*

- *C= 100*
- training data 분류 : 정확
- Margine : 작음
- 사용: Noise 적은곳



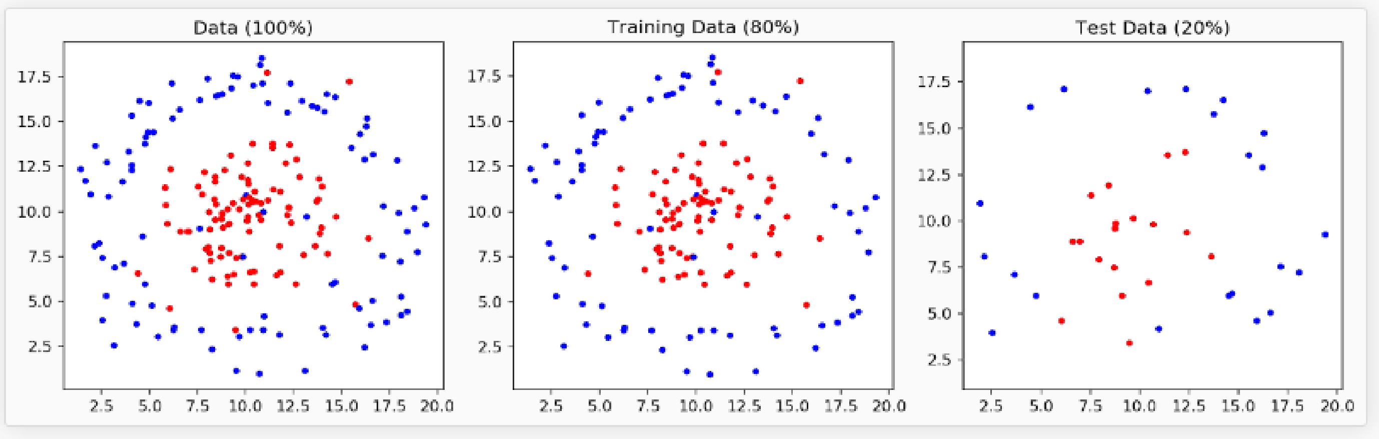
kernel='linear', C = 1



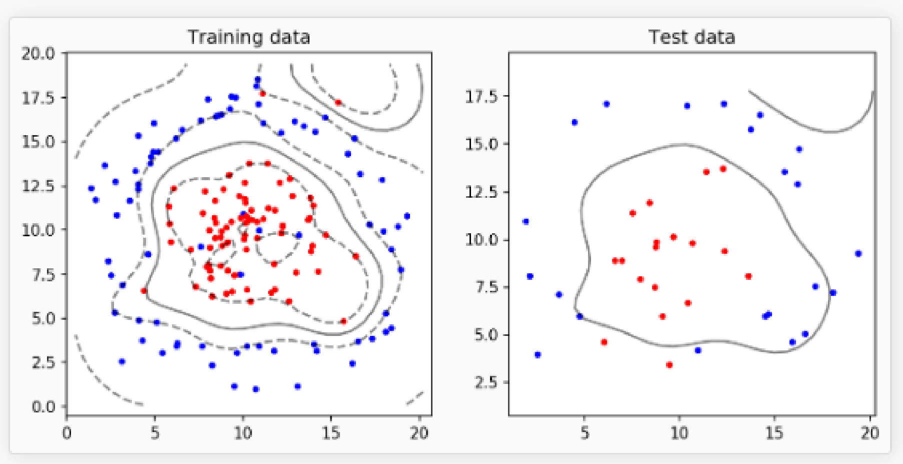
kernel='linear', C = 100

SVM 코딩

- *NonLinearly Separable Data with Noise*



- *Kernel tric 사용*
- clf = svm.SVC(kernel='rbf', C = 10.0, gamma=0.1)



SVM 코딩

- *# Read data & # Split data to train and test on 80-20 ratio*
- x, labels = read_data("points_class_0_nonLinear.txt", "points_class_1_nonLinear.txt")
- X_train, X_test, y_train, y_test = train_test_split(x, labels, test_size = 0.2, random_state=0)

- print("Displaying data.")
- *# Plot data*
- plot_data(X_train, y_train, X_test, y_test)

- print("Training SVM ...")
- *# make a classifier*
- clf = svm.SVC(C = 10.0, kernel='rbf', gamma=0.1)

- *# Train classifier*
- clf.fit(X_train, y_train)
- *# Make predictions on unseen test data*
- clf_predictions = clf.predict(X_test)
- print("Displaying decision function.")
- *# Plot decision function on training and test data*
- plot_decision_function(X_train, y_train, X_test, y_test, clf)

SVM 코딩

- *# Grid Search*
- print("Performing grid search ... ")
- *# Parameter Grid*
- param_grid = {'C': [0.1, 1, 10, 100], 'gamma': [1, 0.1, 0.01, 0.001, 0.00001, 10]}

- *# Make grid search classifier*
- clf_grid = GridSearchCV(svm.SVC(), param_grid, verbose=1)

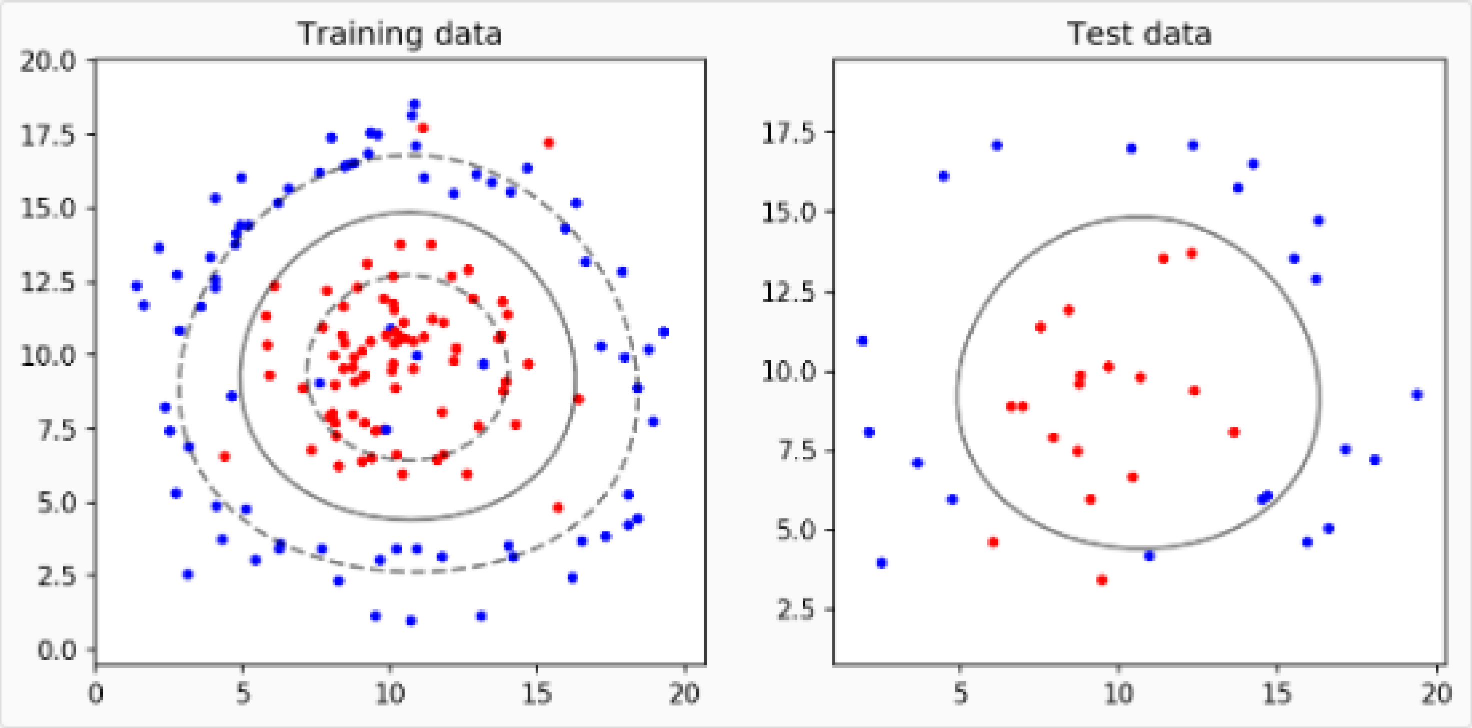
- *# Train the classifier*
- clf_grid.fit(X_train, y_train)

- *# clf = grid.best_estimator_()*
- print("Best Parameters:\n", clf_grid.best_params_)
- print("Best Estimators:\n", clf_grid.best_estimator_)
- print("Displaying decision function for best estimator.")

- *# Plot decision function on training and test data*
- plot_decision_function(X_train, y_train, X_test, y_test, clf_grid)

SVM 코딩

- *최적화*



kernel='rbf', C=1, gamma=0.01

Q & A

감사합니다.