9. Low-shot learning with large-scale diffusion

논문 : https://arxiv.org/pdf/1706.02332v2.pdf

요약

- 고전 semi-supervised learning의 diffusion method를 활용해서 정확도를 향상할수있는지 실험
    - 컴퓨팅 환경이 좋아졌으니 연산량/메모리 한계를 극복할 수 있으니 여러 기법들을 시도한 듯
- 참고문헌들이 제안한 좋은 기법들을 짬뽕한 논문이라 참고문헌을 읽어야 이해가 되는 논문 (거의 survey 논문 수준)
    - **간단히 정리후 추가 정리 예정**

시작 전 용어 이해

Diffusion method (Diffusion Processes for Retrieval Revisited 참고)

- Query Sample을 기준으로 인접한 Data들에 동일한 label을 붙이면서 정확도를 향상시키는 기법
- N개의 데이터가 존재할때 N*N의 행렬을 만들고 KNN기법을 활용하여 데이터별 인접 데이터를 표기
    - 오늘날 딥러닝에서 N*N 행렬을 만든다는거는 너무 많은 메모리를 사용하는것이 문제
- 그림과 같이 Unlabeled data에 정확도를 향상시키는데 활용

Figure 1: Illustration of usefulness of diffusion processes for retrieval. Affinities before (left) and after (right) the diffusion are used to assign each element to the highlighted query samples according to their pairwise affinities. As can be seen, pairwise affinities alone are not sufficient to capture the intrinsic structure of the data manifold. Applying a diffusion process spreads affinities and allows to significantly improve retrieval performance (best viewed in color).

제안기법

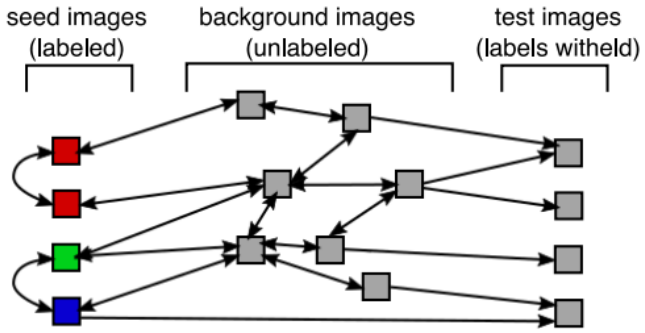

Figure 1: The diffusion setup. The arrows indicate the direction of diffusion. No diffusion is performed from the test, for the rest of the graph the edges are bidirectional (ie. the graph matrix is symmetric). Except mentioned otherwise, the edges have no weights.

- Few-shot Learning에 다른 논문들이 제안한 Semi-Supervised Learning 기법 끼얹기 (그림 1)
    - labeled data(seed data)로 학습
    - unlabeled data(background data)와 labeled data를 k-NN graph로 묶어서 diffusion method를 활용해서 labeling
    - data별로 bidirectional edge를 가짐
    - test image에서 difussion이 일어나지 않도록 단방향으로 구현 (label 정보를 받을수만 있도록 설정됨)
- 학습 방법
    - base domain
        - 전체 데이터셋 중 일부분만 labeling이 되있는 거대한 데이터셋
        - 상황 구현을 위해 논문은 ImageNet을 사용 (실험은 두가지 방법으로 진행)
            1. labeled data만 사용해서 fine-tune
            2. labeled data로 fine-tune 후 unlabeled data를 diffusion에 사용
    - novel domain
        - 새로 배우고 싶은 class로 구성된 데이터셋
            - labeled data는 few-shot learning처럼 굉장히 적은상태
            - unlabeled data는 상황에 따라 사용여부가 갈림
        - 여기서도 두가지 상황으로 분류
            1. base domain 1번과 pair
                - 학습된 모델로 unlabeled data/labeled data(train + test)의 feature representation 연산
                - unlabeled data는 background data로 사용
                - labeled data는 k-NN graph에 포함시키고 diffusion iteration 진행
            2. base domain 2번과 pair
                - unlabeled data는 사용안함
                - 나머지는 1번상황과 동일

기호	설명
N	데이터 개수로 labeled data와 unlabeled data가 섞여있음 $N = n_{\mathrm{L}} + n_{\mathrm{B}}$
C	diffuse할 class의 개수 = novel class의 개수
$\mathbf{L}$	Label Matrix $N \times C$ i번째 행의 정보는 i번째 데이터(이미지)의 클래스 확률값이 저장됨 초기에는 labeled data는 one-hot vector로, unlabeled data는 모든 클래스에 대해 0 값을 가짐
$\mathbf{W}$	Affinity matrix $N \times N$ 이지만 다른 논문의 기법들로 빠르게 계산 가능 초기에 인접 포인트는 1, 아니면 0 값을 가짐 (자기 자신으로 diffusion못하게 설정)

Propagating label

- unlabeled data에 label을 propagate하는 방법으로 k-NN graph를 만들어서 labeled data의 label 정보를 diffusion하는 방법을 사용함
- feature representation 간 거리 값으로 adjacency matrix를 만들어야 하나, 데이터 N개에 대한 adjacency matrix의 크기는 NxN으로 너무 큼
    - Billion-scale similarity search with GPUs 논문과 같이 GPU를 활용한 병렬처리로 k-NN graph 계산 시간을 단축
- 일반적인 diffusion 방식은 iteration에 따라 L 값이 점점 커짐

$$\mathbf{L}_{t+1} = \mathbf{W}\mathbf{L}_t$$

- 식 1) diffusion에 Normailization 적용

$$\mathbf{L}_{t+1} = \eta(\mathbf{W}\mathbf{L}_t). \qquad (1)$$

- η : Normalize function
  - Multiclass assumption 에서는?
    - L1-normalize를 행별로 적용함
  - 클래스별 데이터 수가 불균등하다면?
    - 열별로 L1-normalize를 적용함
  - Multiclass에서 데이터 수가 불균등하면?
    - 둘다 적용함~ (식 2, 3)

$$\mathbf{L}\mathbf{1}_{\mathrm{C}} = \mathbf{1}_{\mathrm{N}} \qquad \mathbf{1}_{\mathrm{N}}^{\top}\mathbf{L} \propto \mathbf{p}_{\mathcal{C}} \tag{2}$$

$$\mathbf{L} \leftarrow \mathbf{L}\,\mathrm{diag}(\mathbf{L}\mathbf{1}_{\mathrm{C}})^{-1}\mathrm{diag}(\mathbf{p}_{\mathcal{C}}) \qquad \mathbf{L} \leftarrow \mathrm{diag}(\mathbf{L}\mathbf{1}_{\mathrm{N}})^{-1}\mathbf{L} \tag{3}$$

- 식 4) Affinity matrix 업데이트
  - An efficient algorithm for large-scale detection of protein families 에서의 Markov Clustering 기법으로 업데이트

$$\mathbf{W}'_t \leftarrow \mathbf{W}_t \cdot \mathbf{W}_t \qquad \mathbf{W}_{t+1} \leftarrow \Gamma_r(\mathbf{W}'_t), \tag{4}$$

$$(\Gamma_r M)_{pq} = (M_{pq})^r \Big/ \sum_{i=1}^{k} (M_{iq})^r$$

실험

- 기존 기법들을 적용해가면서 정확도 확인
  - 표 1
    - ImageNet으로 학습 후 YFCC100M 데이터셋에 대한 few-shot learning 진행
      - none : background image를 사용하지 않았을 때 결과
      - F1M : YFCC100M에서 1M (10의 6제곱)개를 background image로 사용해서 diffusion한 결과
      - imnet : 학습에 사용하지않았던 ImageNet의 unseen label image를 background image로 사용(label 없이)해서 diffusion했을 때의 결과
    - edge weighting / normalize operator를 변경해가면서 정확도 비교
    - 논문에서 이것저것 사용해봤으나 눈에띄는 성능변화는 없음
    - 결론
      - edge weighting은 안하는게 낫다
      - normalization은 안쓰는것보단 쓰는게 낫다
  - 그림 2
    - 좌측 그림) diffusion step에 따른 정확도 변화 관찰
    - 우측 그림) novel class에 속하는 데이터를 background image를 변화시켰을때 / k-NN의 k를 변화시켰을 때의 정확도 관찰
    - 결론
      - diffusion step은 적절한 값을 찾아야함
      - unlabeled data는 많을 수록 좋다
      - 보편적으로 k값을 늘릴수록 성능향상이 보인다

background	none	F1M	imnet
edge weighting			
constant	62.7±0.68	65.4±0.55	73.3±0.72
Gaussian weighting*	62.7±0.66	65.4±0.58	73.6±0.71
meaningful neighbors*	62.7±0.68	40.0±0.20	73.6±0.62
η operator			
none	40.6±0.18	41.1±0.10	42.3±0.19
Sinkhorn	61.1±0.69	56.8±0.50	72.3±0.72
column-wise	62.7±0.68	65.4±0.55	73.3±0.72
non-linear transform* Γ_r	62.7±0.68	65.4±0.55	73.3±0.72
class frequency prior*	62.7±0.66	65.4±0.60	73.3±0.65

Table 1: Variations on weighting for edges and normalization steps on iterates of $\mathbf{L}$. The tests are performed for $n = 2$ and $k = 30$, with 5 runs on the group 1 validation images. Variants that require a parameter (*e.g.*, the σ of the Gaussian weighting) are indicated with a "*". In this case we report only the best result, see Appendix B material for full results. In the rest of the paper, we use the variants indicated **in bold**, since they are simple and do not add any parameter.

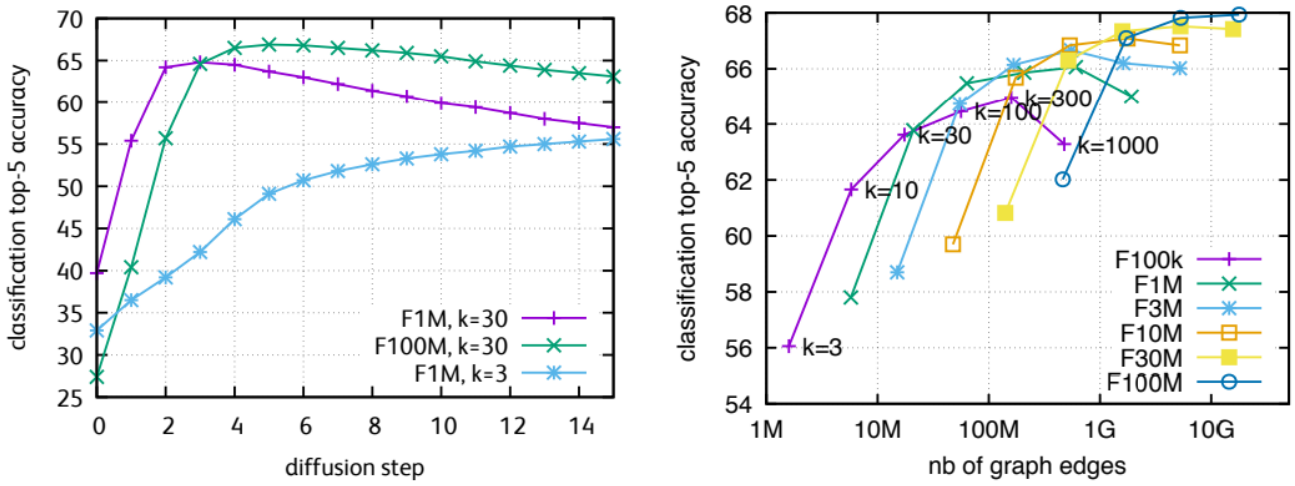

Figure 2: Classification performance with $n = 2$ — Left: accuracy as a function of the iteration — Right: various settings of k and n_{B}, ordered by total number of edges (average of 5 test runs, with cross-validated number of iterations). Appendix E material provides a complementary analysis.

- 다른 few-shot learning 기법과 성능 비교
  - 표 2
    - 제안기법과 8. Low-shot Visual Recognition by Shrinking and Hallucinating Features ([16]으로 표기)의 성능 비교
    - out-of-domain diffusion
      - 표 1처럼 YFCC100M 데이터셋에서 test에 안쓰는 데이터를 일부 떼서 background data로 사용
    - in-domain Imagenet
      - 표 1처럼 학습에 사용하지 않은 Imagenet 데이터를 background data로 사용
    - logistic regression
      - 그냥 transfer learning한 것
      - 특이한점은 실험에 사용한 pre-train ResNet50이 ImageNet으로 이미 학습이 됨
      - 개인적으로 Seen label들이니 다른 기법보다 정확도가 높게 나오는게 당연하다고 생각됨
    - combined
      - logistic regression 모델과 diffusion classifier의 결과를 weighted average했을 때 결과
      - 혼자 앙상블을 썼기때문에 best일수밖에 없음
    - [16]
      - 모델 구조는 4. Matching Networks for One Shot Learning를 사용

	out-of-domain diffusion				in-domain	logistic	combined		[16]
n	none	F1M	F10M	F100M	Imagenet	regression	+F10M	+ F100M	
1	57.1±0.53	60.0±0.52	61.4±0.68	62.3±0.59	68.0±0.64	57.3±0.51	62.0±0.75	62.6±0.63	60.6
2	62.5±0.50	65.5±0.50	66.8±0.44	67.8±0.62	73.2±0.51	66.0±0.59	68.7±0.43	69.2±0.60	68.9
5	68.4±0.38	70.6±0.31	71.9±0.48	73.1±0.61	77.8±0.35	76.4±0.26	76.9±0.23	77.4±0.27	77.3
10	72.7±0.16	74.2±0.30	75.3±0.05	76.2±0.15	80.1±0.14	80.9±0.21	81.3±0.17	81.5±0.18	80.6
20	76.0±0.28	77.0±0.21	77.5±0.17	78.6±0.15	81.4±0.10	83.7±0.15	83.9±0.15	84.1±0.09	82.5

Table 2: Comparison of classifiers for different values of n, with $k = 30$ for the diffusion results. For the out-of-domain setup, the "none" column indicates that the diffusion solely relies on the labelled images. The in-domain diffusion corresponds to the column "imagenet". The results of most-right column [16] are state-of-the-art on this benchmark to our knowledge, generally outperforming the results of matching networks and model regression [38, 39] in this setting.