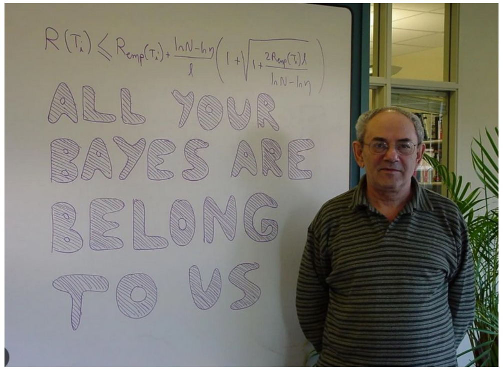
Support Vector Machine

Introduction

- Support Vector Machine → Work in vector space
- Created by Vladimir Vapnik in the 1960s in the USSR! Did not gain much popularity until the last decade of the last century
  - It is still one of the most popular classifiers (Nothing stops us from not using it as a regressor)
- Vapnik moved to Bell Labs in the 1990s where he demonstrated that SVMs outperformed nascent NNs for certain problems
  - But before that he submitted 3 papers in Neurips.
- This resulted in the popularity of SVMs


$R(T_i) \leq R_{emp}(T_i) + \frac{\ln N - \ln \eta}{l}\left(1 + \sqrt{1 + \frac{2R_{emp}(T_i)l}{\ln N - \ln \eta}}\right)$
ALL YOUR
BAYES ARE
BELONG
TO US

From MIT OCW lectures of Prof. Patrick Winston (MIT 6.034 Artificial Intelligence, Fall 2010)

High level introduction

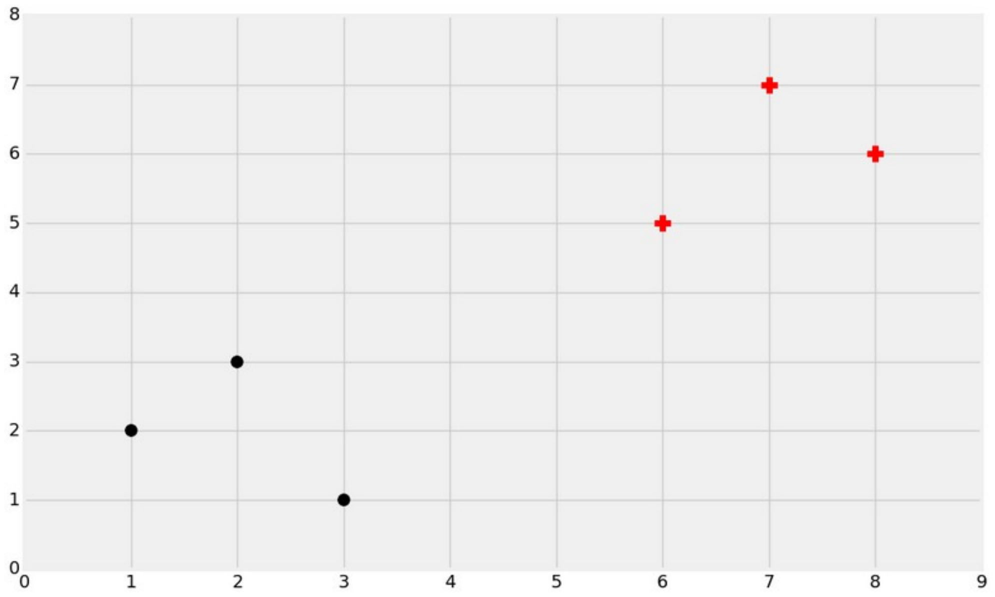
1) Work in vector space 2) Can separate classes into two groups at a time (popularly called positives and negative)

- Goal: Find the hyperplane that best separates these two classes

High level introduction

1) Work in vector space 2) Can separate classes into two groups at a time (popularly called positives and negative)

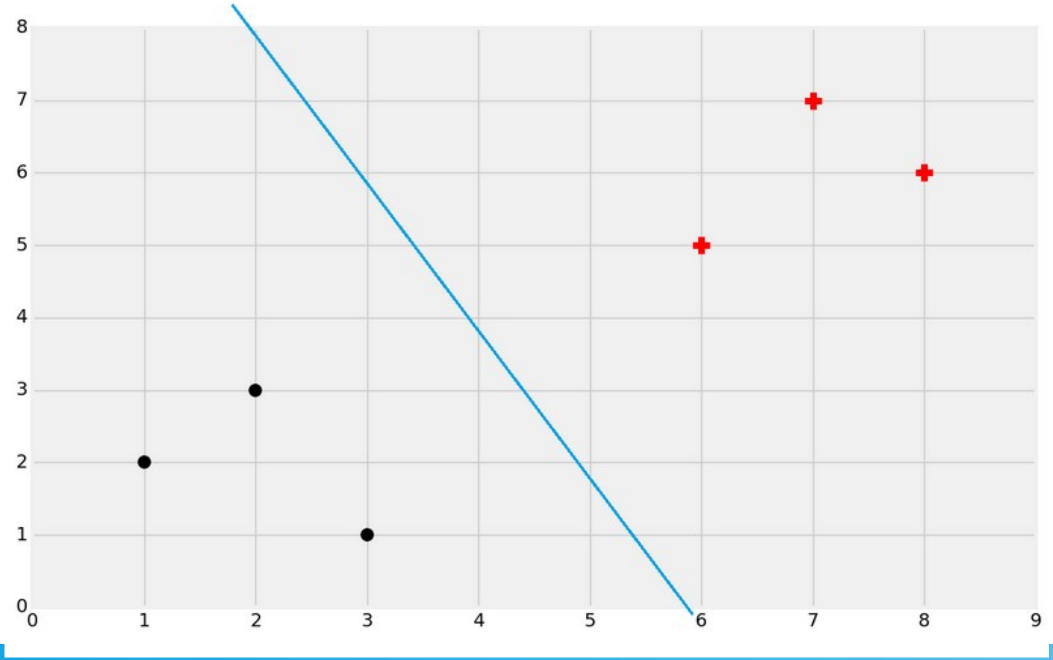
- Goal: Find the hyperplane that best separates these two classes



High level introduction

1) Work in vector space 2) Can separate classes into two groups at a time (popularly called positives and negative)

- Goal: Find the hyperplane that best separates these two classes



Algorithm to derive the hyperplane

On blackboard

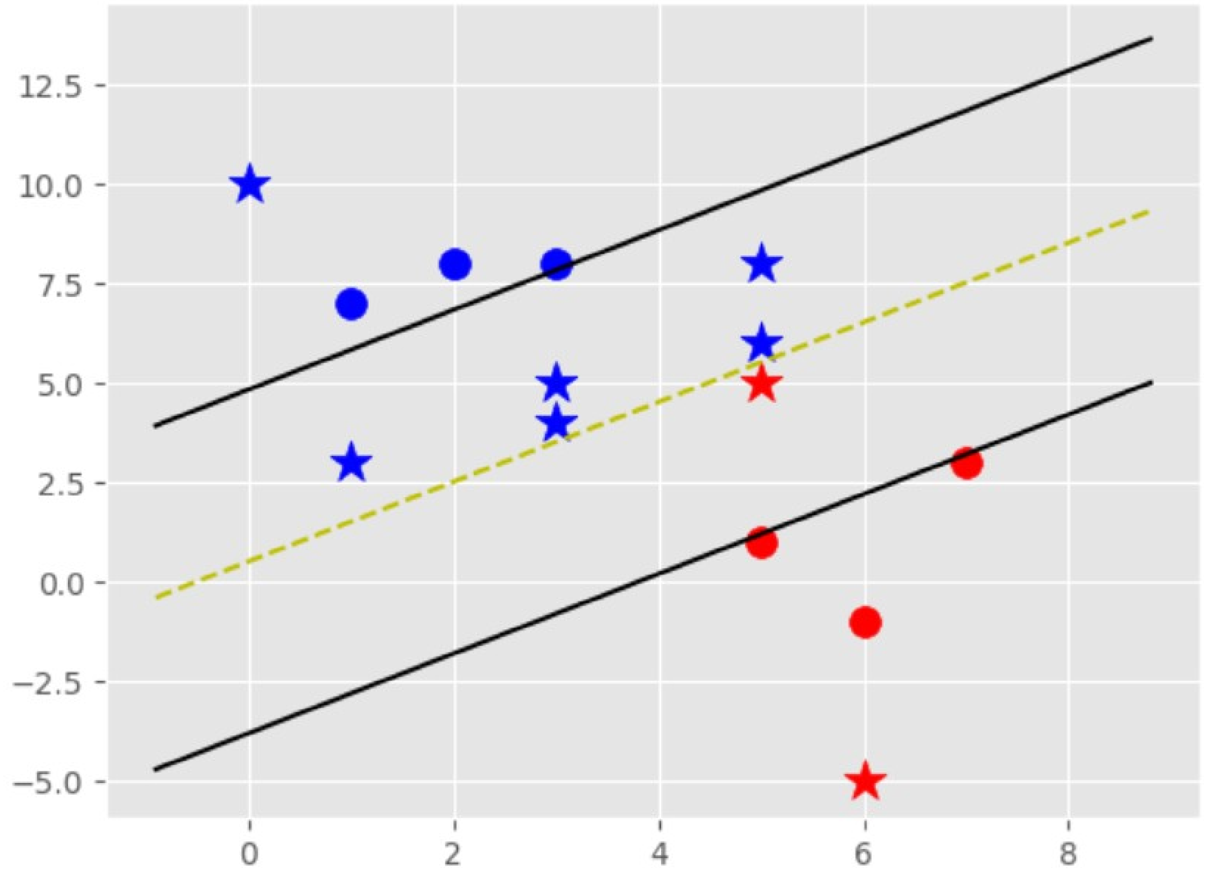
Summary of the problem

- Width = 2/||w||
  - → Maximize width
  - → Minimize 0.5*||w||^2
  - → Subject to the constaint equation Yi(Xi.W+b)-1 = 0

- Exercise : Write out the constraint equation
- Exercise : Write out the constraint optimization problem

Implement a SVM from scratch and visualize

Outcome



Kernels

- Polynomial
- Radial basis function (default)

Soft Margin Support Vector Machine and slack

CVXOPT

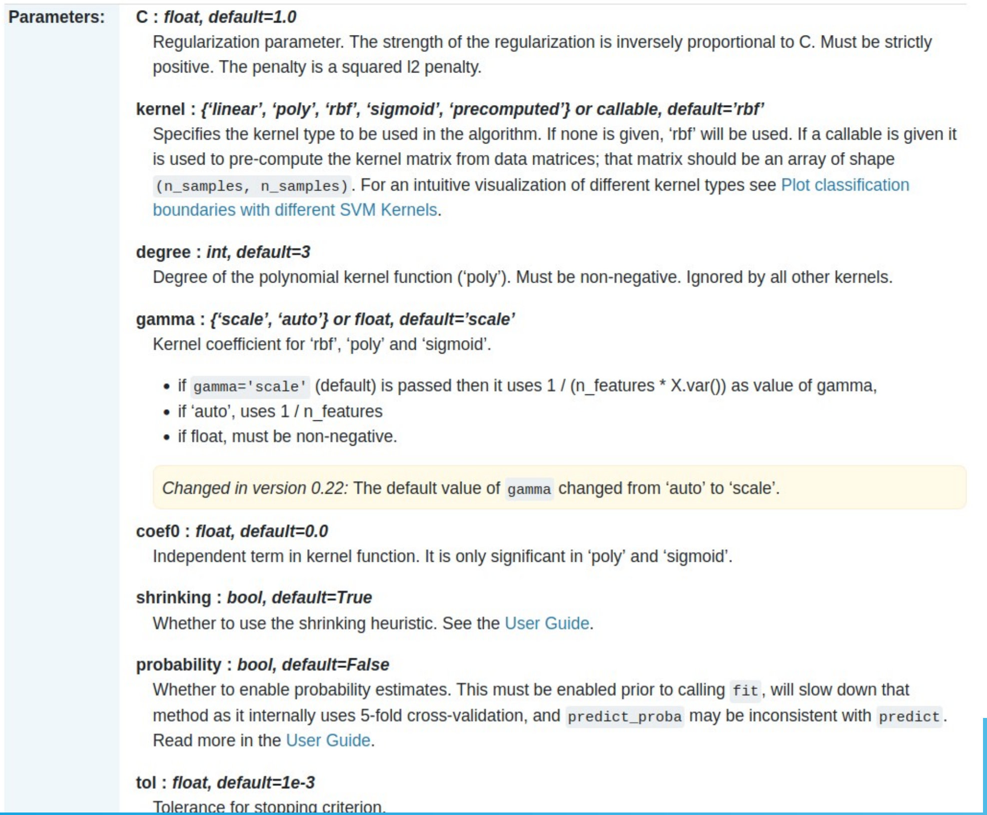
Multiclassification

- One vs Rest

- One vs One

Parameters of the SVM : multiclassification

- https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html



Parameters:

C : *float, default=1.0*
Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.

kernel : *{'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'} or callable, default='rbf'*
Specifies the kernel type to be used in the algorithm. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape `(n_samples, n_samples)`. For an intuitive visualization of different kernel types see Plot classification boundaries with different SVM Kernels.

degree : *int, default=3*
Degree of the polynomial kernel function ('poly'). Must be non-negative. Ignored by all other kernels.

gamma : *{'scale', 'auto'} or float, default='scale'*
Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.

- if `gamma='scale'` (default) is passed then it uses 1 / (n_features * X.var()) as value of gamma,
- if 'auto', uses 1 / n_features
- if float, must be non-negative.

Changed in version 0.22: The default value of `gamma` changed from 'auto' to 'scale'.

coef0 : *float, default=0.0*
Independent term in kernel function. It is only significant in 'poly' and 'sigmoid'.

shrinking : *bool, default=True*
Whether to use the shrinking heuristic. See the User Guide.

probability : *bool, default=False*
Whether to enable probability estimates. This must be enabled prior to calling `fit`, will slow down that method as it internally uses 5-fold cross-validation, and `predict_proba` may be inconsistent with `predict`. Read more in the User Guide.

tol : *float, default=1e-3*
Tolerance for stopping criterion.