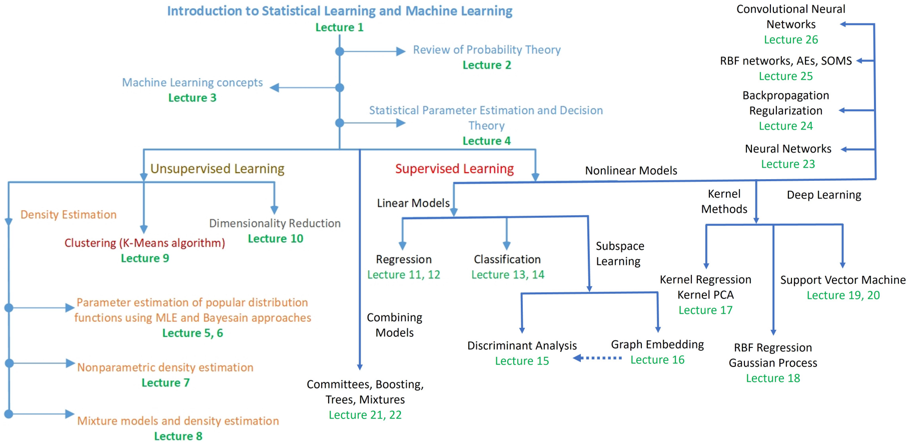
Statistical Learning and Machine Learning

Lecture 18 - Kernel Methods 2

October 13, 2021

Course overview and where do we stand



Why kernel methods?

Main idea in kernel methods:

- The prediction of the model is based on a linear combination of dot-products between data points:

$$\kappa(\mathrm{x}, \mathrm{x}') = \phi(\mathrm{x})^T \phi(\mathrm{x}') \tag{1}$$

  where $\phi(\mathrm{x})$ is an (a-priori defined) (non)linear transformation of the data points x.
- Observation: $\kappa(\cdot, \cdot)$ is a (scalar) value, and this is not depending on the dimensionality of x. $\kappa(\cdot, \cdot)$ is called *kernel function*.
- Whenever a method can be expressed using (only) dot-products between data points, a so-called *dual representation* of the method is defined using $\kappa(\cdot, \cdot)$
- The dot-product between the data points can be replaced by any nonlinear function (satisfying the properties of kernel functions).
- Using a nonlinear kernel function, the nonlinear counterpart of the linear method is defined.

Constructing kernels

We can define a kernel function by:

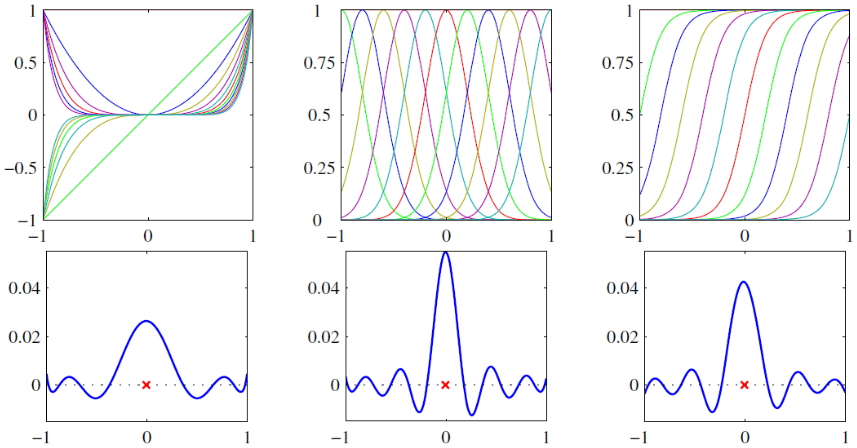
- using a nonlinear function $\phi(\mathrm{x})$ to calculate:

$$\kappa(\mathrm{x}, \mathrm{x}') = \phi(\mathrm{x})^T \phi(\mathrm{x}') \tag{2}$$

- using a set of basis functions $\phi_j(\mathrm{x})$, $j = 1, \ldots, M$ and calculate:

$$\kappa(\mathrm{x}, \mathrm{x}') = \sum_{j=1}^{M} \phi_j(\mathrm{x})^T \phi_j(\mathrm{x}') \tag{3}$$

Constructing kernels



Top: Basis functions using polynomials, Gaussians and ligistic sigmoids
Bottom: corresponding $\kappa(x, x')$ plotted as a function of x for $x' = 0$.

Constructing kernels

Alternatively we can construct a kernel function (expressing similarity) directly, like:

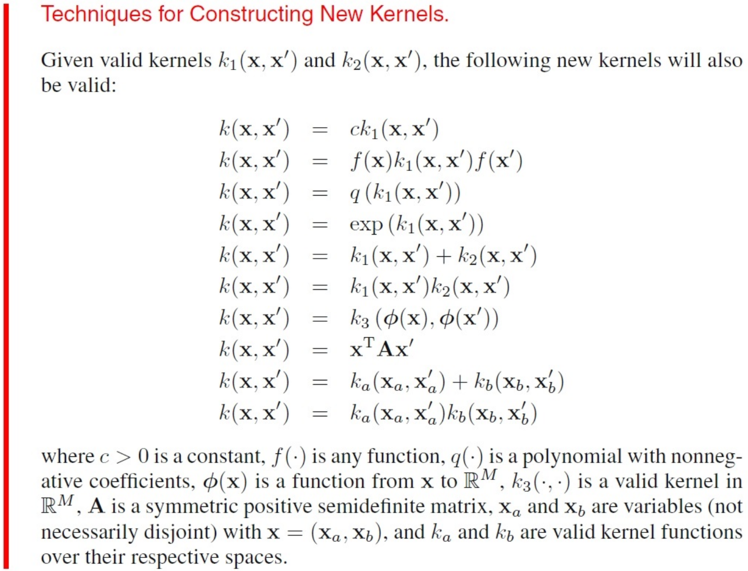
$$\kappa(\mathrm{x}, \mathrm{z}) = \left(\mathrm{x}^T \mathrm{z}\right)^2 . \tag{4}$$

However, not any function is a kernel function. It needs to correspond to a dot-product:

$$\begin{aligned} \kappa(\mathrm{x}, \mathrm{z}) &= \left(\mathrm{x}^T \mathrm{z}\right)^2 = (x_1 z_1 + x_2 z_2)^2 \\ &= x_1^2 z_1^2 + 2 x_1 z_1 x_2 z_2 + x_2^2 z_2^2 \\ &= (x_1^2, \sqrt{2} x_1 x_2, x_2^2)(z_1^2, \sqrt{2} z_1 z_2, z_2^2) \\ &= \phi(\mathrm{x})^T \phi(\mathrm{z}) \end{aligned} \tag{5}$$

A *necessary and sufficient condition* for a kernel function $\kappa(\mathrm{x}, \mathrm{x}')$ to be a valid kernel function is that the corresponding K is positive semi-definite for any choices of x_n, $n = 1, \ldots, N$.

Constructing kernels



Techniques for Constructing New Kernels.

Given valid kernels $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$, the following new kernels will also be valid:

$$
\begin{aligned}
k(\mathbf{x}, \mathbf{x}') &= c k_1(\mathbf{x}, \mathbf{x}') \\
k(\mathbf{x}, \mathbf{x}') &= f(\mathbf{x}) k_1(\mathbf{x}, \mathbf{x}') f(\mathbf{x}') \\
k(\mathbf{x}, \mathbf{x}') &= q\left(k_1(\mathbf{x}, \mathbf{x}')\right) \\
k(\mathbf{x}, \mathbf{x}') &= \exp\left(k_1(\mathbf{x}, \mathbf{x}')\right) \\
k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') \\
k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}') k_2(\mathbf{x}, \mathbf{x}') \\
k(\mathbf{x}, \mathbf{x}') &= k_3\left(\boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}')\right) \\
k(\mathbf{x}, \mathbf{x}') &= \mathbf{x}^{\mathrm{T}} \mathbf{A} \mathbf{x}' \\
k(\mathbf{x}, \mathbf{x}') &= k_a(\mathbf{x}_a, \mathbf{x}_a') + k_b(\mathbf{x}_b, \mathbf{x}_b') \\
k(\mathbf{x}, \mathbf{x}') &= k_a(\mathbf{x}_a, \mathbf{x}_a') k_b(\mathbf{x}_b, \mathbf{x}_b')
\end{aligned}
$$

where $c > 0$ is a constant, $f(\cdot)$ is any function, $q(\cdot)$ is a polynomial with nonnegative coefficients, $\boldsymbol{\phi}(\mathbf{x})$ is a function from $\mathbf{x}$ to $\mathbb{R}^M$, $k_3(\cdot, \cdot)$ is a valid kernel in $\mathbb{R}^M$, $\mathbf{A}$ is a symmetric positive semidefinite matrix, $\mathbf{x}_a$ and $\mathbf{x}_b$ are variables (not necessarily disjoint) with $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$, and k_a and k_b are valid kernel functions over their respective spaces.

Constructing kernels

Commonly used kernel functions:

- Polynomial kernel of order M:

$$\kappa(\mathrm{x}, \mathrm{x}') = \left(\mathrm{x}^T \mathrm{x}' + c\right)^M \tag{6}$$

- Gaussian kernel based on Euclidean distance:

$$\kappa(\mathrm{x}, \mathrm{x}') = exp\left(-\frac{\|\mathrm{x} - \mathrm{x}'\|^2}{2\sigma^2}\right) \tag{7}$$

- Gaussian kernel based on distance function $D(\cdot, \cdot)$:

$$\kappa(\mathrm{x}, \mathrm{x}') = exp\left(-\frac{D(\mathrm{x}, \mathrm{x}')^2}{2\sigma^2}\right) \tag{8}$$

Radial Basis Function Networks

Given a set of data points $\mathrm{x}_n \in \mathbb{R}^D,\ n = 1, \ldots, N$ and targets $\mathrm{t}_n \in \mathbb{R}^C,\ n = 1, \ldots, N$, RBF networks apply the following process:

- Select K prototype vectors $\boldsymbol{\mu}_k \in \mathbb{R}^D,\ k = 1, \ldots, K$
- Transform $\mathrm{x}_n,\ n = 1, \ldots, N$ to $\phi_n,\ n = 1, \ldots, N \in \mathbb{R}^K$ using a radial basis function $\phi(r)$, where $r = \|\mathrm{x} - \boldsymbol{\mu}\|$
- Train a linear regression model from $\tilde{\phi}_n$ to t_n:

$$\mathrm{W} = \left(\Phi\Phi^T\right)^{-1}\Phi\mathrm{T}^T \tag{9}$$

where $\Phi = [\tilde{\phi}_1, \ldots, \tilde{\phi}_N] \in \mathbb{R}^{(K+1)\times N}$, $\tilde{\phi}_n = [1, \phi_n^T]^T$ and $\mathrm{T} \in \mathbb{R}^{C\times N}$

Radial Basis Function Networks

Example Radial Basis Functions

$$\phi(r) = e^{-r^2/2\sigma^2}, \quad \text{Gaussian},$$

$$\phi(r) = \frac{1}{(\sigma^2 + r^2)^{\alpha}}, \quad \alpha > 0,$$

$$\phi(r) = \left(\sigma^2 + r^2\right)^{\beta}, \quad 0 < \beta < 1,$$

$$\phi(r) = r, \quad \text{linear},$$

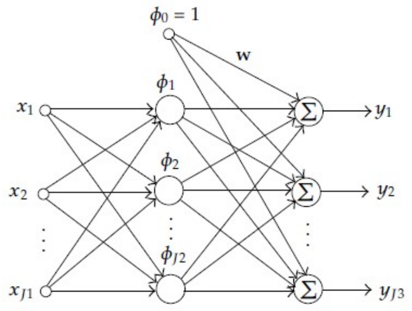
$$\phi(r) = r^2 \ln(r), \quad \text{thin-plate spline},$$

$$\phi(r) = \frac{1}{1 + e^{(r/\sigma^2) - \theta}}, \quad \text{logistic function},$$

Radial Basis Function Networks

The prototype vectors $\boldsymbol{\mu}_k \in \mathbb{R}^D,\ k = 1, \ldots, K$ can be:

- all training data points $\mathrm{x}_n,\ n = 1, \ldots, N$. Then $K = N$
- randomly selected training data points $\mathrm{x}_n,\ n = 1, \ldots, N$
- K mean vectors obtained by clustering the training data points, e.g. by applying the K-Means algorithm



Gaussian Processes

Let us consider the linear regression model:

$$y(\mathrm{x}) = \mathrm{w}^T \phi(\mathrm{x}) \tag{10}$$

We treat w as a parameter vector drawn from the density function:

$$p(\mathrm{w}) = \mathcal{N}(\mathrm{w}|0, \alpha^{-1}\mathrm{I}). \tag{11}$$

where α is the precision (inverse variance) of the distribution.

Note: The linear regression model is also a function of α:

$$y(\alpha, \mathrm{x}) = \mathrm{w}(\alpha)^T \phi(\mathrm{x}) \tag{12}$$

Gaussian Processes

Given a set of data points $\mathrm{x}_n,\ n = 1, \ldots, N$, the joint distribution of the function values is $\mathrm{y} = [y(\mathrm{x}_1), \ldots, y(\mathrm{x}_N)]^T$ and is calculated by:

$$\mathrm{y} = \Phi \mathrm{w} \tag{13}$$

where $\Phi = [\phi(\mathrm{x}_1)^T, \ldots, \phi(\mathrm{x}_N)^T]^T$.

Since w is drawn from a Gaussian distribution, y is a Gaussian too:

$$\begin{aligned} \mathbb{E}[\mathrm{y}] &= \Phi\, \mathbb{E}[\mathrm{w}] = 0 & (14) \\ cov[\mathrm{y}] &= \mathbb{E}[\mathrm{y}\mathrm{y}^T] = \Phi\, \mathbb{E}[\mathrm{w}\mathrm{w}^T]\Phi^T = \frac{1}{\alpha}\Phi\Phi^T = \mathrm{K} & (15) \end{aligned}$$

where K is the Gram matrix:

$$K_{nm} = \kappa(\mathrm{x}_n, \mathrm{x}_m) = \frac{1}{\alpha}\phi(\mathrm{x}_n)^T\phi(\mathrm{x}_m) \tag{16}$$

and $\kappa(\cdot, \cdot)$ is the kernel function.

Gaussian Processes: Regression

We model the noise on the observed target values for data point x_n by:

$$t_n = y(\mathrm{x}_n) + \epsilon_n \tag{17}$$

where ϵ is a random noise variable (independent for each data point).

For noise processes following a Gaussian distribution with precision β:

$$p(t_n|y(\mathrm{x}_n)) = \mathcal{N}(t_n|y(\mathrm{x}_n), \beta^{-1}) \tag{18}$$

The joint distribution of $\mathrm{t} = [t_1, \dots, t_N]^T$ conditioned on $\mathrm{y} = [y_1, \dots, y_N]^T$ is:

$$p(\mathrm{t}|\mathrm{y}) = \mathcal{N}(\mathrm{t}|\mathrm{y}, \beta^{-1}\mathrm{I}_N). \tag{19}$$

Gaussian Processes: Regression

Using $p(\mathrm{y}) = \mathcal{N}(\mathrm{y}|0, \mathrm{K})$ we get:

$$p(\mathrm{t}) = \int p(\mathrm{t}|\mathrm{y})p(\mathrm{y})d\mathrm{y} = \mathcal{N}(\mathrm{t}|0, \mathrm{C}) \tag{20}$$

where the covariance matrix C has elements:

$$C(\mathrm{x}_n, \mathrm{x}_m) = \kappa(\mathrm{x}_n, \mathrm{x}_m) + \beta^{-1}\delta_{nm}. \tag{21}$$

$\delta_{nm} = 1$ for $n = m$, and $\delta_{nm} = 0$ for $n \neq m$.

Since the two Gaussian sources of randomness (w.r.t. $y(\mathrm{x})$ and ϵ) are independent, the joint covariance is equal to the summation of the two.

Gaussian Processes: Regression

Given a new data point x_{N+1}, we want to evaluate the $p(t_{N+1}|\mathrm{t})$.

The joint distribution is:

$$p(\mathrm{t}_{N+1}) = \mathcal{N}(\mathrm{t}_{N+1}|0, \mathrm{C}_{N+1}) \tag{22}$$

where

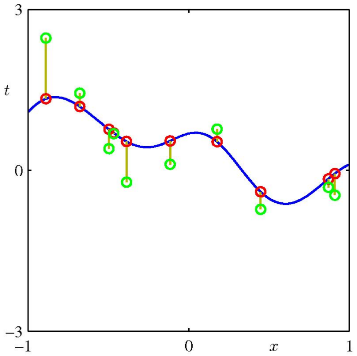
$$\mathrm{C}_{N+1} = \begin{bmatrix} \mathrm{C}_N & \mathrm{k} \\ \mathrm{k}^T & c \end{bmatrix} \tag{23}$$

with $\mathrm{k} = [\kappa(\mathrm{x}_1, \mathrm{x}_{N+1}), \ldots, \kappa(\mathrm{x}_N, \mathrm{x}_{N+1})^T]$ and $c = \kappa(\mathrm{x}_{N+1}, \mathrm{x}_{N+1}) + \beta^{-1}$.

The conditional distribution $p(t_{N+1}|\mathrm{t})$ is a Gaussian with:

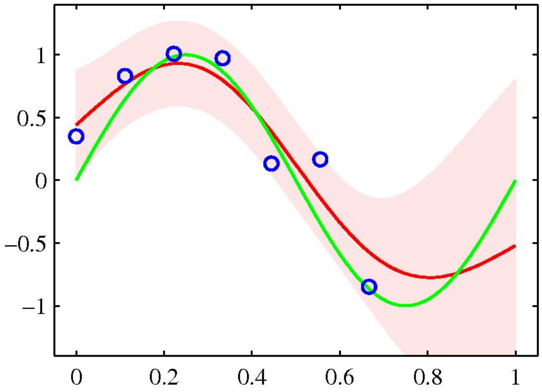
$$\begin{aligned} m(\mathrm{x}_{N+1}) &= \mathrm{k}^T \mathrm{C}_N^{-1} \mathrm{t} & (24) \\ \sigma^2(\mathrm{x}_{N+1}) &= c - \mathrm{k}^T \mathrm{C}_N^{-1} \mathrm{t}. & (25) \end{aligned}$$

Gaussian Processes: Regression



Line: a sample function from the Gaussian process prior over functions
Red points: the values of $y(\mathrm{x}_n)$, $n = 1, \ldots, N$
Green points: targets t_n obtained by adding Gaussian noise to each $y(\mathrm{x}_n)$

Gaussian Processes: Regression



Green line: sinusoidal function generating data points (after adding noise)
Red line: the mean of the Gaussian process predictive distribution
Shaded region: plus and minus two standard deviations on the mean