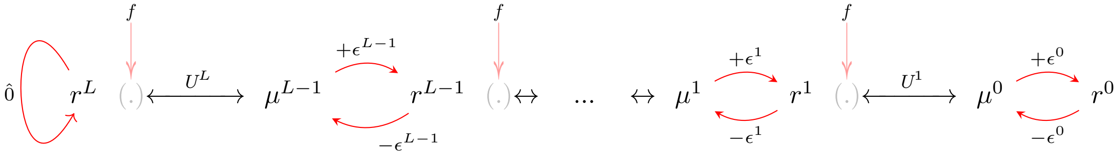
Predictive Coding (Reviews)

Contents

1 Predictive Coding

1.1 Unsupervised



$$\begin{aligned}
\text{Parameter: } \mathcal{U}_{Len=L} &= \left[\boldsymbol{U}^L_{(D^{L-1}\times D^L)}, \boldsymbol{U}^{L-1}_{(D^{L-2}\times D^{L-1})}, \dots, \boldsymbol{U}^2_{(D^1\times D^2)}, \boldsymbol{U}^1_{(D^0\times D^1)}\right] \\
\text{Parameter: } \mathcal{R}_{P\,Len=L} &= \left[\boldsymbol{r_p}^L_{(D^L\times 1)}, \quad \boldsymbol{r_p}^{L-1}_{(D^{L-1}\times 1)}, \quad \dots \quad, \boldsymbol{r_p}^2_{(D^2\times 1)}, \quad \boldsymbol{r_p}^1_{(D^1\times 1)}\right] \\
\mathcal{R}_P\left[1:\right] + \left[\boldsymbol{X}\right] \rightarrow \quad \mathcal{R}_{C\,Len=L} &= \left[\qquad\qquad \boldsymbol{r_c}^{L-1}_{(D^{L-1}\times 1)}, \quad \boldsymbol{r_c}^{L-2}_{(D^{L-2}\times 1)}, \quad \dots \quad, \boldsymbol{r_c}^1_{(D^1\times 1)}, \quad \boldsymbol{r_c}^0_{(D^0\times 1)} = \boldsymbol{X}\right] \\
\mathcal{U} \times f\left(\mathcal{R}_P\right) \rightarrow \quad \mathcal{M}_{Len=L} &= \left[\qquad\qquad \boldsymbol{\mu}^{L-1}_{(D^{L-1}\times 1)}, \quad \boldsymbol{\mu}^{L-2}_{(D^{L-2}\times 1)}, \quad \dots \quad, \boldsymbol{\mu}^1_{(D^1\times 1)}, \quad \boldsymbol{\mu}^0_{(D^0\times 1)}\right] \\
\mathcal{R}_C - \mathcal{M} \rightarrow \quad \mathcal{E}_{C\,Len=L} &= \left[\qquad\qquad \boldsymbol{\epsilon_c}^{L-1}_{(D^{L-1}\times 1)}, \quad \boldsymbol{\epsilon_c}^{L-2}_{(D^{L-2}\times 1)}, \quad \dots \quad, \boldsymbol{\epsilon_c}^1_{(D^1\times 1)}, \qquad \boldsymbol{\epsilon_c}^0_{(D^0\times 1)}\right] \\
\left[\vec{0}\right] + \mathcal{E}_C\left[:-1\right] \rightarrow \quad \mathcal{E}_{P\,Len=L} &= \left[\vec{0}_{(D^L\times 1)}, \quad \boldsymbol{\epsilon_p}^{L-1}_{(D^{L-1}\times 1)}, \quad \dots \quad , \quad \boldsymbol{\epsilon_p}^2_{(D^2\times 1)}, \quad \boldsymbol{\epsilon_p}^1_{(D^1\times 1)}\right]
\end{aligned}$$

$$\begin{aligned}
-\frac{\partial \boldsymbol{F}}{\partial \mathcal{R}_{p_i}} = -\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{r}_{p_i}} &= \boldsymbol{\epsilon}_{p_i} - \boldsymbol{\epsilon}_{c_i} \cdot \frac{\partial \boldsymbol{\mu}_i}{\partial \boldsymbol{r}_{p_i}} = \boldsymbol{\epsilon}_{p_i} - \left(\mathcal{U}_i \cdot \boldsymbol{\epsilon}_{c_i}\right) \odot f'^T\left(\mathcal{R}_{p_i}\right) \\
&= \epsilon_{p_i} - \epsilon_{c_i} \cdot \frac{\partial \boldsymbol{u}_i \cdot f(\boldsymbol{r}_{p_i})}{\partial \boldsymbol{r}_{p_i}} = \epsilon_{p_i} - \epsilon_{c_i} \cdot \left(\left[\boldsymbol{u}_i^T \cdot (I)\right] \odot f'^T\left(\boldsymbol{r}_{p_i}\right)\right) \\
-\frac{\partial \boldsymbol{F}}{\partial \mathcal{U}_i} = -\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}_i} &= \boldsymbol{\epsilon}_{c_i} \cdot \frac{\partial \boldsymbol{\mu}_i}{\partial \boldsymbol{u}_i} = \boldsymbol{\epsilon}_{c_i} \cdot f^T\left(\mathcal{R}_{p_i}\right) \\
&= \epsilon_{c_i} \cdot \frac{\partial \boldsymbol{u}_i \cdot f(\boldsymbol{r}_{p_i})}{\partial \boldsymbol{u}_i} = \epsilon_{c_i} \cdot \left((I) \cdot \boldsymbol{f}_i^T\right)
\end{aligned}$$

2 Predictive Coding Approximates Backprop Along Arbitrary Computation Graphs [1]

$$y = r^L \overset{U^L}{\longleftrightarrow} \underbrace{\mu^{L-1}, r^{L-1}}_{n^{L-1}} \leftrightarrow \text{...........} \leftrightarrow \underbrace{\mu^1, r^1}_{n^1} \overset{U^1}{\longleftrightarrow} \underbrace{\mu^0, r^0}_{n^0}$$

2.1 Literature

- **vs**– PC relies on local & Hebbian updates $\overset{\text{Vs}}{\longleftrightarrow}$ BP uses chain-rule
- **vs**– Neurons send unidirectional signals $\overset{\text{Vs}}{\longleftrightarrow}$ BP bi-directional.
- Neurons (N) have soma & axons. axons goes into synapses to its children (docks into somas like dendrites of other Ns), so, you cannot send gradient via feedback loop.
- In *Inference* step, PC tries to go in the error direction to predict better, also stay in its state which has predicted by its parents.
- If generative model assumes Gaussians distibution for data, uses Free Energy evaluation, & KL-divergence as optimizer, then:

$$\frac{dr_i}{\partial dt} = -\frac{\partial F}{\partial r_i} = \epsilon_i - \sum \epsilon_j \frac{\partial \mu_j}{\partial r_i} \qquad j \in C(v_i)$$
$$\frac{du}{dt} = -\frac{\partial F}{\partial u} = \epsilon_i \frac{\partial \mu_i}{\partial u_i}$$

- The slowness of the PC is due to its *Inference* process, which runs in an iterative manner.
- **Similarity**– Parameter-Linear $\star$ is common characteristics of PC & BP.

2.2 Contributions

- PC is approximated BP in its *Inference* step under certain assumptions.

2.3 Limits

-

2.4 Questions

- Why we do *Inference* on a wrong mapping? (model did not *Learn* yet)
- Why we do *Learn* on a wrong *hypothesis*? (model *Inference* is not correct)

2.5 Future Research

- Each neuron does an explicit task & elicits its co-workers to communicate their results.
- Neuron decides about its aftercoming weights? Like a federated system that has a center Neuron that gets the data and sends it to the next level of Ns. In this model, each N in every layer decides which N of its next level to hire and use for processing. For example, the centered N looks into a face and decides to send the pic (or cropped pic) to the next level Ns that associates with eye, noise, lip. Then, lip decides to use color, lines, etc. It can be top-down or bottom-up; like first color can be seen then lip, or first the connection between noise, eye, and lip will be measured, then face will be recognized. Or you can just identify the face by only the relative connection of eyes, lip, noise. Maybe, we need only the graph of ON neurons or synapses.
- PC is not parallel: first learn first layer, then the layer before, and so on.

3 Terms

- **Parameter-Linear**: Linear operation on weights followed by non-linearity on Ns in both PC & BP.
- Partial derivation with place holder:

$$\frac{\partial(\mathbf{A}.\mathbf{x})}{\partial \mathbf{x}} = \mathbf{A}^T \cdot (I) \tag{1}$$

$$\frac{\partial(\mathbf{A}.\mathbf{x})}{\partial \mathbf{A}} = (I) \cdot \mathbf{x}^T \tag{2}$$

- –

4 Guide

- **N**: Neuron

5 Cite

References

[1] Beren Millidge, Alexander Tschantz, and Christopher L Buckley. Predictive coding approximates backprop along arbitrary computation graphs. *Neural Computation*, 34(6):1329–1368, 2022.