Sapienza University of Rome

Master in Artificial Intelligence and Robotics
Master in Engineering in Computer Science

Machine Learning

A.Y. 2020/2021

Prof. L. Iocchi, F. Patrizi

9. Kernel Methods

L. Iocchi, F. Patrizi

Summary

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

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

Kernels

they are extensions of the basis functions and they are useful to even increase the applicability of methods

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

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 instances $\mathbf{x}, \mathbf{x}'$

$k(\mathbf{x}, \mathbf{x}')$ is called a *kernel* function.

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

for models which are based on a fixed nonlinear feature space mapping φ(x), the kernel function is given by this relation

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:

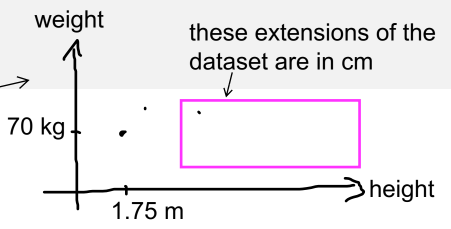
- symmetric: $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$
- non-negative: $k(\mathbf{x}, \mathbf{x}') \geq 0$.

Note: Not strictly required!

important is the normalization of data

Input normalization

for ex. if we consider a sample charachterized by the height (in meters) and weight (in kg) of people, and we extend the dataset with values of the height in cm, from the point of view of the representation of the problem nothing changes, but the solution will change



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 $\boxed{\bar{x} = \frac{x - min}{max - min}}$ the min is 0 and the max is 1
  min, max: minimum and maximum input values in D
- normalization (standardization) $\boxed{\bar{x} = \frac{x - \mu}{\sigma}}$ after this standardization the dataset will have mean=0 and standard deviation=1
  μ mean and σ standard deviation of input values in D
- unit vector $\boxed{\bar{x} = \frac{x}{||x||}}$

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

Kernel families

Linear

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

Polynomial

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

Radial Basis Function (RBF)

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

Sigmoid

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

how do we use this kernel?

Kernelized linear models

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

we define an error function for ex. a squared error function + regularization term

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

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

Optimal solution

we call it alpha

α

$$\hat{\mathbf{w}} = (X^TX + \lambda I_N)^{-1}X^T\mathbf{t} = X^T\boxed{(XX^T + \lambda I_N)^{-1}\mathbf{t}},$$

with I_N the $N \times N$ identity matrix.

identity matrix NxN

Kernelized linear models

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

linear cobination of the samples in dataset x_n multiplied by coeff Alpha_n

then $\hat{\mathbf{w}} = X^T\boldsymbol{\alpha} = \sum_{n=1}^N \alpha_n \mathbf{x}_n$.

Hence we have $y(\mathbf{x};\hat{\mathbf{w}}) = \hat{\mathbf{w}}^T\mathbf{x} = \sum_{n=1}^N \alpha_n \mathbf{x}_n^T\mathbf{x}$.

x is used to indicate a general sample
while x_n is a specific sample

If we consider a linear kernel $k(\mathbf{x},\mathbf{x}') = \mathbf{x}^T\mathbf{x}'$, we can rewrite the model as

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

with

$\boldsymbol{\alpha} = (K + \lambda I_N)^{-1}\mathbf{t}$, and $K = XX^T$ **Gram matrix** summarize

Kernelized linear models

with linear kernel we will represent a linear function

Linear model with linear kernel $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T\mathbf{x}'$

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

k(x_n, x)

Solution

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

Gram matrix

should be positive semidefinite for all possible choices of the set {xn}

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

k(x_i, x_j)

cross product of all the possible pairs in the dataset

Kernelized linear models

written with k

Linear model with any kernel k

so we have this formulation both for linear kernel and for any kernel

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

Solution

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

Gram matrix

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

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

we explored a variety of learning algorithms based on nonlinear kernels. One of the significant limitations of many such algorithms is that the kernel function k(xn, xm) must be evaluated for all possible pairs xn and xm of training points, which can be computationally infeasible during training and can lead to excessive computation times when making predictions for new data points.

now we examine the kernel-based algorithms that have sparse solutions, so that predictions for new inputs depend only on the kernel function evaluated at a subset of the training data points.

An important property of support vector machines is that the determination of the model parameters corresponds to a convex optimization problem, and so any local solution is also a global optimum.

solving problems in classification, regression, and novelty detection.

Kernelized SVM - classification

In SVM, solution has the form:

$$\hat{\mathbf{w}} = \sum_{n=1}^{N} \alpha_n \mathbf{x}_n$$

$\boxed{y(x) = w^T \phi(x) + b}$

Linear model (with linear kernel)

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

Kernel trick

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

Note: w_0 also estimated from $\boldsymbol{\alpha}$

Kernelized SVM - classification

we have something like this:

x_n^T x_m

and we replace it with

Lagrangian problem for kernelized SVM classification

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

when the kernel is linear
we have the same result

Solution

$$a_n = \ldots$$

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

Kernelized linear regression

we have

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

Minimize the regularized loss function

error that measures how much the prediction is
distance from the value that we have in the dataset

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

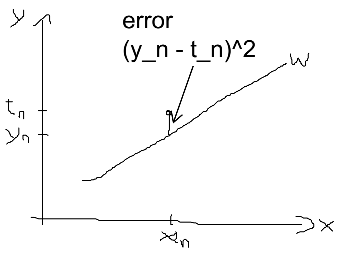
regularization term to control and avoid overfitting

where $y_n = \mathbf{w}^T \mathbf{x}_n$.

Kernelized linear regression

Consider $E(y_n, t_n) = (y_n - t_n)^2$: i.e., regularized linear regression.

Solution



$$\boxed{\begin{aligned}\hat{\mathbf{w}} &= (\mathbf{X}^T\mathbf{X} + \lambda I_N)^{-1}\mathbf{X}^T\mathbf{t} = \mathbf{X}^T\boldsymbol{\alpha} \\ \boldsymbol{\alpha} &= (\mathbf{X}\mathbf{X}^T + \lambda I_N)^{-1}\mathbf{t}\end{aligned}}$$

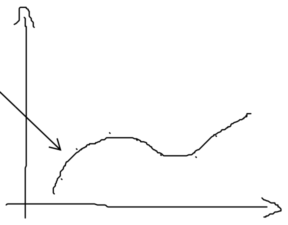
Predictions are made using:

$$y(\mathbf{x}; \hat{\mathbf{w}}) = \sum_{n=1}^{N} \alpha_n \boxed{\mathbf{x}_n^T \mathbf{x}.}$$

we can apply the kernel trick and so we have the kernelized model

Kernelized linear regression

for ex. if we apply a polynomial kernel (cubic with 3 degrees)



Apply the kernel trick:

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

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

Issue: computation of K requires $|D|^2$ operations and K is not sparse.

this method may not be practical when the dataset is large

Kernelized SVM - regression

Consider

we have a different error function, it is not a quadratic function but it is called insensitive error function

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

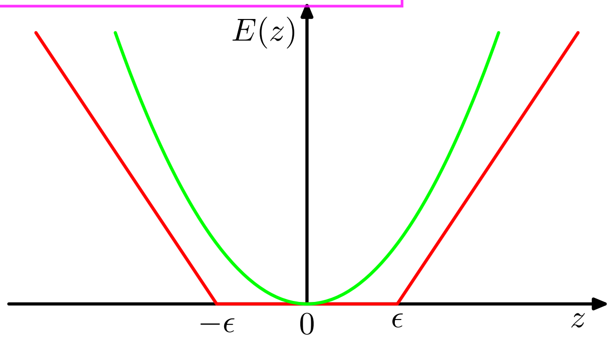
inverse of normalization term

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

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

if the distance between t and y is less then epsilon

error is proportional to the distance from the epsilon boundary



Plot of an epsilon-insensitive error function (in red) in which the error increases linearly with distance beyond the insensitive region. Also shown for comparison is the quadratic error function (in green).

z = y_n - t_n

Not differentiable $\rightarrow$ difficult to solve.

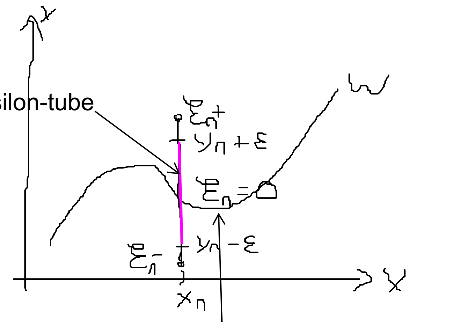
Kernelized SVM - regression

we can re-express the optimization problem by introducing slack variables.

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

$$\boxed{\begin{array}{l} t_n \leq y_n + \epsilon + \xi_n^+ \\ t_n \geq y_n - \epsilon - \xi_n^- \end{array}}$$

epsilon-tube



if t_n is inside the epsilon-tube

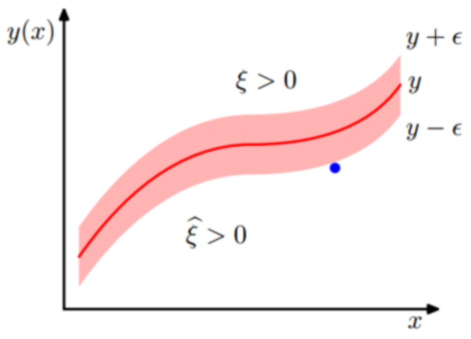
Points inside the ϵ-tube $y_n - \epsilon \leq t_n \leq y_n + \epsilon \Rightarrow \xi_n = 0$

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

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

with $y_n = y(\mathbf{x}_n; \mathbf{w})$

Illustration of SVM regression, showing the regression curve together with the ϵ-insensitive 'tube'. Also shown are examples of the slack variables ξ and $\widehat{\xi}$. Points above the ϵ-tube have $\xi > 0$ and $\widehat{\xi} = 0$, points below the ϵ-tube have $\xi = 0$ and $\widehat{\xi} > 0$, and points inside the ϵ-tube have $\xi = \widehat{\xi} = 0$.



Kernelized SVM - regression

epsilon is up to we, we need to choose a good value for it

Loss function can be rewritten as:

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

subject to the constraints:

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

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

Kernelized SVM - regression

Lagrangian problem

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

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

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

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

Prediction

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

Kernelized SVM - regression

one of the best solution to solve ML problems

From Karush-Kuhn-Tucker (KKT) condition (see Bishop Sect. 7.1.4)
Support vectors contribute to predictions

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

for all the points inside the epsilon-tube that not contribute to the solution, the lagrangian is zero. For these points we don't need to compute the kernel

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

for the SVM in classification, SV are points on the maximum margin
for the SVM in regrassion, SV are points outside the epsilon-tube

in both casses only SVs contribute to the solution but they are different in regression and classification

Example: support vectors and ϵ insensitive tube

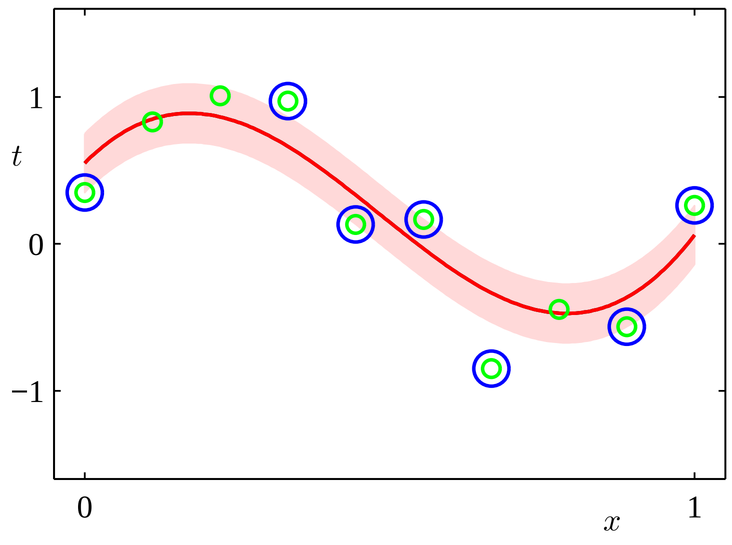


Illustration of the v-SVM for regression applied to the sinusoidal synthetic data set using Gaussian kernels. The predicted regression curve is shown by the red line, and the -insensitive tube corresponds to the shaded region. Also, the data points are shown in green, and those with support vectors are indicated by blue circles.

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