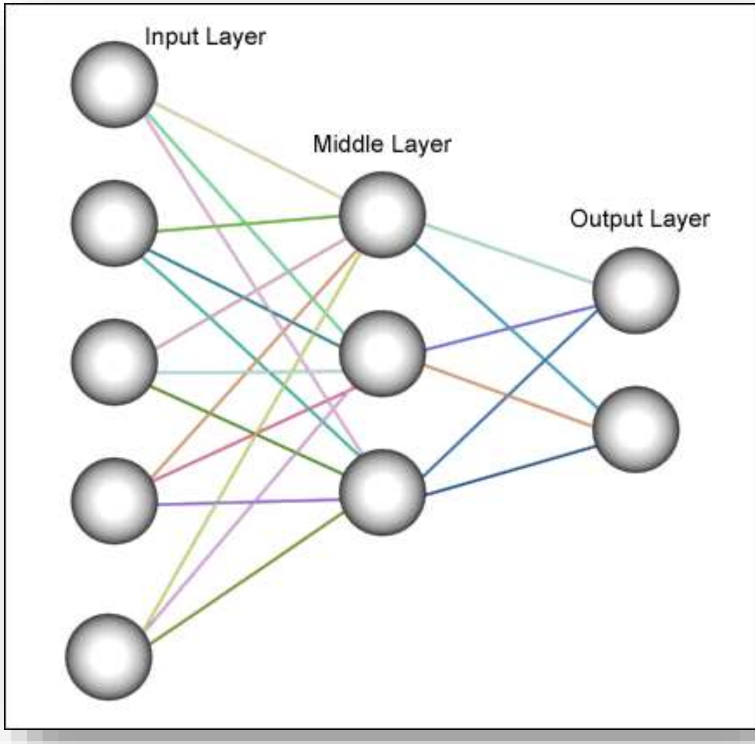
IN THE NAME OF ALLAH

Neural Networks

Shahrood University of Technology
Hossein Khosravi

Multi Layer Perceptron



Training Modes

- **Incremental mode (on-line, sequential, stochastic, or per-observation):**
  - Weights updated after each instance is presented
  - Order of presentation should be randomized
  - Benefits: less storage, stochastic search through weight space helps avoid local minima.
  - Disadvantage: hard to establish theoretical convergence conditions.
- **Batch mode (off-line or per-epoch):**
  - Weights updated after all the patterns are presented
  - Benefits: accurate estimate of the gradient, convergence to local minimum is guaranteed under simpler conditions

Stopping criterions

- Sensible stopping criterions:
  - Total mean squared error change
    - Back-prop is considered to have converged when the absolute rate of change in the average squared error per epoch is sufficiently small (in the range [0.01, 0.1]).
  - Generalization based criterion:
    - After each epoch the NN is tested for generalization using a different set of examples (validation set).
    - If the generalization performance is adequate then stop.

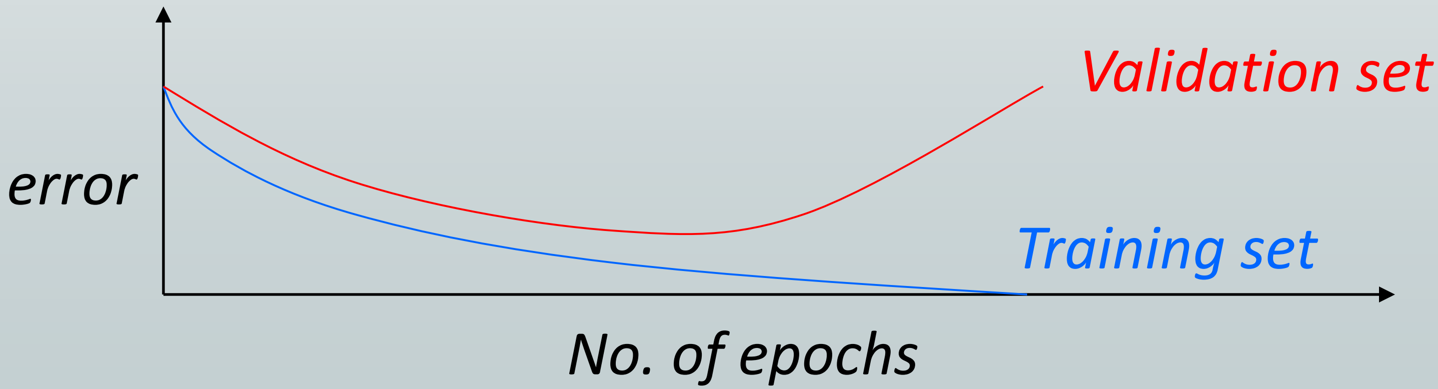
Use of Available Data Set for Training

- **Available data sets are normally split into three sets:**
  - Training set
    - Use to update the weights. Patterns in this set are repeatedly in random order. The weight update equation are applied after a certain number of patterns.
  - Validation set
    - Use to decide when to stop training only by monitoring the error.
  - Test set
    - Use to test the performance of the neural network. It should not be used as part of the neural network development cycle.

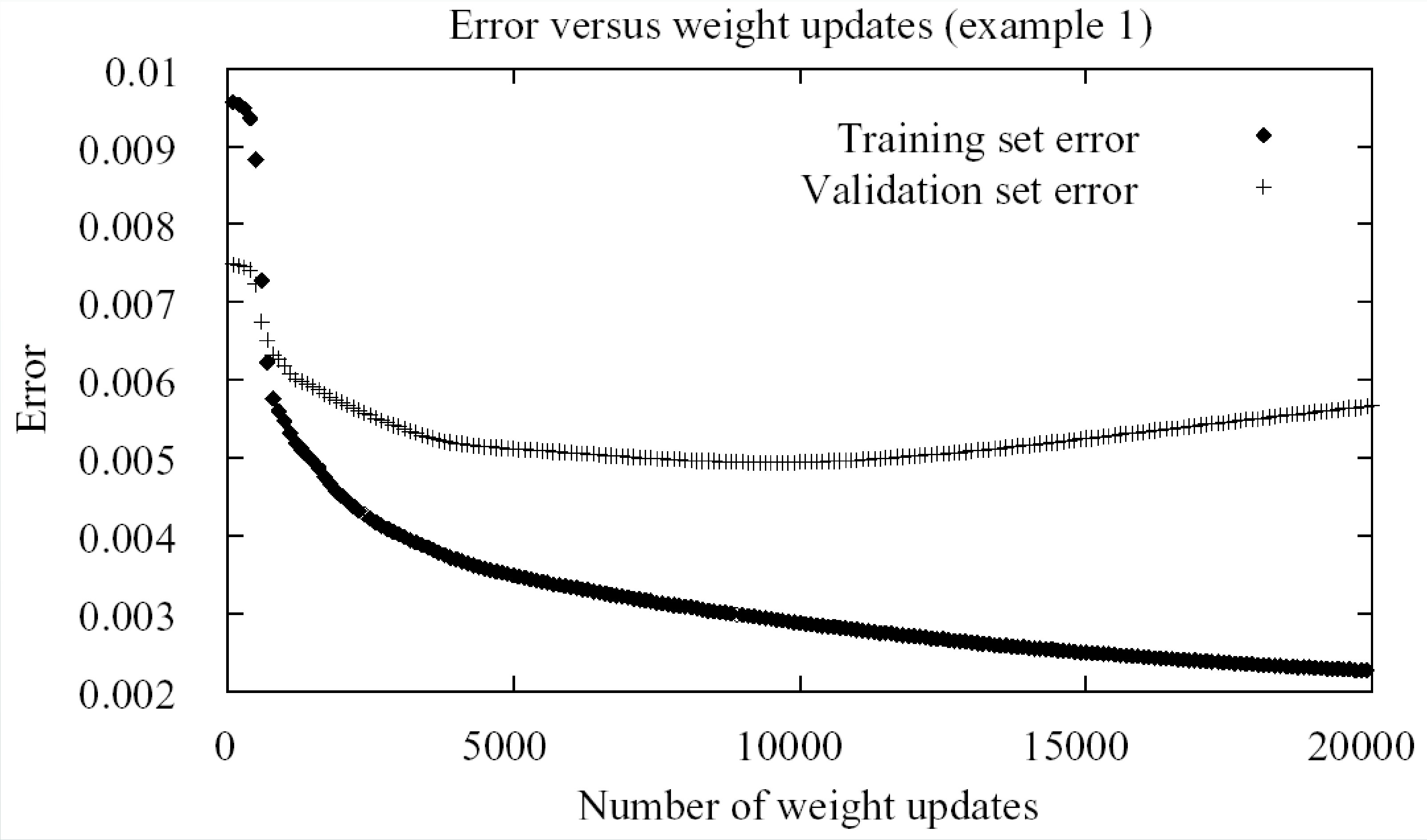
Earlier Stopping - Good Generalization

- Running too many epochs may **overtrain** the network and result in overfitting and perform poorly in generalization.

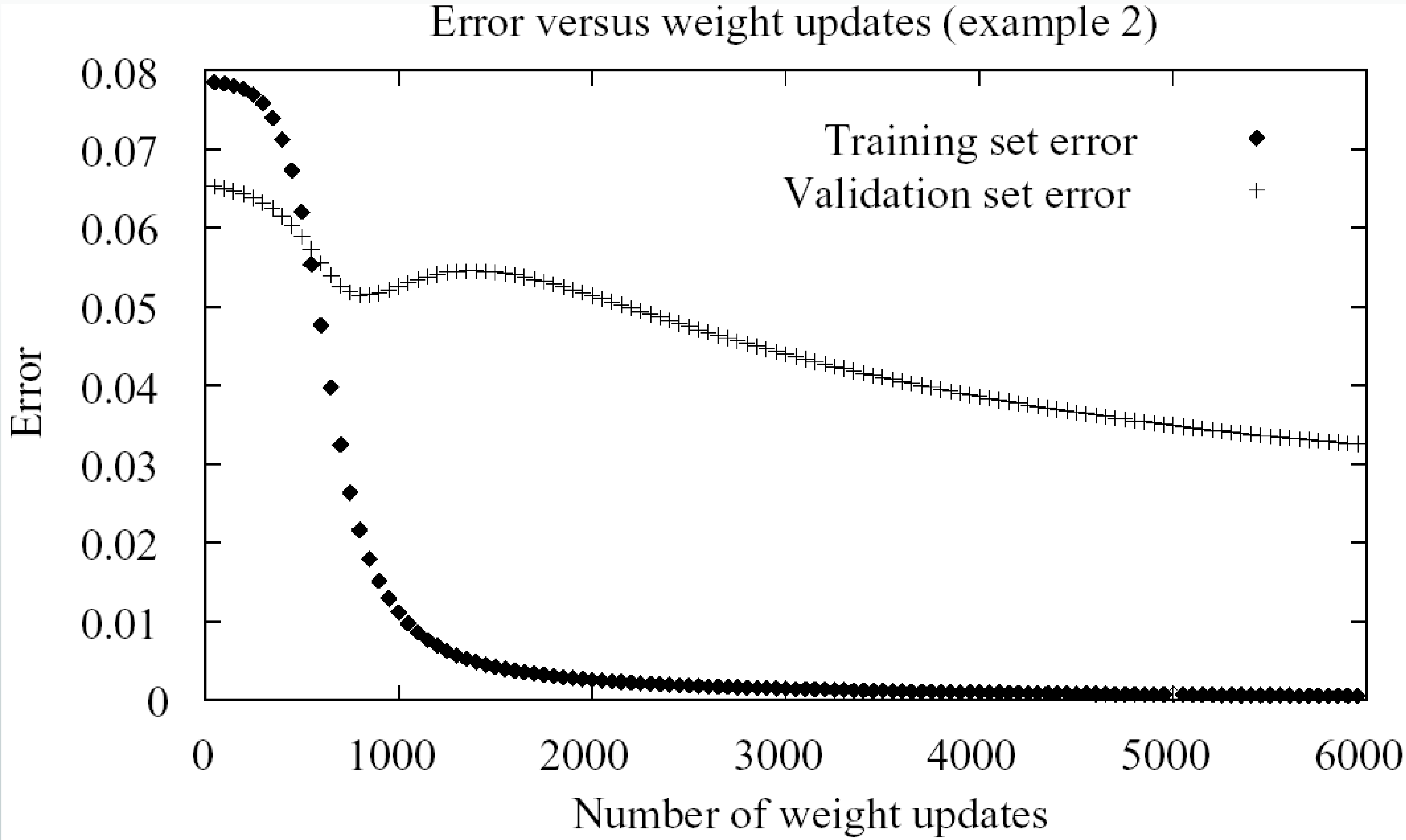
- Keep a hold-out validation set and test accuracy after every epoch. Maintain weights for best performing network on the validation set and stop training when error increases beyond this.



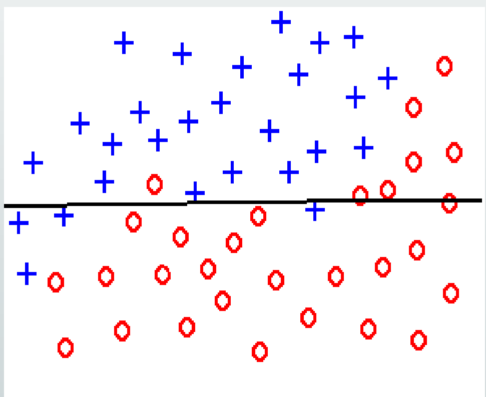
Overfitting in ANNs (1/2)



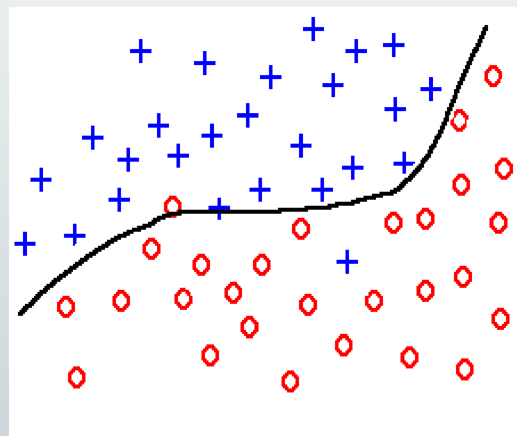
Overfitting in ANNs (2/2)



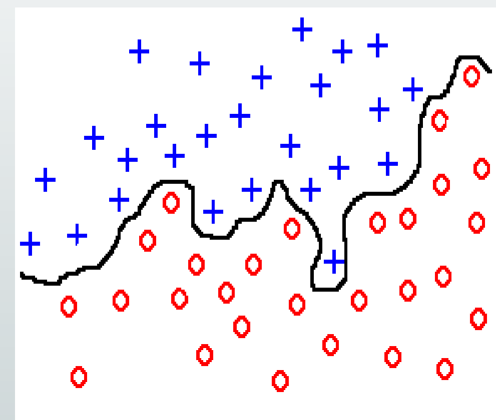
Overfitting and underfitting



underfitting



good fit



overfitting

Model Selection

- Too few hidden units prevent the network from adequately fitting the data and learning the concept.
- Too many hidden units leads to overfitting.
- Unfortunately there is no mathematical theorem
- But some rule of thumb
  - More classes, more hidden units
  - More Training samples, more hidden units
  - Try at least 3 structure, e.g. 20, 40 and 60 hidden units
  - Fortunately MLP is not so sensitive to the number of hidden units

NN Design

- Data representation
- Network Topology
- Network Parameters
- Training

Data Representation

- Data representation **depends on the problem**. In general NNs work on **continuous (real valued) attributes**. Therefore symbolic attributes are encoded into continuous ones.
- Attributes of different types may have different ranges of values which affect the training process. **Normalization** may be used, like the following one which scales each attribute to assume values between 0 and 1.

$$x_i = \frac{x_i - \mathrm{min}_\mathrm{i}}{\mathrm{max}_i - \mathrm{min}_\mathrm{i}}$$

for each value x_i of attribute i, where min_i and max_i are the minimum and maximum value of that attribute over the training set.

Network Topology

- The number of layers and neurons depend on the specific task. In practice this issue is solved by trial and error.

- Two types of adaptive algorithms can be used:
  - Start from a large network and successively remove some neurons and links until network performance degrades.
  - Begin with a small network and introduce new neurons until performance is satisfactory.

Network parameters

- How are the weights initialized?
- How is the learning rate chosen?
- How many hidden layers and how many neurons?
- How many examples in the training set?

Initialization of weights

- In general, initial weights are randomly chosen, with typical values between -1.0 and 1.0 or -0.5 and 0.5.
- If some inputs are much larger than others, random initialization may bias the network to give much more importance to larger inputs. In such a case, weights can be initialized as follows:

$$w_{ij} = \pm \frac{1}{2m} \sum_{i=1,\ldots,m} \frac{1}{|x_i|}$$

For weights from the input to the first layer

$$w_{jk} = \pm \frac{1}{2n} \sum_{j=1,\ldots,n} \frac{1}{\varphi\left(\sum_i w_{ij} x_i\right)}$$

For weights from the first to the second layer

Choice of learning rate

- The right value of η depends on the application. Values between 0.1 and 0.9 have been used in many applications.

- It is common to start with large values and decrease monotonically.
  - Start with 0.9 and decrease every 5 epochs
  - Use a Gaussian function
  - $\eta = 1/k$
  - ...

Size of Training set

- Rule of thumb:
  - The number of training examples should be at least five to ten times the number of weights of the network.
- Other rule:

$$N > \frac{|W|}{(1-a)}$$

|W|= number of weights
a = expected accuracy

Example: Fish Sorting

- Problem: Sorting incoming fish on a conveyor belt according to species
- Assume that we have only two kinds of fish:
  - Sea bass
  - Salmon

The Problem

What ***we*** see

0	3	2	5	4	7	6	9	8
3	0	1	2	3	4	5	6	7
2	1	0	3	2	5	4	7	6
5	2	3	0	1	2	3	4	5
4	3	2	1	0	3	2	5	4
7	4	5	2	3	0	1	2	3
6	5	4	3	2	1	0	3	2
9	6	7	4	5	2	3	0	1
8	7	6	5	4	3	2	1	0

What a ***computer*** sees

Typical Decision Process

- Fish Face Recognition?
- Salmon tastes better?
- What kind of information can distinguish one species from the others?
  - length, width, weight, number and shape of fins, tail shape, etc.

- What can cause problems during sensing?
  - lighting conditions, position of fish on the conveyor belt, camera noise, etc.

- What are the steps in the process?
  - capture image → isolate fish → take measurements → make decision

Example: Feature Selection

- Assume a fisherman (domain knowledge) told us that a sea bass is generally longer than a salmon.
  - We can use length as a feature and decide between sea bass and salmon according to a threshold on length.
  - How can we choose this threshold?

Example: Feature Selection

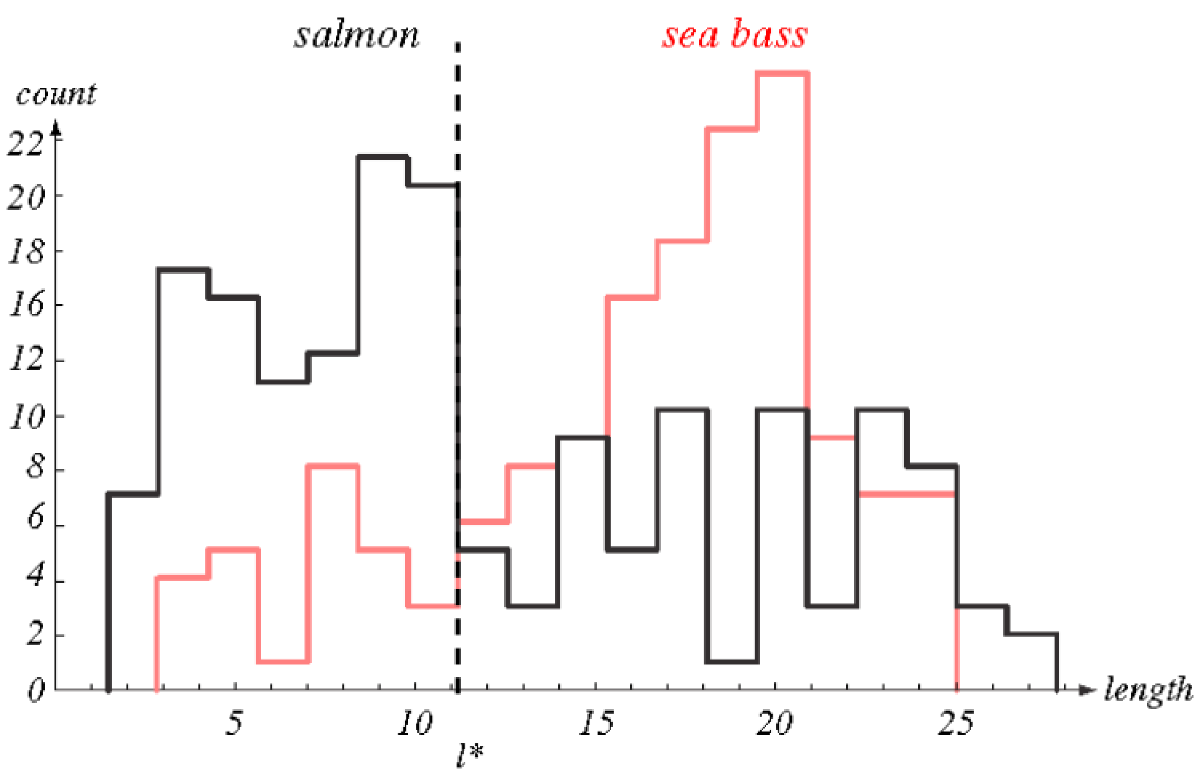


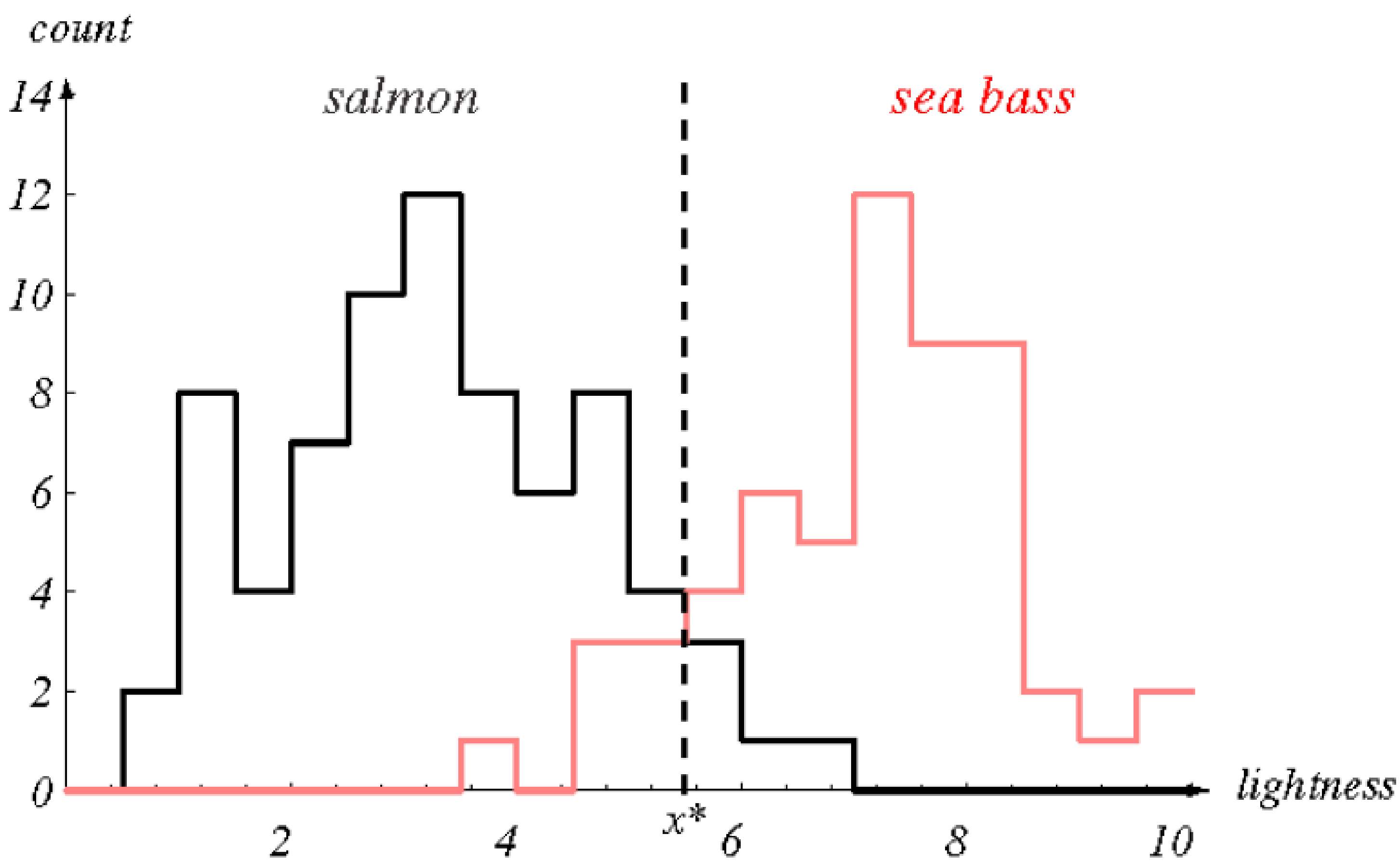
Figure 2: *Histograms* of the length feature for two types of fish in *training samples*. How can we choose the threshold l^* to make a reliable decision?

Example: Feature Selection

- Even though sea bass is longer than salmon on the average, there are many examples of fish where this observation does not hold.

- Try another feature: average lightness of the fish scales.

Example: Feature Selection



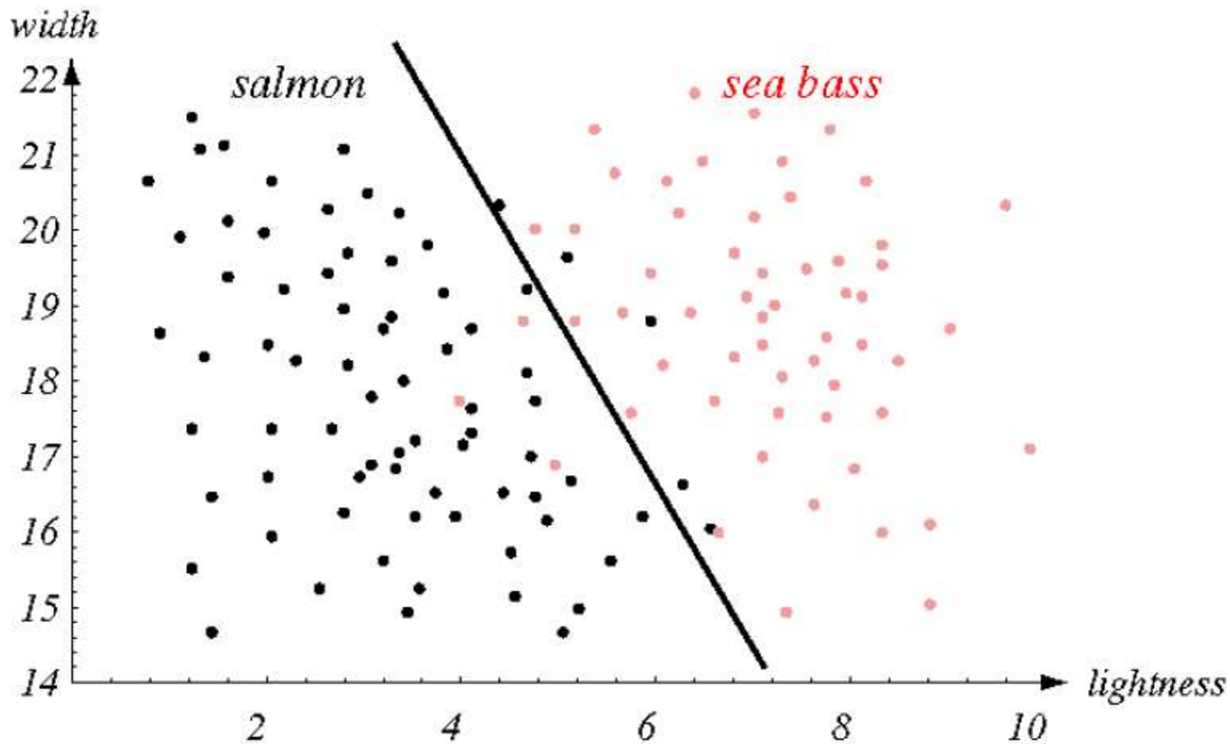
Histogram of the lightness feature for two types of fish in training samples. It looks easier to choose the threshold x^* but we still cannot make a perfect decision.

Example: Multiple Features

- We can use two features in our decision:
  - lightness: $\boldsymbol{x}_1$
  - length: $\boldsymbol{x}_2$

- Each fish image is now represented as a point (feature vector) $X = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ in a two-dimensional feature space

Example: Multiple Features



Scatter plot of lightness and length features for training samples. We can draw a decision boundary to divide the feature space into two regions.

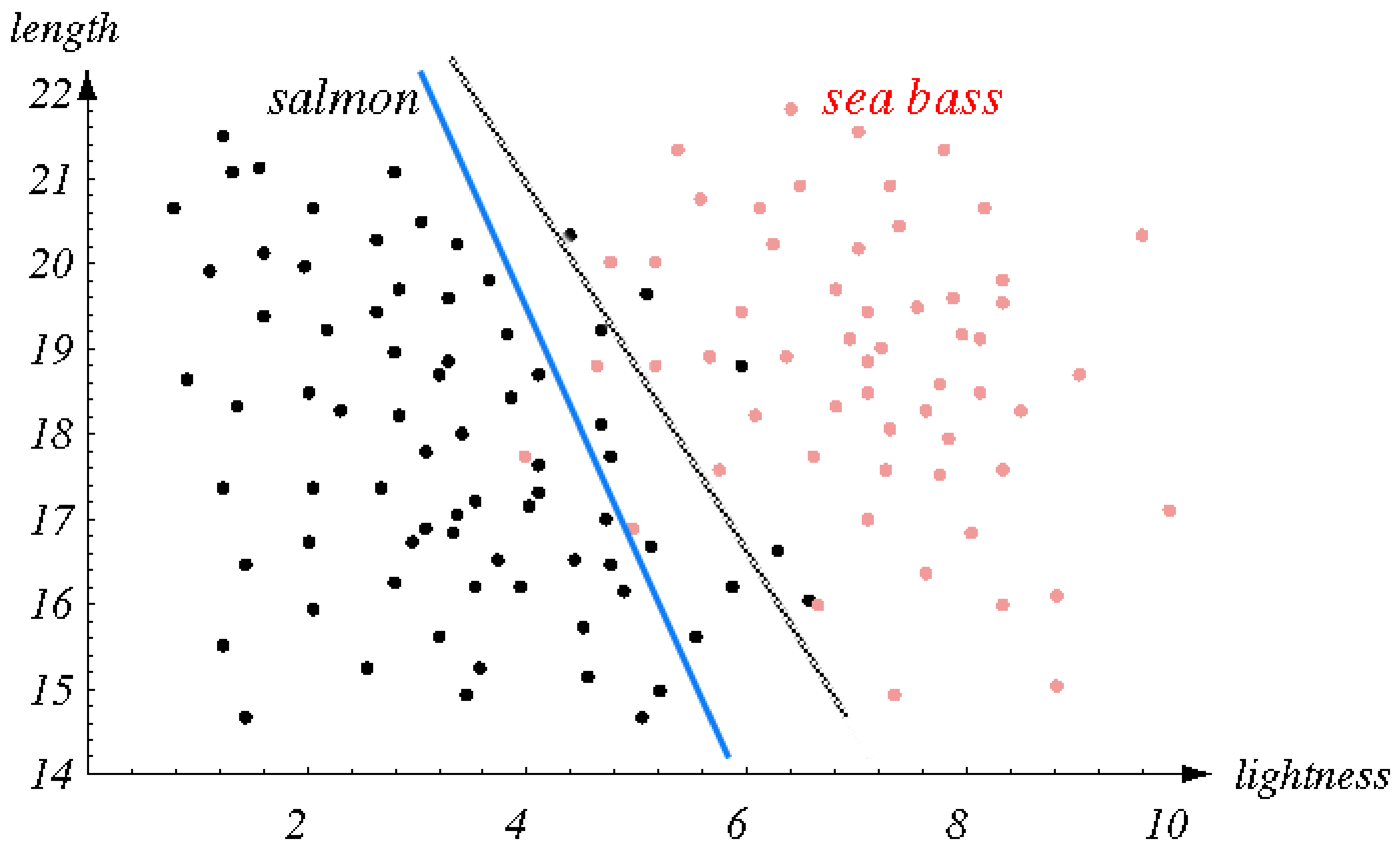
Example: Cost of Error

- We should also consider costs of different errors we make in our decisions.

- For example, if the fish packing company knows that:
  - Customers who buy salmon will object vigorously if they see sea bass in their cans.
  - Customers who buy sea bass will not be unhappy if they occasionally see some expensive salmon in their cans.

- How does this knowledge affect our decision?

Example: Multiple Features



Scatter plot of lightness and length features for training samples with distinct costs. (blue is the new classifier)

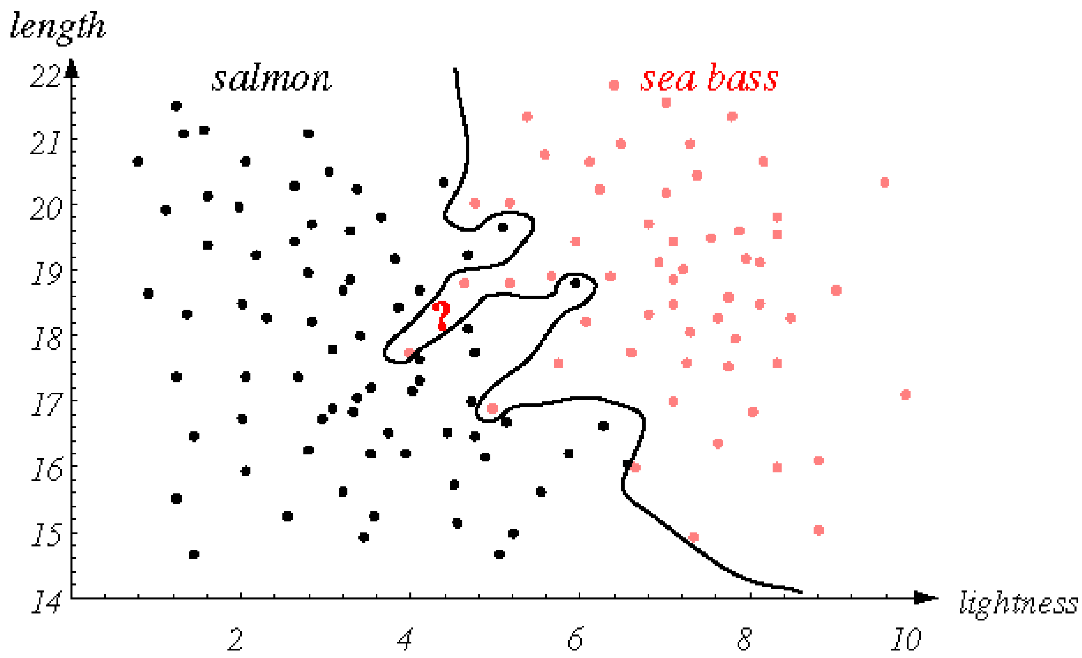
Remarks on Multiple Features

- Does adding more features always improve the results?
  - Avoid unreliable features.
  - Be careful about correlations with existing features.
  - Be careful about measurement costs.
  - Be careful about noise in the measurements.

- Is there some curse for working in very high dimensions?
  - Curse of dimensionality

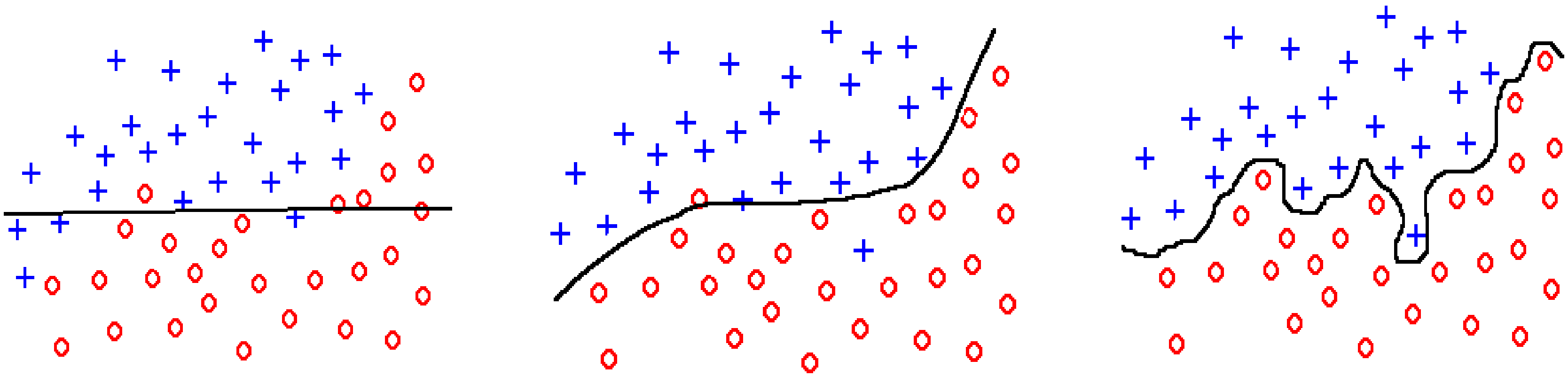
Example: Decision Boundaries

- Can we do better with another decision rule?
- More complex models result in more complex boundaries.



We may distinguish training samples perfectly but how can we predict how well we can generalize to unknown samples?

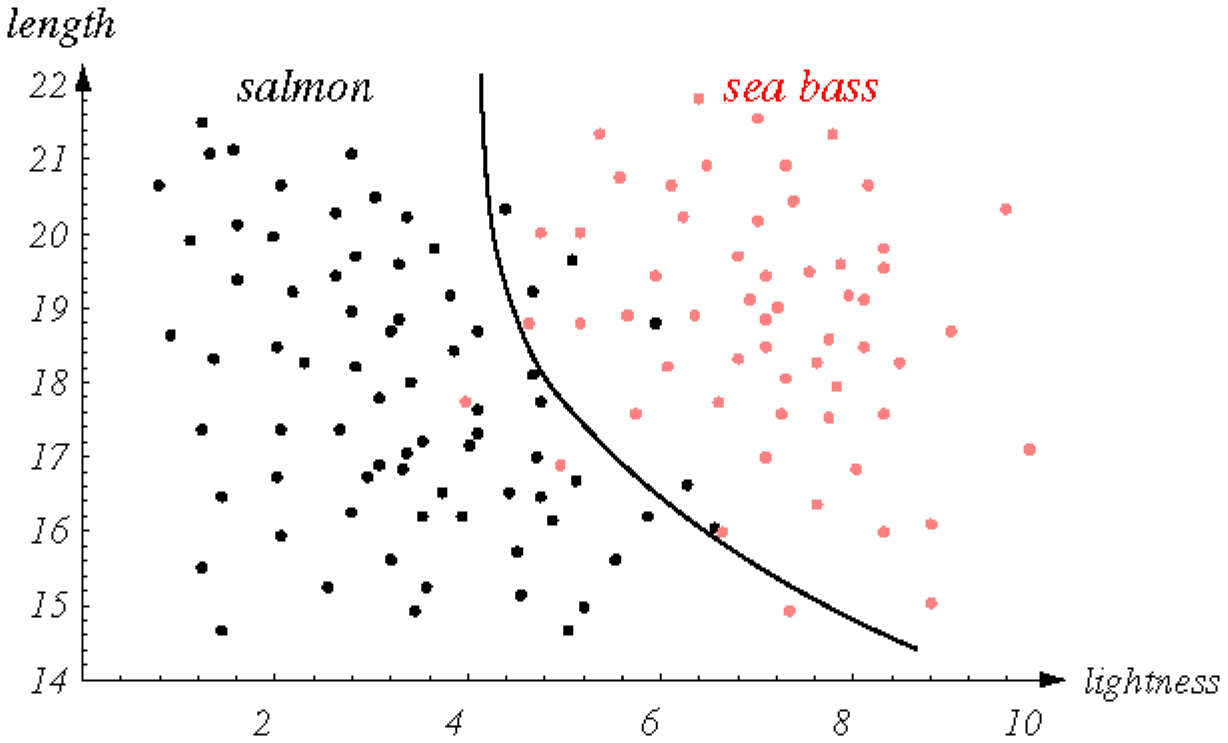
Overfitting and underfitting



underfitting good fit overfitting

Example: Decision Boundaries

- How can we manage the tradeoff between complexity of decision rules and their performance to unknown samples?



Different criteria lead to different decision boundaries.

A generalized delta rule

- If η is **small** then the algorithm learns the weights very **slowly**, while if η is **large** then the large changes of the weights may cause an unstable behavior with **oscillations** of the weight values.
- A technique for tackling this problem is the introduction of a **momentum term** in the delta rule which **takes into account previous updates**.
- **Generalized Delta rule**:
  - $\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) + \alpha \Delta w_{ji}(n-1)$
  - α is the momentum constant; it has the effect of negative feedback
  - Momentum term accelerates the descent in steady downhill directions

Remarks on momentum

- We can rewrite this equation as a time series with index t . It goes from 0 to the current time n:

$$\Delta w_{ji}(n) = \eta \sum_{t=0}^{n} \alpha^{n-t} \delta_j(t) y_i(t) = -\eta \sum_{t=0}^{n} \alpha^{n-t} \frac{\partial E(t)}{\partial w_{ji}(t)}$$

- The current adjustment $\Delta w_{ji}(n)$ is the sum of an exponentially weighted time series.
- To be convergent, the momentum constant must be restricted in the range $0 \leq \alpha < 1$
- When successive $\frac{\partial E(t)}{\partial W_{ji}(t)}$ take the same sign:
  - Weight update is accelerated (speed up downhill).
- When successive $\frac{\partial E(t)}{\partial W_{ji}(t)}$ have different signs:
  - Weight update is damped (stabilize oscillation)

Applications of FFNN

Classification, pattern recognition:

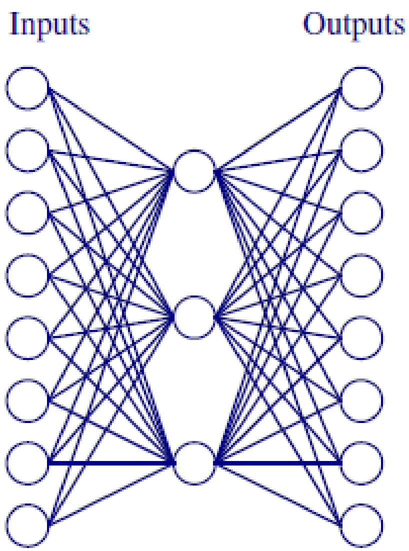
- FFNN can be applied to tackle non-linearly separable learning tasks.
  - Recognizing printed or handwritten characters
  - Face recognition
  - Classification of loan applications into credit-worthy and non-credit-worthy groups
  - Analysis of sonar radar to determine the nature of the source of a signal

Regression and forecasting:

- FFNN can be applied to learn non-linear functions (regression) and in particular functions whose inputs is a sequence of measurements over time (time series).

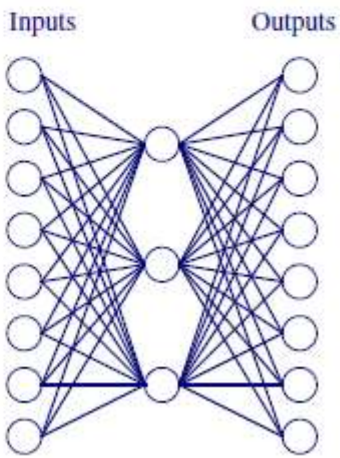
Compression!

Learning Hidden Layer Representations



Input		Output
10000000	→	10000000
01000000	→	01000000
00100000	→	00100000
00010000	→	00010000
00001000	→	00001000
00000100	→	00000100
00000010	→	00000010
00000001	→	00000001

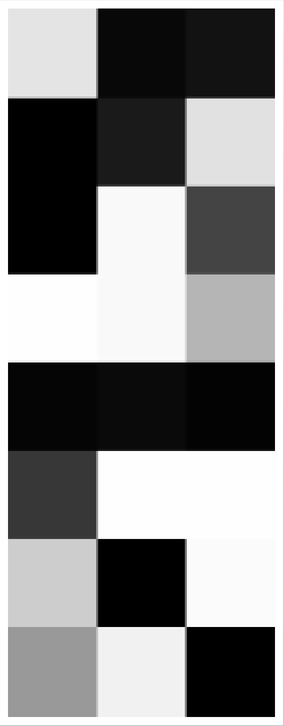
Learned Hidden Layer Representations



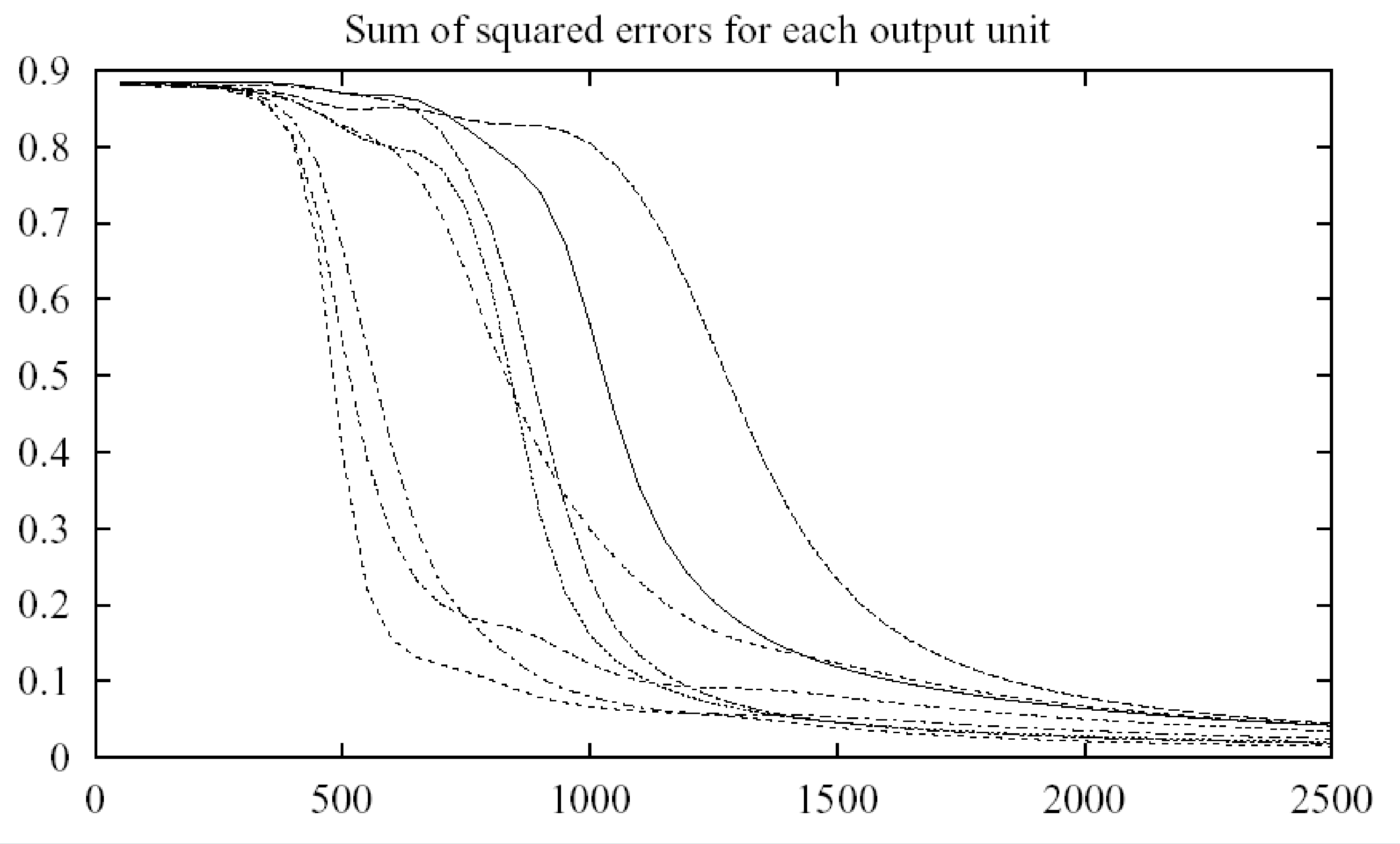
Input		Hidden Values				Output
10000000	→	.89	.04	.08	→	10000000
01000000	→	.01	.11	.88	→	01000000
00100000	→	.01	.97	.27	→	00100000
00010000	→	.99	.97	.71	→	00010000
00001000	→	.03	.05	.02	→	00001000
00000100	→	.22	.99	.99	→	00000100
00000010	→	.80	.01	.98	→	00000010
00000001	→	.60	.94	.01	→	00000001

Learned Hidden Layer Representations

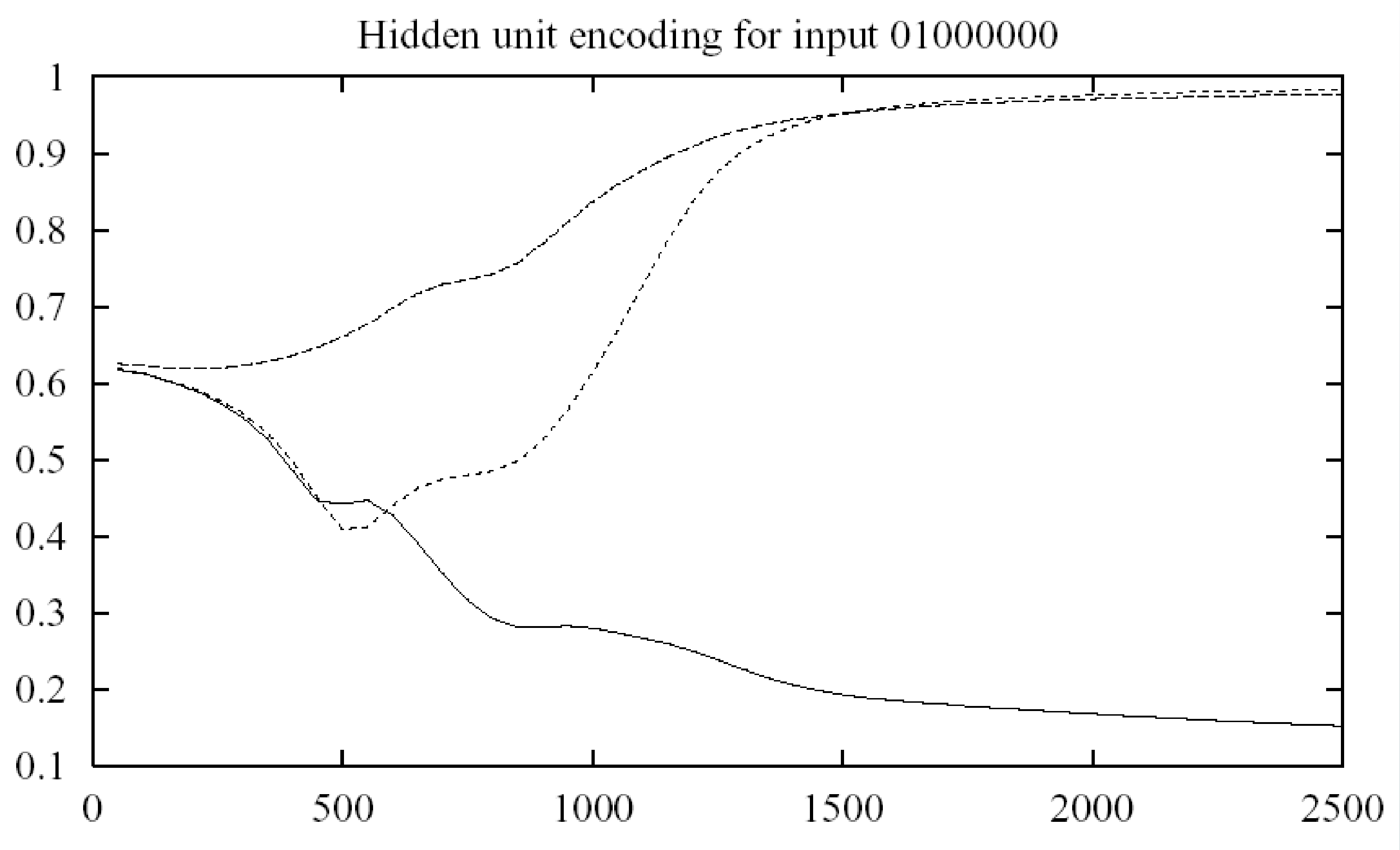
- Learned encoding is similar to standard 3-bit binary code.
- **Automatic** discovery of useful hidden layer representations is a key feature of ANN
- Note: The hidden layer representation is **compressed**



Training (1/3)

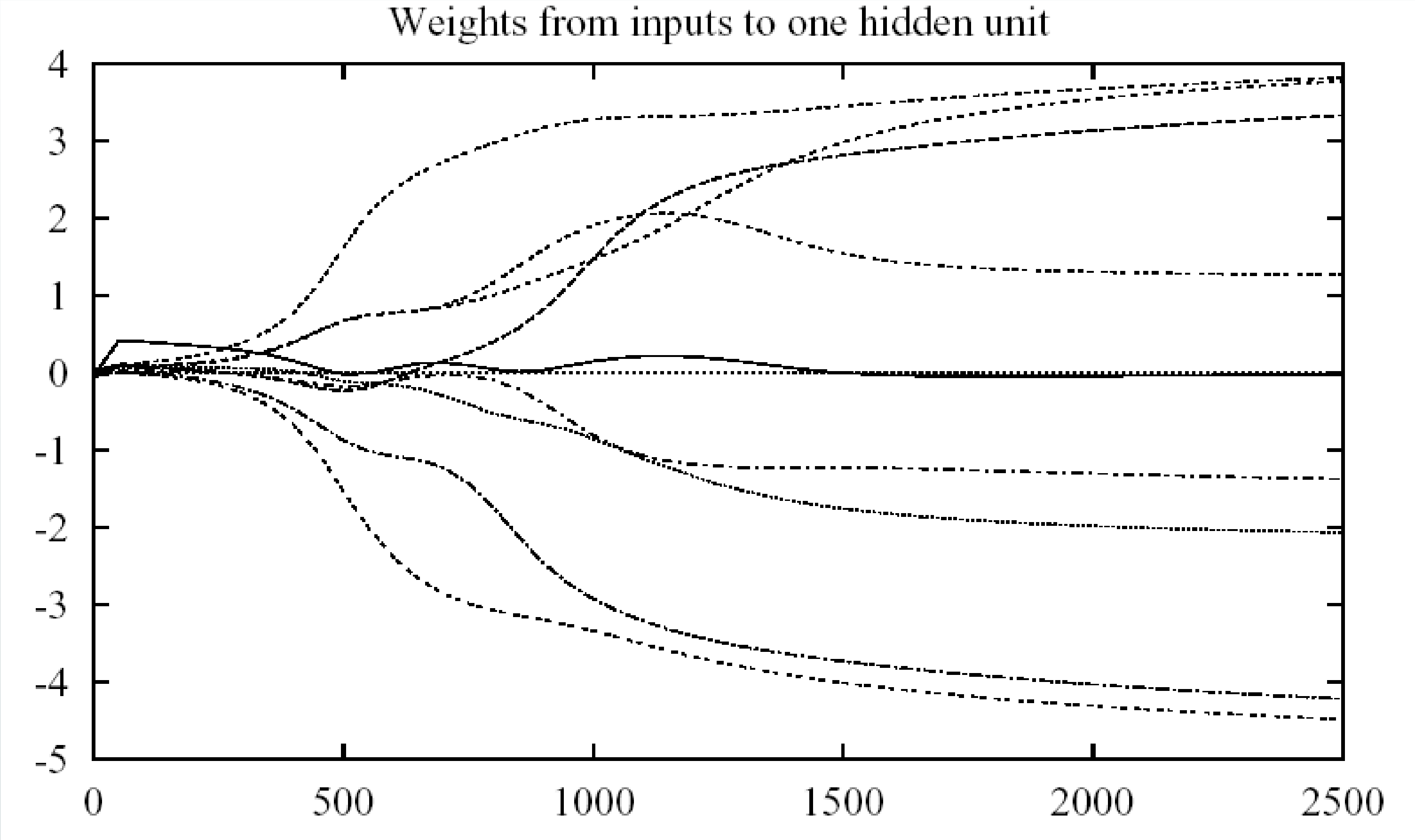

Sum of squared errors for each output unit
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
500
1000
1500
2000
2500

Training (2/3)



Hidden unit encoding for input 01000000
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
500
1000
1500
2000
2500

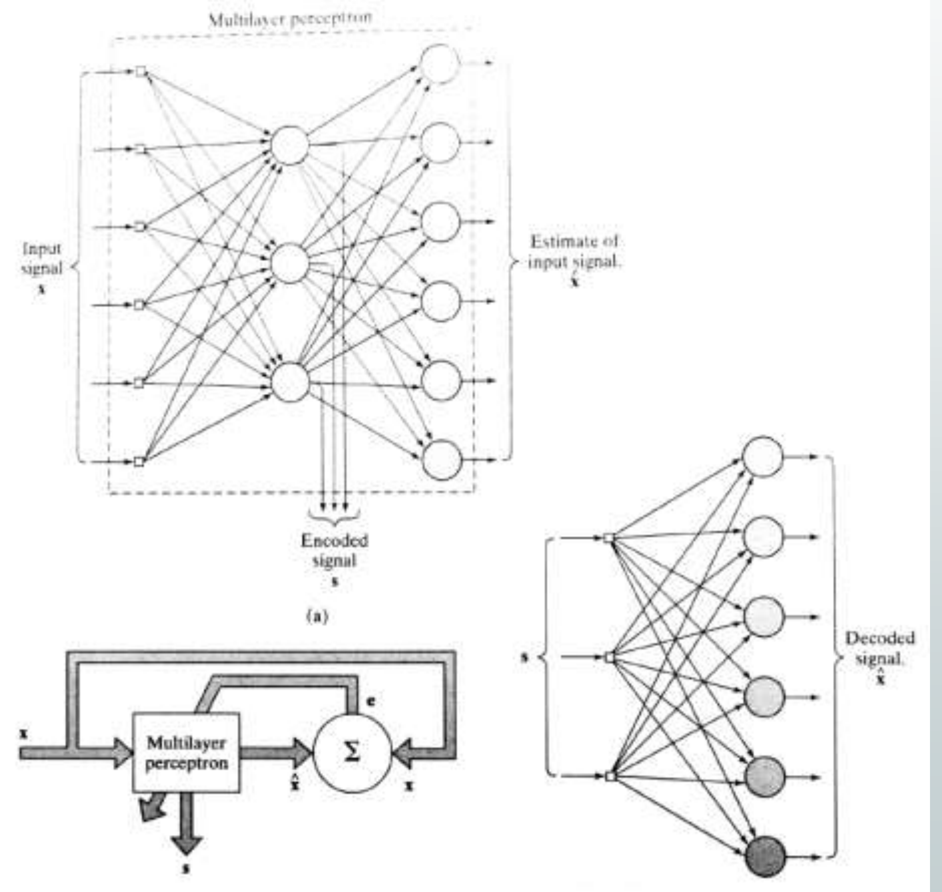
Training (3/3)



Weights from inputs to one hidden unit
4
3
2
1
0
-1
-2
-3
-4
-5
0
500
1000
1500
2000
2500

Example: Data Compression

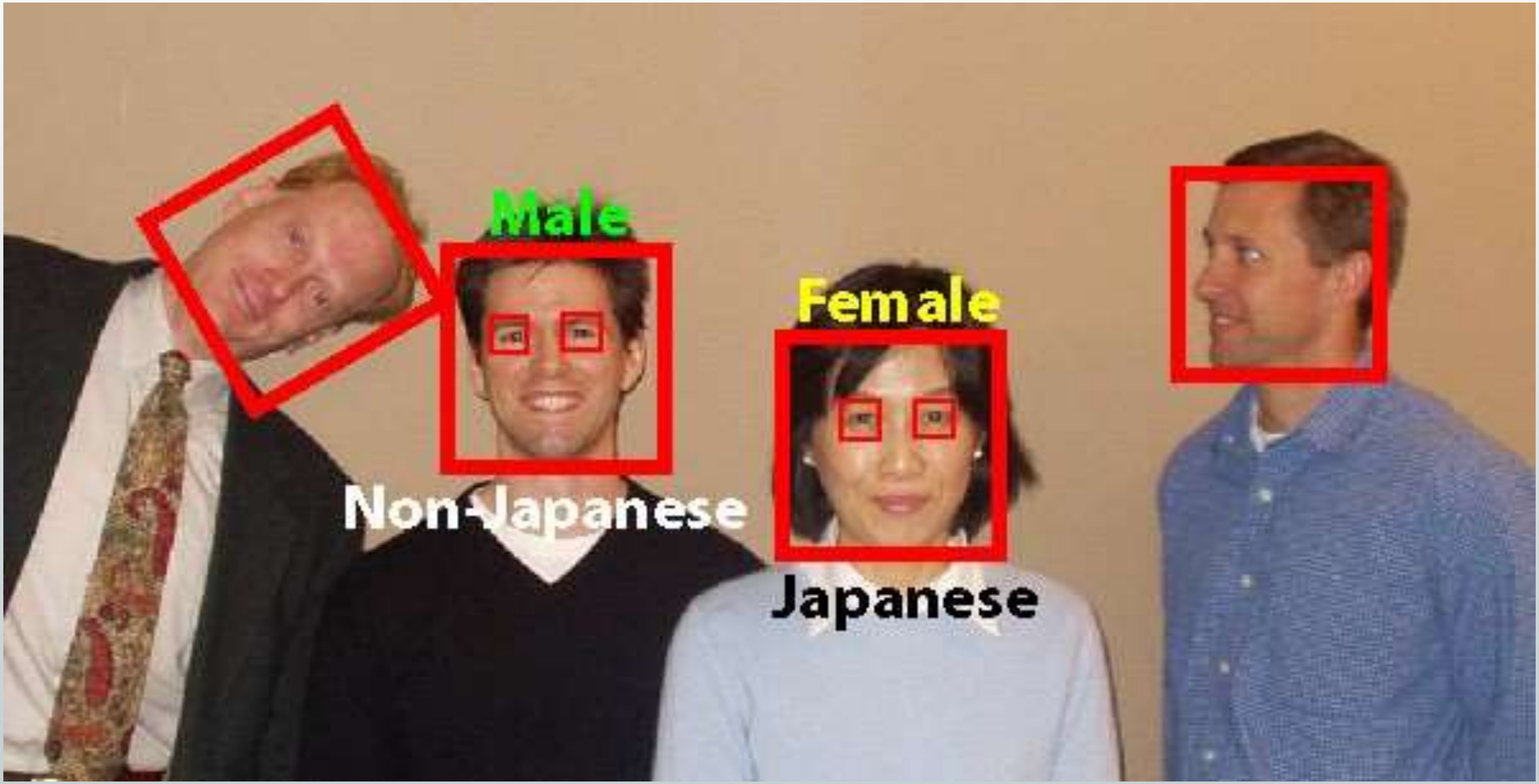
- Construct an auto-associative memory where Input = Output
- Train with small hidden layer
- Encode using input-to-hidden weights
- Send or store hidden layer activation.
- Decode received or stored hidden layer activation with the hidden-to-output weights.



Neural Net for object recognition from images

- Objective
  - Identify interesting objects from input images
    - Face recognition
      - Locate faces, happy/sad faces, gender, face pose, orientation
      - Recognize specific faces: authorization
    - Vehicle recognition (traffic control or safe driving assistant)
      - Passenger car, van, pick up, bus, truck
    - Traffic sign detection
- Challenges
  - Image size (100x100, 1024x1024)
  - Object size, pose and object orientation
  - Illuminations

Example



Example: Face Detection Challenges

pose variation

lighting condition variation

facial expression variation

Normal procedures

- Training (identify your problem and build specific model)
  - Build training dataset
    - Isolate sample images
      - Images containing faces
    - Extract regions containing the objects
      - region containing faces
    - Normalization (size and illumination)
      - 200x200 etc.
    - Select counter-class examples
      - Non-face regions
  - Determine Neural Net
    - Input layers are determined by the input images
      - E.g., a 200x200 image requires 40,000 input dimensions, each containing a value between 0-255
    - Neural net architectures
      - A three layer FF NN (two hidden layers) is a common practice
    - Output layers are determined by the learning problem
      - Bi-class classification or multi-class classification
  - Train Neural Net

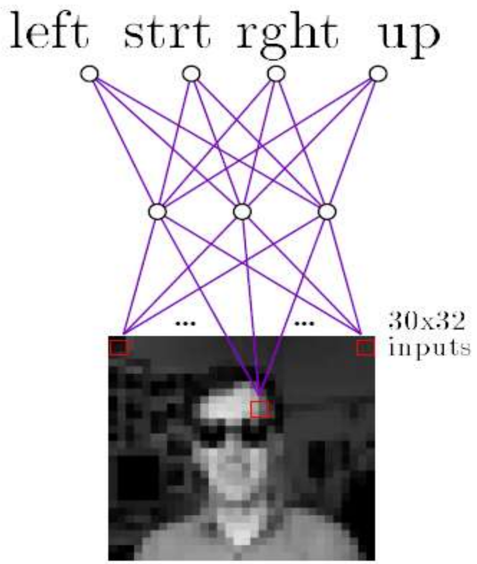
Normal procedures

- Test
  - Given a test image
    - Select a small region (considering all possibilities of the object location and size)
      - Scanning from the top left to the bottom right
      - Sampling at different scale levels
    - Feed the region into the network, determine whether this region contains the object or not
    - Repeat the above process
      - Which is a time consuming process

Neural Nets for Face Pose Recognition

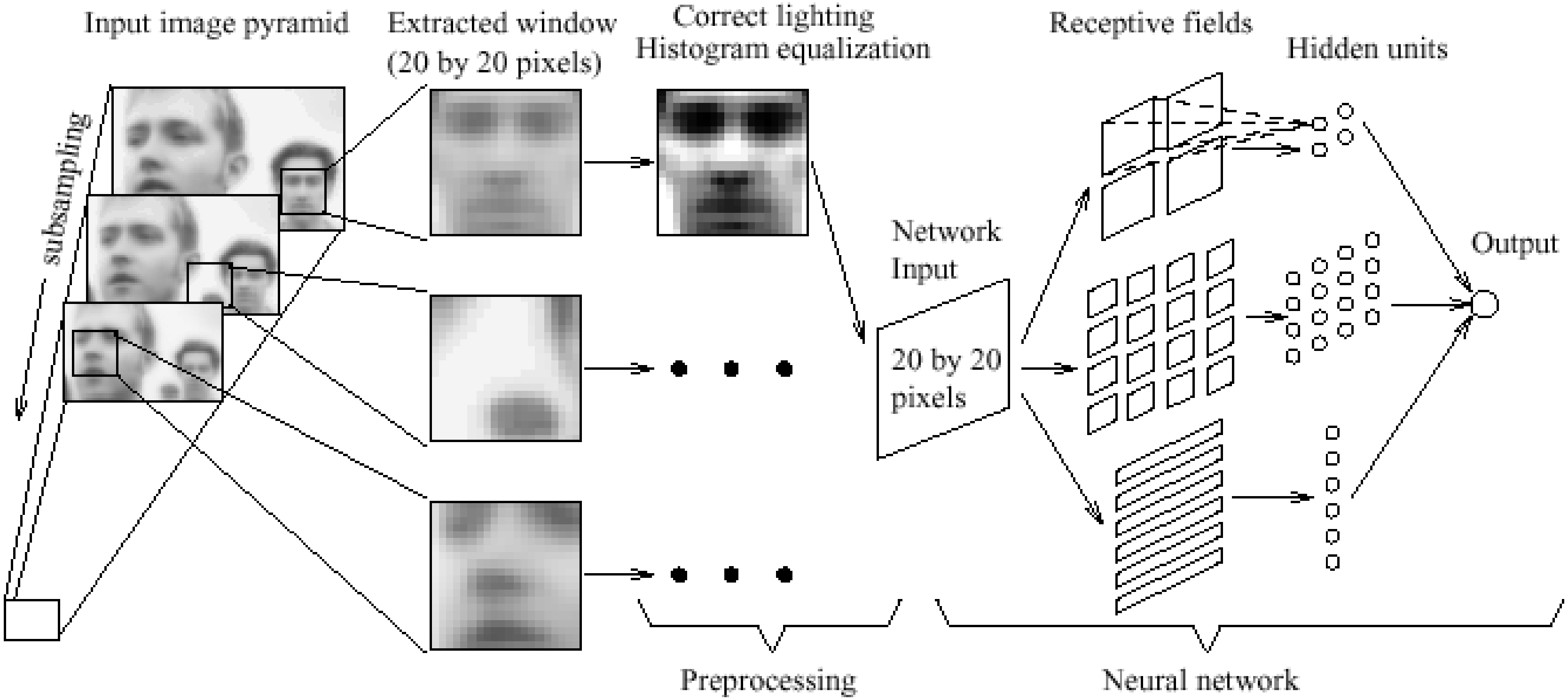
Head pose (1-of-4): 90% accuracy

Face recognition (1-of-20): 90% accuracy



Typical input images

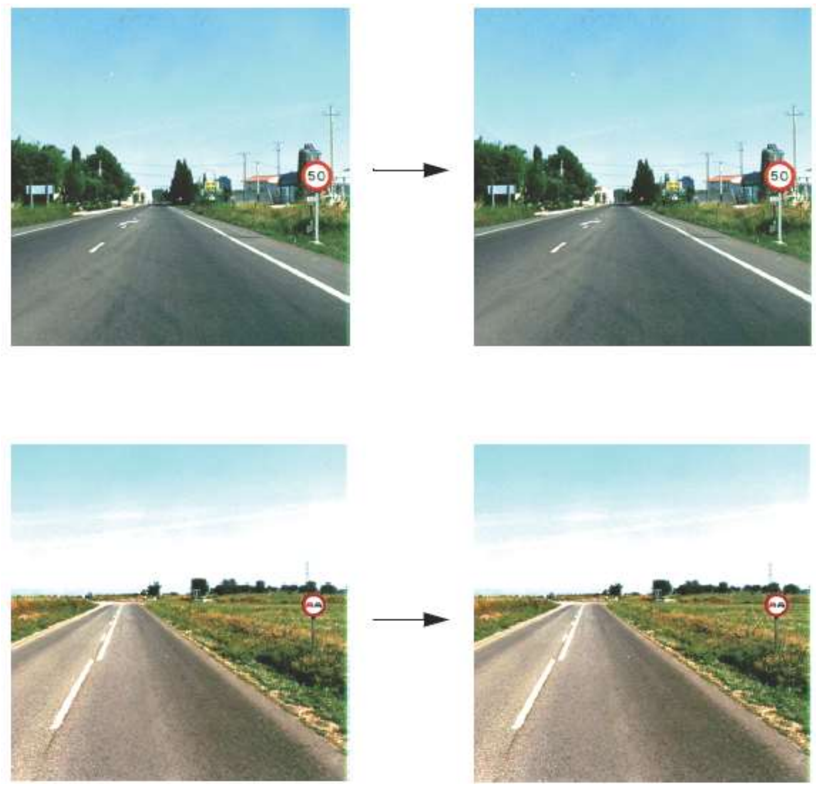
Neural Net Based Face Detection



- Large training set of faces and small set of non-faces
- Training set of non-faces automatically built up:
    - Set of images with no faces
    - Every ‘face’ detected is added to the face training set.

Traffic sign detection

- Demo
- http://www.mathworks.com/products/demos/videoimage/traffic_sign/vipwarningsigns.html
- Intelligent traffic light control system
  - Instead of using loop detectors (like metal detectors)
  - Using surveillance video: Detecting vehicle and bicycles



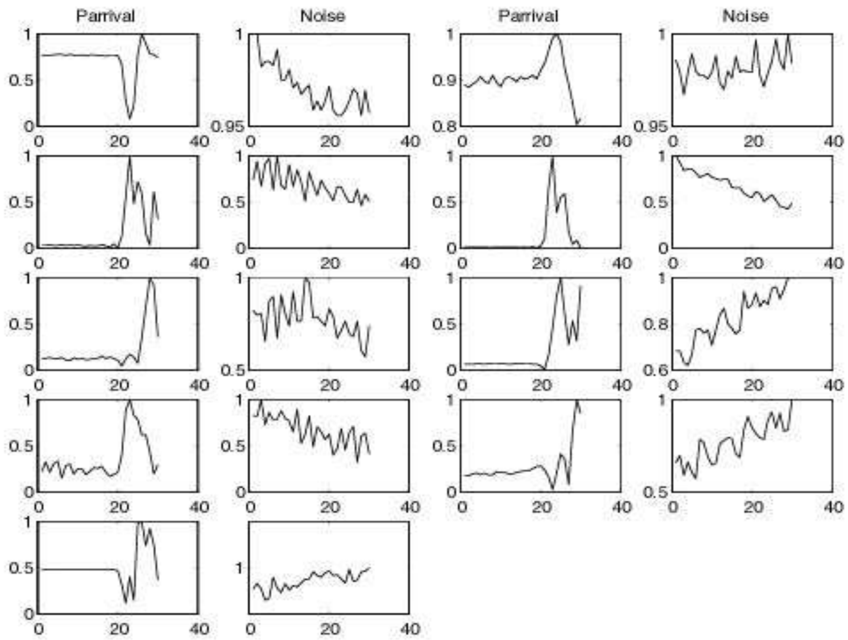
Vehicle Detection

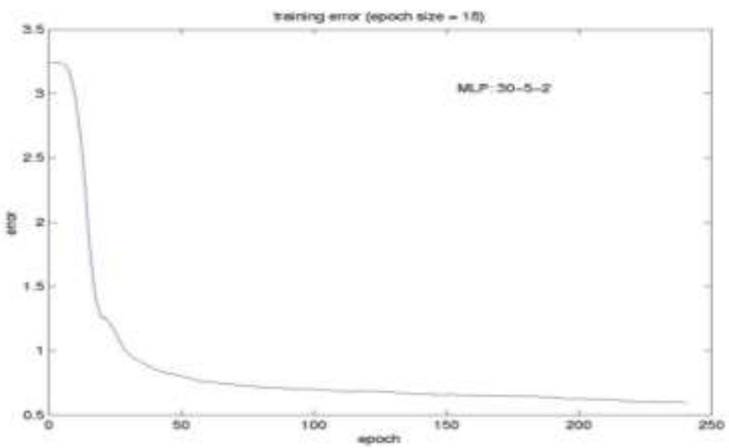
- Intelligent vehicles aim at improving the driving safety by machine vision techniques

http://www.mobileye.com/visionRange.shtml

Identification of the P-wave arrival

- Configuration
  - -30 inputs: 20^{th} sample corresponding to P-wave arrival
  - -2 outputs: corresponding to the noise and P-wave arrival
  - -1 hidden layer: 5 nodes
  - -Learning rate: 0.1, Momentum: 0.8
- Results
  - -Training set: including 18 P-wave arrival and noise segments
  - -Classification rate: 94.5%
  - -Testing set: including 58 P-wave arrival and noise segments
  - -Classification rate: 82%





Resources: Datasets

- UCI Repository: http://www.ics.uci.edu/~mlearn/MLRepository.html
- UCI KDD Archive: http://kdd.ics.uci.edu/summary.data.application.html
- Statlib: http://lib.stat.cmu.edu/
- Delve: http://www.cs.utoronto.ca/~delve/

Resources: Journals

- Journal of Machine Learning Research
- Machine Learning
- Neural Computation
- Neural Networks
- IEEE Transactions on Neural Networks
- IEEE Transactions on Pattern Analysis and Machine Intelligence
- Annals of Statistics
- Journal of the American Statistical Association
- IEEE Trans. On Knowledge and Data Engineering
- Data Mining and Knowledge Discovery
- ...

Resources: Conferences

- International Joint Conference on Artificial Intelligence (IJCAI)
- International Conference on Machine Learning (ICML)
- Neural Information Processing Systems (NIPS)
- American Association for Artificial Intelligence (AAAI)
- Uncertainty in Artificial Intelligence (UAI)
- International Conference on Neural Networks (Europe)
- ACM Knowledge Discovery and Data Mining (KDD)
- IEEE International Conference on Data Mining (ICDM)
- ...