IUST
Iran University of
Science and Technology

# SVM & Kernel


Iran University of Science and Technology

By: M. S. Tahaei Ph.D.


Fall 2024

Courtesy: slides are adopted partly from Dr. Soleymani, Sharif University

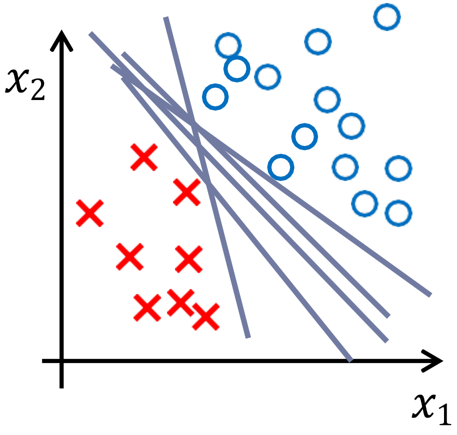
# Outline

- Margin concept
- Hard-Margin SVM
- Soft-Margin SVM
- Dual Problems of Hard-Margin SVM and Soft-Margin SVM
- Nonlinear SVM
  - Kernel trick
- Kernel methods

# Margin

- Which line is better to select as the boundary to provide more generalization capability?

Larger margin provides better generalization to unseen data

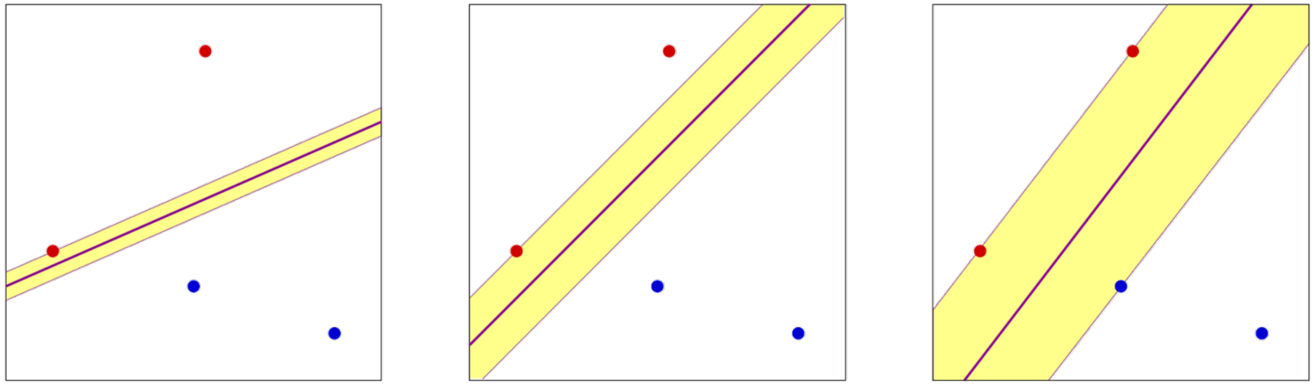



- **Margin** for a hyperplane that separates samples of two linearly separable classes is:
  - The smallest distance between the decision boundary and any of the training samples

# What is better linear separation

- Linearly separable data

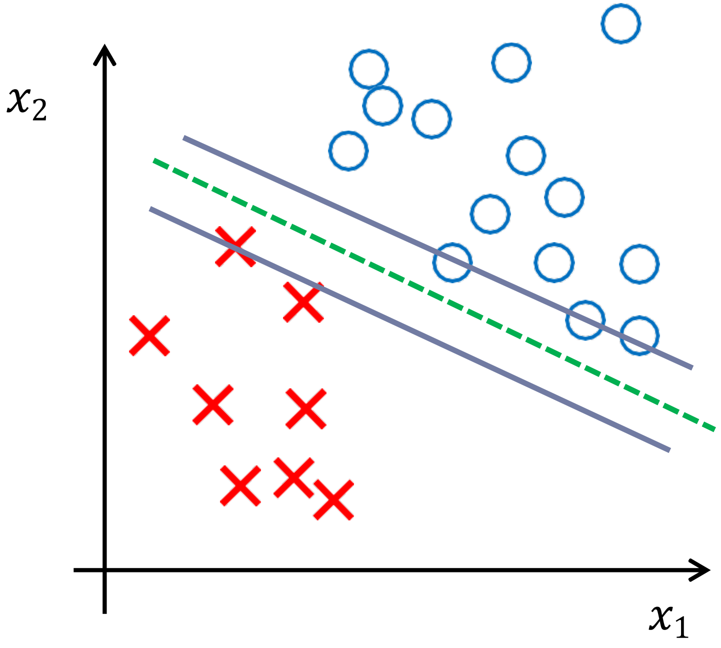
- Which line is better?

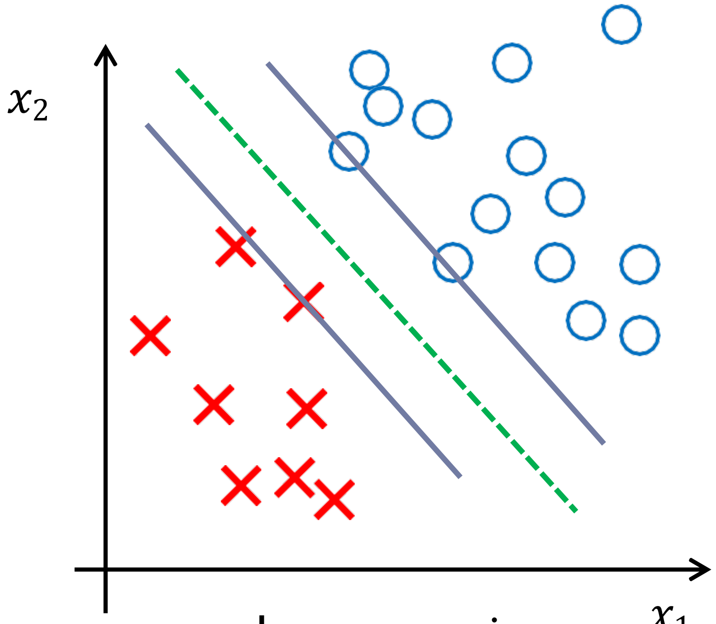


- Why the bigger margin?

# Maximum margin

- SVM finds the solution with maximum margin
  - Solution: a hyperplane that is farthest from all training samples




- The hyperplane with the largest margin has equal distances to the nearest sample of both classes

# Finding $\boldsymbol{w}$ with large margin

- Two preliminaries:
  - Pull out $w_0$
  - $\boldsymbol{w}$ is $[w_1, \dots, w_d]$

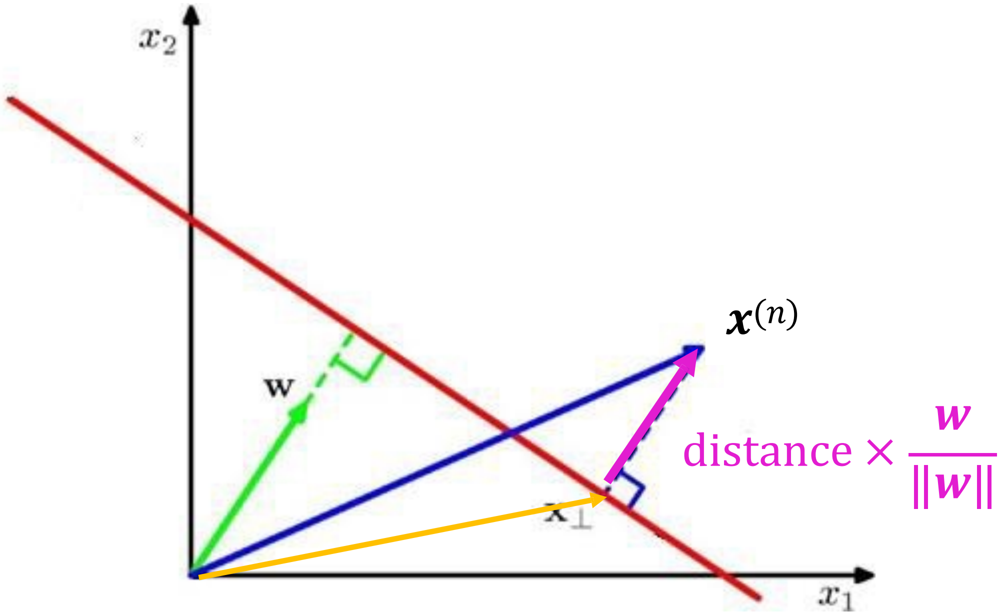
$$\boldsymbol{w}^T\boldsymbol{x} + w_0 = 0$$

We have no $x_0$

- Normalize $\boldsymbol{w}, w_0$
  - Let $\boldsymbol{x}^{(n)}$ be the nearest point to the plane
  - $\left|\boldsymbol{w}^T\boldsymbol{x}^{(n)} + w_0\right| = 1$

# Distance between an $\boldsymbol{x}^{(n)}$ and the plane

$$\text{distance} = \frac{\left|\boldsymbol{w}^T \boldsymbol{x}^{(n)} + w_0\right|}{||\boldsymbol{w}||}$$



# The optimization problem

$$\max_{\boldsymbol{w}, w_0} \frac{2}{||\boldsymbol{w}||}$$

$$\text{s.t.} \quad \min_{n=1,\dots,N} \left| \boldsymbol{w}^T \boldsymbol{x}^{(n)} + w_0 \right| = 1$$

From all the hyperplanes that correctly classify data

Notice: $\left| \boldsymbol{w}^T \boldsymbol{x}^{(n)} + w_0 \right| = y^{(n)} \left( \boldsymbol{w}^T \boldsymbol{x}^{(n)} + w_0 \right)$

$$\min_{\boldsymbol{w}, w_0} \frac{1}{2} ||\boldsymbol{w}||^2$$

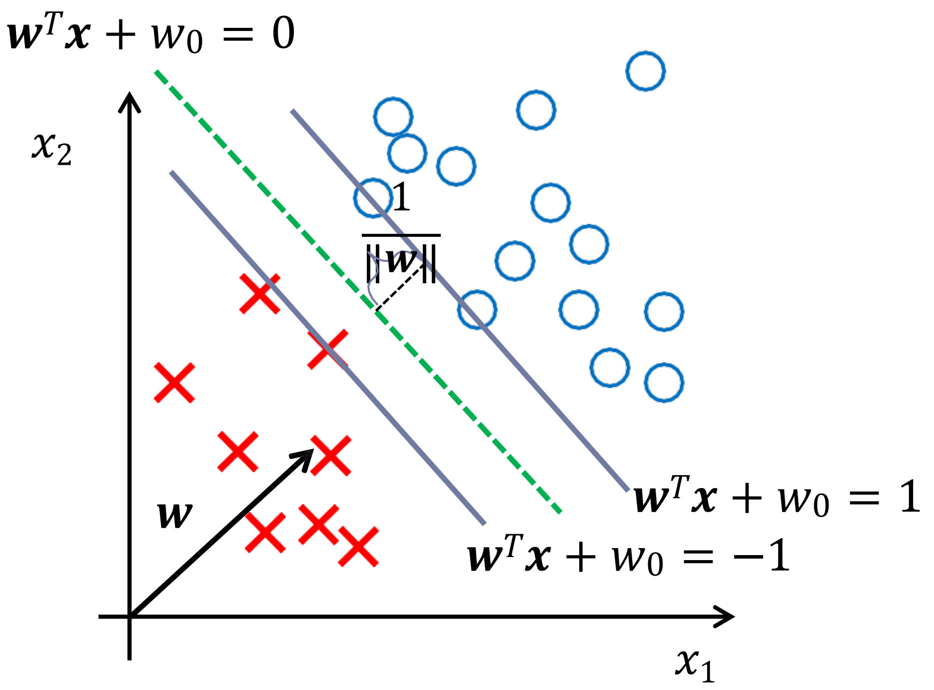
$$\text{s.t.} \quad y^{(n)} \left( \boldsymbol{w}^T \boldsymbol{x}^{(n)} + w_0 \right) \geq 1 \quad n = 1, \dots, N$$

# Hard-margin SVM: Optimization problem

$$\max_{\boldsymbol{w},w_0} \frac{2}{\|\boldsymbol{w}\|}$$

$$\text{s.t.} \left|\boldsymbol{w}^T\boldsymbol{x}^{(n)} + w_0\right| \geq 1 \;, n = 1, \ldots, N$$

Margin: $\frac{2}{\|\boldsymbol{w}\|}$



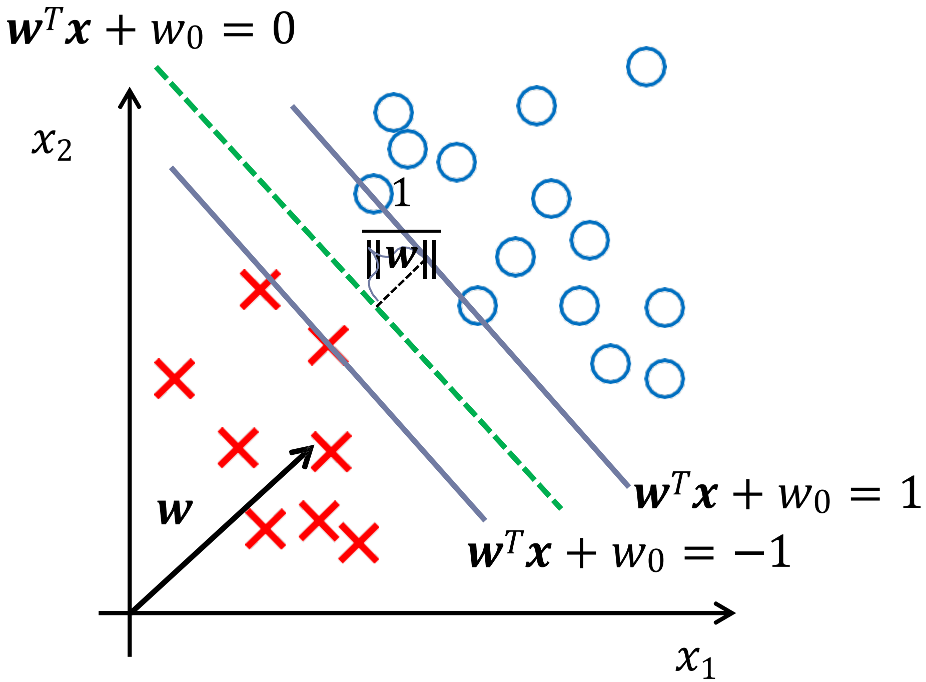
# Hard-margin SVM: Optimization problem

$$\max_{\boldsymbol{w}, w_0} \frac{2}{\|\boldsymbol{w}\|}$$

$$\text{s.t.} \left(\boldsymbol{w}^T \boldsymbol{x}^{(n)} + w_0\right) \geq 1 \quad \forall y^{(n)} = 1$$

$$\left(\boldsymbol{w}^T \boldsymbol{x}^{(n)} + w_0\right) \leq -1 \quad \forall y^{(n)} = -1$$

Margin: $\frac{2}{\|\boldsymbol{w}\|}$



# Hard-margin SVM: Optimization problem

We can equivalently optimize:

$$\min_{\boldsymbol{w}, w_0} \frac{1}{2} \boldsymbol{w}^T \boldsymbol{w}$$

$$\text{s.t.} \quad y^{(n)}\left(\boldsymbol{w}^T \boldsymbol{x}^{(n)} + w_0\right) \geq 1 \quad n = 1, \dots, N$$

- It is a convex Quadratic Programming (QP) problem
  - There are computationally efficient packages to solve it.
  - It has a global minimum (if any).

# Quadratic programming

$$
\begin{aligned}
\min_{\boldsymbol{x}} \frac{1}{2} \boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{c}^T \boldsymbol{x} \\
\text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \\
\boldsymbol{E}\boldsymbol{x} = \boldsymbol{d}
\end{aligned}
$$

# Dual formulation of the SVM

- We are going to introduce the *dual* SVM problem which is equivalent to the original *primal* problem. The dual problem:
  - is often easier
  - gives us further insights into the optimal hyperplane
  - enable us to exploit the kernel trick

# Optimization: Lagrangian multipliers

$$p^* = \min_{\boldsymbol{x}} f(\boldsymbol{x})$$

$$\text{s.t. } g_i(\boldsymbol{x}) \leq 0 \quad i = 1, \dots, m$$

$$h_i(\boldsymbol{x}) = 0 \quad i = 1, \dots, p$$

Lagrangian multipliers

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{i=1}^{m} \alpha_i g_i(\boldsymbol{x}) + \sum_{i=1}^{p} \lambda_i h_i(\boldsymbol{x})$$

$$\max_{\{\alpha_i \geq 0\}, \{\lambda_i\}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{\lambda}) = \begin{cases} \infty & \text{any } g_i(\boldsymbol{x}) > 0 \\ \infty & \text{any } h_i(\boldsymbol{x}) \neq 0 \\ f(\boldsymbol{x}) & \text{otherwise} \end{cases}$$

$$p^* = \min_{\boldsymbol{x}} \max_{\{\alpha_i \geq 0\}, \{\lambda_i\}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{\lambda})$$

$$\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_m]$$

$$\boldsymbol{\lambda} = [\lambda_1, \dots, \lambda_p]$$

# Optimization: Dual problem

- In general, we have:

$$\max_x \min_y h(x,y) \le \min_y \max_x h(x,y)$$

- **Primal problem**: $p^* = \min_{\boldsymbol{x}} \max_{\{\alpha_i \ge 0\},\{\lambda_i\}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{\lambda})$

- **Dual problem**: $d^* = \max_{\{\alpha_i \ge 0\},\{\lambda_i\}} \min_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{\lambda})$
  - Obtained by swapping the order of min and max
  - $d^* \le p^*$

- When the original problem is convex ($f$ and $g$ are convex functions and $h$ is affine), we have strong duality $d^* = p^*$

# Hard-margin SVM: Dual problem

$$\min_{\boldsymbol{w},w_0} \frac{1}{2}\|\boldsymbol{w}\|^2$$

$$\text{s.t.} \quad y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)} + w_0\right) \geq 1 \quad i = 1, \dots, N$$

- By incorporating the constraints through lagrangian multipliers, we will have:

$$\min_{\boldsymbol{w},w_0} \max_{\{\alpha_n \geq 0\}} \left\{\frac{1}{2}\|\boldsymbol{w}\|^2 + \sum_{n=1}^{N} \alpha_n \left(1 - y^{(n)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(n)} + w_0\right)\right)\right\}$$

- Dual problem (changing the order of min and max in the above problem):

$$\max_{\{\alpha_n \geq 0\}} \min_{\boldsymbol{w},w_0} \left\{\frac{1}{2}\|\boldsymbol{w}\|^2 + \sum_{n=1}^{N} \alpha_n \left(1 - y^{(n)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(n)} + w_0\right)\right)\right\}$$

# Hard-margin SVM: Dual problem

$$\max_{\{\alpha_n \geq 0\}} \min_{\boldsymbol{w}, w_0} \mathcal{L}(\boldsymbol{w}, w_0, \boldsymbol{\alpha})$$

$$\mathcal{L}(\boldsymbol{w}, w_0, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + \sum_{n=1}^{N} \alpha_n \left(1 - y^{(n)}(\boldsymbol{w}^T \boldsymbol{x}^{(n)} + w_0)\right)$$

$$\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}, w_0, \boldsymbol{\alpha}) = 0 \Rightarrow \boldsymbol{w} - \sum_{n=1}^{N} \alpha_n y^{(n)} \boldsymbol{x}^{(n)} = \boldsymbol{0}$$

$$\Rightarrow \boldsymbol{w} = \sum_{n=1}^{N} \alpha_n y^{(n)} \boldsymbol{x}^{(n)}$$

$$\frac{\partial \mathcal{L}(\boldsymbol{w}, w_0, \boldsymbol{\alpha})}{\partial w_0} = 0 \Rightarrow \underbrace{-\sum_{n=1}^{N} \alpha_n y^{(n)} = 0}$$

$w_0$ do not appear, instead, a "global" constraint on $\boldsymbol{\alpha}$ is created.

# Substituting

$$\boldsymbol{w} = \sum_{n=1}^{N} \alpha_n y^{(n)} \boldsymbol{x}^{(n)} \qquad \sum_{n=1}^{N} \alpha_n y^{(n)} = 0$$

In the Largrangian

$$\mathcal{L}(\boldsymbol{w}, w_0, \boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \sum_{n=1}^{N} \alpha_n \left(1 - y^{(n)}(\boldsymbol{w}^T\boldsymbol{x}^{(n)} + w_0)\right)$$

# Substituting

$$\mathbf{w} = \sum_{n=1}^{N} \alpha_n y^{(n)} \mathbf{x}^{(n)} \qquad \sum_{n=1}^{N} \alpha_n y^{(n)} = 0$$

In the Largrangian

$$\mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{n=1}^{N} \alpha_n\left( -y^{(n)}(\mathbf{w}^T\mathbf{x}^{(n)} + w_0)\right)$$

We get

$$\mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\alpha}) = \sum_{n=1}^{N} \alpha_n$$

# Substituting

$$\boldsymbol{w} = \sum_{n=1}^{N} \alpha_n y^{(n)} \boldsymbol{x}^{(n)} \qquad \sum_{n=1}^{N} \alpha_n y^{(n)} = 0$$

In the Largrangian

$$\mathcal{L}(\boldsymbol{w}, w_0, \boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \sum_{n=1}^{N} \alpha_n( \quad - y^{(n)}(\boldsymbol{w}^T\boldsymbol{x}^{(n)} \qquad ))$$

We get

$$\mathcal{L}(\boldsymbol{w}, w_0, \boldsymbol{\alpha}) = \sum_{n=1}^{N} \alpha_n$$

# Substituting

$$\boldsymbol{w} = \sum_{n=1}^{N} \alpha_n y^{(n)} \boldsymbol{x}^{(n)} \qquad \sum_{n=1}^{N} \alpha_n y^{(n)} = 0$$

In the Largrangian

$$\mathcal{L}(\boldsymbol{w}, w_0, \boldsymbol{\alpha}) = \frac{1}{2} \boldsymbol{w}^T \boldsymbol{w} + \sum_{n=1}^{N} \alpha_n \left( \quad - y^{(n)} (\boldsymbol{w}^T \boldsymbol{x}^{(n)} \qquad ) \right)$$

We get

$$\mathcal{L}(\boldsymbol{\alpha}) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y^{(n)} y^{(m)} \boldsymbol{x}^{(n)T} \boldsymbol{x}^{(m)}$$

Maximize w.r.t. $\boldsymbol{\alpha}$ subject to $\alpha_n \geq 0$ for $n = 1, \ldots, N$ and $\sum_{n=1}^{N} \alpha_n y^{(n)} = 0$

# Hard-margin SVM: Dual problem

$$\max_{\boldsymbol{\alpha}} \left\{ \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y^{(n)} y^{(m)} \boldsymbol{x}^{(n)T} \boldsymbol{x}^{(m)} \right\}$$

Subject to $\sum_{n=1}^{N} \alpha_n y^{(n)} = 0$

$$\alpha_n \geq 0 \quad n = 1, \dots, N$$

- It is a convex QP

# Solution

- Quadratic programming:

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\boldsymbol{\alpha}^T \begin{bmatrix} y^{(1)}y^{(1)}{\boldsymbol{x}^{(1)}}^T\boldsymbol{x}^{(1)} & \cdots & y^{(1)}y^{(N)}{\boldsymbol{x}^{(1)}}^T\boldsymbol{x}^{(N)} \\ \vdots & \ddots & \vdots \\ y^{(N)}y^{(1)}{\boldsymbol{x}^{(N)}}^T\boldsymbol{x}^{(1)} & \cdots & y^{(N)}y^{(N)}{\boldsymbol{x}^{(N)}}^T\boldsymbol{x}^{(N)} \end{bmatrix}\boldsymbol{\alpha} + (-\mathbf{1})^T\boldsymbol{\alpha}$$

$$\begin{aligned} \text{s.t.}\ -\boldsymbol{\alpha} &\leq \mathbf{0} \\ \boldsymbol{y}^T\boldsymbol{\alpha} &= \mathbf{0} \end{aligned}$$

# Finding the hyperplane

- After finding $\boldsymbol{\alpha}$ by QP,we find $\boldsymbol{w}$:

$$\boldsymbol{w} = \sum_{n=1}^{N} \alpha_n y^{(n)} \boldsymbol{x}^{(n)}$$

- How to find $w_0$?
  - we discuss it after introducing support vectors

# Karush-Kuhn-Tucker (KKT) conditions

- Necessary conditions for the solution $[\boldsymbol{w}^*, w_0^*, \boldsymbol{\alpha}^*]$:
  - $\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}, w_0, \boldsymbol{\alpha}) \Big|_{\boldsymbol{w}^*, w_0^*, \boldsymbol{\alpha}^*} = 0$
  - $\dfrac{\partial \mathcal{L}(\boldsymbol{w}, w_0, \boldsymbol{\alpha})}{\partial w_0} \Big|_{\boldsymbol{w}^*, w_0^*, \boldsymbol{\alpha}^*} = 0$
  - $\alpha_n^* \geq 0 \quad n = 1, \dots, N$
  - $y^{(n)}\left(\boldsymbol{w}^{*T} \boldsymbol{x}^{(n)} + w_0^*\right) \geq 1 \quad n = 1, \dots, N$
  - $\alpha_i^* \left(1 - y^{(n)}\left(\boldsymbol{w}^{*T} \boldsymbol{x}^{(n)} + w_0^*\right)\right) = 0 \quad n = 1, \dots, N$

$$\min_{\boldsymbol{x}} f(\boldsymbol{x})$$
$$\text{s.t. } g_i(\boldsymbol{x}) \leq 0 \quad i = 1, \dots, m$$

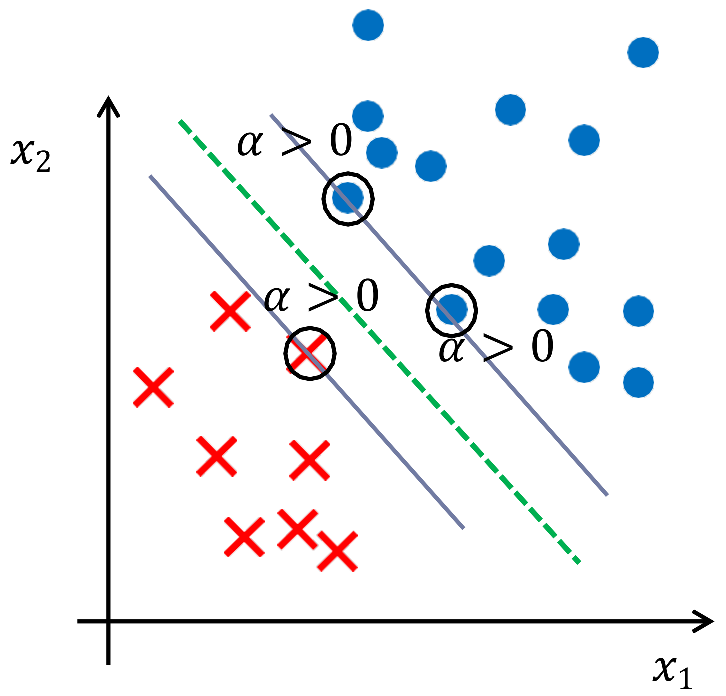
$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\alpha}) = f(\boldsymbol{x}) + \sum \alpha_i \, g_i(\boldsymbol{x})$$

In general, the optimal $\boldsymbol{x}^*, \boldsymbol{\alpha}^*$ satisfies KKT conditions:

$$\nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\alpha}) \Big|_{\boldsymbol{x}^*, \boldsymbol{\alpha}^*} = 0$$
$$\alpha_i^* \geq 0 \quad i = 1, \dots, m$$
$$g_i(\boldsymbol{x}^*) \leq 0 \quad i = 1, \dots, m$$
$$\alpha_i^* g_i(\boldsymbol{x}^*) = 0 \quad i = 1, \dots, m$$

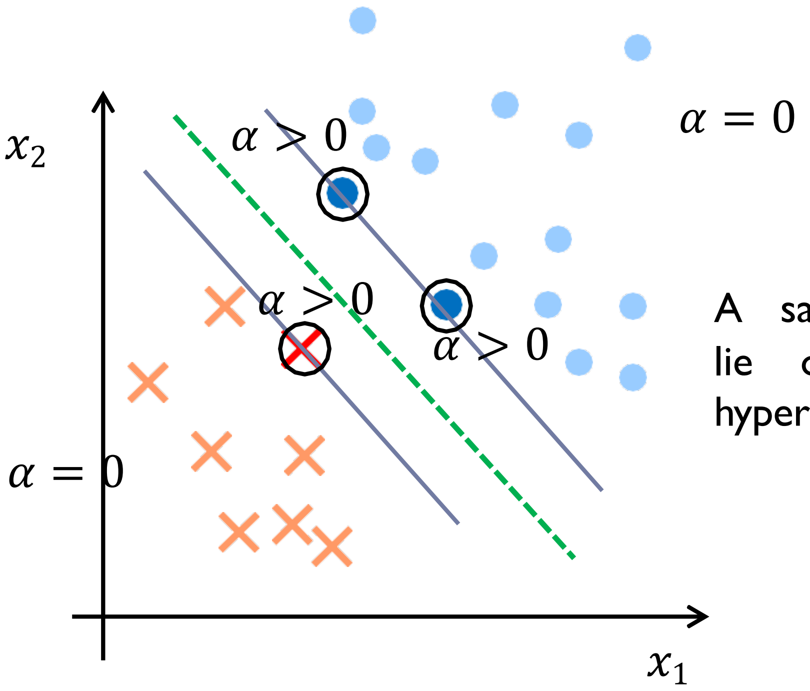
# Hard-margin SVM: Support vectors

- **Inactive** constraint: $y^{(n)}(\mathbf{w}^T\mathbf{x}^{(n)} + w_0) > 1$
  - $\Rightarrow \alpha_n = 0$ and thus $\mathbf{x}^{(n)}$ is not a support vector.
- **Active** constraint: $y^{(n)}(\mathbf{w}^T\mathbf{x}^{(n)} + w_0) = 1$
  - $\Rightarrow \alpha_n$ can be greater than 0 and thus $\mathbf{x}^{(i)}$ can be a support vector.



# Hard-margin SVM: Support vectors

- **Inactive** constraint: $y^{(n)}(\boldsymbol{w}^T\boldsymbol{x}^{(n)} + w_0) > 1$
  - $\Rightarrow \alpha_n = 0$ and thus $\boldsymbol{x}^{(n)}$ is not a support vector.
- **Active** constraint: $y^{(n)}(\boldsymbol{w}^T\boldsymbol{x}^{(n)} + w_0) = 1$

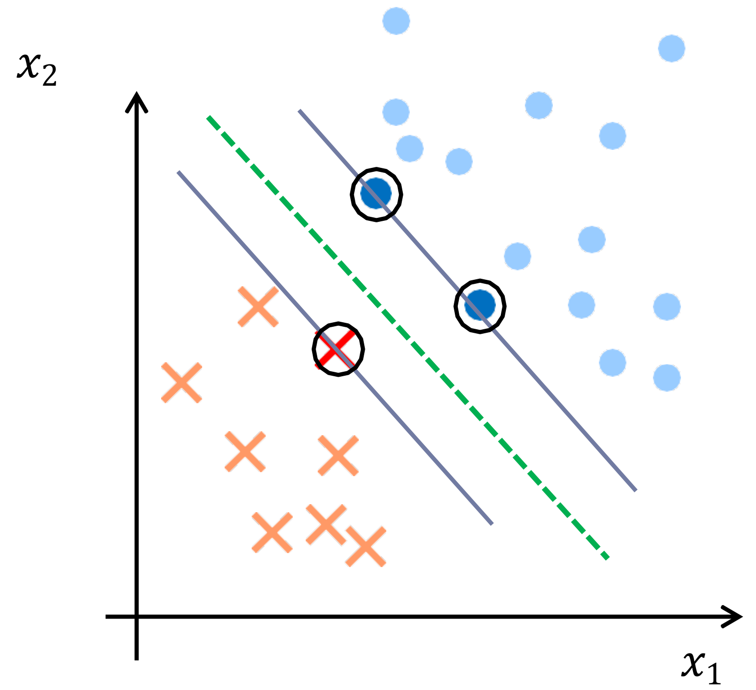



A sample with $\alpha_n = 0$ can also lie on one of the margin hyperplanes

# Hard-margin SVM: Support vectors

- Support Vectors (SVs)= $\{\boldsymbol{x}^{\,n} \quad \alpha_n > 0\}$
- The **direction** of hyper-plane can be found only based on support vectors:

$$\boldsymbol{w} = \sum_{\alpha_n > 0} \alpha_n \, y^{(n)} \boldsymbol{x}^{(n)}$$



# Finding the hyperplane

- After finding $\boldsymbol{\alpha}$ by QP,we find $\boldsymbol{w}$:

$$\boldsymbol{w} = \sum_{n=1}^{N} \alpha_n y^{(n)} \boldsymbol{x}^{(n)}$$

- How to find $w_0$?
  - Each of the samples that has $\alpha_s > 0$ is on the margin, thus we solve for $w_0$ using any of SVs:

$$\left|\boldsymbol{w}^T \boldsymbol{x}^{(s)} + w_0\right| = 1$$

$$y^{(s)}\left(\boldsymbol{w}^T \boldsymbol{x}^{(s)} + w_0\right) = 1$$

$$\Rightarrow w_0 = y^{(s)} - \boldsymbol{w}^T \boldsymbol{x}^{(s)}$$

# Hard-margin SVM: Dual problem Classifying new samples using only SVs

- Classification of a new sample $\boldsymbol{x}$:

$$y = \text{sign}\left(w_0 + \boldsymbol{w}^T \boldsymbol{x}\right)$$

$$y = \text{sign}\left(w_0 + \left(\sum_{\alpha_n > 0} \alpha_n y^{(n)} \boldsymbol{x}^{(n)}\right)^T \boldsymbol{x}\right)$$

$$y = \text{sign}(\underbrace{y^{(s)} - \sum_{\alpha_n > 0} \alpha_n y^{(n)} \boldsymbol{x}^{(n)T} \boldsymbol{x}^{(s)}}_{w_0} + \sum_{\alpha_n > 0} \alpha_n y^{(n)} \boldsymbol{x}^{(n)T} \boldsymbol{x})$$

Support vectors are sufficient to predict labels of new samples

- The classifier is based on the expansion in terms of dot products of $\boldsymbol{x}$ with support vectors.

# Hard-margin SVM: Dual problem

$$\max_{\boldsymbol{\alpha}} \left\{ \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y^{(n)} y^{(m)} \boldsymbol{x}^{(n)^T} \boldsymbol{x}^{(m)} \right\}$$

Subject to $\sum_{n=1}^{N} \alpha_n y^{(n)} = 0$

$$\alpha_n \geq 0 \quad n = 1, \dots, N$$

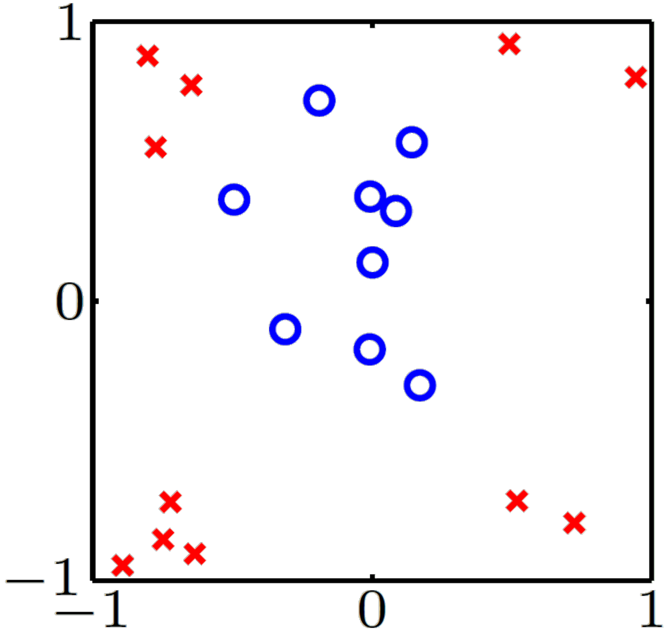
- Only the dot product of each pair of training data appears in the optimization problem
  - An important property that is helpful to extend to non-linear SVM

# In the transformed space

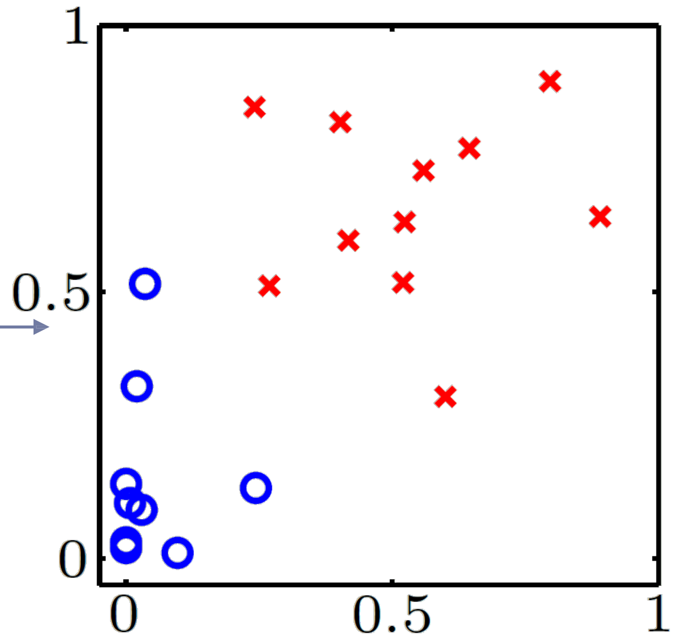
$$\max_{\boldsymbol{\alpha}} \left\{ \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y^{(n)} y^{(m)} \phi\left(\boldsymbol{x}^{(n)}\right)^T \phi\left(\boldsymbol{x}^{(m)}\right) \right\}$$

Subject to $\sum_{n=1}^{N} \alpha_n y^{(n)} = 0$

$\alpha_n \geq 0 \quad n = 1, \ldots, N$

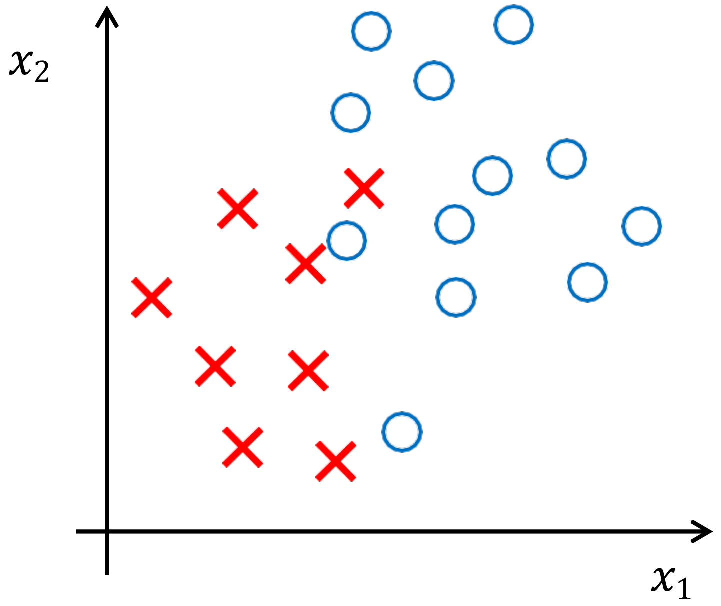


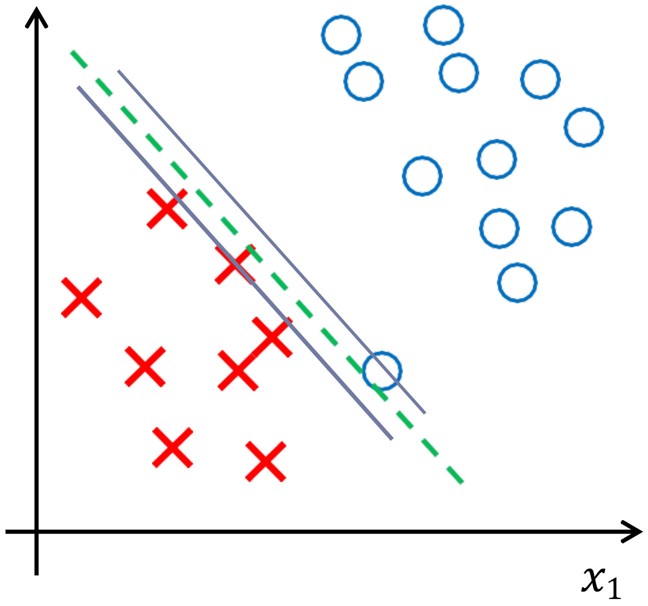

$\phi(.)$



# Beyond linear separability

- How to extend the hard-margin SVM to allow classification error
  - Overlapping classes that can be approximately separated by a linear boundary
  - Noise in the linearly separable classes



# Beyond linear separability: Soft-margin SVM

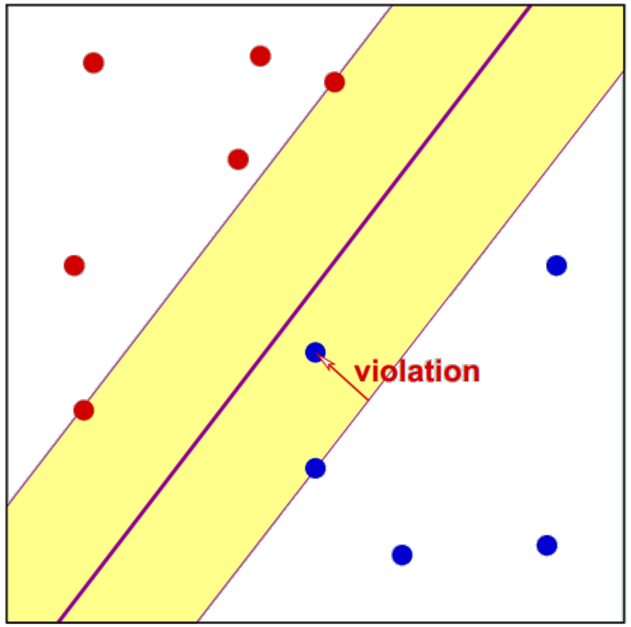
- Minimizing the number of misclassified points?!
  - NP-complete

- Soft margin:
  - Maximizing a margin while trying to minimize the *distance* between misclassified points and their correct margin plane

# Error measure

- Margin violation amount $\xi_n$ ($\xi_n \geq 0$):
  - $y^{(n)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(n)} + w_0\right) \geq 1 - \xi_n$

- Total violation: $\sum_{n=1}^{N} \xi_n$



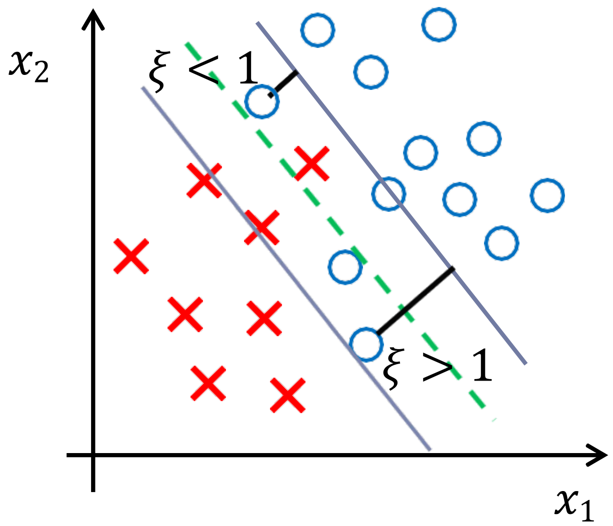
# Soft-margin SVM: Optimization problem

- SVM with slack variables: allows samples to fall within the margin, but penalizes them

$$\min_{\boldsymbol{w}, w_0, \{\xi_n\}_{n=1}^{N}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C \sum_{n=1}^{N} \xi_n$$

$$\text{s.t.} \quad y^{(n)}\left(\boldsymbol{w}^T \boldsymbol{x}^{(n)} + w_0\right) \geq 1 - \xi_n \quad n = 1, \dots, N$$

$$\xi_n \geq 0$$




$\xi_n$: **slack** variables

$0 < \xi_n < 1$: if $\boldsymbol{x}^{(n)}$ is correctly classified but inside margin

$\xi_n > 1$: if $\boldsymbol{x}^{(n)}$ is misclassifed

# Soft-margin SVM

- linear penalty (hinge loss) for a sample if it is misclassified or lied in the margin
  - tries to maintain $\xi_n$ small while maximizing the margin.
  - always finds a solution (as opposed to hard-margin SVM)
  - more robust to the outliers
- Soft margin problem is still a convex QP

# Soft-margin SVM: Parameter $C$

- $C$ is a tradeoff parameter:
  - small $C$ allows margin constraints to be easily ignored
    - large margin
  - large $C$ makes constraints hard to ignore
    - narrow margin
- $C \to \infty$ enforces all constraints: hard margin
- $C$ can be determined using a technique like cross-validation

# Soft-margin SVM: Cost function

$$\min_{\boldsymbol{w}, w_0, \{\xi_n\}_{n=1}^N} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n=1}^{N} \xi_n$$

$$\text{s.t.} \quad y^{(n)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(n)} + w_0\right) \geq 1 - \xi_n \quad n = 1, \dots, N$$

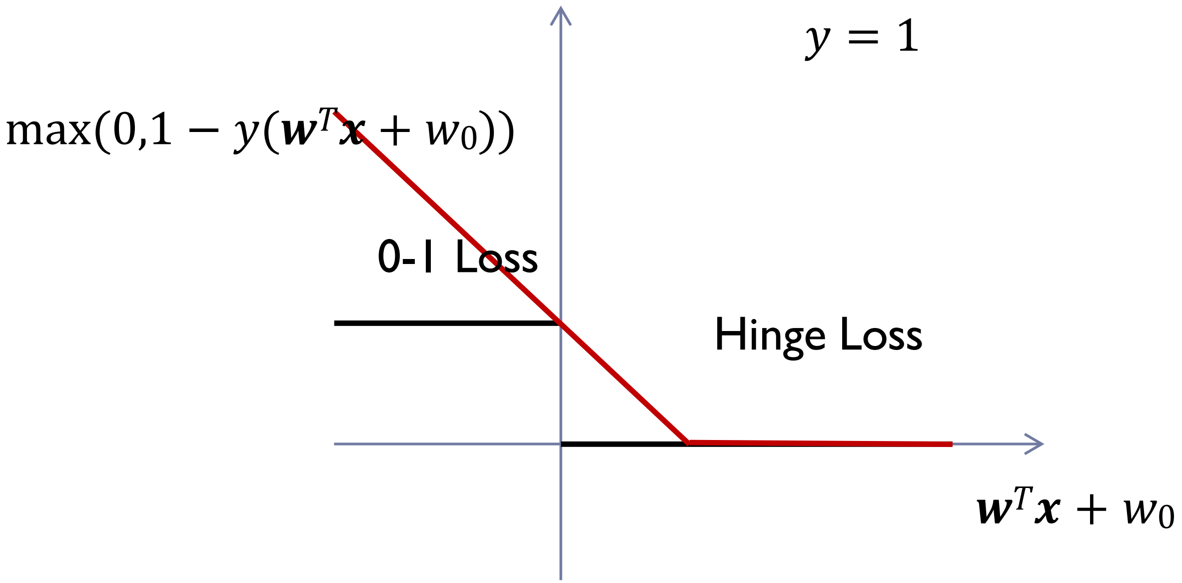
$$\xi_n \geq 0$$

- It is equivalent to the unconstrained optimization problem:

$$\min_{\boldsymbol{w}, w_0} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n=1}^{N} \max(0, 1 - y^{(n)}(\boldsymbol{w}^T\boldsymbol{x}^{(n)} + w_0))$$

# SVM loss function

- Hinge loss vs. 0-1 loss



# Lagrange formulation

$$\mathcal{L}(\boldsymbol{w}, w_0, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n=1}^{N}\xi_n + \sum_{n=1}^{N}\alpha_n\left(1 - \xi_n - y^{(n)}(\boldsymbol{w}^T\boldsymbol{x}^{(n)} + w_0)\right) - \sum_{n=1}^{N}\beta_n\xi_n$$

- Minimize w.r.t. $\boldsymbol{w}, w_0, \boldsymbol{\xi}$ and maximize w.r.t. $\alpha_n \geq 0$ and $\beta_n \geq 0$

$$\min_{\boldsymbol{w}, w_0, \{\xi_n\}_{n=1}^{N}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n=1}^{N}\xi_n$$

$$\text{s.t.} \quad y^{(n)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(n)} + w_0\right) \geq 1 - \xi_n \quad n = 1, \ldots, N$$

$$\xi_n \geq 0$$

# Soft-margin SVM: Dual problem

$$\max_{\boldsymbol{\alpha}} \left\{ \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y^{(n)} y^{(m)} \boldsymbol{x}^{(n)^T} \boldsymbol{x}^{(m)} \right\}$$

Subject to $\sum_{n=1}^{N} \alpha_n y^{(n)} = 0$

$$0 \leq \alpha_n \leq C \quad n = 1, \ldots, N$$

- After solving the above quadratic problem, $\boldsymbol{w}$ is find as:

$$\boldsymbol{w} = \sum_{n=1}^{N} \alpha_n \, y^{(n)} \boldsymbol{x}^{(n)}$$

# Soft-margin SVM: Support vectors

- SupportVectors: $\alpha_n > 0$
  - If $0 < \alpha_n < C$ (**margin** support vector)

    SVs on the margin

    $$y^{(n)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(n)} + w_0\right) = 1 \qquad (\xi_n = 0)$$

  - If $\alpha = C$ (**non-margin** support vector)

    SVs on or over the margin

    $$y^{(n)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(n)} + w_0\right) < 1 \qquad (\xi_n > 0)$$

$$C - \alpha_n - \beta_n = 0$$

# Nonlinear SVM

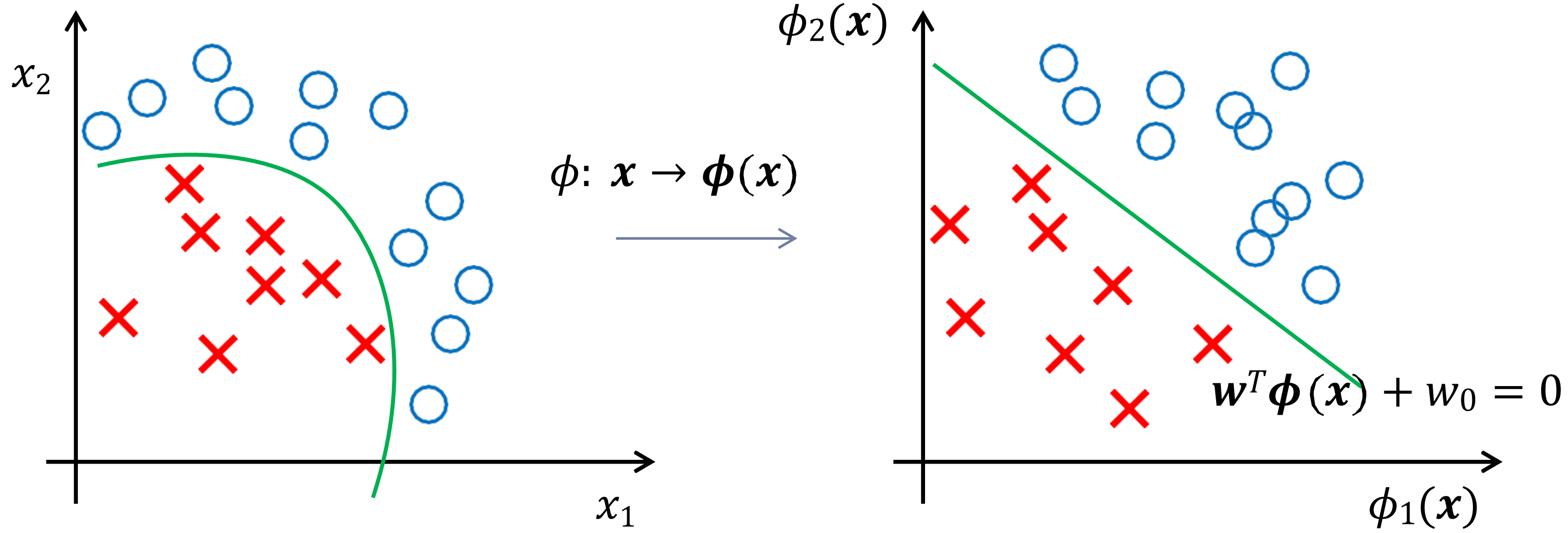
- Assume a transformation $\phi: \mathbb{R}^d \to \mathbb{R}^m$ on the feature space
  - $\boldsymbol{x} \to \boldsymbol{\phi}(\boldsymbol{x})$

$\boldsymbol{\phi}(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}), \dots, \phi_m(\boldsymbol{x})]$

$\{\phi_1(\boldsymbol{x}),\dots,\phi_m(\boldsymbol{x})\}$: set of basis functions (or features)

$\phi_i(\boldsymbol{x}): \mathbb{R}^d \to \mathbb{R}$

- Find a hyper-plane in the transformed feature space:



# Soft-margin SVM in a transformed space: Primal problem

- Primal problem:

$$\min_{\boldsymbol{w},w_0} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n=1}^{N} \xi_n$$

$$\text{s.t.} \quad y^{(n)}\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}^{(n)}) + w_0\right) \geq 1 - \xi_n \quad n = 1, \dots, N$$

$$\xi_n \geq 0$$

  - $\boldsymbol{w} \in \mathbb{R}^m$: the weights that must be found
  - If $m \gg d$ (very high dimensional feature space) then there are many more parameters to learn

# Soft-margin SVM in a transformed space: Dual problem

- Optimization problem:

$$\max_{\boldsymbol{\alpha}} \left\{ \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y^{(n)} y^{(m)} \boldsymbol{\phi}\left(\boldsymbol{x}^{(n)}\right)^T \boldsymbol{\phi}\left(\boldsymbol{x}^{(m)}\right) \right\}$$

Subject to $\sum_{n=1}^{N} \alpha_n y^{(n)} = 0$

$$0 \leq \alpha_n \leq C \quad n = 1, \dots, N$$

- If we have inner products $\boldsymbol{\phi}\left(\boldsymbol{x}^{(i)}\right)^T \boldsymbol{\phi}\left(\boldsymbol{x}^{(j)}\right)$, only $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_N]$ needs to be learnt.
  - not necessary to learn $m$ parameters as opposed to the primal problem

# Classifying a new data

$$y = sign\left(w_0 \ + \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x})\right)$$

where $\boldsymbol{w} = \sum_{\alpha_n > 0} \alpha_n \, y^{(n)} \boldsymbol{\phi}(\boldsymbol{x}^{(n)})$

and $w_0 \ = y^{(s)} - \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}^{(s)})$

# Kernel SVM

- Learns linear decision boundary in a high dimension space without explicitly working on the mapped data

- Let $\boldsymbol{\phi}(\boldsymbol{x})^T\boldsymbol{\phi}(\boldsymbol{x}') = K(\boldsymbol{x}, \boldsymbol{x}')$ (kernel)

- Example: $\boldsymbol{x} = [x_1, x_2]$ and second-order $\boldsymbol{\phi}$:

$$\boldsymbol{\phi}(\boldsymbol{x}) = [1, x_1, x_2, x_1^2, x_2^2, x_1 x_2]$$

$$K(\boldsymbol{x}, \boldsymbol{x}') = 1 + x_1 x_1' + x_2 x_2' + x_1^2 x_1'^2 + x_2^2 x_2'^2 + x_1 x_1' x_2 x_2'$$

# Kernel trick

- Compute $K(\boldsymbol{x}, \boldsymbol{x}')$ without transforming $\boldsymbol{x}$ and $\boldsymbol{x}'$

- Example: Consider $K(\boldsymbol{x}, \boldsymbol{x}') = (1 + \boldsymbol{x}^T\boldsymbol{x}')^2$

$$= (1 + x_1x_1' + x_2x_2')^2$$
$$= 1 + 2x_1x_1' + 2x_2x_2' + x_1^2x_1'^2 + x_2^2x_2'^2 + 2x_1x_1'x_2x_2'$$

This is an inner product in:

$$\boldsymbol{\phi}(\boldsymbol{x}) = [1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2]$$
$$\boldsymbol{\phi}(\boldsymbol{x}') = [1, \sqrt{2}x_1', \sqrt{2}x_2', x_1'^2, x_2'^2, \sqrt{2}x_1'x_2']$$

# Polynomial kernel: Degree two

- We instead use $K(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^T\boldsymbol{x}' + 1)^2$ that corresponds to:

  $d$-dimensional feature space $\boldsymbol{x} = [x_1, \dots, x_d]^T$

$$\boldsymbol{\phi}(\boldsymbol{x}) = \left[1, \sqrt{2}x_1, \dots, \sqrt{2}x_d, x_1^2, \dots, x_d^2, \sqrt{2}x_1x_2, \dots, \sqrt{2}x_1x_d, \sqrt{2}x_2x_3, \dots, \sqrt{2}x_{d-1}x_d\right]^T$$

# Polynomial kernel

- This can similarly be generalized to d-dimensioan $\boldsymbol{x}$ and $\phi$s are polynomials of order $M$:

$$K(\boldsymbol{x}, \boldsymbol{x}') = (1 + \boldsymbol{x}^T\boldsymbol{x}')^M$$
$$= \left(1 + x_1x_1' + x_2x_2' + \cdots + x_dx_d'\right)^M$$

- Example: SVM boundary for a polynomial kernel
  - $w_0 + \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}) = 0$

    $\Rightarrow w_0 + \sum_{\alpha_i>0} \alpha_i y^{(i)} \boldsymbol{\phi}\left(\boldsymbol{x}^{(i)}\right)^T \boldsymbol{\phi}(\boldsymbol{x}) = 0$

    $\Rightarrow w_0 + \sum_{\alpha_i>0} \alpha_i y^{(i)} k(\boldsymbol{x}^{(i)}, \boldsymbol{x}) = 0$

    $\Rightarrow w_0 + \sum_{\alpha_i>0} \alpha_i y^{(i)} \left(1 + \boldsymbol{x}^{(i)^T}\boldsymbol{x}\right)^M = 0$ $\Rightarrow$ Boundary is a polynomial of order $M$

# Why kernel?

- kernel functions $K$ can indeed be efficiently computed, with a cost proportional to $d$ (the dimensionality of the input) instead of $m$.

- Example: consider the second-order polynomial transform:

$$\boldsymbol{\phi}(\boldsymbol{x}) = [1, x_1, \dots, x_d, x_1^2, x_1 x_2, \dots, x_d x_d]^T \qquad m = 1 + d + d^2$$

$$\boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\phi}(\boldsymbol{x}') = 1 + \sum_{i=1}^{d} x_i x_i' + \underbrace{\sum_{i=1}^{d}\sum_{j=1}^{d} x_i x_j x_i' x_j'}_{\sum_{i=1}^{d} x_i x_i' \times \sum_{j=1}^{d} x_j x_j'} \qquad O(m)$$

$$\boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\phi}(\boldsymbol{x}') = 1 + (x^T x') + (x^T x')^2 \qquad O(d)$$

# Gaussian or RBF kernel

- If $K(\boldsymbol{x}, \boldsymbol{x}^{'})$ is an inner product in some transformed space of x, it is good

- $K(\boldsymbol{x}, \boldsymbol{x}^{'}) = \exp(-\frac{\|\boldsymbol{x}-\boldsymbol{x}^{'}\|^2}{\gamma})$

- Take one dimensional case with $\gamma = 1$:

$$K(x, x^{'}) = \exp(-(x - x^{'})^2)$$

$$= \exp(-x^2)\exp(-x'^2)\exp(2xx')$$

$$= \exp(-x^2)\exp(-x'^2)\sum_{k=1}^{\infty}\frac{2^k x^k x'^k}{k!}$$

# Some common kernel functions

- Linear: $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T\boldsymbol{x}'$
- Polynomial: $k(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^T\boldsymbol{x}' + 1)^M$
- Gaussian: $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{\gamma})$
- Sigmoid: $k(\boldsymbol{x}, \boldsymbol{x}') = \tanh(a\boldsymbol{x}^T\boldsymbol{x}' + b)$

# Kernel formulation of SVM

- Optimization problem:

$$\max_{\boldsymbol{\alpha}} \left\{ \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y^{(n)} y^{(m)} \quad k(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(m)}) \right\}$$

Subject to $\sum_{n=1}^{N} \alpha_n y^{(n)} = 0$

$$0 \le \alpha_n \le C \quad n = 1, \dots, N$$

$$\boldsymbol{Q} = \begin{bmatrix} y^{(1)} y^{(1)} K(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(1)}) & \cdots & y^{(1)} y^{(N)} K(\boldsymbol{x}^{(N)}, \boldsymbol{x}^{(1)}) \\ \vdots & \ddots & \vdots \\ y^{(N)} y^{(1)} K(\boldsymbol{x}^{(N)}, \boldsymbol{x}^{(1)}) & \cdots & y^{(N)} y^{(N)} K(\boldsymbol{x}^{(N)}, \boldsymbol{x}^{(N)}) \end{bmatrix}$$

# Classifying a new data

$$y = sign\left(w_0 + \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x})\right)$$

$$\text{where } \boldsymbol{w} = \sum_{\alpha_n > 0} \alpha_n \, y^{(n)} \boldsymbol{\phi}(\boldsymbol{x}^{(n)})$$

$$\text{and } w_0 = y^{(s)} - \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}^{(s)})$$

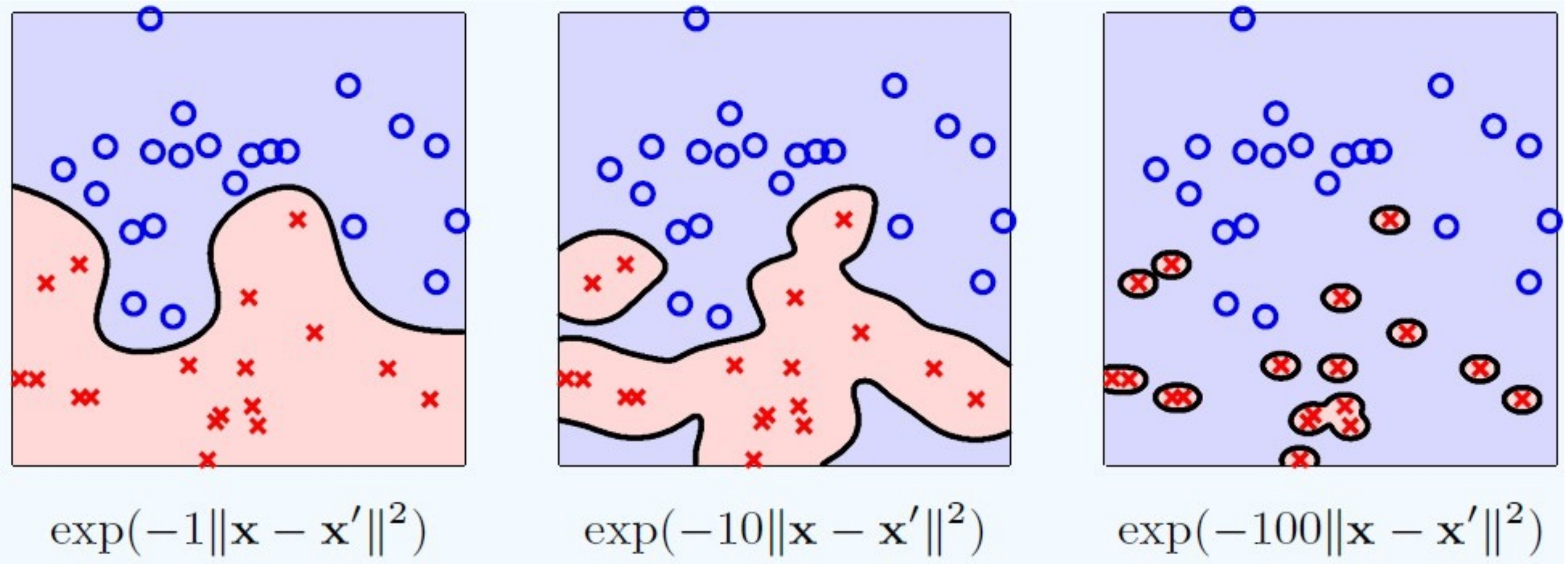
$$y = sign\left(w_0 + \sum_{\alpha_n > 0} \alpha_n \, y^{(n)} \, k(\boldsymbol{x}^{(n)}, \boldsymbol{x})\right)$$

$$w_0 = y^{(s)} - \sum_{\alpha_n > 0} \alpha_n \, y^{(n)} \, k(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(s)})$$

# Gaussian kernel

- Example: SVM boundary for a gaussian kernel
  - Considers a Gaussian function around each data point.
  - $w_0 + \sum_{\alpha_i > 0} \alpha_i y^{(i)} \exp(-\frac{\|\boldsymbol{x} - \boldsymbol{x}^{(i)}\|^2}{\sigma}) = 0$
  - SVM + Gaussian Kernel can classify any arbitrary training set
    - Training error is zero when $\sigma \to 0$
      - All samples become support vectors (likely overfiting)

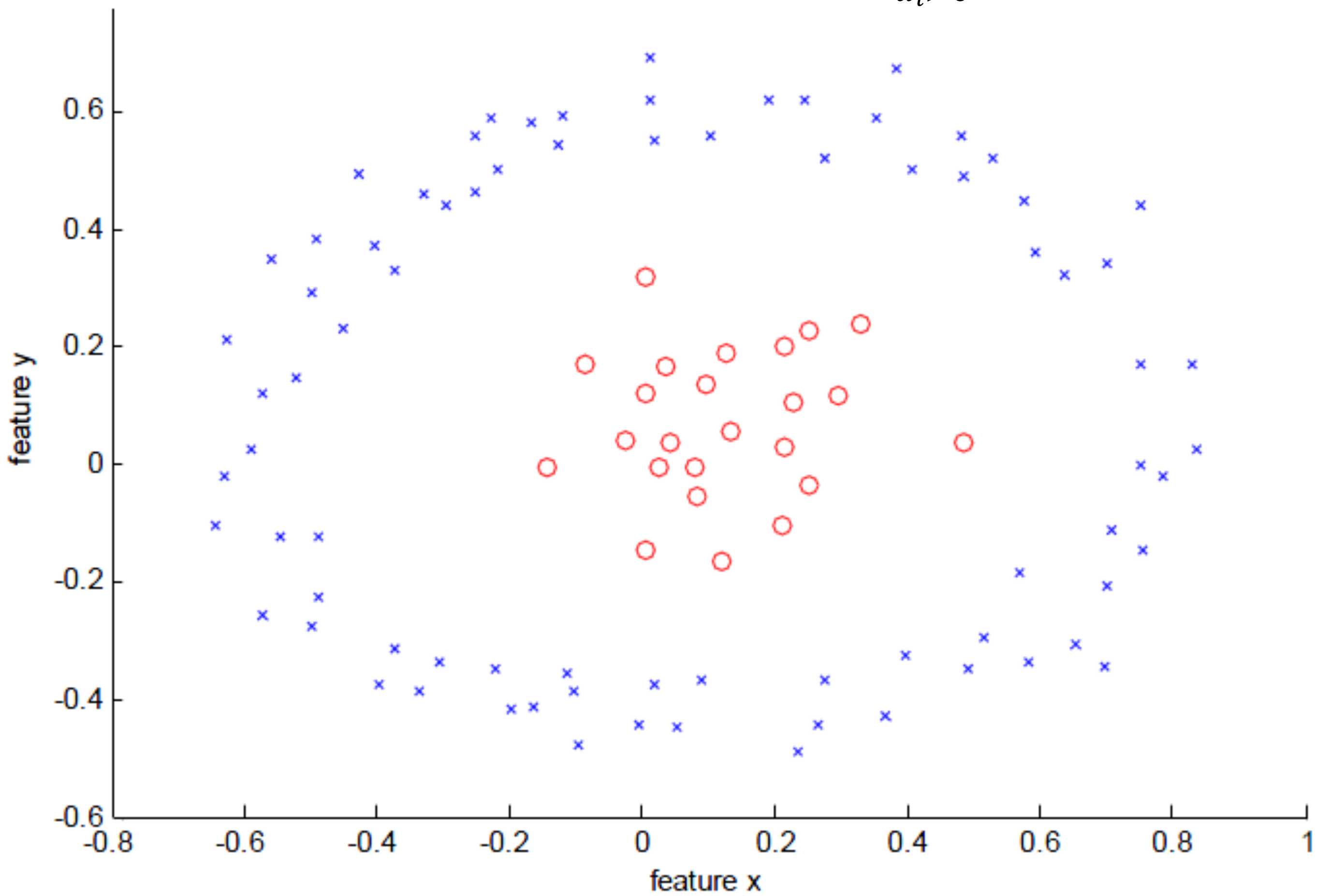
# Hard margin Example




- For narrow Gaussian (large $\sigma$), even the protection of a large margin cannot suppress overfitting.

# SVM Gaussian kernel: Example

$$f(\boldsymbol{x}) = w_0 + \sum_{\alpha_i > 0} \alpha_i y^{(i)} \exp\left(-\frac{\left\|\boldsymbol{x} - \boldsymbol{x}^{(i)}\right\|^2}{2\sigma^2}\right)$$






This example has been adopted from Zisserman's slides

# Kernel trick: Idea

- Kernel trick $\rightarrow$ Extension of many well-known algorithms to kernel-based ones
  - By substituting the dot product with the kernel function
    - $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\phi}(\boldsymbol{x}')$
    - $k(\boldsymbol{x}, \boldsymbol{x}')$ shows the dot product of $\boldsymbol{x}$ and $\boldsymbol{x}'$ in the transformed space.

- Idea: when the input vectors appears only in the form of dot products, we can use kernel trick
  - Solving the problem without explicitly mapping the data
    - Explicit mapping is expensive if $\boldsymbol{\phi}(\boldsymbol{x})$ is very high dimensional

# Constructing kernels

- Construct kernel functions directly
  - Ensure that it is a valid kernel
    - Corresponds to an inner product in some feature space.

- Example: $k(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^T \boldsymbol{x}')^2$
  - Corresponding mapping: $\boldsymbol{\phi}(\boldsymbol{x}) = \left[x_1^2, \sqrt{2}x_1 x_2, x_2^2\right]^T$ for $\boldsymbol{x} = [x_1, x_2]^T$

- We need a way to test whether a kernel is valid without having to construct $\boldsymbol{\phi}(\boldsymbol{x})$

## Valid kernel: Necessary & sufficient conditions

[Shawe-Taylor & Cristianini 2004]

- Gram matrix $\boldsymbol{K}_{N\times N}$: $K_{ij} = k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$
  - Restricting the kernel function to a set of points $\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(N)}\}$

$$K = \begin{bmatrix} k(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(1)}) & \cdots & k(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(N)}) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}^{(N)}, \boldsymbol{x}^{(1)}) & \cdots & k(\boldsymbol{x}^{(N)}, \boldsymbol{x}^{(N)}) \end{bmatrix}$$

- **Mercer** Theorem: The kernel matrix is **Symmetric Positive Semi-Definite** (for any choice of data points)
  - Any symmetric positive definite matrix can be regarded as a kernel matrix, that is as an inner product matrix in some space

# Extending linear methods to kernelized ones

- Kernelized version of linear methods
  - Linear methods are famous
    - Unique optimal solutions, faster learning algorithms, and better analysis
  - However, we often require nonlinear methods in real-world problems and so we can use kernel-based version of these linear algorithms

- Replacing inner products with kernels in linear algorithms ⇒ very flexible methods
  - We can operate in the mapped space without ever computing the coordinates of the data in that space

# Which information can be obtained from kernel?

- Example: we know all pairwise distances
  - $d\left(\boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{z})\right)^2 = ||\boldsymbol{\phi}(\boldsymbol{x}) - \boldsymbol{\phi}(\boldsymbol{z})||^2 = k(\boldsymbol{x}, \boldsymbol{x}) + k(\boldsymbol{z}, \boldsymbol{z}) - 2k(\boldsymbol{x}, \boldsymbol{z})$
  - Therefore, we also know distance of points from center of mass of a set

- Many dimensionality reduction, clustering, and classification methods can be described according to pairwise distances.
  - This allow us to introduce kernelized versions of them

# Kernels for structured data

- Kernels also can be defined on general types of data
  - Kernel functions do not need to be defined over vectors
    - just we need a symmetric positive definite matrix
- Thus, many algorithms can work with general (non-vectorial) data
  - Kernels exist to embed strings, trees, graphs, …
- This may be more important than nonlinearity
  - kernel-based version of classical learning algorithms for recognition of structured data

# Kernel trick advantages: summary

- Operating in the mapped space without ever computing the coordinates of the data in that space
- Besides vectors, we can introduce kernel functions for structured data (graphs, strings, etc.)
- Much of the geometry of the data in the embedding space is contained in all pairwise dot products
- In many cases, inner product in the embedding space can be computed efficiently.

# Resources

- C. Bishop, "Pattern Recognition and Machine Learning", Chapter 6.1-6.2, 7.1.

- Yaser S. Abu-Mostafa, et al., "Learning from Data", Chapter 8.