Recurrent Neural Networks (RNNs)

- Hopfield Networks
- Simulated Annealing
- Boltzmann Machine
- Types of RNNs

Hopfield Networks (HNs)

- Associative Memory

. Store a set of p patterns

$\xi_i^\mu, \mu = 1, \cdots, p$ (pattern index), $i = 1, \cdots, n$ (unit index)

in such a way that when presented with a new pattern ζ_i, the network responds by producing whichever one of stored patterns most closely resembles ζ_i.

. The model:

The activation value of the ith unit:

$$S_i = sgn\left(\sum_{j=1}^{n} w_{ij} S_j - \theta_i\right),$$

where

$$sgn(x) = \begin{cases} +1 & \text{if } x \geqq 0 \\ -1 & otherwise \end{cases}$$

w_{ij}: the connection weight from the jth unit to the ith unit

θ_i: the bias term of the ith unit

In the Hopfield model,

$w_{ij} = w_{ji}$ and $w_{ii} = 0$

. Storing patterns

(1) stability condition for storing one pattern $\xi_i \in (-1, +1)$:

$$sgn\left(\sum_{j=1}^{n} w_{ij} \xi_j\right) = \xi_i \quad \forall i$$

If $w_{ij} \propto \xi_i \xi_j$ (Hebb's rule), the stability condition is satisfied since $\xi_j^2 = 1$. For convenience,

$$w_{ij} = \frac{1}{n} \xi_i \xi_j.$$

Then, the pattern ξ_i is referred to as an attractor and the pattern $-\xi_i$ (a reverse state) is also an attractor.

(2) storing many patterns:

Make w_{ij} as a superposition of terms; that is,

$$w_{ij} = \frac{1}{n}\sum_{\mu=1}^{p} \xi_i^\mu \xi_j^\mu,$$

where p represents the total number stored patterns.

The stability of a particular pattern ξ_i^ν:

$$sgn(h_i^\nu) = \xi_i^\nu \quad \forall i,$$

where

$$h_i^\nu = \sum_{j=1}^{n} w_{ij}\xi_j^\nu = \frac{1}{n}\sum_{j=1}^{n}\sum_{\mu=1}^{p} \xi_i^\mu \xi_j^\mu \xi_j^\nu$$

$$= \xi_i^\nu + \frac{1}{n}\sum_{j=1}^{n}\sum_{\mu\neq\nu}^{p} \xi_i^\mu \xi_j^\mu \xi_j^\nu \ (crosstalk\ term)$$

(3) storage capacity

Let $C_i^\nu = -\xi_i^\nu \cdot crosstalk\ term$

$$= -\xi_i^\nu \cdot \frac{1}{n}\sum_{j=1}^{n}\sum_{\mu\neq\nu}^{p} \xi_i^\mu \xi_j^\mu \xi_j^\nu$$

Then, if $C_i^\nu < 0$, the stability condition is satisfied

if $C_i^\nu > 1$, it changes the sign of h_i^ν and makes the ith bit of pattern ν unstable.

Assuming purely random patterns, with equal probability for $\xi_i^\mu = 1$ and $\xi_i^\mu = -1$, independently for each i and μ. Then, the probability P_{err} that any chosen bit is unstable is

$$P_{err} = P(C_i^\nu > 1).$$

Here, C_i^ν is $1/n$ times the sum of about np independent random numbers, each of which is +1 or −1. Then, for large np,

$C_i^\nu \sim N(0, \sigma^2)$, where $\sigma^2 = \frac{p}{n}$ by central limit theorem (CLT).

In this case,

$$P_{err} = \frac{1}{\sqrt{2\pi}\,\sigma} \int_1^\infty e^{-\frac{x^2}{2\sigma^2}} dx = \frac{1}{2}\left(1 - erf\left(\sqrt{\frac{n}{2p}}\right)\right),$$

where

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-u^2} du.$$

Let $P_{err} < 0.01$. Then, $p_{\max} \approx 0.15n$ (maximum storage capacity).

Here, $p_{\max}$ is an upper bound since it only tells the initial stability.

- Energy Function

. Let us define the energy function H as

$$H = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} S_i S_j.$$

Then, it always decreases (or remains constant) as the system evolves according to its dynamical rule; that is, the attractors (or memorized patterns) are at the local minima of the energy function.

. Let S_i' be the new value of S_i for some particular unit i. Then,

$$S_i' = sgn\left(\sum_{j=1}^{n} w_{ij} S_j\right)$$

If $S_i' = S_i$, then $H' - H = 0$

If $S_i' = -S_i$, then

$$H' - H = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} S_i' S_j + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} S_i S_j$$

$$= S_i \sum_{j=1}^{n} w_{ij} S_j < 0$$

This implies that the energy decreases every time an S_i changes.

. Spurious states

(1) The reverse states are also minima and have the same energy as the original patterns.

(2) Stable mixture states corresponding to linear combination of an odd number of patterns are also stable states.

- Optimization problems

. The Hopfield network is expected to find a configuration which minimizes an energy function.

. The optimization problems are transformed into minimizing the energy function or cost (objective) function.

. Relaxation methods for optimization for the convergence to the global optimum:

(1) simulated annealing (Kirkpatric et al., 1983)

(2) mean field annealing (Soukoulis et al., 1983)

Simulated Annealing

. At each temperature T, the solid is allowed to reach the thermal equilibrium, characterized by a probability of being in a state with energy E given by the Boltzmann distribution:

$$P(E) = \frac{1}{Z(T)} e^{-E/(k_B T)},$$

where $Z(T)$ represents a normalization factor (or partition function), k_B represents a Boltzmann constant, and $e^{-E/(k_B T)}$ represents a Boltzmann factor.

. The Boltzmann distribution is used to simulate the evolution to the thermal equilibrium of a solid for the temperature T.

- Simulated Annealing Algorithm (Metropolis, 1953)

. A sequence of Metropolis algorithm is evaluated at a sequence of (decreasing) temperature T. Then, the probability of configuration j given the configuration i in the next sequence is given by

$$P(j|i) = \begin{cases} 1 & \text{if } \Delta C_{ij} \leqq 0 \\ e^{(-\Delta C_{ij}/C)} & otherwise \end{cases},$$

where $\Delta C_{ij} = C(j) - C(i)$ (difference of cost functions)

(Metropolis criterion) In the equilibrium state,

$$P(i) = \frac{1}{Q(C)} e^{(-C(i)/C)},$$

where $Q(C)$ represents a normalization constant and C is a control parameter (temperature).

Simulate Annealing Procedure

begin
  initialize parameters; $m = 0$; (scheduling index)
  repeat
    repeat
      perturb (config i -> config j, ΔC_{ij});
      if $\Delta C_{ij} \leqq 0$, then accept;
      else if $e^{-\Delta C_{ij}/C} > Random[0,1)$, then accept;
      if accept, then update (config j);
    until equilibrium;
    $C_{m+1} = f(C_m)$; $m = m+1$;
  until stop criterion = true (system is frozen);
end;

. This process can be described by inhomogeneous Markov chain. For the convergence to the global optimum, the following conditions (sufficient conditions) should be satisfied:

$\lim_{k \to \infty} C_k = 0$ and $C_k \geqq O\left(\frac{1}{\log k}\right)$

(Laarhoven and Aarts, "Simulated Annealing," 1987)

. For the fast simulated annealing, the Cauchy-Lorentz visiting distribution can be used:

$\frac{C_k}{C_k + (\Delta C_{ij})^2}$ is used instead of $e^{-\Delta C_{ij}/C_k}$,

where $C_k = T_0/k$.

(Szu and Hartley, Physics Letter A, 122:157, 1987)

- Boltzmann Annealing

. Optimization in non-convex cost functions

1. $g(\boldsymbol{x})$: probability density of state-space of D parameters $\boldsymbol{x} = \{x_1, x_2, \cdots, x_D\}$
2. $h(\boldsymbol{x})$: probability density for the acceptance of new cost function given the previous value
3. $T(k)$: schedule of annealing for the temperature T in annealing-time steps k

. The class of Gaussian-Markovian systems

$$g(\boldsymbol{x}) = \frac{1}{Z(T)} e^{(-E(\boldsymbol{x})/(k_B T))} = \frac{1}{(2\pi T)^{D/2}} e^{(-\Delta \boldsymbol{x}^2/(2T))},$$

where $\Delta x = x - x_0$ (the deviation of x from x_0)

. The acceptance probability: chances of obtaining a new state E_{k+1} relative to a previous state E_k

$$h(\boldsymbol{x}) = \frac{e^{(-E_{k+1}/T)}}{e^{(-E_{k+1}/T)} + e^{(-E_k/T)}} = \frac{1}{1 + e^{(\Delta E/T)}},$$

where $\Delta E = E_{k+1} - E_k$.

. Given $g(\boldsymbol{x})$, it is sufficient to obtain a global minimum of $E(\boldsymbol{x})$ if T is selected to be not faster than $T(k) = T_0/\log k$.

$$g_k = \frac{1}{Z(T)} e^{(-E/(k_B T(k)))}$$

(H. Szu and R. Hartley, 1987)

. In order to statistically assure, any point in x space can be sampled infinitely often in annealing time:

$$\prod_{k=k_0}^{\infty}(1-g_k)=0 \text{ and } \sum_{k=k_0}^{\infty} g_k=\infty.$$

Let $E=k_BT_0$. Then,

$$\sum_{k=k_0}^{\infty} g_k=\frac{1}{Z(T)}\sum_{k=k_0}^{\infty} e^{(-\log k)}=\frac{1}{Z(T)}\sum_{k=k_0}^{\infty}\frac{1}{k}=\infty.$$

- Fast Annealing

. The Cauchy distribution:

$$g_k(x)=\frac{T(k)}{(\Delta x^2+T^2(k))^{(D+1)/2}},$$

where

$$T(k)=\frac{T_0}{k} \text{ and the usual value of } D=1.$$

. Statistical assurance for the convergence to the global optimum:

$$\prod_{k=k_0}^{\infty}(1-g_k)=0 \quad \text{and} \quad \sum_{k=k_0}^{\infty} g_k \approx \frac{T_0}{\Delta x^{D+1}}\sum_{k=k_0}^{\infty}\frac{1}{k}=\infty.$$

Boltzmann Machine (BM)

- The model

. There are visible (input and output) units and hidden units.

. All connections are symmetric; that is, $w_{ij} = w_{ji}$

. The units are stochastic:

$P(S_i = 1) = \frac{1}{1 + e^{-2\beta h_i}}$, where

$h_i = \sum_{j=1}^{n} w_{ij} S_j$ and $\beta = \frac{1}{T}$.

$P(S_i = -1) = 1 - P(S_i = 1)$.

That is, the stochastic version of Hopfield network.

. After equilibrium, the probability of finding the system in a particular state $\{S_i\}$ is determined by the Boltzmann-Gibbs distribution:

$P\{S_i\} = \frac{1}{Z} e^{-\beta H\{S_i\}}$, where

$H\{S_i\} = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} S_i S_j$ (energy function) and

$Z = \sum_{S_i} e^{-\beta H\{S_i\}}$ (partition function)

- Learning of Boltzmann machine

The probability of finding visible units in state α irrespective of state β in the freely running system is

$$P_\alpha = \sum_\beta P_{\alpha\beta} = \frac{1}{Z}\sum_\beta e^{-\beta H_{\alpha\beta}}, \text{ where}$$

$$Z = \sum_\alpha \sum_\beta e^{-\beta H_{\alpha\beta}}, \quad H_{\alpha\beta} = -\frac{1}{2}\sum_i \sum_j w_{ij} S_i^{\alpha\beta} S_j^{\alpha\beta}$$

For a set of desired probabilities R_α for these states, the relative entropy (Kullback-Leibler divergence) is defined by

$$E = D(R_\alpha \| P_\alpha) = \sum_\alpha R_\alpha \log \frac{R_\alpha}{P_\alpha}.$$

$E \geqq 0$ and $E = 0$ if and only if $P_\alpha = R_\alpha \quad \forall \alpha$.

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = \eta \sum_\alpha \frac{R_\alpha}{P_\alpha} \frac{\partial P_\alpha}{\partial w_{ij}}, \quad \frac{\partial P_\alpha}{\partial w_{ij}} = \beta\left(\sum_\beta S_i^{\alpha\beta} S_j^{\alpha\beta} P_{\alpha\beta} - P_\alpha \langle S_i S_j \rangle\right),$$

where

$$\langle S_i S_j \rangle = \frac{1}{Z}\sum_\lambda \sum_\mu e^{-\beta H_{\lambda\mu}} S_i^{\lambda\mu} S_j^{\lambda\mu}.$$

Therefore,

$$\begin{aligned}\Delta w_{ij} &= \eta\beta\left(\sum_\alpha \sum_\beta R_\alpha P_{\beta|\alpha} S_i^{\alpha\beta} S_j^{\alpha\beta} - \langle S_i S_j \rangle\right) \\ &= \eta\beta\left(\overline{\langle S_i S_j \rangle}_{clamped} - \langle S_i S_j \rangle_{free}\right)\end{aligned}$$

Here, $\overline{\langle S_i S_j \rangle}_{clamped}$ implies that the value of $\langle S_i S_j \rangle$ when the visible units are clamped in state α, averaged over α's according to their probabilities R_α.

- Operation of Boltzmann machine

Step 1. Wait until the equilibrium state for some temperature $T > 0$.

Step 2. Measure the correlation $\langle S_i S_j \rangle$ by taking a time average of $S_i S_j$.

Step 3. The visible units are clamped in each of α for which $R_\alpha > 0$.

Step 4. Wait until the equilibrium state.

Step 5. Measure the correlation $\overline{\langle S_i S_j \rangle}_{clamped}$.

Step 6. Network weights are updated by

$w_{ij}^{\text{new}} = w_{ij}^{\text{old}} + \Delta w_{ij}$, where $\Delta w_{ij} = \eta\beta\left(\overline{\langle S_i S_j \rangle}_{clamped} - \langle S_i S_j \rangle_{free}\right)$.

- Applications

. Pros: The BM learns probabilities rather than particular patterns.

. Cons: long training time, local minima problem.
-> One remedy is using the simulated annealing
(or fast simulated annealing).

. The BM can be applied to
various pattern classification problems,
learning vector quantization (LVQ), and
various combinatorial optimization problems.

Types of Recurrent Neural Networks (RNNs)

. A popular way to recognize (or reproduce) sequences has been to use partially recurrent networks.
. In most cases, the feedback connections are fixed, and updating is synchronous with one update for all units at each time step (sequential networks).
. There are context units that receive feedback signals.
. Examples:
(Jordan, 1986; Stornetta, 1988; Mozer, 1989; Elman, 1990)

- Backpropagation through time (BTT)

. Synchronous dynamics and discrete time:
the output of unit i at time $t+1$ is determined by

$$V_i(t+1) = f(h_i(t)) = f\left(\sum_{j=1}^{n} w_{ij} V_j(t) + \xi_i(t)\right),$$

where $\xi_i(t)$ represents the external input of the ith unit at time t.

. A sequence of maximum length T = A feedforward net with $T-1$ layers -> For the learning of w_{ij}s', the back-propagation algorithm in MLP can be applied.
For long sequences, the approach becomes impractical due to duplication of units.

. Learning without duplicating units

(real-time recurrent learning (RTRL), Williams and Zipser, 1989)

The output of unit i at time t:

$$V_i(t) = f(h_i(t-1)) = f\left(\sum_{j=1}^{n} w_{ij} V_j(t-1) + \xi_i(t-1)\right) \quad \text{... (1)}$$

The error measure on unit k at time t:

$$E_k(t) = \begin{cases} \zeta_k(t) - V_k(t) & \text{if } \zeta_k(t) \textit{ is the desired output at time } t \\ 0 & \textit{otherwise} \end{cases}$$

The total cost function:

$$E = \sum_{t=0}^{T} E(t), \text{ where } E(t) = \frac{1}{2}\sum_{k}\left(E_k(t)\right)^2$$

The gradient descent with respect to w_{pq} is determined by

$$\Delta w_{pq}(t) = -\eta \frac{\partial E(t)}{\partial w_{pq}} = \eta \sum_{k} E_k(t) \frac{\partial V_k(t)}{\partial w_{pq}} \quad \text{... (2)}$$

In the recurrent back-propagation algorithm,

$$\frac{\partial V_i(t)}{\partial w_{pq}} = f'(h_i(t-1))\left(\delta_{ip} V_q(t-1) + \sum_{j} w_{ij} \frac{\partial V_j(t-1)}{\partial w_{pq}}\right).$$

Learning Algorithm:

Step 1. At each time step, update V_i according to (1).

Step 2. Update w_{pq} as $w_{pq}(t) + \Delta w_{pq}(t)$ using (2).

. Memory requirement: for N fully connected units, N^3 derivatives to maintain and updating each takes a time proportional to N or N^4 in all at each time step.

. If the learning rate η is sufficiently small, the real-time learning is possible. However, it may have stability problem.

. Applications:
sequence learning tasks –
sequence recognition (eg. speech recognition),
sequence reproduction (eg. learning a set of songs),
sequence (or temporal) association, etc.

Conclusion

- The analysis of data distribution is required before selecting the machine learning model.
- The given task (or machine learning problem) can be classified according to the class of learning models.
- The pros and cons of machine learning models should be investigated for the given task.
- The optimization of the structure of machine learning model is necessary to achieve the improved and stable performances (generalization capacity).