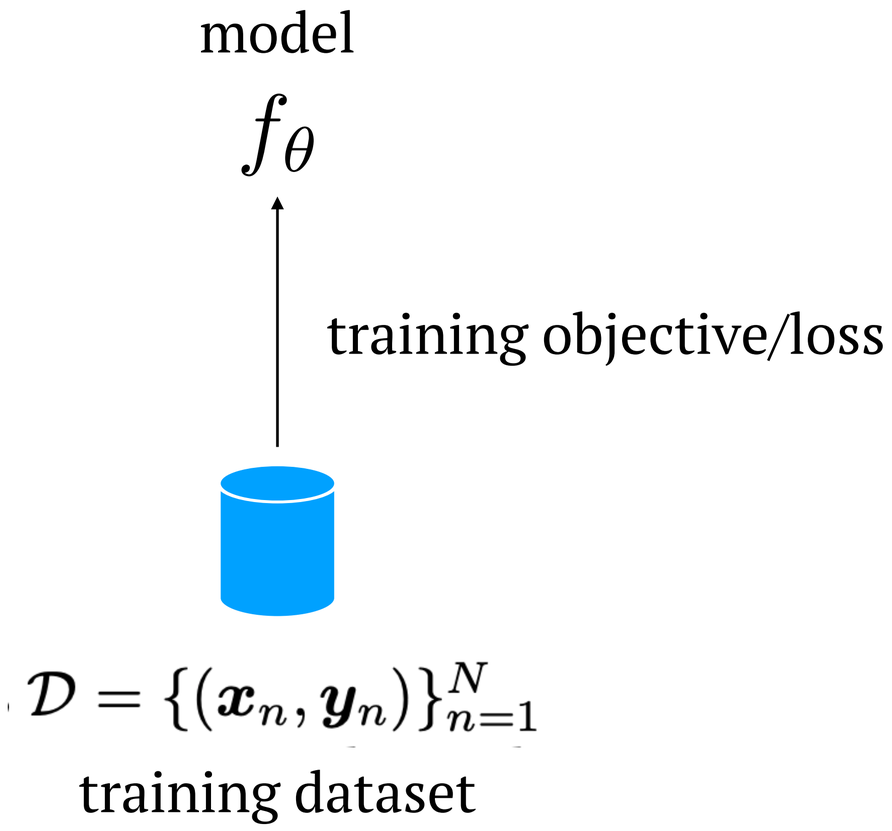
Nonparametric Models (KNN and KDE)

COMP3314 — Week 7

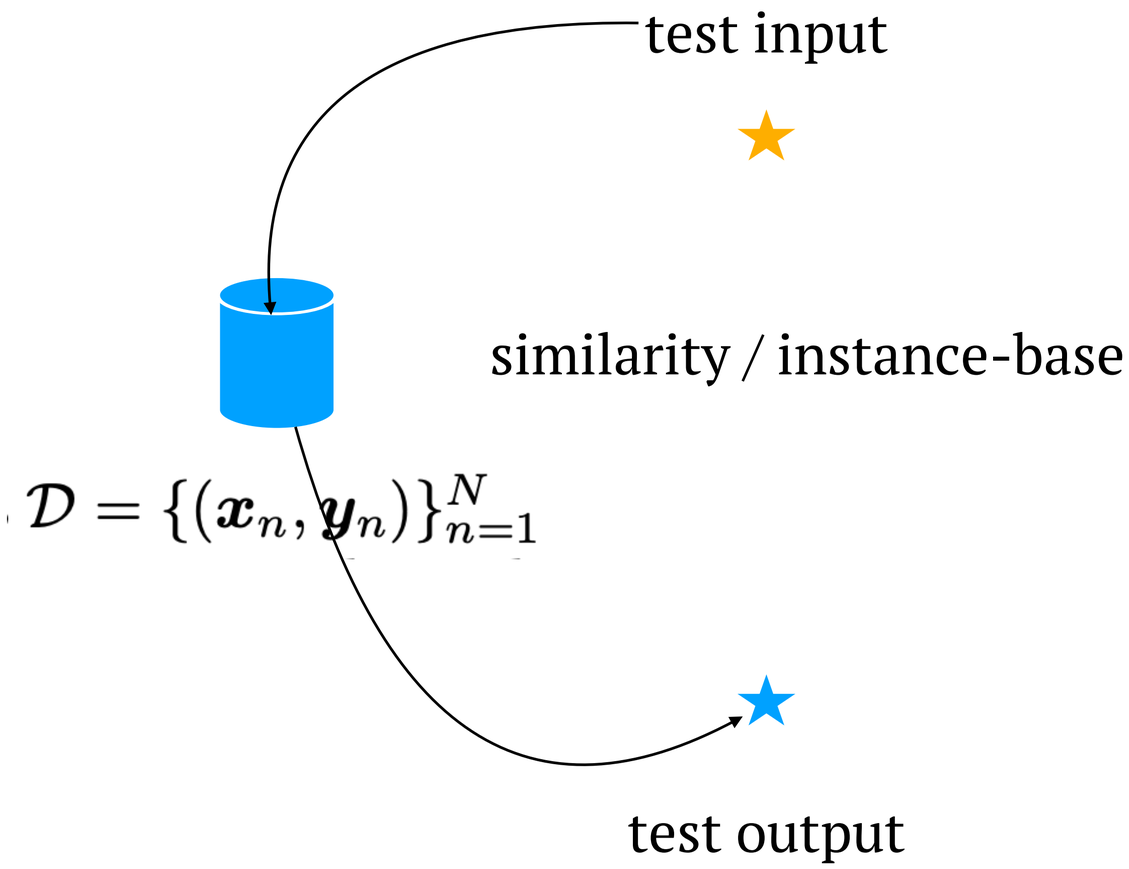
Lingpeng Kong
Department of Computer Science, The University of Hong Kong

Based on: Probabilistic Machine Learning by Kevin Murphy
Slides from: Saw Shier Nee with special thanks!

Parametric Models v.s. Nonparametric models

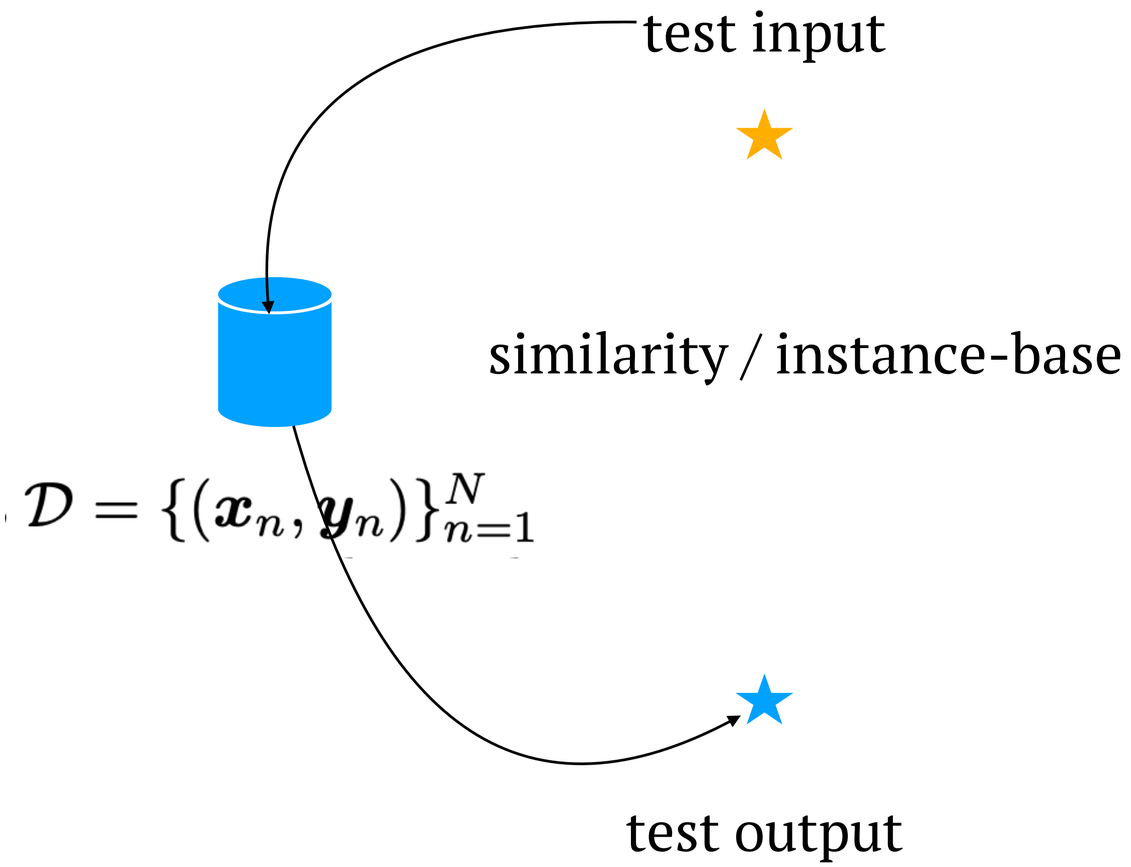


parameteric models



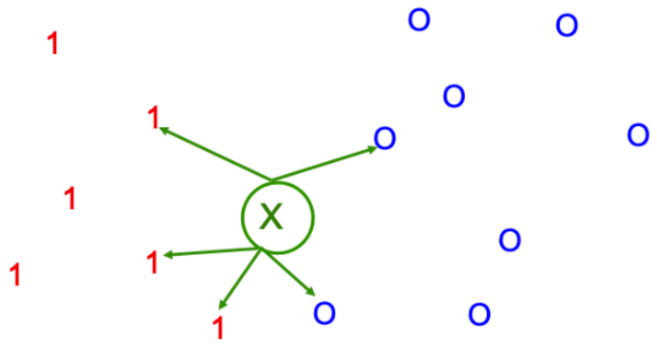
nonparameteric models

K Nearest Neighbor (KNN) Classification



$$p(y = c|\boldsymbol{x}, \mathcal{D}) = \frac{1}{K} \sum_{n \in N_K(\boldsymbol{x}, \mathcal{D})} \mathbb{I}(y_n = c)$$

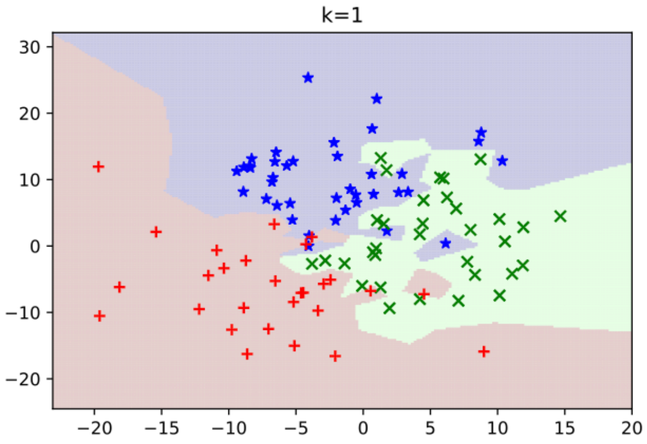
K $d(\boldsymbol{x}, \boldsymbol{x'})$



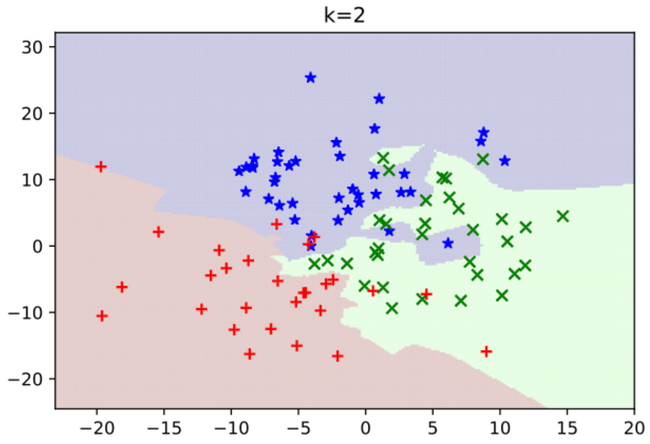
$$d_{\mathbf{M}}(\boldsymbol{x}, \boldsymbol{\mu}) = \sqrt{(\boldsymbol{x} - \boldsymbol{\mu})^{\mathsf{T}} \mathbf{M} (\boldsymbol{x} - \boldsymbol{\mu})}$$

Mahalanobis distance

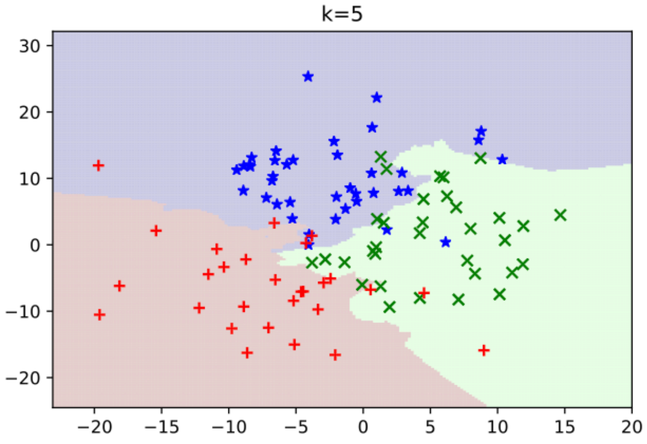
Overfitting in KNN



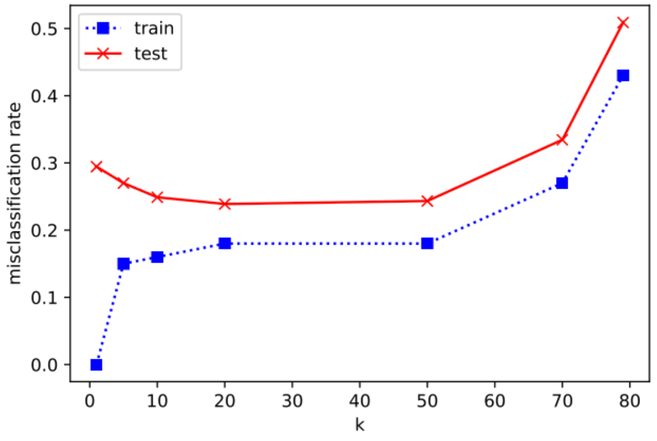
(a)



(b)



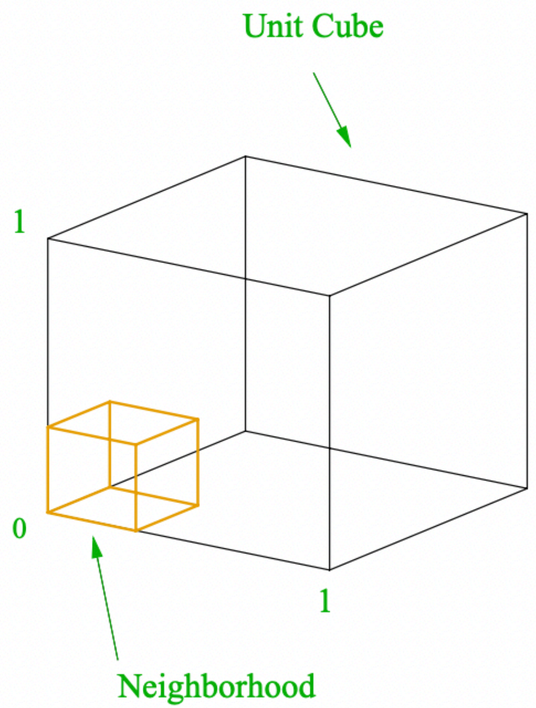
(c)

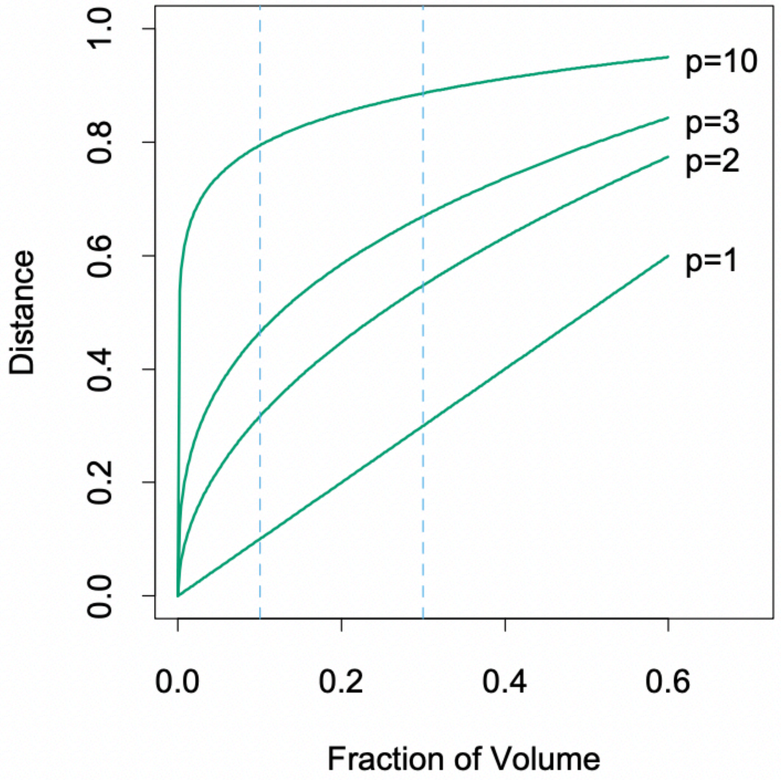


(d)

Which one is more likely to overfit, smaller K or larger K? Why?

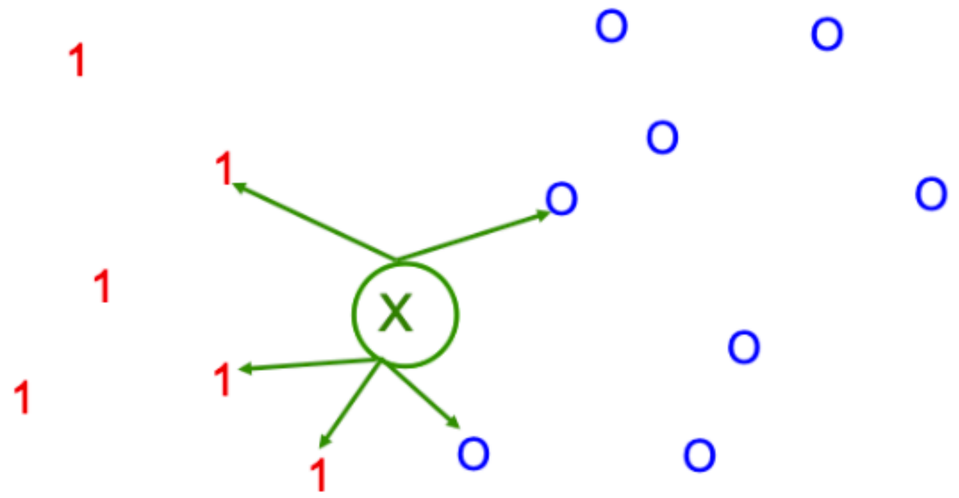
The Curse of Dimensionality





The curse of dimensionality is well illustrated by a subcubical neighborhood for uniform data in a unit cube. The figure on the right shows the side-length of the subcube needed to capture a fraction r of the volume of the data, for different dimensions p. In ten dimensions we need to cover 80% of the range of each coordinate to capture 10% of the data.

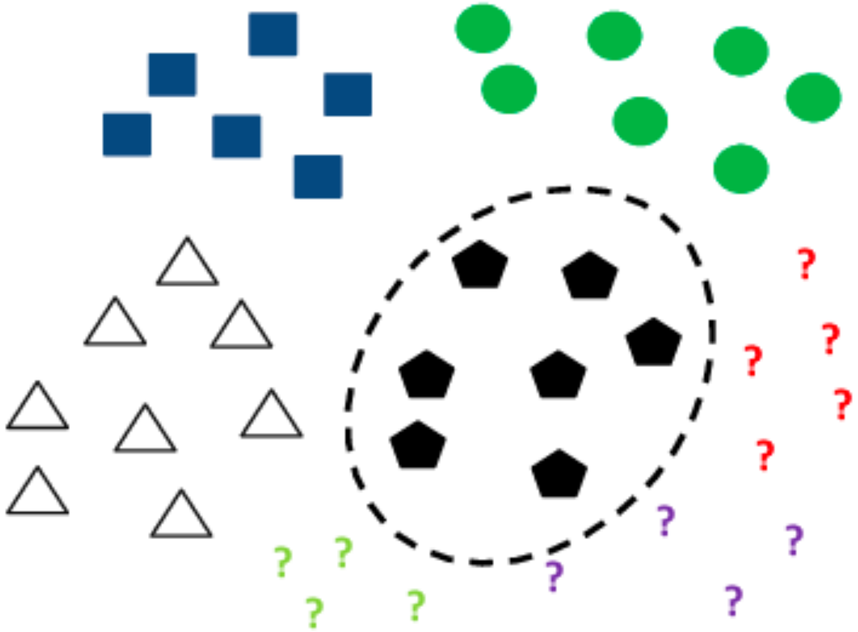
Efficient Speed and Memory Methods

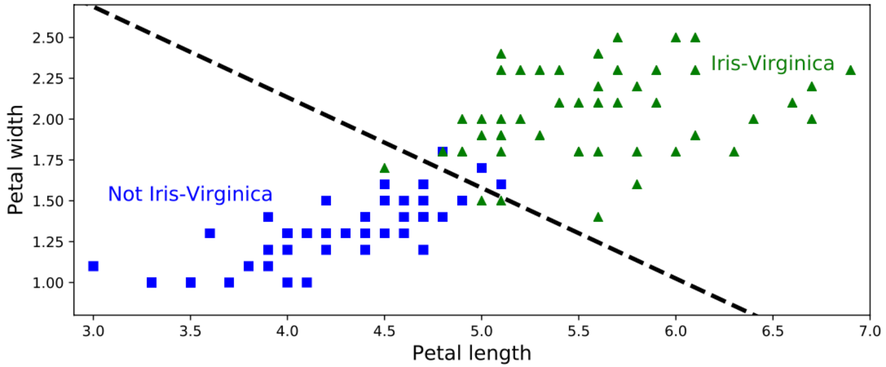


(1) Store only points that affect the decision boundaries.

(2) Hashing method, e.g., Locality-sensitive hashing.

Open Set Recognition





$$f(\boldsymbol{x};\boldsymbol{\theta}) = b + \boldsymbol{w}^{\mathsf{T}}\boldsymbol{x} = b + \sum_{d=1}^{D} w_d x_d$$

$$\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^{N}$$

Open world / novelty detection / lief-long learning

Closed world assumption

Learning Distance Metrics

$$d_{\mathbf{M}}(\boldsymbol{x}, \boldsymbol{\mu}) = \sqrt{(\boldsymbol{x} - \boldsymbol{\mu})^{\mathsf{T}} \mathbf{M} (\boldsymbol{x} - \boldsymbol{\mu})}$$

$M = I$ Euclidean Distance

metrics learning

$\boldsymbol{e} = f(\boldsymbol{x})$ When f is a DNN — deep metric learning

Large margin nearest neighbors

$$\mathcal{L}_{\text{pull}}(\mathbf{M}) = \sum_{i=1}^{N} \sum_{j \in N_i} d_{\mathbf{M}}(\boldsymbol{x}_i, \boldsymbol{x}_j)^2$$

$$\mathcal{L}_{\text{push}}(\mathbf{M}) = \sum_{i=1}^{N} \sum_{j \in N_i} \sum_{l=1}^{N} \mathbb{I}\left(y_i \neq y_l\right) \left[m + d_{\mathbf{M}}(\boldsymbol{x}_i, \boldsymbol{x}_j)^2 - d_{\mathbf{M}}(\boldsymbol{x}_i, \boldsymbol{x}_l)^2\right]_+$$

semidefinite programming

Deep Metric Learning

$$\boldsymbol{e} = f(\boldsymbol{x}; \boldsymbol{\theta}) \in \mathbb{R}^L$$

learn an embedding to a lower dimensional "semantic" space, suffer less from the "curse of dimensionality"

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta}) = ||\hat{\boldsymbol{e}}_i - \hat{\boldsymbol{e}}_j||_2^2$$

normalized Euclidean distance

$\hat{\boldsymbol{e}} = \boldsymbol{e}/||\boldsymbol{e}||_2$

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta}) = \hat{\boldsymbol{e}}_i^\top \hat{\boldsymbol{e}}_j$$

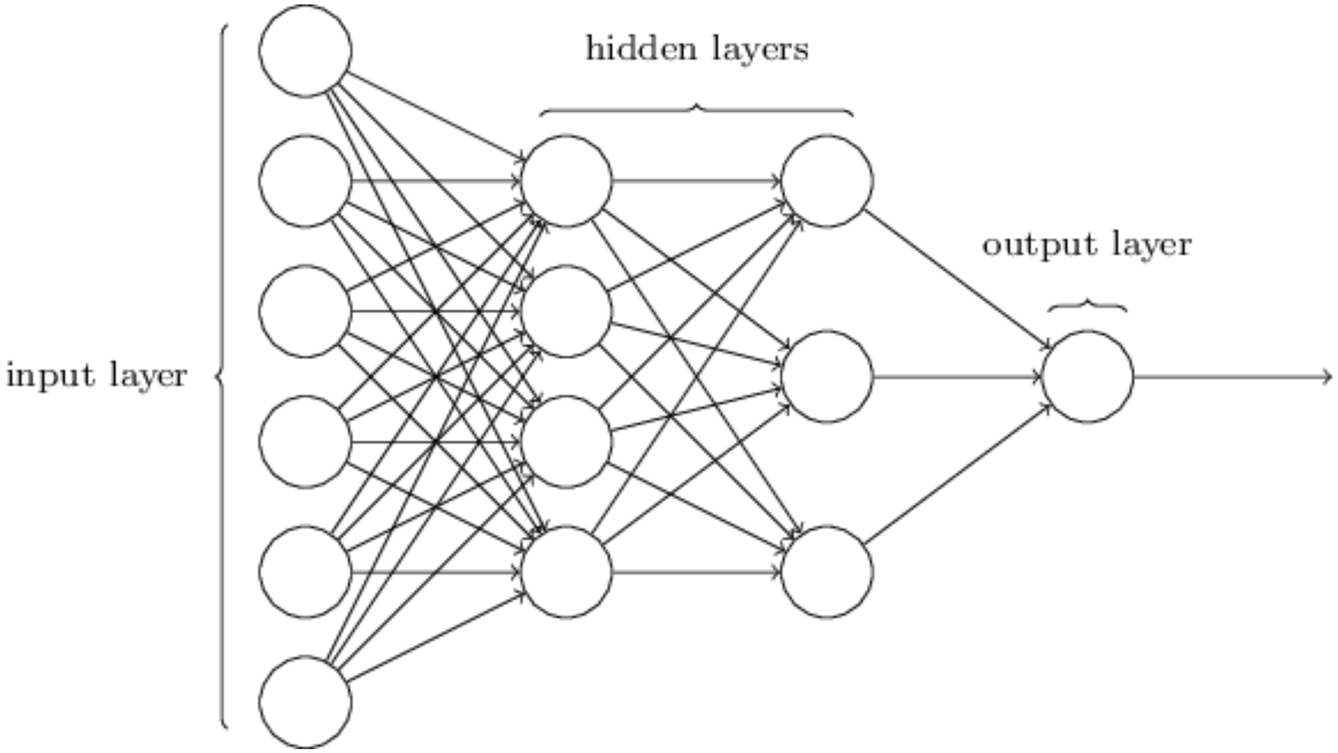
confine similarity

$$\mathcal{S} = \{(i, j) : y_i = y_j\}$$

i,j closer, i, k far away.

Classification Losses

$$\boldsymbol{e} = f(\boldsymbol{x}; \boldsymbol{\theta}) \in \mathbb{R}^L$$



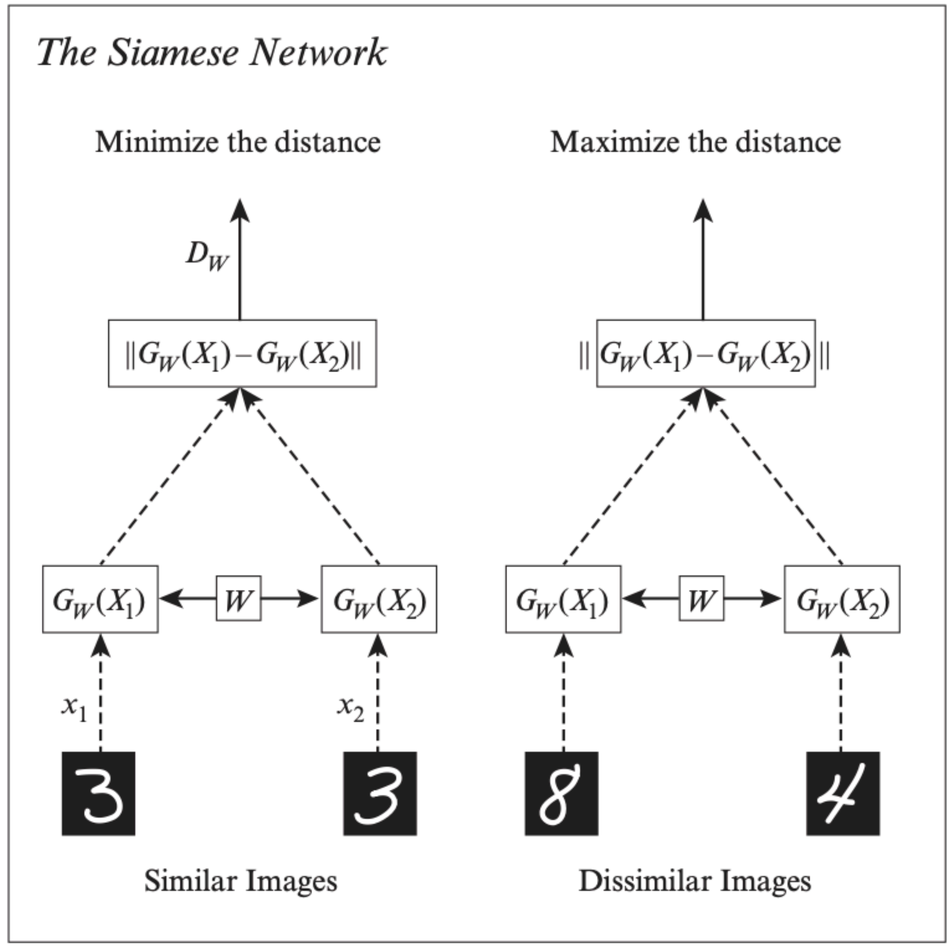
Learn a classifier and use the (second-to-last) layer as an embedding function.

Ranking Losses

$$\boldsymbol{e} = f(\boldsymbol{x}; \boldsymbol{\theta}) \in \mathbb{R}^L$$

Pairwise (contrastive) loss

$$\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{x}_i, \boldsymbol{x}_j) = \mathbb{I}(y_i = y_j)\, d(\boldsymbol{x}_i, \boldsymbol{x}_j)^2 + \mathbb{I}(y_i \neq y_j)\, [m - d(\boldsymbol{x}_i, \boldsymbol{x}_j)]_+^2$$

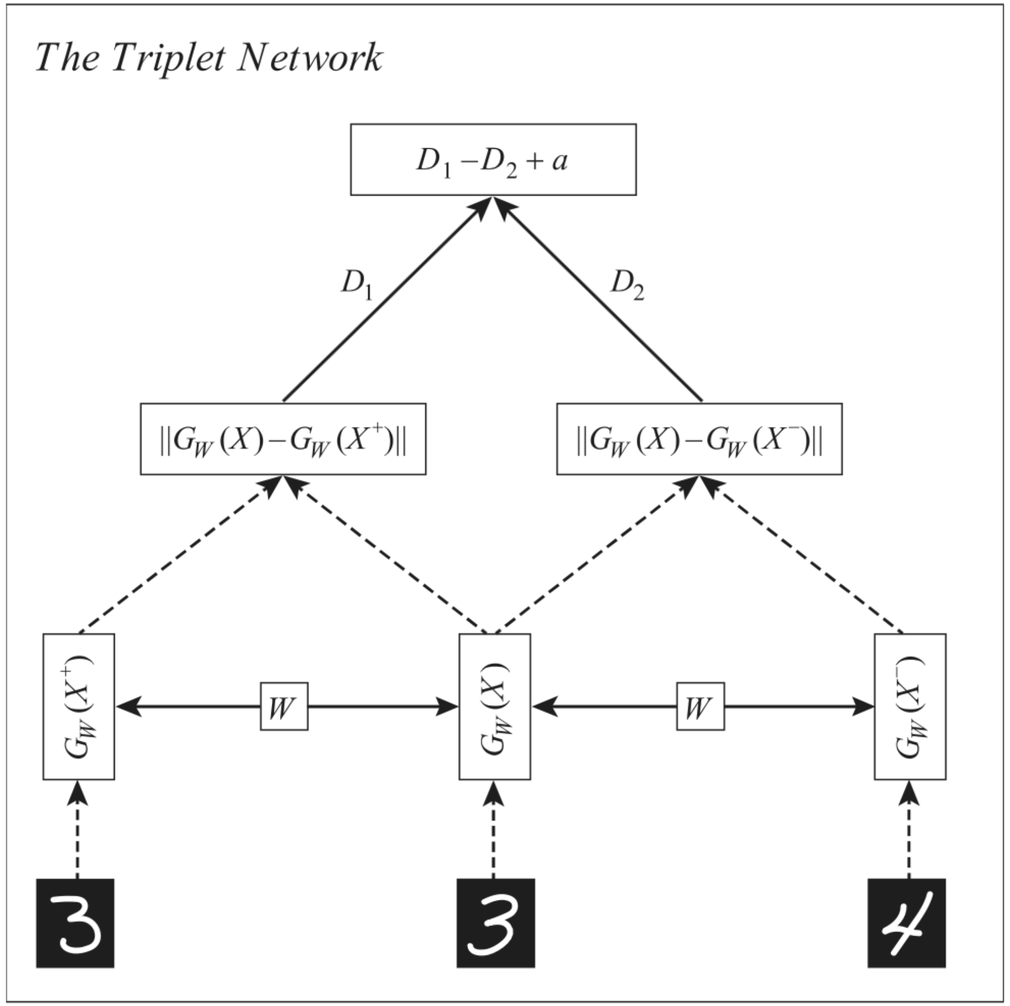


Triple Losses

$$\boldsymbol{e} = f(\boldsymbol{x}; \boldsymbol{\theta}) \in \mathbb{R}^L$$

optimization of the positive pairs is independent of the negative pairs, which can make their magnitudes incomparable

$$\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{x}_i, \boldsymbol{x}_i^+, \boldsymbol{x}_i^-) = [d_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_i^+)^2 - d_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_i^-)^2 + m]_+$$



N-pairs Loss (InfoNCE)

$$\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{x}_i, \boldsymbol{x}_j^+, \{\boldsymbol{x}_k^-\}_{k=1}^{N-1}) = \log \left(1 + \sum_{k=1}^{N-1} \exp(\hat{\boldsymbol{e}}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)^\top \hat{\boldsymbol{e}}_{\boldsymbol{\theta}}(\boldsymbol{x}_k^-) - \hat{\boldsymbol{e}}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)^\top \hat{\boldsymbol{e}}_{\boldsymbol{\theta}}(\boldsymbol{x}_j^+)) \right)$$

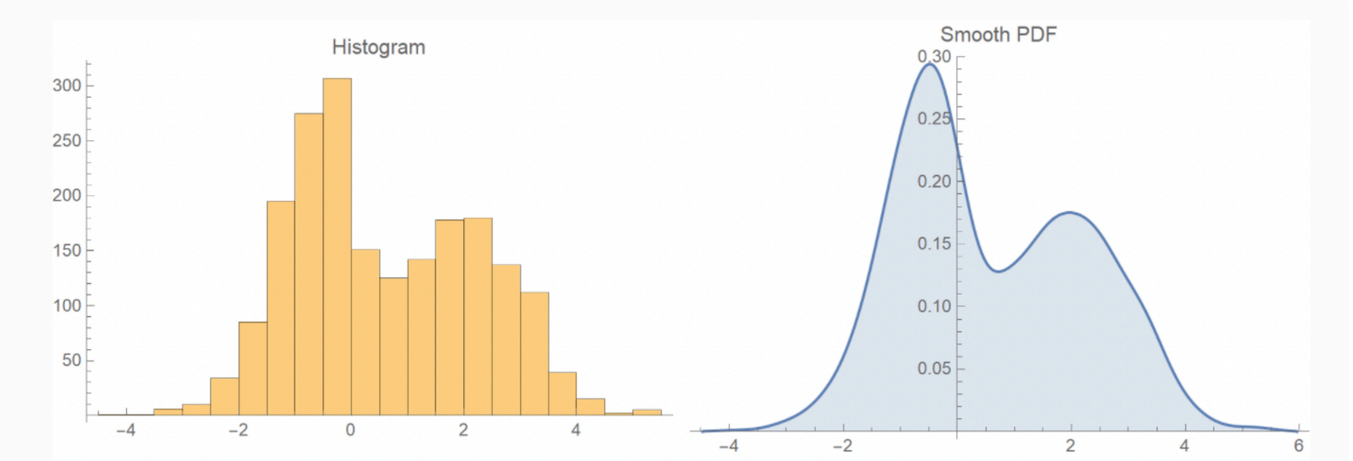
$$= \log \frac{\exp(\hat{\boldsymbol{e}}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)^\top \hat{\boldsymbol{e}}_{\boldsymbol{\theta}}(\boldsymbol{x}_j^+))}{\exp(\hat{\boldsymbol{e}}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)^\top \hat{\boldsymbol{e}}_{\boldsymbol{\theta}}(\boldsymbol{x}_j^+)) + \sum_{k=1}^{N-1} \exp(\hat{\boldsymbol{e}}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)^\top \hat{\boldsymbol{e}}_{\boldsymbol{\theta}}(\boldsymbol{x}_k^-))}$$

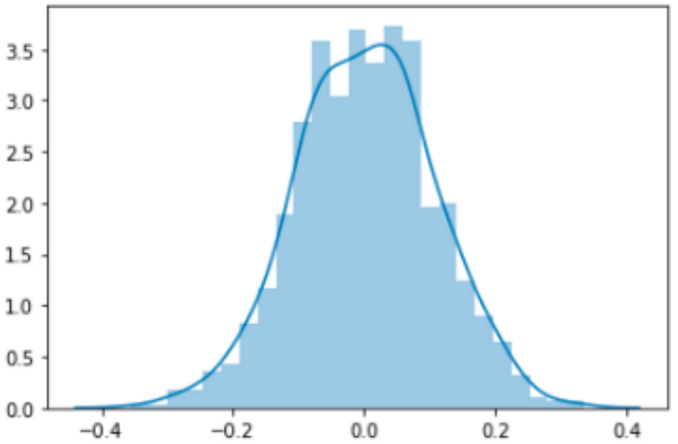
Softmax function:

$$\text{softmax}(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ for j = 1,...,K}$$

Kernel Density Estimation (KDE)

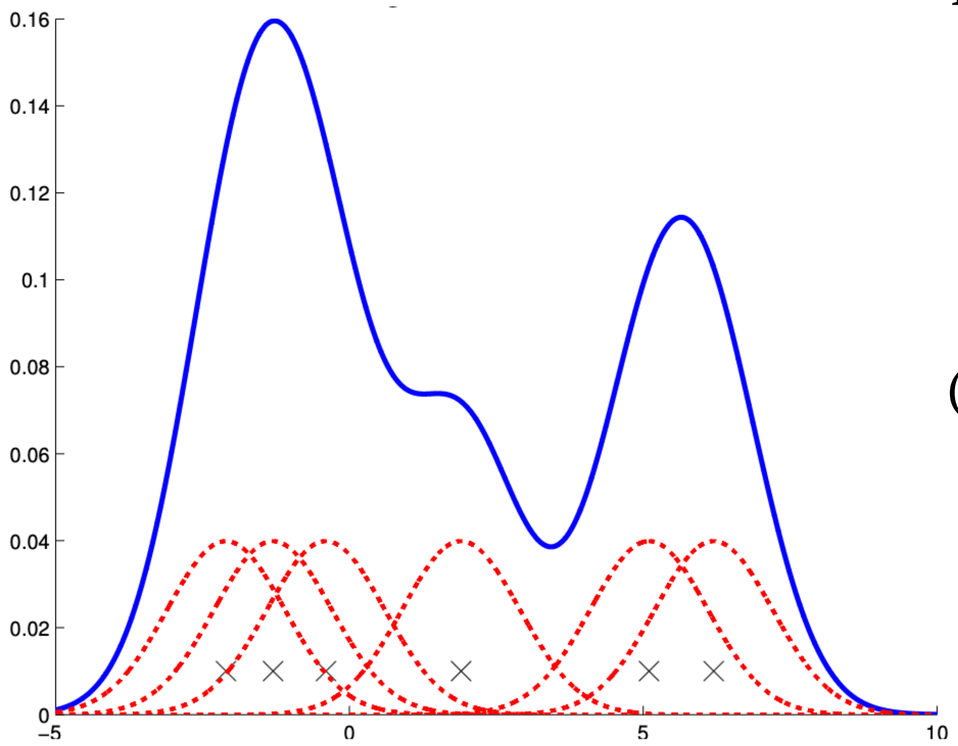
Density Estimation





Estimate $p(\boldsymbol{x})$ from seen data points

Kernel Density Estimation (KDE)



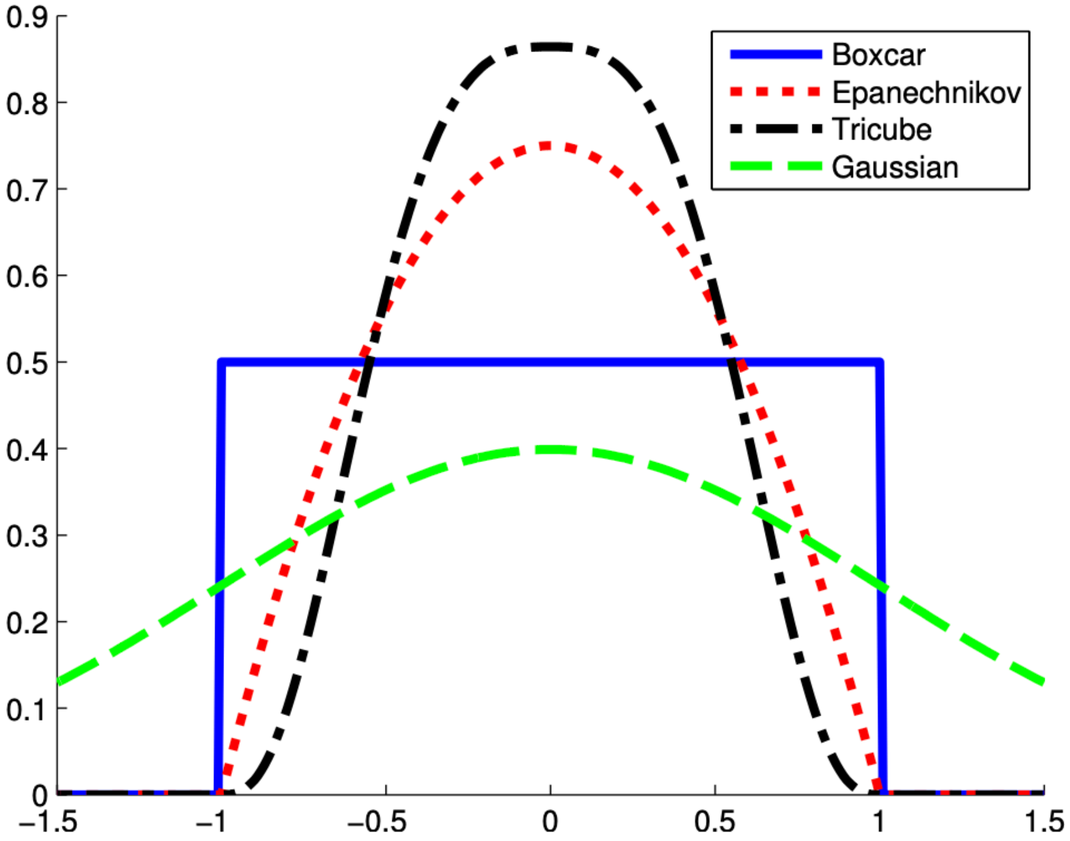
Parzen window density estimator:

$$p(\boldsymbol{x}|\mathcal{D}) = \frac{1}{N}\sum_{n=1}^{N}\mathcal{K}_h\left(\boldsymbol{x}-\boldsymbol{x}_n\right)$$

(In comparison) Gaussian mixture model:

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \frac{1}{K}\sum_{k=1}^{K}\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \sigma^2\mathbf{I})$$

Density Kernels

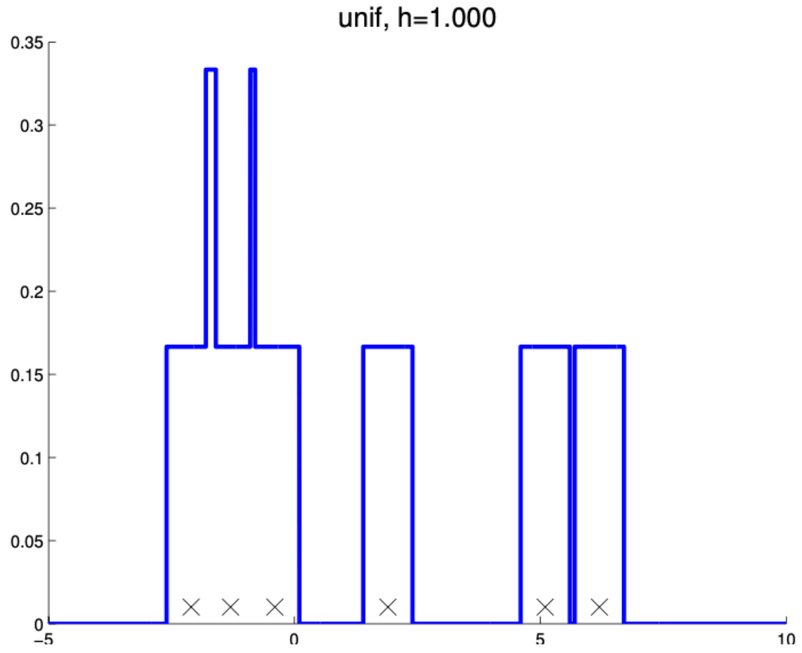


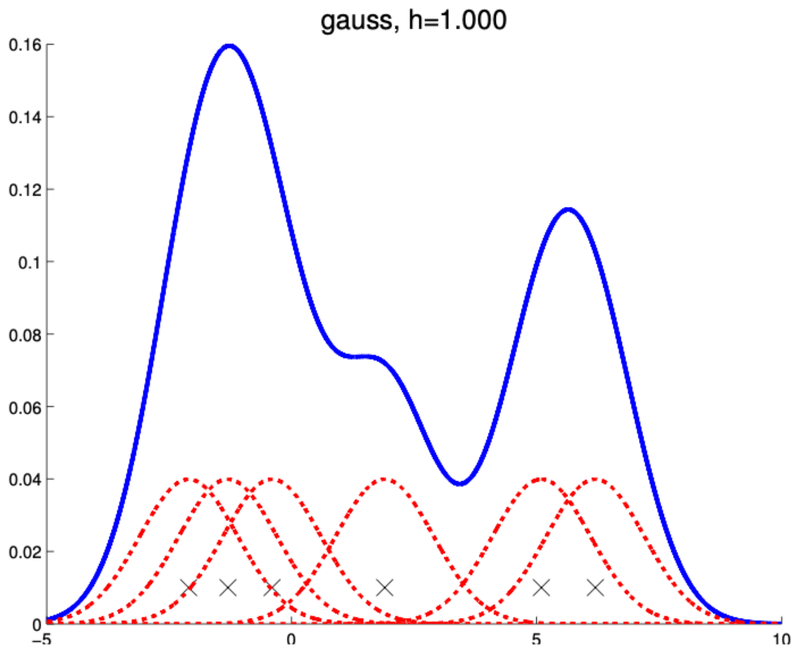
$$\mathcal{K} : \mathbb{R} \to \mathbb{R}_+$$

$$\int \mathcal{K}(x)dx = 1$$

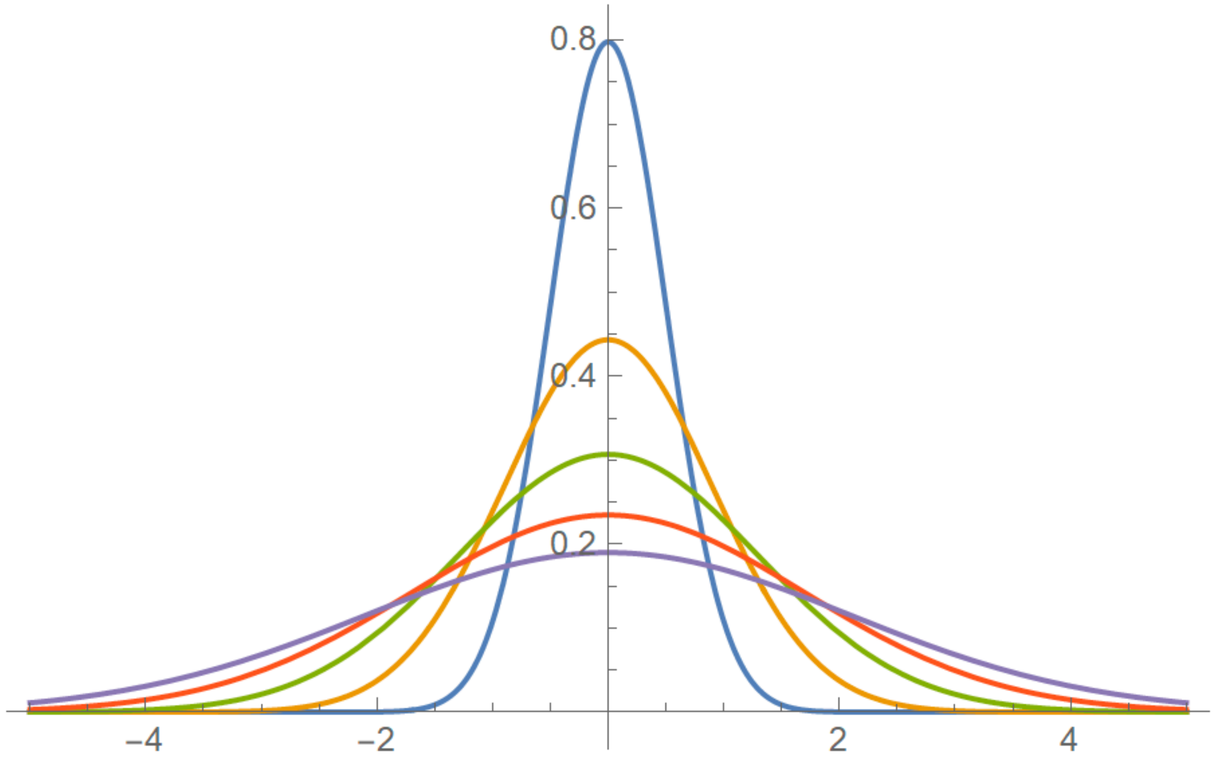
$$\mathcal{K}(-x) = \mathcal{K}(x)$$

Nonparametric (Parzen) Density Estimator





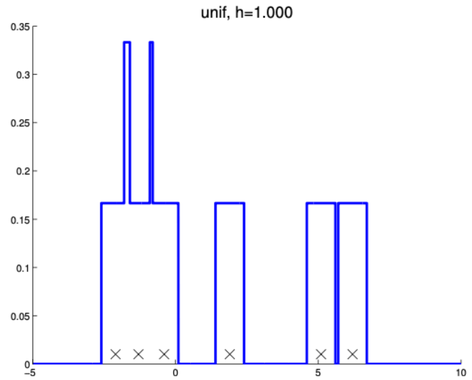
Bandwidth



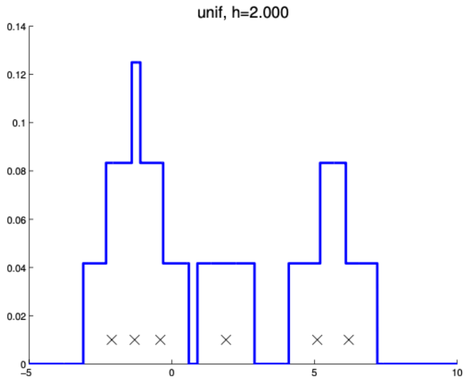
$$\mathcal{K}(x) = \frac{1}{(2\pi)^{\frac{1}{2}}} e^{-x^2/2}$$

$$\mathcal{K}_h(x) \triangleq \frac{1}{h}\mathcal{K}(\frac{x}{h})$$

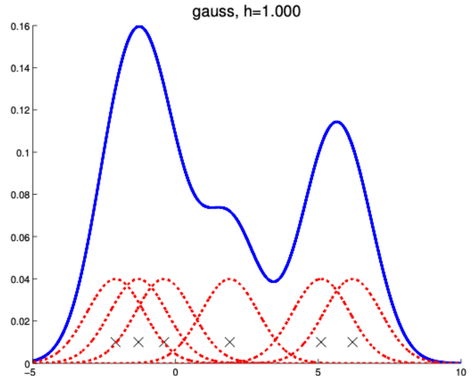
How to Choose Bandwidth



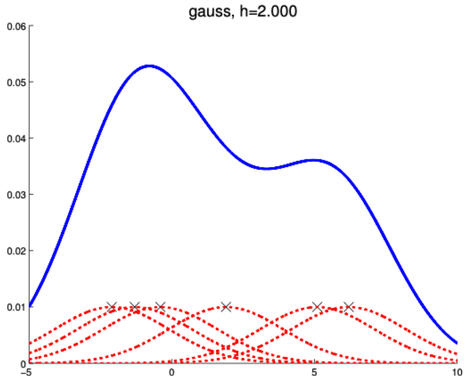
(a)



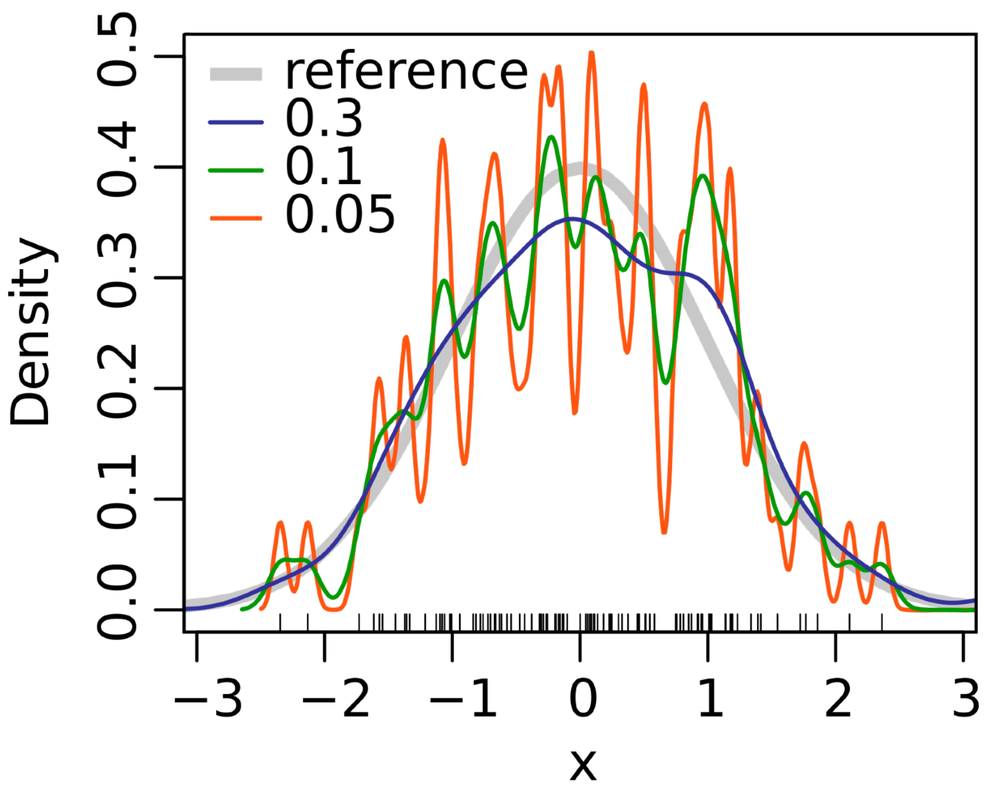
(b)



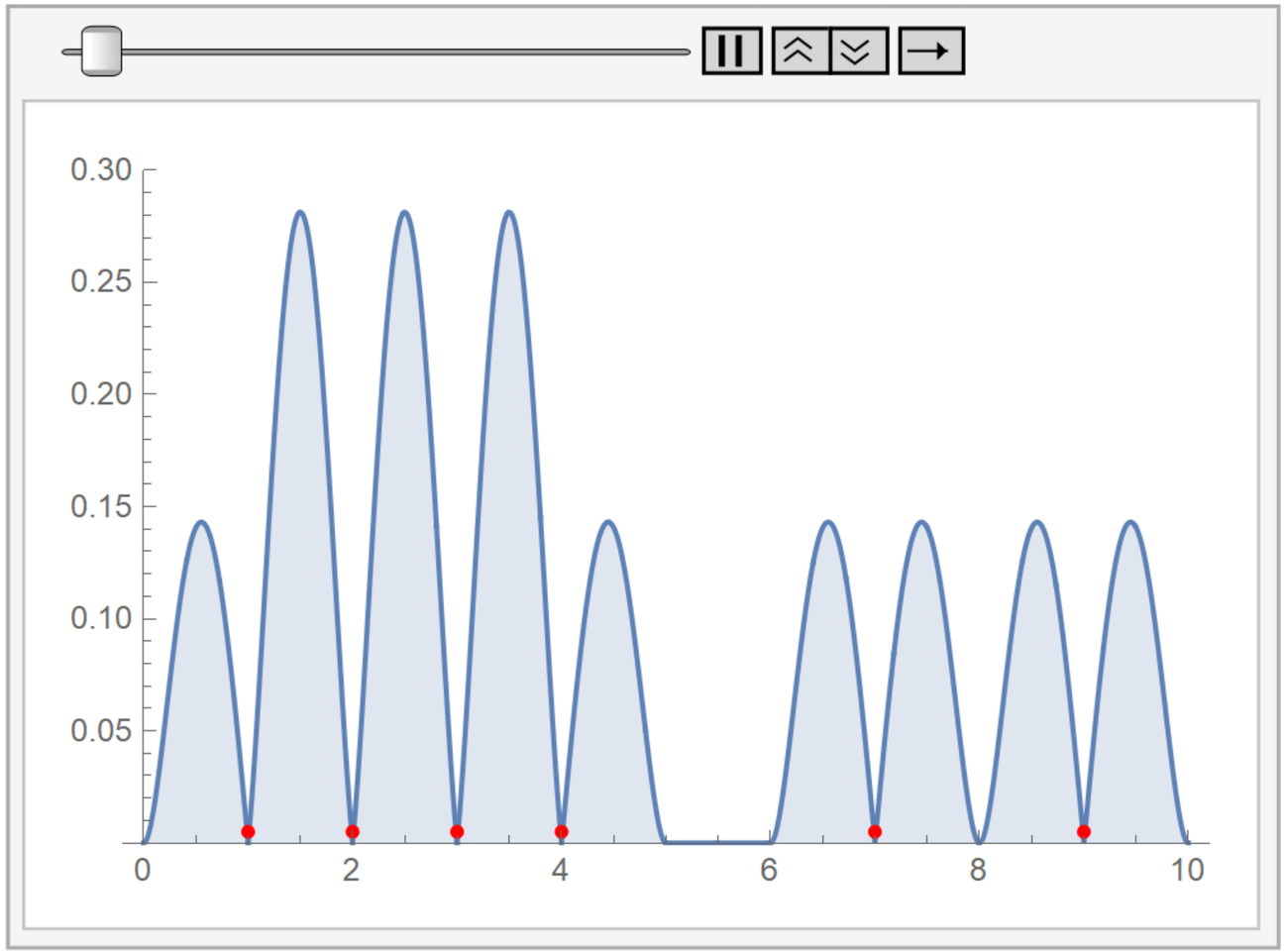
(c)



(d)



How to Choose Bandwidth



https://ekamperi.github.io/math/2020/12/08/kernel-density-estimation.html