Support Vector Machine (SVM) and Kernel Methods

CE-717: Machine Learning
Sharif University of Technology
Fall 2016

Soleymani
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?

- **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

Larger margin provides better generalization to unseen data
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

  ![Diagram showing SVM decision boundaries](image)

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

  ![Diagram showing larger margin](image)
Finding \( \mathbf{w} \) with large margin

- **Two preliminaries:**
  - Pull out \( w_0 \)
  - \( \mathbf{w} \) is \([w_1, \ldots, w_d]\)
  
  \[
  \mathbf{w}^T \mathbf{x} + w_0 = 0
  \]
  We have no \( x_0 \)

- **Normalize \( \mathbf{w}, w_0 \)**
  - Let \( x^{(n)} \) be the nearest point to the plane
  - \( |\mathbf{w}^T x^{(n)} + w_0| = 1 \)
Distance between an $x^{(n)}$ and the plane

$$\text{distance} = \frac{|w^T x^{(n)} + w_0|}{||w||}$$
The optimization problem

\[
\begin{align*}
\max_{\mathbf{w}, \mathbf{w}_0} & \quad \frac{2}{\|\mathbf{w}\|} \\
\text{s. t.} & \quad \min_{n=1, \ldots, N} |\mathbf{w}^T \mathbf{x}^{(n)} + \mathbf{w}_0| = 1
\end{align*}
\]

From all the hyperplanes that correctly classify data

\[
\begin{align*}
\text{Notice: } & \quad |\mathbf{w}^T \mathbf{x}^{(n)} + \mathbf{w}_0| = y^{(n)}(\mathbf{w}^T \mathbf{x}^{(n)} + \mathbf{w}_0) \\
\min_{\mathbf{w}, \mathbf{w}_0} & \quad \frac{1}{2} \|\mathbf{w}\|^2 \\
\text{s. t.} & \quad y^{(n)}(\mathbf{w}^T \mathbf{x}^{(n)} + \mathbf{w}_0) \geq 1 \quad n = 1, \ldots, N
\end{align*}
\]
Hard-margin SVM: Optimization problem

\[
\max_{w, w_0} \frac{2}{\|w\|} \\
\text{s. t. } |w^T x^{(n)} + w_0| \geq 1, \ n = 1, \ldots, N
\]
Hard-margin SVM: Optimization problem

\[
\max_{\mathbf{w}, w_0} \frac{2}{\|\mathbf{w}\|} \quad \text{s.t.} \quad (\mathbf{w}^T \mathbf{x}^{(n)} + w_0) \geq 1 \quad \forall y^{(n)} = 1 \\
(\mathbf{w}^T \mathbf{x}^{(n)} + w_0) \leq -1 \quad \forall y^{(n)} = -1
\]
Hard-margin SVM: Optimization problem

We can equivalently optimize:

\[
\begin{align*}
\min_{w, w_0} & \quad \frac{1}{2} w^T w \\
\text{s.t.} & \quad y^{(n)} (w^T x^{(n)} + w_0) \geq 1 \quad n = 1, \ldots, N
\end{align*}
\]

- 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

\[
\min_{x} \frac{1}{2} x^T Q x + c^T x \\
\text{s.t. } A x \leq b \\
E x = d
\]
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

\[ p^* = \min_x f(x) \]
\[ \text{s.t. } g_i(x) \leq 0 \quad i = 1, \ldots, m \]
\[ h_i(x) = 0 \quad i = 1, \ldots, p \]

Lagrangian multipliers

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

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

\[ p^* = \min_x \max_{\{\alpha_i \geq 0\},\{\lambda_i\}} \mathcal{L}(x, \alpha, \lambda) \]
\[ \alpha = [\alpha_1, \ldots, \alpha_m] \]
\[ \lambda = [\lambda_1, \ldots, \lambda_p] \]
Optimization: Dual problem

In general, we have:
\[
\max_x \min_y h(x, y) \leq \min_y \max_x h(x, y)
\]

Primal problem: 
\[
p^* = \min_x \max_{\{\alpha_i \geq 0\}, \{\lambda_i\}} \mathcal{L}(x, \alpha, \lambda)
\]

Dual problem: 
\[
d^* = \max_{\{\alpha_i \geq 0\}, \{\lambda_i\}} \min_x \mathcal{L}(x, \alpha, \lambda)
\]

Obtained by swapping the order of min and max

\[
d^* \leq p^*
\]

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

\[
\begin{align*}
\min_{\mathbf{w}, \mathbf{w}_0} \quad & \frac{1}{2} \|\mathbf{w}\|^2 \\
\text{s.t.} \quad & y^{(i)} (\mathbf{w}^T \mathbf{x}^{(i)} + w_0) \geq 1 \quad i = 1, \ldots, N
\end{align*}
\]

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

\[
\begin{align*}
\min_{\mathbf{w}, \mathbf{w}_0} \max_{\{\alpha_n \geq 0\}} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{n=1}^{N} \alpha_n \left( 1 - y^{(n)} (\mathbf{w}^T \mathbf{x}^{(n)} + w_0) \right) \right\}
\end{align*}
\]

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

\[
\begin{align*}
\max_{\{\alpha_n \geq 0\}} \min_{\mathbf{w}, \mathbf{w}_0} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{n=1}^{N} \alpha_n \left( 1 - y^{(n)} (\mathbf{w}^T \mathbf{x}^{(n)} + w_0) \right) \right\}
\end{align*}
\]
Hard-margin SVM: Dual problem

\[
\begin{align*}
\max_{\{\alpha_n \geq 0\}} \min_{w, w_0} \mathcal{L}(w, w_0, \alpha) \\
\mathcal{L}(w, w_0, \alpha) &= \frac{1}{2} \|w\|^2 + \sum_{n=1}^{N} \alpha_n (1 - y^{(n)} (w^T x^{(n)} + w_0)) \\
\nabla_w \mathcal{L}(w, w_0, \alpha) &= 0 \Rightarrow w - \sum_{n=1}^{N} \alpha_n y^{(n)} x^{(n)} = 0 \\
&\Rightarrow w = \sum_{n=1}^{N} \alpha_n y^{(n)} x^{(n)} \\
\frac{\partial \mathcal{L}(w, w_0, \alpha)}{\partial w_0} &= 0 \Rightarrow - \sum_{n=1}^{N} \alpha_n y^{(n)} = 0
\end{align*}
\]

\(w_0\) do not appear, instead, a “global” constraint on \(\alpha\) is created.
Substituting

\[ \mathbf{w} = \sum_{n=1}^{N} \alpha_n y^{(n)} \mathbf{x}^{(n)} \quad \sum_{n=1}^{N} \alpha_n y^{(n)} = 0 \]

In the Lagrangian

\[ \mathcal{L}(\mathbf{w}, \mathbf{w}_0, \alpha) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + \sum_{n=1}^{N} \alpha_n \left( 1 - y^{(n)} (\mathbf{w}^T \mathbf{x}^{(n)} + \mathbf{w}_0) \right) \]
Substituting

\[
\mathbf{w} = \sum_{n=1}^{N} \alpha_n y^{(n)} \mathbf{x}^{(n)} \quad \sum_{n=1}^{N} \alpha_n y^{(n)} = 0
\]

In the Lagrangian

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

We get

\[
\mathcal{L}(\mathbf{w}, w_0, \alpha) = \sum_{n=1}^{N} \alpha_n
\]
Substituting

\[
\mathbf{w} = \sum_{n=1}^{N} \alpha_n y^{(n)} \mathbf{x}^{(n)} \quad \sum_{n=1}^{N} \alpha_n y^{(n)} = 0
\]

In the Lagrangian

\[
\mathcal{L}(\mathbf{w}, \mathbf{w}_0, \alpha) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + \sum_{n=1}^{N} \alpha_n \left( - y^{(n)} (\mathbf{w}^T \mathbf{x}^{(n)}) \right)
\]

We get

\[
\mathcal{L}(\mathbf{w}, \mathbf{w}_0, \alpha) = \sum_{n=1}^{N} \alpha_n
\]
Substituting

\[
\mathbf{w} = \sum_{n=1}^{N} \alpha_n y^{(n)} \mathbf{x}^{(n)} \quad \sum_{n=1}^{N} \alpha_n y^{(n)} = 0
\]

In the Lagrangian

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

We get

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

Maximize w.r.t. \( \alpha \) subject to \( \alpha_n \geq 0 \) for \( n = 1, ..., N \) and \( \sum_{n=1}^{N} \alpha_n y^{(n)} = 0 \)
Hard-margin SVM: Dual problem

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

Subject to \[\sum_{n=1}^{N} \alpha_n y^{(n)} = 0\]
\[\alpha_n \geq 0 \quad n = 1, \ldots, N\]

- It is a convex QP

22
Solution

- Quadratic programming:

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

s.t. \(-\alpha \leq 0\)

\[y^T \alpha = 0\]
Finding the hyperplane

- After finding $\alpha$ by QP, we find $w$:
  \[ w = \sum_{n=1}^{N} \alpha_n y^{(n)} x^{(n)} \]

- How to find $w_0$?
  - we discuss it after introducing support vectors
Karush-Kuhn-Tucker (KKT) conditions

- **Necessary conditions for the solution** \([\mathbf{w}^*, \mathbf{w}_0^*, \alpha^*] \): 

  \[ \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \mathbf{w}_0, \alpha) |_{\mathbf{w}^*, \mathbf{w}_0^*, \alpha^*} = 0 \]

  \[ \frac{\partial \mathcal{L}(\mathbf{w}, \mathbf{w}_0, \alpha)}{\partial \mathbf{w}_0} |_{\mathbf{w}^*, \mathbf{w}_0^*, \alpha^*} = 0 \]

  \[ \alpha_n^* \geq 0 \quad n = 1, ..., N \]

  \[ y^{(n)}(\mathbf{w}^T \mathbf{x}^{(n)} + \mathbf{w}_0^*) \geq 1 \quad n = 1, ..., N \]

  \[ \alpha_i^* \left(1 - y^{(n)}(\mathbf{w}^T \mathbf{x}^{(n)} + \mathbf{w}_0^*)\right) = 0 \quad n = 1, ..., N \]

In general, the optimal \( \mathbf{x}^*, \alpha^* \) satisfies KKT conditions:

\[ \min_{\mathbf{x}} f(\mathbf{x}) \quad \text{s.t.} \quad g_i(\mathbf{x}) \leq 0 \quad i = 1, ..., m \]

\[ \mathcal{L}(\mathbf{x}, \alpha) = f(\mathbf{x}) + \sum \alpha_i g_i(\mathbf{x}) \]

\[ \nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \alpha) |_{\mathbf{x}^*, \alpha^*} = 0 \]

\[ \alpha_i^* \geq 0 \quad i = 1, ..., m \]

\[ g_i(\mathbf{x}^*) \leq 0 \quad i = 1, ..., m \]

\[ \alpha_i^* g_i(\mathbf{x}^*) = 0 \quad i = 1, ..., m \]
Karush-Kuhn-Tucker (KKT) conditions

Inactive constraint ($\alpha = 0$)

Active constraint
Hard-margin SVM: Support vectors

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

- **Active** constraint: $y^{(n)}(w^T x^{(n)} + w_0) = 1$
  - $\Rightarrow \alpha_n$ can be greater than 0 and thus $x^{(i)}$ can be a support vector.
Hard-margin SVM: Support vectors

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

A sample with \( \alpha_n = 0 \) can also lie on one of the margin hyperplanes.
Hard-margin SVM: Support vectors

- Support Vectors (SVs) = \{x^{(n)} | \alpha_n > 0\}

- The **direction** of hyper-plane can be found only based on support vectors:

  \[ w = \sum_{\alpha_n > 0} \alpha_n y^{(n)} x^{(n)} \]
Finding the hyperplane

- After finding \( \alpha \) by QP, we find \( w \):

\[
w = \sum_{n=1}^{N} \alpha_n y^{(n)} x^{(n)}
\]

- 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:

\[
|w^T x^{(s)} + w_0| = 1
\]

\[
y^{(s)}(w^T x^{(s)} + w_0) = 1
\]

\[\Rightarrow w_0 = y^{(s)} - w^T x^{(s)}\]
Hard-margin SVM: Dual problem
Classifying new samples using only SVs

Classification of a new sample $x$:

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

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

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

Support vectors are sufficient to predict labels of new samples

The classifier is based on the expansion in terms of dot products of $x$ with support vectors.
Hard-margin SVM: Dual problem

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

Subject to \[ \sum_{n=1}^{N} \alpha_n y^{(n)} = 0 \]
\[ \alpha_n \geq 0 \quad n = 1, \ldots, N \]

- 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

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

Subject to

\[
\sum_{n=1}^{N} \alpha_n y^{(n)} = 0
\]

\[
\alpha_n \geq 0 \quad n = 1, \ldots, N
\]
Beyond linear separability

- When training samples are not linearly separable, it has no solution.

- How to extend it to find a solution even though the classes are not exactly linearly separable.
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 $\xi_n$ ($\xi_n \geq 0$):
  \[ y^{(n)}(w^T x^{(n)} + w_0) \geq 1 - \xi_n \]

- 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

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

s.t. \[ y^{(n)}(\mathbf{w}^T \mathbf{x}^{(n)} + w_0) \geq 1 - \xi_n \quad n = 1, \ldots, N \]
\[ \xi_n \geq 0 \]

\(\xi_n\): slack variables

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

\(\xi_n > 1\): if \(\mathbf{x}^{(n)}\) is misclassified
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

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

s.t. \( y^{(n)}(\mathbf{w}^T \mathbf{x}^{(n)} + w_0) \geq 1 - \xi_n \quad n = 1, \ldots, N \)
\( \xi_n \geq 0 \)

- It is equivalent to the unconstrained optimization problem:

\[
\min_{\mathbf{w}, w_0} \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_{n=1}^N \max(0, 1 - y^{(n)}(\mathbf{w}^T \mathbf{x}^{(n)} + w_0))
\]
SVM loss function

- Hinge loss vs. 0-1 loss

\[
\max(0, 1 - y(w^T x + w_0))
\]

Diagram:
- Hinge Loss
- 0-1 Loss
- \( w^T x + w_0 \)
- \( y = 1 \)
Lagrange formulation

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

- Minimize w.r.t. \( w, w_0, \xi \) and maximize w.r.t. \( \alpha_n \geq 0 \) and \( \beta_n \geq 0 \)

\[
\min_{w, w_0, \{\xi_n\}_{n=1}^N} \frac{1}{2} \|w\|^2 + C \sum_{n=1}^{N} \xi_n \\
\text{s.t. } y^{(n)}(w^T x^{(n)} + w_0) \geq 1 - \xi_n \quad n = 1, \ldots, N \\
\xi_n \geq 0
\]
Lagrange formulation

\[ \mathcal{L}(w, w_0, \xi, \alpha, \beta) = \frac{1}{2} \|w\|^2 + C \sum_{n=1}^{N} \xi_n + \sum_{n=1}^{N} \alpha_n \] (1)
Soft-margin SVM: Dual problem

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

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, \(w\) is found as:

\[
w = \sum_{n=1}^{N} \alpha_n y^{(n)} x^{(n)}
\]
Soft-margin SVM: Support vectors

- **Support Vectors:** $\alpha_n > 0$
  - If $0 < \alpha_n < C$ (**margin** support vector)  
    \[ y^{(n)}(\mathbf{w}^T \mathbf{x}^{(n)} + w_0) = 1 \quad (\xi_n = 0) \]
  - SVs on the margin

- If $\alpha = C$ (**non-margin** support vector)  
  \[ y^{(n)}(\mathbf{w}^T \mathbf{x}^{(n)} + w_0) < 1 \quad (\xi_n > 0) \]
  - SVs on or over the margin

$C - \alpha_n - \beta_n = 0$
SVM: Summary

- Hard margin: maximizing margin

- Soft margin: handling noisy data and overlapping classes
  - Slack variables in the problem

- Dual problems of hard-margin and soft-margin SVM
  - Classifier decision in terms of support vectors

- Dual problems lead us to non-linear SVM method easily by kernel substitution
Not linearly separable data

- Noisy data or overlapping classes
  - Near linearly separable

- Non-linear decision surface
  - Transform to a new feature space
Nonlinear SVM

- Assume a transformation $\phi: \mathbb{R}^d \to \mathbb{R}^m$ on the feature space
  - $x \to \phi(x)$
    - $\phi(x) = [\phi_1(x), \ldots, \phi_m(x)]$
    - $\{\phi_1(x), \ldots, \phi_m(x)\}$: set of basis functions (or features)
    - $\phi_i(x): \mathbb{R}^d \to \mathbb{R}$

- Find a hyper-plane in the transformed feature space:
  - $\mathbf{w}^T \phi(x) + w_0 = 0$
Soft-margin SVM in a transformed space: Primal problem

- **Primal problem:**

\[
\begin{align*}
\min_{w, w_0} & \quad \frac{1}{2} \|w\|^2 + C \sum_{n=1}^{N} \xi_n \\
\text{s.t.} & \quad y^{(n)}(w^T \phi(x^{(n)}) + w_0) \geq 1 - \xi_n \quad n = 1, \ldots, N \\
& \quad \xi_n \geq 0
\end{align*}
\]

- \( w \in \mathbb{R}^m \): 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:**
  \[
  \max_{\alpha} \left\{ \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y^{(n)} y^{(m)} \phi(x^{(n)})^T \phi(x^{(m)}) \right\}
  \]
  Subject to \[\sum_{n=1}^{N} \alpha_n y^{(n)} = 0\]

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

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

\[ \hat{y} = \text{sign}(w_0 + w^T \phi(x)) \]

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

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

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

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

- Example: $x = [x_1, x_2]$ and second-order $\phi$:
  $$\phi(x) = [1, x_1, x_2, x_1^2, x_2^2, x_1x_2]$$

$$K(x, x') = 1 + x_1 x'_1 + x_2 x'_2 + x_1^2 x'_1^2 + x_2^2 x'_2^2 + x_1 x'_1 x_2 x'_2$$
Kernel trick

- Compute $K(x, x')$ without transforming $x$ and $x'$

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

  
  
  \[
  = (1 + x_1 x'_1 + x_2 x'_2)^2 \\
  = 1 + 2x_1 x'_1 + 2x_2 x'_2 + x_1^2 x'_1^2 + x_2^2 x'_2^2 + 2x_1 x'_1 x_2 x'_2
  \]

  This is an inner product in:

  \[
  \phi(x) = [1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1 x_2] \\
  \phi(x') = [1, \sqrt{2}x'_1, \sqrt{2}x'_2, x'_1^2, x'_2^2, \sqrt{2}x'_1 x'_2]
  \]
Polynomial kernel: Degree two

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

  $d$-dimensional feature space $x = [x_1, ..., x_d]^T$

  \[
  \phi(x) = [1, \sqrt{2}x_1, ..., \sqrt{2}x_d, x_1^2, ..., x_d^2, \sqrt{2}x_1x_2, ..., \sqrt{2}x_1x_d, \sqrt{2}x_2x_3, ..., \sqrt{2}x_{d-1}x_d]^T
  \]
Polynomial kernel

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

$$K(x, x') = (1 + x^T x')^M$$

$$= (1 + x_1 x'_1 + x_2 x'_2 + \cdots + x_d x'_d)^M$$

Example: SVM boundary for a polynomial kernel

$$w_0 + w^T \phi(x) = 0$$

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

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

$$\Rightarrow w_0 + \sum_{\alpha_i > 0} \alpha_i y^{(i)} \left(1 + x^{(i)^T} x\right)^M = 0$$  \text{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:

$$\phi(x) = [1, x_1, \ldots, x_d, x_1^2, x_1 x_2, \ldots, x_d x_d]^T \quad m = 1 + d + d^2$$

$$\phi(x)^T \phi(x') = 1 + \sum_{i=1}^{d} x_i x'_i + \sum_{i=1}^{d} \sum_{j=1}^{d} x_i x_j x'_i x'_j \quad O(m)$$

$$\sum_{i=1}^{d} x_i x'_i \times \sum_{j=1}^{d} x_j x'_j$$

$$\phi(x)^T \phi(x') = 1 + (x^T x') + (x^T x')^2 \quad O(d)$$
Gaussian or RBF kernel

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

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

- 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**: $k(x, x') = x^T x'$
- **Polynomial**: $k(x, x') = (x^T x' + 1)^M$
- **Gaussian**: $k(x, x') = \exp\left(-\frac{\|x-x'\|^2}{\gamma}\right)$
- **Sigmoid**: $k(x, x') = \tanh(ax^T x' + b)$
Kernel formulation of SVM

- Optimization problem:

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

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

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

\[
Q = \begin{bmatrix}
    y^{(1)}y^{(1)}K(x^{(1)}, x^{(1)}) & \cdots & y^{(1)}y^{(N)}K(x^{(N)}, x^{(1)}) \\
    \vdots & \ddots & \vdots \\
    y^{(N)}y^{(1)}K(x^{(N)}, x^{(1)}) & \cdots & y^{(N)}y^{(N)}K(x^{(N)}, x^{(N)})
\end{bmatrix}
\]
Classifying a new data

\[ \hat{y} = \text{sign}(w_0 + w^T \phi(x)) \]

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

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

\[ \hat{y} = \text{sign} \left( w_0 + \sum_{\alpha_n > 0} \alpha_n y^{(n)} k(x^{(n)}, x) \right) \]

\[ w_0 = y^{(s)} - \sum_{\alpha_n > 0} \alpha_n y^{(n)} k(x^{(n)}, x^{(s)}) \]
Gaussian kernel

- Example: SVM boundary for a gaussian kernel
  - Considers a Gaussian function around each data point.

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

- SVM + Gaussian Kernel can classify any arbitrary training set
  - Training error is zero when \( \sigma \to 0 \)
    - All samples become support vectors (likely overfitting)
For narrow Gaussian (large $\sigma$), even the protection of a large margin cannot suppress overfitting.
SVM Gaussian kernel: Example

\[ f(x) = w_0 + \sum_{\alpha_i > 0} \alpha_i y^{(i)} \exp\left(-\frac{\|x - x^{(i)}\|^2}{2\sigma^2}\right) \]
SVM Gaussian kernel: Example

\[ \sigma = 1.0 \quad C = \infty \]

This example has been adopted from Zisserman’s slides
SVM Gaussian kernel: Example

$\sigma = 1.0 \quad C = 100$

This example has been adopted from Zisserman’s slides
SVM Gaussian kernel: Example

$\sigma = 1.0 \quad C = 10$

This example has been adopted from Zisserman’s slides
SVM Gaussian kernel: Example

$\sigma = 1.0 \quad C = \infty$

This example has been adopted from Zisserman’s slides
SVM Gaussian kernel: Example

\[ \sigma = 0.25 \quad C = \infty \]
SVM Gaussian kernel: Example

\[ \sigma = 0.1 \quad C = \infty \]

This example has been adopted from Zisserman’s slides
Kernel trick: Idea

- Kernel trick → Extension of many well-known algorithms to kernel-based ones
  - By substituting the dot product with the kernel function
    - \( k(x, x') = \phi(x)^T \phi(x') \)
    - \( k(x, x') \) shows the dot product of \( x \) and \( x' \) 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 \( \phi(x) \) is very high dimensional
Kernel trick: Idea (Cont’d)

- Instead of using a mapping \( \phi: \mathcal{X} \leftarrow \mathcal{F} \) to represent \( x \in \mathcal{X} \) by \( \phi(x) \in \mathcal{F} \), a similarity function \( k: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R} \) is used.
- We specify only an inner product function between points in the transformed space (not their coordinates).
- In many cases, the inner product in the embedding space can be computed efficiently.
Constructing kernels

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

- Example: \( k(x, x') = (x^T x')^2 \)
  - Corresponding mapping: \( \phi(x) = [x_1^2, \sqrt{2}x_1x_2, x_2^2]^T \) for \( x = [x_1, x_2]^T \)

- We need a way to test whether a kernel is valid without having to construct \( \phi(x) \)
Valid kernel: Necessary & sufficient conditions

- Gram matrix \( K_{N \times N} : K_{ij} = k(x^{(i)}, x^{(j)}) \) [Shawe-Taylor & Cristianini 2004]

- Restricting the kernel function to a set of points \( \{x^{(1)}, x^{(2)}, \ldots, x^{(N)}\} \)

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

- **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
Example: kernelized minimum distance classifier

- If $\|x - \mu_1\| < \|x - \mu_2\|$ then assign $x$ to $C_1$

\[
(x - \mu_1)^T (x - \mu_1) < (x - \mu_2)^T (x - \mu_2)
\]

\[
-2x^T \mu_1 + \mu_1^T \mu_1 < -2x^T \mu_2 + \mu_2^T \mu_2
\]
Which information can be obtained from kernel?

- Example: we know all pairwise distances
  \[ d(\phi(x), \phi(z))^2 = ||\phi(x) - \phi(z)||^2 = k(x, x) + k(z, z) - 2k(x, z) \]
  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.
Example: Kernel ridge regression

\[
\min_w \sum_{n=1}^{N} (w^T x^{(n)} - y^{(n)})^2 + \lambda w^T w
\]

\[
\sum_{n=1}^{N} 2x^{(n)}(w^T x^{(n)} - y^{(n)}) + 2\lambda w \Rightarrow w = \sum_{n=1}^{N} \alpha_n x^{(n)}
\]

\[
\alpha_n = -\frac{1}{\lambda} (w^T x^{(n)} - y^{(n)})
\]
Example: Kernel ridge regression (Cont’d)

\[
\min_w \sum_{n=1}^{N} \left( w^T \phi(x^{(n)}) - y^{(n)} \right)^2 + \lambda w^T w
\]

\[w = \sum_{n=1}^{N} \alpha_n \phi(x^{(n)})\]

- Dual representation:

\[
J(\alpha) = \alpha^T \Phi \Phi^T \Phi \Phi^T \alpha - 2 \alpha^T \Phi \Phi^T y + y^T y + \lambda \alpha^T \Phi \Phi^T \alpha
\]

\[
J(\alpha) = \alpha^T KK\alpha - 2 \alpha^T Ky + y^T y + \lambda \alpha^T K\alpha
\]

\[
\nabla_\alpha J(\alpha) = 0 \Rightarrow \alpha = (K + \lambda I_N)^{-1} y
\]
Example: Kernel ridge regression (Cont’d)

- Prediction for new $x$:
  \[ f(x) = w^T \phi(x) \quad w = \Phi^T \alpha \]

  \[
  = \alpha^T \Phi \phi(x) \\
  = \begin{bmatrix}
    K(x^{(1)}, x) \\
    \vdots \\
    K(x^{(N)}, x)
  \end{bmatrix}^T (K + \lambda I_N)^{-1} y
  \]
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 function for objects

- **Sets**: Example of kernel function for sets:

  \[ k(A, B) = 2^{|A \cap B|} \]

- **Strings**: The inner product of the feature vectors for two strings can be defined as

  - e.g. sum over all common subsequences weighted according to their frequency of occurrence and lengths

  \[
  \begin{array}{cccccc}
  A & E & G & A & T & E \\
  E & G & T & E & A & G \\
  \end{array}
  \]

  \[
  \begin{array}{cccccc}
  A & G & A & E & G & A \\
  T & G & A & T & G & \end{array}
  \]
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