Kernel methods

{Lecture 19}

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

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

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

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

or even

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

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

Consider $P = \int_R d\vec{x}\, p(\vec{x})$

R ← region around $\vec{x}$

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

$$\begin{cases} E[K] = PN \\ \text{var}[K] = NP(1-P) \end{cases} \Rightarrow K \simeq NP$$

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

but also has to be large enough to produce a sizable K

$$p(\vec{x}) \simeq \frac{K}{NV}.$$

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

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

$$p(\vec{x}) = \underbrace{\frac{1}{N} \sum_{n=1}^{N} \frac{1}{h^D} \underbrace{k\left(\frac{\vec{x}-\vec{x}_n}{h}\right)}_{\text{can interpret as a cube centered around } \vec{x}_n}}_{\text{sum over hypercube kernels}}$$

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

$= \underline{\underline{1}}$, as expected

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

$$p(\vec{x}) = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{\sqrt{2\pi h^2}} \underbrace{e^{-\frac{\|\vec{x}-\vec{x}_n\|^2}{2h^2}}}_{\text{gaussian placed over each } \vec{x}_n}$$

& 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:

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

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:

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

$$\lambda \geq 0$$

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

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

$$\equiv \sum_{n=1}^{N} a_n \underbrace{\varphi_i(\vec{x}_n)}_{\substack{\varphi_{ni} \leftarrow \text{design matrix} \\ = \varphi^T_{in}}} = \sum_n \varphi^T_{in} a_n.$$

So, $\vec{w} = \varphi^T\vec{a}$ ← vector of dim N, can be used instead of $\vec{w}$ in a dual representation:

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

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

$$+ \frac{\lambda}{2}\vec{a}^T\varphi\varphi^T\vec{a} \;=$$

$$\overset{?}{=} \frac{1}{2} \overbrace{a_{n'} \varphi_{n'j'} a_{n''} \varphi_{n''j''} \varphi_{nj'} \varphi_{nj''}}^{a_{n'} \varphi_{n'j'} \varphi^T_{j'n} \varphi_{nj''} \varphi^T_{j''n''} a_{n''}} -$$

$$- a_{n'} \varphi_{n'j} \underbrace{\varphi_{nj}}_{\varphi^T_{jn}} t_n + \frac{1}{2} t_n t_n + \frac{\lambda}{2} \vec{a}^T \varphi \varphi^T \vec{a} =$$

$$= \frac{1}{2} \vec{a}^T \varphi \varphi^T \varphi \varphi^T \vec{a} - \vec{a}^T \varphi \varphi^T \vec{t} + \frac{1}{2} \vec{t}^T \vec{t} + \frac{\lambda}{2} \vec{a}^T \varphi \varphi^T \vec{a}$$

Define $\underbrace{K}_{\text{gram matrix}} = \varphi \varphi^T$

gram matrix, $N \times N$ symm.

$$K_{nm} = \varphi_{nj} \underbrace{\varphi^T_{jm}}_{\varphi_{mj}} = \varphi_j(\vec{x}_n) \varphi_j(\vec{x}_m) \overset{?}{=}$$

sum over j

$$\overset{?}{=} \underbrace{k(\vec{x}_n, \vec{x}_m)}_{\vec{\varphi}^T(\vec{x}_n) \cdot \vec{\varphi}(\vec{x}_m)} = K_{mn}.$$

$\vec{\varphi}^T(\vec{x}_n) \cdot \vec{\varphi}(\vec{x}_m)$, kernel function

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

$$+ \frac{\lambda}{2} \vec{a}^T K \vec{a}.$$

$$\frac{\partial J}{\partial a_n} = \frac{1}{2} \frac{\partial}{\partial a_n} (a_m K_{mm'} K_{m'p} a_p) -$$

$$- \frac{\partial}{\partial a_n} (a_m K_{mm'} t_{m'}) + \frac{\lambda}{2} \frac{\partial}{\partial a_n} (a_m K_{mm'} a_{m'}) =$$

$$= \frac{1}{2} (K_{nm'} K_{m'p} a_p + \overbrace{a_m K_{mm'} K_{m'n}}^{K_{nm'} K_{m'm} a_m}) -$$

$$- K_{nm'} t_{m'} + \frac{\lambda}{2} (\underbrace{K_{nm'} a_{m'}}_{a_{m'} K_{m'n}} + a_m K_{mn}) \overset{?}{=}$$

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

$$(K_{nm'}K_{m'm} + \lambda \overbrace{K_{nm'})\,a_m}^{\delta_{m'm}} = K_{nm'}t_{m'}$$

$$\Downarrow$$

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

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

Finally,

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

$$\ominus \quad a_n \Phi_{nj}\varphi_j(\vec{x}) = (K + \lambda \mathbb{1}_N)^{-1}_{nm} t_m \varphi_j(\vec{x}_n)\varphi_j(\vec{x}) =$$

$$= \underbrace{k(\vec{x}_n, \vec{x})}_{\equiv k_n(\vec{x})} (K + \lambda \mathbb{1}_N)^{-1}_{nm} t_m = \underline{\underline{\vec{k}^T (K + \lambda \mathbb{1}_N)^{-1}\vec{t}}}.$$

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 $\vec{\varphi}(\vec{x})$ 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

$$k(\vec{x},\vec{z}) = (\vec{x}^T\vec{z})^2 \underset{\substack{\uparrow \\ \text{2D for simplicity}}}{=} (x_1 z_1 + x_2 z_2)^2 =$$

$$= x_1^2 z_1^2 + 2x_1x_2z_1z_2 + x_2^2 z_2^2 =$$

$$= \begin{pmatrix} x_1^2 & \sqrt{2}\,x_1x_2 & x_2^2 \end{pmatrix} \begin{pmatrix} z_1^2 \\ \sqrt{2}\,z_1z_2 \\ z_2^2 \end{pmatrix} \equiv \vec{\varphi}(\vec{x})^T \vec{\varphi}(\vec{z}).$$

M=3 here (3 features)

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

However, ideally we do not want to construct $\vec{\varphi}(\vec{x})$ 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, $k(\vec{x}, \vec{x}') = \sum_{i=1}^{M_1} \varphi_i(\vec{x}) \varphi_i(\vec{x}') + \sum_{j=1}^{M_2} \psi_j(\vec{x}) \psi_j(\vec{x}') \equiv$

new feature space → $\equiv \sum_{i=1}^{M_1+M_2} \tilde{\varphi}_i(\vec{x}) \tilde{\varphi}_i(\vec{x}')$ if some features are the same the dimension of the new feature space is $< M_1 + M_2$

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

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

$$= \sum_{i,j} \varphi_i(\vec{x}) \psi_j(\vec{x}) \varphi_i(\vec{x}') \psi_j(\vec{x}') =$$

$\varphi_i(\vec{x}) \psi_j(\vec{x}) \equiv \tilde{\varphi}_k(\vec{x})$

← new feature space

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

(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

<u>etc.</u>

We can use these rules to construct complicated kernels

For example,

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

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

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

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

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

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

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

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

$$y(\vec{x}) = \frac{\sum_n \Big[\int du\, u\, \cancel{f(\vec{x}-\vec{x}_n, u)}^{\,0} + t_n \overbrace{\int du\, f(\vec{x}-\vec{x}_n, u)}^{\overset{\shortparallel}{g(\vec{x}-\vec{x}_n)}} \Big]}{\sum_m g(\vec{x}-\vec{x}_m)} \;=$$

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

$$\begin{cases} k(\vec{x},\vec{x}_n) = \dfrac{g(\vec{x}-\vec{x}_n)}{\sum_m g(\vec{x}-\vec{x}_m)}, \\[2ex] g(\vec{x}) = \displaystyle\int_{-\infty}^{\infty} dt\, f(\vec{x},t) . \end{cases}$$

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

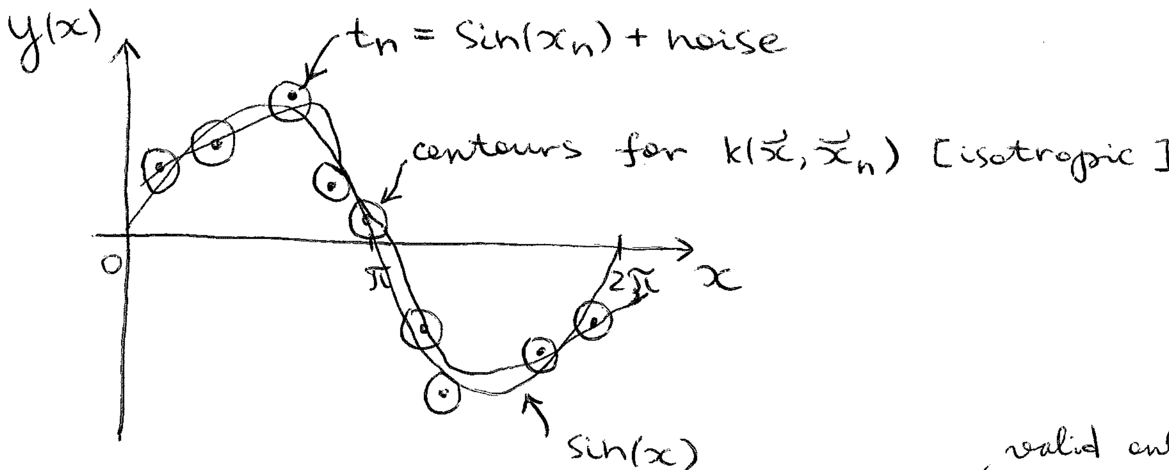
This is the same as (3.61) & (3.64)
(*)&(**) obtained by "regular"
Bayesian regression

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.

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

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

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



valid only around x_1

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

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