Machine Intelligence 1

1.5 Radial Basis Function Networks

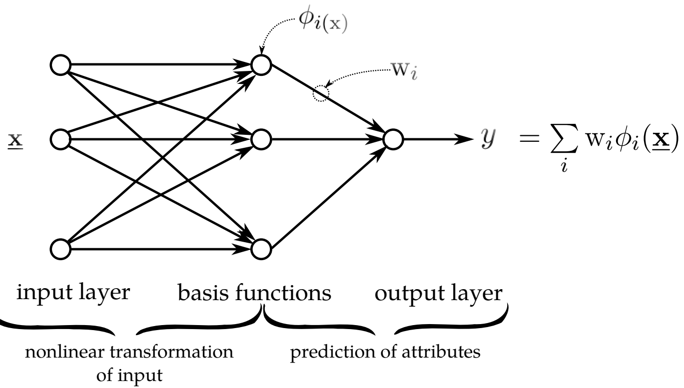
Prof. Dr. Klaus Obermayer

Fachgebiet Neuronale Informationsverarbeitung (NI)

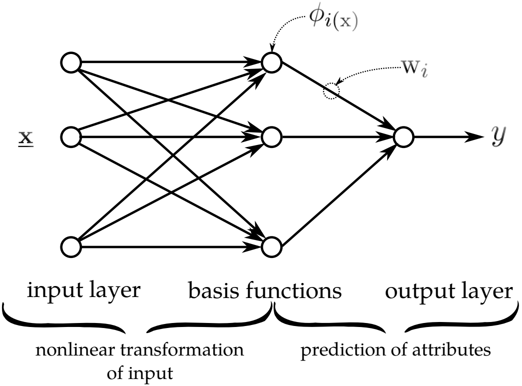
WS 2016/2017

1.5.1 Network Architecture

Network architecture



Network architecture



$$y = \sum_i \mathrm{w}_i \phi_i(\underline{\mathbf{x}})$$

General principle

- two layered network $\rightarrow$ expansion into basis functions /features
- sine-waves (Fourier), polynomials (Taylor), sigmoid functions (MLP)

Radial basis functions (RBF)

- **Distance dependent basis functions**

$$\phi_i(\underline{\mathbf{x}}) = \widetilde{\phi}_i(D[\underline{\mathbf{x}}, \underline{\mathbf{t}}_i])$$

where $\underline{\mathbf{t}}_i$ are parameters specifying the location of the i-th basis function

Radial basis functions (RBF)

- **Distance dependent basis functions**

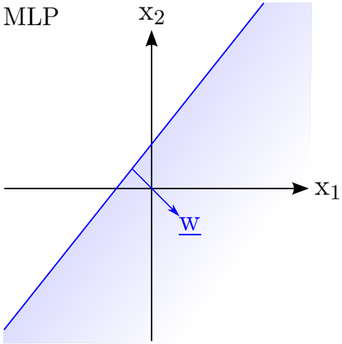
$$\phi_i(\underline{\mathbf{x}}) \quad = \quad \widetilde{\phi}_i(D[\underline{\mathbf{x}}, \underline{\mathbf{t}}_i])$$

  where $\underline{\mathbf{t}}_i$ are parameters specifying the location of the i-th basis function

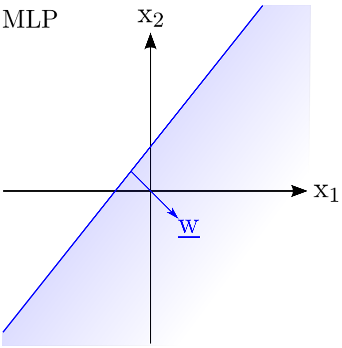
- **Common choice:** Gaussian functions

$$\phi_i(\underline{\mathbf{x}}) \quad \propto \quad \exp\left(- \frac{\|\underline{\mathbf{x}} - \underline{\mathbf{t}}_i\|^2}{2\sigma_i^2} \right)$$

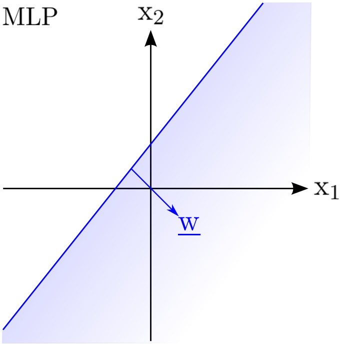
MLP vs. RBF

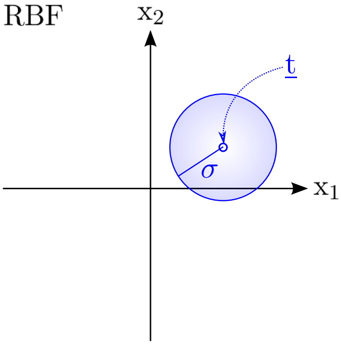


MLP vs. RBF



MLP vs. RBF

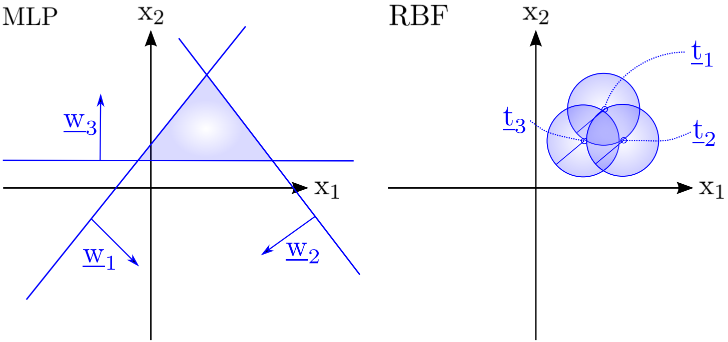




$\Rightarrow$ "discriminative features"

$\Rightarrow$ "categories/classes"

MLP vs. RBF


MLP
x_2
x_1
$\underline{w}_3$
$\underline{w}_1$
$\underline{w}_2$
RBF
x_2
x_1
$\underline{t}_1$
$\underline{t}_2$
$\underline{t}_3$

RBFs: pro & contra

fast convergence during learning

$\rightarrow$ few parameters have to be changed per training point

$\rightarrow$ "credit assignment" is simple

RBFs: pro & contra

fast convergence during learning

$\rightarrow$ few parameters have to be changed per training point

$\rightarrow$ "credit assignment" is simple

"curse of dimensionality"

complete coverage of input space requires $\sim n^d$ basis functions
(d: dimension, n: no. of basis functions along one dimension)
$d = 20, n = 10 \leadsto 10^{20}$ basis functions

RBFs: pro & contra

fast convergence during learning

$\rightarrow$ few parameters have to be changed per training point

$\rightarrow$ "credit assignment" is simple

"curse of dimensionality"

complete coverage of input space requires $\sim n^d$ basis functions
(d: dimension, n: no. of basis functions along one dimension)
$d = 20, n = 10 \rightsquigarrow 10^{20}$ basis functions

$\Rightarrow$ **RBF-networks are useful for**

- low dimensional data or
- datasets with a pronounced cluster structure

1.5.2 Model Selection – Learning

Problem setting & model class

Regression: Real-valued targets

$$\left\{\left(\underline{\mathbf{x}}^{(\alpha)}, y_T^{(\alpha)}\right)\right\}, \quad \alpha \in \{1, \ldots, p\}, \qquad \underline{\mathbf{x}} \in \mathbb{R}^d, \quad y_T \in \mathbb{R}$$

Problem setting & model class

Regression: Real-valued targets

$$\left\{ \left(\underline{\mathbf{x}}^{(\alpha)}, y_T^{(\alpha)} \right) \right\}, \quad \alpha \in \{1, \ldots, p\}, \qquad \underline{\mathbf{x}} \in \mathbb{R}^d, \quad y_T \in \mathbb{R}$$

Model class:

$$y_{(\underline{\mathbf{x}})} = \sum_{i=1}^{M} \mathrm{w}_i \exp\left(- \frac{\|\underline{\mathbf{x}} - \underline{\mathbf{t}}_i\|^2}{2\sigma_i^2} \right)$$

1. $\underline{\mathbf{t}}_i$: centroids of basis functions
2. σ_i: range of basis functions
3. w_i: weights of the second layer

Model selection / learning

1. $\underline{\mathbf{t}}_i$: determination of centroids — unsupervised
2. σ_i: range of basis functions — heuristics
3. w_i: weights of the second layer — supervised

$\Rightarrow$ **2-Step Learning Procedure:** RBFs $\rightarrow$ weights

Model selection / learning

1. $\underline{\mathbf{t}}_i$: determination of centroids — unsupervised
2. σ_i: range of basis functions — heuristics
3. w_i: weights of the second layer — supervised

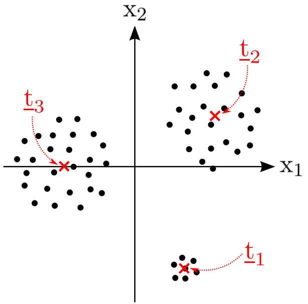
$\Rightarrow$ **2-Step Learning Procedure:** RBFs $\rightarrow$ weights

Alternative: supervised learning of all parameters

But: non-convex problem with local minima

Determination of centroids $\underline{\mathbf{t}}_i$

k-means clustering (online)

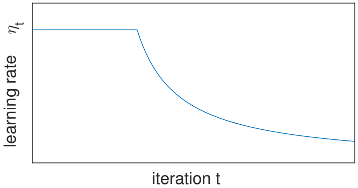


Initialize $\underline{\mathbf{t}}_i$
BEGIN loop

1. Choose data point $\underline{\mathbf{x}}^{(\alpha)}$
2. Closest centroid $\underline{\mathbf{t}}_i$: $i = \operatorname{argmin}_j \left|\underline{\mathbf{t}}_j - \underline{\mathbf{x}}^{(\alpha)}\right|$
3. Update $\underline{\mathbf{t}}_i$ as: $\Delta \underline{\mathbf{t}}_i = \eta_t \left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{t}}_i\right)$

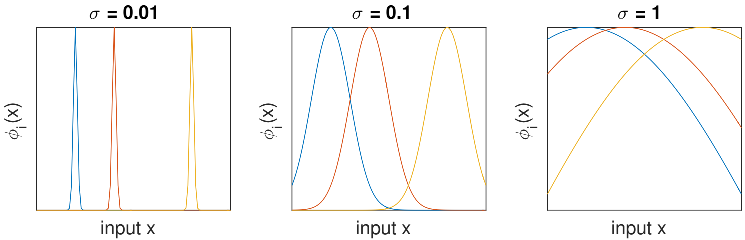
END loop

- Adaptive learning rate
  - first constant $\eta_t = \eta_0$
  - then decaying $\eta_t = \frac{\eta_0}{t}$



Overfitting vs. underfitting

- Gaussian "variances" σ_i determine overfitting

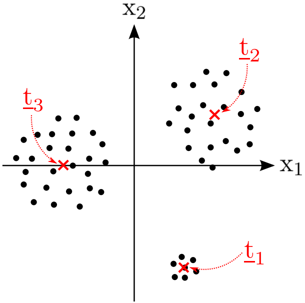


- good σ_i yield basis functions ϕ_i that
  - are sufficiently *different* to their neighbors
  - *overlap* with neighboring basis functions

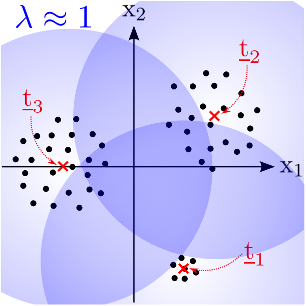
Determination of variances σ_i

Goal: sufficient overlap between neighboring basis functions

Problem: coverage vs. resolution payoff



Determination of variances σ_i



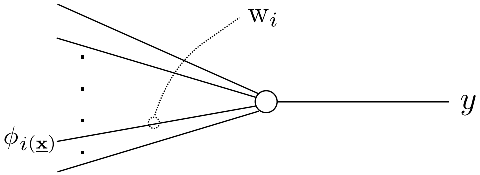
Goal: sufficient overlap between neighboring basis functions

Problem: coverage vs. resolution payoff

Heuristic

$$\sigma_i \quad = \quad \lambda \min_{j \neq i} \left\| \underline{\mathbf{t}}_i - \underline{\mathbf{t}}_j \right\|, \qquad \lambda \approx 2$$

Determination of output weights w_i



(linear)
connectionist neuron

Cost function: quadratic error

$$E^T = \frac{1}{2p} \sum_{\alpha=1}^{p} \left(y_T^{(\alpha)} - \sum_{i=1}^{M} \mathrm{w}_i \underbrace{\phi_{i(\underline{\mathbf{x}}^{(\alpha)})}}_{:=\phi_i^{(\alpha)}} \right)^2$$

Optimization (1)

$$\frac{\partial E^T}{\partial \mathrm{w}_k} = -\frac{1}{p} \sum_{\alpha=1}^{p} \left(y_T^{(\alpha)} - \sum_{i=1}^{M} \mathrm{w}_i \phi_i^{(\alpha)} \right) \phi_k^{(\alpha)} \overset{!}{=} 0$$

Optimization (1)

$$\frac{\partial E^T}{\partial \mathrm{w}_k} = -\frac{1}{p} \sum_{\alpha=1}^{p} \left(y_T^{(\alpha)} - \sum_{i=1}^{M} \mathrm{w}_i \phi_i^{(\alpha)} \right) \phi_k^{(\alpha)} \stackrel{!}{=} 0$$

$$\sum_{i=1}^{M} \left(\sum_{\alpha=1}^{p} \phi_k^{(\alpha)} \phi_i^{(\alpha)} \right) \mathrm{w}_i = \sum_{\alpha=1}^{p} \phi_k^{(\alpha)} y_T^{(\alpha)}$$

Optimization (1)

$$\frac{\partial E^T}{\partial \mathrm{w}_k} = -\frac{1}{p}\sum_{\alpha=1}^{p}\left(y_T^{(\alpha)} - \sum_{i=1}^{M}\mathrm{w}_i\phi_i^{(\alpha)}\right)\phi_k^{(\alpha)} \stackrel{!}{=} 0$$

$$\sum_{i=1}^{M}\left(\sum_{\alpha=1}^{p}\phi_k^{(\alpha)}\phi_i^{(\alpha)}\right)\mathrm{w}_i = \sum_{\alpha=1}^{p}\phi_k^{(\alpha)}y_T^{(\alpha)}$$

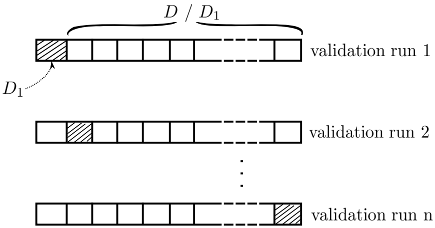
Matrix Notation

$$\underbrace{(\underline{\mathbf{\Phi}}^\top\underline{\mathbf{\Phi}})}_{\text{known}}\underline{\mathbf{w}} = \underbrace{\underline{\mathbf{\Phi}}^\top\underline{\mathbf{y}}_T}_{\text{known}} \quad\Rightarrow\quad \underline{\mathbf{w}} = \underbrace{(\underline{\mathbf{\Phi}}^\top\underline{\mathbf{\Phi}})^{-1}}_{\text{if invertible}}\underline{\mathbf{\Phi}}^\top\underline{\mathbf{y}}_T$$

$$\begin{aligned}
\underline{\mathbf{\Phi}} &= \{\phi_k^{(\alpha)}\} && p \text{ x } M \text{ matrix}\\
\underline{\mathbf{w}} &= \{\mathrm{w}_i\} && M \text{ vector}\\
\underline{\mathbf{y}}_T &= \{y_T^{(\alpha)}\} && p \text{ vector}
\end{aligned}$$

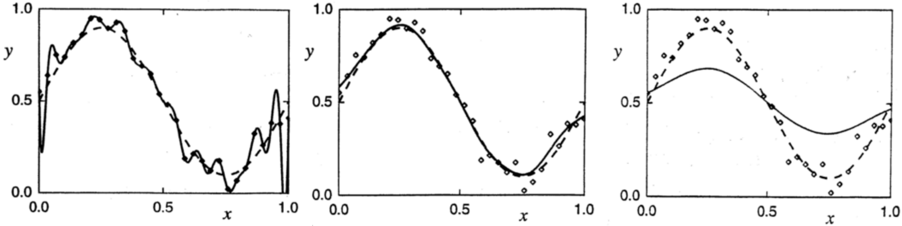
Validation

- test-set method analog to MLP
- use n-fold cross-validation to estimate $\widehat{E}^G = \frac{1}{p} \sum_j \sum_{\alpha \in D_j} e^{(\alpha)}$.
  - training with D/D_i includes all three model selection steps



- use nested n-fold cross-validation to determine parameters

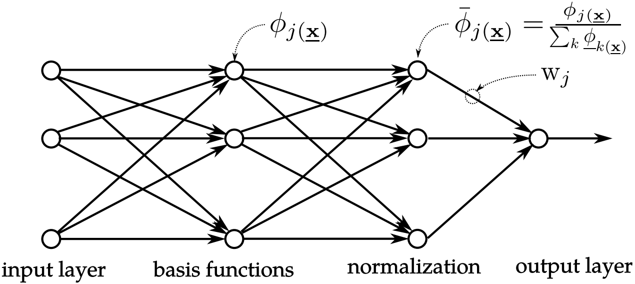
Comment: number of basis functions

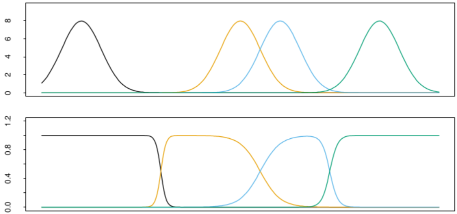


- too many basis functions ⇒ over-fitting
- too few basis functions ⇒ under-fitting

⇒ determine number by nested n-fold cross validation

Comment: normalization layer





Comment: regularization

- Matrix $\underline{\mathbf{\Phi}}^\top \underline{\mathbf{\Phi}}$ often not invertible!
- Add *ridge regression* term $\lambda E^R_{[\underline{\mathbf{w}}]} = \lambda \|\underline{\mathbf{w}}\|^2$ to cost function
  - regularized solution $\underline{\mathbf{w}} = \underbrace{\left(\underline{\mathbf{\Phi}}^\top \underline{\mathbf{\Phi}} + \lambda \underline{\mathbf{I}}\right)^{-1}}_{\text{invertible for } \lambda > 0} \underline{\mathbf{\Phi}}^\top \underline{\mathbf{y}}_T$
  - larger λ yield *smoother* functions

$\Rightarrow$ Use nested n-fold cross validation to determine λ

Comment: two-step procedure vs. gradient descend

The two-step procedure...

- ...is much faster
- ...has usually equal performance
- ...can use additional unlabeled data

1.5.3 RBF-networks and Regularization

General model classes

General learning problem

observations: $\left\{ \left(\underline{\mathbf{x}}^{(\alpha)}, y_T^{(\alpha)} \right) \right\}, \quad \alpha \in \{1, \ldots, p\}$

model class: all continuous and differentiable functions $y_{(\underline{\mathbf{x}})}$

cost function: $E^T = \frac{1}{2p} \sum_{\alpha=1}^{p} \left(y(\underline{\mathbf{x}}^{(\alpha)}) - y_T^{(\alpha)} \right)^2$

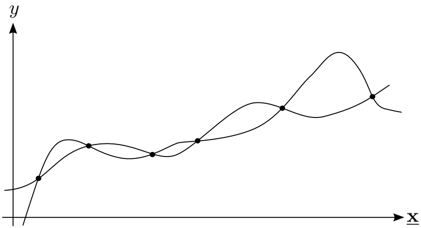
General model classes

General learning problem

observations: $\left\{ \left(\underline{\mathbf{x}}^{(\alpha)}, y_T^{(\alpha)} \right) \right\}, \quad \alpha \in \{1, \dots, p\}$

model class: all continuous and differentiable functions $y_{(\underline{\mathbf{x}})}$

cost function: $E^T = \frac{1}{2p} \sum_{\alpha=1}^{p} \left(y(\underline{\mathbf{x}}^{(\alpha)}) - y_T^{(\alpha)} \right)^2$



many functions are consistent with the data

$\Rightarrow$ **ill-posed learning problem**

Regularization

New cost function: $R = E^T + \lambda E^R$ (Tikhonov, 1963)

Regularization

New cost function: $R = E^T + \lambda E^R$ (Tikhonov, 1963)

$$y(\underline{\mathbf{x}}) = \int d\underline{\mathbf{k}} \quad e^{(i\underline{\mathbf{k}}^T\underline{\mathbf{x}})}\, \tilde{y}(\underline{\mathbf{k}}) \qquad \text{(Fourier transform)}$$

Regularization

New cost function: $R = E^T + \lambda E^R$ (Tikhonov, 1963)

$$y(\underline{\mathbf{x}}) = \int d\underline{\mathbf{k}} \quad e^{(i\underline{\mathbf{k}}^T \underline{\mathbf{x}})} \, \tilde{y}(\underline{\mathbf{k}})$$ (Fourier transform)

$$E^R = \frac{1}{2} \int d\underline{\mathbf{k}} \frac{\left|\tilde{y}(\underline{\mathbf{k}})\right|^2}{\widetilde{G}(\underline{\mathbf{k}})}$$ (regularization)

Regularization

New cost function: $R = E^T + \lambda E^R$ (Tikhonov, 1963)

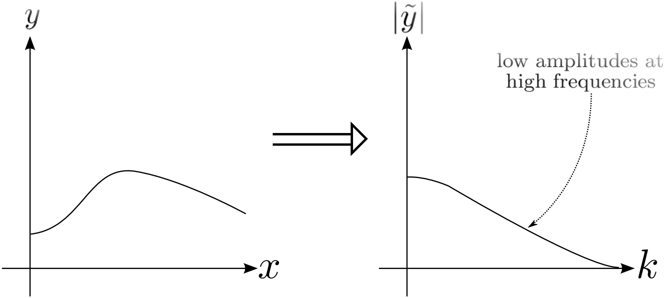
$$y(\underline{\mathbf{x}}) = \int d\underline{\mathbf{k}} \quad e^{(i\underline{\mathbf{k}}^T\underline{\mathbf{x}})}\ \tilde{y}(\underline{\mathbf{k}}) \qquad \text{(Fourier transform)}$$

$$E^R = \frac{1}{2}\int d\underline{\mathbf{k}} \frac{\left|\tilde{y}(\underline{\mathbf{k}})\right|^2}{\widetilde{G}(\underline{\mathbf{k}})} \qquad \text{(regularization)}$$

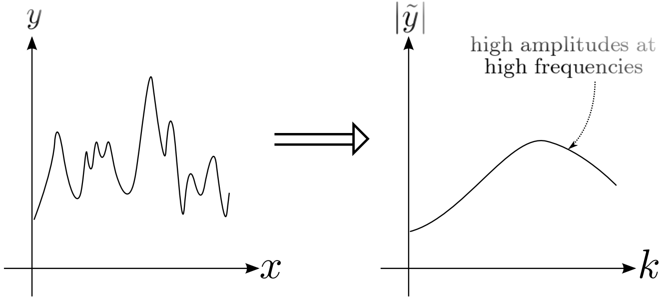
- Filter $\widetilde{G}(\underline{\mathbf{k}})$ imposes (soft-)constraints on $y(\tilde{k}) \rightsquigarrow$ functions $y(x)$.

$\Rightarrow$ **well-posed problem** (existence, uniqueness, continuity, see Haykin, ch. 5)

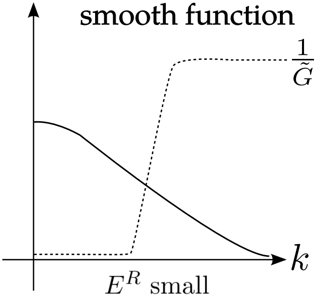
Smooth functions in Fourier space

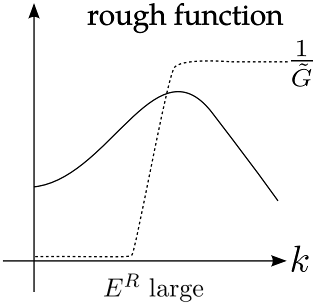


Rough functions in Fourier space



Effects of regularization





high pass $\widetilde{G}^{-1} \Rightarrow$ implicit smoothness constraint
e.g. for E^R from before

Result of model selection

$$\inf_{\underline{\mathbf{w}}} R \quad = \quad E^T_{[\underline{\mathbf{w}}]} + \lambda E^R_{[\underline{\mathbf{w}}]}$$

$$y(\underline{\mathbf{x}}) = \sum_{\alpha=1}^{p} \mathrm{w}_\alpha \, G(\underline{\mathbf{x}} - \underline{\mathbf{x}}^{(\alpha)}) \qquad \text{(RBF-network depending on filter)}$$

with

$$G(\underline{\mathbf{x}}) = \int d\underline{\mathbf{k}} \; e^{(i\underline{\mathbf{k}}^T \underline{\mathbf{x}})} \, \widetilde{G}(\underline{\mathbf{k}}) \qquad \text{(Fourier-transform of filter)}$$

Result of model selection

$$\inf_{\underline{\mathbf{w}}} R \quad = \quad E^T_{[\underline{\mathbf{w}}]} + \lambda E^R_{[\underline{\mathbf{w}}]}$$

$$y(\underline{\mathbf{x}}) = \sum_{\alpha=1}^{p} \mathrm{w}_\alpha \, G(\underline{\mathbf{x}} - \underline{\mathbf{x}}^{(\alpha)})$$ (RBF-network depending on filter)

with $$G(\underline{\mathbf{x}}) = \int d\underline{\mathbf{k}} \; e^{(i\underline{\mathbf{k}}^T \underline{\mathbf{x}})} \, \widetilde{G}(\underline{\mathbf{k}})$$ (Fourier-transform of filter)

- prior knowledge determines shape of basis functions
- location of data points determine location of centroids (unsupervised)

Result of model selection

$$\inf_{\underline{\mathbf{w}}} R \quad = \quad E^T_{[\underline{\mathbf{w}}]} + \lambda E^R_{[\underline{\mathbf{w}}]}$$

$$y(\underline{\mathbf{x}}) = \sum_{\alpha=1}^{p} \mathrm{w}_\alpha \, G(\underline{\mathbf{x}} - \underline{\mathbf{x}}^{(\alpha)}) \qquad \text{(RBF-network depending on filter)}$$

with

$$G(\underline{\mathbf{x}}) = \int d\underline{\mathbf{k}} \, e^{(i\underline{\mathbf{k}}^T \underline{\mathbf{x}})} \, \widetilde{G}(\underline{\mathbf{k}}) \qquad \text{(Fourier-transform of filter)}$$

- prior knowledge determines shape of basis functions
- location of data points determine location of centroids (unsupervised)

$$\underline{\mathbf{w}} = \frac{1}{p\lambda} \underline{\mathbf{G}}^{-1} \left(\underline{\mathbf{G}}^{-1} + \frac{1}{p\lambda} \underline{\mathbf{I}} \right)^{-1} \underline{\mathbf{y}}_T \,, \quad \text{where} \quad G_{\alpha\beta} = G(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{x}}^{(\beta)})$$

(see supplementary material)

Result of model selection

$$\inf_{\underline{\mathbf{w}}} R \quad = \quad E^T_{[\underline{\mathbf{w}}]} + \lambda E^R_{[\underline{\mathbf{w}}]}$$

$$y(\underline{\mathbf{x}}) = \sum_{\alpha=1}^{p} \mathrm{w}_\alpha \, G(\underline{\mathbf{x}} - \underline{\mathbf{x}}^{(\alpha)}) \qquad \text{(RBF-network depending on filter)}$$

with

$$G(\underline{\mathbf{x}}) = \int d\underline{\mathbf{k}} \; e^{(i\underline{\mathbf{k}}^T \underline{\mathbf{x}})} \, \widetilde{G}(\underline{\mathbf{k}}) \qquad \text{(Fourier-transform of filter)}$$

- prior knowledge determines shape of basis functions
- location of data points determine location of centroids (unsupervised)

$$\underline{\mathbf{w}} = \frac{1}{p\lambda} \underline{\mathbf{G}}^{-1} \left(\underline{\mathbf{G}}^{-1} + \frac{1}{p\lambda} \underline{\mathbf{I}} \right)^{-1} \underline{\mathbf{y}}_T \,, \quad \text{where} \quad G_{\alpha\beta} = G(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{x}}^{(\beta)})$$

- solution equivalent to ridge regression

(see supplementary material)

Example: Gaussian filters

Prior on smooth functions: penalize high frequencies

Example: Gaussian filters

Prior on smooth functions: penalize high frequencies

$$E^R \quad = \quad \frac{1}{2} \int d\underline{\mathbf{k}} \frac{|\tilde{y}(\underline{\mathbf{k}})|^2}{G(\underline{\mathbf{k}})} \quad = \quad \int d\underline{\mathbf{k}} \; \underbrace{e^{\sigma^2 \underline{\mathbf{k}}^2}}_{\text{high pass}} \; \left| \tilde{y}_{(\underline{\mathbf{k}})} \right|^2$$

Example: Gaussian filters

Prior on smooth functions: penalize high frequencies

$$E^R = \frac{1}{2} \int d\underline{\mathbf{k}} \frac{|\tilde{y}(\underline{\mathbf{k}})|^2}{G(\underline{\mathbf{k}})} = \int d\underline{\mathbf{k}} \underbrace{e^{\sigma^2 \underline{\mathbf{k}}^2}}_{\text{high pass}} \left| \tilde{y}_{(\underline{\mathbf{k}})} \right|^2$$

leads to:

$$G(\underline{\mathbf{x}}) \sim \exp\left(-\frac{\underline{\mathbf{x}}^2}{\sigma^2} \right)$$

Example: Gaussian filters

Prior on smooth functions: penalize high frequencies

$$E^R = \frac{1}{2}\int d\underline{\mathbf{k}} \frac{|\tilde{y}(\underline{\mathbf{k}})|^2}{G(\underline{\mathbf{k}})} = \int d\underline{\mathbf{k}} \underbrace{e^{\sigma^2 \underline{\mathbf{k}}^2}}_{\text{high pass}} \left|\tilde{y}_{(\underline{\mathbf{k}})}\right|^2$$

leads to:

$$G(\underline{\mathbf{x}}) \sim \exp\left(-\frac{\underline{\mathbf{x}}^2}{\sigma^2}\right)$$

$\Rightarrow$ close connection between RBF-networks and regularization

$\Rightarrow$ yet another model selection procedure for RBF-networks

Example: Gaussian filters

Prior on smooth functions: penalize high frequencies

$$E^R = \frac{1}{2} \int d\underline{\mathbf{k}} \frac{|\tilde{y}(\underline{\mathbf{k}})|^2}{G(\underline{\mathbf{k}})} = \int d\underline{\mathbf{k}} \underbrace{e^{\sigma^2 \underline{\mathbf{k}}^2}}_{\text{high pass}} \left|\tilde{y}_{(\underline{\mathbf{k}})}\right|^2$$

leads to:

$$G(\underline{\mathbf{x}}) \sim \exp\left(-\frac{\underline{\mathbf{x}}^2}{\sigma^2}\right)$$

$\Rightarrow$ close connection between RBF-networks and regularization

$\Rightarrow$ yet another model selection procedure for RBF-networks

Problem: Number of basis functions = no. of data points (large!)

$\rightsquigarrow$ sparse expansion desirable

$\rightsquigarrow$ support vector machines

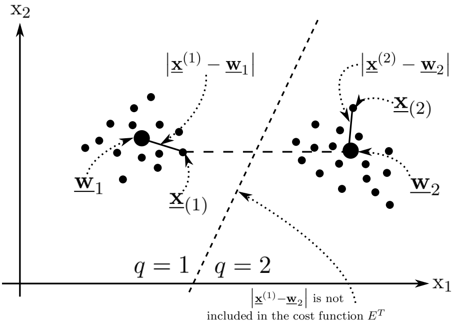
End of Section 1.5

the following slides contain

OPTIONAL MATERIAL

Batch K-Means

- prototypes: $\underline{\mathbf{t}}_q, q = 1, \ldots, M$
- binary assignment: $m_q^{(\alpha)} = 1$ if $\underline{\mathbf{x}}^{(\alpha)}$ belongs to cluster q, 0 else
- clustering **cost function**: $E\left[\{m_q^{(\alpha)}\}, \{\underline{\mathbf{t}}_q\}\right] = \frac{1}{2p} \sum_{q,\alpha} m_q^{(\alpha)} \|\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{t}}_q\|^2$



Batch K-means: algorithm

Initialization of $\underline{\mathbf{t}}_q, q = 1, \ldots, M$ (e.g.around data's center of mass)
BEGIN loop

1. assign every data point to its nearest prototype
   $m_q^{(\alpha)} = 1$ if $q = \operatorname{argmin}_\gamma \left| \underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{t}}_\gamma \right|$ $\qquad 0$, else
2. choose $\underline{\mathbf{t}}_q$ such that E^T is minimal (for the given -new- assignments)
   $$\underline{\mathbf{t}}_q = \frac{\sum\limits_\alpha m_q^{(\alpha)} \underline{\mathbf{x}}^{(\alpha)}}{\sum\limits_\alpha m_q^{(\alpha)}}$$ (center of mass of its assigned data)

END loop

Clustering: illustration