Lecture 27

11/28/07 1

Spatial Effects

- Previous simulation assumes that each agent is equally likely to interact with each other
- So strategy interactions are proportional to fractions in population
- More realistically, interactions with "neighbors" are more likely
  - "Neighbor" can be defined in many ways
- Neighbors are more likely to use the same strategy

11/28/07 2

Spatial Simulation

- Toroidal grid
- Agent interacts only with eight neighbors
- Agent adopts strategy of most successful neighbor
- Ties favor current strategy

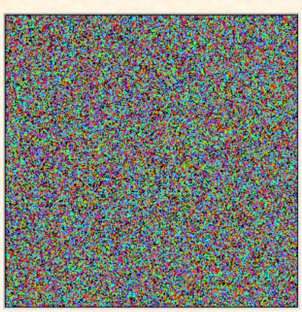
11/28/07 3

NetLogo Simulation of Spatial IPD

Run SIPD.nlogo

11/28/07 4

Typical Simulation (t = 1)

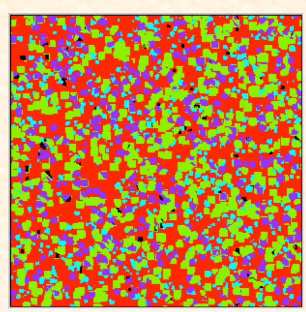


Colors:

ALL-C
TFT
RAND
PAV
ALL-D

11/28/07 5

Typical Simulation (t = 5)

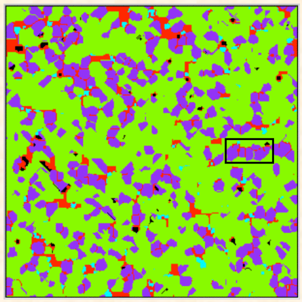


Colors:

ALL-C
TFT
RAND
PAV
ALL-D

11/28/07 6

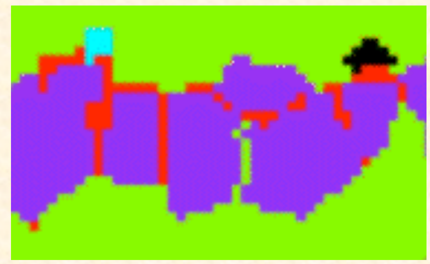
Typical Simulation (t = 10)



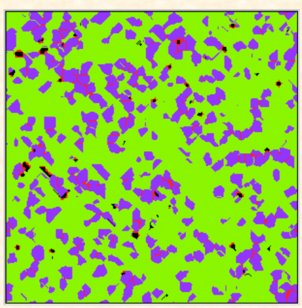
Colors:

ALL-C
TFT
RAND
PAV
ALL-D

Typical Simulation (t = 10) Zooming In



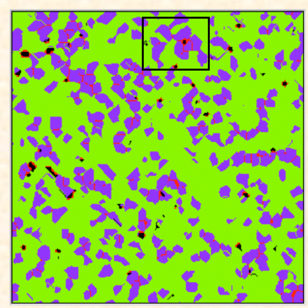
Typical Simulation (t = 20)



Colors:

ALL-C
TFT
RAND
PAV
ALL-D

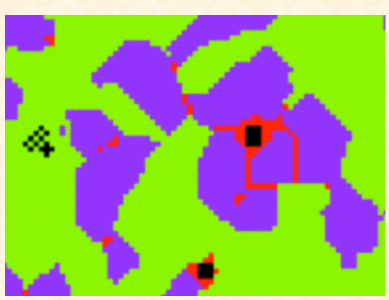
Typical Simulation (t = 50)



Colors:

ALL-C
TFT
RAND
PAV
ALL-D

Typical Simulation (t = 50) Zoom In



SIPD Without Noise

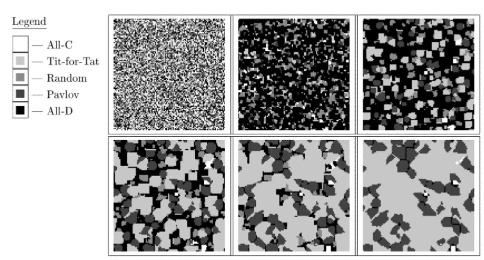


Figure 17.4 Competition in the spatial iterated Prisoner's Dilemma without noise

Conclusions: Spatial IPD

- Small clusters of cooperators can exist in hostile environment
- Parasitic agents can exist only in limited numbers
- Stability of cooperation depends on expectation of future interaction
- Adaptive cooperation/defection beats unilateral cooperation or defection

Additional Bibliography

1. von Neumann, J., & Morgenstern, O. *Theory of Games and Economic Behavior*, Princeton, 1944.
2. Morgenstern, O. "Game Theory," in *Dictionary of the History of Ideas*, Charles Scribners, 1973, vol. 2, pp. 263-75.
3. Axelrod, R. *The Evolution of Cooperation*. Basic Books, 1984.
4. Axelrod, R., & Dion, D. "The Further Evolution of Cooperation," *Science* **242** (1988): 1385-90.
5. Poundstone, W. *Prisoner's Dilemma*. Doubleday, 1992.

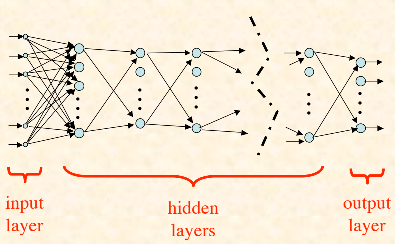
Part VII

VII. Neural Networks and Learning

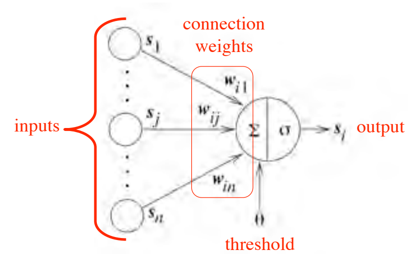
Supervised Learning

- Produce desired outputs for training inputs
- Generalize reasonably & appropriately to other inputs
- Good example: pattern recognition
- Feedforward multilayer networks

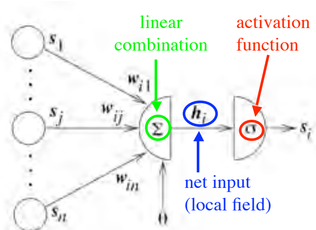
Feedforward Network



Typical Artificial Neuron



Typical Artificial Neuron



Equations

Net input: $h_i = \left(\sum_{j=1}^{n} w_{ij} s_j \right) - \theta$

$\mathbf{h} = \mathbf{W}\mathbf{s} - \theta$

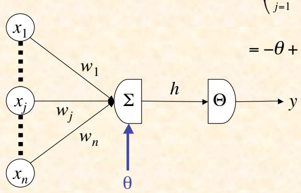
Neuron output: $s'_i = \sigma(h_i)$

$\mathbf{s}' = \sigma(\mathbf{h})$

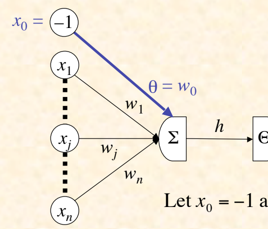
Treating Threshold as Weight

$$h = \left(\sum_{j=1}^{n} w_j x_j \right) - \theta$$

$$= -\theta + \sum_{j=1}^{n} w_j x_j$$



Treating Threshold as Weight



$$h = \left(\sum_{j=1}^{n} w_j x_j \right) - \theta$$

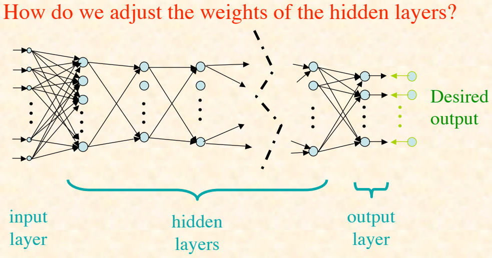
$$= -\theta + \sum_{j=1}^{n} w_j x_j$$

Let $x_0 = -1$ and $w_0 = \theta$

$$h = w_0 x_0 + \sum_{j=1}^{n} w_j x_j = \sum_{j=0}^{n} w_j x_j = \tilde{\mathbf{w}} \cdot \tilde{\mathbf{x}}$$

Credit Assignment Problem

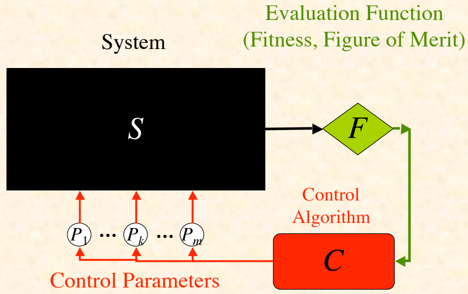
How do we adjust the weights of the hidden layers?



NetLogo Demonstration of Back-Propagation Learning

Run Artificial Neural Net.nlogo

Adaptive System



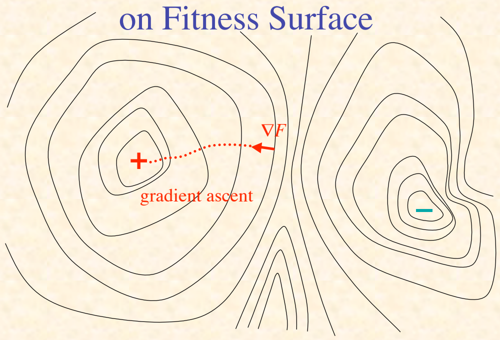
Gradient

$\dfrac{\partial F}{\partial P_k}$ measures how F is altered by variation of P_k

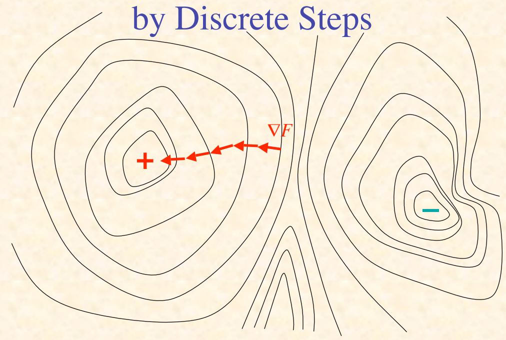
$$\nabla F = \begin{pmatrix} \partial F / \partial P_1 \\ \vdots \\ \partial F / \partial P_k \\ \vdots \\ \partial F / \partial P_m \end{pmatrix}$$

∇F points in direction of maximum increase in F

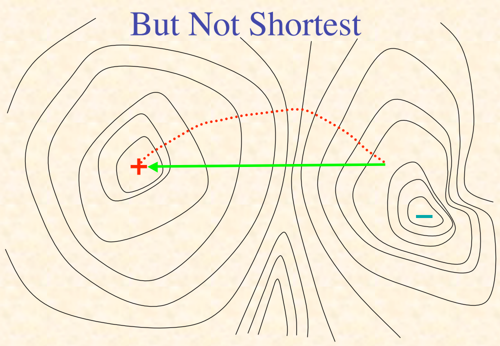
Gradient Ascent on Fitness Surface



Gradient Ascent by Discrete Steps



Gradient Ascent is Local But Not Shortest



Gradient Ascent Process

$$\dot{\mathbf{P}} = \eta \nabla F(\mathbf{P})$$

Change in fitness:

$$\dot{F} = \frac{\mathrm{d}F}{\mathrm{d}t} = \sum_{k=1}^{m} \frac{\partial F}{\partial P_k}\frac{\mathrm{d}P_k}{\mathrm{d}t} = \sum_{k=1}^{m} (\nabla F)_k \dot{P}_k$$

$$\dot{F} = \nabla F \cdot \dot{\mathbf{P}}$$

$$\dot{F} = \nabla F \cdot \eta \nabla F = \eta \|\nabla F\|^2 \geq 0$$

Therefore gradient ascent increases fitness
(until reaches 0 gradient)

General Ascent in Fitness

Note that any adaptive process $\mathbf{P}(t)$ will increase fitness provided:

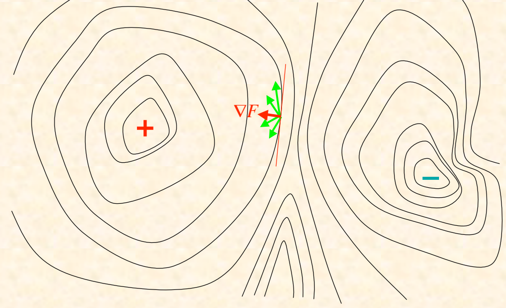
$$0 < \dot{F} = \nabla F \cdot \dot{\mathbf{P}} = \|\nabla F\| \|\dot{\mathbf{P}}\| \cos\varphi$$

where φ is angle between ∇F and $\dot{\mathbf{P}}$

Hence we need $\cos\varphi > 0$

or $|\varphi| < 90^\circ$

General Ascent on Fitness Surface



Fitness as Minimum Error

Suppose for Q different inputs we have target outputs $\mathbf{t}^1, \ldots, \mathbf{t}^Q$

Suppose for parameters $\mathbf{P}$ the corresponding actual outputs are $\mathbf{y}^1, \ldots, \mathbf{y}^Q$

Suppose $D(\mathbf{t},\mathbf{y}) \in [0,\infty)$ measures difference between target & actual outputs

Let $E^q = D(\mathbf{t}^q, \mathbf{y}^q)$ be error on qth sample

Let $F(\mathbf{P}) = -\sum_{q=1}^{Q} E^q(\mathbf{P}) = -\sum_{q=1}^{Q} D[\mathbf{t}^q, \mathbf{y}^q(\mathbf{P})]$

Gradient of Fitness

$$\nabla F = \nabla\left(-\sum_q E^q\right) = -\sum_q \nabla E^q$$

$$\frac{\partial E^q}{\partial P_k} = \frac{\partial}{\partial P_k} D(\mathbf{t}^q, \mathbf{y}^q) = \sum_j \frac{\partial D(\mathbf{t}^q, \mathbf{y}^q)}{\partial y_j^q} \frac{\partial y_j^q}{\partial P_k}$$

$$= \frac{\mathrm{d}\, D(\mathbf{t}^q, \mathbf{y}^q)}{\mathrm{d}\, \mathbf{y}^q} \cdot \frac{\partial \mathbf{y}^q}{\partial P_k}$$

$$= \nabla_{\mathbf{y}^q} D(\mathbf{t}^q, \mathbf{y}^q) \cdot \partial \mathbf{y}^q / \partial P_k$$

Jacobian Matrix

Define Jacobian matrix $\mathbf{J}^q = \begin{pmatrix} \partial y_1^q / \partial P_1 & \cdots & \partial y_1^q / \partial P_m \\ \vdots & \ddots & \vdots \\ \partial y_n^q / \partial P_1 & \cdots & \partial y_n^q / \partial P_m \end{pmatrix}$

Note $\mathbf{J}^q \in \Re^{n \times m}$ and $\nabla D(\mathbf{t}^q, \mathbf{y}^q) \in \Re^{n \times 1}$

Since $(\nabla E^q)_k = \dfrac{\partial E^q}{\partial P_k} = \sum_j \dfrac{\partial y_j^q}{\partial P_k} \dfrac{\partial D(\mathbf{t}^q, \mathbf{y}^q)}{\partial y_j^q}$,

$\therefore \nabla E^q = (\mathbf{J}^q)^{\mathrm{T}} \nabla D(\mathbf{t}^q, \mathbf{y}^q)$

Derivative of Squared Euclidean Distance

Suppose $D(\mathbf{t},\mathbf{y}) = \|\mathbf{t} - \mathbf{y}\|^2 = \sum_i (t_i - y_i)^2$

$$\frac{\partial D(\mathbf{t} - \mathbf{y})}{\partial y_j} = \frac{\partial}{\partial y_j} \sum_i (t_i - y_i)^2 = \sum_i \frac{\partial (t_i - y_i)^2}{\partial y_j}$$

$$= \frac{\mathrm{d}(t_j - y_j)^2}{\mathrm{d}\, y_j} = -2(t_j - y_j)$$

$$\therefore \frac{\mathrm{d}\, D(\mathbf{t},\mathbf{y})}{\mathrm{d}\, \mathbf{y}} = 2(\mathbf{y} - \mathbf{t})$$

Gradient of Error on q^{th} Input

$$\frac{\partial E^q}{\partial P_k} = \frac{\mathrm{d}\,D(\mathbf{t}^q, \mathbf{y}^q)}{\mathrm{d}\,\mathbf{y}^q} \cdot \frac{\partial \mathbf{y}^q}{\partial P_k}$$

$$= 2(\mathbf{y}^q - \mathbf{t}^q) \cdot \frac{\partial \mathbf{y}^q}{\partial P_k}$$

$$= 2\sum_j (y_j^q - t_j^q) \frac{\partial y_j^q}{\partial P_k}$$

$$\nabla E^q = 2(\mathbf{J}^q)^{\mathrm{T}} (\mathbf{y}^q - \mathbf{t}^q)$$

11/28/07 37

Recap

$$\dot{\mathbf{P}} = \eta \sum_q (\mathbf{J}^q)^{\mathrm{T}} (\mathbf{t}^q - \mathbf{y}^q)$$

To know how to decrease the differences between actual & desired outputs,

we need to know elements of Jacobian, $\partial y_j^q / \partial P_k$,

which says how jth output varies with kth parameter (given the qth input)

The Jacobian depends on the specific form of the system, in this case, a feedforward neural network

11/28/07 38