Non-parametric Methods

$$D = \{x_i, y_i\}$$

e.g. regression

1) Assume PDF for y_i

$$y(\vec{x}) \sim N(y \mid \mu(\vec{x}, \vec{w}), \sigma^2)$$

2) Take a model for $\mu(\vec{x}, \vec{w})$

(e.g Neural net, basis functions)

3) Fit $\vec{w}$

⇒ parametric methods

⇒ distribution is fixed

- its shape
- the mean function

Non-param methods

today

Gaussian Processes

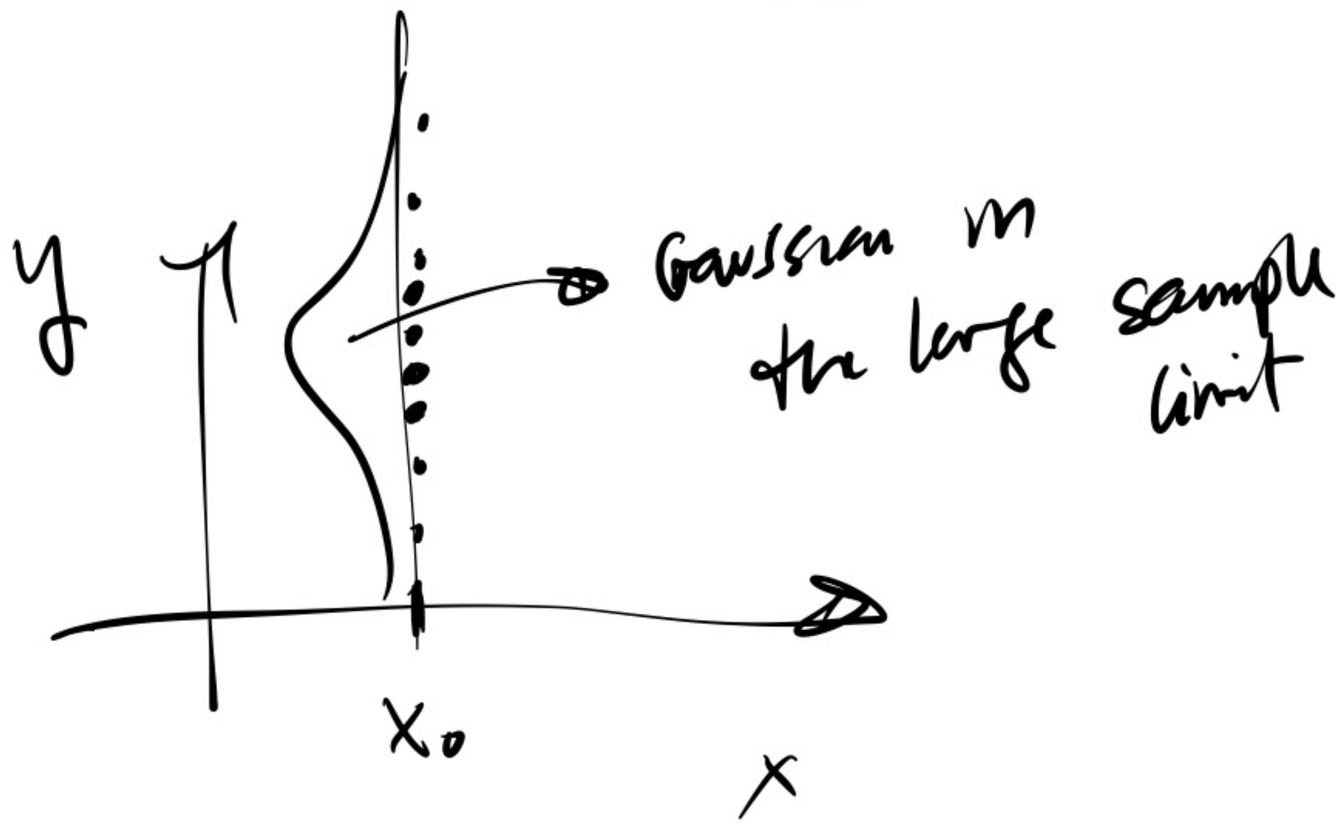
⇒ distribution is not fixed a priori

⇒ mean function is not fixed a priori

The distrib is fixed by data

(if dataset is larger N) the distrib.

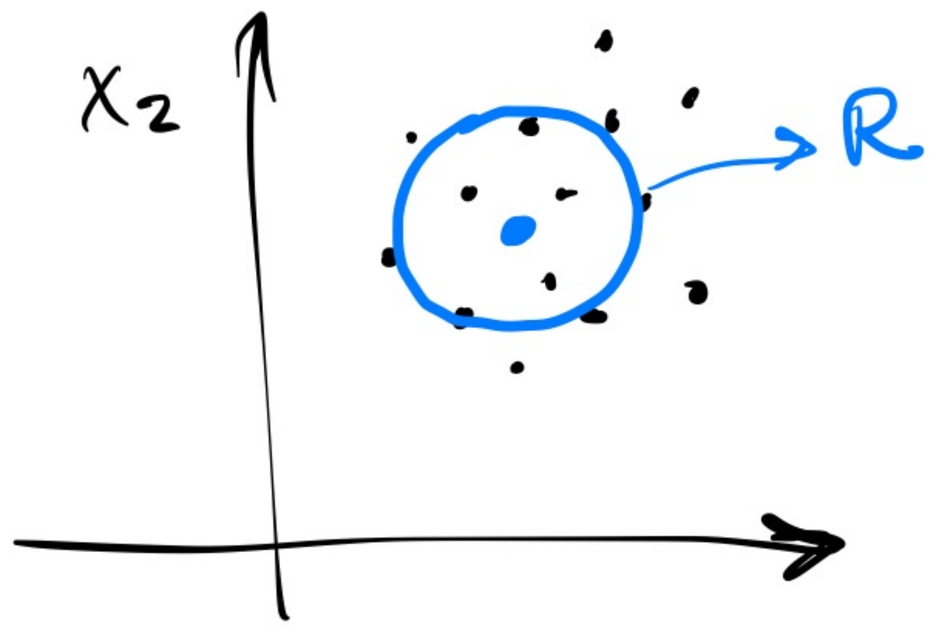
will be more flexible



$$D = \{ \vec{x}_i \}_{i=1}^{N}$$

estimate the PDF of $\vec{x}$

$p(\vec{x})$



Prob. mass associated to R

$$P = \int_R p(\vec{x}) d\vec{x}$$ { prob of each point to fall in R

The prob for k (out of N) points

$$\mathrm{Bin}(K \mid N, P) = \frac{N!}{K!\,(N-K)!} P^K (1-P)^{N-K}$$

$$\mathbb{E}[K] = NP$$

$$\mathbb{E}[K/N] = P \qquad (I)$$

$$\mathrm{Var}[K] = NP(1-P)$$

variance of K/N?

$$\mathrm{Var}[K/N] = \mathbb{E}\left[\left(\frac{K}{N}\right)^2\right] - \left(\mathbb{E}\left[\frac{K}{N}\right]\right)^2$$

$$= \frac{1}{N^2}\mathbb{E}[K^2] - P^2$$

$$= \frac{1}{N^2}\Big(\underbrace{\mathrm{Var}[K]}_{NP(1-P)} + \underbrace{(\mathbb{E}[K])^2}_{N^2P^2}\Big) - P^2$$

$$= \frac{P(1-P)}{N}$$ } decreases with N

large N dist. peaked at the mean

– Assume that R is small enough

(II) $$P \approx p(\vec{x})\, V$$

$p(\vec{x})$ roughly const in R

$$\frac{k}{N} \approx p(\vec{x})\, V \Rightarrow \boxed{p(\vec{x}) \approx \frac{k}{NV}}$$

Two possibilities to proceed:

1) Fix k and determine V from data

↳ k-nearest neighbors

2) Fix V and determine k from data

↳ (kernel methods)

k-nearest neighbors

k fixed ⇒ determines the model complexity

$\vec{x}$

e.g. classification M classes $\{C_m\}$

dataset N points $N = \sum_m^M N_m$

Task is to classify $\vec{x}$

↳ consider k nearest neighbors to $\vec{x}$

k_m = # of points (out of k) belonging to C_m

$$p(\vec{x}|C_m) = \frac{k_m}{N_m \cdot V}, \quad p(\vec{x}) = \frac{k}{N \cdot V}$$

Take class priors $p(C_m) = N_m/N$

$$p(C_m|\vec{x}) = \frac{p(\vec{x}|C_m)\, p(C_m)}{p(\vec{x})}$$

$$= \frac{k_m}{N_m \cdot V} \cdot \frac{N_m}{N} \cdot \frac{N \cdot V}{k} = \frac{k_m}{k}$$

The best class is the one where k_m/k is maximal

↳ majority vote

For regression :

Consider both $\vec{x}$ and y to be random

$$\langle y(\vec{x}) \rangle = \frac{\int dy \cdot y \cdot p(\vec{x}, y)}{\int dy \cdot p(\vec{x}, y)} =$$

↓

"regression function"

$$= \frac{\int dy \cdot y \cdot p(y|\vec{x}) \cdot \cancel{p(\vec{x})}}{\cancel{p(\vec{x})}}$$

$$= \int dy \cdot y \cdot p(y|\vec{x}) = \mathbb{E}[y|\vec{x}]$$

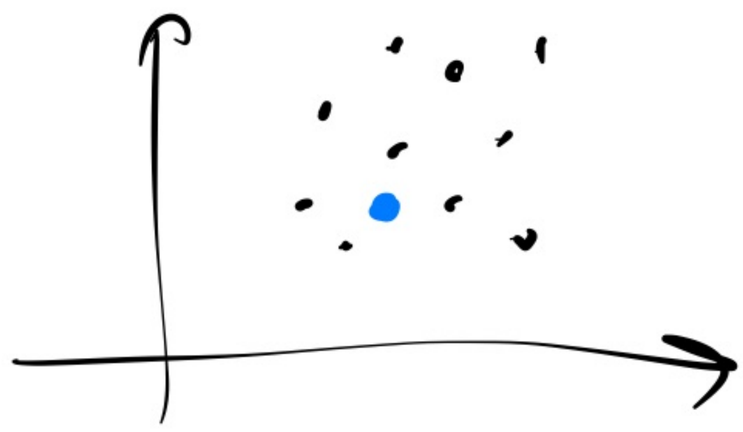
↳ approximated in two ways

- $\mathbb{E} \approx$ average over training sample
- conditioning at a point
  ↳ conditioning at a region close to the point

$$\langle y(\vec{x}) \rangle \approx \frac{1}{k} \sum_{i \in V(\vec{x})}^{k} y_i$$

} average output of all the k neighbors

$K = N \Rightarrow$ simplest model



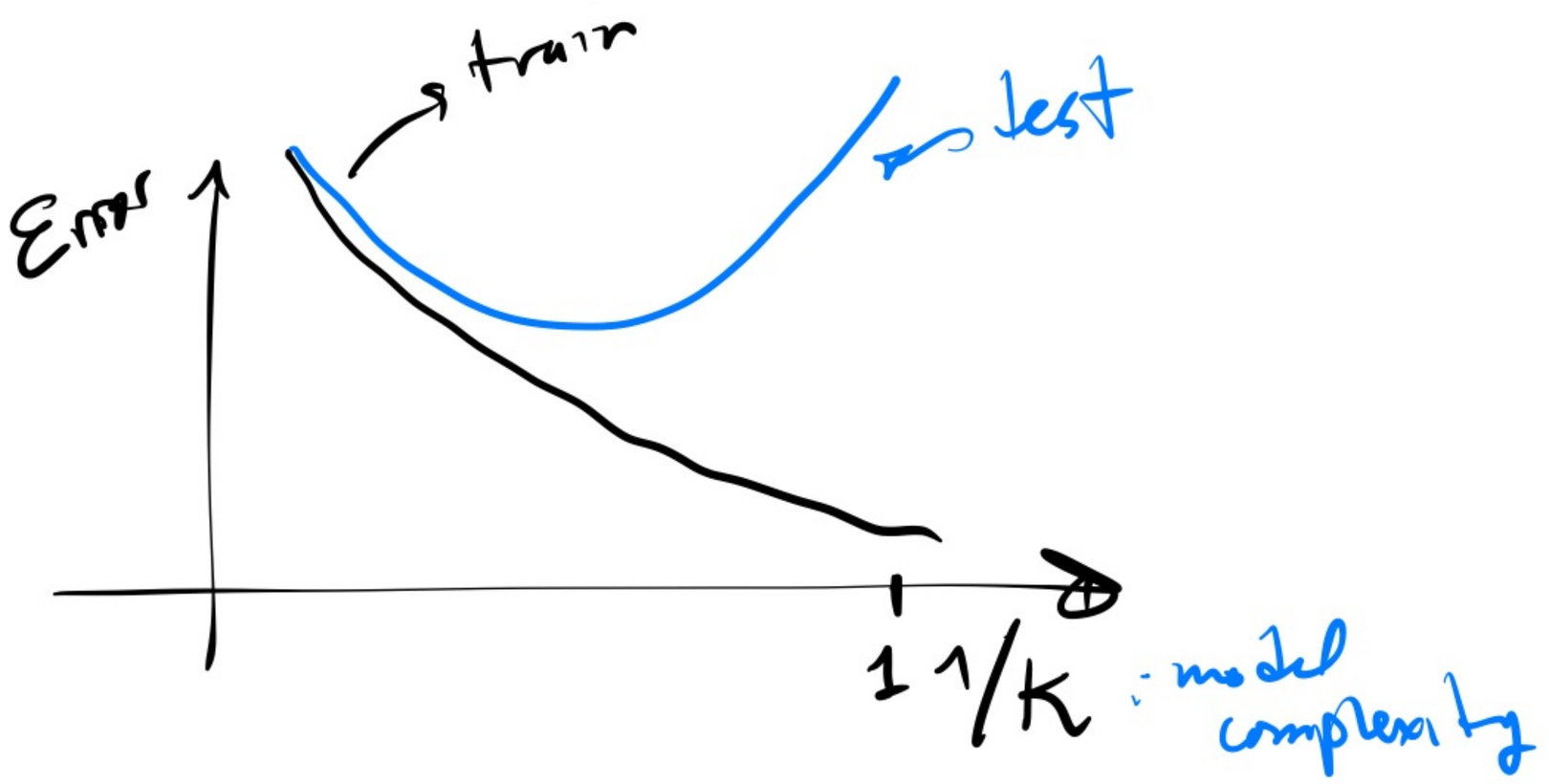
↓ high bias
low variance

↓ average of all the points

$K = 1 \Rightarrow$ more complex model

↓ overfit

↘ large variance (and low bias)

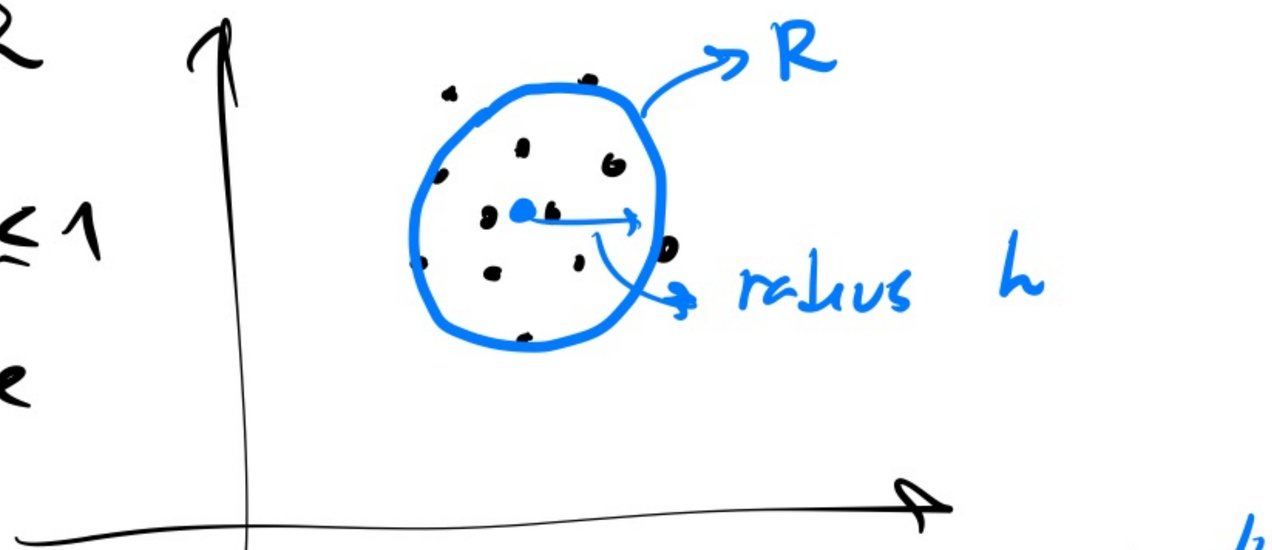


Kernel methods

Fix V and estimate k from the data

Count k enter in R

$$k(\vec{u}) = \begin{cases} 1 & \text{if } |\vec{u}| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$



$$k\left(\frac{\vec{x} - \vec{x_i}}{h}\right) = \begin{cases} 1 & \forall i \text{ inside } R \\ 0 & \text{otherwise} \end{cases}$$

} → "kernel function"

$$K = \sum_{i=1}^{N} k\left(\frac{\vec{x} - \vec{x}_i}{h}\right) \qquad (II)$$

reinterpreted as sum of N balls centered on $\vec{x}_i$

$$p(\vec{x}) = \frac{K}{NV} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{V(h)} k\left(\frac{\vec{x} - \vec{x}_i}{h}\right)$$

1962 Parzen proved ↑ is an unbiased estimates of the true PDF

i.e. $\lim_{N \to \infty} p(\vec{x}) = p_{true}(\vec{x})$

under mild conditions on $k(\vec{u})$

- $\sup_{-\infty < u < \infty} |k(u)| < \infty$
- $\int_{-\infty}^{\infty} du\, |k(u)| < \infty$
- $\lim_{u \to \infty} |u\, k(u)| < \infty$
- $\int_{-\infty}^{\infty} k(u)\, du = 1$ { the volume $V(h)$ is absorbed inside $k(u)$

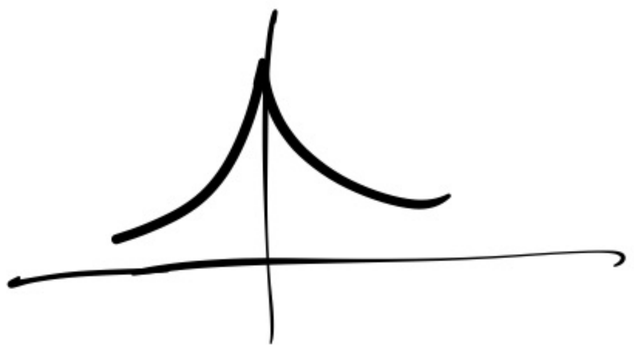
Examples of kernels

- Gaussian kernel

$$k(u;h) = \frac{1}{\sqrt{2\pi}\,h} e^{-u^2/2h^2}$$

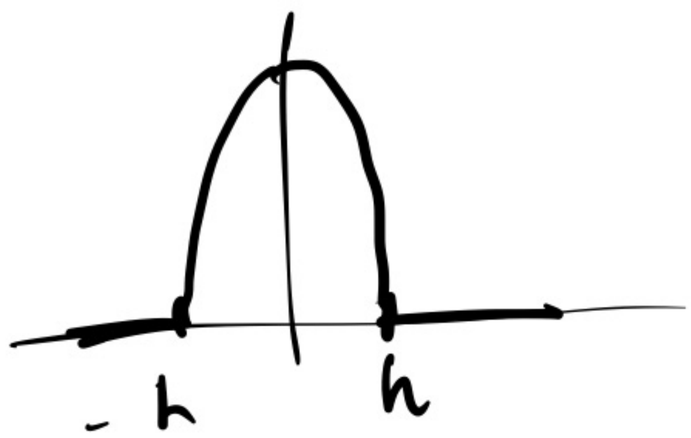
- Exponential

$$k(u;h) = \frac{1}{2h} e^{-|u|/h}$$



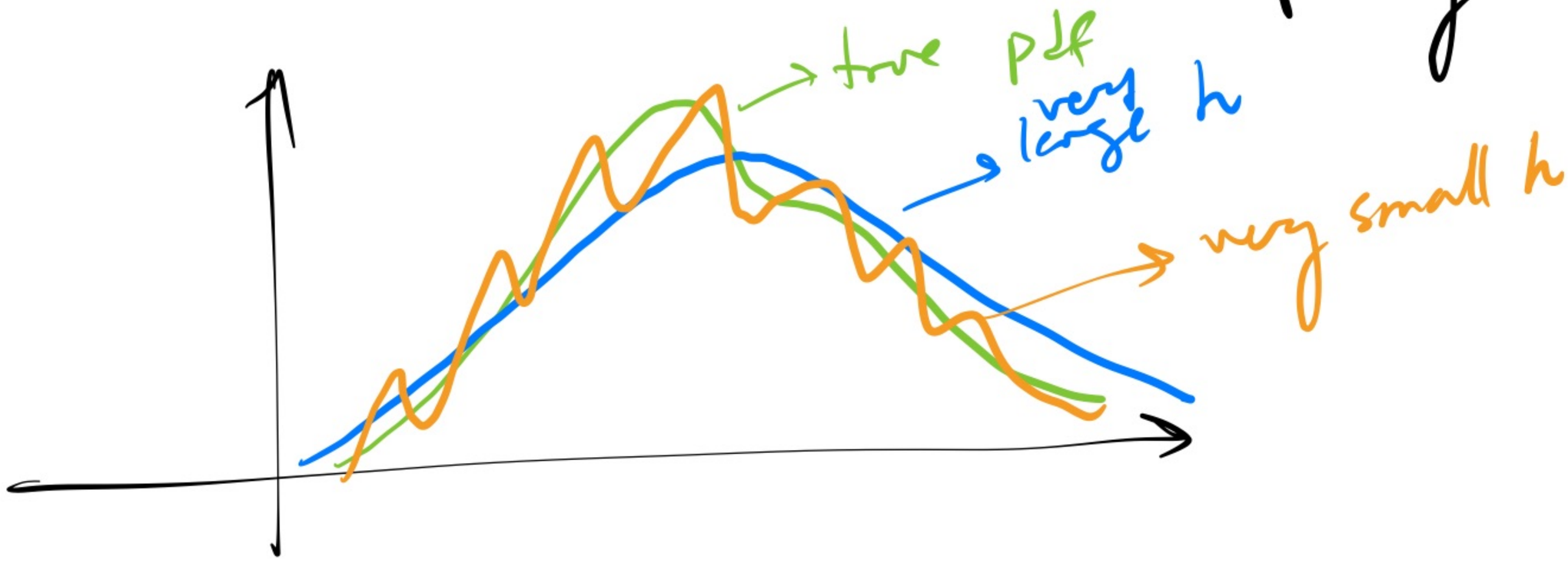
- Cosine kernel

$$k(u;h) = \begin{cases} \frac{\pi}{4h}\cos\left(\frac{\pi u}{2h}\right) & u \le h \\ 0 & u > h \end{cases}$$



∴ many other kernels

Parameter h ⇒ controls the model complexity



Kernel regression

$$D = \{\vec{x}_i, y_i\}$$

$$\langle y(\vec{x}) \rangle = \frac{\int dy \cdot y \cdot p(\vec{x}, y)}{\int dy \; p(\vec{x}, y)}$$

$$p(\vec{x}, y) = \frac{1}{N} \sum_{i=1}^{N} k_x(\vec{x} - \vec{x}_i; h_x)\, k_y(y - y_i; h_y)$$

Generalization of Specht '91

$$\langle y(\vec{x}) \rangle = \frac{\sum_i k_x(\vec{x} - \vec{x}_i; h_x) \int_{-\infty}^{\infty} dy \cdot y \cdot k(y - y_i; h_y)}{\sum_i k_x(\vec{x} - \vec{x}_i; h_x) \int_{-\infty}^{\infty} dy \cdot k_y(y - y_i; h_y)}$$

if $k_y(x; h_y) \Rightarrow$ follows a distrib.

$$\int dy \cdot y \cdot k(y - y_i; h_y) = y_i$$

$$\langle y(\vec{x}) \rangle = \frac{\sum_i y_i \cdot k_x(\vec{x} - \vec{x}_i; h_x)}{\sum_i k_x(\vec{x} - \vec{x}_i; h_x)}$$

Nadaraya-Watson model

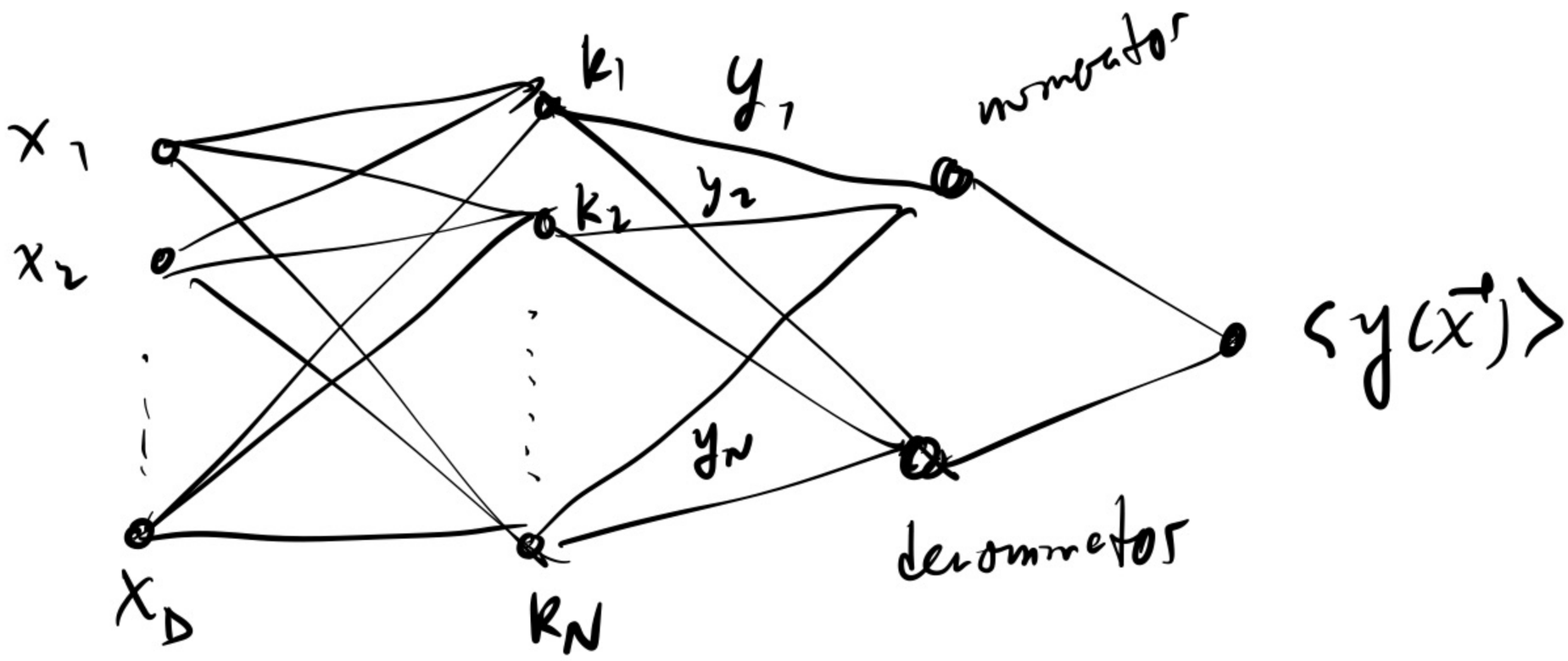
weighted average of all the data points

in the original paper (Specht)

$\Downarrow$ h optimised by minimising MSE

MLE

$$\mathcal{L}(h_x, h_y) = \prod_{i=1}^{N} \frac{1}{N-1} \sum_{j \neq i}^{N-1} k_x(\vec{x}_i - \vec{x}_j; h_x) \times k_y(y_i - y_j; h_y)$$



= General Regression Neural Network (Specht)

Training is a one-step procedure

= the whole dataset stored in memory

Prediction is costly

Classification with kernel methods

⇒ Generative approach

$$p(C_k|\vec{x}) \Leftarrow p(\vec{x}|C_k)$$

$$p(\vec{x}|C_k) = \frac{1}{N_k} \sum_{i=1}^{N_k} k_x(\vec{x}-\vec{x}_i; h)$$

$$p(C_k) = N_k/N$$

$$p(C_k|\vec{x}) \propto p(C_k) \frac{1}{N_k} \sum_{i=1}^{N_k} k_x(\vec{x}-\vec{x}_i, h)$$

$$\hat{C}_k = \text{argmax}\ p(C_k|\vec{x})$$

→ Probabilistic Neural network ⇒