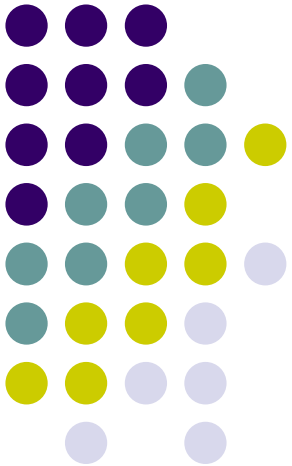
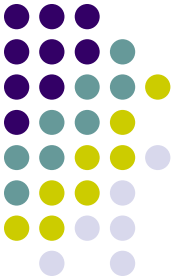
# Neural Networks: Radial-Basis Function Networks

Prof. dr. sc. Sven Lončarić

University of Zagreb
Faculty of Electrical Engineering and Computing
sven.loncaric@fer.hr
http://www.fer.unizg.hr/ipg

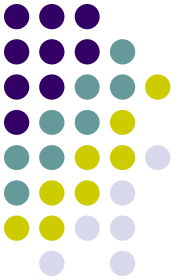


# Lecture Overview I



- Introduction
- Cover's Theorem on the Separability of Patterns
- Interpolation Problem
- Interpolation using RBF Networks
- Generalized RBF Networks
- Supervised Learning as an Ill-Posed Hypersurface Reconstruction Problem
- Regularization Theory
- Regularization Networks

# Lecture Overview II



- XOR Problem (revisited)
- Comparison of RBF Networks and Multilayer Perceptrons
- Learning Strategies
- Discussion
- Problems

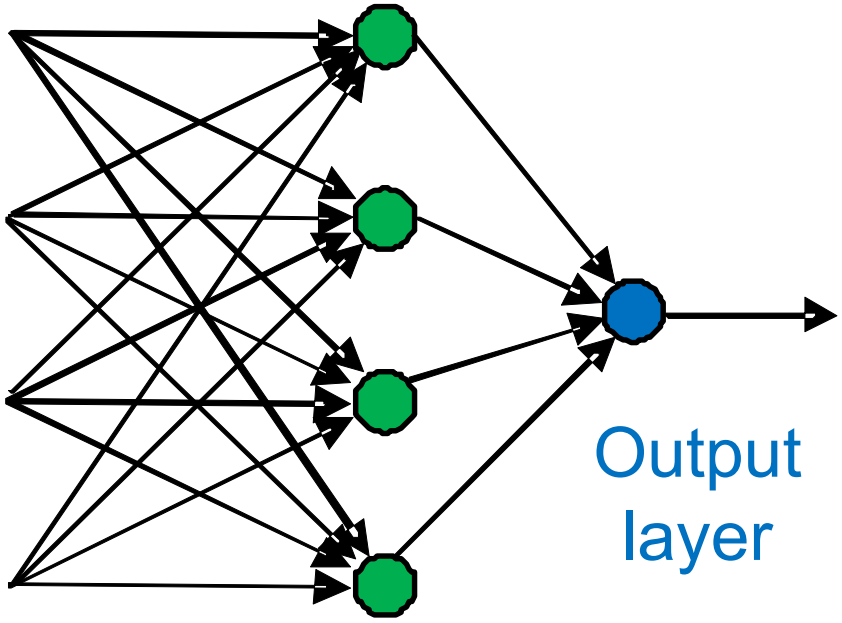
# Introduction

- Radial-Basis Function (RBF) Networks
  - In Croatian: *radijalne mreže*
- Learning in multilayer perceptron networks based on the BP algorithm is interpreted as an optimization problem (least squares minimization)
- Learning in RBF networks is interpreted as a curve-fitting (approximation) problem in a high-dimensional space
- A curve (surface) which must be approximated is a input-output function defined by the training data

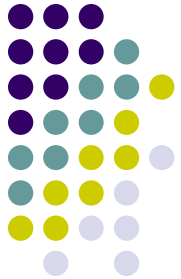
# Structure of a RBF Network I

- The RBF network is comprised of three layers:
  - the input layer
  - the hidden layer (which has a different role than hidden layers in multilayer perceptron)
  - the output layer
- Transformation from the input layer to the hidden layer is *nonlinear*
- Transformation from the hidden layer to the output layer is *linear*

# Structure of a RBF Network II



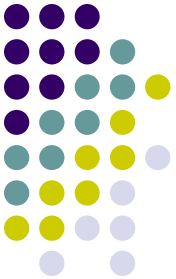
# Cover's Theorem I



- When using RBF networks the problem of pattern classification is solved by transforming it into a high-dimensional space in a nonlinear manner
- Justification is found in Cover's theorem on the separability of patterns:

  *A complex pattern-classification problem cast in a high-dimensional space nonlinearly is more likely to be linearly separable than in a low-dimensional space.*

# Cover's Theorem II Interpretation



- From the perceptron theory it is known that the classification problem is easily solvable if input samples are linearly separable
- RBF network as a classifier:
  1. The hidden layer transforms input data nonlinearly so classes become linearly separable
  2. The output layer is linear and may now easily perform the classification of two linearly separable classes

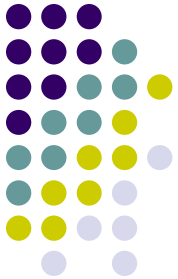
# Cover's Theorem III

- Let X = { $\mathbf{x}_1$, $\mathbf{x}_2$, …, $\mathbf{x}_N$ } be a set of input samples where each sample belongs to one of two classes $X^+$ and $X^-$, we call this a *dichotomy*
- Let the input vector $\mathbf{x}$ be *P*-dimensional
- For each $\mathbf{x}$ form a new vector:

$$\boldsymbol{\varphi}(\mathbf{x}) = [\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x}), \ldots, \varphi_M(\mathbf{x})]^T$$

- Then $\boldsymbol{\varphi}(\mathbf{x})$ is a transformation of input vector from the input space into a new *M*-dimensional hidden space
- The function $\varphi_i(\mathbf{x})$ is called a *hidden function*
  - Its role is similar to the role of a hidden neuron in a multilayer network

# Cover's Theorem IV



- A dichotomy (binary partition) $\{X^+, X^-\}$ is $\varphi$-separabe if there exists a *M*-dimensional vector **w** such that:

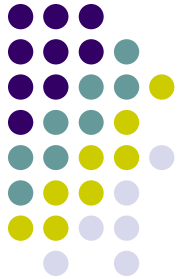
$$\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) \geq 0, \quad \mathbf{x} \in X^+$$
$$\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) < 0, \quad \mathbf{x} \in X^-$$

- A hyperplane given by the equation

$$\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) = 0$$

defines a separation surface in $\varphi$ (hidden) space

# Cover's Theorem V

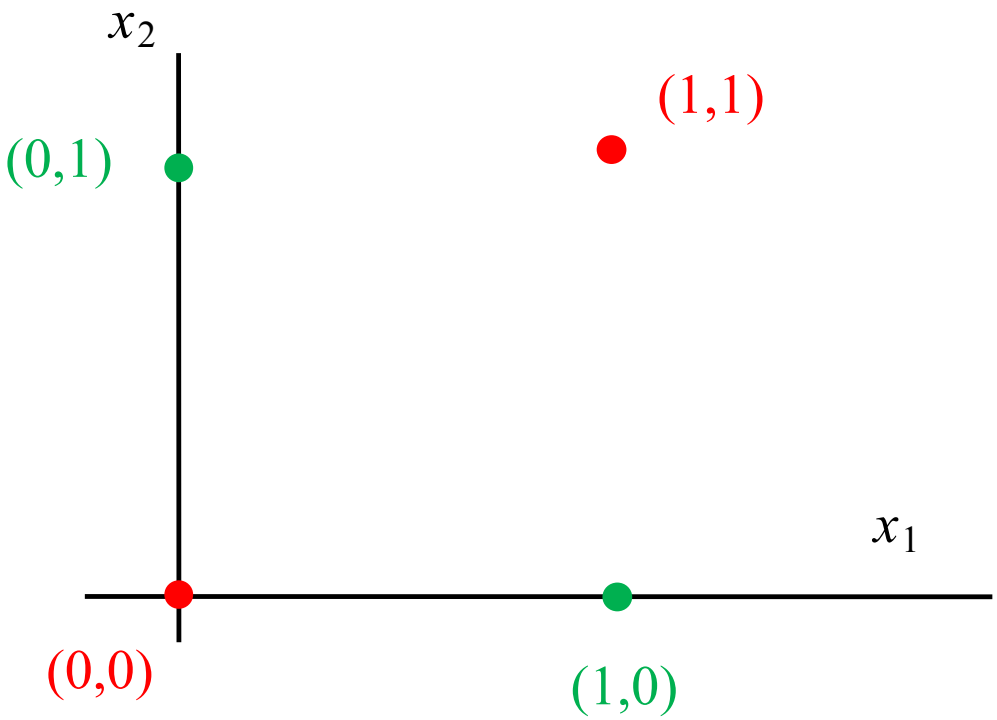


- The inverse image of a separating hyperplane is the separating surface in the input pattern space:

$$\left\{\mathbf{x} : \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) = 0\right\}$$

# Example: XOR Problem I

- 0 XOR 0 = 0
- 1 XOR 1 = 0
- 0 XOR 1 = 1
- 1 XOR 0 = 1



# Example: XOR Problem II

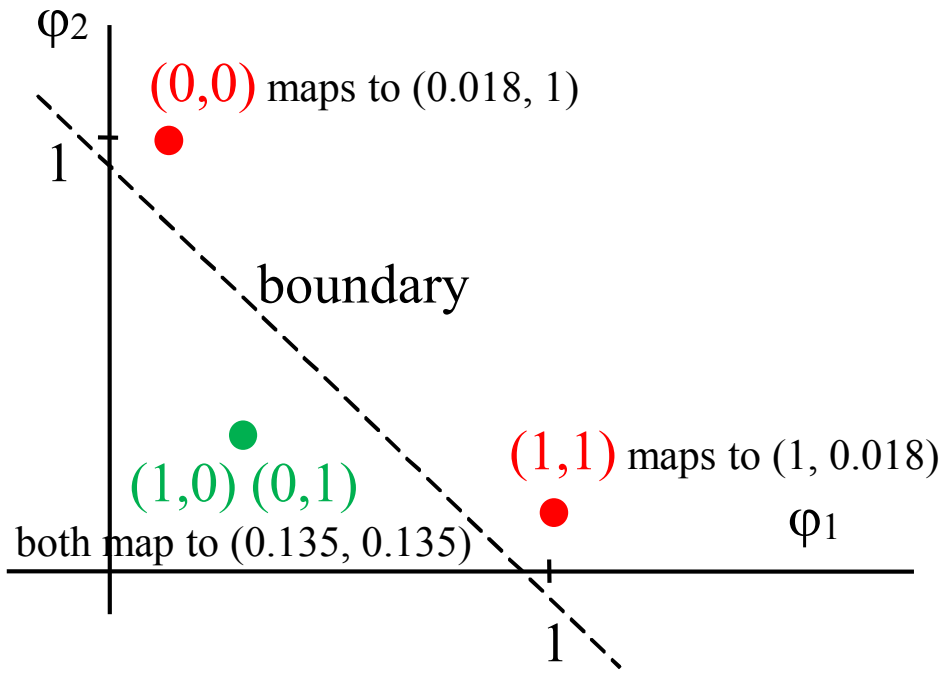
- Let us define the hidden functions as:

$$\varphi_1(\mathbf{x}) = e^{-\|\mathbf{x}-\mathbf{t}_1\|^2}, \quad \mathbf{t}_1 = [1,1]^T$$
$$\varphi_2(\mathbf{x}) = e^{-\|\mathbf{x}-\mathbf{t}_2\|^2}, \quad \mathbf{t}_2 = [0,0]^T$$

- The input vectors are then transformed into $\varphi$ (hidden) space as follows (next slide)

# Example: XOR Problem III

- All samples are clearly linearly separable in the new space
- Hence the classification problem may be solved using a linear classifier such as the perceptron (the output layer of a RBF network)



# Interpolation Problem I

- Consider a network with an input layer, a hidden layer, and an output layer containing a single neuron
- Such network performs a *nonlinear mapping* from the input space into the hidden space followed by a *linear mapping* from the hidden space to the output space
- Overall, such network represents a mapping

$$s: \mathbb{R}^P \rightarrow \mathbb{R}$$

- This mapping may be interpreted as a hypersurface

$$\Gamma \subset \mathbb{R}^{P+1}$$

# Interpolation Problem II

- Training of such network constitutes the optimization of a fitting procedure for the hypersurface $\Gamma$, which is based on known data points represented as input-output examples (patterns)
- The generalization phase is equivalent to the interpolation between given input-output examples
- Such interpretation leads toward the theory of *multivariable interpolation* in high-dimensional space
- Then the interpolation problem may be stated as follows (next slide):

# Interpolation Problem III

- Given a set of *N* different points

$$\{\boldsymbol{x}_i \in \mathbb{R}^P | i = 1,2, \dots, N\}$$

and a corresponding set of *N* real numbers

$$\{d_i \in \mathbb{R} | i = 1,2, \dots, N\}$$

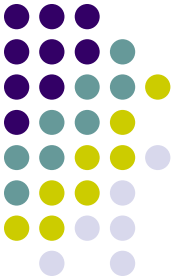
find a function

$$F: \mathbb{R}^P \to \mathbb{R}$$

that satisfies the interpolation condition

$$F(\boldsymbol{x}_i) = d_i, \qquad i = 1,2, \dots, N$$

# Interpolation using RBF Networks I



- RBF networks use the function *F* that has the following form:

$$F(\mathbf{x}) = \sum_{i=1}^{N} w_i \varphi\left(\left\|\mathbf{x} - \mathbf{x}_i\right\|\right)$$

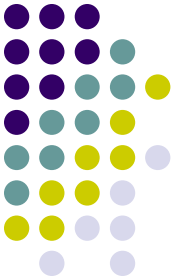
where

$$\left\{\varphi\left(\left\|\mathbf{x} - \mathbf{x}_i\right\|\right) \mid i = 1,2,\ldots,N\right\}$$

is a set of N arbitrary (generally nonlinear) functions which are known as radial basis functions

- The known data points $\mathbf{x}_i$ are taken to be centers of the radial basis functions

# Interpolation using RBF Networks II



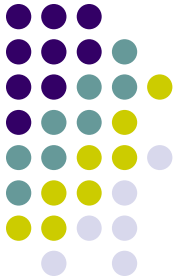
- Combining the interpolating condition and the definition of *F* yields the following set of simultaneous linear equations:

$$\begin{bmatrix} \varphi_{11} & \varphi_{12} & \cdots & \varphi_{1N} \\ \varphi_{21} & \varphi_{22} & \cdots & \varphi_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ \varphi_{N1} & \varphi_{N2} & \cdots & \varphi_{NN} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{bmatrix}$$

where

$$\varphi_{ji} = \varphi(\|\mathbf{x}_j - \mathbf{x}_i\|), \quad j, i = 1, 2, \ldots, N$$

# Interpolation using RBF Networks III



- Let **d** and **w** represent the desired response vector and linear weight vector respectively:

$$\mathbf{d} = [d_1, d_2, \ldots, d_N]^T$$

$$\mathbf{w} = [w_1, w_2, \ldots, w_N]^T$$

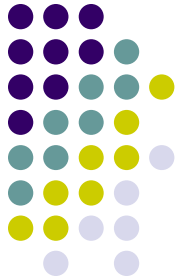
- Let $\mathbf{\Phi}$ denote an N×N matrix with elements $\varphi_{ij}$:

$$\mathbf{\Phi} = \{\varphi_{ji}\}, \quad j, i = 1, 2, \ldots, N$$

- We call this matrix $\mathbf{\Phi}$ the interpolation matrix
- A set of linear equations from the previous slide then becomes a matrix equation:

$$\mathbf{\Phi w} = \mathbf{d}$$

# Interpolation using RBF Networks IV

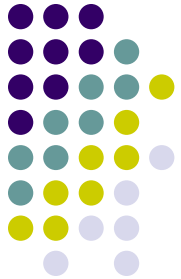


- Let us assume all $\mathbf{x}_1, \ldots, \mathbf{x}_N$ are different vectors.
- Consider a class of radial basis functions having the property that the corresponding interpolation matrix $\Phi$ is positive definite
- Some examples (often used in practice) of RBFs which satisfy that property are:

$$\varphi(r) = \frac{1}{(r^2 + c^2)^{1/2}}, \quad c > 0, \quad r \geq 0$$ Inverse multiquadrics

$$\varphi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right), \quad \sigma > 0, \quad r \geq 0$$ Gaussian functions

# Interpolation using RBF Networks V



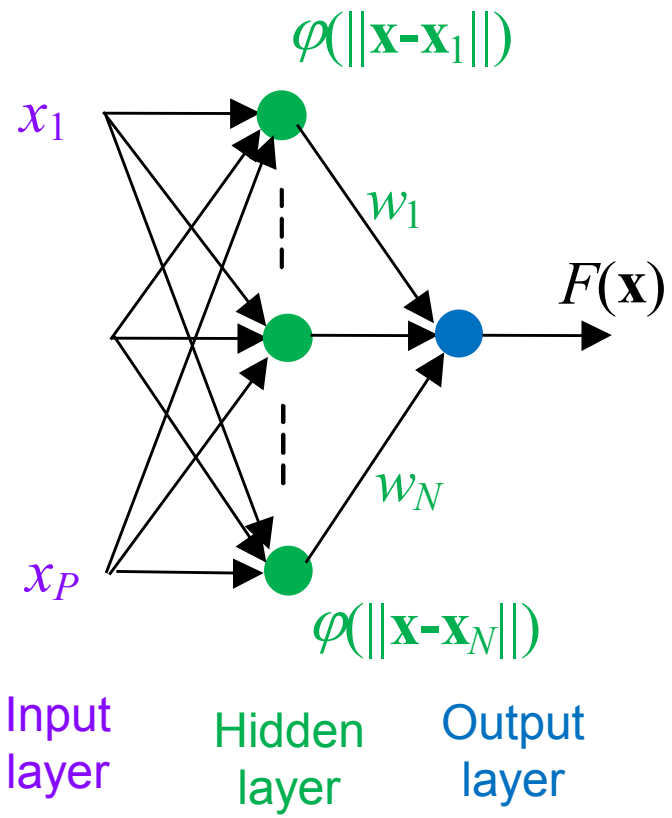
- There exists a $\Phi$ is nonsingular and/or positive definite large class of RBFs for which
  - See Micchelli's theorem (1986) for details
- As the matrix $\Phi$ is positive definite there exists an inverse matrix hence the unknown vector of weights may be computed using:

$$\mathbf{w} = \mathbf{\Phi}^{-1}\mathbf{d}$$

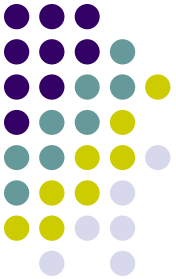
- In practice solution may be unstable if the matrix $\Phi$ is close to singular
- However, the inverse may be stabilized using the results of regularization theory
  - We'll return to regularization theory later

# Interpolation using RBF Networks VI

- *i*-th neuron of the hidden layer implements the function $\varphi(\|\mathbf{x}-\mathbf{x}_i\|)$
- The output neuron computes a weighted linear combination of its inputs



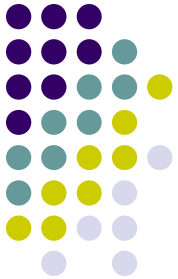
# Generalized RBF Networks I



- It is now clear that for each input sample $\mathbf{x}_i$ there must exist a corresponding neuron in the hidden layer
- For a large number of input samples this may present a problem
- Then instead of *N* a reduced number *M* << *N* of radial basis functions may be used

$$F(\mathbf{x}) = \sum_{i=1}^{M} w_i \varphi(\|\mathbf{x} - \mathbf{x}_i\|)$$

# Generalized RBF Networks II



- The interpolation matrix $\Phi$ for the case of a reduced number of RFBs has dimensions of $N \times M$, hence the inverse matrix does not exist
  - The system is overdetermined

$$\begin{bmatrix} \varphi_{11} & \varphi_{12} & \cdots & \varphi_{1M} \\ \varphi_{21} & \varphi_{22} & \cdots & \varphi_{2M} \\ \vdots & \vdots & & \vdots \\ \varphi_{i1} & \varphi_{i2} & \cdots & \varphi_{iM} \\ \vdots & \vdots & & \vdots \\ \varphi_{N1} & \varphi_{N2} & \cdots & \varphi_{NM} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_i \\ \vdots \\ d_N \end{bmatrix}$$

- The weights for this case may be found using the pseudoinverse matrix

$$\mathbf{w} = \mathbf{\Phi}^{+}\mathbf{d} = (\mathbf{\Phi}^{T}\mathbf{\Phi})^{-1}\mathbf{\Phi}^{T}\mathbf{d}$$

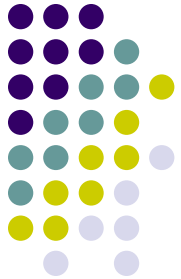
# Modifications

- Instead of using a reduced set of RBFs there are also other possible useful modifications:

1. The centers of RBFs do not need to be determined by input samples but may instead be set to some other values.
2. If Gaussian functions are used, then each mapping from input to hidden space may use a different width parameter $\sigma$.
3. We may add threshold operation to the output neuron.

- If a modification introduces additional network parameters then all such parameters must be determined during the learning process

# Learning as an Inverse Problem

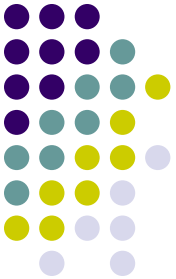


- *Learning* may be considered as the problem of reconstructing some surface given a set of points which may be spatially far apart
- In this interpretation *learning* is an inverse problem
  - We known only a finite number of input-output pairs from which a complete surface must be determined
- This inverse problem may be well-posed or may be ill-posed
- Consider the unknown mapping

$$F : X \rightarrow Y$$

where *X* denotes domain and *Y* codomain

# Well-Posed Problem



- **Definition**: The problem of reconstructing the mapping *F* is said to be *well-posed* if three conditions are satisfied:

  1. Existence: for every input **x** there does exist an output $y = F(\mathbf{x})$
  2. Uniqueness: $F(\mathbf{x})=F(\mathbf{t})$ if and only if $\mathbf{x}=\mathbf{t}$
  3. Continuity:

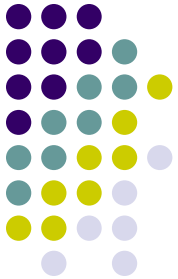
$$\forall \varepsilon > 0, \exists \delta = \delta(\varepsilon) \mid \rho_x(\mathbf{x}, \mathbf{t}) < \delta \Rightarrow \rho_y(F(\mathbf{x}), F(\mathbf{t})) < \varepsilon$$

  where $\rho_x$ and $\rho_y$ are distances between vectors in their respective spaces

# Ill-Posed Problem

- **Definition**: The problem of reconstructing the mapping *F* is ill-posed if and only if it is not well-posed.

# Supervised Learning

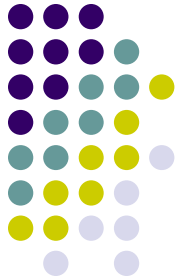


- Supervised learning is an ill-posed problem of a surface reconstruction:

1. Input samples do not contain sufficient information, so uniqueness condition is not fulfilled
2. Due to noise and general uncertainty continuity and existence conditions may not be fulfilled

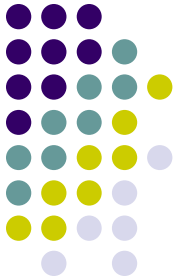
- To make the learning well-posed we must introduce some **a priori** knowledge about the mapping *F*
- Such knowledge may be contained in the redundancy of learning samples

# Regularization Theory I



- Andrey Tikhonov, 1963
- Regularization theory allows us to find a solution to ill-posed problems with some desireable properties
- The basic idea of regularization is to stabilize the solution by introducing an additional term which contains prior knowledge about the mapping *F* (e.g. continuity, smoothness)
- The unknown mapping F is now determined via minimization of a cost function *E*(*F*) which is comprised of two terms

# Regularization Theory II



- The standard error term (*interpolation error*) measures the error between the desired response $d_i$ and the mapping $F(x_i)$ of the input $x_i$:

$$E_s(F) = \sum_{i=1}^{N} [d_i - F(\mathbf{x}_i)]^2$$

- The regularizing term depends on the geometric properties of the approximating function $F$

$$E_c(F) = \|\mathbf{P}F\|^2$$

Here **P** is a linear differential operator which acts on $F$.

# Solution using Regularization I

- The quantity to be minimized in regularization theory is

$$E(F) = E_s(F) + \lambda E_c(F)$$

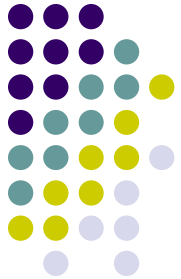
where $\lambda$ is regularization parameter

- For a specific choice of the operator **P** the optimal solution *F* which minimizes the quantity E(F) has the form:

$$F_\lambda(\mathbf{x}) = \frac{1}{\lambda}\sum_{i=1}^{N}[d_i - F(\mathbf{x}_i)]G(\mathbf{x};\mathbf{x}_i)$$

where G(. ; .) is Green's function which depends on the particular choice of the operator **P**

# Solution using Regularization II



- A general solution to the regularized problem has the form

$$F(\mathbf{x}) = \sum_{i=1}^{N} w_i G(\mathbf{x}, \mathbf{x}_i)$$

  where $G(\mathbf{x},\mathbf{x}_i)$ is the Green's function

- If the stabilizer P is *translationally* invariant then Green's function depends only on the difference

$$F(\mathbf{x}) = \sum_{i=1}^{N} w_i G(\mathbf{x} - \mathbf{x}_i)$$

- If the stabilizer P is both *translationally* and *rotationally* invariant then Green's function must be a **radial-basis function**:

$$F(\mathbf{x}) = \sum_{i=1}^{N} w_i G(\|\mathbf{x} - \mathbf{x}_i\|)$$

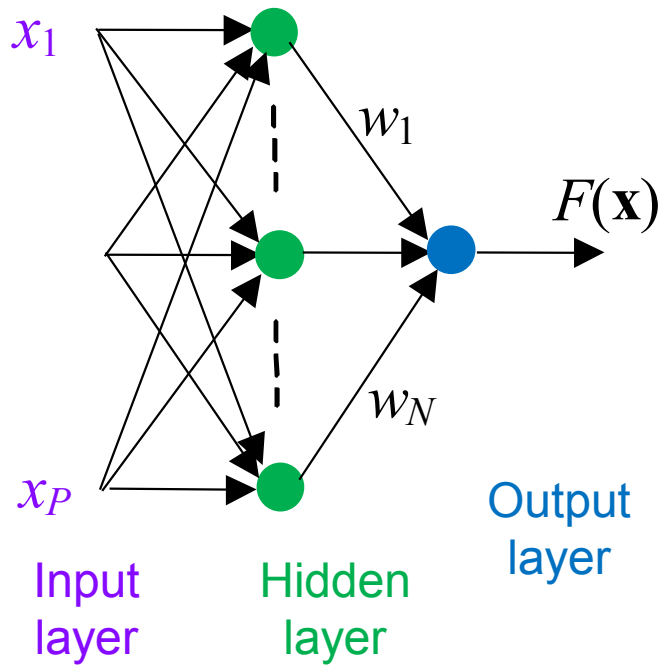
# Regularization Networks

- Conclusion: regularization of the solution to the interpolation problem of the separating surface naturally leads to radial-basis functions
- RBF networks represent an architecture which solves the interpolation problem having desirable properties:

1. It is a **universal approximator** in that it can approximate any multivariate continuous function given a sufficiently large number of hidden units.
2. It has the **best-approximation property** meaning that there always exists a choice of coefficients that approximates *F* better than all other possible choices.
3. The solution is **optimal** in the sense that is minimizes the error functional from slide 33.

# Regularization Networks

- The hidden layer of N Green's functions $G(||\mathbf{x} - \mathbf{x}_i||)$ which must be *positive definite*
  - One possible choice of Green's function is Gaussian
- The output layer implements weighted linear sum



# Example: XOR Problem (revisited) I

- Let the radial-basis function be a non-normalized multivariate Gaussian:

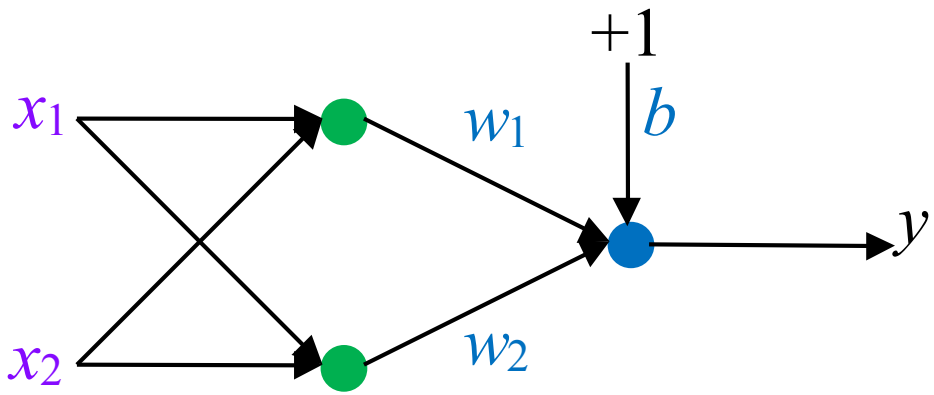
$$G\left(\left\|\mathbf{x}-\mathbf{t}_i\right\|\right)=\exp\left(\frac{1}{2}\left\|\mathbf{x}-\mathbf{t}_i\right\|^2\right), \quad i=1,2$$

  where the centers $\mathbf{t}_1 = [1 \;\; 1]^T$ i $\mathbf{t}_2 = [0 \;\; 0]^T$ are fixed

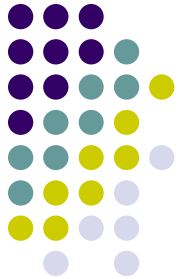
- Let the single output layer neuron have an additional input *b* (the threshold)

# Example:
# XOR Problem (revisited) II

- The structure of such RBF network is:



# Example: XOR Problem (revisited) III



- The input-output equation is:

$$y(\mathbf{x}) = \sum_{i=1}^{2} w_i G\left(\left\|\mathbf{x} - \mathbf{t}_i\right\|\right) + b$$

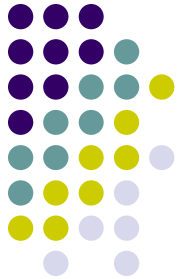
- The interpolation condition must hold for all learning examples:

$$y(\mathbf{x}_j) = d_j, \quad j = 1,2,3,4$$

- Denote:

$$g_{ji} = G\left(\left\|\mathbf{x}_j - \mathbf{t}_i\right\|\right), \quad j = 1,2,3,4; \quad i = 1,2$$

# Example: XOR Problem (revisited) IV



- We then obtain a matrix equation

$$\mathbf{Gw} = \mathbf{d}$$

where:

$$\mathbf{G} = \begin{bmatrix} g_{11} & g_{12} & 1 \\ g_{21} & g_{22} & 1 \\ g_{31} & g_{32} & 1 \\ g_{41} & g_{42} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0.1353 & 1 \\ 0.3679 & 0.3679 & 1 \\ 0.1353 & 1 & 1 \\ 0.3679 & 0.3679 & 1 \end{bmatrix}$$

$x_1$=(1,1)

$x_2$=(0,1)

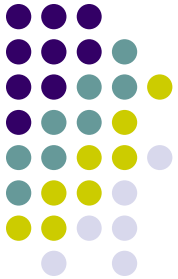
$x_3$=(0,0)

$x_4$=(1,0)

$$\mathbf{w} = \begin{bmatrix} w_1 & w_2 & b \end{bmatrix}^T$$

$$\mathbf{d} = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}^T$$

# Example: XOR Problem (revisited) V

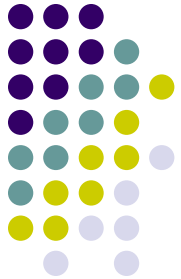


- Obtained linear system is overdetermined as there are more equations than there are unknowns
- The matrix **G** is not square
- The solution is found using a pseudoinverse:

$$\mathbf{w} = \mathbf{G}^{+}\,\mathbf{d} = (\mathbf{G}^{T}\,\mathbf{G})^{-1}\,\mathbf{G}^{T}$$

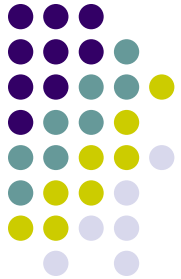
- We obtain $\mathbf{w} = [2.5027 \;\; 2.5027 \;\; -1.8413]^{T}$
- The first two weights are equal due to symmetry of the XOR problem and of the fixed centers $\mathbf{t}_1$ and $\mathbf{t}_2$

# Comparison of RBF Networks and Multilayer Perceptrons



1. RBF network has a single hidden layer while MLP use many layers.
2. All neurons of MLP commonly use the same model while hidden neurons of a RBF network may be different.
3. The hidden layer of a RBF network is nonlinear and the output layer is linear; for MLP all neurons are usually nonlinear.
4. The argument of the activation function in a RBF network is distance between the input sample and the RBF's center; for MLP the argument of the activation function is a dot product of the input vector and the weight vector.

# Learning Strategies

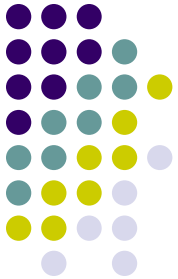


- There are many possible learning strategies for RBF networks which mainly depend on how the centers are selected

- Some of them are:

1. Fixed centers selected at random
2. Self-Organized Selection of Centers
3. Supervised Selection of Centers

# Fixed Centers I



- In this approach the centers of RBFs are randomly selected to some fixed values $\mathbf{t}_i$

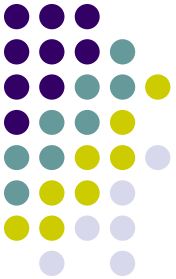
$$G\left(\left\|\mathbf{x}-\mathbf{t}_i\right\|\right)=\exp\left(-\frac{M}{d^2}\left\|\mathbf{x}-\mathbf{t}_i\right\|^2\right), \quad i=1,2,\ldots,M$$

  where $M$ is the number of centers and $d$ is the maximum distance between the chosen centers.

- This in effect sets the standard deviation of a multivariate Gaussian to

$$\sigma=\frac{d}{\sqrt{2M}}$$

# Fixed Centers II



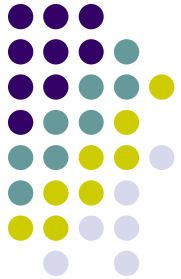
- Such choice of stanard deviation ensures the individual RBFs are neither too peaked or too flat
- The only parameters which must be learned are unknown weights **w**
- The weights may be computed using the pseudoinverse method:

$$\mathbf{w} = \mathbf{G}^{+}\mathbf{d}$$

where the matrix $\mathbf{G}=\{g_{ji}\}$ and where

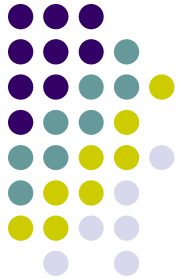
$$g_{ji} = \exp\left(-\frac{M}{d^2}\left\|\mathbf{x}_j - \mathbf{t}_i\right\|^2\right), \quad j = 1,\ldots,N; \quad i = 1,\ldots,M$$

# Self-Organizing Centers I



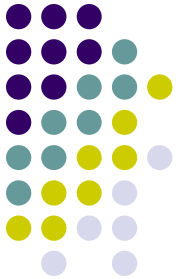
- In this approach the centers of RBFs are not fixed but are instead allowed to move in a selforganazing manner
- The selforganization allows the centers of RBFs to move to the areas containing many training samples

# Self-Organizing Centers II



- The positions of RBFs centers may be determined using the K-means clustering algorithm
- The values of weights *w* are then computed using supervised learning
- A LMS algorithm may be used for supervised learning
- The inputs of hidden layer neurons are inputs into the LMS algorithm

# Supervised Learning I



- This is the most generalized form where all free parameters of the network undergo a supervised learning process
- In this approach we define the instantaneous value of the cost function for all input-output pairs:

$$E = \frac{1}{2}\sum_{j=1}^{N} e_j^2$$

where $N$ is the size of training sample and $e_j$ is the error signal

# Supervised Learning II

- The error signal $e_j$ is defined by:

$$\begin{aligned} e_j &= d_j - F(\mathbf{x}_j) \\ &= d_j - \sum_{i=1}^{M} w_i G\left(\left\|\mathbf{x}_j - \mathbf{t}_i\right\|_A\right) \end{aligned}$$

where:

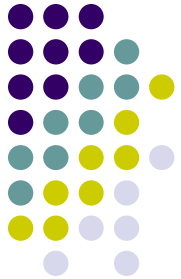
$$\left\|\mathbf{z}\right\|_{\mathbf{A}}^2 = \left\langle \mathbf{z}, \mathbf{z} \right\rangle_{\mathbf{A}} = \mathbf{z}^T \mathbf{A}\mathbf{z}$$

and a matrix **A** is positive definite.

# Supervised Learning III

- Free parameters of the RBF network which must be determined to minimize the instantaneous value of the cost function are:
  - output weights $\mathbf{w}_i$
  - the centers of radial-basis functions $\mathbf{t}_i$
  - the matrix of dot products $\mathbf{A}$
- These parameters are determined via the iterative method of steepest descent

# Supervised Learning IV



- An experimental study by Wettscherck and Dietterich (1992) showed that:

  1. RBF networks with unsupervised learning of centers' locations and supervised learning of output weights did not generalize nearly as well as multilayer perceptrons.

  2. Generalized RBF networks where all parameters are learned were able to generalize better than multlayer perceptrons.

# Applications of RBF Networks

- Image processing
- Speech recognition
- Adaptive equalization
- Medical diagnosis
- Source localisation in radar and sonar
- Analysis of stohastic signals

# Excersises

- The weights **w** obtained in the example XOR Problem Revisited (slides 37-41) are just one possible solution given fixed center positions
- Find at least one other solution of weights **w** which can solve the XOR problem and which differs from the demonstrated solution