Towards Hierarchical Explanation

Christiaan Hinrik Albert Anna

FACT-AI 2020

Table of Contents

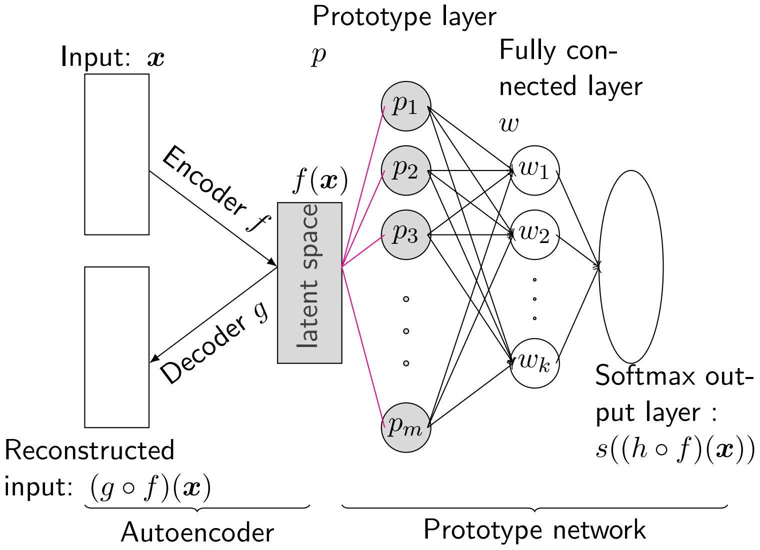
Original paper

Li, Oscar and Liu, Hao and Chen, Chaofan and Rudin, Cynthia.
Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions.
Thirty-Second AAAI Conference on Artificial Intelligence, 2018

Idea

- Broadly speaking, (convolutional) neural nets are not interpretable
- Instead of explaining predictions after training, integrate explanations in training goal
- Learn a fixed amount of prototypes which *represent the entire dataset*

The prototype network



Building the loss function

Loss function

Loss = Reconstruction error

$$L((f, g)), D) = R(g \circ f, D)$$

Building the loss function

Loss function

Loss = Crossentropy loss + Reconstruction error

$$L((f, g, h), D) = E(h \circ f, D) + R(g \circ f, D)$$

Building the loss function

Loss function

Loss = Crossentropy loss + Reconstruction error + Regularization terms

$$L((f,g,h),D) = E(h \circ f, D) + R(g \circ f, D) + R_1 + R_2$$

Building the loss function

Loss function

Loss = Crossentropy loss + Reconstruction error + Regularization terms

$$L((f,g,h),D) = E(h \circ f, D) + R(g \circ f, D) + R_1 + R_2$$

Regularization terms for prototypes $\boldsymbol{p_1}, \ldots, \boldsymbol{p_m}$

$$R_1(\boldsymbol{p_1}, \boldsymbol{p_2}, \ldots, \boldsymbol{p_m}, D) = \frac{1}{m} \sum_{j=1}^{m} \min_{i \in [1,n]} ||\boldsymbol{p}_j - f(\boldsymbol{x}_i)||_2^2$$

$$R_2(\boldsymbol{p_1}, \boldsymbol{p_2}, \ldots, \boldsymbol{p_m}, D) = \frac{1}{n} \sum_{i=1}^{n} \min_{j \in [1,m]} ||\boldsymbol{p}_j - f(\boldsymbol{x}_i)||_2^2$$

Building the loss function

Loss function

Loss = Crossentropy loss + Reconstruction error + Regularization terms

$$L((f,g,h),D) = \lambda_{\text{class}} E(h \circ f, D) + \lambda_R R(g \circ f, D) + \lambda_1 R_1 + \lambda_2 R_2$$

Regularization terms for prototypes $\boldsymbol{p_1}, \ldots, \boldsymbol{p_m}$

$$R_1(\boldsymbol{p_1}, \boldsymbol{p_2}, \ldots, \boldsymbol{p_m}, D) = \frac{1}{m} \sum_{j=1}^{m} \min_{i \in [1,n]} ||\boldsymbol{p}_j - f(\boldsymbol{x}_i)||_2^2$$

$$R_2(\boldsymbol{p_1}, \boldsymbol{p_2}, \ldots, \boldsymbol{p_m}, D) = \frac{1}{n} \sum_{i=1}^{n} \min_{j \in [1,m]} ||\boldsymbol{p}_j - f(\boldsymbol{x}_i)||_2^2$$

Reproducing results

- MNIST digits
- 15 prototypes

Reproducing results

- MNIST digits
- 15 prototypes
- Autoencoder with four convolutional layers

Reproducing results

- MNIST digits
- 15 prototypes
- Autoencoder with four convolutional layers
- Learning rate 0.0001, Epochs 1500

Reproducing results

- MNIST digits
- 15 prototypes
- Autoencoder with four convolutional layers
- Learning rate 0.0001, Epochs 1500
- Test accuracy: 98.879% (Paper reports 99.22%)

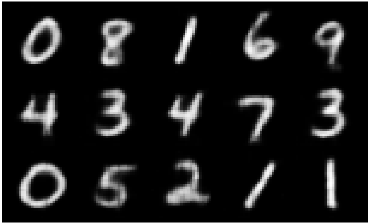
Learned prototypes

- Original results:

Learned prototypes

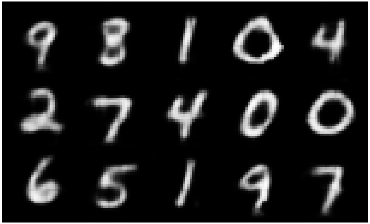
- Original results:

- Reproduced results:



However...

- Reproduced results with another seed:



- (Accuracy still 98.71%)

The hierarchical idea

- If $m > K$, multiple prototypes of 1 class
- Sometimes prototype network does not learn a prototype for each class
- If $m = K$, 1 prototype for each class
- Cannot capture intraclass differences

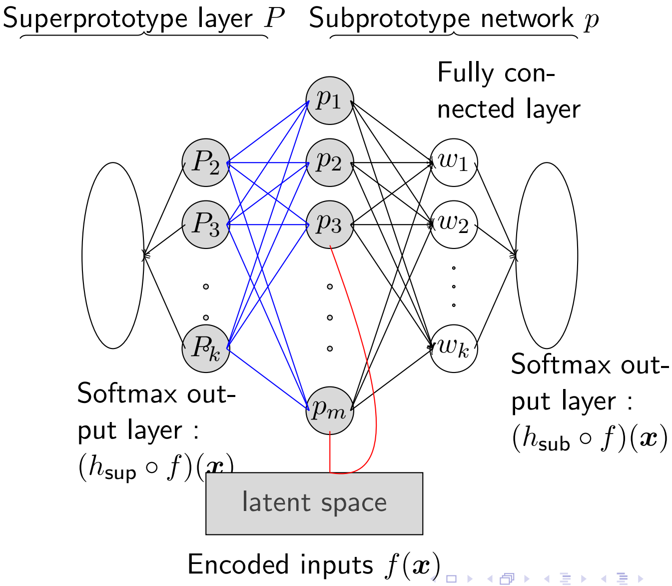
The hierarchical idea

- If $m > K$, multiple prototypes of 1 class
- Sometimes prototype network does not learn a prototype for each class
- If $m = K$, 1 prototype for each class
- Cannot capture intraclass differences

Possible solution: *superprototypes*

Input example $\prec$ Subprototype $\prec$ Superprototype,
where $\prec$ means "more specific than"

Our architecture



Two new loss terms

New loss term

$L((f,g,h),D) =$
$\lambda_{\text{class}} E(h \circ f, D) + \lambda_R R(g \circ f, D) + \lambda_1 R_1 + \lambda_2 R_2 + \lambda_3 R_3 + \lambda_4 R_4$

$$R_1(\boldsymbol{p_1}, \dots, \boldsymbol{p_m}, D) = \frac{1}{m} \sum_{j=1}^{m} \min_{i \in [1,n]} ||\boldsymbol{p}_j - f(\boldsymbol{x}_i)||_2^2$$

$$R_2(\boldsymbol{p_1}, \dots, \boldsymbol{p_m}, D) = \frac{1}{n} \sum_{i=1}^{n} \min_{j \in [1,m]} ||\boldsymbol{p}_j - f(\boldsymbol{x}_i)||_2^2$$

$$R_3(\boldsymbol{P_1}, \dots, \boldsymbol{P_K}, \boldsymbol{p_1}, \dots, \boldsymbol{p_m}) = \frac{1}{K} \sum_{k=1}^{K} \min_{j \in [1,m]} ||\boldsymbol{P}_k - \boldsymbol{p}_j||_2^2$$

$$R_4(\boldsymbol{P_1}, \dots, \boldsymbol{P_K}, \boldsymbol{p_1}, \dots, \boldsymbol{p_m}) = \frac{1}{m} \sum_{j=1}^{m} \min_{k \in [1,K]} ||\boldsymbol{P}_k - \boldsymbol{p}_j||_2^2$$

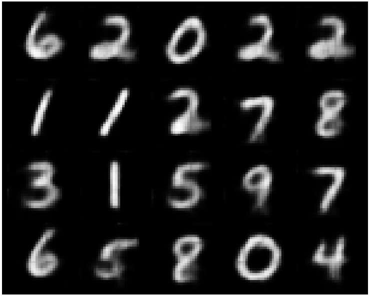
Results

- Accuracy for superprototype classifier: 98.86%
- Accuracy for subprototype classifier: 99.02%

Superprototypes and subprototypes

- K Superprototypes (fixed layer)

- m Subprototypes (learnable FC layer)



Transparency & Fairness

- Model interclass and intraclass variation
- Some hierarchical interpretability
- Possibly discover biases in dataset by looking at (sub)prototypes

Thank you for your attention

Thank you for your attention

Any questions?