University of Rome "La Sapienza"

Master in Artificial Intelligence and Robotics

Machine Learning

A.Y. 2018/2019

Prof. Luca Iocchi

Sapienza University of Rome, Italy
Master in Artificial Intelligence and Robotics
Machine Learning (2018/19)

12. Kernels Methods

Luca Iocchi

Summary

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

References
C. Bishop. Pattern Recognition and Machine Learning. Sect. 7.1

Sapienza University of Rome, Italy - Machine Learning (2018/2019)

Kernels

So far:
Objects represented as fixed-length feature-vectors $\mathbf{x} \in \mathbb{R}^M$ or $\phi(\mathbf{x})$.

Issue:
what about objects with variable length or infinite dimensions?

Examples:

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

Kernels

Approach:

use a *similarity measure* $k(\mathbf{x}, \mathbf{x}') \geq 0$ between the objects $\mathbf{x}, \mathbf{x}'$.
$k(\mathbf{x}, \mathbf{x}')$ is called a kernel function.

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

Kernels

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

Typically k is:

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

Note: Not strictly required!

Kernel examples

Linear

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

Polynomial

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

Radila Basis Function (RBF)

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

Sigmoid

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

Kernelized linear models

Consider a linear model $y(\mathbf{x}; \mathbf{w}) = \mathbf{w}^T \mathbf{x}$

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

$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix}$ is the design matrix.

Optimal solution

$\hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X} + \lambda I_M)^{-1} \mathbf{X}^T \mathbf{t} = \mathbf{X}^T (\mathbf{X}\mathbf{X}^T + \lambda I_N)^{-1} \mathbf{t}$,
with I_N the $N \times N$ identity matrix.

Kernelized linear models

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

then $\hat{\mathbf{w}} = \mathbf{X}^T\boldsymbol{\alpha} = \sum_{i=1}^{N} \alpha_i \mathbf{x}_i$.

Hence we have $y(\mathbf{x}; \hat{\mathbf{w}}) = \hat{\mathbf{w}}^T\mathbf{x} = \sum_{i=1}^{N} \alpha_i \mathbf{x}_i^T \mathbf{x}$.

If we consider a linear kernel $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T\mathbf{x}'$, we can rewrite the model as $y(\mathbf{x}; \hat{\mathbf{w}}) = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x})$

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

Sapienza University of Rome, Italy - Machine Learning (2018/2019)

Kernelized linear models

Linear model with any kernel k

$$y(\mathbf{x}; \hat{\mathbf{w}}) = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x})$$

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, replace the inner product with some kernel.

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

Kernelized SVM - classification

Solution has the form:

$$\hat{\mathbf{w}} = \sum_{i=1}^{N} \alpha_i \mathbf{x}_i$$

Linear model (with linear kernel)

$$y(\mathbf{x}, \hat{\mathbf{w}}) = \mathrm{sign}(w_0 + \sum_{i=1}^{N} \alpha_i \mathbf{x}_i^T \mathbf{x})$$

Kernel trick

$$y(\mathbf{x}, \hat{\mathbf{w}}) = \mathrm{sign}\left(w_0 + \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x})\right)$$

Kernelized SVM - classification

Lagrangian problem for kernelized SVM classification

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

Solution

$$a_i = \dots$$

$$w_0 = \frac{1}{|SV|} \sum_{\mathbf{x}_k \in SV} \left(t_k - \sum_{\mathbf{x}_j \in S} a_j t_j k(\mathbf{x}_k, \mathbf{x}_j) \right)$$

SVM - regression

Linear model for regression $y = \mathbf{w}^T \mathbf{x}$ and data set $D = \{(\mathbf{x}_i, t_i)_{i=1}^N\}$

Minimize the regularized loss function

$$J(\mathbf{w}) = \sum_{i=1}^{N} E(y_i, t_i) + \lambda \|\mathbf{w}\|^2,$$

where $y_i = \mathbf{w}^T \mathbf{x}_i$.

Note: if $E(y_i, t_i) = (y_i - t_i)^2$ it is equivalent to regularized linear regression.

SVM - regression

Solution

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X} + \lambda I_M)^{-1}\mathbf{X}^T\mathbf{t} = \mathbf{X}^T\boldsymbol{\alpha}$$

Predictions are made using:

$$y(\mathbf{x}; \hat{\mathbf{w}}) = \sum_{i=1}^{N} \hat{\alpha}_i \mathbf{x}_i^T \mathbf{x}.$$

Kernelized SVM - regression

Apply the kernel trick:

$$y(\mathbf{x}; \hat{\mathbf{w}}) = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x}), \text{ with } \boldsymbol{\alpha}{=}(K + \lambda I_N)^{-1}\mathbf{t}$$

Due to K all data points are involved and $\boldsymbol{\alpha}$ is not sparse.

Kernelized SVM - regression

Consider

$$J(\mathbf{w}) = C \sum_{i=1}^{N} E_\epsilon(y_i, t_i) + \frac{1}{2}\|\mathbf{w}\|^2,$$

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

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

Not differentiable $\rightarrow$ difficult to solve.

Kernelized SVM - regression

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

$$t_i \leq y_i + \epsilon + \xi_i^+$$
$$t_i \geq y_i - \epsilon - \xi_i^-$$

Points inside the ϵ-tube $y_i - \epsilon \leq t_i \leq y_i + \epsilon \Rightarrow \xi_i = 0$

$\xi_i^+ > 0 \Rightarrow t_i > y_i + \epsilon$

$\xi_i^- > 0 \Rightarrow t_i < y_i - \epsilon$

with $y_i = y(\mathbf{x}_i; \mathbf{w})$

Kernelized SVM - regression

Loss function can be rewritten as:

$$J(\mathbf{w}) = C \sum_{i=1}^{N} (\xi_i^+ + \xi_i^-) + \frac{1}{2} \|\mathbf{w}\|^2,$$

subject to the constraints:

$$\begin{aligned} t_i &\leq y(\mathbf{x}_i, \mathbf{w}) + \epsilon + \xi_i^+ \\ t_i &\geq y(\mathbf{x}_i, \mathbf{w}) - \epsilon - \xi_i^- \\ \xi_i^+ &\geq 0 \\ \xi_i^- &\geq 0 \end{aligned}$$

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

Kernelized SVM - regression

Lagrangian problem

$$\tilde{L}(\mathbf{a}, \mathbf{a}') = \ldots \sum_{n=1}^{N} \sum_{m=1}^{N} \ldots k(\mathbf{x}_n, \mathbf{x}_m) \ldots$$

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

Prediction

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

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

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

Kernelized SVM - regression

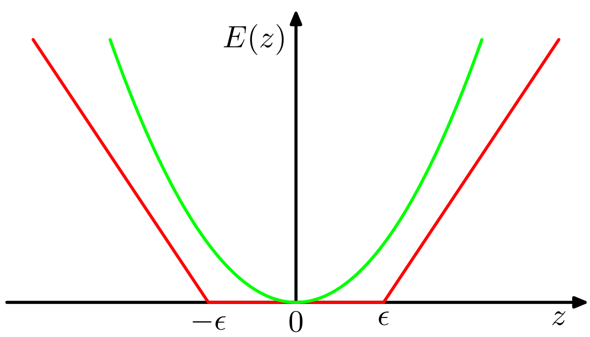
Support vectors contribute to predictions

$\hat{a}_n > 0 \Rightarrow \epsilon + \xi_n + y_n - t_n = 0$
data point lies on or above upper boundary of the ϵ-tube

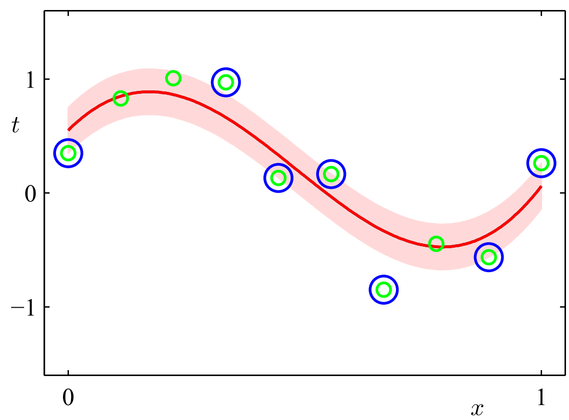
$\hat{a}'_n > 0 \Rightarrow \epsilon + \xi_n - y_n + t_n = 0$
data point lies on or below lower boundary of the ϵ-tube

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



ϵ insensitive error function (red)



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