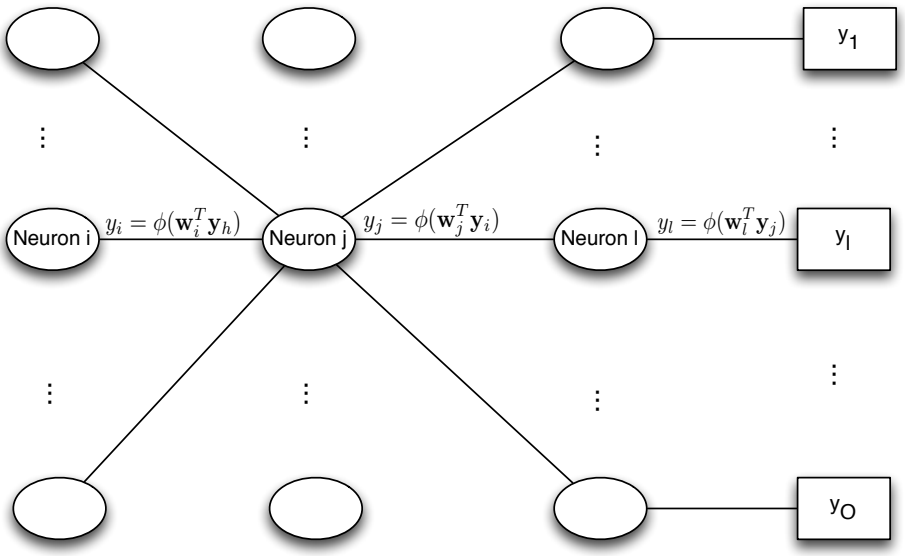
Lecture 26 - (Probably Final) Lecture on Backpropagation & Feature Selection

November 8, 2017

1 Error Backpropagation

- Now to address hidden layers... We have to deal with the credit assignment problem.
- Now, the hidden layers...



- Suppose we want to update w_{ji} where j is a hidden layer.
- The error objective function overall N data points is

$$E(n) = \frac{1}{2}\sum_{n=1}^{N} e_n^2 = \sum_{n=1}^{N} (d_n - y_n)^2 = \sum_{n=1}^{N} (d_n - \phi_l(v_l(n)))^2 \quad (1)$$

$$\begin{aligned} \frac{\partial E(n)}{\partial w_{lj}} &= \frac{\partial E(n)}{\partial e_l(n)} \frac{\partial e_l(n)}{\partial y_l(n)} \frac{\partial y_l(n)}{\partial v_l(n)} \frac{\partial v_l(n)}{\partial w_{lj}} && (2) \\ &= [e_l][-1][\phi'(v_l(n))][y_{jl}(n)] && (3) \\ & && (4) \end{aligned}$$

- Let's define a *local gradient* $\delta_l(n)$:

$$\begin{aligned} \delta_l(n) &= -\frac{\partial E(n)}{\partial v_l(n)} && (5) \\ &= e_l(n)\phi'(v_l(n)) && (6) \end{aligned}$$

- Similarly,

$$\begin{aligned} \delta_j(n) &= -\frac{\partial E(n)}{\partial v_j(n)} && (7) \\ &= -\frac{\partial E(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} && (8) \\ &= -\frac{\partial E(n)}{\partial y_j(n)} \phi'(v_j(n)) && (9) \end{aligned}$$

- Note that:

$$\begin{aligned} \frac{\partial E(n)}{\partial y_j(n)} &= \sum_l e_l(n) \frac{\partial e_l(n)}{\partial y_j(n)} && (10) \\ &= \sum_l e_l(n) \frac{\partial e_l(n)}{\partial v_l(n)} \frac{\partial v_l(n)}{\partial y_j(n)} && (11) \\ &= \sum_l e_l(n)[-\phi'(v_l(n))][w_{kj}(n)] && (12) \end{aligned}$$

- So,

$$\delta_j(n) = -\frac{\partial E(n)}{\partial y_j(n)}\phi'(v_j(n)) \quad (13)$$

$$= -\left[\sum_l e_l(n)[-\phi'(v_l(n))][w_{kj}(n)]\right]\phi'(v_j(n)) \quad (14)$$

$$= \phi'(v_j(n))\sum_l \delta_l(n)w_{kj}(n) \quad (15)$$

- So, you can write the gradient at a hidden neuron in terms of the local gradient and the connected neurons in the next layer

$$\delta w_{ij}(n) = \eta\delta_j(n)y_i(n) \quad (16)$$

2 Rate of Learning

- Back-propagation algorithm provides an approximation to the trajectory in the weight space computed by the method of steepest descent
- Small η: smoother trajectory, slower learning
- Too Large η: large changes in the synaptic weights, may become unstable (oscillatory)
- **Include a momentum term**: tries to increase rate of learning while avoiding instability
- Recall: $\Delta w_{ji}(n) = \eta\delta_j(n)y_i(n) = \eta\frac{\partial E(n)}{\partial v_j(n)}y_i(n)$
- **The Generalized Delta rule (Delta rule with momentum):**
  $\Delta w_{ji}(n) = \alpha\Delta w_{ji}(n-1) + \eta\delta_j(n)y_i(n)$
- Can write the generalized delta rule as a time series (index t):

$$\Delta w_{ji}(n) = -\eta\sum_{t=0}^{n}\alpha^{(n-t)}\delta_j(t)y_i(t) \quad (17)$$

$$= -\eta\sum_{t=0}^{n}\alpha^{(n-t)}\frac{\partial E(t)}{\partial w_{ji}(t)} \quad (18)$$

- Observations & Comments:

    - $\Delta w_{ji}(n)$ is the sum of exponentially weighted time series. For it to converge, $0 \leq |\alpha| < 1$, If $\alpha = 0$, then you are operating without momentum
    - Inclusion of momentum accelerate descent in steady downhill directions
    - Inclusion of momentum has a stabilizing effect in directions that oscillate in sign
    - Momentum may prevent termination/convergence in a shallow minimum
    - The learning rate *can* be connection dependent, η_{ji}, can even set it to zero for some connections
  - See ADAM paper (posted on Canvas) for a more current approach for learning rate adaptation.

3 Online vs. Batch Update

- Thus far we have focused on Online learning. *What is meant by online learning?*
- You can also do: Batch update or (Stochastic) Mini-Batch Update

4 Overfitting, Generalization Ability, and Cross Validation

- *What is overfitting?*
- *What is cross validation?*
- *What does it mean to have good generalizability?*
- *What about unexplored regions of the feature space?*

- If a neural network learns too many input-output samples perfectly, the network may end up memorizing the training data (and its noise) = overfitting or overtraining
- Generalization is mostly influenced by three factors:
  1. Size of training set (may or may not have control)

2. Architecture of NN (have control)
3. Physical complexity of problem (no control)

- Goal is to reduce generalization error (error when given inputs from the test set)
- When complexity is small compared to sample size, performance is generally insensitive to training data size
- As target function gets more complex, relative to N, the size of the training set has more effect

5 A Sampling of Heuristics/Methods for making Back-Propagation Algorithm Perform Better

- Maximizing Information Content
  - Provide training samples that provide the largest information content
  - Use example with largest training error
  - Use example radically different than the ones before
  - Emphasizing scheme, present more difficult patterns to the network, difficulty is determined by error
  - Problems with emphasizing scheme:
    - Distribution of samples in an epoch is distorted
    - Outliers or mislabeled samples can cause major problems
- Activation Function: MLPs may learn better with activation functions that are antisymmetric rather than nonsymmetric. *What about RELU?*

$$\phi(-v) = -\phi(v) \tag{19}$$

- Target values need to be in the range of the sigmoid.
- Normalization (*What about batch normalization?*)
  - It is good if the input variables are uncorrelated
  - It is good if variances are approximately equal
- Initialization

6 Network Pruning Techniques

- **Network Growing:** Start with a small MLP and add to it when unable to meet design specifications
- **Network Pruning:** Start with a large MLP and prune it by eliminating weights (driving them to zero)
- **Complexity Regularization:** Need an appropriate trade-off between reliability of training data and goodness of the model/NN architecture
- Can realize trade-off my minimizing the total risk: $R(w) = E_s(w) + \lambda E_c(w)$ where E_s is the performance/error measure, E_c is the complexity penalty, and λ is a regularization parameter that represents the relative importance of the complexity penalty with respect to the performance measure term
  - $\lambda = 0$: Training based only on training samples
  - $\lambda \to \infty$: Training samples are unreliable, Minimize complexity

6.1 Weight Decay

$$E_c(w) = \|\mathbf{w}\|^2 = \sum_i w_i^2 \tag{20}$$

- Drives some weights to be small.
- Weights with litle or no influence are excess/unnecessary weights. The goal: encourage unnecessary weights to go to zero and thereby improve generalization or at least have the smallest values that can solve the problem.
- *So, how would you include weight decay in your Back-Propagation Training Algorithm?*

7 Forward and Backward Feature Selection

- Sequential Forward Feature Selection and Sequential Backward Feature Selection are wrapper methods for feature selection

- FFS: Sequentially add features, evaluating performance with some pre-determined performance metric with each potential next feature, adding the feature that improves results, this is a simple greedy search
- BFS: Sequentially remove features, evaluating performance with some performance metric, removing the feature that does least reduces the performance metric when removed
- Downsides to these approaches: Slow, Specific to metric and problems used to determine feature set
- Need a performance metric for these approaches.
  - Best to be problem dependent
  - For classification, can use classifier accuracy on training data? (requires training for each potential feature set, which can add significantly to time)
  - Can also use ability to maintain class separability:
    * Cluster validity metrics (e.g. Dunn's Index, Davies-Bouldin Index, Silhoutte Index)
    * Fisher ratio: $F(X) = tr\left\{\mathbf{S}_m \left(\mathbf{S}_w + \lambda I\right)^{-1}\right\}$ where $\mathbf{S}_b = \sum_i \pi_i \left(\mu_i - \mu_0\right)\left(\mu_i - \mu_0\right)^T$, $\mathbf{S}_m = E[(x - \mu_0)(x - \mu_0)^T]$