Neural Networks

Sven Koenig, USC

Russell and Norvig, 3rd Edition, Sections 18.7.1-18.7.4

These slides are new and can contain mistakes and typos.
Please report them to Sven (skoenig@usc.edu).

Inductive Learning for Classification

- Labeled examples

Feature_1	Feature_2	Class
true	true	true
true	false	false
false	true	false

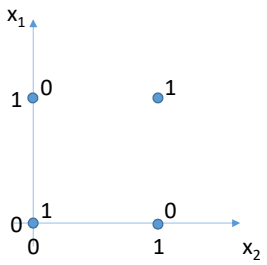
Learn f(Feature_1, Feature_2) = Class from
f(true, true) = true
f(true, false) = false
f(false, true) = false
The function needs to be consistent with all labeled examples
and should make the fewest mistakes on the unlabeled examples.

- Unlabeled examples

Feature_1	Feature_2	Class
false	false	?

Example: Neural Network Learning

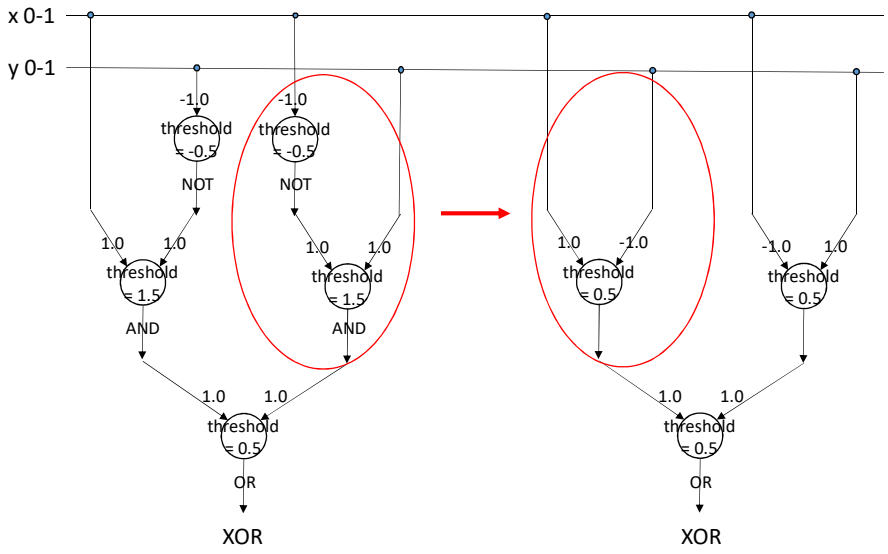
- Can perceptrons represent all Boolean functions? – no!
  f(Feature_1, …, Feature_n) ≡ some propositional sentence
- An XOR cannot be represented with a single perceptron!



Example: Neural Network Learning

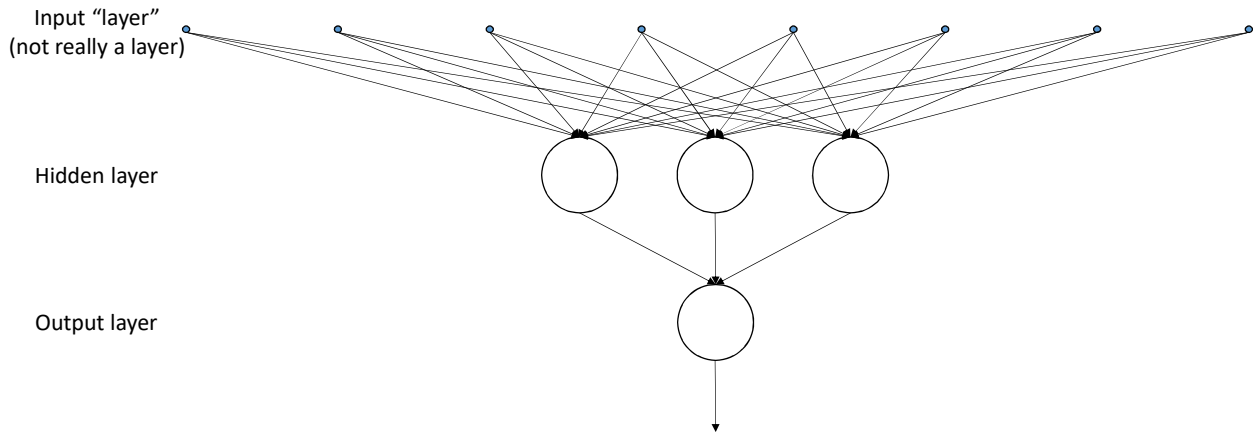
- Can perceptrons represent all Boolean functions? – no!
  f(Feature_1, …, Feature_n) ≡ some propositional sentence
- An XOR cannot be represented with a single perceptron!
- However,
  - XOR(x,y) ≡ (x AND NOT y) OR (NOT x AND y).
  - AND, OR and NOT can be represented with single perceptrons.
- Thus, XOR can be represented with a network of perceptrons (also called a neural network).
- Neural networks can represent all Boolean functions!

Example: Neural Network Learning



Example: Neural Network Learning

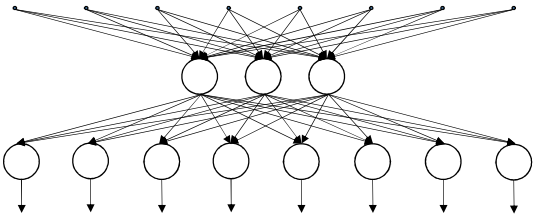
- We will use “three-layer” feed-forward networks as network topology.



Example: Neural Network Learning

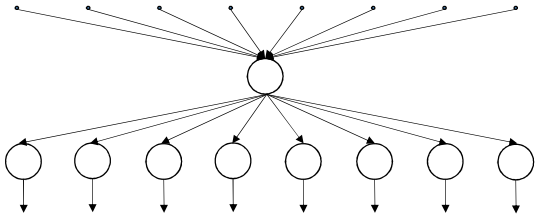
- Neural networks can automatically discover useful representations.
- If there are too few perceptrons in the hidden layer, the neural network might not be able to learn a function that is consistent with the labeled examples.
- If there are too many perceptrons in the hidden layer, then the neural network might not be able to generalize well, that is, make few mistakes on the unlabeled examples.

Example: Neural Network Learning

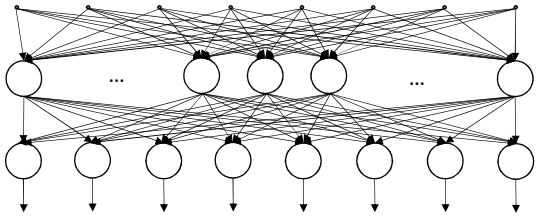


input	hidden values			output
10000000	0.89	0.04	0.08	10000000
01000000	0.15	0.99	0.99	01000000
00100000	0.01	0.97	0.27	00100000
00010000	0.99	0.97	0.71	00010000
00001000	0.03	0.05	0.02	00001000
00000100	0.01	0.11	0.88	00000100
00000010	0.80	0.01	0.98	00000010
00000001	0.60	0.94	0.01	00000001

Example: Neural Network Learning

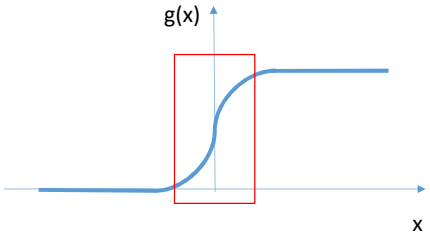


Example: Neural Network Learning



Example: Neural Network Learning

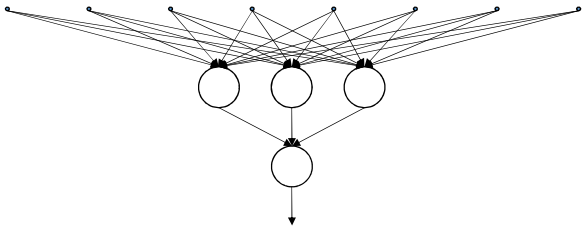
- One can use non-binary inputs. However, avoid operating in the (red) region where small changes in the input cause large changes in the output. Rather, use several outputs by using several perceptrons in the output layer.



Example: Neural Network Learning

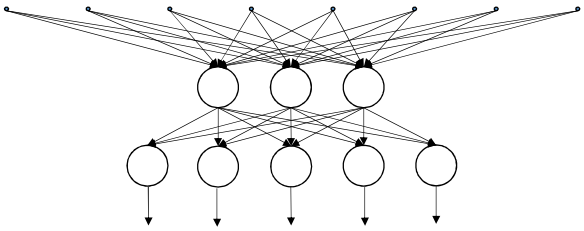
- Example with real-values inputs and outputs: early autonomous driving

brightness of pixels in the image (each input = one pixel)



steering direction [0 = sharp left .. 1 = sharp right]

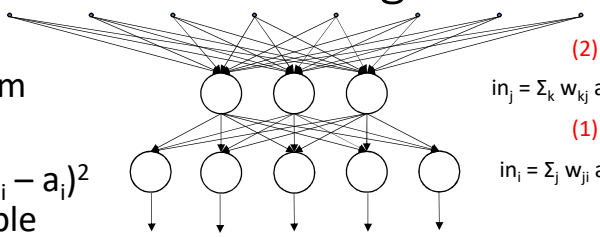
brightness of pixels in the image (each input = one pixel)



sharp left left center right sharp right

to determine a unique steering direction, fit a Gaussian to the outputs

Example: Neural Network Learning



- Backpropagation algorithm (see handout for details)
- Minimize Error := $0.5\ \Sigma_i (y_i - a_i)^2$ for a single labeled example with the approximation of gradient descent (for a small positive learning rate α), where y_i is the desired *i*th output for the labeled example
- (1) d Error / d w_{ji} := - $\Delta[i]\ a_j$, where $\Delta[i] := (y_i - a_i)\ g'(in_i)$ ← basically the same derivation as for a single perceptron
- (2) d Error / d w_{kj} := - $\Delta[j]\ a_k$, where $\Delta[j] := \Sigma_i\ \Delta[i]\ w_{ji}\ g'(in_j)$
- The errors are "propagated back" from the outputs to the inputs, hence the name "backpropagation"

Example: Neural Network Learning

- Backpropagation algorithm

```
function BACK-PROP-LEARNING(examples, network) returns a neural network
  inputs: examples, a set of examples, each with input vector x and output vector y
          network, a multilayer network with L layers, weights w_{i,j}, activation function g
  local variables: Δ, a vector of errors, indexed by network node

  for each weight w_{i,j} in network do
      w_{i,j} ← a small random number
  repeat
      for each example (x, y) in examples do
          /* Propagate the inputs forward to compute the outputs */
          for each node i in the input layer do
              a_i ← x_i
          for ℓ = 2 to L do
              for each node j in layer ℓ do
                  in_j ← Σ_i w_{i,j} a_i
                  a_j ← g(in_j)
          /* Propagate deltas backward from output layer to input layer */
          for each node j in the output layer do
              Δ[j] ← g'(in_j) × (y_j − a_j)
          for ℓ = L − 1 to 1 do
              for each node i in layer ℓ do
                  Δ[i] ← g'(in_i) Σ_j w_{i,j} Δ[j]
          /* Update every weight in network using deltas */
          for each weight w_{i,j} in network do
              w_{i,j} ← w_{i,j} + α × a_i × Δ[j]
  until some stopping criterion is satisfied
  return network
```

Note: This pseudo code from Russell and Norvig 3rd edition is wrong in the textbook, so be careful here!

called one epoch

Example: Neural Network Learning

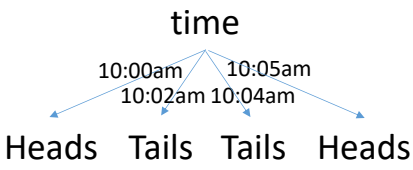
- Overfitting (= adapting to sampling noise)

time	coin flip
10:00am	Heads
10:02am	Tails
10:04am	Tails
10:05am	Heads

We want:

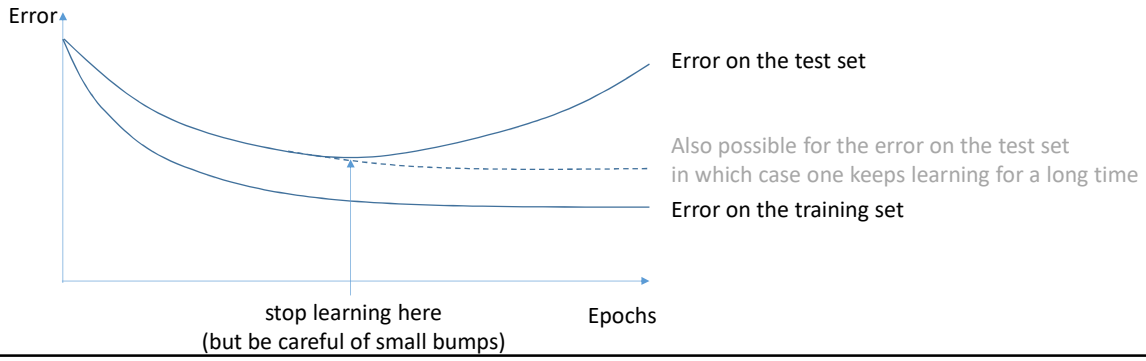
½: Heads; ½: Tails

We get the decision tree:



Example: Neural Network Learning

- Cross validation by splitting the labeled examples into a training set (often 2/3 of the examples) and a test set (often 1/3 of the examples), using only the training set for learning and only the test set to decide when to stop learning



Example: Decision Tree (and Rule) Learning

- Overfitting can also occur for decision tree learning.
  - During decision tree learning, prevent recursive splitting on features that are not clearly relevant, even if the examples at a decision tree node then have different classes.
  - After decision tree learning, recursively undo splitting on features close to the leaf nodes of the decision tree that are not clearly relevant even if the examples at a decision tree node then have different classes (back pruning).

Example: Neural Network Learning

- Properties (some versus decision trees)
  - Deal easily with real-valued feature values
  - Are very tolerant of noise in feature and class values of examples
  - Make classifications that are difficult to explain (even to experts!)
  - Need a long learning time
- "Neural networks are the 2nd best way of doing just about anything"
- Early applications
  - Pronunciation (cat vs. cent)
  - Handwritten character recognition
  - Face detection

Example: Neural Network Learning

- Want to play around with neural network learning?
- Go here: http://aispace.org/neural/
- Or here: http://playground.tensorflow.org/