Machine Learning

Lecture 6: Support Vector Machine

Feng Li

fli@sdu.edu.cn
https://funglee.github.io

School of Computer Science and Technology
Shandong University

Fall 2017

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)

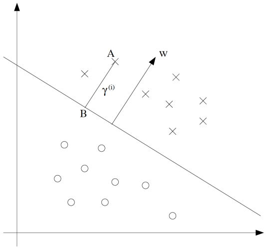
Support Vector Machine

- A hyperplane based linear classifier defined by ω and b
- Prediction rule: $y = sign(\omega^T x + b)$
- 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: $w^T x + b = 0$, where w is the normal vector
- The margin $\gamma^{(i)}$ is the distance between $x^{(i)}$ and the hyperplane

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



Margin

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

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

- Now, the margin is signed
  - If $y^{(i)} = 1$, $\gamma^{(i)} \geq 0$; otherwise, $\gamma^{(i)} < 0$

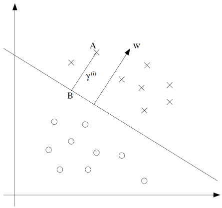
Margin

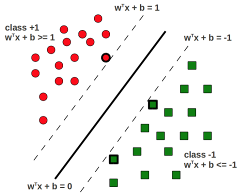
- Geometric margin

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

- Scaling (w, 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 new data is further away from the hyperplane
- There exist a infinite number of hyperplanes, but which one is the best?

$$\begin{aligned} &\max_{\gamma,w,b} \quad \gamma \\ &s.t. \;\; y^{(i)}(w^T x^{(i)} + b) \geq \gamma \|w\|, \quad \forall i \end{aligned}$$

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

Maximizing The Margin

- Scaling (w, b) such that $\min_i\{y^{(i)}(w^T x^{(i)} + b)\} = 1$, the representation of the margin becomes $1/\|w\|$

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

Support Vector Machine (Primal Form)

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

$$\min_{w,b} \quad w^T w$$
$$s.t. \quad y^{(i)}(w^T x^{(i)} + b) \geq 1, \quad \forall i$$

- 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 and Lagrange Duality Review

- Considering the following optimization problem

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

  with variable $w \in \mathbb{R}^n$, domain $\mathcal{D}$, optimal value p^*

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

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

  - 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 $\mathcal{G} : \mathbb{R}^k \times \mathbb{R}^l \to \mathbb{R}$

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

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

- Lower bounds property: If $\alpha \succeq 0$, then $\mathcal{G}(\alpha, \beta) \leq p^*$
  Proof: If $\tilde{\omega}$ is feasible and $\alpha \succeq 0$, then

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

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

Lagrange Dual Problem

- Lagrange dual problem

$$\begin{array}{cl} \max_{\alpha,\beta} & \mathcal{G}(\alpha,\beta) \\ s.t. & \alpha \succeq 0, \quad \forall i = 1, \cdots, k \end{array}$$

- 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 $\alpha \succeq 0$, $(\alpha, \beta) \in \mathbf{dom}\ \mathcal{G}$
- Often simplified by making implicit constraint $(\alpha, \beta) \in \mathbf{dom}\ \mathcal{G}$ explicit

Weak Duality V.s. Strong Duality

- Weak duality: $d^* \leq p^*$
  - Always holds (for convex and nonconvex problems)
  - Can be sued 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

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

if it is strictly feasible, i.e.,

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

Karush-Kuhn-Tucker (KKT) Conditions

- Let w^* 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

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

  - Primal feasibility

$$\begin{aligned} g_i(w^*) &\le 0,\ \forall i = 1, \cdots, k \\ h_j(w^*) &= 0,\ \forall j = 1, \cdots, l \end{aligned}$$

  - Dual feasibility

$$\alpha_i \ge 0,\ \forall i = 1, \cdots, k$$

  - Complementary slackness

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

Optimal Margin Classifier

- Primal 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

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

Optimal Margin Classifier (Contd.)

- Calculate the Lagrange dual function $\mathcal{G}(\alpha) = \inf_{w,b} \mathcal{L}(w, b, \alpha)$

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

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

- The Lagrange dual function

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

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 teh 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^*,

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

- Given w^*, 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

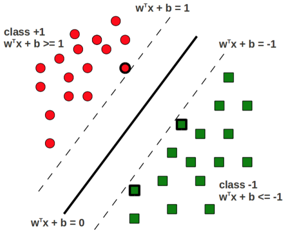
$$b^* = y^{(i)} - \omega^{*T}x^{(i)}$$

- 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, $\alpha_i[1 - y^{(i)}(w^T x^{(i)} + b)] = 0$
  - α_i is non-zero only if $x^{(i)}$ lies on the one of the two margin boundaries. i.e., for which $y^{(i)}(w^T x^{(i)} + b) = 1$
- These data samples are called **support vector** (i.e., support vectors "support" the margin boundaries)



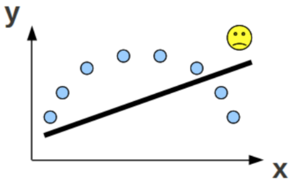
- Redefine w

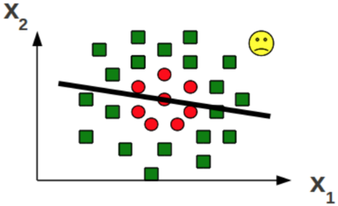
$$w = \sum_{s \in \mathcal{S}} \alpha_s y^{(s)} x^{(s)}$$

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

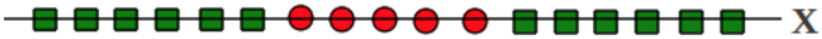




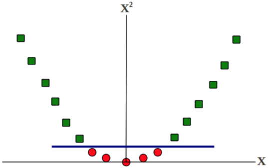
- Kernels: Make linear model work in nonlinear settings
  - By mapping data to higher dimensions where it exhibits linear patterns
  - Apply the linear model in the new input space
  - Mapping is equivalent to changing the feature representation

Feature Mapping

- Consider the following binary classification problem



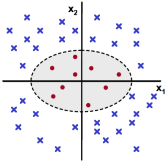
  - Each sample is represented by a single feature x
  - No linear separator exists for this data
- Now map each example as $x \to \{x, x^2\}$
  - Each example now has two features ("derived" from the old representation)
- Data now becomes linearly separable in the new representation



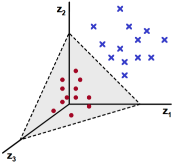
- Linear in the new representation $\equiv$ nonlinear in the old representation

Feature Mapping (Contd.)

- Another example



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

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

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

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

- 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 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
- Moreover the kernel $K(x,z)$ also computes the dot product $\phi(x)^T \phi(z)$

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

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., $\int f^2(x)dx < \infty$

- 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 definite matrix

Some Examples of Kernels

- Linear (trivial) Kernal:

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

- Quadratic Kernel

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

- Polynomial Kernel (of degree d)

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

- 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 $\mathcal{F}$

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

- 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

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

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

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

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

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

- 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

- Prediction for a test example x (assume $b = 0$)

$$y = \text{sign}(\omega^T x) = \text{sign}\left(\sum_{s \in \mathcal{S}} \alpha_s y^{(s)} {x^{(s)}}^T x\right)$$

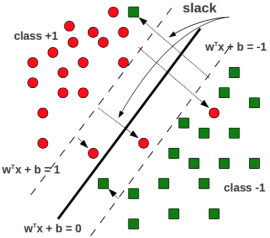
- Replacing each example with its feature mapped representation ($x \to \phi(x)$)

$$y = \text{sign}\left(\sum_{s \in \mathcal{S}} \alpha_s y^{(s)} {x^{(s)}}^T x\right) = \text{sign}\left(\sum_{s \in \mathcal{S}} \alpha_s y^{(s)} K(x^{(s)}, x)\right)$$

- 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 $\omega = \sum_{s \in \mathcal{S}} \alpha_s y^{(s)} x^{(s)}$ can be computed and stored as a $n \times 1$ vector, so the support vectors need not be stored

Regularized SVM

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



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

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

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

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

  - ξ_i is called slack variable

- Non-separable case: We will allow misclassified training examples
  - But we want their number to be minimized, by minimizing the sum of the slack variables $\sum_i \xi_i$
- Reformulating the SVM problem by introducing slack variables ξ_i

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

- 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

- Lagrangian

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

- KKT conditions
  - $\bigtriangledown_w\mathcal{L}(w,b,\xi,\alpha,r)=0 \quad\Rightarrow\quad w=\sum_{i=1}^m \alpha_i y^{(i)}x^{(i)}$
  - $\bigtriangledown_b\mathcal{L}(w,b,\xi,\alpha,r)=0 \quad\Rightarrow\quad \sum_{i=1}^m \alpha_i y^{(i)}=0$
  - $\bigtriangledown_{\xi_i}\mathcal{L}(w,b,\xi,\alpha,r)=0 \quad\Rightarrow\quad \alpha_i+r_i=C$, for $\forall i$
  - $\alpha_i,r_i,\xi_i\geq 0$, for $\forall i$
  - $y^{(i)}(w^Tx^{(i)}+b)+\xi_i-1\geq 0$, for $\forall i$
  - $\alpha_i(y^{(i)}(w^Tx^{(i)}+b)+\xi_i-1)=0$, for $\forall i$
  - $r_i\xi_i=0$, for $\forall i$

- Dual problem (derived according to the KKT conditions)

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

- Use existing QP solvers to address the above optimization problem

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

- Optimal values for α_i $(i = 1, \cdots, m)$: α_i^*
- How to calculate the optimal values of w and b?
    - Since $w = \sum_{i=1}^m \alpha_i y^{(i)} x^{(i)}$, we have

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

    - How about b?

- 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

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

for those i's

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

$$y^{(i)}(w^T x^{(i)} + b) = 1$$

  for those i's
- Hence,

$$w^T x^{(i)} + b = y^{(i)}$$

  for $\{i : 0 < \alpha_i < C\}$
- We finally calculate b as

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

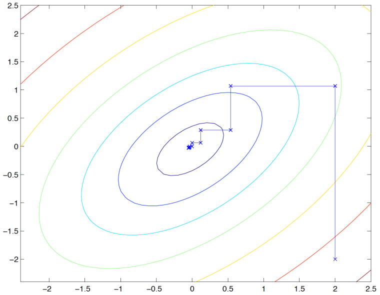
- Some useful corollaries according to the KKT conditions
  - When $\alpha_i = 0$, $y^{(i)}(w^T x^{(i)} + b) \geq 1$
  - When $\alpha_i = C$, $y^{(i)}(w^T x^{(i)} + b) \leq 1$
  - When $0 < \alpha_i < C$, $y^{(i)}(w^T x^{(i)} + b) = 1$

Coordinate Ascent Algorithm

- Consider the following unconstrained optimization problem

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

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

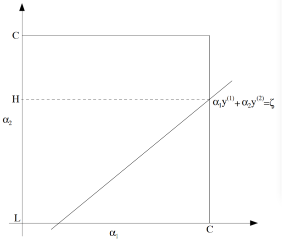


Sequential Minimal Optimization (SMO) Algorithm

- Coordinate ascent algorithm cannot be applied since $\sum_{i=0}^{m} \alpha_i y^{(i)} = 0$
- The basic idea of SMO
  Repeat the following steps until convergence
  - Select some pair of α_i and α_j to update next (using a heuristic that tries to pick the two that will allow us to make the biggest progress towards the global maximum)
  - Re-optimize $\mathcal{G}(\alpha)$ with respect to α_i and α_j, while holding all the other α_k's $(k \neq i, j)$ fixed

- Take α_1 and α_2 for example

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



- Given α_1, α_2 will be bounded by L and H such that $L \leq \alpha_2 \leq H$
  - If $y^{(1)} y^{(2)} = -1$, $H = \min(C, C + \alpha_2 - \alpha_1)$ and $L = \max(0, \alpha_2 - \alpha_1)$
  - If $y^{(1)} y^{(2)} = 1$, $H = \min(C, \alpha_1 + \alpha_2)$ and $L = \max(0, \alpha_1 + \alpha_2 - C)$

- Our objective function becomes

$$f(\alpha) = f(\alpha_1, \alpha_2) = f((\zeta - \alpha_2 y^{(2)}) y^{(1)}, \alpha_2, \alpha_3, \cdots, \alpha_m)$$

- Resolving the following optimization problem

$$\begin{aligned} \max_{\alpha_2} \quad & f((\zeta - \alpha_2 y^{(2)}) y^{(1)}, \alpha_2, \alpha_3, \cdots, \alpha_m) \\ s.t. \quad & L \leq \alpha_2 \leq H \end{aligned}$$

where $\{\alpha_3, \cdots, \alpha_m\}$ are treated as constants

- Solving
$$\frac{\partial}{\partial \alpha_2} f((\zeta - \alpha_2 y^{(2)}) y^{(1)}, \alpha_2) = 0$$
gives the updating rule for α_2 (without considering the corresponding constraints)
$$\alpha_2^+ = \alpha_2 + \frac{y^{(2)}(E_1 - E_2)}{K_{11} - 2K_{12} + K_{22}}$$
where
  - $K_{i,j} =< x_i, x_j >$
  - $f_i = \sum_{i=1}^m y^{(i)} \alpha_j K_{i,j} + b$
  - $E_i = f_i - y^{(i)}$
  - $V_i = \sum_{j=3}^m y^{(j)} \alpha_i K_{i,j} = f_i - \sum_{j=1}^2 y^{(j)} \alpha_j K_{i,j} - b$
- Considering $L \leq \alpha_2 \leq H$, we have
$$\alpha_2 = \begin{cases} H & \text{if } \alpha_2^+ > H \\ \alpha_2^+ & \text{if } L \leq \alpha_2^+ \leq H \\ L & \text{if } \alpha_2^+ < L \end{cases}$$

- How about b?
  - When $\alpha_1 \in (0, C)$, $b^+ = (\alpha_1 - \alpha_1^+)y^{(1)}K_{11} + (\alpha_2 - \alpha_2^+)K_{12} - E_1 + b = b_1$
  - When $\alpha_2 \in (0, C)$, $b^+ = (\alpha_1 - \alpha_1^+)y^{(1)}K_{12} + (\alpha_2 - \alpha_2^+)K_{22} - E_2 + b = b_2$
  - When $\alpha_1, \alpha_2 \in (0, C)$, $b^+ = b_1 = b_2$
  - When $\alpha_1, \alpha_2 \in \{0, C\}$, $b^+ = (b_1 + b_2)/2$
- More details can be found at
  - J. Platt, Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines, Microsoft Research Technical Report, 1998.
  - `https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-98-14.pdf`

Thanks!

Q & A