Kernel Methods

TTT4185 Machine Learning for Signal Processing

Giampiero Salvi

Department of Electronic Systems
NTNU

HT2021

Outline

1. Kernel Methods
   - Memory-Based Methods
   - Dual Representation
   - Constructing Kernels
   - Gaussian Processes

Outline

1. Kernel Methods
   - Memory-Based Methods
   - Dual Representation
   - Constructing Kernels
   - Gaussian Processes

Parametric Models

1. define a parametric model, for example

$$
\begin{aligned}
y(\mathbf{x}, \mathbf{w}) &= \mathbf{w}^T \mathbf{x} \\
y(\mathbf{x}, \mathbf{w}) &= w_0 + w_1 x + w_2 x^2 \dots \\
y(\mathbf{x}, \mathbf{w}) &= \mathbf{w}^T \phi(\mathbf{x}) \\
p(\mathbf{x}, \mathcal{C}_k) &= \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \\
&\dots
\end{aligned}
$$

2. fit model parameters to data $\mathcal{D}$ (ML, MAP)
3. throw away the data and use the model for predictions

Special Case: Full Bayesian Methods

$$p(t|\mathcal{D}) = \int_\theta p(t|\theta, \mathcal{D}) p(\theta|\mathcal{D}) d\theta$$

- marginalize out the value of the parameters
- bust still define an underlying parametric model

Memory-Based Methods

- use the training data $\mathcal{D}$ directly to make predictions
- example: k-nearest neighbours
- we need a metric (e.g. Euclidean distance) to determine similarity

Dual Representation: Kernel

- many linear parametric models can be recast
- representation of observations $\mathbf{x}_n$ is not important singularly
- what matters is pair-wise relationship between points $k(x, x')$ (kernel)
- as long as we can compute $k(.,.)$ the dimensionality of the input does not matter

Example: Linear Regression

Linear model:

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}) \quad \text{with} \quad \boldsymbol{\phi}(\mathbf{x}) = \left[\phi_1(\mathbf{x}) \quad \dots \quad \phi_M(\mathbf{x})\right]^T$$

MAP estimation with zero-mean isotopic prior:

$$J(\mathbf{w}) = \underbrace{\frac{1}{2} \sum_{n=1}^{N} \left\{\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n) - t_n\right\}^2}_{\text{least squares}} + \underbrace{\frac{\lambda}{2} \mathbf{w}^T \mathbf{w}}_{\text{regularization}}$$

Differentiating:

$$\nabla J(\mathbf{w}) = 0 \quad \Rightarrow \quad \mathbf{w} = -\frac{1}{\lambda} \sum_{n=1}^{N} \left\{\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n) - t_n\right\} \boldsymbol{\phi}(\mathbf{x}_n)$$

Linear Prediction: Dual representation

$$\mathbf{w} = -\frac{1}{\lambda}\sum_{n=1}^{N}\left\{\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n) - t_n\right\}\boldsymbol{\phi}(\mathbf{x}_n) = \sum_{n=1}^{N} a_n\boldsymbol{\phi}(\mathbf{x}_n) = \boldsymbol{\Phi}^T\mathbf{a}$$

Where we have defined:

$$\mathbf{a} = \begin{bmatrix} a_1, & \dots, & a_N \end{bmatrix}^T, \quad a_n = -\frac{1}{\lambda}\left\{\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n) - t_n\right\}$$

$$\boldsymbol{\Phi} = \begin{bmatrix} \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & \dots & \phi_M(\mathbf{x}_1) \\ \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & \dots & \phi_M(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(\mathbf{x}_N) & \phi_2(\mathbf{x}_N) & \dots & \phi_M(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\phi}(\mathbf{x}_1)^T \\ \boldsymbol{\phi}(\mathbf{x}_2)^T \\ \vdots \\ \boldsymbol{\phi}(\mathbf{x}_N)^T \end{bmatrix} \quad \text{design matrix}$$

Linear Prediction: Dual representation Properties

Parameter space:

$$\mathbf{w} = \begin{bmatrix} w_1, & \dots, & w_M \end{bmatrix}^T \in \mathbb{R}^M \quad \text{same dimensionality as the features}$$

Dual space:

$$\mathbf{a} = \begin{bmatrix} a_1, & \dots, & a_N \end{bmatrix}^T \in \mathbb{R}^N \quad \text{number of data points}$$

$$a_n = -\frac{1}{\lambda}\{\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n) - t_n\} \quad \text{is the prediction error for the } n\text{th data point}$$

Regularized sum of squares: Dual representation

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^T\mathbf{\Phi}\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{\Phi}^T\mathbf{a} - \mathbf{a}^T\mathbf{\Phi}\mathbf{\Phi}\mathbf{t} + \frac{1}{2}\mathbf{t}^T\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^T\mathbf{\Phi}\mathbf{\Phi}^T\mathbf{a}$$

Gram matrix

$$\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}^T \quad \text{that is } K_{nm} = \phi(\mathbf{x}_n)^T\phi(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m)$$

Then

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^T\mathbf{K}\mathbf{K}\mathbf{a} - \mathbf{a}^T\mathbf{K}\mathbf{t} + \frac{1}{2}\mathbf{t}^T\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^T\mathbf{K}\mathbf{a}$$

Solving for $\nabla J(\mathbf{a}) = 0$:

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

Linear prediction model: Dual representation

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \phi(\mathbf{x}) = \mathbf{a}^T \mathbf{\Phi} \phi(\mathbf{x}) = k(\mathbf{x})^T (\mathbf{K} + \lambda \mathbf{I}_N)^{-1} \mathbf{t}$$

where

$$k(\mathbf{x})^T = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}) & k(\mathbf{x}_2, \mathbf{x}) & \dots & k(\mathbf{x}_N, \mathbf{x}) \end{bmatrix}$$

- to express $y(\mathbf{x})$ in $\mathbf{w}$ we needed to invert an $M \times M$ matrix
- now we need to invert an $N \times N$ matrix
- usually $N >> M$!

Dual representation: advantages

- no need to define $\phi(\mathbf{x})$ explicitly
- it can work even with very high dimensional features (even ∞)
- no need to use all the N training points (support vector machines)

Constructing Kernels

Indirect approach (use basis functions

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}') = \sum_{i=1}^{M} \phi_i(\mathbf{x}) \phi_i(x')$$

Direct approach:

- use scalar product in some space

Example: Quadratic Kernel

Direct approach: use scalar product in some space

Example: $\mathbf{x}, \mathbf{z} \in \mathbb{R}^2$,

$$
\begin{aligned}
k(\mathbf{x}, \mathbf{z}) &= (\mathbf{x}^T\mathbf{z})^2 = (x_1 z_1 + x_2 z_2)^2 = \\
&= x_1^2 z_1^2 + 2 x_1 z_1 x_2 z_2 + x_2^2 z_2^2 = \\
&= (x_1^2, \sqrt{2} x_1 x_2, x_2^2)(z_1^2, \sqrt{2} z_1 z_2, z_2^2)^T = \\
&= \phi(x)^T \phi(z)
\end{aligned}
$$

therefore:

$$
\phi(x) = (x_1^2,\ \sqrt{2} x_1 x_2,\ x_2^2)^T \in \mathbb{R}^3
$$

Quadratic Kernel Remarks

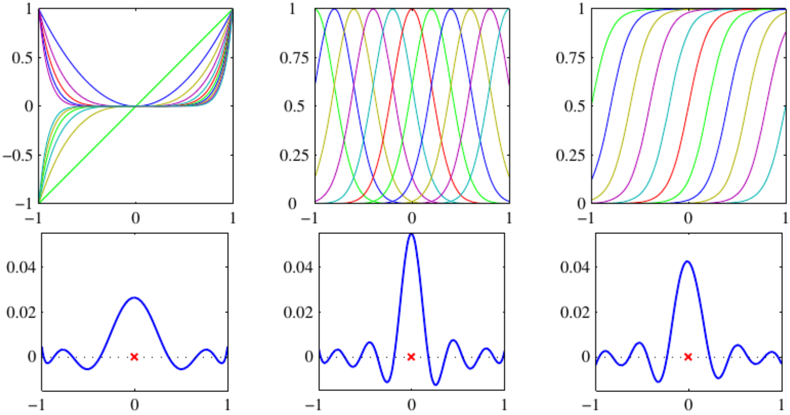
- input space is 2 dimensional ($\mathbf{x} \in \mathbb{R}^2$)
- feature space is 3 dimensional ($\boldsymbol{\phi}(\mathbf{x}) \in \mathbb{R}^3$)
- all we need for inference is $k(\mathbf{x}, \mathbf{z}) \quad \Rightarrow$ we still work in 2 dimensions
- ...but have the advantages of 3 dimensions
- more in general for $\mathbf{x} \in \mathbb{R}^D$, the features are $\frac{D(D+1)}{2}$ dimensional

Practical Example

- We need to do regression based on 1000 training images
- each image is 32×32 pixels (1024 dimensions)
- the quadratic kernel gives us features in 524.800 dimensions
- but we only need to invert a 1000×1000 matrix.

Basis Functions and Kernels

One dimensional $x \in \mathbb{R}$, kernel $k(x, x')$ for $x' = 0$



polynomial Gaussian sigmoid

Kernels as Building Blocks

If $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$ are valid kernels, the following are also valid:

$k(\mathbf{x}, \mathbf{x}') = c\ k_1(\mathbf{x}, \mathbf{x}')$	scaled version, $c > 0$
$k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x}) k_1(\mathbf{x}, \mathbf{x}') f(\mathbf{x}')$	any function $f(.)$
$k(\mathbf{x}, \mathbf{x}') = q\big(k_1(\mathbf{x}, \mathbf{x}')\big)$	$q(.)$ polynomial n.n. coeff.
$k(\mathbf{x}, \mathbf{x}') = \exp(k_1(\mathbf{x}, \mathbf{x}'))$	exponential
$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}')$	sum
$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') k_2(\mathbf{x}, \mathbf{x}')$	product
$k(\mathbf{x}, \mathbf{x}') = k_1(\boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}'))$	change of basis
$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{A} \mathbf{x}'$	$\mathbf{A}$ symm. positive semidef.
$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a) + k_b(\mathbf{x}_b, \mathbf{x}'_b)$	subspace sum
$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a) k_b(\mathbf{x}_b, \mathbf{x}'_b)$	subspace product

Polynomial Kernels

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}')^2$$ only square terms

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}' + c)^2, c > 0$$ also linear terms

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

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}' + c)^M, c > 0$$ also linear terms

Gaussian Kernel

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

- not interpreted as probability distribution
- check validity

$$||\mathbf{x} - \mathbf{x}'||^2 = \mathbf{x}^T\mathbf{x} + (\mathbf{x}')^T\mathbf{x}' - 2\mathbf{x}^T\mathbf{x}'$$

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(\frac{\mathbf{x}^T\mathbf{x}}{2\sigma^2}\right)\exp\left(\frac{(\mathbf{x}')^T\mathbf{x}'}{2\sigma^2}\right)\exp\left(\frac{-2\mathbf{x}^T\mathbf{x}'}{2\sigma^2}\right)$$

- features $\phi(\mathbf{x})$ associated with Gaussian kernel have infinite dimensionality!!

Generality of Kernels

- we are not interested in the representation for $\mathbf{x}$
- we can define kernels on discrete objects (non-vectorial spaces)

Example 1: A_1 and A_2 are subsets of a finite set

$$k(A_1, A_2) = 2^{|A_1 \cap A_2|}$$

Example 2: strings (variable length sequences of discrete symbols)

- text classification, spam filters
- gene analysis

Kernels over Generative Models

Given a generative model $p(\mathbf{x})$, define kernel as

$$k(\mathbf{x}, \mathbf{x}') = p(\mathbf{x})p(\mathbf{x}') \qquad \text{simple distribution}$$

$$k(\mathbf{x}, \mathbf{x}') = \sum_i p(\mathbf{x}|i)p(\mathbf{x}'|i)p(i) \qquad \text{mixture of distributions}$$

- combining discriminative and generative ideas

Example (Hidden Markov Model)

$$\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_L\} \qquad \text{observation sequence}$$

$$\mathbf{Z} = \{\mathbf{z}_1, \dots, \mathbf{z}_L\} \qquad \text{latent variable}$$

$$k(\mathbf{X}, \mathbf{X}') = \sum_{\mathbf{Z}} p(\mathbf{X}|\mathbf{Z}) p(\mathbf{X}'|\mathbf{Z}) p(\mathbf{Z}) \qquad \text{sequences of equal lenght}$$

- can be extended to sequences of variable length

Gaussian Processes

Note: this is big topic, we only mention it here

Standard regression model:

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$$ M basis functions$\phi_i(\mathbf{x})$

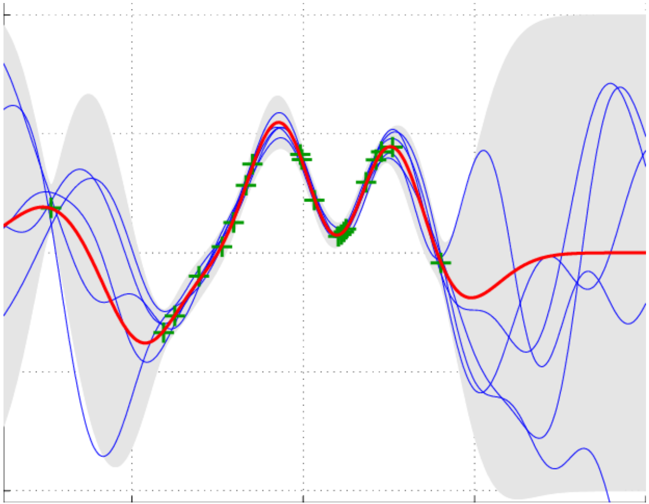
$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$$ parameter prior

Gaussian process

probability distribution over functions $y(\mathbf{x})$ such that if we consider an arbitrary set of points $\mathbf{x}_1, \dots, \mathbf{x}_N$ the joint probability $p(\mathbf{x}_1, \dots, \mathbf{x}_N)$ is Gaussian.

Solved with kernels

Gaussian Processes Regression Example



Source: 10.1109/MSP.2013.2250352 (DOI)