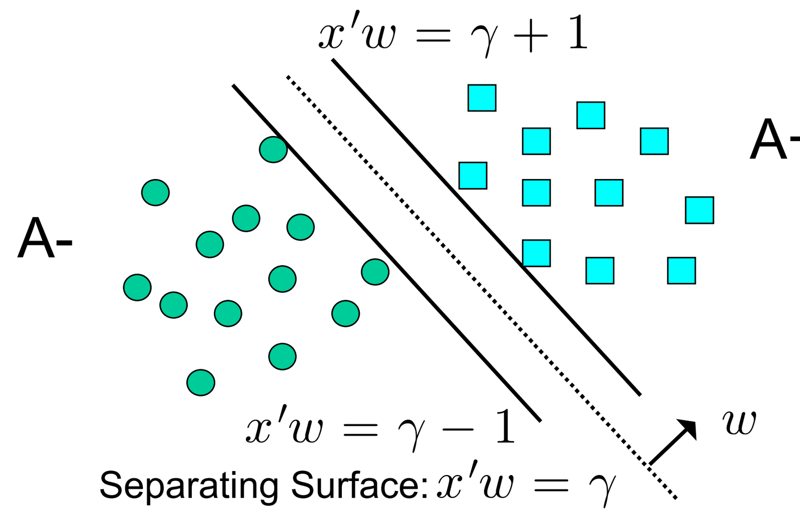
Support Vector Machines: Overview

Supervised Machine Learning Problem: Discriminate class A+ from class A-



Given m points in the n dimensional space R^n

Represented by an m x n matrix A

Membership of each point A_i in the classes +1 or -1 is specified by:

An m x m diagonal matrix D with ± 1 along its diagonal

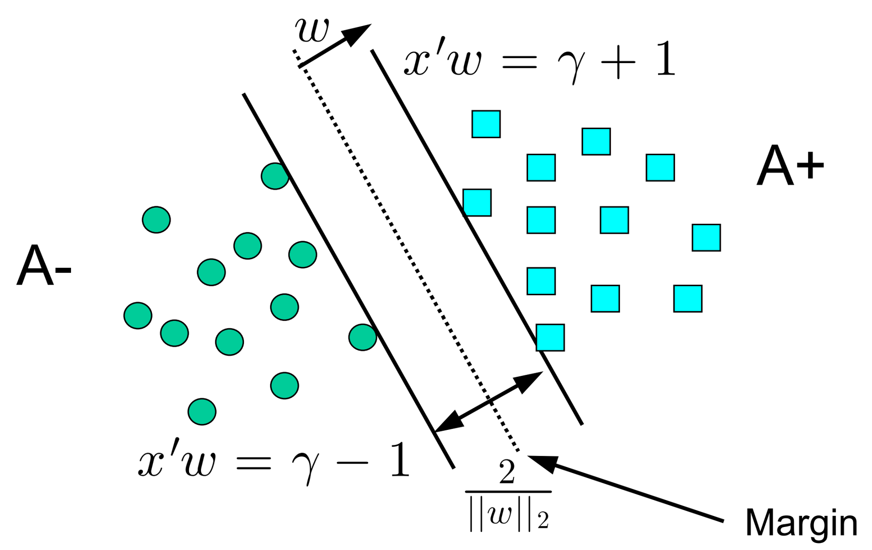
Separate by two bounding planes: $x'w = \gamma \pm 1$ such that:

$$A_i w \geq \gamma + 1, \text{ for } D_{ii} = +1$$
$$A_i w \leq \gamma - 1, \text{ for } D_{ii} = -1$$

More succinctly: $D(Aw - e\gamma) \geq e$

where e is a vector of ones.

Maximize Margin Between Separating Planes



The (Linear) Support Vector Machine Formulation

Solve the following quadratic program:

$$\min_{w,\gamma,y} \quad \nu e'y + \|w\|_2^2$$
$$\text{s.t. } D(Aw - e\gamma) + y \geq e$$
$$y \geq 0$$

where y = nonnegative error (slack) vector.

ν is a parameter which indicates how much emphasis is to be given on reducing misclassification error, in contrast with maximizing the margin. It is typically chosen via cross-validation.

Active Support Vector Machine (ASVM)

- Fast algorithm which utilizes an active set method for solving the quadratic program
- Requires no specialized solvers or software tools, apart from a freely available equation solver
- Inverts a matrix of the order of the number of features
- Guaranteed to converge in a finite number of iterations

Changes to SVM Formulation

- Allow soft margin error (y) to contribute in a *quadratic* fashion, instead of *linear*.
- Maximize the margin between the separating hyperplanes with respect to orientation (w) as well as location relative to the origin (γ).

soft margin error | margin orientation | margin location

$$\min_{w,\gamma,y} \tfrac{1}{2}\nu\, y'y + \tfrac{1}{2}(w'w + \gamma^2)$$
$$\text{s.t. } D(Aw - e\gamma) + y \geq e$$

The Wolfe Dual Problem

- The Wolfe dual is an equivalent quadratic program with nonnegativity constraints only that is easier to solve.

$$\min_{0 \leq u \in R^m} \tfrac{1}{2}u'(\tfrac{I}{\nu} + D(AA' + ee')D)u - e'u$$
$$(w = A'Du, \quad y = u/\nu, \quad \gamma = -e'Du)$$

- I = identity matrix
- Non-negativity constraints only, which leads to active set algorithm
- u = dual variable

Active Set Algorithm: Idea

- Partition dual variables into:
  - nonbasic variables: $u_i = 0$
  - basic variables: $u_i > 0$
- Algorithm is an iterative procedure.
  - Choose a working set of variables corresponding to active constraints to be nonbasic
  - Calculate the global minimum on basic variables
  - Appropriately update working set
- Goal is to find appropriate working set.
  - When found, global minimum on basic variables is solution to problem

Setting It Up

- Make substitutions to simplify formulation:

$$H = D[A \quad -e], \quad Q = \tfrac{I}{\nu} + HH'$$

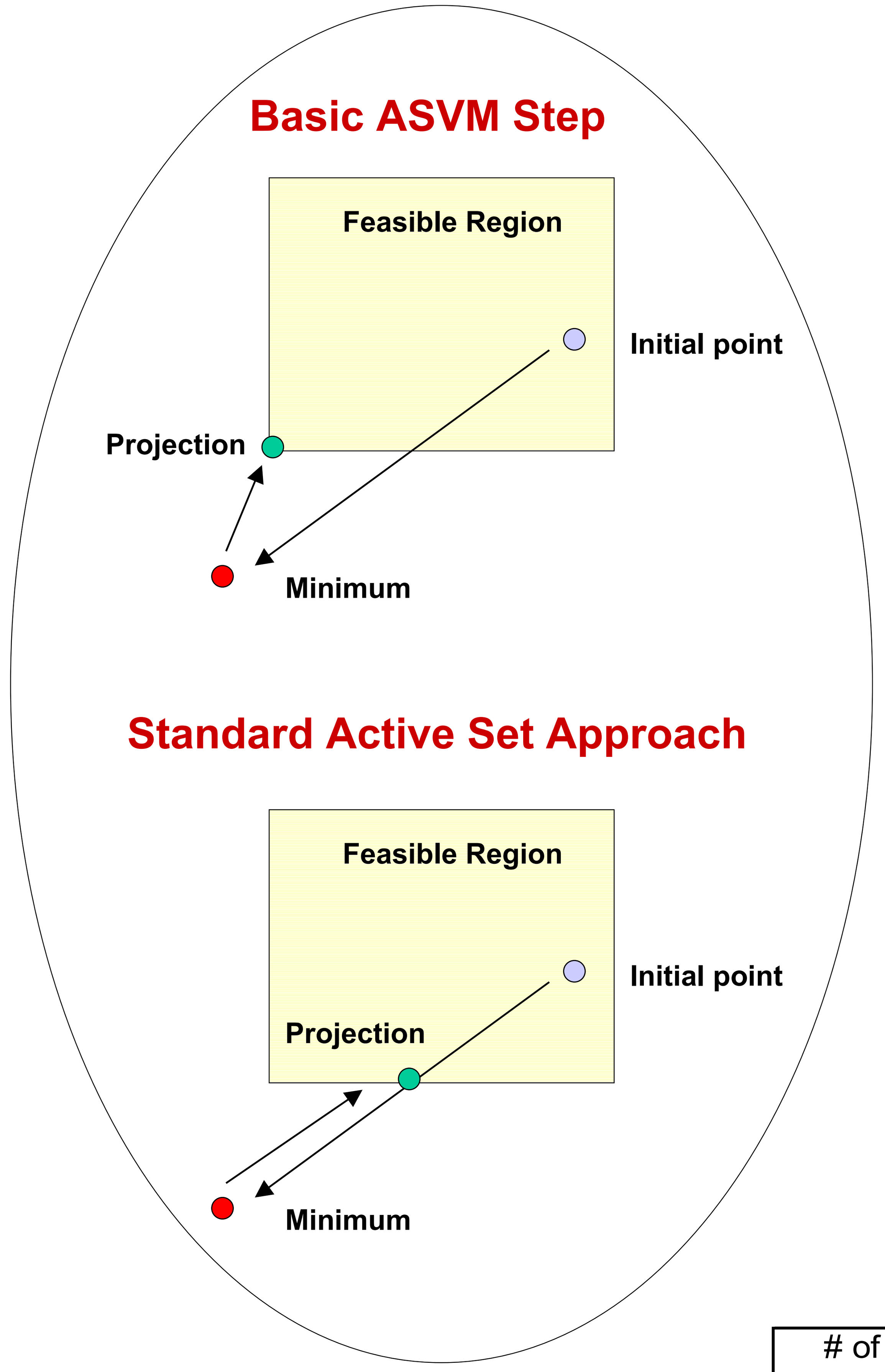
- Dual problem then becomes:

$$\min_{0 \leq u \in R^m} \tfrac{1}{2}u'Qu - e'u$$

- When computing Q^{-1}, we use
  - Sherman-Morrison-Woodbury identity:

$$Q^{-1} = (\tfrac{I}{\nu} + HH')^{-1} = \nu(I - H(\tfrac{I}{\nu} + H'H)^{-1}H')$$

  - Only need to invert a much smaller matrix of size (n + 1) x (n + 1)



Active Set Algorithm: Essentials

- Initialize u to the global minimum, then enforce nonnegativity constraint.

$$\min_{0 \leq u \in R^m} \tfrac{1}{2}u'Qu - e'u \xrightarrow{\text{Set gradient = 0}} u = (Q^{-1}e)_+$$

- While basic set is smaller than previous iteration:
  - Define basic and nonbasic sets:

$$B = \{j | u_j > 0\}, \quad N = \{j | u_j = 0\}$$

  - Find minimum on basic set:

$$u_B = (Q_{BB}^{-1} e_B)_+, \quad u_N = 0$$

  - Repeat.

- Summary:
  - The ASVM basic approach finds the minimum on a set of basic variables, then projects onto the feasible region.
  - This differs from other active set methods, which "backtrack" onto the feasible region.

Experiments

- Compared ASVM with standard formulation (SVM-QP), run under both CPLEX 6.5 and SVM[light] 3.10b
  - measured generalization accuracy and running time
- Tuning set w/ tenfold cross-validation used to find appropriate values of ν
- Massive Gaussian data generated by NDC generator
- All experiments run on Locop2
  - 400 MHz Pentium II Xeon, 2 Gigabytes available memory
  - Windows NT Server 4.0, Visual C++ 6.0

Experiments on UCI datasets: ASVM is fast!

Dataset m x n	Algorithm	Training Correctness	Testing Correctness	Time (CPU sec)
Liver Disorders	CPLEX	70.76%	68.41%	7.87
	SVM*light*	70.37%	68.12%	0.26
345 x 6	ASVM	70.40%	67.25%	0.03
Cleveland Heart	CPLEX	87.50%	84.20%	4.17
	SVM*light*	87.50%	84.20%	0.17
297 x 13	ASVM	87.24%	85.56%	0.05
Pima Diabetes	CPLEX	77.36%	76.95%	128.90
	SVM*light*	77.36%	76.95%	0.19
768 x 8	ASVM	78.04%	78.12%	0.08

Dataset m x n	Algorithm	Training Correctness	Testing Correctness	Time (CPU sec)
Ionosphere	CPLEX	92.81%	88.60%	9.84
	SVM*light*	92.81%	88.60%	0.23
351 x 34	ASVM	93.29%	87.75%	0.26
Tic Tac Toe	CPLEX	65.34%	65.34%	206.52
	SVM*light*	65.34%	65.34%	0.23
958 x 9	ASVM	70.27%	69.72%	0.05
Votes	CPLEX	96.02%	95.85%	27.26
	SVM*light*	96.02%	95.85%	0.06
435 x 16	ASVM	96.73%	96.07%	0.09

ASVM runs quickly on massive datasets

# of Points	# of features	Iterations	Training Correctness	Testing Correctness	Time (CPU min)
4 million	32	5	86.09%	86.06%	38.04
7 million	32	5	86.10%	86.28%	95.57

Notes:

- Data was in core for these experiments. The algorithm can easily be extended for larger datasets.
- Convergence is guaranteed in a *finite* number of iterations.
- Nonlinear kernels are possible, but slower – the Sherman-Morrison-Woodbury identity cannot be used.
- ASVM is available on the web for download at `http://www.cs.wisc.edu/dmi/asvm`