CMSC 491N/691N **Introduction to Neural Networks** **Spring 2001**

Review 1 (Chapters. 1, 2, 6, 3)

1. **Basics**
   - Comparison between human brain and von Neumann architecture
   - Processing units
   - Activation/output functions (threshold, linear-threshold, sigmoid)
   - Network architecture (hidden nodes, feed-forward/recurrent nets, layered)
   - Connection and weights
   - Types of learning (supervised/unsupervised), Hebbian rule
2. **Single Layer networks** (Perceptron, Adaline, and the delta rule)
   - Architecture
   - Decision boundary and the problem of linear separability ($\boldsymbol{b} + \sum_{\boldsymbol{i}=1}^{\boldsymbol{n}} \boldsymbol{x_i w_i} = 0$)
   - Hebbian nets ($\Delta \boldsymbol{w_i} = \boldsymbol{x_i} \cdot \boldsymbol{t}$)
   - Perceptron learning rule (only when $\boldsymbol{t} \neq \boldsymbol{y} : \Delta \boldsymbol{w_i} = a \cdot \boldsymbol{x_i} \cdot \boldsymbol{t}$)
   - Perceptron convergence theorem
   - Delta learning rule in Adaline (driven by error: $\Delta \boldsymbol{w_i} = a \cdot \boldsymbol{x_i} \cdot (\boldsymbol{t} - \boldsymbol{y\_in})$)
   - Gradient descent approach in deriving delta learning rule
     squared error: $\boldsymbol{E} = (\boldsymbol{t} - \boldsymbol{y\_in})^2$ or $\boldsymbol{E} = \sum_{\boldsymbol{p}=1}^{\boldsymbol{P}} (\boldsymbol{t}(\boldsymbol{p}) - \boldsymbol{y\_in}(\boldsymbol{p}))^2$
     $\Delta \boldsymbol{w_i} \propto -\partial \boldsymbol{E} / \partial \boldsymbol{w_i}$
3. **Backpropagation (BP) Networks**
   - Multi-layer feed-forward architecture with hidden nodes of non-linear and differentiable activation functions
   - Motivation to have hidden nodes (representational power). Why non-linear?
   - Feed forward computing
   - BP learning
     - Training samples
     - Obtain errors at output layer (feed-forward phase): $\boldsymbol{d_k} = (\boldsymbol{t_k} - \boldsymbol{y_k}) \boldsymbol{f}'(\boldsymbol{y\_in_k})$
     - Obtain errors at hidden layer (error backpropagation phase): $\boldsymbol{d_j} = \boldsymbol{d\_in_j} \cdot \boldsymbol{f}'(\boldsymbol{y\_in_j})$
       and $\boldsymbol{d\_in_j} = \sum_{\boldsymbol{k}=1}^{\boldsymbol{m}} \boldsymbol{d_k w_{jk}}$
     - Learning procedure (batch and sequential modes)
     - In what sense BP learning generalizes delta rule of Adaline
     - Why BP learning works (gradient descent to minimize error): $\Delta \boldsymbol{w_{ij}} = -a \cdot \partial \boldsymbol{E} / \partial \boldsymbol{w_{ij}}$
   - Issues of practical concerns
     - Bias, error bound, training data, initial weights, number and size of hidden layers;
     - Learning rate (momentum, adaptive rate)
   - Advantages and problems with BP learning
     - Powerful (general function approximator); easy to use; wide applicability; good generalization
     - Local minima; overfitting; parameters may be hard to determine; network paralysis; long learning time, hard to accommodate new samples (non-incremental learning)

4. **Pattern Association and Associative memory (AM)**

- Simple AM
  Associative memory (AM) (content-addressable/associative recall; pattern correction/completion)
  Network architecture: single layer or two layers of non-linear units
- Hebbian rule: $w_{ij} = \sum_{p=1}^{P} s_i(p) \cdot t_j(p)$
  - Correlation matrix: $W = \sum_{p=1}^{P} s^T(p) \cdot t(p)$ (assuming both ***s*** and ***t*** are row vectors).
  - Principal and cross-talk term: $s(k)W = \|s(k)\| + \sum_{p \neq k} s(k)s^T(p) \cdot t(p)$
- Delta rule: $\Delta w_{ij} = a \cdot (t_j - y_j) \cdot x_i$ or , $\Delta w_{ij} = a \cdot (t_j - y_j) \cdot x_i \cdot f'(y\_in_j)$ derived following gradient descent ($\Delta w_{ij} = -a \cdot \partial E / \partial w_{ij}$)
- Auto-associative memory: $t(p) = s(p),\ p = 1, 2, \ldots P$.
  Storage capacity: up to ***n-1*** mutually orthogonal patterns of dimension ***n***
- Iterative autoassociative memory
  - Motives (comparing with non-iterative recall)
  - Using the output of the current iteration as input of the next iteration (stop when a state repeats)
  - Dynamic system (stable states, attractors, genuine and spurious memories)
- Hopfield model for autoassociative memory
  - Network architecture (single-layer, fully connected, recurrent)
  - Weight matrix for Hopfield model (symmetric with zero diagonal elements)

$$w_{ij} = \begin{cases} \sum_{p=1}^{P} s_i(p) \cdot s_j(p) & \text{if } i \neq j \\ 0 & \text{otherwise} \end{cases}$$

  - Recall procedure (iterative until stabilized)
  - Stability of dynamic systems
    - Ideas of Lyapunov function/energy function (monotonically non-increasing and bounded from below)
    - Convergence of Hopfield AM: its an energy function
    - $$E = -0.5 \sum_{i \neq j} \sum_{j} y_i y_j w_{ij} - \sum_i x_i y_i + \sum_i q_i y_i$$
  - Storage capacity of Hopfield AM ($P \approx n / (2 \log_2 n)$).
- Bidirectional AM (BAM)
  - Architecture: two layers of non-linear units
  - Weight matrix: $W_{n \times m} = \sum_{p=1}^{P} s^T(p) \cdot t(p)$
  - Recall: bi-directional (from ***x*** to ***y*** and ***y*** to ***x***); recurrent
  - Analysis
    - Energy function: $L = -XWY^T = -\sum_{j=1}^{m} \sum_{i=1}^{n} x_i w_{ij} y_j$
    - Storage capacity: $P = O(\max(n, m))$