Chapter 6:
Multilayer Neural Networks

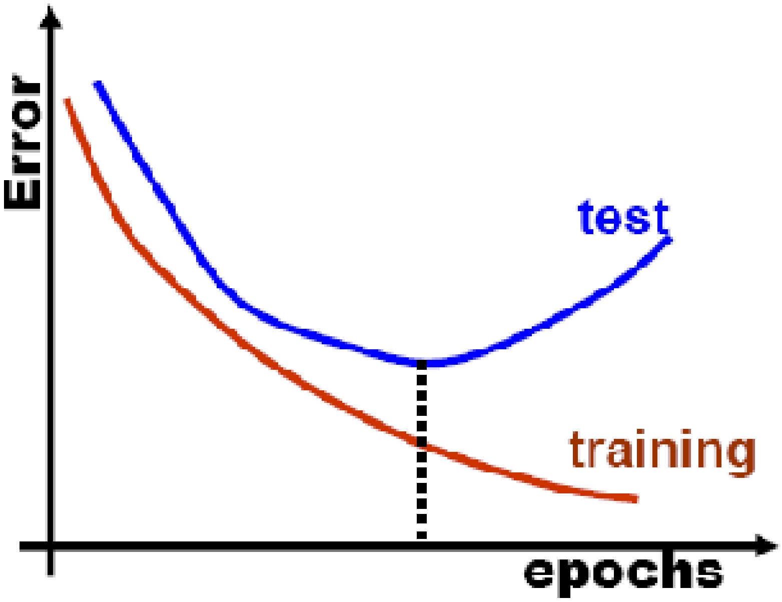
- Training Strategy
- Learning Rate
- Training Algorithm
- Practical techniques
- Analysis of ANN

TRAINING PROTOCOLS

- **Stochastic training:** patterns are chosen randomly from the training set (that is, training data can be considered a random variable) and the network weights are updated after each pattern presentation (following the **case update** model).

- **Batch training:** all training patterns are presented to the network before learning takes place, and (following the **epoch update** model) all the weights are updated after a full pass (epoch) was executed.

- The performance of trained network is usually evaluated with a **test set** of patterns. Excessive training might lead to **over-fitting** the training set.



TRAINING PROCEDURE

1. **Initialize** the weights with a random values (usually within the *[-1,1]* range). Setting all the weights to zero is not recommended.

2. **Feed** the next training specimen (input pattern) to the net, propagate the signal (feed-forward), and calculate the output error as:

   <output error> = <desired output> – <observed output>

3. **Calculate** the change to be applied to each weight by propagating backward the error. The change Δw for weight w is function of the error in the neurode to which w is weighting one input connection.

4a. **Update** the weights if the training strategy is **case updating** (also known as **exemplar updating**):

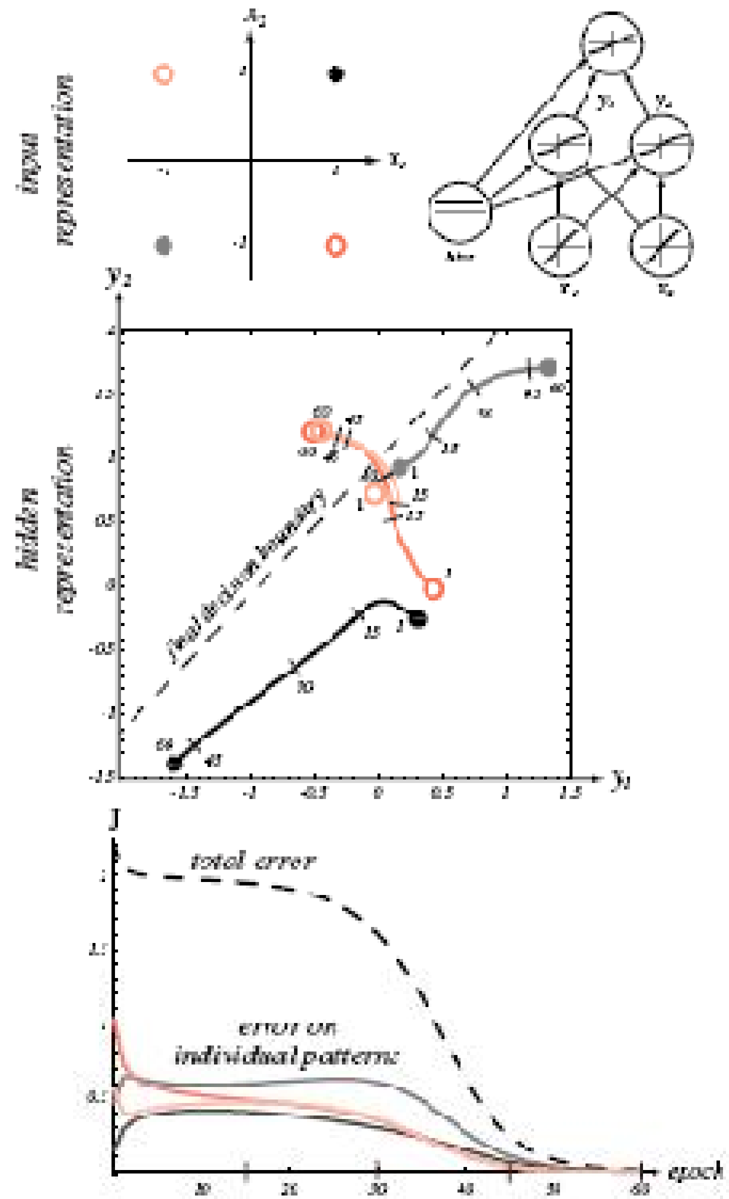
$$W_{new} = W_{old} + \Delta W$$

4b. **Accumulate** the weight change if strategy is **epoch updating**:

$$c = c + \Delta W$$

for adjusting the weights at the end of the current **training pass**, and apply the change after the last training specimen was seen:

$$W_{new} = W_{old} + c$$


input representation
hidden representation
final decision boundary
total error
error on individual patterns
epoch
J

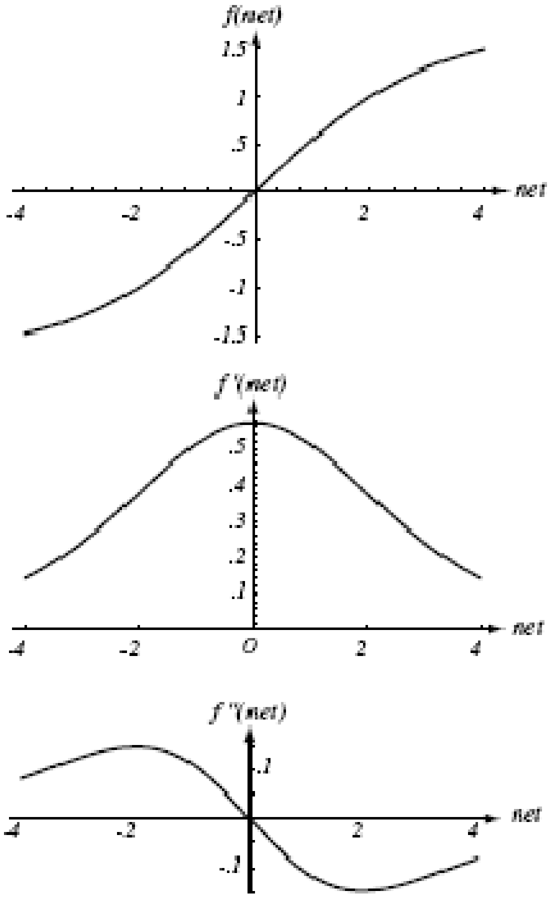


FIGURE 6.14. A useful activation function $f(net)$ is an anti-symmetric sigmoid. For the parameters given in the text, $f(net)$ is nearly linear in the range $-1 < net < +1$ and its second derivative, $f''(net)$, has extrema near $net \simeq \pm 2$. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*.

TRAINING STRATEGY

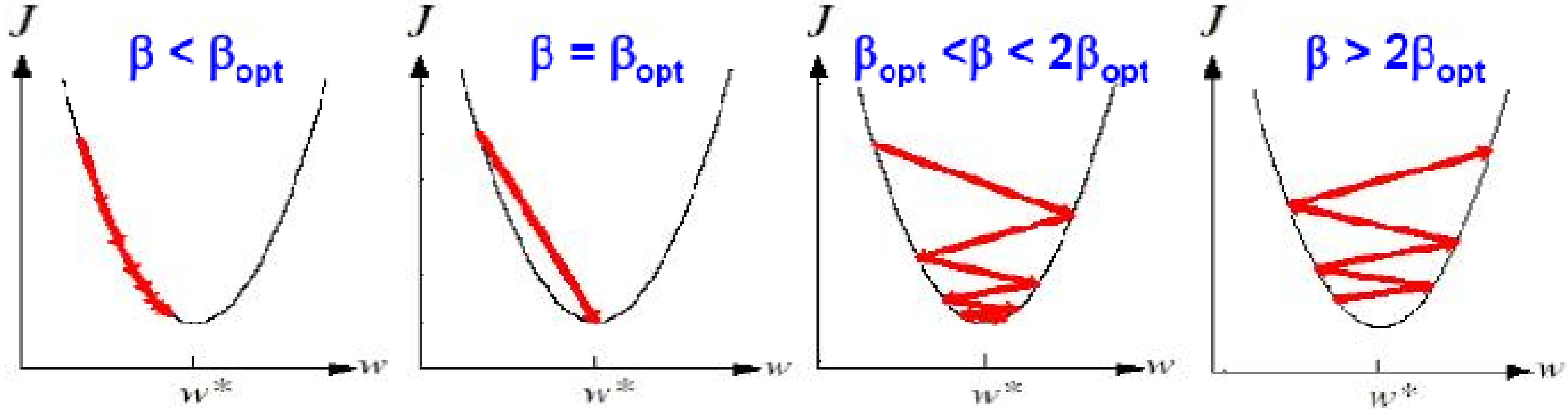
- Network is considered trained, and the training process terminated, if the total cumulative error for a full training pass is below a predefined threshold known as the **training error rate**.
- **Stopping criterion:**
  - The average mean-squared error for the entire training set (of *N* samples) is smaller than some pre-define threshold.

$$J_N = \frac{1}{N}\sum_{s=1}^{N} Error_s = \frac{1}{N}\sum_{s=1}^{N}\left(\frac{1}{2}\sum_{k=1}^{c}\left(y_k^{desired} - y_k^{actual}\right)^2\right)_s \le \theta$$

  - A pre-defined maximum number of training passes has been executed or the maximum time allocated for training was exceeded (meaning that the training failed to reach the error target within pre-defined limits).
- If the training failed, then some of correction methods might be applied:
  - Restart the training with a different set of initial weights.
  - Reconfigure the network by changing the number of middle layers, and/or the number of neurodes in those layers.
  - Weaken the end-of-training criteria by setting a higher error rate.

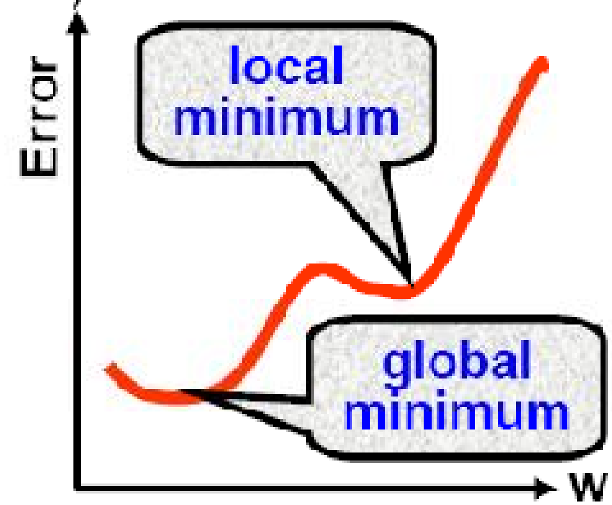
LEARNING RATE

- The learning rate is a positive number ($\beta > 0$) with the following impact:
  - If it is too small, then the convergence might be needlessly slow.
  - If it is too large, the convergence might overshoot (and miss the minimum).
- **Optimal value:** β_{opt} leading to minimum error in one learning step.
- Recommended range: $0 < \beta < 1$.
  - If the training specimen has little or no noise, then $\beta \rightarrow 1$.
  - The noisier the training specimen, the lower should be the value of β.
- Design alternatives:
  - Learning rate is constant throughout the training session.
  - Learning rate is decreased as the training progresses.
- If learning rate is small enough to ensure convergence, then its only influence is on the speed at which the ANN error attains the minimum.



MOMENTUM

- **Problem:** occasionally, when the error slope (i.e. *dJ/dw*) is very small, the training error set stops decreasing and stalls at some value higher than the acceptable level; in this case, the back-propagation network might not train within a reasonable period of time.
- **Idea:** recall the inertial behaviour of physical objects that tend to preserve their motion state unless acted upon by an outside force.
- **Solution:** training failures can be avoided by adding a **momentum** term that will allow the weight vector to continuously change towards the error reduction (and move out of a local minimum).
- **Momentum constant** is a positive integer ($\alpha > 0$) that will enable weight changes even in the absence of error; a term accounting for previous weight change that supplies the needed "push" to go past the local minimum.



- Current change: $\Delta w(s+1) = \Delta w_{bp} + \alpha \Delta w(s)$
- Previous change: $\Delta w(s) = w(s) - w(s-1)$

GENERALIZED DELTA RULE

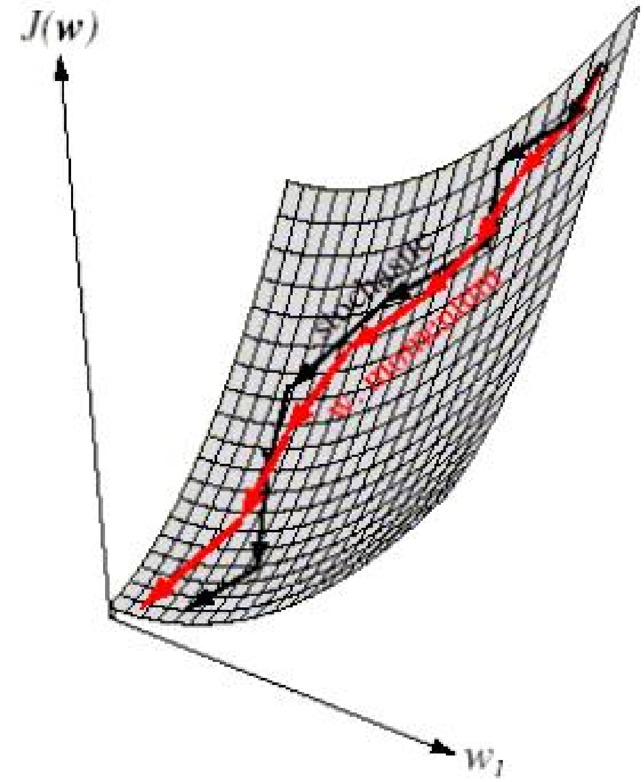
- **Learning rule:**.

$$w(s+1) = w(s) + \Delta w_{bp}(s) + \alpha\left[w(s) - w(s-1)\right]$$

- **Back-propagation error:** ΔW_{bp}

$$\Delta w_{kj} = \beta \delta_k f(net_j)$$

  - **Output Layer:** $\delta_k = f'(net_k)(y_k - z_k)$
  - **Middle layer:** $\delta_j = f'(net_j)\left(\sum_{k=1}^{c} w_{kj}\delta_k\right)$
  - **Input layer:** $f(net_i) = x_i$



The incorporation of momentum into stochastic gradient descent (red arrows) reduces the variation in overall gradient directions and speeds learning.

TRAINING ALGORITHM (1/2)

1. **Set:**
   θ = maximum acceptable error ($\theta > 0$)
   N = number of patterns in the training set ($N > 1$).
   P_{max} = maximum of training passes ($P_{max} > 1$)
2. **Initialize:**
   network weights, w_{kj}, to random values.
   reset pass counter, $p = 0$
3. **Set:** $p \leftarrow p+1$
4. **Pass initialization:**
   set weight variations to 0, $\Delta w_{kj} = 0$
   for epoch update, set cumulative pass errors to 0, $c_{kj} = 0$
   reset pattern (sample) counter, $s = 0$
5. **Set:** $s \leftarrow s+1$
6. **Feed-forward** the pattern s to the net, propagate the values through layers, and calculate the error for each neurode in the output layer.

$$\delta_k = f'(net_k)(y_k - z_k)$$

TRAINING ALGORITHM (2/2)

7. **Back-propagate** the network errors to middle layers.

$$\delta_j = f'(net_j)\left(\sum_{k=1}^{c} w_{kj}\delta_k\right)$$

8. **Calculate** back-propagation weight adjustments, knowing that for input layer $f(net_i) = x_i$.

$$\Delta w_{kj} = \beta\delta_k f(net_j) + \alpha\Delta w_{kj}^{prev}$$

$$\Delta w_{kj}^{prev} = \Delta w_{kj}$$

Case updating: $w_{kj} = w_{kj} + \Delta w_{kj}$

Epoch updating: $c_{kj} = c_{kj} + \Delta w_{kj}$

9. **End-of-pass check:** if ($s < N$), then go to step 5.
10. **Epoch updating:** $w_{kj} = w_{kj} + c_{kj}$
11. **Check termination conditions:**

Success: total error is below pre-defined threshold ($\nabla J \leq \theta$)

Failure: number of training passes exceeds maximum limit ($p \geq P$)

If no termination condition is satisfied, then go to step 3.

PRATICAL TECHNIQUES

❑ Weights:

- Initial weights cannot be set 0 (see back-propagation). Usually set to random values.
- **Uniform learning** is achieved if all weights reach final equilibrium at the same time.
- **Weight decay:** heuristic recommended to avoid over-fitting, and consisting of lowering the weight value after the back-propagation update; it might degrade performance.

$$w(s+1) = (1-\varepsilon)\left(w(s) + \Delta w_{bp}(s) + \alpha \Delta w^{prev}(s)\right)$$

❑ Adding Noise:

- For improving the robustness of the network, a certain amount noise is added to the input patterns (artificially enlarging the training set)
- The smaller the training set, the more the noise added (as high as 60 to 80%).

❑ Learning Rate:

- should be small (0.1 … 0.3) to avoid high oscillation of the back-propagated error (due to large modifications in the cell weights) and to increase the chances to reach the absolute minimum during the gradient-descent learning.
- should decrease when the amount of noise is larger.
- may be decreased dynamically as the learning progresses.

❑ Momentum Constant:

- should be kept high (0.5 … 0.9) to compensate for lower learning constant (and to speed-up the training). Increased above 0.9 may adversely affect the learning.

IMPLEMENTATION ISSUES

- **Input:**
  - **Training set** should provide enough patterns coverage.
  - Input patterns might be scaled to fit a standard size.
  - Training phase is more effective if noise is added to the training specimen (i.e. to enhance the patterns coverage).
  - Classification accuracy should be verified with a comprehensive **test set** (including no patterns from the training set).

- **Architecture:**
  - Networks with two hidden layers can model almost any problem.
  - Network size (number of layers and/or neurodes):
    - Network too large: **over-fitting** the training data.
    - Network too small: **over-generalization** of training data.
  - To improve performance, add the support of a symbolic classifier.
    - ANN classifier are faster but not better than symbolic classifiers.
    - Input information is augmented with pattern features extracted by the symbolic classifier: local and/or global features.
    - Classification accuracy is increased: patterns partially classified by the ANN classifier end up being fully classified with the additional help provided by the symbolic classifier.

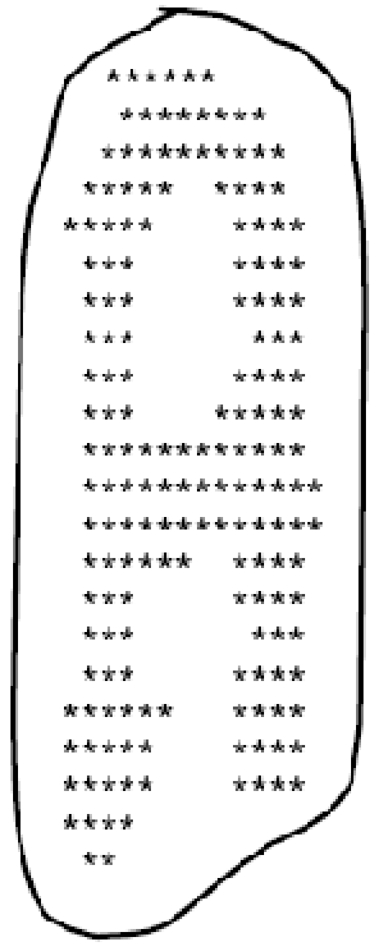
NETWORK MODELS

- **Feed-forward Network:**
  - **Perceptrons**: no hidden layers
  - **Multi-layer networks**: with hidden layers.
  - **Convolutional Networks**: type of multi-layer networks embedding prior knowledge into the network.
    - **Time Delay Neural Networks** (TDNN): hidden units accept inputs only from a restricted range of input cells (i.e. cells having a similar spatial shift). Hidden units at "delayed" locations (for instance, shifted to right) accept inputs from the input cell that are similarly shifted.

- **Recurrent Network:**
  - **Hopfield Networks**: networks where all units are both input and output units, connected by bi-directional links and having symmetric weights (that is, $w_{ik} = w_{ki}$).
    - Behaviour follows the **associative memory** model (i.e. network output is that of the training pattern that most closely resembles the current input).
  - **Boltzmann Machines**: also use symmetric weights, but some units are neither input nor output units.
    - Activation function is stochastic (i.e. the probability of the output being 1 is function of weighted input).

EXAMPLE OF PATTERN RECOGNITION APPLICATION

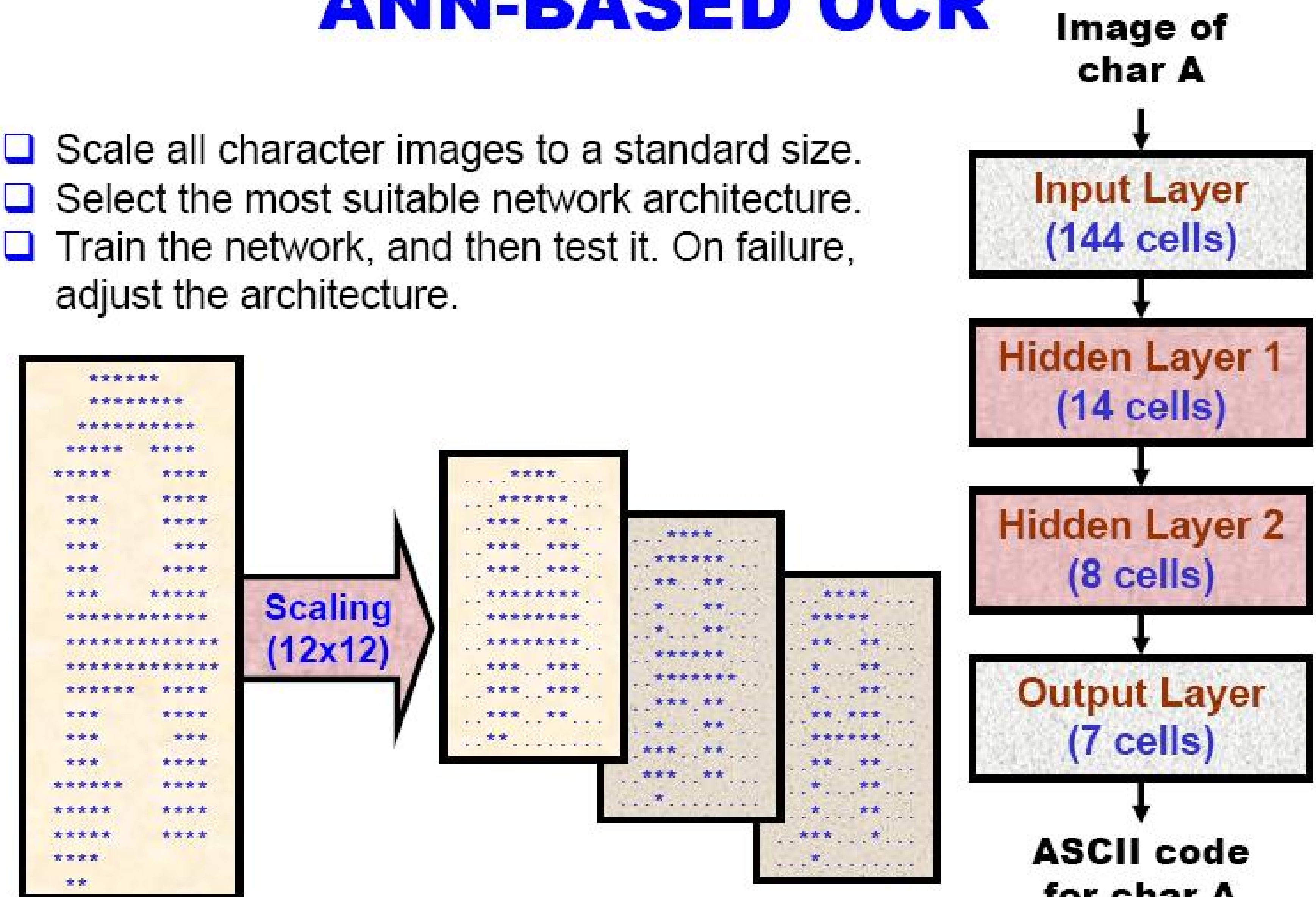
- **Optical Character Recognition (OCR):** recognizing and classifying digitized images of written [English] characters and/or text.
  - Most difficult: recognizing hand-written characters and multi-lingual texts.
  - Relies on natural language processing techniques for increasing the recognition accuracy.



- **OCR for English text:**
  - Characters to classify: 62 (26 upper-case letters, 26 lower-case letters, 10 digits).
  - Additional symbols: punctuation marks, math and logic operators, currency symbols, etc.

ANN-BASED OCR

- Scale all character images to a standard size.
- Select the most suitable network architecture.
- Train the network, and then test it. On failure, adjust the architecture.



ANALYSIS OF ANN (1/2)

- **Expressiveness:**
  - Neural nets are attribute-based representations without the expressive power of general logic representations.
  - It was concluded that the number of hidden units required to represent a Boolean function of **n** inputs is $\mathbf{2^n/n}$.
  - Designing a good topology is yet based on designer's experience.

- **Computational Efficiency:**
  - Depends on the amount of computation time required to train the network for fitting a given set of sample patterns.
  - Worst-case: number of training epochs is exponential in **n**, the number of inputs.
  - In practice: convergence rate is highly variable.

- **Generalization:**
  - Neural nets generalize well functions/patterns for which they are well-suited (i.e. without interactions between inputs, and where the output varies smoothly with the input).

ANALYSIS OF ANN (2/2)

- **Sensitivity to Noise:**
  - Neural nets are very tolerant to noise in the input data (because they do non-linear regression).
  - Do not provide probability distribution on the output values.

- **Transparency:**
  - Neural nets follow the black box model.
  - They offer no explanation on why the given output is reasonable (they do not explain through logical derivation their decisions).

- **Prior Knowledge:**
  - Quality of the output is highly dependent on the level of knowledge acquired through training (i.e. the type and the numbers of the patterns used for training).
  - Tailoring the network is essentially an heuristic technique.

EVALUATION OF ANN

- **Advantages:**
  - Neural nets perform at their best in problems of “memory association” (that is, clustering and classification).
  - For a given input, the output is obtained relatively quickly.
  - Benefit from parallel processing hardware.
  - Easy to use (few parameters to set, algorithms easy to implement).
  - Low sensitivity to noisy input and graceful degradation (small changes in input does not normally cause a change in output).

- **Disadvantages:**
  - No explanation is provided for the output yielded from a given input.
  - Large networks (many cells and layers) require a long training time,
  - New training usually overwrites old representations (unless they are interleaved with the new patterns).
  - Modeling a problem through a neural network implementation is highly heuristic (without a well-defined criteria for selecting the network architecture).

CONCLUDING REMARKS

- Multi-layer nonlinear neural networks (nets with more than two layers of modifiable weights) trained by gradient descent method such as back-propagation perform a maximum-likelihood estimation of the pattern classification based on the current weights values within the model defined by the network topology.
  - Problem model: network topology.
  - Arbitrary decision boundaries: enabled by the presence of hidden units.
  - Simple learning algorithm for feed-forward back-propagation networks: generalized delta rule (based on LMS procedure of Widrow and Hoff).

- Despite its limited plausibility as a psychological model for learning and human memory, the artificial neural networks are widely used and their error-correction learning model has been very important in the brief history of connectionism.