Term project 2 guideline

TA Jaehoon Shin

No Cheating

- If you just copied the codes from Internet (include github) or other students' code, you will get **0 point and be noticed to department.**
  - We have cheating detection tool.
  - Actually, TA changed the main part of code, so finding source code or copying the previous year's code will not be helpful.

Stackoverflow is better than TA

- When you have problem with coding, googling will be more helpful than sending e-mails to TA.
- If you asking schedule or submission or at least algorithms, Tas can help. However, if you ask **coding**, Tas will likely to say that **'I don't know'**

Predictive coding model

- Prob1.py

- Things to do
  - 1. Fill the skeleton code!
    - Class NetworkForPredictiveCoding – def inference
    - Class NetworkForPredictiveCoding – def parameters_update
  - 2. Show the learning curve!
    - Using matplotlib.pyplot, plot the 'learning curve'
  - 3. Discussion!
    - What is the role and function of a single neuron in the predictive coding?
    - Why predictive coding is biologically more plausible?
    - What is happening in the inference step?
    - What is happening in the parameter update step?

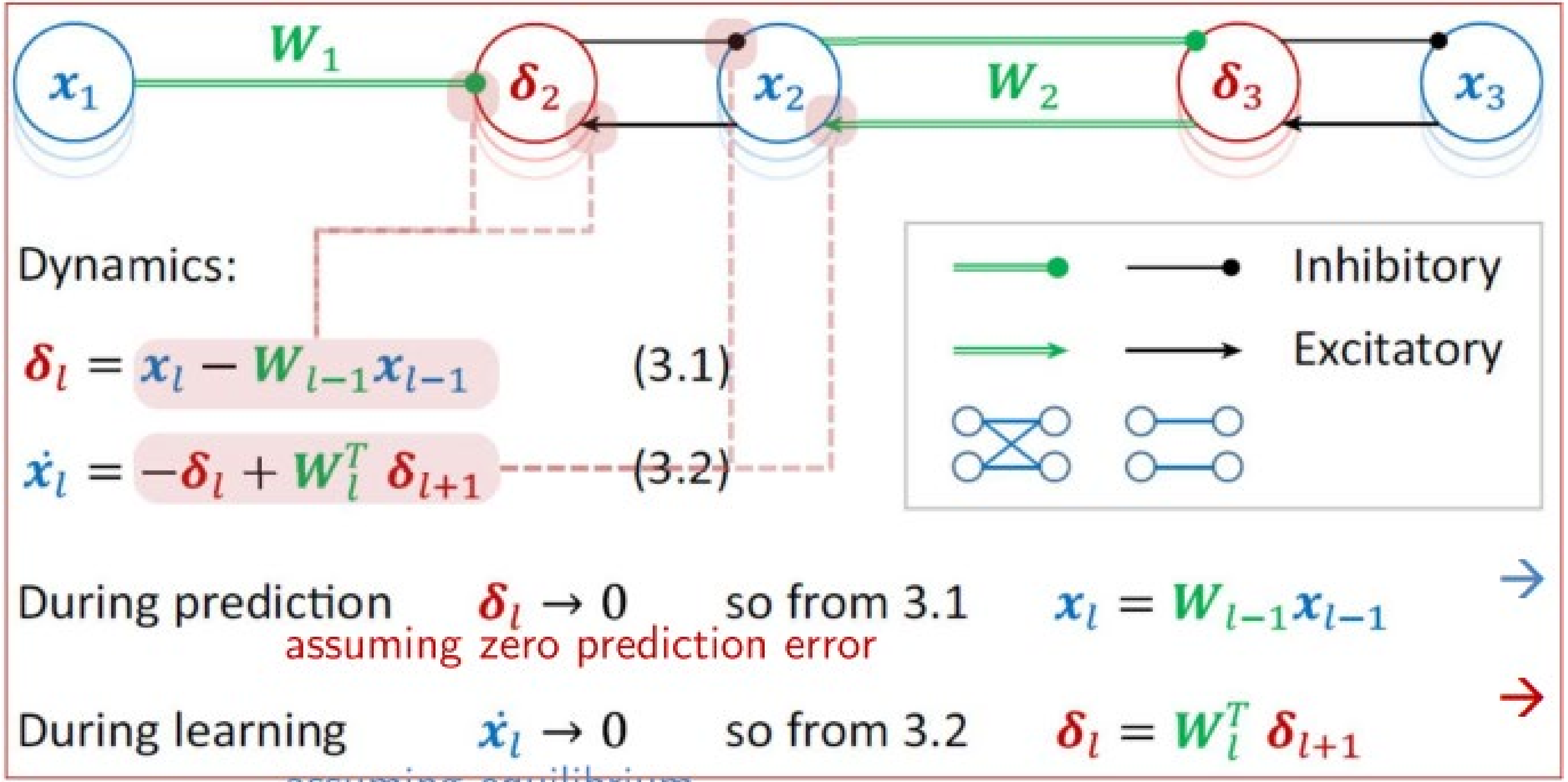
Predictive coding model

- Data : MNIST (http://yann.lecun.com/exdb/mnist/)

- The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

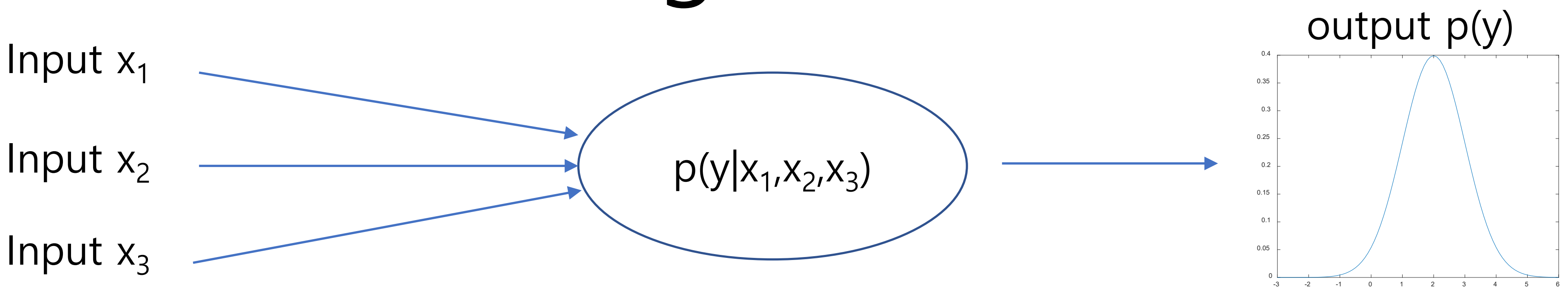
- Input : image / Output : Digit (0~9)

Revisiting lecture slide

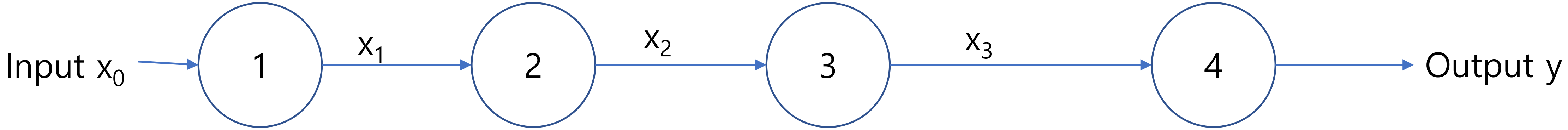


assuming equilibrium

Predictive coding model



- Neuron makes prediction 'y' from the input 'x_1, x_2, x_3', based on the $p(y|x_1,x_2,x_3)$ distribution.
- All neurons only know and control their own p(O|I) distribution.



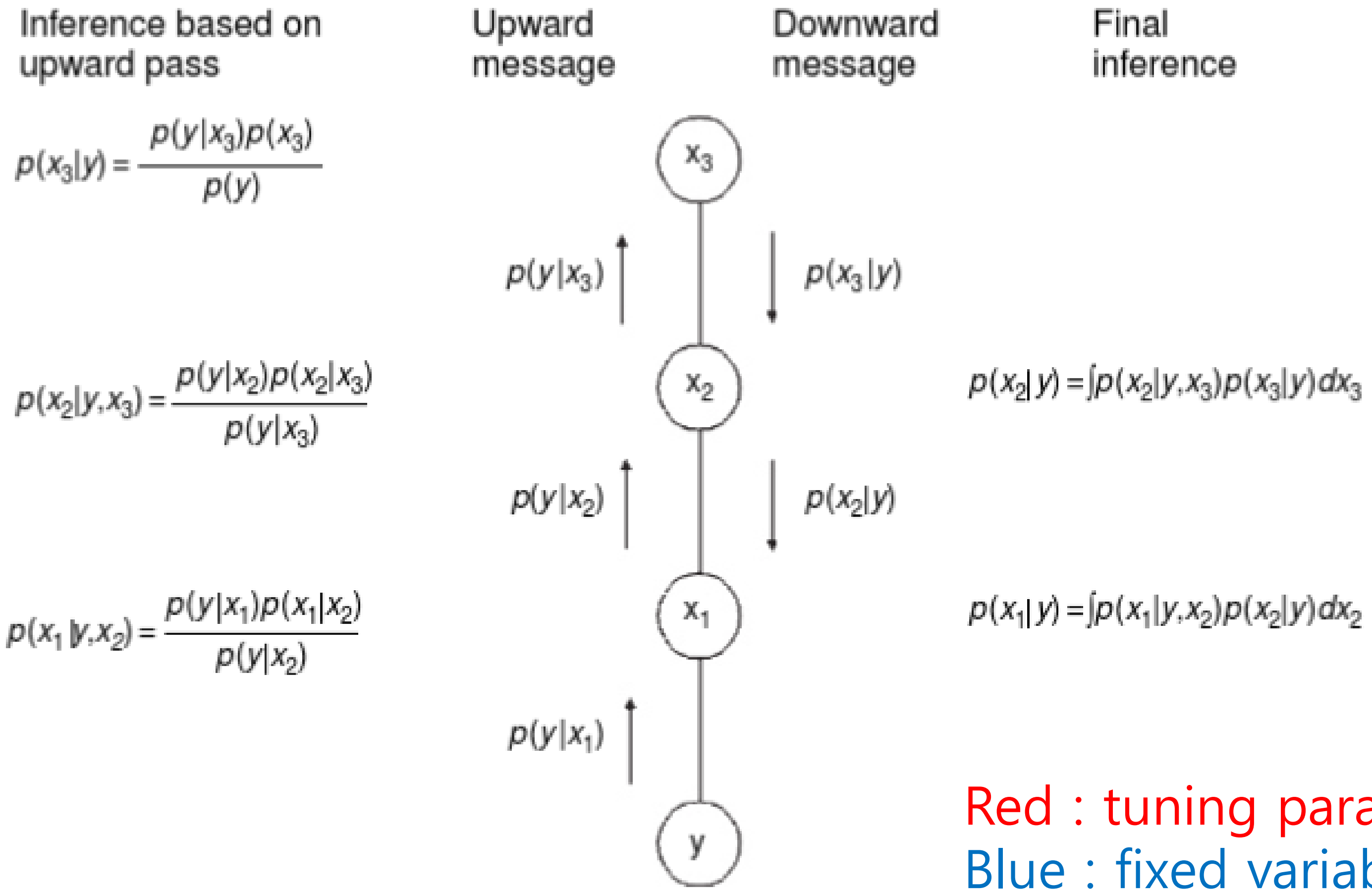
- What we want to maximize is $p(y_{correct})$.
- $p(y_{correct}) = \int p(y_{correct}|x_3) * p(x_3)dx_3 = \int p(y_{correct}|x_3) * \int p(x_3|x_2) * p(x_2)dx_2dx_3 = \ldots$
- Computationally Intractable

Predictive coding model

0-a. Initialized (randomized) neuronal weights ($p(x_i|x_{i+1})$)

0-b. Observation (Fixed input $p(x_3)$ and output (labels, $p(y)$))

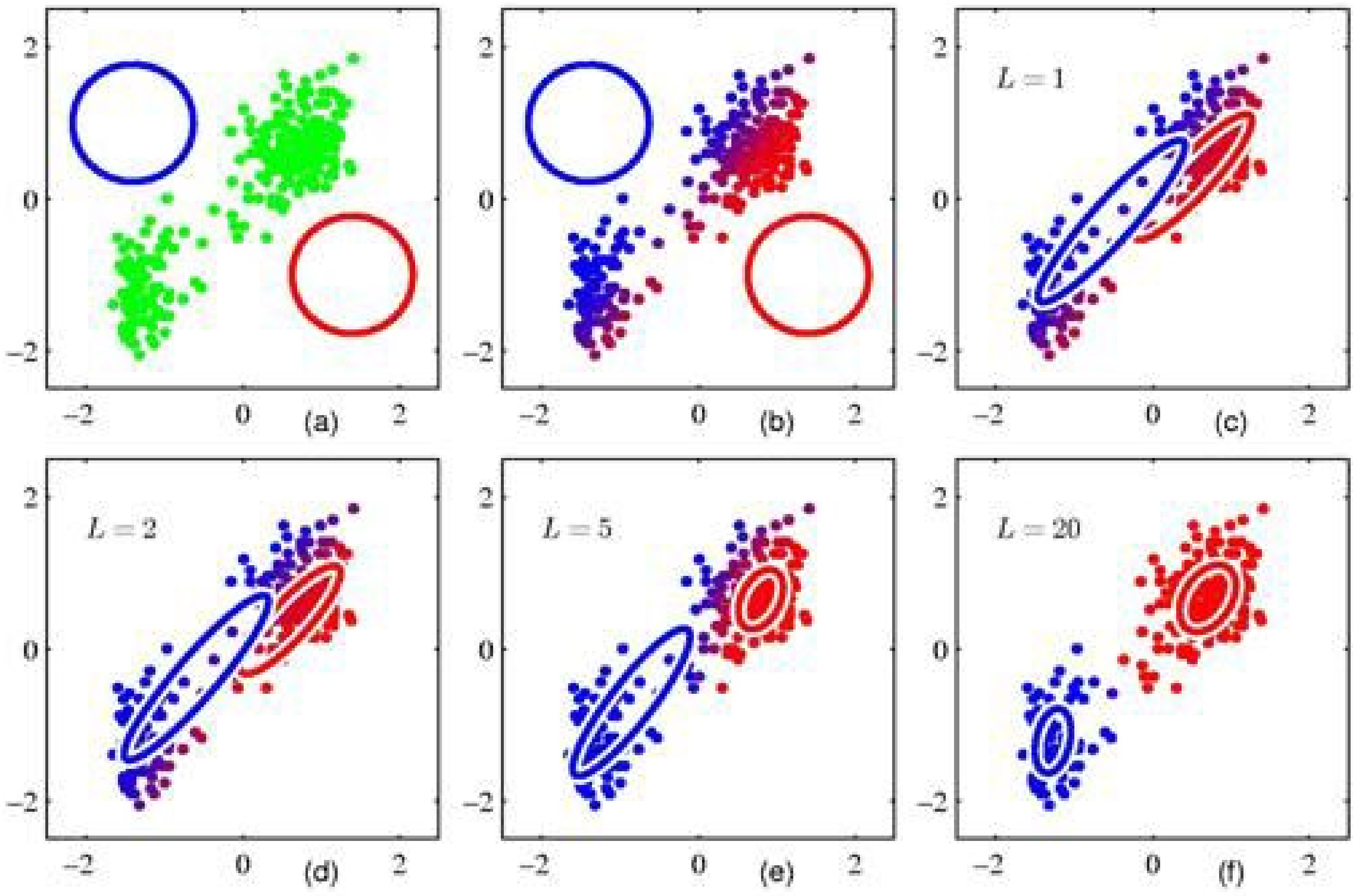
1. Estimate neuronal states ($p(x_i)$) that maximize the likelihood of observations with current neuronal weights ($p(x_i|x_{i+1})$)

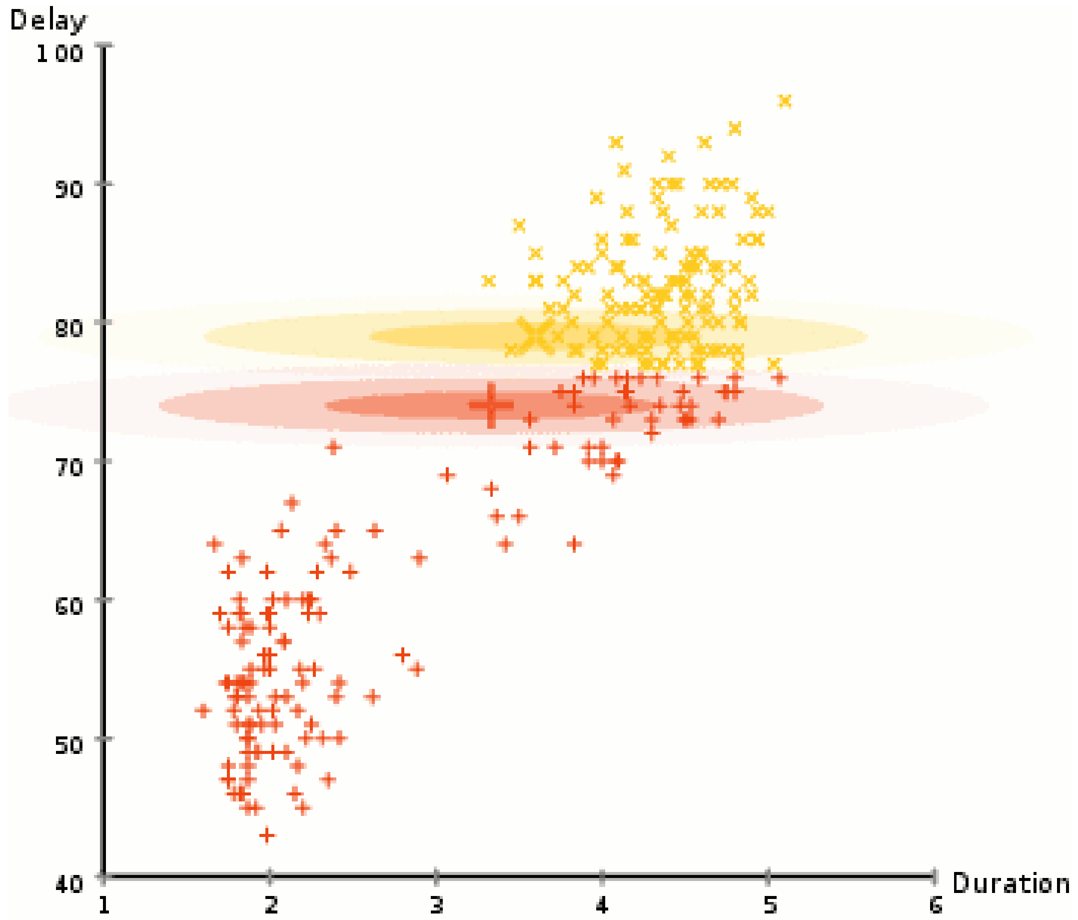


2. Estimate neuronal weights ($p(x_i|x_{i+1})$) that maximize the likelihood of observations with current neuronal states ($p(x_i)$)

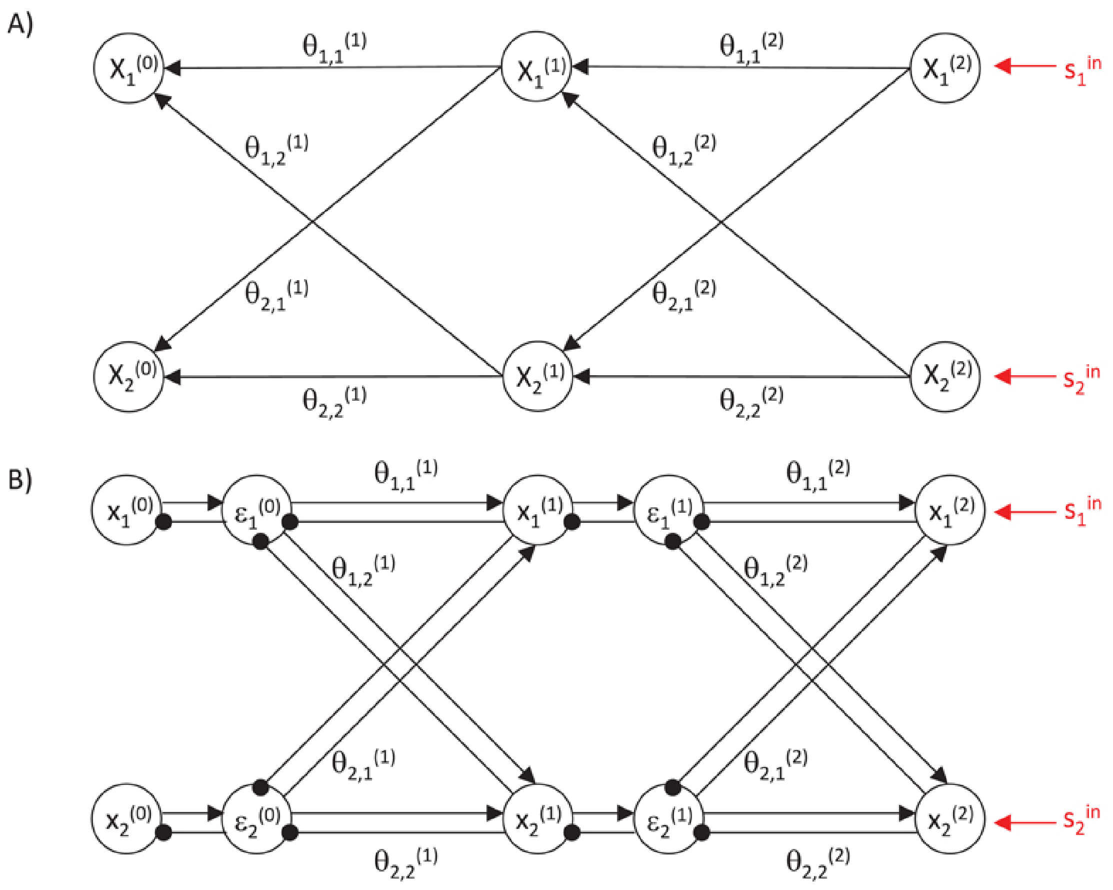
Red : tuning parameters in that step
Blue : fixed variables in that step

Cf) Expectation – maximization algorithm





Predictive Coding Model - Structure



Terms

$X^{(k)}_i$: neuronal activation of neuron i at layer k

$\theta^{(k)}_{i,j}$: synaptic weight between neuron i and j between layer k-1 and k

s^{in}_i : input at neuron i at input layer

l_{max} : last layer (input layer)

Layer order is reversed in the code!!!

Assumption

-The conditional probability of neuronal activation at the layer l follows the normal distribution:

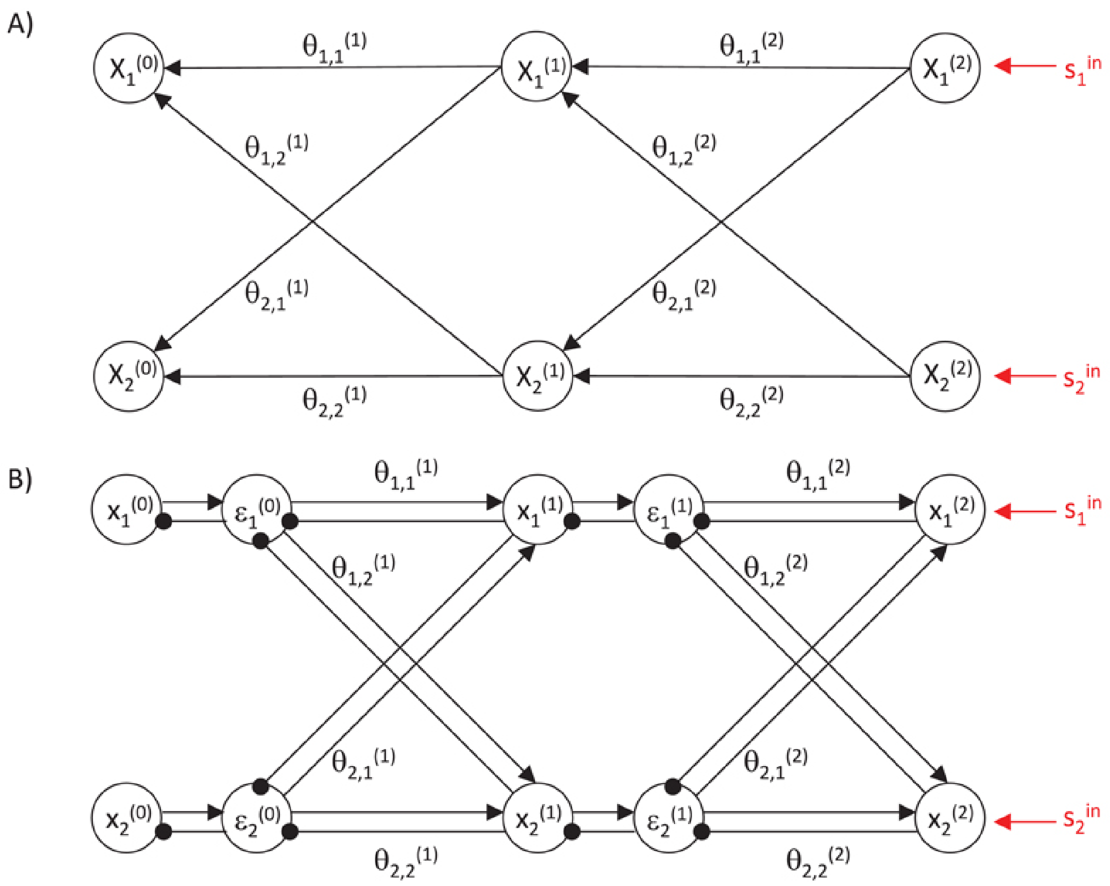
$$P\left(x_i^{(l)} | \bar{x}^{(l+1)}\right) = \mathcal{N}\left(x_i^{(l)}; \mu_i^{(l)}, \Sigma_i^{(l)}\right)$$

$$\mu_i^{(l)} = \sum_{j=1}^{n^{(l+1)}} \theta_{i,j}^{(l+1)} f\left(x_j^{(l+1)}\right) + b_i^{(l)}$$

Goal

-Find θ that maximizes $P(x|x^{lmax})$

Predictive Coding Model - Inference



Goal : maximize F

$$F = \ln\left(P\left(\bar{x}^{(0)}, \ldots, \bar{x}^{(l_{\max}-1)} \middle| \bar{x}^{(l_{\max})}\right)\right).$$

Feedforward NN

$$F = \sum_{l=0}^{l_{\max}-1} \ln\left(P\left(\bar{x}^{(l)} \middle| \bar{x}^{(l+1)}\right)\right)$$

Normal Distribution

$$F = \sum_{l=0}^{l_{\max}-1} \sum_{i=1}^{n^{(l)}} \left[\ln \frac{1}{\sqrt{2\pi \Sigma_i^{(l)}}} - \frac{\left(x_i^{(l)} - \mu_i^{(l)}\right)^2}{2\Sigma_i^{(l)}}\right]$$

Remove constant

$$F = -\frac{1}{2} \sum_{l=0}^{l_{\max}-1} \sum_{i=1}^{n^{(l)}} \frac{\left(x_i^{(l)} - \mu_i^{(l)}\right)^2}{\Sigma_i^{(l)}}$$

derivate

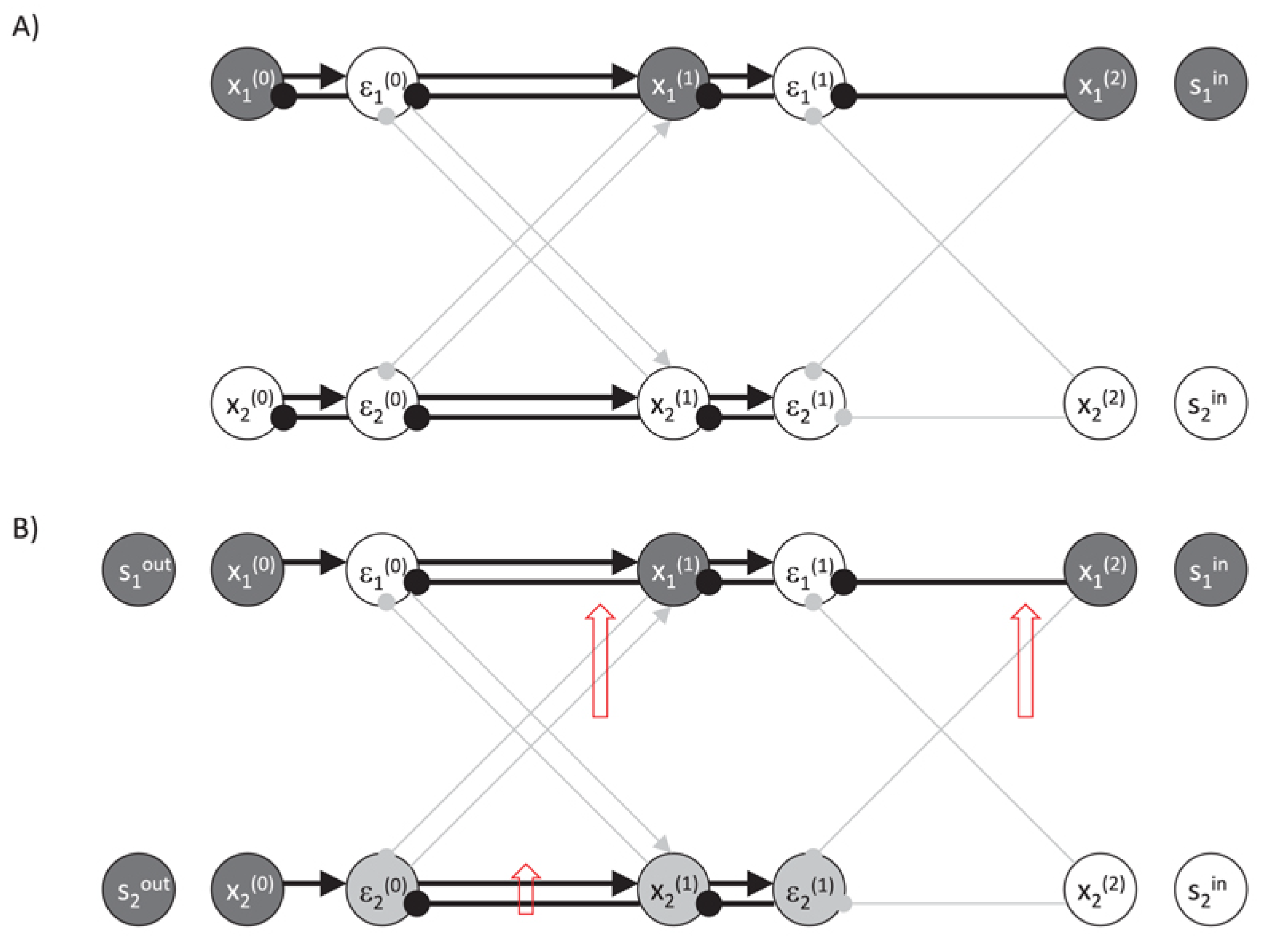
$$\frac{\partial F}{\partial x_b^{(a)}} = -\frac{x_b^{(a)} - \mu_b^{(a)}}{\Sigma_b^{(a)}} + \sum_{i=1}^{n^{(a-1)}} \frac{x_i^{(a-1)} - \mu_i^{(a-1)}}{\Sigma_i^{(a-1)}} \theta_{i,b}^{(a)} f'\left(x_b^{(a)}\right).$$

$$\dot{x}_b^{(a)} = -\varepsilon_b^{(a)} + \sum_{i=1}^{n^{(a-1)}} \varepsilon_i^{(a-1)} \theta_{i,b}^{(a)} f'\left(x_b^{(a)}\right)$$

$$\dot{\boldsymbol{x}}_l = -\boldsymbol{\delta}_l + \boldsymbol{W}_l^T \boldsymbol{\delta}_{l+1}$$

$$\varepsilon_i^{(l)} = \frac{x_i^{(l)} - \mu_i^{(l)}}{\Sigma_i^{(l)}}$$

Predictive Coding Model - Update



$$\frac{\partial F^*}{\partial \theta_{b,c}^{(a)}} = \varepsilon_b^{*(a-1)} f\left(x_c^{*(a)}\right)$$

F* : optimized F from infer

Neural weight gradient

During learning $\dot{x}_l \rightarrow 0$

for all Data **do**
- $\bar{x}^{(0)} \leftarrow \bar{s}^{out}$
- $\bar{x}^{(l_{max})} \leftarrow \bar{s}^{in}$
- **repeat**
  - Inference
- **until** convergence
- Update weights

… where's bias update? (see next page)

Predictive Coding Model - Update

- Main: find $\frac{\partial F^*}{\partial W} \& \frac{\partial F^*}{\partial b}$ $when\ W\ is\ neuronal\ weight\ (\theta)\ and\ b\ is\ neuronal\ bias$
- **F***(entropy) works as the likelihood of neuronal states, so W(weight) & b(bias) should maximize **F***
  => Find W & b such that $\frac{\partial F^*}{\partial W} = 0 \ \& \ \frac{\partial F^*}{\partial b} = 0$
  => $\frac{\partial F^*}{\partial W} \& \frac{\partial F^*}{\partial b}$ become gradients of W & b
- As previous slides, $F = -\frac{1}{2}\sum\sum\frac{\left(x_i^l - \mu_i^l\right)^2}{\Sigma_i^l} = -\frac{1}{2}\sum\sum\left(\varepsilon_i^l * \varepsilon_i^l * \Sigma_i^l\right)$

$$\therefore \ \frac{\partial F^*}{\partial w} = -\frac{1}{2}\sum\sum 2 * \varepsilon_j^l * \frac{\partial \varepsilon_j^l}{\partial W} * \Sigma_j^l = -\sum\sum \Sigma_j^l * \varepsilon_j^l * \frac{\partial \varepsilon_j^l}{\partial W}$$

similarly $\frac{\partial F^*}{\partial b} = -\sum\sum \Sigma_j^l * \varepsilon_j^l * \frac{\partial \varepsilon_j^l}{\partial b}$, since $\varepsilon_j^l = \frac{x_i^l - \mu_i^l}{\Sigma_i^l} = \frac{x_i^l - \sum W^l * f\left(x_j^{l+1}\right) - b^{l+1}}{\Sigma_i^l}$

$$\therefore \frac{\partial \varepsilon_i^k}{\partial W_{i,j}^l} = \frac{\partial}{\partial W_{i,j}^l}\left(\frac{x_i^k - \sum_{j=0}^{n(k)} W_{i,j}^{k+1} * f\left(x_j^{k+1}\right) - b_i^{k+1}}{\Sigma_i^l}\right)$$

$$= \begin{cases} when\ k \neq l-1, there\ is\ no\ W_{i,j}^l\ dependet\ term, so\ 0 \\ when\ k = l-1, \frac{\partial}{\partial W_{i,j}^l}\left(\frac{-\sum_{j=0}^{n(k)} W_{i,j}^{k+1} * f\left(x_j^{k+1}\right)}{\Sigma_i^l}\right) = -W_{i,j}^{k+1} * f\left(x_i^{k+1}\right) * \frac{1}{\Sigma_i^k} \end{cases}$$

similarly $\frac{\partial \varepsilon_i^k}{\partial b_i^l} = \begin{cases} when\ k \neq l-1,\ 0 \\ when\ k = l-1, \frac{\partial}{\partial b_i^l}\left(\frac{-b_i^{k+1}}{\Sigma_i^l}\right) = -\frac{1}{\Sigma_i^k} \end{cases}$

$$\therefore \frac{\partial F^*}{\partial w_{i,j}^l} = -\sum_{l=0}^{n} \sum_{i=1}^{n(l)} \Sigma_i^l * \varepsilon_i^l * \frac{\partial \varepsilon_i^k}{\partial W_{i,j}^l} = -\Sigma_i^{l-1} * \varepsilon_i^{l-1} * \left(-W_{i,j}^l * f\left(x_i^l\right) * \frac{1}{\Sigma_i^l}\right) = \varepsilon_i^{l-1} * W_{i,j}^l * f\left(x_i^l\right)$$

similarly $\frac{\partial F^*}{\partial b_i^l} = -\Sigma_i^{l-1} * \varepsilon_i^{l-1} * -\frac{1}{\Sigma_i^k} = \varepsilon_i^{l-1}$

Predictive Coding Model - Tips

- Functions.py
  - f(x, activation_function) : Output of neuron with 'neuronal state x' and 'activation function activation_function'
  - f_deriv(x,activation_function) : Derivate of output of neuron with 'neuronal state x' and 'activation function activation_function'
  - A@B : mat_mul(a,b) = matrix multiplication = element-wise multiplication
- In inference and parameters_update…
  - Neuronal_output_layer : Neuronal output from neurons = $f(x_i^{(l+1)})$
  - Neuronal_derivate_layer: Derivatives of neuronal output from neurons = $f'(x_i^{(l+1)})$
  - bias : Neuronal bias = $b_i^{(l)}$
  - self.variance : Neuronal states' standard deviation = $\Sigma_i^{(l)}$
  - error : Errors in the Inference = $\varepsilon_i^{(l)}$
  - current_entropy, previous_entropy : Entropy in current, previous iterations = F
  - weight_gradient, bias_gradient: Gradients of neuronal weights = $dF/d\theta$, dF/db
- When there are debug error related to dimension, please check 'batch size' dimension!

Additional tips

- Layer order is reversed in the code.
  - Slide 9~13 : input at layer l_{max}, output (label) at layer 0
  - Code : input at layer 0, output (label) at l_{max}

- Bias layer index is slightly different in the code
  - bias = self.bias[l - 1].repeat(1, size_of_batch)
  - Neuronal_state_array[l] = self.Weight[l - 1] @ F.f(Neuronal_state_array[l - 1], self.activation_function) + bias

Remember

- Deadline : **2023/06/18 23:59 pm**
- File name should be **2023xxxx_yourname_term3.zip**
- File should include…
  - 1 report **(2023xxxx_yourname_term2s.docs or pdf)** includes result figures and discussion of prob1
  - 2. complete prob1.py code
- Your skeleton codes should be able to run **only with files you submitted!**