Acknowledgement:
Many of these slides are derived from Tom Mitchell, Pascal Poupart, Pieter Abbeel, Eric Eaton, Carlos Guestrin, William Cohen, and Andrew Moore.
Backpropagation Algorithm

- Two phases:
  - Forward phase: compute output $z_j$ for each node $j$
  - Backward phase: compute output $\delta_j$ for each node $j$
Forward Phase

- Propagate inputs forward through the network to compute the output of each node
- Output $z_j$ at node $j$
  - $z_j = h(a_j)$ where $a_j = \sum_i w_{ji} z_i$
Backward Phase

- Use chain rule to recursively compute gradient
  - For each weight $w_{ji}$
    \[ \delta_{ji} = \frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}} \]
  - Let $\delta_j = \frac{\partial E_n}{\partial a_j}$
  - Then $\delta_j = \begin{cases} h'(a_j)(z_j - y_j) & \text{base case: } j \in \text{output nodes} \\ h'(a_j) \sum_k w_{kj}\delta_k & \text{recursion: } j \in \text{hidden nodes} \end{cases}$
  - Since $a_j = \sum_k w_{ji}z_i$ then $\frac{\partial a_j}{\partial w_{ji}} = z_i$
  - Therefore, $\delta_{ji} = \frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}} = \delta_j z_i$
Analysis

- Efficient gradient computation: linear in number of weights

- Convergence:
  - Slow convergence (linear rate)
  - May get trapped in local optima

- Prone to overfitting:
  - Solution: early stopping, regularization (add penalty term to the objective function), dropout
Slow Convergence

- **Issue**: gradient is not always ideal
- **Illustration**:
Adaptive Gradients

- **Idea**: adjust the learning rate of each dimension separately

- **AdaGrad**:
  - \[ r_t \leftarrow r_{t-1} + \left( \frac{\partial E_n}{\partial w_{ji}} \right)^2 \] (sum of squares of partial derivative)
  - \[ w_{ji} \leftarrow w_{ji} - \frac{\eta}{\sqrt{r_t}} \frac{\partial E_n}{\partial w_{ji}} \] (update rule)

- **Problem**: learning rate \( \frac{\eta}{\sqrt{r_t}} \) decays too quickly
RMSprop

- **Idea**: divide by root mean square (RMS) (instead of square root of the sum) of partial derivatives

- **RMSprop**

  \[ r_t \leftarrow \alpha r_{t-1} + (1 - \alpha) \left( \frac{\partial E_n}{\partial w_{ji}} \right)^2 \quad (0 \leq \alpha \leq 1) \]

- \[ w_{ji} \leftarrow w_{ji} - \eta \frac{\partial E_n}{\sqrt{r_t} \partial w_{ji}} \] (update rule)

- **Problem**: gradient lacks momentum
Adaptive Moment Estimation

- **Idea**: replace gradient by its moving average to induce momentum

- **Adam**:
  
  \[
  r_t \leftarrow \alpha r_{t-1} + (1 - \alpha) \left( \frac{\partial E_n}{\partial w_{ji}} \right)^2 \quad (0 \leq \alpha \leq 1)
  \]

  \[
  s_t \leftarrow \beta s_{t-1} + (1 - \beta) \left( \frac{\partial E_n}{\partial w_{ji}} \right) \quad (0 \leq \beta \leq 1)
  \]

- \( w_{ji} \leftarrow w_{ji} - \frac{\eta}{\sqrt{r_t}} s_t \) (update rule)