

Kernel methods

Lecture 11

Recall that $k(\vec{x}, \vec{x}') = \underbrace{\vec{\Phi}(\vec{x})^T \cdot \vec{\Phi}(\vec{x}')}_{\text{feature space mapping}}$

Note that

$$k(\vec{x}, \vec{x}') = k(\vec{x}', \vec{x}).$$

$$\text{If } \vec{\Phi}(\vec{x}) = \vec{x} \Rightarrow k(\vec{x}, \vec{x}') = \underbrace{\vec{x}^T \vec{x}'}_{\text{linear kernel}}$$

$$\text{Often, } k(\vec{x}, \vec{x}') = \underbrace{k(\vec{x} - \vec{x}')}_{\text{translation into kernel}}$$

or even

$$k(\vec{x}, \vec{x}') = \underbrace{k(\|\vec{x} - \vec{x}'\|)}_{\text{homogeneous kernel (radial basis functions)}}$$

For example, kernels are used in density estimation: suppose we draw a sample from $p(\vec{x})$ & wish to estimate it.

$$\text{Consider } p = \int_{\substack{R \\ \nwarrow \text{region around } \vec{x}}} d\vec{x} p(\vec{x})$$

$$\text{Prob}(K \text{ out of } N \text{ points fell within } R) = \frac{N!}{K!(N-K)!} p^K (1-p)^{N-K}$$

$$\begin{cases} E[K] = pN \\ \text{Var}[K] = Np(1-p) \end{cases} \Rightarrow K \approx Np$$

If R is small, $P \approx p(\bar{x})V$.

but also has to be large enough to produce a sizable K $\left. \begin{array}{l} \uparrow \\ \leftarrow \end{array} \right\} p(\bar{x}) \approx \frac{K}{NV}$.

If R is a hypercube centered on \bar{x} , of size $h \rightarrow V = h^D$

we can define $k(\vec{u}) = \begin{cases} 1, & |u_i| \leq \frac{1}{2} \quad \forall i=1, \dots, D \\ 0, & \text{otherwise} \end{cases}$

Then $K = \sum_{n=1}^N k\left(\frac{\bar{x} - \bar{x}_n}{h}\right)$, yielding

$$p(\bar{x}) = \frac{1}{N} \sum_{n=1}^N \frac{1}{h^D} k\left(\frac{\bar{x} - \bar{x}_n}{h}\right)$$

can interpret as a cube centered around \bar{x}_n

sum over hypercube kernels

Note that $\int d\vec{x} p(\vec{x}) = \frac{1}{N} \sum_{n=1}^N \frac{1}{h^D} \int d\vec{x} k\left(\frac{\vec{x} - \bar{x}_n}{h}\right) =$
 $= 1$, as expected

Hypercube kernels are discontinuous \Rightarrow
 \Rightarrow can use Gaussians instead:

$$p(\vec{x}) = \frac{1}{N} \sum_{n=1}^N \frac{1}{\sqrt{2\pi}h^2} e^{-\frac{\|\vec{x} - \bar{x}_n\|^2}{2h^2}}$$

Gaussian placed over each \bar{x}_n &

the sum of Gaussians normalized:

$$\int d\vec{x} p(\vec{x}) = 1$$

h plays the role of a smoothing prm.

We can choose any other kernel subject

to:
$$\begin{cases} \int k(\vec{u}) \geq 0, \\ \int d\vec{u} k(\vec{u}) = 1. \end{cases}$$

This class of models is called Parzen density estimators. Note that the model has no fitting prms, just uses $\{\vec{x}_n\}$ directly.

Dual representations

Consider a linear regression model with regularization:

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

$\lambda \geq 0$

$$\frac{\partial J}{\partial w_i} = \lambda w_i + \sum_{n=1}^N (\underbrace{w_j y_j(\vec{x}_n) - t_n}_{\text{sum over } j \text{ implied}}) y_i(\vec{x}_n) = 0,$$

$$w_i = -\frac{1}{\lambda} \sum_{n=1}^N (w_j y_j(\vec{x}_n) - t_n) y_i(\vec{x}_n) \equiv$$

$$\equiv \sum_{n=1}^N a_n \underbrace{y_i(\vec{x}_n)}_{\phi_{ni}} = \sum_n \phi_{in}^T a_n$$

$\phi_{ni} \leftarrow$ design matrix
" ϕ_{in}^T

So, $\vec{w} = \Phi^T \vec{a}$ \leftarrow vector of dim N , can be

used instead of \vec{w} in a dual representation:

$$J(\vec{a}) = \frac{1}{2} \sum_n [(\vec{a}^T \Phi)_j \phi_{nj} - t_n]^2 + \frac{\lambda}{2} \vec{a}^T \Phi \Phi^T \vec{a} =$$

$$= \frac{1}{2} \sum_n [a_{n'} \phi_{n'j} \phi_{nj} - t_n][a_{n''} \phi_{n''j} \phi_{nj} - t_n] +$$

$$+ \frac{\lambda}{2} \vec{a}^T \Phi \Phi^T \vec{a} \quad \textcircled{=}$$

$$\begin{aligned} \textcircled{=} & \frac{1}{2} \overbrace{a_n^T \Phi_{n,j}^T \Phi_{j,h}^T \Phi_{h,n}^T a_n}^{a_n^T \Phi_{n,j}^T \Phi_{j,h}^T \Phi_{h,n}^T a_n} - \\ & - a_n^T \Phi_{n,j} \underbrace{\Phi_{j,n}}_{\Phi_{j,n}^T} t_n + \frac{1}{2} t_n^T t_n + \frac{\lambda}{2} \vec{a}^T \Phi \Phi^T \vec{a} = \\ & = \frac{1}{2} \vec{a}^T \Phi \Phi^T \Phi \Phi^T \vec{a} - \vec{a}^T \Phi \Phi^T \vec{t} + \frac{1}{2} \vec{t}^T \vec{t} + \frac{\lambda}{2} \vec{a}^T \Phi \Phi^T \vec{a} \end{aligned}$$

Define $\underbrace{K}_{\text{gram matrix}}, N \times N$ symm.

$$K_{nm} = \Phi_{n,j} \underbrace{\Phi_{j,m}^T}_{\Phi_{m,j}} = \Psi_j(\vec{x}_n) \underbrace{\Psi_j(\vec{x}_m)}_{\text{sum over } j} \textcircled{=}$$

$$\textcircled{=} \underbrace{k(\vec{x}_n, \vec{x}_m)}_{\vec{\Psi}^T(\vec{x}_n) \cdot \vec{\Psi}(\vec{x}_m)}, \text{ kernel function}$$

$$\text{Now, } J(\vec{a}) = \frac{1}{2} \vec{a}^T K K \vec{a} - \vec{a}^T K \vec{t} + \frac{1}{2} \vec{t}^T \vec{t} + \frac{\lambda}{2} \vec{a}^T K \vec{a}$$

$$\begin{aligned} \frac{\partial J}{\partial a_n} &= \frac{1}{2} \frac{\partial}{\partial a_n} (a_m K_{mm'} K_{m'p} a_p) - \\ & - \frac{\partial}{\partial a_n} (a_m K_{mm'} t_{m'}) + \frac{\lambda}{2} \frac{\partial}{\partial a_n} (a_m K_{mm'} a_{m'}) = \\ & = \frac{1}{2} (K_{nm'} K_{m'p} a_p + \underbrace{a_m K_{mm'} K_{m'n}}_{K_{nm'} K_{m'n} a_m}) - \\ & - K_{nm'} t_{m'} + \frac{\lambda}{2} (\underbrace{K_{nm'} a_{m'}}_{a_{m'} K_{m'n}} + a_m K_{mn}) \textcircled{=} \end{aligned}$$

$$\textcircled{=} K_{nm'} K_{m'm} a_m - K_{nm'} t_{m'} + \lambda K_{nm} a_m = 0, \text{ or}$$

$$(K_{nm'} K_{m'm} + \lambda \underbrace{K_{nm'}}_{\delta_{m'm}}) a_m = K_{nm'} t_{m'}$$

⇓

$$(K_{m'm} + \lambda) a_m = t_{m'}, \text{ or}$$

$$(K + \lambda \mathbb{1}_N) \vec{a} = \vec{t} \Rightarrow \vec{a} = \underline{\underline{(K + \lambda \mathbb{1}_N)^{-1} \vec{t}}}$$

Finally,

$$y(\vec{x}) = \vec{w}^T \vec{\phi}(\vec{x}) = \underbrace{\vec{a}^T}_{\substack{\text{vector,} \\ \text{dim } M \\ \# \text{ weights}}} \Phi \vec{\phi}(\vec{x}) \textcircled{=}$$

$$\begin{aligned} \textcircled{=} a_n \Phi_{nj} \phi_j(\vec{x}) &= (K + \lambda \mathbb{1}_N)^{-1}_{nm} t_m \phi_j(\vec{x}_n) \phi_j(\vec{x}) = \\ &= \underbrace{k(\vec{x}_n, \vec{x})}_{\text{" } k_n(\vec{x})} (K + \lambda \mathbb{1}_N)^{-1}_{nm} t_m = \underline{\underline{\vec{k}^T (K + \lambda \mathbb{1}_N)^{-1} \vec{t}}} \end{aligned}$$

So, least-squares solution $y(\vec{x})$ is expressed entirely through the kernel functions.

Here we have to invert an $N \times N$ rather than an $M \times M$ matrix (typically, $N \gg M$). However, there are advantages to working directly with $k(\vec{x}, \vec{x}')$, since constructing $\vec{\phi}(\vec{x})$ explicitly can be avoided.

Types of kernels

Idea: construct kernel functions directly.
Valid kernels must correspond to a scalar product in some (possibly ∞) feature space.

For example, consider

$$k(\vec{x}, \vec{z}) = (\vec{x}^T \vec{z})^2 = (x_1 z_1 + x_2 z_2)^2 \quad \ominus$$

↑
2D for simplicity

$$\begin{aligned} \ominus \quad & x_1^2 z_1^2 + 2x_1 x_2 z_1 z_2 + x_2^2 z_2^2 = \\ & = \underbrace{\begin{pmatrix} x_1^2 & \sqrt{2} x_1 x_2 & x_2^2 \end{pmatrix}}_{\vec{\psi}(\vec{x})} \begin{pmatrix} z_1^2 \\ \sqrt{2} z_1 z_2 \\ z_2^2 \end{pmatrix} = \vec{\psi}(\vec{x})^T \vec{\psi}(\vec{z}) \end{aligned}$$

↑
M=3 here
(3 features)

So, $k(\vec{x}, \vec{z})$ is a valid kernel.

However, ideally we do not want to construct $\vec{\psi}(\vec{x})$ explicitly.

One can show that K , with $K_{nm} = k(\vec{x}_n, \vec{x}_m)$, should be positive semi-definite: $\lambda_i \geq 0$, $i=1, \dots, N$ for any data set $\{\vec{x}_n\}$.

This can serve as a check.

Typically, kernels are constructed out of simpler building blocks using various composition properties:

if $k_1(\vec{x}, \vec{x}')$ & $k_2(\vec{x}, \vec{x}')$ are valid kernels:

(1) $k(\vec{x}, \vec{x}') = \underbrace{c}_{\text{const} > 0} k_1(\vec{x}, \vec{x}')$ is also a valid kernel

(2) $k(\vec{x}, \vec{x}') = f(\vec{x}) k_1(\vec{x}, \vec{x}') f(\vec{x}')$ is valid

(3) $k(\vec{x}, \vec{x}') = k_1(\vec{x}, \vec{x}') + k_2(\vec{x}, \vec{x}')$ is valid

Indeed, $k(\vec{x}, \vec{x}') = \sum_{i=1}^{M_1} \psi_i(\vec{x}) \psi_i(\vec{x}') + \sum_{j=1}^{M_2} \varphi_j(\vec{x}) \varphi_j(\vec{x}') \equiv$

new feature space $\rightarrow \equiv \sum_{i=1}^{M_1+M_2} \tilde{\psi}_i(\vec{x}) \tilde{\psi}_i(\vec{x}')$

if some features are the same the dimension of the new feature space is $< M_1 + M_2$

(4) $k(\vec{x}, \vec{x}') = k_1(\vec{x}, \vec{x}') k_2(\vec{x}, \vec{x}') =$

$$= \left[\sum_{i=1}^{M_1} \psi_i(\vec{x}) \psi_i(\vec{x}') \right] \left[\sum_{j=1}^{M_2} \varphi_j(\vec{x}) \varphi_j(\vec{x}') \right] =$$

$$= \sum_{i,j} \psi_i(\vec{x}) \varphi_j(\vec{x}) \psi_i(\vec{x}') \varphi_j(\vec{x}') = \sum_{i,j} \tilde{\psi}_{ij}(\vec{x}) \tilde{\psi}_{ij}(\vec{x}')$$

$\psi_i(\vec{x}) \varphi_j(\vec{x}) \equiv \tilde{\psi}_{ij}(\vec{x})$

$$= \sum_{k=1}^{M_1 M_2} \tilde{\psi}_k(\vec{x}) \tilde{\psi}_k(\vec{x}')$$

(5) Similarly, $k(\vec{x}, \vec{x}') = (k_1(\vec{x}, \vec{x}'))^M$ is valid
 $M \in \mathbb{Z}, M > 0$

(6) $k(\vec{x}, \vec{x}') = e^{k_1(\vec{x}, \vec{x}')}$ is valid

etc.

We can use these rules to construct complicated kernels

For example,

$$\begin{cases} k(\vec{x}, \vec{x}') = \vec{x}^T \vec{x}' & \text{is valid (linear kernel)} \\ k(\vec{x}, \vec{x}') = c & \text{is valid} \\ & c > 0, M=1 \end{cases}$$

Then $k(\vec{x}, \vec{x}') = \vec{x}^T \vec{x}' + c$ is valid.

Further, $k(\vec{x}, \vec{x}') = (\vec{x}^T \vec{x}' + c)^M$ is valid.

Another valid kernel is gaussian:

$$k(\vec{x}, \vec{x}') = e^{-\frac{\|\vec{x} - \vec{x}'\|^2}{2\sigma^2}}$$

Indeed, $k(\vec{x}, \vec{x}') = e^{-\frac{(\vec{x}^T \vec{x} + \vec{x}'^T \vec{x}' - 2\vec{x}^T \vec{x}')}{2\sigma^2}} =$

$$= \underbrace{e^{-\frac{\vec{x}^T \vec{x}}{2\sigma^2}}}_{f(\vec{x})} \underbrace{e^{\frac{\vec{x}^T \vec{x}'}{\sigma^2}}}_{\text{valid}} \underbrace{e^{-\frac{\vec{x}'^T \vec{x}'}{2\sigma^2}}}_{f(\vec{x}')} =$$

Nadaraya - Watson model

Consider dataset $\{\vec{x}_n, t_n\}$, use

Parzen density estimator:

$$p(\vec{x}, t) = \frac{1}{N} \sum_{n=1}^N f(\vec{x} - \vec{x}_n, t - t_n)$$

$f(\vec{x}, t)$ is the component density function: $\int dt d\vec{x} f(\vec{x}, t) = 1$

Now, recall that

$$y(\vec{x}) = E[t|\vec{x}] = \underbrace{\int_{-\infty}^{\infty} dt t p(t|\vec{x})}_{\substack{\text{regression f'n} \\ p(\vec{x}, t) \\ p(\vec{x})}} = \frac{\int dt t p(\vec{x}, t)}{\int dt p(\vec{x}, t)} \quad (\ominus)$$

$$\ominus \quad \frac{\sum_n \int dt t f(\vec{x} - \vec{x}_n, t - t_n)}{\sum_m \int dt f(\vec{x} - \vec{x}_m, t - t_m)}$$

Assume that $\int_{-\infty}^{\infty} dt t f(\vec{x}, t) = 0, \forall \vec{x}$

Then $t - t_n = u \Rightarrow t = u + t_n$

$$y(\vec{x}) = \frac{\sum_n \left[\int du u f(\vec{x} - \vec{x}_n, u) + t_n \int du f(\vec{x} - \vec{x}_n, u) \right]}{\sum_m g(\vec{x} - \vec{x}_m)} \quad (\ominus)$$

$$\ominus \quad \frac{\sum_n g(\vec{x} - \vec{x}_n) t_n}{\sum_m g(\vec{x} - \vec{x}_m)} = \underbrace{\sum_n k(\vec{x}, \vec{x}_n) t_n}_{\substack{\text{kernel regression} \\ \text{(NW model)}}} \quad (*)$$

$$\left\{ \begin{aligned} k(\vec{x}, \vec{x}_n) &= \frac{g(\vec{x} - \vec{x}_n)}{\sum_m g(\vec{x} - \vec{x}_m)}, \\ g(\vec{x}) &= \int_{-\infty}^{\infty} dt f(\vec{x}, t). \end{aligned} \right.$$

Note that $\sum_{n=1}^N k(\vec{x}, \vec{x}_n) = 1 \quad (**)$

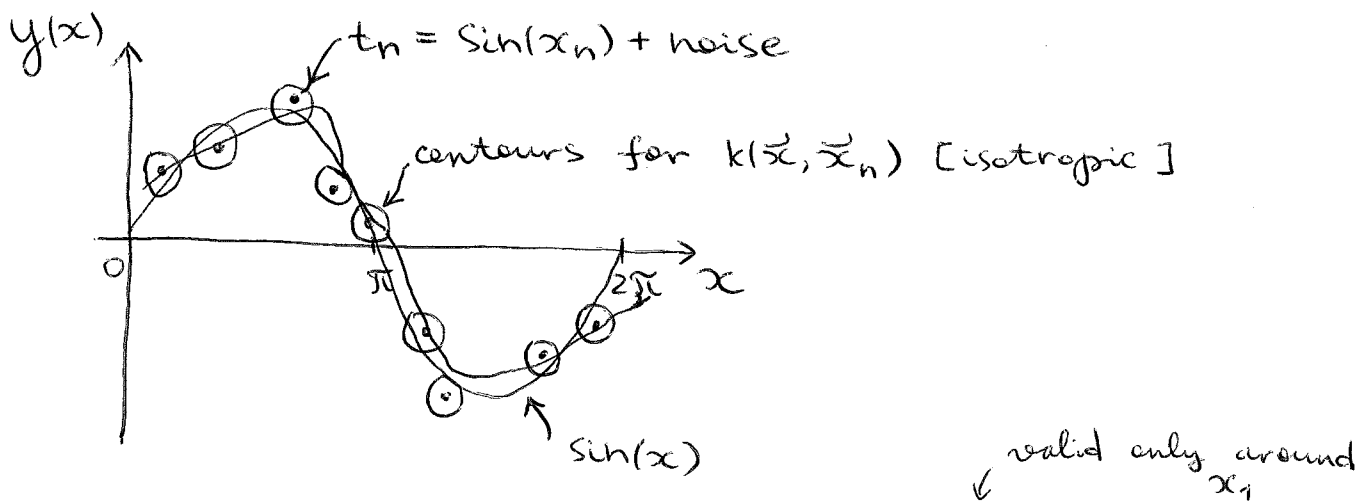
This is the same as (3.61) & (3.64) obtained by "regular" Bayesian regression

As before, $y(\vec{x}) = \sum_{n=1}^N k(\vec{x}, \vec{x}_n) t_n$ is influenced more by those \vec{x}_n that are close to \vec{x} , for the localized kernel.

Ex. Consider $f(x, t) = e^{-\frac{x^2}{2\sigma^2}} e^{-\frac{t^2}{2\sigma^2}}$

(1D) Then $g(x) = \sqrt{2\pi\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$, and $\{x_n, t_n\}_1^N$

$$k(\vec{x}, \vec{x}_n) = \frac{e^{-\frac{(x-x_n)^2}{2\sigma^2}}}{\sum_{m=1}^N e^{-\frac{(x-x_m)^2}{2\sigma^2}}} \Rightarrow y(\vec{x}) = \sum_n k(\vec{x}, \vec{x}_n) t_n$$



$$N=1: k(\vec{x}, \vec{x}_1) = 1 \Rightarrow y(\vec{x}) = t_1$$

$N > 1$: at a given \vec{x} , $y(\vec{x})$ is influenced by several kernels