Kernel Methods
Motivation

- Non-linear decision boundary.
- Efficient computation of inner products in high dimension.
- Flexible selection of more complex features.
This Lecture

- Definitions
- SVMs with kernels
- Closure properties
- Sequence Kernels
Non-Linear Separation

- Linear separation impossible in most problems.
- Non-linear mapping from input space to high-dimensional feature space: $\Phi: X \rightarrow F$.
- Generalization ability: independent of $\dim(F)$, depends only on $\rho$ and $m$. 

\[ \Phi: X \rightarrow F. \]
Kernel Methods

**Idea:**
- Define $K : X \times X \to \mathbb{R}$, called kernel, such that:
  \[
  \Phi(x) \cdot \Phi(y) = K(x, y).
  \]
- $K$ often interpreted as a similarity measure.

**Benefits:**
- Efficiency: $K$ is often more efficient to compute than $\Phi$ and the dot product.
- Flexibility: $K$ can be chosen arbitrarily so long as the existence of $\Phi$ is guaranteed (symmetry and positive definiteness condition).
**PDS Condition**

- **Definition:** a kernel $K : X \times X \rightarrow \mathbb{R}$ is **positive definite symmetric (PDS)** if for any $\{x_1, \ldots, x_m\} \subseteq X$, the matrix $K = [K(x_i, x_j)]_{ij} \in \mathbb{R}^{m \times m}$ is symmetric positive semi-definite (SPSD).

- **K SPSD** if symmetric and one of the 2 equiv. cond.’s:
  - its eigenvalues are non-negative.
  - for any $c \in \mathbb{R}^{m \times 1}$, $c^\top K c = \sum_{i,j=1}^{n} c_i c_j K(x_i, x_j) \geq 0$.

- **Terminology:** PDS for kernels, SPSD for kernel matrices (see (Berg et al., 1984)).
Example - Polynomial Kernels

Definition:

\[ \forall x, y \in \mathbb{R}^N, \ K(x, y) = (x \cdot y + c)^d, \quad c > 0. \]

Example: for \( N = 2 \) and \( d = 2 \),

\[ K(x, y) = (x_1 y_1 + x_2 y_2 + c)^2 \]

\[
\begin{bmatrix}
    x_1^2 & x_1 x_2 \\
    x_2^2 & x_2 y_2 \\
\end{bmatrix}
\cdot
\begin{bmatrix}
    y_1^2 & y_1 y_2 \\
    y_2^2 & y_2 y_2 \\
\end{bmatrix}
\]

\[
= \begin{bmatrix}
    \sqrt{2} x_1 x_2 \\
    \sqrt{2c} x_1 \\
    \sqrt{2c} x_2 \\
    c \\
\end{bmatrix}
\cdot
\begin{bmatrix}
    \sqrt{2} y_1 y_2 \\
    \sqrt{2c} y_1 \\
    \sqrt{2c} y_2 \\
    c \\
\end{bmatrix}
\]

XOR Problem

- Use second-degree polynomial kernel with $c = 1$:

\[
\begin{aligned}
&\text{Linearly non-separable} \\
&\text{Linearly separable by} \\
&\quad x_1 x_2 = 0.
\end{aligned}
\]
Other Standard PDS Kernels

- Gaussian kernels:

\[ K(x, y) = \exp \left( - \frac{||x - y||^2}{2\sigma^2} \right), \quad \sigma \neq 0. \]

- Sigmoid Kernels:

\[ K(x, y) = \tanh(a(x \cdot y) + b), \quad a, b \geq 0. \]
This Lecture

- Definitions
- SVMs with kernels
- Closure properties
- Sequence Kernels
Reproducing Kernel Hilbert Space

(Aronszajn, 1950)

Theorem: Let $K : X \times X \to \mathbb{R}$ be a PDS kernel. Then, there exists a Hilbert space $H$ and a mapping $\Phi$ from $X$ to $H$ such that

$$\forall x, y \in X, \ K(x, y) = \Phi(x) \cdot \Phi(y).$$

Furthermore, the following reproducing property holds:

$$\forall f \in H_0, \forall x \in X, \ f(x) = \langle f, \Phi(x) \rangle = \langle f, K(x, \cdot) \rangle.$$
Notes:

• \( H \) is called the reproducing kernel Hilbert space (RKHS) associated to \( K \).

• A Hilbert space such that there exists \( \Phi: X \rightarrow H \) with \( K(x, y) = \Phi(x) \cdot \Phi(y) \) for all \( x, y \in X \) is also called a feature space associated to \( K \). \( \Phi \) is called a feature mapping.

• Feature spaces associated to \( K \) are in general not unique.
Consequence: SVMs with PDS Kernels

(Boser, Guyon, and Vapnik, 1992)

- **Constrained optimization:**

  \[
  \max_\alpha \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j K(x_i, x_j)
  \]

  subject to: \(0 \leq \alpha_i \leq C\) \(\wedge\) \(\sum_{i=1}^{m} \alpha_i y_i = 0, i \in [1, m].\)

- **Solution:**

  \[h(x) = \text{sgn} \left( \sum_{i=1}^{m} \alpha_i y_i K(x_i, x) + b \right), \quad \Phi(x_j) \cdot \Phi(x_i)\]

  with \(b = y_i - \sum_{j=1}^{m} \alpha_j y_j K(x_j, x_i)\) for any \(x_i\) with \(0 < \alpha_i < C\).
**SVMs with PDS Kernels**

- **Constrained optimization:**

  $$\max_\alpha 2 \mathbf{1}^\top \alpha - (\alpha \circ \mathbf{y})^\top \mathbf{K}(\alpha \circ \mathbf{y})$$

  subject to: $$0 \leq \alpha \leq C \land \alpha^\top \mathbf{y} = 0.$$  

- **Solution:**

  $$h = \text{sgn}\left(\sum_{i=1}^{m} \alpha_i \mathbf{y}_i \mathbf{K}(\mathbf{x}_i, \cdot) + b\right),$$

  with $$b = \mathbf{y}_i - (\alpha \circ \mathbf{y})^\top \mathbf{K}\mathbf{e}_i$$ for any $$\mathbf{x}_i$$ with $$0 < \alpha_i < C.$$
Generalization: Representer Theorem

(Kimeldorf and Wahba, 1971)

**Theorem:** Let $K: X \times X \to \mathbb{R}$ be a PDS kernel and $H$ its corresponding RKHS. Then, for any non-decreasing function $G: \mathbb{R} \to \mathbb{R}$ and any $L: \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ the optimization problem

$$\operatorname{argmin}_{h \in H} F(h) = \operatorname{argmin}_{h \in H} G(\|h\|^2_H) + L(h(x_1), \ldots, h(x_m))$$

admits a solution of the form $h^* = \sum_{i=1}^{m} \alpha_i K(x_i, \cdot)$.

If $G$ is further assumed to be increasing, then any solution has this form.
• **Proof:** let $H_1 = \text{span}(\{K(x_i, \cdot): i \in [1, m]\})$. Any $h \in H$ admits the decomposition $h = h_1 + h^\perp$ according to $H = H_1 \oplus H_1^\perp$.

• Since $G$ is non-decreasing,
  \[
  G(\|h_1\|^2) \leq G(\|h_1\|^2 + \|h^\perp\|^2) = G(\|h\|^2).
  \]

• By the reproducing property, for all $i \in [1, m]$,
  \[
  h(x_i) = \langle h, K(x_i, \cdot) \rangle = \langle h_1, K(x_i, \cdot) \rangle = h_1(x_i).
  \]

• Thus, $L(h(x_1), \ldots, h(x_m)) = L(h_1(x_1), \ldots, h_1(x_m))$ and $F(h_1) \leq F(h)$.

• If $G$ is increasing, then $F(h_1) < F(h)$ and any solution of the optimization problem must be in $H_1$. 
Kernel-Based Algorithms

- PDS kernels used to extend a variety of algorithms in classification and other areas:
  - regression.
  - ranking.
  - dimensionality reduction.
  - clustering.

- But, how do we define PDS kernels?
This Lecture

- Definitions
- SVMs with kernels
- Closure properties
- Sequence Kernels
Closure Properties of PDS Kernels

Theorem: Positive definite symmetric (PDS) kernels are closed under:

- sum,
- product,
- tensor product,
- pointwise limit,
- composition with a power series.
### Closure Properties - Proof

**Proof:** closure under *sum*:

\[ c^\top K c \geq 0 \land c^\top K' c \geq 0 \Rightarrow c^\top (K + K') c \geq 0. \]

- closure under *product*: \( K = MM^\top \),

\[
\sum_{i,j=1}^{m} c_i c_j (K_{ij} K'_{ij}) = \sum_{i,j=1}^{m} c_i c_j \left( \sum_{k=1}^{m} M_{ik} M_{jk} K'_{ij} \right) \\
= \sum_{k=1}^{m} \left[ \sum_{i,j=1}^{m} c_i c_j M_{ik} M_{jk} K'_{ij} \right] = \sum_{k=1}^{m} z_k^\top K' z_k \geq 0,
\]

with \( z_k = \begin{bmatrix} c_1 M_{1k} \\ \vdots \\ c_m M_{mk} \end{bmatrix} \).
• Closure under tensor product:
  • definition: for all \( x_1, x_2, y_1, y_2 \in X \),

\[
(K_1 \otimes K_2)(x_1, y_1, x_2, y_2) = K_1(x_1, x_2)K_2(y_1, y_2).
\]

• thus, PDS kernel as product of the kernels

\[
(x_1, y_1, x_2, y_2) \to K_1(x_1, x_2) \quad (x_1, y_1, x_2, y_2) \to K_2(y_1, y_2).
\]

• Closure under pointwise limit: if for all \( x, y \in X \),

\[
\lim_{n \to \infty} K_n(x, y) = K(x, y),
\]

Then, \((\forall n, c^\top K_n c \geq 0) \Rightarrow \lim_{n \to \infty} c^\top K_n c = c^\top K c \geq 0\).
• Closure under **composition with power series**:  
  
  • assumptions: $K$ PDS kernel with $|K(x, y)| < \rho$ for all $x, y \in X$ and $f(x) = \sum_{n=0}^{\infty} a_n x^n$, $a_n \geq 0$ power series with radius of convergence $\rho$.  
  
  • $f \circ K$ is a PDS kernel since $K^n$ is PDS by closure under product, $\sum_{n=0}^{N} a_n K^n$ is PDS by closure under sum, and closure under pointwise limit.  
  
  **Example**: for any PDS kernel $K$, $\exp(K)$ is PDS.
This Lecture

- Definitions
- SVMs with kernels
- Closure properties
- Sequence Kernels
Sequence Kernels

Definition: Kernels defined over pairs of strings.

- Motivation: computational biology, text and speech classification.

- Idea: two sequences are related when they share some common substrings or subsequences.

- Example: sum of the product of the counts of common substrings.
Weighted Transducers

\[ T(x, y) = \text{Sum of the weights of all accepting paths with input } x \text{ and output } y. \]

\[ T(abb, baa) = .1 \times .2 \times .3 \times .1 + .5 \times .3 \times .6 \times .1 \]
Rational Kernels over Strings

(Cortes et al., 2004)

- **Definition**: a kernel $K : \Sigma^* \times \Sigma^* \to \mathbb{R}$ is **rational** if $K = T$ for some weighted transducer $T$.

- **Definition**: let $T_1 : \Sigma^* \times \Delta^* \to \mathbb{R}$ and $T_2 : \Delta^* \times \Omega^* \to \mathbb{R}$ be two weighted transducers. Then, the **composition** of $T_1$ and $T_2$ is defined for all $x \in \Sigma^*$, $y \in \Omega^*$ by

  $$(T_1 \circ T_2)(x, y) = \sum_{z \in \Delta^*} T_1(x, z) \ T_2(z, y).$$

- **Definition**: the **inverse** of a transducer $T : \Sigma^* \times \Delta^* \to \mathbb{R}$ is the transducer $T^{-1} : \Delta^* \times \Sigma^* \to \mathbb{R}$ obtained from $T$ by swapping input and output labels.
Composition

Theorem: the composition of two weighted transducer is also a weighted transducer.

Proof: constructive proof based on composition algorithm.

• states identified with pairs.

• $\epsilon$-free case: transitions defined by

$$E = \bigcup \left\{ \left( (q_1, q'_1), a, c, w_1 \times w_2, (q_2, q'_2) \right) \right\}.$$  

• general case: use of intermediate $\epsilon$-filter.
Composition Algorithm
\(\varepsilon\)-Free Case

Complexity: \(O(\max(|T_1|, |T_2|))\) in general, linear in some cases.
Redundant $\varepsilon$-Paths Problem

(MM et al. 1996)

$$T_1 \quad \tilde{T}_1 \quad T_2 \quad \tilde{T}_2$$

$$T = \tilde{T}_1 \circ F \circ \tilde{T}_2.$$
PDS Rational Kernels
General Construction

**Theorem:** for any weighted transducer \( T : \Sigma^* \times \Sigma^* \to \mathbb{R} \), the function \( K = T \circ T^{-1} \) is a PDS rational kernel.

**Proof:** by definition, for all \( x, y \in \Sigma^* \),

\[
K(x, y) = \sum_{z \in \Delta^*} T(x, z) T(y, z).
\]

- \( K \) is pointwise limit of \( (K_n)_{n \geq 0} \) defined by
  \[
  \forall x, y \in \Sigma^*, \quad K_n(x, y) = \sum_{|z| \leq n} T(x, z) T(y, z).
  \]
- \( K_n \) is PDS since for any sample \((x_1, \ldots, x_m)\),
  \[
  K_n = A A^\top \quad \text{with} \quad A = (K_n(x_i, z_j))_{i \in [1, m], j \in [1, N]}.
  \]
Counting Transducers

X may be a string or an automaton representing a regular expression.

Counts of $Z$ in $X$: sum of the weights of accepting paths of $Z \circ T_X$. 

- $X = ab$
- $Z = bbabaabba$
- $T_X$
- $X: X/1$
- $b: \varepsilon/1$
- $a: \varepsilon/1$
- $0$
- $1/1$
- $\varepsilon \varepsilon a b \varepsilon \varepsilon \varepsilon \varepsilon \varepsilon \varepsilon$
- $\varepsilon \varepsilon \varepsilon \varepsilon \varepsilon ab \varepsilon \varepsilon \varepsilon$
Transducer Counting Bigrams

Counts of $Z$ given by $Z \circ T_{\text{bigram}} \circ ab$. 
Transducer Counting Gappy Bigrams

Counts of \( Z \) given by \( Z \circ T_{\text{gappy bigram}} \circ ab \), gap penalty \( \lambda \in (0, 1) \).
Kernels for Other Discrete Structures

Similarly, PDS kernels can be defined on other discrete structures:

- Images,
- graphs,
- parse trees,
- automata,
- weighted automata.
References


References


Shortest-Distance Problem

Definition: for any regulated weighted transducer $T$, define the shortest distance from state $q$ to $F$ as

$$d(q, F) = \bigoplus_{\pi \in P(q, F)} w[\pi].$$

Problem: compute $d(q, F)$ for all states $q \in Q$.

Algorithms:

- Generalization of Floyd-Warshall.
All-Pairs Shortest-Distance Algorithm

( MM, 2002)

Assumption: closed semiring (not necessarily idempotent).


Properties:

- Time complexity: $\Omega(|Q|^3 (T_\oplus + T_\otimes + T_\star))$.
- Space complexity: $\Omega(|Q|^2)$ with an in-place implementation.
Closed Semirings

(Lehmann, 1977)

- **Definition:** a semiring is closed if the closure is well defined for all elements and if associativity, commutativity, and distributivity apply to countable sums.

- **Examples:**
  - Tropical semiring.
  - Probability semiring when including infinity or when restricted to well-defined closures.
**Pseudocode**

\texttt{Gen-All-Pairs}(G) 
\begin{align*}
1 & \quad \text{for } i \leftarrow 1 \text{ to } |Q| \text{ do} \\
2 & \quad \quad \text{for } j \leftarrow 1 \text{ to } |Q| \text{ do} \\
3 & \quad \quad \quad d[i, j] \leftarrow \bigoplus_{e \in E \cap P(i, j)} w[e] \\
4 & \quad \text{for } k \leftarrow 1 \text{ to } |Q| \text{ do} \\
5 & \quad \quad \text{for } i \leftarrow 1 \text{ to } |Q|, i \neq k \text{ do} \\
6 & \quad \quad \quad \text{for } j \leftarrow 1 \text{ to } |Q|, j \neq k \text{ do} \\
7 & \quad \quad \quad \quad d[i, j] \leftarrow d[i, j] \oplus (d[i, k] \otimes d[k, k]^* \otimes d[k, j]) \\
8 & \quad \text{for } i \leftarrow 1 \text{ to } |Q|, i \neq k \text{ do} \\
9 & \quad \quad d[k, i] \leftarrow d[k, k]^* \otimes d[k, i] \\
10 & \quad \quad d[i, k] \leftarrow d[i, k] \otimes d[k, k]^* \\
11 & \quad d[k, k] \leftarrow d[k, k]^*
\end{align*}
Single-Source Shortest-Distance Algorithm

(anonymous, 2002)

- **Assumption**: $k$-closed semiring.

\[ \forall x \in \mathbb{K}, \bigoplus_{i=0}^{k+1} x^i = \bigoplus_{i=0}^{k} x^i. \]

- **Idea**: generalization of relaxation, but must keep track of weight added to $d[q]$ since the last time $q$ was enqueued.

- **Properties**:
  - works with any queue discipline and any $k$-closed semiring.
  - Classical algorithms are special instances.
Pseudocode

**Generic-Single-Source-Shortest-Distance** \((G, s)\)

1. \textbf{for} \(i \leftarrow 1\) to \(|Q|\)
2. \hspace{1em} \textbf{do} \(d[i] \leftarrow r[i] \leftarrow 0\)
3. \(d[s] \leftarrow r[s] \leftarrow 1\)
4. \(S \leftarrow \{s\}\)
5. \textbf{while} \(S \neq \emptyset\)
6. \hspace{1em} \textbf{do} \(q \leftarrow \text{head}(S)\)
7. \hspace{2em} \textbf{Decqueue}(S)
8. \hspace{1em} \(r' \leftarrow r[q]\)
9. \hspace{1em} \(r[q] \leftarrow 0\)
10. \hspace{1em} \textbf{for} each \(e \in E[q]\)
11. \hspace{2em} \textbf{do if} \(d[n[e]] \neq d[n[e]] \oplus (r' \otimes w[e])\)
12. \hspace{2em} \textbf{then} \(d[n[e]] \leftarrow d[n[e]] \oplus (r' \otimes w[e])\)
13. \hspace{2em} \(r[n[e]] \leftarrow r[n[e]] \oplus (r' \otimes w[e])\)
14. \hspace{2em} \textbf{if} \(n[e] \notin S\)
15. \hspace{3em} \textbf{then DecQueue}(S, n[e])
16. \(d[s] \leftarrow 1\)
Notes

- **Complexity:**
  - depends on queue discipline used.
    \[ O(|Q| + (T_\oplus + T_\otimes + C(A))|E| \max_{q \in Q} N(q) + (C(I) + C(E)) \sum_{q \in Q} N(q)) \]
  - coincides with that of Dijkstra and Bellman-Ford for shortest-first and FIFO orders.
  - linear for acyclic graphs using topological order.
    \[ O(|Q| + (T_\oplus + T_\otimes)|E|) \]

- **Approximation:** \( \epsilon \)-\( k \)-closed semiring, e.g., for graphs in probability semiring.