Let us construct $L^*(\lambda)$ for SVM:

$$L^*(\alpha_i, \lambda_i) = \min_{w, w_0} \frac{1}{2}\|w\|^2 + C\sum_i \xi_i - \sum_i \alpha_i \left(y_i(w^T\phi(x_i) + w_0) + \xi_i - 1\right) - \sum_i \lambda_i \xi_i$$

(assuming $\alpha_i \geq 0$, $\lambda_i \geq 0$, the RHS argument of min is strictly convex)

$$= \frac{1}{2}\left(\sum_i \alpha_i y_i \phi(x_i)\right)^T \left(\sum_j \alpha_j y_j \phi(x_j)\right) + \sum_i \xi_i \cancel{(C - \alpha_i - \lambda_i)} \; 0 - \sum_i \alpha_i \left(y_i \sum_j y_j \alpha_j \phi^T(x_j)\phi(x_i)\right) + \sum_i \alpha_i - \sum_i \alpha_i y_i w_0$$

(Substituting for w from KKT necessary conditions)

$$= -\frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \phi^T(x_i)\phi(x_j) + \sum_i \alpha_i - w_0 \cancel{\sum \alpha_i y_i}$$

0

$$\boxed{L^*(\alpha_i, \lambda_i) = -\frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \phi^T(x_i)\phi(x_j) + \sum_i \alpha_i}$$

The dual function

Our dual problem is

(Refer to class notes of 22/01/2013)

$$\max -\frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \phi^T(x_i)\phi(x_j) + \sum_i \alpha_i$$

$\lambda_i \geq 0$

$\alpha_i \geq 0$ & $\alpha_i \leq C$ & $\sum \alpha_i y_i = 0$

new set of constraints

The constraints ($\lambda_i + \alpha_i = C$ & $\sum \alpha_i y_i = 0$) obtained while solving to obtain the dual function will also reflect in the constraints in the dual optimisation problem

Our final dual optimisation problem:

$$\boxed{\begin{aligned} \max_{\alpha_i} \; & -\frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \phi^T(x_i)\phi(x_j) + \sum_i \alpha_i \\ \text{s.t.} \; & C \geq \alpha_i \geq 0 \qquad [\text{DSVM}] \\ & \sum_i \alpha_i y_i = 0 \end{aligned}}$$

We hope that the solution α_i

to the dual [DSVM] problem recovers solution (ω, ω_0) to the primal SVM.

And in fact by Theorem 82 from class notes of 22/1/2013, the duality gap = 0, i.e

DSVM gives us solution to the original svm problem.

Q: Why is the dual interesting

i) Mainly for this course:

The $\phi^T(x_i)\,\phi(x_j)$

Let us try to write even the SVM decision function in terms of the dot product.

$$f(x) = w^T \phi(x) + w_0$$

$$= \sum_i \alpha_i y_i \phi^T(x_i)\phi(x) + w_0$$

Can I write w_0 also in terms of the dot product?

Using the results from last class (22/1/2013):

for any $\alpha_i \in (0, c)$

$$y_i(w^T \phi(x_i) + w_0) = 1$$

$$\Rightarrow \quad w_0 = y_i - \sum_j \alpha_j y_j \phi^T(x_i)\phi(x_j)$$

The bottom line is that if I could compute $\phi^T(x_i)\phi(x_j)$ more directly than having to first compute $\phi(x)$ vector & thene compute dot products, I could capture very complex object (x_i)

Let $\phi^T(x_i)\phi(x_j) = K(x_i, x_j)$

The kernel function.

$K(x_i, x_j)$ is a kernel fn iff $\exists\ \phi : \mathcal{X} \to \mathbb{R}^m$ & $K(x_i, x_j) = \phi^T(x_i)\phi(x_j)$

In terms of the kernel function the dual svm (DSVM) problem is:

$$\boxed{\begin{aligned} \max_{[\alpha_j \dots]} \quad & -\frac{1}{2}\sum_i \sum_j y_i y_j \alpha_i \alpha_j K(x_i, x_j) + \sum_i \alpha_i \\ \text{s.t.} \quad & \sum_i \alpha_i y_i = 0 \\ & \alpha_i \in [0, C] \end{aligned}} \quad (*)$$

It turns that there are efficient solvers for DSVM

(SMO algo: Sequential Minimal Optimization)

Some guiding heuristics for picking pairs of (α_i, α_j) for solving $(*)$

(0) Start with all $\alpha_i = 0$ or random

(1) Pick in pairs (α_i, α_j) since $\sum_i \alpha_i y_i = 0$. Thus, you cannot pick only one α_i at an iteration

(2) You can choose the α_i / α_j in (1) based on violation of constraints in the KKT (see last class notes)

Eg: $\forall \alpha_i \in (0, C)$

$y_i(\omega^T \phi(x_i) + \omega_0) = 1$

or $\forall \alpha_i = 0$

$$y_i\left(w^T\phi(x_i) + w_0\right) \geq 1$$

OR $\forall\ \alpha_i = C$

$$y_i\left(w^T\phi(x_i) + w_0\right) \leq 1$$

Examples of $\phi(x)$ for which $K(x_i, x_j)$ can be computed efficiently without enumerating

$$\left[\phi_1(x)\ \ \phi_2(x)\ \ \ldots\ \ \phi_m(x)\right]$$

Eg: String kernels

Subsequence kernels

Tree kernels

Graph kernels

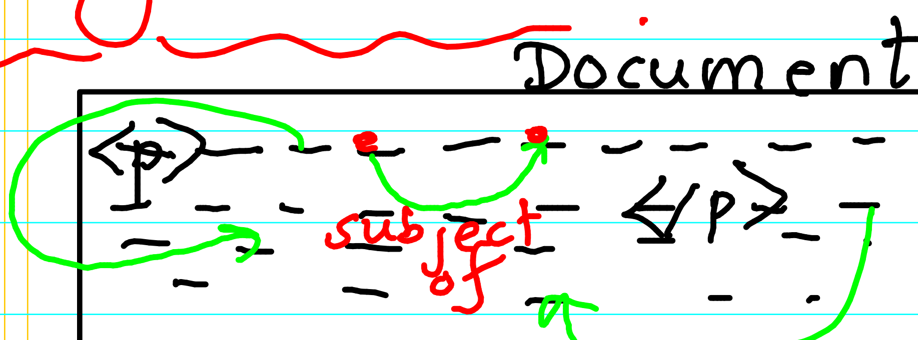
Convolution kernels

Rational kernels

First order relational kernels

Examples of kernels that enable relational learning

String kernels:



→ can be seen as a sequence of words

→ Can be seen as a DOM Tree

→ seen as a graph

Ram ate the apple
(subject) (verb)

In string kernel, we treat the object (document) as a sequence of tokens/words/characters

Doc D1

w_1 w_2 w_3

. . . . w_t

(w_1 - - w_t are words)

Doc D2

v_1 v_2 . .

- - . v_r

(v_1 . . v_r are words)

A simple model:

$$\phi(D) = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

i^{th} elements indicates if word u_i occurs in D

A more complex model:

$$\phi(D) = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

element indicates if word sequence $[u_{i_1}, u_{i_2} \dots u_{i_p}]$ occurs in D