Lecture 28

Gradient Ascent Process

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

Change in fitness:

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

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

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

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:

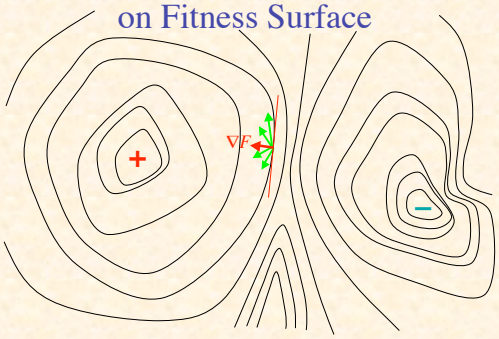
$$0 < \dot{F} = \nabla F \cdot \dot{\mathbf{P}} = \left\| \nabla F \right\| \left\| \dot{\mathbf{P}} \right\| \cos\varphi$$

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

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

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

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

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

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

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

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

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

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

Recap

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

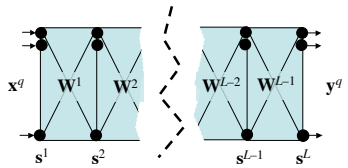
To know how to decrease the differences between actual & desired outputs,

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

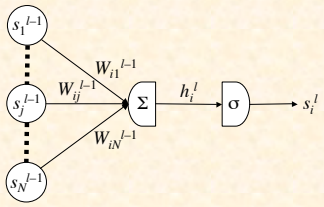
Multilayer Notation



Notation

- L layers of neurons labeled 1, …, L
- N_l neurons in layer l
- $\mathbf{s}^l$ = vector of outputs from neurons in layer l
- input layer $\mathbf{s}^1 = \mathbf{x}^q$ (the input pattern)
- output layer $\mathbf{s}^L = \mathbf{y}^q$ (the actual output)
- $\mathbf{W}^l$ = weights between layers l and l+1
- Problem: find how outputs y_i^q vary with weights W_{jk}^l (l = 1, …, L–1)

Typical Neuron



Error Back-Propagation

We will compute $\frac{\partial E^q}{\partial W_{ij}^l}$ starting with last layer ($l = L-1$)

and working back to earlier layers ($l = L-2,\ldots,1$)

Delta Values

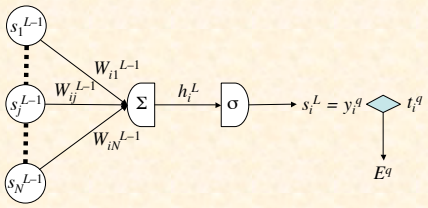
Convenient to break derivatives by chain rule:

$$\frac{\partial E^q}{\partial W_{ij}^{l-1}} = \frac{\partial E^q}{\partial h_i^l}\frac{\partial h_i^l}{\partial W_{ij}^{l-1}}$$

Let $\delta_i^l = \frac{\partial E^q}{\partial h_i^l}$

So $\frac{\partial E^q}{\partial W_{ij}^{l-1}} = \delta_i^l \frac{\partial h_i^l}{\partial W_{ij}^{l-1}}$

Output-Layer Neuron



Output-Layer Derivatives (1)

$$\delta_i^L = \frac{\partial E^q}{\partial h_i^L} = \frac{\partial}{\partial h_i^L}\sum_k \left(s_k^L - t_k^q\right)^2$$

$$= \frac{\mathrm{d}\left(s_i^L - t_i^q\right)^2}{\mathrm{d}\, h_i^L} = 2\left(s_i^L - t_i^q\right)\frac{\mathrm{d}\, s_i^L}{\mathrm{d}\, h_i^L}$$

$$= 2\left(s_i^L - t_i^q\right)\sigma'\left(h_i^L\right)$$

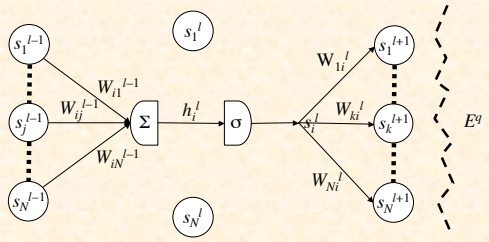
Output-Layer Derivatives (2)

$$\frac{\partial h_i^L}{\partial W_{ij}^{L-1}} = \frac{\partial}{\partial W_{ij}^{L-1}}\sum_k W_{ik}^{L-1} s_k^{L-1} = s_j^{L-1}$$

$$\therefore \frac{\partial E^q}{\partial W_{ij}^{L-1}} = \delta_i^L s_j^{L-1}$$

where $\delta_i^L = 2\left(s_i^L - t_i^q\right)\sigma'\left(h_i^L\right)$

Hidden-Layer Neuron



Hidden-Layer Derivatives (1)

Recall $\dfrac{\partial E^q}{\partial W_{ij}^{l-1}} = \delta_i^l \dfrac{\partial h_i^l}{\partial W_{ij}^{l-1}}$

$$\delta_i^l = \frac{\partial E^q}{\partial h_i^l} = \sum_k \frac{\partial E^q}{\partial h_k^{l+1}} \frac{\partial h_k^{l+1}}{\partial h_i^l} = \sum_k \delta_k^{l+1} \frac{\partial h_k^{l+1}}{\partial h_i^l}$$

$$\frac{\partial h_k^{l+1}}{\partial h_i^l} = \frac{\partial \sum_m W_{km}^l s_m^l}{\partial h_i^l} = \frac{\partial W_{ki}^l s_i^l}{\partial h_i^l} = W_{ki}^l \frac{\mathrm{d}\sigma(h_i^l)}{\mathrm{d}h_i^l} = W_{ki}^l \sigma'(h_i^l)$$

$$\therefore \delta_i^l = \sum_k \delta_k^{l+1} W_{ki}^l \sigma'(h_i^l) = \sigma'(h_i^l) \sum_k \delta_k^{l+1} W_{ki}^l$$

Hidden-Layer Derivatives (2)

$$\frac{\partial h_i^l}{\partial W_{ij}^{l-1}} = \frac{\partial}{\partial W_{ij}^{l-1}} \sum_k W_{ik}^{l-1} s_k^{l-1} = \frac{\mathrm{d} W_{ij}^{l-1} s_j^{l-1}}{\mathrm{d} W_{ij}^{l-1}} = s_j^{l-1}$$

$$\therefore \frac{\partial E^q}{\partial W_{ij}^{l-1}} = \delta_i^l s_j^{l-1}$$

where $\delta_i^l = \sigma'(h_i^l) \sum_k \delta_k^{l+1} W_{ki}^l$

Derivative of Sigmoid

Suppose $s = \sigma(h) = \dfrac{1}{1+\exp(-\alpha h)}$ (logistic sigmoid)

$$\mathrm{D}_h s = \mathrm{D}_h \left[1+\exp(-\alpha h)\right]^{-1} = -\left[1+\exp(-\alpha h)\right]^{-2} \mathrm{D}_h \left(1+e^{-\alpha h}\right)$$

$$= -\left(1+e^{-\alpha h}\right)^{-2} \left(-\alpha e^{-\alpha h}\right) = \alpha \frac{e^{-\alpha h}}{\left(1+e^{-\alpha h}\right)^2}$$

$$= \alpha \frac{1}{1+e^{-\alpha h}} \frac{e^{-\alpha h}}{1+e^{-\alpha h}} = \alpha s \left(\frac{1+e^{-\alpha h}}{1+e^{-\alpha h}} - \frac{1}{1+e^{-\alpha h}}\right)$$

$$= \alpha s (1-s)$$

Summary of Back-Propagation Algorithm

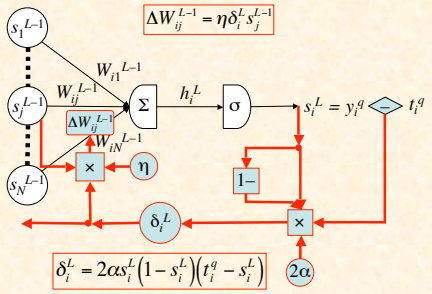
Output layer: $\delta_i^L = 2\alpha s_i^L \left(1-s_i^L\right)\left(s_i^L - t_i^q\right)$

$$\frac{\partial E^q}{\partial W_{ij}^{L-1}} = \delta_i^L s_j^{L-1}$$

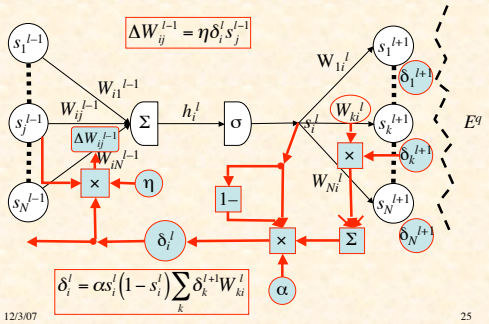
Hidden layers: $\delta_i^l = \alpha s_i^l \left(1-s_i^l\right) \sum_k \delta_k^{l+1} W_{ki}^l$

$$\frac{\partial E^q}{\partial W_{ij}^{l-1}} = \delta_i^l s_j^{l-1}$$

Output-Layer Computation



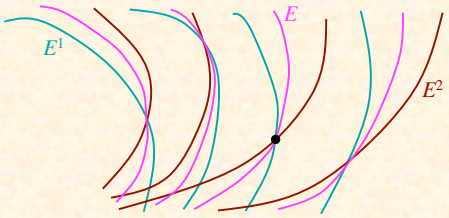
Hidden-Layer Computation



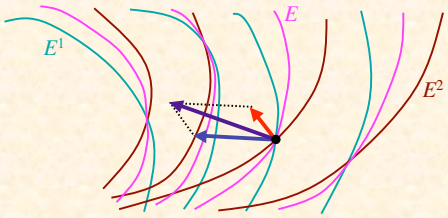
Training Procedures

- Batch Learning
  - on each *epoch* (pass through all the training pairs),
  - weight changes for all patterns accumulated
  - weight matrices updated at end of epoch
  - accurate computation of gradient
- Online Learning
  - weight are updated after back-prop of each training pair
  - usually randomize order for each epoch
  - approximation of gradient
- Doesn't make much difference

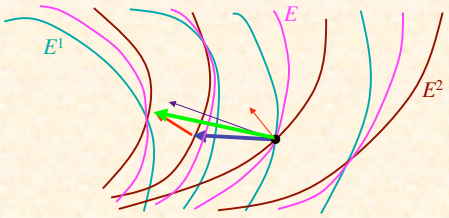
Summation of Error Surfaces



Gradient Computation in Batch Learning



Gradient Computation in Online Learning



The Golden Rule of Neural Nets

Neural Networks are the
second-best way
to do *everything*!

VIII. Review of Key Concepts

Complex Systems

- Many interacting elements
- Local vs. global order: entropy
- Scale (space, time)
- Phase space
- Difficult to understand
- Open systems

Many Interacting Elements

- Massively parallel
- Distributed information storage & processing
- Diversity
  - avoids premature convergence
  - avoids inflexibility

Complementary Interactions

- Positive feedback / negative feedback
- Amplification / stabilization
- Activation / inhibition
- Cooperation / competition
- Positive / negative correlation

Emergence & Self-Organization

- Microdecisions lead to macrobehavior
- Circular causality (macro / micro feedback)
- Coevolution
  - predator/prey, Red Queen effect
  - gene/culture, niche construction, Baldwin effect

Pattern Formation

- Excitable media
- Amplification of random fluctuations
- Symmetry breaking
- Specific difference vs. generic identity
- Automatically adaptive

Stigmergy

- Continuous (quantitative)
- Discrete (qualitative)
- Coordinated algorithm
  - non-conflicting
  - sequentially linked

Emergent Control

- Stigmergy
- Entrainment (distributed synchronization)
- Coordinated movement
  - through attraction, repulsion, local alignment
  - in concrete or abstract space
- Cooperative strategies
  - nice & forgiving, but reciprocal
  - evolutionarily stable strategy

Attractors

- Classes
  - point attractor
  - cyclic attractor
  - chaotic attractor
- Basin of attraction
- Imprinted patterns as attractors
  - pattern restoration, completion, generalization, association

Wolfram's Classes

- Class I: point
- Class II: cyclic
- Class III: chaotic
- Class IV: complex (edge of chaos)
  - persistent state maintenance
  - bounded cyclic activity
  - global coordination of control & information
  - order for free

Energy / Fitness Surface

- Descent on energy surface / ascent on fitness surface
- Lyapunov theorem to prove asymptotic stability / convergence
- Soft constraint satisfaction / relaxation
- Gradient (steepest) ascent / descent
- Adaptation & credit assignment

Biased Randomness

- Exploration vs. exploitation
- Blind variation & selective retention
- Innovation vs. incremental improvement
- Pseudo-temperature
- Diffusion
- Mixed strategies

Natural Computation

- Tolerance to noise, error, faults, damage
- Generality of response
- Flexible response to novelty
- Adaptability
- Real-time response
- Optimality is secondary

Student Course Evaluation!
(We will do it in class this time)