Summary of Paper: Fast Training of Recurrent Networks Based on EM Algorithm (1998)

Muthu Chidambaram

Department of Computer Science, University of Virginia

https://qdata.github.io/deep2Read/

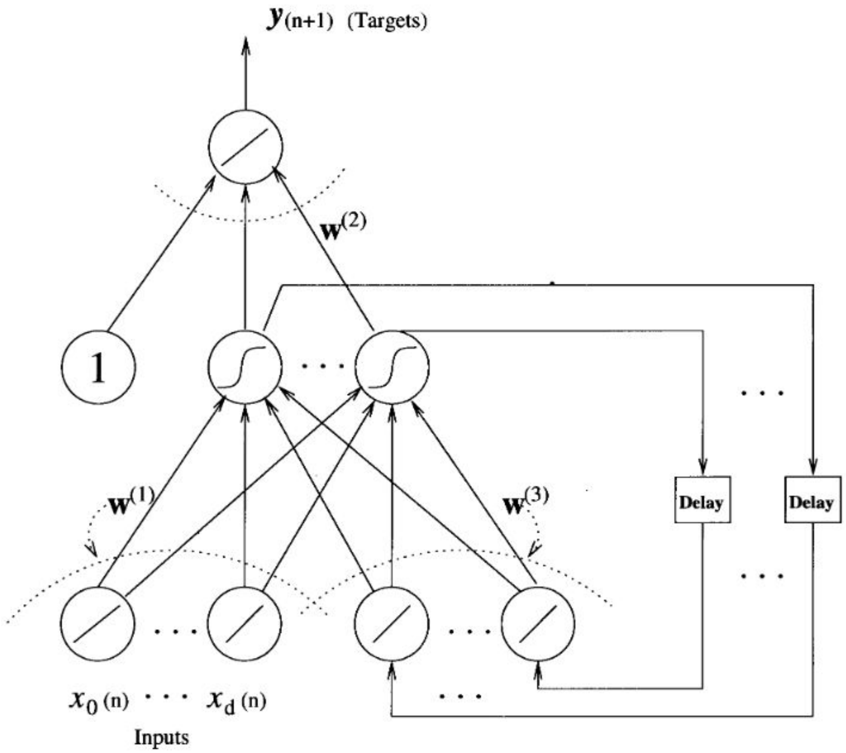
Fast Training of Recurrent Networks Based on EM Algorithm (1998)

- Authors: Sheng Ma, Chuanyi Ji
- Proposes internal-representation-based training of recurrent networks using EM
  - Prior work was based off of heuristics for internal targets

Fast Training of Recurrent Networks Based on EM Algorithm (1998)

- First establishes probabilistic models for targets of recurrent network hidden units
- Then uses EM + mean-field approximation to decompose training into a set of feedforward neurons
- Considers discrete-time recurrent networks with sigmoid activations

Fast Training of Recurrent Networks Based on EM Algorithm (1998)



Fast Training of Recurrent Networks Based on EM Algorithm (1998)

- Overview of EM: Introduce hidden variables with missing data (Dmiss), original data (Di), complete data (Dc)
  - E step: $Q(\Theta|\Theta^p) = \boldsymbol{E}\{\ln P(D_c|\Theta)|D_I,\Theta^p\} = \int P(D_{\text{mis}}|D_I,\Theta^p)\ln P(D_c|\Theta)\,dD_{\text{mis}}$
  - M step: $\Theta^{p+1} = \arg\max_{\Theta} Q(\Theta|\Theta^p).$
- Choose hidden random vars in EM to be the desired hidden targets
  - Markov property for modeling recurrent networks probabilistically

$$Q(\Theta|\Theta^p) = \ln P(\{\vec{x}\}|\Theta) + \int P(\{\vec{z}\}|\{y\},\{\vec{x}\},\Theta^p) \cdot \ln P(\{y\},\{\vec{z}\}|\{\vec{x}\},\Theta)\,d\{\vec{z}\}.$$

$$P(\{\vec{z}\}|\{t\},\{\vec{x}\},\Theta) = \prod^{N} P(\vec{z}(n+1)|\vec{z}(n),y(n+1),\vec{x}(n),\Theta)$$

$$P(\{t\},\{\vec{z}\}|\{\vec{x}\},\Theta) = \prod_{n=1}^{N} P(y(n+1),\vec{z}(n)f|\vec{z}(n),\vec{x}(n),\Theta).$$

Fast Training of Recurrent Networks Based on EM Algorithm (1998)

- Uses Gaussian to model conditional probability distributions due to correspondence with quadratic errors

$$P(\vec{z}(n+1)|\vec{z}(n), \vec{x}(n), \Theta) = B_1 \exp(-\lambda_1 E_1(n+1))$$

$$P(y(n+1)|\vec{z}(n+1), \Theta) = B_2 \exp(-\lambda_2 E_2(n+1))$$

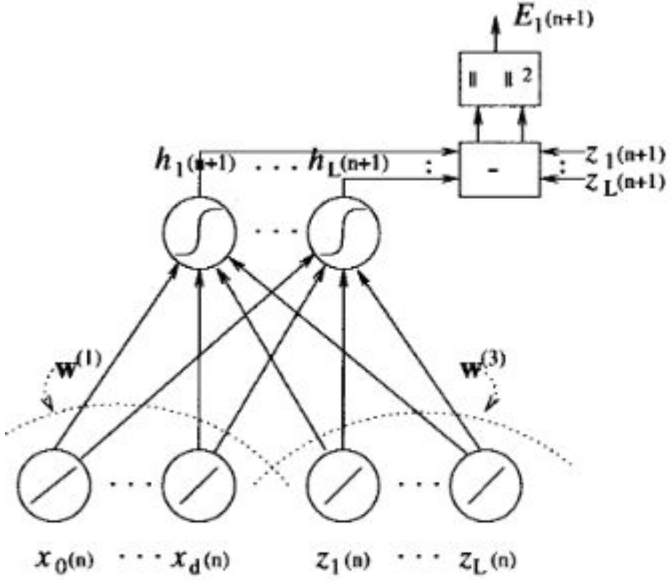
$$E_1(n+1) = ||\vec{z}(n+1) - \vec{h}(n+1)|^2$$

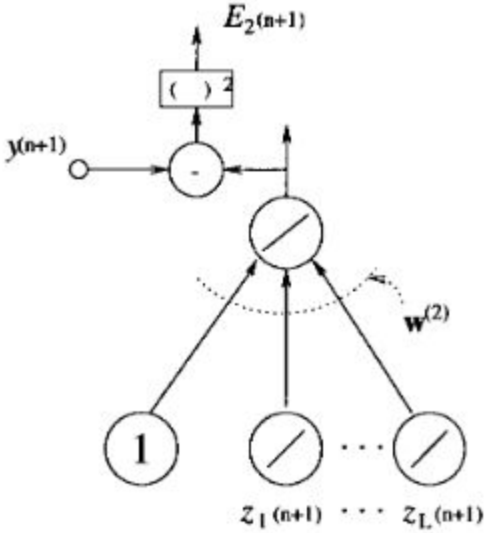
$$E_2(n+1) = (y(n+1) - \vec{z}(n+1)^T \cdot \vec{w}^{(2)})^2$$

$$h_j(n+1) = g(\vec{x}(n)^T \cdot \vec{w}_j^{(1)} + \vec{z}(n)^T \cdot \vec{w}_j^{(3)}).$$

$$P(y(n+1)|\vec{z}(n+1), \vec{x}(n), \Theta) = P(y(n+1)|\vec{z}(n+1), \Theta)$$

Fast Training of Recurrent Networks Based on EM Algorithm (1998)





Fast Training of Recurrent Networks Based on EM Algorithm (1998)

$$P(y(n+1), \vec{z}(n+1) | \vec{z}(n), \vec{x}(n), \Theta)$$
$$= P(y(n+1) | \vec{z}(n+1), \vec{z}(n), \vec{x}(n), \Theta) P(\vec{z}(n+1) | \vec{z}(n)$$
$$\vec{x}(n), \Theta)$$
$$= P(y(n+1) | \vec{z}(n+1), \Theta) P(\vec{z}(n+1) | \vec{z}(n), \vec{x}(n), \Theta)$$
$$= A_{yz} \exp\left(-\lambda_1 E_1(n+1) - \lambda_2 E_2(n+1)\right) \qquad (16)$$

$$P(y(n+1) | \vec{z}(n), \vec{x}(n), \Theta) = A_y \exp\left(-\lambda_3 E_3(n+1)\right)$$

$$E_3(n+1) = (y(n+1) - \vec{h}(n+1)^T \cdot \vec{w}^{(2)})^2.$$

$$P(\{\vec{z}\} | \{t\}, \{\vec{x}\}, \Theta)$$
$$= \prod_{n=1}^{N} A_z \exp\left(-\tfrac{1}{2} (\vec{z}(n+1) - \hat{\vec{z}}(n+1))^T \cdot \Sigma^{-1} (\vec{z}(n+1) - \hat{\vec{z}}(n+1))\right)$$

$$P(\vec{z}(n+1) | y(n+1), \vec{z}(n), \vec{x}(n), \Theta)$$
$$= \frac{P(y(n+1), \vec{z}(n+1) | \vec{z}(n), \vec{x}(n), \Theta)}{P(y(n+1) | \vec{z}(n), \vec{x}(n), \Theta)}.$$

$$P(\vec{z}(n+1) | y(n+1), \vec{z}(n), \vec{x}(n), \Theta)$$
$$= A_z \exp\left(-\tfrac{1}{2} (\vec{z}(n+1) - \hat{\vec{z}}(n+1))^T \cdot \Sigma^{-1} (\vec{z}(n+1) - \hat{\vec{z}}(n+1))\right)$$

$$\hat{z}_j(n+1) = h_j(n+1) + \frac{\lambda_2 w_j^{(2)}}{||\lambda_1 + \lambda_2 ||\vec{w}^{(2)}||^2} e(n+1)$$

$$e(n+1) = y(n+1) - \vec{h}(n+1)^T \cdot \vec{w}^{(2)}$$

$$P(\{t\}, \{\vec{z}\} | \{\vec{x}\}, \Theta)$$
$$= \prod_{n=1}^{N} A_{yz} \exp\left(-\lambda_1 E_1(n+1) - \lambda_2 E_2(n+1)\right)$$

Fast Training of Recurrent Networks Based on EM Algorithm (1998)

- Can now apply established probabilistic models to EM
- Use self-consistent mean-field approximation to approximate EM integral $P(u_i|u_j) \approx P(u_i|\boldsymbol{E}u_j)$

$$Q(\Theta|\Theta^p) = \ln P(\{\vec{x}\}|\Theta) + \sum_{k=1}^{N} \int \prod_{n=1}^{N} \cdot P(\vec{z}(n+1)|y(n+1), \vec{z}(n), \vec{x}(n), \Theta^p) \cdot \ln P(y(k+1), \vec{z}(k+1)|\vec{z}(k), \vec{x}(k), \Theta) \cdot d\{\vec{z}\}.$$

$$\boldsymbol{E}u_i = \iint u_i P(u_i|u_j) P(u_j)\, du_i\, du_j \approx \int u_i P(u_i|\boldsymbol{E}u_j)\, du_i$$

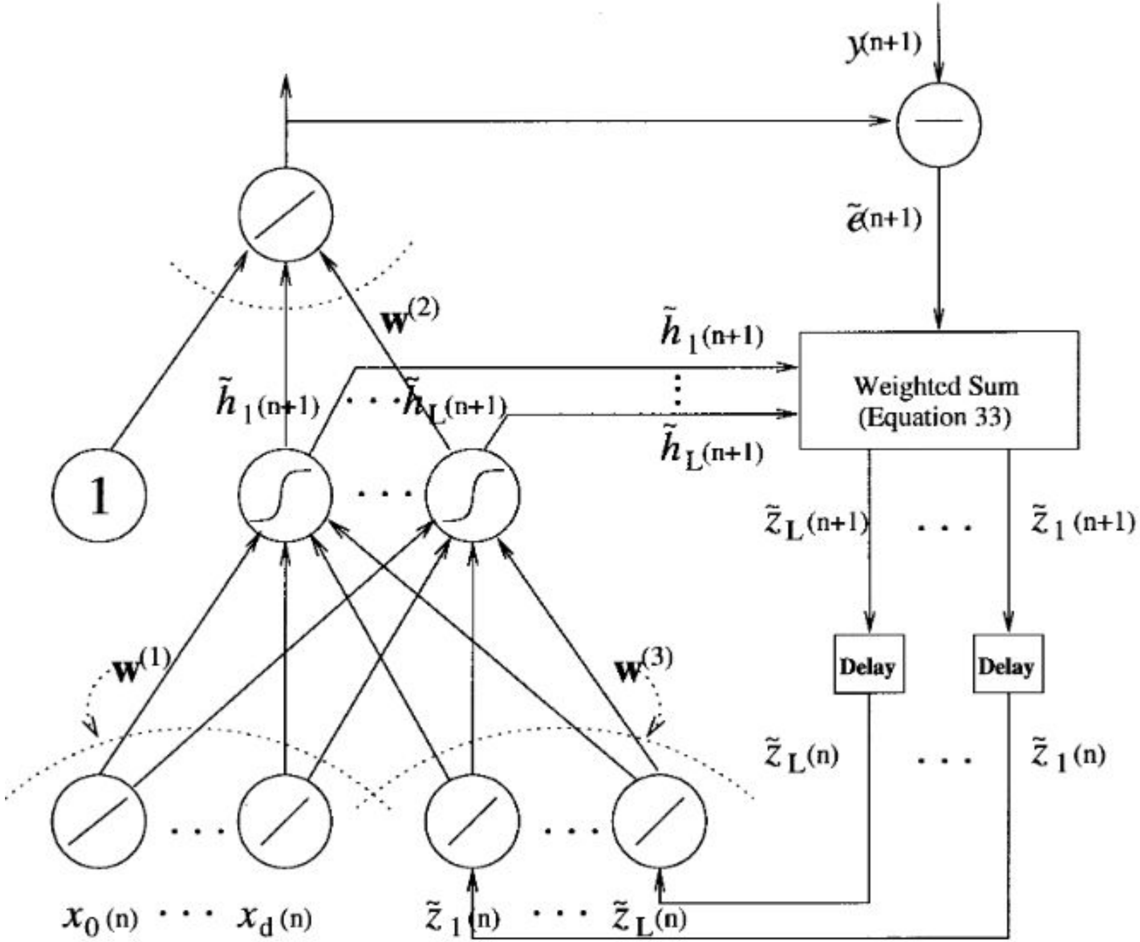
$$P(\vec{z}(n+1)|y(n+1), \vec{z}(n), \vec{x}(n), \Theta^p) \approx P(\vec{z}(n+1)|y(n+1), \tilde{\vec{z}}(n), \vec{x}(n), \Theta^p)$$

$$\tilde{h}_j(n+1) = g(\vec{x}(n)^T \cdot \vec{w}_j^{(1)} + \tilde{\vec{z}}(n)^T \cdot \vec{w}_j^{(3)})$$

$$\tilde{e}(n+1) = y(n+1) - \tilde{\vec{h}}(n+1)^T \cdot \vec{w}^{(2)^p}$$

$$\tilde{z}_j(n+1) \approx \tilde{h}_j(n+1) + \frac{\lambda_2 w_j^{(2)^p}}{\lambda_1 + \lambda_2 ||\vec{w}^{(2)^p}||} \tilde{e}(n+1)$$

Fast Training of Recurrent Networks Based on EM Algorithm (1998)



Fast Training of Recurrent Networks Based on EM Algorithm (1998)

- Mean-field approximation reduces maximization step to training a single sigmoidal neuron

$$Q(\Theta|\Theta^p) = \boldsymbol{E}_z\{\ln P(\{y\}, \{\vec{z}\}, \{\vec{x}\}|\Theta) || \{y\}, \{\vec{x}\}, \Theta^p\} \approx E_c - E_p - E_h - E_o$$

$$\vec{w}^{(2)^{p+1}} = \arg\min_{\vec{w}^{(2)}} (E_o + E_p) = \arg\min_{\vec{w}^{(2)}} \lambda_2 \sum_n (\vec{w}^{(2)^T} \tilde{\vec{z}}(n) - y(n))^2 + E_p$$

$$(\vec{w}_j^{(1)^{p+1}}, \vec{w}_j^{(3)^{p+1}}) = \arg\min_{w_j^{(1)}, w_j^{(3)}} \sum_n (\tilde{z}_j(n) - g(\vec{x}(n)^T \cdot \vec{w}_j^{(1)} + \tilde{\vec{z}}(n)^T \cdot \vec{w}_j^{(3)}))^2$$

$$E_p = \frac{\lambda_2 N ||\vec{w}^{(2)}||^2}{2(\lambda_1 + \lambda_2 ||\vec{w}^{(2)^p}||^2)} + \frac{\lambda_2 N (||\vec{w}^{(2)}||^2 ||\vec{w}^{(2)^p}||^2 - (\vec{w}^{(2)})^T (\vec{w}^{(2)^p})(\vec{w}^{(2)^p})^T \vec{w}^{(2)})}{2\lambda_1(\lambda_1 + \lambda_2 ||\vec{w}^{(2)^p}||^2}$$

Fast Training of Recurrent Networks Based on EM Algorithm (1998)

- Reduce nonlinear optimization of single neuron to linear using taylor series expansion

$$w_{\text{opt}} = \arg\min_{w} \sum_{n=1}^{N} (v(n) - g(w^T \cdot u(n)))^2$$

$$w_{\text{opt}} \approx \arg\min_{w} \sum_{n=1}^{N} c(n)^2 (w^T \cdot u(n) - g^{-1}(v(n)))^2$$

$$g(w^T \cdot u(n)) = v(n) + c(n)(w^T \cdot u(n) - g^{-1}(v(n))) + o(|w^T \cdot u(n) - g^{-1}(v(n))|)$$

$$w_{\text{opt}} = (U^T G^T G U)^{-1} U^T G^T V$$

Fast Training of Recurrent Networks Based on EM Algorithm (1998)

Randomly initialize $\vec{w}_j^{(1)}, \vec{w}_j^{(3)}$ and $w_j^{(2)}$ for $1 \leq j \leq L$.

E-step:

Compute the expectation of the desired hidden states $\tilde{\vec{z}}(n)$ recursively according to (**33**)–(**35**) (illustrated by **Fig. 3**).

M-step:

a) Compute $\vec{w}_j^{(1)}$ and $\vec{w}_j^{(3)}$ given by (**41**) through linear weighted regressions ((**47**)).

b) Compute $\vec{w}^{(2)}$ by solving (**42**).

Go back to the E-step until certain convergence criteria are satisfied.

Fast Training of Recurrent Networks Based on EM Algorithm (1998)

- Conclusion: Significant speed increase due to reducing training recurrent nets to training of individual neurons

References

- http://users.ece.gatech.edu/~jic/em-nn-98.pdf