Quadratic Programming and Kernel Methods

Jianlin Cheng, PhD

Computer Science Department

University of Missouri, Columbia

Fall, 2014

Reference

- C. Campbell. An Introduction to Kernel Methods
- Reading assignment: read the paper and submit a one page summary. (Due on Nov. 14 – Friday)

Quadratic Optimization

Optimization Approaches?

Kernel Methods

- A systematic and principled approach to training learning machines
- Achieve good generalization performance using statistical learning theory or Bayesian arguments
- Kernel methods for classification, regression and novelty detection
- Optimization of a convex cost function (quadratic programming)
- Applications

Classification Problem

- Diabetes data set: http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary/diabetes
- First **10**, 20, 30, …, 50, or all the data points

Diabetes Data Example

Label

Input Feature Vector

y_i

x_i, <d_1, …, d_8>

$1 <= i <= m$

```
-1  1:6.000000 2:148.000000 3:72.000000 4:35.000000 5:0.000000 6:33.599998 7:0.627000 8:50.000000
+1  1:1.000000 2:85.000000 3:66.000000 4:29.000000 5:0.000000 6:26.600000 7:0.351000 8:31.000000
-1  1:8.000000 2:183.000000 3:64.000000 4:0.000000 5:0.000000 6:23.299999 7:0.672000 8:32.000000
+1  1:1.000000 2:89.000000 3:66.000000 4:23.000000 5:94.000000 6:28.100000 7:0.167000 8:21.000000
-1  1:0.000000 2:137.000000 3:40.000000 4:35.000000 5:168.000000 6:43.099998 7:2.288000 8:33.000000
+1  1:5.000000 2:116.000000 3:74.000000 4:0.000000 5:0.000000 6:25.600000 7:0.201000 8:30.000000
-1  1:3.000000 2:78.000000 3:50.000000 4:32.000000 5:88.000000 6:31.000000 7:0.248000 8:26.000000
+1  1:10.000000 2:115.000000 3:0.000000 4:0.000000 5:0.000000 6:35.299999 7:0.134000 8:29.000000
-1  1:2.000000 2:197.000000 3:70.000000 4:45.000000 5:543.000000 6:30.500000 7:0.158000 8:53.000000
-1  1:8.000000 2:125.000000 3:96.000000 4:0.000000 5:0.000000 6:0.000000 7:0.232000 8:54.000000
+1  1:4.000000 2:110.000000 3:92.000000 4:0.000000 5:0.000000 6:37.599998 7:0.191000 8:30.000000
-1  1:10.000000 2:168.000000 3:74.000000 4:0.000000 5:0.000000 6:38.000000 7:0.537000 8:34.000000
+1  1:10.000000 2:139.000000 3:80.000000 4:0.000000 5:0.000000 6:27.100000 7:1.441000 8:57.000000
-1  1:1.000000 2:189.000000 3:60.000000 4:23.000000 5:846.000000 6:30.100000 7:0.398000 8:59.000000
-1  1:5.000000 2:166.000000 3:72.000000 4:19.000000 5:175.000000 6:25.799999 7:0.587000 8:51.000000
-1  1:7.000000 2:100.000000 3:0.000000 4:0.000000 5:0.000000 6:30.000000 7:0.484000 8:32.000000
-1  1:0.000000 2:118.000000 3:84.000000 4:47.000000 5:230.000000 6:45.799999 7:0.551000 8:31.000000
-1  1:7.000000 2:107.000000 3:74.000000 4:0.000000 5:0.000000 6:29.600000 7:0.254000 8:31.000000
+1  1:1.000000 2:103.000000 3:30.000000 4:38.000000 5:83.000000 6:43.299999 7:0.183000 8:33.000000
-1  1:1.000000 2:115.000000 3:70.000000 4:30.000000 5:96.000000 6:34.599998 7:0.529000 8:32.000000
+1  1:3.000000 2:126.000000 3:88.000000 4:41.000000 5:235.000000 6:39.299999 7:0.704000 8:27.000000
+1  1:8.000000 2:99.000000 3:84.000000 4:0.000000 5:0.000000 6:35.400002 7:0.388000 8:50.000000
-1  1:7.000000 2:196.000000 3:90.000000 4:0.000000 5:0.000000 6:39.799999 7:0.451000 8:41.000000
-1  1:9.000000 2:119.000000 3:80.000000 4:35.000000 5:0.000000 6:29.000000 7:0.263000 8:29.000000
-1  1:11.000000 2:143.000000 3:94.000000 4:33.000000 5:146.000000 6:36.599998 7:0.254000 8:51.000000
-1  1:10.000000 2:125.000000 3:70.000000 4:26.000000 5:115.000000 6:31.100000 7:0.205000 8:41.000000
-1  1:7.000000 2:147.000000 3:76.000000 4:0.000000 5:0.000000 6:39.400002 7:0.257000 8:43.000000
+1  1:1.000000 2:97.000000 3:66.000000 4:15.000000 5:140.000000 6:23.200001 7:0.487000 8:22.000000
+1  1:13.000000 2:145.000000 3:82.000000 4:19.000000 5:110.000000 6:22.200001 7:0.245000 8:57.000000
+1  1:5.000000 2:117.000000 3:92.000000 4:0.000000 5:0.000000 6:34.099998 7:0.337000 8:38.000000
+1  1:5.000000 2:109.000000 3:75.000000 4:26.000000 5:0.000000 6:36.000000 7:0.546000 8:60.000000
-1  1:3.000000 2:158.000000 3:76.000000 4:36.000000 5:245.000000 6:31.600000 7:0.851000 8:28.000000
+1  1:3.000000 2:88.000000 3:58.000000 4:11.000000 5:54.000000 6:24.799999 7:0.267000 8:22.000000
+1  1:6.000000 2:92.000000 3:92.000000 4:0.000000 5:0.000000 6:19.900000 7:0.188000 8:28.000000
+1  1:10.000000 2:122.000000 3:78.000000 4:31.000000 5:0.000000 6:27.600000 7:0.512000 8:45.000000
+1  1:4.000000 2:103.000000 3:60.000000 4:33.000000 5:192.000000 6:24.000000 7:0.966000 8:33.000000
+1  1:11.000000 2:138.000000 3:76.000000 4:0.000000 5:0.000000 6:33.200001 7:0.420000 8:35.000000
-1  1:9.000000 2:102.000000 3:76.000000 4:37.000000 5:0.000000 6:32.900002 7:0.665000 8:46.000000
-1  1:2.000000 2:90.000000 3:68.000000 4:42.000000 5:0.000000 6:38.200001 7:0.503000 8:27.000000
-1  1:4.000000 2:111.000000 3:72.000000 4:47.000000 5:207.000000 6:37.099998 7:1.390000 8:56.000000
+1  1:3.000000 2:180.000000 3:64.000000 4:25.000000 5:70.000000 6:34.000000 7:0.271000 8:26.000000
+1  1:7.000000 2:133.000000 3:84.000000 4:0.000000 5:0.000000 6:40.200001 7:0.696000 8:37.000000
+1  1:7.000000 2:106.000000 3:92.000000 4:18.000000 5:0.000000 6:22.700001 7:0.235000 8:48.000000
-1  1:9.000000 2:171.000000 3:110.000000 4:24.000000 5:240.000000 6:45.400002 7:0.721000 8:54.000000
+1  1:7.000000 2:159.000000 3:64.000000 4:0.000000 5:0.000000 6:27.400000 7:0.294000 8:40.000000
-1  1:0.000000 2:180.000000 3:66.000000 4:39.000000 5:0.000000 6:42.000000 7:1.893000 8:25.000000
+1  1:1.000000 2:146.000000 3:56.000000 4:0.000000 5:0.000000 6:29.700001 7:0.564000 8:29.000000
+1  1:2.000000 2:71.000000 3:70.000000 4:27.000000 5:0.000000 6:28.000000 7:0.586000 8:22.000000
-1  1:7.000000 2:103.000000 3:66.000000 4:32.000000 5:0.000000 6:39.099998 7:0.344000 8:31.000000
+1  1:7.000000 2:105.000000 3:0.000000 4:0.000000 5:0.000000 6:0.000000 7:0.305000 8:24.000000
+1  1:1.000000 2:103.000000 3:80.000000 4:11.000000 5:82.000000 6:19.400000 7:0.491000 8:22.000000
+1  1:1.000000 2:101.000000 3:50.000000 4:15.000000 5:36.000000 6:24.200001 7:0.526000 8:26.000000
+1  1:5.000000 2:88.000000 3:66.000000 4:21.000000 5:23.000000 6:24.400000 7:0.342000 8:30.000000
```

A Classification Learning Task

The learning task. Let us consider a binary classification task with datapoints $\mathbf{x}_i$ $(i = 1, \ldots, m)$ having corresponding labels $y_i = \pm 1$ and let the decision function be:

$$f(\mathbf{x}) = \text{sign}\,(\mathbf{w} \cdot \mathbf{x} + b) \tag{1}$$

If the dataset is separable then the data will be correctly classified if $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) > 0\ \forall i$. Clearly this relation is invariant under a positive rescaling of the argument inside the $sign$-function, hence we can define a *canonical hyperplane* such that $\mathbf{w} \cdot \mathbf{x} + b = 1$ for the closest points on one side and $\mathbf{w} \cdot \mathbf{x} + b = -1$ for the closest on the other. For the separating

Visualization of Binary Classification

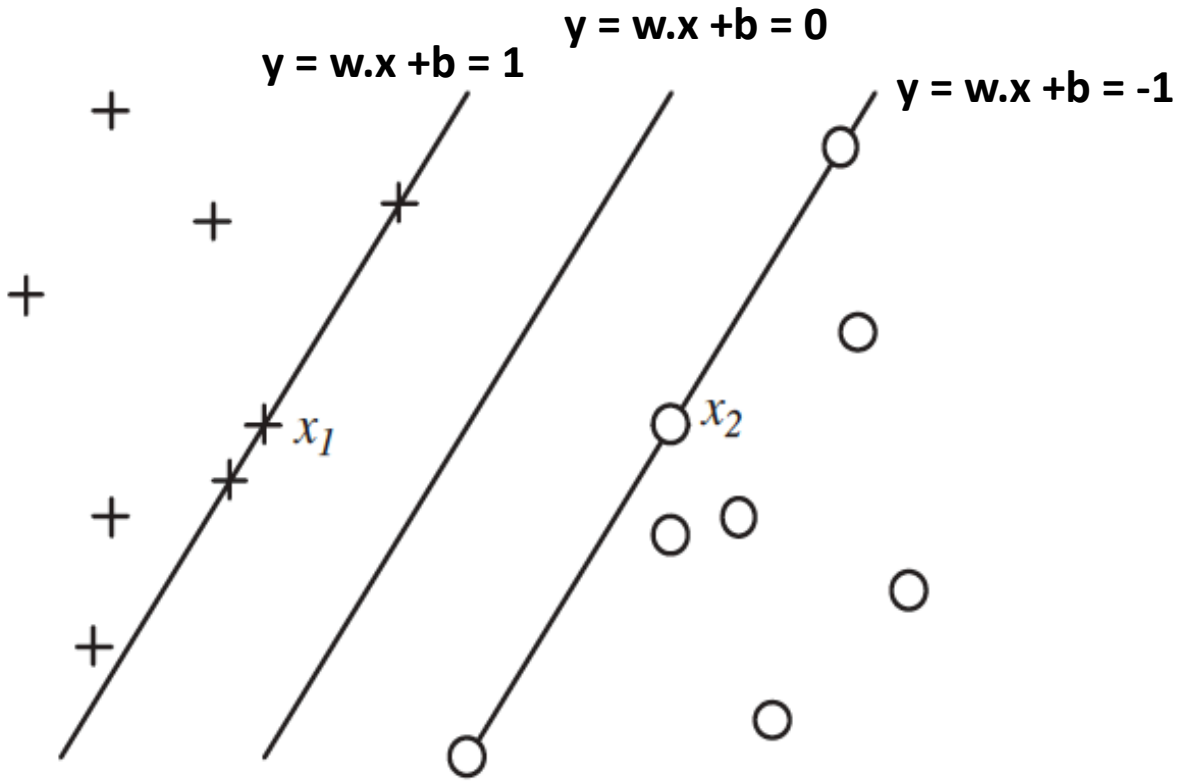


Figure 1. The *margin* is the perpendicular distance between the separating hyperplane and a hyperplane through the closest points (these are *support vectors*). The region between the hyperplanes on each side is called the *margin band*. $\mathbf{x}_1$ and $\mathbf{x}_2$ are examples of support vectors of opposite sign.

Visualization of Binary Classification

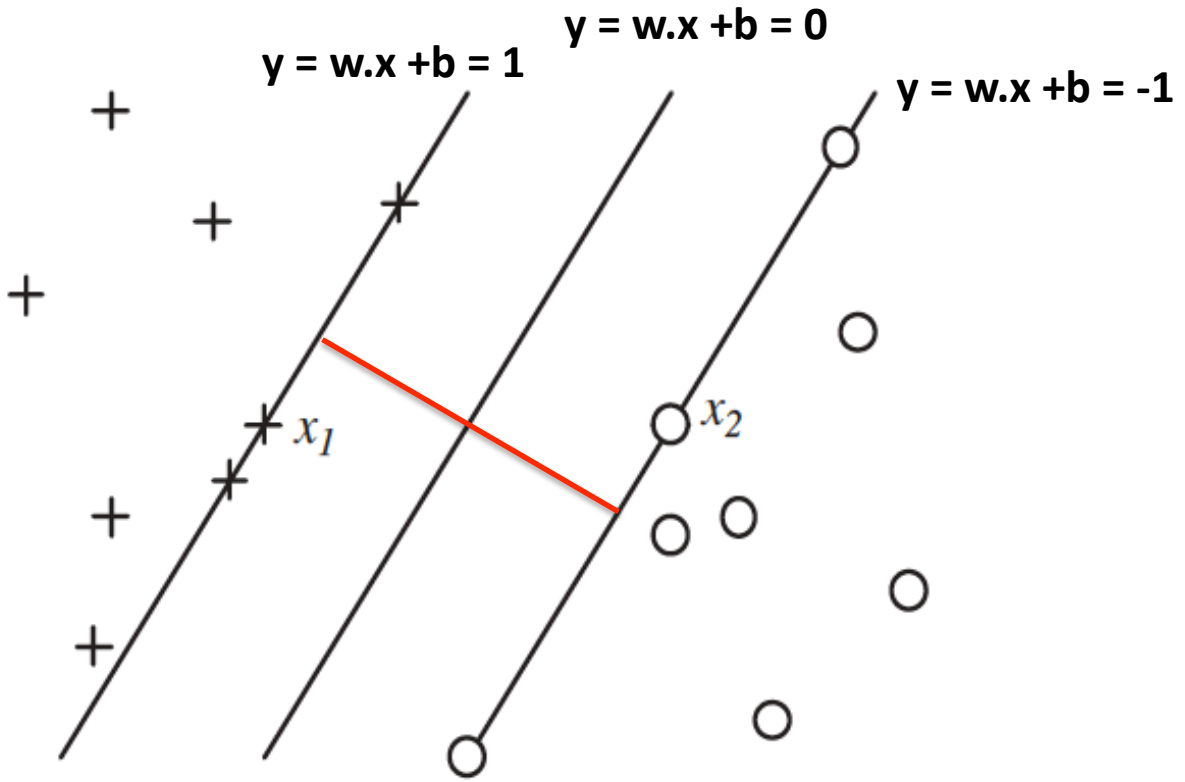


Figure 1. The *margin* is the perpendicular distance between the separating hyperplane and a hyperplane through the closest points (these are *support vectors*). The region between the hyperplanes on each side is called the *margin band*. $\mathbf{x}_1$ and $\mathbf{x}_2$ are examples of support vectors of opposite sign.

Visualization of Binary Classification

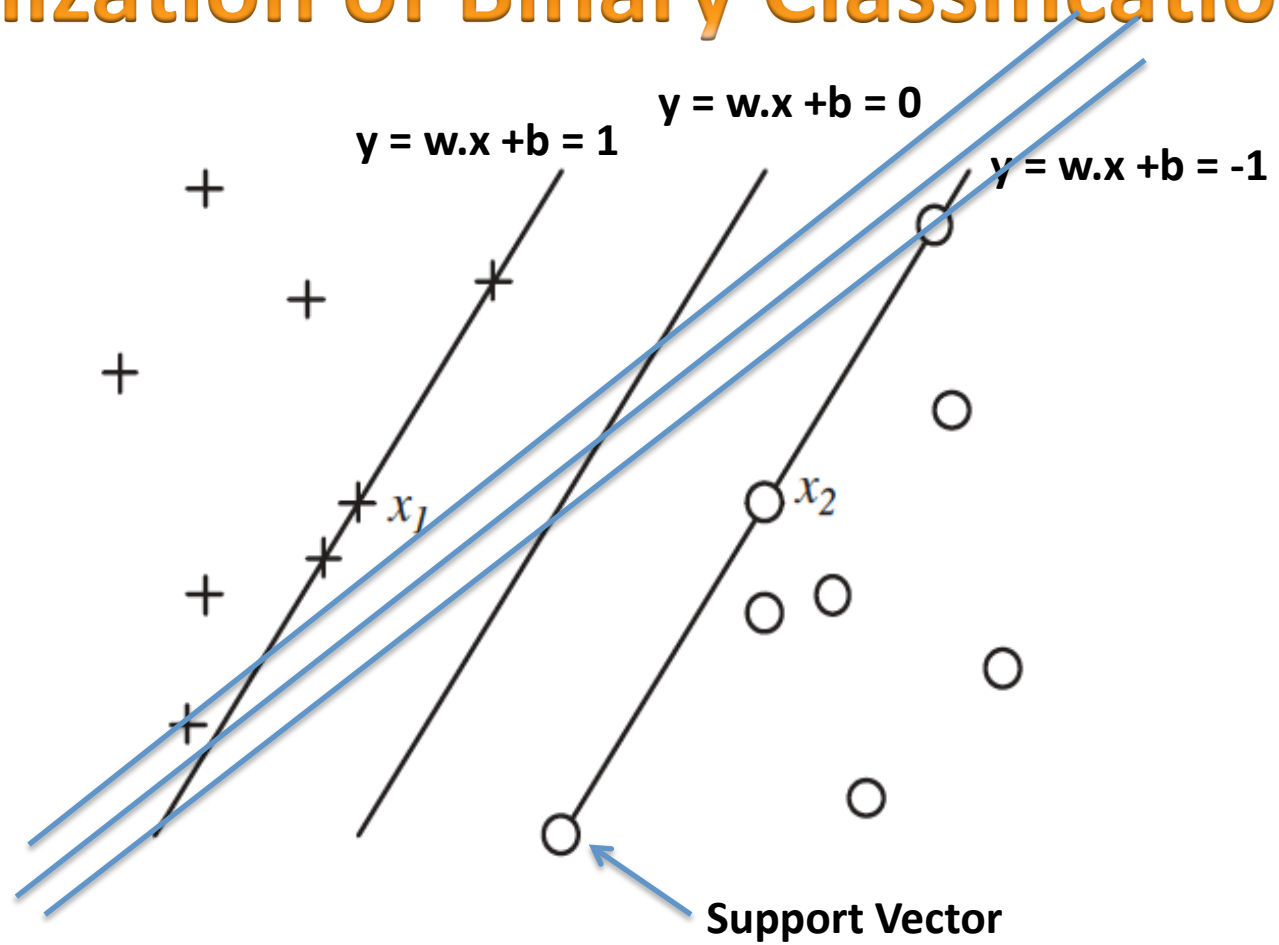


Figure 1. The *margin* is the perpendicular distance between the separating hyperplane and a hyperplane through the closest points (these are *support vectors*). The region between the hyperplanes on each side is called the *margin band*. $\mathbf{x}_1$ and $\mathbf{x}_2$ are examples of support vectors of opposite sign.

Visualization of Binary Classification

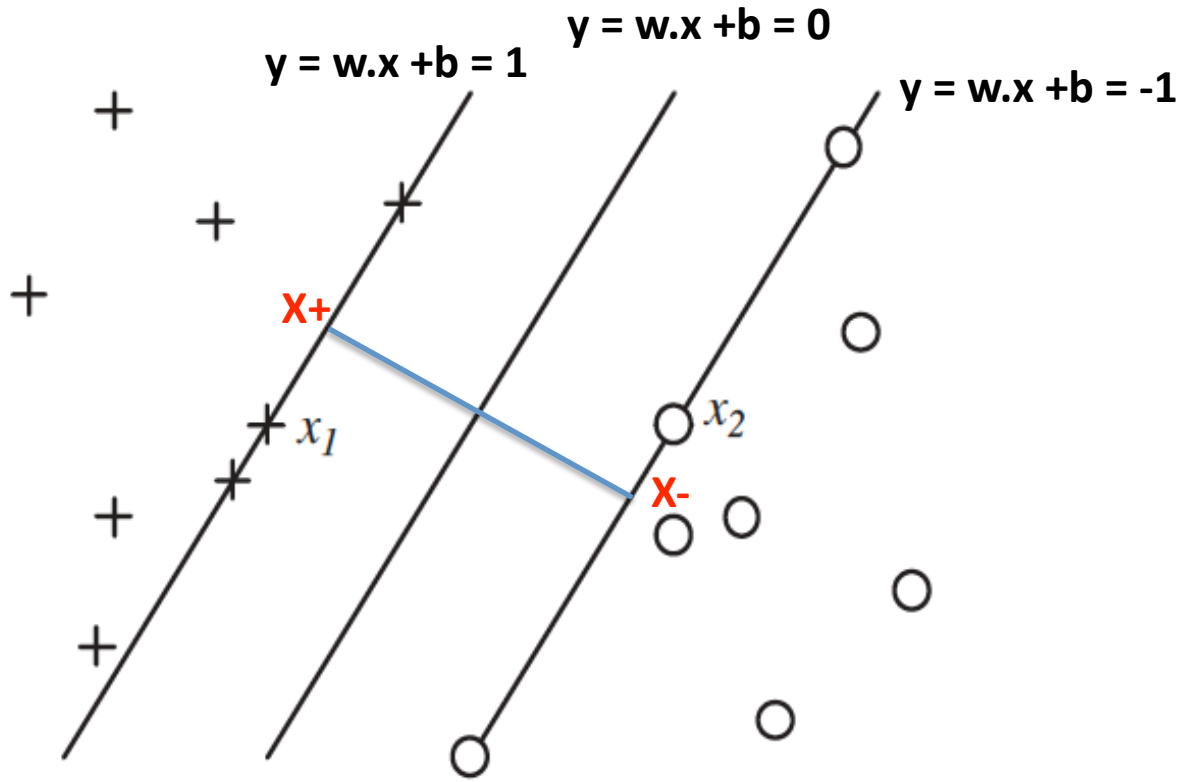


Figure 1. The *margin* is the perpendicular distance between the separating hyperplane and a hyperplane through the closest points (these are *support vectors*). The region between the hyperplanes on each side is called the *margin band*. $\mathbf{x}_1$ and $\mathbf{x}_2$ are examples of support vectors of opposite sign.

Maximize Margin

If the dataset is separable then the data will be correctly classified if $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) > 0 \; \forall i$. Clearly this relation is invariant under a positive rescaling of the argument inside the $sign$-function, hence we can define a *canonical hyperplane* such that $\mathbf{w} \cdot \mathbf{x} + b = 1$ for the closest points on one side and $\mathbf{w} \cdot \mathbf{x} + b = -1$ for the closest on the other. For the separating hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 0$ the normal vector is clearly $\mathbf{w} / ||\mathbf{w}||$. Hence the margin is given by the projection of $\mathbf{x}_1 - \mathbf{x}_2$ onto this vector where $\mathbf{x}_1$ and $\mathbf{x}_2$ are closest points on opposite sides of the separating hyperplane (see Figure 1). Since $\mathbf{w} \cdot \mathbf{x}_1 + b = 1$ and $\mathbf{w} \cdot \mathbf{x}_2 + b = -1$ this means the margin is $\gamma = 1 / ||\mathbf{w}||$. To maximize the margin, the task is therefore:

$$\min \left[\frac{1}{2} ||\mathbf{w}||^2 \right] \tag{2}$$

subject to the constraints:

$$y_i \left(\mathbf{w} \cdot \mathbf{x}_i + b \right) \geq 1 \qquad \forall i \tag{3}$$

Minimize Weights is Equivalent to Reducing the Function Complexity – Bayesian Perspective

- Smaller weights make function simpler
- Smaller weights make function smoother
- Pursuing smaller weights become a general paradigm in function optimization

Minimization of Primal Objective Function

$$L = \frac{1}{2}(\mathbf{w} \cdot \mathbf{w}) - \sum_{i=1}^{m} \alpha_i \left(y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1\right) \tag{4}$$

where α_i are Lagrange multipliers (hence $\alpha_i \geq 0$). Taking the derivatives with respect to b and $\mathbf{w}$ gives:

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \tag{5}$$

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i \tag{6}$$

Maximization of Dual Objective Function

$$W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right) \tag{7}$$

which must be maximized with respect to the α_i subject to the constraint:

$$\alpha_i \geq 0 \qquad \sum_{i=1}^{m} \alpha_i y_i = 0 \tag{8}$$

This is a constrained quadratic optimization problem.

Questions: what if there are only two examples (one positive x^+ and one negative x^-)?

Karush-Kuhn-Tucker (KKT) Condition

- $w = \Sigma a_i y_i x_i$. (1)
- $\Sigma a_i y_i = 0$ (2)
- $y_i(x_i.w+b) - 1 >= 0$, $i = 1, \ldots, l$ (3)
- $a_i >= 0$ (4)
- $\boldsymbol{a_i(y_i(w.x_i+b)-1) = 0}$ (5)

(5) is called complementary slackness due to the Lagrange theory and can be explained in intuition.

Quadratic Programming Solver

Name	Brief info
AIMMS	
AMPL	A popular modeling language for large-scale mathematical optimization.
APMonitor	
CPLEX	Popular solver with an API (C,C++,Java,.Net, Python, Matlab and R). Free for academics.
EXCEL Solver Function	
GAMS	
Gurobi	Solver with parallel algorithms for large-scale linear programs, quadratic programs and mixed-integer programs. Free for academic use.
IMSL	A set of mathematical and statistical functions that programmers can embed into their software applications.
Maple	General-purpose programming language for mathematics. Solving a quadratic problem in Maple is accomplished via its QPSolve command.
MATLAB	A general-purpose and matrix-oriented programming-language for numerical computing. Quadratic programming in MATLAB requires the Optimization Toolbox in addition to the base MATLAB product
Mathematica	A general-purpose programming-language for mathematics, including symbolic and numerical capabilities.
MOSEK	A solver for large scale optimization with API for several languages (C++,java,.net, Matlab and python)
NAG Numerical Library	A collection of mathematical and statistical routines developed by the Numerical Algorithms Group for multiple programming languages (C, C++, Fortran, Visual Basic, Java and C#) and packages (MATLAB, Excel, R, LabVIEW). The Optimization chapter of the NAG Library includes routines for quadratic programming problems with both sparse and non-sparse linear constraint matrices, together with routines for the optimization of linear, nonlinear, sums of squares of linear or nonlinear functions with nonlinear, bounded or no constraints. The NAG Library has routines for both local and global optimization, and for continuous or integer problems.
OpenOpt	BSD licensed universal cross-platform numerical optimization framework, see its QP page and other problems involved. Uses NumPy arrays and SciPy sparse matrices.
OptimJ	Free Java-based Modeling Language for Optimization supporting multiple target solvers and available as an Eclipse plugin.[9][10]
TOMLAB	Supports global optimization, integer programming, all types of least squares, linear, quadratic and unconstrained programming for MATLAB. TOMLAB supports solvers like Gurobi, CPLEX, SNOPT and KNITRO.

Kernel Substitution

$$W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right)$$

Kernel substitution. This constrained quadratic programming (QP) problem will give an optimal separating hyperplane with a maximal margin if the data is separable. However, we have still not exploited the second observation from theorem 1: the error bound does not depend on the dimension of the space. This feature enables us to give an alternative kernel representation of the data which is equivalent to a mapping into a high dimensional space where the two classes of data are more readily separable. This space is called *feature space* and must be a pre-Hilbert or inner product space. For the dual objective function in (7) we notice that the datapoints, $\mathbf{x}_i$, only appear inside an inner product. Thus the mapping is achieved through a replacement of the inner product:

$$\mathbf{x}_i \cdot \mathbf{x}_j \rightarrow \phi\left(\mathbf{x}_i\right) \cdot \phi(\mathbf{x}_j) \tag{9}$$

The functional form of the mapping $\phi(\mathbf{x}_i)$ does not need to be known since it is implicitly defined by the choice of *kernel:*

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) \tag{10}$$

Common Kernels

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma^2} \tag{11}$$

Other choices of kernel are possible, e.g.:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^d \qquad K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\beta \mathbf{x}_i \cdot \mathbf{x}_j + b) \tag{12}$$

Kernel-Based Quadratic Optimization

$$W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{13}$$

subject to the constraints of Equation (8). The associated *Karush-Kuhn-Tucker* (KKT) conditions are:

$$\begin{aligned} y_i \left(\mathbf{w} \cdot \mathbf{x}_i + b\right) - 1 &\geq 0 \qquad \forall i \\ \alpha_i &\geq 0 \qquad \forall i \\ \alpha_i \left(y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1\right) &= 0 \qquad \forall i \end{aligned} \tag{14}$$

which are always satisfied when a solution is found. Test examples are evaluated using a decision function given by the sign of:

$$f(\mathbf{z}) = \sum_{i=1}^{m} y_i \alpha_i K(\mathbf{x}_i, \mathbf{z}) + b \tag{15}$$

Can weight vector w be explicitly represented?

Allow for Training Error

a positive slack variable ξ_i:

$$y_i \left(\mathbf{w} \cdot \mathbf{x}_i + b \right) \geq 1 - \xi_i \tag{20}$$

and the task is now to minimize the sum of errors $\sum_{i=1}^{m} \xi_i$ in addition to $||\mathbf{w}||^2$:

$$\min \left[\frac{1}{2} \mathbf{w} \cdot \mathbf{w} + C \sum_{i=1}^{m} \xi_i \right] \tag{21}$$

This is readily formulated as a primal objective function:

$$\begin{aligned} L(\mathbf{w}, b, \alpha, \xi) \quad = \quad & \frac{1}{2} \mathbf{w} \cdot \mathbf{w} + C \sum_{i=1}^{m} \xi_i \\ & - \sum_{i=1}^{m} \alpha_i \left[y_i \left(\mathbf{w} \cdot \mathbf{x}_i + b \right) - 1 + \xi_i \right] - \sum_{i=1}^{m} r_i \xi_i \end{aligned} \tag{22}$$

Quadratic Optimization Function Tolerating Training Errors

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i = 0 \tag{23}$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{m} \alpha_i y_i = 0 \tag{24}$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - r_i = 0 \tag{25}$$

$$W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)$$

KKT Conditions

$$1.\partial_{\mathbf{w}}\mathcal{L}_P = 0 \;\rightarrow\; \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x}_i = 0$$

$$2.\partial_b \mathcal{L}_P = 0 \;\rightarrow\; \sum_i \alpha_i y_i = 0$$

$$3.\partial_\xi \mathcal{L}_P = 0 \;\rightarrow\; C - \alpha_i - \mu_i = 0$$

$$4.\text{constraint-1} \quad y_i(\mathbf{w}^T\mathbf{x}_i - b) - 1 + \xi_i \geq 0$$

$$5.\text{constraint-2} \quad \xi_i \geq 0$$

$$6.\text{multiplier condition-1} \quad \alpha_i \geq 0$$

$$7.\text{multiplier condition-2} \quad \mu_i \geq 0$$

$$8.\text{complementary slackness-1} \quad \alpha_i \left[y_i(\mathbf{w}^T\mathbf{x}_i - b) - 1 + \xi_i\right] = 0$$

$$9.\text{complementary slackness-1} \quad \mu_i \xi_i = 0$$

How can we identify the data points needing slackness?

Max Welling, 2005

Novelty Detection

- Novelty detection: identification of new or unknown data that a machine learning system has not been trained with and was not previously aware of.
- Application: detection of a disease or potential fault whose class may be under-represented in the training data.

An Novelty Detection Approach

- Find a hypersphere with a minimal radius R and center a which contains most of the data: novel test points lie outside the boundary of this hypersphere.
- The effect of outliers is reduced by using slack variables ξ_i to allow for data points outside the sphere and the task is to minimize the volume of the sphere and number of data points outside.

Minimization of Objective

$$\min \left[R^2 + \frac{1}{m\nu} \sum_i \xi_i \right]$$

subject to the constraints:

$$(\mathbf{x}_i - \mathbf{a})^T (\mathbf{x}_i - \mathbf{a}) \leq R^2 + \xi_i$$

Primal and Dual Objective Function

primal objective function is then:

$$L(R, \mathbf{a}, \alpha_i, \xi_i) = R^2 + \frac{1}{m\nu}\sum_{i=1}^{m}\xi_i$$
$$-\sum_{i=1}^{m}\alpha_i\left(R^2 + \xi_i - (\mathbf{x}_i \cdot \mathbf{x}_i - 2\mathbf{a}\cdot\mathbf{x}_i + \mathbf{a}\cdot\mathbf{a})\right) - \sum_{i=1}^{m}\gamma_i\xi_i \quad (28)$$

with $\alpha_i \geq 0$ and $\gamma_i \geq 0$. After kernel substitution the dual formulation amounts to maximization of:

$$W(\alpha) = \sum_{i=1}^{m}\alpha_i K(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j=1}^{m}\alpha_i\alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \quad (29)$$

Outlier Detection Function

with respect to α_i and subject to $\sum_{i=1}^{m} \alpha_i = 1$ and $0 \leq \alpha_i \leq 1/m\nu$. If $m\nu > 1$ then *at bound* examples will occur with $\alpha_i = 1/m\nu$ and these correspond to outliers in the training process. Having completed the training process a test point $\mathbf{z}$ is declared novel if:

$$K(\mathbf{z}, \mathbf{z}) - 2\sum_{i=1}^{m} \alpha_i K(\mathbf{z}, \mathbf{x}_i) + \sum_{i,j=1}^{m} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) - R^2 \geq 0 \qquad (30)$$

Algorithms to Train SVM

For classification, regression or novelty detection we see that the learning task involves optimization of a quadratic cost function and thus techniques from quadratic programming are most applicable including quasi-Newton, conjugate gradient and primal-dual interior point methods. Certain QP packages are readily applicable such as MINOS and LOQO. These methods can be used to train an SVM rapidly but they have the disadvantage that the kernel matrix is stored in memory. For small datasets this is practical and QP routines are the best choice, but for larger datasets alternative techniques have to be used. These split into two categories:

Gradient Descent

algorithm [15]. For binary classification (with no soft margin or bias) this is a simple gradient ascent procedure on (13) in which $\alpha_i > 0$ initially and the α_i are subsequently sequentially updated using:

$$\alpha_i \leftarrow \beta_i \theta(\beta_i) \quad \text{where } \beta_i = \alpha_i + \eta \left(1 - y_i \sum_{j=1}^{m} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j)\right) \tag{54}$$

and $\theta(\beta)$ is the Heaviside step function. The optimal learning rate η can be readily evaluated: $\eta = 1/K(\mathbf{x}_i, \mathbf{x}_i)$ and a sufficient condition for convergence is $0 < \eta K(\mathbf{x}_i, \mathbf{x}_i) < 2$. With the given decision function of

$$W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)$$

Decomposition and Sequential Minimal Optimization

Decomposition method provide a better approach: only use a fixed size subset of data, with a_i for the remainder for fixed.

The limiting case of decomposition is the Sequential Minimal Optimization (SMO) algorithm of Platt [33] in which only two α_i are optimized at each iteration. The smallest set of parameters which can be optimized with each iteration is plainly two if the constraint $\sum_{i=1}^{m} \alpha_i y_i = 0$ is to hold. Remarkably, if only two parameters are optimized and the rest kept fixed then it is possible to derive an analytical solution which can be executed using few numerical operations. The algorithm therefore selects two La-

SMO Algorithm

Consider a binary classification problem with a dataset $(x_1, y_1), \ldots, (x_n, y_n)$, where x_i is an input vector and $y_i \in \{-1, +1\}$ is a binary label corresponding to it. A soft-margin support vector machine is trained by solving a quadratic programming problem, which is expressed in the dual form as follows:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j K(x_i, x_j) \alpha_i \alpha_j,$$

subject to:

$$0 \leq \alpha_i \leq C, \quad \text{for } i = 1, 2, \ldots, n,$$

$$\sum_{i=1}^{n} y_i \alpha_i = 0$$

where C is an SVM hyperparameter and $K(x_i, x_j)$ is the kernel function, both supplied by the user; and the variables α_i are Lagrange multipliers.

Algorithm

SMO is an iterative algorithm for solving the optimization problem described above. SMO breaks this problem into a series of smallest possible sub-problems, which are then solved analytically. Because of the linear equality constraint involving the Lagrange multipliers α_i, the smallest possible problem involves two such multipliers. Then, for any two multipliers α_1 and α_2, the constraints are reduced to:

$$0 \leq \alpha_1, \alpha_2 \leq C,$$
$$y_1\alpha_1 + y_2\alpha_2 = k,$$

and this reduced problem can be solved analytically: one needs to find a minimum of a one-dimensional quadratic function. k is the sum over the rest of terms in the equality constraint, which is fixed in each iteration.

The algorithm proceeds as follows:

1. Find a Lagrange multiplier α_1 that violates the Karush–Kuhn–Tucker (KKT) conditions for the optimization problem.
2. Pick a second multiplier α_2 and optimize the pair (α_1, α_2).
3. Repeat steps 1 and 2 until convergence.

When all the Lagrange multipliers satisfy the KKT conditions (within a user-defined tolerance), the problem has been solved. Although this algorithm is guaranteed to converge, heuristics are used to choose the pair of multipliers so as to accelerate the rate of convergence.

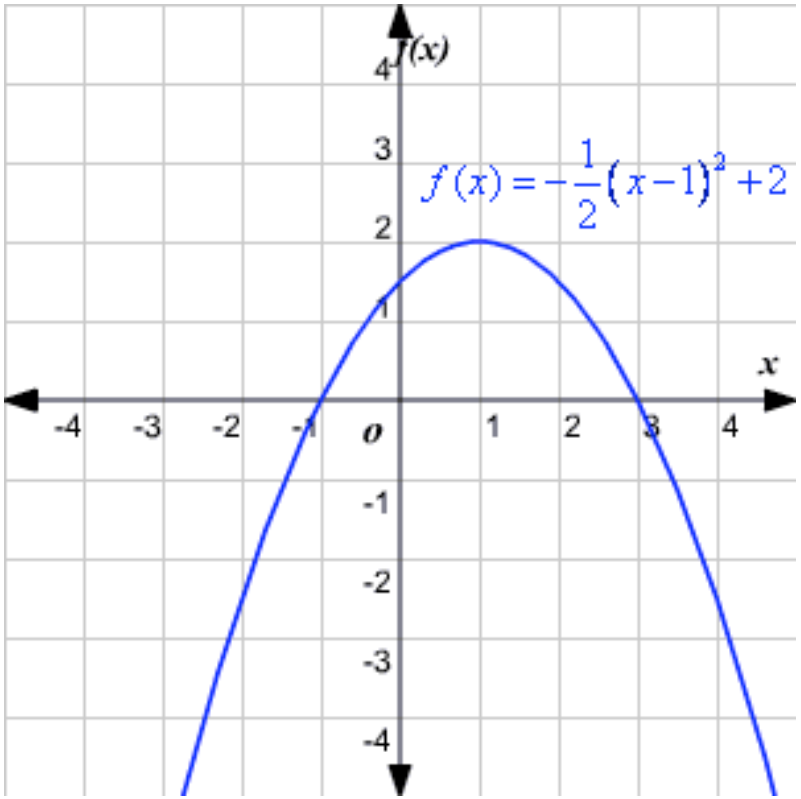
$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j K(x_i, x_j) \alpha_i \alpha_j,$$

$$0 \leq \alpha_1, \alpha_2 \leq C,$$
$$y_1 \alpha_1 + y_2 \alpha_2 = k$$

Keep all other terms except a_1, a_2 fixed, get a new function like

$a_1 + (k-a_1) +$ other_as $- \frac{1}{2} (y_1 y_1 a_1 a_1 K(x_1,x_1) + y_2 y_2 a_2 a_2 K(x_2,x_2) + 2 y_1 y_2 a_1 (k-a_1) K(x_1,x_2) + y_1 a_1 *$Other $+ (k - y_1 a_1) *$other + other terms)

Optimize Single Variable Quadratic Function



Check three values:

0, C, and x-coordinate of vertex