

Sapienza University of Rome

Master in Artificial Intelligence and Robotics

Machine Learning

A.Y. 2025/2026

Prof. Luca Iocchi

9. Kernel Methods

Luca Iocchi

Summary

- Kernelized linear models
- Kernel functions
- Kernelized SVM - classification
- Kernelized SVM - regression

References

C. Bishop. Pattern Recognition and Machine Learning. Chap. 6, Sect. 7.1

Linear models

Consider a linear model $y(\mathbf{x}; \mathbf{w}) = \mathbf{w}^T \mathbf{x}$ with dataset $D = \{(\mathbf{x}_n, t_n)_{n=1}^N\}$

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (\mathbf{w}^T \mathbf{x}_n - t_n)^2 + \lambda \mathbf{w}^T \mathbf{w}$$

$$E(\mathbf{w}) = (\mathbf{t} - \mathbf{X}\mathbf{w})^T (\mathbf{t} - \mathbf{X}\mathbf{w}) + \lambda \|\mathbf{w}\|^2$$

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix} \text{ design matrix, } \quad \mathbf{t} = \begin{bmatrix} t_1 \\ \vdots \\ t_N \end{bmatrix} \text{ output vector}$$

Solution

Optimal solution by solving $\nabla E(w) = 0$
explicit notation

$$w^* = -\frac{1}{\lambda} \sum_{n=1}^N (w^T x_n - t_n) x_n$$

Let $\alpha_n = -\frac{1}{\lambda}(w^T x_n - t_n)$

Then

$$w^* = \sum_{n=1}^N \alpha_n x_n$$

$$y(x; w^*) = w^{*T} x = \sum_{n=1}^N \alpha_n x_n^T x$$

Solution

Optimal solution by solving $\nabla E(w) = 0$

matrix notation

$$w^* = (X^T X + \lambda I_N)^{-1} X^T t = X^T (X X^T + \lambda I_N)^{-1} t$$

Let $\alpha = (X X^T + \lambda I_N)^{-1} t$

Then

$$w^* = X^T \alpha$$

$$y(x; w^*) = w^{*T} x = \sum_{n=1}^N \alpha_n x_n^T x$$

Linear models

Linear model

$$y(\mathbf{x}; \mathbf{w}^*) = y(\mathbf{x}; \boldsymbol{\alpha}) = \sum_{n=1}^N \alpha_n \mathbf{x}_n^T \mathbf{x}$$

Solution

$$\boldsymbol{\alpha} = (\mathbf{X}\mathbf{X}^T + \lambda I_N)^{-1} \mathbf{t} = (K + \lambda I_N)^{-1} \mathbf{t}$$

Gram matrix

$$K = \mathbf{X}\mathbf{X}^T = \begin{bmatrix} \mathbf{x}_1^T \mathbf{x}_1 & \cdots & \mathbf{x}_1^T \mathbf{x}_N \\ \vdots & \ddots & \vdots \\ \mathbf{x}_N^T \mathbf{x}_1 & \cdots & \mathbf{x}_N^T \mathbf{x}_N \end{bmatrix}$$

Kernel functions

Similarity function

$$k : X \times X \rightarrow \mathbb{R}$$

$k(\mathbf{x}, \mathbf{x}')$ similarity of input samples \mathbf{x} and \mathbf{x}'

Linear kernel

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

Kernelized linear models

Linear model with any kernel k

$$y(\mathbf{x}; \boldsymbol{\alpha}) = \sum_{n=1}^N \alpha_n k(\mathbf{x}_n, \mathbf{x})$$

Solution

$$\boldsymbol{\alpha} = (K + \lambda I_N)^{-1} \mathbf{t}$$

Gram matrix

$$K = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}$$

Kernel trick

Kernel trick or kernel substitution

If input vector \mathbf{x} appears in an algorithm only in the form of an inner product $\mathbf{x}^T \mathbf{x}'$, replace the inner product with some kernel $k(\mathbf{x}, \mathbf{x}')$.

- Can be applied to any \mathbf{x} (even infinite size)
- No need to know $\phi(\mathbf{x})$
- Directly extend many well-known algorithms

Kernels

Extend linear models to non-linear functions.
Allow input with variable length or infinite dimensions?

Examples:

- strings
- trees
- image features
- time-series
- ...

Kernels

Approach:

use a *similarity measure* $k(x, x') \geq 0$ between the instances x, x'
 $k(x, x')$ is a *kernel* function.

Note: If we have $\phi(x)$ a possible choice is $k(x, x') = \phi(x)^T \phi(x')$.

Kernels

Definition

Kernel function: a real-valued function $k(x, x') \in \mathbb{R}$, for $x, x' \in \mathcal{X}$, where \mathcal{X} is some abstract space.

Typically k is:

- symmetric: $k(x, x') = k(x', x)$
- non-negative: $k(x, x') \geq 0$.

Note: Not strictly required!

Input normalization

Input data in the dataset D must be normalized in order for the kernel to be a good *similarity measure* in practice.

Several types of normalizations:

- min-max $\bar{x} = \frac{x - \min}{\max - \min}$
 \min, \max : minimum and maximum input values in D
- normalization (standardization) $\bar{x} = \frac{x - \mu}{\sigma}$
 μ mean and σ standard deviation of input values in D
- unit vector $\bar{x} = \frac{x}{\|x\|}$

In the following, we assume the use of normalized input data.

Kernel families

Linear

$$k(x, x') = x^T x'$$

Polynomial

$$k(x, x') = (\beta x^T x' + \gamma)^d, \quad d \in \{2, 3, \dots\}$$

Radial Basis Function (RBF)

$$k(x, x') = \exp(-\beta |x - x'|^2)$$

Sigmoid

$$k(x, x') = \tanh(\beta x^T x' + \gamma)$$

Kernelized SVM - classification

In SVM, solution has the form:

$$w^* = \sum_{n=1}^N \alpha_n x_n$$

Linear model (with linear kernel)

$$y(x; \alpha) = \text{sign} \left(w_0 + \sum_{n=1}^N \alpha_n x_n^T x \right)$$

Kernel trick

$$y(x; \alpha) = \text{sign} \left(w_0 + \sum_{n=1}^N \alpha_n k(x_n, x) \right)$$

Note: w_0 also estimated from α

Kernelized SVM - classification

Lagrangian problem for kernelized SVM classification

$$\tilde{L}(a) = \sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m k(x_n, x_m)$$

Solution

$$a_n = \dots$$

$$w_0 = \frac{1}{|SV|} \sum_{x_i \in SV} \left(t_i - \sum_{x_j \in S} a_j t_j k(x_i, x_j) \right)$$

Kernelized linear regression

Linear model for regression $y = w^T x$ and data set $D = \{(x_n, t_n)\}_{n=1}^N\}$

Minimize the regularized loss function

$$E(w) = \sum_{n=1}^N E(y_n, t_n) + \lambda \|w\|^2,$$

where $y_n = w^T x_n$.

Kernelized linear regression

Apply the kernel trick:

$$y(\mathbf{x}; \mathbf{w}^*) = \sum_{n=1}^N \alpha_n k(\mathbf{x}_n, \mathbf{x})$$

$$\boldsymbol{\alpha} = (\mathbf{K} + \lambda \mathbf{I}_N)^{-1} \mathbf{t}$$

Issue: computation of \mathbf{K} requires $> N^2$ operations and \mathbf{K} is not sparse.

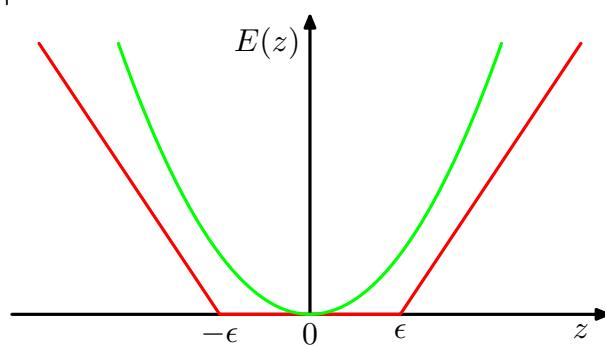
Kernelized SVM - regression

Consider

$$E(\mathbf{w}) = C \sum_{n=1}^N E_\epsilon(y_n, t_n) + \frac{1}{2} \|\mathbf{w}\|^2$$

with C inverse of λ and E_ϵ an ϵ -insensitive error function:

$$E_\epsilon(y, t) = \begin{cases} 0 & \text{if } |y - t| < \epsilon \\ |y - t| - \epsilon & \text{otherwise} \end{cases}.$$



Not differentiable \rightarrow difficult to solve.

Kernelized SVM - regression

Introduce *slack variables* $\xi_n^+, \xi_n^- \geq 0$:

$$\begin{aligned} t_n &\leq y_n + \epsilon + \xi_n^+ \\ t_n &\geq y_n - \epsilon - \xi_n^- \end{aligned}$$

Points inside the ϵ -tube $y_n - \epsilon \leq t_n \leq y_n + \epsilon \Rightarrow \xi_n = 0$

$$\xi_n^+ > 0 \Rightarrow t_n > y_n + \epsilon$$

$$\xi_n^- > 0 \Rightarrow t_n < y_n - \epsilon$$

with $y_n = y(x_n; w)$

Kernelized SVM - regression

Loss function can be rewritten as:

$$E(w) = C \sum_{n=1}^N (\xi_n^+ + \xi_n^-) + \frac{1}{2} \|w\|^2,$$

subject to the constraints:

$$\begin{aligned} t_n &\leq y(x_n; w) + \epsilon + \xi_n^+ \\ t_n &\geq y(x_n; w) - \epsilon - \xi_n^- \\ \xi_n^+ &\geq 0 \\ \xi_n^- &\geq 0 \end{aligned}$$

This is a standard quadratic program (QP), can be “easily” solved.

Kernelized SVM - regression

Lagrangian problem

$$\tilde{L}(a, a') = \dots \sum_{n=1}^N \sum_{m=1}^N a_n a_m \dots k(x_n, x_m) \dots$$

from which we compute \hat{a}_n , \hat{a}'_m (sparse values, most of them are zero) and

$$\hat{w}_0 = t_n - \epsilon - \sum_{m=1}^N (\hat{a}_m - \hat{a}'_m) k(x_n, x_m)$$

for some data point n such that $0 < a_n < C$

Prediction

$$y(x) = \sum_{n=1}^N (\hat{a}_n - \hat{a}'_n) k(x, x_n) + \hat{w}_0$$

Kernelized SVM - regression

From Karush-Kuhn-Tucker (KKT) condition (see Bishop Sect. 7.1.4)

Support vectors contribute to predictions

$$\hat{a}_n > 0 \Rightarrow \epsilon + \xi_n + y_n - t_n = 0$$

data point lies on or above upper boundary of the ϵ -tube

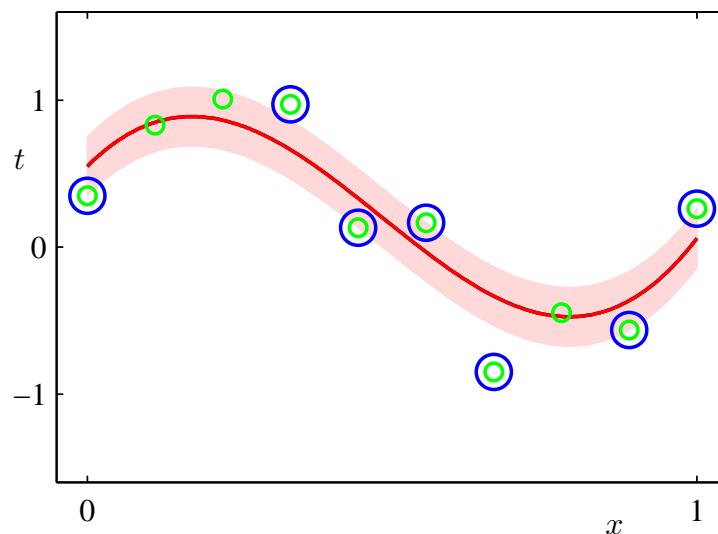
$$\hat{a}'_n > 0 \Rightarrow \epsilon + \xi_n - y_n + t_n = 0$$

data point lies on or below lower boundary of the ϵ -tube

All other data points inside the ϵ -tube have $\hat{a}_n = 0$ and $\hat{a}'_n = 0$ and thus do not contribute to prediction.

Kernelized SVM - regression

Example: support vectors and ϵ insensitive tube



Summary

- Kernel methods overcome difficulties in defining non-linear models
- Kernelized SVM is one of the most effective ML method for classification and regression
- Still requires model selection and hyper-parameters tuning