Dynamic Routing Between Capsules (CapsNet)

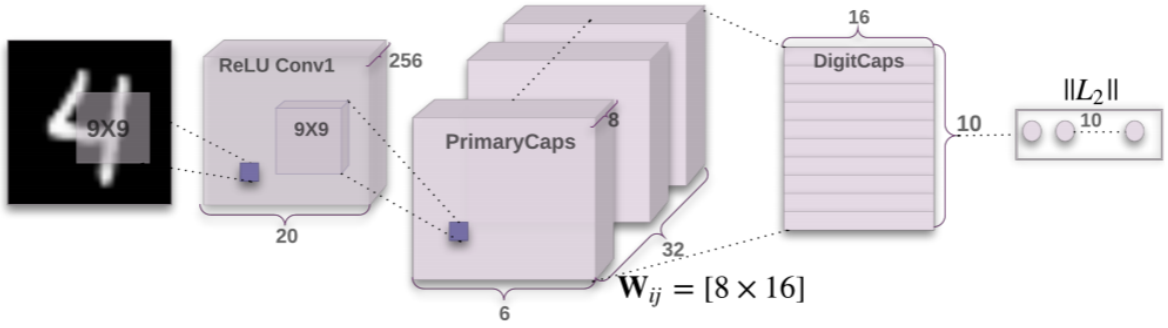
Capsules

Entity와 Property of entity로 나누어져 있을 때, 캡슐에 들어있는 약 알갱이들이 property, 그리고 그 properties를 한 번에 담고 있는 것을 캡슐이라고 비유한 것.

장점은 객체의 공간적인 관계를 고려하여 올바르게 매칭 된 경우에만 맞다고 판단할 수 있다.

또한 Max-Pooling같이 큰 특징만 잡는데 있어 애매할 수 있는 단점을 dynamic routing을 통해 보안할 수 있다. 물론 올바른 연결점을 찾는 알고리즘을 적용하는데 있어 변수가 있기 때문에 이 점을 보안해줘야 한다.

CapsNet 구조



시작은 CNN과 똑같이 필터를 가진 커널을 통해 피쳐맵 생성. 그 다음 PrimaryCaps를 convolution filter와 reshape을 통해 캡슐 한 개당 8개의 property를 가지는 피쳐맵으로 바꿔준다.

예를 들어 28x28x1의 입력 값을 9x9짜리 256개의 필터를 가진 커널을 통해 20x20짜리 256개의 피쳐맵을 생성. 이렇게 생성된 20x20짜리 256개 피쳐맵에 9x9짜리 32x8개의 피쳐맵을 만드는 convolution filter 적용 => PrimaryCaps. 이렇게 만들어진 벡터를 reshape 시켜서 캡슐 하나당 8개의 property를 가지게 한다 (6x6x(32x8) -> (6x6x32x)8 형태). 이러면 1152개의 캡슐에 각각 8개의 property를 가지게 된다. 이렇게 만들어진 캡슐들을 dynamic routing을 통해 이어준다.

Dynamic routing

Procedure 1 Routing algorithm.

1: **procedure** ROUTING($\hat{\boldsymbol{u}}_{j|i}, r, l$)
2: for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow 0$.
3: **for** r iterations **do**
4: for all capsule i in layer l: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$ ▷ `softmax` computes Eq. 3
5: for all capsule j in layer $(l+1)$: $\mathbf{s}_j \leftarrow \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}$
6: for all capsule j in layer $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$ ▷ `squash` computes Eq. 1
7: for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i}.\mathbf{v}_j$
return $\mathbf{v}_j$

총 두개의 캡슐 레이어 (PrimaryCaps, DigitCaps)을 각각 layer l, layer l+1로 지정해주고, 서로 이어주는 b_ij를 0으로 지정해주며 생성한다. 그 다음 b_ij를 SoftMax를 적용해 coupling coefficient (c_ij)로 만든다. SoftMax하면 c_ij의 합이 1이 되게 만들면서 캡슐-i와 캡슐-j의 연결강도가 된다.

그 이후 8x16 크기의 가중치 행렬 W_ij를 PrimaryCaps에 곱해서 property를 8개에서 16개로 바꿔준다. 이렇게 바뀌진 값을 c_ij와 곱해서 길이가 16인 PrimaryCaps를 DigitCaps에 얼마나 보낼지 정해주면 같은 DigitCaps로 연결된 캡슐 벡터들을 더해서 s_j를 구할 수 있다.

마지막으로 이 연결된 캡슐 벡터들의 합인 s_j를 squash function을 통해 0이나 1에 한없이 가깝게 만들어준다. Squash 해준 v_j 캡슐은 non-linearity를 추가할 수 있고 entity의 존재 확률을 0과 1사이의 값으로 표현할 수 있다. 이렇게 DigitCaps가 완료되면 b_ij를 PrimaryCaps와 DigitCaps의 곱으로 업데이트 해주면 routing iteration이 한번 끝나게 된다. 이 과정을 반복하면 dynamic routing이 된다.