Support Vector Machine (SVM) and Kernel Methods
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
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
Hard-margin SVM: Optimization problem

\[
\begin{align*}
\max_{M, w, w_0} & \quad \frac{2M}{\|w\|} \\
\text{s.t.} & \quad (w^T x^{(i)} + w_0) \geq M \quad \forall x^{(i)} \in C_1 \quad \rightarrow \quad y^{(i)} = 1 \\
& \quad (w^T x^{(i)} + w_0) \leq -M \quad \forall x^{(i)} \in C_2 \quad \rightarrow \quad y^{(i)} = -1
\end{align*}
\]
Hard-margin SVM: Optimization problem

\[
\max_{M,w,w_0} \frac{2M}{\|w\|}
\]

s.t. \( y^{(i)}(w^T x^{(i)} + w_0) \geq M \quad i = 1, \ldots, N \)
Hard-margin SVM: Optimization problem

We can set \( w' = \frac{w}{M}, w_0' = \frac{w_0}{M} \):

\[
\max_{w', w_0'} \frac{2}{\|w'\|}
\]

s.t. \( y(i) (w'^T x(i) + w_0') \geq 1 \quad i = 1, \ldots, N \)

The place of boundary and margin lines do not change.
We can equivalently optimize:

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

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

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

- 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 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
**Soft-margin SVM**

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

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

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

- \( \xi_i \): slack variables
- \( \xi_i > 1 \): if \( x^{(i)} \) misclassified
- \( 0 < \xi_i < 1 \): if \( x^{(i)} \) correctly classified but inside margin
Soft-margin SVM

- linear penalty (hinge loss) for a sample if it is misclassified or lied in the margin
  - tries to maintain $\xi_i$ 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

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

s.t. \quad y^{(i)} \left( \mathbf{w}^T \mathbf{x}^{(i)} + w_0 \right) \geq 1 - \xi_i \quad i = 1, \ldots, n

\quad \xi_i \geq 0

It is equivalent to the unconstrained optimization problem:

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

- Hinge loss vs. 0-1 loss

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

\[ y = 1 \]

\[ w^T x + w_0 \]
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
  - enable us to exploit the kernel trick
  - gives us further insights into the optimal hyperplane
Optimization: Lagrangian multipliers

\[ p^* = \min_x f(x) \]

s.t. \( g_i(x) \leq 0 \quad i = 1, \ldots, m \)
\( h_i(x) = 0 \quad i = 1, \ldots, p \)

Lagrangian multipliers

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

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

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

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

- **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

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

- By incorporating the constraints through lagrangian multipliers, we will have:
  \[
  \min_{w, w_0} \max_{\{\alpha_i \geq 0\}} \left\{ \frac{1}{2} \|w\|^2 + \sum_{i=1}^{N} \alpha_i (1 - y^{(i)}(w^T x^{(i)} + w_0)) \right\}
  \]

- Dual problem (changing the order of min and max in the above problem):
  \[
  \max_{\{\alpha_i \geq 0\}} \min_{w, w_0} \left\{ \frac{1}{2} \|w\|^2 + \sum_{i=1}^{N} \alpha_i (1 - y^{(i)}(w^T x^{(i)} + w_0)) \right\}
  \]
Hard-margin SVM: Dual problem

\[
\begin{align*}
\max_{\{\alpha_i \geq 0\}} \quad & \min_{\mathbf{w}, \mathbf{w}_0} \mathcal{L}(\mathbf{w}, \mathbf{w}_0, \alpha) \\
\mathcal{L}(\mathbf{w}, \mathbf{w}_0, \alpha) &= \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{i=1}^{N} \alpha_i \left(1 - y^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + \mathbf{w}_0)\right) \\
\n\end{align*}
\]

\[
\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \mathbf{w}_0, \alpha) = 0 \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{N} \alpha_i y^{(i)} \mathbf{x}^{(i)}
\]

\[
\frac{\partial \mathcal{L}(\mathbf{w}, \mathbf{w}_0, \alpha)}{\partial \mathbf{w}_0} = 0 \Rightarrow \quad \sum_{i=1}^{N} \alpha_i y^{(i)} = 0
\]

\[w_0 \text{ do not appear; instead, a “global” constraint on } \alpha \text{ is created.}\]
Hard-margin SVM: Dual problem

\[
\max_{\alpha} \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y^{(i)} y^{(j)} x^{(i)T} x^{(j)} \right\}
\]

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

\[\alpha_i \geq 0 \quad i = 1, ..., N\]

- It is a convex QP
- By solving the above problem first we find \(\alpha\) and then \(w = \sum_{i=1}^{N} \alpha_i y^{(i)} x^{(i)}\)
- \(w_0\) can be set by making the margin equidistant to classes (we discuss it more after defining support vectors).
Karush-Kuhn-Tucker (KKT) conditions

- **Necessary conditions for the solution** $[\mathbf{w}^*, \mathbf{w}_0^*, \alpha^*]$: 
  - $\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \mathbf{w}_0, \alpha)|_{\mathbf{w}^*, \mathbf{w}_0^*, \alpha^*} = 0$
  - $\frac{\partial \mathcal{L}(\mathbf{w}, \mathbf{w}_0, \alpha)}{\partial \mathbf{w}_0}|_{\mathbf{w}^*, \mathbf{w}_0^*, \alpha^*} = 0$
  - $\alpha^*_i \geq 0 \quad i = 1, \ldots, N$
  - $y^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + \mathbf{w}_0^*) \geq 1 \quad i = 1, \ldots, N$
  - $\alpha^*_i \left(1 - y^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + \mathbf{w}_0^*)\right) = 0 \quad i = 1, \ldots, N$

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

- $\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \alpha)|_{\mathbf{x}^*, \alpha^*} = 0$
- $\alpha^*_i \geq 0 \quad i = 1, \ldots, m$
- $g_i(\mathbf{x}^*) \leq 0 \quad i = 1, \ldots, m$
- $\alpha^*_i g_i(\mathbf{x}^*) = 0 \quad i = 1, \ldots, m$
Hard-margin SVM: Support vectors

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

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

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

- **Active** constraint: \( y^{(i)}(w^T x^{(i)} + w_0) = 1 \)
  - \( \Rightarrow \alpha_i \) can be greater than 0 and thus \( x^{(i)} \) can be a support vector.

![Diagram](image)

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

- Support Vectors (SVs) = \{x^{(i)} | \alpha_i > 0\}

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

\[
\mathbf{w} = \sum_{\alpha_i > 0} \alpha_i \ y^{(i)} \mathbf{x}^{(i)}
\]
Hard-margin SVM: Dual problem
Classifying new samples using only SVs

Classification of a new sample $x$:

\[ \hat{y} = \text{sign} \left( w_0 + w^T x \right) \]
\[ \hat{y} = \text{sign} \left( w_0 + \left( \sum_{\alpha_i > 0} \alpha_i y^{(i)} x^{(i)} \right)^T x \right) \]
\[ \hat{y} = \text{sign} (w_0 + \sum_{\alpha_i > 0} \alpha_i y^{(i)} x^{(i)^T} x) \]

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.
How to find $w_0$

- At least one of the $\alpha$’s is strictly positive ($\alpha_s > 0$ for support vector).

- The KKT complementary slackness condition tells us that the constraint corresponding to this non-zero $\alpha_s$ is equality.
  - $w_0$ is found such that the following constraint is satisfied.
    \[
    y_s (w_0 + w^T x_s) = 1
    \]

- We can find $w_0$ using one of the support vectors:
  \[
  \Rightarrow w_0 = y_s - w^T x_s
  \]

  \[
  = y_s - \sum_{\alpha_n > 0} \alpha_n y^{(n)} x^{(n)T} x_s
  \]
Hard-margin SVM: Dual problem

\[
\max_{\alpha} \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y^{(i)} y^{(j)} \mathbf{x}^{(i)T} \mathbf{x}^{(j)} \right\}
\]

Subject to \( \sum_{i=1}^{N} \alpha_i y^{(i)} = 0 \)

\( \alpha_i \geq 0 \quad i = 1, \ldots, N \)

- Only the dot product of each pair of training data appears in the optimization problem
  - This is an important property that is helpful to extend to non-linear SVM (the cost function does not depend explicitly on the dimensionality of the feature space).
Soft-margin SVM: Dual problem

\[
\max_{\alpha} \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y^{(i)} y^{(j)} x^{(i)T} x^{(j)} \right\}
\]

Subject to \( \sum_{i=1}^{N} \alpha_i y^{(i)} = 0 \)

\( 0 \leq \alpha_i \leq C \quad i = 1, \ldots, N \)

- By solving the above quadratic problem first we find \( \alpha \) and then find \( w = \sum_{i=1}^{N} \alpha_i y^{(i)} x^{(i)} \) and \( w_0 \) is computed from SVs.

- For a test sample \( x \) (as before):
  \[
  \hat{y} = \text{sign}(w_0 + w^T x) = \text{sign}(w_0 + \sum_{\alpha_i > 0} \alpha_i y^{(i)} x^{(i)T} x)
  \]
Soft-margin SVM: Support vectors

- **Support Vectors:** $\alpha > 0$
  - If $0 < \alpha < C$: SVs on the margin, $\xi = 0$.
  - If $\alpha = C$: SVs on or over the margin.
Primal vs. dual hard-margin SVM problem

- **Primal problem of hard-margin SVM**
  - $N$ inequality constraints
  - $d + 1$ number of variables

- **Dual problem of hard-margin SVM**
  - one equality constraint
  - $N$ positivity constraints
  - $N$ number of variables (Lagrange multipliers)
  - Objective function more complicated

- The dual problem is helpful and instrumental to use the kernel trick
Primal vs. dual soft-margin SVM problem

- **Primal problem of soft-margin SVM**
  - $N$ inequality constraints
  - $N$ positivity constraints
  - $N + d + 1$ number of variables

- **Dual problem of soft-margin SVM**
  - one equality constraint
  - $2N$ positivity constraints
  - $N$ number of variables (Lagrange multipliers)
  - Objective function more complicated

- The dual problem is helpful and instrumental to use the kernel trick
Not linearly separable data

- Noisy data or overlapping classes
  (we discussed about it: soft margin)
  - Near linearly separable

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

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

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

  $w^T \phi(x) + w_0 = 0$
Soft-margin SVM in a transformed space: 
Primal problem

- Primal problem:

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

- \( \mathbf{w} \in \mathbb{R}^m \): 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:

\[
\max_\alpha \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y^{(i)} y^{(j)} \phi(x^{(i)})^T \phi(x^{(i)}) \right\}
\]

Subject to \( \sum_{i=1}^{N} \alpha_i y^{(i)} = 0 \)

\( 0 \leq \alpha_i \leq C \quad i = 1, \ldots, n \)

- If we have inner products \( \phi(x^{(i)})^T \phi(x^{(j)}) \), only \( \alpha = [\alpha_1, \ldots, \alpha_N] \) needs to be learnt.

- It is not necessary to learn \( m \) parameters as opposed to the primal problem
Kernelized soft-margin SVM

- Optimization problem: \[ k(x, x') = \phi(x)^T \phi(x') \]

\[
\begin{align*}
\max_{\alpha} \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y^{(i)} y^{(j)} k(x^{(i)}, x^{(j)}) \right\} \\
\text{Subject to } \sum_{i=1}^{N} \alpha_i y^{(i)} = 0 \\
0 \leq \alpha_i \leq C \quad i = 1, \ldots, n
\end{align*}
\]

- Classifying a new data:

\[
\hat{y} = \text{sign}(w_0 + w^T \phi(x)) = \text{sign}(w_0 + \sum_{\alpha_i > 0} \alpha_i y^{(i)} \phi(x^{(i)})^T \phi(x))
\]

\[
k(x^{(i)}, x)
\]
Kernel trick

Kernel: the dot product in the mapped feature space \( \mathcal{F} \)

\[ k(x, x') = \phi(x)^T \phi(x') \]

\[ \phi(x) = [\phi_1(x), \ldots, \phi_m(x)]^T \]

\( k(x, x') \) shows the similarity of \( x \) and \( x' \).

The distance can be found as accordingly

Kernel trick → Extension of many well-known algorithms to kernel-based ones

By substituting the dot product with the kernel function
Kernel trick: Idea

- Idea: when the input vectors appear only in the form of dot products, we can use other choices of kernel function instead of inner product
  - Solving the problem without explicitly mapping the data
    - Explicit mapping is expensive if $\phi(x)$ is very high dimensional

- Instead of using a mapping $\phi: X \leftarrow F$ to represent $x \in X$ by $\phi(x) \in F$, a similarity function $k: X \times X \rightarrow \mathbb{R}$ is used.
  - The data set can be represented by $N \times N$ matrix $K$.
  - 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.
Why kernel?

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

- Example: consider the second-order polynomial transform:

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

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

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

$$\phi(x)^T \phi(x') = 1 + (x^T x') + (x^T x')^2 \quad O(d)$$
Polynomial kernel: Efficient kernel function

- An efficient kernel function relies on a carefully constructed \textit{specific} transforms to allow fast computation.

- Example: for polynomial kernels, certain combinations of coefficients that would still make $K$ easy ($\gamma > 0$ and $c > 0$)

$$
\phi(x) = \left[ c \times 1, \sqrt{2c\gamma} \times x_1, \ldots, \sqrt{2c\gamma} \times x_d, \gamma \times x_1^2, \gamma \times x_1 x_2, \ldots, \gamma \right]
$$
Polynomial kernel: Degree two

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

  $\phi(x)$

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

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

- In general, $k(x, z) = (x^T z)^M$ contains all monomials of order $M$.

- This can similarly be generalized to include all terms up to degree $M$, $k(x, z) = (x^T z + c)^M$ ($c > 0$)

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

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

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

  \[ \Rightarrow w_0 + \sum_{\alpha_i > 0} \alpha_i y^{(i)} \left( x^{(i)^T} x + c \right)^M = 0 \quad \text{Boundary is a polynomial of order } M \]
Some common kernel functions

- $K$ is a mathematical and computational shortcut that allows us to combine the transform *and* the inner product into a single more efficient function.

- **Linear**: $k(x, x') = x^T x'$

- **Polynomial**: $k(x, x') = (x^T x' + c)^M \quad c \geq 0$

- **Gaussian**: $k(x, x') = \exp\left(-\frac{||x-x'||^2}{2\sigma^2}\right)$

- **Sigmoid**: $k(x, x') = \tanh(ax^T x' + b)$
Gaussian kernel

- Gaussian: \( k(x, x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right) \)
- Infinite dimensional feature space

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

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

- SVM + Gaussian Kernel can classify any arbitrary training set
  - Training error is zero when \( \sigma \to 0 \)
    - All samples become support vectors (likely overfitting)
SVM: Linear and Gaussian kernels example

Linear kernel: $C = 0.1$

Gaussian kernel: $C = 1, \sigma = 0.25$

some SVs appear to be far from the boundary?

[Zisserman’s slides]
For narrow Gaussian (large $\sigma$), even the protection of a large margin cannot suppress overfitting.
SVM Gaussian kernel: Example

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

Source: Zisserman’s slides
SVM Gaussian kernel: Example

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

$$f(x) = 0$$

$$f(x) = 1$$

Source: Zisserman’s slides
SVM Gaussian kernel: Example

\[ \sigma = 1.0 \quad C = 100 \]

Source: Zisserman’s slides
SVM Gaussian kernel: Example

$\sigma = 1.0 \quad C = 10$

Source: Zisserman’s slides
SVM Gaussian kernel: Example

\[ \sigma = 1.0 \quad C = \infty \]
SVM Gaussian kernel: Example

$\sigma = 0.25 \quad C = \infty$

Source: Zisserman’ s slides
SVM Gaussian kernel: Example

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

Source: Zisserman’s slides
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** $\mathbf{K}_{N \times N}: K_{ij} = k(x^{(i)}, x^{(j)})$  
  
- Restricting the kernel function to a set of points \{\(x^{(1)}, x^{(2)}, \ldots, x^{(N)}\}\)

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

- **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
Construct Valid Kernels

\[
\begin{align*}
    k(x, x') &= c k_1(x, x') \\
    k(x, x') &= f(x) k_1(x, x') f(x') \\
    k(x, x') &= q(k_1(x, x')) \\
    k(x, x') &= \exp(k_1(x, x')) \\
    k(x, x') &= k_1(x, x') + k_2(x, x') \\
    k(x, x') &= k_1(x, x') k_2(x, x') \\
    k(x, x') &= k_3(\phi(x), \phi(x')) \\
    k(x, x') &= x^T A x' \\
    k(x, x') &= k_a(x_a, x'_a) + k_b(x_b, x'_b) \\
    k(x, x') &= k_a(x_a, x'_a) k_b(x_b, x'_b)
\end{align*}
\]

- \( c > 0 \), \( k_1 \): valid kernel
- \( f(\cdot) \): any function
- \( q(\cdot) \): a polynomial with coefficients \( \geq 0 \)
- \( k_1, k_2 \): valid kernels
- \( \phi(x) \): a function from \( x \) to \( \mathbb{R}^M \)
- \( k_3(\cdot, \cdot) \): a valid kernel in \( \mathbb{R}^M \)
- \( A \): a symmetric positive semi-definite matrix
- \( x_a \) and \( x_b \) are variables (not necessarily disjoint) with \( x = (x_a, x_b) \), and \( k_a \) and \( k_b \) are valid kernel functions over their respective spaces.
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

- If a problem can be formulated such that feature vectors only appear in inner products, we can extend it to a kernelized one
  - When we only need to know the dot product of all pairs of data (about the geometry of the data in the feature space)

- By replacing inner products with kernels in linear algorithms, we obtain 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
  \[
  d(\phi(x), \phi(z))^2 = \|\phi(x) - \phi(z)\|^2 = k(x, x) + k(z, z) - 2k(x, z)
  \]
  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
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 that we can use kernel-based version of the 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

\[
\begin{array}{cccccccccc}
A & E & G & A & T & E & A & G & G \\
E & G & T & E & A & G & A & E & G & A & T & G
\end{array}
\]
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.
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
  - Lead us to non-linear SVM method easily by kernel substitution
  - Also, classifier decision in terms of support vectors
- Kernel SVM’s
  - Learns linear decision boundary in a high dimension space without explicitly working on the mapped data