Learning Outcomes

- Pattern classification
- Artificial Neural Networks
- Perceptron learning
- MLP
- AutoMLP
- The Mnist database
- Applications of ANN in Document Analysis
Pattern Classification

• More than a program
  • Usually, we think a program is something written by an experienced person.
Pattern Classification

• More than a program
  • Usually, we think a program is something written by an experienced person.
  • Often, the program isn’t complete without “experience” of its own.
Pattern Classification

• More than a program
  • Usually, we think a program is something written by an experienced person.
  • Often, the program isn’t complete without “experience” of its own.
  • The idea of writing programs that use data (experience) to create better programs than people can write directly.
Pattern Classification

- Pattern classification systems make decisions
- Decisions are usually made autonomously
- Decisions are not pre-programmed
- Decision “rules” are derived from data
SE, AI, PR

- Software Engineering
  - Manual creation of specifications, manual implementation, full control over details of execution.

- Artificial Intelligence (Rule-Based Systems)
  - Manual creation of specifications, specifications are directly executable (rule interpreters, etc.). Details of execution are automated.

- Pattern Recognition, Machine Learning
  - Programmer chooses category of application, but detailed specifications are automatically derived from data. Execution is automated.
Fish Classification: An example

- “Sorting incoming Fish on a conveyor according to species using optical sensing”
- Species
  - Sea bass
  - Salmon
Problem Analysis

- set up a camera and take some sample images to extract features
- feature types
  - length (positive real number)
  - lightness (positive real number)
  - width (positive real number)
  - number of fins (non-negative integer)
  - shape of fins (one-of a set of possible categories)
  - position of the mouth (one of a set of possible categories)
  - ...
- base the decision of which kind of fish it is on these measurements
Preprocessing

- raw camera image may be 1024 x 1024 pixels
  - >1 million numbers
- contains lots of irrelevant data
  - background
  - dirt
  - ...
- feature extraction
  - data reduction—computational efficiency
  - remove irrelevant variation
- feature measurements are passed to the classifier
Overall classifier
Feature Vector

- collection of measurements like an “object” or “structure” or “database record”
- example
  
  `{ length = 21cm, 
  lightness = 0.73, 
  width = 8.3cm, 
  number_of_fins = 3, 
  shape_of_fins = {square, triangular, square}, 
  position_of_the_mouth = {front} }`
Dataset

like a spreadsheet table (with millions of rows...)
- training data: class + features
- test data: features only

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>class</td>
<td>length</td>
<td>lightness</td>
<td>width</td>
<td>#fins</td>
<td>shapes</td>
<td>mouth</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>28.59</td>
<td>0.35</td>
<td>15.99</td>
<td>2</td>
<td>s,s</td>
<td>f</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>22.18</td>
<td>0.37</td>
<td>13.31</td>
<td>1</td>
<td>t</td>
<td>b</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>16.29</td>
<td>0.28</td>
<td>16.89</td>
<td>2</td>
<td>t,t</td>
<td>f</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>29.87</td>
<td>0.46</td>
<td>14.49</td>
<td>1</td>
<td>t</td>
<td>b</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>27.65</td>
<td>0.43</td>
<td>13.11</td>
<td>2</td>
<td>s,s</td>
<td>f</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>29.75</td>
<td>0.57</td>
<td>13.70</td>
<td>2</td>
<td>s,s</td>
<td>f</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>26.43</td>
<td>0.30</td>
<td>13.54</td>
<td>1</td>
<td>s</td>
<td>f</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>24.13</td>
<td>0.25</td>
<td>12.50</td>
<td>1</td>
<td>t</td>
<td>f</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>27.26</td>
<td>0.47</td>
<td>14.05</td>
<td>1</td>
<td>t</td>
<td>f</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>24.13</td>
<td>0.32</td>
<td>12.23</td>
<td>1</td>
<td>t</td>
<td>f</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>22.48</td>
<td>0.58</td>
<td>14.14</td>
<td>1</td>
<td>t</td>
<td>f</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>21.14</td>
<td>0.48</td>
<td>16.16</td>
<td>1</td>
<td>t</td>
<td>f</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>16.11</td>
<td>0.44</td>
<td>15.75</td>
<td>2</td>
<td>t,t</td>
<td>f</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>21.26</td>
<td>0.34</td>
<td>12.71</td>
<td>2</td>
<td>t,t</td>
<td>f</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>28.54</td>
<td>0.51</td>
<td>17.00</td>
<td>1</td>
<td>s</td>
<td>f</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>18.99</td>
<td>0.25</td>
<td>12.31</td>
<td>1</td>
<td>s</td>
<td>f</td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>26.98</td>
<td>0.39</td>
<td>16.60</td>
<td>2</td>
<td>t,t</td>
<td>f</td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>20.50</td>
<td>0.29</td>
<td>16.56</td>
<td>2</td>
<td>t,s</td>
<td>f</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>22.67</td>
<td>0.45</td>
<td>15.61</td>
<td>2</td>
<td>t,t</td>
<td>f</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>17.20</td>
<td>0.37</td>
<td>16.03</td>
<td>2</td>
<td>t,s</td>
<td>b</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>21.38</td>
<td>0.26</td>
<td>12.08</td>
<td>2</td>
<td>t,t</td>
<td>f</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>20.73</td>
<td>0.41</td>
<td>14.93</td>
<td>2</td>
<td>t,s</td>
<td>f</td>
</tr>
</tbody>
</table>
Histogram

frequency of each feature
- “binned” for real-valued features

let us guess a classifier by eye
A Decision Rule is a function mapping feature vectors into classes

Decision(features) =
  IF features.length<11 THEN return “salmon”
  ELSE return “sea bass”
Empirical Error Rate

fraction of misclassified samples among total samples
“empirical” because it is estimated from a data sample
Finding Good Features and Decision Boundaries

using lightness, we can separate the two classes better than using length
moving the decision boundary changes the error rate
Multiple Features

Scatterplot

Decision Boundary

salmon

sea bass
Linear Decision Function

\[
\text{Decision(features)} = \\
\quad \text{IF } \text{features.lightness} \times 0.6 + \text{features.width} \times 0.11 - 21.3 < 0 \\
\quad \text{THEN return \textquote{salmon}} \\
\quad \text{ELSE return \textquote{sea bass}}
\]

decision function depends on two numerical features

feature values enter linearly into the decision function
Generalization

decision boundaries that work perfectly on a few samples...
Overtraining

... may fail to generalize well to new data (overtraining) ensuring good generalization is a key problem in pattern recognition
Avoid overtraining

choose a tradeoff between...
- performance on the training data
- “complexity” of the classifier / decision boundary

decision boundary is too complex

probably a good tradeoff
Example Classifiers

- Decision trees
- Nearest neighbors
- Support Vector Machines
- Neural networks
- Naive Bayesian classifiers
- etc.
Artificial Neural Networks

• Mainly inspired by human brain

• A Human brain is
  • Capable of computationally demanding perceptual acts – like, recognition of faces, speech
  • Highly parallel computing structure
  • Imprecise information processing
  • Collection of more than 10 billion interconnected neuron
How can brain learn?

- a few pounds of gray and white stuff
- composed of neurons
- how does it work?
Biological Neuron

- Biochemical reactions to receive, process and transmit information
- Dendrite
  - Tree like network of nerve fibers
- Axon
  - Single long fiber extending from cell body
- Synapses
  - Transmission of signals from one neuron to another at synapses
  - Complex chemical process
    - Specific transmitter substance is released from sending end of junction
    - Raise or lower the electrical potential inside the body of receiving cell
    - Based on a threshold a pulse is sent down the axon and the cell is 'fired'
Biological Neuron

[Diagram of a biological neuron showing various cellular components such as dendrites, microtubules, neurofibrils, neurotransmitter, receptor, synaptic vesicles, synapse (Axodendritic), synapse (Axosomal), synaptic cleft, axonal terminal, node of Ranvier, myelin sheath (Schwann cell), nucleus, nucleolus, membrane, mitochondrion, smooth ER, and microfilament, microtubule, axon.]
The Perceptron

- Binary classifier functions
- Threshold activation function
Neuron on $u_i$

Input:
- $i_{i1}$
- $i_{i2}$
- $i_{i3}$
- $\ldots$
- $i_{in}$

Weights:
- $w_{i1}$
- $w_{i2}$
- $w_{i3}$
- $\ldots$
- $w_{in}$

Network function:
\[\text{net}(i_{i1}, \ldots, i_{in}, w_{i1}, \ldots, w_{in})\]

Activation function:
\[f_{akt}(\text{net}) = o_i\]

Output:
\[o_i \in O\]
Activation Function

- Linear function
- Step function
- Sigmoid function
Non Linear Activation Functions

\[ f(x) = \frac{1}{1 + e^{-x}} \]

\[ \frac{\partial f}{\partial x} = f(x)(1 - f(x)) \]

\[ f(x) = \tanh(x) \]

\[ \frac{\partial f}{\partial x} = (1 + f(x))(1 - f(x)) \]
History

• McCulloch and Pitts introduced the Perceptron in 1943.
  • Simplified model of a biological neuron
  • rate coding of activation
  • abstract as linear threshold machine
    ◦ firing rate = activation
    ◦ sum up
    ◦ if total above trigger, then start firing
History

- Fell out of favor in the late 1960's
  - (Minsky and Papert)
  - Perceptron limitations
- Resurgence in the mid 1980's
  - Nonlinear Neuron Functions
  - Back-propagation training
A simple classifier

• In its simplest form, a learner’s job is to produce a classifier.

• A classifier takes objects as input and assigns each one to a class.

• Most simply, objects are represented as vectors of features and classes are 0 or 1 i.e.

• A two class classifier
Basic Elements

• Each component of the feature vector becomes an input unit.

• An additional input unit is added that always has an input of 1.

• Each input unit is connected to a sum unit with a weight. The input value is multiplied by the weight and all are summed.

• If the sum is greater than zero, the output is “1”, else “0”.
Training A Perceptron

• Given a training set, would like a way to set the weights to correctly classify the labeled instances.

• Networks with one layer of nets are very well understood at this point.

• the perceptron training procedure
  http://en.wikipedia.org/wiki/Perceptron


The Math

- \( i \) is a feature.
- \( x \) is the input (\( x_i \) is the \( i \)th component).
- \( w_i \) is weight vector.
- \( y = (\sum_i x_i \times w_i) > 0 \).
- \( t \) is the target (right output for \( x \)).
- \( \alpha \) is the learning rate (amount to change the weights).
- \( \Delta w_i \) is the amount we plan to change weight \( i \).

For all features \( i \):
- \( \Delta w_i = \alpha(t - y)x_i \)
The Rule

<table>
<thead>
<tr>
<th>$t$</th>
<th>$y$</th>
<th>$\Delta w_i = \alpha(t - y)x_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$t - y$ is zero (right answer). Nothing needs to change.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$t - y$ is $-1$ (output too big). If $x_i &gt; 0$, $w_i$ is decreased to make sum smaller.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>$t - y$ is $1$ (output too small). If $x_i &gt; 0$, $w_i$ is increased to make sum bigger.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$t - y$ is zero (right answer). Nothing needs to change.</td>
</tr>
</tbody>
</table>
Using the Rule

- Start off with the weight vector set to an arbitrary value.
- For each example in the training set, apply the rule, changing the weights as needed.
  - If weights have changed, repeat.
- Stops changing when stops making mistakes.
- If there is a way of setting the weights that makes no mistakes, it will be found eventually.
- Otherwise, might bounce around.
- Very related to gradient descent and hill-climbing: local changes to reduce error score.
Linear Separability

- Perceptron can only achieve perfect score if data is linearly separable. That is, if we visualize the instances as points in a high dimensional space, there needs to be linear surface that separates the positive and negative examples.

- Minsky and Papert pointed out that data sets that are not linearly separable can cause the perceptron much headaches.
Problems with View

- Minsky:
  - a single layer of linear threshold machines is not powerful enough
- Bigger problems:
  - can’t even learn XOR or parity
  - learning algorithm is supervised; where does training data come from?
  - the brain has many layers, how do they all get trained?
  - how do we train multiple layers of threshold machines?
Multi Layer Perceptrons

- Modification of the standard linear perceptron
- uses three or more layers of neurons (nodes) with nonlinear activation functions
- more powerful than the perceptron in that it can distinguish data that is not linearly separable
- Backpropagation (Rumelhart86, Werbos74)
- can train multilayer networks and learn xor.
Design Process for Neural Networks

- Selection of network topology
- Selection of learning algorithm
- Representation of input and output values
- Selection of parameters
  - Nos of hidden nodes
  - Learning rate
  - Epochs
- Implementation
- „verification“ (test)
Problem of over-fitting
Some other problems

• problem
  • MLPs are hard to train
  • performance sensitive to chosen parameter values
  • optimal parameter values depend on dataset
Some other problems

- problem
  - MLPs are hard to train
  - performance sensitive to chosen parameter values
  - optimal parameter values depend on dataset
AutoMLP

- self-tuning MLP classifier (AutoMLP)
  - automatically adjust learning parameters
AutoMLP

- self-tuning MLP classifier (AutoMLP)
  - automatically adjust learning parameters

- key ideas
  - train a population of MLPs in parallel
  - sample parameter space according to some pdf
  - train each MLP for a few epochs
  - natural selection for next generation
    - select half of the classifiers with better performance
    - change architecture to generate new solutions

- internal validation
Character Recognition

- Is it '9'
- NN systems in place reading roughly half of all checks in US
The MNIST Database

- A training set of 60,000 examples
- A test set of 10,000 examples
- contain binary images of handwritten digits
MLP vs autoMLP

- MNIST dataset
- MLP with sparse manual search
  - nhidden = 20, 80
  - eta = 0.01, 0.1
- AutoMLP with default settings
Applications of Neural Networks in Document Analysis

TABLE 1
Neural Approaches to the Preprocessing of Document Images

<table>
<thead>
<tr>
<th>Task</th>
<th>Neural architecture</th>
<th>Input to neural network</th>
<th>Output of neural network</th>
<th>Refs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binarization</td>
<td>MLP</td>
<td>Features in 5x5 moving window</td>
<td>Foreground/Background pixel</td>
<td>[10]</td>
</tr>
<tr>
<td>Image restoration</td>
<td>MLP</td>
<td>Pixels in a moving window</td>
<td>Restored value of current pixel</td>
<td>[12]</td>
</tr>
<tr>
<td>Removal of touching lines</td>
<td>MLP with receptive fields</td>
<td>Histogram of chord lengths</td>
<td>Membership of current pixel</td>
<td>[13]</td>
</tr>
<tr>
<td>Page de-skew</td>
<td>MLP</td>
<td>Features computed from the page</td>
<td>Skew angle</td>
<td>[14]</td>
</tr>
<tr>
<td>Symbol de-skew</td>
<td>MLP</td>
<td>Moments computed from the symbol</td>
<td>Skew angle</td>
<td>[15]</td>
</tr>
<tr>
<td>Symbol thinning</td>
<td>SOM</td>
<td>Normalized bitmap of the symbol</td>
<td>Thinned symbol</td>
<td>[16]</td>
</tr>
</tbody>
</table>

Artificial Neural Networks for Document Analysis and Recognition
Simone Marinai, Marco Gori, Fellow, IEEE, and Giovanni Soda, Member, IEEE Computer Society
IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 27, NO. 1, JANUARY 2005
Applications of Neural Networks in Document Analysis

### Table 2
Neural Approaches to Layout Analysis

<table>
<thead>
<tr>
<th>Task</th>
<th>Classification level</th>
<th>Neural architecture</th>
<th>Input to neural network</th>
<th>Output of neural network</th>
<th>Refs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page segmentation</td>
<td>Pixel</td>
<td>MLP</td>
<td>Pixels in a window around current pixel</td>
<td>Background, halftone, text &amp; line drawings</td>
<td>[24]</td>
</tr>
<tr>
<td>Text location in color images</td>
<td>Pixel</td>
<td>Parallel combination of 3 MLPs: one for each color band</td>
<td>Pixels in a window around current pixel</td>
<td>Text, non-text</td>
<td>[30]</td>
</tr>
<tr>
<td>Multi-resolution segmentation</td>
<td>Pixel</td>
<td>MLP</td>
<td>Moments of wavelets computed in a window around current pixel</td>
<td>Image, text, graphics</td>
<td>[28]</td>
</tr>
<tr>
<td>Identification of text blocks - Global features</td>
<td>Region</td>
<td>MLP</td>
<td>Region size, number of black pixels, properties of connected components</td>
<td>Text, non-text</td>
<td>[26]</td>
</tr>
<tr>
<td>Identification of text blocks - Local features</td>
<td>Region</td>
<td>SOM</td>
<td>Occurrences of predefined 3x3 masks in the region</td>
<td>Text, non-text</td>
<td>[23]</td>
</tr>
<tr>
<td>Region labeling - Voting of labels of few random windows</td>
<td>Region</td>
<td>MLP</td>
<td>Gradient vector and luminance computed in a window</td>
<td>Background, printed &amp; handwritten characters, photo, image</td>
<td>[27]</td>
</tr>
<tr>
<td>Color clustering for text extraction</td>
<td>Region</td>
<td>SOM for color clustering</td>
<td>Pixel color (R,G,B)</td>
<td>Pixel color cluster</td>
<td>[31]</td>
</tr>
<tr>
<td>Logical labeling of text blocks</td>
<td>Region</td>
<td>Recurrent neural network</td>
<td>Block size, number of lines, etc., and label of previous block</td>
<td>Date, Address, etc.</td>
<td>[32]</td>
</tr>
<tr>
<td>Logical labeling of text lines</td>
<td>Region</td>
<td>Recurrent neural network</td>
<td>Features of connected components</td>
<td>Postal code or not</td>
<td>[33]</td>
</tr>
<tr>
<td>Form classification</td>
<td>Page</td>
<td>MLP</td>
<td>Line crossings</td>
<td>Form Identifier</td>
<td>[34]</td>
</tr>
<tr>
<td>Page classification in digital libraries</td>
<td>Page</td>
<td>MLP</td>
<td>MXY tree encoding</td>
<td>Page class (title, index, regular, etc.)</td>
<td>[35]</td>
</tr>
</tbody>
</table>