Development of Games

Lecture 16

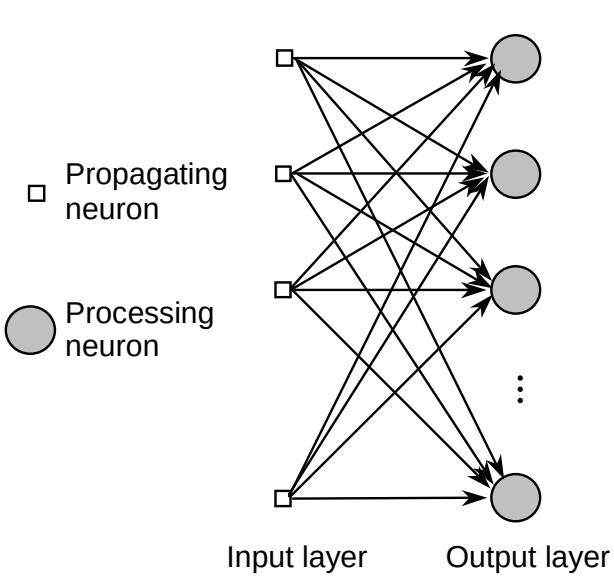
Neural Networks in Games

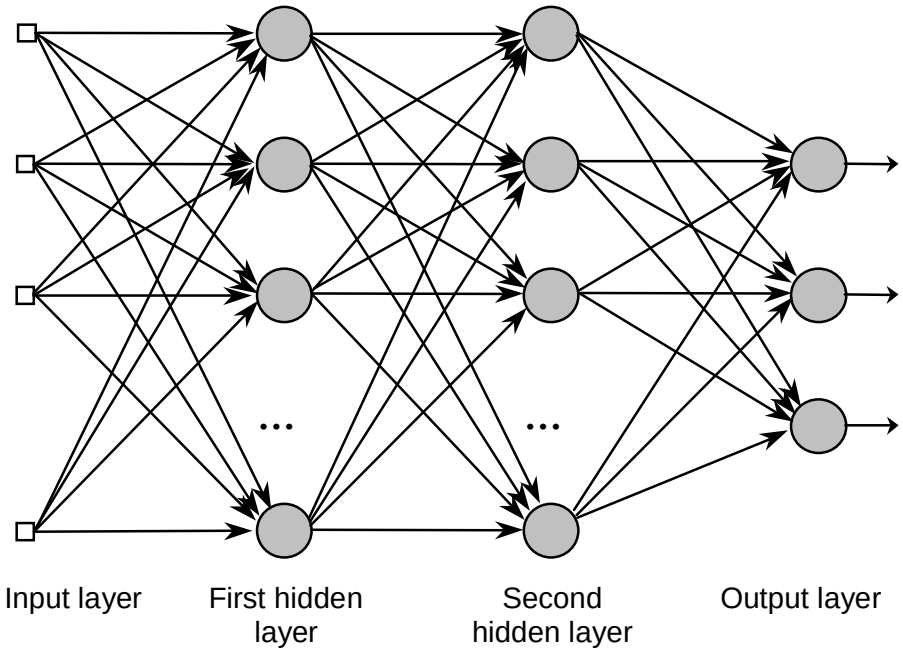
Main Characteristics of Artificial Neural Networks - recap

- **Massively parallel processing**
- **Each unit (“neuron”) is a simple processing element**
- **Network consists of many units, each of them connected to many other units**
- **Knowledge is distributed across numerical weights of the interconnections**
- **Learning usually consists of adjusting the interconnection weights in order to solve a given task**

Architecture of Artificial Neural Networks

- Example: Feedforward Networks





Single-layer feedforward Multi-layer feedforward

Note: each connection has a weight (not shown in the figure)

Main algorithm of training

```
repeat
      for each training pattern
            train on that pattern
      end for loop
until the error is 'acceptably low'
```

Kinds of sigmoid used in perceptrons

Exponential

$$f(s) = \frac{1}{1 + e^{-2\alpha s}}$$

Rational

$$f(s) = \frac{s}{|s| + \alpha}$$

Hyperbolic tangent

$$f(s) = th\frac{s}{\alpha} = \frac{e^{-\frac{s}{\alpha}} - e^{-\frac{s}{\alpha}}}{e^{\frac{s}{\alpha}} + e^{-\frac{s}{\alpha}}}$$

Formulas for error back propagation algorithm

Modification of weights of synapses of j^{th} neuron connected with i^{th} ones, x_j – state of j^{th} neuron (output)

$$w_{ij}(t+1) = w_{ij}(t) + r g_j x'_i \quad (1)$$

For output layer

$$g_j = y'_j(1 - y_j)(d_j - y_j) \quad (2)$$

For hidden layers
k – number of neuron in next layer connected with j^{th} neuron

$$g_j = x'_j(1 - x'_j)\sum_k g_k w_{jk} \quad (3)$$

The main step of training on a pattern may now be expanded into the following steps.

1. Present the pattern at the input layer
2. Let the hidden units evaluate their output using the pattern
3. Let the output units evaluate their output using the result in step 2) from the hidden units.

   The steps 1) - 3) are collectively known as the *forward pass* since information is flowing forward, in the natural sense, through the network.

4. Apply the target pattern to the output layer
5. Calculate the δ's on the output nodes according to (2), (1)
6. Train each output node using gradient descent (1)
7. For each hidden node, calculate its δ according to (3), (1)
8. For each hidden node, use the δ found in step 7) to train according to gradient descent (1)

   Steps 4) - 8) are collectively known as the *backward pass*

Step 7) involves *propagating* the δ's from those output nodes in the hidden unit's fan-out *back* towards this node so that it can process them. This is where the name of the algorithm comes from.

Classification of models of neural networks

- Tutoring
  - Supervised learning
  - Unsupervised learning
  - Reinforcement learning
- Structure
  - Forward or recurrent networks
  - With regular or not links
  - Describes by full-links graph or no
  - Static or dynamic (constructive learning)
- Signals (inputs or outputs, hidden)
  - Binary
  - Analog
- Time
  - Discrete
  - Continuous
- Kind of giving of inputs and getting of outputs
  - State of synapses
  - State of neurons
  - Weights of synapses

Tasks solved by neural networks

- Classification (recognition) in games
  - In diagnostic systems
  - In monitoring systems
  - In recognition systems of robots
  - In speech recognition
  - In security systems for authentication
- Clusterization
  - In Data Mining for extracting of knowledge
  - In search systems for indexing of documents
- Prediction in games
  - In financial analyzing
  - In control systems of mobile robots
- Approximation
  - In control systems of technological processes

Neural Networks and Game AI

- simplify coding of complex state machines or rules-based systems
- potential for AI to adapt as the game is played

Jeff Hannan, creator of AI for **Colin McRae Rally 2.0** →

"I am very confident about it as well, because I understand what the neural net is doing. I haven't just created a big mysterious black box, I can map out the internal workings of it."

Colin McRae Rally 2.0

- Used standard feedforward multilayer perceptron neural network
- Constructed with the simple aim of keeping the car to the racing line

BATTLECRUISER: 3000AD

- 1998 space simulation game
- 'I am a Gammulan Criminal up against a Terran EarthCOM ship. My ship is 50% damaged but my weapon systems are fully functional and my goal is still unresolved' what do I do?

http://www.3000ad.com/shots/bc3k.shtml

Black and White

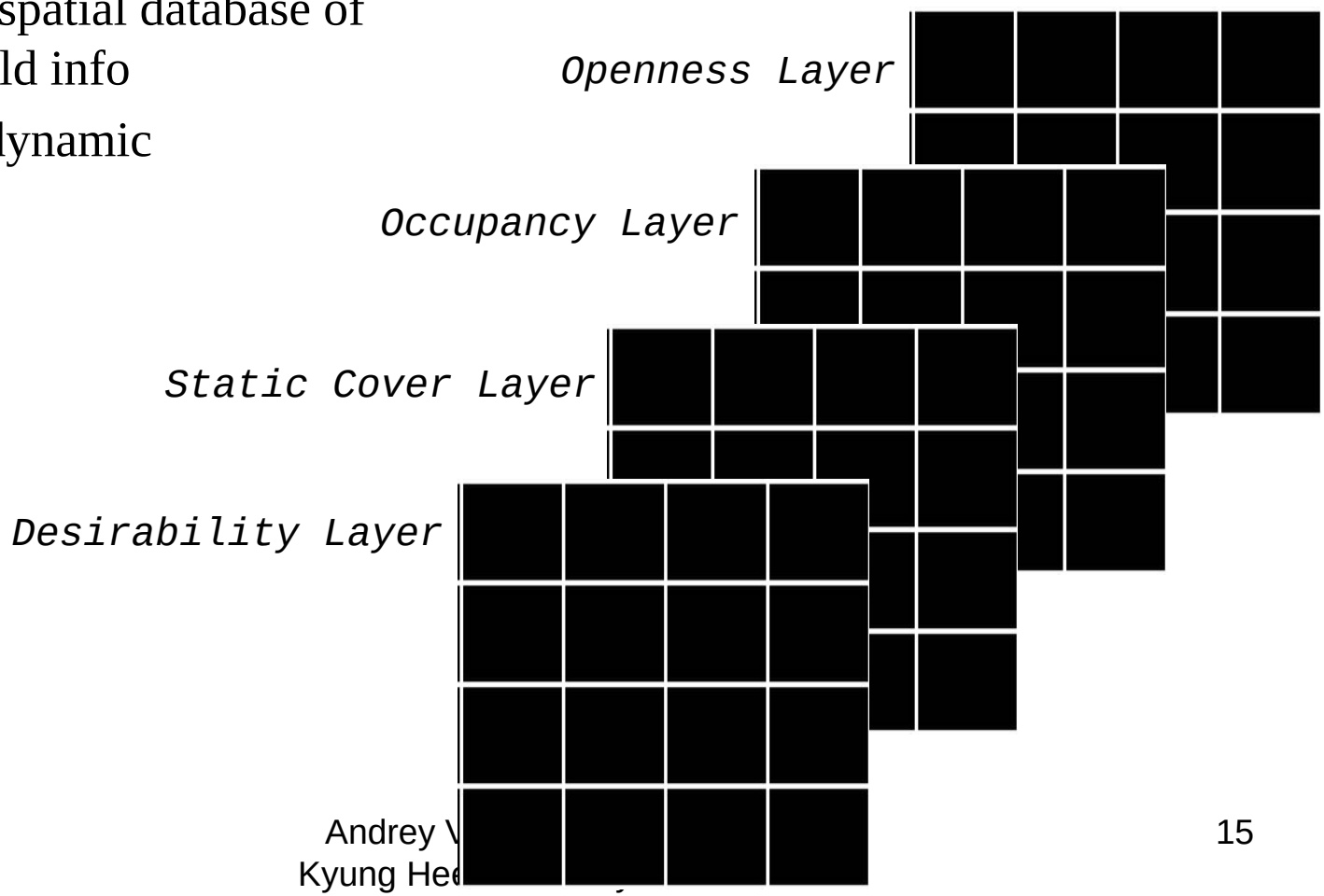
- Uses neural networks to teach the creature behaviors
- “Tickling” increase certain weights
- Hitting reduces certain weights

Usefulness in Games

- *Control* – motor controller, for example, control of a race car or airplane
- *Threat Assessment* – input number of enemy ground units, aerial units, etc.
- *Attack or Flee* – input health, distance to enemy, class of enemy, etc.
- *Anticipation* – Predicting player's next move, input previous moves, output prediction

Influence Maps Refresher

- Creates a spatial database of game world info
- Static or dynamic



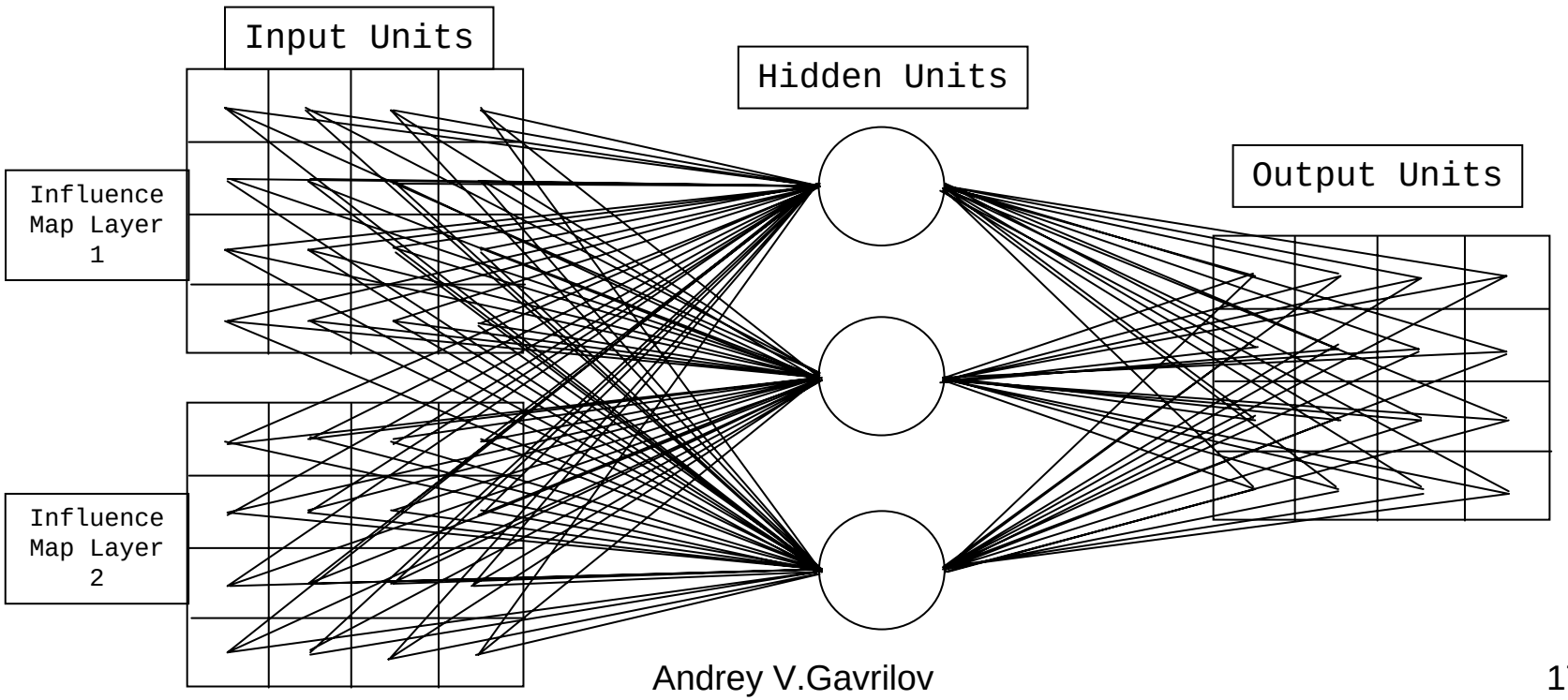
Neural Networks and Influence Maps

- Usually influence map layers are combined using a weighted sum
- Neural network can be used instead

- Finding correct weight for each layer usually results in trial and error
- Choosing relevant layers can be difficult
- Potential loss of important information if factors cancel each other

Computational Complexity

- Could lead to a very large number of calculations



Optimizations

- Only analyze relevant portion of map
- Reduce grid resolution
- Train network during development, not in-game

Design Details

- Need an input for each cell in each layer of the influence map
- Need one output for each cell on map
- Hidden units are arbitrary, usually 10-20 with some guess and test to prune it

Different Decisions and Personalities

- One network for each decision
- Implemented as one network with a different array of weights for each decision
- Different personalities can have multiple arrays of weights for each decision

Training

- Datasets from gaming sessions of human vs. human are best
- Must decide whether training will occur in-game, during development, or both
- Learning during play provides for adaptations against individual players

Conclusions

- Neural networks provide ability to provide more human-like AI
- Takes rough approximation and hard-coded reactions out of AI design (i.e. Rules and FSMs)
- Still require a lot of fine-tuning during development