

# Digression to Lagrangian Duality



- The Primal Problem

Primal:

$$\begin{aligned} \min_w \quad & f(w) \\ \text{s.t.} \quad & g_i(w) \leq 0, \quad i = 1, \dots, k \\ & h_i(w) = 0, \quad i = 1, \dots, l \end{aligned}$$

The generalized Lagrangian:

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

the  $\alpha$ 's ( $\alpha_i \geq 0$ ) and  $\beta$ 's are called the Lagrangian multipliers

Lemma:

$$\max_{\alpha, \beta, \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta) = \begin{cases} f(w) & \text{if } w \text{ satisfies primal constraint s} \\ \infty & \text{o/w} \end{cases}$$

A re-written Primal:

$$\min_w \max_{\alpha, \beta, \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta)$$

# Lagrangian Duality, cont.

---



- Recall the Primal Problem:

$$\min_w \max_{\alpha, \beta, \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta)$$

- The Dual Problem:

$$\max_{\alpha, \beta, \alpha_i \geq 0} \min_w \mathcal{L}(w, \alpha, \beta)$$

- Theorem (weak duality):**

$$d^* = \max_{\alpha, \beta, \alpha_i \geq 0} \min_w \mathcal{L}(w, \alpha, \beta) \leq \min_w \max_{\alpha, \beta, \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta) = p^*$$

- Theorem (strong duality):**

Iff there exist a saddle point of  $\mathcal{L}(w, \alpha, \beta)$ , we have

$$d^* = p^*$$

# The KKT conditions



- If there exists some saddle point of  $\mathcal{L}$ , then the saddle point satisfies the following "Karush-Kuhn-Tucker" (KKT) conditions:

$$\frac{\partial}{\partial w_i} \mathcal{L}(w, \alpha, \beta) = 0, \quad i = 1, \dots, k$$

$$\frac{\partial}{\partial \beta_i} \mathcal{L}(w, \alpha, \beta) = 0, \quad i = 1, \dots, l$$

$$\alpha_i g_i(w) = 0, \quad i = 1, \dots, m$$

$$g_i(w) \leq 0, \quad i = 1, \dots, m$$

$$\alpha_i \geq 0, \quad i = 1, \dots, m$$

- **Theorem:** If  $w^*$ ,  $\alpha^*$  and  $\beta^*$  satisfy the KKT condition, then it is also a solution to the primal and the dual problems.

# Solving optimal margin classifier



- Recall our opt problem:

$$\begin{array}{ll}\max_{w,b} & \frac{1}{\|w\|} \\ \text{s.t} & y_i(w^T x_i + b) \geq 1, \quad \forall i\end{array}$$

- This is equivalent to

$$\begin{array}{ll}\min_{w,b} & \frac{1}{2} w^T w \\ \text{s.t} & 1 - y_i(w^T x_i + b) \leq 0, \quad \forall i\end{array} \quad (*)$$

- Write the Lagrangian:

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^m \alpha_i [y_i(w^T x_i + b) - 1]$$

- Recall that (\*) can be reformulated as  $\min_{w,b} \max_{\alpha_i \geq 0} \mathcal{L}(w, b, \alpha)$

Now we solve its **dual problem**:  $\max_{\alpha_i \geq 0} \min_{w,b} \mathcal{L}(w, b, \alpha)$

# The Dual Problem

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^m \alpha_i [y_i (w^T x_i + b) - 1]$$



$$\max_{\alpha_i \geq 0} \min_{w, b} \mathcal{L}(w, b, \alpha)$$

- We minimize  $\mathcal{L}$  with respect to  $w$  and  $b$  first:

$$\nabla_w \mathcal{L}(w, b, \alpha) = w - \sum_{i=1}^m \alpha_i y_i x_i = 0, \quad (*)$$

$$\nabla_b \mathcal{L}(w, b, \alpha) = \sum_{i=1}^m \alpha_i y_i = 0, \quad (**)$$

Note that (\*) implies:

$$w = \sum_{i=1}^m \alpha_i y_i x_i \quad (***)$$

- Plug (\*\*\*) back to  $\mathcal{L}$ , and using (\*\*), we have:

$$\mathcal{L}(w, b, \alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$



# The Dual problem, cont.

- Now we have the following dual opt problem:

$$\max_{\alpha} \mathcal{J}(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

$$\text{s.t. } \alpha_i \geq 0, \quad i = 1, \dots, m$$

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

- This is, (again,) a **quadratic programming** problem.

- A global maximum of  $\alpha_i$  can always be found.
- But what's the big deal??
- Note two things:

1.  $w$  can be recovered by  $w = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i$  See next ...

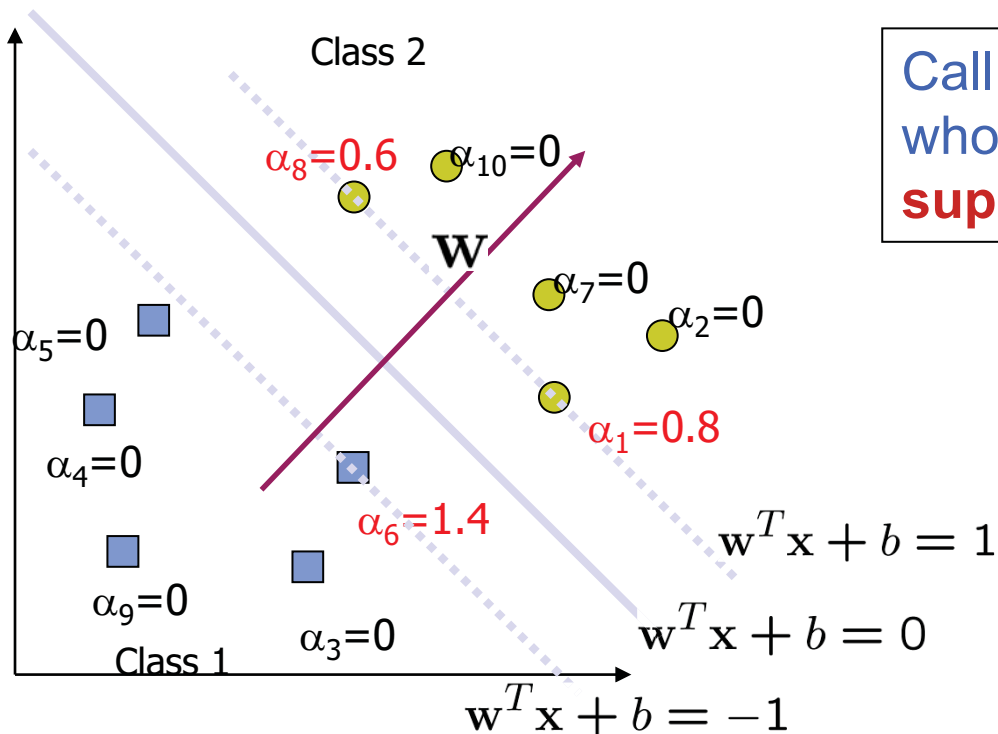
2. The "kernel"  $\mathbf{x}_i^T \mathbf{x}_j$  More later ...

# I. Support vectors



- Note the KKT condition --- only a few  $\alpha_i$ 's can be nonzero!!

$$\alpha_i g_i(w) = 0, \quad i = 1, \dots, m$$



Call the training data points whose  $\alpha_i$ 's are nonzero the **support vectors** (SV)

# Support vector machines



- Once we have the Lagrange multipliers  $\{\alpha_i\}$ , we can reconstruct the parameter vector  $w$  as a weighted combination of the training examples:

$$w = \sum_{i \in SV} \alpha_i y_i \mathbf{x}_i$$

- For testing with a new data  $z$

- Compute

$$w^T z + b = \sum_{i \in SV} \alpha_i y_i (\mathbf{x}_i^T z) + b$$

and classify  $z$  as class 1 if the sum is positive, and class 2 otherwise

- Note:  $w$  need not be formed explicitly



# Interpretation of support vector machines



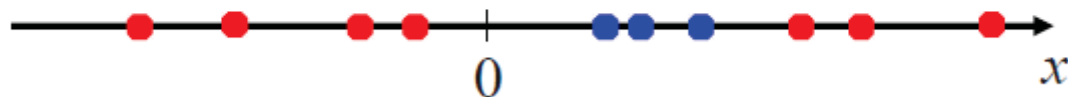
- The optimal  $\mathbf{w}$  is a linear combination of a small number of data points. This “sparse” representation can be viewed as data compression as in the construction of kNN classifier
- To compute the weights  $\{\alpha_i\}$ , and to use support vector machines we need to specify only the inner products (or kernel) between the examples  $\mathbf{x}_i^T \mathbf{x}_j$
- We make decisions by comparing each new example  $\mathbf{z}$  with only the support vectors:

$$y^* = \text{sign} \left( \sum_{i \in SV} \alpha_i y_i (\mathbf{x}_i^T \mathbf{z}) + b \right)$$

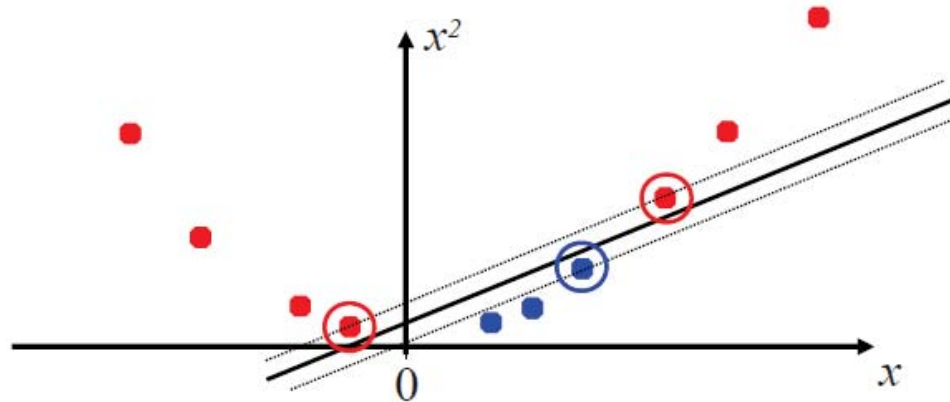


## II. The Kernel Trick

- Is this data linearly-separable?



- How about a quadratic mapping  $\phi(x_i)$ ?





## II. The Kernel Trick

---

- Recall the SVM optimization problem

$$\max_{\alpha} \quad \mathcal{J}(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

$$\text{s.t.} \quad 0 \leq \alpha_i \leq C, \quad i = 1, \dots, m$$

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

- The data points only appear as **inner product**
- As long as we can calculate the inner product in the feature space, we do not need the mapping explicitly
- Many common geometric operations (angles, distances) can be expressed by inner products
- Define the kernel function  $K$  by  $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$



## II. The Kernel Trick

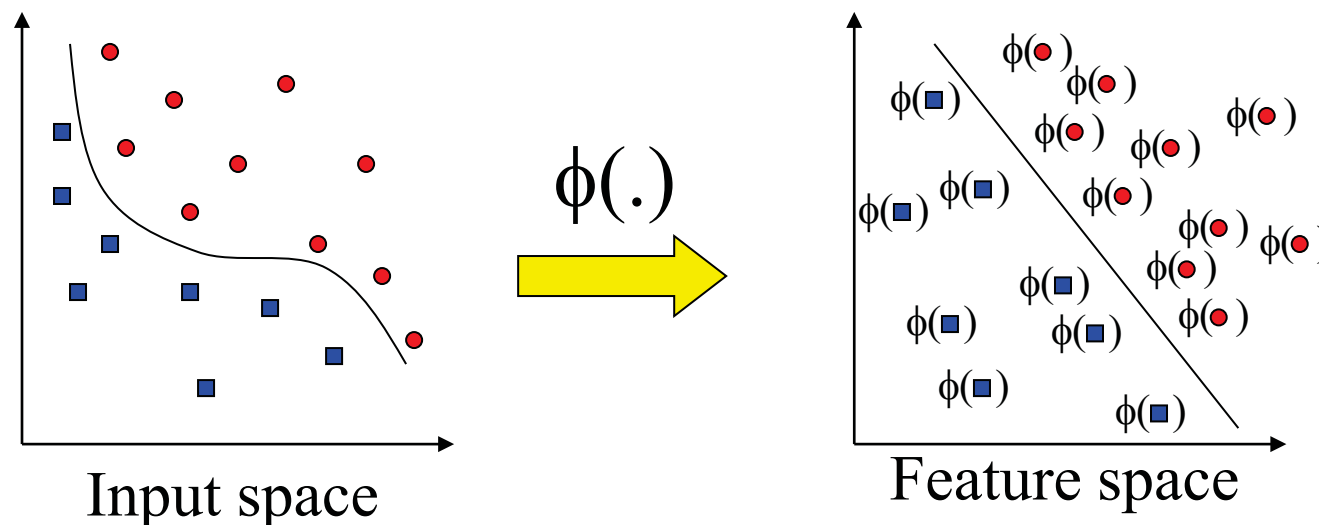
- Computation depends on feature space
  - Bad if its dimension is much larger than input space

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{s.t.} \quad & \alpha_i \geq 0, \quad i = 1, \dots, m \\ & \sum_{i=1}^m \alpha_i y_i = 0. \end{aligned}$$

Where  $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^t \phi(\mathbf{x}_j)$

$$y^*(z) = \text{sign} \left( \sum_{i \in SV} \alpha_i y_i K(\mathbf{x}_i, z) + b \right)$$

# Transforming the Data



Note: feature space is of higher dimension than the input space in practice

- Computation in the feature space can be costly because it is high dimensional
  - The feature space is typically infinite-dimensional!
- The kernel trick comes to rescue

# An Example for feature mapping and kernels



- Consider an input  $\mathbf{x}=[x_1, x_2]$
- Suppose  $\phi(\cdot)$  is given as follows

$$\phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = 1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2$$

- An inner product in the feature space is

$$\left\langle \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right), \phi\left(\begin{bmatrix} x_1' \\ x_2' \end{bmatrix}\right) \right\rangle =$$

- So, if we define the **kernel function** as follows, there is no need to carry out  $\phi(\cdot)$  explicitly

$$K(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^T \mathbf{x}')^2$$

# More examples of kernel functions

---



- Linear kernel (we've seen it)

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$$

- Polynomial kernel (we just saw an example)

$$K(\mathbf{x}, \mathbf{x}') = \left(1 + \mathbf{x}^T \mathbf{x}'\right)^p$$

where  $p = 2, 3, \dots$ . To get the feature vectors we concatenate all  $p$ th order polynomial terms of the components of  $\mathbf{x}$  (weighted appropriately)

- Radial basis kernel

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\|\mathbf{x} - \mathbf{x}'\|^2\right)$$

In this case the feature space consists of functions and results in a non-parametric classifier.



# The essence of kernel

---

- Feature mapping, but “without paying a cost”

- E.g., polynomial kernel

$$K(x, z) = (x^T z + c)^d$$

- How many dimensions we’ve got in the new space?
- How many operations it takes to compute  $K()$ ?

- Kernel design, any principle?

- $K(x, z)$  can be thought of as a similarity function between  $x$  and  $z$
- This intuition can be well reflected in the following “Gaussian” function  
(Similarly one can easily come up with other  $K()$  in the same spirit)

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

- Is this necessarily lead to a “legal” kernel?

(in the above particular case,  $K()$  is a legal one, do you know how many dimension  $\phi(x)$  is?)

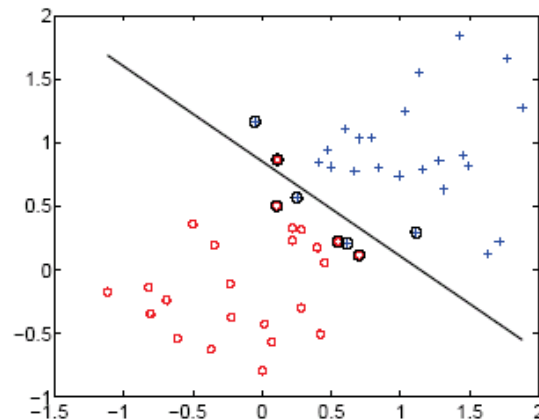


# Kernel matrix

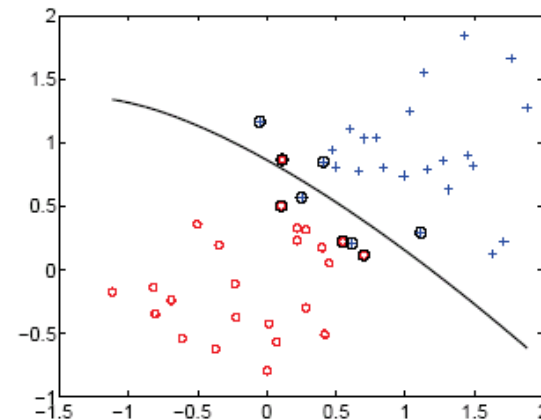


- Suppose for now that  $K$  is indeed a valid kernel corresponding to some feature mapping  $\phi$ , then for  $x_1, \dots, x_m$ , we can compute an  $m \times m$  matrix  $K = \{K_{i,j}\}$ , where  $K_{i,j} = \phi(x_i)^T \phi(x_j)$
- This is called a **kernel matrix**!
- Now, if a kernel function is indeed a valid kernel, and its elements are dot-product in the transformed feature space, it must satisfy:
  - Symmetry  $K = K^T$   
proof  $K_{i,j} = \phi(x_i)^T \phi(x_j) = \phi(x_j)^T \phi(x_i) = K_{j,i}$
  - Positive –semidefinite  $y^T K y \geq 0 \quad \forall y$   
proof?
  - Mercer's theorem

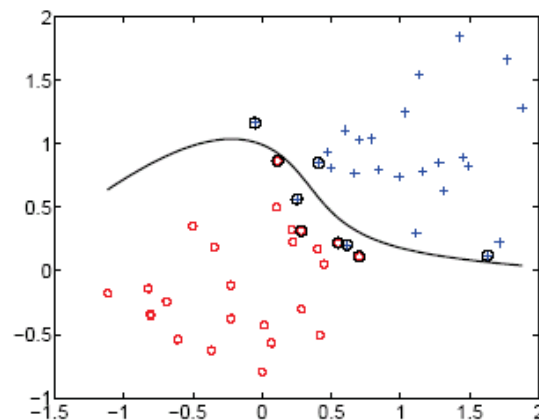
# SVM examples



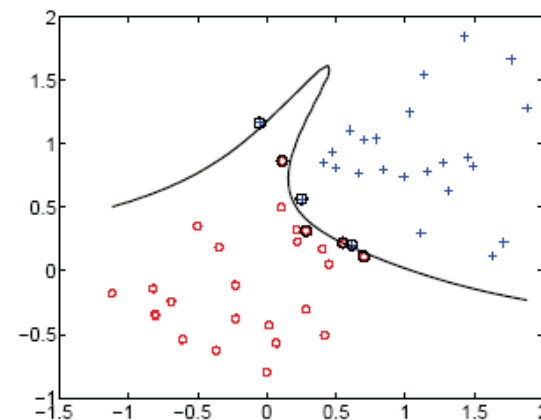
linear



$2^{nd}$  order polynomial

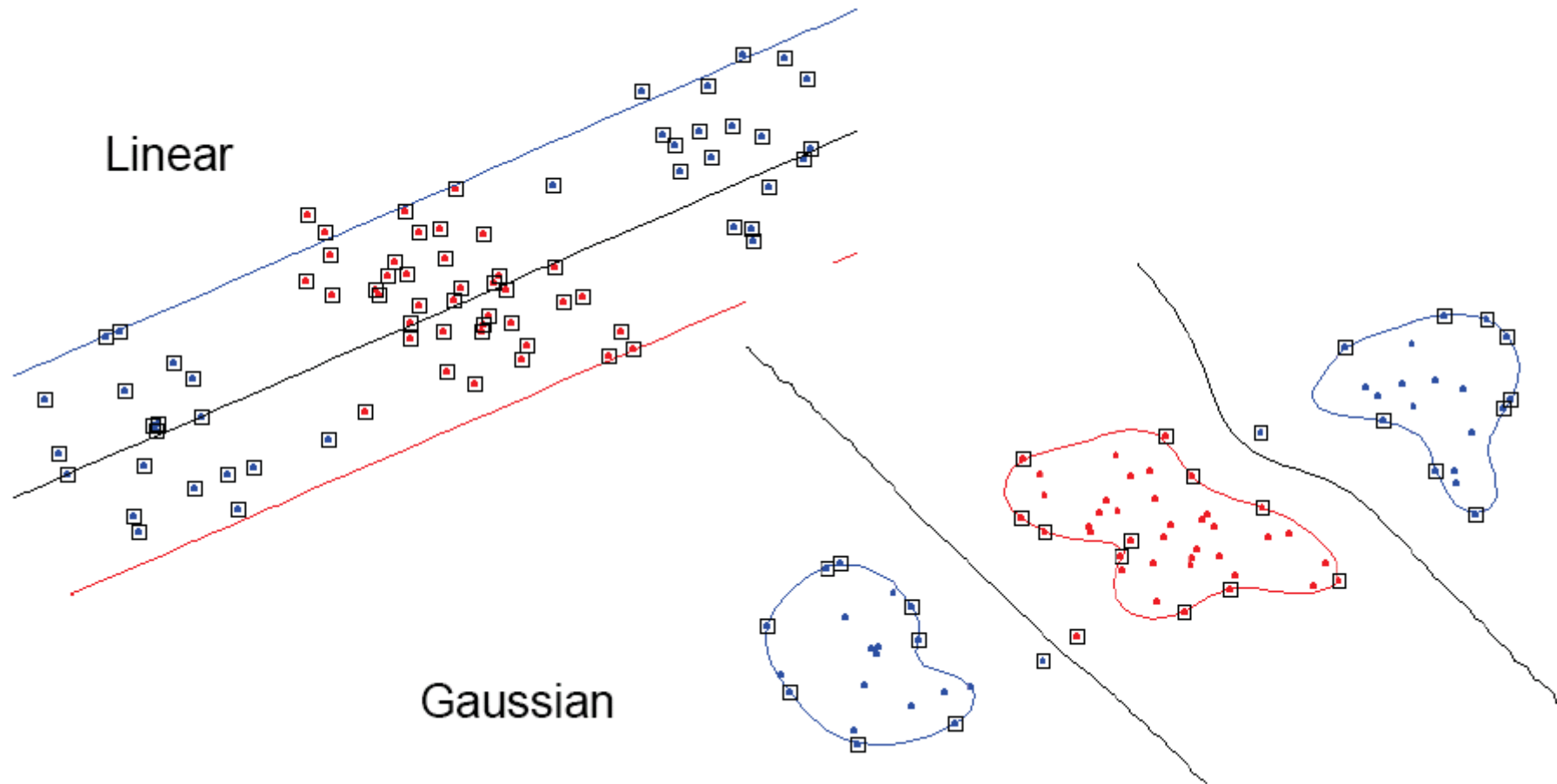


$4^{th}$  order polynomial



$8^{th}$  order polynomial

# Examples for Non Linear SVMs – Gaussian Kernel



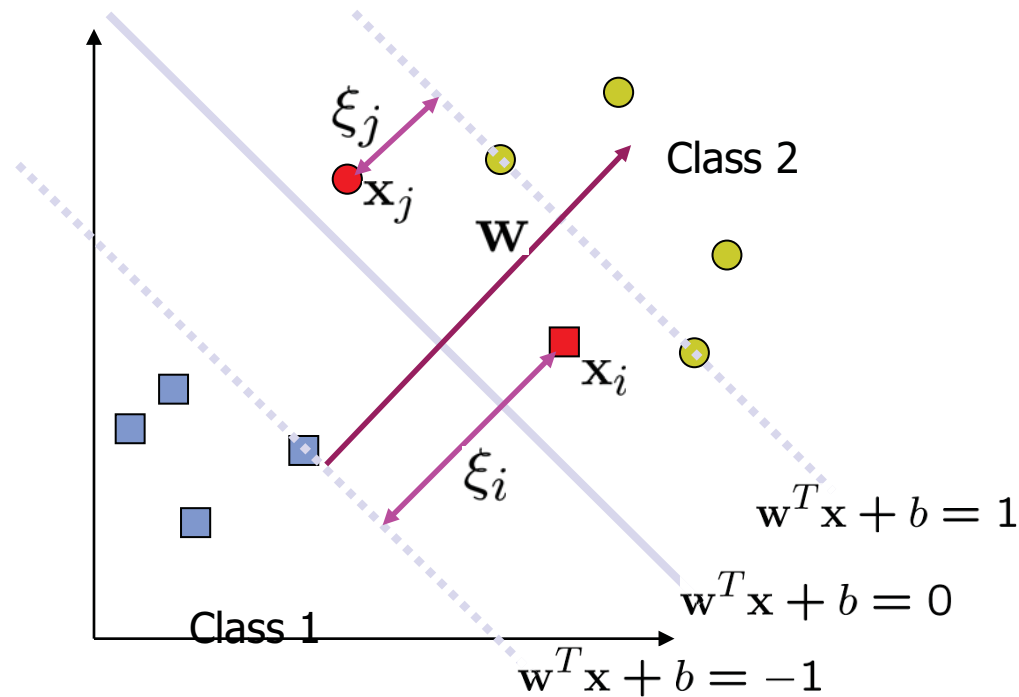
# Example Kernel

---



- $x_i$  is a bag of words
- Define  $\phi(x_i)$  as a count of every  $n$ -gram up to  $n=k$  in  $x_i$ .
  - This is huge space  $26^k$
  - What are we measuring by  $\phi(x_i)^t \phi(x_j)$ ?
- Can we compute the same quantity on input space?
  - Efficient linear dynamic program!
- Kernel is a measure of similarity
- Must be positive semi-definite

# Non-linearly Separable Problems



- We allow “error”  $\xi_i$  in classification; it is based on the output of the discriminant function  $w^T x + b$
- $\xi_i$  approximates the number of misclassified samples

# Soft Margin Hyperplane

---



- Now we have a slightly different opt problem:

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

- $\xi_i$  are “slack variables” in optimization
- Note that  $\xi_i=0$  if there is no error for  $\mathbf{x}_i$
- $\xi_i$  is an upper bound of the number of errors
- $C$  : tradeoff parameter between error and margin

# Hinge Loss

---



$$\operatorname{argmin}_{\{w,b\}} \underbrace{w^t w}_{\text{regularization}} + \underbrace{\lambda \sum_1^m \max(1 - y_i(w^t x_i + b), 0)}_{\text{Loss: hinge loss}}$$

regularization

Loss: hinge loss



# The Optimization Problem

---

- The dual of this new constrained optimization problem is

$$\begin{aligned} \max_{\alpha} \quad & \mathcal{J}(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j) \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, \quad i = 1, \dots, m \\ & \sum_{i=1}^m \alpha_i y_i = 0. \end{aligned}$$

- This is very similar to the optimization problem in the linear separable case, except that there is an upper bound  $C$  on  $\alpha_i$  now
- Once again, a QP solver can be used to find  $\alpha_i$



# The SMO algorithm

---

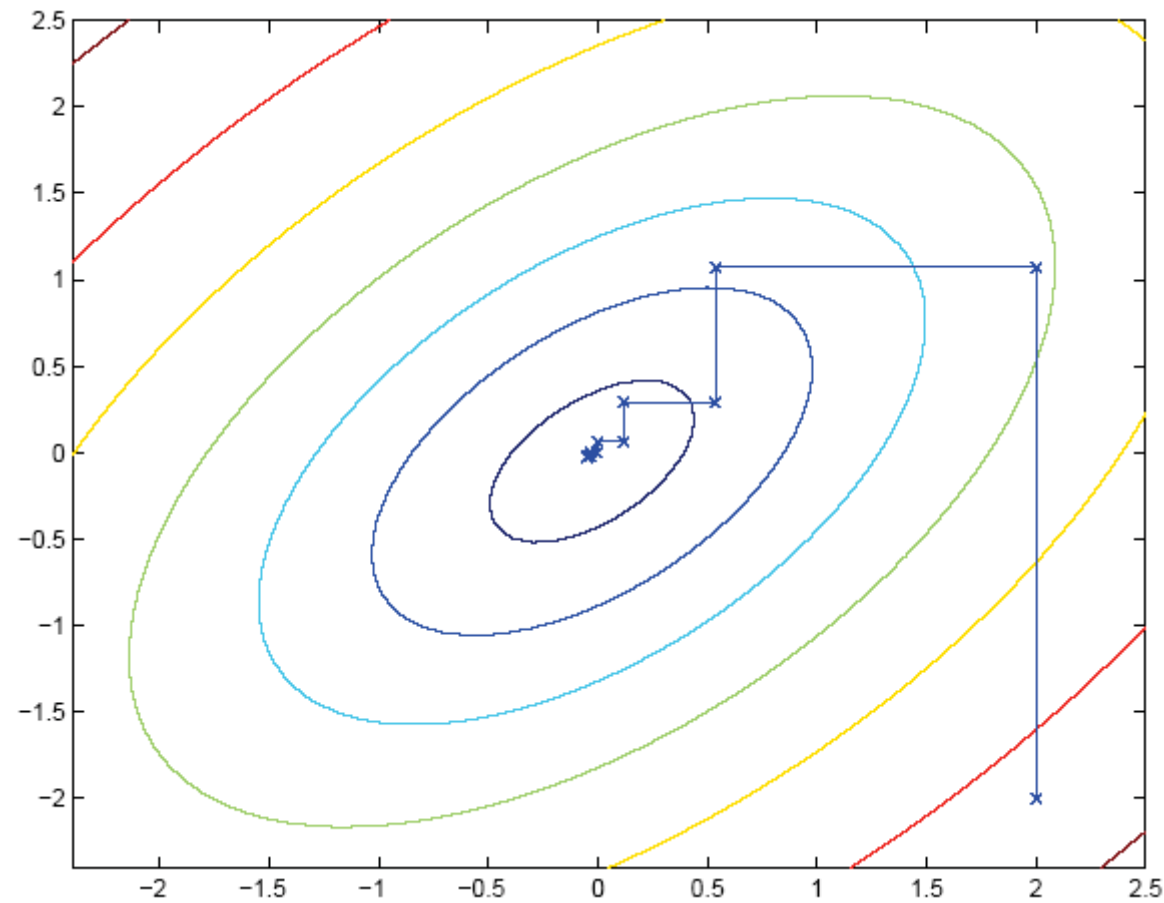


- Consider solving the **unconstrained** opt problem:

$$\max_{\alpha} W(\alpha_1, \alpha_2, \dots, \alpha_m)$$

- We've already seen several opt algorithms!
  - ?
  - ?
  - ?
- Coordinate ascend:

# Coordinate ascend



# Sequential minimal optimization

---



- Constrained optimization:

$$\max_{\alpha} \quad \mathcal{J}(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

$$\text{s.t.} \quad 0 \leq \alpha_i \leq C, \quad i = 1, \dots, m$$

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

- Question: can we do coordinate along one direction at a time (i.e., hold all  $\alpha_{[-i]}$  fixed, and update  $\alpha_i$ ?)

# The SMO algorithm

---



Repeat till convergence

1. Select some pair  $\alpha_i$  and  $\alpha_j$  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).
2. Re-optimize  $J(\alpha)$  with respect to  $\alpha_i$  and  $\alpha_j$ , while holding all the other  $\alpha_k$ 's ( $k \neq i; j$ ) fixed.

Will this procedure converge?

# Convergence of SMO



$$\max_{\alpha} \mathcal{J}(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

**KKT:**

$$\text{s.t. } 0 \leq \alpha_i \leq C, \quad i = 1, \dots, k$$
$$\sum_{i=1}^m \alpha_i y_i = 0.$$

- Let's hold  $\alpha_3, \dots, \alpha_m$  fixed and reopt J w.r.t.  $\alpha_1$  and  $\alpha_2$

# Convergence of SMO



- The constraints:

$$\alpha_1 y_1 + \alpha_2 y_2 = \xi$$

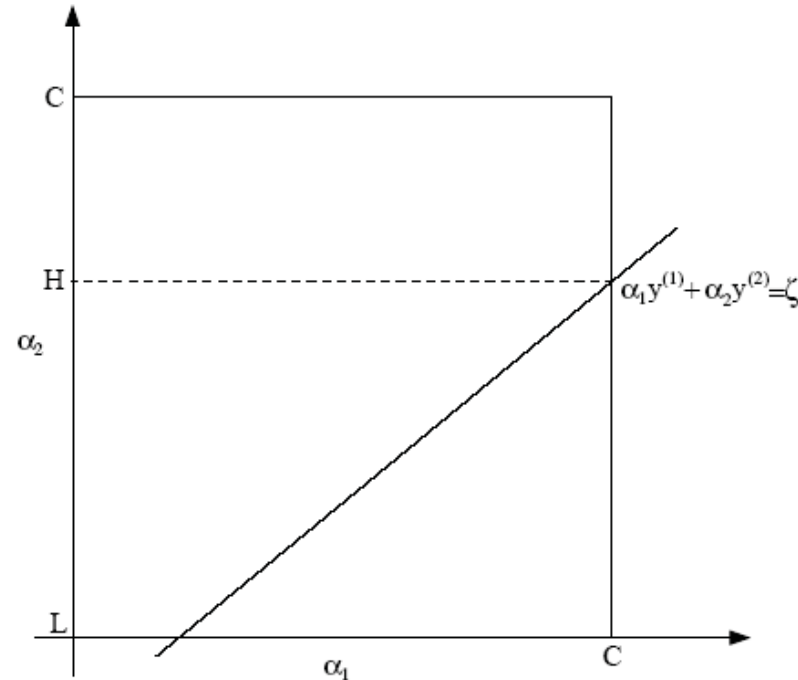
$$0 \leq \alpha_1 \leq C$$

$$0 \leq \alpha_2 \leq C$$

- The objective:

$$\mathcal{J}(\alpha_1, \alpha_2, \dots, \alpha_m) = \mathcal{J}((\xi - \alpha_2 y_2) y_1, \alpha_2, \dots, \alpha_m)$$

- Constrained opt:



# Summary

---



- Max-margin decision boundary
- Constrained convex optimization
  - Duality
  - The KKT conditions and the support vectors
  - Non-separable case and slack variables
  - The SMO algorithm