Artificial Intelligence

Neural Networks

Lesson 11:
Training Radial Basis Function Networks

Vincenzo Piuri

Università degli Studi di Milano

Contents

- Radial basis function network
- Initialization
- C-means clustering
- Training radial basis function network
- Recognition of handwritten digits

Radial Basis Function Network

- Fixed learning task $L_{fixed} = \{l_1, \dots l_m\}$ with m training patterns $l = (\vec{\imath}^{(l)}, \vec{o}^{(l)})$

- **Simple radial basis function network**
  - One hidden neuron $v_k, k = 1, \dots, m$ for each training pattern:
    - $\forall k \in \{1, \dots, m\}$: $\overrightarrow{w_{v_k}} = \vec{\imath}^{(l_k)}$
  - If the activation function is Gaussian, the radii σ_k are chosen heuristically:
    - $\forall k \in \{1, \dots, m\}$: $\sigma_k = \frac{d_{max}}{\sqrt{2m}}$
    - $d_{max} = \max_{l_j, l_k \in L_{fixed}} d(\vec{\imath}^{(l_j)}, \vec{\imath}^{(l_k)})$

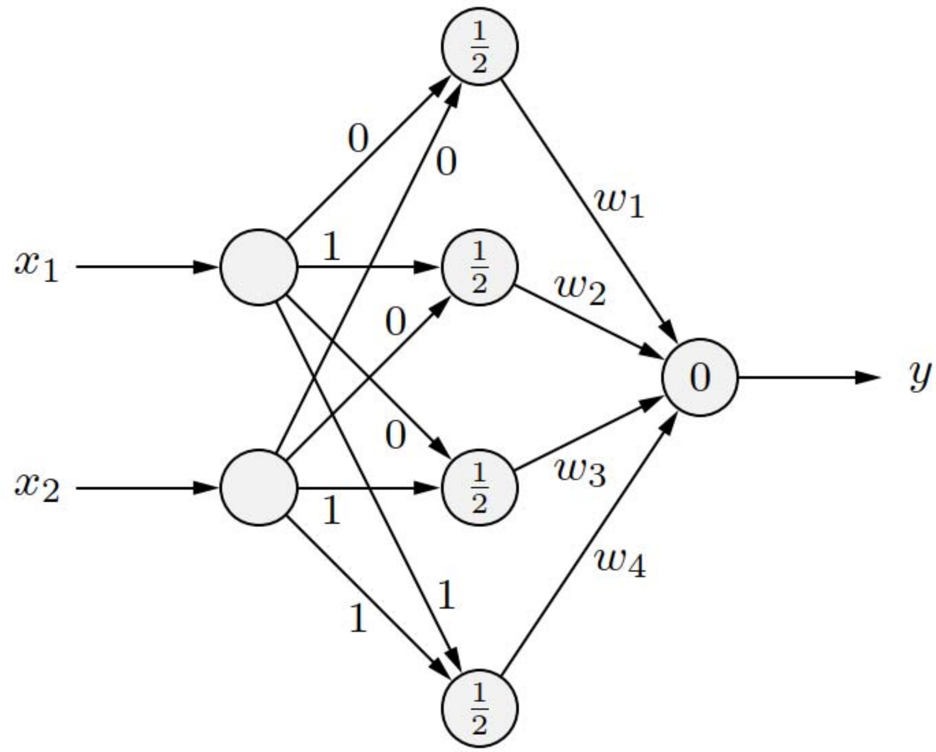
Initialization (1)

- Initializing the connections from the hidden to the output neurons
  - $\forall u$: $\sum_{k=1}^{m} w_{uv_m} out_{v_m}^{(l)} - \theta_u = o_u^{(l)}$
  - $\mathbf{A} \cdot \overrightarrow{w_u} = \overrightarrow{o_u}$
  - Where $\overrightarrow{o_u} = (o_u^{(l_1)}, \dots, o_u^{(l_m)})^T$ is the vector of desired outputs, $\theta_u = 0$, and
  - $\mathbf{A} = \begin{Bmatrix} out_{v_1}^{(l_1)} & out_{v_2}^{(l_1)} & \cdots & out_{v_m}^{(l_1)} \\ out_{v_1}^{(l_2)} & out_{v_2}^{(l_2)} & \cdots & out_{v_m}^{(l_2)} \\ \vdots & \vdots & \vdots & \vdots \\ out_{v_1}^{(l_m)} & out_{v_2}^{(l_m)} & \cdots & out_{v_m}^{(l_m)} \end{Bmatrix}$
  - Can be solved by inverting $\mathbf{A}$
    - $\overrightarrow{w_u} = \mathbf{A}^{-1}\overrightarrow{o_u}$

Initialization (2)

- Simple radial basis function network for the bi-implication $x_1 \leftrightarrow x_2$

x_1	x_2	y
0	0	1
1	0	0
0	1	0
1	1	1



Initialization (3)

- Simple radial basis function network for the bi-implication $x_1 \leftrightarrow x_2$

$$\mathbf{A} = \begin{pmatrix} 1 & e^{-2} & e^{-2} & e^{-4} \\ e^{-2} & 1 & e^{-4} & e^{-2} \\ e^{-2} & e^{-4} & 1 & e^{-2} \\ e^{-4} & e^{-2} & e^{-2} & 1 \end{pmatrix} \qquad \mathbf{A}^{-1} = \begin{pmatrix} \frac{a}{D} & \frac{b}{D} & \frac{b}{D} & \frac{c}{D} \\ \frac{b}{D} & \frac{a}{D} & \frac{c}{D} & \frac{b}{D} \\ \frac{b}{D} & \frac{c}{D} & \frac{a}{D} & \frac{b}{D} \\ \frac{c}{D} & \frac{b}{D} & \frac{b}{D} & \frac{a}{D} \end{pmatrix}$$

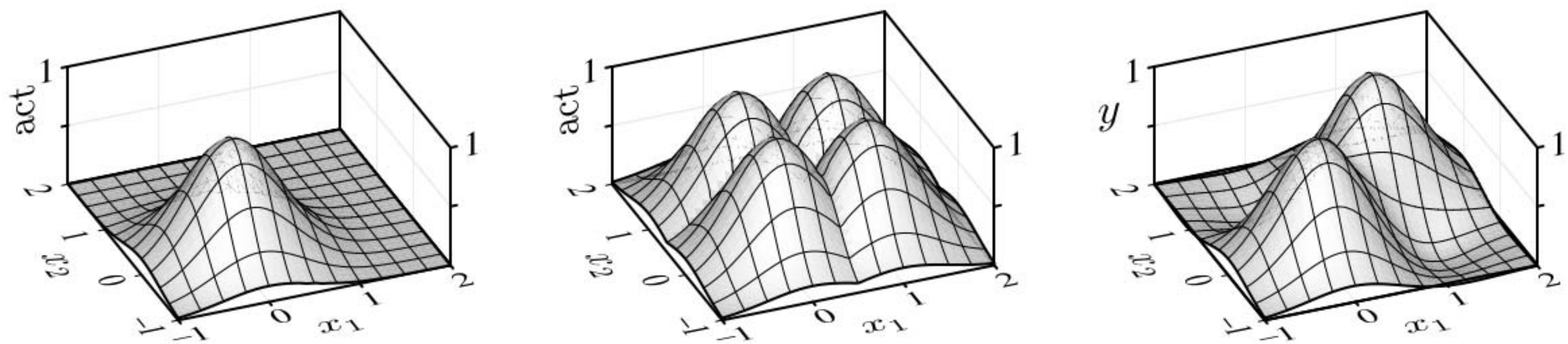
where

$$\begin{aligned} D &= 1 - 4e^{-4} + 6e^{-8} - 4e^{-12} + e^{-16} && \approx 0.9287 \\ a &= 1 \quad - 2e^{-4} + e^{-8} && \approx 0.9637 \\ b &= -e^{-2} + 2e^{-6} - e^{-10} && \approx -0.1304 \\ c &= e^{-4} - 2e^{-8} + e^{-12} && \approx 0.0177 \end{aligned}$$

$$\vec{w}_u = \mathbf{A}^{-1} \cdot \vec{o}_u = \frac{1}{D} \begin{pmatrix} a+c \\ 2b \\ 2b \\ a+c \end{pmatrix} \approx \begin{pmatrix} 1.0567 \\ -0.2809 \\ -0.2809 \\ 1.0567 \end{pmatrix}$$

Initialization (4)

- Simple radial basis function network for the bi-implication $x_1 \leftrightarrow x_2$



single basis function   all basis functions   output

  - Initialization leads already to a perfect solution of the learning task
  - Subsequent training is not necessary

Initialization (5)

- General **Radial Basis Functions**
- Select subset of k training patterns as centers
  - Connection weights for hidden neurons: training patterns
  - Connection weights for output neurons:
    - $$\mathbf{A} = \begin{pmatrix} 1 & out_{v_1}^{(l_1)} & out_{v_2}^{(l_1)} & \cdots & out_{v_k}^{(l_1)} \\ 1 & out_{v_1}^{(l_2)} & out_{v_2}^{(l_2)} & \cdots & out_{v_k}^{(l_2)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & out_{v_1}^{(l_m)} & out_{v_2}^{(l_m)} & \cdots & out_{v_k}^{(l_m)} \end{pmatrix}$$
    - Use the pseudo inverse matrix to compute the weights since system over-determined

Initialization (6)

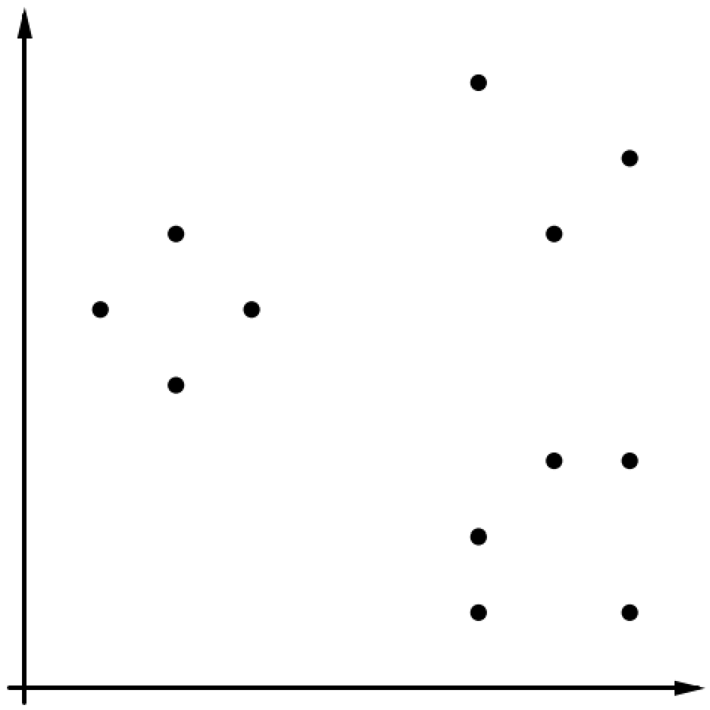
- How to choose the radial basis function centers
  - **All data points** as centers
    - Only radius and output weights need to be determined
    - Output values can be achieved exactly
    - Computing the weights can become infeasible
  - **Random subset**
    - Fast, only radius and output weights need to be determined
    - Performance depends on the choice of data points
  - **Clustering**
    - C-means clustering
    - Learning vector quantization

C-means Clustering (1)

1. Choose a number c of clusters to be found (user input)
2. Initialize the cluster centers randomly
3. Data point assignment
   – Assign each data point to the closest cluster center
4. Cluster center update
   – Compute new cluster centers as the mean vectors of the assigned data points

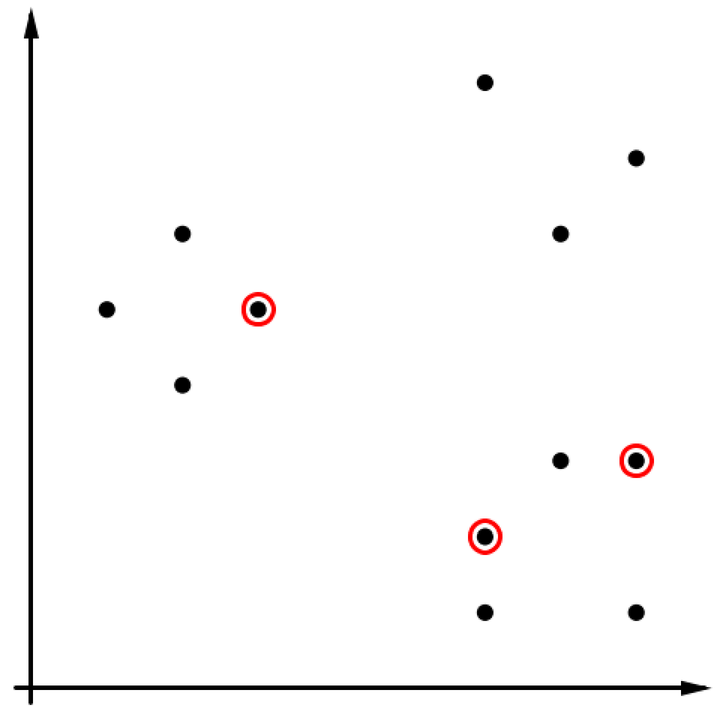
• Repeat steps 3, 4 until clusters centers do not change anymore

C-means Clustering (2)



Data set to cluster.

Choose $c = 3$ clusters.
(From visual inspection, can be difficult to determine in general.)

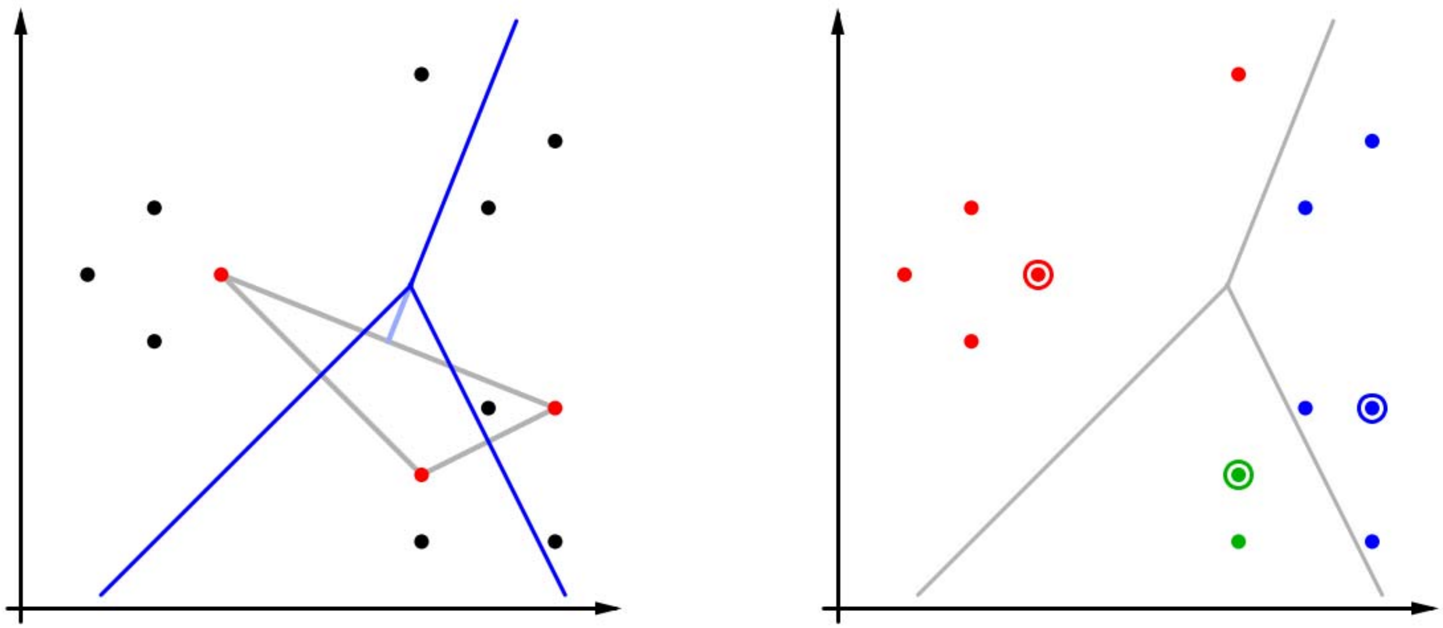


Initial position of cluster centers.

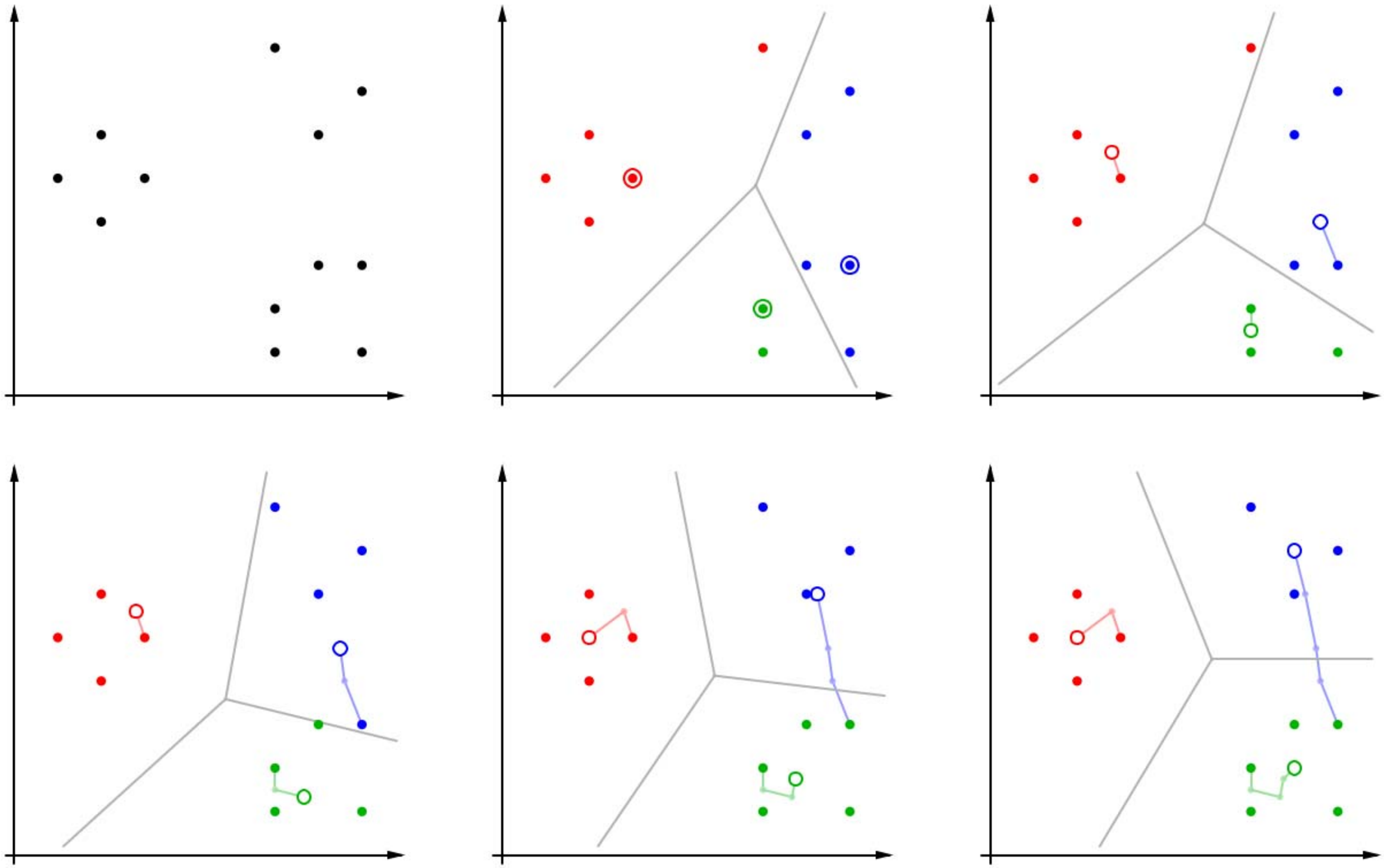
Randomly selected data points.
(Alternative methods include e.g. latin hypercube sampling)

C-means Clustering (3)

- **Delaunay Triangulation**: simple triangle (shown in gray on the left)
- **Voronoi Diagram**: mid-perpendiculars of the triangle's edges (shown in blue on the left, in gray on the right)



C-means Clustering (4)



Training RBFs (1)

- Update rules are analogous to multi-layer perceptrons
  - Weights from the hidden to the output neurons
    - Gradient

$$\vec{\nabla}_{\vec{w}_u} e_u^{(l)} = \frac{\partial e_u^{(l)}}{\partial \vec{w}_u} = -2(o_u^{(l)} - \mathrm{out}_u^{(l)})\, \vec{\mathrm{in}}_u^{(l)},$$

    - Weight update rule

$$\Delta \vec{w}_u^{(l)} = -\frac{\eta_3}{2} \vec{\nabla}_{\vec{w}_u} e_u^{(l)} = \eta_3 (o_u^{(l)} - \mathrm{out}_u^{(l)})\, \vec{\mathrm{in}}_u^{(l)}$$

Training RBFs (2)

– Center coordinates (weights from the input to the hidden neurons)

- Gradient

$$\vec{\nabla}_{\vec{w}_v} e^{(l)} = \frac{\partial e^{(l)}}{\partial \vec{w}_v} = -2 \sum_{s \in \mathrm{succ}(v)} (o_s^{(l)} - \mathrm{out}_s^{(l)}) w_{su} \frac{\partial\, \mathrm{out}_v^{(l)}}{\partial\, \mathrm{net}_v^{(l)}} \frac{\partial\, \mathrm{net}_v^{(l)}}{\partial \vec{w}_v}$$

- Weight update rule

$$\Delta \vec{w}_v^{(l)} = -\frac{\eta_1}{2} \vec{\nabla}_{\vec{w}_v} e^{(l)} = \eta_1 \sum_{s \in \mathrm{succ}(v)} (o_s^{(l)} - \mathrm{out}_s^{(l)}) w_{sv} \frac{\partial\, \mathrm{out}_v^{(l)}}{\partial\, \mathrm{net}_v^{(l)}} \frac{\partial\, \mathrm{net}_v^{(l)}}{\partial \vec{w}_v}$$

- Special case: **Gaussian activation function**

$$\frac{\partial\, \mathrm{out}_v^{(l)}}{\partial\, \mathrm{net}_v^{(l)}} = \frac{\partial f_{\mathrm{act}}(\,\mathrm{net}_v^{(l)}, \sigma_v)}{\partial\, \mathrm{net}_v^{(l)}} = \frac{\partial}{\partial\, \mathrm{net}_v^{(l)}}\, e^{-\frac{\left(\mathrm{net}_v^{(l)}\right)^2}{2\sigma_v^2}} = -\frac{\mathrm{net}_v^{(l)}}{\sigma_v^2}\, e^{-\frac{\left(\mathrm{net}_v^{(l)}\right)^2}{2\sigma_v^2}}.$$

Training RBFs (3)

– Radii of radial basis functions

- Gradient

$$\frac{\partial e^{(l)}}{\partial \sigma_v} = -2 \sum_{s \in \mathrm{succ}(v)} (o_s^{(l)} - \mathrm{out}_s^{(l)}) w_{su} \frac{\partial \, \mathrm{out}_v^{(l)}}{\partial \sigma_v}.$$

- Radii update rule

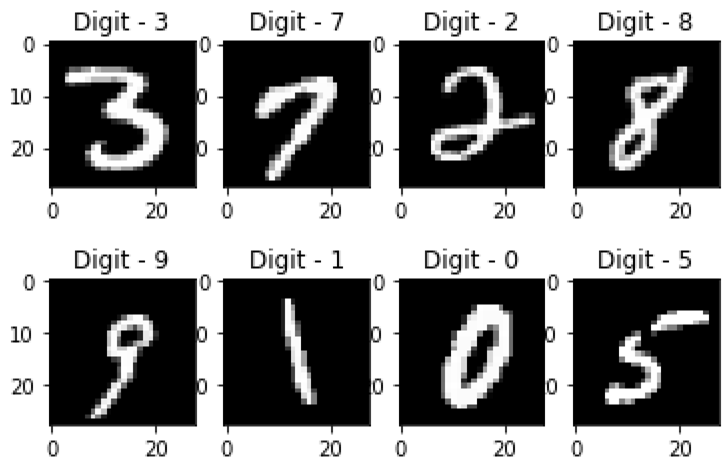
$$\Delta \sigma_v^{(l)} = -\frac{\eta_2}{2} \frac{\partial e^{(l)}}{\partial \sigma_v} = \eta_2 \sum_{s \in \mathrm{succ}(v)} (o_s^{(l)} - \mathrm{out}_s^{(l)}) w_{sv} \frac{\partial \, \mathrm{out}_v^{(l)}}{\partial \sigma_v}.$$

- Special case: **Gaussian activation function**

$$\frac{\partial \, \mathrm{out}_v^{(l)}}{\partial \sigma_v} = \frac{\partial}{\partial \sigma_v} e^{-\frac{\left(\mathrm{net}_v^{(l)}\right)^2}{2\sigma_v^2}} = \frac{\left(\mathrm{net}_v^{(l)}\right)^2}{\sigma_v^3} e^{-\frac{\left(\mathrm{net}_v^{(l)}\right)^2}{2\sigma_v^2}}.$$

Recognition of Handwritten Digits (1)

- Various classifiers
  - Nearest Neighbor (1NN)
  - Decision Tree (C4.5)
  - Multi-Layer Perceptron (MLP)
  - Learning Vector Quantization (LVQ)
  - Radial Basis Function Network (RBF)
  - Support Vector Machine (SVM)



Recognition of Handwritten Digits (2)

- Number of RBF training phases
  - 1 phase: find output connection weights with inverse
  - 2 phase: find RBF centers (e.g. clustering) plus 1 phase
  - 3 phase: 2 phase plus error backpropagation training
- Initialization of RBF centers
  - Random choice of data points
  - C-means Clustering
  - Learning Vector Quantization
  - Decision Tree (one RBF center per leaf)

Recognition of Handwritten Digits (3)

Classifier	Accuracy
Nearest Neighbor (1NN)	97.68%
Learning Vector Quantization (LVQ)	96.99%
Decision Tree (C4.5)	91.12%
2-Phase-RBF (data points)	95.24%
2-Phase-RBF (c-means)	96.94%
2-Phase-RBF (LVQ)	95.86%
2-Phase-RBF (C4.5)	92.72%
3-Phase-RBF (data points)	97.23%
3-Phase-RBF (c-means)	98.06%
3-Phase-RBF (LVQ)	98.49%
3-Phase-RBF (C4.5)	94.83%
Support Vector Machine (SVM)	98.76%
Multi-Layer Perceptron (MLP)	97.59%

- LVQ: 200 vectors (20 per class)
  C4.5: 505 leaves
  c-means: 60 centers(?) (6 per class)
  SVM: 10 classifiers, $\approx$ 4200 vectors
  MLP: 1 hidden layer with 200 neurons
- Results are medians of three training/test runs.
- Error backpropagation improves RBF results.