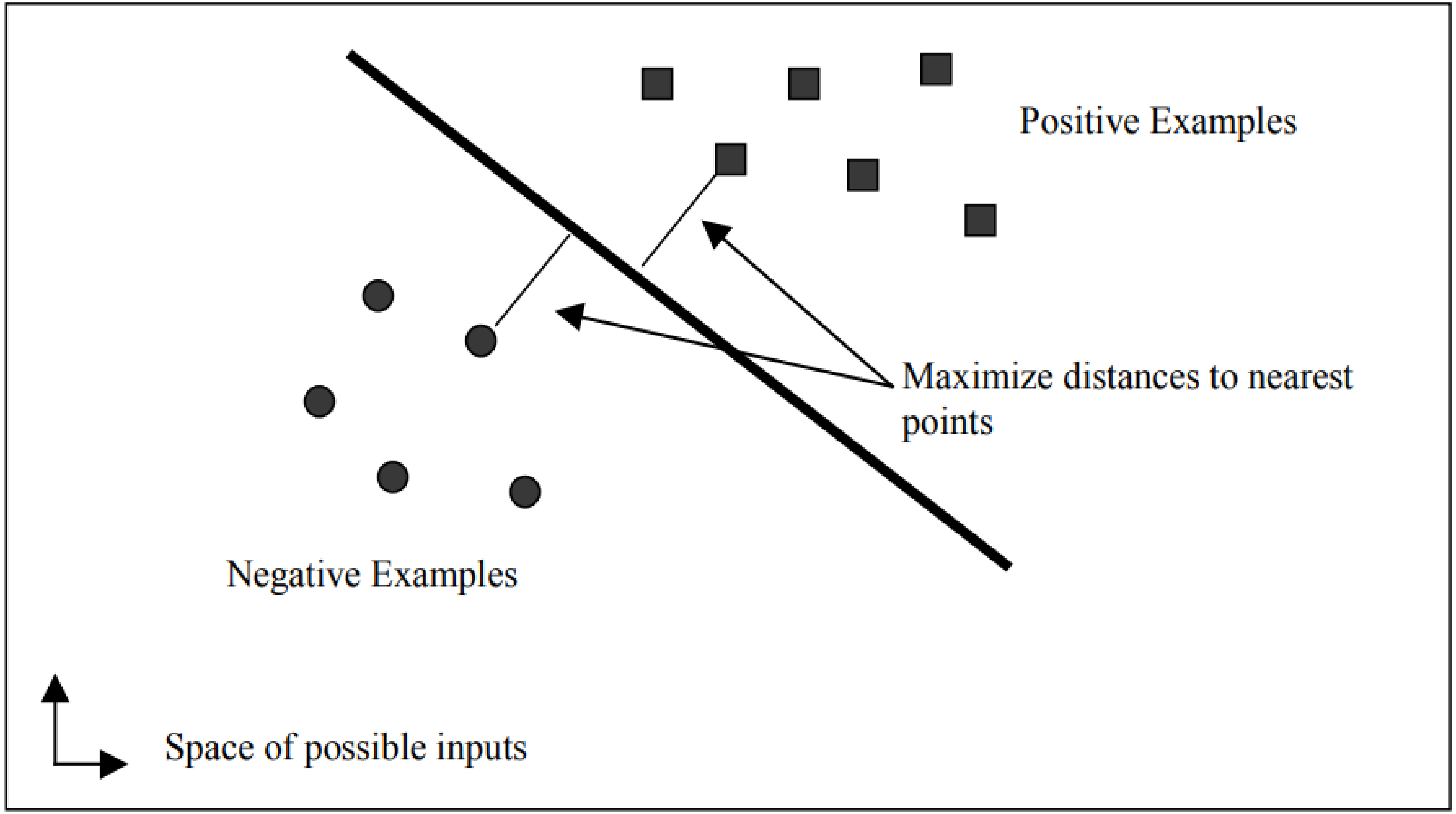
# CS709 Convex Optimization

Quadratic programming (SVM optimization)

# Quadratic Optimization

$$\begin{aligned} &\text{minimize} \quad && \tfrac{1}{2}\mathbf{x}^{\mathrm{T}} Q\mathbf{x} + \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ &\text{subject to} \quad && A\mathbf{x} \leq \mathbf{b}, \end{aligned}$$

# Support Vector Machines



# SVM primal

$$\min_{\vec{w},b} \frac{1}{2} ||\vec{w}||^2 \text{ subject to } y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1, \forall i,$$

Quadratic in w, with linear inequality constraint

# SVM Dual

$$\min_{\vec{\alpha}} \Psi(\vec{\alpha}) = \min_{\vec{\alpha}} \tfrac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j (\vec{x}_i \cdot \vec{x}_j) \alpha_i \alpha_j - \sum_{i=1}^{N} \alpha_i ,$$

$$\alpha_i \geq 0, \forall i, \qquad \sum_{i=1}^{N} y_i \alpha_i = 0.$$

# SVM (Dual) with slack

$$\min_{\vec{\alpha}} \Psi(\vec{\alpha}) = \min_{\vec{\alpha}} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j K(\vec{x}_i, \vec{x}_j) \alpha_i \alpha_j - \sum_{i=1}^{N} \alpha_i ,$$

$$0 \le \alpha_i \le C, \forall i,$$

$$\sum_{i=1}^{N} y_i \alpha_i = 0.$$

# Two approaches used

- SMO (Sequential Minimal Optimization)
- CVXOPT (Quadratic Solver)

# Sequential Minimal Optimization

Preferred because of following reasons:

- Generally faster
- Scalable
- Simple
- Easy to implement
- No storage for Kernel matrix

# SMO (Contd.)

We chose two Lagrangian multipliers (alpha1 and alpha2) and analytically solved to find the following update rule for alpha2 and alpha1.

$$\alpha_2^{\text{new}} = \alpha_2 + \frac{y_2(E_1 - E_2)}{\eta},$$

$$\alpha_2^{\text{new,clipped}} = \begin{cases} H & \text{if} \quad \alpha_2^{\text{new}} \geq H; \\ \alpha_2^{\text{new}} & \text{if} \quad L < \alpha_2^{\text{new}} < H; \\ \text{L} & \text{if} \quad \alpha_2^{\text{new}} \leq L. \end{cases}$$

$$\alpha_1^{\text{new}} = \alpha_1 + s(\alpha_2 - \alpha_2^{\text{new,clipped}}).$$

# Selecting Lagrangian Parameters

- Iterate over entire training set.
- If training example does not fulfill KKT conditions within tolerance, we take its Lagrangian as one of the Lagrangian parameters.
- Second Lagrangian parameter is chosen from following in order:
    - Repeated pass over all non-bound examples
    - If we can’t any alpha_j from above then we iterate over all training set.

# Finding w and b

After finding alpha_i we calculate w and b using equations

$$\vec{w} = \sum_{i=1}^{N} y_i \alpha_i \vec{x}_i , \quad b = \vec{w} \cdot \vec{x}_k - y_k \text{ for some } \alpha_\mathrm{k} > 0.$$

Which in turn is used to calculate predicted output u

$$u = \sum_{j=1}^{N} y_j \alpha_j K(\vec{x}_j , \vec{x}) - b,$$

# Time to Train (SMO vs. CVXOPT)

| No. of samples | Time units (CVXOPT) | Time units (SMO) |
|---|---|---|
| 1000 | 3.5486 | 9.1620 |
| 2000 | 17.1108 | 15.8780 |
| 3000 | 43.4945 | 40.6005 |

# New Implementation

- Based on choosing the alphas such that the error difference of corresponding alphas is maximum.
- Comparing the performance with RasseLegin implementation.
  - A pre-existing implementation

# Performance

| | RasseLegin | New implementation |
|---|---|---|
| Time Units | 0.3835519999999999 | 0.13043199999999988 |

Sample size: 3000
Accuracy: 100%

New implementation took lesser time than the pre-existing implementation