Lecture 6: Support Vector Machine

Feng Li

Shandong University

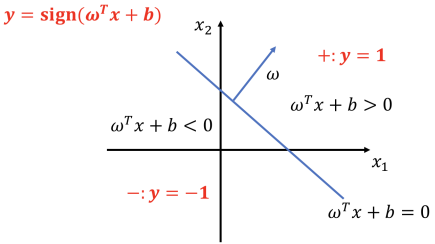
fli@sdu.edu.cn

December 28, 2021

Outline

Hyperplane

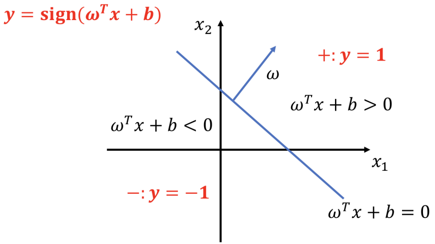
- Separates a *n*-dimensional space into two half-spaces



- Defined by an outward pointing normal vector $\omega \in \mathbb{R}^n$
- Assumption: The hyperplane passes through origin. If not,
  - have a bias term b; we will then need both ω and b to define it
  - $b > 0$ means moving it parallely along ω ($b < 0$ means in opposite direction)

Support Vector Machine

- A hyperplane based linear classifier defined by ω and b
- Prediction rule: $y = \mathrm{sign}(\omega^T x + b)$

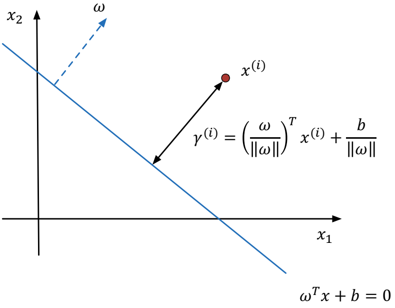


- Given: Training data $\{(x^{(i)}, y^{(i)})\}_{i=1,\cdots,m}$
- Goal: Learn ω and b that achieve the maximum **margin**
- For now, assume that entire training data are correctly classified by (ω, b)
  - Zero loss on the training examples (non-zero loss later)

Margin

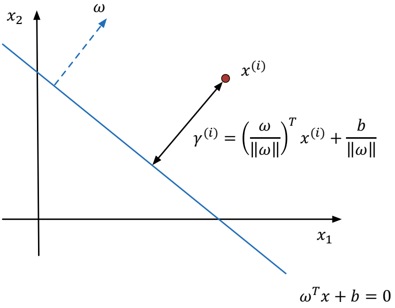
- Hyperplane: $\omega^T x + b = 0$, where ω is the normal vector
- The margin $\gamma^{(i)}$ is the *signed* distance between $x^{(i)}$ and the hyperplane

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



Margin (Contd.)

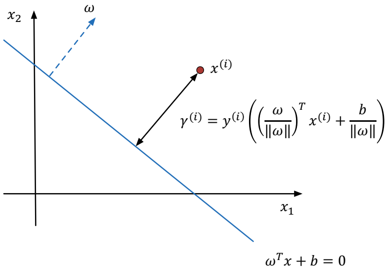
- Hyperplane: $\omega^T x + b = 0$, where ω is the normal vector
- The margin $\gamma^{(i)}$ is the distance between $x^{(i)}$ and the hyperplane
- Now, the margin is signed
  - If $y^{(i)} = 1$, $\gamma^{(i)} \geq 0$; otherwise, $\gamma^{(i)} < 0$



Margin (Contd.)

- Geometric margin

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

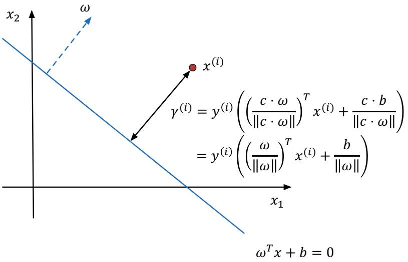


Margin (Contd.)

- Geometric margin

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

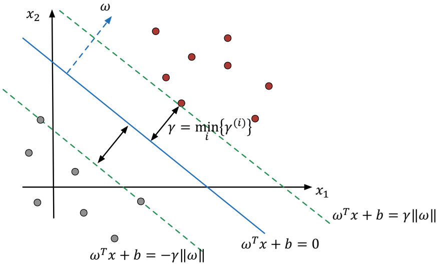
- Scaling (ω, b) does not change $\gamma^{(i)}$



Margin (Contd.)

- Geometric margin $\gamma^{(i)} = y^{(i)} \left((\omega/\|\omega\|)^T x^{(i)} + b/\|\omega\| \right)$
- Scaling (ω, b) does not change $\gamma^{(i)}$
- With respect to the whole training set, the margin is written as

$$\gamma = \min_i \gamma^{(i)}$$



Maximizing The Margin

- The hyperplane actually serves as a decision boundary to differentiating positive labels from negative labels
- We make more confident decision if larger margin is given, i.e., the data sample is further away from the hyperplane
- There exist a infinite number of hyperplanes, but which one is the best?

$$\max_{\omega, b} \min_i \{\gamma^{(i)}\}$$

Maximizing The Margin (Contd.)

- There exist a infinite number of hyperplanes, but which one is the best?

$$\max_{\omega,b} \quad \min_i \{\gamma^{(i)}\}$$

- It is equivalent to

$$\begin{aligned} \max_{\gamma,\omega,b} \quad & \gamma \\ s.t. \quad & \gamma^{(i)} \geq \gamma, \quad \forall i \end{aligned}$$

- Since

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

  the constraint becomes

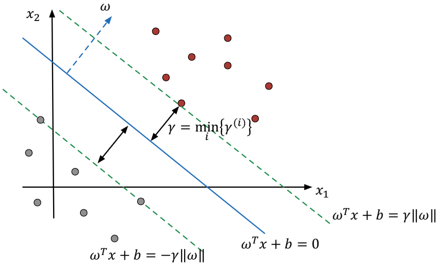
$$y^{(i)}(\omega^T x^{(i)} + b) \geq \gamma \|\omega\|, \ \forall i$$

Maximizing The Margin (Contd.)

- Formally,

$$\max_{\gamma,\omega,b} \quad \gamma$$

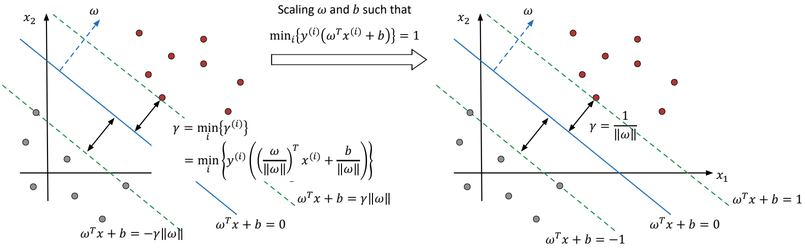
$$s.t. \quad y^{(i)}(\omega^T x^{(i)} + b) \geq \gamma\|\omega\|, \ \forall i$$



Maximizing The Margin (Contd.)

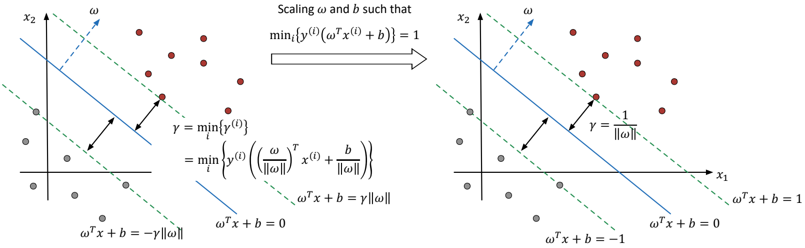
- Scaling (ω, b) such that $\min_i\{y^{(i)}(\omega^T x^{(i)} + b)\} = 1$,

$$\gamma = \min_i \left\{ y^{(i)} \left(\left(\frac{\omega}{\|\omega\|} \right)^T x^{(i)} + \frac{b}{\|\omega\|} \right) \right\} = \frac{1}{\|\omega\|}$$



- The problem becomes

$$\max_{\omega,b} \quad 1/\|\omega\|$$
$$s.t. \quad y^{(i)}(\omega^T x^{(i)} + b) \geq 1, \forall i$$



Support Vector Machine (Primal Form)

- Maximizing $1/\|\omega\|$ is equivalent to minimizing $\|\omega\|^2 = \omega^T\omega$

$$\begin{aligned} \min_{\omega,b} \quad & \omega^T\omega \\ s.t. \quad & y^{(i)}(\omega^T x^{(i)} + b) \geq 1, \forall i \end{aligned}$$

- This is a quadratic programming (QP) problem!
  - Interior point method
    (https://en.wikipedia.org/wiki/Interior-point_method)
  - Active set method
    (https://en.wikipedia.org/wiki/Active_set_method)
  - Gradient projection method
    (http://www.ifp.illinois.edu/~angelia/L13_constrained_gradient.pdf)
  - ...
- Existing generic QP solvers is of low efficiency, especially in face of a large training set

Convex Optimization Review

- Optimization Problem
- Lagrangian Duality
- KKT Conditions
- Convex Optimization

S. Boyd and L. Vandenberghe, 2004. Convex Optimization. Cambridge university press.

Optimization Problems

- Considering the following optimization problem

$$\begin{aligned} \min_{\omega} \quad & f(\omega) \\ s.t. \quad & g_i(\omega) \leq 0, i = 1, \cdots, k \\ & h_j(\omega) = 0, j = 1, \cdots, l \end{aligned}$$

with variable $\omega \in \mathbb{R}^n$, domain $\mathcal{D} = \bigcap_{i=1}^{k} \textbf{dom} g_i \cap \bigcap_{j=1}^{l} \textbf{dom} h_j$, optimal value p^*

  - Objective function $f(\omega)$
  - k inequality constraints $g_i(\omega) \leq 0, i = 1, \cdots, k$
  - l equality constraints $h_j(\omega) = 0, j = 1, \cdots, l$

Lagrangian

- Lagrangian: $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^k \times \mathbb{R}^l \rightarrow \mathbb{R}$, with $\mathbf{dom}\mathcal{L} = \mathcal{D} \times \mathbb{R}^k \times \mathbb{R}^l$

$$\mathcal{L}(\omega, \alpha, \beta) = f(\omega) + \sum_{i=1}^{k} \alpha_i g_i(\omega) + \sum_{j=1}^{l} \beta_j h_j(\omega)$$

  - Weighted sum of objective and constraint functions
  - α_i is Lagrange multiplier associated with $g_i(\omega) \leq 0$
  - β_j is Lagrange multiplier associated with $h_j(\omega) = 0$

Lagrange Dual Function

- The Lagrange dual function $\mathcal{G} : \mathbb{R}^k \times \mathbb{R}^l \to \mathbb{R}$

$$\begin{aligned} \mathcal{G}(\alpha, \beta) &= \inf_{\omega \in \mathcal{D}} \mathcal{L}(\omega, \alpha, \beta) \\ &= \inf_{\omega \in \mathcal{D}} \left(f(\omega) + \sum_{i=1}^{k} \alpha_i g_i(\omega) + \sum_{j=1}^{l} \beta_j h_j(\omega) \right) \end{aligned}$$

- $\mathcal{G}$ is concave, can be $-\infty$ for some α, β

The Lower Bounds Property

- If $\alpha \succeq 0$, then $\mathcal{G}(\alpha, \beta) \leq p^*$, where p^* is the optimal value of the primal problem
- Proof: If $\tilde{\omega}$ is feasible and $\alpha \succeq 0$, then

$$f(\tilde{\omega}) \geq \mathcal{L}(\tilde{\omega}, \alpha, \beta) \geq \inf_{\omega \in \mathcal{D}} \mathcal{L}(\omega, \alpha, \beta) = \mathcal{G}(\alpha, \beta)$$

minimizing over all feasible $\tilde{\omega}$ gives $p^* \geq \mathcal{G}(\alpha, \beta)$

Lagrange Dual Problem

- Lagrange dual problem

$$\begin{aligned} \mathsf{max}_{\alpha,\beta} \quad & \mathcal{G}(\alpha,\beta) \\ s.t. \quad & \alpha \succeq 0, \quad \forall i = 1, \cdots, k \end{aligned}$$

- Find the best low bound on p^*, obtained from Lagrange dual function
- A convex optimization problem (optimal value denoted by d^*)
- α, β are dual feasible if $\alpha \succeq 0$, $(\alpha,\beta) \in$ **dom** $\mathcal{G}$ and $\mathcal{G} > -\infty$
- Often simplified by making implicit constraint $(\alpha,\beta) \in$ **dom** $\mathcal{G}$ explicit

Weak Duality

- Weak duality: $d^* \leq p^*$
  - Always holds
  - Can be used to find nontrivial lower bounds for difficult problems
  - Optimal duality gap: $p^* - d^*$

Complementary Slackness

- Let ω^* be a primal optimal point and (α^*, β^*) be a dual optimal point
- If strong duality holds, then

$$\alpha_i^* g_i(\omega^*) = 0$$

for $\forall i = 1, 2, \cdots, k$

Complementary Slackness (Proof)

- We have

$$
\begin{aligned}
f(\omega^*) &= \mathcal{G}(\alpha^*, \beta^*) \\
&= \inf_{\omega} \left(f(\omega) + \sum_{i=1}^{k} \alpha_i^* g_i(\omega) + \sum_{j=1}^{l} \beta_j^* h_j(\omega) \right) \\
&\leq f(\omega^*) + \sum_{i=1}^{k} \alpha_i^* g_i(\omega^*) + \sum_{j=1}^{l} \beta_j^* h_j(\omega^*) \leq f(\omega^*)
\end{aligned}
$$

- The last two inequalities hold with equality, such that we have

$$\sum_{i=1}^{k} \alpha_i^* g_i(\omega^*) = 0$$

- Since each term, i.e., $\alpha_i^* g_i(\omega^*)$, is nonpositive, we thus conclude

$$\alpha_i^* g_i(\omega^*) = 0, \quad \forall i = 1, 2, \cdots, k$$

Karush-Kuhn-Tucker (KKT) Conditions

- Let ω^* and (α^*, β^*) by any primal and dual optimal points wither zero duality gap (i.e., the strong duality holds), the following conditions should be satisfied
  - Stationarity: Gradient of Lagrangian with respect to ω vanishes

$$\bigtriangledown f(\omega^*) + \sum_{i=1}^{k} \alpha_i \bigtriangledown g_i(\omega^*) + \sum_{j=1}^{l} \beta_j \bigtriangledown h_j(\omega^*) = 0$$

  - Primal feasibility

$$g_i(\omega^*) \leq 0, \ \forall i = 1, \cdots, k$$
$$h_j(\omega^*) = 0, \ \forall j = 1, \cdots, l$$

  - Dual feasibility

$$\alpha_i^* \geq 0, \ \forall i = 1, \cdots, k$$

  - Complementary slackness

$$\alpha_i^* g_i(\omega^*) = 0, \ \forall i = 1, \cdots, k$$

Convex Optimization Problem

- Problem Formulation

$$\begin{aligned} \min_{\omega} \quad & f(\omega) \\ s.t. \quad & g_i(\omega) \leq 0, i = 1, \cdots, k \\ & A\omega - b = 0 \end{aligned}$$

  - f and g_i $(i = 1, \cdots, k)$ are convex
  - A is a $l \times n$ matrix, $b \in \mathbb{R}^l$

Weak Duality V.s. Strong Duality

- Weak duality: $d^* \leq p^*$
  - Always holds
  - Can be used to find nontrivial lower bounds for difficult problems
- Strong duality: $d^* = p^*$
  - Does not hold in general
  - (Usually) holds for convex problems
  - Conditions that guarantee strong duality in convex problems are called **constraint qualifications**

Slater's Constraint Qualification

- Strong duality holds for a convex prblem

$$\begin{aligned} \min_{\omega} \quad & f(\omega) \\ s.t. \quad & g_i(\omega) \leq 0, i = 1, \cdots, k \\ & A\omega - b = 0 \end{aligned}$$

if it is strictly feasible, i.e.,

$$\exists \omega \in \mathbf{relint}\mathcal{D} : g_i(\omega) < 0, i = 1, \cdots, m, A\omega = b$$

KKT Conditions for Convex Optimization

- For convex optimization problem, the KKT conditions are also sufficient for the points to be primal and dual optimal
  - Suppose $\widetilde{\omega}$, $\widetilde{\alpha}$, and $\widetilde{\beta}$ are any points satisfying the following KKT conditions

$$
\begin{aligned}
& g_i(\widetilde{\omega}) \leq 0, \ \forall i = 1, \cdots, k \\
& h_j(\widetilde{\omega}) = 0, \ \forall j = 1, \cdots, l \\
& \widetilde{\alpha}_i \geq 0, \ \forall i = 1, \cdots, k \\
& \widetilde{\alpha}_i g_i(\widetilde{\omega}) = 0, \ \forall i = 1, \cdots, k \\
& \nabla f(\widetilde{\omega}) + \sum_{i=1}^{k} \widetilde{\alpha}_i \nabla g_i(\widetilde{\omega}) + \sum_{j=1}^{l} \widetilde{\beta}_j \nabla h_j(\widetilde{\omega}) = 0
\end{aligned}
$$

  then they are primal and dual optimal with strong duality holding

Optimal Margin Classifier

- Primal (convex) problem formulation

$$\begin{aligned} \min_{\omega,b} \quad & \frac{1}{2}\|\omega\|^2 \\ s.t. \quad & y^{(i)}(\omega^T x^{(i)} + b) \geq 1, \quad \forall i \end{aligned}$$

- The Lagrangian

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

- The Lagrange dual function

$$\mathcal{G}(\alpha) = \inf_{\omega,b} \mathcal{L}(\omega, b, \alpha)$$

Optimal Margin Classifier

- Dual problem formulation

$$
\begin{aligned}
\max_{\alpha} \quad & \inf_{\omega, b} \mathcal{L}(\omega, b, \alpha) \\
s.t. \quad & \alpha_i \geq 0, \quad \forall i
\end{aligned}
$$

- The Lagrangian

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

- The Lagrange dual function

$$
\mathcal{G}(\alpha) = \inf_{\omega, b} \mathcal{L}(\omega, b, \alpha)
$$

Optimal Margin Classifier (Contd.)

- Dual problem formulation

$$\begin{aligned} \max_{\alpha} \quad & \mathcal{G}(\alpha) = \inf_{\omega, b} \mathcal{L}(\omega, b, \alpha) \\ s.t. \quad & \alpha_i \geq 0 \ \ \forall i \end{aligned}$$

Optimal Margin Classifier (Contd.)

- According to KKT conditions, minimizing $\mathcal{L}(\omega, b, \alpha)$ over ω and b

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

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

- The Lagrange dual function becomes

$$\mathcal{G}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)}$$

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

Optimal Margin Classifier (Contd.)

- Dual problem formulation

$$
\begin{aligned}
\max_{\alpha} \quad & \mathcal{G}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)} \\
s.t. \quad & \alpha_i \geq 0 \ \ \forall i \\
& \sum_{i=1}^{m} \alpha_i y^{(i)} = 0
\end{aligned}
$$

- It is a *convex* optimization problem, so the strong duality ($p^* = d^*$) holds and the KKT conditions are respected
- Quadratic Programming problem in α
  - Several off-the-shelf solvers exist to solve such QPs
  - Some examples: quadprog (MATLAB), CVXOPT, CPLEX, IPOPT, etc.

SVM: The Solution

- Once we have the α^*,

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

- Given ω^*, how to calculate the optimal value of b?

SVM: The Solution

- Since $\alpha_i^*(y^{(i)}({\omega^*}^T x^{(i)} + b) - 1) = 0, \;$ for $\forall i$, we have

$$y^{(i)}({\omega^*}^T x^{(i)} + b^*) = 1$$

  for $\{i : \alpha_i^* > 0\}$

- Then, for $\forall i$ such that $\alpha_i^* > 0$, we have

$$b^* = y^{(i)} - {\omega^*}^T x^{(i)}$$

- For robustness, we calculated the optimal value for b by taking the average

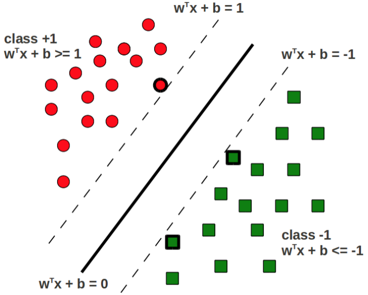
$$b^* = \frac{\sum_{i:\alpha_i^*>0}(y^{(i)} - {\omega^*}^T x^{(i)})}{\sum_{i=1}^m \mathbf{1}(\alpha_i^* > 0)}$$

SVM: The Solution (Contd.)

- Most α_i's in the solution are zero (sparse solution)
  - According to KKT conditions, for the optimal α_i's,

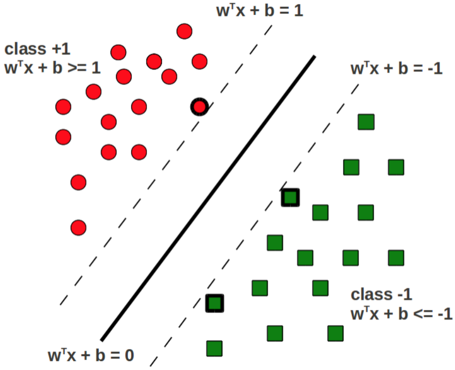
$$\alpha_i \left(1 - y^{(i)}(\omega^T x^{(i)} + b)\right) = 0$$

  - α_i is non-zero only if $x^{(i)}$ lies on the one of the two margin boundaries. i.e., for which $y^{(i)}(\omega^T x^{(i)} + b) = 1$



SVM: The Solution (Contd.)

- These data samples are called **support vector** (i.e., support vectors "support" the margin boundaries)

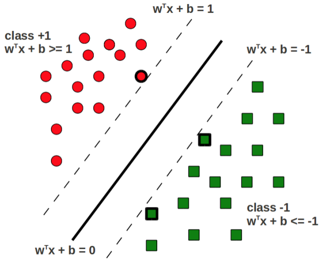


SVM: The Solution (Contd.)

- Redefine ω^*

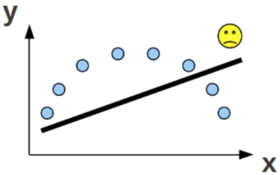
$$\omega^* = \sum_{s \in \mathcal{S}} \alpha_s^* y^{(s)} x^{(s)}$$

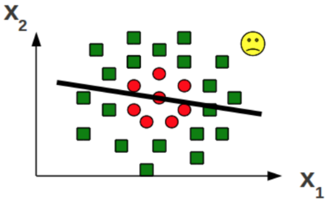
where $\mathcal{S}$ denotes the indices of the support vectors



Kernel Methods

- Motivation: Linear models (e.g., linear regression, linear SVM etc.) cannot reflect the nonlinear pattern in the data

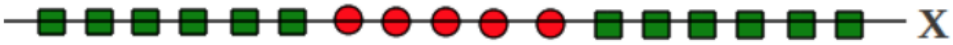




- Kernels: Make linear model work in nonlinear settings
  - By mapping data to higher dimensions where it exhibits linear patterns
  - Apply the linear model in the new input space
  - Mapping is equivalent to changing the feature representation

Feature Mapping

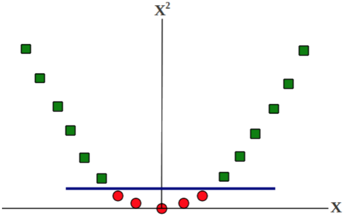
- Consider the following binary classification problem



  - Each sample is represented by a single feature x
  - No linear separator exists for this data

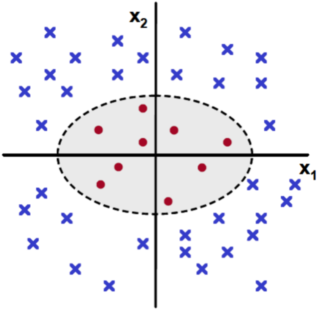
Feature Mapping (Contd.)

- Now map each example as $x \to \{x, x^2\}$
  - Each example now has two features ("derived" from the old representation)
- Data now becomes linearly separable in the new representation



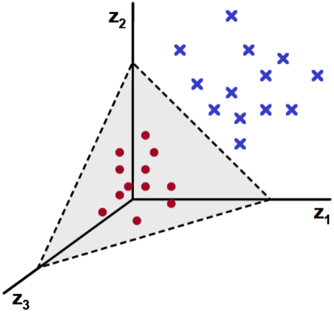
Feature Mapping (Contd.)

- Another example
  - Each sample is defined by $x = \{x_1, x_2\}$
  - No linear separator exists for this data



Feature Mapping (Contd.)

- Now map each example as $x = \{x_1, x_2\} \rightarrow z = \{x_1^2, \sqrt{2}x_1x_2, x_2^2\}$
  - Each example now has three features ("derived" from the old representation)
- Data now becomes linearly separable in the new representation



Feature Mapping (Contd.)

- Consider the follow feature mapping ϕ for an example $x = \{x_1, \cdots, x_n\}$

$$\phi : x \rightarrow \{x_1^2, x_2^2, \cdots, x_n^2, x_1x_2, x_1x_2, \cdots, x_1x_n, \cdots, x_{n-1}x_n\}$$

- It is an example of a quadratic mapping
  - Each new feature uses a pair of the original features

Feature Mapping (Contd.)

- Problem: Mapping usually leads to the number of features blow up!
  - Computing the mapping itself can be inefficient, especially when the new space is very high dimensional
  - Storing and using these mappings in later computations can be expensive (e.g., we may have to compute inner products in a very high dimensional space)
  - Using the mapped representation could be inefficient too
- Thankfully, kernels help us avoid both these issues!
  - The mapping does not have to be explicitly computed
  - Computations with the mapped features remain efficient

Kernels as High Dimensional Feature Mapping

- Let's assume we are given a function K (kernel) that takes as inputs x and z

$$
\begin{aligned}
K(x,z) &= (x^T z)^2 \\
&= (x_1 z_1 + x_2 z_2)^2 \\
&= x_1^2 z_1^2 + x_2^2 z_2^2 + 2 x_1 x_2 z_1 z_2 \\
&= (x_1^2, \sqrt{2} x_1 x_2, x_2^2)^T (z_1^2, \sqrt{2} z_1 z_2, z_2^2)
\end{aligned}
$$

- The above function K implicitly defines a mapping ϕ to a higher dim. space

$$\phi(x) = \{x_1^2, \sqrt{2} x_1 x_2, x_2^2\}$$

- Simply defining the kernel in a certain way gives a higher dim. mapping ϕ
  - The mapping does not have to be explicitly computed
  - Computations with the mapped features remain efficient

Kernels: Formal Definition

- Each kernel K has an associated feature mapping ϕ
- ϕ takes input $x \in \mathcal{X}$ (input space) and maps it to $\mathcal{F}$ (feature space)
- Kernel $K(x, z) = \phi(x)^T \phi(z)$ takes two inputs and gives their similarity in $\mathcal{F}$ space

$$\phi : \mathcal{X} \to \mathcal{F}$$
$$K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$

- $\mathcal{F}$ needs to be a vector space with a dot product defined upon it
  - Also called a Hilbert Space
- Can just any function be used as a kernel function?
  - No. It must satisfy Mercer's Condition

- For K to be a kernel function
  - There must exist a Hilbert Space $\mathcal{F}$ for which K defines a dot product
  - The above is true if K is a positive definite function

$$\int\int f(x)K(x,z)f(z)dxdz > 0 \quad (\forall f \in L_2)$$

for all functions f that are "square integrable", i.e.,

$$\int_{-\infty}^{\infty} f^2(x)dx < \infty$$

Mercer's Condition (Contd.)

- Let K_1 and K_2 be two kernel functions then the followings are as well:
  - Direct sum: $K(x, z) = K_1(x, z) + K_2(x, z)$
  - Scalar product: $K(x, z) = \alpha K_1(x, z)$
  - Direct product: $K(x, z) = K_1(x, z) K_2(x, z)$
  - Kernels can also be constructed by composing these rules

The Kernel Matrix

- For K to be a kernel function
  - The kernel function K also defines the Kernel Matrix over the data (also denoted by K)
  - Given m samples $\{x^{(1)}, x^{(2)}, \cdots, x^{(m)}\}$, the (i, j)-th entry of K is defined as

$$K_{i,j} = K(x^{(i)}, x^{(j)}) = \phi(x^{(i)})^T \phi(x^{(j)})$$

- $K_{i,j}$: Similarity between the i-th and j-th example in the feature space $\mathcal{F}$
- K: $m \times m$ matrix of pairwise similarities between samples in $\mathcal{F}$ space
- K is a symmetric matrix
- K is a positive semi-definite matrix

Some Examples of Kernels

- Linear (trivial) Kernal:

$$K(x, z) = x^T z$$

- Quadratic Kernel

$$K(x, z) = (x^T z)^2 \quad \text{or} \quad (1 + x^T z)^2$$

- Polynomial Kernel (of degree d)

$$K(x, z) = (x^T z)^d \quad \text{or} \quad (1 + x^T z)^d$$

- Gaussian Kernel

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

- Sigmoid Kernel

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

Using Kernels

- Kernels can turn a linear model into a nonlinear one
- Kernel $K(x, z)$ represents a dot product in some high dimensional feature space $\mathcal{F}$

$$K(x, z) = (x^T z)^2 \quad \text{or} \quad (1 + x^T z)^2$$

- Any learning algorithm in which examples only appear as dot products $({x^{(i)}}^T x^{(j)})$ can be kernelized (i.e., non-linearlized)
  - By replacing the ${x^{(i)}}^T x^{(j)}$ terms by $\phi(x^{(i)})^T \phi(x^{(j)}) = K(x^{(i)}, x^{(j)})$
- Most learning algorithms are like that
  - SVM, linear regression, etc.
  - Many of the unsupervised learning algorithms too can be kernelized (e.g., K-means clustering, Principal Component Analysis, etc.)

Kernelized SVM Training

- SVM dual Lagrangian

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

$$\text{s.t.} \quad \sum_{i=1}^{m} \alpha_i y^{(i)} = 0$$

$$\alpha_i \geq 0, \quad \forall i$$

Kernelized SVM Training (Contd.)

- Replacing $< x^{(i)}, x^{(j)} >$ by $\phi(x^{(i)})^T \phi(x^{(j)}) = K(x^{(i)}, x^{(j)}) = K_{ij}$

$$
\begin{aligned}
\max_{\alpha} \quad & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j K_{i,j} \\
\text{s.t.} \quad & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \\
& \alpha_i \geq 0, \ \forall i
\end{aligned}
$$

- SVM now learns a linear separator in the kernel defined feature space $\mathcal{F}$
  - This corresponds to a non-linear separator in the original space $\mathcal{X}$

Kernelized SVM Prediction

- Define the decision boundary ${\omega^*}^T\phi(x) + b^*$ in the higher-dimensional feature space

$$
\begin{aligned}
\omega^* &= \sum_{i:\alpha_i^*>0} \alpha_i^* y^{(i)} \phi(x^{(i)}) \\
b^* &= y^{(i)} - {\omega^*}^T \phi(x^{(i)}) \\
&= y^{(i)} - \sum_{j:\alpha_j^*>0} \alpha_j^* y^{(j)} \phi^T(x^{(j)}) \phi(x^{(i)}) \\
&= y^{(i)} - \sum_{j:\alpha_j^*>0} \alpha_j^* y^{(j)} K_{ij}
\end{aligned}
$$

Kernelized SVM Prediction (Contd.)

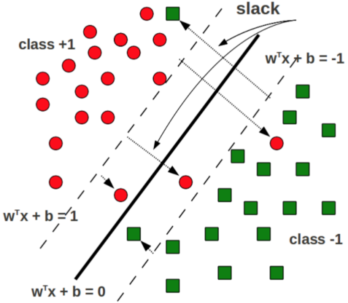
- Given a test data sample x

$$
\begin{aligned}
y &= \operatorname{sign}\left(\sum_{i:\alpha_i^*>0} \alpha_i^* y^{(i)} \phi(x^{(i)})^T \phi(x) + b^*\right) \\
&= \operatorname{sign}\left(\sum_{i:\alpha_i^*>0} \alpha_i^* y^{(i)} K(x^{(i)}, x) + b^*\right)
\end{aligned}
$$

- Kernelized SVM needs the support vectors at the test time (except when you can write $\phi(x)$ as an explicit, reasonably-sized vector)
  - In the unkernelized version $\omega = \sum_{i:\alpha_i^*>0} \alpha_i^* y^{(i)} x^{(i)} + b^*$ can be computed and stored as a $n \times 1$ vector, so the support vectors need not be stored

Soft-Margin SVM

- We allow some training examples to be misclassified, and some training examples to fall within the margin region



Soft-Margin SVM (Contd.)

- Recall that, for the separable case (training loss = 0), the constraints were

$$y^{(i)}(\omega^T x^{(i)} + b) \geq 1 \quad \mathit{for}\ \forall i$$

- For the non-separable case, we relax the above constraints as:

$$y^{(i)}(\omega^T x^{(i)} + b) \geq 1 - \xi_i \quad \mathit{for}\ \forall i$$

  - ξ_i is called slack variable
- Non-separable case
  - We will allow misclassified training samples, but we want the number of such samples to be minimized, by minimizing the sum of the slack variables $\sum_i \xi_i$

Soft-Margin SVM (Contd.)

- Reformulating the SVM problem by introducing slack variables ξ_i

$$
\begin{aligned}
\min_{\omega,b,\xi} \quad & \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{m}\xi_i \\
s.t. \quad & y^{(i)}(\omega^T x^{(i)} + b) \geq 1 - \xi_i, \quad \forall i = 1, \cdots, m \\
& \xi_i \geq 0, \quad \forall i = 1, \cdots, m
\end{aligned}
$$

- The parameter C controls the relative weighting between the following two goals
  - Small $C \Rightarrow \|\omega\|^2/2$ dominates $\Rightarrow$ prefer large margins
    - but allow potential large number of misclassified training examples
  - Large $C \Rightarrow C\sum_{i=1}^{m}\xi_i$ dominates $\Rightarrow$ prefer small number of misclassified examples
    - at the expense of having a small margin

Soft-Margin SVM (Contd.)

- Lagrangian

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

- KKT conditions (the optimal values of ω, b, ξ, α, and r should satisfy the following conditions)
  - $\nabla_\omega \mathcal{L}(\omega, b, \xi, \alpha, r) = 0 \quad \Rightarrow \quad \omega^* = \sum_{i=1}^m \alpha_i^* y^{(i)} x^{(i)}$
  - $\nabla_b \mathcal{L}(\omega, b, \xi, \alpha, r) = 0 \quad \Rightarrow \quad \sum_{i=1}^m \alpha_i^* y^{(i)} = 0$
  - $\nabla_{\xi_i} \mathcal{L}(\omega, b, \xi, \alpha, r) = 0 \quad \Rightarrow \quad \alpha_i^* + r_i^* = C$, for $\forall i$
  - $\alpha_i^*, r_i^*, \xi_i^* \geq 0$, for $\forall i$
  - $y^{(i)}({\omega^*}^T x^{(i)} + b^*) + \xi_i^* - 1 \geq 0$, for $\forall i$
  - $\alpha_i^*(y^{(i)}(\omega^* x^{(i)} + b^*) + \xi_i^* - 1) = 0$, for $\forall i$
  - $r_i^* \xi_i^* = 0$, for $\forall i$

Soft-Margin SVM (Contd.)

- Dual problem

$$
\begin{aligned}
\max_{\alpha} \quad & \mathcal{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j < x^{(i)}, x^{(j)} > \\
s.t. \quad & 0 \leq \alpha_i \leq C, \quad \forall i = 1, \cdots, m \\
& \sum_{i=1}^{m} \alpha_i y^{(i)} = 0
\end{aligned}
$$

- Use existing QP solvers to address the above optimization problem

Soft-Margin SVM (Contd.)

- Optimal values for α_i ($i = 1, \cdots, m$)
- How to calculate the optimal values of ω and b?
  - Use KKT conditions !

Soft-Margin SVM (Contd.)

- By resolving the above optimization problem, we get the optimal value of α_i $(i = 1, \cdots, m)$
- How to calculate the optimal values of ω and b?
  - According to the KKT conditions, we have

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

  - How about b^*?

Soft-Margin SVM (Contd.)

- Since $\alpha_i^* + r_i^* = C$, for $\forall i$, we have

$$r_i^* = C - \alpha_i^*, \ \forall i$$

- Since $r_i^* \xi_i^* = 0$, for $\forall i$, we have

$$(C - \alpha_i^*)\xi_i^* = 0, \ \forall i$$

- For $\forall i$ such that $\alpha_i^* \neq C$, we have $\xi_i = 0$, and thus

$$\alpha_i^*(y^{(i)}({\omega^*}^T x^{(i)} + b^*) - 1) = 0$$

Soft-Margin SVM (Contd.)

- For $\forall i$ such that $0 < \alpha_i^* < C$, we have

$$y^{(i)}({\omega^*}^T x^{(i)} + b^*) = 1$$

- Hence,

$${\omega^*}^T x^{(i)} + b^* = y^{(i)}$$

  for $\forall i$ such that $0 < \alpha_i^* < C$

- We finally calculate b as

$$b^* = \frac{\sum_{i:0<\alpha_i^*<C}(y^{(i)} - {\omega^*}^T x^{(i)})}{\sum_{i=1}^m \mathbf{1}(0 < \alpha_i^* < C)}$$

Soft-Margin SVM (Contd.)

- Soft-margin SVM classifier

$$
\begin{aligned}
y &= \operatorname{sign}\left(\omega^{*T} x + b^*\right) \\
&= \operatorname{sign}\left(\sum_{i=1}^{m} \alpha_i^* y^{(i)} < x^{(i)}, x > + b^*\right)
\end{aligned}
$$

Soft-Margin SVM (Contd.)

- Some useful corollaries according to the KKT conditions
  - When $\alpha_i^* = 0$, $y^{(i)}({\omega^*}^T x^{(i)} + b^*) \geq 1$
  - When $\alpha_i^* = C$, $y^{(i)}({\omega^*}^T x^{(i)} + b^*) \leq 1$
  - When $0 < \alpha_i^* < C$, $y^{(i)}({\omega^*}^T x^{(i)} + b^*) = 1$
- For $\forall i = 1, \cdots, m$, $x^{(i)}$ is
  - correctly classified if $\alpha_i^* = 0$
  - misclassified if $\alpha_i^* = C$
  - a support vector if $0 < \alpha_i^* < C$

Corollary

For $\forall i = 1, 2, \cdots, m$*, when* $\alpha_i^* = 0$*,* $y^{(i)}({\omega^*}^T x^{(i)} + b^*) \geq 1$*.*

Proof.

$$
\begin{aligned}
&\because \alpha_i^* = 0, \alpha_i^* + r_i^* = C \\
&\therefore r_i^* = C \\
&\because r_i^* \xi_i^* = 0 \\
&\therefore \xi_i^* = 0 \\
&\because y^{(i)}({\omega^*}^T x^{(i)} + b^*) + \xi_i^* - 1 \geq 0 \\
&\therefore y^{(i)}({\omega^*}^T x^{(i)} + b^*) \geq 1
\end{aligned}
$$

□

Soft-Margin SVM (Contd.)

Corollary

For $\forall i = 1, 2, \cdots, m$, *when* $\alpha_i^* = C$, $y^{(i)}({\omega^*}^T x^{(i)} + b^*) \leq 1$

Proof.

$$
\begin{aligned}
&\because \alpha_i^* = C,\ \alpha_i^*(y^{(i)}({\omega^*}^T x^{(i)} + b^*) + \xi_i^* - 1) = 0 \\
&\therefore y^{(i)}({\omega^*}^T x^{(i)} + b^*) + \xi_i^* - 1 = 0 \\
&\because \xi_i^* \geq 0 \\
&\therefore y^{(i)}({\omega^*}^T x^{(i)} + b^*) = 1 - \xi^* \leq 1
\end{aligned}
$$

□

Soft-Margin SVM (Contd.)

Corollary

For $\forall i = 1, 2, \cdots, m$, *when* $0 < \alpha_i^* < C$, $y^{(i)}({\omega^*}^T x^{(i)} + b^*) = 1$.

Proof.

$$
\begin{aligned}
&\because 0 < \alpha_i^* < C, \alpha_i^* + r_i^* = C \\
&\therefore 0 < r_i^* < C \\
&\because r_i^* \xi_i^* = 0 \\
&\therefore \xi_i^* = 0 \\
&\because 0 < \alpha_i^* < C, \alpha_i^*(y^{(i)}({\omega^*}^T x^{(i)} + b) + \xi_i^* - 1) = 0 \\
&\therefore y^{(i)}({\omega^*}^T x^{(i)} + b^*) + \xi_i^* - 1 = 0 \\
&\therefore y^{(i)}({\omega^*}^T x^{(i)} + b^*) = 1
\end{aligned}
$$

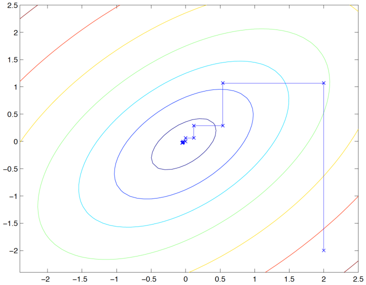
□

Coordinate Ascent Algorithm

- Consider the following unconstrained optimization problem

$$\max_{\alpha} \mathcal{J}(\alpha_1, \alpha_2, \cdots, \alpha_m)$$

- Repeat the following step until convergence
  - For each i, $\alpha_i = \arg\max_{\alpha_i} \mathcal{J}(\alpha_1, \cdots, \alpha_{i-1}, \alpha_i, \alpha_{i+1}, \cdots, \alpha_m)$
- For some α_i, fix the other variables and re-optimize $\mathcal{J}(\alpha)$ with respect to α_i



Sequential Minimal Optimization (SMO) Algorithm

- Coordinate ascent algorithm cannot be applied since $\sum_{i=0}^{m} \alpha_i y^{(i)} = 0$
- The basic idea of SMO

Algorithm 1 SMO algorithm

1: **Given** a starting point $\alpha \in$ **dom** $\mathcal{J}$
2: **repeat**
3: Select some pair of α_i and α_j to update next (using a heuristic that tries to pick the two α's);
4: Re-optimize $\mathcal{J}(\alpha)$ with respect to α_i and α_j, while holding all the other α_k's $(k \neq i, j)$ fixed
5: **until** convergence criterion is satisfied

SMO Algorithm (Contd.)

- Convergence criterion

$$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0, \quad 0 \leq \alpha_i \leq C, \quad \forall i = 1, \cdots, m$$

$$y^{(i)} \left(\sum_{j=1}^{m} \alpha_j y^{(j)} < x^{(i)}, x^{(j)} > + b \right) = \begin{cases} \geq 1, & \forall i : \alpha_i = 0 \\ = 1, & \forall i : 0 < \alpha_i < C \\ \leq 1, & \forall i : \alpha_i = C \end{cases}$$

SMO Algorithm (Contd.)

- Take α_1 and α_2 for example

$$\begin{aligned}\mathcal{J}(\alpha_1^+, \alpha_2^+) &= \alpha_1^+ + \alpha_2^+ - \frac{1}{2}K_{11}{\alpha_1^+}^2 - \frac{1}{2}K_{22}{\alpha_2^+}^2 - SK_{12}\alpha_1^+\alpha_2^+ \\ &\quad -y^{(1)}V_1\alpha_1^+ - y^{(2)}V_2\alpha_2^+ + \Psi\end{aligned}$$

where

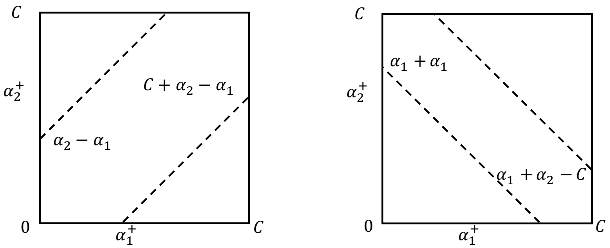
$$\begin{cases} K_{ij} = < x^{(i)}, x^{(j)} > \\ S = y^{(1)}y^{(2)} \\ \Psi = \sum_{i=3}^m \alpha_i - \frac{1}{2}\sum_{i=3}^m \sum_{j=3}^m y^{(i)}y^{(j)}\alpha_i\alpha_j K_{ij} \\ V_i = \sum_{j=3}^m y^{(j)}\alpha_j K_{ij} \end{cases}$$

SMO Algorithm (Contd.)

- Define

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

- Lower bound L and upper bound H for α_2^+:
  - When $y^{(1)} y^{(2)} = -1$, $H = \min\{C, C+\alpha_2-\alpha_1\}$ and $L = \max\{0, \alpha_2-\alpha_1\}$
  - When $y^{(1)} y^{(2)} = 1$, $H = \min\{C, \alpha_2+\alpha_1\}$ and $L = \max\{0, \alpha_1+\alpha_2-C\}$



(a) $y^{(1)} y^{(2)} = -1$ (b) $y^{(1)} y^{(2)} = 1$

SMO Algorithm (Contd.)

- Address the following optimization problem

$$\begin{array}{ll} \max_{\alpha_2} & \mathcal{J}(\alpha_1^+ = (\zeta - \alpha_2^+ y^{(2)}) y^{(1)}, \alpha_2^+) \\ \text{s.t.} & L \le \alpha_2^+ \le H \end{array}$$

- Find the extremum by letting the first derivative (with respect to α_2^+) to be zero as follows

$$\begin{aligned} & \frac{\partial}{\partial \alpha_2^+} f((\zeta - \alpha_2^+ y^{(2)}) y^{(1)}, \alpha_2^+) \\ = \; & -S + 1 + SK_{11}(\zeta y^{(1)} - S\alpha_2^+) - K_{22}\alpha_2^+ - SK_{12}(\zeta y^{(1)} - S\alpha_2^+) \\ & + K_{12}\alpha_2^+ + y^{(2)} V_1 - y^{(2)} V_2 = 0 \end{aligned}$$

SMO Algorithm (Contd.)

- By assuming $E_i = \sum_{j=1}^{m} y^{(j)}\alpha_j K_{ij} + b - y^{(i)}$,

$$\alpha_2^+ = \alpha_2 + \frac{y^{(2)}(E_1 - E_2)}{K_{11} - 2K_{12} + K_{22}}$$

- Since α_2^+ should be in the range of $[L, H]$,

$$\alpha_2^+ = \begin{cases} H, & \alpha_2^+ > H \\ \alpha_2^+, & L \le \alpha_2^+ \le H \\ L, & \alpha_2^+ < L \end{cases}$$

SMO Algorithm (Contd.)

- Updating b to verify if the convergence criterion is satisfied
  - When $0 < \alpha_1^+ < C$,

$$b_1^+ = -E_1 - y^{(1)} K_{11}(\alpha_1^+ - \alpha_1) - y^{(2)} K_{21}(\alpha_2^+ - \alpha_2) + b$$

  - When $0 < \alpha_2^+ < C$,

$$b_2^+ = -E_2 - y^{(1)} K_{12}(\alpha_1^+ - \alpha_1) - y^{(2)} K_{22}(\alpha_2^+ - \alpha_2) + b$$

  - when $0 < \alpha_1^+ < C$ and $0 < \alpha_2^+ < C$ both hold,

$$b^+ = b_1^+ = b_2^+$$

  - When α_1^+ and α_2^+ are on the bound (i.e., $\alpha_1 = 0$ or $\alpha_1 = C$ and $\alpha_2 = 0$ or $\alpha_2 = C$), all values between b_1^+ and b_2^+ satisfy the KKT conditions

$$b^+ = (b_1^+ + b_2^+)/2$$

SMO Algorithm (Contd.)

- Updating E_i

$$E_i^+ = \sum_{j=1}^{2} y^{(j)} \alpha_j^+ K_{ij} + \sum_{j=3}^{m} y^{(j)} \alpha_j^+ K_{ij} + b^+ - y^{(i)}$$

SMO Algorithm (Contd.)

- How to choose the target variable (i.e., α_1 and α_2 in our case)?
  - Both α_1 and α_2 should violate the KKT conditions
  - Since the step size of updating α_2 depends on $|E_1 - E_2|$, a greedy method suggests we should choose the one maximizing $|E_1 - E_2|$

Thanks!

Q & A