Paper Presentation

Insights for incremental learning

Zheng Shou
03-04-2015
Overview


Previous approach


What is incremental learning for CNNs?

- We have a CNN trained for old classes
- A new class coming in with training data
- Goal: Grow CNN incrementally. Comparing with training all classes from scratch, we want to 1. speed up training procedure; 2. Improve accuracy on new class; 3. Keep performances on old classes
Previous approach

- **Approach 1: one-level expansion**

  - Flat expansion: add new classes nodes in last layer
  - Copy weights of L0 to L0’
  - Randomly initialize weights between new classes nodes and the second last layer
  - Use training data of new class to train L0’
Approach 2: two-level expansion

- Do one-level expansion first to obtain $L_0'$
- Partition old and new classes into several superclasses
- Do one-level expansion for each superclass, just in last layer only include old and new nodes in this superclass
- Training: train each networks separately
- Testing: decide superclass first and then its fine-label
How to leverage similarity of old class and new class?


Motivation: Humans learn much better when the examples are not randomly presented but organized in a order which first illustrates easy concepts, and gradually more complex ones.

Basic idea: construct training points easier to learn - starting from data that is easier to learning, and ending with the target training data distribution.
Curriculum Learning

- Learning deep architectures
  - Non-convex optimization problem
  - Many local minima
  - Target dataset - black curve
  - Easy training dataset - red curve, more smooth
  - Reach better local minima
Experiments on shape recognition: The task is to classify geometrical shapes into 3 classes (rectangle, ellipse, triangle)

3-hidden layers deep net

Use two stage curriculum:

Easy  Easy  Easy  Target  Target  Target
Curriculum Learning

- **Results:**
  - Vertical axis: error rate
  - Horizontal axis: switch epoch (#epochs training on easy)
  - Each box corresponds to 20 random initializations
  - As switch epoch increasing, the final accuracy improves
  - Imply that curriculum learning helps deep nets to reach better local minima
Insight for adjusting order of training points

- How to use curriculum to leverage similarity of old class and new class to help incremental learning?

- Model the similarity between new class and old similar classes as one part of the easiness

\( N \): #training points in each epoch

\( m \): the number of training epochs

\( S_i \): the similarity score vector. The old class A is the most similar class of new class B. When inputing data of B, \( O_i \) is the predicted score at node of A. In coarse network, set \( S_i = O_i \); in fine network, set \( S_i = 1 - O_i \).

\( D_i \): the distance to feature mean \( D_i = \| F_i - \bar{F} \| \)

\( R_i \): integer, with maximum value \( R \). Indicate the times of training point \( i \) existed in these \( m \) epochs.

\[ \min \sum_{i=1}^{N} \underbrace{(S_i + \lambda \cdot D_i) \times R_i}_{\text{constant}} \]

s.t. \( \sum_{i=1}^{N} R_i = m \cdot N, R_i \in Z^+, 1 \leq R_i \leq R \)
Insight for designing loss function

- Yi Sun, Yuheng Chen, Xiaogang Wang, Xiaoou Tang, Deep Learning Face Representation by Joint Identification - Verification, NIPS 2014

- Identification - Enlarge difference between different classes

- Verification - Decrease difference among same classes
Insight for designing loss function


- Identification loss - maximize difference between training data of different classes, basically using softmax loss

- Verification loss - minimize difference of extracted features at second last layer among training data of same classes
In incremental learning, when training coarse network and fine networks, also use

Identification loss + $\lambda \times$ Verification loss

- Identification loss - maximize difference between training data of different classes, basically using softmax loss
  → This is more important when training coarse network, set $\lambda$ a bit smaller

- Verification loss - minimize difference of extracted features at second last layer among training data of same classes
  → This is more important when training fine network, set $\lambda$ a bit larger
Thank you!