

2018 CS420 Machine Learning, Lecture 3

Hangout from Prof. Andrew Ng. <http://cs229.stanford.edu/notes/cs229-notes3.pdf>

Support Vector Machines and Kernel Methods

Weinan Zhang

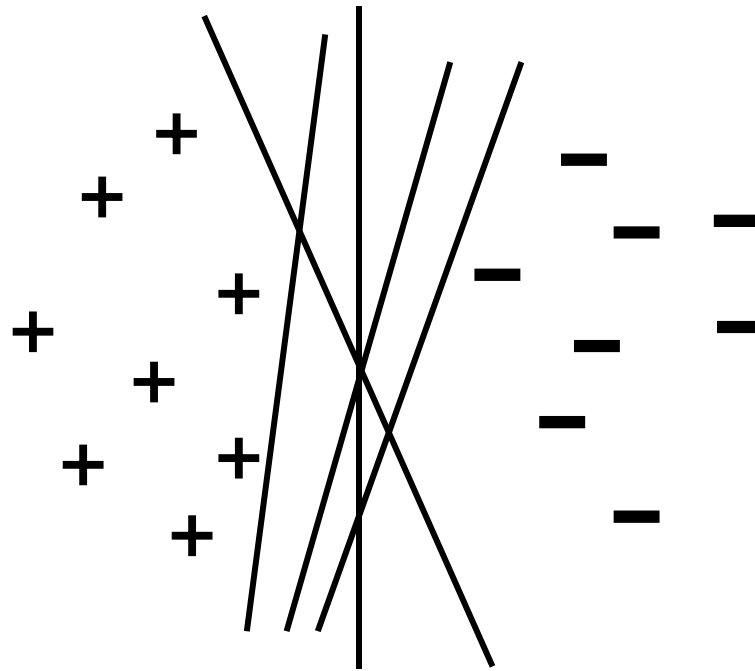
Shanghai Jiao Tong University

<http://wnzhang.net>

<http://wnzhang.net/teaching/cs420/index.html>

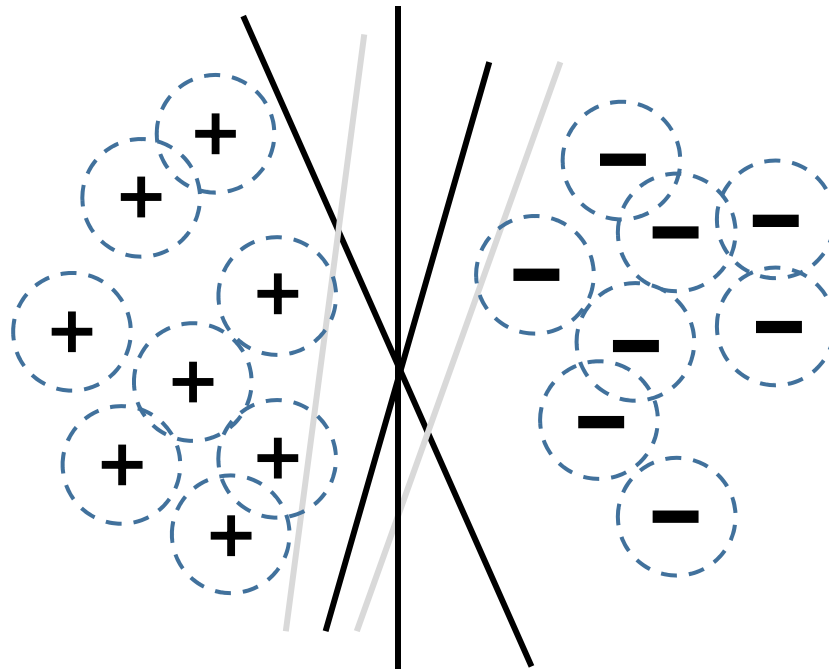
Linear Classification

- For linear separable cases, we have multiple decision boundaries



Linear Classification

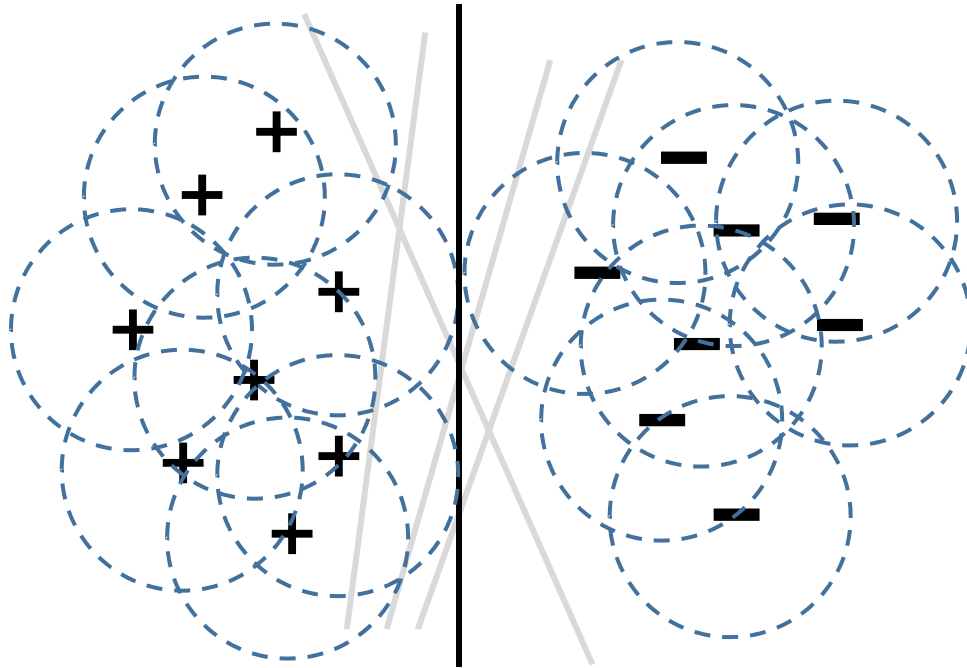
- For linear separable cases, we have multiple decision boundaries



- Ruling out some separators by considering data noise

Linear Classification

- For linear separable cases, we have multiple decision boundaries

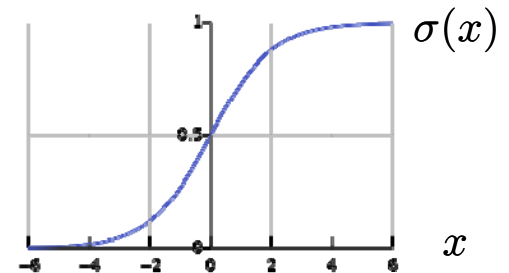


- The intuitive optimal decision boundary: the largest margin

Review: Logistic Regression

- Logistic regression is a binary classification model

$$p_{\theta}(y = 1|x) = \sigma(\theta^{\top} x) = \frac{1}{1 + e^{-\theta^{\top} x}}$$
$$p_{\theta}(y = 0|x) = \frac{e^{-\theta^{\top} x}}{1 + e^{-\theta^{\top} x}}$$



- Cross entropy loss function

$$\mathcal{L}(y, x, p_{\theta}) = -y \log \sigma(\theta^{\top} x) - (1 - y) \log(1 - \sigma(\theta^{\top} x))$$

- Gradient

$$\begin{aligned} \frac{\partial \mathcal{L}(y, x, p_{\theta})}{\partial \theta} &= -y \frac{1}{\sigma(\theta^{\top} x)} \sigma(z)(1 - \sigma(z))x - (1 - y) \frac{-1}{1 - \sigma(\theta^{\top} x)} \sigma(z)(1 - \sigma(z))x \\ &= (\sigma(\theta^{\top} x) - y)x \\ \theta &\leftarrow \theta + \eta(y - \sigma(\theta^{\top} x))x \end{aligned}$$

$$\frac{\partial \sigma(z)}{\partial z} = \sigma(z)(1 - \sigma(z))$$

Label Decision

- Logistic regression provides the probability

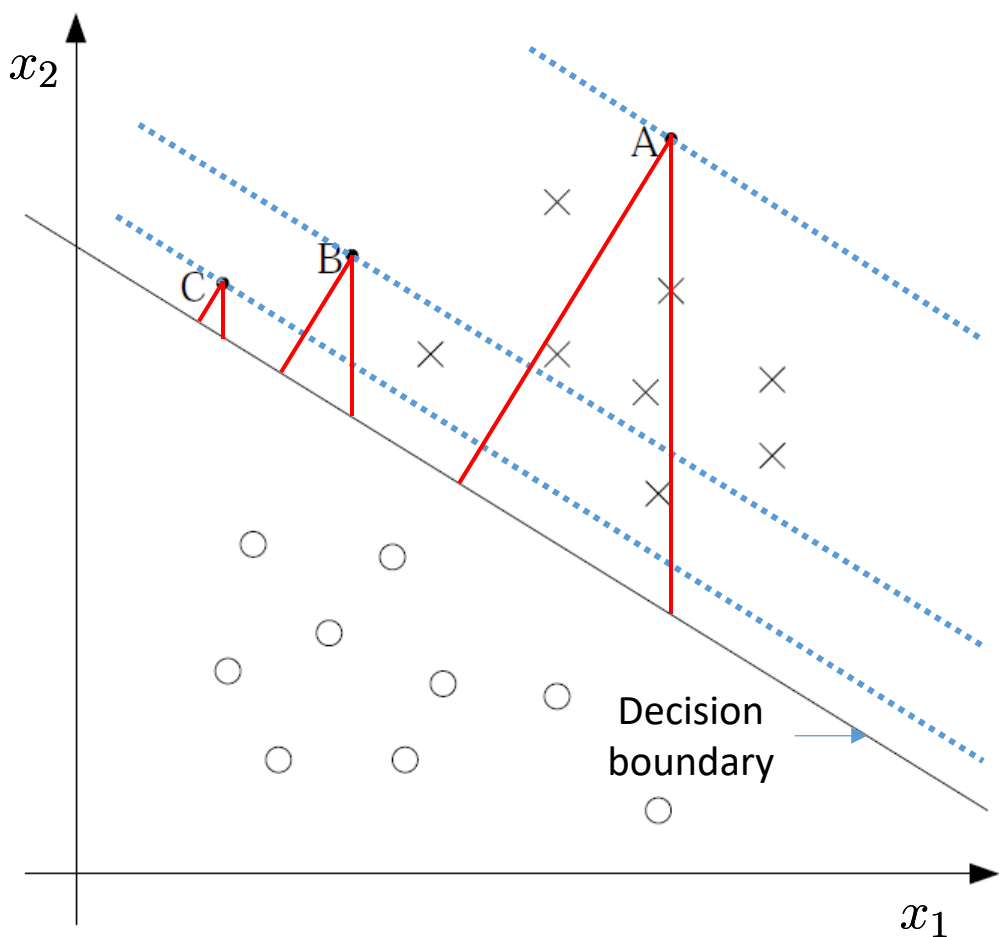
$$p_{\theta}(y = 1|x) = \sigma(\theta^{\top} x) = \frac{1}{1 + e^{-\theta^{\top} x}}$$

$$p_{\theta}(y = 0|x) = \frac{e^{-\theta^{\top} x}}{1 + e^{-\theta^{\top} x}}$$

- The final label of an instance is decided by setting a threshold h

$$\hat{y} = \begin{cases} 1, & p_{\theta}(y = 1|x) > h \\ 0, & \text{otherwise} \end{cases}$$

Logistic Regression Scores



$$s(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

$$p_{\theta}(y = 1|x) = \frac{1}{1 + e^{-s(x)}}$$

$$s(x) = \theta_0 + \theta_1 x_1^{(A)} + \theta_2 x_2^{(A)}$$

Logistic函数横坐标x代表离
决策面距离

$$s(x) = \theta_0 + \theta_1 x_1^{(B)} + \theta_2 x_2^{(B)}$$

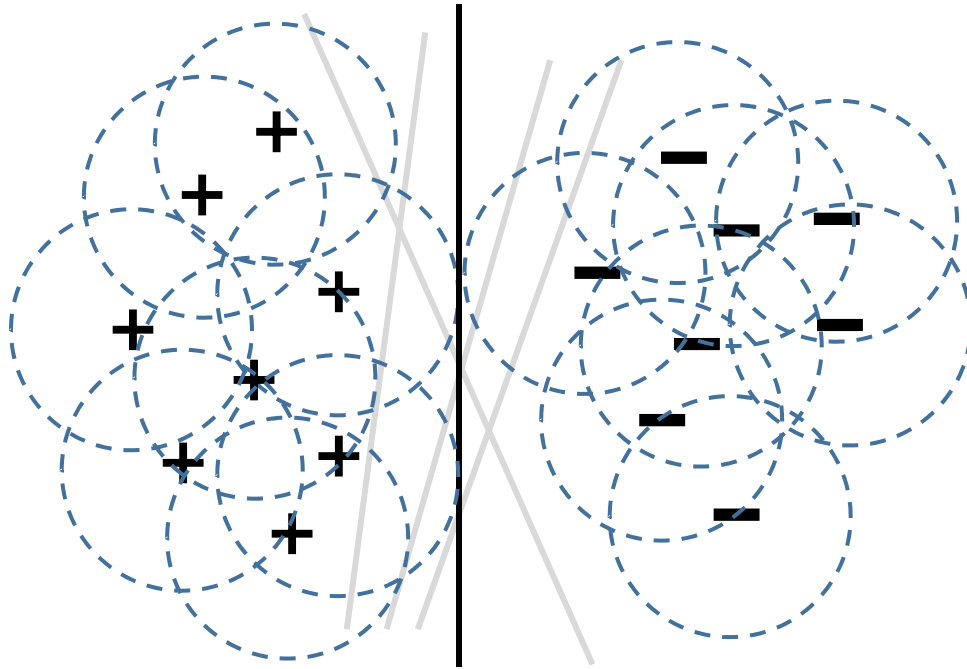
$$s(x) = \theta_0 + \theta_1 x_1^{(C)} + \theta_2 x_2^{(C)}$$

$$s(x) = 0$$

The higher score, the larger distance to the decision boundary, the higher confidence

Linear Classification

- The intuitive optimal decision boundary: the highest confidence



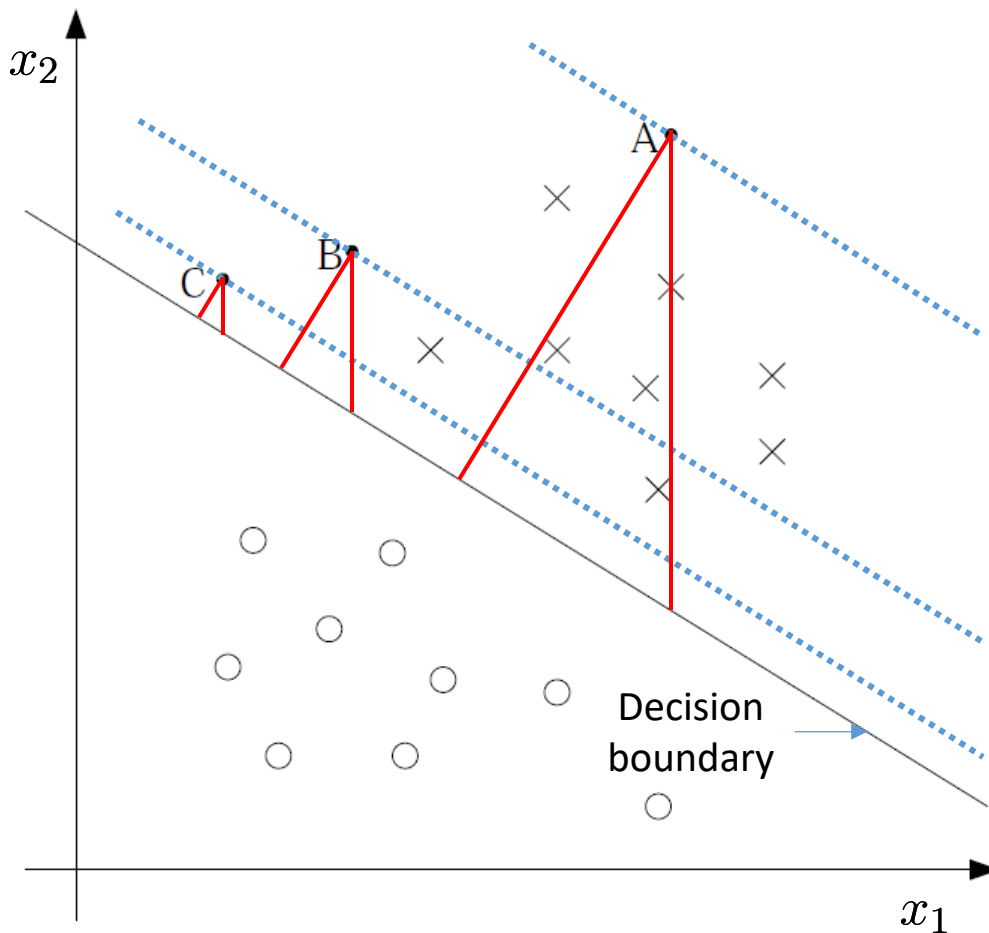
Notations for SVMs

- Feature vector x
- Class label $y \in \{-1, 1\}$
- Parameters
 - Intercept b
 - Feature weight vector w
- Label prediction

$$h_{w,b}(x) = g(w^\top x + b)$$

$$g(z) = \begin{cases} +1 & z \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

Logistic Regression Scores



$$s(x) = b + w_1x_1 + w_2x_2$$

$$p_{\theta}(y = 1|x) = \frac{1}{1 + e^{-s(x)}}$$

$$s(x) = b + w_1x_1^{(A)} + w_2x_2^{(A)}$$

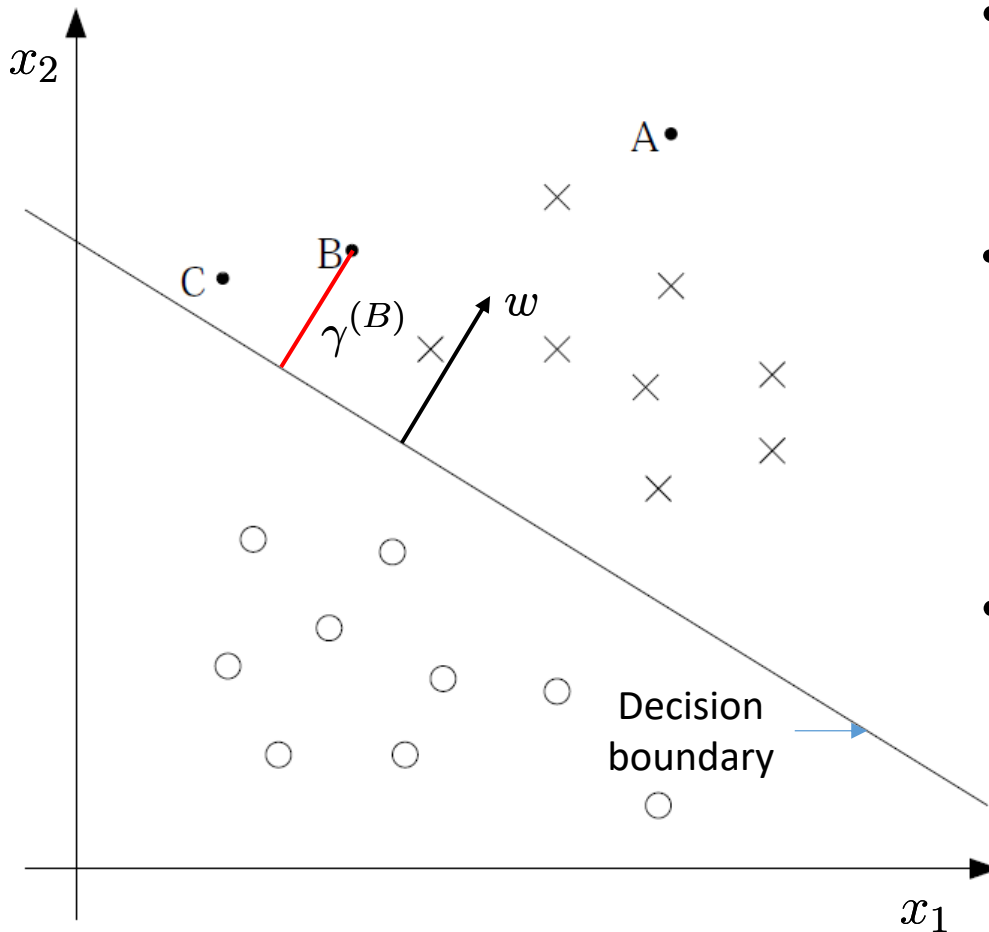
$$s(x) = b + w_1x_1^{(B)} + w_2x_2^{(B)}$$

$$s(x) = b + w_1x_1^{(C)} + w_2x_2^{(C)}$$

$$s(x) = 0$$

The higher score, the larger distance to the separating hyperplane, the higher confidence

Margins



- Functional margin

$$\hat{\gamma}^{(i)} = y^{(i)}(w^\top x^{(i)} + b)$$

- Note that the separating hyperplane won't change with the magnitude of (w, b)

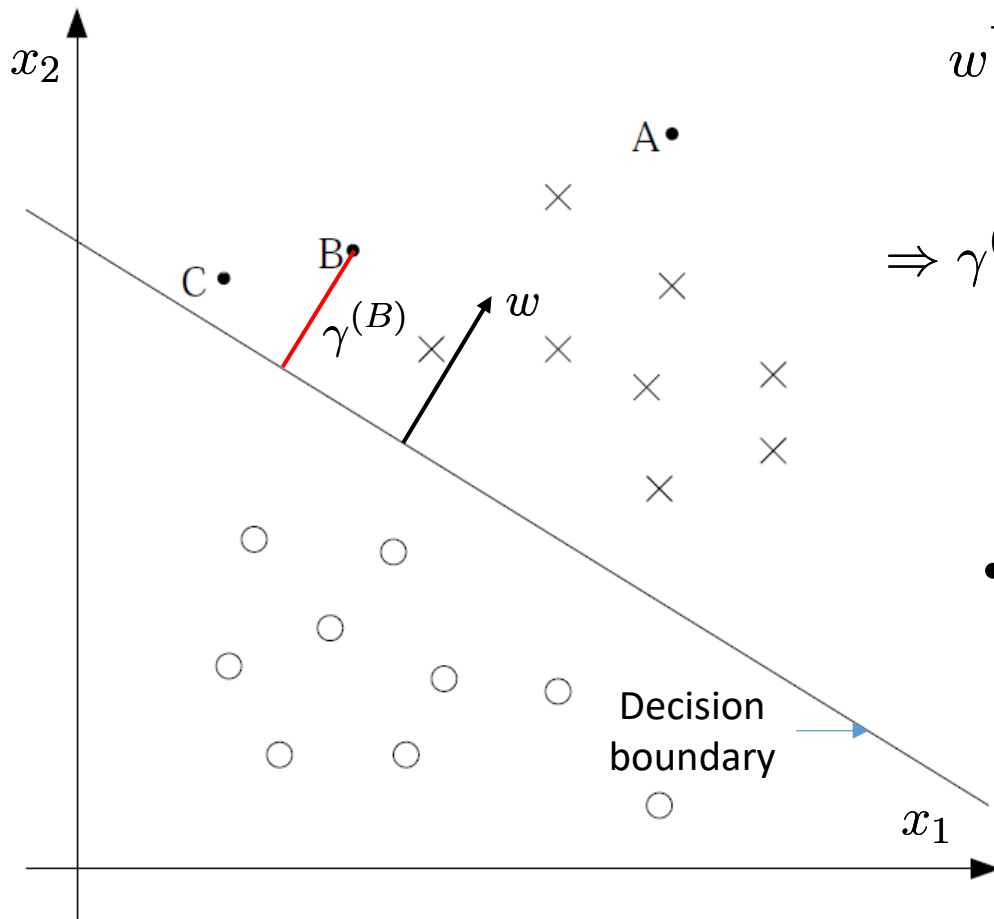
$$g(w^\top x + b) = g(2w^\top x + 2b)$$

- Geometric margin

$$\gamma^{(i)} = y^{(i)}(w^\top x^{(i)} + b)$$

$$\text{where } \|w\|^2 = 1$$

Margins



- Decision boundary

$$w^T \left(x^{(i)} - \gamma^{(i)} y^{(i)} \frac{w}{\|w\|} \right) + b = 0$$

$$\begin{aligned} \Rightarrow \gamma^{(i)} &= y^{(i)} \frac{w^T x^{(i)} + b}{\|w\|} \\ &= y^{(i)} \left(\left(\frac{w}{\|w\|} \right)^T x^{(i)} + \frac{b}{\|w\|} \right) \end{aligned}$$

- Given a training set

$$S = \{(x_i, y_i)\}_{i=1 \dots m}$$

the smallest geometric margin

$$\gamma = \min_{i=1 \dots m} \gamma^{(i)}$$

Objective of an SVM

- Find a separable hyperplane that maximizes the minimum geometric margin

$$\begin{aligned} \max_{\gamma, w, b} \quad & \gamma \\ \text{s.t.} \quad & y^{(i)}(w^\top x^{(i)} + b) \geq \gamma, \quad i = 1, \dots, m \\ & \|w\| = 1 \quad (\text{non-convex constraint}) \end{aligned}$$

球壳非凸集，不满足 $tx+(1-t)y \in S$

- Equivalent to normalized functional margin

$$\begin{aligned} \max_{\hat{\gamma}, w, b} \quad & \frac{\hat{\gamma}}{\|w\|} \quad (\text{non-convex objective}) \\ \text{s.t.} \quad & y^{(i)}(w^\top x^{(i)} + b) \geq \hat{\gamma}, \quad i = 1, \dots, m \end{aligned}$$

$\hat{\gamma}$ 为Functional Margin， γ 为Geographical Margin

Objective of an SVM

- Functional margin scales w.r.t. (w, b) without changing the decision boundary.
 - Let's fix the functional margin at 1.

$$\hat{\gamma} = 1$$

- Objective is written as

$$\begin{aligned} \max_{w, b} \quad & \frac{1}{\|w\|} \\ \text{s.t.} \quad & y^{(i)}(w^\top x^{(i)} + b) \geq 1, \quad i = 1, \dots, m \end{aligned}$$

- Equivalent with

$$\begin{aligned} \min_{w, b} \quad & \frac{1}{2} \|w\|^2 \\ \text{s.t.} \quad & y^{(i)}(w^\top x^{(i)} + b) \geq 1, \quad i = 1, \dots, m \end{aligned}$$

This optimization problem can be efficiently solved by quadratic programming

A Digression of Lagrange Duality in Convex Optimization

Boyd, Stephen, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Lagrangian for Convex Optimization

- A convex optimization problem

$$\begin{aligned} \min_w \quad & f(w) \\ \text{s.t.} \quad & h_i(w) = 0, \quad i = 1, \dots, l \end{aligned}$$

- The Lagrangian of this problem is defined as

$$\mathcal{L}(w, \beta) = f(w) + \sum_{i=1}^l \beta_i h_i(w)$$

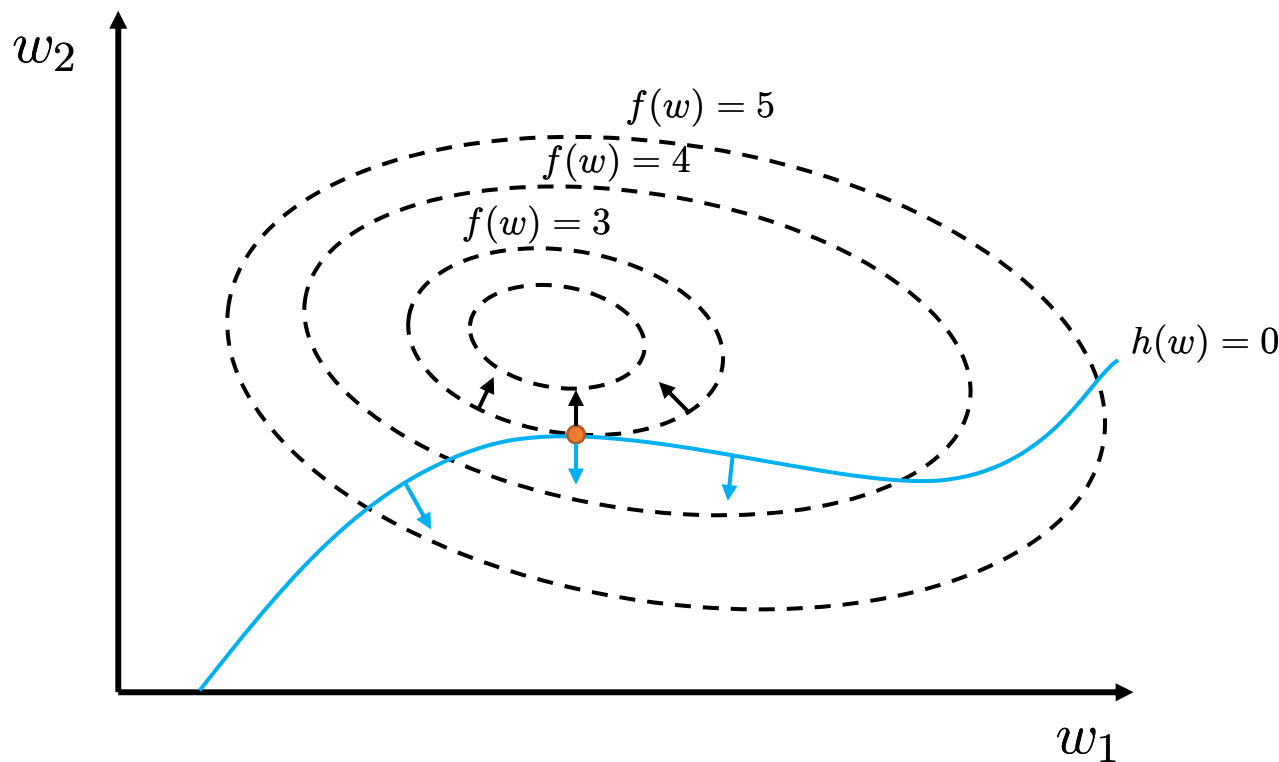
↑
Lagrangian multipliers

- Solving

$$\frac{\partial \mathcal{L}(w, \beta)}{\partial w} = 0 \quad \frac{\partial \mathcal{L}(w, \beta)}{\partial \beta} = 0$$

yields the solution of the original optimization problem.

Lagrangian for Convex Optimization



$$\mathcal{L}(w, \beta) = f(w) + \beta h(w)$$

$$\frac{\partial \mathcal{L}(w, \beta)}{\partial w} = \frac{\partial f(w)}{\partial w} + \beta \frac{\partial h(w)}{\partial w} = 0$$

i.e., two gradients on
the same direction

$$(\text{grad } f + \text{grad } h)w = 0$$


With Inequality Constraints

- A convex optimization problem

$$\begin{aligned} \min_w \quad & f(w) \\ \text{s.t.} \quad & g_i(w) \leq 0, \quad i = 1, \dots, k \\ & h_i(w) = 0, \quad i = 1, \dots, l \end{aligned}$$

- The Lagrangian of this problem is defined as

$$\mathcal{L}(w, \alpha, \beta) = f(w) + \sum_{i=1}^k \alpha_i g_i(w) + \sum_{i=1}^l \beta_i h_i(w)$$


Lagrangian multipliers

Primal Problem

- A convex optimization

$$\begin{aligned} \min_w \quad & f(w) \\ \text{s.t.} \quad & g_i(w) \leq 0, \quad i = 1, \dots, k \\ & h_i(w) = 0, \quad i = 1, \dots, l \end{aligned}$$

- The Lagrangian

$$\mathcal{L}(w, \alpha, \beta) = f(w) + \sum_{i=1}^k \alpha_i g_i(w) + \sum_{i=1}^l \beta_i h_i(w)$$

- The primal problem

$$\theta_{\mathcal{P}}(w) = \max_{\alpha, \beta: \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta)$$

- If a given w violates any constraints, i.e.,

$$g_i(w) > 0 \quad \text{or} \quad h_i(w) \neq 0$$

- Then $\theta_{\mathcal{P}}(w) = +\infty$

Primal Problem

- A convex optimization

$$\begin{aligned} \min_w \quad & f(w) \\ \text{s.t.} \quad & g_i(w) \leq 0, \quad i = 1, \dots, k \\ & h_i(w) = 0, \quad i = 1, \dots, l \end{aligned}$$

- The Lagrangian

$$\mathcal{L}(w, \alpha, \beta) = f(w) + \sum_{i=1}^k \alpha_i g_i(w) + \sum_{i=1}^l \beta_i h_i(w)$$

- The primal problem

$$\theta_{\mathcal{P}}(w) = \max_{\alpha, \beta: \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta)$$

- Conversely, if all constraints are satisfied for w
- Then $\theta_{\mathcal{P}}(w) = f(w)$

$$\theta_{\mathcal{P}}(w) = \begin{cases} f(w) & \text{if } w \text{ satisfies primal constraints} \\ +\infty & \text{otherwise} \end{cases}$$

Primal Problem

$$\theta_{\mathcal{P}}(w) = \begin{cases} f(w) & \text{if } w \text{ satisfies primal constraints} \\ +\infty & \text{otherwise} \end{cases}$$

- The minimization problem

$$\min_w \theta_{\mathcal{P}}(w) = \min_w \max_{\alpha, \beta: \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta)$$

is the same as the original problem

$$\begin{aligned} \min_w \quad & f(w) \\ \text{s.t.} \quad & g_i(w) \leq 0, \quad i = 1, \dots, k \\ & h_i(w) = 0, \quad i = 1, \dots, l \end{aligned}$$

- Define the value of the primal problem $p^* = \min_w \theta_{\mathcal{P}}(w)$

Dual Problem

- A slightly different problem

$$\theta_{\mathcal{D}}(\alpha, \beta) = \min_w \mathcal{L}(w, \alpha, \beta)$$

- Define the dual optimization problem

$$\max_{\alpha, \beta: \alpha_i \geq 0} \theta_{\mathcal{D}}(\alpha, \beta) = \max_{\alpha, \beta: \alpha_i \geq 0} \min_w \mathcal{L}(w, \alpha, \beta)$$

- Min & Max exchanged compared to the primal problem

$$\min_w \theta_{\mathcal{P}}(w) = \min_w \max_{\alpha, \beta: \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta)$$

- Define the value of the dual problem

$$d^* = \max_{\alpha, \beta: \alpha_i \geq 0} \min_w \mathcal{L}(w, \alpha, \beta)$$

Primal Problem vs. Dual Problem

$$d^* = \max_{\alpha, \beta: \alpha_i \geq 0} \min_w \mathcal{L}(w, \alpha, \beta) \leq \min_w \max_{\alpha, \beta: \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta) = p^*$$

- Proof

$$\min_w \mathcal{L}(w, \alpha, \beta) \leq \mathcal{L}(w, \alpha, \beta), \forall w, \alpha \geq 0, \beta$$

$$\Rightarrow \max_{\alpha, \beta: \alpha \geq 0} \min_w \mathcal{L}(w, \alpha, \beta) \leq \max_{\alpha, \beta: \alpha \geq 0} \mathcal{L}(w, \alpha, \beta), \forall w$$

$$\Rightarrow \max_{\alpha, \beta: \alpha \geq 0} \min_w \mathcal{L}(w, \alpha, \beta) \leq \min_w \max_{\alpha, \beta: \alpha \geq 0} \mathcal{L}(w, \alpha, \beta)$$

□

- But under certain condition $d^* = p^*$

Karush-Kuhn-Tucker (KKT) Conditions

- If f and g_i 's are convex and h_i 's are affine, and suppose g_i 's are all strictly feasible
- then there must exist w^*, α^*, β^*
 - w^* is the solution of the primal problem
 - α^*, β^* are the solutions of the dual problem
 - and the values of the two problems are equal $p^* = d^* = \mathcal{L}(w^*, \alpha^*, \beta^*)$
- And w^*, α^*, β^* satisfy the KKT conditions 关于 w , 为常值函数

$$\frac{\partial}{\partial w_i} \mathcal{L}(w^*, \alpha^*, \beta^*) = 0, \quad i = 1, \dots, n$$

$$\frac{\partial}{\partial \beta_i} \mathcal{L}(w^*, \alpha^*, \beta^*) = 0, \quad i = 1, \dots, l$$

KKT dual
complementarity
condition

$$\longrightarrow \alpha_i^* g_i(w^*) = 0, \quad i = 1, \dots, k$$

$$g_i(w^*) \leq 0, \quad i = 1, \dots, k$$

$$\alpha^* \geq 0, \quad i = 1, \dots, k$$

- Moreover, if some w^*, α^*, β^* satisfy the KKT conditions, then it is also a solution to the primal and dual problems.
- More details please refer to Boyd "Convex optimization" 2004.

Now Back to SVM Problem

Objective of an SVM

- SVM objective: finding the optimal margin classifier

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2} \|w\|^2 \\ \text{s.t.} \quad & y^{(i)}(w^\top x^{(i)} + b) \geq 1, \quad i = 1, \dots, m \end{aligned}$$

- Re-wright the constraints as

$$g_i(w) = -y^{(i)}(w^\top x^{(i)} + b) + 1 \leq 0$$

so as to match the standard optimization form

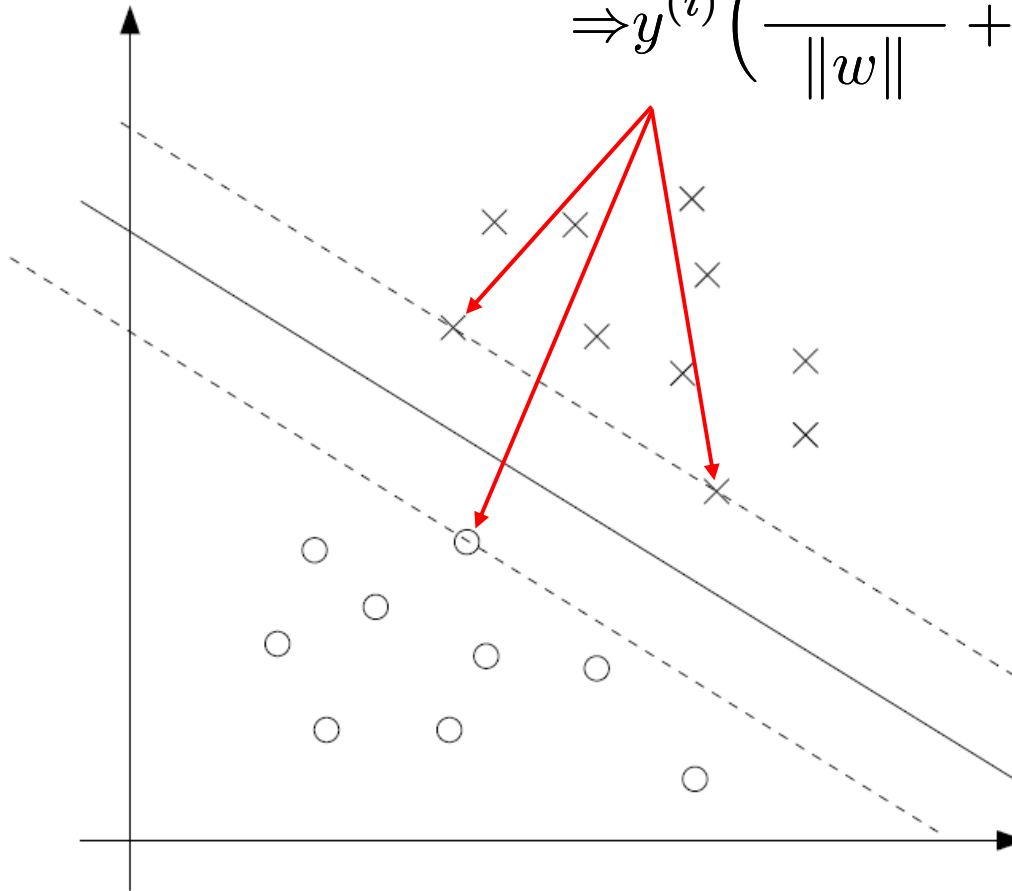
$$\begin{aligned} \min_w \quad & f(w) \\ \text{s.t.} \quad & g_i(w) \leq 0, \quad i = 1, \dots, k \\ & h_i(w) = 0, \quad i = 1, \dots, l \end{aligned}$$

Equality Cases

$$g_i(w) = -y^{(i)}(w^\top x^{(i)} + b) + 1 = 0$$

$$\Rightarrow y^{(i)} \left(\frac{w^\top x^{(i)}}{\|w\|} + \frac{b}{\|w\|} \right) = \frac{1}{\|w\|}$$

↑
Geometric margin



The g_i 's = 0 cases correspond to the training examples that have functional margin exactly equal to 1.

Objective of an SVM

- SVM objective: finding the optimal margin classifier

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2} \|w\|^2 \\ \text{s.t.} \quad & -y^{(i)}(w^\top x^{(i)} + b) + 1 \leq 0, \quad i = 1, \dots, m \end{aligned}$$

- Lagrangian

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^m \alpha_i [y^{(i)}(w^\top x^{(i)} + b) - 1]$$

- No β or equality constraints in SVM problem

Solving

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^m \alpha_i [y^{(i)} (w^\top x^{(i)} + b) - 1]$$

- Derivatives

$$\frac{\partial}{\partial w} \mathcal{L}(w, b, \alpha) = w - \sum_{i=1}^m \alpha_i y^{(i)} x^{(i)} = 0 \quad \Rightarrow \quad w = \sum_{i=1}^m \alpha_i y^{(i)} x^{(i)}$$

$$\frac{\partial}{\partial b} \mathcal{L}(w, b, \alpha) = \sum_{i=1}^m \alpha_i y^{(i)} = 0$$

- Then Lagrangian is re-written as

$$\begin{aligned} \mathcal{L}(w, b, \alpha) &= \frac{1}{2} \left\| \sum_{i=1}^m \alpha_i y^{(i)} x^{(i)} \right\|^2 - \sum_{i=1}^m \alpha_i [y^{(i)} (w^\top x^{(i)} + b) - 1] \\ &= \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m y^{(i)} y^{(j)} \alpha_i \alpha_j x^{(i)\top} x^{(j)} \left[-b \sum_{i=1}^m \alpha_i y^{(i)} \right] = 0 \end{aligned}$$

Solving α^*

- Dual problem

$$\max_{\alpha \geq 0} \theta_{\mathcal{D}}(\alpha) = \max_{\alpha \geq 0} \min_{w, b} \mathcal{L}(w, b, \alpha)$$

$$\max_{\alpha} \quad W(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m y^{(i)} y^{(j)} \alpha_i \alpha_j x^{(i)\top} x^{(j)}$$

$$\text{s.t.} \quad \alpha_i \geq 0, \quad i = 1, \dots, m$$

$$\sum_{i=1}^m \alpha_i y^{(i)} = 0$$

- To solve α^* with some methods e.g. SMO
 - We will get back to this solution later

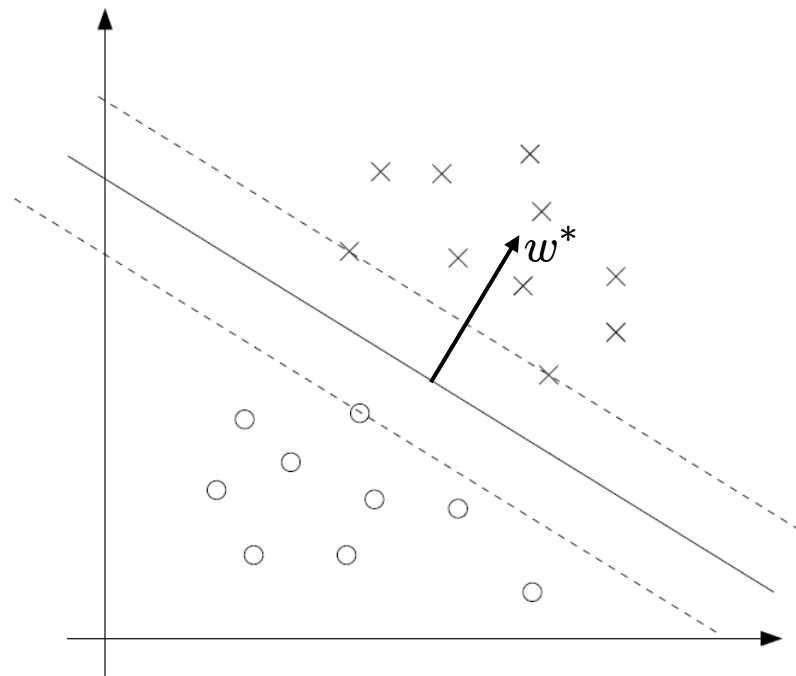
Solving w^* and b^*

- With α^* solved, w^* is obtained by

$$w = \sum_{i=1}^m \alpha_i y^{(i)} x^{(i)}$$

- Only supporting vectors with $\alpha > 0$
- With w^* solved, b^* is obtained by

$$b^* = - \frac{\max_{i:y^{(i)}=-1} w^{*\top} x^{(i)} + \min_{i:y^{(i)}=1} w^{*\top} x^{(i)}}{2}$$

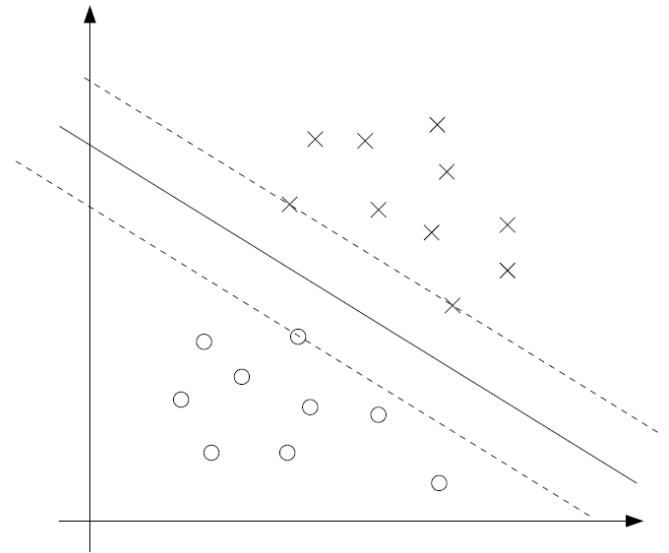


Predicting Values

- With the solutions of w^* and b^* , the predicting value (i.e. functional margin) of each instance is

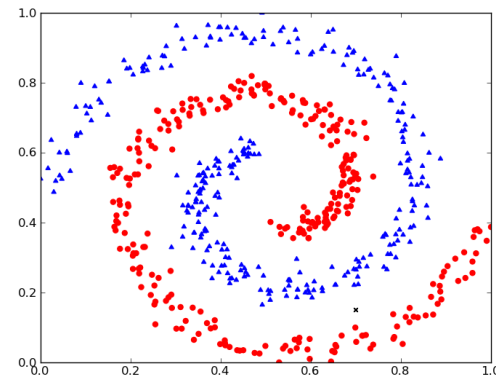
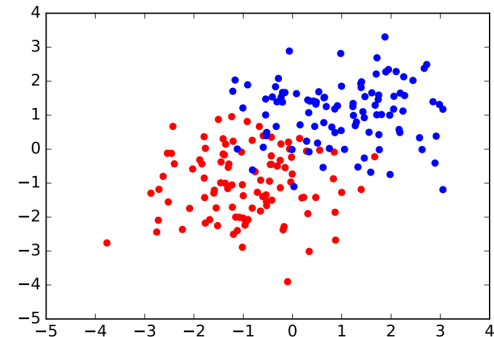
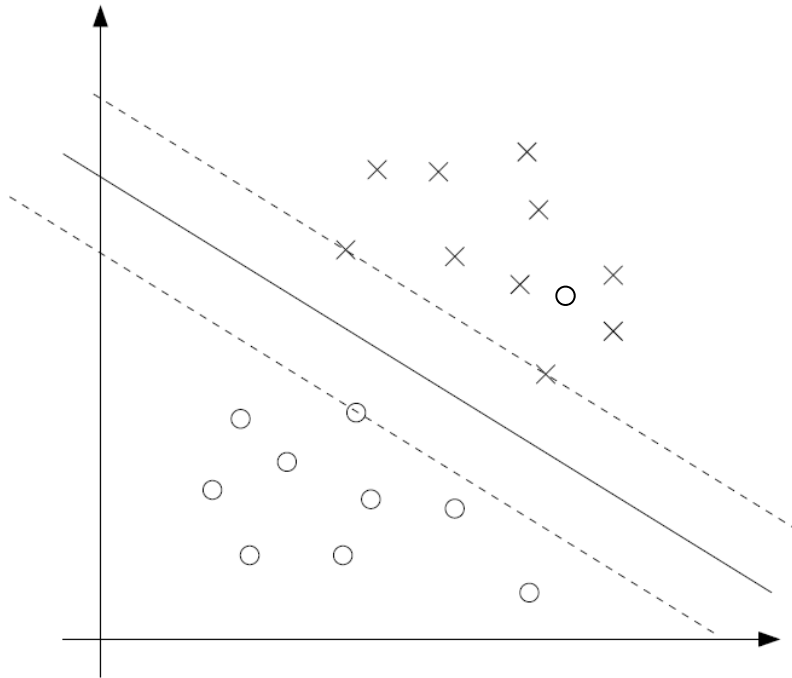
$$\begin{aligned}w^{*\top}x + b^* &= \left(\sum_{i=1}^m \alpha_i y^{(i)} x^{(i)} \right)^\top x + b^* \\&= \sum_{i=1}^m \alpha_i y^{(i)} \langle x^{(i)}, x \rangle + b^*\end{aligned}$$

- We only need to calculate the inner product of x with the supporting vectors



Non-Separable Cases

- The derivation of the SVM as presented so far assumes that the data is linearly separable.
- More practical cases are linearly non-separable.



Dealing with Non-Separable Cases

- Add slack variables $\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i \leftarrow \text{L1 regularization}$
s.t. $y^{(i)}(w^\top x^{(i)} + b) \geq 1 - \xi_i, \quad i = 1, \dots, m$
 $\xi_i \geq 0, \quad i = 1, \dots, m$

- Lagrangian

$$\mathcal{L}(w, b, \xi, \alpha, r) = \frac{1}{2} w^\top w + C \sum_{i=1}^m \xi_i - \sum_{i=1}^m \alpha_i [y^{(i)}(x^\top w + b) - 1 + \xi_i] - \sum_{i=1}^m r_i \xi_i$$

- Dual problem

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m y^{(i)} y^{(j)} \alpha_i \alpha_j x^{(i)\top} x^{(j)}$$

$$\text{s.t. } 0 \leq \alpha_i \leq C, \quad i = 1, \dots, m \quad \text{Surprisingly, this is the only change}$$

$$\sum_{i=1}^m \alpha_i y^{(i)} = 0$$

Efficiently solved by SMO algorithm

SVM Hinge Loss vs. LR Loss

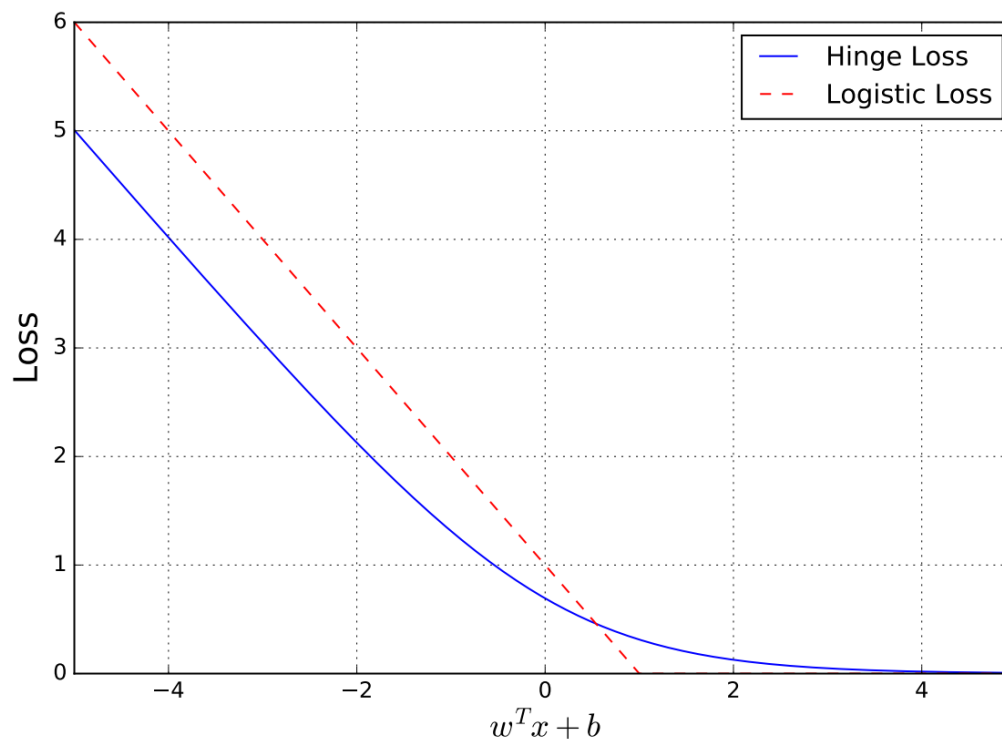
- SVM Hinge loss

$$\frac{1}{2}\|w\|^2 + C \sum_{i=1}^m \max(0, 1 - y_i(w^\top x_i + b))$$

- LR log loss

$$-y_i \log \sigma(w^\top x_i + b) - (1 - y_i) \log(1 - \sigma(w^\top x_i + b))$$

- If $y = 1$



Coordinate Ascent (Descent)

- For the optimization problem

$$\max_{\alpha} W(\alpha_1, \alpha_2, \dots, \alpha_m)$$

- Coordinate ascent algorithm

Loop until convergence: {

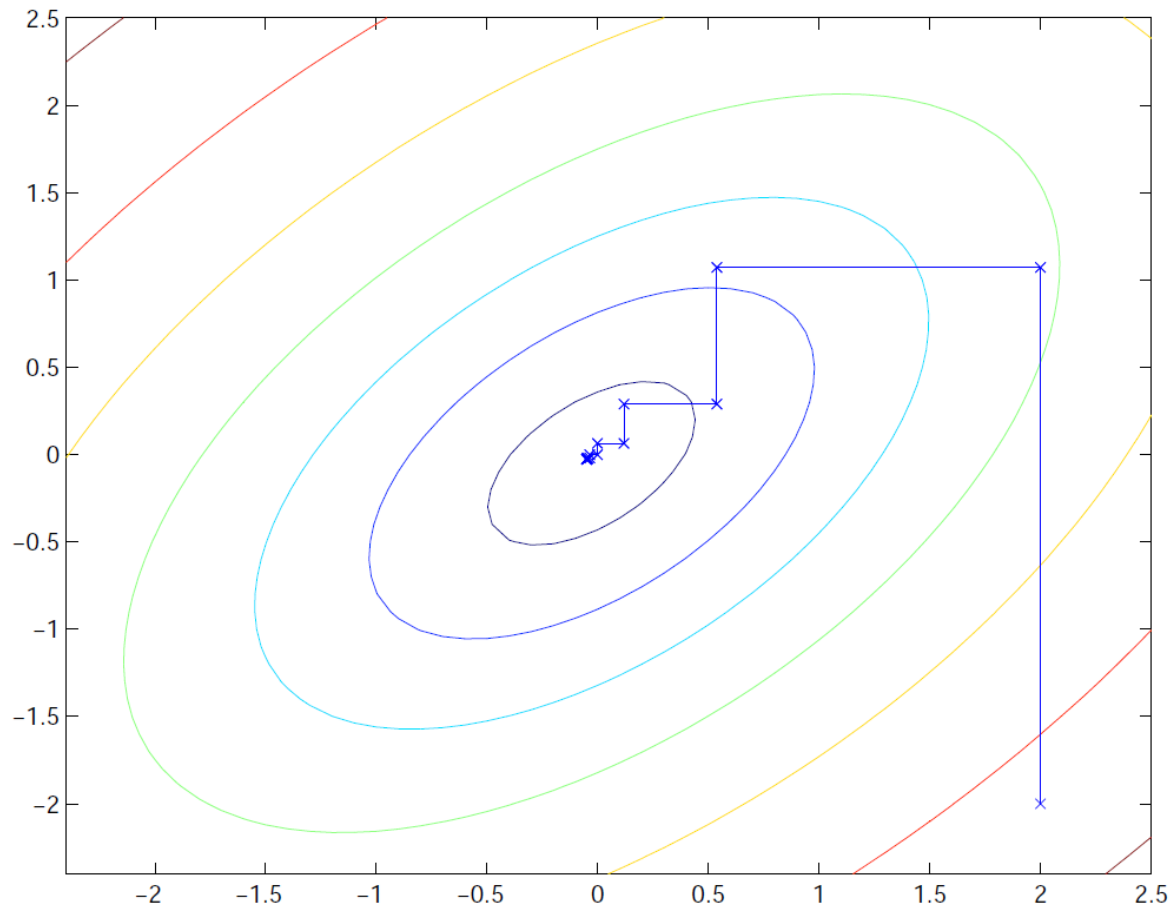
For $i = 1, \dots, m$ {

$$\alpha_i := \arg \max_{\hat{\alpha}_i} W(\alpha_1, \dots, \alpha_{i-1}, \hat{\alpha}_i, \alpha_{i+1}, \dots, \alpha_m)$$

}

}

Coordinate Ascent (Descent)



A two-dimensional coordinate ascent example

SMO Algorithm

- SMO: sequential minimal optimization
- SVM optimization problem

$$\begin{aligned} \max_{\alpha} \quad & W(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m y^{(i)} y^{(j)} \alpha_i \alpha_j x^{(i)\top} x^{(j)} \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, \quad i = 1, \dots, m \\ & \sum_{i=1}^m \alpha_i y^{(i)} = 0 \end{aligned}$$

- Cannot directly apply coordinate ascent algorithm because

$$\sum_{i=1}^m \alpha_i y^{(i)} = 0 \Rightarrow \alpha_i y^{(i)} = - \sum_{j \neq i} \alpha_j y^{(j)}$$

SMO Algorithm

- Update two variable each time

Loop until convergence {

1. Select some pair α_i and α_j to update next
2. Re-optimize $W(\alpha)$ w.r.t. α_i and α_j

}

- Convergence test: whether the change of $W(\alpha)$ is smaller than a predefined value (e.g. 0.01)
- Key advantage of SMO algorithm is the update of α_i and α_j (step 2) is efficient

SMO Algorithm

$$\max_{\alpha} \quad W(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m y^{(i)} y^{(j)} \alpha_i \alpha_j x^{(i)\top} x^{(j)}$$

$$\text{s.t. } 0 \leq \alpha_i \leq C, \quad i = 1, \dots, m$$

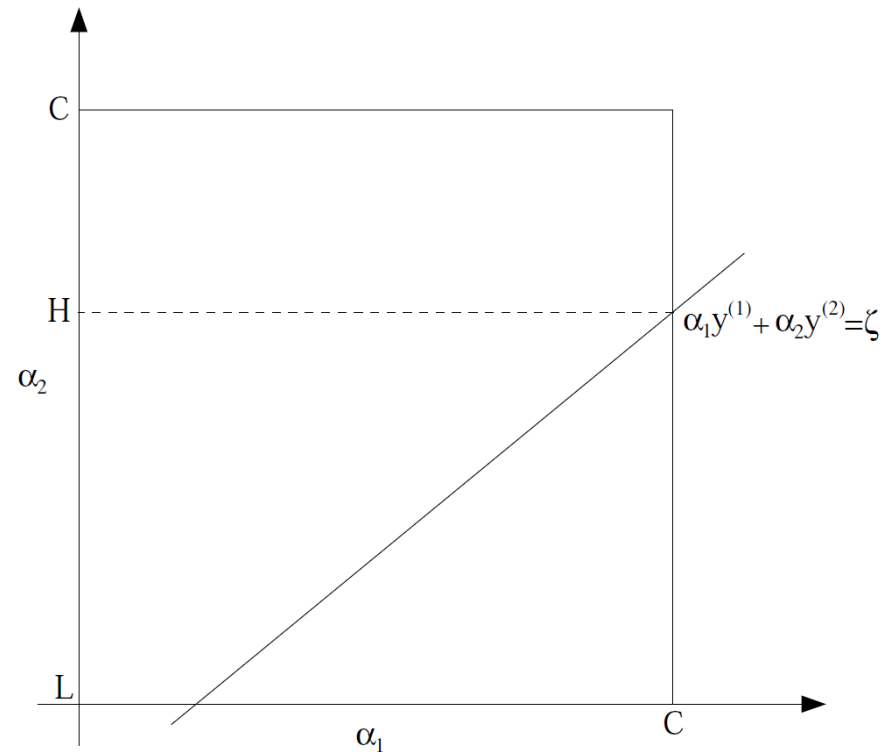
$$\sum_{i=1}^m \alpha_i y^{(i)} = 0$$

- Without loss of generality, hold $\alpha_3 \dots \alpha_m$ and optimize $W(\alpha)$ w.r.t. α_1 and α_2

$$\alpha_1 y^{(1)} + \alpha_2 y^{(2)} = - \sum_{i=3}^m \alpha_i y^{(i)} = \zeta$$

$$\Rightarrow \alpha_2 = -\frac{y^{(1)}}{y^{(2)}} \alpha_1 + \frac{\zeta}{y^{(2)}}$$

$$\alpha_1 = (\zeta - \alpha_2 y^{(2)}) y^{(1)}$$



SMO Algorithm

- With $\alpha_1 = (\zeta - \alpha_2 y^{(2)}) y^{(1)}$, the objective is written as

$$W(\alpha_1, \alpha_2, \dots, \alpha_m) = W((\zeta - \alpha_2 y^{(2)}) y^{(1)}, \alpha_2, \dots, \alpha_m)$$

- Thus the original optimization problem

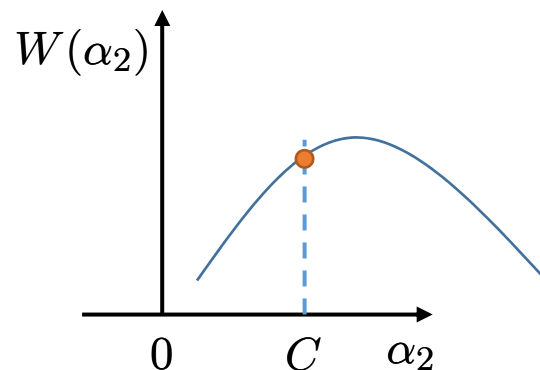
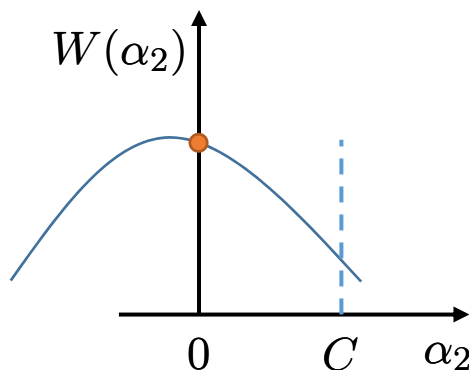
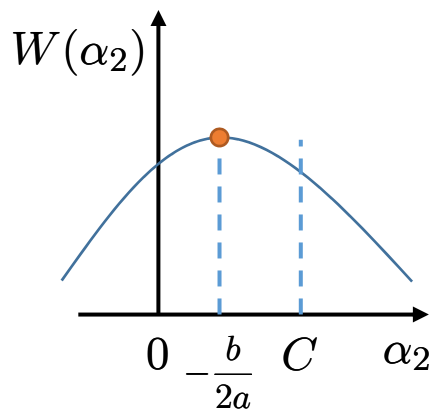
$$\begin{aligned} \max_{\alpha} \quad & W(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m y^{(i)} y^{(j)} \alpha_i \alpha_j x^{(i)\top} x^{(j)} \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, \quad i = 1, \dots, m \\ & \sum_{i=1}^m \alpha_i y^{(i)} = 0 \end{aligned}$$

is transformed into a quadratic optimization problem w.r.t. α_2

$$\begin{aligned} \max_{\alpha_2} \quad & W(\alpha_2) = a\alpha_2^2 + b\alpha_2 + c \\ \text{s.t.} \quad & 0 \leq \alpha_2 \leq C \end{aligned}$$

SMO Algorithm

- Optimizing a quadratic function is much efficient

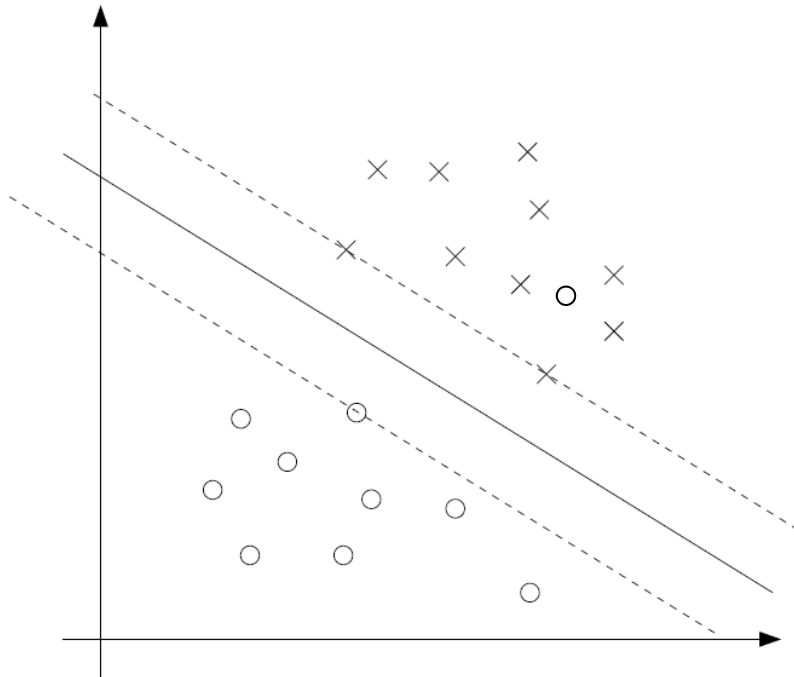


$$\begin{aligned} \max_{\alpha_2} \quad & W(\alpha_2) = a\alpha_2^2 + b\alpha_2 + c \\ \text{s.t.} \quad & 0 \leq \alpha_2 \leq C \end{aligned}$$

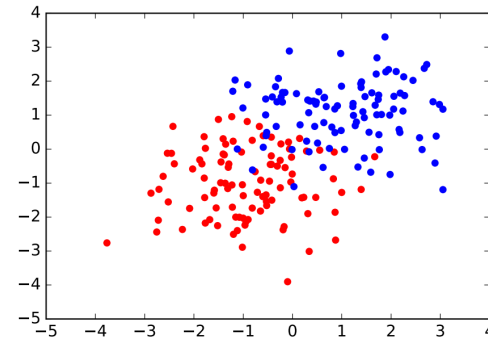
Kernel Methods

Non-Separable Cases

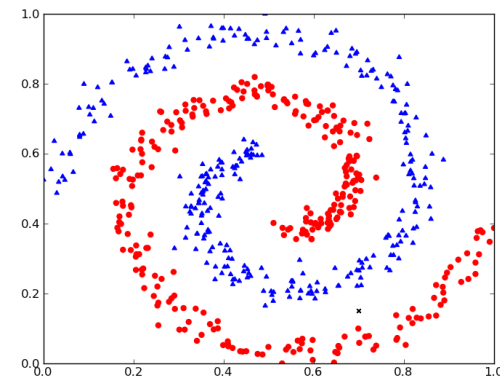
- More practical cases are linearly non-separable.



Linearly separable case



May be solved by slack variables



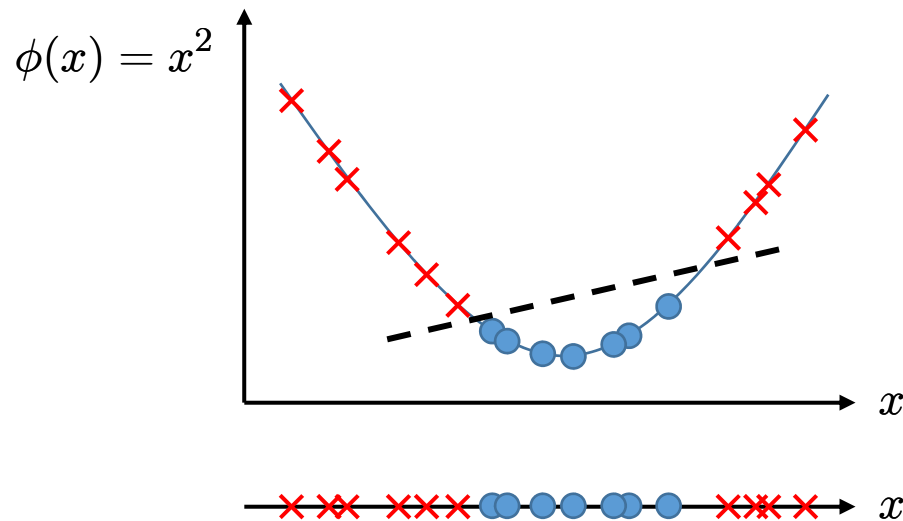
Cannot be solved by slack variables

Non-Separable Cases

- More practical cases are linearly non-separable.
- Solution: mapping feature vectors to a higher-dimensional space

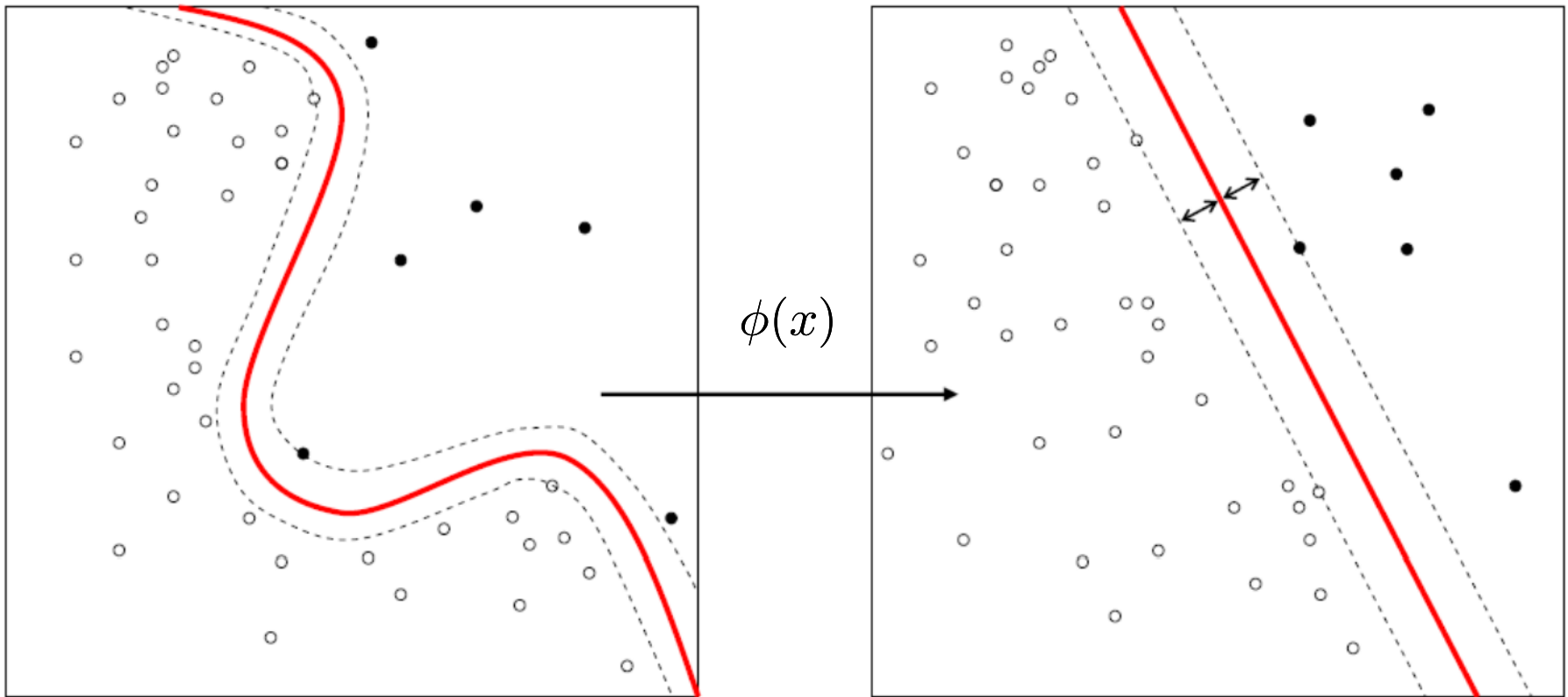
$$\phi(x)$$

- An example



Non-Separable Cases

- More generally, mapping feature vectors to a different space



Feature Mapping Functions

- SVM only cares about the inner products

$$W(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m y^{(i)} y^{(j)} \alpha_i \alpha_j \mathbf{x}^{(i)\top} \mathbf{x}^{(j)}$$

- With the feature mapping function $\phi(x)$

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

- Kernel $K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \phi(\mathbf{x}^{(i)})^\top \phi(\mathbf{x}^{(j)})$

Kernel

- With the example feature mapping function

$$\phi(x) = \begin{bmatrix} x \\ x^2 \\ x^3 \end{bmatrix}$$

- The corresponding kernel is

$$\begin{aligned} K(x^{(i)}, x^{(j)}) &= \phi(x^{(i)})^\top \phi(x^{(j)}) \\ &= x^{(i)}x^{(j)} + x^{(i)2}x^{(j)2} + x^{(i)3}x^{(j)3} \end{aligned}$$

[Kernel Trick]

- For lots of cases, we only need $K(x^{(i)}, x^{(j)})$, thus we can directly define $K(x^{(i)}, x^{(j)})$ without explicitly defining $\phi(x^{(i)})$
 - For example, suppose $x^{(i)}, x^{(j)} \in \mathbb{R}^n$

$$K(x^{(i)}, x^{(j)}) = (x^{(i)\top} x^{(j)})^2$$

Kernel Example

- For example, suppose $x^{(i)}, x^{(j)} \in \mathbb{R}^n$

If $n = 3$, the mapping function is

$$\begin{aligned} K(x^{(i)}, x^{(j)}) &= (x^{(i)\top} x^{(j)})^2 \\ &= \left(\sum_{k=1}^n x_k^{(i)} x_k^{(j)} \right) \left(\sum_{l=1}^n x_l^{(i)} x_l^{(j)} \right) \\ &= \sum_{k=1}^n \sum_{l=1}^n x_k^{(i)} x_k^{(j)} x_l^{(i)} x_l^{(j)} \quad \Rightarrow \quad \phi(x) = \begin{bmatrix} x_1 x_1 \\ x_1 x_2 \\ x_1 x_3 \\ x_2 x_1 \\ x_2 x_2 \\ x_2 x_3 \\ x_3 x_1 \\ x_3 x_2 \\ x_3 x_3 \end{bmatrix} \\ &= \sum_{k,l=1}^n (x_k^{(i)} x_l^{(i)}) (x_k^{(j)} x_l^{(j)}) \end{aligned}$$

- Note that calculating $\phi(x)$ takes $O(n^2)$ time, while calculating $K(x^{(i)}, x^{(j)})$ only takes $O(n)$ time

Kernel for Measuring Similarity

- Intuitively, for two instances x and z , if $\phi(x)$ and $\phi(z)$ are close together, then we expect

$$K(x, z) = \phi(x)^\top \phi(z)$$

to be large, and vice versa.

- Gaussian kernel (a very widely used kernel)

$$K(x, z) = \exp \left(-\frac{\|x - z\|^2}{2\sigma^2} \right)$$

- Also called radial basis function (RBF) kernel
- Then what is the feature mapping function for this kernel?

Kernel Matrix

- Consider a finite set of instances $\{x^{(1)}, \dots, x^{(m)}\}$
- The corresponding Kernel Matrix K is defined as $\{K_{ij}\}_{i,j=1,\dots,m}$
- The kernel matrix K must be symmetric since

$$K_{ij} = K(x^{(i)}, x^{(j)}) = \phi(x^{(i)})^\top \phi(x^{(j)}) = \phi(x^{(j)})^\top \phi(x^{(i)}) = K(x^{(j)}, x^{(i)}) = K_{ji}$$

- If we define $\phi_k(x)$ as the k -th coordinate of the vector $\phi(x)$, then for any vector $z \in \mathbb{R}^m$, we have

$$\begin{aligned} z^\top K z &= \sum_i \sum_j z_i K_{ij} z_j \\ &= \sum_i \sum_j z_i \phi(x^{(i)})^\top \phi(x^{(j)}) z_j = \sum_i \sum_j z_i \sum_k \phi_k(x^{(i)}) \phi_k(x^{(j)}) z_j \\ &= \sum_k \sum_i \sum_j z_i \phi_k(x^{(i)}) \phi_k(x^{(j)}) z_j = \sum_k \left(\sum_i z_i \phi_k(x^{(i)}) \right)^2 \geq 0 \end{aligned}$$

- Therefore, K is semi-definite

Valid (Mercer) Kernel

James Mercer
UK Mathematician
1883-1932



- Theorem (Mercer)

Let $K : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$ be given. Then for K to be a valid (Mercer) kernel, it is necessary and sufficient that for any $\{x^{(1)}, \dots, x^{(m)}\}$, $m < \infty$, the corresponding kernel matrix is symmetric positive semi-definite.

- Example valid kernels

- RBF kernel $K(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right)$
- Simple polynomial kernel $K(x, z) = (x^\top z)^d$
- Cosine similarity kernel $K(x, z) = \frac{x^\top z}{\|x\| \cdot \|z\|}$

Sigmoid Kernel

$$K(x, z) = \tanh(\alpha x^\top z + c)$$

$$\tanh(b) = \frac{1 - e^{-2b}}{1 + e^{-2b}}$$

- Neural networks use sigmoid as activation function
- SVM with a sigmoid kernel is equivalent to a 2-layer perceptron

(We shall return to this after the study of neural networks)

Generalized Linear Models

Review: Linear Regression

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \\ \vdots \\ \mathbf{x}^{(n)} \end{bmatrix} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & \dots & x_d^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & \dots & x_d^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^{(n)} & x_2^{(n)} & x_3^{(n)} & \dots & x_d^{(n)} \end{bmatrix} \quad \boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_d \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

- Prediction $\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\theta} = \begin{bmatrix} \mathbf{x}^{(1)}\boldsymbol{\theta} \\ \mathbf{x}^{(2)}\boldsymbol{\theta} \\ \vdots \\ \mathbf{x}^{(n)}\boldsymbol{\theta} \end{bmatrix}$

- Objective $J(\boldsymbol{\theta}) = \frac{1}{2}(\mathbf{y} - \hat{\mathbf{y}})^\top (\mathbf{y} - \hat{\mathbf{y}}) = \frac{1}{2}(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})$

Review: Matrix Form of Linear Reg.

- Objective

$$J(\boldsymbol{\theta}) = \frac{1}{2}(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) \quad \min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

- Gradient

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\mathbf{X}^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})$$

- Solution

$$\begin{aligned} \frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \mathbf{0} &\rightarrow \mathbf{X}^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) = \mathbf{0} \\ &\rightarrow \mathbf{X}^\top \mathbf{y} = \mathbf{X}^\top \mathbf{X} \boldsymbol{\theta} \\ &\rightarrow \hat{\boldsymbol{\theta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} \end{aligned}$$

Generalized Linear Models

- Dependence

$$y = f(\theta^\top \phi(x))$$

- Feature mapping function $\phi(x) : \mathbb{R}^d \mapsto \mathbb{R}^h$
- Mapped feature matrix $\Phi_{n \times h}$

$$\Phi = \begin{bmatrix} \phi(x^{(1)}) \\ \phi(x^{(2)}) \\ \vdots \\ \phi(x^{(i)}) \\ \vdots \\ \phi(x^{(n)}) \end{bmatrix} = \begin{bmatrix} \phi_1(x^{(1)}) & \phi_2(x^{(1)}) & \cdots & \phi_h(x^{(1)}) \\ \phi_1(x^{(2)}) & \phi_2(x^{(2)}) & \cdots & \phi_h(x^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(x^{(i)}) & \phi_2(x^{(i)}) & \cdots & \phi_h(x^{(i)}) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(x^{(n)}) & \phi_2(x^{(n)}) & \cdots & \phi_h(x^{(n)}) \end{bmatrix}$$

Matrix Form of Kernel Linear Regression

- Objective

$$J(\boldsymbol{\theta}) = \frac{1}{2}(\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\theta})^\top (\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\theta}) \quad \min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

- Gradient

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\boldsymbol{\Phi}^\top (\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\theta})$$

- Solution

$$\begin{aligned} \frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \mathbf{0} &\rightarrow \boldsymbol{\Phi}^\top (\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\theta}) = \mathbf{0} \\ &\rightarrow \boldsymbol{\Phi}^\top \mathbf{y} = \boldsymbol{\Phi}^\top \boldsymbol{\Phi} \boldsymbol{\theta} \\ &\rightarrow \hat{\boldsymbol{\theta}} = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \mathbf{y} \end{aligned}$$

Matrix Form of Kernel Linear Regression

- With the Algebra trick

$$(\mathbf{P}^{-1} + \mathbf{B}^{\top} \mathbf{R}^{-1} \mathbf{B})^{-1} \mathbf{B}^{\top} \mathbf{R}^{-1} = \mathbf{P} \mathbf{B}^{\top} (\mathbf{B} \mathbf{P} \mathbf{B}^{\top} + \mathbf{R})^{-1}$$

- The optimal parameters with L2 regularization

$$\begin{aligned} \hat{\boldsymbol{\theta}} &= (\boldsymbol{\Phi}^{\top} \boldsymbol{\Phi} + \lambda \mathbf{I}_h)^{-1} \boldsymbol{\Phi}^{\top} \mathbf{y} \\ &= \boldsymbol{\Phi}^{\top} (\boldsymbol{\Phi} \boldsymbol{\Phi}^{\top} + \lambda \mathbf{I}_n)^{-1} \mathbf{y} \end{aligned}$$

for prediction, we never actually need access $\boldsymbol{\Phi}$

$$\begin{aligned} \hat{\mathbf{y}} &= \boldsymbol{\Phi} \hat{\boldsymbol{\theta}} = \boldsymbol{\Phi} \boldsymbol{\Phi}^{\top} (\boldsymbol{\Phi} \boldsymbol{\Phi}^{\top} + \lambda \mathbf{I}_n)^{-1} \mathbf{y} \\ &= \mathbf{K} (\mathbf{K} + \lambda \mathbf{I}_n)^{-1} \mathbf{y} \end{aligned}$$

where the kernel matrix $\mathbf{K} = \{K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})\}$